Power BI from Rookie to Rock Star

Book 4: Power BI Modelling and DAX

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

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 four of the series. In this book, you will learn about data modeling in Power BI with a heavy focus on DAX. You will learn about all types of calculations in Power BI such as Calculated column, table, and measure and their differences. You will learn many scenarios of writing calculations in Power BI. Through many examples, you will learn all about the modeling in Power BI, which is relationships, hierarchies, date dimension, DAX functions and expressions, etc. If you want to learn more about other parts of Power BI not just Modeling, 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with the fast pace of updates for Power BI Desktop, it is impossible to publish a paperback book because it will be outdated in a few months.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.

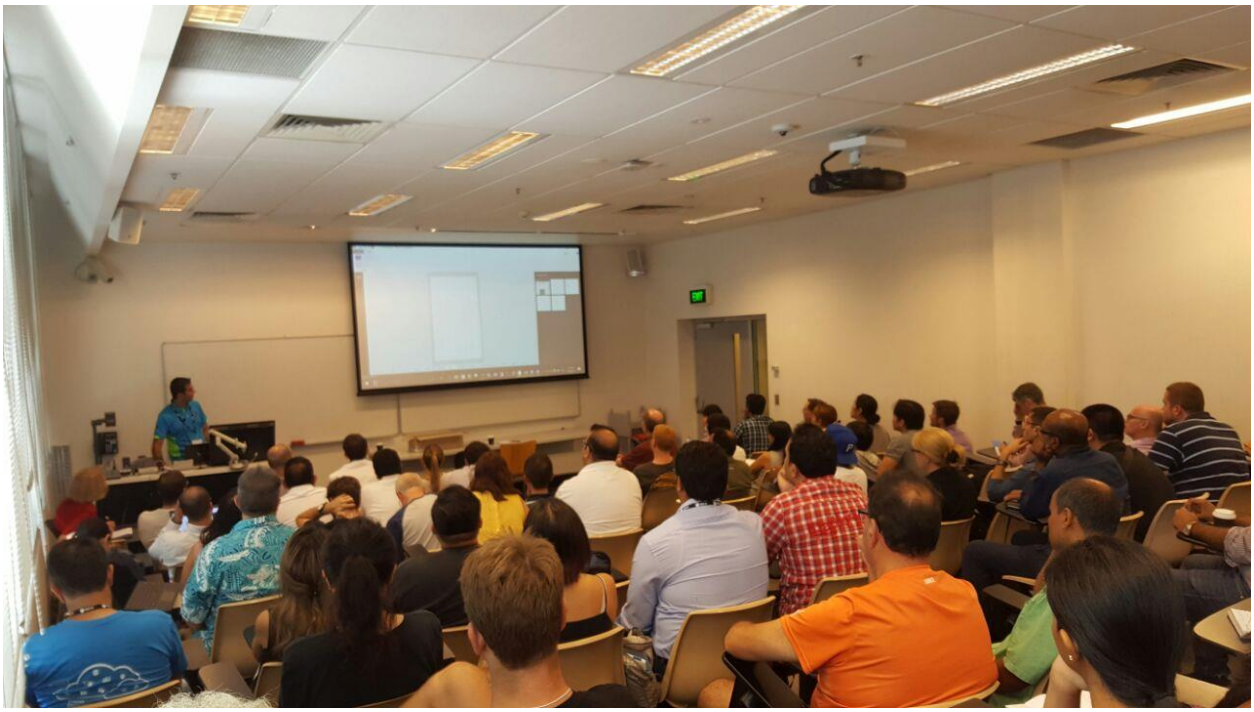
Upcoming Training Courses

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



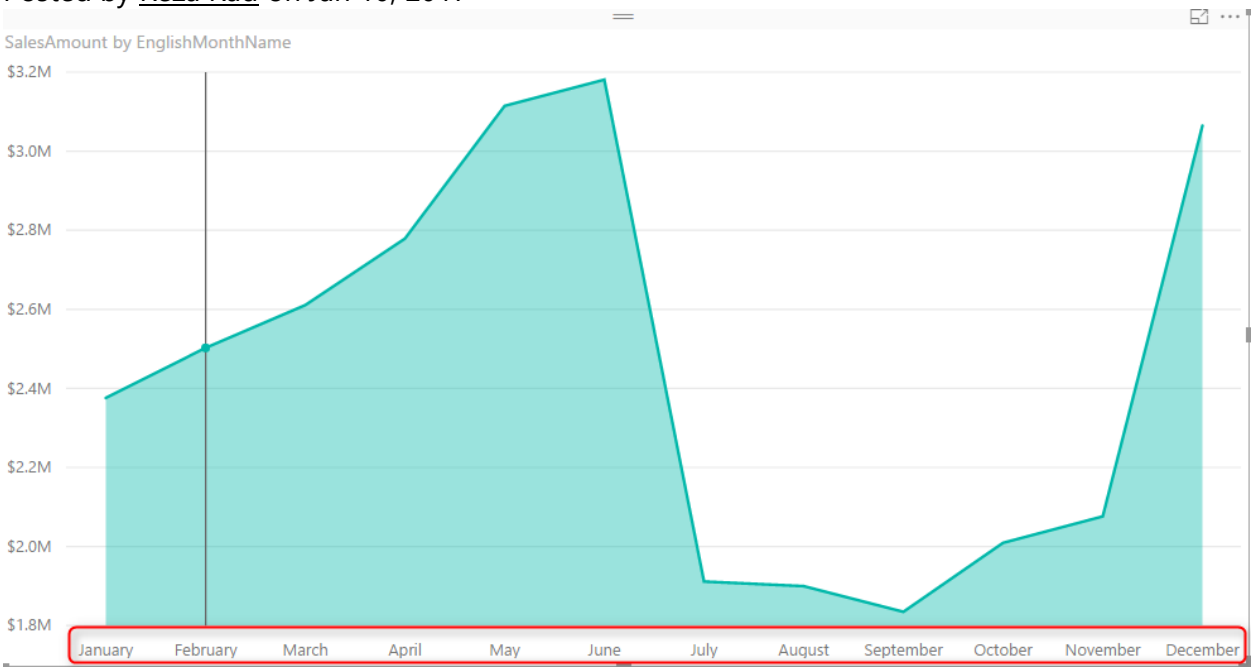
Heading Table of Content

Detailed Table of Content

Part I: Basics of Modeling

Sort By Column in Power BI

Posted by Reza Rad on Jan 10, 2017



Power BI users Sorting in most of the visualizations, you can choose to sort ascending or descending based on specified data fields. However, the field itself can be sorted based on another column. This feature called as Sort By Column in Power BI. This is not a new or advanced feature in Power BI. This is a very basic feature that enhances your visualization significantly. Let's look at this simple feature with an example. If you like to learn more about Power BI; read Power BI online book from Rookie to Rock Star.

Prerequisite

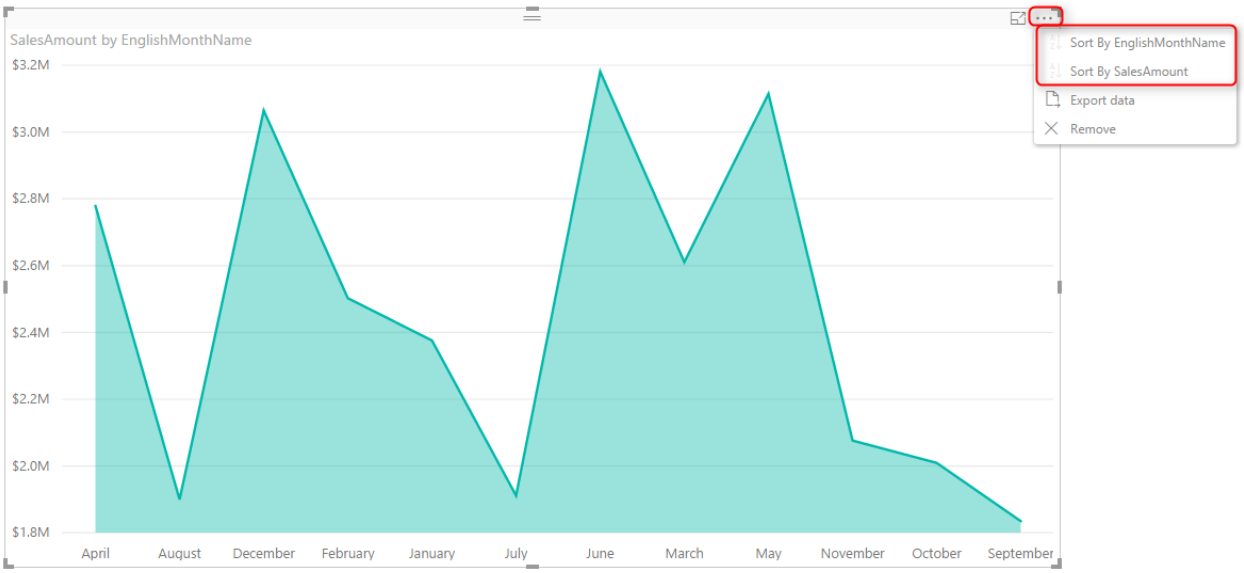
For running example of this post, you will need the AdventureWorksDW sample database, or you can download an **Excel version** of it from here:

Download

Sorting in Power BI

By Default, a field's data is sorted by that field itself. It means if the field is numeric it will be ordered based on the number, if it is text it will be ordered alphabetically. Let's look at it through an example;

Create a New Power BI Desktop file, and Get Data from AdventureWorksDW, from DimDate, and FactInternetSales Tables. Check the relationship between these tables to be only based on an active relationship between OrderDateKey in the FactInternetSales and DateKey in the DimDate. Remove any extra relationships. Create a visualization (Area Chart for example) with SalesAmount as Values, and EnglishMonthName as Axis.



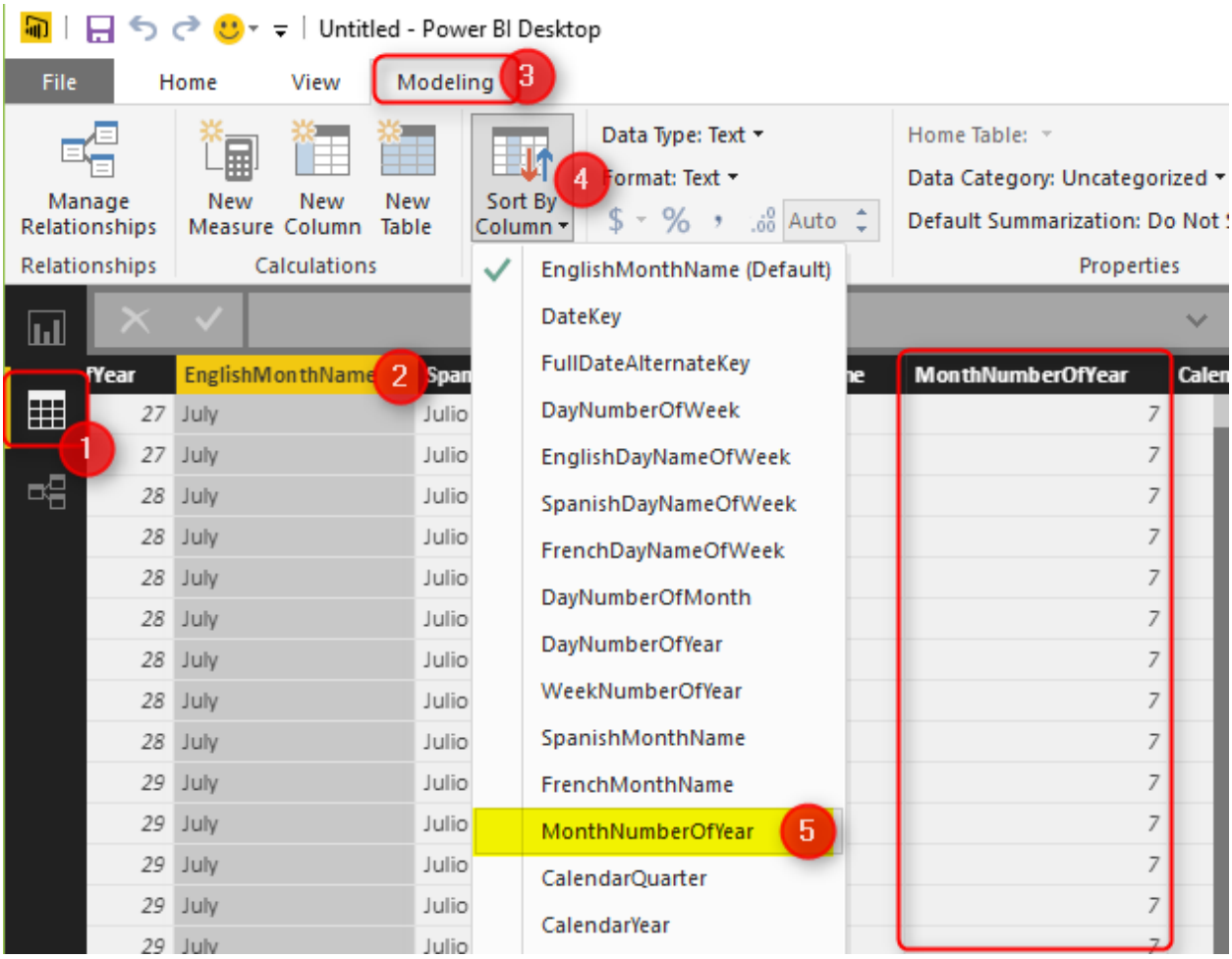
If you click on the three-dot button on the top right-hand side of the chart, you can see the sorting option for this visual. You can choose to sort based on fields that used in this visual, which can be either SalesAmount or EnglishMonthName. If you sort it based on Sales Amount, it is showing you from biggest value to the lowest or reverse. This is the normal behavior; a field will be ordered by values in the field itself.

Problem in Sorting

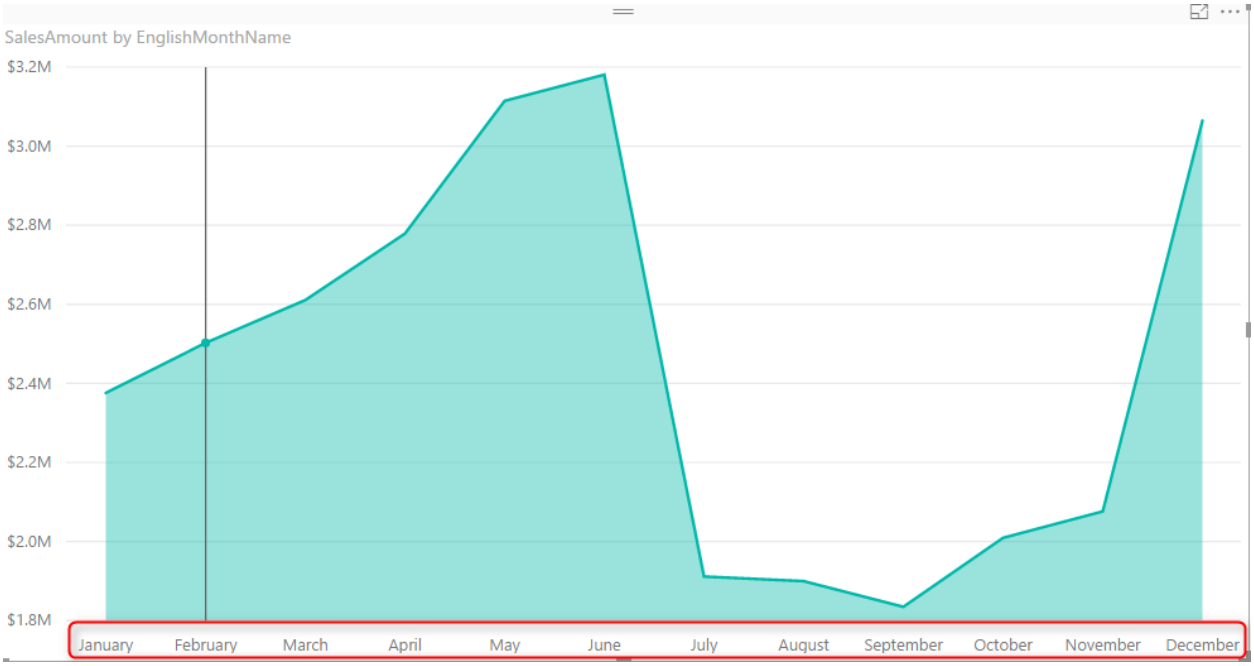
The problem happens when you want a Text field to be ordered based on something different than the value of the field. For example, if you look at the above chart, you can see that month ordered from April to September. This is not the order of months; this is alphabetical order. If you change the sorting of visual, it will only change it from A to Z, or Z to A. To make it in the order of month numbers you have to do it differently.

Sort By Column

If you wish a column to be sorted by values of another column, you can simply do that in the Data tab of Power BI Desktop. First, go to Data Tab, Select the field that you want to be sorted (EnglishMonthName in this example), and then from the menu option under Modeling choose Sort Column By, and select the field that contains numeric values of months (1, 2, 3, …12). This field in our example is MonthNumberOfYear.



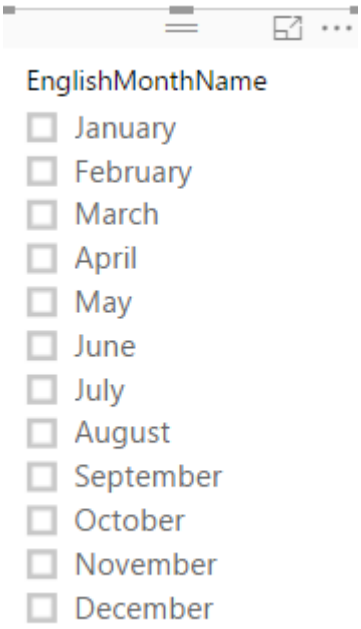
You can see in the screenshot above that MonthNumberOfYear is showing the numeric value of each month. After applying this change, simply go back to the report, and you will see the correct ordering of EnglishMonthName now.



More Cases to use Sort By Column

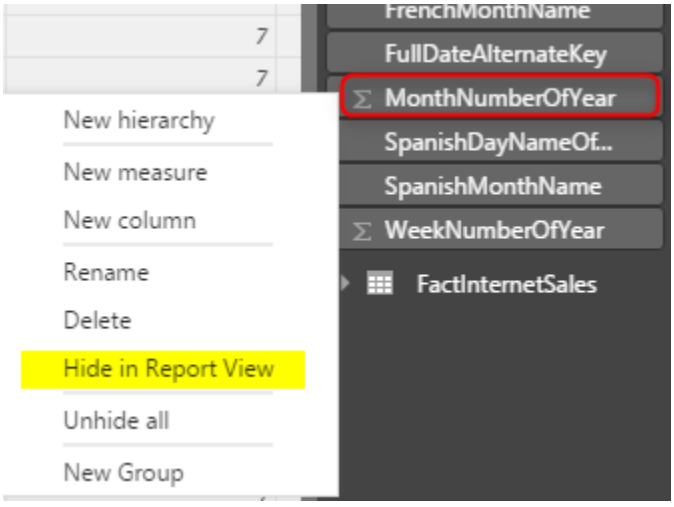
When you use this type of sorting, all visuals will be working based on this ordering. If you have a slicer, items in the slicer will be sorted and ordered based on this. This would be the way of filtering for that column from now onward. You can also think about many other possibilities and usages of this feature, here are a few samples;

- Ordering Month Names based on Financial Period order. In this case, you will need a FiscalMonthNumberOfYear column.
- Ordering items in slicer based on something different than the text. For example Priority or categories or anything like that.



Hide the Column Used for Sorting

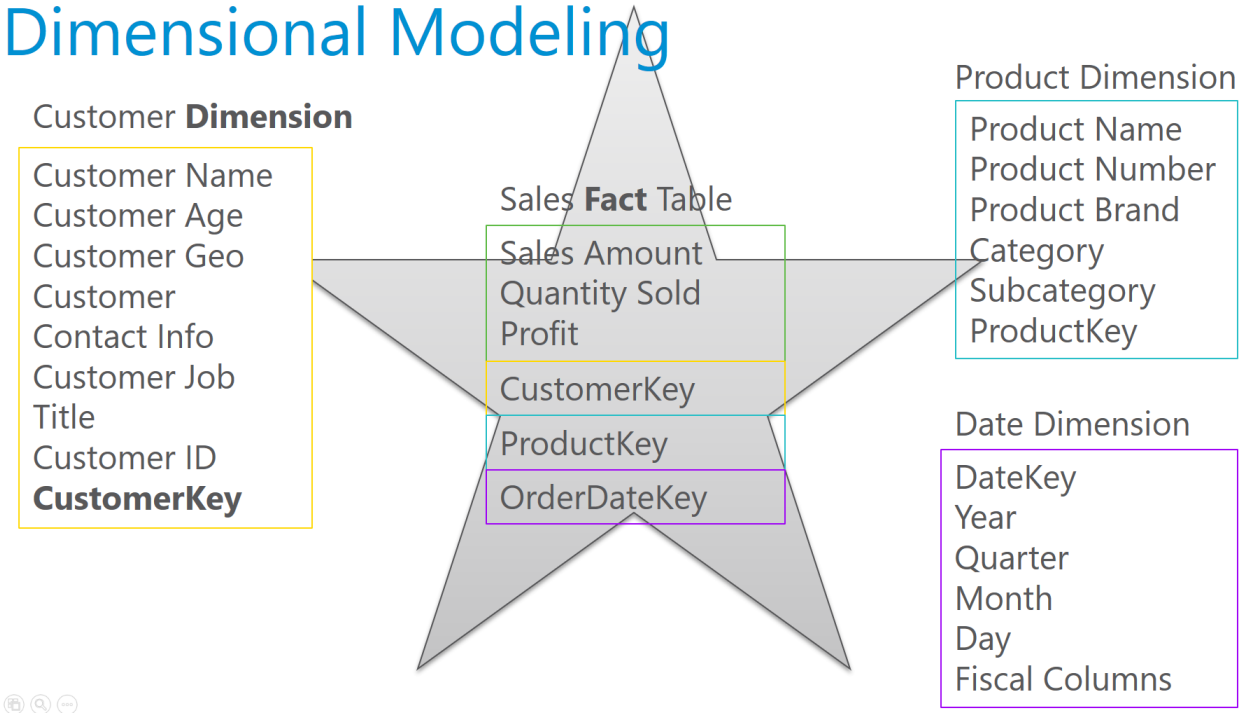
The column used for sorting (in this example MonthNumberOfYear) is not normally used for visualization itself. So you can simply Hide it from Report View.



This is a recommended approach because having too many fields in the report view is confusing for end users.

Data Preparation; First and Foremost Important Task in Power BI

Posted by Reza Rad on Dec 21, 2016

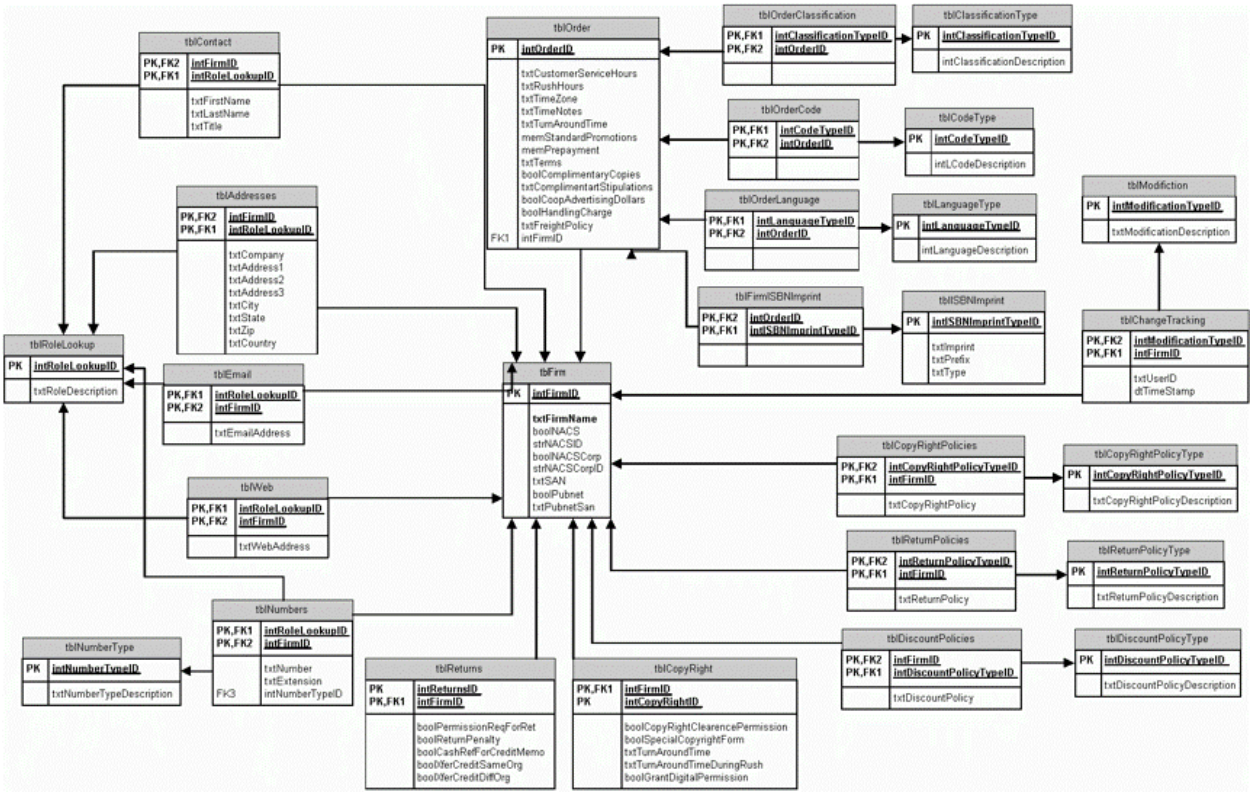


I've talked about Data Preparation many times in conferences such as PASS Summit, BA Conference, and many other conferences. Every time I talk about this I realize how much more I need to explain this. Data Preparation tips are basic but very important. In my opinion as someone who worked with BI systems for more than 15 years, this is the most important task in building in a 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.

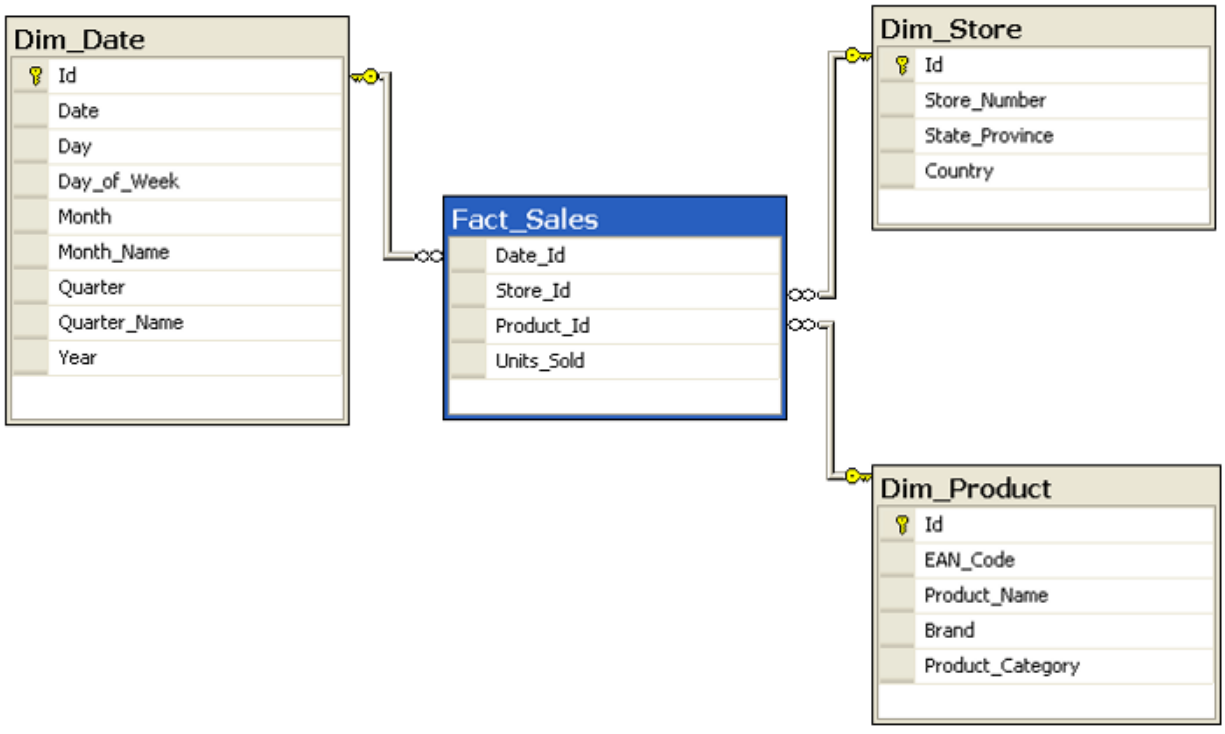
Transactional databases are built for CRUD operations (Create, Retrieve, Update, Delete rows). Because of this single purpose, transactional databases are build in Normalized way, to reduce redundancy and increase the consistency of the data. For example, there should be one table for Product Category 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***

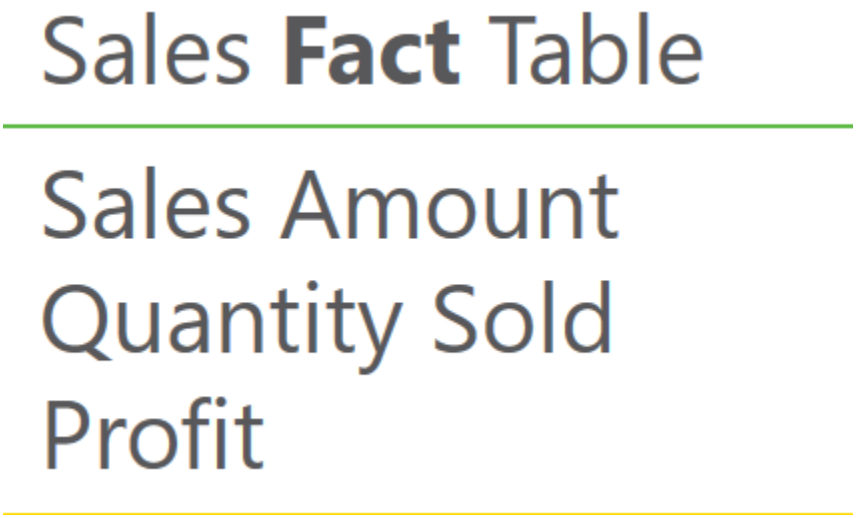
Schema. Building a star schema or dimensional modeling is your most important task as a BI developer.

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;



Dimension Table

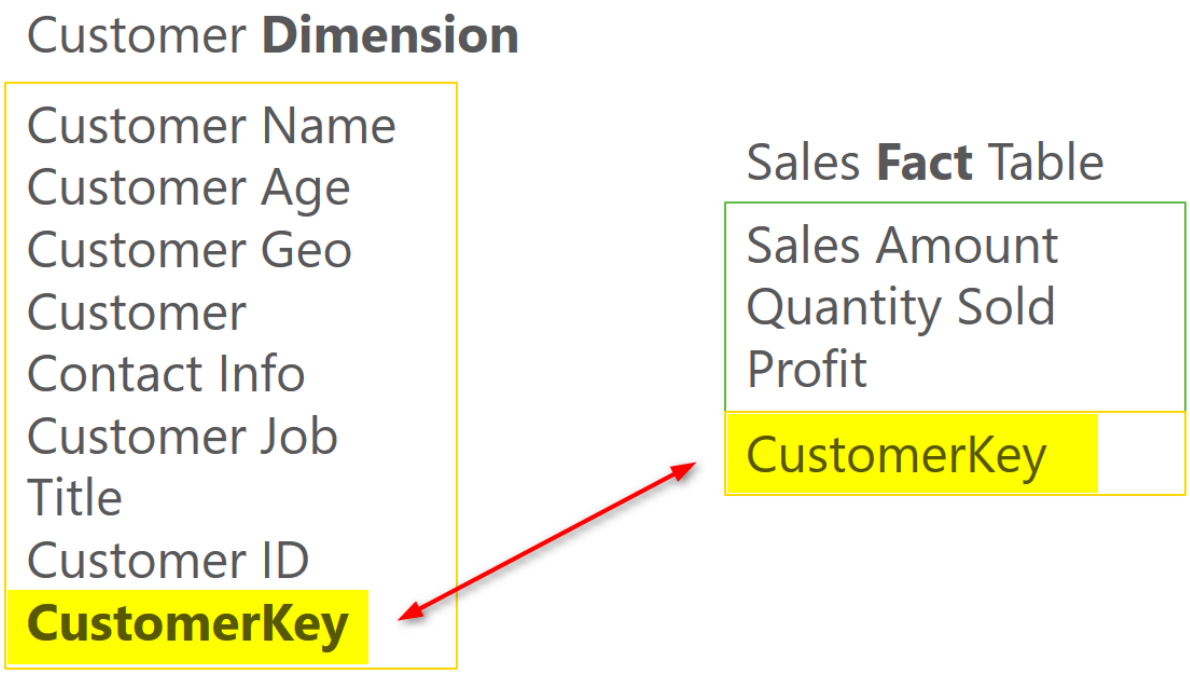
Any descriptive information will be kept in Dimension tables. For example; customer name, customer age, customer geoinformation, customer contact information, customer job, customer id, and any other customer related information will be kept in a table named Customer Dimension.

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

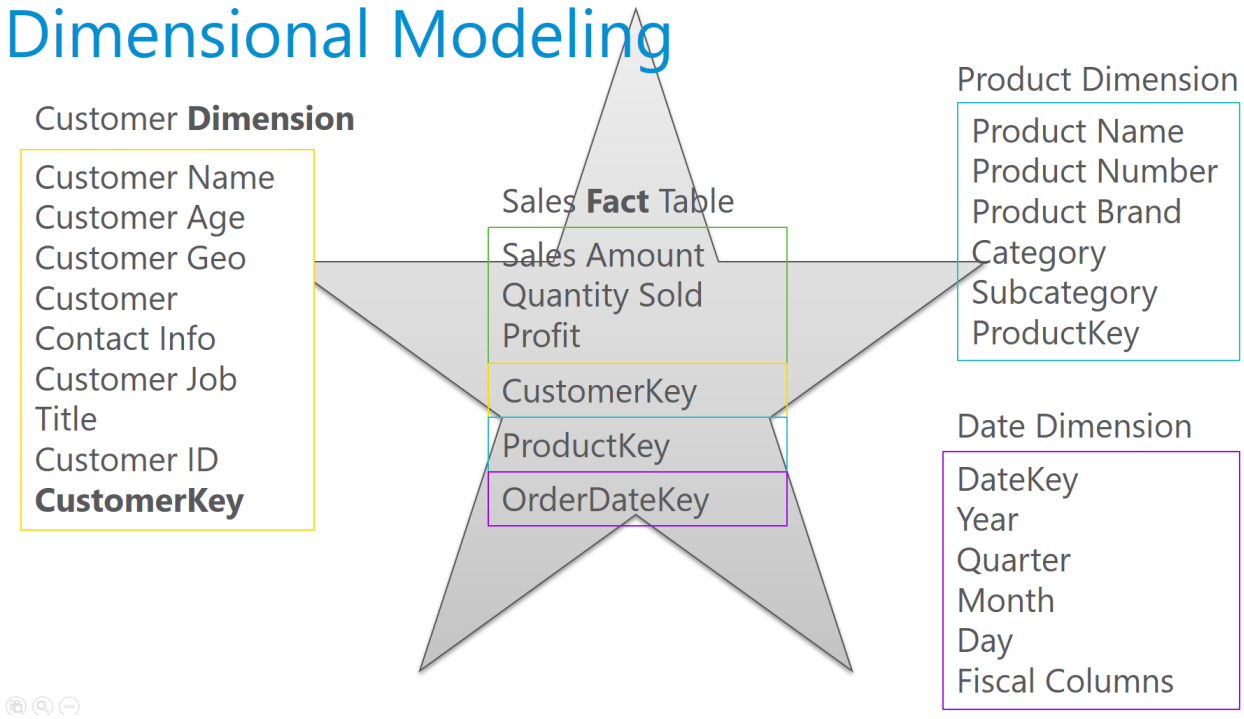
auto increment (Identity in SQL Server terminology). This column should be unique per each row in the dimension table. This column will be the primary key of this table and will be used in fact table as a relationship. This column SHOULDN'T be the ID of the source system. There are several reasons for 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;



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.



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. Some 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?

A 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?

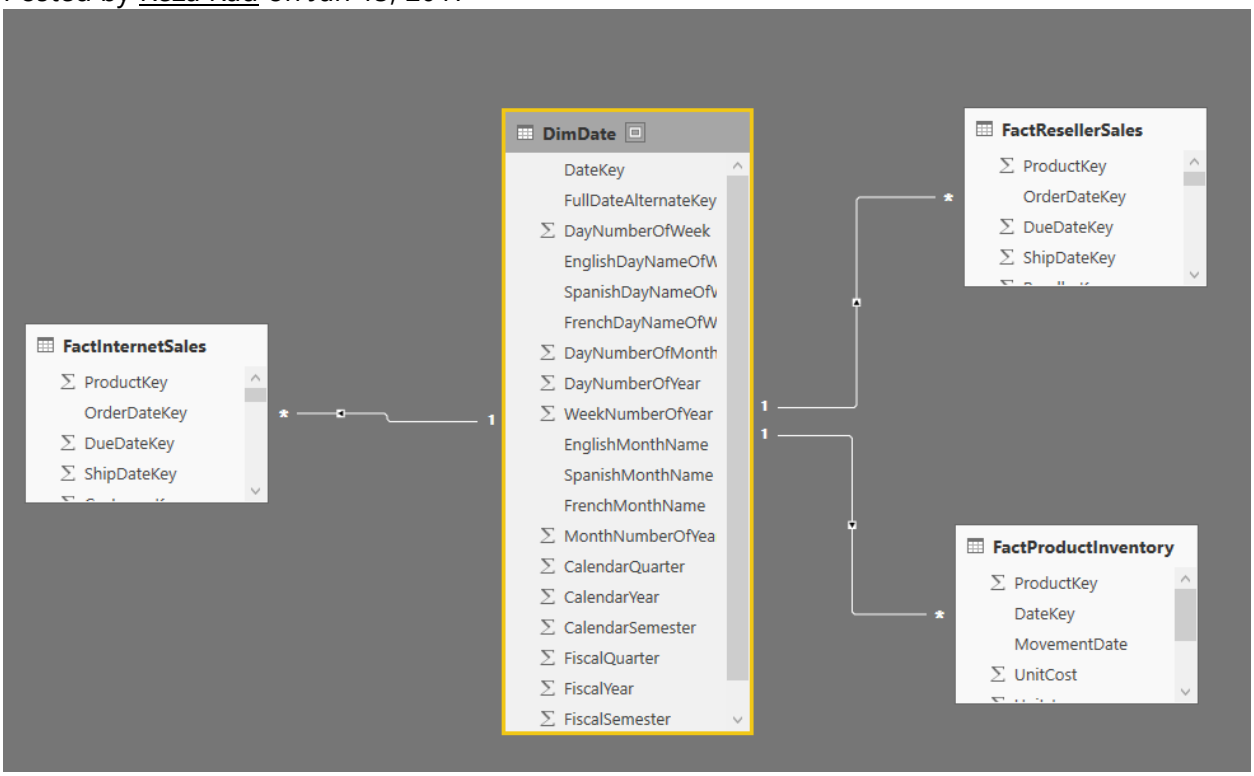
Filtering part of the data before loading it into memory is cost and performance effective.
Solution: DO Filter part of the data that is not required.

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.

Do You Need a Date Dimension?

Posted by Reza Rad on Jan 13, 2017



I've heard this question many times; "Do you think we need a date dimension?". some years ago, the date dimension has been used more, but nowadays less! The main reason is that some tools are using their built-in date hierarchy. As an example, Power BI has a built-in date hierarchy that is enabled by default on top of all date fields which gives you a simple hierarchy of year, quarter, month, and day. Having this hierarchy brings the question most of the time that do I need to have a date dimension still? of I can simply use this date hierarchy? In this post, I'll answer this question and explain it with reasons. Majority of this post is conceptual and can be used across all BI tools, however, part of it is Power BI focused. If you like to learn more about Power BI, read Power BI online book from Rookie to Rock Star.

What is Date Dimension?

Date Dimension is a table that has one record per each day, no more, no less! Depends on the period used in the business you can define the start and end of the date dimension. For example, your date dimension can start from 1st of Jan 1980 to 31st of December of 2030. For every year normally you will have 365 records (one record per year), except leap years with 366 records. Here is an example screenshot of a date dimension records;

	Year	Month	Day	FullDateAl...	DateKey	DateFullName	Fiscal Year	Fiscal Quarter	Calendar Quar...	IsWeekDay	DayOfWeek	Month Name	Day of Week Name
1	2013	6	15	6/15/2013	20130615	15 June 2013	2013	4	2	0	6	June	Saturday
2	2013	6	16	6/16/2013	20130616	16 June 2013	2013	4	2	0	0	June	Sunday
3	2013	6	17	6/17/2013	20130617	17 June 2013	2013	4	2	1	1	June	Monday
4	2013	6	18	6/18/2013	20130618	18 June 2013	2013	4	2	1	2	June	Tuesday
5	2013	6	19	6/19/2013	20130619	19 June 2013	2013	4	2	1	3	June	Wednesday
6	2013	6	20	6/20/2013	20130620	20 June 2013	2013	4	2	1	4	June	Thursday
7	2013	6	21	6/21/2013	20130621	21 June 2013	2013	4	2	1	5	June	Friday
8	2013	6	22	6/22/2013	20130622	22 June 2013	2013	4	2	0	6	June	Saturday
9	2013	6	23	6/23/2013	20130623	23 June 2013	2013	4	2	0	0	June	Sunday
10	2013	6	24	6/24/2013	20130624	24 June 2013	2013	4	2	1	1	June	Monday
11	2013	6	25	6/25/2013	20130625	25 June 2013	2013	4	2	1	2	June	Tuesday
12	2013	6	26	6/26/2013	20130626	26 June 2013	2013	4	2	1	3	June	Wednesday
13	2013	6	27	6/27/2013	20130627	27 June 2013	2013	4	2	1	4	June	Thursday
14	2013	6	28	6/28/2013	20130628	28 June 2013	2013	4	2	1	5	June	Friday
15	2013	6	29	6/29/2013	20130629	29 June 2013	2013	4	2	0	6	June	Saturday
16	2013	6	30	6/30/2013	20130630	30 June 2013	2013	4	2	0	0	June	Sunday
17	2013	7	1	7/1/2013	20130701	01 July 2013	2014	1	3	1	1	July	Monday
18	2013	7	2	7/2/2013	20130702	02 July 2013	2014	1	3	1	2	July	Tuesday
19	2013	7	3	7/3/2013	20130703	03 July 2013	2014	1	3	1	3	July	Wednesday
20	2013	7	4	7/4/2013	20130704	04 July 2013	2014	1	3	1	4	July	Thursday
21	2013	7	5	7/5/2013	20130705	05 July 2013	2014	1	3	1	5	July	Friday
22	2013	7	6	7/6/2013	20130706	06 July 2013	2014	1	3	0	6	July	Saturday

Date Dimension is not a big dimension as it would be only ~3650 records for ten years, or even ~36500 rows for 100 years (and you might never want to go beyond that). Columns will normally be all descriptive information about the date, such as Date itself, year, month, quarter, half year, the day of the month, the day of year….

Date Dimension will be normally loaded once and used many times after it. So it shouldn't be part of your every night ETL or data load process.

Why Date Dimension?

Date Dimension is useful for scenarios mentioned below;

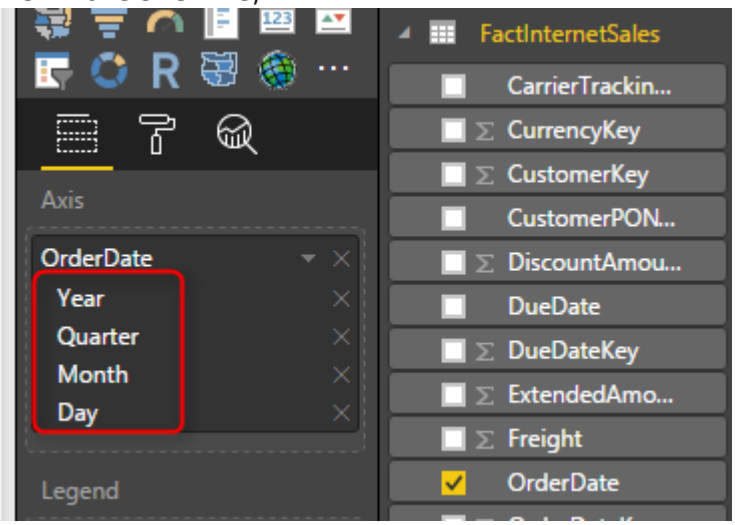
1. Ability to slice and dice by many date attributes such as week number, half year, the day of the year, etc.
2. Consistency in Analysis

3. Ability to do analysis based on public holidays (Easter Monday, Good Friday, etc)
4. Some BI tools extended functions need to work with a Date Dimension

Let's look at reasons one by one;

Powerful Slice and Dice

If you have worked with Power BI, you know that there is a date hierarchy built-in which gives you fields such as Year, Quarter, Month, and Day. This hierarchy normally adds automatically to date fields. Here is an example of how it looks like;

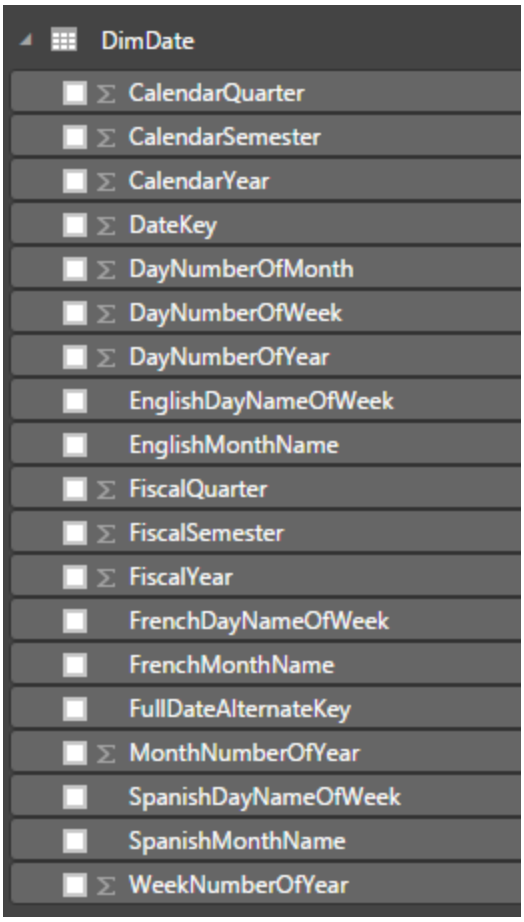


This built-in hierarchy is very helpful, but what about times that you want to slice and dice date fields based on something other than these four fields? For example, questions that you cannot answer with this hierarchy are;

1. Weekday analysis of revenue (by Monday, Tuesday, …., Sunday)
2. Which week in the year generates the least revenue or most revenue?
3. What about Fiscal? What is the revenue for Fiscal year?
4. And many other questions.

The fact is; when you use the simple hierarchy of Year, Quarter, Month, and Day, you are limited to only slice and dice by these fields. If you want to step beyond that, then you have to create additional fields to do it. That's when a date dimension with all of these fields is handy. Here are some fields that you

can slice and dice based on that in Date Dimension (and this is just a very short list, a real date dimension might have twice, three times, or 4 times more fields than this! Yes more fields means more power in slicing and dicing)



You can answer questions above without a date dimension of course, for example; to answer the first question above, you will need to add Weekday Name because you would need to sort it based on Weekday number, so you have to bring that as well. To answer the second question, you need to bring Week number of year. For the third question, you need to bring fiscal columns, and after a while, you will have heaps of date related columns in your fact table!

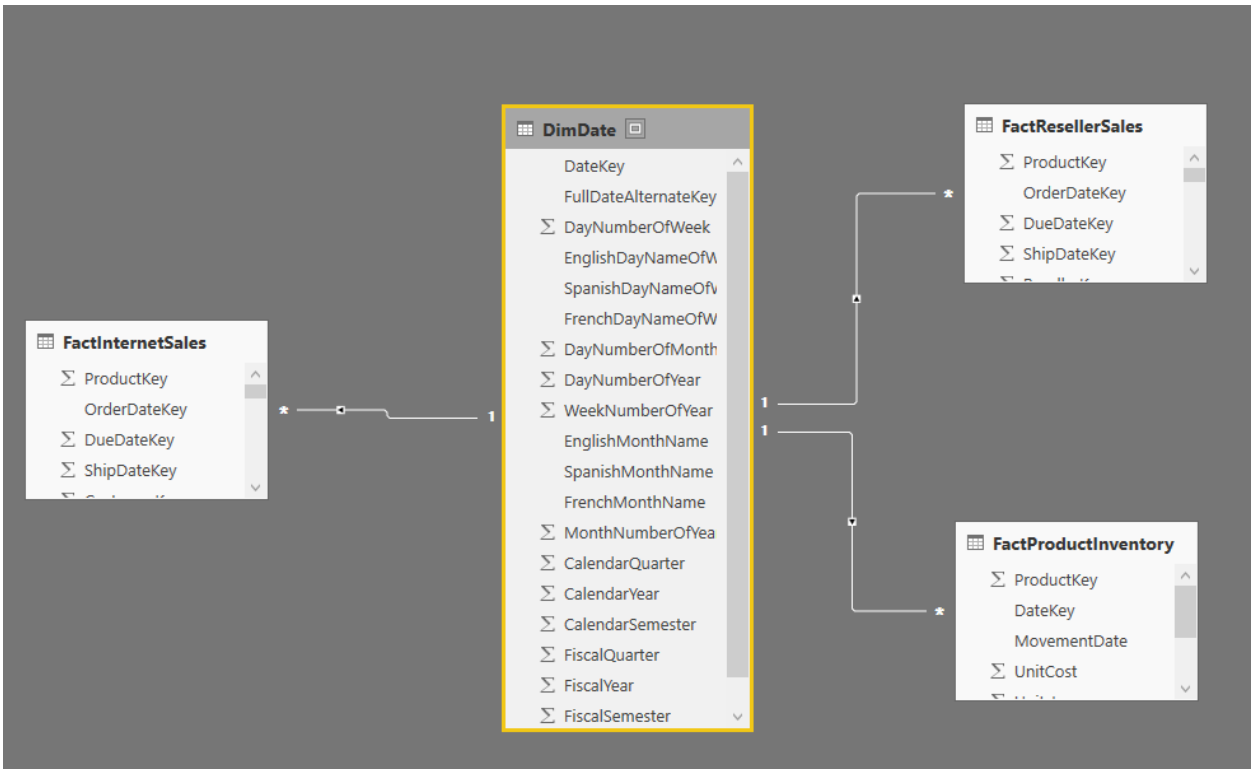
And the worst part is that this is only this fact table (let's say Sales fact table). Next month you'll bring Inventory fact table, and the story begins with date

fields in the Inventory fact table. This simply leads us to the second reason for the date dimension; Consistency.

Consistency in Analysis

You don't want to be able to slice and dice your Sales data by the week number of year, but not having this feature in Inventory data! If you repeat these columns in all fact tables, you will have big fact tables which are not consistent though, if you add a new column somewhere you have to add it to all other tables. What about a change in the calculation? Believe me, you never want to do it.

Date Dimension, on the other hand, is a dimension that is shared between all fact tables. You add all fields and calculations here, and fact tables are only related to this. This is a consistent approach. Here is a simple view of a data warehouse schema with a date dimension shared between two fact tables.



Public Holidays Insight

In most businesses nowadays public holidays or special event's insight is an important aspect of reporting and analysis. Some of the analysis is like;

- Was the sales in Easter this year better than last year? (you don't want to calculate Easter in Power BI of course)
- How is the revenue is public holidays compared with weekends?
- How were the sales for a store in the opening day? (special event)
- and many other questions….

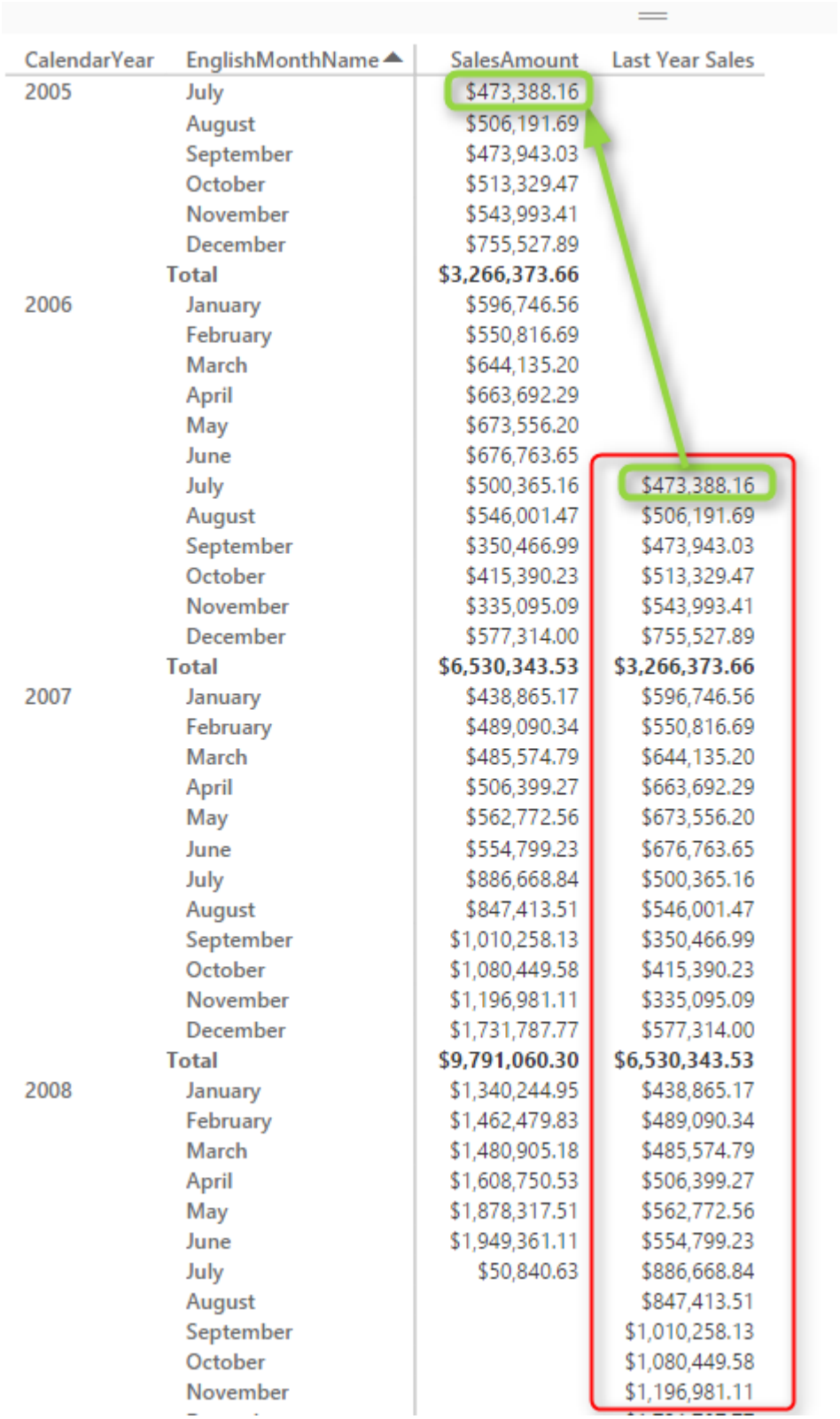
One of the best use cases of date dimension is public holidays and special events. I have shown previously in another post how to fetch public holidays live in Power BI. Calculating some public holidays are time-consuming and not easy. Easter, for example, changes every year. You can also have some fields in date dimension that mention what the special day is (opening, closing, massive sale….). Your date dimension can be enhanced a lot with insight from these fields. Here is an example of public holidays information in a date dimension;

	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

Extended Functions for Time Intelligence

BI Tools in the market has some extended functions that give you insight related to date, named as time intelligence functions. For examples, talking about Power BI, you can leverage heaps of DAX functions, such as TotalYTD to calculate year to date, ParallelPeriod to find the period parallel to the current period in the previous year or quarter and many other functions. These functions sometimes need to work with a date dimension. Otherwise they won't return a correct result!

Yes, you've read it correctly, DAX time intelligence functions won't work properly if you don't have a proper date dimension. proper date dimension is a table that has a full date column in one of the fields and it has **no gaps** in dates. This means that if you are using Sales transaction table which has an OrderDate field as an input to DAX time intelligence functions, and if there is no sales transaction for 1st of January, then the result of time intelligence functions won't be correct! It won't give you an error, but it won't show you the correct result as well, this is the worst type of error might happen. So having a Date Dimension can be helpful in this kind of scenarios as well.



CalendarYear	EnglishMonthName	SalesAmount	Last Year Sales
2005	July	$473,388.16	
	August	$506,191.69	
	September	$473,943.03	
	October	$513,329.47	
	November	$543,993.41	
	December	$755,527.89	
	Total	**$3,266,373.66**	
2006	January	$596,746.56	
	February	$550,816.69	
	March	$644,135.20	
	April	$663,692.29	
	May	$673,556.20	
	June	$676,763.65	
	July	$500,365.16	$473,388.16
	August	$546,001.47	$506,191.69
	September	$350,466.99	$473,943.03
	October	$415,390.23	$513,329.47
	November	$335,095.09	$543,993.41
	December	$577,314.00	$755,527.89
	Total	**$6,530,343.53**	**$3,266,373.66**
2007	January	$438,865.17	$596,746.56
	February	$489,090.34	$550,816.69
	March	$485,574.79	$644,135.20
	April	$506,399.27	$663,692.29
	May	$562,772.56	$673,556.20
	June	$554,799.23	$676,763.65
	July	$886,668.84	$500,365.16
	August	$847,413.51	$546,001.47
	September	$1,010,258.13	$350,466.99
	October	$1,080,449.58	$415,390.23
	November	$1,196,981.11	$335,095.09
	December	$1,731,787.77	$577,314.00
	Total	**$9,791,060.30**	**$6,530,343.53**
2008	January	$1,340,244.95	$438,865.17
	February	$1,462,479.83	$489,090.34
	March	$1,480,905.18	$485,574.79
	April	$1,608,750.53	$506,399.27
	May	$1,878,317.51	$562,772.56
	June	$1,949,361.11	$554,799.23
	July	$50,840.63	$886,668.84
	August		$847,413.51
	September		$1,010,258.13
	October		$1,080,449.58
	November		$1,196,981.11

In this post, I've explained how to use some of DAX time intelligence functions in Power BI.

Do I Need a Date Dimension?

Now is the time to answer the question; "Do I need a date dimension"? The answer is: Yes, IMHO. Let me answer that in this way: I can do a BI solution without date dimension, but would you build a table without hammer?! Would you build a BI solution that has some date columns in each fact table, and they are not consistent with each other? would you overlook having a special event or public holidays dates insight in your solution? Would you only stick to year, quarter, month, and date slicing and dicing? Would you like to write all your time intelligence functions yourself? You don't. Everyone has the same answer when I explain it all.

I have been working with countless BI solutions in my work experience, and there is always a requirement for a Date Dimension. Sooner or later you get to the point that you need to add it. In my opinion, add it at the very beginning, so you don't get into the hassle of some rework.

How to Build a Date Dimension?

Now that you want to build a date dimension, there are many ways to build it. Date Dimension is a dimension table that you normally load once and use it always. Because 1st of Jan 2017 is always 1st of Jan 2017! Data won't change, except bringing new holiday or special event dates. I mention some of the ways to build a date dimension as below (there are many more ways to do it if you do a bit of search in Google);

Date Dimension T-SQL Script

You can use a written T-SQL Script and load date dimension into a relational database such as SQL Server, Oracle, MySQL, etc. Here is a T_SQL example I wrote a few years ago.

Power Query Script

I have written a series of blog posts explaining how you can do the whole date dimension in Power Query. The blog post is a bit old, I will write a new one shortly and will show an easier way of doing this in Power BI.

- Date Dimension in Power Query – Part 1: Calendar Columns
- Date Dimension in Power Query – Part 2: Fiscal Columns
- Date Dimension in Power Query – Part 3: Public Holidays

Fetch Public Holidays

I have written another blog post that mentioned how to fetch public holidays from the internet and attach it to a date dimension in Power BI.

DAX Calendar Functions

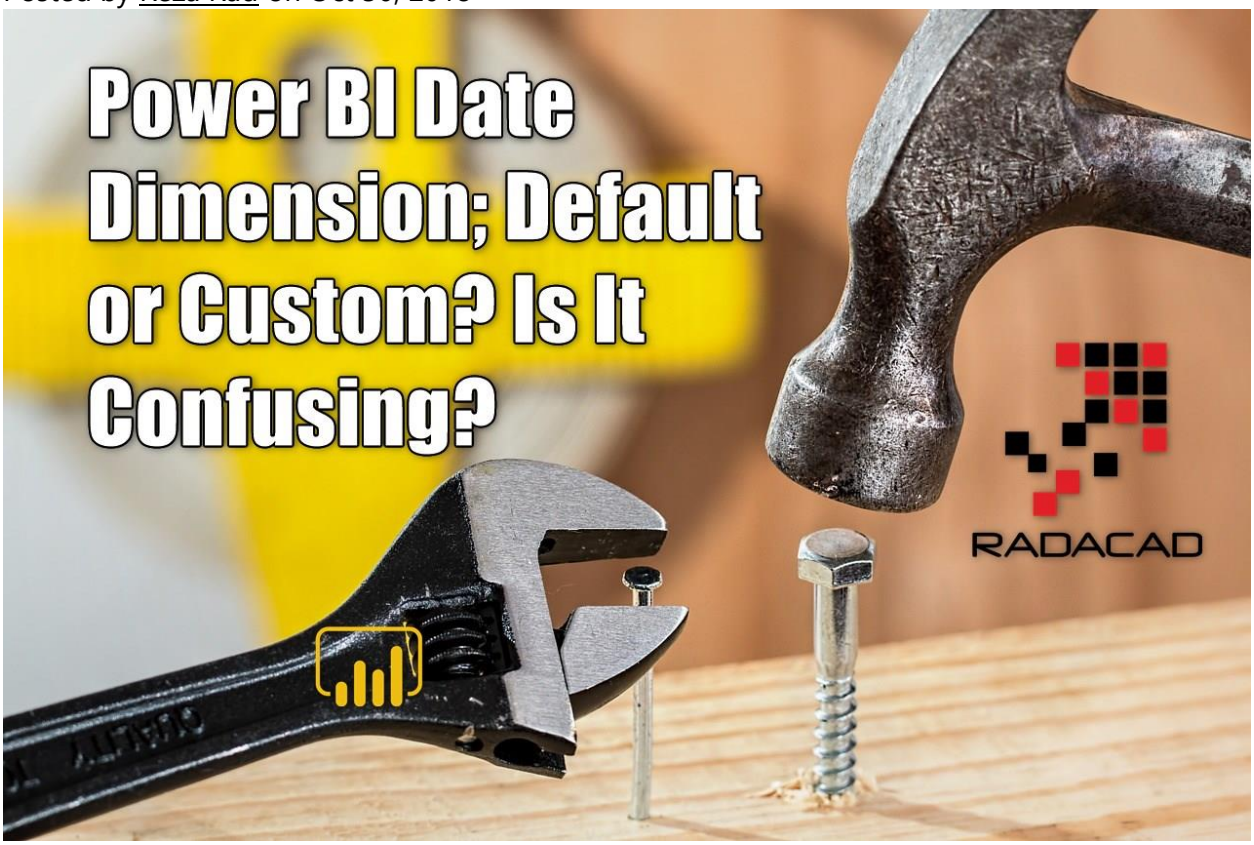
You can use Calendar() function in DAX to generate a date dimension very quickly as well.

Using Date Dimension Multiple Times?

Using a dimension multiple times in a data warehouse called Role Playing dimension. I have explained how to do role-playing dimension for DimDate in this blog post.

Power BI Date Dimension; Default or Custom? Is It Confusing?

Posted by Reza Rad on Oct 30, 2018



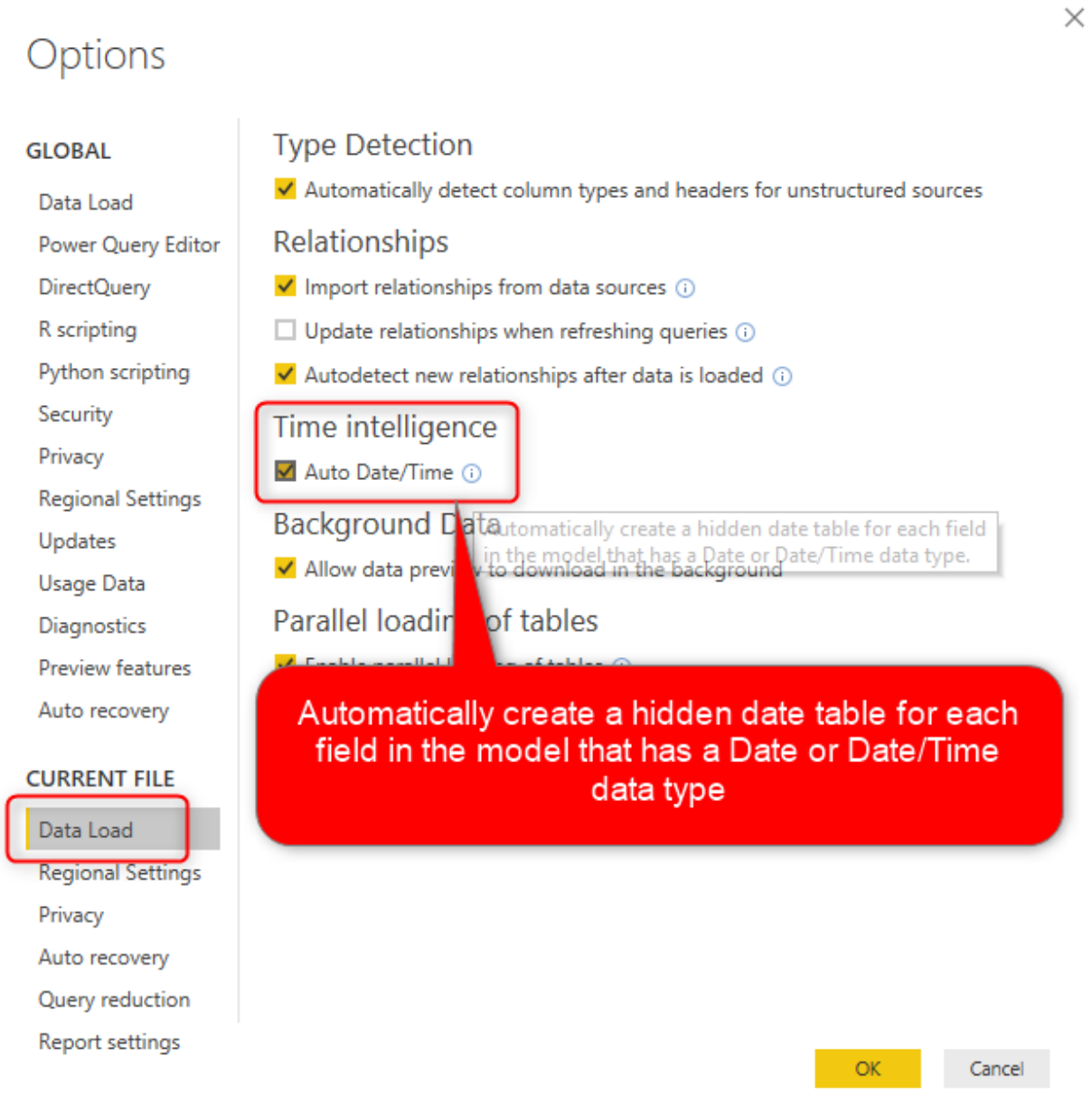
If you have worked with Power BI for some time, you know that there are two types of the Date dimensions; Custom or built-in/Default. It is always confusing for people, that which date dimension is good to use, and what is the difference between these two approaches. Also based on the selection of the type of Date dimension, your DAX calculations may differ a little bit. In this post, I'll explain all you need to know about the default and custom date dimension in Power BI and help you to choose the right one for your Power BI solution. If you like to learn more about Power BI, read the Power BI book from Rookie to Rock Star.

Power BI Default Date Dimension

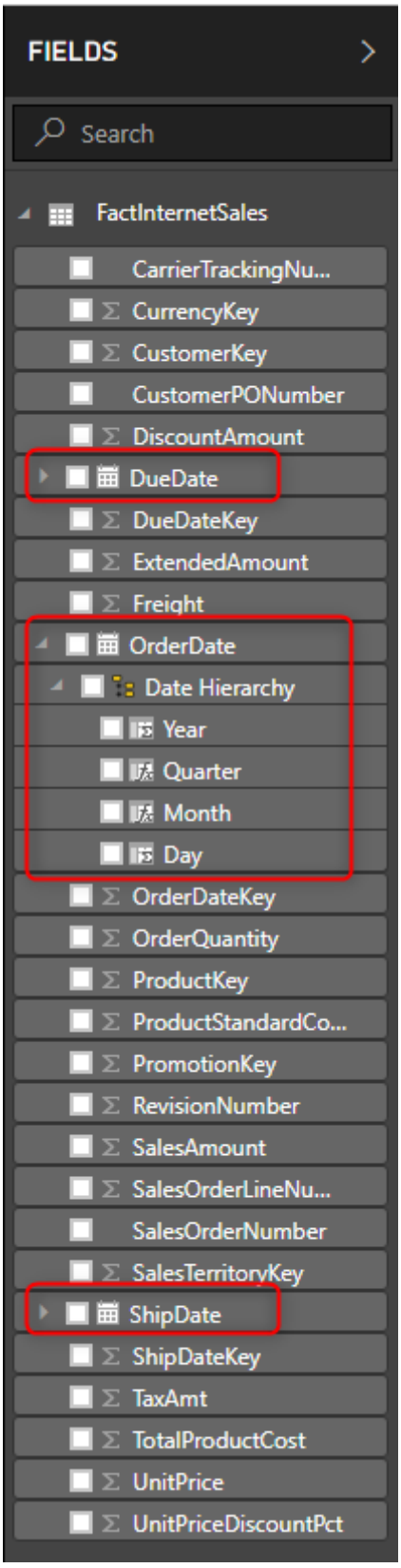
The Power BI date dimension is still a bit of mystery for some people. Many people still are not aware that there is a default or built-in date dimension. This date dimension hasn't existed at the very beginning of Power BI Desktop (which was July 2018). It came a bit after into the party. The purpose of this default date dimension is to ease the way that time intelligence calculations work in Power BI and make the whole experience easier for the user. Let's see how this dimension works.

Power BI Creates a Default Date Dimension for every single date field in your dataset

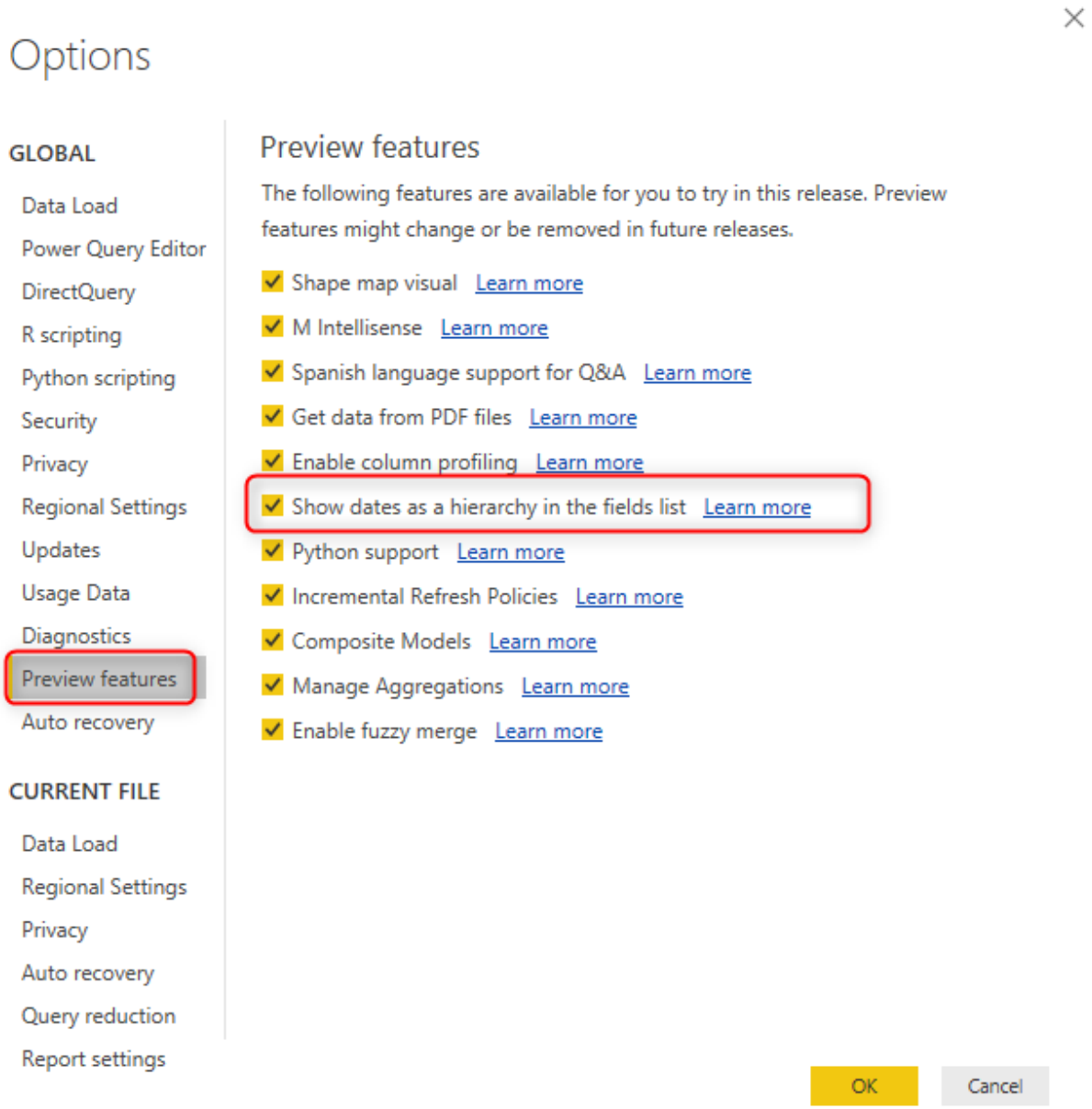
By default (you can change the default configuration), Power BI creates a date dimension for every single date field in your dataset. The date dimension is a generic date dimension with normal fields such as; Year, Month, Quarter, Day, etc. The configuration in Power BI Desktop that allows the model to create the default date dimension is here:
In the Power BI Desktop, File menu -> Option and Settings -> Options
In the Options window, under Current File, Data Load; Time Intelligence: Auto Date/Time



Having this item enabled means that Power BI automatically create a hidden date table for each field in the model that has a date or date/time data type. The reason that it is not easy to find the default Date table is that it is HIDDEN! The main reason for having this table hidden is that user will face too much confusion if you see a Date table per each date data type column. Here is a model created with the default Date dimension:


FIELDS
Search
FactInternetSales
CarrierTrackingNu...
CurrencyKey
CustomerKey
CustomerPONumber
DiscountAmount
DueDate
DueDateKey
ExtendedAmount
Freight
OrderDate
Date Hierarchy
Year
Quarter
Month
Day
OrderDateKey
OrderQuantity
ProductKey
ProductStandardCo...
PromotionKey
RevisionNumber
SalesAmount
SalesOrderLineNu...
SalesOrderNumber
SalesTerritoryKey
ShipDate
ShipDateKey
TaxAmt
TotalProductCost
UnitPrice
UnitPriceDiscountPct

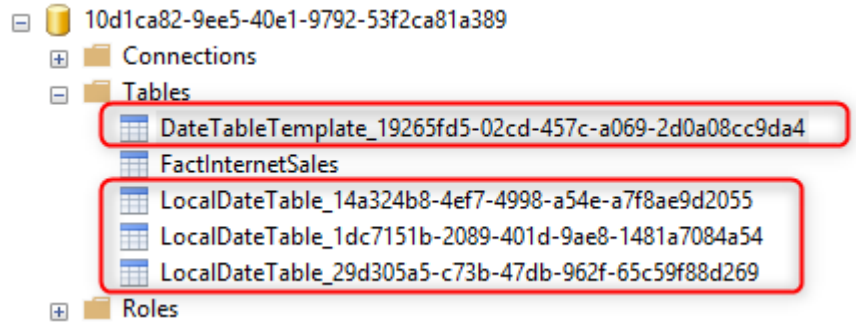
In the screenshot above you see that there are three date fields in the FactInternetSales: DueDate, OrderDate, and ShipDate. There is a date dimension TABLE for each of these fields that you cannot see here. But if you have an option in the Power BI settings enabled, then you can see the Date hierarchy under that table, which shows there is a table behind the scene. (Note that even if you don't see the date hierarchy in this view, it doesn't mean that Power BI default date dimension is not created). If you want to enable the feature to SHOW you the Date hierarchy, you can do it under Options, Preview Features. and enable "Show dates as a hierarchy in the fields list."



Why a Date table for every single Date Field?

Well, now that you know, there is a default Date dimension, your second question might be: Why Power BI creates it multiple times, for each date field?! The reason is that the Date usually acts like a role-playing dimension. Sometimes it might be Due Date, sometimes Ship Date, and sometimes, Order Date. All of these options are date fields, but their values and behavior are

different. If you read my post about Calculated tables, you know that one of the ways to do role-playing dimension in Power BI is to create copies of the Date dimension. This is what Power BI does behind the scene automatically. Power BI Creates a template for Date table first, and then copy it for every single date or date/time field in the model. Here is a glance at the behind the scene of the Power BI model:

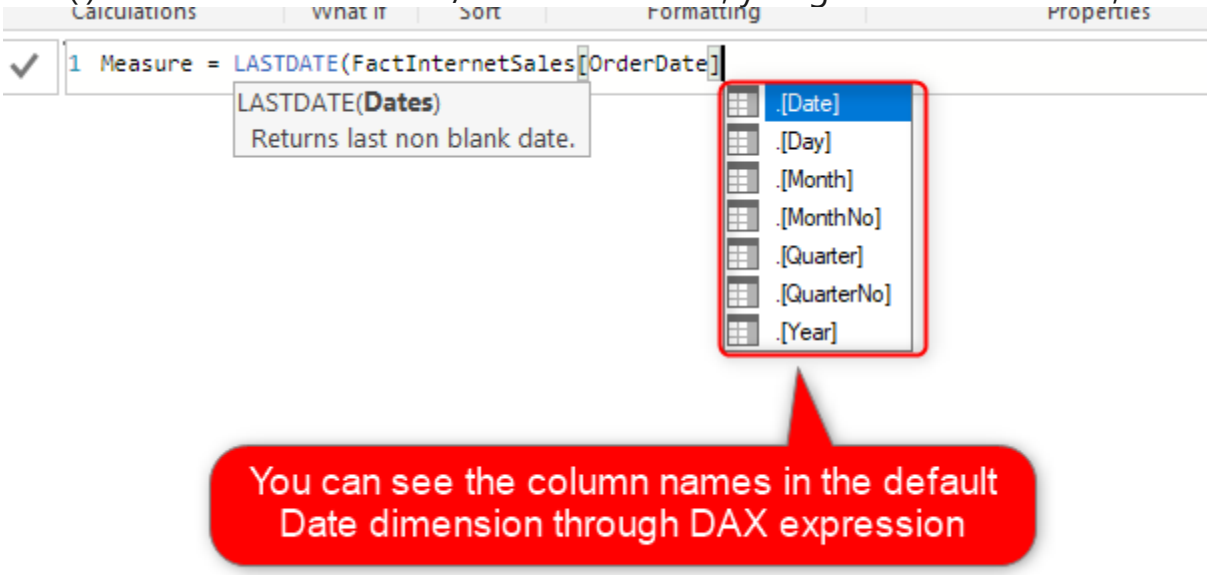


Note that you cannot see the view above in the Power BI Desktop. You can use tools like Power BI Helper to get to that information though. In the screenshot above, you can see that there is a DateTableTemplate, and then three date tables copied from that template.

What does the Default Date table look like?

There are many variations of the Date dimension in the world, and you may think, what does the default Date table look like? What columns are available there and what columns are not? Here is lit of columns:

The view above can be generated with tools such as Power BI Helper. But if you are interested to see the list of these column names in the Power BI Desktop, One way is to see it when you write a DAX expression, after typing dot (.) after the Date or Date/Time field name, you get the list of fields;



This might have been the mystery for many people when they write DAX statement; What is the list of fields that comes after the dot (.) in front of a date field name? Now you know the answer;

The Date field is NOT a field from the Power BI model point of view. It is a TABLE, a hidden table, and because of that, you can choose which column in that table you want to use within your expression.

Writing Time Intelligence Calculations with the Default Date Dimension

Writing DAX expressions for the time intelligence calculations using the default date dimension is simpler. You need to use your Date field name plus a ".[Date]" at the end. This means that you are using the [Date] field of the hidden Date table for the expression. For example, here is how a year to date calculation looks like:

```
1 Sales YTD = TOTALYTD(
2     SUM(FactInternetSales[SalesAmount]),
3     FactInternetSales[OrderDate].[Date])
```

If you don't use the ".[Date]" then you won't get the correct result, because Time Intelligence calculations need a DATE column to work with, and with the ".[Date]", you choose the date column in the default hidden Date table.

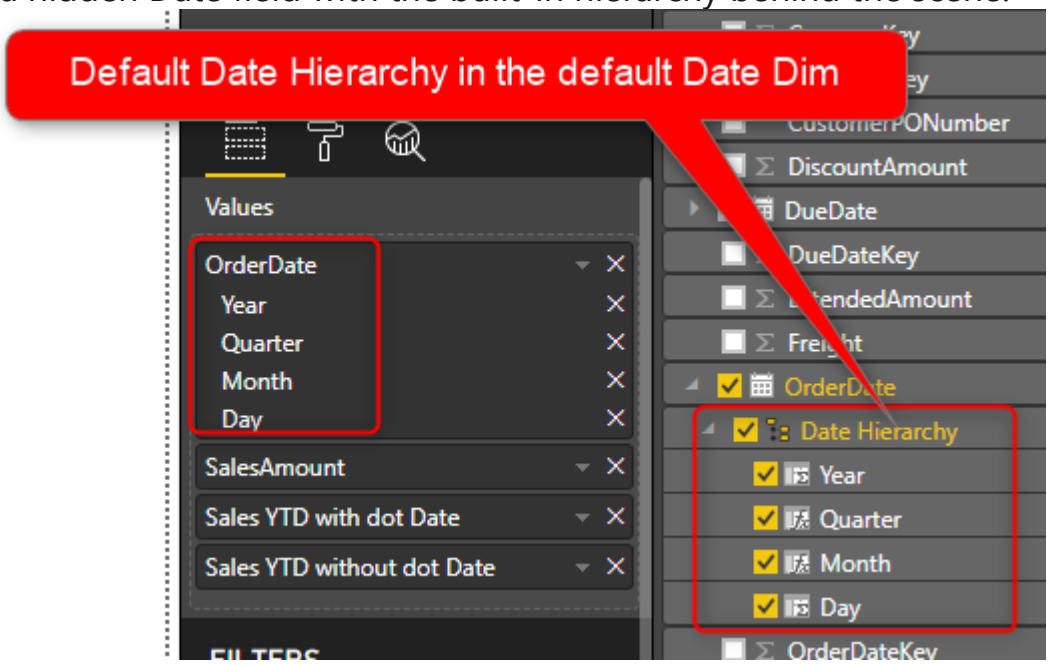
```
1 Sales YTD with dot Date = TOTALYTD(
2     SUM(FactInternetSales[SalesAmount]),
3     FactInternetSales[OrderDate].[Date])
```

Year	Quarter	Month	Day	SalesAmount	Sales YTD with dot Date	Sales YTD without dot Date
2005	Qtr 3	July	1	14,477.34	14,477.34	14,477.34
2005	Qtr 3	July	2	13,931.52	28,408.86	13,931.52
2005	Qtr 3	July	3	15,012.18	43,421.04	15,012.18
2005	Qtr 3	July	4	7,156.54	50,577.58	7,156.54
2005	Qtr 3	July	5	15,012.18	65,589.75	15,012.18
2005	Qtr 3	July	6	14,313.08	79,902.83	14,313.08
2005	Qtr 3	July	7	7,855.64	87,758.47	7,855.64
2005	Qtr 3	July	8	7,855.64	95,614.11	7,855.64
2005	Qtr 3	July	9	20,909.78	116,523.89	20,909.78
2005	Qtr 3	July	10	10,556.53	127,080.42	10,556.53
2005	Qtr 3	July	11	14,313.08	141,393.50	14,313.08
2005	Qtr 3	July	12	14,134.80	155,528.30	14,134.80
2005	Qtr 3	July	13	7,156.54	162,684.84	7,156.54

As you can see, the calculation doesn't work if you do not include the ".[Date]" in the expression. But if you include it, then all is good to do. Writing time intelligence based expressions using the default Date Dimension is very easy as you've seen here.

Default Date Dimension has a Built-in Date Hierarchy

One of the main benefits of using the default Date dimension is the built-in Date hierarchy of Year/Quarter/Month/Day it provides. Whenever you drag a Date field, Power BI automatically shows the hierarchy under visual, because there is a hidden Date field with the built-in hierarchy behind the scene.



Default Date Dimension Consumes Memory

Oh yes! Of course like any other table structures in the Power BI in-memory based structure, every date table, consumes memory. But it would do the same even if you create your custom date dimension! Whenever you do the role-playing dimension scenario, you are also consuming even more memory! The main difference is that Power BI default Date dimension will be created even if you do not want to do the date-based analysis on a date field! For

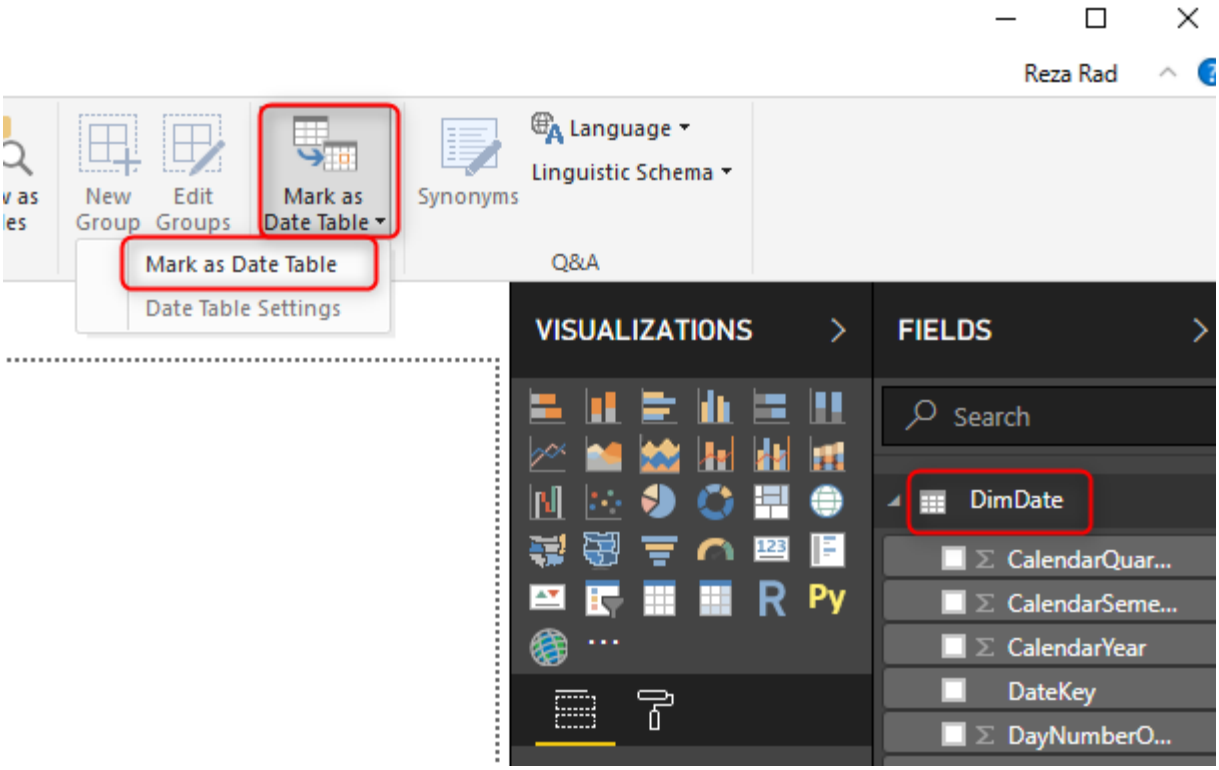
example, even if you don't use DueDate in your date-based analysis, still Power BI creates a date dimension for it. You cannot stop it for one field. You have to either stop the default creation of the Date dimension for the entire model or use it for the entire model; you cannot customize it per field. But with the custom date dimension, you can.

Custom Date Dimension

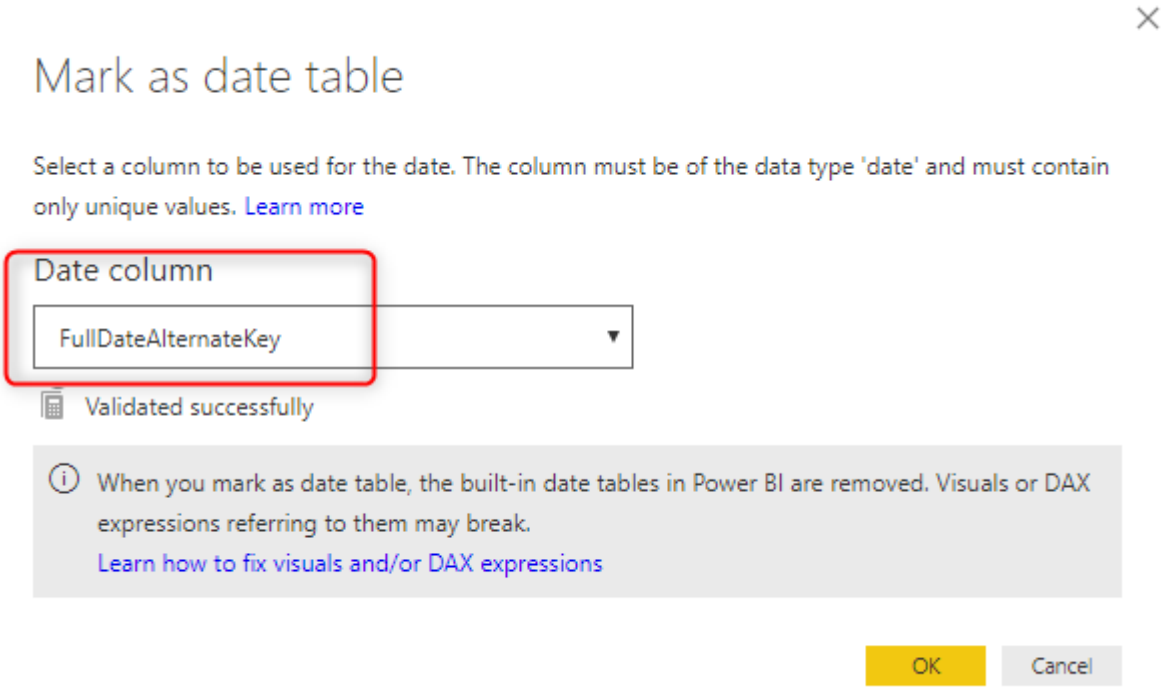
I have written a blog series about creating a custom Date dimension, and you can find many other examples of creating Date dimension in Power BI. There are many variations for that. You may think if the default Date dimension is available, then why should I have a custom date dimension? Well, there are some reasons for doing that. Let's first check how to you can use a custom date dimension.

Mark as Date Table

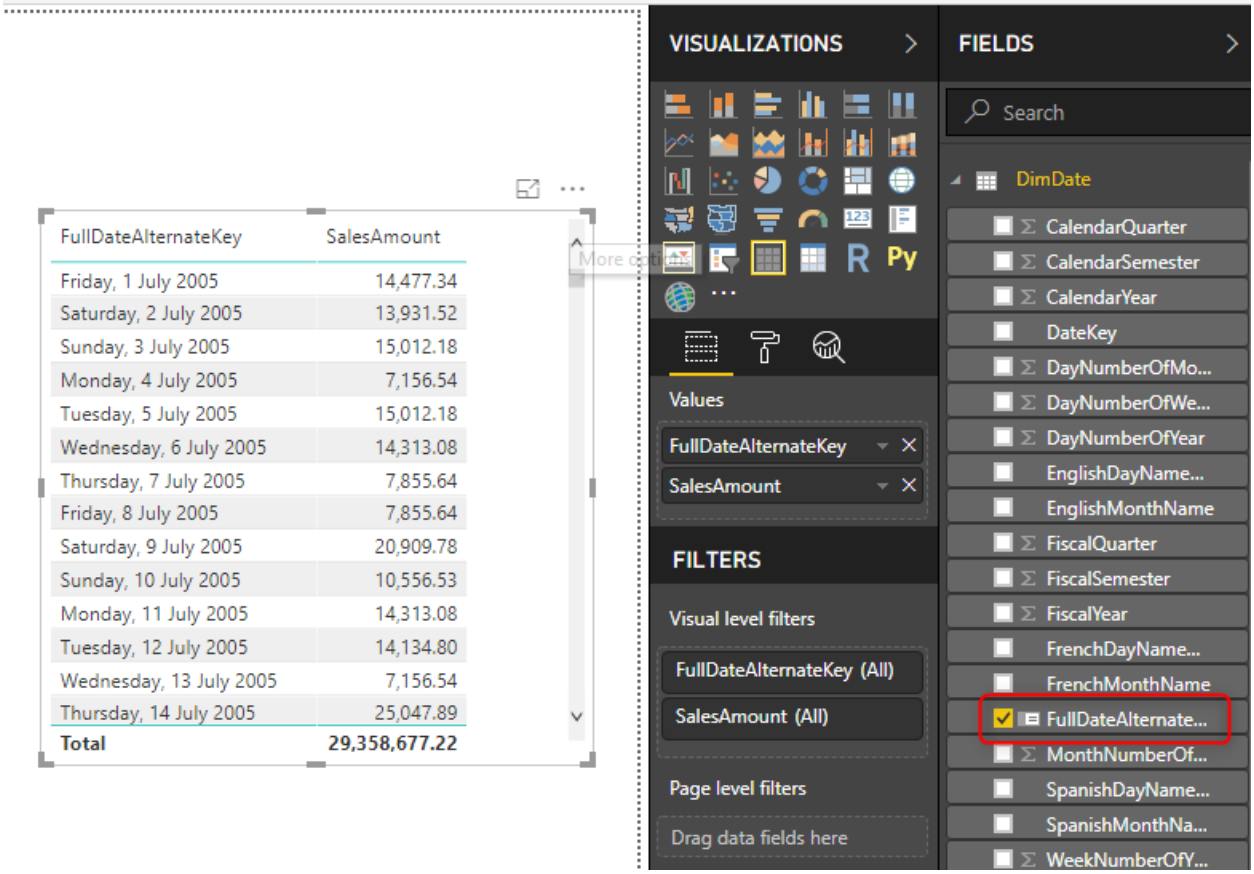
To use your custom Date dimension, you have to mark your table as a Date table. This is a very important step for Power BI because then It will know that the table to be used for time intelligence calculations is this table. You can, of course, have more than one Date table to be marked as the Date table (Because of the same reason of role-playing dimensions). If you have your custom Date table, here is how to mark it as a Date table; Go to Modelling tab in the Power BI Desktop, Then choose the custom Date table of yours, and select Mark as Date Table.



You have to select a full date column as the Date column of the Date Table as well.



Usually after this change, if you look at the icon of Date fields under your custom Date table, you will see them differently (without the default Date hierarchy), which shows the table now successfully marked as a Date table.



Writing Time Intelligence Calculations with Custom Date Dimension

The prerequisite for this step is to mark the table as Date Table, which we have done in the previous step, now you can write a DAX expression as easy as below:

```
1 Sales YTD = TOTALYTD(
2     SUM(FactInternetSales[SalesAmount]),
3     DimDate[FullDateAlternateKey])
```

```
1 Sales YTD = TOTALYTD(
2     SUM(FactInternetSales[SalesAmount]),
3     DimDate[FullDateAlternateKey])
```

FullDateAlternateKey	SalesAmount	Sales YTD
Friday, 1 July 2005	14,477.34	14,477.34
Saturday, 2 July 2005	13,931.52	28,408.86
Sunday, 3 July 2005	15,012.18	43,421.04
Monday, 4 July 2005	7,156.54	50,577.58
Tuesday, 5 July 2005	15,012.18	65,589.75
Wednesday, 6 July 2005	14,313.08	79,902.83
Thursday, 7 July 2005	7,855.64	87,758.47
Friday, 8 July 2005	7,855.64	95,614.11
Saturday, 9 July 2005	20,909.78	116,523.89
Sunday, 10 July 2005	10,556.53	127,080.42
Monday, 11 July 2005	14,313.08	141,393.50
Tuesday, 12 July 2005	14,134.80	155,528.30
Wednesday, 13 July 2005	7,156.54	162,684.84
Thursday, 14 July 2005	25,047.89	187,732.73

As you can see, in this expression, we do not need ".[Date]" to get to the date field. Because there is no hidden Date table for this column, you refer to the column to get the Date field. If you use the ".[Date]" here, you get an error, because there is no hidden Date table here.

Default or Custom Date Dimension?

Now that you know all about the default Date dimension, and custom Date dimension, is time for the main question: What is the difference?! when to choose what? The answer like many other situations is Depends! If you want to use generic date-based analysis and you want to build a model easier and quicker, then default Date dimension is very helpful in those scenarios. But if you want to do a special date-based analysis (as an example; public holiday based analysis), then custom Date dimension will give you more power. Let's look at differences in details:

Modeling with the default Date Dimension is Much Easier

It is true that using custom Date table means taking care of relationships, marking as a date table, creating the default hierarchy, etc. If you use custom Date dimension, you have to spend more time on doing all these configurations, whereas the default Date table, will take care of everything for you.

If you have a Date field, and you are not using it, Remove It!

When you have a date field, Power BI automatically creates a table for it, create the relationship of that to the main date field, and the table consumes memory! If you do not intend to use that field, then remove it to save memory. This is also important for any other fields that you do not use in the model; remove it to get better memory consumption, but it is even more important for Date fields because behind the scene you will have not just a field, but also a table.

Customized and Complex Analysis is Easier with Custom Date Dimension

While the default Date dimension is easier for generic date-based analysis, The custom Date dimension is a very powerful option when it comes to customized analysis. As an example, let's say you want to create a date-based analysis based on public holidays. How can you do that with the default Date dimension? Well, the answer is to create a list of public holidays as a table and merge or join it to the date field, which means creating your custom Date dimension. Here in this post, I explained an example of fetching public holidays for a custom Date dimension in Power BI.

	date	name	url	description	flag_day	age	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	null	null
3	6/02/2015	Waitangi Day	https://en.wikipedia.org/wiki/Waitangi_Day	null	1	175	null
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: Ēostre; Greek: Πάσχα, Paskha; Aramaic: אסתר Pas...	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	null
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: Crīstesmæsse, 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... In South Africa, Boxing Day was renamed Day of Goodwill in 1994. In Irela	null	null	Proclamation Day; St. Stephen's Day
12	28/12/2015	Boxing Day Public Holiday	https://en.wikipedia.org/wiki/Boxing_Day	Boxing Day is traditionally a day following Christmas when wealthy pe... In South Africa, Boxing Day was renamed Day of Goodwill in 1994. In Irela	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	null	null
15	6/02/2016	Waitangi Day	https://en.wikipedia.org/wiki/Waitangi_Day	null	1	176	null
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: Ēostre; Greek: Πάσχα, Paskha; Aramaic: אסתר Pas...	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	null
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: Crīstesmæsse, 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... In South Africa, Boxing Day was renamed Day of Goodwill in 1994. In Irela	null	null	Proclamation Day; St. Stephen's Day

Another example is when you need more than the default hierarchy when you want to create a weekly hierarchy, financial calendar hierarchy, and many other scenarios. The default Date dimension is good for generic analysis, but not when it comes to more customization.

Common Mistake to Avoid!

One big common mistake that I've seen a lot, and it comes from the confusion of the default Date dimension vs. custom Date table, is that we see both in one model! Let's see what I mean:
Let's say you want to use a custom Date dimension, and you add it to your model, you create the relationship to the date field in the fact table, and then you **DO NOT Mark it as a Date Table**! That is where all mistakes start! When you do not mark it as Date Table, then you allow Power BI to create the default date dimension for the date field even under this table (considering that the creation of default date tables in enabled in the options, which is ON by default). As a result, you have a custom Date table, but you also have a default Date dimension for the date field under your custom date table! It means you are using extra memory twice! And your DAX expressions also becomes even more wrong like this:

```
1 Sales YTD = TOTALYTD(
2     SUM(FactInternetSales[SalesAmount]),
```

3 *DimDate[FullDateAlternateKey].[Date])*

This is Wrong! You will get the correct result in the visualization, but doing it this way is Wrong! Because if you are going to use the default Date table, then what is the point of adding extra custom Date dimension? If you are going to use the custom Date table, then you HAVE TO mark it as Date Table.

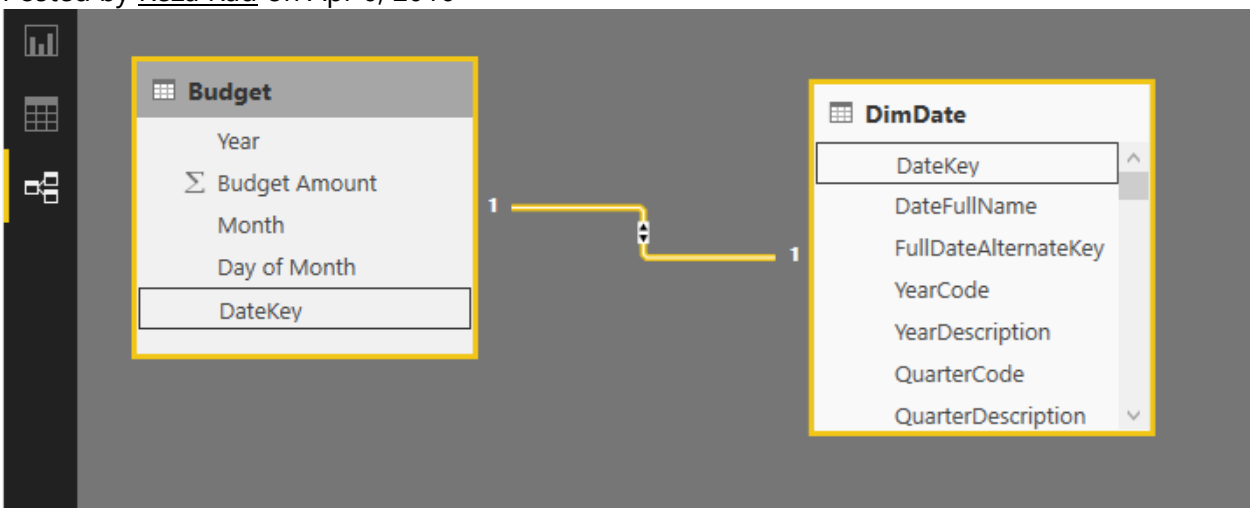
Summary

Date dimension and its behavior in Power BI can be confusing if you don't know about the default Date dimension, and how to use your custom Date dimension. In this post, I explained the differences between these two and elaborated the difference. Please let me know in the comments below if you still have any questions about the Date dimension for your Power BI solution.

Part II: Relationships

Relationship in Power BI with Multiple Columns

Posted by Reza Rad on Apr 6, 2016



You can create relationships in Power BI between tables. Relationships are useful for some functions to work across multiple tables and produce the result. The relationship between tables also makes visualization and report elements more efficient, because the result of selection in one chart can affect another chart from a different table. However, there is a limitation in the Power BI relationship that you can't create a relationship based on more than one column. In other words, if you want to create a relationship (or join two tables) with two or more columns, you cannot! Fortunately, there is a workaround that I'll explain in this post. For this post, you need to be familiar with Power BI and Power Query, if you are not, read them through Power BI online book.

Defining the Problem

Assume that we have a budget table with the fiscal year, fiscal period, and budget amount. Here is a screenshot of this table:

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	Mth12	50000
2011	Mth1	50000
2011	Mth2	55000
2011	Mth3	55000
2011	Mth4	65000
2011	Mth5	55000

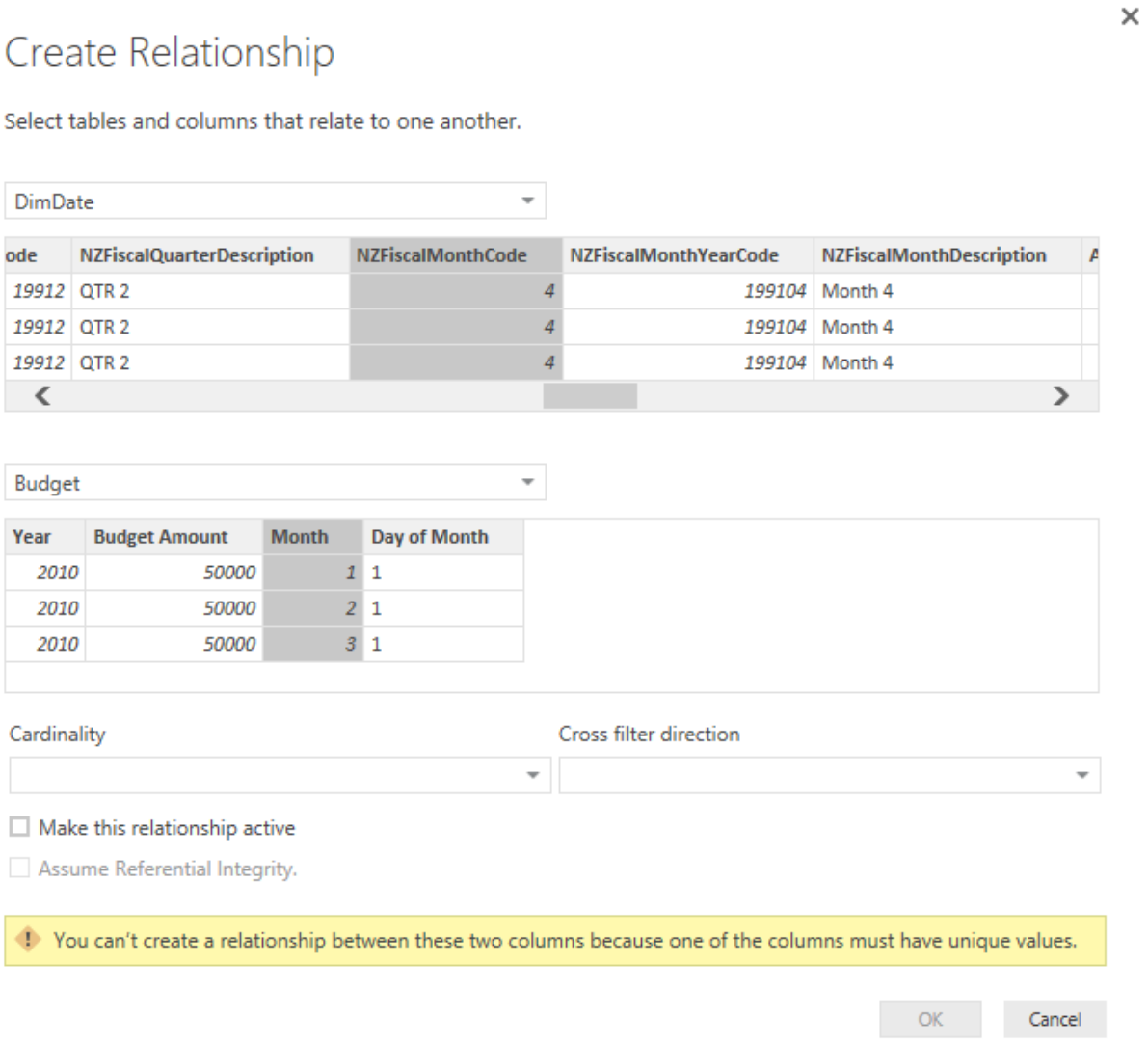
If I want to do date analysis and date based calculations, it is best to create a relationship between the budget table and a date dimension. Here is a view of my date dimension: (Here is an example of creating date dimension with Power Query, and the script for creating date dimension in SQL Server)

erOfYear	ISOWeekYearCode	WeekDay	WeekDayName	NZFiscalYearCode	NZFiscalYearDescription	NZFiscalQuarterCode	NZFiscalQuarterYearCode	NZFiscalQuarterDescription	NZFiscalMonthCode	NZFiscalMonthYearCode	NZFiscalMonthD
27	199027	3	Tuesday	1991	FY 1991	2	19912	QTR 2	4	199104	Month 4
28	199028	3	Tuesday	1991	FY 1991	2	19912	QTR 2	4	199104	Month 4
29	199029	3	Tuesday	1991	FY 1991	2	19912	QTR 2	4	199104	Month 4
30	199030	3	Tuesday	1991	FY 1991	2	19912	QTR 2	4	199104	Month 4
31	199031	3	Tuesday	1991	FY 1991	2	19912	QTR 2	4	199104	Month 4
32	199032	3	Tuesday	1991	FY 1991	2	19912	QTR 2	5	199105	Month 5
33	199033	3	Tuesday	1991	FY 1991	2	19912	QTR 2	5	199105	Month 5
34	199034	3	Tuesday	1991	FY 1991	2	19912	QTR 2	5	199105	Month 5
35	199035	3	Tuesday	1991	FY 1991	2	19912	QTR 2	5	199105	Month 5
27	199127	3	Tuesday	1992	FY 1992	2	19922	QTR 2	4	199204	Month 4
28	199128	3	Tuesday	1992	FY 1992	2	19922	QTR 2	4	199204	Month 4
29	199129	3	Tuesday	1992	FY 1992	2	19922	QTR 2	4	199204	Month 4
30	199130	3	Tuesday	1992	FY 1992	2	19922	QTR 2	4	199204	Month 4
31	199131	3	Tuesday	1992	FY 1992	2	19922	QTR 2	4	199204	Month 4
32	199132	3	Tuesday	1992	FY 1992	2	19922	QTR 2	5	199205	Month 5
33	199133	3	Tuesday	1992	FY 1992	2	19922	QTR 2	5	199205	Month 5
34	199134	3	Tuesday	1992	FY 1992	2	19922	QTR 2	5	199205	Month 5
35	199135	3	Tuesday	1992	FY 1992	2	19922	QTR 2	5	199205	Month 5
28	199228	3	Tuesday	1993	FY 1993	2	19932	QTR 2	4	199304	Month 4

To join these two tables, I can add a day column to the budget table, and then join them based on three columns: fiscal year, fiscal period, and day (day of the month). So here is the budget table with the new day column added and month value a bit polished to remove “Mth” from the month label;

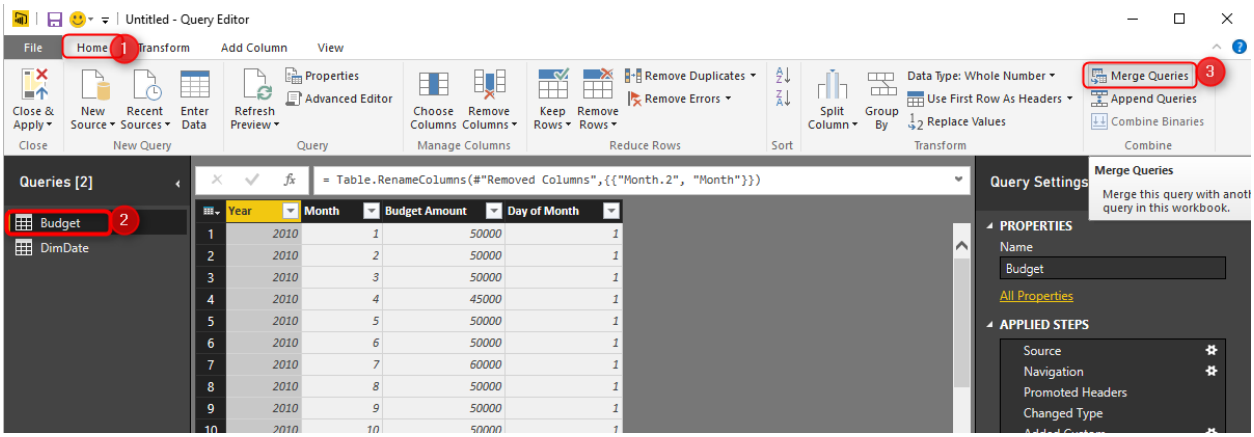
	Year	Month	Budget Amount	Day of Month
1	2010	1	50000	1
2	2010	2	50000	1
3	2010	3	50000	1
4	2010	4	45000	1
5	2010	5	50000	1
6	2010	6	50000	1
7	2010	7	60000	1
8	2010	8	50000	1
9	2010	9	50000	1
10	2010	10	50000	1
11	2010	11	50000	1
12	2010	12	50000	1
13	2011	1	50000	1
14	2011	2	55000	1
15	2011	3	55000	1

Now if I want to create a relationship between the date dimension and budget table based on these three columns I cannot! The create relationship dialog doesn't allow me to select multiple columns, and because with a single column, a Key field won't be identified so the relationship can't be created.

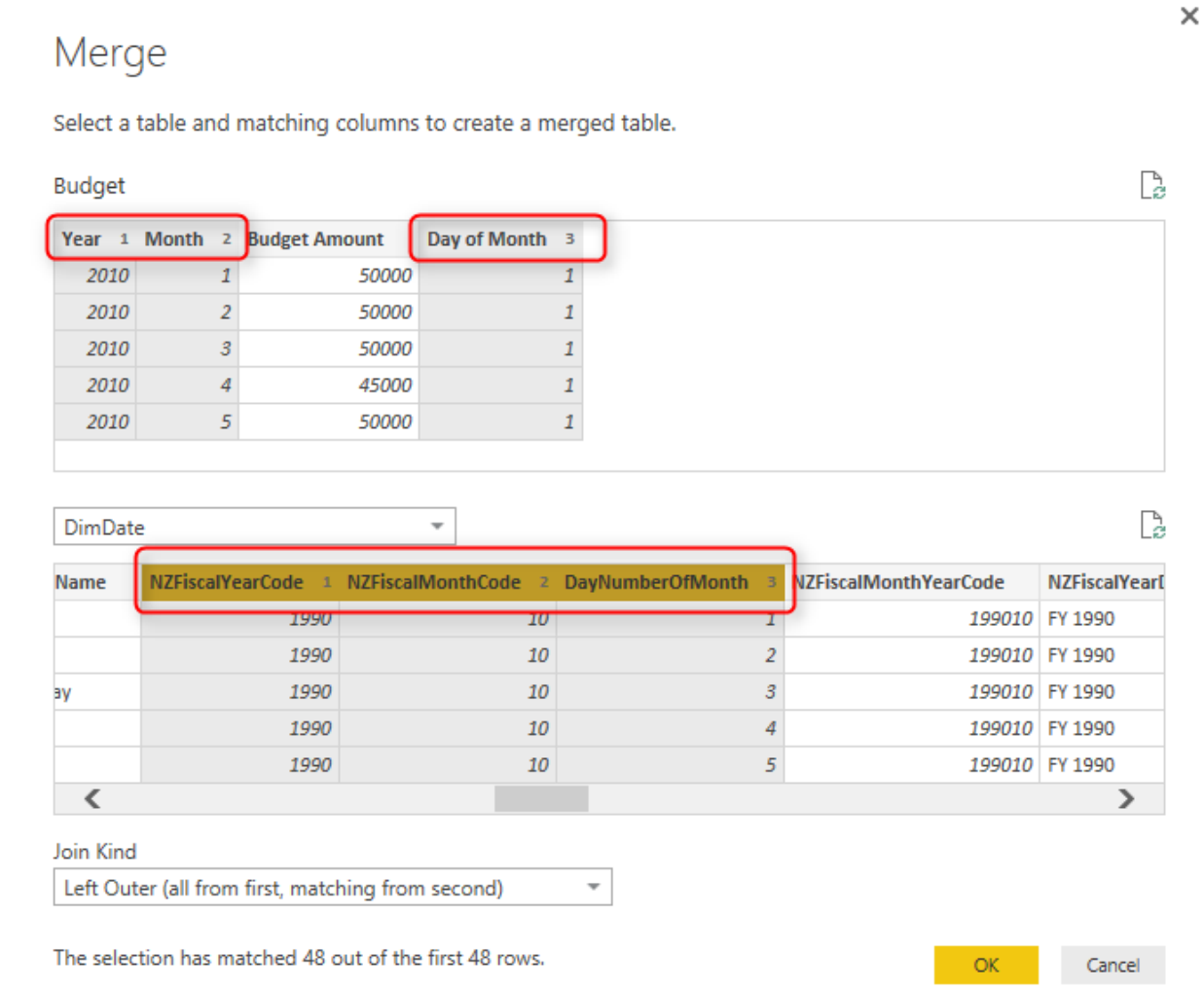


Workaround

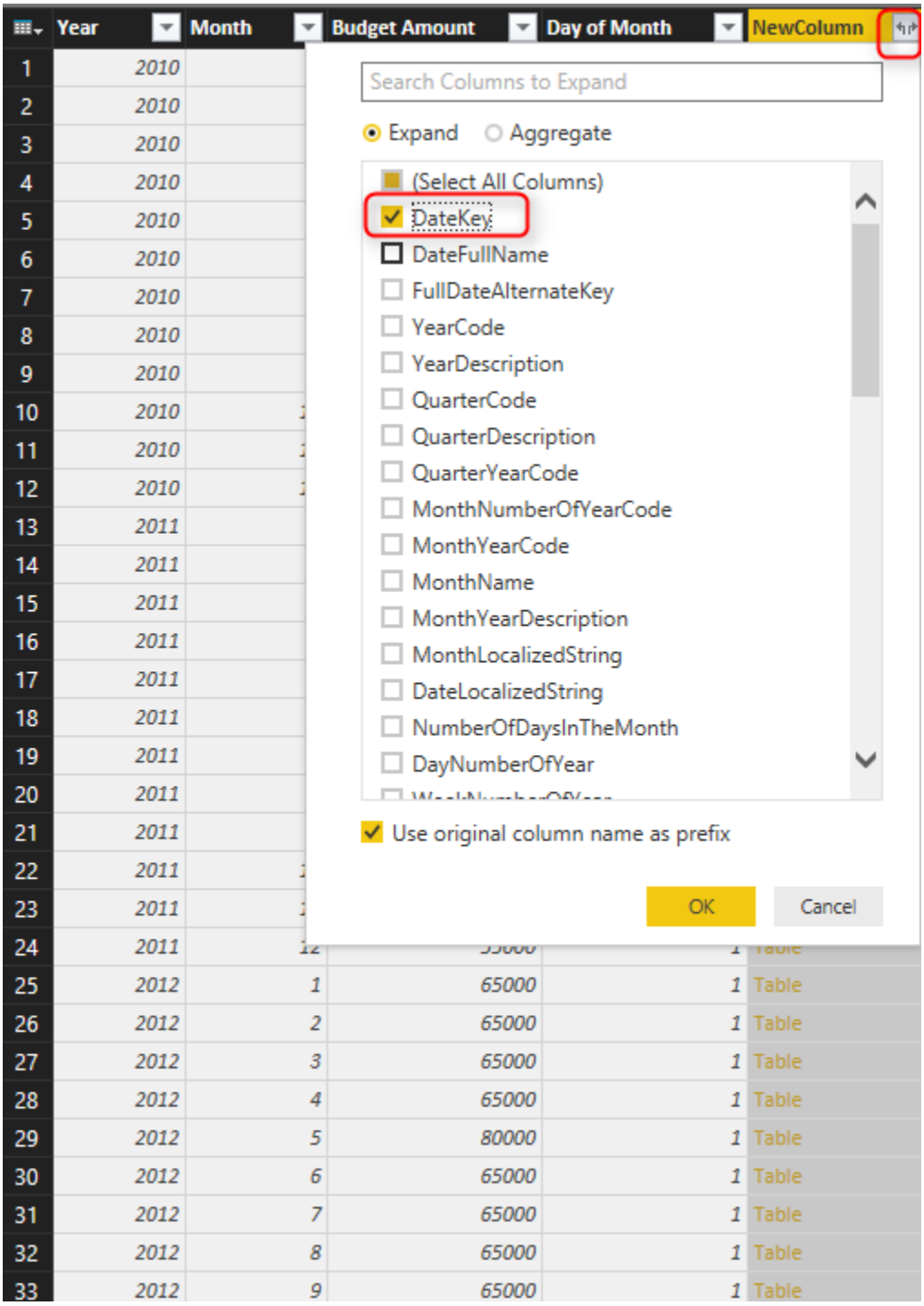
The workaround for this problem is easy. Power BI doesn't allow relationship in model based on multiple columns, but Power Query can join tables with as many as columns we want. So what I can do as a workaround is to join budget table to date dimension in Power Query and fetch the date key. Then I'll use the date key as a single field relationship in Power BI modeling section.
First I open Merge Queries from the Combine section of Home tab;



Here is how I join two tables based on multiple columns: I can hold the CTRL key and select columns one by one (in the right order of joining)



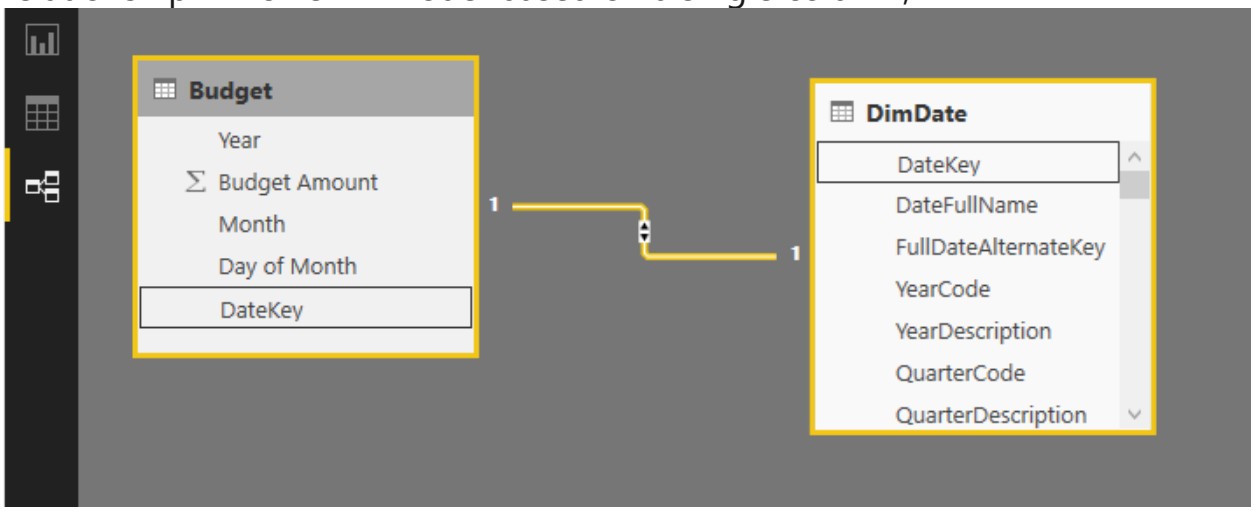
Then I'll have the new table embedded as a result of the join;



So I'll pick the key field from the embedded table;

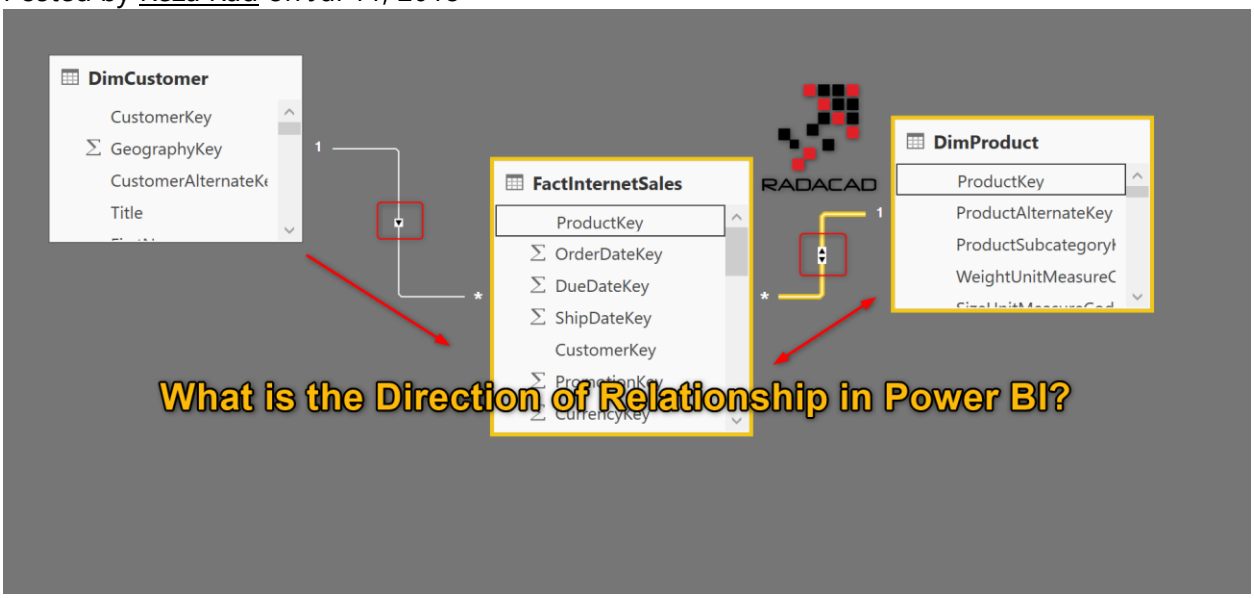
	Year	Month	Budget Amount	Day of Month	DateKey
1	2010	1	50000	1	20090401
2	2010	2	50000	1	20090501
3	2010	3	50000	1	20090601
4	2010	4	45000	1	20090701
5	2010	5	50000	1	20090801
6	2010	6	50000	1	20090901
7	2010	7	60000	1	20091001
8	2010	8	50000	1	20091101
9	2010	9	50000	1	20091201
10	2010	10	50000	1	20100101
11	2010	11	50000	1	20100201
12	2010	12	50000	1	20100301
13	2011	1	50000	1	20100401

And after save and closing the query editor window, I can create the relationship in Power BI model based on a single column;



What is the Direction of Relationship in Power BI?

Posted by Reza Rad on Jul 11, 2018



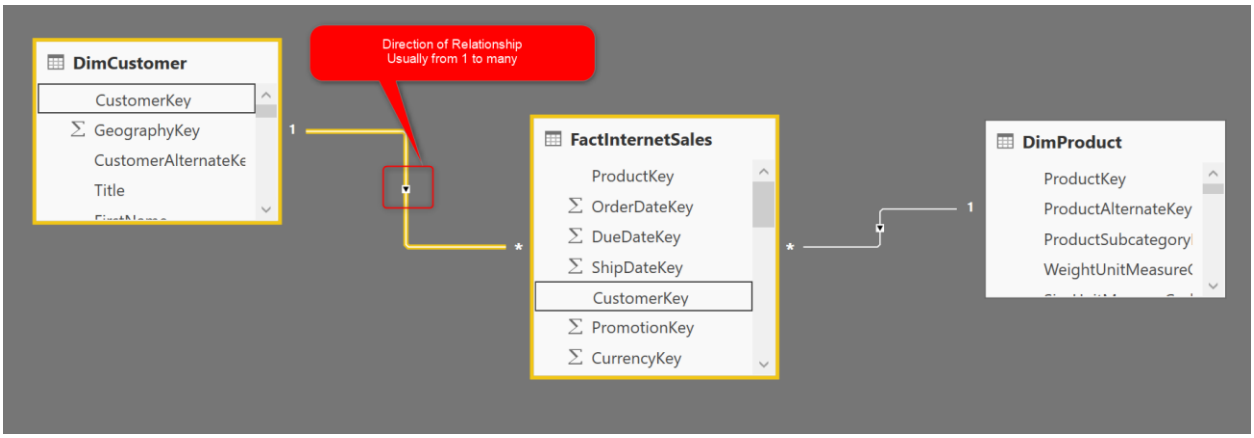
Relationships in Power BI are a bit different from other database management systems. In most of the systems, you have a relationship, and there is no "Direction" for it. In Power BI, however, there is a direction for the relationship. The direction of a relationship plays a critical role in the way that filtering works in Power BI. Understanding the direction of the relationship is an important step towards the modeling in Power BI. In this post, you will learn about what the direction of the relationship is, and what is the difference between both directional or single-directional relationship. We will not talk about how to resolve the relationship issues in this post because it will make this post long post. Later on, I will write about the best practices for covering both directional relationships. To learn more about Power BI, read Power BI book from Rookie to Rock Star.

Sample Dataset

If you want to walk through the example of this post, create a new Power BI Desktop file, and get data from AdventureWorksDW and select DimEmployee as the only table to get data from. Here is how to access the AdventureWorksDW dataset;
Enter Your Email to download the file (required)

Download

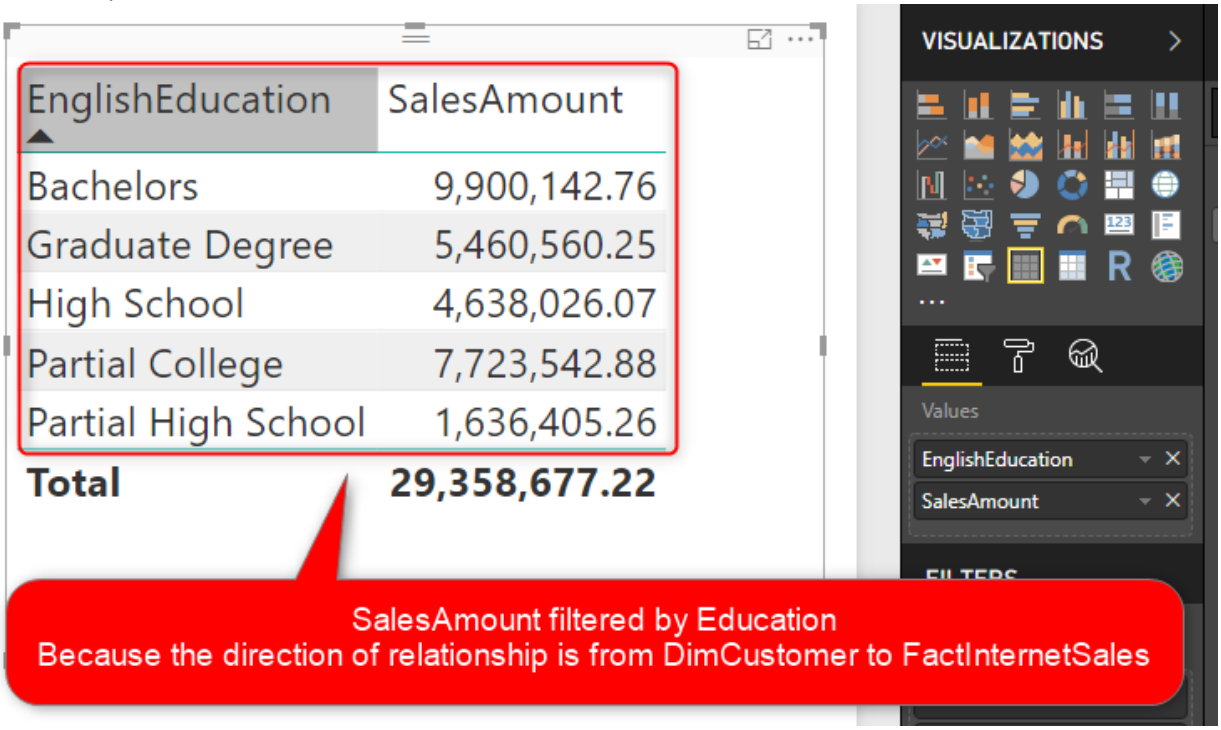
Open a new Power BI Desktop, and Get Data from AdventureWorksDW and select these tables; DimCustomer, DimProduct, FactInternetSales. Load the data into Power BI. After loading data into Power BI, In the relationship tab, you should see all three tables related to each other. You can see the direction of relationship which is usually from one side of the relationship to the "many" side of it.



What is the Meaning of the Direction of the Relationship?

The most important question is what the direction of a relationship means? The answer is; It means Filtering. Whatever direction of the relationship is, that means how Power BI filters the data. In the above screenshot, you can see the direction of the relationship is from DimCustomer to the FactInternetSales. It means any column from DimCustomer can filter the data in the

FactInternetSales. As an example; you can slice and dice the SalesAmount (in the FactInternetSales table) using the EnglishEduction (in the DimCustomer), as below;



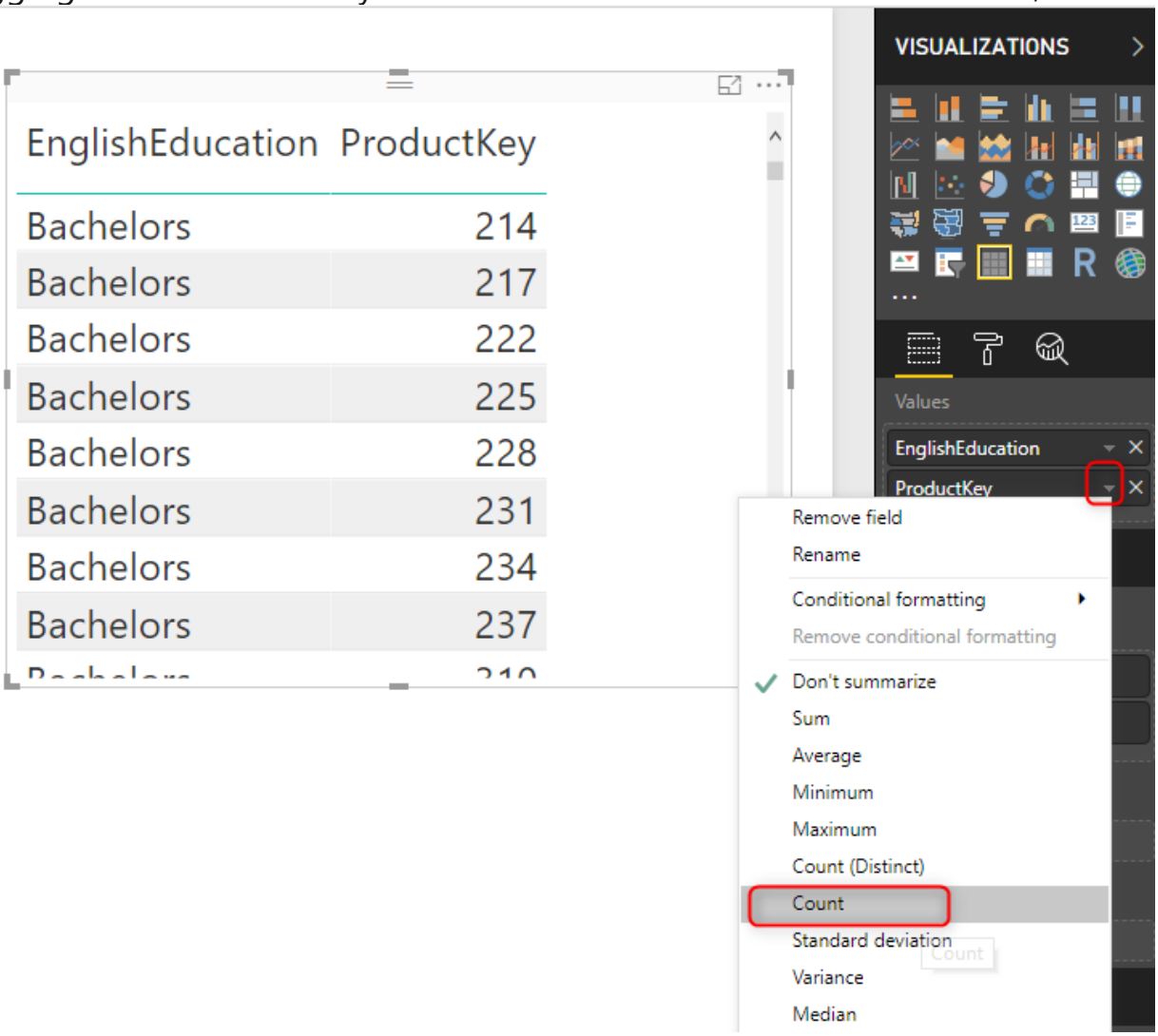
So far, nothing seems strange or weird. You can slice and dice the data of the fact table also using columns from DimProduct because there is a relationship direction from DimProduct to FactInternetSales. The problem (or let's say the strange part) comes when you want to filter in a different direction than what is defined in the relationship. Let's see an example.

The direction of a relationship means the way that filter propagates in Power BI

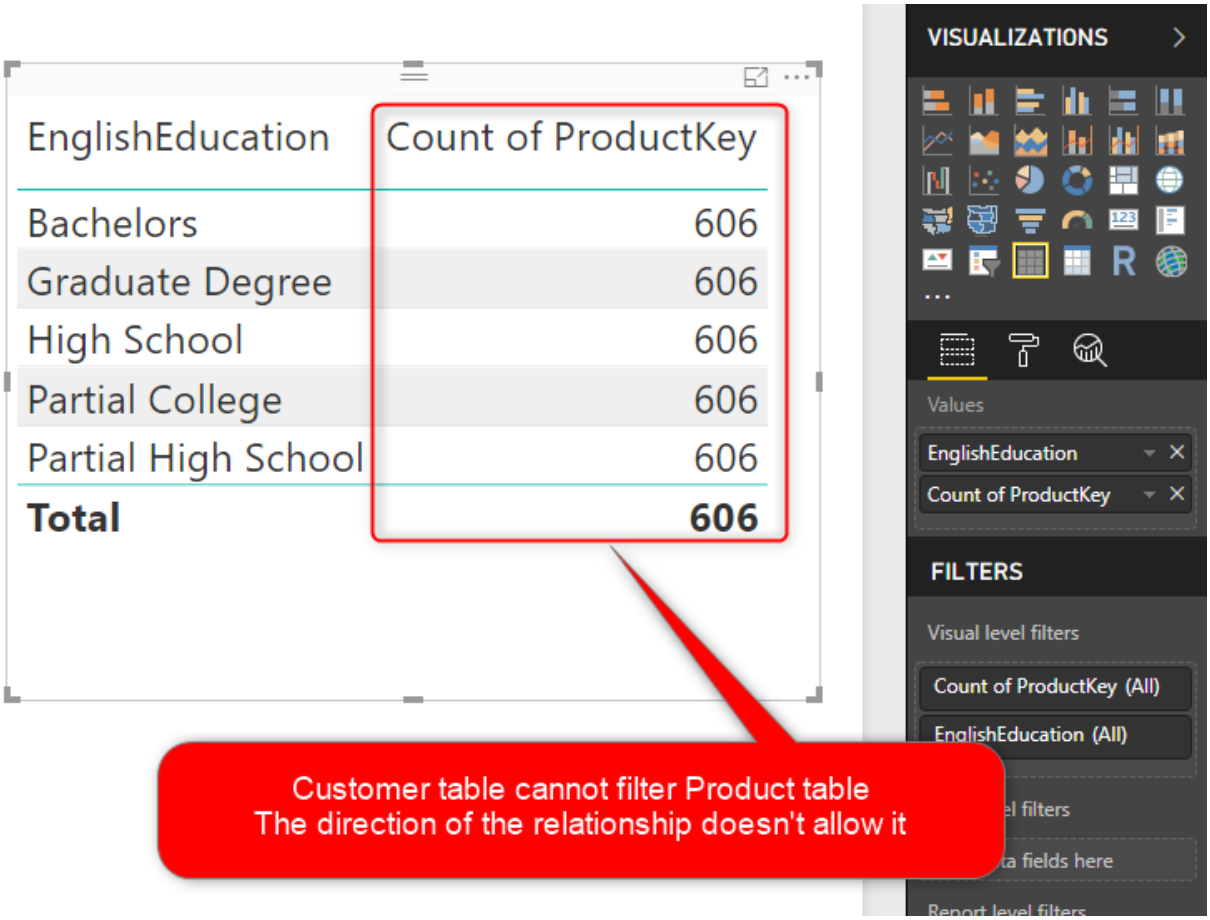
Filtering Based on Different Direction

There are sometimes that you need to filter against the direction of the relationship. Let's look at an example. Let's assume you want to get the number of products been sold in each education category. You can create a table with EnglishEduction (from DimCustomer), and ProductKey (from

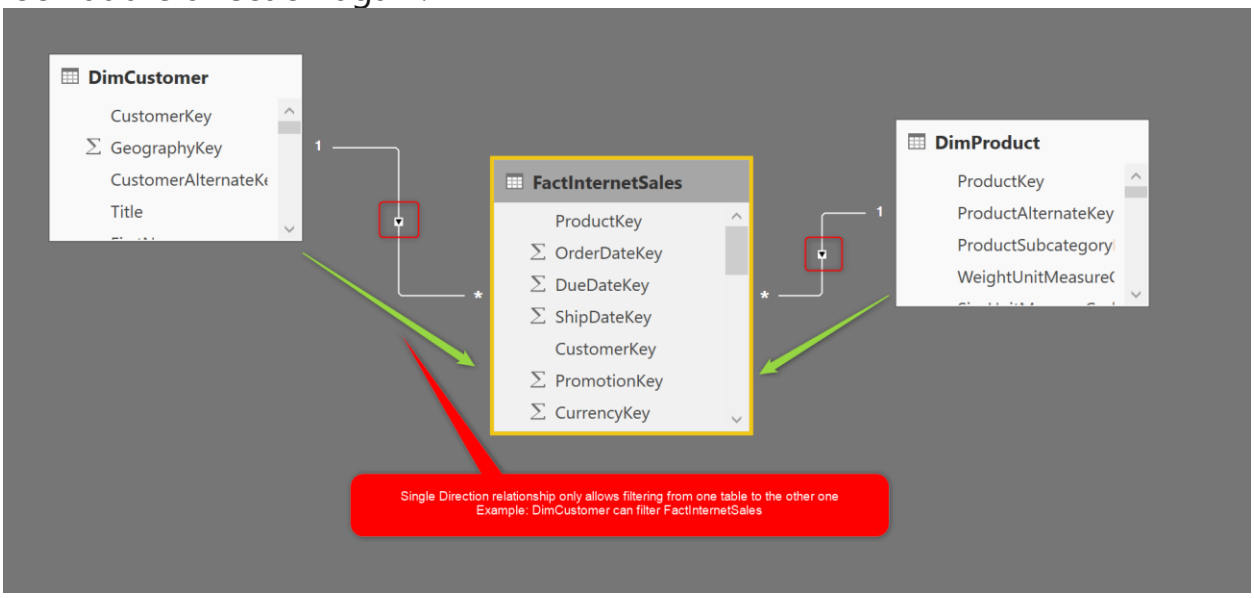
DimProduct) as the first step. Then in the field's list of the visual, change the aggregation of ProductKey to Count as shown in the screenshot below;



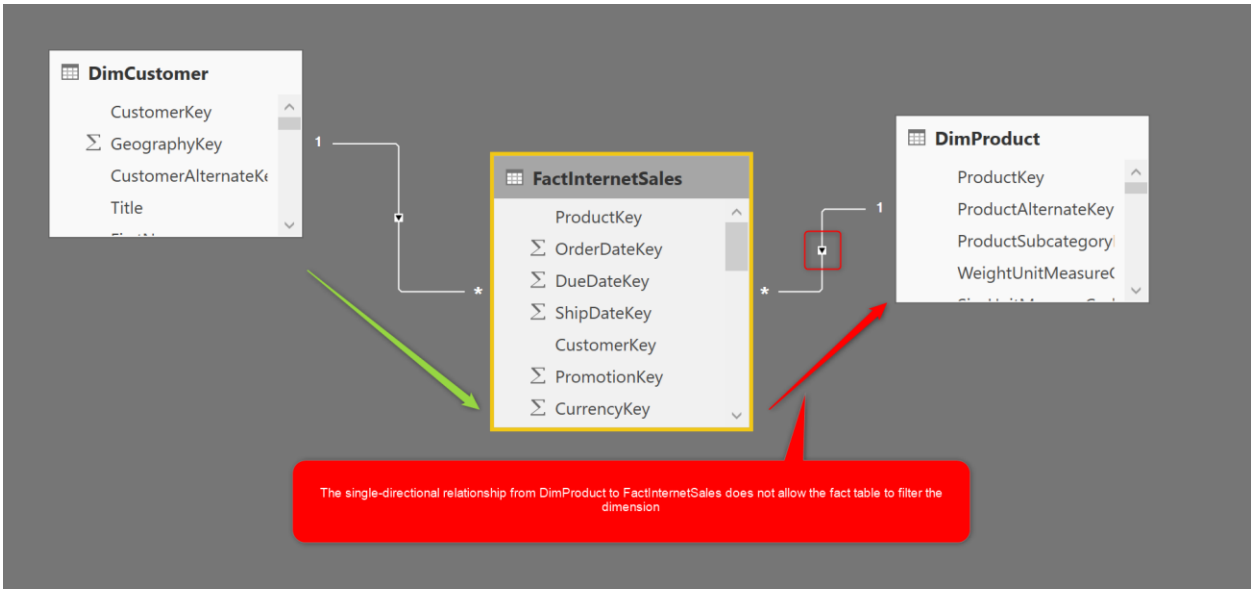
The result would be the count of products for each customer education category. But wait, the result doesn't look correct! It shows 606 for every education category!



As you can see in the above screenshot, the count of ProductKey is 606 for every EnglishEducation. The reason is The DIRECTION of the relationship. Let's look at the direction again.



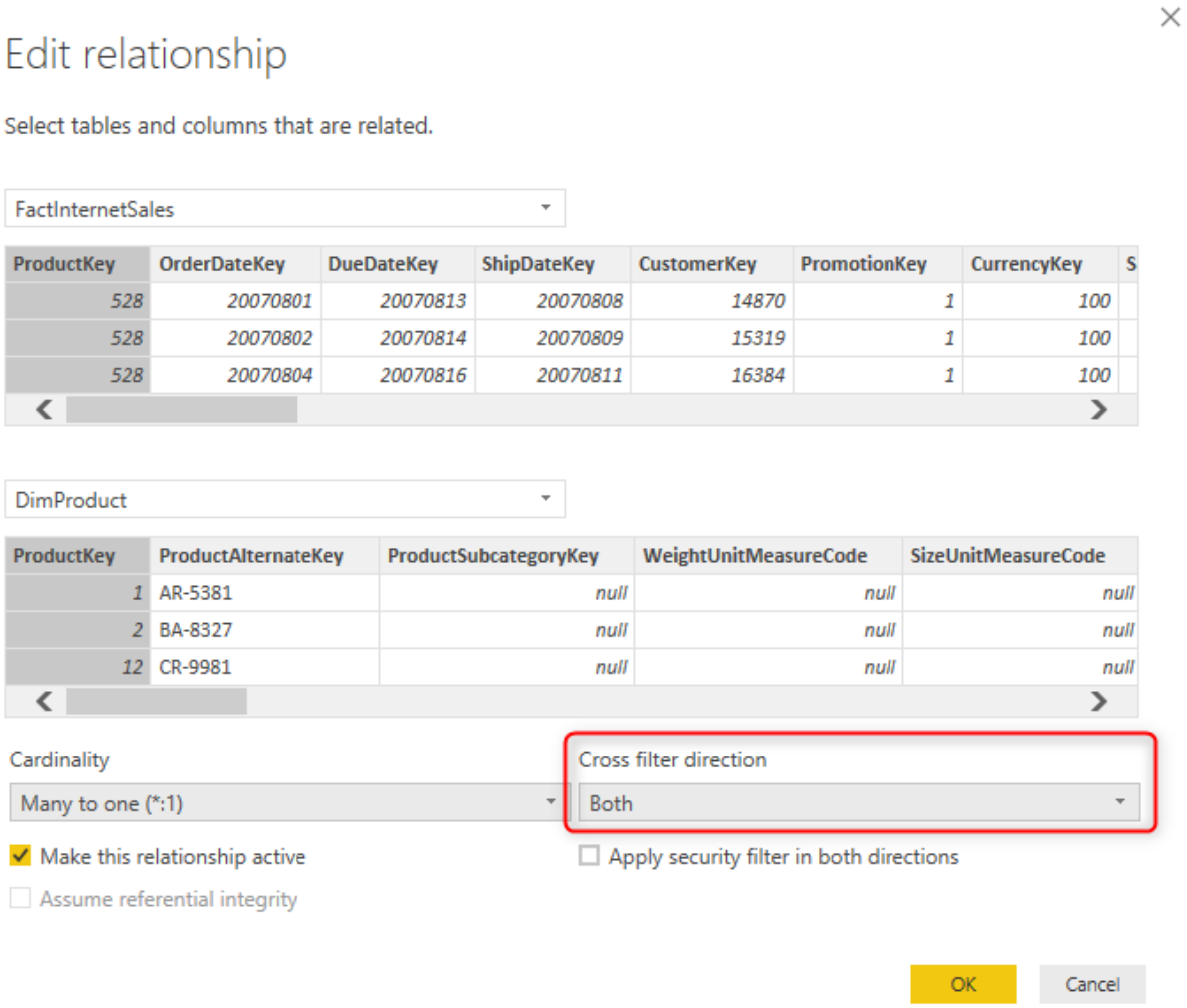
As you can see in the above screenshot, only filtering from DimProduct or DimCustomer to the FactInternetSales is allowed with the current direction of the relationship. However, what we are trying to achieve in this example is a bit different. We want to filter DimProduct based on the selection of Education in DimCustomer.



You do need to have a different direction in the relationship to get it working.

Both-Directional Relationship

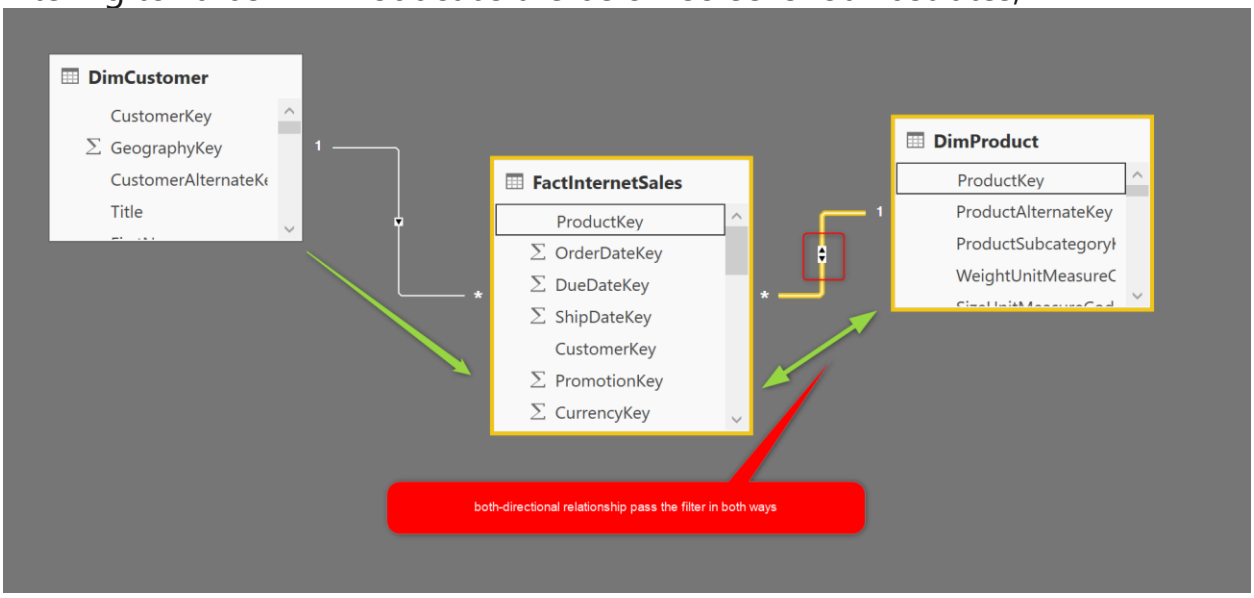
For example above to work, you need to change the direction of relationship to both-directional to get it working. Please note that you SHOULD NOT do this all the time. The both-directional relationship has a significant drawback about performance (we will talk about it later). For now, to see what the both-directional relationship does, double-click on the relationship line between DimProduct and FactInternetSales and make it both directional.



Now you should see the result correctly in the table;

EnglishEducation	Count of ProductKey
Bachelors	158
Graduate Degree	155
High School	157
Partial College	158
Partial High School	153
Total	**606**

The reason that both-directional relationship works here is that it will enable filtering towards DimProduct as the below screenshot illustrates;



So, here you go; now you know what the direction of the relationship is, and what is the difference between both directional and single directional relationship. Before you go; you need to read one important critical note about the both-directional relationship, however!

Be Careful! Performance Considerations for Both-Directional Relationship

After you've done the example above, it seems that both-directional relationship is good! Then you may think; why we should not be using it all the time! If you have been using Power BI Desktop in the early days of the second half of 2015, the default type of direction was both-directional. That time, I got many calls and emails from people that their model is slow! Why you think was that? Because of both-directional relationship!

The both-directional relationship is one of the ways you can kill the performance of your Power BI Model!

Yes, you read it correctly. The both-directional relationship is causing performance issues. Also, you cannot always create both directional relationship, because it will create a loop of filtering sometimes! So what is the solution? The solution of both-directional relationship is not short enough to talk about it in this post. I will point out two methods, and then later on in future posts, I'll explain them in details.

Method 1: Change the Data Model! Design Appropriately

Yes, The right data model does not need many places to be marked as a both-directional relationship. If your model needs the both-directional relationship in the majority of the relationships, then your model is not designed well. I have explained a bit about modeling principles in this post. I will write more about it later. Good modeling can resolve the need for both-directional relationship.

Method 2: Using CrossFilter DAX function ONLY IF the first method does not work

Only and only if you have designed your model properly, and still you cannot get what you need, then you can write a DAX expression using CrossFilter to

get the result you want. Doing it this way is still using both-directional relationship for that calculation. However, the both-directional relationship would be used only for calculating that single measure. All other times, performance should be normal. I will write more about CrossFilter function in DAX in another post.

Summary

The direction of the relationship plays a very important role in modeling in Power BI. The direction of the relationship means the way that filter propagates in Power BI. The single-directional relationship will filter one table based on the other one. Sometimes you need to filter in a different direction, that is when the both-directional relationship comes into play. However, both directional relationship comes with a cost of performance issues. Do not use both-directional relationships blindly. Make sure you have designed your model in the right way first, and if that doesn't work, then try other methods such as CrossFilter DAX functions. I will write later about how to resolve the both-directional issue in a Power BI model.

UseRelationship or Role-Playing Dimension; Dealing with Inactive Relationships in Power BI

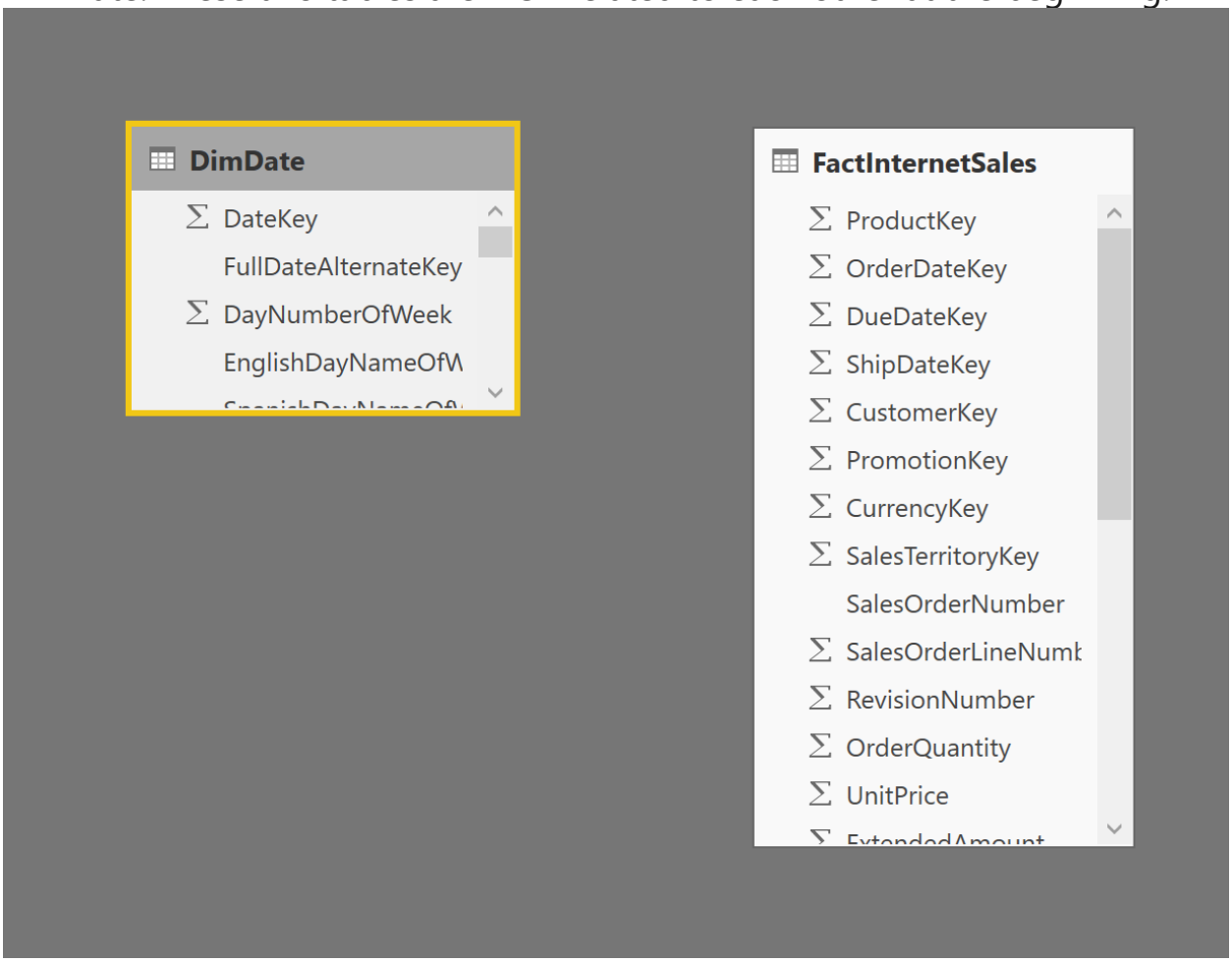
Posted by Reza Rad on Nov 20, 2018

In a Power BI model, relationships are important for passing filters. Filter propagates through relationships. However, sometimes you create the relationship between two tables, and the relationship is a dashed line. In this post, I'll explain to you everything you need to know about a dashed relationship, or as it called Inactive relationship. I will explain two different methods that you can deal with this kind of relationship. So, ready? Let's go through it.

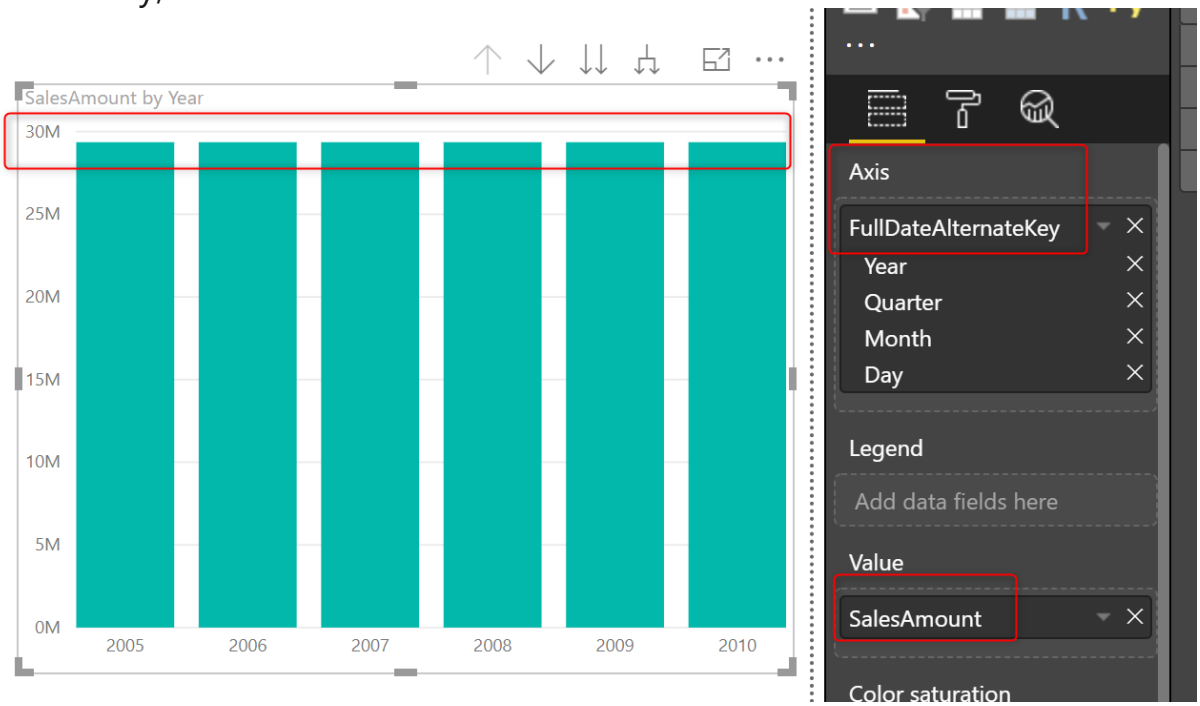
If you want to learn more about Power BI, read Power BI book from Rookie to Rock Star.

Why Relationships in Power BI?

To start this post, the very first topic to discuss is to understand why a relationship in Power BI is important? A relationship in relational database systems is important to link tables to each other, but in Power BI, the relationship also plays another even more important role; Filtering.
To understand how the relationship works, let's check this example:
I have a sample Power BI file getting data from the AdventureWorksDW Excel file example, and I get information from two tables: FactInternetSales, and DimDate. These two tables are NOT related to each other at the beginning.



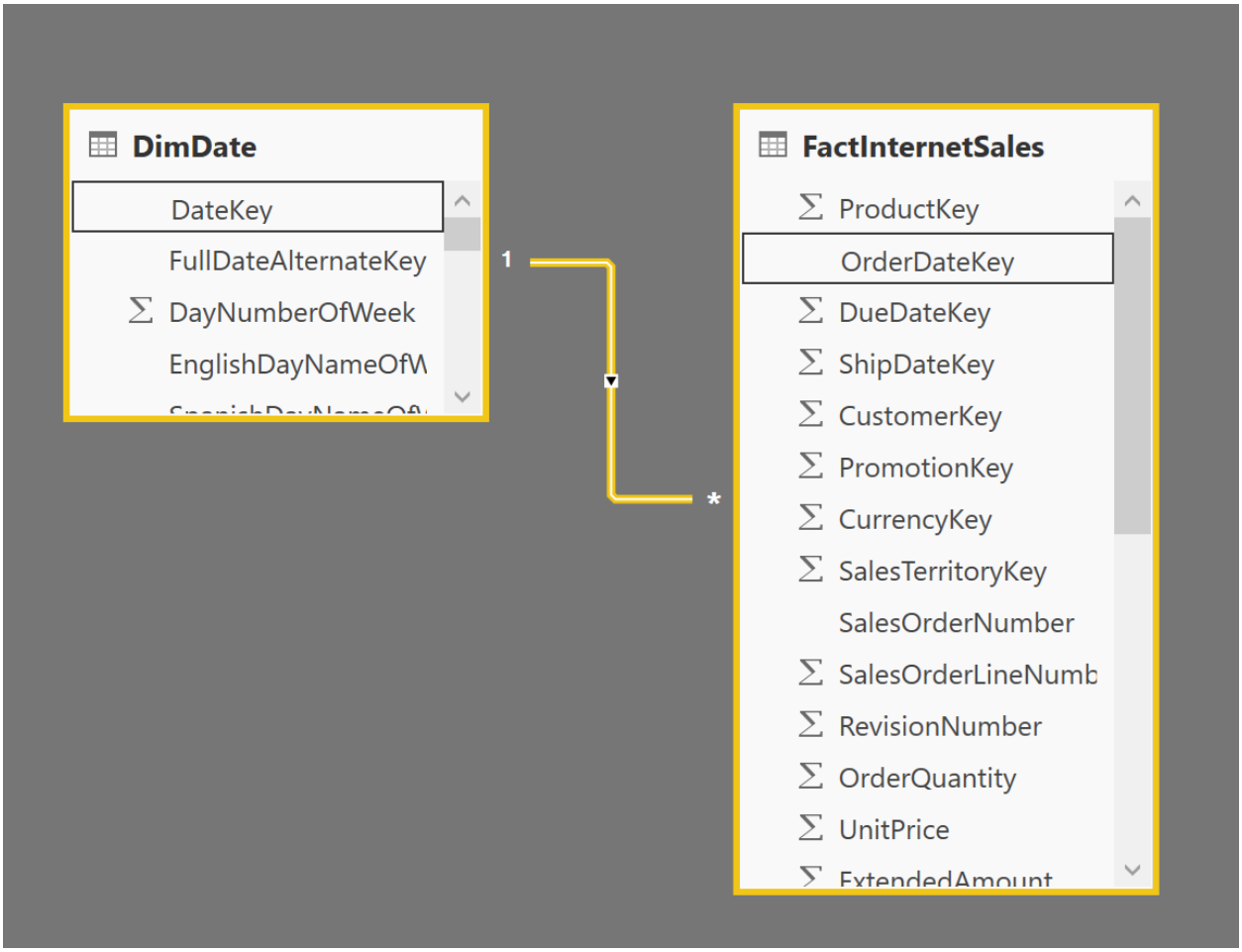
Now, let's create a simple column chart with the SalesAmount from the FactInternetSales table, and the FullDateAlternateKey from the DimDate table. Because the FullDateAlternateKey is a date field, Power BI brings the default hierarchy, and I'll see the visual slicing and dicing data by the highest level of the hierarchy, which is Year.



But wait, it isn't slicing and dicing! It is showing the same SalesAmount for every single year from 2005 to 2010! The value is very close to $30 million which is the total of the sales in my dataset. The fact is that the FullDateAlternateKey field is NOT filtering the FactSalesAmount table.

Relationship Means Filtering

Now, let's create the relationship between these two tables, based on the OrderDateKey in the FactInternetSales table and the DateKey in the DimDate table;



That's it, let's go and check the same visualization again:

As you can see the same visual, this time filters by the date field. Or better to say; DimDate can now FILTER the FactInternetSales table. All of that because of the relationship. Without a relationship, we cannot filter data across tables just by itself, and you may need to do some DAX expressions instead.

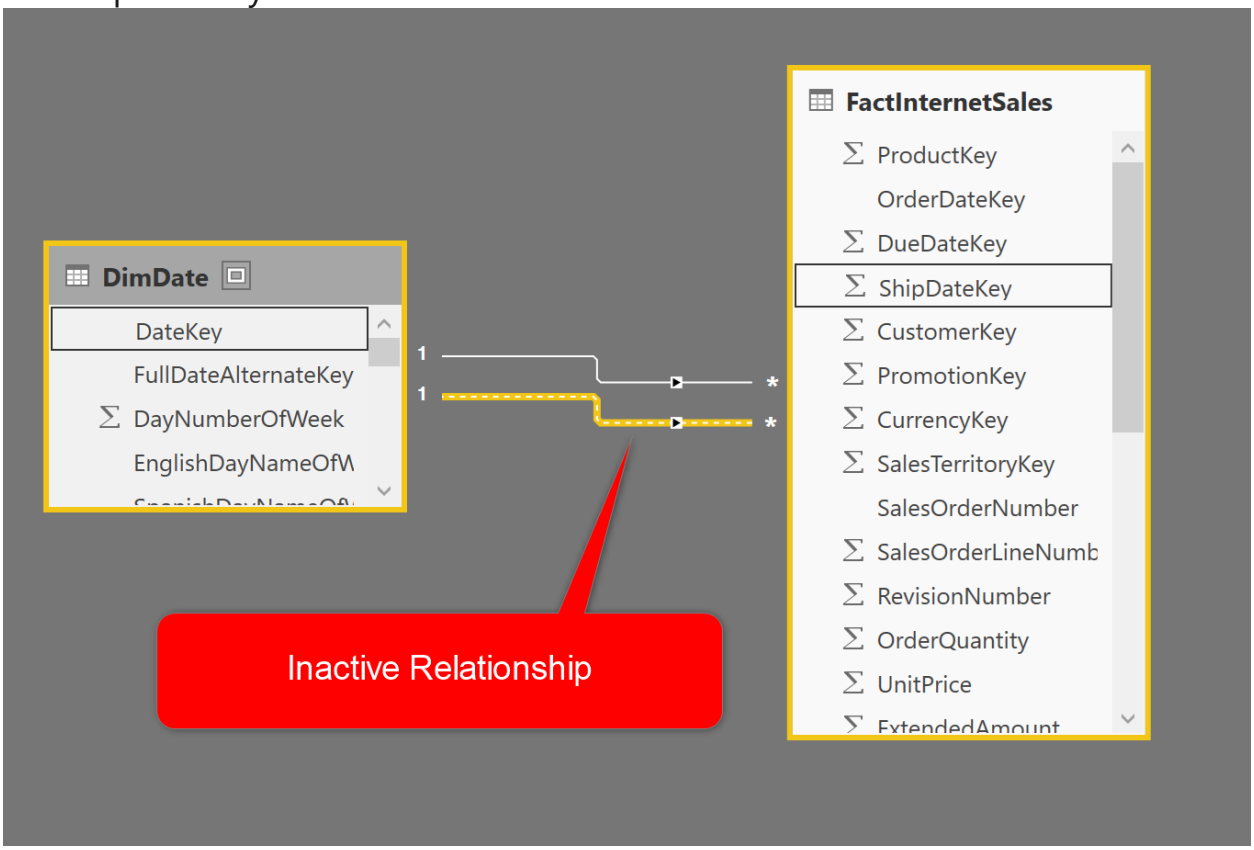
A relationship in Power BI means Filtering, and ability to slice and dice a table by another table.

Now that you now relationships are for Filtering let's check out what the inactive relationship is.

Inactive Relationship

The type of relationship you have seen above is called an active relationship. There is another type of relationship called Inactive. Let's see how an inactive

relationship will be created. In the previous example, we sliced and diced data by the OrderDateKey field, because that was the field connected through the relationship to the DimDate table. Now, let's say we want to slice and dice data by the ShipDateKey. The very simple approach is to create another relationship between the DimDate table and FactInternetSales but this time to the ShipDateKey. Here is the result:

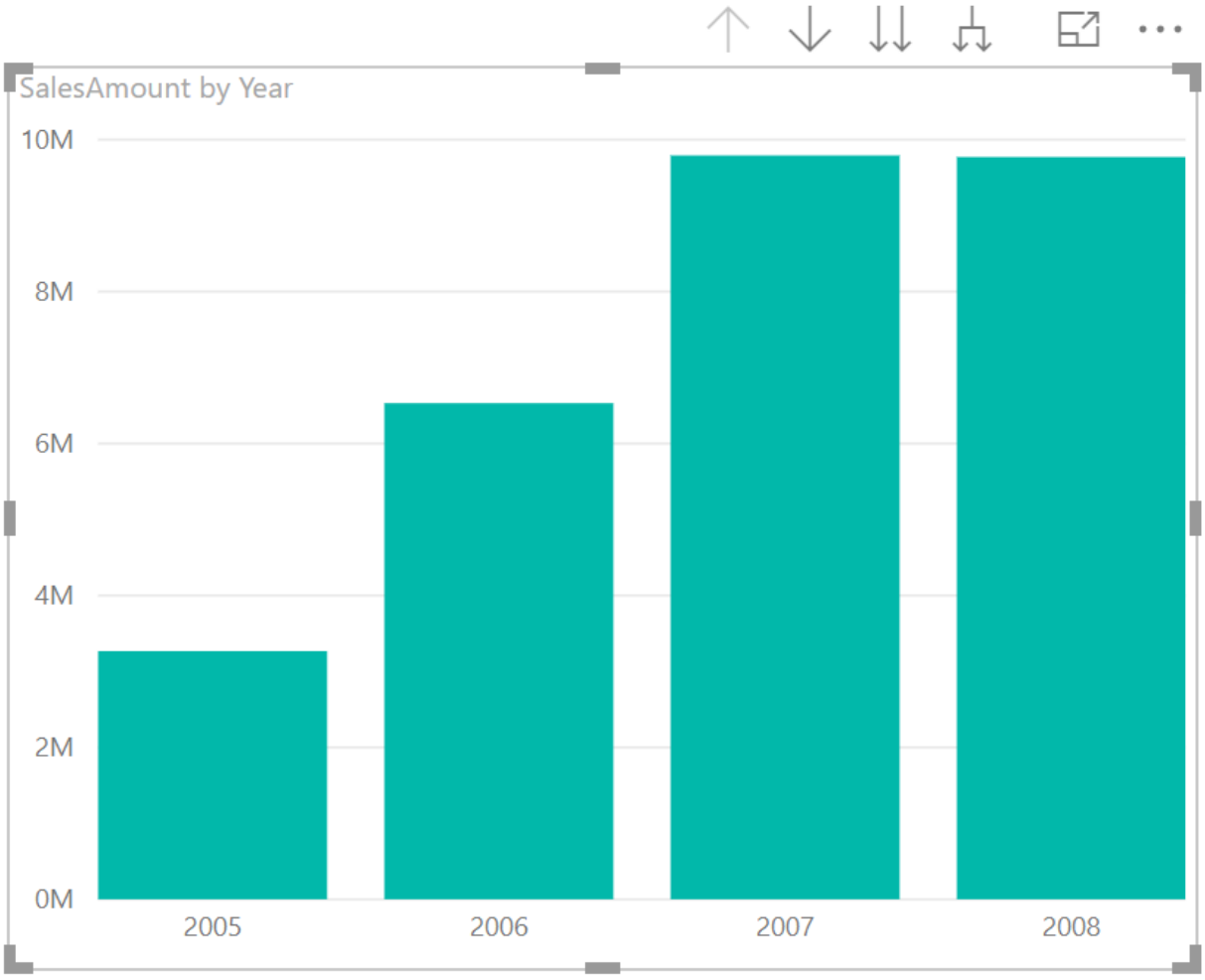


As you can see this new type of relationship is different. It is dashed line, compared to the active, which was a solid line. This is an inactive relationship. You can only have one active relationship between two tables. Any other relationships will become inactive.

You can only have one active relationship between two tables. Any additional relationships will become inactive.

An inactive relationship doesn't pass filtering. It doesn't do anything by itself. I still see many people creating inactive relationships in their model thinking

that just the inactive relationship by itself will do some filtering. It doesn't. If I use the FullDateAlternateKey from the DimDate table to slice and dice the SalesAmount from the FactInternetSales table, which field I'm filtering based on? The field that is related through an Active relationship. Here is a result for that (which is same as what you have seen in the previous example because the inactive relationship doesn't do anything. It is just the active relationship that passes the filter);



Inactive Relationship Doesn't pass the filtering by itself. It needs treatment!

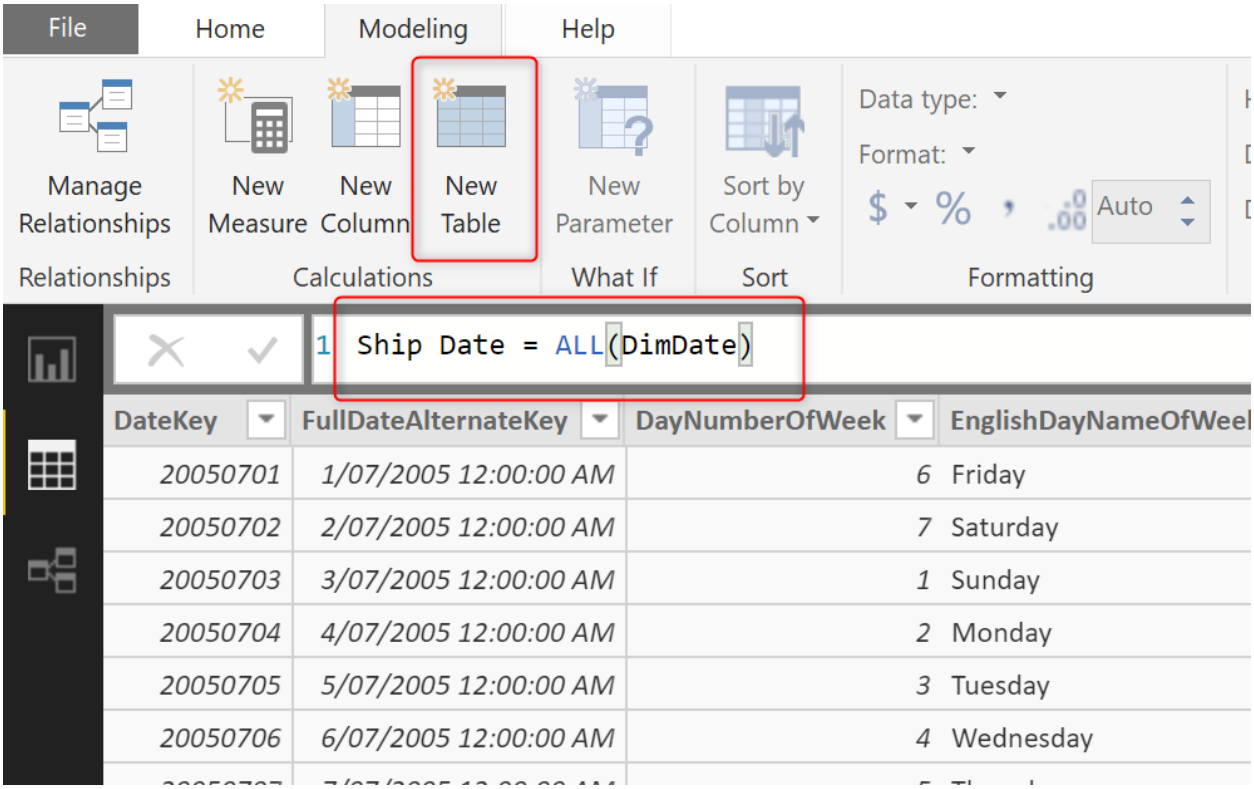
Yes, the inactive relationship needs a special treatment to work. Let's see how this can work. I explain two treatments for an inactive relationship; Role-playing dimension, and UseRelationship method.

Role-playing Dimension

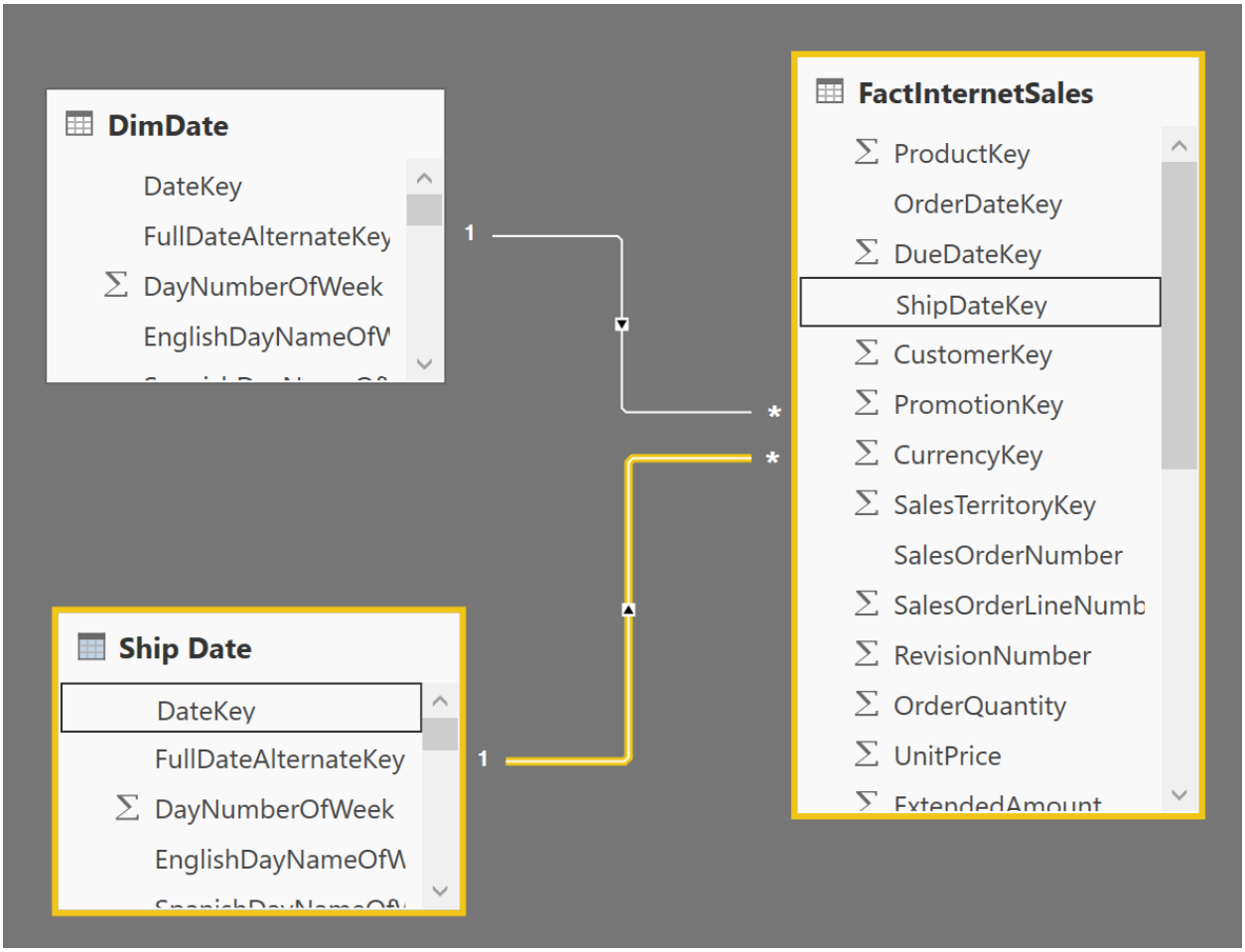
A dimension that plays the role of multiple dimensions, called role-playing dimension in the data warehousing terminologies. In the above example, DimDate is going to play the role of Order Date in some scenarios, and the role of Ship Date in other scenarios, and also sometimes role of Due Date in other times. I already explained a sample usage of Calculated tables in DAX to implement a role-playing dimension, so let's go through it very quickly here too.

One method to deal with the inactive relationship is to remove the cause to create it! If having multiple relationships between two tables is causing the creation of inactive relationship, one way to avoid it seems to be creating multiple instances of the same table, and then you would need only one relationship not more than that.

Let's create a copy of the DimDate, One way to create the copy is to use a Calculated Table with ALL DAX function in it;



ALL is a function that gives you the entire table. In this case, we are creating a copy of the DimDate table, and calling it ShipDate. Now you can create a normal active relationship between ShipDate and the FactInternetSales table (I have removed the inactive relationship from the previous section);



If you like to learn about the benefits of calculated tables, I recommend reading my post about calculated tables here.

And now, as a result, you have slice and dice by the ShipDate table as well as the Order Date (or let's say DimDate table);

Role-playing dimension is one of the ways that you can handle an inactive relationship, but be careful of memory consumption!

Copy only small tables

Role-playing dimension method is copying the table, and you will have double up memory consumption. The extra memory consumption can be overlooked if the table is small. A Date table is a small table. For every year, it is 365 rows, and for 20 years, it will be around 7,000 rows. It is very small compared to a fact table with millions of rows. This solution is good for small tables. But don't use this method for big tables. If you have a dimension table with 5 million rows and 30 columns, then role-playing dimension method means consumption of the same amount of space twice or three times or more.

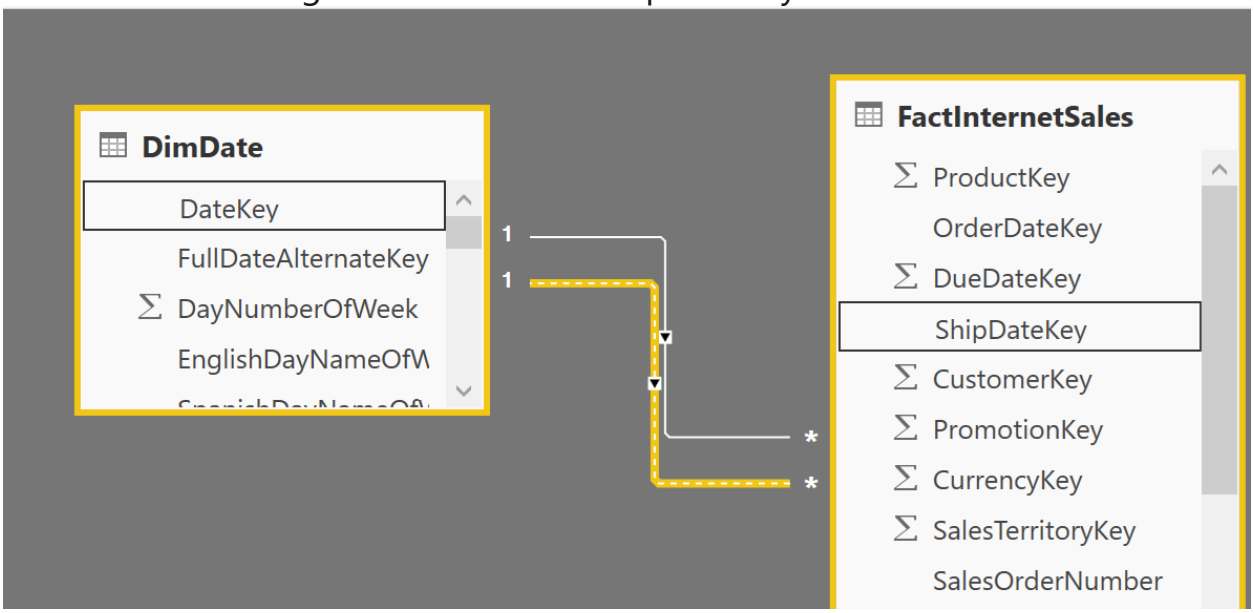
Avoid role-playing dimension if you have large dimension. This method is only good for small tables.

UseRelationship Function in DAX

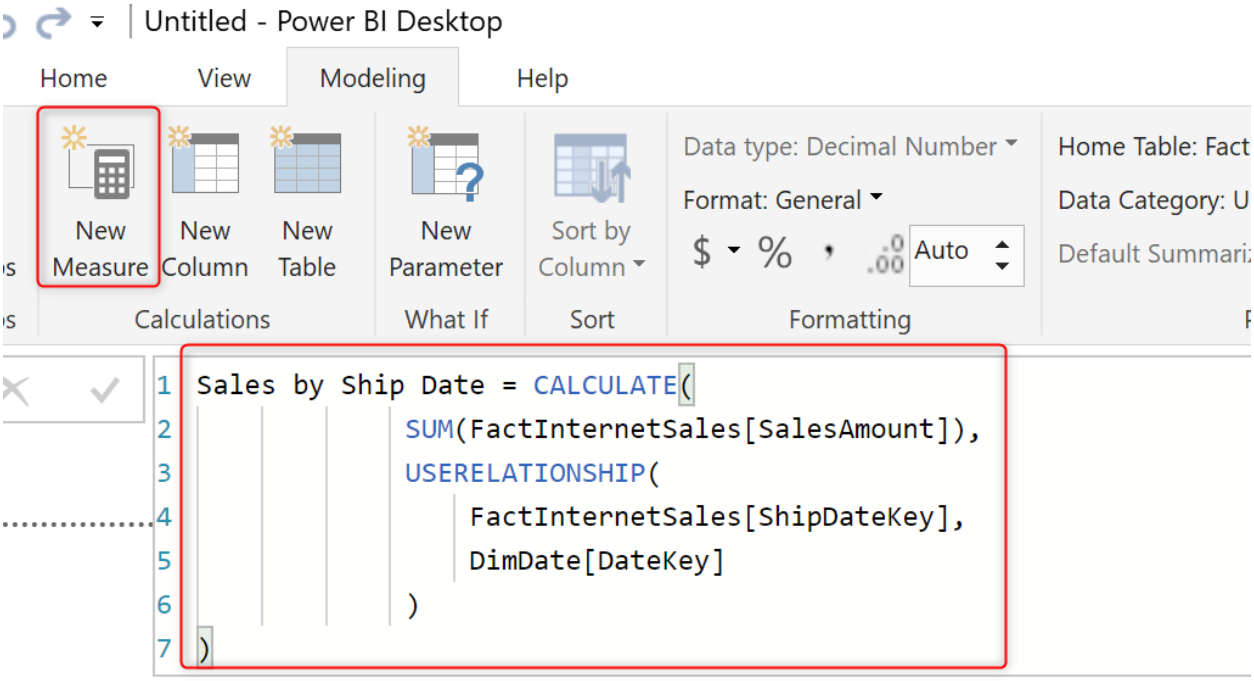
Another method to handle inactive relationship is to use a function in DAX called UseRelationship. This DAX function is saying to Power BI that for this

expression, use this relationship, even if it is inactive. Let's see how this function works.

If we continue the same example of slicing and dicing by Ship Date and assume that there is no Ship Date calculated table created, then we can do it this way; Create the inactive relationship between DimDate and FactInternetSales again based on the ShipDateKey.



Now, let's create a Measure in Power BI with below expression:



1 Sales by Ship Date = CALCULATE(
2 SUM(FactInternetSales[SalesAmount]),
3 USERELATIONSHIP(
4 FactInternetSales[ShipDateKey],
5 DimDate[DateKey]
6)
7)

This measure calculates the sum of sales by ship date. The whole secret is the usage of the UseRelationship function. This is a really simple function to use; you need to provide two input columns to it, the two columns that are two sides of the relationship. Their order is not important.

UseRelationship (<column 1>, <column 2>)

The important tip to consider is that you HAVE to have an existing inactive relationship for this function to work. Otherwise you get the error below:

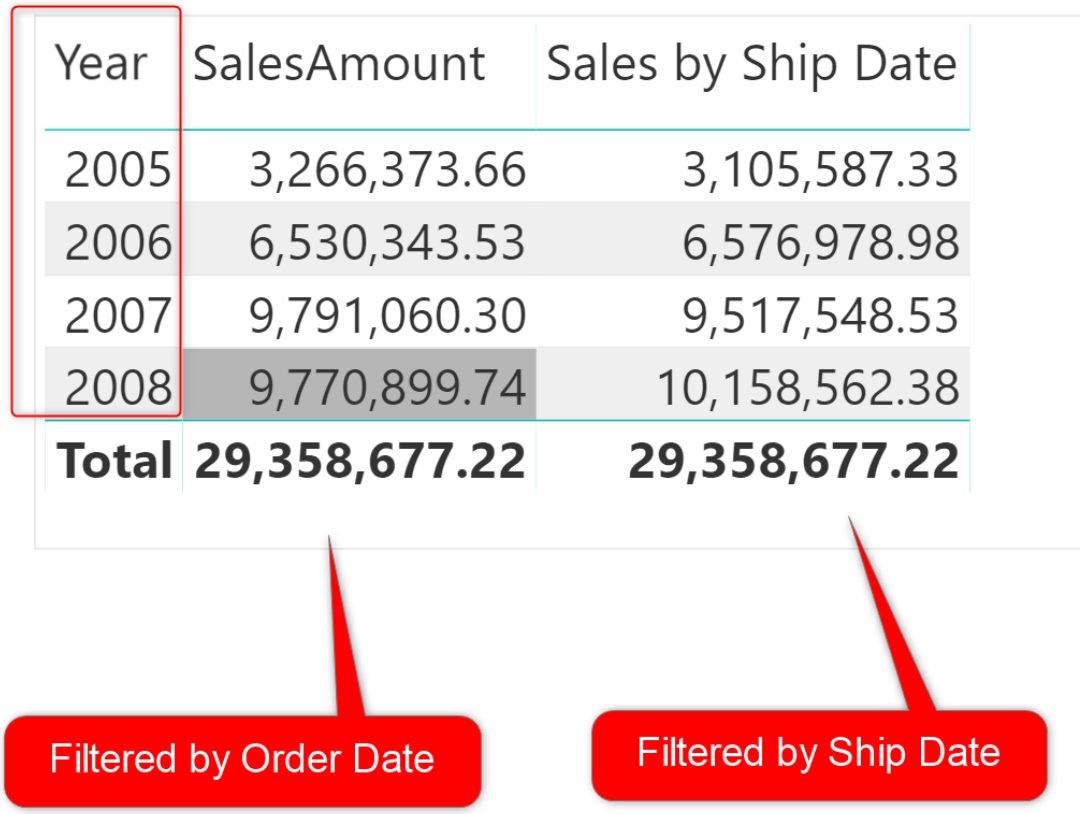
```
1  Sales by Ship Date = CALCULATE(
2                  SUM(FactInternetSales[SalesAmount]),
3                  USERELATIONSHIP(
4                      FactInternetSales[ShipDateKey],
5                      DimDate[DateKey]
6                  )
7  )
```

USERELATIONSHIP function can only use the two columns references participating in relationship.

The inactive relationship must exist otherwise the UseRelationship doesn't work.

One table filters the other table based on multiple fields

The main benefit of using this method is that you can now have the DimDate table to filter the fact table based on both ShipDateKey and OrderDateKey at the same time, as illustrated below:



As you can see in the above screenshot, one date table filters fact table based on multiple fields. One is based on OrderDateKey which is an active relationship, and the other one is based on ShipDateKey through the usage of the UseRelationship method in the measure.

This method doesn't consume extra memory. However, you do need to create a measure for every single calculation with the UseRelationship function.

Summary

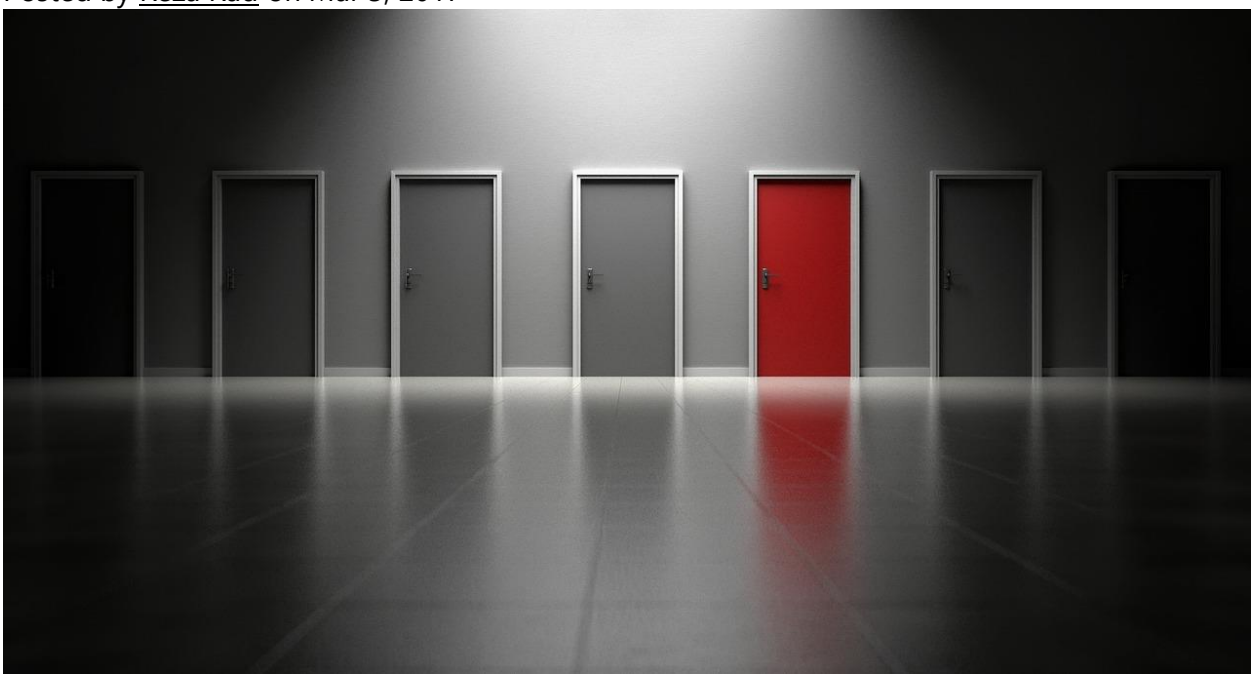
In this post, you learned about inactive relationships, and how to handle them through two methods; Role-playing dimension, and UseRelationship function in DAX. Role-playing dimension method is good for smaller tables where the

extra memory consumption is not the issue. UseRelationship method, on the other hand, can be a good substitute when the tables are bigger. There are other benefits such as getting one table filtering based on multiple fields at the same time as you've seen. Which of these methods or any other methods do you use? Please share it through the comments below, or if you have any questions, please don't hesitate to ask.

Part III: DAX and Calculations

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 the 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:

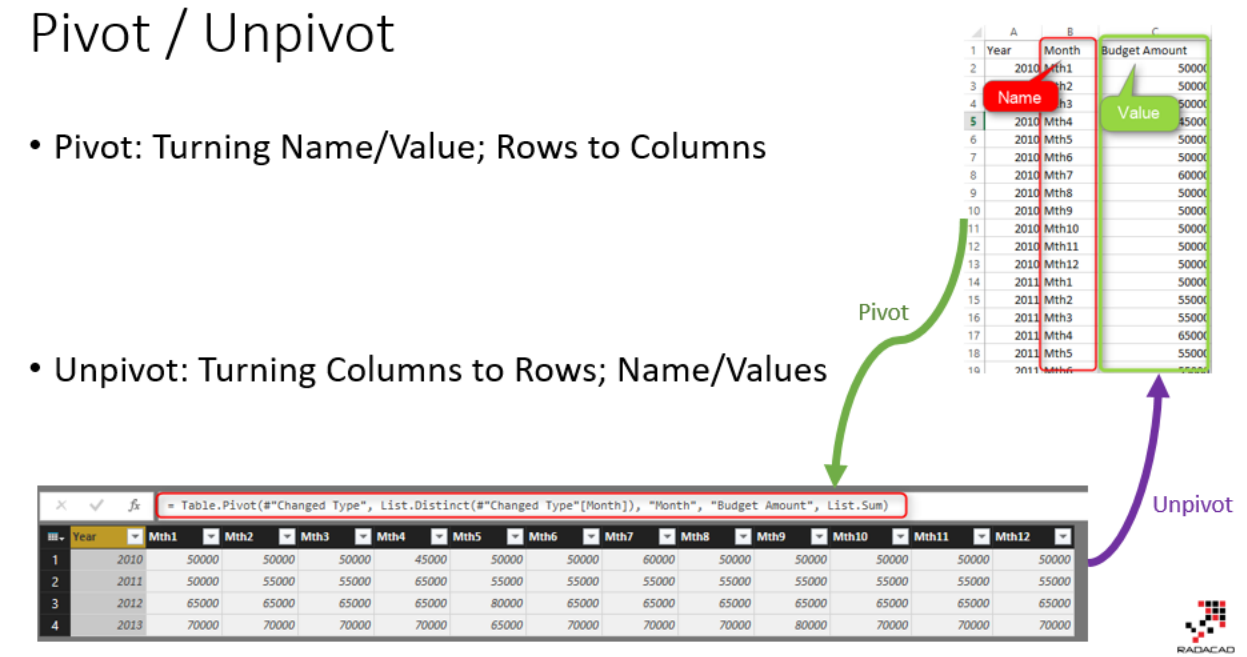
```
Sales Rolling 12 Months =
CALCULATE(
    SUM(FactInternetSales[SalesAmount]),
    DATESBETWEEN
          (DimDate[FullDateAlternateKey],
              NEXTDAY(SAMEPERIODLASTYEAR(LASTDATE(DimDate[FullDateAlternateKey]))),
              LASTDATE(DimDate[FullDateAlternateKey])
           ),
    ALL(DimDate)
    )
```

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.

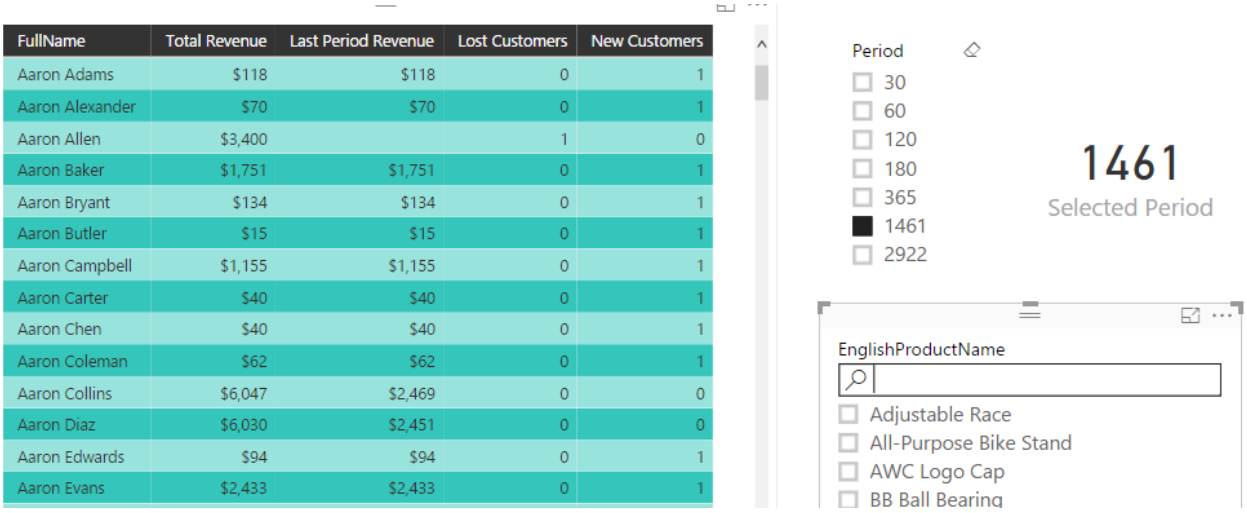
Example Usage of M

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;



Example Usage of DAX?

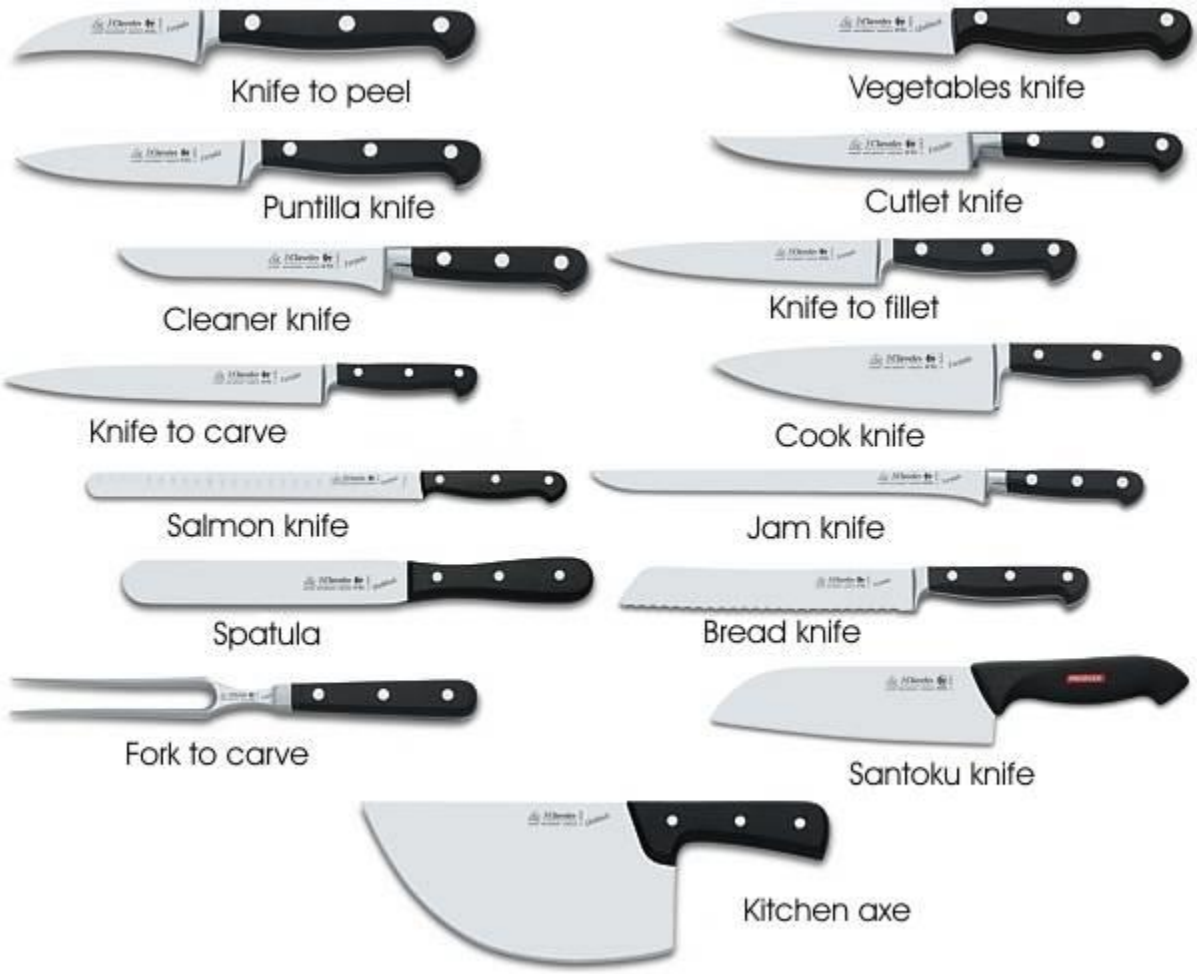
DAX can be used for many calculations for analyzing data. For example, calculating Year To Date, Calculating Rolling 12 Months Average, or anything like that. Here is an example which based on selection criteria in the report and few simple DAX expressions we can do a customer retention case with DAX;



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

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 the 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 combination, etc. However, if you use a DAX calculationn 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.

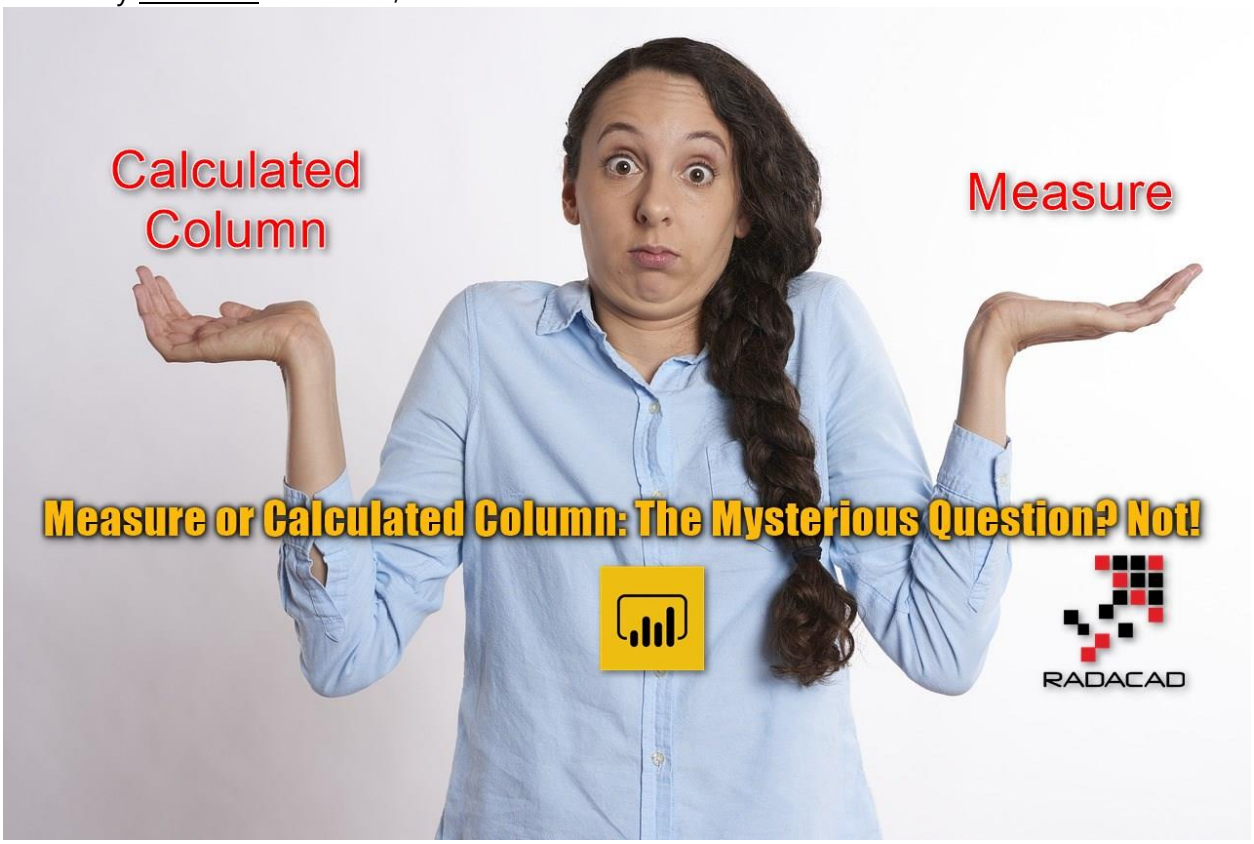
As a Power BI Developer Which Language Is Important to Learn?

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.

Measure vs. Calculated Column: The Mysterious Question? Not!

Posted by Reza Rad on Oct 21, 2017



Despite all articles, blog posts, and videos on the topic of DAX Measures and Calculated columns, I still hear here and there that people ask what the difference between Measure and Calculated Column is? What situation should we use each of these? And on the other hand, what is the difference of creating column here, or in Power Query? I have written previously about "M or DAX, that is the question" which explains situations that you need to use Power Query or DAX. In this post, I'm going to explain what is the difference between DAX Calculated Column and Measure. If you want to learn more about Power BI, read Power BI book from Rookie to Rock Star.

Read this If You have any of Questions Below

This blog post is written for you if you want to understand the difference between Calculated Column and Measure in DAX, and have any of below questions;

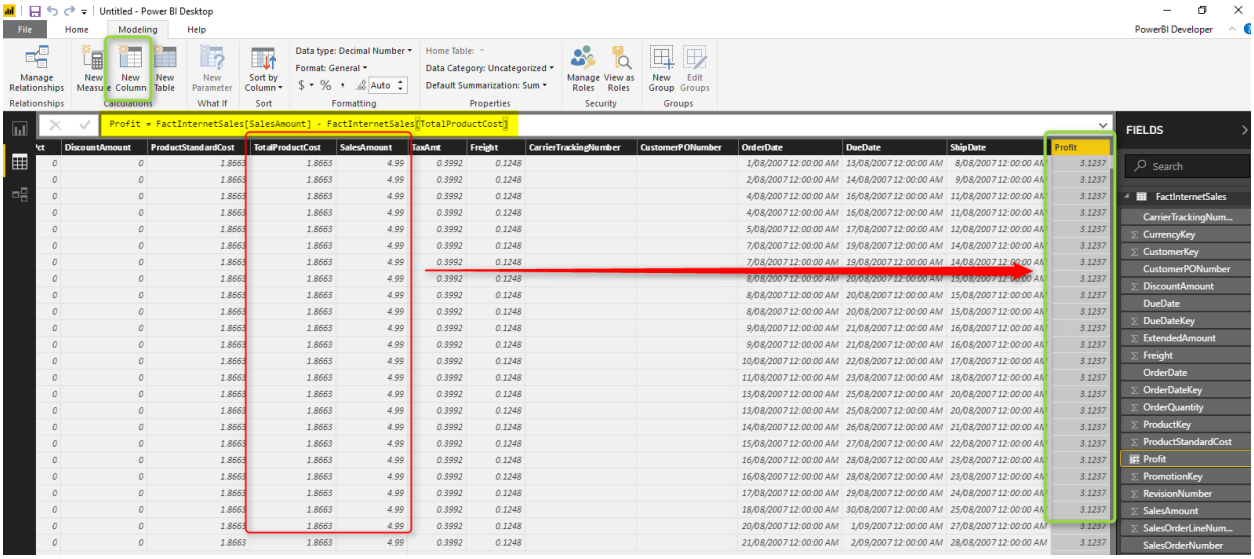
- What is Calculated Column?
- What is Measure?
- When should I write calculated column or measure?
- What is their difference in Performance?
- What are operations that I cannot do with these?
- And many other questions about the difference between these two types of calculations in DAX.

What is Calculated Column?

To understand the difference between these two types of calculation, it is necessary to understand how these are working one by one. Calculated Column is a column like any other columns, created in the table. However, the result of a calculated column is comping from calculating an expression (DAX). Usually, calculated column leverages a DAX expression that applies to every row in the dataset, and the result of that will be stored in the new column.

Example: Profit as a calculated column

Consider a table that we have sales and costs information in it. Calculating Profit in such table would be simply deducting costs from sales for every row. So this basically would be a calculated column.

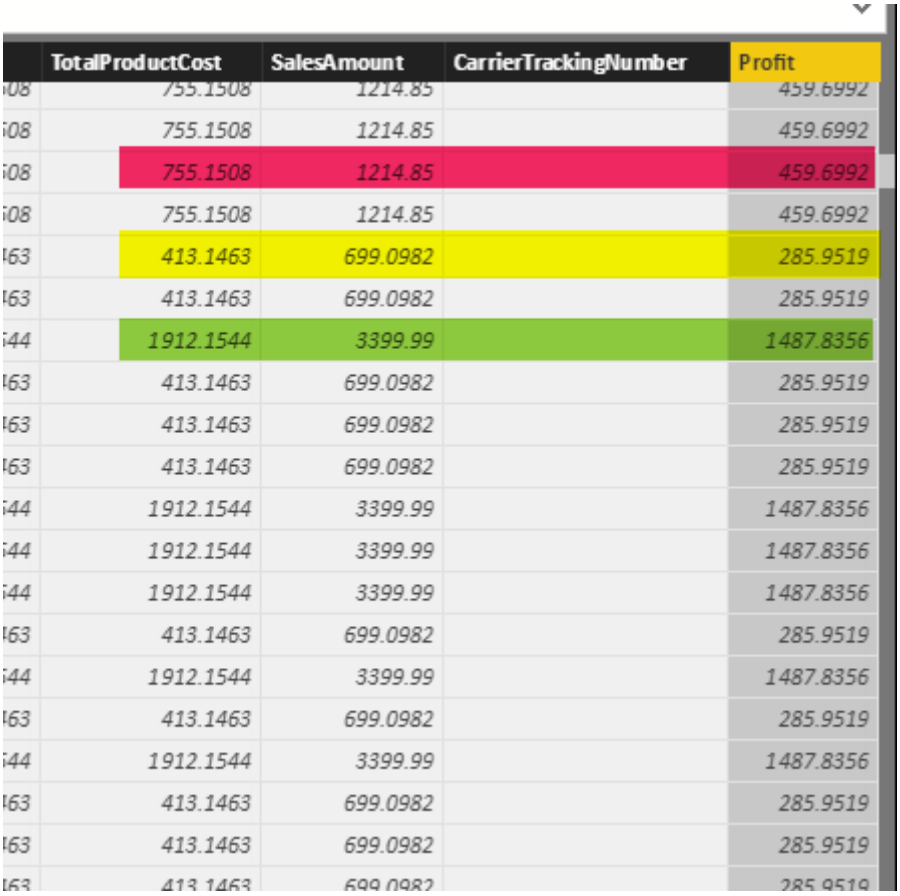


Expression:

```
1 Profit = FactInternetSales[SalesAmount] -
  FactInternetSales[TotalProductCost]
```

Row by Row Calculation: Row Context

One of the very important concepts about the calculation that you apply in Calculated Column (In the majority of the cases, not always); is that the calculation in one row at a time, or in other words; row by row calculation. In below table; you can see the calculation result for every row stored into the new column;

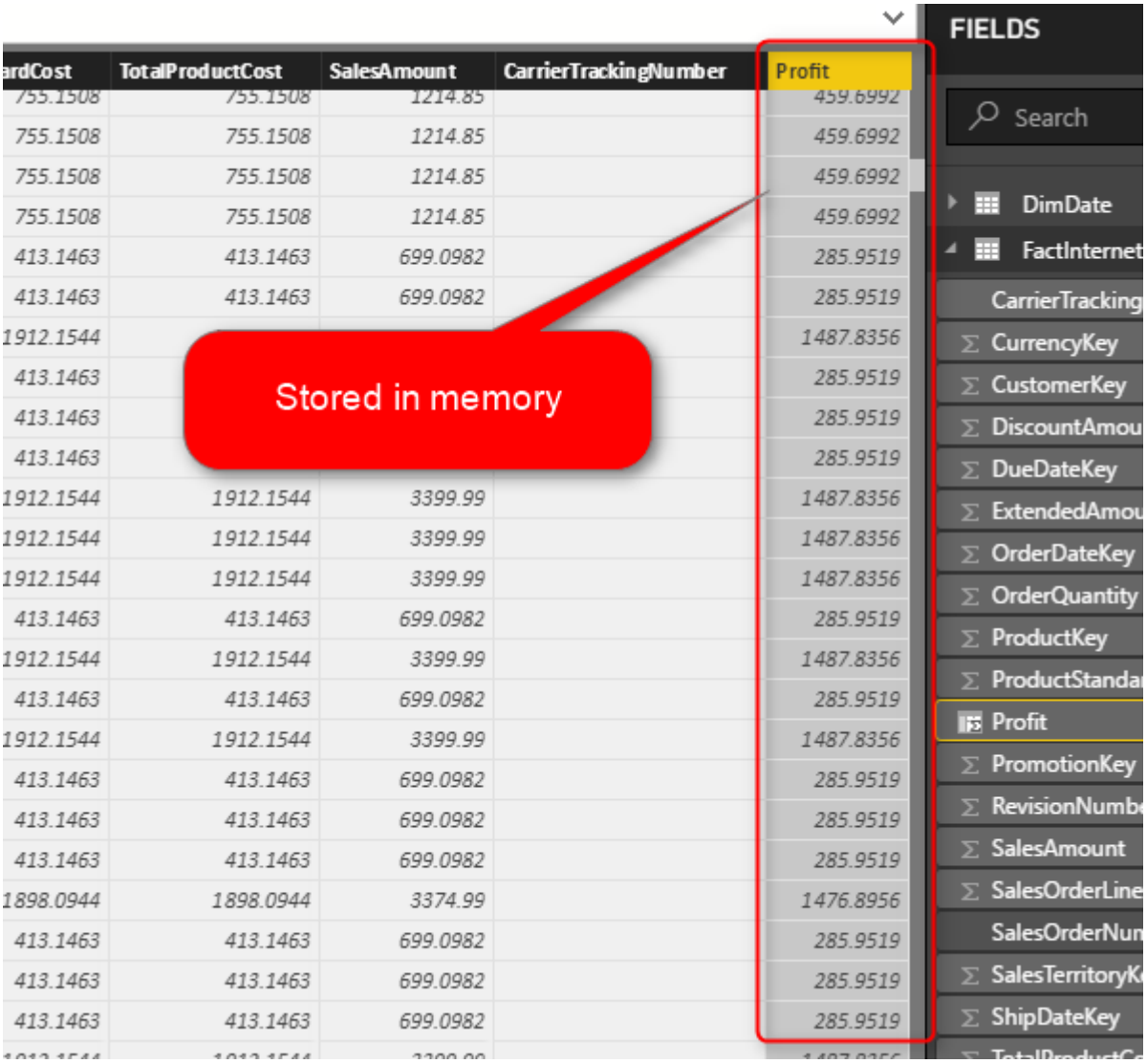


	TotalProductCost	SalesAmount	CarrierTrackingNumber	Profit
08	755.1508	1214.85		459.6992
08	755.1508	1214.85		459.6992
08	755.1508	1214.85		459.6992
08	755.1508	1214.85		459.6992
63	413.1463	699.0982		285.9519
63	413.1463	699.0982		285.9519
44	1912.1544	3399.99		1487.8356
63	413.1463	699.0982		285.9519
63	413.1463	699.0982		285.9519
63	413.1463	699.0982		285.9519
44	1912.1544	3399.99		1487.8356
44	1912.1544	3399.99		1487.8356
44	1912.1544	3399.99		1487.8356
63	413.1463	699.0982		285.9519
44	1912.1544	3399.99		1487.8356
63	413.1463	699.0982		285.9519
44	1912.1544	3399.99		1487.8356
63	413.1463	699.0982		285.9519
63	413.1463	699.0982		285.9519
63	413.1463	699.0982		285.9519

Row by row calculation called Row Context in DAX terminologies.

Stored in Memory

Calculated Column stores values in the memory, like any other columns. The calculation happens at Refresh time, and the result will be stored in the memory.



This means that more calculated memory you have, more memory consumption you will end up with, and your refresh time will be longer as well. However, many calculations are very simple, so your refresh time might not be affected too much.

Calculated Column highlights

Based on the above explanations, here are highlights of a calculated column;

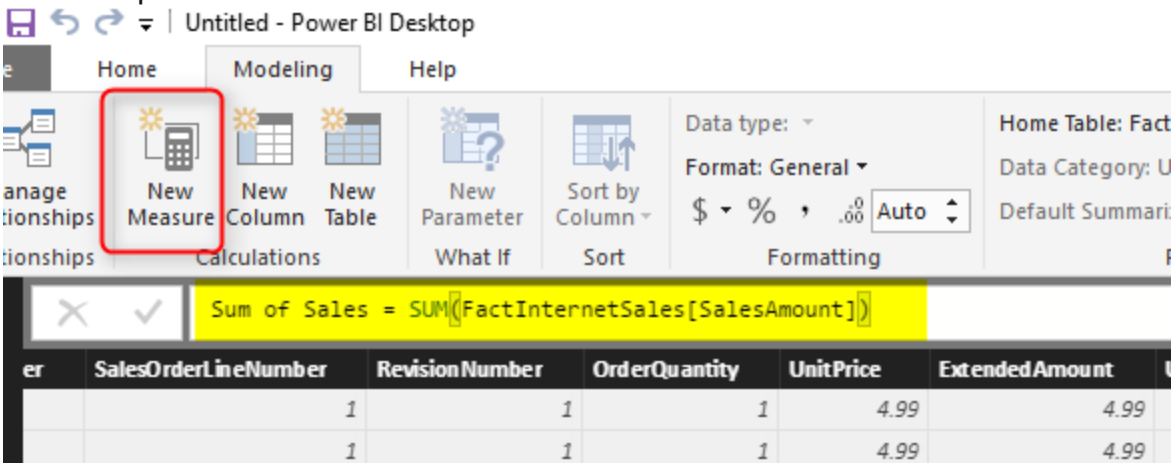
- Row by row calculation: Row Context (usually, not always)
- Stored in the memory (consumes RAM)
- calculated at the time of refreshing the report (either scheduled basis, or manual)

What is Measure?

A measure is usually a calculation that works on an aggregated level basis. This aggregation can be as simple as a sum of sales or can be a little bit more complex, such as calculating monthly average sales in a rolling 12 months period. Measures have a dynamic nature; they affect on a subset of data from one or more table. Hence, the subset of data can be changed through the filters applied in the Power BI Report; then the calculation will have to be evaluated dynamically. So Measures are not pre-calculated, they will be calculated on the fly when adding it in the report.

Example: Sum of Sales

Measures are usually aggregations. A very simple aggregation we can use as an example is a sum of sales.



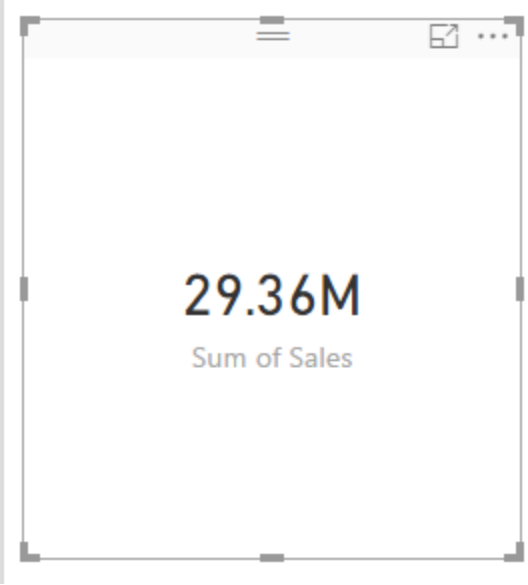
Aggregation can be done with some functions in DAX, such as Sum, SumX, Average, Calculate, and heaps of other aggregation functions. Now, let's answer the most important question:

How to see the Value of the Measure?

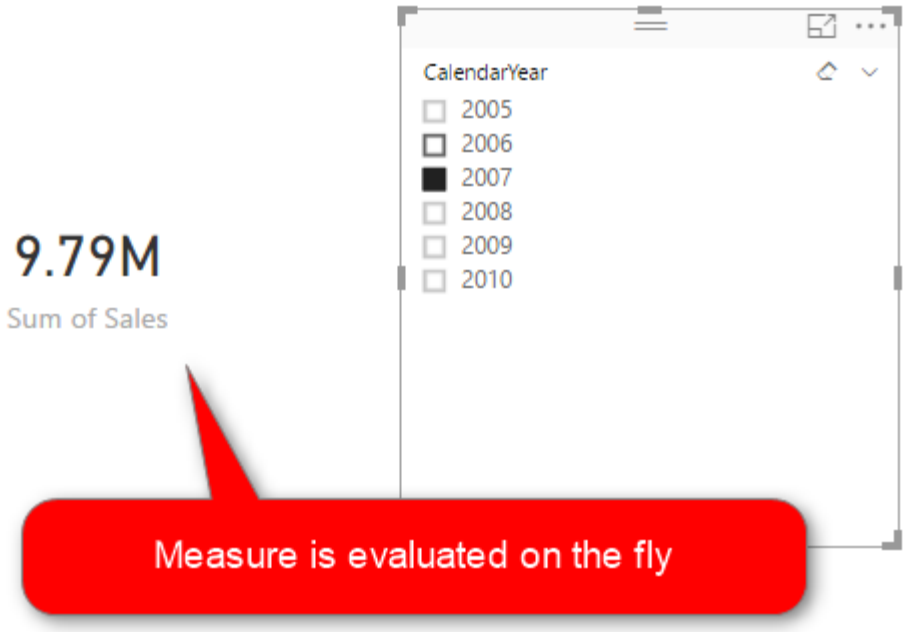
Measures are calculated on the fly. This is, in fact, one of the most conceptual differences between a measure and calculated column. Okay, measure values

are calculated on the fly, so how you can see the value?! The answer is by putting that into a report!

If I drag the measure above in a report as a card visual, then I would get a result;



When there is no filter applied in the report, this will return the total of sales, $29.36M. However, if I add a slicer in the report, and select a value in it, I'll see a different result;

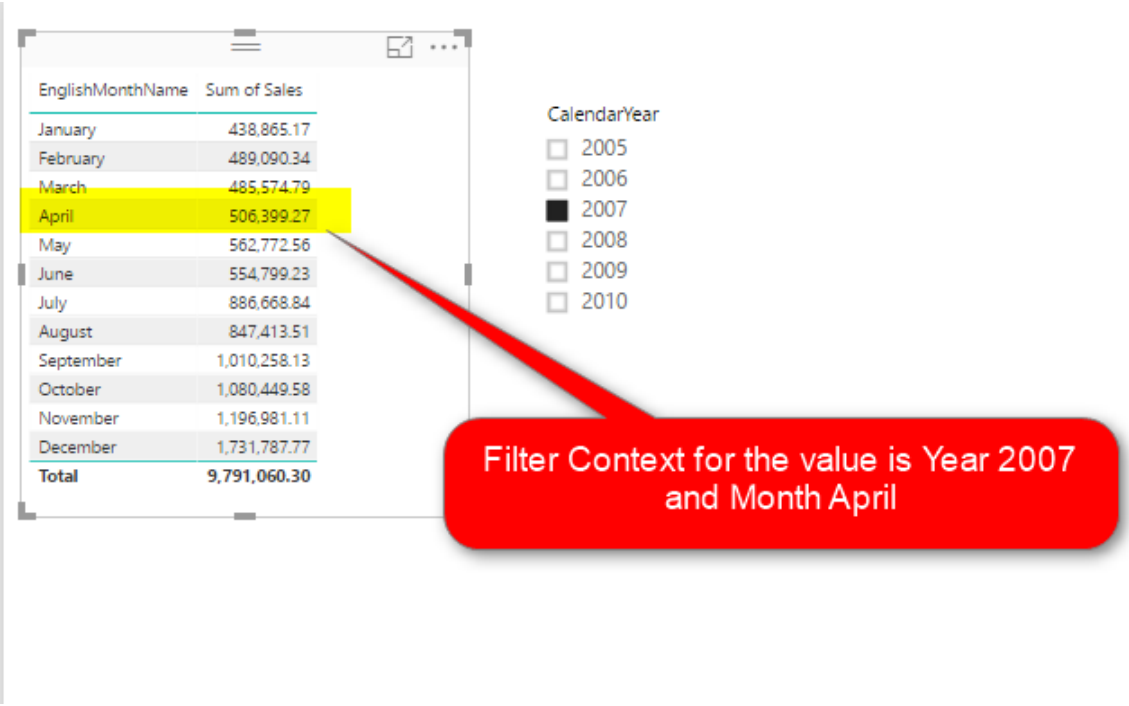


Now the measure calculation only shows me the sum of sales for the year 2007, which is $9.79M.

Filter Context

Measure evaluates on the fly; if there is a slicer value for 2007, then the calculation will be done on the subset of data which is for 2007. If there is a table in visualization somewhere that slice and dice data by Education category, the result of the measure will take that into account as well. We can then say this;

Measure evaluates the value based on the subset of data selected by filters, slicers, or slicing and dicing components of visuals in the report. This filtered dataset, called Filter Context.



Filter Context is a combination of all filters that effect on the calculation of measure. There are much more to talk about when we say filter context. However, this should be enough for understanding the rest of this article.

Measures do not consume RAM; they consume CPU

Based on what you've learned above; measure calculation is done on the fly. This means to measure value is not stored in the memory. The measure will not consume Memory or RAM at all. On the other hand, Measures consume CPU, because their calculation should be done right at the time of visualizing it. If you change a filter or slicer, the calculation should be done again. Because the response time should be fast, then this calculation happens by CPU.

What is the side effect?

If you have many measures in your report, and their calculation is also complex calculation, then with changing every filter or slicer, you end up with

a lot of rounding circles which shows CPU is desperately working hard to calculate all values.

Measures highlights

Based on the above explanations, here are highlights of a Measure;

- Calculated based on all filters: Filter Context (usually, not always)
- Is not Stored and is not pre-calculated
- Calculated on the Fly when you put it on a report page when you change a slicer, filter, or click on a column chart or any other visual to highlight and it affect this measure's value.
- Consumes CPU for calculation.

When to use Measure, and when to use Calculated Column

Now that you know about these two types of calculation, we come to the critical question: When to use which? Do you need a Measure for your calculation or Calculated Column? The answer to this question is depends on what you want to calculate? This is an important question that you should be asking yourself when you want to create a new calculation:

Is the calculation row by row? Or it is an aggregation? Is it going to be affected by filter criteria in the report?

If the calculation is row by row (example: Profit = Sales – Cost, or Full name = First Name & " " & Last Name), then Calculated Column is what you need.

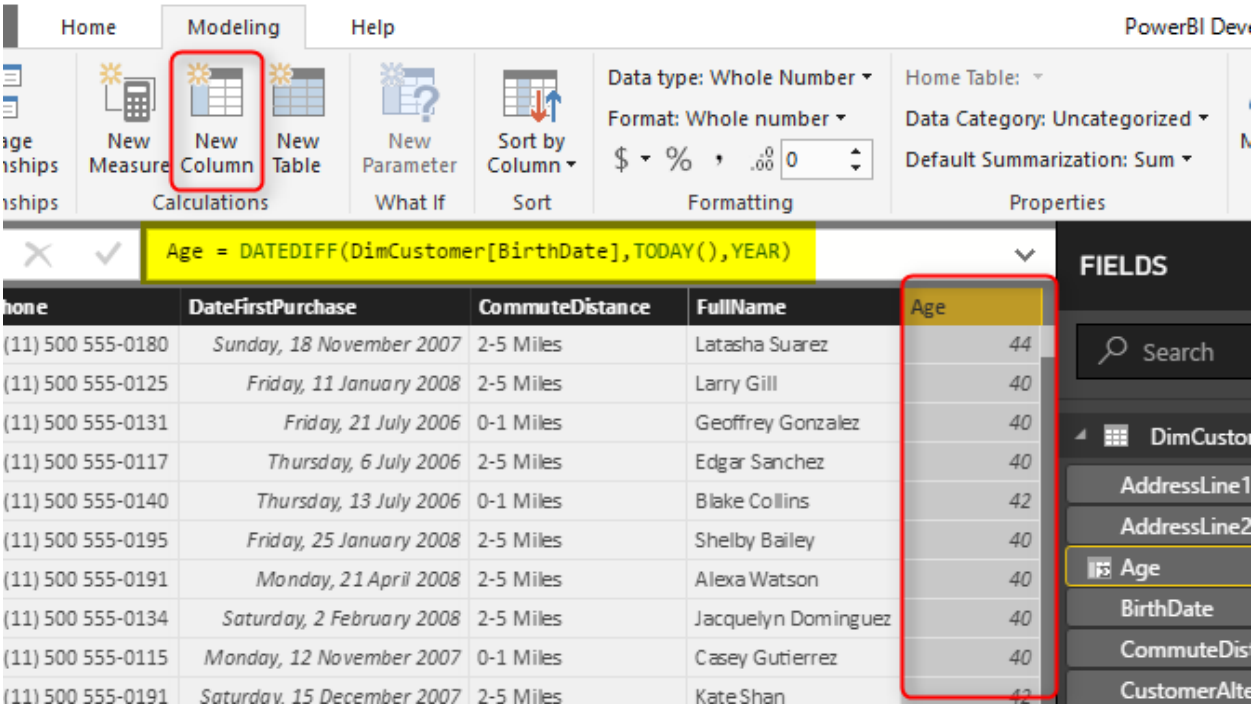
If the calculation is an aggregation or it is going to be affected by filter criteria in the report (example: Sum of Sales = Sum(Sales), or Sales Year to Date = TotalYTD(….)), then Measure is your friend.

Let's go through some examples;

Example 1: Calculating the age of customers

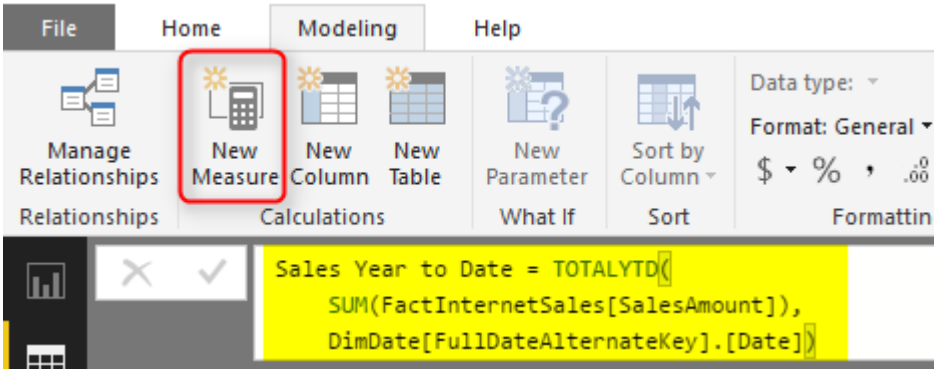
Age of customers does not change based on filters! It is only dependent on one thing; birthdate of the customer. And in the customer table, usually, you

have the birthdate as a field. So this calculation can be simply a calculated column, which evaluates row by row for every customer.



Example 2: Calculating Sales Year to Date

Year to date calculation depends on the filter criteria in the report, and also it is an aggregation. It becomes very complicated to calculation year to date for all variations of fields (per day, per month, per customer, per product, etc). So this needs to be a Measure.



every time you put this measure into a report it calculates based on the filter criteria of the report;

Year	Quarter	Month	Sales Year to Date
2005	Qtr 3	July	473,388.16
2005	Qtr 3	August	979,579.85
2005	Qtr 3	September	1,453,522.89
2005	Qtr 4	October	1,966,852.36
2005	Qtr 4	November	2,510,845.77
2005	Qtr 4	December	3,266,373.66
2006	Qtr 1	January	596,746.56
2006	Qtr 1	February	1,147,563.25
2006	Qtr 1	March	1,791,698.45
2006	Qtr 2	April	2,455,390.74
2006	Qtr 2	May	3,128,946.94
2006	Qtr 2	June	3,805,710.59
2006	Qtr 3	July	4,306,075.74
2006	Qtr 3	August	4,852,077.21
2006	Qtr 3	September	5,202,544.20
2006	Qtr 4	October	5,617,934.44

Calculated Column or Power Query?

When it comes to calculating row by row, then Power Query is a better option in the majority of the cases. I have explained before what is M or DAX, and what are scenarios that you need to use each. Calculated Columns (in the majority of the cases, not always), can be implemented by Power Query. Read my other post about M or DAX; That is the question to learn more about this.

Measure: The Hidden Gem of DAX

You can do a Calculated column in the majority of the cases in Power Query as well, and in fact, it is much better to do that in Power Query in those cases. This means the hidden gem of DAX is Measure. Measure calculation is dynamic, on the fly, and based on filters applied in the report. The dynamic nature of measure calculation, make it the invincible feature of DAX or Power BI. You have seen in the above calculation that Year to Date value is showed

by month. If you simply bring Day value in the table, then this calculation will evaluate on a daily basis and works still perfectly fine;

Year	Quarter	Month	Day	Sales Year to Date
2005	Qtr 3	July	1	14,477.34
2005	Qtr 3	July	2	28,408.86
2005	Qtr 3	July	3	43,421.04
2005	Qtr 3	July	4	50,577.58
2005	Qtr 3	July	5	65,589.75
2005	Qtr 3	July	6	79,902.83
2005	Qtr 3	July	7	87,758.47
2005	Qtr 3	July	8	95,614.11
2005	Qtr 3	July	9	116,523.89
2005	Qtr 3	July	10	127,080.42
2005	Qtr 3	July	11	141,393.50
2005	Qtr 3	July	12	155,528.30
2005	Qtr 3	July	13	162,684.84
2005	Qtr 3	July	14	187,732.73
2005	Qtr 3	July	15	198,963.36
2005	Qtr 3	July	16	213,276.44
2005	Qtr 3	July	17	227,411.24
2005	Qtr 3	July	18	234,364.50
2005	Qtr 3	July	19	259,933.21
2005	Qtr 3	July	20	271,188.84
2005	Qtr 3	July	21	285,501.92

If you do it on a quarter level, the year to date calculation evaluates on the quarter level;

Year	Quarter	Sales Year to Date
2005	Qtr 3	1,453,522.89
2005	Qtr 4	3,266,373.66
2006	Qtr 1	1,791,698.45
2006	Qtr 2	3,805,710.59
2006	Qtr 3	5,202,544.20
2006	Qtr 4	6,530,343.53
2007	Qtr 1	1,413,530.30
2007	Qtr 2	3,037,501.36
2007	Qtr 3	5,781,841.84
2007	Qtr 4	9,791,060.30
2008	Qtr 1	4,283,629.96
2008	Qtr 2	9,720,059.11
2008	Qtr 3	9,770,899.74
2008	Qtr 4	9,770,899.74

This Dynamic nature of Measure calculation in DAX is something that you cannot find in many tools. That is why Measures are so commonly used in DAX. 70% of your time when you write DAX is used for writing measures, if not more!

Summary: Calculated Column vs. Measure in a nutshell

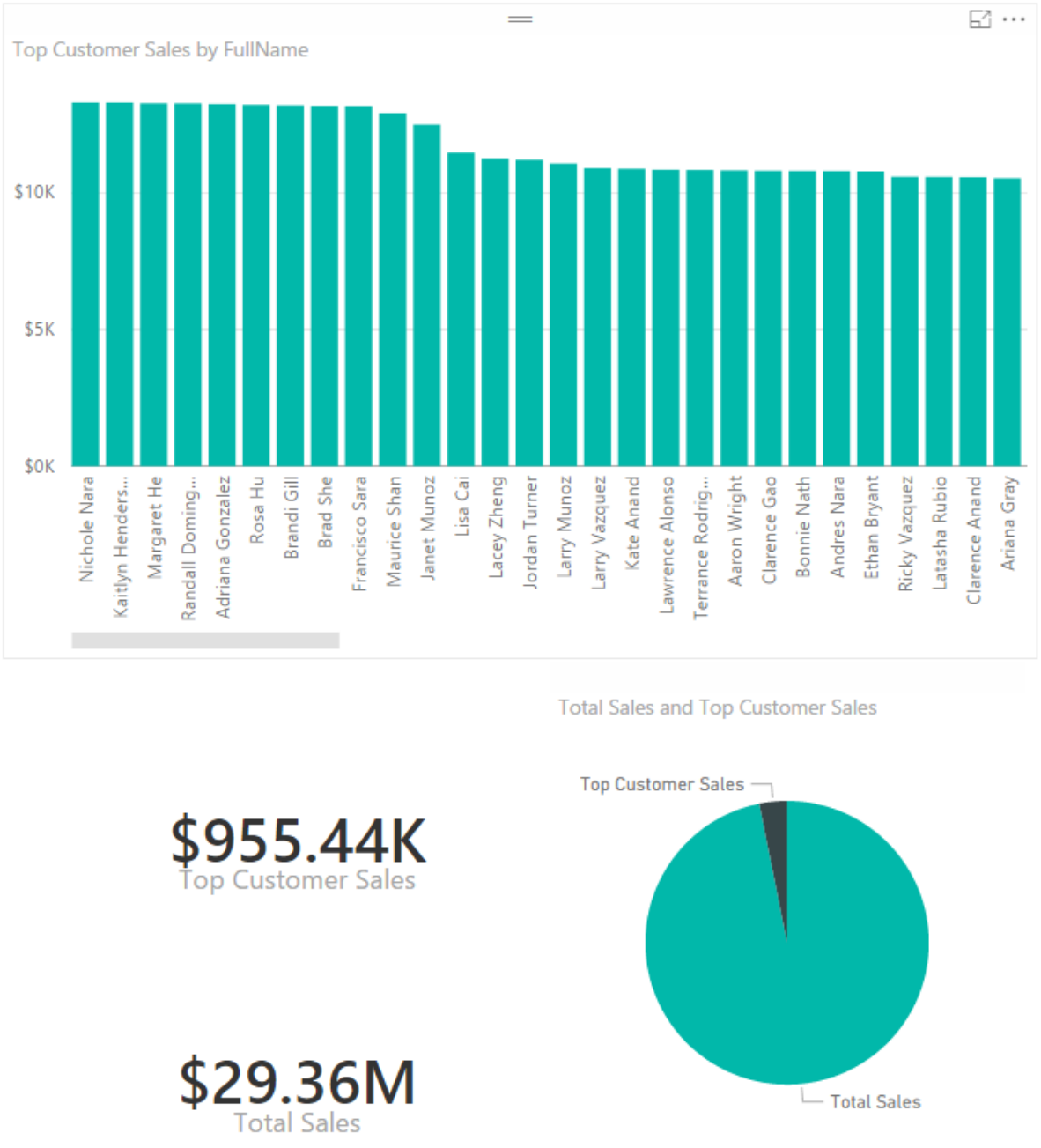
Let's wrap up it all and go through a comparison of these two types of calculations in DAX;

Measure	Calculated Column
Is not stored	Stored in Memory
Calculated on the fly	Calculates at the time of Refresh Report
Consmes CPU	Consumes Memory
Usually is a result of an aggregation	Row by row calculation usually
Value can be seen when adding in the report	Value can be seen in the column in Data tab
DAX usually is the best place for this calculation	In the majority of the cases can be done in Power Query
Example: Sales Year to Date	Example: Profit
Sales YTD = TotalYTD (Sum(Sales), DateField.[Date])	Profit = Sales - Cost

Hopefully, this post, helped you to understand the difference between these two types of calculation. If you have any questions, please don't hesitate to post your questions below.

Scenarios of Using Calculated Tables in Power BI

Posted by Reza Rad on Apr 1, 2016



Calculated tables first introduced in September 2015 update of Power BI Desktop. The name speaks for itself; these are tables created by calculation. As these are in-memory tables, their calculation is based on DAX (Data Analysis

eXpression language). There are many benefits of using Calculated tables, such as using them for role-playing dimensions (for example having more than one date dimension in a model), There are some DAX functions and expressions that returns a table as a result, and using them as a table in your model sometimes is helpful. For example, you might want to create a table for top 10 customers and then use that as the main source table in many reports. In this post, I'll explain to you some use cases of calculated tables.

Prerequisite

For running examples of this post you need to install Adventure Works DW database on SQL Server, or you can download **Excel version** of it from here:

Download

Role Playing Dimension

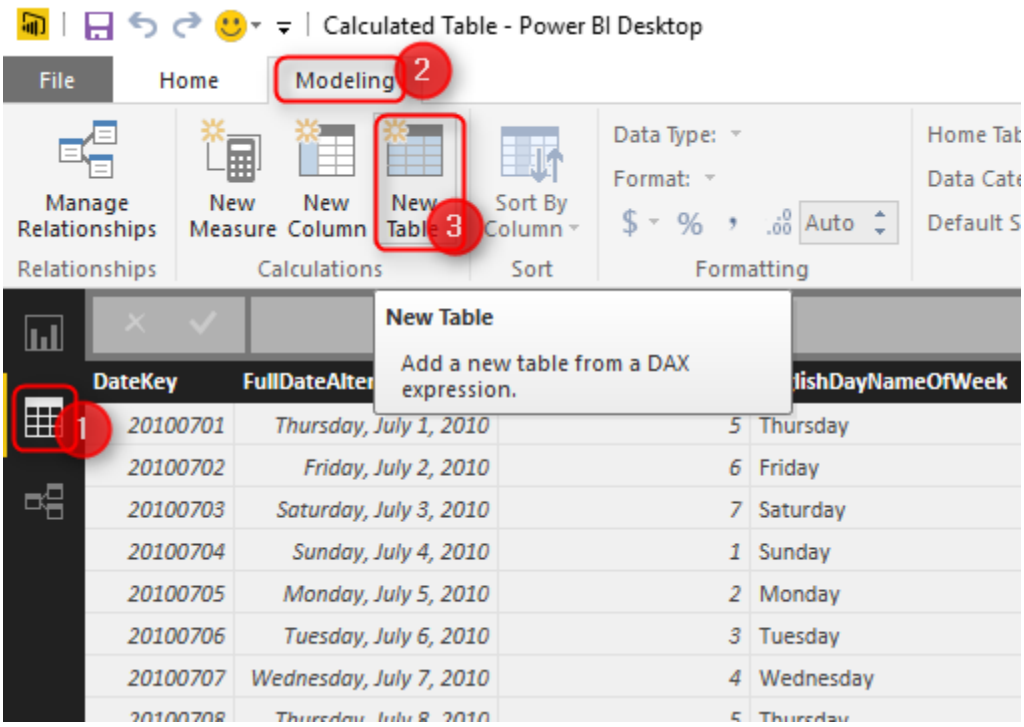
The very first functionality that appears in mind when we talk about Calculated Tables is the ability to create role play dimensions. Role Play dimensions are dimensions with the same structure and data rows that play different roles in our data model. For example Date Dimension is a generic dimension. However, in a sales transaction table, you might have more than one date columns to relate with the date dimension. In the example below we have three date fields in FactInternetSales table: Order Date, Ship Date, and Due Date.

ProductKey	OrderDateKey	DueDateKey	ShipDateKey	CustomerKey	PromotionKey	CurrencyKey	SalesTerritoryKey
528	20130128	20130209	20130204	14870	1	100	4
528	20130129	20130210	20130205	15319	1	100	4
528	20130131	20130212	20130207	16384	1	100	4
528	20130131	20130212	20130207	15476	1	100	4
528	20130201	20130213	20130208	15861	1	100	4
528	20130203	20130215	20130210	26017	1	100	4
528	20130203	20130215	20130210	14761	1	100	4
528	20130204	20130216	20130211	22038	1	100	4
528	20130204	20130216	20130211	22163	1	100	4
528	20130204	20130216	20130211	16018	1	100	4
528	20130205	20130217	20130212	25839	1	100	4

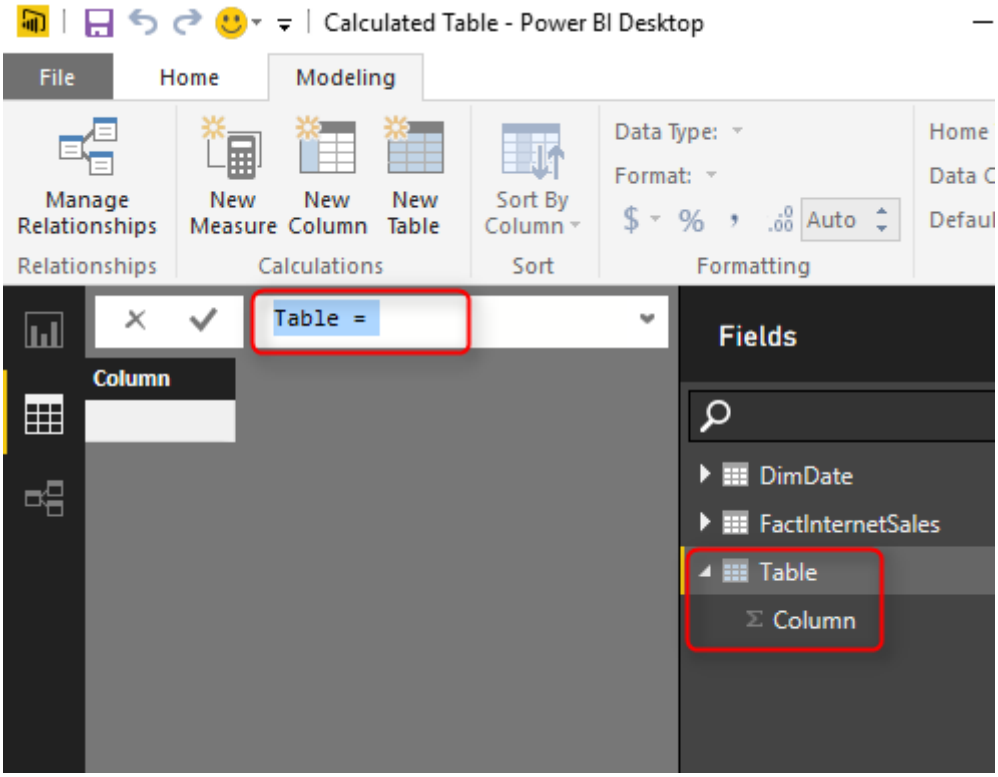
There three fields should be related to three different date dimensions (because tabular model which Power BI is based on that, doesn't support role-playing dimensions built-in). So what you can do is to load the date dimension once in getting Data section from the SQL Server (Here is the T-SQL script for creating date dimension), or by a blank query from Power Query script (Here is the Power Query script for creating the date dimension). Here is the example date dimension loaded in Power BI Desktop (through getting Data):

DateKey	FullDateAlternateKey	DayNumberOfWeek	EnglishDayNameOfWeek	SpanishDayNameOfWeek	FrenchDayNameOfW
20100701	Thursday, July 1, 2010	5	Thursday	Jueves	Jeudi
20100702	Friday, July 2, 2010	6	Friday	Viernes	Vendredi
20100703	Saturday, July 3, 2010	7	Saturday	Sábado	Samedi
20100704	Sunday, July 4, 2010	1	Sunday	Domingo	Dimanche
20100705	Monday, July 5, 2010	2	Monday	Lunes	Lundi
20100706	Tuesday, July 6, 2010	3	Tuesday	Martes	Mardi
20100707	Wednesday, July 7, 2010	4	Wednesday	Miércoles	Mercredi
20100708	Thursday, July 8, 2010	5	Thursday	Jueves	Jeudi
20100709	Friday, July 9, 2010	6	Friday	Viernes	Vendredi
20100710	Saturday, July 10, 2010	7	Saturday	Sábado	Samedi
20100711	Sunday, July 11, 2010	1	Sunday	Domingo	Dimanche
20100712	Monday, July 12, 2010	2	Monday	Lunes	Lundi
20100713	Tuesday, July 13, 2010	3	Tuesday	Martes	Mardi
20100714	Wednesday, July 14, 2010	4	Wednesday	Miércoles	Mercredi
20100715	Thursday, July 15, 2010	5	Thursday	Jueves	Jeudi
20100716	Friday, July 16, 2010	6	Friday	Viernes	Vendredi
20100717	Saturday, July 17, 2010	7	Saturday	Sábado	Samedi
20100718	Sunday, July 18, 2010	1	Sunday	Domingo	Dimanche
20100719	Monday, July 19, 2010	2	Monday	Lunes	Lundi
20100720	Tuesday, July 20, 2010	3	Tuesday	Martes	Mardi
20100721	Wednesday, July 21, 2010	4	Wednesday	Miércoles	Mercredi

Now I can create role-playing dimensions with creating a Calculated table:

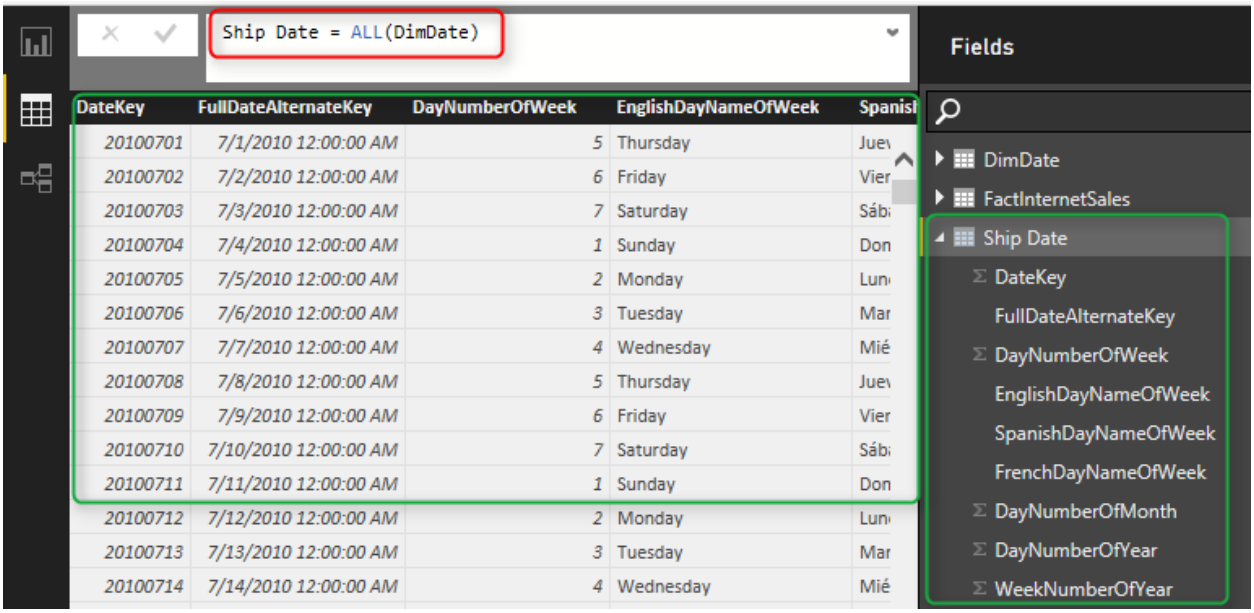


and this will create a table in memory for me and allows me to write the definition of the table



The language for table definition is DAX (Data Analysis eXpression). If you want to know more about DAX you have to read DAX section of the Power BI book. For now, let's keep it simple to see how it works in action. We want an exact copy of the DimDate table here. So I can use ALL function in DAX as below:

1 Ship Date = ALL(DimDate)



DateKey	FullDateAlternateKey	DayNumberOfWeek	EnglishDayNameOfWeek	Spanish
20100701	7/1/2010 12:00:00 AM	5	Thursday	Jue
20100702	7/2/2010 12:00:00 AM	6	Friday	Vier
20100703	7/3/2010 12:00:00 AM	7	Saturday	Sábi
20100704	7/4/2010 12:00:00 AM	1	Sunday	Don
20100705	7/5/2010 12:00:00 AM	2	Monday	Lun
20100706	7/6/2010 12:00:00 AM	3	Tuesday	Mar
20100707	7/7/2010 12:00:00 AM	4	Wednesday	Mié
20100708	7/8/2010 12:00:00 AM	5	Thursday	Jue
20100709	7/9/2010 12:00:00 AM	6	Friday	Vier
20100710	7/10/2010 12:00:00 AM	7	Saturday	Sábi
20100711	7/11/2010 12:00:00 AM	1	Sunday	Don
20100712	7/12/2010 12:00:00 AM	2	Monday	Lun
20100713	7/13/2010 12:00:00 AM	3	Tuesday	Mar
20100714	7/14/2010 12:00:00 AM	4	Wednesday	Mié

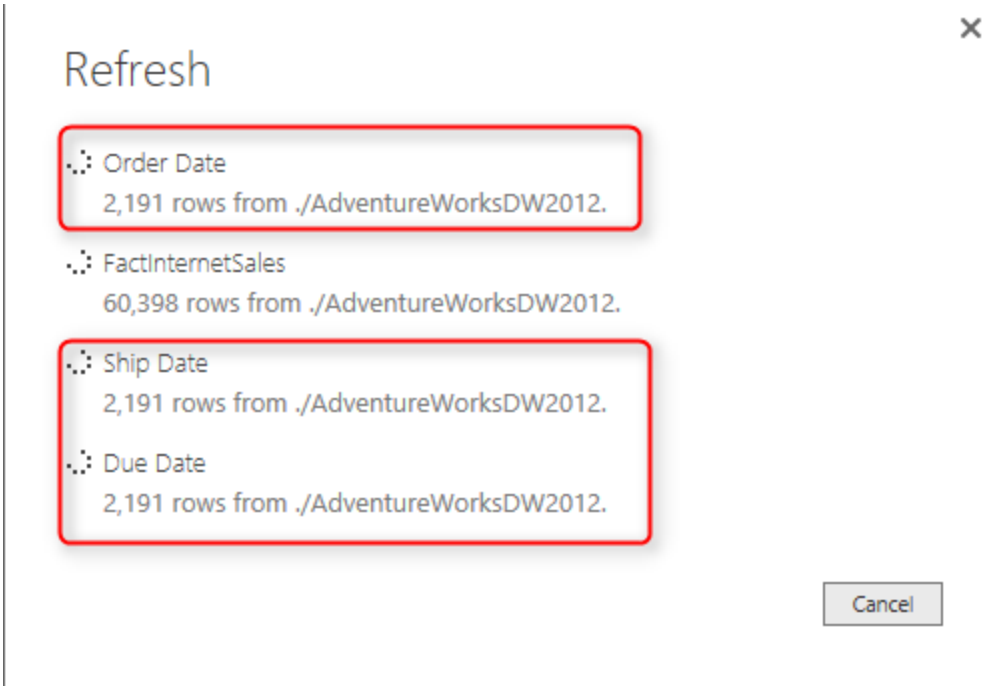
As soon as I type the expression above and press Enter, I'll see the result underneath it as data rows, and also a list of columns in the Fields pane. I've created my role dimension as simple as that. Now I can set up the relationship;

For the relationship above I've also created a Due Date dimension, and renamed the original DimDate to Order Date.

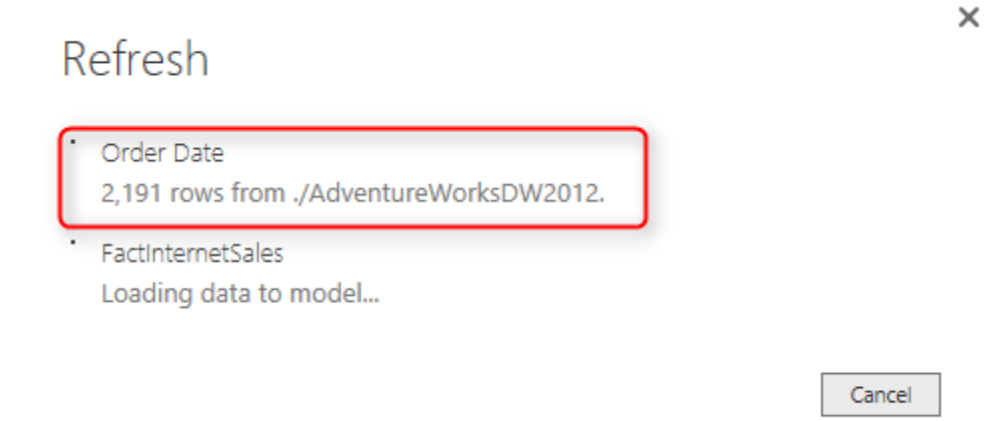
In-Memory Structure, Less Refresh Time

Calculated table loads into memory, so your Power BI file size will increase. However, you don't need to read them again from the external data source. Yes, you can create multiple views in the source database and connect to them through getting Data section with Power Query. However their data need to populate from the source database every time a refresh happens (either scheduled or manual).

WITHOUT Calculated Tables: Here is an example of three date tables loaded from the external data source:



WITH Calculated Tables: and there is only one date dimension loaded (for the role-playing dimension example above):



As you can see this is much more efficient regarding reducing the refresh time. However, the memory consumption would be the same in both methods. Date dimension was a narrow data set example. You might need role-playing for big data tables so that Calculated tables will save you a lot of time in refreshing data in such case.

DAX Table Functions

Some DAX functions return a table. For example, ALL function which I used in the role-playing sample above. ALL was a simple example of a DAX function that returns the whole copy of the source table. Let's have a look at some other examples and see how it works in other scenarios.

Top 100 Customers as a Calculated Table

There are many examples that a business considers top 10 or top 20 customers and filter down the whole dashboard and set of reports only for them. Usually, the main reason is that top 10, 20 customers will bring the majority of revenue to the business. Fortunately, there is a TOPN function in DAX which helps us to build such calculation. TOPN function returns a table. With TOPN we can choose how many rows we want in the result set, and the grouping function to be applied (if there is any) and the aggregation (if there is any).

In this example, I want to show you how to use a calculated table to generate a list of top 100 customers. As a business requirement, I want to visualize the total revenue from the top 100 customers and compare it with total revenue of the whole business. There might be different ways to calculate and visualize it, but I want to do it with a calculated table as a sample scenario.

Summarize

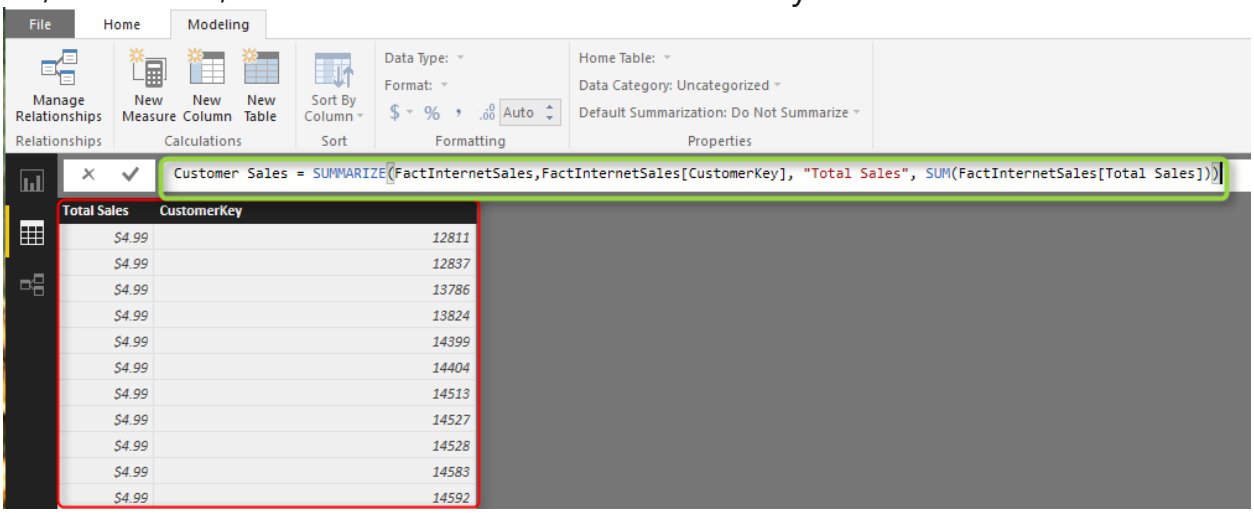
Summarize is a DAX function that generates a grouped by list from a table. Summarize works similar to Group By in T-SQL. So if I want to create a table with CustomerKeys and their total sales amount, I can write this expression:

```
1 Customer Sales =
  SUMMARIZE(FactInternetSales,FactInternetSales[CustomerKey], "Total
  Sales", SUM(FactInternetSales[Total Sales]))
```

Here are details about parameters I passed in the expression above to Summarize function:

- First parameter: Source Table. FactInternetSales is the source table that I want the group by (summarize) operation to be applied to it.
- Second Parameter: Group by Column. CustomerKey in the FactInternetSales table is the column that I want to use as the key for grouping.
- Third parameter: Output Column Name. I named the output calculated column name as Total Sales.
- Fourth parameter: Output Column Calculation. Here I write the calculation for the output column, which sums of Total Sales Column.

So, as a result, I will have a table with CustomerKey and Total Sales.



TOPN

Now that we have the list of customers with their total sales, it is easy to get top 100 customers. I can use TOPN function like this to create another calculated table (I could do this example with only one calculated table instead of two, but I only did it with two tables to help you understand the logic better);

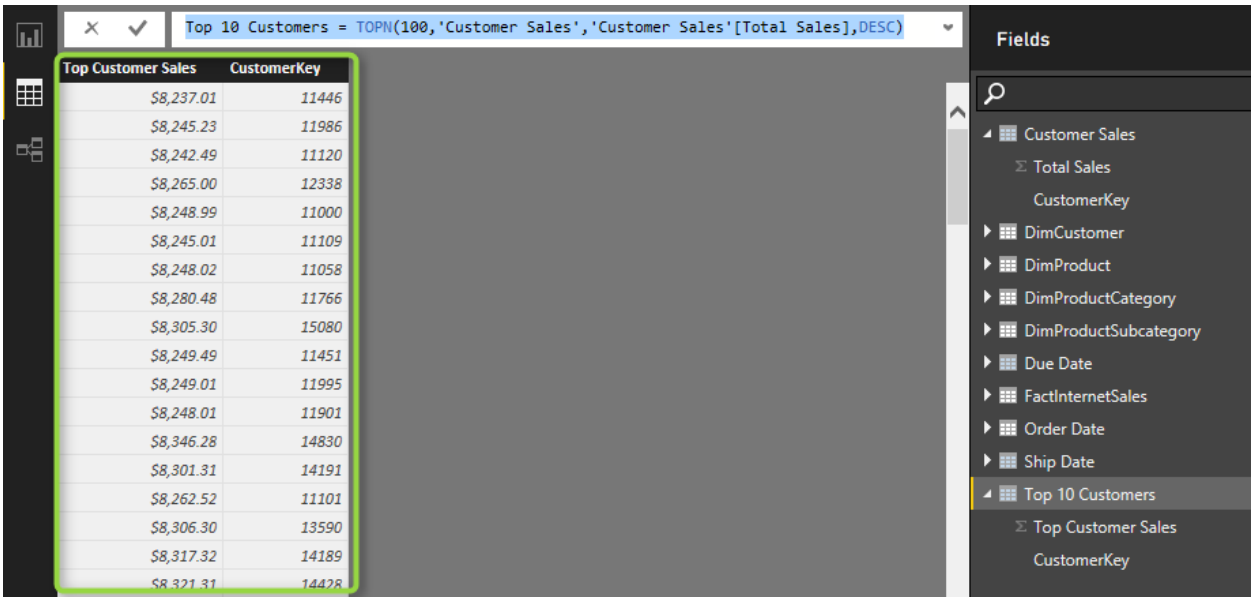
```
Top 10 Customers = TOPN(100,'Customer Sales','Customer Sales'[Total Sales],DESC)
```

And expression above means:

- First parameter: Number of rows to return. One hundred for top 100 customers.
- Second parameter: Source Table. Customer Sales is the source of this operation which we want to fetch top 100 customers from it.

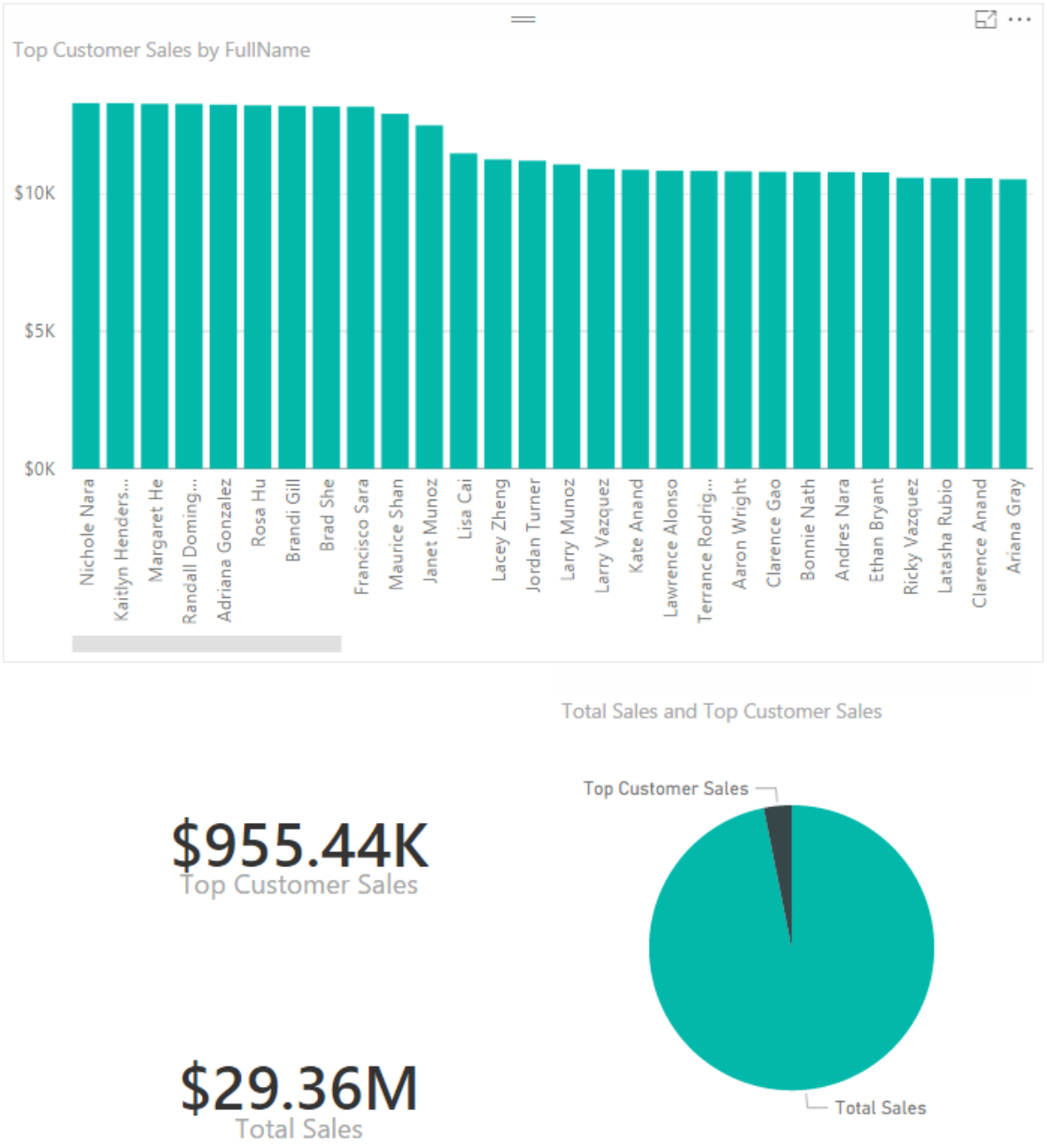
- Third parameter: Order By Column. I want to order the table based on Total Sales of each Customer.
- Fourth parameter: Order By expression (ASC, or DESC). To get top 100 customers, I have to order it by Total Sales DESC.

And here is the result:



I also renamed the Total Sales column to Top Customer Sales (as you see in the screenshot above).

Now I can simply build a report in Power BI to show the difference between Total Sales and Top Customer Sales (If you don't know how to create visualization below follow the visualization chapters of Power BI book):



Great, We've used calculated tables for getting some insight out of top 100 customers and compared it with the total. There are many other cases that you can use Calculated Table for. Chris Webb mentioned using Calendar function in his blog post as a calculated table to start building a date dimension.

Limitations

As any other DAX related limitations, calculated tables very first limitation is the memory. You need to have enough memory to use these tables. This limitation is also an advantage on the other hand because in-memory structure makes these calculations fast.

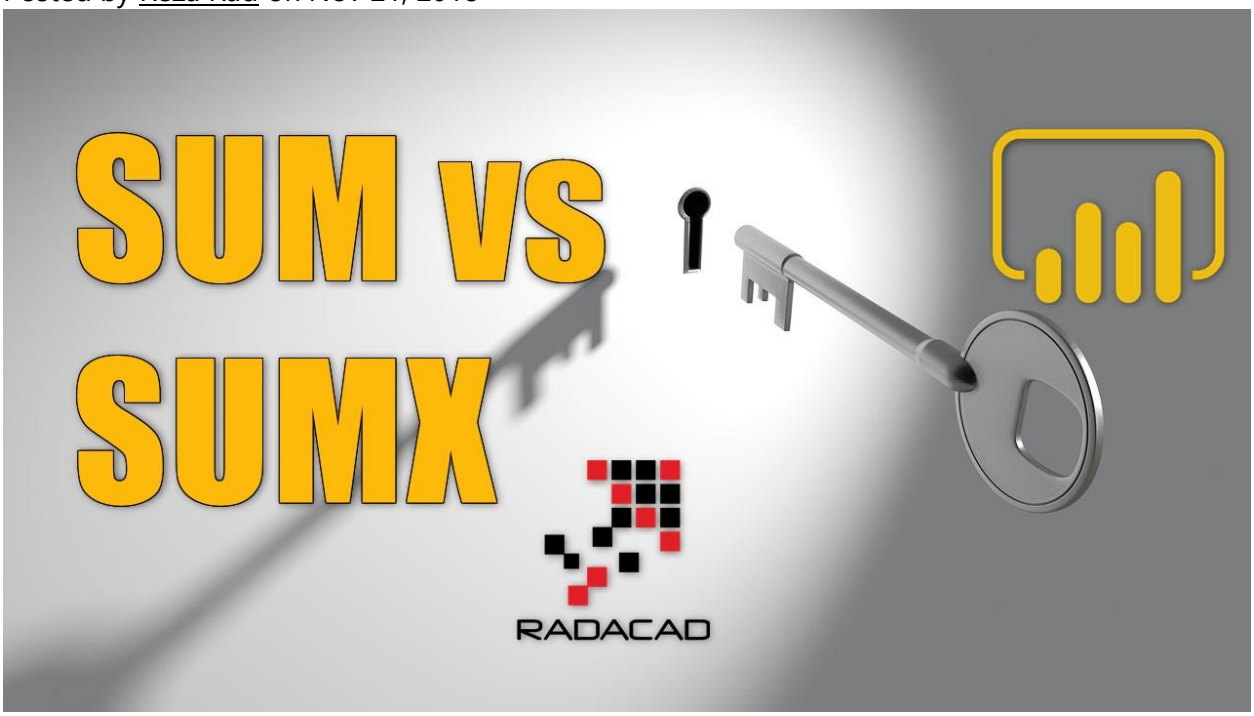
The other limitation which I like to mention at this stage is: Not Inheriting Formatting.

By not inheriting formatting I mean the calculated table doesn't inherit format from the source table. In some complex scenarios where the calculation comes from many tables that might not be necessary. But for our role-playing simple example above; If my original date dimension has some formatting configuration. Such as setting DateKey to a "Do Not Summarize", or some other configuration, then I would like to see the same in the calculated table fetched out of this.

The formatting applied on the calculated table columns also will be overwritten after each change in the DAX expression.

SUM vs. SUMX; What is the Difference of the two DAX Functions in Power BI?

Posted by Reza Rad on Nov 21, 2018



Sum and Sumx are functions that often founded to be misleading for many Power BI users. As both functions are doing the aggregation, it seems a bit confusing what is the actual difference between these two. There are many blog posts and articles about each function. I always explain the difference with simple demos in my courses and presentations, and people find it easy to understand. So, I thought better to write it in a blog post for everyone to read. If you want to learn more about Power BI, read Power BI from Rookie to Rock Star book.

SUM: Aggregation Function

SUM is a simple aggregation function. It summarizes a value based on a filter context. For example, if I have a measure like:

1 Sum of Sales = SUM(FactInternetSales[SalesAmount])

This measure is simply calculating the summarized value of the SalesAmount across the entire fact table when there is no filter selected. And if I have a filter somewhere in my visualization, then it will calculate the sum of the filtered context;

EnglishEducation	Sum of Sales
Bachelors	$9,900,142.76
Graduate Degree	$5,460,560.25
High School	$4,638,026.07
Partial College	$7,723,542.88
Partial High School	$1,636,405.26
Total	**$29,358,677.22**

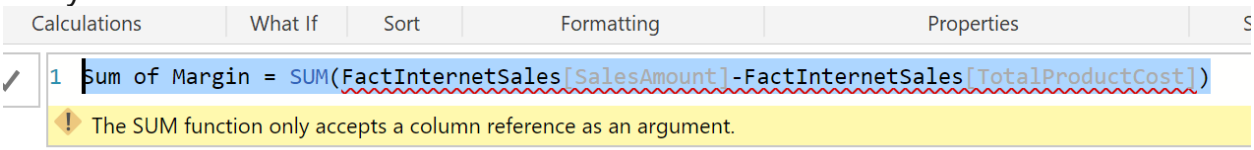
All other aggregation functions are also working the same; Average, Min, Max, Count, etc. Now let's see when SUM functions fall short.

SUMX: Some of an Expression

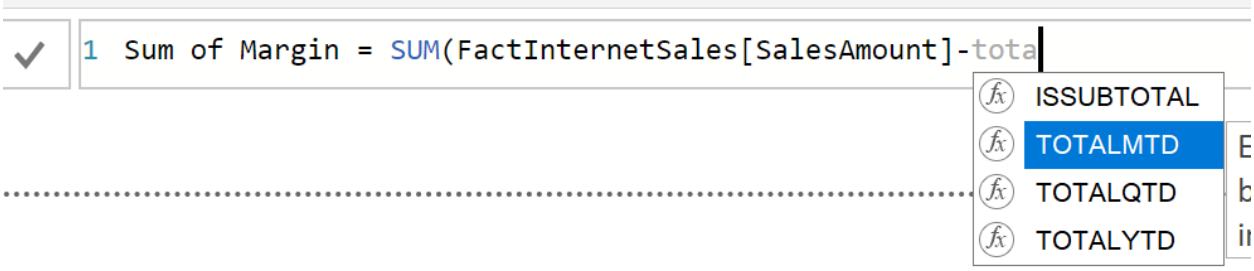
Let's now calculate the sum of margin, which is: the sum of sales minus cost. This calculation is considering that we do NOT have a column as a margin in our model, and we do not want to create that column. Let's see how it is possible through a measure to calculate the sum of Margin.

Margin calculation is: SalesAmount – TotalProductCost

But you cannot write a measure like below:

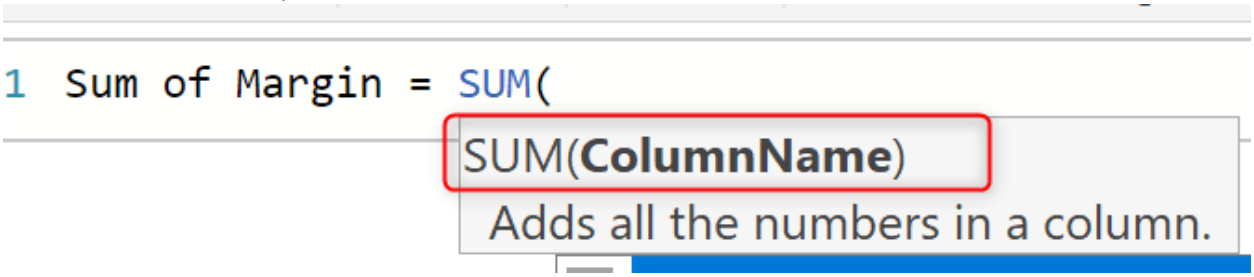


When you start writing that measure, you don't even get the DAX intelligence for the second part of your expression:



DAX intellisence doesn't show the TotalProductCost column from the FactInternetSales table, but the column is definitely in the table. The intellisence in DAX is always reliable; If it doesn't allow you to write something somewhere, it means based on the structure of your expression or functions that you've used, it is not probably the right place to write it. So why you cannot write such a simple statement?

Because SUM only accepts a column name as input. Here is the structure of the SUM function;



As you can see the input is just one column name. It cannot be one column minus another one; that means an expression. So, what is the way to do it? One way is to use multiple sum functions, such as below code:

```
1 Sum of Margin = SUM(FactInternetSales[SalesAmount])-
  SUM(FactInternetSales[TotalProductCost])
```

And it would work. However, for long expressions, this way of writing will become hardly readable. If you add one Sum in front of every column name, you may end up with expressions such as below;

```
1 A measure with few SUMs =
2 if((SUM(FactInternetSales[SalesAmount])
```

```
 3  -SUM(FactInternetSales[TotalProductCost]))
 4  /SUM(FactInternetSales[OrderQuantity])
 5  >SUM(FactInternetSales[ExtendedAmount])
 6  ,SUM(FactInternetSales[ExtendedAmount])
 7  -SUM(FactInternetSales[SalesAmount])
 8  ,SUM(FactInternetSales[OrderQuantity])
 9  *(SUM(FactInternetSales[UnitPrice])
10 -SUM(FactInternetSales[UnitPriceDiscountPct])))
```

Looks scary. Well, there is another way; use SUMX. SUMX is the sum of expression, the X at the end of this function is for eXpression. This function gives you the sum of any expression. Here is a way to use it:

SumX(<table name>,<expression>)

For SUMX to work, you need to specify a table name. When you use SUM, you do not need a table name, because one column only belongs to one table. But when you use SUMX, you may write an expression which uses columns from other tables. In the example for Margin, both columns are coming from the same table; FactInternetSales. So, our expression would be:

```
1 Sum of Margin = SUMX(
2 FactInternetSales,
3 FactInternetSales[SalesAmount]-FactInternetSales[TotalProductCost]
4 )
```

```
Sum of Margin = SUMX(
        FactInternetSales,
        FactInternetSales[SalesAmount]-FactInternetSales[TotalProductCost]
)
```

EnglishEducation	Sum of Sales	Sum of Margin
Bachelors	$9,900,142.76	4,064,815.22
Graduate Degree	$5,460,560.25	2,252,766.47
High School	$4,638,026.07	1,896,523.38
Partial College	$7,723,542.88	3,186,803.34
Partial High School	$1,636,405.26	679,975.24
Total	**$29,358,677.22**	**12,080,883.65**

SUMX is the sum of expression, but the SUM is just summarizing values of one single column.

SUMX is an Iterator Function

How is SUMX behind the scene doing the calculation? SUMX will go through every single record of the input table and run the expression for that record; it stores the result of that into temporary memory.

TotalProductCost	SalesAmount	[SalesAmount]-[TotalProductCost]
26.1763	69.99	43.8137
13.09	35	21.91
11.2163	29.99	18.7737
6.9223	8.99	2.0677
6.9223	8.99	2.0677
9.1593	24.49	15.3307
6.9223	8.99	2.0677
419.7784	769.49	349.7116
755.1508	1214.85	459.6992
419.7784	769.49	349.7116
8.2205	21.98	13.7595
11.2163	29.99	18.7737
11.2163	29.99	18.7737
308.2179	564.99	256.7721
308.2179	564.99	256.7721
12.1924	32.6	20.4076
6.9223	8.99	2.0677
13.0863	34.99	21.9037
3.7363	9.99	6.2537
6.9223	8.99	2.0677
12.1924	32.6	20.4076
11.2163	29.99	18.7737
9.1593	24.49	15.3307
419.7784	769.49	349.7116
6.9223	8.99	2.0677
13.0863	34.99	21.9037
12.1924	32.6	20.4076
26.1763	69.99	43.8137
419.7784	769.49	349.7116
6.9223	8.99	2.0677
3.7363	9.99	6.2537

Calculates the expression ONE ROW AT A TIME, and stores the result in TEMPORARY memory

At the end of parsing the table and calculating all values for every single column, it will summarize them all together, because it is SUMx, releases the temporary memory, and visualize the result;

TotalProductCost	SalesAmount	[SalesAmount]-[TotalProductCost]
26.1763	69.99	43.8137
13.09	35	21.91
11.2163	29.99	18.7737
6.9223	8.99	2.0677
6.9223	8.99	2.0677
9.1593	24.49	15.3307
6.9223	8.99	2.0677
419.7784	769.49	349.7116
755.1508	1214.85	459.6992
419.7784	769.49	349.7116
8.2205	21.98	13.7595
11.2163	29.99	18.7737
11.2163	29.99	18.7737
308.2179	564.99	256.7721
308.2179	564.99	256.7721
12.1924	32.6	20.4076
6.9223	8.99	2.0677
13.0863	34.99	21.9037
3.7363	9.99	6.2537
6.9223	8.99	2.0677
12.1924	32.6	20.4076
11.2163	29.99	18.7737
9.1593	24.49	15.3307
419.7784	769.49	349.7116
6.9223	8.99	2.0677
13.0863	34.99	21.9037
12.1924	32.6	20.4076
26.1763	69.99	43.8137
419.7784	769.49	349.7116
6.9223	8.99	2.0677

SUM of all values will be calculated, and then temporary memory will be released

Because of this ITERATION nature of the SUMX function, it is also called as Iterator function. Other iterator functions are: AverageX, MinX, MAXX, CountaX, etc.

Iterator functions are looping through all rows in the input table and storing the expression result in temporary memory storage. At the end they apply the aggregation on the temporary storage

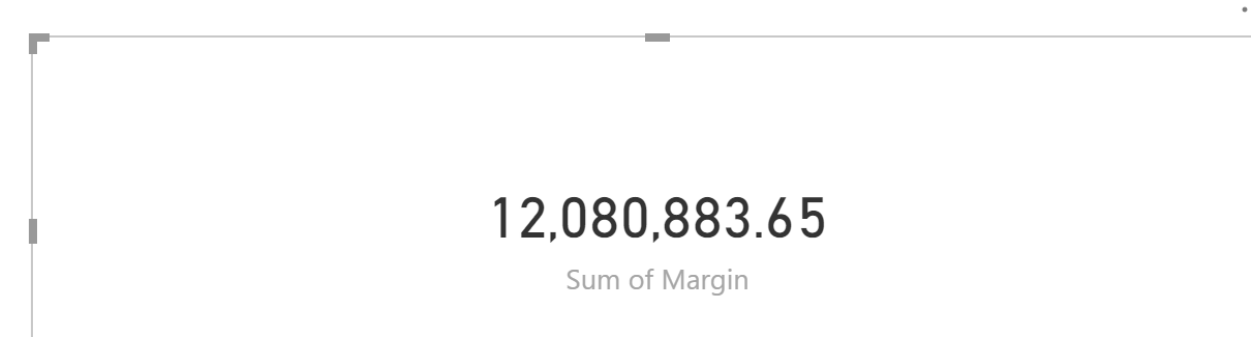
results, release the memory usage, and visualize the calculation result.

One important understanding about SUMX so far, is that SUMX uses memory (temporarily, but still uses memory) for calculation. Another important finding is that it calculates values row by row. If you run SUMX on a very large table with a complex expression, you probably need to wait a bit for results to come through.

The hidden gem of the SUMX; Table Input

The example, you have seen so far about the SUMX was an easy one, which you could even write it without SUMX (remember the way we did it with multiple SUM functions). But the hidden gem of using SUMX function is not just the flexibility on the expression; it is also the flexibility on the table input. Let's say you want to calculate total margin in an expression. How are we going to do that? Well, you may say we use the same SUMX statement that we have used so far, which gives us the result below;

```
1  Sum of Margin = SUMX(
2          FactInternetSales,
3          FactInternetSales[SalesAmount]-FactInternetSales[TotalProductCost]
4  )
```



But the expression above is not always giving you the total of margin. If you slice and dice it by a column here is the result;

```
1 Sum of Margin = SUMX(
2             FactInternetSales,
3             FactInternetSales[SalesAmount]-FactInternetSales[TotalProductCost]
4 )
```

EnglishEducation	Sum of Sales	Sum of Margin
Bachelors	$9,900,142.76	4,064,815.22
Graduate Degree	$5,460,560.25	2,252,766.47
High School	$4,638,026.07	1,896,523.38
Partial College	$7,723,542.88	3,186,803.34
Partial High School	$1,636,405.26	679,975.24
Total	**$29,358,677.22**	**12,080,883.65**

Filter context (or let's say whatever filters the visual) will impact the calculation result. So, when you are looking at Bachelors education category, the sum of Margin for that is not the total margin, it is just sum of margin for that category.

ALL is an interesting function, which I write about it later in another blog post, in the meantime, if I use ALL function to give me the entire table regardless of the filter context, this is what my expression and the result would look like:

```
1 Total Margin = SUMX(
2 ALL(FactInternetSales),
3 FactInternetSales[SalesAmount]-FactInternetSales[TotalProductCost]
4 )
```

```
Total Margin = SUMX(
        ALL(FactInternetSales),
        FactInternetSales[SalesAmount]-FactInternetSales[TotalProductCost]
)
```

EnglishEducation	Sum of Sales	Sum of Margin	Total Margin
Bachelors	$9,900,142.76	4,064,815.22	12,080,883.65
Graduate Degree	$5,460,560.25	2,252,766.47	12,080,883.65
High School	$4,638,026.07	1,896,523.38	12,080,883.65
Partial College	$7,723,542.88	3,186,803.34	12,080,883.65
Partial High School	$1,636,405.26	679,975.24	12,080,883.65
Total	**$29,358,677.22**	**12,080,883.65**	**12,080,883.65**

How does this work? ALL is a function that returns a table as output. SUMX is a function that gets a table as input. So they can work with each other nicely! ALL can be the input table of the SUMX function. Nesting or cascading functions and tables into each other is something that happens very often in DAX. Because ALL is a function that passes the entire table regardless of the filter context, so, we get the full FactInternetSales table with no filters, and the result would always be the total margin.

You may think, what is the usage of such a thing? Well, you can use it to calculate the percentage of the margin for each education category. Here is how it works:

```
Percentage of Margin = DIVIDE([Sum of Margin],[Total Margin])
```

EnglishEducation	Sum of Sales	Sum of Margin	Total Margin	Percentage of Margin
Bachelors	$9,900,142.76	4,064,815.22	12,080,883.65	33.65%
Graduate Degree	$5,460,560.25	2,252,766.47	12,080,883.65	18.65%
High School	$4,638,026.07	1,896,523.38	12,080,883.65	15.70%
Partial College	$7,723,542.88	3,186,803.34	12,080,883.65	26.38%
Partial High School	$1,636,405.26	679,975.24	12,080,883.65	5.63%
Total	**$29,358,677.22**	**12,080,883.65**	**12,080,883.65**	**100.00%**

Any TABLE can be the Input for SUMX

It is not just ALL function that can be the input for SUMX. You can also use any other functions that return table, or any other tables as the input for the SUMX. For example, expression below is giving us the Filtered result of the FactInternetSales table, when the Education category is "High School";

```
Sum of Margin for High School = SUMX(
    FILTER(
        FactInternetSales,
        RELATED(DimCustomer[EnglishEducation])="High School"
        ),
    FactInternetSales[SalesAmount]-FactInternetSales[TotalProductCost]
)
```

EnglishEducation	Sum of Sales	Sum of Margin	Sum of Margin for High School
Bachelors	$9,900,142.76	$4,064,815.22	
Graduate Degree	$5,460,560.25	$2,252,766.47	
High School	$4,638,026.07	$1,896,523.38	$1,896,523.4
Partial College	$7,723,542.88	$3,186,803.34	
Partial High School	$1,636,405.26	$679,975.24	
Total	**$29,358,677.22**	**$12,080,883.65**	**$1,896,523.4**

In this example, FILTER function used as the input for SUMX to give us the calculation result only on a filtered dataset.

1 Sum of Margin for High School = SUMX(
2 FILTER(

3 FactInternetSales,
4 RELATED(DimCustomer[EnglishEducation])="High School"
5),
6 FactInternetSales[SalesAmount]-FactInternetSales[TotalProductCost]
7)

This can be done with other functions as well. Here, for example, I used the CalculateTable function to do the filtering:

```
1 Sum of Sales by Customer = SUMX(
2 CALCULATETABLE(
3 FactInternetSales,
4 DimCustomer[EnglishEducation]="High School"
5 ),
6 FactInternetSales[SalesAmount]-FactInternetSales[TotalProductCost])
```

The result is:

```
1 Sum of Sales by Customer = SUMX(
2         CALCULATETABLE(
3             FactInternetSales,
4             DimCustomer[EnglishEducation]="High School"
5         ),
6         FactInternetSales[SalesAmount]-FactInternetSales[TotalProductCost])
```

EnglishEducation	Sum of Sales	Sum of Margin	Sum of Margin for High School	Sum of Sales by Customer
Bachelors	$9,900,142.76	$4,064,815.22		$1,896,523.4
Graduate Degree	$5,460,560.25	$2,252,766.47		$1,896,523.4
High School	$4,638,026.07	$1,896,523.38	$1,896,523.4	$1,896,523.4
Partial College	$7,723,542.88	$3,186,803.34		$1,896,523.4
Partial High School	$1,636,405.26	$679,975.24		$1,896,523.4
Total	**$29,358,677.22**	**$12,080,883.65**	**$1,896,523.4**	**$1,896,523.4**

SUMX is a function that you can set expression and a table as the input. Having the ability to change both the expression, and the table (as a filter or something else), makes this function a generic function to use in DAX.

There is a function that can act more generic than SUMX, called Calculate. I'll write about it in another blog post.

Summary

Sum and SumX both are functions calculating aggregation. However, the SUMX calculates the aggregation on an expression resolved from a table which can be dynamically calculated as well. SUMX is a generic and powerful function, that is why we see the usage of that a lot in DAX. One thing to remember is that SUMX like any other iterator functions, is consuming temporary memory storage and doing the calculation one row at a time, then aggregates it. Do you use SUMX? Let me know where and in which scenarios do you use SUMX for, and ask any questions in the comments below.

IF and Filter is Different! Be Careful (DAX)

Posted by Reza Rad on Mar 20, 2017

Color	Sum of Red Products - With Filter
Red	$7,724,330.52
Total	**$7,724,330.52**

Color	Sum of Red Products - With IF
Black	$0.00
Blue	$0.00
Multi	$0.00
NA	$0.00
Red	$7,724,330.52
Silver	$0.00
White	$0.00
Yell...	$0.00
Total	**$7,724,330.52**

DAX has many functions to write conditional expressions. For example, you might want to calculate the sum of the sales amount for all "Red" products. You can achieve it by using SUMX or Calculate, and functions such as IF or Filter to write a conditional expression for product color to be equal to "Red". At first, you might think these functions will have the same in the result set, but there is a difference that should not be overlooked. In this post, I'll explain what type of problem might happen if you don't use these functions wisely. If you want to learn more about Power BI; read Power BI book from Rookie to Rock Star.

Prerequisite

For running this example you would need to have a copy of AdventureWorksDW database in SQL Server, or you can download Excel version of it from here:

Enter Your Email to download the file (required)

Download

Brief of Functions

IF

"IF" is a conditional filtering expression function for DAX. You can write a conditional expression including Then and Else part of it. It simply works with this syntax;

```
1 IF(<conditional expression>, <what happens if true>, <what happens if false>)
```

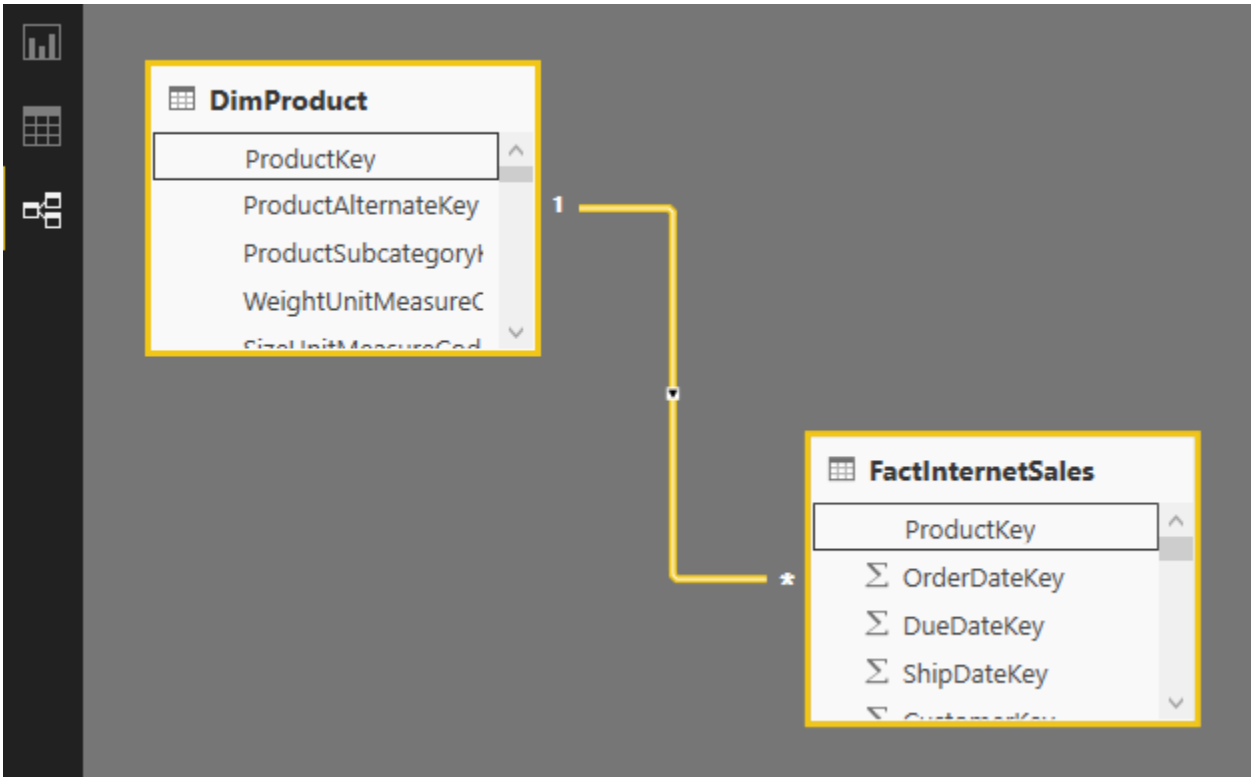
Filter

"Filter" is a function that filters data set based on a custom filter. For example, you can filter only products with "Red" color. Here is an example Filter expression;

```
1 FILTER( <table>, <filter condition>)
```

Sample Data Set

For this example, you need to bring FactInternetSales, and DimProduct into your Power BI Model. The relationship between these tables automatically should be detected by Power BI. It should be based on ProductKey between two tables. Here is how the relationship looks like;



Conditional SUM

There are multiple ways of calculating conditional sum in DAX. You can use SUMX or CALCULATE. Both of these functions calculate an expression (In this case it would be the sum of sales amount from FactInternetSales), based on a filter (which would be our conditional expression to find "Red" products). I will use SUMX in this example, but the same concept applies to Calculate function as well. Here is how you can use SUMX for calculating the sum of "Red" products;

Method 1 – SumX with FILTER

In the first method, I can use SUMX expression and filter the data set to be only "Red" products. Create a new Measure in FactInternetSales with this expression;

```
Sum of Red Products - With Filter = SUMX(
                                    FILTER(FactInternetSales,
```

```
4
5           RELATED(DimProduct[Color])="Red")

        ,FactInternetSales[SalesAmount]
                                                    )
```

As you can see in the above expression, I have used a simple FILTER function to filter everything in FactInternetSales when Color or product is "Red". I have used the RELATED function because Color is a column in DimProduct and Related Function goes through the relationship from Many (FactInternetSales) to One (DimProduct) and allow us to do the filtering based on a column in a related table.

Method 2 – SumX with IF

We can achieve the same result with SUMX and IF condition together. In this case, the condition comes as IF statement in the expression part of SUMX. Here is the new measure's code;

```
  Sum of Red Products - With IF = SUMX(
1                       FactInternetSales,
2
3       IF(RELATED(DimProduct[Color])="Red",
4
5       FactInternetSales[SalesAmount],
6                                                   0)
                    )
```

In this expression, instead of filtering data with FILTER function, I have used a conditional expression to identify if the color of the product is "Red" or not, if it is "Red", then I use SalesAmount for sum calculation, otherwise I use zero (means don't summarize for other product colors).

Method 3 – Calculate with Simple Conditional Expression

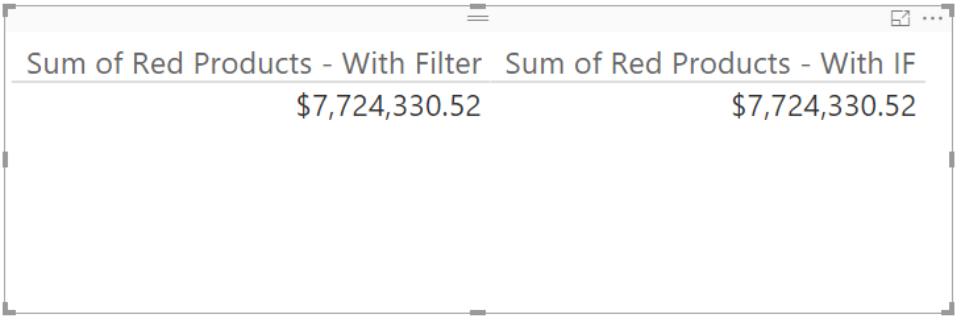
There are many other methods of calculating the conditional sum, but just adding this one because it looks different; If I use Calculate Function with the simple expression for checking the color of the product as a new measure;

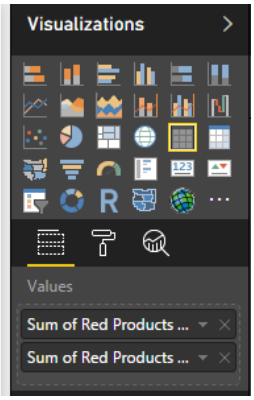
```
Sum of Red Products - Calculate Simple Expression = CALCULATE(
                                                    SUM(FactInternetSales[SalesAmou
1
2 nt]),
3
                        DimProduct[Color]="Red"
4

                    )
```

Writing DimProduct[Color]="Red" in this way is similar to writing a condition for every result set. The final result will be sum of Red Products.

Testing Results – Similar

If you bring both measures in Power BI as a Table Visual you will see the result of both are identical, and it will show you total sales amount for products with "Red" Color correctly;



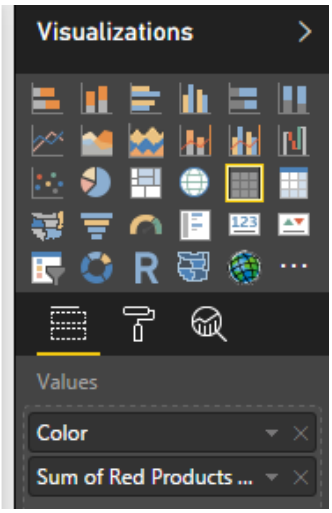


Different Results

The result for the measure is perfectly similar, however, if you use one of these measures for a dataset you will see the result of data set is different, and it changes the result significantly. For example, if you use "Sum of Red Products

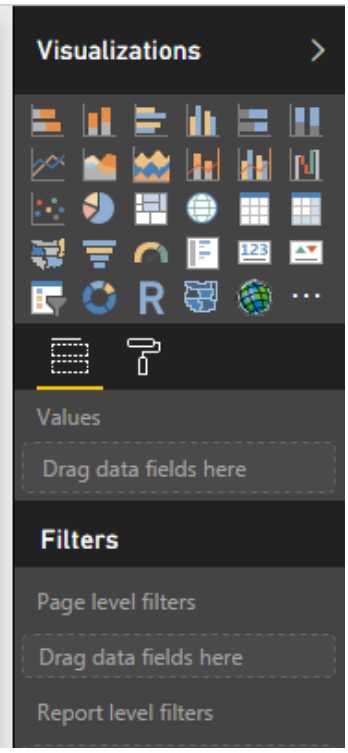
– With Filter" only in a table with "Color" from DimProduct, here is what you will see:

C...	Sum of Red Products - With Filter
Red	$7,724,330.52
Total	**$7,724,330.52**



If you use "Sum of Red Products – With IF" only in a table with "Color" from DimProduct, you will see the different result;

Color	Sum of Red Products - With IF
Black	$0.00
Blue	$0.00
Multi	$0.00
NA	$0.00
Red	$7,724,330.52
Silver	$0.00
White	$0.00
Yell...	$0.00
Total	**$7,724,330.52**



In both cases, the total is similar. However the table with a FILTER measure will automatically filter the data set, and only shows the result set for RED products, where the second table with IF measure, will show all products with

zero in front of all colors, except Red. These two are VERY different from the user point of view, while the final total value is similar. The reason is that IF apply a conditional expression on the result set, where FILTER works differently and filters the data set to the custom filter expression. Notice that we don't have any Visual, Report, or Page Level filter applied in this example. Filtering happened automatically because of the FILTER function.

If you bring the last method's result into a table (Sum of Red Products – Calculate Simple Expression), you will see the calculation happens on every row in the result set. It won't filter the data set, but the filter applies on calculating the final result for every row.

Color	Sum of Red Products - Calculate Simple Expression
Black	$7,724,330.52
Blue	$7,724,330.52
Grey	$7,724,330.52
Multi	$7,724,330.52
NA	$7,724,330.52
Red	$7,724,330.52
Silver	$7,724,330.52
Silver/Black	$7,724,330.52
White	$7,724,330.52
Yellow	$7,724,330.52
Total	**$7,724,330.52**

Which One to Use?

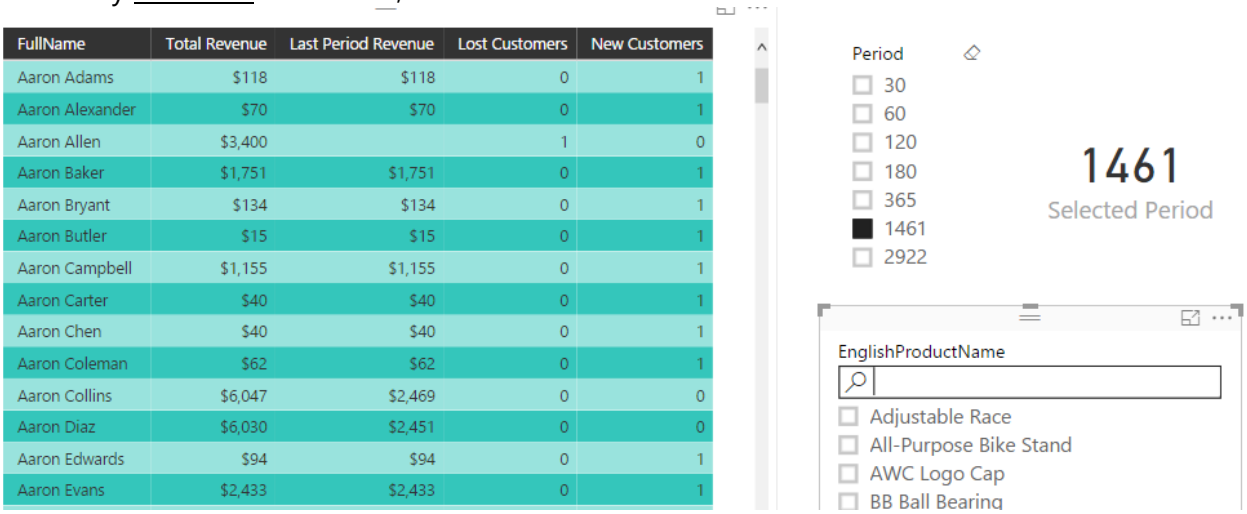
Like many other situations; It Depends! Are you only interested in the final result (sum of sales amount for "Red" products in this example)? If yes, then FILTER gives you the faster result, because it first filters the data set, and then apply SUM on it. However, if you want to show a detailed view of records, then

FILTER function will also filter the dataset, which might not be something you want, in those cases IF would give you the correct response. If you want to calculate the percentage for each row, the Calculate method might generate a more reliable result. Notice that there is nothing wrong about these functions, they are working exactly as they should. However, Not knowing their actual behavior might cause some confusion for you and your users. So use them wisely and don't overlook their differences.

Part IV: Real-world scenarios of DAX Expressions

Lost Customers DAX Calculation for Power BI

Posted by Reza Rad on Nov 22, 2016



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

One of the sample scenarios that DAX can be used as a calculation engine is customer retention. In the area of customer retention businesses might be interested to see who there lost customers or new customers are in the specific period. This can be calculated in the Power Query or the data source level, however, if we want to keep the period dynamic, then DAX is the great asset to do it. In this post, I'll explain how you can write a simple DAX calculation to find new customers and lost customers over a dynamic period in Power BI. If you are interested in learning more about Power BI read the Power BI online book; from Rookie to Rock Star.

Defining Terms

Before starting let's define terms that we use here. There might be different definitions for Lost or New Customers, the definition that I use is very simple for customer retention.

New Customer(s): Any customer who purchased any product in the last specified period considering the fact that the customer didn't purchase anything from this business before this period.

Lost Customer(s): Any customer who hasn't purchased any product in the last specified period considering the fact that he/she previously purchased from this business.
The method explained below through an example.

Prerequisite

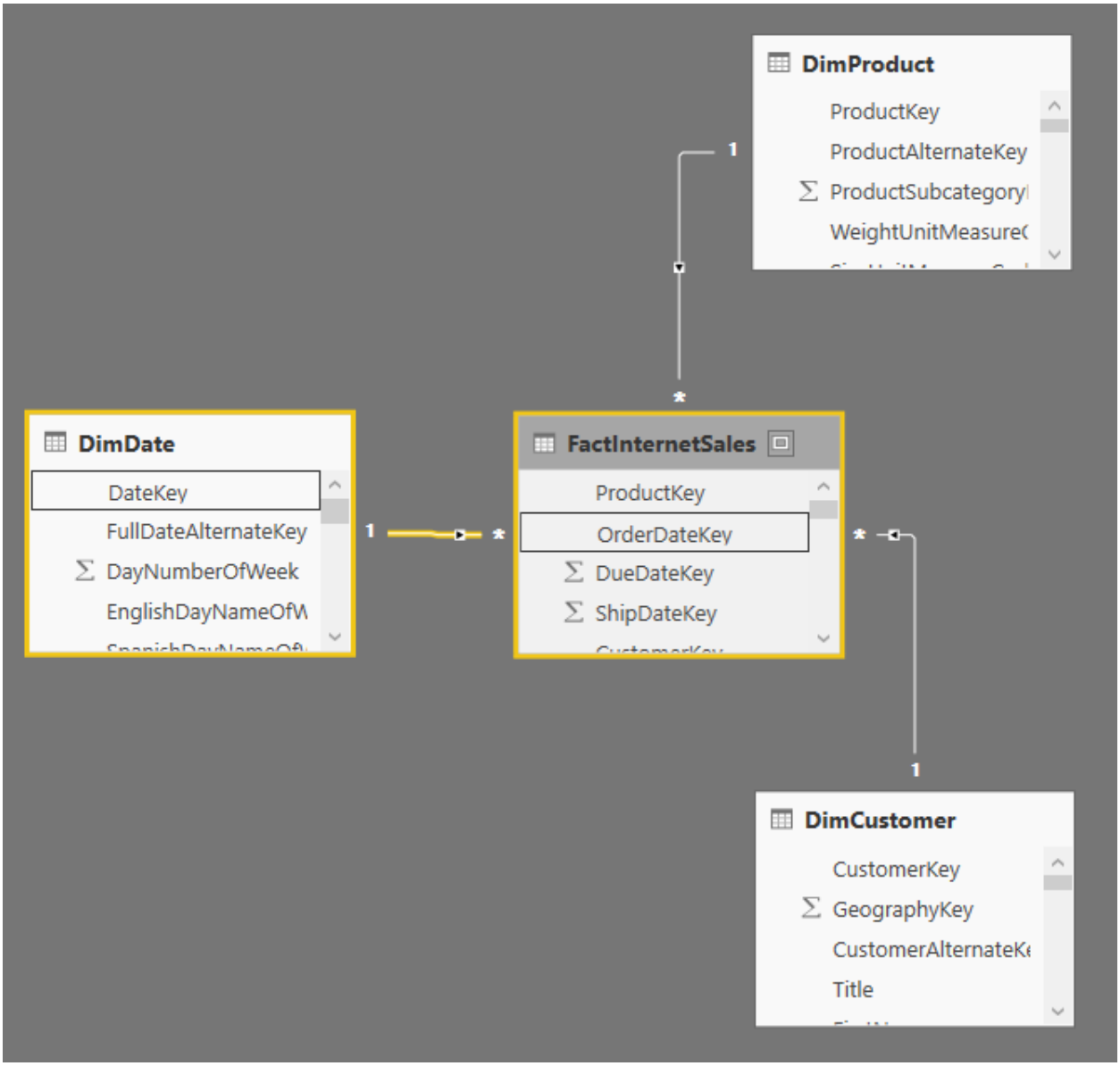
For running this example, you need to have AdventureWorksDW database or the Excel version of it.

Scenario

In the AdventureWorksDW, we want to find out who our new customers and also lost customers are. We want to have the flexibility to select the period (that determines if the customer is lost or new) in the slicer. We need these tables in AdventureWorksDW; FactInternetSales, DimDate, DimCustomer, and DimProduct.

The Model

Let's start by getting data from the AdventureWorksDW SQL Server database. Tables that we need for this model are FactInternetSales, DimDate, DimCustomer, and DimProduct. You don't need to apply any Power Query transformation. Just load the tables in the Power BI. and check the relationship to be as below. Note that the relationship between DimDate and FactInternetSales is only one active relationship based on OrderDateKey (remove other relationships between these two tables).



Period Slicer

Also, we need to have a Period Slicer. This is for being able to dynamically select the period (for example last 30 days, or last 365 days). You can create this table anywhere, in SQL Server DB. in Power Query, in Excel…. I have created that in my model itself through "Enter Data". Here is how the table looks like;

	Period
1	30
2	60
3	120
4	180
5	365
6	1461
7	2922

Table Name is Period, and the column name is also Period.

Method

What I do for the calculation is to calculate the total revenue from each customer first. This is the total revenue regardless of the period. Means sum of SalesAmount for everything that this customer bought for all times (I call this Total Revenue). Then I create another measure to calculate the sum of SalesAmount only for the selected period (in Days), Let's call this Last Period Revenue. Once we have these calculations finding new or lost costumers are easy as below;

Lost Customer(s): Customers that their Total Revenue is greater than zero, but their Last Period Revenue is not.

New Customer(s): Customers that their Total Revenue is equal to their Last Period Revenue and greater than zero.

The reason for greater than zero condition is that in the database we might have some customer records that haven't purchased anything at all (seems odd, but we have to consider it, as there might be some exceptions in some business). I do all of the steps one by one in a separate measure; this is to give you an understanding of the process. These all can be done in one measure.

To start the very first thing is to identify the selected value in the slicer through the measure.

Measure for the Selected Value in the Slicer: Selected Period

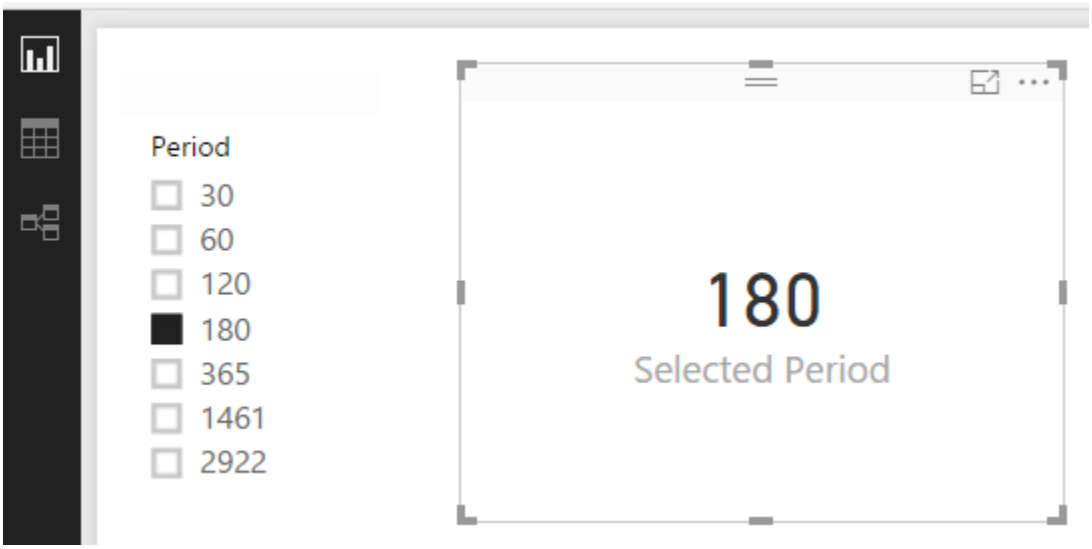
First things first are to identify what item is selected in the Period slicer. DAX doesn't understand the SELECTED item. However it understands the CONTEXT in which the DAX expression evaluates. This means; if I select one or more values in the slicer, DAX expression will execute on top of that selection. This selection is the filter context of DAX expression. That's simple. Now for this particular slicer, we want the user always to select one item, so it is also good to check if multiple items are selected or not. So let's create the **measure** (You can create a measure in the Data Tab, under Modeling, New Measure), named Selected Period under Period table with calculation below (Note that this should be a measure, not calculated column);

```
1 Selected Period = IF(COUNTROWS(Period)=1,
2                       MAX(Period[Period]),
3                                                   BLANK()
4                   )
```

The expression above just check if a number of records under the selected context (which will be selected slicer items) is one, then it will fetch the MAX of the period. Why maximum when it is only one row? Because DAX always works with a set of rows. Even though we know it will be one row after passing this condition, DAX still works with a set of rows. The set of rows might have one or more rows. When it has only one row, we can use MIN or MAX or things like that to fetch it as a single value from the set of rows. And finally, when multiple items are selected, it will return blank.

Let's see how the measure works in action;

Create a sample report with going to Report tab, and create a slicer with Period. Also, add a Card visualization with Selected Period measure (created above) as the value. Now if you select items in the slicer you will see what item is selected in the card visualization.



This Selected Period measure in the calculation of Sales Amount for the selected period. Which we will do it in the next steps.

Total Revenue

There are multiple ways of calculating the total revenue for each customer. I use a method that uses a new Measure. Create a new measure in the Data tab, under DimCustomer and name it Total Revenue. Write DAX expression below for it;

```
Total Revenue =
SUMX(RELATEDTABLE(FactInternetSales),FactInternetSales[SalesAmount])
```

This DAX expression uses SumX function which will calculate the sum of an expression (FactInternetSales[SalesAmount]) on a filtered table. The filtered table in this example is RelatedTable(FactInternetSales). RelatedTable will go through each Customer record in DimCustomer, and considering the existing relationship between DimCustomer and FactInternetSales in the model (As showed in the diagram earlier in this post), it will generate a subset of FactInternetSales for each customer, which is only the subset that has this customerKey in it. So the result of the sum of SalesAmount for this subset will be total revenue for each customer.

Let's see how this works in action;

Go back to the Report page, and create a Table with Full Name from DimCustomer, and also the new measure Total Revenue

FullName	Total Revenue
Aaron Adams	$118
Aaron Alexander	$70
Aaron Allen	$3,400
Aaron Baker	$1,751
Aaron Bryant	$134
Aaron Butler	$15
Aaron Campbell	$1,155
Aaron Carter	$40
Aaron Chen	$40
Aaron Coleman	$62
Aaron Collins	$6,047
Aaron Diaz	$6,030
Aaron Edwards	$94
Aaron Evans	$2,433
Aaron Flores	$1,539
Aaron Foster	$4,912
Aaron Gonzales	$1,810
Aaron Gonzalez	$133
Aaron Green	$27
Aaron Griffin	$72
Aaron Hall	$29
Aaron Hayes	$3,113
Aaron Henderson	$27
Aaron Hernandez	$94
Aaron Hill	$36
Aaron Hughes	$4,456
Aaron Jai	$575
Aaron Jenkins	$120
Aaron King	$4,758
Aaron Kumar	$2,049
Aaron Lal	$2,310
Aaron Li	$3,170
Total	**$29,358,677**

In front of each customer, you can see the total revenue for that customer. Now let's calculate the selected period's revenue.

Total Revenue for the Selected Period in the Slicer

Let's call this Last Period Revenue. In this last period revenue, all we want to calculate is the total sales for each customer in the period of last X days, where X is coming from the selection of period slicer. We can use a few functions to help along the way. Here is the measure with full DAX expression. I'll explain details after;

```
Last Period Revenue = CALCULATE(
                        SUM(FactInternetSales[SalesAmount]),
1                                                DATESBETWEEN(
2
3                       DimDate[FullDateAlternateKey],
4
5                       DATEADD(LASTDATE(DimDate[FullDateAlternateKey]),-
6 1*[Selected Period],DAY),
7
8                       LASTDATE(DimDate[FullDateAlternateKey])
                                                          )
                                                 )
```

Let's Start from Calculate; Calculation function is a simple function concerning the basics of how it works, and it is also the motherhood of all functions in DAX, there are heaps of calculations you can do with this function. How does this work? Calculate calculates an expression (first parameter) on a set of filters (second parameter, third parameter….). In the example above the expression is Sum(FactInternetSales[SalesAmount]), and the filter is all the parameters after that. In the above expression, you can see that we have only one filter which is DatesBetween. So calculate simply resolve the sum of SalesAmount for the subset of data that comes out of DatesBetween function. Let's see what DatesBetween Does;

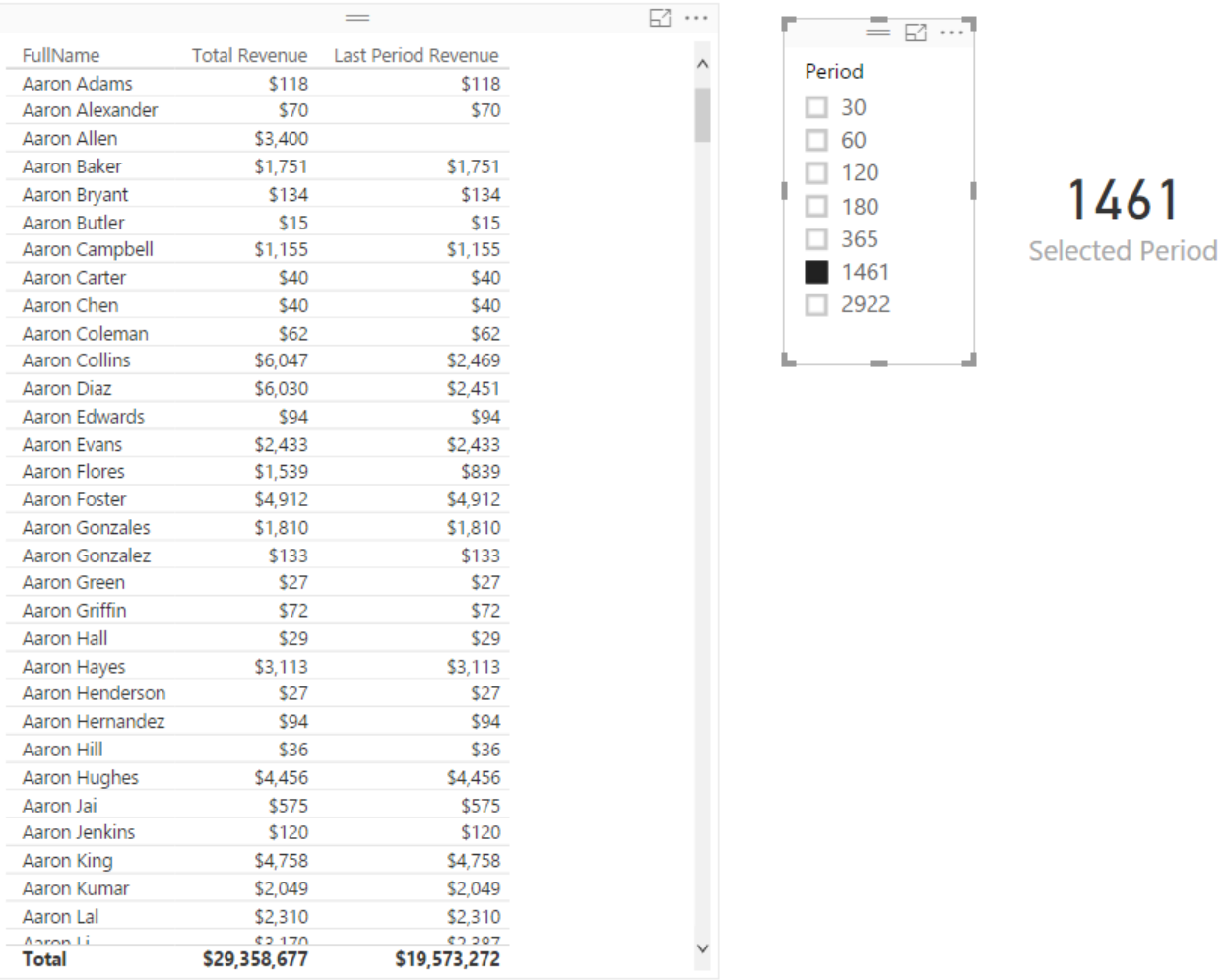
DatesBetween is a self explanatory function. It will return a subset of data for the dates between the start date and end date! It has three parameters; date field, start date, and end date;

```
DATESBETWEEN(
        DimDate[FullDateAlternateKey],
        DATEADD(LASTDATE(DimDate[FullDateAlternateKey]),-1*[Selected Period],DAY),
        LASTDATE(DimDate[FullDateAlternateKey])
                                                )
```

In example above end date is LASTDATE(DimDate[FullDateAlternateKey]). This means the maximum date that we have considering what filter context we are in. The main important thing is calculating the start date. Start date is the last date minus selected period. For doing this calculation, I used DATEADD function which reduces the number of days with (-1*[Selected Period]) in the highlighted line above in the expression.

As a result, this whole DAX expression returns total sales amount only for the selected period. Let's see how it works in the report;

Go back to the report and add the new measure to the table.



FullName	Total Revenue	Last Period Revenue
Aaron Adams	$118	$118
Aaron Alexander	$70	$70
Aaron Allen	$3,400	
Aaron Baker	$1,751	$1,751
Aaron Bryant	$134	$134
Aaron Butler	$15	$15
Aaron Campbell	$1,155	$1,155
Aaron Carter	$40	$40
Aaron Chen	$40	$40
Aaron Coleman	$62	$62
Aaron Collins	$6,047	$2,469
Aaron Diaz	$6,030	$2,451
Aaron Edwards	$94	$94
Aaron Evans	$2,433	$2,433
Aaron Flores	$1,539	$839
Aaron Foster	$4,912	$4,912
Aaron Gonzales	$1,810	$1,810
Aaron Gonzalez	$133	$133
Aaron Green	$27	$27
Aaron Griffin	$72	$72
Aaron Hall	$29	$29
Aaron Hayes	$3,113	$3,113
Aaron Henderson	$27	$27
Aaron Hernandez	$94	$94
Aaron Hill	$36	$36
Aaron Hughes	$4,456	$4,456
Aaron Jai	$575	$575
Aaron Jenkins	$120	$120
Aaron King	$4,758	$4,758
Aaron Kumar	$2,049	$2,049
Aaron Lal	$2,310	$2,310
Aaron Li	$3,170	$2,387
Total	**$29,358,677**	**$19,573,272**

Last Period Revenue by default when no period is selected in the slicer will be blank. When you select values, this will bring total revenue for that number of days from the last date in DimDate. In my sample data set, I can only start something if I select last 1461 days which is last four years. In a different version of AdventureWorksDW, you might get different results because their data is different. Now let's find what customers are lost.

Lost Customer(s) for the Selected Period

What table above says is that for a selected period, some customers have not purchased anything. For example, Aaron Allen has a total revenue of #3,400, but in the last 1461 days, he didn't purchase anything. So what this says is that

this customer is lost in the selected period! So Lost Customer(s) calculation is as simple as this:

```
1 Lost Customers = IF([Total Revenue]>0 && [Last Period Revenue]<=0
2                ,1
3                                ,0)
```

This is a simple IF expression which says if the customer has a Total Revenue greater than zero, and did not purchase anything in the last period (Last Period Revenue is less than or equal to zero), then this customer is lost.

New Customer(s) for the Selected Period

New customers, on the other hand, are customers who only purchased in the last period. This means their Total Revenue is equal to Last Period Revenue and greater than zero. Here is the calculation;

```
1 New Customers = IF([Total Revenue]>0 && [Last Period Revenue]=[Total
2 Revenue]
3                ,1
                                 ,0)
```

Let's see the final result in the report; Add Lost Customers and New Customers to the table, and you will see the result;

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
Aaron Flores	$1,539	$839	0	0
Aaron Foster	$4,912	$4,912	0	1
Aaron Gonzales	$1,810	$1,810	0	1
Aaron Gonzalez	$133	$133	0	1
Aaron Green	$27	$27	0	1
Aaron Griffin	$72	$72	0	1
Aaron Hall	$29	$29	0	1
Aaron Hayes	$3,113	$3,113	0	1
Aaron Henderson	$27	$27	0	1
Total	**$29,358,677**	**$19,573,272**	**0**	**0**

As you can see our formula successfully determined that Aaron Allen is lost in the last 1461 days period (if you increase or decrease the period you will see the different result). Also, those customers which their total revenue is equal to last period revenue are considered as New Customers (that's why you see so many new customers in the above table). Note that a customer might not fall in any of these categories. For example;

FullName	Total Revenue	Last Period Revenue	Lost Customers	New Customers
Aaron Kumar	$2,049	$2,049	0	1
Aaron Lal	$2,310	$2,310	0	1
Aaron Li	$3,170	$2,387	0	0
Aaron McDonald	$120	$120	0	1
Aaron Mitchell	$49	$49	0	1
Aaron Nelson	$2,882	$810	0	0
Aaron Patterson	$1,184	$1,184	0	1

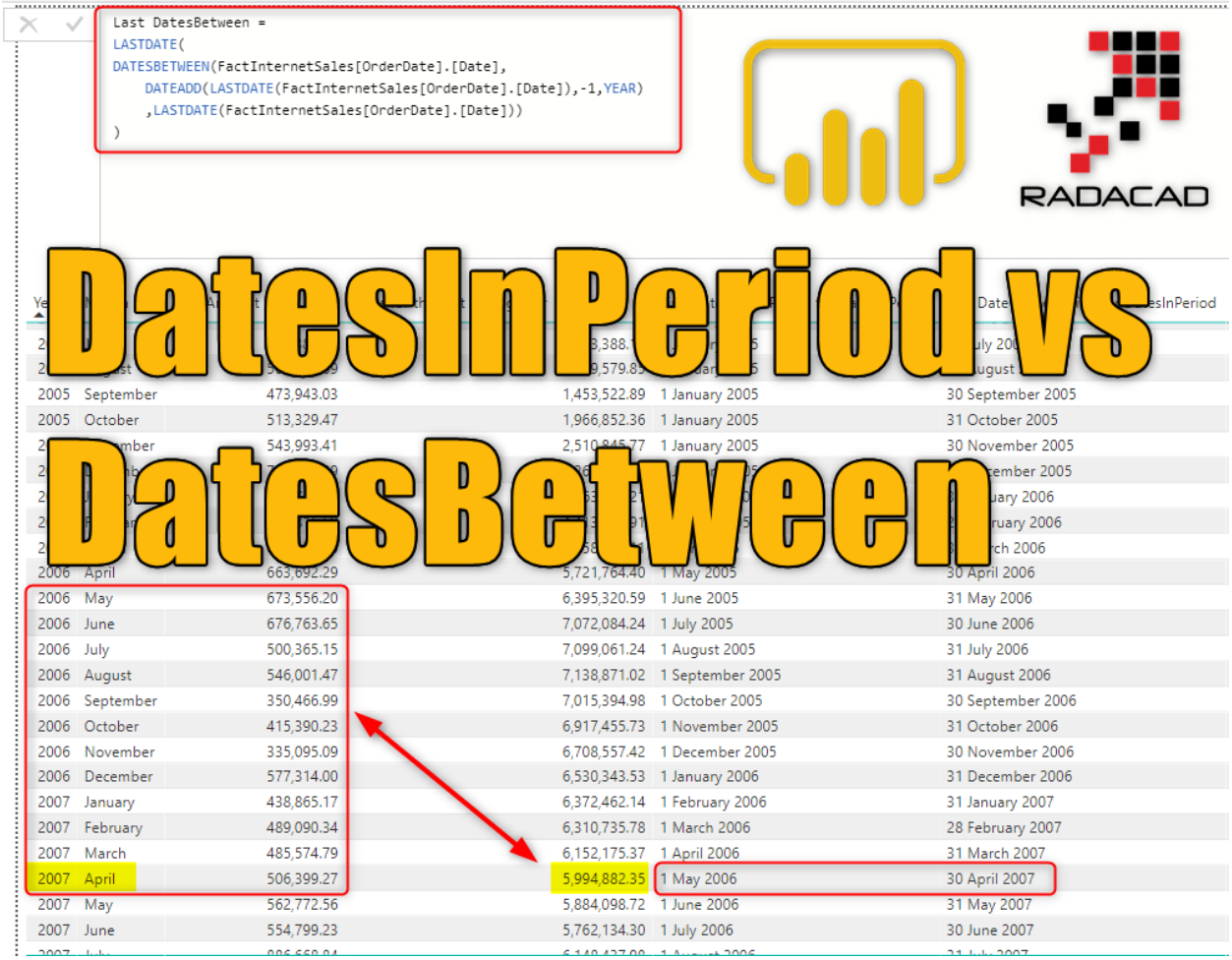
Aaron Li and Aaron Nelson are neither lost or new customers. They are customers who purchased before this period and still purchased in the last period as well.

Bringing Other Dimensions

These measures also work if you bring another table which is related to other tables in the model as a new dimension. For example, you can bring Product as another slicer and then you will see the product by product which customers are lost and which are new customers. Hope this calculation helps you to expand customer retention in your Power BI Model.

DatesInPeriod vs. DatesBetween; DAX Time Intelligence for Power BI

Posted by Reza Rad on Sep 11, 2018



2005	September	473,943.03	1,453,522.89	1 January 2005	30 September 2005
2005	October	513,329.47	1,966,852.36	1 January 2005	31 October 2005
2005	November	543,993.41	2,510,845.77	1 January 2005	30 November 2005
2006	April	663,692.29	5,721,764.40	1 May 2005	30 April 2006
2006	May	673,556.20	6,395,320.59	1 June 2005	31 May 2006
2006	June	676,763.65	7,072,084.24	1 July 2005	30 June 2006
2006	July	500,365.15	7,099,061.24	1 August 2005	31 July 2006
2006	August	546,001.47	7,138,871.02	1 September 2005	31 August 2006
2006	September	350,466.99	7,015,394.98	1 October 2005	30 September 2006
2006	October	415,390.23	6,917,455.73	1 November 2005	31 October 2006
2006	November	335,095.09	6,708,557.42	1 December 2005	30 November 2006
2006	December	577,314.00	6,530,343.53	1 January 2006	31 December 2006
2007	January	438,865.17	6,372,462.14	1 February 2006	31 January 2007
2007	February	489,090.34	6,310,735.78	1 March 2006	28 February 2007
2007	March	485,574.79	6,152,175.37	1 April 2006	31 March 2007
2007	April	506,399.27	5,994,882.35	1 May 2006	30 April 2007
2007	May	562,772.56	5,884,098.72	1 June 2006	31 May 2007
2007	June	554,799.23	5,762,134.30	1 July 2006	30 June 2007

There are many time intelligence functions in DAX, and each of those is helping in aspects of analyzing data on dates. There are two functions which work very similar to each other but have a bit different usage; DatesInPeriod, and DatesBetween. In this post, I will show you what the difference between these two functions, and scenarios that you can use each is. DatesBetween and DatesInPeriod both give you a period of dates, but let's see their main difference. If you like to learn more about Power BI; read Power BI book from Rookie to Rock Star.

Sample Dataset

For examples of this post, you need the FactInternetSales table from AdventureWorksDW example.

DatesInPeriod

The *DatesInPeriod* function in DAX will give you all dates within a period. The period can be one of these: Day, Month, Quarter, Year. Here is the syntax of using this function;

```
DATESINPERIOD(<dates>,<start_date>,<number_of_intervals>,<interval>
)
```

Here is a description of input parameters;

- <dates>: The date field (like many other time intelligence functions, this function also requires a date field)
- <start_date>: The start date that period starts from/to it (depends if the interval is a positive or negative number)
- <number_of_intervals>: a positive or negative number that starts from the start date based on the interval
- <interval>: Year, Quarter, Month, or Day intervals

The output of this function is a table of dates within the period specified. Let's see how this function can be used. For example, If you want to get all dates in the last year's period from the date of the filter context, it can be a calculation like this;

```
DATESINPERIOD(FactInternetSales[OrderDate].[Date],
    LASTDATE(FactInternetSales[OrderDate].[Date]),-1,YEAR)
```

Note that FactInternetSales[OrderDate] is just a normal date field in the FactInternetSales table and the reason that I used ".[Date]" at the end of it, is because I am using the built-in date dimension of Power BI. If you use your date dimension and have set it as a date table, then you should exclude the ".[Date]" part of this expression. To get the current filter context's date as the start date, I used the *LASTDATE()* DAX function, and we are going a Year back in the interval. So the number of intervals is **–**1. The expression above returns a

table, and cannot be used as a measure. To show you how this can work, I put it inside another function as a measure.

Example: Sales for the Last Rolling Year from the current Date

An example of using DatesInPeriod is to calculate the sales of the last year from the current date. In the expressions above, you've seen how we can get all dates in the last year from the current date in the filter context. We need to put it inside a Calculate statement to get Sum of Sales for that period.

```
1 Sales for the Last Rolling Year =
2 CALCULATE(
3     SUM(FactInternetSales[SalesAmount]),
4 DATESINPERIOD(
5     FactInternetSales[OrderDate].[Date],
6     LASTDATE(FactInternetSales[OrderDate].[Date]),
7     -1,
8     YEAR)
9 )
```

```
Sales for the Last Rolling Year =
CALCULATE(
    SUM(FactInternetSales[SalesAmount]),
DATESINPERIOD(
    FactInternetSales[OrderDate].[Date],
    LASTDATE(FactInternetSales[OrderDate].[Date]),
    -1,
    YEAR)
)
```

Year			
2005	Ja		
2005	Fe		
2005	M		
2005	Ap		
2005	M		
2005	Ju		
2005	Ju		
2005	August		
2005	September	473,943.03	1,453,522.89
2005	October	513,329.47	1,966,852.36
2005	November	543,993.41	2,510,845.77
2005	December	755,527.89	3,266,373.66
2006	January	596,746.56	3,863,120.21
2006	February	550,816.69	4,413,936.91
2006	March	644,135.20	5,058,072.11
2006	April	663,692.29	5,721,764.40
2006	May	673,556.20	6,395,320.59
2006	June	676,763.65	7,072,084.24
2006	July	500,365.15	7,099,061.24
2006	August	546,001.47	7,138,871.02
2006	September	350,466.99	7,015,394.98
2006	October	415,390.23	6,917,455.73
2006	November	335,095.09	6,708,557.42
2006	December	577,314.00	6,530,343.53
2007	January	438,865.17	6,372,462.14
2007	February	489,090.34	6,310,735.78
2007	March	485,574.79	6,152,175.37
2007	April	506,399.27	5,994,882.35
2007	May	562,772.56	5,884,098.72
2007	June	554,799.23	5,762,134.30
2007	July	886,668.84	6,148,437.98
2007	August	847,413.51	6,449,850.02
2007	September	1,010,258.13	7,109,641.16
2007	October		

What is the period of the calculation?

The important question in the above calculation is that what is the period of the calculation? Is this from the first of the year? Or is it starting from a different date? What is included and what is excluded? The answer is that; DatesInPeroid starts from the <start_date> (which in this case is the month in every row of the table visualized in the screenshot above), and it will go one year back (because the interval is the year, and the number of intervals is -1). For example; If the current month April 2007, then it will go one year back from that date. But does it mean it will start from April 2006, or May 2006? Well, DatesInBetween is a smart function and will exclude the start date to avoid double counting. It will start in May 2006. Let's see what the period start and period end is.

To get the period start and period end, you can create two measures below using *FIRSTDATE()* and *LASTDATE()* functions;

```
1 First Date in the Period for DatesInPeriod =
2 FIRSTDATE(
3 DATESINPERIOD(FactInternetSales[OrderDate].[Date],
4     LASTDATE(FactInternetSales[OrderDate].[Date]),-1,YEAR)
5 )
```

and for the last date;

```
1 Last Date in the Period for DatesInPeriod =
2 LASTDATE(
3 DATESINPERIOD(FactInternetSales[OrderDate].[Date],
4     LASTDATE(FactInternetSales[OrderDate].[Date]),-1,YEAR)
5 )
```

Now you can see the period clearly in Power BI;

```
Last DatesBetween =
LASTDATE(
DATESBETWEEN(FactInternetSales[OrderDate].[Date],
    DATEADD(LASTDATE(FactInternetSales[OrderDate].[Date]),-1,YEAR)
    ,LASTDATE(FactInternetSales[OrderDate].[Date]))
)
```

Year	Month	SalesAmount	Sales for the Last Rolling Year	First Date in the Period for DatesInPeriod	Last Date in the Period for DatesInPeriod
2005	July	473,388.16	473,388.16	1 January 2005	31 July 2005
2005	August	506,191.69	979,579.85	1 January 2005	31 August 2005
2005	September	473,943.03	1,453,522.89	1 January 2005	30 September 2005
2005	October	513,329.47	1,966,852.36	1 January 2005	31 October 2005
2005	November	543,993.41	2,510,845.77	1 January 2005	30 November 2005
2005	December	755,527.89	3,266,373.66	1 January 2005	31 December 2005
2006	January	596,746.56	3,863,120.21	1 February 2005	31 January 2006
2006	February	550,816.69	4,413,936.91	1 March 2005	28 February 2006
2006	March	644,135.20	5,058,072.11	1 April 2005	31 March 2006
2006	April	663,692.29	5,721,764.40	1 May 2005	30 April 2006
2006	May	673,556.20	6,395,320.59	1 June 2005	31 May 2006
2006	June	676,763.65	7,072,084.24	1 July 2005	30 June 2006
2006	July	500,365.15	7,099,061.24	1 August 2005	31 July 2006
2006	August	546,001.47	7,138,871.02	1 September 2005	31 August 2006
2006	September	350,466.99	7,015,394.98	1 October 2005	30 September 2006
2006	October	415,390.23	6,917,455.73	1 November 2005	31 October 2006
2006	November	335,095.09	6,708,557.42	1 December 2005	30 November 2006
2006	December	577,314.00	6,530,343.53	1 January 2006	31 December 2006
2007	January	438,865.17	6,372,462.14	1 February 2006	31 January 2007
2007	February	489,090.34	6,310,735.78	1 March 2006	28 February 2007
2007	March	485,574.79	6,152,175.37	1 April 2006	31 March 2007
2007	April	506,399.27	5,994,882.35	1 May 2006	30 April 2007
2007	May	562,772.56	5,884,098.72	1 June 2006	31 May 2007
2007	June	554,799.23	5,762,134.30	1 July 2006	30 June 2007

As you can see in the yellow highlighted section; for April 2007, the Rolling Last Year Sales is $5,994,882.35, which is for the period between the 1st of May 2006 to 30th of April 2007. As you can see it starts not from the 30th of April 2006 to avoid double counting. DatesInPeriod makes your like much easier to calculate dates in a period. That is why it is called DatesInPeriod! So the value of Rolling Last Year Sales is the accumulation of all sales from May 2006 to April 2007.

DatesInPeriod is perfect DAX function for calculating standard periods which follow Day, Month, Quarter, and Year intervals. It will exclude unnecessary dates for you.

DatesBetween

DatesBetween function in DAX is a more generic version of DatesInPeriod. You have more flexibility with this function. With this function, you do not need to worry about the interval or number of intervals. This function will give you all the dates between a start date and an end date. Here is the syntax of this function;

```
1 DATESBETWEEN(<dates>,<start_date>,<end_date>)
```

Parameters are:

- <dates>: The date field (like many other time intelligence functions, this function also requires a date field)
- <start_date>: The start date that period starts from it (unlike DatesInPeriod, this cannot go backward from the start date. It always go forward from there)
- <end_date>: The end date that period ends there.

The output of this function is a table of dates from the start_date to the end_date **including** both start and end date.
An important understanding of this function is that the function itself doesn't go back or forth from the start date to give you the period. You have to calculate the start or the end date first, and then get the period based on that. For example; Let's say we want to calculate dates in the last rolling year from the current date in the filter context (similar to the example we have done with DatesInPeriod). You need first to find out what your start date is.

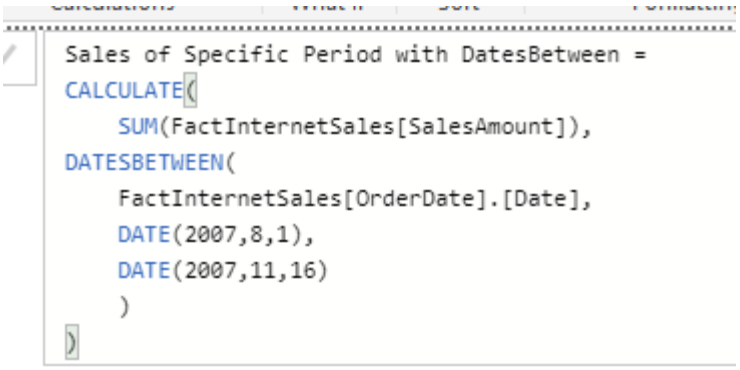
```
1 DATEADD(LASTDATE(FactInternetSales[OrderDate].[Date]),-1,YEAR)
```

The expression above is using *DATEADD()* function to calculate the start date which is going to be a year before (because the interval is -1) from the start date, which is calculated with LASTDATE().
After calculating the start date, you can use it inside a DatesBetween function like this;

```
1 DATESBETWEEN(FactInternetSales[OrderDate].[Date],
2     DATEADD(LASTDATE(FactInternetSales[OrderDate].[Date]),-1,YEAR)
3     ,LASTDATE(FactInternetSales[OrderDate].[Date])
4 )
```

The first parameter is just the date field. The second parameter is the start date that we have calculated, and the last parameter is the end date. DatesBetween is a good function to use when the start and end of the period are determined. Here is an example of calculating the sale of a specific period.

```
1 Sales of Specific Period with DatesBetween =
2 CALCULATE(
3     SUM(FactInternetSales[SalesAmount]),
4 DATESBETWEEN(
5     FactInternetSales[OrderDate].[Date],
6     DATE(2007,8,1),
7     DATE(2007,11,16)
8     )
9 )
```



DatesInPeriod vs DatesBetween

Now let's see if we use the DatesBetween for calculating the period and get the start and end of that period what we get as a result;

```
1 First DatesBetween =
2 FIRSTDATE(
3 DATESBETWEEN(
4     FactInternetSales[OrderDate].[Date],
5     DATEADD(LASTDATE(FactInternetSales[OrderDate].[Date]),-1,YEAR)
6     ,LASTDATE(FactInternetSales[OrderDate].[Date])
7     )
```

```
8 )
```

and the calculation for the end of the period;

```
1 Last DatesBetween =
2 LASTDATE(
3 DATESBETWEEN(
4     FactInternetSales[OrderDate].[Date],
5     DATEADD(LASTDATE(FactInternetSales[OrderDate].[Date]),-1,YEAR)
6     ,LASTDATE(FactInternetSales[OrderDate].[Date])
7     )
8 )
```

Here is the result compared to DatesInPeriod;

```
Last DatesBetween =
LASTDATE(
DATESBETWEEN(
    FactInternetSales[OrderDate].[Date],
    DATEADD(LASTDATE(FactInternetSales[OrderDate].[Date]),-1,YEAR)
    ,LASTDATE(FactInternetSales[OrderDate].[Date])
    )
)
```

Year	Month	SalesAmount	Sales for the Last Rolling Year	First Date in the Period for DatesInPeriod	Last Date in the Period for DatesInPeriod	First DatesBetween	Last DatesBetween
2006	June	676,763.65	7,072,084.24	1 July 2005	30 June 2006	30 June 2005	30 June 2006
2006	July	500,365.15	7,099,061.24	1 August 2005	31 July 2006	31 July 2005	31 July 2006
2006	August	546,001.47	7,138,871.02	1 September 2005	31 August 2006	31 August 2005	31 August 2006
2006	September	350,466.99	7,015,394.98	1 October 2005	30 September 2006	30 September 2005	30 September 2006
2006	October	415,390.23	6,917,455.73	1 November 2005	31 October 2006	31 October 2005	31 October 2006
2006	November	335,095.09	6,708,557.42	1 December 2005	30 November 2006	30 November 2005	30 November 2006
2006	December	577,314.00	6,530,343.53	1 January 2006	31 December 2006	31 December 2005	31 December 2006
2007	January	438,865.17	6,372,462.14	1 February 2006	31 January 2007	31 January 2006	31 January 2007
2007	February	489,090.34	6,310,735.78	1 March 2006	28 February 2007	28 February 2006	28 February 2007
2007	March	485,574.79	6,152,175.37	1 April 2006	31 March 2007	31 March 2006	31 March 2007
2007	April	506,399.27	5,994,882.35	1 May 2006	30 April 2007	30 April 2006	30 April 2007
2007	May	562,772.56	5,884,098.72	1 June 2006	31 May 2007	31 May 2006	31 May 2007
2007	June	554,799.23	5,762,134.30	1 July 2006	30 June 2007	30 June 2006	30 June 2007
2007	July	886,668.84	6,148,437.98	1 August 2006	31 July 2007	31 July 2006	31 July 2007
2007	August	847,413.51	6,449,850.02	1 September 2006	31 August 2007	31 August 2006	31 August 2007
2007	September	1,010,258.13	7,109,641.16	1 October 2006	30 September 2007	30 September 2006	30 September 2007
2007	October	1,080,449.58	7,774,700.51	1 November 2006	31 October 2007	31 October 2006	31 October 2007
2007	November	1,196,981.11	8,636,586.53	1 December 2006	30 November 2007	30 November 2006	30 November 2007
2007	December	1,731,787.77	9,791,060.30	1 January 2007	31 December 2007	31 December 2006	31 December 2007
2008	January	1,340,244.95	10,692,440.08	1 February 2007	31 January 2008	31 January 2007	31 January 2008
2008	February	1,462,479.83	11,665,829.57	1 March 2007	29 February 2008	28 February 2007	29 February 2008
2008	March	1,480,905.18	12,661,159.96	1 April 2007	31 March 2008	31 March 2007	31 March 2008
2008	April	1,608,750.53	13,763,511.22	1 May 2007	30 April 2008	30 April 2007	30 April 2008
2008	May	1,878,317.51	15,079,056.17	1 June 2007	31 May 2008	31 May 2007	31 May 2008
2008	June	1,949,361.11	16,473,618.05	1 July 2007	30 June 2008	30 June 2007	30 June 2008

As you can see in the above screenshot, the output of DatesBetween INCLUDES both start and end date, it will start from 30th of April 2006, while the DatesInPeriod starts from 1st of My 2006. so the first difference between these two functions is that one of the is inclusive of both dates (DatesBetween).

DatesBetween is a period of dates inclusive of both start and end date. DatesInPeriod is giving you the period of dates and excluding unwanted dates.

Another difference between these two is the input parameters that you have. Sometimes, you have the start and end date, and you want to get all dates in that period, DatesBetween is a good function to use in this situation. Sometimes, you do not have both ends of the period, you have one, and the interval, in that case, DatesInPeriod is your best friend. There are many scenarios that you can use DatesBetween and DatesInPeriod instead of the other one, here is an example that I wrote a previous dynamic period calculation with DatesBetween.

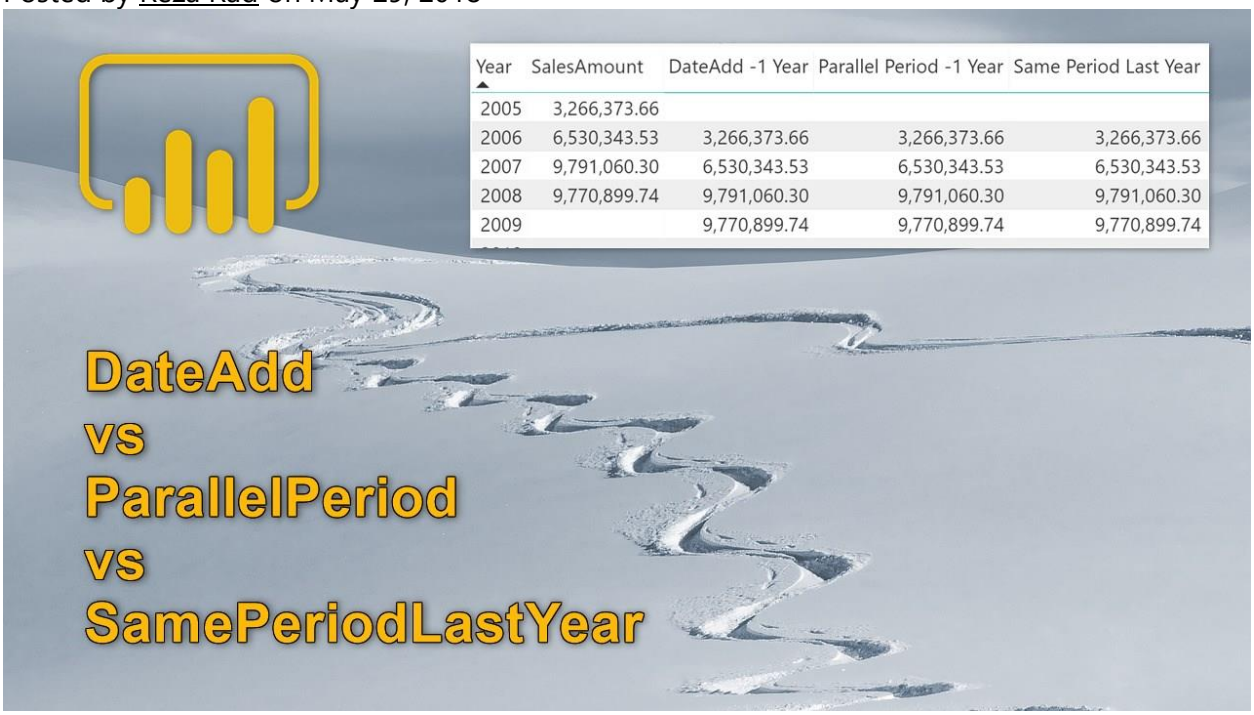
If you have the start and end date, and you want to get all dates in that period, DatesBetween is a good function to use. However, Sometimes, you do not have both ends of the period, you have one, and the interval, in that case, DatesInPeriod is your best friend

Summary

DatesBetween and DatesInPeriod are DAX functions to give you a period of dates. DatesBetween gives you dates from a start date to an end date. DatesInPeriod will give you an interval of dates from a particular period. Each function has its usages; you can tweak and change your expressions with each of these functions to get the same result as the other function (like anything else in DAX!). However, these two functions will give you good power in different situations of calculating a period.

DateAdd vs ParallelPeriod vs SamePeriodLastYear; DAX Time Intelligence Question

Posted by Reza Rad on May 29, 2018



Using DAX time intelligence functions for a while; you may ask this question from yourself that what is the difference between functions below;

- SamePeriodLastYear function vs. using ParallelPeriod with Year parameter
- ParallelPeriod for a month vs. DateAdd for a month ago
- and many other questions that lead to this final question: Which function should be used in which situation?

Let's take a look at these questions and their responses in more details through this post. If you want to learn more about Power BI: read Power BI book from Rookie to Rock Star.

SamePeriodLastYear

Let's start with the SamePeriodLastYear function; this function will give you exactly what it explains; same PERIOD but last year! Same period; means if you

are looking at data on the day level, it would be the same day last year. If you are slicing and dicing in a month or quarter level; this would give you the same month or quarter last year. You can use the function simply just by providing a date field:

SamePeriodLastYear(<date field>)

the image below shows how the SamePeriodLastYear works for Date

```
SamePeriodLastYear D To =
LASTDATE(
SAMEPERIODLASTYEAR(DimDate[FullDateAlternateKey].[Date])
)
```

Year	Quarter	Month	Day	SamePeriodLastYear D From	SamePeriodLastYear D
2006	Qtr 1	January	1	January 1, 2005	January 1, 2005
2006	Qtr 1	January	2	January 2, 2005	January 2, 2005
2006	Qtr 1	January	3	January 3, 2005	January 3, 2005
2006	Qtr 1	January	4	January 4, 2005	January 4, 2005
2006	Qtr 1	January	5	January 5, 2005	January 5, 2005
2006	Qtr 1	January	6	January 6, 2005	January 6, 2005
2006	Qtr 1	January	7	January 7, 2005	January 7, 2005
2006	Qtr 1	January	8	January 8, 2005	January 8, 2005
2006	Qtr 1	January	9	January 9, 2005	January 9, 2005
2006	Qtr 1	January	10	January 10, 2005	January 10, 2005
2006	Qtr 1	January	11	January 11, 2005	January 11, 2005

The SamePeriodLastYear function like many other time intelligence functions needs a date field to work. This is how you can get this function working:

1 SAMEPERIODLASTYEAR(DimDate[FullDateAlternateKey].[Date])

The code above returns a table with one single column: date. This is not returning one single value. Means you cannot use it directly in a measure. You have to use this function as a filter function. In the screenshot above; I have

used the SamePeriodLastYear inside a LastDate, and also a FirstDate to get the range of dates for each filter context selection.

```
1 SamePeriodLastYear D To =
2 LASTDATE(
3 SAMEPERIODLASTYEAR(DimDate[FullDateAlternateKey].[Date])
4 )
```

As you can see in the screenshot above; it shows that the SamePeriodLastYear returns the same date last year when your filter context is in day level. If your filter context is at month level; then you'll get the same month last year. The screenshot below shows it;

Year	Quarter	Month	SamePeriodLastYear D From	SamePeriodLastYear D To
2006	Qtr 1	January	January 1, 2005	January 31, 2005
2006	Qtr 1	February	February 1, 2005	February 28, 2005
2006	Qtr 1	March	March 1, 2005	March 31, 2005
2006	Qtr 2	April	April 1, 2005	April 30, 2005
2006	Qtr 2	May	May 1, 2005	May 31, 2005
2006	Qtr 2	June	June 1, 2005	June 30, 2005
2006	Qtr 3	July	July 1, 2005	July 31, 2005
2006	Qtr 3	August	August 1, 2005	August 31, 2005
2006	Qtr 3	September	September 1, 2005	September 30, 2005
2006	Qtr 4	October	October 1, 2005	October 31, 2005
2006	Qtr 4	November	November 1, 2005	November 30, 2005
2006	Qtr 4	December	December 1, 2005	December 31, 2005

For example; for September 2006, SamePeriodLastYear returns September 2005.

SamePeriodLastYear returns the equivalent period to the filter context from last year. For 1st of Sep 2006, it will return date period of 1st of Sep 2005. For Q4 of 2006, it will return Q4 of 2005. If filter context is in DAY level; it will return the same DAY last year, if the filter context is in Month level, it will return same Month last year.

SamePeriodLastYear function when used in a real-world scenario it will act as a filter, and you can get the Sales of the same period last year with that using an expression like this:

```
Same Period Last Year = CALCULATE(
    SUM(FactInternetSales[SalesAmount]),
    SAMEPERIODLASTYEAR(
        DimDate[FullDateAlternateKey].[Date])
)
```

```
Same Period Last Year = CALCULATE(
    SUM(FactInternetSales[SalesAmount]),
    SAMEPERIODLASTYEAR(
        DimDate[FullDateAlternateKey].[Date]))
```

Year				
2005	Qtr 3	July	473,388.16	
2005	Qtr 3	August	506,191.69	
2005	Qtr 3	September	473,943.03	
2005	Qtr 4	October	513,329.47	
2005	Qtr 4	November	543,993.41	
2005	Qtr 4	December	755,527.89	
2006	Qtr 1	January	596,746.56	
2006	Qtr 1	February	550,816.69	
2006	Qtr 1	March	644,135.20	
2006	Qtr 2	April	663,692.29	
2006	Qtr 2	May	673,556.20	
2006	Qtr 2	June	676,763.65	
2006	Qtr 3	July	500,365.15	473,388.16
2006	Qtr 3	August	546,001.47	506,191.69
2006	Qtr 3	September	350,466.99	473,943.03
2006	Qtr 4	October	415,390.23	513,329.47
2006	Qtr 4	November	335,095.09	543,993.41

ParallelPeriod

ParallelPeriod is another function that gives you the ability to get the parallel period to the current period. You can navigate to periods in the past or future. You need three parameters for this function:

ParllelPeriod(<date field>, <number of intervals>, <interval>)

You can choose the interval to be Month, Quarter, or Year. And the number of intervals can be negative (to go to past), or positive (to go to the future). This is an example of using ParallelPeriod:

```
Parallel Period D M From =
FIRSTDATE(
PARALLELPERIOD(DimDate[FullDateAlternateKey].[Date], -1, MONTH )
)
```

Ye			
2005 Qtr 1	February	January 1, 2005	January 31, 2005
2005 Qtr 1	March	February 1, 2005	February 28, 2005
2005 Qtr 2	April	March 1, 2005	March 31, 2005
2005 Qtr 2	May	April 1, 2005	April 30, 2005
2005 Qtr 2	June	May 1, 2005	May 31, 2005
2005 Qtr 3	July	June 1, 2005	June 30, 2005
2005 Qtr 3	August	July 1, 2005	July 31, 2005
2005 Qtr 3	September	August 1, 2005	August 31, 2005
2005 Qtr 4	October	September 1, 2005	September 30, 2005
2005 Qtr 4	November	October 1, 2005	October 31, 2005
2005 Qtr 4	December	November 1, 2005	November 30, 2005
2006 Qtr 1	January	December 1, 2005	December 31, 2005

For every month, the ParallelPeriod expression will return a month before that, because in the parameters, we mentioned the month before:

1 PARALLELPERIOD(DimDate[FullDateAlternateKey].[Date], -1, MONTH)

ParallelPeriod can be used to fetch the Sales of last month like this:

```
1 Parallel Period -1 Month = CALCULATE(
2     SUM(FactInternetSales[SalesAmount]),
3     PARALLELPERIOD(
4         DimDate[FullDateAlternateKey].[Date],
5         -1,
6         MONTH)
7 )
```

```
Parallel Period -1 Month = CALCULATE(
    SUM(FactInternetSales[SalesAmount]),
    PARALLELPERIOD(
        DimDate[FullDateAlternateKey].[Date],
        -1,
        MONTH))
```

2005	Qtr 3	July	29	17,688.07	
2005	Qtr 3	July	30	14,402.34	
2005	Qtr 3	July	31	15,012.18	
2005	Qtr 3	August	1	17,891.35	473,388.16
2005	Qtr 3	August	2	10,734.81	473,388.16
2005	Qtr 3	August	3	11,230.63	473,388.16
2005	Qtr 3	August	4	17,534.79	473,388.16
2005	Qtr 3	August	5	15,711.28	473,388.16
2005	Qtr 3	August	6	25,390.43	473,388.16
2005	Qtr 3	August	7	14,313.08	473,388.16
2005	Qtr 3	August	8	11,255.63	473,388.16
2005	Qtr 3	August	9	13,906.52	473,388.16
2005	Qtr 3	August	10	21,088.06	473,388.16
2005	Qtr 3	August	11	17,891.35	473,388.16
2005	Qtr 3	August	12	21,469.62	473,388.16

As you can see in the above screenshot; ParallelPeriod will return sales of the entire last month, even if you are looking at the day level. This brings us to an important conclusion:

ParallelPeriod gives the result of a period parallel to this period (in the past or future), which is statically determined in the Interval parameter; Can be Month, Quarter, or Year.

Understanding this fact; now we can answer this question:

What is the difference between SamePeriodLastYear and ParallelPeriod?

The first difference is that ParallelPeriod gives you the option to go as many as intervals you want back or forward. If you want to get the sales for last months; then ParallelPeriod is your friend. For calculating the sales of 2 years ago, then ParallelPeriod is your friend.

Dynamic Period is another difference between these two functions; If you think that the result of SamePeriodLastYear and the ParallelPeriod (when it is used with Year interval) are the same, continue reading. Below is an example of these two measures:

```
Parallel Period -1 Year = CALCULATE(
    SUM(FactInternetSales[SalesAmount]),
    PARALLELPERIOD(
        DimDate[FullDateAlternateKey].[Date],
        -1,
        YEAR))
```

Year	Quarter	Month	SalesAmount	Same Period Last Year	Parallel Period -1 Year
2005	Qtr 3	July	473,388.16		
2005	Qtr 3	August	506,191.69		
2005	Qtr 3	September	473,943.03		
2005	Qtr 4	October	513,329.47		
2005	Qtr 4	November	543,993.41		
2005	Qtr 4	December	755,527.89		
2006	Qtr 1	Janu			3,266,373.66
2006	Qtr 1	Febr			3,266,373.66
2006	Qtr 1	Marc			3,266,373.66
2006	Qtr 2	April			3,266,373.66
2006	Qtr 2	May	673,556.20		3,266,373.66
2006	Qtr 2	June	676,763.65		3,266,373.66
2006	Qtr 3	July	500,365.15	473,388.16	3,266,373.66
2006	Qtr 3	August	546,001.47	506,191.69	3,266,373.66
2006	Qtr 3	September	350,466.99	473,943.03	3,266,373.66

ParallelPeriod with Year Interval returns the sales for the ENTIRE Last Year: 2005

SamePeriodLastYear returns the sales for the SAME PERIOD last year: Aug 2005

For August 2006 for example; the SamePeriodLastYear gives us the sales of August 2005. However, the ParallelPeriod with year interval returns the sales for the entire year 2005.

DateAdd

DateAdd is a function that adds or subtracts a number of days/months/quarters/years from or to a date field. DateAdd can be used like this:

DateAdd(<date field>, <number of intervals>, <interval>)

DateAdd used in an example below to return the period for a month ago.

```
DateAdd D To =
LASTDATE(
DATEADD(DimDate[FullDateAlternateKey].[Date],-1,MONTH)
)
```

Ye				
2005	Qtr 1	February	January 1, 2005	January 31, 2005
2005	Qtr 1	March	February 1, 2005	February 28, 2005
2005	Qtr 2	April	March 1, 2005	March 31, 2005
2005	Qtr 2	May	April 1, 2005	April 30, 2005
2005	Qtr 2	June	May 1, 2005	May 31, 2005
2005	Qtr 3	July	June 1, 2005	June 30, 2005
2005	Qtr 3	August	July 1, 2005	July 31, 2005
2005	Qtr 3	September	August 1, 2005	August 31, 2005
2005	Qtr 4	October	September 1, 2005	September 30, 2005
2005	Qtr 4	November	October 1, 2005	October 31, 2005
2005	Qtr 4	December	November 1, 2005	November 30, 2005
2006	Qtr 1	January	December 1, 2005	December 31, 2005
2006	Qtr 1	February	January 1, 2006	January 31, 2006

DateAdd can be used in a Day level too. This brings us to the first difference of ParallelPeriod and DateAdd;

DateAdd can work on an interval of DAY, Month, Quarter, or Year, but ParallelPeriod only works on intervals of Month, Quarter, and Year.

This is the example expression to calculate the sales for yesterday:

```
1 DateAdd -1 Day = CALCULATE(
2     SUM(FactInternetSales[SalesAmount]),
3     DATEADD(
4         DimDate[FullDateAlternateKey].[Date],
```

```
5       -1,
6       DAY))
```

DateAdd vs ParallelPeriod

Comparing these two functions with each other; you can see that DateAdd works on the period dynamically (like SamePeriodLastYear), but the ParallelPeriod works statically on the interval mentioned as the parameter.

```
DateAdd -1 Year = CALCULATE(
    SUM(FactInternetSales[SalesAmount]),
    DATEADD(
        DimDate[FullDateAlternateKey].[Date],
        -1,
        YEAR))
```

Year	Quarter	Month	SalesAmount	Parallel Period -1 Year	DateAdd -1 Year
2005	Qtr 3	July	473,388.16		
2005	Qtr 3	August	506,191.69		
2005	Qtr 3	September	473,943.03		
2005	Qtr 4	October	513,329.47		
2005	Qtr 4	November	543,993.41		
2005	Qtr 4	December	755,527.89		
2006	Qtr 1	January	596,746.56	3,266,373.66	
2006	Qtr 1	February	550,816.69	3,266,373.66	
2006	Qtr 1	March	644,135.20	3,266,373.66	
2006	Qtr 2	April	663,692.29	3,266,373.66	
2006	Qtr 2	May	673,556.20	3,266,373.66	
2006	Qtr 2	June	676,763.65	3,266,373.66	
2006	Qtr 3	July	500,365.15	3,266,373.66	473,388.16
2006	Qtr 3	August	546,001.47	3,266,373.66	506,191.69
2006	Qtr 3	September	350,466.99	3,266,373.66	473,943.03
2006	Qtr 4	October	415,390.23	3,266,373.66	513,329.47
2006	Qtr 4	November	335,095.09	3,266,373.66	543,993.41

That leads us to the conclusion that DateAdd(<date field>,-1, Year) is similar to SamePeriodLastYear. However, one difference is still there:

DateAdd vs SamePeriodLastYear

SamePeriodLastYear only goes one year back, DateAdd can go two years back or even more. DateAdd is a customized version of SamePeriodLastYear.

```
DateAdd -2 Year = CALCULATE(
    SUM(FactInternetSales[SalesAmount]),
    DATEADD(
        DimDate[FullDateAlternateKey].[Date],
        -2,
        YEAR))
```

Year	Quarter	Month	SalesAmount	DateAdd -1 Year	Same Period Last Year	DateAdd -2 Year
2006	Qtr 3	August	546,001.47	506,191.69	506,191.69	
2006	Qtr 3	September	350,466.99	473,943.03	473,943.03	
2006	Qtr 4	October	415,390.23	513,329.47	513,329.47	
2006	Qtr 4	November	335,095.09	543,993.41	543,993.41	
2006	Qtr 4	December	577,314.00	755,527.89	755,527.89	
2007	Qtr 1	January	438,865.17	596,746.56	596,746.56	
2007	Qtr 1	February	489,090.34	550,816.69	550,816.69	
2007	Qtr 1	March	485,57			
2007	Qtr 2	April	506,39			
2007	Qtr 2	May	562,77			
2007	Qtr 2	June	554,79			
2007	Qtr 3	July	886,668.84	500,365.16	500,365.16	473,388.16

DateAdd can go more than 1 year back or forward, but SamePeriodLastYear only goes one year back

Conclusion

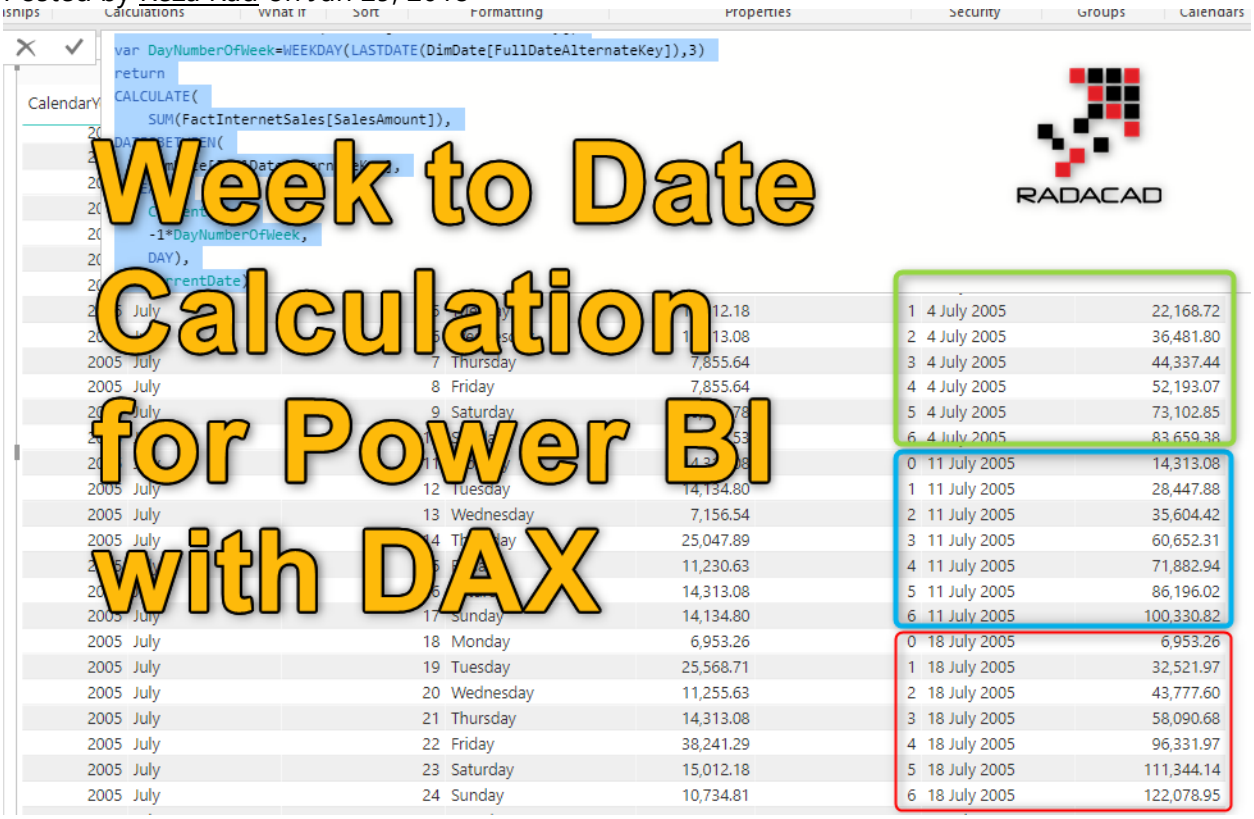
In summary, there are differences between these three functions:

- DateAdd and SamePeriodLastYear both work based on the DYNAMIC period in the filter context
- ParallelPeriod is working STATICALLY based on the interval selected in the parameter
- ParallelPeriod and DateAdd can go more than one interval back and forward, while SamePeriodLastYear only goes one year back.

- DateAdd works on the interval of DAY, as well as the month, quarter and year, but ParallelPeriod only works on month, quarter, and year.
- Depends on the filter context you may get a different result from these functions. If you get the same result in a year level context, it doesn't mean that all these functions are the same! Look more into the detailed context.

Week to Date Calculation in Power BI with DAX

Posted by Reza Rad on Jun 29, 2018



There are some predefined DAX time intelligence calculations that help you to get analytics over time, such as year to date, same period last year, etc. However, there is no calculation for Week to Date built-in. I have found it quite a demand for some of the businesses, as many of businesses work on a weekly period rather than monthly. So here in this post; I will explain a method to do week to date calculation with DAX. If you want to learn more about Power BI, read Power BI book from Rookie to Rock Star.

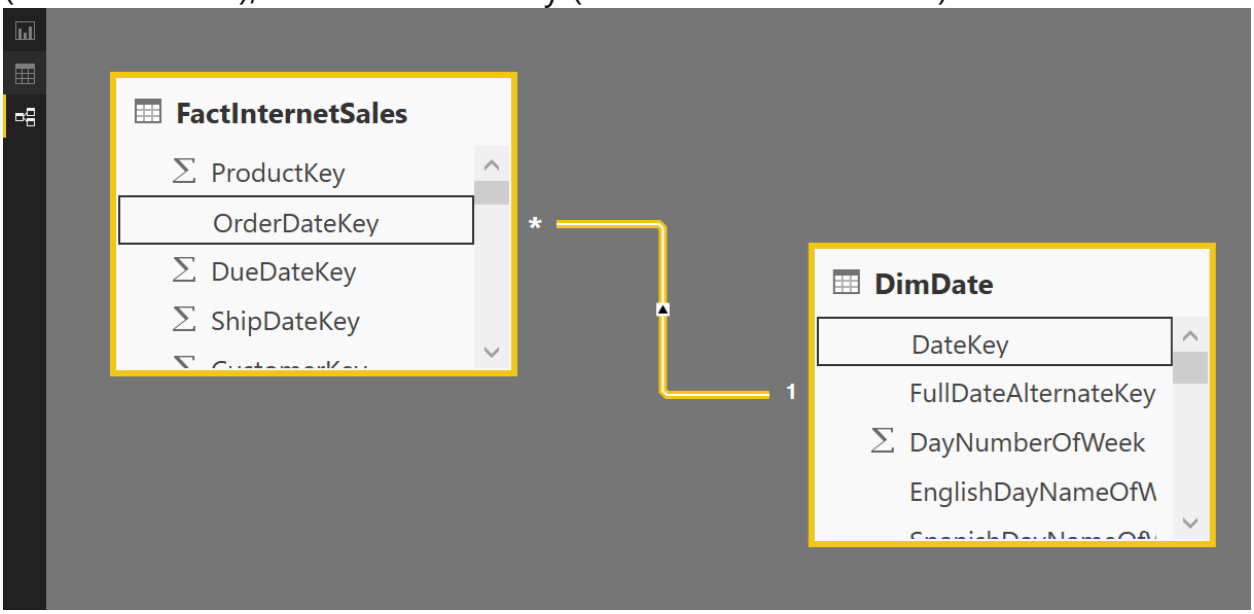
Introduction

There is a set of functions for calculating Year to Date (TotalYTD), Quarter to Date (TotalQTD), and Month to Date (TotalMTD). However, for calculating Week to Date; there is no built-in function. There are many ways to calculate

the week to date. One of the methods is using functions such as DatesBetween and WeekDay to calculate the period between the first day of the week and the date of the filter context. Let's see how it works.

Sample Dataset

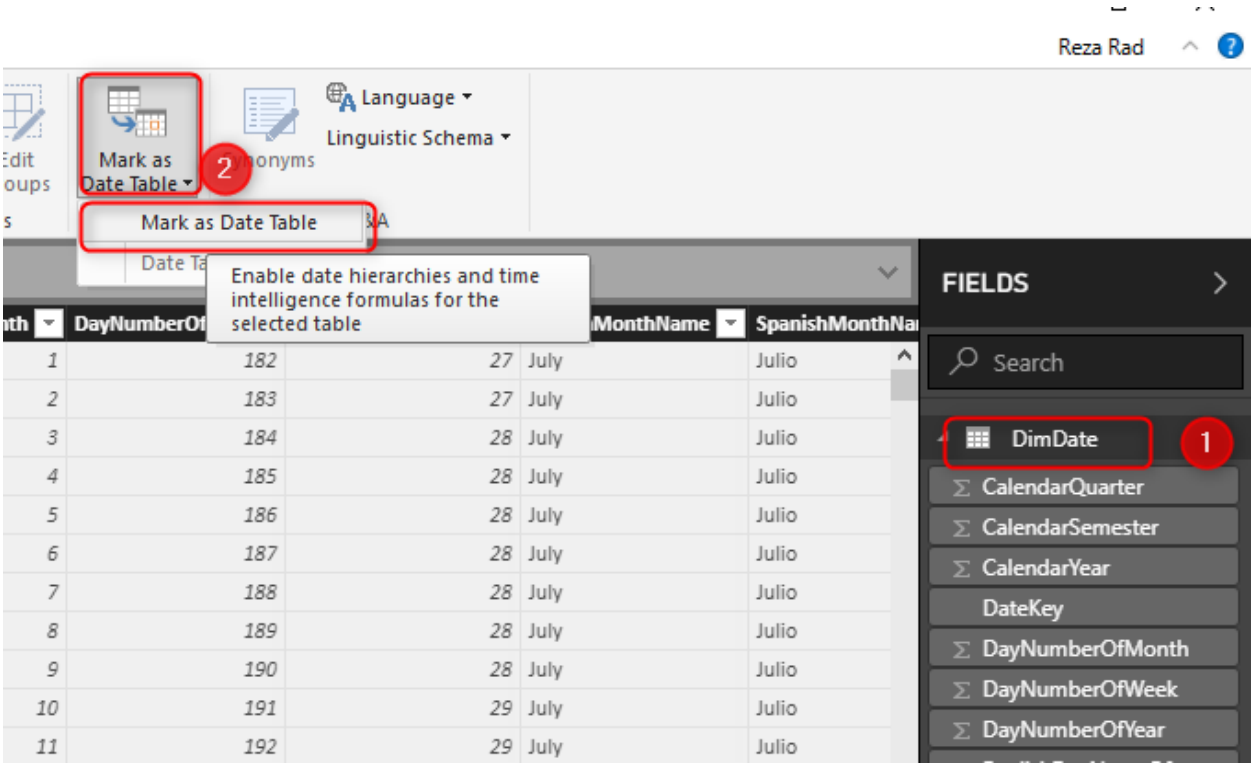
If you want to use this example; create a Power BI file connected to AdventureWorks data source and load FactInternetSales, and DimDate into the model. Create a connection between these two tables based on DateKey (from DimDate), and OrderDateKey (from FactInternetSales).



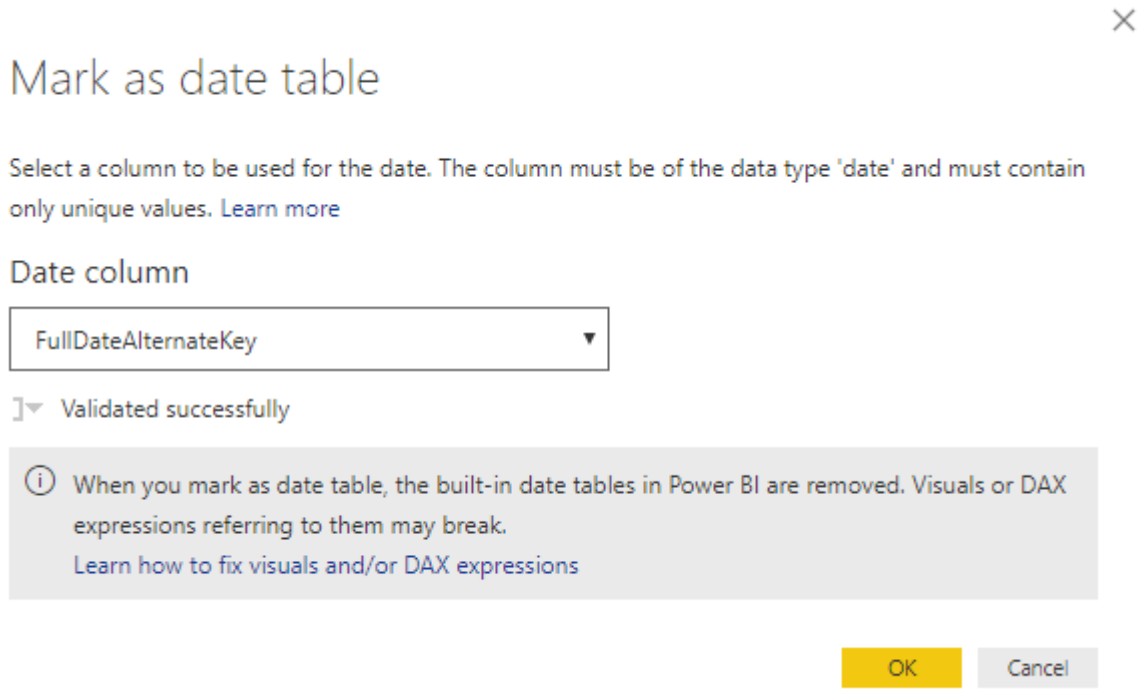
Mark As Date Table

** for a week to date calculation to work; it is not mandatory to have a date table, you can even use the built-in date table of Power BI. all below calculations would work even if you use the built-in date table, you need to add.[date] at the end of your date field to get it working.*

Because in this example; we are going to use DimDate as our date table, select this table in the list of fields (in report tab, or data tab), and mark it as date table.



Select FullDateAlternateKey as the date column.



Create a table visual with CalendarYear, EnglishMonthName, DayNumberOfMonth, and EnglishDayNameOfWeek, and SalesAmount

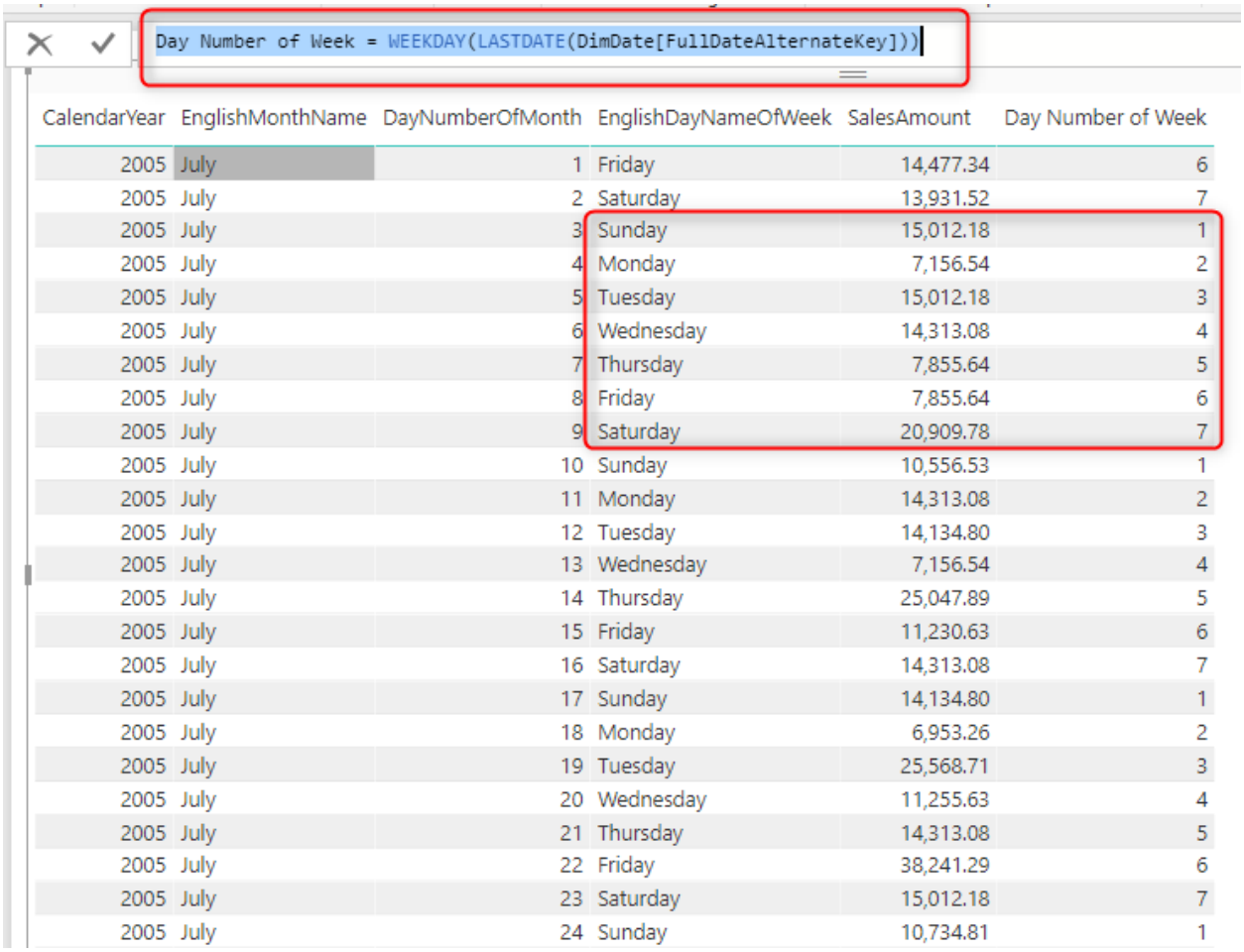
CalendarYear	EnglishMonthName	DayNumberOfMonth	EnglishDayNameOfWeek	SalesAmount
2005	July	1	Friday	14,477.34
2005	July	2	Saturday	13,931.52
2005	July	3	Sunday	15,012.18
2005	July	4	Monday	7,156.54
2005	July	5	Tuesday	15,012.18
2005	July	6	Wednesday	14,313.08
2005	July	7	Thursday	7,855.64
2005	July	8	Friday	7,855.64
2005	July	9	Saturday	20,909.78
2005	July	10	Sunday	10,556.53
2005	July	11	Monday	14,313.08
2005	July	12	Tuesday	14,134.80
2005	July	13	Wednesday	7,156.54
2005	July	14	Thursday	25,047.89
2005	July	15	Friday	11,230.63
2005	July	16	Saturday	14,313.08
2005	July	17	Sunday	14,134.80
2005	July	18	Monday	6,953.26
2005	July	19	Tuesday	25,568.71
2005	July	20	Wednesday	11,255.63

WeekDay DAX Function

To start the solution; let's first find out what is the day number of the week for any given date. We already have the day number of week in the DimDate provided by AdventureWorks. However, we calculate it again, just in case you use the built-in date table. Using LastDate function, we get the date value of the current filter context and wrap it inside a WeekDay function to fetch the day number of the week. Here is the DAX statement:

```
Day Number of Week =
WEEKDAY(LASTDATE(DimDate[FullDateAlternateKey]))
```

The result of this measure would be the day number of the week starting from Sunday as 1, ending Saturday as 7.

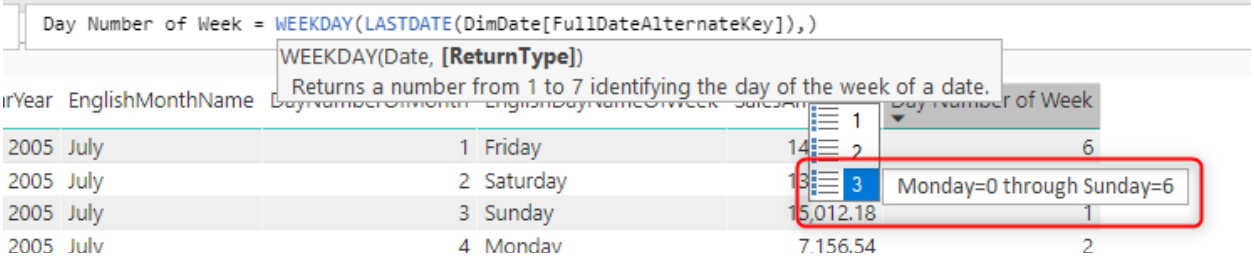


```
Day Number of Week = WEEKDAY(LASTDATE(DimDate[FullDateAlternateKey]))
```

CalendarYear	EnglishMonthName	DayNumberOfMonth	EnglishDayNameOfWeek	SalesAmount	Day Number of Week
2005	July	1	Friday	14,477.34	6
2005	July	2	Saturday	13,931.52	7
2005	July	3	Sunday	15,012.18	1
2005	July	4	Monday	7,156.54	2
2005	July	5	Tuesday	15,012.18	3
2005	July	6	Wednesday	14,313.08	4
2005	July	7	Thursday	7,855.64	5
2005	July	8	Friday	7,855.64	6
2005	July	9	Saturday	20,909.78	7
2005	July	10	Sunday	10,556.53	1
2005	July	11	Monday	14,313.08	2
2005	July	12	Tuesday	14,134.80	3
2005	July	13	Wednesday	7,156.54	4
2005	July	14	Thursday	25,047.89	5
2005	July	15	Friday	11,230.63	6
2005	July	16	Saturday	14,313.08	7
2005	July	17	Sunday	14,134.80	1
2005	July	18	Monday	6,953.26	2
2005	July	19	Tuesday	25,568.71	3
2005	July	20	Wednesday	11,255.63	4
2005	July	21	Thursday	14,313.08	5
2005	July	22	Friday	38,241.29	6
2005	July	23	Saturday	15,012.18	7
2005	July	24	Sunday	10,734.81	1

Starting from Monday

Not always in all businesses, the week starts from Sunday. In fact, in many businesses, the week starts on Monday. WeekDay function has a second parameter which can determine the starting day of the week. Parameter name is Return Type.

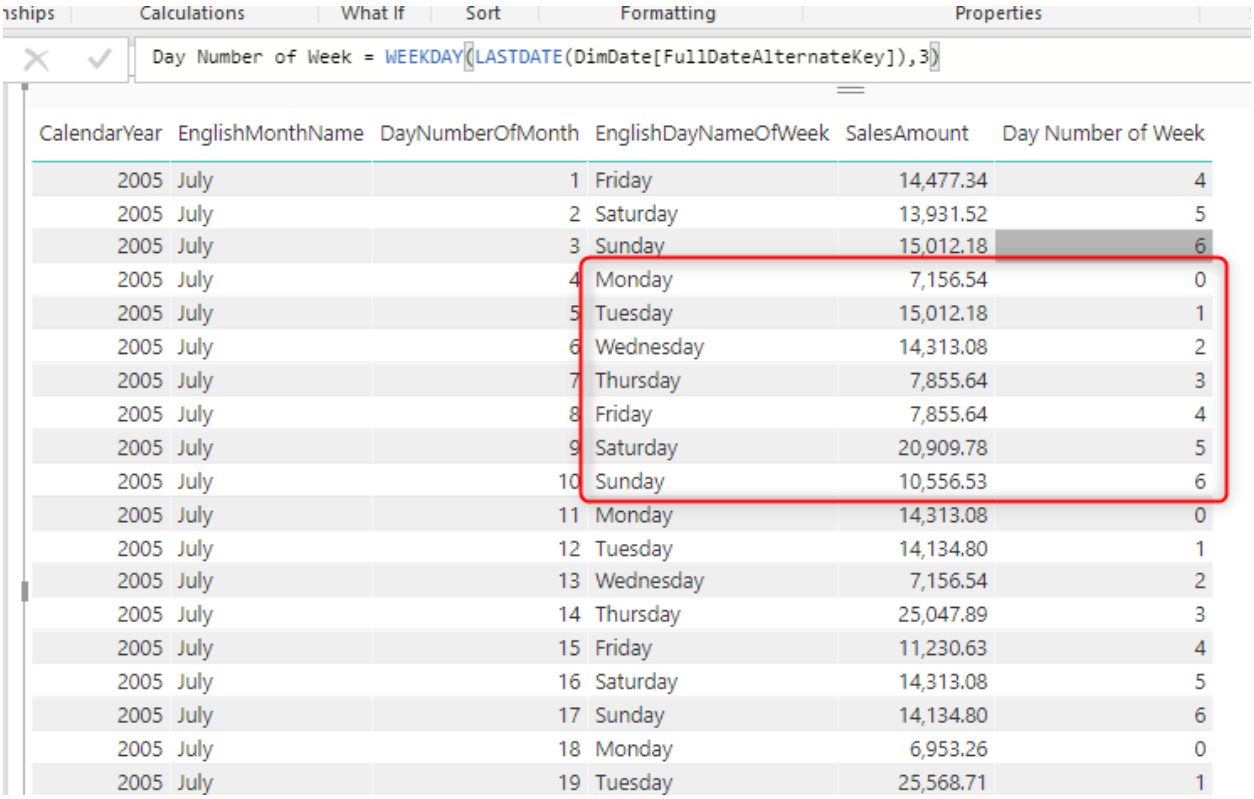


The default value is 1. it means Sunday is 1, and Saturday 7.

If you change it to 2; Monday will be 1, and Sunday 7.

If you change it to 3; Monday will be 0, and Sunday 6. This one will appeal more for the rest of our calculation. Set this parameter to 3.

```
Day Number of Week =
WEEKDAY(LASTDATE(DimDate[FullDateAlternateKey]),3)
```



CalendarYear	EnglishMonthName	DayNumberOfMonth	EnglishDayNameOfWeek	SalesAmount	Day Number of Week
2005	July	1	Friday	14,477.34	4
2005	July	2	Saturday	13,931.52	5
2005	July	3	Sunday	15,012.18	6
2005	July	4	Monday	7,156.54	0
2005	July	5	Tuesday	15,012.18	1
2005	July	6	Wednesday	14,313.08	2
2005	July	7	Thursday	7,855.64	3
2005	July	8	Friday	7,855.64	4
2005	July	9	Saturday	20,909.78	5
2005	July	10	Sunday	10,556.53	6
2005	July	11	Monday	14,313.08	0
2005	July	12	Tuesday	14,134.80	1
2005	July	13	Wednesday	7,156.54	2
2005	July	14	Thursday	25,047.89	3
2005	July	15	Friday	11,230.63	4
2005	July	16	Saturday	14,313.08	5
2005	July	17	Sunday	14,134.80	6
2005	July	18	Monday	6,953.26	0
2005	July	19	Tuesday	25,568.71	1

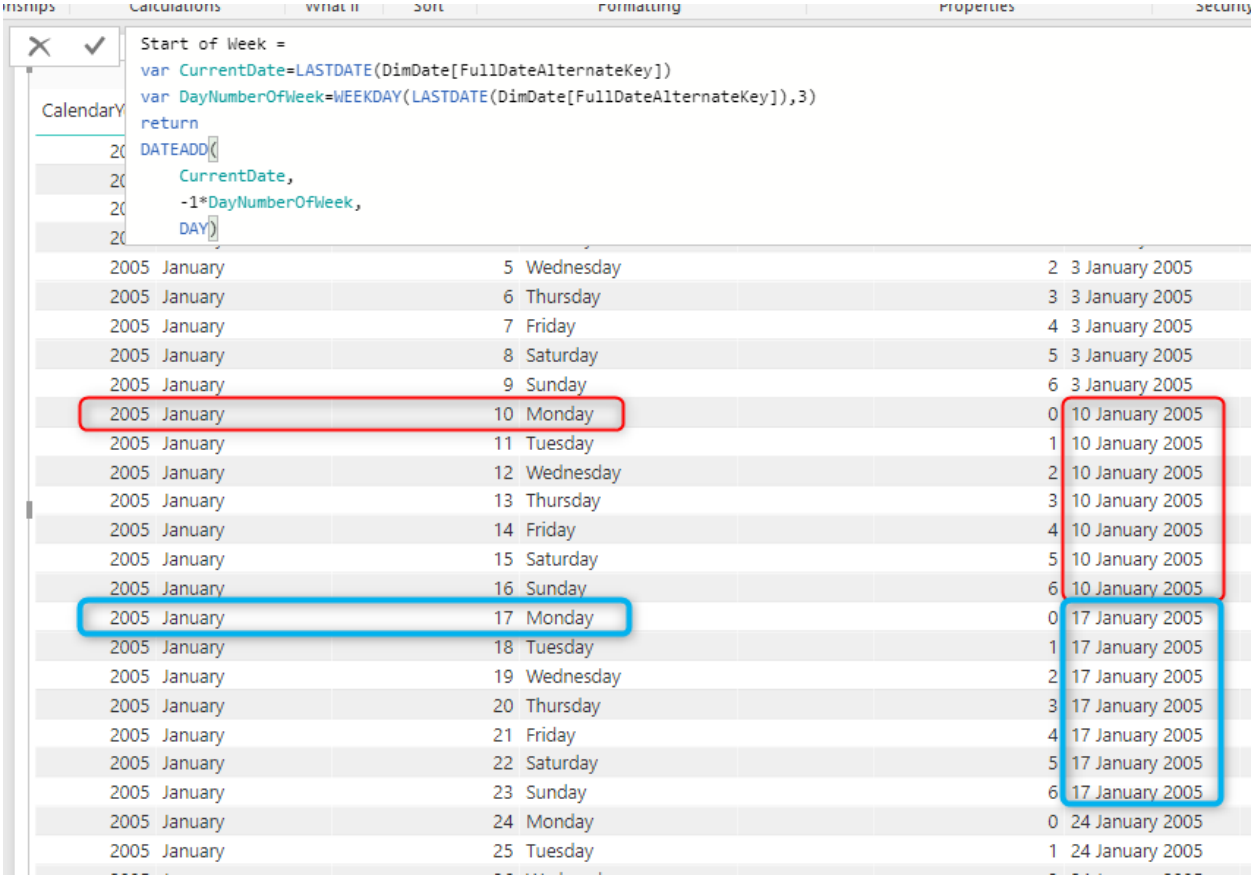
Start of the Week

Now that we know the day of the week, it is easy to calculate the start of the week. You need to go that number back as days interval. For example; Wednesday is day 2 of the week. If you go two days back, you get Monday. Using DateAdd function you can go as many days back you want. Here is the DAX expression:

```
Start of Week =
var CurrentDate=LASTDATE(DimDate[FullDateAlternateKey])
var
DayNumberOfWeek=WEEKDAY(LASTDATE(DimDate[FullDateAlternateKey
]),3)
return
```

```
7 DATEADD(
8     CurrentDate,
      -1*DayNumberOfWeek,
      DAY)
```

The heart of expression above is DateAdd calculation; we go the number of days back (that is the reason for -1*DayNumberOfWeek). Here is the result. For every given date, we get the Monday (which is the start of that week) of that week as the week start date.



2005	January	5	Wednesday	2	3 January 2005
2005	January	6	Thursday	3	3 January 2005
2005	January	7	Friday	4	3 January 2005
2005	January	8	Saturday	5	3 January 2005
2005	January	9	Sunday	6	3 January 2005
2005	January	10	Monday	0	10 January 2005
2005	January	11	Tuesday	1	10 January 2005
2005	January	12	Wednesday	2	10 January 2005
2005	January	13	Thursday	3	10 January 2005
2005	January	14	Friday	4	10 January 2005
2005	January	15	Saturday	5	10 January 2005
2005	January	16	Sunday	6	10 January 2005
2005	January	17	Monday	0	17 January 2005
2005	January	18	Tuesday	1	17 January 2005
2005	January	19	Wednesday	2	17 January 2005
2005	January	20	Thursday	3	17 January 2005
2005	January	21	Friday	4	17 January 2005
2005	January	22	Saturday	5	17 January 2005
2005	January	23	Sunday	6	17 January 2005
2005	January	24	Monday	0	24 January 2005
2005	January	25	Tuesday	1	24 January 2005

Week to Date Calculation

Now that we have the start of the week, we can calculate all dates between that date and the date of the current filter context using DatesBetween, and wrap it inside a Calculate to calculate Week to Date.

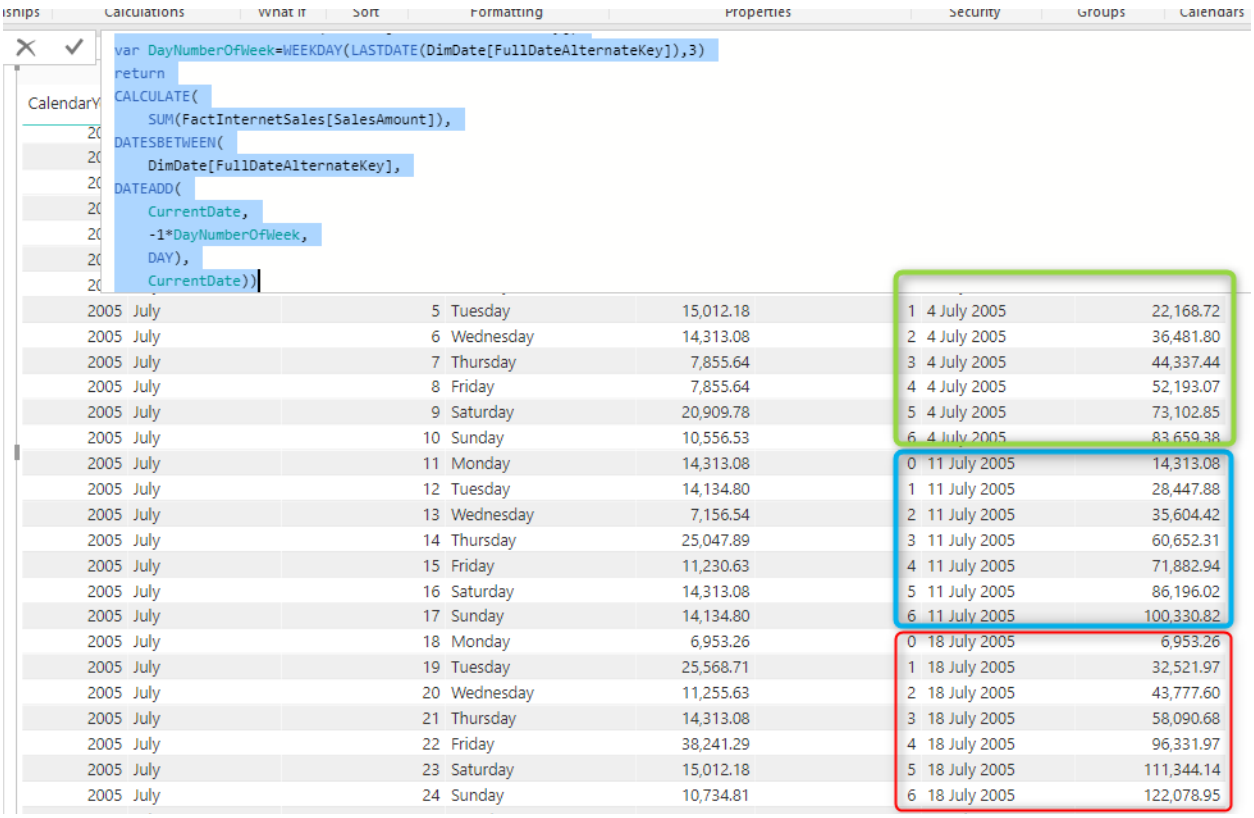
```
1 DATESBETWEEN(
2     DimDate[FullDateAlternateKey],
```

```
3 DATEADD(
4     CurrentDate,
5     -1*DayNumberOfWeek,
6     DAY),
7     CurrentDate)
```

DAX expression above is using DatesBetween to give us all dates between the start of the week (calculated in the previous step), and the date of the current filter context (LastDate(DimDate[FullDateAlternateKey])).
We can then wrap it inside a Calculate function to get the week to date as below;

```
1
2  Week to Date Sales =
3  var CurrentDate=LASTDATE(DimDate[FullDateAlternateKey])
4  var
5  DayNumberOfWeek=WEEKDAY(LASTDATE(DimDate[FullDateAlternateKe
6  y]),3)
7  return
8  CALCULATE(
9      SUM(FactInternetSales[SalesAmount]),
1  DATESBETWEEN(
0      DimDate[FullDateAlternateKey],
1  DATEADD(
1      CurrentDate,
1      -1*DayNumberOfWeek,
2      DAY),
1      CurrentDate))
3
```

Here is the result;



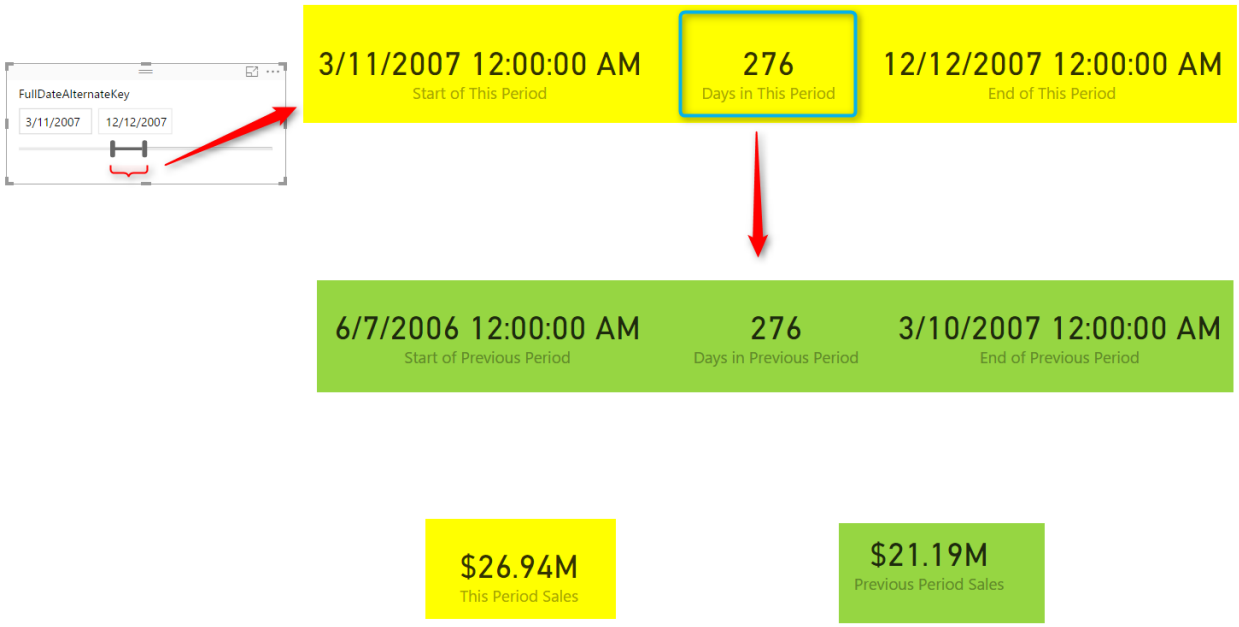
Just for a reference, here is the entire Week to Date calculation with DAX again:

```
Week to Date Sales =
var CurrentDate=LASTDATE(DimDate[FullDateAlternateKey])
var
DayNumberOfWeek=WEEKDAY(LASTDATE(DimDate[FullDateAlternateKey]),3)
return
CALCULATE(
    SUM(FactInternetSales[SalesAmount]),
DATESBETWEEN(
    DimDate[FullDateAlternateKey],
DATEADD(
    CurrentDate,
    -1*DayNumberOfWeek,
    DAY),
    CurrentDate))
```

1
3

Previous Dynamic Period DAX Calculation

Posted by Reza Rad on Jan 12, 2017



Parallel Period is a function that helps you fetching the previous period of a Month, Quarter, or Year. However, if you have a dynamic range of date, and you want to find the previous period of that dynamic selection, then Parallel Period can't give you the answer. As an example; if the user selected a date range from 1st of May 2008 to 25th of November 2008, the previous period should be calculated based on a number of days between these two dates which is 208 days, and based on that previous period will be from 5th of October 2007 to 30th of April 2008. The ability to do such calculation is useful for reports that user wants to compare the value of the current period with whatever period it was before this. In this post I'll show you an easy method for doing this calculation, I will be using one measure for each step to help you understand the process easier. If you like to learn more about DAX and Power BI, read Power BI online book from Rookie to Rock Star.

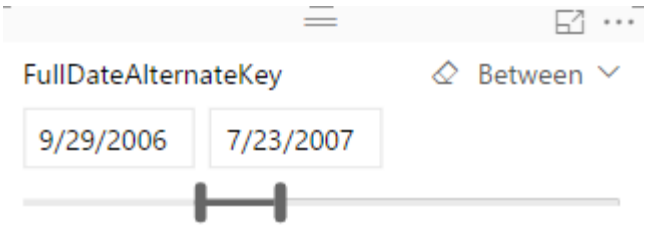
Prerequisite

For running example of this post, you will need the AdventureWorksDW sample database, or you can download an **Excel version** of it from here:

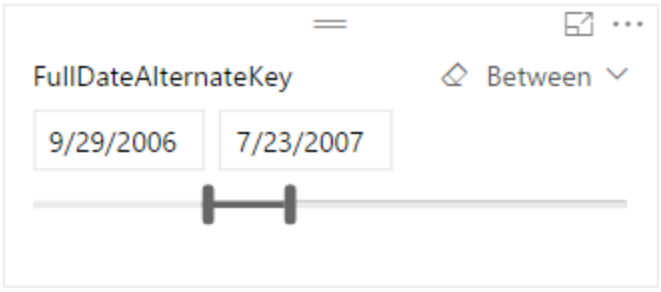
Download

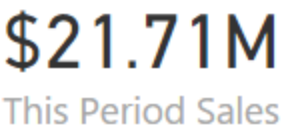
Current Period

I will go through this with an example; Create a new Power BI Desktop file and choose DimDate, and FactInternetSales from AdventureWorksDW. Make sure that there is only one **Active** relationship between these two tables based on OrderDateKey in the FactInternetSales table and DateKey in the DimDate table. Now add a slicer for FullDateAlternateKey in the page



Also, add a Card visual which shows SalesAmount from the FactInternetSales table.





I normally prefer to create an explicit measure for this type of calculations, that's why I have created a measure named "This Period Sales" with DAX code below; (the measure for This Period Sales is not necessary, because Power BI does the same calculation automatically for you)

```
1 This Period Sales = SUM(FactResellerSales[SalesAmount])
```

Start of Current Period

To understand the current period, an easy way can be the calculation of the start, the end of period and the number of days between these two. Start of Period is simple. I create a measure under DimDate, as below:

```
1 Start of This Period = FIRSTDATE(DimDate[FullDateAlternateKey])
```

FirstDate() DAX function returns the first available date in the current evaluation context, which will be whatever filtered in the date range.

End of Current Period

Same as the start of the period, for the end of the period I will use a simple calculation, but this time with *LastDate()* to find the latest date in the current selection.

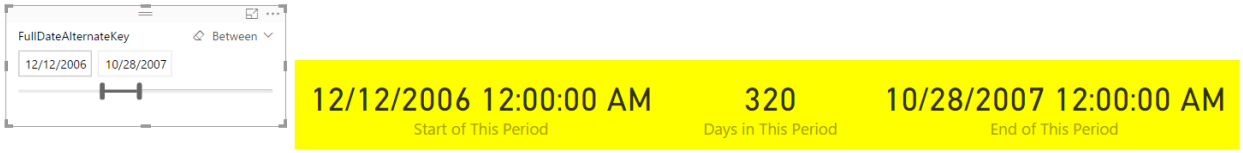
```
1 End of This Period = LASTDATE(DimDate[FullDateAlternateKey])
```

Days in This Period

Next easy step is understanding the number of days between start and end of the period, which is simply by using *DateDiff()* DAX function as below;

```
1 Days in This Period = DATEDIFF([Start of This Period],[End of This Period],DAY)
```

I add them all in the report as Card Visuals (one for each measure), and here is the result so far;



Previous Period

After finding a number of days in this period, start, and end of the current period, it is a simple calculation to find the previous period. The previous period calculation should be number of days in this period minus start of the current period. To exclude the start of the period to calculate twice, I'll move one more day back. Here is the calculation step by step, I'll start with the Start of Previous Period;

Start of Previous Period

Using DateAdd to reduce the number of days from DimDate

```
1 DATEADD(DimDate[FullDateAlternateKey],-1*[Days in This Period],DAY)
```

DateAdd() DAX function adds a number of intervals to a **date set**. In this example interval is DAY, and date set is all dates in DimDate[FullDateAlternateKey] field (because DateAdd doesn't work with single date), and the number of intervals is Days in This Period multiplied by -1 (to move dates backward rather than forward).

Fetch the First Date of the result

```
1 FIRSTDATE(DATEADD(DimDate[FullDateAlternateKey],-1*[Days in This Period],DAY))
```

FirstDate() used here to fetch first value only.
Move one day back

```
1 PREVIOUSDAY(FIRSTDATE(DATEADD(DimDate[FullDateAlternateKey],-1*[Days in This Period],DAY)))
```

To exclude current date from the selection we always move one day back, that's what *PreviousDay()* DAX function does. It always returns a day before the input date.
These are not three separate DAX expressions or measure; this is only one measure which I explained step by step. Here is the full expression:

```
  Start of Previous Period =
1 PREVIOUSDAY(FIRSTDATE(DATEADD(DimDate[FullDateAlternateKey],-1*[Days in This Period],DAY)))
```

End of Previous Period

Similar to the Start of Previous Period calculation, this calculation is the same the only difference is using LastDate();

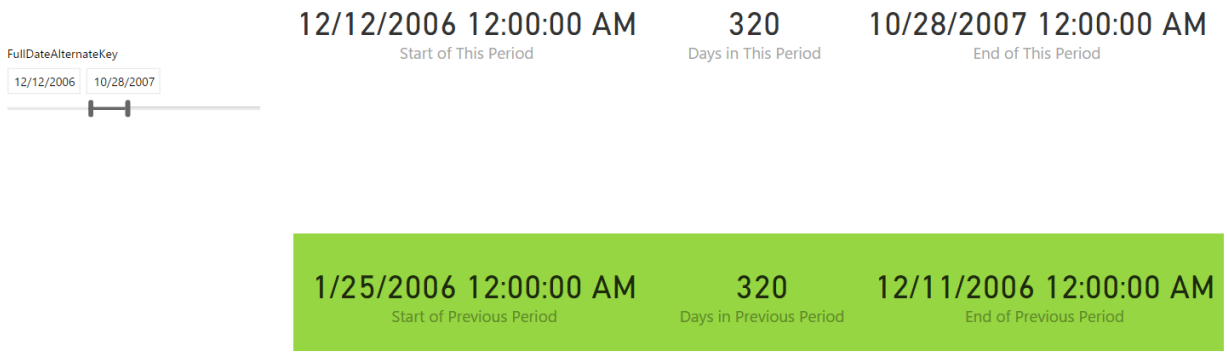
```
  End of Previous Period =
1 PREVIOUSDAY(LASTDATE(DATEADD(DimDate[FullDateAlternateKey],-1*[Days in This Period],DAY)))
```

Days in Previous Period

You don't need to create this measure, I have only created this to do a sanity check, to see; do I have the same number of days in this period compared with a previous period or not;

```
1 Days in Previous Period = DATEDIFF([Start of Previous Period],[End of Previous Period],DAY)
```

Now if I add all of these measures to the report with card visuals again I can see previous period calculation works correctly;



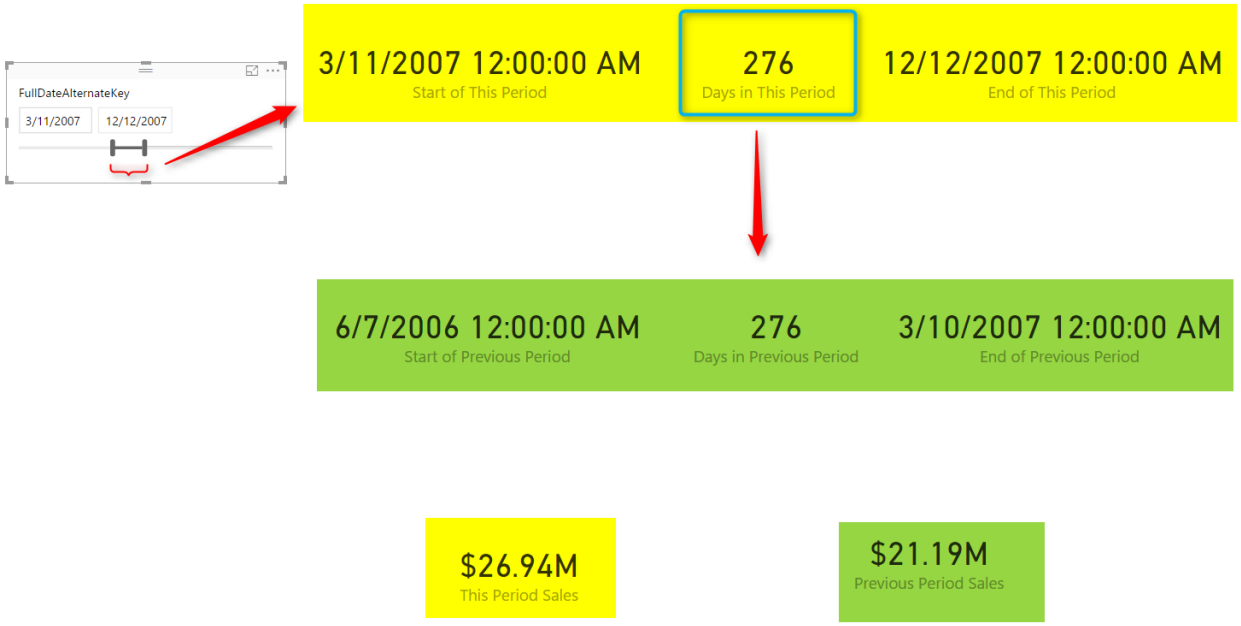
With every change you apply in the date range slicer, you can see the previous period calculates the range again, it will always be the same number of days as the current period, but the same number of days BEFORE. in the screenshot above you can see that start of the previous period is 321 days before start of this period (1 more days because the end of the previous period does not exactly start of this period, it is one day before. we don't want to duplicate values of date in current and previous calculations).

Previous Period Sales

Now as an example I have created another measure to show you the sum of SalesAmount for the previous period. The calculation here uses DatesBetween() DAX function to fetch all the dates between the start of the previous period and end of the previous period;

```
Previous Period Sales = CALCULATE(SUM(FactResellerSales[SalesAmount])
                        ,DATESBETWEEN(
DimDate[FullDateAlternateKey],
                                      [Start of Previous
Period],
                                      [End of Previous
Period]),
                                      ALL(DimDate) )
```

Showing all of these on a page now;

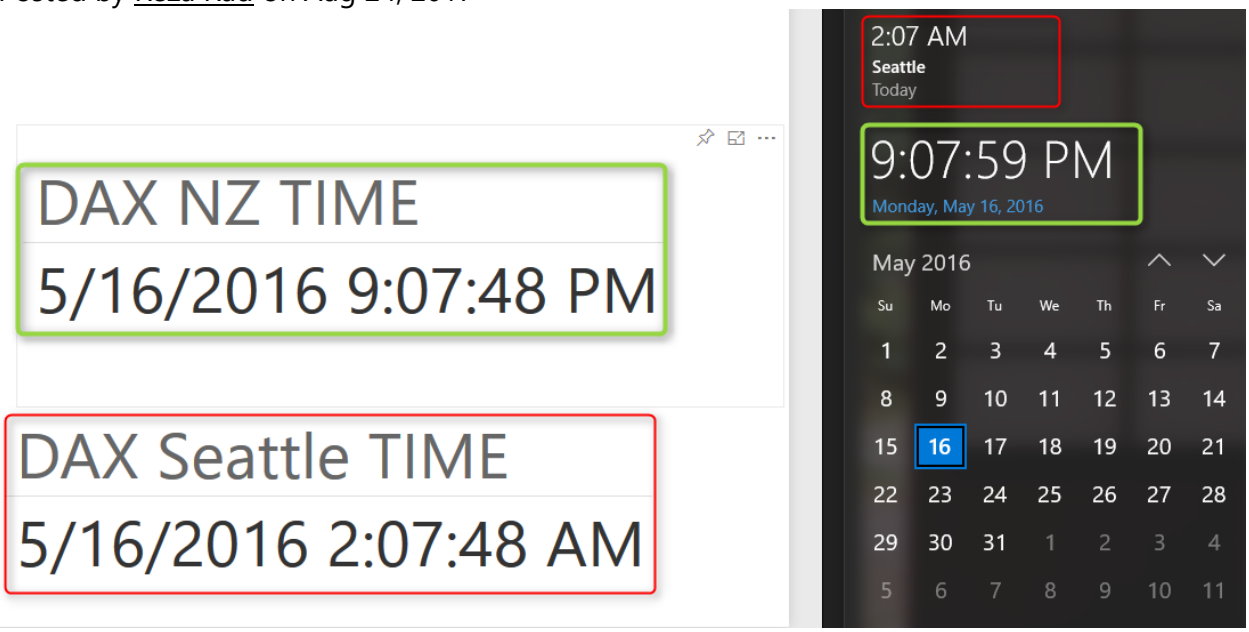


Summary

This was a very quick and simple post to show you a useful DAX calculation to find Dynamic Previous Period based on the selection of date range in Power BI report page. I have used number of DAX functions such as FirstDate(), LastDate(), DateAdd(), DateDiff(), and PreviousDay() to do calculations. Calculation logic is just counting a number of days in the current period and reducing it from the start and end of the current period to find previous period.

Solving DAX Time Zone Issue in Power BI

Posted by Reza Rad on Aug 24, 2017



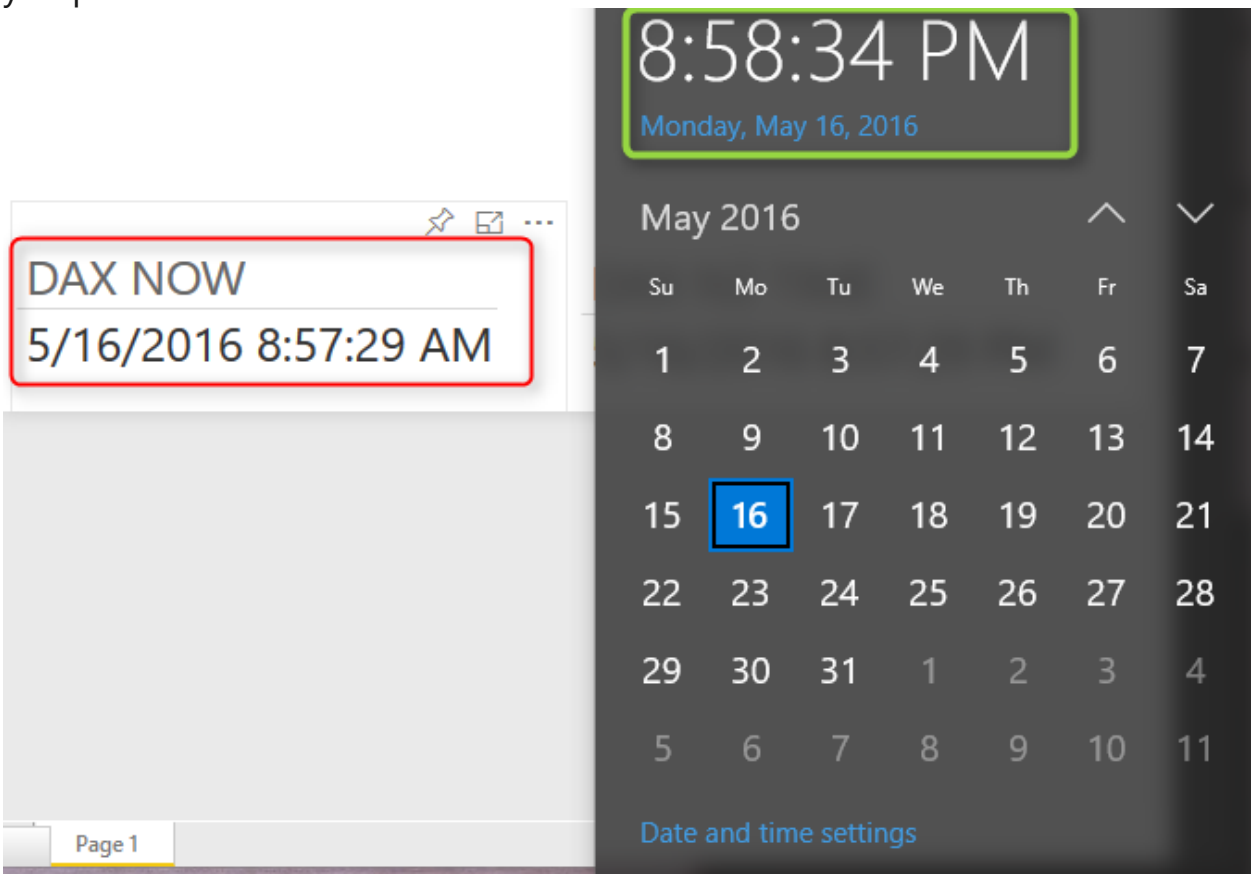
Power BI is a cloud service, and that means Power BI files are hosted somewhere. Some DAX functions such as Date/Time functions work on system date/time on the server their file is hosted on. So If you use DAX functions such as TODAY() or NOW() you will not get your local date/time, You will fetch server's date/time. In this blog post I'll explain methods of solving this issue, so you could use Power BI to resolve your specific time zone's date and time. If you want to learn more about Power BI, read Power BI online book; Power BI from Rookie to Rock Star.

Defining the Problem

Using DAX functions on your local Power BI file is different from what you will see in Power BI website especially when date and time functions have been used. The reason is that DAX works with the date and time of the system that hosted the Power BI file. Power BI is a could-based service, and that means Power BI files will be hosted on a server somewhere in the world, that might not be on the same time zone as your city is. So as a result when you used

functions that work with the current date and time; such as TODAY() or NOW() in DAX you will not get your local current date and time. At this stage there is time zone feature in DAX functions to help to resolve this, so I suggest few options resolve it as below.

The screenshot below shows a Power BI report published on Power BI website, and the result of DAX NOW() function there compared with the local date/time on the client system. Please note that you won't see this anomaly in Power BI Desktop, because in that case file is running on your local system, and the result would be your local date/time, you will only face this issue when you publish the solution to Power BI website.



Method 1 – DAX Formula Manipulation

One easy way of solving this is to add a time offset to the date/time function in DAX. Power BI date/time seems to be GMT. So if I want to show my local

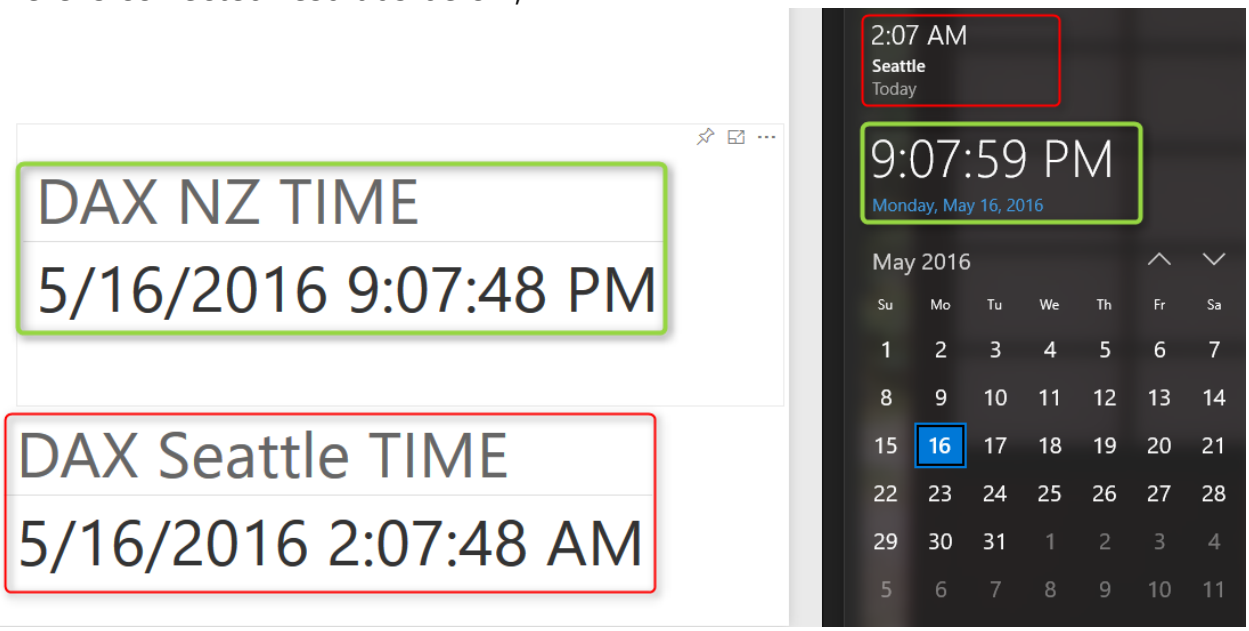
time in Auckland, I have to add 12 hours to it. Or for Seattle, I have to reduce 7 hours from it.
So I create a new calculation as DAX NZ TIME with this code:

```
1 DAX NZ TIME = NOW()+(12/24)
```

and another for DAX Seattle Time with this code:

```
1 DAX Seattle TIME = NOW()-(7/24)
```

Here is corrected result as below;



This method works but has an issue which I deal with it later on.

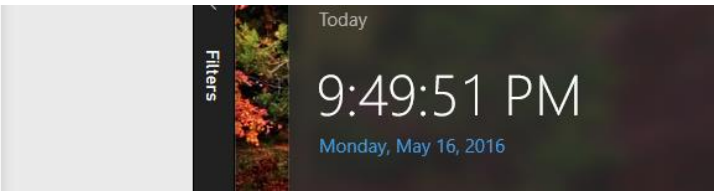
Method 2 – Power Query DateTimeZone Functions

Thanks to my friend Ken Puls who mentioned this method to me in PASS BA conference, I come with this second option. Fortunately, in Power Query, there is set of functions for DateTimeZone. Ken already has a blog post about time zones with Power Query which is a good read and recommended.
DateTimeZone functions have options such as fetching local time or switching time zones. For this purpose, I can use DateTimeZone.SwitchZone function to switch the server's local time to my time zone's date and time.

```
1 = DateTimeZone.SwitchZone(DateTimeZone.LocalNow(),12,0)
```

12 is hours, and 0 is minutes for the new time zone. Script above will turn the local time zone to NZ time. For turning that into Seattle time, I have to set parameters to -7, and 0.
And here is the result set:

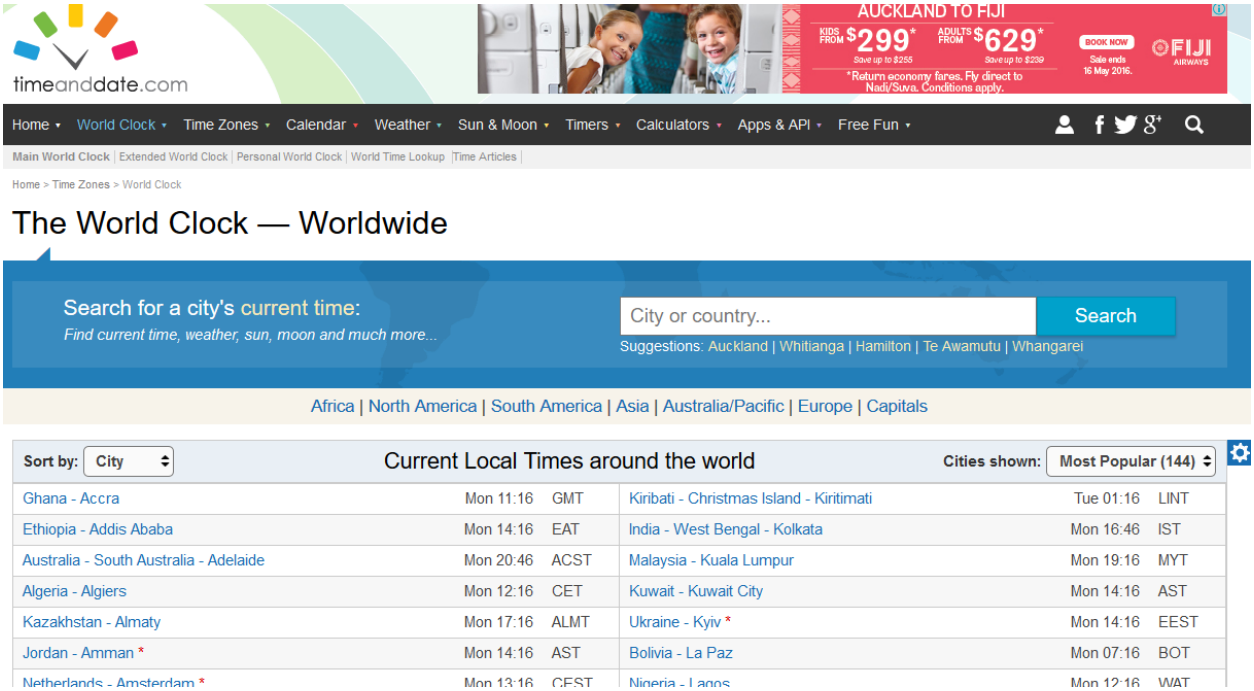
NZ TIME
5/16/2016 9:49:30 PM



You can also use other functions such as DateTime.AddZone in Power Query to turn the local date-time to specific time zone.
Well the above solution works like DAX method, but both suffer from similar issue; Day Light Saving Time. This is the reason that if you try the code above in summer or winter, you might get a different result!

Method 3 – Web Query with Power Query

Day Light Saving is a big challenge, because each time zone, city, or country might have different daylight saving time. Even same city might have different dates for DST (Daylight Saving Time) for different years! Power Query is intelligent enough to help with Time Zone issue but doesn't have a directory of all DST times for all time zones. Fortunately, Power Query can query web URL. And some websites give you the current date and time for your specific city, country, or time zone. And those websites usually consider DST correctly. One of these websites is TimeAndDate.com. As you see in the screenshot below this website gives me the current date and time for most of the cities around the world;



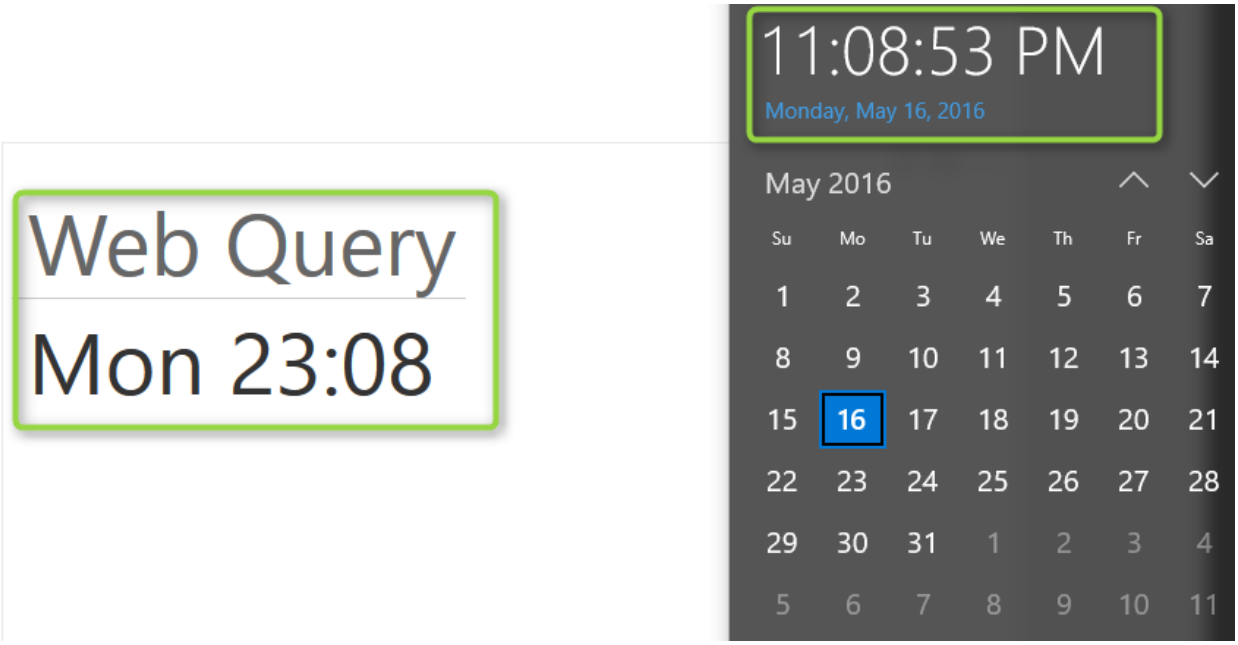
In Power Query, we can use functions such as Web.Page() and Web.Contents() to read tables in a web page, and then fetch part of it that we want with some other transformations. I won't be explaining details of using timeanddate.com URL to fetch the local city here because it would make this post very long. I refer you to my other post about reading some date/time information for different time zones which is similar to the method I've used here. If you want to understand how the code below works read the post here. For this part I will be using another website which gives me current date and time in Auckland, and here is Power Query code:

```
let
    Source = Web.Page(Web.Contents("http://localtimes.info/Oceania/New_Zealand/Auckland/")),
    Data1 = Source{1}[Data],
    #"Changed Type" = Table.TransformColumnTypes(Data1,{{"Column1", type text}, {"Column2", type text}}),
    date = #"Changed Type"{1}[Column2],
    time=#"Changed Type"{0}[Column2],
    datetime=DateTime.FromText(date&" "&time)
```

in

datetime

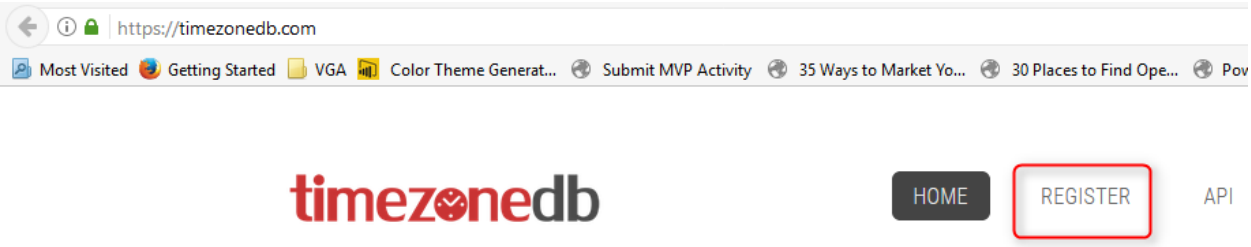
And here is the result with the correct DST and time zone;



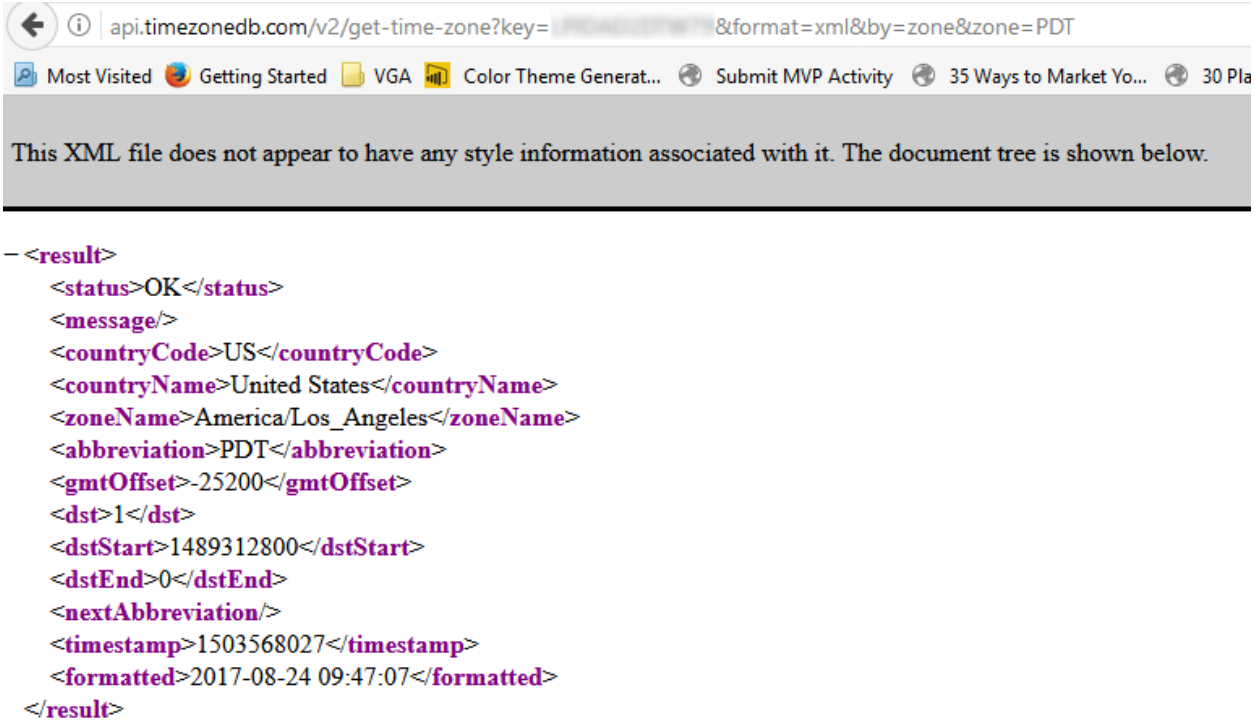
Method 3 Revisited – with Xml.Document

The method mentioned with web query uses Web.Page Power Query function and hence it requires gateway setup to work. Thanks to Yingwei Yang from Microsoft team who suggested this approach; there is another way which doesn't require gateway setup: using Xml.Document function. Let's go through that solution.

Timezonedb.com is the website that has an API to return timezone information. Fortunately API is free to use. You need to register for the API;



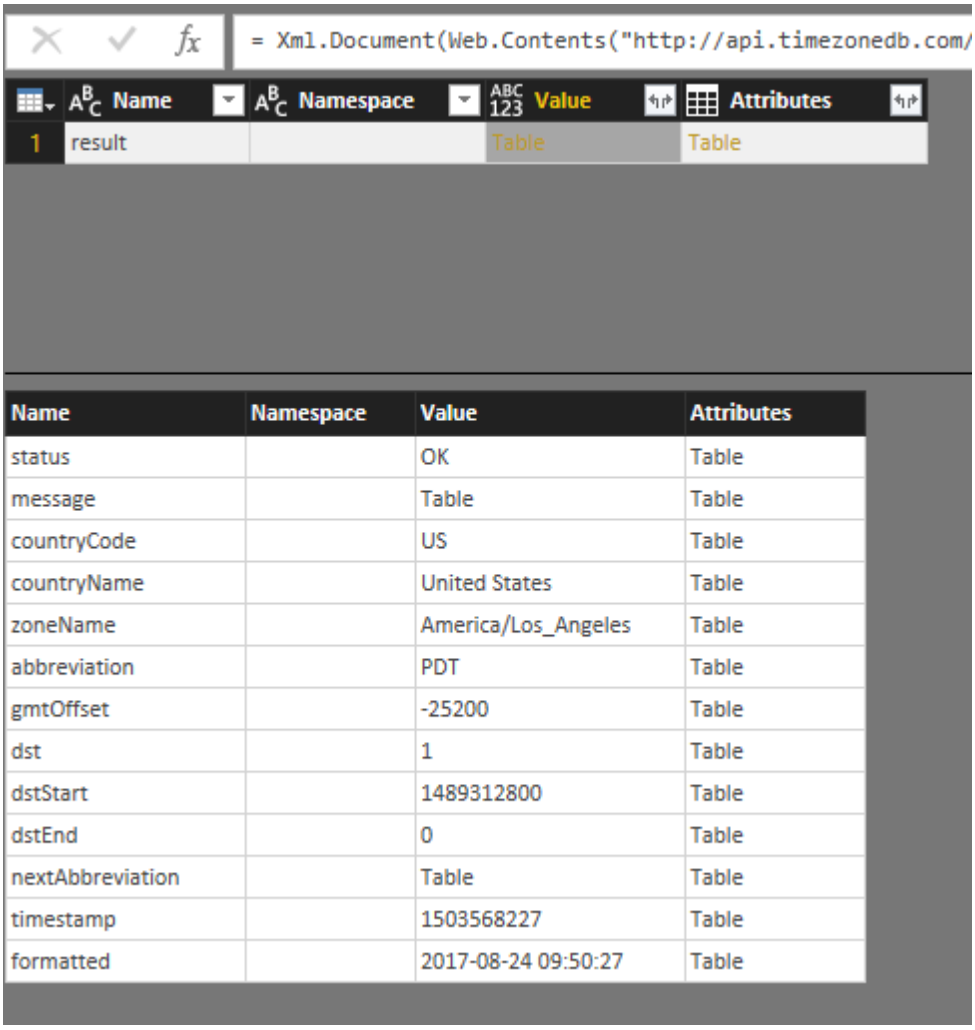
after registering you will receive an API Key which you can use in an API URL as below:



To learn more about API read this link.
Now that we have an API to work with, we can use Xml.Document function to read data from it. Here is how to do it. Start with a Blank Query in Power Query, then go to View -> Advanced Editor and replace the whole query with below script:

```
let
    Source =
Xml.Document(Web.Contents("http://api.timezonedb.com/v2/get-time-
zone?key=XYZ&format=xml&by=zone&zone=PDT")),
    Value = Source{0}[Value],
    Value1 = Value{12}[Value]
in
    Value1
```

This method is the recommended method from all the above options.



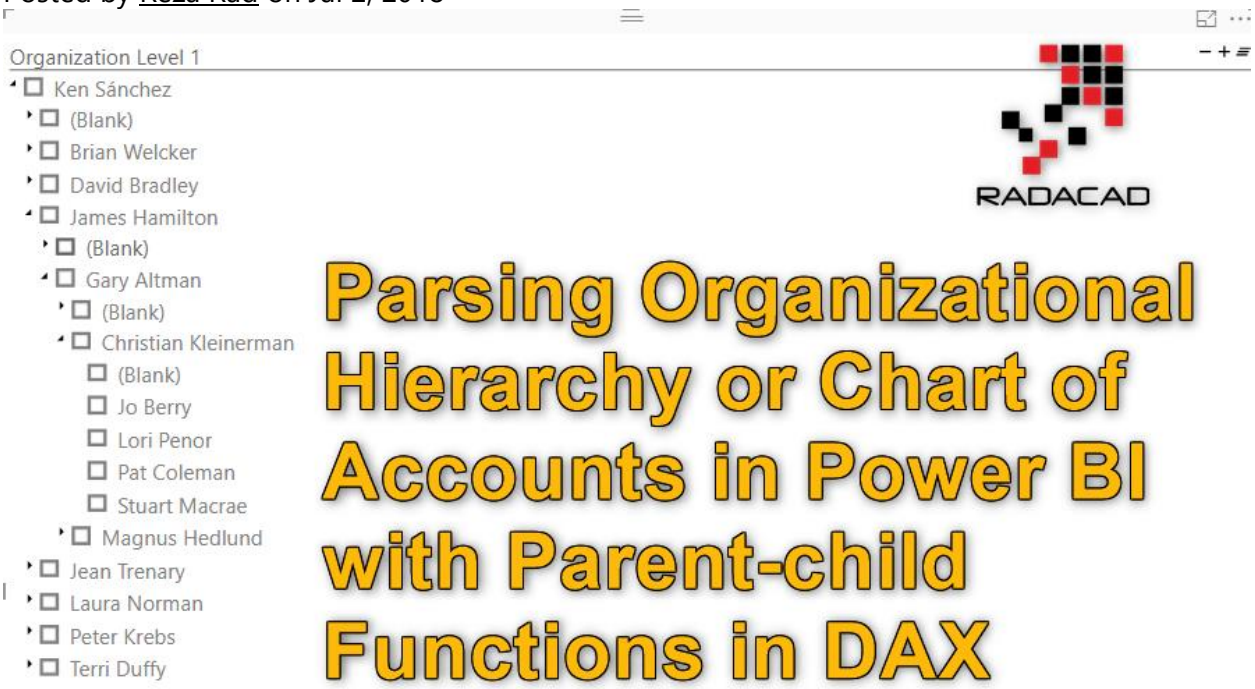
Name	Namespace	Value	Attributes
status		OK	Table
message		Table	Table
countryCode		US	Table
countryName		United States	Table
zoneName		America/Los_Angeles	Table
abbreviation		PDT	Table
gmtOffset		-25200	Table
dst		1	Table
dstStart		1489312800	Table
dstEnd		0	Table
nextAbbreviation		Table	Table
timestamp		1503568227	Table
formatted		2017-08-24 09:50:27	Table

Other Methods

At the time of writing this post, I've only thought about these three methods. You might have an idea about another method. In that case, don't hesitate to share it here.

Parsing Organizational Hierarchy or Chart of Accounts in Power BI with Parent-child Functions in DAX

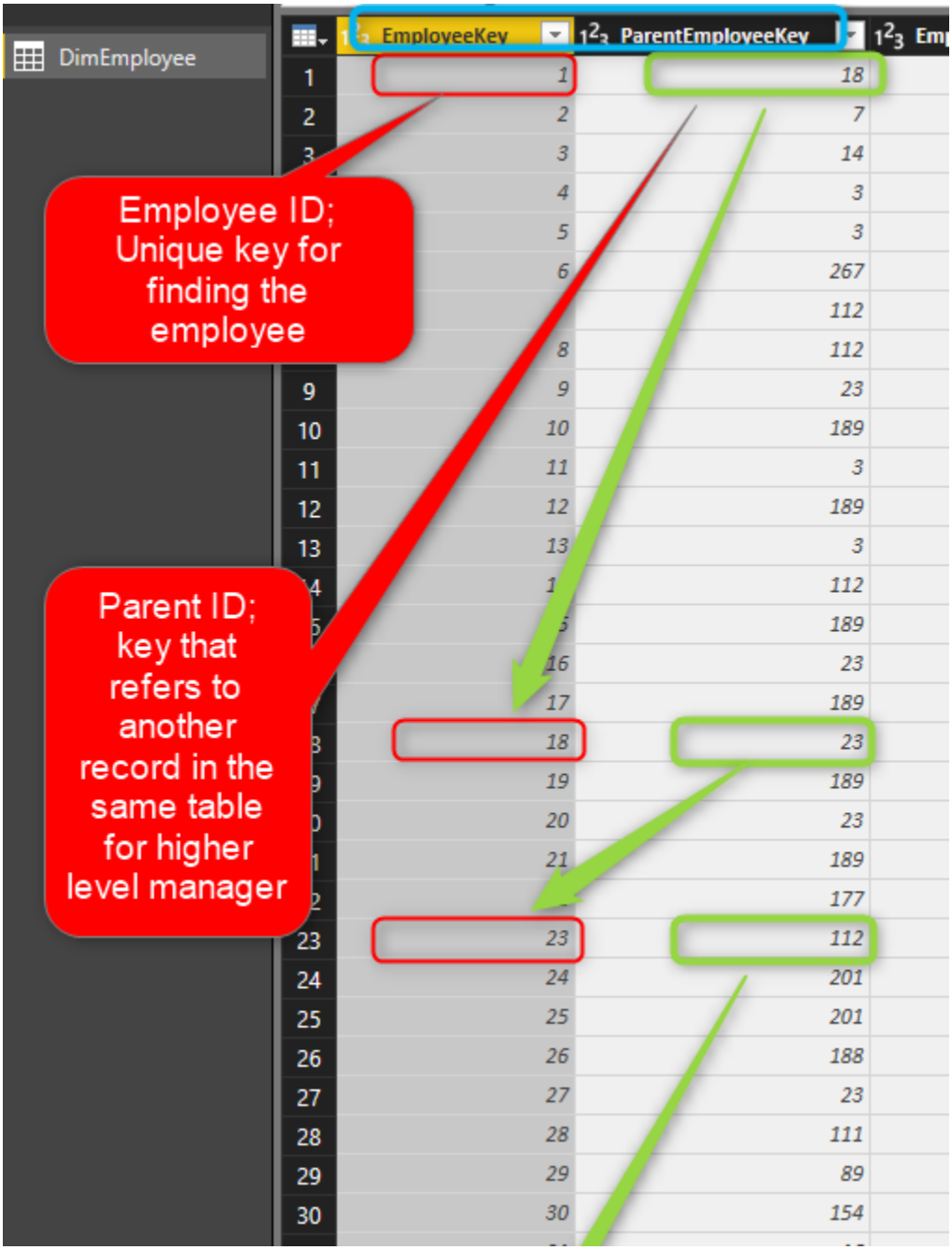
Posted by Reza Rad on Jul 2, 2018



Parent-child functions in DAX are very useful for parsing an organizational hierarchy or something like a chart of accounts. Usually, for hierarchies that the number of levels is not determined, you need to use a different method, and parent-child functions in DAX are a big help for that type of hierarchies. In this post, you will learn what functions are involved in this class of functions, and how to use it to parse an organizational chart. The same method can be used for a chart of accounts. If you want to learn more about Power BI, read Power BI book, from Rookie to Rock Star.

Introduction

Organizational chart or chart of accounts are specific types of hierarchy. Because it is not usually clear that how many levels of hierarchy you get, the hierarchy structure is stored in two columns across the table; ID, and Parent ID. ID usually points to the existing row as the unique key, and Parent ID usually points to another row in the same table as the ID of manager, or parent, or higher level's member. Only these two columns together build the hierarchy. Here is an example;



DAX has a set of functions named as Parent-child functions very useful for parsing this type of hierarchy. Let's see how these functions work.

Sample Dataset

If you want to walk through the example of this post, create a new Power BI Desktop file, and get data from AdventureWorksDW and select DimEmployee

as the only table to get data from. Here is how to access the AdventureWorksDW dataset;

Enter Your Email to download the file (required)

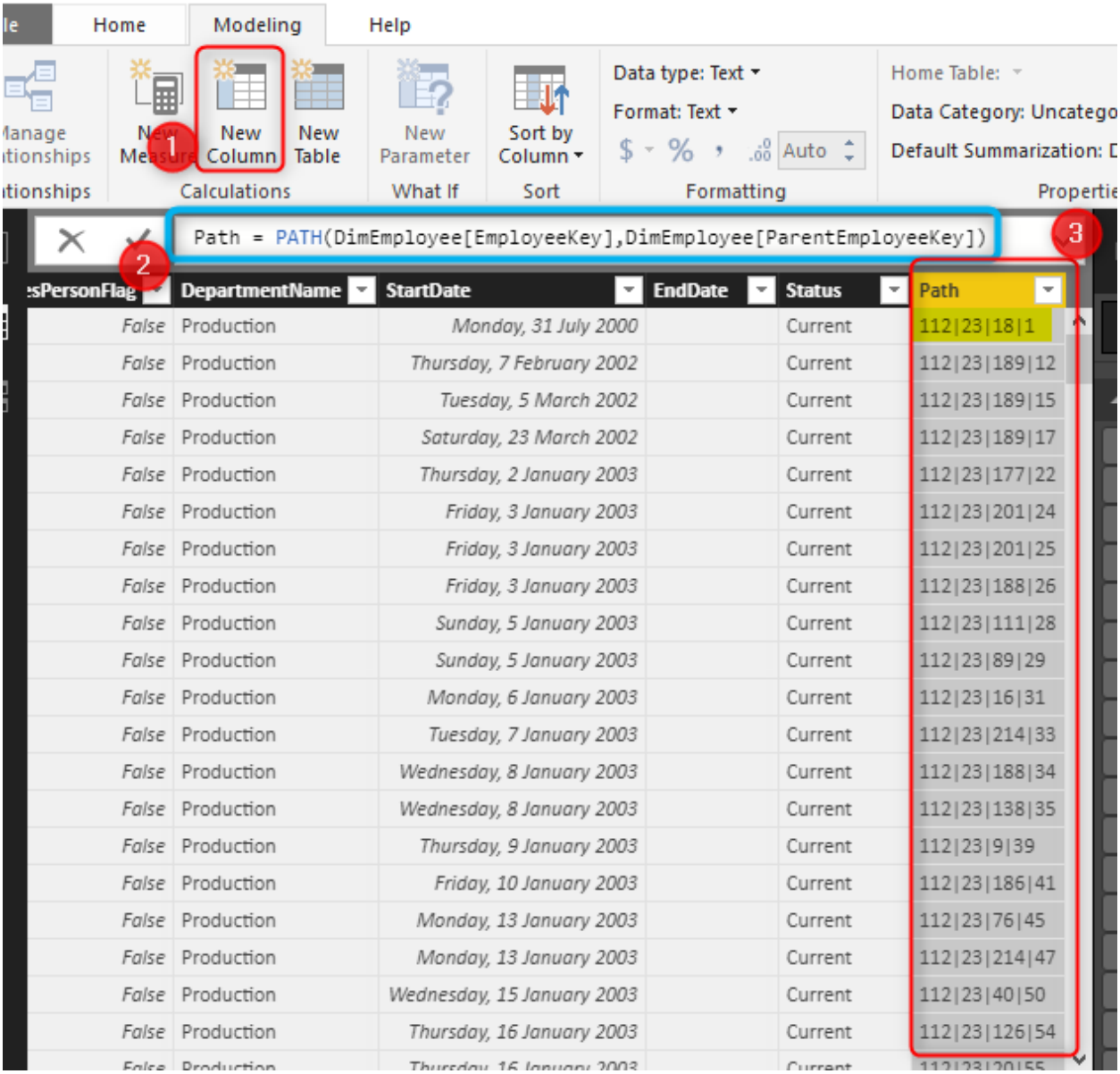
Download

Path Function: Finding the entire path from one member

The first function is named as Path. This function gets two parameters; ID and Parent ID. The Path is a very simple function to use. You need to create a calculated column with this function with the code below;

```
Path =
PATH(DimEmployee[EmployeeKey],DimEmployee[ParentEmployeeKey])
```

Here is the result;



DepartmentName	StartDate	EndDate	Status	Path
Production	Monday, 31 July 2000		Current	112|23|18|1
Production	Thursday, 7 February 2002		Current	112|23|189|12
Production	Tuesday, 5 March 2002		Current	112|23|189|15
Production	Saturday, 23 March 2002		Current	112|23|189|17
Production	Thursday, 2 January 2003		Current	112|23|177|22
Production	Friday, 3 January 2003		Current	112|23|201|24
Production	Friday, 3 January 2003		Current	112|23|201|25
Production	Friday, 3 January 2003		Current	112|23|188|26
Production	Sunday, 5 January 2003		Current	112|23|111|28
Production	Sunday, 5 January 2003		Current	112|23|89|29
Production	Monday, 6 January 2003		Current	112|23|16|31
Production	Tuesday, 7 January 2003		Current	112|23|214|33
Production	Wednesday, 8 January 2003		Current	112|23|188|34
Production	Wednesday, 8 January 2003		Current	112|23|138|35
Production	Thursday, 9 January 2003		Current	112|23|9|39
Production	Friday, 10 January 2003		Current	112|23|186|41
Production	Monday, 13 January 2003		Current	112|23|76|45
Production	Monday, 13 January 2003		Current	112|23|214|47
Production	Wednesday, 15 January 2003		Current	112|23|40|50
Production	Thursday, 16 January 2003		Current	112|23|126|54

As you can see the output column has all the hierarchy from the current member. The first record's data means: the current record's ID is 1, the manager of that is the record with ID of 18, and manager of that is the record with ID of 23, and then the top-level manager is 112. You can see 112 is the top level manager in all records as well.

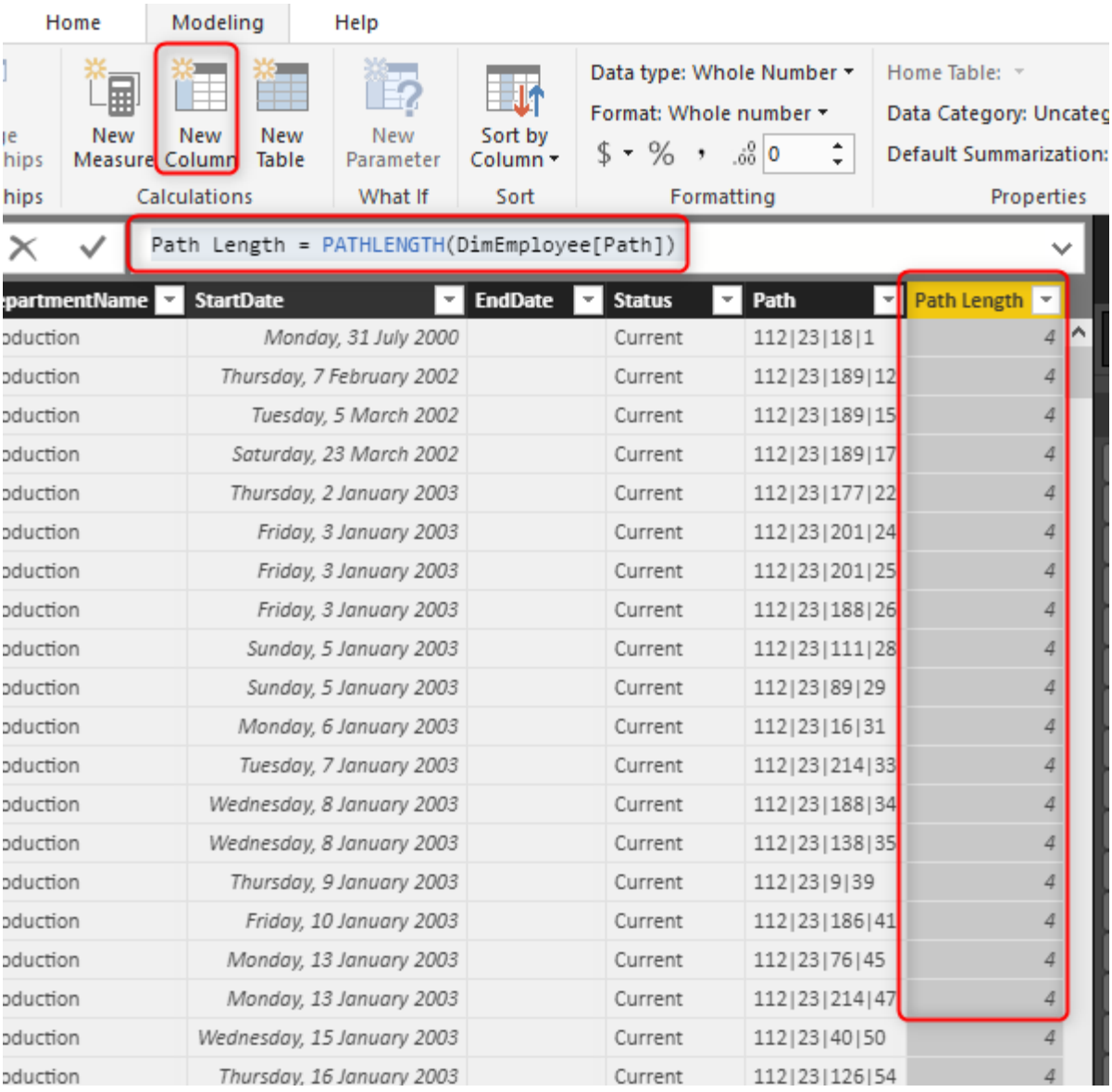
Finding the Length of Path; PathLength Function

Your next step is to find out how many levels of management you have in the hierarchy. You can use PathLength Function to find out the count of levels for

each row. Create a calculated column with PathLength function. This function gets the result of Path function as the input so that you can use the column created in the previous step as the input of this function.

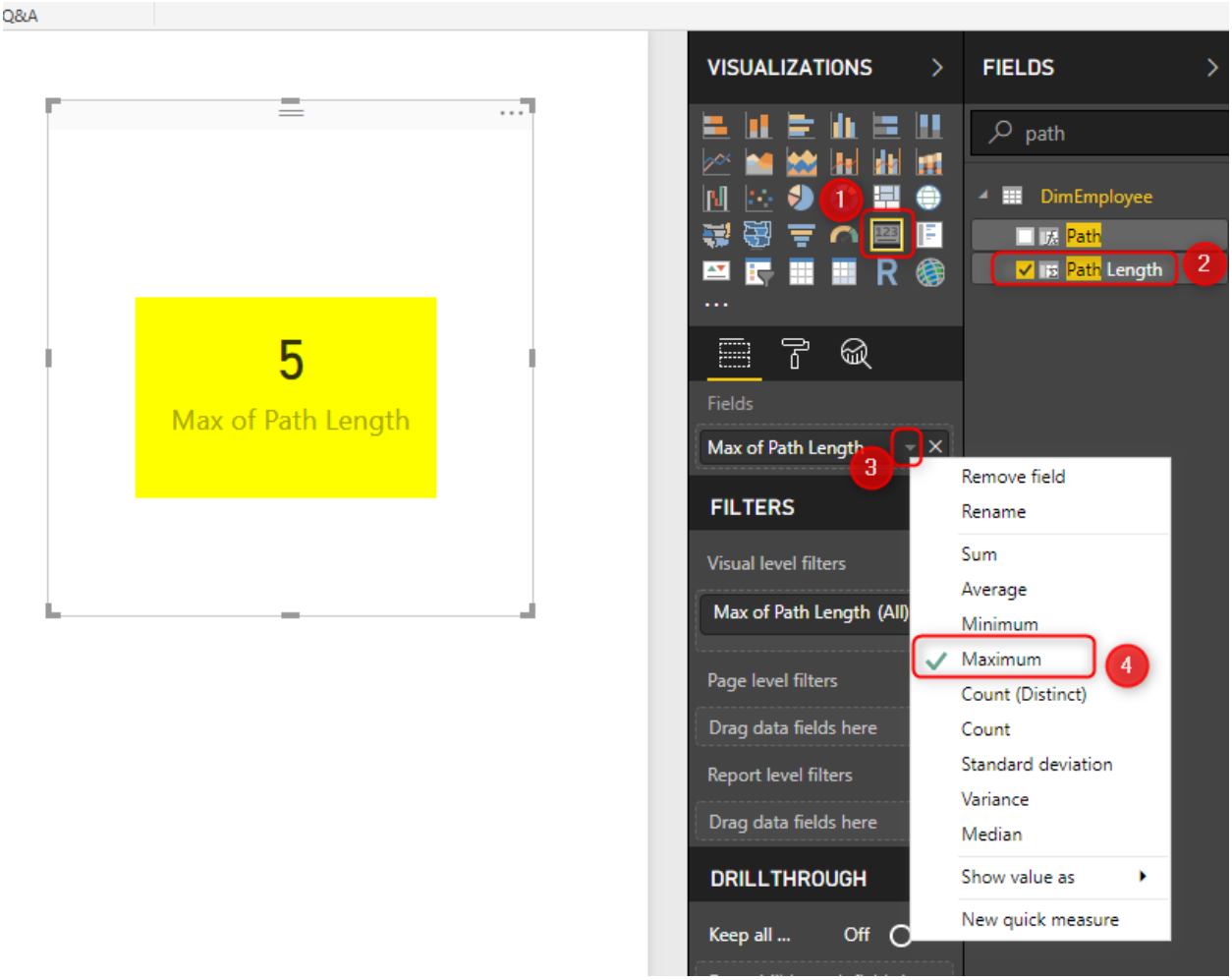
1 Path Length = PATHLENGTH(DimEmployee[Path])

Here is the result;



partmentName	StartDate	EndDate	Status	Path	Path Length
oduction	Monday, 31 July 2000		Current	112|23|18|1	4
oduction	Thursday, 7 February 2002		Current	112|23|189|12	4
oduction	Tuesday, 5 March 2002		Current	112|23|189|15	4
oduction	Saturday, 23 March 2002		Current	112|23|189|17	4
oduction	Thursday, 2 January 2003		Current	112|23|177|22	4
oduction	Friday, 3 January 2003		Current	112|23|201|24	4
oduction	Friday, 3 January 2003		Current	112|23|201|25	4
oduction	Friday, 3 January 2003		Current	112|23|188|26	4
oduction	Sunday, 5 January 2003		Current	112|23|111|28	4
oduction	Sunday, 5 January 2003		Current	112|23|89|29	4
oduction	Monday, 6 January 2003		Current	112|23|16|31	4
oduction	Tuesday, 7 January 2003		Current	112|23|214|33	4
oduction	Wednesday, 8 January 2003		Current	112|23|188|34	4
oduction	Wednesday, 8 January 2003		Current	112|23|138|35	4
oduction	Thursday, 9 January 2003		Current	112|23|9|39	4
oduction	Friday, 10 January 2003		Current	112|23|186|41	4
oduction	Monday, 13 January 2003		Current	112|23|76|45	4
oduction	Monday, 13 January 2003		Current	112|23|214|47	4
oduction	Wednesday, 15 January 2003		Current	112|23|40|50	4
oduction	Thursday, 16 January 2003		Current	112|23|126|54	4

To find out the size of hierarchy, you need to find out the maximum PathLength value. You can create a report visual, and show the Maximum of Path Length field there to see what is the maximum number of levels in your dataset.



As you can see, the maximum number of levels in the dataset of the example in this blog post is 5.

PathItem; Finding specific levels of the hierarchy

The next step is to create a column for each level of the hierarchy. Using PathItem, you can find out the item for each level of the hierarchy. PathItem gets three parameters;

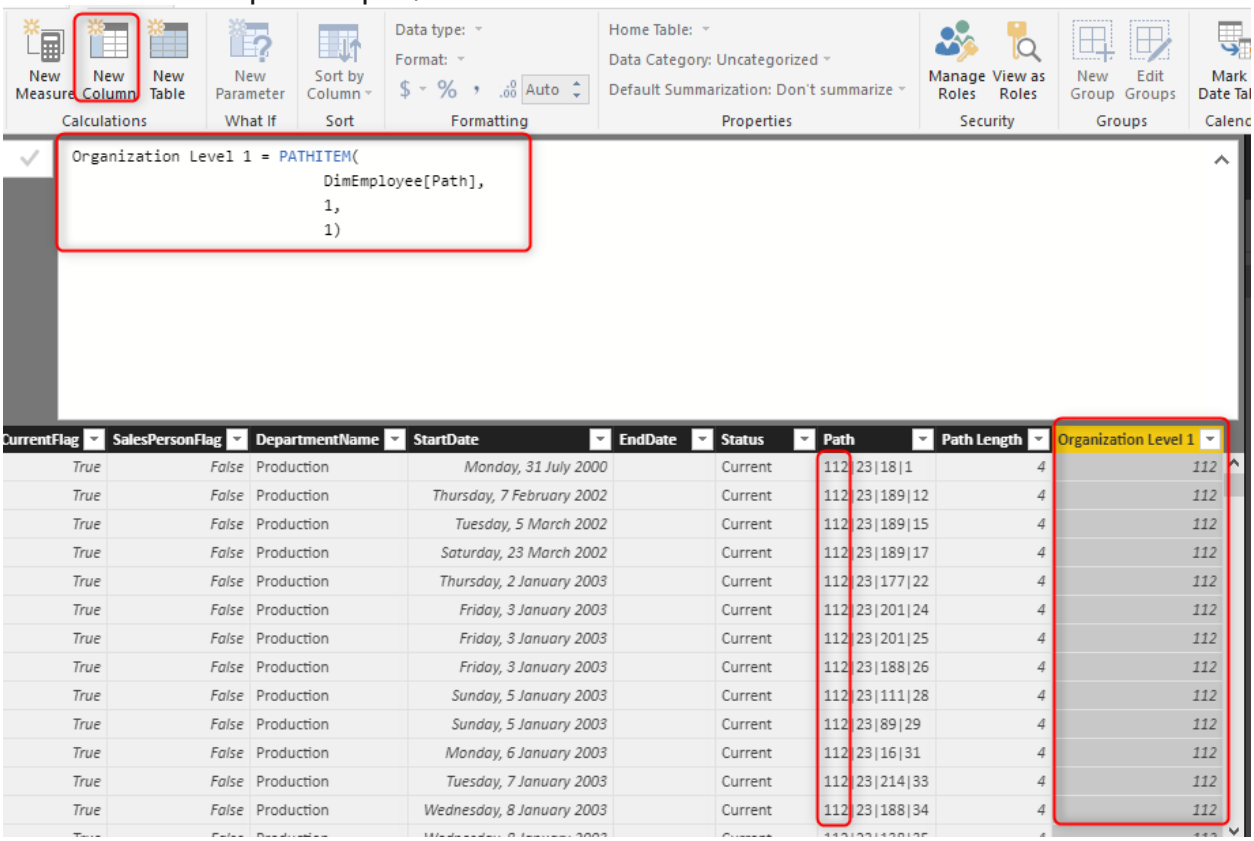
- The output of the Path function; which we can use our Path calculated column for it.
- The position of the item. Starting from 1, 1 means the highest level (big boss in the organizational hierarchy)

- The output's data type. 1 means number, 0 means text. We need a number output (to search the employeekey in the table based on that later on), so we use 1 as the input here.

The code will be a calculated column as below;

```
1 Organization Level 1 = PATHITEM(
2                        DimEmployee[Path],
3                        1,
4                        1)
```

Here is the sample output;



CurrentFlag	SalesPersonFlag	DepartmentName	StartDate	EndDate	Status	Path	Path Length	Organization Level 1
True	False	Production	Monday, 31 July 2000		Current	112|23|18|1	4	112
True	False	Production	Thursday, 7 February 2002		Current	112|23|189|12	4	112
True	False	Production	Tuesday, 5 March 2002		Current	112|23|189|15	4	112
True	False	Production	Saturday, 23 March 2002		Current	112|23|189|17	4	112
True	False	Production	Thursday, 2 January 2003		Current	112|23|177|22	4	112
True	False	Production	Friday, 3 January 2003		Current	112|23|201|24	4	112
True	False	Production	Friday, 3 January 2003		Current	112|23|201|25	4	112
True	False	Production	Friday, 3 January 2003		Current	112|23|188|26	4	112
True	False	Production	Sunday, 5 January 2003		Current	112|23|111|28	4	112
True	False	Production	Sunday, 5 January 2003		Current	112|23|89|29	4	112
True	False	Production	Monday, 6 January 2003		Current	112|23|16|31	4	112
True	False	Production	Tuesday, 7 January 2003		Current	112|23|214|33	4	112
True	False	Production	Wednesday, 8 January 2003		Current	112|23|188|34	4	112

As you can see 112 is the ID of the big boss and the first level of management in the Path column.

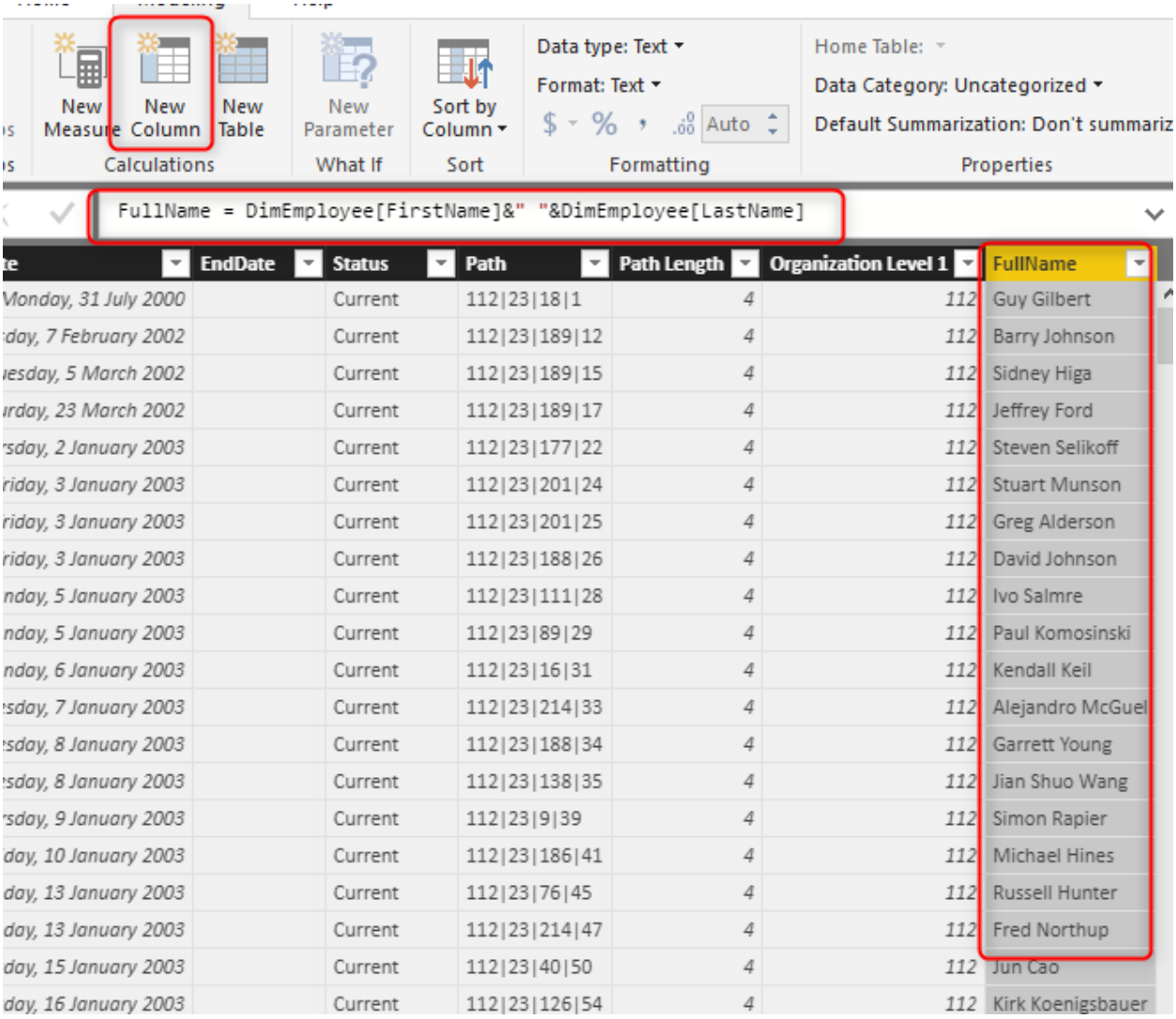
PathItemReverse; start from the lowest level

If you don't want to start from the highest level, then you can use the PathItemReverse function. Everything will be similar to using the PathItem

function; the only difference is that this time, the position starts with index 1 for the lowest level of the hierarchy.

LookupValue; to find the name of the employee

Having just ID of the manager is not usually enough. You may need to get the name of the employee too. In the DimEmployee table, we do not have a full name field. So first add a full name field as below;



EndDate	Status	Path	Path Length	Organization Level 1	FullName
Monday, 31 July 2000	Current	112|23|18|1	4	112	Guy Gilbert
day, 7 February 2002	Current	112|23|189|12	4	112	Barry Johnson
esday, 5 March 2002	Current	112|23|189|15	4	112	Sidney Higa
rday, 23 March 2002	Current	112|23|189|17	4	112	Jeffrey Ford
sday, 2 January 2003	Current	112|23|177|22	4	112	Steven Selikoff
riday, 3 January 2003	Current	112|23|201|24	4	112	Stuart Munson
riday, 3 January 2003	Current	112|23|201|25	4	112	Greg Alderson
riday, 3 January 2003	Current	112|23|188|26	4	112	David Johnson
nday, 5 January 2003	Current	112|23|111|28	4	112	Ivo Salmre
nday, 5 January 2003	Current	112|23|89|29	4	112	Paul Komosinski
nday, 6 January 2003	Current	112|23|16|31	4	112	Kendall Keil
sday, 7 January 2003	Current	112|23|214|33	4	112	Alejandro McGuel
sday, 8 January 2003	Current	112|23|188|34	4	112	Garrett Young
sday, 8 January 2003	Current	112|23|138|35	4	112	Jian Shuo Wang
sday, 9 January 2003	Current	112|23|9|39	4	112	Simon Rapier
day, 10 January 2003	Current	112|23|186|41	4	112	Michael Hines
day, 13 January 2003	Current	112|23|76|45	4	112	Russell Hunter
day, 13 January 2003	Current	112|23|214|47	4	112	Fred Northup
day, 15 January 2003	Current	112|23|40|50	4	112	Jun Cao
day, 16 January 2003	Current	112|23|126|54	4	112	Kirk Koenigsbauer

Now you can use the LookupValue function to get the full name of the employee that we found by PathItem function. LookupValue asks for three parameters;

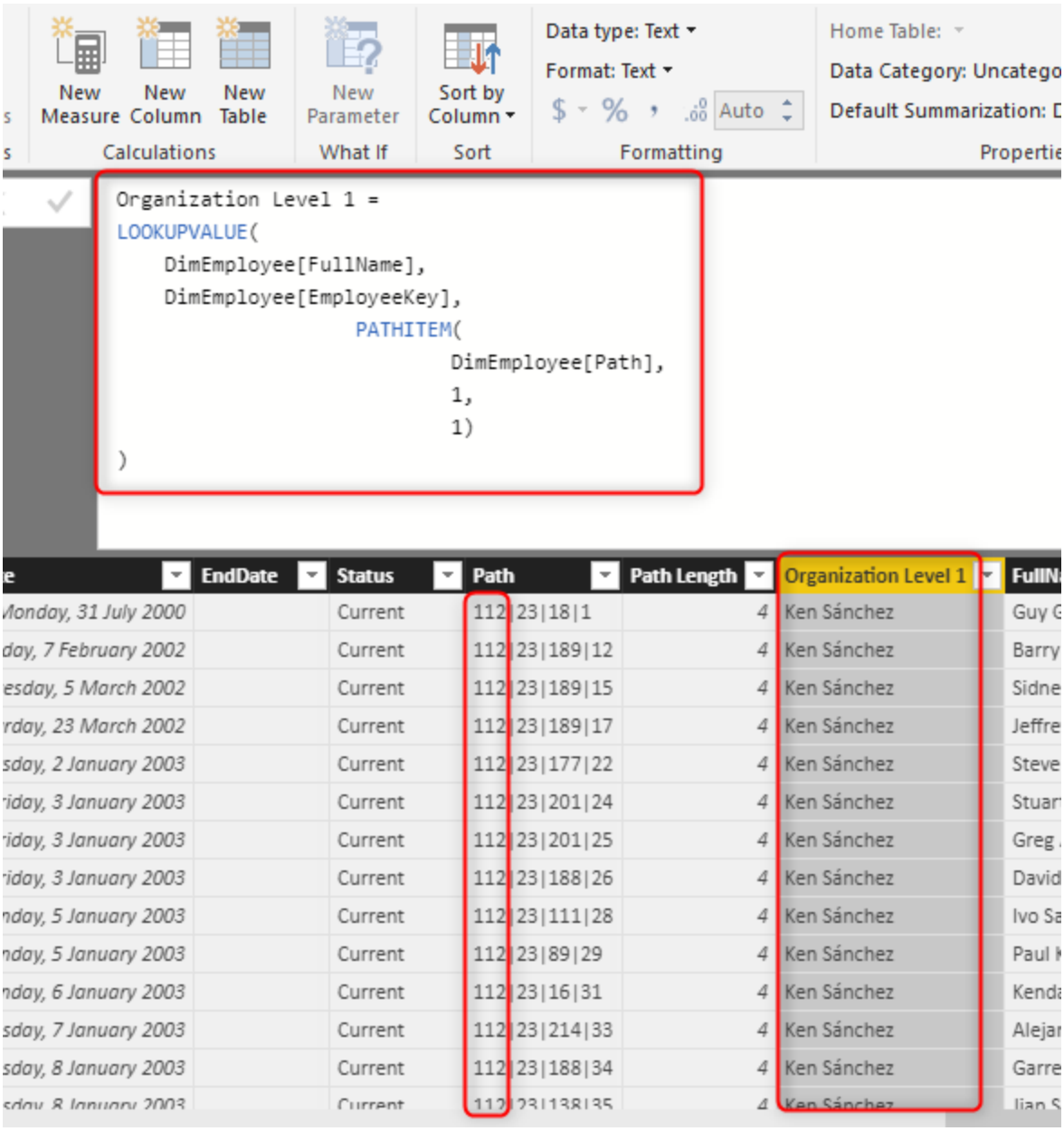
- The output column

- The column to search into for the keyword
- the keyword (keyword in our scenario is coming from the result of the PathItem function)

Here is the code altogether for organization level 1:

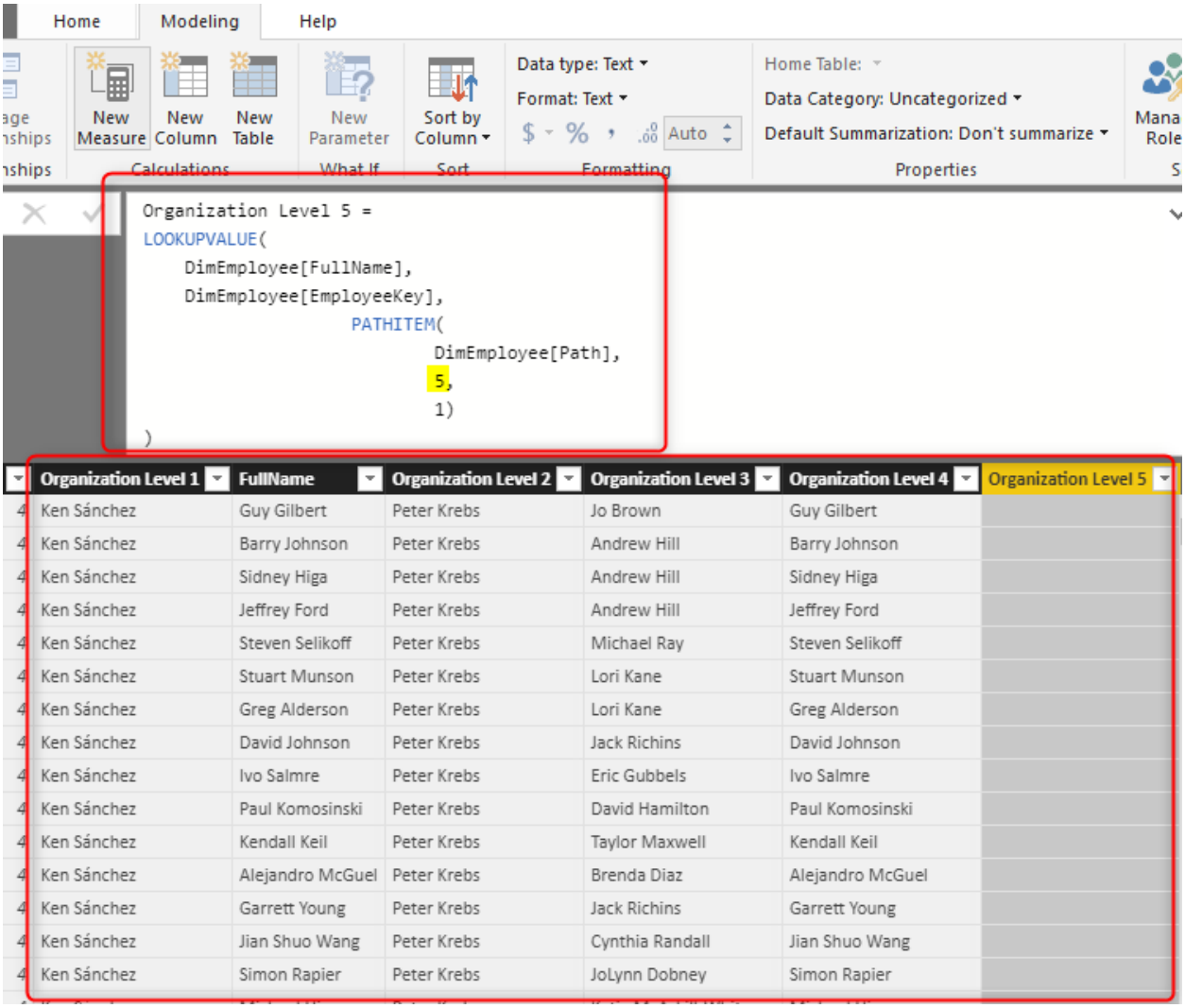
```
1 Organization Level 1 =
2 LOOKUPVALUE(
3     DimEmployee[FullName],
4     DimEmployee[EmployeeKey],
5                 PATHITEM(
6                     DimEmployee[Path],
7                     1,
8                     1)
9 )
```

and the result is as below;



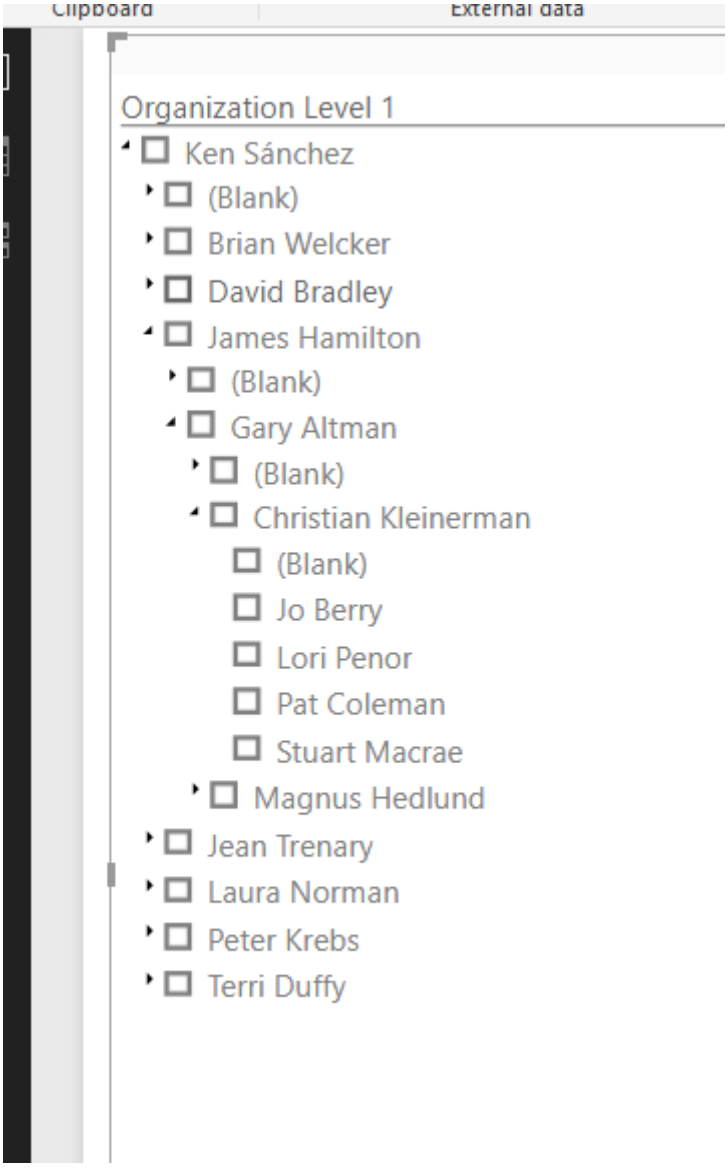
Create one column per hierarchy levels

Finally, you need to create one column per hierarchy level. All you need to do is to copy the code for PathItem and LookupValue and only change the position parameter of that. Here is the final result for five levels;



Organization Level 1	FullName	Organization Level 2	Organization Level 3	Organization Level 4	Organization Level 5
Ken Sánchez	Guy Gilbert	Peter Krebs	Jo Brown	Guy Gilbert	
Ken Sánchez	Barry Johnson	Peter Krebs	Andrew Hill	Barry Johnson	
Ken Sánchez	Sidney Higa	Peter Krebs	Andrew Hill	Sidney Higa	
Ken Sánchez	Jeffrey Ford	Peter Krebs	Andrew Hill	Jeffrey Ford	
Ken Sánchez	Steven Selikoff	Peter Krebs	Michael Ray	Steven Selikoff	
Ken Sánchez	Stuart Munson	Peter Krebs	Lori Kane	Stuart Munson	
Ken Sánchez	Greg Alderson	Peter Krebs	Lori Kane	Greg Alderson	
Ken Sánchez	David Johnson	Peter Krebs	Jack Richins	David Johnson	
Ken Sánchez	Ivo Salmre	Peter Krebs	Eric Gubbels	Ivo Salmre	
Ken Sánchez	Paul Komosinski	Peter Krebs	David Hamilton	Paul Komosinski	
Ken Sánchez	Kendall Keil	Peter Krebs	Taylor Maxwell	Kendall Keil	
Ken Sánchez	Alejandro McGuel	Peter Krebs	Brenda Diaz	Alejandro McGuel	
Ken Sánchez	Garrett Young	Peter Krebs	Jack Richins	Garrett Young	
Ken Sánchez	Jian Shuo Wang	Peter Krebs	Cynthia Randall	Jian Shuo Wang	
Ken Sánchez	Simon Rapier	Peter Krebs	JoLynn Dobney	Simon Rapier	

You can then visualize this data using any visuals, but hierarchy slicer custom visual provides a nice output like below, and it acts like a slicer too;

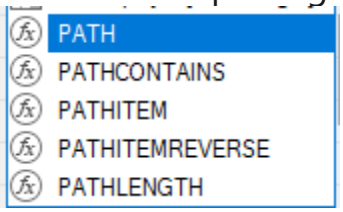


PathContains

We talked about all Parent-child functions in this example, except PathContains. PathContains is a function that searches through a path for an ID. One example usage of this function is to apply dynamic Row Level Security for an organizational chart. Here in this post, I wrote a full example of using the PathContains function for RLS.

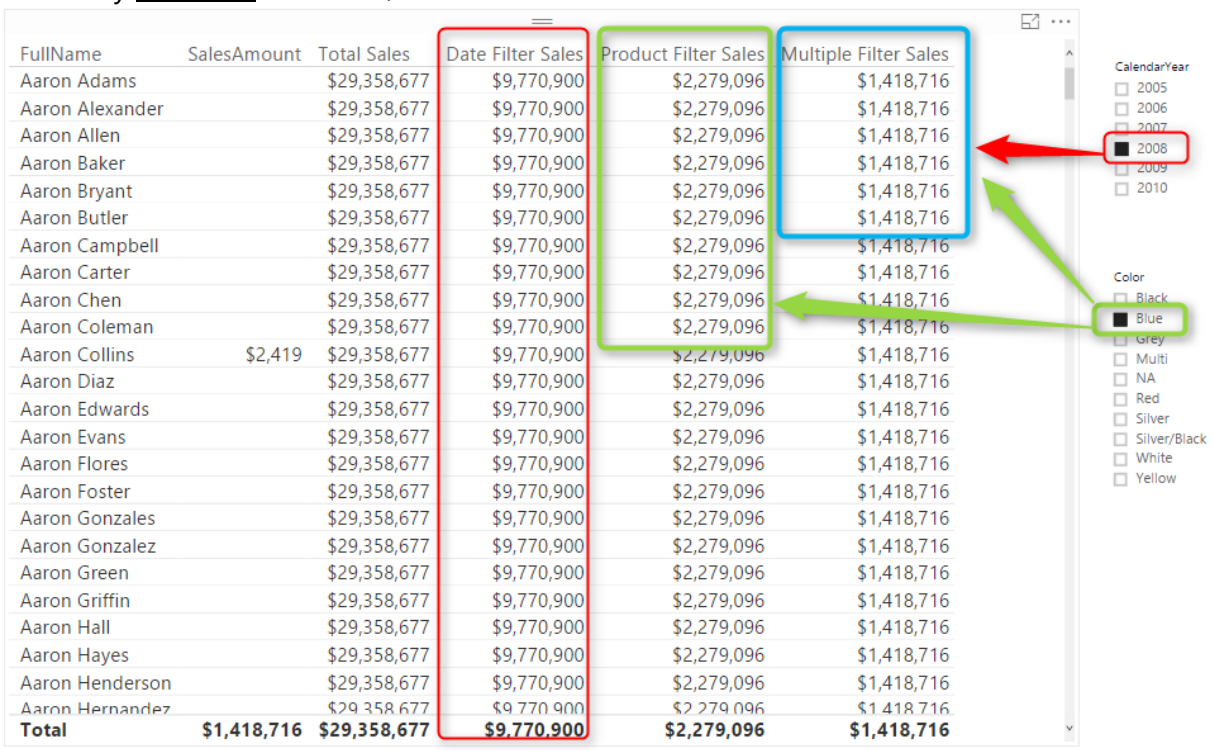
Summary

Parent-child functions in DAX are simple to use but very powerful and functional in Power BI and DAX. Using these functions will give you the ability to parse hierarchies such as organizational chart or chart of accounts in a recursive mode. In this post, you have seen an example of using parent-child functions for parsing an organizational chart.



Overwrite Interaction of Power BI with DAX

Posted by Reza Rad on Jan 24, 2017



FullName	SalesAmount	Total Sales	Date Filter Sales	Product Filter Sales	Multiple Filter Sales
Aaron Adams		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Alexander		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Allen		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Baker		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Bryant		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Butler		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Campbell		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Carter		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Chen		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Coleman		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Collins	$2,419	$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Diaz		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Edwards		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Evans		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Flores		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Foster		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Gonzales		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Gonzalez		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Green		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Griffin		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Hall		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Hayes		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Henderson		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Hernandez		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Total	**$1,418,716**	**$29,358,677**	**$9,770,900**	**$2,279,096**	**$1,418,716**

I have written a blog post while ago about using Edit Interaction to control the interaction of visuals. There are however sometimes that you want to have some of the filters to be applied on (some measures) of a single visual, but others not. In this post, I'll explain one method of writing DAX expressions in a way that overwrite the way that Power BI visuals interact. You will learn how to write a DAX expression that some filters effect that, some not. Let's see how the method works. If you want to learn more about Power BI; read Power BI online book from Rookie to Rock Star.

Prerequisite

For running example of this post, you will need the AdventureWorksDW sample database, or you can download an **Excel version** of it from here:

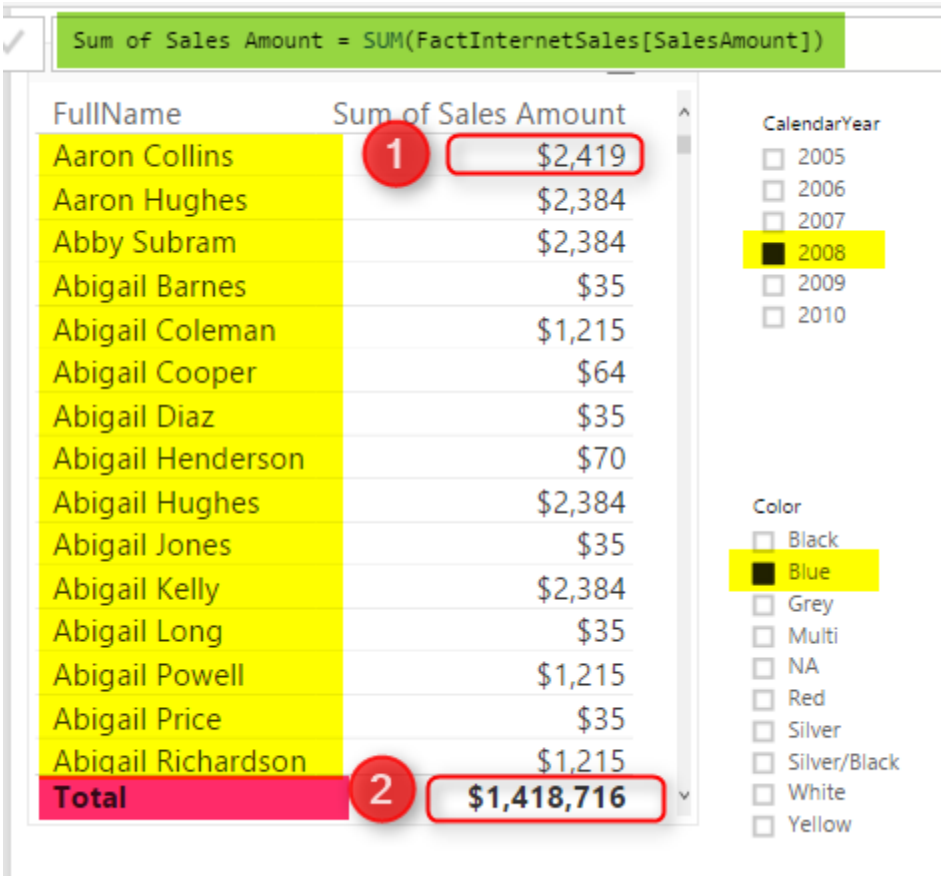
Download

Filter Context in DAX

It is impossible to start explaining this method without talking about the filter context. Filter context is everything you used for filtering and slicing and dicing in the report. As an example; create a Power BI model based on DimDate, DimCustomer, DimProduct, and FactInternetSales. Make sure that you have only one active relationship between FactInternetSales (OrderDateKey) and DimDate (DateKey). Create a Measure with DAX code below:

1 Sum of Sales Amount = SUM(FactInternetSales[SalesAmount])

Now Create a report with a Table of Full Name (from DimCustomer), Sum of Sales Amount. Also create two slicers; one for Calendar Year, and another for Product Color.



In the above screenshot, you can see that the result of Sum of Sales Amount is not always the same value. IT DEPENDS! Depends on what filter you have selected, or what values you have sliced and diced based on. For example,

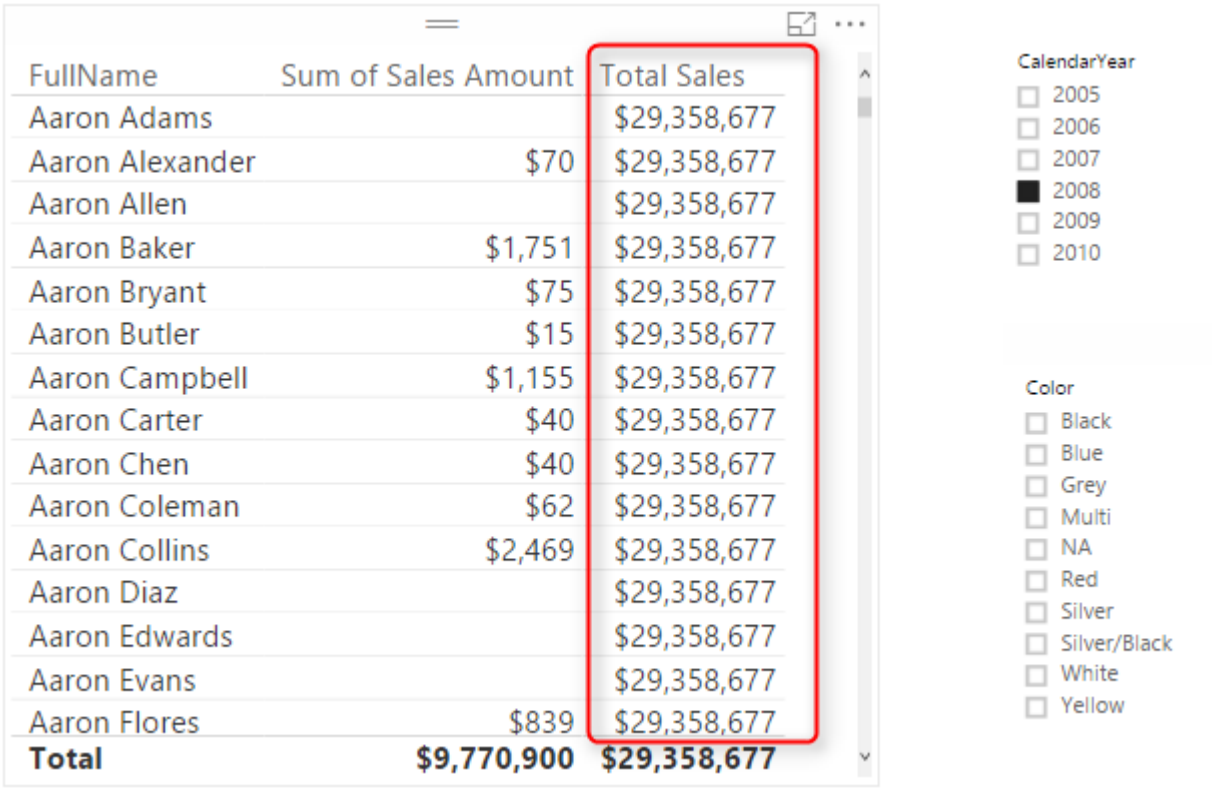
Highlight numbered 1, shows sum of Sales Amount for product Color Blue, the Calendar Year 2008, and Customer Full Name "Aaron Collins". While the highlight numbered 2, shows sum of Sales Amount for the Year 2008, and color Blue, but for all Customers. What you see here is Filter Context.
Filter Context is all filters, slicers, highlight, and slicing and dicing applied to a report or visual. Filter Context for number 1 in the above image is product Color Blue, the Calendar Year 2008, and Customer Full Name "Aaron Collins".
Everything in DAX resolves based on Filter Context and Row Context. However, there are some ways to control the context. Controlling the context means controlling the interaction of visuals. In above example, with any change in the slicer, filter context changes, and the result of Sum(SalesAmount) also changes. However, if we write a DAX expression that doesn't change with selecting a slicer, that means we have controlled the context. Let's look at some examples.

Total Sales Regardless of Filters

As an example; you can create a measure that returns total sales amount regardless of what is selected in slicers or filters, regardless of what the filter context is. For this, you can use either Iterator (SumX, MinX, MaxX,….) or Calculate function. Create a measure below;

```
1 Total Sales =
SUMX(ALL(FactInternetSales),FactInternetSales[SalesAmount])
```

In above DAX expression, **ALL** function will act regardless of filter context. No matter what Filter context is ALL will return everything, and as a result, **SUMX** will calculate the sum of SalesAmount for all rows. Here is a screenshot of the report;



FullName	Sum of Sales Amount	Total Sales
Aaron Adams		$29,358,677
Aaron Alexander	$70	$29,358,677
Aaron Allen		$29,358,677
Aaron Baker	$1,751	$29,358,677
Aaron Bryant	$75	$29,358,677
Aaron Butler	$15	$29,358,677
Aaron Campbell	$1,155	$29,358,677
Aaron Carter	$40	$29,358,677
Aaron Chen	$40	$29,358,677
Aaron Coleman	$62	$29,358,677
Aaron Collins	$2,469	$29,358,677
Aaron Diaz		$29,358,677
Aaron Edwards		$29,358,677
Aaron Evans		$29,358,677
Aaron Flores	$839	$29,358,677
Total	**$9,770,900**	**$29,358,677**

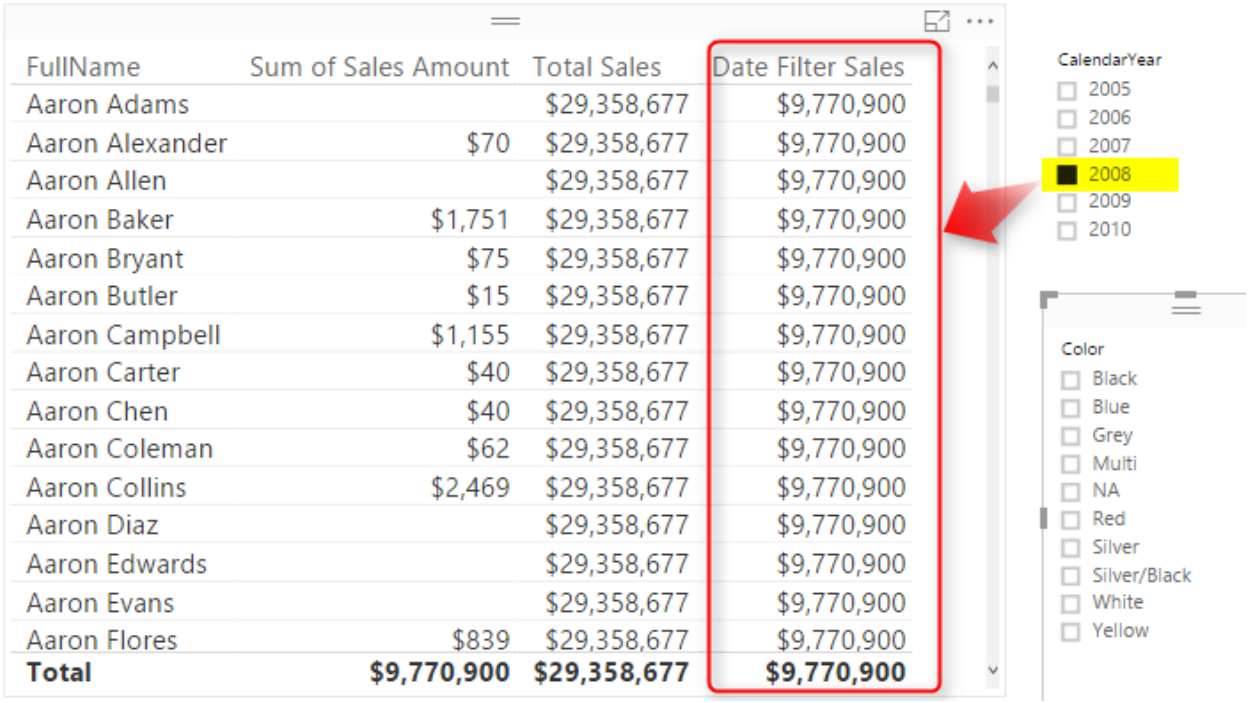
Doesn't matter what filter you select, or what slicer you click, the result for the measure is always total value. Now let's control the context a bit different.

Total Sales Filterable Only by Date Selection

Let's take one step forward with bringing one selection criteria in the measure. For this measure we want to create a Total Sales that can be only changed when a date selection happens (Year in our example), but nothing else. Because we need multiple filters now, I'll do it this time with CALCUALTE function where I can specify multiple filters. Here is the code:

```
1 Date Filter Sales = CALCULATE( SUM(FactInternetSales[SalesAmount]),
2         DATESBETWEEN(DimDate[FullDateAlternateKey],
3                 FIRSTDATE(DimDate[FullDateAlternateKey]),
4                 LASTDATE(DimDate[FullDateAlternateKey])
5         ),
6         ALL(FactInternetSales)
7 )
```

In measure above we have two filters; ALL(FactInternetSales), and DatesBetween(). DatesBetween brings everything from the FirstDate to the LastDate. FirstDate and LastDate will depend on the date selection in the slicer. As a result DatesBetween will return the filter context of date selection. However everything else will be ignored by ALL(FactInternetSales). The result will be a filter which is the junction of these two filters. Here is the result;



FullName	Sum of Sales Amount	Total Sales	Date Filter Sales
Aaron Adams		$29,358,677	$9,770,900
Aaron Alexander	$70	$29,358,677	$9,770,900
Aaron Allen		$29,358,677	$9,770,900
Aaron Baker	$1,751	$29,358,677	$9,770,900
Aaron Bryant	$75	$29,358,677	$9,770,900
Aaron Butler	$15	$29,358,677	$9,770,900
Aaron Campbell	$1,155	$29,358,677	$9,770,900
Aaron Carter	$40	$29,358,677	$9,770,900
Aaron Chen	$40	$29,358,677	$9,770,900
Aaron Coleman	$62	$29,358,677	$9,770,900
Aaron Collins	$2,469	$29,358,677	$9,770,900
Aaron Diaz		$29,358,677	$9,770,900
Aaron Edwards		$29,358,677	$9,770,900
Aaron Evans		$29,358,677	$9,770,900
Aaron Flores	$839	$29,358,677	$9,770,900
Total	**$9,770,900**	**$29,358,677**	**$9,770,900**

You can see that the value in this new measure (Date Filter Sales) changes by any selection in Calendar Year slicer, but nothing else. The result of this measure will always sum of sales amount for all transactions in the selected year. If nothing is selected in Year slicer, then this column's value will be similar to Total Sales Measure.

How if we want to enable multiple filters then? Let's look at another measure.

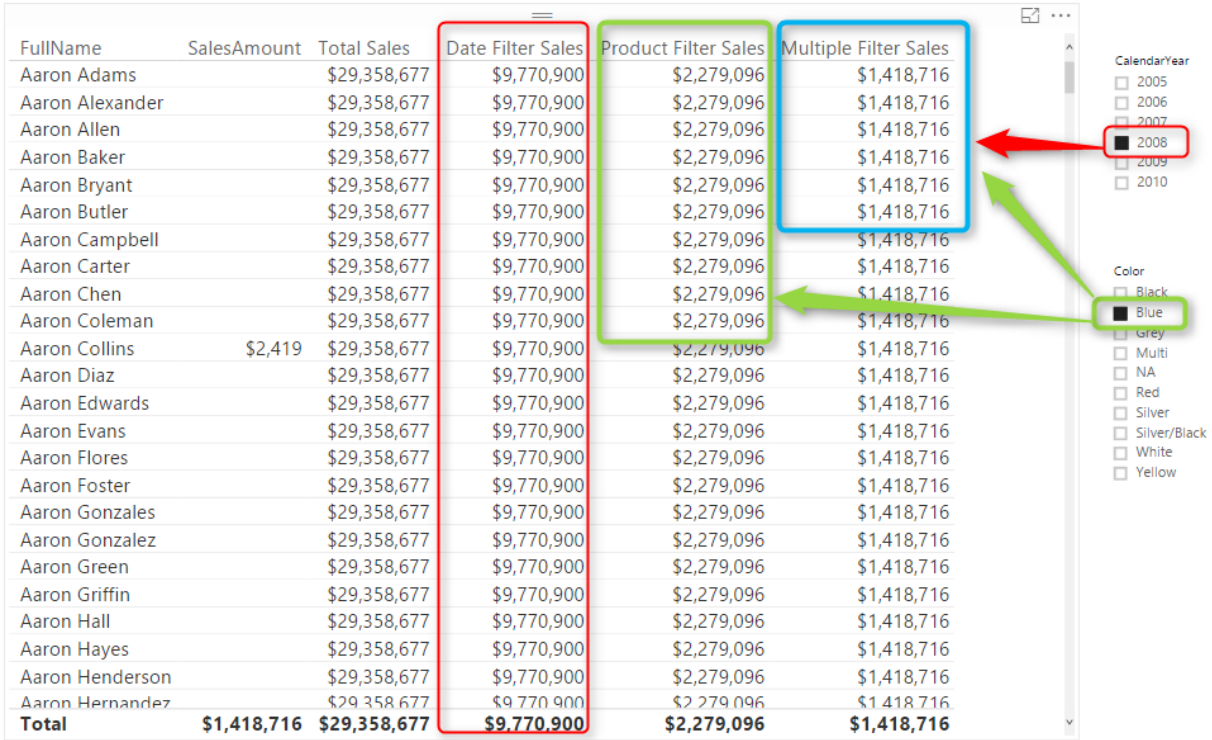
Measure Filterable with Multiple Selections

Let's go even one step further and add a measure that can be only affected by selecting values in date and product slicer, but nothing else. You know the answer already I believe. You need to add one more filter to the list of filters.

I'll do it this time with a RelatedTable function, but you can do it with other methods as well. Here is the new measure;

```
Multiple Filter Sales = CALCULATE( SUM(FactInternetSales[SalesAmount]),
        ALL(FactInternetSales),
        RELATEDTABLE(DimProduct),
        DATESBETWEEN(DimDate[FullDateAlternateKey],FIRSTDATE(DimDat
e[FullDateAlternateKey]),LASTDATE(DimDate[FullDateAlternateKey]))
)
```

The above measure is similar to the previous measure with only one more filter: RelatedTable(DimProduct). This filter will return only subset of select products. As a result of this measure Product and Date selection will be effective;



FullName	SalesAmount	Total Sales	Date Filter Sales	Product Filter Sales	Multiple Filter Sales
Aaron Adams		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Alexander		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Allen		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Baker		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Bryant		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Butler		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Campbell		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Carter		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Chen		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Coleman		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Collins	$2,419	$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Diaz		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Edwards		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Evans		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Flores		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Foster		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Gonzales		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Gonzalez		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Green		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Griffin		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Hall		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Hayes		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Henderson		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Aaron Hernandez		$29,358,677	$9,770,900	$2,279,096	$1,418,716
Total	**$1,418,716**	**$29,358,677**	**$9,770,900**	**$2,279,096**	**$1,418,716**

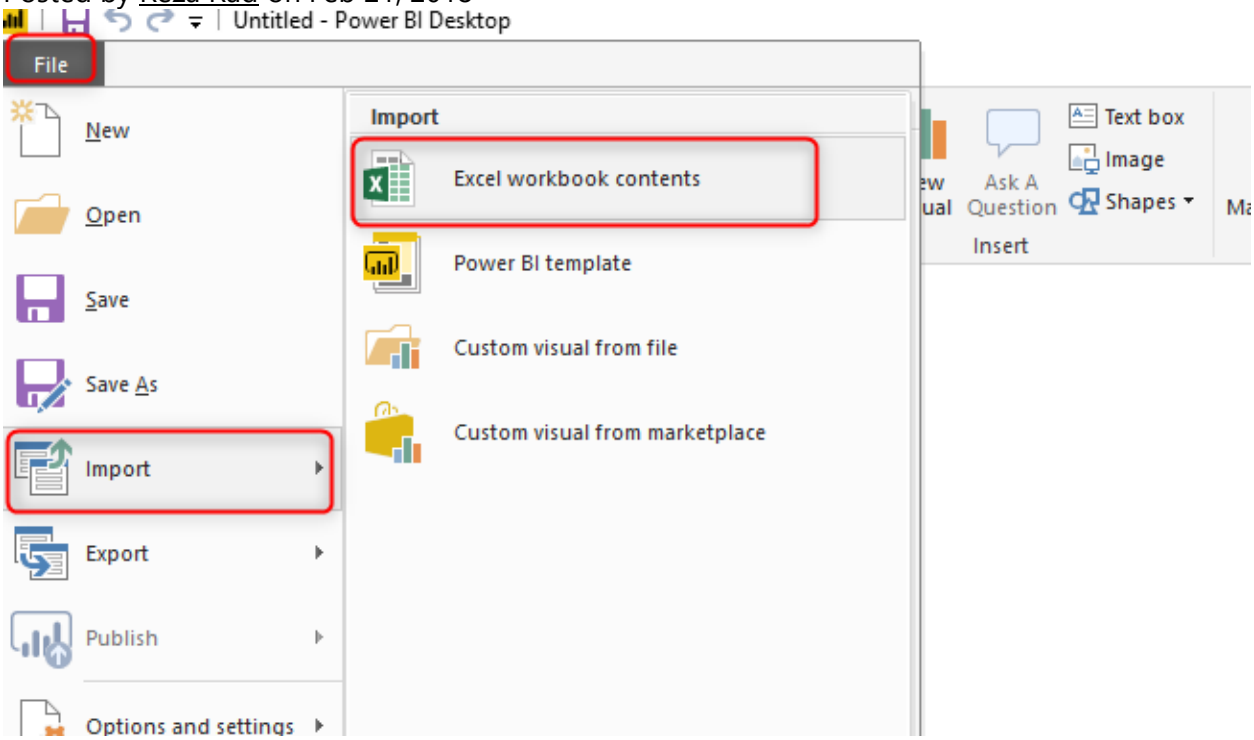
Summary

As you can see simply with DAX expressions, you can control the filter context, or in other words, you can control the interaction in Power BI. Note that you can write DAX expressions in many different ways, the expression above is not

the only way of controlling filter context. Iterators and Calculate function can be very helpful in changing this interaction.

Power BI Issue Fix: Import from Excel Workbook Contents; Password Protection Failure

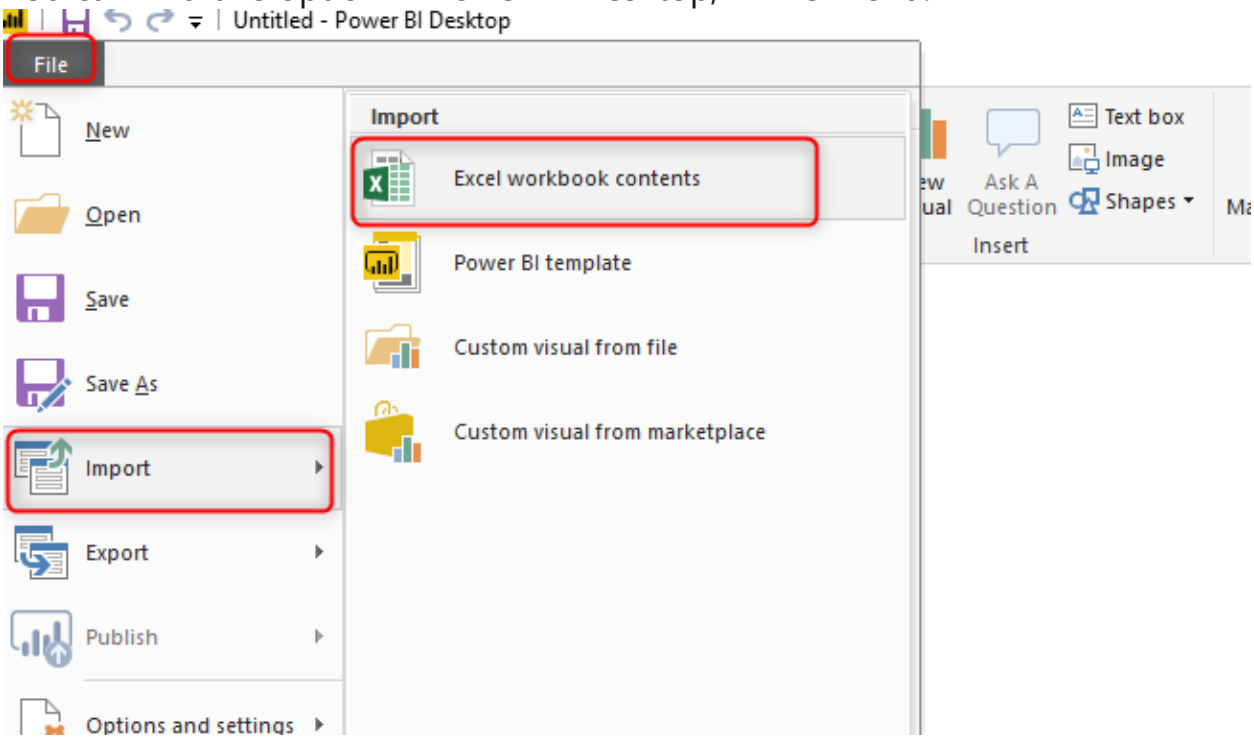
Posted by Reza Rad on Feb 21, 2018



One of the great features of Power BI Desktop is the ability to import your entire Power Pivot Excel model into it. This is an awesome feature because you can import the entire model including tables, relationships, calculations, and hierarchies into the Power BI. This is a great migration feature from Excel to Power BI. However, this feature occasionally doesn't work as expected. One of these situations is when your Excel file is password protected. In this post I'll show you what happens when your Excel file is password protected, what error do you receive in Power BI, and how to fix this issue and continue successfully with importing your model into Power BI.

Import Excel Workbook Content

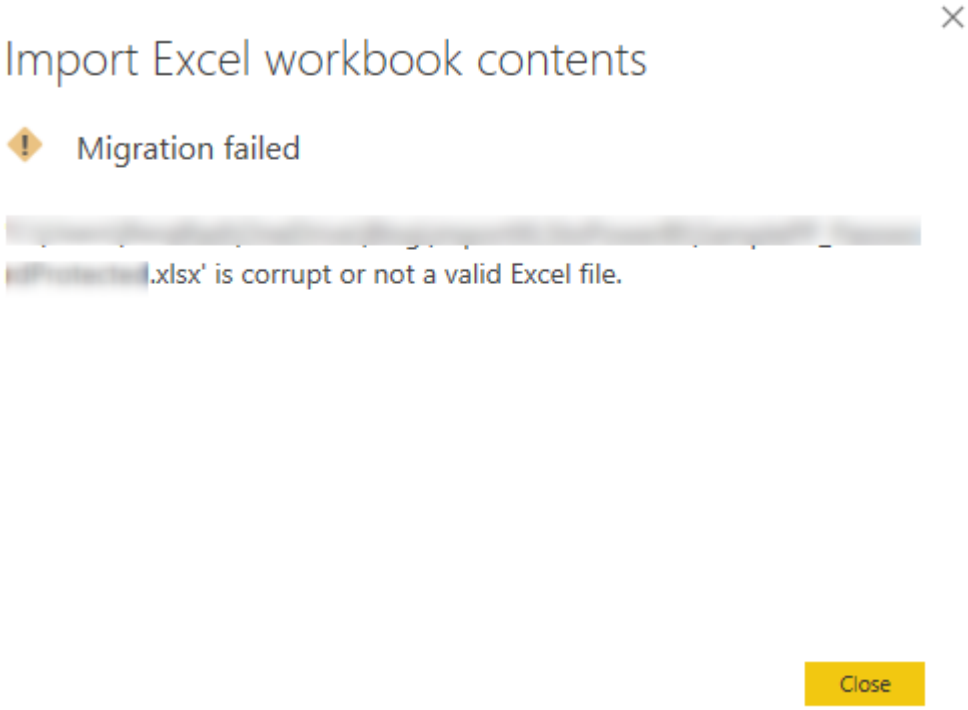
Getting data from an excel file is different from this option that we are talking about now. When you get data from an excel file, you only care about the data stored in the Excel file. However, sometimes you have already an Excel file with Power Pivot model in it, this Excel file might have been through the development process and maintained by a business user from finance team for about two years already. It would take a really long time to re-create such model in Power BI from scratch. Instead, you would like to migrate it to Power BI. The process of migration from an Excel Power Pivot model to Power BI exists, fortunately, and this process is called Import Excel Workbook Content. You can find this option in Power BI Desktop, in File Menu.



This process usually is smooth and straightforward, however, sometimes you get errors through this process. Below is an example;

Error: '*.xlsx' is corrupt or not a valid Excel file

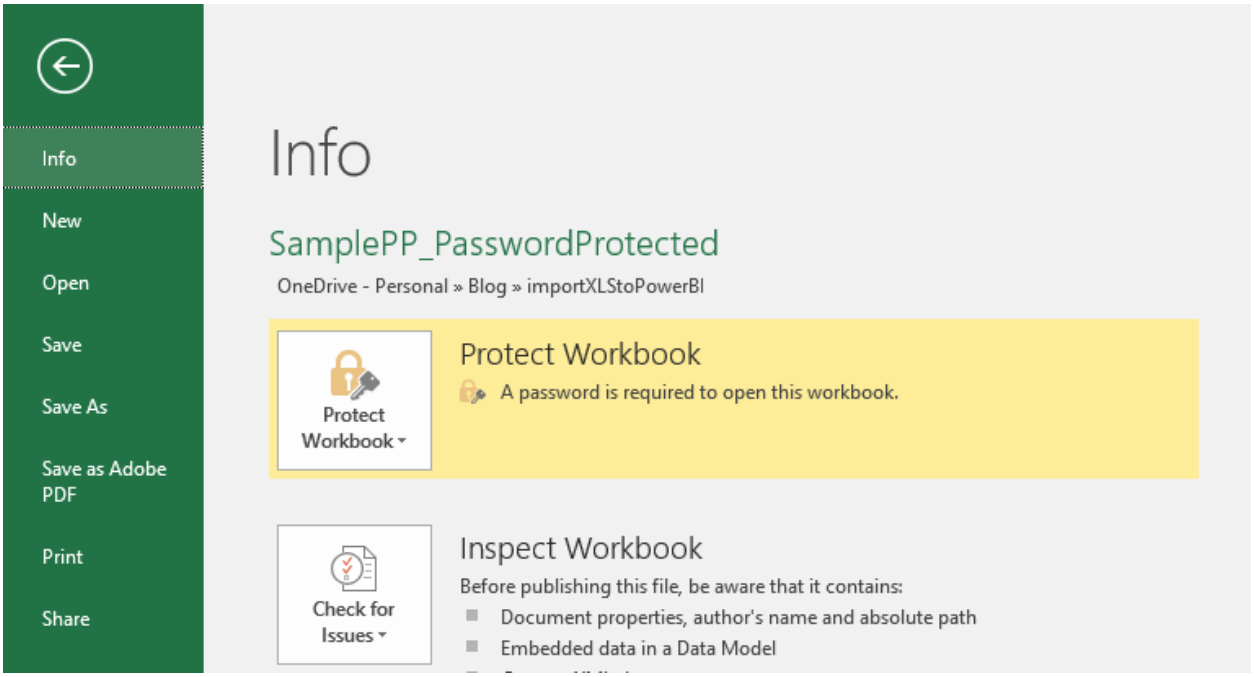
Sometimes the errors through the process are not much clear. For example, you may receive the error below, and this is happening when your Excel file is working pretty fine in Excel.



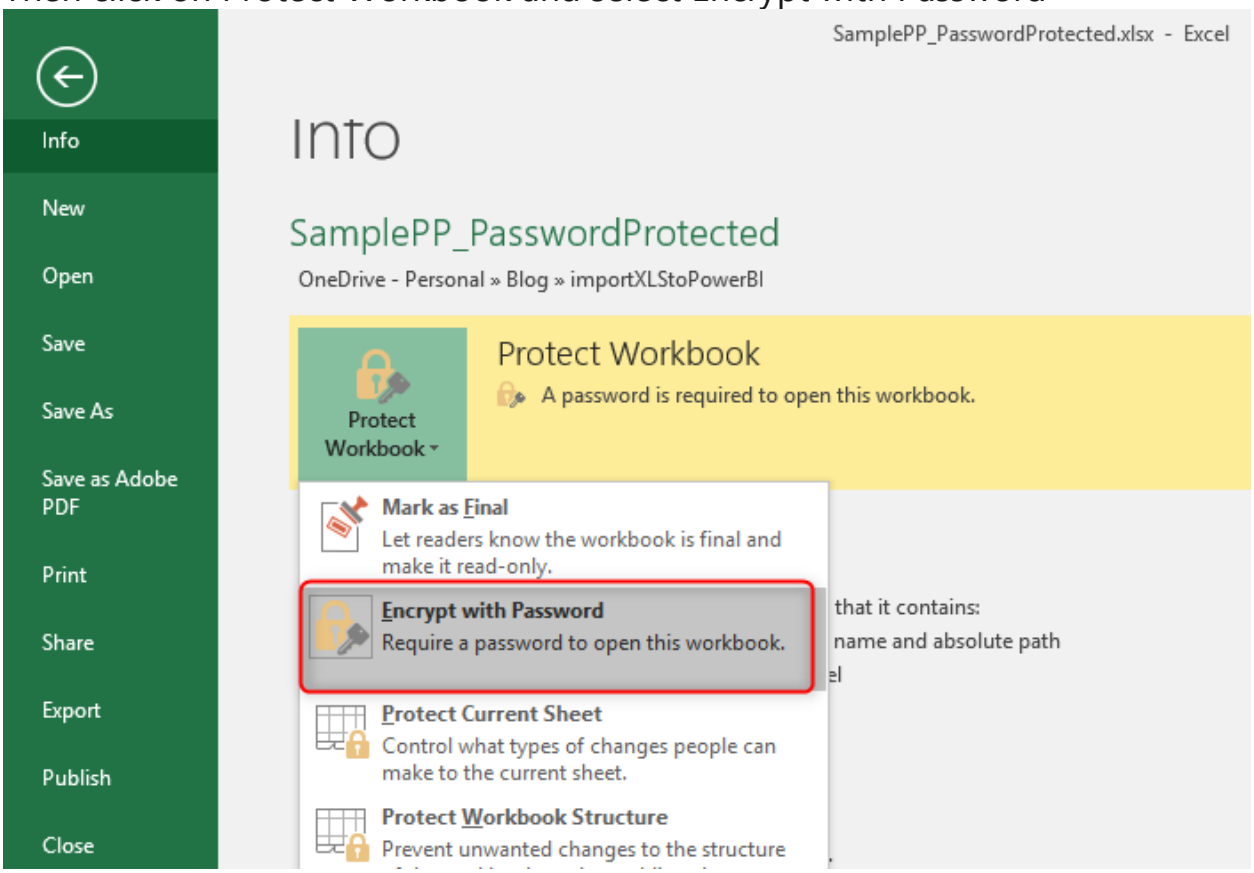
One of the reasons for this error is actually; Password Protection of the Excel File

Remove the Password Protection from the excel file

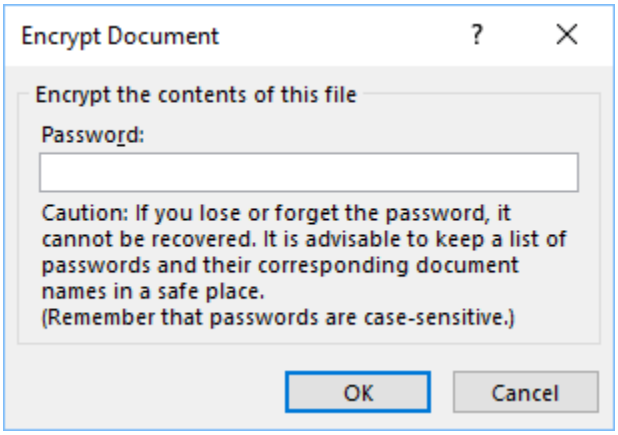
I have seen that one of the reasons for this error is that the Excel file is password protected. If your file has the same situation, then you can remove the password through this process:
Open the Excel file in Microsoft Excel, then go to File Menu,



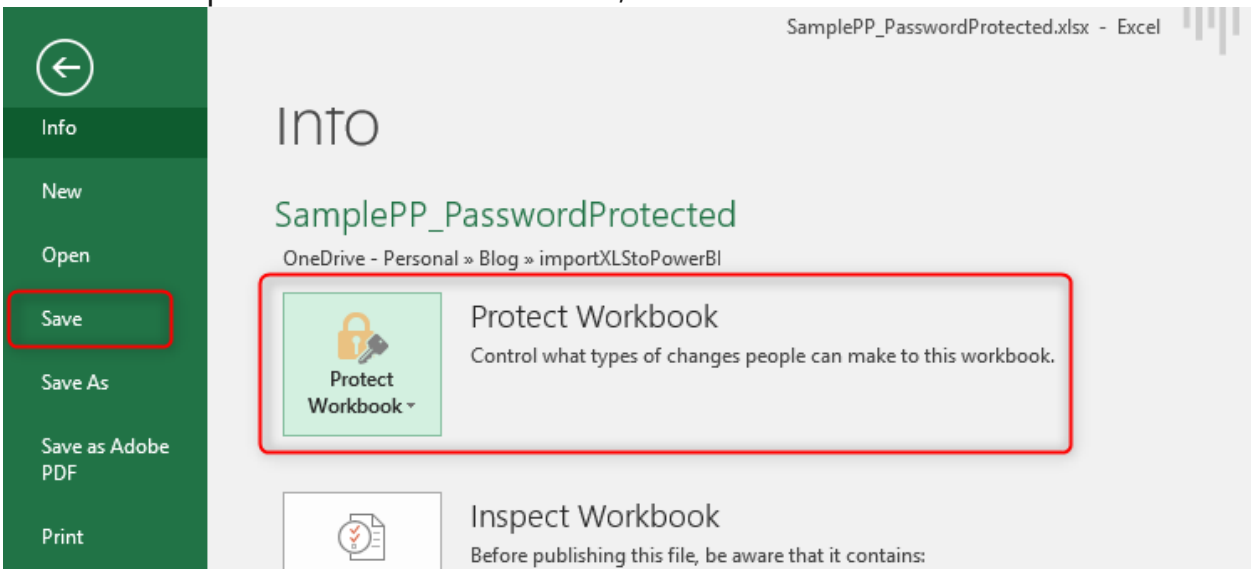
Then click on Protect Workbook and select Encrypt with Password



In the new Password window, remove the password

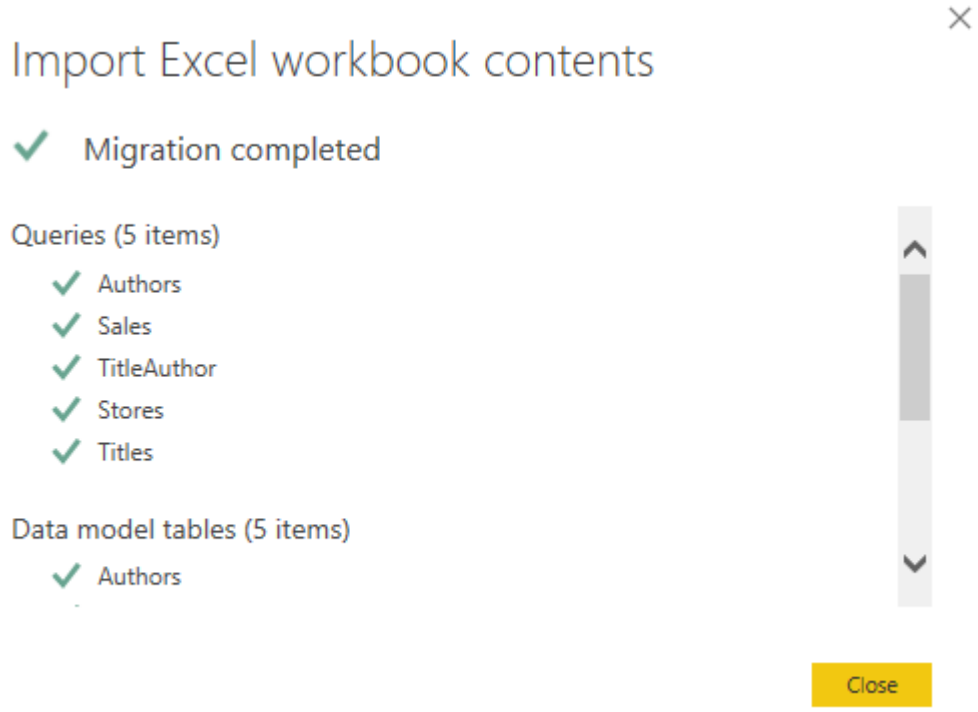


click ok. The password is now removed, save the file



Import into Power BI

After removing the password, you can try to import it again; this time migration would complete successfully.

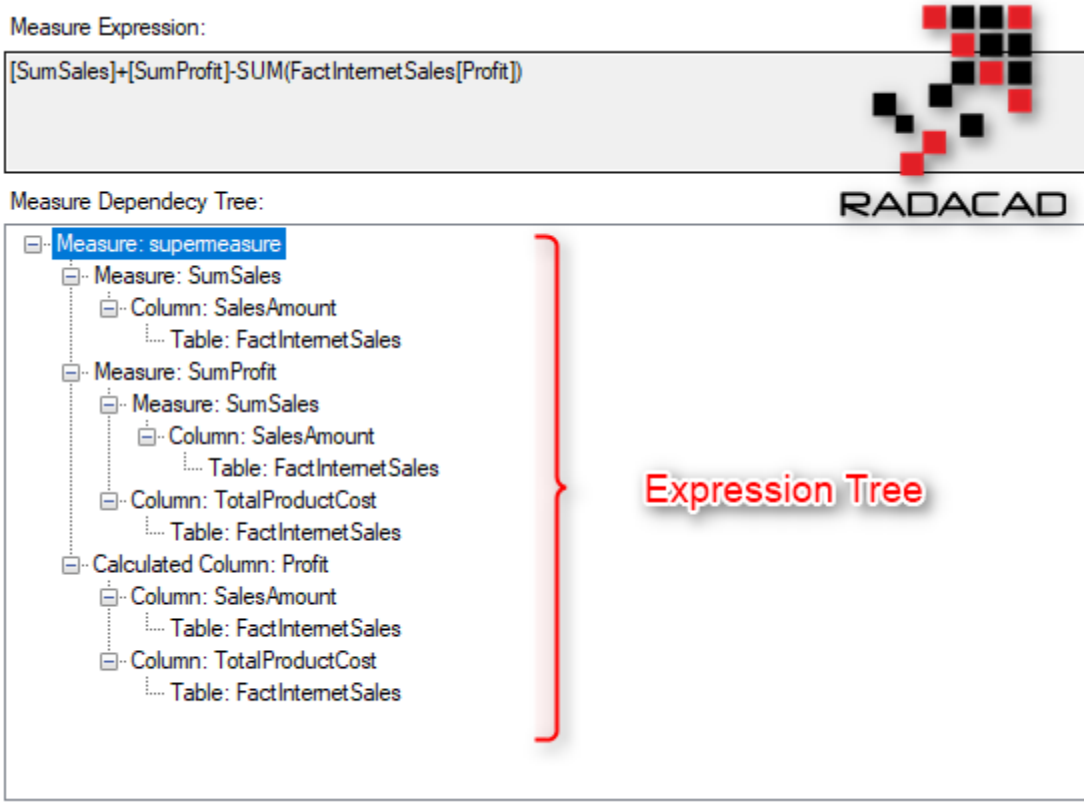


Note: there might be other reasons that also fails the migration. Password protection is not the only reason for failure, but it is one of them.

Part V: A Tool to Help: Power BI Helper

Expression Dependency Tree: New Feature of Power BI Helper

Posted by Reza Rad on Nov 29, 2017



I'm excited to share the news with you that we have added a new feature in Power BI Helper; Expression Tree. Expression Tree will expand the tree of expression for a Measure or calculated column, you can see what other measures are used to create this expression, and where other measures, calculated columns, or even normal columns are located (in which table). This feature is in addition to previous two features of this tool which were; Showing tables and fields used in visualization pages of a Power BI Report, and ability to search for a column or table that used in visualization pages of a report. In this post, I'll explain how this new feature works.

Defining the Problem

You have a Power BI with many measures that are referenced each other; it is not easy to find out which measure is used to create which calculation. If you want to go through the list of all measures, columns, and calculated columns used in a measure it may take hours to find it. Here is an example of a measure's code;

```
1 supermeasure = [SumSales]+[SumProfit]-SUM(FactInternetSales[Profit])
```

This measure is sourced from two other measures and also a column in a table. Finding the column was easy, however, finding the description of other measures are not, you need to go to Power BI Desktop, select the other measures one by one and see the expression for those, and it is very likely that those measures also have used other measures inside their expressions;

```
1 SumProfit = [SumSales] - SUM(FactInternetSales[TotalProductCost])
```

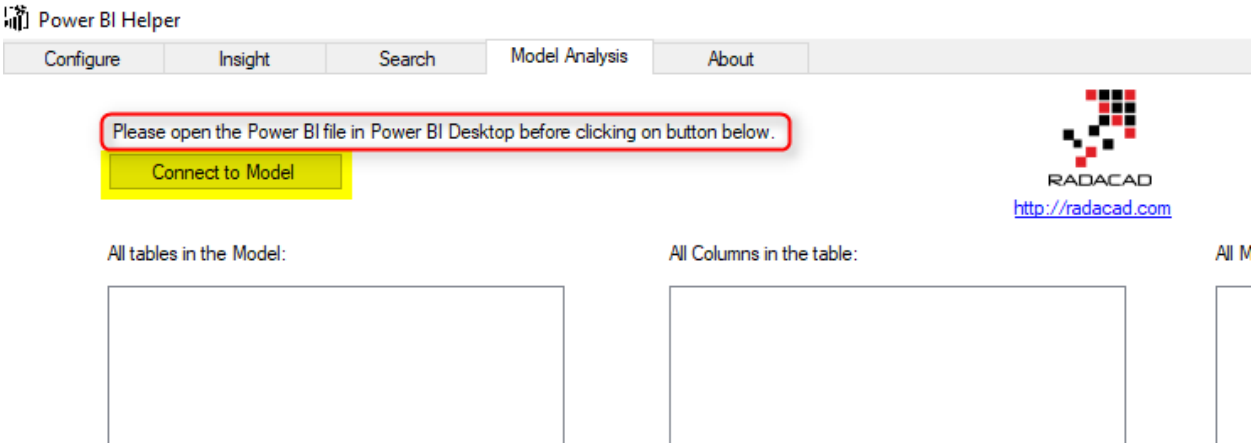
In real-world scenarios of using Power BI and DAX, you end up with a big list of measures and expressions. Finding the expression tree is always a challenge!

Download Power BI Helper

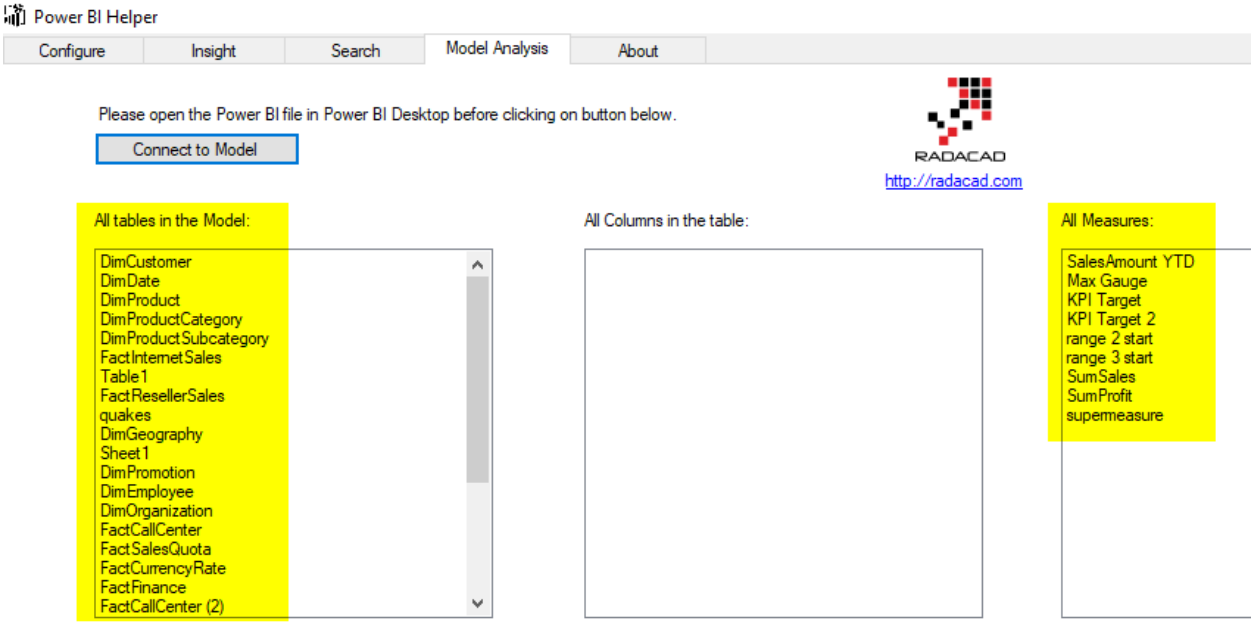
Let's now have a look at this functionality in Power BI Helper.

Connect to the Power BI Model and Get List of Tables, Columns, and Measures

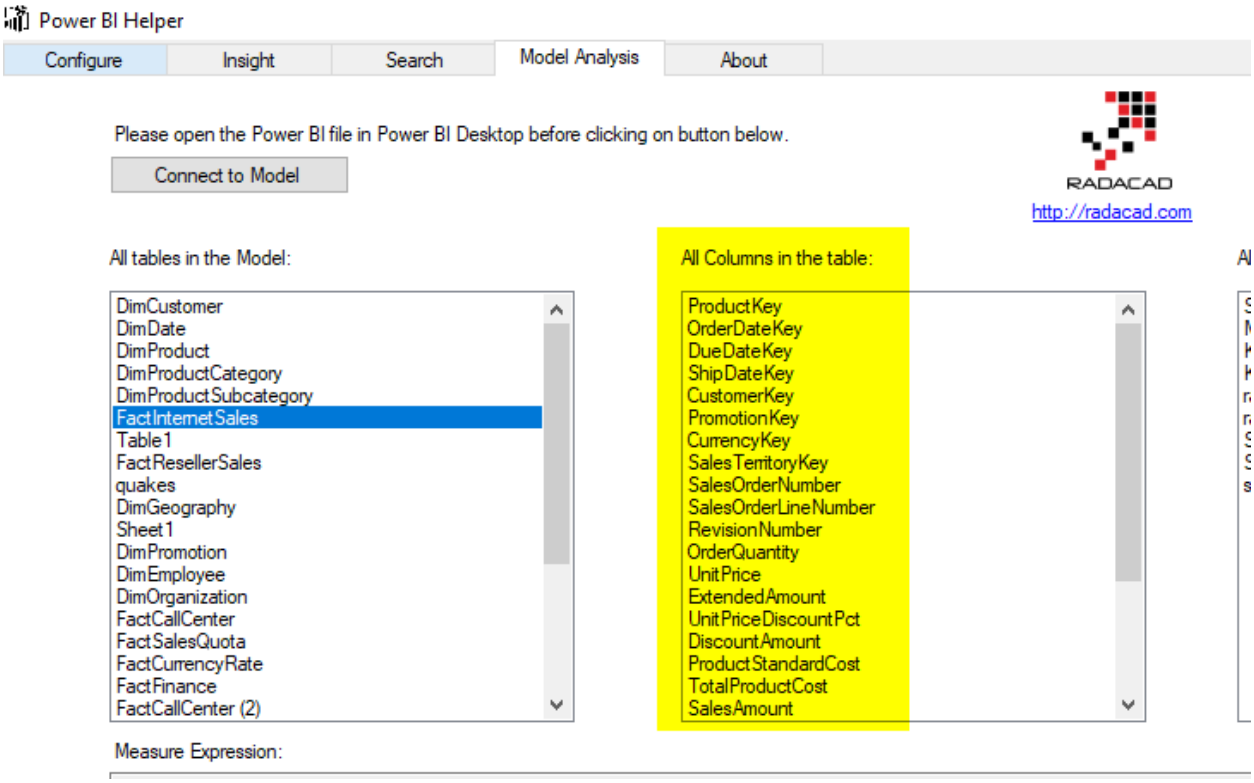
For this version of Power BI Helper, you need to have Power BI Desktop file opened on your machine, when it is opened, you can connect to that model;



After connecting to the model, you will see the list of all tables in the model and all measures;

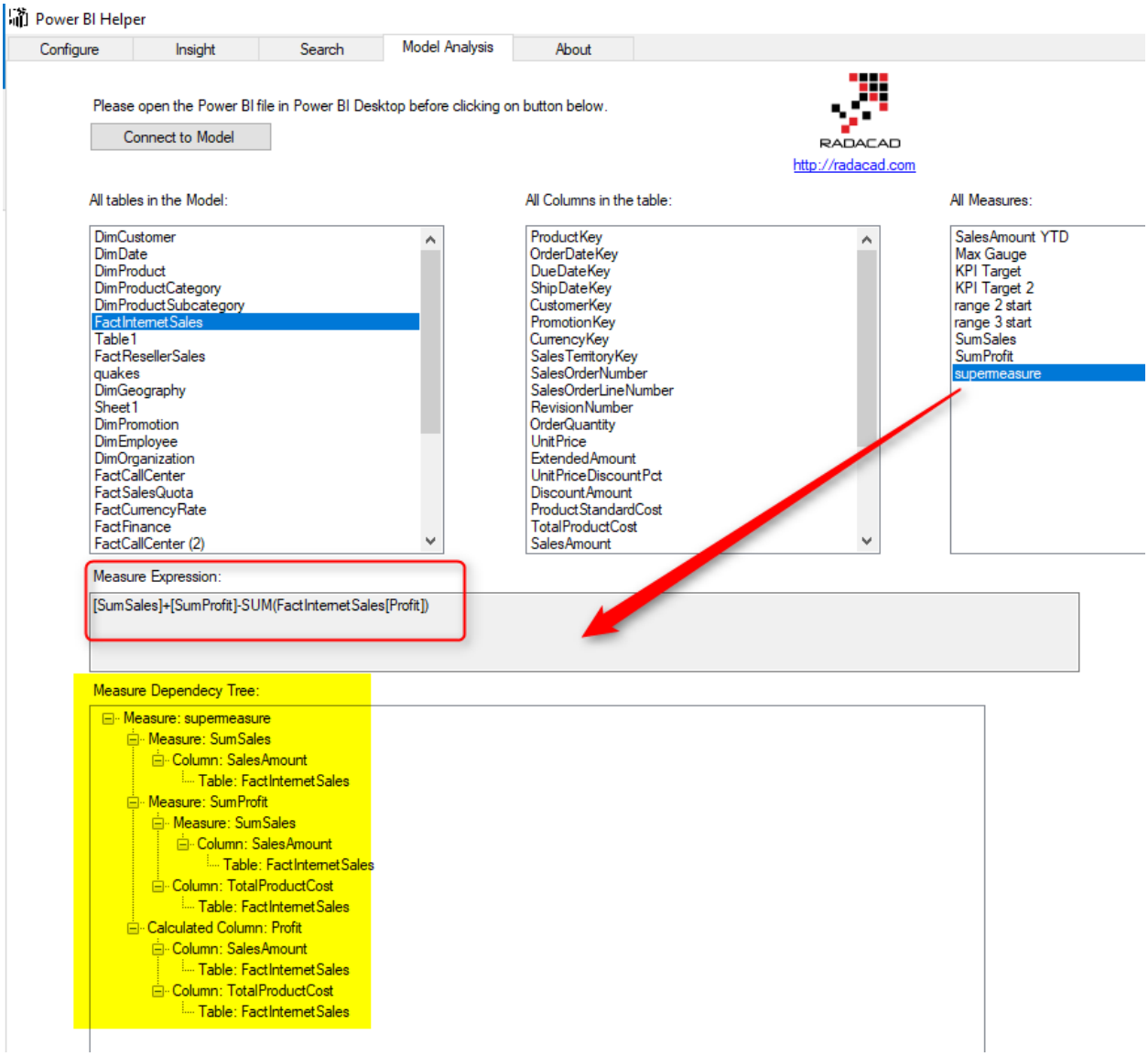


With selecting every table, you can see the list of all columns under that table too;

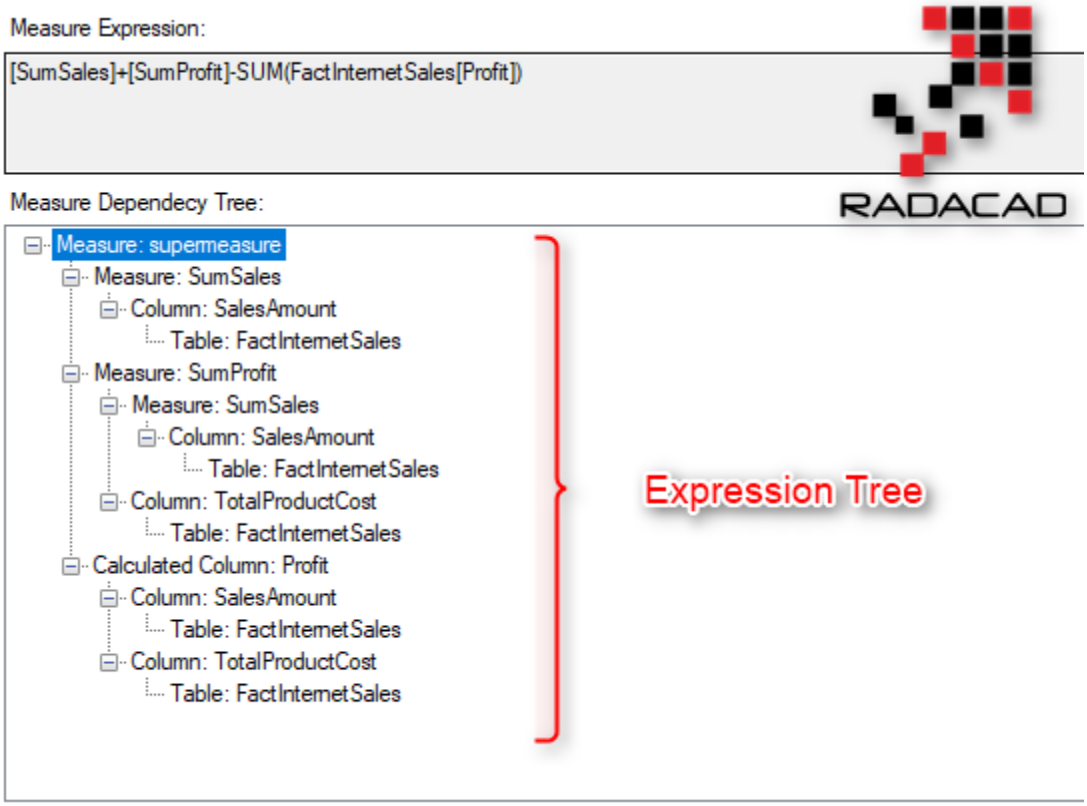


View the Measure's Expression Tree and Expression Line

If you click on any measure in the list of measures, the expression of that will be shown in the Measure Expression section, and the Measure Dependency Tree will be showed in a tree-style visualization;



Now with this expression tree, you can easily learn what measures, calculated columns are, or columns from tables are participated in creating this measure;



Other Features of Power BI Helper

If you want to learn about other features of Power BI Helper, read the posts below;

Power BI Cleanup Tool; Time Saving with Power BI Helper

Searching a Field or Table in Power BI Visualization: Power BI Helper

Summary

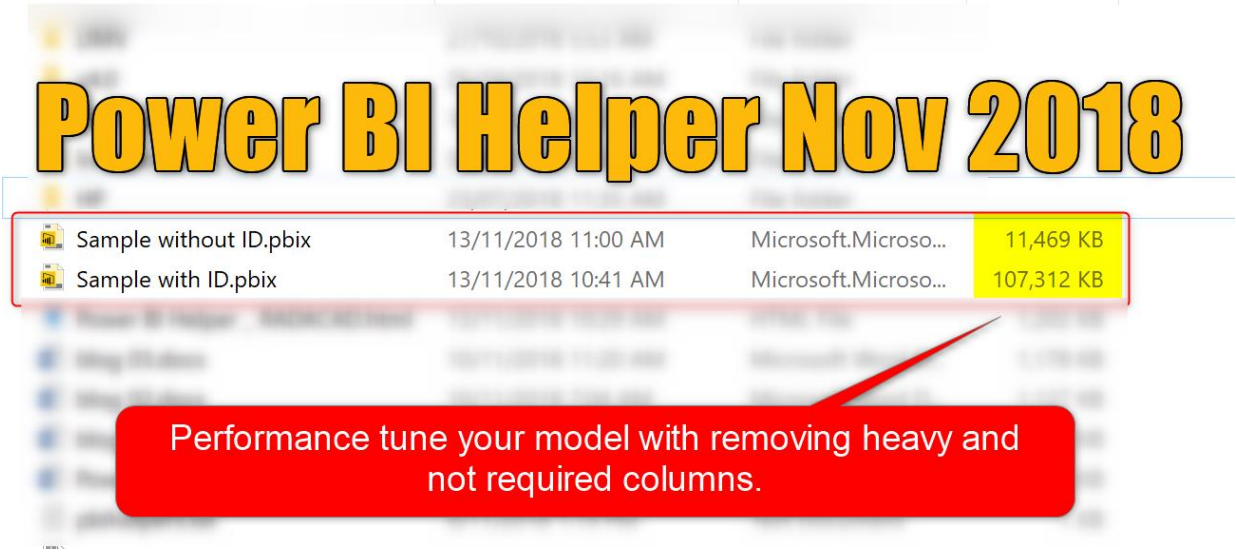
This is the version 0.3 of Power BI Helper, and still, long list of upcoming features exists. The feature of the expression tree is still under test; it will help developers to find the tree of expression dependency easier. It will help new developers in your Power BI project to get along with your model and report faster, rather than spending lots of time and money to get familiar with it. Let us know if you have any suggestions for upcoming features.

We need your help

If you like to help us, please test this with your Power BI files and let us know the result, this tool is under test, and any help from you is the most welcome

Find the Most Expensive Columns for Performance Tuning, Bookmarks, and more with Power BI Helper Version November 2018

Posted by Reza Rad on Nov 13, 2018



We are excited to share you the November 2018 version of Power BI helper with many interesting features. With this version, you get the list of all pages and list of all bookmarks. You can find all bookmarks related to a page. You can also through this version find out the data columns that consume most of the space in the data model. We also added the ability to do update to the latest version through the application itself. Let's check out these new features.

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

- Version 4.0: Beautify M Script and Row Level Security Documentation

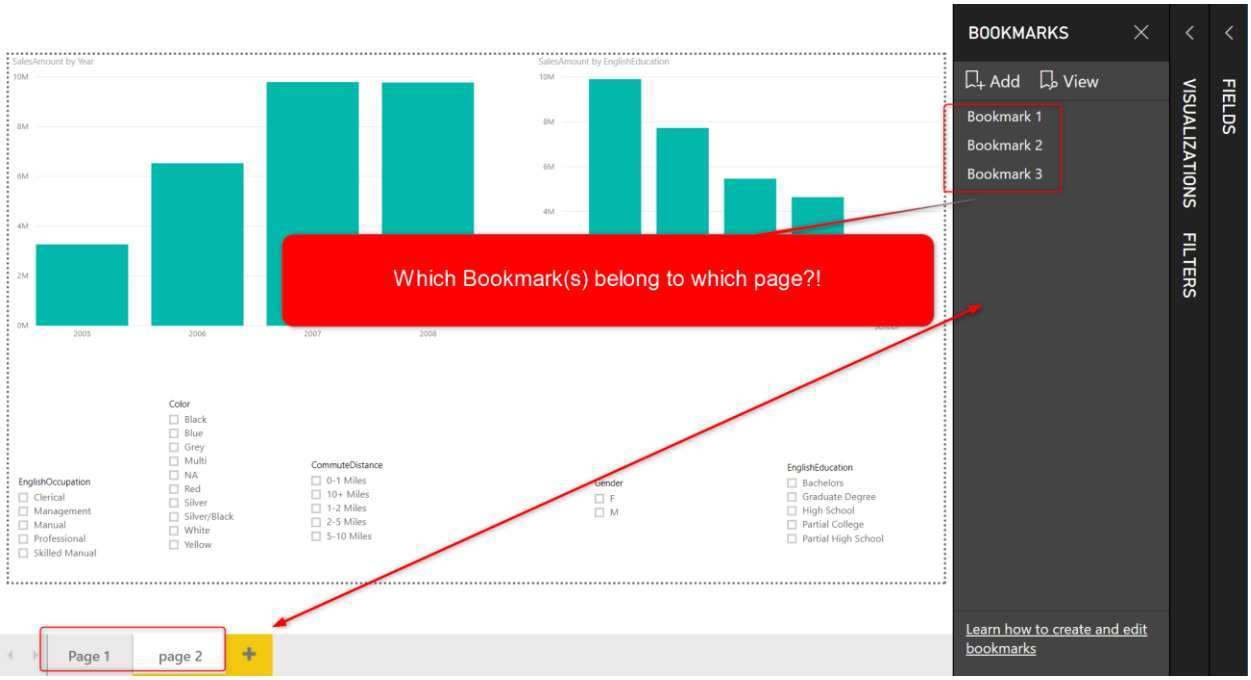
Download

To Download Power BI Helper, click here:

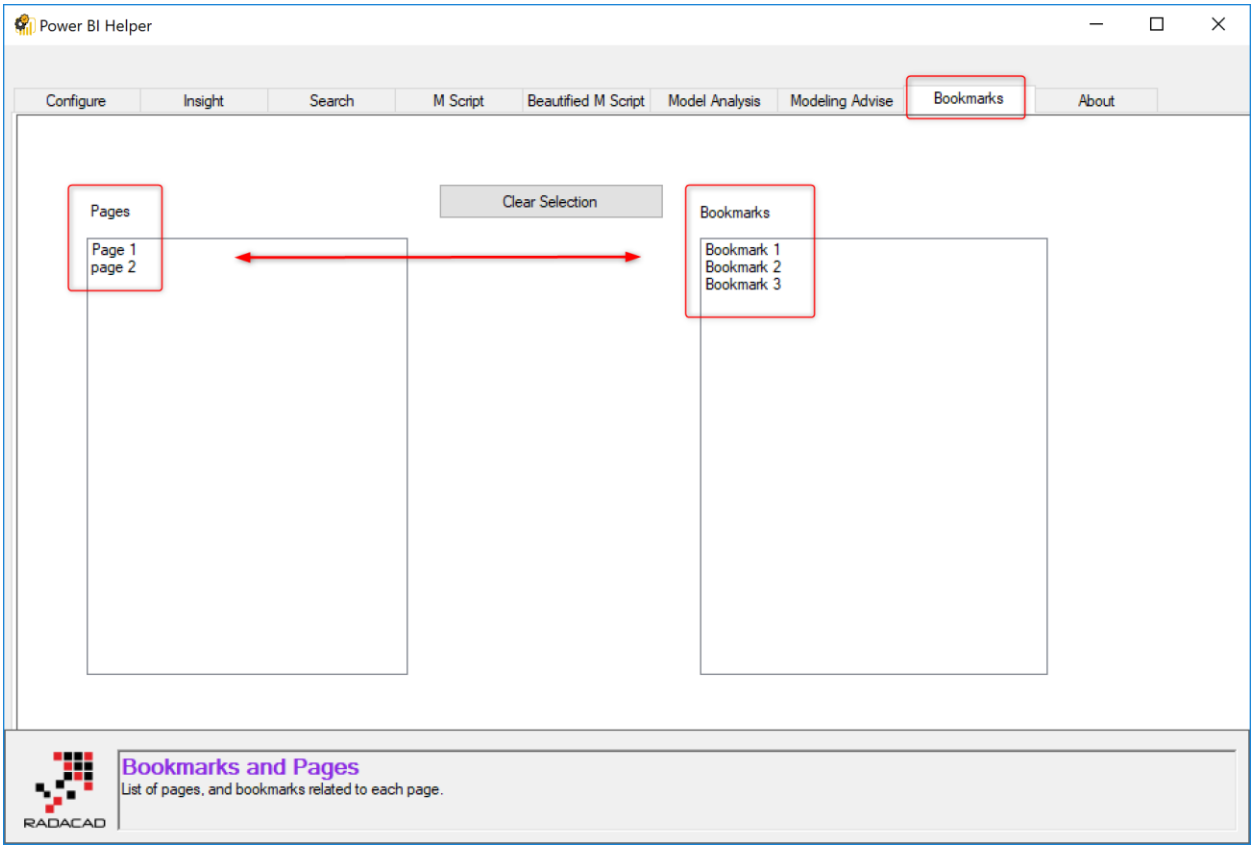
Download Power BI Helper from here.

Bookmarks and Pages

If you have multiple bookmarks saved from different states of a Power BI page, and you have many pages, and many bookmarks, there is no way to find out all bookmarks pointing to a specific page, unless checking them one by one.

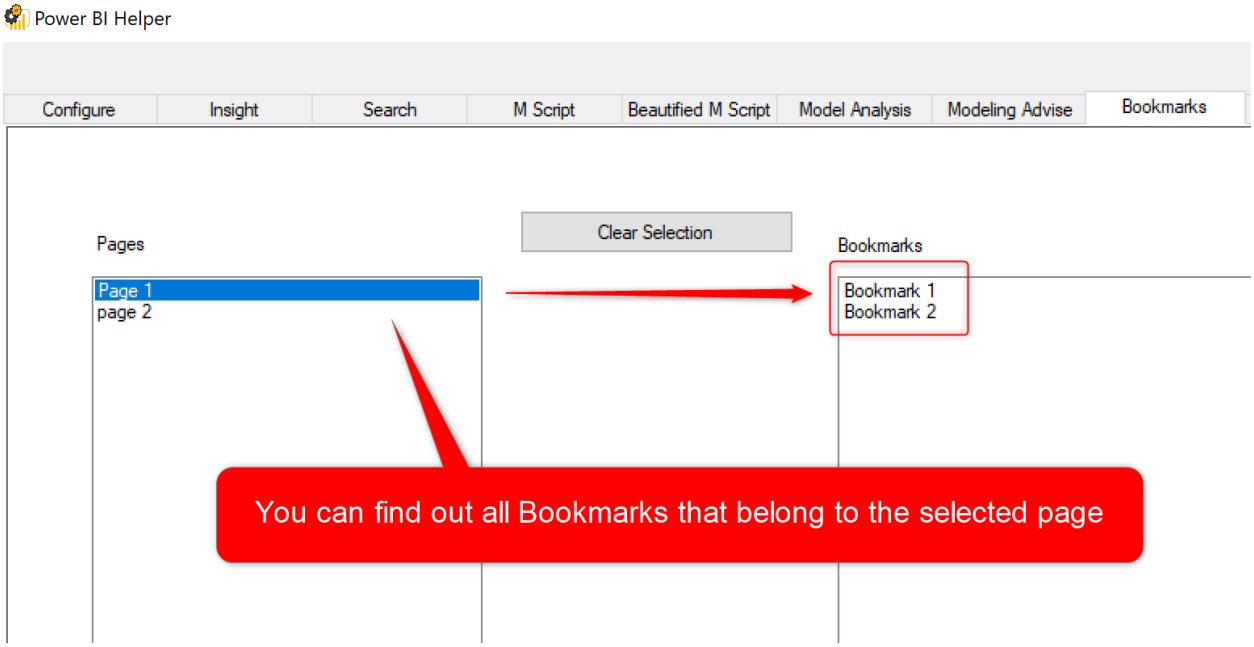


This can be important information to know, because you may want to update all bookmarks related to one page or do something with them. We implemented a feature in Power BI Helper that gives you the full list of all pages and all bookmarks. This is in the Bookmarks

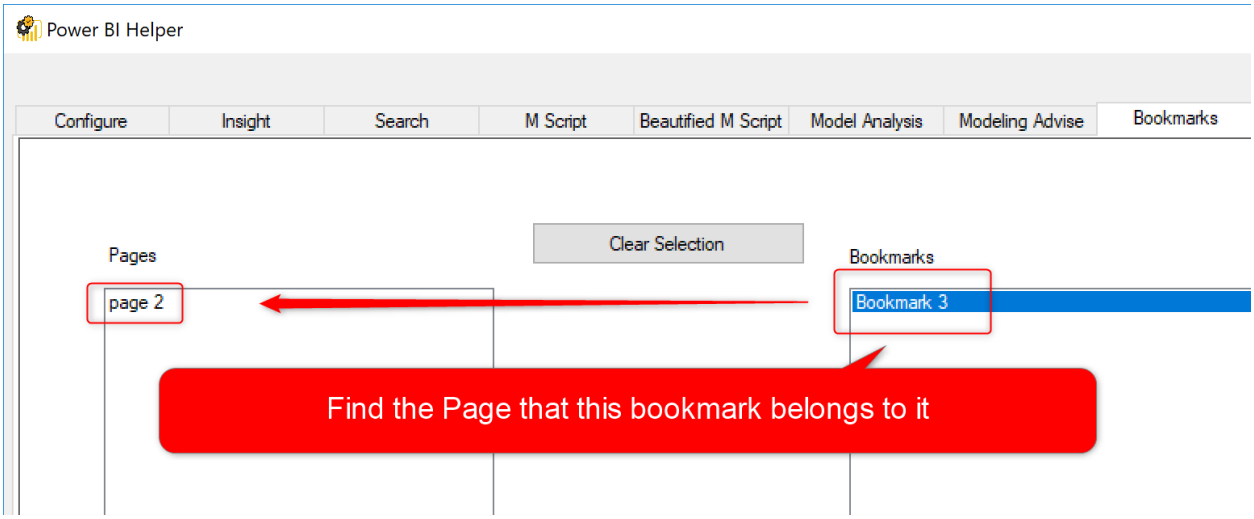


If you select a page, then all bookmarks will be filtered to show only those that are pointing to that page.

Find Bookmarks that belong to the selected page



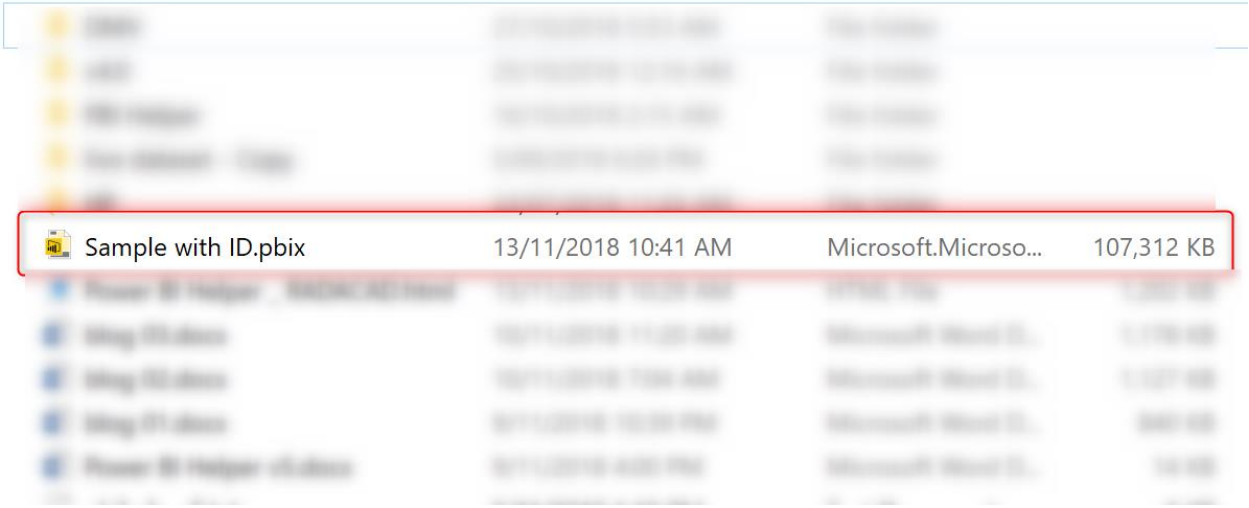
Or if you select a bookmark, you can see which page the bookmark is pointing to



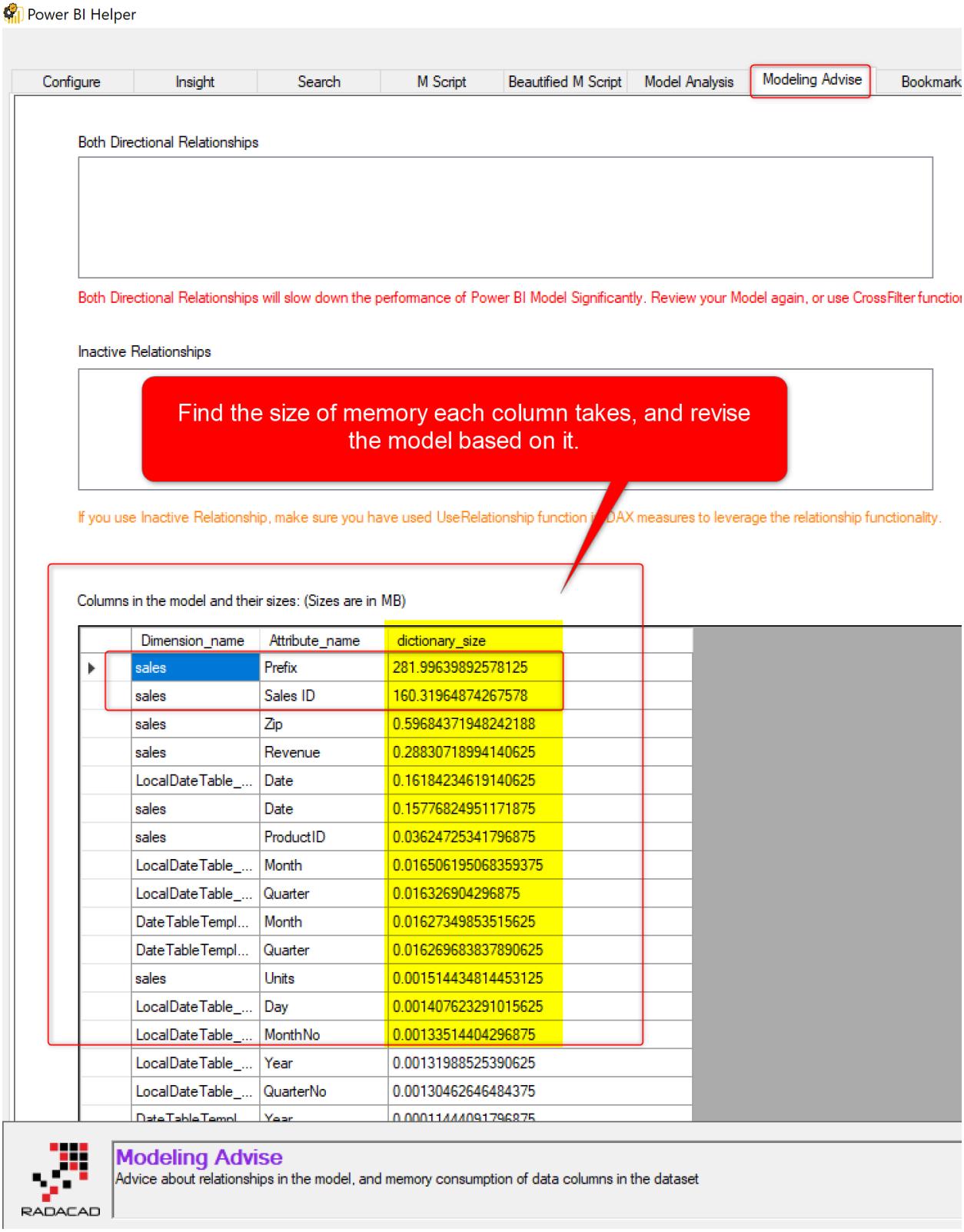
Column Sizes: Performance Tuning Guide

Doing the proper modeling is always a challenge in Power BI, especially when you are working on a model for a while. One of the aspects of modeling is to make sure you do not bring very heavy memory-intensive columns. We used DMVs to find out the amount of memory that each column occupies in the model.

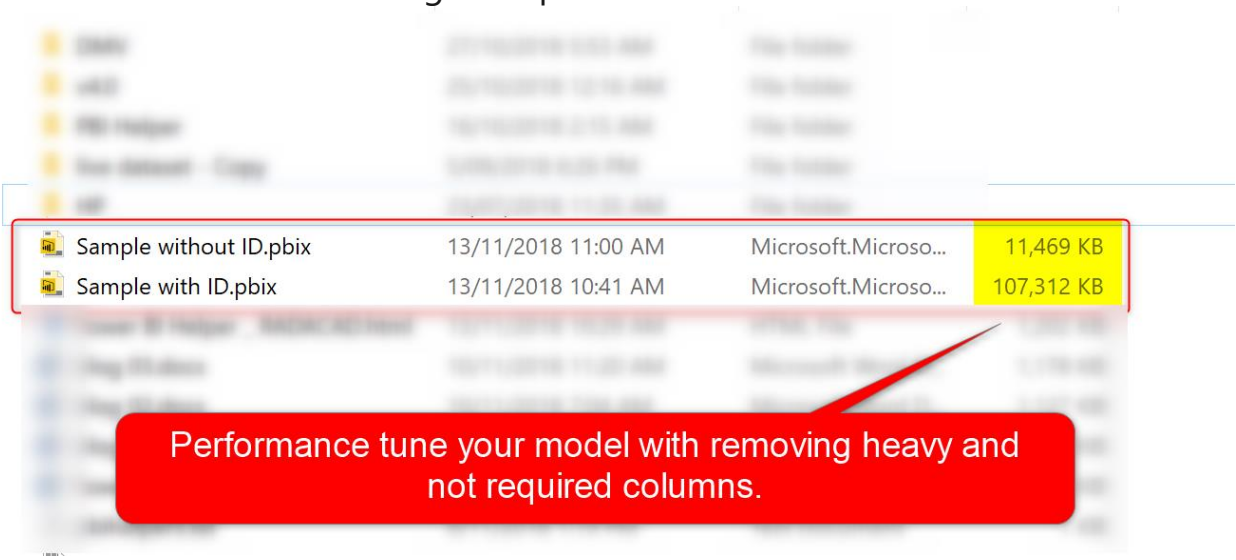
Here is a Power BI model which has the size of …



We can open it through Power BI Helper, and in the Modeling Advice tab, there is a new section that tells us how much space each column consumes:


Power BI Helper
Configure
Insight
Search
M Script
Beautified M Script
Model Analysis
Modeling Advise
Bookmark
Both Directional Relationships
Both Directional Relationships will slow down the performance of Power BI Model Significantly. Review your Model again, or use CrossFilter functio
Inactive Relationships
Find the size of memory each column takes, and revise the model based on it.
If you use Inactive Relationship, make sure you have used UseRelationship function in DAX measures to leverage the relationship functionality.
Columns in the model and their sizes: (Sizes are in MB)
Dimension_name
Attribute_name
dictionary_size
sales Prefix 281.99639892578125
sales Sales ID 160.31964874267578
sales Zip 0.59684371948242188
sales Revenue 0.28830718994140625
LocalDateTable_... Date 0.16184234619140625
sales Date 0.15776824951171875
sales ProductID 0.03624725341796875
LocalDateTable_... Month 0.016506195068359375
LocalDateTable_... Quarter 0.016326904296875
DateTableTempl... Month 0.01627349853515625
DateTableTempl... Quarter 0.016269683837890625
sales Units 0.001514434814453125
LocalDateTable_... Day 0.001407623291015625
LocalDateTable_... MonthNo 0.0013514404296875
LocalDateTable_... Year 0.00131988525390625
LocalDateTable_... QuarterNo 0.00130462646484375
RADACAD
Modeling Advise
Advice about relationships in the model, and memory consumption of data columns in the dataset

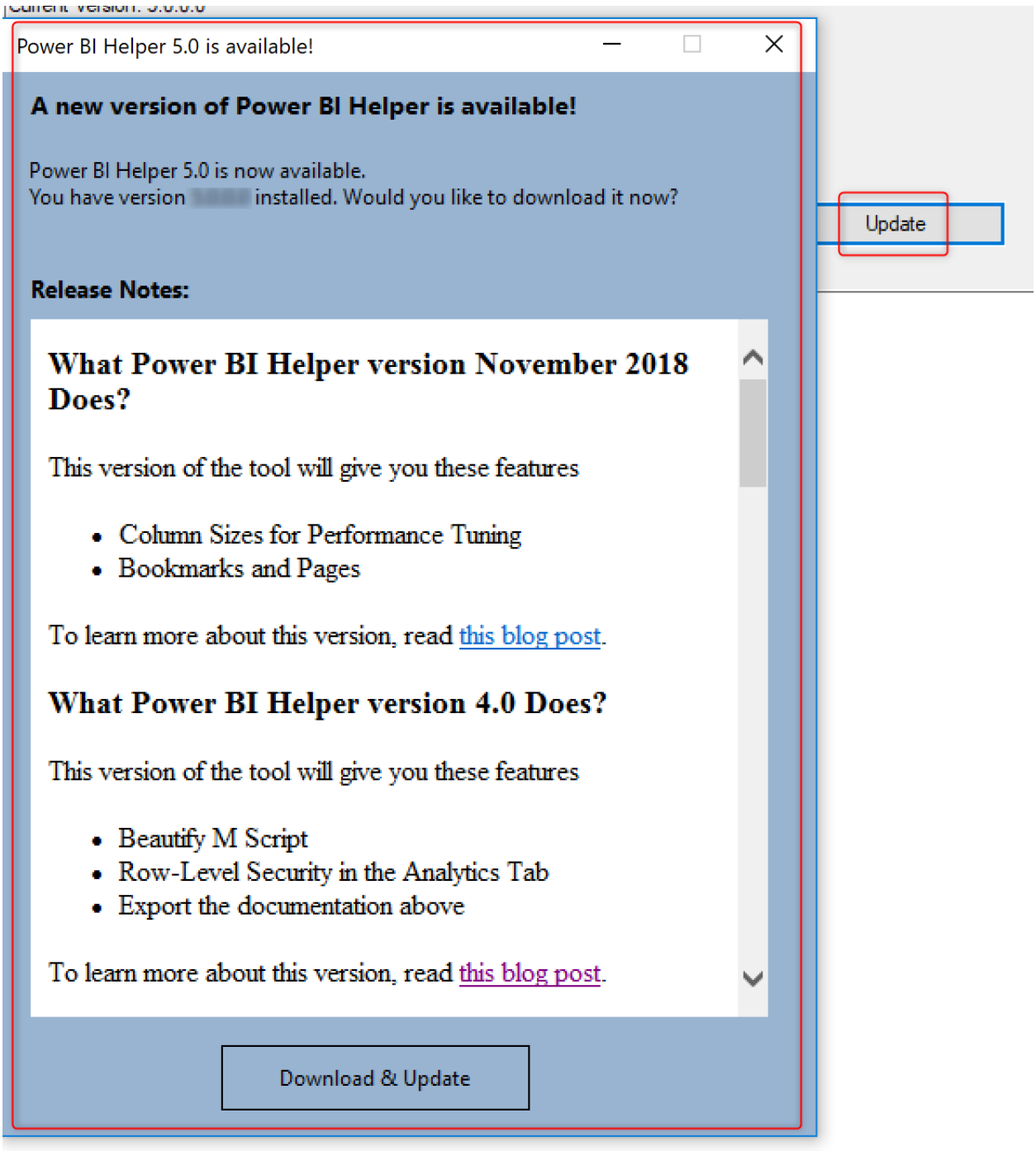
It is clear that the columns Prefix and Sales ID in the above list are the most expensive columns. And obviously, this is because of the cardinality of that. If we remove this column (considering that it is not required in the model), then here is the result after saving the *.pbix file:



Wow! from 107 MB down to 11MB, saving of 90% memory size just with removing heavy, but unnecessary columns.
This very simple feature can give you an idea about expensive columns for the compression engine of Power BI, and then you can control the model based on that.

Update from the Application

Previous versions of Power BI Helper didn't have the "check for updates" feature. And every time you had to keep an eye on the newer available version. We heard you! Now the application will check for updates, and if there are any updates available, it will let you know;



You have the option to decide to update the application or not.

Power BI Helper is a tool for the community, from the community. It is a free tool now, and it will remain as a free tool forever. We want it to help you with all Power BI development needs. So, if you have any ideas worth to add to the product, please don't hesitate to let us know.

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