

# Power BI from Rookie to Rock Star

## Book 3: Power Query and Data Transformation in Power BI

**Author: Reza Rad**

**Edition: 7, January 2019**



**RADACAD**

PUBLISHED BY  
RADACAD Systems Limited  
<http://radacad.com>

24 Riverhaven Drive,  
Wade Heads,  
Whangaparaoa 0932  
New Zealand

Copyright © 2019 by RADACAD

All rights reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

**Cover:** Virgin Silberhorn Ebnefluh, Pixabay

# About the New Edition and New Structure

The Power BI from Rookie to Rock Star been such a popular book from the time that it published, and I added content to it every single week. After edition 3 which released July 2017, there have been many contents added. The edition 3 itself was more than 1100 pages, and If I wanted to continue the book as an all-in-one, it would have been more than 2000 pages now. So I decided to break the book into a book series. Each book in this series is a complete book and can be read individually. However, each book covers a specific area of the Power BI, and if you want to learn Power BI from ground zero to sky hero, you would need to read them all. Here is the new structure:

- Book 1: Power BI Essentials
- Book 2: Visualization with Power BI
- **Book 3: Power Query and Data Transformation in Power BI**
- Book 4: Power BI Data Modelling and DAX
- Book 5: Pro Power BI Architecture

This book is the book three of the series. You will learn all about Power Query in this book. Through many demos and examples, you will learn all aspects of Power Query, from getting Data to different types of transformations. You will also learn about features such as Custom functions that can make your code more automated. You will learn about the scripting language of Power Query which is called M. at the end of the book, you will learn some use cases of using Power Query; for example, for creating a Date dimension. If you want to learn more about other parts of Power BI not just Power Query, your answer is within books 1 to 5.

# About the book; Quick Intro from Author

In July 2015, after the first release of Power BI Desktop, I had been encouraged to publish a Power BI online book through a set of blog posts. The main reason to publish this book online was that with the fast pace of updates for Power BI Desktop, it is impossible to publish a paperback book because it will be outdated in a few months. From that time till now, I've been writing blog posts (or sections) of this book almost weekly in RADACAD blog. So far, I have more than 60 sections wrote for this book. The book covers all aspects of Power BI; from data preparation to modeling, and visualization. From novice to the professional level, that's why I called it Power BI from Rookie to Rock Star.

You can start reading this book with no prerequisite. Each section can be read by itself; normally you don't need to follow a specific order. However, there are some sections, that need an example previously built in another section. These sections have a prerequisite section mentioning this requirement.

After a year and half of writing online, I decided to release this book as a PDF version as well, for two reasons; First to help community members who are more comfortable with PDF books, or printed version of materials. Second; as a giveaway in my Power BI training courses. Feel free to print this book and keep it in your library, and enjoy. This book is FREE!

This book will be updated with newer editions (hopefully every month), so you can download the latest version of it anytime from my blog post here:

<http://www.radacad.com/online-book-power-bi-from-rookie-to-rockstar>

Because I've been writing these chapters and sections from mid-2015, there are some topics or images or sections outdated with new changes in Power BI. I will do my best to update any changes in the next few editions. However, to keep you informed; There is a date at the beginning of each section under the header that mentioned the publish date of that section.



## About Author

Reza Rad is a [Microsoft Regional Director](#), an Author, Trainer, Speaker and Consultant. He has a BSc in Computer engineering; he has more than 15 years' experience in data analysis, BI, databases, programming, and development mostly on Microsoft technologies. He is a [Microsoft Data Platform MVP](#) for eight continuous years (from 2011 till now) for his dedication in Microsoft BI. Reza is an active blogger and co-founder of [RADACAD](#). Reza is also co-founder and co-organizer of [Difinity](#) conference in New Zealand.

His articles on different aspects of technologies, especially on MS BI, can be found on his blog: <http://www.radacad.com/blog>.

He wrote some books on MS SQL BI and also is writing some others, He was also an active member on online technical forums such as MSDN and Experts-Exchange, and was a moderator of MSDN SQL Server forums, and is an MCP, MCSE, and MCITP of BI. He is the leader of [the New Zealand Business Intelligence users group](#). He is also the author of very popular book [Power BI from Rookie to Rock Star](#), which is free with more than 1100 pages of content.

He is an International Speaker in Microsoft Ignite, Microsoft Business Applications Summit, Data Insight Summit, PASS Summit, SQL Saturday and SQL user groups. And He is a Microsoft Certified Trainer.

Reza's passion is to help you find the best data solution; he is Data enthusiast.



## Who should read this book?

BI Developers and Consultants who want to know how to develop solutions with this technology. BI Architects and Decision Makers who want to make their decision about using or not using Power BI in their BI applications. Business Analysts who want to have a better tool for playing with the data and learn tricks of producing insights easier. The book titled "Power BI from Rookie to Rockstar" and that means it will cover a wide range of readers. I'll start by writing 100 level, and we will go deep into 400 level at some stage. So, if you don't know what Power BI is, or If you are familiar with Power BI but want to learn some deep technical topics about Power Query M language, then this book is for you.

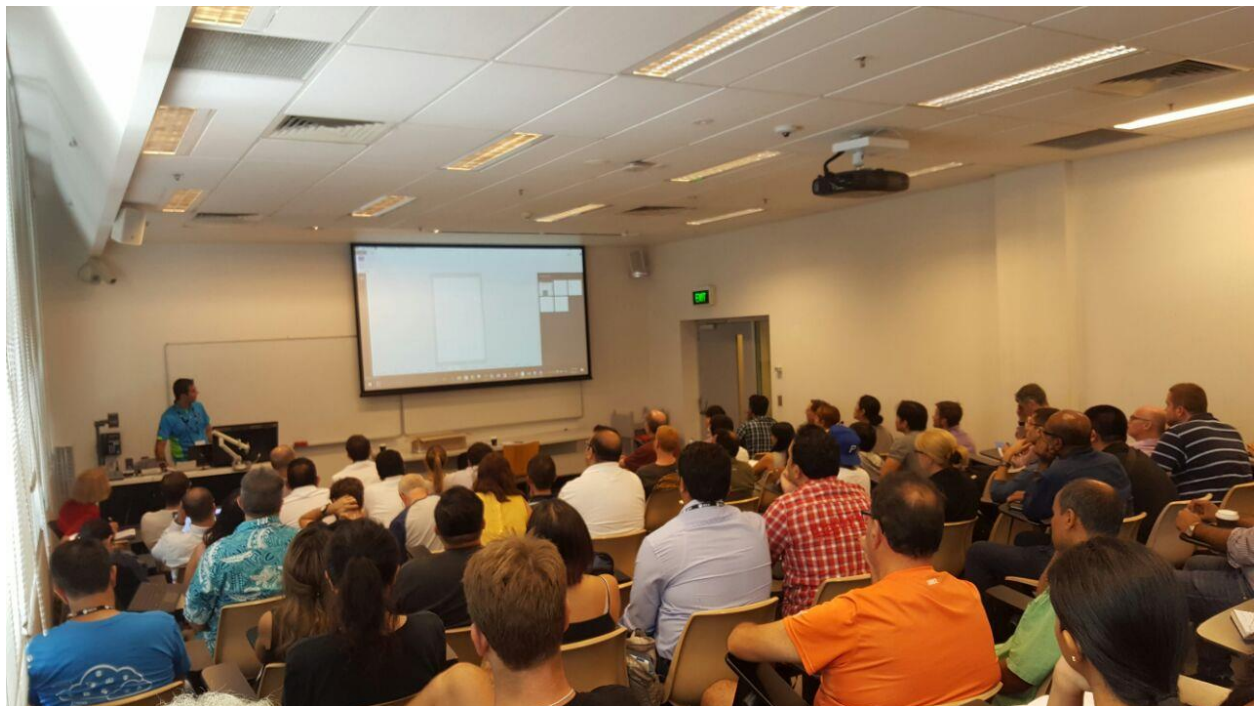
# Upcoming Training Courses

Reza runs Power BI training courses both online and in-person. RADACAD also runs Advanced Analytics with R, Power BI, Azure Machine Learning and SQL Server courses ran by Dr. Leila Etaati. Our courses run both online and in-person in major cities and countries around the world.

Check the schedule of upcoming courses here:

<http://radacad.com/events>

<http://radacad.com/power-bi-training>



# Heading Table of Content

About the New Edition and New Structure .....	3
About the book; Quick Intro from Author .....	4
About Author .....	5
Who should read this book?.....	6
Upcoming Training Courses.....	7
Heading Table of Content .....	8
Detailed Table of Content .....	11
<a href="#"><u>Part I: Getting Started with Power Query</u></a>	
What Is Power Query? Introduction to Data Mash-up Engine of Power BI .....	22
Get Started with Power Query: Movies Data Mash-Up.....	34
Data Preparation; First and Foremost Important Task in Power BI.....	61
<a href="#"><u>Part II: Get Data</u></a>	
Power BI Get Data From Excel: Everything You Need to Know.....	71
Power BI Get Data: From Azure SQL Database.....	88
Meetup Data Source for Power BI.....	115
<a href="#"><u>Part III: Transformations</u></a>	
Reference vs Duplicate in Power BI; Power Query Back to Basics .....	142
Append vs. Merge in Power BI and Power Query .....	157
Choose the Right Merge Join Type in Power BI .....	167
How to Change Joining Types in Power BI and Power Query.....	180
Find Mismatch Rows with Power Query in Power BI .....	186
Dates Between Merge Join in Power Query .....	193
Pivot and Unpivot with Power BI.....	202
Warning! Misleading Power Query Filtering.....	210
Grouping in Power Query; Getting The Last Item in Each Group .....	218
Fuzzy Matching in Power BI and Power Query; Match based on Similarity Threshold ..	231
Fetch Files and/or Folders with Filtering and Masking: Power Query.....	243
<a href="#"><u>Part IV: Dealing with Errors</u></a>	
Exception Reporting in Power BI: Catch the Error Rows in Power Query .....	249

Flawless Date Conversion in Power Query .....	268
Make Your Numeric Division Faultless in Power Query .....	279
<a href="#"><u>Part V: Power Query Formula Language: M</u></a>	
Power Query Formula Language: M .....	288
M or DAX? That is the Question! .....	301
Basics of M: Power Query Formula Language .....	309
Basics of Value Structures in M – Power Query Formula Language .....	321
Power Query Formula Language M : Table Functions Part 1 .....	334
List.Accumulate Hidden Gem of Power Query List Functions in Power BI .....	345
Power Query; Convert Time Stamp to Date Time .....	357
Get List of Queries in Power BI .....	359
Power Query Library of Functions; Shared Keyword .....	368
Writing Custom Functions in Power Query M .....	380
Day Number of Year, Power Query Custom Function .....	389
Power Query Function that Returns Multiple Values .....	399
Custom Functions Made Easy in Power BI Desktop .....	405
Search for a Column in the Entire Database with Table.ColumnNames in Power Query and Power BI .....	429
Power BI Custom Connector: Connect to Any Data Sources. Hello World! .....	442
<a href="#"><u>Part VI: Performance Tuning</u></a>	
Not Folding; the Black Hole of Power Query Performance .....	458
Performance Tip for Power BI; Enable Load Sucks Memory Up .....	473
Watch Your Steps! Power Query Performance Caution for Power BI .....	491
<a href="#"><u>Part VII: Power Query Use Cases</u></a>	
Create a Date Dimension in Power BI in 4 Steps – Step 1: Calendar Columns .....	497
Create a Date Dimension in Power BI in 4 Steps – Step 2: Fiscal Columns .....	510
Create a Date Dimension in Power BI in 4 Steps – Step 3: Public Holidays .....	520
Power Query Not for BI: Event Date and Time Scheduler – Part 1 .....	531
Power Query Not for BI: Event Date and Time Scheduler – Part 2 .....	549
Power Query Not for BI: Event Date and Time Scheduler – Part 3 .....	558

[Part VIII: A Tool to Help: Power BI Helper](#)

Exposing M Code and Query Metadata of Power BI (PBIX) File .....	567
Export the Entire M Power Query Script from a Power BI File, New Version of Power BI Helper, Search based on Field Description in the Model.....	576
Beautify M Script and Extract Row Level Security with Power BI Helper Version 4.0 .....	584
Other modules of the book.....	593
Power BI Training.....	594

# Detailed Table of Content

About the New Edition and New Structure .....	3
About the book; Quick Intro from Author .....	4
About Author .....	5
Who should read this book?.....	6
Upcoming Training Courses.....	7
Heading Table of Content .....	8
Detailed Table of Content .....	11
What Is Power Query? Introduction to Data Mash-up Engine of Power BI .....	22
What Is Power Query?.....	23
How to Use Power Query? .....	25
What Can You Do With Power Query? .....	25
Get Data From Wide Range of Sources.....	25
Apply Transformation In a Development Editor .....	30
Load Data into Destination .....	31
What Are Features of Power Query Premium?.....	32
Get Started with Power Query: Movies Data Mash-Up.....	34
Let's Get Started .....	36
Query Editor .....	40
Use a Query as a Reference.....	44
Append Queries.....	45
Extract First Characters.....	46
Remove Columns.....	49
Split Column .....	50
Replace Values .....	53
Trim.....	55
Applied Steps .....	55
Final Merge.....	56
Summary.....	60



Data Preparation; First and Foremost Important Task in Power BI.....	61
Why Data Preparation? .....	61
How to Design a Star Schema?.....	64
Design Tips .....	67
More to Come .....	69
Power BI Get Data From Excel: Everything You Need to Know.....	71
Excel Data Source.....	71
Loading Excel Tables into Power Query .....	73
Loading Excel Sheets into Power Query .....	75
What Happens If Excel Contains Formatting? .....	76
What Happens to Power View Sheets in Excel? .....	76
Pivot Tables and Pivot Charts?.....	77
What If Your Excel Table Has Merged Cells? .....	78
Example Excel Data Source: Olympic Games .....	79
Summary .....	87
Power BI Get Data: From Azure SQL Database.....	88
Preparation .....	89
Get Data From Azure SQL Database.....	98
Schedule Data Refresh.....	105
Direct Connection to Azure SQL Database from Power BI Website.....	109
Summary .....	113
Meetup Data Source for Power BI.....	115
About Meetup .....	115
Get Data with Power BI.....	119
TimeStamp to Date Time.....	124
Visualization .....	126
Adding More Data .....	130
Reference vs Duplicate in Power BI; Power Query Back to Basics .....	142
Duplicate .....	142
Reference .....	147
Duplicate vs. Reference .....	152
Summary .....	156
Append vs. Merge in Power BI and Power Query.....	157

Why Combine Queries? .....	157
Append .....	158
Merge .....	161
Choose the Right Merge Join Type in Power BI .....	167
What is Merge? .....	167
How to Merge Queries .....	169
Merging Based on Multiple Columns .....	171
Merge Types.....	172
Summary .....	178
How to Change Joining Types in Power BI and Power Query.....	180
Power BI Desktop .....	181
Power Query .....	183
Find Mismatch Rows with Power Query in Power BI .....	186
Sample Data Tables .....	187
Merge in Power Query.....	188
Summary .....	192
Dates Between Merge Join in Power Query .....	193
Problem Definition .....	194
Grain Matching .....	194
Step 1: Calculating Duration .....	195
Step 2: Creating List of Dates.....	196
Step 3: Expand List to Day Level.....	197
Merging Tables on the Same Grain .....	198
Final Step: Cleansing .....	200
Summary .....	201
Pivot and Unpivot with Power BI.....	202
Pivot: Turning Name, Value Rows to Columns .....	202
Unpivot; Turning Columns to Rows; Name, Values .....	207
Warning! Misleading Power Query Filtering.....	210
Filtering in Power Query .....	211
Basic Filtering.....	211
Misleading Behavior of Basic Filtering.....	212

Advanced Filtering: Correct Filtering .....	215
Summary .....	217
Grouping in Power Query; Getting The Last Item in Each Group .....	218
Learning Objectives for this section .....	218
Scenario.....	219
Get Data .....	219
Group By Transformation.....	220
First and Last Item in each Group.....	224
Fuzzy Matching in Power BI and Power Query; Match based on Similarity Threshold ..	231
Fuzzy Merge .....	234
Power Query Functions.....	239
Summary .....	242
Fetch Files and/or Folders with Filtering and Masking: Power Query .....	243
Fetch All Files in a Folder .....	243
Fetch All Files and Folders .....	245
Fetch Files and Folders with Masking.....	246
Exception Reporting in Power BI: Catch the Error Rows in Power Query .....	249
Error Happens .....	250
Why didn't Power Query Editor Catch the Error?.....	252
Dealing with Errors: Catching the Error Rows .....	255
Exception Report .....	266
Summary .....	267
Flawless Date Conversion in Power Query.....	268
Different Formats of Date.....	269
Automatic Type Conversion .....	271
Date Conversion Issue .....	274
Date Conversion Using Locale .....	276
Summary .....	278
Make Your Numeric Division Faultless in Power Query .....	279
Simple Division Calculation .....	280
Error Output .....	281
Infinity.....	283
NaN.....	284

Null Check .....	284
Function to Check All Anomalies.....	285
Summary .....	286
Power Query Formula Language: M .....	288
M or DAX? That is the Question!.....	301
What is M?.....	301
What is DAX?.....	302
Example Usage of M .....	303
Example Usage of DAX?.....	304
Calculated Column Dilemma.....	304
What Questions Can DAX Answer? .....	307
What Questions Can M Answer? .....	307
As a Power BI Developer Which Language Is Important to Learn? .....	307
Basics of M: Power Query Formula Language .....	309
What is M?.....	309
Syntax of M.....	310
End of the Line.....	312
Variable Names.....	313
Special characters .....	314
Escape character .....	314
Step by Step Coding.....	315
Literals .....	316
Function Call.....	317
Comments .....	318
A real-world example .....	319
Basics of Value Structures in M – Power Query Formula Language.....	321
Five Main Value Structures in Power Query.....	322
Navigating Through List .....	327
Navigating Through Record.....	328
Navigating Through Table.....	329
Concatenate List or Records .....	332
Summary.....	333

Power Query Formula Language M: Table Functions Part 1 .....	334
List.Accumulate Hidden Gem of Power Query List Functions in Power BI .....	345
List Transformations in Graphical Interface of Power Query .....	345
List Functions in M; Power Query Formula Language .....	346
List.Accumulate Function .....	348
Accumulate to Calculate Sum .....	348
Accumulate to Calculate Max .....	349
Accumulate as Product or Divide .....	351
Accumulate as Count .....	351
Accumulate as Concatenate (with a delimiter or without) .....	352
Accumulate as Count Token Exact Match .....	353
Accumulate as Count Token Partial Match .....	353
Accumulate as Conditions on Records .....	354
Summary .....	355
Power Query; Convert Time Stamp to Date Time .....	357
What is Timestamp .....	357
Power Query Convert Timestamp to Date Time .....	357
Get List of Queries in Power BI .....	359
Question: How to Fetch Name of All Queries into One Query? .....	360
Answer: Using #shared or #sections Keywords .....	360
Sample Scenario of Usage .....	365
Power Query Library of Functions; Shared Keyword .....	368
#shared Keyword .....	369
Use the Result set as a Table .....	373
Documentation of Function .....	376
Enumerators .....	378
Summary .....	378
Writing Custom Functions in Power Query M .....	380
Day Number of Year, Power Query Custom Function .....	389
Power Query Function that Returns Multiple Values .....	399
Return First and Last Dates of Month .....	400
Don't Limit Yourself .....	404

Custom Functions Made Easy in Power BI Desktop .....	405
What is Custom Function?.....	405
Benefits of Custom Function .....	406
How to Create a Custom Function? .....	407
Using Generator .....	418
Consuming Function .....	420
Editing Function.....	423
Limitations .....	425
Example at the End.....	426
Summary .....	427
Search for a Column in the Entire Database with Table.ColumnNames in Power Query and Power BI .....	429
The Problem.....	430
Table.ColumnNames .....	433
Search through Columns.....	437
Summary .....	440
Power BI Custom Connector: Connect to Any Data Sources. Hello World! .....	442
What is a Custom Connector? .....	442
Install Power Query SDK.....	444
Create the First Project .....	445
Coding Language: M.....	446
Structure of Query Files .....	447
Write a Sample Function .....	449
Testing the Result .....	450
Publishing Custom Connector .....	451
Using the Connector .....	452
Summary.....	455
Not Folding; the Black Hole of Power Query Performance .....	458
Query Folding .....	459
Is Query Folding Good or Bad?.....	461
Can I see the Native Query? .....	462
So Why Not Query Folding? .....	463

Example: Merge Columns .....	464
How Do I know Which Transformations Folds? .....	471
Performance Tip for Power BI; Enable Load Sucks Memory Up .....	473
Load Mechanism for Power Query .....	473
Example Scenario .....	475
Build the Transformations .....	476
Default Behavior: Enable Load .....	484
What about Hiding from Report? .....	486
Disable Load to Save Memory .....	486
Summary .....	490
Watch Your Steps! Power Query Performance Caution for Power BI .....	491
Too Many Variables .....	491
What is the Solution? .....	495
Create a Date Dimension in Power BI in 4 Steps – Step 1: Calendar Columns .....	497
Introduction .....	497
Why Power Query? .....	498
Getting Started .....	499
Parameters .....	500
First Step: Build the Base Query .....	501
Next Step: Adding Extra Calendar Columns .....	505
Script .....	508
Summary .....	509
Create a Date Dimension in Power BI in 4 Steps – Step 2: Fiscal Columns .....	510
The parameter for Fiscal Year Star .....	510
Creating Fiscal Columns .....	511
Code of this example .....	517
Summary .....	519
Create a Date Dimension in Power BI in 4 Steps – Step 3: Public Holidays .....	520
Live API for Public Holidays .....	521
Get Data from Web API .....	522
Add Parameters to Public Holidays URL .....	524



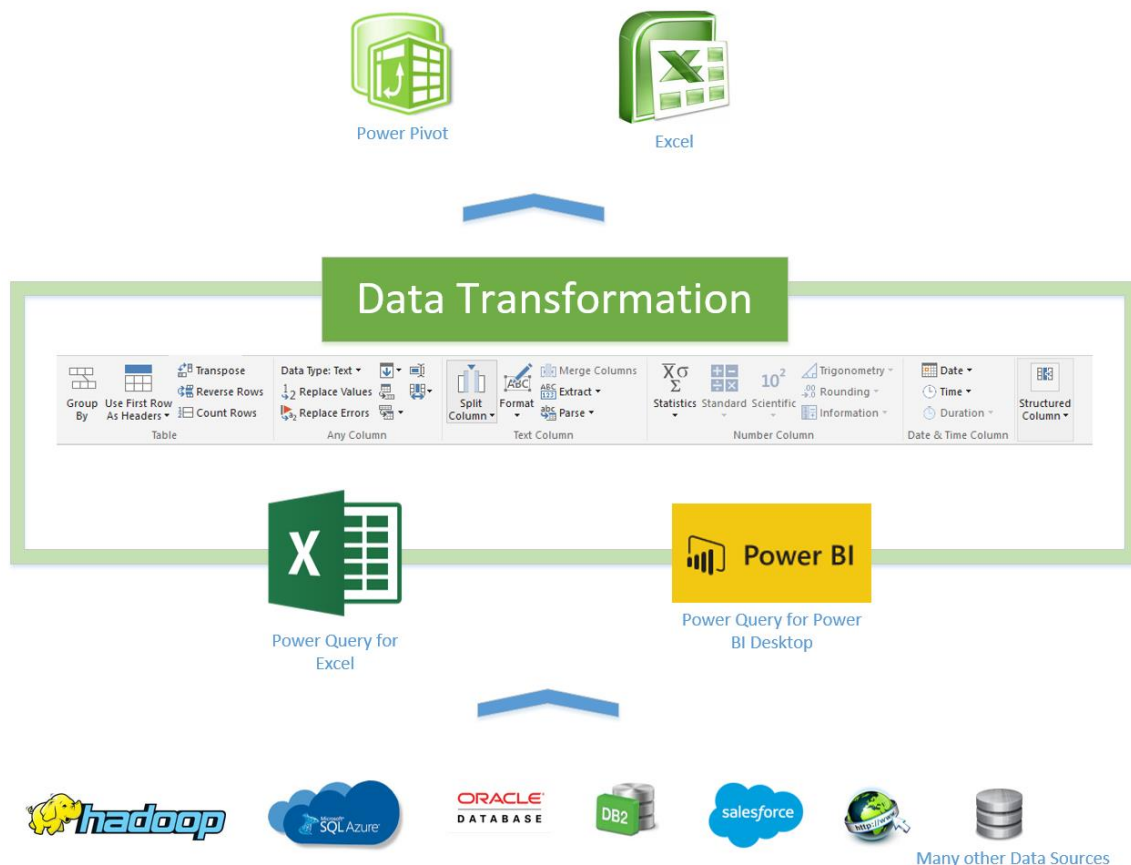
Merge Date Dimension with Public Holidays .....	526
Summary.....	530
Power Query Not for BI: Event Date and Time Scheduler – Part 1 .....	531
Why? .....	532
What You Will Learn?.....	536
Get Data from Web.....	537
Apply Basic Transformations .....	540
Insert a Step.....	544
Append Queries .....	546
Next Steps .....	548
Power Query Not for BI: Event Date and Time Scheduler – Part 2 .....	549
Group Rows .....	549
Concatenate Rows.....	551
Final Polish .....	556
Next Steps .....	557
Power Query Not for BI: Event Date and Time Scheduler – Part 3 .....	558
Using Parameters .....	558
Summary .....	565
Call to Action.....	565
Exposing M Code and Query Metadata of Power BI (PBIX) File .....	567
Structure of a PBIX File .....	567
DataMashup File: Everything You Need.....	569
M Script File .....	570
Metadata Information in XML Format .....	572
Summary .....	575
Export the Entire M Power Query Script from a Power BI File, New Version of Power BI Helper, Search based on Field Description in the Model.....	576
Export M Script.....	577
Download Power BI Helper (FREE) .....	577
Search based on Field Description or Properties in the Model.....	581
Power BI Helper on a Virtual Machine .....	583
More Features?.....	583
Beautify M Script and Extract Row Level Security with Power BI Helper Version 4.0 .....	584

Download.....	585
Row-Level Security .....	589
Summary .....	592
Other modules of the book.....	593
Power BI Training.....	594

# Part I: Getting Started with Power Query

# What Is Power Query? Introduction to Data Mash-up Engine of Power BI

Published Date: August 15, 2015



When you get data in Power BI, you use Power Query Component. In this chapter, you will learn about What Power Query is, and what are different types of sources that Power Query can connect. Power Query also has a great list of transformations that can be applied on the data set as well (which will be covered in next chapter), and the Power Query formula language M can be used for complex and powerful data transformation situations (will be covered in a chapter after).

In this section, you will read an introduction to Power Query. You will learn;

- What is Power Query?
- What types of works can be done with Power Query?
- What are requirements to run Power Query?

- What are features of Power Query Premium?

## What Is Power Query?

Power Query previously named as Data Explorer. Data Explorer was released as a public preview for the [first time in February 2013](#). Data Explorer then renamed to Power Query at July 2013, and from that time it had lots of enhancement on the product. Power Query is on a regular and frequent update plan by Microsoft team, and usually, you can see monthly updates on this, here is the [latest update notes](#) (released yesterday!) Power Query has been tested a lot during this period and nowadays used in many real-world data transformations and BI solutions.

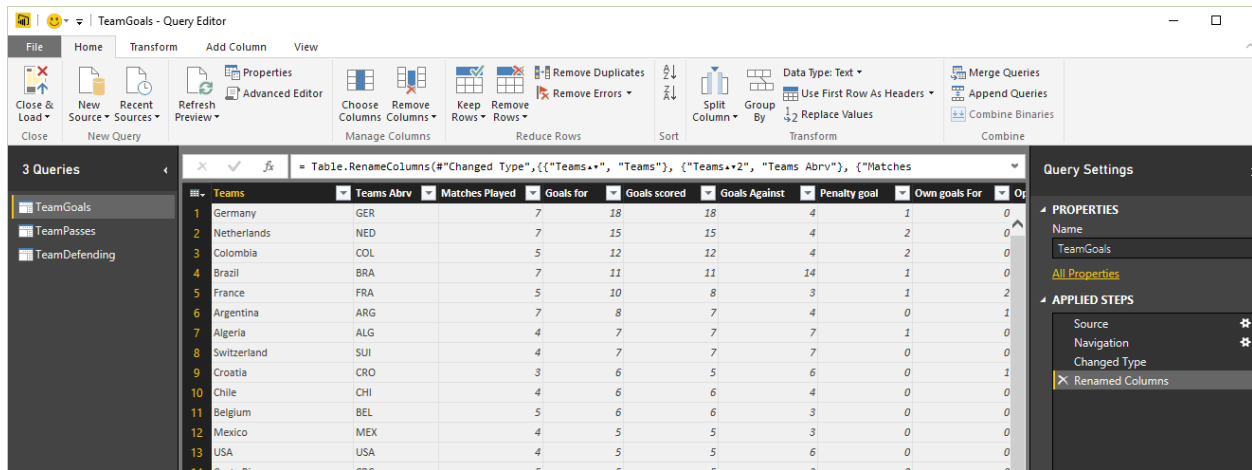
Power Query is a data extraction and transformation engine. The engine comes with a formula language and a graphical tool. The graphical tool has two major setup versions; one embedded in Power BI Desktop tool and the other one as an Add-In for Excel. The graphical tool has a list of transformations that can be applied on a data set, and it also supports different data sources. However, the Power Query formula language is much more powerful than the GUI. There are some features in Power Query engine that not yet has been implemented through GUI, but they are available through M (formula language).

Power Query can connect to a set of data sources and read data from them. Set of data sources is variable from text files, to web URLs, from database systems to some applications. A wide range of data sources is supported. So to respond to one of the very first questions that usually appears when I introduce this product that Can Power Query connect to Oracle? Sure it does! Not only Oracle, but also MySQL, PostgreSQL, DB2, Sybase, and Teradata.

Power Query can apply many transformations to the data set. You can apply simple transformations such as trimming a text value and applying numeric calculations to complex transformations easily such as pivot and unpivot. Power Query uses a function library for applying transformations, and the function library contains heaps of transformations for every data type such as table, text, record, list, date, number and so on.

Power Query graphical interface is so easy to work with that even business analyst, or a power user can work with it, on the other hand, Power Query M language is so powerful that can be used for complex real-world challenges of data transformations. Power Query can load the result set into an Excel spreadsheet, or it can load it into Power Pivot for data modeling. The version of Power Query used in Power BI Desktop loads the result set into a Power Pivot model. I will go through details of Power Pivot in future

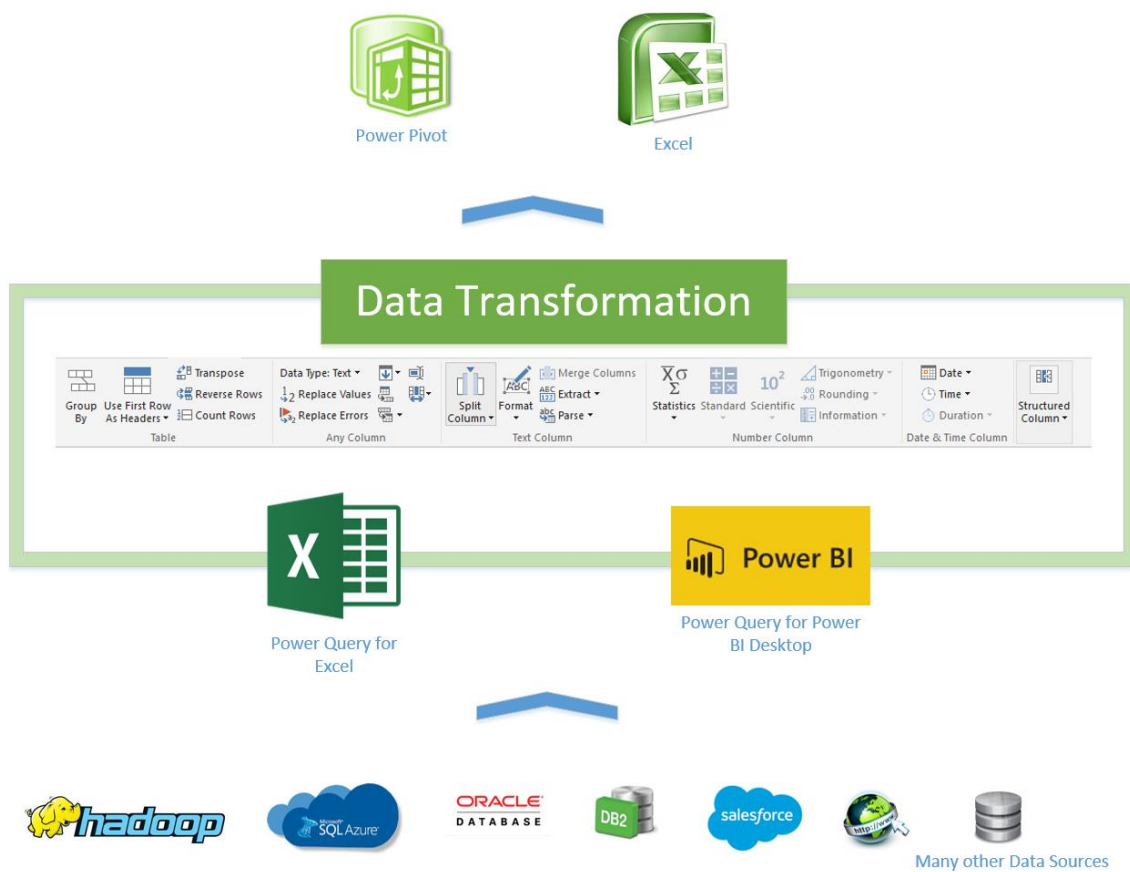
chapters, for now, it would be enough to know that Power Pivot is In-Memory tabular data model engine. Here is a screenshot of the Query Editor window



The screenshot shows the Power Query Editor window with a table named 'Table1' containing football match data. The table has columns for Teams, Teams Abbrv, Matches Played, Goals for, Goals scored, Goals Against, Penalty goal, and Own goals For. The data is filtered to show matches from 1 to 14.

Teams	Teams Abbrv	Matches Played	Goals for	Goals scored	Goals Against	Penalty goal	Own goals For
1 Germany	GER	7	18	18	4	1	0
2 Netherlands	NED	7	15	15	4	2	0
3 Colombia	COL	5	12	12	4	2	0
4 Brazil	BRA	7	11	11	14	1	0
5 France	FRA	5	10	8	3	1	2
6 Argentina	ARG	7	8	7	4	0	1
7 Algeria	ALG	4	7	7	7	1	0
8 Switzerland	SUI	4	7	7	7	0	0
9 Croatia	CRO	3	6	5	6	0	1
10 Chile	CHI	4	6	6	4	0	0
11 Belgium	BEL	5	6	6	3	0	0
12 Mexico	MEX	4	5	5	3	0	0
13 USA	USA	4	5	5	6	0	0
14 Costa Rica	CRI	5	5	5	2	0	0

In below you can see a high-level diagram of Power Query conceptually:



## How to Use Power Query?

Power Query is available in three different setups:

1. As an Excel Add-In for Excel 2010 and 2013
2. Embedded in Excel 2016
3. Embedded in Power BI Desktop

So if you want to install then you have to install one of the options below:

### **Excel Add-In for Excel 2010 and 2013:**

<https://www.microsoft.com/en-us/download/details.aspx?id=39379>

Please note that the link above might change because Power Query updates frequently and a new version will be available almost every month. So you can simply Google it as Power Query Excel add-in.

### **Excel 2016 download link:**

<https://products.office.com/en-us/office-2016-preview>

At the time of writing this blog post, Excel 2016 is in the preview stage, so the link is likely to change.

### **Power BI Desktop:**

<https://powerbi.microsoft.com/desktop>

## What Can You Do With Power Query?

### Get Data From Wide Range of Sources

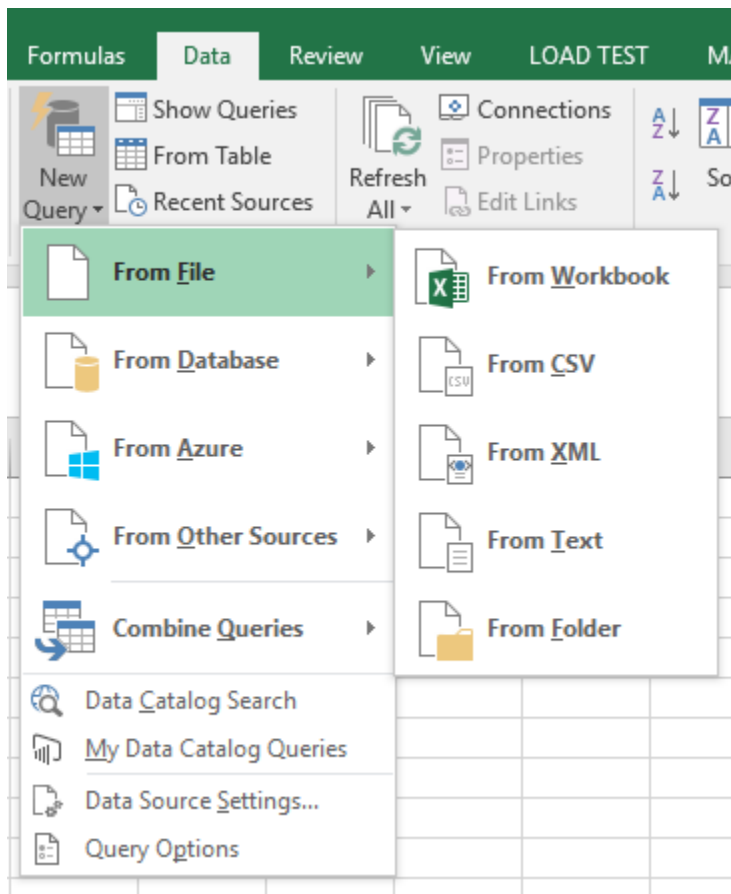
With Power Query, you can connect to a wide range of data sources. SQL Server or DB2 or Oracle.... All of this database are supported as a source. You can even connect to an Analysis Services instance and fetch data from it. You can connect to file data structures such as text files, XML, CSV, and Excel. You can even read the list of files in a folder! You can connect to a range of applications such as Facebook, Salesforce, CRM Online, etc. and get data from them. You can get data from Azure services such as Azure SQL



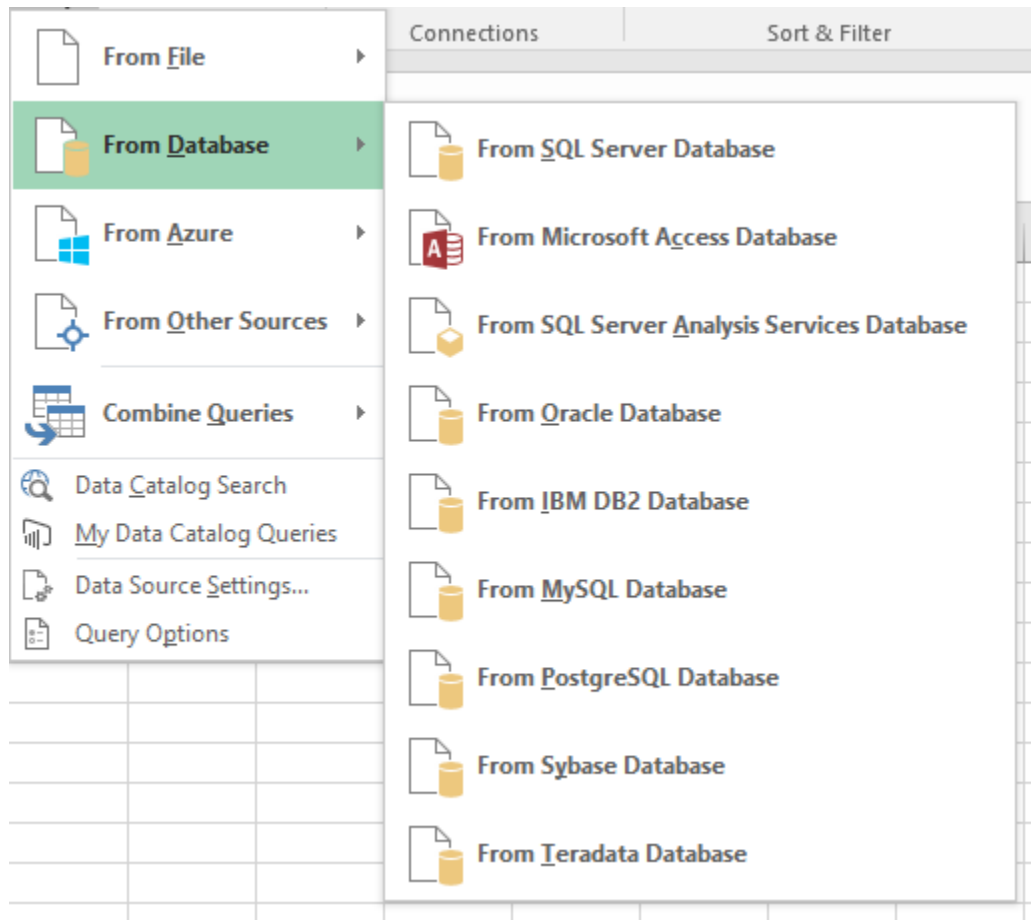
database, Azure HD Insight, Azure Blob storage, etc. There are many data sources supported for Power Query (and obviously for Power BI). Also, more data sources will be available in every update of Power Query or Power BI.

Here is an example set of data sources supported in Power Query (Excel version):

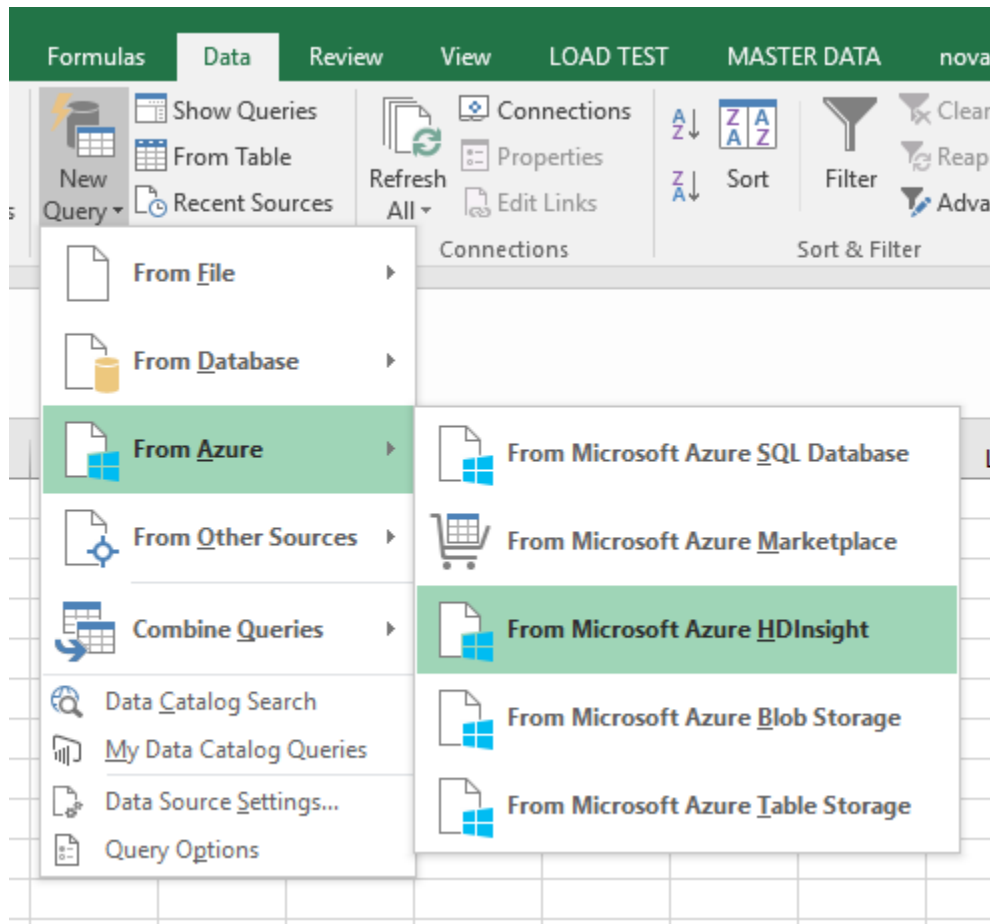
### File Data Sources



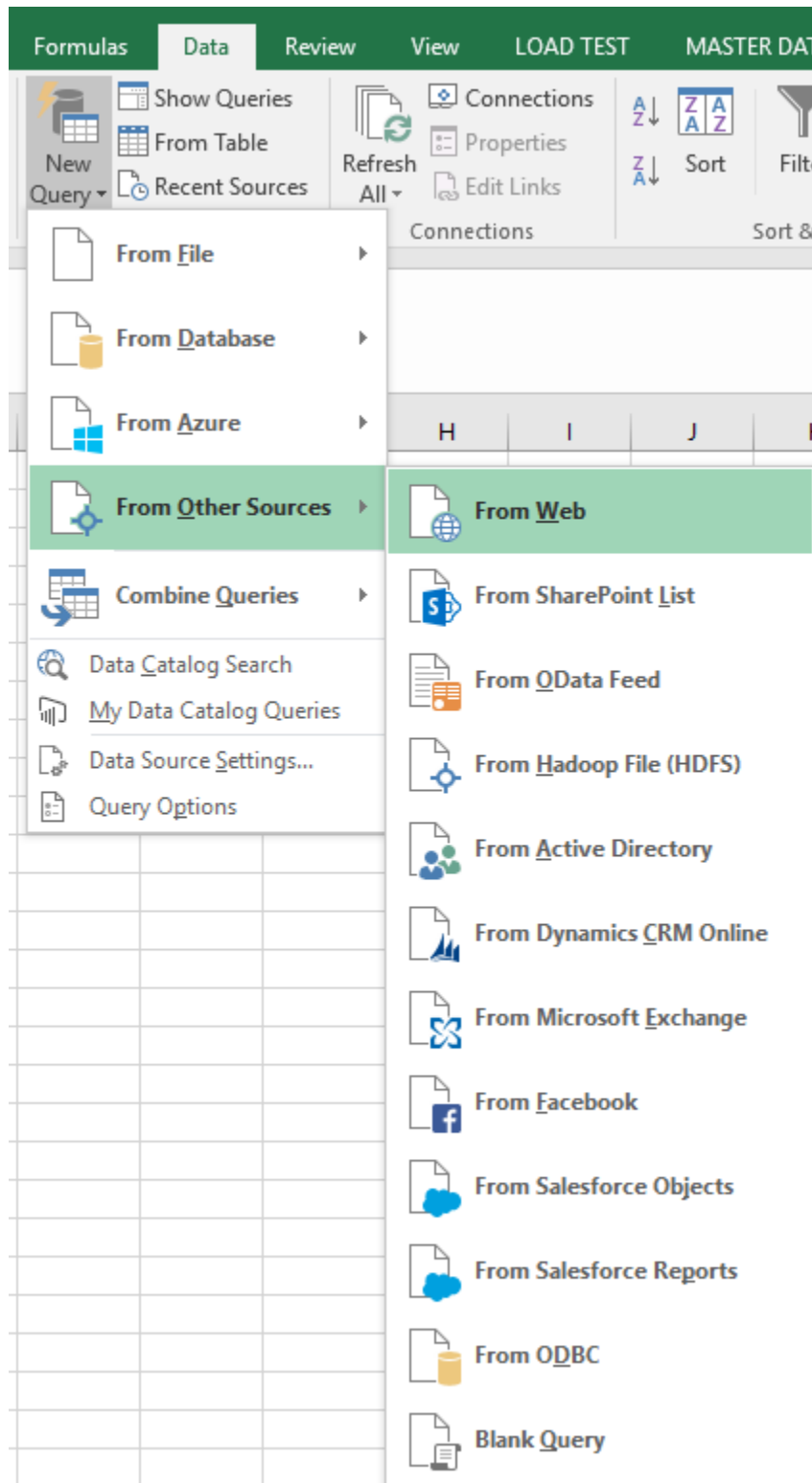
### Databases



Azure

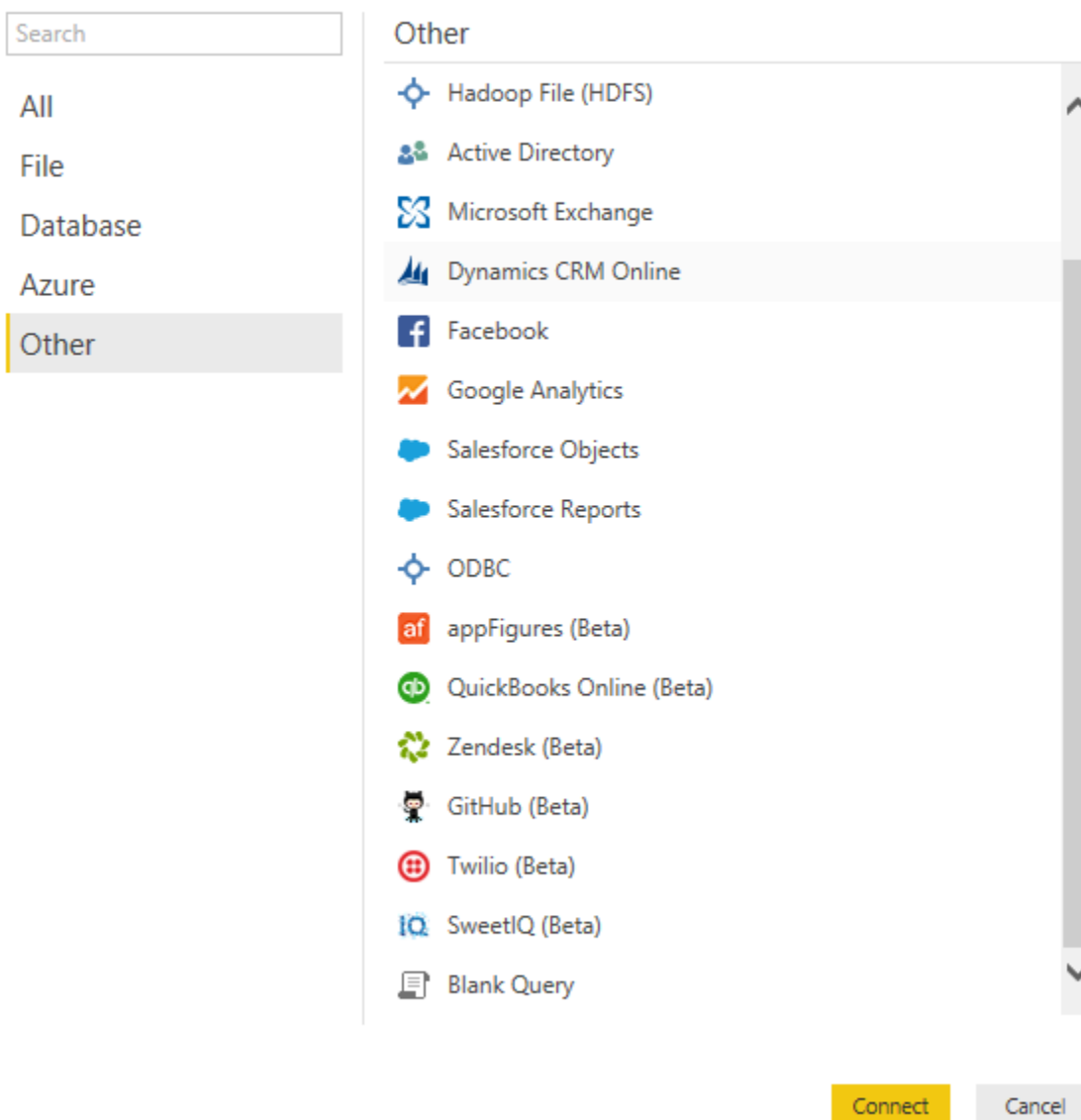


Other Sources



The Power Query version in Power BI Desktop supports some new applications that still is not implemented as Power Query for Excel; you see some of them below:

## Get Data



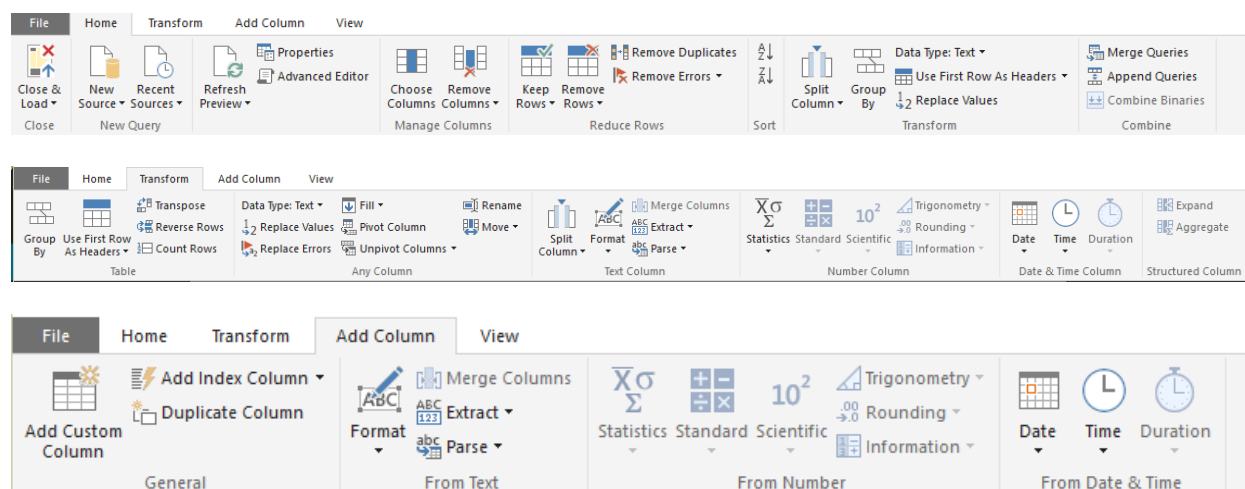
## Apply Transformation In a Development Editor

Power Query looks at the data values with data types such as Table, Record, List, DateTime, Text, Number, Boolean, etc. There are many data transformation functions for any of these data types. You can apply Merge (similar to join) or Append (similar to UNION) to two tables. You can apply text functions such as getting part of a string, trimming it or length of the string. You can apply mathematical functions. You can apply

DateTime functions such as functions for the year, Month, day and week. There are two way to apply these transformations;

1. From Query Editor: Graphical User Interface
2. From M query language: scripting language

Query Editor will give you a great experience of most common transformations through the very easy user interface. You can apply most of the transformation with the matter of a few clicks. The Query Editor in Power BI Desktop or Power Query Add-In for Excel has many common transformations listed. You can see some of them in below screenshots:

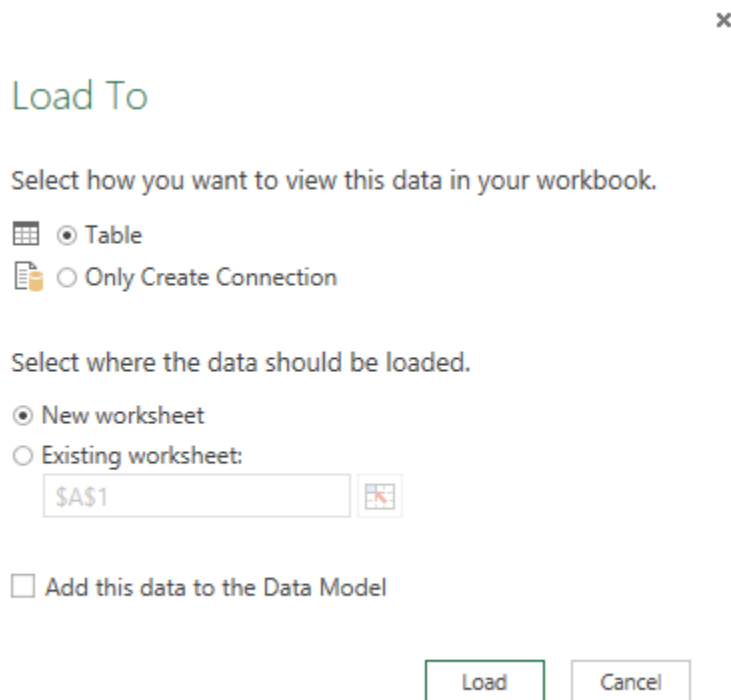


M is the formula language behind the scenes of Power Query. Everything you do in the Query Editor will be translated to an M script. M contains a full list of functions that you can use. So the powerful side of Power Query is M. I will go deep into details of M in this book because you would need it for solving complex challenges. M is a functional language, and it has a simple structure. The screenshot below shows an M Code. The details of information about M scripting will be covered in the next sections.

```
let
    Source = Folder.Contents("C:\Users\Reza\Dropbox\Speaking"),
    TypeAdded=Table.AddColumn(Source,"Type",each Value.Is([Content],type table)),
    Folders=Table.SelectRows(TypeAdded, each [Type]=true),
    Sorted=Table.Sort(Folders,{"Date created", Order.Descending})
in
    Sorted
```

## Load Data into Destination

You can use Power Pivot as the destination for Power Query to load result set into a data model, or you can use a simple Excel spreadsheet for loading data. If you use Power BI Desktop the result set of Power Query automatically will be loaded into a model.



## What Are Features of Power Query Premium?

This question might sound weird at first glance, but makes sense when you think about it that all features I mentioned above are available for free! You don't have to pay anything for it. Getting data from different sources, applying all kind of transformations to it, and loading it into a data model is all free. So now the question makes sense; What are features of Power Query Premium?

### Using Data Catalog

Data Catalog is a metadata definition service that you can define data sources from your organizational data stores or from public data stores that you trust. You can define descriptors for the data structure so Power Query can search through the Data Catalog and fetch information based on it.



## **Sharing Queries**

You can share your Power Query scripts and queries within your organization

## **Management using of Shared Queries**

You can check the usage of queries that you've shared

As you see in above most of the features for Power Query Premium is related to Office 365 usage for sharing or Power BI and Azure for data catalog and structure. Most of the features in Power Query (Essential features I have to say) is available for free!

In summary in this section you've learned about What is Power Query and what are components of it, you've learned features of Power Query, and now you are probably thinking about the usage of it in scenarios and challenges that you might have right now! Good start, in next sections I will go through the experience of getting data with Power Query and Power BI Desktop.

# Get Started with Power Query: Movies Data Mash-Up

Published Date: September 1, 2015



As another section of the [Power BI online book: from Rookie to Rockstar](#), I would like to get started working with Power Query. From my point of view learning through an example is the best way to learn new technology. For this post, I have decided to use the movie's data to be mashed up. I used this example because the movie's data is a fun example at the early sections of the book, you all watch movies, and you will see many familiar titles here. If you want to learn about Power Query or you need a Power Query introduction before this example, read the previous post: [What Is Power Query?](#) [Introduction to Data Mashup Engine of Power BI](#).

You can use either Power Query for Excel or Power Query as part of the Power BI Desktop for running this example. I use two data sets for this example:

1. Worldwide gross sales information of movies

This information is available on <http://www.boxofficemojo.com> website, as below:

Daily Box Office (Sun.) | Weekend Box Office (Aug. 28–30) | #1 Movie: 'Straight Outta Compton' | Showtimes

# Box Office Mojo

Search Site

Social

Facebook
Twitter

Features

News
Release Sched.
Showtimes
at

Box Office

Daily
Weekend
Weekly
Monthly
Quarterly
Seasonal
Yearly
All Time
Chart Watch
International

Indices

Movies A-Z
Studios
People

All Time Box Office

WORLDWIDE GROSSES

#1-100 - #101-200 - #201-300 - #301-400 - #401-500 - #501-600 - #601-615


Pink highlight = official revisions of older movies  
Gold highlight = now playing or recent movies

Rank	Title	Studio	Worldwide	Domestic / %	Overseas / %	Year^
1	Avatar	Fox	\$2,788.0	\$760.5 27.3%	\$2,027.5 72.7%	2009^
2	Titanic	Par.	\$2,186.8	\$658.7 30.1%	\$1,528.1 69.9%	1997^
3	Jurassic World	Uni.	\$1,636.7	\$643.1 39.3%	\$993.6 60.7%	2015
4	Marvel's The Avengers	BV	\$1,519.6	\$623.4 41.0%	\$896.2 59.0%	2012
5	Furious 7	Uni.	\$1,511.7	\$351.0 23.2%	\$1,160.7 76.8%	2015
6	Avengers: Age of Ultron	BV	\$1,401.3	\$457.5 32.6%	\$943.8 67.4%	2015
7	Harry Potter and the Deathly Hallows Part 2	WB	\$1,341.5	\$381.0 28.4%	\$960.5 71.6%	2011
8	Frozen	BV	\$1,274.2	\$400.7 31.4%	\$873.5 68.6%	2013
9	Iron Man 3	BV	\$1,215.4	\$409.0 33.7%	\$806.4 66.3%	2013

1. Top 250 movies ranked by people in IMDB website

IMDB is the movie database on the internet that users can rate movies. List of top 250 movies rated by users [listed here](#) in the website as below:

35 | Page



[Movies, TV & Showtimes](#)
[Celebs, Events & Photos](#)
[News & Community](#)
[Watchlist](#)

## IMDb Charts

# Top 250 Movies

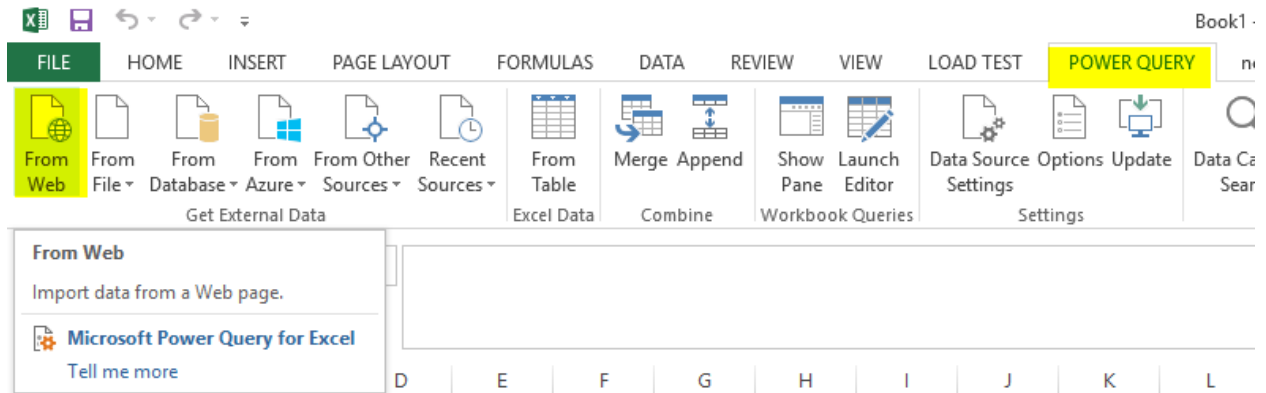
As voted by regular IMDb users

Showing 250 Titles Sort by:

Rank & Title	IMDb Rating	Your Rating	
1. <a href="#">The Shawshank Redemption</a> (1994)	★ 9.2	☆	+
2. <a href="#">The Godfather</a> (1972)	★ 9.2	☆	+
3. <a href="#">The Godfather: Part II</a> (1974)	★ 9.0	☆	+
4. <a href="#">The Dark Knight</a> (2008)	★ 8.9	☆	+
5. <a href="#">Schindler's List</a> (1993)	★ 8.9	☆	+
6. <a href="#">12 Angry Men</a> (1957)	★ 8.9	☆	+
7. <a href="#">Pulp Fiction</a> (1994)	★ 8.9	☆	+
8. <a href="#">The Good, the Bad and the Ugly</a> (1966)	★ 8.9	☆	+

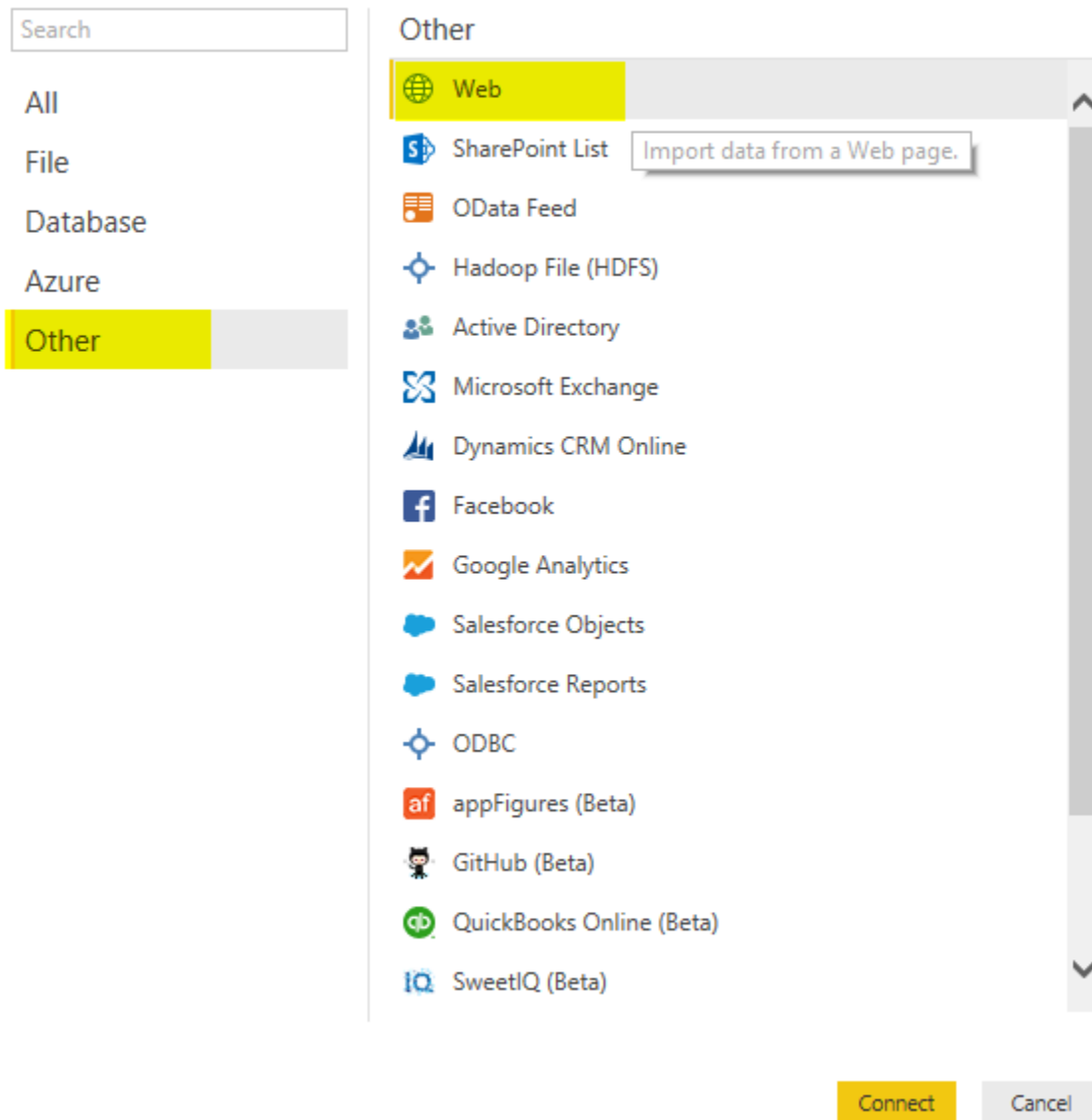
## Let's Get Started

Start by getting gross sales data; Open Excel, then Power Query Tab, and then from Web;



Or Open Power BI Desktop and Get Data from Web

## Get Data



Then Enter the web page URL for the top 100 sold movies all the time from this link:  
<http://www.boxofficemojo.com/alltime/world/>

## From Web

Enter a Web page URL.

URL

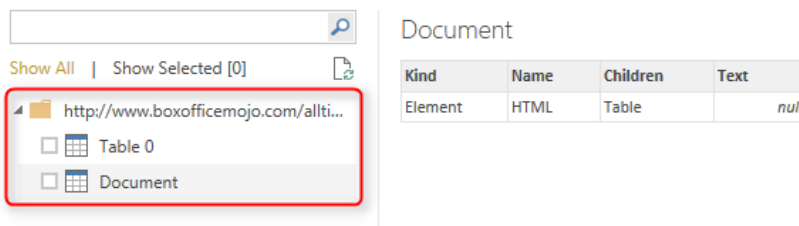
<http://www.boxofficemojo.com/alltime/world/?pagenum=1&p=.htm>

OK

Cancel

Click OK, after quick processing; you will see a Navigator window. Power Query will check for any tables in the HTML web page and will come back with a list of tables on the left side under the URL address;

## Navigator



Document

Kind	Name	Children	Text
Element	HTML	Table	null

Click on Table 0. You will see a preview of data in the table in the main pane. Now tick the checkbox for Table 0 and click on Edit button in Navigator

## Navigator

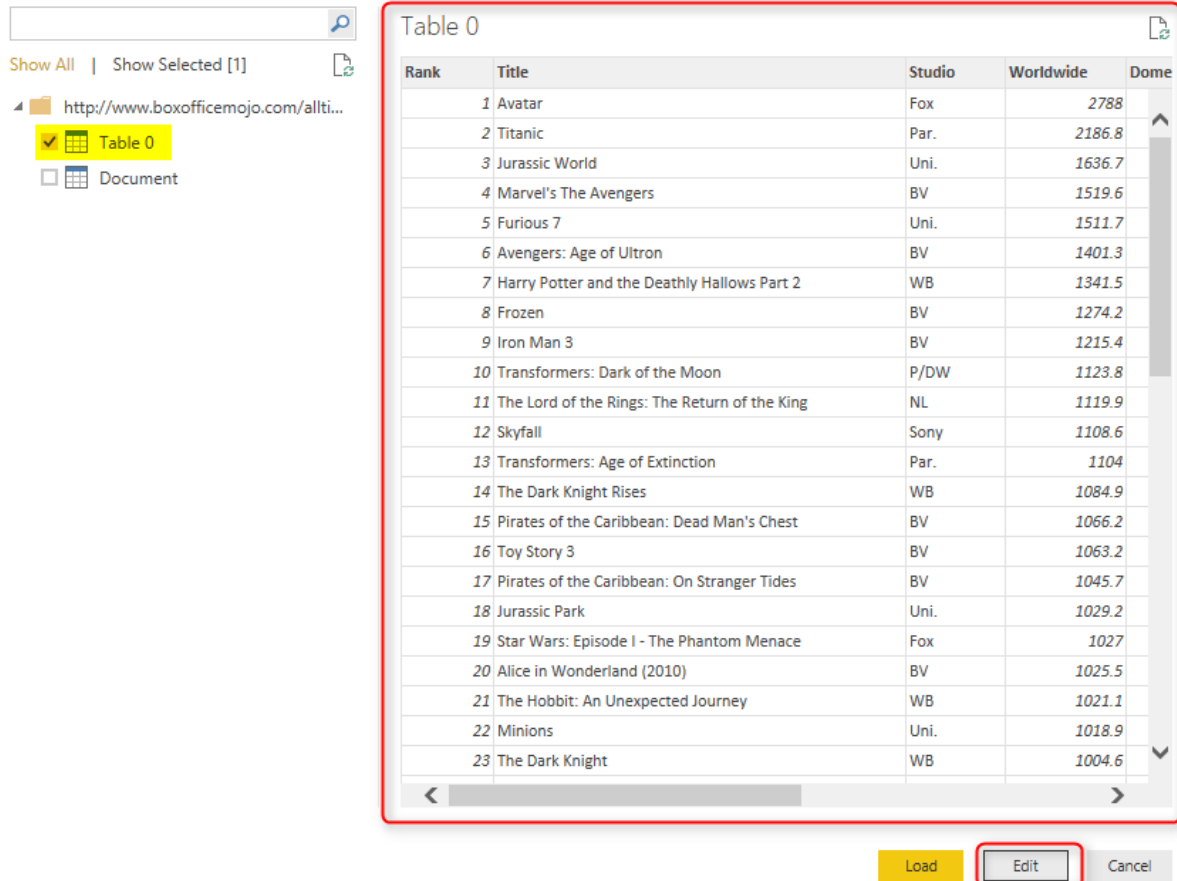


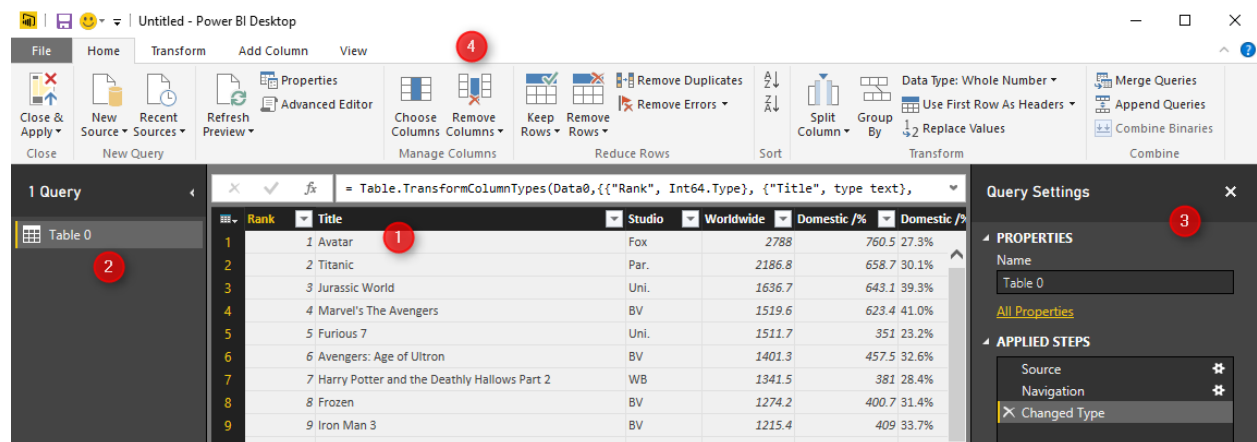
Table 0

Rank	Title	Studio	Worldwide	Domestic
1	Avatar	Fox	2788	
2	Titanic	Par.	2186.8	
3	Jurassic World	Uni.	1636.7	
4	Marvel's The Avengers	BV	1519.6	
5	Furious 7	Uni.	1511.7	
6	Avengers: Age of Ultron	BV	1401.3	
7	Harry Potter and the Deathly Hallows Part 2	WB	1341.5	
8	Frozen	BV	1274.2	
9	Iron Man 3	BV	1215.4	
10	Transformers: Dark of the Moon	P/DW	1123.8	
11	The Lord of the Rings: The Return of the King	NL	1119.9	
12	Skyfall	Sony	1108.6	
13	Transformers: Age of Extinction	Par.	1104	
14	The Dark Knight Rises	WB	1084.9	
15	Pirates of the Caribbean: Dead Man's Chest	BV	1066.2	
16	Toy Story 3	BV	1063.2	
17	Pirates of the Caribbean: On Stranger Tides	BV	1045.7	
18	Jurassic Park	Uni.	1029.2	
19	Star Wars: Episode I - The Phantom Menace	Fox	1027	
20	Alice in Wonderland (2010)	BV	1025.5	
21	The Hobbit: An Unexpected Journey	WB	1021.1	
22	Minions	Uni.	1018.9	
23	The Dark Knight	WB	1004.6	

Load Edit Cancel

## Query Editor

After clicking on Edit, you will see the Query Editor window opened. This is an editor that you will spend most of your time on data mash-up here.



1 Query

Table 0

Rank	Title	Studio	Worldwide	Domestic /%	Domestic /%
1	Avatar	Fox	2788	760.5	27.3%
2	Titanic	Par.	2186.8	658.7	30.1%
3	Jurassic World	Uni.	1636.7	643.1	39.3%
4	Marvel's The Avengers	BV	1519.6	623.4	41.0%
5	Furious 7	Uni.	1511.7	351	23.2%
6	Avengers: Age of Ultron	BV	1401.3	457.5	32.6%
7	Harry Potter and the Deathly Hallows Part 2	WB	1341.5	381	28.4%
8	Frozen	BV	1274.2	400.7	31.4%
9	Iron Man 3	BV	1215.4	409	33.7%
10	Transformers: Dark of the Moon	P/DW	1123.8	357.4	31.4%

Query Settings

PROPERTIES

Name: Table 0

APPLIED STEPS

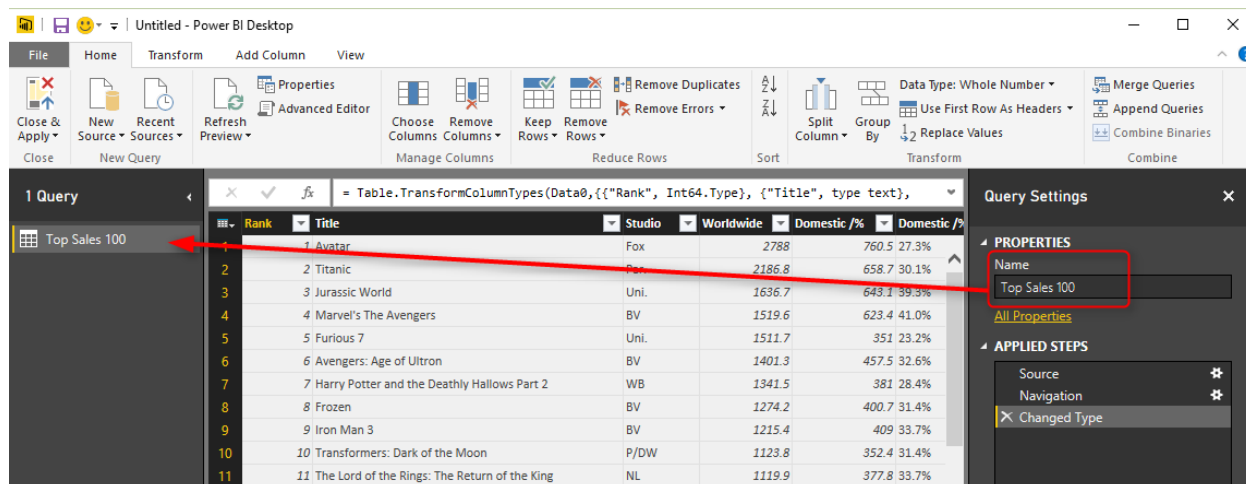
- Source
- Navigation
- Changed Type



Query Editor has four main sections (numbers matched to screenshot above);

1. Main dataset pane; This is the central area that the result set will be displayed as a preview with a limited number of rows
2. List of Queries; Left-hand side pane will show a list of all queries in this solution or file
3. Query Settings pane; Properties such as Name of the query can be set here. Also, a list of all applied steps to the current query is visible in this pane.
4. Transformations Menu; Power Query has many transformations options in GUI that are available through the menu in the top section

Rename the existing query to Top Sales 100



Rank	Title	Studio	Worldwide	Domestic /%	Domestic /%
1	Avatar	Fox	2788	760.5	27.3%
2	Titanic	Par.	2186.8	658.7	30.1%
3	Jurassic World	Uni.	1636.7	643.1	39.3%
4	Marvel's The Avengers	BV	1519.6	623.4	41.0%
5	Furious 7	Uni.	1511.7	351	23.2%
6	Avengers: Age of Ultron	BV	1401.3	457.5	32.6%
7	Harry Potter and the Deathly Hallows Part 2	WB	1341.5	381	28.4%
8	Frozen	BV	1274.2	400.7	31.4%
9	Iron Man 3	BV	1215.4	409	33.7%
10	Transformers: Dark of the Moon	P/DW	1123.8	352.4	31.4%
11	The Lord of the Rings: The Return of the King	NL	1119.9	377.8	33.7%


Our goal in this example is to join the data set of global gross sales with the IMDB user rating, and then analyze to see what are best sellers in movie titles among the best-rated movies or not? So the more data in gross sales we get would give us better analysis. The above URL only gives us top 100 sold movies. But the option to go to pages for rest of the result set is available;


# Box Office Mojo

Search Site **All Time Box Office**

Search...

**Social**

 Facebook


 Twitter

**Features**

[News](#)

[Release Sched.](#)

[Showtimes](#)

at 

**WORLDWIDE GROSSES**

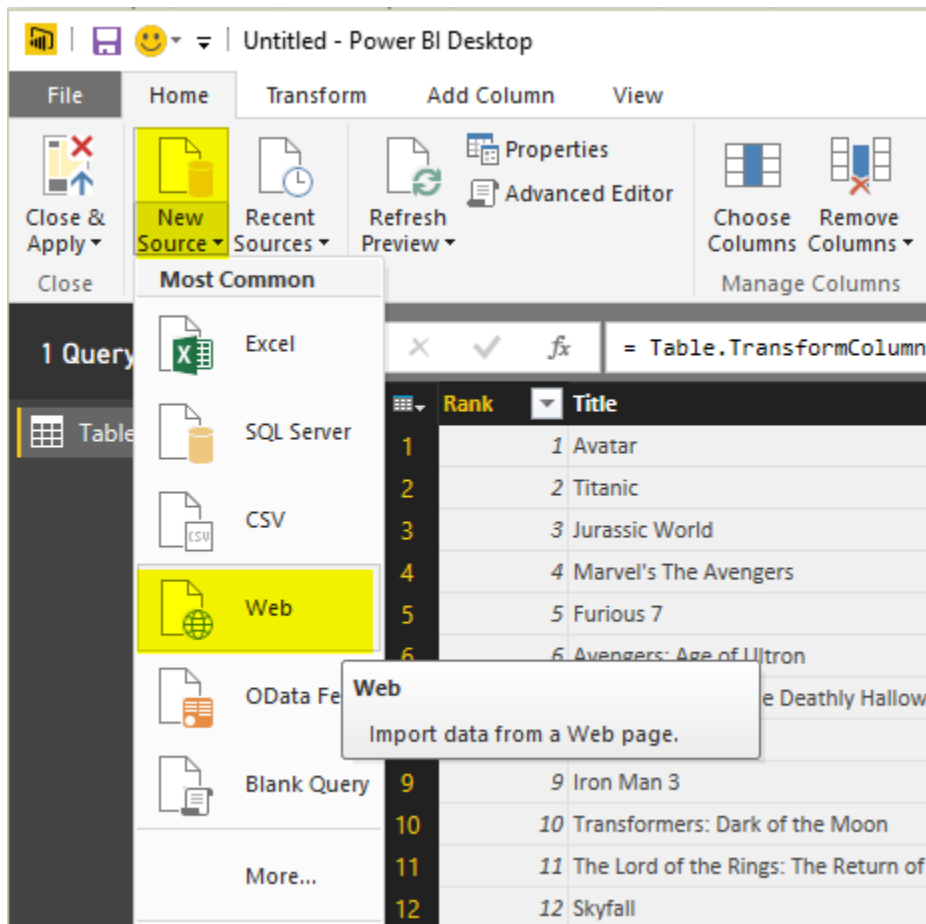
**#1-100 - #101-200 - #201-300 - #301-400 - #401-500 - #501-600 - #601-615**

Pink highlight = official revisions of older movies  
Gold highlight = now playing or recent movies

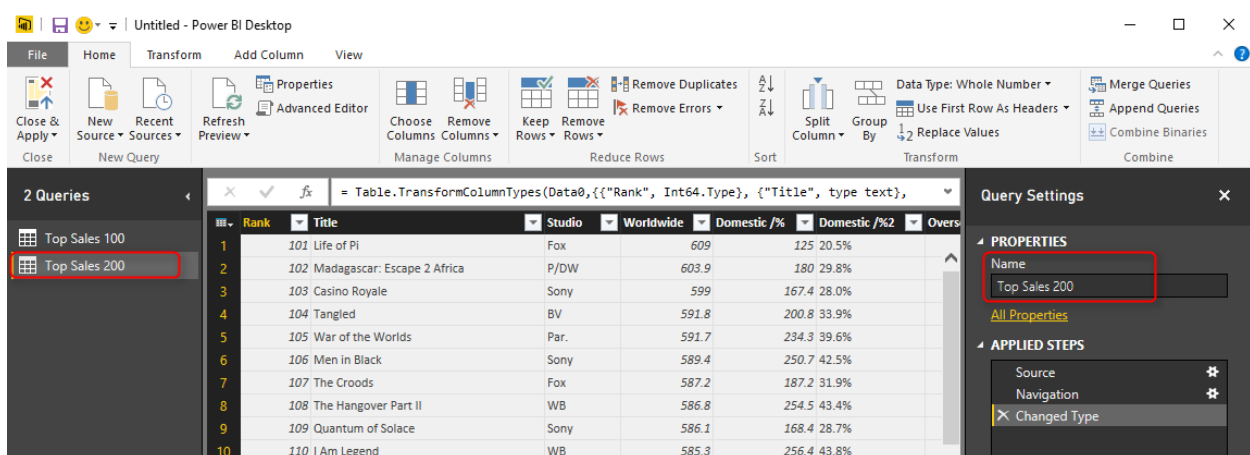
Rank	Title	Studio	Worldwide	Domestic / %	Overseas / %	Year^
1	<b>Avatar</b>	Fox	<b>\$2,788.0</b>	\$760.5 27.3%	\$2,027.5 72.7%	2009^

So Let's add the list of movies from 101 to 200 in best sellers;

In the existing Query Editor window go to New Source, and then choose From Web.  
Enter the URL as <http://www.boxofficemojo.com/alltime/world/?pagenum=2&p=.htm>



This will lead you to the top second 100 movies sold. Click on Table 0 in navigator window and then OK. in the Query Editor rename this query as Top Sales 200



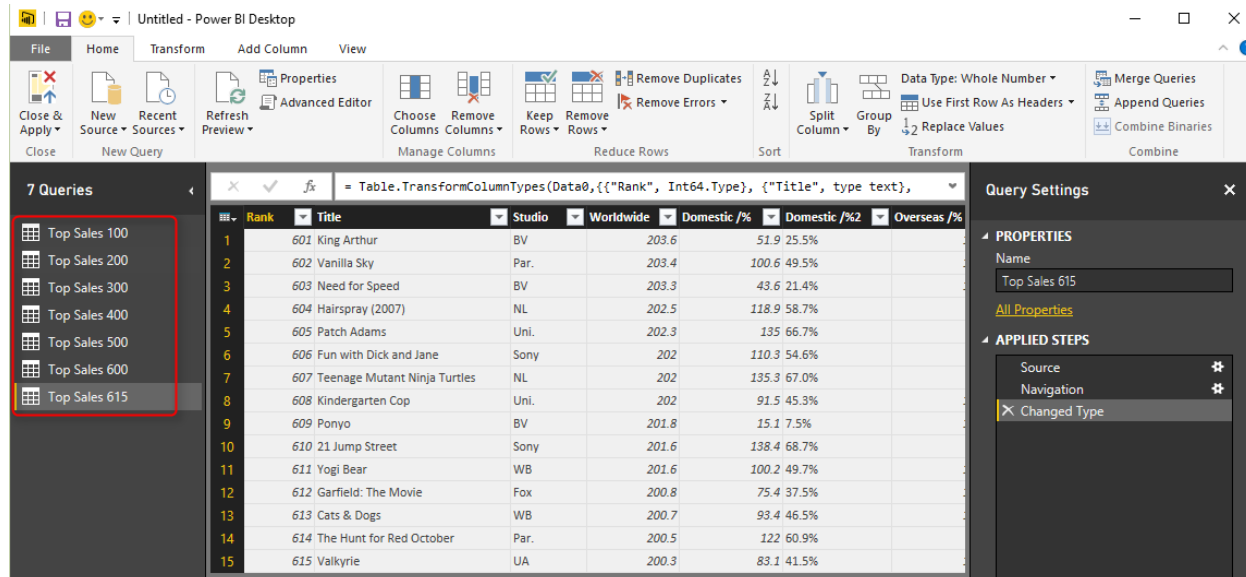
Follow this process for links below;

<http://www.boxofficemojo.com/alltime/world/?pagenum=3&p=.htm>

<http://www.boxofficemojo.com/alltime/world/?pagenum=4&p=.htm>

...

Bring data for all top 615 movies in Power Query



The screenshot shows the Power BI Desktop interface. On the left, a list of 7 queries is displayed, with 'Top Sales 615' selected. The main area shows a table with columns: Rank, Title, Studio, Worldwide, Domestic /%, Domestic /%2, and Overseas /%. The table contains 15 rows of movie data. The right pane shows the 'Query Settings' for 'Top Sales 615', with the 'APPLIED STEPS' list containing 'Source', 'Navigation', and 'Changed Type'.

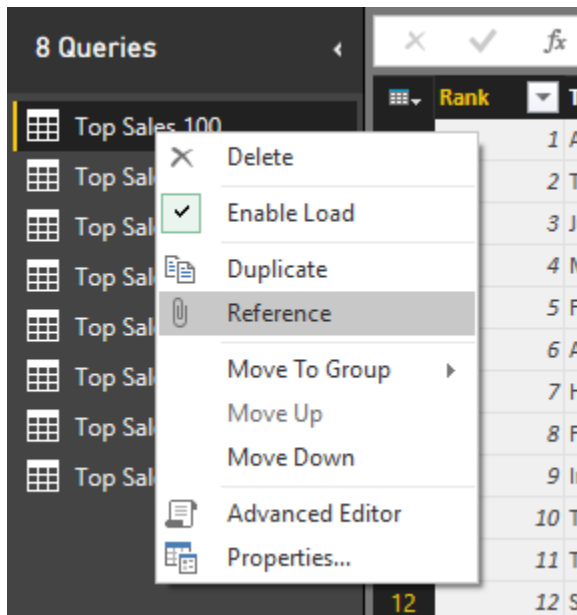
Rank	Title	Studio	Worldwide	Domestic /%	Domestic /%2	Overseas /%
1	601 King Arthur	BV	203.6	51.9	25.5%	
2	602 Vanilla Sky	Par.	203.4	100.6	49.5%	
3	603 Need for Speed	BV	203.3	43.6	21.4%	
4	604 Hairspray (2007)	NL	202.5	118.9	58.7%	
5	605 Patch Adams	Uni.	202.3	135	66.7%	
6	606 Fun with Dick and Jane	Sony	202	110.3	54.6%	
7	607 Teenage Mutant Ninja Turtles	NL	202	135.3	67.0%	
8	608 Kindergarten Cop	Uni.	202	91.5	45.3%	
9	609 Ponyo	BV	201.8	15.1	7.5%	
10	610 21 Jump Street	Sony	201.6	138.4	68.7%	
11	611 Yogi Bear	WB	201.6	100.2	49.7%	
12	612 Garfield: The Movie	Fox	200.8	75.4	37.5%	
13	613 Cats & Dogs	WB	200.7	93.4	46.5%	
14	614 The Hunt for Red October	Par.	200.5	122	60.9%	
15	615 Valkyrie	UA	200.3	83.1	41.5%	

As you see in query editor, all of these queries are separate from each other. Let's combine them all. In database and SQL world that can be done with UNION. Here in Power Query we can do Append Queries;

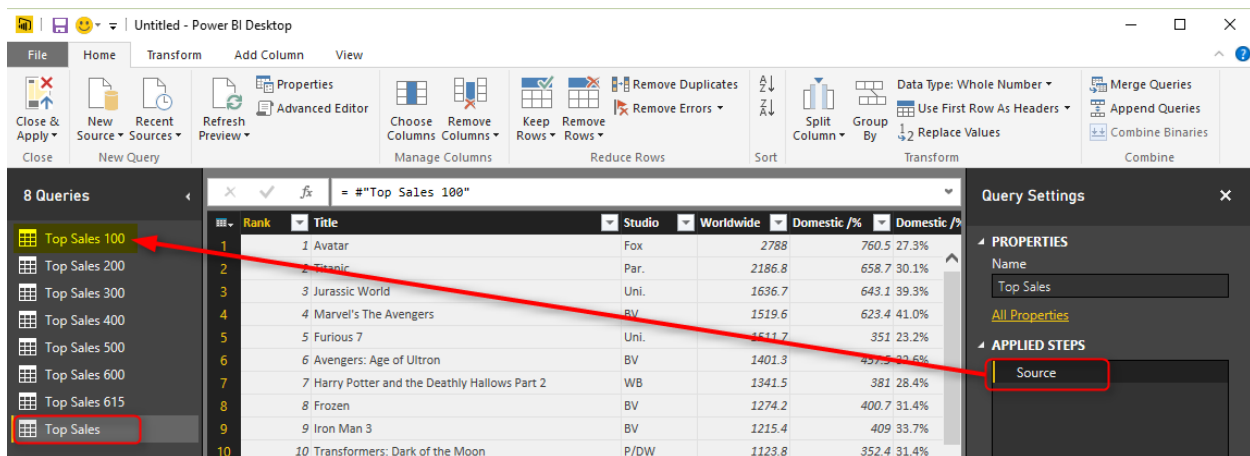
## Use a Query as a Reference

First Create a reference from Top Sales 100 (because for this example I want to keep that query as is);

Right click on Top Sales 100, and from the pop-up menu choose Reference



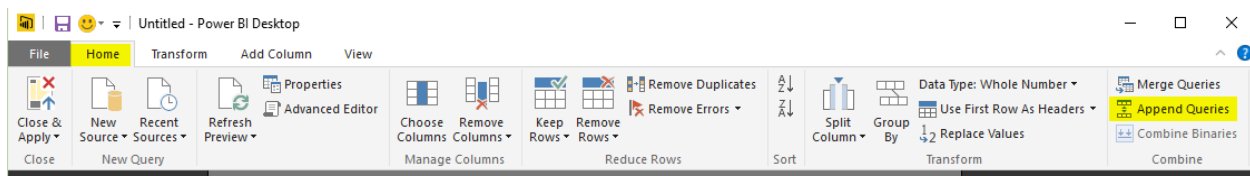
This will create a new query that uses Top Sales 100 as the source (or reference). Rename this new query to be just “Top Sales.”



## Append Queries

Now let's combine queries into this new query;

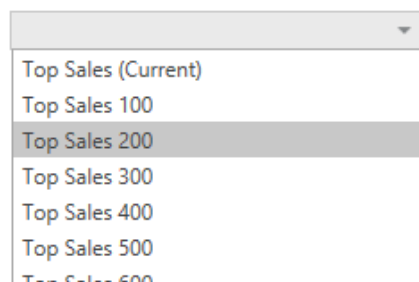
Click on Top Sales and then from the menu (Home) click on Append Queries



For append to work you need two queries; the first query is the query that you are on it (Top Sales), the second query name should be entered in the Append dialog box;

## Append


Select the table to append.



OK

Cancel

as you see in the screenshot above you can choose other queries. For append to work, best queries have to be in the same structure (number of columns, the order of columns, the data type of columns....). Choose Top Sales 200 in this window and click OK. This will create another step in the query setting named Appended Query. And the result set in the main pane (if you scroll down) will show you first top 200 movies sold.

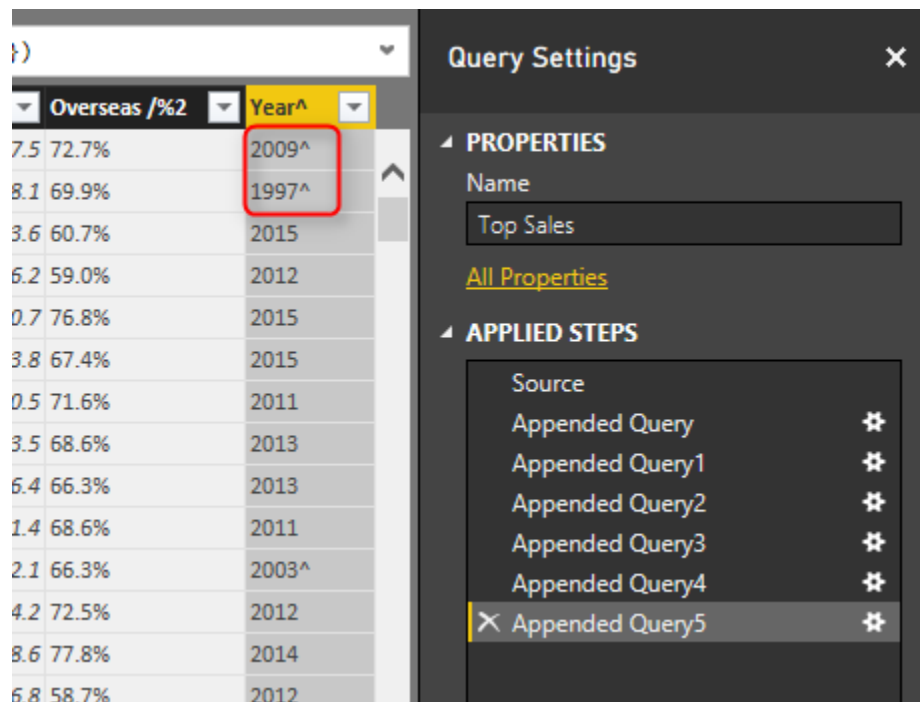


Rank	Title	Studio	Worldwide	Domestic /%	Domes
95	Iron Man 2	Par.	623.9	312.4 50.1	
96	Ratatouille	BV	623.7	206.4 33.1	
97	How to Train Your Dragon 2	Fox	618.9	177 28.1	
98	The Lost World: Jurassic Park	Uni.	618.6	229.1 37.1	
99	The Passion of the Christ	NM	611.9	370.8 60.1	
100	Mamma Mia!	Uni.	609.8	144.1 23.1	
101	Life of Pi	Fox	609	125 20.1	
102	Madagascar: Escape 2 Africa	P/DW	603.9	180 29.1	
103	Casino Royale	Sony	599	167.4 28.1	
104	Tangled	BV	591.8	200.8 33.1	
105	War of the Worlds	Par.	591.7	234.3 39.1	

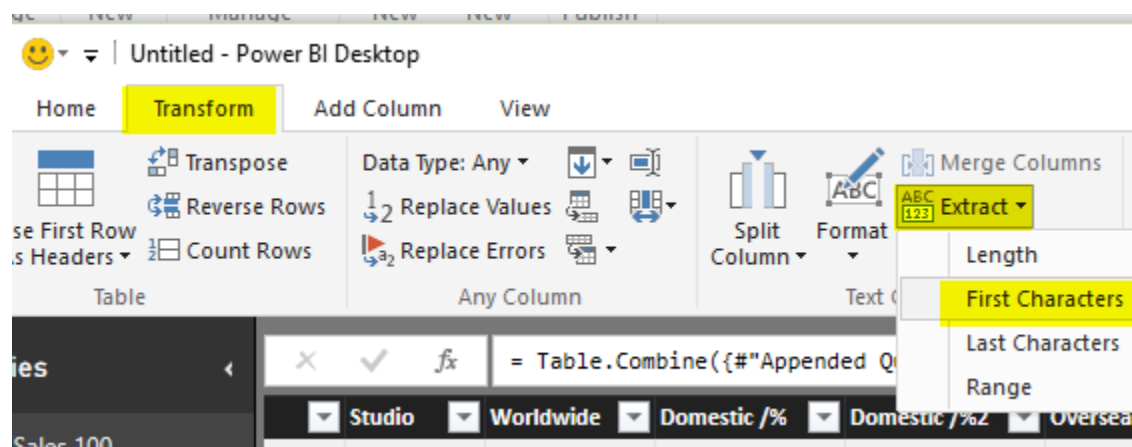
Continue this process to add all 615 top movies into Top Sales query.

## Extract First Characters

After doing this change Let's clean the Year column data; Year column has a special character in some values as below;



Click on Year Column, and then from Transform menu under Text Column click on Extract, and then choose First Characters



Enter 4 in the Extract First Characters dialog box (because the year isn't more than four characters). Then click on OK.

## Extract First Characters

Enter how many starting characters to keep.

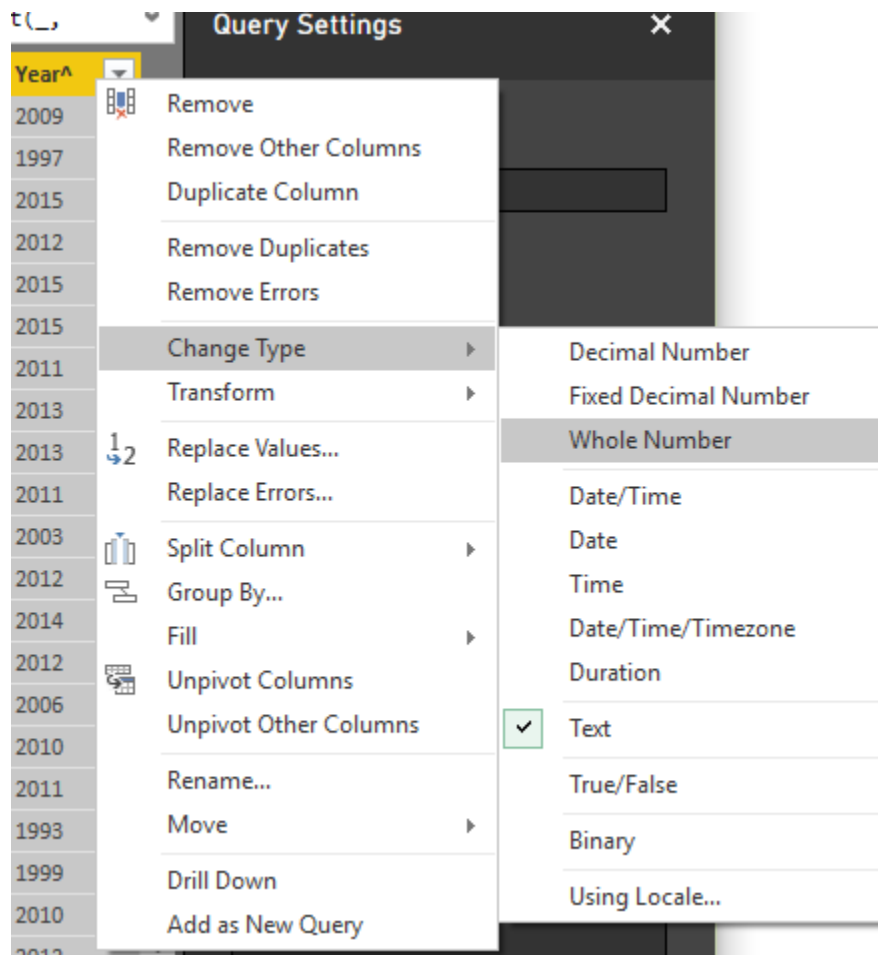
Count

OK

Cancel

You will see that year column is clean now without any extra characters. That was easy data transform. This option in the transformation menu (Extract First Characters) [has been added recently in Power BI Desktop](#).

You can even now change the data type of this column to the whole number. Right click on Year column and then under Change Type choose the Whole Number.

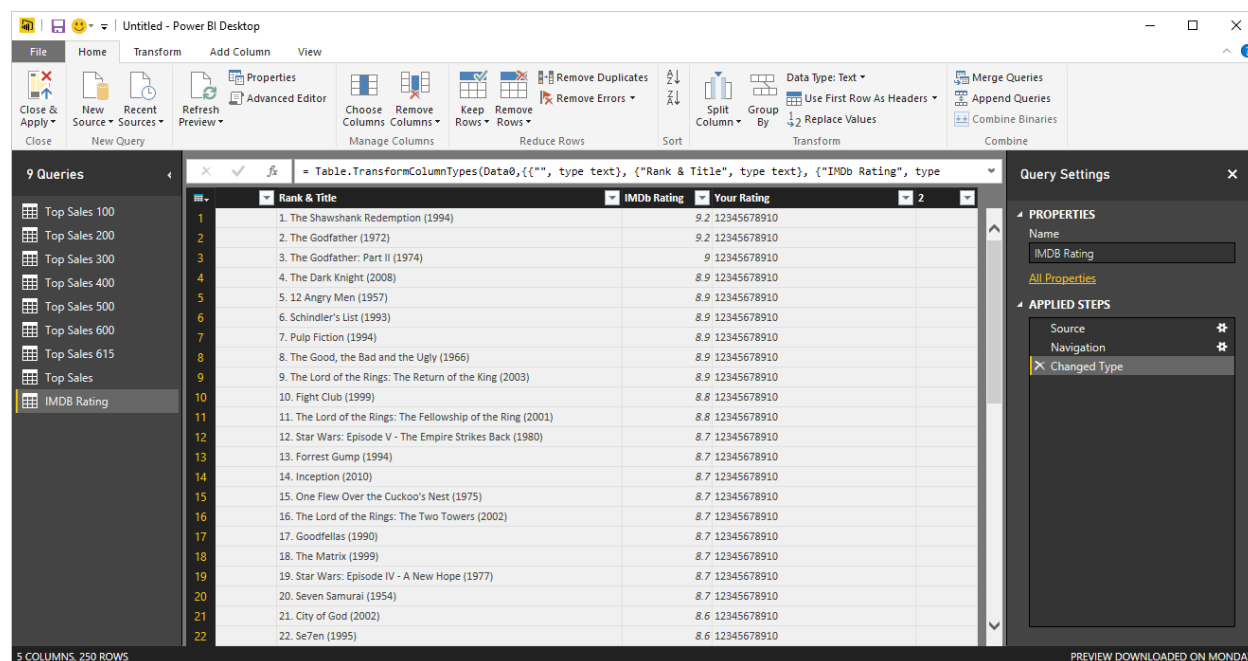




Great We've done enough with the first data set. Let's work on the second data set (IMDB user rating);

Go to Home Tab in Query Editor again, and Get data from the New Source and Web. Enter the URL as <http://www.imdb.com/chart/top>

In the Navigator window, Table 0 contains the data that we want, so load it with clicking on OK. the data loads into the Query Editor as the screenshot below illustrates

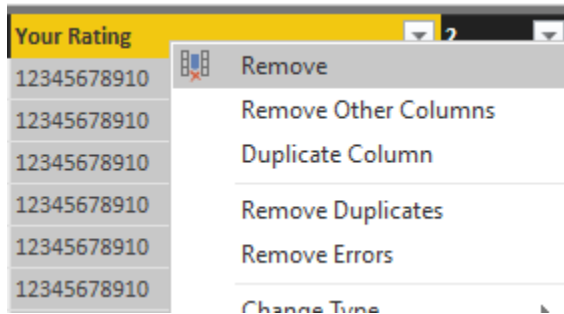


Rank & Title	IMDb Rating	Your Rating
1. The Shawshank Redemption (1994)	9.2	12345678910
2. The Godfather (1972)	9.2	12345678910
3. The Godfather: Part II (1974)	9	12345678910
4. The Dark Knight (2008)	8.9	12345678910
5. 12 Angry Men (1957)	8.9	12345678910
6. Schindler's List (1993)	8.9	12345678910
7. Pulp Fiction (1994)	8.9	12345678910
8. The Good, the Bad and the Ugly (1966)	8.9	12345678910
9. The Lord of the Rings: The Return of the King (2003)	8.9	12345678910
10. Fight Club (1999)	8.8	12345678910
11. The Lord of the Rings: The Fellowship of the Ring (2001)	8.8	12345678910
12. Star Wars: Episode V - The Empire Strikes Back (1980)	8.7	12345678910
13. Forrest Gump (1994)	8.7	12345678910
14. Inception (2010)	8.7	12345678910
15. One Flew Over the Cuckoo's Nest (1975)	8.7	12345678910
16. The Lord of the Rings: The Two Towers (2002)	8.7	12345678910
17. Goodfellas (1990)	8.7	12345678910
18. The Matrix (1999)	8.7	12345678910
19. Star Wars: Episode IV - A New Hope (1977)	8.7	12345678910
20. Seven Samurai (1954)	8.7	12345678910
21. City of God (2002)	8.6	12345678910
22. Se7en (1995)	8.6	12345678910

Rename the query to IMDB Rating.

## Remove Columns

You can see that there are three useless columns in the data set; the first column, and the last two columns. Remove these columns simply by clicking on them and then right click and Remove.

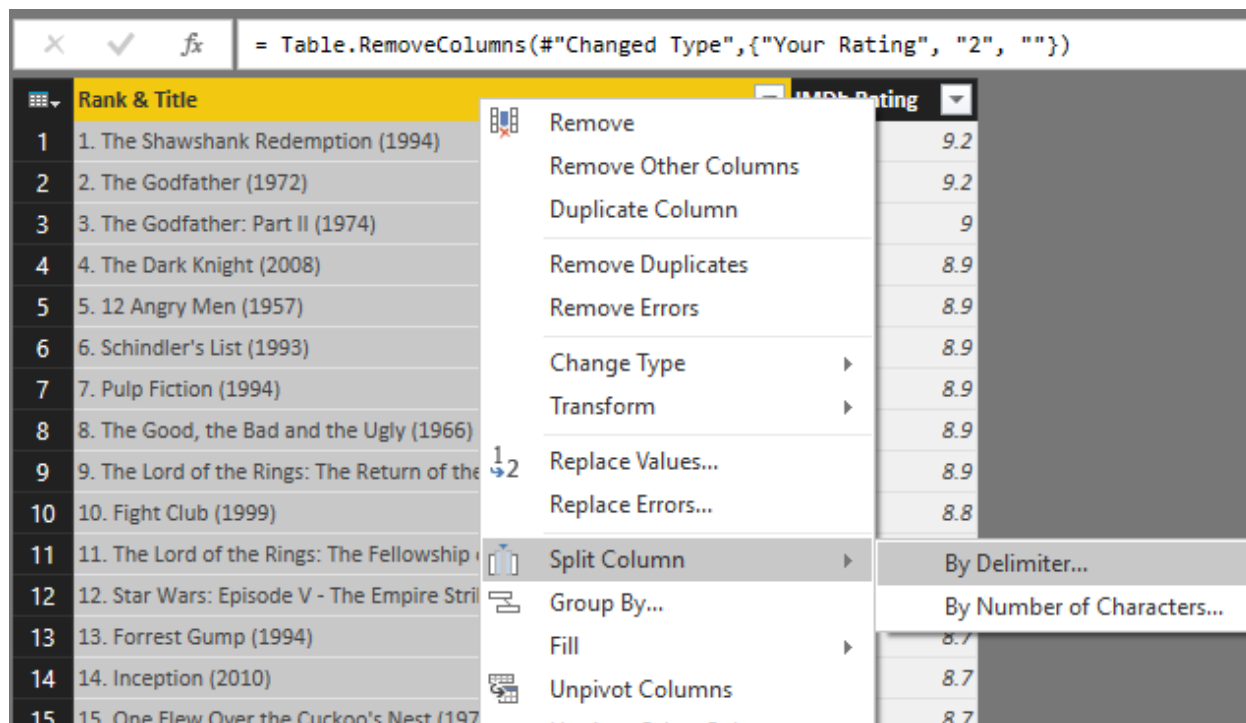


## Split Column

Now in the result set, we have two columns; Rank & Title, and IMDb Rating. Rank & Title is a combined column which contains rank, title, and year of the movie. Let's split these values;

	Rank & Title	IMDb Rating
1	1. The Shawshank Redemption (1994)	9.2
2	2. The Godfather (1972)	9.2
3	3. The Godfather: Part II (1974)	9
4	4. The Dark Knight (2008)	8.9
5	5. 12 Angry Men (1957)	8.9
6	6. Schindler's List (1993)	8.9
7	7. Pulp Fiction (1994)	8.9
8	8. The Good, the Bad and the Ugly (1966)	8.9
9	9. The Lord of the Rings: The Return of the King (2003)	8.9
10	10. Fight Club (1999)	8.8
11	11. The Lord of the Rings: The Fellowship of the Ring (2001)	8.8
12	12. Star Wars: Episode V - The Empire Strikes Back (1980)	8.7
13	13. Forrest Gump (1994)	8.7

A single dot separates rank (.). So we can use Split Column transformation to split it easily; Right click on Rank & Title column first. then Choose Split Column, and then By Delimiter



In the Split Column by the Delimiter dialog box, you can choose one of the common delimiters such as comma or color ... or you can use a custom delimiter. Set it to Custom, and enter a single dot (.) in the box underneath. You can also specify how the split works. The default option is At each occurrence of the delimiter. This default option might not be best for our case, because sometimes there might be a dot in the movie's title. So select the split method as At the left-most delimiter. This option will scan text from the left, and will stop splitting after finding the first delimiter.

## Split Column by Delimiter

Specify the delimiter used to split the text column.

Select or enter delimiter

--Custom--

.

Split

☒ At the left-most delimiter

☐ At the right-most delimiter

☐ At each occurrence of the delimiter

▸ Advanced options

OK

Cancel

after the split the result set would look like below;

	Rank & Title.1	Rank & Title.2	IMDb Rating
1	1	The Shawshank Redemption (1994)	9.2
2	2	The Godfather (1972)	9.2
3	3	The Godfather: Part II (1974)	9
4	4	The Dark Knight (2008)	8.9
5	5	12 Angry Men (1957)	8.9
6	6	Schindler's List (1993)	8.9
7	7	Pulp Fiction (1994)	8.9
8	8	The Good, the Bad and the Ugly (1966)	8.9
9	9	The Lord of the Rings: The Return of the King (2003)	8.9
10	10	Fight Club (1999)	8.8

Rename the Rank & Title.1 column to Rank.

Now Let's split title and year. Year value is surrounded between brackets, so we can use the same split column method, this time using open bracket as below;

## Split Column by Delimiter

Specify the delimiter used to split the text column.

Select or enter delimiter  
--Custom--  
(  
Split  
☐ At the left-most delimiter  
☒ At the right-most delimiter  
☐ At each occurrence of the delimiter

Advanced options

OK

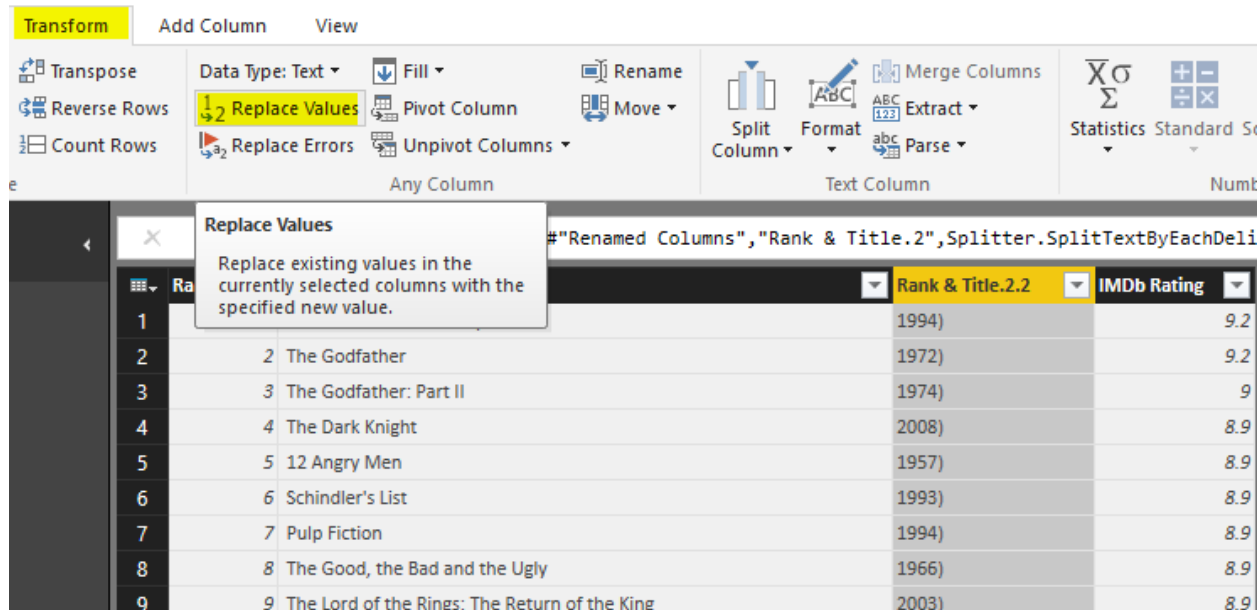
Cancel

The result set looks like below screenshot;

Rank	Rank & Title.2.1	Rank & Title.2.2	IMDb Rating
1	1 The Shawshank Redemption	1994)	9.2
2	2 The Godfather	1972)	9.2
3	3 The Godfather: Part II	1974)	9
4	4 The Dark Knight	2008)	8.9
5	5 12 Angry Men	1957)	8.9
6	6 Schindler's List	1993)	8.9
7	7 Pulp Fiction	1994)	8.9
8	8 The Good, the Bad and the Ugly	1966)	8.9
9	9 The Lord of the Rings: The Return of the King	2003)	8.9
10	10 Fight Club	1999)	8.8
11	11 The Lord of the Rings: The Fellowship of the Ring	2001)	8.8

## Replace Values

Rank & Title.2.2 column has the year value with an extra close bracket. Click on this column and then from Transform menu under Any Column click on Replace Values



**Transform** | Add Column | View

Transpose | Reverse Rows | Count Rows | Data Type: Text | Fill | Pivot Column | Unpivot Columns | Rename | Move | Split Column | Format | Merge Columns | Extract | Parse | Statistics | Standard S

**Replace Values**

Replace existing values in the currently selected columns with the specified new value.

Rank	Rank & Title.2.2	IMDb Rating
1	1994)	9.2
2	2 The Godfather	9.2
3	3 The Godfather: Part II	9
4	4 The Dark Knight	8.9
5	5 12 Angry Men	8.9
6	6 Schindler's List	8.9
7	7 Pulp Fiction	8.9
8	8 The Good, the Bad and the Ugly	8.9
9	9 The Lord of the Rings: The Return of the King	8.9

Replace close bracket with an empty string as below;

## Replace Values

Replace one value with another in the selected columns.

Value To Find

Replace With

☐ Match entire cell contents

OK

Cancel

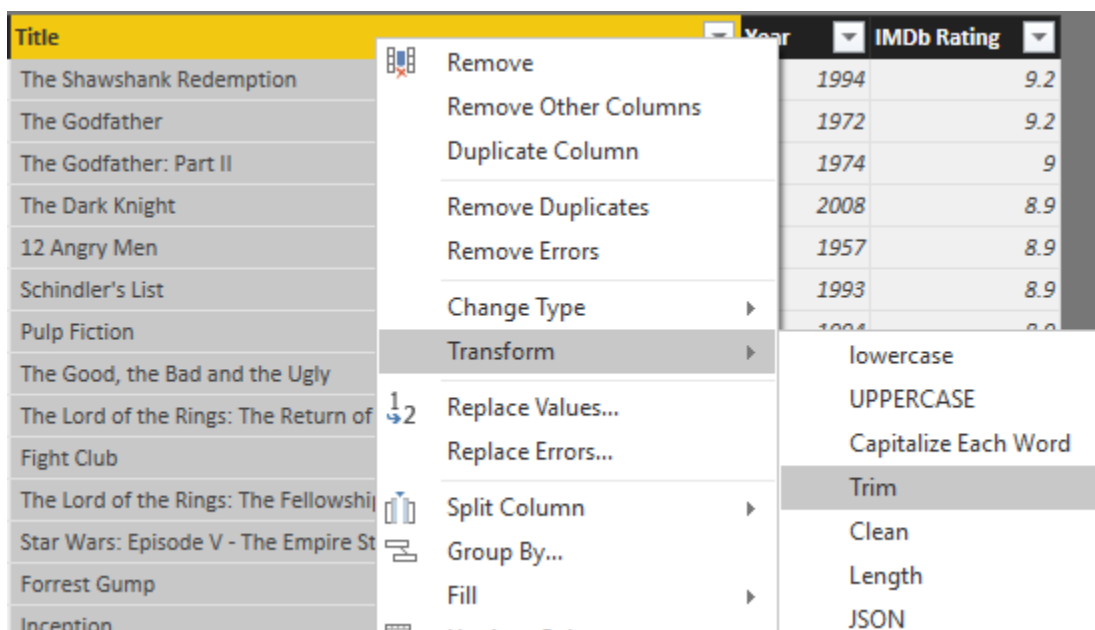
Result set would have the close bracket removed. rename the column to Year, and change its data type to Whole number (change data type with right click on the column)

Rank	Rank & Title.2.1	Year	IMDb Rating
1	1 The Shawshank Redemption	1994	9.2
2	2 The Godfather	1972	9.2
3	3 The Godfather: Part II	1974	9
4	4 The Dark Knight	2008	8.9
5	5 12 Angry Men	1957	8.9
6	6 Schindler's List	1993	8.9

## Trim

Also, rename the Rank & Title.2.1 column to Title. Because this column might have extra spaces at the beginning and end of values (as the result of split column steps), let's remove extra spaces;

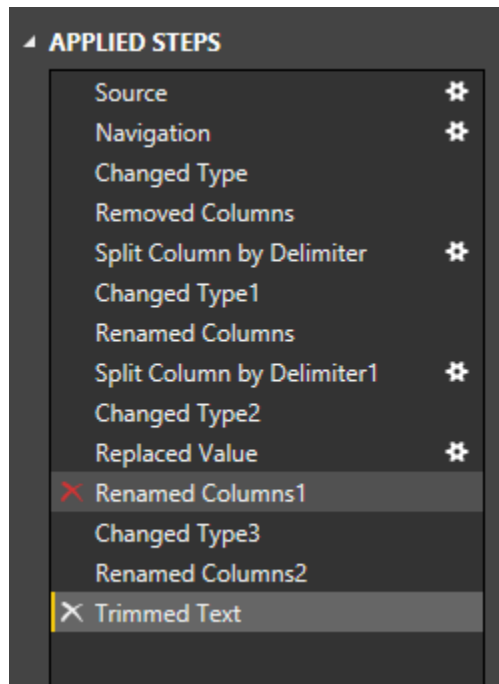
Right click on this column and then under transform choose Trim. This will remove all heading and trailing spaces from values in this column.



Awesome, our work with this data set has been finished as well.

## Applied Steps

One of the most useful sections of the Query Editor window is Applied Steps in the Query Settings Pane. This section of the Query Editor window is very useful for debugging and tracking steps and changes. You can see all the steps that you've applied on the current data set in this pane. And this is not all of it! You can click on a step, and the main pane will show you the data at that step! Such an awesome way of keeping track of steps.



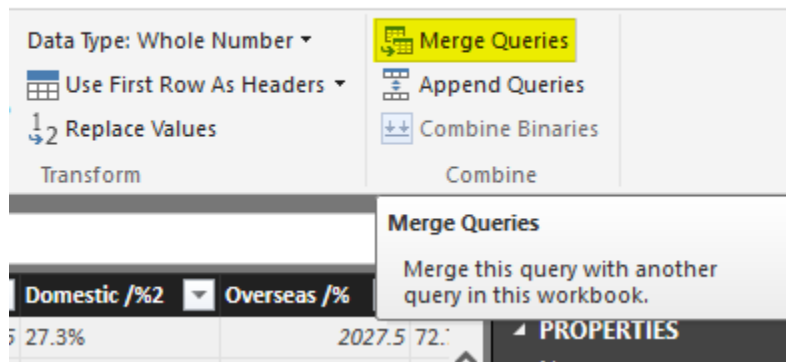
You can even remove a step, or you can change settings of a step with clicking on remove icon (on the left side of step) or setting icon (on the right side of the step, but only for steps that settings apply to them).

## Final Merge

We've prepared both data sets for a final merge together to see how best-selling movies are among top user rated films. So we are one step away from this result. We have to merge these two data sets or Join them in another word.

Click on Top Sales query and create a reference of it, name the new query as Merge Result. Now click on Merge Result, and then from Home tab, under Combine choose Merge Queries





Merge Queries is equivalent to Join in SQL or database terminology.

Merging queries requires two queries; the first query is the query that you are currently on it (Merge Result), and you can choose the second query in the Merge dialog box. Choose the second query as IMDB Rating. Now select joining keys as Title (you can also choose multiple joining columns with pressing ctrl keyboard key). Set also join kind to Left outer join (this will only select all records from the first query with matched rows of that from the second query)

## Merge

Select a table and matching columns to create a merged table.


Rank	Title	Studio	Worldwide	Domestic /%	Domestic /%2	Overseas /%	Overseas /%
1	Avatar	Fox	2788	760.5	27.3%	2027.5	72.7%
2	Titanic	Par.	2186.8	658.7	30.1%	1528.1	69.9%
3	Jurassic World	Uni.	1636.7	643.1	39.3%	993.6	60.7%
4	Marvel's The Avengers	BV	1519.6	623.4	41.0%	896.2	59.0%
5	Furious 7	Uni.	1511.7	351	23.2%	1160.7	76.8%

IMDB Rating

Rank	Title	Year	IMDb Rating
1	The Shawshank Redemption	1994	9.2
2	The Godfather	1972	9.2
3	The Godfather: Part II	1974	9
4	The Dark Knight	2008	8.9
5	12 Angry Men	1957	8.9

Join Kind

Inner (only matching rows)

 The selection has matched 58 out of the first 615 rows.

OK

Cancel

Notice in the screenshot above that merge dialog mentioned only 58 records out of 615 movies matched! It means only 58 of best seller movies are among top user rated list! Such a pity. The screenshot showed only Inner Join result, but you choose Left Outer and then click on OK to look at the data;

Joining experience in Power Query is a bit different from database tables. As a result of the join, you will get the first table with a new column for the new table. This new column holds table values which need to be expanded. If you click on the column header icon, you can choose which columns of the nested table you want to expand.

fx = Table.NestedJoin(Source,{"Title"},#"IMDb Rating",{"Title"},"NewColumn",JoinKind.Inner)

	Studio	Worldwide	Domestic /%	Domestic /%2	Overseas /%	Overseas /%2	Year^	NewColumn
Return of the King	WB	1004.6	534.9	53.2%				
Fellowship of the Ring	NL	1119.9	377.8	33.7%				
	Par.	677.9	330.3	48.7%				
	WB	1084.9	448.1	41.3%				
	WB	825.5	292.6	35.4%				
Two Towers	BV	1063.2	415	39.0%				
	Uni.	1029.2	402.5	39.1%				
	WB	463.5	171.5	37.0%				
	BV	987.5	422.8	42.8%				
	Par.	675	188	27.9%				
	DW	481.8	216.5	44.9%	265.3	55.1%	1998	Table
	BV	936.7	380.8	40.7%	555.9	59.3%	2003	Table
	Par.	389.9	248.2	63.6%	141.8	36.4%	1981	Table
y	TriS	519.8	204.8	39.4%	315	60.6%	1991	Table

Search Columns to Expand

☒ Expand ☐ Aggregate

☒ (Select All Columns)

☒ Rank

☒ Title

☒ Year

☒ IMDb Rating

☒ Use original column name as prefix

OK Cancel

Let's keep all columns and click OK. You can now see some movies that are among best sellers but not in top 250 users rated list of IMDB; There are movies name such as Iron Man 3, Skyfall, Furious 7 and list goes on. Play with that yourself to see what you explore!

Rank	Title	Studio	Worldwide	NewColumn.Rank	NewColumn.Title	NewColumn.Year	NewColumn.IMDb Rating
1	23 The Dark Knight	WB	1004.6	4 The Dark Knight		2008	8.9
2	11 The Lord of the Rings: The Return of the King	NL	1119.9	9 The Lord of the Rings: The Return of the King		2003	8.9
3	41 The Lord of the Rings: The Fellowship of the Ring	NL	871.5	11 The Lord of the Rings: The Fellowship of the Ring		2001	8.8
4	80 Forrest Gump	Par.	677.9	13 Forrest Gump		1994	8.7
5	14 The Dark Knight Rises	WB	1084.9	61 The Dark Knight Rises		2012	8.4
6	46 Inception	WB	825.5	14 Inception		2010	8.7
7	16 Toy Story 3	BV	1063.2	78 Toy Story 3		2010	8.3
8	34 The Lord of the Rings: The Two Towers	NL	926	16 The Lord of the Rings: The Two Towers		2002	8.7
9	24 The Lion King	BV	987.5	54 The Lion King		1994	8.4
10	81 Interstellar	Par.	675	29 Interstellar		2014	8.6
11	76 Inside Out	BV	702.5	53 Inside Out		2015	8.4
12	68 Up	BV	731.3	114 Up		2009	8.2
13	82 The Sixth Sense	BV	672.8	159 The Sixth Sense		1999	8.1
14	32 Finding Nemo	BV	936.7	161 Finding Nemo		2003	8.1
15	18 Jurassic Park	Uni.	1029.2	203 Jurassic Park		1993	8
16	56 Guardians of the Galaxy	BV	774.2	210 Guardians of the Galaxy		2014	8
17	87 Pirates of the Caribbean: The Curse of the Black Pearl	BV	654.3	228 Pirates of the Caribbean: The Curse of the Black Pearl		2003	8
18	63 X-Men: Days of Future Past	Fox	748.1	240 X-Men: Days of Future Past		2014	8
19	1 Avatar	Fox	2788			null	null
20	2 Titanic	Par.	2186.8			null	null
21	3 Jurassic World	Uni.	1636.7			null	null
22	4 Marvel's The Avengers	BV	1519.6			null	null
23	5 Furious 7	Uni.	1511.7			null	null
24	6 Avengers: Age of Ultron	BV	1401.3			null	null
25	7 Harry Potter and the Deathly Hallows Part 2	WB	1341.5			null	null
26	8 Frozen	BV	1274.2			null	null
27	9 Iron Man 3	BV	1215.4			null	null
28	10 Transformers: Dark of the Moon	P/DW	1123.8			null	null
29	12 Skyfall	Sony	1108.6			null	null
30	13 Transformers: Age of Extinction	Par.	1104			null	null
31	15 Pirates of the Caribbean: Dead Man's Chest	BV	1066.2			null	null
32	17 Pirates of the Caribbean: On Stranger Tides	BV	1045.7			null	null
33	19 Star Wars: Episode I - The Phantom Menace	Fox	1027			null	null
34	20 Alice in Wonderland (2010)	BV	1025.5			null	null
35	21 The Hobbit: An Unexpected Journey	WB	1021.1			null	null

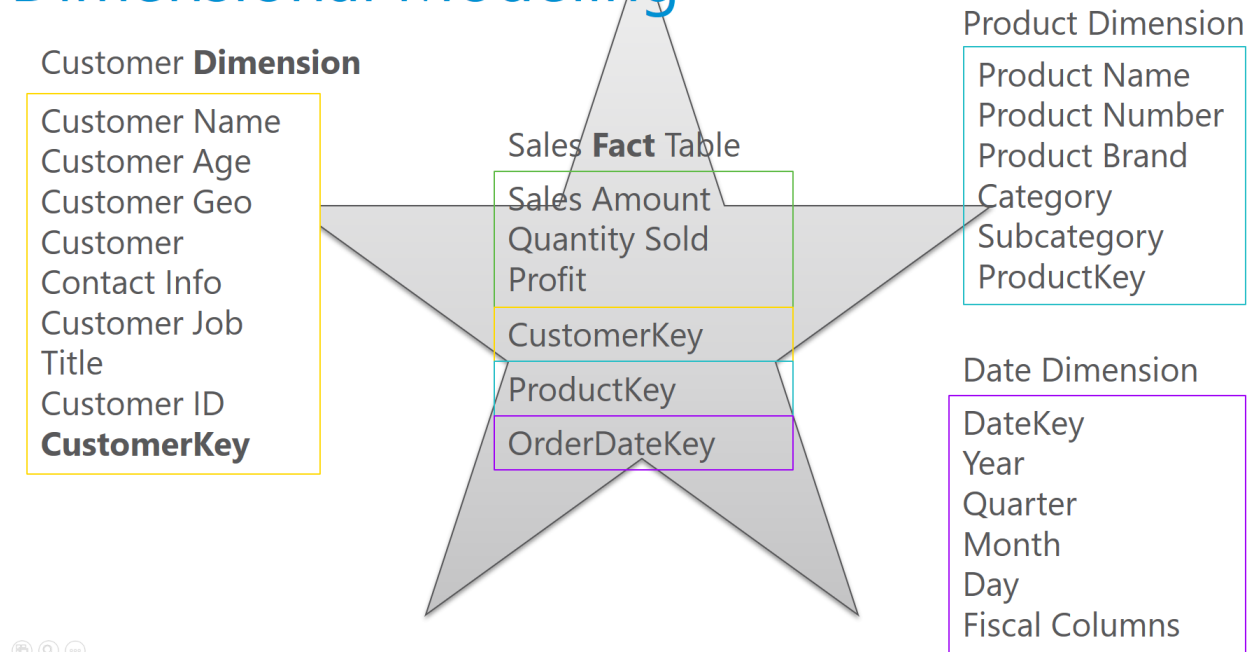
## Summary

In this section, you've learned the basics of Power Query through an example. You've seen how Power Query can analyze tables in a web page and load it into query editor. You've experienced Query Editor, and you've learned how to apply some transformations. You've learned that transformations such as a split column, replace values, change the data type, and extract part of the text are easy transformations that can be simply done through Power Query editor. In next sections, I will explain different types of data sources that Power Query or Power BI can work with through the Get Data Experience. You will see that Power Query and Power BI can get data from text files such as CSV, Text as well as database connections such as MySQL, Oracle, and SQL Server, it can also bring data from on-premises data stores as well as cloud Azure-based services.

# Data Preparation; First and Foremost Important Task in Power BI

Posted by [Reza Rad](#) on Dec 21, 2016

## Dimensional Modeling

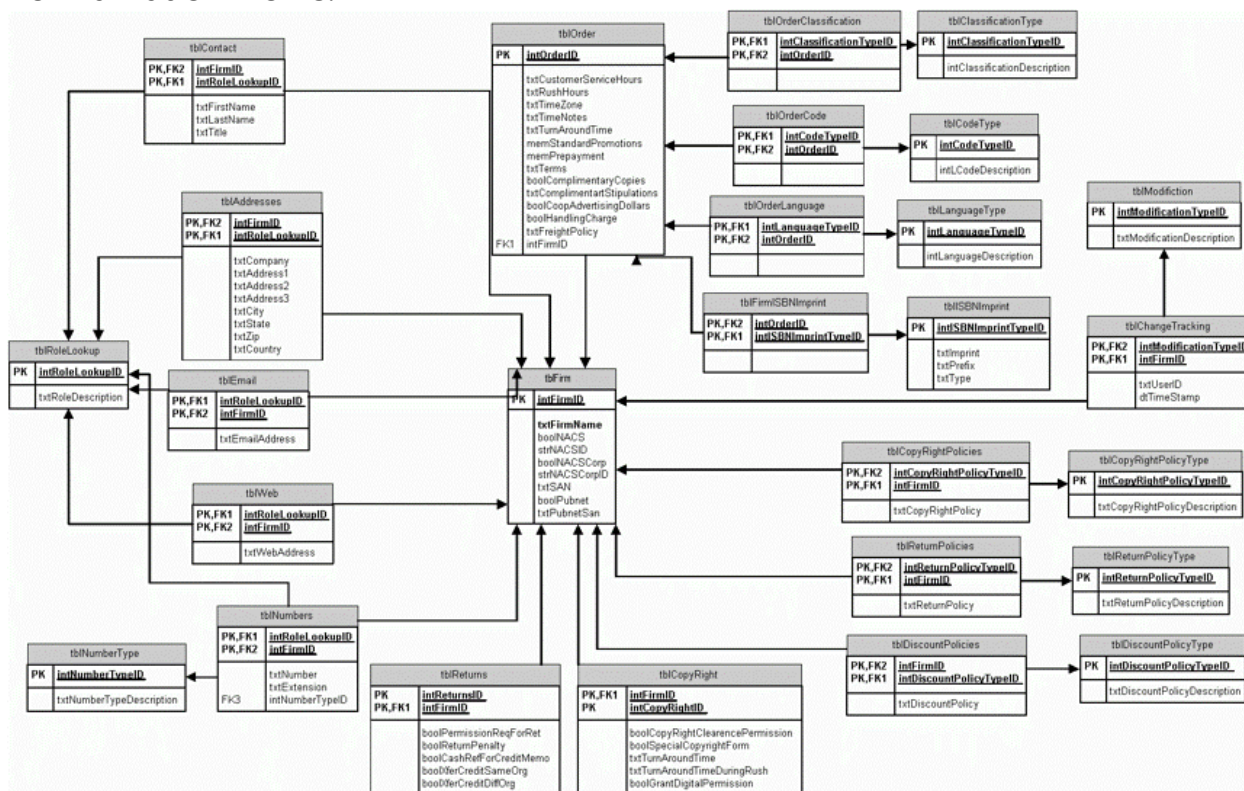


I've talked about Data Preparation many times in conferences such as PASS Summit, BA Conference, and many other conferences. Every time I talk about this I realize how much more I need to explain this. Data Preparation tips are basic but very important. In my opinion as someone who worked with BI systems for more than 15 years, this is the most important task in building in BI system. In this post, I'll explain why data preparation is necessary and what are five basic steps you need to be aware of when building a data model with Power BI (or any other BI tools). This post is conceptual, and you can apply these rules to any tools.

## Why Data Preparation?

All of us have seen many data models like the screenshot below. Transactional databases have the nature of many tables and the relationship between tables.

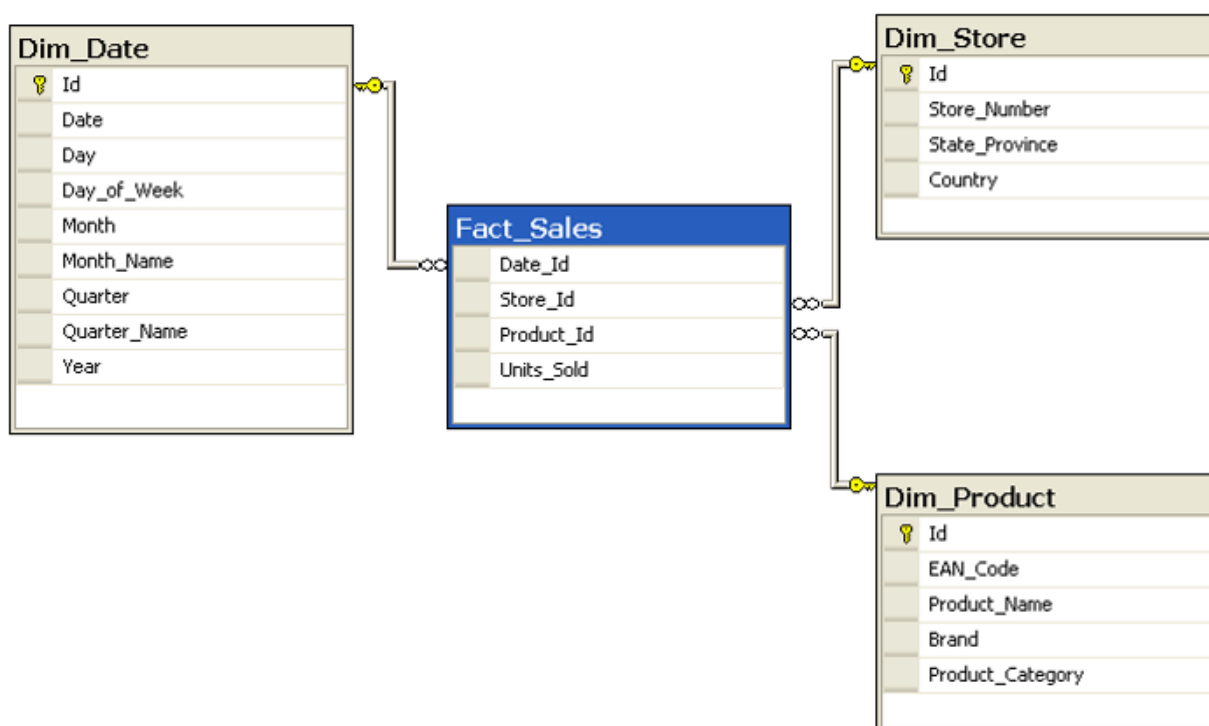
Transactional databases are built for CRUD operations (Create, Retrieve, Update, Delete rows). Because of this single purpose, transactional databases are built in Normalized way, to reduce redundancy and increase the consistency of the data. For example, there should be one table for Product Category with a Key related to a Product Table, because then whenever a product category name changes there is only one record to update and all products related to that will be updated automatically because they are just using the key. There are books to read if you are interested in how database normalization works.



I'm not going to talk about how to build these databases. In fact for building a data model for a BI system you need to avoid this type of modeling! This model works perfectly for the transactional database (when there are systems and operators do data entry and modifications). However, this model is not good for a BI system. There are several reasons for that, here are the two most important reasons;

1. The model is hard to understand for a Report User
2. Too many tables and many relationships between tables makes a reporting query (that might use 20 of these tables at once) very slow and not efficient.

You never want to wait for hours for a report to respond. The response time of reports should be fast. You also never want your report users (of self-service report users) to understand the schema above. It is sometimes even hard for a database developer to understand how this works in the first few hours! You need to make your model simpler, with a few tables and relationships. Your first and the most important job as a BI developer should be transforming above schema to something like below;



This model is far simpler and faster. There is one table that keeps sales information (Fact\_Sales), and a few other tables which keep descriptions (such as Product description, name, brand, and category). There is one relationship that connects the table in the middle (Fact) to tables around (Dimensions). This is the best model to work within a BI system. This model is called **Star**

**Schema.** Building a star schema or dimensional modeling is your most important task as a BI developer.

## How to Design a Star Schema?

To build a star schema for your data model, I strongly suggest you take one step at a time. What I mean by that is choosing one or few use cases and start building the model for that. For example; instead of building a data model for the whole Dynamics AX or CRM which might have more than thousands of tables, choose Sales side of it, or Purchasing, or GL. After choosing one subject area (for example Sales), then start building the model for it considering what is required for the analysis.

### Fact Table

Fact tables are tables that are holding numeric and additive data normally. For example, quantity sold, or sales amount, or discount, or cost, or things like that. These values are numeric and can be aggregated. Here is an example of a fact table for sales;

## Sales **Fact** Table

---

Sales Amount  
Quantity Sold  
Profit

---

### Dimension Table

Any descriptive information will be kept in Dimension tables. For example; customer name, customer age, customer geoinformation, customer contact



information, customer job, customer id, and any other customer related information will be kept in a table named Customer Dimension.

## Customer **Dimension**

Customer Name  
Customer Age  
Customer Geo  
Customer  
Contact Info  
Customer Job  
Title  
Customer ID  
**CustomerKey**

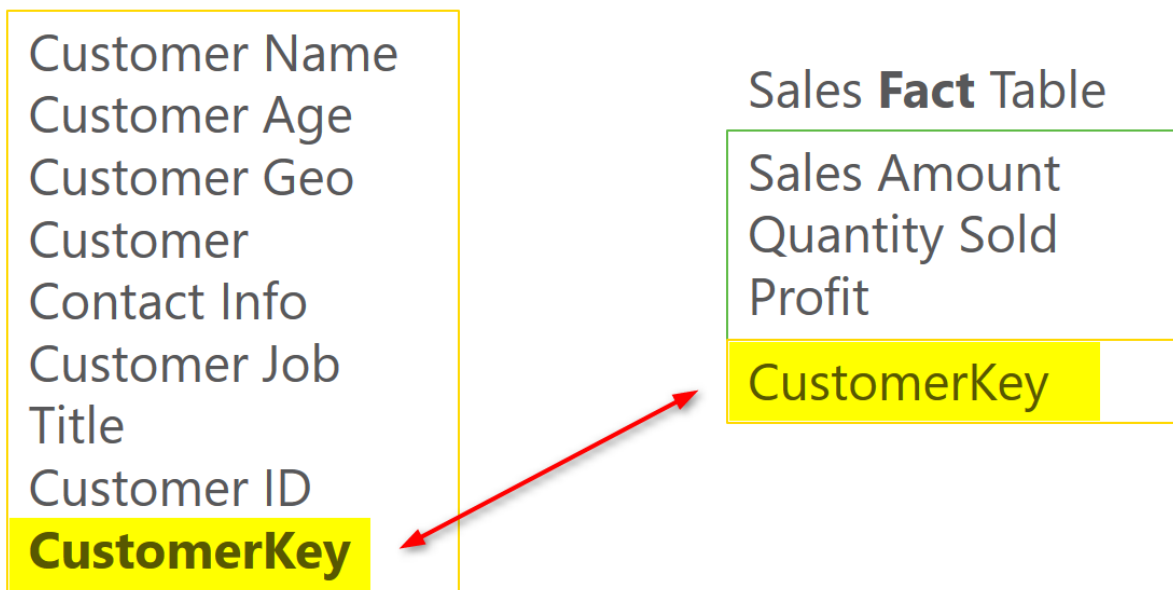
each dimension table should contain a key column. This column should be numeric (integer or big integer depends on the size of dimension) which is auto increment (Identity in SQL Server terminology). This column should be unique per each row in the dimension table. This column will be the primary key of this table and will be used in fact table as a relationship. This column

SHOULDN'T be the ID of the source system. There are several reasons for why. This column is called **Surrogate Key**. Here are a few reasons why you need to have the surrogate key:

- Codes (or IDs) in source system might be Text, not Integer.
- Short Int, Int, or Big Int are the best data types for the surrogate key because these are values which will be used in the fact table. Because fact table is the largest table in the dataset, it is important to keep it in the smallest size possible (using Int data types for dimension foreign keys is one of the main ways of doing that).
- Codes (or IDs) might be recycled.
- You might want to keep track of changes (Slowly Changing Dimension), so one ID or Code might be used for multiple rows.
- ...

Surrogate key of the dimension should be used in the fact table as a foreign key. Here is an example;

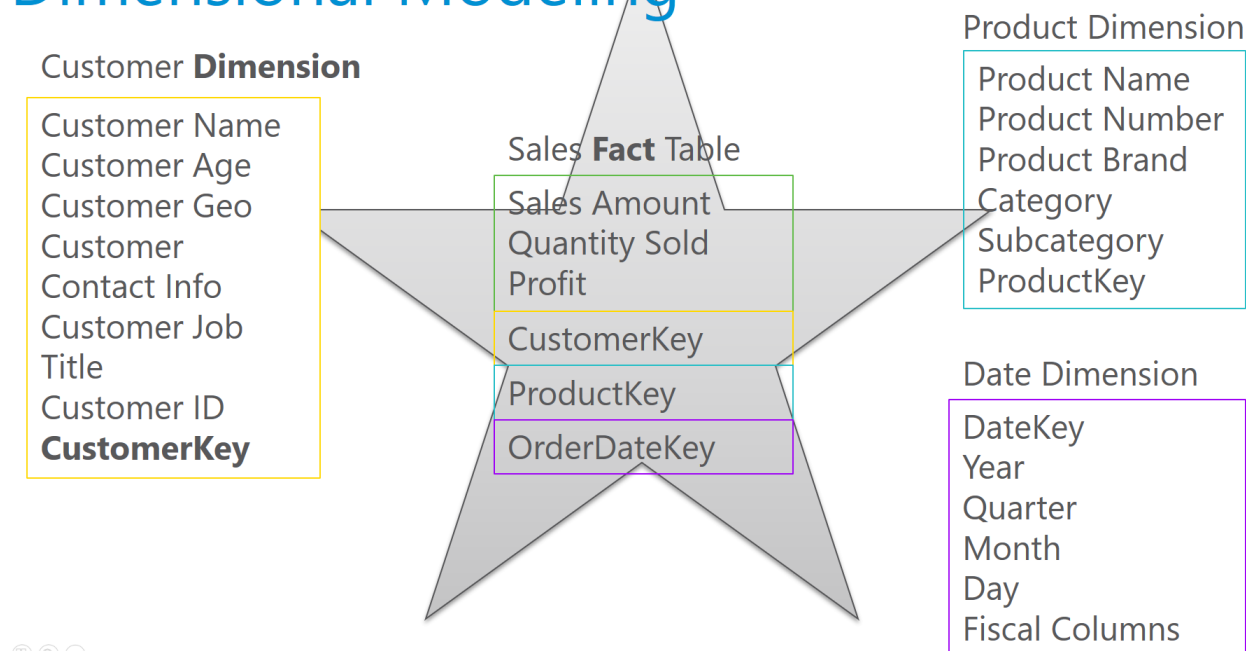
### Customer **Dimension**



Other dimensions should also be added in the same way. In the example below a Date Dimension and Product, Dimension is also created. You can easily see in the screenshot below that why it is called star schema; Fact table

is in the middle, and all other dimensions are around with one relationship from fact table to other dimensions.

## Dimensional Modeling



### Design Tips

Building star schema or dimensional modeling is something that you have to read books about it to get it right. I will write some more blog posts and explain these principles more in details. However it would be great to leave some tips here for you to get things started towards better modeling. These tips are simple but easy to overlook. A number of BI solutions that I have seen suffer from not obeying these rules are countless. These are rules that if you do not follow, you will be soon far away from proper data modeling, and you have to spend ten times more to build your model proper from the beginning. Here are tips:

#### Tip 1: DO NOT add tables/files as is

Why?

Tables can be joined together to create more flatten and simpler structure.

Solution: DO create a flatten structure for tables (especially dimensions)

### **Tip 2: DO NOT flatten your FACT table**

Why?

Fact table is the largest entities in your model. Flattening them will make them even larger!

Solution: DO create relationships to dimension tables

### **Tip 3: DO NOT leave naming as is**

Why?

Names such as std\_id, or dimStudent are confusing for users.

Solution: DO set naming of your tables and columns for the end user

### **Tip 4: DO NOT leave data types as is**

Why?

Some data types are spending memory (and CPU) like decimals. Appropriate data types are also helpful for engines such as Q&A in Power BI which is working based on the data model.

Solution: DO set proper data types based on the data in each field

### **Tip 5: DO NOT load the whole data if you don't require it**

Why?

Filtering part of the data before loading it into memory is cost and performance effective.

Solution: DO Filter part of the data that is not required.

## More to Come

This was just a very quick introduction to data preparation with some tips. This is the beginning of blog post series I will write in the future about principles of dimensional modeling and how to use them in Power BI or any other BI tools. You can build a BI solution without using these concepts and principles, but your BI system will slow down, and users will suffer from using it after few months, I'll give you my word on it. Stay tuned for future posts on this subject.

## Part II: Get Data

# Power BI Get Data From Excel: Everything You Need to Know

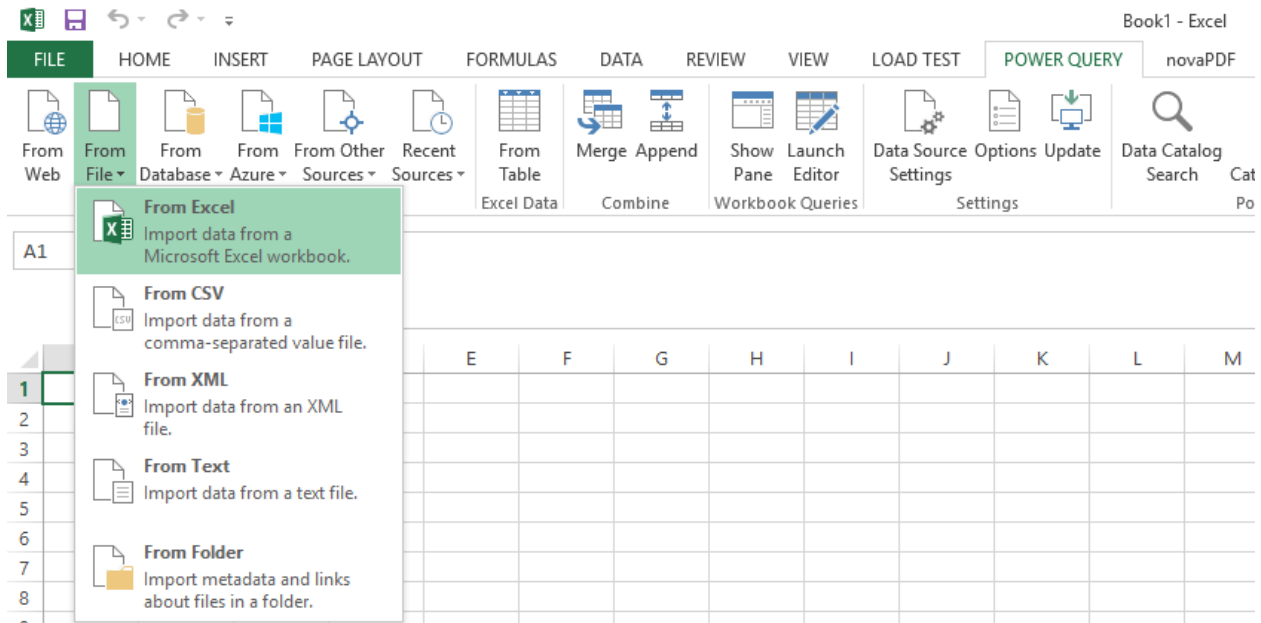
Posted by [Reza Rad](#) on Sep 2, 2015



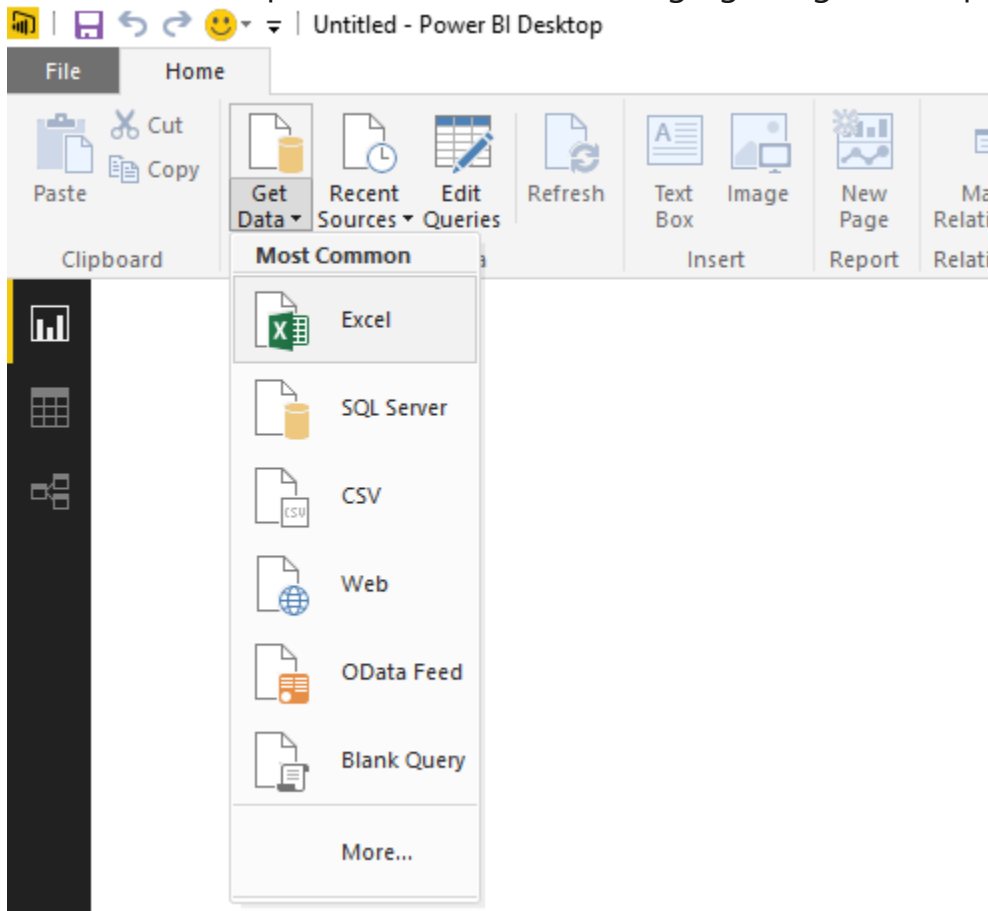
In the [Previous section](#), you learned about Power Query through an example of data mash-up of movies. Also prior than that you've learned about [Power BI](#) and its components in [Power BI online book from rookie to rockstar](#). In this section, I would like to start an exploration of different data sources in Power BI, and I want to start that with an Excel source. Excel source seems to be an easy one, but on the other hand, it is one of the most common sources of the data. In this section, I want to share some tips about Excel data source, and then I show an example of working with Olympic data source in an Excel file.

## Excel Data Source

Power Query or Power BI can connect to many data sources, one of the supported data sources in Excel. Power Query for Excel get data from Excel in this way:



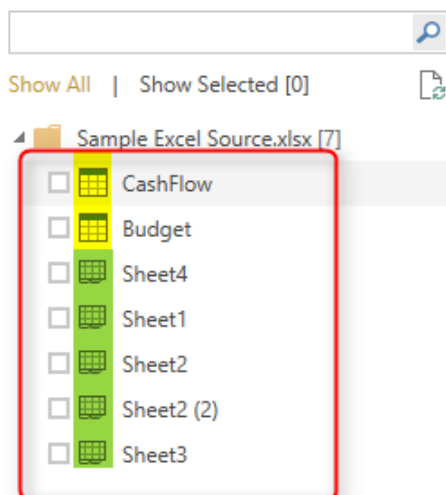
Power BI Desktop connects to Excel through getting Data experience





For getting data from excel you need to specify the path of the file. After specifying the file, Power Query will analyze contents of the file and distinguish all sheets and tables in the file and list them in the Navigator dialog box as a preview;

## Navigator



## CashFlow

Period	Base	Down	Up	Cash Flow
Start	null	0	5000	5000
Jan	5000	503	0	-503
Feb	4497	1670	0	-1670
Mar	2827	0	4802	4802
Apr	7629	1198	0	-1198
May	6431	3526	0	-3526
Jun	2905	0	1826	1826
Jul	4731	2284	0	-2284
Aug	2447	0	3250	3250
Sep	5697	1780	0	-1780
Oct	3917	0	2667	2667
Nov	6584	0	1500	1500
Dec	8084	0	2475	2475
End	10559	null	null	null

As you see in the Navigator dialog box screenshot above, Power Query will distinguish Tables and Sheets and show the appropriate icon for any of these types (note to the highlighted section in the screenshot above).

## Loading Excel Tables into Power Query

If you have data in an Excel Table then it can be easily detected and picked by Power Query as you see in the screenshot above, CashFlow is a table in Excel. The screenshot below shows the Excel table for CashFlow;

	A	B	C	D	E
1	Period	Base	Down	Up	Cash Flow
2	Start		\$0	\$5,000	\$5,000
3	Jan	\$5,000	\$503	\$0	-\$503
4	Feb	\$4,497	\$1,670	\$0	-\$1,670
5	Mar	\$2,827	\$0	\$4,802	\$4,802
6	Apr	\$7,629	\$1,198	\$0	-\$1,198
7	May	\$6,431	\$3,526	\$0	-\$3,526
8	Jun	\$2,905	\$0	\$1,826	\$1,826
9	Jul	\$4,731	\$2,284	\$0	-\$2,284
10	Aug	\$2,447	\$0	\$3,250	\$3,250
11	Sep	\$5,697	\$1,780	\$0	-\$1,780
12	Oct	\$3,917	\$0	\$2,667	\$2,667
13	Nov	\$6,584	\$0	\$1,500	\$1,500
14	Dec	\$8,084	\$0	\$2,475	\$2,475
15	End	\$10,559			

As you can see in the screenshot below Power Query (or Power BI) fetched the table fully

## Navigator

Show All | Show Selected [0]

Sample Excel Source.xlsx [7]

- ☒ CashFlow
- ☐ Budget
- ☐ Sheet4
- ☐ Sheet1
- ☐ Sheet2
- ☐ Sheet2 (2)
- ☐ Sheet3

### CashFlow

Period	Base	Down	Up	Cash Flow
Start	null	0	5000	5000
Jan	5000	503	0	-503
Feb	4497	1670	0	-1670
Mar	2827	0	4802	4802
Apr	7629	1198	0	-1198
May	6431	3526	0	-3526
Jun	2905	0	1826	1826
Jul	4731	2284	0	-2284
Aug	2447	0	3250	3250
Sep	5697	1780	0	-1780
Oct	3917	0	2667	2667
Nov	6584	0	1500	1500
Dec	8084	0	2475	2475
End	10559	null	null	null

## Loading Excel Sheets into Power Query

You can get data directly from Excel sheets as well, No matter if you have tables or not. Power Query will always read the data from Excel sheets from all cells that contain data. If you have even two data sets in one Excel sheet Power Query still read that and load it correctly. Here is an example of an Excel sheet with two data sets;

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Period	Base	Down	Up	Cash Flow										
2	Start		0	5000	5000										
3	Jan	5000	503	0	-503										
4	Feb	4497	1670	0	-1670										
5	Mar	2827	0	4802	4802										
6	Apr	7629	1198	0	-1198										
7	May	6431	3526	0	-3526										
8	Jun	2905	0	1826	1826										
9	Jul	4731	2284	0	-2284										
10	Aug	2447	0	3250	3250										
11	Sep	5697	1780	0	-1780										
12	Oct	3917	0	2667	2667										
13	Nov	6584	0	1500	1500										
14	Dec	8084	0	2475	2475										
15	End	10559													
16															
17															
18															
19															
20															
21															
22															
23															
24															
25															
26															
27															
28															
29															
30															
31															
32															
33															
34															

Period	Base	Down	Up	Cash Flow
Start		0	5000	5000
Jan	5000	503	0	-503
Feb	4497	1670	0	-1670
Mar	2827	0	4802	4802
Apr	7629	1198	0	-1198
May	6431	3526	0	-3526
Jun	2905	0	1826	1826
Jul	4731	2284	0	-2284
Aug	2447	0	3250	3250
Sep	5697	1780	0	-1780
Oct	3917	0	2667	2667
Nov	6584	0	1500	1500
Dec	8084	0	2475	2475
End	10559			

When you load this sheet in Power Query only data range of data cells up to the last cell's column and row will be fetched;

Navigator

Show All | Show Selected [0]

- Sample Excel Source.xlsx [7]
  - CashFlow
  - Budget
  - Sheet4
  - Sheet1
  - Sheet2**
  - Sheet2 (2)
  - Sheet3

Sheet2

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10	Column11	Column12	Column13	Column14	Column15
Period	Base	Down	Up	Cash Flow										
Start		null	0	5000	5000	null	null	null	null	null	null	null	null	null
Jan	5000	503	0	-503	null	null	null	null	null	null	null	null	null	null
Feb	4497	1670	0	-1670	null	null	null	null	null	null	null	null	null	null
Mar	2827	0	4802	4802	null	null	null	null	null	null	null	null	null	null
Apr	7629	1198	0	-1198	null	null	null	null	null	null	null	null	null	null
May	6431	3526	0	-3526	null	null	null	null	null	null	null	null	null	null
Jun	2905	0	1826	1826	null	null	null	null	null	null	null	null	null	null
Jul	4731	2284	0	-2284	null	null	null	null	null	null	null	null	null	null
Aug	2447	0	3250	3250	null	null	null	null	null	null	null	null	null	null
Sep	5697	1780	0	-1780	null	null	null	null	null	null	null	null	null	null
Oct	3917	0	2667	2667	null	null	null	null	null	null	null	null	null	null
Nov	6584	0	1500	1500	null	null	null	null	null	null	null	null	null	null
Dec	8084	0	2475	2475	null	null	null	null	null	null	null	null	null	null
End	10559	null	null	null	null	null	null	null	null	null	null	null	null	null

Load Edit Cancel

## What Happens If Excel Contains Formatting?

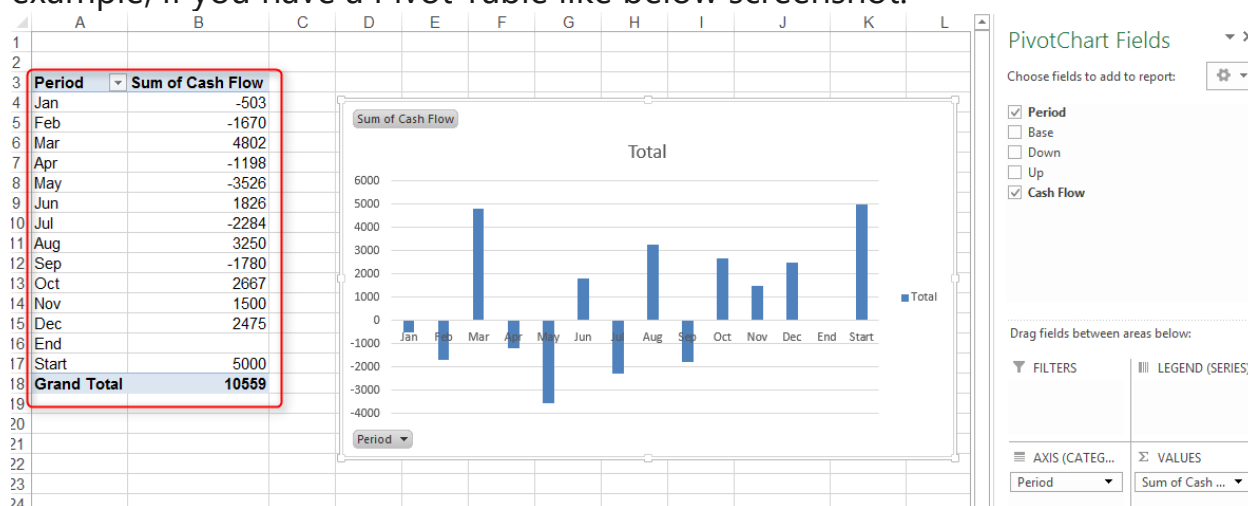
Any formatting such as color, font, and the data type of cells would be ignored when it loads into Power Query. The reason is that Power Query is a data mash-up tools not modeling or visualization tools. You can apply these formatting later in the model (Power Pivot) or the report (Power View). As an example; if your data values contain decimal points such as 12.94 and in Excel, you've formatted cell to have zero decimal points (Excel will show cell value like 13 in this case). Power Query still fetches the original value which is 12.94.

## What Happens to Power View Sheets in Excel?

You can have Power View sheets in Excel, and your Power View sheets can contain Data values, such as a table. However, Power Query won't load anything from a Power View Sheet at this stage. Because usually, Power View sheet uses a data source you can use that source directly in Power Query. Power View data source might be sources such as Pivot Table or SSAS Tabular connection that can be both used in Power Query directly.

## Pivot Tables and Pivot Charts?

You cannot fetch data from Pivot Charts! Why? Because the chart is a visualization element, so the same principle that I said for Power View in above paragraph works here, connect to Pivot Chart's source directly. You can get data from PivotTables however. PivotTables data can be fetched exactly as they shaped in Excel file with the same structure of columns and rows. For example, if you have a Pivot Table like below screenshot:



After loading that in Power Query, you will see the Pivot Table the same (without formatting);

**Navigator**

Show All | Show Selected [0]

Sample Excel Source.xlsx [7]

- ☐ CashFlow
- ☐ Budget
- ☐ Sheet4
- ☐ Sheet1
- ☐ Sheet2
- ☐ Sheet2 (2)
- ☐ Sheet3

**Sheet4**

Period	Sum of Cash Flow
Jan	-503
Feb	-1670
Mar	4802
Apr	-1198
May	-3526
Jun	1826
Jul	-2284
Aug	3250
Sep	-1780
Oct	2667
Nov	1500
Dec	2475
End	null
Start	5000
Grand Total	10559

## What If Your Excel Table Has Merged Cells?

Merged Cells in Excel are commonly used. You can have tables with cells merged vertically or horizontally or even both. Here is an example of a table with merged cells;

Period	Service Center	Budget	Total Budget
1/1	IT	100000	600000
	Finance	500000	
2/1	IT	150000	600000
	Finance	450000	
3/1	IT	170000	570000
	Finance	400000	
4/1	IT	200000	600000
	Finance	400000	

Power Query still reads cells in their original detailed format. It means merged cells won't be merged in Power Query; they will be seen as separate cells.

## Navigator

Sheet3

Column1	Column2	Column3	Column4	Column5	Column6	Column7
Period	Service Center	Budget	Total Budget			
1/1/2015	IT	100000	600000	null	null	
null	Finance	500000	null	null	null	
2/1/2015	IT	150000	600000	null	null	
null	Finance	450000	null	null	null	
3/1/2015	IT	170000	570000	null	null	
null	Finance	400000	null	null	null	
4/1/2015	IT	200000	600000	null	null	
null	Finance	400000	null	null	null	

You can use transformations such as Fill Down to fill null values in the remaining cells of merged cells later on. Also, note in the above screenshot that the green highlighted column is a calculated column. This calculated value will be fetched in Power Query as static values.

## Example Excel Data Source: Olympic Games

I've found a list of all medalists in Olympic games from the very first game (1896) till 2008 (unfortunately 2012 London games are not included in this data set). This is, fortunately, a public list made available by [The Guardian](#), that you can download [here](#). The list is well structured with the main sheet for all medalists as below;

List of medalists at the Games of the Olympiad per edition, sport, discipline, gender and event									
DISCLAIMER: The IOC Research and Reference Service endeavours to provide you with accurate and up-to-date information. However, it offers no guarantees, express or implied, as to the accuracy or completeness of the information provided.									
City	Edition	Sport	Discipline	Athlete	NO	Gender	Event	Event gender	Medal
Athens	1896	Aquatics	Swimming	HAJOS, Alfred	HUN	Men	100m freestyle	M	Gold
Athens	1896	Aquatics	Swimming	HERSCHMANN, Otto	AUT	Men	100m freestyle	M	Silver
Athens	1896	Aquatics	Swimming	DRIVAS, Dimitrios	GRE	Men	100m freestyle for sailors	M	Bronze
Athens	1896	Aquatics	Swimming	MALOKINIS, Ioannis	GRE	Men	100m freestyle for sailors	M	Gold
Athens	1896	Aquatics	Swimming	CHASAPIS, Spiridon	GRE	Men	100m freestyle for sailors	M	Silver
Athens	1896	Aquatics	Swimming	CHOROPHAS, Efstathios	GRE	Men	1200m freestyle	M	Bronze
Athens	1896	Aquatics	Swimming	HAJOS, Alfred	HUN	Men	1200m freestyle	M	Gold
Athens	1896	Aquatics	Swimming	ANDREOU, Joannis	GRE	Men	1200m freestyle	M	Silver
Athens	1896	Aquatics	Swimming	CHOROPHAS, Efstathios	GRE	Men	400m freestyle	M	Bronze
Athens	1896	Aquatics	Swimming	NEUMANN, Paul	AUT	Men	400m freestyle	M	Gold
Athens	1896	Aquatics	Swimming	PEPANOS, Antonios	GRE	Men	400m freestyle	M	Silver
Athens	1896	Athletics	Athletics	LANE, Francis	USA	Men	100m	M	Bronze
Athens	1896	Athletics	Athletics	SZOKOLYI, Alajos	HUN	Men	100m	M	Bronze
Athens	1896	Athletics	Athletics	BURKE, Thomas	USA	Men	100m	M	Gold

As you see in the above sheet, all medalists with their main sports category and discipline and detailed event are available. Name of Athletes and their

gender and Medals as well as the Olympic game (year) all are listed. Countries information listed as three character code. These three-character codes are available as a reference in another sheet of the file named IOC Country Codes as below;

Country	Int Olympic	ISO code	Country
Afghanistan	AFG	AF	Afghanistan
Albania	ALB	AL	Albania
Algeria	ALG	DZ	Algeria
American Samoa*	ASA	AS	American Samoa*
Andorra	AND	AD	Andorra
Angola	ANG	AO	Angola
Antigua and Barbuda	ANT	AG	Antigua and Barbuda
Argentina	ARG	AR	Argentina
Armenia	ARM	AM	Armenia
Aruba*	ARU	AW	Aruba*
Australia	AUS	AU	Australia
Austria	AUT	AT	Austria
Azerbaijan	AZE	AZ	Azerbaijan
Bahamas	BAH	BS	Bahamas
Bahrain	BRN	BH	Bahrain
Bangladesh	BAN	BD	Bangladesh
Barbados	BAR	BB	Barbados

Data is well structured and loading it into Power BI would be just matter of seconds! Start getting this information by getting Data From Excel and then address the downloaded Excel file. In Navigator dialog box choose All Medalists and IOC Country Codes sheets both to be checked. Also, note that All Medalists sheet's data shows in non-merged cell style (as you've learned earlier in this section; merged cells will be un-merged in Power Query). After selecting these two sheets click on Edit.



## Navigator

Search:

Show All | Show Selected [2]

- Summer Olympic medallists 1896 to...
- ☒ ALL MEDALISTS
- ☐ BREAKDOWN
- ☐ COUNTRY TOTALS
- ☒ IOC COUNTRY CODES
- ☐ TEAM EVENTS FIXED, ALL YRS T...
- ☐ Sheet8
- ☐ Chart2
- ☐ Chart1
- ☐ IOC COUNTRY CODES 1
- ☐ TOPOLYMPIANS

## ALL MEDALISTS

Preview downloaded on Sunday

Column1	Column2	Column3	Column4	Column5
List of medallists at the Games of the Olympiad per edition, sport, disciplir	null	null	null	
null	null	null	null	
DISCLAIMER: The IOC Research and Reference Service endeavours to prov	null	null	null	
null	null	null	null	
City	Edition	Sport	Discipline	Athlete
Athens	1896	Aquatics	Swimming	HAJOS,
Athens	1896	Aquatics	Swimming	HERSCH
Athens	1896	Aquatics	Swimming	DRIVAS
Athens	1896	Aquatics	Swimming	MALOK
Athens	1896	Aquatics	Swimming	CHASAI
Athens	1896	Aquatics	Swimming	CHORO
Athens	1896	Aquatics	Swimming	HAJOS,
Athens	1896	Aquatics	Swimming	ANDRE
Athens	1896	Aquatics	Swimming	CHORO
Athens	1896	Aquatics	Swimming	NEUMA
Athens	1896	Aquatics	Swimming	PEPANI
Athens	1896	Athletics	Athletics	LANE, F
Athens	1896	Athletics	Athletics	SZOKOI
Athens	1896	Athletics	Athletics	BURKE,
Athens	1896	Athletics	Athletics	HOFMA
Athens	1896	Athletics	Athletics	CURTIS
Athens	1896	Athletics	Athletics	GOULD

Load

Edit

Cancel

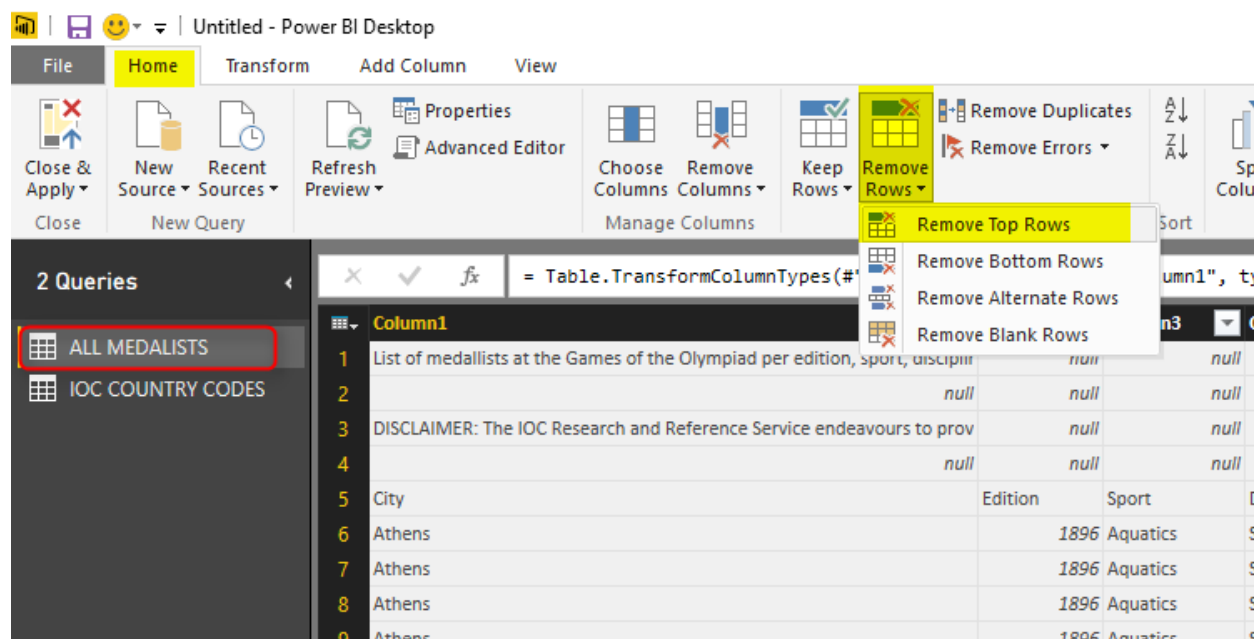
There are just a few changes that we need to make;

## Remove Rows

All Medalists query contains four heading rows which we don't need them (it is just title and disclaimer) so better to remove those;

Go to All Medalists query and then click on Remove Rows in the Home menu.

From the "Remove Rows" popup menu, choose Remove Top Rows

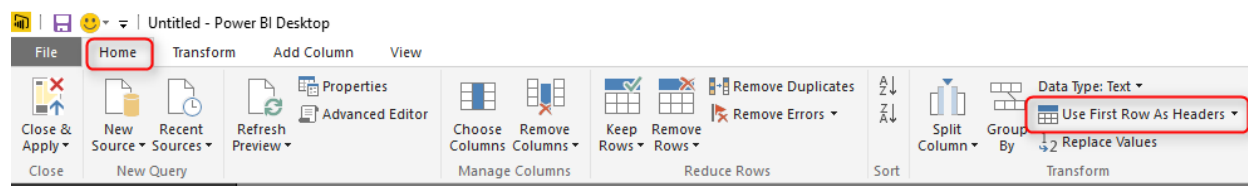


Remove Top 4 Rows in the dialog box by entering 4. and the result will look like below;

	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10
1	City	Edition	Sport	Discipline	Athlete	NOC	Gender	Event	Event_gender	Medal
2	Athens	1896	Aquatics	Swimming	HAJOS, Alfred	HUN	Men	100m freestyle	M	Gold
3	Athens	1896	Aquatics	Swimming	HERSCHMANN, Otto	AUT	Men	100m freestyle	M	Silver
4	Athens	1896	Aquatics	Swimming	DRIVAS, Dimitrios	GRE	Men	100m freestyle for sailors	M	Bronze
5	Athens	1896	Aquatics	Swimming	MALOKINIS, Ioannis	GRE	Men	100m freestyle for sailors	M	Gold
6	Athens	1896	Aquatics	Swimming	CHASAPIS, Spiridon	GRE	Men	100m freestyle for sailors	M	Silver
7	Athens	1896	Aquatics	Swimming	CHOROPHAS, Efstathios	GRE	Men	1200m freestyle	M	Bronze
8	Athens	1896	Aquatics	Swimming	HAJOS, Alfred	HUN	Men	1200m freestyle	M	Gold
9	Athens	1896	Aquatics	Swimming	ANDREOU, Joannis	GRE	Men	1200m freestyle	M	Silver
10	Athens	1896	Aquatics	Swimming	CHOROPHAS, Efstathios	GRE	Men	400m freestyle	M	Bronze
11	Athens	1896	Aquatics	Swimming	NEUMANN, Paul	AUT	Men	400m freestyle	M	Gold
12	Athens	1896	Aquatics	Swimming	PEPANOS, Antonios	GRE	Men	400m freestyle	M	Silver
13	Athens	1896	Athletics	Athletics	LANE, Francis	USA	Men	100m	M	Bronze
14	Athens	1896	Athletics	Athletics	SZOKOLYI, Alajos	HUN	Men	100m	M	Bronze
15	Athens	1896	Athletics	Athletics	BURKE, Thomas	USA	Men	100m	M	Gold

## Use First Row As Headers

Headers of the query as you see in the screenshot above are Column1, Column2.... Fortunately, in the data set, the first row contains column headers. We can simply set that to be used for column headers in Power Query. in the Home tab click on Use First Row As Headers.



Dataset will look like the screenshot below after above change

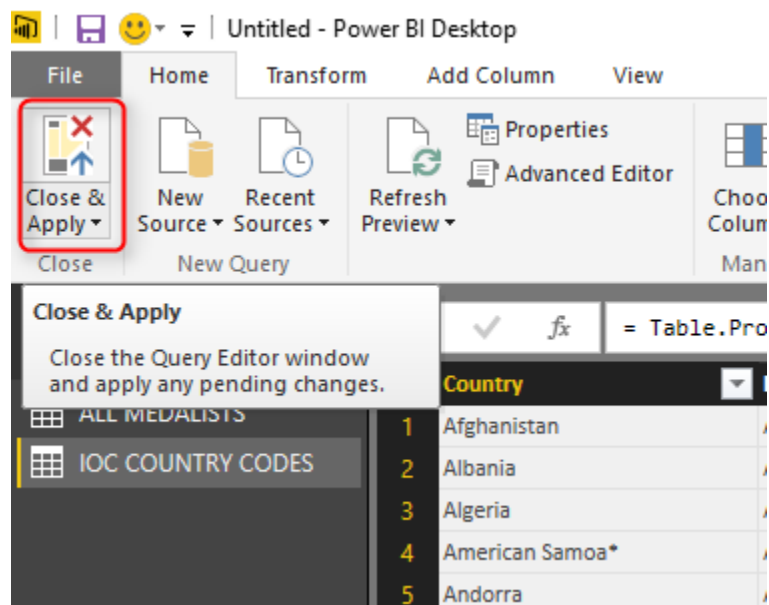
#	City	Edition	Sport	Discipline	Athlete	NOC	Gender	Event	Event_gender	Medal
1	Athens	1896	Aquatics	Swimming	HAJOS, Alfred	HUN	Men	100m freestyle	M	Gold
2	Athens	1896	Aquatics	Swimming	HERSCHMANN, Otto	AUT	Men	100m freestyle	M	Silver
3	Athens	1896	Aquatics	Swimming	DRIVAS, Dimitrios	GRE	Men	100m freestyle for sailors	M	Bronze
4	Athens	1896	Aquatics	Swimming	MALOKINIS, Ioannis	GRE	Men	100m freestyle for sailors	M	Gold
5	Athens	1896	Aquatics	Swimming	CHASAPIS, Spiridon	GRE	Men	100m freestyle for sailors	M	Silver
6	Athens	1896	Aquatics	Swimming	CHOROPHAS, Efstathios	GRE	Men	1200m freestyle	M	Bronze
7	Athens	1896	Aquatics	Swimming	HAJOS, Alfred	HUN	Men	1200m freestyle	M	Gold
8	Athens	1896	Aquatics	Swimming	ANDREOU, Joannis	GRE	Men	1200m freestyle	M	Silver
9	Athens	1896	Aquatics	Swimming	CHOROPHAS, Efstathios	GRE	Men	400m freestyle	M	Bronze
10	Athens	1896	Aquatics	Swimming	NEUMANN, Paul	AUT	Men	400m freestyle	M	Gold
11	Athens	1896	Aquatics	Swimming	PEPANOS, Antonios	GRE	Men	400m freestyle	M	Silver
12	Athens	1896	Athletics	Athletics	LANE, Francis	USA	Men	100m	M	Bronze
13	Athens	1896	Athletics	Athletics	SZOKOLYI, Alajos	HUN	Men	100m	M	Bronze
14	Athens	1896	Athletics	Athletics	BURKE, Thomas	USA	Men	100m	M	Gold
15	Athens	1896	Athletics	Athletics	HOFMANN, Fritz	GER	Men	100m	M	Silver
16	Athens	1896	Athletics	Athletics	CURTIS, Thomas	USA	Men	110m hurdles	M	Gold

For IOC Country Codes query just set the first row as headers. No more changes are required

#	Country	Int Olympic Committee code	ISO code	Country_1
1	Afghanistan	AFG	AF	Afghanistan
2	Albania	ALB	AL	Albania
3	Algeria	ALG	DZ	Algeria
4	American Samoa*	ASA	AS	American Samoa*
5	Andorra	AND	AD	Andorra
6	Angola	ANG	AO	Angola
7	Antigua and Barbuda	ANT	AG	Antigua and Barbuda
8	Argentina	ARG	AR	Argentina
9	Armenia	ARM	AM	Armenia
10	Aruba*	ARU	AW	Aruba*
11	Australia	AUS	AU	Australia
12	Austria	AUT	AT	Austria
13	Azerbaijan	AZE	AZ	Azerbaijan
14	Bahamas	BAH	BS	Bahamas
15	Bahrain	BRN	BH	Bahrain

## Close and Load

After above changes, we can now load the result set into the Power BI model to build report for it. You can simply click on Close and Apply menu button in Home Tab



Power BI will load result sets into memory for further modeling and reporting.

## Apply Query Changes

ALL MEDALISTS  
Waiting for other queries...

IOC COUNTRY CODES  
Evaluating...

Cancel

## A bit of Modeling

This section is not about data modeling. However, I want you to be able to play with this data set and build some nice reports with it (if you can't wait till modeling and visualization chapters of this book). So Let's create the relationship between All medalists and IOC Country Code. For doing this go to Relationship tab in Power BI Desktop and then click on Manage Relationship. Then create a new relationship as below;



## Create Relationship

Select tables and columns that relate to one another.

ALL MEDALISTS

City	Edition	Sport	Discipline	Athlete	NOC	Gender	Event	Event_gender
London	1908	Athletics	Athletics	CARTMELL, John Nathaniel	USA	Men	4x400m relay	M
London	1908	Athletics	Athletics	HAMILTON, William Frank	USA	Men	4x400m relay	M
London	1908	Athletics	Athletics	SHEPPARD, Melvin	USA	Men	4x400m relay	M
London	1908	Athletics	Athletics	TAYLOR, John Baxter	USA	Men	4x400m relay	M
Stockholm	1912	Athletics	Athletics	LINDBERG, Edward F.	USA	Men	4x400m relay	M

IOC COUNTRY CODES

Country	Int Olympic Committee code	ISO code	Country_1
Afghanistan	AFG	AF	Afghanistan
Albania	ALB	AL	Albania
Algeria	ALG	DZ	Algeria
American Samoa*	ASA	AS	American Samoa*
Andorra	AND	AD	Andorra

Advanced options

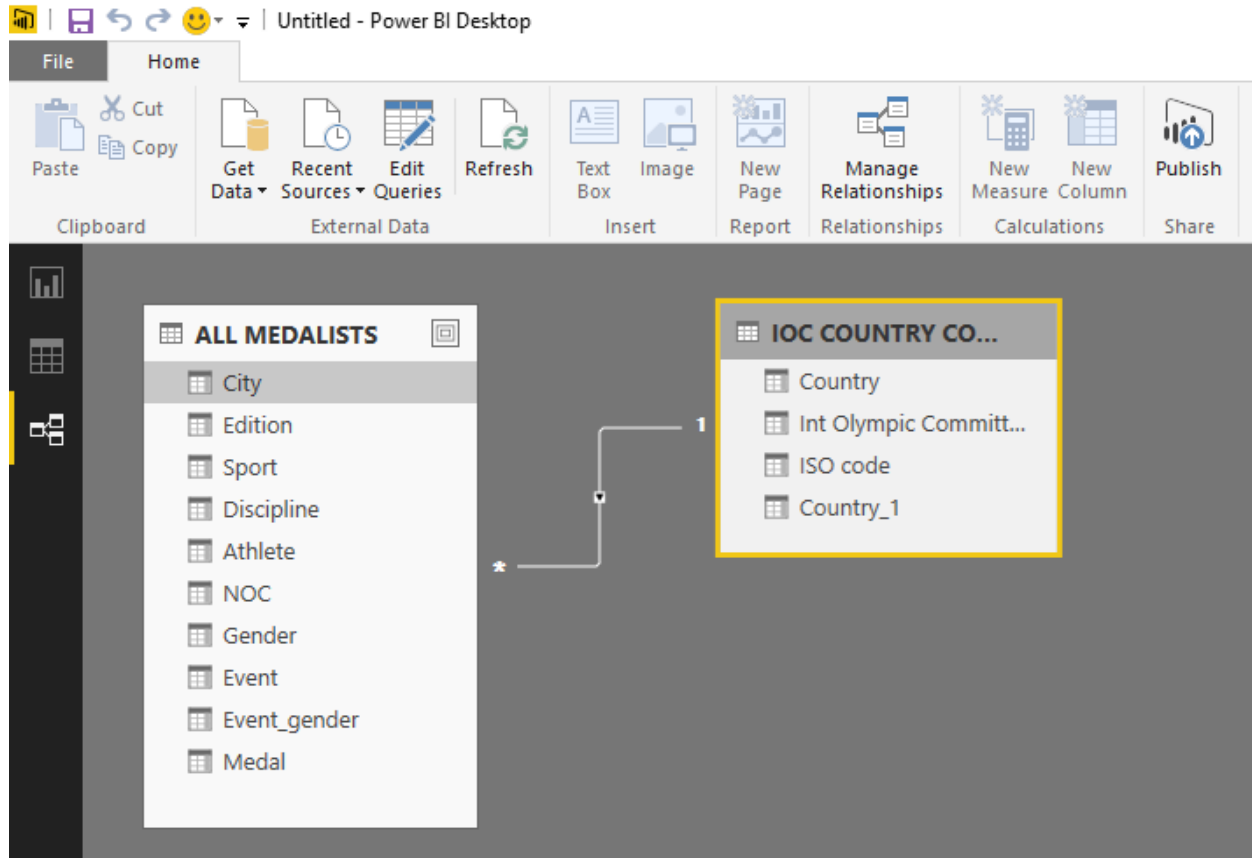
Cardinality  
Many to One (\*:1)

Cross filter direction  
Single

☒ Make this relationship active

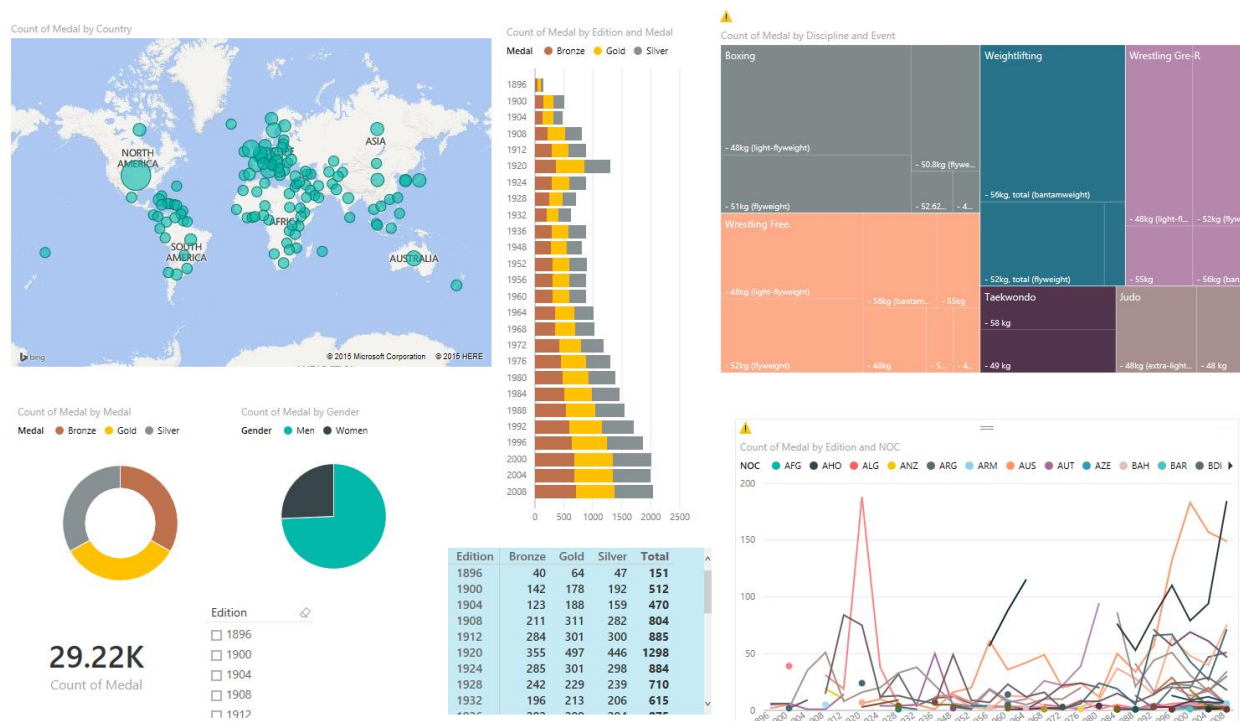
OK Cancel

After the relationship has been created, you will see it in the relationship diagram



## Play Time!

Now it's your turn to play with this data set and make some nice reports and visualization items. Here is what I've built with this data set:



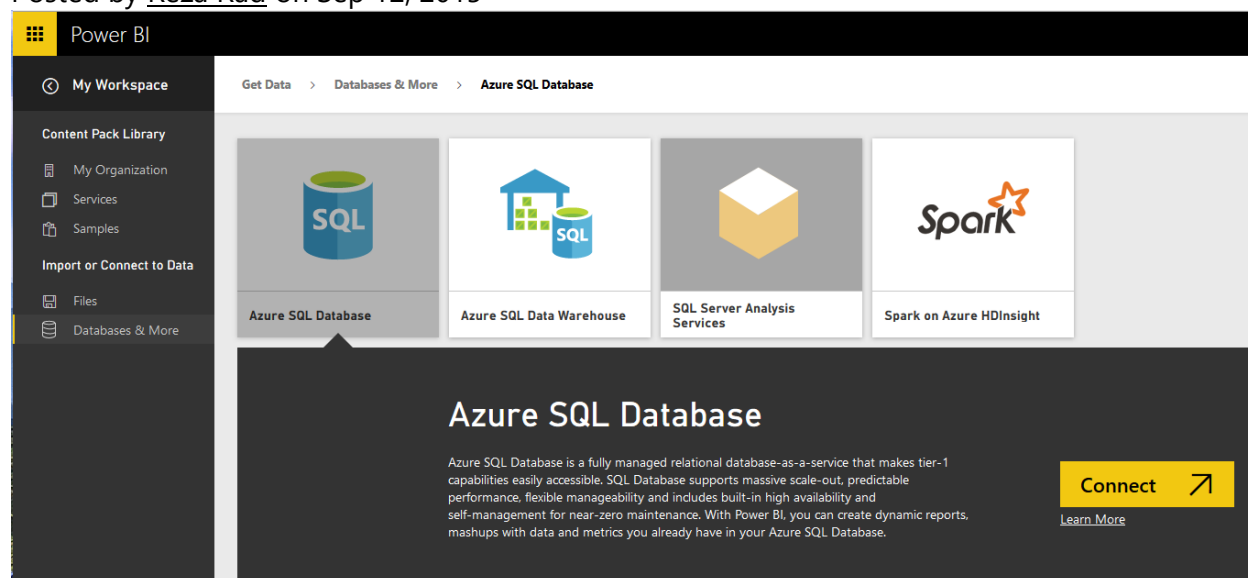
Don't be panic if you can't build report above. I will go through step by step process of building this report later in the Reporting Chapters.

## Summary

In this section, you've learned some tips of working with Excel Data source from Power Query or Power BI Desktop. You've learned that merged cells would be loaded into Power Query Un-merged. Formatting won't be considered at the time of loading data into Power Query, and Power Query can load data from Pivot Tables, Excel Tables, and Sheets. You've also learned a real-world example of fetching Olympic medalists data from Excel file into Power BI Desktop. In next sections, I will get you through the journey of Getting data from some other data sources.

# Power BI Get Data: From Azure SQL Database

Posted by [Reza Rad](#) on Sep 12, 2015



Power BI and Power Query can connect to files such as Excel, CSV, text files and on-premises databases such as SQL Server, Oracle, MySQL. Power BI can connect to many data sources on the cloud such as Azure SQL Database, Azure SQL Data Warehouse, etc. In this part, you will learn how to connect from Power BI Desktop to Azure SQL Database. There is also a way of connecting to Azure SQL Database with a direct connection from the Power BI website which will be explored in this section as well. You will also learn how you can schedule your report to refresh data loaded from Azure SQL DB. So, In general, you should expect to learn everything related to Power BI relation to Azure SQL Database in this section.

In this section you will learn;

- How to connect from Power BI Desktop to Azure SQL Database
- Schedule Power BI for refreshing data from Azure SQL Database
- Direct Connection to Azure SQL Database from Power BI Website



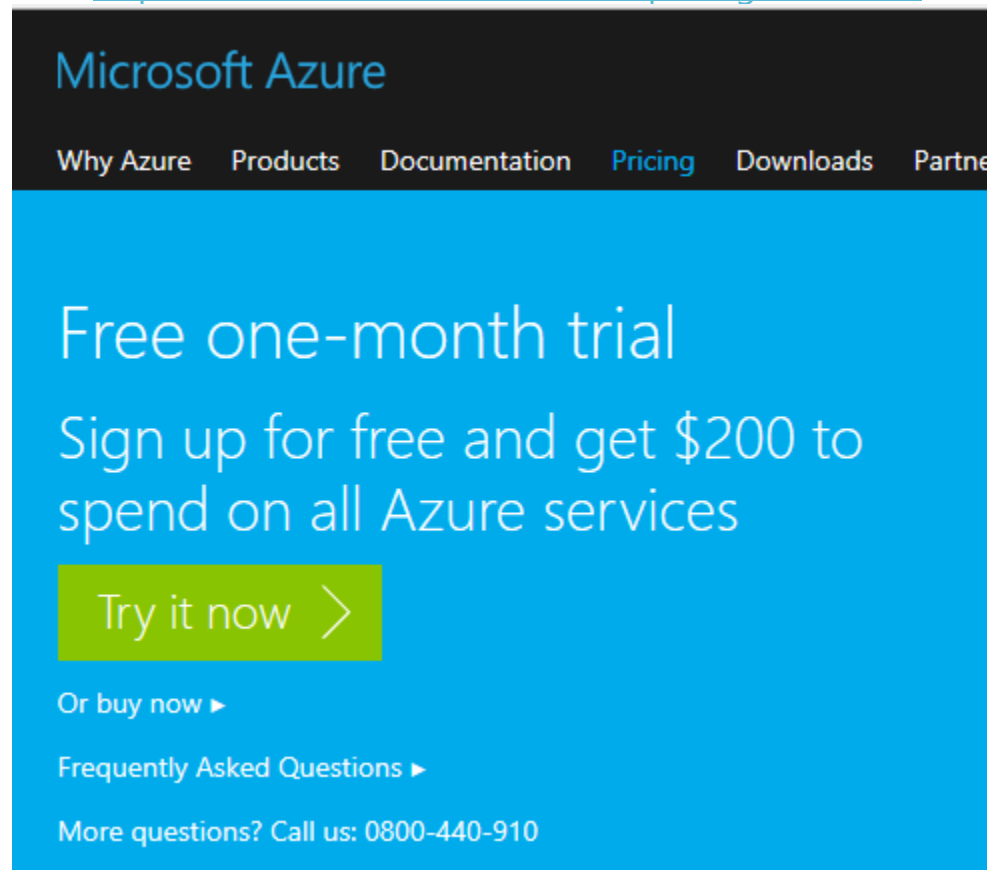
## Preparation

For this section, you need an Azure SQL Database. I use AdventureWorksLT example; this is the sample database in Azure SQL Database templates that you can easily install and configure. If you have this database set up on Azure, then you can skip this step. For creating an Azure SQL Database for AdventureWorksLT database follow below steps;

### **Do you need an Azure subscription to run this experiment?**

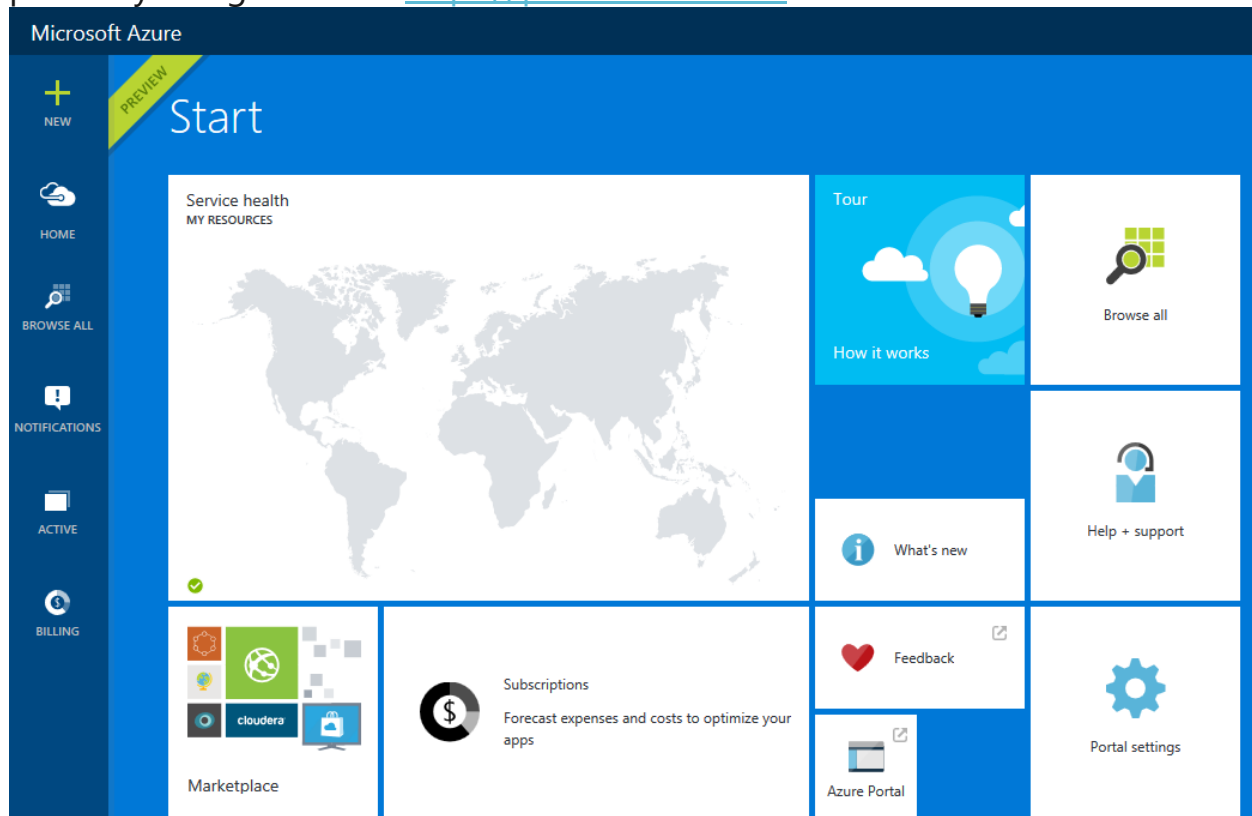
Yes, but don't worry if you don't have it. You can have an Azure subscription free for 25 days with 200\$ free credit for you to use, use the trial version. You can start the trial version by following this

URL: <https://azure.microsoft.com/en-us/pricing/free-trial/>

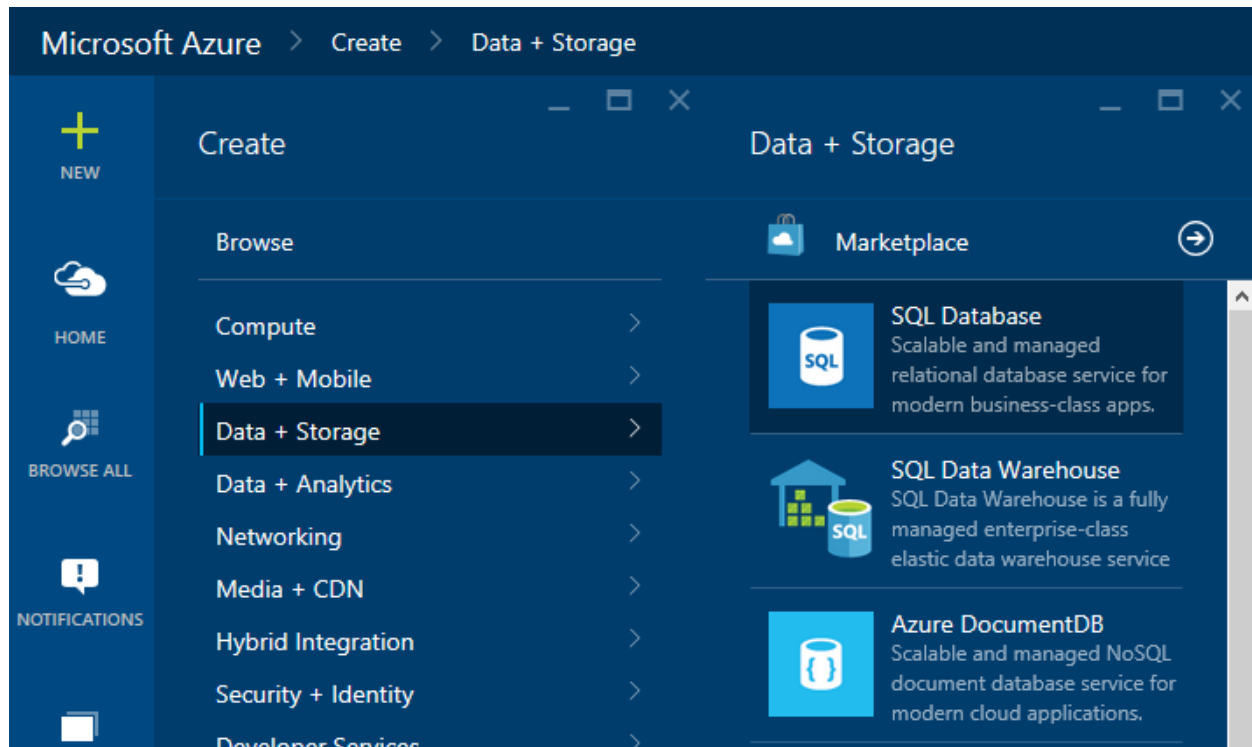


After setting up your Azure Account, go to Azure Portal. I have to mention here that there are two versions of the management portal for Azure. The new

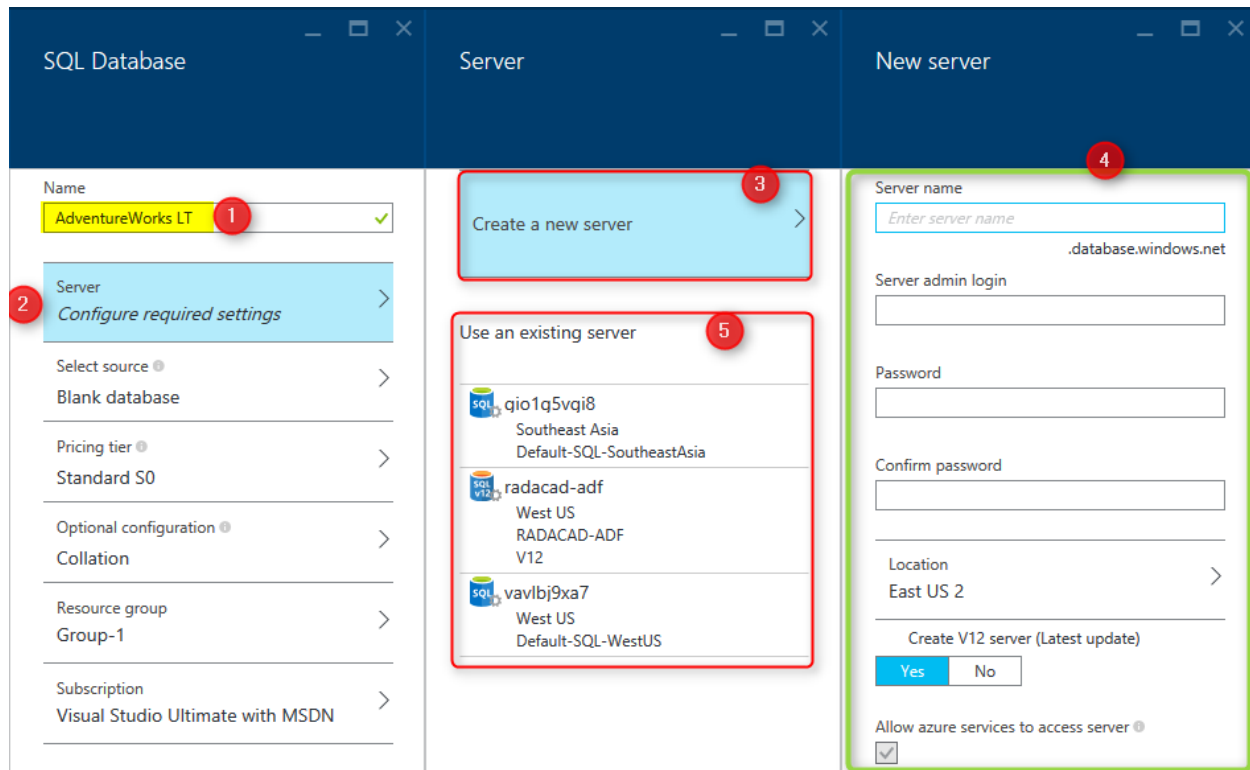
Azure portal which is tablet friendly, with a newer and better look and feels, and the old management portal. Screenshots and steps described in this example all have been done in the new Azure Portal. You can go to the Azure portal by using this URL: <https://portal.azure.com>



You can manage your Azure services in the management portal by creating new services, editing existing services. Talking about Azure services is out of scope for this example, and you need to read books on that topic. However, for this example let's smoothly continue steps to create an Azure SQL Database. Click on New on the top left side and then under Data + Storage choose SQL Database

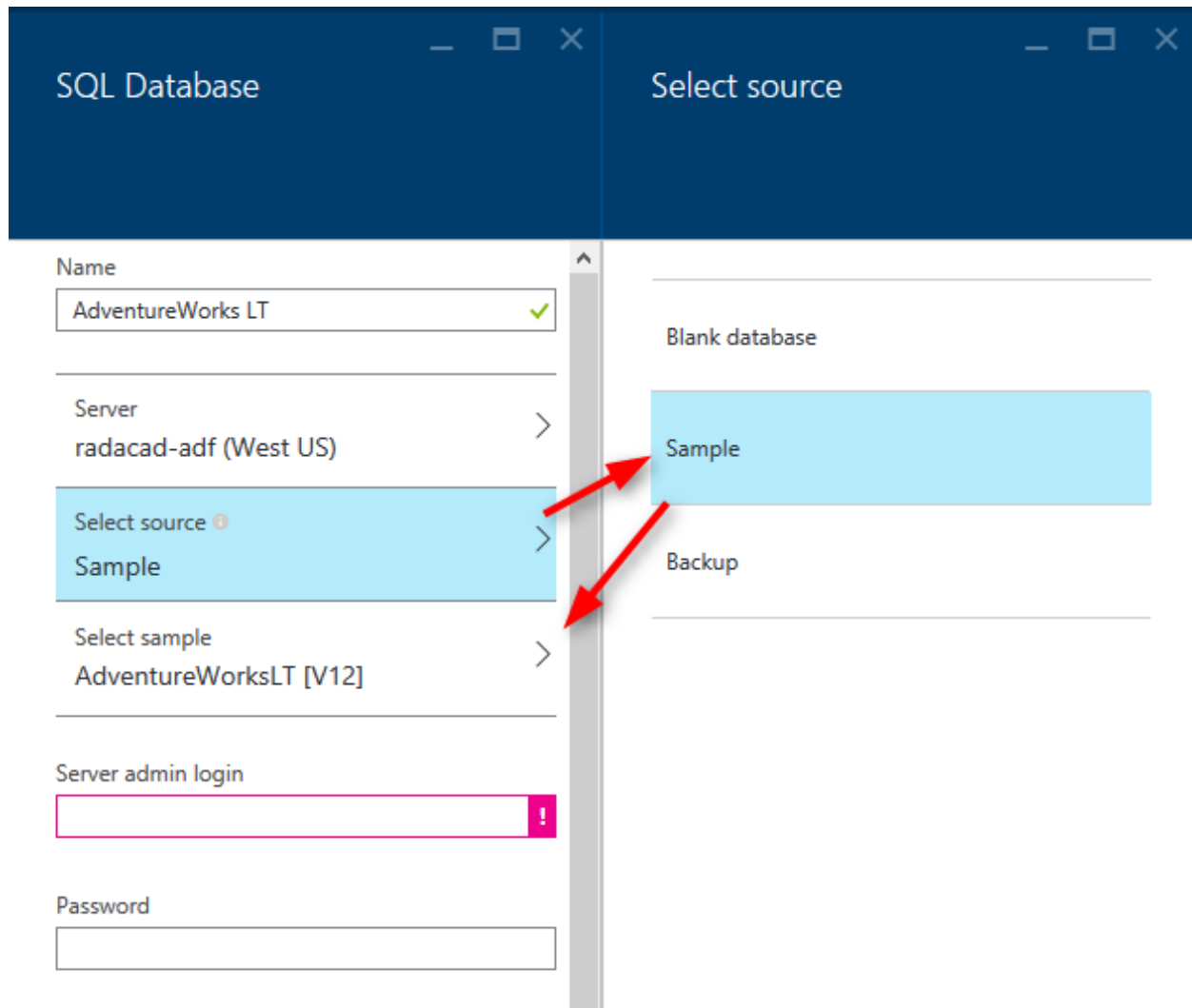


In the SQL Database Create pane, name the database as AdventureWorks LT. You have to choose the server also. The server is like a SQL Server instance that this database will be hosted on that. You can choose from an existing server, or you can create a new Server.

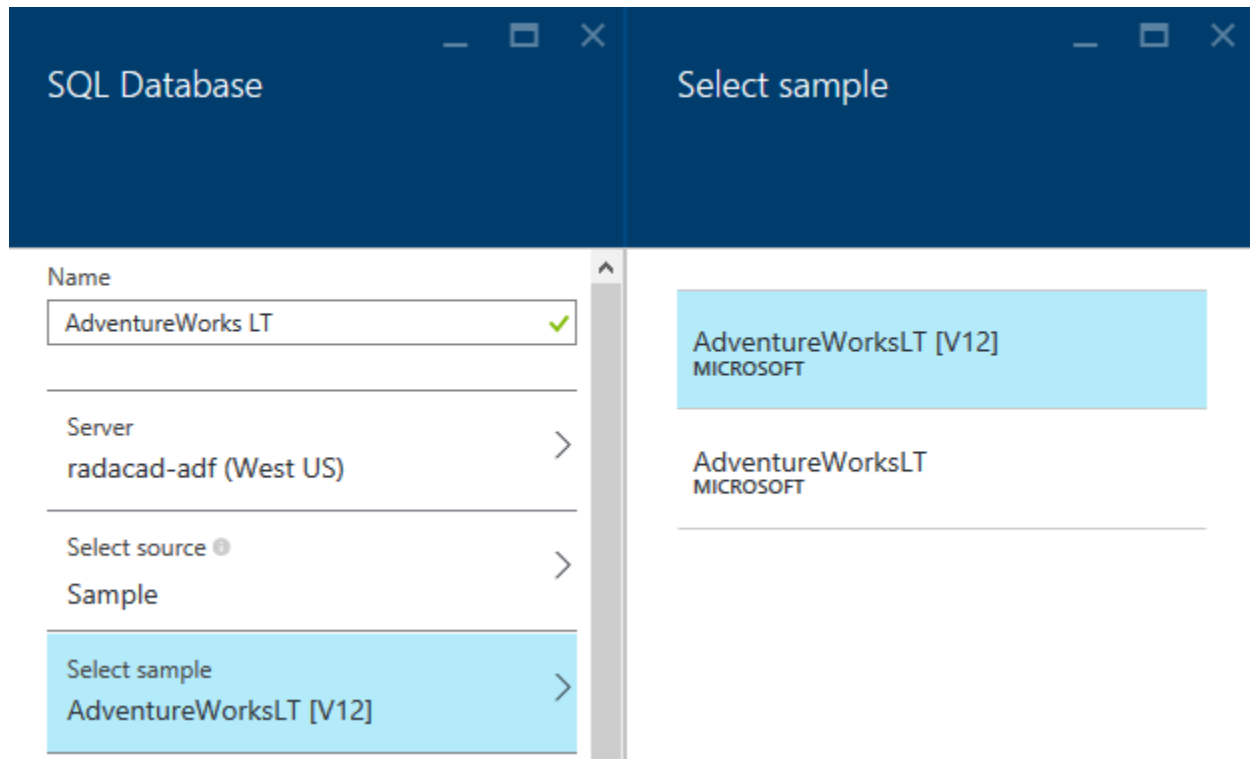


As you can see in the screenshot above, after going to Configure required Settings for the Server, you can choose to create a new server (numbered 3), which will redirect you to a new pane for setting up the server (numbered 4). or you can choose an existing server (numbered 5).

After setting up the server, you have to select the source for the database. For this example choose Sample. After choosing sample, you will see the Select Sample option below appears.



Choose the sample as AdventureWorksLT [V12], and then type in the server admin login and password (you have defined that when you set up the Server)



You have also to choose a pricing tier, and resource group. For pricing tier use one of the tiers (you can better choose yourself), and then for the resource group you can choose an existing one or create a new. A resource group is a grouping for Azure services; you can have a resource group and add all related azure services under that. For example, you can have a resource group for Power BI Online Book and create all examples of this book under that. Please note that the resource group name should not have spaces in the name, but it can have dashes.

## SQL Database

Name

AdventureWorks LT ✓

Server

radacad-adf (West US) >

Select source ⓘ

Sample >

Select sample

AdventureWorksLT [V12] >

Server admin login

reza ✓

Password

•••••••• ✓

Pricing tier ⓘ

Standard S0 >

Optional configuration ⓘ

Collation >

Resource group

RADACAD-ADF 🔒

Subscription


Visual Studio Ultimate with MSDN 🔒






☒ Pin to Startboard



Create

After all the configuration click on Create, so the SQL DB creates. The tick on the checkbox for the pin to Startboard will bring the SQL DB on the first welcome page (start board) of the Azure Portal. It may take a bit time for the database to be created. After completion of creating database process, you will be redirected to database page in the azure portal (if you didn't, then click on the AdventureWorks LT database on start board to go to its pane). the screenshot below is showing the database created



 **AdventureWorksLT**  
SQL Database



**Essentials**  

**Resource group**  
RADACAD-ADF

**Status**  
Online

**Location**  
West US

**Subscription name**  
Visual Studio Ultimate with MSDN

**Subscription ID**  
71f18500-c4c8-4b70-a37e-97ad201c94ce

**Server name**  
[radacad-adf.database.windows.net](#)

**Pricing tier**  
S0 Standard (10 DTUs)

**Geo-replication role**  
Not configured

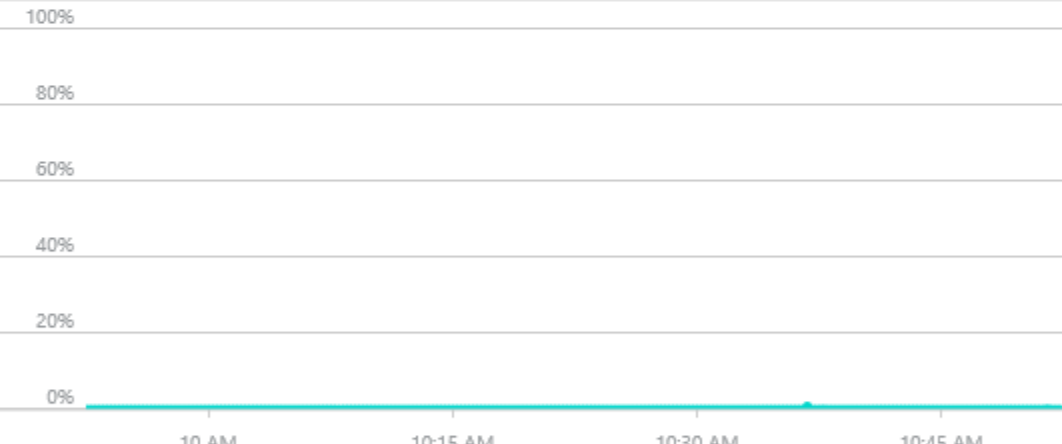
**Server version**  
V12

**Connection strings**  
[Show database connection strings](#)

[All settings](#) →

**Monitoring**

**Resource Utilization** [Edit](#)



DTU PERCENTAGE

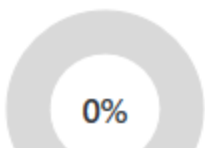
**0 %**

100%  
80%  
60%  
40%  
20%  
0%

10 AM 10:15 AM 10:30 AM 10:45 AM

**Usage**

**Database Size**




**CURRENT**  
**6.94 MB**

**0%**

**THRESHOLD**

**Pricing tier**  
ADVENTUREWORKSLT



Now you are all set, example database is ready to be used in Power BI.

## Get Data From Azure SQL Database

You can connect to Azure SQL Database from Power BI Desktop or Power Query for Excel. Both methods work the same. Let's go through the connection from Power BI Desktop. Before starting steps, I have to mention that Power BI Desktop connection to Azure SQL Database is an off-line connection. The off-line connection here means the data from Azure SQL Database will be loaded into the Power BI model and then reports will use the data in the model, this disconnected way of connection is what I call off-line. The off-line connection to Azure SQL DB can be scheduled in the Power BI website to be refreshed to populated updated data from the database. In this section, we will create the connection from Power BI Desktop to Azure SQL DB, and in the next section following you will learn how to schedule the data refresh.








Open the Power BI Desktop and Get Data from Azure SQL Database

### Get Data

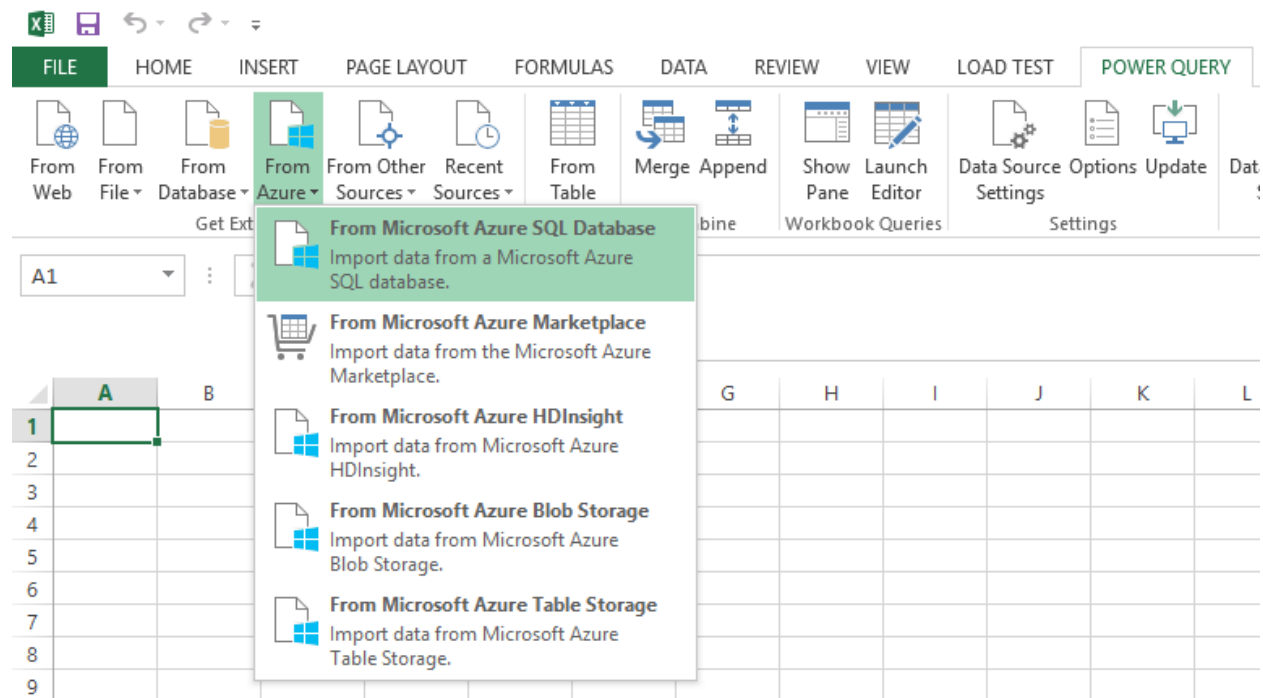
  
  

- All
- File
- Database
- Azure**
- Other

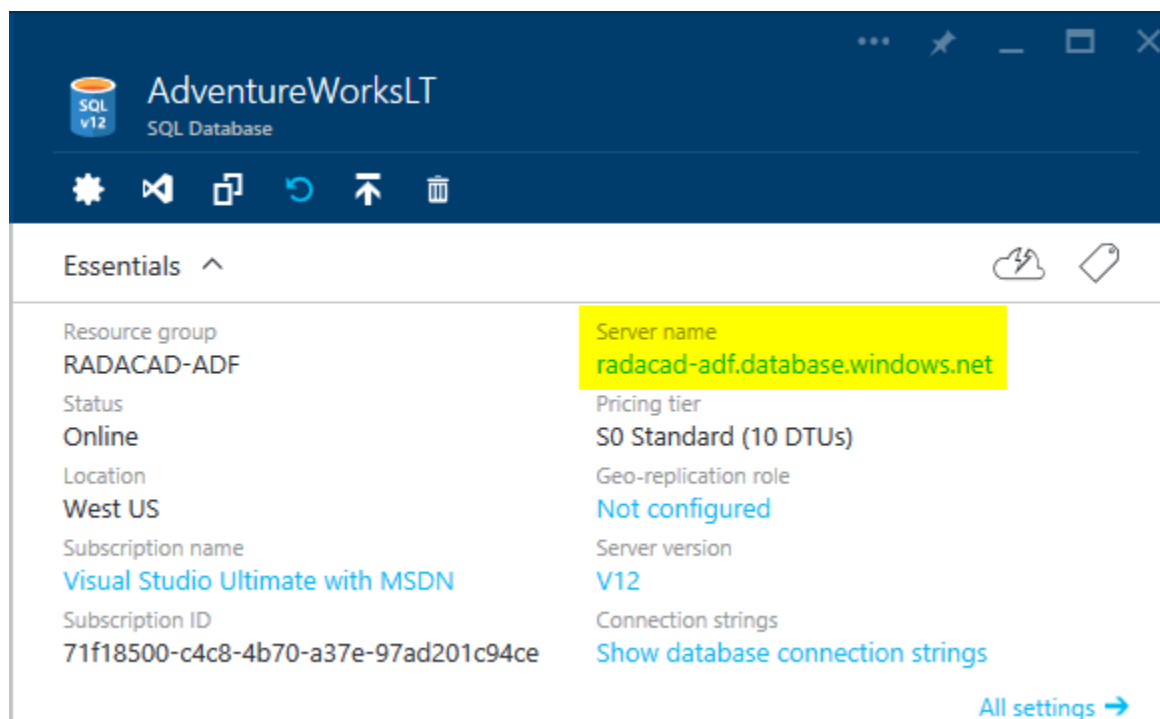
#### Azure

-  Microsoft Azure SQL Database
-  Microsoft Azure SQL Data Warehouse
-  Microsoft Azure Marketplace
-  Microsoft Azure HDInsight
-  Microsoft Azure Blob Storage
-  Microsoft Azure Table Storage
-  Azure HDInsight Spark (Beta)

In Power Query for Excel, you can also follow the path mentioned in the screenshot below



You need to enter the server name in SQL Server Database dialog box. Remember that you've set up the server when you created Azure SQL DB. If you don't know what the server for your database is, simply find it through Azure portal under the Azure SQL DB pane;



Type in the server name in SQL Server Database dialog box in Power BI Desktop. And type in the database as AdventureWorks LT. Leave the SQL statement as is. then press OK

## SQL Server Database

Import data from a SQL Server database.

Server

radacad-adf.database.windows.net

Database (optional)

AdventureWorksLT

SQL statement (optional)

OK

Cancel

It is very likely that you get to the window that says Unable to Connect. This window is saying that Power BI Desktop cannot connect to Azure SQL DB and the reason is that the Azure SQL Server didn't allow your IP address to pass through its firewall. I have to mention that Azure SQL Server by default doesn't

allow external IP addresses to connect to it. If you want to connect to any databases on Azure SQL Server, you have to allow the IP of that machine to pass through. This is not your internal network IP; this is the IP that your internet connection has. You can find the IP easily. It is mentioned in Unable to Connect error message below!

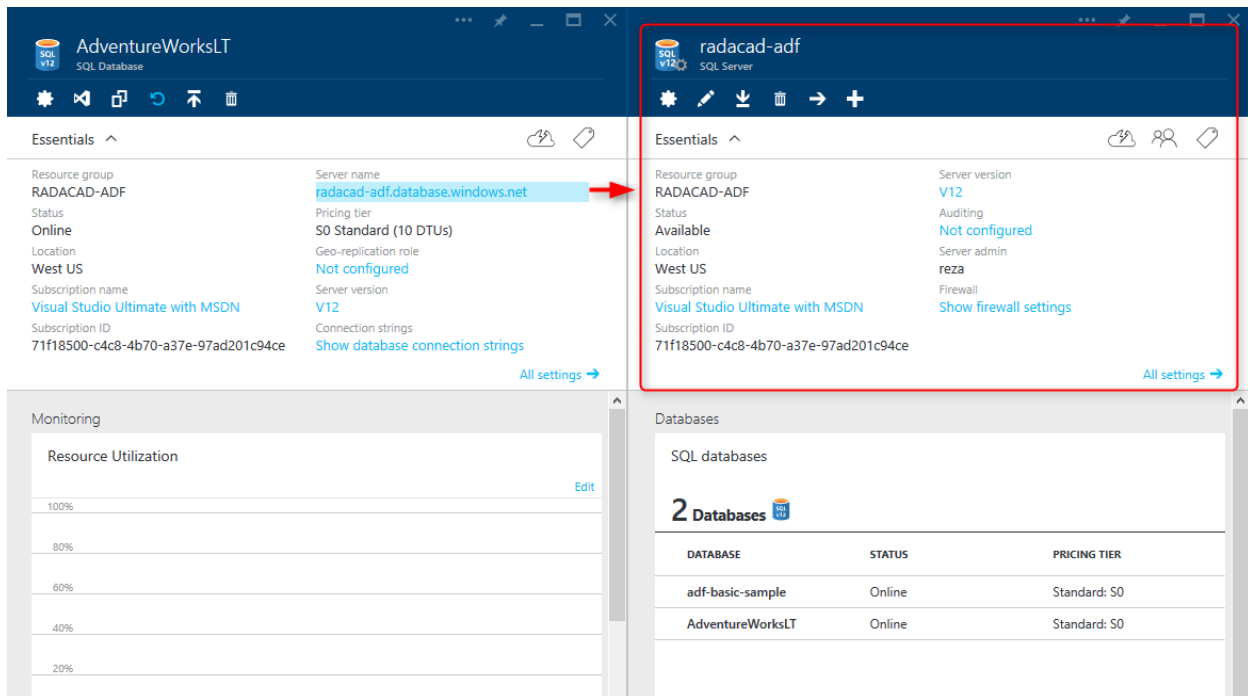
*We encountered an error while trying to connect to ".....". Details: Microsoft SQL: Cannot open server'...' Requested by the login. A client with IP address'210.246.15.145' is not allowed to access the server. To enable access, use the Windows Azure Management Portal or run sp\_set\_firewall\_rule on the master database to create a firewall rule for this IP address or address range. It may take up to five minutes for this change to take effect."*

## Unable to Connect

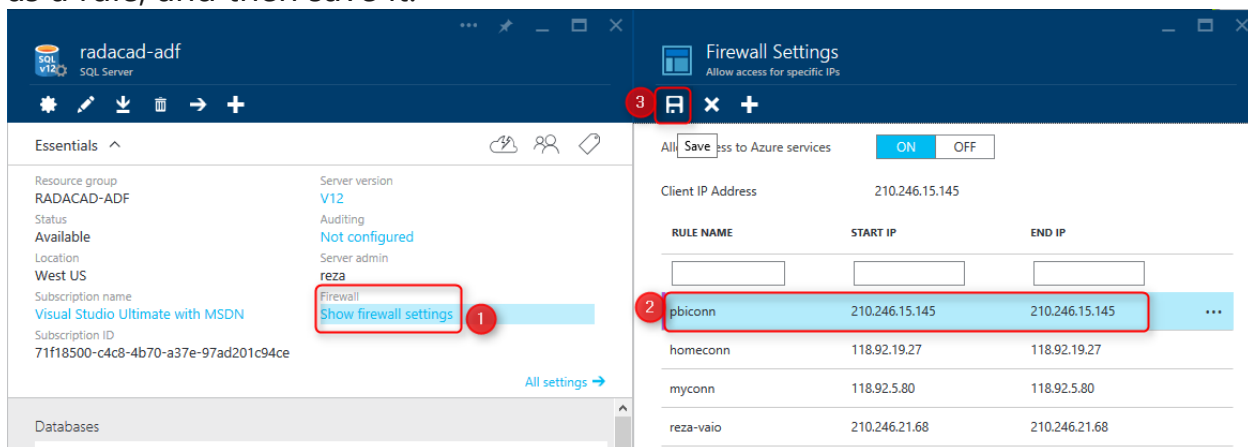
We encountered an error while trying to connect to "radacad-adf.database.windows.net: AdventureWorksLT". Details: "Microsoft SQL: Cannot open server 'radacad-adf' requested by the login. Client with IP address '210.246.15.145' is not allowed to access the server. To enable access, use the Windows Azure Management Portal or run sp\_set\_firewall\_rule on the master database to create a firewall rule for this IP address or address range. It may take up to five minutes for this change to take effect."

[Retry](#)[Edit](#)[Cancel](#)

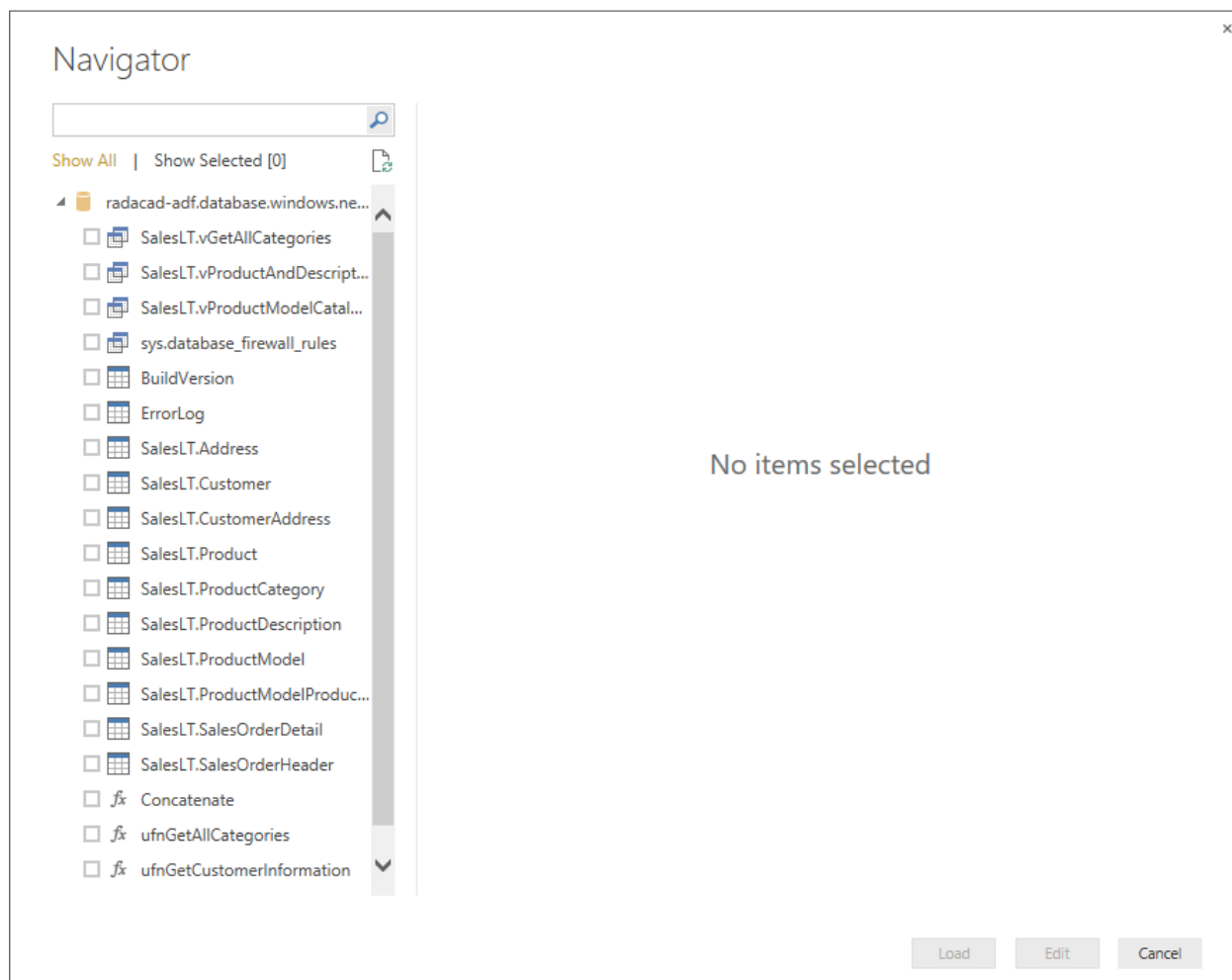
So Let's add the IP in firewall pass list. Go to Azure Portal again. if you closed it, open it again, go to Browse All, then Choose SQL Databases, then choose AdventureWorksLT database in the list, and then click on the server name of it to open the Azure SQL Server administration pane



Click on Show firewall settings. In the Firewall Settings pane, enter the new IP as a rule, and then save it.

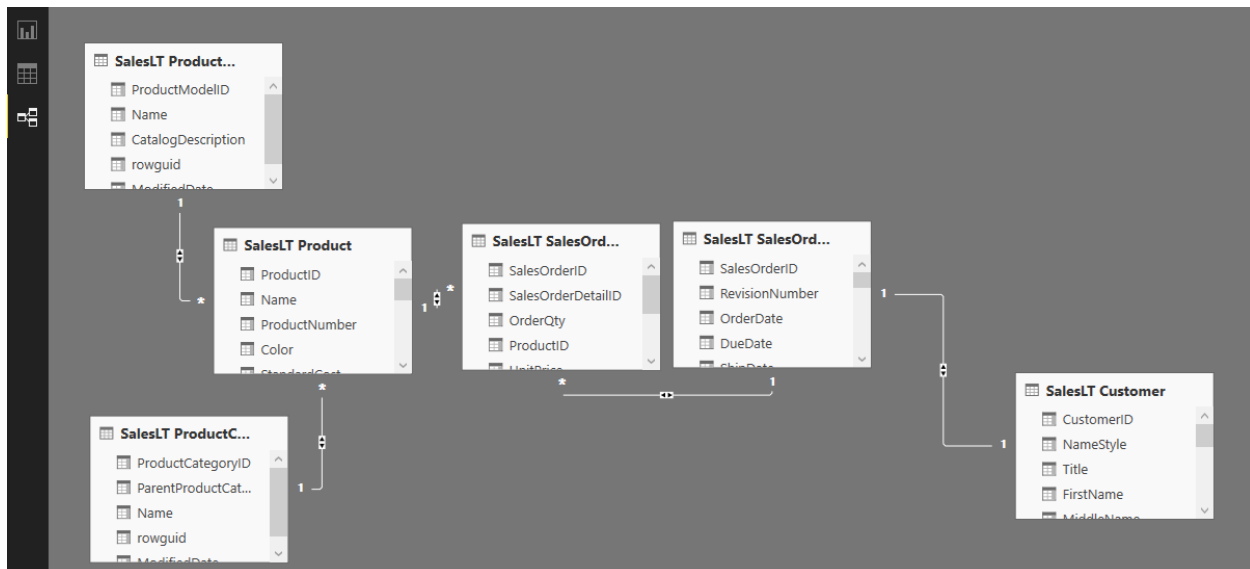


Now you can try again to connect from Power BI Desktop (if you get that error again, wait for few minutes and try again. Sometimes it takes few minutes for changes to take effect). After a successful connection, you should be able to see Navigator dialog box with the structure of AdventureWorksLT database

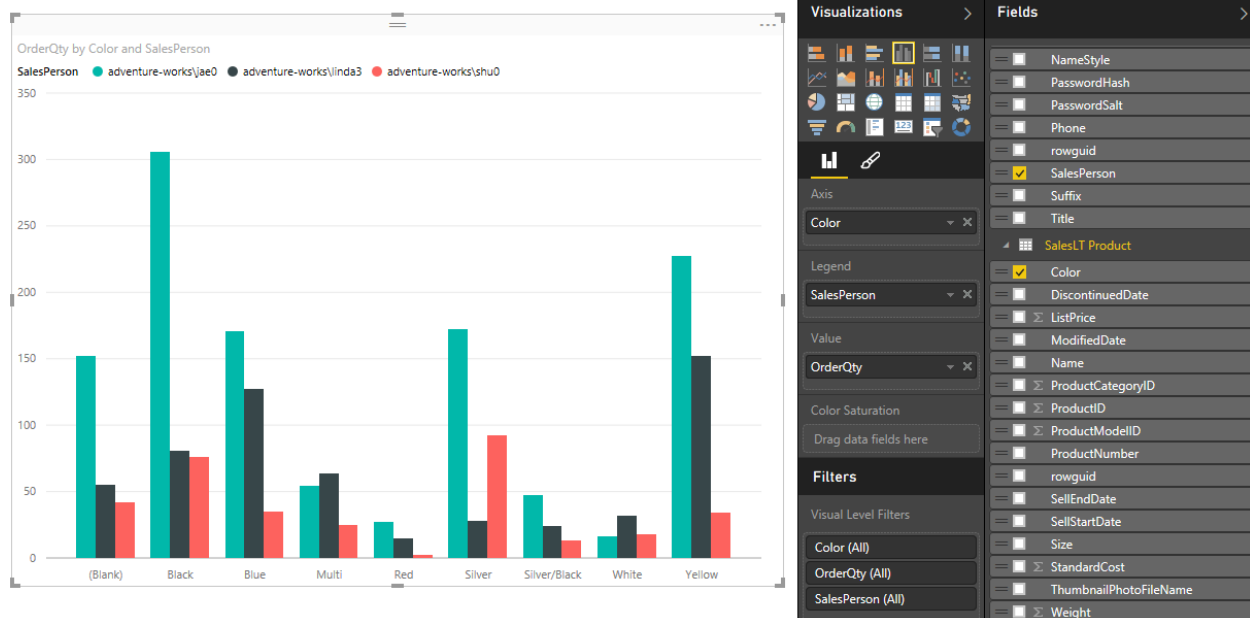


As you can see in the navigator all views, tables and functions will be listed. You can choose multiple objects and then continue editing them in the Edit Queries or Power Query Editor window. For this example, I've chosen these tables: Customer, Product, ProductCategory, ProductModel, SalesOrderDetail, and SalesOrderHeader.

You can then choose these tables with their fields to be used for building a report without any modification in Power Query or Data tab. You can even see that Power BI and Power Query understand the relationship in Azure SQL Database and load the same relationship in the Power BI model.



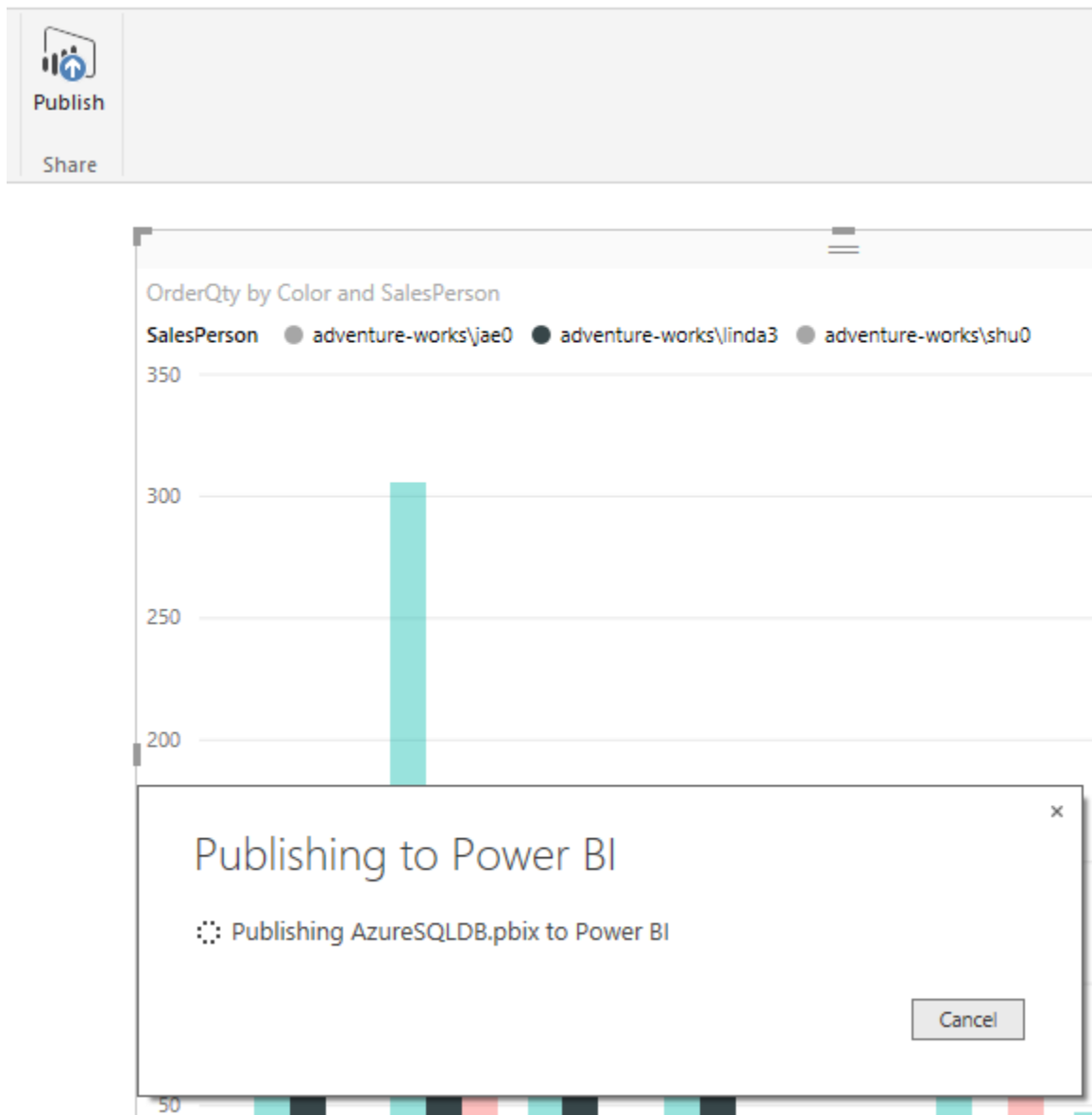
This section is still one of the early chapters of Power BI online book, and I don't want to discuss visualization and modeling. However for this example, I've built a simple chart, the chart is a clustered column chart with Color (from Product table) as Axis, and SalesPerson (from Customer) as Legend, and OrderQty (from SalesOrderDetail) as Value.



The chart is simple, but still revealing something interesting. Jae0 made the most sales, however color wise each sales person did the best in a specific color. shu0 was best at Yellow. lina3 at Silver, and Jae0 at Black. Now you can



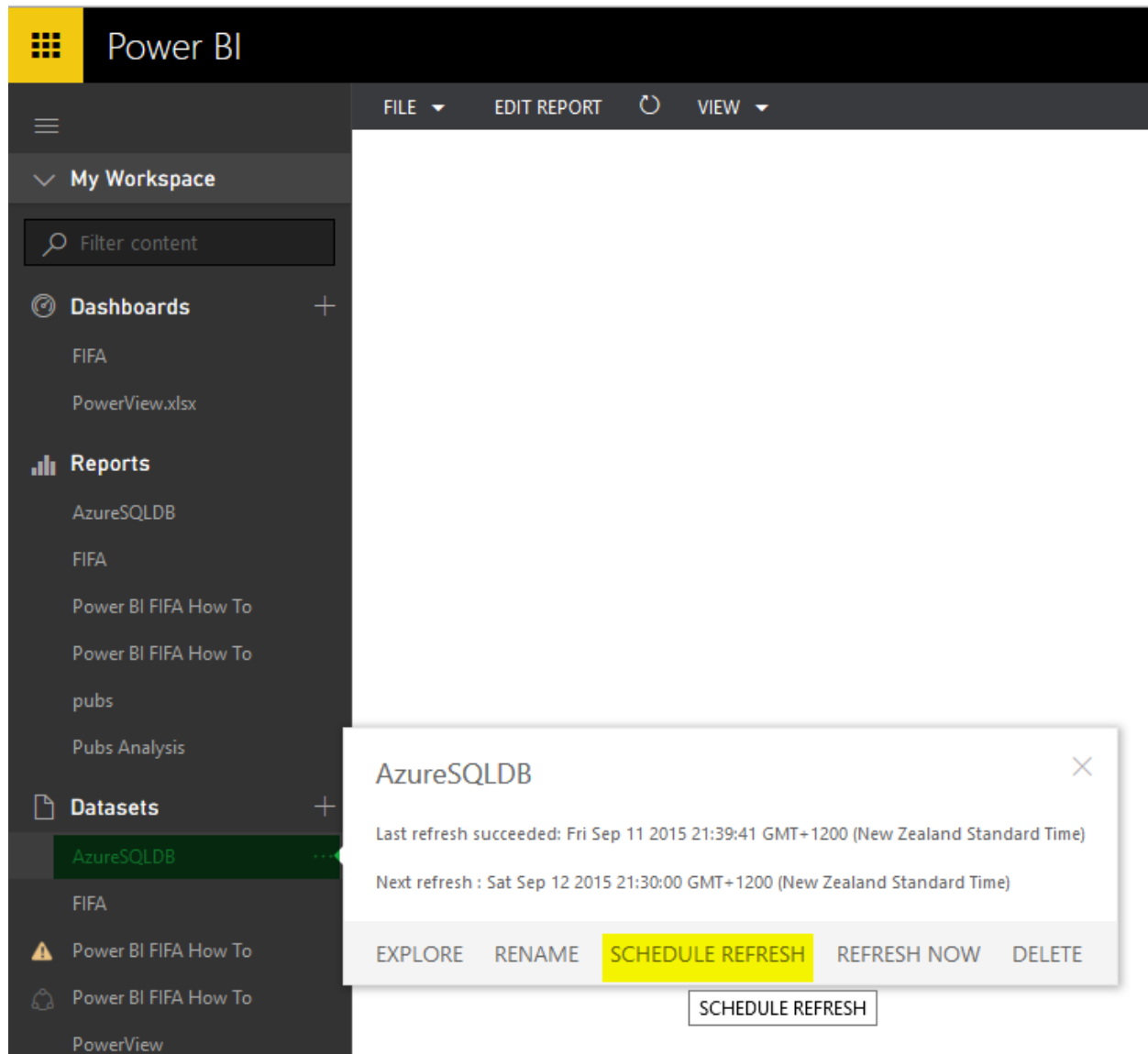
publish your report into Power BI website with the Publish menu option. After publishing the report, you would be able to see that under your Power BI account in the website.



## Schedule Data Refresh

In Power BI website you can set up scheduled data refresh for data sets, but not all datasets support this feature. Fortunately, Azure SQL Database

supports it. To find out the list of all data sets that support data refresh read [this link](#). To schedule, a refresh in Power BI website, under Datasets click on ellipsis besides the data source that you want. And then choose Schedule Refresh.



Set the Data Source Credentials for Azure SQL Database

## Settings

General Dashboards **Datasets** Workbooks

AzureSQLDB

FIFA

Power BI FIFA How To

PowerView

pubs

Pubs Analysis

## Settings for AzureSQLDB

Last refresh succeeded: Fri Sep 11 2015 21:39:41 GMT+1200 (New Zealand Standard Time)  
Next refresh: Sat Sep 12 2015 21:30:00 GMT+1200 (New Zealand Standard Time)

## Data Source Credentials

AdventureWorksLT-rac

## Schedule Refresh

## Configure AzureSQLDB

Server

radacad-adf.database.windows.net

Database

AdventureWorksLT

Authentication Method:

Basic

Username

reza

Password

••••••••

Sign In

Cancel

And then you can schedule refresh. You can choose the frequency to be daily or weekly. And you can add multiple times on the day under that.

## Settings for AzureSQLDB

Last refresh succeeded: Fri Sep 11 2015 21:39:41 GMT+1200 (New Zealand Standard Time)

Next refresh: Sat Sep 12 2015 21:30:00 GMT+1200 (New Zealand Standard Time)

### ▲ Data Source Credentials

AdventureWorksLT-radacad-adf.database.windows.net

[Edit credentials](#)

#### ▲ Schedule Refresh

Keep your data up-to-date

Yes



Refresh frequency

Daily

Time Zone

(UTC+12:00) Auckland, Wellington

Time

9 30 PM X

[Add another time](#)

☒ Send refresh failure notification email to me

[Refresh History](#)

Apply

Discard

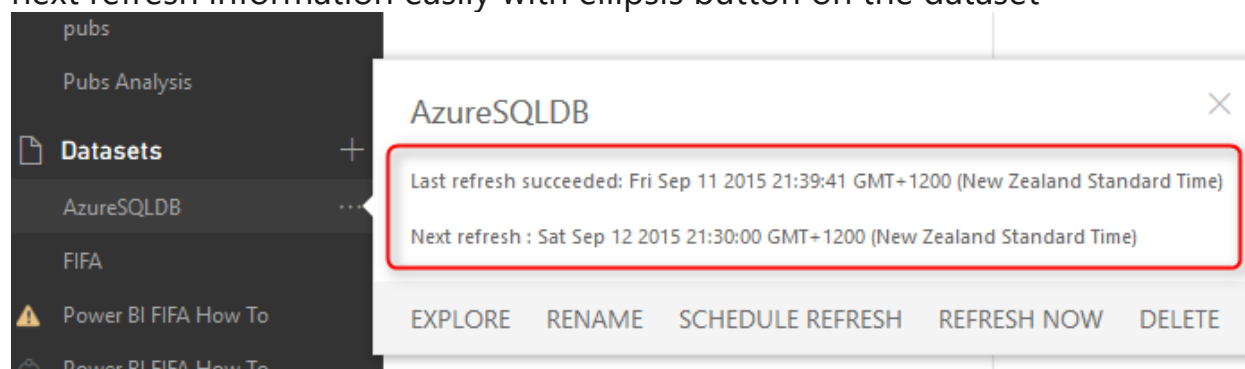
If you have a scheduled refresh set up for a while, then you can see the history of a refresh as well

## Refresh History

Details	Type	Start	End	Status	Fail Message
	Scheduled	9/11/2015, 9:39:01 PM	9/11/2015, 9:39:41 PM	Completed	
	Scheduled	9/10/2015, 9:40:00 PM	9/10/2015, 9:40:32 PM	Completed	
	Scheduled	9/9/2015, 9:42:03 PM	9/9/2015, 9:42:28 PM	Completed	
	Scheduled	9/8/2015, 9:43:01 PM	9/8/2015, 9:43:28 PM	Completed	
	Scheduled	9/7/2015, 9:44:00 PM	9/7/2015, 9:44:29 PM	Completed	
	Scheduled	9/6/2015, 9:35:05 PM	9/6/2015, 9:35:46 PM	Completed	
	Scheduled	9/5/2015, 9:35:02 PM	9/5/2015, 9:35:30 PM	Completed	
	Scheduled	9/4/2015, 9:45:02 PM	9/4/2015, 9:45:33 PM	Completed	
	Scheduled	9/3/2015, 9:45:04 PM	9/3/2015, 9:45:57 PM	Completed	
	Scheduled	9/2/2015, 9:31:00 PM	9/2/2015, 9:31:34 PM	Completed	

Close

After setting up the schedule refresh, you can see the latest refresh and the next refresh information easily with ellipsis button on the dataset

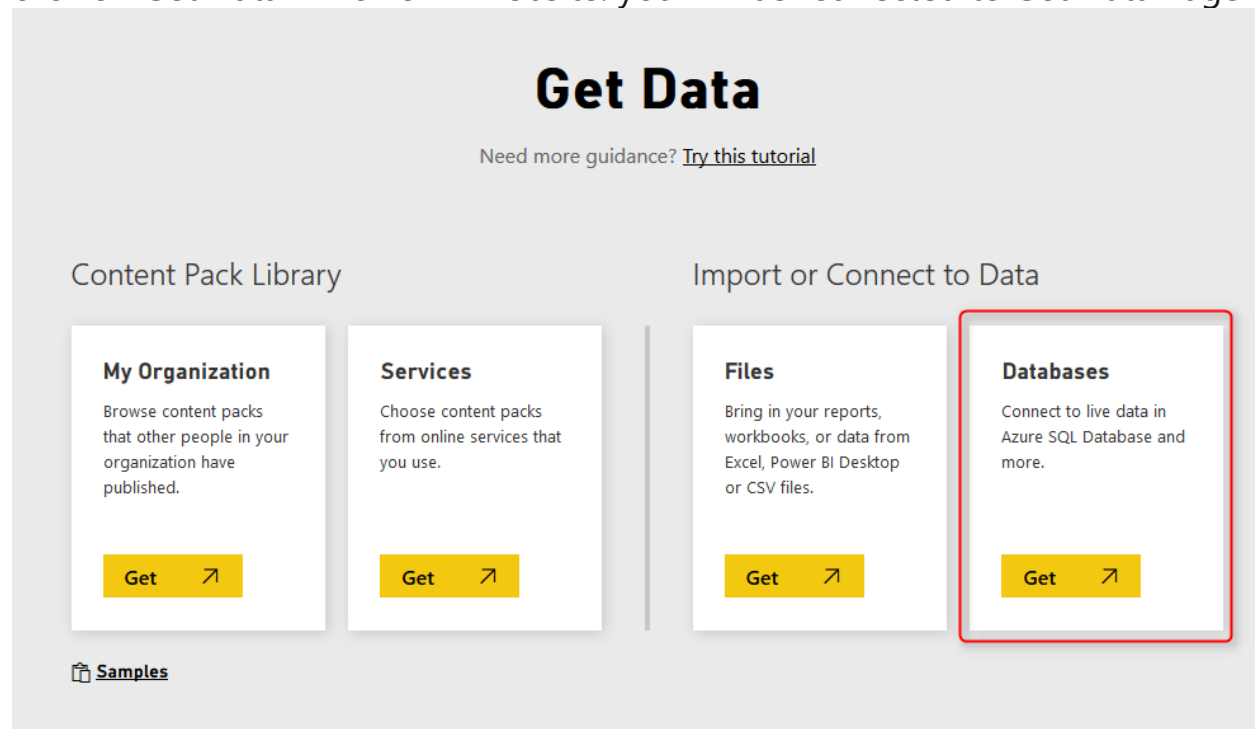


## Direct Connection to Azure SQL Database from Power BI Website

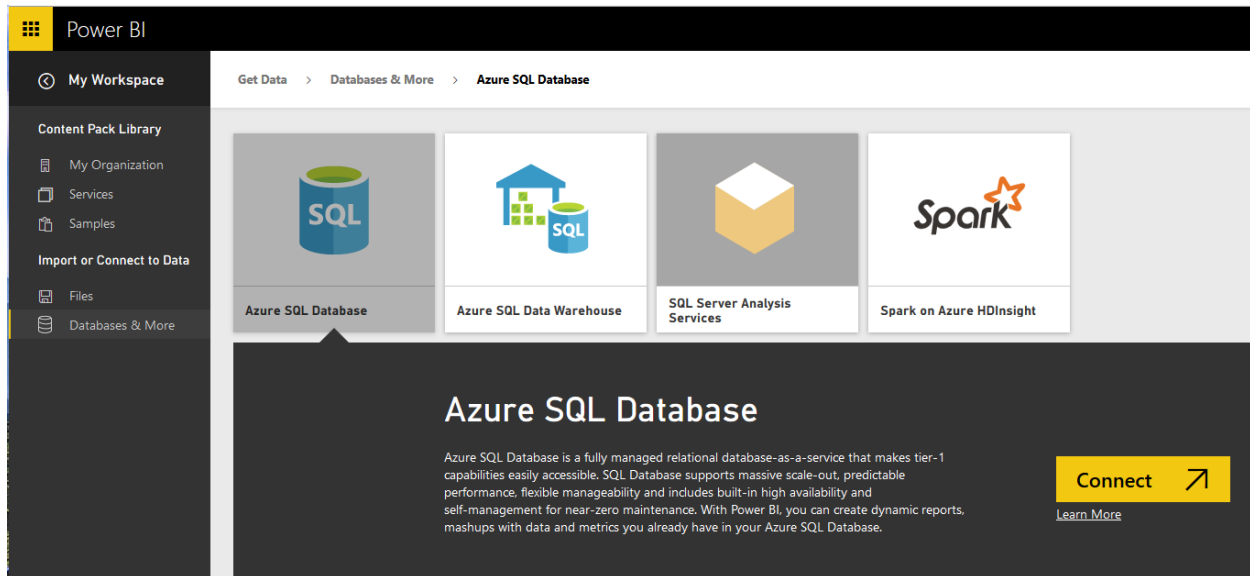
So far you've learned how to connect from Power BI Desktop or Power Query to Azure SQL database. However, the connection in that way is off-line, and you need to set up a scheduled refresh to keep data up-to-date. Fortunately, there is another way of connection which is Direct Connection. The direct

connection won't load data into Power BI model, it directly brings data into the report, and you won't need to schedule refresh anymore, because data is always up-to-date. There will be however a lag for loading the report depends on the data and volume required loaded in the report.

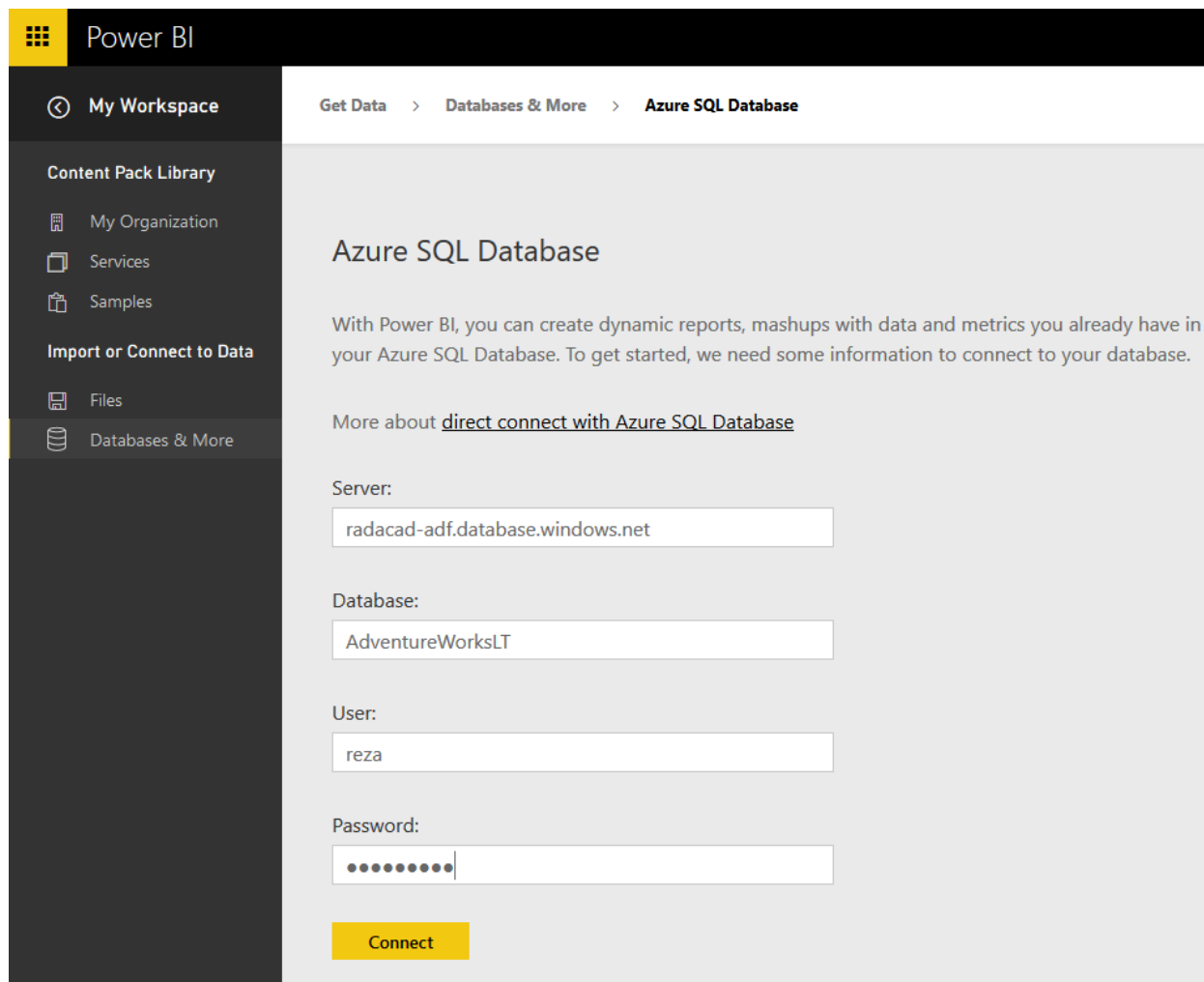
You can set Direct connection only from Power BI website at the moment. And this is one of Power BI Pro features, so you need to buy the pro plan (which costs 9.99\$ per user per month at the moment). To set a direct connection, click on Get Data in Power BI website. you will be redirected to Get Data Page



Click on the Get option from Databases. You can see that some databases are supported through this type of connection; Azure SQL Database, Azure SQL Data Warehouse, SQL Server Analysis Services, and Spark on Azure HDInsight. Click on Azure SQL Database and then Connect.



Set the connection information such as the server, database name, and user and password



Power BI

My Workspace

Content Pack Library

- My Organization
- Services
- Samples

Import or Connect to Data

- Files
- Databases & More

Get Data > Databases & More > Azure SQL Database

### Azure SQL Database

With Power BI, you can create dynamic reports, mashups with data and metrics you already have in your Azure SQL Database. To get started, we need some information to connect to your database.

More about [direct connect with Azure SQL Database](#)

Server:

radacad-adf.database.windows.net

Database:

AdventureWorksLT

User:

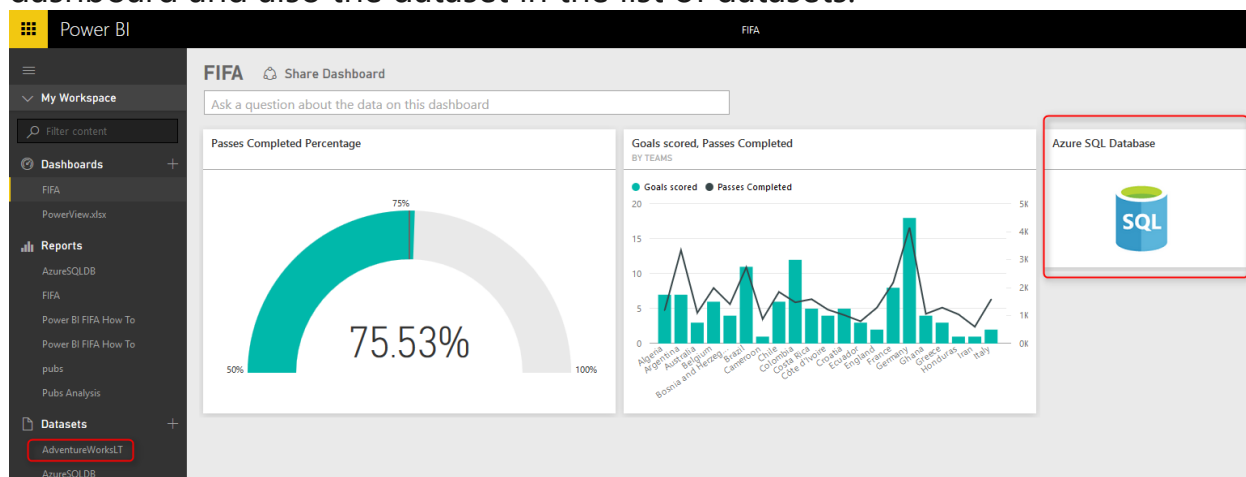
reza

Password:

●●●●●●●●

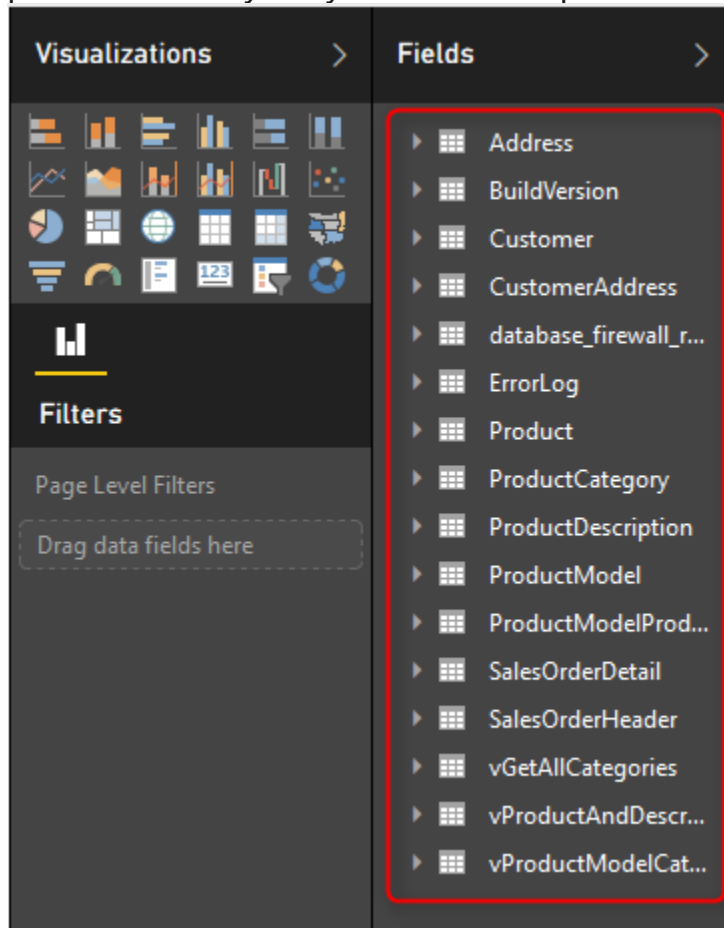
Connect

After creating the connection, you will see Azure SQL Database in your dashboard and also the dataset in the list of datasets.





by click on any of these items, you will be redirected to online Power BI report designer. You can see that all tables and views now are listed in the Fields pane, and ready for you to build reports from them.



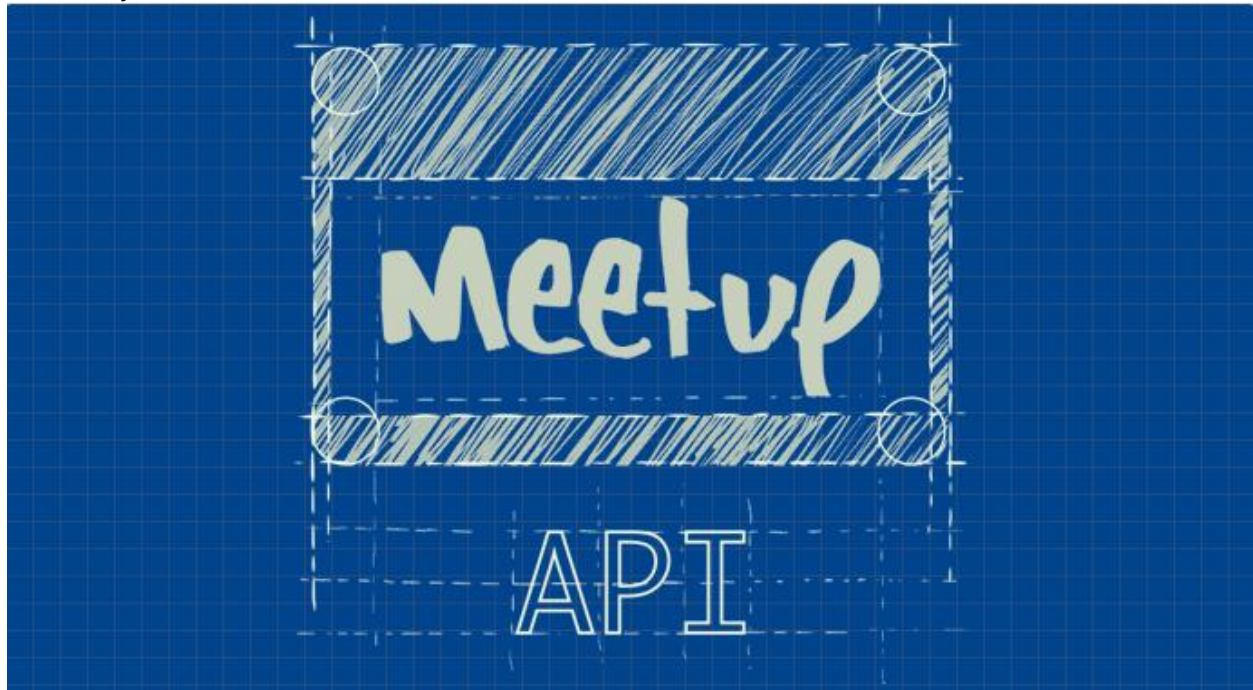
## Summary

In this lesson, you've learned how to the connection from Power BI Desktop or Power Query in Excel to Azure SQL Database. You've also learned some basics about Azure SQL Database settings in Azure Portal. Then you've learned that you need to set firewall in Azure Portal for the server to pass your IP for connections. You then learned how you could set scheduled refresh for the data in Power BI website. At the last section, you've learned that you can create a direct connection from Power BI website to Azure SQL Database that

doesn't load data into a model and works online with the data. In the next sections of this chapter, you will learn about other data sources that you can connect from Power BI.

# Meetup Data Source for Power BI

Posted by [Reza Rad](#) on Jun 2, 2016

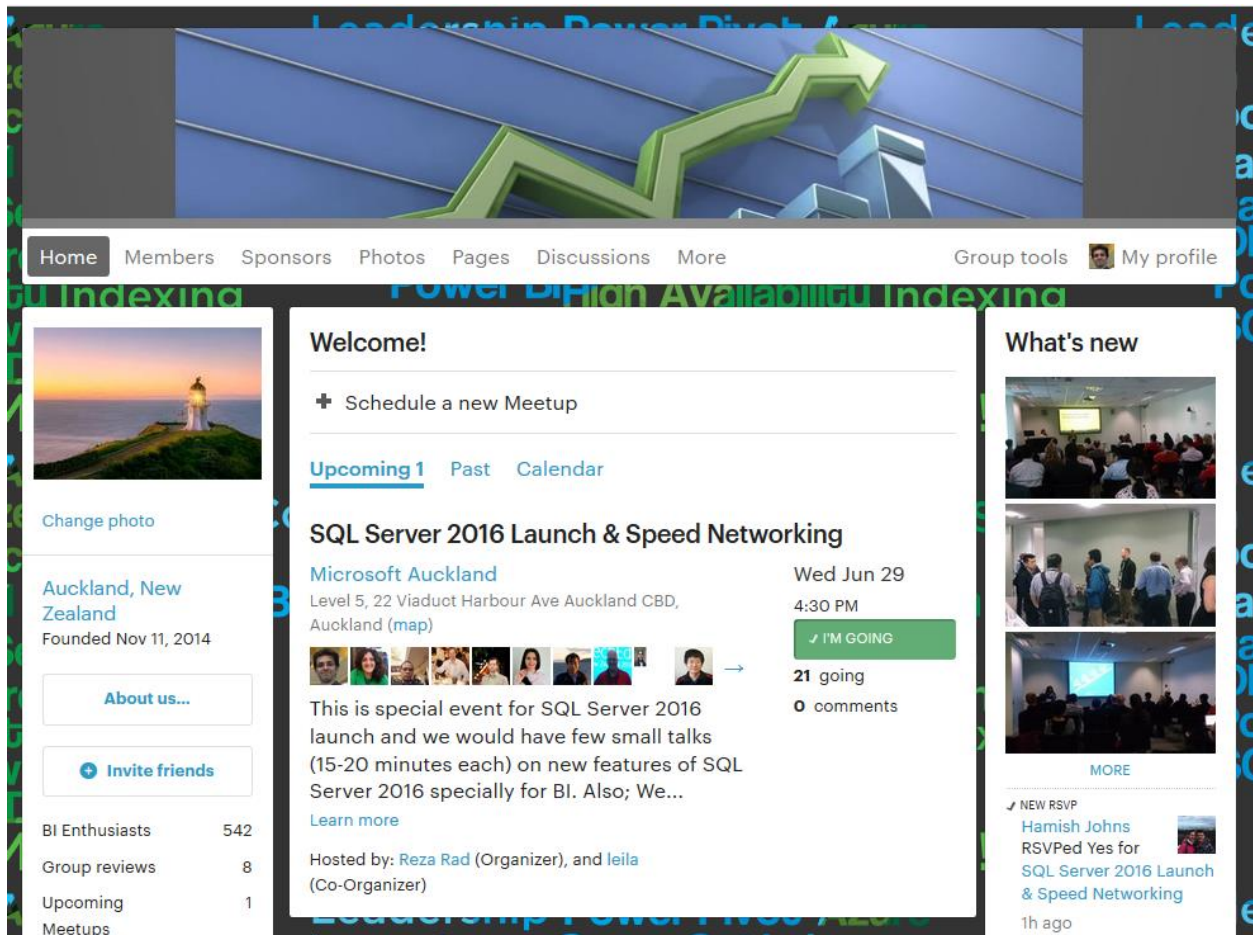


There are many SaaS (Software as a Service) data sources for Power BI, such as MailChimp, Salesforce, etc. However I haven't found anything for Meetup. Meetup, on the other hand, is a data source that event organizers such as myself use mostly. In this post, I will be using Meetup API to connect from Power BI Desktop and read the JSON output of Meetup API to build some nice visualizations in Power BI. You can read more about Power Query in [Power BI online book](#) from Rookie to Rock Star.

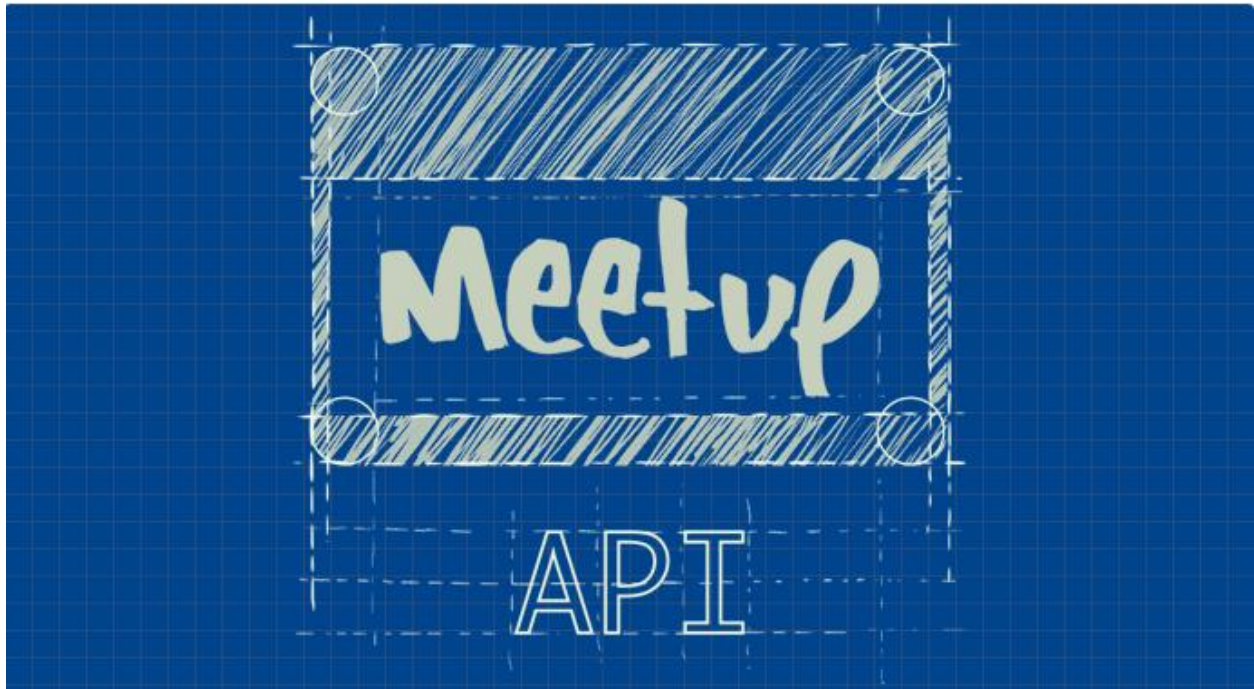
## About Meetup

If you haven't heard about [Meetup](#) so far; it is a platform for scheduling in-person events. As an organizer, I announce meetings for my Meetup group (such as [New Zealand BI user group](#)), and Meetup helps with the registration for the event. As an audience, I can join to groups in my area of interest and be informed about their upcoming meetings and RSVP those to attend or not attend. Meetup website itself gives some useful information to organizers,

however, as a Power BI geek, I would love to dig into its data with a better tool.

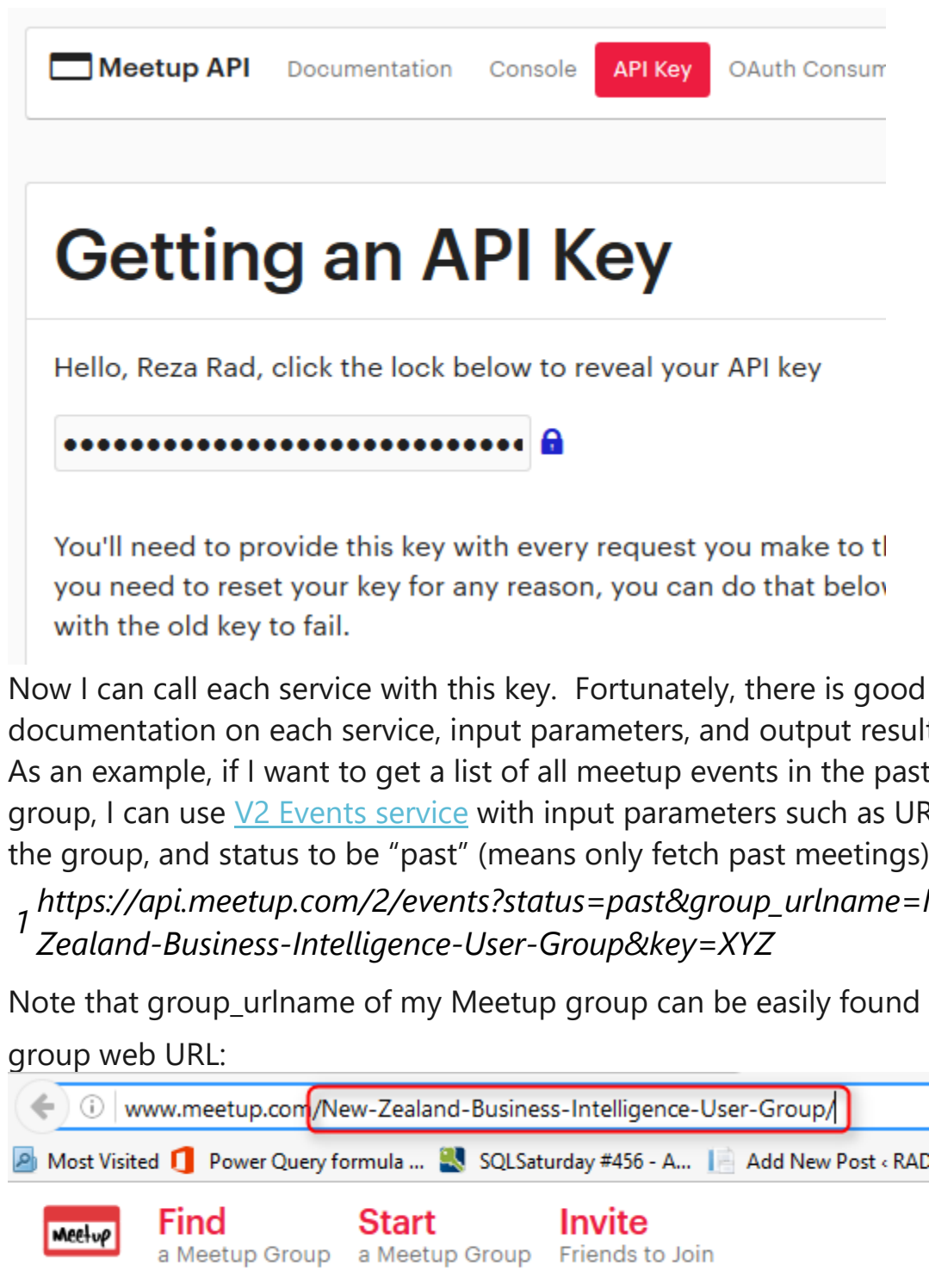


Fortunately, [Meetup has a set of API](#) restful services which returns output as JSON format.



Restful service means there are some URLs that I can browse, and it gives me a response in JSON format. First of all, I need to use my API key. API key would be different for anyone, and sorry I cannot share my API key with you, that's why it is blurred in the screenshot below. You can access your API key from [here](#).





Meetup API Documentation Console **API Key** OAuth Consum

## Getting an API Key

Hello, Reza Rad, click the lock below to reveal your API key

..... 🔒

You'll need to provide this key with every request you make to t... you need to reset your key for any reason, you can do that below with the old key to fail.

Now I can call each service with this key. Fortunately, there is good documentation on each service, input parameters, and output result set [here](#). As an example, if I want to get a list of all meetup events in the past for my group, I can use [V2 Events service](#) with input parameters such as URL name of the group, and status to be "past" (means only fetch past meetings);

`https://api.meetup.com/2/events?status=past&group_urlname=New-Zealand-Business-Intelligence-User-Group&key=XYZ`

Note that group\_urlname of my Meetup group can be easily found on my group web URL:

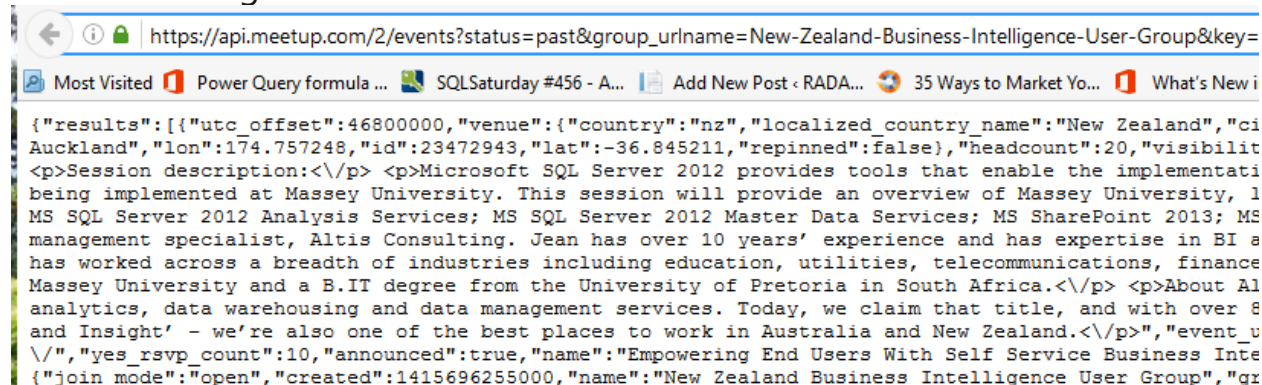
`www.meetup.com/New-Zealand-Business-Intelligence-User-Group/`

Find a Meetup Group Start a Meetup Group Invite Friends to Join

as per the screenshot above my group URL name is: New-Zealand-Business-Intelligence-User-Group

and you have to use your own API Key instead of XYZ in URL above.

So if I browse URL above it gives me a JSON result. JSON is a format for data which usually doesn't have enter or space between fields, so as a result, I would see a big text result like this:




## Get Data with Power BI

Now let's browse URL above in Power BI and Power Query to see what can I do with it. I open a Power BI Desktop file and Get Data from the Web. And I enter URL above there (with my API key obviously);

### From Web

Enter a Web page URL.

URL

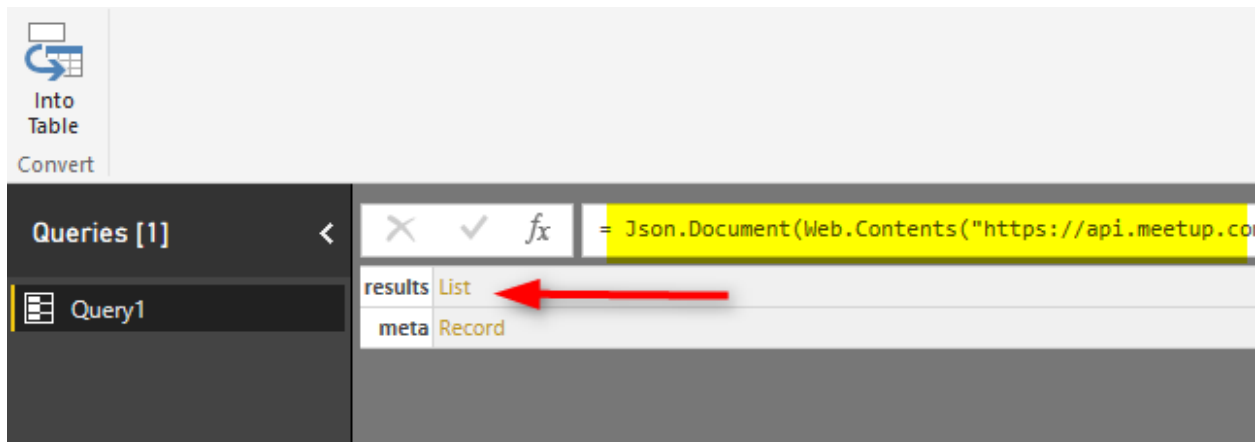


☒ Basic ☐ Advanced

OK

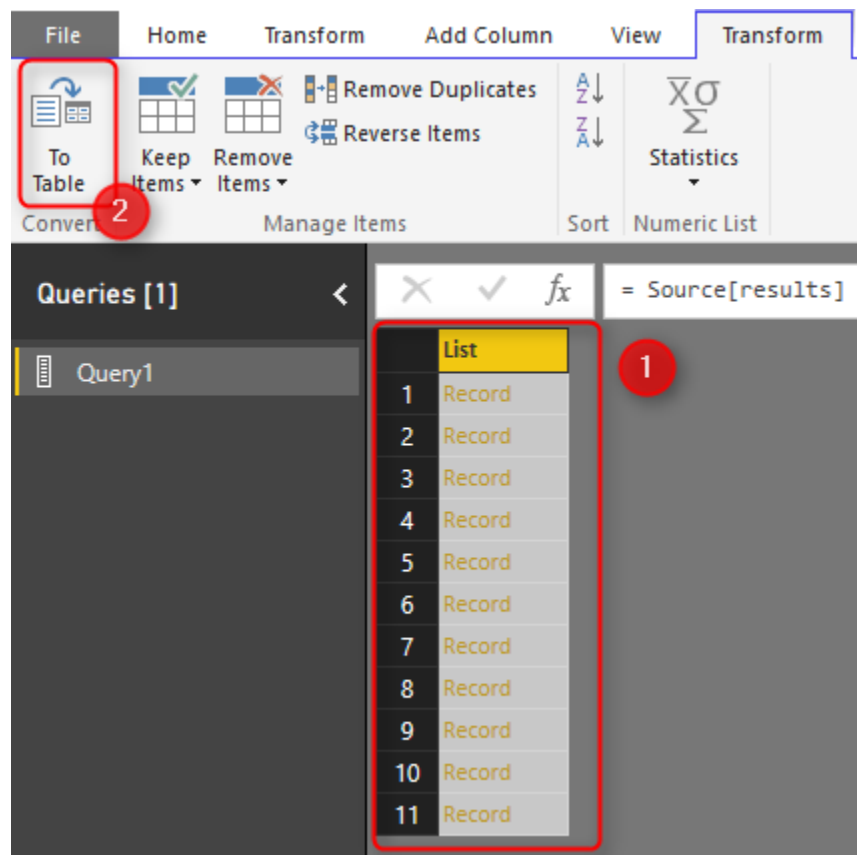
Cancel

The result of this will show me a record in Power Query editor window with results and meta columns. You can also see in the screenshot below that Power Query uses *JSON.Document* and *Web.Contents* to get JSON result from a web URL.



Now click on the results section List to expand that. This will drill down into the list, and you will see a list with multiple items in the next step. This list is a result set in our JSON data. The result set of the URL we run returns a list of events. That means this list is a list of events, and each row in this list is an event. As you can see I have a record in each row. I can convert the main list to the table with first highlighting the whole column, and then click on To Table option from the Transform menu in Power Query.





When I click on To Table, I will see To Table dialog box that asks for two configuration options, first one for the delimiter, and a second one for handling extra columns.

## To Table

Create a table from a list of values.

Select or enter delimiter

None

How to handle extra columns

Show as errors

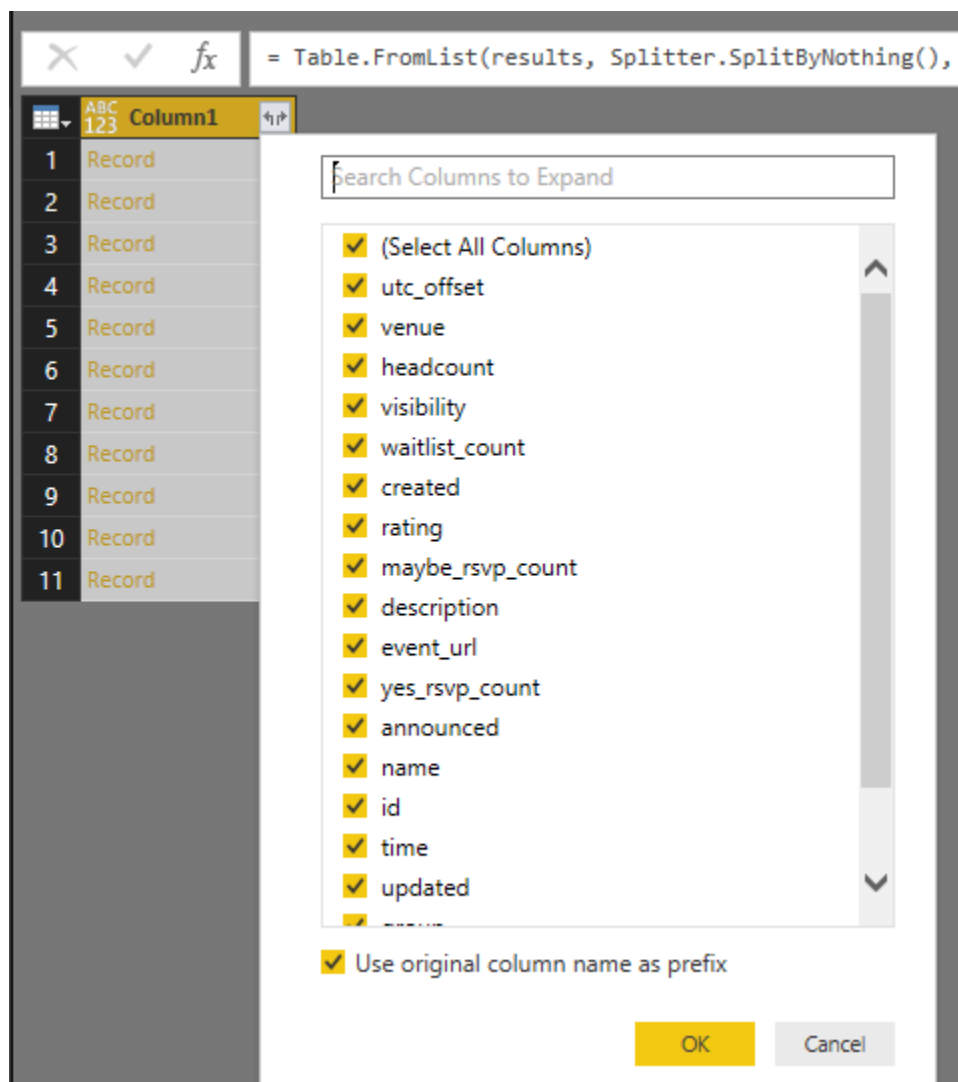
OK

Cancel

I don't do any changes and click on OK with default values. The result now would be a table with a single column, which has a record in each row. I can see the Expand button on my column1 header.



Click on the expand button as mentioned in the screenshot above, and it will list all columns in records and asks me to choose those I want.



Well let's bring them all to see what we have, so I leave it with select all and click on OK. Here is my result set loaded in Power Query now:

	Column1.utc_offset	Column1.venue	Column1.headcount	Column1.visibility	Column1.waitlist_count	Column1.created	Column1.rating	Column1.maybe_rsvp_count	Column1.description
1	46800000	Record	20	public	0	1.4157E+12	Record	0	<p>Session description</p><p>Micro
2	43200000	Record	0	public	0	1.42778E+12	Record	0	<p>Session Description</p><p>This p
3	43200000	Record	0	public	0	1.43121E+12	Record	0	Speaker</p><p>Darryl Wolfardt</p>
4	43200000	Record	0	public	0	1.43566E+12	Record	0	<p><b>Session Details</b></p><p><b>H
5	43200000	Record	0	public	0	1.43851E+12	Record	0	<p>Session Description</p><p>Power
									Reza Rad is an Author, Trainer, Speake
									<p>Session Description</p><p>Mach

Now I have event information such as Name, Description, venue and other fields. Some values are a record by themselves, such as Venue and Rating. I can expand them if I want to. Let's leave it as is for now. I've loaded my first service result into Power Query. Now one important thing before any other

step is to check time value. Because I want to do date/time analysis for my events, so it is important to see if I have my time value fetched. In the screenshot below it shows my Time column as a numeric value.

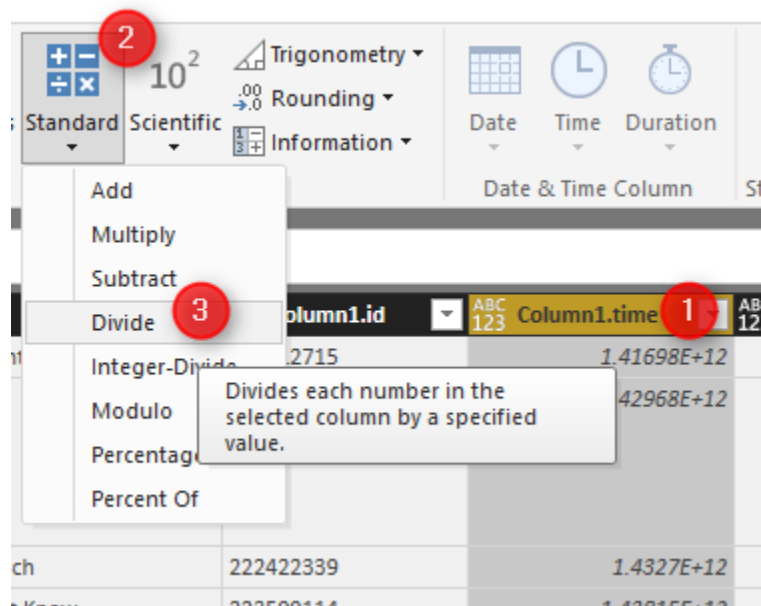
ABC 123 Column1.name	ABC 123 Column1.id	ABC 123 Column1.time	ABC 123 Column1.updated
Empowering End Users With Self Service Business Intelligence	218612715	1.41698E+12	1.46412E+12
BIML is the shizzle!!	221511579	1.42968E+12	1.42969E+12
Show Me Potential Customers: Data Mining Approach	222422339	1.4327E+12	1.43286E+12
Top 5 Functionalities of Power Query that You Don't Know	223599114	1.43815E+12	1.43816E+12
Be Smarter with Azure Machine Learning	224346643	1.44057E+12	1.44221E+12
SQL Saturday Auckland	224997769	1.44382E+12	1.44391E+12

## TimeStamp to Date Time

In the documentation of this service in Meetup mentioned that Time column is:

time = UTC start time of the event, in milliseconds since the epoch

That means it is timestamp formatted. The timestamp value is a number of seconds from epoch which is 1970-01-01 00:00:00. [I have previously written about how to change timestamp value to date time](#), and it is fairly easy with adding seconds to it. However for this case, our value is not seconds, it is milliseconds, so I have to first divide it by 1000.



This makes my number smaller now, and I can now feed it into adding a custom column as Event Time as below;

```
1 #datetime(1970,1,1,0,0,0)+#duration(0,0,0,[Column1.time])
```

## Add Custom Column

New column name

Event Time

Custom column formula:

```
=#datetime(1970,1,1,0,0,0)+#duration(0,0,0,[Column1.time])
```

Available columns:

Column1.name  
Column1.id  
Column1.time  
Column1.updated  
Column1.group  
Column1.status  
Column1.duration

<< Insert

[Learn about Power BI Desktop formulas](#)

✓ No syntax errors have been detected.

OK

Cancel

If you want more information about this conversion read my blog post [here](#).

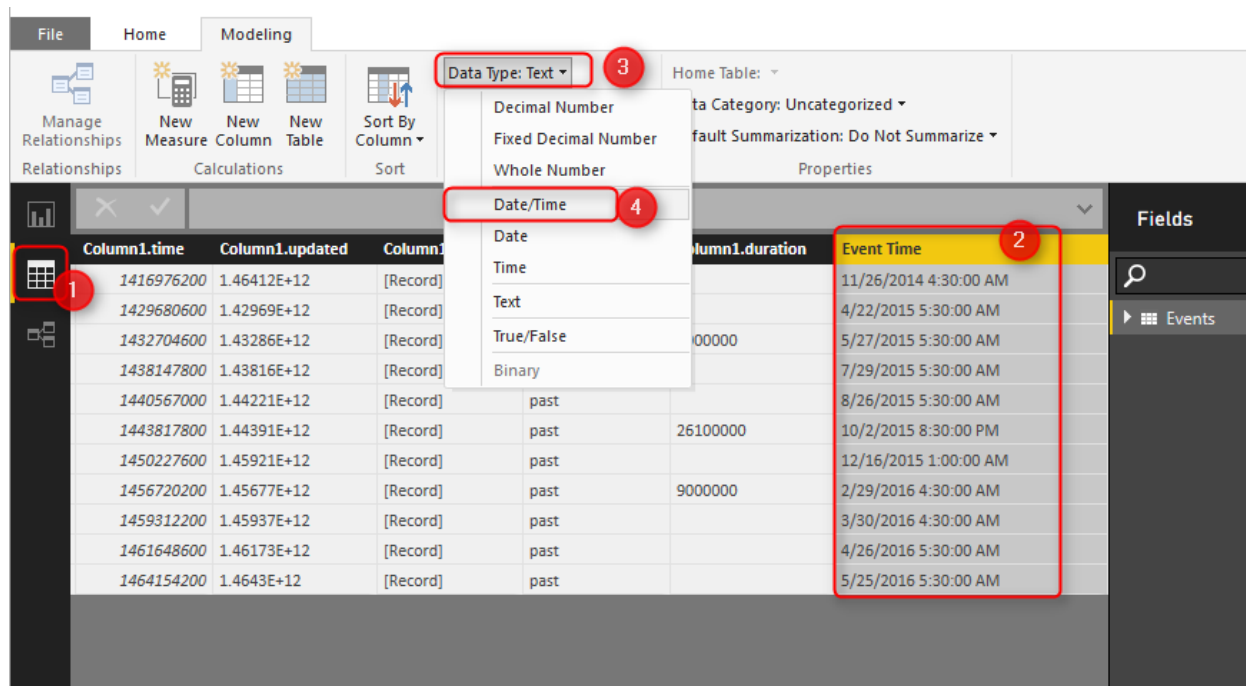
Now this will give me a date-time value as below:

ABC 123	Event Time
I/	11/26/2014 4:30:00 AM
I/	4/22/2015 5:30:00 AM
J	5/27/2015 5:30:00 AM
I/	7/29/2015 5:30:00 AM
I/	8/26/2015 5:30:00 AM
J	10/2/2015 8:30:00 PM

Please note that times are UTC times, We never have meetings 4 AM! I won't be changing UTC time for now, because I want more Month analysis and this is fine for that analysis.

## Visualization

Let's visualize what we've got so far. The first step I have to make sure the data type of my column in Power BI model loaded as Date Time. I can go to Data Tab, click on my Event Time column and under Modeling tab check the data type, and if it is text or anything else, change it to Date/Time

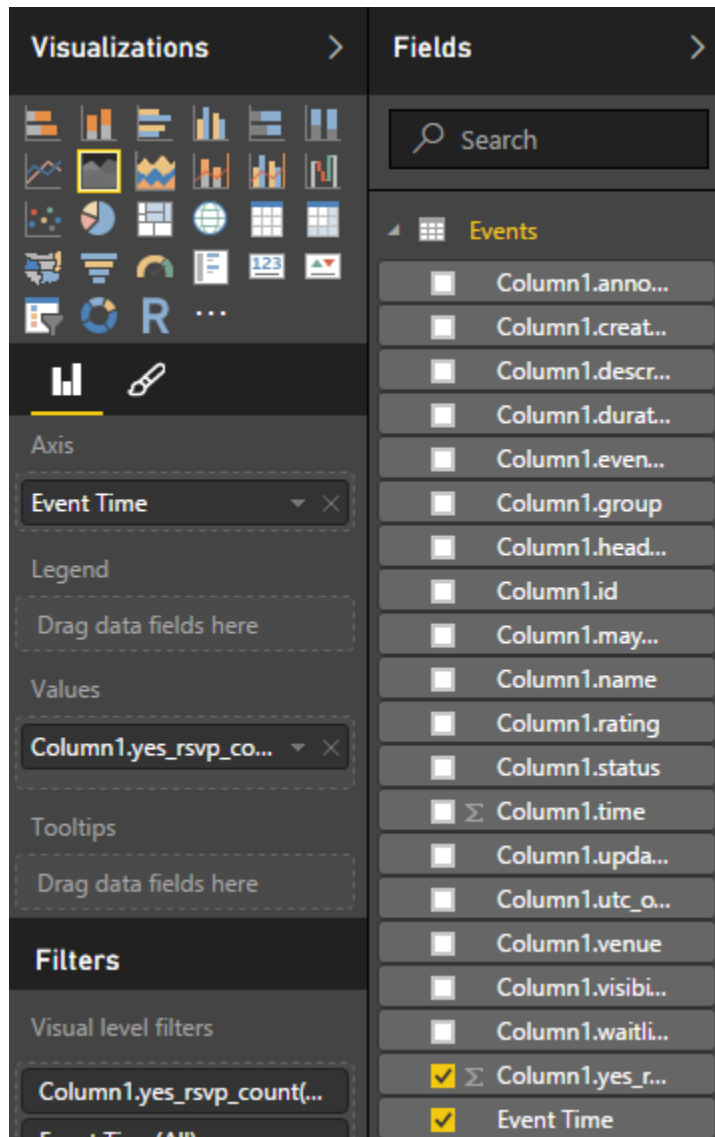


a warning message will be displayed for reports that are already using this field, as I don't have any report elements yet, I'll confirm and continue. I also have to make sure that yes\_rsvp\_count column (which is showing a number of audience for each session) set as data type whole number. If it is not the case, then I change it to be the whole number as below.

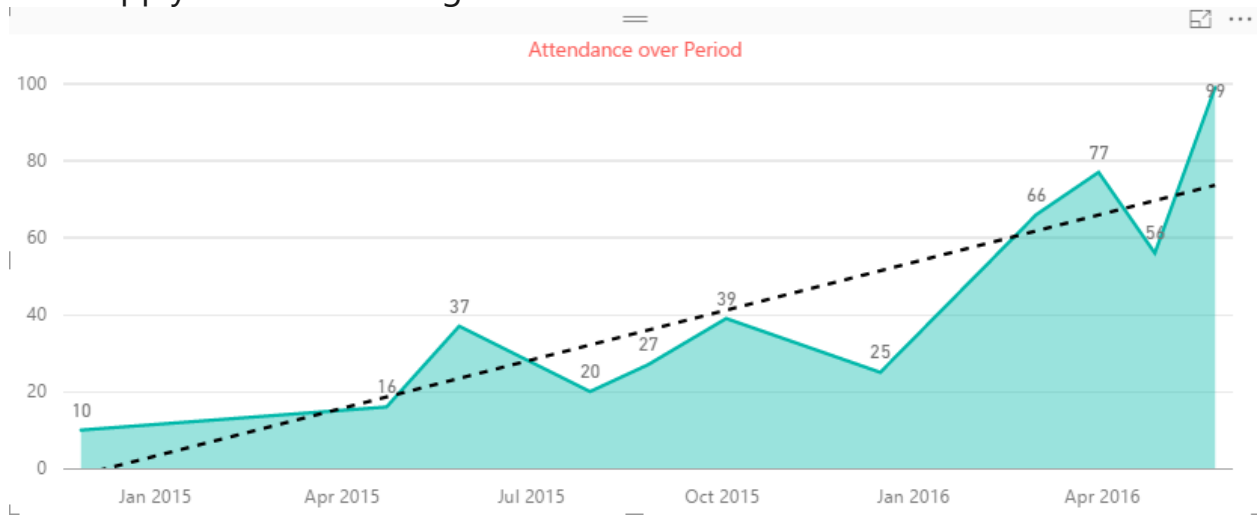
<b>Data Type: Whole Number</b> ▾ Format: Whole Number ▾ \$ ▾ % ▾ , ▾ .00 0 ▴ ▾ Formatting		Home Table: ▾ Data Category: Uncategorized ▾ Default Summarization: Sum ▾ Properties
Column1.event_url		Column1.yes_rsvp_count
er 2012 provid	http://www.meetup.com/New-Zealand-Business-Intelligence-User	10
will be a deep c	http://www.meetup.com/New-Zealand-Business-Intelligence-User	16
arketing depart	http://www.meetup.com/New-Zealand-Business-Intelligence-User	37
ic nowadays, P	http://www.meetup.com/New-Zealand-Business-Intelligence-User	20
rovides a facili	http://www.meetup.com/New-Zealand-Business-Intelligence-User	27
rofessionals an	http://www.meetup.com/New-Zealand-Business-Intelligence-User	39
ent for 2015, a	http://www.meetup.com/New-Zealand-Business-Intelligence-User	25
ptional opport	http://www.meetup.com/New-Zealand-Business-Intelligence-User	66
een Microsoft S	http://www.meetup.com/New-Zealand-Business-Intelligence-User	77
ous who would	http://www.meetup.com/New-Zealand-Business-Intelligence-User	56
you can Do wit	http://www.meetup.com/New-Zealand-Business-Intelligence-User	99

Let's build charts now. I start by an Area Chart to show the Event Time as an Axis (without hierarchy), and yes\_rsvp\_count as value.

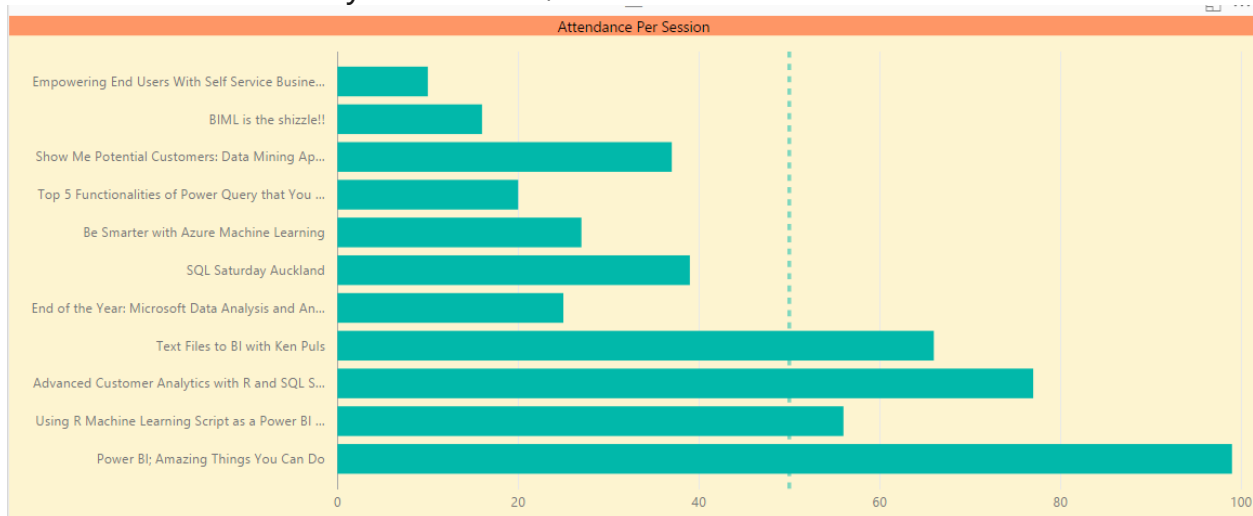




I also apply some formatting and build a chart like this:



Well, it shows our group is performing well, the number of the audience raised from 10 to 99! and trend line shows that we are progressing better and better. I can build some other visualizations such as bar chart with the name of events and number of RSVP yes as below;



I can see the very first event that had more than 50 audiences was the event that my friend [Ken Puls Excel MVP](#) from Canada came here to speak in our user group, which was a start into our popular events afterward.

## Adding More Data

I can add more data to build more meaningful insight out of it. For example, I can fetch Members information to show their locations (city and country) on the map. For this I use Members API service which I can call as below;

1 [https://api.meetup.com/members?group\\_urlname=New-Zealand-Business-Intelligence-User-Group&key=XYZ](https://api.meetup.com/members?group_urlname=New-Zealand-Business-Intelligence-User-Group&key=XYZ)

fx = Table.ExpandRecordColumn(#"Converted to Table", "Column1",

	ABC 123 Column1.zip	ABC 123 Column1.country	ABC 123 Column1.city	ABC 123 Column1.joined	ABC 123 Column1.topics
1	meetup1	nz	Auckland	Thu Nov 26 17:29:49 EST 2015	List
2	meetup1	nz	Auckland	Wed Apr 08 06:49:50 EDT 2015	List
3	meetup1	nz	Auckland	Thu Dec 17 02:29:11 EST 2015	List
4	meetup1	nz	Auckland	Sun May 24 23:49:20 EDT 2015	List
5	meetup1	nz	Auckland	Mon May 02 21:29:32 EDT 2016	List
6	meetup1	nz	Auckland	Tue Jan 19 06:24:15 EST 2016	List
7	1150	nz	Auckland	Thu Jan 28 12:06:41 EST 2016	List
8	meetup1	nz	Auckland	Thu Jun 25 20:22:03 EDT 2015	List
9	meetup1	nz	Auckland	Sat Apr 16 18:11:17 EDT 2016	List
10	meetup1	nz	Auckland	Thu Nov 12 04:05:07 EST 2015	List
11	meetup1	nz	Auckland	Wed Sep 09 22:32:28 EDT 2015	List

This time I haven't explained step by step, but the process is similar to what we've done for Events, so I skip this part. Now I want to create a custom column concatenated of city and country as below;

### Add Custom Column

New column name

City Full

Custom column formula:

= [Column1.city]&" "&[Column1.country]

Available columns:

Column1.zip  
Column1.country  
Column1.city  
Column1.joined  
Column1.topics  
Column1.link  
Column1.bio

<< Insert

[Learn about Power BI Desktop formulas](#)

✓ No syntax errors have been detected.

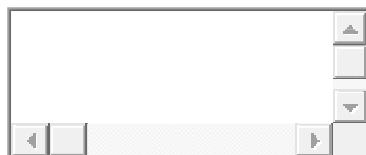
OK

Cancel

I then will be using this new City Full column because it gives me a more reliable map location than the city itself (because a city with the same name might be part of two different countries).

## RSVP Data

RSVP data for each event is really useful for me because I can understand who attend in most events, and then contact them and thanks them for their commitment to the group. RSVP data can be fetched for each event. So I have to check it for each event ID that what is the RSVP for it. Here is RSVP service URL:



1 [https://api.meetup.com/2/rsvps?event\\_id=230678189&key=XYZ](https://api.meetup.com/2/rsvps?event_id=230678189&key=XYZ)

Note that you have to use your event\_id for the URL above.

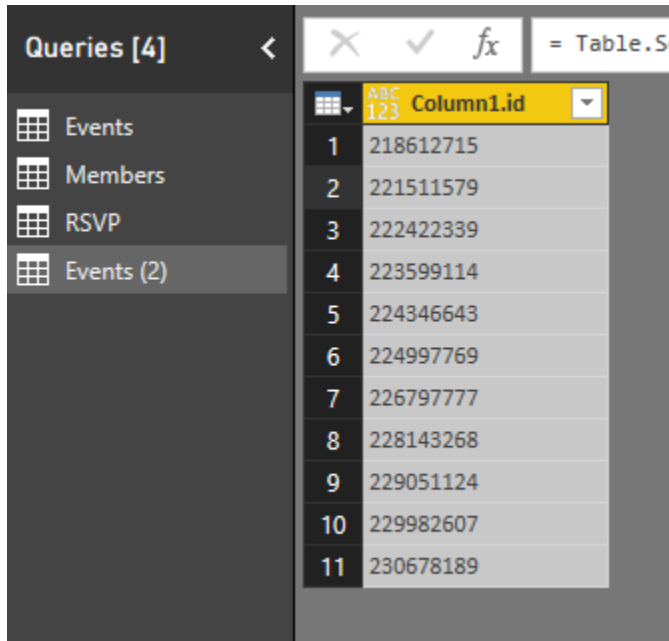
I start by fetching result of above URL into Power Query, and here is what I get after expanding the JSON list and records;

▼ ABC 123	Column1.mtime	▼ ABC 123	Column1.response	▼ ABC 123	Column1.tallies	▼ ABC 123	Column1.guests	▼ ABC 123	Column1.member.member...	▼ ABC 123	Column1.event
96	1.46372E+12	no	Record			0		5915204	Record		
74	1.46175E+12	yes	Record			0		66306872	Record		
40	1.46174E+12	yes	Record			0		200870054	Record		
21	1.46397E+12	yes	Record			0		195710156	Record		
53	1.46288E+12	yes	Record			0		204872467	Record		
74	1.4618E+12	yes	Record			0		160955302	Record		
80	1.46268E+12	yes	Record			0		204040342	Record		
49	1.46407E+12	yes	Record			0		119363682	Record		
90	1.46181E+12	yes	Record			0		149123682	Record		
94	1.46373E+12	yes	Record			0		155618572	Record		
39	1.46414E+12	no	Record			0		160813312	Record		
27	1.46173E+12	yes	Record			0		199892152	Record		

Note that I've also expanded Members column to get Member ID, this is what I would use later to join to Members query. I also have a column for the response which says if this member responded yes to the RSVP or not. I name this query as RSVP.

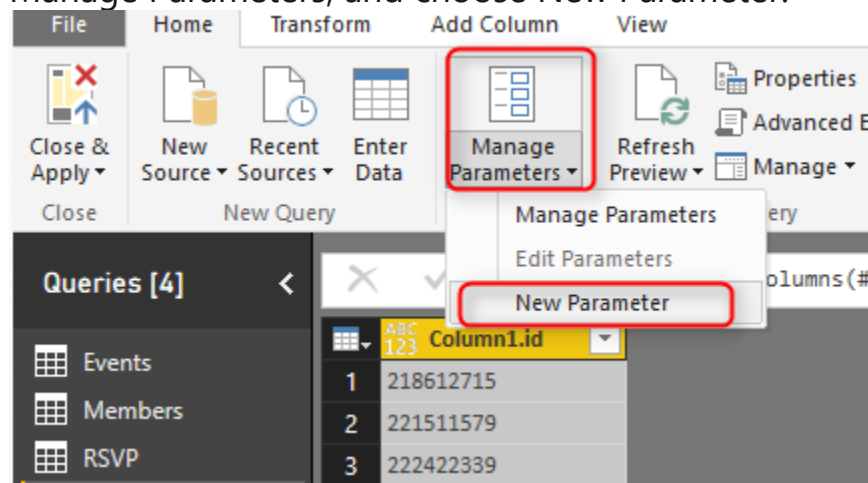
## Add a Parameter and Create Function

The query above is only for one event, but I am looking for data for every event. So I have to create a function for getting RSVP result and apply that on every row in Events query. I create a duplicate version of Events, and remove everything and keep the only column1.id (which is event id).



	Column1.id	
1	218612715	
2	221511579	
3	222422339	
4	223599114	
5	224346643	
6	224997769	
7	226797777	
8	228143268	
9	229051124	
10	229982607	
11	230678189	

Now I create a parameter to make my RSVP query parametric. Click on Manage Parameters, and choose New Parameter.



I create a parameter with name EventID and default value as one of my event IDs;

## Parameters

123 EventID
✕

New

Name  
EventID

Description

☒ Required

Type  

Any

Allowed Values  

Any value

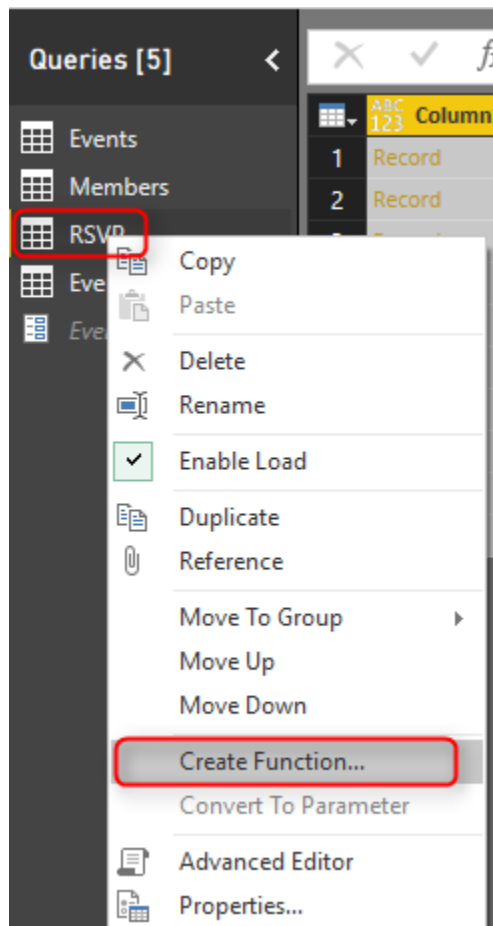
Current Value  

218612715

Now I go to advanced editor tab of my RSVP query, and change the script to use EventID variable as below;

```
let
    event=EventID,
    Source =
1  Json.Document(Web.Contents("https://api.meetup.com/2/rsvps?event_id=" & Text.From(event) & "&key=XYZ")),
2  results = Source[results],
3  #"Converted to Table" = Table.FromList(results, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
4  #"Expanded Column1" = Table.ExpandRecordColumn(#"Converted to Table", "Column1", {"venue", "created",
5  "member_photo", "answers", "rsvp_id", "mtime", "response", "tallies", "guests", "member", "event", "group"},
6  {"Column1.venue", "Column1.created", "Column1.member_photo", "Column1.answers", "Column1.rsvp_id",
7  "Column1.mtime", "Column1.response", "Column1.tallies", "Column1.guests", "Column1.member",
8  "Column1.event", "Column1.group"}),
9  #"Expanded Column1.member" = Table.ExpandRecordColumn(#"Expanded Column1", "Column1.member",
    {"member_id"}, {"Column1.member.member_id"})
in
    #"Expanded Column1.member"
```

Now that my query works with a parameter, I can right click on it, and Create Function.



I name the function as GetRSVP, and it gets EventID as an input parameter

## Create Function

Enter names for the new function and its parameters.

Function name

GetRSVP

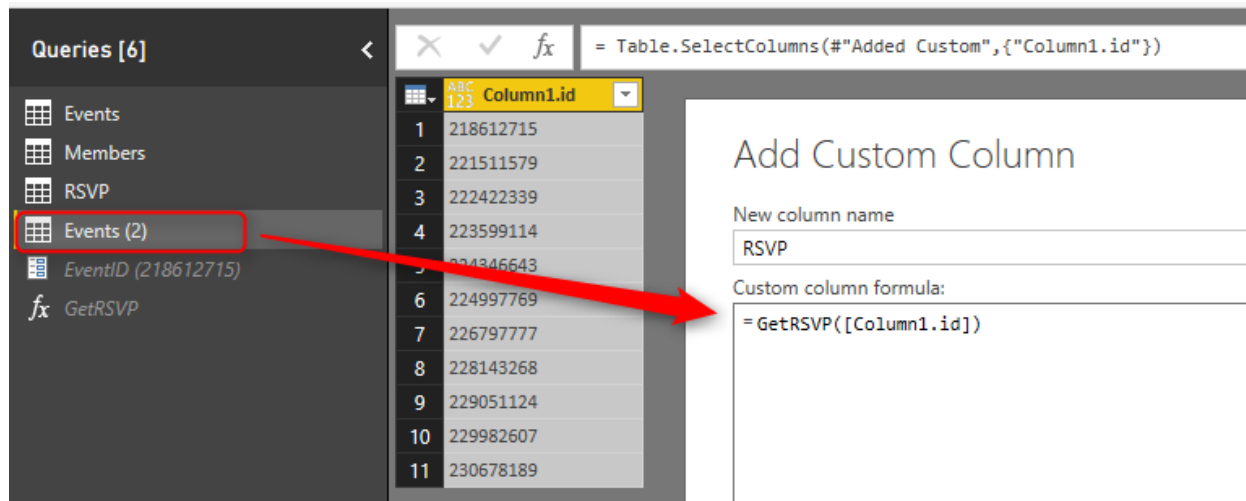
Parameter name (EventID)

EventID

OK

Cancel

Now I can go to my copy of Events query which named Events(2) and adds a custom function that calls GetRSVP as below



Queries [6]

- Events
- Members
- RSVP
- Events (2)
- EventID (218612715)
- GetRSVP

Column1.id

1	218612715
2	221511579
3	222422339
4	223599114
5	224346643
6	224997769
7	226797777
8	228143268
9	229051124
10	229982607
11	230678189

Add Custom Column

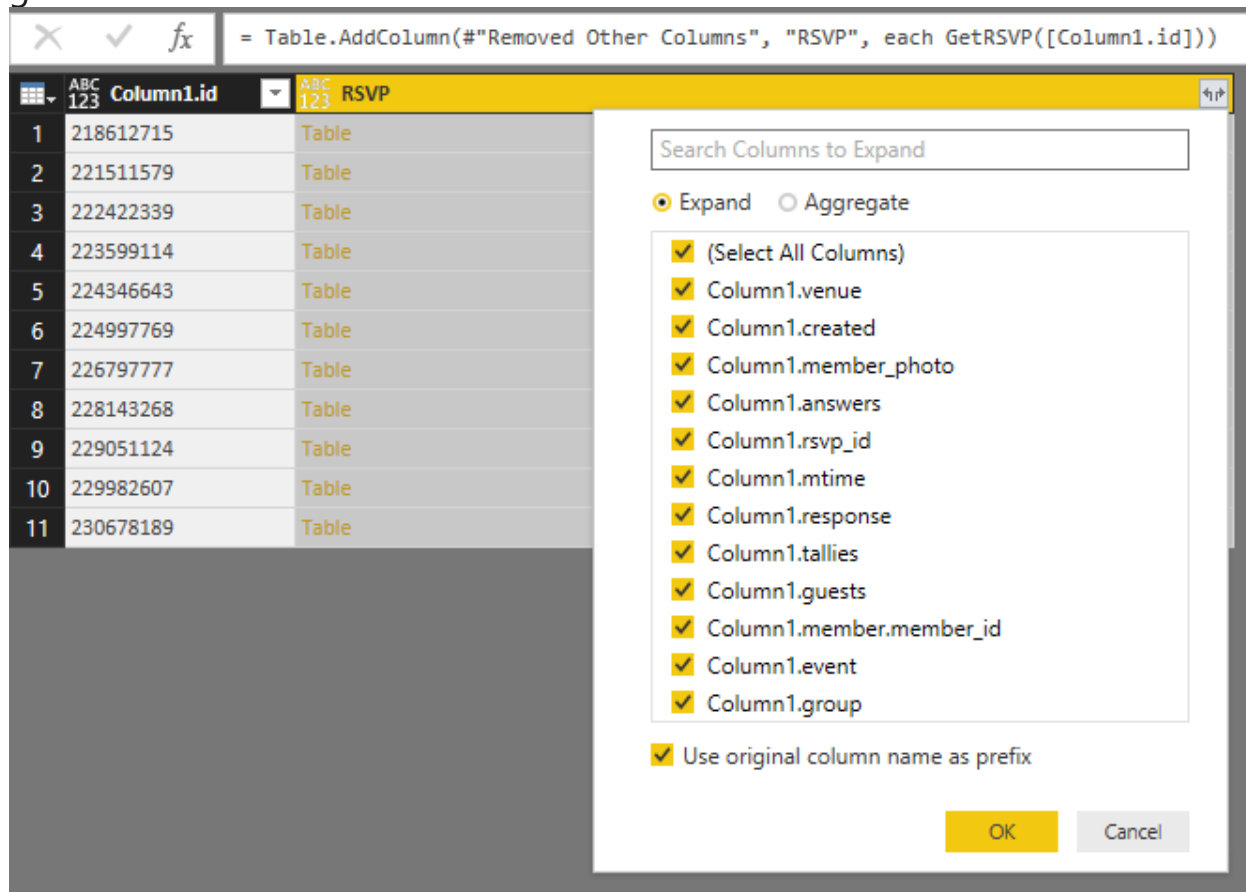
New column name

RSVP

Custom column formula:

=GetRSVP([Column1.id])

Result set would be a column with tables in each row, which I can expand to get RSVP columns there



Column1.id

1	218612715	Table
2	221511579	Table
3	222422339	Table
4	223599114	Table
5	224346643	Table
6	224997769	Table
7	226797777	Table
8	228143268	Table
9	229051124	Table
10	229982607	Table
11	230678189	Table

RSVP

Search Columns to Expand

☒ Expand ☐ Aggregate

- ☒ (Select All Columns)
- ☒ Column1.venue
- ☒ Column1.created
- ☒ Column1.member\_photo
- ☒ Column1.answers
- ☒ Column1.rsvp\_id
- ☒ Column1.mtime
- ☒ Column1.response
- ☒ Column1.tallies
- ☒ Column1.guests
- ☒ Column1.member.member\_id
- ☒ Column1.event
- ☒ Column1.group

☒ Use original column name as prefix

OK Cancel

Note that you may hit the privacy configuration warning and errors at this step; because this is just for test. I went to Files -> Options and Settings ->



Option, and change privacy setting to ignore. This is not best practice, you have to set it up appropriate for a production environment, but here is just a demo and test, so let's have fun.

## Options

### GLOBAL

Data Load  
DirectQuery  
R Scripting  
Security  
**Privacy**  
Updates  
Usage Data  
Diagnostics

### Privacy Levels

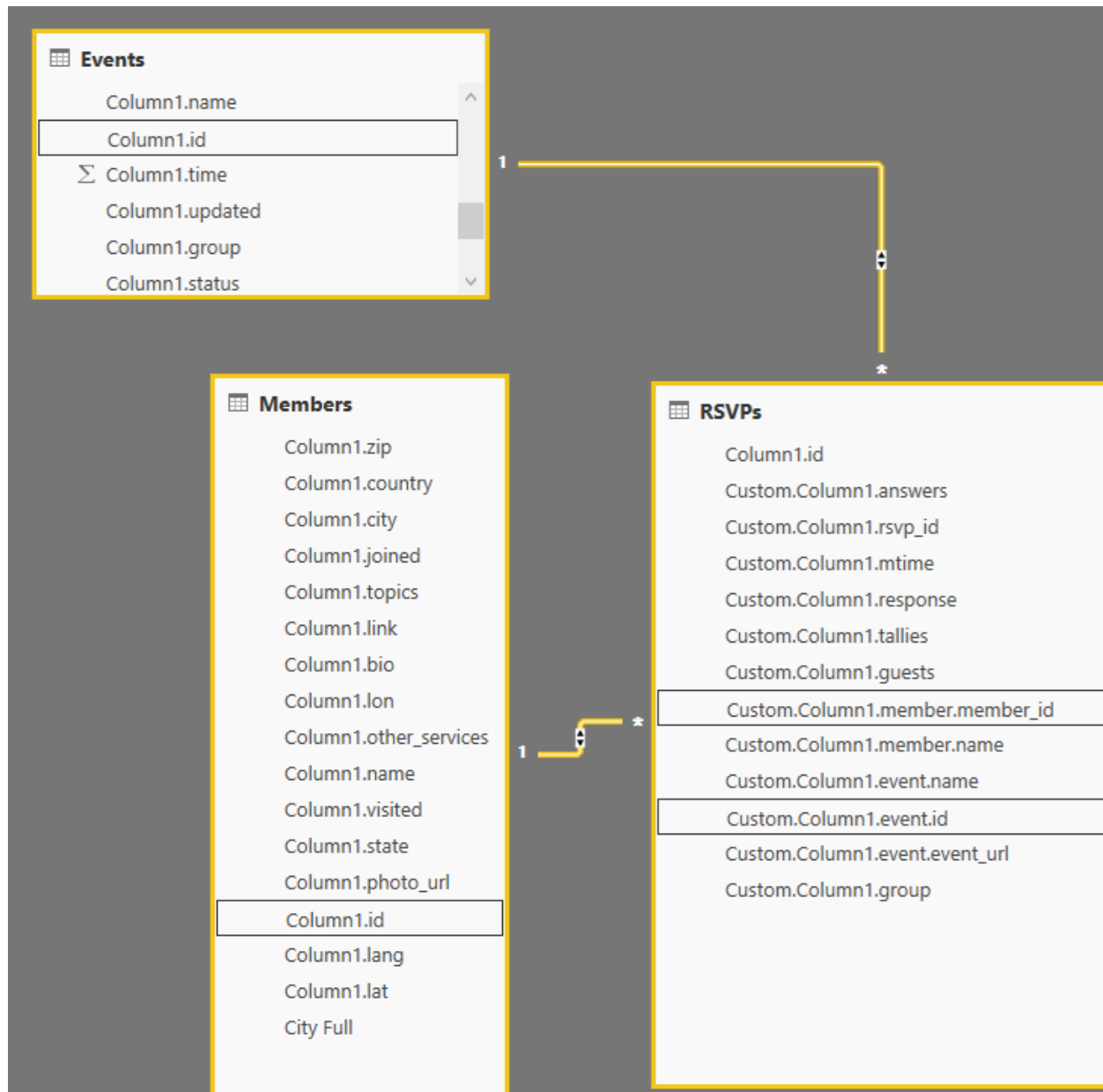
- ☐ Always combine data according to your Privacy Level settings for each source
- ☐ Combine data according to each file's Privacy Level settings
- ☒ Always ignore Privacy Level settings ⓘ

[Learn more about Privacy Levels](#)

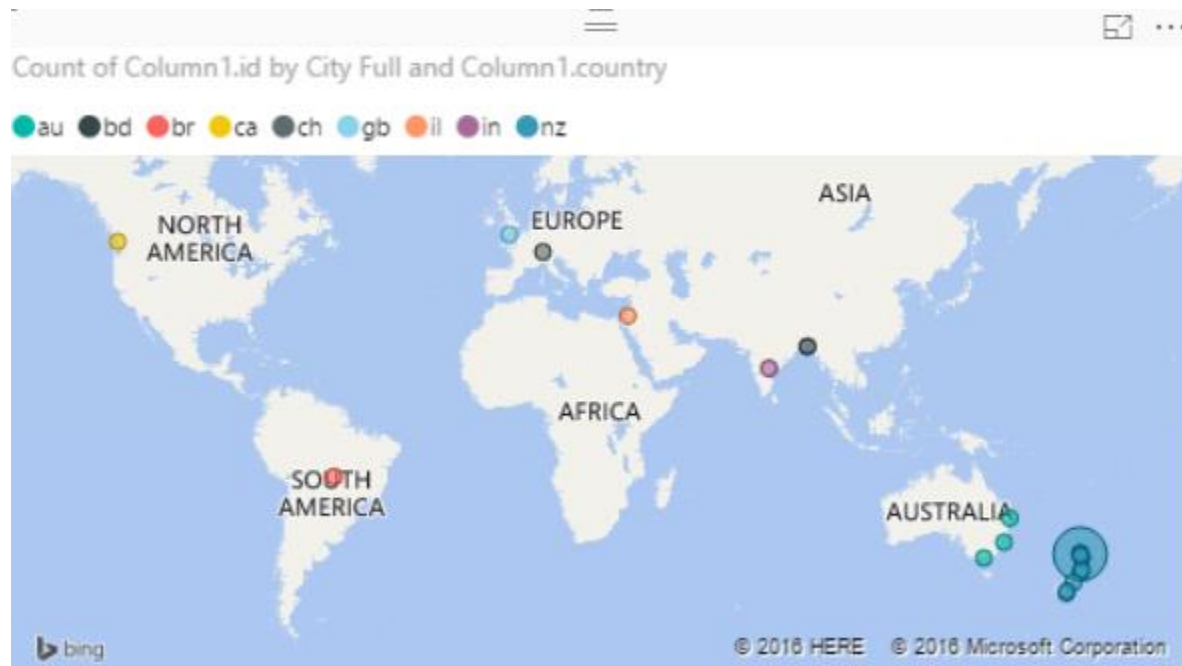
I rename this query as RSVPs and Close and Load data into Power BI. I'll create a relationship as below;

Events and RSVPs= column1.id from Events, Custom.Column1.event.id from RSVPs

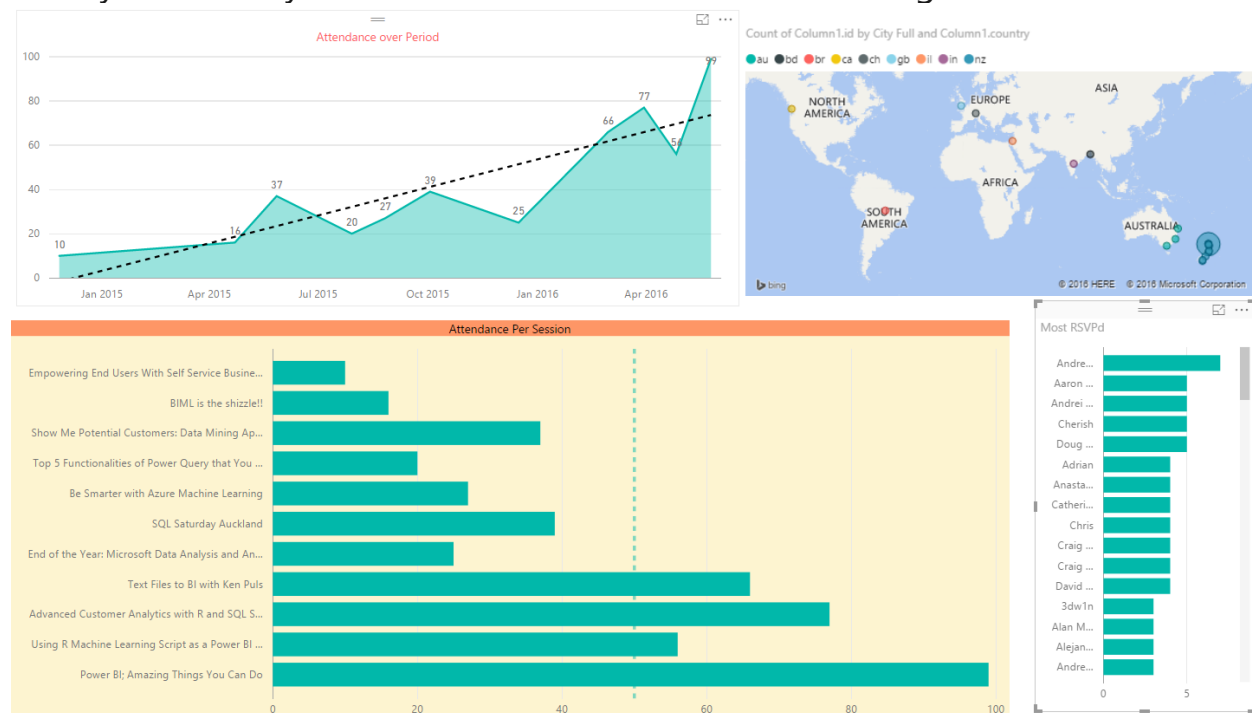
RSVPs and Members= Custom.Column1.member.member\_id from RSVPs, Column1.id from Members



I can now visualize members by their geolocation city information in a map



It is interesting that we have some members in our user group in other continents. However the majority are in Auckland, New Zealand as expected. I can also build another bar chart for members who RSVPed yes mostly. Here is my final visualization with some formatting:



I can still dig into more details with adding more data into my Power BI model and visualization, but I leave it to you now from here to see how far you can go with it. Have fun with Power BI and Meetup API.

## Part III: Transformations

# Reference vs Duplicate in Power BI; Power Query Back to Basics

Posted by [Reza Rad](#) on Sep 25, 2018



When you work with tables and queries in Power Query and Power BI, you get the option to copy them through these actions: Duplicate, or Reference. It has always been a question in my sessions and courses that what is the actual difference between these two actions. The explanation is simple but very important to understand. Because when you know the difference, you will use it properly. In this short blog post, I'll explain what the Reference, and the difference of that with Duplicate is. To learn more about Power BI; read [Power BI from Rookie to Rock Star book](#).

## Duplicate

If you are looking to copy an entire query with all of its steps, then Duplicate is your friend. Let's see this in action. As an example, Let's assume that we got the data from a web page that shows us the best seller's movies information. If you have done [the movies example of my book](#) previously, the website is BoxOfficeMojo. Here is the link to the page:

<https://www.boxofficemojo.com/alltime/world/?pagenum=1&p=.htm>

In Power Query, we got data From Web, and selected this source;

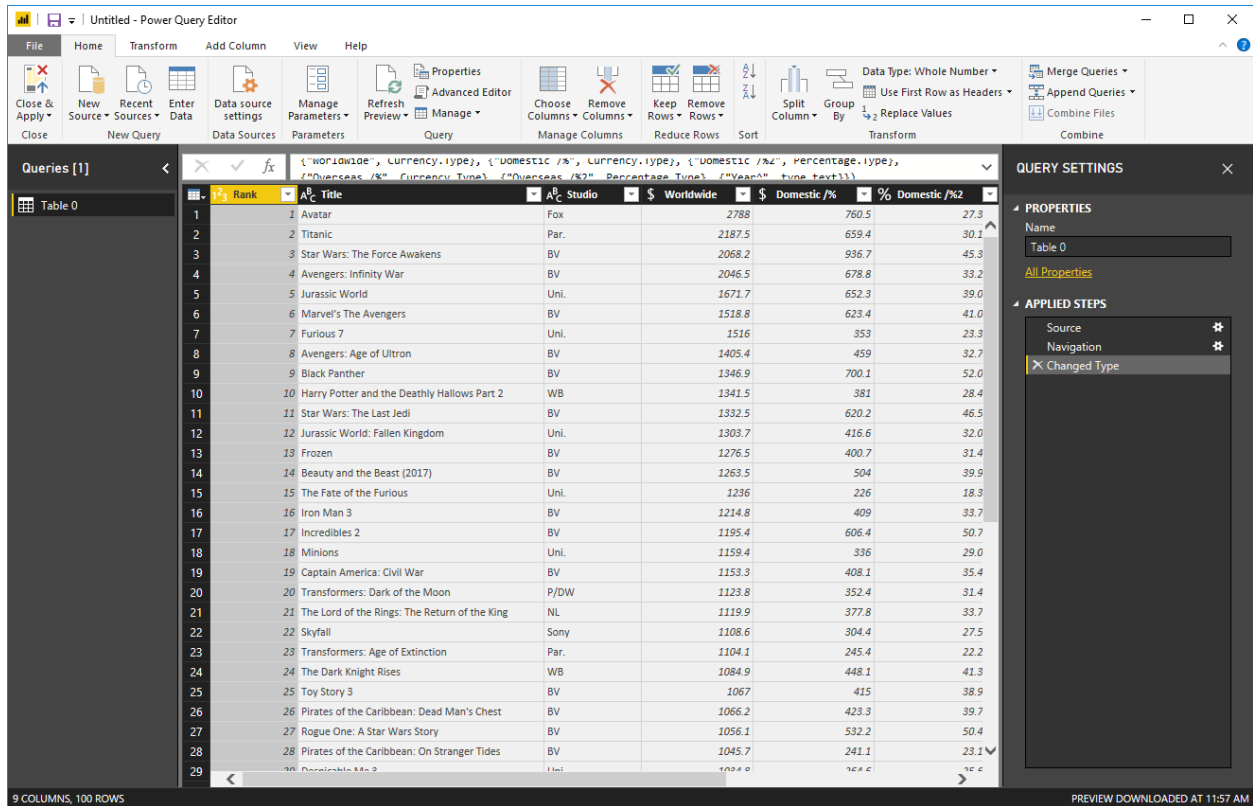


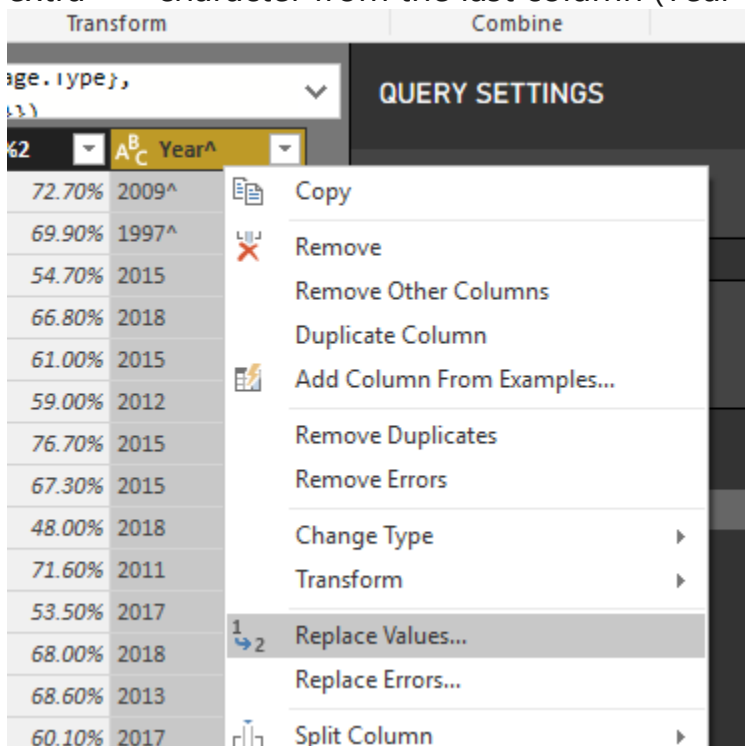
Table 0

Rank	Title	Studio	Worldwide	Domestic /%	% Domestic /%2
1	Avatar	Fox	2788	760.5	27.3
2	Titanic	Par.	2187.5	659.4	30.1
3	Star Wars: The Force Awakens	BV	2068.2	936.7	45.3
4	Avengers: Infinity War	BV	2046.5	678.8	33.2
5	Jurassic World	Uni.	1671.7	652.3	39.0
6	Marvel's The Avengers	BV	1518.8	623.4	41.0
7	Furious 7	Uni.	1516	353	23.3
8	Avengers: Age of Ultron	BV	1405.4	459	32.7
9	Black Panther	BV	1346.9	700.1	52.0
10	Harry Potter and the Deathly Hallows Part 2	WB	1341.5	381	28.4
11	Star Wars: The Last Jedi	BV	1332.5	620.2	46.5
12	Jurassic World: Fallen Kingdom	Uni.	1303.7	416.6	32.0
13	Frozen	BV	1276.5	400.7	31.4
14	Beauty and the Beast (2017)	BV	1263.5	504	39.9
15	The Fate of the Furious	Uni.	1236	226	18.3
16	Iron Man 3	BV	1214.8	409	33.7
17	Incredibles 2	BV	1195.4	606.4	50.7
18	Minions	Uni.	1159.4	336	29.0
19	Captain America: Civil War	BV	1153.3	408.1	35.4
20	Transformers: Dark of the Moon	P/DW	1123.8	352.4	31.4
21	The Lord of the Rings: The Return of the King	NL	1119.9	377.8	33.7
22	Skyfall	Sony	1108.6	304.4	27.5
23	Transformers: Age of Extinction	Par.	1104.1	245.4	22.2
24	The Dark Knight Rises	WB	1084.9	448.1	41.3
25	Toy Story 3	BV	1067	415	38.9
26	Pirates of the Caribbean: Dead Man's Chest	BV	1066.2	423.3	39.7
27	Rogue One: A Star Wars Story	BV	1056.1	532.2	50.4
28	Pirates of the Caribbean: On Stranger Tides	BV	1045.7	241.1	23.1
29	Despicable Me 2	Uni.	1034.0	764.2	73.4

9 COLUMNS, 100 ROWS

PREVIEW DOWNLOADED AT 11:57 AM

Let's say for this table; we do some transformations. For example; removing extra "^" character from the last column (Year column);



Transform

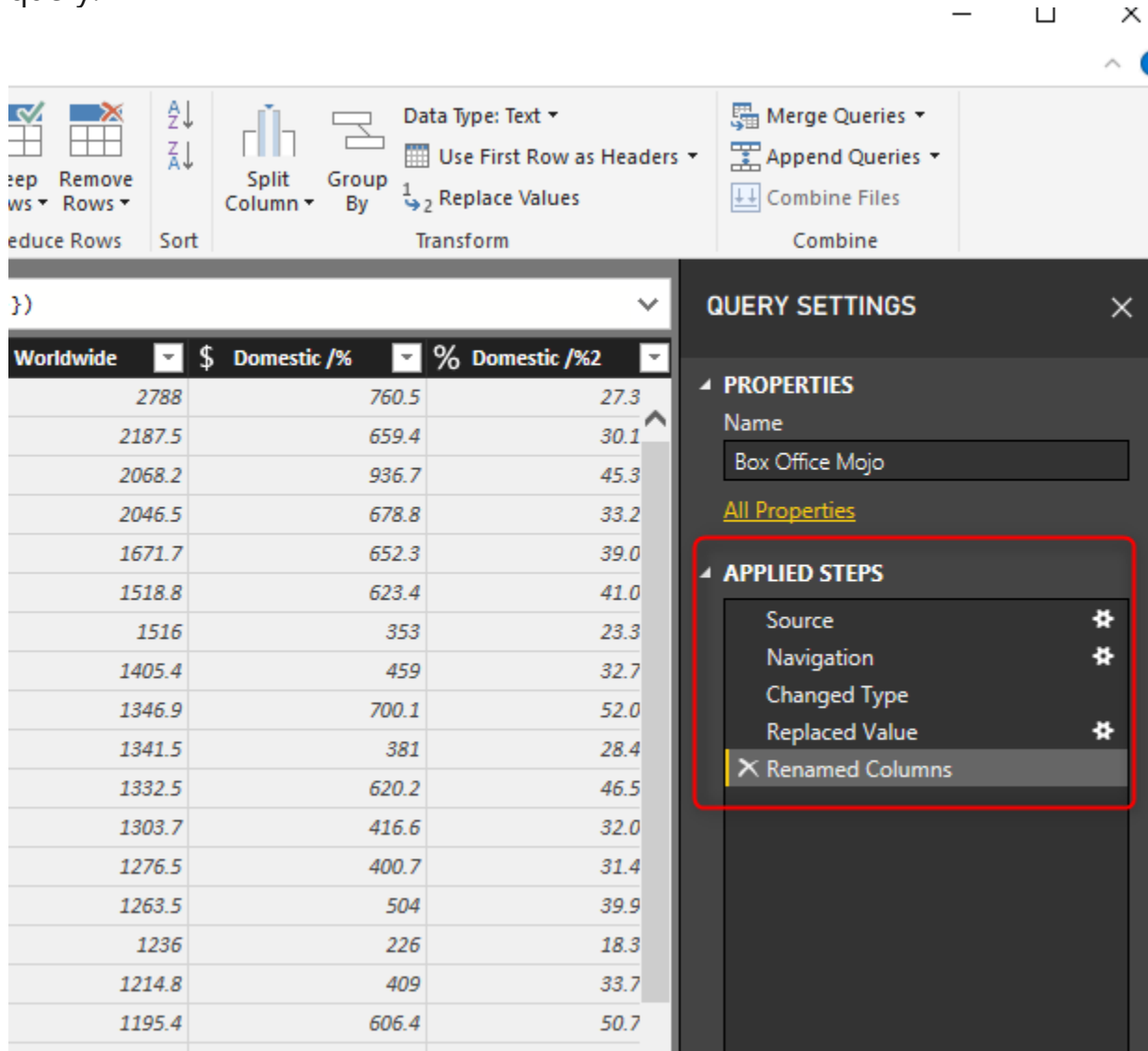
Combine

QUERY SETTINGS

Year^

- Copy
- Remove
- Remove Other Columns
- Duplicate Column
- Add Column From Examples...
- Remove Duplicates
- Remove Errors
- Change Type
- Transform
- Replace Values...
- Replace Errors...
- Split Column

and then some other transformations, so we end up with some steps for this query.



The screenshot displays the Power BI Power Query editor. The main area shows a table with the following data:

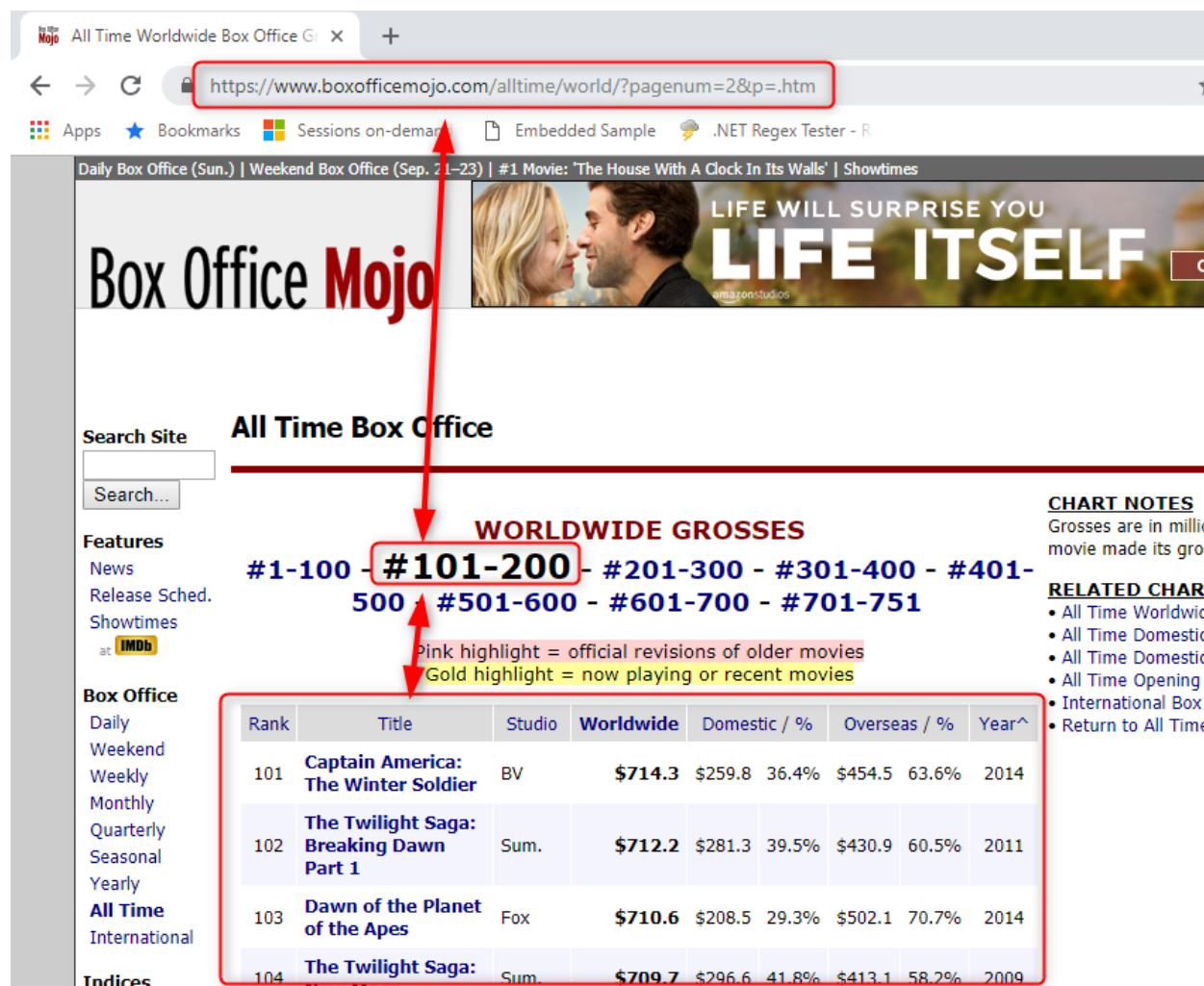
Worldwide	\$ Domestic /%	% Domestic /%2
2788	760.5	27.3
2187.5	659.4	30.1
2068.2	936.7	45.3
2046.5	678.8	33.2
1671.7	652.3	39.0
1518.8	623.4	41.0
1516	353	23.3
1405.4	459	32.7
1346.9	700.1	52.0
1341.5	381	28.4
1332.5	620.2	46.5
1303.7	416.6	32.0
1276.5	400.7	31.4
1263.5	504	39.9
1236	226	18.3
1214.8	409	33.7
1195.4	606.4	50.7

The right-hand pane shows the 'QUERY SETTINGS' window. The 'APPLIED STEPS' section is highlighted with a red box and contains the following steps:

- Source
- Navigation
- Changed Type
- Replaced Value
- Renamed Columns

After doing all these transformations, you realize that this data is only for the first hundred best seller movies because that web page doesn't have the remaining movies. To get the remaining, you need to navigate to page 2, which has a different URL, but the same data structure.





Box Office Mojo

All Time Box Office

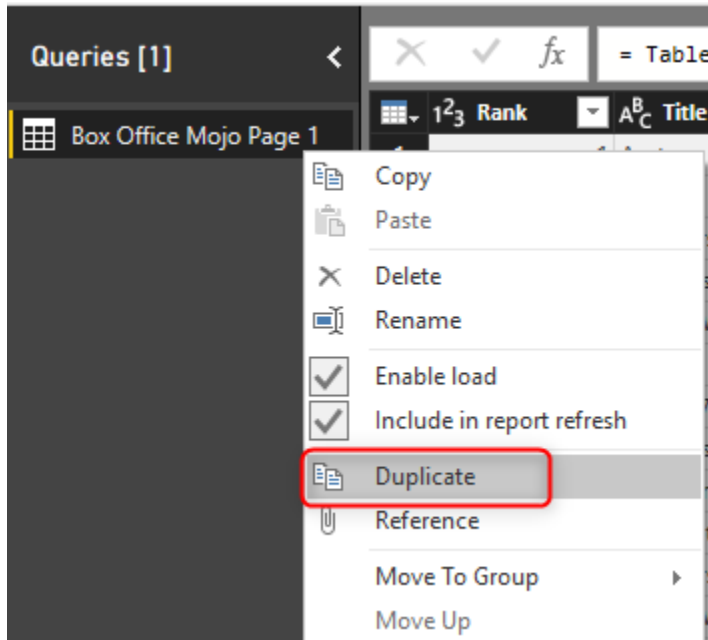
WORLDWIDE GROSSES

#1-100 - **#101-200** - #201-300 - #301-400 - #401-500 - #501-600 - #601-700 - #701-751

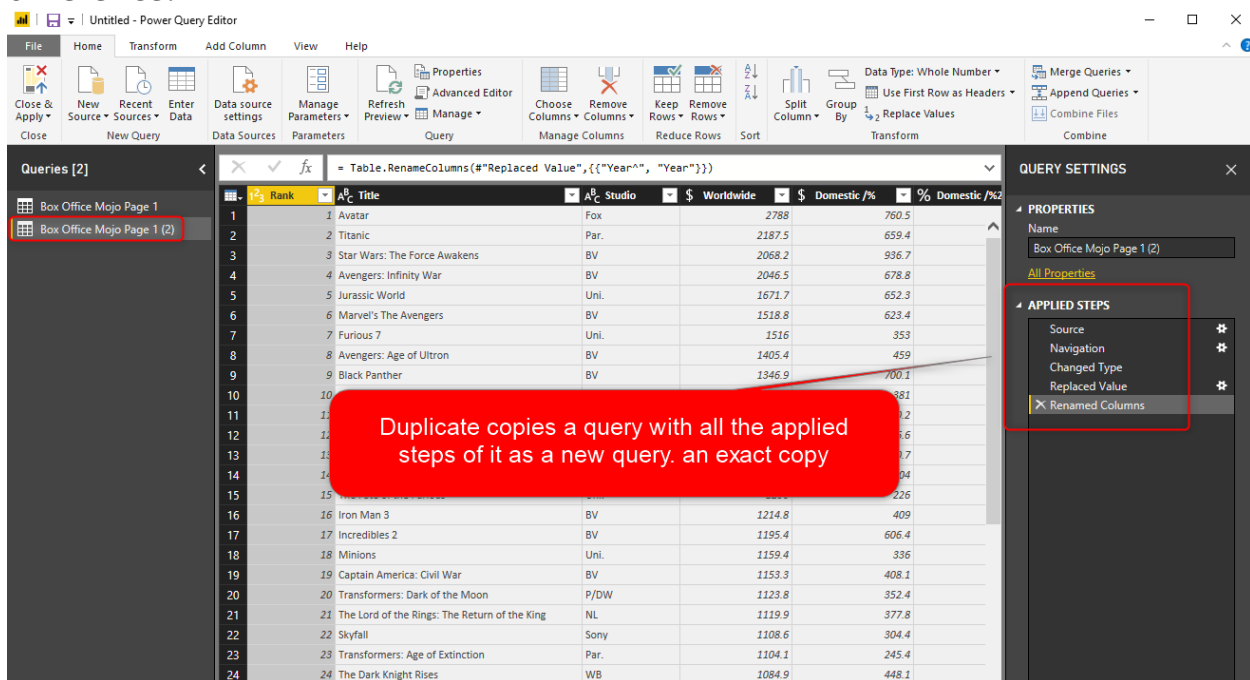
Pink highlight = official revisions of older movies  
Gold highlight = now playing or recent movies

Rank	Title	Studio	Worldwide	Domestic / %	Overseas / %	Year^
101	Captain America: The Winter Soldier	BV	\$714.3	\$259.8 36.4%	\$454.5 63.6%	2014
102	The Twilight Saga: Breaking Dawn Part 1	Sum.	\$712.2	\$281.3 39.5%	\$430.9 60.5%	2011
103	Dawn of the Planet of the Apes	Fox	\$710.6	\$208.5 29.3%	\$502.1 70.7%	2014
104	The Twilight Saga: New Moon	Sum.	\$709.7	\$296.6 41.8%	\$413.1 58.2%	2009

Well, what you need to do? You have to do all those steps on page 2 as well. Let's keep this example static and basic, (Because in complex scenarios when you have many pages, you may use functions and parameters to loop through all pages and combine them all. If you are interested in learning about that, [read my post here](#)). Let's say you want to do all those steps that you have done for page one, now for page two. To do that; you can leverage Duplicate. Create a duplicate of Box Office Mojo (I called it; Box Office Mojo Page 1)

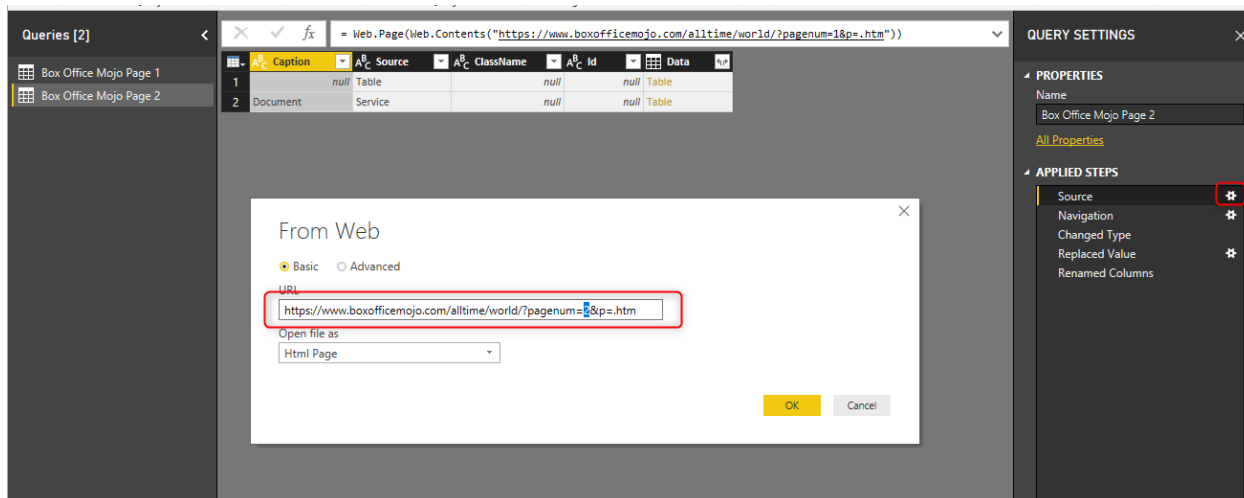


When you create the Duplicate query, it will be an exact copy of the first query, with all steps of it. These two queries are exactly like each other. No difference!



*Duplicate copies a query with all the applied steps of it as a new query; an exact copy.*

After creating the copy, then you can go to the source step to change the URL:



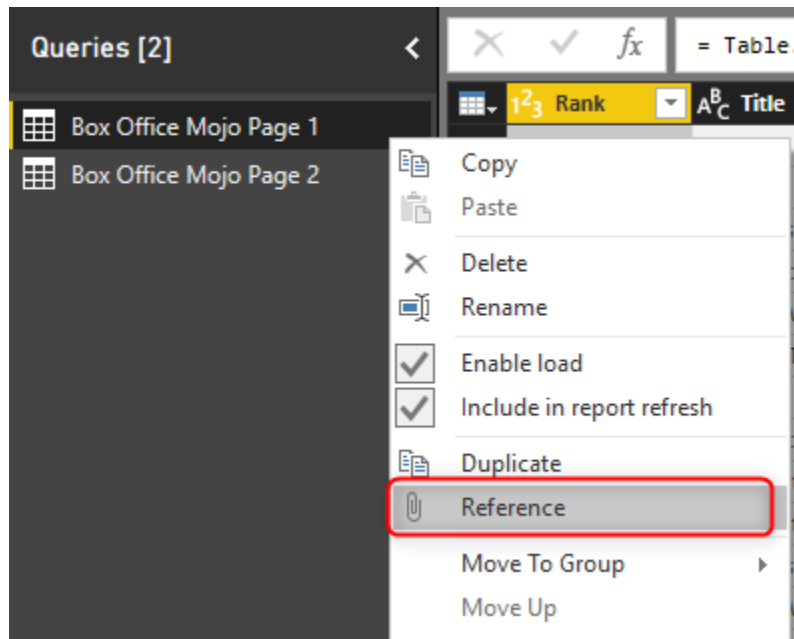
Using Duplicate, you managed to copy a query with all steps in it, and then make changes in your new query. Your original query is intact.

*Duplicate is the option to choose when you want to copy a query, but do a different configuration in steps.*

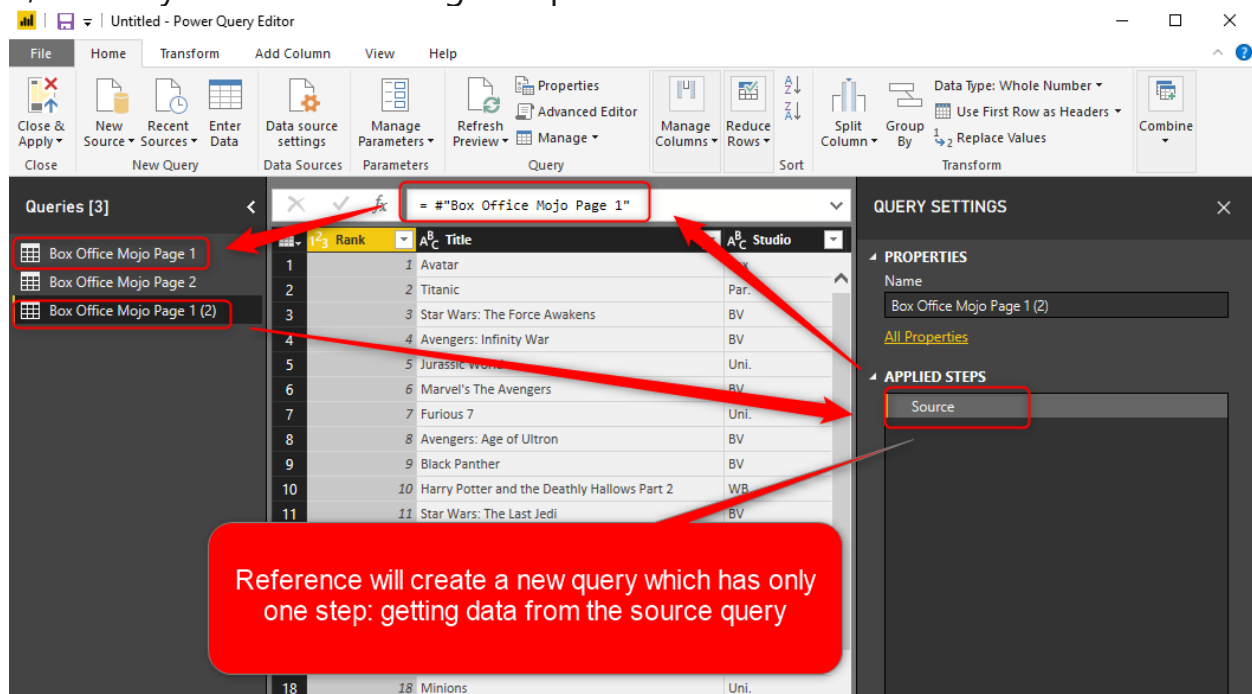
## Reference

Reference is another way of copying a query. However, the big difference is that; When you reference a query, the new query will have only one step: sourcing from the original query. A referenced query, will not have the applied steps of the original query. Let's see this option in action. Continuing the example above; let's say we want to create a new query that is the result of combining the page 1 and page 2 result. However, we do NOT want to change any of the existing queries, because we want to use those as the source for other operations.

With a right click on Box Office Mojo Page 1, I can create a Reference.



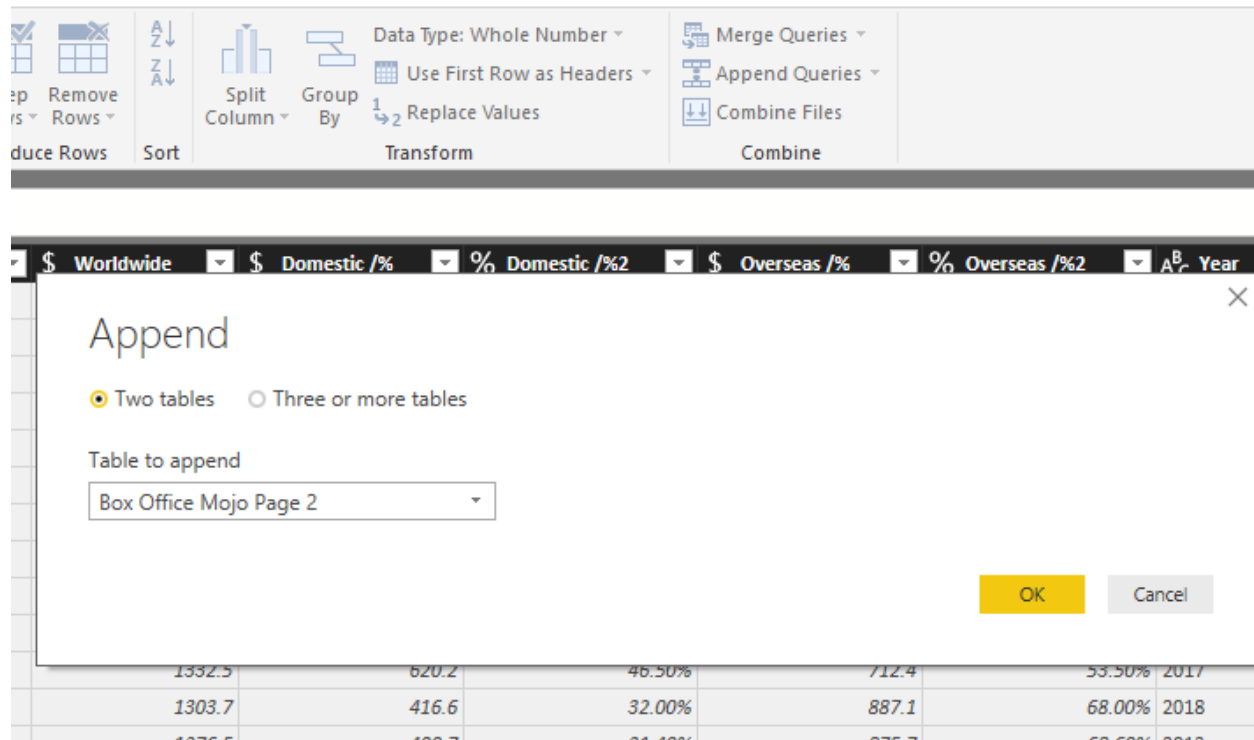
Reference will create a new query which is a copy of the Box Office Mojo Page 1, but only contains one single step:



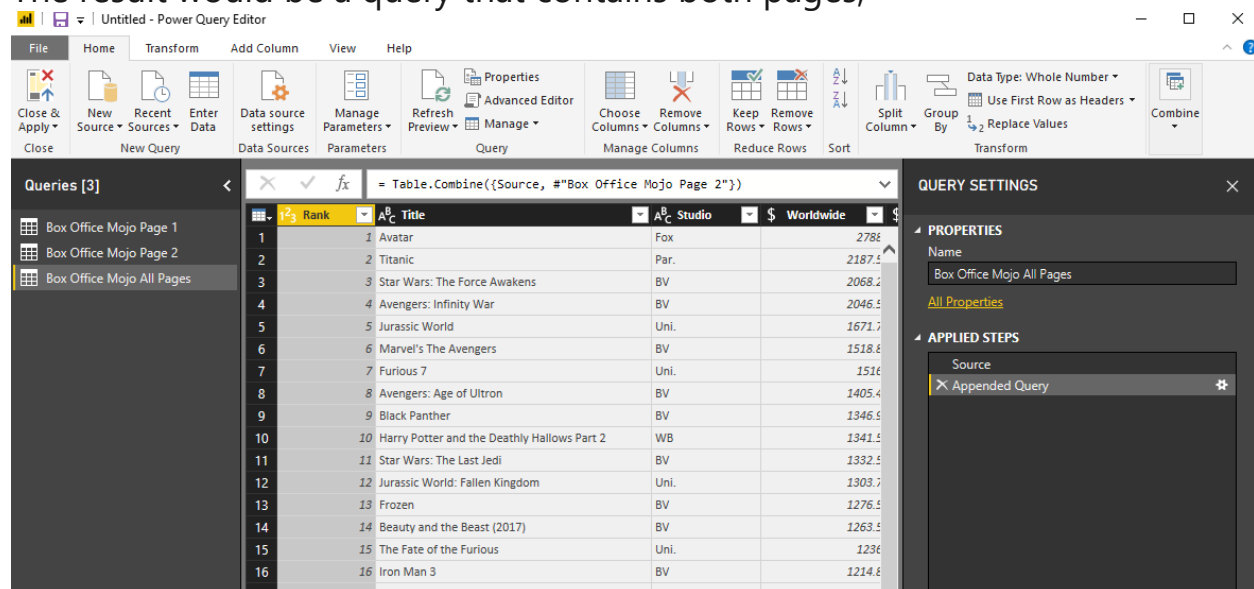
The only step in the new query is sourcing the data from the original query. What does it mean? It means if you make changes in the original query, then this new query will be impacted.

*Reference will create a new query which has only one step: Getting data from the original query.*

Now we can use this query, to append to the Box Office Mojo Page 2;



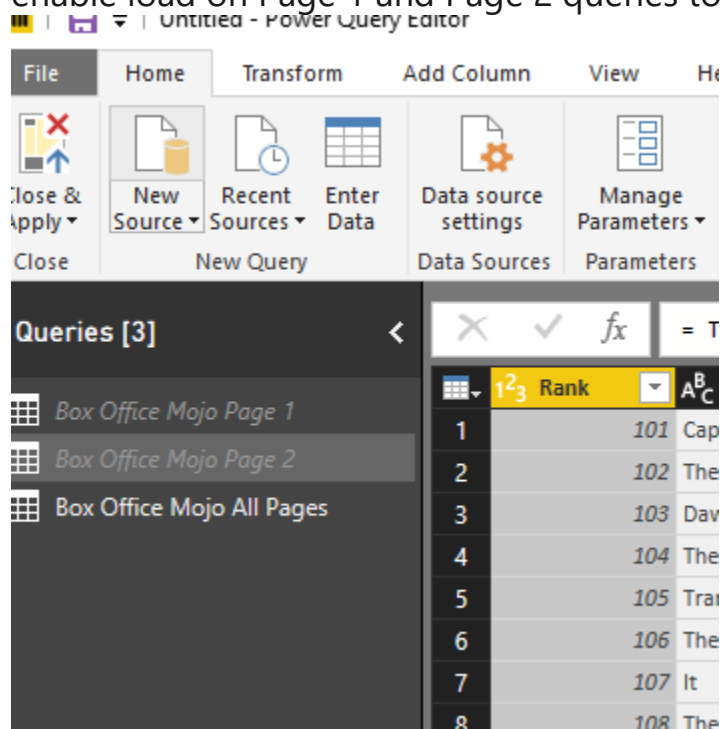
The result would be a query that contains both pages;



To learn more about append and the difference of that with Merge, [read my blog post here](#). In this example; we used Reference option to create a copy of the original query, and then continue some extra steps. There are many other usages for the Reference.

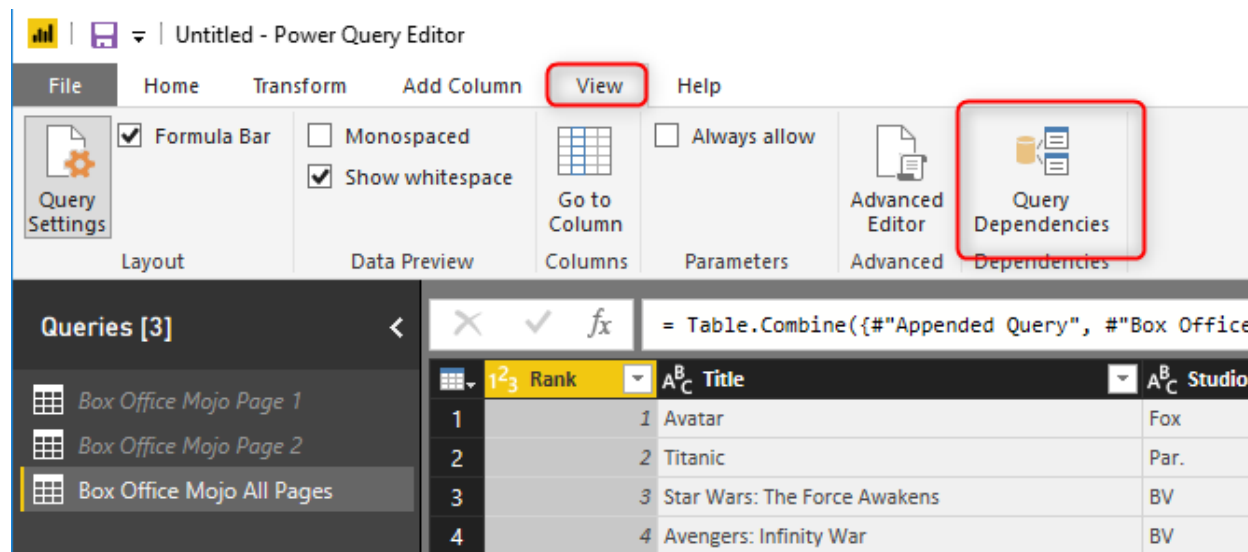
*Reference is a good choice when you want to branch a query into different paths. One path that follows some steps, and another that follows different steps, and both are sharing some steps in the original query.*

After doing the append in this example, it is a good idea to uncheck the enable load on Page 1 and Page 2 queries to [save some memory in Power BI](#).



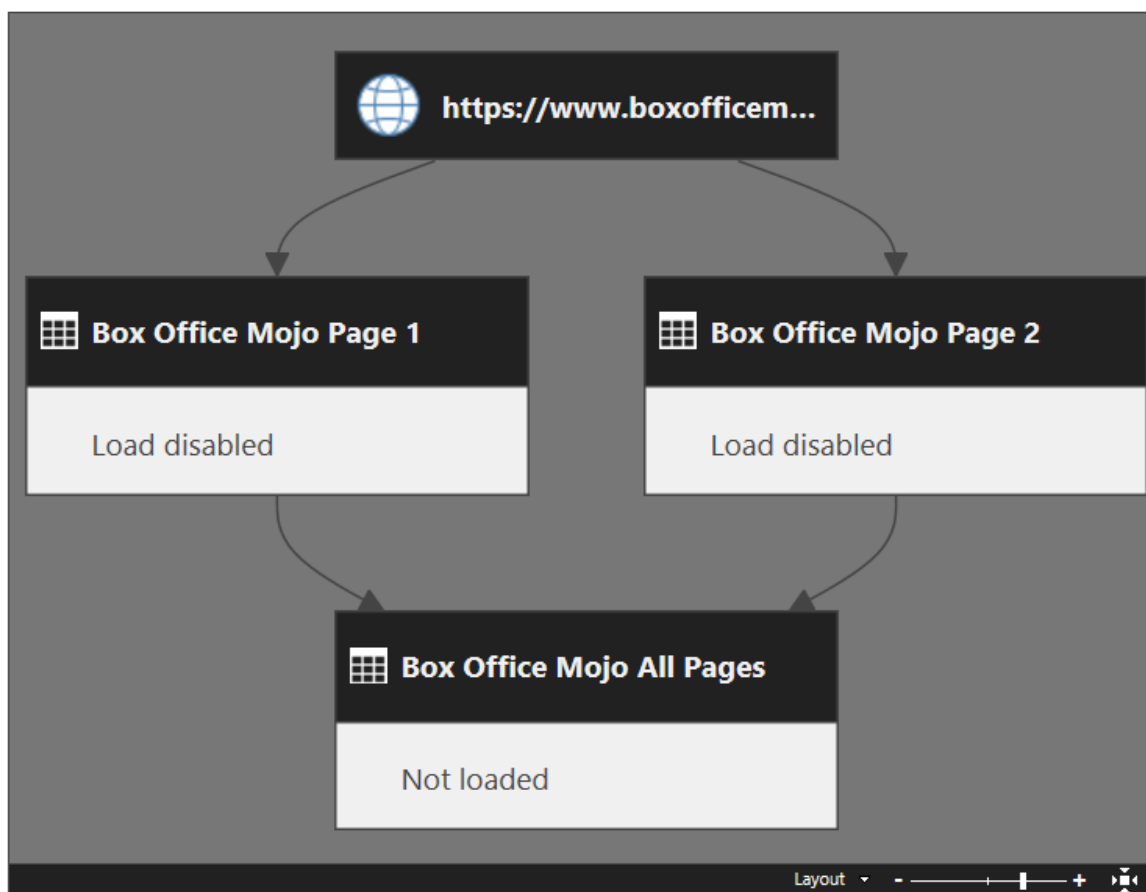
## Query Dependency

Finding out that which query is dependent (or referenced from) which query can be a bit challenging when you have too many queries. That is why we have the Query Dependency menu option in the View tab of Power Query;



For our example above, this is the query dependency diagram;

Query Dependencies



## **Duplicate vs. Reference**

Now that you know there are two options when you copy a query let's have a closer look at their difference.

### **Isolation from the Original or Dependency to the Original**

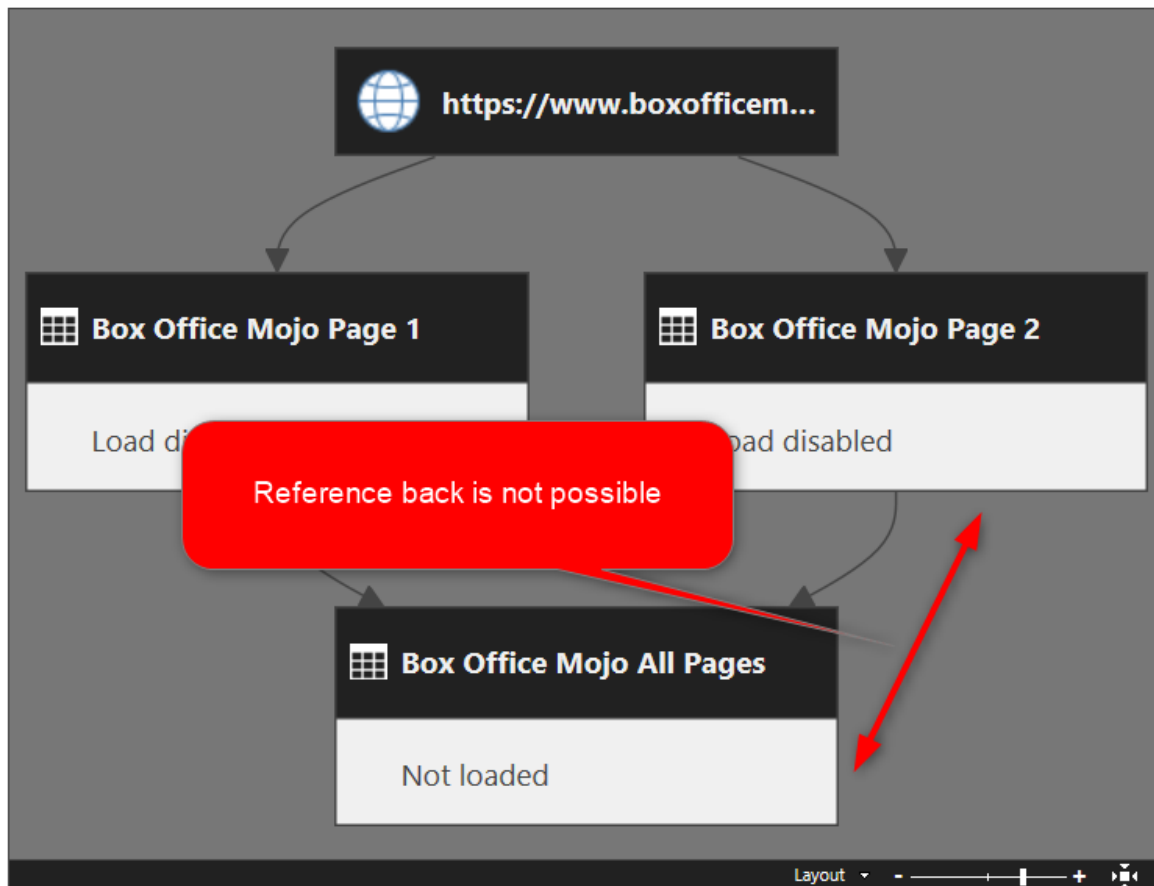
Duplicate creates a new copy with all the existing steps. The new copy will be isolated from the original query. You can make changes in the original or the new query, and they will NOT affect each other. Reference, on the other hand, is a new copy with only one single step: getting data from the original query. If you make a change in the original query, the new query will be impacted. For example; If you remove a column from the original query, the new query will not have it if it used the Reference method for copying.

### **Limitation of the Reference**

You can not use referenced queries in all situations. As an example; If you have a Query 1, and then you created a reference from that as Query 2. You cannot use the result from Query 2 in Query 1! It will create a circular reference. You are combining a query with reference to the query itself, It is impossible!



## Query Dependencies



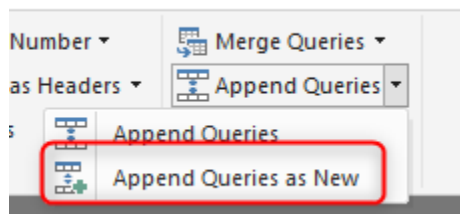
Close

## Some actions that invoke Reference or Duplicate

There are some actions in the Power Query that trigger Reference or Duplicate, let's check those options:

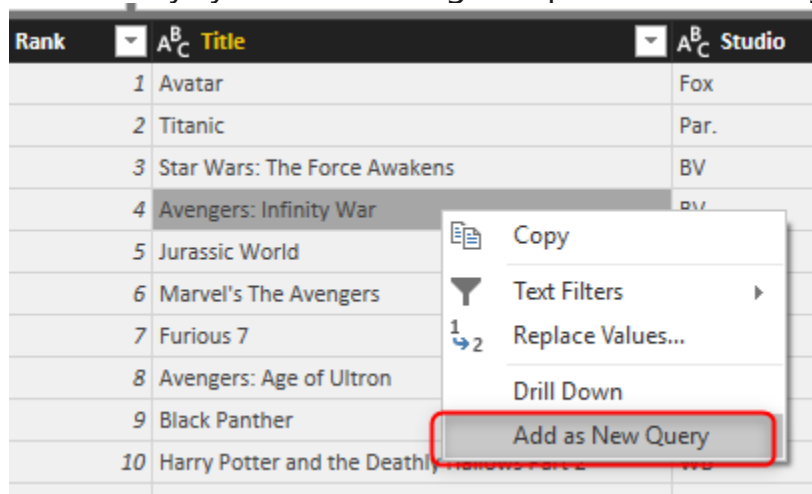
**Append Queries as New / or Merge Queries as New is a Reference action**

These two actions are creating a reference from the original query, and then they do Append or Merge with other queries.

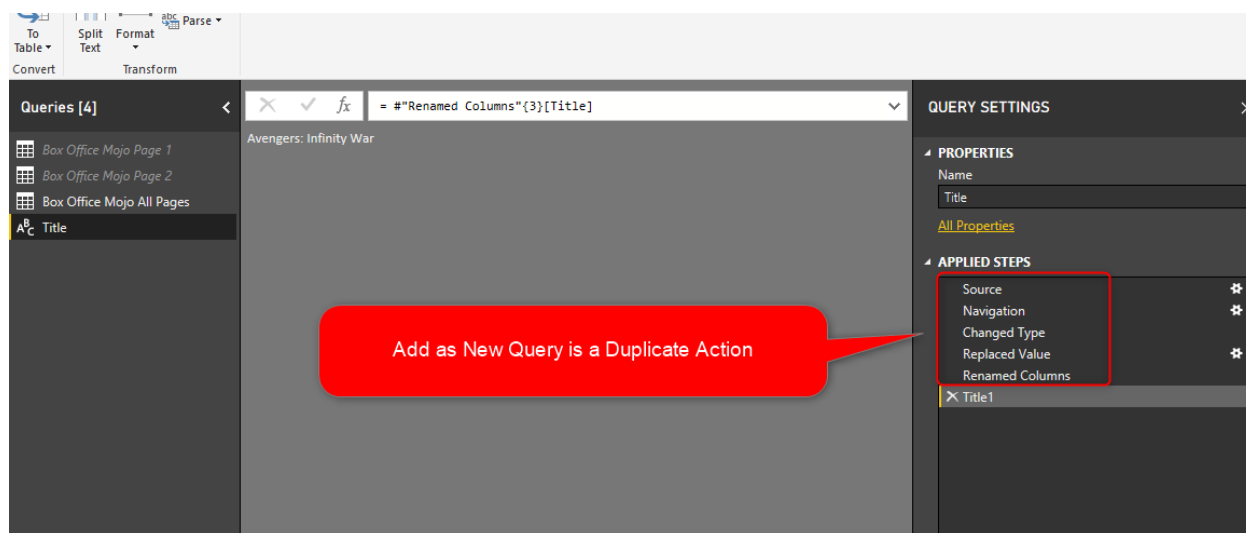


### Add as New Query is a Duplicate action

Believe it or not, when you right click on a column or cell and select “Add as New Query” you are creating a duplicate of the original query.

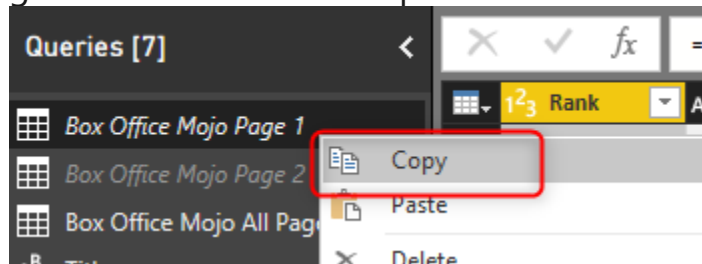


This can be misleading sometimes, because you may expect the new query to source from the original, and with the change of original, this query also to change. However, the truth is that this is a duplicate action, and after this action, your original query and the new copy will be isolated from each other.

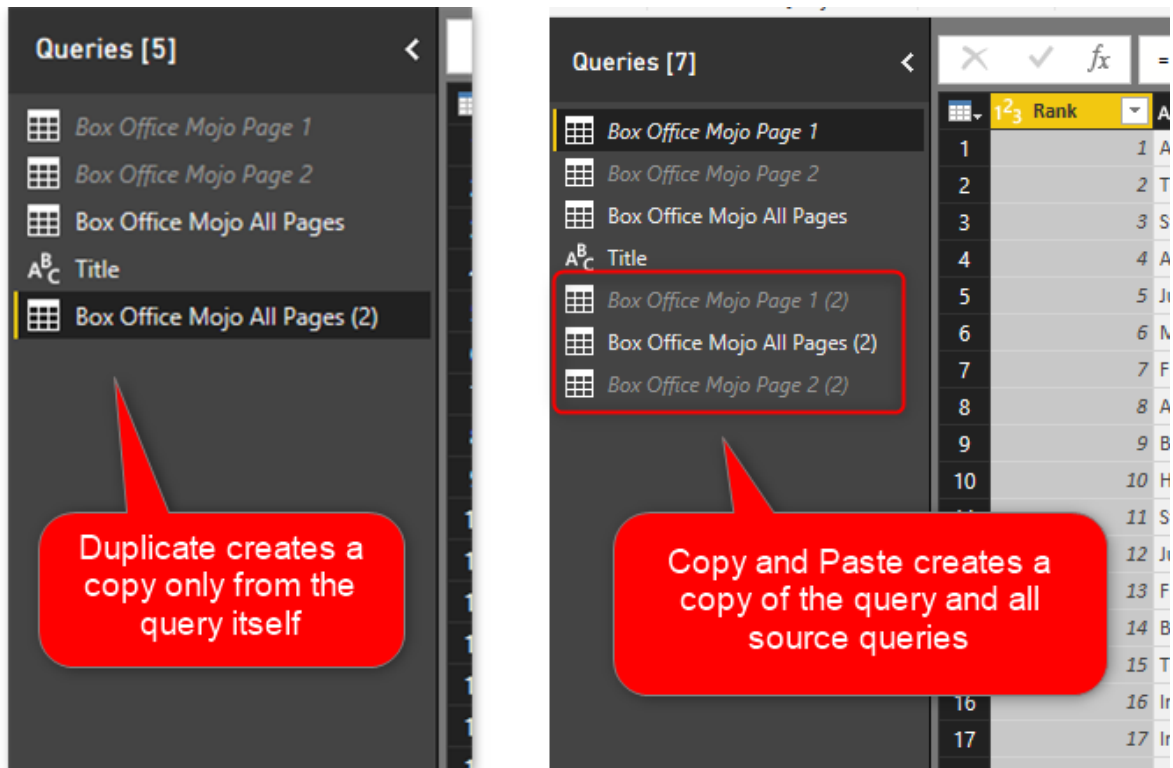


### Copy and Paste is Neither Duplicate Nor Reference!

This is another misconception that Copy and Paste are similar to Duplicate. It is not, and it is not Reference either. When you do this action on a simple query (I mean a query that is not sourced from any other queries), then you get a result similar to Duplicate.



But when you do the Copy and Paste on a query that is sourced from other queries; the result is a copy of all original queries. Here is the result of Copy and Paste on Box Office Mojo All Pages (which is sourced from Page 1 and Page 2);



## Summary

Duplicate and Reference are two different actions, and they are also different from Copy and Paste of a query. Duplicate will give you an exact copy of the query with all steps, and Reference will create a reference to the original query instead as a new query. Duplicate is a good option to choose when you want the two copies to be isolated from each other; Reference is a good option when you create different branches from one original query. There are some actions in Power Query that trigger Duplicate or Reference as listed in this blog post. Hope this was a good post for you to understand the difference between these two actions clearly, and use them wisely from now on.

# Append vs. Merge in Power BI and Power Query

Posted by [Reza Rad](#) on Jan 5, 2017



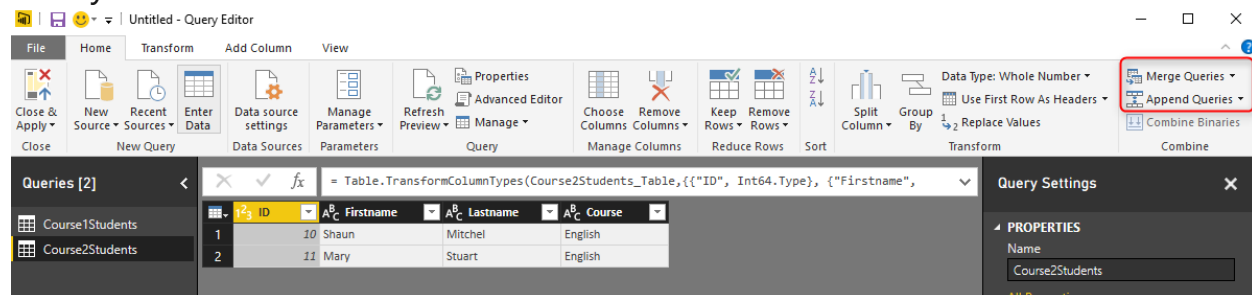
Combining two queries in Power Query or Power BI is one of the most basic and also essential tasks that you would need to do in most data preparation scenarios. There are two types of combining queries; Merge, and Append. Database developers easily understand the difference, but the majority of Power BI users are not developers. In this post, I'll explain the difference between Merge and Append, and situations that you should use each. If you want to learn more about Power BI, read [Power BI online book, from Rookie to Rock Star](#).

## Why Combine Queries?

This might be the first question comes into your mind; Why should I combine queries? The answer is that; You can do most of the things you want in a single query. However, it will be very complicated with hundreds of steps very quickly. On the other hand, your queries might be used in different places. For example one of them might be used as a table in Power BI model, and also playing the part of data preparation for another query. Combining queries is a

big help in writing better and simpler queries. I'll show you some examples of combining queries.

The result of a combine operation on one or more queries will be only one query. You can find Append or Merge in the Combine Queries section of the Query Editor in Power BI or Excel.



## Append

Append means results of two (or more) queries (which are tables themselves) will be combined into one query in this way:

- Rows will be appended after each other. (for example, appending a query with 50 rows with another query with 100 rows, will return a result set of 150 rows)
- Columns will be the same number of columns for each query\*. (for example, col1, col2,..., col10 in the first query, after appending with same columns in the second query will result into one query with a single set of col1,col2, ..., col10)

There is an exception for the number of columns which I'll talk about it later.

Let's first look at what Append looks like in action;

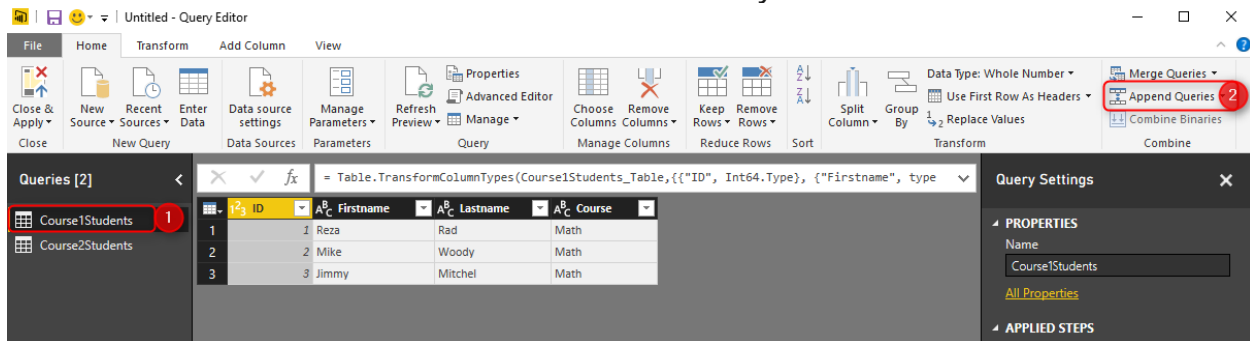
Consider two sample data sets; one for students of each course, Students of course 1:

ID	Firstname	Lastname	Course
1	Reza	Rad	Math
2	Mike	Woody	Math
3	Jimmy	Mitchel	Math

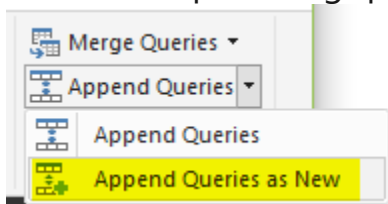
and Students of course 2:

ID	Firstname	Lastname	Course
10	Shaun	Mitchel	English
11	Mary	Stuart	English

To append these queries, Click on one of them and select Append Queries from the Combine section of Home tab in Query Editor



If you want to keep the existing query result as it is and create a new query with the appended result choose Append Queries as New, otherwise select Append Queries. In this example, I'll do Append Queries as New, because I want to keep existing queries intact.



You can choose what is the primary table (normally this is the query that you have selected before clicking on Append Queries), and the table to append.

## Append

☒ Two tables ☐ Three or more tables

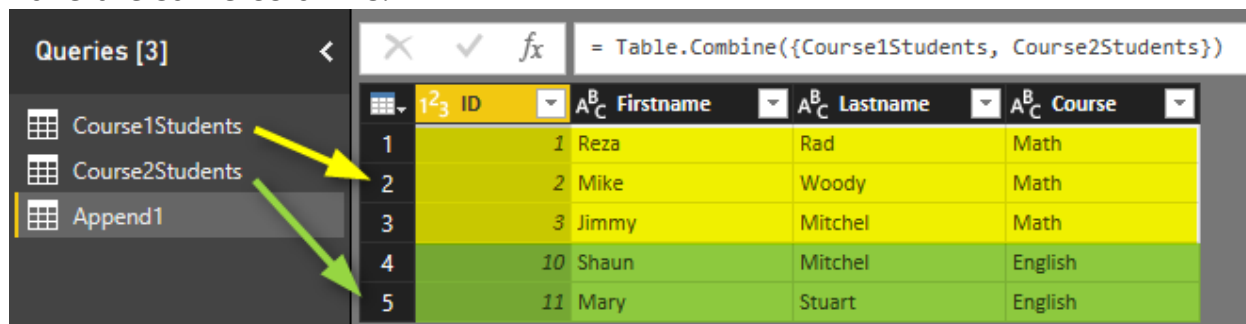
Primary table

Course1Students

Table to append to the primary table

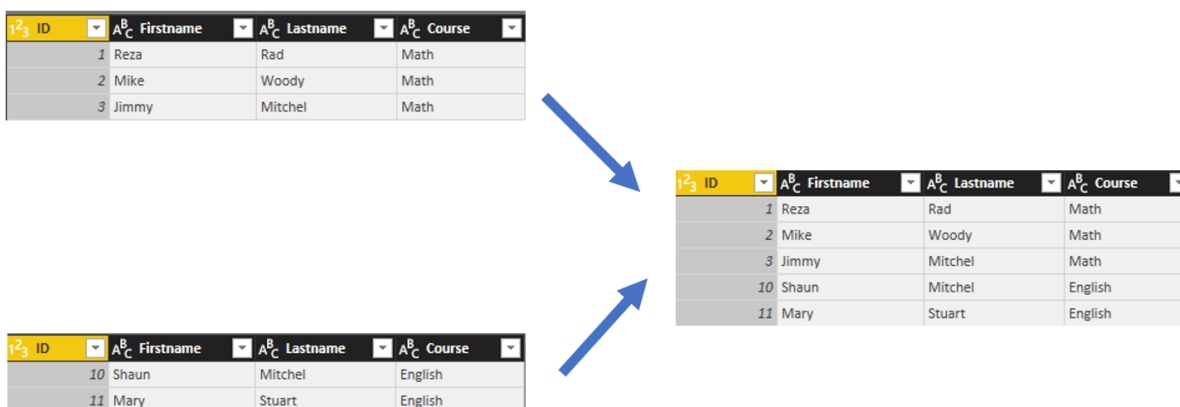
Course2Students

You can also choose to append Three or more tables and add tables to the list as you wish. For this example I have only two tables, so I'll continue with the above configuration. Append Queries append rows after each other, and because column names are exactly similar in both queries, the result set will have the same columns.



ID	Firstname	Lastname	Course
1	Reza	Rad	Math
2	Mike	Woody	Math
3	Jimmy	Mitchel	Math
10	Shaun	Mitchel	English
11	Mary	Stuart	English

The result of Append as simple as that



ID	Firstname	Lastname	Course
1	Reza	Rad	Math
2	Mike	Woody	Math
3	Jimmy	Mitchel	Math

ID	Firstname	Lastname	Course
10	Shaun	Mitchel	English
11	Mary	Stuart	English

ID	Firstname	Lastname	Course
1	Reza	Rad	Math
2	Mike	Woody	Math
3	Jimmy	Mitchel	Math
10	Shaun	Mitchel	English
11	Mary	Stuart	English

Append is similar to UNION ALL in T-SQL.

### What about Duplicates?

Append Queries will NOT remove duplicates. You have to use Group By or Remove Duplicate Rows to get rid of duplicates.

### What if columns in source queries are not exactly matched?

Append requires columns to be exactly similar to work in the best condition. If columns in source queries are different, append still works, but will create one



column in the output per each new column, if one of the sources doesn't have that column the cell value of that column for those rows will be null.

## Merge

Merge is another type of combining queries which are based on matching rows, rather than columns. The output of Merge will be a single query with;

- There should be joining or matching criteria between two queries. (for example StudentID column of both queries to be matched with each other)
- Number of rows will be dependent on matching criteria between queries
- Number of Columns will be dependent on what columns selected in the result set. (Merge will create a structured column as a result).

Understanding how Merge works might look a bit more complicated, but it will be very easy with an example, let's have a look at that in action;

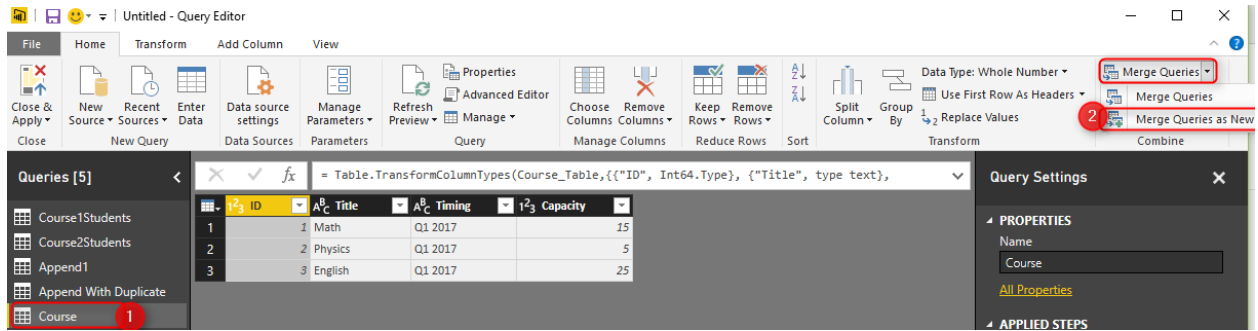
In addition to tables in the first example, consider that there is another table for Course's details as below:

1 <sup>2</sup> <sub>3</sub> ID	A <sup>B</sup> <sub>C</sub> Title	A <sup>B</sup> <sub>C</sub> Timing	1 <sup>2</sup> <sub>3</sub> Capacity
1	Math	Q1 2017	15
2	Physics	Q1 2017	5
3	English	Q1 2017	25

Now if I want to combine Course query with the Appended result of courseXstudents to see which students are part of which course with all details in each row, I need to use Merge Queries. Here is the appended result again;

1 <sup>2</sup> <sub>3</sub> ID	A <sup>B</sup> <sub>C</sub> Firstname	A <sup>B</sup> <sub>C</sub> Lastname	A <sup>B</sup> <sub>C</sub> Course
1	Reza	Rad	Math
2	Mike	Woody	Math
3	Jimmy	Mitchel	Math
10	Shaun	Mitchel	English
11	Mary	Stuart	English

Select Course Query first, and then Select Merge Queries (as New)



Merging Queries require joining criteria. Joining criteria is field(s) in each source query that should be matched with each other to build the resulting query. In this example, I want to Merge Course query with Append1, based on Title of the course.

## Merge

Select tables and matching columns to create a merged table.

Course 1

ID	Title 3	Timing	Capacity
1	Math	Q1 2017	15
2	Physics	Q1 2017	5
3	English	Q1 2017	25

Append1 2

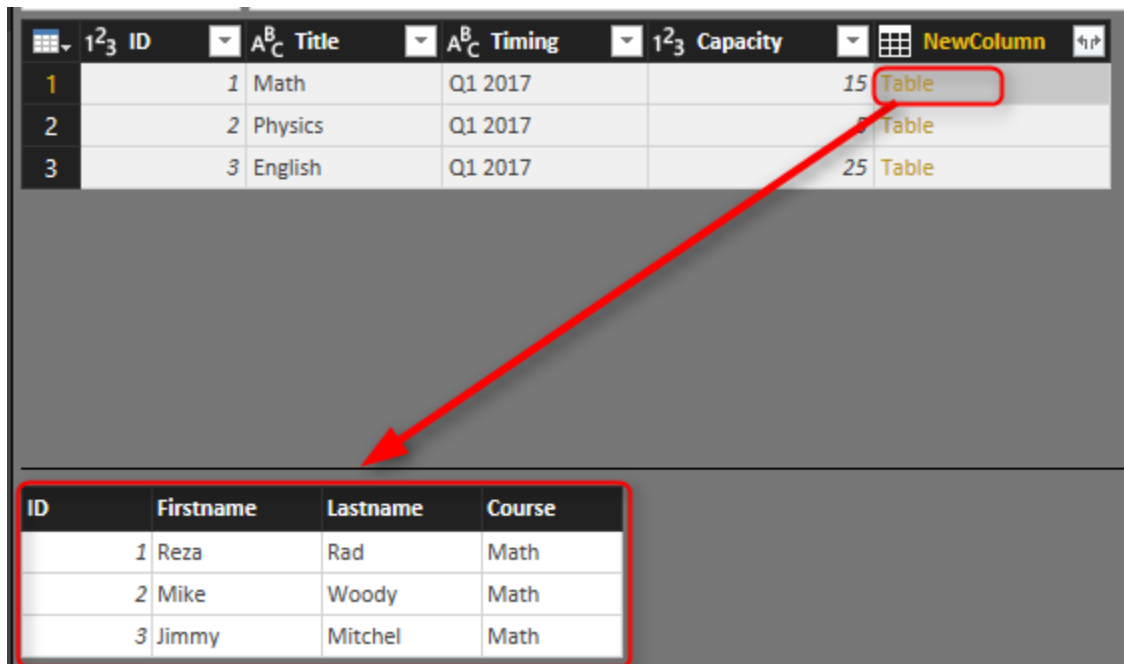
ID	Firstname	Lastname	Course 4
1	Reza	Rad	Math
2	Mike	Woody	Math
3	Jimmy	Mitchel	Math
10	Shaun	Mitchel	English
11	Mary	Stuart	English

Join Kind

Left Outer (all from first, matching from second)

**i** The selection has matched 2 out of the first 3 rows.

I'll talk about types of join later. For now, continue the selection, and you will see these two queries match with each other based on the Course title, result query will be same as the first query (Course in this example), plus one additional column named as NewColumn with a table in each cell. This is a structured column which can be expanded into underlying tables. If you click on an empty area of the cell containing one of these tables, you will see the sub table underneath.

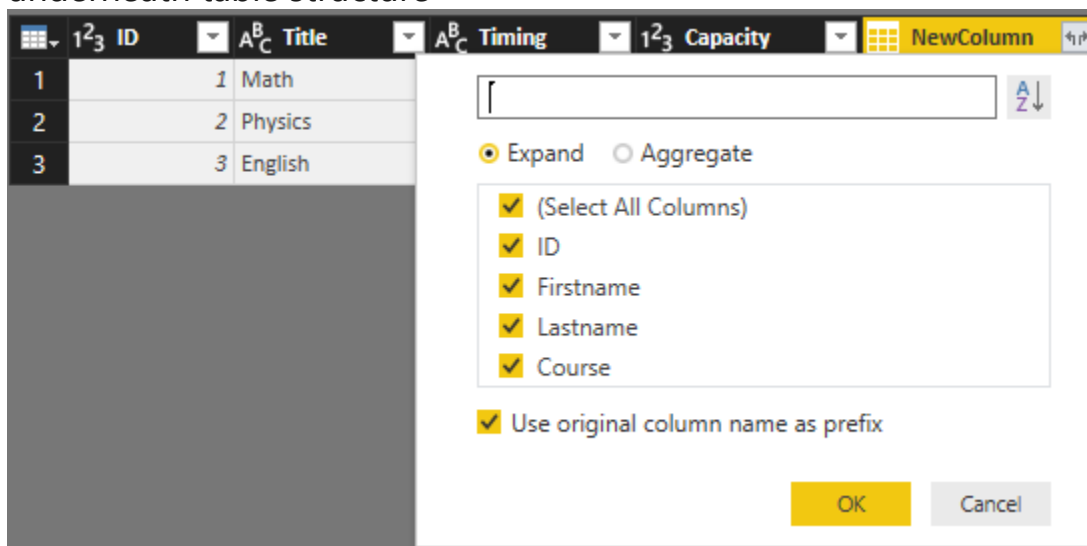


ID	Title	Timing	Capacity	NewColumn
1	Math	Q1 2017	15	Table
2	Physics	Q1 2017	5	Table
3	English	Q1 2017	25	Table

ID	Firstname	Lastname	Course
1	Reza	Rad	Math
2	Mike	Woody	Math
3	Jimmy	Mitchel	Math

Now click on Expand column icon, and expand the New Column to all underneath table structure

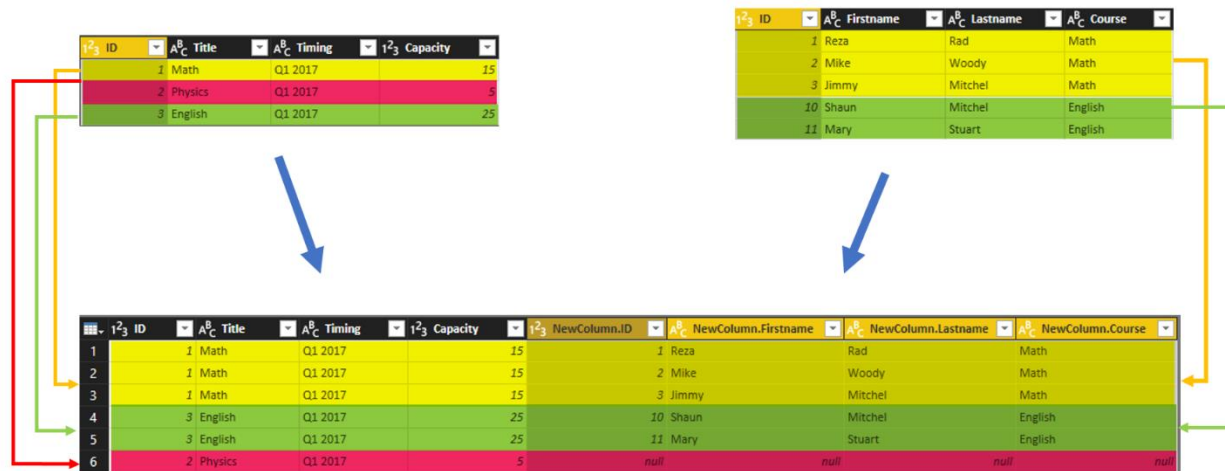


ID	Title	Timing	Capacity	NewColumn
1	Math			
2	Physics			
3	English			

The result will be a table including columns from both tables, and rows matching with each other.

ID	Title	Timing	Capacity	NewColumn.ID	NewColumn.Firstname	NewColumn.Lastname	NewColumn.Course
1	Math	Q1 2017	15	1	Reza	Rad	Math
2	Math	Q1 2017	15	2	Mike	Woody	Math
3	Math	Q1 2017	15	3	Jimmy	Mitchel	Math
4	English	Q1 2017	25	10	Shaun	Mitchel	English
5	English	Q1 2017	25	11	Mary	Stuart	English
6	Physics	Q1 2017	5	null	null	null	null

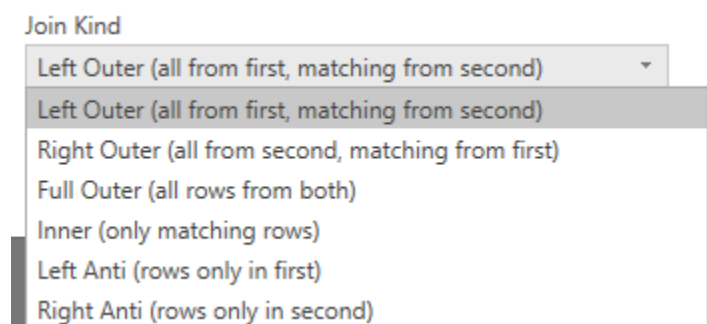
Columns in the left-hand side are coming from Course table, columns in the right-hand side are coming from Students table. Values in the rows only appear in matching criteria. First three rows are students of Math course, then two students for the English course, and because there is no student for Physics course you will see null values for students columns.



Merge is similar to JOIN in T-SQL

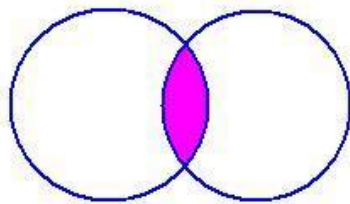
## Join Types

There are six types of joins supported in Power BI as below, depends on the effect on the result set based on matching rows, each of these types works differently.

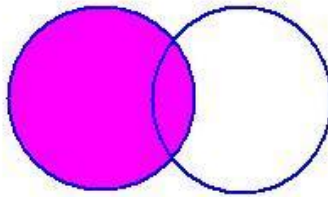


Explaining what each join type will do is a different post which I wrote about it [here](#). For now, this picture explains it very well:

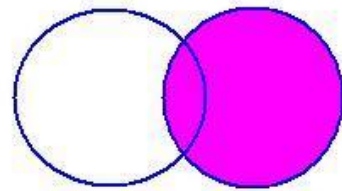
## JOINS AND SET OPERATIONS IN RELATIONAL DATABASES



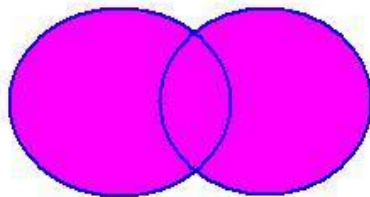
Inner join (result similar to Intersect)



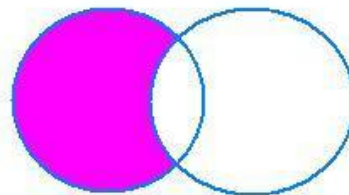
Left outer join



Right outer join



Full outer join



Minus

Picture referenced from [http://www.udel.edu/evelyn/SQL-Class2/SQLclass2\\_Join.html](http://www.udel.edu/evelyn/SQL-Class2/SQLclass2_Join.html)

# Choose the Right Merge Join Type in Power BI

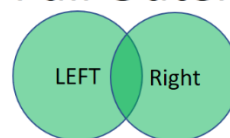
Posted by [Reza Rad](#) on Jul 26, 2017

1 <sub>2</sub> order_id	A <sub>2</sub> order_date	1.2 amount	1 <sub>2</sub> customer_id
1	07/04/1776	234.56	1
2	03/14/1760	78.5	3
3	05/23/1784	124	2
4	09/03/1790	65.5	3
5	9/04/2000	100	6

Matching

Not Matching

## Full Outer



1 <sub>2</sub> order_id	A <sub>2</sub> order_date	1.2 amount	1 <sub>2</sub> customer_id	A <sub>2</sub> first_name	A <sub>2</sub> last_name	A <sub>2</sub> email	A <sub>2</sub> address	A <sub>2</sub> city	A <sub>2</sub> state	1 <sub>2</sub> zipcode
1	07/04/1776	234.56	1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	03/14/1760	78.5	3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
4	09/03/1790	65.5	3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
3	05/23/1784	124	2	John	Adams	jadam@usa.gov	1250 Hancock St	Quincy	MA	2169
				James	Madison	jmadison@usa.gov	11350 Constitution Hwy	Orange	VA	22960
				James	Monroe	jmonroe@usa.gov	2050 James Monroe Parkway	Charlottesville	VA	22902
5	9/04/2000	100	6							

Matching

Not Matching

1 <sub>2</sub> customer_id	A <sub>2</sub> first_name	A <sub>2</sub> last_name	A <sub>2</sub> email	A <sub>2</sub> address	A <sub>2</sub> city	A <sub>2</sub> state	1 <sub>2</sub> zipcode
1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	John	Adams	jadam@usa.gov	1250 Hancock St	Quincy	MA	2169
3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
4	James	Madison	jmadison@usa.gov	11350 Constitution Hwy	Orange	VA	22960
5	James	Monroe	jmonroe@usa.gov	2050 James Monroe Parkway	Charlottesville	VA	22902

Previously I have written a blog post explaining two ways of combining data sets; [Append vs. Merge](#). In this post, I want to explain in details what is the difference between all types of Merge Type and explaining how to choose the right merge (or Join) type. These Merge types are very similar to join types in relational databases. Because many of people who work with Power BI might not have experience working with relational databases, so I think this post is a good explanation in details what are these types and when to use them. If you want to learn more about Power BI; read [Power BI book from Rookie to Rock Star](#).

## What is Merge?

Combining two data sets can be done in multiple ways. One of the ways of combining data sets is Merging datasets. Merge is similar to Join in relational databases. Merging two data sets requires some joining fields, and the result will be combined set of columns from both data sets.

1 <sup>2</sup> order_id	A <sup>0</sup> order_date	1.2 amount	1 <sup>2</sup> customer_id
1	07/04/1776	234.56	1
2	03/14/1760	78.5	3
3	05/23/1784	124	2
4	09/03/1790	65.5	3

Merge

1 <sup>2</sup> order_id	A <sup>0</sup> order_date	1.2 amount	1 <sup>2</sup> customer_id	A <sup>0</sup> first_name	A <sup>0</sup> last_name	A <sup>0</sup> email	A <sup>0</sup> address	A <sup>0</sup> city	A <sup>0</sup> state	1 <sup>2</sup> zipcode
1	07/04/1776	234.56	1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	03/14/1760	78.5	3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
4	09/03/1790	65.5	3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
3	05/23/1784	124	2	John	Adams	jadams@usa.gov	1250 Hancock St	Quincy	MA	2169

1 <sup>2</sup> customer_id	A <sup>0</sup> first_name	A <sup>0</sup> last_name	A <sup>0</sup> email	A <sup>0</sup> address	A <sup>0</sup> city	A <sup>0</sup> state	1 <sup>2</sup> zipcode
1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	John	Adams	jadams@usa.gov	1250 Hancock St	Quincy	MA	2169
3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
4	James	Madison	jmadison@usa.gov	11350 Constitution Hwy	Orange	VA	22960
5	James	Monroe	jmonroe@usa.gov	2050 James Monroe Parkway	Charlottesville	VA	22902

Let's go through it with an example;

Consider two data tables as below

\*Data tables are sourced from [this web page](#). Download it from the link at the top of this post.

Customers Table:

customer_id	first_name	last_name	email	address	city	state	zipcode
1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	John	Adams	jadams@usa.gov	1250 Hancock St	Quincy	MA	2169
3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
4	James	Madison	jmadison@usa.gov	11350 Constitution Hwy	Orange	VA	22960
5	James	Monroe	jmonroe@usa.gov	2050 James Monroe Parkway	Charlottesville	VA	22902

Orders Table:

order_id	order_date	amount	customer_id
1	07/04/1776	\$234.56	1
2	03/14/1760	\$78.50	3
3	05/23/1784	\$124.00	2
4	09/03/1790	\$65.50	3

Merging these two tables, gives you a dataset with the combined set of columns like below;

1 <sup>2</sup> order_id	A <sup>0</sup> order_date	1.2 amount	1 <sup>2</sup> customer_id	A <sup>0</sup> first_name	A <sup>0</sup> last_name	A <sup>0</sup> email	A <sup>0</sup> address	A <sup>0</sup> city	A <sup>0</sup> state	1 <sup>2</sup> zipcode
1	07/04/1776	234.56	1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	03/14/1760	78.5	3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
4	09/03/1790	65.5	3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
3	05/23/1784	124	2	John	Adams	jadams@usa.gov	1250 Hancock St	Quincy	MA	2169



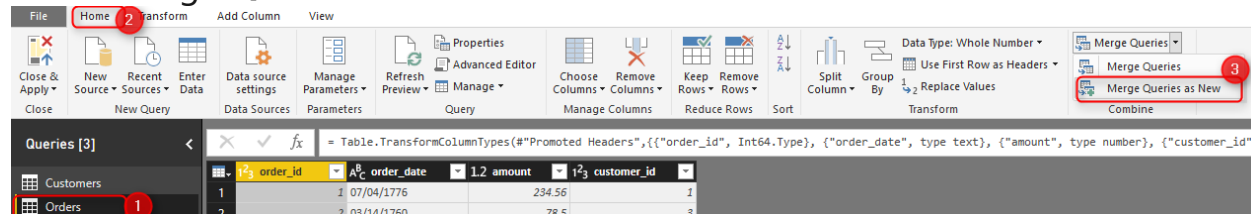
For merging data sets, you need to have some joining fields. In this example the joining field is customer\_id. To get a data set that includes all columns from both tables based on customer\_id relationship, this is how you can join tables to each other:

## How to Merge Queries

Select the first query (for this example: Orders), and then from the home tab, Merge queries. There are two options here:

- Merge Queries: This will amend the existing query (orders), to be the result of Merging.
- Merge Queries as New: This will not change the existing query. It will create a reference from it, and the result of merging would be another query.

Select Merge Queries as New.



In the Merge configuration window, select the second table (Customers), then select the joining the field in each table (Customer\_id). You will also see a number of matching rows as extra information there.



## Merge

Select tables and matching columns to create a merged table.

1

Orders

order_id	order_date	amount	customer_id
1	07/04/1776	234.56	1
2	03/14/1760	78.5	3
3	05/23/1784	124	2
4	09/03/1790	65.5	3

3

2

Customers

customer_id	first_name	last_name	email	address	city	state
1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA
2	John	Adams	jadams@usa.gov	1250 Hancock St	Quincy	MA
3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA
4	James	Madison	jmadison@usa.gov	11350 Constitution Hwy	Orange	VA
5	James	Monroe	jmonroe@usa.gov	2050 James Monroe Parkway	Charlottesville	VA

4

Join Kind

Left Outer (all from first, matching from second)

✓ The selection has matched 4 out of the first 4 rows.

OK Cancel

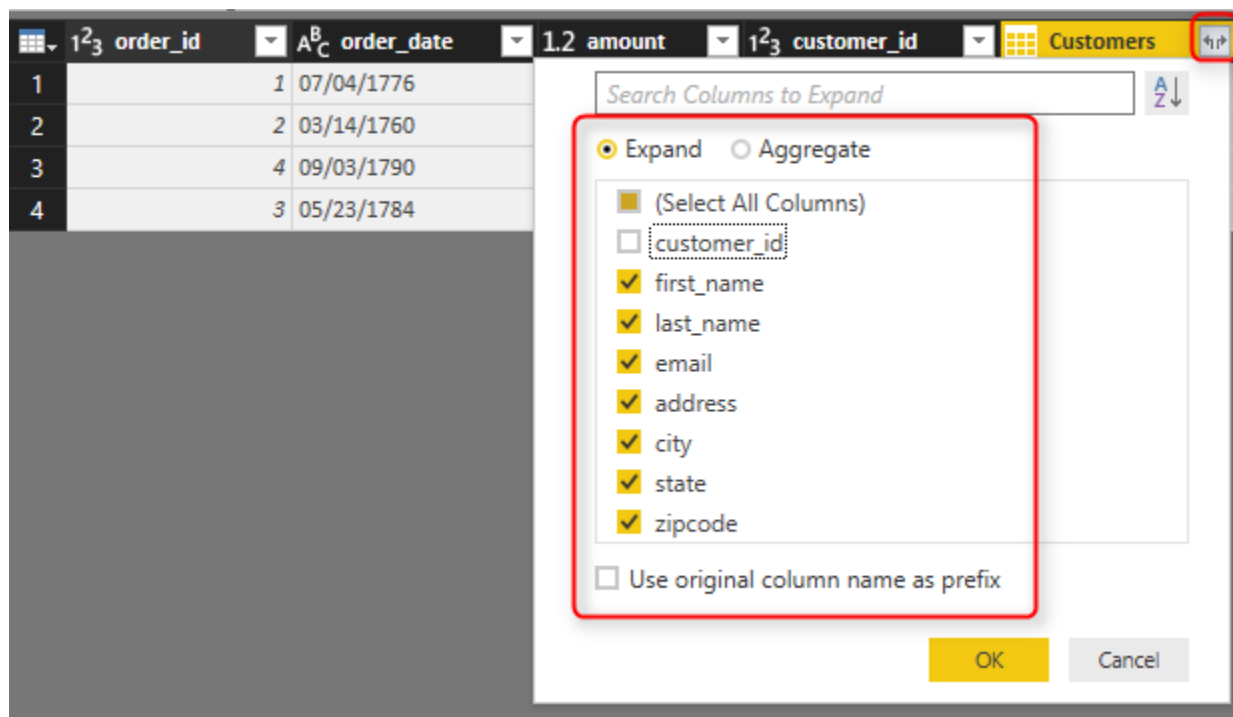
The result of this operation will be a new query named as Merge1, which has the combined result of these two queries.

Customers	Orders	Merge1
1	1	1
2	2	2
3	3	3
4	4	4

customer_id	first_name	last_name	email	address	city	state	zipcode
1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121

The table in Customers column is a sub table from customer table for records matching with that customer\_id. You can then expand it to columns you want;



and the final result will be now all columns in one query;

	1 <sup>2</sup> order_id	A <sup>B</sup> order_date	1.2 amount	1 <sup>2</sup> customer_id	A <sup>B</sup> first_name	A <sup>B</sup> last_name	A <sup>B</sup> email	A <sup>B</sup> address	A <sup>B</sup> city	A <sup>B</sup> state	1 <sup>2</sup> zipcode
1	1	07/04/1776	234.56	1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	2	03/14/1760	78.5	3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
3	4	09/03/1790	65.5	3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
4	3	05/23/1784	124	2	John	Adams	jadams@usa.gov	1250 Hancock St	Quincy	MA	2169

This process is called Merge Join. However, there are some configurations you can do for this;

## Merging Based on Multiple Columns

You can easily use multiple columns for merging two datasets. Just select them in order with holding Ctrl key of the keyboard.

## Merge

Select tables and matching columns to create a merged ta

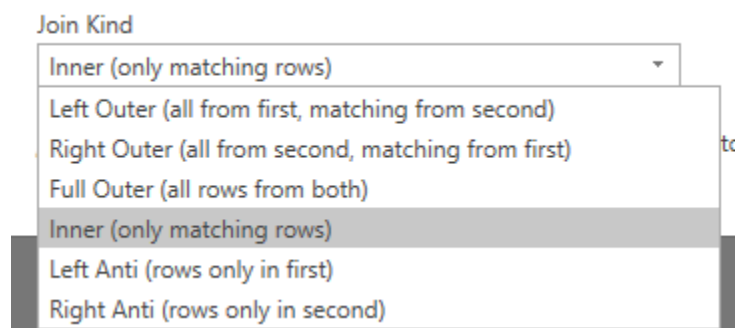
Orders

order_id	order_date	amount	customer_id
1	07/04/1776	234.56	1
2	03/14/1760	78.5	3
3	05/23/1784	124	2
4	09/03/1790	65.5	3

This method can be really helpful because the relationship tab in Power BI Desktop doesn't allow you to create a relationship based on multiple columns. In Power Query, however, you can create the merge and create a unique field for the relationship in Power BI Desktop. [Here](#) is my blog post about it.

## Merge Types

In addition to the merging column (or joining field); the type of Merge is very important. You can get a totally different result set with choosing a different type of merge. Here is where you can see the Join Kind (or Merge Type);



At the moment of writing this blog post, there are six types of joins. About a year ago or even before that fewer number of joins were available. However, you could always [choose your Join type in Power Query M script](#). Every one of these joins gets a different result set, let's see what their difference is.

## Left and Right

To start, you need to know the concept of Left and Right tables (or queries). When you merge two data sets, the first query is considered as LEFT and the second as RIGHT.

### Merge

Select tables and matching columns to create a merged table.

LEFT

order_id	order_date	amount	customer_id
1	07/04/1776	234.56	1
2	03/14/1760	78.5	3
3	05/23/1784	124	2
4	09/03/1790	65.5	3

RIGHT

customer_id	first_name	last_name	email	address
1	George	Washington	gWASHINGTON@usa.gov	3200 Mt Vernon
2	John	Adams	jADAMS@usa.gov	1250 Hancock
3	Thomas	Jefferson	tJEFFERSON@usa.gov	931 Thomas
4	James	Madison	jMADISON@usa.gov	11350 Consti
5	James	Monroe	jMONROE@usa.gov	2050 James M

In the example above; Orders is LEFT query, and Customers is the RIGHT query. You can change them if you want of course. Understanding this is important because most of Join Kinds works with the concept of left or right or both.

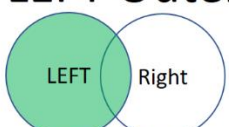
### Left Outer (All from first, matching from second)

The first type of Join/Merge is Left Outer. This means the LEFT query is the important one. All records from this query (LEFT or FIRST) will be shown in the

result set plus their matching rows in the right (or second table). This type of join is the default type. If you don't specify the Join Kind, it will always be Left Outer.

For example in the first Merge example we have done, you can see that the result set is four records, representing four records from the left table (orders), and their matching rows in the customer table.

**LEFT Outer**



order_id	order_date	amount	customer_id
1	07/04/1776	234.56	1
2	03/14/1760	78.5	3
3	05/23/1784	124	3
4	09/03/1790	65.5	3
5	9/04/2000	100	6

order_id	order_date	amount	customer_id	first_name	last_name	email	address	city	state	zipcode
1	07/04/1776	234.56	1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	03/14/1760	78.5	3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
3	09/03/1790	65.5	3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
4	05/23/1784	124	2	John	Adams	jadams@usa.gov	1250 Hancock St	Quincy	MA	2169
5	9/04/2000	100	6							

customer_id	first_name	last_name	email	address	city	state	zipcode
1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	John	Adams	jadams@usa.gov	1250 Hancock St	Quincy	MA	2169
3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
4	James	Madison	jmadison@usa.gov	11350 Constitution Hwy	Orange	VA	22960
5	James	Monroe	jmonroe@usa.gov	2050 James Monroe Parkway	Charlottesville	VA	22902

Matching: Rows 1, 2, 3, 4 of the result set correspond to matching rows in the customer table.

Not Matching: Row 5 of the result set (customer\_id 6) does not have a matching row in the customer table.

As you can see in the screenshot above; there are two customers who won't be in the result set. Customers with id 4 and 5. because these rows are not matching with the customer\_id field in the orders table. In LEFT Outer merge, only records from Left table with matching rows of the right table will be selected.

### Right Outer (all rows from second, matching from first)

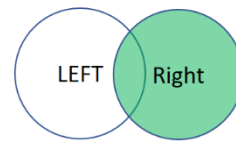
Sometimes you need to fetch all rows from the second table, regardless if they exist in the first table or not. In that case, you would need to use another type of Join called Right Outer. With this type of Join, you get all rows from the RIGHT (or second) table, with their matching rows from left (or first table). Here is an example:

order_id	order_date	amount	customer_id
1	07/04/1776	234.56	1
2	03/14/1760	78.5	3
3	05/23/1784	124	2
4	09/03/1790	65.5	3
5	9/04/2000	100	6

Matching

Not Matching

## RIGHT Outer



order_id	order_date	amount	customer_id	first_name	last_name	email	address	city	state	zipcode
1	07/04/1776	234.56	1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	03/14/1760	78.5	3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
4	09/03/1790	65.5	3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
3	05/23/1784	124	2	John	Adams	jadams@usa.gov	1250 Hancock St	Quincy	MA	2169
				James	Madison	jmadison@usa.gov	11350 Constitution Hwy	Orange	VA	22960
				James	Monroe	jmonroe@usa.gov	2050 James Monroe Parkway	Charlottesville	VA	22902

customer_id	first_name	last_name	email	address	city	state	zipcode
1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	John	Adams	jadams@usa.gov	1250 Hancock St	Quincy	MA	2169
3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
4	James	Madison	jmadison@usa.gov	11350 Constitution Hwy	Orange	VA	22960
5	James	Monroe	jmonroe@usa.gov	2050 James Monroe Parkway	Charlottesville	VA	22902

As you can see in the above screenshot; all rows from customers table is showed in the result set. However only four rows of that are matching with the orders table. If there is a record in orders table that doesn't match it will come as Null (two red rows in the result set).

### Full Outer (all rows from both)

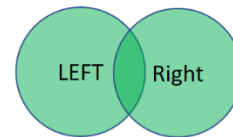
This type of join/merge is normally the safest among all. This will return all rows from both tables (matching and non-matching). You will have all rows from a first table, and all rows from the second table, and all matching rows. With this method, you won't lose any records.

123 order_id	A0C order_date	1.2 amount	123 customer_id
1	07/04/1776	234.56	1
2	03/14/1760	78.5	3
3	05/23/1784	124	2
4	09/03/1790	65.5	3
5	9/04/2000	100	6

Matching

Not Matching

## Full Outer



123 order_id	A0C order_date	1.2 amount	123 customer_id	A0C first_name	A0C last_name	A0C email	A0C address	A0C city	A0C state	123 zipcode
1	07/04/1776	234.56	1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	03/14/1760	78.5	3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
4	09/03/1790	65.5	3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
3	05/23/1784	124	2	John	Adams	jadams@usa.gov	1250 Hancock St	Quincy	MA	2169
				James	Madison	jmadison@usa.gov	11350 Constitution Hwy	Orange	VA	22960
				James	Monroe	jmonroe@usa.gov	2050 James Monroe Parkway	Charlottesville	VA	22902
5	9/04/2000	100	6							

Matching

Not Matching

123 customer_id	A0C first_name	A0C last_name	A0C email	A0C address	A0C city	A0C state	123 zipcode
1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	John	Adams	jadams@usa.gov	1250 Hancock St	Quincy	MA	2169
3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
4	James	Madison	jmadison@usa.gov	11350 Constitution Hwy	Orange	VA	22960
5	James	Monroe	jmonroe@usa.gov	2050 James Monroe Parkway	Charlottesville	VA	22902

This result set as you can see is seven rows. Four rows matching in both tables. 2 rows only in the customer table, but not in the orders table. One row only in orders table, but not in the customers table.

## Inner (only matching rows)

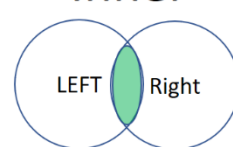
This method only selects matching rows. You will not have any record with null values (because these records generated as a result of not matching). Here is an example;

123 order_id	A0C order_date	1.2 amount	123 customer_id
1	07/04/1776	234.56	1
2	03/14/1760	78.5	3
3	05/23/1784	124	2
4	09/03/1790	65.5	3
5	9/04/2000	100	6

Matching

Not Matching

## Inner



123 order_id	A0C order_date	1.2 amount	123 customer_id	A0C first_name	A0C last_name	A0C email	A0C address	A0C city	A0C state	123 zipcode
1	07/04/1776	234.56	1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	03/14/1760	78.5	3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
4	09/03/1790	65.5	3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
3	05/23/1784	124	2	John	Adams	jadams@usa.gov	1250 Hancock St	Quincy	MA	2169

Matching

Not Matching

123 customer_id	A0C first_name	A0C last_name	A0C email	A0C address	A0C city	A0C state	123 zipcode
1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	John	Adams	jadams@usa.gov	1250 Hancock St	Quincy	MA	2169
3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
4	James	Madison	jmadison@usa.gov	11350 Constitution Hwy	Orange	VA	22960
5	James	Monroe	jmonroe@usa.gov	2050 James Monroe Parkway	Charlottesville	VA	22902



There are only four rows that are matching between two tables. Rows with customer\_id 1, 2, and 3. all not matching rows will be excluded from the result set.

### Let Anti (rows only in first)

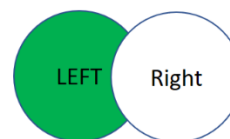
If you are only interested in rows from the LEFT (first) table, then this is the option to select. This means rows that are in the first table and DO NOT match with the second table. So, only Not matching rows from the first table. With Anti options, you always get null for the second data set, because these rows don't exist there. Anti options are good for finding rows that exist in one table but not in the other one. Here is an example:

1 <sup>2</sup> order_id	A <sup>0</sup> order_date	1.2 amount	1 <sup>2</sup> customer_id
1	07/04/1776	234.56	1
2	03/14/1760	78.5	3
3	05/23/1784	124	2
4	09/03/1790	65.5	3
5	9/04/2000	100	6

Matching

Not Matching

### Left Anti



1 <sup>2</sup> order_id	A <sup>0</sup> order_date	1.2 amount	1 <sup>2</sup> customer_id	A <sup>0</sup> first_name	A <sup>0</sup> last_name	A <sup>0</sup> email	A <sup>0</sup> address	A <sup>0</sup> city	A <sup>0</sup> state	1 <sup>2</sup> zipcode
5	9/04/2000	100	6	null	null	null	null	null	null	null

1 <sup>2</sup> customer_id	A <sup>0</sup> first_name	A <sup>0</sup> last_name	A <sup>0</sup> email	A <sup>0</sup> address	A <sup>0</sup> city	A <sup>0</sup> state	1 <sup>2</sup> zipcode
1	George	Washington	gswashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	John	Adams	jadams@usa.gov	1250 Hancock St	Quincy	MA	2169
3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
4	James	Madison	jmadison@usa.gov	11350 Constitution Hwy	Orange	VA	22960
5	James	Monroe	jmonroe@usa.gov	2050 James Monroe Parkway	Charlottesville	VA	22902

This will find the only one row that exists in the Orders table and does not match with any of the rows in the customer's table.

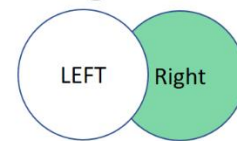
### Right Anti (rows only in second)

Similar to Left Anti; this method will give you only not matching rows.

However, this time from the second (Right) table. You can find out what rows in the right table are not matching with the left table. Here is the example;

123 order_id	A6 order_date	1.2 amount	123 customer_id
1	07/04/1776	234.56	1
2	03/14/1760	78.5	3
3	05/23/1784	124	2
4	09/03/1790	65.5	3
5	9/04/2000	100	6

## Right Anti



123 order_id	A6 order_date	1.2 amount	123 customer_id	A6 first_name	A6 last_name	A6 email	A6 address	A6 city	A6 state	123 zipcode
null	null	null	null	James	Madison	jmadison@usa.g...	11350 Constitutio...	Orange	VA	22960
null	null	null	null	James	Monroe	jmonroe@usa.gov	2050 James Monr...	Charlottesville	VA	22902

	123 customer_id	A6 first_name	A6 last_name	A6 email	A6 address	A6 city	A6 state	123 zipcode
Matching	1	George	Washington	gWASHINGTON@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
	2	John	Adams	jADAMS@usa.gov	1250 Hancock St	Quincy	MA	2169
	3	Thomas	Jefferson	tJEFFERSON@usa.gov	931 Thomas Jefferson Pkwy	Charlottesville	VA	22902
Not Matching	4	James	Madison	jmadison@usa.gov	11350 Constitution Hwy	Orange	VA	22960
	5	James	Monroe	jmonroe@usa.gov	2050 James Monroe Parkway	Charlottesville	VA	22902

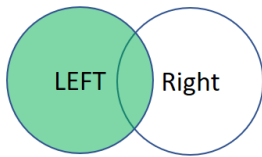
This result set is all rows from the customer table (right table) that does NOT match with orders (first table).

## Summary

Different Join Kinds in Merge returns a different result set. Make sure to select the right join kind to avoid any issues later. There are six types of joins as below;

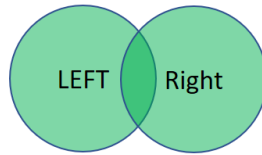
- Left Outer: Rows from left table and matching with the right
- Right Outer: Rows from right table and matching with the left
- Full Outer: Rows from both tables (matching or not matching)
- Inner: Only matching rows from both tables
- Left Anti: Not matching rows from the left table
- Right Anti: Not matching rows from the right table

## LEFT Outer



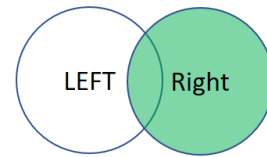
All rows from left and matching from right

## Full Outer



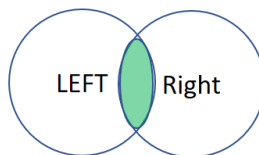
All rows from both: matching and not matching

## RIGHT Outer



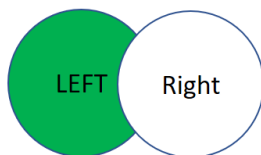
All rows from right and matching from left

## Inner



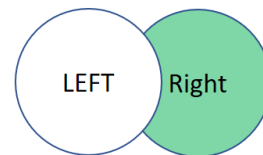
Only matching rows

## Left Anti



Not matching rows from left

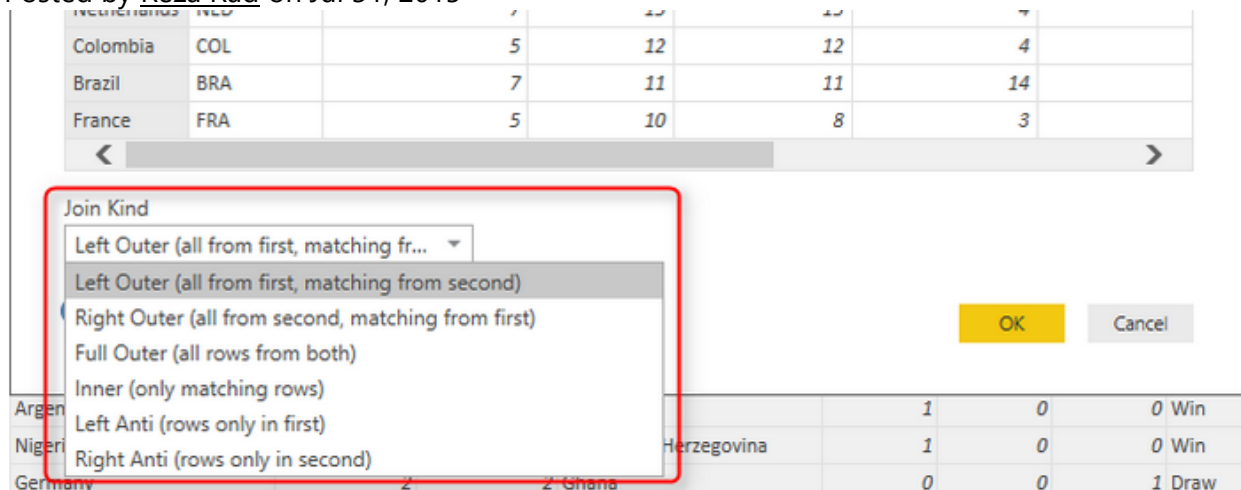
## Right Anti



Not matching rows from right

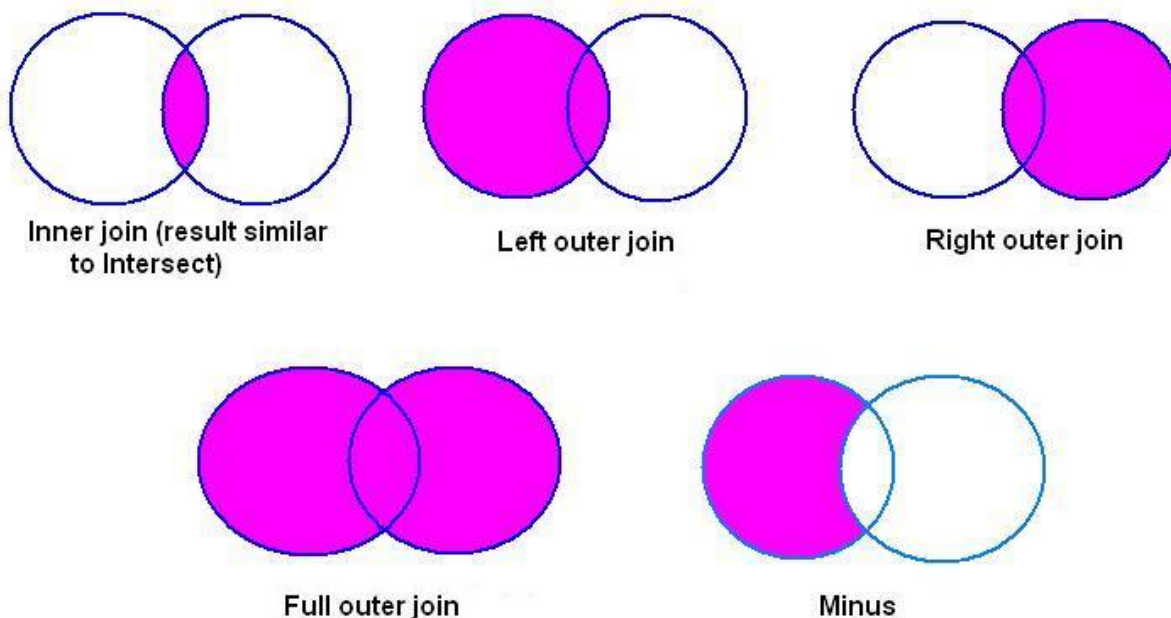
# How to Change Joining Types in Power BI and Power Query

Posted by [Reza Rad](#) on Jul 31, 2015



Joining tables is not a new concept, I bet all readers of my blog at least have a clue about that. However there are different types of joins, and applying these types of Joins are not all possible through Power Query GUI. [Power BI recently took a step and implemented that in the GUI](#), however, you might like to know how to apply that in the Power Query. The trick is that M is your friend, You can do whatever you want behind the scenes with M script. I don't want to go through the details of explaining every join type here. The picture below illustrated it perfectly;

### JOINS AND SET OPERATIONS IN RELATIONAL DATABASES

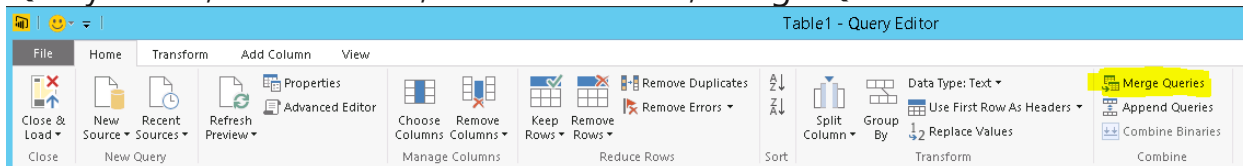


Picture referenced from [http://www.udel.edu/evelyn/SQL-Class2/SQLclass2\\_Join.html](http://www.udel.edu/evelyn/SQL-Class2/SQLclass2_Join.html)

Now let's see how to use joins through Power BI and Power Query;

## Power BI Desktop

In Power BI Desktop you can join two tables with Merge menu item in the Query Editor, in Home tab, Under Combine, Merge Queries.



The Merge Window will appear with the ability to select the first table (Left part of the join), and the second table (Right part of the join). You can choose columns that you want to participate as joining key within an order (you can choose multiple columns with Ctrl Key). And there is join kindly that you can choose.

## Merge

Select a table and matching columns to create a merged table.

Venue	Stage	Match	Team 1	Result 1	Result 2	Team 2	Win	Loss	Draw	Resu
Arena Corinthians	Group A	1	Brazil	3	1	Croatia	1	0	0	Win
Arena das Dunas	Group A	2	Mexico	1	0	Cameroon	1	0	0	Win
Arena Fonte Nova	Group B	3	Spain	1	5	Netherlands	0	1	0	Loss
Arena Pantanal	Group B	4	Chile	3	1	Australia	1	0	0	Win
Estádio Mineirão	Group C	5	Colombia	3	0	Greece	1	0	0	Win

TeamGoals

Teams^v	Teams^v2	Matches Played^v	Goals for^v	Goals scored^v	Goals Against^v	Penalty goal^v
Germany	GER	7	18	18	4	
Netherlands	NED	7	15	15	4	
Colombia	COL	5	12	12	4	
Brazil	BRA	7	11	11	14	
France	FRA	5	10	8	3	

Join Kind

- Left Outer (all from first, matching fr...
- Left Outer (all from first, matching from second)
- Right Outer (all from second, matching from first)
- Full Outer (all rows from both)
- Inner (only matching rows)
- Left Anti (rows only in first)
- Right Anti (rows only in second)

OK Cancel

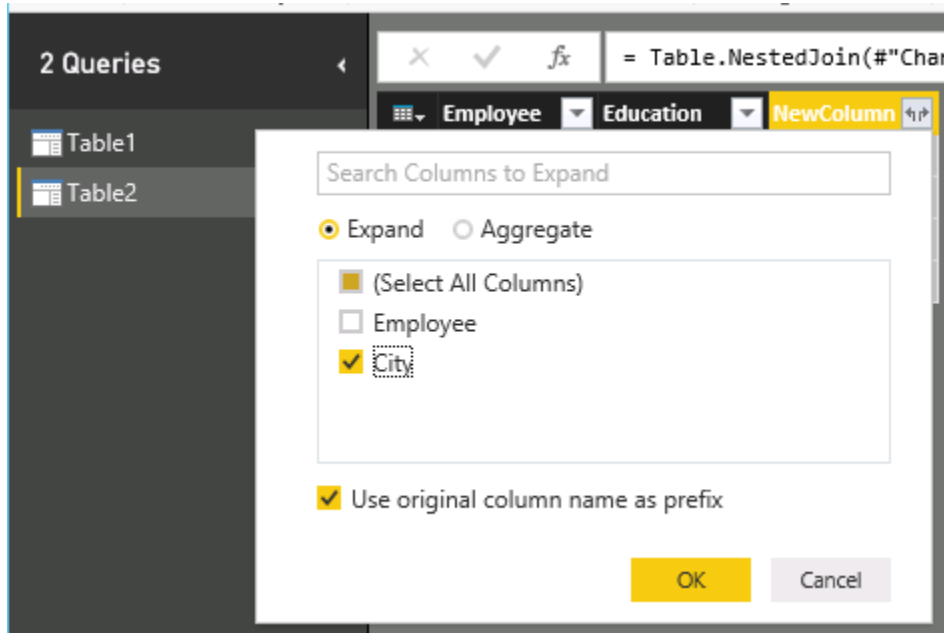
Argentina			1	0	0	Win
Nigeria			1	0	0	Win
Germany			0	0	1	Draw

The default behavior is left outer join, which means all records from the first table if there is any record in the first table that matches record(s) in the second table it would be listed as well.

After joining tables, the second table will appear as a field that has table value in its cells. What you need to do is to select columns that you want to show in the result set.

	Employee	Education	NewColumn
1	Mark	PHD	Table
2	Fernando	MsC	Table
3	Reza	Bachelor	Table
4	Steve	Diploma	Table

Choose columns as below:



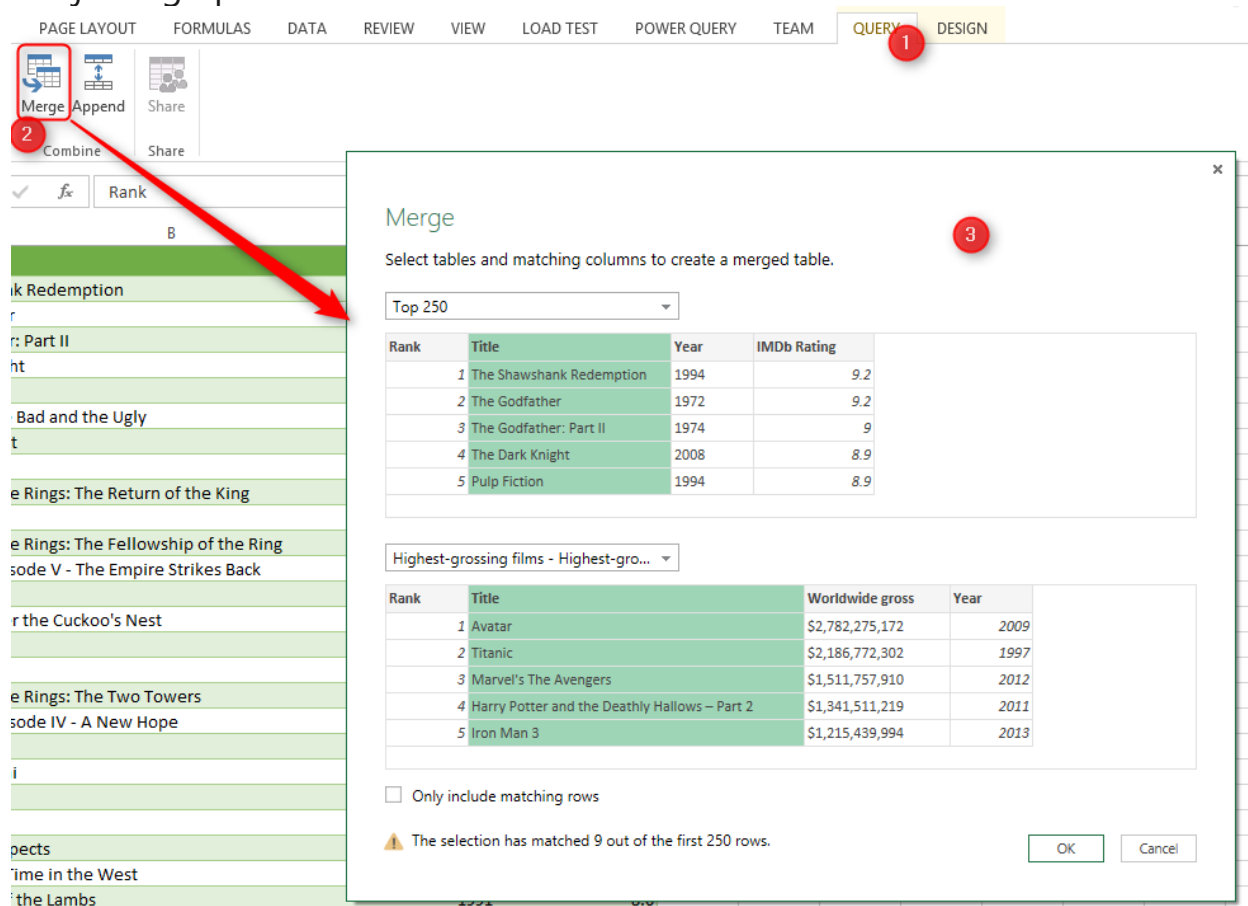
You can also choose from one of the other join types as mentioned below:

- Left Outer (all from first, matching from second): this option was the default behavior previously within Merge dialog
- Right Outer (all from second, matching from first)
- Full Outer (all rows from both)
- Inner (only matching rows); this option was available previously through "Choose only matching rows" option in Merge dialog
- Left Anti (rows only in first)
- Right Anti (rows only in second)

## Power Query

At the time of writing this blog post, The Power Query Editor (GUI) only supports two types of joins mentioned above: Left Join, and Inner Join.

You should follow the same path through Merge Queries, and then you will see joining options as below:

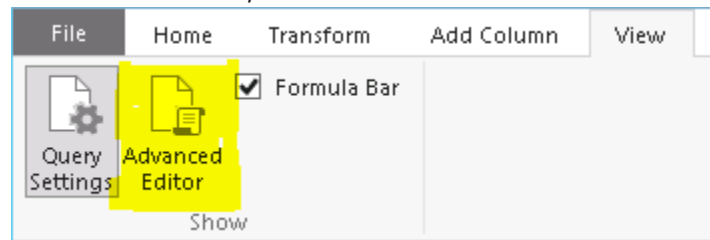


As you see the default behavior is Left join. You can change it to inner join with selecting "Only include matching rows".

## Change Through M

You can apply any join type that you want simply by going to advanced editor, and changing the M script as below:

Go to View tab, and click on Advanced Editor:





In the Advanced Editor query window, you can see the M script that builds the result set. Find the join function and change the JoinKind.

Advanced Editor

## Table2

```
let
    Source = Excel.Workbook(File.Contents("C:\Users\adm_radr\Documents\join.xlsx"), null, true),
    Table2_Table = Source[[Item="Table2",Kind="Table"]][Data],
    #"Changed Type" = Table.TransformColumnTypes(Table2_Table,{{"Employee", type text}, {"Education", type text}}),
    #"Merged Queries" = Table.NestedJoin(#"Changed Type",{"Employee"},Table1,{"Employee"},"NewColumn",JoinKind.FullOuter),
    #"Expanded NewColumn" = Table.ExpandTableColumn(#"Merged Queries", "NewColumn", {"City"}, {"NewColumn.City"})
in
    #"Expanded NewColumn"
```

JoinKind is an enumeration type that can have below values:

- JoinKind.Inner=0
- JoinKind.LeftOuter=1
- JoinKind.RightOuter=2
- JoinKind.FullOuter=3
- JoinKind.LeftAnti=4
- JoinKind.RightAnti=5

So you can change it as you want.

This feature I reckon soon will be available on Power Query Editor GUI as well, but till that time the above description hopefully helps you in any situation that you want to set a join type.

# Find Mismatch Rows with Power Query in Power BI

Posted by [Reza Rad](#) on Aug 20, 2018

## Merge

Select tables and matching columns to create a merged table.

Customer table from website

CustomerKey	FirstName	MiddleName	LastName	EmailAddress
11000	Jon	V	Yang	jon24@adventure-works.com
11001	Eugene	L	Huang	eugene10@adventure-works.com
11002	Ruben	null	Torres	ruben35@adventure-works.com
11003	Christy	null	Zhu	christy12@adventure-works.com
11004	Elizabeth	null	Johnson	elizabeth5@adventure-works.com

Customer table from application

CustomerKey	FirstName	MiddleName	LastName	BirthDate	MaritalStatus	Suffix	Gender
11000	Jon	V	Yang	8/04/1966	M	null	M
11001	Eugene	L	Huang	14/05/1965	S	null	M
11002	Ruben	null	Torres	12/08/1965	M	null	M
11003	Christy	null	Zhu	15/02/1968	S	null	F
11004	Elizabeth	null	Johnson	8/08/1968	S	null	F

Join Kind

Left Anti (rows only in first)

 The selection has matched 18476 out of the first 18480 rows.

OK

Cancel

Finding rows that are in one table, but not the other is one of the most common scenarios happening in any data related applications. You may have customer records coming from two sources, and want to find data rows that exist in one, but not the other. In Power Query, you can use Merge to combine data tables. Merge can be also used for finding mismatch records. You will learn through this blog post, how in Power Query you can find out which

records are missing with Merge, and then report it in Power BI. To learn more about Power BI, read [Power BI book from Rookie to Rock Star](#).

## Sample Data Tables

In this sample scenario, I have two customer tables; one customer table is coming from the website, and another customer table coming from an application. Here is the customer table from the website:

CustomerKey	FirstName	MiddleName	LastName	EmailAddress
11000	Jon	V	Yang	jon24@adventure-works.com
11001	Eugene	L	Huang	eugene10@adventure-works.com
11002	Ruben		Torres	ruben35@adventure-works.com
11003	Christy		Zhu	christy12@adventure-works.com
11004	Elizabeth		Johnson	elizabeth5@adventure-works.com
11005	Julio		Ruiz	julio1@adventure-works.com
11006	Janet	G	Alvarez	janet9@adventure-works.com
11007	Marco		Mehta	marco14@adventure-works.com
11008	Rob		Verhoff	rob4@adventure-works.com
11013	Ian	M	Jenkins	ian47@adventure-works.com
11014	Sydney		Bennett	sydney23@adventure-works.com
11015	Chloe		Young	chloe23@adventure-works.com
11016	Wyatt	L	Hill	wyatt32@adventure-works.com
11017	Shannon		Wang	shannon1@adventure-works.com

And the other table that comes from the application:

CustomerKey	FirstName	MiddleName	LastName	BirthDate	MaritalStatus	Suffix	Gender
11000	Jon	V	Yang	8/04/1966	M		M
11001	Eugene	L	Huang	14/05/1965	S		M
11002	Ruben		Torres	12/08/1965	M		M
11003	Christy		Zhu	15/02/1968	S		F
11004	Elizabeth		Johnson	8/08/1968	S		F
11005	Julio		Ruiz	5/08/1965	S		M
11006	Janet	G	Alvarez	6/12/1965	S		F
11007	Marco		Mehta	9/05/1964	M		M
11008	Rob		Verhoff	7/07/1964	S		F
11009	Shannon	C	Carlson	1/04/1964	S		M
11010	Jacquelyn	C	Suarez	6/02/1964	S		F
11017	Shannon		Wang	26/06/1944	S		F
11018	Clarence	D	Rai	9/10/1944	S		M
11019	Luke	L	Lal	7/03/1978	S		M
11020	Jordan	C	King	20/09/1978	S		M
11021	Destiny		Wilson	3/09/1978	S		F

The customer table from the website has the email address field, plus name and CustomerKey. The table coming from the application has fields such as birthdate, gender, and marital status, plus name and customer key. The link field in this scenario is CustomerKey which exists in both tables.

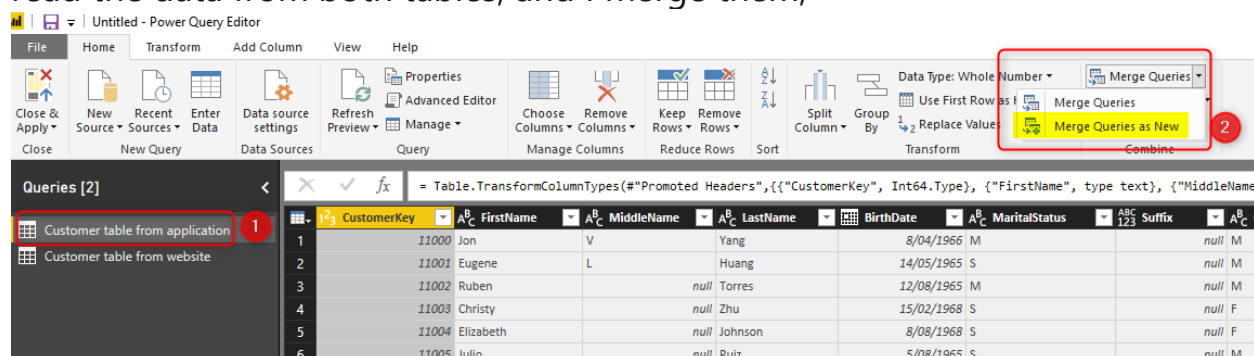
CustomerKey	FirstName	MiddleName	LastName	BirthDate	MaritalStatus	Suffix	Gender
11000	Jon	V	Yang	8/04/1966	M		M
11001	Eugene	L	Huang	14/05/1965	S		M
11002	Ruben		Torres	12/08/1965	M		M
11003	Christy		Zhu	15/02/1968	S		F
11004	Elizabeth		Johnson	8/08/1968	S		F
11005	Julio		Ruiz	5/08/1965	S		M
11006	Janet	G	Alvarez	6/12/1965	S		F
11007	Marco		Mehta	9/05/1964	M		M
11008	Rob		Verhoff	7/07/1964	S		F
11009	Shannon	C	Carlson	2/04/1964	S		M
11010	Jacquelyn	C	Suarez	6/02/1964	S		F
11011	Shannon		Wang	26/06/1944	S		F
11018	Clarence	D	Rai	9/10/1944	S		M
11019	Luke	L	Lal	7/03/1978	S		M
11020	Jordan	C	King	20/09/1978	S		M
11021	Destiny		Wilson	3/09/1978	S		F
11022	Ethan	G	Zhang	12/10/1978	M		M
11023	Seth	M	Edwards	11/10/1978	M		M
11024	Russell		Xie	17/09/1978	M		M
11025	Alejandro		Beck	23/12/1945	M		M
11026	Harold		Sai	3/04/1946	S		M
11027	Jessie	R	Zhao	7/12/1946	M		M
11028	Jill		Jimenez	11/04/1946	M		F
11029	Jimmy	L	Moreno	21/12/1946	M		M
11030	Bethany	G	Yuan	22/02/1947	M		F
11031	Theresa	G	Ramos	22/08/1947	M		F
11032	Denise		Stone	11/06/1947	M		F
11033	Aime		Nath	23/09/1947	M		M
11034	Ebony		Gonzalez	19/06/1947	M		F
11035	Wendy		Dominguez	24/02/1948	M		F
11036	Jennifer	C	Russell	18/12/1978	M		F
11037	Chloe	M	Garcia	27/11/1977	S		F

CustomerKey	FirstName	MiddleName	LastName	EmailAddress
11000	Jon	V	Yang	jon24@adventure-works.com
11001	Eugene	L	Huang	eugene10@adventure-works.com
11002	Ruben		Torres	ruben35@adventure-works.com
11003	Christy		Zhu	christy12@adventure-works.com
11004	Elizabeth		Johnson	elizabeth5@adventure-works.com
11005	Julio		Ruiz	julio1@adventure-works.com
11006	Janet	G	Alvarez	janet9@adventure-works.com
11007	Marco		Mehta	marco14@adventure-works.com
11008	Rob		Verhoff	rob4@adventure-works.com
11013	Ian	M	Jenkins	ian47@adventure-works.com
11014	Sydney		Bennett	sydney23@adventure-works.com
11015	Chloe		Young	chloe23@adventure-works.com
11016	Wyatt	L	Hill	wyatt32@adventure-works.com
11017	Shannon		Wang	shannon1@adventure-works.com
11018	Clarence	D	Rai	clarence32@adventure-works.com
11019	Luke	L	Lal	luke18@adventure-works.com
11020	Jordan	C	King	jordan73@adventure-works.com
11021	Destiny		Wilson	destiny7@adventure-works.com
11022	Ethan	G	Zhang	ethan20@adventure-works.com
11023	Seth	M	Edwards	seth46@adventure-works.com
11024	Russell		Xie	russell17@adventure-works.com
11025	Alejandro		Beck	alejandro45@adventure-works.com
11026	Harold		Sai	harold3@adventure-works.com
11027	Jessie	R	Zhao	jessie16@adventure-works.com
11028	Jill		Jimenez	jill13@adventure-works.com
11029	Jimmy	L	Moreno	jimmy9@adventure-works.com
11030	Bethany	G	Yuan	bethany10@adventure-works.com
11031	Theresa	G	Ramos	theresa13@adventure-works.com

## Merge in Power Query

Previously I have explained [What the Merge is, and what is the difference of that to Append](#). I also explained in another blog post, about [different types of Merge Kind](#). In this example, you would find a couple of those join kinds useful; Left Anti Join, and Right Anti Join. In Below Power Query example, I read the data from both tables, and I merge them;



The screenshot shows the Power Query Editor interface. The 'Queries' pane on the left lists two queries: 'Customer table from application' and 'Customer table from website'. The main area displays a preview of the merged data. The ribbon at the top includes tabs for File, Home, Transform, Add Column, View, and Help. The 'Merge Queries' button is highlighted in the ribbon, and the 'Merge Queries as New' option is selected in the dropdown menu.

## Left Anti Join: Records Only in the First Table

When I merge these two queries, I select Left Anti Join, that gives me rows that exist only in the first table (customer table from application):

Merge

Select tables and matching columns to create a merged table.

1 Customer table from application

2

CustomerKey	FirstName	MiddleName	LastName	BirthDate	MaritalStatus	Suffix	Gender
11000	Jon	V	Yang	8/04/1966	M	null	M
11001	Eugene	L	Huang	14/05/1965	S	null	M
11002	Ruben	null	Torres	12/08/1965	M	null	M
11003	Christy	null	Zhu	15/02/1968	S	null	F
11004	Elizabeth	null	Johnson	8/08/1968	S	null	F

3 Customer table from website

4

CustomerKey	FirstName	MiddleName	LastName	EmailAddress
11000	Jon	V	Yang	jon24@adventure-works.com
11001	Eugene	L	Huang	eugene10@adventure-works.com
11002	Ruben	null	Torres	ruben35@adventure-works.com
11003	Christy	null	Zhu	christy12@adventure-works.com
11004	Elizabeth	null	Johnson	elizabeth5@adventure-works.com

5 Join Kind

Left Anti (rows only in first)

6 The selection has matched 18476 out of the first 18478 rows.

OK Cancel

This will give you only rows that exist in the first table “Customer table from application.”

	CustomerKey	FirstName	MiddleName	LastName	BirthDate	MaritalStatus	Suffix	Gender	Customer table from website
1	11009	Shannon	C	Carlson	1/04/1964	S	null	M	Table
2	11010	Jacquelyn	C	Suarez	6/02/1964	S	null	F	Table

Then you can remove the last column in the output because the table value would not have any rows (this are mismatch rows)

	CustomerKey	FirstName	MiddleName	LastName	BirthDate	MaritalStatus	Suffix	Gender
1	11009	Shannon	C	Carlson	1/04/1964	S	null	M
2	11010	Jacquelyn	C	Suarez	6/02/1964	S	null	F


## Right Anti Join: Records Only in the Second Table

The same Approach can be used for rows that exist only in the second table, using the Right Anti Join

### Merge

Select tables and matching columns to create a merged table.

Customer table from application



CustomerKey	FirstName	MiddleName	LastName	BirthDate	MaritalStatus	Suffix	Gender
11000	Jon	V	Yang	8/04/1966	M	null	M
11001	Eugene	L	Huang	14/05/1965	S	null	M
11002	Ruben	null	Torres	12/08/1965	M	null	M
11003	Christy	null	Zhu	15/02/1968	S	null	F
11004	Elizabeth	null	Johnson	8/08/1968	S	null	F

Customer table from website

CustomerKey	FirstName	MiddleName	LastName	EmailAddress
11000	Jon	V	Yang	jon24@adventure-works.com
11001	Eugene	L	Huang	eugene10@adventure-works.com
11002	Ruben	null	Torres	ruben35@adventure-works.com
11003	Christy	null	Zhu	christy12@adventure-works.com
11004	Elizabeth	null	Johnson	elizabeth5@adventure-works.com

Join Kind

Right Anti (rows only in second)

OK

Cancel

But right Anti Join will give you a result which looks a bit weird if you do not expand the table;

1	CustomerKey	FirstName	MiddleName	LastName	BirthDate	MaritalStatus	Suffix	Gender	Customer table from website
	null	null	null	null	null	null	null	null	Table

You DO NEED to expand the table for the second query to get mismatch rows when you use RIGHT ANIT Join. Which is an extra step, but still works fine. You can remove all columns from the first table, and expand the last column;

	CustomerKey	FirstName	MiddleName	LastName	EmailAddress
1	11013	Ian	M	Jenkins	ian47@adventure-works.com
2	11014	Sydney		null Bennett	sydney23@adventure-works.com
3	11015	Chloe		null Young	chloe23@adventure-works.com
4	11016	Wyatt	L	Hill	wyatt32@adventure-works.com

## Left Anti with Changing Order of Tables; Works similar to Right Anti

Or you can switch the order of tables at the time of Merge, and use Left Anti join to get the same output;

### Merge

Select tables and matching columns to create a merged table.

Customer table from website

CustomerKey	FirstName	MiddleName	LastName	EmailAddress
11000	Jon	V	Yang	jon24@adventure-works.com
11001	Eugene	L	Huang	eugene10@adventure-works.com
11002	Ruben	null	Torres	ruben35@adventure-works.com
11003	Christy	null	Zhu	christy12@adventure-works.com
11004	Elizabeth	null	Johnson	elizabeth5@adventure-works.com

Customer table from application

CustomerKey	FirstName	MiddleName	LastName	BirthDate	MaritalStatus	Suffix	Gender
11000	Jon	V	Yang	8/04/1966	M	null	M
11001	Eugene	L	Huang	14/05/1965	S	null	M
11002	Ruben	null	Torres	12/08/1965	M	null	M
11003	Christy	null	Zhu	15/02/1968	S	null	F
11004	Elizabeth	null	Johnson	8/08/1968	S	null	F

Join Kind

Left Anti (rows only in first)

The selection has matched 18476 out of the first 18480 rows.

OK

Cancel

The output of this is similar to the step we have done before (it gives you the records only in the second table "customers from the website"):

	CustomerKey	FirstName	MiddleName	LastName	EmailAddress	Customer table from application
1	11013	Ian	M	Jenkins	ian47@adventure-works.com	Table
2	11014	Sydney	null	Bennett	sydney23@adventure-works.com	Table
3	11015	Chloe	null	Young	chloe23@adventure-works.com	Table
4	11016	Wyatt	L	Hill	wyatt32@adventure-works.com	Table

## Summary


In summary, this post focused on two of the least common join types in Power Query; **Left Anti** join, and **Right Anti** join. These two join types are very useful when you want to find records that exist in one of the tables, but not the other one. All you do need is to select the right order of tables and merge type in the Merge command graphical interface. If you like [to know more about other types of joins, read this post](#).



# Dates Between Merge Join in Power Query

Posted by [Reza Rad](#) on Aug 15, 2017

## Dates Between Merge with Matching Grains



1 <sup>2</sup>	CustomerID	A <sup>B</sup> <sub>C</sub> Name	A <sup>B</sup> <sub>C</sub> City	FromDate	ToDate
1	1	Reza	Auckland	1/01/2017	31/12/2017
2	2	John	Sydney	1/01/2017	19/02/2017
3	2	John	Melbourne	20/02/2017	31/12/2017

1 <sup>2</sup>	ID	CustomerID	ProductID	SalesAmount	Date
1	1	1	234	100	15/01/2017
2	2	2	123	20	20/01/2017
3	3	1	4	120	14/03/2017
4	4	2	23	60	25/04/2017

1 <sup>2</sup>	CustomerID	A <sup>B</sup> <sub>C</sub> Name	A <sup>B</sup> <sub>C</sub> City	1 <sup>2</sup> 123	Dates
407	2	John	Sydney		11/02/2017
408	2	John	Sydney		12/02/2017
409	2	John	Sydney		13/02/2017
410	2	John	Sydney		14/02/2017
411	2	John	Sydney		15/02/2017
412	2	John	Sydney		16/02/2017
413	2	John	Sydney		17/02/2017
414	2	John	Sydney		18/02/2017
415	2	John	Sydney		19/02/2017
416	2	John	Melbourne		20/02/2017
417	2	John	Melbourne		21/02/2017
418	2	John	Melbourne		22/02/2017
419	2	John	Melbourne		23/02/2017
420	2	John	Melbourne		24/02/2017
421	2	John	Melbourne		25/02/2017
422	2	John	Melbourne		26/02/2017

Using Merge in Power Query gives you the ability to join on an EQUAL join with one or more fields between two tables. However, in some situations, you need to do the Merge Join not based on equality of values, based on other comparison options. One of the very common use cases is to Merge Join two queries based on dates between. In this example, I am going to show you how to use Merge Join to merge based on dates between. If you want to learn more about joining tables in Power Query read [this blog post](#). To learn more about Power BI, read [Power BI book from Rookie to Rock Star](#).

### Download Sample Data Set

Download the data set and sample from here:

Enter Your Email to download the file (required)

Download

## Problem Definition

There are some situations that you need to join two tables based on dates between the not exact match of two dates. For example; consider scenario below:

There are two tables; Sales table includes sales transactions by Customer, Product, and Date. And Customer table has detailed information about customer including ID, Name, and City. Here is a screenshot of Sales Table:

	1 <sup>2</sup> <sub>3</sub> ID	1 <sup>2</sup> <sub>3</sub> CustomerID	1 <sup>2</sup> <sub>3</sub> ProductID	1 <sup>2</sup> <sub>3</sub> SalesAmount	1 <sup>2</sup> <sub>3</sub> Date
1	1	1	234	100	15/01/2017
2	2	2	123	20	20/01/2017
3	3	1	4	120	14/03/2017
4	4	2	23	60	25/04/2017

Customer's table has the history details of changes through the time. For example, the customer ID 2, has a track of change. John was living in Sydney for some time, then moved to Melbourne after that.

	1 <sup>2</sup> <sub>3</sub> CustomerID	A <sup>B</sup> <sub>C</sub> Name	A <sup>B</sup> <sub>C</sub> City	1 <sup>2</sup> <sub>3</sub> FromDate	1 <sup>2</sup> <sub>3</sub> ToDate
1	1	Reza	Auckland	1/01/2017	31/12/2017
2	2	John	Sydney	1/01/2017	19/02/2017
3	2	John	Melbourne	20/02/2017	31/12/2017

The problem we are trying to solve is to join these two tables based on their customer ID and find out the City related to that for that specific period. We have to check the Date field from Sales Table to fit into FromDate and ToDate of the Customer table.

## Grain Matching

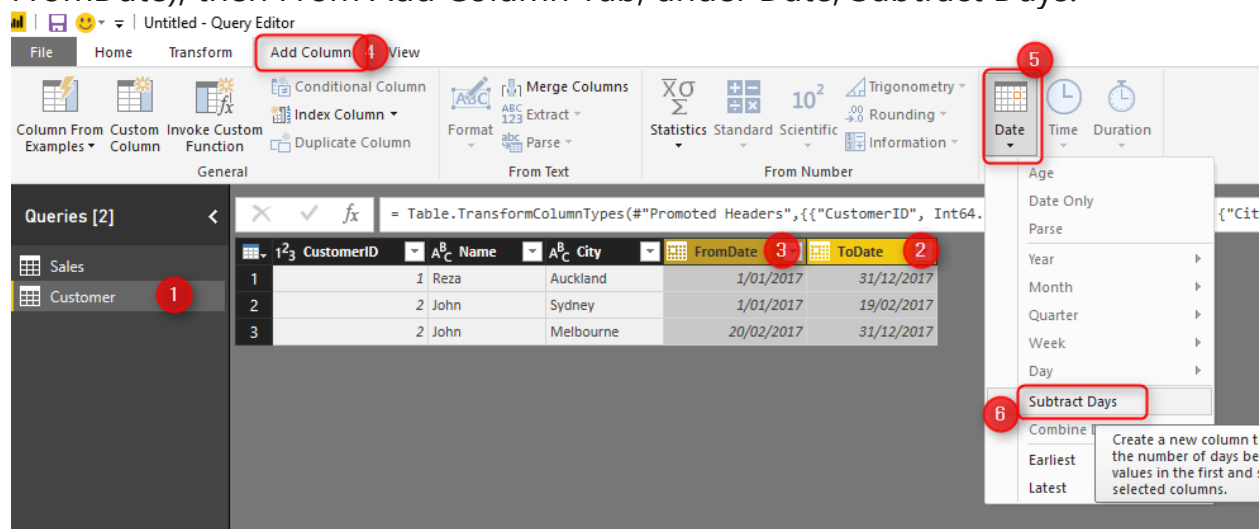
One of the easiest ways of matching two tables is to bring them both to the same grain. In this example, the Sales Table is at the grain of Customer, Product, and Date. However, the Customer table is at the grain of Customer and a change in properties such as City. We can change the grain of the

customer table to be on Customer and Date. That means Having one record per every customer and every day.

Before applying this change, there is a little warning I would like to explain; with changing grain of a table to more detailed grain, the number of rows for that table will increase significantly. It is fine to do it as an intermediate change, but if you want to make this change as a final query to be loaded in Power BI, then you need to think about your approach more carefully.

## Step 1: Calculating Duration

The first step in this approach is to find out how many days is the duration between FromDate and ToDate in the customer table for each row. That simply can be calculated with selecting two columns (First ToDate, then FromDate), then From Add Column Tab, under Date, Subtract Days.



The screenshot shows the Power Query Editor interface. The 'Add Column' tab is selected in the ribbon. The 'Date' dropdown menu is open, and the 'Subtract Days' option is highlighted. The 'Customer' table is selected in the left pane. The data table shows columns: CustomerID, Name, City, FromDate, and ToDate. The formula bar shows: `= Table.TransformColumnTypes(#"Promoted Headers",{{"CustomerID", Int64}}`

Then you will see the new column added which is the duration between From and To dates

	CustomerID	Name	City	FromDate	ToDate	DateDifference
1	1	Reza	Auckland	1/01/2017	31/12/2017	364
2	2	John	Sydney	1/01/2017	19/02/2017	49
3	2	John	Melbourne	20/02/2017	31/12/2017	314

## Step 2: Creating List of Dates

The second step is to create a list of dates for every record, starting from FromDate, adding one day at a time, for the number of occurrence in DateDifference column.

There is a generator that you can easily use to create a list of dates. List. Dates are a Power Query function which will generate a list of dates. Here is the syntax for this table;

*1 List.Dates(<start date>,<occurrence>,<duration>)*

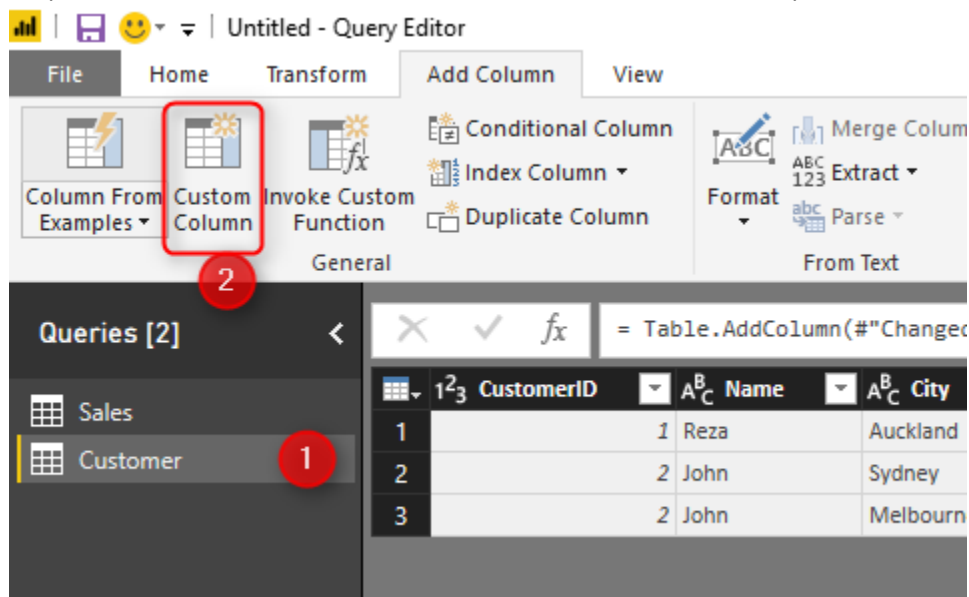
- start date in this scenario will come from FromDate column
- Occurrence would come from DateDifference plus one.

Duration should be in a day Level. Duration has 4 input arguments:

*1 #duration(<day>,<hour>,<minute>,<second>)*

a daily duration would be: #duration(1,0,0,0)

So, we need to add a custom column to our table;



The screenshot shows the Power Query Editor interface. The 'Add Column' tab is active, and the 'Custom Column' button is highlighted with a red box and a red circle with the number 2. The 'Queries' pane on the left shows the 'Customer' table selected, indicated by a red circle with the number 1. The formula bar at the top shows the expression: `= Table.AddColumn(#"Changed", "Dates", each List.Dates([FromDate], [DateDifference] + 1, #duration(1, 0, 0, 0)))`. Below the formula bar, a preview of the data table is shown with columns: CustomerID, Name, and City. The data rows are:

CustomerID	Name	City
1	Reza	Auckland
2	John	Sydney
3	John	Melbourn

The custom column expression can be as below;

*1 List.Dates([FromDate],[DateDifference]+1,#duration(1,0,0,0))*

I named this column as Dates.

Here is the result:

fx = Table.AddColumn(#"Inserted Date Subtraction", "Dates", each List.Dates([FromDate],[DateDifference]+1,#duration(1,0,0,0)))

	CustomerID	Name	City	FromDate	ToDate	DateDifference	Dates
1	1	Reza	Auckland	1/01/2017	31/12/2017	364	List
2	2	John	Sydney	1/01/2017	19/02/2017	49	List
3	2	John	Melbourne	20/02/2017	31/12/2017	314	List

List

20/02/2017
21/02/2017
22/02/2017
23/02/2017
24/02/2017
25/02/2017
26/02/2017
27/02/2017
28/02/2017
1/03/2017
2/03/2017
3/03/2017
4/03/2017
5/03/2017
6/03/2017
7/03/2017
8/03/2017
9/03/2017
10/03/2017
11/03/2017

The Dates column now has a list in every row. This list is a list of dates. Next step is to expand it.

### Step 3: Expand List to Day Level

The last step to change the grain of this table, is to expand the Dates column. To expand, click on the Expand button.

fx = Table.AddColumn(#"Inserted Date Subtraction", "Dates", each List.Dates([FromDate],[DateDifference],#duration(1,0,0,0)))

	CustomerID	Name	City	FromDate	ToDate	DateDifference	Dates
1	1	Reza	Auckland	1/01/2017	31/12/2017		List
2	2	John	Sydney	1/01/2017	19/02/2017		List
3	2	John	Melbourne	20/02/2017	31/12/2017		List

Expand to New Rows  
Extract Values...

Expanding to new rows will give you a data set with all dates;

	1 <sup>2</sup> 3 CustomerID	A <sup>B</sup> <sub>C</sub> Name	A <sup>B</sup> <sub>C</sub> City	FromDate	ToDate	1 <sup>2</sup> 3 DateDifference	ABC 123 Dates
1	1	Reza	Auckland	1/01/2017	31/12/2017	364	1/01/2017
2	1	Reza	Auckland	1/01/2017	31/12/2017	364	2/01/2017
3	1	Reza	Auckland	1/01/2017	31/12/2017	364	3/01/2017
4	1	Reza	Auckland	1/01/2017	31/12/2017	364	4/01/2017
5	1	Reza	Auckland	1/01/2017	31/12/2017	364	5/01/2017
6	1	Reza	Auckland	1/01/2017	31/12/2017	364	6/01/2017
7	1	Reza	Auckland	1/01/2017	31/12/2017	364	7/01/2017
8	1	Reza	Auckland	1/01/2017	31/12/2017	364	8/01/2017
9	1	Reza	Auckland	1/01/2017	31/12/2017	364	9/01/2017
10	1	Reza	Auckland	1/01/2017	31/12/2017	364	10/01/2017
11	1	Reza	Auckland	1/01/2017	31/12/2017	364	11/01/2017

Now you can remove FromDate, ToDate, and DateDifference. We don't need these three columns anymore.

	1 <sup>2</sup> 3 CustomerID	A <sup>B</sup> <sub>C</sub> Name	A <sup>B</sup> <sub>C</sub> City	ABC 123 Dates
407	2	John	Sydney	11/02/2017
408	2	John	Sydney	12/02/2017
409	2	John	Sydney	13/02/2017
410	2	John	Sydney	14/02/2017
411	2	John	Sydney	15/02/2017
412	2	John	Sydney	16/02/2017
413	2	John	Sydney	17/02/2017
414	2	John	Sydney	18/02/2017
415	2	John	Sydney	19/02/2017
416	2	John	Melbourne	20/02/2017
417	2	John	Melbourne	21/02/2017
418	2	John	Melbourne	22/02/2017
419	2	John	Melbourne	23/02/2017
420	2	John	Melbourne	24/02/2017
421	2	John	Melbourne	25/02/2017
422	2	John	Melbourne	26/02/2017

The table above is the same customer table but on different grain. We can now easily see on which dates John was in Sydney, and which dates in Melbourne. This table now can be easily merged with the sales table.


## Merging Tables on the Same Grain

When both tables are at the same grain, then you can easily merge them.



## Merge


Select tables and matching columns to create a merged table.

Sales


ID	CustomerID 1	ProductID	SalesAmount	Date 2
1	1	234	100	15/01/2017
2	2	123	20	20/01/2017
3	1	4	120	14/03/2017
4	2	23	60	25/04/2017

12

Customer




CustomerID 1	Name	City	Dates 2
1	Reza	Auckland	1/01/2017
1	Reza	Auckland	2/01/2017
1	Reza	Auckland	3/01/2017
1	Reza	Auckland	4/01/2017
1	Reza	Auckland	5/01/2017

12

Join Kind

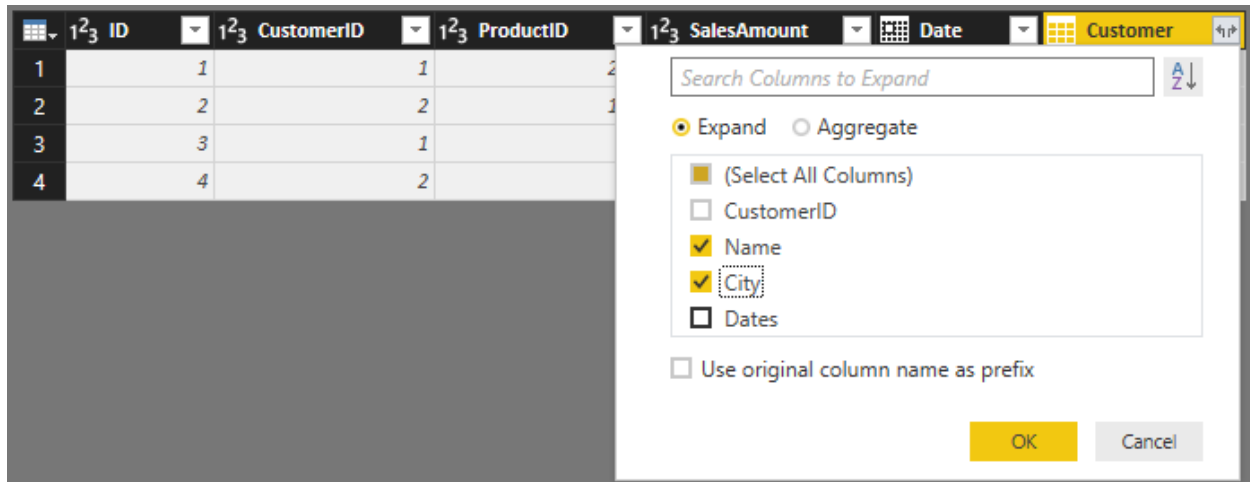
Left Outer (all from first, matching from second)

 The selection has matched 4 out of the first 4 rows.

OK

Cancel

Merge should be between two tables, based on CustomerID and Dates. You need to hold Ctrl key to select more than one column. And make sure you select them in the same order in both tables. After merge then you can expand and only select City and Name from the other table;

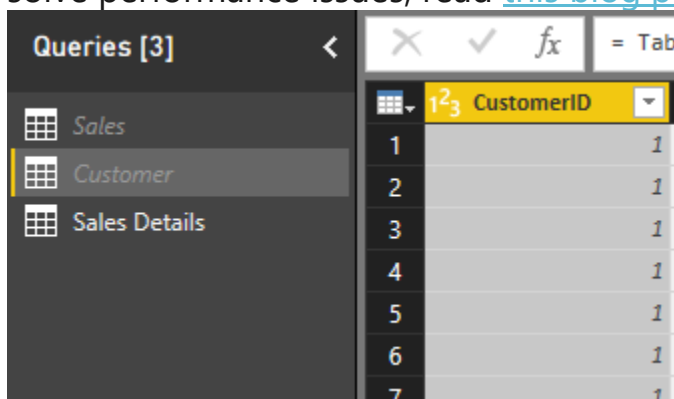


The final result shows that two sales transactions for John happened at two different times that John has been in two different cities of Sydney and Melbourne.

ID	CustomerID	ProductID	SalesAmount	Date	Name	City
1	1	234	100	15/01/2017	Reza	Auckland
2	3	4	120	14/03/2017	Reza	Auckland
3	2	123	20	20/01/2017	John	Sydney
4	4	23	60	25/04/2017	John	Melbourne

## Final Step: Cleansing

You won't need the first two tables after merging them, you can disable their load to avoid extra memory consumption (especially for Customer table which should be big after grain change). To learn more about Enable Load and to solve performance issues, read [this blog post](#).





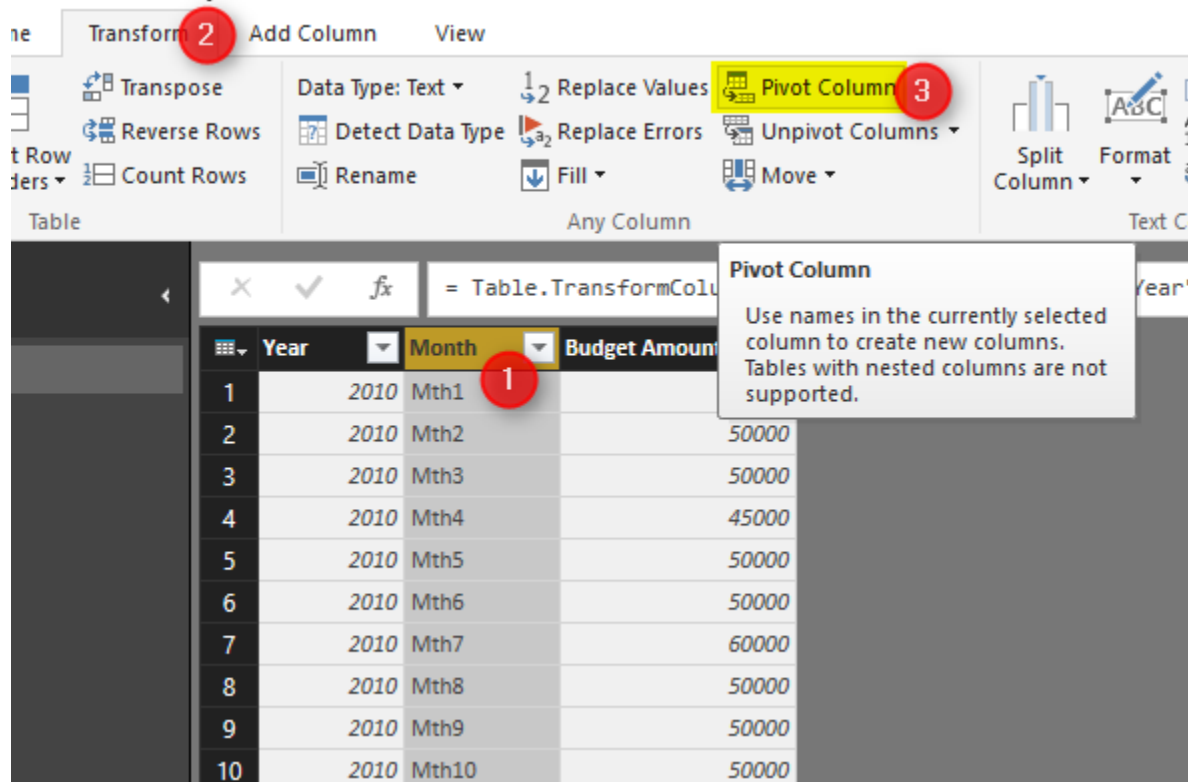
## Summary

There are multiple ways of joining two tables based on the non-equality comparison. Matching grain is one of them and works perfectly fine, and easy to implement. In this post, you've learned how to use grain matching to do this joining and get the join result based on dates between comparison. With this method, be careful to disable the load of the table that you've changed the grain for it to avoid performance issues afterward.

# Pivot and Unpivot with Power BI

Posted by [Reza Rad](#) on Apr 7, 2016

Untitled - Query Editor



The screenshot shows the Power Query Editor interface. The 'Transform' tab is active, and the 'Pivot Column' option is highlighted with a red circle labeled '3'. A table with the following data is displayed:

	Year	Month	Budget Amount
1	2010	Mth1	
2	2010	Mth2	50000
3	2010	Mth3	50000
4	2010	Mth4	45000
5	2010	Mth5	50000
6	2010	Mth6	50000
7	2010	Mth7	60000
8	2010	Mth8	50000
9	2010	Mth9	50000
10	2010	Mth10	50000

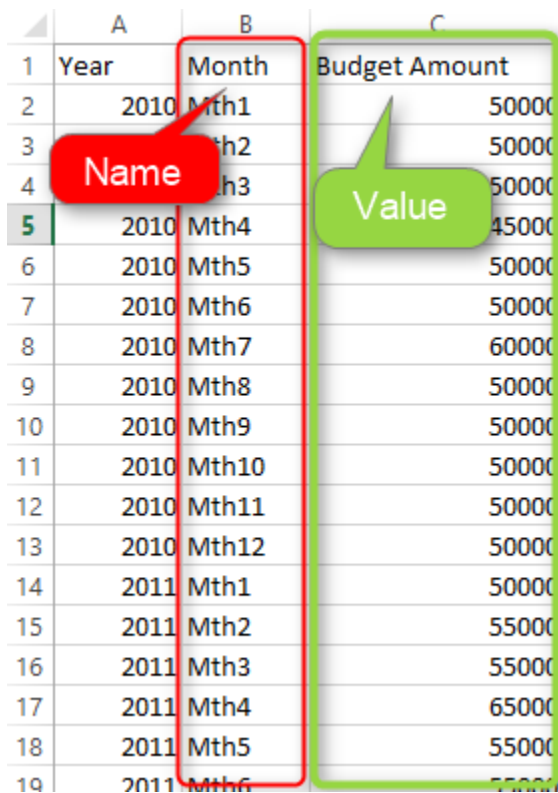
A tooltip for 'Pivot Column' is displayed, stating: 'Use names in the currently selected column to create new columns. Tables with nested columns are not supported.'

Turning columns to rows or rows to columns is easy with Power Query and Power BI. There are many situations that you get a name, value data source, and wants to convert that into columns with values underneath.

On the other hand, many times you get multiple columns and want to change it to name, value structure with a column for name, another column for value. That's why Pivot and Unpivot are for. In this post, I'll get you through the basic pivot and unpivot. If you want to read more about Power Query, read it from [Power BI online book](#).

## Pivot: Turning Name, Value Rows to Columns

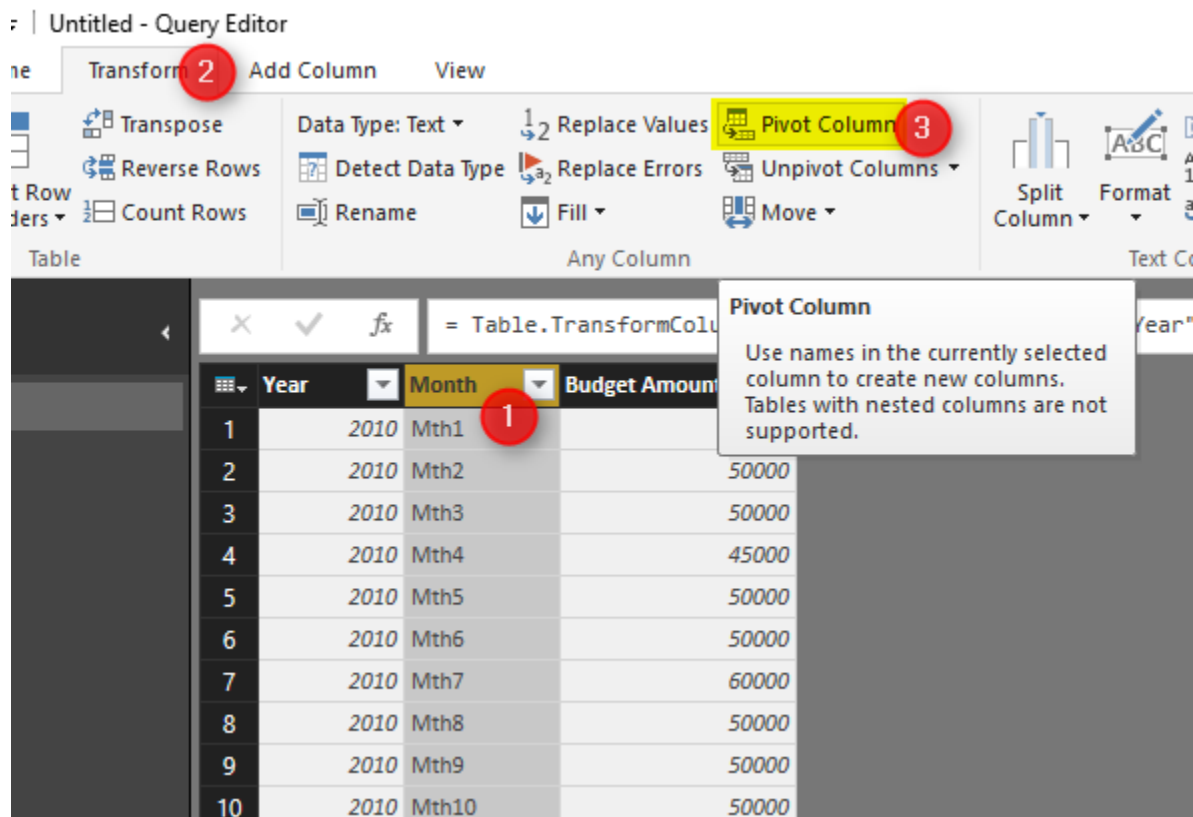
Consider we have a data source like this:



The image shows an Excel table with three columns: Year, Month, and Budget Amount. The 'Month' column is highlighted with a red box and labeled 'Name' with a red callout. The 'Budget Amount' column is highlighted with a green box and labeled 'Value' with a green callout. The table contains data for the years 2010 and 2011, with months Mth1 through Mth12 for 2010 and Mth1 through Mth6 for 2011.

	A	B	C
1	Year	Month	Budget Amount
2	2010	Mth1	50000
3		Mth2	50000
4		Mth3	50000
5	2010	Mth4	45000
6	2010	Mth5	50000
7	2010	Mth6	50000
8	2010	Mth7	60000
9	2010	Mth8	50000
10	2010	Mth9	50000
11	2010	Mth10	50000
12	2010	Mth11	50000
13	2010	Mth12	50000
14	2011	Mth1	50000
15	2011	Mth2	55000
16	2011	Mth3	55000
17	2011	Mth4	65000
18	2011	Mth5	55000
19	2011	Mth6	55000

above data set is budget information. If we want to spread the table with a column for every month, we can simply use Pivot as below:  
first click on the column that contains *names*, in this example it would be Month column. Then from Transform menu tab, choose Pivot Column.



The Pivot dialog box asks you to choose the *Value* column, which is Budget Amount in this example

## Pivot Column

Use the names in column "Month" to create new columns.

Values Column ⓘ

Budget Amount

Advanced options

[Learn more about Pivot Column](#)

OK

Cancel

and then simply and easily I have the pivoted result set;

= Table.Pivot(#"Changed Type", List.Distinct(#"Changed Type"[Month]), "Month", "Budget Amount", List.Sum)													
Year	Mth1	Mth2	Mth3	Mth4	Mth5	Mth6	Mth7	Mth8	Mth9	Mth10	Mth11	Mth12	
2010	50000	50000	50000	45000	50000	50000	60000	50000	50000	50000	50000	50000	50000
2011	50000	55000	55000	65000	55000	55000	55000	55000	55000	55000	55000	55000	55000
2012	65000	65000	65000	65000	80000	65000	65000	65000	65000	65000	65000	65000	65000
2013	70000	70000	70000	70000	65000	70000	70000	70000	70000	80000	70000	70000	70000

You can see that I have a column for each month now. Year column was just passed through. I can have as many as columns I want to pass through. The important factor for Pivot is that there should be a name column, and a value column. You can also see the Table.Pivot script of Power Query generated for this example in above screenshot.

Now let's see what happens if name value is a bit different;

Year	Month	Budget Amount
2010	Mth1	50000
2010	Mth2	50000
2010	Mth3	50000
2010	Mth4	45000
2010	Mth5	50000
2010	Mth6	50000
2010	Mth7	60000
2010	Mth8	50000
2010	Mth9	50000
2010	Mth10	50000
2010	Mth11	50000
2010	Mth11	50000
2010	Mth12	50000
2011	Mth1	50000
2011	Mth2	55000
2011	Mth3	55000
2011	Mth5	55000
2011	Mth6	55000
2011	Mth7	55000
2011	Mth8	55000

In this example, I have two records for a single name (Mth 11 2010 is repeated). and for some names, I don't have any records at all (Mth 4 2011 is missing)

Pivot dialog has the option to choose the aggregation function, and that is especially for cases that a name appeared more than once in the data set.

Default aggregation is Sum,

## Pivot Column

Use the names in column "Month" to create new columns.

Values Column ⓘ

Budget Amount

Advanced options

Aggregate Value Function ⓘ

Sum

[Learn more about Pivot Column](#)

OK

Cancel

So the default Pivot will result as below:

	Year	Mth1	Mth2	Mth3	Mth4	Mth5	Mth6	Mth7	Mth8	Mth9	Mth10	Mth11	Mth12
1	2010	50000	50000	50000	45000	50000	50000	60000	50000	50000	50000	100000	50000
2	2011	50000	55000	55000	null	55000	55000	55000	55000	55000	55000	55000	55000
3	2012	65000	65000	65000	65000	80000	65000	65000	65000	65000	65000	65000	65000
4	2013	70000	70000	70000	70000	65000	70000	70000	70000	80000	70000	70000	70000

However, if the aggregation is set to Do Not Aggregate. then you will get an error when a name is repeated in the data set

## Pivot Column

Use the names in column "Month" to create new columns.

Values Column ⓘ

Budget Amount

Advanced options

Aggregate Value Function ⓘ

Don't Aggregate

[Learn more about Pivot Column](#)

OK

Cancel

Here is the error value in the result set;

fx = Table.Pivot(#"Changed Type", List.Distinct(#"Changed Type"[Month]), "Month", "Budget Amount")

	Year	Mth1	Mth2	Mth3	Mth4	Mth5	Mth6	Mth7	Mth8	Mth9	Mth10	Mth11	Mth12
1	2010	50000	50000	50000	45000	50000	50000	60000	50000	50000	50000	50000	50000
2	2011	50000	55000	55000	null	55000	55000	55000	55000	55000	55000	55000	55000
3	2012	65000	65000	65000	65000	80000	65000	65000	65000	65000	65000	65000	65000
4	2013	70000	70000	70000	70000	65000	70000	70000	70000	80000	70000	70000	70000

and the error would be:

*Expression.Error: There were too many elements in the enumeration to complete the operation.*

*Details:*

*List*

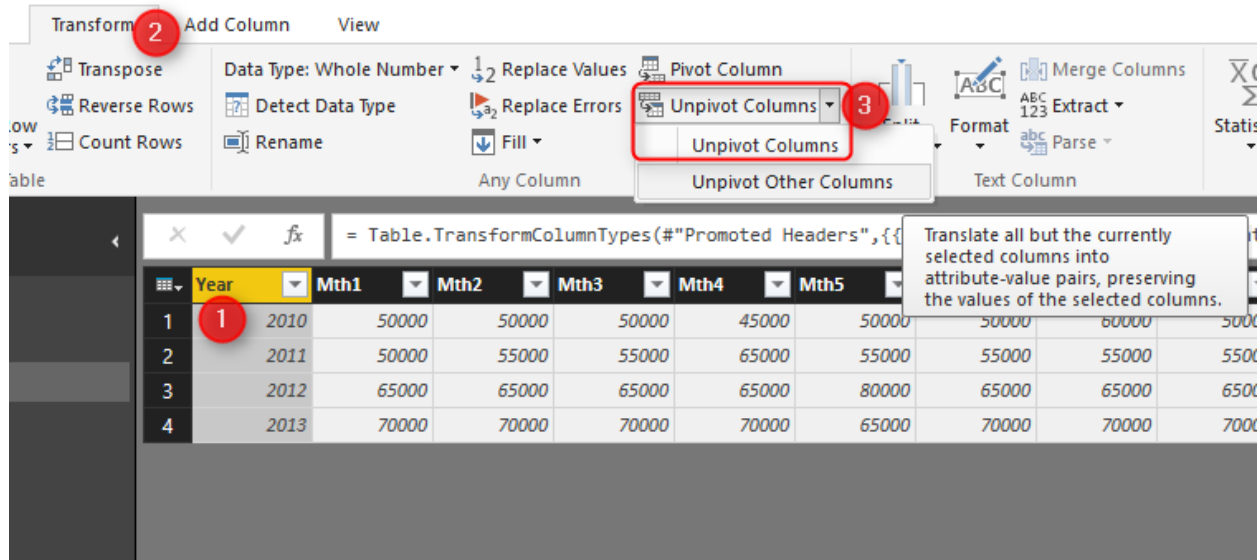
So Pivot is easy and simple to do, but you have to be careful about the nature and quality of the source data set. If it is normal to have a name repeated in the source data, then an aggregation needs to be set properly. If you expect each name to appear once, then setting it as Do Not Aggregate works better because you can use error handling mechanism in Power Query to handle error somehow.

## Unpivot; Turning Columns to Rows; Name, Values

Unpivot does the reverse. It turns multiple column headers into a single column but in rows. And store their values in another column. Here is an example of Budget data that usually you get from the finance department;

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Year	Mth1	Mth2	Mth3	Mth4	Mth5	Mth6	Mth7	Mth8	Mth9	Mth10	Mth11	Mth12
2	2010	50000	50000	50000	45000	50000	50000	60000	50000	50000	50000	50000	50000
3	2011	50000	55000	55000	65000	55000	55000	55000	55000	55000	55000	55000	55000
4	2012	65000	65000	65000	65000	80000	65000	65000	65000	65000	65000	65000	65000
5	2013	70000	70000	70000	70000	65000	70000	70000	70000	80000	70000	70000	70000

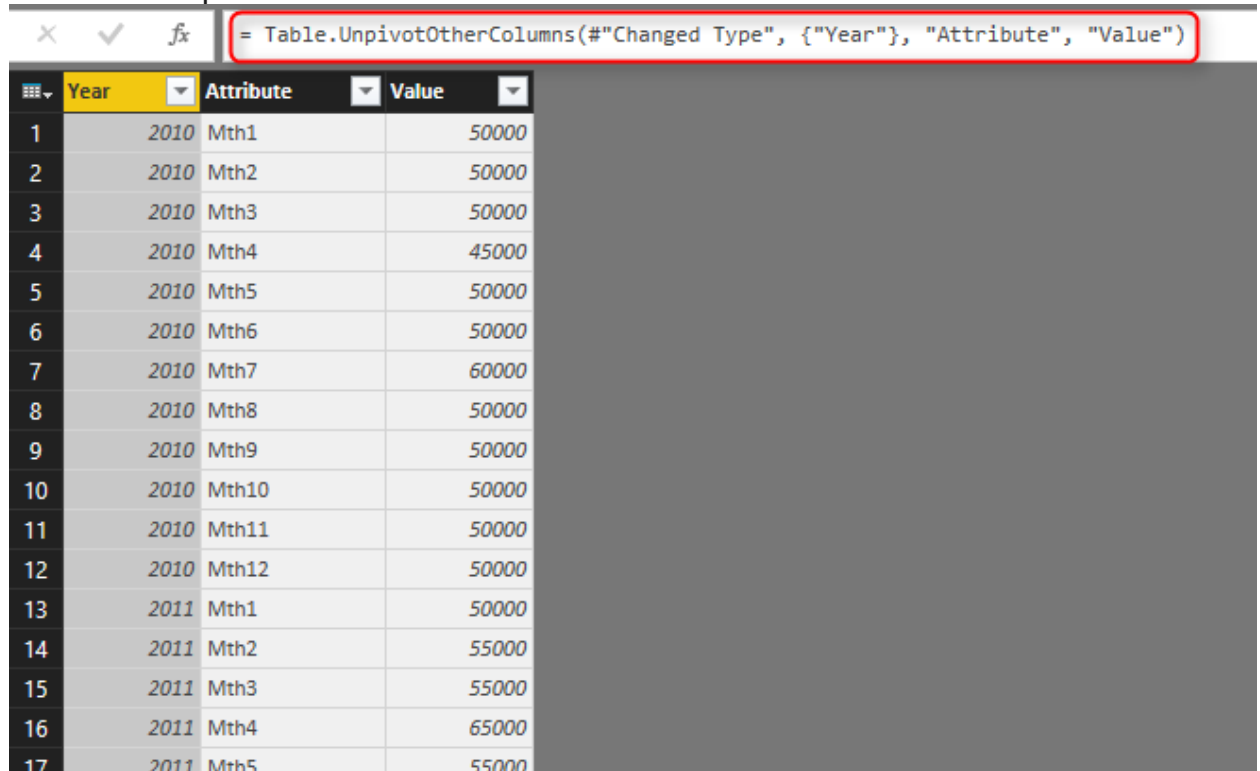
You can click on Columns that you want to unpivot, and then select Unpivot columns (or you can do the reverse, select pass through columns, and select unpivot other columns);



The screenshot shows the Power Query Transform ribbon. The 'Unpivot Columns' button is highlighted with a red circle and a tooltip that reads: "Translate all but the currently selected columns into attribute-value pairs, preserving the values of the selected columns." The data table below shows columns Year, Mth1, Mth2, Mth3, Mth4, and Mth5.

	Year	Mth1	Mth2	Mth3	Mth4	Mth5
1	2010	50000	50000	50000	45000	50000
2	2011	50000	55000	55000	65000	55000
3	2012	65000	65000	65000	80000	65000
4	2013	70000	70000	70000	70000	65000

and then unpivoted result set would be as below:



The screenshot shows the Power Query Transform ribbon. The 'Unpivot Other Columns' button is highlighted with a red circle. The data table below shows columns Year, Attribute, and Value.

	Year	Attribute	Value
1	2010	Mth1	50000
2	2010	Mth2	50000
3	2010	Mth3	50000
4	2010	Mth4	45000
5	2010	Mth5	50000
6	2010	Mth6	50000
7	2010	Mth7	60000
8	2010	Mth8	50000
9	2010	Mth9	50000
10	2010	Mth10	50000
11	2010	Mth11	50000
12	2010	Mth12	50000
13	2011	Mth1	50000
14	2011	Mth2	55000
15	2011	Mth3	55000
16	2011	Mth4	65000
17	2011	Mth5	55000

As you see columns and their values are now converted to rows split into only two columns: attribute, and value.

If you get a repetitive column in the source data like below;



A	B	C	D	E	F	G	H	I	J	K	L	M	N
Year	Mth1	Mth2	Mth3	Mth4	Mth5	Mth6	Mth7	Mth8	Mth8	Mth9	Mth10	Mth11	Mth12
2010	50000	50000	50000	45000	50000	50000	60000	50000	50000	50000	50000	50000	50000
2010	50000	50000	50000	55000	50000	50000	60000	50000	50000	50000	50000	50000	50000
2011	50000	55000	55000	65000	55000	55000	55000	55000	55000	55000	55000	55000	55000
2012	65000	65000	65000	65000	80000	65000	65000	65000	65000	65000	65000	65000	65000
2013	70000	70000	70000	70000	65000	70000	70000	70000	70000	80000	70000	70000	70000

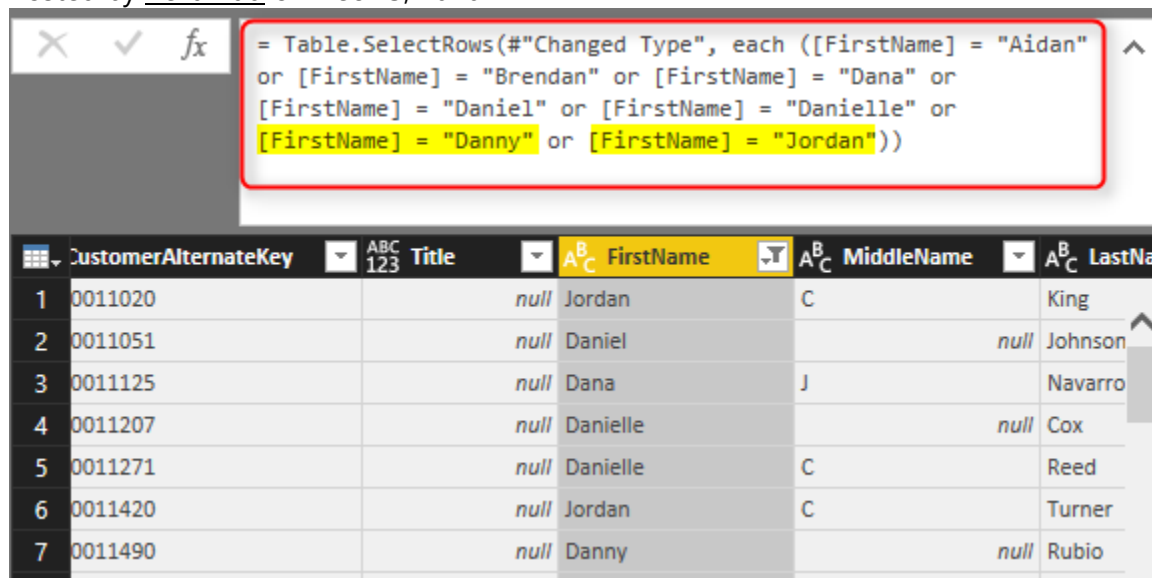
Then you would get that repeated in the attribute field after unpivot;

	Year	Attribute	Value
1	2010	Mth1	50000
2	2010	Mth2	50000
3	2010	Mth3	50000
4	2010	Mth4	45000
5	2010	Mth5	50000
6	2010	Mth6	50000
7	2010	Mth7	60000
8	2010	Mth8	50000
9	2010	Mth8_1	50000
10	2010	Mth9	50000
11	2010	Mth10	50000
12	2010	Mth11	50000
13	2010	Mth12	50000

So the best way to handle that is to identify the repetitive column before applying unpivot. You can do these types of checking with Power Query scripts and other functions, If you want to read more about Power Query read it from [Power BI online book](#).

# Warning! Misleading Power Query Filtering

Posted by [Reza Rad](#) on Dec 15, 2016



The screenshot shows the Power Query Editor. At the top, the M formula bar contains the following code:

```
= Table.SelectRows("#Changed Type", each ([FirstName] = "Aidan" or [FirstName] = "Brendan" or [FirstName] = "Dana" or [FirstName] = "Daniel" or [FirstName] = "Danielle" or [FirstName] = "Danny" or [FirstName] = "Jordan"))
```

Below the formula bar, a table of customer data is displayed. The table has columns: CustomerAlternateKey, Title, FirstName, MiddleName, and LastName. The data is as follows:

	CustomerAlternateKey	Title	FirstName	MiddleName	LastName
1	0011020	null	Jordan	C	King
2	0011051	null	Daniel	null	Johnson
3	0011125	null	Dana	J	Navarro
4	0011207	null	Danielle	null	Cox
5	0011271	null	Danielle	C	Reed
6	0011420	null	Jordan	C	Turner
7	0011490	null	Danny	null	Rubio

Filtering in Power Query component of Power BI is easy. However it can be misleading very easily as well. I have seen the usage of this filtering inappropriately in many cases. Many people simply believe in what they see, rather than seeing behind the scenes. Power Query filtering is totally different when you do basic filtering or advanced filtering, and the result of filtering will be different too. In this post, I'll show you through a very simple example of how misleading it can be and what is the correct way to do filtering in Power Query. This post is a must read for everyone who uses filtering in Power Query. If you like to learn more about Power BI, read the [Power BI online book from Rookie to Rock Star](#).

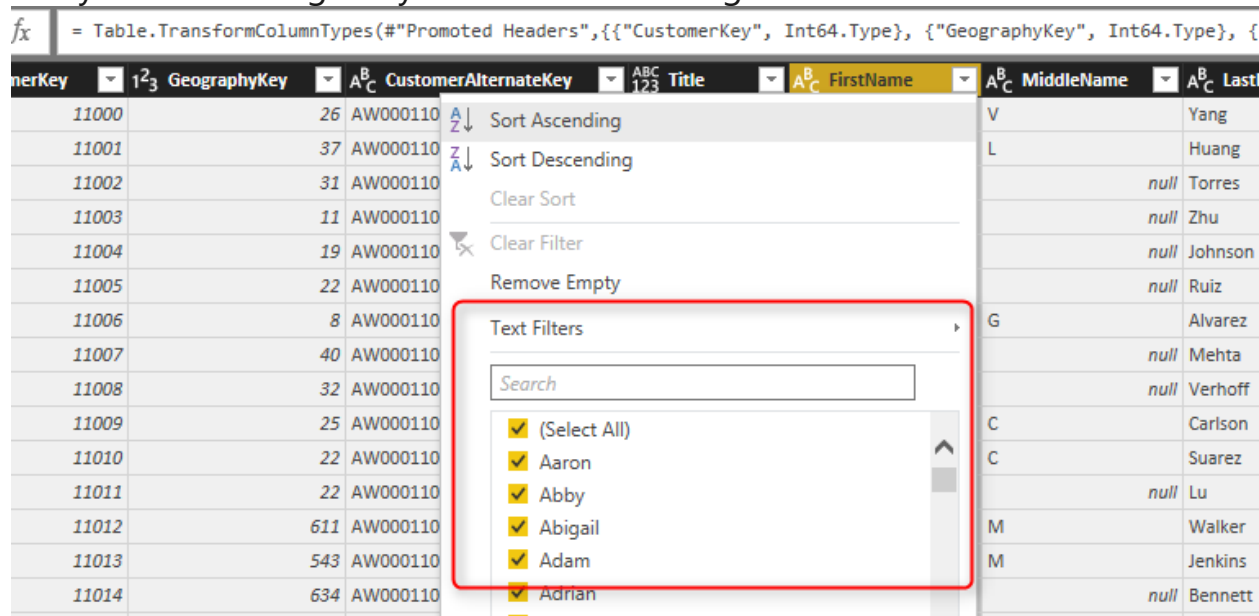
## Prerequisite

For this example, I'll use the AdventureWorksDW SQL Server database example. Only one table which is DimCustomer.

## Filtering in Power Query

Filtering rows is one of the most basic requirements in a data preparation tool such as Power Query, and Power Query do it strongly with an Excel-like behavior for filtering. To see what it looks like open Power BI Desktop (or Excel), and Get Data from the database and table above; DimCustomer, Click on Edit to open the Query Editor window.

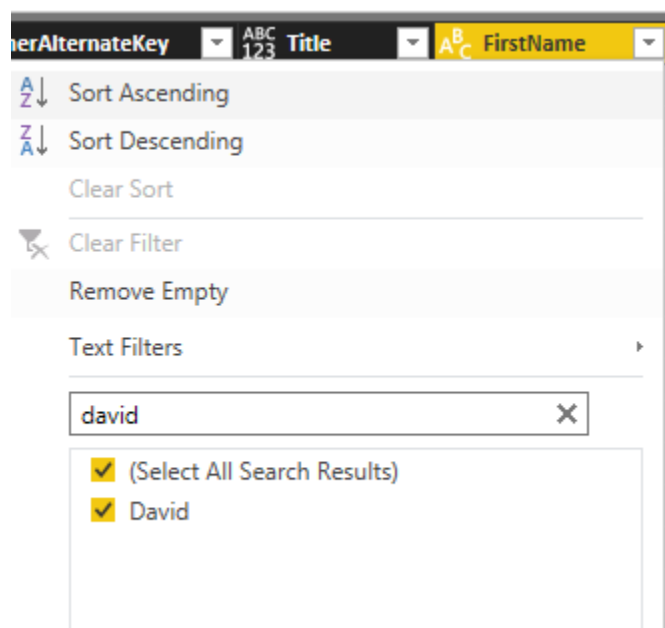
In the Query Editor window, there is a button on the top right-hand side of every column that gives you access to filtering rows



You can simply filter values in the search box provided.

## Basic Filtering

For basic filtering, all you need to do is typing in the text, number, or date you want in the search box and find values to select from the list. For example, if you want to filter all data rows to be only for the FirstName of David, you can simply type it and select and click on OK.



Result then will be only records with David as the first name.

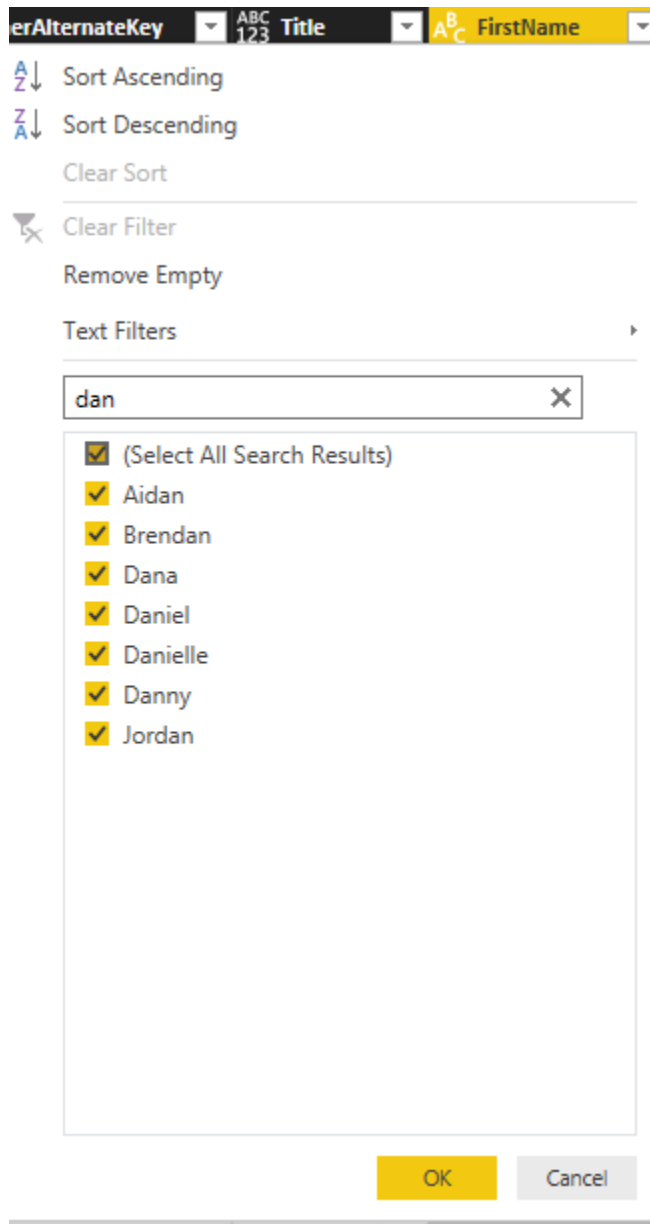
	123 CustomerKey	123 GeographyKey	ABC CustomerAlternateKey	ABC Title	ABC FirstName	ABC MiddleName	ABC LastName
1	11168	348	AW00011168	null	David	null	Rodriguez
2	11267	302	AW00011267	null	David	null	Diaz
3	11377	161	AW00011377	Mr.	David	R.	Robinett
4	11701	347	AW00011701	null	David	null	Hayes
5	12017	60	AW00012017	null	David	E	Simmons
6	13396	631	AW00013396	null	David	null	Kumar
7	14374	343	AW00014374	null	David	null	Yang
8	15203	315	AW00015203	null	David	null	Lewis
9	15636	127	AW00015636	null	David	P	White
10	15783	31	AW00015783	null	David	null	Alexander
11	15908	329	AW00015908	null	David	M	Wilson
12	15991	635	AW00015991	null	David	null	Taylor

Exactly as you expect. Now let's see when it is misleading.

## Misleading Behavior of Basic Filtering

If you want to do equity filtering, basic filtering is great. For example, you only want to pick David, and that's what we've done in the above example.

However what if you want to do similarity filtering? For example, let's say you are interested in FirstNames that has three characters of "Dan" in it. You can do it in the same search box of basic filtering, and this will give you a list of all first names that have "Dan" in it.



And then you can click on OK. The result will be showing you all first names with “Dan” characters in it. So everything looks exactly as expected, right?

	123 CustomerKey	123 GeographyKey	A <sup>B</sup> <sub>C</sub> CustomerAlternateKey	ABC 123 Title	A <sup>B</sup> <sub>C</sub> FirstName	A <sup>B</sup> <sub>C</sub> MiddleName	A <sup>B</sup> <sub>C</sub> LastName
1	11020	53	AW00011020	null	Jordan	C	King
2	11051	21	AW00011051	null	Daniel	null	Johnson
3	11125	37	AW00011125	null	Dana	J	Navarro
4	11207	635	AW00011207	null	Danielle	null	Cox
5	11271	360	AW00011271	null	Danielle	C	Reed
6	11420	189	AW00011420	null	Jordan	C	Turner
7	11490	217	AW00011490	null	Danny	null	Rubio
8	11504	315	AW00011504	null	Jordan	P	Turner
9	11541	547	AW00011541	null	Aidan	D	Ford
10	11711	54	AW00011711	null	Daniel	null	Johnson
11	11798	611	AW00011798	null	Brendan	Q	Cox
12	11818	312	AW00011818	null	Daniel	C	Cox
13	11858	361	AW00011858	null	Jordan	A	Cox
14	11950	312	AW00011950	null	Jordan	null	Cox
15	12084	545	AW00012084	null	Aidan	O	Ford
16	12361	22	AW00012361	null	Dana	C	Cox
17	12587	9	AW00012587	null	Danny	null	Turner
18	12678	15	AW00012678	null	Dana	S	Turner

However, as a Power Query geek, I always look at the M script of each step to see what is happening behind the scene. For this case, here is the Power Query M script;

✕ ✓  $f_x$

= Table.SelectRows(#"Changed Type", each ([FirstName] = "Aidan" or [FirstName] = "Brendan" or [FirstName] = "Dana" or [FirstName] = "Daniel" or [FirstName] = "Danielle" or [FirstName] = "Danny" or [FirstName] = "Jordan"))

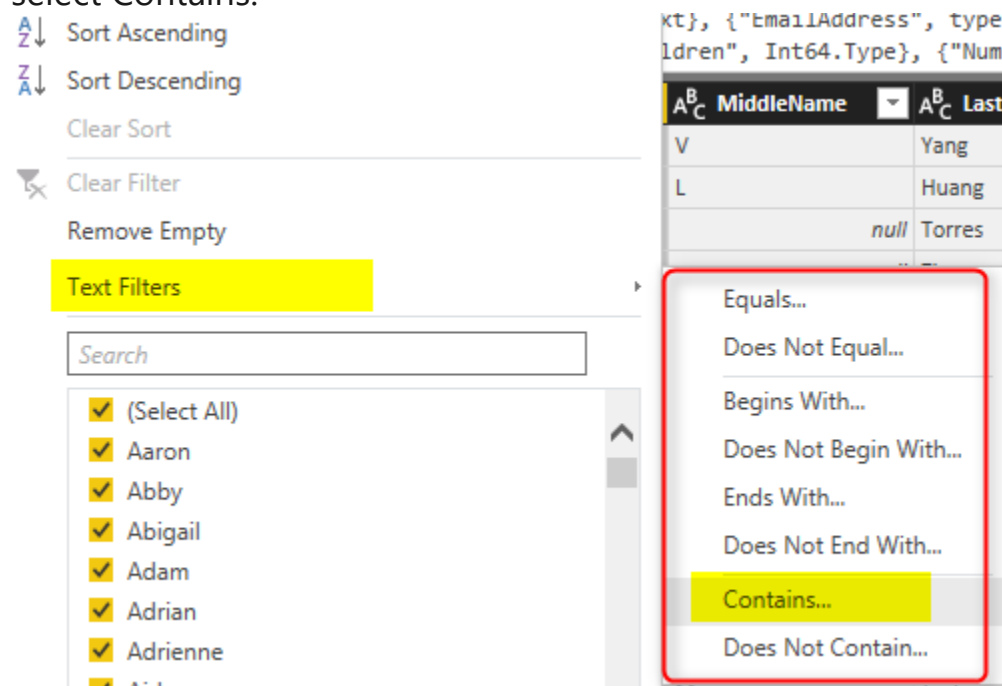
	CustomerAlternateKey	ABC 123 Title	A <sup>B</sup> <sub>C</sub> FirstName	A <sup>B</sup> <sub>C</sub> MiddleName	A <sup>B</sup> <sub>C</sub> LastName
1	0011020	null	Jordan	C	King
2	0011051	null	Daniel		null Johnson
3	0011125	null	Dana	J	Navarro
4	0011207	null	Danielle		null Cox
5	0011271	null	Danielle	C	Reed
6	0011420	null	Jordan	C	Turner
7	0011490	null	Danny		null Rubio

The script tells the whole story. Even though you typed in "Dan" and Power Query showed you all FirstNames that has "Dan" in it. The script still uses equity filters for every individual FirstName. For this data set, there won't be any issue obviously, because all FirstNames with "Dan" is already selected. However, if new data rows are coming into this table in the future, and they

have records with FirstNames that are not one of these values, for example, Dandy, it won't be picked! As a result, the filter won't work exactly as you expect. That's why I say this is misleading.

## Advanced Filtering: Correct Filtering

Basic filtering is very handy with that search box and suggestions of items while you are typing, but it is misleading easily as you have seen above, and can't give you what you expect in many real-world scenarios. Advanced Filtering is simply what you can see on top of the Search box, which depends on the data type of column can be Text Filters, Number Filters, Date Filters, etc. If we want to do the same "Dan" filter here, I can go to Text Filtering and select Contains.



You can even choose filters such as Begins With or Ends With or many other choices. For this example, I'll select Contains, and type in Dan in the text box of Contains.

## Filter Rows

☒ Basic ☐ Advanced

Show rows where: FirstName

contains dan

☒ And ☐ Or

Type or select a value

OK

Cancel

After clicking on OK, you will see the same result set (like the time that you did basic filtering with "Dan"). However the M script this time is different;

= Table.SelectRows(#"Changed Type", each Text.Contains([FirstName], "dan"))							
	CustomerAlternateKey	ABC 123 Title	A <sup>B</sup> C <sub>C</sub> FirstName	A <sup>B</sup> C <sub>C</sub> MiddleName	A <sup>B</sup> C <sub>C</sub> LastName		Nam
1	0011020		null Jordan	C	King		
2	0011420		null Jordan	C	Turner		
3	0011504		null Jordan	P	Baker		
4	0011541		null Aidan	D	Ross		
5	0011798		null Brendan	Q	Chande		
6	0011858		null Jordan	A	Coleman		
7	0011950		null Jordan		null Carter		
8	0012084		null Aidan	O	Perry		
9	0012887		null Brendan	L	Pal		
10	0013018		null Brendan		null Xie		
11	0013095		null Jordan		null Griffin		
12	0013103		null Brendan		null Raji		
13	0013250		null Jordan	J	King		

You can see the big difference now, This time M script used **Text.Contains** function. This function will pick every new value with "Dan" in the FirstName. There will be no values missing in this way. You will be amazed at how many Power Query scenarios are not working correctly because of this very simple reason. There are advanced filters for all types of data types.



## Summary

Basic Filtering is good only if you want to do equity filtering for values that exist in the current data set. However it won't work correctly if you want to check ranges, or contains or things that are not an exact equity filter.

Advanced Filtering is the correct way of filtering in Power Query, and there are advanced filters for all types of data types; Numbers, Text, Date.... This is a very simple fact, but not considering that will bring lots of unwanted behavior and incorrect insight to your Power BI solution.

# Grouping in Power Query; Getting The Last Item in Each Group

Posted by [Reza Rad](#) on Aug 22, 2016

	1.2 CustomerKey	1.2 Order Count	1.2 Total Revenue	Order Details	1.2 First SalesAmount	1.2 Last SalesAmount
1	21768	2	4118.26	Table	3578.27	539.99
2	28389	1	3399.99	Table	3399.99	3399.99
3	25863	5	4631.11	Table	3399.99	2.29
4	14501	2	2994.0882	Table	699.0982	2294.99
5	11003	9	8139.29	Table	3399.99	2.29
6	27645	5	6051.31	Table	3578.27	54.99
7	16624	4	5938.25	Table	3578.27	35
8	11005	6	8121.33	Table	3374.99	2384.07
9	11011	4	8133.04	Table	3399.99	53.99
10	27621	3	5971.33	Table	3578.27	8.99
11	27616	5	5998.61	Table	3578.27	2.29

pDateKey	CustomerKey	PromotionKey	CurrencyKey	SalesTerritoryKey	SalesOrderNumber	SalesOrderLineNumber	RevisionNumber	OrderQuantity	UnitPrice	
20050708	25863	1	100		1 SO43699		1	1	1	3399.99
20080203	25863	1	100		1 SO62866		1	1	1	1214.85
20080203	25863	1	100		1 SO62866		2	1	1	4.99
20080203	25863	1	100		1 SO62866		3	1	1	8.99
20080203	25863	2	100		1 SO62866		4	1	1	2.29

Power BI or Power Query in Excel (or Get Data and Transform as the new name of it) can do many data transformations. One of these transformations is grouping rows by a number of fields. If you use the graphical user interface in Power Query for Power BI or Excel, you have a number of options to get some aggregated results such as a count of rows, maximum or minimum, an average of each group, or sum... But there are still heaps of operations that you cannot perform through GUI, such as getting the last item in each group, or first item. Fortunately, with M (Power Query Formula Language), you can apply any of these operations you want. In this post I'll show you how to get the benefit of both; start with GUI to write the Group by the command for you, and then customize it in M script to achieve what you want. If you like to learn more about Power BI read [Power BI online book; from Rookie to Rock Star](#). If you like to learn more about Power Query, start with the [Introduction to Power Query](#).

## Learning Objectives for this section

By completing the example of this post, you will learn;

- How to perform Group By in Power Query/Power BI
- How to leverage pre-defined aggregation functions in Group By GUI window
- How to extend grouping possibilities with changes in Power Query script

### **Prerequisite**

For this example, I will be using AdventureWorksDW sample Microsoft SQL Server database. You can download it from [here](#).

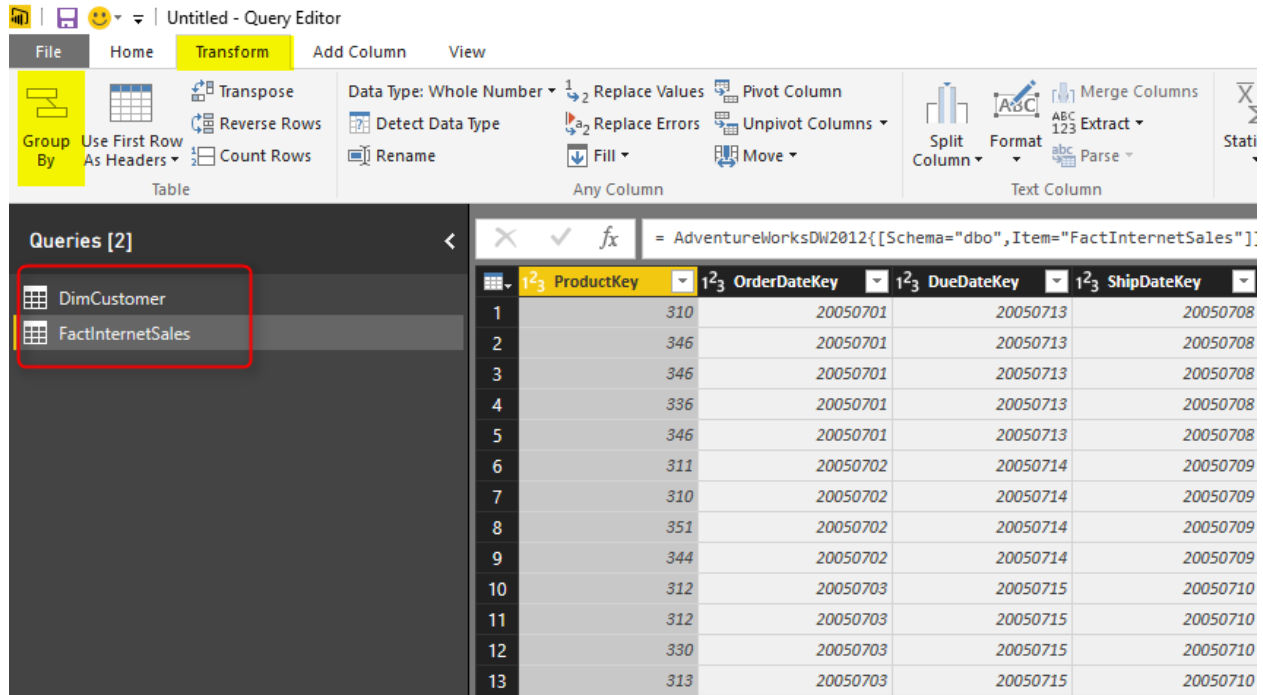
### **Scenario**

The scenario that I want to solve as an example is this:

FactInternetSales has sales transaction information for each customer, by each product, each order date, and some other information. We want to have a grouped table by the customer, which has the number of sales transaction by each customer, total sales amount for that customer, the first and the last sales amount for that customer. First and last defined by the first and last order date for the transaction.

### **Get Data**

Let's start by getting data from SQL Server, Choose AdventureWorksDW as the source database, and select DimCustomer and FactInternetSales as the only tables for this example. Click on Edit to move into Power Query window.



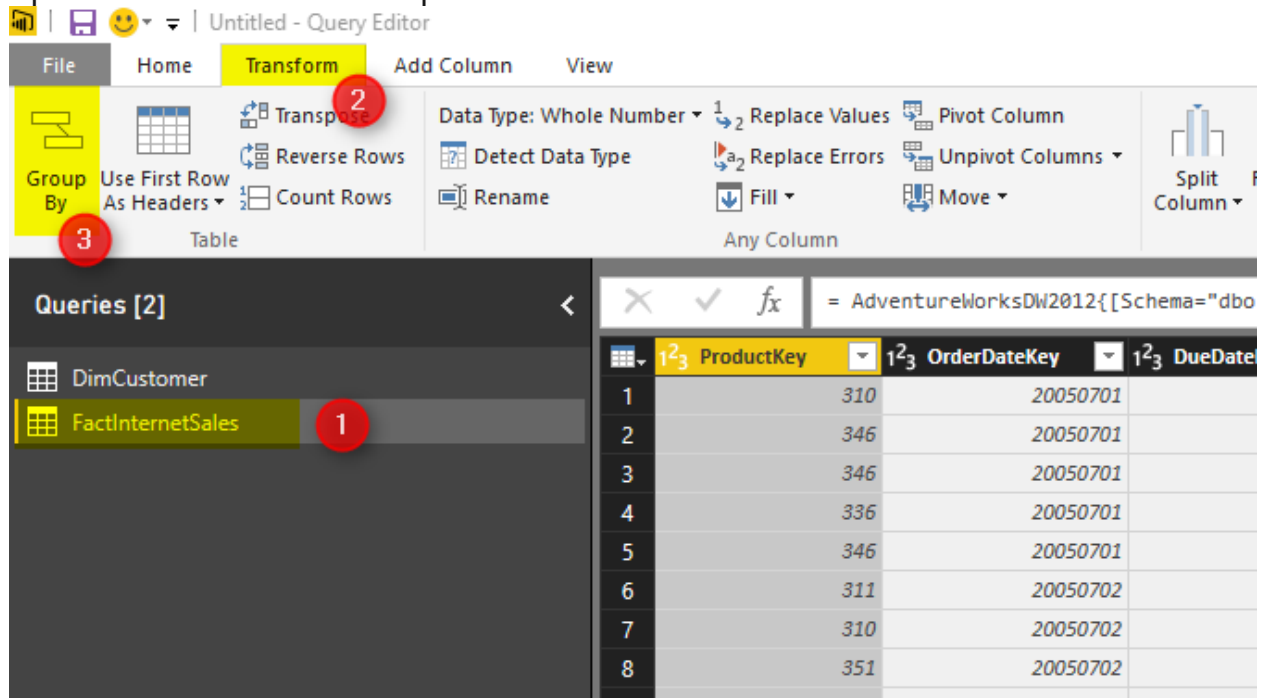
Queries [2]

- DimCustomer
- FactInternetSales

	ProductKey	OrderDateKey	DueDateKey	ShipDateKey
1	310	20050701	20050713	20050708
2	346	20050701	20050713	20050708
3	346	20050701	20050713	20050708
4	336	20050701	20050713	20050708
5	346	20050701	20050713	20050708
6	311	20050702	20050714	20050709
7	310	20050702	20050714	20050709
8	351	20050702	20050714	20050709
9	344	20050702	20050714	20050709
10	312	20050703	20050715	20050710
11	312	20050703	20050715	20050710
12	330	20050703	20050715	20050710
13	313	20050703	20050715	20050710

## Group By Transformation

FactInternetSales table is the one we want to apply all transformations in. So Click on FactInternetSales first, then from Transform Tab, select Group By option as the first menu option.

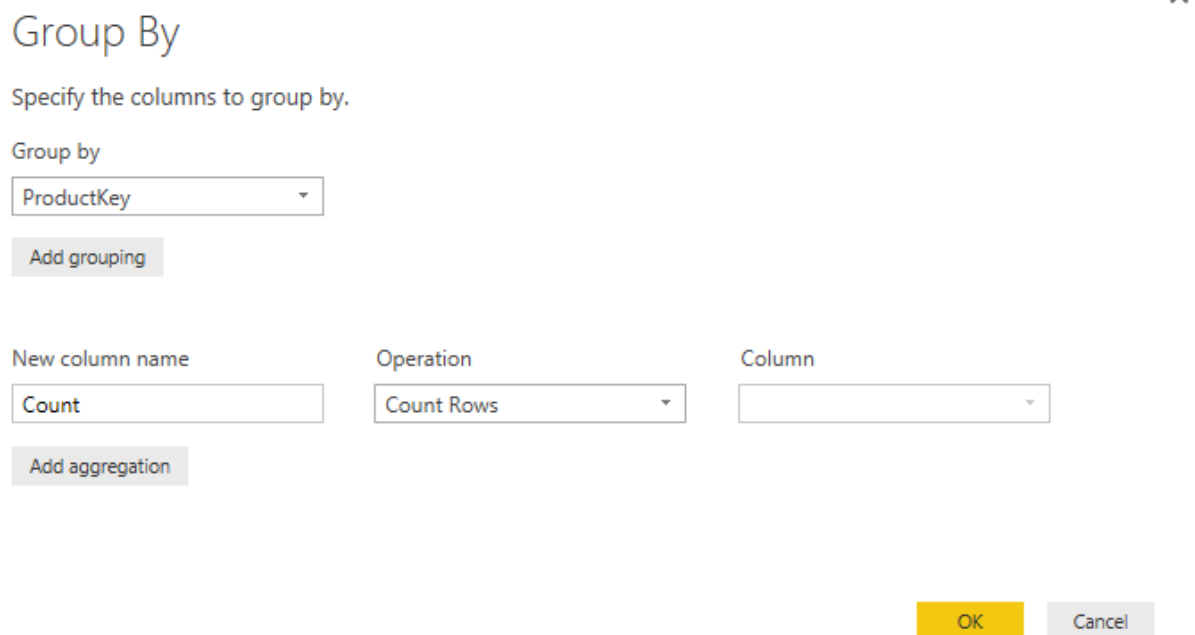


Queries [2]

- DimCustomer
- FactInternetSales

	ProductKey	OrderDateKey	DueDateKey
1	310	20050701	
2	346	20050701	
3	346	20050701	
4	336	20050701	
5	346	20050701	
6	311	20050702	
7	310	20050702	
8	351	20050702	

This will open the Group By dialog window with some configuration options



The image shows the 'Group By' dialog window in Power BI. It has a title bar 'Group By' with a close button 'x'. Below the title is the instruction 'Specify the columns to group by.' There are two main sections. The first section is for grouping, with a 'Group by' dropdown menu showing 'ProductKey' and an 'Add grouping' button. The second section is for aggregation, with three columns: 'New column name' (text input with 'Count'), 'Operation' (dropdown menu with 'Count Rows'), and 'Column' (empty dropdown menu). There is an 'Add aggregation' button below this section. At the bottom right are 'OK' and 'Cancel' buttons.

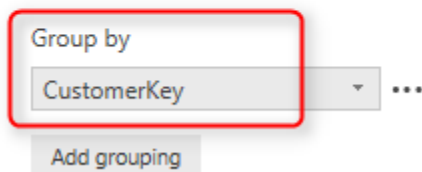
By default Group By happens on the selected columns. Because usually, the first column is the selected column (in our table ProductKey), then the column mentioned under group by section is also ProductKey. You can change this to another column and add or remove columns in this section.

### Choose the Group By Field

Based on your final grain of the output table the group by field will be defined. In this example we want the final table to have one record per Customer, so CustomerKey (which is the identifier for each customer) should be our Group By Column.

## Group By

Specify the columns to group by.



Group by

CustomerKey

Add grouping

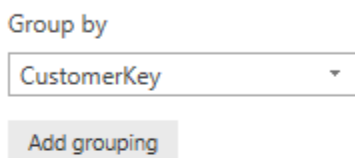
Note that you can add as many fields as you want in the Group By section. The result would be one record per combination of unique values from all these fields.

### Add Aggregation Fields

Group by result would be one record per each unique combination of all fields set in the "group by" section. Also you can also have some aggregated columns. Here is a list of operations you can have by default:

## Group By

Specify the columns to group by.



Group by

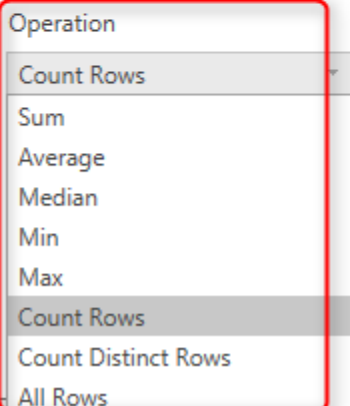
CustomerKey

Add grouping

New column name

Count

Add aggregation



Operation

Count Rows

Sum

Average

Median

Min

Max

Count Rows

Count Distinct Rows

All Rows

Most of the items above are self-explanatory. For example; when you want to count a number of the sales transaction. You can use Count Rows. If you want total Sales amount for each group, you can choose Sum, and then in the Column section choose the column as SalesAmount. All Rows will generate a sub-table in each element of the aggregated table that contains all rows in that group.

Columns that I want to create in this section are:

Order Count (Count Rows), Total Revenue (Sum of Sales Amount), Order Details (All Rows)

## Group By

Specify the columns to group by.

Group by

CustomerKey

Add grouping

New column name	Operation	Column
Order Count	Count Rows	
Total Revenue	Sum	SalesAmount
Order Details	All Rows	

Add aggregation

Adding aggregated columns is as easy as that. Now If you click on OK, you will see the result;

	CustomerKey	Order Count	Total Revenue	Order Details
1	21768	2	4118.26	Table
2	28389	1	3399.99	Table
3	25863	5	4631.11	Table
4	14501	2	2994.0882	Table
5	11003	9	8139.29	Table
6	27645	5	6051.31	Table
7	16624	4	5938.25	Table
8	11005	6	8121.33	Table
9	11011	4	8133.04	Table
10	27621	3	5971.33	Table
11	27616	5	5998.61	Table
12	20042	3	3150.3982	Table
13	16351	2	5873.26	Table
14	16517	2	4698.76	Table
15	27606	3	5997.33	Table
16	13513	4	5918.24	Table
17	27601	1	3578.27	Table
18	13591	9	8355.27	Table
19	16483	2	4118.26	Table

ProductKey	OrderDateKey	DueDateKey	ShipDateKey	CustomerKey	PromotionKey	CurrencyKey	SalesTerritoryKey	SalesOrderNumber
346	20050701	20050713	20050708	11003	1	6	9	SO43701
361	20070709	20070721	20070716	11003	1	6	9	SO51315
478	20070709	20070721	20070716	11003	1	6	9	SO51315
477	20070709	20070721	20070716	11003	1	6	9	SO51315
225	20070709	20070721	20070716	11003	1	6	9	SO51315
564	20071111	20071123	20071118	11003	1	6	9	SO57783
541	20071111	20071123	20071118	11003	1	6	9	SO57783
530	20071111	20071123	20071118	11003	1	6	9	SO57783
480	20071111	20071123	20071118	11003	1	6	9	SO57783

As you can see Order Count and Total Revenue show the aggregated result of each group, and Order Details (if you click not on the “Table” itself, but on a blank area on that cell) contains the actual rows in each group. This detailed view can be used for many other calculations or transformations later on. In many cases, you will find the All rows option useful.

## First and Last Item in each Group

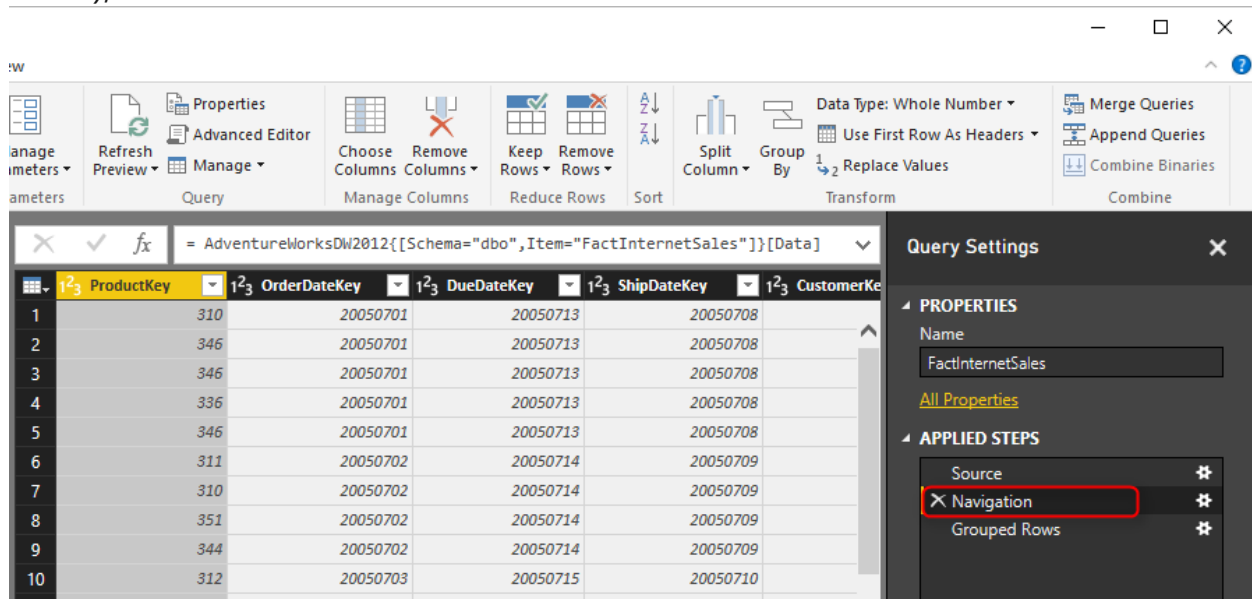
Getting some default aggregation was as easy as selecting them in Group By window. However, not all types of operations are listed there. For example in the detailed table above you can see that customer 11003 had nine sales transaction, and they happened in different Order dates, getting the first and last order date is easy with Max and Min operations. However getting the sales amount or product key associated with that record, or in other words



getting the first and last item in each group isn't possible through GUI. Fortunately, we can use M (Power Query formula language) to achieve this easily.

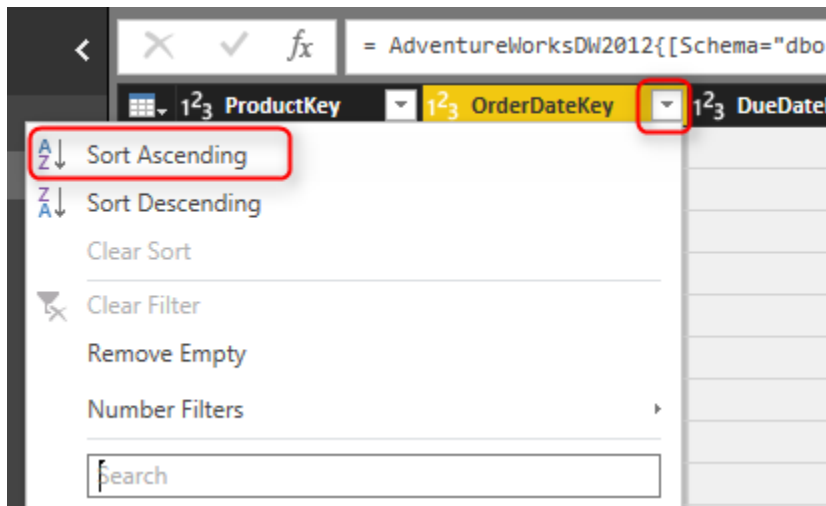
## Sort By Date

To get the first or last item in each group, I have to order the table based on that date column. Sorting is possible simply through GUI., and I have to apply that to the step before group by operation. So from the right-hand side applied steps list I'll select Navigation (which is the step before Grouped Rows);



The screenshot displays the Power BI Desktop interface with the Power Query editor. The ribbon at the top includes tabs for 'Query', 'Transform', and 'Combine'. The 'Sort' button in the 'Transform' tab is highlighted. The data table shows columns: ProductKey, OrderDateKey, DueDateKey, ShipDateKey, and CustomerKey. The 'Query Settings' pane on the right shows the 'APPLIED STEPS' list with 'Navigation' selected and highlighted by a red box.

Now in this view, you can order simply by clicking on OrderDateKey and Choose Sort Ascending.



This will create another step, and asks you do you want to INSERT this step here?

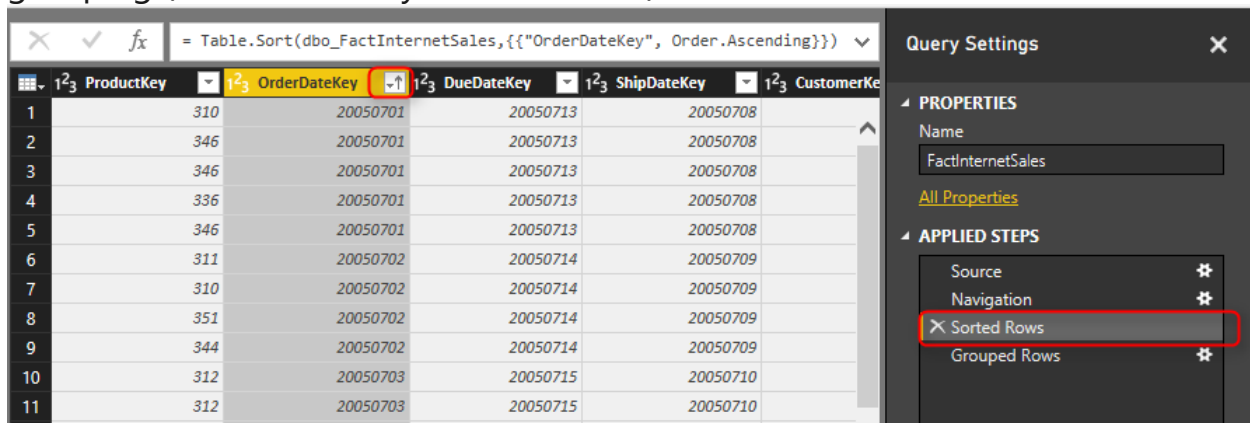
## Insert Step

Are you sure you want to insert a step? Inserting an intermediate step may affect subsequent steps, which could cause your query to break.

Insert

Cancel

Click on Insert to confirm you want to insert it here before the Grouped Rows. And then you will see a Sorted Rows step added before Grouped Rows. This means Grouped Rows will use the output of Sorted Rows step as the input for grouping (which is exactly what we want).



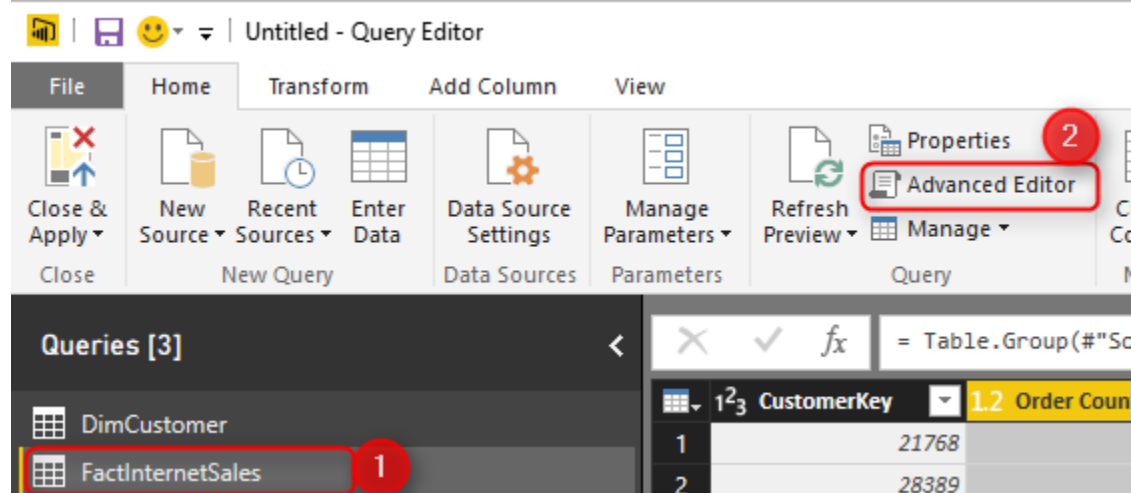
Now you can go to the Grouped Rows step to see the result hasn't changed, but the sub-tables are sorted now. All we need from here is to get the first and last item in the subtable.

\*\* If you want to sort based on multiple columns go to the Advanced Editor and add as many as sections you want to Table. Sort input parameters.

### List.First and List.Last

Fortunately, Power Query has a bunch of operations on List that we can use. List.First will return the first item in the list (based on the order defined in the list), and List.Last will return the last item. So let's use them in the Group By operation to fetch the first and last sales amount.

To make changes, you need to go to script editor in Power Query which can be achieved via Advanced Editor option in the Home tab. You have to make sure that you are in the FactInternetSales Query first.



Advanced Editor will show you M script that builds the output and the group by the command as well.

```

1 let
2     Source = Sql.Databases("."),
3     AdventureWorksDW2012 =
4     Source{[Name="AdventureWorksDW2012"]}[Data],
5

```

```

6  dbo_FactInternetSales =
7  AdventureWorksDW2012[[Schema="dbo",Item="FactInternetSales"]][Data]
8  ,
    #"Sorted Rows" = Table.Sort(dbo_FactInternetSales,{{"OrderDateKey",
    Order.Ascending}}),
    #"Grouped Rows" = Table.Group(#"Sorted Rows", {"CustomerKey"},
    {{"Order Count", each Table.RowCount(_), type number}, {"Total Revenue",
    each List.Sum([SalesAmount]), type number}, {"Order Details", each _, type
    table}})
in
    #"Grouped Rows"

```

The script in above code section created automatically when you did transformations through GUI. The line with Table.Group is the line that does all the grouping and aggregation. It is a long line, so let me format it better for easier understanding;

```

    #"Grouped Rows" = Table.Group(#"Sorted Rows",
6          {"CustomerKey"},
7          {
8          {"Order Count", each Table.RowCount(_),
9  type number},
10         {"Total Revenue", each
11 List.Sum([SalesAmount]), type number},
12         {"Order Details", each _, type table}
13         }
    )

```

Script below is the same script. I just put some enters and tabs to format it better for reading. The above section shows Table.Group section of the script. As you can see Table.Group gets a table as input, which is the #"Sorted Rows" table from the previous step. The group by field is "CustomerKey". and then a set of aggregated columns one after each other (which is highlighted in the code above). Each column has the name of the column, type of transformation (or aggregation), and the data type of the column. For example:

```

    {"Total Revenue", each
10 List.Sum([SalesAmount]), type number},

```

Total Revenue is the name of the column. Calculation for this column is the Sum of [SalesAmount] which is one of the fields in the table, and the output is of type number.

So by now you should be thinking of how each is to create a new aggregated column here; by adding similar column in the script. I add the new column after Order Details column, so I need an extra comma (,) after that line, and the new lines would be;

```

1 let
2     Source = Sql.Databases("."),
3     AdventureWorksDW2012 =
4 Source{[Name="AdventureWorksDW2012"]}[Data],
5     dbo_FactInternetSales =
6 AdventureWorksDW2012{[Schema="dbo",Item="FactInternetSales"]}[Data],
7
8     #"Sorted Rows" = Table.Sort(dbo_FactInternetSales,{"OrderDateKey",
9 Order.Ascending}),
10    #"Grouped Rows" = Table.Group(#"Sorted Rows",
11                                {"CustomerKey"},
12                                {
13                                {"Order Count", each Table.RowCount(_),
14 type number},
15                                {"Total Revenue", each
16 List.Sum([SalesAmount]), type number},
17                                {"Order Details", each _, type table},
18                                {"First SalesAmount", each
19 List.First([SalesAmount]), type number},
20                                {"Last SalesAmount", each
21 List.Last([SalesAmount]), type number}
22                                },
23                                )
24 in
25    #"Grouped Rows"

```

Marked lines above use List.First and List.Last on the same structure that List.Sum worked. Because we have already sorted the table based on OrderDate so the first item would be the first sales transaction, and the last item would be the last.

Here is the output of this change:

	1.2 CustomerKey	1.2 Order Count	1.2 Total Revenue	Order Details	1.2 First SalesAmount	1.2 Last SalesAmount
1	21768	2	4118.26	Table	3578.27	539.99
2	28389	1	3399.99	Table	3399.99	3399.99
3	25863	5	4631.11	Table	3399.99	2.29
4	14501	2	2994.0882	Table	699.0982	2294.99
5	11003	9	8139.29	Table	3399.99	2.29
6	27645	5	6051.31	Table	3578.27	54.99
7	16624	4	5938.25	Table	3578.27	35
8	11005	6	8121.33	Table	3374.99	2384.07
9	11011	4	8133.04	Table	3399.99	53.99
10	27621	3	5971.33	Table	3578.27	8.99
11	27616	5	5998.61	Table	3578.27	2.29

pDateKey	CustomerKey	PromotionKey	CurrencyKey	SalesTerritoryKey	SalesOrderNumber	SalesOrderLineNumber	RevisionNumber	OrderQuantity	UnitPrice	Ex
20050708	25863	1	100		1 SO43699		1	1	3399.99	
20080203	25863	1	100		1 SO62866		1	1	1214.85	
20080203	25863	1	100		1 SO62866		2	1	4.99	
20080203	25863	1	100		1 SO62866		3	1	8.99	
20080203	25863	2	100		1 SO62866		4	1	2.29	

You can see that the first and the last SalesAmount picked correctly from each group as two new columns.

\*\* Note that with adding some changes in script editor that are not supported through GUI, you will lose the GUI configuration window section. As a result of the change here you cannot change Grouping configuration in GUI anymore, if you want to change it, you have to go to Advanced Editor for this section. So if you are a GUI fan, better to apply all required configuration first, and then add extra logic in the code.

# Fuzzy Matching in Power BI and Power Query; Match based on Similarity Threshold

Posted by [Reza Rad](#) on Oct 23, 2018

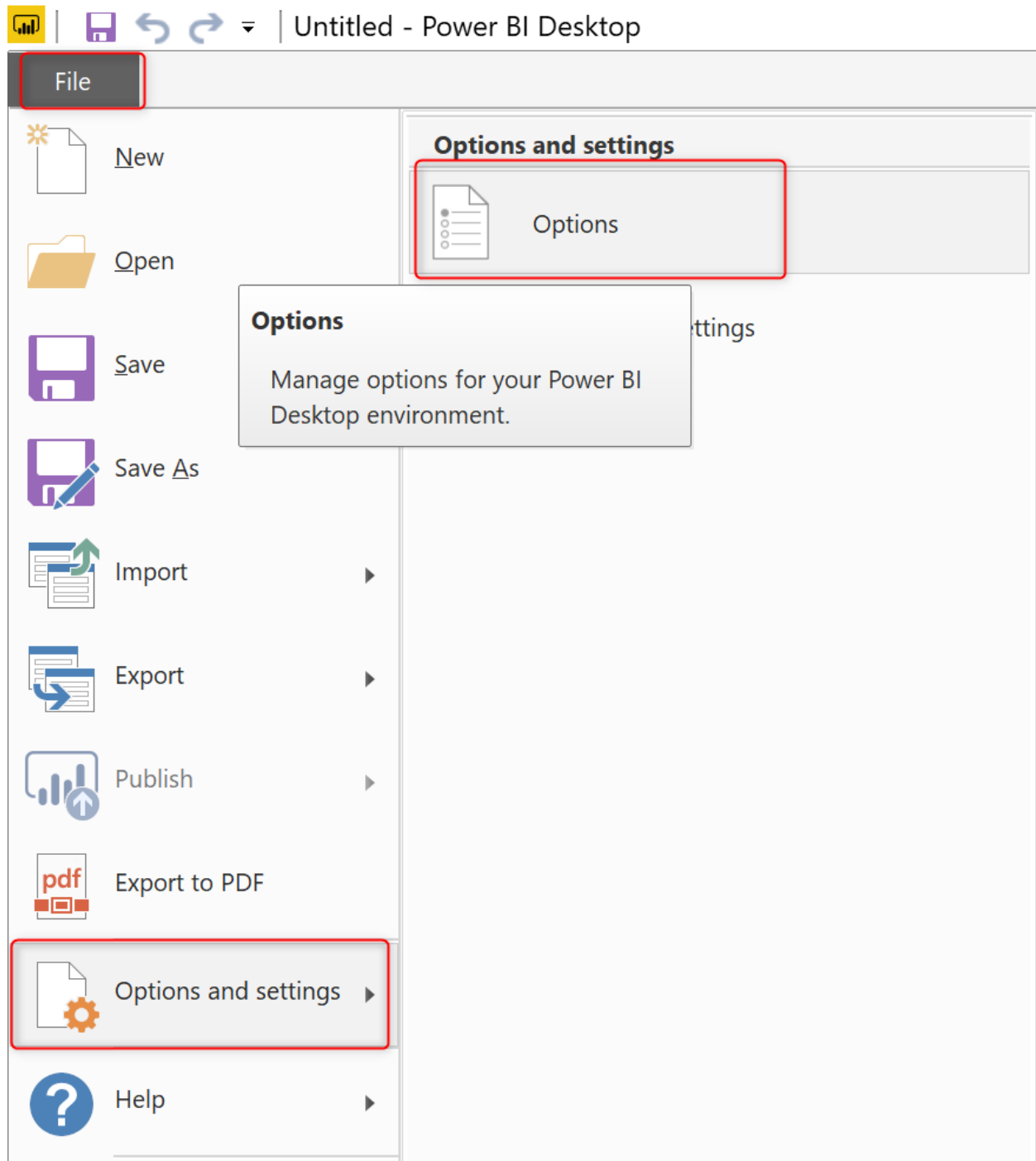
1 <sup>2</sup> 3 ID	A <sup>B</sup> C Name	A <sup>B</sup> C Department	A <sup>B</sup> C Department Name
1	1 Mike Anderson	Information Technology	Information Technology
2	2 Antonio Martin	informatica tecnologica	Information Technology
3			Management
4			null
5	5 Paolo Adrian	Sales	Sales
6	6 Antony Page	Sale	Sales
7	7 Brian Farmer	Managmnt	Management

Fuzzy Matching

After a long wait, in the October 2018 release of Power BI Desktop, we saw the fuzzy matching feature added finally. Yay! Have you ever wanted to match two tables together but not on exact matches, but also a threshold of similarity? If your answer to this question is yes, then this feature is built for you. Let's explore in details how the fuzzy matching works in Power BI. To learn more about Power BI, read [Power BI from Rookie to Rock Star](#).

## Enable the Preview Feature

At the time of writing this blog post, Fuzzy matching is a preview feature, and you have to enable it in Power BI Desktop -> Files -> Options and Settings -> Options;



In the Options window, under Preview Features, select the checkbox beside "Enable fuzzy merge"





## Options

### GLOBAL

Data Load  
Power Query Editor  
DirectQuery  
R scripting  
Python scripting  
Security  
Privacy  
Regional Settings  
Updates  
Usage Data  
Diagnostics  
**Preview features**  
Auto recovery

### CURRENT FILE

Data Load  
Regional Settings  
Privacy  
Auto recovery  
Query reduction  
Report settings

### Preview features

The following features are available for you to try in this release. Preview features might change or be removed in future releases.

- ☒ Shape map visual [Learn more](#)
- ☒ M Intellisense [Learn more](#)
- ☒ Spanish language support for Q&A [Learn more](#)
- ☒ Get data from PDF files [Learn more](#)
- ☒ Enable column profiling [Learn more](#)
- ☒ Show dates as a hierarchy in the fields list [Learn more](#)
- ☒ Python support [Learn more](#)
- ☐ Incremental Refresh Policies [Learn more](#)
- ☐ Composite Models [Learn more](#)
- ☐ Manage Aggregations [Learn more](#)
- ☒ Enable fuzzy merge [Learn more](#)

**OK**

Cancel

After this step, you'll need to close the Power BI Desktop and open it again.

### Sample Dataset

for this example; I will be using a sample dataset which has two very simple tables below;

A "source" table which is the data of employees and their departments. Notice that the Department field has data quality issues. We have department values

such as “Sales” and “Sale”. Or another example is “Management” and “Management”.

	ID	Name	Department
1	1	Mike Anderson	Information Technology
2	2	Antonio Martin	informatica tecnologica
3	3	John Jefferson	Management
4	4	Joe McCarthy	Mangmt
5	5	Paolo Adrian	Sales
6	6	Antony Page	Sale
7	7	Brian Farmer	Managmnt

A “Department” table which has a list of all departments;

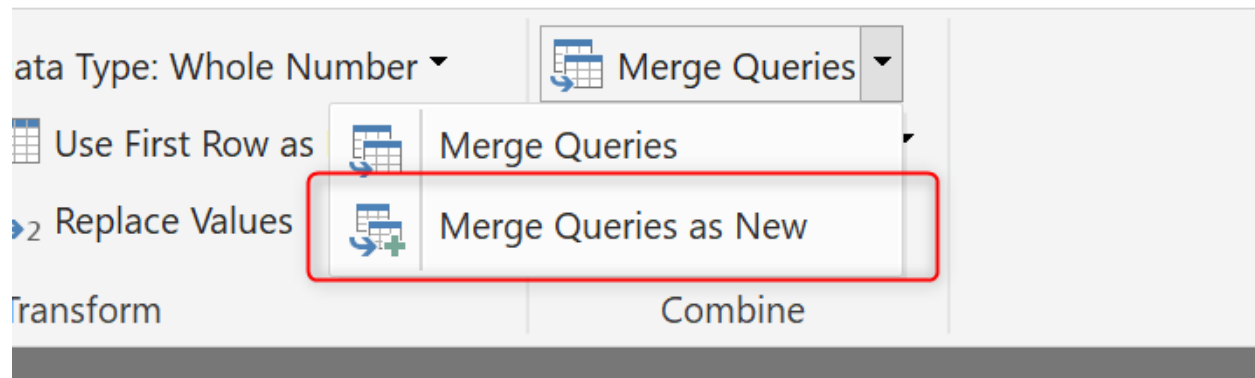
	Department Name
1	Information Technology
2	Sales
3	Management

As you can see the list of Department Names are clean in this table, and this is the table that should be used to clean the “source” table. Now let’s see how this is possible?

## Fuzzy Merge

Fuzzy Merge is a way of joining two tables together, but not on exact matching criteria, on the similarity threshold. If you want to learn what is the Merge operation itself and the difference of that with Append, [read my blog post here](#). If you want to learn more details about what is Merge and the different types of join or merge, [read my other blog post here](#). Merge or Join

is simply the act of combining two tables with different structures, but with link/join columns, to access columns from one of the tables in the other one. To use Merge operation on the “source” query, You can click on the Merge Queries as New option in the Home tab of Power Query Editor window.



```
{"Name", type text}, {"Department", type text}})
```


Then you can select the second table and choose Department as the joining field



## Merge


Select tables and matching columns to create a merged table.

source



ID	Name	Department
1	Mike Anderson	Information Technology
2	Antonio Martin	informatica tecnologica
3	John Jefferson	Management
4	Joe McCarthy	Mangmt
5	Paolo Adrian	Sales

Department



Department Name
Information Technology
Sales
Management

Join Kind

Left Outer (all from first, matching from second)

☐ Use fuzzy matching to compare the merge

> Fuzzy merge options

OK

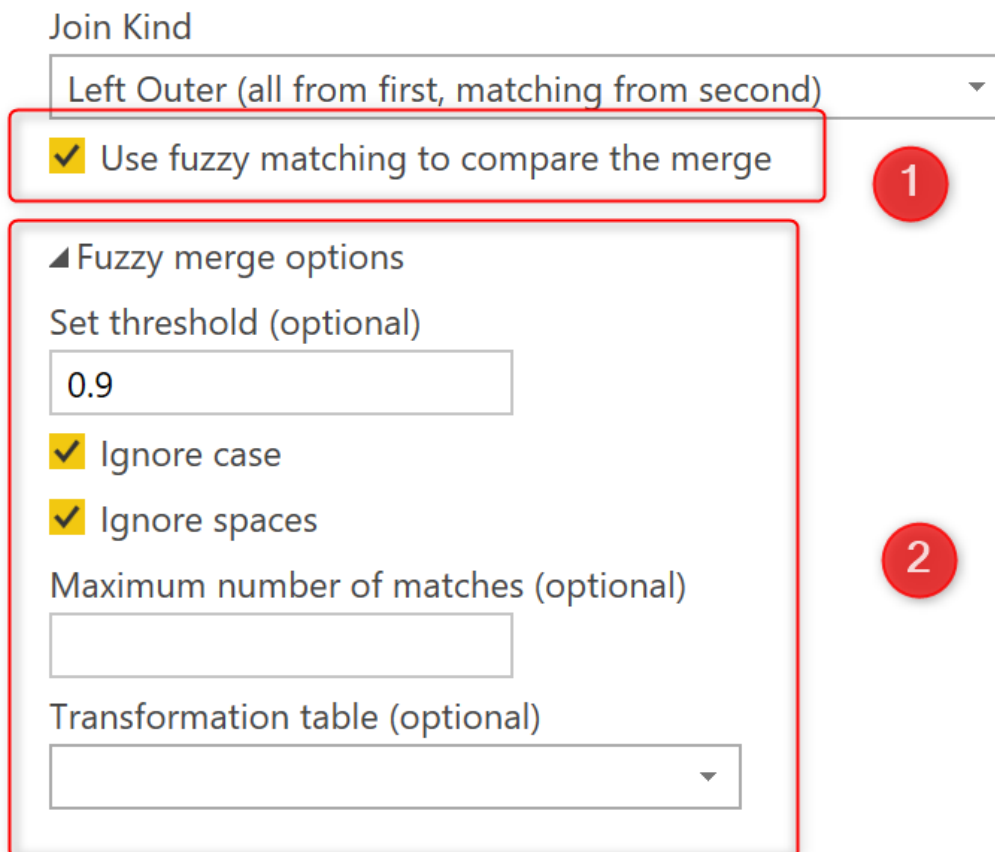
Cancel

This process will give you the output below: (result below is after expanding the merge's column output);

	1 <sup>2</sup> 3 ID	A <sup>B</sup> <sub>C</sub> Name	A <sup>B</sup> <sub>C</sub> Department	A <sup>B</sup> <sub>C</sub> Department Name
1		1 Mike Anderson	Information Technology	Information Technology
2		2 Antonio Martin	informatica tecnologica	null
3		3 John Jefferson	Management	Management
4		4 Joe McCarthy	Mangmt	null
5		5 Paolo Adrian	Sales	Sales
6		6 Antony Page	Sale	null
7		7 Brian Farmer	Managmnt	null

You can see that the Merge operation only finds the EXACT Matching scenarios. Department "Sale" doesn't match with the Department table, because it is missing an "S" at the end to match with the "Sales".

Now, let's see how Fuzzy match works here. To use the Fuzzy Merge, select the checkbox under the Merge tables dialog box;



Join Kind

Left Outer (all from first, matching from second)

☒ Use fuzzy matching to compare the merge

▲ Fuzzy merge options

Set threshold (optional)

0.9

☒ Ignore case

☒ Ignore spaces

Maximum number of matches (optional)

1

Transformation table (optional)

None

When you enable the fuzzy matching, then you can configure it in the "fuzzy merge operations". You can leave everything optional. Or set values. Let's first see the sample output of this operation and then see what the options are. This is the sample output of Fuzzy Merge:

1 <sup>2</sup> 3 ID	A <sup>B</sup> C Name	A <sup>B</sup> C Department	A <sup>B</sup> C Department Name
1	1 Mike Anderson	Information Technology	Information Technology
2	2 Antonio Martin	informatica tecnologica	Information Technology
3	3 John Jefferson	Management	Management
4	4 Joe McCarthy	Mangmt	null
5	5 Paolo Adrian	Sales	Sales
6	6 Antony Page	Sale	Sales
7	7 Brian Farmer	Managmnt	Management

You can see the three highlighted records, which was not recognized as the exact match in the normal merge operation, is not matching the output of the fuzzy merge. Fuzzy merge will check the similarity between joining fields, and if their similarity is more than the threshold configuration, it will pass it as a successful match. You can see that “Managmnt” can match with “Management” with this threshold configuration, but the “Mangmt” doesn’t, it shows that the threshold of similarity is higher than the similarity rate of these two text values with each other.

You can play with Options of Fuzzy Merge and get different outputs. Here is an explanation of these options:

Option	Acceptable Value	Description
Threshold	a value between 0.00 to 1.00	if the similarity of the two text values is more than the threshold, it will be considered as a successful match. Value 1.00 means exact match.
Ignore Case	true/false	If you want the similarity algorithm to work regardless of the upper or lower case letters, then select this option.
Ignore Space	true/false	If you want the similarity algorithm to work regardless of the number of spaces in

Maximum Number of  
Matches

a numeric positive value,  
between 0 to 2147483647

the text, then select this  
option.

The number of rows that can  
be matched to one value.

This is like a mapping table,  
let's check it out a bit later in  
this post. It gives you the  
option to use your mapping  
table. This table should have  
at least two columns of "To"  
and "From".

Transformation Table

table

## Power Query Functions

In addition to the option added in the graphical interface of Power Query, we also have two Power Query Functions that do the Fuzzy Merge, Functions are:

***Table.FuzzyJoin***

***Table.FuzzyNestedJoin***

Functions above both do have the same fuzzy configurations. Their only difference is that one of them gives you the expanded output (FuzzyJoin), the other one gives you the same output as the one that you see in the graphical interface with the table column output after merge (FuzzyNestedJoin). If you use these two functions directly in M script, you will have a couple of more parameters to set, which are for concurrency and culture settings.

These are parameters of the two functions above;

table1

key1 (optional)

table2

key2 (optional)

newColumnName

*Example: abc*

joinKind (optional)

▲ joinOptions (optional)

ConcurrentRequests (optional)

*Example: 123*

Culture (optional)

*Example: abc*

IgnoreCase (optional)

*Example: true*

IgnoreSpace (optional)



## Transformation Table

Sometimes in the merge operation, you need a mapping table. This table is called here as Transformation Table. Here is an example of a mapping table:

	A <sup>B</sup> <sub>C</sub> From	A <sup>B</sup> <sub>C</sub> To
1	Information Technology	IT
2	Sales	Sales
3	Management	Board

Note that this table should have at least the two columns of "To" and "From". And don't forget that Power Query is case sensitive!

Now you can select this table in your Merge operation in the Fuzzy configuration as below;

Join Kind

Left Outer (all from first, matching from second)

☒ Use fuzzy matching to compare the merge

▲ Fuzzy merge options

Set threshold (optional)

0.1

☒ Ignore case

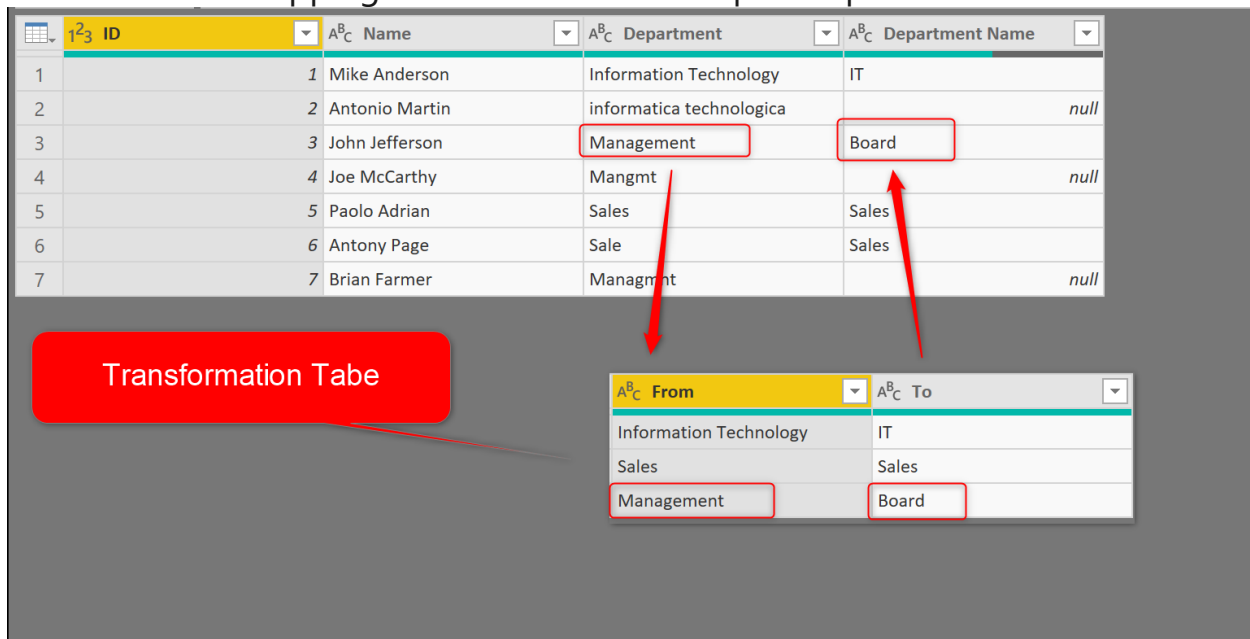
☒ Ignore spaces

Maximum number of matches (optional)

Transformation table (optional)

mapping

This process is like merging “source” table, which is the first table in our Merge, with the “Department” table based on the “Department” and then “Department Name” column, then merging it with the “mapping” table, based on the “To” column and “Department Name”. The output will bring the “To” column of the mapping table. Here is the sample output:



ID	Name	Department	Department Name
1	Mike Anderson	Information Technology	IT
2	Antonio Martin	informatica tecnologica	null
3	John Jefferson	Management	Board
4	Joe McCarthy	Mangmt	null
5	Paolo Adrian	Sales	Sales
6	Antony Page	Sale	Sales
7	Brian Farmer	Managmt	null

From	To
Information Technology	IT
Sales	Sales
Management	Board

## Summary

Matching based on similarity threshold, or Fuzzy matching is a fantastic feature added to Power Query and Power BI, however, it is still a preview feature, and it may have some more configuration coming up. Please try it in your dataset, and let me know if you have any questions in the comment below.

# Fetch Files and/or Folders with Filtering and Masking: Power Query

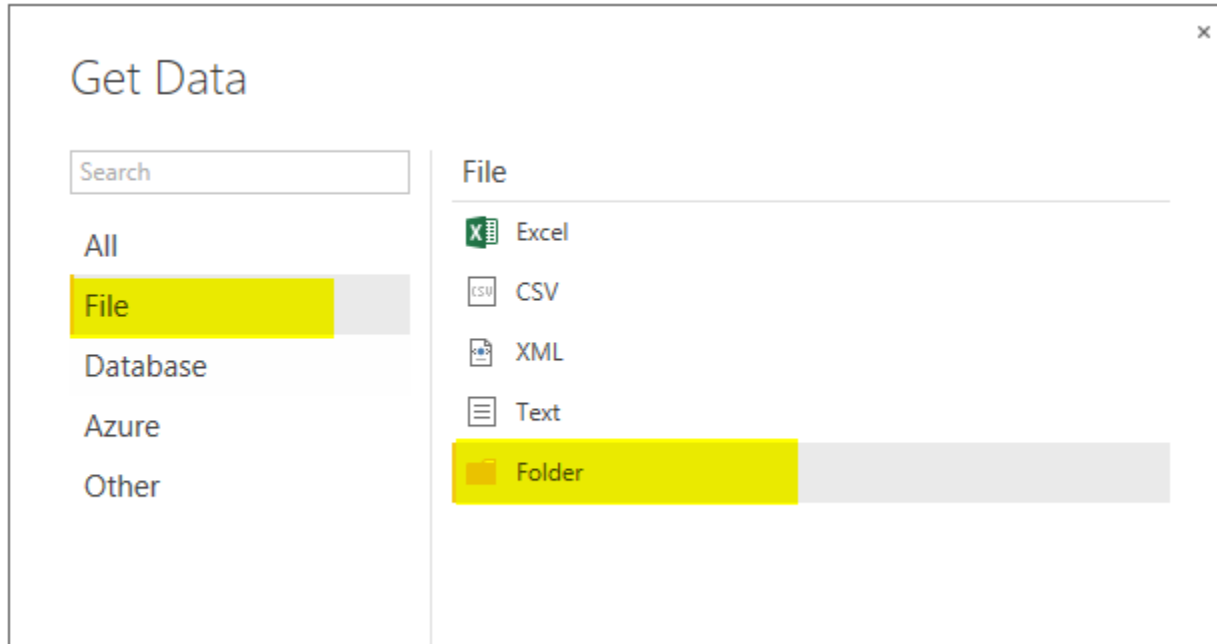
Posted by [Reza Rad](#) on Aug 3, 2015

	Content	Name	Extension	Date accessed	Date modified	Date
1	Table	Auckland CodeCamp 2014		12/13/2014 10:14:54 AM	9/13/2014 2:03:17 PM	9
2	Table	Auckland SQL User Group - 16 May 2012		12/13/2014 10:09:50 AM	1/1/2014 10:09:10 AM	
3	Table	CloudnEnterprise_Symbols_Public_v2.02		2/1/2015 12:43:29 PM	2/1/2015 12:43:29 PM	
4	Binary	CloudnEnterprise_Symbols_Public_v2.02.zip	.zip	2/1/2015 12:43:27 PM	2/1/2015 12:43:10 PM	
5	Binary	DQS.txt	.txt	12/13/2014 10:09:50 AM	11/19/2012 9:23:50 PM	
6	Table	DWBI-VC- Aug 2012		12/13/2014 10:09:51 AM	1/1/2014 9:40:14 AM	
7	Table	Error Handling		12/13/2014 10:04:44 AM	1/1/2014 9:30:34 AM	
8	Table	INF		12/13/2014 10:09:50 AM	1/1/2014 9:37:06 AM	

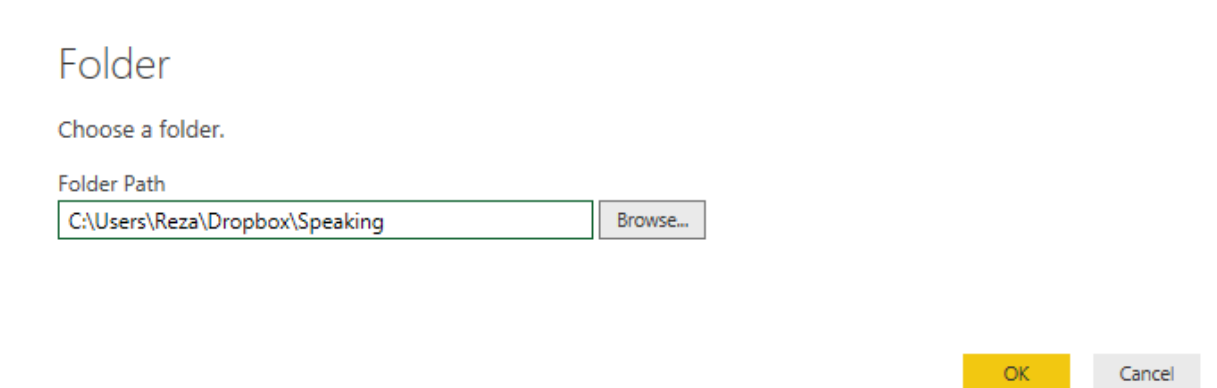
Fetching list of Files in a folder is easy with Power Query. It is one of the built-in source types. However, fetching a list of folders is not a built-in function. In this post, I'll show you how to fetch a list of files, and also fetch the only a list of folders. The method explained in a way that you can customize the code and apply any conditions as you want later on — conditions such as File or folder name masking to fetch only names that contain special character strings.

## Fetch All Files in a Folder

For fetching all files in a folder you can simply use the GUI Get Data, and under File, choose Folder



## Browse for Folder



and then simply you will see the list of all files.

Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	Folder Path
1	Binary	Thumbs.db	7/30/2015 10:32:42 PM	7/30/2015 10:32:42 PM	7/30/2015 10:32:42 PM	Record	C:\Users\Reza\Dropbox\Speaking\CloudEnterprise_Symbols_Public_v2.C
2	Binary	Table Functions.txt	7/27/2015 11:14:38 PM	3/6/2014 10:11:47 AM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\Demo
3	Binary	Table Functions Part 1.xlsx	7/27/2015 11:14:38 PM	2/12/2014 1:14:49 AM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\Demo
4	Binary	Movies.xlsx	7/27/2015 11:14:38 PM	3/5/2014 9:33:43 PM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\Demo
5	Binary	M.xlsx	7/27/2015 11:14:38 PM	2/10/2014 11:46:30 PM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\Demo
6	Binary	M.sample.txt	7/27/2015 11:14:38 PM	2/10/2014 11:46:41 PM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\Demo
7	Binary	Generator.txt	7/27/2015 11:14:38 PM	3/10/2014 2:57:43 PM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\Demo
8	Binary	Error Handling.xlsx	7/27/2015 11:14:38 PM	3/6/2014 1:35:56 PM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\Demo
9	Binary	Error Handling.txt	7/27/2015 11:14:38 PM	3/10/2014 1:38:06 PM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\Demo
10	Binary	Date Dimension.xlsx	7/27/2015 11:14:38 PM	2/15/2014 12:43:03 PM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\Demo
11	Binary	Date Dimension.txt	7/27/2015 11:14:38 PM	3/6/2014 11:05:58 AM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\Demo
12	Binary	Date Dimension - With Public Holidays.xlsx	7/27/2015 11:14:38 PM	2/16/2014 2:29:58 PM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\Demo
13	Binary	D2.xlsx	7/27/2015 11:14:38 PM	11/6/2014 3:07:44 PM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\Demo
14	Binary	Custom Function.xlsx	7/27/2015 11:14:38 PM	2/13/2014 4:28:11 PM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\Demo
15	Binary	Custom Function.txt	7/27/2015 11:14:38 PM	3/6/2014 10:42:41 AM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\Demo
16	Binary	Thumbs.db	7/27/2015 11:14:35 PM	5/28/2015 1:29:50 PM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\
17	Binary	Reza Rad_Top 5 Power Query M Functions that You Don't Kn	7/28/2015 12:03:58 AM	7/28/2015 12:03:58 AM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\
18	Binary	Reza Rad_Top 5 Power Query M Functions that You Don't Kn	7/27/2015 11:14:39 PM	11/8/2014 10:33:55 AM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\
19	Binary	Power Query Demo.rin	7/27/2015 11:14:38 PM	11/8/2014 10:33:55 AM	7/27/2015 11:14:35 PM	Record	C:\Users\Reza\Dropbox\Speaking\NZBIUG July 2015\Power Query\

As you see in the above table extracted, there is a column named Content, which has the content of the file. You can click on that to see the content of the file if you want.

## What is the M Code Written Behind the Scenes

The code behind the scenes for this transformation used [Folder.Files](#) function

```
1 let
2   Source = Folder.Files("C:\Users\Reza\Dropbox\Speaking")
3 in
4   Source
```

## Fetch All Files and Folders

There is another M function to fetch all files and folders listed under a folder, named [Folder.Contents](#). This function returns the Content column with a data type of the record and content of it. For Files data type usually is Binary, and for Folders it is a table. So it can be easily distinguished and separated. Here is a list of all files and folders fetched:

```
1 let
2   Source = Folder.Contents("C:\Users\Reza\Dropbox\Speaking")
3 in
4   Source
```

The result set contains both files and folders

	Content	Name	Extension	Date accessed	Date modified	Date
1	Table	Auckland CodeCamp 2014		12/13/2014 10:14:54 AM	9/13/2014 2:03:17 PM	9
2	Table	Auckland SQL User Group - 16 May 2012		12/13/2014 10:09:50 AM	1/1/2014 10:09:10 AM	
3	Table	CloudnEnterprise_Symbols_Public_v2.02		2/1/2015 12:43:29 PM	2/1/2015 12:43:29 PM	
4	Binary	CloudnEnterprise_Symbols_Public_v2.02.zip	.zip	2/1/2015 12:43:27 PM	2/1/2015 12:43:10 PM	
5	Binary	DQS.txt	.txt	12/13/2014 10:09:50 AM	11/19/2012 9:23:50 PM	
6	Table	DWBI-VC- Aug 2012		12/13/2014 10:09:51 AM	1/1/2014 9:40:14 AM	
7	Table	Error Handling		12/13/2014 10:04:44 AM	1/1/2014 9:30:34 AM	
8	Table	INF		12/13/2014 10:09:50 AM	1/1/2014 9:37:06 AM	

As you see the Content column shows the data type of the value. To check if a record is Folder or not, we have to compare its data type to Table. We can add a column to the table to check the data type of the content column in each record. The data type can be checked with [Value.Is](#) function.

```
1 let
2   Source = Folder.Contents("C:\Users\Reza\Dropbox\Speaking"),
3   TypeAdded=Table.AddColumn(Source,"Type",each Value.Is([Content],type table))
4 in
5   TypeAdded
```

And the result set shows if the record is folder or not

	Attributes	Folder Path	Type
6 AM	Record	C:\Users\Reza\Dropbox\Speaking\	TRUE
7 AM	Record	C:\Users\Reza\Dropbox\Speaking\	TRUE
9 PM	Record	C:\Users\Reza\Dropbox\Speaking\	TRUE
7 PM	Record	C:\Users\Reza\Dropbox\Speaking\	FALSE
0 AM	Record	C:\Users\Reza\Dropbox\Speaking\	FALSE
5 AM	Record	C:\Users\Reza\Dropbox\Speaking\	TRUE
5 AM	Record	C:\Users\Reza\Dropbox\Speaking\	TRUE
5 AM	Record	C:\Users\Reza\Dropbox\Speaking\	TRUE
5 AM	Record	C:\Users\Reza\Dropbox\Speaking\	FALSE
6 PM	Record	C:\Users\Reza\Dropbox\Speaking\	TRUE

To fetch only folders, we can filter the data set with Table.SelectRows function. I've sorted the result set descending by creating date of the folder. Here is the code:

```

1 let
2   Source = Folder.Contents("C:\Users\Reza\Dropbox\Speaking"),
3   TypeAdded=Table.AddColumn(Source,"Type",each Value.Is([Content],type table)),
4   Folders=Table.SelectRows(TypeAdded, each [Type]=true),
5   Sorted=Table.Sort(Folders,{"Date created", Order.Descending})
6
7 in
8   Sorted

```

And the result:

	Content	Name	Extension	Date accessed	Date modified	Date created
1	Table	NZBIUG July 2015		7/27/2015 11:14:35 PM	7/27/2015 11:14:35 PM	7/27/2015 11:14:17 PM
2	Table	SQL Rally Nordic 2015		3/4/2015 8:45:25 PM	3/4/2015 8:45:25 PM	3/3/2015 11:31:56 PM
3	Table	SQL Saturday 374 Vienna		2/28/2015 6:28:23 PM	2/28/2015 6:28:23 PM	2/19/2015 9:03:43 PM
4	Table	CloudnEnterprise_Symbols_Public_v2.02		2/1/2015 12:43:29 PM	2/1/2015 12:43:29 PM	2/1/2015 12:43:29 PM
5	Table	TechDays Hong Kong 2015		2/12/2015 4:14:33 PM	2/12/2015 4:14:33 PM	2/1/2015 12:20:56 PM
6	Table	SQL Saturday 337 Oregon		12/13/2014 10:14:55 AM	11/2/2014 11:15:19 AM	10/31/2014 9:49:55 PM
7	Table	SQL Saturday 352 Sydney		2/7/2015 2:59:03 PM	2/7/2015 2:59:03 PM	10/24/2014 10:59:19 PM
8	Table	SQL Social Brisbane 2014		12/13/2014 10:14:55 AM	9/17/2014 4:28:51 PM	9/17/2014 4:28:26 PM
9	Table	MVP ComCamp 2014		12/13/2014 10:14:53 AM	9/17/2014 4:28:26 PM	9/17/2014 4:28:26 PM
10	Table	PASS Summit 2014		12/13/2014 10:14:52 AM	9/13/2014 6:04:20 PM	9/13/2014 5:44:53 PM
11	Table	Auckland CodeCamp 2014		12/13/2014 10:14:54 AM	9/13/2014 2:03:17 PM	9/13/2014 10:07:16 AM
12	Table	TecheEd NZ 2014		1/11/2015 3:45:45 PM	1/11/2015 3:45:45 PM	7/30/2014 5:43:35 PM

## Fetch Files and Folders with Masking

Now that you've got through the M code, it would be really easy to apply any masking option to this. Here are a couple of examples:

### 1- Fetch Only Folders Created after Specific Date

```

1 let
2   Source = Folder.Contents("C:\Users\Reza\Dropbox\Speaking"),

```

```

3 TypeAdded=Table.AddColumn(Source,"Type",each Value.Is([Content],type table)),
4 Folders=Table.SelectRows(TypeAdded, each [Type]=true),
5 Sorted=Table.Sort(Folders,{"Date created", Order.Descending}),
6 Filtered=Table.SelectRows(Sorted, each [Date created]>DateTime.FromText("2015-1-1"))
7
8 in
9 Filtered

```

## Result

	Content	Name	Extension	Date accessed	Date modified	Date created
1	Table	NZBIUG July 2015		7/27/2015 11:14:35 PM	7/27/2015 11:14:35 PM	7/27/2015 11:14:17 PM
2	Table	SQL Rally Nordic 2015		3/4/2015 8:45:25 PM	3/4/2015 8:45:25 PM	3/3/2015 11:31:56 PM
3	Table	SQL Saturday 374 Vienna		2/28/2015 6:28:23 PM	2/28/2015 6:28:23 PM	2/19/2015 9:03:43 PM
4	Table	CloudnEnterprise_Symbols_Public_v2.02		2/1/2015 12:43:29 PM	2/1/2015 12:43:29 PM	2/1/2015 12:43:29 PM
5	Table	TechDays Hong Kong 2015		2/12/2015 4:14:33 PM	2/12/2015 4:14:33 PM	2/1/2015 12:20:56 PM

## 2- Fetch Only Files with .txt extension and name similar to "amp"

```

1 let
2 Source = Folder.Files("C:\Users\Reza\Dropbox\Speaking"),
3 Sorted=Table.Sort(Source,{"Date created", Order.Descending}),
4 Filtered=Table.SelectRows(Sorted, each [Extension]=".txt" and Text.Contains([Name],"amp"))
5
6 in
7 Filtered

```

## Result:

	Content	Name	Extension	Date accessed	Date modified
1	Binary	M sample.txt	.txt	7/27/2015 11:14:38 PM	2/10/2015
2	Binary	M sample.txt	.txt	2/19/2015 9:23:56 PM	2/10/2015
3	Binary	M sample.txt	.txt	12/13/2014 10:15:07 AM	2/10/2015
4	Binary	M sample.txt	.txt	12/13/2014 10:15:04 AM	2/10/2015
5	Binary	M sample.txt	.txt	12/13/2014 10:09:55 AM	2/10/2015

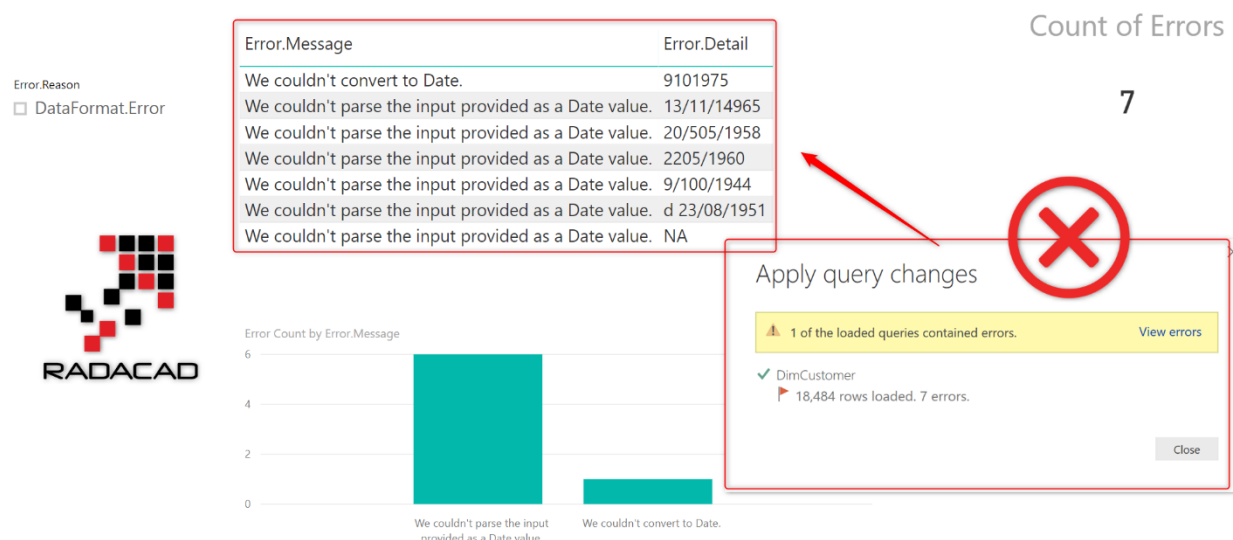
## Part IV: Dealing with Errors



# Exception Reporting in Power BI: Catch the Error Rows in Power Query

Posted by [Reza Rad](#) on Nov 23, 2018

## Exception Report: Error Rows

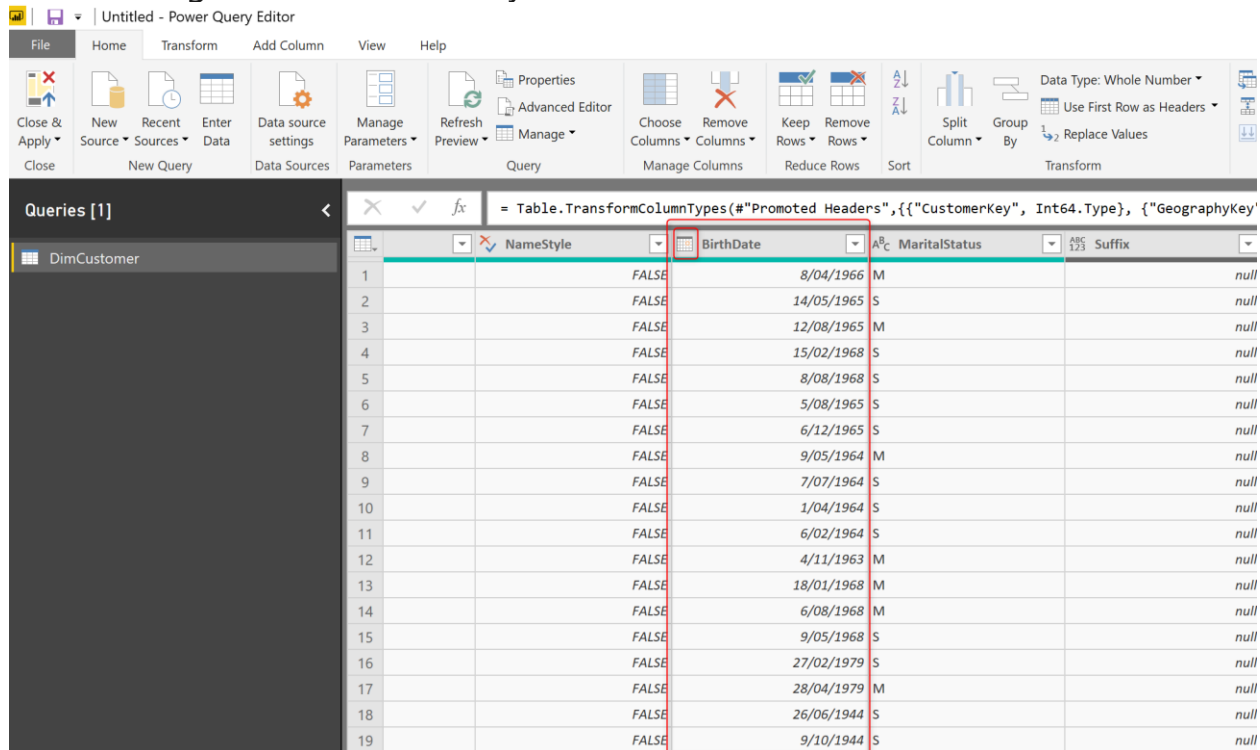


To build a robust BI system, you need to cater for errors and handle errors carefully. If you build a reporting solution that the refresh of that fails everytime an error occurs, it is not a robust system. Errors can happen by many reasons. In this post, I'll show you a way to catch potential errors in Power Query and how to build an exception report page to visualize the error rows for further investigation. The method that you learn here will save your model from failing at the time of refresh. Means you get the dataset updated, and you can catch any rows caused the error in an exception report page. To learn more about Power BI, read [Power BI book from Rookie to Rock Star](#).

### Sample Dataset

I will use a sample Excel file as a data source which has 18,484 customer rows in it. In the sample Dataset, we have a BirthDate field beside all other fields,

which supposed to have a date value in it. Here is what the data looks like when I bring it into Power Query:

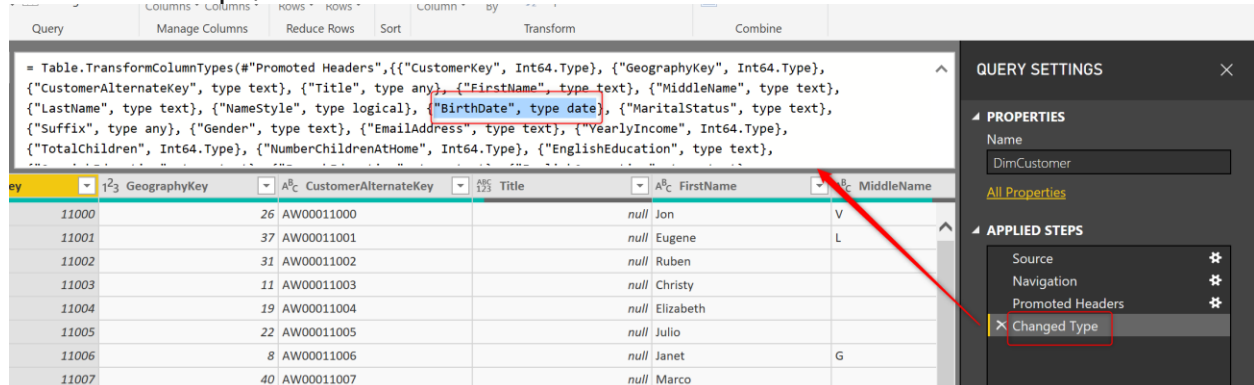


Power Query Editor window showing a table with the following columns: NameStyle, BirthDate, MaritalStatus, and Suffix. The BirthDate column is highlighted with a red box, indicating it is the focus of the discussion.

	NameStyle	BirthDate	MaritalStatus	Suffix
1	FALSE	8/04/1966	M	null
2	FALSE	14/05/1965	S	null
3	FALSE	12/08/1965	M	null
4	FALSE	15/02/1968	S	null
5	FALSE	8/08/1968	S	null
6	FALSE	5/08/1965	S	null
7	FALSE	6/12/1965	S	null
8	FALSE	9/05/1964	M	null
9	FALSE	7/07/1964	S	null
10	FALSE	1/04/1964	S	null
11	FALSE	6/02/1964	S	null
12	FALSE	4/11/1963	M	null
13	FALSE	18/01/1968	M	null
14	FALSE	6/08/1968	M	null
15	FALSE	9/05/1968	S	null
16	FALSE	27/02/1979	S	null
17	FALSE	28/04/1979	M	null
18	FALSE	26/06/1944	S	null
19	FALSE	9/10/1944	S	null

## Error Happens

When I get this dataset in my Power Query Editor window (as you see in the above screenshot), Power Query automatically converts the data type of the BirthDate column to Date. You can see this automatic data type conversion in the list of Steps;

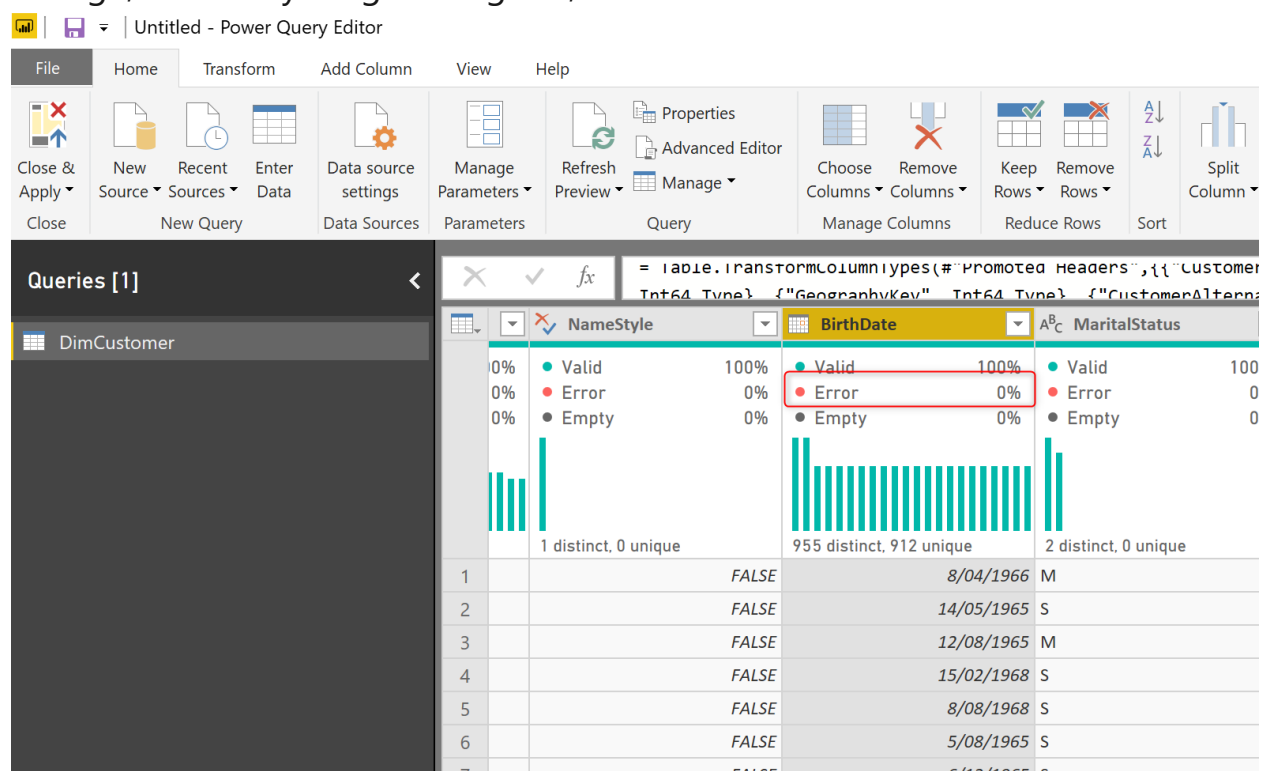


Power Query Editor window showing the 'Applied Steps' list on the right. The 'Changed Type' step is highlighted with a red box, indicating the automatic conversion of the BirthDate column to Date.

Step	Applied Step
1	Source
2	Navigation
3	Promoted Headers
4	Changed Type

Of course, you can turn off this automatic data type detection of Power Query, but that is not my point. I want the dataset to fail to show you how to deal with it. Errors happen in Power Query in the real world, and I'm here to show you how to find them.

As you can see in the Power Query Editor, I see no errors for this data type change, and everything looks great;




The screenshot shows the Power Query Editor interface. The 'Queries [1]' pane on the left lists 'DimCustomer'. The main area displays the query's data, with columns 'NameStyle', 'BirthDate', and 'MaritalStatus'. The 'BirthDate' column is highlighted, and its data type summary shows 0% Valid, 0% Error, and 0% Empty. The data table below shows rows with 'NameStyle' as FALSE and 'BirthDate' as dates.

	NameStyle	BirthDate	MaritalStatus
1	FALSE	8/04/1966	M
2	FALSE	14/05/1965	S
3	FALSE	12/08/1965	M
4	FALSE	15/02/1968	S
5	FALSE	8/08/1968	S
6	FALSE	5/08/1965	S
7	FALSE	6/12/1965	S

Now I load this dataset into Power BI, by using Close and Apply in the Query Editor window, and I expect everything to load successfully, however, this is coming out of the blue!




## Apply query changes

 1 of the loaded queries contained errors.

[View errors](#)

✓ DimCustomer

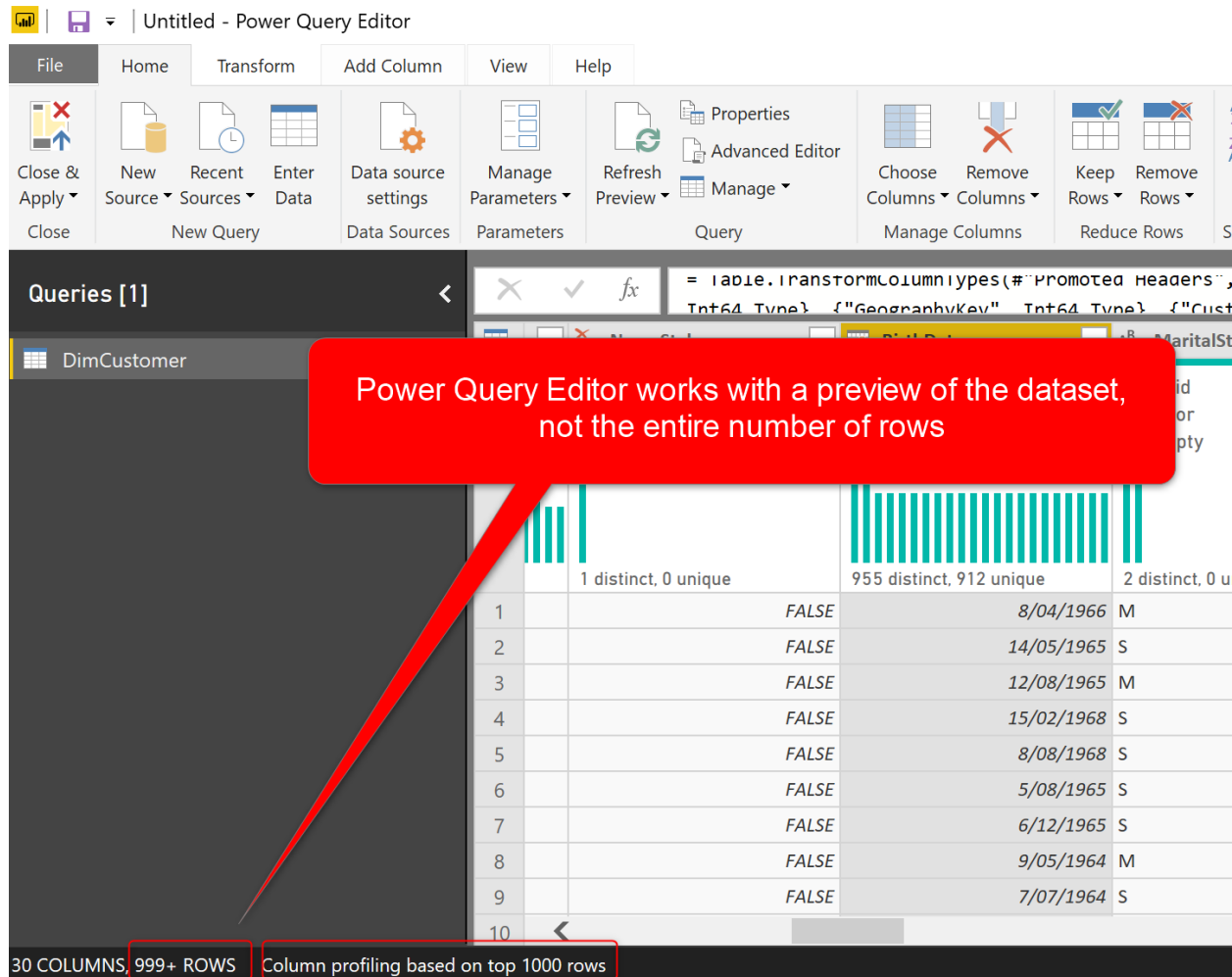
 18,484 rows loaded. 7 errors.

Close

Does this sound familiar? Yes, if you have worked with Power BI for a while, you might have experienced it. No errors in Power Query Editor, but when we load the data into Power BI, there are errors! How is that possible? Let's first find out why this happens.

### **Why didn't Power Query Editor Catch the Error?**

Power Query Editor always work with a preview of the dataset, the size of the preview depends on how many columns you have, sometimes it is 1000 rows, sometimes 200 rows. If you click on a Query in the Power Query editor window, you can see this stated down below in the status bar;



Power Query Editor works with a preview of the dataset, not the entire number of rows

1 distinct, 0 unique	955 distinct, 912 unique	2 distinct, 0 u
1	FALSE	8/04/1966 M
2	FALSE	14/05/1965 S
3	FALSE	12/08/1965 M
4	FALSE	15/02/1968 S
5	FALSE	8/08/1968 S
6	FALSE	5/08/1965 S
7	FALSE	6/12/1965 S
8	FALSE	9/05/1964 M
9	FALSE	7/07/1964 S
10		

30 COLUMNS 999+ ROWS Column profiling based on top 1000 rows

The reason for Power Query to use the preview dataset is mainly because of speeding up the transformation development process. Imagine if you have a table with 10 million rows, every single transformation that you want to apply on that dataset would take a long time, and you have to wait for it before you start doing the next step. The wait for the response each time will slow down your development process. This is the reason why working on a preview on the dataset is a preferred option. You can apply all transformations you want on the preview, and when you are happy with it, then apply it to the entire dataset. Usually, the first 1000 rows or the first 200 rows are a good sample of the entire dataset, and you can expect to see most of the data challenges there. Usually, not always of course.

How will the transformation be applied to the entire dataset then? When you load the data into Power BI. That means when you click on Close and APPLY in the Power Query Editor window. That APPLY means apply those transformations now on the entire dataset. That is the reason, why the load process may take longer especially if the dataset is big.

*Power Query Editor always works with a preview of the data, to make the development process fast. When you load the data in Power BI, transformations will be applied on the entire dataset.*

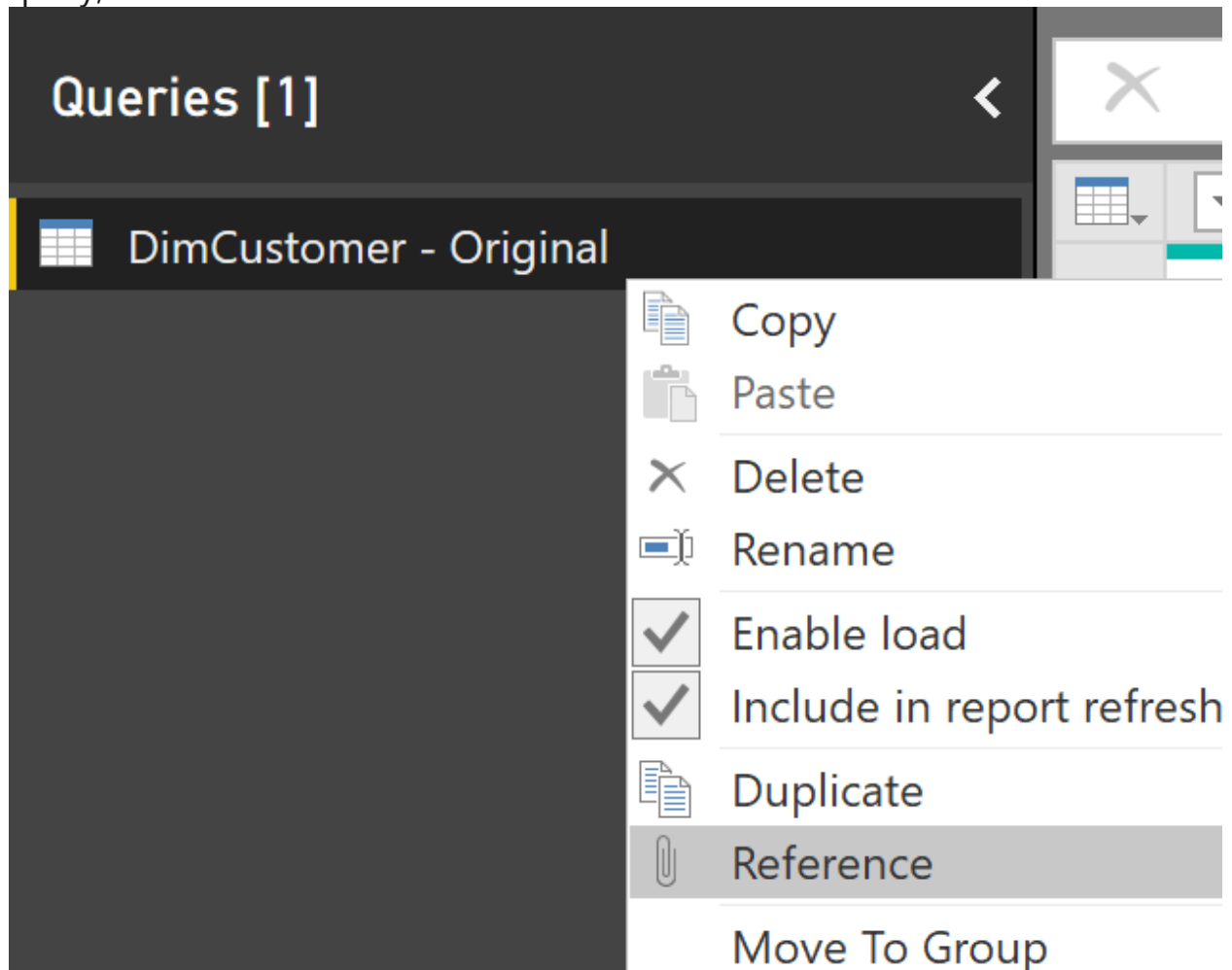
Now that you know how Power Query Editor deals with the preview of the data, you can guess why the error above happened? The reason is; The preview of the data (which was about 1000 rows) had no issues with the transformations applied (in this case automatic data type change to Date for the BirthDate column). However, the entire dataset (which is about 18K rows) have problems with that transformation! When you see the error above in the Power BI Desktop, then you can click on View errors and go to Power Query editor and see those rows, and deal with them somehow, and fix it. However, that is not enough.

*What if the error doesn't happen in Power BI Desktop, but happens in a scheduled refresh in the Power BI Service?*

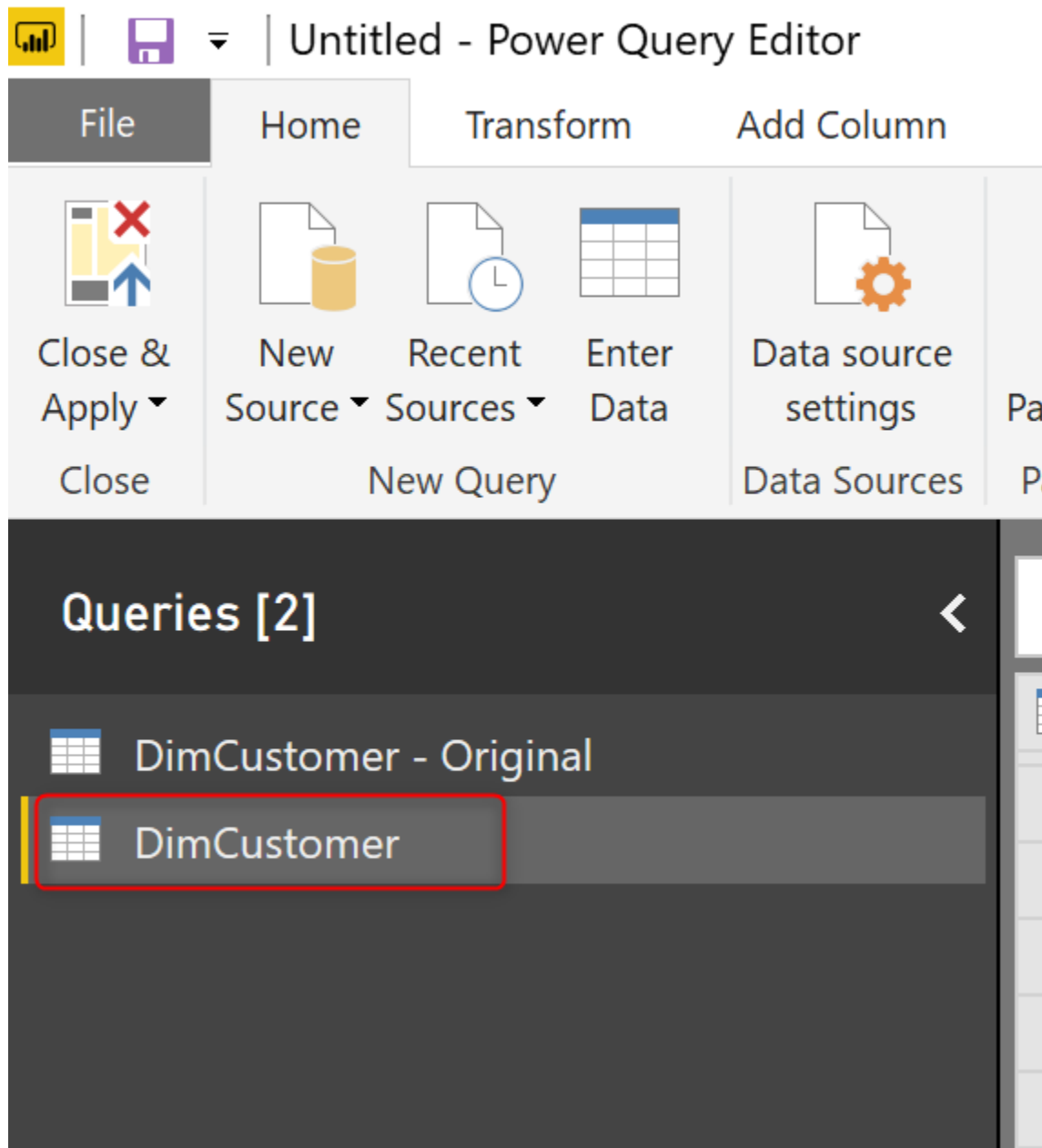
True! Fixing errors in Power BI Desktop is easy, but consider that the error didn't happen in the desktop too, and you got your Power BI report published to the website, and scheduled it to refresh. Then the next day you see the report failed to refresh with an error! You have to learn how to deal with the error rows beforehand before it cause the scheduled refresh to fail. Let's see how to deal with it then.

## Dealing with Errors: Catching the Error Rows

To deal with errors, you have to catch the error before it loads into Power BI. One way to do it is to create two references of the same table, one as the final query, and the other one as Error Rows.



in the screenshot above, I renamed the DimCustomer table to DimCustomer – Original and then created a Reference from it. If you would like to learn what Reference is, [read my article about Reference vs. Duplicate here](#). The new referenced query can be called as DimCustomer. This would be the clean query with no errors (we will remove errors from it in the next step);



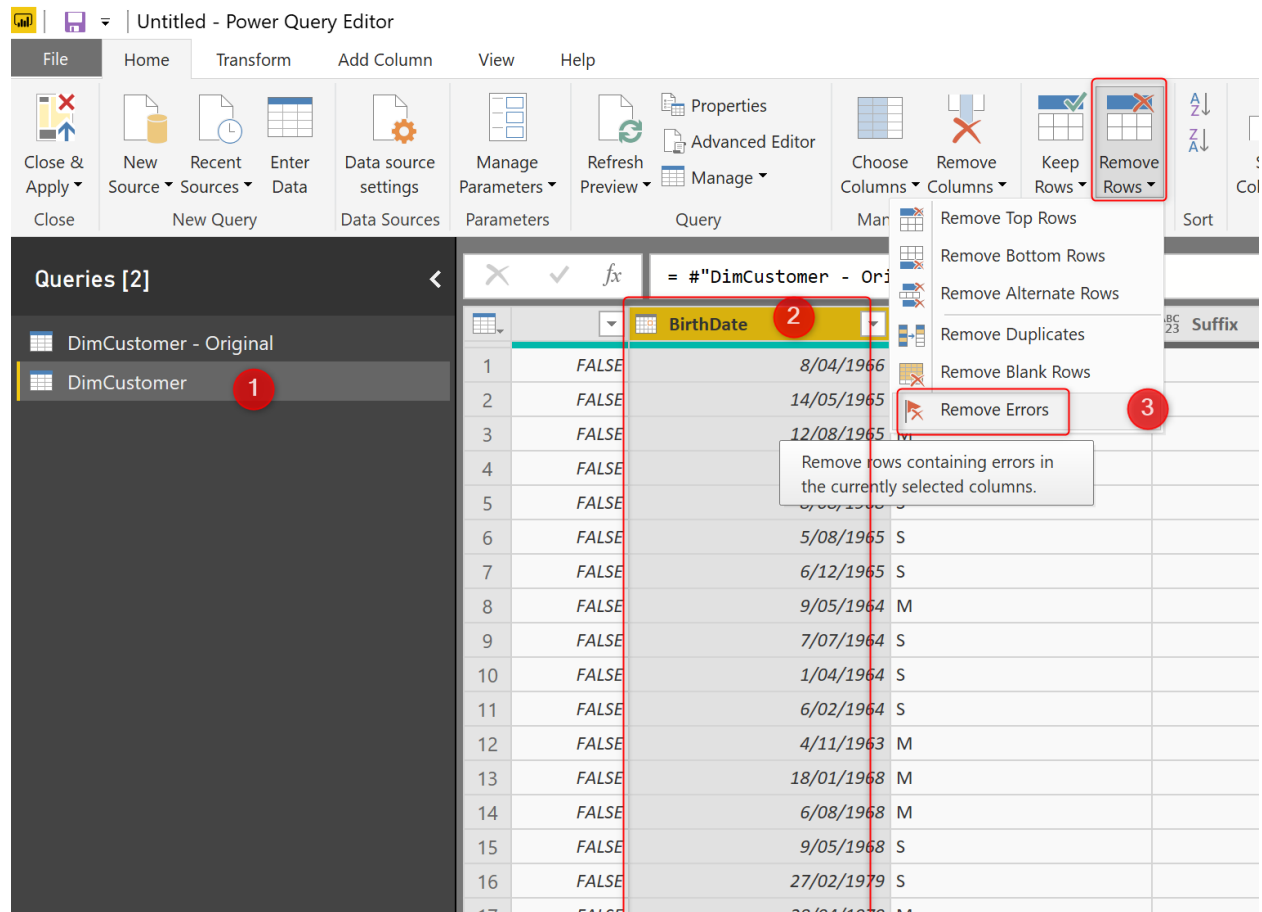
The new table is the table that would be clean with no errors, and we can use it in the report. Let's clean this from any errors

### **Remove Errors from the Table loading into Power BI**

As DimCustomer would be the final query for us, I want to remove errors from it. Removing errors is a simple option in the Home tab, under Reduce Rows ->

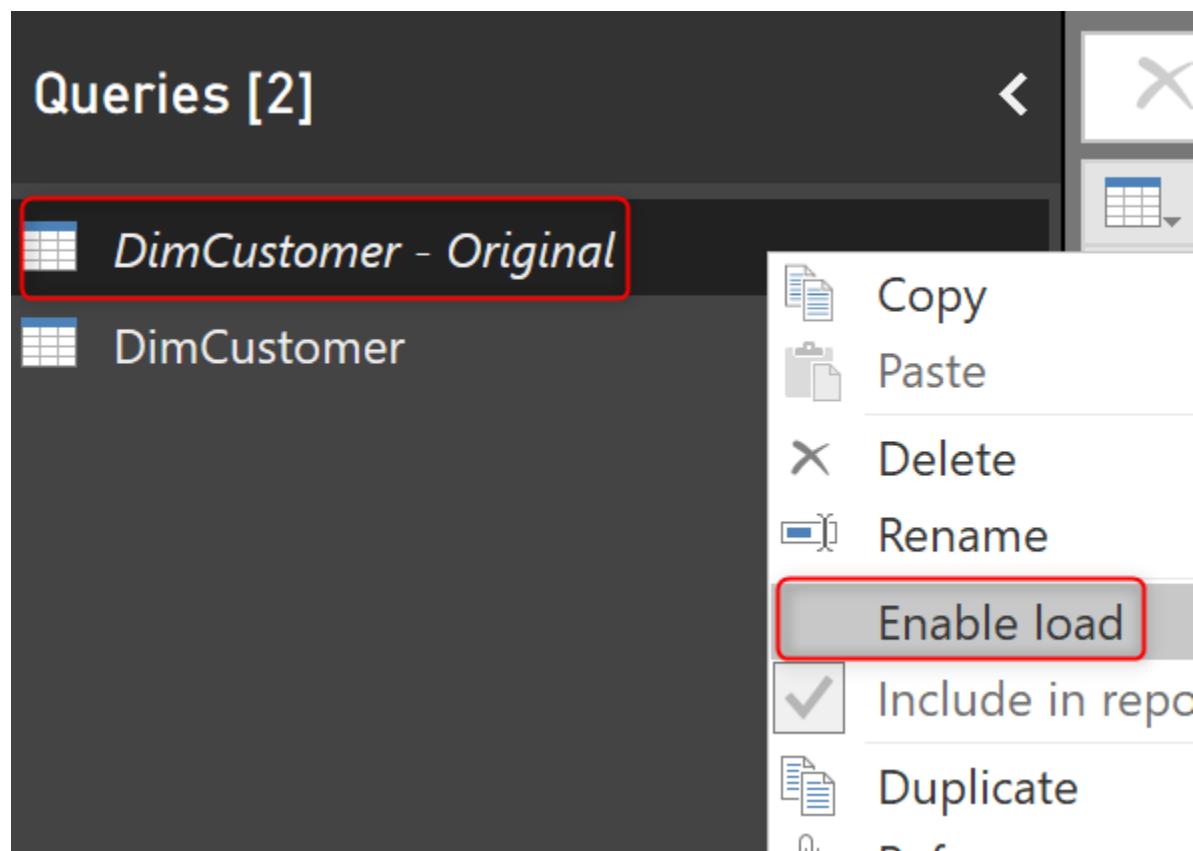


Remove Rows -> Remove Errors. Make sure that you select the BirthDate column before that.



You can also do this for all columns if you want; by selecting all columns and then using Remove Errors. This post is just a sample on one column and can be extended to all for sure.

Remove Errors will be a step in the data transformations step, and it means that when you click on APPLY, it will apply on the entire dataset, so as a result, when the data type change cause an error, the next step after that which is Remove Errors, will wipe the rows that caused the error. But the DimCustomer – Original still may cause the error, so we have to uncheck the Enable Load of that query.



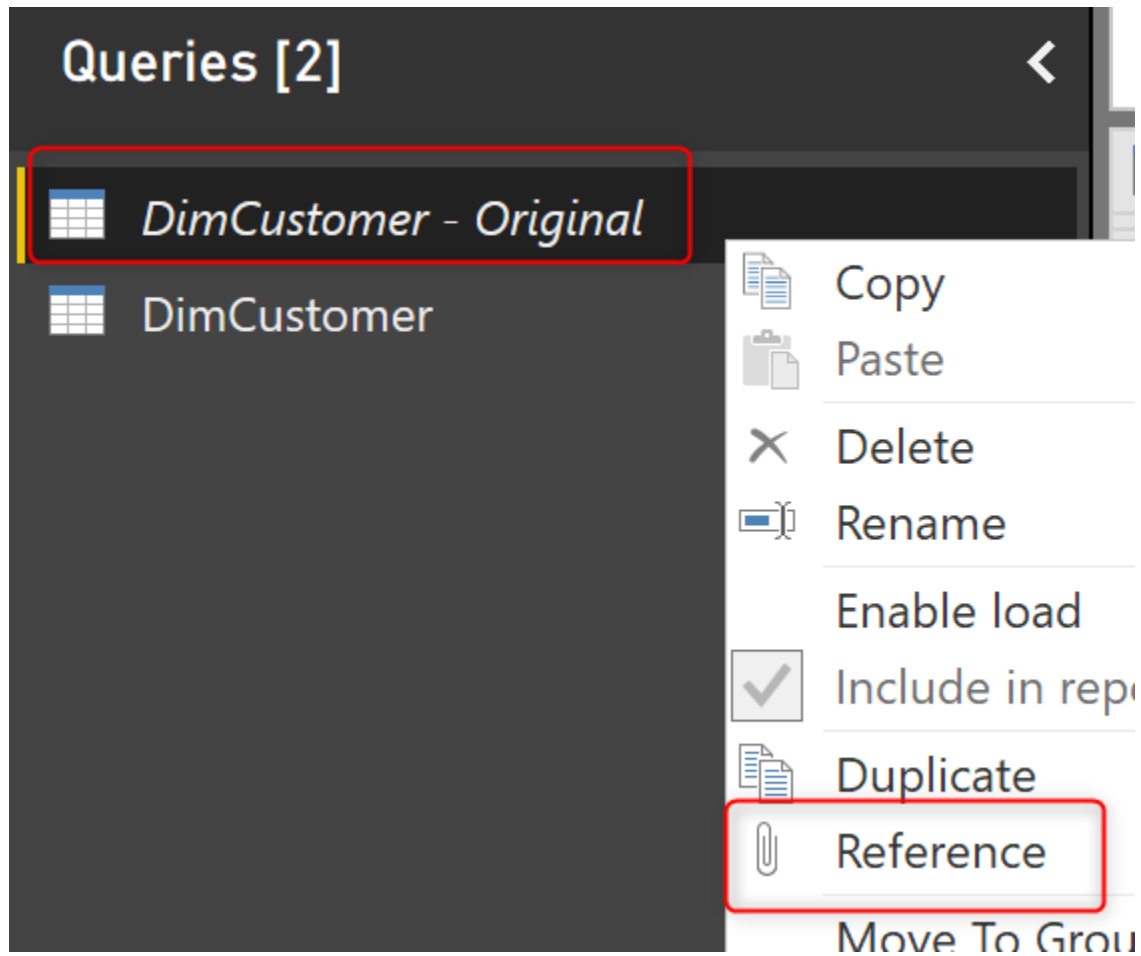
Now we have successfully removed the errors, and if we load the data into Power BI. There would be no errors happening.

But wait! What about those error rows? How can we catch them? We need to catch those rows and investigate what happened and think about an action plan to fix them, right? So, we do need another query reference from the original query, but to keep the error rows.

### **Keep Errors in the Exception Table**

Similar to the Remove Errors option, there is also an option to Keep Errors. If you have seen this option before, you may have wondered what the use of such a thing is? Well, here is the exact use case scenario. Keep Errors will help to catch the error rows in an exception table.

Create another reference from the DimCustomer – Original.



Rename this new query as DimCustomer Error Rows. For this query, we have to Keep Errors, which can be found close to where the Remove Errors is, but under Keep Rows.

Untitled - Power Query Editor

File Home Transform Add Column View Help

Close & Apply New Source Recent Enter Data Data source settings Manage Parameters Refresh Preview Properties Advanced Editor Manage Choose Remove Keep Rows Remove Columns Columns Rows Rows

Queries [3]

- DimCustomer - Original
- DimCustomer
- DimCustomer - Error Rows** 1

BirthDate 2

Keep Errors 3

Keep only rows containing errors in the currently selected columns.

		BirthDate	
1	FALSE	8/04/1966	
2	FALSE	14/05/1965	S
3	FALSE		
4	FALSE		
5	FALSE	8/08/1968	S
6	FALSE	5/08/1965	S
7	FALSE	6/12/1965	S
8	FALSE	9/05/1964	M
9	FALSE	7/07/1964	S
10	FALSE	1/04/1964	S
11	FALSE	6/02/1964	S
12	FALSE	4/11/1963	M

Now this table would only keep rows that cause an error. Here is a sample set;

Untitled - Power Query Editor

File Home Transform Add Column View Help

Close & Apply New Source Recent Enter Data Data source settings Manage Parameters Refresh Preview Properties Advanced Editor Manage Choose Remove Keep Rows Remove Rows Sort Split Column Group By

Queries [3]

- DimCustomer - Original
- DimCustomer
- DimCustomer - Error Rows**

`= Table.SelectRowsWithErrors(Source, {"BirthDate"})`

	NameStyle	BirthDate	AB <sub>C</sub> MaritalStatus
1		FALSE Error	S
2		FALSE Error	M
3		FALSE Error	M
4		FALSE Error	M
5		FALSE Error	S
6		FALSE Error	S
7		FALSE Error	M

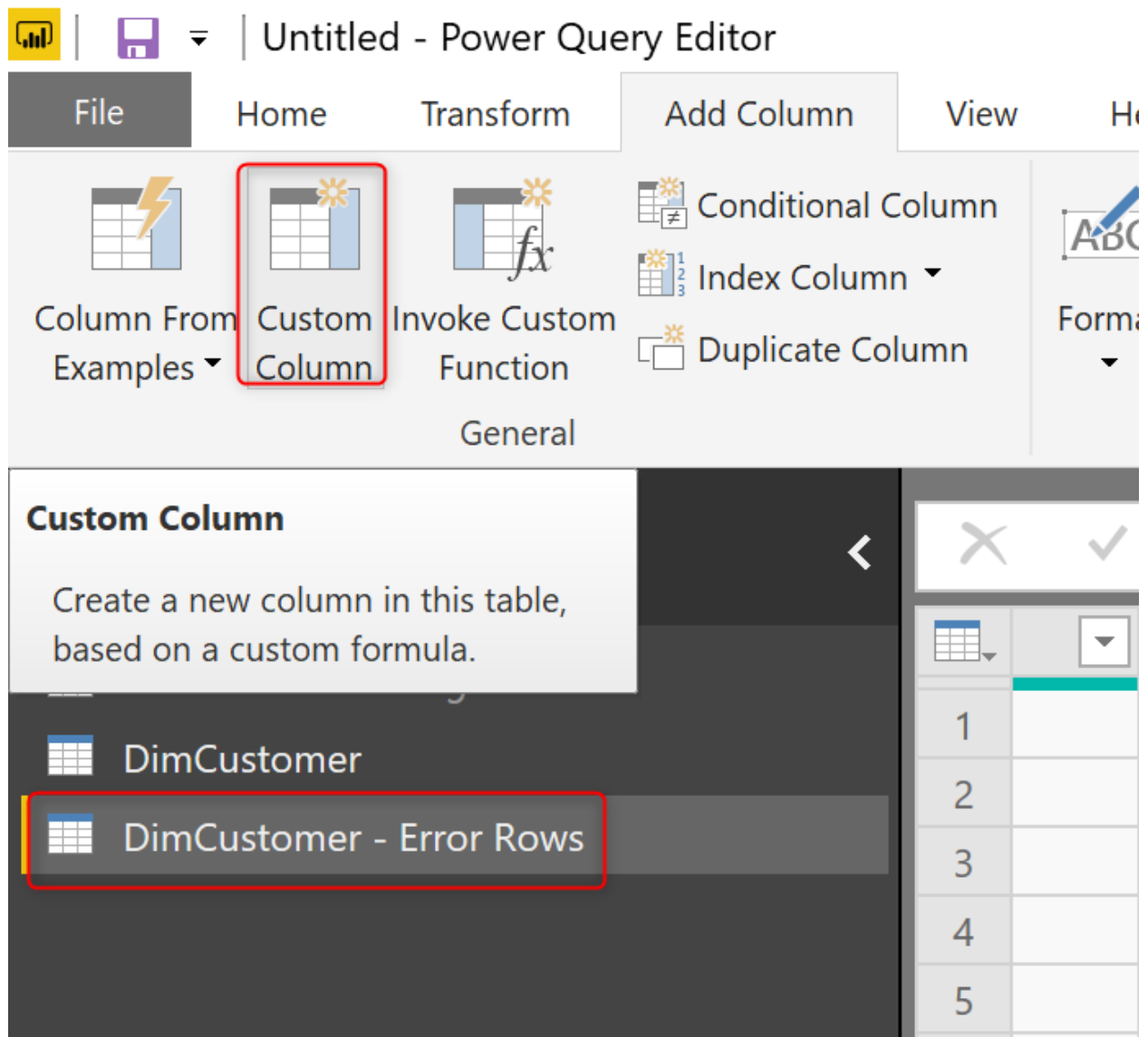
This is not the end of the story. If you load this new table of DimCustomer – Error Rows in Power BI, you will end up with the same error again. Why? Well,

because this query is going to return error rows! You need to remove the Error occurred from this dataset.

### **Getting Error Details**

If you remove the error column from the exception table we have created, then you would have no details about the error happened, and it would be hard to track it back and troubleshoot. The best would be catching the error details. The error message and the value that caused the error are important details that you don't want to miss. Follow steps below to get that information.

In the Error Rows table, add a Custom Column.



In the Custom Column editor, write ***try*** and then after a space, the name of the field that caused the error. In our example: BirthDate;

1 try [BirthDate]

## Custom Column

New column name

Custom

Custom column formula:

= try [BirthDate]

Available columns:

FirstName  
 MiddleName  
 LastName  
 NameStyle  
 BirthDate  
 MaritalStatus  
 Suffix  
 Gender

&lt;&lt; Insert

[Learn about Power BI Desktop formulas](#)

✓ No syntax errors have been detected.

OK

Cancel

**try** (all lowercase), is a keyword in M that will catch the error details. Instead of returning just an error, it will return a record containing the error details such as the source value and the error message. Below screenshot shows how the output of **try** would come;

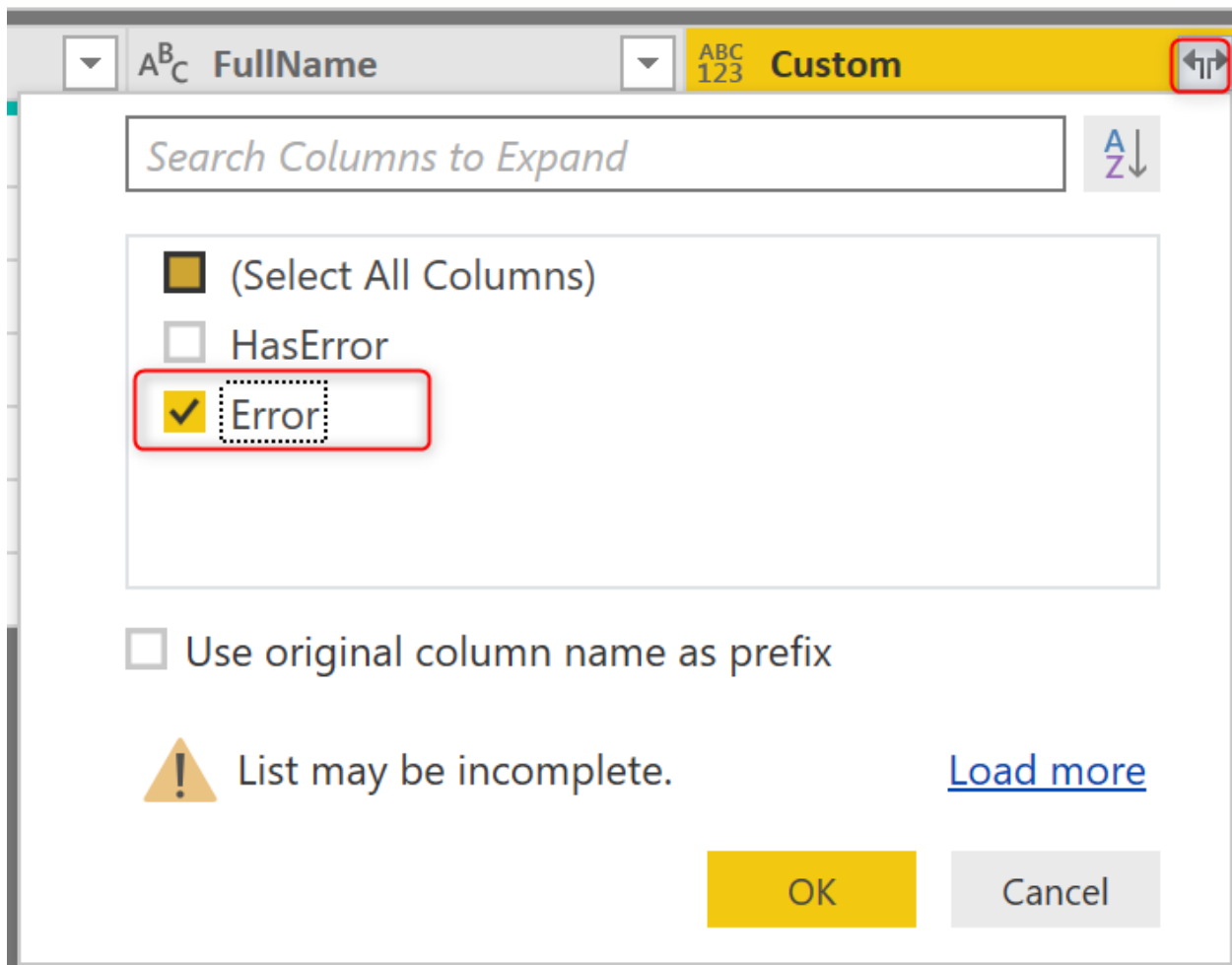
FROM TEXT FROM NUMBER FROM DATE & TIME

fx = Table.AddColumn("#Kept Errors", "Custom", each try [BirthDate])

	dressLine2	Phone	DateFirstPurchase	CommuteDistance	FullName	Custom
1		161-555-0125	19/09/2007	10+ Miles	John White	Record
2		334-555-0173	23/08/2007	10+ Miles	Nathan Hughes	Record
3		178-555-0116	19/05/2006	2-5 Miles	Logan Turner	Record
4		1 (11) 500 555-0166	11/09/2007	2-5 Miles	Morgan Sanchez	Record
5		1 (11) 500 555-0190	7/06/2006	5-10 Miles	Dustin Goldstein	Record
6		391-555-0176	28/06/2006	0-1 Miles	Hailey Cox	Record
7		1 (11) 500 555-0118	13/10/2007	0-1 Miles	Ian Bell	Record

HasError TRUE  
Error Record

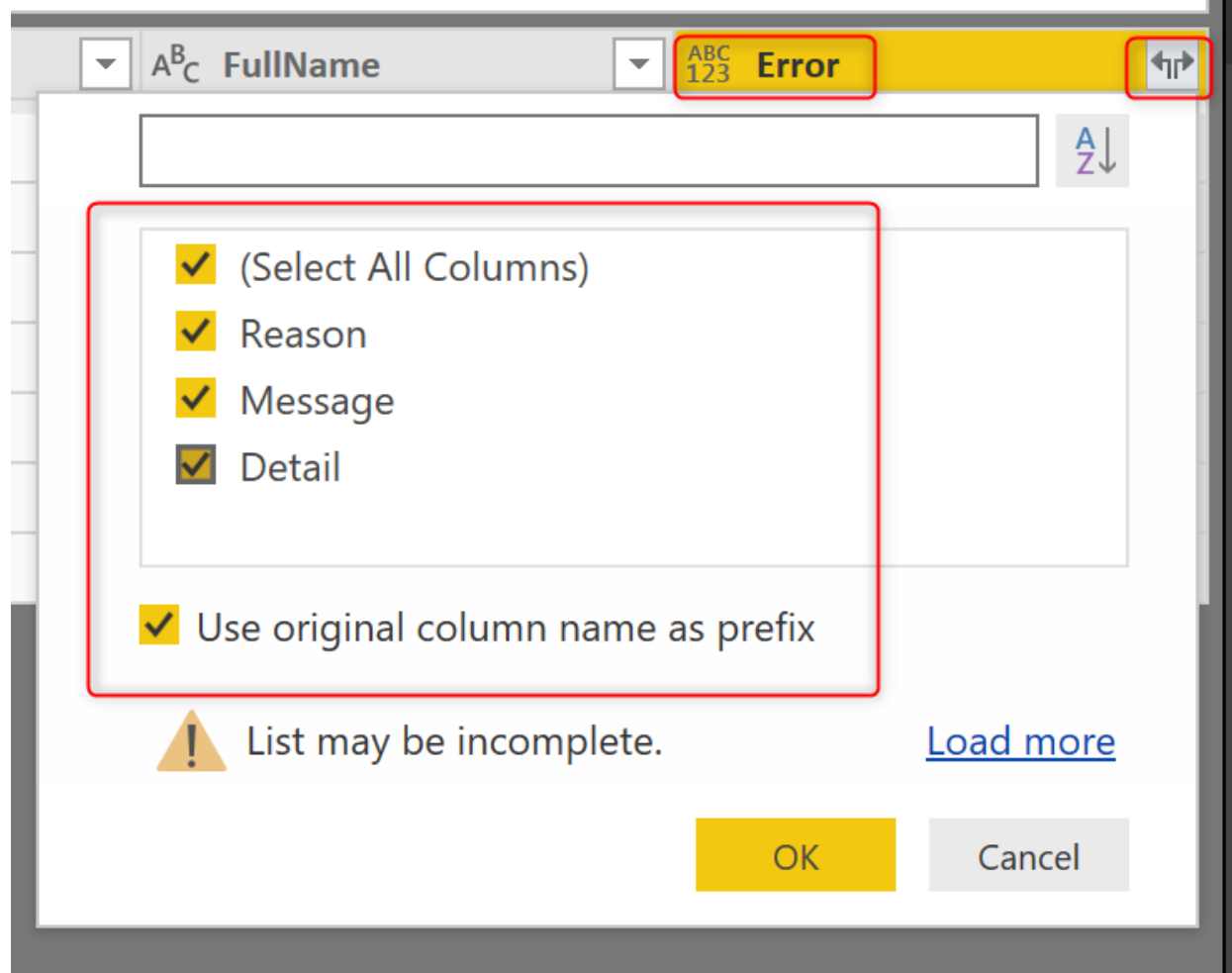
The Record output of the "try" will have two fields; HasError (which we already know it is going to be true), and the Error. The Error is another record with more details. Click on Expand on the Custom column, and select Error.



In the output column named Error, click on Expand again and this time select all columns;



r } )



It is good to have the original column name as a prefix because then you would know that these are error detail columns.

Now you would get the full details of the error as below;

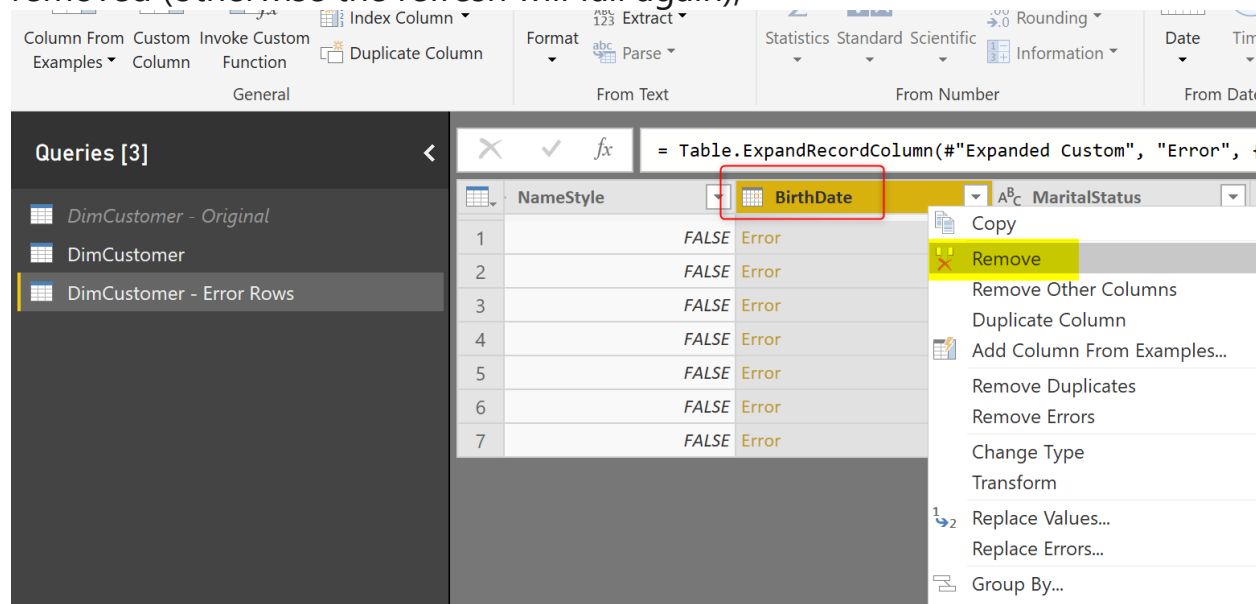
'Expanded Custom', "Error", {"Reason", "Message", "Detail"}, {"Error.Reason",

ABC 123 Error.Reason	ABC 123 Error.Message	ABC 123 Error.Detail
DataFormat.Error	We couldn't parse the input provided as a Date value.	9/100/1944
DataFormat.Error	We couldn't parse the input provided as a Date value.	d 23/08/1951
DataFormat.Error	We couldn't parse the input provided as a Date value.	13/11/14965
DataFormat.Error	We couldn't parse the input provided as a Date value.	2205/1960
DataFormat.Error	We couldn't convert to Date.	9101975
DataFormat.Error	We couldn't parse the input provided as a Date value.	20/505/1958
DataFormat.Error	We couldn't parse the input provided as a Date value.	NA

The information above is your most valuable asset for the exception reporting.

## Remove Error Column

Now the last step before loading the data into Power BI is to remove the column that causes the Error. In our example; the BirthDate Column should be removed (otherwise the refresh will fail again);



## Exception Report

You can now load the data into Power BI. You will have two tables; DimCustomer, and DimCustomer – Error Rows. DimCustomer is the table that you can use for your normal reporting. DimCustomer – Error Rows is the table that you can use for exception reporting. The exception report is the report that can be used for troubleshooting and will list all the errors to users for further investigation. Make sure that there is no relationship between these two tables.

Here is a sample report visual I created that shows the errors;

# Exception Report: Error Rows



## Summary

Errors happen, and you have to deal with them. Instead of waiting for the error to happen and then finding it a month after it caused, it is better to catch them as soon as they happen. In this article, you learned a way to deal with error rows. In RADACAD we always create an exception report for Power BI reports. That way, we are always sure that the refresh won't fail because of the error, and we would also have a place for investigating the errors, which is called the Exception Report. Do you have an exception report? If not, go and create one, If yes, tell us about your experience down below in the comments.

# Flawless Date Conversion in Power Query

Posted by [Reza Rad](#) on Jan 6, 2017

	Date1	Date2	Date3	Date4
1	1/1/2017	1/1/2017	1/1/2017	1/1/2017
2	1/2/2017	1/2/2017	1/2/2017	2/1/2017
3	1/3/2017	1/3/2017	1/3/2017	3/1/2017
4	1/4/2017	1/4/2017	1/4/2017	4/1/2017
5	1/5/2017	1/5/2017	1/5/2017	5/1/2017
6	1/6/2017	1/6/2017	1/6/2017	6/1/2017
7	1/7/2017	1/7/2017	1/7/2017	7/1/2017
8	1/8/2017	1/8/2017	1/8/2017	8/1/2017
9	1/9/2017	1/9/2017	1/9/2017	9/1/2017
10	1/10/2017	1/10/2017	1/10/2017	10/1/2017
11	1/11/2017	1/11/2017	1/11/2017	11/1/2017
12	1/12/2017	1/12/2017	1/12/2017	12/1/2017
13	1/13/2017	1/13/2017	1/13/2017	Error
14	1/14/2017	1/14/2017	1/14/2017	Error
15	1/15/2017	1/15/2017	1/15/2017	Error
16	1/16/2017	1/16/2017	1/16/2017	Error
17	1/17/2017	1/17/2017	1/17/2017	Error
18	1/18/2017	1/18/2017	1/18/2017	Error
19	1/19/2017	1/19/2017	1/19/2017	Error
20	1/20/2017	1/20/2017	1/20/2017	Error
21	1/21/2017	1/21/2017	1/21/2017	Error
22	1/22/2017	1/22/2017	1/22/2017	Error
23	1/23/2017	1/23/2017	1/23/2017	Error
24	1/24/2017	1/24/2017	1/24/2017	Error
25	1/25/2017	1/25/2017	1/25/2017	Error
26	1/26/2017	1/26/2017	1/26/2017	Error
27	1/27/2017	1/27/2017	1/27/2017	Error
28	1/28/2017	1/28/2017	1/28/2017	Error
29	1/29/2017	1/29/2017	1/29/2017	Error
30	1/30/2017	1/30/2017	1/30/2017	Error
31	1/31/2017	1/31/2017	1/31/2017	Error

Date Conversion is one of the simplest conversions in Power Query, however, depends on locale on the system that you are working with Date Conversion might return a different result. In this post, I'll show you an example of an

issue with date conversion and how to resolve it with Locale. In this post, you'll learn that Power Query date conversion is dependent on the system that this conversion happens on, and can be fixed to a specific format. If you want to learn more about Power BI; Read [Power BI online book; from Rookie to Rock Star](#).

### **Prerequisite**

The sample data set for this post is here: [book1](#)

## **Different Formats of Date**

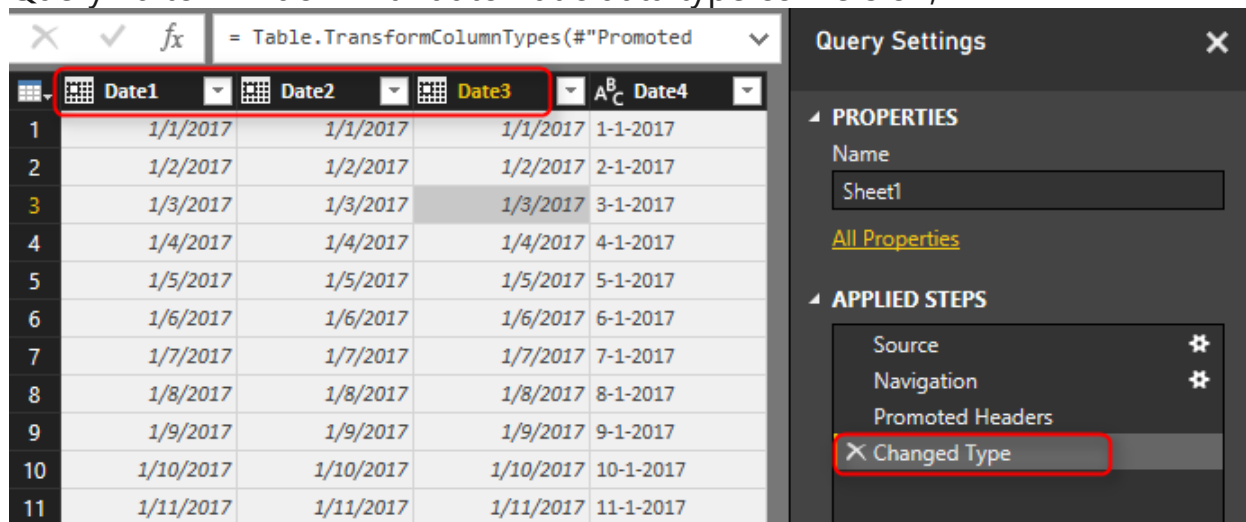
Most of the countries use YMD format. However some of them use MDY or DMY more frequently. [This Wikipedia page](#) explains different formats of date in each country. Dataset below have different formats in it;

Date1	Date2	Date3	Date4
2017-1-1	2017/1/1	1-1-2017	1-1-2017
2017-1-2	2017/1/2	1-2-2017	2-1-2017
2017-1-3	2017/1/3	1-3-2017	3-1-2017
2017-1-4	2017/1/4	1-4-2017	4-1-2017
2017-1-5	2017/1/5	1-5-2017	5-1-2017
2017-1-6	2017/1/6	1-6-2017	6-1-2017
2017-1-7	2017/1/7	1-7-2017	7-1-2017
2017-1-8	2017/1/8	1-8-2017	8-1-2017
2017-1-9	2017/1/9	1-9-2017	9-1-2017
2017-1-10	2017/1/10	1-10-2017	10-1-2017
2017-1-11	2017/1/11	1-11-2017	11-1-2017
2017-1-12	2017/1/12	1-12-2017	12-1-2017
2017-1-13	2017/1/13	1-13-2017	13-1-2017
2017-1-14	2017/1/14	1-14-2017	14-1-2017
2017-1-15	2017/1/15	1-15-2017	15-1-2017
2017-1-16	2017/1/16	1-16-2017	16-1-2017
2017-1-17	2017/1/17	1-17-2017	17-1-2017
2017-1-18	2017/1/18	1-18-2017	18-1-2017
2017-1-19	2017/1/19	1-19-2017	19-1-2017
2017-1-20	2017/1/20	1-20-2017	20-1-2017
2017-1-21	2017/1/21	1-21-2017	21-1-2017
2017-1-22	2017/1/22	1-22-2017	22-1-2017
2017-1-23	2017/1/23	1-23-2017	23-1-2017
2017-1-24	2017/1/24	1-24-2017	24-1-2017
2017-1-25	2017/1/25	1-25-2017	25-1-2017
2017-1-26	2017/1/26	1-26-2017	26-1-2017
2017-1-27	2017/1/27	1-27-2017	27-1-2017
2017-1-28	2017/1/28	1-28-2017	28-1-2017
2017-1-29	2017/1/29	1-29-2017	29-1-2017
2017-1-30	2017/1/30	1-30-2017	30-1-2017
2017-1-31	2017/1/31	1-31-2017	31-1-2017

In the above table, first two columns (Date1, and Date2) are in YYYY-MM-DD and YYYY/MM/DD format, which is the most common format of the date. Date3 is MM-DD-YYYY format (very common in the USA and some other countries), and Date4 is DD-MM-YYYY format, which is most common in New Zealand, Australia, and some other countries. Now let's see what happens if we load this data into Power BI.

## Automatic Type Conversion

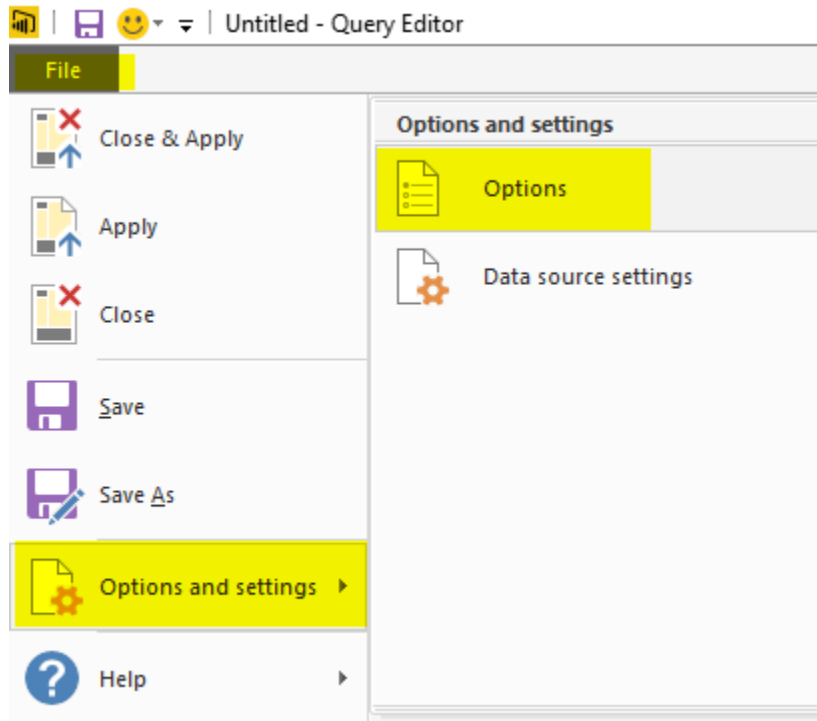
Power BI leverages automatic data type conversion. This automatic action sometimes is useful, sometimes not! If you open a Power BI Desktop file and get data from the specified data set. In Query Editor you will see the data types converted automatically at some level. Here is the data loaded into the Query Editor window with automatic data type conversion;



	Date1	Date2	Date3	Date4
1	1/1/2017	1/1/2017	1/1/2017	1-1-2017
2	1/2/2017	1/2/2017	1/2/2017	2-1-2017
3	1/3/2017	1/3/2017	1/3/2017	3-1-2017
4	1/4/2017	1/4/2017	1/4/2017	4-1-2017
5	1/5/2017	1/5/2017	1/5/2017	5-1-2017
6	1/6/2017	1/6/2017	1/6/2017	6-1-2017
7	1/7/2017	1/7/2017	1/7/2017	7-1-2017
8	1/8/2017	1/8/2017	1/8/2017	8-1-2017
9	1/9/2017	1/9/2017	1/9/2017	9-1-2017
10	1/10/2017	1/10/2017	1/10/2017	10-1-2017
11	1/11/2017	1/11/2017	1/11/2017	11-1-2017

You can see the Changed Type step that applied automatically and converted the first three columns in the data set to the data type of Date and left the Date4 as data type Text. Automatic Date type conversion understands characters like / or -, and apply conversion correctly in both cases (Date1, and Date2)

\* If you are running this example on your machine you might see a different result because Power Query uses the locale of the machine to do data type conversion. Locale of my machine is US format, so it understands the format of USA and converts it automatically. If you have different locale, the conversion might result differently. We will go through that in a second. If the automatic data type conversion is not something you want, you can remove that step. Or if you want to disable the automatic type conversion, Go to File, Options, and Settings, then Options.



In the Options window, under Current File, Data Load section. You can enable or disable the automatic data type detection if you wish to. (In this example I'll keep it enabled)



## Options

### GLOBAL

Data Load  
Query Editor  
DirectQuery  
R Scripting  
Security  
Privacy  
Updates  
Usage Data  
Diagnostics  
Preview Features  
Auto Recovery

### CURRENT FILE

Data Load  
Regional Settings  
Privacy  
Auto Recovery

#### Type Detection

☒ Automatically detect column types and headers for unstructured sources

#### Relationships

☒ Import relationships from data sources ⓘ  
☐ Update relationships when refreshing queries ⓘ  
☒ Autodetect new relationships after data is loaded ⓘ

#### Time Intelligence

☒ Auto Date/Time ⓘ

#### Background Data

☒ Allow data preview to download in the background

#### Parallel Loading of Tables

☒ Enable parallel loading of tables ⓘ

OK

Cancel

You can also check the Locale of your current machine in the Regional Settings section of Options Window;

## Options

### GLOBAL

Data Load  
Query Editor  
DirectQuery  
R Scripting  
Security  
Privacy  
Updates  
Usage Data  
Diagnostics  
Preview Features  
Auto Recovery

### CURRENT FILE

Data Load  
**Regional Settings**  
Privacy  
Auto Recovery

Locale ⓘ

English (United States) ▼

OK

Cancel

## Date Conversion Issue

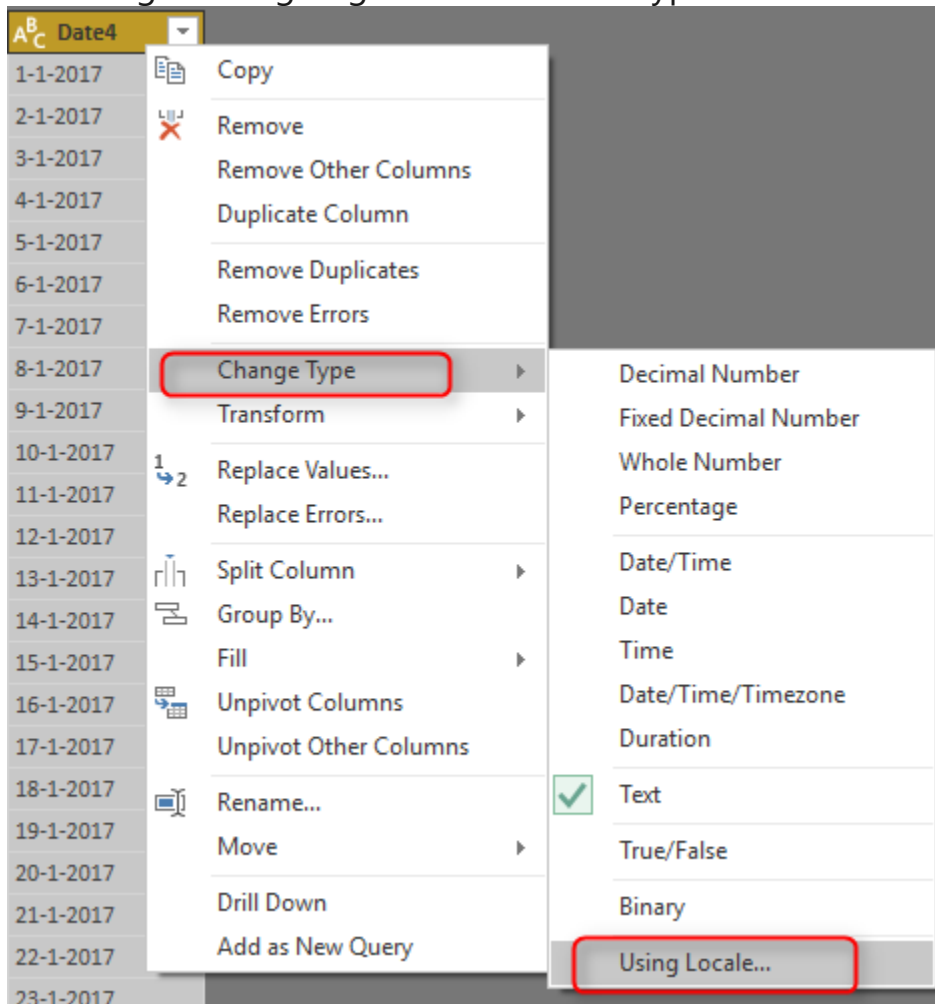
Date4 in the data set isn't converted properly, and that's because the Locale of the current system (my machine) is English (United States). I can change the locale to English (New Zealand) in the Options window and Refresh the data, but this will corrupt the existing Date Conversion of Date3 column. If I try to change the data type of Date4 column myself, the result will not be correct, and I'll see some errors;

	Date1	Date2	Date3	Date4
1	1/1/2017	1/1/2017	1/1/2017	1/1/2017
2	1/2/2017	1/2/2017	1/2/2017	2/1/2017
3	1/3/2017	1/3/2017	1/3/2017	3/1/2017
4	1/4/2017	1/4/2017	1/4/2017	4/1/2017
5	1/5/2017	1/5/2017	1/5/2017	5/1/2017
6	1/6/2017	1/6/2017	1/6/2017	6/1/2017
7	1/7/2017	1/7/2017	1/7/2017	7/1/2017
8	1/8/2017	1/8/2017	1/8/2017	8/1/2017
9	1/9/2017	1/9/2017	1/9/2017	9/1/2017
10	1/10/2017	1/10/2017	1/10/2017	10/1/2017
11	1/11/2017	1/11/2017	1/11/2017	11/1/2017
12	1/12/2017	1/12/2017	1/12/2017	12/1/2017
13	1/13/2017	1/13/2017	1/13/2017	Error
14	1/14/2017	1/14/2017	1/14/2017	Error
15	1/15/2017	1/15/2017	1/15/2017	Error
16	1/16/2017	1/16/2017	1/16/2017	Error
17	1/17/2017	1/17/2017	1/17/2017	Error
18	1/18/2017	1/18/2017	1/18/2017	Error
19	1/19/2017	1/19/2017	1/19/2017	Error
20	1/20/2017	1/20/2017	1/20/2017	Error
21	1/21/2017	1/21/2017	1/21/2017	Error
22	1/22/2017	1/22/2017	1/22/2017	Error
23	1/23/2017	1/23/2017	1/23/2017	Error
24	1/24/2017	1/24/2017	1/24/2017	Error
25	1/25/2017	1/25/2017	1/25/2017	Error
26	1/26/2017	1/26/2017	1/26/2017	Error
27	1/27/2017	1/27/2017	1/27/2017	Error
28	1/28/2017	1/28/2017	1/28/2017	Error
29	1/29/2017	1/29/2017	1/29/2017	Error
30	1/30/2017	1/30/2017	1/30/2017	Error
31	1/31/2017	1/31/2017	1/31/2017	Error

You can see that conversion didn't happen correctly, and also it returned Error in some cells because my machine is expecting MM/DD/YYYY, but the date format in the column is DD/MM/YYYY which is not that format. So it can convert first 12 records because it places the day as a month in the result! And the rest it can't because there is no month 13 or more.

## Date Conversion Using Locale

Fortunately, you can do date conversion using specific Locale. All you need to do is to go through right click and data type conversion using Locale;



In the Change Type with Locale, choose the Data Type to be Date (Normally Locale is for Date, Time, and Numbers). And then set Locale to be English (New Zealand). You can also see some sample input values for this locale there.



## Change Type with Locale

Change the data type and select the locale of origin.

Data Type

Date

Locale

English (New Zealand)

Sample input values:

29/03/2016

Tuesday, 29 March 2016

29 March

March 2016

OK

Cancel

With this simple change, you can now see the Date4 column converted correctly. You still see that in MM/DD/YYYY format in Query Editor window, and that's because my machine's date format is this. The actual column data type is Date, however, which is the correct format to work with.

```

1 let
2     Source = Excel.Workbook(File.Contents("C:\Users\Reza\SkyDrive\Blog\DateConversion\Book1.xlsx"), null, true),
3     Sheet1_Sheet = Source[[Item="Sheet1",Kind="Sheet"]][Data],
4     #"Promoted Headers" = Table.PromoteHeaders(Sheet1_Sheet),
5     #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{
6         {"Date1", type date},
7         {"Date2", type date},
8         {"Date3", type date},
9         {"Date4", type text}
10    }
11    ),
12     #"Changed Type with Locale" = Table.TransformColumnTypes(#"Changed Type", {"Date4", type date}), "en-
13     NZ")
14 in
15     #"Changed Type with Locale"
```

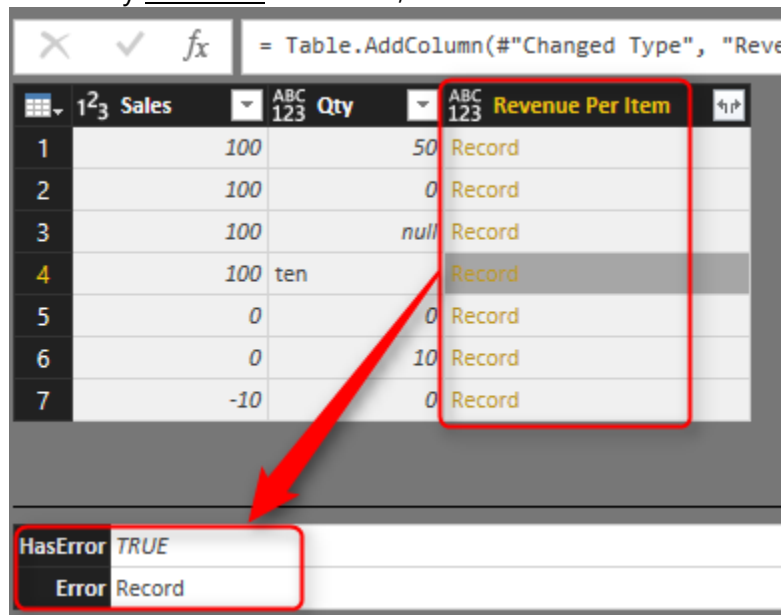
In the script, you can see the difference of data type change using Locale which uses locale as the last parameter, instead of without locale. This brings a very important topic in mind; correct date type conversion is locale-dependent, and you can get it always works if you mention Locale in date type conversion.

## Summary

Date Type conversion can be tricky depends on the locale of system you are working on. To get the correct Date Type conversion recommendation is to use Locale for type conversion. Some of the formats might work even without using Locale. For example, I have seen YYYYMMDD is working fine in all locales I worked with so far, but DDMMYYYY or MMDDYYYY might work differently.

# Make Your Numeric Division Faultless in Power Query

Posted by [Reza Rad](#) on Jun 20, 2016



The screenshot shows the Power Query Editor interface. At the top, the formula bar contains the expression: `= Table.AddColumn(#"Changed Type", "Revenue Per Item", ...)`. Below the formula bar is a table with the following data:

	Sales	Qty	Revenue Per Item
1	100	50	Record
2	100	0	Record
3	100	null	Record
4	100	ten	Record
5	0	0	Record
6	0	10	Record
7	-10	0	Record

At the bottom of the editor, the 'HasError' column is highlighted with a red box, showing the value 'TRUE' for the first row and 'Error' for the subsequent rows. A red arrow points from the 'Revenue Per Item' column to the 'HasError' column.

When you work with data, it is normal that you apply numeric calculations. Numeric calculations in Power Query depends on the nature of data returns different results. One of the most error-prone calculations is division. Power Query behaves differently when you divide a number by zero, zero by zero, number by null, and non-numeric values. One of the most frustrating facts is that not all of these calculations end up to an error. So you can't just remove error rows. In this post, I'll explain some examples of output for the division and a method to find these rows.

## Sample Data Set

For this post, I'll use a sample excel file which has most of the possible combinations that I might face in a division calculation. The table below is some records with Sales Amount and Quantity. And as a simple calculation, I want to find out Revenue Per Item which would be the result of  $[Sales]/[Quantity]$ .

	A	B
1	Sales	Qty
2	100	50
3	100	0
4	100	
5	100	ten
6	0	0
7	0	10
8	-10	0

In the table above there are nulls, texts, zeros, negative, and positive values. Now let's bring the table into Power Query (Excel or Power BI) and apply the division

## Simple Division Calculation

Here is the data set loaded into Power Query. As you can see the Quantity column shows the data type as numeric and text.

	1 <sup>2</sup> Sales	ABC 123 Qty
1	100	50
2	100	0
3	100	null
4	100	ten
5	0	0
6	0	10
7	-10	0

Now If I apply a simple division calculation as a new custom column

### Add Custom Column

New column name

Revenue Per Item

Custom column formula:

= [Sales]/[Qty]

The result would be as below;



fx = Table.AddColumn("#Changed Type", "Revenue Per Item", each [Sales]/[Qty])

	123 Sales	ABC 123 Qty	ABC 123 Revenue Per Item
1	100	50	2
2	100	0	Infinity
3	100	null	null
4	100	ten	Error
5	0	0	NaN
6	0	10	0
7	-10	0	-Infinity

As you see the result set has different outputs depends on the inputs. If the number is divided by a zero value result would be positive or negative infinity (depends on the number). If one of the values be null, the result set would be null. If zero divided by zero, then the result would be NaN! And in case of dividing a number by string or reverse there will be an error raised. Now Let's look at each output separately.

## Error Output

Error in the sample above happened when one of the values is not number. Fortunately, errors can be simply found by TRY keyword. Here is how I change the calculation of Revenue Per Item:

1 = try [Sales]/[Qty]

### Add Custom Column

New column name

Revenue Per Item

Custom column formula:

= try [Sales]/[Qty]

Result set this time would be a Record for each calculation.

fx = Table.AddColumn(#"Changed Type", "Revenue Per Item", ...)

	Sales	Qty	Revenue Per Item
1	100	50	Record
2	100	0	Record
3	100	null	Record
4	100	ten	Record
5	0	0	Record
6	0	10	Record
7	-10	0	Record

HasError	Error
TRUE	Record

The record has two columns: HasError (which says does this record contains error or not), and [Error](#) Record (which would be the error happened in details). So I can add a custom column with a condition on HasError to see if the record contains error or not. In expression below, if I find an error, I will return zero as a result.

```
1 = if [Revenue Per Item][HasError] then 0 else [Revenue Per Item][Value]
```

## Add Custom Column

New column name

Revenue Per Item - Error Removed

Custom column formula:

```
= if [Revenue Per Item][HasError] then 0 else [Revenue Per Item][Value]
```

Available columns:

Sales

Qty

Revenue Per Item

The result set this time would be:

	1 <sup>2</sup> Sales	ABC 123 Qty	ABC 123 Revenue Per Item	ABC 123 Revenue Per Item - Error Removed
1	100	50	Record	2
2	100	0	Record	Infinity
3	100	null	Record	null
4	100	ten	Record	0
5	0	0	Record	NaN
6	0	10	Record	0
7	-10	0	Record	-Infinity

In this example, I just returned zero if I find the error. But you can return an error message if you like with [Revenue Per Item][ErrorMessage]. This method is great error handling method when an error out of the blue happens in your data set. I always recommend using TRY method to get rid of errors that might stop the whole solution to work properly.

I have to mention that steps above are separated to show you how the output of try expression looks like. You can combine both steps above in single step with TRY OTHERWISE as below (Thanks to [Maxim Zelensky](#) for pointing this out);

1 = try [Sales]/[Qty] otherwise 0

## Infinity

Error output can be handled with TRY. However, Infinity and -Infinity are not errors! These are numerical values in Power Query, named [Number.PositiveInfinity](#) and [Number.NegativeInfinity](#).

	1 <sup>2</sup> Sales	ABC 123 Qty	ABC 123 Revenue Per Item	ABC 123 Revenue Per Item - Error Removed
1	100	50	Record	2
2	100	0	Record	Infinity
3	100	null	Record	null
4	100	ten	Record	0
5	0	0	Record	NaN
6	0	10	Record	0
7	-10	0	Record	-Infinity

- PositiveInfinity happens when a positive number divided by zero
- NegativeInfinity happens when a negative number divided by zero

You can't find these with error handling because as I mentioned earlier, these are not error values! You can, however, check these values to see if a value is `NegativeInfinity` or `PositiveInfinity` with sample code below:

- 1 `PInfinity=(if x=Number.PositiveInfinity then false else true),`
- 2 `NInfinity=(if x=Number.NegativeInfinity then false else true)`

## NaN

NaN is another output which happens when zero is divided by zero. [NaN](#) is a number value like positive and negative infinity. So you can't use error handling to spot them out of millions of records.

	1 <sup>2</sup> Sales	ABC 123 Qty	ABC 123 Revenue Per Item	ABC 123 Revenue Per Item - Error Removed
1	100	50	Record	2
2	100	0	Record	Infinity
3	100	null	Record	null
4	100	ten	Record	0
5	0	0	Record	NaN
6	0	10	Record	0
7	-10	0	Record	-Infinity

You can find it with [Number.IsNaN](#) function which works as below;

- 1 `Nan=(if Number.IsNaN(x) then false else true)`

## Null Check

Null values always happen in the data, and best practice is always to replace them with default values. In numeric calculations, if a null value appears in one of the values, the result of the calculation will be null.

	1 <sup>2</sup> Sales	ABC 123 Qty	ABC 123 Revenue Per Item	ABC 123 Revenue Per Item - Error Removed
1	100	50	Record	2
2	100	0	Record	Infinity
3	100	null	Record	null
4	100	ten	Record	0
5	0	0	Record	NaN
6	0	10	Record	0
7	-10	0	Record	-Infinity

You can find nulls with if a condition such as below;

- 1 `Null=(if x=null then false else true)`


## Function to Check All Anomalies

Anomalies in outputs such as the above examples happen in most of the cases, and I found it useful to have a function to check all these options. The function below checks Null, NaN, PositiveInfinity, and NegativeInfinity. It doesn't check errors, however. Error handling is best to be applied on the calculation level as we've done earlier in this post. Here is the code for the function:

```

1 let
2     Source = (x as any) =>
3         let
4             Null=(if x=null then false else true),
5             Pinfinity=(if x=Number.PositiveInfinity then false else true),
6             Ninfinity=(if x=Number.NegativeInfinity then false else true),
7             Nan=(if Number.IsNaN(x) then false else true)
8         in
9             Null and Pinfinity and Ninfinity and Nan
10 in
11     Source

```

 Advanced Editor

## ValidateDivisionResult

```

let
    Source = (x as any) =>
        let
            Null=(if x=null then false else true),
            Pinfinity=(if x=Number.PositiveInfinity then false else true),
            Ninfinity=(if x=Number.NegativeInfinity then false else true),
            Nan=(if Number.IsNaN(x) then false else true)
        in
            Null and Pinfinity and Ninfinity and Nan
in
    Source

```

With function above, now I can add a custom column to my data set, and validate rows;

## Add Custom Column

New column name

Validation Result

Custom column formula:

```
=ValidateDivisionResult([#"Revenue Per Item - Error Removed"])
```

Final result set shows me which record is validated and which one is not. Note that the row containing error has been handled previously, so it is validating as true here.

	123 Sales	ABC 123 Qty	ABC 123 Revenue Per Item	ABC 123 Error Removed	ABC 123 Validation Result
1	100	50	Record	2	TRUE
2	100	0	Record	Infinity	FALSE
3	100	null	Record	null	FALSE
4	100	ten	Record	0	TRUE
5	0	0	Record	NaN	FALSE
6	0	10	Record	0	TRUE
7	-10	0	Record	-Infinity	FALSE

## Summary

Divide by zero is not the only error that happens in Power Query. Power Query returns different results such as Error, NaN, Positive Infinity, Negative Infinity, and Null. For a proper faultless calculation, you have to consider all these exceptions. In this post, You've learned how to spot these exceptions easily and make your calculation faultless with a small amount of effort. If you like to learn more about Power Query and Power BI read [Power BI online book; from Rookie to Rock Star](#).

# Part V: Power Query Formula Language: M

# Power Query Formula Language: M

Posted by [Reza Rad](#) on Feb 7, 2014

```
let
    TableA = #table({"CustomerId", "TranDate", "TranCount"},
        {
            {1, DateTime.FromText("2014-01-01 01:00:00.000"), 10},
            {1, DateTime.FromText("2014-01-01 02:00:00.000"), 5},
            {1, DateTime.FromText("2014-01-03 01:00:00.000"), 5},
            {1, DateTime.FromText("2014-01-04 02:00:00.000"), 80}
        }
    ),

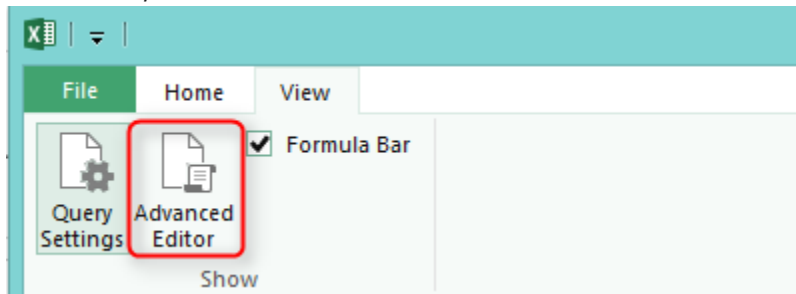
    TableB = #table({"CustomerId", "TranDate", "TranCount"},
        {
            {1, DateTime.FromText("2014-01-01 02:00:00.000"), 10},
            {1, DateTime.FromText("2014-01-01 03:00:00.000"), 5},
            {1, DateTime.FromText("2014-01-02 01:00:00.000"), 20},
            {1, DateTime.FromText("2014-01-02 03:00:00.000"), 15},
            {2, DateTime.FromText("2014-01-01 01:00:00.000"), 5},
            {2, DateTime.FromText("2014-01-01 02:00:00.000"), 80}
        }
    ),
    TableATransformed = Table.Sort(
        Table.AddColumn(TableA, "Date", each Date.From([TranDate]))
        , {"CustomerId", "TranDate"}
    )
in
    Table.Group(TableATransformed, {"CustomerId", "Date"}, {"Total", each List.Last([TranCount])})
```

In the [previous post](#), I described what Power Query is, and how we can use that for self-service ETL. You've how to work with Power Query menus and connect to different data sources, and apply multiple transformations on the data. In this post, I'll go one step closer to the core of Power Query Formula Language known as M. In this post, you will learn about the structure of M language with demo samples.

As you've learned in the previous section, Power Query uses a GUI in Excel Add-In to fetch data from different sources and transform it with some functions. Every change that you apply on the dataset through GUI will be translated to the formula language "M". M is a functional language. M is a powerful language, and the good news is that M is much more powerful than what you see in the Excel GUI of Power Query. The GUI doesn't implement all functionality of M. So for advanced use of Power Query you would require to work with M directly. So as much as you expert yourself in M would result in



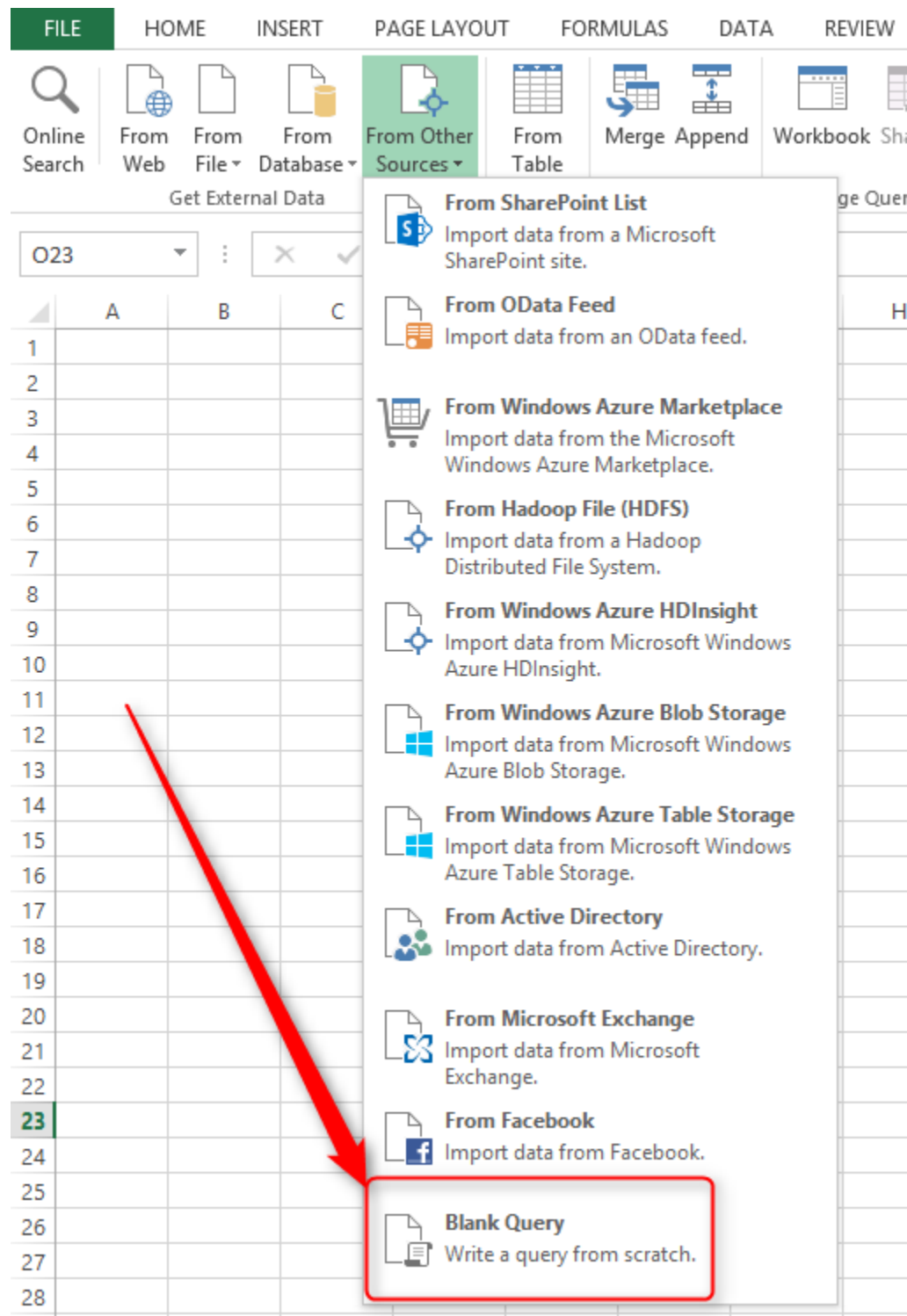
better use of Power Query. So I dare to say learning M is not only the fundamental step but also the most important step in learning Power Query. Follow these steps to get into the Query mode of Power Query; Open Query Editor (In the previous post you've learned that you can open query editor from Excel's Power Query tab). In the Query Editor window go to View tab, and click on Advanced Editor menu item.



You will see the Query script window opens.

Now let's go through some features of M language with a sample; In this sample, I don't use an external data sources, I use static tables to show you how we can do everything with just script.

1- Open an empty excel sheet, and in the Power Query tab, In the "Get External Data" section click on "From Other Sources" and then choose "Blank Query".



2- In the Query Editor window, go to the View tab, and click on Advanced Editor. You will see the script below in advanced editor window:

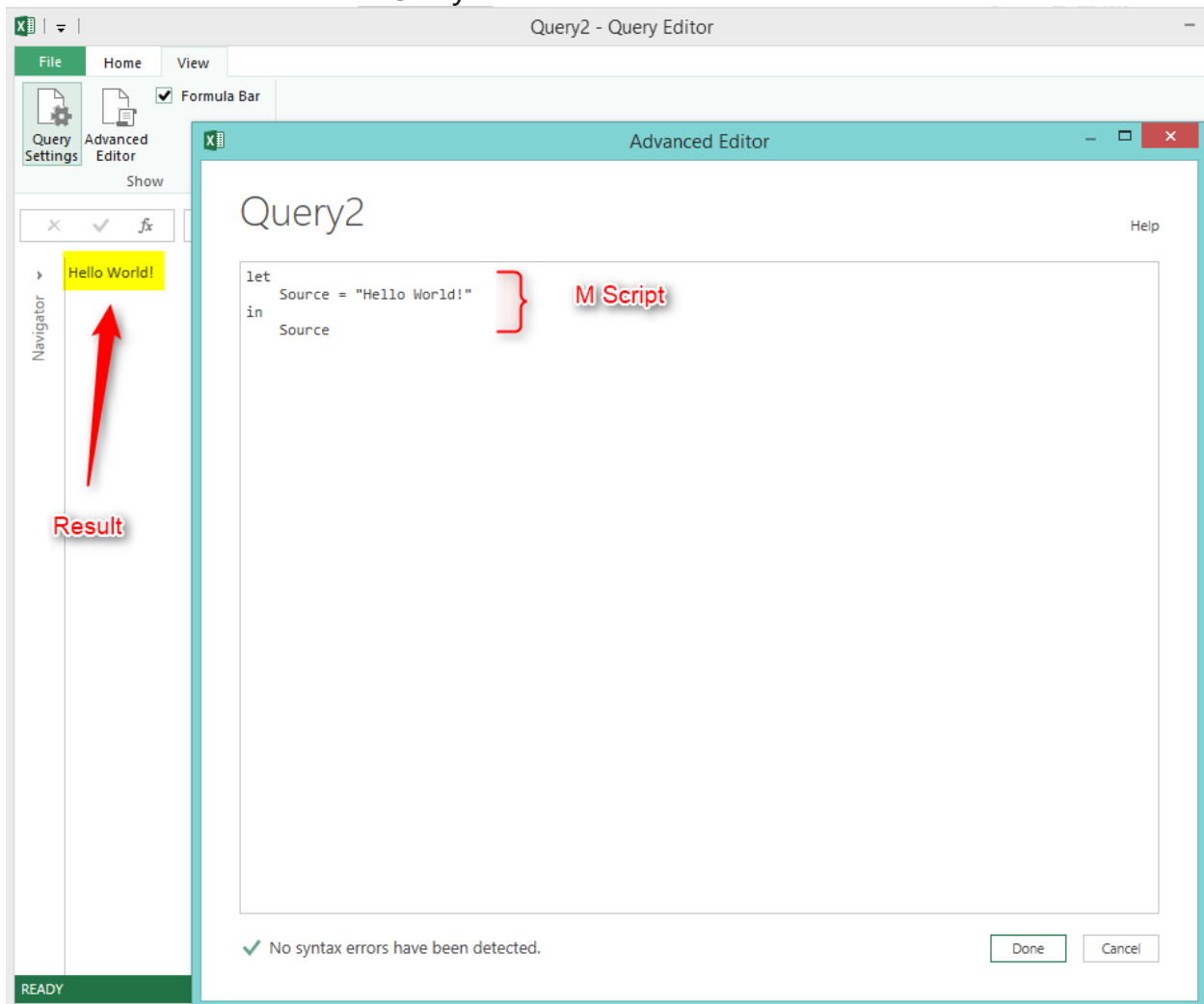
*let*

*Source = ""*

*in*

*Source*

3- Type inside double quotes the string "Hello World!", And then click OK. you will see the result in the Query Editor window as screenshot below illustrates



Wow, That was your first M script. And it was a very simple script. You see that script editor is not that much powerful like Visual Studio or SSMS, So the editor only is the location for writing the script with a very high-level validation bar in the left down part of the window.

The script that you've written has two parts; Let, and In.

**Let;** is the definition area. Here we define variables, records, lists, etc. as you see in this example we defined a variable named "Source", and we assigned a string to it: "Hello World!".

If you define more than one variable, record, or list, you can separate them by a single comma.

For example:

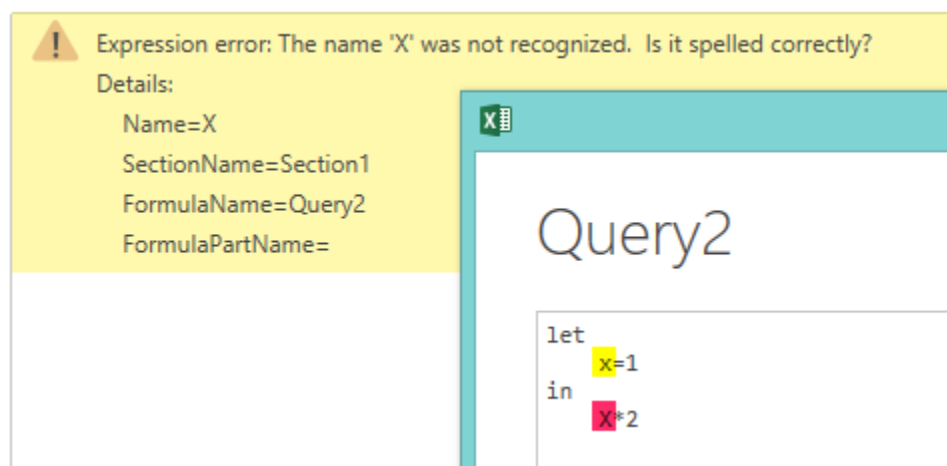
*let*

*x=12,*

*Source="Hello World!"*

**IN;** is the functional area. You can write the result out to the output with this section. In our example, we write down the value of "Source" variable into the Power Query editor window.

4- **M is case sensitive.** You can check that with the script showed in below screenshot.



5- M contains Literals or Values. Values can be used as below;

*1 // number*

*"Hello World!" // string*

*true //logical*

6- Comments; Comments can be determined as a single line or multiple lines.

*// this is comment markup for single line comments*

*/\* this is multiple lines comment markup \*/*

7- Variables define simply in the "let" section with naming the variable and assigning it. For example "Source" variable in the example above.

*let*

*// This is single line comment*

*x=1,*

*y=2*

*in*

*/\* this is*

*multiple lines*

*comment \*/*

*x+y*

8- Defining records and tables; You can define records in let section within two brackets { }, columns in the record will be separated by comma.

a sample record can be defined as following: {1,"Reza","Rad"}

For defining a table you can follow this structure:

*TableX=#table( {"Column A","Column B"},{ {1,11},{2,22} })*

9- Now let's do a real sample with data.

Define a table as below:

*let*

*TableA = #table({"CustomerId", "TranDate","TranCount"},*

*{*

*{1,DateTime.FromText("2014-01-01 01:00:00.000"),10},*

*{1,DateTime.FromText("2014-01-01 02:00:00.000"),5},*

*{1,DateTime.FromText("2014-01-03 01:00:00.000"),5},*

*{1,DateTime.FromText("2014-01-04 02:00:00.000"),80}*

*})*

in

### TableA

The result set of TableA would be as below:

✕
✓
fx

= #table({"CustomerId", "TranDate", "TranCount"},  
 {  
   {1,DateTime.FromText("2014-01-01 01:00:00.000"),10},  
   {1,DateTime.FromText("2014-01-01 02:00:00.000"),5},  
   {1,DateTime.FromText("2014-01-03 01:00:00.000"),5},  
 })

Navigator

	CustomerId	TranDate	TranCount
1	1	1/1/2014 1:00:00 AM	10
2	1	1/1/2014 2:00:00 AM	5
3	1	1/3/2014 1:00:00 AM	5
4	1	1/4/2014 2:00:00 AM	80

As you see, we've used the function **DateTime.FromText** in this example. This function converts text to DateTime data type.

The table defined in this example has three columns; CustomerId, TranDate, TranCount

10- Define second table as below:

```
TableB = #table({"CustomerId", "TranDate", "TranCount"},
{
  {1,DateTime.FromText("2014-01-01 02:00:00.000"),10},
  {1,DateTime.FromText("2014-01-01 03:00:00.000"),5},
  {1,DateTime.FromText("2014-01-02 01:00:00.000"),20},
  {1,DateTime.FromText("2014-01-02 03:00:00.000"),15},
  {2,DateTime.FromText("2014-01-01 01:00:00.000"),5},
  {2,DateTime.FromText("2014-01-01 02:00:00.000"),80}
})
```

this is an illustration of the second table

	CustomerId	TranDate	TranCount
1	1	1/1/2014 2:00:00 AM	10
2	1	1/1/2014 3:00:00 AM	5
3	1	1/2/2014 1:00:00 AM	20
4	1	1/2/2014 3:00:00 AM	15
5	2	1/1/2014 1:00:00 AM	5
6	2	1/1/2014 2:00:00 AM	80

11- the purpose of this part of the example is to group TableA by date. For grouping this table, we transform it to another table with a new column: date. Date column would only contain the date part of TranDate column (not the time portion of it). We use **Table.AddColumn** function for this purpose.

*TableATransformed= Table.AddColumn(TableA,"Date",each  
Date.From([TranDate]))*

***Table.AddColumn(<table name>,<new column name>, expression for  
new column)***

here is the full script for this example:

*let*

```
TableA = #table({"CustomerId", "TranDate", "TranCount"},
{
{1,DateTime.FromText("2014-01-01 01:00:00.000"),10},
{1,DateTime.FromText("2014-01-01 02:00:00.000"),5},
{1,DateTime.FromText("2014-01-03 01:00:00.000"),5},
{1,DateTime.FromText("2014-01-04 02:00:00.000"),80}
}),
TableB = #table({"CustomerId", "TranDate", "TranCount"},
{
{1,DateTime.FromText("2014-01-01 02:00:00.000"),10},
{1,DateTime.FromText("2014-01-01 03:00:00.000"),5},
{1,DateTime.FromText("2014-01-02 01:00:00.000"),20},
{1,DateTime.FromText("2014-01-02 03:00:00.000"),15},
```

```
{2,DateTime.FromText("2014-01-01 01:00:00.000"),5},
{2,DateTime.FromText("2014-01-01 02:00:00.000"),80}
}},
TableATransformed= Table.AddColumn(TableA,"Date",each
Date.From([TranDate]))
```

in

*TableATransformed*

And the result is:

	CustomerId	TranDate	TranCount	Date
1	1	1/1/2014 1:00:00 AM	10	1/1/2014
2	1	1/1/2014 2:00:00 AM	5	1/1/2014
3	1	1/3/2014 1:00:00 AM	5	1/3/2014
4	1	1/4/2014 2:00:00 AM	80	1/4/2014

12- in this step, we sort records with **Table.Sort** function as below:

```
TableATransformed=Table.Sort(
    Table.AddColumn(TableA,"Date",each Date.From([TranDate]))
    ,{"CustomerId","TranDate"}
)
```

Table.Sort works in this structure;

**Table.Sort (<table>,<columns to be sorted>)**

Columns to be sorted can be defined in order like a record. For our example, we used {"CustomerId","TranDate"}, which means sorting will be applied on CustomerId first and then on TranDate. this expression is similar to this T-SQL order by clause; order by CustomerId, TranDate

13- We use **Table.Group** function to group records by the Date Column.

```
Table.Group(TableATransformed,{"CustomerId","Date"},{"Total",each
List.Last([TranCount])})
```



***Table.Group(<table>,<group key columns>,{<name of the new aggregated column>,<expression>})***

**List.Last**([TranCount]) function will return the last record's TranCount.

The result would be as below:

	CustomerId	Date	Total
1	1	1/1/2014	5
2	1	1/3/2014	5
3	1	1/4/2014	80

14- Now repeat the same expression for Table B.

```
TableBTransformed=Table.Sort(
    Table.AddColumn(TableB,"Date",each Date.From([TranDate]))
    ,{"CustomerId","TranDate"}
    ) ,
```

```
TableBGrouped=Table.Group(TableBTransformed,{"CustomerId","Date"},{"Total",
each List.Last([TranCount])})
```

15- In this step we want to join TableA and TableB on two fields; CustomerId, and Date. we will use Table.Join function for this purpose.

**Table.Join** works with this syntax:

***Table.Join(<first table>,<first table keys>,<second table>,<second table keys>,<JoinKind optional>,<JoinAlgorithm optional>)***

The important things to note before applying join in Power Query is that the Table.Join works only on datasets with different column names. So if there be a column with a similar name in both tables, it would return an error. In this example, both tables have a similar structure with similar column names, so we use **Table.PrefixColumns** to change the name of columns for one of the tables.

***Table.PrefixColumns(<table>,"Prefix")***

The result of Table.PrefixColumn would be the same table with the "Prefix" at the beginning of each column name.

so this expression

*Table.PrefixColumns(TableAGrouped,"TableA")*

will result

	TableA.CustomerId	TableA.Date	TableA.Total
1	1	1/1/2014	5
2	1	1/3/2014	5
3	1	1/4/2014	80

now we can join two tables based on the CustomerId and Date column, as below:

*in*

*Table.Join(Table.PrefixColumns(TableAGrouped,"TableA"),{"TableA.CustomerId", "TableA.Date"},TableBGrouped,{"CustomerId", "Date"},JoinKind.FullOuter)*

Join kind can have any of these values:

JoinKind.Inner=0

JoinKind.LeftOuter=1

JoinKind.RightOuter=2

JoinKind.FullOuter=3

JoinKind.LeftAnti=4

JoinKind.RightAnti=5

(Left Anti and Right Anti will return only records from a table that doesn't have a match in the other table -based on left/right respectively)

you can use codes instead of enumeration. this means that expression below would return same result as previous one:

*Table.Join(Table.PrefixColumns(TableAGrouped,"TableA"),{"TableA.CustomerId", "TableA.Date"},TableBGrouped,{"CustomerId", "Date"},3)*

But it is highly recommended to use enumeration because it is much easier to read.

result of above expression would be as below:

	TableA.CustomerId	TableA.Date	TableA.Total	CustomerId	Date	Total
1	1	1/1/2014	5	1	1/1/2014	5
2	null	null	null	1	1/2/2014	15
3	null	null	null	2	1/1/2014	80
4	1	1/3/2014	5	null	null	null
5	1	1/4/2014	80	null	null	null

Here is the full script for this example:

*let*

```

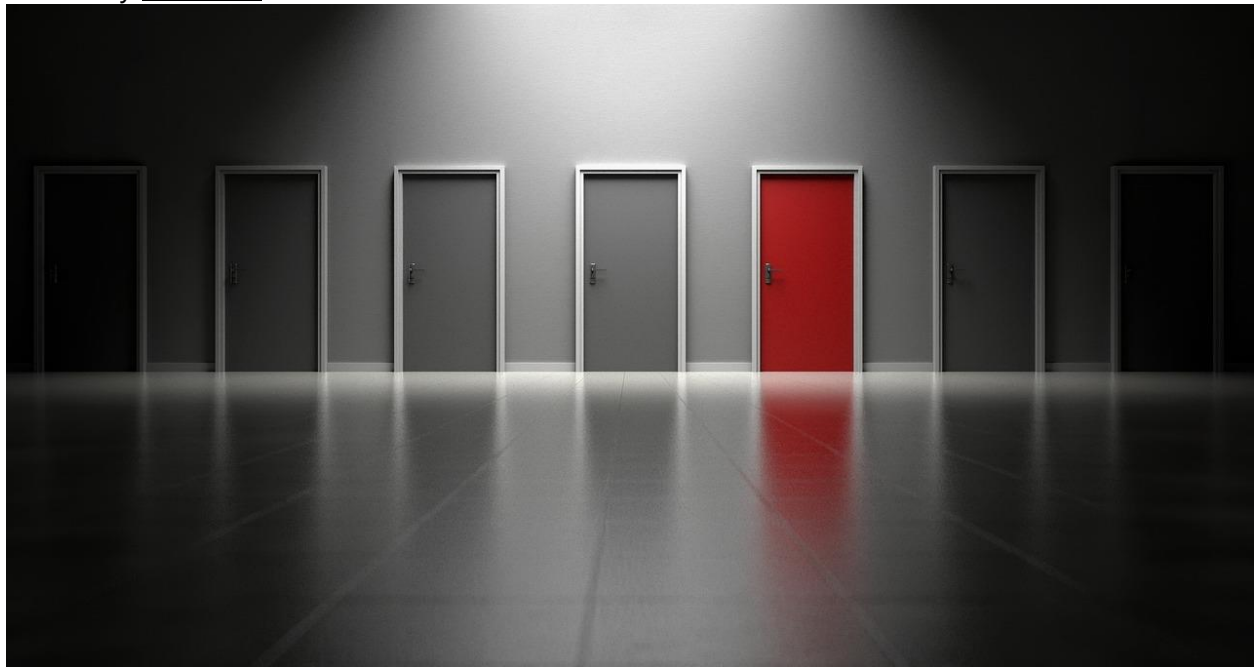
    TableA = #table({"CustomerId", "TranDate", "TranCount"},
    {
        {1, DateTime.FromText("2014-01-01 01:00:00.000"), 10},
        {1, DateTime.FromText("2014-01-01 02:00:00.000"), 5},
        {1, DateTime.FromText("2014-01-03 01:00:00.000"), 5},
        {1, DateTime.FromText("2014-01-04 02:00:00.000"), 80}
    }),
    TableB = #table({"CustomerId", "TranDate", "TranCount"},
    {
        {1, DateTime.FromText("2014-01-01 02:00:00.000"), 10},
        {1, DateTime.FromText("2014-01-01 03:00:00.000"), 5},
        {1, DateTime.FromText("2014-01-02 01:00:00.000"), 20},
        {1, DateTime.FromText("2014-01-02 03:00:00.000"), 15},
        {2, DateTime.FromText("2014-01-01 01:00:00.000"), 5},
        {2, DateTime.FromText("2014-01-01 02:00:00.000"), 80}
    }),
    TableATransformed=Table.Sort(
        Table.AddColumn(TableA, "Date", each Date.From([TranDate]))
        , {"CustomerId", "TranDate"}
    ),
    TableBTransformed=Table.Sort(

```

```
Table.AddColumn(TableB,"Date",each Date.From([TranDate]))
,{"CustomerId","TranDate"}
) ,
TableAGrouped=Table.Group(TableATransformed,{"CustomerId","Date"},{"T
otal",each List.Last([TranCount])}),
TableBGrouped=Table.Group(TableBTransformed,{"CustomerId","Date"},{"T
otal",each List.Last([TranCount])})
in
Table.Join(Table.PrefixColumns(TableAGrouped,"TableA"),{"TableA.CustomerId","
TableA.Date"},TableBGrouped,{"CustomerId","Date"},JoinKind.FullOuter)
We will go through more data transformation in next blog posts.
```

# M or DAX? That is the Question!

Posted by [Reza Rad](#) on Mar 3, 2017



What is the main difference between M and DAX? Why can we do a calculated column in two different places? What are the pros and cons of each? Which one should I use for creating a profit column? Why I cannot do all of it in only one; DAX or M! Why two different languages?! Why the structure of these two are so different? ... If any of these are your questions, then you need to read this post. In this post, I'll go through differences of these two languages, and explain why, when, where of it. Normally I don't get this question asked from students of my Power BI course, because I elaborate the difference in details. However, if you have this question, this is a post for you. If you would like to learn more about Power BI; read [Power BI book; from Rookie to Rock Star](#).

## What is M?

M is the scripting language behind the scene for Power Query. M is the informal name of this language. The formal name is Power Query Formula Language! Which is long, and even Microsoft refer it to M. M stands for many

things, but one of the most common words of it is Mashup. Which means this language is capable of data mashup, and transformation. M is a functional language. And structure of M script can be similar to this:

```
let
    FirstAndLastDayOfTheMonth = (date) =>
        let
            dated=Date.FromText(date) ,
            year=Date.Year(dated) ,
            month=Date.Month(dated) ,
            FirstDateText=Text.From(year)&"-"&Text.From(month)&"-01" ,
            FirstDate=Date.FromText(FirstDateText) ,
            daysInMonth=Date.DaysInMonth(dated) ,
            LastDateText=Text.From(year)&"-"&Text.From(month)&"-"&Text.From(daysInMonth) ,
            LastDate=Date.FromText(LastDateText) ,
            record=Record.AddField([], "First Date of Month",FirstDate) ,
            resultset=Record.AddField(record,"Last Date of Month",LastDate)
        in
            resultset
in
    FirstAndLastDayOfTheMonth("30/07/2015")
```

[Source: Day Number of Year Function in Power Query](#)

M is a step by step language structure. Usually (Not always), every line in M script is a data transformation step. And the step after that will use the result of the previous step. It is usually easy to follow the structure of M language for a programmer. Because it is understandable with programming blocks of Let and In, and some other programming language features alike.

## What is DAX?

DAX is Data Analysis Expression Language. This is the common language between SQL Server Analysis Services Tabular, Power BI, and Power Pivot in Excel. DAX is an expression language, and unlike M, it is very similar to Excel functions. DAX has many common functions with Excel. However, DAX is much more powerful than Excel formula in many ways. Here is an example DAX expression:

```

Sales Rolling 12 Months =
CALCULATE(
    SUM(FactInternetSales[SalesAmount]),
    DATESBETWEEN
        (DimDate[FullDateAlternateKey],
        NEXTDAY(SAMEPERIODLASTYEAR(LASTDATE(DimDate[FullDateAlternateKey]))),
        LASTDATE(DimDate[FullDateAlternateKey])
    ),
    ALL(DimDate)
)

```

Source: [Secret of Time Intelligence Functions in Power BI](#)

DAX calculations are built in a way that makes sense mostly for Excel users. Normally Excel users are very comfortable with this language. Everything goes through functions. DAX doesn't have programming blocks in it and is a combination of function uses, filters, and expressions.

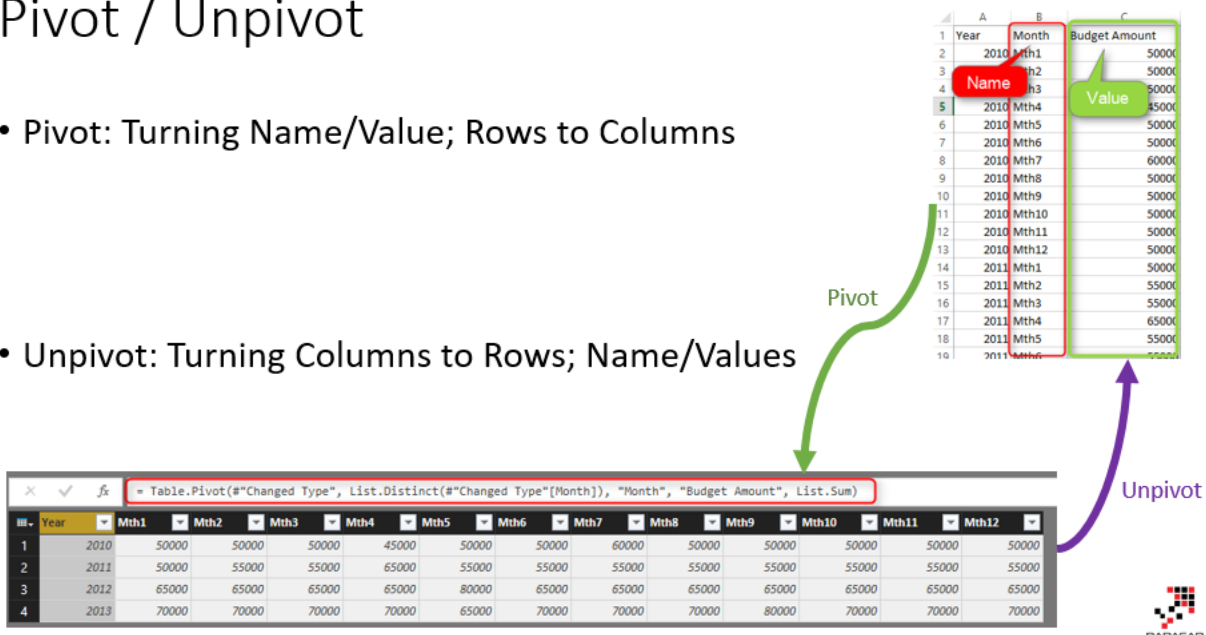
## Example Usage of M

M can be used in many data transformation scenarios. As an example, it can be used to Pivot or Unpivot Data, To [Group](#) it based on some columns. Here is how a [Pivot/Unpivot](#) can work in Power Query;

### Pivot / Unpivot

- Pivot: Turning Name/Value; Rows to Columns

- Unpivot: Turning Columns to Rows; Name/Values



## Example Usage of DAX?

DAX can be used for many calculations for analyzing data. For example, calculating Year To Date, Calculating [Rolling 12 Months Average](#), or anything like that. Here is an example which based on selection criteria in the report and few simple DAX expressions we can do a [customer retention](#) case with DAX;

FullName	Total Revenue	Last Period Revenue	Lost Customers	New Customers
Aaron Adams	\$118	\$118	0	1
Aaron Alexander	\$70	\$70	0	1
Aaron Allen	\$3,400		1	0
Aaron Baker	\$1,751	\$1,751	0	1
Aaron Bryant	\$134	\$134	0	1
Aaron Butler	\$15	\$15	0	1
Aaron Campbell	\$1,155	\$1,155	0	1
Aaron Carter	\$40	\$40	0	1
Aaron Chen	\$40	\$40	0	1
Aaron Coleman	\$62	\$62	0	1
Aaron Collins	\$6,047	\$2,469	0	0
Aaron Diaz	\$6,030	\$2,451	0	0
Aaron Edwards	\$94	\$94	0	1
Aaron Evans	\$2,433	\$2,433	0	1

Period

☐ 30

☐ 60

☐ 120

☐ 180

☐ 365

☒ 1461

☐ 2922

**1461**  
Selected Period

EnglishProductName

☐ Adjustable Race

☐ All-Purpose Bike Stand

☐ AWC Logo Cap

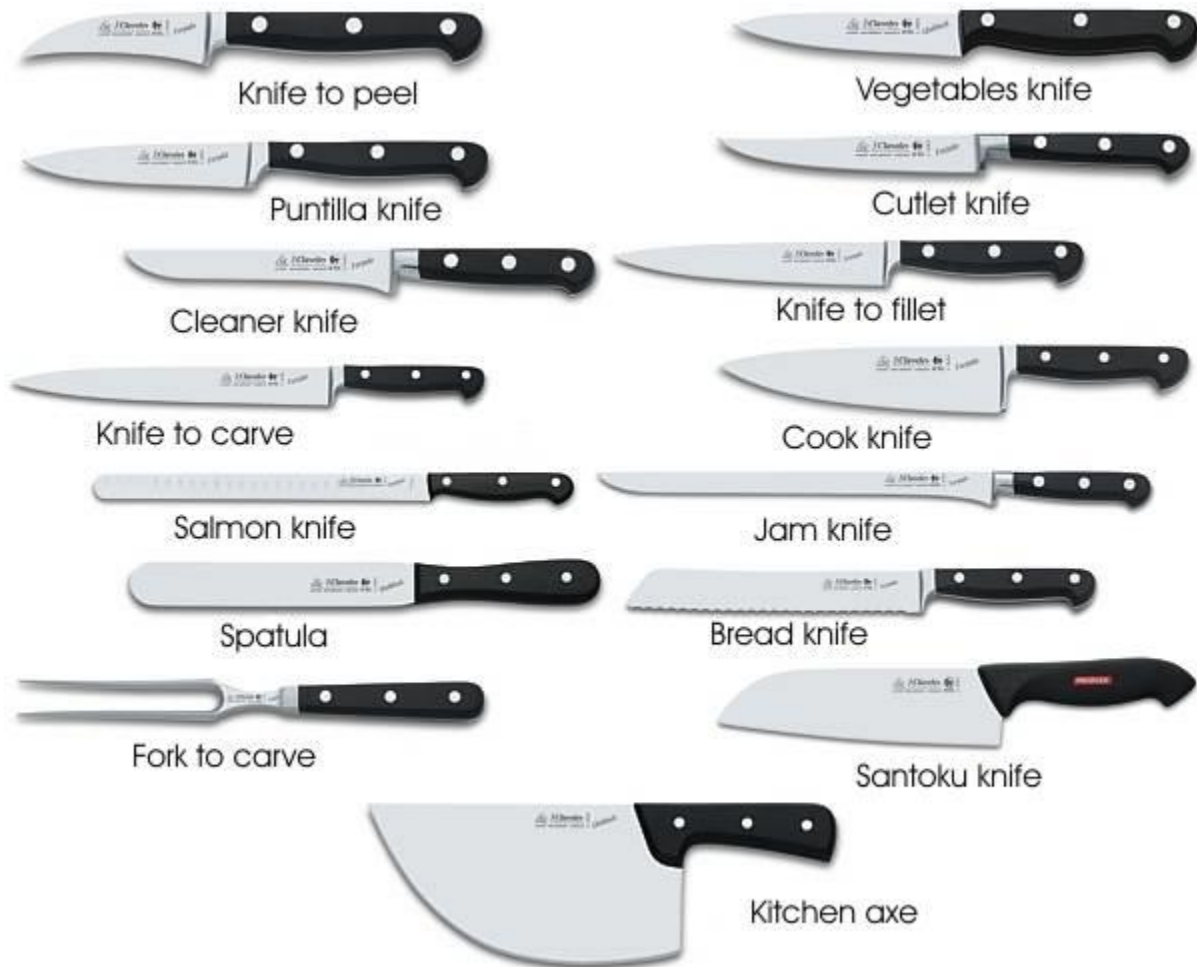
☐ BB Ball Bearing

## Calculated Column Dilemma

The main question of choosing between DAX and M comes from calculated column dilemma in my opinion. You can create many calculated columns in both M or DAX, and it is confusing where is the best place to do it, or why there are two different places to do it?! As an example; you can create a full name which is concatenated of FirstName and LastName column. You can do that in M, and also in DAX. So this question comes up that: Why two different places? Which one is best to use? Can we always use one language?

To answer this question, I would like to use another example; There are many types of knives, and you can use almost all of them to cut the cheese!





reference: <http://forkitchen.blogspot.co.nz/2008/10/what-are-different-types-of-kitchen.html>

Almost every knife in the above picture can be used for cutting cheese except one of them! So why there are so many knives for cutting cheese?! The answer is that; these are not knives for cutting cheese! Each knife is good for doing one special case. For cutting bread, bread knife gives you the best result. For cutting a fillet, you normally need another type of knife. But as you agree, for some cases (such as cutting cheese!) you can use many of these knives. Let's know to go back to the original question;

## **Why can I create a same calculated column in DAX or M?**

These two languages are built independently. They built in a way that they can handle most of the business-related solutions. So, as a result, there are some use cases that both languages are capable of doing it. As an example, both of these languages can easily be used to create a concatenated column of two other columns.

### **Which one is best?**

The quick answer is Depends! Depends on the type of usage. If you want to create a concatenated column; Power Query (M) is a better option in my view, because that is normally like the ETL part of your BI solution, you can simply build your model and data sets in a way you like it to be. But if you want to create something like Year To Date; you can do that in Power Query or M, but it will be lots of code, and you have to consider many combinations of possibilities to create a correct result, while in DAX you can simply create that with the usage of TotalYTD function. So the answer is; there is no best language between these two. The type of usage identifies which one is best. Normally any changes to prepare the data for the model is best to be done in M, and any analysis calculation on top of the model is best to be done in DAX.

### **Two Languages for Two Different Purposes**

There are many programming languages in the world; each language has its pros and cons. JavaScript is a language of web scripting, which is very different from ASP.NET or PHP. The same thing happens here. When M born, it meant to be a language for data transformation, and it is still that language. DAX was created to answer business analysis questions.

## What Questions Can DAX Answer?

DAX is the analytical engine in Power BI. It is the best language to answer analytical questions which their responses will be different based on the selection criteria in the report. For example; You might want to calculate Rolling 12 Months Average of Sales. It is really hard if you want to calculate that in M, because you have to consider all different types of possibilities; Rolling 12 months for each product, for every customer, for every combinations, etc. However if you use a DAX calculation for it, the analytical engine of DAX take care of all different combinations selected through Filter Context in the report.

## What Questions Can M Answer?

M is Data Transformation engine in Power BI. You can use M for doing any data preparation and data transformation before loading that into your model. Instead of bringing three tables of DimProduct, DimProductSubcategory, and DimProductCategory, you can merge them all in Power Query, and create a single DimProduct including all columns from these tables, and load that into the model. Loading all of these into the model and using DAX to relate these to each other means consuming extra memory for something that is not required to be in the model. M can combine those three tables and based on “Step Based” operational structure of M, they can be simply used to create a final data set.

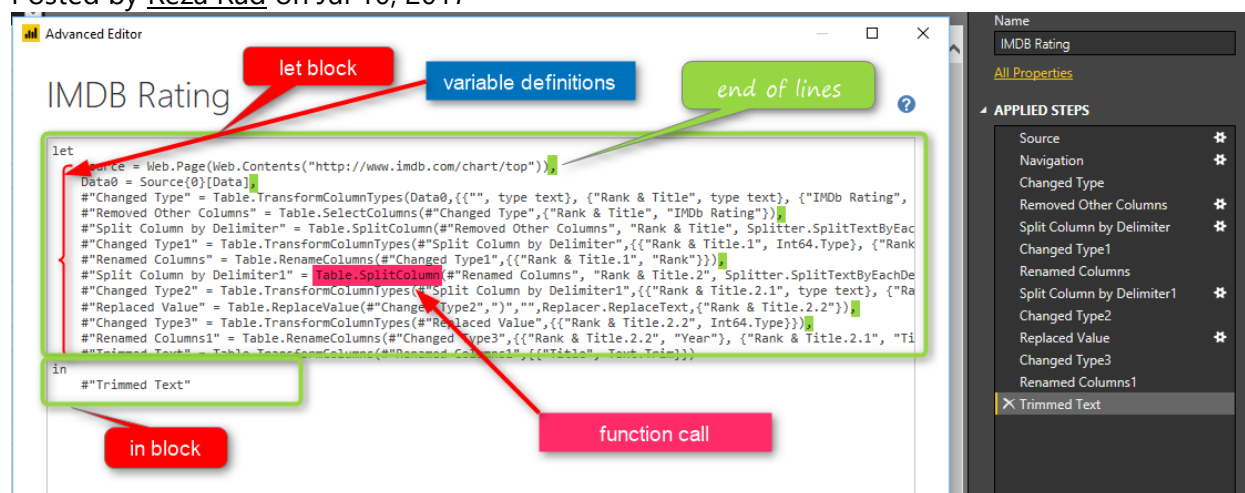
## As a Power BI Developer Which Language Is Important to Learn?

Both! With no hesitation! M is your ETL language, and DAX is the analytical language. You cannot live with only one. If you want to be an expert in Power BI, you should be an expert in both of these languages. There are some cases

that one of the languages will be used more than the other one. However, you will need a very good understanding of both languages to understand which one is best for which purpose, and easily can use it in real-world scenarios.

# Basics of M: Power Query Formula Language

Posted by [Reza Rad](#) on Jul 10, 2017



M is the powerful language behind the scene of Power Query. Any transformation you apply will be written in M language. For many, M looks like a scary language. In this post, I like to explain a bit of basic of M. Not mentioning any functions. Mainly I want to explain to you how the M language syntax is structured. Once you know the syntax, then everything becomes simple. M is a language that you can learn it's syntax easily. As a Power Query developer; I highly recommend you to spend time on M, because there are MANY operations that you can with M, but you might not be able to do it simply with the graphical interface. If you would like to learn more about Power BI, read [Power BI book from Rookie to Rock Star](#).

## What is M?

M is an informal name of Power Query Formula Language. The formal name is so long that no one uses that, everyone calls it M! M stands for Data Mashup, some say stands for Data Modeling. M is a functional language, and it is important to know functions of it. However, each language has a structure and syntax which is the beginner level of learning that language. In this post, I will

explain the syntax of M. Before learning M; I would like you to read this sentence loud;

*M is much more powerful than the graphical interface of Power Query*

Yes, you read it correct! The graphical interface of Power Query is changing every month. Every month new functionality comes to this graphical interface. But the fact is all of these functionalities has been in the language for many years! If you knew the language, you could easily use them, instead of waiting for graphical interface option for it. There are heaps of examples for it. One very small example is [here](#): you can extend your Grouping ability in Power Query with a minor change in M script of it.

## Syntax of M

The syntax of this language is simple. It always has two blocks of programming: LET expression block, and IN expression block. Here is the most simple M syntax;

```
1 let
2   x=1
3 in
4   x
```

**let** and **in** are reserved words. Before going even one step further, the first and foremost thing you need to know;

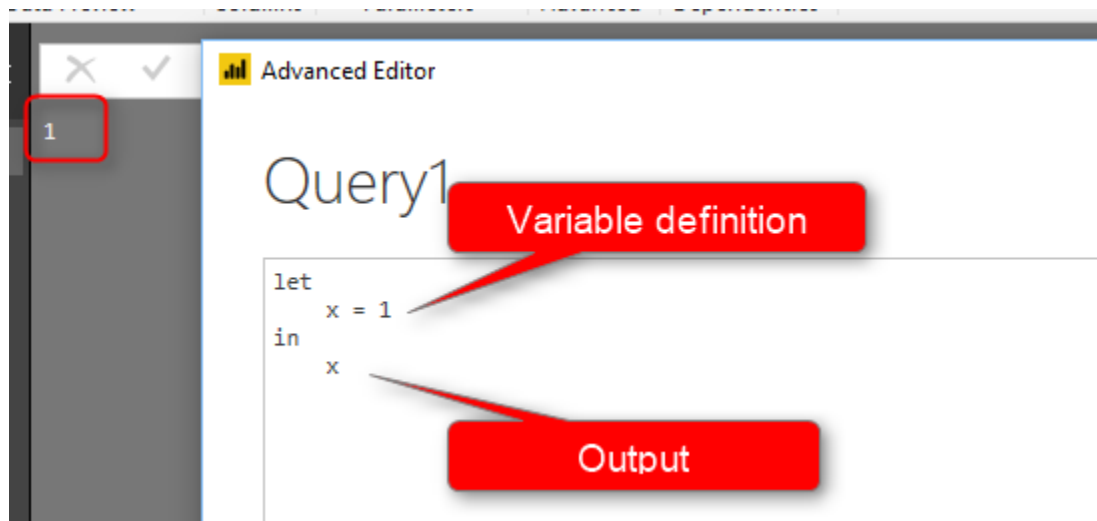
*M (Power Query Formula Language) is Case Sensitive. There is a difference between x and X.*

What are these two programming blocks:

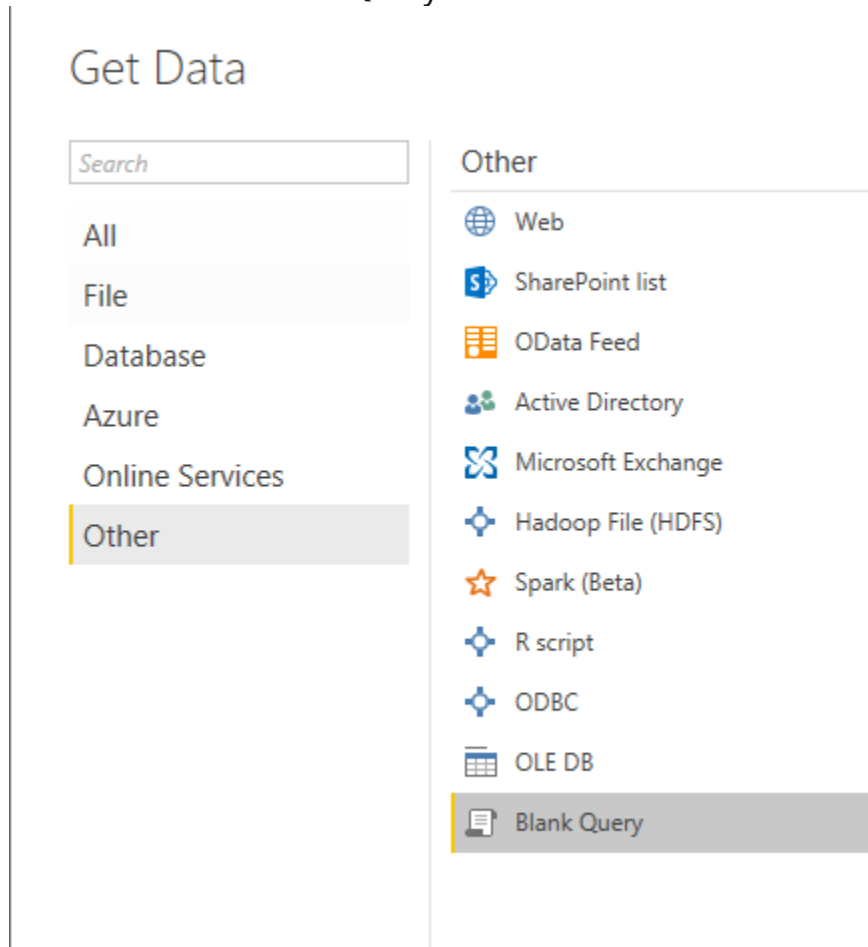
let: definition of all variables

in: output! Yes, it means out! just named as in. Everything you put in this block will be the output of your query.

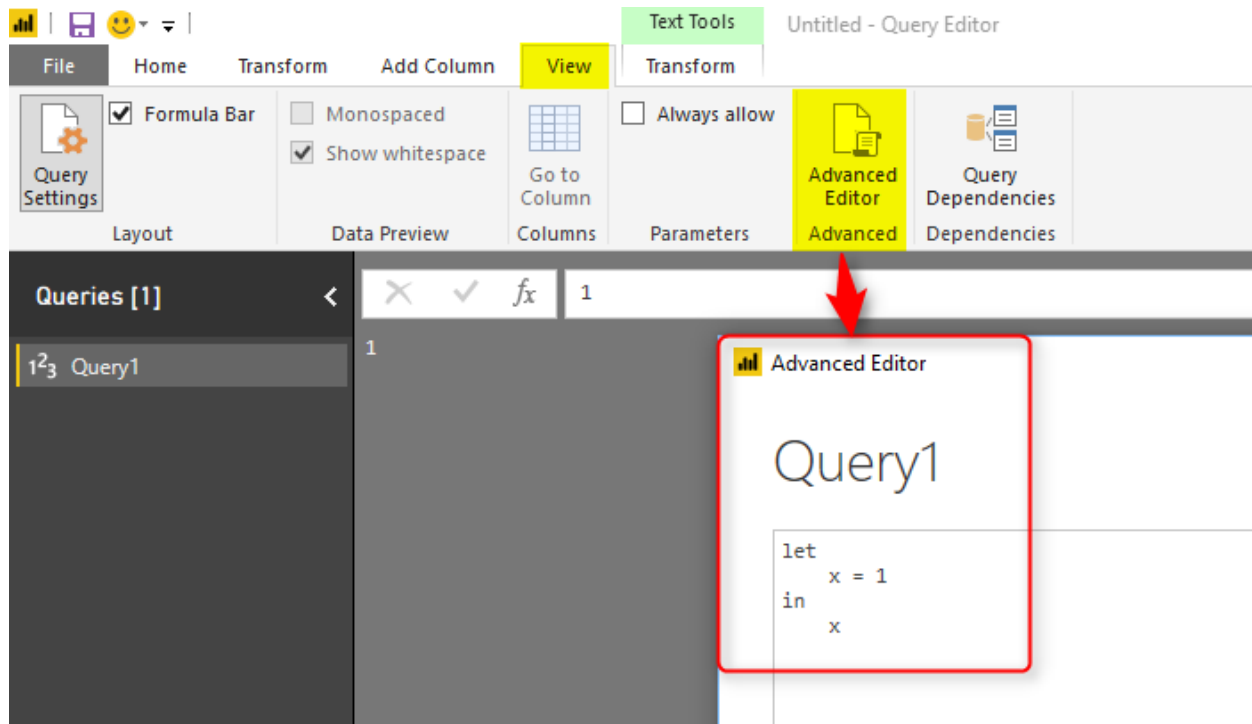
So basically, the query below means defining a variable named as x, assigning the value 1 to it, and showing it as the result set. So the query will return 1.



to run this example, you need to Open Power BI Desktop. Go to getting Data, start with New Blank Query.



then in View tab, select advanced Editor;



Make sure when you write the script that you put reserved words such as `let` and `in` all lowercase. Also your variable name should be the same case in both `let` and section.

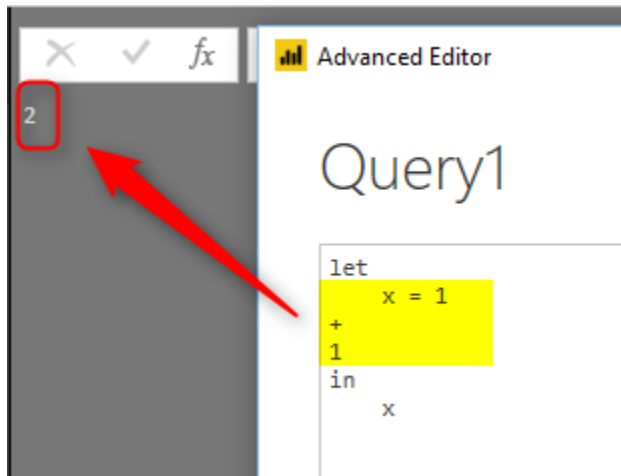
As you can see, there is no need to define data types for the variable. It will be automatically assigned when the first assignment occurs.

If you specify a text value, then variable would be a text data type automatically.

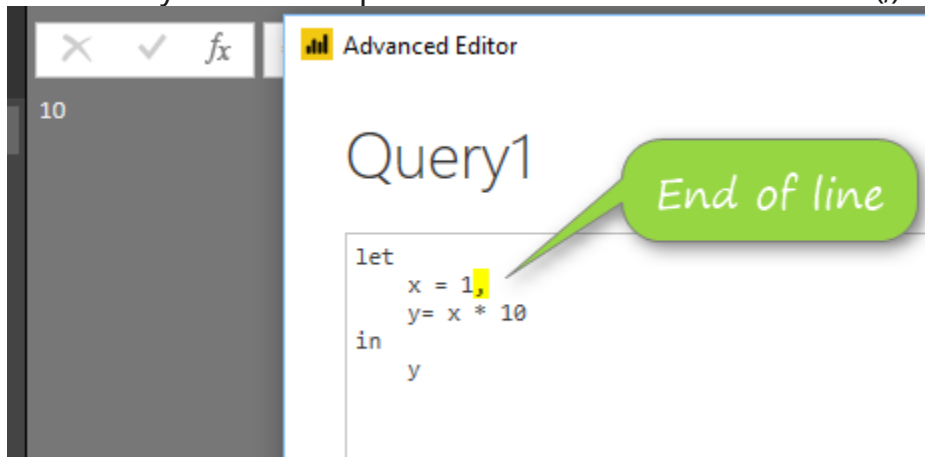
## End of the Line

Lines of codes in M continues if you don't put the end of the line character.





As you can see in the above example, the line continues, and x will be equal to  $x = 1 + 1$ . If you want to put an end for a line use comma(.). Example here:



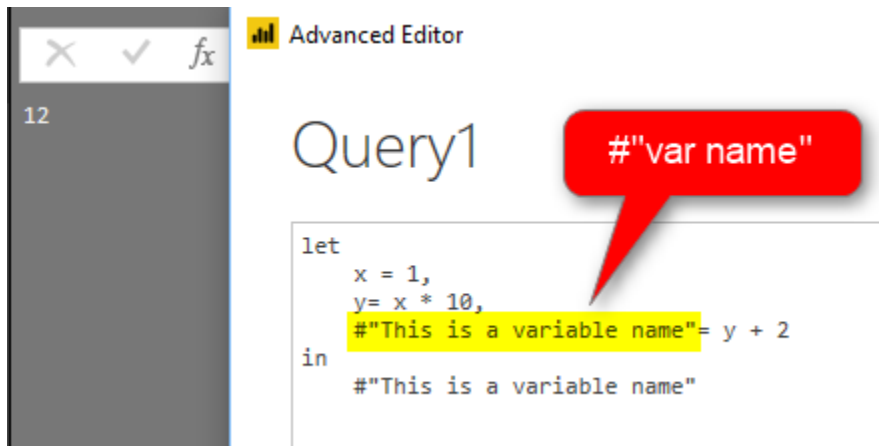
every line needs a comma(,) to finish. Except the last line before "in".

## Variable Names

Name of variables can be all one word, like Source. Or it can have spaces in it. In case that you have some characters such as space, then you need to put the name inside double quote (") and put a hashtag at the beginning of it(#).

Something similar to:

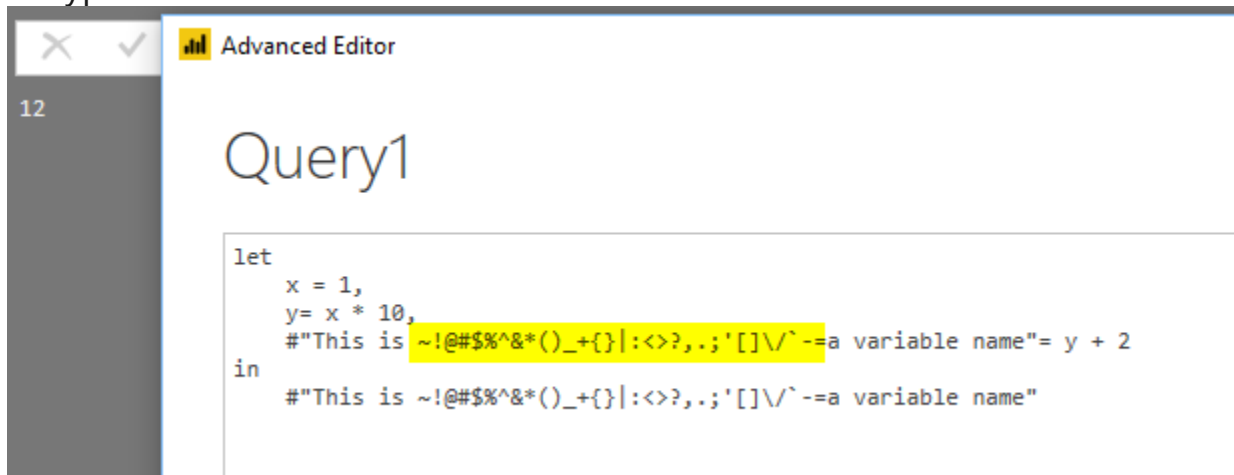
```
1 #"This is a variable name"
```



A variable name can contain special characters, here is an example:

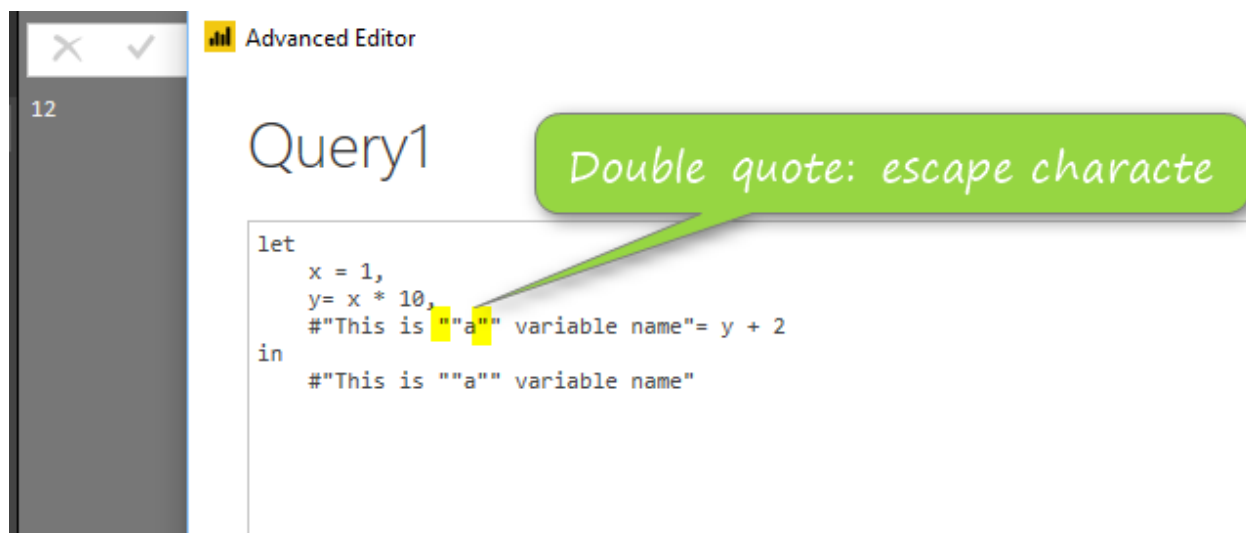
## Special characters

Variable names can have a special character, as you can see below variable has all types of characters in it and still runs well.



## Escape character

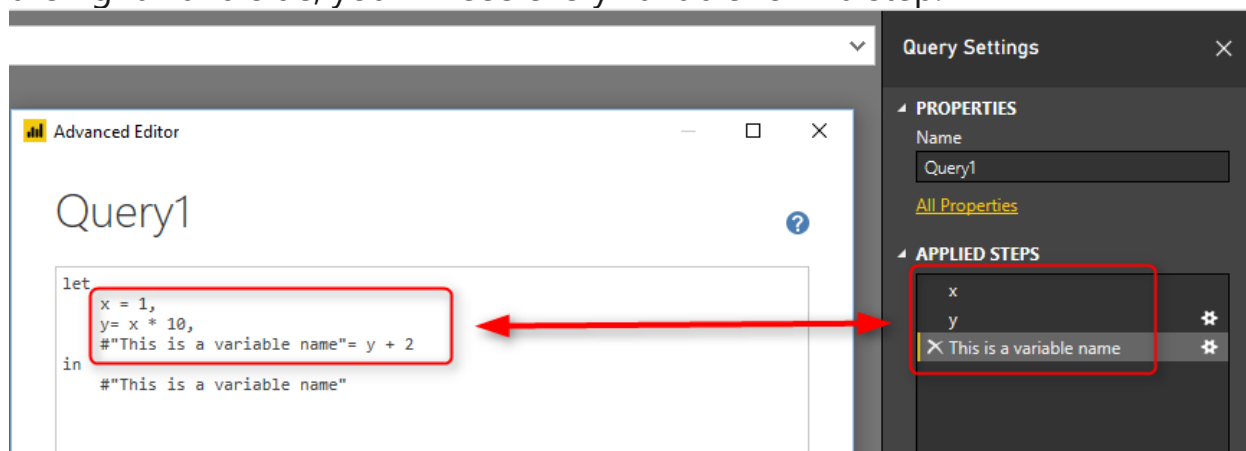
Double quote (") is escape character. You can use it to define variables with names that have another double quote in it. Here is an example:



first double quote (highlighted) above is necessary to be before the second double quote (which is part of the variable name).

## Step by Step Coding

Power Query is a step by step transformation. Every transformation usually happens in step. While you are writing the code, you can also notice that in the right-hand side, you will see every variable form a step.

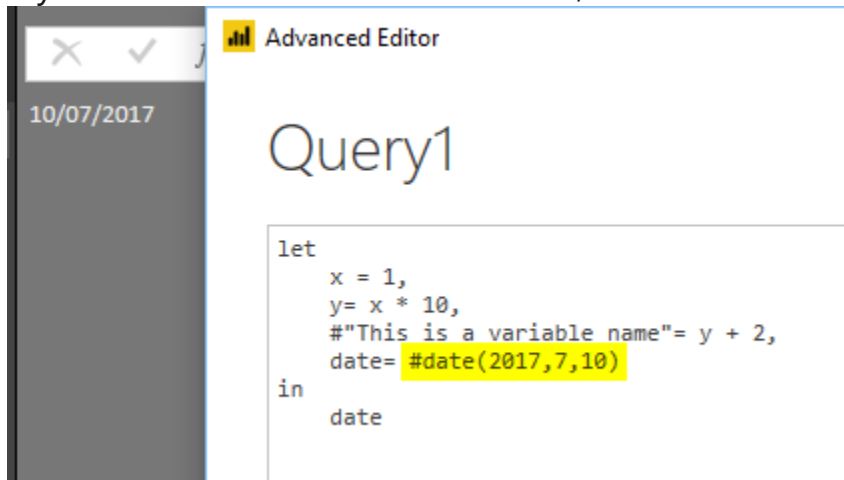


in the screenshot above, you can see every variable is determined as a step. And if the variable has space in the name, it will show it with spaces in the list of applied steps.

The last variable is always specified in the **in** section.

## Literals

There are different ways of defining every literal in Power Query. For example, if you want to define a date variable, here is how to do it;



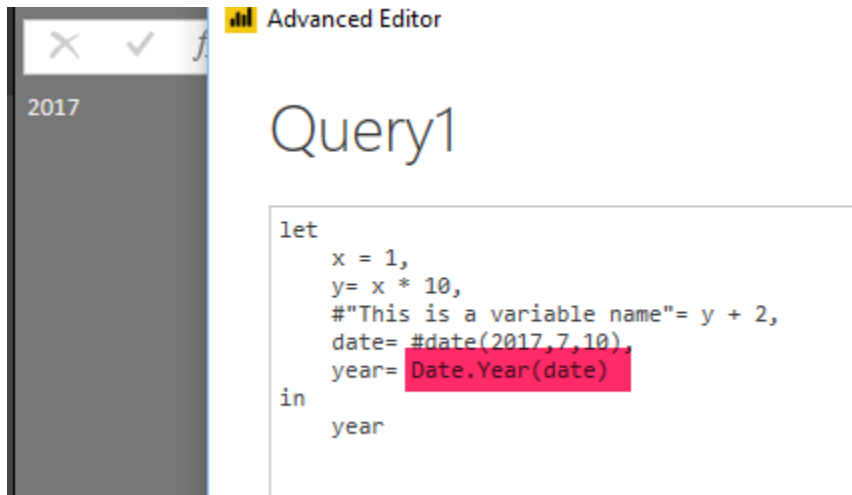
for defining all other types of literals, here is the reference table to use:

<b>Kind</b>	<b>Literal</b>
<i>Null</i>	<code>null</code>
<i>Logical</i>	<code>true    false</code>
<i>Number</i>	<code>0    1    -1    1.5    2.3e-5</code>
<i>Time</i>	<code>#time(09,15,00)</code>
<i>Date</i>	<code>#date(2013,02,26)</code>
<i>DateTime</i>	<code>#datetime(2013,02,26, 09,15,00)</code>
<i>DateTimeZone</i>	<code>#datetimezone(2013,02,26, 09,15,00, 09,00)</code>
<i>Duration</i>	<code>#duration(0,1,30,0)</code>
<i>Text</i>	<code>"hello"</code>
<i>Binary</i>	<code>#binary("AQID")</code>
<i>List</i>	<code>{1, 2, 3}</code>
<i>Record</i>	<code>[ A = 1, B = 2 ]</code>
<i>Table</i>	<code>#table({"X","Y"},{{0,1},{1,0}})</code>
<i>Function</i>	<code>(x) =&gt; x + 1</code>
<i>Type</i>	<code>type { number }    type table [ A = any, B = text ]</code>

\* for function and type; I'll write another post later to explain how these types work.

## Function Call

M is a functional language, and for doing almost everything, you need to call a function for it. Functions can be easily called with the name of the function and specifying parameters for it.



the screenshot above uses Date.Year function, which fetches the year part of a date. Functions names start always with capital letters: **Date.Year()**

## Comments

Like any programming language, you can put some comments in your code. It can be in two forms;

Single line commentary with a double slash (//)

```
let
    // this is a comment line and will not be executed
    x = 1,
    y = x * 10,
    #"This is a variable name" = y + 2,
    date = #date(2017, 7, 10),
    year = Date.Year(date)
in
    year
```

Multi-line commentary between slash and stars (/ \* comments \*/)

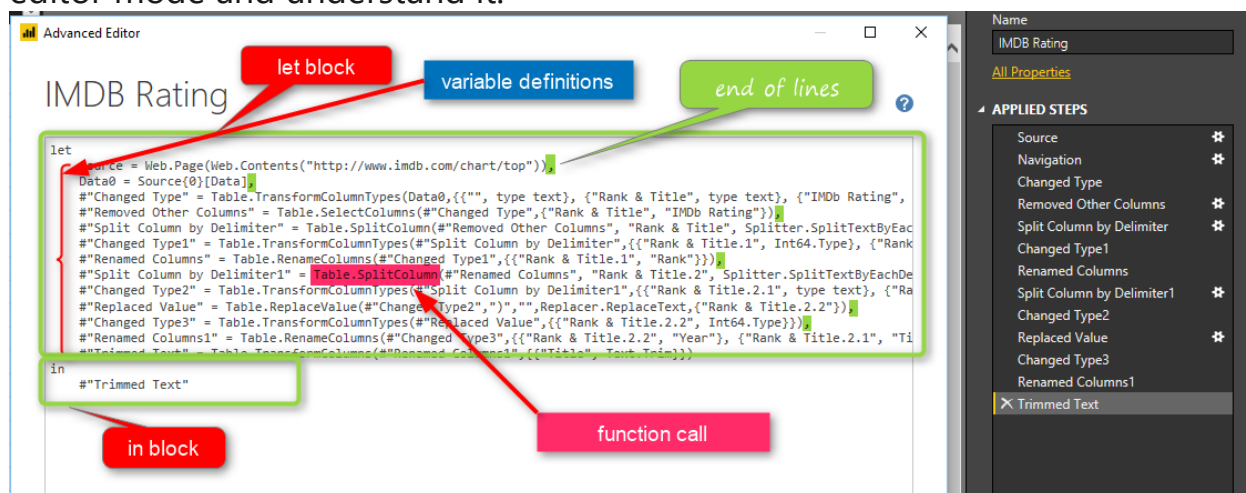
```

let
    /* this is a
    multi
    line
    comment and will not be executed
    */
    x = 1,
    y = x * 10,
    #"This is a variable name" = y + 2,
    date = #date(2017, 7, 10),
    year = Date.Year(date)
in
    year[

```

## A real-world example

Now that you know some basics let's look at an existing query in advanced editor mode and understand it.



in the screenshot above, you can see all the basics mentioned so far:

1. let and in block
2. variable names matching steps applied in the query
3. some variable names with the hashtag and double quote: #"var name"
4. end of the line characters: comma
5. calling many functions

There are still many parts of this query that you might not understand.

Specially when using functions. You need to learn what functions are doing to

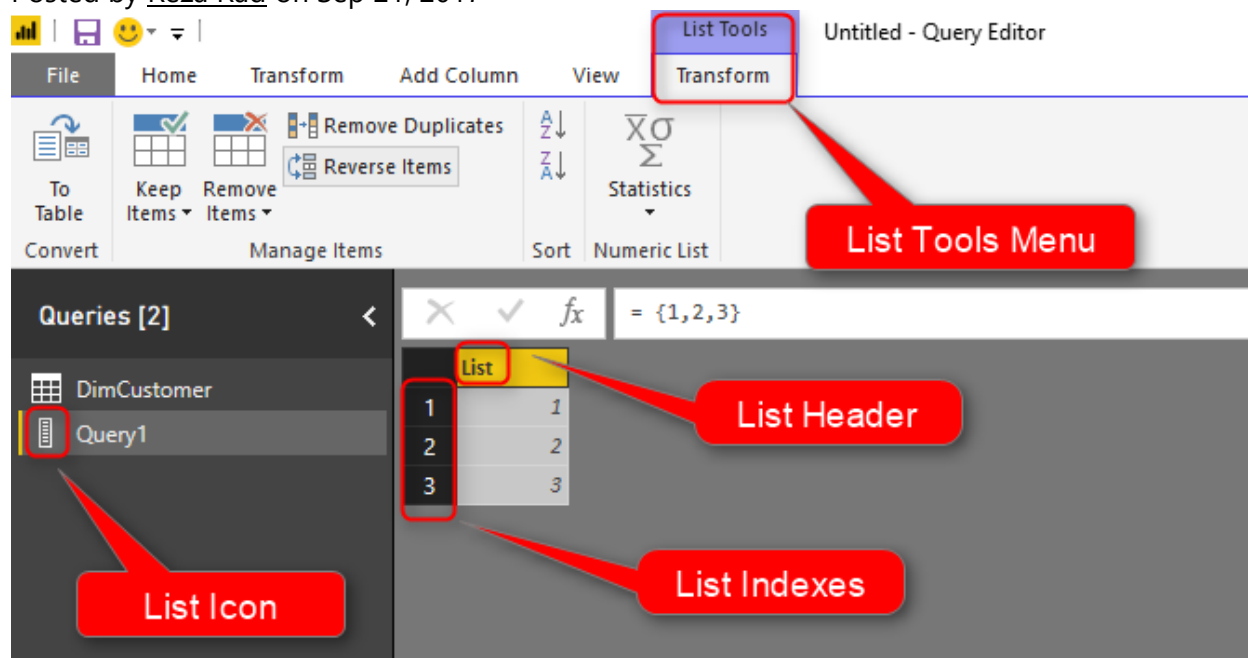
understand the code fully. I have written a blog post, that explains how to [use the #shared keyword to get documentation of all functions in Power Query](#).

In next posts, I'll explain another level of structures in M.



# Basics of Value Structures in M – Power Query Formula Language

Posted by [Reza Rad](#) on Sep 21, 2017



A couple of months ago, I've written a [blog post about the Basics of M](#) and explained few basics about this language. In this post, I'm going to the next step and will explain a few other structure definitions in this language. In this post, you will learn about Tables, Records, Lists, and how to navigate through structures. These structures are main value structures in Power Query and M. Every data value in Power Query is in one of these value structures, and it is important that you can work with these structures. To learn more about Power BI; read [Power BI book from Rookie to Rock Star](#).

## Prerequisite

To understand parts of code from this post, you might need first to read [Basics of M post](#).

## Five Main Value Structures in Power Query

Power Query has five structures for values. Data is either in one of these five structure types. By structure type, I don't mean the data type. I mean the way that data is stored. Sometimes data stored as a simple value like text, date, or number. Sometimes it is a complex value like a table, record, list, or function.

### Primitive Value

Any single part data type considered a primitive value. Examples:

12 – a number value

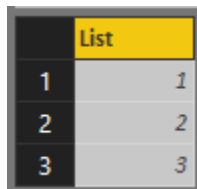
"text sample" – text value

2017/09/21 – date value

null – null value

### List

A list is a structure that has only one column, but multiple rows. Each row identified with an index. Example of a list in Power Query window;



List	
1	1
2	2
3	3

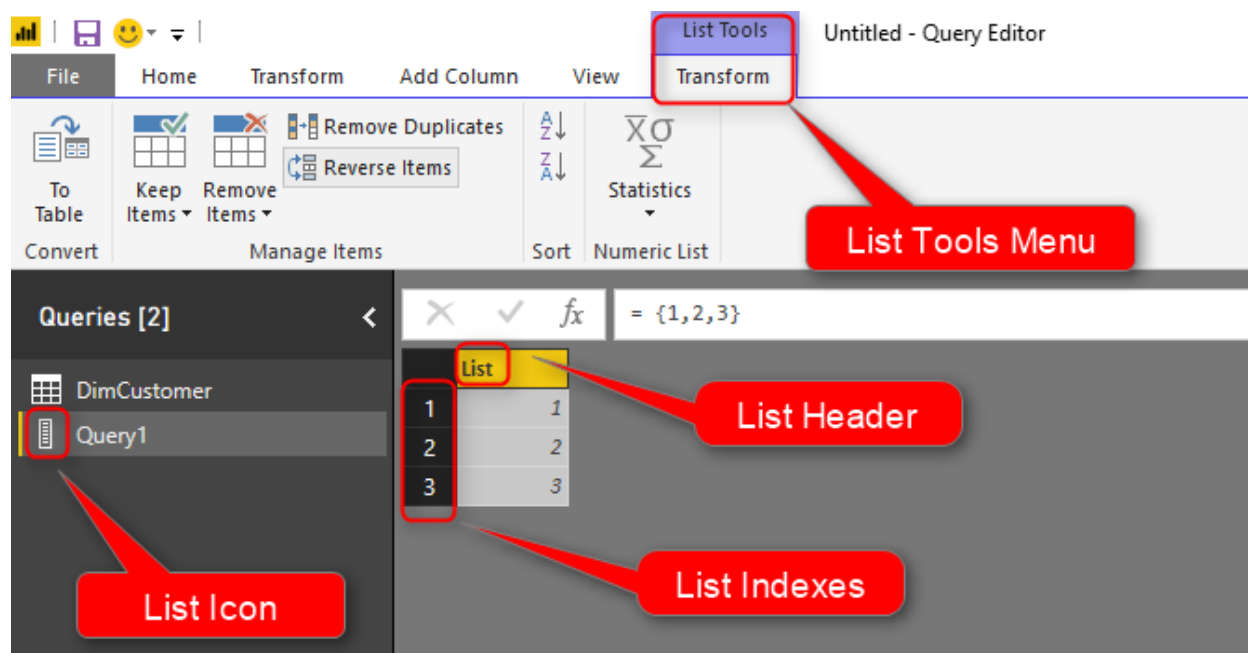
The M script to define a list is as below;

```
1 Source = {1,2,3}
```

List Definition is always started with { and ends with }, items placed in between with a comma separator;

```
1 List = {<item 1>,<item 2>,<item 3>}
```

There are some ways to understand if a structure is a list or not. in the screenshot below all mentioned;



- List Icon: There will be a specific icon for the list in the Queries pane.
- List Tools: When you select a list, you will see List Tools menu. This menu gives you some options later on for changing and transforming the list.
- List Header: At the top of the list column, you will see "List" name.
- List Indexes: Every row in the list should have a numeric index starts from zero.

A list can have items that are different in data types. Here is an example;

= {1,"text value", #date(2017,9,21)}	
List	
1	1
2	text value
3	21/09/2017

Here is the line definition for this list;

1 Source = {1,"text value", #date(2017,9,21)}

## Record

A record is a structure with a single row, but multiple columns. However, the way that record is showed in Query editor is vertical! The main reason is that scrolling to the right is always harder than scrolling to down. So, Record is

only visualized similarly to list. However, it is a totally different structure. This is how a record looks like in Power Query;

✕	✓	<i>fx</i>	= [Column 1=1,Column 2=2]
Column 1	1		
Column 2	2		

As you can see the record showed vertically, but every column header is visible there.

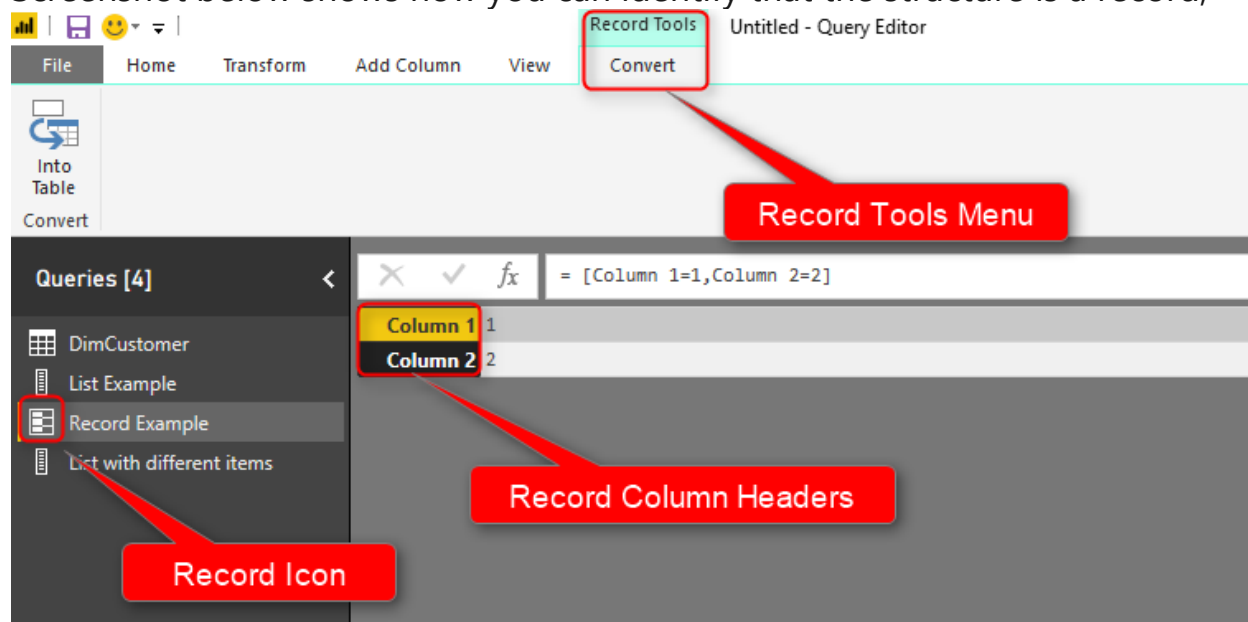
Here is the script for defining the record;

```
1 Source = [Column 1=1,Column 2=2]
```

Record definition always starts with [ and ends with ]. for every column you will have the column name before = sign, and the value of that after the = sign.

```
1 Record = [ Column 1 = <value>, Column 2 = <value> ]
```

Screenshot below shows how you can identify that the structure is a record;



- Record Icon: There is a specific Icon that determines the object is a Record.
- Record Column Headers: You can see column headers in the record. In a list, you can see only numbers, but in a record, you see column names.
- Record Tools Menu: Every time you select a record object, you will see Record Tools which gives you the option to convert it to a table.

A record also can have different types of items. It is different columns.

✕	✓	<i>fx</i>	= [Column Text="text value", Column Number=12]
Column Text	text value		
Column Number	12		

Here is the definition of the record above;

```
1 Source = [Column Text="text value", Column Number=12]
```

## Table

A table is a structure that is most well known among others. A table is a combination of multiple rows and multiple columns. Here is a table sample;

	ABC 123 Column A	ABC 123 Column B
1	1	10
2	2	20

To create a table through M script, you can run a script such as below;

```
1 Source = #table(
2 {"Column A","Column B"},
3 {
4 {1,10},
5 {2,20}
6 }
7 )
```

Table definition always starts with #table, then inside the bracket, you have to set of brackets; one set for defining headers, and the other set for all row values. Here is how this works;

```
1 Source = #table(
2 {"Column A","Column B"}, // all column headers
3 // start of row values
4 {1,10}, // row one
5 {2,20} // row two
6 // end of row values
7 )
```

If you see a table, then you can recognize it immediately. Because the table is the only structure that has multiple rows and multiple columns in it.

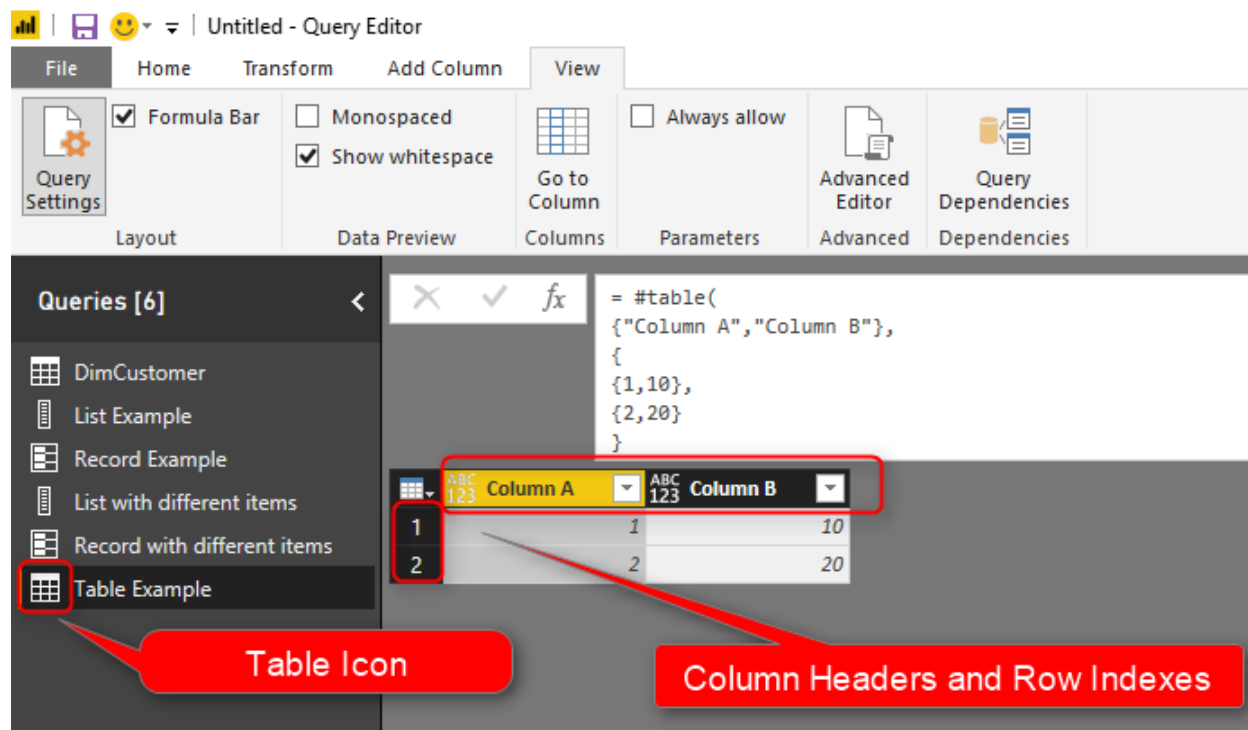


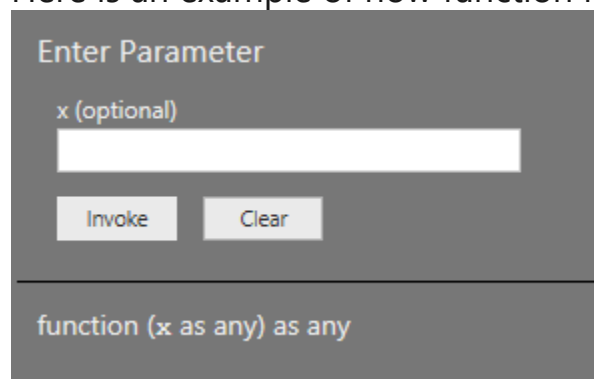
Table Icons shows that this is a table, and multiple columns and rows are only possible in a table.

You can also have a table with different items in each cell.

## Function

A function is a data type that performs an operation and gives you a result.

Here is an example of how function looks like in the query editor window;



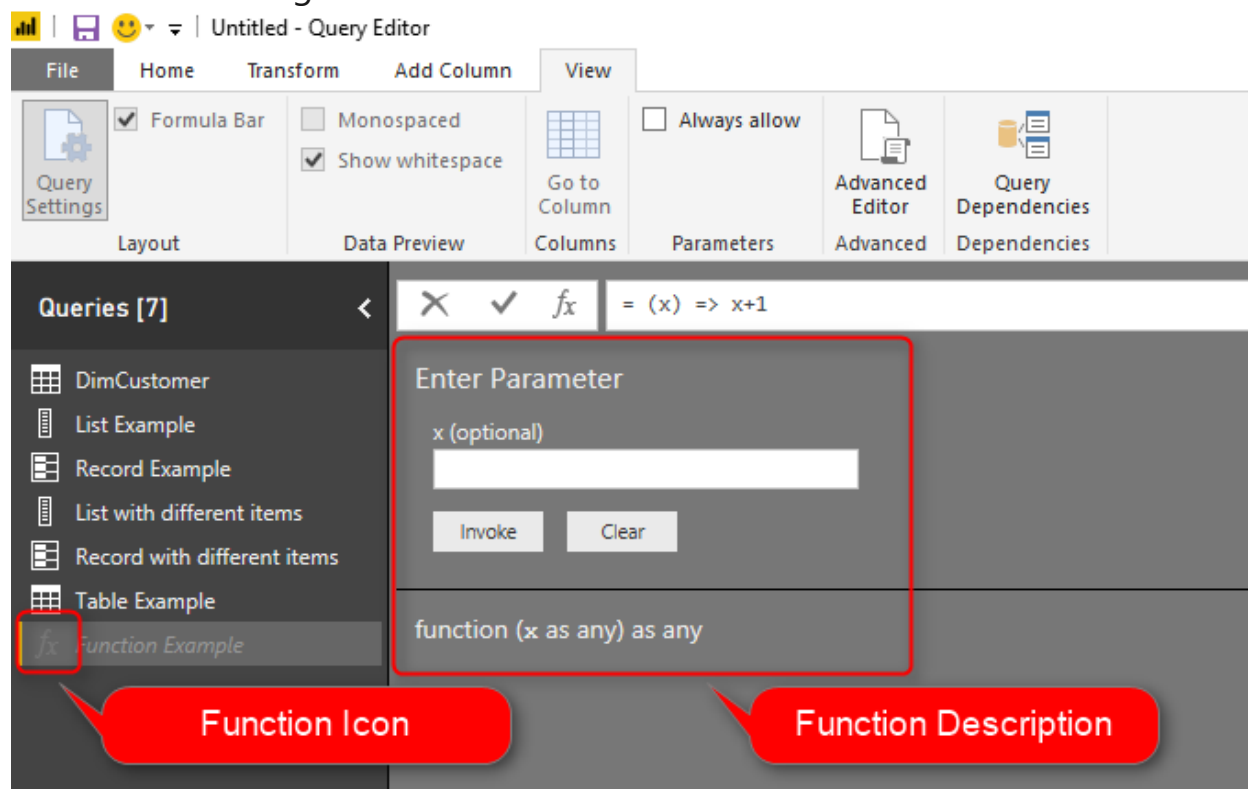
A function can be defined based on Lambda language as below;

1 Source = (x) => x+1

Function Definition has many details in it. I'll talk about it separately in another post. For now, just knowing this is enough that function has an input and output. Input and output are separated from each other with => signs.

1 Source = (<input of function>) => <output of function>

You can easily understand a function based on its specific icon, and also the function call dialog box.



A function is one of the most powerful features in Power Query, and can't be explained only in few paragraphs. Stay tuned, and I'll write more details about functions in other posts separately.

## Navigating Through List

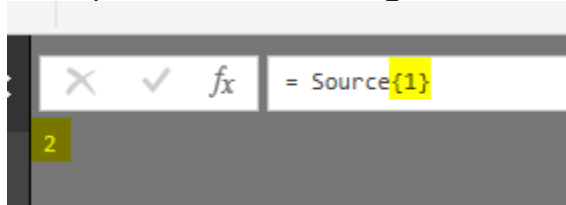
Now that you know what list is, and how to define it, let's look at how you can navigate through the list. Each list item has a row index. You can easily navigate to that item using that index.

```
1 let
2   Source = {1,2,3},
3   Source1 = Source{1}
4 in
5   Source1
```

For navigating through a list, simply use a bracket and put the index of the item in the bracket. Now here is the tricky part:

### *Index starts from zero*

When you look at the list in the query editor window, the index starts from 1. However, the actual index starts from zero. So if you want to drill down to a specific item in the list, you have to use a zero-based index to get to that. The example above will navigate to the **second item** in the list.



So this is simply the syntax to navigate through the list;

```
1 List(<item index starting from zero>)
```

## List Functions

There are also some functions that work on a list. For example, you can get the count of items in a list, with `List.Count()` function.



I will write more about List functions in the future separately.

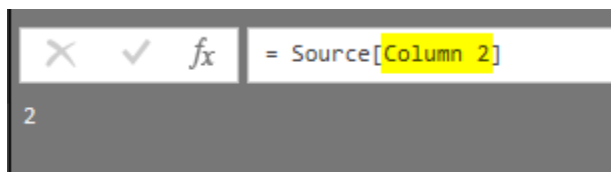
## Navigating Through Record

To navigate through a record, you need to use the column name for that record.

```
1 let
2   Source = [Column 1=1,Column 2=2],
3   #"Column 2" = Source[Column 2]
4 in
5   #"Column 2"
```

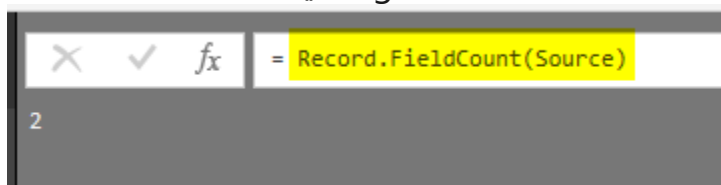
You can simply put the column name inside bracket [ and ], and as a result, you will have the item;





## Record Functions

The Record also has a lot of functions. For example, you can use the Record.FieldCount() to get the count of columns in a record.



I will write another post about functions for Record.

## Navigating Through Table

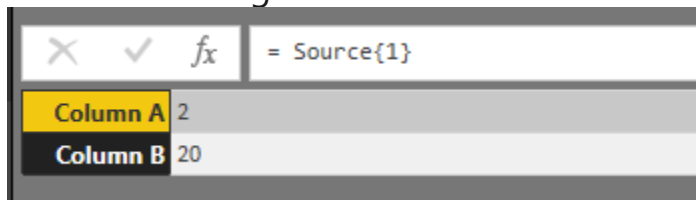
Navigating through a table happens a lot in real world scenarios when you want to drill down into a specific area of the table. For the table, you can use different methods to navigate;

### Navigate with Record Index: Drill Down to Record

You can navigate to any records in the table simply with putting the record's index (zero-based) inside { and }. Here is an example;

```
1 let
2     Source = #table(
3     {"Column A","Column B"},
4     {
5     {1,10},
6     {2,20}
7     }
8     ),
9     record=Source{1}
10 in
11     record
```

This would bring the second row of the table as a Record structure;



## Navigate with Column Name: Drill Down to List

You can also fetch every column of the table with referring to that column name inside [ and ]. The result would be a list.

```

1 let
2     Source = #table(
3     {"Column A","Column B"},
4     {
5     {1,10},
6     {2,20}
7     }
8     ),
9     #"Column A" = Source[Column A]
10 in
11     #"Column A"
```

Navigating to a column will result in a list;



fx	
=	Source[Column A]
<b>List</b>	
1	1
2	2

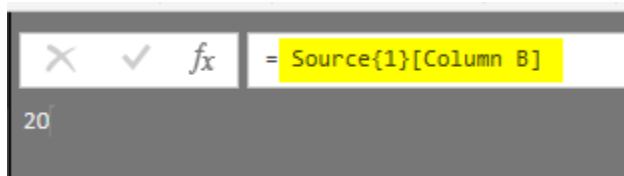
## Navigate with Row Index and Column Name: Drill Down to individual Value

Sometimes you want to drill down to a specific cell, for that you need row index (zero-based), and column name both.

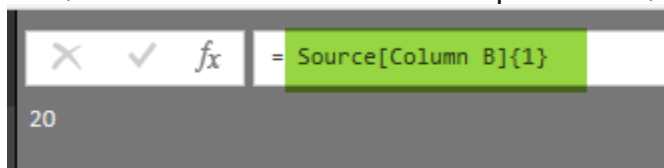
```

1 let
2     Source = #table(
3     {"Column A","Column B"},
4     {
5     {1,10},
6     {2,20}
7     }
8     ),
9     #"Column B" = Source{1}[Column B]
10 in
11     #"Column B"
```

The script above navigates to the second row (Index 1 belongs to the second row), and column B. Result would be a primitive value in this case;



You can also do this navigation the other way around. Navigate to column first, and then to row. An example below;



### Navigate with Filter Criteria

For the majority of the cases, you want to navigate based on criteria. For example, you want to get the Column B value of the row that Column A value for that row is something specific. Like writing a SQL Statement clause. In the table below. For example; we want to navigate to the column B where column A is 1. You cannot do that search based on an index. However, you can search based on criteria.

	ABC 123	Column A	ABC 123	Column B
1		1		10
2		2		20

in the table below. For example; we want to navigate to the column B where column A is 1. You cannot do that search based on an index. However, you can search based on criteria.

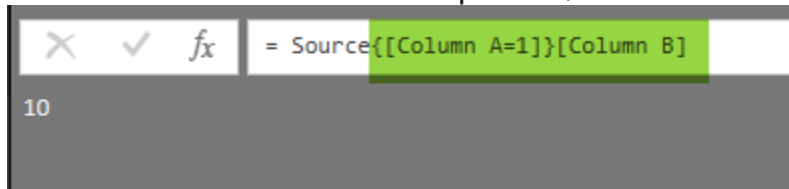
Here is how you can search;

```

1 let
2     Source = #table(
3         {"Column A","Column B"},
4         {
5             {1,10},
6             {2,20}
7         }
8     ),
9     filtered=Source{[Column A=1]}[Column B]
10 in
11     filtered

```

The result would be 10 as expected;



The structure simply is as below;

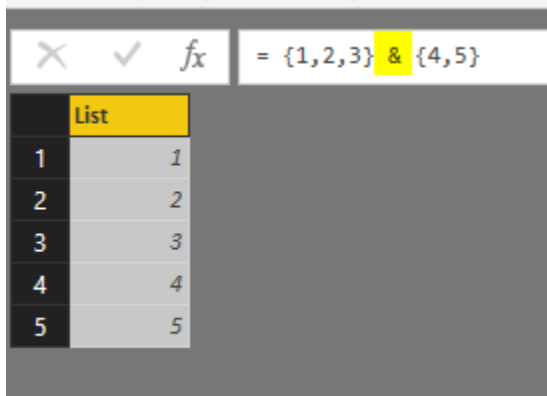
1 = <Table Name>{[Column Name="criteria 1", Column Name="criteria 2"]}[Output Column]

This is similar to SQL Where and Select Clause.

## Concatenate List or Records

You can concatenate list or records together with ampersand (&) character.

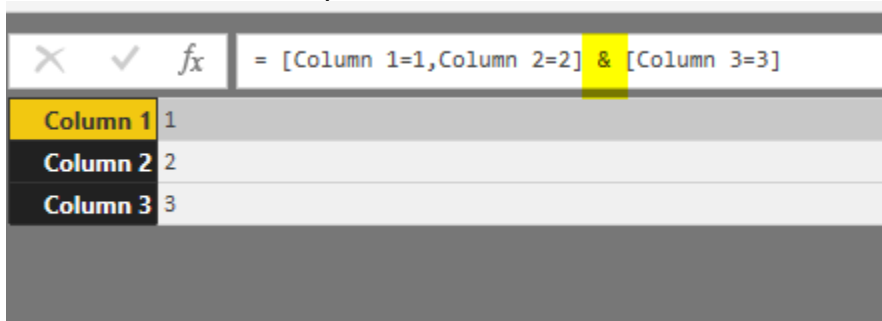
Here is an example for list concatenation;



The screenshot shows the Power Query formula bar with the formula `= {1,2,3} & {4,5}`. Below the formula bar, a table is displayed with the following data:

	List
1	1
2	2
3	3
4	4
5	5

and here is an example for record concatenation;



The screenshot shows the Power Query formula bar with the formula `= [Column 1=1,Column 2=2] & [Column 3=3]`. Below the formula bar, a table is displayed with the following data:

Column 1	Column 2	Column 3
1	2	3

For concatenating tables; you need to use either Append or Merge. [I explained fully in details about Append and Merge in this post.](#)

## Summary

In this post, you learned some of the basics about different value structures in Power BI; Primitive values, list, record, table, and function. You learned how to navigate through list, record, and table with different methods with M script. In future posts, I'll explain more details about each value structure and functions related to that structure.

# Power Query Formula Language M: Table Functions Part 1

Posted by [Reza Rad](#) on Feb 10, 2014

	CustomerId	Date	TableA.Total	TableB.Total
1	1	1/1/2014	5	5
2	1	1/2/2014	0	15
3	1	1/3/2014	5	15
4	1	1/4/2014	80	15
5	2	1/1/2014	0	80

In a [previous post](#), you've learned about the Formula language of the Power Query known as "M". You've learned that M is a functional language that applies data transformations through the script. In examples of the previous post, you've learned some of the table functions such as Table.AddColumn, Table.Sort, Table.Join, and Table.PrefixColumns. In this post, we will discover more table functions through an example.

In this post you will learn these functions:

*Table.AddColumn*

*Table.RemoveColumns*

*Table.ReorderColumns*

*Table.SelectColumns*

*Table.Sort*

*Table.ReplaceValue*

*Table.FillDown*

*Table.AddIndexColumn*

**Table.RenameColumns**

This function renames one or more columns of a table to desired name(s).

Let's apply this function on a sample table. Here is the sample table generated from the previous example:

	TableA.CustomerId	TableA.Date	TableA.Total	CustomerId	Date	Total
1	1	1/1/2014	5	1	1/1/2014	5
2	null	null	null	1	1/2/2014	15
3	null	null	null	2	1/1/2014	80
4	1	1/3/2014	5	null	null	null
5	1	1/4/2014	80	null	null	null

this is the script that generates above table:

*let*

```

    TableA = #table({"CustomerId", "TranDate", "TranCount"},
    {
        {1,DateTime.FromText("2014-01-01 01:00:00.000"),10},
        {1,DateTime.FromText("2014-01-01 02:00:00.000"),5},
        {1,DateTime.FromText("2014-01-03 01:00:00.000"),5},
        {1,DateTime.FromText("2014-01-04 02:00:00.000"),80}
    }),
    TableB = #table({"CustomerId", "TranDate", "TranCount"},
    {
        {1,DateTime.FromText("2014-01-01 02:00:00.000"),10},
        {1,DateTime.FromText("2014-01-01 03:00:00.000"),5},
        {1,DateTime.FromText("2014-01-02 01:00:00.000"),20},
        {1,DateTime.FromText("2014-01-02 03:00:00.000"),15},
        {2,DateTime.FromText("2014-01-01 01:00:00.000"),5},
        {2,DateTime.FromText("2014-01-01 02:00:00.000"),80}
    }),
    TableATransformed=Table.Sort(
        Table.AddColumn(TableA,"Date",each Date.From([TranDate]))
        ,{"CustomerId","TranDate"}
    )

```

```

    ),
    TableBTransformed=Table.Sort(
        Table.AddColumn(TableB,"Date",each Date.From([TranDate]))
       ,{"CustomerId","TranDate"}
    ),
    TableAGrouped=Table.Group(TableATransformed,{"CustomerId","Date"},{"Total",each List.Last([TranCount])}),
    TableBGrouped=Table.Group(TableBTransformed,{"CustomerId","Date"},{"Total",each List.Last([TranCount])}),
    ResultTable=Table.Join(Table.PrefixColumns(TableAGrouped,"TableA"),{"TableA.CustomerId","TableA.Date"},TableBGrouped,{"CustomerId","Date"},JoinKind.FullOuter)
in
    ResultTable

```

Now we want to rename last three column of the table with a new prefix "TableB.". Here is how we can do that with Table.RenameColumns

```

let
    ....
    ,RenamedTable=Table.RenameColumns(ResultTable,{
        {"CustomerId","TableB.CustomerId"},{"Date","TableB.Date"},{"Total","TableB.Total"} })
in
    RenamedTable

```

The result would be the below table

	TableA.CustomerId	TableA.Date	TableA.Total	TableB.CustomerId	TableB.Date	TableB.Total
1	1	1/1/2014	5	1	1/1/2014	5
2	null	null	null	1	1/2/2014	15
3	null	null	null	2	1/1/2014	80
4	1	1/3/2014	5	null	null	null
5	1	1/4/2014	80	null	null	null



Here is the structure of Table.RenameColumns

*Table.RenameColumns(<table>,( {<column name before change>,<column name after change>},{repeat for more columns} } )*

### Table.AddColumn

This function adds a column to the table. You would assign the name of the new column and the script to generate a new column. The new column generator can be a static value or an expression.

In this example, we want to add three columns that be generated based on expressions from existing columns of RenamedTable. We want a new CustomerId Column to be generated from TableA.CustomerId, but if it is null, then it should get its value from TableB.CustomerId.

Here is the script to generate the new CustomerId Column:

*,CustomerColumnAddedTable=Table.AddColumn(RenamedTable,"CustomerId",each if [TableA.CustomerId] is null then [TableB.CustomerId] else [TableA.CustomerId])*  
in

*CustomerColumnAddedTable*

Result table showed below:

	TableA.CustomerId	TableA.Date	TableA.Total	TableB.CustomerId	TableB.Date	TableB.Total	CustomerId
1	1	1/1/2014	5	1	1/1/2014	5	1
2	null	null	null	1	1/2/2014	15	1
3	null	null	null	2	1/1/2014	80	2
4	1	1/3/2014	5	null	null	null	1
5	1	1/4/2014	80	null	null	null	1

As you see in the above expression we used IF structure which is the very handy conditional structure in M. the <IF> structure is simply as below:

*If <condition>*

*then <if true>*

*else <if false>*

The good thing about If is that it can be used within expressions to build the table/record/lists, etc (as you seen in this example).

The **EACH** keyword is required to generate expression for each record of the dataset.

here is structure of Table.AddColumn function:

*Table.AddColumn(<table>,<new column name>,<new column generator>,<data type optional>)*

we use AddColumn function to add another column as below:

*,CustomerColumnAddedTable=Table.AddColumn(RenamedTable,"CustomerId",each if [TableA.CustomerId] is null then [TableB.CustomerId] else [TableA.CustomerId])*

*,DateColumnAddedTable=Table.AddColumn(CustomerColumnAddedTable,"Date",each if [TableA.Date] is null then [TableB.Date] else [TableA.Date])*  
in

*DateColumnAddedTable*

The result showed below:

	TableA.CustomerId	TableA.Date	TableA.Total	TableB.CustomerId	TableB.Date	TableB.Total	CustomerId	Date
1	1	1/1/2014	5	1	1/1/2014	5	1	1/1/2014
2	null	null	null	1	1/2/2014	15	1	1/2/2014
3	null	null	null	2	1/1/2014	80	2	1/1/2014
4	1	1/3/2014	5	null	null	null	1	1/3/2014
5	1	1/4/2014	80	null	null	null	1	1/4/2014

## Table.RemoveColumns

This function removes one or more columns from the table.

We can use this function to remove extra columns such as TableA.CustomerId, TableB.CustomerId, TableA.Date, and TableB.Date from the table.

Here is the script:

*,ColumnsRemoved=Table.RemoveColumns(DateColumnAddedTable,{"TableA.CustomerId","TableB.CustomerId","TableA.Date","TableB.Date"})*

in

### *ColumnsRemoved*

Result would be as below:

	TableA.Total	TableB.Total	CustomerId	Date
1	5	5	1	1/1/2014
2	null	15	1	1/2/2014
3	null	80	2	1/1/2014
4	5	null	1	1/3/2014
5	80	null	1	1/4/2014

structure of Table.RemoveColumns is as below:

*Table.RemoveColumn(<table>,{columns to remove separated by comma})*

### **Table.ReorderColumns**

changes order of columns in table.

here is the expression that we used to reorder columns of this table:

*,ColumnsOrdered=Table.ReorderColumns(ColumnsRemoved,{"CustomerId"  
,"Date","TableA.Total","TableB.Total"})*

in

### *ColumnsOrdered*

Result would be as below:

	CustomerId	Date	TableA.Total	TableB.Total
1	1	1/1/2014	5	5
2	1	1/2/2014	null	15
3	2	1/1/2014	null	80
4	1	1/3/2014	5	null
5	1	1/4/2014	80	null

here is the structure of Table.ReorderColumns

*Table.ReorderColumns(<table>,{order of columns as desired in output})*

### **Table.SelectColumns**

This function selects specific columns from a table with the order defined. This function is a combination of Table.RemoveColumns and Table.ReorderColumns.

this script will result same table as above but with only a single function  
Table.SelectColumns instead of using Table.RemoveColumns and  
Table.ReorderColumns.

```
,ColumnsSelected=Table.SelectColumns(DateColumnAddedTable,{"CustomerId",  
"Date","TableA.Total","TableB.Total"})
```

*in*

*ColumnsSelected*

Result would be the same

	CustomerId	Date	TableA.Total	TableB.Total
1	1	1/1/2014	5	5
2	1	1/2/2014	null	15
3	2	1/1/2014	null	80
4	1	1/3/2014	5	null
5	1	1/4/2014	80	null

## Table.Sort

Sorts a table on the desired ordering list

```
,SortedTable=Table.Sort(ColumnsSelected,{"CustomerId","Date"})
```

*in*

*SortedTable*

Result would be as below:

	CustomerId	Date	TableA.Total	TableB.Total
1	1	1/1/2014	5	5
2	1	1/2/2014	null	15
3	1	1/3/2014	5	null
4	1	1/4/2014	80	null
5	2	1/1/2014	null	80

Here is the Table.Sort function structure

```
Table.Sort(<table>,{sorting criteria})
```

## Table.ReplaceValue

This function replace all old values with new value from the specific column in the table.

the expression below would replace all NULL values in TableA.Total with 0.

```
,ValueReplaced=Table.ReplaceValue(SortedTable,null,0,Replacer.ReplaceValue,{"TableA.Total"})
```

*in*

*ValueReplaced*

Result showed below:

	CustomerId	Date	TableA.Total	TableB.Total
1	1	1/1/2014	5	5
2	1	1/2/2014	0	15
3	1	1/3/2014	5	null
4	1	1/4/2014	80	null
5	2	1/1/2014	0	80

Here is the structure of Table.ReplaceValue function

```
Table.ReplaceValue(<table>,<old value>,<new value>,<replacer function>,{column name})
```

replacer function can be Replacer.ReplaceValue or Replacer.ReplaceText

### **Table.FillDown**

This function will fill the value from upper record down to the null values of the records after that.

Here is the example: (we want to fill down the values of columns TableB.Total, that means if a record has a null value for this column, then the value would be fetched from the most recent top record that has a not null value)

```
,FilledDown=Table.FillDown(ValueReplaced,"TableB.Total")
```

*in*

*FilledDown*

result would be as below:

	CustomerId	Date	TableA.Total	TableB.Total
1	1	1/1/2014	5	5
2	1	1/2/2014	0	15
3	1	1/3/2014	5	15
4	1	1/4/2014	80	15
5	2	1/1/2014	0	80

Here is the structure of Table.FillDown function

*Table.FillDown(<table>,<Column>)*

### Table.AddIndexColumn

This function adds an identity auto-increment column to the table. You can specify the column name, initial value, and the increment seed.

This is a sample expression:

*,IndexAdded=Table.AddIndexColumn(FilledDown,"Index",10000,10)*

*in*

*IndexAdded*

Result is as below:

	CustomerId	Date	TableA.Total	TableB.Total	Index
1	1	1/1/2014	5	5	10000
2	1	1/2/2014	0	15	10010
3	1	1/3/2014	5	15	10020
4	1	1/4/2014	80	15	10030
5	2	1/1/2014	0	80	10040

Here is the structure for this function:

*Table.AddIndexColumn(<table>,<column name>,<initial value>,<increment seed>)*

Here is the full script for this example:

*let*

```

    TableA = #table({"CustomerId", "TranDate", "TranCount"},
    {
        {1,DateTime.FromText("2014-01-01 01:00:00.000"),10},
        {1,DateTime.FromText("2014-01-01 02:00:00.000"),5},
    }

```

```

{1,DateTime.FromText("2014-01-03 01:00:00.000"),5},
{1,DateTime.FromText("2014-01-04 02:00:00.000"),80}
}),
TableB = #table({"CustomerId", "TranDate", "TranCount"},
{
{1,DateTime.FromText("2014-01-01 02:00:00.000"),10},
{1,DateTime.FromText("2014-01-01 03:00:00.000"),5},
{1,DateTime.FromText("2014-01-02 01:00:00.000"),20},
{1,DateTime.FromText("2014-01-02 03:00:00.000"),15},
{2,DateTime.FromText("2014-01-01 01:00:00.000"),5},
{2,DateTime.FromText("2014-01-01 02:00:00.000"),80}
}),
TableATransformed=Table.Sort(
    Table.AddColumn(TableA,"Date",each Date.From([TranDate]))
    ,{"CustomerId","TranDate"}
    ) ,
TableBTransformed=Table.Sort(
    Table.AddColumn(TableB,"Date",each Date.From([TranDate]))
    ,{"CustomerId","TranDate"}
    ) ,
    TableAGrouped=Table.Group(TableATransformed,{"CustomerId","Date"},{"T
otal",each List.Last([TranCount])}),
    TableBGrouped=Table.Group(TableBTransformed,{"CustomerId","Date"},{"T
otal",each List.Last([TranCount])}),
    ResultTable=Table.Join(Table.PrefixColumns(TableAGrouped,"TableA"),{"Table
A.CustomerId","TableA.Date"},TableBGrouped,{"CustomerId","Date"},JoinKind.Ful
lOuter)
    ,RenamedTable=Table.RenameColumns(ResultTable,{

```

```
{ "CustomerId", "TableB.CustomerId" }, { "Date", "TableB.Date" }, { "Total", "TableB.Total" } } )
```

```
    , CustomerColumnAddedTable = Table.AddColumn(RenamedTable, "CustomerId",
    each if [TableA.CustomerId] is null then [TableB.CustomerId] else
    [TableA.CustomerId])
```

```
    , DateColumnAddedTable = Table.AddColumn(CustomerColumnAddedTable,
    "Date", each if [TableA.Date] is null then [TableB.Date] else [TableA.Date])
```

```
    , ColumnsRemoved = Table.RemoveColumns(DateColumnAddedTable, { "TableA.CustomerId",
    "TableB.CustomerId", "TableA.Date", "TableB.Date" })
```

```
    , ColumnsOrdered = Table.ReorderColumns(ColumnsRemoved, { "CustomerId",
    "Date", "TableA.Total", "TableB.Total" })
```

```
    , ColumnsSelected = Table.SelectColumns(DateColumnAddedTable, { "CustomerId",
    "Date", "TableA.Total", "TableB.Total" })
```

```
    , SortedTable = Table.Sort(ColumnsSelected, { "CustomerId", "Date" })
```

```
    , ValueReplaced = Table.ReplaceValue(SortedTable, null, 0, Replacer.ReplaceValue,
    { "TableA.Total" })
```

```
    , FilledDown = Table.FillDown(ValueReplaced, "TableB.Total")
```

```
    , IndexAdded = Table.AddIndexColumn(FilledDown, "Index", 10000, 10)
```

in

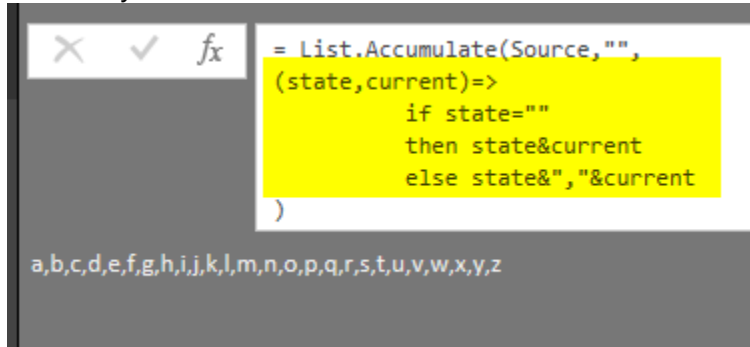
*IndexAdded*

There are many other table functions in Power Query which we will go through them with more samples in next blog posts.



# List.Accumulate Hidden Gem of Power Query List Functions in Power BI

Posted by [Reza Rad](#) on Dec 12, 2017

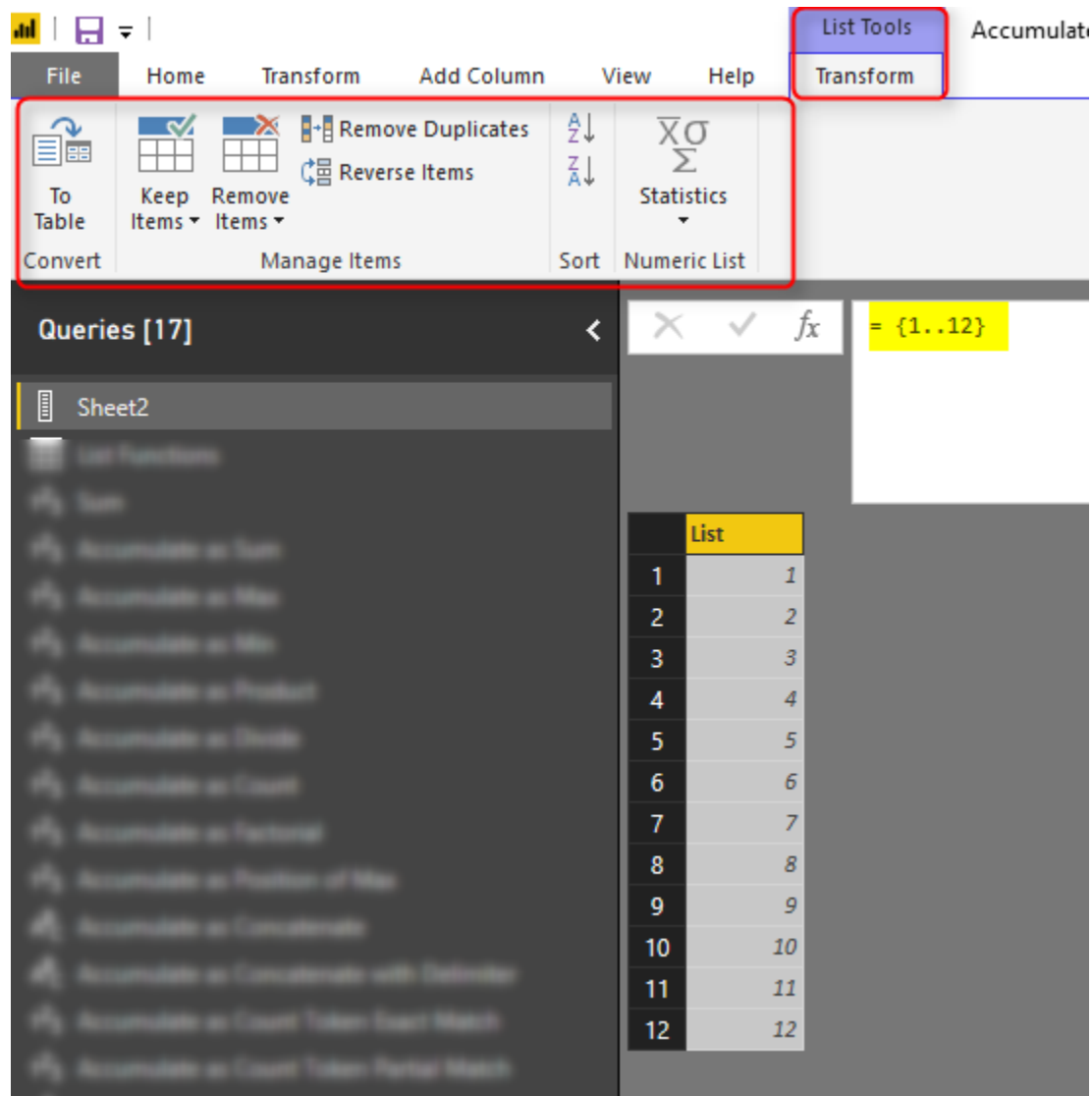


There are some List transformations available in Power Query's graphical interface. However, the number of functions in the graphical interface is very limited. In this post, I'm going to explain about a function that is powerful and is not yet listed in the graphical interface. List.Accumulate is a function that loops through items of a list and applies a transformation. This function at the time of writing this post is only available through Power Query M scripting. If you like to learn more about Power BI, read [Power BI book from Rookie to Rock Star](#).

## List Transformations in Graphical Interface of Power Query

If you have a list, you can see what transformations are available to be applied to it using the graphical interface. You can create a list with a simple M script as below;

```
1 = {1..12}
```



This code generates a list of numbers from 1 to 12. Now you can see in the top menu under List Tools, that there are some transformations available for List. These items are very few options such as; convert to a table, keep items, remove items, etc. Altogether considering all options in every element this list is not more than 20 functions. However, let's look at the number of List functions in M script.

## List Functions in M; Power Query Formula Language

I have written previously about the [usage of the #shared key to finding all functions available in M](#). you can then filter it to only "List." Functions and then

you will end up with a bit list of functions; 69 functions! More than three times of what you see in the graphical interface.

	ABC Name	ABC 123 Value
1	List.Accumulate	Function
2	List.AllTrue	Function
3	List.Alternate	Function
4	List.AnyTrue	Function
5	List.Average	Function
6	List.Buffer	Function
7	List.Combine	Function
8	List.Contains	Function
9	List.ContainsAll	Function
10	List.ContainsAny	Function
11	List.Count	Function
12	List.Covariance	Function
13	List.DateTimeZones	Function
14	List.Datetimes	Function
15	List.Dates	Function
16	List.Difference	Function
17	List.Distinct	Function
18	List.Durations	Function
19	List.FindText	Function
20	List.First	Function
21	List.FirstN	Function
22	List.Generate	Function
23	List.InsertRange	Function
24	List.Intersect	Function
25	List.IsDistinct	Function
26	List.IsEmpty	Function
27	List.Last	Function
28	List.LastN	Function
29	List.MatchesAll	Function

Some of these functions Some of the functions in this list are very useful and powerful. Example of those functions is List.Dates, List.Numbers, List.Generate, List.Accumulate and many others. We cannot go through all functions in one blog post. In this post, I'll be covering List.Accumulate and in future posts; I'll talk about other functions.

## List.Accumulate Function

List.Accumulate is a function that can easily save some steps in your Power Query transformations, instead of applying multiple steps, you can simply use List.Accumulate to overcome what you want. List.Accumulate function loops through the list and accumulate value as a result. This function usually needs three parameters; the list itself, seed, and accumulator. Here are parameters explained in details;

- List; the list that we want to apply the transformation to it.
- Seed; is the initial value.
- Accumulator; is a function. This function determines what accumulation calculation happens on items of the list. The way that this function is defined is exactly [the way that you write a function in Power Query M script using Lambda expressions](#).

Best way to learn about seed and accumulator is through some examples, let's apply some transformations with List.Accumulate and see how these two parameters are working.

### Accumulate to Calculate Sum

The sum is a function that is accumulating every two values in the list till the end of the list. If you want to write Sum with List.Accumulate, you can do it with this expression:

```
1 = List.Accumulate(Source,0,(state,current)=>state+current)
```

The function part of this expression is: `(state, current)=>state+current`

state is the value accumulated in the calculation. current is the current item in the list. seed is the initial value of the state

Let's see how the calculation works. To clarify it more in details, I explained the value of the state, current, and accumulator in every step;

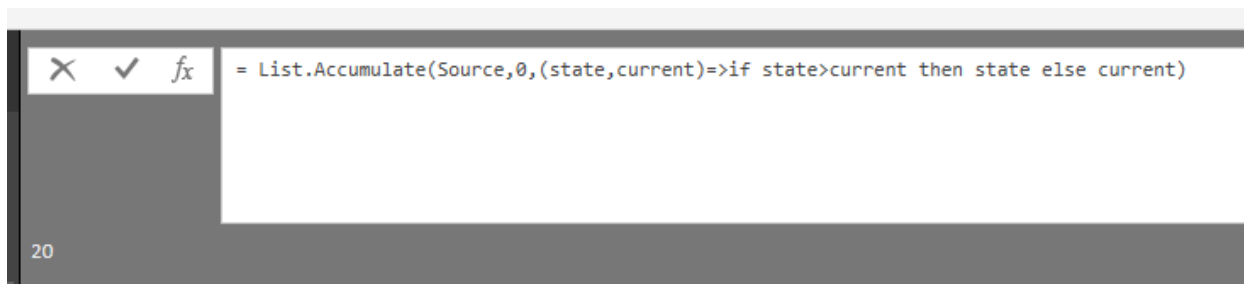
List	state	current	(state,current)=>state+current
1	0	1	1
2	1	2	3
3	3	3	6
4	6	4	10
5	10	5	15
6	15	6	21
7	21	7	28
8	28	8	36
9	36	9	45
10	45	10	55
11	55	11	66
12	66	12	78
13	78	13	91
14	91	14	105
15	105	15	120
16	120	16	136
17	136	17	153
18	153	18	171
19	171	19	190
20	190	20	210

List.Accumulate loops through every item in the list and run accumulator function. The very first time, the state value is equal to seed. Which in this case is zero. Current is the current value in the list. For the very first item that value is 1. so accumulator result is  $state+current=0+1=1$ . this value then will be the state of the next item on the list. For the next item, the state is 1 (calculated from the previous row), and current is 2.  $state+current$  becomes  $1+2=3$ . This process continues through the whole list, so the final state value for a list from 1 to 20, becomes 210, which is equal to the sum of those values. In every row of the list, we added that to the previous row's result.

## Accumulate to Calculate Max

Learning how the accumulate function can cover basic tasks, help you to understand how accumulator function works. For applying Max, you need to compare every two items in the list and pick the one which is bigger. Here is the script;

$1 = \text{List.Accumulate}(\text{Source}, 0, (\text{state}, \text{current}) \Rightarrow \text{if } \text{state} > \text{current} \text{ then } \text{state} \text{ else } \text{current})$



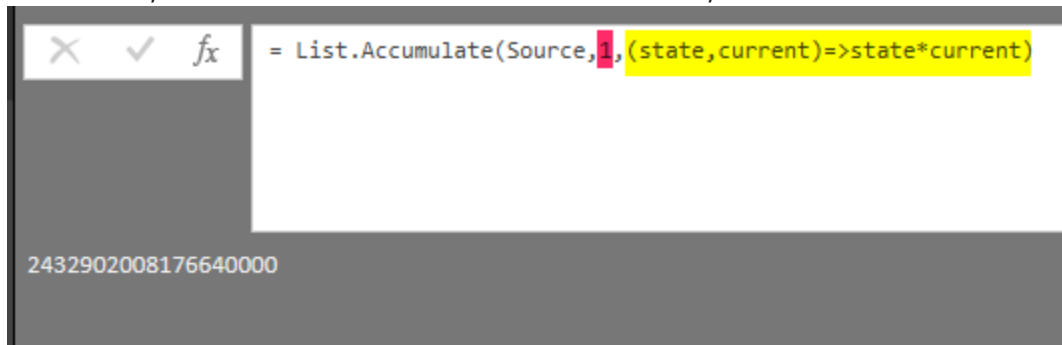
remember the same thing about the state, current, and seed; this is how the calculation works;

List	state	current	(state,current)=>if state>current then state else current
1	0	1	1
2	1	2	2
3	2	3	3
4	3	4	4
5	4	5	5
6	5	6	6
7	6	7	7
8	7	8	8
9	8	9	9
10	9	10	10
11	10	11	11
12	11	12	12
13	12	13	13
14	13	14	14
15	14	15	15
16	15	16	16
17	16	17	17
18	17	18	18
19	18	19	19
20	19	20	20

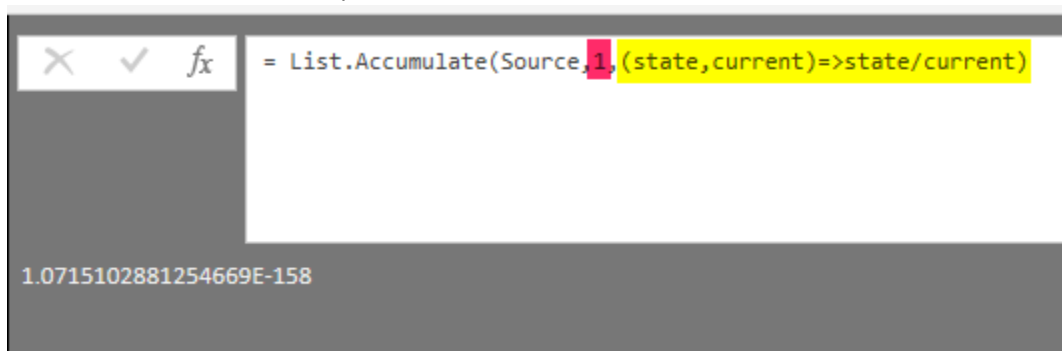
the function **(state, current) => if state > current then state else current** will run on every row and give you the result. Remember that seed value should be a value less than any other values for this case.

## Accumulate as Product or Divide

you can use the same logic with a different accumulator to calculate Product or Divide, Here is the calculation for Product;

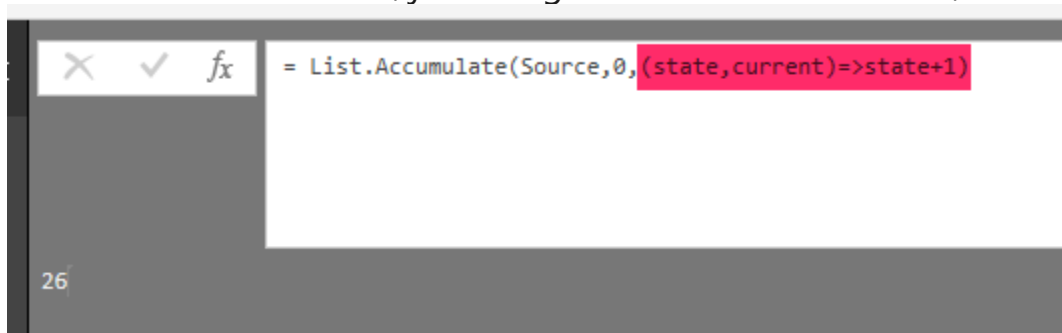


for Product or Divide, you need to set the seed as 1, because if it is zero, then divide or multiply considering zero will end up zero always. Here is the calculation for Divide;



## Accumulate as Count

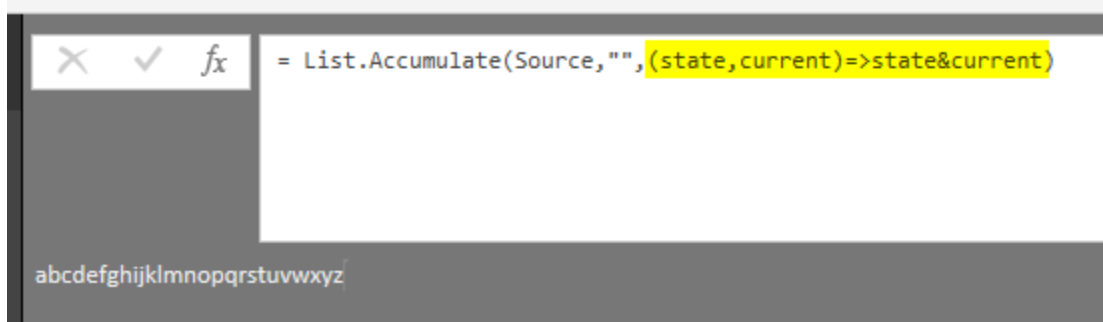
Count of a list is the number of items in a list. This can be achieved with no need of the current item, just using seed and state as below;



in every row, the value would be plus one of the state value from previous row's calculation. As a result, you get the count.

## Accumulate as Concatenate (with a delimiter or without)

So far, we applied to accumulate on lists which had a number in every item. Now, let's apply it to a list of text items. To concatenate two items you need to add them one after each other with concatenation character which is the ampersand (&).

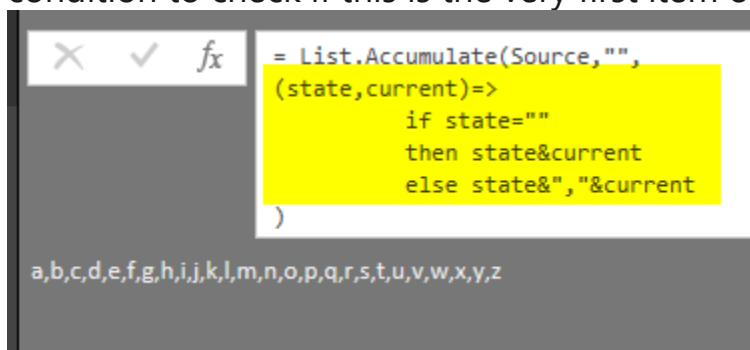


```
= List.Accumulate(Source, "", (state, current) => state & current)
```

abcdefghijklmnopqrstuvwxyz

This worked on a list of items which are text values {"a".."z"}, and the result is the concatenation of all items in the list. Please note that the seed, in this case, is an empty text.

If you want to add a delimiter between items, you can add the delimiter plus a condition to check if this is the very first item or not.



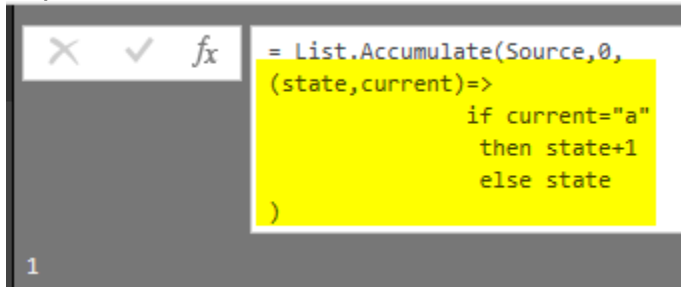
```
= List.Accumulate(Source, "",  
  (state, current) =>  
    if state=""  
    then state & current  
    else state & "," & current  
)
```

a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z



## Accumulate as Count Token Exact Match

Now that you've learned the logic of the accumulator, you can apply it to do any expressions. For example, if you want to calculate the count of items in the list which their value matches exactly with a value, you can write this expression;



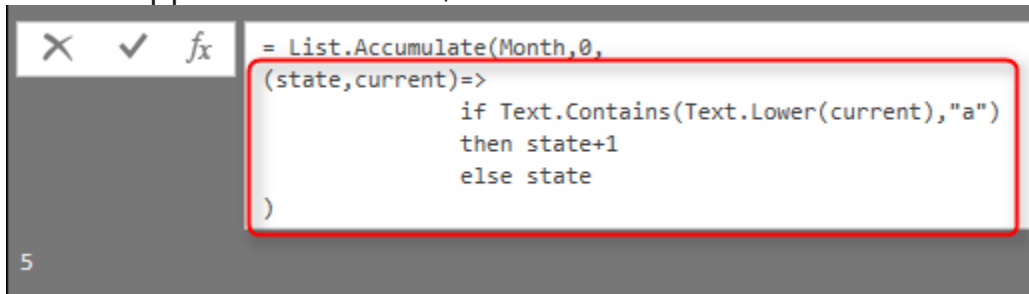
```
= List.Accumulate(Source,0,
(state,current)=>
    if current="a"
    then state+1
    else state
)
```

```
1 = List.Accumulate(Source,0,
2 (state,current)= >
3     if current="a"
4     then state+1
5     else state
6 )
```

the logic is simple, if the item matches with "a" (which in this case is our token), then count it, otherwise don't

## Accumulate as Count Token Partial Match

Similar to the previous calculation. However this time we want to count the item, even it partially matches the text. This can be done with the help of Text.Contains function. and because of Text.Contains might not find the lower case or upper case matches, we convert it to lower case beforehand.



```
= List.Accumulate(Month,0,
(state,current)=>
    if Text.Contains(Text.Lower(current),"a")
    then state+1
    else state
)
```

```

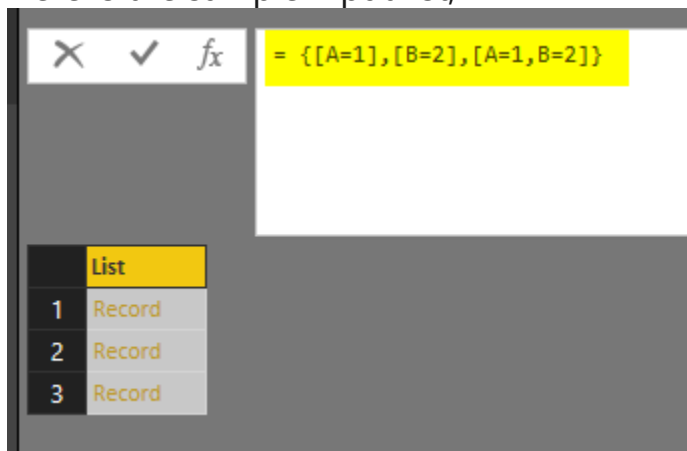
1 = List.Accumulate(Month,0,
2 (state,current)=>
3     if Text.Contains(Text.Lower(current),"a")
4     then state+1
5     else state
6)

```

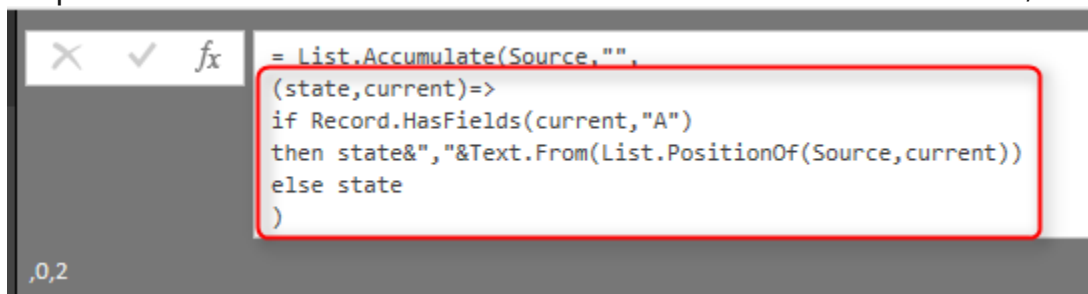
## Accumulate as Conditions on Records

So far, we went through a lot of use cases and examples of List.Accumulate function. You understand that this function can be so powerful and useful in many scenarios. However, the main use cases of List.Accumulate is to apply to scenarios which other functions cannot resolve easily. List.The sum might be better used than List.Accumulate which only calculates the sum. However, there are many scenarios that many steps are needed to get the result you want with normal functions. In those cases, List.Accumulate is your friend. For example; consider the situation that you have a list, and this list is a list of records! Every record might have a different set of fields; you want to fetch only records that have a specific field on their list, and get their position as a concatenated result. This process, using other list functions might take some steps, however, with List.Accumulate that is easy.

Here is the sample input list;



As you can see the list includes records, and to find out what each record has, you need to expand it. A list of records cannot be expanded because it will add the number of columns and list can have only one column. So you have to convert it to a table, and then expand it. As you feel now; there are some steps required to get the result we wanted. However, List.Accumulate can be a big help in this scenario. Here is the calculation with List.Accumulate;



```
1 = List.Accumulate(Source, "",
2 (state, current) =>
3 if Record.HasFields(current, "A")
4 then state & ", " & Text.From(List.PositionOf(Source, current))
5 else state
6 )
```

Record.HasFields used to determine if a record contains a field ("A" in this example).

List.PositionOf used to get the position of that record which satisfy the criteria above.

## Summary

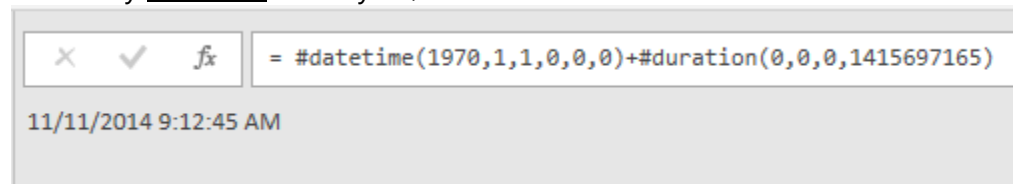
This post explained how List.Accumulate works. List.Accumulate is a very powerful function, that can easily save many steps in your Power Query transformations. In this post, you've learned basics of this function with using it for simple operations such as Sum, Divide, Product, Max, Count, etc. You also learned that the main power of this function is when basic functions cannot operate easily. You learned that the accumulator function gives you full

power to write exactly what you want. In future posts, I'll write about other List functions that can be very powerful, but you still don't have it in the graphical interface.

Have you liked the List.Accumulate function as I do? If yes, please share your story that how this function can be helpful for you in the comments below.

# Power Query; Convert Time Stamp to Date Time

Posted by [Reza Rad](#) on May 31, 2016



Power Query has a number of Date and Time functions that convert values to date and time. However, I haven't found a function that converts a timestamp value. Fortunately, it is easy to calculate a date-time from a timestamp value. In this post, I'll explain an easy way of converting the timestamp to date time. To Learn more about Power Query read [Power BI online book](#); from Rookie to Rock Star.

## What is Timestamp

The timestamp is a whole number value, which is the number of seconds from date 1970-01-01 00:00:00. For example; timestamp 100 means 1970-01-01 00:01:40, or timestamp 86400 means 1970-01-02 00:00:00. So the calculation is easy; We have to add the timestamp as seconds to the date/time 1970-01-01 00:00:00.

## Power Query Convert Timestamp to Date Time

Once we know what is definition of timestamp, and how to calculate date/time from it; easily we can use **#duration(0,0,0,<timestamp value>)** to show duration in seconds, and add it to the **#datetime(1970,1,1,0,0,0)** which is date time 1970-01-01 00:00:00.

So as a result, here is the code to convert timestamp to date time;

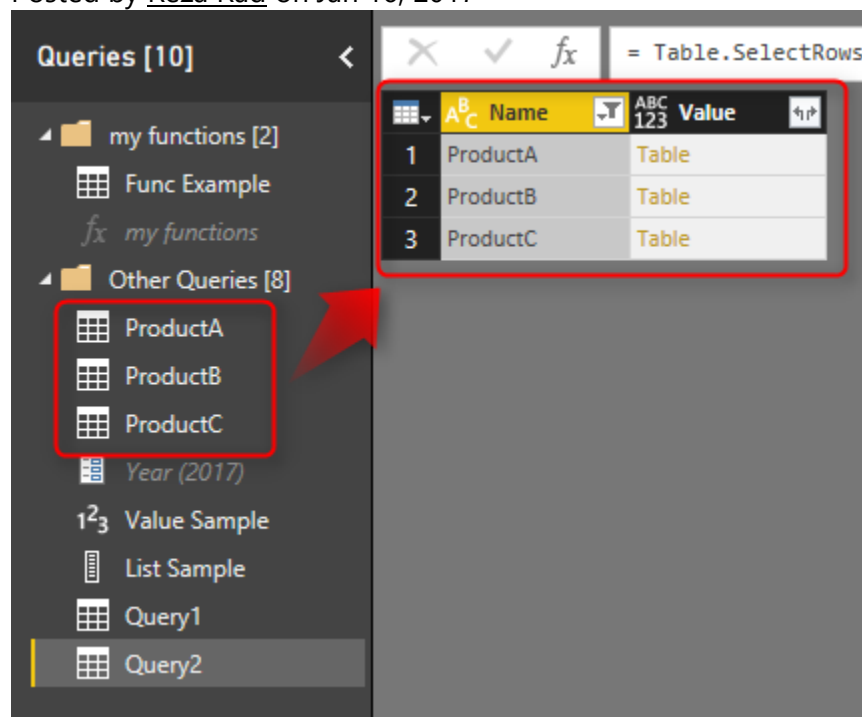
*1 DateTimedValue=#datetime(1970,1,1,0,0,0)+#duration(0,0,0,1415697165)*

the above query will respond 11/11/2014 9:12:45 AM.

Please note that you have to replace the 1415697165 number with the field name containing timestamp values, or with your static timestamp value in the query.

# Get List of Queries in Power BI

Posted by [Reza Rad](#) on Jan 16, 2017



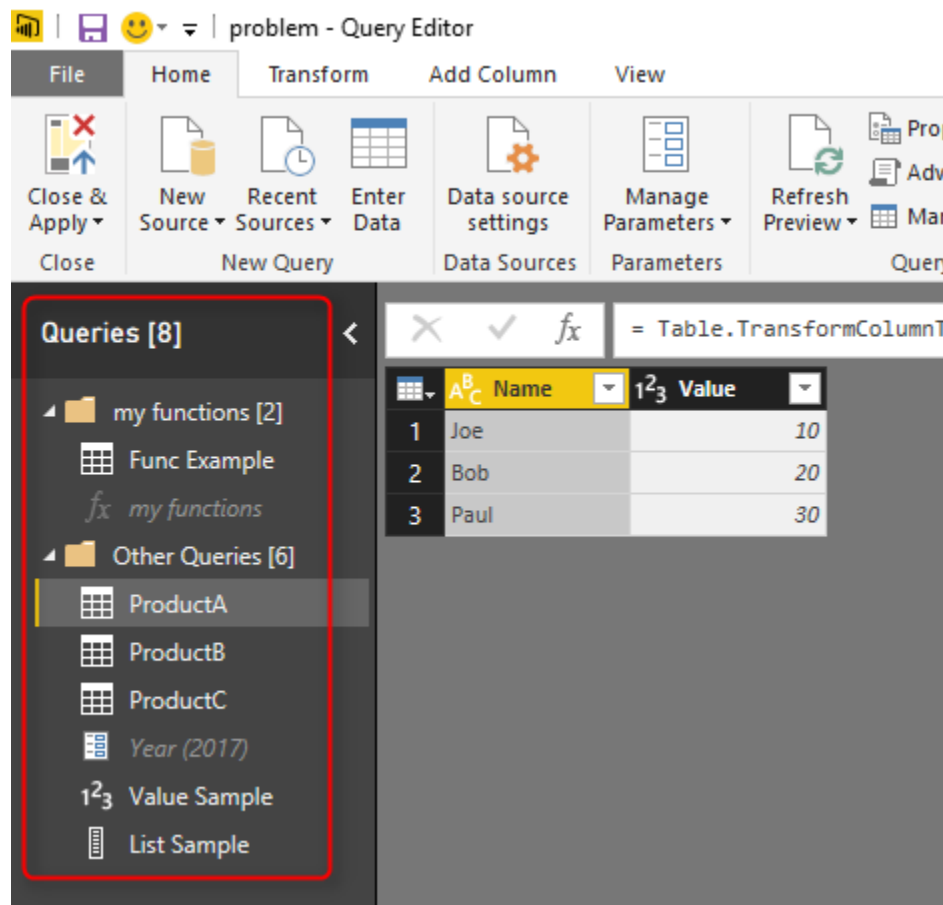
Power Query is the component of getting data in Power BI. But have you used Power Query to get metadata of the Power BI queries itself? In this post, I'll show you a quick and simple way of using Power Query to get metadata (name of queries and the data in queries) from all queries in Power BI. I have previously explained how to use a #shared keyword to get a list of all functions in Power Query; this post shows how to use #shared or #sections to get all queries (and parameters, and functions, and lists... ) from Power BI. If you want to learn more about Power BI; read [Power BI online book from Rookie to Rock Star](#).

\* Thanks to [Alex Arvidsson](#) who brought this question, and was the cause of writing this blog post.

## Question: How to Fetch Name of All Queries into One Query?

Consider below Power BI file that has functions, parameters, and queries.

Queries also return tables, values, and lists;



The question is how I can get a list of all these queries and their values as a new query? Let's see the answer

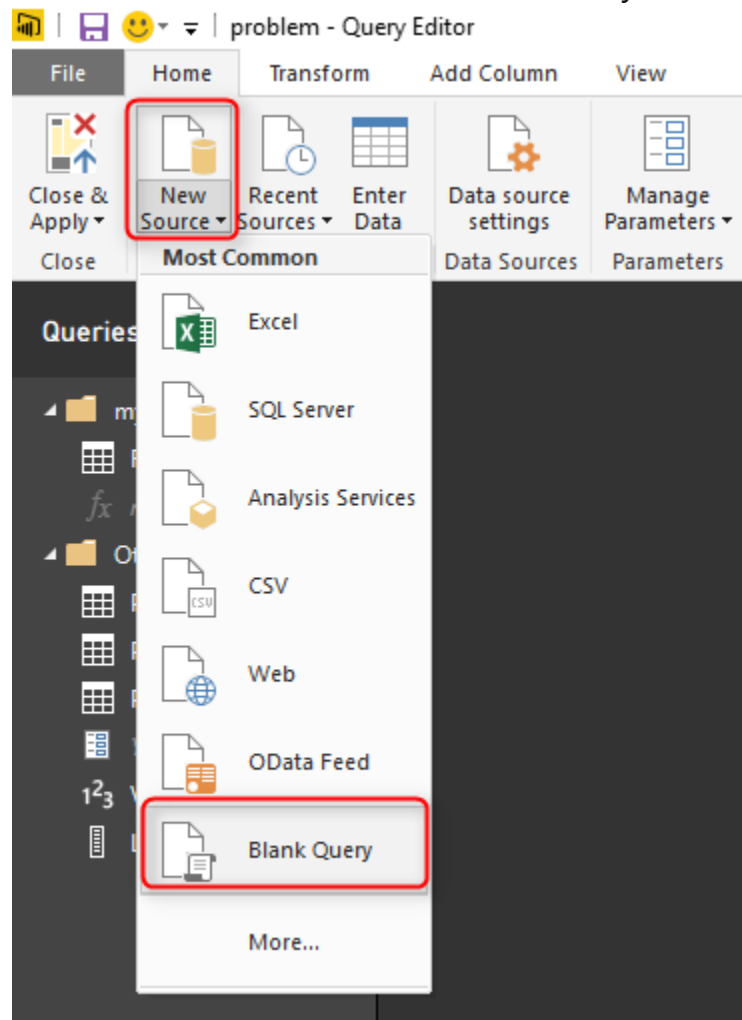
## Answer: Using #shared or #sections Keywords

### #shared to Get Current File's Queries Plus All Functions

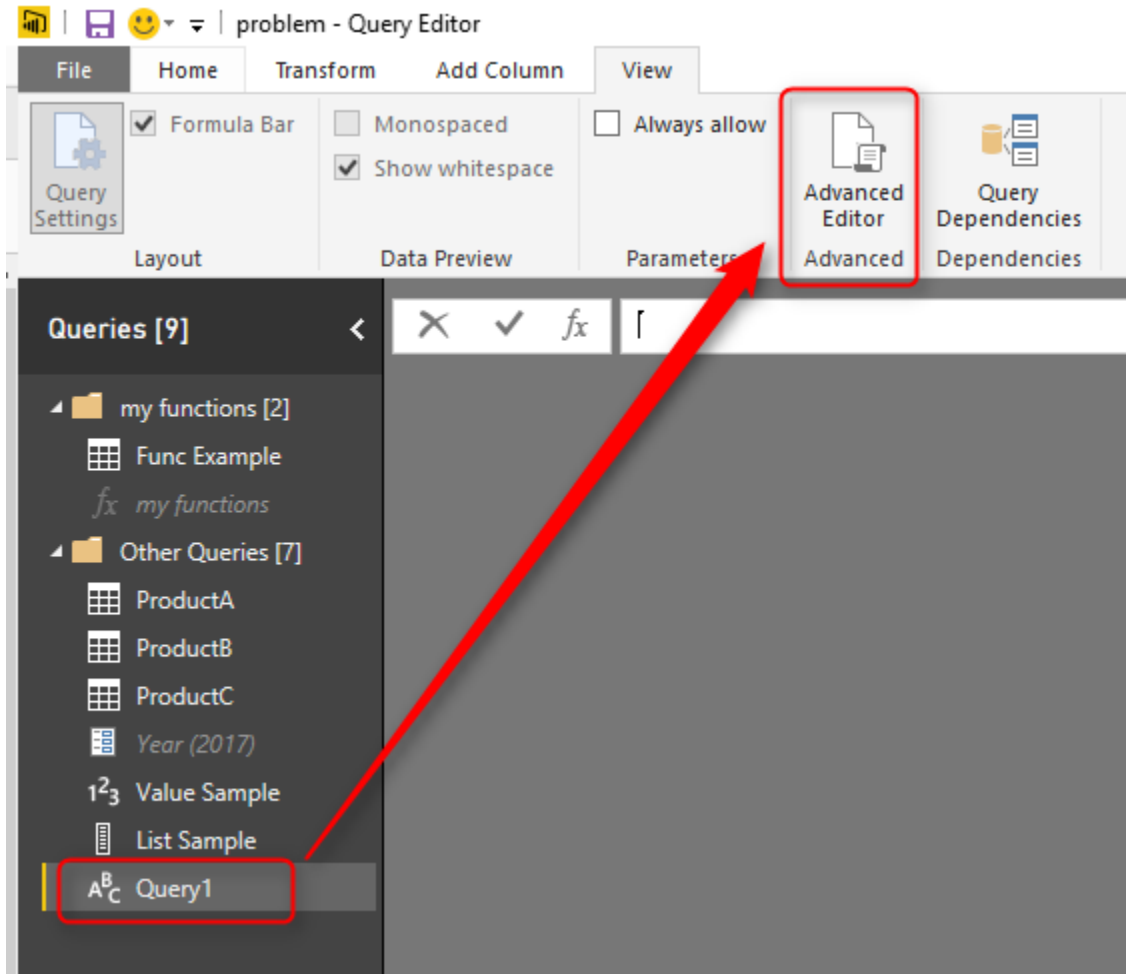
I have previously explained [what #shared keyword does](#); it is a keyword that returns a list of all functions, and enumerators in Power Query. It can be used as a document library in the Power Query itself. It will also fetch all queries in the existing Power BI file. Here is how you can use it:



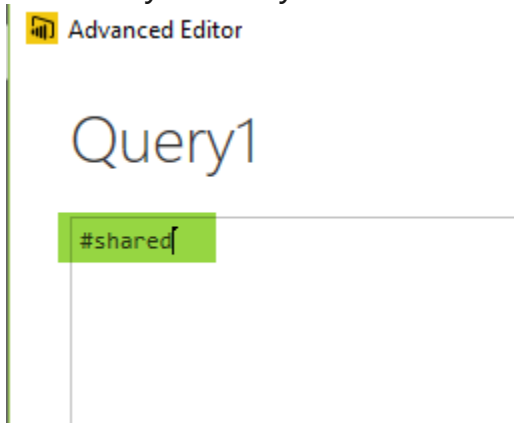
## Create a New Source, from Blank Query.



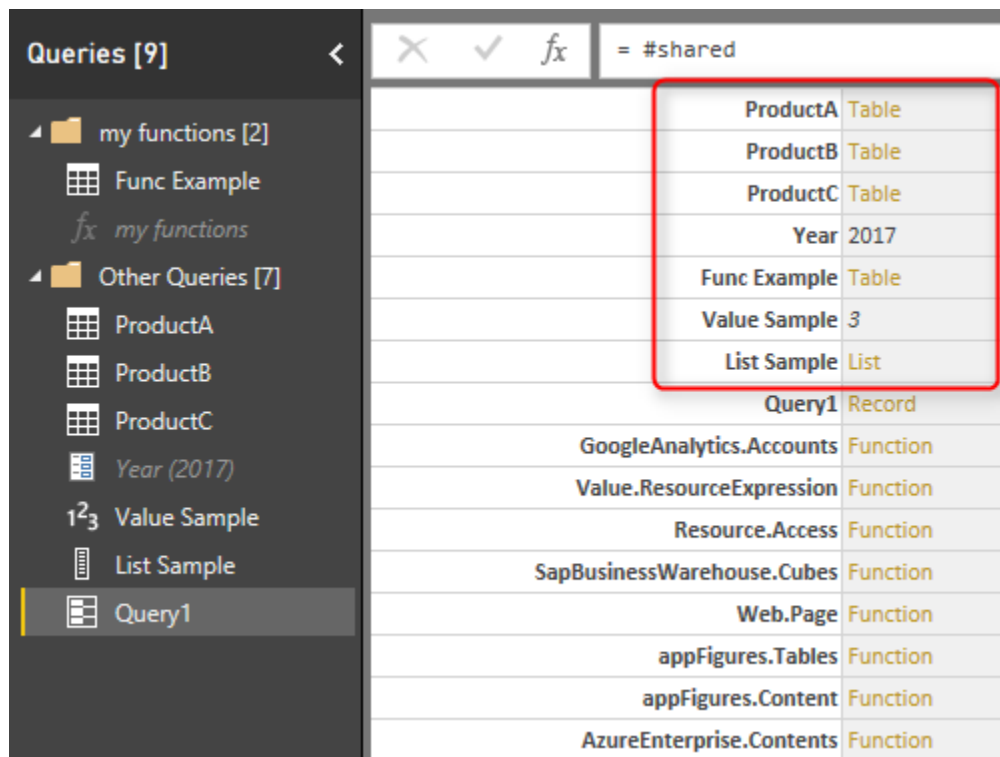
Then go to Advanced Editor of that query (from Home tab or View tab)



write only one keyword in the Advanced Editor: **#shared**



And Click on Done. The result will come up quickly;



Queries [9]	
my functions [2]	
Func Example	
my functions	
Other Queries [7]	
ProductA	
ProductB	
ProductC	
Year (2017)	
Value Sample	
List Sample	
Query1	

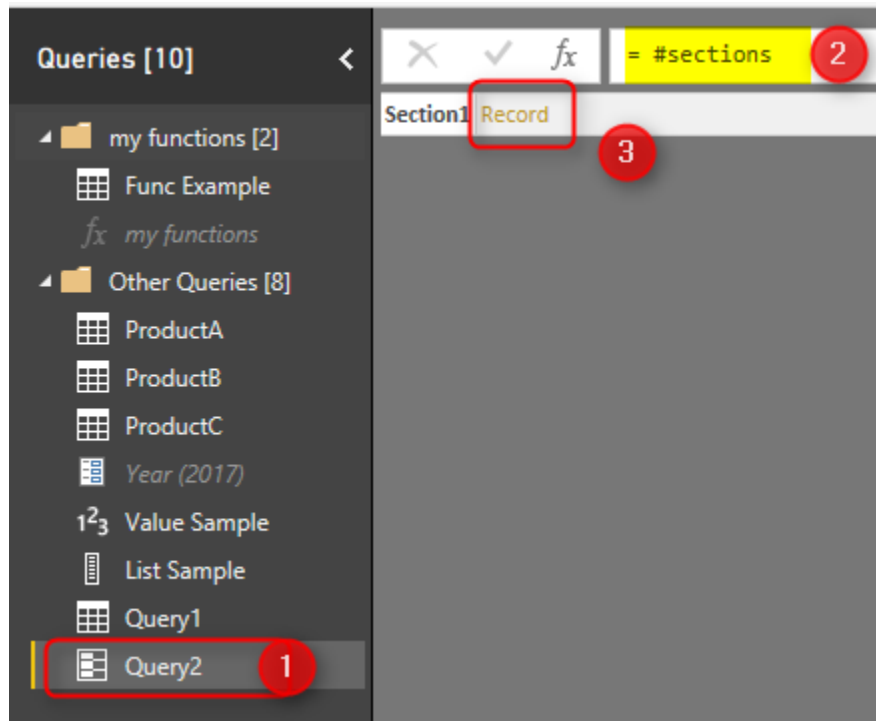
  

= #shared	
ProductA	Table
ProductB	Table
ProductC	Table
Year	2017
Func Example	Table
Value Sample	3
List Sample	List
Query1	Record
GoogleAnalytics.Accounts	Function
Value.ResourceExpression	Function
Resource.Access	Function
SapBusinessWarehouse.Cubes	Function
Web.Page	Function
appFigures.Tables	Function
appFigures.Content	Function
AzureEnterprise.Contents	Function

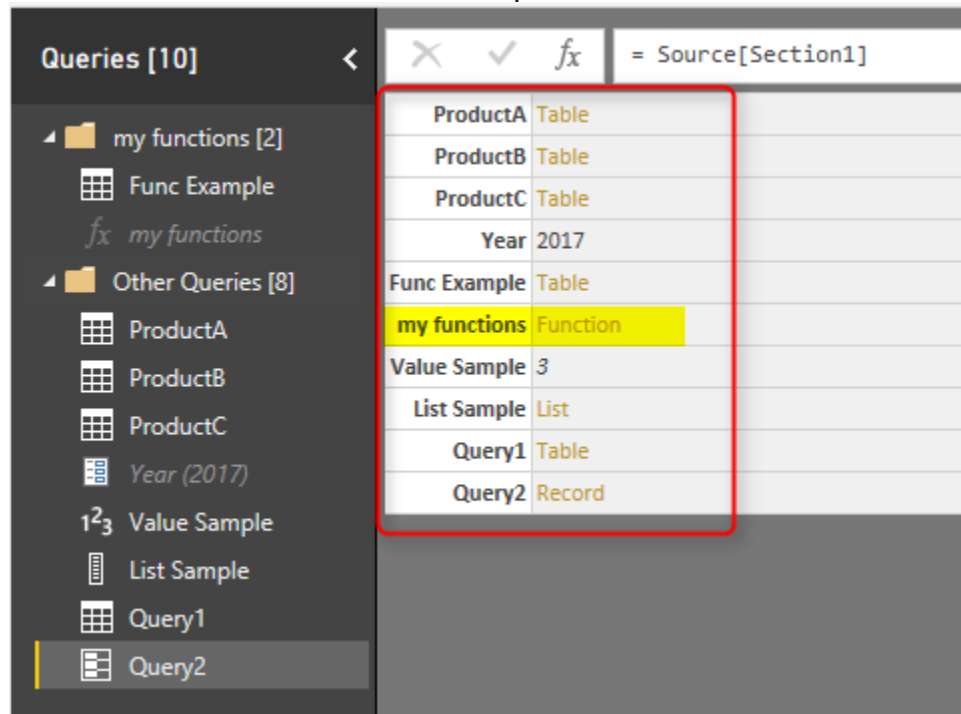
You can see in the result that `#shared` fetch all existing queries, plus all built-in functions in Power Query. The section which is marked in the above result set in where you can find queries from the current file. Note that the `Query1` itself (which includes the `#shared` keyword) is listed there. The limitation of this method is that it won't return your custom function; "my function" in this example. The next method, however, would pick that as well.

### #sections To Get Current File's Queries

The other way of a fetching list of queries is using the `#sections` keyword. The `#sections` keyword will give you a list of all sections in Power Query (This post is then't right place to explain what sections are, but for now, consider every query here as a section). so same method this time using `#sections` will return result below;

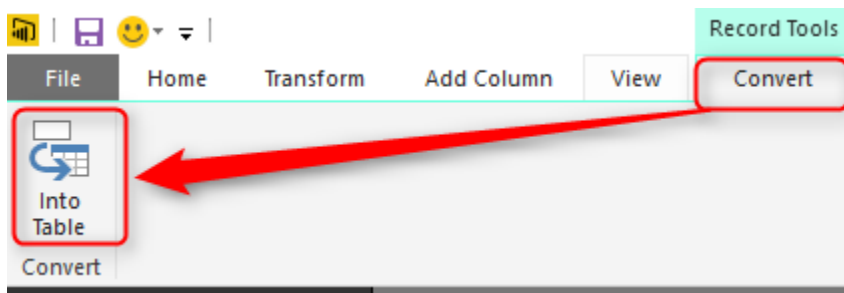


The result is a record that you can simply click on the Record to see what are columns in there. Columns are queries in the current file:

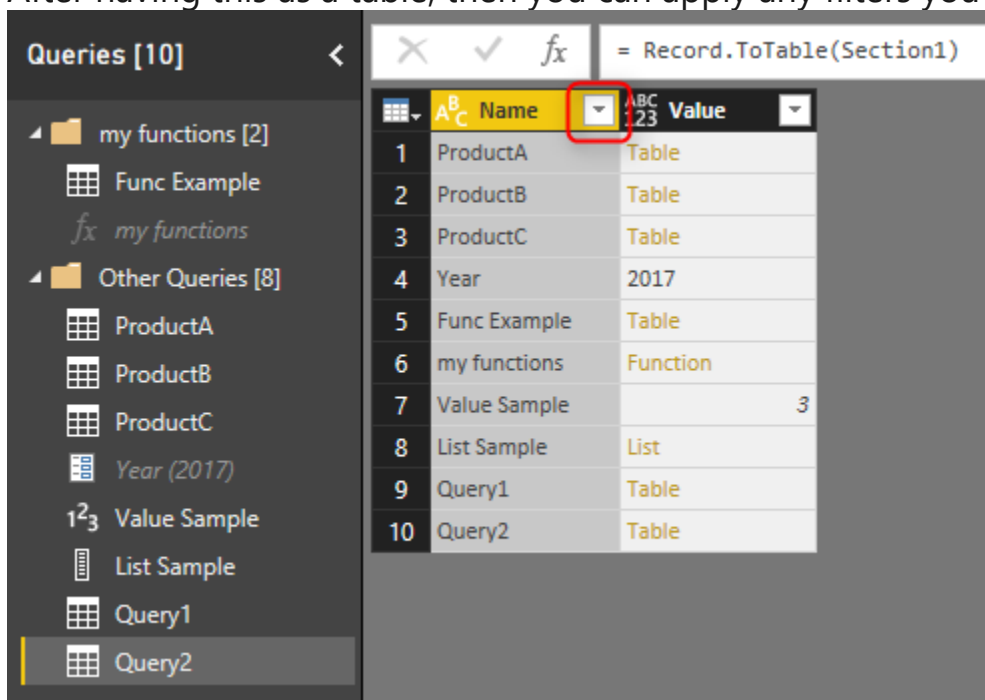


This method also returns functions in the current file, which the previous method with #shared didn't. So this is a better method if you are interested in fetching function's names as well.

The result is a record which can be converted to a table (this gives you better filtering options in the GUI). You can find the Convert Into Table under Record Tools Convert Section menu.



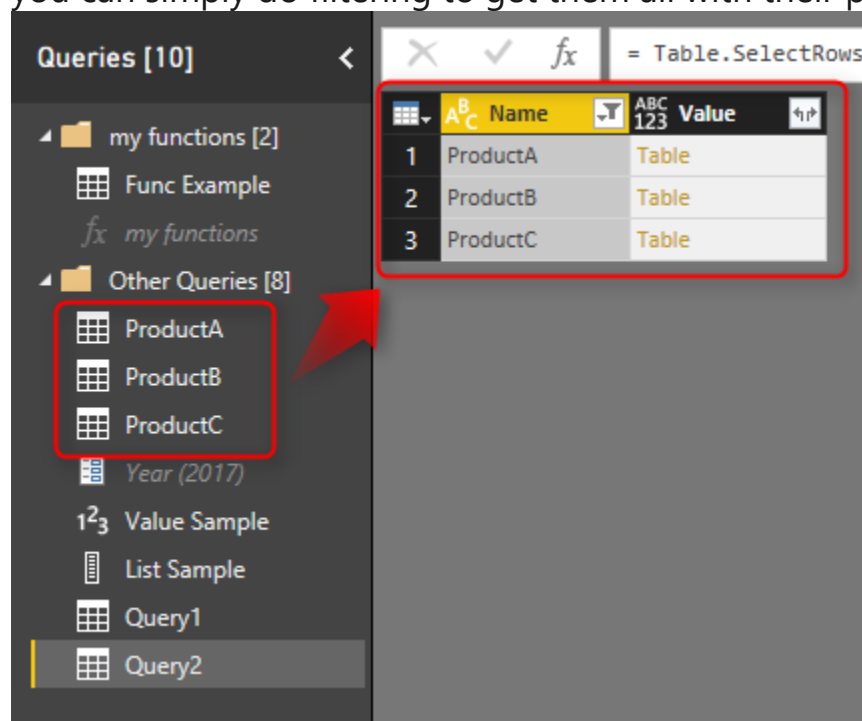
After having this as a table, then you can apply any filters you want;



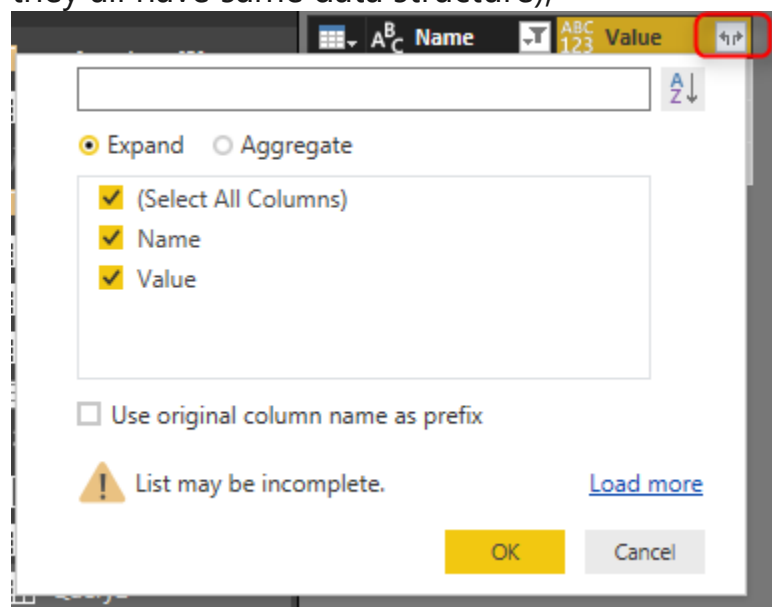
## Sample Scenario of Usage

There are many usages of getting the name of queries in another query. One sample usage of that can be getting different queries coming from different places, and the only way to identify the source is query name. In this case,

instead of manually adding a column to each query and then combining them, you can use this method to get the query name dynamically. In the screenshot below ProductA, ProductB, and ProductC are coming as source queries, and you can simply do filtering to get them all with their product names.



And you can expand it to tables underneath if you want to (this would work if they all have same data structure);

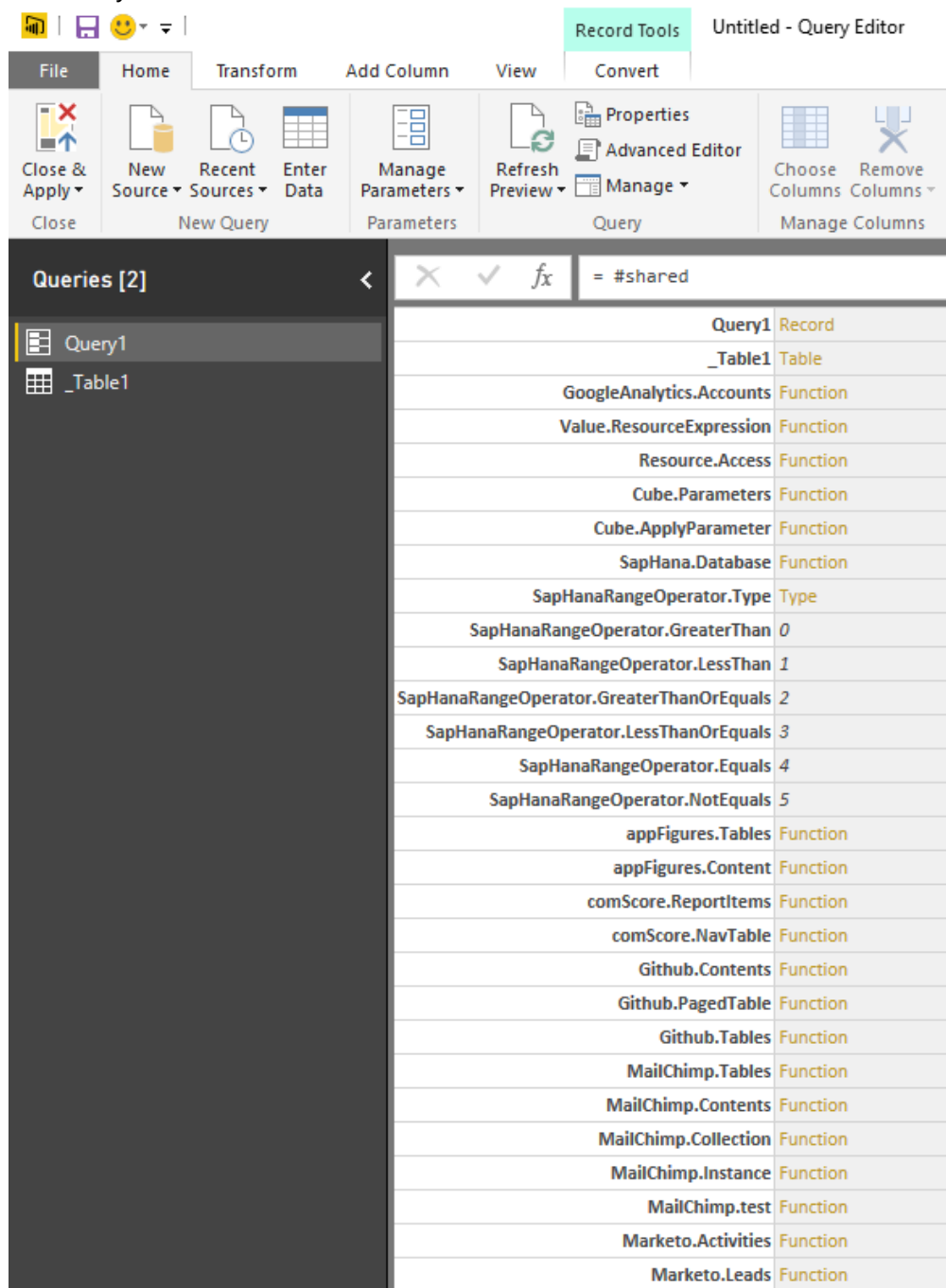


and as a result you have combined result of all queries, with the query name as another column;

	A <sup>B</sup> C Name	ABC 123 Name.1	ABC 123 Value.1
1	ProductA	Joe	10
2	ProductA	Bob	20
3	ProductA	Paul	30
4	ProductB	Joe	30
5	ProductB	Bob	60
6	ProductB	Paul	90
7	ProductC	Joe	1
8	ProductC	Bob	2
9	ProductC	Paul	3

# Power Query Library of Functions; Shared Keyword

Posted by [Reza Rad](#) on Jun 20, 2016



The screenshot shows the Power Query Editor interface. The 'Record Tools' tab is active, displaying the 'Library of Functions' pane on the left. The pane lists various functions and their categories. The main area shows the query results for the selected function, which is '= #shared'.

Function Name	Category
Query1	Record
_Table1	Table
GoogleAnalytics.Accounts	Function
Value.ResourceExpression	Function
Resource.Access	Function
Cube.Parameters	Function
Cube.ApplyParameter	Function
SapHana.Database	Function
SapHanaRangeOperator.Type	Type
SapHanaRangeOperator.GreaterThan	0
SapHanaRangeOperator.LessThan	1
SapHanaRangeOperator.GreaterThanOrEquals	2
SapHanaRangeOperator.LessThanOrEquals	3
SapHanaRangeOperator.Equals	4
SapHanaRangeOperator.NotEquals	5
appFigures.Tables	Function
appFigures.Content	Function
comScore.ReportItems	Function
comScore.NavTable	Function
Github.Contents	Function
Github.PagedTable	Function
Github.Tables	Function
MailChimp.Tables	Function
MailChimp.Contents	Function
MailChimp.Collection	Function
MailChimp.Instance	Function
MailChimp.test	Function
Marketo.Activities	Function
Marketo.Leads	Function

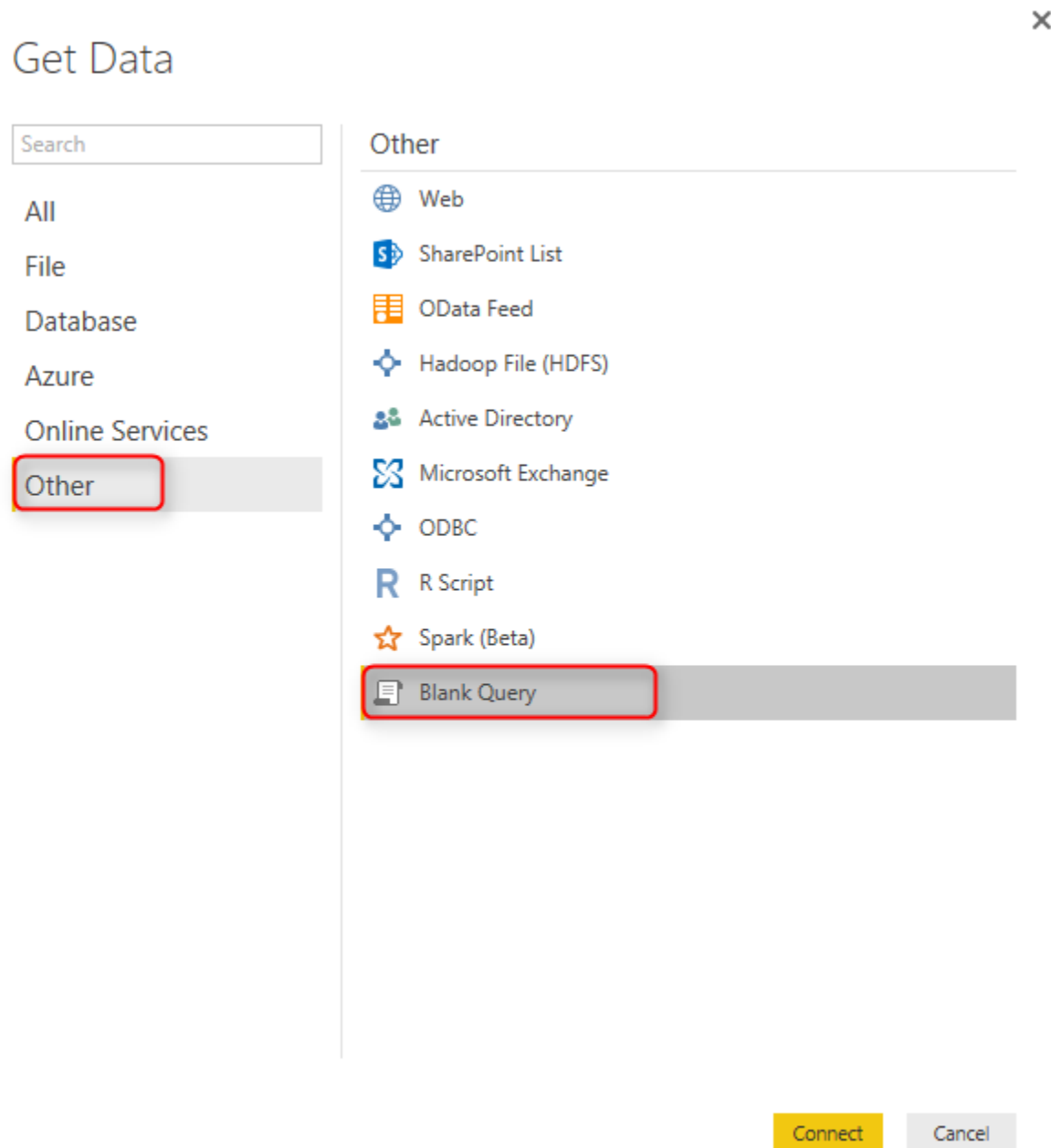
As I mentioned earlier in [Power BI online book](#), Power Query is a functional language. Knowing functions is your best helper when you work with a



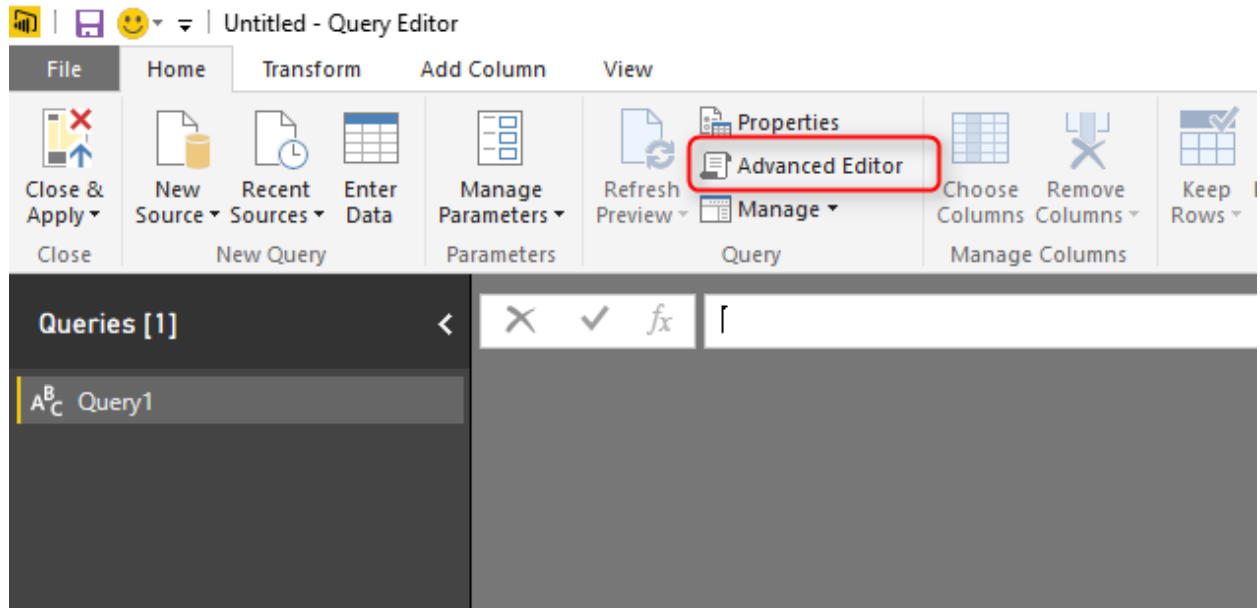
functional language. Fortunately, Power Query both in Excel and Power BI can use the shared keyword to reveal a document library of all functions. [I wrote about shared keyword](#) almost 2.5 years ago when it was only an add-in for excel. However, I still see people in my webinars who are new with #shared keyword functionality and amazed how helpful this little keyword is. So I decided to explain it with the new Power BI. With the method in this post, you can find any function you want easily in Power Query, and you won't need an internet connection to search in functions.

## **#shared Keyword**

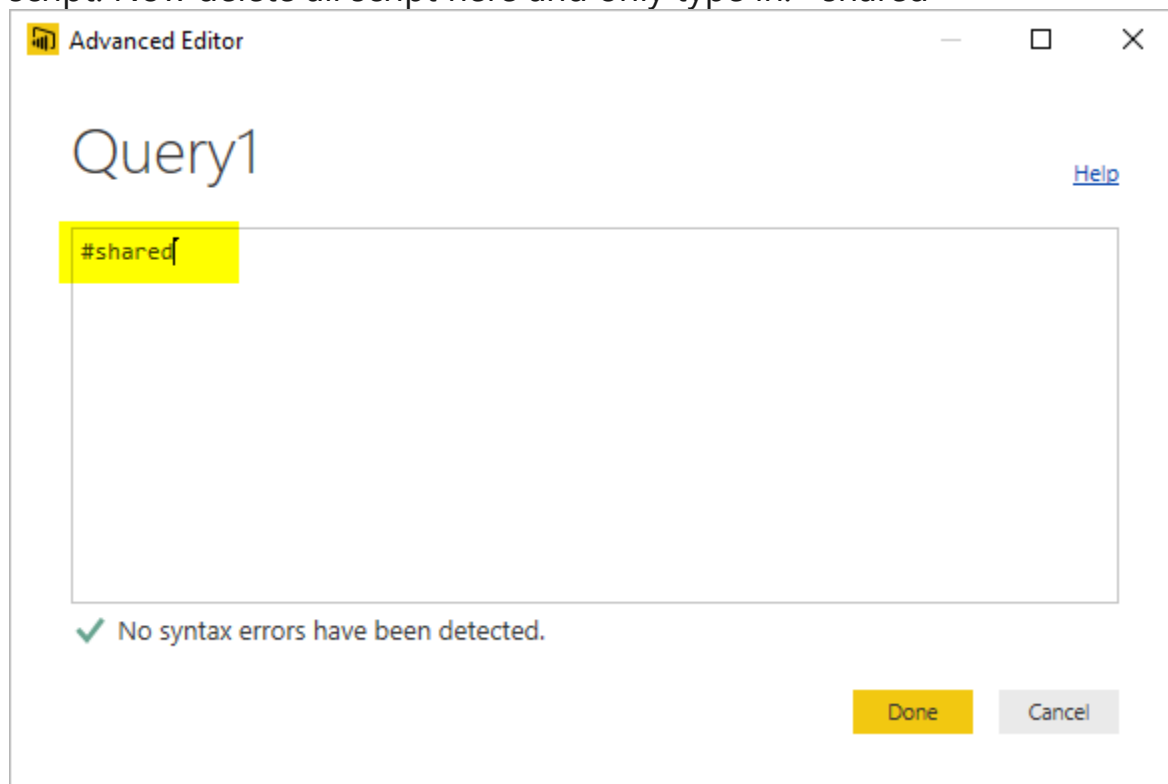
Shared is a keyword that loads all functions, and enumerators in the result set. You can simply create a blank query in Power BI (or Excel)



After opening a blank query, go to Advanced Editor



Here in the script editor is where you usually write or modify power query script. Now delete all script here and only type in: #shared



After clicking on Done, you will see a list of all functions and enumerators in power query. You will also see other queries in your Power BI solution or workbook plus other custom functions.

Record Tools    Untitled - Query Editor

File    Home    Transform    Add Column    View    Convert

Close & Apply    New Source    Recent Sources    Enter Data    Manage Parameters    Refresh Preview    Properties    Advanced Editor    Manage    Choose Columns    Remove Columns    Manage Columns

Close    New Query    Parameters    Query

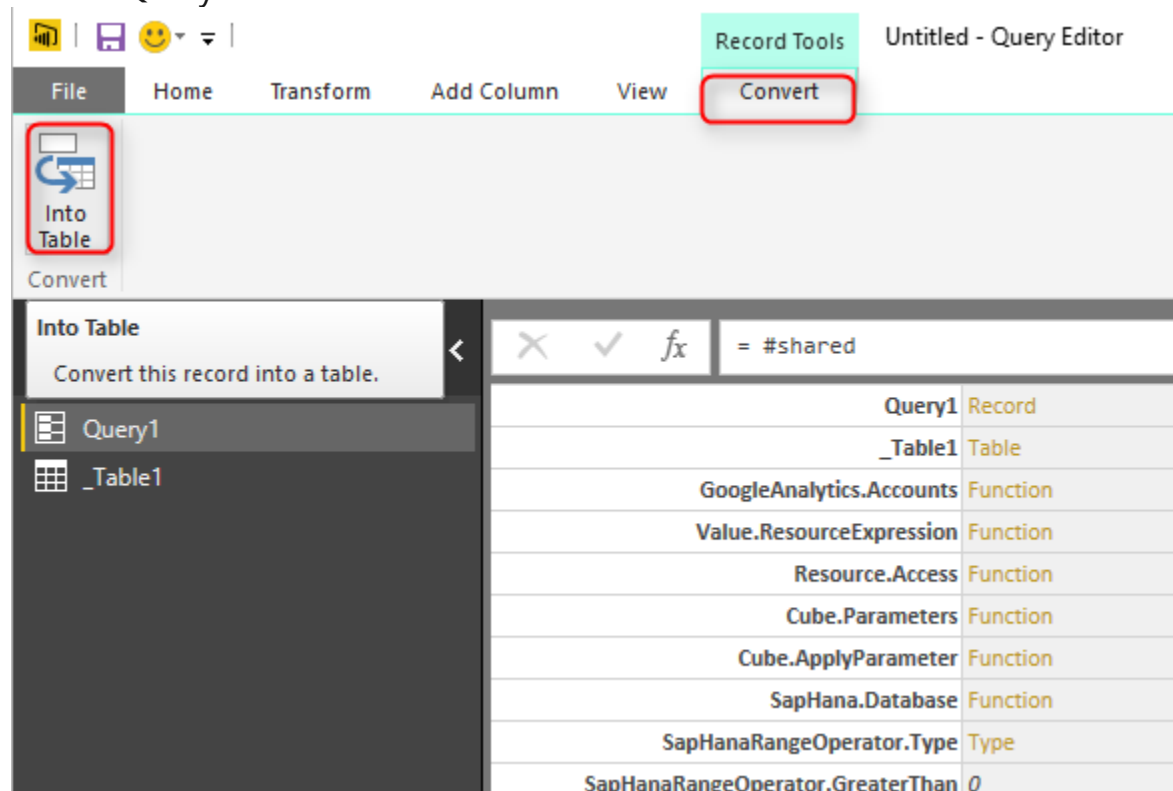
Queries [2]    <    X    ✓    fx    = #shared

Query1	Record
_Table1	Table
GoogleAnalytics.Accounts	Function
Value.ResourceExpression	Function
Resource.Access	Function
Cube.Parameters	Function
Cube.ApplyParameter	Function
SapHana.Database	Function
SapHanaRangeOperator.Type	Type
SapHanaRangeOperator.GreaterThan	0
SapHanaRangeOperator.LessThan	1
SapHanaRangeOperator.GreaterThanOrEquals	2
SapHanaRangeOperator.LessThanOrEquals	3
SapHanaRangeOperator.Equals	4
SapHanaRangeOperator.NotEquals	5
appFigures.Tables	Function
appFigures.Content	Function
comScore.ReportItems	Function
comScore.NavTable	Function
Github.Contents	Function
Github.PagedTable	Function
Github.Tables	Function
MailChimp.Tables	Function
MailChimp.Contents	Function
MailChimp.Collection	Function
MailChimp.Instance	Function
MailChimp.test	Function
Marketo.Activities	Function
Marketo.Leads	Function

Result set here is a Record structure that has functions, enumerators, and queries in each item of the record. Now let's explore the record more in details.

## Use the Result set as a Table

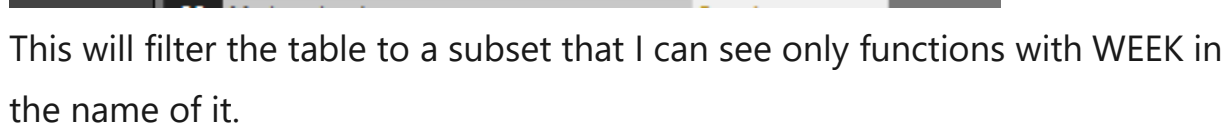
The result above is loaded in Power Query, and that is the greatest feature of Power Query itself that can turn this result into a table.



This will convert the result set record to a table. And the table is really easy to search in as you know.

✕ ✓ <i>f<sub>x</sub></i> = Record.ToTable(Query1)		
	ABC Name	ABC 123 Value
1	Query1	Table
2	_Table1	Table
3	GoogleAnalytics.Accounts	Function
4	Value.ResourceExpression	Function
5	Resource.Access	Function
6	Cube.Parameters	Function
7	Cube.ApplyParameter	Function
8	SapHana.Database	Function
9	SapHanaRangeOperator.Type	Type
10	SapHanaRangeOperator.GreaterThan	0
11	SapHanaRangeOperator.LessThan	1
12	SapHanaRangeOperator.GreaterThanOrEquals	2
13	SapHanaRangeOperator.LessThanOrEquals	3
14	SapHanaRangeOperator.Equals	4
15	SapHanaRangeOperator.NotEquals	5
16	appFigures.Tables	Function
17	appFigures.Content	Function
18	comScore.ReportItems	Function
19	comScore.NavTable	Function
20	Github.Contents	Function
21	Github.PagedTable	Function
22	Github.Tables	Function
23	MailChimp.Tables	Function
24	MailChimp.Contents	Function
25	MailChimp.Collection	Function
26	MailChimp.Instance	Function
27	MailChimp.test	Function
28	Marketo.Activities	Function
29	Marketo.Leads	Function
30	Marketo.Tables	Function

I can now simply search in the function list. Let's see for example what function I can find for working with WEEK;



fx = Table.SelectRows("#Converted to Table", each ([Name] = "Date.AddWeeks" or [Name] = "Date.DayOfWeek" or [Name]

	A <sup>B</sup> C Name	ABC 123 Value
1	Date.IsInPreviousWeek	Function
2	Date.IsInPreviousNWeeks	Function
3	Date.IsInCurrentWeek	Function
4	Date.IsInNextWeek	Function
5	Date.IsInNextNWeeks	Function
6	Date.AddWeeks	Function
7	Date.StartOfWeek	Function
8	Date.EndOfWeek	Function
9	Date.DayOfWeek	Function
10	Date.WeekOfMonth	Function
11	Date.WeekOfYear	Function

**function (dateTime as any, numberOfWeeks as number) as any**

Returns the date, datetime, or datetimezone result from adding numberOfWeeks weeks to the datetime value dateTime.

dateTime: The date, datetime, or datetimezone value to which weeks are being added.  
 numberOfWeeks: The number of weeks to add.

Example: Add 2 weeks to the date, datetime, or datetimezone value representing the date 5/14/2011.

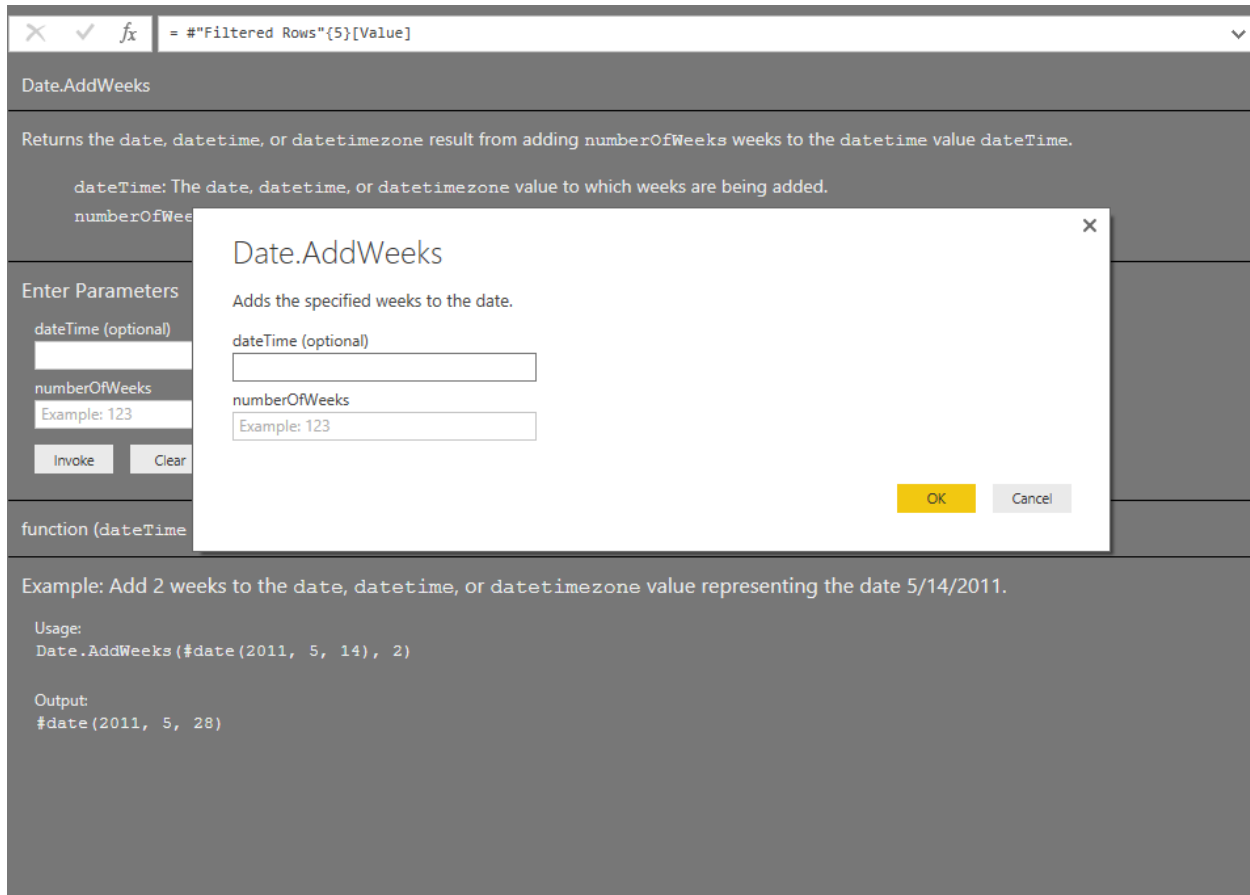
Usage:  
 Date.AddWeeks (#date (2011, 5, 14), 2)

Output:  
 #date (2011, 5, 28)

## Documentation of Function

Now for example if I want to see the Date.AddWeeks function I can click on the "function" link of it in the value column and this will redirect me to the documentation of this function, and will bring a dialog box to invoke the function!





The screenshot shows the Power BI interface with the formula bar containing `= #"Filtered Rows"{5}[Value]`. The `Date.AddWeeks` function is selected, and its documentation is displayed in a grey panel. The documentation includes the function's purpose, parameters (`dateTime` and `numberOfWeeks`), an example, and usage. A modal dialog titled `Date.AddWeeks` is open, allowing the user to enter parameters and click `OK` or `Cancel`.

**Date.AddWeeks**

Returns the date, datetime, or datetimezone result from adding `numberOfWeeks` weeks to the datetime value `dateTime`.

`dateTime`: The date, datetime, or datetimezone value to which weeks are being added.

`numberOfWeeks`

**Enter Parameters**

`dateTime` (optional)

`numberOfWeeks`

Example: 123

Invoke Clear

**function** (`dateTime`

**Example:** Add 2 weeks to the date, datetime, or datetimezone value representing the date 5/14/2011.

**Usage:**

`Date.AddWeeks(#date(2011, 5, 14), 2)`

**Output:**

`#date(2011, 5, 28)`

You can see the documentation in grey that also includes examples of how to call this function. For invoking the function, I can simply provide parameters, and click OK or Invoke;



The screenshot shows the `Date.AddWeeks` modal dialog with the following parameters entered:

**Date.AddWeeks**

Adds the specified weeks to the date.

`dateTime` (optional)

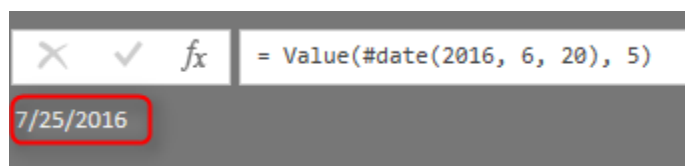
6/20/2016

`numberOfWeeks`

5

OK Cancel

And it calls the function and shows me the result as below;



## Enumerators

Finding enumerators is also easy with the help of #shared keyword. Here I can see enumerators for JoinKind and JoinAlgorithm;

	ABC	Name	Value
1		JoinKind.Type	Type
2		JoinKind.Inner	0
3		JoinKind.LeftOuter	1
4		JoinKind.RightOuter	2
5		JoinKind.FullOuter	3
6		JoinKind.LeftAnti	4
7		JoinKind.RightAnti	5
8		Table.Join	Function
9		Table.AddJoinColumn	Function
10		Table.NestedJoin	Function
11		JoinAlgorithm.Type	Type
12		JoinAlgorithm.Dynamic	0
13		JoinAlgorithm.PairwiseHash	1
14		JoinAlgorithm.SortMerge	2
15		JoinAlgorithm.LeftHash	3
16		JoinAlgorithm.RightHash	4
17		JoinAlgorithm.LeftIndex	5
18		JoinAlgorithm.RightIndex	6

## Summary

You've learned how #shared keyword can be helpful for getting the list of all functions and enumerators in Power Query. You learned you could convert the result set into a table and filter that to find the particular function you are looking for. This is superb, especially for people like myself who can't remember things well. You know how I look for functions? This post explained my method! I use the #shared keyword to find the function I want and start

working on that. For me, the #shared is the keyword that I use more than any other queries in Power query side of Power BI.

# Writing Custom Functions in Power Query M

Posted by [Reza Rad](#) on Feb 12, 2014



One of the most powerful features of M is that you can write custom functions to re-use part of your code. Custom functions can be single or multiple liners, they will be written in Lambda style syntax. In this post, you will learn how to write functions and invoke them in M.

Custom functions and some of the other features of M such as error handling, completely distinguish this language from expressions in SSIS. I'm saying this because I've heard questions like: Is the M similar to SSIS expressions? The answer is: NO! I am a big fan of SSIS (those of you who know me would agree that), But SSIS expressions are just for calculation and creating transformations based on expressions. SSIS expressions is not a language, it is an expression. M in the other hand is a language, and it is a functional language, that means you don't need to write Main function, etc, M would handle most of them. and you have features such as writing custom functions and error handling (You cannot find most of these advanced features through Power Query GUI, So start scripting M )

## Basic Syntax

Functions in M can be written in this format:

*(x) => x+1*

This is lambda syntax (that previously used in LINQ if you come from .NET development background). The line above is equal to pseudo-code below:

Function anonymous (x)

```
{  
  return x+1  
}
```

As you see lambda made it much simpler to define the function with just that single line. So the function above get a parameter from the input and adds 1 to it. Please note that datatype of the parameter not defined, so that means if a text is input, then an error would occur, so you would require to implement the error handling as well (I'll describe Error Handling in M in future blog posts).

So let's see how this function works and how we can call this function. Script below shows how to define the function and invoke it with a parameter:

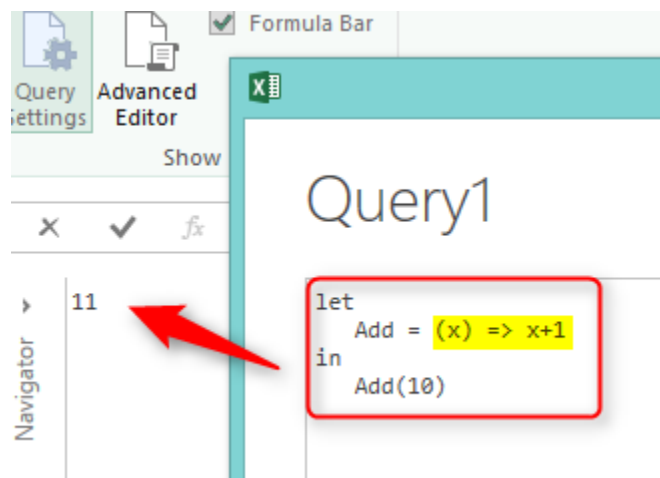
*let*

*Add = (x) => x+1*

*in*

*Add(10)*

The result of the above expression is: 11



In the example above we've named the function as "Add" and then we call that with Add(inputparam).

### Parameters

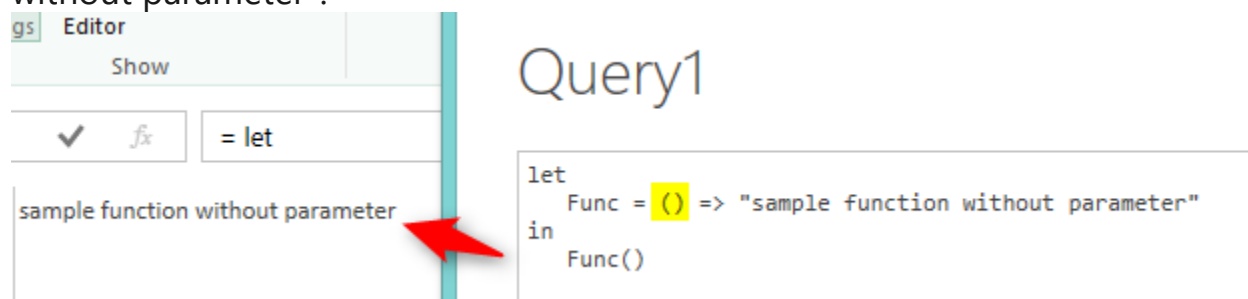
If you want to define a function with more parameters, then add parameters as below:

*$(x,y,z) \Rightarrow x+y+z$*

you can also define a function without parameters, as below:

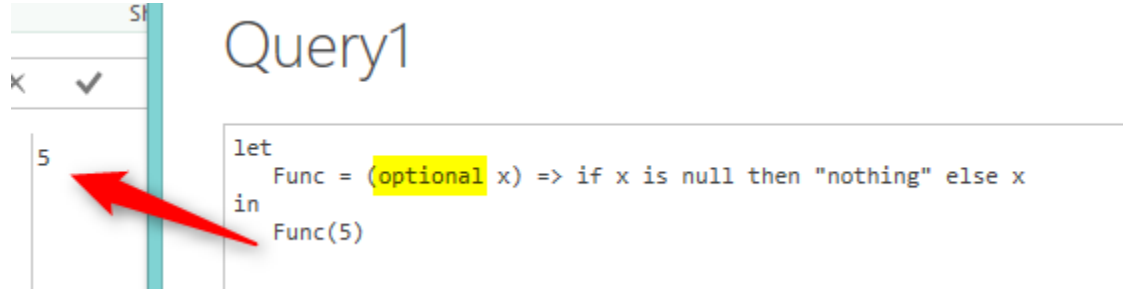
*$() \Rightarrow \text{"sample function without parameter"}$*

This function gets no parameters and returns the text "sample function without parameter".



### Optional Parameters

Parameters are defined as Required by default. That means you should specify the parameter at the time of invoking the function. So if you want to define an optional parameter, use the Optional keyword as below:

*let**Func = (optional x) => if x is null then "nothing" else x**in**Func(5)*

## Variables in Function

Samples above showed how to write single line expression function, but in most of the cases, you would require to write multiple line function that contains variables inside the function. In that case, you can define the function within the LET / IN structure as below:

*let**Func = (x) =>**let**<body>**in**<return value>**in**Func(5)*

As you see in the above script, another set of Let/In added inside the function. You can write the body of the function in the LET clause. And if you want to add multiple lines there, or if you want to define variables, do this as usual with a single comma at the end of each line. Finally, you can return value in the IN clause.

### Example

Following example show how we can use structure above to create a function that return number of days passed from the start of the year until that date.

*let*

*DayPassedInYear = (x) =>*

*let*

*MonthList=List.Numbers(1,Date.Month(DateTime.FromText(x))-1),*

*Year=Date.Year(DateTime.FromText(x)),*

*DaysInMonthList=List.Transform(MonthList,each*

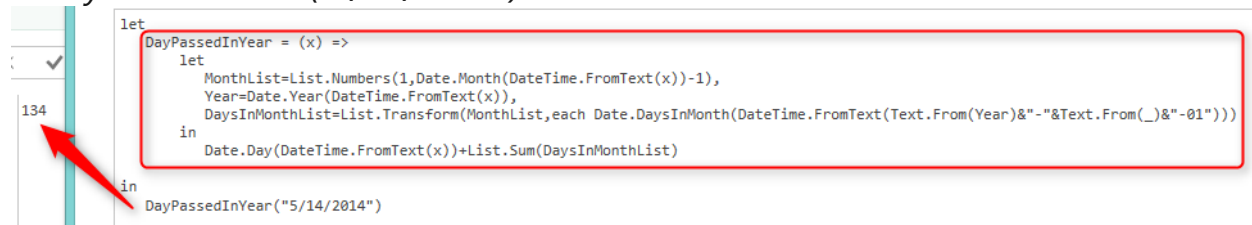
*Date.DaysInMonth(DateTime.FromText(Text.From(Year)&"-"&Text.From(\_)&"-01")))*

*in*

*Date.Day(DateTime.FromText(x))+List.Sum(DaysInMonthList)*

*in*

*DayPassedInYear("5/14/2014")*



### Line by Line Description

Let's go through the script line by line;

*DayPassedInYear = (x) =>*



In above line we define the function name as "DayPassedInYear", this function require a single parameter, definition of the body would come in next lines.

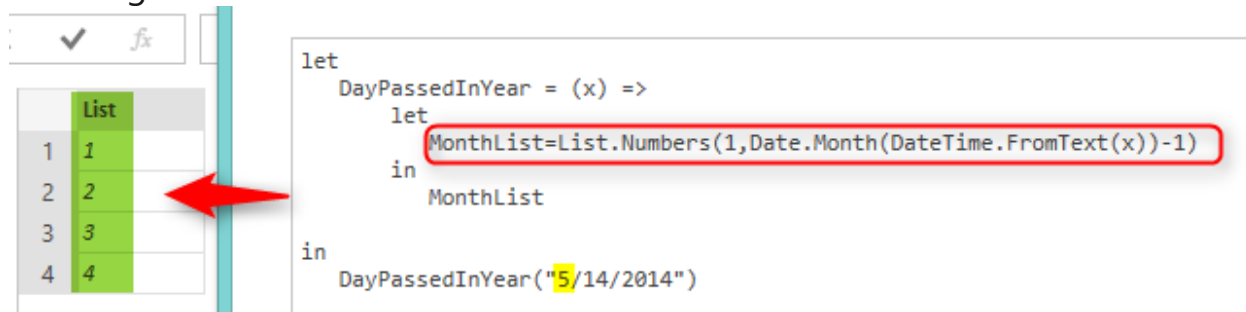
*let*

*MonthList=List.Numbers(1,Date.Month(DateTime.FromText(x))-1),*

We used Let to define a multi-line body. The first line of the body is defining a variable that creates a list of Month numbers from the first month (1) to the previous month of the input date. For calculating the previous month of the input date, we used this expression: *Date.Month(DateTime.FromText(x))-1* .

And this part generates a list based on numbers from a beginning number (1) to ending number (which would be the result of previous month function): *List.Numbers(1,...)*

So as a result, the MonthList would be a variable that contains a list of months from the first month up to a prior month in the current year. Let's see how that single line would return the result:



	List
1	1
2	2
3	3
4	4

```

let
    DayPassedInYear = (x) =>
        let
            MonthList=List.Numbers(1,Date.Month(DateTime.FromText(x))-1)
        in
            MonthList
in
    DayPassedInYear("5/14/2014")

```

Next line in the script calculates year of the input date and store that into a variable named "Year":

*Year=Date.Year(DateTime.FromText(x)),*

Consider that you should end each line with a single comma (if you don't want to logic of two lines to be parsed together).

The final line of the body combined from multiple expressions, I'll describe the one by one;

*DateTime.FromText(Text.From(Year)&"-"&Text.From(\_)&"-01")*

Above expression will return the first day of a month. Please note that there is a single underscore in the expression. That underscore would be used in EACH expression. EACH is a single parameter function. We used EACH in this example to apply a transformation to every item in the list. Actually we want to replace each month number in the list, with the number of days in that month. So we use EACH single liner function to fetch a number of days in that month. When you use EACH, you can use underscore as the parameter marker. In simpler words, if you have a list as below:

ListA

1  
2  
3

And if you transform that list with List.Transform function as below:

*List.Transform(ListA,each \_\*10)*

The result would be:

ListA

10  
20  
30

So the result of This line below:

*DaysInMonthList=List.Transform(MonthList,each  
Date.DaysInMonth(DateTime.FromText(Text.From(Year)&"-"&Text.From(\_)&"-  
01"))))*

Would be:



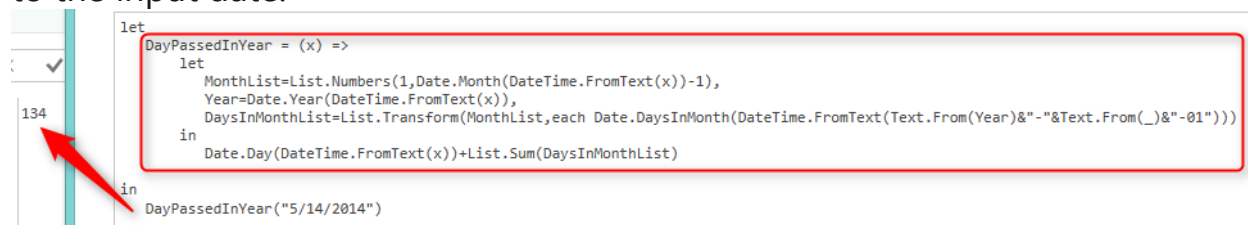
As you see each list item which had the month number earlier, now replaced (transformed) with the number of days in that month.

In the final step of the function we return the result;

*in*

*Date.Day(DateTime.FromText(x))+List.Sum(DaysInMonthList)*

IN clause used to return the result, and we use Date.Day function to return the day part of input date. And we add that with the sum of all values from the list. As a result, we would see the number of days passed from the first of the year to the input date.



## Summary

In this blog post, you've learned how to define functions and how to invoke them. You've learned that you can define optional or required parameters. You can also define multi-line functions that contain variables inside the body. You've seen a sample function that shows how useful are functions in the real

world. You've also learned about EACH keyword which can be used as a single parameter function (especially in lists and tables).

# Day Number of Year, Power Query Custom Function

Posted by [Reza Rad](#) on Jul 28, 2015

```

let
    DayNumberOfYear = (date) =>
        let
            dated = try DateTime.FromText(date),
            month = Date.Month(dated[Value]),
            MonthList = List.Numbers(1, month - 1),
            year = Date.Year(dated[Value]),
            TransformedMonthList = List.Transform(
                MonthList,
                each Text.From(year) & "-" & Text.From(_) & "-1"),
            DateList = List.Transform(
                TransformedMonthList,
                each DateTime.FromText(_)),
            DaysList = List.Transform(
                DateList,
                each Date.DaysInMonth(_))
        in
            if dated[HasError]
            then dated[Error]
            else List.Sum(DaysList) + Date.Day(dated[Value])
in
    DayNumberOfYear("07/28/2015")

```

There are some Date and DateTime built-in functions in Power Query which are helpful. There is also a function for DayNumberOfYear. However, I've thought it would be a good example to go through writing a function that uses Generators, Each singleton function, and error handling all inside a custom function. Through this post you will also learn;

- how to create Custom Function
- how to use Generators as a loop structure
- and how to user Error Handling.

Let's consider this date as today's date: 28th of July of 2015 (this is the date of this blog post)

There might be a number of methods to calculate the day number of year for this date (which is 209). I use one of them here. Steps are as below;

- fetch a number of days for each month from January of this year (of the date above) to the previous month (of the date above).

- Calculate the sum of values above will give me the number of days in all month prior than this month.
- Add the day number of the date to the calculated sum above.

Some of the calculations can be helped through with Power Query Date functions. So let's start;

1 – Create a function in Power Query called DayNumberOfYear as below

*If you don't know where to write below code:*

- *Open Excel, Go to Power Query Tab, Click On Get Data from Source, Blank Query, In the Query Editor window go to View tab, and click on Advanced Editor.*

*Open Power BI, click on Edit Queries, In the Query Editor window go to View tab, and click on Advanced Editor.*

```

1 let                                     //start of the code
2   DayNumberOfYear= (date) =>           //Function name, and input
3   parameter
4       let                               //start of function body
5           dated=DateTime.FromText(date) //date conversion from text
6       in                                 //start of function output
7           dated                         //function body output
8 in                                     //start of output lines generated
   DayNumberOfYear("07/28/2015")        //call function by a value

```

I've put some comments in above script to help you understand each line. In general, DayNumberOfYear is the name of the function. It accepts an input parameter "date". and convert the parameter from text value to DateTime. The last line of the code calls the function with a specific date ("07/28/2015").

**\*\* Note that Date Conversion function is locale dependent. So if the date time of your system is no MM/DD/YYYY then you have to enter date as it formatted in your system (look below the clock on right-hand side bottom of your monitor to check the format).**

The result of the above script will be:

**07/28/2015 12:00:00 a.m.**

## 2 – Fetch Month Number and generate a list of all prior months.

Fetching month number is easily possible with `Date.Month` function. The remaining part is looping through months from January of this year to the previous month (of the given date). Unfortunately there is no loop structure in Power Query M language yet, but fortunately, we can use Generator functions for that. A generator function is a function that produces/generate a list based on some parameters. For example, you can generate a list of dates from a start date, based on the given occurrence of a period. Or you can generate a list of numbers. For this example, we want to generate a list of numbers, starting from 1 (month January) to the current month number minus 1 (previous month).

Here is the code:

```

1  let
2      DayNumberOfYear= (date) =>
3          let
4              dated=DateTime.FromText(date),
5              month=Date.Month(dated),//month number
6              MonthList=List.Numbers(1,month-1) // generate list of months
7  from Jan to previous month
8      in
9          MonthList
10 in
    DayNumberOfYear("07/28/2015")

```

The result is a list of month numbers as below:

	List
1	1
2	2
3	3
4	4
5	5
6	6

Generator Function used in the above code is List.Numbers. This function generate a list of numbers starting from a value.

### 3 – Transform list to full date list

We have to calculate a number of days for each month in the function. A number of days in each month can be fetched by Date.DaysInMonth function. However this function accepts a full DateTime data type, and the value that we have in our list members are text. So we have to produce a DateTime value from it. For generating a full date we need the year portion as well, we use Date.Year function to fetch that.

Here is the code to transform the list:

```

1  let
2      DayNumberOfYear= (date) =>
3          let
4              dated=DateTime.FromText(date),
5              month=Date.Month(dated),
6              MonthList=List.Numbers(1,month-1),
7              year=Date.Year(dated), // fetch year
8              TransformedMonthList=List.Transform // transform list
9                  (MonthList,
10                     each Text.From(year)&"-"&Text.From(_)&"-1") // generate
11 full text value
12         in
13             TransformedMonthList
14 in
    DayNumberOfYear("07/28/2015")

```

The result is:



	List
1	2015-1-1
2	2015-2-1
3	2015-3-1
4	2015-4-1
5	2015-5-1
6	2015-6-1

As you see the above result is not still of DateTime data type, we've only generated full date as a text value. Now we can convert values to DateTime data type

Here is the code:

```

1 let
2     DayNumberOfYear= (date) =>
3         let
4             dated=DateTime.FromText(date),
5             month=Date.Month(dated),
6             MonthList=List.Numbers(1,month-1),
7             year=Date.Year(dated),
8             TransformedMonthList=List.Transform
9                 (MonthList,
10                  each Text.From(year)&"-"&Text.From(_)&"-1"),
11             DateList=List.Transform(
12                 TransformedMonthList,
13                 each DateTime.FromText(_))//transform to DateTime value
14         in
15             DateList
16 in
17     DayNumberOfYear("07/28/2015")

```

and the result:

	List
1	1/1/2015 12:00:00 AM
2	2/1/2015 12:00:00 AM
3	3/1/2015 12:00:00 AM
4	4/1/2015 12:00:00 AM
5	5/1/2015 12:00:00 AM
6	6/1/2015 12:00:00 AM

#### 4 – Transform the list to List of DaysNumberOfMonths

We use DaysInMonth function to fetch the number of days in each month from the list. Here is the code:

```

1 let
2     DayNumberOfYear= (date) =>
3         let
4             dated=DateTime.FromText(date),
5             month=Date.Month(dated),
6             MonthList=List.Numbers(1,month-1),
7             year=Date.Year(dated),
8             TransformedMonthList=List.Transform
9                 (MonthList,
10                  each Text.From(year)&"-"&Text.From(_)&"-1"),
11             DateList=List.Transform(
12                 TransformedMonthList,
13                 each DateTime.FromText(_)),
14             DaysList=List.Transform(
15                 DateList,
16                 each Date.DaysInMonth(_))
17         in
18             DaysList
19 in
20     DayNumberOfYear("07/28/2015")

```

and the result:

	List
1	31
2	28
3	31
4	30
5	31
6	30

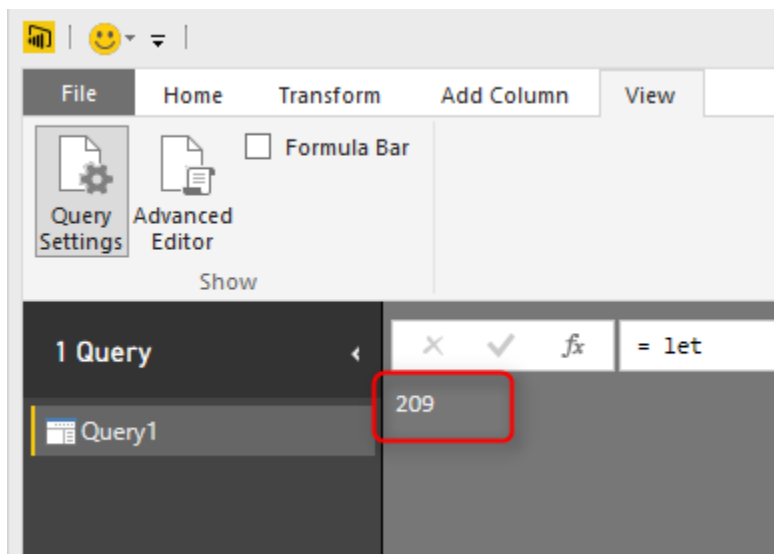
5 – Calculate the Sum of Dates and Add day number of this month to it  
 The list is ready to use; we need to sum it up only. And then add the current day Date.Day from the given date to it.  
 Here is the code:

```

1 let
2     DayNumberOfYear= (date) =>
3         let
4             dated=DateTime.FromText(date),
5             month=Date.Month(dated),
6             MonthList=List.Numbers(1,month-1),
7             year=Date.Year(dated),
8             TransformedMonthList=List.Transform
9                 (MonthList,
10                  each Text.From(year)&"-"&Text.From(_)&"-1"),
11             DateList=List.Transform(
12                 TransformedMonthList,
13                 each DateTime.FromText(_)),
14             DaysList=List.Transform(
15                 DateList,
16                 each Date.DaysInMonth(_))
17         in
18             List.Sum(DaysList)//sum of values in the list
19             +Date.Day(dated)//current date's day number
20 in
21     DayNumberOfYear("07/28/2015")

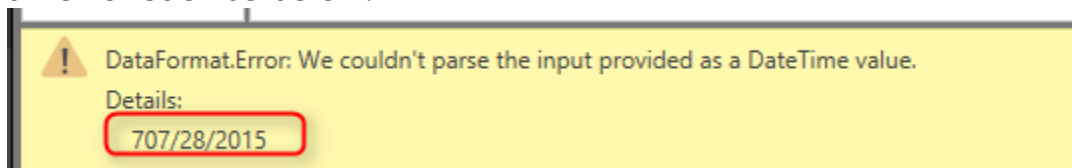
```

and the result is:



## 6- Error Handling

The function is working, but if the given date format is wrong then we will face an error such as below:



So let's add few lines of error handling to the code. We can simply use Try clause to the code as below:

```

1 let
2     DayNumberOfYear= (date) =>
3         let
4             dated= try DateTime.FromText(date),
5             month=Date.Month(dated[Value]),
6             MonthList=List.Numbers(1,month-1),
7             year=Date.Year(dated[Value]),
8             TransformedMonthList=List.Transform
9                 (MonthList,
10                  each Text.From(year)&"-"&Text.From(_)&"-1"),
11             DateList=List.Transform(
12                 TransformedMonthList,
13                 each DateTime.FromText(_)),
14             DaysList=List.Transform(

```

```

15         DateList,
16         each Date.DaysInMonth(_))
17     in
18     if dated[HasError]
19     then dated[Error]
20     else List.Sum(DaysList)+Date.Day(dated[Value])
21 in
22 DayNumberOfYear("707/28/2015")

```

Result for a bad formatted given date is :

fx = let	
Reason	DataFormat.Error
Message	We couldn't parse the input provided as a DateTime value.
Detail	707/28/2015

Here is the full code of the script:

```

1 let
2     DayNumberOfYear= (date) =>
3     let
4         dated= try DateTime.FromText(date),
5         month=Date.Month(dated[Value]),
6         MonthList=List.Numbers(1,month-1),
7         year=Date.Year(dated[Value]),
8         TransformedMonthList=List.Transform
9             (MonthList,
10             each Text.From(year)&"-"&Text.From(_)&"-1"),
11         DateList=List.Transform(
12             TransformedMonthList,
13             each DateTime.FromText(_)),
14         DaysList=List.Transform(
15             DateList,
16             each Date.DaysInMonth(_))
17     in
18     if dated[HasError]
19     then dated[Error]

```

```
20         else List.Sum(DaysList)+Date.Day(dated[Value])
21 in
22     DayNumberOfYear("707/28/2015")
```

In this post you've learned:

- A Function that calculates Day Number of Year for a given date
- Creating a Custom Function
- using Generators as Loop structure
- Error Handling

# Power Query Function that Returns Multiple Values

Posted by [Reza Rad](#) on Jul 30, 2015

```

let
    FirstAndLastDayOfTheMonth = (date) =>
        let
            dated=Date.FromText(date) ,
            year=Date.Year(dated) ,
            month=Date.Month(dated) ,
            FirstDateText=Text.From(year)&"-"&Text.From(month)&"-01" ,
            FirstDate=Date.FromText(FirstDateText) ,
            daysInMonth=Date.DaysInMonth(dated) ,
            LastDateText=Text.From(year)&"-"&Text.From(month)&"-"&Text.From(daysInMonth) ,
            LastDate=Date.FromText(LastDateText) ,
            record=Record.AddField([], "First Date of Month", FirstDate) ,
            resultSet=Record.AddField(record, "Last Date of Month", LastDate)
        in
            resultSet
in
    FirstAndLastDayOfTheMonth("30/07/2015")

```

Yesterday in [NZ BI user group meeting](#), I had been asked that does Power Query custom functions return only one value as the result set? Or they can return multiple values. I've answered. Yes, and I've explained that through a sentence how to do it with Records, List, or Table. Then I thought this might be a question of many people out there. So I've written this blog post to illustrate how to return multiple values from a custom function in Power Query.

If you don't know how to create a custom function, please read my other [blog post with an example of Day Number of Year function for Power Query](#). In this post, I'll show you through an example of how to return multiple results from a Power Query function.

As you probably know Power Query function return single value by default, and that is the value result of the operation in the **"in"** clause of the function. Now how to return multiple values? Simply by returning different type of object. The trick is that Power Query custom function can return any single

object. And that *object* can be simple structure object such as Date, Text, Number. Or it can be multiple value objects such as Record, List, and Table.

To understand the difference between Record, List, and Table:

- Record: Is a single record structure with one or more fields. Each field has a field name and a field value.
- List: Is a single column structure with one or more rows. Each row contains a value.
- Table: Multiple rows and columns data structure (as you probably all know it)

Above objects can hold multiple values. So the only thing you need to do in return one of the above objects based on your requirement. In the example below, I've returned a Record. As a result set, but you can do it with other two data types.

## Return First and Last Dates of Month

As an example, I would like to write a function that fetches both first and last date of a month, the input parameter of this function is a date value with text data types, such as "30/07/2015".

**\*\* Note that Date Conversion function is locale dependent. So if the date time of your system is no DD/MM/YYYY then you have to enter date as it formatted in your system (look below the clock on right-hand side bottom of your monitor to check the format).**

Let's start by calculation of the First Date of the Month

### First Date of the Month

We need to fetch the year, and the month, and then built a date string for the first day (day 1) of that month and year, and finally convert it to Date datatype. Here is the script:

```
1 let
```

```
2   FirstAndLastDayOfTheMonth = (date) => //function definition
```

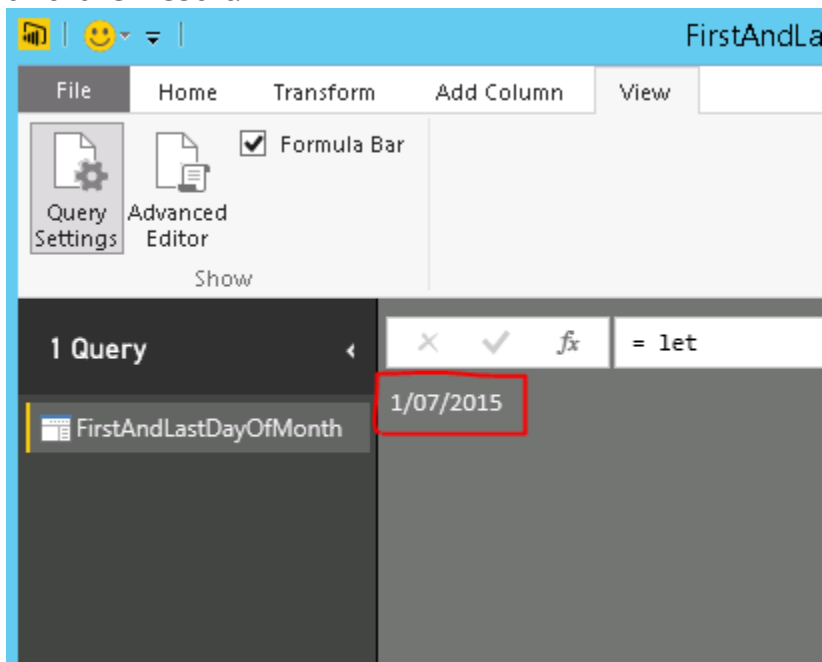


```

3      let
4          dated=Date.FromText(date),//convert input text to Date
5          year=Date.Year(dated),//fetch year
6          month=Date.Month(dated),//fetch month
7          FirstDateText=Text.From(year)&"-"&Text.From(month)&"-
8 01",//generate text value of the first date
9          FirstDate=Date.FromText(FirstDateText)//convert text value to
10 date
11      in
12          FirstDate//return result of the function
in
    FirstAndLastDayOfTheMonth("30/07/2015")//function call

```

and the Result:



## Last Date of the Month

For fetching the last date of the month, we use the same method of the first date, except one change. Which is the day part of the calculation should be the number of days in the month, which comes from `Date.DaysInMonth` function.

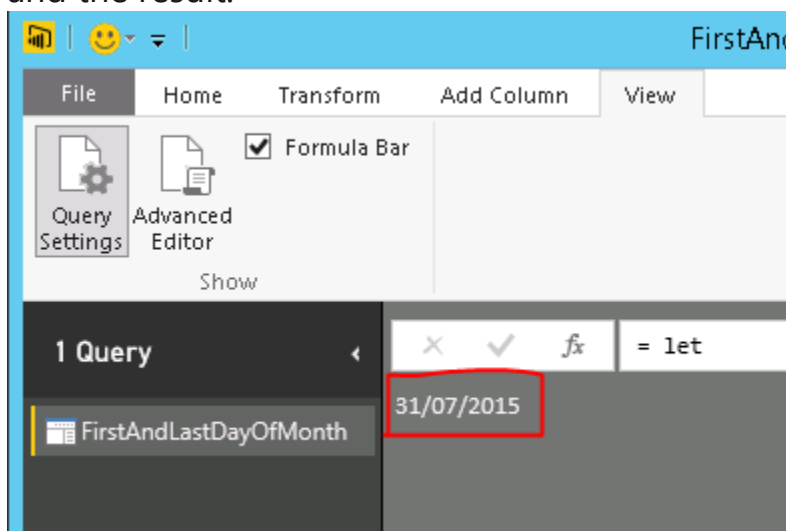
Here is the script:

```

let
1  FirstAndLastDayOfTheMonth = (date) =>
2      let
3          dated=Date.FromText(date),
4          year=Date.Year(dated),
5          month=Date.Month(dated),
6          FirstDateText=Text.From(year)&"-"&Text.From(month)&"-01",
7          FirstDate=Date.FromText(FirstDateText),
8          daysInMonth=Date.DaysInMonth(dated),//fetch number of days
9  in month
10         LastDateText=Text.From(year)&"-"&Text.From(month)&"-
11 "&Text.From(daysInMonth),
12         LastDate=Date.FromText(LastDateText)
13     in
14         LastDate
15 in
    FirstAndLastDayOfTheMonth("30/07/2015")

```

and the result:



### Combining both values into a Record and returning Record as a Result

Now we have both values, and we want to return them both. I'll create an empty record first. An Empty record can be created simply with this : `{ }`.

Then I used Record.AddField function to add fields one by one.

Record.AddField gets three parameters: the record that field will be added to it, the name of the new field, and value of the new field.

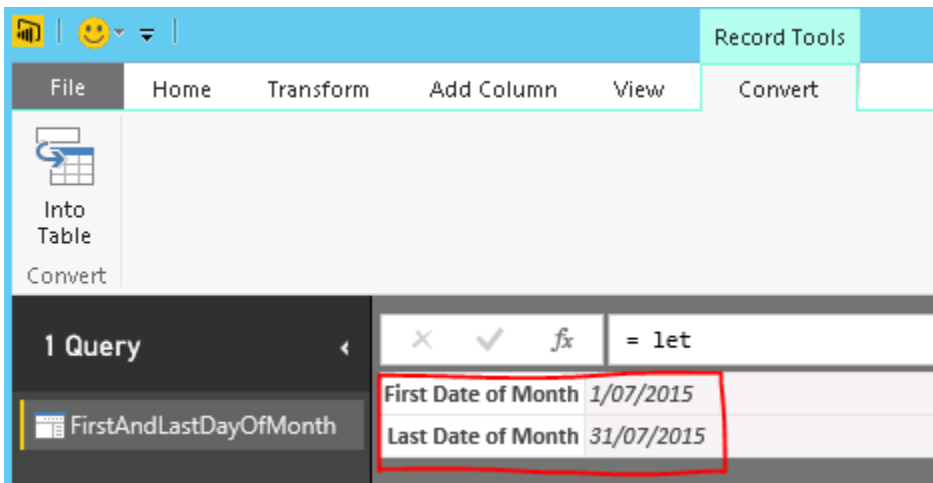
Here is the script:

```

1  let
2      FirstAndLastDayOfTheMonth = (date) =>
3          let
4              dated=Date.FromText(date),
5              year=Date.Year(dated),
6              month=Date.Month(dated),
7              FirstDateText=Text.From(year)&"-"&Text.From(month)&"-01",
8              FirstDate=Date.FromText(FirstDateText),
9              daysInMonth=Date.DaysInMonth(dated),
10             LastDateText=Text.From(year)&"-"&Text.From(month)&"-"
11             &Text.From(daysInMonth),
12             LastDate=Date.FromText(LastDateText),
13             record=Record.AddField([], "First Date of Month", FirstDate),
14             resultset=Record.AddField(record, "Last Date of Month", LastDate)
15         in
16             resultset
17 in
    FirstAndLastDayOfTheMonth("30/07/2015")

```

and the result:



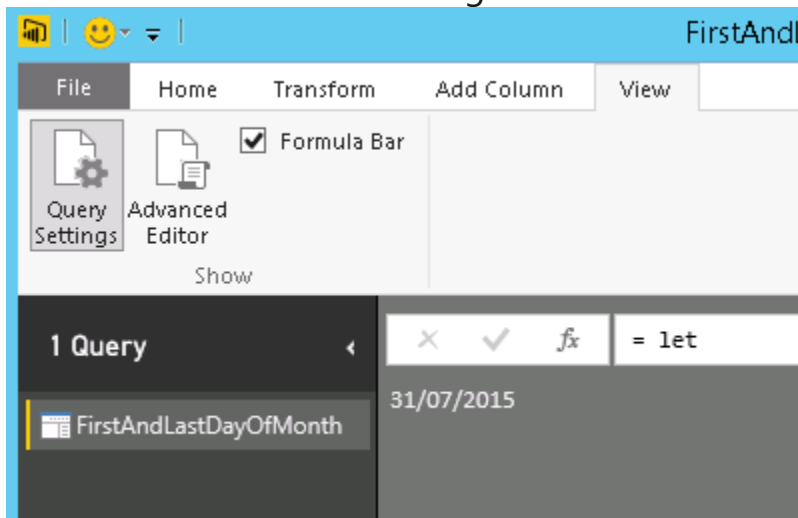
First Date of Month	Last Date of Month
1/07/2015	31/07/2015

How to access this Record's values

As you see in above, the function returns a record with two fields. Now we can access fields by name, with a script like this:

*1 FirstAndLastDayOfTheMonth("30/07/2015")[Last Date of Month]*

and the result would be a single value



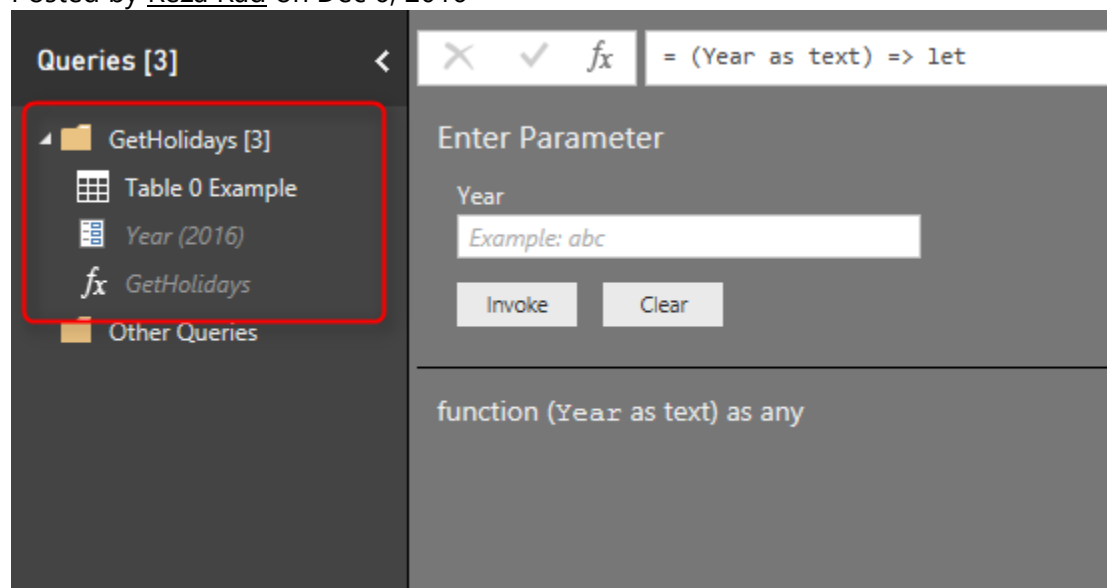
Here is the full code of example if you want to try it yourself.

## Don't Limit Yourself

You can have another record in a field's value, you can have a list in a field's value, and you can have a table in a field's value. So you can create any data structure that you want as the result set of your function.

# Custom Functions Made Easy in Power BI Desktop

Posted by [Reza Rad](#) on Dec 6, 2016



I have written a lot about [Power Query M scripting language](#), and [how to create custom functions](#) with that. With recent updates of Power BI Desktop, creating custom functions made easier and easier every month. This started with bringing Parameters a few months ago and adding source query for the function in [November update of Power BI Desktop](#). In this blog post you will learn how easy it is to create a custom function now, what are benefits of doing it in this way, and limitations of it. If you like to learn more about Power BI; read [Power BI online book from Rookie to Rock Star](#).

## What is Custom Function?

Custom Function in the simple definition is a query that runs by other queries. The main benefit of having a query to run by other queries is that you can repeat some steps on the same data structure. Let's see that as an example: Website below listed public holidays in New Zealand:

<http://publicholiday.co.nz/>

For every year, there is a page, and pages are similar to each other, each page contains a table of dates and descriptions of holidays. Here is, for example, public holidays of 2016:

<http://publicholiday.co.nz/nz-public-holidays-2016.html>



	Holiday	Date
✓	New Year's Day	January, Friday 1 <sup>st</sup>
✓	Day after New Year's Day	January, Saturday 2 <sup>nd</sup> (observed Monday 4 <sup>th</sup> )
✓	Wellington Anniversary	January, Monday 25 <sup>th</sup>
✓	Auckland Anniversary	February, Monday 1 <sup>st</sup>
✓	Nelson Anniversary	February, Monday 1 <sup>st</sup>
✓	Waitangi Day	February, Saturday 6 <sup>th</sup>

You can simply use Power BI Get Data from Web menu option of Power Query to get the public holidays of this page. You can also make some changes in the date format to make it a proper Date data type. Then you probably want to apply same steps on all other pages for other years (2015, 2017, 2018...), So instead of repeating the process, you can reuse an existing query. Here is where the Custom Function comes to help.

## Benefits of Custom Function

- Re-Use of Code
- Increasing Consistency
- Reducing Redundancy

With a Custom function, you can re-use a query multiple times. If you want to change part of it, there is only one place to make that change, instead of

multiple copies of that. You can call this function from everywhere in your code. And you are reducing redundant steps which normally causes extra maintenance of the code.

## How to Create a Custom Function?

Well, that's the question I want to answer in this post. Previously (about a year ago), creating custom functions was only possible through M scripting, with lambda expressions, that method still works, but you need to be comfortable with writing M script to use that method (To be honest I am still a fan of that method). Recently Power BI Desktop changed a lot; you can now create a function without writing any single line of code.

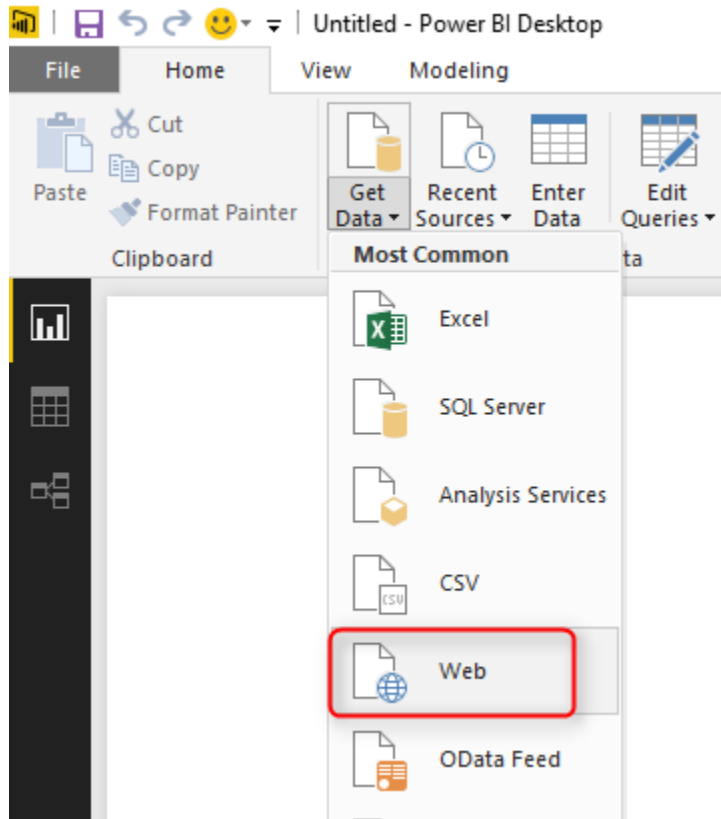
Through an example, I'll show you how to create a custom function. This example is fetching all public holidays from the website above, and appending them all in a single table in Power Query. We want the process to be dynamic, so if a new year's data appear in that page, that will be included as well. Let's see how it works.

### Building the Main Query

For creating a custom function you always need to build the main query and then convert that to a function. Our main query is a query that will be repeated later on by calling from other queries. In this example, our main query is the query that processes the holiday's table in every page, and return that in the proper format of Date data type and Text for a description of the holiday. We can do that for one of the years (doesn't matter which one). I start with the year 2016 which has this URL:

<http://publicholiday.co.nz/nz-public-holidays-2016.html>

Open a Power BI Desktop and start by getting Data from Web



Use the 2016's web address in the From Web page;

## From Web

☒ Basic ☐ Advanced

Enter a Web page URL.

URL

<http://publicholiday.co.nz/nz-public-holidays-2016.html>

OK

Cancel

In the Navigator, you can see that Table 0 is the table containing the data we are after. Select this table and click Edit.



Navigator

Table View Web View

Table 0

	Holiday	Date
null	New Year's Day	January, Friday 1st
null	Day after New Year's Day	January, Saturday 2nd (observed Monday 4th)
null	Wellington Anniversary	January, Monday 25th
null	Auckland Anniversary	February, Monday 1st
null	Nelson Anniversary	February, Monday 1st
null	Waitangi Day	February, Saturday 6th (observed Monday 8th)
null	Taranaki Anniversary	March, Monday 14th
null	Otago Anniversary	March, Monday 21st
null	Good Friday	March, Friday 25th
null	Easter Monday	March, Monday 28th
null	Easter Tuesday ?	March, Tuesday 29th
null	Southland Anniversary	March, Tuesday 29th
null	Daylight Saving ends	April, Sunday 3rd
null	ANZAC Day	April, Monday 25th
null	Queen's Birthday	June, Monday 6th
null	Daylight Saving starts	September, Sunday 25th
null	South Canterbury Anniversary	September, Monday 26th
null	Hawke's Bay Anniversary	October, Friday 21st
null	Labour Day	October, Monday 24th
null	Marlborough Anniversary	October, Monday 31st
null	Canterbury Anniversary	November, Friday 11th

Load Edit Cancel

This will open Query Editor Window for you; You can now make some changes in the query, For example, remove the first column.

Untitled - Query Editor

File Home Transform Add Column View

Close & Apply New Source Recent Sources Enter Data Data source settings Manage Parameters Refresh Preview Properties Advanced Editor Choose Columns Remove Columns Keep Rows Remove Rows

Queries [1]

Table 0

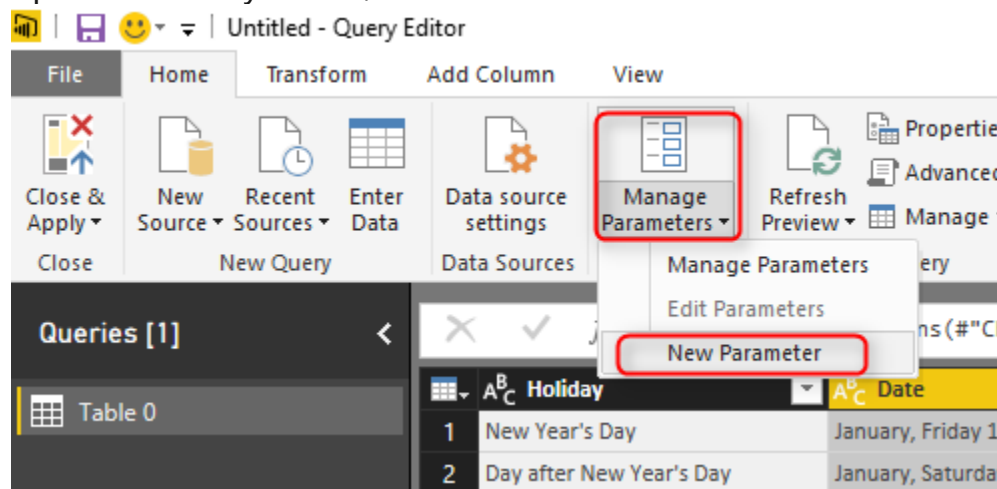
Table.TransformColumnTypes(Data0,{{" ", type text}, {"Holiday", type

	Holiday	Date
1	January, Friday 1st	January, Friday 1st
2	January, Saturday 2nd (observed Monday 4th)	January, Saturday 2nd (observed Monday 4th)
3	January, Monday 25th	January, Monday 25th
4	February, Monday 1st	February, Monday 1st
5	February, Monday 1st	February, Monday 1st
6	February, Saturday 6th (observed Monday 8th)	February, Saturday 6th (observed Monday 8th)
7	March, Monday 14th	March, Monday 14th

The first column was an empty column which we removed. Now you can see two columns; Holiday (which is the description of the holiday), and Date. Date column is a Text data type, and it doesn't have year part in it. If you try to convert it to Date data type, you will either get an error in each cell or incorrect date as a result (depends on the locale setting of your computer). To convert this text to a date format, we need to bring a Year value in the query. The year value for this query can be statistically set to 2016. But because we want to make it dynamic so let's use a Parameter. This Parameter later will be used as input of the query.

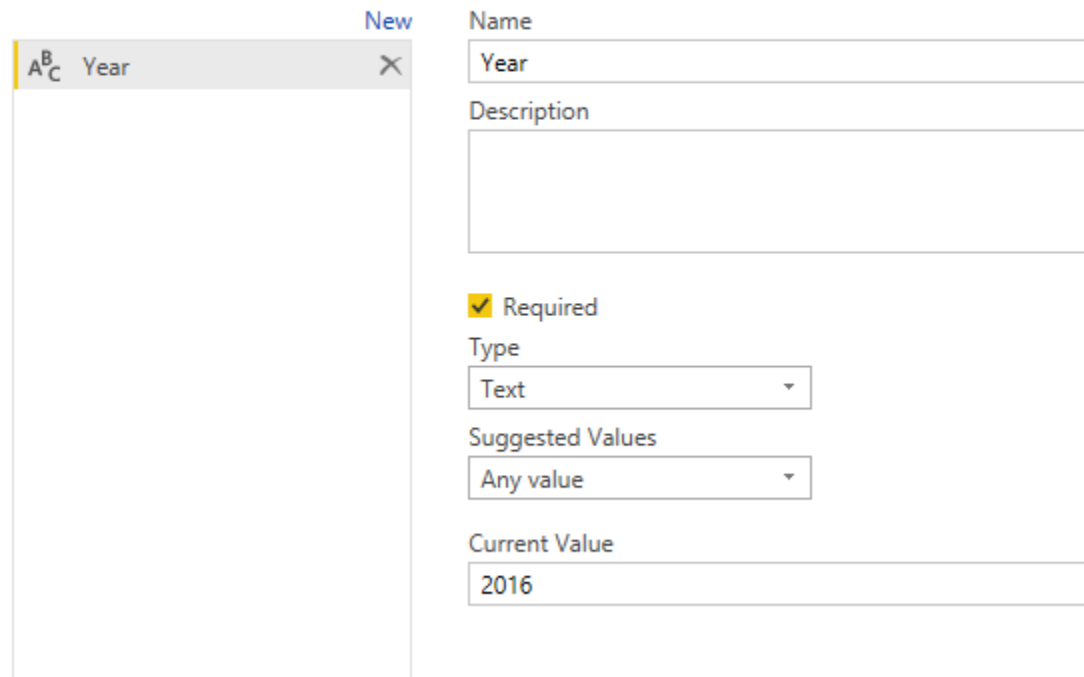
### Parameter Definition

Parameters are ways to pass values to other queries. Normally for custom functions, you need to use parameters. Click on Manage Parameters menu option in Query Editor, and select New Parameter.



There are different types of parameters you can use, but to keep it simple, create a parameter of type Text, with all default selections. Set the Current Value to be 2016. And name it as Year.

## Parameters



**Parameters**

**Name**  
Year

**Description**

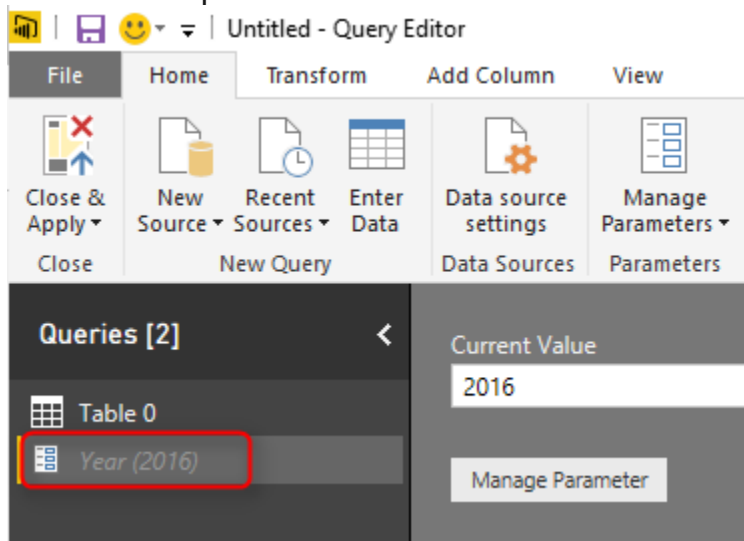
☒ **Required**

**Type**  
Text

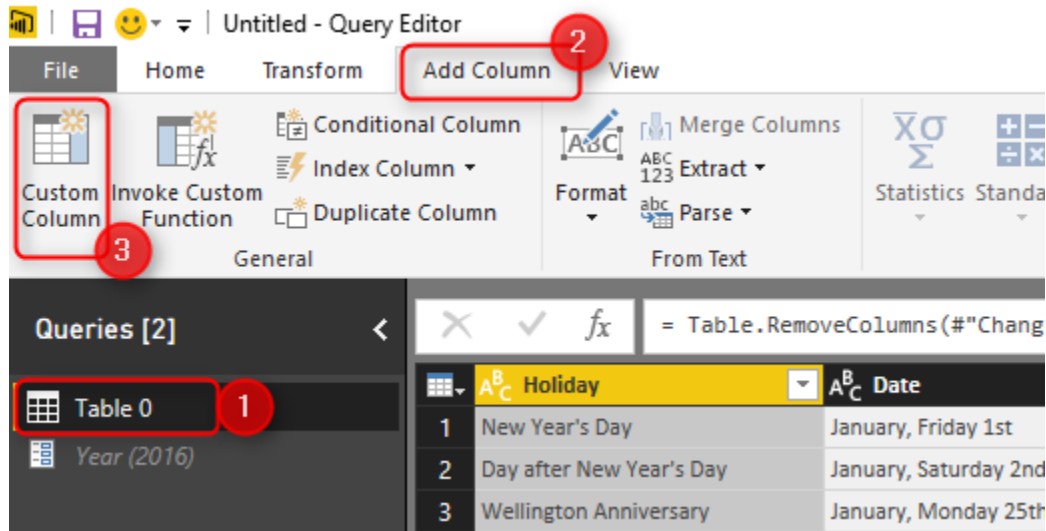
**Suggested Values**  
Any value

**Current Value**  
2016

After creating the Parameter, you can see that in Queries pane with a specific icon for the parameter.



Now we can add a column in Table 0 with the value from this parameter. Click on Table 0, and from Add Column menu option, click on Add Custom Column.



Name the Custom column as Year, and write the expression to be equal Year (remember names in Power Query are case sensitive)

## Add Custom Column

New column name

Year

Name of Column

Custom column formula:

= Year

Parameter Name

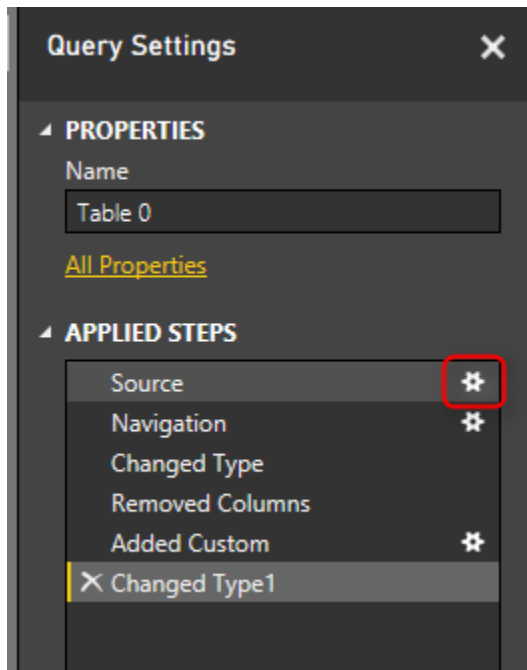
Now you can see year value added to the table. I have also changed its data type to be Text

	<sup>B</sup> <sub>C</sub> Holiday	<sup>B</sup> <sub>C</sub> Date	<sup>B</sup> <sub>C</sub> Year
1	New Year's Day	January, Friday 1st	2016
2	Day after New Year's Day	January, Saturday 2nd (observed Monday 4th)	2016
3	Wellington Anniversary	January, Monday 25th	2016
4	Auckland Anniversary	February, Monday 1st	2016
5	Nelson Anniversary	February, Monday 1st	2016
6	Waitangi Day	February, Saturday 6th (observed Monday 8th)	2016
7	Taranaki Anniversary	March, Monday 14th	2016
8	Otago Anniversary	March, Monday 21st	2016
9	Good Friday	March, Friday 25th	2016
10	Easter Monday	March, Monday 28th	2016
11	Easter Tuesday ?	March, Tuesday 29th	2016
12	Southland Anniversary	March, Tuesday 29th	2016
13	Daylight Saving ends	April, Sunday 3rd	2016
14	ANZAC Day	April, Monday 25th	2016
15	Queen's Birthday	June, Monday 6th	2016
16	Daylight Saving starts	September, Sunday 25th	2016
17	South Canterbury Anniversary	September, Monday 26th	2016
18	Hawke's Bay Anniversary	October, Friday 21st	2016
19	Labour Day	October, Monday 24th	2016

Now that we have created Parameter we can use that parameter as an input to the query's source URL as well.

## URL Parameterization

One of the main benefits of Parameters is that you can use that in a URL. In our case, the URL which contains 2016 as the year, can be dynamic using this parameter. Also for converting a query to a custom function using parameters is one of the main steps. Let's add Parameter in the source of this query then; While Table 0 query is selected, in the list of Steps, click on the Setting icon for Source step



This will bring the very first step of the query where we get data from Web and provided the URL. Not in the top section change the From Web window to Advanced.



## From Web

☐ Basic ☒ Advanced

Enter a Web page URL.

URL parts

Add Part

URL preview

<http://publicholiday.co.nz/nz-public-holidays-2016.html>

Open file as

Html Page

Command timeout in minutes (optional)

HTTP Request Header Parameters (optional)

Type or select a value

Add Header

OK

Cancel

The advanced option gives you the ability to split the URL into portions. What we want to do is to put Text portions for beginning and end of the string, and make the year part of it dynamic coming from URL. So Add another part, and put setting as below;

## From Web

☐ Basic ☒ Advanced

Enter a Web page URL.

URL parts

Add Part

URL preview

<http://publicholiday.co.nz/nz-public-holidays-{Year}.html>

The configuration above means that in the first part of URL we put everything before 2016 which is:

<http://publicholiday.co.nz/nz-public-holidays->

The second part of URL is coming from the parameter, and we use Year parameter for that

the third part of URL is the remaining part after the year which is:

[.html](#)

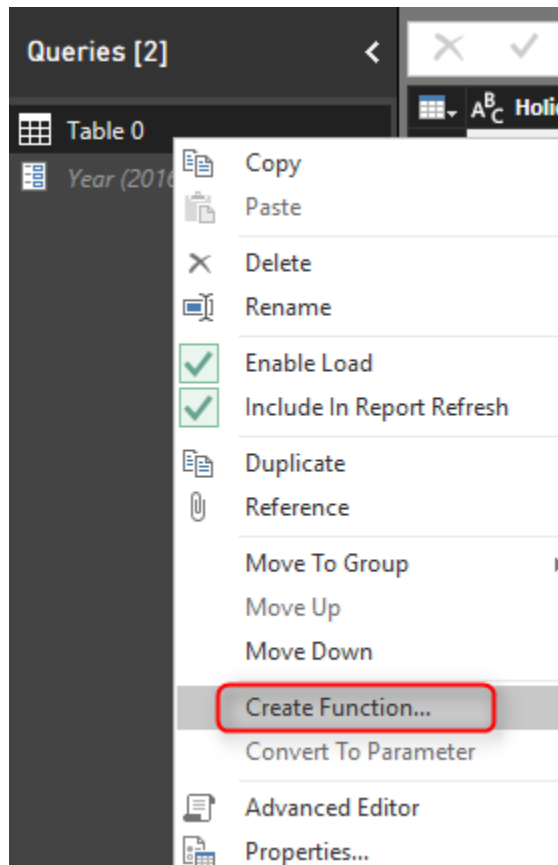
altogether these will make the URL highlighted above which instead of {Year} it will have 2016, or 2015 or other values.

Click on OK. You won't see any changes yet, even if you click on the last step of this query, because we have used same year code as the parameter value. If you change the parameter value and refresh the query, you will see changes, but we don't want to do it in this way.

### Convert Query to Function

After using a parameter in the source of query, we can convert it to function. Right click on the Table 0 query and select Create Function.





Name the function as GetHolidays and click on OK.

## Create Function

Enter a name for the new function.

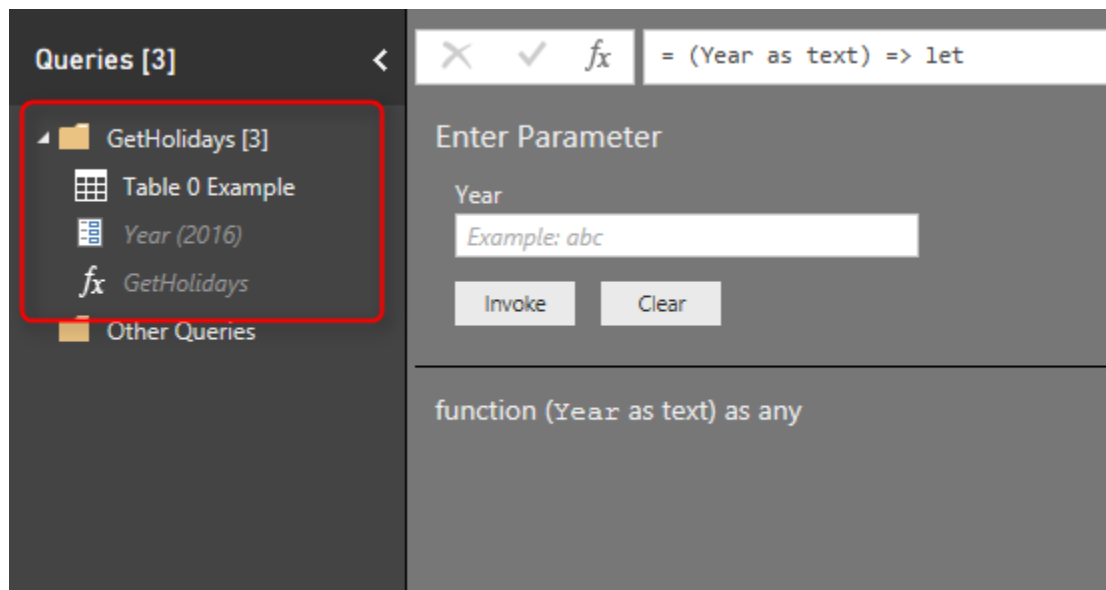
Function name

GetHolidays

OK

Cancel

You will now see a group (folder) created with the name of GetHolidays including three objects; main query (Table 0), Parameter (year), and function (GetHolidays).



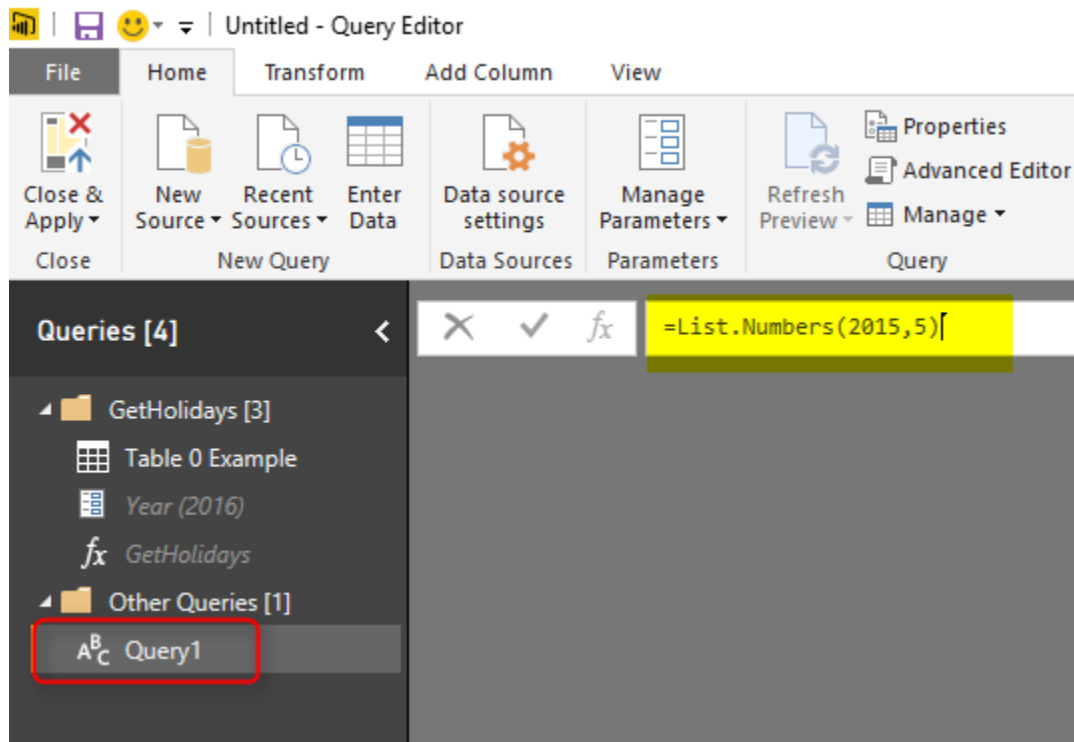
The function itself marked with *fx* icon, and that is the function we will use to call from other queries. However, the main query and parameter are still necessary for making changes in the function. I will explain this part later. All happened here is that there is a copy of the Table 0 query created as a function. And every time you call this function with an input parameter (which will be year value), this will give you the result (which is public holidays table for that year). Let's now consume this table from another query, but before that let's create a query that includes a list of years.

## Using Generator

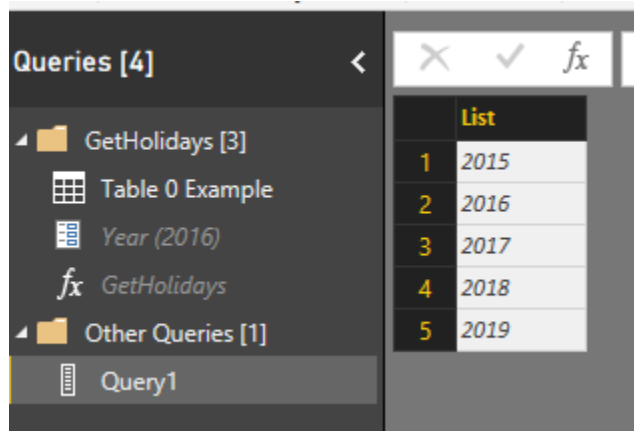
Generators are a topic of its own and can't be discussed in this post. All I can tell you for now is that Generators are functions that generate a list. This can be used for creating loop structure in Power Query. I'll write about that in another post. For this example, I want to create a list of numbers from 2015 for five years. So I'll use List.Numbers generator function for that. In your Query Editor Window, create a New Source from Home tab, and choose Blank Query.

This will create a Query1 for you. Click on Query1 in Queries pane, and in the Formula bar type in below script:

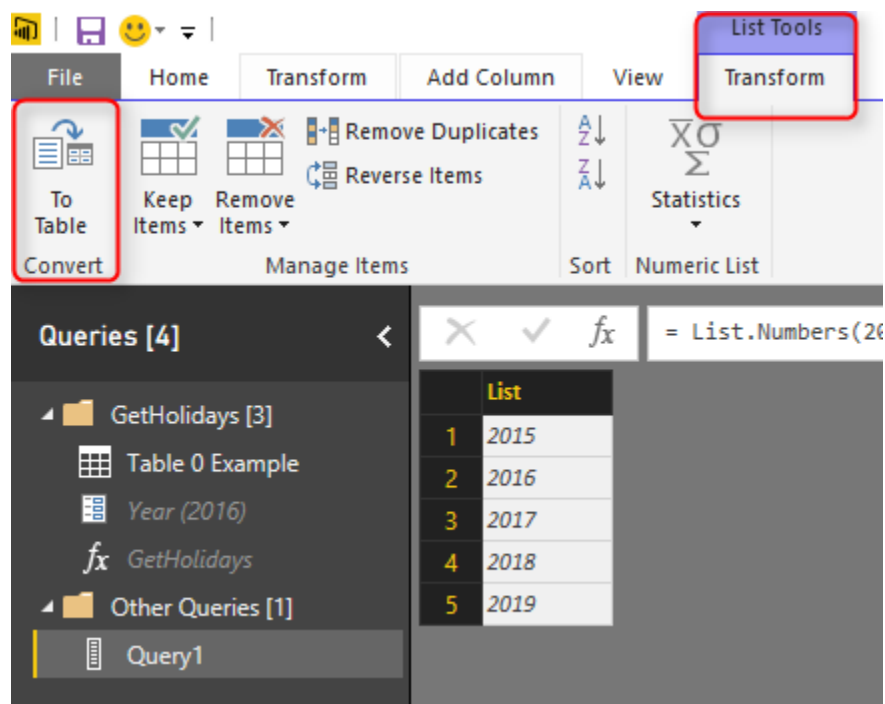
```
1 = List.Numbers(2015,5)
```



After entering the expression press Enter and you will see a list generated from 2015 for five numbers. That's the work done by a generator function.



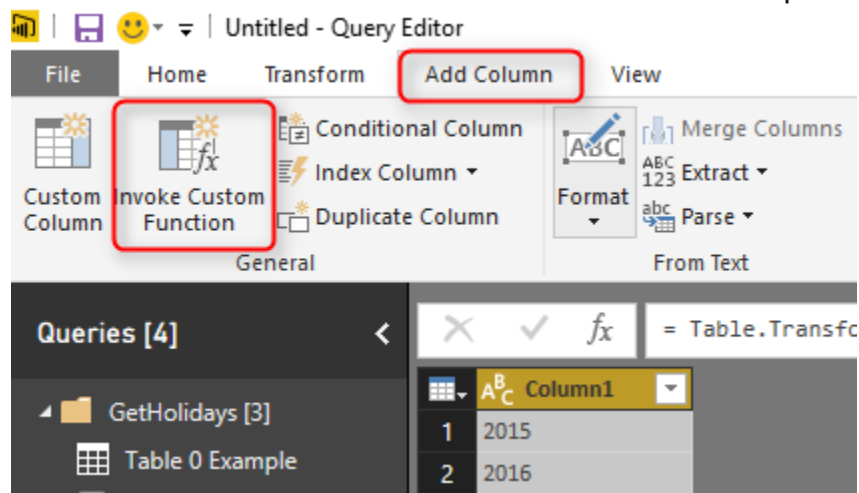
This is a List and can be converted to Table simple from List Tools menu option.



Convert this to Table with all default settings, and now you can see a table with Column1 which is year value. Because the value is the whole number, I have changed it to Text as well (to match the data type of parameter).

## Consuming Function

Consuming a function in Query Editor from a table is easy. Go to Add Columns, and click on Invoke Custom Function option.



In the Invoke Custom Function window, choose the function (named GetHolidays), the input parameter is from the table column name Column1, and name the output column as Holidays.

## Invoke Custom Function

Invoke a custom function defined in this file for each row.

New column name

Holidays

Function query

GetHolidays

Year

Column1

OK

Cancel

Now when you click on OK, you will see a new column added with a table in each cell. These tables are results of calling that function with the input parameter which is the value of Column1 in each row. If you click on a blank area of a cell with Table hyperlink, you will see the table structure below it.

	Column1	Holidays
1	2015	Table
2	2016	Table
3	2017	Table
4	2018	Table
5	2019	Table

Holiday	Date	Year
New Year's Day	January, Sunday 1st (observed Tuesday 3rd)	2017
Day after New Year's Day	January, Monday 2nd	2017
Wellington Anniversary	January, Monday 23rd	2017
Auckland Anniversary	January, Monday 30th	2017
Nelson Anniversary	January, Monday 30th	2017
Waitangi Day	February, Monday 6th	2017
Taranaki Anniversary	March, Monday 13th	2017
Otago Anniversary	March, Monday 20th	2017
Daylight Saving ends	April, Sunday 2nd	2017
Good Friday	April, Friday 14th	2017
Easter Monday	April, Monday 17th	2017
Easter Tuesday ?	April, Tuesday 18th	2017
Southland Anniversary	April, Tuesday 18th	2017
ANZAC Day	April, Tuesday 25th	2017
Queen's Birthday	June, Monday 5th	2017
Daylight Saving starts	September, Sunday 24th	2017
South Canterbury Anniversary	September, Monday 25th	2017
Hawke's Bay Anniversary	October, Friday 20th	2017
Labour Day	October, Monday 23rd	2017
Marlborough Anniversary	October, Monday 30th	2017

Interesting, isn't it? It was so easy to implement. All done from GUI, not a single line of code to run this function or pass parameters, things made easy with custom functions.

## Editing Function

If you want to make modifications in function, you can simply modify the main query (which is Table 0 Example). For example, let's create a full date format from that query in this way;

Click on Table 0 Example query and split the Date column with delimiter Comma, you will end up having a column now for Month values and another for Day (Note that I have renamed this column respectively);

	A <sup>B</sup> <sub>C</sub> Holiday	A <sup>B</sup> <sub>C</sub> Month	A <sup>B</sup> <sub>C</sub> Day	A <sup>B</sup> <sub>C</sub> Year
1	New Year's Day	January	Friday 1st	2016
2	Day after New Year's Day	January	Saturday 2nd (observed Monday 4th)	2016
3	Wellington Anniversary	January	Monday 25th	2016
4	Auckland Anniversary	February	Monday 1st	2016
5	Nelson Anniversary	February	Monday 1st	2016
6	Waitangi Day	February	Saturday 6th (observed Monday 8th)	2016
7	Taranaki Anniversary	March	Monday 14th	2016
8	Otago Anniversary	March	Monday 21st	2016
9	Good Friday	March	Friday 25th	2016
10	Easter Monday	March	Monday 28th	2016
11	Easter Tuesday ?	March	Tuesday 29th	2016
12	Southland Anniversary	March	Tuesday 29th	2016
13	Daylight Saving ends	April	Sunday 3rd	2016
14	ANZAC Day	April	Monday 25th	2016
15	Queen's Birthday	June	Monday 6th	2016
16	Daylight Saving starts	September	Sunday 25th	2016
17	South Canterbury Anniversary	September	Monday 26th	2016
18	Hawke's Bay Anniversary	October	Friday 21st	2016
19	Labour Day	October	Monday 24th	2016
20	Marlborough Anniversary	October	Monday 31st	2016
21	Canterbury Anniversary	November	Friday 11th	2016
22	Westland Anniversary	November	Monday 28th	2016
23	Chatham Islands Anniversary	November	Monday 28th	2016
24	Christmas Day	December	Sunday 25th (observed Tuesday 27th)	2016
25	Boxing Day	December	Monday 26th	2016

Now if you go back to Query1, you will see changes in the results table immediately.

	Column1	Holidays
1	2015	Table
2	2016	Table
3	2017	Table
4	2018	Table
5	2019	Table

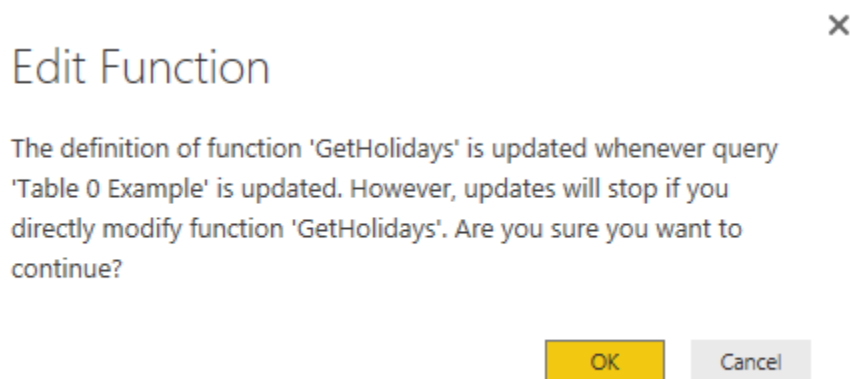
Holiday	Month	Day	Year
New Year's Day	January	Sunday 1st (observed Tuesday 3rd)	2017
Day after New Year's Day	January	Monday 2nd	2017
Wellington Anniversary	January	Monday 23rd	2017
Auckland Anniversary	January	Monday 30th	2017
Nelson Anniversary	January	Monday 30th	2017
Waitangi Day	February	Monday 6th	2017
Taranaki Anniversary	March	Monday 13th	2017
Otago Anniversary	March	Monday 20th	2017
Daylight Saving ends	April	Sunday 2nd	2017
Good Friday	April	Friday 14th	2017
Easter Monday	April	Monday 17th	2017
Easter Tuesday ?	April	Tuesday 18th	2017
Southland Anniversary	April	Tuesday 18th	2017
ANZAC Day	April	Tuesday 25th	2017
Queen's Birthday	June	Monday 5th	2017
Daylight Saving starts	September	Sunday 24th	2017
South Canterbury Anniversary	September	Monday 25th	2017
Hawke's Bay Anniversary	October	Friday 20th	2017
Labour Day	October	Monday 23rd	2017
Marlborough Anniversary	October	Monday 30th	2017



## Limitations

### Edit Script for the Function

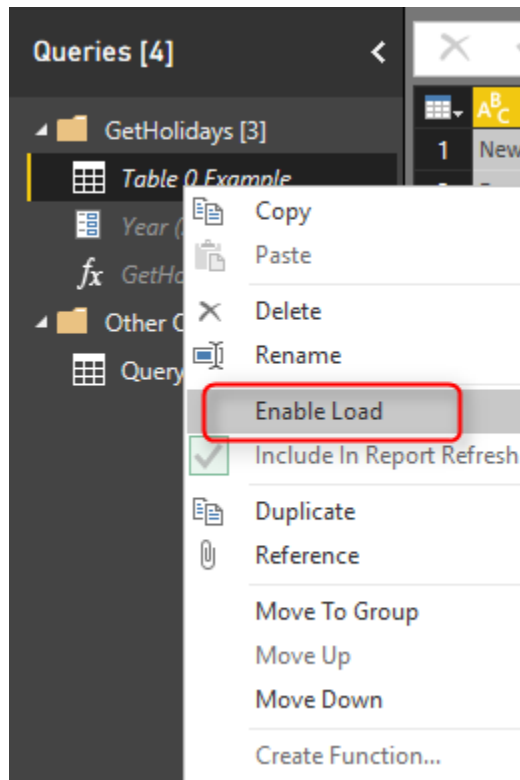
Changes are fine as long as you don't want to edit the M script of the function if you want to do so, then the function definition and the query definition split apart. And you will get below message:



It is alright to make changes in the Advanced Editor of the source query, and then the function will be updated based on that, but if you want to change the function itself, then the query will be separated.

### Disable Load of the Source Query

If you have read my [blog post about Enable Load in Power Query](#), you already know that queries that are not used in the model should not be loaded. By Default, the source query (in this example named as Table 0 Example) will be loaded into the model. This means one extra table, and consuming more memory. So remember to uncheck the Enable Load for this query;

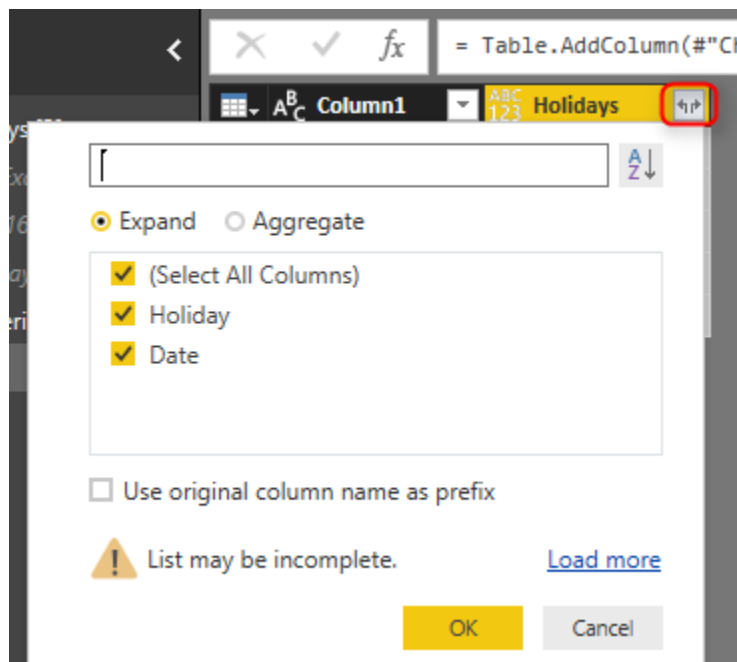


## Parameterized URLs

Custom Functions that use parameterized URLs (like this example) cannot be scheduled to refresh in Power BI. That's a big limitation which I hope to be lifted very quickly.

## Example at the End

I have done some other changes and changed the data type to Date format. Now the final query which is expanded table from all underlying queries include all public holidays. I'll leave that part to you to continue and build rest of the example. For that functionality you need to use Expand;



and here is the final result;

	Holiday	Date
47	Westland Anniversary	11/28/2016
48	Chatham Islands Anniversary	11/28/2016
49	Christmas Day	12/27/2016
50	Boxing Day	12/26/2016
51	New Year's Day	1/3/2017
52	Day after New Year's Day	1/2/2017
53	Wellington Anniversary	1/23/2017
54	Auckland Anniversary	1/30/2017
55	Nelson Anniversary	1/30/2017
56	Waitangi Day	2/6/2017
57	Taranaki Anniversary	3/13/2017
58	Otago Anniversary	3/20/2017
59	Daylight Saving ends	4/2/2017
60	Good Friday	4/14/2017
61	Easter Monday	4/17/2017
62	Easter Tuesday ?	4/18/2017

## Summary

In this post, you have learned how easy it is to create a custom function from a query, all from the graphical interface. You have seen that I haven't wrote any

single line of code to do it. and all happened through GUI. You have seen that you can change the definition of function simply by changing the source query. and you also have seen how easy it is to call/consume a function from another query. This method can be used a lot in real world Power Query scenarios.

# Search for a Column in the Entire Database with Table.ColumnNames in Power Query and Power BI

Posted by [Reza Rad](#) on Jul 10, 2018

fx = Table.TransformColumnTypes("#Expanded Column",{{"Column", type text}})

	A <sup>B</sup> <sub>C</sub> Name	ABC 123 Data	A <sup>B</sup> <sub>C</sub> Schema	A <sup>B</sup> <sub>C</sub> Item	A <sup>B</sup> <sub>C</sub> Kind	A <sup>B</sup> <sub>C</sub> Column
1	AdventureWorksDWBuild...	Table	dbo	AdventureWorksDWBuildVersion	Table	DBVersion
2	AdventureWorksDWBuild...	Table	dbo	AdventureWorksDWBuildVersion	Table	VersionDate
3	DatabaseLog	Table	dbo	DatabaseLog	Table	DatabaseLogID
4	DatabaseLog	Table	dbo	DatabaseLog	Table	PostTime
5	DatabaseLog	Table	dbo	DatabaseLog	Table	DatabaseUser
6	DatabaseLog	Table	dbo	DatabaseLog	Table	Event
7	DatabaseLog	Table	dbo	DatabaseLog	Table	Schema
8	DatabaseLog	Table	dbo	DatabaseLog	Table	Object
9	DatabaseLog	Table	dbo	DatabaseLog	Table	TSQL
10	DatabaseLog	Table	dbo	DatabaseLog	Table	XmlEvent
11	DimAccount	Table	dbo	DimAccount	Table	AccountKey
12	DimAccount	Table	dbo	DimAccount	Table	ParentAccountKey
13	DimAccount	Table	dbo	DimAccount	Table	AccountCodeAlternateKey
14	DimAccount	Table	dbo	DimAccount	Table	ParentAccountCodeAlternateKey
15	DimAccount	Table	dbo	DimAccount	Table	AccountDescription
16	DimAccount	Table	dbo	DimAccount	Table	AccountType
17	DimAccount	Table	dbo	DimAccount	Table	CustomMembers
18	DimAccount	Table	dbo	DimAccount	Table	ValueType
19	DimAccount	Table	dbo	DimAccount	Table	AccountOptions
20	DimAccount	Table	dbo	DimAccount	Table	DimAccount(AccountKey)
21	DimAccount	Table	dbo	DimAccount	Table	DimAccount(ParentAccountKey)
22	DimAccount	Table	dbo	DimAccount	Table	FactFinance
23	DimAccount	Table	dbo	DimAccount	Table	CurrencyKey
24	DimCurrency	Table	dbo	DimCurrency	Table	CurrencyAlternateKey
25	DimCurrency	Table	dbo	DimCurrency	Table	CurrencyName
26	DimCurrency	Table	dbo	DimCurrency	Table	

List of all columns in every table or view

**Search through all columns in a data source with Table.ColumnNames in Power Query and Power BI**

Sometimes for tables with too many columns, and also for databases with too many tables, you do need a bit of help to explore the data. As an example; you know that you are looking for a column named "account status", but the column does not exist in the accounts table. You need to search through all database tables for that column and find out which table has that value in it. Power Query has a great function that can help in such a scenario.

Table.ColumnNames is a function that we are going to check out in this post.

If you want to learn more about Power BI, read [Power BI book from Rookie to Rock Star](#).

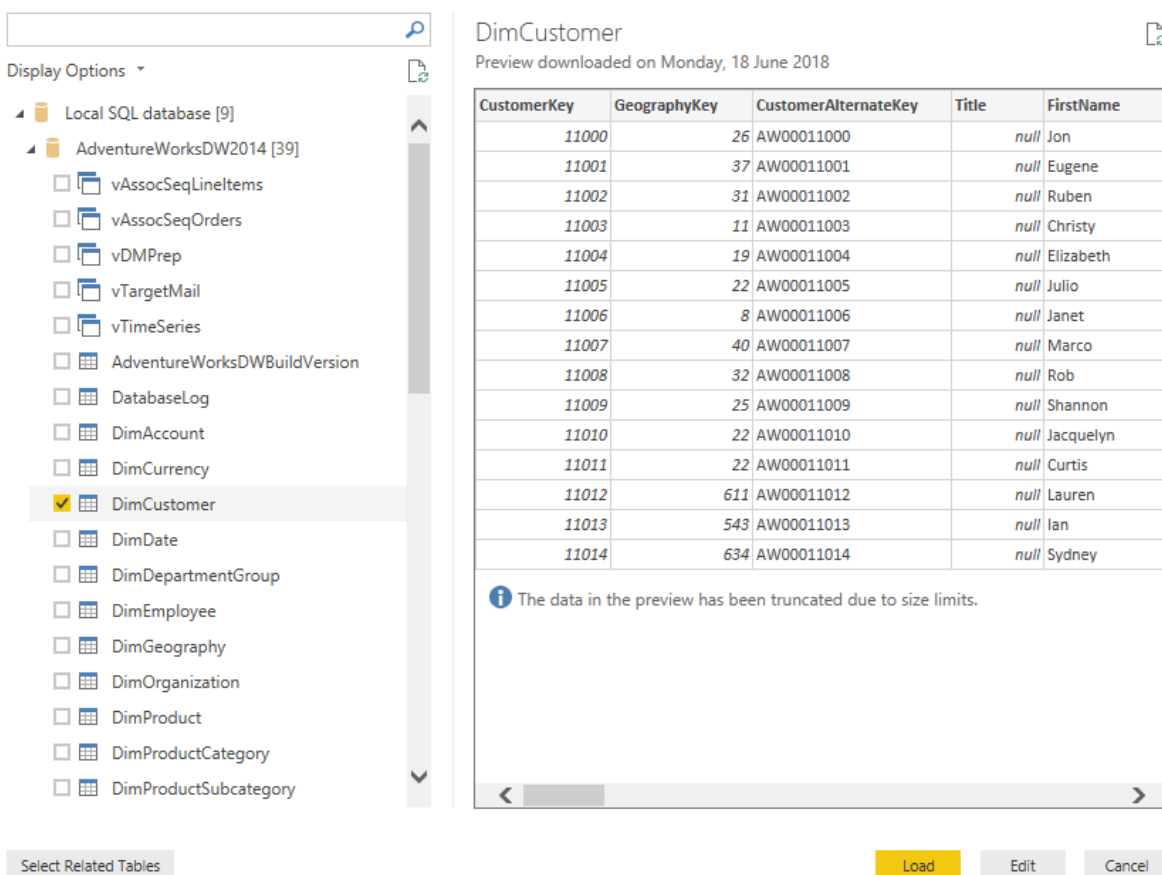
## The Problem

You have a database with hundreds of tables or even more, and each table has many columns. You are looking for a specific column in the database and want to find out tables that such column exists in. Power Query has a function named `Table.ColumnNames` which gives you a list of all columns in a table as the output. Let's see how this function works.

## Sample Dataset

For this example; you can connect to the [AdventureWorksDW database](#). Then select any of the tables to import data from the source database.

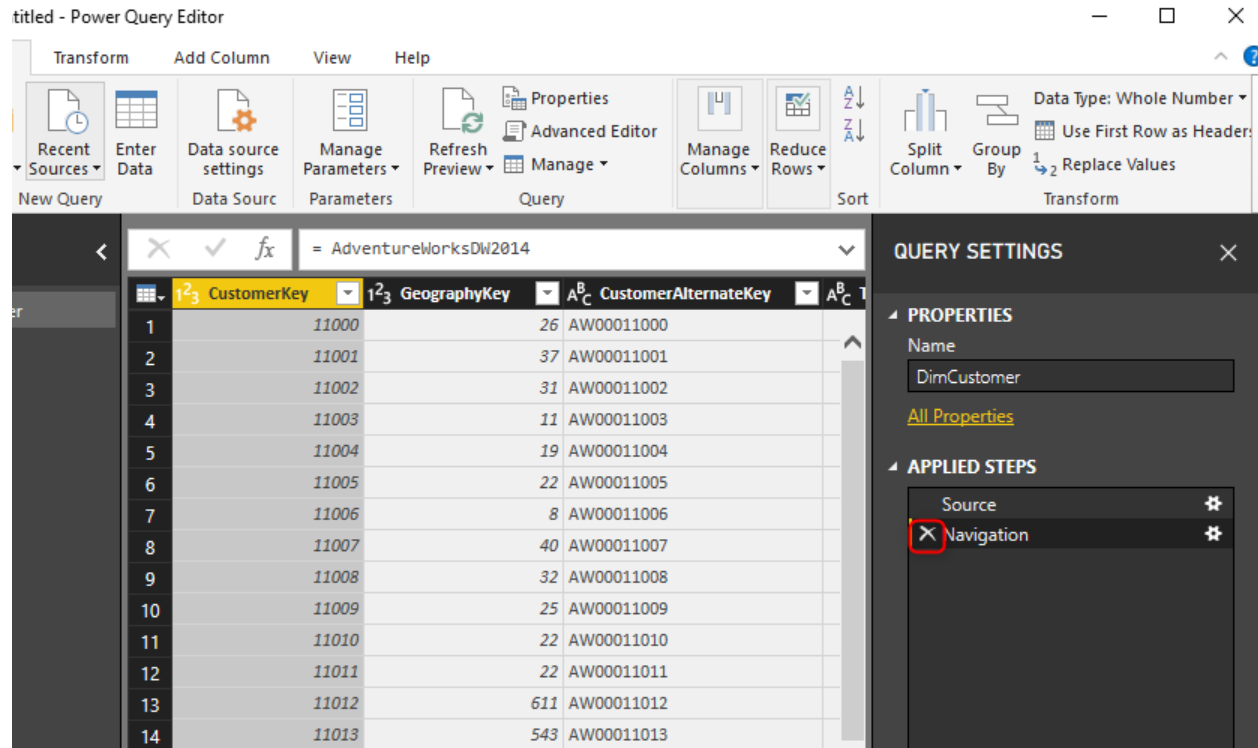
### Navigator



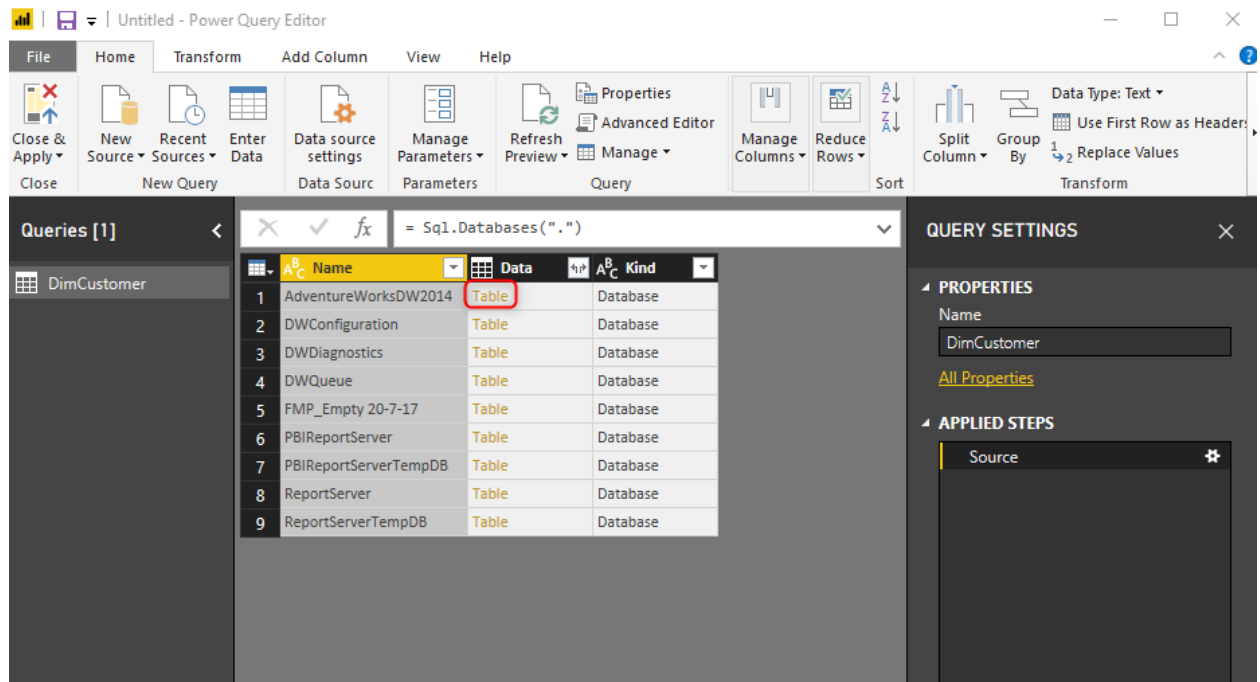
The screenshot shows the Power Query Navigator window. On the left, the 'AdventureWorksDW2014 [39]' database is expanded, listing various tables. 'DimCustomer' is selected with a checkmark. On the right, a preview of the 'DimCustomer' table is displayed, showing columns: CustomerKey, GeographyKey, CustomerAlternateKey, Title, and FirstName. The preview shows 14 rows of data. Below the table, a message states: 'The data in the preview has been truncated due to size limits.' At the bottom of the window, there are buttons for 'Select Related Tables', 'Load', 'Edit', and 'Cancel'.

CustomerKey	GeographyKey	CustomerAlternateKey	Title	FirstName
11000	26	AW00011000	null	Jon
11001	37	AW00011001	null	Eugene
11002	31	AW00011002	null	Ruben
11003	11	AW00011003	null	Christy
11004	19	AW00011004	null	Elizabeth
11005	22	AW00011005	null	Julio
11006	8	AW00011006	null	Janet
11007	40	AW00011007	null	Marco
11008	32	AW00011008	null	Rob
11009	25	AW00011009	null	Shannon
11010	22	AW00011010	null	Jacquelyn
11011	22	AW00011011	null	Curtis
11012	611	AW00011012	null	Lauren
11013	543	AW00011013	null	Ian
11014	634	AW00011014	null	Sydney

In the list of steps; you will find one step named as Navigation. This is the step that we have navigated to the table. Remove this step.

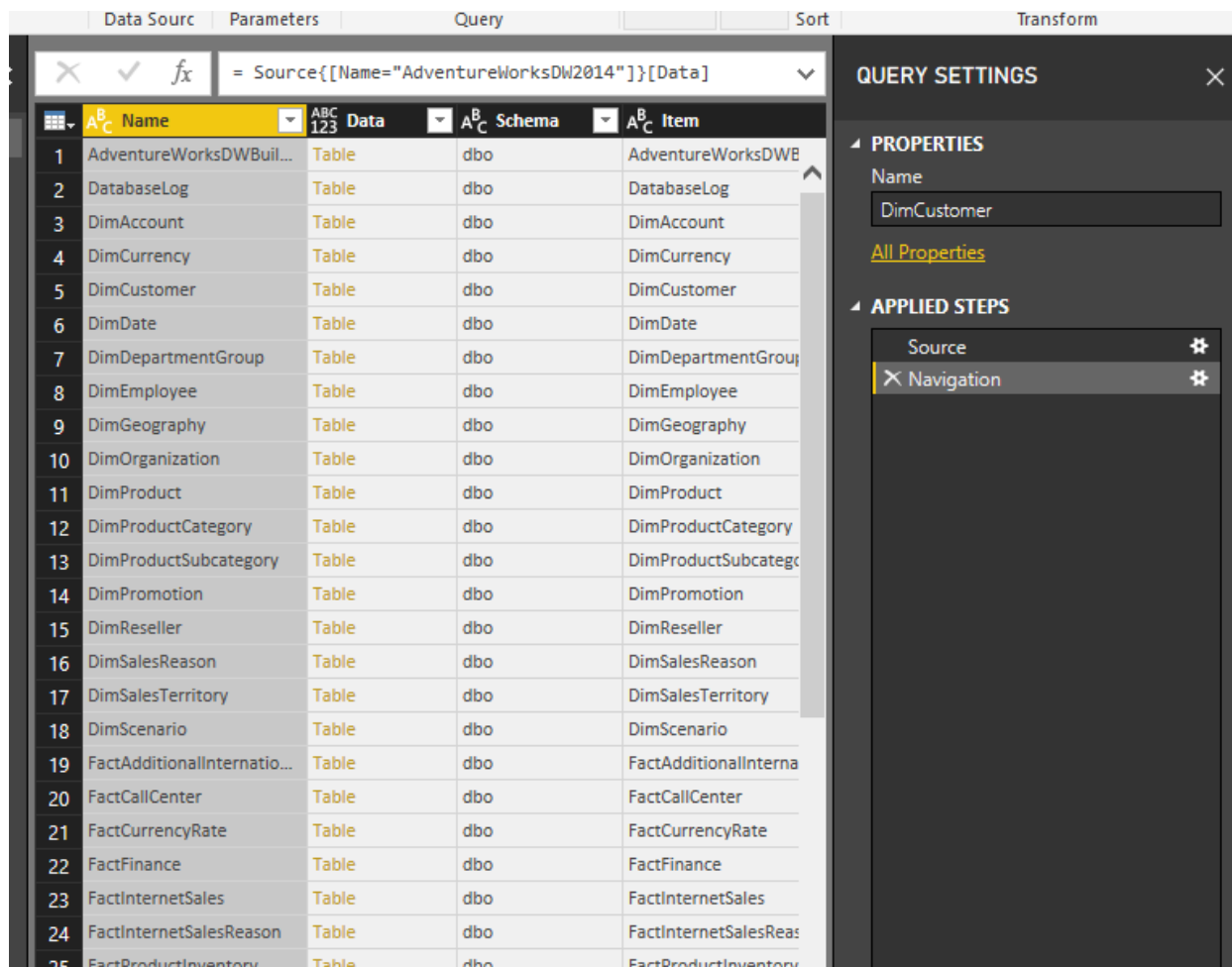


Then you will most probably see the list of all databases that you have access to (if you haven't chosen a specific database at the connect to SQL Server section). Click on the "Table" in the Data row of the database of AdventureWorks2014 (or any other databases that you want to explore columns in it).



This action will give you the list of all tables under that database.





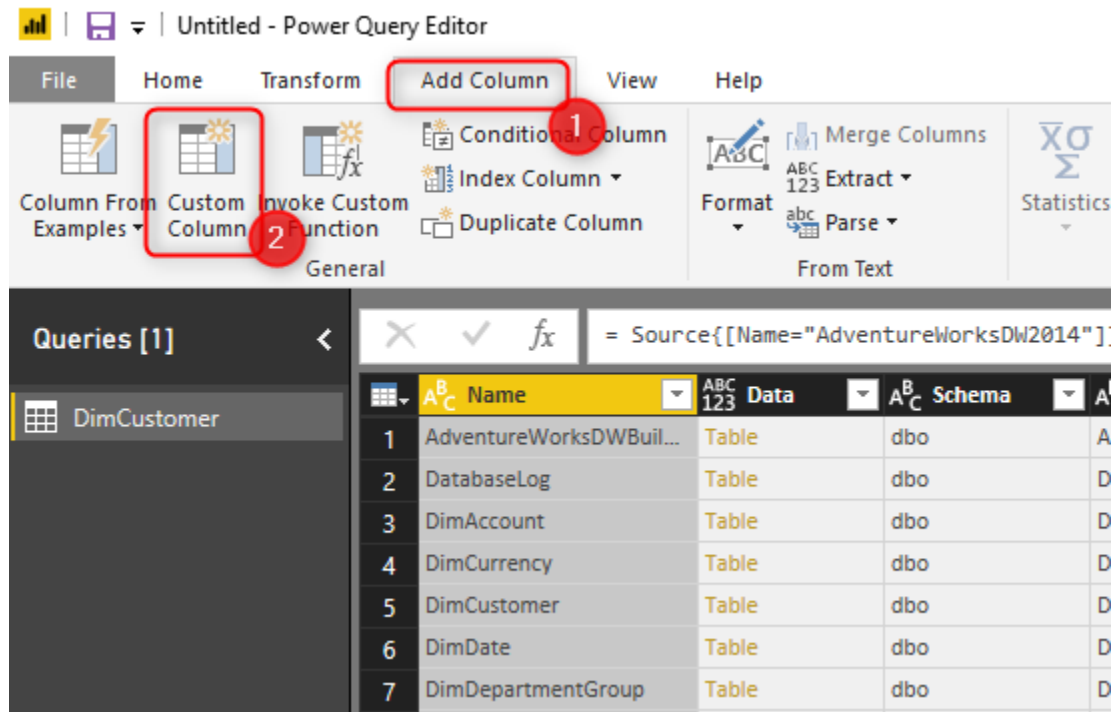
The screenshot displays the Power Query Editor interface. The main area shows a list of tables from the 'AdventureWorksDW2014' database. The 'Name' column is highlighted. The 'QUERY SETTINGS' pane on the right shows the 'Name' property set to 'DimCustomer' and the 'APPLIED STEPS' list containing 'Source' and 'Navigation'.

ABC 123	Name	Data	Schema	Item
1	AdventureWorksDWBuild...	Table	dbo	AdventureWorksDWBuild...
2	DatabaseLog	Table	dbo	DatabaseLog
3	DimAccount	Table	dbo	DimAccount
4	DimCurrency	Table	dbo	DimCurrency
5	DimCustomer	Table	dbo	DimCustomer
6	DimDate	Table	dbo	DimDate
7	DimDepartmentGroup	Table	dbo	DimDepartmentGroup
8	DimEmployee	Table	dbo	DimEmployee
9	DimGeography	Table	dbo	DimGeography
10	DimOrganization	Table	dbo	DimOrganization
11	DimProduct	Table	dbo	DimProduct
12	DimProductCategory	Table	dbo	DimProductCategory
13	DimProductSubcategory	Table	dbo	DimProductSubcategory
14	DimPromotion	Table	dbo	DimPromotion
15	DimReseller	Table	dbo	DimReseller
16	DimSalesReason	Table	dbo	DimSalesReason
17	DimSalesTerritory	Table	dbo	DimSalesTerritory
18	DimScenario	Table	dbo	DimScenario
19	FactAdditionalInternatio...	Table	dbo	FactAdditionalInternatio...
20	FactCallCenter	Table	dbo	FactCallCenter
21	FactCurrencyRate	Table	dbo	FactCurrencyRate
22	FactFinance	Table	dbo	FactFinance
23	FactInternetSales	Table	dbo	FactInternetSales
24	FactInternetSalesReason	Table	dbo	FactInternetSalesReason
25	FactProductInventory	Table	dbo	FactProductInventory

Now let's assume we want to search for Account column in the entire database. Let's see how this is possible.

## Table.ColumnNames

The Table.ColumnNames function in Power Query will give you the list of all the columns in any given table. All you need to do is to call this function by passing the table as the input, and you will get a list of column names as the output. To use it in the existing example, click on Add Column, and then Add Custom Column.



In the Custom Column expression section; you can write the `Table.ColumnNames` function with the input parameter of `Data` (`Data` is the column that includes the data table). Please note that Power Query is case sensitive, and `Table.ColumnNames` should be written exactly as mentioned here in this blog post.

*1 = Table.ColumnNames([Data])*

## Custom Column

New column name

Column

Custom column formula:

`= Table.ColumnNames([Data])`

Available columns:

Name

Data

Schema

Item

Kind

&lt;&lt; Insert

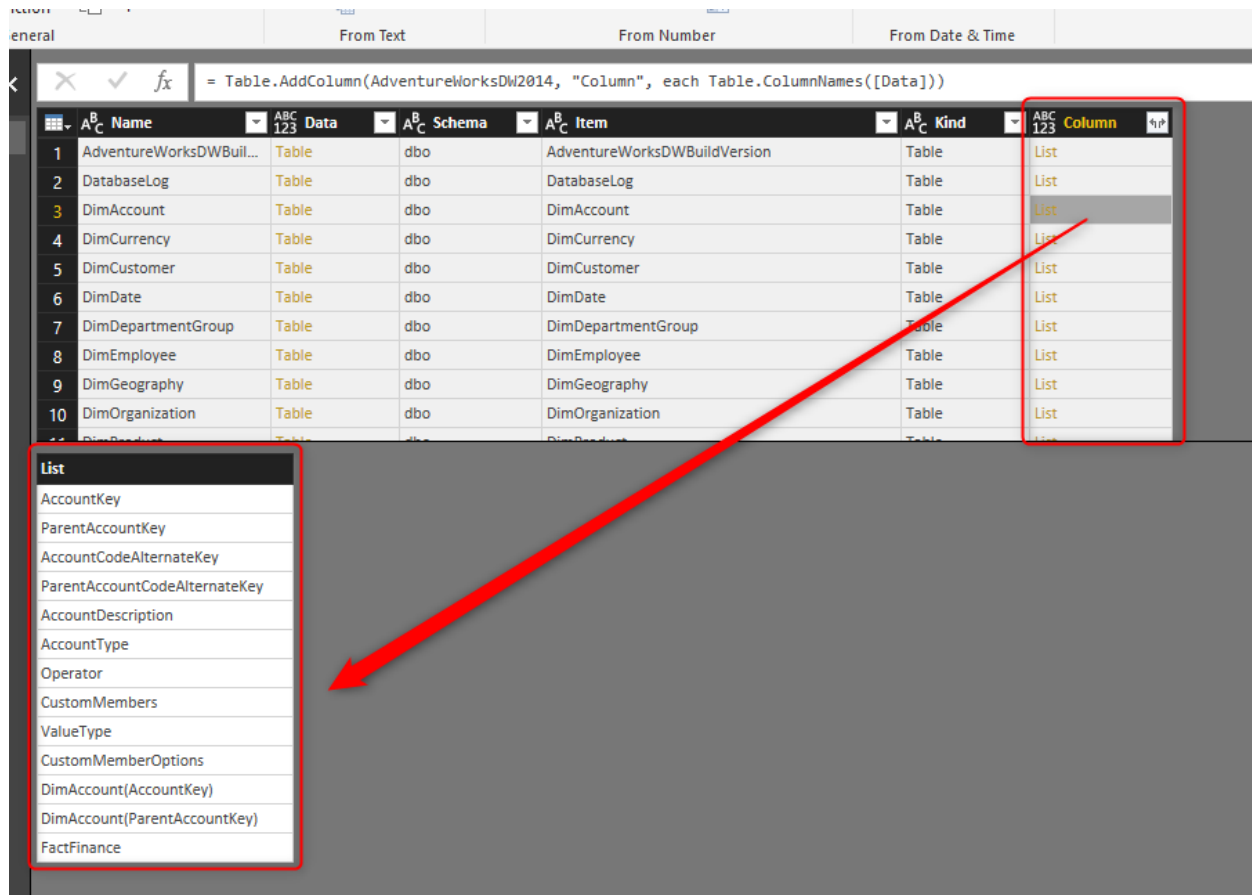
[Learn about Power BI Desktop formulas](#)

✓ No syntax errors have been detected.

OK

Cancel

This action will give you a new column with a list in every cell. This list is the list of column names for every table.



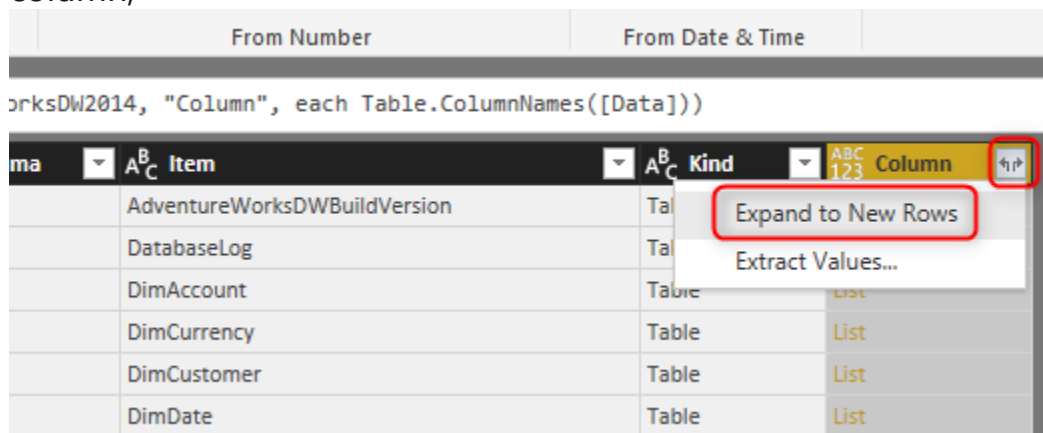
The screenshot shows the Power Query Editor interface. At the top, the formula bar contains the M code: `= Table.AddColumn(AdventureWorksDW2014, "Column", each Table.ColumnNames([Data]))`. Below the formula bar, a table is displayed with the following columns: **Name**, **Data**, **Schema**, **Item**, **Kind**, and **Column**. The **Column** column contains a list of column names. A red box highlights the **Column** column, and a red arrow points to a dropdown menu showing a list of column names.

	Name	Data	Schema	Item	Kind	Column
1	AdventureWorksDWBuildVersion	Table	dbo	AdventureWorksDWBuildVersion	Table	List
2	DatabaseLog	Table	dbo	DatabaseLog	Table	List
3	DimAccount	Table	dbo	DimAccount	Table	List
4	DimCurrency	Table	dbo	DimCurrency	Table	List
5	DimCustomer	Table	dbo	DimCustomer	Table	List
6	DimDate	Table	dbo	DimDate	Table	List
7	DimDepartmentGroup	Table	dbo	DimDepartmentGroup	Table	List
8	DimEmployee	Table	dbo	DimEmployee	Table	List
9	DimGeography	Table	dbo	DimGeography	Table	List
10	DimOrganization	Table	dbo	DimOrganization	Table	List

The dropdown menu shows the following list of column names:

- AccountKey
- ParentAccountKey
- AccountCodeAlternateKey
- ParentAccountCodeAlternateKey
- AccountDescription
- AccountType
- Operator
- CustomMembers
- ValueType
- CustomMemberOptions
- DimAccount(AccountKey)
- DimAccount(ParentAccountKey)
- FactFinance

Now to get the list of all columns in all tables, you need to expand this column;



The screenshot shows the Power Query Editor interface. The formula bar contains the M code: `AdventureWorksDW2014, "Column", each Table.ColumnNames([Data]))`. Below the formula bar, a table is displayed with the following columns: **Item**, **Kind**, and **Column**. The **Column** column contains a list of column names. A red box highlights the **Column** column, and a red arrow points to a dropdown menu showing 'Expand to New Rows' and 'Extract Values...'.

Item	Kind	Column
AdventureWorksDWBuildVersion	Table	List
DatabaseLog	Table	List
DimAccount	Table	List
DimCurrency	Table	List
DimCustomer	Table	List
DimDate	Table	List

After expanding, you will have all columns in all tables listed under this "Column" field. You can convert it to Text data type.

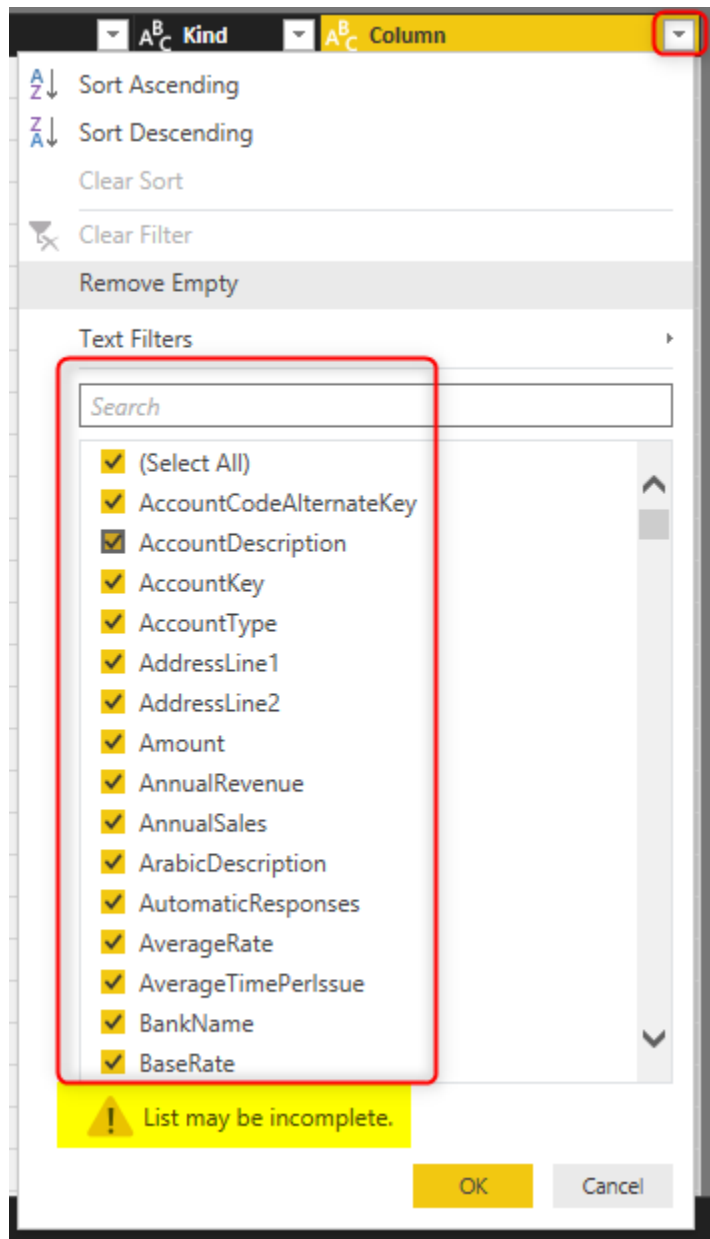
fx = Table.TransformColumnTypes("#Expanded Column",{"Column", type text})

	A <sup>B</sup> <sub>C</sub> Name	ABC 123 Data	A <sup>B</sup> <sub>C</sub> Schema	A <sup>B</sup> <sub>C</sub> Item	A <sup>B</sup> <sub>C</sub> Kind	A <sup>B</sup> <sub>C</sub> Column
1	AdventureWorksDWBuild...	Table	dbo	AdventureWorksDWBuildVersion	Table	DBVersion
2	AdventureWorksDWBuild...	Table	dbo	AdventureWorksDWBuildVersion	Table	VersionDate
3	DatabaseLog	Table	dbo	DatabaseLog	Table	DatabaseLogID
4	DatabaseLog	Table	dbo	DatabaseLog	Table	PostTime
5	DatabaseLog	Table	dbo	DatabaseLog	Table	DatabaseUser
6	DatabaseLog	Table	dbo	DatabaseLog	Table	Event
7	DatabaseLog	Table	dbo	DatabaseLog	Table	Schema
8	DatabaseLog	Table	dbo	DatabaseLog	Table	Object
9	DatabaseLog	Table	dbo	DatabaseLog	Table	TSQL
10	DatabaseLog	Table	dbo	DatabaseLog	Table	XmlEvent
11	DimAccount	Table	dbo	DimAccount	Table	AccountKey
12	DimAccount	Table	dbo	DimAccount	Table	ParentAccountKey
13	DimAccount	Table	dbo	DimAccount	Table	AccountCodeAlternateKey
14	DimAccount	Table	dbo	DimAccount	Table	ParentAccountCodeAlternateKey
15	DimAccount	Table	dbo	DimAccount	Table	AccountDescription
16	DimAccount	Table	dbo	DimAccount	Table	AccountType
17	DimAccount	Table	dbo	DimAccount	Table	Operator
18	DimAccount	Table	dbo	DimAccount	Table	CustomMembers
19	DimAccount	Table	dbo	DimAccount	Table	ValueType
20	DimAccount	Table	dbo	DimAccount	Table	CustomMemberOptions
21	DimAccount	Table	dbo	DimAccount	Table	DimAccount(AccountKey)
22	DimAccount	Table	dbo	DimAccount	Table	DimAccount(ParentAccountKey)
23	DimAccount	Table	dbo	DimAccount	Table	FactFinance
24	DimCurrency	Table	dbo	DimCurrency	Table	CurrencyKey
25	DimCurrency	Table	dbo	DimCurrency	Table	CurrencyAlternateKey
26	DimCurrency	Table	dbo	DimCurrency	Table	CurrencyName

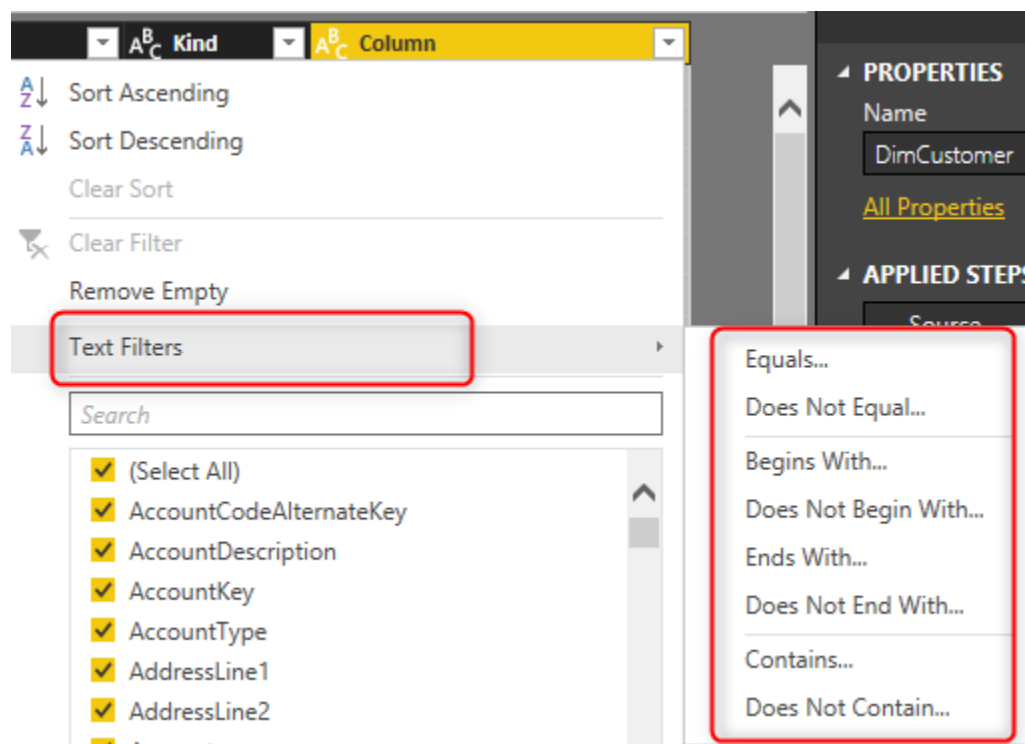
List of all columns in every table or view

## Search through Columns

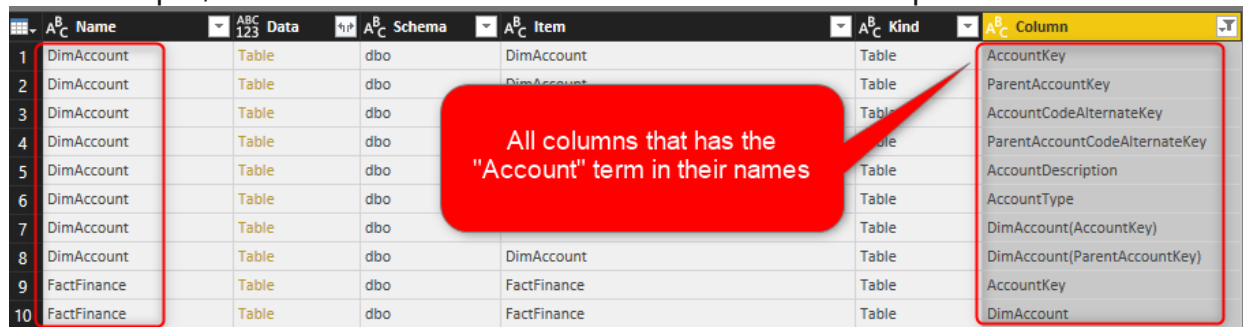
Now that we got the list of all column names, then searching through it is very simple. You can use the basic search, but remember if you have more than 1000 columns in your data source, this will show you a limited list.



The best way to search into this list is using Advanced Search options. There you can choose criteria such as Contains, Equals, Begins With, or Ends With, etc.



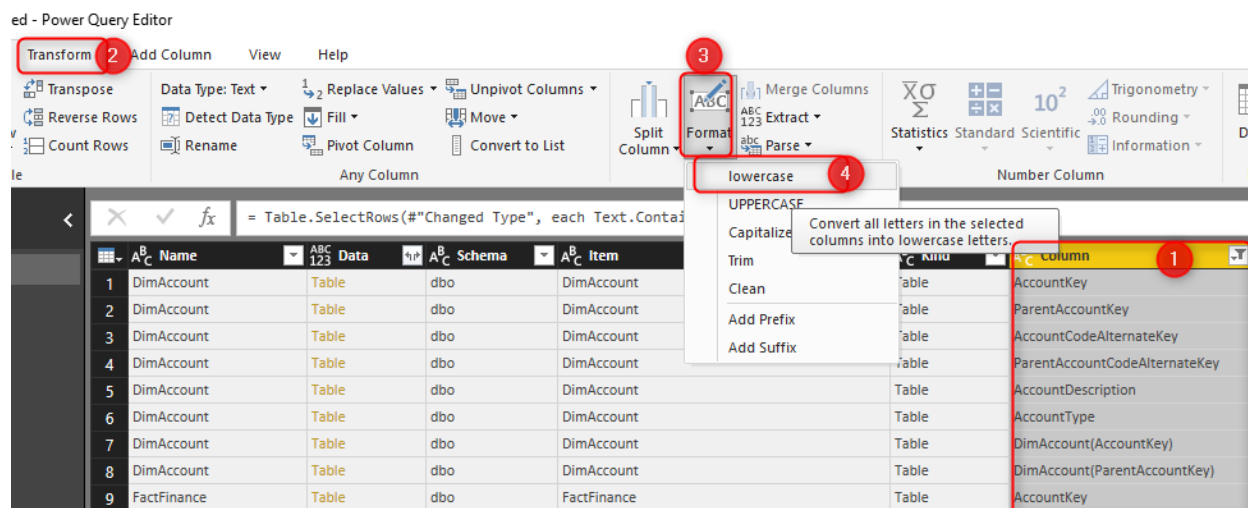
For example, a search for Contains... "Account" will end up with below result:



	Name	Data	Schema	Item	Kind	Column
1	DimAccount	Table	dbo	DimAccount	Table	AccountKey
2	DimAccount	Table	dbo	DimAccount	Table	ParentAccountKey
3	DimAccount	Table	dbo	DimAccount	Table	AccountCodeAlternateKey
4	DimAccount	Table	dbo	DimAccount	Table	ParentAccountCodeAlternateKey
5	DimAccount	Table	dbo	DimAccount	Table	AccountDescription
6	DimAccount	Table	dbo	DimAccount	Table	AccountType
7	DimAccount	Table	dbo	DimAccount	Table	DimAccount(AccountKey)
8	DimAccount	Table	dbo	DimAccount	Table	DimAccount(ParentAccountKey)
9	FactFinance	Table	dbo	FactFinance	Table	AccountKey
10	FactFinance	Table	dbo	FactFinance	Table	DimAccount

## Be Careful of Case Sensitivity

Power Query is a case-sensitive language. There is a difference between "Account" as a text, and "account" as a text. One is using capital A, and the other one; lowercase a. To make sure you can always search for an item, regardless of the case sensitivity of that; you can first convert the column names all to lower case or upper case. To do that, select the "Column" field, and from the Transform tab, select transform to Lower.



now the whole column names list will be lowercase, and you can search through it only with lowercase values;

## Filter Rows

☒ Basic ☐ Advanced

Keep rows where 'Column'

contains  account

☒ And ☐ Or

OK

Cancel

## Summary

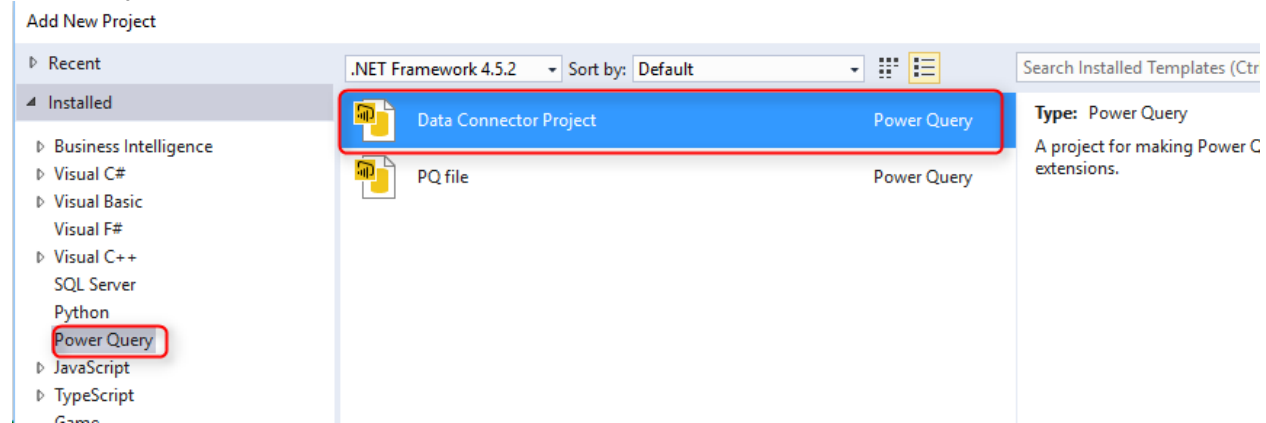
Sometimes simple transformations such as getting the list of columns from a table can be a big help for data exploration. In this post, you've seen how this can be helpful to search through all columns in a database. This approach can be used for any data sources, regardless if they are SQL Server databases or anything else. As long as the data source has a table structure, then you can get the list of all columns from that table. This approach is particularly useful when you connect to databases with thousands of tables, and each table has



hundreds of columns; CRM or Dynamics data sources is one of those examples.

# Power BI Custom Connector: Connect to Any Data Sources. Hello World!

Posted by [Reza Rad](#) on Jul 27, 2017

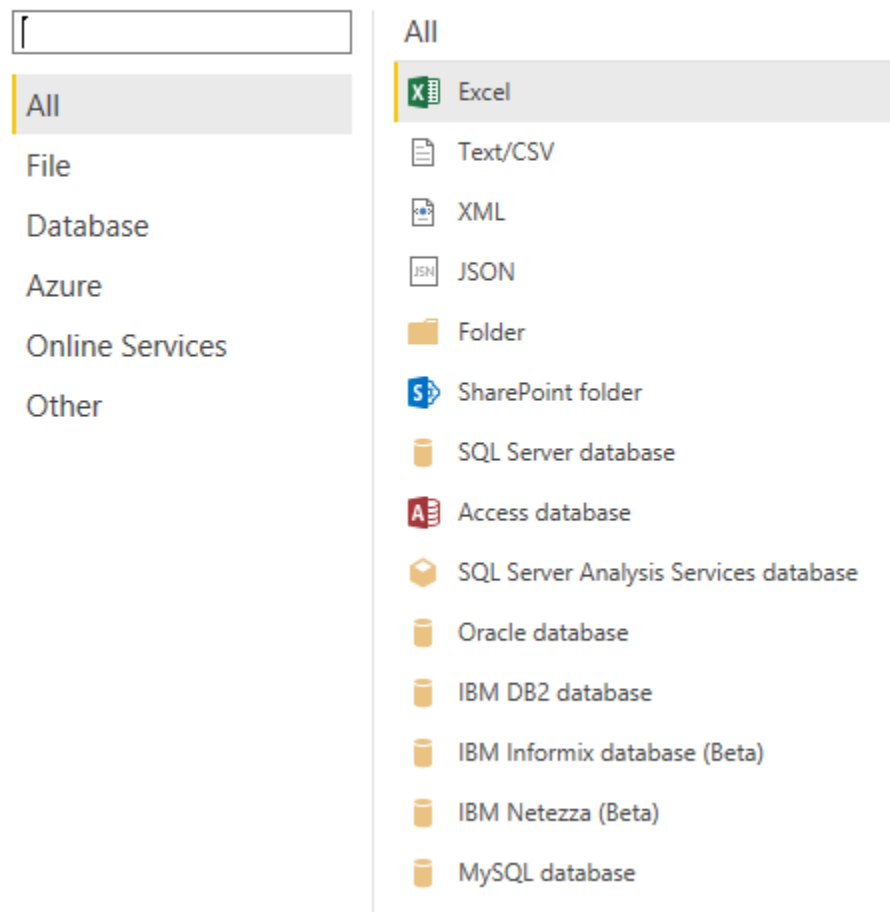


One of the recent features in Power BI Desktop is the ability to create your custom connector and use it when connecting to the data source. This feature looks just one feature, but it opens doors to many possibilities. With this feature, you can write your custom connector to any types of data source that is not already available. Many of you want to connect to some data sources already and waiting for the connector for it. With Customer Connector, you can write your component for it, and use it as many times you want. Custom Connectors should be created in Visual Studio with M script. This post is the first post of blog series about creating custom connectors. In this first post, I'll explain what Custom Connector is and how to create a very basic custom connector. If you want to learn more about Power BI; read [Power BI online book from Rookie to Rock Star](#).

## What is a Custom Connector?

Power BI has a set of existing connectors which you can find them in the Get Data section of Power BI (or Power Query).

## Get Data



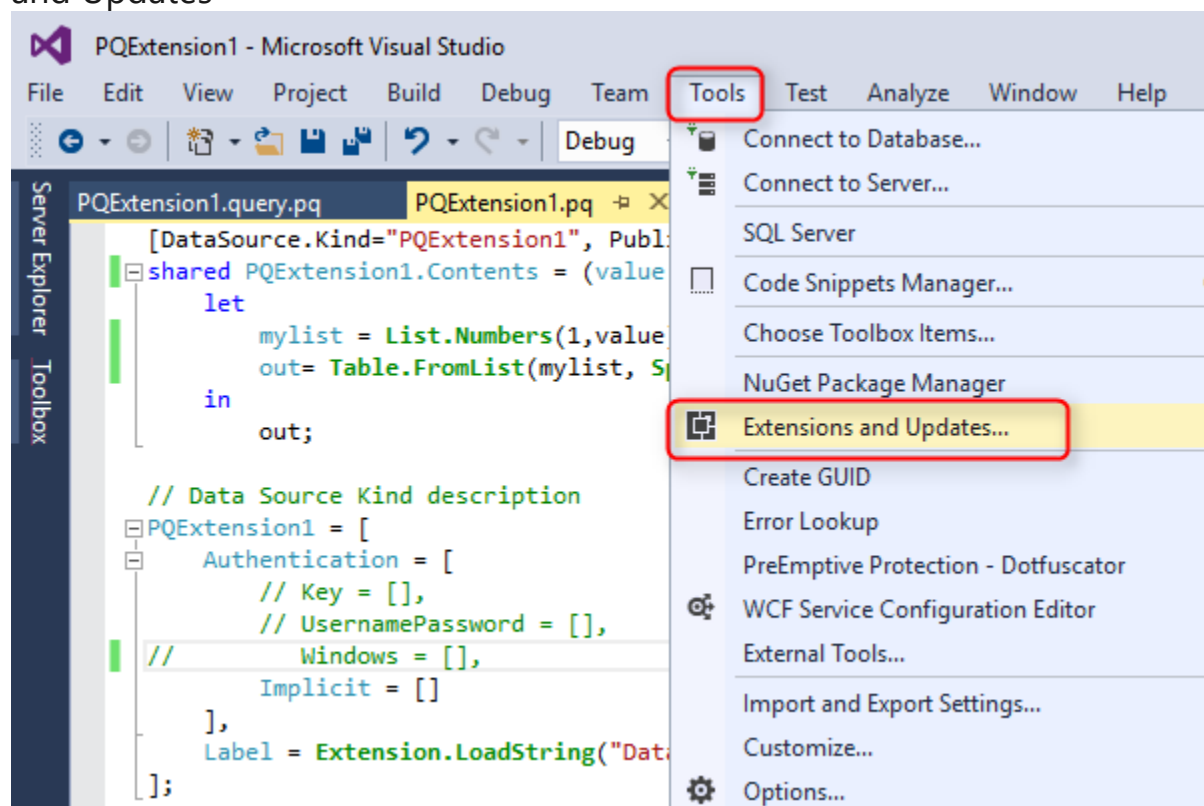
Connectors in this list are connections to the major types of data sources, such as SQL Server, Oracle, some software as a service, such as Salesforce, CRM Online, etc. However, this list is not including all types of data sources. There are always data sources that you want to connect, and is not available on this list. Power BI recently introduced a feature called Custom Data Connector, or simpler Custom Connector. With Custom Connector you can code your connection provider and re-use it multiple times. Your custom connector will appear in the list of getting Data in Power BI, similar to other connectors. Let's have a look at how this custom connector can be built.

## Prerequisites

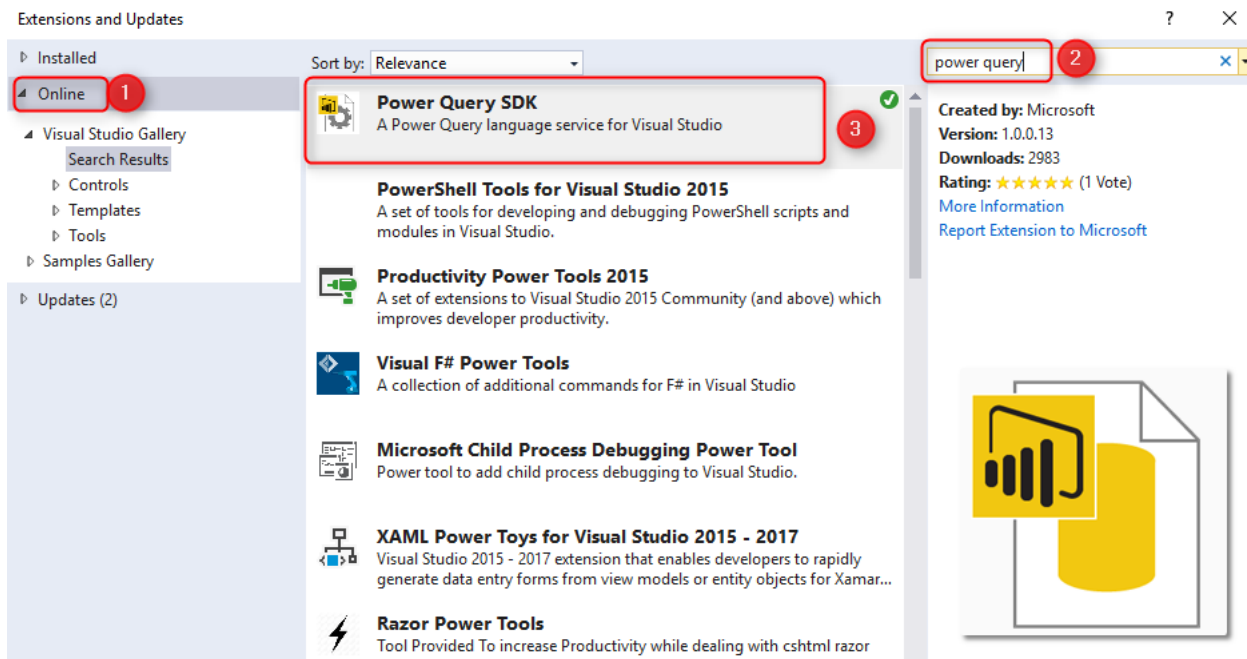
For running examples of this post, you need to have Visual Studio installed. You can download the community edition of that for free.

## Install Power Query SDK

You need to install Power Query SDK on Visual Studio as the very first action. This SDK easily can be found in the Visual Studio Tools menu -> Extensions and Updates

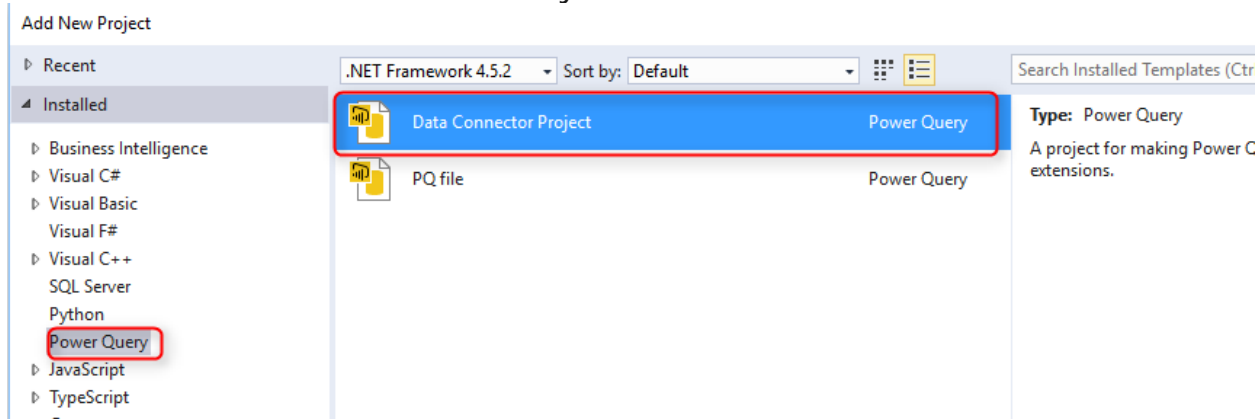


In the list of Online search for Power Query and you will easily find Power Query SDK. simply download and install it.



## Create the First Project

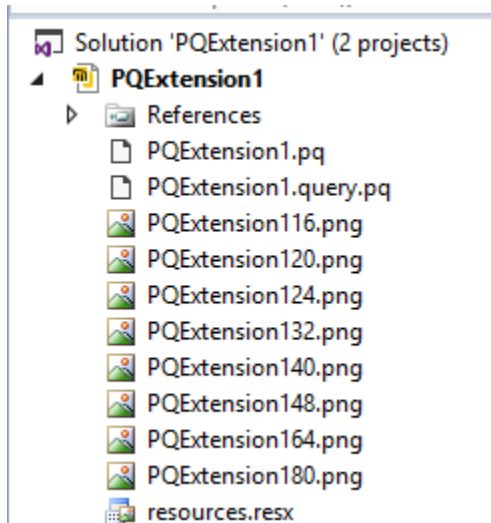
After installing the SDK, then you should be able to create a project of type Data Connector. From List of Installed templates choose Power Query, and then select the Data Connector Project.



I just left the name as default PQExtension1. You can call it anything.

## Project Structure

Data Connector project has a simple structure. It has some files as below;



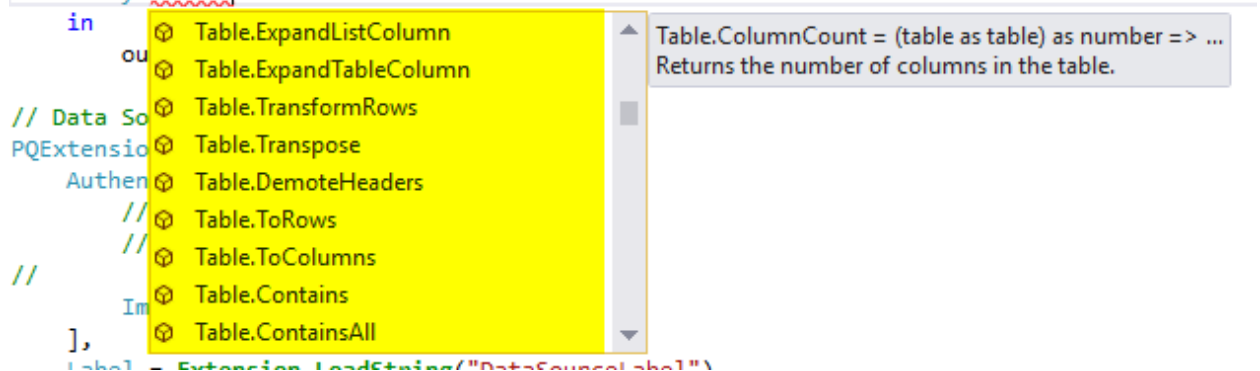
- <connector name>.pq file: The code and metadata of the script and connection to the data source. Majority of the development is happening in this file.
- <connector name>.query.pq: Calling the function in the <connector name>.pq file. Combination of these two files generates the custom connector's script. This is a test file.
- \*.png: Icon files.
- resources.resx: Resource file.

## Coding Language: M

Coding language for the custom connector is M: Power Query Formula Language. If you like to learn about the [basics of M, read my post here](#). Good news is that M has an intellisense here in Visual Studio.

```
[DataSource.Kind="PQExtension1", Publish="PQExtension1.Publish"]
shared PQExtension1.Contents = (value as number) =>
    let
        mylist = List.Numbers(1,value),
        out= Table.FromList(mylist, Splitter.SplitByNothing(), null, null, ExtraValues.Error)
    in
        Table.AddColumn(out, "Index", mylist)
],
Label = Extension.LoadSetting("DataSourceLabel")\

```



For an M script geek like me, this is one of the best news I could get!

## Structure of Query Files

The main query file is <connector name>.pq. This file includes all the code and M script to connect to the data source and fetch the data out in the desired format. The first time you create the project, this file will come with some default sections.

```
1 // This file contains your Data Connector logic
2 section PQExtension2;
3
4 [DataSource.Kind="PQExtension2", Publish="PQExtension2.Publish"]
5 shared PQExtension2.Contents = (optional message as text) =>
6     let
7         _message = if (message <> null) then message else "(no message)",
8         a = "Hello from PQExtension2: " & _message
9     in
10        a;
11
12 // Data Source Kind description
13 PQExtension2 = [
14     Authentication = [
15         // Key = [],
16         // UsernamePassword = [],
```

```

17     // Windows = [],
18     Implicit = []
19 ],
20 Label = Extension.LoadString("DataSourceLabel")
21 ];
22
23 // Data Source UI publishing description
24 PQExtension2.Publish = [
25     Beta = true,
26     Category = "Other",
27     ButtonText = { Extension.LoadString("ButtonTitle"),
28 Extension.LoadString("ButtonHelp") },
29     LearnMoreUrl = "https://powerbi.microsoft.com/",
30     SourceImage = PQExtension2.Icons,
31     SourceTypeImage = PQExtension2.Icons
32 ];
33
34 PQExtension2.Icons = [
35     Icon16 = { Extension.Contents("PQExtension216.png"),
36 Extension.Contents("PQExtension220.png"),
37     Extension.Contents("PQExtension224.png"),
38     Extension.Contents("PQExtension232.png") },
39     Icon32 = { Extension.Contents("PQExtension232.png"),
40 Extension.Contents("PQExtension240.png"),
41     Extension.Contents("PQExtension248.png"),
42     Extension.Contents("PQExtension264.png") }
43 ];

```

Explaining all functions and code above might be a bit out of scope for this introduction post. I will explain these in details in future posts. For now just a very brief explanation;

- Code above contains a function called <connector name>.Contents. This function will return the result set that will be the input for Power BI when connecting to this connector.



- The credentials configuration for this data source will be configured in an Authentication section.
- <connector name>.Publish is for configuring the location and configuration of showing this connector in getting Data section of Power BI.
- <connector name>.Icons are a list of Icons for the connector.

In the <connector name>.query.pq, then you will see only a function call to the same function which is defined above.

```
1 let
2     result = PQExtension2.Contents()
3 in
4     result
```

This file is mainly to perform testing here in Visual Studio.

If you run this project, you would be able to see the result (after setting the authentication of course)

## Write a Sample Function

To make the first connector; I'm not going to explain how to use OData and pass authentication to get data from a web service. That will make things complicated. The very first example I want to show is a simple function that you pass the number to it, and it will give you a table with one single column with values starting from 1, adding one at a time and finishing at that number. It is just a list of numbers. I found it is the easiest way to understand how things work.

In PQExtension1.pq file, change the Contents function as below;

```
1 [DataSource.Kind="PQExtension1", Publish="PQExtension1.Publish"]
2 shared PQExtension1.Contents = (value as number) =>
3     let
4         mylist = List.Numbers(1,value),
5         out= Table.FromList(mylist, Splitter.SplitByNothing(), null, null,
6 ExtraValues.Error)
7     in
```

*out;*

In this function, I have used List.Numbers to generate a list of numbers from 1.

Adding one number at a time, to the specified number.

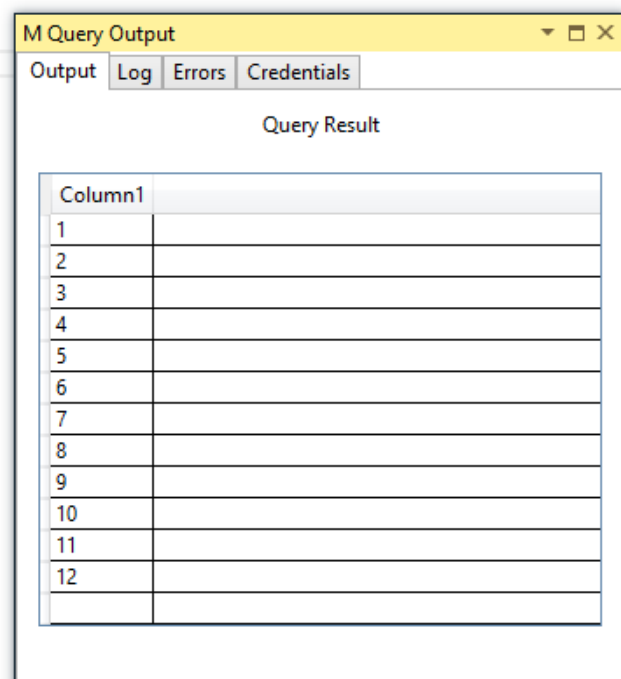
After writing the code above, you can call it from the query.pq file with this test line;

```
1 let
2   result = PQExtension1.Contents(12)
3 in
4   result
```

## Testing the Result

You can test the query in Visual Studio. just run the project, and you will see the result for 12 rows

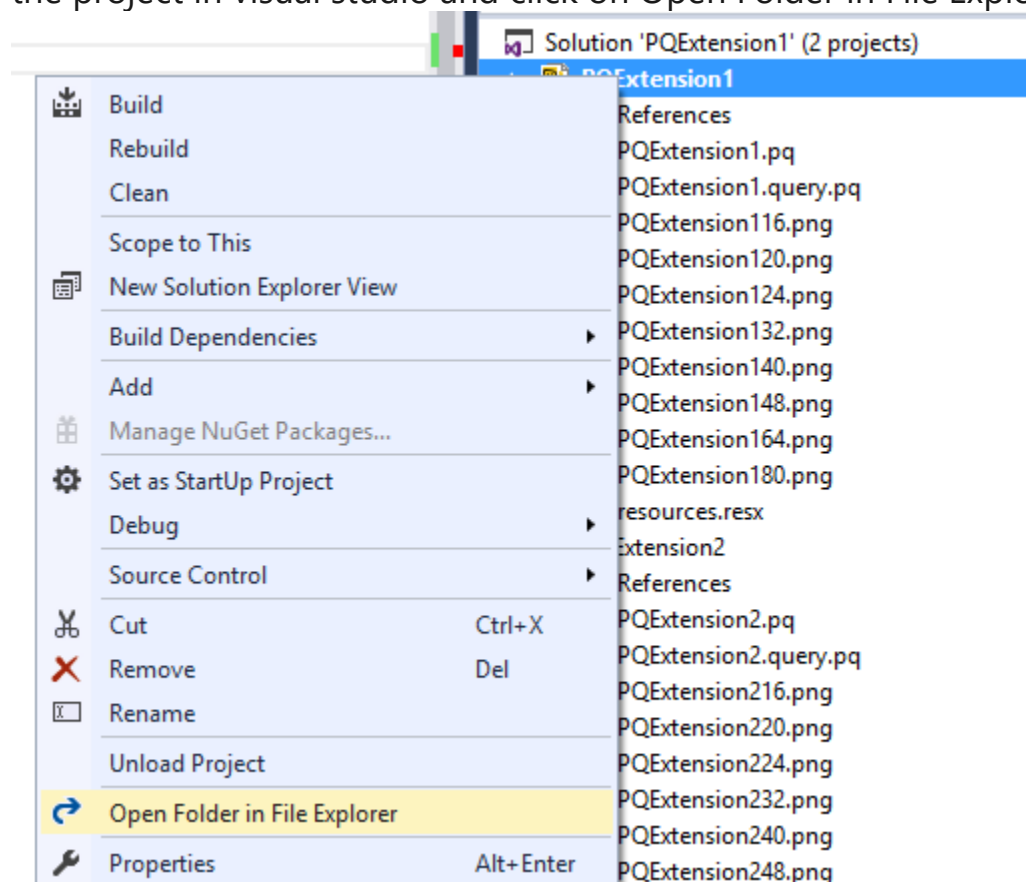
```
// Use this file to write queries to test your data connector
let
|   result = PQExtension1.Contents(12)
in
   result
```



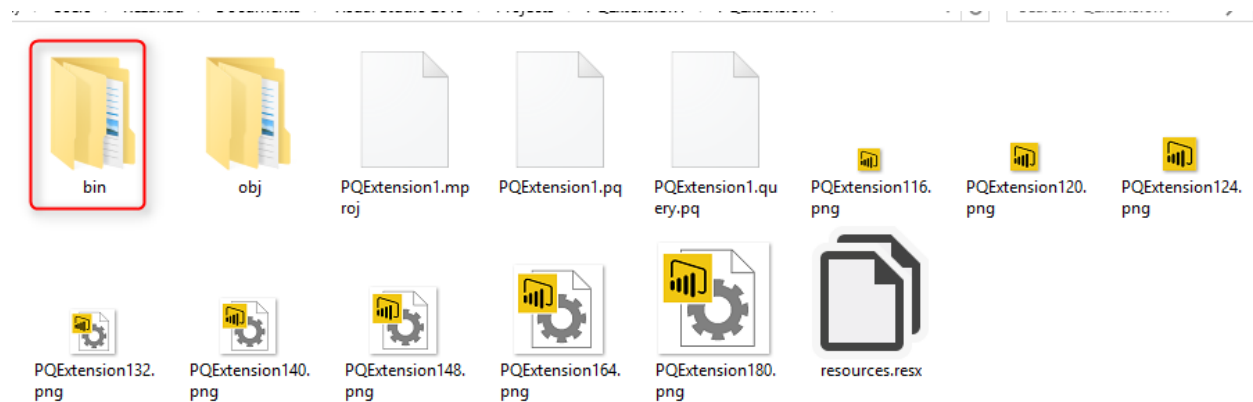
Query Result	
Column1	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

## Publishing Custom Connector

To publish the custom connector, you have to build it first. This will create a \*.mez file in the debug folder of the project. To find that folder, right click on the project in visual studio and click on Open Folder in File Explorer.



In the project, folder go to Bin Folder, and then Debug, find PQExtension1.mez there.



Copy the \*.mez file from here into a folder in My Documents. folder name should be exactly this:

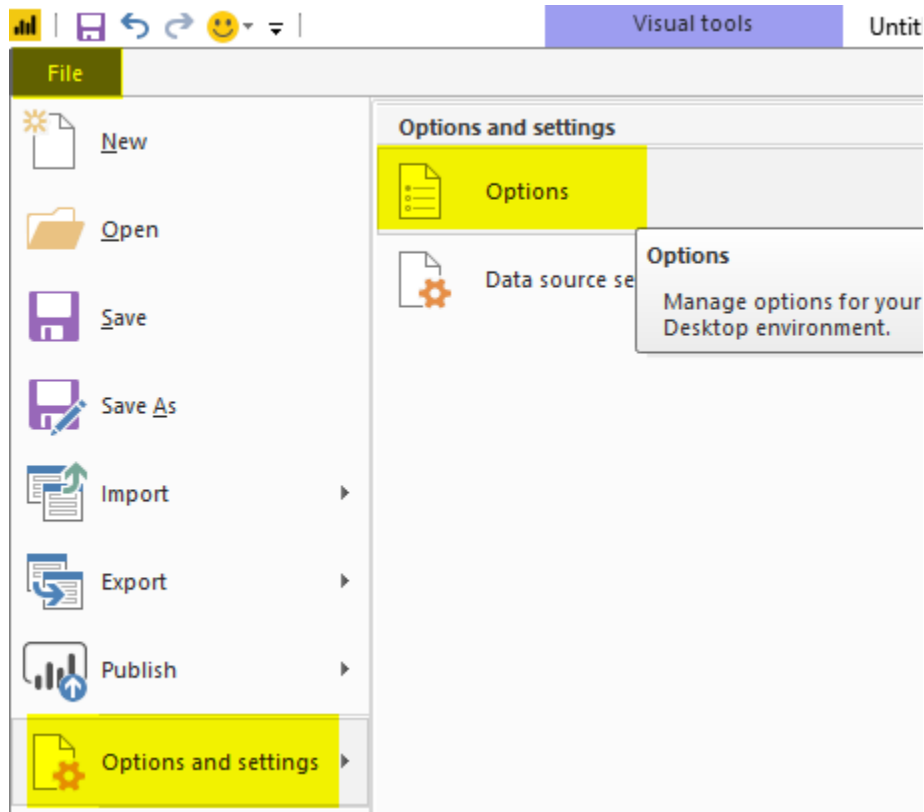
***Microsoft Power BI Desktop\Custom Connectors***

Create the folder above if it doesn't exist.

## Using the Connector

After copying the \*.mez file in the documents' custom connectors folder, then open Power BI Desktop.

At the time of writing this post, custom connectors is still a preview feature. To enable this feature; Go to File, Options



select custom connector preview options. You will need to close Power BI Desktop after this action and re-open it.

## Options

### GLOBAL

Data Load  
Query Editor  
DirectQuery  
R scripting  
Security  
Privacy  
Updates  
Usage Data  
Diagnostics  
**Preview features**  
Auto recovery

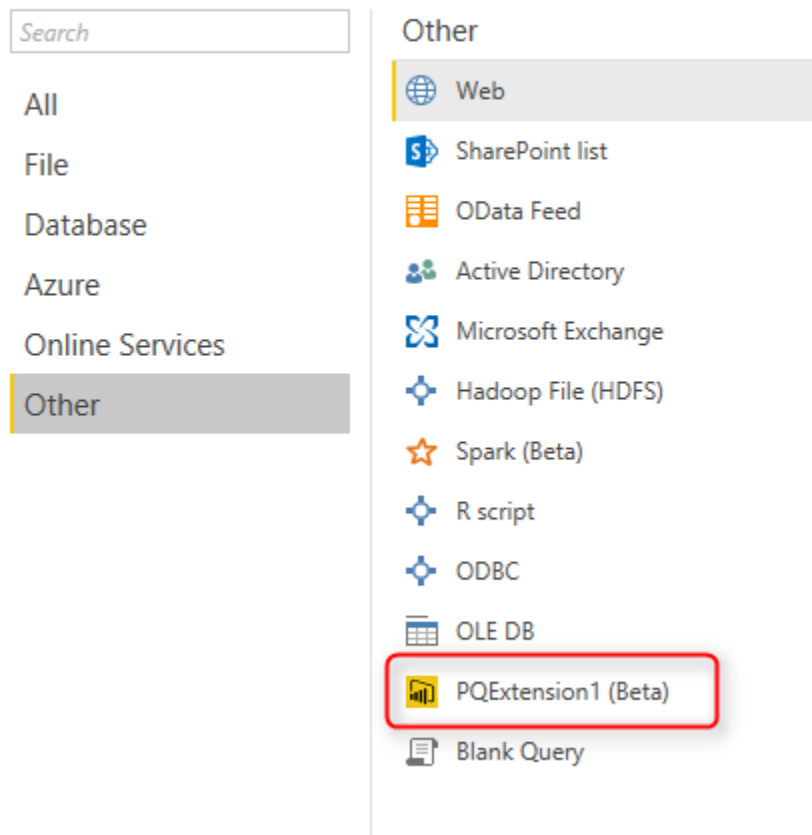
### Preview features

The following features are available for you to try in this release. Preview features might change or be removed in future releases.

- ✓ Shape map visual [Learn more](#)
- ✓ Custom report themes [Learn more](#)
- ✓ Numeric range slicer [Learn more](#)
- ✓ Spanish language support for Power BI Q&A [Learn more](#)
- ✓ Power BI Service Live Connection [Learn more](#)
- ✓ Quick measures [Learn more](#)
- ✓ Custom data connectors [Learn more](#)

After re-opening Power BI Desktop you should see your new connector under Other section (or by searching it);

## Get Data



With selecting this connector, you will be asked for an input value (remember, this is the numerical value input for the function that generates the list);

From PQExtension1.Contents

value  
10

OK

Cancel

Don't worry about "From PQExtension1.Contents" now. In future posts, I'll explain how we can customize all of this stuff. After entering a value, you will see the result immediately;

10

☐ X

Column1
1
2
3
4
5
6
7
8
9
10

Load

Edit

Cancel

## Summary

The custom connector that we have created in this post doesn't connect to OData and authenticate through a process to bring some interesting data for you. However, this example was very basic one to show you the main components of a Custom Connector. In the next blog posts in this series, I will write about how to custom each part, and we will have much more complex examples of that. If you are interested in learning more about this the [documentation of Power BI team in GitHub](#) is already a great resource for it. In my opinion after the custom visuals which was a great milestone for Power BI visualization; Custom Connector is a big milestone for Power Query. You can now write your connector and connect to the world of data,

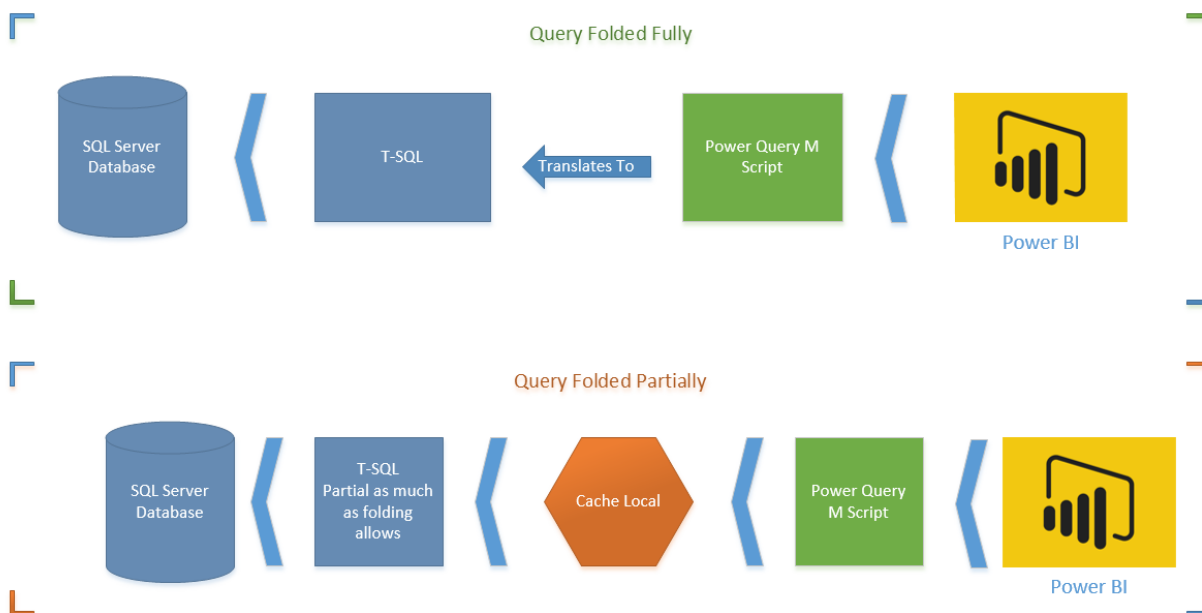
anywhere, anything. Are you excited about this feature; please let me know your opinion in the comments section below.



## Part VII: Performance Tuning

# Not Folding; the Black Hole of Power Query Performance

Posted by [Reza Rad](#) on Nov 15, 2016



Have you ever thought does your Power Query Transformations happens in the data source side (server side), or on the local memory (client side)? When you use a relational data source, a query can be run on the data source, but it depends on transformations. Some transformations can be translated to query language of the data source, some not. As an example; recently (I believe from last few releases) Power BI Desktop added a feature called Merge Columns. Merge Columns concatenate columns to each other to either create a new column or replace them with the new concatenated result. Previously you could do the concatenation with adding concatenation simply with **&** character, what you've done was adding a new custom column, and writing M expression to concatenate columns. Now with the new Merge Column, this is much easier, you select columns and apply Merge Columns. This easiness does come with a price, a high price I'd say, the price of reducing

the performance of Power Query and as a result Power BI! Merge Columns doesn't support query folding, and it means it will affect performance badly. In this post, I'll show you how this cause the performance issue, and how it can be solved. Note that Merge Columns is an example here, this situation might happen with some other transformations as well. If you like to learn more about Power BI; read the [Power BI online book from Rookie to Rock Star](#).

## Query Folding

I can't start talking about the issue without explaining what Query Folding is, so let's start with that. Query Folding means translating Power Query (M) transformations into native query language of the data source (for example T-SQL). In other words; when you run Power Query script on top of a SQL Server database, query folding will translate the M script into T-SQL statements, and fetch the final results.

Here is an example of M Script:

```
let
    Source = Sql.Databases("."),
1   AdventureWorks2012 = Source{Name="AdventureWorks2012"}[Data],
2   Sales_SalesOrderHeader =
3   AdventureWorks2012{[Schema="Sales",Item="SalesOrderHeader"]}[Data],
4   #"Added Conditional Column" =
5   Table.AddColumn(Sales_SalesOrderHeader, "Custom", each if [SubTotal]
6   >= 100 then "0" else "1" )
7 in
    #"Added Conditional Column"
```

And here is the folded version of that translated to native T-SQL query:

```
1 select [].[SalesOrderID] as [SalesOrderID],
2    [].[RevisionNumber] as [RevisionNumber],
3    [].[OrderDate] as [OrderDate],
4    [].[DueDate] as [DueDate],
5    [].[ShipDate] as [ShipDate],
```

```
6  [].[Status] as [Status],
7  [].[OnlineOrderFlag] as [OnlineOrderFlag],
8  [].[SalesOrderNumber] as [SalesOrderNumber],
9  [].[PurchaseOrderNumber] as [PurchaseOrderNumber],
10 [].[AccountNumber] as [AccountNumber],
11 [].[CustomerID] as [CustomerID],
12 [].[SalesPersonID] as [SalesPersonID],
13 [].[TerritoryID] as [TerritoryID],
14 [].[BillToAddressID] as [BillToAddressID],
15 [].[ShipToAddressID] as [ShipToAddressID],
16 [].[ShipMethodID] as [ShipMethodID],
17 [].[CreditCardID] as [CreditCardID],
18 [].[CreditCardApprovalCode] as [CreditCardApprovalCode],
19 [].[CurrencyRateID] as [CurrencyRateID],
20 [].[SubTotal] as [SubTotal],
21 [].[TaxAmt] as [TaxAmt],
22 [].[Freight] as [Freight],
23 [].[TotalDue] as [TotalDue],
24 [].[Comment] as [Comment],
25 [].[rowguid] as [rowguid],
26 [].[ModifiedDate] as [ModifiedDate],
27 case
28     when [].[SubTotal] >= 100
29     then '0'
30     else '1'
31 end as [Custom]
32 from
33 (
34     select [Table].[SalesOrderID] as [SalesOrderID],
35           [Table].[RevisionNumber] as [RevisionNumber],
36           [Table].[OrderDate] as [OrderDate],
37           [Table].[DueDate] as [DueDate],
38           [Table].[ShipDate] as [ShipDate],
39           [Table].[Status] as [Status],
```

```

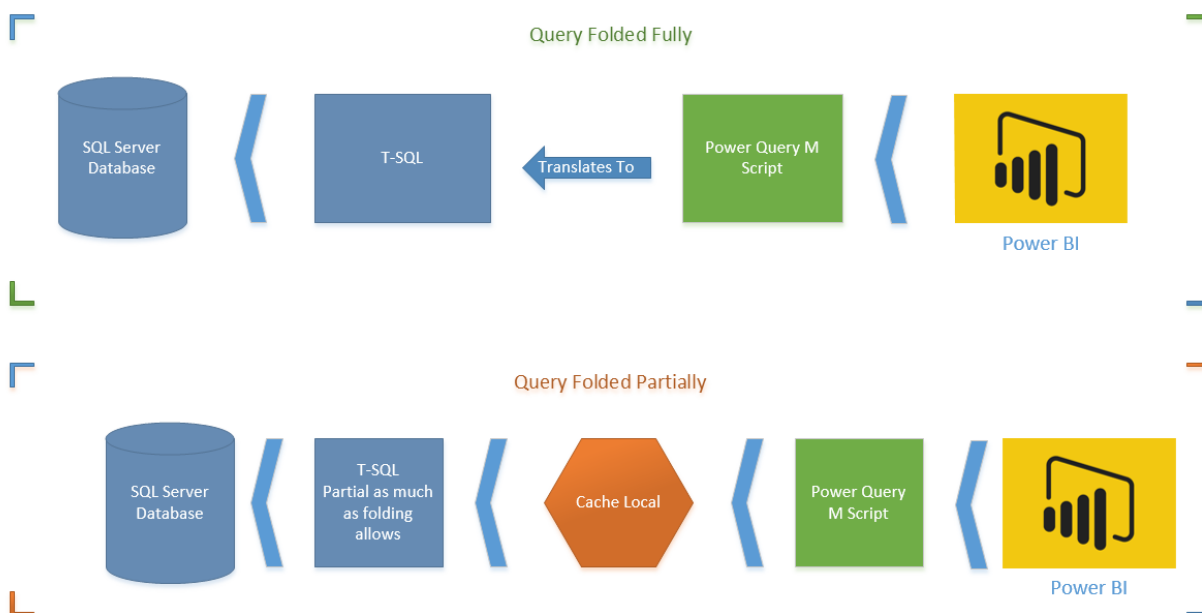
40     [$Table].[OnlineOrderFlag] as [OnlineOrderFlag],
41     [$Table].[SalesOrderNumber] as [SalesOrderNumber],
42     [$Table].[PurchaseOrderNumber] as [PurchaseOrderNumber],
43     [$Table].[AccountNumber] as [AccountNumber],
44     [$Table].[CustomerID] as [CustomerID],
45     [$Table].[SalesPersonID] as [SalesPersonID],
46     [$Table].[TerritoryID] as [TerritoryID],
47     [$Table].[BillToAddressID] as [BillToAddressID],
48     [$Table].[ShipToAddressID] as [ShipToAddressID],
49     [$Table].[ShipMethodID] as [ShipMethodID],
50     [$Table].[CreditCardID] as [CreditCardID],
51     [$Table].[CreditCardApprovalCode] as [CreditCardApprovalCode],
52     [$Table].[CurrencyRateID] as [CurrencyRateID],
53     [$Table].[SubTotal] as [SubTotal],
54     [$Table].[TaxAmt] as [TaxAmt],
55     [$Table].[Freight] as [Freight],
56     [$Table].[TotalDue] as [TotalDue],
57     [$Table].[Comment] as [Comment],
58     convert(nvarchar(max), [$Table].[rowguid]) as [rowguid],
59     [$Table].[ModifiedDate] as [ModifiedDate]
60 from [Sales].[SalesOrderHeader] as [$Table]
61 ) as [ ]

```

You can see as an example how the conditional column script in M translated to Case statement in T-SQL.

## Is Query Folding Good or Bad?

Good obviously. Why? Because performance is much higher to run transformations on billions of records in the data source, rather than bringing millions of records into the cache and applying some transformations on it.

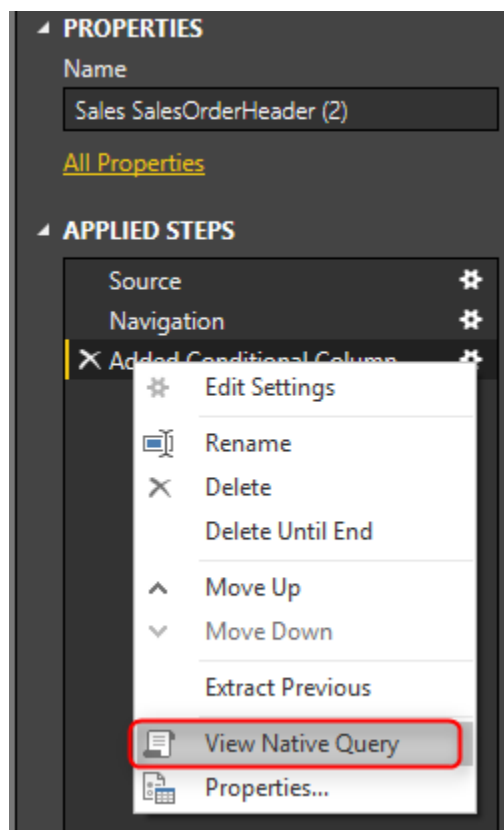


The first diagram shows an M script fully folded (translated to T-SQL). This is the best situation. Server-side operation applies all transformations on the data set and returns only desired result set.

The second diagram shows Query Folding partially supported (only up to specific step). In this case, T-SQL brings the data before that step. And data will be loaded in the local cache, and rest of transformations happens on M engine side. You have to avoid this option as much as possible.

## Can I see the Native Query?

The question that might come into your mind right now is that; Can I see the Native Query that M script translates to it? The answer is Yes. If Query Folding is supported on a step, you can right-click on that step and click on View Native Query.



## So Why Not Query Folding?

Query Folding is enabled by default. However, in some cases, it is not supported. For example, if you are doing some transformations on a SQL Server table in Power Query and then join it with a web query, the query folding stops from the time you bring the external data source. That means transformations will happen on the data of the SQL Server table. Then before joining to web query, it will be fetched into the cache, and then the rest of steps happens by M engine. You would need to bring data from different data sources in Power BI, and this is the ability that Power Query gives to you. So sometimes you have to step beyond query folding, and there might be no better way of doing that.

There are also some Transformations in Power Query that Query Folding doesn't support them. Example? Merge Columns! Fortunately, there are workarounds for this situation. Let's dig into this more in details.

## Example: Merge Columns

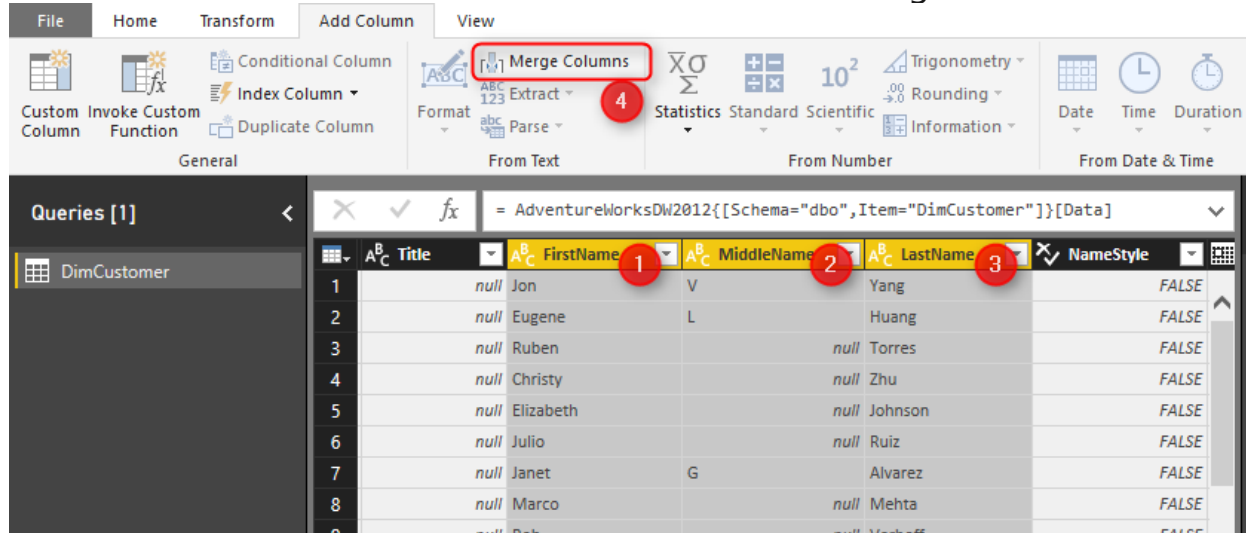
Merge Columns concatenates columns to each other in the order of selection of columns. You can also specify the delimiter character(s). So simply if you want to create a full name from the first name and last name, you can select them in the right order, and from either Transform tab or Add Column tab choose Merge Columns. Let's see this through an example;

### Prerequisite for running the example

You need to have AdventureWorksDW database installed on SQL Server. Or alternatively, you can use any table in SQL Server that has two string columns which can be concatenated.

### Merge Column Transformation

Create a new Power BI file in Power BI Desktop, and Get Data from SQL Server with Import Data Mode. Get Data from DimCustomer table only, and click on Edit. When in Query Editor. Select First Name, Middle Name, and Last Name in the correct order, and then from Add Column Select Merge Columns.



The screenshot shows the Power BI Query Editor interface. The ribbon at the top includes 'File', 'Home', 'Transform', 'Add Column', and 'View'. The 'Add Column' tab is active, and the 'Merge Columns' button is highlighted with a red circle labeled '4'. Below the ribbon, the data view shows a table with columns: 'Title', 'FirstName', 'MiddleName', 'LastName', and 'NameStyle'. The columns 'FirstName', 'MiddleName', and 'LastName' are selected in order, indicated by red circles labeled '1', '2', and '3' respectively. The data view shows a list of customer records with their first, middle, and last names.

	Title	FirstName	MiddleName	LastName	NameStyle
1	null	Jon	V	Yang	FALSE
2	null	Eugene	L	Huang	FALSE
3	null	Ruben		Torres	FALSE
4	null	Christy		Zhu	FALSE
5	null	Elizabeth		Johnson	FALSE
6	null	Julio		Ruiz	FALSE
7	null	Janet	G	Alvarez	FALSE
8	null	Marco		Mehta	FALSE
9	null	Rob		Verhoff	FALSE



In the Merge Columns Specify the Separator to be space, and name the new column to be Full Name.

## Merge Columns

Choose how to merge the selected columns.

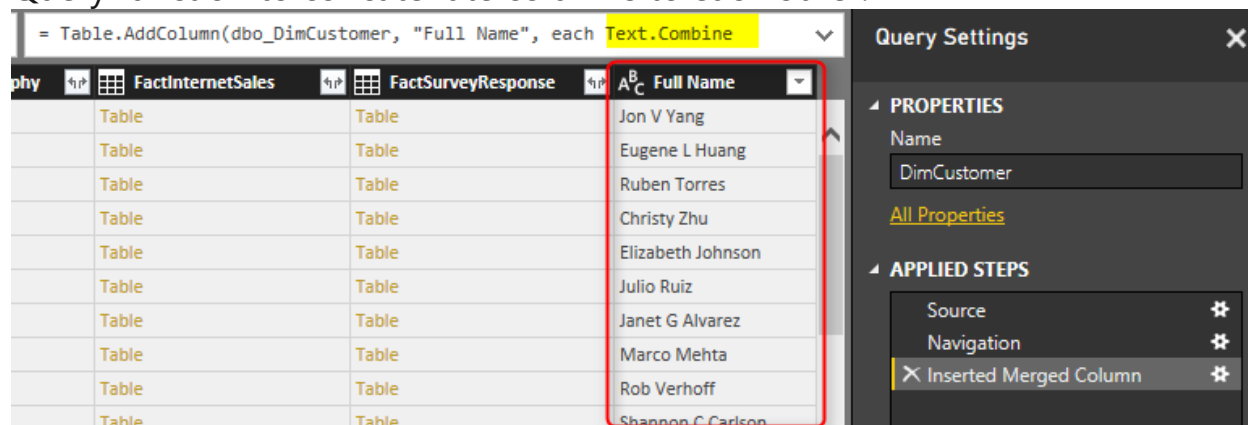
Separator

Space

New column name (optional)

Full Name

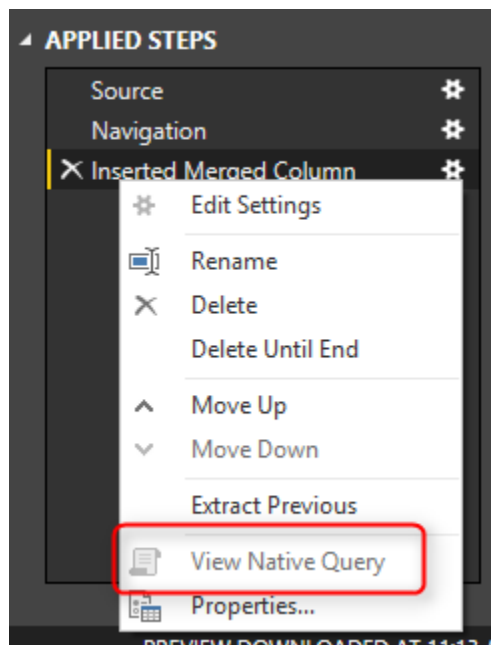
You will see the new column generates simply and adds to the end of all columns. You also may notice that Merge Columns uses Text.Combine Power Query function to concatenate columns to each other.



The screenshot shows the Power BI interface. At the top, the formula bar displays the M code: `= Table.AddColumn(dbo_DimCustomer, "Full Name", each Text.Combine`. Below the formula bar, a table with three columns is visible: `FactInternetSales`, `FactSurveyResponse`, and `Full Name`. The `Full Name` column contains concatenated names from the other two tables. A red box highlights the `Full Name` column header and its data. To the right, the 'Query Settings' pane is open, showing the 'APPLIED STEPS' section with 'Inserted Merged Column' selected.

FactInternetSales	FactSurveyResponse	Full Name
Table	Table	Jon V Yang
Table	Table	Eugene L Huang
Table	Table	Ruben Torres
Table	Table	Christy Zhu
Table	Table	Elizabeth Johnson
Table	Table	Julio Ruiz
Table	Table	Janet G Alvarez
Table	Table	Marco Mehta
Table	Table	Rob Verhoff
Table	Table	Shannon C Carlson

Now to see the problem with Query Folding, right click on Inserted Merge Column step in Applied Steps section. You will see that View Native Query is disabled.



Rule of thumb is that: When View Native Query is not enabled, that step won't be folded! that means the data will be loaded into cache up the step before this step. and then rest of the operation will happen locally. To understand how it works, right click on the step before which was Navigation. You will see the View Native Query. Click on that, and you can see the T-SQL query for that which is as below;

```

1 select [$Table].[CustomerKey] as [CustomerKey],
2    [$Table].[GeographyKey] as [GeographyKey],
3    [$Table].[CustomerAlternateKey] as [CustomerAlternateKey],
4    [$Table].[Title] as [Title],
5    [$Table].[FirstName] as [FirstName],
6    [$Table].[MiddleName] as [MiddleName],
7    [$Table].[LastName] as [LastName],
8    [$Table].[NameStyle] as [NameStyle],
9    [$Table].[BirthDate] as [BirthDate],
10   [$Table].[MaritalStatus] as [MaritalStatus],
11   [$Table].[Suffix] as [Suffix],
12   [$Table].[Gender] as [Gender],
13   [$Table].[EmailAddress] as [EmailAddress],
14   [$Table].[YearlyIncome] as [YearlyIncome],
15   [$Table].[TotalChildren] as [TotalChildren],

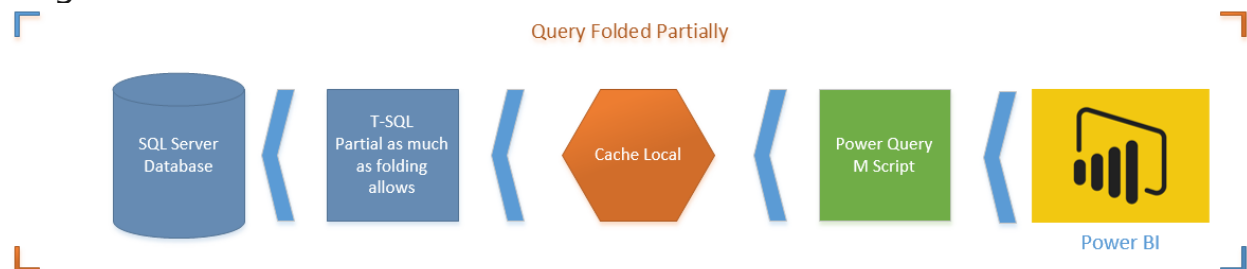
```

```

16  [$Table].[NumberChildrenAtHome] as [NumberChildrenAtHome],
17  [$Table].[EnglishEducation] as [EnglishEducation],
18  [$Table].[SpanishEducation] as [SpanishEducation],
19  [$Table].[FrenchEducation] as [FrenchEducation],
20  [$Table].[EnglishOccupation] as [EnglishOccupation],
21  [$Table].[SpanishOccupation] as [SpanishOccupation],
22  [$Table].[FrenchOccupation] as [FrenchOccupation],
23  [$Table].[HouseOwnerFlag] as [HouseOwnerFlag],
24  [$Table].[NumberCarsOwned] as [NumberCarsOwned],
25  [$Table].[AddressLine1] as [AddressLine1],
26  [$Table].[AddressLine2] as [AddressLine2],
27  [$Table].[Phone] as [Phone],
28  [$Table].[DateFirstPurchase] as [DateFirstPurchase],
29  [$Table].[CommuteDistance] as [CommuteDistance],
30  [$Table].[FullName] as [FullName]
31 from [dbo].[DimCustomer] as [$Table]

```

You can see that this is a simple query from DimCustomer Table. What will happen here in this scenario is that Power Query cannot translate Text.Combine to T-SQL Query. So data up to step before will be loaded into the cache. It means the query for a step before (which is above query) will run on the database server, the result will come to the cache, and then Text.Combine will happen on the local memory in the cache. Here is a diagram of how it works;

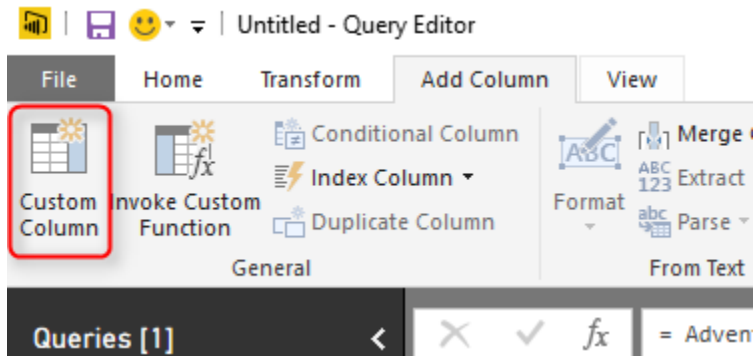


In this example, the data set is so small, but if data set is big, then not folding cause performance issues. It is taking much longer to load the whole data set

into the cache and then apply transformations, rather than doing transformations in the data source, and just loading the result into Power BI.

### **Solution: Simple Concatenate with Add Column**

Now remove the step for Inserted Merged Column, and go to Add Column Tab, and select Custom Column



In the Add Custom Column write below expression to generate Full Name;

1 =

2 [FirstName]

3 &

4 " "

5 &

6 (if [MiddleName]=null then "" else [MiddleName])

7 &

8 " "

9 &[LastName]

## Add Custom Column

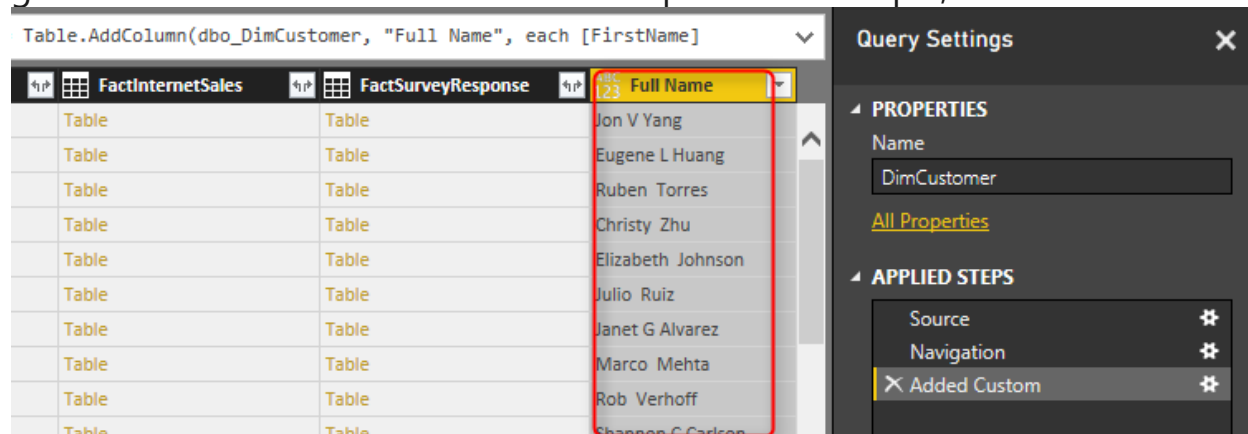
New column name

Full Name

Custom column formula:

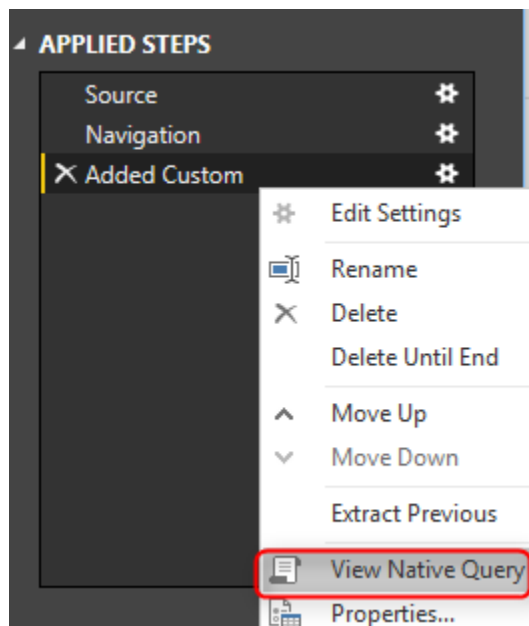
```
=
[FirstName]
&
" "
&
(if [MiddleName]=null then "" else [MiddleName])
&
" "
&[LastName]
```

This expression used the concatenation character which is **&**. and also checked if Middle Name is null or not. Result in Power Query side is the same, and it generates the Full Name column like the previous example;



The screenshot displays the Power Query Editor interface. At the top, the formula bar shows the code: `Table.AddColumn(dbo_DimCustomer, "Full Name", each [FirstName]`. Below this, a table with three columns is visible: `FactInternetSales`, `FactSurveyResponse`, and `Full Name`. The `Full Name` column contains concatenated values like "Jon V Yang", "Eugene L Huang", etc. A red box highlights the `Full Name` column header. On the right, the 'Query Settings' pane is open, showing the 'APPLIED STEPS' list with 'Added Custom' selected.

However it is different for Query Folding. Right Click on the Added Custom step, and you will see the Native Query this time.



Query is simply the same with a new concatenated column added.

```

1 select [].[CustomerKey] as [CustomerKey],
2    [].[GeographyKey] as [GeographyKey],
3    [].[CustomerAlternateKey] as [CustomerAlternateKey],
4    [].[Title] as [Title],
5    [].[FirstName] as [FirstName],
6    [].[MiddleName] as [MiddleName],
7    [].[LastName] as [LastName],
8    [].[NameStyle] as [NameStyle],
9    [].[BirthDate] as [BirthDate],
10   [].[MaritalStatus] as [MaritalStatus],
11   [].[Suffix] as [Suffix],
12   [].[Gender] as [Gender],
13   [].[EmailAddress] as [EmailAddress],
14   [].[YearlyIncome] as [YearlyIncome],
15   [].[TotalChildren] as [TotalChildren],
16   [].[NumberChildrenAtHome] as [NumberChildrenAtHome],
17   [].[EnglishEducation] as [EnglishEducation],
18   [].[SpanishEducation] as [SpanishEducation],
19   [].[FrenchEducation] as [FrenchEducation],
20   [].[EnglishOccupation] as [EnglishOccupation],

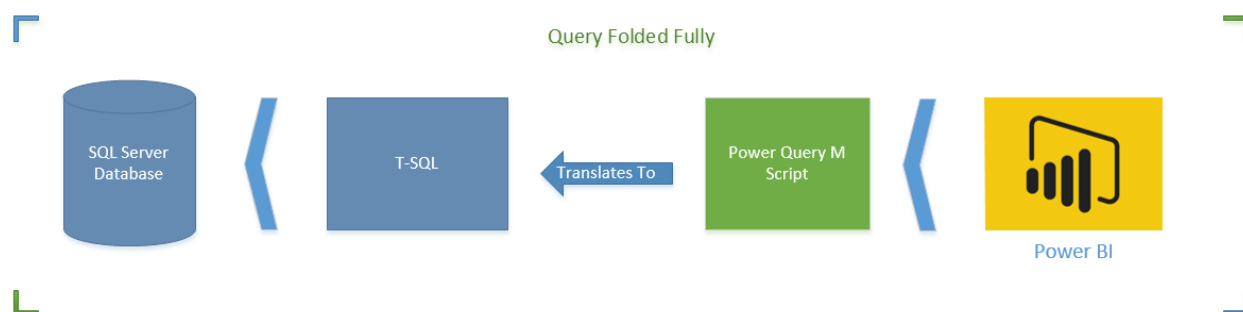
```

```

21  [].[SpanishOccupation] as [SpanishOccupation],
22  [].[FrenchOccupation] as [FrenchOccupation],
23  [].[HouseOwnerFlag] as [HouseOwnerFlag],
24  [].[NumberCarsOwned] as [NumberCarsOwned],
25  [].[AddressLine1] as [AddressLine1],
26  [].[AddressLine2] as [AddressLine2],
27  [].[Phone] as [Phone],
28  [].[DateFirstPurchase] as [DateFirstPurchase],
29  [].[CommuteDistance] as [CommuteDistance],
30  [].[FullName] as [FullName],
31  ((([].[FirstName] + ' ') + (case
32    when [].[MiddleName] is null
33    then ''
34    else [].[MiddleName]
35  end)) + ' ') + [].[LastName] as [Full Name]
36 from [dbo].[DimCustomer] as [.]

```

This time there won't be an intermediate cache. Transformation happens in the data source with the T-SQL query, and the result will be loaded into Power BI.



## How Do I know Which Transformations Folds?

It is a great question. It is important to understand which step/transformation folds and which doesn't. To understand that simply right click on every step and see if the View Native Query is enabled or not. If it is enabled, Query Folding is supported for that step, otherwise not. Also Note that Query Folding is not supported for data sources such as web query, or CSV or things

like that. Query Folding at the moment is only supported for data stores that support a native query language. For Web, Folder, CSV... there is no native query language, so you don't need to worry about Query Folding.

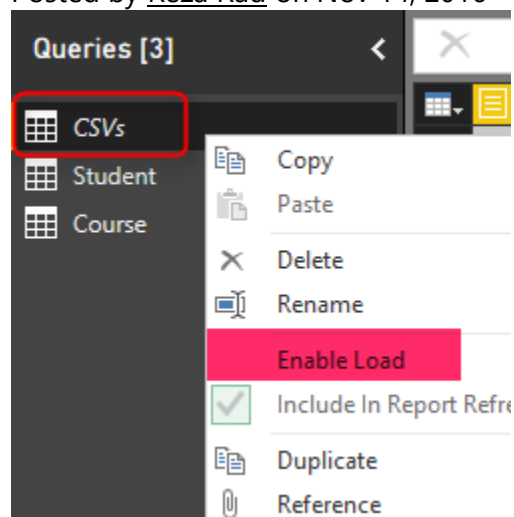
\*Important Note: At the time of writing this post Merge Columns doesn't support Query Folding. I have reported this to Power Query Team, and they are working on it, to solve the issue. The Merge Columns is very likely to support query folding very soon as the result of bug fix. However, there are always some other transformations that don't support query folding. This post is written to give you an understanding of what kind of issue might happen, and how to resolve it.

My advice to you as a performance best practice is that when working with a relational data source (such as SQL Server). Always check the query folding. Sometimes it is not supported. So use another approach for the transformation. Don't fall into the black hole of not folding. Otherwise your query might take ages to run.



# Performance Tip for Power BI; Enable Load Sucks Memory Up

Posted by [Reza Rad](#) on Nov 14, 2016



In the area of performance tuning a Power BI model, many things have to be considered, most of them around the consumption of the CPU and RAM. One of the most basic but important consideration is minimizing the usage of memory. By default, all queries from Query Editor will be loaded into the memory of Power BI Model. In this post, I'll show you an example to disable the load for some queries, especially queries that used as the intermediate transformation to produce the final query for the model. This is a very basic tip but very important when your model grows big. If you want to learn more about Power BI, read the [Power BI online book from Rookie to Rock Star](#).

## Prerequisite

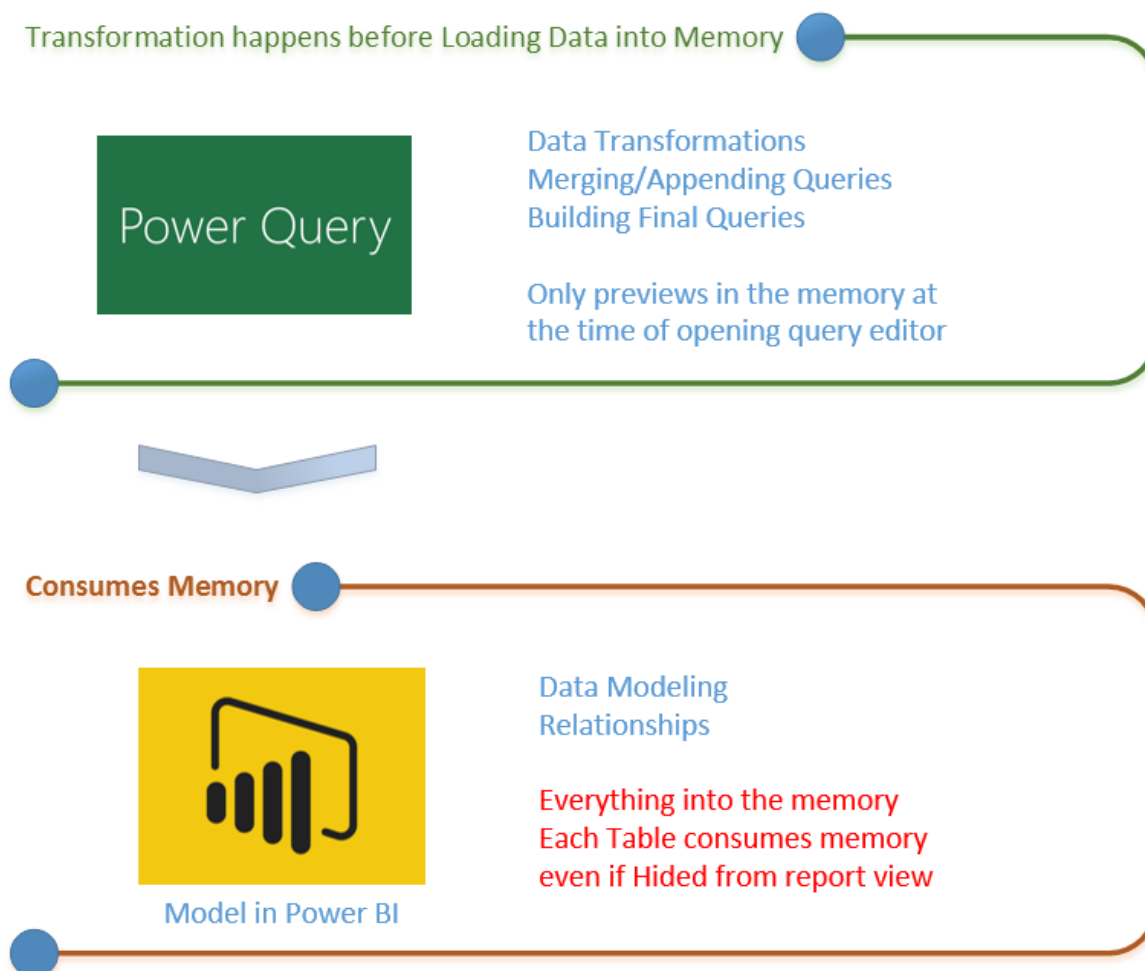
For running examples of this book, you need to download the ZIP file here;

[Download ZIP file for CSVs](#)

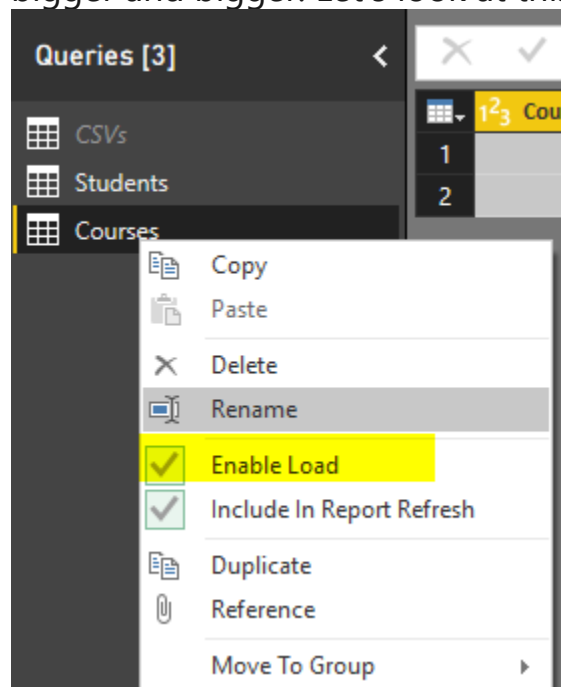
## Load Mechanism for Power Query

By Default, all queries in Power Query will be loaded into the Power BI model. This behavior might be the desired behavior if you are connecting to a proper

star schema modeled data warehouse because normally you don't need to make many changes in the structure of queries. However this brings some issues if you are connected to a transactional data store, some files, web source, and many other non-star schema modeled data sources. Normally when you get data from a data source, you apply transformations for rows and columns, and merge queries or append them, and you might end up to have five tables out of 10 queries as final queries. By default when you close and apply your query editor window, all queries will be loaded into the model no matter if you want to use them in your final model or not.



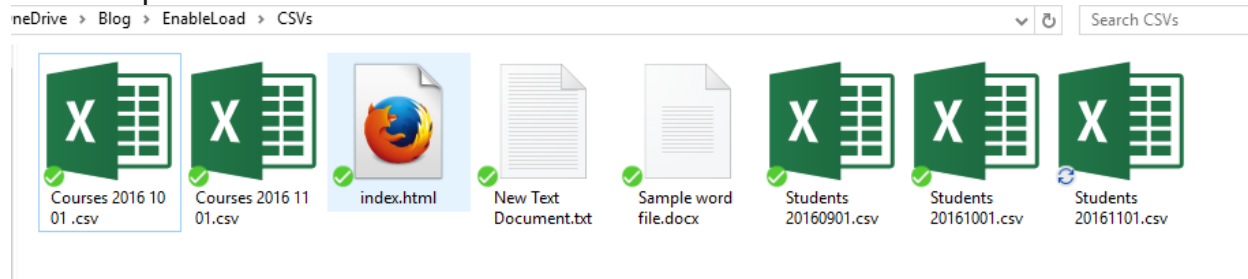
For every query that loads into model, memory will be consumed. And Memory is our asset in the Model; less memory consumption leads to better performance in most of the cases. I have seen lots of models that people Hide the unwanted query from the model; This approach doesn't help to the performance because a hidden query will still consume the memory. The best approach is to disable loading before closing the query editor. Disabling Load doesn't mean the query won't be refreshed, it only means the query won't be loaded into the memory. When you click on Refresh model in Power BI, or when a scheduled refresh happens even queries marked as Disable Load will be refreshed, but their data will be used as an intermediate source for other queries instead of loading directly into the model. This is a very basic performance tuning tip, but very important when your Power BI model grows bigger and bigger. Let's look at this through an example.



## Example Scenario

In this example Scenario, I want to get a list of all files from a directory. There are two types of CSV files in the directory, some Students files, and some

course files, both are CSV files, but different structured. Also, there might be some other files in the directory, such as Word files or HTML files which I don't want to process.

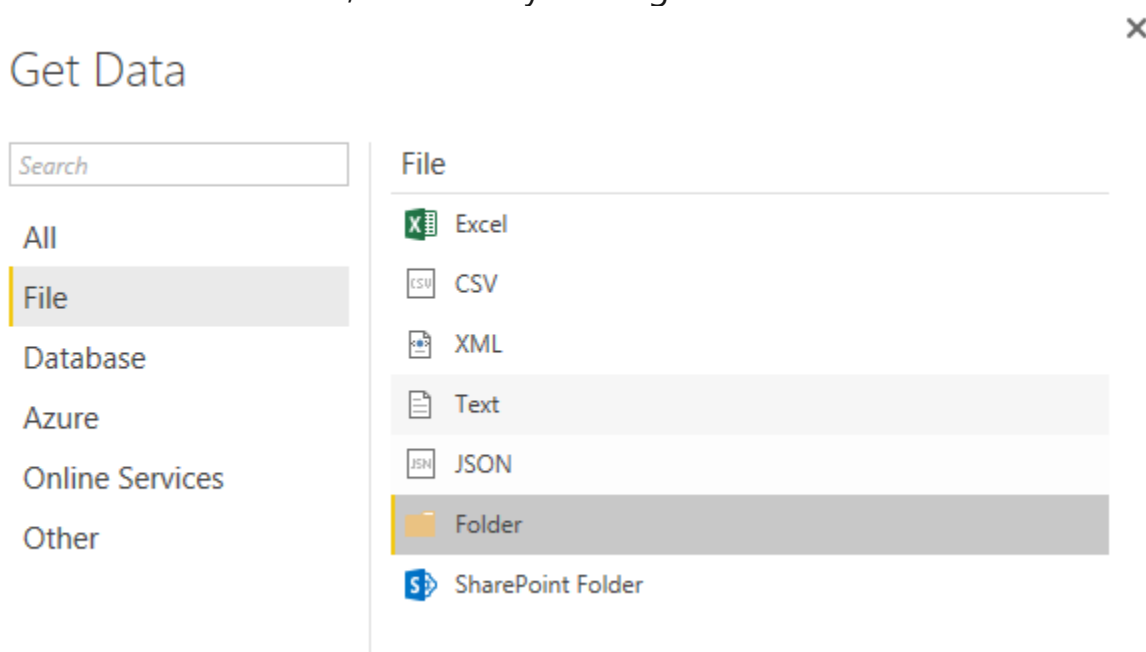


The aim is to load data rows of all students and courses as two separate tables into the model. Instead of fetching files from the folder twice, we use one query to fetch the files and then use it as a reference for other queries. The referenced query itself doesn't need to be loaded into the model. Let's build the solution and see how it works in action.

## Build the Transformations

### Get Data from Folder

Open a new Power BI file, and start by Getting Data from Folder



Enter the path of the folder that contains all files (Files in this folder can be downloaded from ZIP file up in the prerequisite section of this post)

## Folder

Choose a folder.

Folder Path

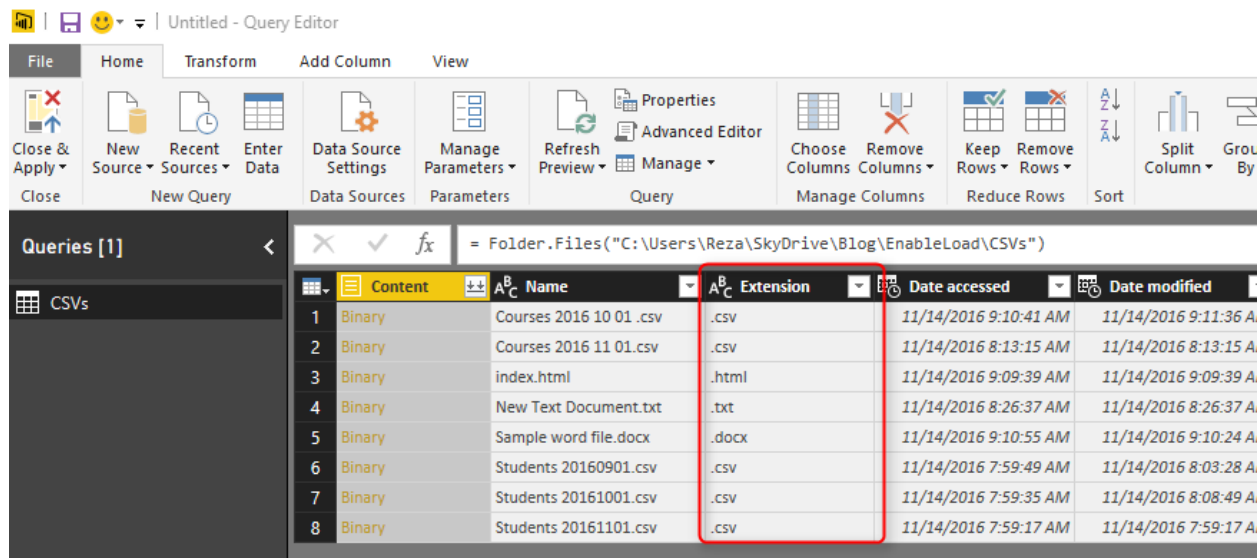
C:\Users\Reza\SkyDrive\Blog\EnableLoad\CSVs

Browse...

OK

Cancel

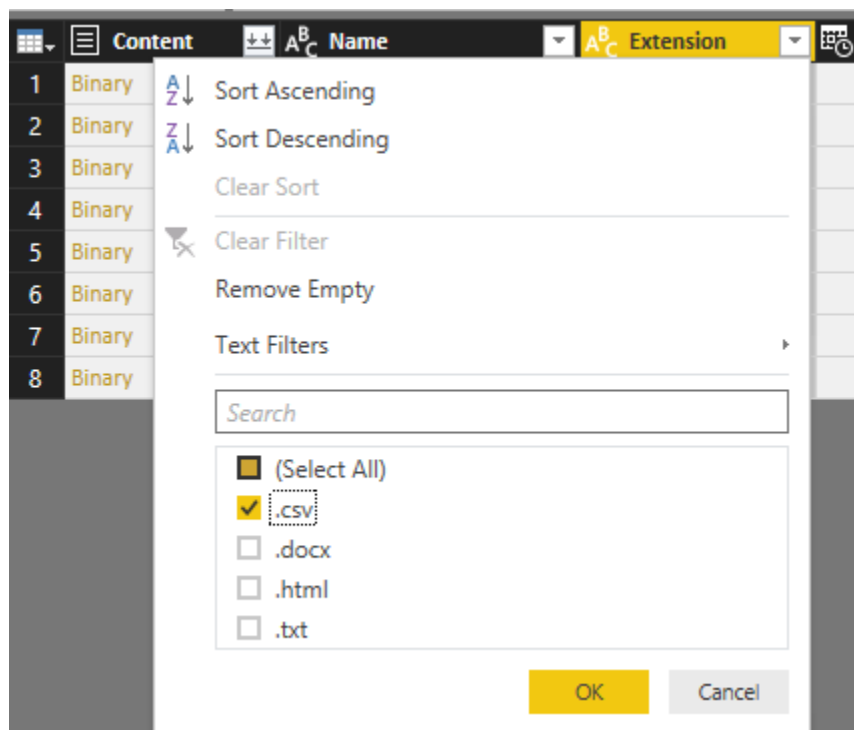
Click on Edit in the preview showed in the navigator window to open the folder content in Query Editor. As you see, there are a number of different files in the folder.



The screenshot shows the Power Query Editor interface. The top ribbon includes tabs for File, Home, Transform, Add Column, and View. The Home tab is active, showing options like Close & Apply, New Source, Recent Sources, Enter Data, Data Source Settings, Manage Parameters, Refresh Preview, Properties, Advanced Editor, Choose Columns, Remove Columns, Keep Rows, Remove Rows, Sort, Split Column, and Group By. The Queries [1] pane on the left shows a query named 'CSVs'. The main area displays the query formula: `= Folder.Files("C:\Users\Reza\SkyDrive\Blog\EnableLoad\CSVs")`. Below the formula, a table of files is shown with columns: Content, Name, Extension, Date accessed, and Date modified. The 'Extension' column is highlighted with a red box, and the 'Date accessed' and 'Date modified' columns are also highlighted. The table lists 8 files, including CSVs, HTML, TXT, and DOCX files.

	Content	Name	Extension	Date accessed	Date modified
1	Binary	Courses 2016 10 01 .csv	.csv	11/14/2016 9:10:41 AM	11/14/2016 9:11:36 A
2	Binary	Courses 2016 11 01.csv	.csv	11/14/2016 8:13:15 AM	11/14/2016 8:13:15 A
3	Binary	index.html	.html	11/14/2016 9:09:39 AM	11/14/2016 9:09:39 A
4	Binary	New Text Document.txt	.txt	11/14/2016 8:26:37 AM	11/14/2016 8:26:37 A
5	Binary	Sample word file.docx	.docx	11/14/2016 9:10:55 AM	11/14/2016 9:10:24 A
6	Binary	Students 20160901.csv	.csv	11/14/2016 7:59:49 AM	11/14/2016 8:03:28 A
7	Binary	Students 20161001.csv	.csv	11/14/2016 7:59:35 AM	11/14/2016 8:08:49 A
8	Binary	Students 20161101.csv	.csv	11/14/2016 7:59:17 AM	11/14/2016 7:59:17 A

Filter the Extension to only .csv. Note that both Course and Student files are “.CSV” files which are what we need.

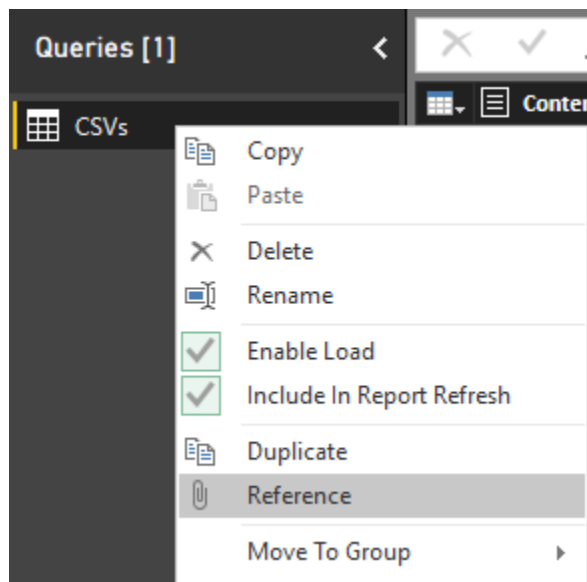


Now the subset includes Course files and Students files which are different in the structure. We have to apply transformations on each set individually.

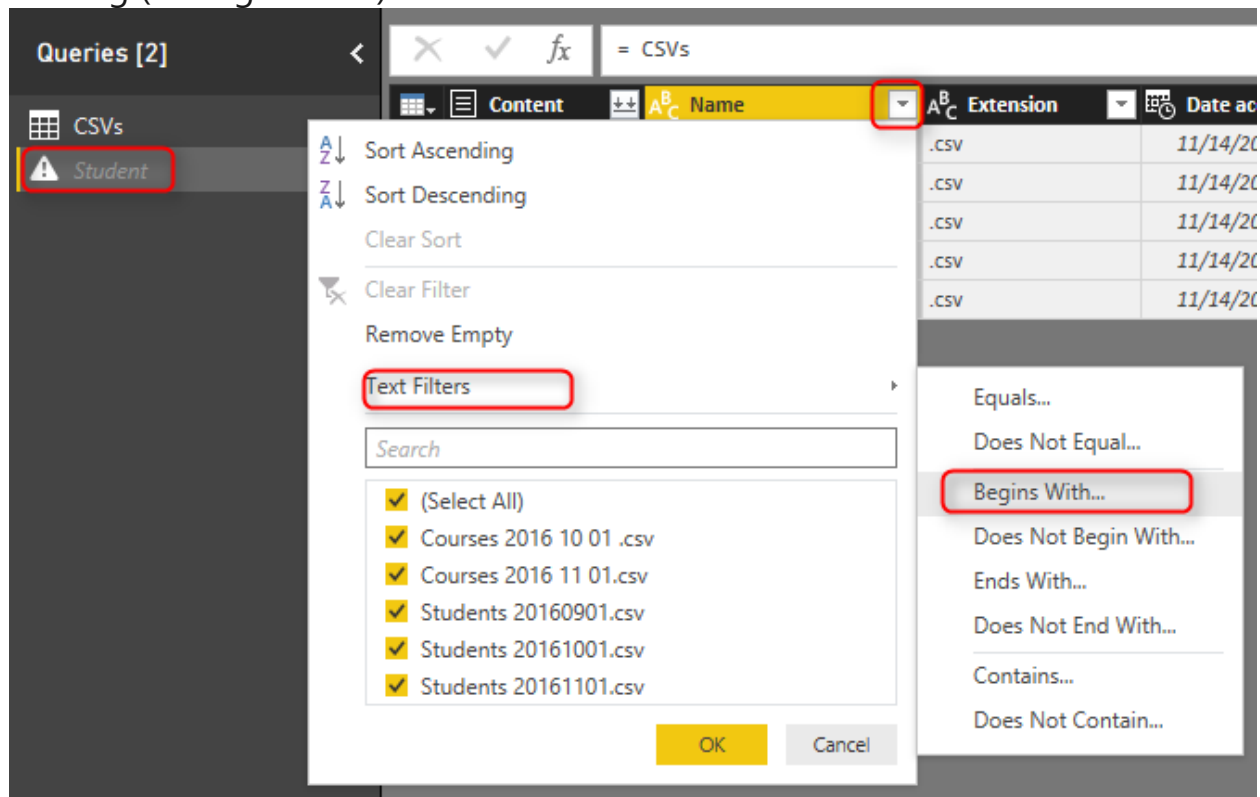
		= Table.SelectRows(Source, each ([Extension] = ".csv"))			
	Content	Name	Extension	Date accessed	Date modified
1	Binary	Courses 2016 10 01 .csv	.csv	11/14/2016 9:10:41 AM	11/14/2016 9:11:36 AM
2	Binary	Courses 2016 11 01.csv	.csv	11/14/2016 8:13:15 AM	11/14/2016 8:13:15 AM
3	Binary	Students 20160901.csv	.csv	11/14/2016 7:59:49 AM	11/14/2016 8:03:28 AM
4	Binary	Students 20161001.csv	.csv	11/14/2016 7:59:35 AM	11/14/2016 8:08:49 AM
5	Binary	Students 20161101.csv	.csv	11/14/2016 7:59:17 AM	11/14/2016 7:59:17 AM

## Students Query

Because I don't want to repeat the process of getting all files from a folder, and I want to split the data into two data sets; one for Students and another one for Courses. I'll generate a REFERENCE from the main query. Right click on the query (which called CSVs in my example), and select Reference.



This will generate a new query named as CSVs (2). This new query is NOT A COPY of the first query. This is only a reference from the first query. Which means if the first query changes, the source for this query will also change. Rename this new query to be Student. Filter the Name column to everything starting (or begins with) "Student".



The reason that I don't type it in the search box and use Text Filters specifically is that the search box will filter the data statistically based on values that exist in the current data set. If in the future new values (file names) comes in the folder this won't consider that. However, the Text Filter will apply on any data set because it will be filtered dynamically. (I'll write a post later to explain that in details). In the Filter Rows window type in Student as a filter for begins with.

## Filter Rows

☒ Basic ☐ Advanced

Show rows where: Name

begins with Student

☒ And ☐ Or

Type or select a value

OK

Cancel

Now you will see only Students files.

= Table.SelectRows(Source, each Text.StartsWith([Name], "Student"))						
	Content	Name	Extension	Date accessed	Date modified	Date create
1	Binary	Students 20160901.csv	.csv	11/14/2016 7:59:49 AM	11/14/2016 8:03:28 AM	11/14/2016
2	Binary	Students 20161001.csv	.csv	11/14/2016 7:59:35 AM	11/14/2016 8:08:49 AM	11/14/2016
3	Binary	Students 20161101.csv	.csv	11/14/2016 7:59:17 AM	11/14/2016 7:59:17 AM	11/14/2016

Click on the Combine Binaries icon on the header of the Content column to combine all CSV files into one.

Queries [2]

- CSVs
- Student

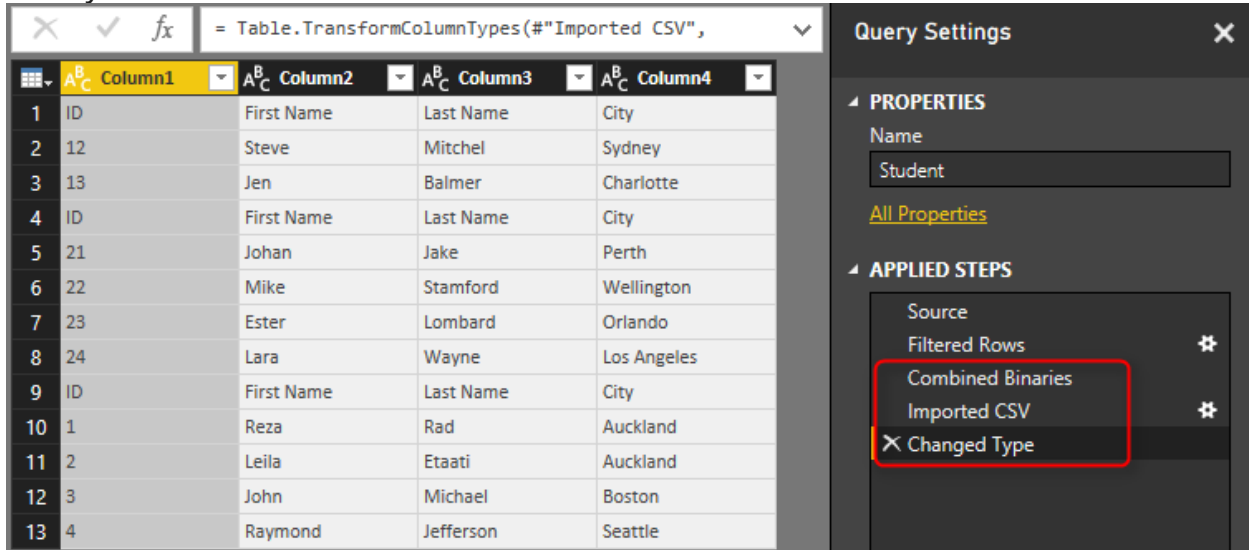
Content

Combine Binaries

= Table.SelectRows(Source, each Text			
	Content	Name	Extension
1	Binary	Students 20160901.csv	.csv
2	Binary	Students 20161001.csv	.csv
3	Binary	Students 20161101.csv	.csv

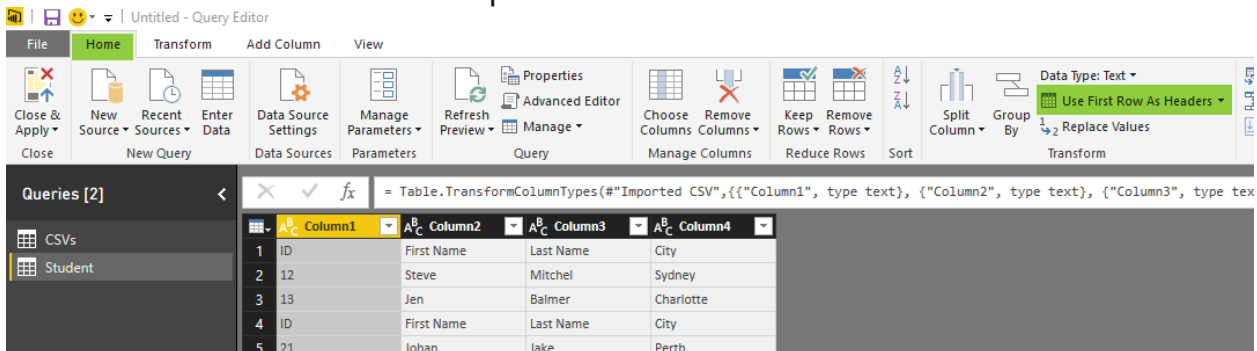


After combining binaries, Power Query will also import the CSV into the table and do automatic data type conversion. You can see all three steps in the Query Editor.



ID	First Name	Last Name	City
12	Steve	Mitchel	Sydney
13	Jen	Balmer	Charlotte
21	Johan	Jake	Perth
22	Mike	Stamford	Wellington
23	Ester	Lombard	Orlando
24	Lara	Wayne	Los Angeles
1	Reza	Rad	Auckland
2	Leila	Etaati	Auckland
3	John	Michael	Boston
4	Raymond	Jefferson	Seattle

The data table needs a couple of changes before being ready. First; set the column names. The first row has column names. So use the menu option of "Use First Row As Headers" to promote the first row to be column headers.



This brings column headers;

fx = Table.PromoteHeaders("#Changed Type")

	ID	First Name	Last Name	City
1	12	Steve	Mitchel	Sydney
2	13	Jen	Balmer	Charlotte
3	ID	First Name	Last Name	City
4	21	Johan	Jake	Perth
5	22	Mike	Stamford	Wellington
6	23	Ester	Lombard	Orlando
7	24	Lara	Wayne	Los Angeles
8	ID	First Name	Last Name	City
9	1	Reza	Rad	Auckland
10	2	Leila	Etaati	Auckland
11	3	John	Michael	Boston
12	4	Raymond	Jefferson	Seattle

Also, you need to remove all extra header rows from the combined data set. There are some rows with "ID, First Name, Last Name, and City" as their values which should be removed from the data set. You can remove them with a Text Filter of Does Not Equal to on the ID column. Values should not be equal to "ID" because every row with "ID" in the first column's value is a header row and should be removed.

### Filter Rows

☒ Basic ☐ Advanced

Show rows where: ID

does not equal ID

☒ And ☐ Or

Type or select a value

OK

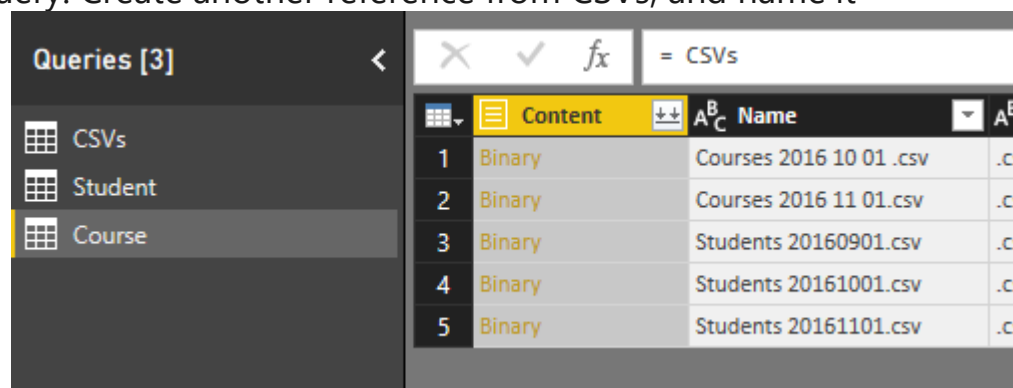
Cancel

Now you have your cleaned data set for Students. Note that I have also applied a data type conversion for column ID to type the whole number;

	ID	First Name	Last Name	City
1	12	Steve	Mitchel	Sydney
2	13	Jen	Balmer	Charlotte
3	21	Johan	Jake	Perth
4	22	Mike	Stamford	Wellington
5	23	Ester	Lombard	Orlando
6	24	Lara	Wayne	Los Angeles
7	1	Reza	Rad	Auckland
8	2	Leila	Etaati	Auckland
9	3	John	Michael	Boston
10	4	Raymond	Jefferson	Seattle

## Courses Query

The other entity is Course which we need to create as a reference from the CSVs query. Create another reference from CSVs, and name it



Queries [3]		
CSVs	Student	Course

Content	Name
1	Binary Courses 2016 10 01 .csv
2	Binary Courses 2016 11 01.csv
3	Binary Students 20160901.csv
4	Binary Students 20161001.csv
5	Binary Students 20161101.csv

Course.

Create a Text Filter for Begins with on the Name Column with the value of "Course"

### Filter Rows

☒ Basic ☐ Advanced

Show rows where: Name

begins with

And then you'll have only Course files. Combine Binaries on that.

	A <sup>B</sup> <sub>C</sub> Column1	A <sup>B</sup> <sub>C</sub> Column2	A <sup>B</sup> <sub>C</sub> Column3	A <sup>B</sup> <sub>C</sub> Column4	A <sup>B</sup> <sub>C</sub> Column5
1	Course Number	Department	title	Capacity	Date
2	2	BI	SSIS Training	30	2016/10/1
3	3	Database	Execution Plans	30	2016/10/15
4	Course Number	Department	title	Capacity	Date
5	12	BI	Power BI Training	30	2016/11/1
6	13	Database	High Availability in a Day	30	2016/12/1

Same as Student query, apply transformations in this order;

- Promote headers with "Use First Row As Headers"
- Create a Text Filter on the Course Number Column that values does not equal to "Course Number".
- Apply data type conversion on Course Number and Capacity to Whole Number and Date to Date.

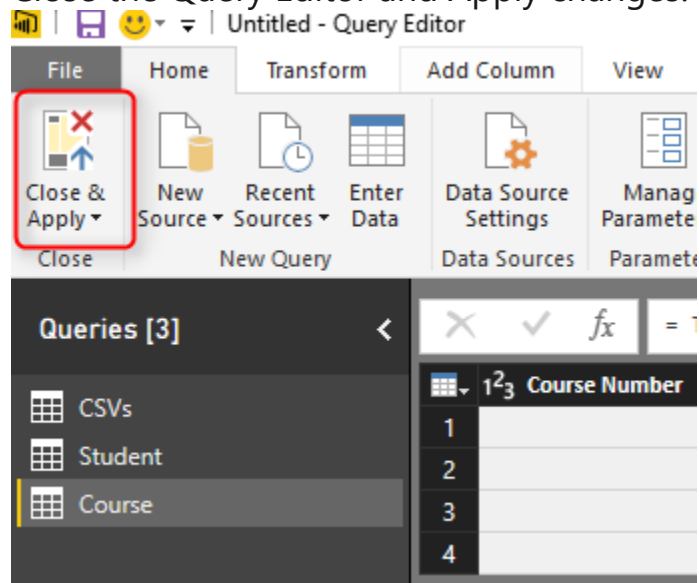
Here is the final data set for course;

	1 <sup>2</sup> <sub>3</sub> Course Number	A <sup>B</sup> <sub>C</sub> Department	A <sup>B</sup> <sub>C</sub> title	1 <sup>2</sup> <sub>3</sub> Capacity	Date
1	2	BI	SSIS Training	30	10/1/2016
2	3	Database	Execution Plans	30	10/15/2016
3	12	BI	Power BI Training	30	11/1/2016
4	13	Database	High Availability in a Day	30	12/1/2016

## Default Behavior: Enable Load

Now to see the problem, without any changes in the default load behavior,

Close the Query Editor and Apply changes.

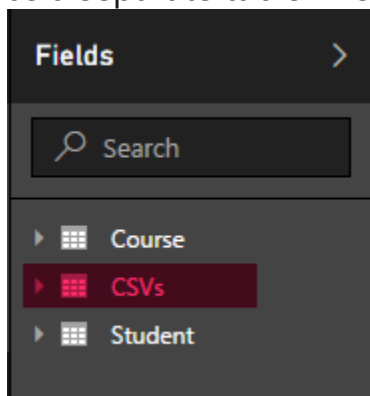


You will see that after the refresh, three queries loads in the model; Student, Course, and CSVs.

## Apply Query Changes

- CSVs  
Creating connection in model...
- Student  
Creating connection in model...
- Course  
Creating connection in model...

Student and Course are expected tables in the model. However, CSVs is not useful. We already fetched everything we wanted from the query, and this is used as an intermediate query for loading Student and Course. Having CSVs as a separate table in our model has two problems;



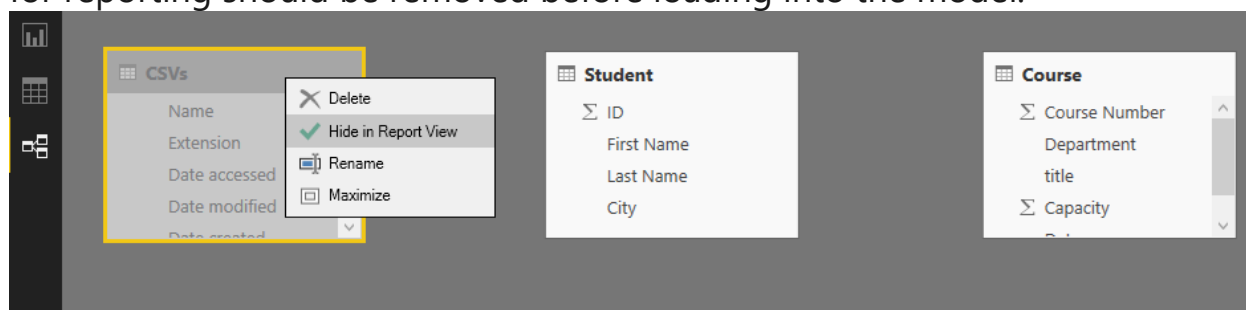
- one extra table; confusion for users
- extra memory consumption

The main issue is the memory consumption for this extra table. and normally memory consumption will reduce the performance and brings a heavier load to the model. In this example, CSVs table only has a few rows. But this is just a sample data, in real-world examples, you might have intermediate tables with

millions of rows. You need to remove every unused table from the Power BI model to enhance the memory consumption.

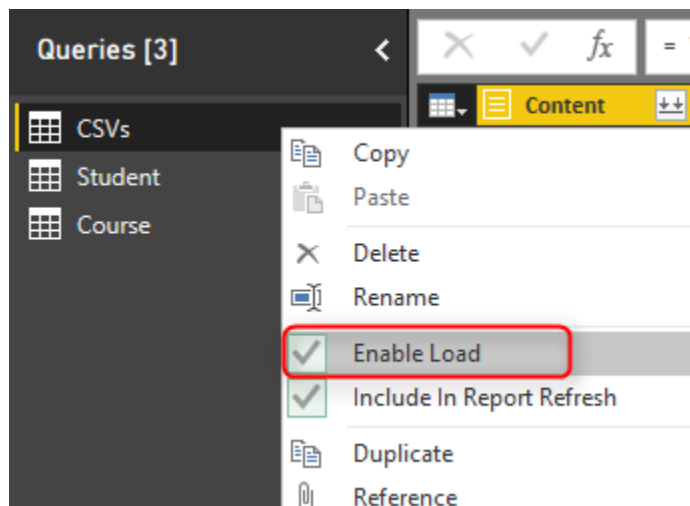
## What about Hiding from Report?

The first issue “Confusion for users” can be solved by hiding the table from report view. You can do that in the Relationship tab of the model, right click on the CSVs table, and click on Hide from the report view. This method HIDEs the table from the report view. However the table still exists, and it consumes memory! It is just hidden. Hiding a table from the report view is good for tables that you need for the modeling. For example, a relationship table (that creates many to many relationships) should be kept in the model but be hidden. Because it creates a relationship, but it is not useful for the report viewer. Any other tables that are not used for a relationship and is not used for reporting should be removed before loading into the model.



## Disable Load to Save Memory

Go back to Query Editor, and right click on CSVs query. You will see that by default every query is checked as Enable Load.



Click on the Enable Load to disable it. You will see a message saying that any visuals attached to this query will not work after this change. Which is fine, because this query is not used (and won't be) for any visualization. The message is: "Disabling load will remove the table from the report, and any visuals that use its columns will be broken."

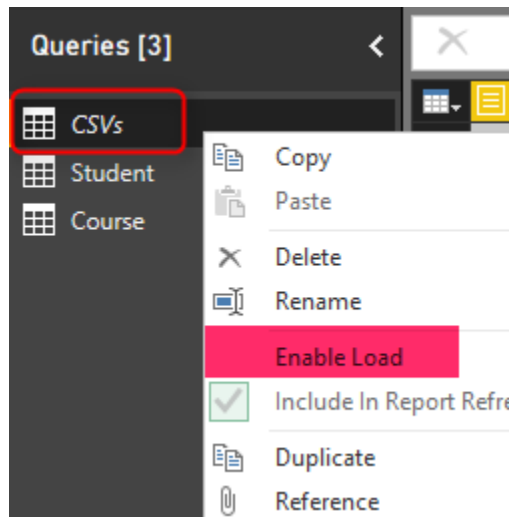
### Possible Data Loss Warning

Disabling load will remove the table from the report, and any visuals that use its columns will be broken.

Continue

Cancel

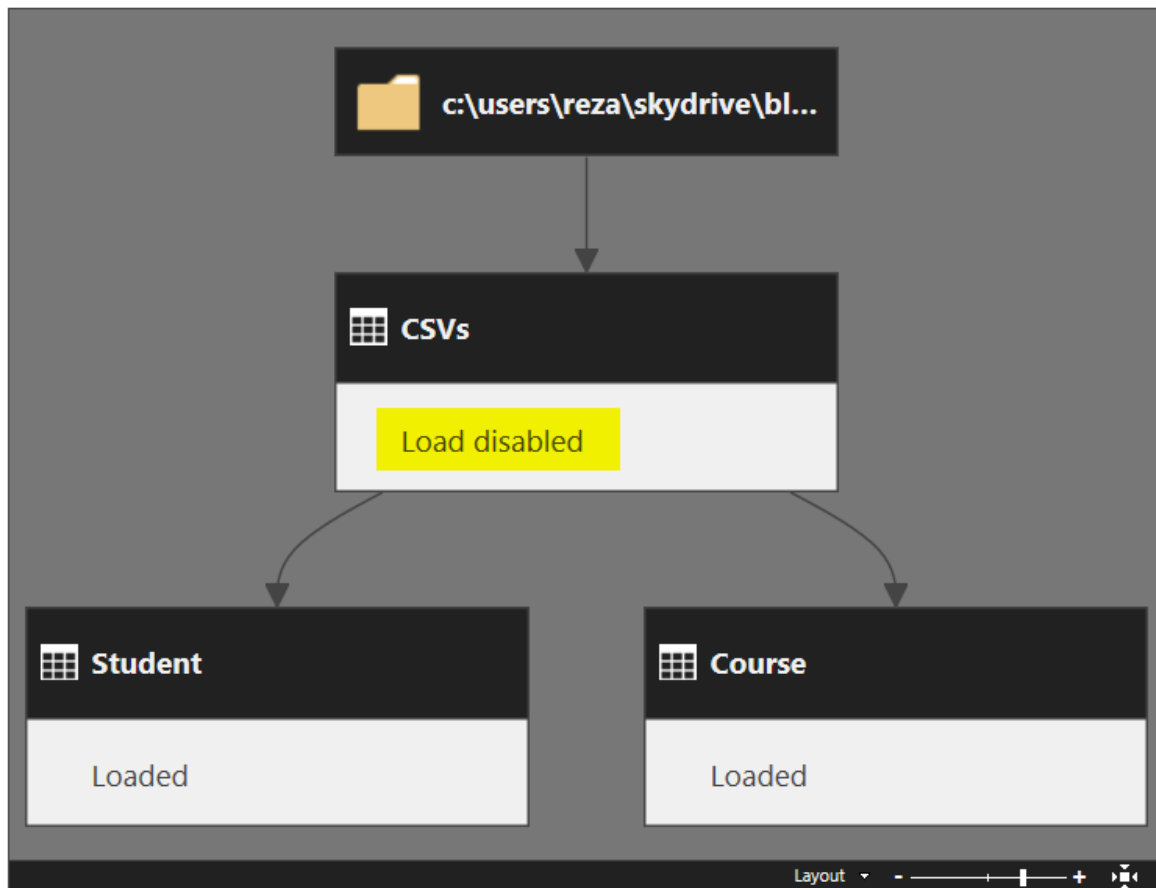
Click on Continue, and you will see the load is disabled for the query now. The query name will be also in *Italic* font illustrating that it won't be loaded into the model. Note that query will be refreshed and if new files come to the directory it will pass it on to Course and Student Queries. It just won't be loaded into the model itself.



You can also see in the View tab of Query Editor, on the Query Dependencies that this query is the source of the other two queries. And in that view, it also shows which query is load disabled.

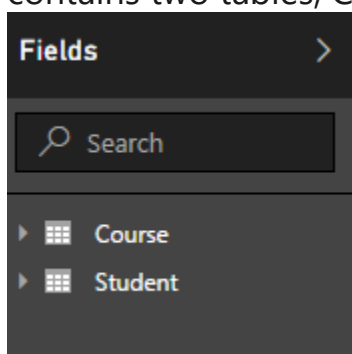


## Query Dependencies



Close

Now close and apply the Query Editor. This time you see that CSVs query won't be loaded into the model. Your relationship tab and the report tab only contains two tables; Course, and Student.

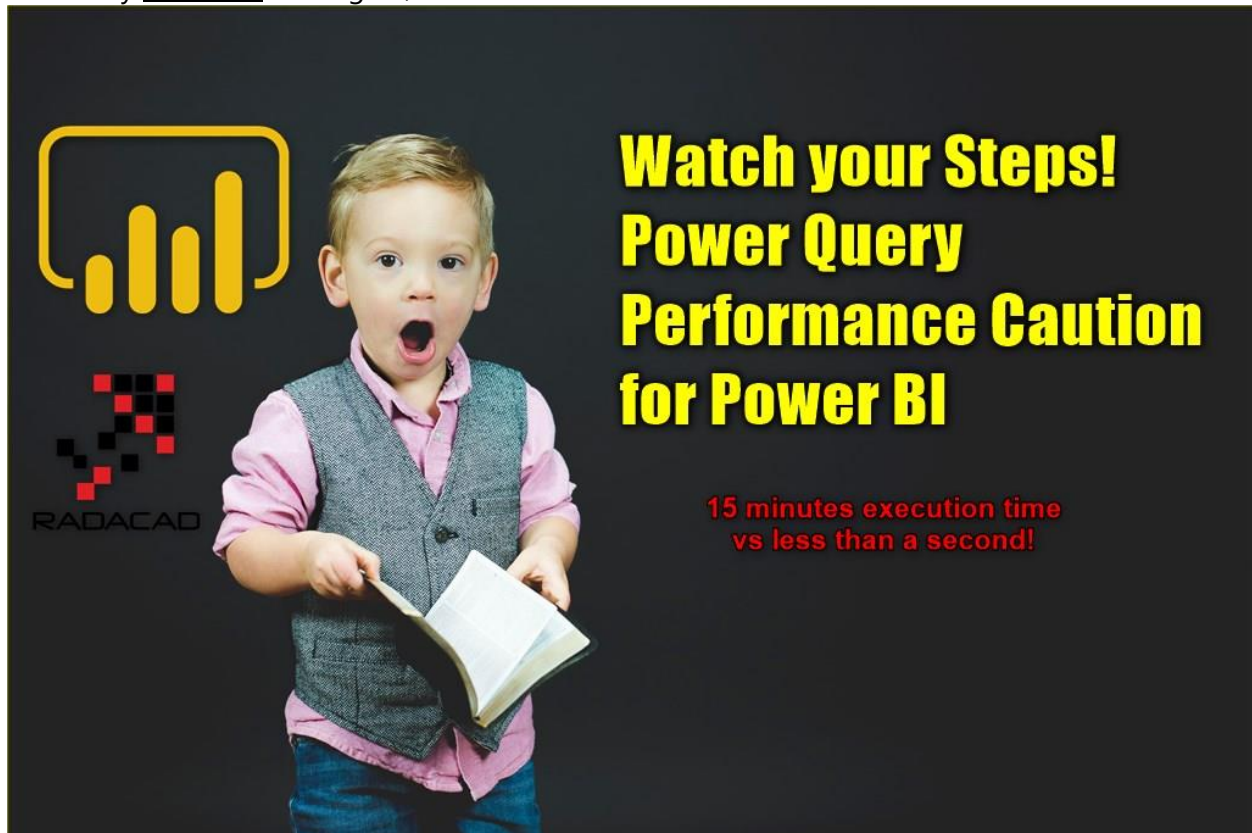


## Summary

In summary, you've seen how an easy tip can save the memory and enhance the performance of the Power BI model. Always remember that Query Editor is your ETL (Extract, Transform, Load) engine. It will apply all transformations before loading the data into the model, but once it finished the transformation all queries will be loaded into the model, and they take memory. By default, all queries are Enabled to Load into the model. Simply change that for queries that are not required in the model.

# Watch Your Steps! Power Query Performance Caution for Power BI

Posted by [Reza Rad](#) on Aug 14, 2018



I work with Power Query transformations every day these days, and I want to share one simple, but critical caution with you. If you use Power Query a lot, this tip can improve the performance of your transformation significantly. The number of steps that you add in a query counts in the performance of your data transformation (if you have too many steps), I like to show this to you through an example. If you like to learn more about Power BI and Power Query, read the [Power BI from Rookie to Rock Star book](#).

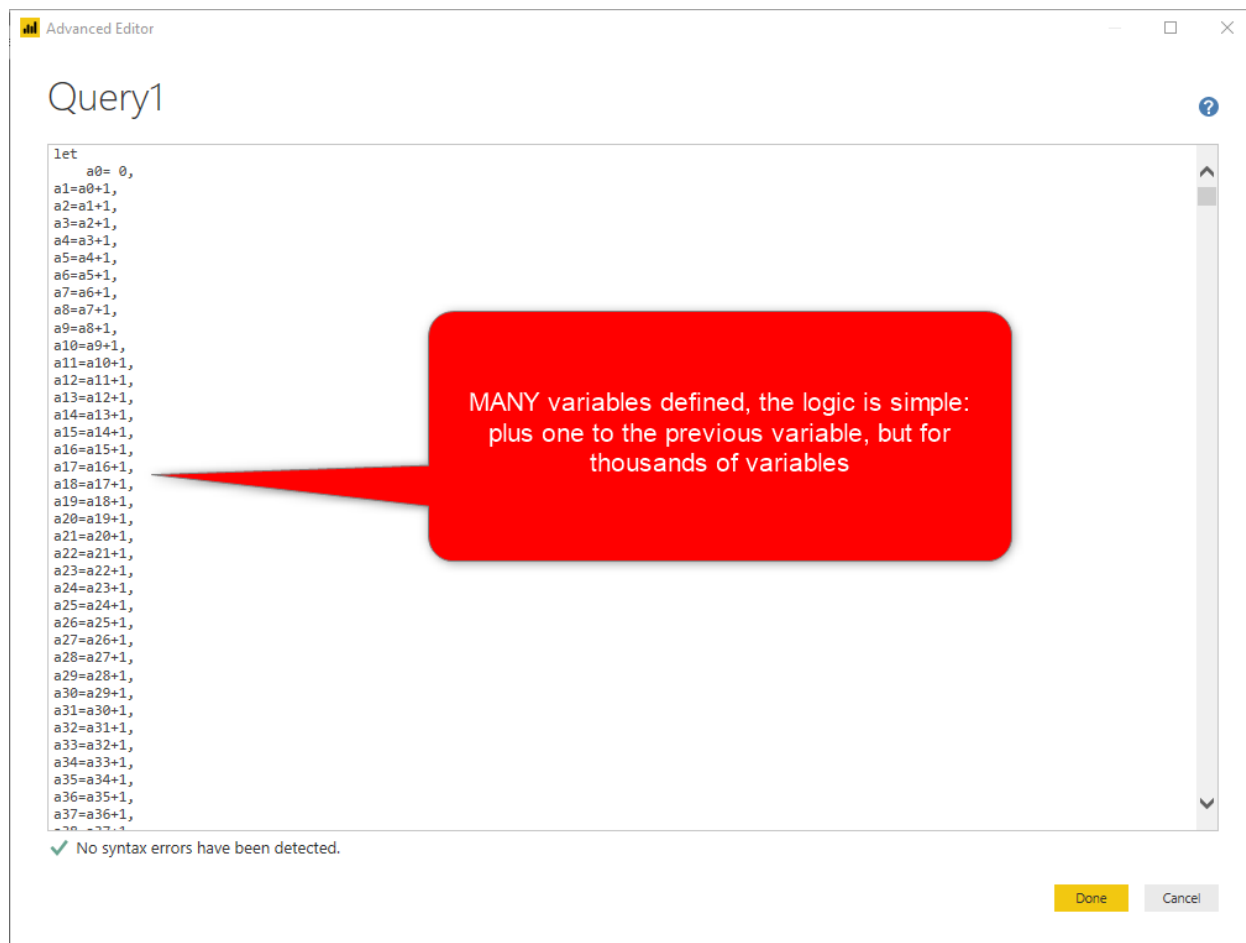
## Too Many Variables

This is a sample Power Query file, which in that I do a very simple transformation. The transformation is adding one to the existing number.

However, in this sample, we are doing it for thousands of steps! One step at a time, we are adding thousands to a number. The main reason to do it this way is to show you what is the performance you get when you have too many variables (or let's say steps) in Power Query.

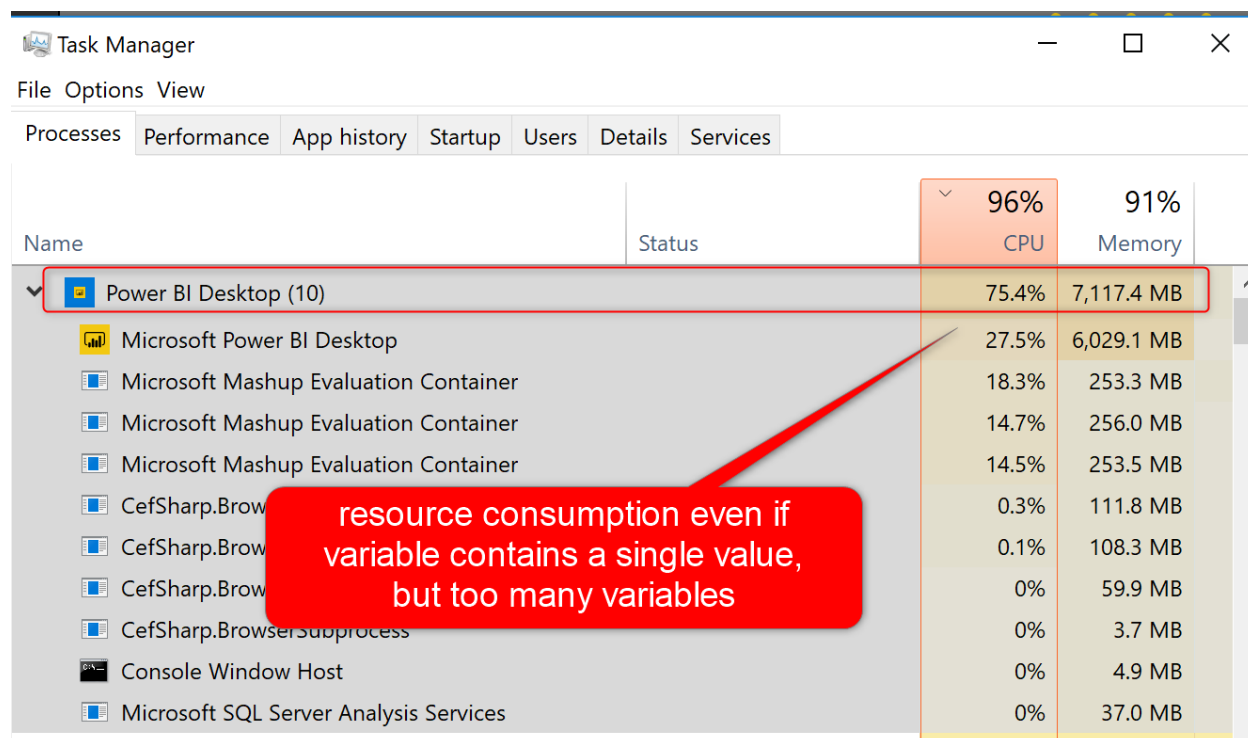
Here is my sample query:

```
1 let
2     a0= 0,
3     a1=a0+1,
4     a2=a1+1,
5     a3=a2+1,
6     ... //
7     ... // each variable used in the next variable with a plus one
8     ... //
9     a1808=a1807+1,
10    a1809=a1808+1,
11    a1810=a1809+1,
12    a1811=a1810+1,
13    a1812=a1811+1
14 in
15     a1812
```



The query above takes 15 minutes to run on my Surface Book 2 machine with Core i7 CPU and 16GB memory! The 15 minutes that you cannot touch Power BI through it. It will not respond to your actions; you have to wait for that time to see the result after that.

Oh! That is a long time for a query that adds one value in every step. Let's see how the resources consumption is in the system with Power BI Desktop and Power Query running. Here it is only halfway through:



Uh! Even with a simple calculation like adding one to a number I had over 10GB memory consumption and 70% CPU usage for a long period! What do you think caused it? The number of variables of course. There is nothing else in this query.

Having too many variables, or too many steps, cause performance issues.

Power Query allocates memory for every variable, and memory consumption raises significantly. The processor also takes lots of time to process that number of variables through the Power Query Engine.

Of course, this example was an exaggerated example of too many variables.

You would never have 1,800 variables. However, this example also was on a variable with a single simple numeric value. In most of the cases, your variables are tables with many data rows and columns. So this can happen in a real-world scenario for you even with hundreds of steps or variables. You need to watch the number of variables in other words.

*Too Many Variables will cause performance issue!*

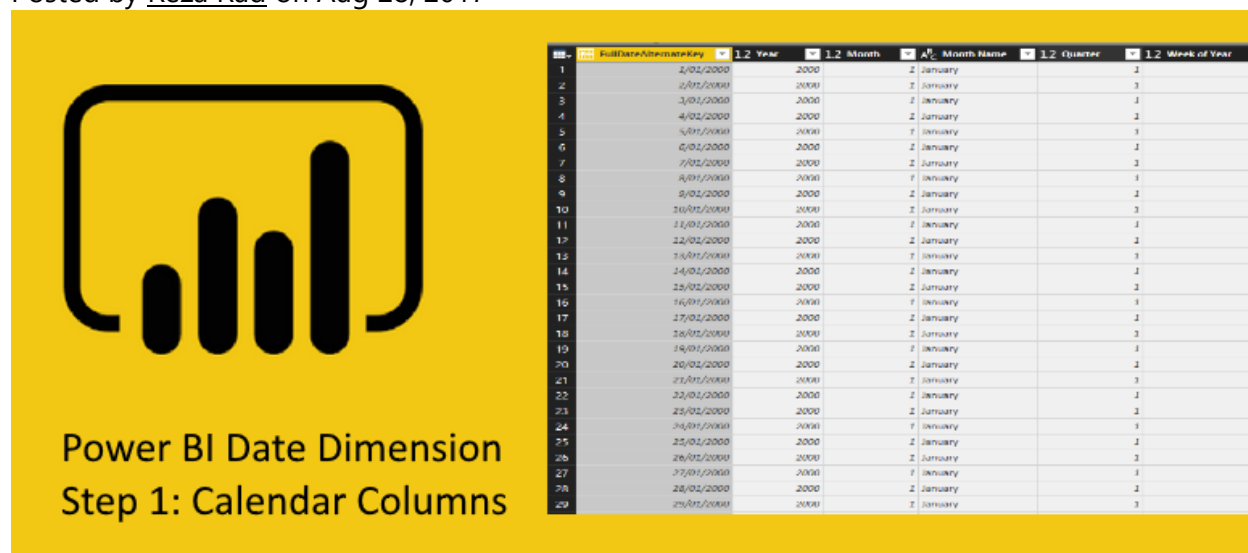


## Part VII: Power Query Use Cases



# Create a Date Dimension in Power BI in 4 Steps – Step 1: Calendar Columns

Posted by [Reza Rad](#) on Aug 28, 2017



I have written multiple blog posts so far about creating a date dimension. However, I still get the question about how to create a date dimension. In this series of blog posts, I am going to explain in details how you can create a date dimension easily in Power BI (based on Power Query). This date dimension will be configurable with start and end date will have fiscal calendar columns, and most importantly will have public holidays fetched live. In this first step, we are creating the based for the date dimension for calendar dates. If you like to learn more about Power BI; read [Power BI book from Rookie to Rock Star](#).

## Introduction

I wrote many years ago about [how to create a date dimension in T-SQL](#) and after a while another post to [create a date dimension in Power Query with M Script](#). I have also written another blog post about the [importance of the date dimension, and why it is critical to have a date dimension](#). In the above posts, I just provided the script to create the date dimension, but I never explained the

procedure. I feel the need that people would like to know how to create the date dimension. So, in this blog series I am going to explain in details from end to end; how to create a date dimension in Power BI using Power Query. At the final post, I will provide the whole script to generate date dimension.

## **Why Power Query?**

When it comes to creating the date (or calendar) dimension in Power BI, there is always a question: Should I create the dimension with Power Query or DAX? This is a very good question to ask. It means that you know that there are multiple ways of creating it. What is the difference? The answer is that for many scenarios these are similar. So, it might not be different to use Power Query or DAX for it. However, there is a big difference.

Power Query can fetch data from live web APIs. This functionality gives you the power to fetch public holidays live from an API. You cannot do this with DAX! Apart from this big difference, the majority of other requirements can be done with both; you can write calculations in both M or DAX to get calendar columns as well as fiscal columns. In many scenarios, public holidays play an important role in analyzing data. You would like to know how the sales were on holidays compared to other holidays, etc.

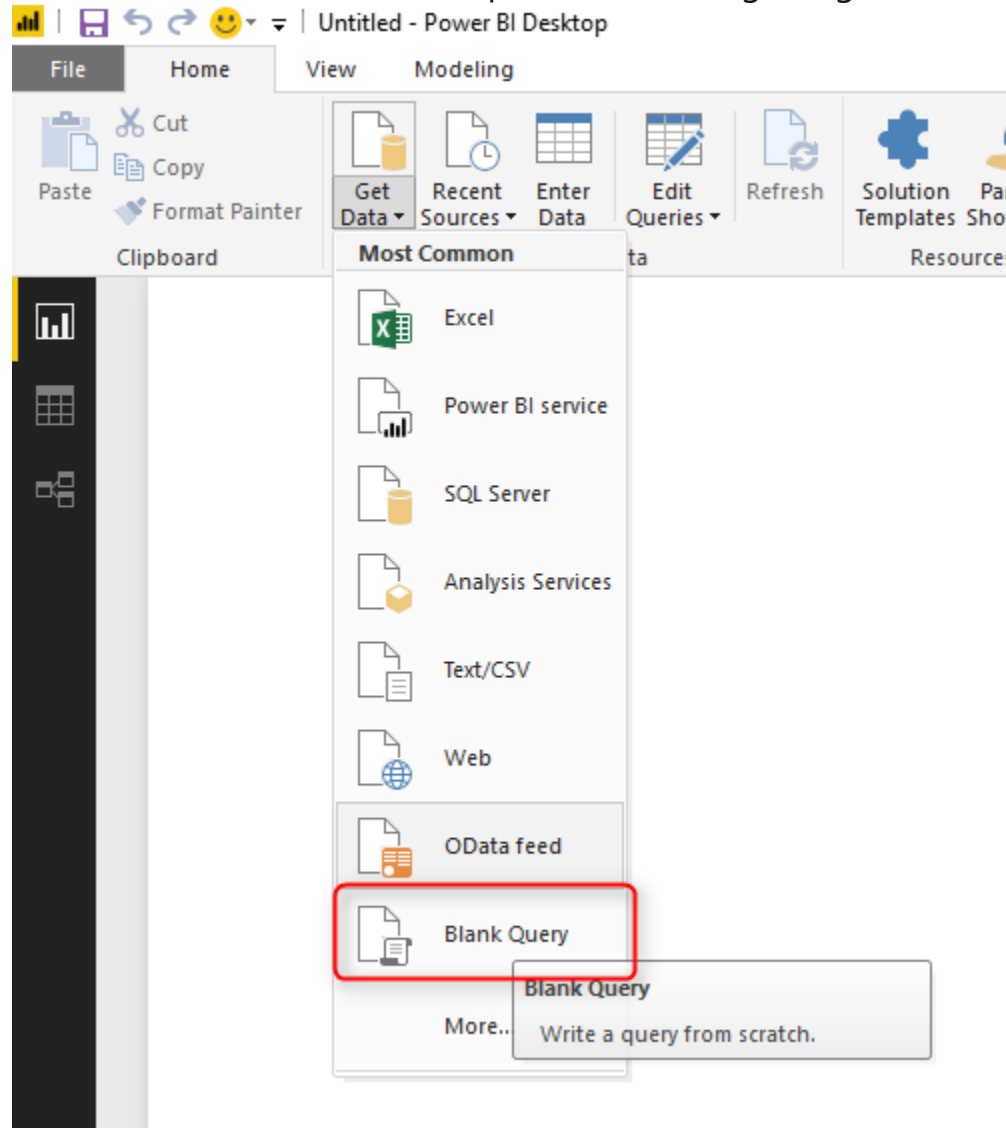
On the other hand side; some people refer to creating date dimension in DAX is easier. Well, that is not a true statement. You can easily copy the whole M script into a new Power Query blank query, and that generates a date dimension for you. As simple as copying DAX expression.

So, based on explanations above, I am going to create the date dimension in Power Query

## Getting Started

For building this date dimension, you don't need any based data set. We will build it from scratch. All you need is to know start year and end year for the date dimension. Let's start building the dimension;

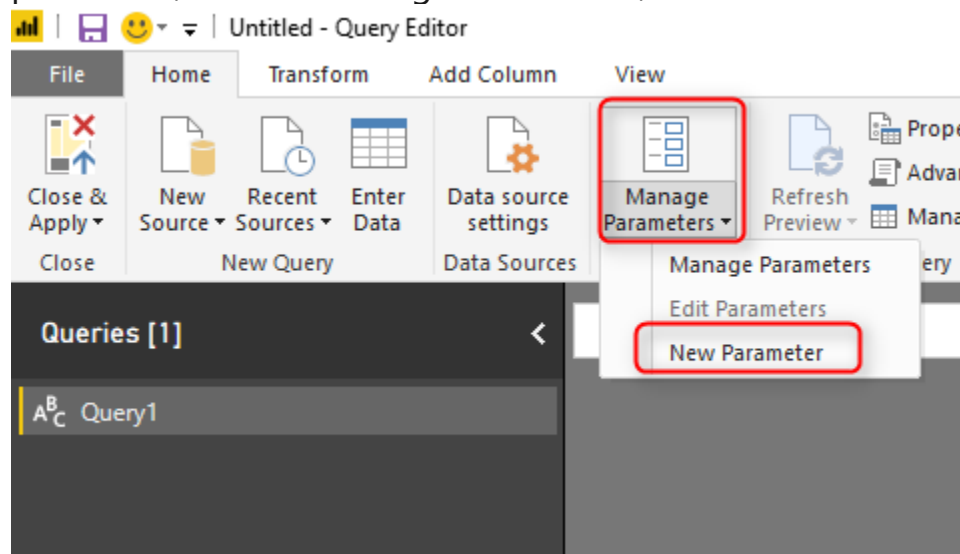
Create a new Power BI Desktop file, start with getting Data -> Blank Query



Blank Query means query will return an empty string. Then you will be redirected to Query Editor window. We will use this query as a base for our date dimension.

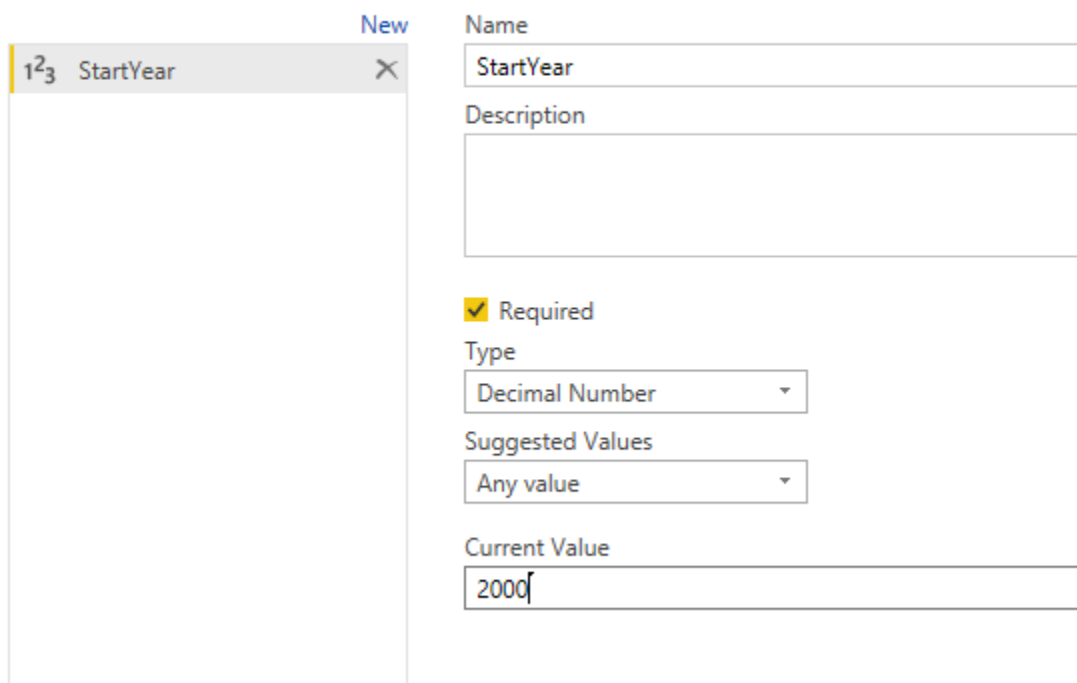
## Parameters

Let's create parameters for Start Year and End Year. For creating every parameter, click on Manage Parameters, and then New Parameter.

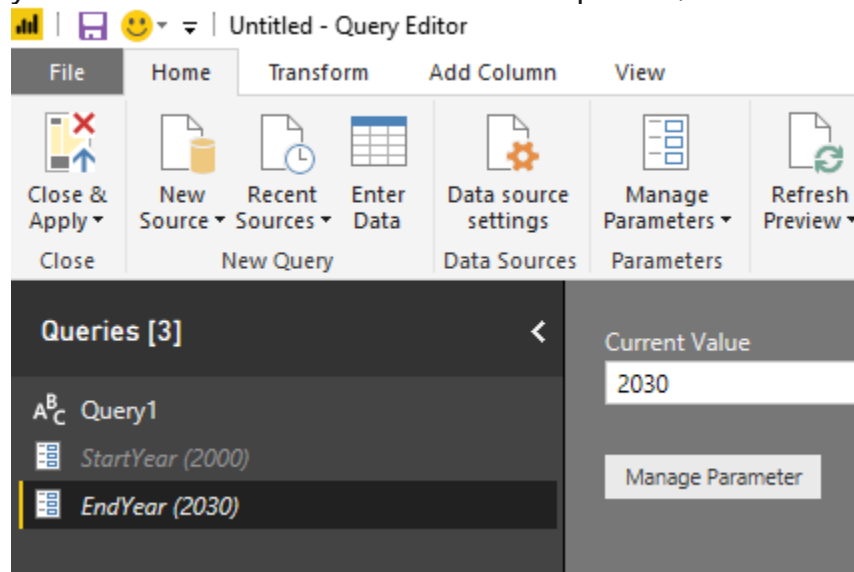


Create one parameter for StartYear with the data type of decimal, and the default value of that you want as a start year

### Parameters

A screenshot of the 'Parameters' dialog box in Power BI. On the left, a list of parameters shows 'StartYear' with a 'New' button next to it. The main area on the right is for configuring the new parameter. It includes a 'Name' field with 'StartYear', a 'Description' field, a 'Required' checkbox that is checked, a 'Type' dropdown set to 'Decimal Number', a 'Suggested Values' dropdown set to 'Any value', and a 'Current Value' field with '2000'.

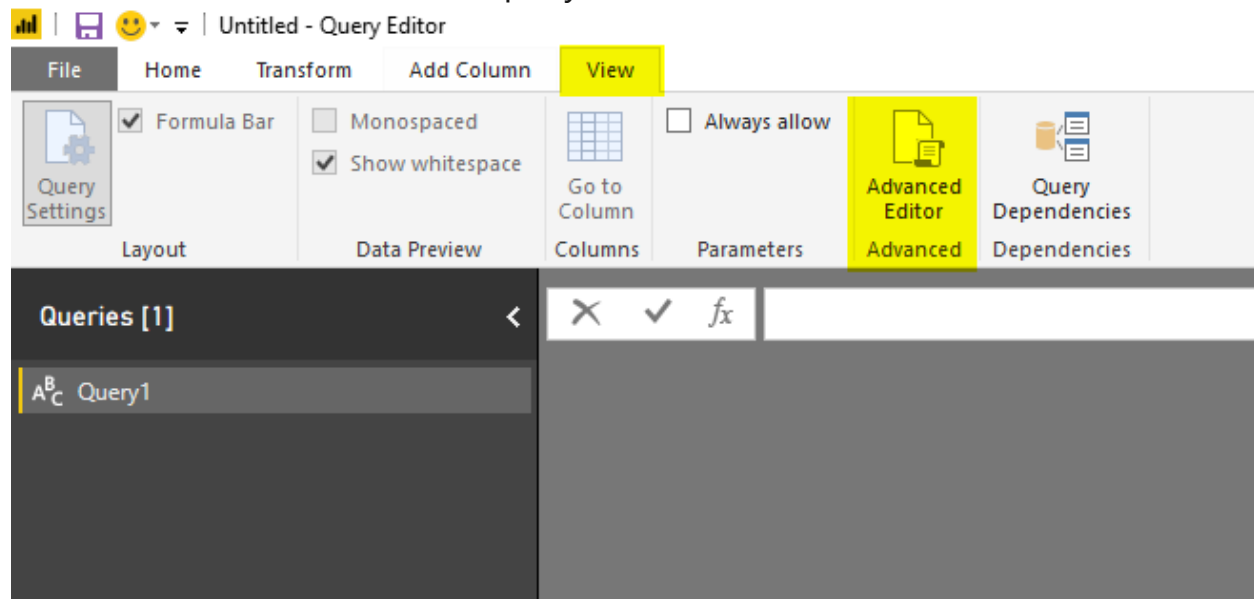
Then create another parameter for EndYear with the same configuration. Now you should have both in the list of queries;



You can always change the value of parameters easily later.

## First Step: Build the Base Query

Click on Query1 generated few steps ago. Then go to View Tab, and select Advanced Editor to see the M query in details;



Let's generate the start date based on start year. And let's consider the first day of January as the start date. This script will give you a start date:

```

1 let
2     StartDate = #date(StartYear,1,1)
3 in
4     StartDate

```

The same thing applies for the end date. However, end date would be 31st of December for the end year;

```

1 let
2     StartDate = #date(StartYear,1,1),
3     EndDate = #date(EndYear,12,31)
4 in
5     EndDate

```

To generate the base query we need to know how many days exists between these two dates;

Subtracting two dates from each other will return a Duration data type, then from Duration data type, we can fetch a number of days using Duration.Days Power Query function;

```

1 let
2     StartDate = #date(StartYear,1,1),
3     EndDate = #date(EndYear,12,31),
4     NumberOfDays = Duration.Days( EndDate - StartDate )
5 in
6     NumberOfDays

```

Now that we have the number of days between two days, we can use a generator to create a list of dates.

## **Date Generator**

Generators are functions that return a list. The List.Dates is a function that returns a list of dates from the start date for a number of occurrences based on duration. Here is the syntax of using this function;

List.Dates(<start date>,<occurrence>,<duration>)

In our example, start date and occurrence is clear. Duration would be #duration (1,0,0,0) meaning one day at a time. Other parameters of the

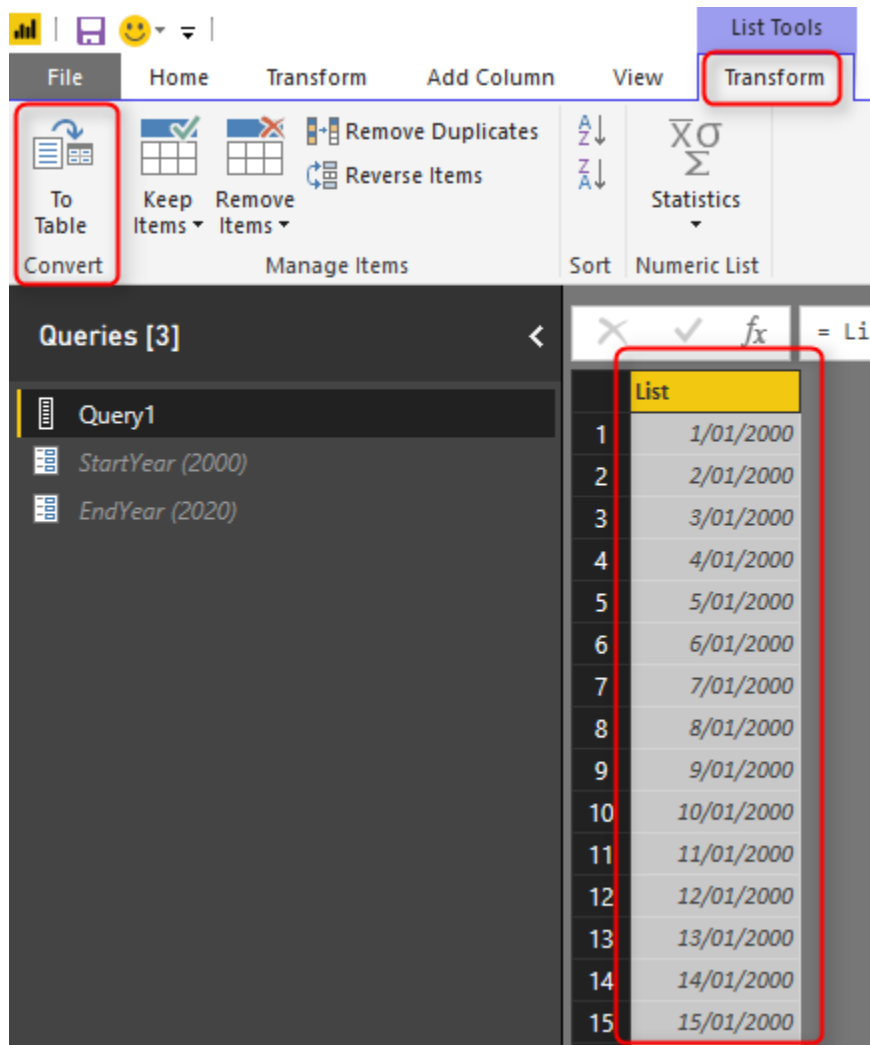
duration data type are an hour, minute, and second. So here is the change in query:

```
1 = List.Dates(StartDate, NumberOfDays+1, #duration(1,0,0,0))
```

The reason to add one to the number of days is that the difference is not including both dates. So adding one day will include the last date as well. So far the query is as below;

```
1 let
2   StartDate = #date(StartYear,1,1),
3   EndDate = #date(EndYear,12,31),
4   NumberOfDays = Duration.Days( EndDate - StartDate ),
5   Dates = List.Dates(StartDate, NumberOfDays+1, #duration(1,0,0,0))
6 in
7   Dates
```

the result would be a list of dates;



Because the result is in a list format, and List in Power Query only can have one column, we need to convert it to the table to be able to add extra columns to it. Converting to table is an easy transformation in List Tools -> Transform tab -> To Table. When converting to a table, you can choose delimiter and some configuration. Leave these configurations as default and click on OK.



## To Table

Create a table from a list of values.

Select or enter delimiter

None

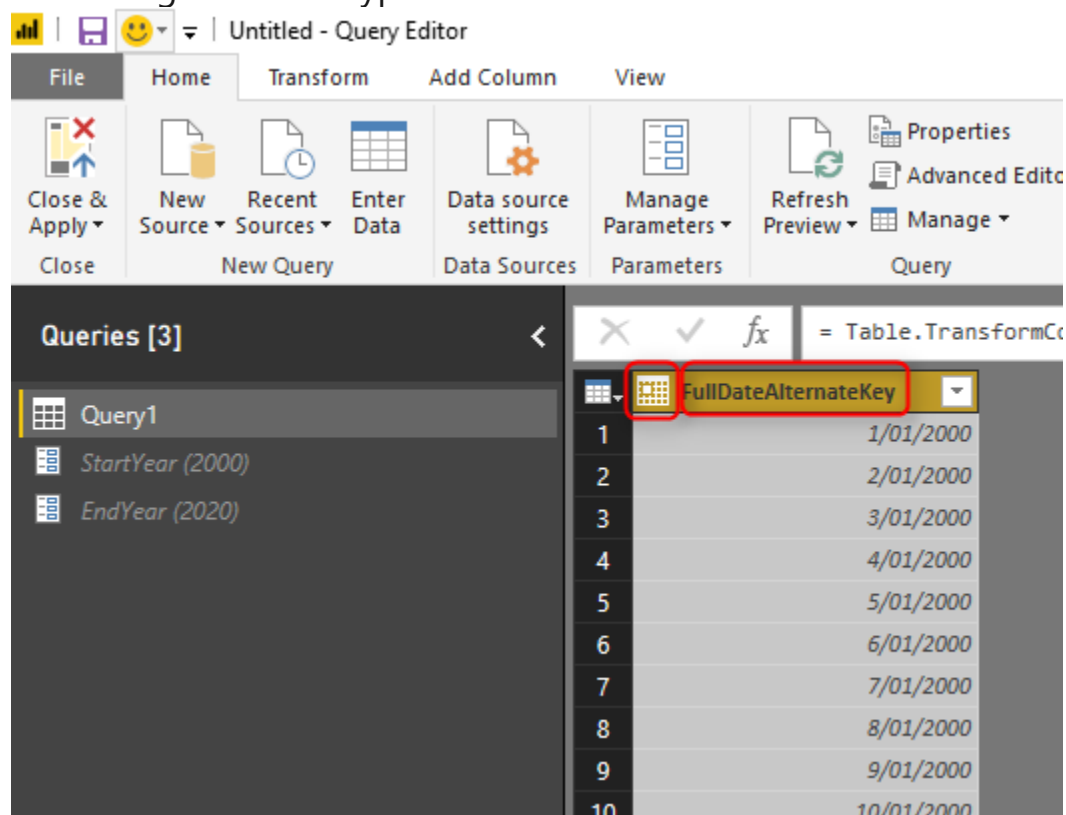
How to handle extra columns

Show as errors

OK

Cancel

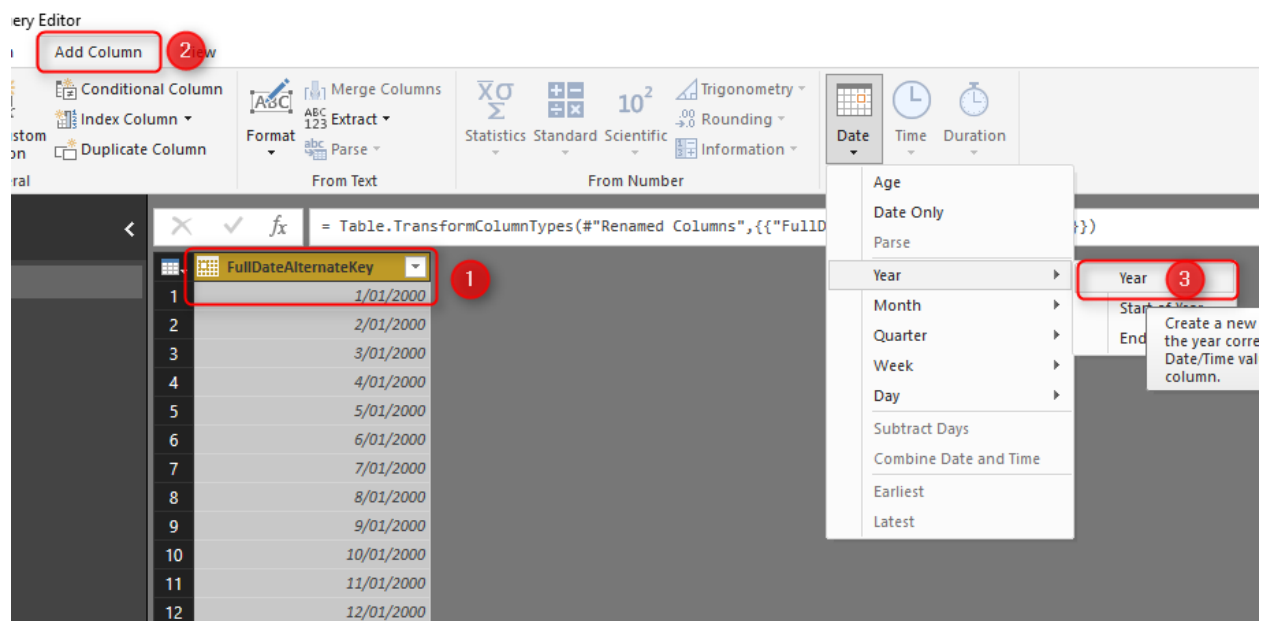
After converting it to the table, rename the column to FullDateAlternateKey, and change the data type of that to Date



## Next Step: Adding Extra Calendar Columns

Congratulations! You have done the first step. First step was creating the base query. Now that we have a column of all dates, then it is easy to add calendar columns to it. Click on FullDateAlternateKey column and then from Add

Columns Menu, under Date and Time Transformation, from Date section select Year



This add a Year column to the query:

	FullDateAlternateKey	1.2 Year
1	1/01/2000	2000
2	2/01/2000	2000
3	3/01/2000	2000
4	4/01/2000	2000
5	5/01/2000	2000
6	6/01/2000	2000
7	7/01/2000	2000
8	8/01/2000	2000
9	9/01/2000	2000
10	10/01/2000	2000
11	11/01/2000	2000
12	12/01/2000	2000
13	13/01/2000	2000
14	14/01/2000	2000
15	15/01/2000	2000
16	16/01/2000	2000
17	17/01/2000	2000
18	18/01/2000	2000
19	19/01/2000	2000

Continue the process to add all calendar columns you want. More you add in this step will give more slicing and dicing power later in Power BI. for creating every column; select FullDateAlternateKey column first, then from Add Column tab, date transformations, select the column you want.

- Query Editor

iform Add Column View

Conditional Column Index Column Duplicate Column

Format Merge Columns Extract Parse

Statistics Standard Scientific Trigonometry Rounding Information

General From Text From Number

fx = Table.AddColumn(#"Inserted Day of Year", "Day Name", each Date.DayOfWeekN

FullDateAlternateKey 1.2 Year 1.2 Month 1.2 Month Name 1.2 Quarter

1 1/01/2000 2000 1 January 1

2 2/01/2000 2000 1 January 1

3 3/01/2000 2000 1 January 1

4 4/01/2000 2000 1 January 1

5 5/01/2000 2000 1 January 1

6 6/01/2000 2000 1 January 1

7 7/01/2000 2000 1 January 1

8 8/01/2000 2000 1 January 1

9 9/01/2000 2000 1 January 1

10 10/01/2000 2000 1 January 1

11 11/01/2000 2000 1 January 1

12 12/01/2000 2000 1 January 1

13 13/01/2000 2000 1 January 1

Date Time Duration

Age Date Only Parse Year Month Quarter Week Day Subtract Days Combine Date and Time Earliest Latest

Month of Month 1.2 Day

Month Start of Month End of Month Days in Month Name of Month

Create a new column with the name of the corresponding value in the selected column

I would recommend all columns below to be added:

Month, Month Name, Quarter, Week of Year, Week of Month, Day, Day of Week, Day of Year, Day Name

Here is the final query so far:

	FullDateAlternateKey	1.2 Year	1.2 Month	1.2 Month Name	1.2 Quarter	1.2 Week of Year	1.2 Week of Month	1.2 Day	1.2 Day of Week	1.2 Day of Year	1.2 Day Name
1	1/01/2000	2000	1	January	1	1	1	1	6	1	Saturday
2	2/01/2000	2000	1	January	1	2	2	2	0	2	Sunday
3	3/01/2000	2000	1	January	1	2	2	3	1	3	Monday
4	4/01/2000	2000	1	January	1	2	2	4	2	4	Tuesday
5	5/01/2000	2000	1	January	1	2	2	5	3	5	Wednesday
6	6/01/2000	2000	1	January	1	2	2	6	4	6	Thursday
7	7/01/2000	2000	1	January	1	2	2	7	5	7	Friday
8	8/01/2000	2000	1	January	1	2	2	8	6	8	Saturday
9	9/01/2000	2000	1	January	1	3	3	9	0	9	Sunday
10	10/01/2000	2000	1	January	1	3	3	10	1	10	Monday
11	11/01/2000	2000	1	January	1	3	3	11	2	11	Tuesday
12	12/01/2000	2000	1	January	1	3	3	12	3	12	Wednesday
13	13/01/2000	2000	1	January	1	3	3	13	4	13	Thursday
14	14/01/2000	2000	1	January	1	3	3	14	5	14	Friday
15	15/01/2000	2000	1	January	1	3	3	15	6	15	Saturday
16	16/01/2000	2000	1	January	1	4	4	16	0	16	Sunday
17	17/01/2000	2000	1	January	1	4	4	17	1	17	Monday
18	18/01/2000	2000	1	January	1	4	4	18	2	18	Tuesday
19	19/01/2000	2000	1	January	1	4	4	19	3	19	Wednesday
20	20/01/2000	2000	1	January	1	4	4	20	4	20	Thursday
21	21/01/2000	2000	1	January	1	4	4	21	5	21	Friday
22	22/01/2000	2000	1	January	1	4	4	22	6	22	Saturday
23	23/01/2000	2000	1	January	1	5	5	23	0	23	Sunday
24	24/01/2000	2000	1	January	1	5	5	24	1	24	Monday
25	25/01/2000	2000	1	January	1	5	5	25	2	25	Tuesday
26	26/01/2000	2000	1	January	1	5	5	26	3	26	Wednesday
27	27/01/2000	2000	1	January	1	5	5	27	4	27	Thursday
28	28/01/2000	2000	1	January	1	5	5	28	5	28	Friday
29	29/01/2000	2000	1	January	1	5	5	29	6	29	Saturday

## Script

Here is the script of date dimension so far;

```

let
    StartDate = #date(StartYear,1,1),
    EndDate = #date(EndYear,12,31),
    NumberOfDays = Duration.Days( EndDate - StartDate ),
    Dates = List.Dates(StartDate, NumberOfDays+1, #duration(1,0,0,0)),
    #"Converted to Table" = Table.FromList(Dates,
1  Splitter.SplitByNothing(), null, null, ExtraValues.Error),
2  #"Renamed Columns" = Table.RenameColumns(#"Converted to
3  Table",{{"Column1", "FullDateAlternateKey"}}),
4  #"Changed Type" = Table.TransformColumnTypes(#"Renamed
5  Columns",{{"FullDateAlternateKey", type date}}),
6  #"Inserted Year" = Table.AddColumn(#"Changed Type", "Year", each
7  Date.Year([FullDateAlternateKey]), type number),
8  #"Inserted Month" = Table.AddColumn(#"Inserted Year", "Month", each
9  Date.Month([FullDateAlternateKey]), type number),
10  #"Inserted Month Name" = Table.AddColumn(#"Inserted Month",
11  "Month Name", each Date.MonthName([FullDateAlternateKey]), type
12  text),
13  #"Inserted Quarter" = Table.AddColumn(#"Inserted Month Name",
14  "Quarter", each Date.QuarterOfYear([FullDateAlternateKey]), type
15  number),
16  #"Inserted Week of Year" = Table.AddColumn(#"Inserted Quarter",
17  "Week of Year", each Date.WeekOfYear([FullDateAlternateKey]), type
18  number),
19  #"Inserted Week of Month" = Table.AddColumn(#"Inserted Week of
20  Year", "Week of Month", each
    Date.WeekOfMonth([FullDateAlternateKey]), type number),
    #"Inserted Day" = Table.AddColumn(#"Inserted Week of Month",
    "Day", each Date.Day([FullDateAlternateKey]), type number),
    #"Inserted Day of Week" = Table.AddColumn(#"Inserted Day", "Day of
    Week", each Date.DayOfWeek([FullDateAlternateKey]), type number),

```

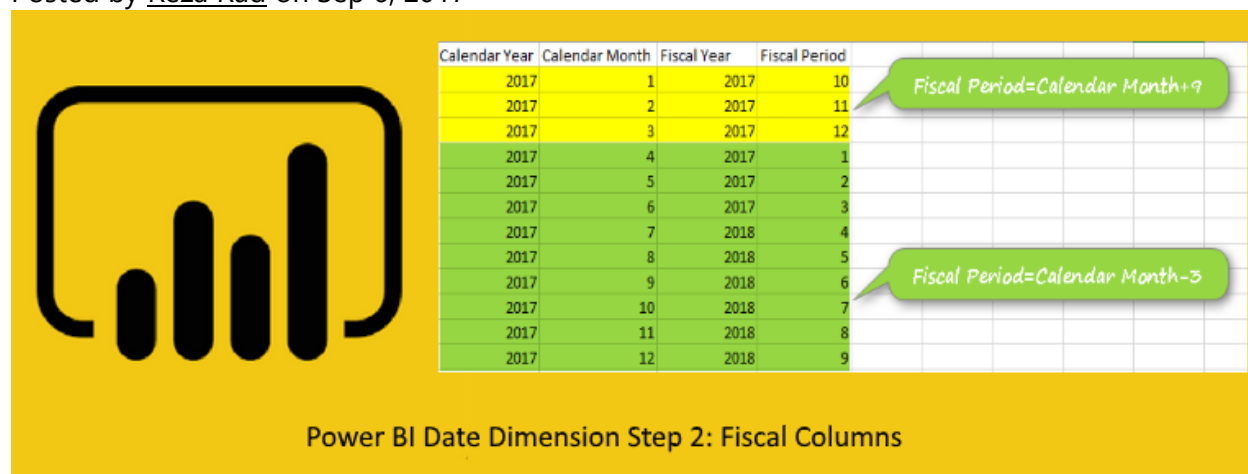
```
#"Inserted Day of Year" = Table.AddColumn(#"Inserted Day of Week",  
"Day of Year", each Date.DayOfYear([FullDateAlternateKey]), type  
number),  
#"Inserted Day Name" = Table.AddColumn(#"Inserted Day of Year",  
"Day Name", each Date.DayOfWeekName([FullDateAlternateKey]), type  
text)  
in  
#"Inserted Day Name"
```

## Summary

In this post, you have seen how easy it is to create a date dimension. The date dimension that we have created in this post is configurable via parameters. You can change parameters for the start date and end date and refresh the query to see a change in the result. This date dimension only had calendar columns, but in the next blog post I'll explain how to add fiscal columns to it, and in the blog post after that, I'll explain how to add public holidays to it. Stay tuned for next posts.

# Create a Date Dimension in Power BI in 4 Steps – Step 2: Fiscal Columns

Posted by [Reza Rad](#) on Sep 6, 2017



In [step 1 of creating date dimension](#), I explained how to create the base query with calendar columns. In this post, I will explain how to add fiscal columns calculated in the date dimension. Many business reports generate on a fiscal year, fiscal quarter, and fiscal period, so having fiscal columns is an important part of a date dimension. For creating the calculation for fiscal columns, I will use generic calculations that works perfectly for all scenarios (when you have any particular month as the start of the fiscal year). To learn more about Power BI; read [Power BI book from Rookie to Rock Star](#).

## Prerequisite

For this post, you need to complete [step 1 of creating date dimension](#).

## The parameter for Fiscal Year Star

Because I want the configuration to be dynamic, and I want to be able to change the fiscal year start to any month, and it works perfectly fine, so I am going to create a parameter for fiscal year start. In Query Editor, create a new parameter. Name it StarOfFiscalYear. This will hold the month value which is the start of fiscal year.

## Parameters

New

1<sup>2</sup>3 StartYear

1<sup>2</sup>3 EndYear

1<sup>2</sup>3 StartOfFiscalYear ✕

Name

StartOfFiscalYear

Description

The Month value which is start of fiscal year. for example; if start is July, then the value here should be 7.

☒ Required

Type

Decimal Number ▾

Suggested Values

Any value ▾

Current Value

7

After creating the parameter, then we can start applying the logic for every fiscal column as a new custom column.

## Creating Fiscal Columns

We do not have built-in functions to calculate fiscal columns. However, calculations are easy. We will go through it one by one, and you will see how the logic can be implemented.

### Fiscal Year

The first and easiest one is to calculate Fiscal year. Consider that months 7 (July) is the start of fiscal year, then we should have this;

Calendar Year	Calendar Month	Fiscal Year	
2017	1	2017	Start of Fiscal Year: July
2017	2	2017	
2017	3	2017	
2017	4	2017	
2017	5	2017	
2017	6	2017	
2017	7	2018	Fiscal Year=Calendar Year
2017	8	2018	
2017	9	2018	
2017	10	2018	
2017	11	2018	
2017	12	2018	

As you can see in the image above; June 2017 considered as the fiscal year 2017. However, July 2017 is part of the fiscal year 2018. So the simple logic can be like this:

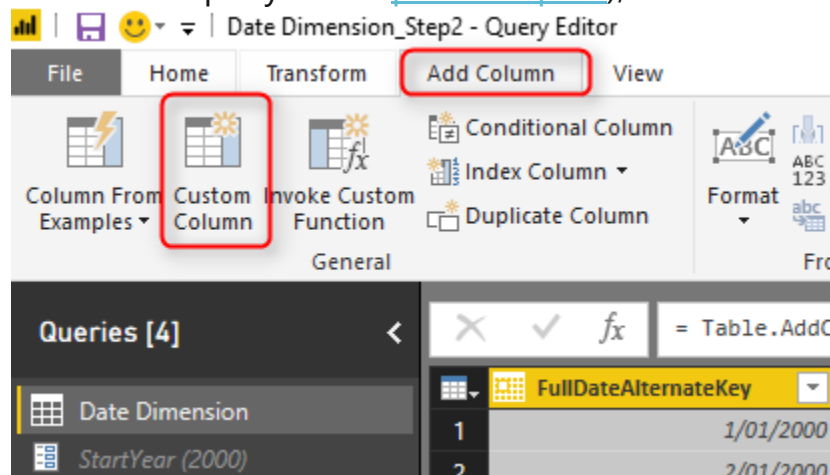
if (calendar month >= fiscal year start)

then fiscal year = calendar year

else fiscal year = calendar year + 1

This code is pseudo code. Don't write that exactly in M! Let's now implement it in M;

Create a Custom column in Date Dimension Query (I've explained how to create that query in the [previous post](#)),



In a custom column you can write your M code as below;



```

1 if [Month] >= StartOfFiscalYear
2 then [Year] + 1
3 else [Year]

```

## Custom Column

New column name

Fiscal Year

Custom column formula:

```

= if [Month] >= StartOfFiscalYear
then [Year] + 1
else [Year]

```

This is the M version of the same logic you've learned above.

After creating every custom column, make sure that you change its data type to the proper one. For this; change it to the Whole Number.

## Fiscal Period or Month

Calculating fiscal period or month is following the same IF expression logic.

But different calculation this time. Here are some samples;

Calendar Year	Calendar Month	Fiscal Year	Fiscal Period
2017	1	2017	7
2017	2	2017	8
2017	3	2017	9
2017	4	2017	10
2017	5	2017	11
2017	6	2017	12
2017	7	2018	1
2017	8	2018	2
2017	9	2018	3
2017	10	2018	4
2017	11	2018	5
2017	12	2018	6

*Fiscal Period = Calendar Month + 6*

*Fiscal Period = Calendar Month - 6*

So as an example; June 2017 is Fiscal period 12 of the fiscal year 2017. July 2017 is fiscal period 1 of the fiscal year 2018. As you can see the logic above; if calendar month is before fiscal year start month, then we add 6 to the calendar month to get the fiscal year, otherwise; we'll reduce six from it. What is 6? Where is this value coming from? Let's look at another example;

In this example; assume start of the fiscal year is in April (month 4);

Calendar Year	Calendar Month	Fiscal Year	Fiscal Period
2017	1	2017	10
2017	2	2017	11
2017	3	2017	12
2017	4	2017	1
2017	5	2017	2
2017	6	2017	3
2017	7	2018	4
2017	8	2018	5
2017	9	2018	6
2017	10	2018	7
2017	11	2018	8
2017	12	2018	9

*Fiscal Period = Calendar Month + 9*

*Fiscal Period = Calendar Month - 3*

Well, that made it more difficult. Didn't? Now we have two numbers; sometimes plus 9, sometimes minus 3! Don't be afraid the logic is simple. Here it is the logic;

If the calendar month is greater than or equal to fiscal year start, then we can say; Calendar Month – (Fiscal Year Start – 1)

if otherwise; we can say; Calendar Month + (12 – Fiscal Year Start + 1)

So, is how to implement it;

Create a new Custom Column for Fiscal Period. Put the code below there;

```
1 if [Month] > StartOfFiscalYear
2 then [Month] - (StartOfFiscalYear - 1)
3 else [Month] + (12 - StartOfFiscalYear + 1)
```

## Custom Column

New column name

Fiscal Period

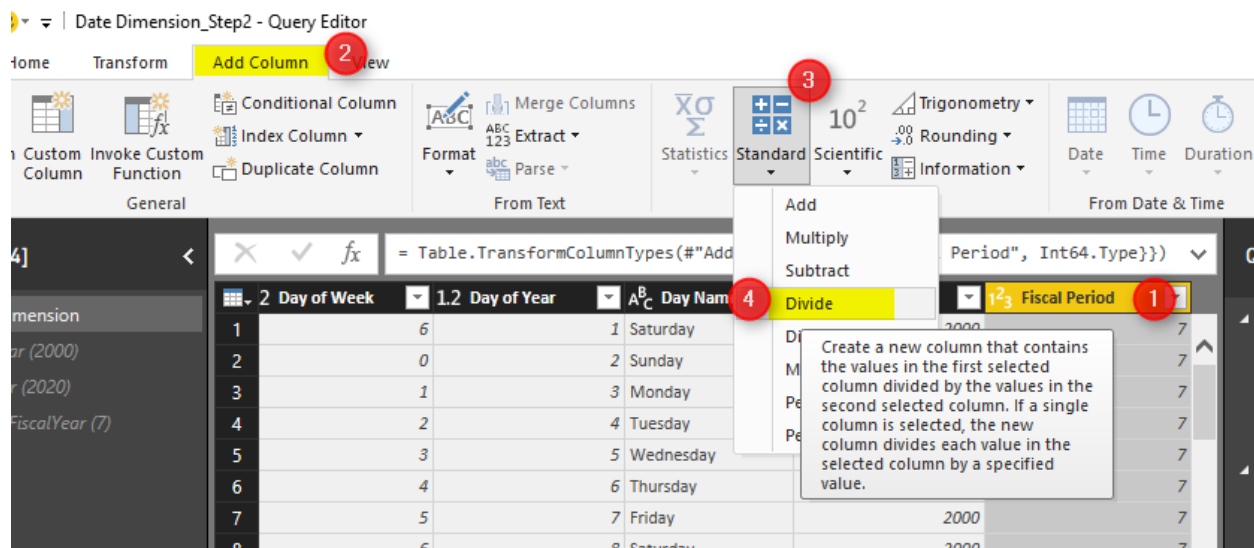
Custom column formula:

```
= if [Month] >= StartOfFiscalYear
then [Month] - (StartOfFiscalYear - 1)
else [Month] + (12 - StartOfFiscalYear + 1)
```

## Fiscal Quarter

Calculating the fiscal quarter should be easy when you have the fiscal period. You can start with dividing by 3. Then Round the value Up to get the fiscal quarter. Here is how to do it;

Click on Fiscal Period Column (calculated from the previous step), then go to Add Column, from Number Column, select Divide under Standard;



Divide it by 3 (number of months in each quarter);

### Divide

Enter a number by which to divide each value in the column.

Value

ABC 123

OK

Cancel

The result will be a column with values that in some cells have decimal places. We need to Round Up this. Click on this column, go to Transform tab, under Number transformation from Rounding, select Round-Up.

Power Query Editor: Date Dimension\_Step2 - Query Editor

Transform Tab: Add Column (2), Split Column (3), Format (4), Rounding (1)

Formula Bar: = Table.AddColumn(#"Changed Type2", "Inserted Division", each [Fiscal

of Year	Day Name	Fiscal Year	Fiscal Period	Inserted Division
1	1 Saturday	2000	7	2.333333333
2	2 Sunday	2000	7	2.333333333

Note that this last transformation is coming from Transform Tab to avoid extra column creation. After this change, then rename the column to Fiscal Quarter;

Day Name	Fiscal Year	Fiscal Period	Fiscal Quarter
Saturday	2000	7	3
Sunday	2000	7	3
Monday	2000	7	3
Tuesday	2000	7	3
Wednesday	2000	7	3
Thursday	2000	7	3
Friday	2000	7	3
Saturday	2000	7	3
Sunday	2000	7	3
Monday	2000	7	3
Tuesday	2000	7	3
Wednesday	2000	7	3
Thursday	2000	7	3
Friday	2000	7	3
Saturday	2000	7	3
Sunday	2000	7	3
Monday	2000	7	3
Tuesday	2000	7	3
Wednesday	2000	7	3
Thursday	2000	7	3
Friday	2000	7	3
Saturday	2000	7	3
Sunday	2000	7	3
Monday	2000	7	3
Tuesday	2000	7	3
Wednesday	2000	7	3

## Fiscal Week

Fiscal week calculation is a bit different from other calculations. Not all business work on fiscal week basis, for example, a business on football league or sports league would work on a weekly basis. This has some varieties. One of them is that a business can say their fiscal year starts at the very first Monday of July each year. Another company might say they have a pre-defined date as the start of fiscal year. So, it depends on the business, the logic is different. The example that I showed you so far is a generic example. Because Fiscal week calculation itself is quite a bit off topic. I'm going to explain that in another post.

## Code of this example

The whole script of M for Date dimension in this example is as below;

```

1 let
2     StartDate = #date(StartYear,1,1),
3     EndDate = #date(EndYear,12,31),
4     NumberOfDays = Duration.Days( EndDate - StartDate ),
5     Dates = List.Dates(StartDate, NumberOfDays+1, #duration(1,0,0,0)),
6     #"Converted to Table" = Table.FromList(Dates,
7     Splitter.SplitByNothing(), null, null, ExtraValues.Error),
8     #"Renamed Columns" = Table.RenameColumns(#"Converted to
9     Table",{{"Column1", "FullDateAlternateKey"}}),
10    #"Changed Type" = Table.TransformColumnTypes(#"Renamed
11    Columns",{{"FullDateAlternateKey", type date}}),
12    #"Inserted Year" = Table.AddColumn(#"Changed Type", "Year", each
13    Date.Year([FullDateAlternateKey]), type number),
14    #"Inserted Month" = Table.AddColumn(#"Inserted Year", "Month", each
15    Date.Month([FullDateAlternateKey]), type number),
16    #"Inserted Month Name" = Table.AddColumn(#"Inserted Month",
17    "Month Name", each Date.MonthName([FullDateAlternateKey]), type
18    text),

```

```

19  #"Inserted Quarter" = Table.AddColumn(#"Inserted Month Name",
20  "Quarter", each Date.QuarterOfYear([FullDateAlternateKey]), type
21  number),
22  #"Inserted Week of Year" = Table.AddColumn(#"Inserted Quarter",
23  "Week of Year", each Date.WeekOfYear([FullDateAlternateKey]), type
24  number),
25  #"Inserted Week of Month" = Table.AddColumn(#"Inserted Week of
26  Year", "Week of Month", each
27  Date.WeekOfMonth([FullDateAlternateKey]), type number),
28  #"Inserted Day" = Table.AddColumn(#"Inserted Week of Month",
29  "Day", each Date.Day([FullDateAlternateKey]), type number),
30  #"Inserted Day of Week" = Table.AddColumn(#"Inserted Day", "Day of
31  Week", each Date.DayOfWeek([FullDateAlternateKey]), type number),
    #"Inserted Day of Year" = Table.AddColumn(#"Inserted Day of Week",
    "Day of Year", each Date.DayOfYear([FullDateAlternateKey]), type
    number),
    #"Inserted Day Name" = Table.AddColumn(#"Inserted Day of Year",
    "Day Name", each Date.DayOfWeekName([FullDateAlternateKey]), type
    text),
    #"Added Custom" = Table.AddColumn(#"Inserted Day Name", "Fiscal
    Year", each if [Month] >= StartOfFiscalYear
    then [Year] + 1
    else [Year]),
    #"Changed Type1" = Table.TransformColumnTypes(#"Added
    Custom",{{"Fiscal Year", Int64.Type}}),
    #"Added Custom1" = Table.AddColumn(#"Changed Type1", "Fiscal
    Period", each if [Month] >= StartOfFiscalYear
    then [Month] - (StartOfFiscalYear - 1)
    else [Month] + (12 - StartOfFiscalYear + 1)),
    #"Changed Type2" = Table.TransformColumnTypes(#"Added
    Custom1",{{"Fiscal Period", Int64.Type}}),
    #"Inserted Division" = Table.AddColumn(#"Changed Type2", "Inserted
    Division", each [Fiscal Period] / 3, type number),

```

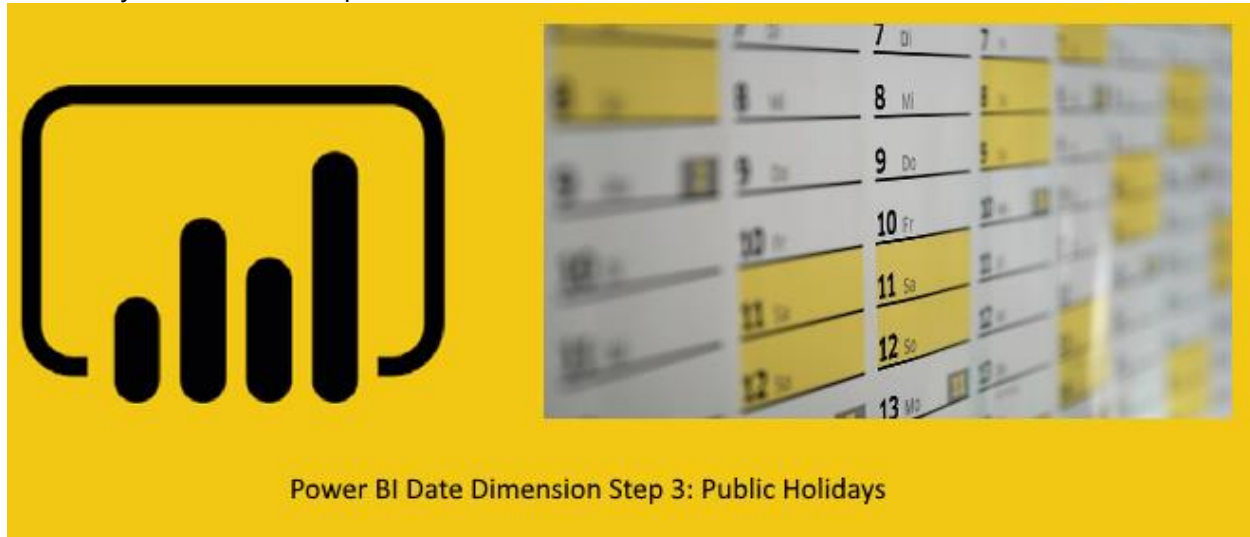
```
#"Rounded Up" = Table.TransformColumns(#"Inserted  
Division",{{"Inserted Division", Number.RoundUp, Int64.Type}},  
#"Renamed Columns1" = Table.RenameColumns(#"Rounded  
Up",{{"Inserted Division", "Fiscal Quarter"}})  
in  
#"Renamed Columns1"
```

## Summary

In summary, you've seen how you can add a fiscal year, period, and a quarter to a date dimension. Examples used in this post are generic. You can use these calculations in any date dimension. Fiscal week calculation is different and is not generic for everyone; I'll write a blog post separately for that. I will talk about adding public holidays to this date dimension in the next post. So stay tuned.

# Create a Date Dimension in Power BI in 4 Steps – Step 3: Public Holidays

Posted by [Reza Rad](#) on Sep 11, 2017



In the last two steps, you learned how you could create a date dimension that has [calendar columns](#), and [fiscal columns](#). In this step, we are going to look at how Power Query can be leveraged to fetch public holidays live from APIs and add that information into the date dimension. This step shows the importance of Power Query compared to DAX for generating a date dimension. There are lots of data analysis that can be done based on public holidays, and that can be done when public holidays information is up-to-date in the date dimension. If you like to learn more about Power BI; read the [Power BI book from Rookie to Rock Star](#).

## Prerequisite

Prerequisite of this step is to have the result of the previous two steps ready;  
[Step 1 – Generate Date Dimension with Calendar Columns](#)  
[Step 2 – Add Fiscal Columns](#)



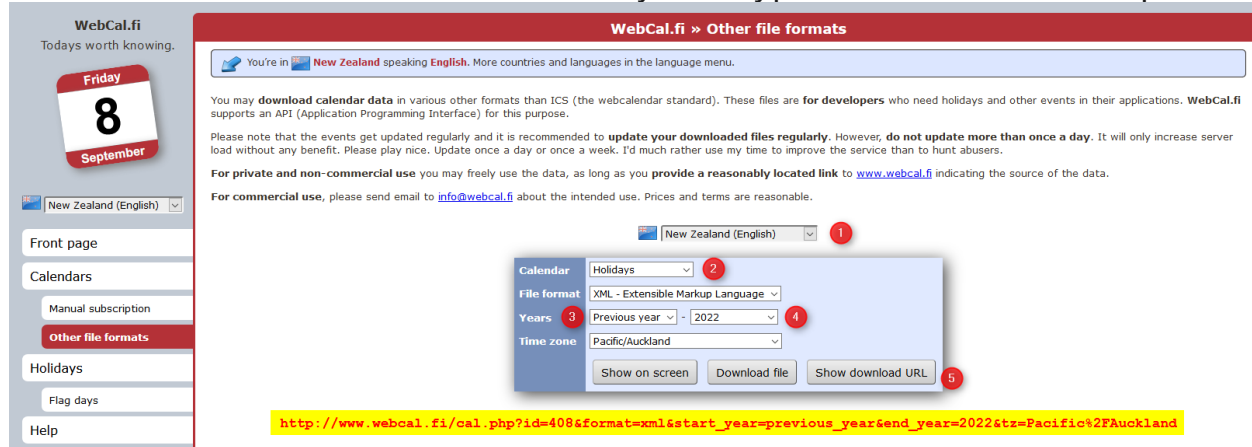
## Live API for Public Holidays

To get public holidays live, you first need an API that is giving you up-to-date information. Some web pages has the list of public holidays. I have already explained in another blog post how to use a web page and query public holidays from there. That method uses custom functions as well, [here](#) you can read about that.

The method of reading data from a web page has an issue already; Web.Page function from Power Query is used to pull data from that page, and this function needs a gateway configuration to work. There is another function Xml.Document that can work even without the gateway. So because of this reason, we'll use Xml.Document and get data from an API that provides the result set as XML.

[WebCal.fi](#) is a great free website with calendars for 36 countries which I do recommend for this example. This website provides the calendars through XML format. There are other websites that may give you the calendar details through a paid subscription. However, this website is a great free one which can be used for this example. WebCal.fi is created by [User Point Inc.](#)

There is a [page on this website](#) that gives you the access to the calendar in XML format. You can choose the country, and type of calendar and the period;



WebCal.fi  
Today's worth knowing.

You're in **New Zealand** speaking **English**. More countries and languages in the language menu.

You may **download calendar data** in various other formats than ICS (the webcalendar standard). These files are **for developers** who need holidays and other events in their applications. WebCal.fi supports an API (Application Programming Interface) for this purpose.

Please note that the events get updated regularly and it is recommended to **update your downloaded files regularly**. However, **do not update more than once a day**. It will only increase server load without any benefit. Please play nice. Update once a day or once a week. I'd much rather use my time to improve the service than to hunt abusers.

**For private and non-commercial use** you may freely use the data, as long as you **provide a reasonably located link** to [www.webcal.fi](#) indicating the source of the data.

**For commercial use**, please send email to [info@webcal.fi](mailto:info@webcal.fi) about the intended use. Prices and terms are reasonable.

Calendar: Holidays  
File format: XML - Extensible Markup Language  
Years: Previous year - 2022  
Time zone: Pacific/Auckland

Show on screen Download file Show download URL

[http://www.webcal.fi/cal.php?id=408&format=xml&start\\_year=previous\\_year&end\\_year=2022&tz=Pacific%2FAuckland](http://www.webcal.fi/cal.php?id=408&format=xml&start_year=previous_year&end_year=2022&tz=Pacific%2FAuckland)

In the page above you can choose a country, calendar, file format, years (from and to), and then click on Show download URL. The URL you see highlighted above is for public holidays of New Zealand from the previous year to 2022. You can generate it for your country and date period. You can even change the previous\_year in the URL above to 2010, and you would still get the same result.

This is the URL I am going to use in this example;

[www.webcal.fi/cal.php?id=408&format=xml&start\\_year=2010&end\\_year=2022&tz=Pacific%2FAuckland](http://www.webcal.fi/cal.php?id=408&format=xml&start_year=2010&end_year=2022&tz=Pacific%2FAuckland)

browsing this URL will download an XML file with data as below;

```
<?xml version="1.0" encoding="utf-8"?>
<events>
  <event>
    <date>2015-01-01</date>
    <name>New Year's Day</name>
    <url>https://en.wikipedia.org/wiki/New_Year%27s_Day</url>
    <description>New Year's Day is observed on January 1, the first day of
the year on the modern Gregorian calendar as well as the Julian
calendar used in ancient Rome. With most countries using the Gregorian
calendar as their main calendar, New Year's Day is the closest thing to
being the world's only truly global public holiday, often celebrated
with fireworks at the stroke of midnight as the new year starts.
January 1 on the Julian calendar currently corresponds to January 14 on
the Gregorian calendar, and it is on that date that followers of some
of the Eastern Orthodox churches celebrate the New Year.</description>
  </event>
  <event>
    <date>2015-01-02</date>
    <name>Day after New Year's Day</name>
  </event>
  <event>
    . . . . .
  </event>
```

Now let's have a look at this data from Power BI

## Get Data from Web API

Get Data from Web, and use the URL you got from above;

## From Web

☒ Basic ☐ Advanced

URL

OK

Cancel

Select the result set and bring it into Power Query;

	date	A <sup>B</sup> <sub>C</sub> name	A <sup>B</sup> <sub>C</sub> url	A <sup>B</sup> <sub>C</sub> description	i <sup>2</sup> <sub>3</sub> flag_day	i <sup>2</sup> <sub>3</sub> age	A <sup>B</sup> <sub>C</sub> alternate_names
1	1/01/2015	New Year's Day	https://en.wikipedia.org/wiki/New_Year%27s_Day	New Year's Day is observed on January 1, the first day of the year on t...	null	null	null
2	2/01/2015	Day after New Year's Day			null	null	null
3	6/02/2015	Waitangi Day	https://en.wikipedia.org/wiki/Waitangi_Day		null	1	175
4	3/04/2015	Good Friday	https://en.wikipedia.org/wiki/Good_Friday	Good Friday (from the senses pious, holy of the word "good"), is a reli...	null	null	null
5	5/04/2015	Easter	https://en.wikipedia.org/wiki/Easter	Easter (Old English: Eostre; Greek: Πάσχα, Paskha; Aramaic: כּנוּס פּאס...	null	null	null
6	6/04/2015	Easter Monday	https://en.wikipedia.org/wiki/Easter_Monday	Easter Monday is the day after Easter Sunday and is celebrated as a ho...	null	null	null
7	25/04/2015	Anzac Day	https://en.wikipedia.org/wiki/Anzac_Day	Anzac Day is a national day of remembrance in Australia and New Zeal...	1	100	
8	1/06/2015	Queen's Birthday	https://en.wikipedia.org/wiki/Queen%27s_Birthday	The Queen's (King's) Official Birthday is the selected day on which the ...	null	null	null
9	26/10/2015	Labour Day	https://en.wikipedia.org/wiki/Labour_Day	Labour Day or Labor Day is an annual holiday to celebrate the econom...	1	null	null
10	25/12/2015	Christmas	https://en.wikipedia.org/wiki/Christmas	Christmas or Christmas Day (Old English: Crīstesmasse, literally "Chris...	null	null	null
11	26/12/2015	Boxing Day	https://en.wikipedia.org/wiki/Boxing_Day	Boxing Day is traditionally a day following Christmas when wealthy pe...	null	null	Proclamation Day; St. Stephen's Day
12	28/12/2015	Boxing Day Public Holiday	https://en.wikipedia.org/wiki/Boxing_Day	In South Africa, Boxing Day was renamed Day of Goodwill in 1994. In Ire...	null	null	Proclamation Day; St. Stephen's Day
13	1/01/2016	New Year's Day	https://en.wikipedia.org/wiki/New_Year%27s_Day	New Year's Day is observed on January 1, the first day of the year on t...	null	null	null
14	4/01/2016	Day after New Year's Day			null	null	null
15	6/02/2016	Waitangi Day	https://en.wikipedia.org/wiki/Waitangi_Day		null	1	176
16	25/03/2016	Good Friday	https://en.wikipedia.org/wiki/Good_Friday	Good Friday (from the senses pious, holy of the word "good"), is a reli...	null	null	null
17	27/03/2016	Easter	https://en.wikipedia.org/wiki/Easter	Easter (Old English: Eostre; Greek: Πάσχα, Paskha; Aramaic: כּנוּס פּאס...	null	null	null
18	28/03/2016	Easter Monday	https://en.wikipedia.org/wiki/Easter_Monday	Easter Monday is the day after Easter Sunday and is celebrated as a ho...	null	null	null
19	25/04/2016	Anzac Day	https://en.wikipedia.org/wiki/Anzac_Day	Anzac Day is a national day of remembrance in Australia and New Zeal...	1	101	
20	6/06/2016	Queen's Birthday	https://en.wikipedia.org/wiki/Queen%27s_Birthday	The Queen's (King's) Official Birthday is the selected day on which the ...	null	null	null
21	24/10/2016	Labour Day	https://en.wikipedia.org/wiki/Labour_Day	Labour Day or Labor Day is an annual holiday to celebrate the econom...	1	null	null
22	25/12/2016	Christmas	https://en.wikipedia.org/wiki/Christmas	Christmas or Christmas Day (Old English: Crīstesmasse, literally "Chris...	null	null	null
23	26/12/2016	Boxing Day	https://en.wikipedia.org/wiki/Boxing_Day	Boxing Day is traditionally a day following Christmas when wealthy pe...	null	null	Proclamation Day; St. Stephen's Day
24				In South Africa, Boxing Day was renamed Day of Goodwill in 1994. In Ire...			

The table includes everything in it. You can keep the first two columns which are date and the name of public holidays. Remove other columns;

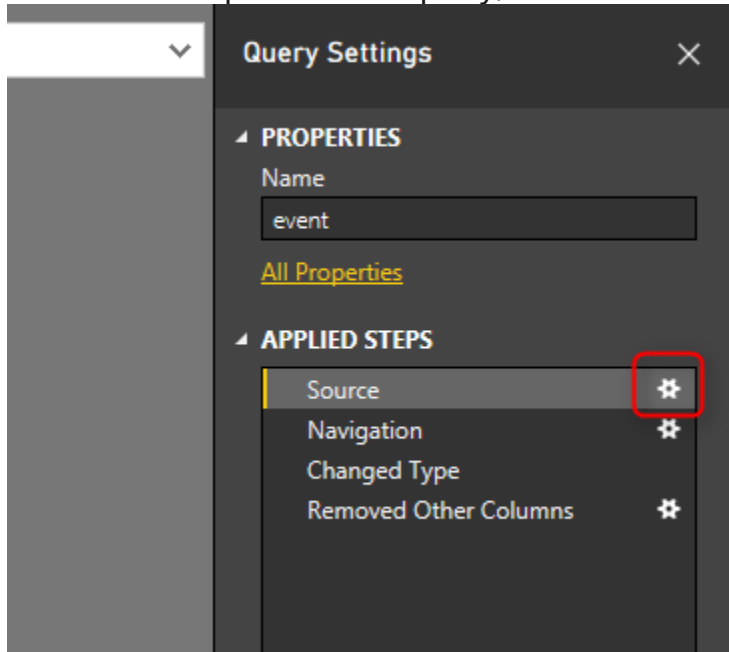
	date	A <sup>B</sup> <sub>C</sub> name	A <sup>B</sup> <sub>C</sub> url
1	1/01/2015	New Year's Day	
2	2/01/2015	Day after New Year's Day	
3	6/02/2015	Waitangi Day	
4	3/04/2015	Good Friday	
5	5/04/2015	Easter	
6	6/04/2015	Easter Monday	
7	25/04/2015	Anzac Day	
8	1/06/2015	Queen's Birthday	

Also if you note in the first step of this script; the function used is Xml.Tables, not Web.Page. So this function won't need a gateway to work.

Let's make this public holidays parametric;

## Add Parameters to Public Holidays URL

in the first step for event query, click on setting for Source step.



select Advanced mode of the get data from web step, and change the URL as below;

✕

## XML

☐ Basic
 ☒ Advanced

URL parts ⓘ

/cal.php?id=408&format=xml&start_year=	...
2010	
&end_year=	
2022	
&tz=Pacific%2FAuckland	

Add part

URL preview

www.webcal.fi/cal.php?id=408&format=xml&start\_year=2010&end\_year=2022&tz=Pacific%2FAuckland

Open file as

Xml Tables ▼

File origin

--None-- ▼

OK
Cancel

As you can see I split the URL into five parts;

- everything before the first parameter
- the first parameter
- the text between first and second parameter
- the second parameter
- text after the second parameter

The URL Preview is still the same at the end. Now we can replace "2010" and "2022" with values from parameters. Click on OK, and then go to Advanced Editor.

In advanced Editor make below changes;

1 let

2 Source =

3 Xml.Tables(Web.Contents("www.webcal.fi/cal.php?id=408&format=xml&start\_year=" & Text.From(StartYear) & "&end\_year=" & Text.From(EndYear) & "&tz=Pacific%2FAuckland")),

```

6  Table0 = Source{0}[Table],
7  #"Changed Type" = Table.TransformColumnTypes(Table0,{{"date", type
    date}, {"name", type text}, {"url", type text}, {"description", type text},
    {"flag_day", Int64.Type}, {"age", Int64.Type}, {"alternate_names", type text}}),
    #"Removed Other Columns" = Table.SelectColumns(#"Changed
    Type",{"date", "name"})
in
    #"Removed Other Columns"

```

Note that in the second line; Text.From(StartYear) replaced the "2010", and Text.From(EndYear) replaced the "2022". Now the query is parametric.

```

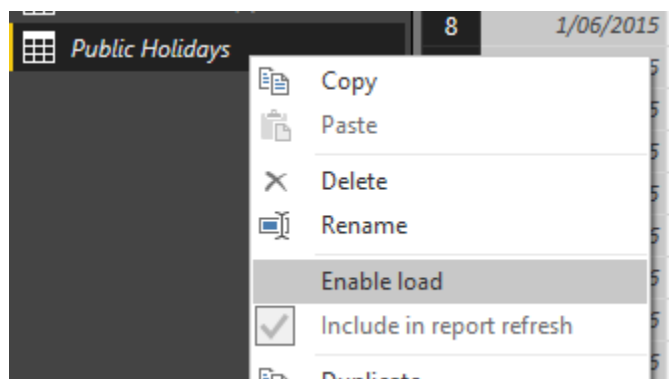
let
    Source = Xml.Tables(Web.Contents("www.webcal.fi/cal.php?id=408&format=xml&start_year="
        & Text.From(StartYear)
        & "&end_year="
        & Text.From(EndYear)
        & "&tz=Pacific%2FAuckland")),
    Table0 = Source{0}[Table],
    #"Changed Type" = Table.TransformColumnTypes(Table0,{{"date", type date}, {"name", type text}, {"
    #"Removed Other Columns" = Table.SelectColumns(#"Changed Type",{"date", "name"})
in
    #"Removed Other Columns"[

```

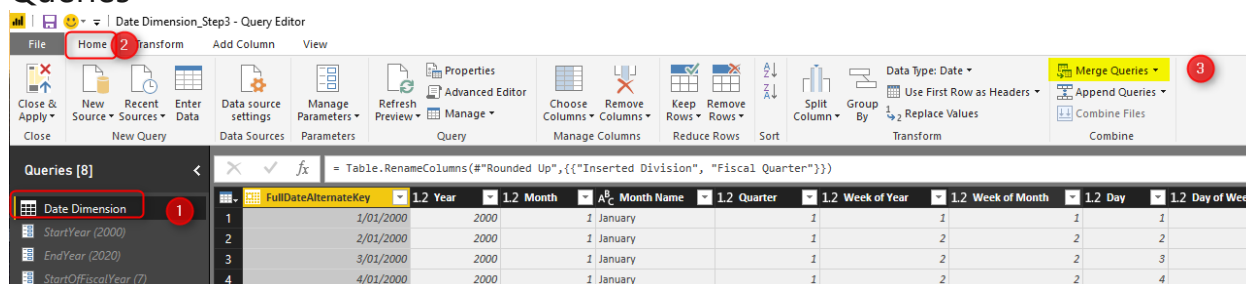
after clicking OK on the advanced editor, you should see now result based on values of parameters.

## Merge Date Dimension with Public Holidays

Now that we've got a list of public holidays, we can merge that back with date dimension. Before that, rename the event query to Public Holidays, and set it to Uncheck enable load. To know why you can [read this post](#).



Then click on Date Dimension, and from Home tab at the end select Merge Queries



in the Merge Join window; select second table as Public Holidays (1), then choose FullDateAlternateKey from date dimension (2), and select date column from Public Holidays (3), make sure join kind is left outer join (4), and click on OK.



## Merge

Select a table and matching columns to create a merged table.

Date Dimension



FullDateAlternateKey	Year	Month	Month Name	Quarter	Week of Year	Week of Month	Day
1/01/2000	2000	1	January	1	1	1	1
2/01/2000	2000	1	January	1	2	2	2
3/01/2000	2000	1	January	1	2	2	3
4/01/2000	2000	1	January	1	2	2	4
5/01/2000	2000	1	January	1	2	2	5


Public Holidays



date	name
1/01/2015	New Year's Day
2/01/2015	Day after New Year's Day
6/02/2015	Waitangi Day
3/04/2015	Good Friday
5/04/2015	Easter

Join Kind

Left Outer (all from first, matching from second)

 The selection has matched 70 out of the first 7671 rows.

OK

Cancel

To learn more about [Merge Join and different types of it, read this blog post](#).

After merging two queries, you will see a result, which has a table in every cell; you can expand that cell easily with clicking on the expand button at the top right corner of that column.



Public Holidays",JoinKind.LeftOuter)

Day Name	Fiscal Year	Fiscal Period	Fiscal Quarter	Public Holidays
Saturday	2000			
Sunday	2000			
Monday	2000			
Tuesday	2000			
Wednesday	2000			
Thursday	2000			
Friday	2000			
Saturday	2000			
Sunday	2000			
Monday	2000			
Tuesday	2000			
Wednesday	2000			
Thursday	2000			
Friday	2000			

Search Columns to Expand

Expand Aggregate

(Select All Columns)

☐ date

☒ name

☐ Use original column name as prefix

OK Cancel

Select name only. Because we do have a date already in the table. We need to know public holiday information which is in the name column.

["name", "Public Holiday"]}}

Day of Week	Day of Year	Day Name	Fiscal Year	Fiscal Period	Fiscal Quarter	Public Holiday
1	4	Thursday	2015	7	3	New Year's Day
2	5	Friday	2015	7	3	Day after New Year's Day
3	6	Saturday	2015	7	3	null
4	0	Sunday	2015	7	3	null
5	1	Monday	2015	7	3	null
6	2	Tuesday	2015	7	3	null
7	3	Wednesday	2015	7	3	null
8	4	Thursday	2015	7	3	null
9	5	Friday	2015	7	3	null
10	6	Saturday	2015	7	3	null
11	0	Sunday	2015	7	3	null
12	1	Monday	2015	7	3	null

You can also add another column at the end that has a true or false value of is that date a public holiday or not. This would be simply a custom column with a conditional expression;

Date Dimension\_Step3 - Query Editor

File Home Transform Add Column View

Column From Examples Custom Invoke Custom Function General

Conditional Column Index Column Duplicate Column

Forma

Queries [81]

fx = Table

the condition can be easily written with an “if” statement as below;

## Custom Column

New column name

Is Public Holiday

Custom column formula:

=if [Public Holiday] = null then false else true

The final result got everything in it;

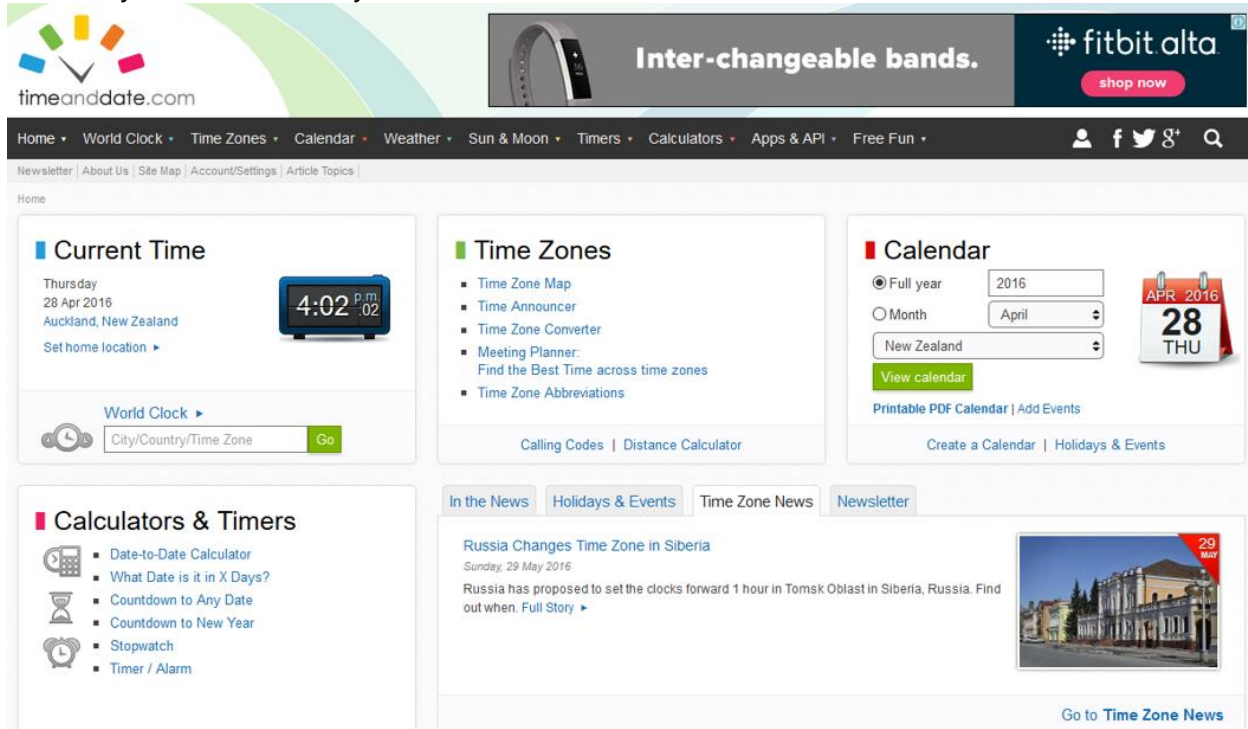
Year	Month	Month Name	Quarter	Week of Year	Week of Month	Day	Day of Week	Day of Year	Day Name	Fiscal Year	Fiscal Period	Fiscal Quarter	Public Holiday	Is Public Holiday
2000	1	January	1	1	1	1	6	1	Saturday	2000	1	1	3	FALSE
2000	1	January	1	2	2	2	0	2	Sunday	2000	1	1	3	FALSE
2000	1	January	1	3	3	3	1	3	Monday	2000	1	1	3	FALSE
2000	1	January	1	4	4	4	2	4	Tuesday	2000	1	1	3	FALSE
2000	1	January	1	5	5	5	3	5	Wednesday	2000	1	1	3	FALSE
2000	1	January	1	6	6	6	4	6	Thursday	2000	1	1	3	FALSE
2000	1	January	1	7	7	7	5	7	Friday	2000	1	1	3	FALSE
2000	1	January	1	8	8	8	6	8	Saturday	2000	1	1	3	FALSE
2000	1	January	1	9	9	9	0	9	Sunday	2000	1	1	3	FALSE
2000	1	January	1	10	10	10	1	10	Monday	2000	1	1	3	FALSE
2000	1	January	1	11	11	11	2	11	Tuesday	2000	1	1	3	FALSE
2000	1	January	1	12	12	12	3	12	Wednesday	2000	1	1	3	FALSE
2000	1	January	1	13	13	13	4	13	Thursday	2000	1	1	3	FALSE
2000	1	January	1	14	14	14	5	14	Friday	2000	1	1	3	FALSE
2000	1	January	1	15	15	15	6	15	Saturday	2000	1	1	3	FALSE
2000	1	January	1	16	16	16	0	16	Sunday	2000	1	1	3	FALSE
2000	1	January	1	17	17	17	1	17	Monday	2000	1	1	3	FALSE
2000	1	January	1	18	18	18	2	18	Tuesday	2000	1	1	3	FALSE
2000	1	January	1	19	19	19	3	19	Wednesday	2000	1	1	3	FALSE
2000	1	January	1	20	20	20	4	20	Thursday	2000	1	1	3	FALSE
2000	1	January	1	21	21	21	5	21	Friday	2000	1	1	3	FALSE
2000	1	January	1	22	22	22	6	22	Saturday	2000	1	1	3	FALSE
2000	1	January	1	23	23	23	0	23	Sunday	2000	1	1	3	FALSE
2000	1	January	1	24	24	24	1	24	Monday	2000	1	1	3	FALSE
2000	1	January	1	25	25	25	2	25	Tuesday	2000	1	1	3	FALSE
2000	1	January	1	26	26	26	3	26	Wednesday	2000	1	1	3	FALSE
2000	1	January	1	27	27	27	4	27	Thursday	2000	1	1	3	FALSE
2000	1	January	1	28	28	28	5	28	Friday	2000	1	1	3	FALSE
2000	1	January	1	29	29	29	6	29	Saturday	2000	1	1	3	FALSE
2000	1	January	1	30	30	30	0	30	Sunday	2000	1	1	3	FALSE
2000	1	January	1	31	31	31	1	31	Monday	2000	1	1	3	FALSE
2000	2	February	2	1	1	1	2	32	Tuesday	2000	2	1	3	FALSE
2000	2	February	2	2	2	2	3	33	Wednesday	2000	2	1	3	FALSE

## Summary

In Summary, you have learned so far through three steps how you create a full general purpose date dimension with Power BI and Power Query. This date Dimension has [calendar columns](#) in it and [fiscal columns](#), and in this post, you’ve learned how to fetch live public holidays into this. Anytime this date dimension gets refreshed the new public holiday’s data will be loaded into it. So you don’t need to be worried about the data to be up-to-date.

# Power Query Not for BI: Event Date and Time Scheduler – Part 1

Posted by [Reza Rad](#) on May 13, 2016

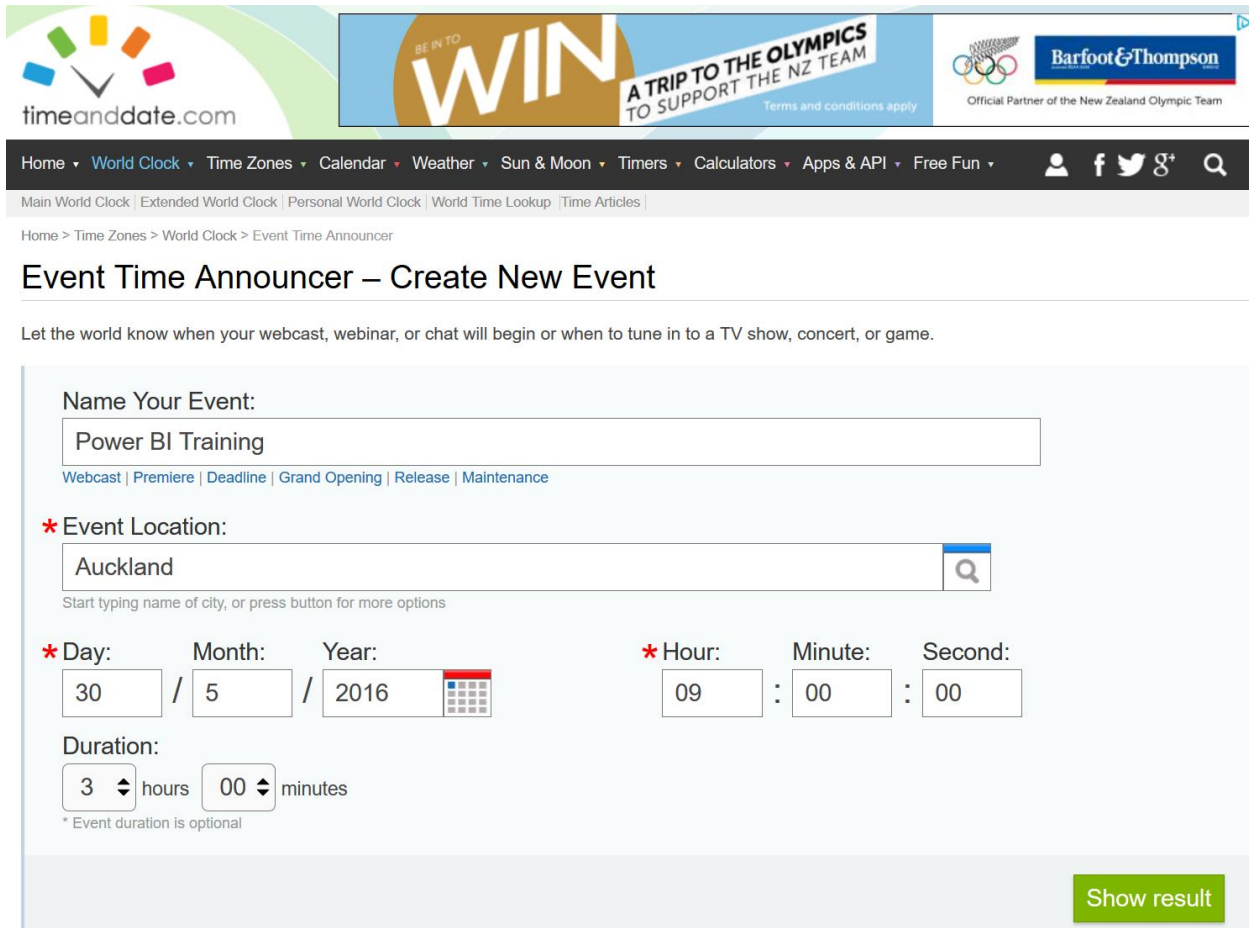


[Previously](#) you have read how to use Power Query as the data transformation component of Power BI, and you [learned how to use it in BI applications](#). You can read more about Power Query usage in BI in [Power BI online book from Rookie to Rock Star](#). In this series, I will explain how to use Power Query for non-BI usage. This data transformation tool can be used anywhere to use input parameters, fetch the data from the web or any other data sources, mash up the data to form the required shape and automate the process. In this series, I will show you a real use case of using Power Query for the non-BI application.

## Why?

As an introduction to this series, I want to take you to the path that leads me to use Power Query here. You might be aware that I am teaching Power BI courses, and most of my courses are online and Live. This means that courses are not recorded videos, it is me on the other side of the line with full interactive audio and video experience with students with Go2Meeting application. Students connecting to me from other places in the world. So I do need an event date/time scheduler that I can announce the date and time of the event in different time zones.

Fortunately, there is a very good [website](#) that helps to find a date/time in different time zones. In this website, I can set my input parameters as the date/time of my event locally (in my city), and the name of the event, and duration.



The screenshot shows the 'Event Time Announcer' page on timeanddate.com. The page has a header with the site logo, navigation links (Home, World Clock, Time Zones, Calendar, Weather, Sun & Moon, Timers, Calculators, Apps & API, Free Fun), and social media icons. Below the header is a breadcrumb trail: Home > Time Zones > World Clock > Event Time Announcer. The main heading is 'Event Time Announcer – Create New Event'. A subheading reads: 'Let the world know when your webcast, webinar, or chat will begin or when to tune in to a TV show, concert, or game.' The form contains the following fields:

- Name Your Event:** A text input field containing 'Power BI Training'. Below it are links: [Webcast](#) | [Premiere](#) | [Deadline](#) | [Grand Opening](#) | [Release](#) | [Maintenance](#).
- \* Event Location:** A text input field containing 'Auckland' with a search button. Below it is the text: 'Start typing name of city, or press button for more options'.
- \* Day:** A dropdown menu showing '30'.
- Month:** A dropdown menu showing '5'.
- Year:** A text input field showing '2016' with a calendar icon.
- \* Hour:** A dropdown menu showing '09'.
- Minute:** A dropdown menu showing '00'.
- Second:** A dropdown menu showing '00'.
- Duration:** A section with '3' hours and '00' minutes dropdowns. Below it is the text: '\* Event duration is optional'.

A green 'Show result' button is located at the bottom right of the form.

Then the website produces a result set of important cities in different time zones with the date and time of the event in their time zones. That's brilliant.

[Africa](#) | [North America](#) | [South America](#) | [Asia](#) | [Australia/Pacific](#) | [Europe](#)

The [Event Time Announcer](#) converts the time of your planned event to local times all over the world for easy sharing and linking.

Event Times around the world					
Sort by: <span>City</span>					Cities shown: <span>Most Popular (144)</span>
Location	Start time	End time	Location	Start time	End time
<a href="#">Accra</a>	Sun 9:00 p.m.	Sun 12:00 Midn	<a href="#">Kiritimati</a>	Mon 11:00 a.m.	Mon 2:00 p.m.
<a href="#">Addis Ababa</a>	Sun 12:00 Midn	Mon 3:00 a.m.	<a href="#">Kolkata</a>	Mon 2:30 a.m.	Mon 5:30 a.m.
<a href="#">Adelaide</a>	Mon 6:30 a.m.	Mon 9:30 a.m.	<a href="#">Kuala Lumpur</a>	Mon 5:00 a.m.	Mon 8:00 a.m.
<a href="#">Algiers</a>	Sun 10:00 p.m.	Mon 1:00 a.m.	<a href="#">Kuwait City</a>	Sun 12:00 Midn	Mon 3:00 a.m.
<a href="#">Almaty</a>	Mon 3:00 a.m.	Mon 6:00 a.m.	<a href="#">Kyiv *</a>	Sun 12:00 Midn	Mon 3:00 a.m.
<a href="#">Amman *</a>	Sun 12:00 Midn	Mon 3:00 a.m.	<a href="#">La Paz</a>	Sun 5:00 p.m.	Sun 8:00 p.m.
<a href="#">Amsterdam *</a>	Sun 11:00 p.m.	Mon 2:00 a.m.	<a href="#">Lagos</a>	Sun 10:00 p.m.	Mon 1:00 a.m.
<a href="#">Anadyr</a>	Mon 9:00 a.m.	Mon 12:00 Noon	<a href="#">Lahore</a>	Mon 2:00 a.m.	Mon 5:00 a.m.
<a href="#">Anchorage *</a>	Sun 1:00 p.m.	Sun 4:00 p.m.	<a href="#">Las Vegas *</a>	Sun 2:00 p.m.	Sun 5:00 p.m.
<a href="#">Ankara *</a>	Sun 12:00 Midn	Mon 3:00 a.m.	<a href="#">Lima</a>	Sun 4:00 p.m.	Sun 7:00 p.m.
<a href="#">Antananarivo</a>	Sun 12:00 Midn	Mon 3:00 a.m.	<a href="#">Lisbon *</a>	Sun 10:00 p.m.	Mon 1:00 a.m.
<a href="#">Asuncion</a>	Sun 5:00 p.m.	Sun 8:00 p.m.	<a href="#">London *</a>	Sun 10:00 p.m.	Mon 1:00 a.m.
<a href="#">Athens *</a>	Sun 12:00 Midn	Mon 3:00 a.m.	<a href="#">Los Angeles *</a>	Sun 2:00 p.m.	Sun 5:00 p.m.
<a href="#">Atlanta *</a>	Sun 5:00 p.m.	Sun 8:00 p.m.	<a href="#">Madrid *</a>	Sun 11:00 p.m.	Mon 2:00 a.m.
<a href="#">Auckland</a>	Mon 9:00 a.m.	Mon 12:00 Noon	<a href="#">Managua</a>	Sun 3:00 p.m.	Sun 6:00 p.m.
<a href="#">Baghdad</a>	Sun 12:00 Midn	Mon 3:00 a.m.	<a href="#">Manila</a>	Mon 5:00 a.m.	Mon 8:00 a.m.
<a href="#">Bangkok</a>	Mon 4:00 a.m.	Mon 7:00 a.m.	<a href="#">Melbourne</a>	Mon 7:00 a.m.	Mon 10:00 a.m.
<a href="#">Barcelona *</a>	Sun 11:00 p.m.	Mon 2:00 a.m.	<a href="#">Mexico City *</a>	Sun 4:00 p.m.	Sun 7:00 p.m.

The main issue here is that; I want to use this list in my course announcement web page, and the list above is overwhelming! If I copy and paste list above, it will make my page so long. The audience will lose interest to read such a long page to find their city and local time zone. What I need is this:

- List of all above cities with the event date time on their time zone, **categorized by time**. That means; Los Angeles, San Francisco, Las Vegas, Seattle, and Portland, should be all listed in one row of my result set table.

Start time	End time	Cities
Mon 1:00 a.m.	Mon 4:00 a.m.	Accra, Reykjavik
Mon 4:00 a.m.	Mon 7:00 a.m.	Addis Ababa, Amman, Ankara, Antananarivo, Athens, Baghdad, Beirut
Mon 10:30 a.m.	Mon 1:30 p.m.	Adelaide, Darwin
Mon 2:00 a.m.	Mon 5:00 a.m.	Algiers, Casablanca, Dublin, Kinshasa, Lagos, Lisbon, London
Mon 7:00 a.m.	Mon 10:00 a.m.	Almaty, Dhaka
Mon 3:00 a.m.	Mon 6:00 a.m.	Amsterdam, Barcelona, Belgrade, Berlin, Brussels, Budapest, Cairo, Ca
Mon 1:00 p.m.	Mon 4:00 p.m.	Anadyr, Auckland, Suva
Sun 5:00 p.m.	Sun 8:00 p.m.	Anchorage
Sun 9:00 p.m.	Sun 12:00 Midn	Asuncion, Atlanta, Boston, Caracas, Columbus, Detroit, Havana, Indian
Mon 8:00 a.m.	Mon 11:00 a.m.	Bangkok, Hanoi, Jakarta
Mon 9:00 a.m.	Mon 12:00 Noon	Beijing, Hong Kong, Kuala Lumpur, Manila, Perth, Shanghai, Singapore,
Mon 6:30 a.m.	Mon 9:30 a.m.	Bengaluru, Kolkata, Mumbai, New Delhi
Sun 8:00 p.m.	Sun 11:00 p.m.	Bogota, Chicago, Dallas, Houston, Kingston, Lima, Mexico City, Minnea
Sun 10:00 p.m.	Mon 1:00 a.m.	Brasilia, Buenos Aires, Halifax, Montevideo, Rio de Janeiro, Santiago, S
Mon 11:00 a.m.	Mon 2:00 p.m.	Brisbane, Canberra, Melbourne, Sydney
Sun 7:00 p.m.	Sun 10:00 p.m.	Calgary, Denver, Edmonton, Guatemala, Managua, Salt Lake City, San S
Mon 5:00 a.m.	Mon 8:00 a.m.	Dubai
Sun 3:00 p.m.	Sun 6:00 p.m.	Honolulu
Mon 6:00 a.m.	Mon 9:00 a.m.	Islamabad, Karachi, Lahore, Tashkent
Mon 5:30 a.m.	Mon 8:30 a.m.	Kabul, Tehran
Mon 6:45 a.m.	Mon 9:45 a.m.	Kathmandu
Mon 3:00 p.m.	Mon 6:00 p.m.	Kiritimati
Sun 6:00 p.m.	Sun 9:00 p.m.	Las Vegas, Los Angeles, Phoenix, San Francisco, Seattle, Vancouver
Mon 10:00 a.m.	Mon 1:00 p.m.	Seoul, Tokyo
Sun 10:30 p.m.	Mon 1:30 a.m.	St. John's
Mon 7:30 a.m.	Mon 10:30 a.m.	Yangon

- I want this list to be in Excel format, so I can send it to my mailing list or attach it as an excel table
- I don't want to produce the data from a website. I want to change the parameters in my Excel file, and it automatically fetches data from this website, and load result set back to my Excel spreadsheet.

So, as a result, I decided to use Power Query because with Power Query I can;

- Have input parameters in an Excel sheet for local date/time, and duration of the event.
- Browse web URL and fetch result set.
- Mash up the output data to produce my desired data set.
- Load result set into Excel.

So Let's now start building this solution step by step.

## What You Will Learn?

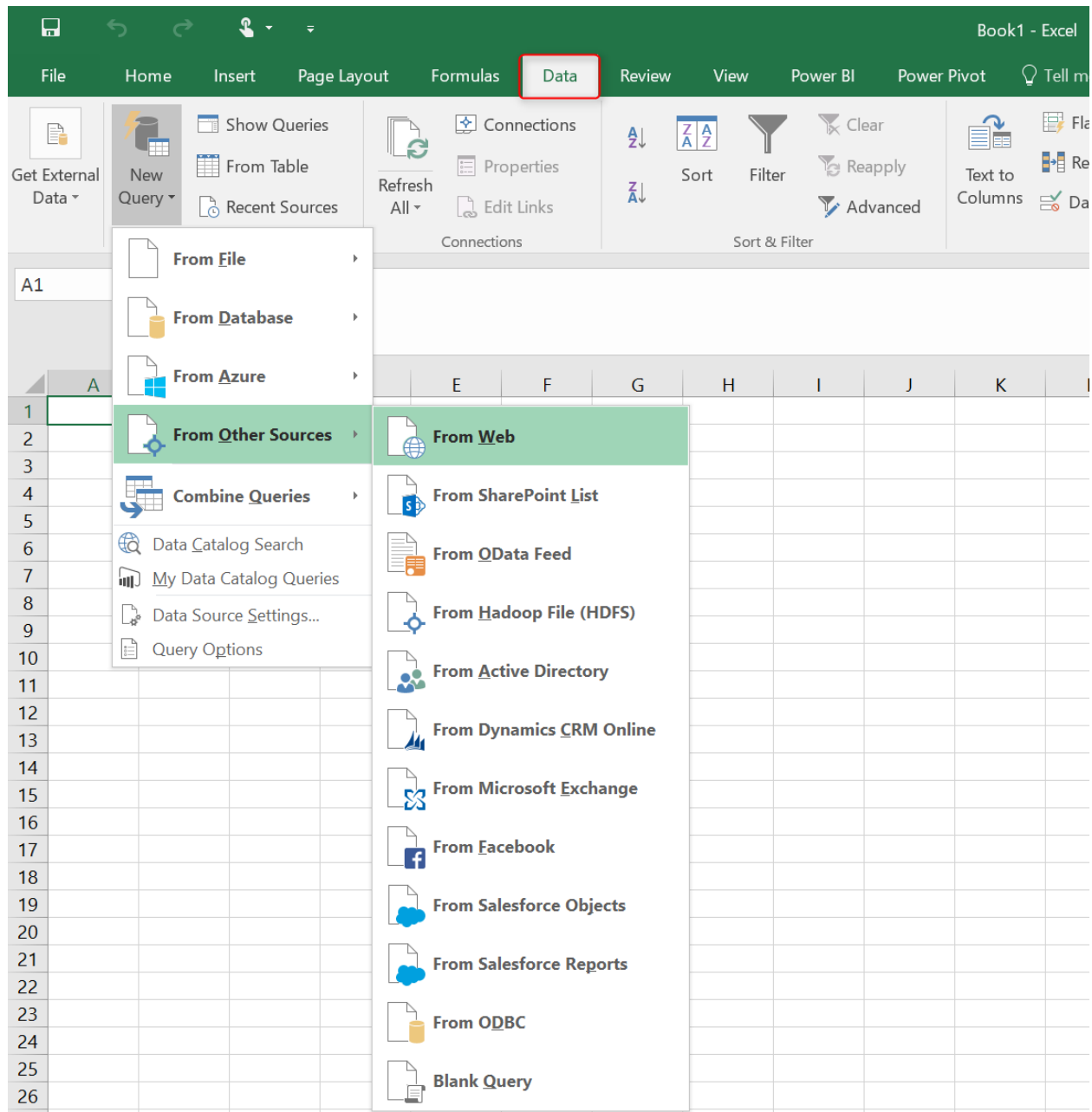
By completing this scenario, you will learn a real-world use case of Power Query for a non-BI solution. You will learn using Power Query for;

- Defining a table for input parameters in Excel sheet
- Using Power Query to read values from that table as input parameter and build web URL for timeanddate.com web query.
- Querying web URL and Fetching the data
- Data Manipulation with Power Query such as; Removing Columns, Replacing Values, Appending Tables, Grouping Rows, Adding Custom Columns, Changing Tables to Column, Columns to Lists, Lists to Tables, and Concatenating Table Columns with Comma.
- Automating the Power Query solution, to take the parameters and produce result set when you hit the Refresh button.

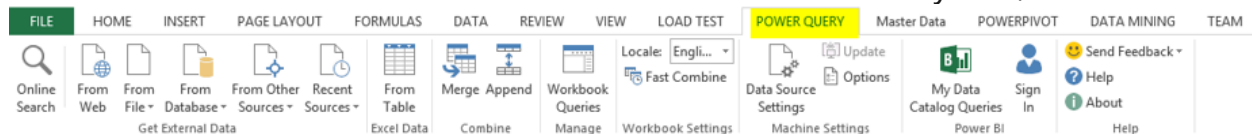
## Prerequisite

This example has been done with Excel 2016 which has Power Query embedded. So if you have Excel 2016, you don't need to install anything. If you are using Excel 2013 or 2010, you need to download and install Power Query for Excel Add-in. Power Query editor experience in all of these versions of Excel are the same. The only difference is that Power Query menu options in Excel 2016 are located under Data Tab;





And in Excel 2013 and 2010 are located under Power Query Tab;

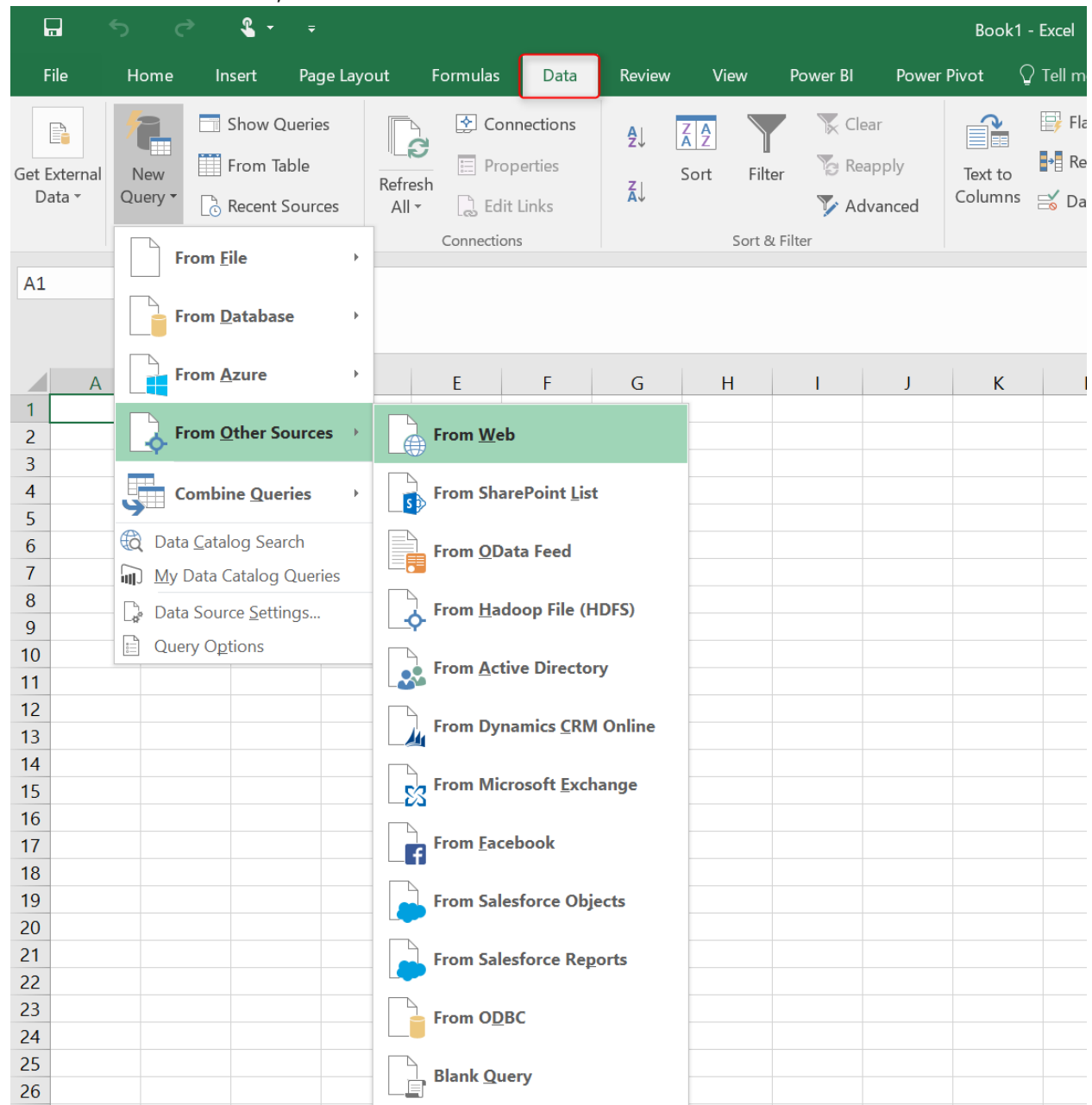


## Get Data from Web

I'll start by the easy part of the work, which is using a static web URL like this:

<http://www.timeanddate.com/worldclock/fixedtime.html?msg=Power+BI+Training&iso=20160530T09&p1=22&ah=3>

In Power Query, I can fetch tables produced in this web page. Follow the menu as Go to Data Tab (In excel 2013/2010 Power Query Tab), New Query, From Other Data Sources, and then From Web



Enter the URL mentioned above in the dialog box

## From Web

Enter a Web page URL.

URL

fixedtime.html?msg=Power+BI+Training&iso=20160530T09&p1=22&ah=3

OK

Cancel

Power Query is smart enough to separate each table, and It shows you a preview of each table. Choose the table that has values with cities and date/times.

## Navigators

☐ Select multiple items  
 Show All | Show Selected [1]

http://www.timeanddate.com/worldclock/fixed...  
 Document  
 Table 0

Table 0

Header	Location	Start time	End time
Sort by: Event Times around the world Cities shown:	Accra	Sun 9:00 p.m.	Sun 12:00 Midn
Sort by: Event Times around the world Cities shown:	Addis Ababa	Sun 12:00 Midn	Mon 3:00 a.m.
Sort by: Event Times around the world Cities shown:	Adelaide	Mon 6:30 a.m.	Mon 9:30 a.m.
Sort by: Event Times around the world Cities shown:	Algiers	Sun 10:00 p.m.	Mon 1:00 a.m.
Sort by: Event Times around the world Cities shown:	Almaty	Mon 3:00 a.m.	Mon 6:00 a.m.
Sort by: Event Times around the world Cities shown:	Amman*	Sun 12:00 Midn	Mon 3:00 a.m.
Sort by: Event Times around the world Cities shown:	Amsterdam*	Sun 11:00 p.m.	Mon 2:00 a.m.
Sort by: Event Times around the world Cities shown:	Anadyr	Mon 9:00 a.m.	Mon 12:00 Noo
Sort by: Event Times around the world Cities shown:	Anchorage*	Sun 1:00 p.m.	Sun 4:00 p.m.
Sort by: Event Times around the world Cities shown:	Ankara*	Sun 12:00 Midn	Mon 3:00 a.m.
Sort by: Event Times around the world Cities shown:	Antananarivo	Sun 12:00 Midn	Mon 3:00 a.m.

Load

Edit

Cancel

I can see in the preview above that there are some extra columns, and also I need to apply some transformations. So I Click on Edit to lunch query editor for further data manipulation.

Table 0 - Query Editor

File Home Transform Add Column View

Close & Load Refresh Preview Advanced Editor Choose Columns Remove Columns Keep Rows Remove Rows Remove Duplicates Remove Errors Split Column Group By Data Type: Text Use First Row As Headers Replace Values Merge Queries Append Queries Combine Binaries New Source Recent Sources

1 query Table 0

Table.TransformColumnTypes(Data0,{{"Header", type text}, {"Location", type text}, {"Start time", type text}, {"End time", type text}, {"Location2", type text}, {"Start time2", type text}, {"End time2", type text}}

	Header	Location	Start time	End time	Location2	Start time2	End time2
1	Sort by: Event Times around the world Cities shown:	Accra	Sun 9:00 p.m.	Sun 12:00 Midn	Kiritimati	Mon 11:00 a.m.	Mon 2:00 p.m.
2	Sort by: Event Times around the world Cities shown:	Addis Ababa	Sun 12:00 Midn	Mon 3:00 a.m.	Kolkata	Mon 2:30 a.m.	Mon 5:30 a.m.
3	Sort by: Event Times around the world Cities shown:	Adelaide	Mon 6:30 a.m.	Mon 9:30 a.m.	Kuala Lumpur	Mon 5:00 a.m.	Mon 8:00 a.m.
4	Sort by: Event Times around the world Cities shown:	Algiers	Sun 10:00 p.m.	Mon 1:00 a.m.	Kuwait City	Sun 12:00 Midn	Mon 3:00 a.m.
5	Sort by: Event Times around the world Cities shown:	Almaty	Mon 3:00 a.m.	Mon 6:00 a.m.	Kyiv*	Sun 12:00 Midn	Mon 3:00 a.m.
6	Sort by: Event Times around the world Cities shown:	Amman*	Sun 12:00 Midn	Mon 3:00 a.m.	La Paz	Sun 5:00 p.m.	Sun 8:00 p.m.
7	Sort by: Event Times around the world Cities shown:	Amsterdam*	Sun 11:00 p.m.	Mon 2:00 a.m.	Lagos	Sun 10:00 p.m.	Mon 1:00 a.m.
8	Sort by: Event Times around the world Cities shown:	Anadyr	Mon 9:00 a.m.	Mon 12:00 Noon	Lahore	Mon 2:00 a.m.	Mon 5:00 a.m.
9	Sort by: Event Times around the world Cities shown:	Anchorage*	Sun 1:00 p.m.	Sun 4:00 p.m.	Las Vegas*	Sun 2:00 p.m.	Sun 5:00 p.m.
10	Sort by: Event Times around the world Cities shown:	Ankara*	Sun 12:00 Midn	Mon 3:00 a.m.	Lima	Sun 4:00 p.m.	Sun 7:00 p.m.
11	Sort by: Event Times around the world Cities shown:	Antananarivo	Sun 12:00 Midn	Mon 3:00 a.m.	Lisbon*	Sun 10:00 p.m.	Mon 1:00 a.m.
12	Sort by: Event Times around the world Cities shown:	Asuncion	Sun 5:00 p.m.	Sun 8:00 p.m.	London*	Sun 10:00 p.m.	Mon 1:00 a.m.
13	Sort by: Event Times around the world Cities shown:	Athens*	Sun 12:00 Midn	Mon 3:00 a.m.	Los Angeles*	Sun 2:00 p.m.	Sun 5:00 p.m.

Query Settings

PROPERTIES

Name

Table 0

All Properties

APPLIED STEPS

Source

Navigation

Changed Type

## Apply Basic Transformations

When query editor lunches and Power Query loads the preview of the table, it also automatically applies some data type changes as the very first step. Here is the data set when it loads in query editor for the first time;

Table.TransformColumnTypes(Data01,{{"Header", type text}, {"Location", type text}, {"Start time", type text}, {"End time", type text}, {"Location2", type text}, {"Start time2", type text}, {"End time2", type text}}

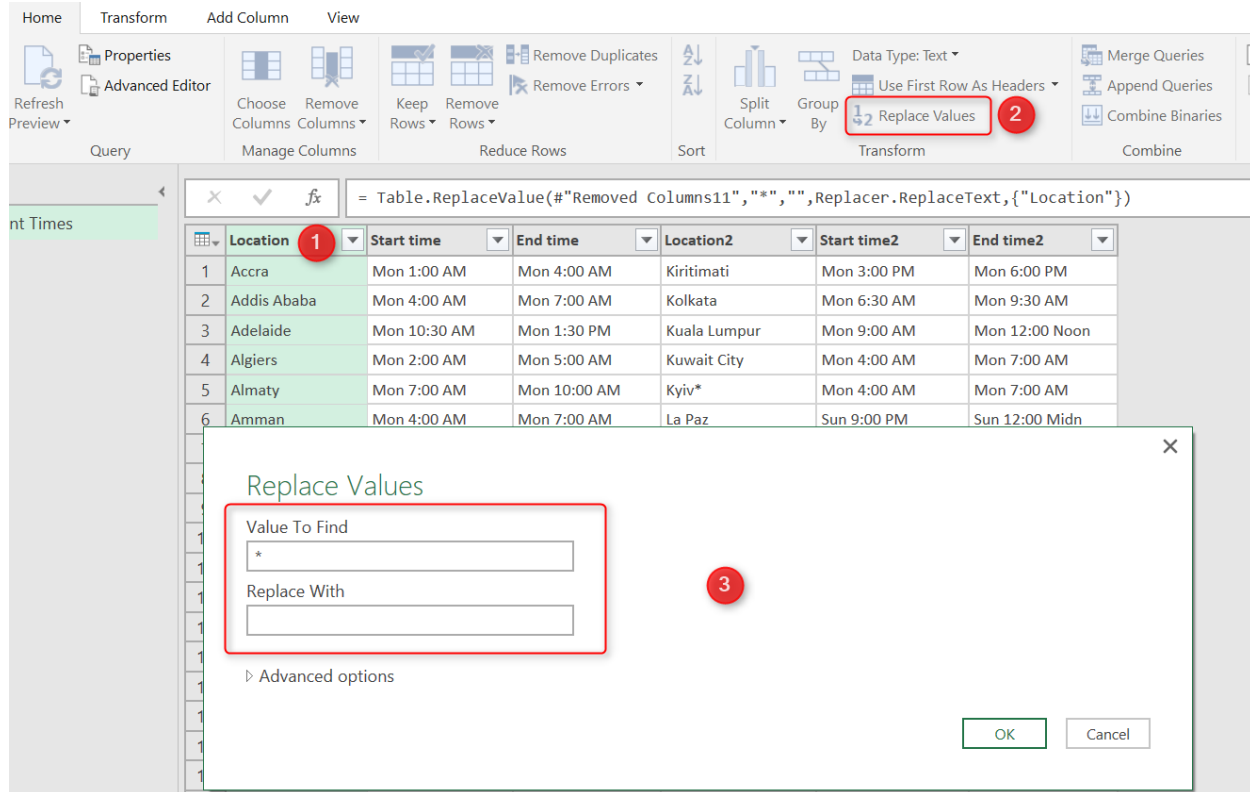
	Header	Location	Start time	End time	Location2	Start time2	End time2
1	Sort by: Event Times around the world Cities shown:	Accra	Mon 1:00 AM	Mon 4:00 AM	Kiritimati	Mon 3:00 PM	Mon 6:00 PM
2	Sort by: Event Times around the world Cities shown:	Addis Ababa	Mon 4:00 AM	Mon 7:00 AM	Kolkata	Mon 6:30 AM	Mon 9:30 AM
3	Sort by: Event Times around the world Cities shown:	Adelaide	Mon 10:30 AM	Mon 1:30 PM	Kuala Lumpur	Mon 9:00 AM	Mon 12:00 Noon
4	Sort by: Event Times around the world Cities shown:	Algiers	Mon 2:00 AM	Mon 5:00 AM	Kuwait City	Mon 4:00 AM	Mon 7:00 AM
5	Sort by: Event Times around the world Cities shown:	Almaty	Mon 7:00 AM	Mon 10:00 AM	Kyiv*	Mon 4:00 AM	Mon 7:00 AM
6	Sort by: Event Times around the world Cities shown:	Amman*	Mon 4:00 AM	Mon 7:00 AM	La Paz	Sun 9:00 PM	Sun 12:00 Midn
7	Sort by: Event Times around the world Cities shown:	Amsterdam*	Mon 3:00 AM	Mon 6:00 AM	Lagos	Mon 2:00 AM	Mon 5:00 AM
8	Sort by: Event Times around the world Cities shown:	Anadyr	Mon 1:00 PM	Mon 4:00 PM	Lahore	Mon 6:00 AM	Mon 9:00 AM
9	Sort by: Event Times around the world Cities shown:	Anchorage*	Sun 5:00 PM	Sun 8:00 PM	Las Vegas*	Sun 6:00 PM	Sun 9:00 PM

There is a Header column which is not giving me any useful information, So I'll remove it by right click on the column and Remove column

Table.TransformColumnTypes(Data01,{{"Header", type text}, {"Location", type text}, {"Start time", type text}, {"End time", type text}, {"Location2", type text}, {"Start time2", type text}, {"End time2", type text}}

	Header	Location	Start time	End time	Location2	Start time2	End time2
1	Sort by: Event Times around the world Cities shown:	Accra	Mon 1:00 AM	Mon 4:00 AM	Kiritimati	Mon 3:00 PM	Mon 6:00 PM
2	Sort by: Event Times around the world Cities shown:	Addis Ababa	Mon 4:00 AM	Mon 7:00 AM	Kolkata	Mon 6:30 AM	Mon 9:30 AM
3	Sort by: Event Times around the world Cities shown:	Adelaide	Mon 10:30 AM	Mon 1:30 PM	Kuala Lumpur	Mon 9:00 AM	Mon 12:00 Noon
4	Sort by: Event Times around the world Cities shown:	Algiers	Mon 2:00 AM	Mon 5:00 AM	Kuwait City	Mon 4:00 AM	Mon 7:00 AM
5	Sort by: Event Times around the world Cities shown:	Almaty	Mon 7:00 AM	Mon 10:00 AM	Kyiv*	Mon 4:00 AM	Mon 7:00 AM
6	Sort by: Event Times around the world Cities shown:	Amman*	Mon 4:00 AM	Mon 7:00 AM	La Paz	Sun 9:00 PM	Sun 12:00 Midn
7	Sort by: Event Times around the world Cities shown:	Amsterdam*	Mon 3:00 AM	Mon 6:00 AM	Lagos	Mon 2:00 AM	Mon 5:00 AM

There are also some \* characters in cities names (at the end of some cities, such as Barcelona\*). So I remove them by clicking on the Location Column, then choose the Replace Values from icons under Home menu, and then in replacing values dialog box I replace \* with a blank.



Home Transform Add Column View

Refresh Preview Properties Advanced Editor

Choose Columns Remove Columns Manage Columns

Keep Rows Remove Rows Reduce Rows

Remove Duplicates Remove Errors

Sort Split Column Group By

Data Type: Text Use First Row As Headers

Replace Values

Merge Queries Append Queries Combine Binaries

Query

nt Times

Location Start time End time Location2 Start time2 End time2

1 Accra Mon 1:00 AM Mon 4:00 AM Kiritimati Mon 3:00 PM Mon 6:00 PM

2 Addis Ababa Mon 4:00 AM Mon 7:00 AM Kolkata Mon 6:30 AM Mon 9:30 AM

3 Adelaide Mon 10:30 AM Mon 1:30 PM Kuala Lumpur Mon 9:00 AM Mon 12:00 Noon

4 Algiers Mon 2:00 AM Mon 5:00 AM Kuwait City Mon 4:00 AM Mon 7:00 AM

5 Almaty Mon 7:00 AM Mon 10:00 AM Kyiv\* Mon 4:00 AM Mon 7:00 AM

6 Amman Mon 4:00 AM Mon 7:00 AM La Paz Sun 9:00 PM Sun 12:00 Midn

Replace Values

Value To Find

\*

Replace With

Advanced options

OK Cancel

I'll do the same step again Location2 Column this time. Here is what my result set looks like so far;



Location Start time End time Location2 Start time2 End time2

1 Accra Sun 9:00 p.m. Sun 12:00 Midn Kiritimati Mon 11:00 a.m. Mon 2:00 p.m.

2 Addis Ababa Sun 12:00 Midn Mon 3:00 a.m. Kolkata Mon 2:30 a.m. Mon 5:30 a.m.

3 Adelaide Mon 6:30 a.m. Mon 9:30 a.m. Kuala Lumpur Mon 5:00 a.m. Mon 8:00 a.m.

4 Algiers Sun 10:00 p.m. Mon 1:00 a.m. Kuwait City Sun 12:00 Midn Mon 3:00 a.m.

5 Almaty Mon 3:00 a.m. Mon 6:00 a.m. Kyiv Sun 12:00 Midn Mon 3:00 a.m.

6 Amman Sun 12:00 Midn Mon 3:00 a.m. La Paz Sun 5:00 p.m. Sun 8:00 p.m.

7 Amsterdam Sun 11:00 p.m. Mon 2:00 a.m. Lagos Sun 10:00 p.m. Mon 1:00 a.m.

8 Anadyr Mon 9:00 a.m. Mon 12:00 Noon Lahore Mon 2:00 a.m. Mon 5:00 a.m.

9 Anchorage Sun 1:00 p.m. Sun 4:00 p.m. Las Vegas Sun 2:00 p.m. Sun 5:00 p.m.

10 Ankara Sun 12:00 Midn Mon 3:00 a.m. Lima Sun 4:00 p.m. Sun 7:00 p.m.

11 Antananarivo Sun 12:00 Midn Mon 3:00 a.m. Lisbon Sun 10:00 p.m. Mon 1:00 a.m.

12 Asuncion Sun 5:00 p.m. Sun 8:00 p.m. London Sun 10:00 p.m. Mon 1:00 a.m.

13 Athens Sun 12:00 Midn Mon 3:00 a.m. Los Angeles Sun 2:00 p.m. Sun 5:00 p.m.

14 Atlanta Sun 5:00 a.m. Sun 8:00 a.m. Madrid Sun 11:00 p.m. Mon 2:00 a.m.

Query Settings

PROPERTIES

Name

Event Times

All Properties

APPLIED STEPS

Source

Navigation

Changed Type

Removed Columns

Replaced Value

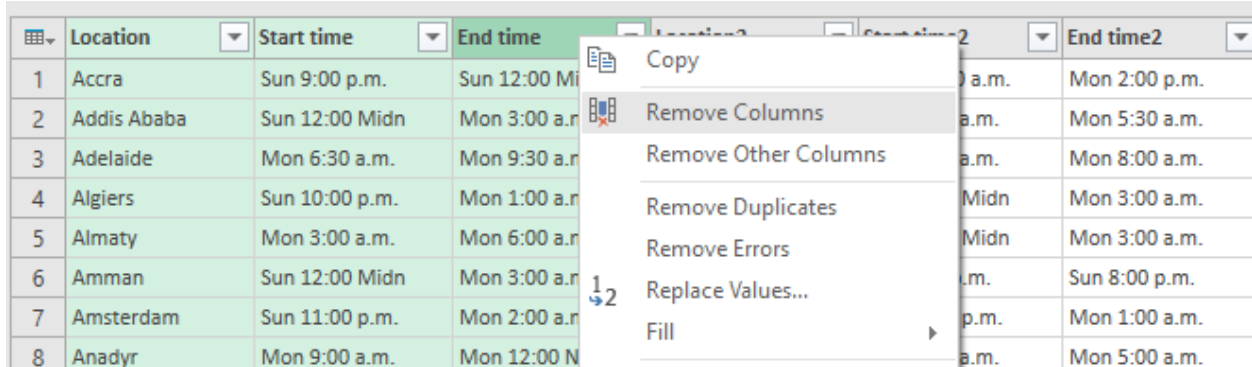
Replaced Value1

As you can see in the screenshot above all applied steps so far are listed in the right pane (numbered 3 in the screenshot above). This is one of the great features of Power Query that allows you rollback changes anytime you want and check the preview of a query for a specific step.

In the screenshot above I have to set of columns; first one numbered 1, and the second one numbered 2. These are not repetitive values. You can check

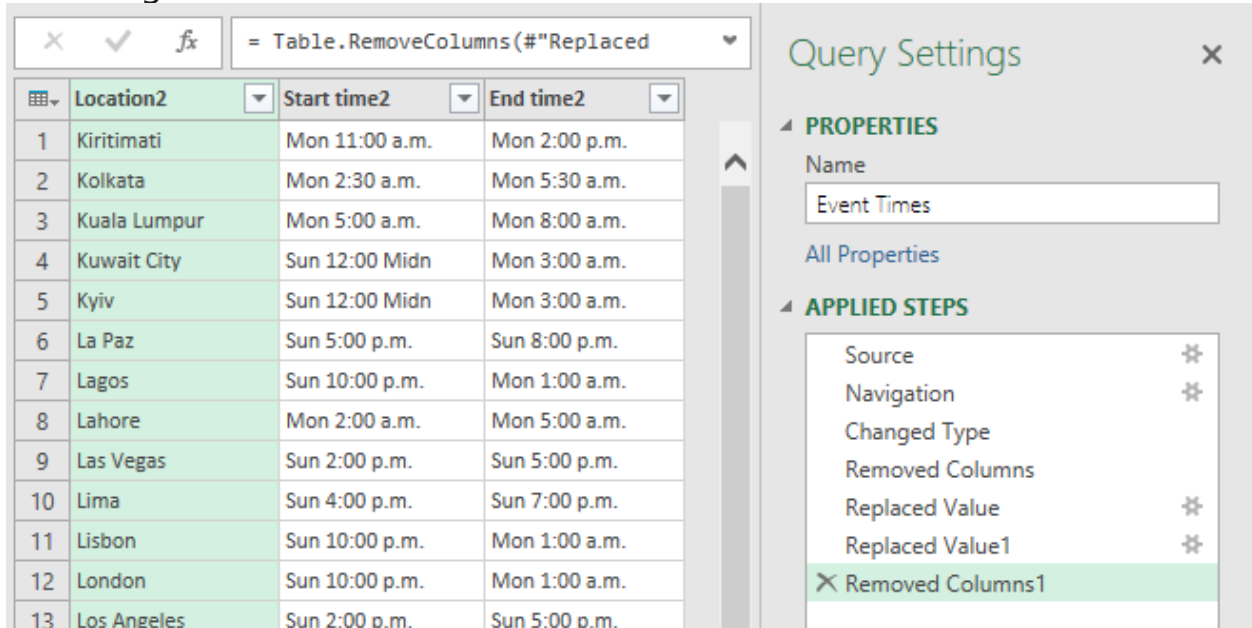
city names in columns Location and Location2. The fact is that timeanddate.com website split the result set into two sets of columns and produced HTML table like that. So for reading the full data set, I have to separate these two set of columns and then combine their values into a single set of columns.

So I'll produce a result set for the second set of columns with removing the first set;



	Location	Start time	End time	Location2	Start time2	End time2
1	Accra	Sun 9:00 p.m.	Sun 12:00 Midn			
2	Addis Ababa	Sun 12:00 Midn	Mon 3:00 a.m.			
3	Adelaide	Mon 6:30 a.m.	Mon 9:30 a.m.			
4	Algiers	Sun 10:00 p.m.	Mon 1:00 a.m.			
5	Almaty	Mon 3:00 a.m.	Mon 6:00 a.m.			
6	Amman	Sun 12:00 Midn	Mon 3:00 a.m.			
7	Amsterdam	Sun 11:00 p.m.	Mon 2:00 a.m.			
8	Anadyr	Mon 9:00 a.m.	Mon 12:00 N			

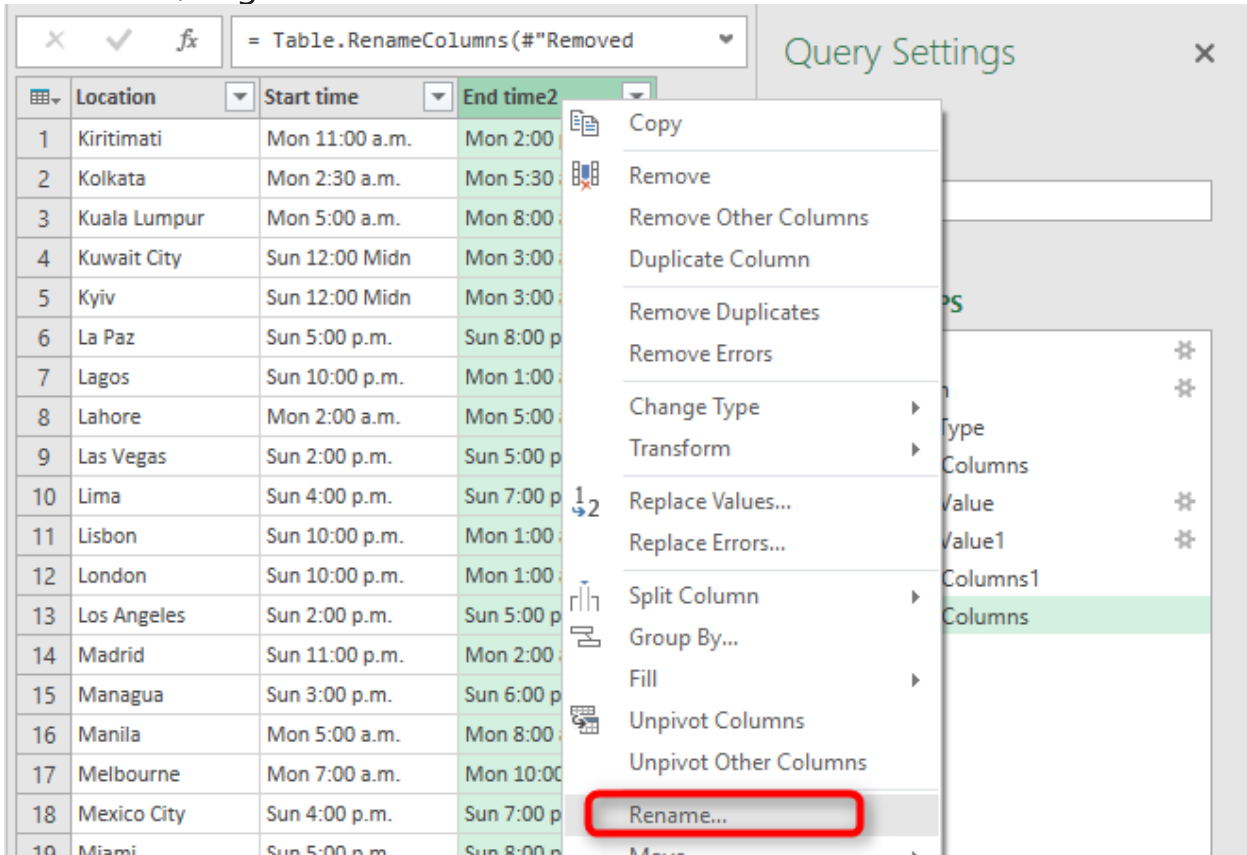
This will give me the second set of columns as below;



	Location2	Start time2	End time2
1	Kiritimati	Mon 11:00 a.m.	Mon 2:00 p.m.
2	Kolkata	Mon 2:30 a.m.	Mon 5:30 a.m.
3	Kuala Lumpur	Mon 5:00 a.m.	Mon 8:00 a.m.
4	Kuwait City	Sun 12:00 Midn	Mon 3:00 a.m.
5	Kyiv	Sun 12:00 Midn	Mon 3:00 a.m.
6	La Paz	Sun 5:00 p.m.	Sun 8:00 p.m.
7	Lagos	Sun 10:00 p.m.	Mon 1:00 a.m.
8	Lahore	Mon 2:00 a.m.	Mon 5:00 a.m.
9	Las Vegas	Sun 2:00 p.m.	Sun 5:00 p.m.
10	Lima	Sun 4:00 p.m.	Sun 7:00 p.m.
11	Lisbon	Sun 10:00 p.m.	Mon 1:00 a.m.
12	London	Sun 10:00 p.m.	Mon 1:00 a.m.
13	Los Angeles	Sun 2:00 p.m.	Sun 5:00 p.m.

Then I rename the column names for this data set. The reason for this rename is that later on when I want to combine two queries together, column names

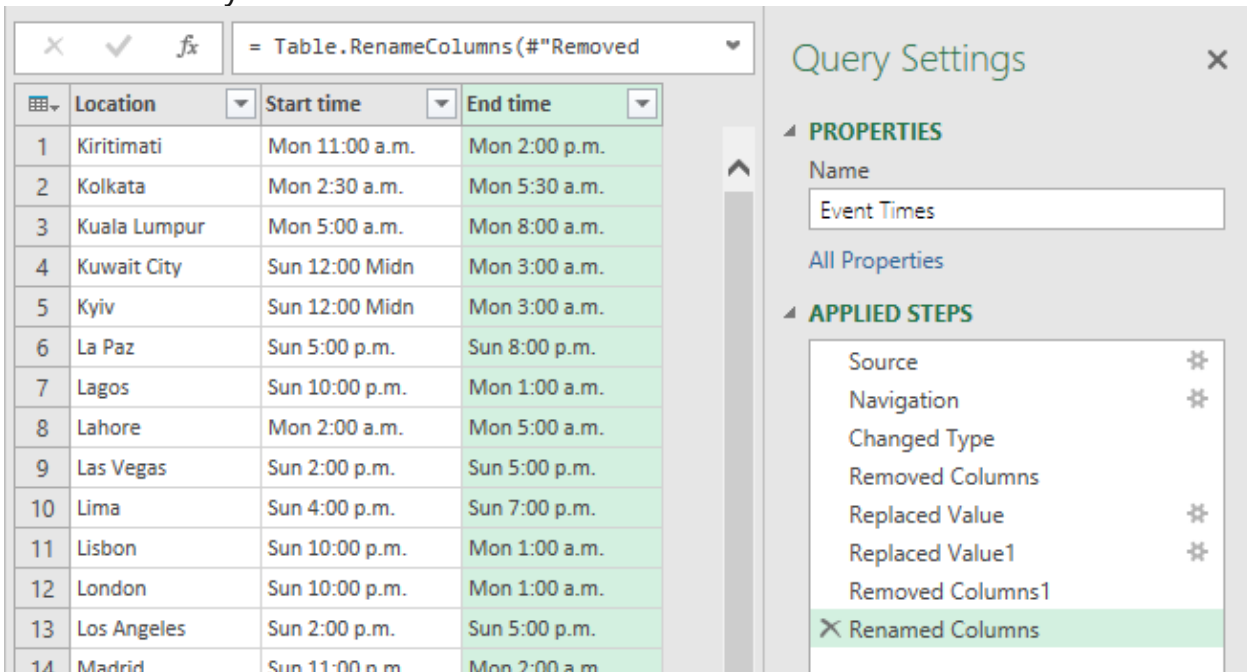
should be the same. I'll rename Columns to be Location, Start Time, End Time. for rename, I right-click on each column and Rename



The screenshot shows the Power Query Editor interface. A table with 18 rows and 3 columns is displayed. The columns are 'Location', 'Start time', and 'End time2'. A right-click context menu is open over the 'End time2' column, showing various options. The 'Rename...' option is highlighted with a red rectangle.

	Location	Start time	End time2
1	Kiritimati	Mon 11:00 a.m.	Mon 2:00
2	Kolkata	Mon 2:30 a.m.	Mon 5:30
3	Kuala Lumpur	Mon 5:00 a.m.	Mon 8:00
4	Kuwait City	Sun 12:00 Midn	Mon 3:00
5	Kyiv	Sun 12:00 Midn	Mon 3:00
6	La Paz	Sun 5:00 p.m.	Sun 8:00 p
7	Lagos	Sun 10:00 p.m.	Mon 1:00
8	Lahore	Mon 2:00 a.m.	Mon 5:00
9	Las Vegas	Sun 2:00 p.m.	Sun 5:00 p
10	Lima	Sun 4:00 p.m.	Sun 7:00 p
11	Lisbon	Sun 10:00 p.m.	Mon 1:00
12	London	Sun 10:00 p.m.	Mon 1:00
13	Los Angeles	Sun 2:00 p.m.	Sun 5:00 p
14	Madrid	Sun 11:00 p.m.	Mon 2:00
15	Managua	Sun 3:00 p.m.	Sun 6:00 p
16	Manila	Mon 5:00 a.m.	Mon 8:00
17	Melbourne	Mon 7:00 a.m.	Mon 10:00
18	Mexico City	Sun 4:00 p.m.	Sun 7:00 p
19	Miami	Sun 5:00 p.m.	Sun 8:00 p

And here is my result set for the second data set:



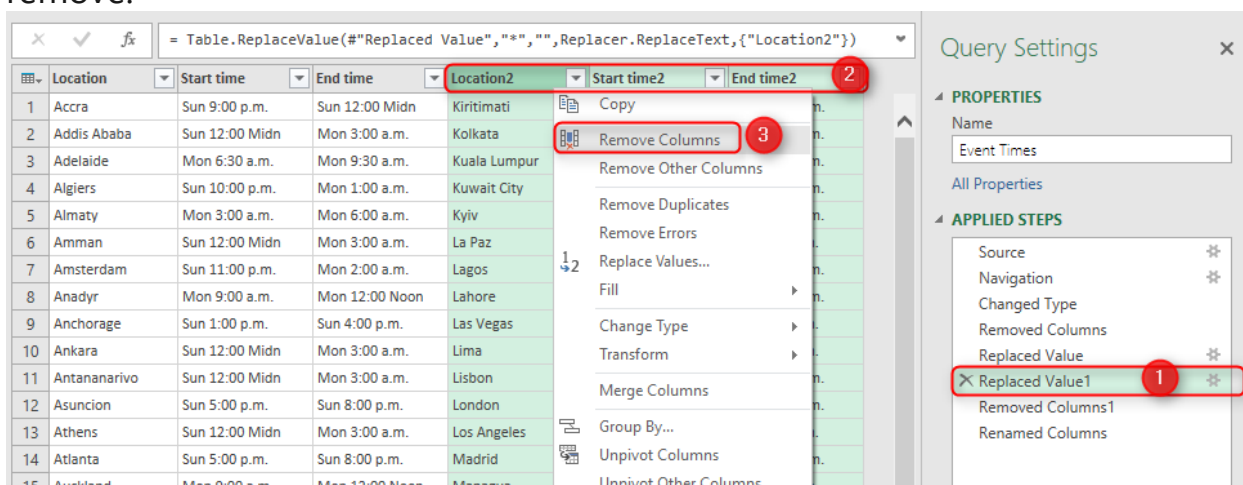
The screenshot shows the Power Query Editor interface after renaming the columns. The table now has columns 'Location', 'Start time', and 'End time'. The 'Query Settings' pane on the right is open, showing the 'APPLIED STEPS' list. The 'Renamed Columns' step is highlighted in green.

	Location	Start time	End time
1	Kiritimati	Mon 11:00 a.m.	Mon 2:00 p.m.
2	Kolkata	Mon 2:30 a.m.	Mon 5:30 a.m.
3	Kuala Lumpur	Mon 5:00 a.m.	Mon 8:00 a.m.
4	Kuwait City	Sun 12:00 Midn	Mon 3:00 a.m.
5	Kyiv	Sun 12:00 Midn	Mon 3:00 a.m.
6	La Paz	Sun 5:00 p.m.	Sun 8:00 p.m.
7	Lagos	Sun 10:00 p.m.	Mon 1:00 a.m.
8	Lahore	Mon 2:00 a.m.	Mon 5:00 a.m.
9	Las Vegas	Sun 2:00 p.m.	Sun 5:00 p.m.
10	Lima	Sun 4:00 p.m.	Sun 7:00 p.m.
11	Lisbon	Sun 10:00 p.m.	Mon 1:00 a.m.
12	London	Sun 10:00 p.m.	Mon 1:00 a.m.
13	Los Angeles	Sun 2:00 p.m.	Sun 5:00 p.m.
14	Madrid	Sun 11:00 p.m.	Mon 2:00 a.m.

## Insert a Step

Now I want to produce another variable for the first set of columns. Instead of creating the whole process again from getting data from the web, I start from one step ago which was before removing columns. As I've mentioned earlier, I can grab that status easily from the Applied Steps section in Query Settings pane. There is a step named Replaced Value1. This step belongs to where I replaced \* characters with blank for Location2 columns.

I Click on Replaced Value1 step and then choose the second set of columns to remove.



The screenshot shows the Power BI Query Editor interface. The main area displays a table with columns: Location, Start time, End time, Location2, Start time2, and End time2. The 'Location2' column is selected, and a context menu is open with 'Remove Columns' highlighted. The 'Query Settings' pane on the right shows the 'APPLIED STEPS' section, where 'Replaced Value1' is highlighted. Red circles and numbers (1, 2, 3) are used to indicate specific actions and elements in the interface.

When I click on Remove columns as above, I see a message says; Are you sure you want to insert a step? I get this message because this is not the latest step in my query editor. As long as I know what I am doing adding an extra step is fine.





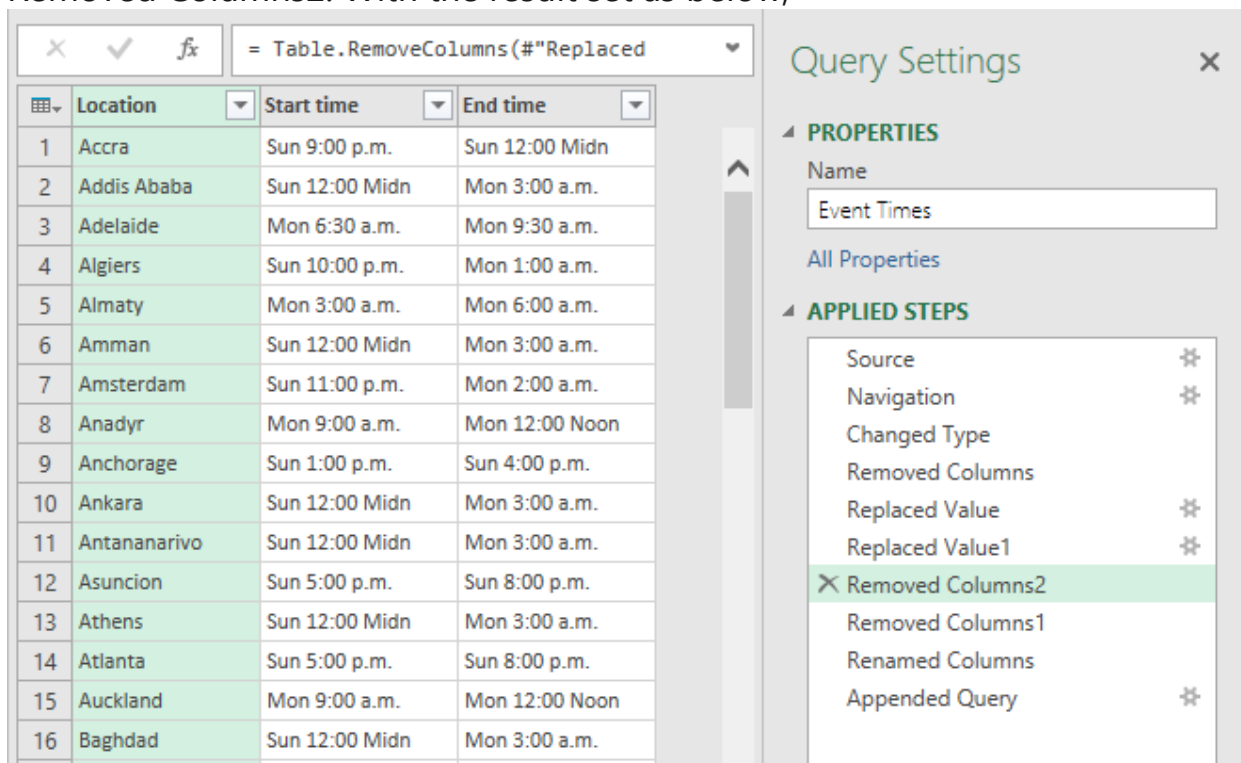
## Insert Step

Are you sure you want to insert a step? Inserting an intermediate step may affect subsequent steps, which could cause your query to break.

Insert

Cancel

I hit Insert for the message above, and a new step will be created called Removed Columns2. With the result set as below;



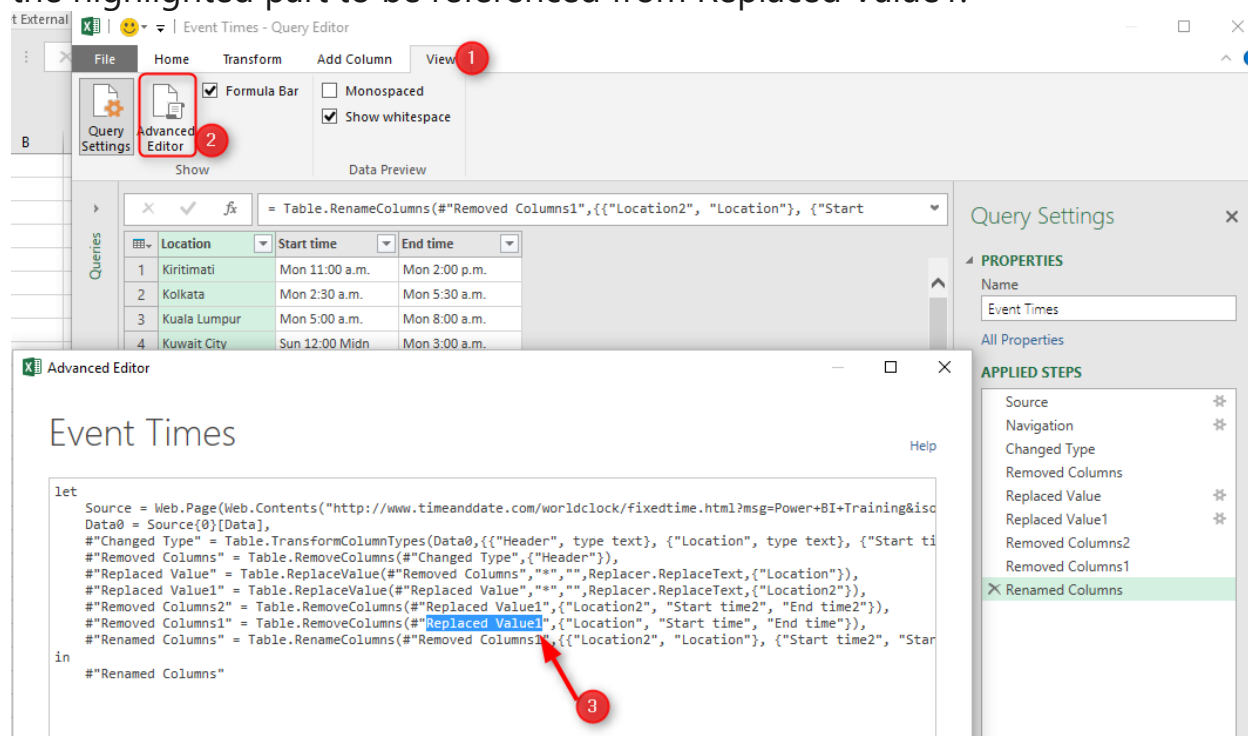
The screenshot shows the Power Query Editor interface. On the left, a table with 16 rows and 4 columns is displayed. The columns are 'Location', 'Start time', and 'End time'. The 'Location' column contains city names, and the 'Start time' and 'End time' columns contain time ranges. On the right, the 'Query Settings' pane is open, showing the 'APPLIED STEPS' list. The steps are: Source, Navigation, Changed Type, Removed Columns, Replaced Value, Replaced Value1, **Removed Columns2** (highlighted), Removed Columns1, Renamed Columns, and Appended Query. The formula bar at the top shows the formula: `= Table.RemoveColumns("#Replaced`.

	Location	Start time	End time
1	Accra	Sun 9:00 p.m.	Sun 12:00 Midn
2	Addis Ababa	Sun 12:00 Midn	Mon 3:00 a.m.
3	Adelaide	Mon 6:30 a.m.	Mon 9:30 a.m.
4	Algiers	Sun 10:00 p.m.	Mon 1:00 a.m.
5	Almaty	Mon 3:00 a.m.	Mon 6:00 a.m.
6	Amman	Sun 12:00 Midn	Mon 3:00 a.m.
7	Amsterdam	Sun 11:00 p.m.	Mon 2:00 a.m.
8	Anadyr	Mon 9:00 a.m.	Mon 12:00 Noon
9	Anchorage	Sun 1:00 p.m.	Sun 4:00 p.m.
10	Ankara	Sun 12:00 Midn	Mon 3:00 a.m.
11	Antananarivo	Sun 12:00 Midn	Mon 3:00 a.m.
12	Asuncion	Sun 5:00 p.m.	Sun 8:00 p.m.
13	Athens	Sun 12:00 Midn	Mon 3:00 a.m.
14	Atlanta	Sun 5:00 p.m.	Sun 8:00 p.m.
15	Auckland	Mon 9:00 a.m.	Mon 12:00 Noon
16	Baghdad	Sun 12:00 Midn	Mon 3:00 a.m.

### Fixing Insert Step reference issue

When I insert a step in Power Query, the step after that will source from the inserted step. This will cause some issues in my scenario. I inserted a step and removed Location2, Start time2, and End time2. And then in Removed Columns1, I want to remove other columns and in Renamed Columns rename

I go to View tab, Advanced Editor (or I can find Advanced Editor button in Home tab). This will open Advanced Editor with M code for me. Then I replace the highlighted part to be referenced from Replaced Value1.



Removed Columns2: the data set for the first set of columns

Renamed Columns: the data set for the second set of columns

Now It's the time to combine these two data sets. There is a menu option that combines two data sets. However, it only appends two QUERIES. As I have everything here in one query (but different steps or let's say variables), so I can't use that option. Fortunately, I can do that with a single line of M scripting.

I click on Advanced Editor to see the M code generated behind the scene. Then I Insert code line below at the end of LET clause, after adding a single comma (which is line separator in Power Query) in the line before it.

```
1 #"Appended Query" = Table.Combine({#"Removed Columns2", #"Renamed Columns" })
```

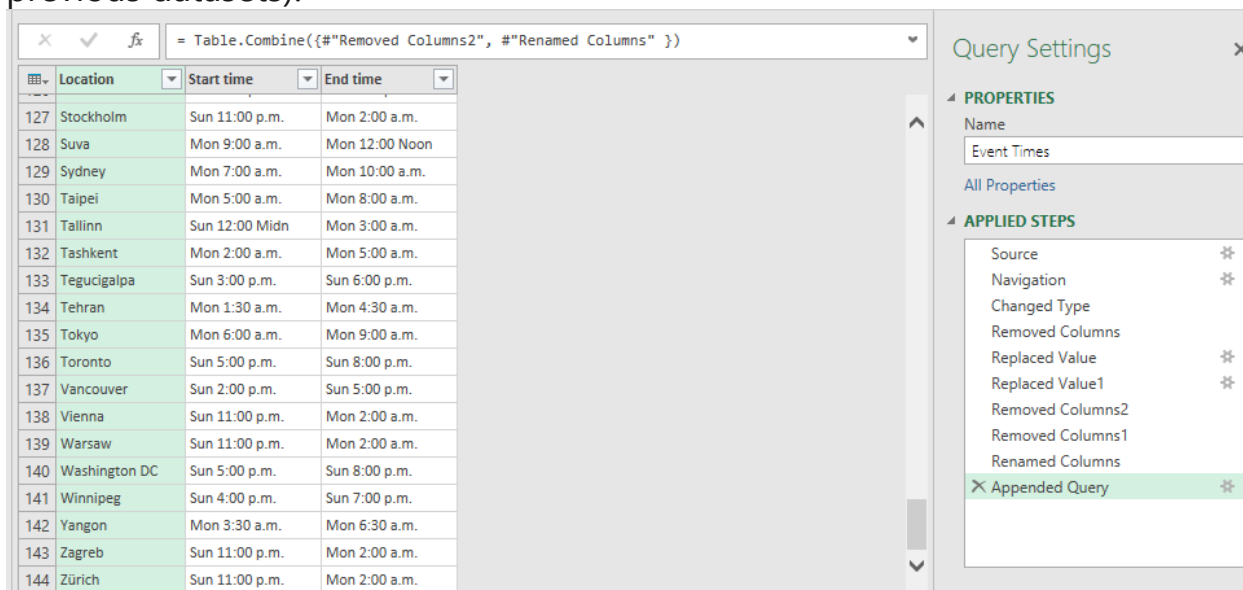
The statement above creates a new data set as a variable called Appended Query. I've used Table.Combine to append queries (or UNION them in SQL and Database terminologies). This function is a simple function and gets datasets to be combined which are Removed Columns1 and Renamed Columns1.

Then I show this step in the result set with writing that in IN section:

```
1 in
```

```
2     #"Appended Query"
```

Now my result set has both datasets combined into one. (I can check this includes 144 columns for 144 cities; combined list of 72 cities in each of previous datasets).



The screenshot shows the Power Query Advanced Editor with the following M code in the formula bar:

```
= Table.Combine({#"Removed Columns2", #"Renamed Columns" })
```

The data table below shows the combined results of the two queries, with columns for Location, Start time, and End time.

	Location	Start time	End time
127	Stockholm	Sun 11:00 p.m.	Mon 2:00 a.m.
128	Suva	Mon 9:00 a.m.	Mon 12:00 Noon
129	Sydney	Mon 7:00 a.m.	Mon 10:00 a.m.
130	Taipei	Mon 5:00 a.m.	Mon 8:00 a.m.
131	Tallinn	Sun 12:00 Midn	Mon 3:00 a.m.
132	Tashkent	Mon 2:00 a.m.	Mon 5:00 a.m.
133	Tegucigalpa	Sun 3:00 p.m.	Sun 6:00 p.m.
134	Tehran	Mon 1:30 a.m.	Mon 4:30 a.m.
135	Tokyo	Mon 6:00 a.m.	Mon 9:00 a.m.
136	Toronto	Sun 5:00 p.m.	Sun 8:00 p.m.
137	Vancouver	Sun 2:00 p.m.	Sun 5:00 p.m.
138	Vienna	Sun 11:00 p.m.	Mon 2:00 a.m.
139	Warsaw	Sun 11:00 p.m.	Mon 2:00 a.m.
140	Washington DC	Sun 5:00 p.m.	Sun 8:00 p.m.
141	Winnipeg	Sun 4:00 p.m.	Sun 7:00 p.m.
142	Yangon	Mon 3:30 a.m.	Mon 6:30 a.m.
143	Zagreb	Sun 11:00 p.m.	Mon 2:00 a.m.
144	Zürich	Sun 11:00 p.m.	Mon 2:00 a.m.

The right-hand pane shows the Query Settings for the 'Appended Query' step, which is highlighted in green. The 'APPLIED STEPS' list includes: Source, Navigation, Changed Type, Removed Columns, Replaced Value, Replaced Value1, Removed Columns2, Removed Columns1, Renamed Columns, and Appended Query.

## Next Steps

It will be long if I mention all the steps in a single post, So stay tuned. Next step would be Grouping rows and applying some transformations to get the desired output. Finally, I will show you how to use parameters in the query and automate the process.

# Power Query Not for BI: Event Date and Time Scheduler – Part 2

Posted by [Reza Rad](#) on May 14, 2016

	Start time	End time	Cities
1	Mon 1:00 AM	Mon 4:00 AM	Accra, Reykjavik
2	Mon 4:00 AM	Mon 7:00 AM	Addis Ababa, Amman, Ankara, Antananarivo, Athens, Baghdad, Beirut, Buchare...
3	Mon 10:30 AM	Mon 1:30 PM	Adelaide, Darwin
4	Mon 2:00 AM	Mon 5:00 AM	Algiers, Casablanca, Dublin, Kinshasa, Lagos, Lisbon, London
5	Mon 7:00 AM	Mon 10:00 AM	Almaty, Dhaka
6	Mon 3:00 AM	Mon 6:00 AM	Amsterdam, Barcelona, Belgrade, Berlin, Brussels, Budapest, Cairo, Cape Town,...
7	Mon 1:00 PM	Mon 4:00 PM	Anadyr, Auckland, Suva
8	Sun 5:00 PM	Sun 8:00 PM	Anchorage
9	Sun 9:00 PM	Sun 12:00 Midn	Asuncion, Atlanta, Boston, Caracas, Columbus, Detroit, Havana, Indianapolis, La...

In the [first part of this series](#), I've explained that Power Query can be used for non-BI scenarios, and here is a real-world use case example; Creating an event date and time scheduler which consider multiple time zones. In a [previous post](#), you've learned how to use Power Query to do some data mashup to build a unified list of cities in different time zones and start and end time of the event in each city. In this post, we will continue steps to build the final result set. If you want to learn more about Power Query, read my Power BI online book; [Power BI from Rookie to Rock Star](#).

## Group Rows

After creating the full list, I want to group rows by their start and end time. I want to create groups of cities where the start time of the event is the same (and end time would be the same).

I start by clicking on the Start time and End time columns, and then I choose Group By from the Home menu under Transform. In Group By window, I can see that Start time and End time already selected as Group By columns (I can add or remove from this window by clicking on – or + buttons if I want to).

I create two new columns for group by result set. One as a count of all rows in each group (this is just out of curiosity to see how many cities I have in each start/end time group). And the other one as all rows for each group (this will generate a table for each group, containing all cities of that group; all cities with same start/end time).

×

Group By

Specify the columns to group by.

Group by

+

Start time

-

End time

-

Group By Columns

New Output Columns

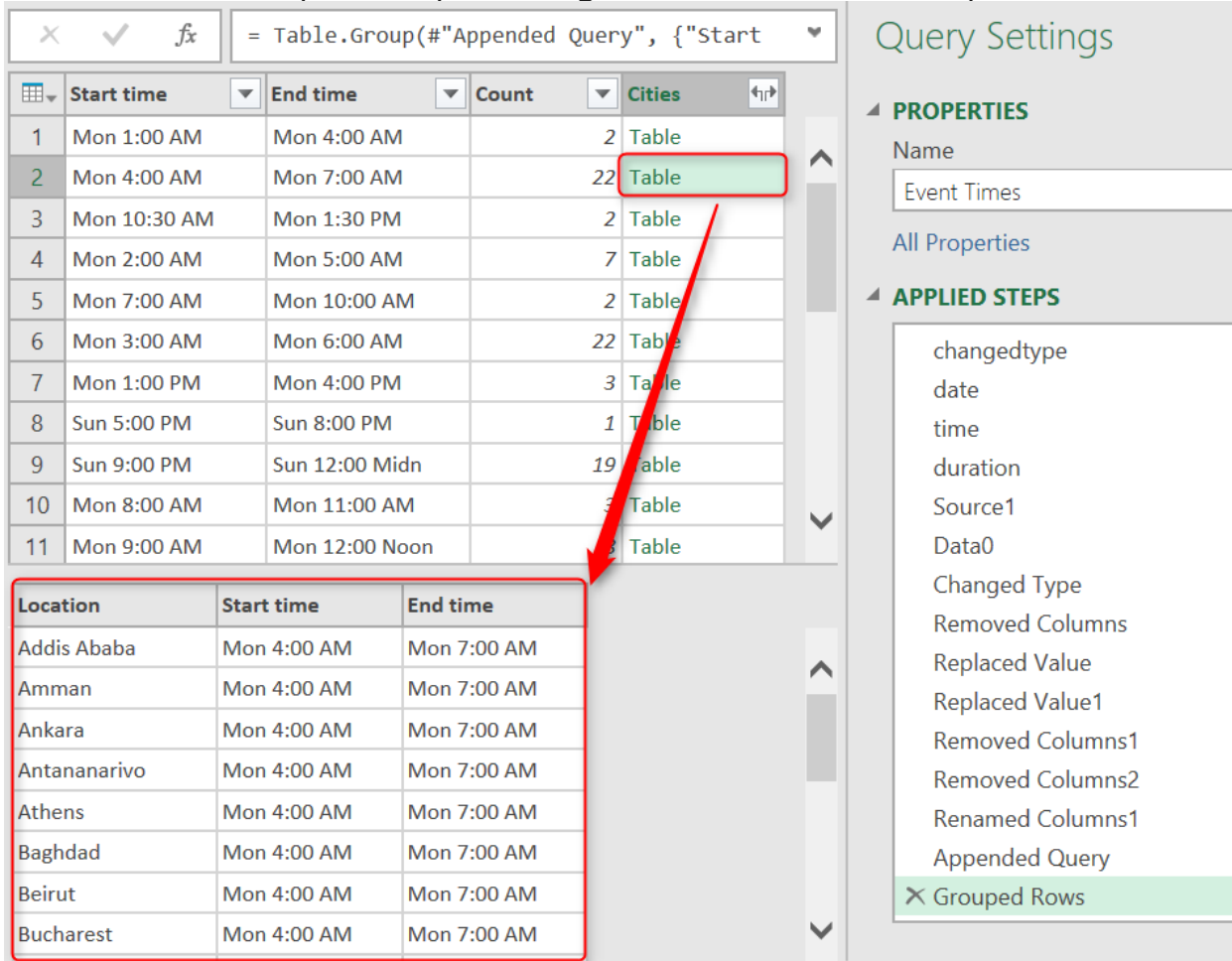
New column name	Operation	Column	+
Count	Count Rows		-
Cities	All Rows		-

OK Cancel

The output of this step will be a table with unique start/end times plus two new columns; count of all rows, and a table embedded column for a list of all cities.

	Start time	End time	Count	Cities
1	Mon 1:00 AM	Mon 4:00 AM	2	Table
2	Mon 4:00 AM	Mon 7:00 AM	22	Table
3	Mon 10:30 AM	Mon 1:30 PM	2	Table
4	Mon 2:00 AM	Mon 5:00 AM	7	Table
5	Mon 7:00 AM	Mon 10:00 AM	2	Table

I can click on a cell in the Cities column to see contents of it (Do not click on the Table word, click on an empty space in that cell. If you click on the Table word a new subsequent step will be generated to fetch that specific table).



The screenshot shows the Power BI interface with a table of event times and a list of cities. The table has columns: Start time, End time, Count, and Cities. The Cities column contains links to tables. A red box highlights a cell in the Cities column, and a red arrow points to a detailed view of the cities for that time slot.

Start time	End time	Count	Cities
Mon 1:00 AM	Mon 4:00 AM	2	Table
Mon 4:00 AM	Mon 7:00 AM	22	Table
Mon 10:30 AM	Mon 1:30 PM	2	Table
Mon 2:00 AM	Mon 5:00 AM	7	Table
Mon 7:00 AM	Mon 10:00 AM	2	Table
Mon 3:00 AM	Mon 6:00 AM	22	Table
Mon 1:00 PM	Mon 4:00 PM	3	Table
Sun 5:00 PM	Sun 8:00 PM	1	Table
Sun 9:00 PM	Sun 12:00 Midn	19	Table
Mon 8:00 AM	Mon 11:00 AM	3	Table
Mon 9:00 AM	Mon 12:00 Noon	3	Table

Location	Start time	End time
Addis Ababa	Mon 4:00 AM	Mon 7:00 AM
Amman	Mon 4:00 AM	Mon 7:00 AM
Ankara	Mon 4:00 AM	Mon 7:00 AM
Antananarivo	Mon 4:00 AM	Mon 7:00 AM
Athens	Mon 4:00 AM	Mon 7:00 AM
Baghdad	Mon 4:00 AM	Mon 7:00 AM
Beirut	Mon 4:00 AM	Mon 7:00 AM
Bucharest	Mon 4:00 AM	Mon 7:00 AM

Query Settings

**PROPERTIES**

Name  
Event Times

[All Properties](#)

**APPLIED STEPS**

- changedtype
- date
- time
- duration
- Source1
- Data0
- Changed Type
- Removed Columns
- Replaced Value
- Replaced Value1
- Removed Columns1
- Removed Columns2
- Renamed Columns1
- Appended Query
- Grouped Rows**

## Concatenate Rows

The content of each cell in the above screenshot is a table containing all cities in that group. For example, for the selected cell above I have a table of 22 cities.

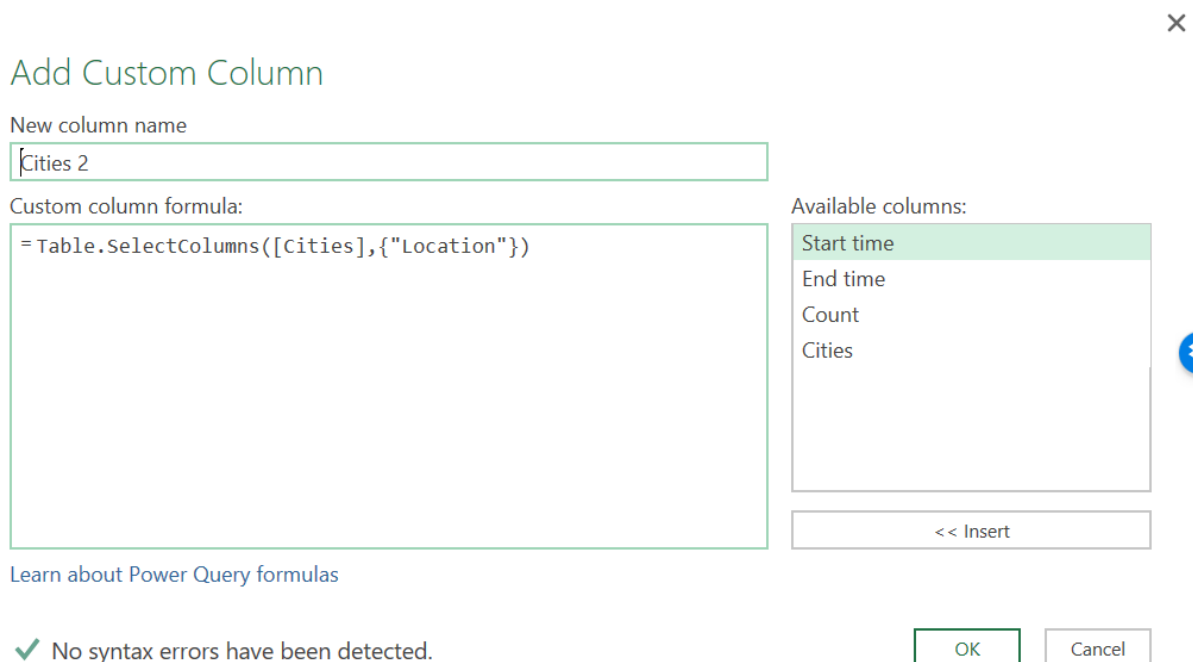
So now my next step is to create a string concatenated list of all these city names, like Addis Ababa, Amman, Ankara, Antananarivo, Athens...

I can transform this column to be concatenated list all in one, but for simplicity and ease of understanding I'll do that step by step by adding a custom column in each step;

### Step 1 to Concatenate: Select Single Column

As the first step to concatenate, I'll fetch a single column from the sub-table which is Location.

I create a new custom column by click on the Add Column menu and then Add Custom Column icon. This will open Add Custom Column window for me where I can define a new column with the name and the expression to calculate the new column.



Add Custom Column

New column name  
Cities 2

Custom column formula:  
= Table.SelectColumns([Cities],{"Location"})

Available columns:  
Start time  
End time  
Count  
Cities

<< Insert

Learn about Power Query formulas

✓ No syntax errors have been detected.

OK Cancel

I name this as Cities 2, and the expression as  
*1 = Table.SelectColumns([Cities],{"Location"})*

Table.SelectColumns only fetches specific columns of the table that we mention in the {}. Here I only fetched Location column so that it will be a single column table as a result;



fx = Table.AddColumn(#"Grouped Rows", "Cities 2", each Table

	Start time	End time	Count	Cities	Cities 2
1	Mon 1:00 AM	Mon 4:00 AM	2	Table	Table
2	Mon 4:00 AM	Mon 7:00 AM	22	Table	Table
3	Mon 10:30 AM	Mon 1:30 PM	2	Table	Table
4	Mon 2:00 AM	Mon 5:00 AM	7	Table	Table
5	Mon 7:00 AM	Mon 10:00 AM	2	Table	Table
6	Mon 3:00 AM	Mon 6:00 AM	22	Table	Table
7	Mon 1:00 PM	Mon 4:00 PM	3	Table	Table
8	Sun 5:00 PM	Sun 8:00 PM	1	Table	Table
9	Sun 9:00 PM	Sun 12:00 Midn	19	Table	Table
10	Mon 8:00 AM	Mon 11:00 AM	3	Table	Table
11	Mon 9:00 AM	Mon 12:00 Noon	8	Table	Table

Location
Addis Ababa
Amman
Ankara
Antananarivo
Athens
Baghdad
Beirut
Bucharest

## Step 2 to Concatenate: Rows to Columns

For concatenation, I want to use a method that needs values to be on COLUMNS not on rows. So I have to transform the inner table to show values in Columns rather than rows. Fortunately, it is easy to do with Table.Transpose. I add a new custom column with this expression:

```
1 = Table.Transpose([Cities 2])
```

This simple expression will turn my single-column row values into multiple columns in a single row. For using Transpose, All you need is the table name.

The table should be single columned. And then it transforms it into rows. Column headers would be auto-named as Column1, Column2,... So this would be the result;

Advanced Editor formula bar: `= Table.AddColumn("#Added Custom", "Cities 4", each Table.Transpose([Cities`

	Start time	End time	Count	Cities	Cities 2	Cities 4
1	Mon 1:00 AM	Mon 4:00 AM	2	Table	Table	Table
2	Mon 4:00 AM	Mon 7:00 AM	22	Table	Table	Table
3	Mon 10:30 AM	Mon 1:30 PM	2	Table	Table	Table
4	Mon 2:00 AM	Mon 5:00 AM	7	Table	Table	Table
5	Mon 7:00 AM	Mon 10:00 AM	2	Table	Table	Table
6	Mon 3:00 AM	Mon 6:00 AM	22	Table	Table	Table
7	Mon 1:00 PM	Mon 4:00 PM	3	Table	Table	Table
8	Sun 5:00 PM	Sun 8:00 PM	1	Table	Table	Table
9	Sun 9:00 PM	Sun 12:00 Midn	19	Table	Table	Table
10	Mon 8:00 AM	Mon 11:00 AM	3	Table	Table	Table
11	Mon 9:00 AM	Mon 12:00 Noon	8	Table	Table	Table

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8
Addis Ababa	Amman	Ankara	Antananarivo	Athens	Baghdad	Beirut	Bucharest

### A bit of code behind

You might wonder what is happening when you add a custom column which is based on an expression which uses an existing column. With a glance at Advanced Editor you can see this is the script line generated for adding this custom column;

```
1 #"Added Custom2" = Table.AddColumn("#Added Custom", "Cities 4", each
  Table.Transpose([Cities 2]))
```

Table.AddColumn adds a new column with name and expression. This function gets three parameters;

- Name of the table to add a column to it: `"#Added Custom"`
- Name of the new column to generate: `Cities 4`
- The expression to populate the new column: `each Table.Transpose([Cities 2])`

*EACH* is a singleton function that applies on every row of the main table and produces the result of expression for that row into the new column.

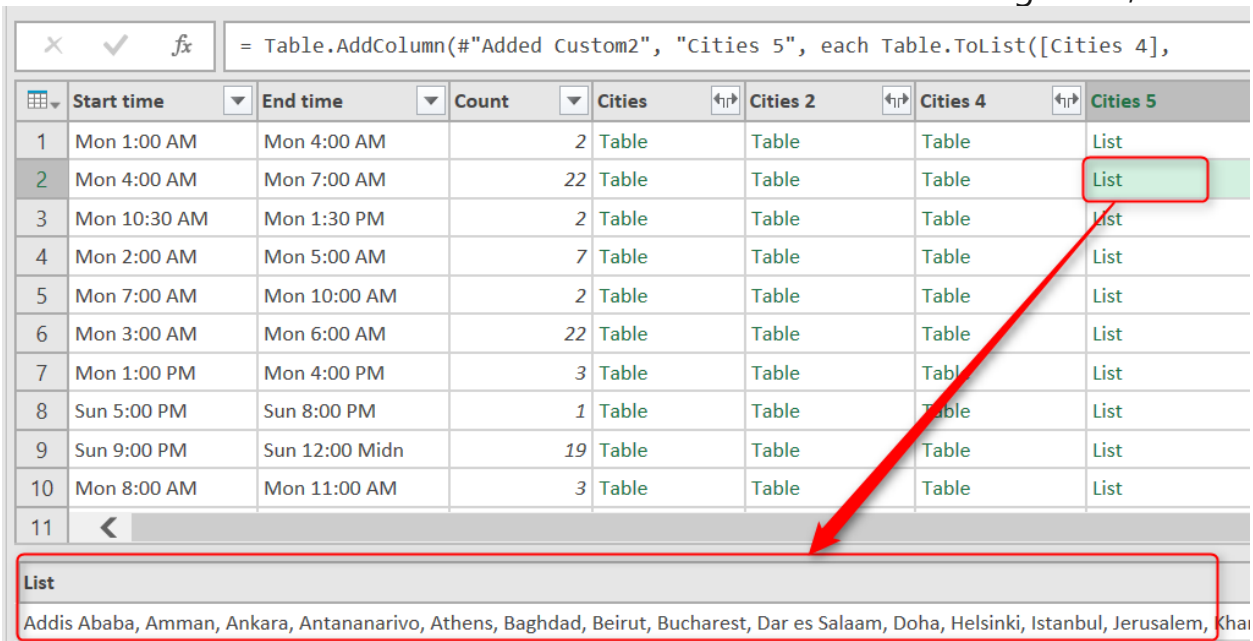
### Step 3 to Concatenate: Concatenate to List

Now that I have values in multiple columns I can concatenate them all into one string with Table.ToList function which converts a table to List. This function can concatenate all columns of a table into one column (because List is a single columned data structure).

The actual concatenation happens by Combiner function; Combiner.CombineTextByDelimiter(", ") which concatenate values with a delimiter which I set to be comma. So here is the expression for my new custom column:

```
1 =Table.ToList([Cities 4],
2
3 Combiner.CombineTextByDelimiter(", ")
4
5)
```

And the result would be a list which contains concatenated string value;



The screenshot shows the Power Query Editor interface. The formula bar at the top contains the expression: `= Table.AddColumn(#"Added Custom2", "Cities 5", each Table.ToList([Cities 4],`. Below the formula bar is a table with the following columns: Start time, End time, Count, Cities, Cities 2, Cities 4, and Cities 5. The table contains 10 rows of data. The Cities 5 column contains the value 'List' for all rows. A red box highlights the 'List' value in row 2, and a red arrow points to the expanded list below the table.

	Start time	End time	Count	Cities	Cities 2	Cities 4	Cities 5
1	Mon 1:00 AM	Mon 4:00 AM	2	Table	Table	Table	List
2	Mon 4:00 AM	Mon 7:00 AM	22	Table	Table	Table	List
3	Mon 10:30 AM	Mon 1:30 PM	2	Table	Table	Table	List
4	Mon 2:00 AM	Mon 5:00 AM	7	Table	Table	Table	List
5	Mon 7:00 AM	Mon 10:00 AM	2	Table	Table	Table	List
6	Mon 3:00 AM	Mon 6:00 AM	22	Table	Table	Table	List
7	Mon 1:00 PM	Mon 4:00 PM	3	Table	Table	Table	List
8	Sun 5:00 PM	Sun 8:00 PM	1	Table	Table	Table	List
9	Sun 9:00 PM	Sun 12:00 Midn	19	Table	Table	Table	List
10	Mon 8:00 AM	Mon 11:00 AM	3	Table	Table	Table	List

Below the table, the expanded list is shown:

```
List
Addis Ababa, Amman, Ankara, Antananarivo, Athens, Baghdad, Beirut, Bucharest, Dar es Salaam, Doha, Helsinki, Istanbul, Jerusalem, Kha
```

### Step 4 to Concatenate: Expand

All I have to now is to expand the list to its values. Expanding a column means expanding its underneath structure. Expand column appears when you have a

multi-value structure in each cell. Multi-value structures in Power Query can be; Table, List, and Record. You can find Expand Column Icon simply beside the column heading and click on it.

✓

fx

= Table.AddColumn("#Added Custom2", "Cities 5", each Table.ToList([Cities 4],

nd time

Count

Cities

Cities 2

Cities 4

Cities 5

1

Mon 4:00 AM

2

Table

Table

Table

List

2

Mon 7:00 AM

22

Table

Table

Table

List

3

Mon 1:30 PM

2

Table

Table

Table

List

4

Mon 5:00 AM

7

Table

Table

Table

List

5

Mon 10:00 AM

2

Table

Table

Table

List

6

Mon 6:00 AM

22

Table

Table

Table

List

After expanding I can see concatenated text values in my main table as below;

fx		= Table.ExpandListColumn("#Added Custom3", "Cities 5")				
Start time	End time	Count	Cities	Cities 2	Cities 4	Cities 5
1 Mon 1:00 AM	Mon 4:00 AM	2	Table	Table	Table	Accra, Reykjavik
2 Mon 4:00 AM	Mon 7:00 AM	22	Table	Table	Table	Addis Ababa, Amman, Ankara, Antananarivo, Athens, Baghdad, Beirut,...
3 Mon 10:30 AM	Mon 1:30 PM	2	Table	Table	Table	Adelaide, Darwin
4 Mon 2:00 AM	Mon 5:00 AM	7	Table	Table	Table	Algiers, Casablanca, Dublin, Kinshasa, Lagos, Lisbon, London

## Final Polish

Now I have Start/End time for each time zone, and concatenated list of all cities in that time zone, so I do a bit of polishing with removing extra columns. I Only keep the Start Time, End Time, and Cities 5. And renaming Cities 5 to Cities.

	Start time	End time	Cities
1	Mon 1:00 AM	Mon 4:00 AM	Accra, Reykjavik
2	Mon 4:00 AM	Mon 7:00 AM	Addis Ababa, Amman, Ankara, Antananarivo, Athens, Baghdad, Beirut, Buchare...
3	Mon 10:30 AM	Mon 1:30 PM	Adelaide, Darwin
4	Mon 2:00 AM	Mon 5:00 AM	Algiers, Casablanca, Dublin, Kinshasa, Lagos, Lisbon, London
5	Mon 7:00 AM	Mon 10:00 AM	Almaty, Dhaka
6	Mon 3:00 AM	Mon 6:00 AM	Amsterdam, Barcelona, Belgrade, Berlin, Brussels, Budapest, Cairo, Cape Town,...
7	Mon 1:00 PM	Mon 4:00 PM	Anadyr, Auckland, Suva
8	Sun 5:00 PM	Sun 8:00 PM	Anchorage
9	Sun 9:00 PM	Sun 12:00 Midn	Asuncion, Atlanta, Boston, Caracas, Columbus, Detroit, Havana, Indianapolis, La...

## Next Steps

I've made my result set as I want with Power Query so far, but still one step left, which is using parameters and Automation; I'll be using an Excel table as input parameters to set my local event date and time, and then Power Query will use those input parameters to refresh the whole result set. Stay tuned for the next post to see how it can be done.

# Power Query Not for BI: Event Date and Time Scheduler – Part 3

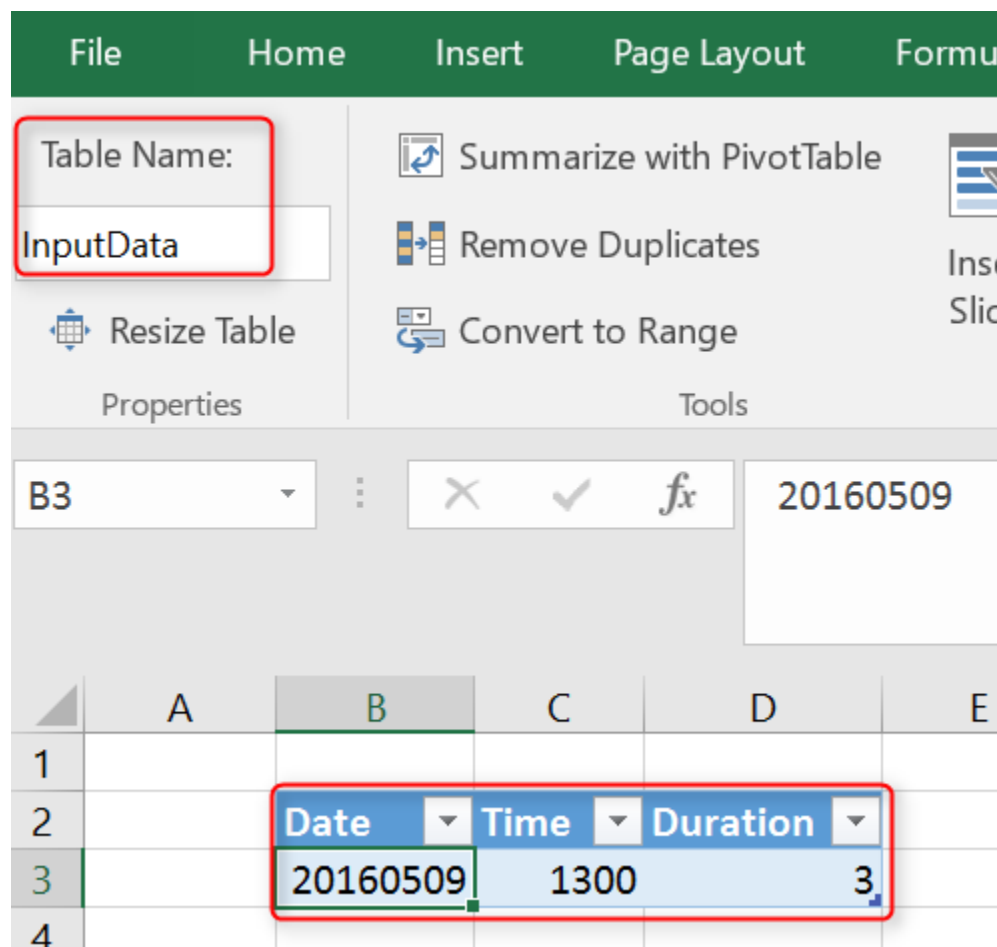
Posted by [Reza Rad](#) on May 16, 2016

<div><div>✕</div><div>✓</div><div><i>fx</i></div></div> <div>= Excel.CurrentWorkbook(){[Name="InputData"]}[Content]</div>			
<div>📅</div>	Date	Time	Duration
1	20160509	1300	3

Previously You have seen in [Part 1](#) and [Part 2](#) of this series, that how you can use Power Query to do data transformation and turn the result of event date/time scheduler of [timeanddate.com](#) website to a better format. We've used a number of basic transformations, and some functions such as `Table.Transpose` to fetch desired output. This part is the last step which we will use parameters to make our query dynamic. At the end of this part, you will learn how to automate your Power Query solution, with a change in parameters in an Excel table, and hit the refresh button a new result set with the desired format will be created. To learn more about Power Query, read my Power BI online book; [Power BI from Rookie to Rock Star](#).

## Using Parameters

We've made all the data manipulation and mash up, and the output result set is exactly as desired. However, one step is left: Using Parameters and making this query dynamic or automate the process. For parameters, I use an Excel table as my configuration table. So I create a table as below;



This table has three columns: Date, Time, and Duration. I separated the date and time for the simplicity of this example: date to be formatted as YYYYMMDD, and Time as HHMM, and duration as an integer value illustrating hours.

The configuration above means the event starts at 9<sup>th</sup> of May 2016, at 1:00 pm New Zealand time (this is what my local time is), with a duration of 3 hours. I named this table as InputData.

I can fetch this table in my Query Editor with a simple M script line as below;  
`1 input = Excel.CurrentWorkbook(){[Name="InputData"]}[Content]`

Excel.CurrentWorkbook reads a table from the current Excel workbook. Name of the table passed as an input parameter. And [Content] will load the content

field of the table (which is a table by itself of all columns and rows of the Excel table) into the variable named input.

<div> <div>✕ ✓ <i>fx</i></div> <div>= Excel.CurrentWorkbook(){[Name="InputData"]}[Content]</div> </div>			
	Date	Time	Duration
1	20160509	1300	3

I can then read the first record of this table into a variable called inputrecord. And use it for fetching each parameter later on. Here is my inputrecord

<div> <div>✕ ✓ <i>fx</i></div> <div>= Table.First(input)</div> </div>	
Date	20160509
Time	1300
Duration	3

*1 inputrecord=Table.First(input),*

Now I can read Date value easily by the expression below:

*1 date=Text.From(inputrecord[Date]),*

For Time I do also use a PadStart with 0 to generate values like 0400 when the time is 4 AM. The reason is that TimeandDate.com asks for a full HHMM string with leading 0 to work correctly. Duration calculation is simple as well.

*1 time=Text.PadStart(Text.From(inputrecord[Time]),4,"0"),*

*2*

*3 duration=Text.From(inputrecord[Duration]),*

Now that I have all three parameters in variables named; date, time, and duration. I can use them in generating web URL to pass to timeanddate.com website as below;

*1 WebURL="http://www.timeanddate.com/worldclock/fixedtime.html?msg=Power+BI+Training&iso="&date&"T"&time&"&p1=22&ah="&duration,*

String concatenation character in Power Query is: &. And expression above generates the web URL for the date/time of the event with a local time of



Auckland, New Zealand with a duration specified. All I have to do now is to fetch data from this URL with the WebURL variable. So I'll change the line below;

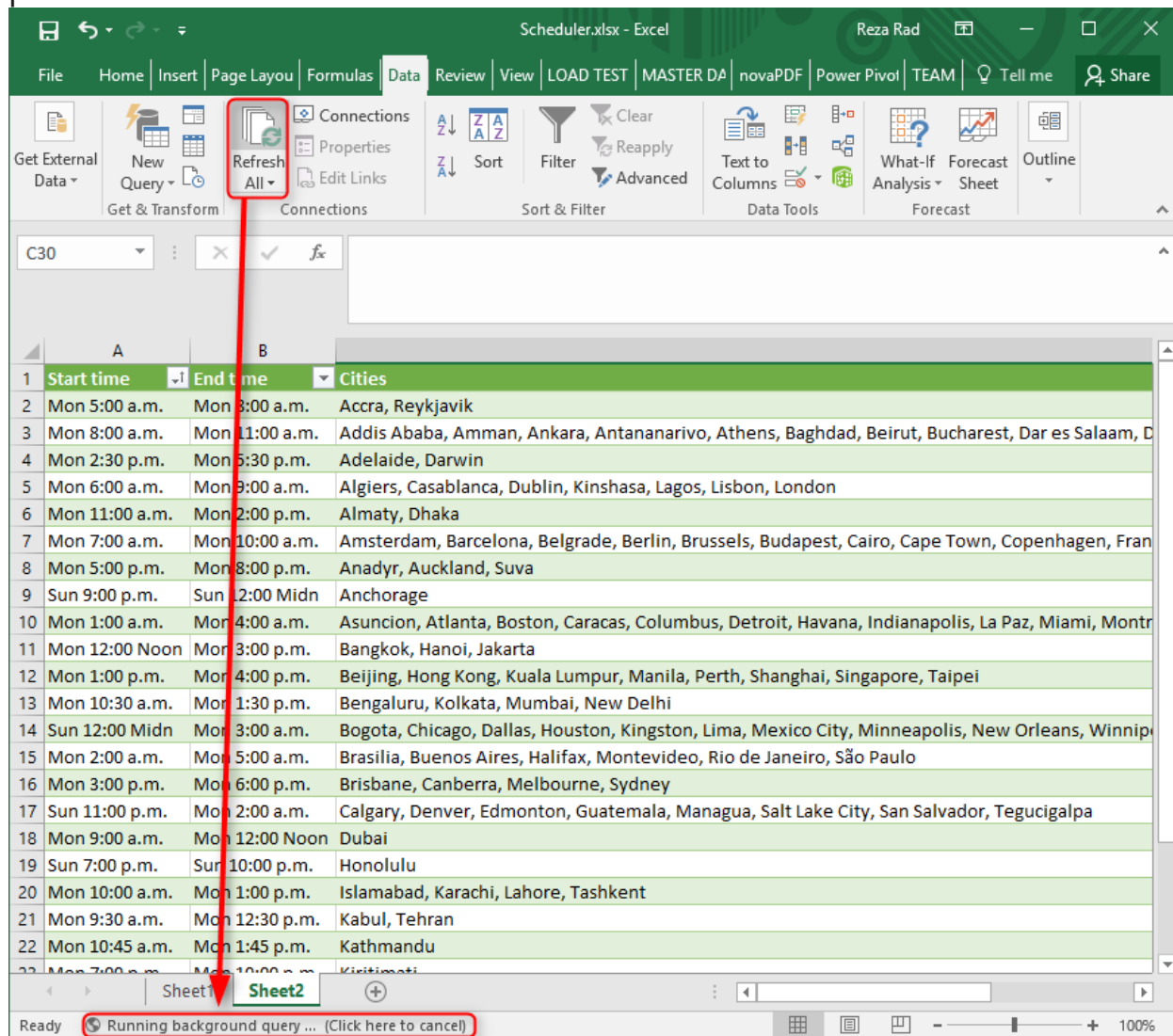
*Source1 =*

*Web.Page(Web.Contents("http://www.timeanddate.com/worldclock/fixedtime.html?msg=Power+BI+Training&iso=20160509T1300&p1=22&ah=3"))*,

To this:

*1 Source1 = Web.Page(Web.Contents(WebURL))*,

Now I have a dynamic query based on parameters. Let's change the parameters in Excel and Refresh the data.



The screenshot shows the Microsoft Excel interface with the 'Data' tab selected. The 'Refresh All' button in the 'Connections' group is highlighted with a red box. A red arrow points from this button to the 'Refresh' button in the status bar at the bottom, which is also highlighted with a red box. The status bar shows 'Running background query ... (Click here to cancel)'.

Start time	End time	Cities
Mon 5:00 a.m.	Mon 8:00 a.m.	Accra, Reykjavik
Mon 8:00 a.m.	Mon 11:00 a.m.	Addis Ababa, Amman, Ankara, Antananarivo, Athens, Baghdad, Beirut, Bucharest, Dar es Salaam, D
Mon 2:30 p.m.	Mon 5:30 p.m.	Adelaide, Darwin
Mon 6:00 a.m.	Mon 9:00 a.m.	Algiers, Casablanca, Dublin, Kinshasa, Lagos, Lisbon, London
Mon 11:00 a.m.	Mon 2:00 p.m.	Almaty, Dhaka
Mon 7:00 a.m.	Mon 10:00 a.m.	Amsterdam, Barcelona, Belgrade, Berlin, Brussels, Budapest, Cairo, Cape Town, Copenhagen, Fran
Mon 5:00 p.m.	Mon 8:00 p.m.	Anadyr, Auckland, Suva
Sun 9:00 p.m.	Sun 12:00 Midn	Anchorage
Mon 1:00 a.m.	Mon 4:00 a.m.	Asuncion, Atlanta, Boston, Caracas, Columbus, Detroit, Havana, Indianapolis, La Paz, Miami, Montr
Mon 12:00 Noon	Mon 3:00 p.m.	Bangkok, Hanoi, Jakarta
Mon 1:00 p.m.	Mon 4:00 p.m.	Beijing, Hong Kong, Kuala Lumpur, Manila, Perth, Shanghai, Singapore, Taipei
Mon 10:30 a.m.	Mon 1:30 p.m.	Bengaluru, Kolkata, Mumbai, New Delhi
Sun 12:00 Midn	Mon 3:00 a.m.	Bogota, Chicago, Dallas, Houston, Kingston, Lima, Mexico City, Minneapolis, New Orleans, Winnip
Mon 2:00 a.m.	Mon 5:00 a.m.	Brasilia, Buenos Aires, Halifax, Montevideo, Rio de Janeiro, São Paulo
Mon 3:00 p.m.	Mon 6:00 p.m.	Brisbane, Canberra, Melbourne, Sydney
Sun 11:00 p.m.	Mon 2:00 a.m.	Calgary, Denver, Edmonton, Guatemala, Managua, Salt Lake City, San Salvador, Tegucigalpa
Mon 9:00 a.m.	Mon 12:00 Noon	Dubai
Sun 7:00 p.m.	Sun 10:00 p.m.	Honolulu
Mon 10:00 a.m.	Mon 1:00 p.m.	Islamabad, Karachi, Lahore, Tashkent
Mon 9:30 a.m.	Mon 12:30 p.m.	Kabul, Tehran
Mon 10:45 a.m.	Mon 1:45 p.m.	Kathmandu
Mon 7:00 a.m.	Mon 10:00 a.m.	Kiritimati

When I refresh the data, I can see in the status bar that Power Query is refreshing the whole data set based on input parameters (illustrated in the screenshot above).

And here is my final result:

	A	B	
1	Start time	End time	Cities
2	Mon 3:30 a.m.	Mon 6:30 a.m.	Accra, Reykjavik
3	Mon 6:30 a.m.	Mon 9:30 a.m.	Addis Ababa, Amman, Ankara, Antananarivo
4	Mon 1:00 p.m.	Mon 4:00 p.m.	Adelaide, Darwin
5	Mon 4:30 a.m.	Mon 7:30 a.m.	Algiers, Casablanca, Dublin, Kinshasa, Lagos
6	Mon 9:30 a.m.	Mon 12:30 p.m.	Almaty, Dhaka
7	Mon 5:30 a.m.	Mon 8:30 a.m.	Amsterdam, Barcelona, Belgrade, Berlin, Br
8	Mon 3:30 p.m.	Mon 6:30 p.m.	Anadyr, Auckland, Suva
9	Sun 7:30 p.m.	Sun 10:30 p.m.	Anchorage
10	Sun 11:30 p.m.	Mon 2:30 a.m.	Asuncion, Atlanta, Boston, Caracas, Columb
11	Mon 10:30 a.m.	Mon 1:30 p.m.	Bangkok, Hanoi, Jakarta
12	Mon 11:30 a.m.	Mon 2:30 p.m.	Beijing, Hong Kong, Kuala Lumpur, Manila, F
13	Mon 9:00 a.m.	Mon 12:00 Noon	Bengaluru, Kolkata, Mumbai, New Delhi
14	Sun 10:30 p.m.	Mon 1:30 a.m.	Bogota, Chicago, Dallas, Houston, Kingston,
15	Mon 12:30 a.m.	Mon 3:30 a.m.	Brasilia, Buenos Aires, Halifax, Montevideo,
16	Mon 1:30 p.m.	Mon 4:30 p.m.	Brisbane, Canberra, Melbourne, Sydney
17	Sun 9:30 p.m.	Mon 12:30 a.m.	Calgary, Denver, Edmonton, Guatemala, Ma
18	Mon 7:30 a.m.	Mon 10:30 a.m.	Dubai
19	Sun 5:30 p.m.	Sun 8:30 p.m.	Honolulu
20	Mon 8:30 a.m.	Mon 11:30 a.m.	Islamabad, Karachi, Lahore, Tashkent
21	Mon 8:00 a.m.	Mon 11:00 a.m.	Kabul, Tehran
22	Mon 9:15 a.m.	Mon 12:15 p.m.	Kathmandu

Here is the full script of the example above:

*let*

*// read Input Data*

*input = Excel.CurrentWorkbook()[{Name="InputData"}][Content],*

*inputrecord=Table.First(input),*

```
date=Text.From(inputrecord[Date]),
```

```
time=Text.PadStart(Text.From(inputrecord[Time]),4,"0"),
```

```
duration=Text.From(inputrecord[Duration]),
```

```
WebURL="http://www.timeanddate.com/worldclock/fixedtime.html?msg=Power+BI+Training&iso="&date&"T"&time&"&p1=22&ah="&duration,
```

```
Source = Web.Page(Web.Contents(WebURL)),
```

```
Data0 = Source{0}[Data],
```

```
#"Changed Type" = Table.TransformColumnTypes(Data0,{{"Header", type text}, {"Location", type text}, {"Start time", type text}, {"End time", type text}, {"Location2", type text}, {"Start time2", type text}, {"End time2", type text}}),
```

```
#"Removed Columns" = Table.RemoveColumns(#"Changed Type",{"Header"}),
```

```
#"Replaced Value" = Table.ReplaceValue(#"Removed Columns", "*", "", Replacer.ReplaceText,{"Location"}),
```

```
#"Replaced Value1" = Table.ReplaceValue(#"Replaced Value", "*", "", Replacer.ReplaceText,{"Location2"}),
```

```
#"Removed Columns2" = Table.RemoveColumns(#"Replaced Value1",{"Location2", "Start time2", "End time2"}),
```

```
#"Removed Columns1" = Table.RemoveColumns(#"Replaced Value1",{"Location", "Start time", "End time"}),
```

```
#"Renamed Columns" = Table.RenameColumns(#"Removed Columns1",{{"Location2", "Location"}, {"Start time2", "Start time"}, {"End time2", "End time"}}),
```

```
#"Appended Query" = Table.Combine({#"Removed Columns2",  
#"Renamed Columns" }},
```

```
#"Grouped Rows" = Table.Group(#"Appended Query", {"Start time", "End  
time"}, {{"Count", each Table.RowCount(_), type number}, {"Cities", each _,  
type table}}),
```

```
#"Added Custom" = Table.AddColumn(#"Grouped Rows", "Cities 2", each  
Table.SelectColumns([Cities],{"Location"})),
```

```
#"Added Custom2" = Table.AddColumn(#"Added Custom", "Cities 4", each  
Table.Transpose([Cities 2])),
```

```
#"Added Custom3" = Table.AddColumn(#"Added Custom2", "Cities 5", each  
Table.ToList([Cities 4],
```

```
Combiner.CombineTextByDelimiter(", ")
```

```
)),
```

```
#"Expanded Cities 5" = Table.ExpandListColumn(#"Added Custom3", "Cities  
5"),
```

```
#"Removed Other Columns" = Table.SelectColumns(#"Expanded Cities  
5",{"Start time", "End time", "Cities 5"}),
```

```
Final = Table.RenameColumns(#"Removed Other Columns",{{"Cities 5",  
"Cities"}})
```

```
in
```

```
Final
```

## Summary

You have seen how Power Query can be useful for even none BI use case scenarios. You've learned how to use Power Query to get input parameters dynamically from Excel spreadsheet and call a web URL with that, fetches the data set, and do data transformations. This was a real-world use case that showed you some strengths of Power Query. No matter if you are a BI developer or not, you will find use cases of Power Query in many situations.

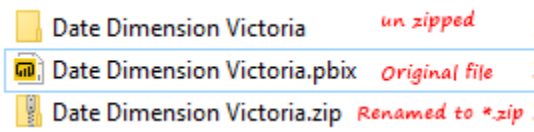
## Call to Action

Tell me real use case scenarios that you have used Power Query for, how did you find Power Query in solving that challenge?

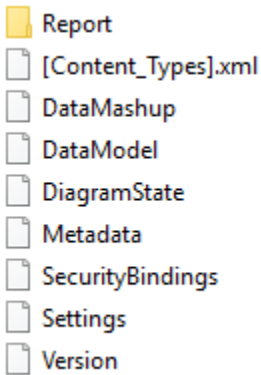
## Part VIII: A Tool to Help: Power BI Helper







Files under this folder are;



Here is a very brief explanation of each file or folder;

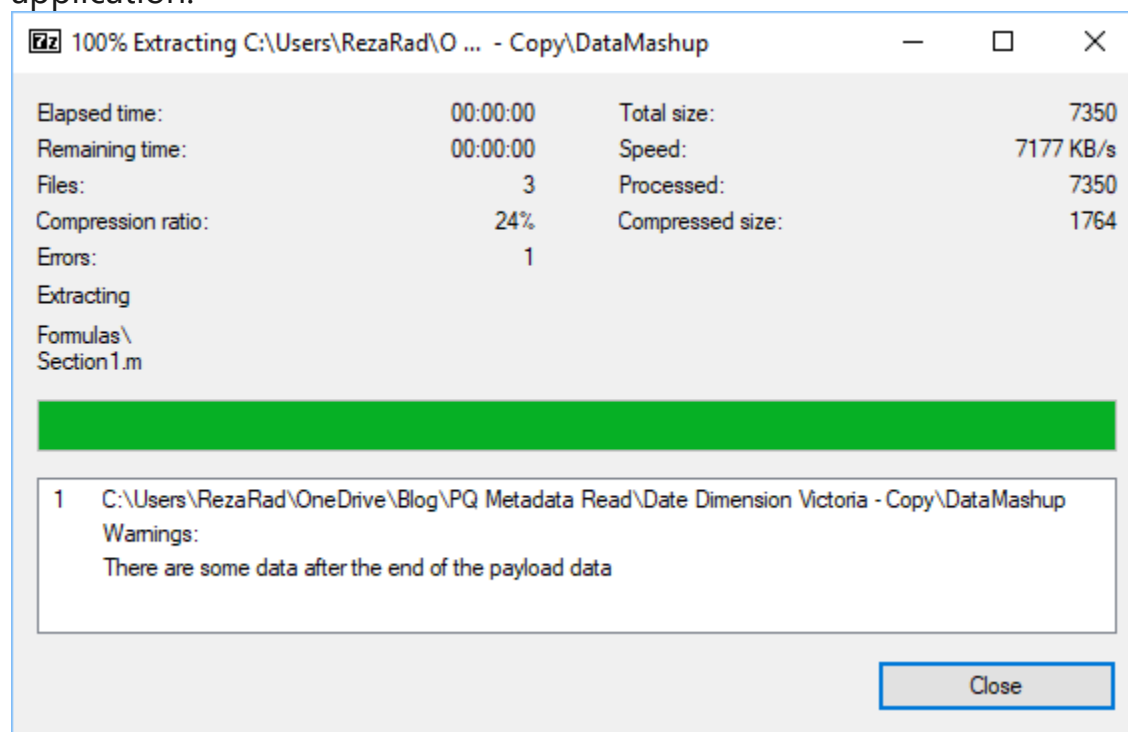
- Report Folder: Layout of the Power BI visual reports (Including two files in the folder)
- [Content\_Types].xml: Includes the content structure of this folder in XML format (older PBIX files might have different content structure)
- DataMashup: This is the file which is the main topic of this blog post. It includes everything related to Power Query side of the file; M scripts, the structure of queries, parameters, and functions.
- DataModel: This includes the data in the model (in a compressed format).
- DiagramState: It seems it stores the table and Matrix information.
- Metadata: Contains all labels (Names). In Power BI Desktop, everything has a GUID assigned to it. What we see as a table name, has a GUID assigned to it. This file is a mapping of GUIDs and names that we see.
- SecurityBindings: I believe this is Row Level Security configuration in the Power BI stored in binary format.
- Settings: I believe it is some of the settings applied in Power BI settings menu options.
- Version: Version of the file.

In the future, I'll write some other posts with a detailed explanation of every file and folder. For now, let's focus on DataMashaup File.






## DataMashup File: Everything You Need

Talking about Power Query; DataMashup file is all you need. It includes everything from the structure of queries, tables, parameters, list, to the actual M scripts behind the scene. You can Fetch all of this information from this single file. Let's look at the structure of this file. If you open this file with a text editor. You will see some binary things first (which are related to the zipped nature of this file), and also some XML information. Yes, this is a zipped file. Let's start with unzipping it into a folder. I've done that with the 7-zip application.



In the folder you will have these files:

-  Config
-  Formulas
-  [Content\_Types].xml

Here are details of each file;


- [Content\_Types].xml: explains the content of this section, very high level information. it is just saying that M script is somewhere in this folder

```
<?xml version="1.0" encoding="utf-8"?>
<Types xmlns="http://schemas.openxmlformats.org/package/2006/content-types">
  <Default Extension="xml" ContentType="text/xml" />
  <Default Extension="m" ContentType="application/x-ms-m" />
</Types>
```

- **Config Folder:** Includes one file: Package.xml, which is the information about settings of Power Query. For example the culture, versioning, and some other configurations.

```
<?xml version="1.0" encoding="utf-8"?>
<Package xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Version>2.46.4732.721</Version>
  <MinVersion>1.5.3296.0</MinVersion>
  <Culture>en-NZ</Culture>
</Package>
```

- **Formulas Folder:** Includes M scripts! Files with \*.m extension.

OneDrive > Blog > PQ Metadata Read > Date Dimension Victoria - Copy > DataMashup~ > Formulas				
Name	Date modified	Type	Size	
 Section1.m	30/05/2017 6:08 PM	M File	7 KB	

You might end up having more than one file if you have more than 1 section, but this is unlikely to happen in usual cases. Usually, everything you do in Query Editor goes under a single section. If you are an M geek and know how to write multiple sections, then you might end up with having multiple files. I will write another post in the future explaining Sections and their usages. For now, let's look at this \*.m file with a text editor:

## M Script File

Section1.m File gives you all M script in the well formatted text;

```

section Section1;

shared Sheet1 = let
    Source = Excel.Workbook(File.Contents("C:\Users\RezaRad\OneDrive\2-Advanced Power Query_RADACAD\Demo\Book1.xlsx"), null, true),
    Sheet1_Sheet = Source{[Item="Sheet1",Kind="Sheet"]}[Data],
    #"Promoted Headers" = Table.PromoteHeaders(Sheet1_Sheet, [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Start Year", Int64.Type}, {"Financial Month Start", Int64.Type}, {"Years to Generate", Int64.Type}})
in
    #"Changed Type";

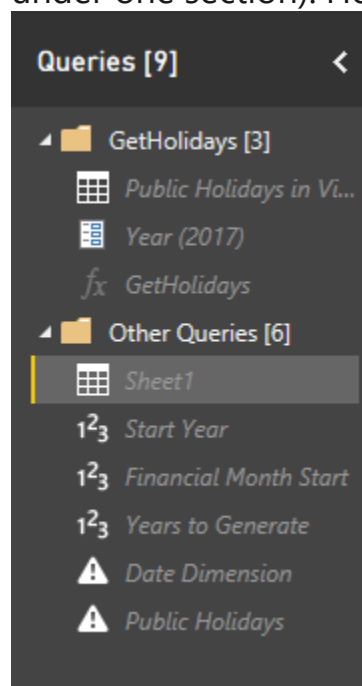
shared #"Start Year" = let
    Source = Excel.Workbook(File.Contents("C:\Users\RezaRad\OneDrive\2-Advanced Power Query_RADACAD\Demo\Book1.xlsx"), null, true),
    Sheet1_Sheet = Source{[Item="Sheet1",Kind="Sheet"]}[Data],
    #"Promoted Headers" = Table.PromoteHeaders(Sheet1_Sheet, [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Start Year", Int64.Type}, {"Financial Month Start", Int64.Type}, {"Years to Generate", Int64.Type}}),
    #"Start Year1" = #"Changed Type"{0}[Start Year]
in
    #"Start Year1";

shared #"Financial Month Start" = let
    Source = Excel.Workbook(File.Contents("C:\Users\RezaRad\OneDrive\2-Advanced Power Query_RADACAD\Demo\Book1.xlsx"), null, true),
    Sheet1_Sheet = Source{[Item="Sheet1",Kind="Sheet"]}[Data],
    #"Promoted Headers" = Table.PromoteHeaders(Sheet1_Sheet, [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Start Year", Int64.Type}, {"Financial Month Start", Int64.Type}, {"Years to Generate", Int64.Type}}),
    #"Financial Month Start1" = #"Changed Type"{0}[Financial Month Start]
in
    #"Financial Month Start1";

shared #"Years to Generate" = let
    Source = Excel.Workbook(File.Contents("C:\Users\RezaRad\OneDrive\2-Advanced Power Query_RADACAD\Demo\Book1.xlsx"), null, true),
    Sheet1_Sheet = Source{[Item="Sheet1",Kind="Sheet"]}[Data],
    #"Promoted Headers" = Table.PromoteHeaders(Sheet1_Sheet, [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Start Year", Int64.Type}, {"Financial Month Start", Int64.Type}, {"Years to Generate", Int64.Type}}),
    #"Years to Generate1" = #"Changed Type"{0}[Years to Generate]
in
    #"Years to Generate1";

```

If you need to send your M scripts to someone, this is the place to copy them from. This includes every single M script all in one place (as long as they are all under one section). Here is the actual Power Query's Query Editor list:



all of these queries, parameters, and functions are in the section1.m file. You can easily understand if they are a query, a parameter or a function. Look at the screenshot below;

```

shared #"Public Holidays in Victoria in 2017" = let
    Source = Web.Page(Web.Contents("http://www.officeholidays.com/countries/australia/victoria/" & Year & ".php")),
    Data0 = Source[Data],
    #"Added Custom" = Table.AddColumn(Data0, "Year", each Year),
    #"Merged Columns" = Table.CombineColumns(#"Added Custom",{"Year", "Date"},Combiner.CombineTextByDelimiter(" ", QuoteStyle.None),"Full Date"),
    #"Changed Type" = Table.TransformColumnTypes(#"Merged Columns",{{"Full Date", type date}}),
    #"Removed Columns" = Table.RemoveColumns(#"Changed Type",{"Day"})
in
    #"Removed Columns";

[ Description = "Input Year column for Public Holidays" ]
shared Year = "2017" meta [IsParameterQuery=true, Type="Text", IsParameterQueryRequired=true];

[ FunctionQueryBinding = "{"exemplarFormulaName":"","Public Holidays in Victoria in 2017"}" ]
GetHolidays = let
    Source = (Year as text) => let
        Source = Web.Page(Web.Contents("http://www.officeholidays.com/countries/australia/victoria/" & Year & ".php")),
        Data0 = Source[Data],
        #"Added Custom" = Table.AddColumn(Data0, "Year", each Year),
        #"Merged Columns" = Table.CombineColumns(#"Added Custom",{"Year", "Date"},Combiner.CombineTextByDelimiter(" ", QuoteStyle.None),"Full Date"),
        #"Changed Type" = Table.TransformColumnTypes(#"Merged Columns",{{"Full Date", type date}}),
        #"Removed Columns" = Table.RemoveColumns(#"Changed Type",{"Day"})
    in
        #"Removed Columns"
in
    Source;

```

Queries are just starting with let expressions. However, for parameters there is a meta information:

```

1 meta [IsParameterQuery=true, Type="Text",
    IsParameterQueryRequired=true]

```

for functions; there is a FunctionQueryBinding section which explain where the function is sourced from;

```

1 [ FunctionQueryBinding = "{"exemplarFormulaName":"","Public Holidays in Victoria in 2017"}" ]

```

Green section above is a simple query

The red section is a parameter, and the Yellow section is a function

So, very easily you can access all queries pane from this area. However, this doesn't tell you the folder structure, or some other query editor settings. For example, many of these queries are not loading in Power BI and disabled for the load. This \*.m file doesn't tell you that. That information, however, exists somewhere else. Let's look at that.

## Metadata Information in XML Format

If you open the DataMashup file in a text editor (like Notepad++), you will see another story. Content starts with some binary characters, but then some XML codes appear.



```

<?xml version="1.0" encoding="utf-8"?><LocalPackageMetadataFile xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Items>
    <Item>
      <ItemLocation>...</ItemLocation>
    </Item>
  </Items>
</LocalPackageMetadataFile>

```

Starting binary data are related to the zipped nature of this file (which we've already gone through in the previous part of this post). After binary characters, some XML code starts, which includes some schema definition. You will easily find the metadata information between these tags;

```

<LocalPackageMetadataFile xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Items>
    <Item>
      <ItemLocation>...</ItemLocation>
    </Item>
  </Items>
</LocalPackageMetadataFile>

```

all useful information are under LocalPackageMetadataFile tags. You can remove everything else. After removing everything and keeping only the content between these tags, then you will have XML structure that includes all metadata information about Power Query's Query Editor;

```

<LocalPackageMetadataFile xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Items>
    <Item>
      <ItemLocation>...</ItemLocation>
    </Item>
  </Items>
</LocalPackageMetadataFile>

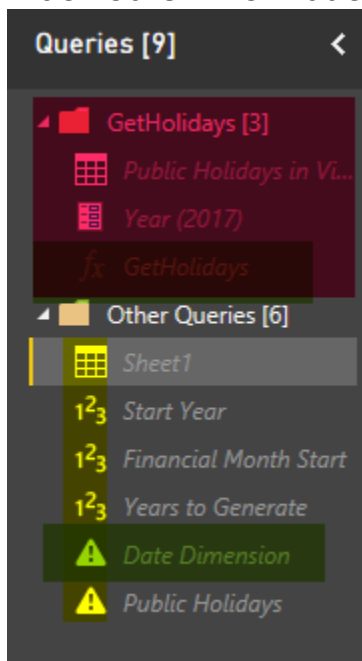
```

As you can see in above screenshot, there are XML tags for all attributes; the name of every step and query (highlighted yellow), and their properties (highlighted green) such as LoadToReportDisabled, ResultType and Value.

If you be a Power Query geek like myself, you will dig into this structure with Power Query Itself! I have opened another Power BI Desktop file, and got data from this file; DataMashup File. I have done some steps, such as removing everything before and after main XML tags, then drilled into different sections of it, and I've got some results such as this:

ItemLocation.ItemPath	IsPriv...	NameUpdatedAfterFil	IsDirectQuery	ResultType	LoadToReportDisabled	QueryGroupID	FillToDataModelEnabled	FillEnabled	FillObjectType
1 Section1/Date Dimension	IO	11	IO	sTable	11	null	null	null	null
2 Section1/Financial Month Start	IO	11	IO	sNumber	11	null	null	null	null
3 Section1/GetHolidays	null	null	null	sFunction	11	420ca76-9e57-4271-98c9-f2fe3d58	null	null	null
4 Section1/Public Holidays	IO	11	IO	sTable	11	420ca76-9e57-4271-98c9-f2fe3d58	null	null	null
5 Section1/Public Holidays in Victoria in 2017	IO	11	IO	sTable	11	420ca76-9e57-4271-98c9-f2fe3d58	null	null	null
6 Section1/Sheet1	IO	11	IO	sTable	11	null	null	null	null
7 Section1/Start Year	IO	11	IO	sNumber	11	null	null	null	null
8 Section1/Year	IO	11	IO	sText	11	420ca76-9e57-4271-98c9-f2fe3d58	ID	ID	sConnectionOnly
9 Section1/Years to Generate	IO	11	IO	sNumber	11	null	null	null	null

As you can see it is fairly obvious that is the query Parameter, Function or a normal Query(highlighted yellow). Is the query inside a group or not (highlighted red), or is it enabled load in the report (highlighted green), and much other information.



Here you go. You have all you need regarding understanding the metadata of tables, queries, parameters, and functions.

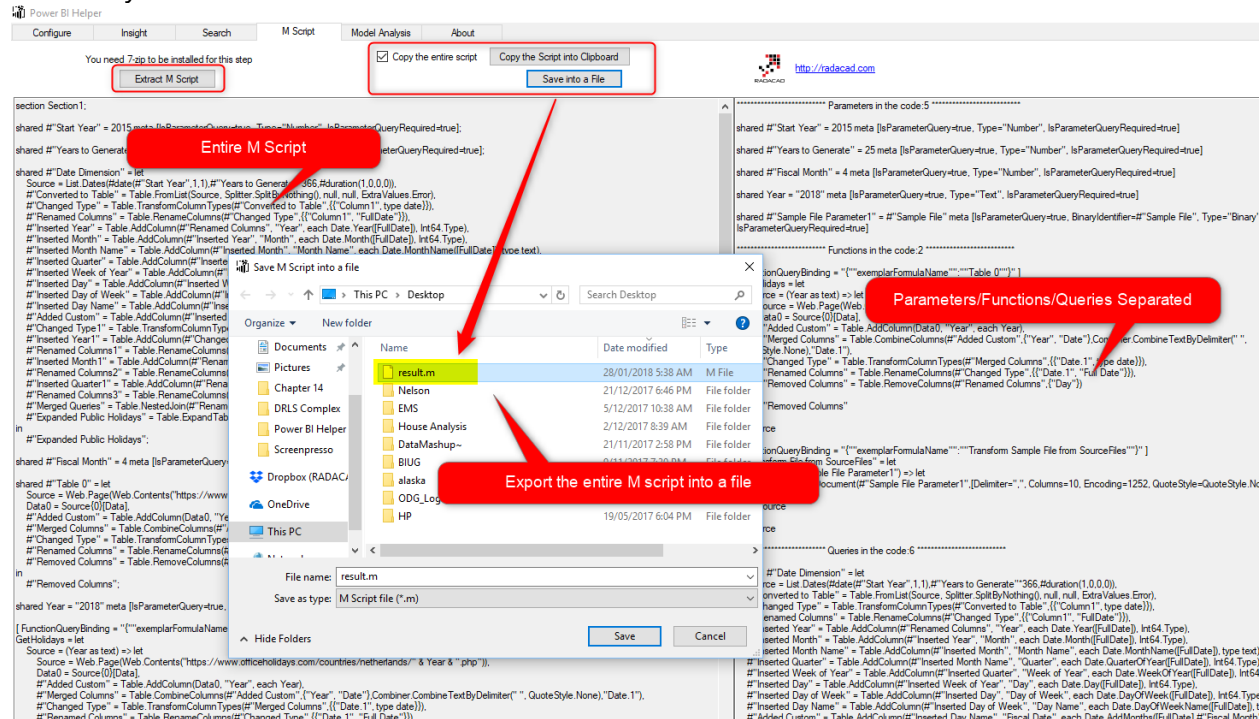
## Summary

DataMashup file in a \*.pbix file included all you need to access to Query Editor's metadata information, and also the M scripts. You can use this file to access to the whole code behind of Power Query component of your Power BI solution. In future posts, I'll go through steps of exploring data from the XML content in DataMashup file and also explain other files under \*.pbix file. There are still a lot of information in XML content of DataMashup file; I encourage you all to explore it with Power Query and open your eyes to the world of metadata of Power Query

Tell me what part of this DataMashup file and metadata information you like most in the comments.

# Export the Entire M Power Query Script from a Power BI File, New Version of Power BI Helper, Search based on Field Description in the Model

Posted by [Reza Rad](#) on Jan 29, 2018



Previous versions of [Power BI Helper](#) had some features related to the visualization part and the modeling part of a Power BI file. This latest release of Power BI helper can export the entire M or Power Query script into a file or clipboard. Sometimes, especially when the number of queries in the Power BI is too many, maintaining the code behind of it can be useful. Not only for the documentation, but also this helps to learn more about what is happening in the ETL side of your solution.

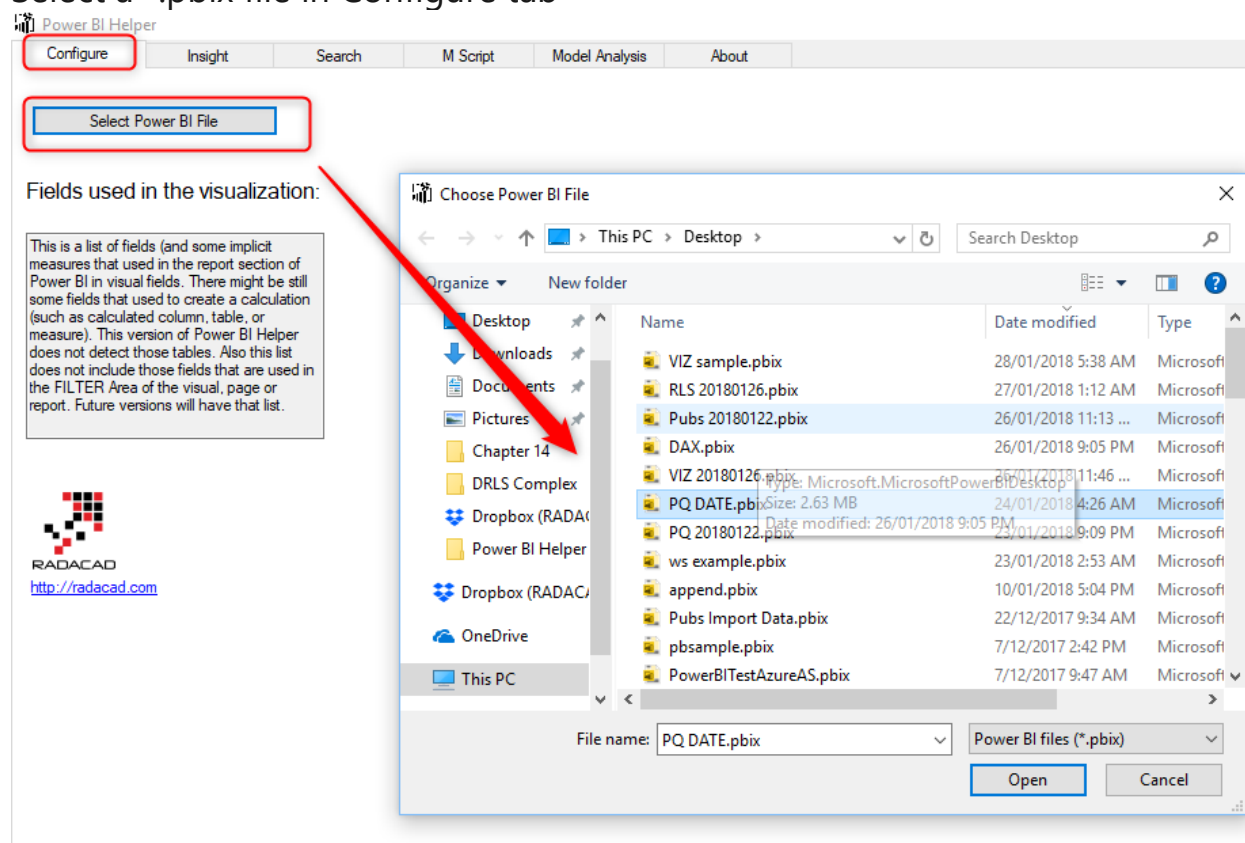


## Export M Script

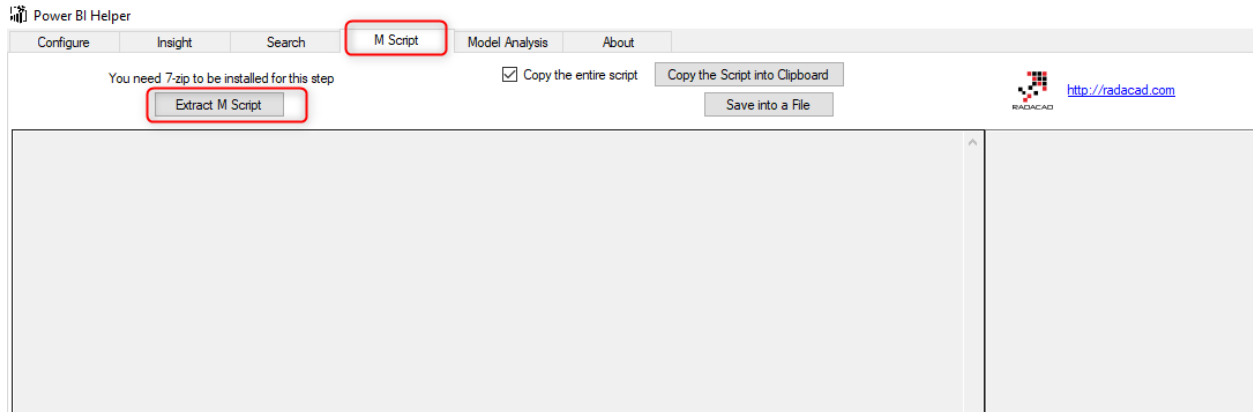
You have to download the latest version of Power BI Helper (Version 1.0). If you have other versions (0.1, 0.2, or 0.3) installed previously, un-install those before this action. After installing the new version, you will see a tab for M Script.

## Download Power BI Helper (FREE)

Select a \*.pbix file in Configure tab



Then Go to M Script tab, and click on extract M Script. This step requires the installation of 7-zip free tool. You can download 7-zip for free from here: <http://7-zip.org>



This simply extracts the entire M script into the text box here. The M script includes all queries, functions, parameters, and any other Power Query objects.

## Power BI Helper

Configure	Insight	Search	M Script	Model Analysis	About
-----------	---------	--------	----------	----------------	-------

You need 7-zip to be installed for this step

☒ Copy the entire script

Copy the Script into Clipboard

Extract M Script

Save into a File

```

section Section1;

shared #"Start Year" = 2015 meta [IsParameterQuery=true, Type="Number", IsParameterQueryRequired=true];

shared #"Years to Generate" = 25 meta [IsParameterQuery=true, Type="Number", IsParameterQueryRequired=true];

shared #"Date Dimension" = let
    Source = List.Dates(#date(#"Start Year",1,1),#"Years to Generate",366,duration(1,0,0)),
    #"Converted to Table" = Table.FromList(Source, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    #"Changed Type" = Table.TransformColumnTypes(#"Converted to Table",{{"Column1", type date}}),
    #"Renamed Columns" = Table.RenameColumns(#"Changed Type",{{"Column1", "FullDate"}}),
    #"Inserted Year" = Table.AddColumn(#"Renamed Columns", "Year", each Date.Year([FullDate]), Int64.Type),
    #"Inserted Month" = Table.AddColumn(#"Inserted Year", "Month", each Date.Month([FullDate]), Int64.Type),
    #"Inserted Month Name" = Table.AddColumn(#"Inserted Month", "Month Name", each Date.MonthName([FullDate]), type text),
    #"Inserted Quarter" = Table.AddColumn(#"Inserted Month Name", "Quarter", each Date.QuarterOfYear([FullDate]), Int64.Type),
    #"Inserted Week of Year" = Table.AddColumn(#"Inserted Quarter", "Week of Year", each Date.WeekOfYear([FullDate]), Int64.Type),
    #"Inserted Day" = Table.AddColumn(#"Inserted Week of Year", "Day", each Date.Day([FullDate]), Int64.Type),
    #"Inserted Day of Week" = Table.AddColumn(#"Inserted Day", "Day of Week", each Date.DayOfWeek([FullDate]), Int64.Type),
    #"Inserted Day Name" = Table.AddColumn(#"Inserted Day of Week", "Day Name", each Date.DayOfWeekName([FullDate]), type text),
    #"Added Custom" = Table.AddColumn(#"Inserted Day Name", "Fiscal Date", each Date.AddMonths([FullDate],#"Fiscal Month"-1)),
    #"Changed Type1" = Table.TransformColumnTypes(#"Added Custom",{{"Fiscal Date", type date}}),
    #"Inserted Year1" = Table.AddColumn(#"Changed Type1", "Year.1", each Date.Year([Fiscal Date]), Int64.Type),
    #"Renamed Columns1" = Table.RenameColumns(#"Inserted Year1",{{"Year.1", "Fiscal Year"}}),
    #"Inserted Month1" = Table.AddColumn(#"Renamed Columns1", "Month.1", each Date.Month([Fiscal Date]), Int64.Type),
    #"Renamed Columns2" = Table.RenameColumns(#"Inserted Month1",{{"Month.1", "Fiscal Period"}}),
    #"Inserted Quarter1" = Table.AddColumn(#"Renamed Columns2", "Quarter.1", each Date.QuarterOfYear([Fiscal Date]), Int64.Type),
    #"Renamed Columns3" = Table.RenameColumns(#"Inserted Quarter1",{{"Quarter.1", "Fiscal Quarter"}}),
    #"Merged Queries" = Table.NestedJoin(#"Renamed Columns3",{"FullDate"},#"Public Holidays",{"Full Date"},"Public Holidays",JoinKind.LeftOuter),
    #"Expanded Public Holidays" = Table.ExpandTableColumn(#"Merged Queries", "Public Holidays", {"Holiday"}, {"Holiday"})
in
    #"Expanded Public Holidays";


shared #"Fiscal Month" = 4 meta [IsParameterQuery=true, Type="Number", IsParameterQueryRequired=true];

shared #"Table 0" = let
    Source = Web.Page(Web.Contents("https://www.officeholidays.com/countries/netherlands/" & Year & ".php")),
    Data0 = Source[0][Data],
    #"Added Custom" = Table.AddColumn(Data0, "Year", each Year),
    #"Merged Columns" = Table.CombineColumns(#"Added Custom",{"Year", "Date"},Combiner.CombineTextByDelimiter(" ", QuoteStyle.None),"Date.1"),
    #"Changed Type" = Table.TransformColumnTypes(#"Merged Columns",{{"Date.1", type date}}),
    #"Renamed Columns" = Table.RenameColumns(#"Changed Type",{{"Date.1", "Full Date"}}),
    #"Removed Columns" = Table.RemoveColumns(#"Renamed Columns",{"Day"})
in
    #"Removed Columns";

shared Year = "2018" meta [IsParameterQuery=true, Type="Text", IsParameterQueryRequired=true];

[ FunctionQueryBinding = "{"exemplarFormulaName":"","Table 0"}" ]
GetHolidays = let
    Source = (Year as text) => let
        Source = Web.Page(Web.Contents("https://www.officeholidays.com/countries/netherlands/" & Year & ".php")),
        Data0 = Source[0][Data],
        #"Added Custom" = Table.AddColumn(Data0, "Year", each Year),
        #"Merged Columns" = Table.CombineColumns(#"Added Custom",{"Year", "Date"},Combiner.CombineTextByDelimiter(" ", QuoteStyle.None),"Date.1"),
        #"Changed Type" = Table.TransformColumnTypes(#"Merged Columns",{{"Date.1", type date}}),
        #"Renamed Columns" = Table.RenameColumns(#"Changed Type",{{"Date.1", "Full Date"}})
    
```

This process also does a minor analysis on how many parameters, queries, and functions you have in your code, and put that into another text box right beside it. All parameters will be listed in one section, then functions, and then queries.

 <http://radacad.com>

```

***** Parameters in the code:5 *****

shared #"Start Year" = 2015 meta [IsParameterQuery=true, Type="Number", IsParameterQueryRequired=true]

shared #"Years to Generate" = 25 meta [IsParameterQuery=true, Type="Number", IsParameterQueryRequired=true]

shared #"Fiscal Month" = 4 meta [IsParameterQuery=true, Type="Number", IsParameterQueryRequired=true]

shared Year = "2018" meta [IsParameterQuery=true, Type="Text", IsParameterQueryRequired=true]

shared #"Sample File Parameter1" = #"Sample File" meta [IsParameterQuery=true, BinaryIdentifier="#Sample File", Type="Binary", IsParameterQueryRequired=true]

***** Functions in the code:2 *****

[ FunctionQueryBinding = {"exemplarFormulaName":"","Table 0"} ]
GetHolidays = let
    Source = (Year as text) => let
        Source = Web.Page(Web.Contents("https://www.officeholidays.com/countries/netherlands/" & Year & ".php")),
        Data0 = Source[0][Data],
        #"Added Custom" = Table.AddColumn(Data0, "Year", each Year),
        #"Merged Columns" = Table.CombineColumns(#"Added Custom",{"Year", "Date"},Combiner.CombineTextByDelimiter(" ",
QuoteStyle.None),"Date.1"),
        #"Changed Type" = Table.TransformColumnTypes(#"Merged Columns",{{"Date.1", type date}}),
        #"Renamed Columns" = Table.RenameColumns(#"Changed Type",{{"Date.1", "Full Date"}}),
        #"Removed Columns" = Table.RemoveColumns(#"Renamed Columns",{"Day"})
    in
        #"Removed Columns"
in
    Source

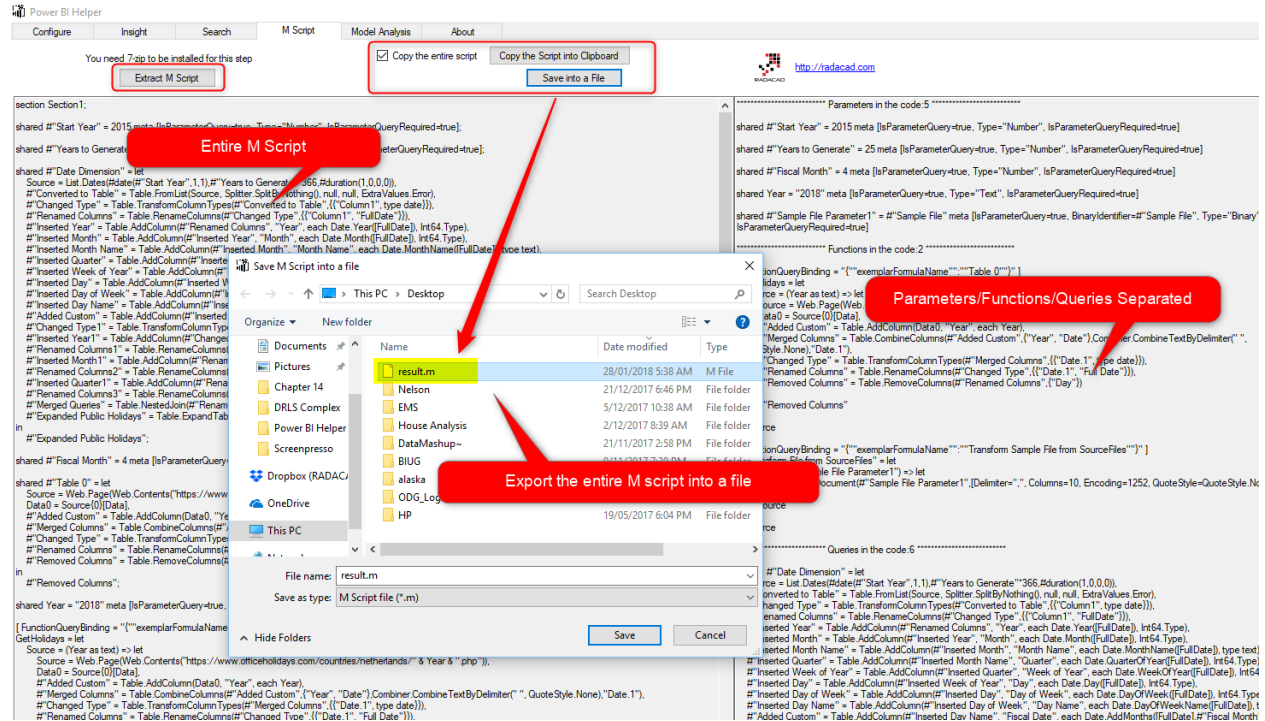
[ FunctionQueryBinding = {"exemplarFormulaName":"","Transform Sample File from SourceFiles"} ]
#"Transform File from SourceFiles" = let
    Source = (#"Sample File Parameter1") => let
        Source = Csv.Document(#"Sample File Parameter1",[Delimiter=";", Columns=10, Encoding=1252, QuoteStyle=QuoteStyle.None])
    in
        Source
in
    Source

***** Queries in the code:6 *****

shared #"Date Dimension" = let
    Source = List.Dates(#date(#"Start Year",1,1),#"Years to Generate"*366,#duration(1,0,0,0)),
    #"Converted to Table" = Table.FromList(Source, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    #"Changed Type" = Table.TransformColumnTypes(#"Converted to Table",{{"Column 1", type date}}),
    #"Renamed Columns" = Table.RenameColumns(#"Changed Type",{{"Column 1", "Full Date"}}),
    #"Inserted Year" = Table.AddColumn(#"Renamed Columns", "Year", each Date.Year(Full Date), Int64.Type),
    #"Inserted Month" = Table.AddColumn(#"Inserted Year", "Month", each Date.Month(Full Date), Int64.Type),
    #"Inserted Month Name" = Table.AddColumn(#"Inserted Month", "Month Name", each Date.MonthName(Full Date), type text),
    #"Inserted Quarter" = Table.AddColumn(#"Inserted Month Name", "Quarter", each Date.QuarterOfYear(Full Date), Int64.Type),
    #"Inserted Week of Year" = Table.AddColumn(#"Inserted Quarter", "Week of Year", each Date.WeekOfYear(Full Date), Int64.Type),
    #"Inserted Day" = Table.AddColumn(#"Inserted Week of Year", "Day", each Date.Day(Full Date), Int64.Type),
    #"Inserted Day of Week" = Table.AddColumn(#"Inserted Day", "Day of Week", each Date.DayOfWeek(Full Date), Int64.Type)

```

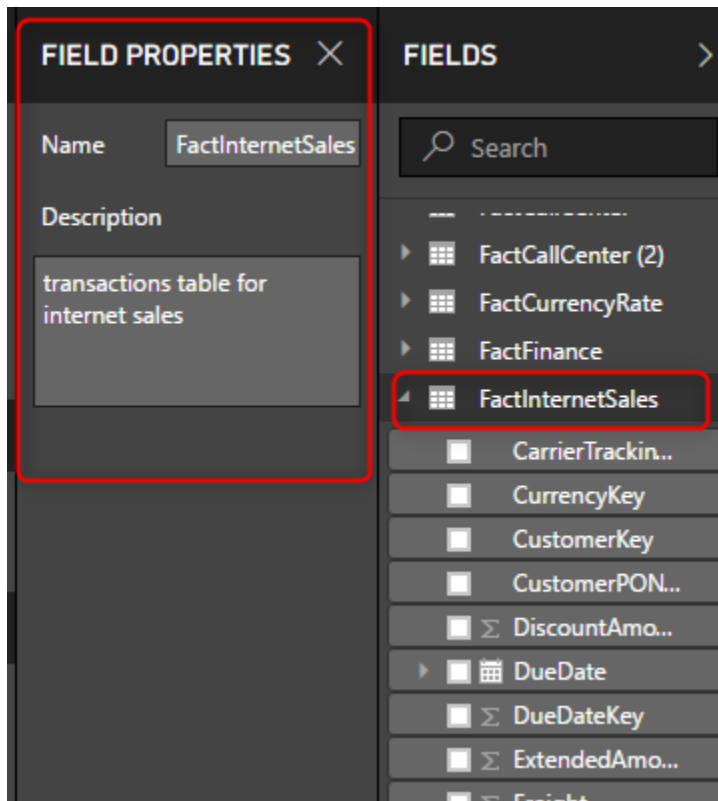
You can now export the entire M Script either into a file (with the extension of \*.m) or copy that into a clipboard.



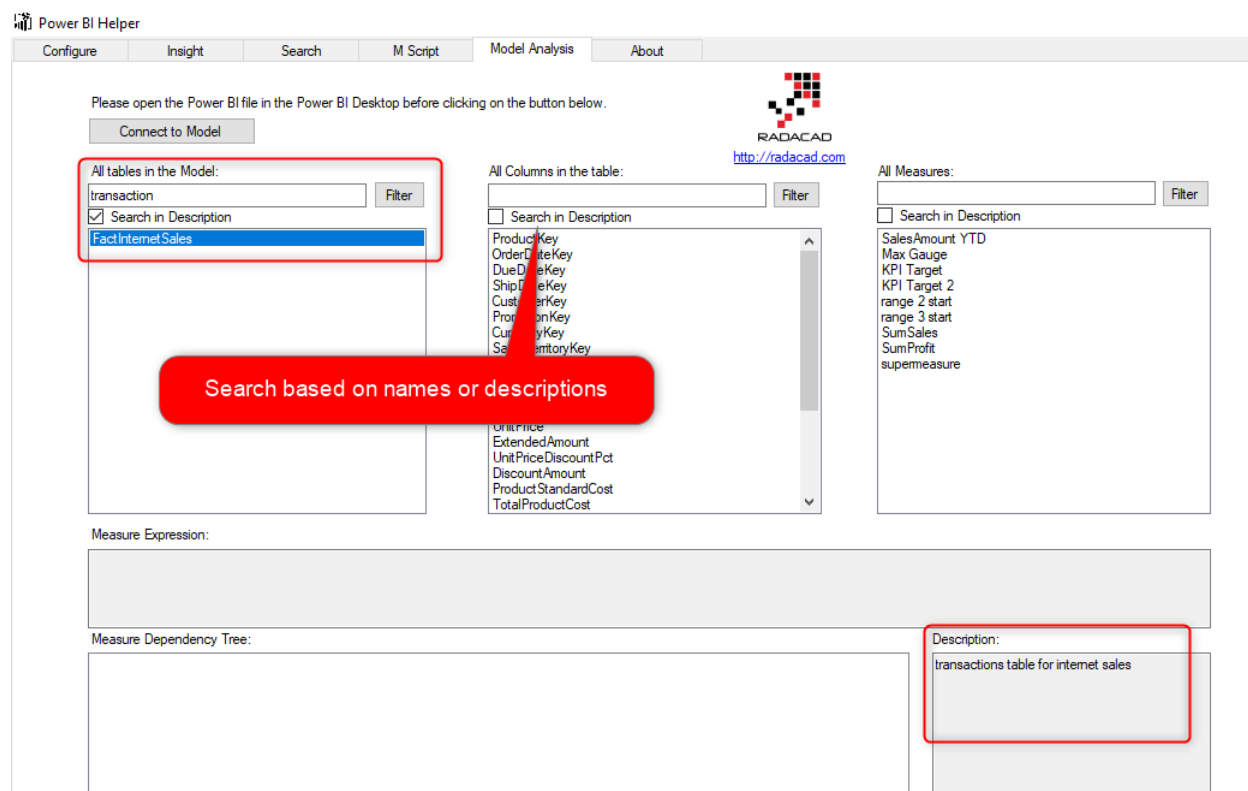
This feature at the moment doesn't apply much of analysis in the code but wait for next few versions; we are going to add a lot of features related to M analysis here later on.

## Search based on Field Description or Properties in the Model

One of the new features added in the Power BI Desktop in December 2017 is to add a description (or properties to fields in the model);



In RADACAD, we'd thought it would be useful to be able to search fields in the model based on these properties. And now in the new version of Power BI Desktop, you can search based on either field/table/measure name or properties of that.



## Power BI Helper on a Virtual Machine

Previous versions of Power BI Helper didn't work on a virtual machine. You could install it, and use the visualization analysis features of it, but the Model analysis didn't work on a VM. With great help on the feedback and test from Gilbert Quevauvilliers ([B.I.](#)), we managed to fix that issue. Now Power BI Helper works perfectly on the Virtual Machines as well.

## More Features?

Let us know if you are looking for some other features in the comment area. We are continuously adding new features to make this tool a free good companion for Power BI Desktop to help developers all around the world.



# Beautify M Script and Extract Row Level Security with Power BI Helper Version 4.0

Posted by [Reza Rad](#) on Aug 27, 2018

Power BI Helper

Configure Insight Search M Script Beautified M Script Model Analysis Modeling Advice About

M Code:

```
section Section1;
```

```
shared #"Customer table from application" = let
```

```
//test here is oned. sample /* here */
```

```
x=12,
```

```
/*
```

```
this
```

```
another ~!@#%$^&*()_ ^%$*-+=<421531,>.?:;'{}[]\|
```

```
:sample /jikh dfsa %ldsfdj overe \ % $
```

```
*/
```

```
Source = Excel.Workbook(File.Contents("C:\Users\RezaRad\OneDrive\Blog\mismatch\sample data.xlsx"), null, true),
```

```
#"Customer table from application_Sheet" = Source{[Item="Customer table from application",Kind="Sheet"]}[Data],
```

```
#"Promoted Headers" = Table.PromoteHeaders("#Customer table from application_Sheet", [PromoteAllScalars=true]),
```

```
#"Changed Type" = Table.TransformColumnTypes("#Promoted Headers",{{"CustomerKey", Int64.Type}, {"FirstName", type text}, {"MiddleName", type text}, {"LastName", type text}, {"MaritalStatus", type text}, {"Suffix", type any}, {"Gender", type text}})
```

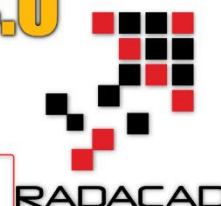
```
in
```

```
#"Changed Type";
```

```
shared #"Customer table from website" = let
```

```
Source = Excel.Workbook(File.Contents("C:\Users\RezaRad\OneDrive\Blog\mismatch\sample data.xlsx"), null, true),
```

**Beautified M Script  
and row-level security  
output in Power BI  
Helper version 4.0  
August 2018**



Power BI Helper is getting new features every time, and this time, we got some exciting features; You can now get your M (Power Query script) code beautified and colorful with version 4.0 of Power BI Helper. We also get the row-level security information exposed through the Helper application. Both information above will be now available when you export the model information to a document. If you like to [learn more about Power BI Helper, read this page.](#)

If you are new to Power BI Helper, read below posts to learn what are existing features of this product:



- [Version 0.1: Tables and Columns used in the visualization](#)
- [Version 0.2: Search a field or table in visualizations and filters](#)
- [Version 0.3: Measure dependency tree](#)
- [Version 1.0: Export entire M script](#)
- [Version 2.0: Connection Types, Multiple PBI files, Relationship advise, and Not used tables](#)
- [Version 3.0: Export Model documentation](#)

## Download

To Download Power BI Helper, click here:

[Download Power BI Helper from here.](#)

### Beautified M Script

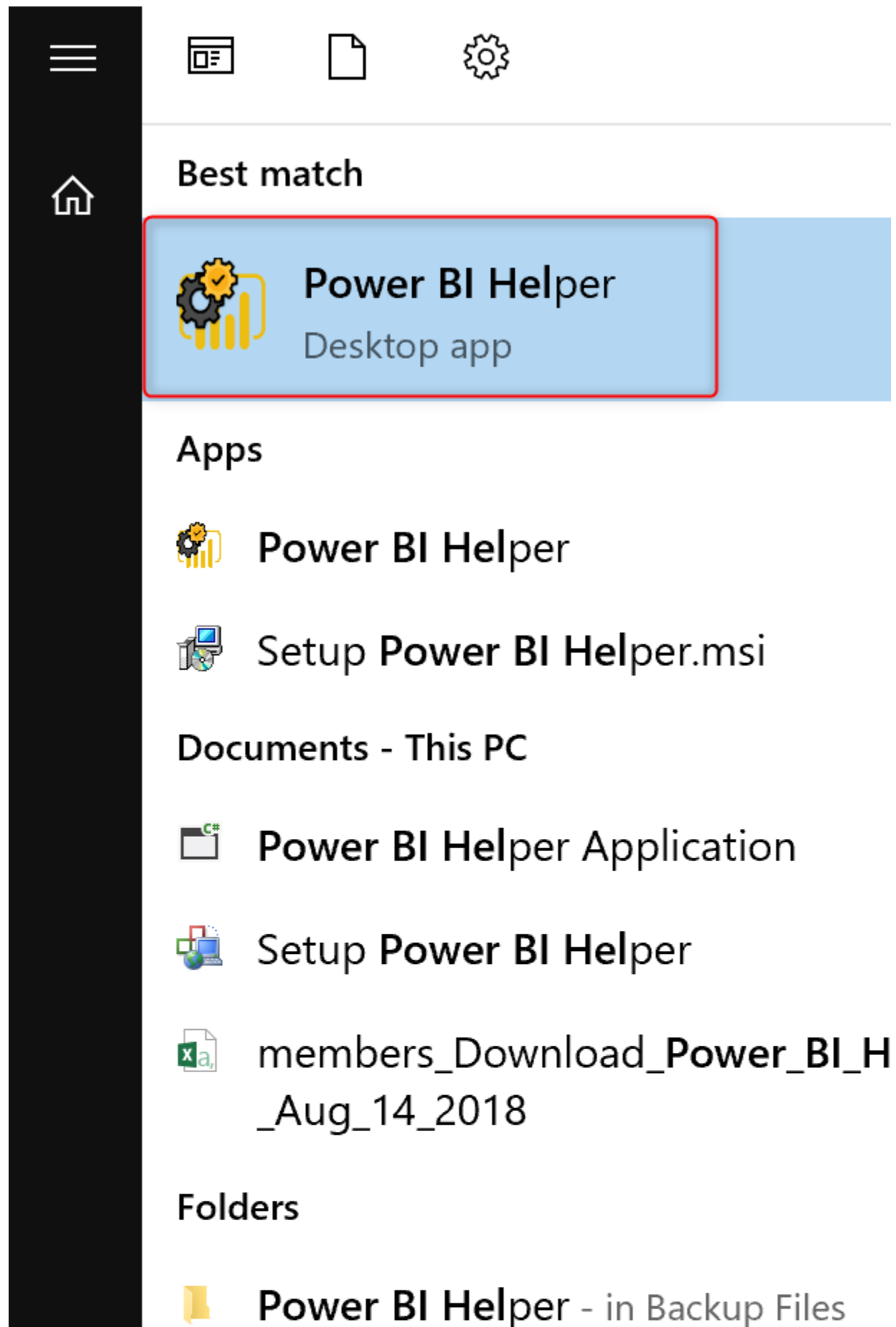
We are very excited to let you know that your M script is not documented all black text, you get a beautiful color coding with that now. Here is a sample output of Power BI Helper for M script;

M Code:

```
section Section1;

shared #"Customer table from application" = let
//test here is oned. sample /* here */
x=12,
/*
this
another ~\|@#%&^*(_)^%$%-+=<421531,>.>./;:'{[]}\
:sample /Jikh dfsa %ldsfjd overe \ % $
*/
Source = Excel.Workbook(File.Contents("C:\Users\RezaRad\OneDrive\Blog\mismatch\sample data.xlsx"), null, true),
#"Customer table from application. Sheet" = Source([Item="Customer table from application",Kind="Sheet"])[Data],
#"Promoted Headers" = Table.PromoteHeaders(#"Customer table from application_Sheet", [PromoteAllScalars=true]),
#"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"CustomerKey", Int64.Type}, {"FirstName", type text}, {"MiddleName", type text}, {"LastName", type text}, {"BirthDate", type date}, {"MaritalStatus", type text}, {"Suffix", type any}, {"Gender", type text}})
in
#"Changed Type";
```

As you can see the color coding above shows comments in greet, reserved words as blue, and function names as purple. You can get this output after three steps in Power BI Helper, Open Power BI Helper (we've got a new icon now )



First Select your \*.Pbix file in the Config tab.

Power BI Helper

Configure Insight Search M Script Model Analysis Modeling Advise About

Select Power BI File

Fields used in the visualization:

This is a list of fields (and some implicit measures that used in the report section of Power BI in visual fields. There might be still some fields that used to create a calculation (such as calculated column, table, or measure). This version of Power BI Helper does not detect those tables. Also this list does not include those fields that are used in the FILTER Area of the visual, page or report. Future versions will have that list.

RADACAD  
<http://radacad.com>

Choose Power BI File

OneDrive > Blog > mismatch

Organize New folder

Quick access Desktop Downloads Documents Pictures 2018-08-25 SQL aggs mismatch Power BI Helper

mismatch

Then in the second step, you can extract the M code in a text box. Note that the text box here shows only the content in text format. The colored output comes in the next step;

Power BI Helper

Configure Insight Search M Script Model Analysis Modeling Advise About

You need 7-zip to be installed for this step

☒ Copy the entire script

section Section 1:

```
shared #"Customer table from application" = let
//test here is oned. sample
/* here */
x=12,
/*
this
another ~!@#%&^&()*_~$*~+=<421531.>?/::{}]]\
sample /jkh dfsa %jdfjd overe \ % $
*/
Source = Excel.Workbook(File.Contents("C:\Users\RezaRad\OneDrive\Blog\mismatch\sample data.xlsx"), null, true),
#"Customer table from application_Sheet" = Source([Item="Customer table from application", Kind="Sheet"])[Data],
#"Promoted Headers" = Table.PromoteHeaders(#"Customer table from application_Sheet", [PromoteAllScalars=true]),
#"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"CustomerKey", Int64.Type}, {"FirstName", type text}, {"MiddleName", type text}, {"LastName", type text}, {"BirthDate", type date}, {"MaritalStatus", type text}, {"Suffix", type any}, {"Gender", type text}})
in
#"Changed Type";

shared #"Customer table from website" = let
Source = Excel.Workbook(File.Contents("C:\Users\RezaRad\OneDrive\Blog\mismatch\sample data.xlsx"), null, true),
#"Customer table from website_Sheet" = Source([Item="Customer table from website", Kind="Sheet"])[Data],
#"Promoted Headers" = Table.PromoteHeaders(#"Customer table from website_Sheet", [PromoteAllScalars=true]),
#"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"CustomerKey", Int64.Type}, {"FirstName", type text}, {"MiddleName", type text}, {"LastName", type text}, {"EmailAddress", type text}})
in
#"Changed Type";
```

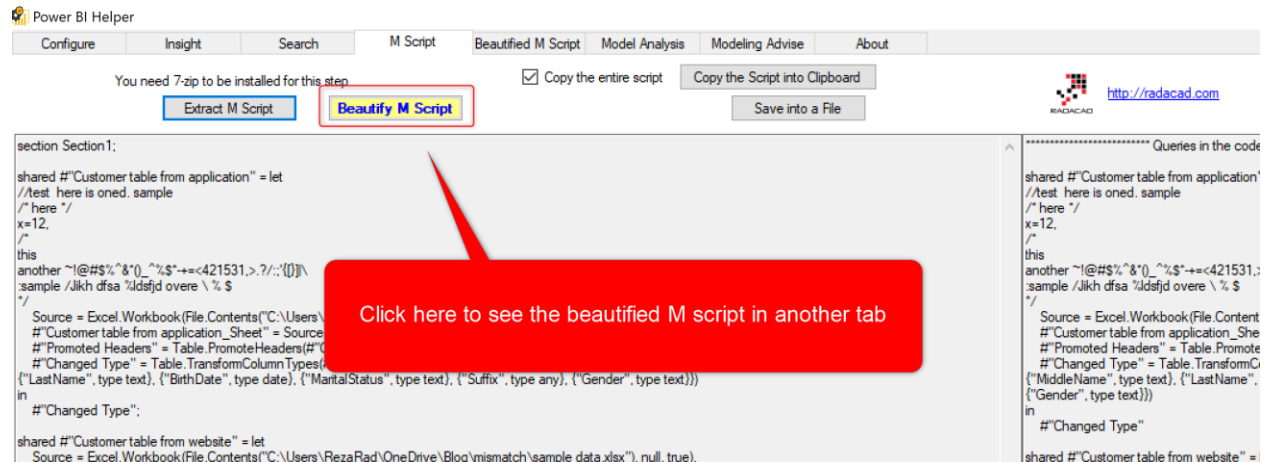
M script in original format

Queries in the code:

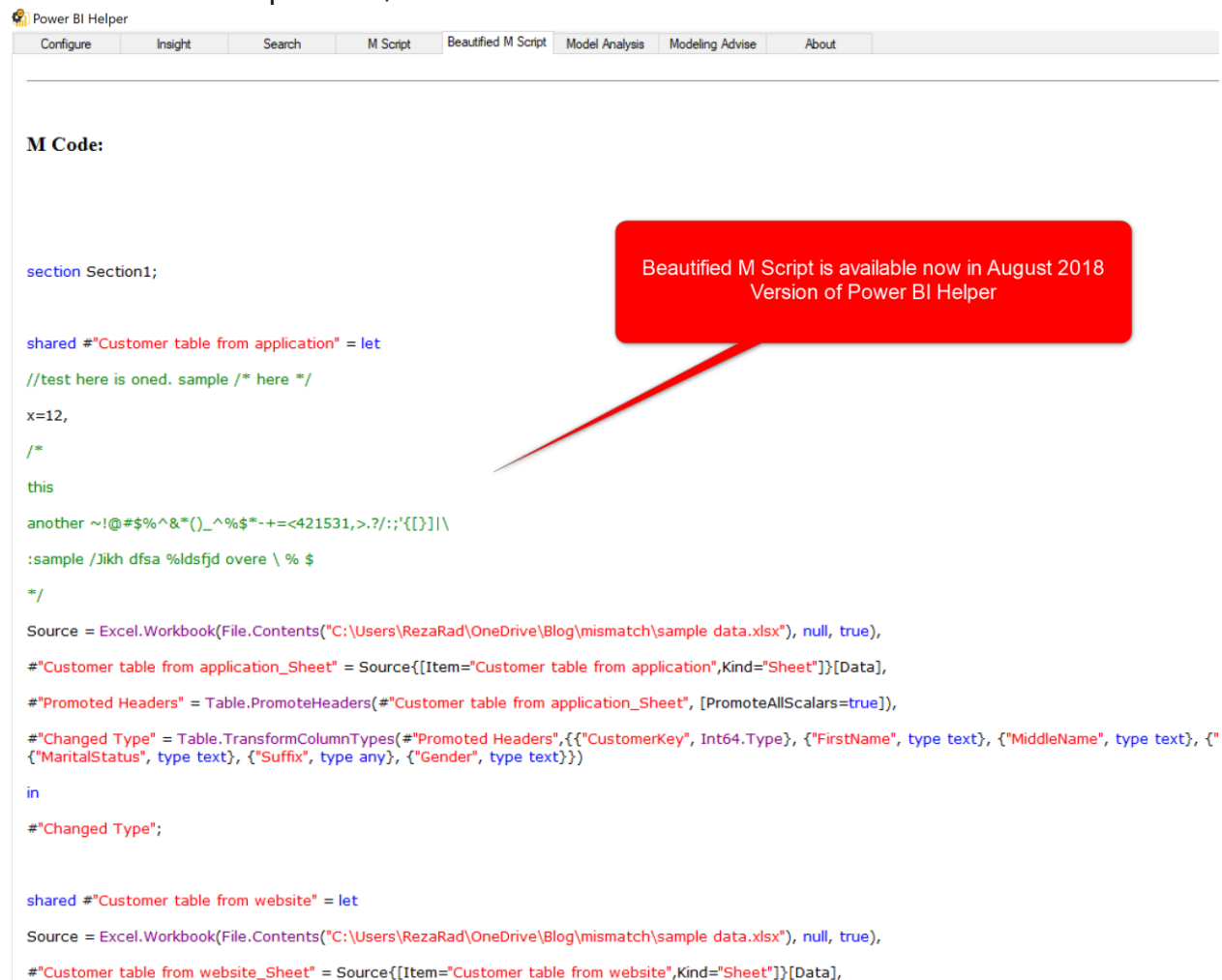
```
shared #"Customer table from application" =
//test here is oned. sample
/* here */
x=12,
/*
this
another ~!@#%&^&()*_~$*~+=<421531.>?/::{}]]\
sample /jkh dfsa %jdfjd overe \ % $
*/
Source = Excel.Workbook(File.Contents(
#"Customer table from application_Sheet" = Table.PromoteHeaders(#"Customer table from application_Sheet", [PromoteAllScalars=true]),
#"Promoted Headers" = Table.PromoteH
#"Changed Type" = Table.TransformCol
{"MiddleName", type text}, {"LastName", ty
{"Gender", type text}})
in
#"Changed Type"

shared #"Customer table from website" = let
Source = Excel.Workbook(File.Contents(
#"Customer table from website_Sheet" = Table.PromoteHeaders(#"Customer table from website_Sheet", [PromoteAllScalars=true]),
#"Promoted Headers" = Table.PromoteH
#"Changed Type" = Table.TransformCol
{"MiddleName", type text}, {"LastName", ty
```

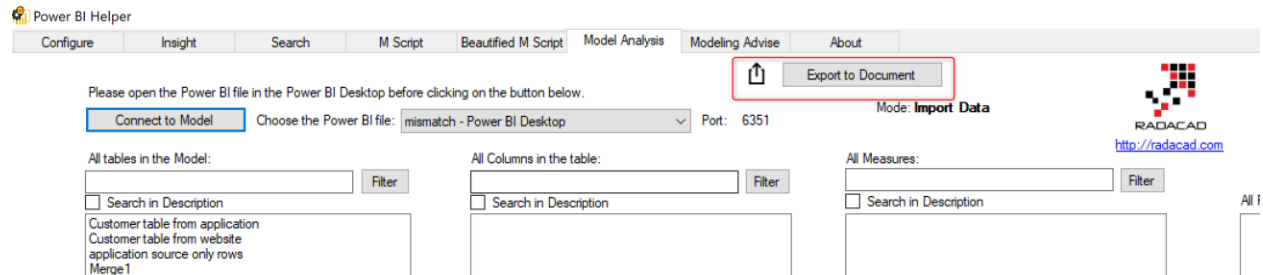
In the last step, you can click on Beautify M Script;



This action will open the Beautified M Script tab with an output of color-coded beautified M script code;



You can also see the output in an exported document;



The HTML export documented output of Power BI Helper will have the beautified M script at the end of it

M Code:

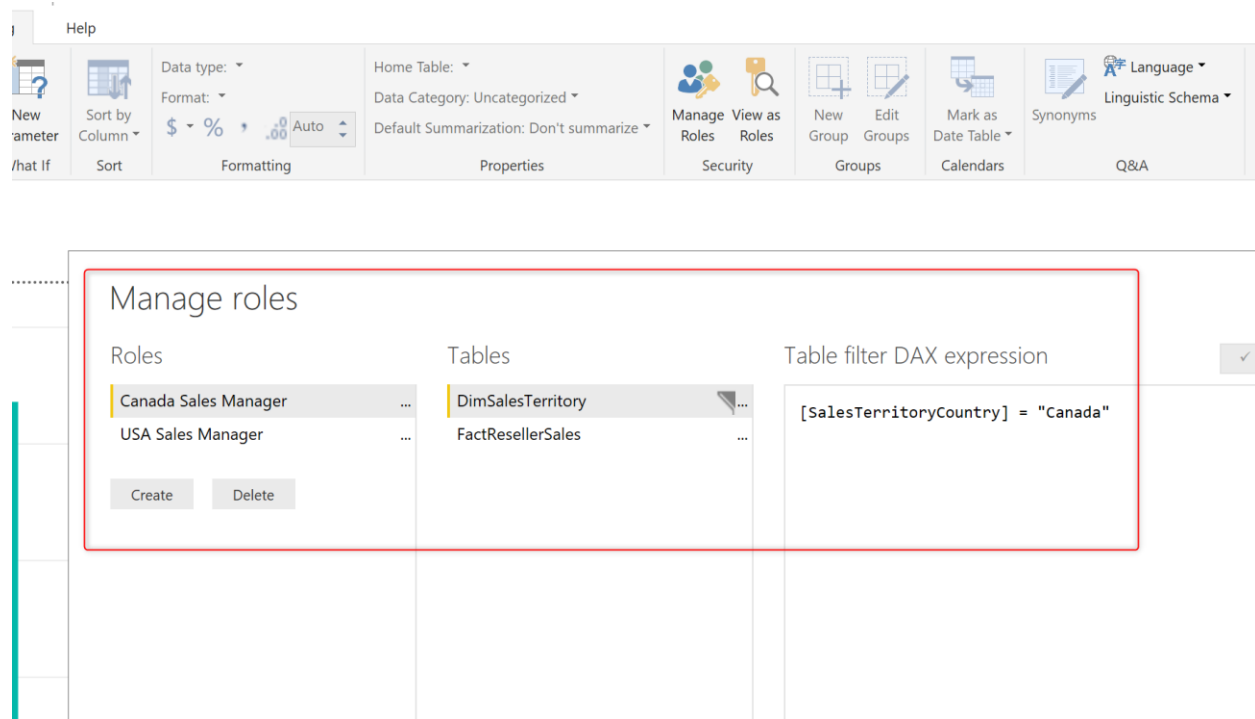
```
section Section1;

shared #"Customer table from application" = let
//test here is oned. sample /* here */
x=12,
/*
this
another ~|@#%&^&*()_!%$*-+=<421531,>./:;'\{\}|\
:sample /Jikh dfsa %kdsfjd overe \ % $
*/
Source = Excel.Workbook(File.Contents("C:\Users\RezaRad\OneDrive\Blog\mismatch\sample data.xlsx"), null, true),
#"Customer table from application_Sheet" = Source([Item="Customer table from application",Kind="Sheet"])[Data],
#"Promoted Headers" = Table.PromoteHeaders(#"Customer table from application_Sheet", [PromoteAllScalars=true]),
#"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"CustomerKey", Int64.Type}, {"FirstName", type text}, {"MiddleName", type text}, {"LastName", type text}, {"BirthDate", type date}, {"MaritalStatus", type text}, {"Suffix", type any}, {"Gender", type text}})
in
#"Changed Type";
```

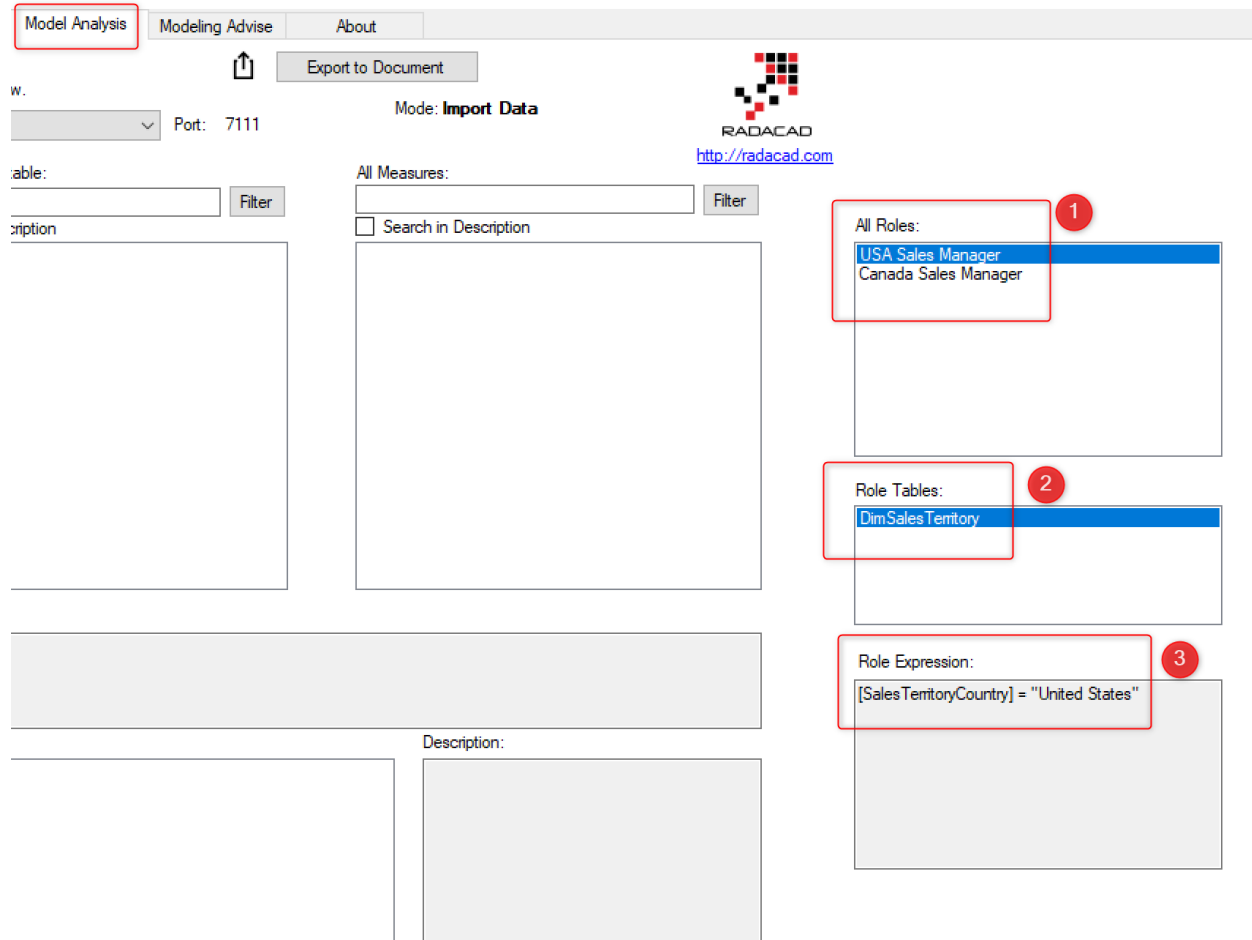
Reading the M script of Power Query, or exporting it in a readable format, been always a challenge. Power BI Helper is making this process simpler and faster for you

## Row-Level Security

Another important information in a Power BI solution is [row-level security implementation](#). The new version of Power BI Helper will reveal this information for you in the Analytics tab. You will have all the information now in there. You can also export the information into a document. Here is an example row-level security implementation in Power BI file;



In the new version of Power BI Helper, the Analytics tab has a section for roles and tables involved in each role, plus the expression for every table;



Model Analysis | Modeling Advise | About

Export to Document

Mode: **Import Data**

Port: 7111

Table:  Filter

Description:

All Measures:  Filter

☐ Search in Description

**1** All Roles:

- USA Sales Manager
- Canada Sales Manager

**2** Role Tables:

- DimSalesTerritory

**3** Role Expression:

[SalesTerritoryCountry] = "United States"

You can also get this information exported into HTML output format as below;

SalesTerritoryRegion			
SalesTerritoryCountry			
SalesTerritoryGroup			
CountryImage			

**List of Roles:**

Role Name	Table Name	Description	Expression
USA Sales Manager	DimSalesTerritory		[SalesTerritoryCountry] = "United States"
Canada Sales Manager	DimSalesTerritory		[SalesTerritoryCountry] = "Canada"

**M Code:**

Row-level security output in the Power BI documentation

`section Section1;`

`shared #"Customer table from application" = let`

## Summary

Power BI Helper version has great feature of M script beautifier and also the row-level security information revealed. Both this information can be exported into Power BI documented output. Power BI helper is evolving every month. This is a free tool and will remain a free tool to help the Power BI community. Please help us with your feedback and suggestions about features you would love to see in next versions



## Other modules of the book

Congratulations on completing the first book of Power BI from Rookie to Rock Star series. You are in the right track, but still more to do. Here are other modules that you can read:

- **Book 1: Power BI Essentials**
- **Book 2: Visualization with Power BI**
- Book 3: Power Query and Data Transformation in Power BI
- **Book 4: Power BI Data Modelling and DAX**
- **Book 5: Pro Power BI Architecture**

# Power BI Training

Reza runs Power BI training courses both online and in-person. RADACAD also runs Advanced Analytics with R, Power BI, Azure Machine Learning and SQL Server courses ran by Dr. Leila Etaati. Our courses run both online and in-person in major cities and countries around the world.

Check the schedule of upcoming courses here:

<http://radacad.com/events>

<http://radacad.com/power-bi-training>

