Created in the cloud with Aspose.Words for Cloud. http://www.aspose.com/products/words/cloud

# Data Dimension in Power Query - Public Holidays Fetched Live

Published Date : February 15, 2014



In[previous post I’ve generated a date dimension with Power Query M script](http://rad.pasfu.com/index.php?/archives/166-Script-to-Generate-Date-Dimension-with-Power-Query-M-With-Financial-Columns.html" \t "_blank). In this post I will add public holidays to the date dimension from live websites. This post only considers New Zealand public holidays, but once you’ve learned how to deal with data, you can apply similar transformations with small changes to any other dataset and fetch public holidays from government or official websites.

**Date dimension** without public holiday columns;

In previous post you’ve seen the script below that generates date dimension (without public holidays);

*let
    // Read Config Table
    ConfigTable=Excel.CurrentWorkbook(){[Name="Table1"]}[Content],
    YearsToAppend=Table.First(ConfigTable)[YearsToGenerate],
    FinancialYearStartingMonth=Table.First(ConfigTable)[FinancialYearStartingMonth],
    // Generate base table
    Source = List.Dates(Date.From(Table.First(ConfigTable)[StartDate]),YearsToAppend\*365,#duration(1, 0, 0, 0)),
    Transformed=List.Transform(Source, each Date.ToRecord(\_)),
    Tabled=Table.FromList(Transformed,Record.FieldValues,{"Year","Month","Day"}),
    //Add Full Date Column

FullDateAddedTable=Table.AddColumn(Tabled,"FullDateAlternateKey",each
Date.FromText(Text.From([Year])&"-"&Text.From([Month])&"-"&Text.From([Day]))),
    DateKeyAdded=Table.AddColumn(FullDateAddedTable,"DateKey",each ([Year]\*10000)+([Month]\*100)+[Day]),

FullDateNameAdded=Table.AddColumn(DateKeyAdded,"DateFullName",each
DateTime.ToText(DateTime.From([FullDateAlternateKey]),"dd MMMM yyyy")),
    // Fiscal Year
    FiscalYearAdded=Table.AddColumn(FullDateNameAdded,
        "Fiscal Year",
         each
              if Date.Month([FullDateAlternateKey])>=FinancialYearStartingMonth then
                    Date.Year([FullDateAlternateKey])+1
              else
                    Date.Year([FullDateAlternateKey])
         ),
    // Fiscal Month
    FiscalQuarterAdded=Table.AddColumn(FiscalYearAdded,
        "Fiscal Quarter",
         each
              if Date.Month([FullDateAlternateKey])>=FinancialYearStartingMonth then
                   Number.IntegerDivide((Date.Month([FullDateAlternateKey])-FinancialYearStartingMonth),3)+1
              else
                   Number.IntegerDivide((12+Date.Month([FullDateAlternateKey])-FinancialYearStartingMonth),3)+1
          ),
    // Calendar Quarter
    CalendarQuarterAdded=Table.AddColumn(FiscalQuarterAdded, "Calendar Quarter",
          each Number.IntegerDivide(Date.Month([FullDateAlternateKey])-1,3)+1
          ),
    // Is Week Day
    WeekDayAdded=Table.AddColumn(CalendarQuarterAdded, "IsWeekDay",
          each
              if
                   Date.DayOfWeek(DateTime.From([FullDateAlternateKey]))=Day.Sunday
                   or
                   Date.DayOfWeek(DateTime.From([FullDateAlternateKey]))=Day.Saturday
             then 0 else 1 ),
    // Day Of Week
    DayOfWeek=Table.AddColumn(WeekDayAdded,"DayOfWeek",each Date.DayOfWeek(DateTime.From([FullDateAlternateKey]))),
    // Month Name
    MonthName=Table.AddColumn(DayOfWeek,"Month Name",each DateTime.ToText(DateTime.From([FullDateAlternateKey]),"MMMM")),
    // Day of Week Name

DayOfWeekName=Table.AddColumn(MonthName,"Day of Week Name",each
DateTime.ToText(DateTime.From([FullDateAlternateKey]),"dddd"))*

*in
    DayOfWeekName*

Here is the screenshot for date dimension:



**Fetch 2014 holidays**

Website below contains list of public holidays in New Zealand for  2014:

[http://www.dol.govt.nz/er/holidaysandleave/publicholidays/publicholidaydates/current.asp](http://www.dol.govt.nz/er/holidaysandleave/publicholidays/publicholidaydates/current.asp%20%22%20%5Ct%20%22_blank)



I’ve written script below to fetch holiday dates from New Zealand government site;

*let
    // Months Table
    MonthList=List.Dates(Date.From("2000-01-01"),12,#duration(31,0,0,0)),
    MonthTable=Table.FromList(MonthList,Splitter.SplitByNothing(),{"Date"}),
    ExtendedMonthTable=Table.AddColumn(MonthTable,"MonthName",
                                   each DateTime.ToText(DateTime.From([Date]),"MMMM")),
    FullMonthTable=Table.AddColumn(ExtendedMonthTable,"MonthNumber",each Date.Month([Date])),
    // Fetch Holiday Dates
    Source = Web.Page(
                      Web.Contents(
              "http://www.dol.govt.nz/er/holidaysandleave/publicholidays/publicholidaydates/current.asp"
                       )),
    Data0 = Source{0}[Data],
    SplitColumnDelimiter = Table.SplitColumn(Data0,"Observed 2014notes",
                    Splitter.SplitTextByDelimiter(" "),
                                   {"Observed 2014notes.1",
                                    "Observed 2014notes.2",
                                    "Observed 2014notes.3"}),
    ChangedType = Table.TransformColumnTypes(
                    SplitColumnDelimiter,
                     {
                        {"", type text},
                        {"Actual Date", type text},
                        {"Observed 2014notes.1", type text},
                        {"Observed 2014notes.2", type number},
                        {"Observed 2014notes.3", type text}}
                         ),
    RemovedColumns = Table.RemoveColumns(ChangedType,{"Observed 2014notes.1"}),
    // Generate full date for holidays
    JoinedTable=Table.AddJoinColumn(RemovedColumns,"Observed 2014notes.3",
                                   FullMonthTable,"MonthName","New Column"),
    MonthNumbered = Table.ExpandTableColumn(
                  JoinedTable, "New Column", {"MonthNumber"}, {"New Column.MonthNumber"}),
    FullDated=Table.AddColumn(MonthNumbered,"FullDate",each
        Date.FromText(Text.From(2014)
             &"-"&Text.From([New Column.MonthNumber])
             &"-"&Text.From([Observed 2014notes.2]))
),
    HolidaySelected=Table.SelectColumns(FullDated,{"FullDate",""}),
    HolidayTable=Table.RenameColumns(HolidaySelected,{"","Description"})
in
    HolidayTable*

Result will be fetched as



**Fetch Holiday dates for 2015 to 2018**

This link contains holiday dates for 2015 to 2018, in a pivoted structure:

[http://www.dol.govt.nz/er/holidaysandleave/publicholidays/publicholidaydates/future-dates.asp](http://www.dol.govt.nz/er/holidaysandleave/publicholidays/publicholidaydates/future-dates.asp%22%20%5Ct%20%22_blank)



Script below is to fetch holiday dates and unpivot them and append years one after each other:

*let
   // Months Table
    MonthList=List.Dates(Date.From("2000-01-01"),12,#duration(31,0,0,0)),
    MonthTable=Table.FromList(MonthList,Splitter.SplitByNothing(),{"Date"}),
    ExtendedMonthTable=Table.AddColumn(MonthTable,"MonthName",
            each DateTime.ToText(DateTime.From([Date]),"MMMM")),
    FullMonthTable=Table.AddColumn(ExtendedMonthTable,"MonthNumber",each Date.Month([Date])),*

*// Cleansing Function
    CleanseColumn = (x) =>
        let
           Result=Text.Trim(Text.Remove(
                    Text.RemoveRange(x,0,
                     if Text.PositionOf(x," or ")>0 then Text.PositionOf(x," or ")+4 else 0
                                    )
                              ,"\*")),
           FirstWordRemoved=Text.RemoveRange(Result,0,Text.PositionOf(Result," ")+1)
        in
           FirstWordRemoved,
    Source = Web.Page(Web.Contents(
        "http://www.dol.govt.nz/er/holidaysandleave/publicholidays/publicholidaydates/future-dates.asp"
                       )),
    Data0 = Source{0}[Data],
    RenamedColumns = Table.RenameColumns(Data0,
             {
                  {"", "Description"},
                  {"Observed Datenotes 2015", "2015"},
                  {"Observed Datenotes 2016", "2016"},
                  {"Observed Datenotes 2017", "2017"},
                  {"Observed Datenotes 2018", "2018"}
             }),
    alternateRemoved=Table.TransformColumns(RenamedColumns,
          {  {"2015", each CleanseColumn(\_)},
            {"2016", each CleanseColumn(\_)},
            {"2017", each CleanseColumn(\_)},
            {"2018", each CleanseColumn(\_)}  }
    ),
    Combined=Table.Combine({
     Table.RenameColumns(
                 Table.AddColumn(
                        Table.SelectColumns(alternateRemoved,{"Description","2015"})
                        ,"Year",each 2015)
                 ,{"2015","MonthDate"}),
     Table.RenameColumns(
                 Table.AddColumn(
                        Table.SelectColumns(alternateRemoved,{"Description","2016"})
                        ,"Year",each 2016)
                 ,{"2016","MonthDate"}),
     Table.RenameColumns(
                 Table.AddColumn(
                        Table.SelectColumns(alternateRemoved,{"Description","2017"})
                        ,"Year",each 2017)
                 ,{"2017","MonthDate"}),
     Table.RenameColumns(
                 Table.AddColumn(
                        Table.SelectColumns(alternateRemoved,{"Description","2018"})
                        ,"Year",each 2018)
                 ,{"2018","MonthDate"})
                           }),
    SplitColumnDelimiter = Table.SplitColumn(Combined,"MonthDate",
               Splitter.SplitTextByDelimiter(" "),{"MonthDate.1", "MonthDate.2"}),
    ChangedType = Table.TransformColumnTypes(SplitColumnDelimiter,
           {{"Description", type text}, {"MonthDate.1", type number}, {"MonthDate.2", type text}}),
    RenamedColumns1 = Table.RenameColumns(ChangedType,
               {{"MonthDate.1", "Day"}, {"MonthDate.2", "Month"}}),*

*Joined=Table.AddJoinColumn(RenamedColumns1 ,{"Month"},
                      FullMonthTable,{"MonthName"},"JoinedColumn"),
    #"Expand JoinedColumn" = Table.ExpandTableColumn(Joined, "JoinedColumn",
                             {"MonthNumber"}, {"JoinedColumn.MonthNumber"}),
    RenamedColumns2 = Table.RenameColumns(#"Expand JoinedColumn",
                           {{"JoinedColumn.MonthNumber", "MonthNumber"}}),
    RemovedColumns = Table.RemoveColumns(RenamedColumns2,{"Month"}),
    RenamedColumns3 = Table.RenameColumns(RemovedColumns,{{"MonthNumber", "Month"}}),
    FullDateAdded=Table.AddColumn(RenamedColumns3 ,"FullDate",
              each Date.FromText(Text.From([Year])&"-"&Text.From([Month])&"-"&Text.From([Day]))),
    RemovedColumns1 = Table.RemoveColumns(FullDateAdded,{"Day", "Year", "Month"}),
    ReorderedColumns = Table.ReorderColumns(RemovedColumns1,{"FullDate", "Description"})
in
    ReorderedColumns*

Here is the result set after applying the script:



**Combine holidays of 2014 with 2015-18**

Because the structure of both holiday tables above are similar, we simply combine them all in a single holiday table with script below:

*let
    Source = Table.Combine({Query2,#"New Zealand public holiday dates 2015-18"})
in
    Source*

This can be done with the GUI of Power Query as well:



**Merge Holidays Table with Date Dimension**

Finally we merge(or join) date dimension with holidays table with below script:

*let
    Source = Table.NestedJoin(Query1,{"FullDateAlternateKey"},Append1,{"FullDate"},"NewColumn"),
    #"Expand NewColumn" = Table.ExpandTableColumn(Source, "NewColumn",
                     {"Description"}, {"NewColumn.Description"}),
    RenamedColumns = Table.RenameColumns(#"Expand NewColumn",{{"NewColumn.Description",
                     "HolidayDescription"}}),
    HolidayFlagAdded=Table.AddColumn(RenamedColumns,"IsPublicHoliday",
             each if [HolidayDescription] is null then 0 else 1),
    SortedRows = Table.Sort(HolidayFlagAdded,{{"FullDateAlternateKey", Order.Ascending}})
in
    SortedRows*

Here is the output:



Merge also can be applied through the GUI options:



You can download the excel spreadsheet with all queries from here:

[http://rad.pasfu.com/ssis/mdatedimholidays/DateDimensionWithPublicHolidays.zip](http://rad.pasfu.com/ssis/mdatedimholidays/DateDimensionWithPublicHolidays.zip%22%20%5Ct%20%22_blank)

# Script to Generate Date Dimension with Power Query M - With Financial Columns

Published Date : February 14, 2014



There are many date dimensions scripts on the internet, and even there are some scripts for the Power Query (Thanks to [Chris Webb for his very early version of the date dimension](http://cwebbbi.wordpress.com/2013/11/19/generating-a-date-dimension-table-in-power-query/%22%20%5Ct%20%22_blank), and also to [Matt Masson for his version of Date Dimension](http://www.mattmasson.com/2014/02/creating-a-date-dimension-with-a-power-query-script/%22%20%5Ct%20%22_blank)). I’ve built a date dimension from scratch (not a copy or extended version), because one of my reasons to build this dimension was Practice! So I did that to practice Power Query more in action. There are some other reasons that why I built this dimension.

Why I built the Date dimension;

* To use it: Date Dimension is one of the most common dimensions in Data Model. It is too rare to you see a data model without date dimension.
* I wanted to build a Power Query copy of my date dimension ([previously I’ve done a date dimension with T-SQL](/index.php?/archives/156-Script-to-Generate-and-Populate-Date-Dimension-Version-2-Adding-Multiple-Financial-Years.html" \t "_blank))
* I used this as a practice to learn Power Query more in action.
* I need a demo for my upcoming speaking and presentations on Power Query.
* I like to share something useful in community.

The Date dimension that I’ve built contains:
-Financial calendar (configured in the config table)
-Weekend and Weekdays

\*Note: This version of date dimension doesn’t contains public holidays, I’ll work on the newer version with public holidays and write another blog post about it.

**Here is the script to build the Date Dimension**:

*let
// Read Config Table
ConfigTable=Excel.CurrentWorkbook(){[Name=”Table1″]}[Content],
YearsToAppend=Table.First(ConfigTable)[YearsToGenerate],
FinancialYearStartingMonth=Table.First(ConfigTable)[FinancialYearStartingMonth],
// Generate base table
Source = List.Dates(Date.From(Table.First(ConfigTable)[StartDate]),YearsToAppend\*365,#duration(1, 0, 0, 0)),
Transformed=List.Transform(Source, each Date.ToRecord(\_)),
Tabled=Table.FromList(Transformed,Record.FieldValues,{“Year”,”Month”,”Day”}),
//Add Full Date Column
FullDateAddedTable=Table.AddColumn(Tabled,”FullDateAlternateKey”,each Date.FromText(Text.From([Year])&”-“&Text.From([Month])&”-“&Text.From([Day]))),
DateKeyAdded=Table.AddColumn(FullDateAddedTable,”DateKey”,each ([Year]\*10000)+([Month]\*100)+[Day]),
FullDateNameAdded=Table.AddColumn(DateKeyAdded,”DateFullName”,each DateTime.ToText(DateTime.From([FullDateAlternateKey]),”dd MMMM yyyy”)),
// Fiscal Year
FiscalYearAdded=Table.AddColumn(FullDateNameAdded,
“Fiscal Year”,
each
if Date.Month([FullDateAlternateKey])>=FinancialYearStartingMonth then
Date.Year([FullDateAlternateKey])+1
else
Date.Year([FullDateAlternateKey])
),
// Fiscal Month
FiscalQuarterAdded=Table.AddColumn(FiscalYearAdded,
“Fiscal Quarter”,
each
if Date.Month([FullDateAlternateKey])>=FinancialYearStartingMonth then
Number.IntegerDivide((Date.Month([FullDateAlternateKey])-FinancialYearStartingMonth),3)+1
else
Number.IntegerDivide((12+Date.Month([FullDateAlternateKey])-FinancialYearStartingMonth),3)+1
),
// Calendar Quarter
CalendarQuarterAdded=Table.AddColumn(FiscalQuarterAdded, “Calendar Quarter”,
each Number.IntegerDivide(Date.Month([FullDateAlternateKey])-1,3)+1
),
// Is Week Day
WeekDayAdded=Table.AddColumn(CalendarQuarterAdded, “IsWeekDay”,
each
if
Date.DayOfWeek(DateTime.From([FullDateAlternateKey]))=Day.Sunday
or
Date.DayOfWeek(DateTime.From([FullDateAlternateKey]))=Day.Saturday
then 0 else 1 ),
// Day Of Week
DayOfWeek=Table.AddColumn(WeekDayAdded,”DayOfWeek”,each Date.DayOfWeek(DateTime.From([FullDateAlternateKey]))),
// Month Name
MonthName=Table.AddColumn(DayOfWeek,”Month Name”,each DateTime.ToText(DateTime.From([FullDateAlternateKey]),”MMMM”)),
// Day of Week Name
DayOfWeekName=Table.AddColumn(MonthName,”Day of Week Name”,each DateTime.ToText(DateTime.From([FullDateAlternateKey]),”dddd”))*

in
DayOfWeekName

You can download the Sample workbook with config table and the script from here:

[date-dimension-with-public-holidays-start](http://radacad.com/wp-content/uploads/2014/02/Date-Dimension-With-Public-Holidays-Start.xlsx)

**What I’ve learned through this example:**

**Loop Structure**As you probably noticed, Power Query (up to current version) doesn’t contain a loop structure.
This is a weakness for this version, but there are methods to do what you want with other structures. List and Table are structures that contains item and records. Fortunately List can be generated with some functions that called as **GENERATORS**. If you notice to the line of text with List.Dates, you would see that this line of code generates a list of dates from the start date with the duration specified up to specific number.
This line of code :

*Source = List.Dates(Date.From(Table.First(ConfigTable)[StartDate]),YearsToAppend\*365,#duration(1, 0, 0, 0)),*

Above line generates the list of dates, and below lines transform that date to a record containing day,month, and year, and finally it would be converted to a table with Table.FromList function:
*Transformed=List.Transform(Source, each Date.ToRecord(\_)),
Tabled=Table.FromList(Transformed,Record.FieldValues,{“Year”,”Month”,”Day”}),*

So as you seen above we generated a list from scratch, and then converted that to a table. you can apply expressions to a new column of the table with EACH single parameter function, and that means you can simulate loop like structure with List/Table structures. I would dedicate a blog post about this topic later with more illustration and detailed samples.

**Date Functions**

There are bunch of date functions used in this sample, such as :

Date.Month : returns the month of specific date

Date.DayOfWeek: returns enumeration values of the weekday, zero would be Sunday, and 6 would be Saturday

DateTime.ToText: generates a text FORMATTED from the date time with specified format.

There are many date functions which I will write about them in another blog post in future.

**Reading information from Excel Workbook**



In this script I used a config table, that table sits in an excel workbook, and I used lines below to read that table from the Excel workbook:

*// Read Config Table
ConfigTable=Excel.CurrentWorkbook(){[Name=”Table1″]}[Content],
YearsToAppend=Table.First(ConfigTable)[YearsToGenerate],
FinancialYearStartingMonth=Table.First(ConfigTable)[FinancialYearStartingMonth],*

Having a config table such as the one above, helps to re-generate data with just a REFRESH button instead of changing hard-coded values in the script.

**Save**