SumProduct

NEWSLETTER #69 - August 2018
www.sumproduct.com | www.sumproduct.com/thought

You can count on us this month

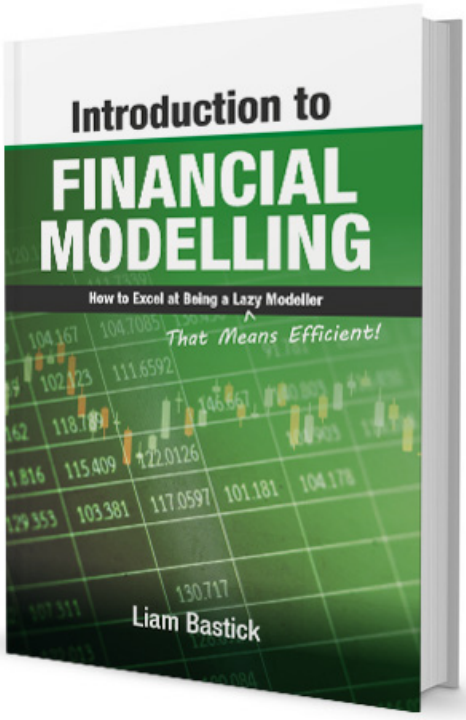
as we field the usual blarney of updates (including a rare appearance from Power Query / Get & Transform), provide tips, Power Query Pointers and keyboard shortcuts. Yup, we have the **COUNT** family of functions to get our teeth into, but **COUNTDRACULA** does appear to be conspicuous by its absence.

Fellow SumProduct colleague and Excel MVP Tim Heng went over to Seattle to soak up the sun (drizzle?) and the Microsoft Business Applications Summit – he reports back *(below)*. We also reproduce one of our most popular articles from our website for some time – how to revise forecasts in Excel *automatically*.

I don't expect anyone to read this cover to cover, but hopefully there's something for everyone in this month's bumper edition. Enjoy!

Liam Bastick, Managing Director, SumProduct

Financial Modelling Book Out Now!!



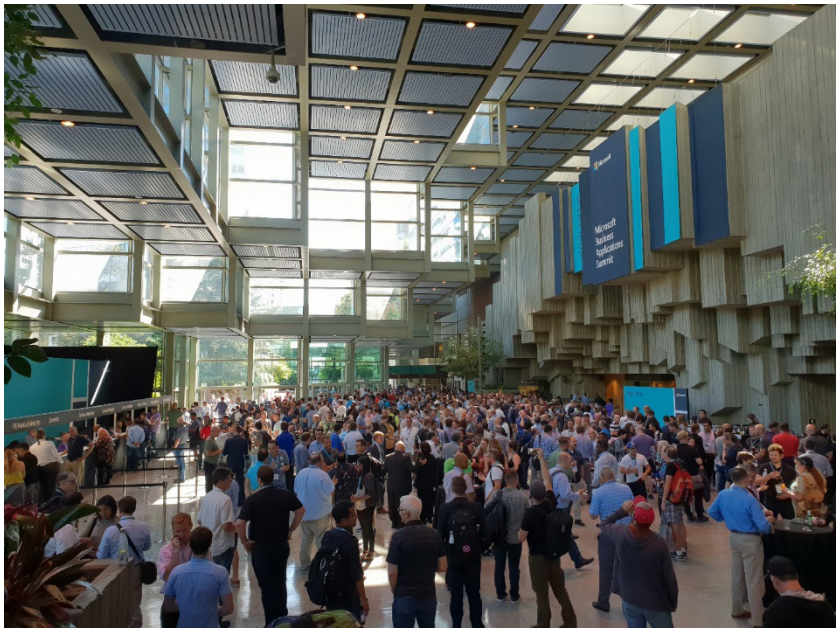
It's taken a while (due to error removals), but SumProduct is [finally!] proud to announce that ***Introduction to Financial Modelling by Liam Bastick (edited by Tim Heng)*** is out now.

It's been a labour of something (love they say, but we are not convinced...) and contains 325 pages explaining the financial modelling basics so that you can get started building models in a professional manner without falling for the usual traps, trials and tribulations.

If you would like your very own copy, email contact@sumproduct.com with the subject "FINANCIAL MODELLING BOOK" and we will be in contact to coordinate payment and delivery. For those that have already expressed an interest, we shall be in contact very soon. Prices are from US$50, A$65, €40 and UK£35, plus postage and packaging. Payment accepted via PayPal only. Discounts are available for bulk orders of 10 or more *(Ed: hope springs eternal!)*.

Microsoft Business Applications Summit – July 2018, Seattle

In case you hadn't heard, we sent our Sydney Director, Tim Heng, off to Seattle to cover the Microsoft Business Applications Summit and find out all the latest gossip in the world of Power BI. Except that this year, Microsoft are really emphasising the use of Dynamics 365 and the "Power platform", being a combination of Power BI, PowerApps and Microsoft Flow.



While Power BI has been around for a few years now, it feels like Microsoft are starting to push the usage of PowerApps and Microsoft Flow, as it introduces the ability to embed Flow and PowerApps into Excel workflows and Power BI dashboards. For those who aren't aware, Flow allows you to implement automated processes, such as scheduling an update or a process to be run every hour or saving incoming email attachments automatically to OneDrive. PowerApps gives you a point-and-click interface to create applications that you can use to draw on and update datasets. Both of these products were first seen at the Data Insights Summit last year and both have been heavily improved upon in the relatively short time since.

For our die-hard Excel fans *(Ed: our office and presumably Bruce Willis included!)*, there were reminders about the different tools that are either on the 'Insider Fast' track at the moment or coming soon to Excel, including Insights and Custom Visuals (both should be familiar to those of you using Power BI), as well as reiterating the introduction of Rich Data Types, Custom functions using JavaScript, and the Azure Machine Learning add-in.

Over the next month, we'll be posting regular blogs going into some of the new Power BI features in detail, replacing the regular Power BI Thursday blog series temporarily. Most of the Power BI tools are going to take a few months to roll out, with the majority of the updates being promised by October 2018 to line up with the six-monthly Dynamics 365 upgrade cycle.

You can check out our Power BI blogs on Thursdays at www.sumproduct.com/blog.

Revising Forecasts

This was a Final Friday Fix several months ago – and the reaction went viral! We thought it might be worth adding to the newsletter given its popularity – just in case you missed it…

Imagine you had just finalised the budget for a project and *(say)* it started in Period 3 and ended in Period 8 as pictured:

Original Forecast

Assumptions

		Total	1	2	3	4	5	6	7	8	
Period	#										
Forecast	$	45			5	6	7	8	9	10	

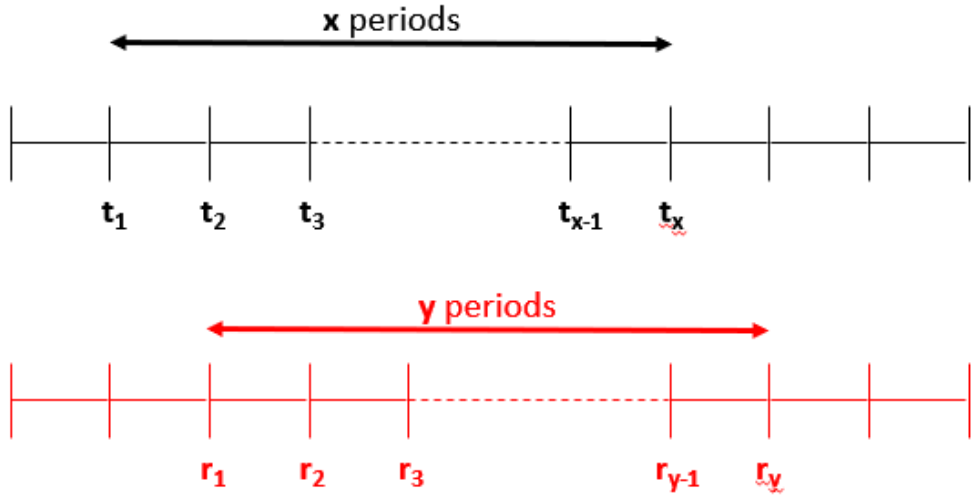
Suddenly, your boss told you the amounts needed to be reallocated on a "similar basis" but for Periods 4 to 15. That's fairly straightforward, as this duration is double the original project length, so you would just attribute half of each period's amount to the new periods, *viz*.

Total	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
45	-	-	-	2.50	2.50	3.00	3.00	3.50	3.50	4.00	4.00	4.50	4.50	5.00	5.00

But what about a general solution? How would you cope with project advancements or delays and changes in duration at the same time? It sounds pretty horrible, but truth be told, finance staff face these types of challenge day-in, day-out.

Assuming no inflationary factors to consider (*e.g.* time value of money), the problem boils down to pro-rating the original numbers across the new number of periods. The revised start and end dates tell you when the calculations begin, but in essence it is the number of periods in the revised forecast that drives the calculations.

Sorry for the algebra, but sometimes that's what's needed in a financial model! Let's assume our original forecast has **x** periods going from start period $\mathbf{t}_1$ to end period $\mathbf{t}_x$, and the revised forecast has **y** periods going from revised start period $\mathbf{r}_1$ to revised end period $\mathbf{r}_y$.



In this illustration, $\mathbf{r}_1$ occurs after $\mathbf{t}_1$, but this does not have to be true necessarily.

Regardless of start and finish dates (which simply governs when the calculations are made), there are basically three scenarios:

1. **x > y**, *i.e.* the revised forecast duration is shorter than the original one
2. **x < y**, *i.e.* the revised forecast duration is longer than the original one
3. **x = y**, *i.e.* the durations of both forecast periods are equal (this effectively simply moves the forecast period).

Let's focus on the first scenario for a moment as it brings into focus how we could go about calculating the revised forecast. If the original duration were longer, then the revised forecast will consider the effects of more than one original period in each period, *e.g.*

In this graphic, the red boxes / yellow shading represent original periods and the blue boxes / borders denote a revised period. If **x > y**, then the blue box must straddle at least two red boxes. It could be more though, which is what is depicted here, where we have:

- a **start period**, where this is the proportion of the earliest original period considered
- **middle [or full] period(s)**, which (when **x > y**) are original periods that must be fully included. There could be more than one. If **x < y**, middle (full) period is not defined
- an **end period**, which is the proportion of the final original period considered.

Sounds confusing? Let's explain with an example:

1. Original Forecast

Original Forecast

Assumptions

		Total	1	2	3	4	5	6	7	8
Period	#									
Forecast	$	36	1	2	3	4	5	6	7	8

		Period
Start Period	#	1
End Period	#	8

{=MIN(IF(LU_Original_Forecast_Data<>0,LU_Periods))}
{=MAX(IF(LU_Original_Forecast_Data<>0,LU_Periods))}

2. Revised Forecast

Revised Forecast

Assumptions

		Period	Check
Start Period	#	4	
End Period	#	6	☑

Revised Forecast

		Total	1	2	3	4	5	6	7	8
Period	#									
Forecast	$	36				5	12	19		

In the original forecasts, the cashflows of $1 to $8 (big spenders here!) were allocated across the first eight periods for a total of a rather exorbitant $36. However, the revised forecast wanted the same profile over just periods 4 to 6 (three periods). That is, the start date $\mathbf{t}_1$ is period 1, **x** is 8 and the final period $\mathbf{t}_x$ ($\mathbf{t}_8$) is period 8.

The start and end dates ($\mathbf{r}_1$ and $\mathbf{r}_3$, periods 4 and 6 respectively) for the revised forecast just denote when the forecast starts and stops. The key information is that there are only 3 (**y**) periods. This means that each period in the revised forecast includes 8/3 (known as the **Period Factor** in the attached Excel file) which equals two and two-thirds (2.67) periods of the old forecast data *viz*.

- Revised Period 4 = Old Period 1 + Old Period 2 + 2/3 of Old Period 3 = 1 + 2 + (2x3)/3 = 5
- Revised Period 5 = 1/3 of Old Period 3 + Old Period 4 + Old Period 5 + 1/3 of Old Period 6 = (1x3)/3 + 4 + 5 + (1x6)/3 = 12
- Revised Period 6 = 2/3 of Old Period 6 + Old Period 7 + Old Period 8 = (2x6)/3 + 7 + 8 = 19.

Our attached Excel file identifies which original periods are used in each revised period,

Forecast Allocation

Period	#	1	2	3	4	5	6	7	8
Revised Flag	[1,0]								
Start	#				-	2.67	5.33		
End	#				2.67	5.33	8.00		

what the start, middle / full and end periods are,

Forecast Allocation

Period	#	1	2	3	4	5	6	7	8
Start Part Period	#	-	-	-	-	3.00	6.00	-	-
Start Full Period	#	-	-	-	-	4.00	7.00	-	-
End Full Period	#	-	-	-	2.00	5.00	8.00	-	-
End Part Period	#	-	-	-	3.00	6.00	-	-	-

And what proportions to use of each:

Forecast Allocation

Period	#	1	2	3	4	5	6	7	8
Start Part %	%	-	-	-	-	33%	67%	-	-
Full Part %	%	-	-	-	-	200%	200%	-	-
End Part %	%	-	-	-	67%	33%	-	-	-

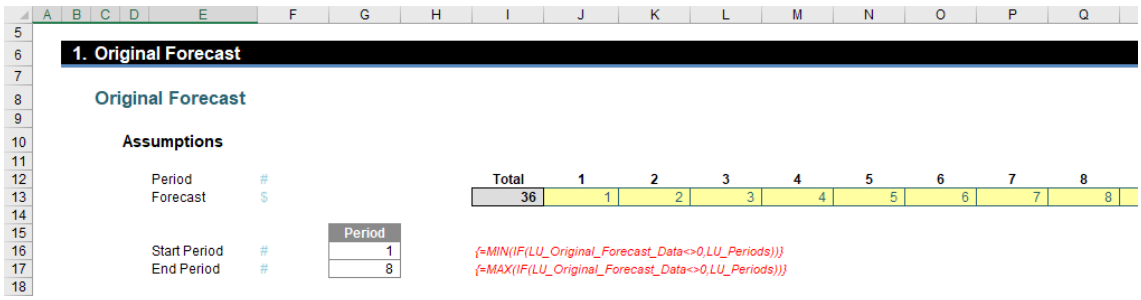
These then cross-multiply the original forecast numbers for the appropriate periods using the **SUMIF** and **SUMIFS** functions to get the values explained above.

When the revised forecast period is longer than the original one, the problem is slightly simpler as there are no middle / full periods (*i.e.* no period of original data is ever in just one revised period). Otherwise, the logic remains the same.

For those who are interested or are insomniacs, the detail is discussed below…

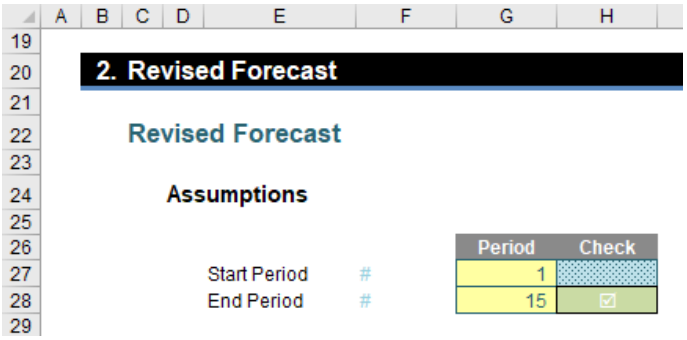
Devil's in the Detail

Here's the detail.



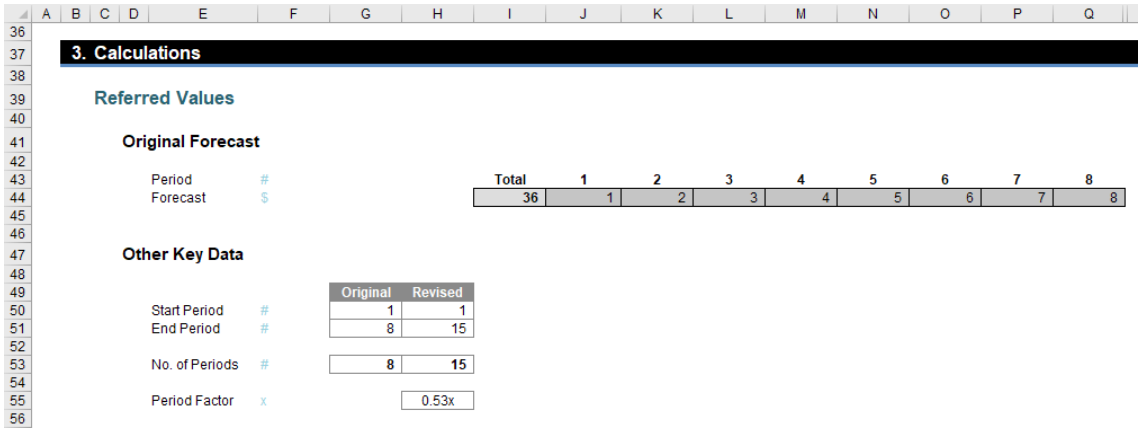
The first section captures the original forecast (inputs) in cells **J13:Q13** and automatically computes the start and end period using the array formulae for **MIN(IF)** and **MAX(IF)** (cells **G16** and **G17** respectively). These must be entered using **CTRL + SHIFT + ENTER** as **IF** will not work across a range (an array) of cells otherwise.

The next section is the Revised Forecast assumptions:



This collects the required start and end periods in cells **G27** and **G28**, together with an error check in cell **H28** to ensure that the end period is not before the start period.

The first part of the next section simply collates all of the date to be used:



3. Calculations

Referred Values

Original Forecast

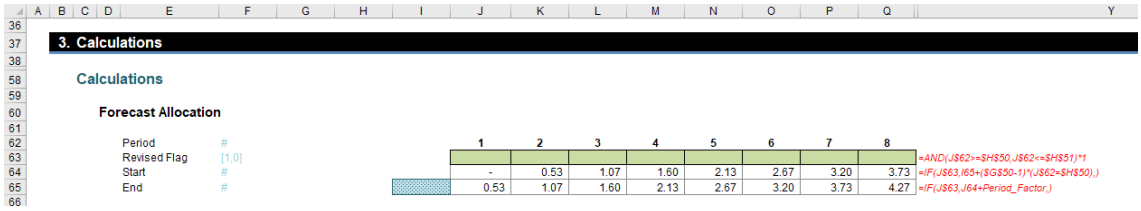
		Total	1	2	3	4	5	6	7	8
Period	#									
Forecast	$	36	1	2	3	4	5	6	7	8

Other Key Data

		Original	Revised
Start Period	#	1	1
End Period	#	8	15
No. of Periods	#	8	15
Period Factor	x		0.53x

The key calculation here is the Period Factor (cell **H55**) which divides the original forecast duration by the revised forecast duration. This represents the number of original periods in each revised period and this is pivotal to all of the calculations.

The next part of this section works out how the original periods are reallocated to the revised periods:



3. Calculations

Calculations

Forecast Allocation

		1	2	3	4	5	6	7	8	
Period	#									
Revised Flag	[1,0]									=AND(J$62>=$H$50,J$62<=$H$51)*1
Start	#	-	0.53	1.07	1.60	2.13	2.67	3.20	3.73	=IF(J$63,I65+($G$50-1)*(J$62=$H$50),)
End	#	0.53	1.07	1.60	2.13	2.67	3.20	3.73	4.27	=IF(J$63,J64+Period_Factor,)

The Revised Flag (row 63) use the formula

=AND(J$62>=$H$50,J$62<=$H$51)*1

to check that the period counters in row 62 are greater than or equal to the revised start period (**H50**) and less than or equal to the revised end period (**H51**). The value is 1 if these assumptions are true and zero (0) otherwise.

The formula for the Start (row 64),

=IF(J$63,I65+($G$50-1)*(J$62=$H$50),)

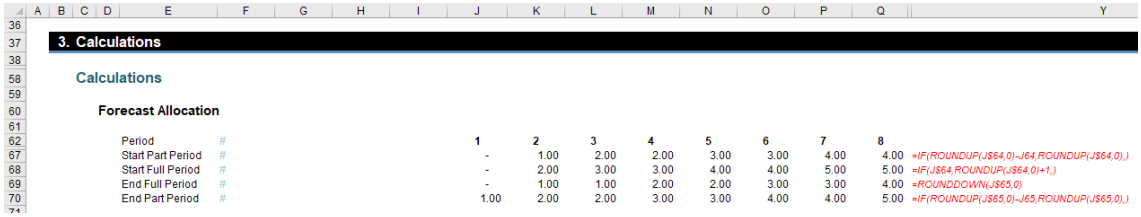
is a simple formula that takes the previous period's closing balance (as long as the flag is active) but also accounts for the fact that the original forecast may not have occurred in Period 1 (*i.e.* it sets the first period $\mathbf{t}_1$ of the original forecast period).

The final formula for the End (row 65),

=IF(J$63,J64+Period_Factor,)

simply adds the Period Factor to the Start period as long as the flag is active. This gives us ultimately the beginning and the end of the blue section in our graphic from before:

The next section starts working out which original periods need to be considered for the start, middle (full) and end:



3. Calculations

Calculations

Forecast Allocation

		1	2	3	4	5	6	7	8	
Period	#									
Start Part Period	#	-	1.00	2.00	2.00	3.00	3.00	4.00	4.00	=IF(ROUNDUP(J$64,0)-J64,ROUNDUP(J$64,0),)
Start Full Period	#	-	2.00	3.00	3.00	4.00	4.00	5.00	5.00	=IF(J$64,ROUNDUP(J$64,0)+1,)
End Full Period	#	-	1.00	1.00	2.00	2.00	3.00	3.00	4.00	=ROUNDDOWN(J$65,0)
End Part Period	#	1.00	2.00	2.00	3.00	3.00	4.00	4.00	5.00	=IF(ROUNDUP(J$65,0)-J65,ROUNDUP(J$65,0),)

The Start Part Period uses the formula

=IF(ROUNDUP(J$64,0)-J64,ROUNDUP(J$64,0),)

Essentially, if Start (row 64) is an integer it uses that period number otherwise it uses the next period (**ROUNDUP(z,0)** rounds **z** up to zero decimal places, *i.e.* the next whole number).

Rows 68 and 69 establish the beginning and the end of the middle (full) period – sort of. Row 69, the calculation for the Start Part Period,

=IF(J$64,ROUNDUP(J$64,0)+1,)

adds one to the Start Part Period (row 67) (as long this is not zero) to avoid any double count. Row 69's formula for the End Full Period,

=ROUNDDOWN(J$65,0)

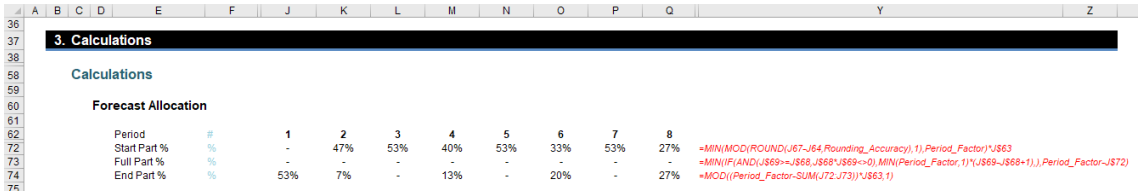
takes the "beginning of the end", that is, up to but not including the End period. Therefore, the way these two dates are calculated it is possible that the Start Full Period could be a period prior to the End Full Period. That is actually pictured in our example *(above)* and is acceptable – it simply means there is no full / middle period in that instance.

The final formula here (row 70) for End Part Period,

=IF(ROUNDUP(J$65,0)-J65,ROUNDUP(J$65,0),)

uses the same logic as per the Start Part Period. This means we now have the relevant original periods identified!

Next, we need to know what percentages should be used for Start and End Part Periods.



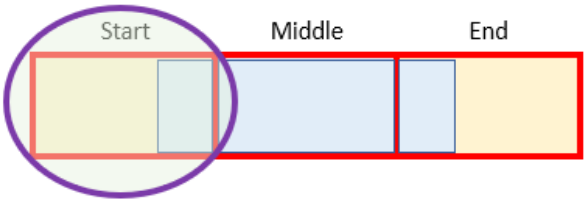
37	**3. Calculations**										
58	Calculations										
60	**Forecast Allocation**										
62	Period	#	1	2	3	4	5	6	7	8	
72	Start Part %	%	-	47%	53%	40%	53%	33%	53%	27%	=MIN(MOD(ROUND(J67-J64,Rounding_Accuracy),1),Period_Factor)*J$63
73	Full Part %	%	-	-	-	-	-	-	-	-	=MIN(IF(AND(J$69>=J$68,J$68*J$69<>0),MIN(Period_Factor,1)*(J$69-J$68+1),),Period_Factor-J$72)
74	End Part %	%	53%	7%	-	13%	-	20%	-	27%	=MOD((Period_Factor-SUM(J72:J73))*J$63,1)

The Full Part % is also calculated as it ensures the End Part % is not overstated.

The formula in row 72 for the Start Part %,

=MIN(MOD(ROUND(J67-J64,Rounding_Accuracy),1),Period_Factor)*J$63

looks horrible but isn't as bad as it seems (honest)! **J67-J64** calculates the proportion Start Part Period less Start (*i.e.* this formula computes the proportion of the first red box that is blue).



ROUND is used to prevent rounding errors and **MOD** is incorporated to ensure this proportion is less than 100% (I've discussed **MOD** in a previous article or two).

The second formula is not pleasant either. The Full Part % (row 73) is given by

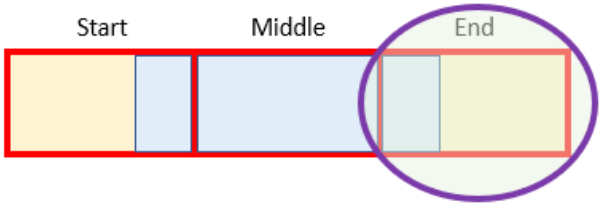
=MIN(IF(AND(J$69>=J$68,J$68*J$69<>0),MIN(Period_Factor,1)*(J$69-J$68+1),),Period_Factor-J$72)

Erm, lovely… Again, once you get your head wrapped around it, it's not so bad. The two **IF** conditions required (inside the **AND** expression) check that the periods are not zero and that the end is not before the beginning (as discussed above). If this test is passed, it takes the **MIN(Period_Factor,1)** (you cannot count more than the forecast amount in an original period) and multiplies this by the number of full original periods in the revised period. This is then restricted so that the sum of the Start Part % and the Full Part % cannot exceed the Period Factor. This number is calculated only to keep the End Part % honest. Talking of which…

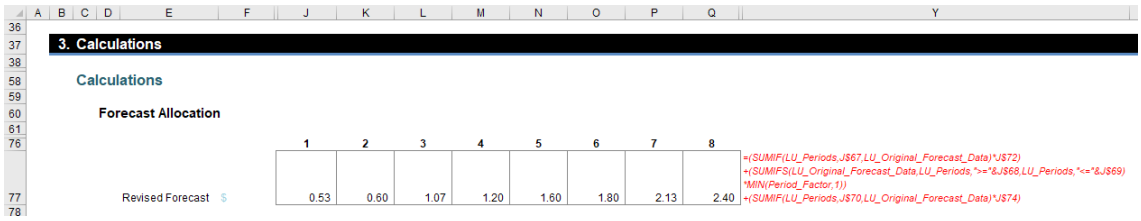
The End Part % (row 74),

=MOD((Period_Factor-SUM(J72:J73))*J$63,1)

just mops up the rest of the Period Factor where the flag is active. This is equal to the section highlighted:



This concludes the percentages needed. We now have identified which periods are the Start Middle and End and what proportions we require for the Start and End. "All" we have to do is multiply it out:



I say "all" because we've left the best to last…

=(SUMIF(LU_Periods,J$67,LU_Original_Forecast_Data)*J$72)
+(SUMIFS(LU_Original_Forecast_Data,LU_Periods,">="&J$68,LU_Periods,"<="&J$69)*MIN(Period_Factor,1))
+(SUMIF(LU_Periods,J$70,LU_Original_Forecast_Data)*J$74)

Again, it's really not that bad! There's three calculations here = one each for the start, middle and end. The first one

=SUMIF(LU_Periods,J$67,LU_Original_Forecast_Data)*J$72

locates the original period to be used for Start and multiplies it by the appropriate proportion (**SUMIF** only sums the range **LU_Original_Forecast_Data** where the counter in LU_Periods is equal to the value in cell **J67**, *i.e.* the correct original period to be used).

The last formula,

=SUMIF(LU_Periods,J$70,LU_Original_Forecast_Data)*J$74

performs a similar operation for the End period. This just leaves

=SUMIFS(LU_Original_Forecast_Data,LU_Periods,">="&J$68,LU_Periods,"<="&J$69)*MIN(Period_Factor,1)

SUMIFS is used here as we need to sum based on two conditions, not one (that the full periods meet the conditions for the Start and End Full Periods). Here, you can clearly see if the End Full Period precedes the Start Full Period, no amounts will be summed. The factor **MIN(Period_Factor,1)** is required when the number of revised periods is greater than the original number of forecast periods (so only the correct proportion) is used and to ensure the amount in a full period is never multiplied by a factor greater than 1 also.

These three added together give us our total:

Word to the Wise

Anyway, apologies for this month's article being a little heavier than usual, but this is a common problem when revising forecasts in Excel. The solution may be a little involved, but I hope you will agree you can always "steal" my template and figure out the formulae at your leisure. This is the curse of modelling sometimes – not always is every essential calculation simple! You can find an example file to download at www.sumproduct.com/blog/article/monday-morning-mulling-april-2018-challenge.

London Excel Developer Conference

Thursday 18 October sees the inaugural Excel extensibility conference in London at the Microsoft Reactor (70 Wilson Street). Many key figures associated with the main Excel extensibility tools and frameworks are going to be there speaking, and more importantly, mingling.

Topics will include the how's and why's of ExcelDNA, PyXLL, XLL + and the new Excel JavaScript APIs. There will also be sessions covering Power Query, VSTO and Add-in Express.

This is a community driven event (so the price at GBP 120 is quite reasonable), driven by a desire to get the top minds in the Excel extensibility space together and working together. Excel MVPs Charles Williams and Gasper Kamensek will be presenting amongst others.

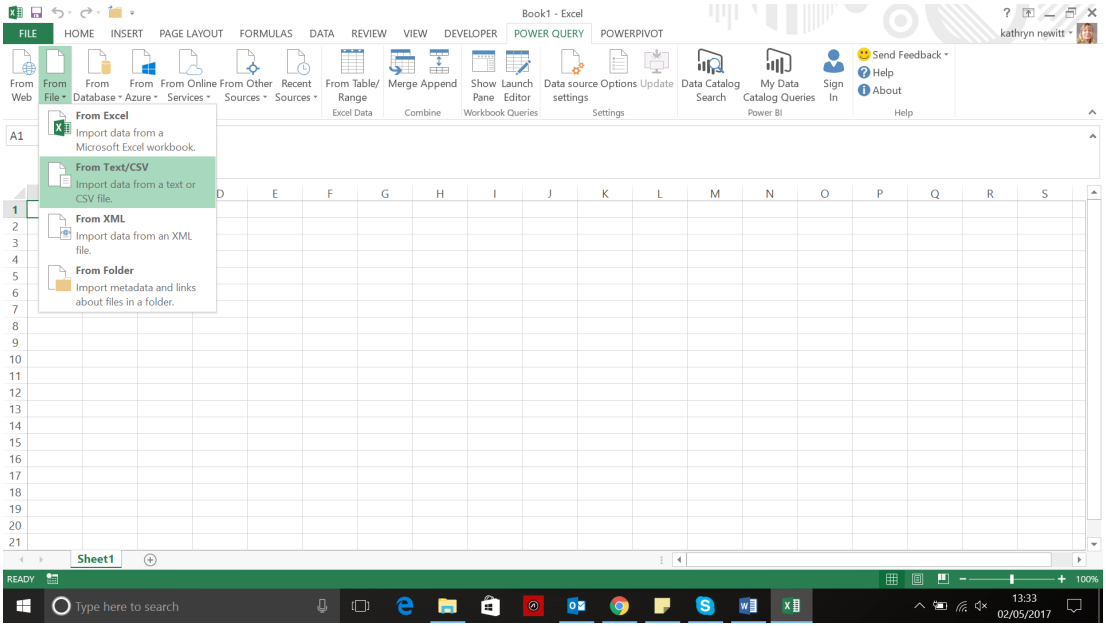
To find out more, check out developexcel.wordpress.com/agenda/ for the agenda and www.eventbrite.co.uk/e/develop-excel-tickets-48199788866 for pricing information and tickets.

Power Query Pointers

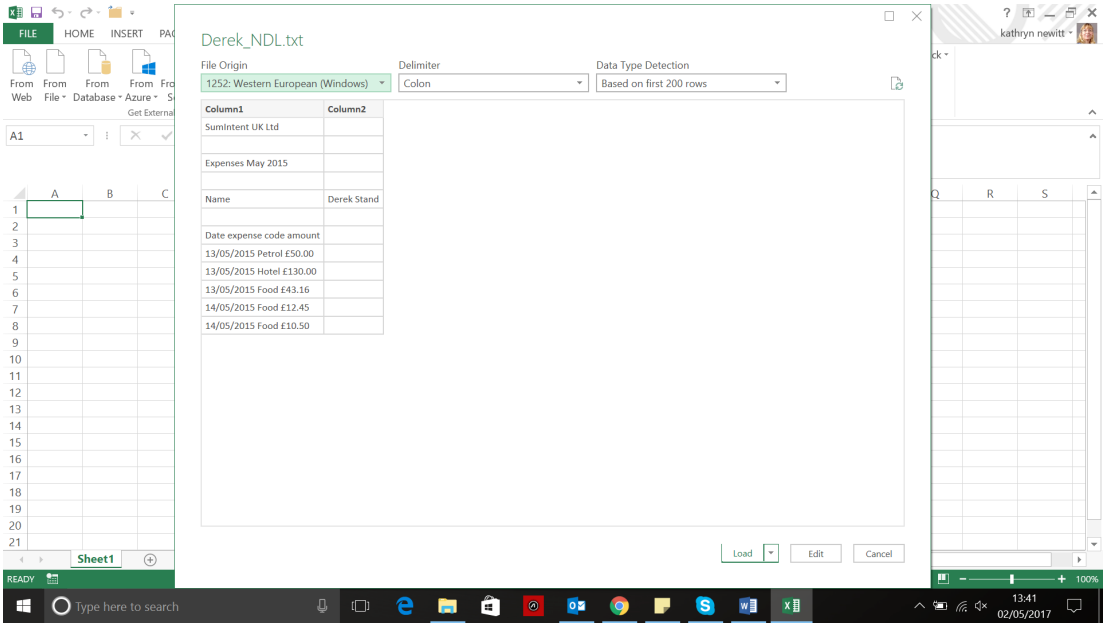
Each month we'll reproduce one of our articles on Power Query (Excel 2010 and 2013) / Get & Transform (Excel 2016) from www.sumproduct.com/blog. If you wish to read more in the meantime, simply check out our Blog section each Wednesday. This month, we look at importing data from non-delimited text files.

As programmers, we are often asked to produce delimited versions of reports which could easily be picked up by Excel. Whilst it is possible to clean non-delimited files in Excel, it's a laborious and repetitive process, which is why it made more sense to pay for a programmer to automate it. Power Query is a **free** method of cleaning up these files, and since the steps are recorded, it can be reapplied.

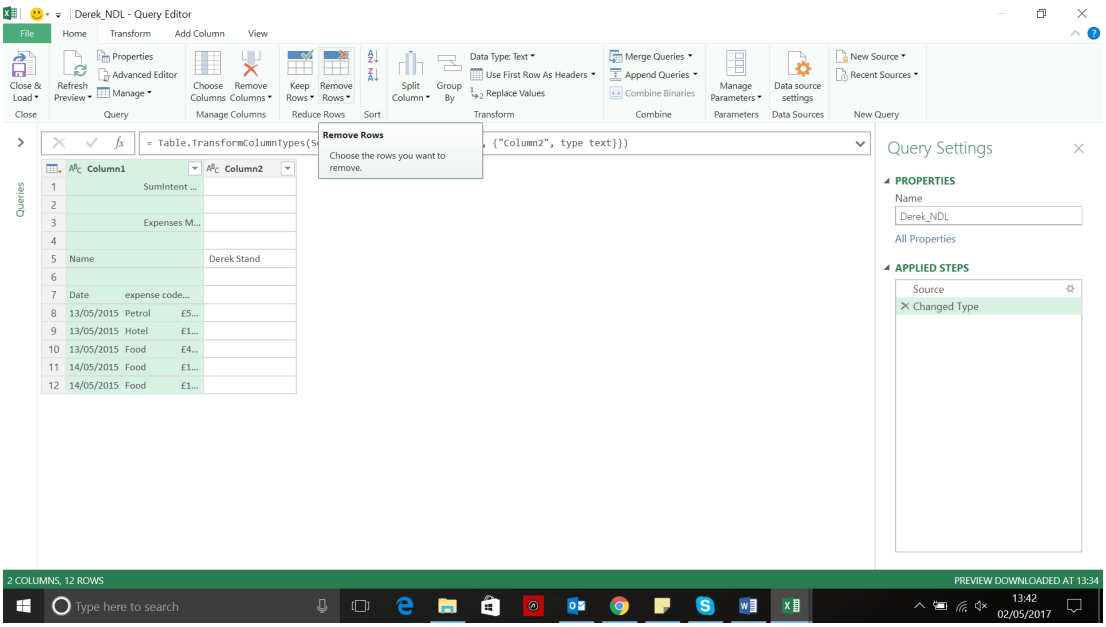
Let's begin by creating a new query from my file, as shown below:



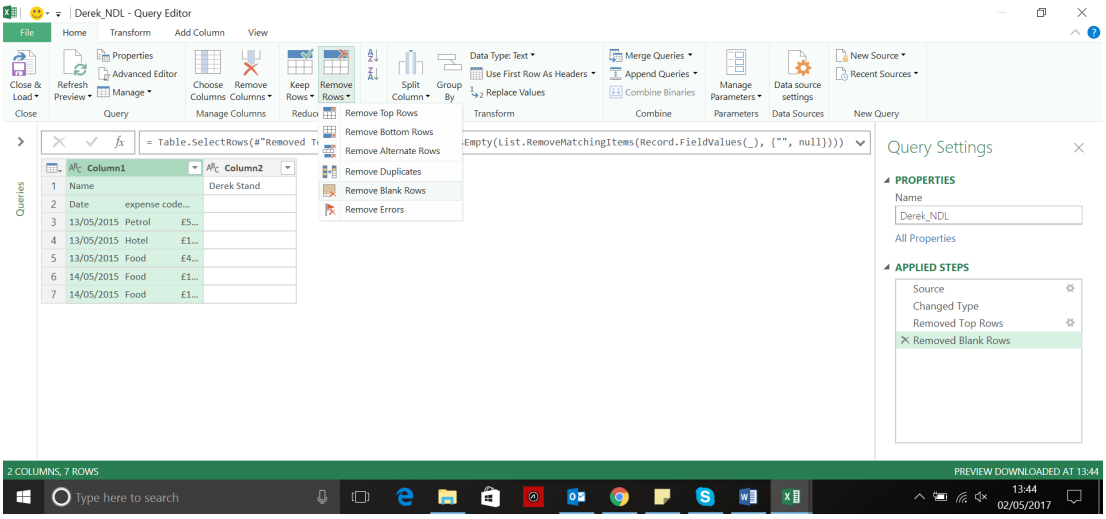
To begin with, let's select the file we require. Power Query has made a valiant effort at delimiting my data as there happened to be a colon between the 'Name' title and the name, but most of it is in one column. The goal here is to split the data into more columns.



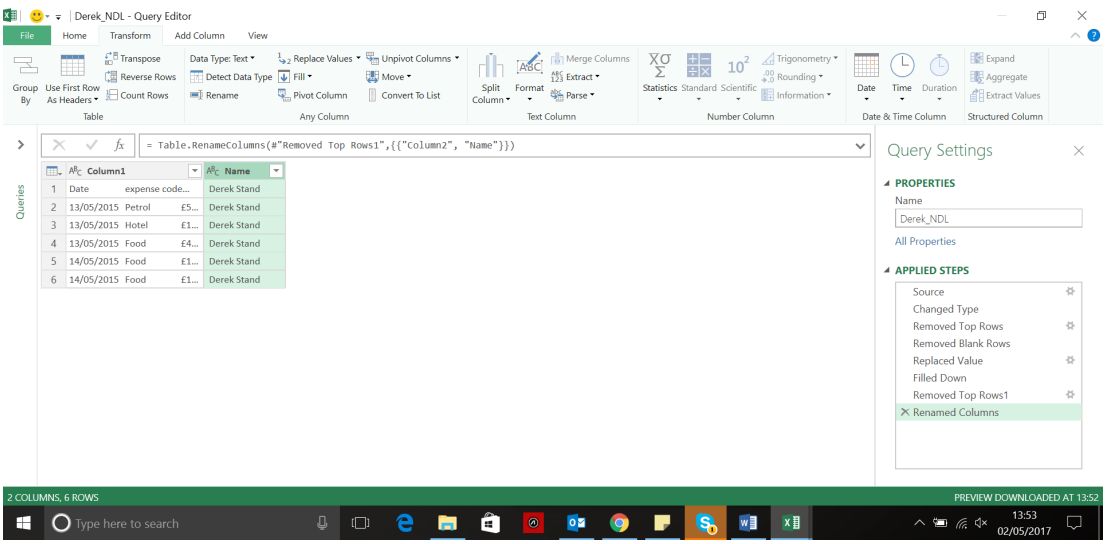
The first four rows don't include any data we want, so we shall edit and remove these. In the 'Home' tab, there is a section to do just this:



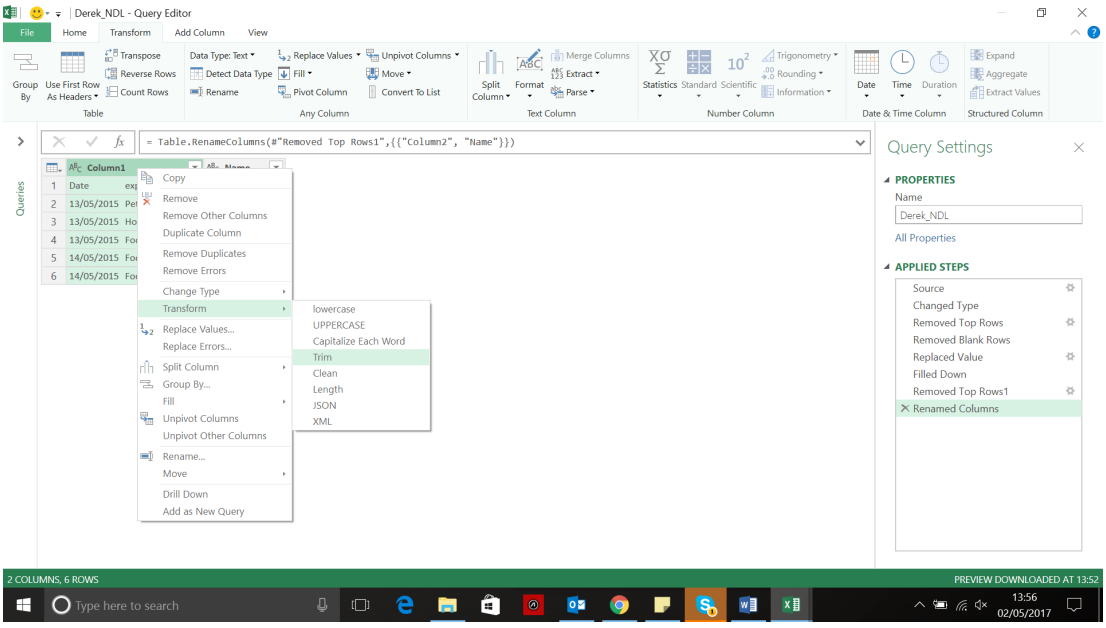
Therefore, we remove the first four rows and then the blank row after the ***Name*** row. We could have done this in one step by just removing the blank rows, as shown below:



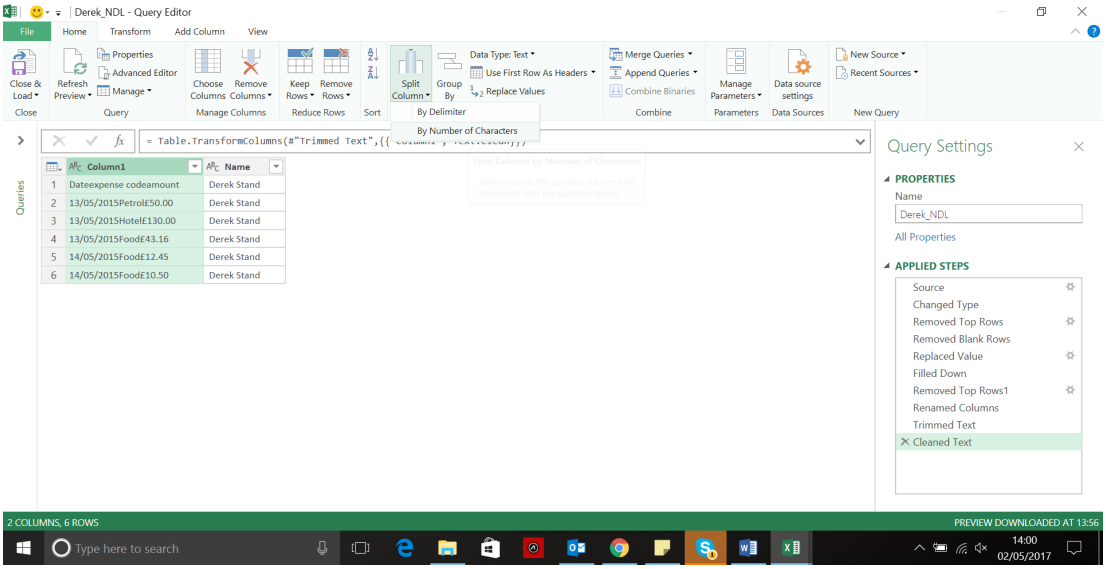
Now we want to populate ***Column2*** all the way down. We've described this process in detail in previously where the aim is to replace the blank names with null and then fill down. We can then remove the first row and rename the column:



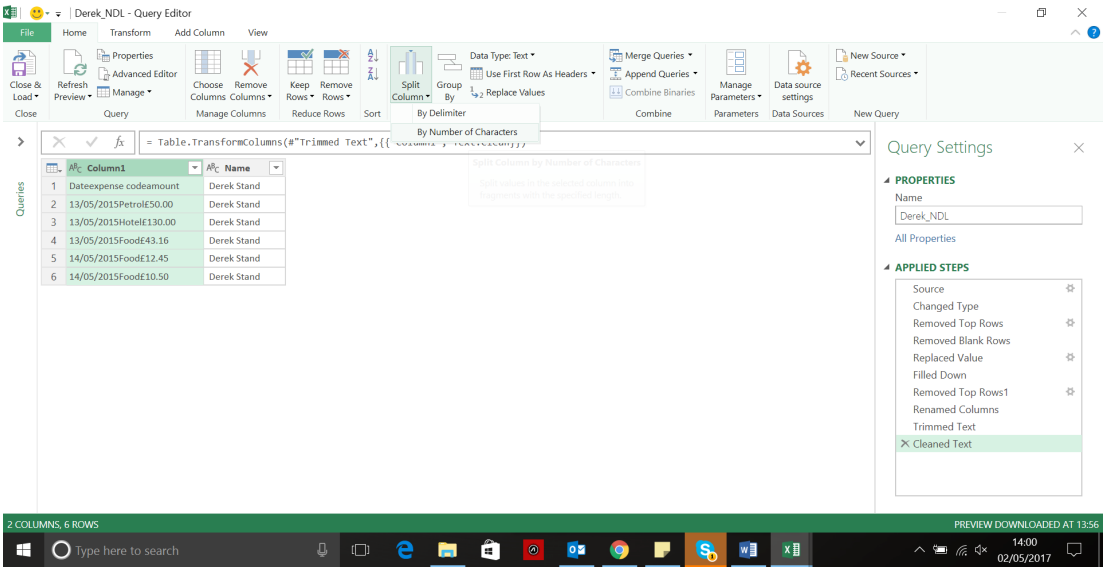
In order to make sure my data in ***Column1*** is as clean as it can be, we'll trim and clean it in order to remove any leading and trailing spaces, along with any annoying escape codes. Right-clicking the column reveals these options:



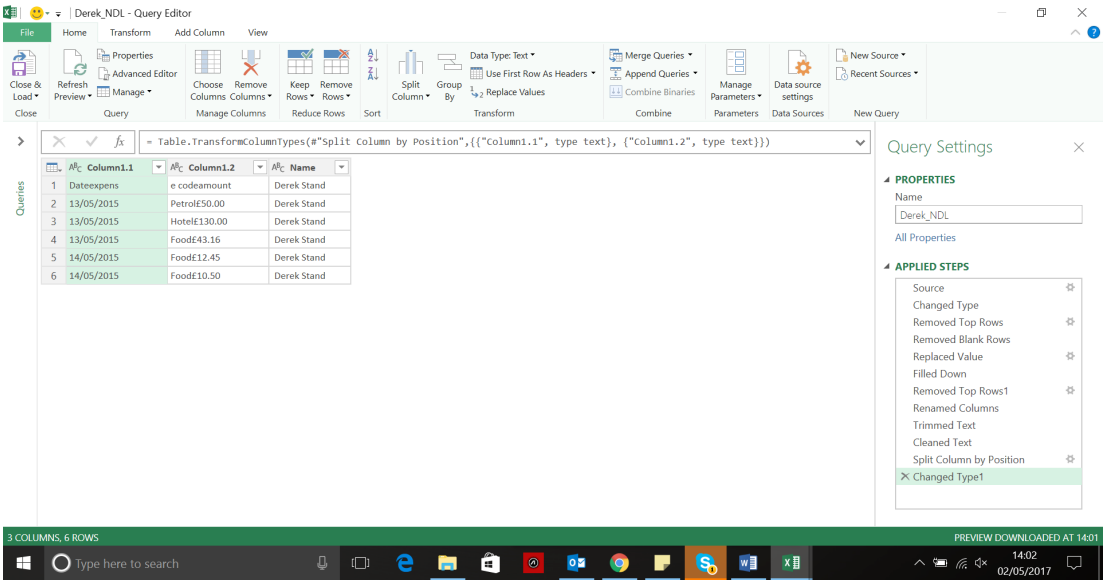
The next step is to break up the data in column 1, and to do this, there is a 'Split Column' option in the 'Home' section:



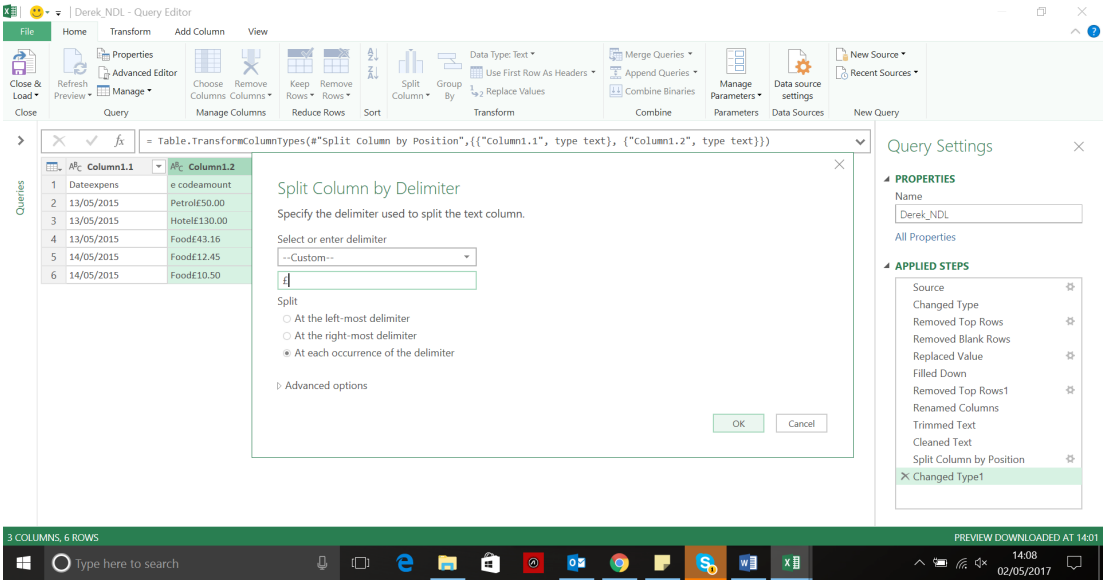
We'll make a guess at where to split the data – since we can edit the step by clicking on the gear icon next to the step in the 'APPLIED STEPS' window, we can adjust this until our data looks right.



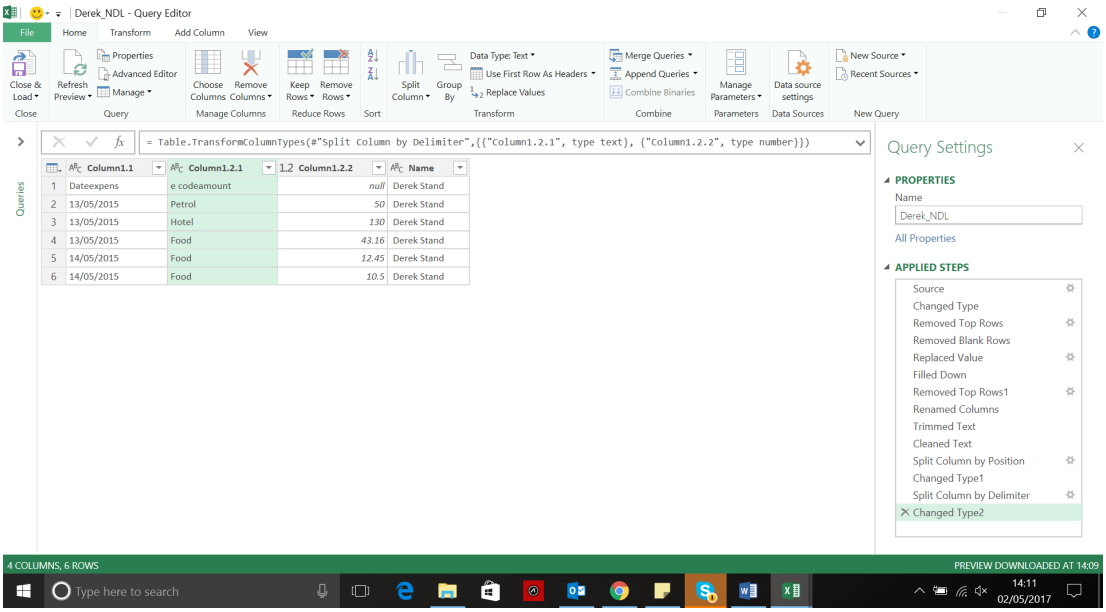
The date data looks good. For the new column ***Column1.2***, let's use the '£' sign as a delimiter instead of using a count because the expense codes vary in length:



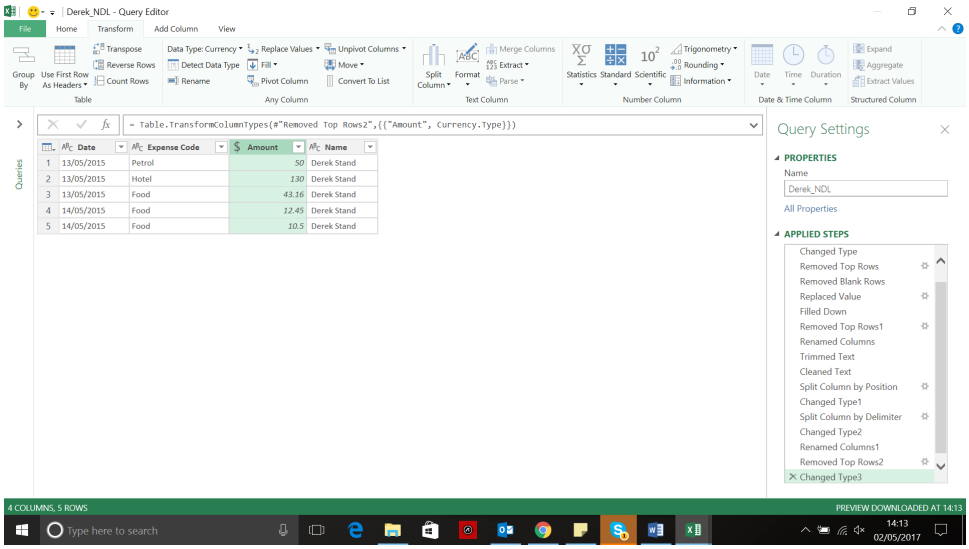
The end result looks promising



There's not much point trying to use the top row as headings, so we'll delete the top row and rename the columns.



There's not much point trying to use the top row as headings, so we'll delete the top row and rename the columns.



The data looks ready to load. This query can be applied to another delimited expenses text file that comes through in a similar format, and the steps can be adjusted as required. Easy!

July 2018 update for Get & Transform in Excel 2016 and the Power Query add-in

One of our catchiest section titles returns as Power Query / Get & Transform has an arguably overdue facelift. It's been a while coming, but late July saw the announcement of new refinements / updates to the Extract, Transform & Load (ETL) tool.

It should be noted that these updates are available as part of an Office 365 subscription. However, if you have Excel 2010 or Excel 2013, you can also take advantage of these updates by downloading the latest Power Query for Excel add-in. The jury's out presently about how the updates will go through the Excel 2016 "perpetual" edition.

The following new features and improvements have been introduced in this release:

- Add Column From Examples enhancements
- Always Use Connection File support
- ODBC Connector enhancements
- Support for alternate Windows credentials in OLEDB connector
- OData V4 Connector enhancements
- Port selection in SAP HANA connector.

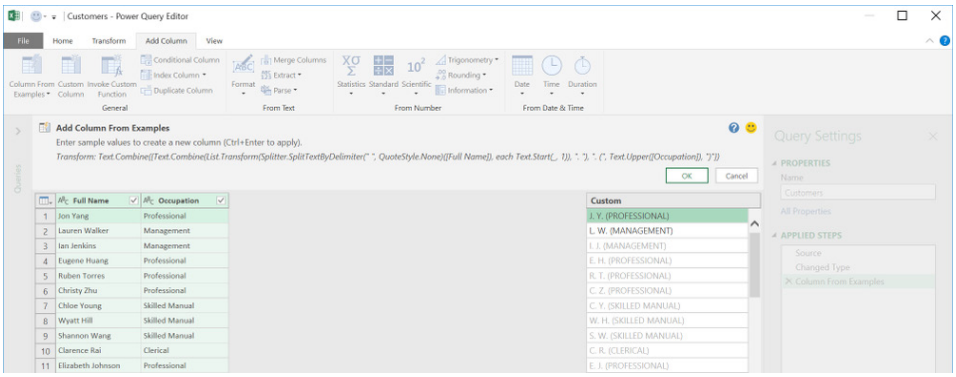
Let's take a brief look at each of these in turn.

Add Column From Examples enhancements

Add Column From Examples, which can be accessed via the 'Add Column' tab on the 'Query Editor' Ribbon, enables users to add new columns that derive data from existing columns by simply providing one or more sample values for your new column. It's a brilliant tool for getting started. The Query Editor then automatically identifies and applies the required transformations for the new column.

This update saw significant enhancements to the 'Add Column From Examples' feature:

- **Composition of data transformations:** users may now derive new columns from examples that require the composition of multiple column transformations. For example, the image below shows how you can extract the name initials followed by the occupation in upper-case in parenthesis from the Full Name and Occupation columns:

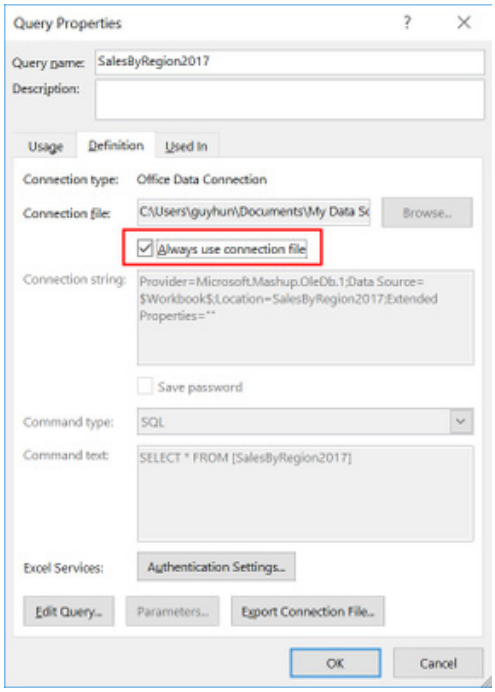


- **Domain specific transformations:** Microsoft has enhanced the set of supported data transformations even further by including specialised, domain-specific transformations, such as additional Date extractions (*e.g.* 5/8/2018 may now become MAY-2018) and more.

Always Use Connection File support

A popular scenario in the data access domain in Excel is the ability to share queries. Today, users are able to export query definitions into an Office Database Connection (ODC) file, then share it across workbooks or with their fellow co-workers. Once someone receives an ODC file, they may consume it and import the query definition into their workbook.

With this update, you can force the query to always use the most up-to-date definition as stored in the ODC file whenever the data is displayed or refreshed. Right-click on a query which is linked to an ODC file in the 'Queries & Connections' side pane, choose 'Properties', go to the 'Definition' tab and select the 'Always use connection file' check box. Clear this check box to use the query definition as stored in the Excel workbook.



Note: if the ODC file is not available, Excel resorts to the query definition that is saved in the workbook. If you want to ensure that the most up-to-date version of the query is always used, make sure that the ODC file is readily accessible and available.

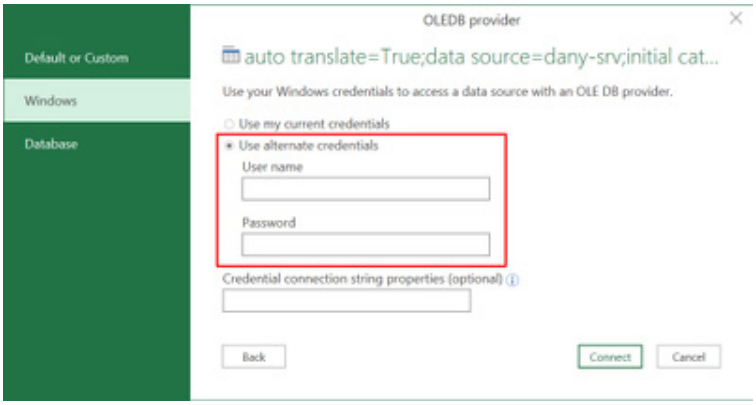
ODBC Connector enhancements

This update introduced the following improvements to the ODBC connector in Excel:

- The 'Keep top rows' transformation command will be pushed down to the ODBC driver, which may improve performance of the connector if the driver and underlying data source support the "top" operator
- If the DSN or connection string specified in the ODBC connector dialog includes a DSN catalog *(sic)*, Excel will narrow down the list of tables exposed in the 'Navigator' dialog accordingly.

Support for alternate Windows credentials in OLEDB connector

This release had enhanced the authentication options for the OLEDB connector and added the ability to provide alternate user credentials when using Windows authentication.



OData V4 Connector enhancements

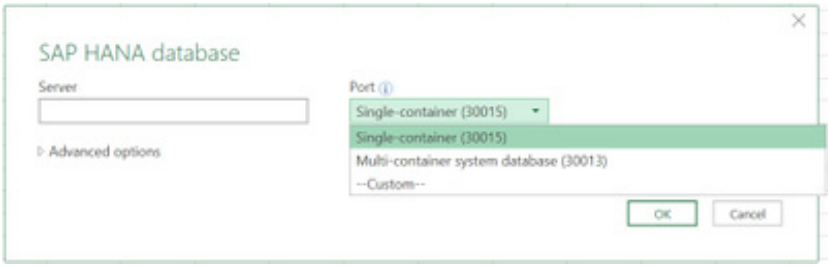
There have been several improvements made to the OData connector in order to provide better support for OData V4:

- Data load and refresh will be significantly faster specially when the feed contains complex types
- Greater resiliency: smarter query folding logic increases the query success rate
- Improved support for complex types and open type navigation columns
- Improved support for custom URLs: when the user specifies OData query options manually, Excel will adjust the type of the imported table according to the response.

Port selection in SAP HANA connector

With this update of the SAP HANA connector, users can explicitly select the port to use when connecting to a SAP HANA database. In previous releases, Port could be specified as part of the 'Server input' field. However, the default port if not specified would not distinguish between single-container and multi-container SAP HANA servers.

With the introduction of this feature, users may now specify what type of SAP HANA server they are connecting to and get an optimised port default for it, or alternatively select the 'Custom' option to specify a different port number.



July Updates for Power BI Desktop

Microsoft delayed its announcements for July updates so that they may be presented first at the Microsoft Business Applications Summit. This has meant we have only just go them into this month's newsletter!

For the record, the July updates – and there are lots of them – are as follows:

Modelling

- Composite models

Reporting

- New visual header with more flexibility and formatting
- Theming update – more visual and page control (Preview)
- Wallpaper formatting
- Tooltips for table and matrix
- Turn tooltips off for visuals
- Slicer accessibility
- Formatting pane improvements
- Stepped line support in line and combo charts
- Turn off combo chart data labels for individual series
- Sorting experience improvement

Analytics

- Distribution factor insights

Custom Visuals

- Power BI certified category
- Disabling specific organizational visual
- Visio custom visual Generally Available
- Mapbox custom visual Generally Available
- DataText Box custom visual
- China Scatter Map custom visual

Data Connectivity

- IBM DB2 DirectQuery connector (Preview)
- Improvements to Web By Example connector
  - Support for importing multiple custom tables
  - Automatic completions for specifying sample values
  - Exposure of attribute selectors in Web.BrowserContents function
- SAP HANA – Default values for variables in Variable Input experience

As usual, let's go through each new feature in turn.

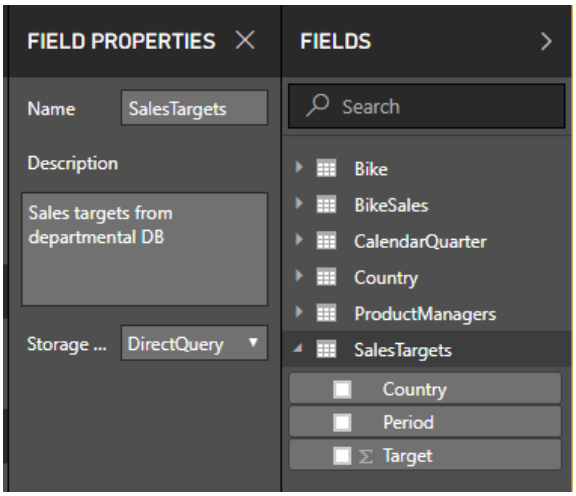
Composite models

Up until now in Power BI, when connecting to a data source using DirectQuery, it is then not possible to connect to any other data source in the same report or to include data that has been imported. This new Composite models feature removes this restriction, allowing a single report to seamlessly combine data from one or more DirectQuery sources, and / or combine data from a mix of DirectQuery sources and import data.

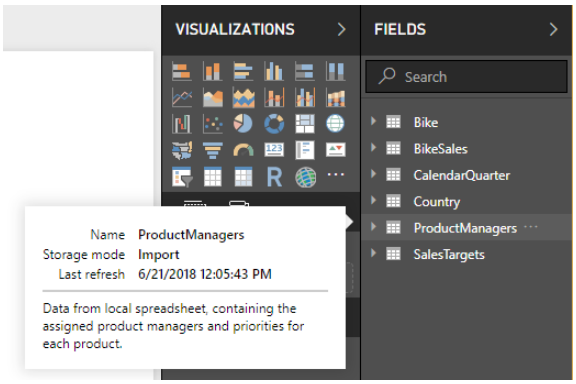
In addition to allowing DirectQuery and import sources to co-exist in the same file, composite models include a new feature called dual storage mode. If you are using DirectQuery currently, all visuals will result in queries being sent to the backend source, even for simple visuals such a Slicer showing all the Product Categories. The ability to define a table as having a storage mode of "Dual" means that a copy of the data for that table will also be imported, and any visuals that reference only columns from this table will use the imported data, and not require a query to the underlying source. The benefits of this are improved performance, and lessened load on the backend source.

To enable these features in Power BI Desktop, you'll need to enable the Preview by going to **File > Options and settings > Options > Preview Features** and select **Composite Models**.

Once you do this, you'll be able to start importing additional tables to your DirectQuery model with no additional changes to your report. If you want to take advantage of the dual storage model, you can set this for a table in the field properties pane.



You'll be able to see the storage mode in the tooltip for the table as well.



If your report has some DirectQuery tables and some import tables, the status bar on the bottom right of your report will show a storage mode of 'Mixed'. Clicking on this allows all tables to easily be switched to import mode.

While this feature is in Preview, it is currently not possible to publish a model that uses any of these features to the Power BI service.

In essence, this Preview feature "Composite models" encompasses three related features:

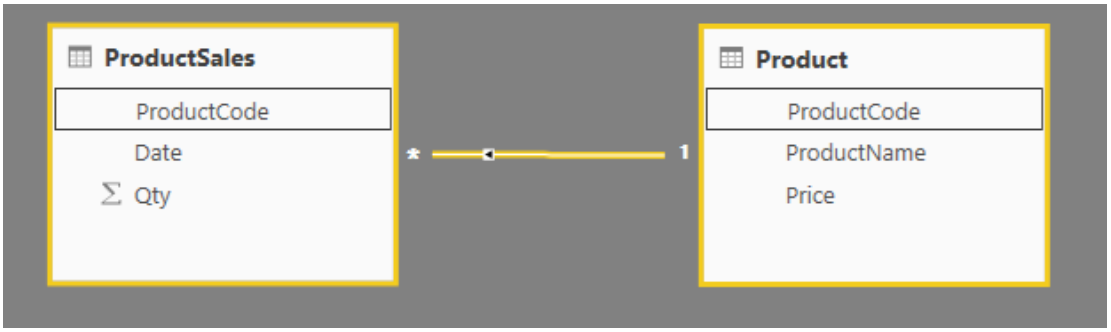
- **Composite models:** Currently in Power BI, when connecting to a data source using DirectQuery, it is then not possible to connect to any other data source in the same report, or to include data that has been imported. The new "Composite models" feature removes this restriction, allowing a single report to seamlessly combine data from one or more DirectQuery sources, and / or combine data from a mix of DirectQuery sources and import data

- **Many to many relationships:** Currently when defining a relationship between two tables in Power BI, at least one of the columns involved in the relationship must contain unique values. However, sometimes that is not the case. For example, two tables that have a column containing the Country, but where the values of Country are not unique in either table. To then join between such tables, it is necessary to use a workaround, introducing additional tables into the model that contain the necessary unique values. The "Many to Many relationship" feature will provide an alternative approach, allowing such tables to be joined directly using a relationship with a cardinality of "Many-to-many". Many have been looking forward to this!

- **Storage mode of "Dual"**: Currently when using DirectQuery, all visuals will result in queries being sent to the backend source, even for simple visuals, *e.g.* a Slicer showing all the Product Categories. The ability to define a table as having a storage mode of "Dual" means that a copy of the data for that table will also be imported, and any visuals that reference only columns from this table will use the imported data, and not require a query to the underlying source. The benefits of this are improved performance, and lessened load on the backend source.

Many to many relationships is important enough to warrant further discussion. When defining a relationship between two tables in Power BI, it is necessary to define the cardinality of that relationship. For example, the relationship between ProductSales and Product (using columns ProductSales[ProductCode] and Product[ProductCode]) would be defined as being Many to One, as there are many sales for each product, and the column in the Product table (ProductCode) is unique. When defining a relationship cardinality as Many to One, One to Many, or 1-1, Power BI will perform validation to ensure this matches the actual data.

For example, consider the simple model below:



Then assume the Product table contained just two rows:

ProductCode	ProductName	Price
A	Name for Product A	20
B	Name for Product B	23

and the Sales table just four rows, including Sales for a product "C" that does not exist in the Product table (due to a referential integrity error).

ProductCode	Date	Qty
A	1/1/2018	10
A	1/2/2018	20
B	1/1//2018	50
C	1/1/2018	1000

A visual that showed ProductName and Price (from the Product table), along with the total Qty for each product (from the ProductSales table) would be as below:

ProductName	Price	Qty
Name for Product B	23	50
Name for Product A	20	30
		1000
Total		**1080**

There is a row in the visual with a blank ProductName, accounting for the sales for the product C. This blank row accounts for:

- any rows in the ProductSales table for which there is no corresponding row in the Product table, *i.e.* there is a referential integrity issue as for product "C" in this example
- any rows in the ProductSales table for which the foreign key column is Null.

Hence in both cases the blank row accounts for sales where the ProductName and Price are unknown. However, it is sometimes the case that the tables are to be joined by two columns, yet neither column is unique. For example, consider the following two tables:

- The Sales table contains sales data by State, with each row containing the sales amount for the type of sale in that state (including states CA, WA, and TX)

State (Sales)	Type	Sales
CA	Internet	60
CA	Store	80
TX	Store	400
WA	Internet	150
WA	Store	100

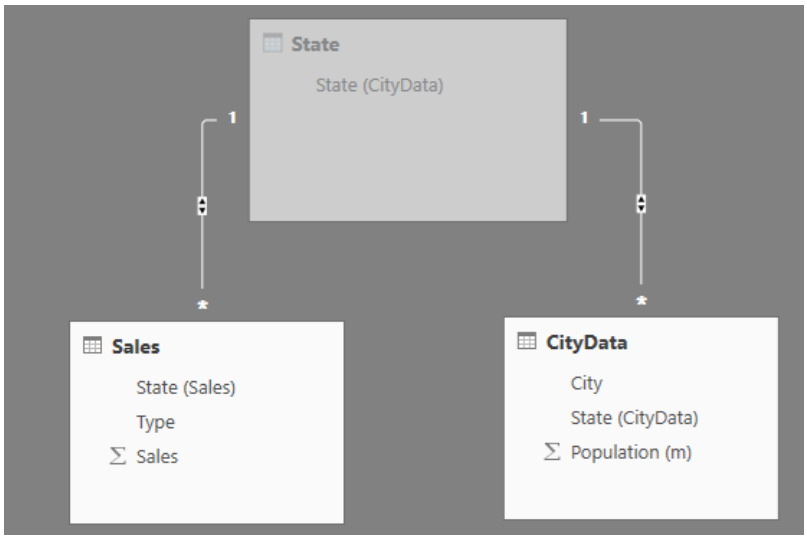
- The CityData table contains data on cities, including the population and state (including states CA, WA, and New York)

State (CityData)	City	Population (m)
CA	Los Angeles	4.00
CA	San Fransisco	0.90
New York	New York	8.50
WA	Seattle	0.70
WA	Spokane	0.20

Whilst there is a column for State in both tables, and it is reasonable to want to report on total Sales by State, along with the total population of each State, the State column is not unique in either table.

In versions of Power BI prior to July 2018, it was not possible to create a relationship directly between these tables. Instead, a common workaround was to:

- Create a third table containing just the unique State IDs. This could be either a calculated table (defined using DAX or M), or a table defined using a query defined in the query editor, and could contain the unique ids drawn from one of the tables, or the unioned full set
- Relate the two original tables to this new table, using regular Many to One relationships.



A visual showing State (from the CityData table) along with the total Population and total Sales would then be as follows:

State (CityData)	Population (m)	Sales
CA	4.90	140
New York	8.50	
WA	0.90	250
Total	**14.30**	**790**

Note that given the use of the state from the CityData table, only those items in State in that table are listed (and hence TX is excluded). Also, unlike the case of Many to One relationships, whilst the total row includes all Sales (including those of TX) the details do not include a blank row covering such mismatched rows. Similarly, there would be no blank row covering any Sales which had a null value for the State.

If the City were also added to the visual, then whilst the population per City is known, the Sales shown for the City would simply repeat the Sales for the corresponding State (as is normally the case when grouping on a column that is not related to some aggregate measure).

State (CityData)	Population (m)	Sales
CA	**4.90**	**140**
Los Angeles	4.00	140
San Fransisco	0.90	140
New York	**8.50**	
New York	8.50	
WA	**0.90**	**250**
Seattle	0.70	250
Spokane	0.20	250
Total	**14.30**	**790**

If the new table Sales were defined to be the union of all States, and made visible in the field list, then the same visual showing State (on the new table) along with the total Population and total Sales would then be as follows:

State	Population (m)	Sales
CA	4.90	140
New York	8.50	
TX		400
WA	0.90	250
Total	**14.30**	**790**

In this case, TX (with Sales but unknown population) and New York (with a known population but no Sales) would be included.

In versions of Power BI since July 2018, it is possible to directly relate such tables without needing to resort to such workarounds. It is possible to set the cardinality of a relationship to "Many to Many", indicating that neither table contains unique values. For such relationships, it is still possible to control which table filters the other table, or to have bi-directional filtering where both tables filter each other.

The cardinality will default to "Many to Many" if it is determined that neither table contains unique values for the columns in the relationship. In this case, a warning is displayed, to ensure it is the intended behaviour, and not the unintended effect of a data issue.

For example, in creating a relationship directly between CityData and Sales, where Filters should flow from CityData to Sales, the resulting 'Relationship' view would then contain the direct "Many to Many" relationship between the two tables. The resulting appearance in the field list, and subsequent behaviour as visuals are created, is then the same as employing the workaround described above where the extra table (with the distinct States in it) is *not* made visible. For example, as above, a visual showing States along with total population and sales would be as follows:

State (CityData)	Population (m)	Sales
CA	4.90	140
New York	8.50	
WA	0.90	250
Total	**14.30**	**790**

Hence the major difference between "Many to many" relationships and the more typical "Many to One" relationships are:

- the values shown will not include a blank row accounting for any mismatched rows in the other table, or rows where the column used in the relationship in the other table is null
- it is not possible to use the **RELATED()** function (as more than one row could be related)
- using the **ALL()** function on a table will not remove filters applied to other tables related to it by a "Many to Many" relationship.

  For example, a measure defined as:

Sales_Total = CALCULATE(SUM('Sales'[Sales]), ALL('Sales'))

in the example above would not remove filters on columns on the related CityData table. Hence a visual showing State, Sales, and Sales total would result in:

State (CityData)	Sales	Sales total
CA	140	140
WA	250	250
Total	**790**	**790**

Therefore, care should be taken to ensure that calculations using **ALL(<Table>)**, such as "% of grand total", are returning the intended result.

You can connect to all sorts of different data sources **when using** Power BI Desktop **or the** Power BI Service, and you can make those data connections in different ways. You can either import data to Power BI, which is the most common way to get data, or you can connect directly to data in its original source repository, which is known as **DirectQuery**. The Power BI and DirectQuery article describes **DirectQuery** in detail.

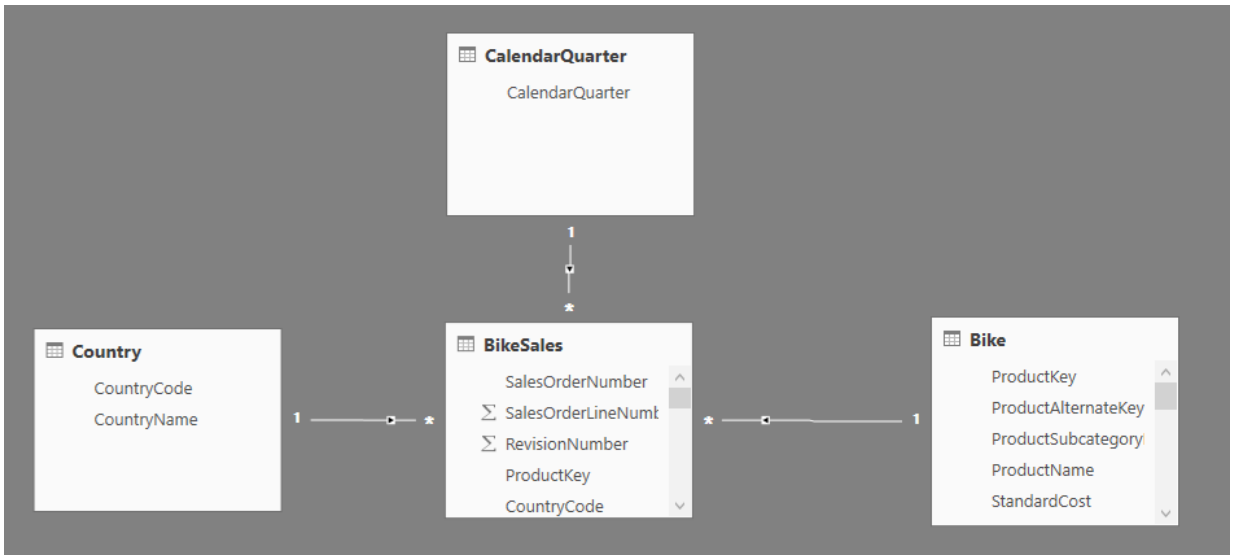
When using DirectQuery, it is now possible to create a Power BI model (*i.e.* a single **.pbix** Power BI Desktop file) that:

- combines data from one or more DirectQuery sources; *and / or*
- combines data from DirectQuery sources and import data.

For example, it is possible to build a model that combines sales data from an enterprise data warehouse, with data on sales targets that is in a departmental SQL Server database, along with some data imported from a spreadsheet. Such a model that combines data from more the one DirectQuery source, or combines DirectQuery with imported data, is referred to as a "composite model".

It is possible to create relationships between tables as normal, even when those tables come from different sources. The one restriction is that any relationships that are cross-source must be defined as having a cardinality of "Many to Many", irrespective of the actual cardinality. The behaviour of such relationships is then the same as normal for "Many to Many" relationships (as described above). Note that within the context of composite models, all imported tables are effectively a single source, irrespective of the actual underlying data source that they are actually imported from.

As an example of a composite model, consider a report that has connected to a corporate data warehouse (in SQLServer), using DirectQuery, where the data warehouse contains data of Sales by Country, Quarter, and Bike (Product).



Certainly, now you could build simple visuals using fields from this source. For example, the visual below shows total Sales amount by ProductName, for a selected quarter.

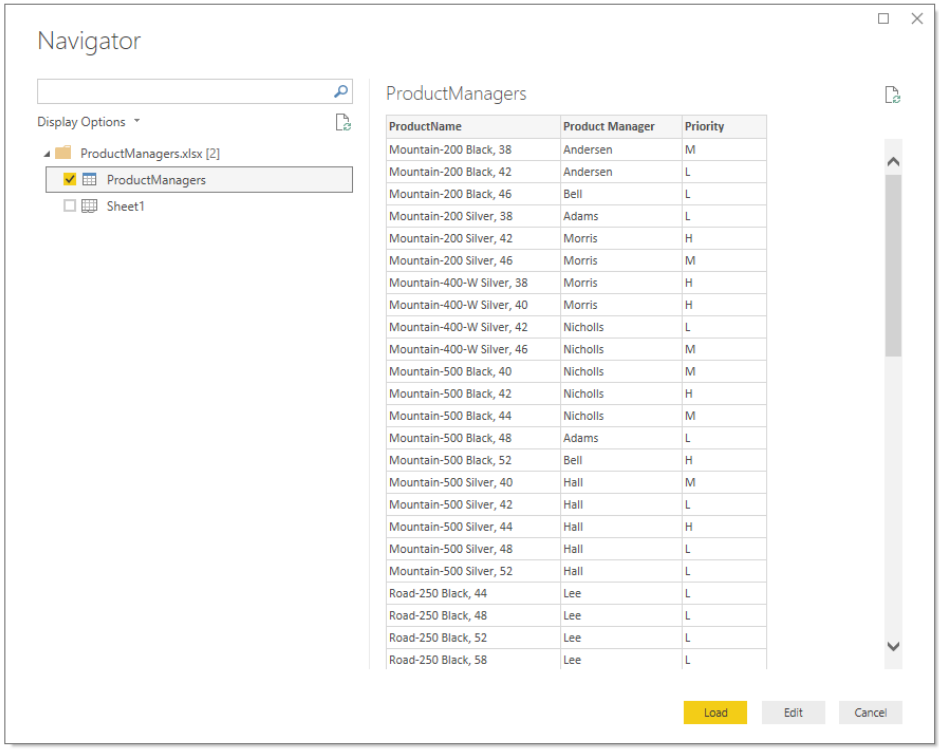
CalendarQuarter

2004 Q2

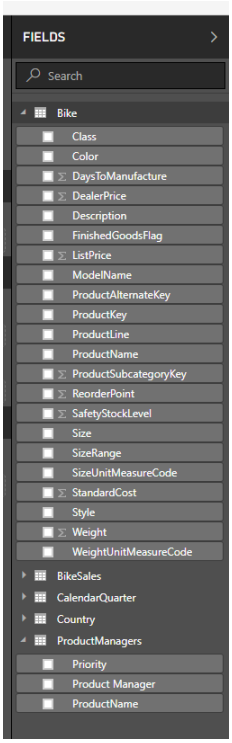
ProductName	SalesAmount
Mountain-200 Black, 38	$725,631.7742
Mountain-200 Black, 42	$638,035.6779
Mountain-200 Black, 46	$546,907.133
Mountain-200 Silver, 38	$668,400.255
Mountain-200 Silver, 42	$570,032.679
Mountain-200 Silver, 46	$547,639.2075
Mountain-400-W Silver, 38	$70,485.284
Mountain-400-W Silver, 40	$82,951.022
Mountain-400-W Silver, 42	$66,330.038
Mountain-400-W Silver, 46	$72,024.264
Mountain-500 Black, 40	$24,299.55
Mountain-500 Black, 42	$28,511.472
Total	$12,299,251.4178

But what if you had some information on the Product Manager who had been assigned to each product, along with the marketing priority, where that data is maintained in an Excel spreadsheet? You might well want to see Sales Amount by Product Manager, yet having this local data added to the corporate data warehouse would likely be unfeasible or take months at best. Whilst it is of course possible to instead import the sales data from the data warehouse, whereupon it can readily be combined with data imported from elsewhere, this is almost certainly not feasible given the reasons that lead to the use of DirectQuery in the first place, namely some combination of the security rules enforced in the underlying source, the need to be able to see the latest data, and the sheer scale of the data.

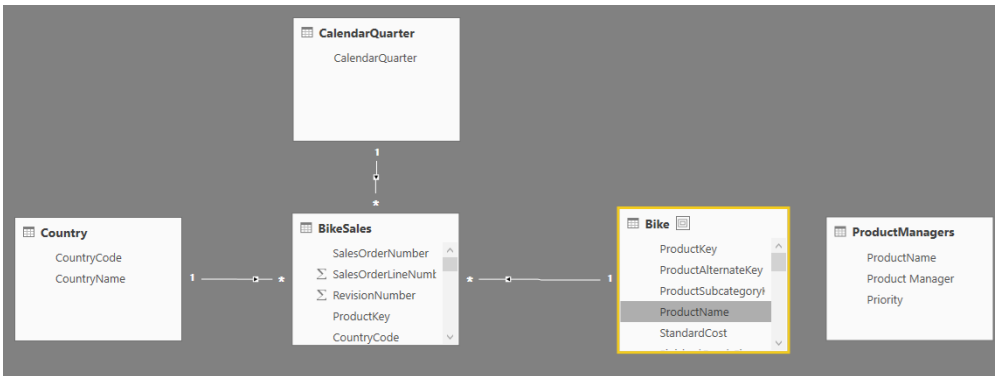
Composite models provide the option of still connecting to the data warehouse using DirectQuery, but also using GetData for additional sources. In this case, we can use GetData, choose Excel, navigate to the spreadsheet containing our local data, and are then able to import the sheet containing the ProductNames, and the assigned SalesManager and Priority.



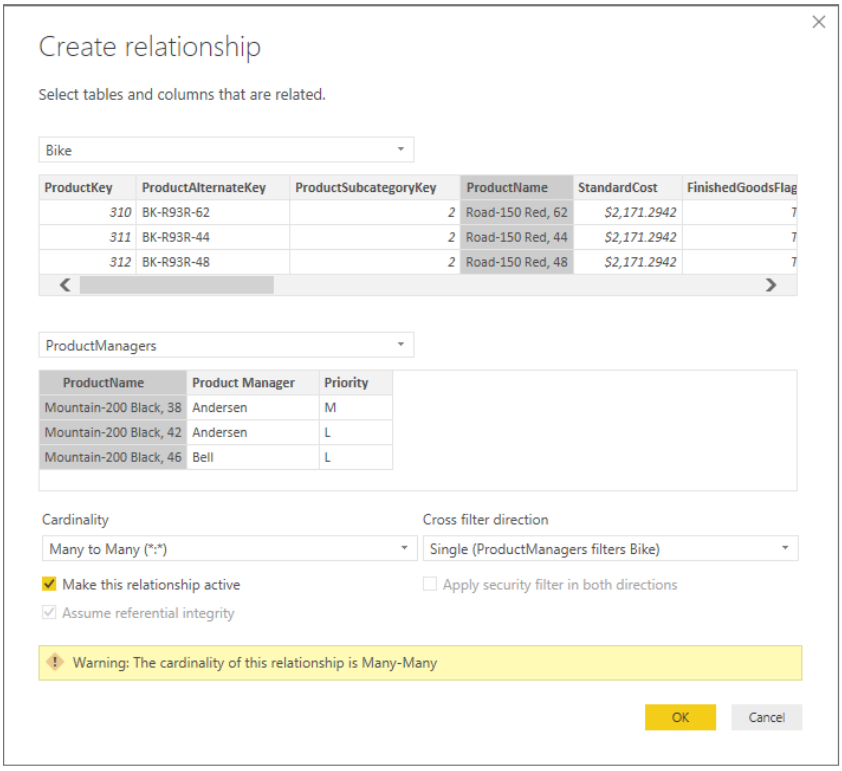
Now in the field list we see the original Bike table (from SQL Server) and a new Product Managers table (with the data from imported from Excel).



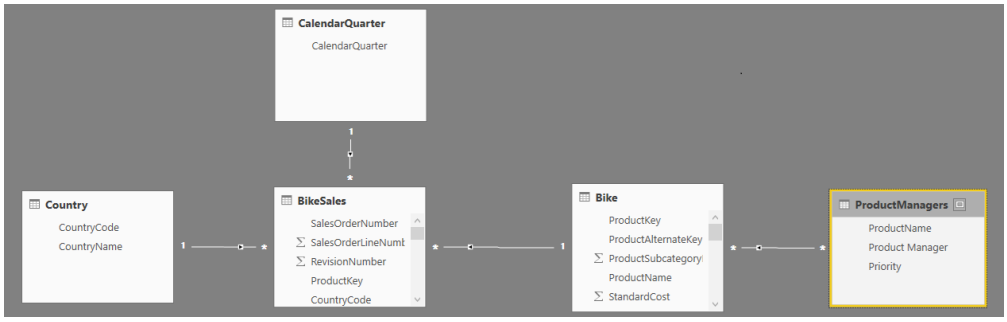
Similarly, looking at the 'Relationship' view, we now see an additional table Product Managers:



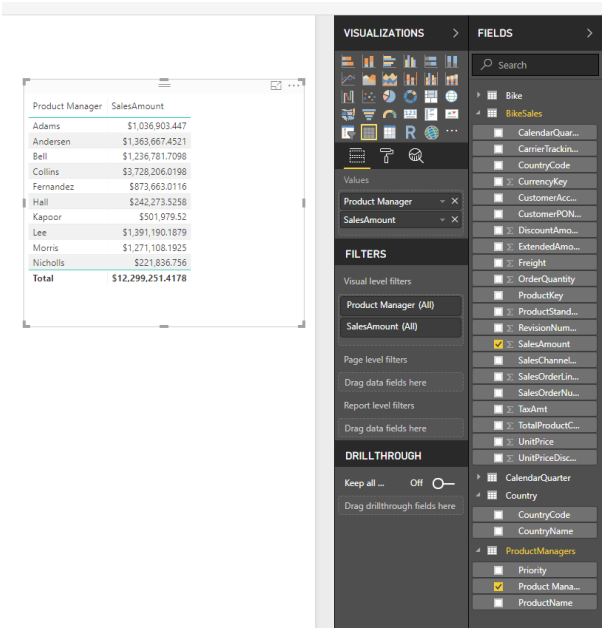
We now need to relate this to the other tables in the model, which we do in the normal way, by creating a relationship between the Bike table (in SQL Server) and the Product Managers table (that is imported), *i.e.* between Bike[ProductName] and ProductManagers[ProductName]. As discussed above, all relationships going across source must have cardinality "Many to Many", and hence that is the default cardinality chosen.



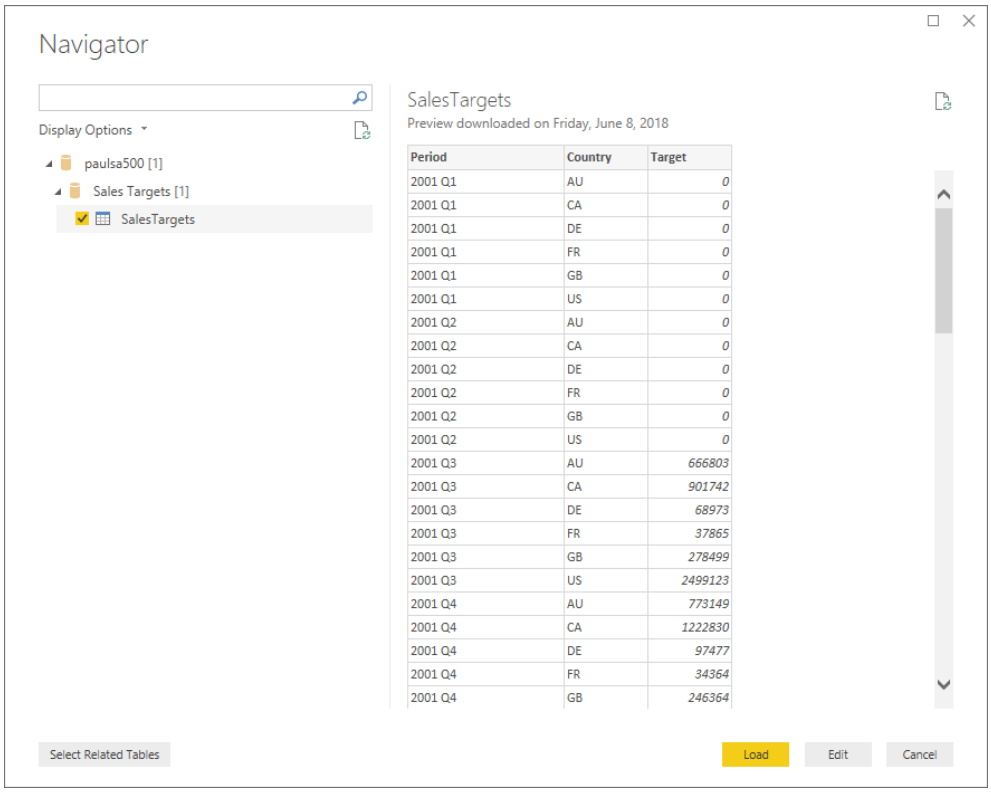
Having created this relationship, it is displayed in the 'Relationship' view as normal.



Now it possible to build visuals using any of the fields in the field list, seamlessly blending data coming from multiple sources. For example, the visual below shows total Sales Amount for each Product Manager.

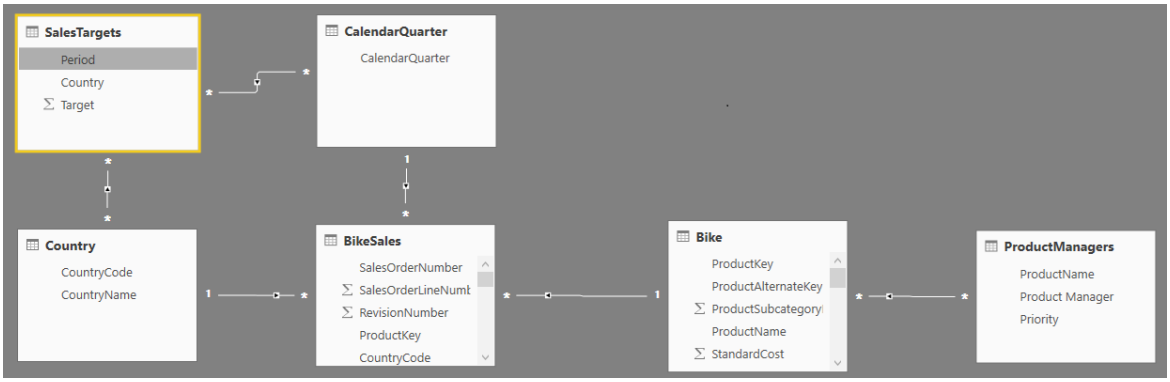


Whilst this example shows a common case of a 'dimension' table (like Product or Customer) being extended with some extra data imported from somewhere else, it is also possible to have tables use DirectQuery over different sources. So, extending the example above, say that SalesTargets per Country and Period are stored in a separate departmental database. You can use GetData to connect to that data as normal.

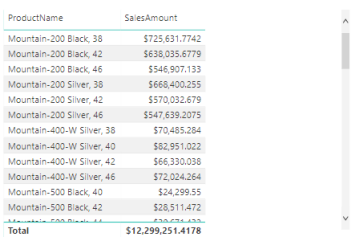


Period	Country	Target
2001 Q1	AU	0
2001 Q1	CA	0
2001 Q1	DE	0
2001 Q1	FR	0
2001 Q1	GB	0
2001 Q1	US	0
2001 Q2	AU	0
2001 Q2	CA	0
2001 Q2	DE	0
2001 Q2	FR	0
2001 Q2	GB	0
2001 Q2	US	0
2001 Q3	AU	666803
2001 Q3	CA	901742
2001 Q3	DE	68973
2001 Q3	FR	37865
2001 Q3	GB	278499
2001 Q3	US	2499123
2001 Q4	AU	773149
2001 Q4	CA	1222830
2001 Q4	DE	97477
2001 Q4	FR	34364
2001 Q4	GB	246364

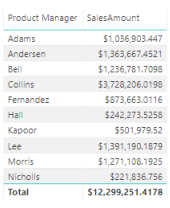
Then similar to above, it is possible to create relationships between the new table and other tables in the model, and create visuals that combine the data.



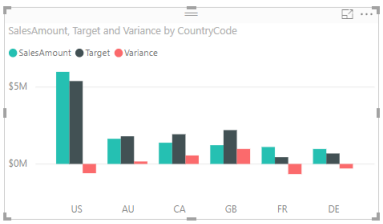
In this case, the lower visual shows total Sales Amount vs Target, with the variance calculation showing the difference, where Sales Amount and Target are coming from two different SQL Server databases.

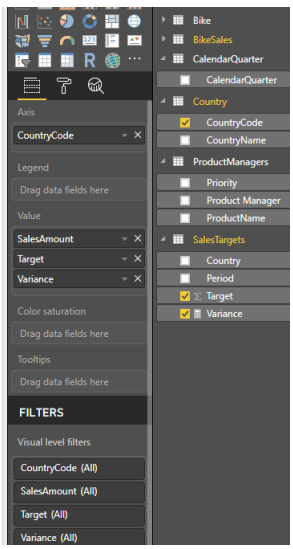


ProductName	SalesAmount
Mountain-200 Black, 38	$725,631.7742
Mountain-200 Black, 42	$638,035.6779
Mountain-200 Black, 46	$546,907.133
Mountain-200 Silver, 38	$668,400.255
Mountain-200 Silver, 42	$570,032.679
Mountain-200 Silver, 46	$547,639.2075
Mountain-400-W Silver, 38	$70,485.284
Mountain-400-W Silver, 40	$82,951.022
Mountain-400-W Silver, 42	$66,330.038
Mountain-400-W Silver, 46	$72,024.264
Mountain-500 Black, 40	$24,299.55
Mountain-500 Black, 42	$28,511.472
Total	$12,299,251.4178

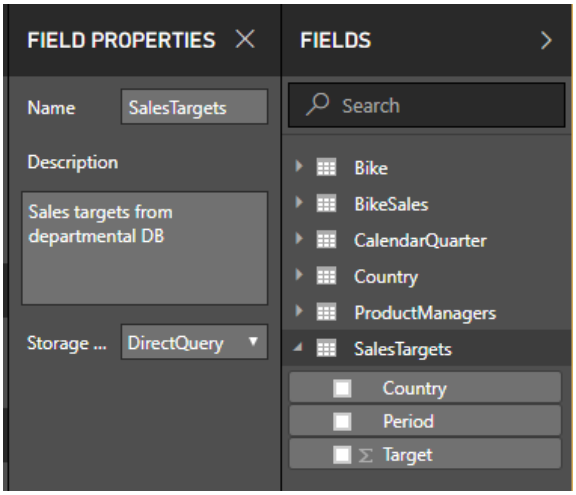


Product Manager	SalesAmount
Adams	$1,036,903.447
Andersen	$1,363,667.4521
Bell	$1,236,781.7098
Collins	$3,728,206.0198
Fernandez	$873,663.0116
Hall	$242,273.5258
Kapoor	$501,979.52
Lee	$1,391,190.1879
Morris	$1,271,108.1925
Nicholls	$221,836.756
Total	$12,299,251.4178

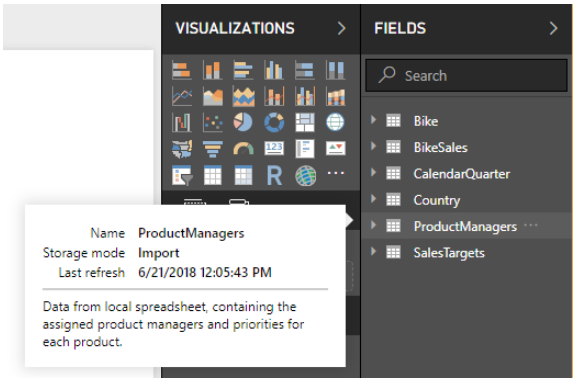




Each table has a 'storage mode', indicating whether it is DirectQuery or Import. This storage mode can be seen and changed by using the 'Property' pane (that can be displayed by selecting 'Properties' from the field list context menu).



The storage mode can also be seen on the tooltip of each table.



For a **pbix** file that has some tables DirectQuery, and some tables import, the status bar will show a storage mode of 'Mixed', and clicking on this allows all tables to easily be switched to import.

Calculated tables can be added to a model that uses DirectQuery, and the DAX defining the calculated table can reference either Imported or DirectQuery tables, or a combination of both. Calculated tables are always imported, and the data is refreshed when the table is refreshed. Hence if a calculated table references a DirectQuery table, whilst visuals referencing the DirectQuery table always shows the latest values in the underlying source, visuals referencing the calculated table show the values at the time the calculated table was last refreshed.

Composite models have some security implications. A query sent to one data source can include data values that have been retrieved from another different source. For the example above, the visual that shows Sales Amount by Product Manager will result in an SQL query being sent to the Sales relational database, where that SQL query might contain the names of Product Managers and their associated Products.

```
SELECT …
FROM …
 inner join (
(SELECT N'Andersen' AS [c52],N'Mountain-200 Black, 38' AS [c4] ) UNION ALL
(SELECT N'Andersen' AS [c52],N'Mountain-200 Black, 42' AS [c4] ) UNION ALL
(SELECT N'Bell' AS [c52],N'Mountain-200 Black, 46' AS [c4] ) UNION ALL
(SELECT N'Bell' AS [c52],N'Mountain-500 Black, 52' AS [c4] ) UNION ALL
…
```

Hence information that is stored in the spreadsheet is now being included in a query sent to the relational database. If this information is confidential, then the security implications of this should be considered. In particular:

- any administrator of the database who can see traces or audit logs would now be able to see this information, even if they did not have permissions to the data in its original source (*e.g.* permissions to the Excel file, in this example)
- the encryption settings for each source should be considered, to avoid information being retrieved from one source using an encrypted connection, but then inadvertently having it included in a query sent to another source using an unencrypted connection.

The Power BI Desktop will display a warning message at the point an action is taken to create a composite model, to allow confirmation that any security implications have been considered.

For similar reasons, care would need to be exercised if opening a Power BI Desktop file sent from an untrusted source. If that file contains composite models, it would mean that information retrieved from one source (using the credentials of the user opening the file) would be sent to another data source as part of the query (where it could possibly be seen by the malicious author of the Power BI Desktop file). Hence upon opening a file for the first time, if it contains multiple sources then a warning will be displayed. This is similar to the warning displayed upon opening a file that contains native SQL queries.

When using DirectQuery, it is always important to be thinking about performance. Primarily that means ensuring that the backend source provides sufficient performance to give a good experience, which normally means visuals should refresh in five seconds or less. Using composite models adds additional performance considerations, as now a single visual can result in it being necessary to send queries to multiple sources, often passing the results from one query across to a second source. This can result in the possible forms of execution:

- a SQL query that includes a large number of literal values. For example, a visual requesting total Sales Amount (from the SQL database) for a set of selected Product Managers (from the related table that was imported from a spreadsheet) would first need to find which Products were managed by those Product Managers, before sending an SQL query including all of the product IDs in a **WHERE** clause
- a SQL query that queries at a lower level of granularity, with the data then being aggregated locally. For the same example as above, as the number of Products meeting the filter on Product Manager becomes very large, at a certain point it becomes inefficient or unfeasible to include them all in a **WHERE** clause. Instead it is necessary to query the relational source at the lower level of Product, and then locally aggregate the results. If the cardinality of Products exceeds a limit of 1 million, then this would result in the query failing
- multiple SQL queries, one per group by value. When the aggregation uses **DistinctCount**, grouped by some column from another source, then if the external source does not support efficient passing of many literal values defining the grouping, then it is necessary to send one SQL query per group by value. For example, a visual requesting a distinct count of Customer Account Number (from the SQL Server table) by Product Manager (from the related table that was imported from a spreadsheet) would need to pass in the details from the Product Managers table in the query sent to SQL Server. Over other sources, *e.g.* Redshift, this is not feasible, and instead there would be one SQL query sent per Sales Manager (up to some practical limit, at which point the query would fail).

Each of these cases has its own implications on performance, with the exact details varying per data source. A good rule of thumb is that whilst the cardinality of the columns used in the relationship joining the two sources remains low (a few thousand) then performance should not be impacted greatly. As this cardinality grows, then more consideration needs to be paid to the impact on the resulting performance.

Additionally, the use of Many-Many relationships means that separate queries must be sent to the underlying source for each total / subtotal level, rather than aggregate the detailed values locally. Hence a simple table visual with totals would send two SQL queries, rather than one.

It should be noted that it is not yet possible to include multidimensional sources in Composite models. This includes:

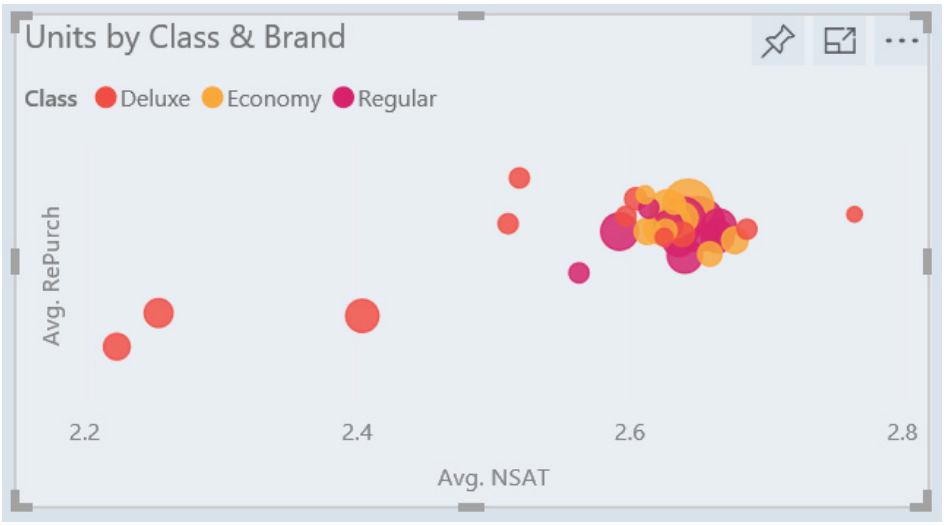
- SAP HANA;
- SAP Business Warehouse;
- SQL Server Analysis Services;
- Power BI datasets.

The limitations of using DirectQuery still apply when using composite models. Many of these limitations will now be per table, depending upon the storage mode of the table. For example, a calculated column on an imported table can refer to other tables, but a calculated column on a DirectQuery table is still restricted to refer only to columns on the same table. Other limitations apply to the model as a whole, if any of the tables within the model are DirectQuery. For example, the QuickInsights and Q&A features are not available on a model if any of the tables within it has a storage mode of DirectQuery.

New visual header with more flexibility and formatting

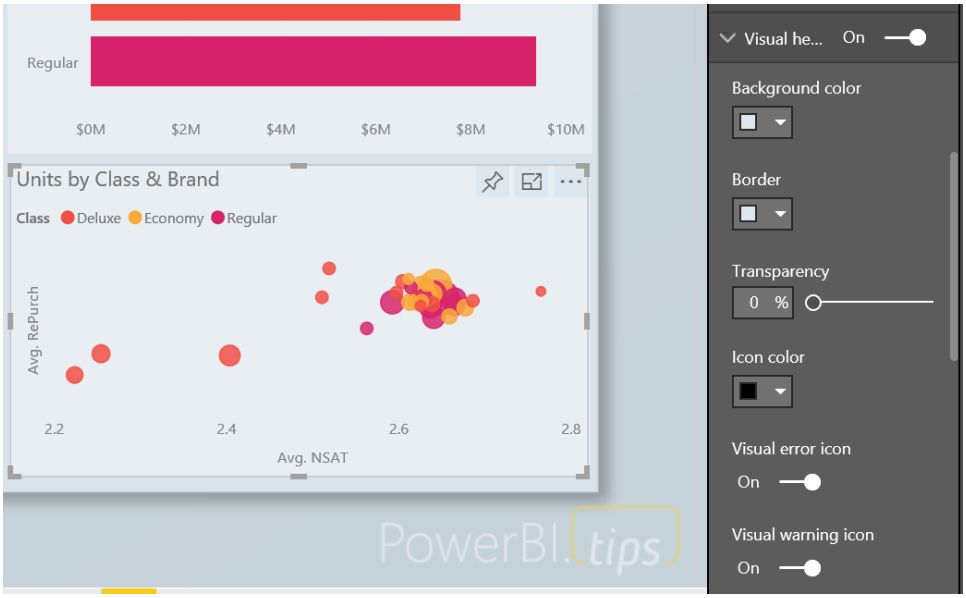
A common complaint from report authors was that the visual's header took up a lot of space and that it didn't fit with the design of their reports. Based on this feedback, Microsoft has changed the visual header.

Since the new visual header is detached from the visual, it can adapt its position based on the layout of the visual. By default, the header will appear inside the visual in-line with the title.



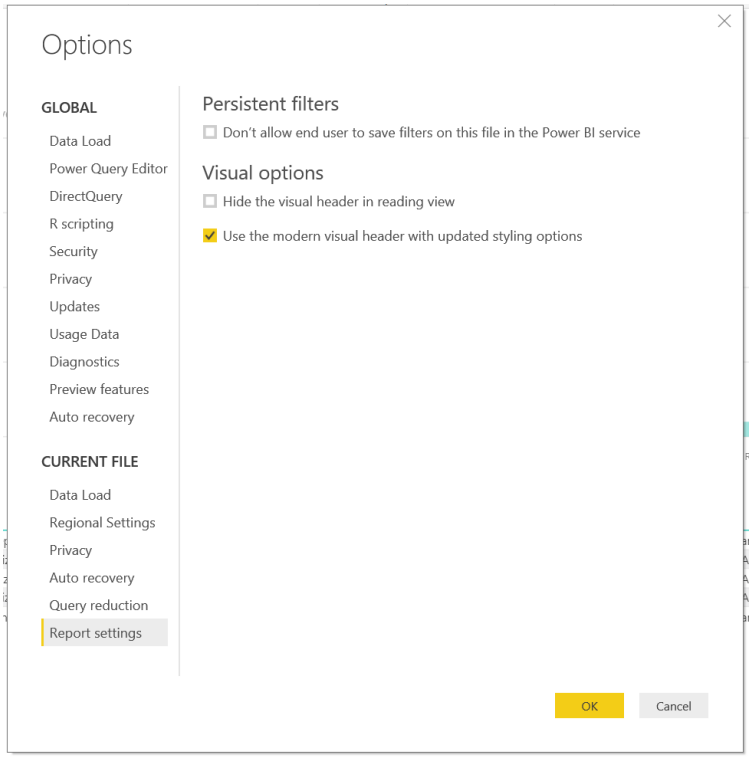
If your visual doesn't have a title, it will instead float on top of the visual, unless the visual is all the way to the top of the report, in which case it snaps to the bottom of the visual.

Every visual now has a new card in the formatting pane, called 'Visual header'. In this card, you'll find options to format the icon colour, the background and border colours and their transparency. You can also choose to turn off individual icons or the whole header for viewers of your report.



Note that the visibility toggles do not affect your report when authoring it. You will need to publish your report and view it in reading mode to see the effect. This is because many of the options in the visual header are important during editing, especially the warning icons.

This new visual header is on by default for all new reports. For existing reports, you will need to enable it in the Options dialog under the Report Settings section. Depending on your report's layout, you may notice some shifts after upgrading reports to the new header, especially with background images.



Theming update – more visual and page control (Preview)

Up until now, the majority of visual properties were able to be themed, but the visual's "container" properties were not. These container settings are things that are standard across all visuals, such as the title, border and background cards. With this month's update, all visual properties can be formatted through theming, with the exception of properties that require data in advanced (e.g. Data colours).

This update also adds both page and wallpaper background colour and transparency to the theme file as well.

Here's an example of the visual style section showing the new properties:

```
"visualStyles": {
    "*": {
      "*": {
        "title": [{
          "show": true,
          "fontColor": { "solid": { "color": "#F17925" } },
          "background": { "solid": { "color": "#000000" } },
          "alignment": "right",
          "fontSize": 11,
          "fontFamily": "Arial"
        }],
        "background": [{
          "show": true,
          "color": { "solid": { "color": "#000000" } },
          "transparency": 50
        }],
        "lockAspect": [{
          "show": true
        }],
        "border": [{
          "show": true,
          "color": { "solid": { "color": "#F17925" } }
        }],
        "visualTooltip": [{
          "type": "Default"
        }],
        "stylePreset": [{
          "name": "None"
        }]
      }
    },
    "page": {
      "*": {
        "background": {
          "color": { "solid": { "color": "#F17925" } },
          "transparency": 0
        },
        "outspace": {
          "color": { "solid": { "color": "#FFFFFF" } },
          "transparency": 0
        }
      }
    }
  }
```

Do note that if you are wildcarding color (using a * on all 3 levels of the visual styles), that colour property will start affecting the color settings for these new properties as well.

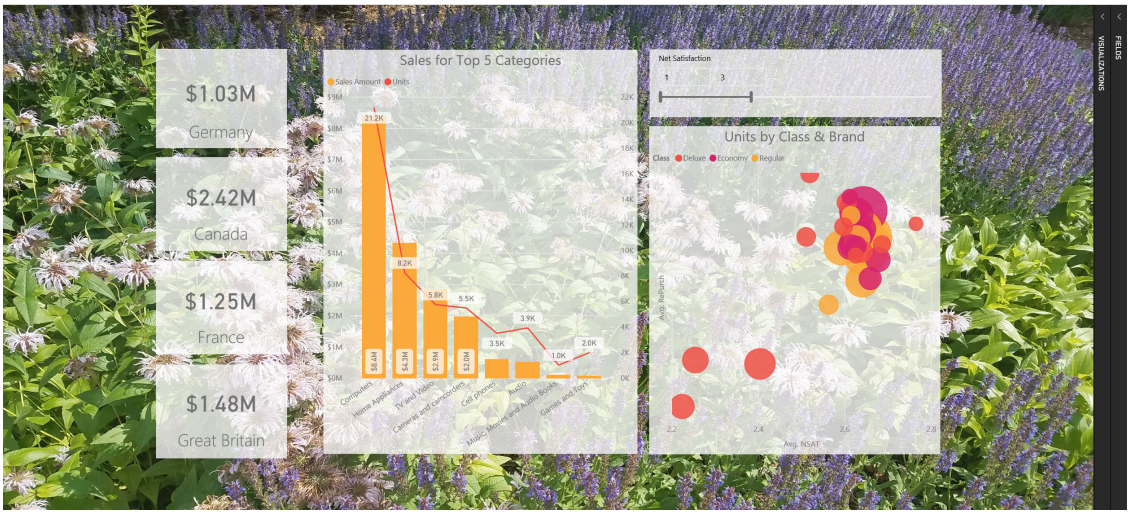
Wallpaper formatting

With our new wallpaper feature, this update lets you format that grey area outside your report page. This allows you to theme the entire report area and give it a more cohesive feel. The formatting options you have are similar to what you currently have for report pages, including colour,

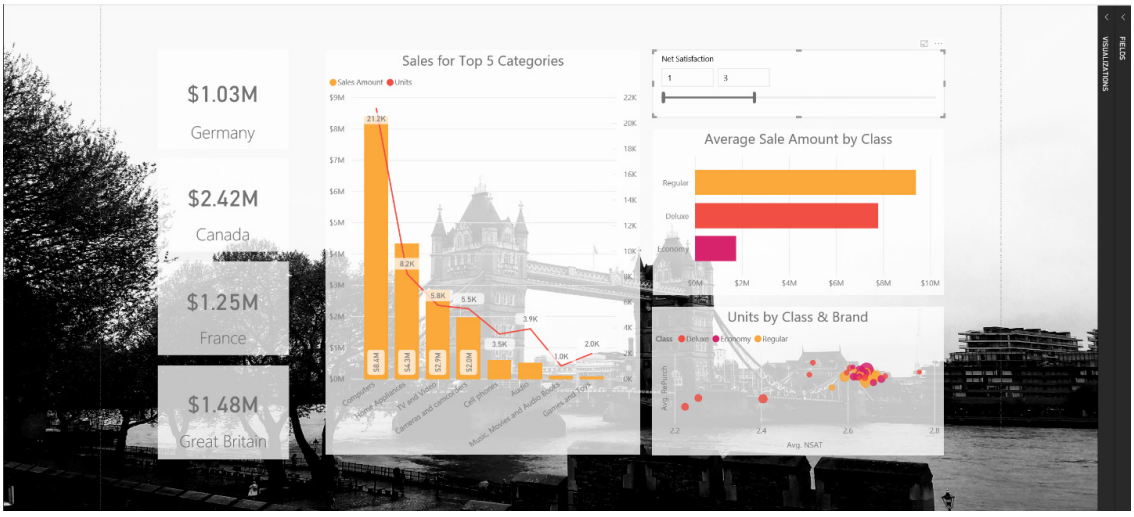
transparency, and uploading an image. You can set the wallpaper per page or use the theming preview to set a new default experience for all your report pages.

This feature can be used to extend the main colour as any background images you are using for layouts to the rest of the space.

You can also make your report page transparent and have your charts float on top of the wallpaper.



This can give your report a very different feeling depending on the image used.



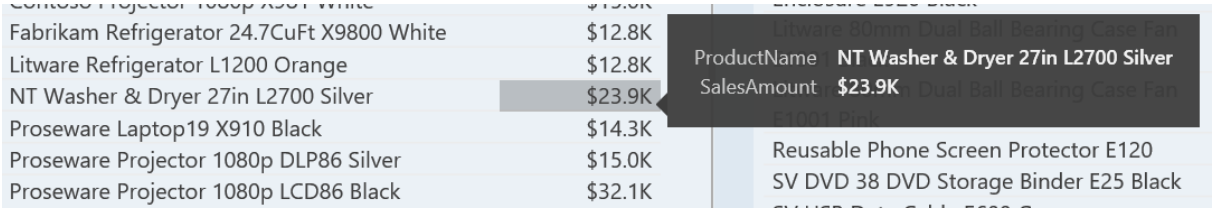
If the page background has a transparency higher that 50%, the software will add a grey dotted line in editing mode only to help you see where the edge of your page is.

As part of this update, the default report page has been modified to provide a completely transparent page background and a white wallpaper.

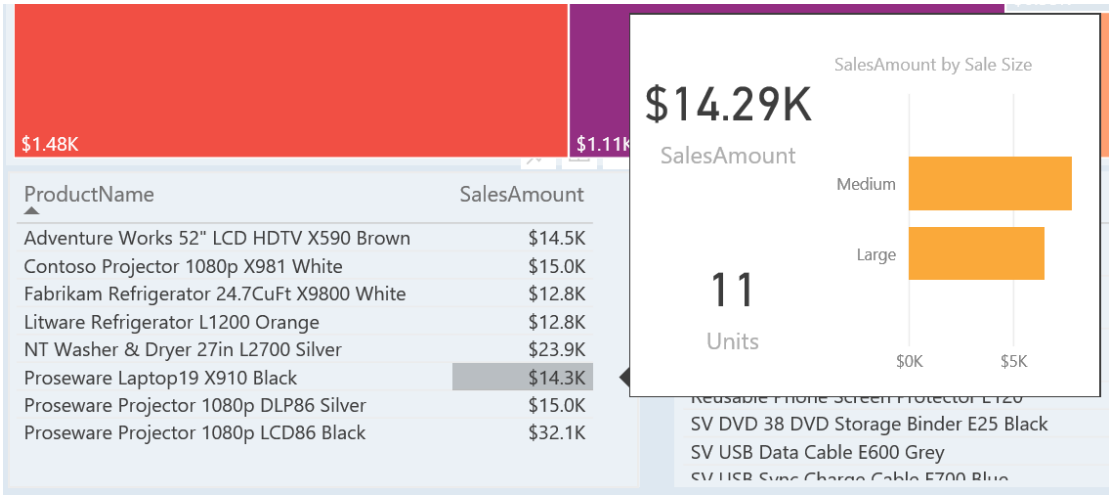
Tooltips for table and matrix

After releasing the report page tooltips preview back in March, one of the most common requests Microsoft received was to have them on the table and matrix visual too. The main reason this wasn't supported from the start was that the table and matrix visual didn't support the standard tooltip that you see on all the other visuals. This has now been rectified.

The table and matrix visuals will automatically light up with the standard tooltips without you needing to do any extra to your report.



If you are using the automatic linking for report page tooltips, those will automatically start working as well. You can also start manually assigning your report page tooltips as well through the 'Tooltips' card.



Turn tooltips off for visuals

It occurred to Microsoft that whilst they were making updates to tooltips to support table and matrix, it might be a good idea to add the same on / off toggle for tooltips for all visuals that support tooltips. If you go to the same tooltip card you use to control report page tooltips, you'll see a toggle has been added to the card. If you turn the toggle off, you'll receive no tooltip at all when hovering over the visual. Simple and useful.

Slicer accessibility

Being able to interact with and change Slicer values is one of the most important tasks for any report consumer, and for a while this wasn't available to users who relied solely on a keyboard to navigate. However, keyboard and screen reader users may now go beyond just consuming a report as is and can start interacting with Slicers on their reports.

When you want to change the current state of a Slicer using just a keyboard, the first thing you'll need to do is enter into the slicer. You can do this with the **CTRL + Right Arrow** keyboard shortcut. This will move your focus to the first element inside the slicer. You can then tab around the various components of the slicer, such as the clear icon, dropdowns and text fields to update values, and the slicer type dropdown. As your focus moves, helpful information is also read out by the screen reader, if you have one on.

This is a big step forward in terms of accessibility - one of several planned over the next few months apparently.

Formatting pane improvements

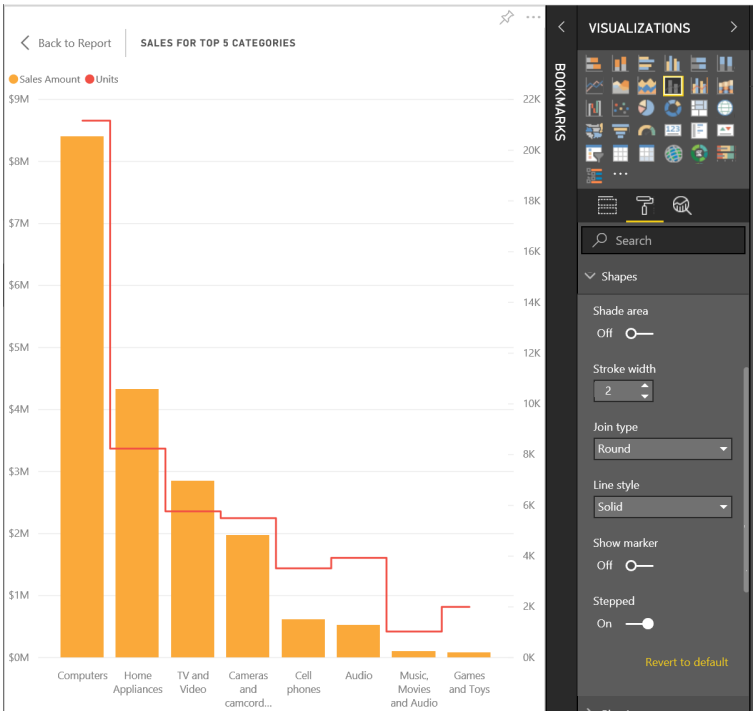
If you are a report author, there's a good chance you probably spend a lot of time in the 'Formatting' pane updating the design of your visuals. Initially, this was quite basic, but this update sees several new features.

Indeed, the updates include:

- a lighter grey color used inside cards, so the entire pane doesn't blend together
- animations and hover effects to give the pane a more modern / polished feel
- titles show above a given control to improve readability and reduce truncation.

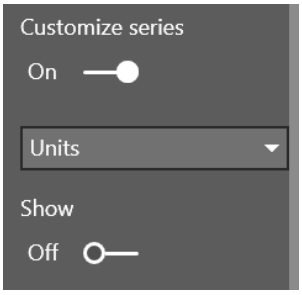
Stepped line support in line and combo charts

As a small formatting improvement this month, this update supports changing the line in combo charts to a stepped format. You'll find this toggle in the 'Shapes' card.

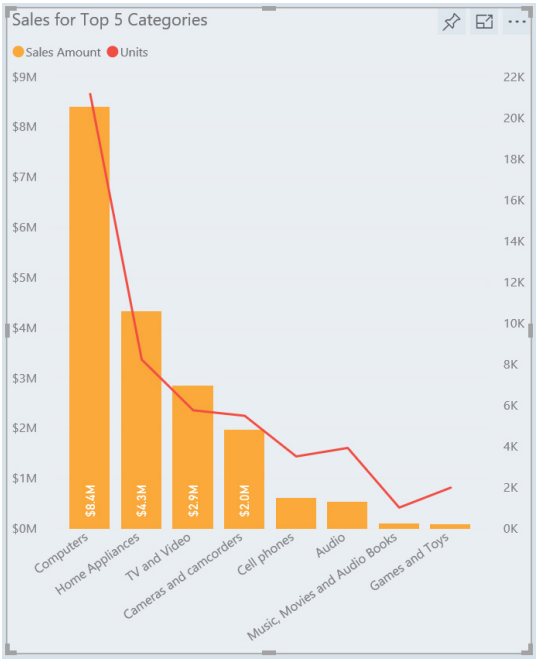


Turn off combo chart data labels for individual series

Last month's update provided the ability to customise the line and column data labels separately for combo charts. This month, you may now turn the data labels off per series as well.

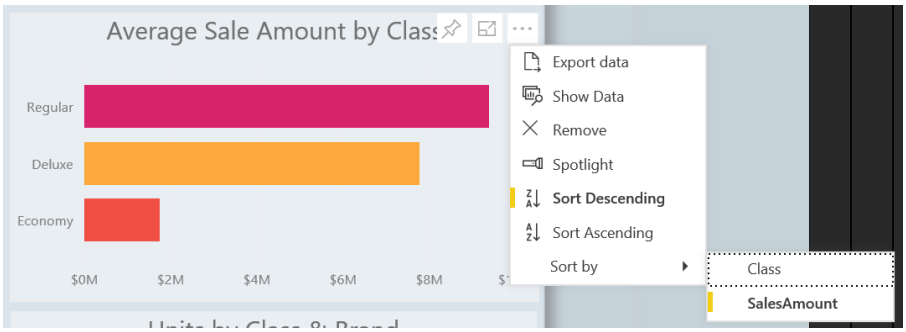


When you turn the 'Show' toggle off, that series' data labels will turn off in the combo chart.



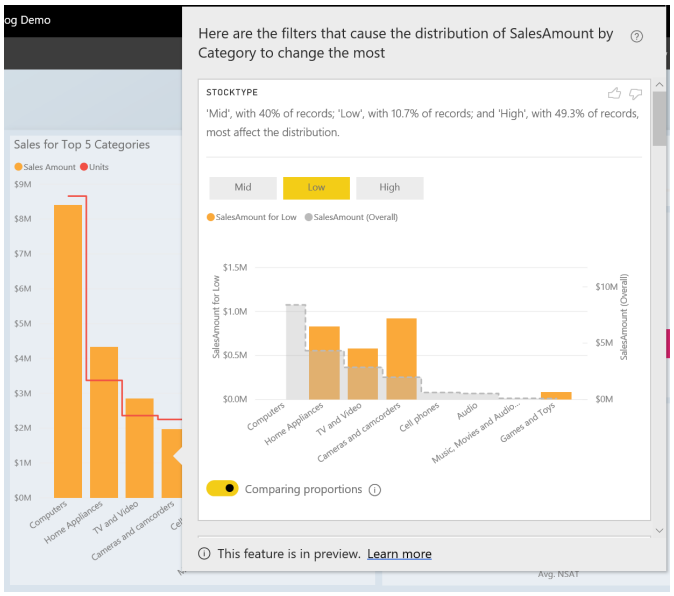
Sorting experience improvement

It's not unfair criticism that the sort experience wasn't very clear or easy to use and wasn't accessible to keyboard users. For this update, Microsoft has improved this by separating the controls to change sort direction from the control to pick the field to sort by. With this change, you can now also fully navigate this options flyout with just the keyboard.



Distribution factor insights

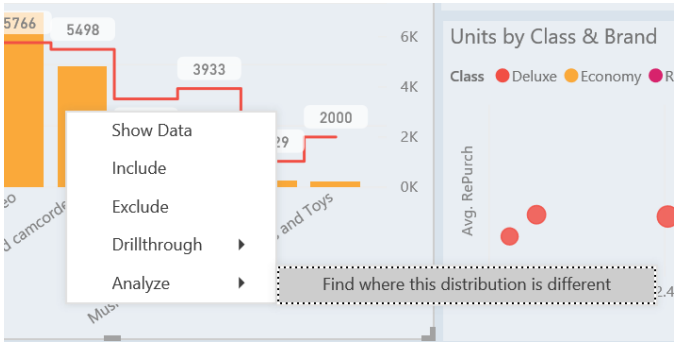
Microsoft has provided a new type of insight that you can find through the 'Insights' feature (if you are lucky enough to have it!), which finds different filters that create significant different distributions compared with the original graph. You'll notice in the insights there is a stepped line to represent the original chart, and the bars of the combo chart to represent the new distribution.



You can switch between different categories of the filter using the slicer above the visual.

In addition, for additive measures, you can use the toggle below the chart to switch between viewing the graph as proportions, using a dual axis, or as absolute values, using a shared axis for both values.

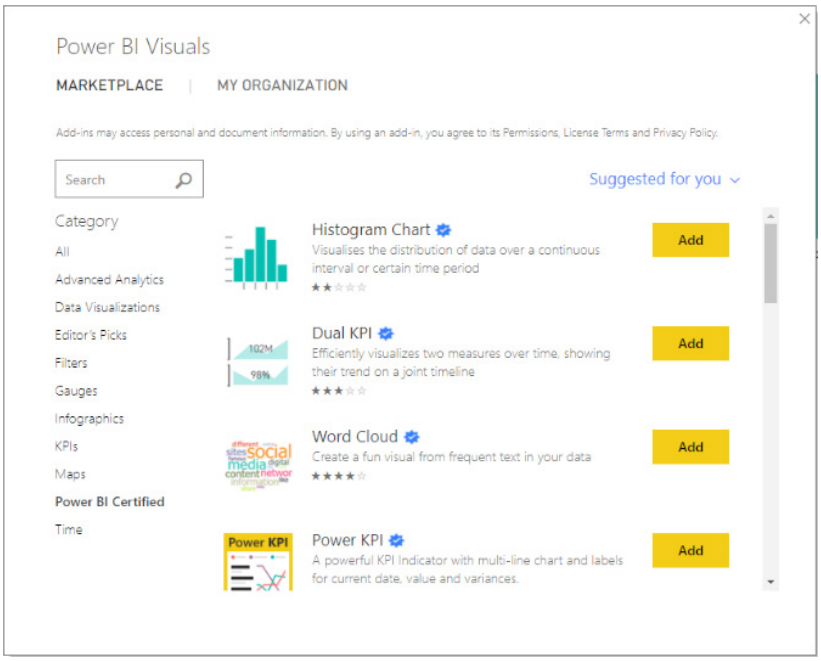
As with other insights, you can find insights through the 'Analyze' option after right-clicking on a data point. This option is called 'Find where this distribution is different'.



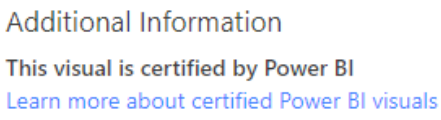
Power BI certified category

It's noCertified custom visuals are visuals in the marketplace that meet code requirements and additional code-check by the Power BI team. To make it easier to find these certified custom visuals, Microsoft updated the Power BI custom visual store to better tag and filter to locate these "certified" visuals.

Once you open Power BI store, you'll see the new category: "Power BI Certified":

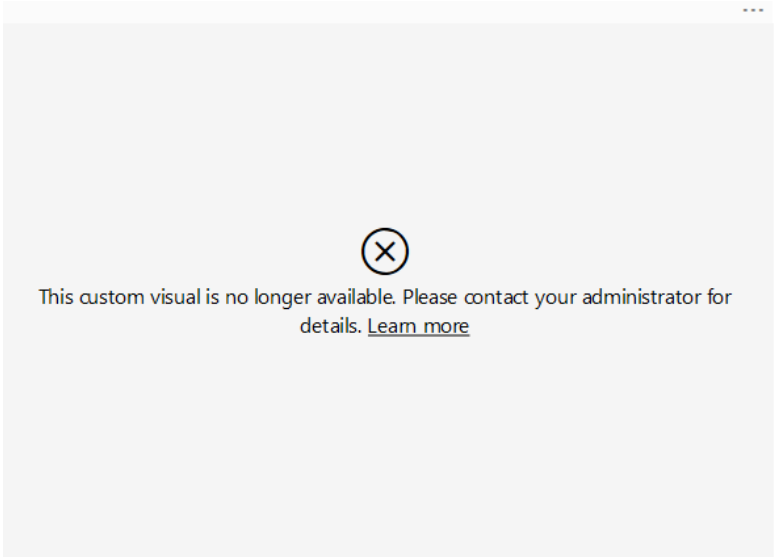


Once you select a certified visual, you'll see a message in the 'Additional Information' section letting you know it's certified:



Disabling specific organizational visuals

Back in April, Microsoft announced that organizational *(sic)* visuals, which are a simple way to deploy and manage custom visuals, were Generally Available. Now, Power BI administrators can disable specific organizational visuals. Once the administrator disables an organizational visual, it will immediately stop showing in existing reports and will display a message letting users know their administrator disabled it. The admin's change will take effect when users restart Power BI Desktop or refresh their browser when using the Power BI service.



Censored!!

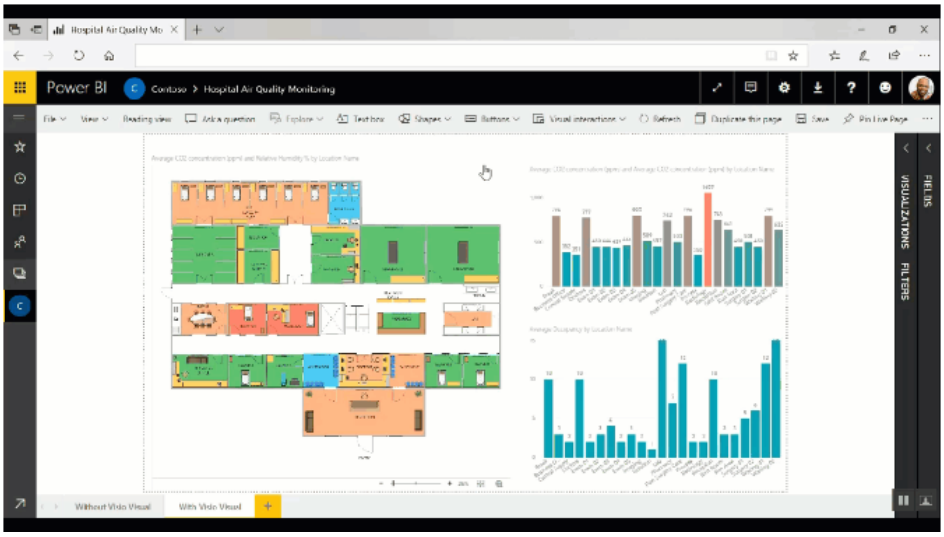
Bookmarks that were saved with the disabled visual will still be valid.

Visio custom visual gallery Generally Available

The Visio custom visual lets you combine the power of Visio with Power BI, allowing you to illustrate and compare data as both diagrams and traditional Power BI visualisations in one place. This integration gives users the ability to combine business and operational intelligence in a single dashboard for unified data perspectives. The situations for this are nearly endless, from building IoT-connected (Internet of Things connected) processes in manufacturing to visualising store inventory in retail. Microsoft launched a public Preview of this custom visual back in August and this update now makes it Generally Available. This will mark the end of Public Preview.

Some updates to the Visio visual since the public preview launch include:

- Support for Power BI Mobile apps
- Change the embedded Visio diagram with another one
- Copy the link of the embedded Visio diagram
- Turn the auto-zoom settings on and off for cross-highlighting and cross-filtering
- Use a #layer to support complex diagrams and improve the performance of the Visio visual in your reports.

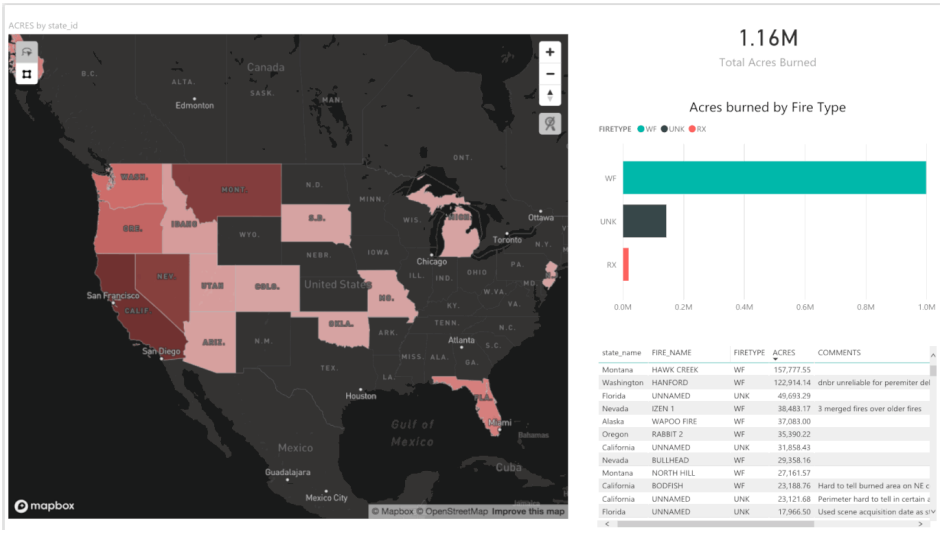


Mapbox custom visual Generally Available

Mapbox provides powerful mapping and location tools for Business Intelligence (BI). Their goal is to craft creative maps and developer-friendly location data, APIs, and software development kits so that you're free to focus on designing, building, and developing your application. Back in March, they released a preview custom visual, so you could take advantage of their powerful mapping capabilities right in your Power BI reports. This month, they are extending their feature set and making the visual Generally Available for everyone to use.

Some of the new features included in this update are:

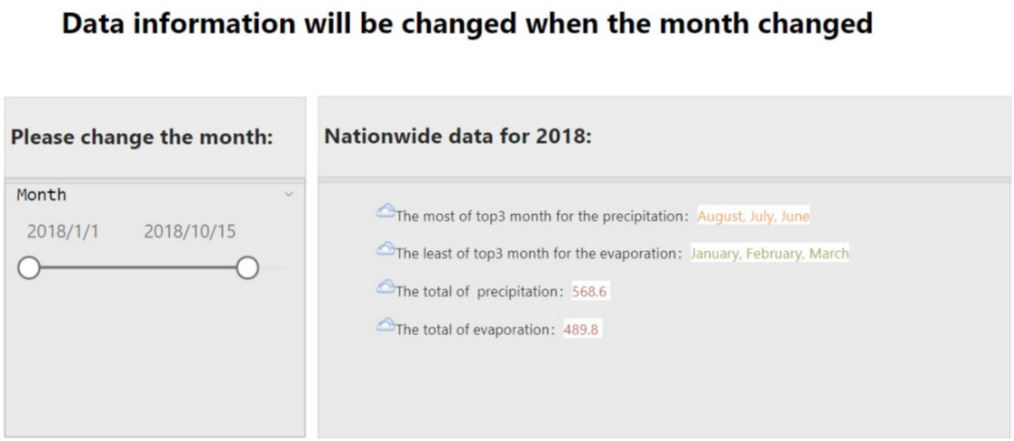
- Choropleth / Polygon maps – The visual has out of the box support for Countries / Regions, US States, and US postal codes. You also can add custom polygons as well (maybe people will remember there are other places on Earth at some point…)
- Cross-highlighting and filtering – In both the choropleth and circle layouts, you can use the Mapbox visual to filter or highlight other visuals on your page
- Support for Drilling – When using Choropleth maps, you can use Power BI's drilling capabilities to move between levels of your hierarchy
- Lasso Select – You can use either a freeform lasso or a polygon selection mode to explore a subset of your data, and drive filters for other visuals on your report
- Support for IE11, Edge, and Safari.



DataText Box custom visual

The DataText Box custom visual by Sharewinfo is a dynamic textbox that supports all kind of different configurations. With this custom visual, you can:

- Create dynamic content by combining static text with dynamic values from your data
- Customize the layout of your text to include tables, paragraphs, pictures, and videos
- Automatically wrap lines of text.



China Scatter Map

The China Scatter Map by Sharewinfo is a bubble map geared towards the Chinese market. With this map, you can display China location data with the size and color of the datapoint based off fields in your model. The map is apparently "very configurable" (!) and lets you switch to specific provincial maps.

IBM DB2 DirectQuery Connector (Preview)

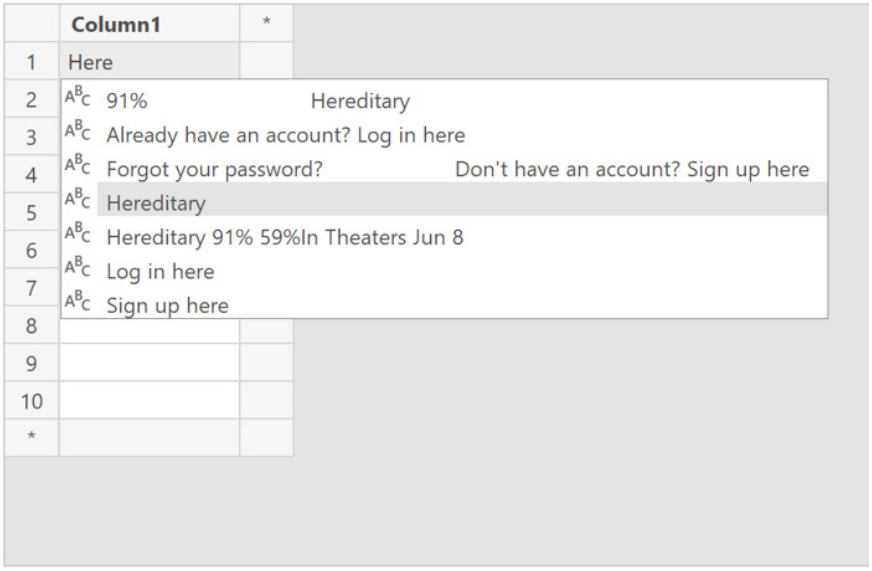
In this month's release, DirectQuery support has been added to the IBM DB2 connector. This feature is currently available as a Preview feature only, so you will need to enable it from the 'Options' dialog first. After enabling this Preview feature, you'll be able to find the DirectQuery option within the IBM DB2 connector dialog.

Improvements to Web By Example Connector

Two months ago, the new Web By Example connector was introduced, allowing any HTML data, not just tables, to be extracted. To do so, you had to provide a few samples for the data that you want to extract, and Power Query will apply smart detection algorithms on top of the HTML page content to identify the entire set of rows to generate.

This month Microsoft has further enhanced this connector in multiple ways:

- **Support for importing multiple custom tables:** it is now possible to enter the 'Extract table using examples' dialog from the 'Navigator' dialog multiple times, allowing you to create multiple custom tables in one pass and selecting one or more custom and out-of-the-box tables from the Navigator list
- **Automatic completions for specifying sample values:** similar to the 'Add Column From Examples' experience (and that's one of Power Query's best!), you now get automatic completions as you type sample values, making it easier for you to provide samples and discover what data values are recognised



- **Exposure of attribute selectors in Web.BrowserContents function:** additionally, you can now import data from attribute selectors by using the Web. BrowserContents function.

SAP HANA – Default values in Variables experience

The SAP HANA Variables experience in the 'Navigator and Edit Variable's dialog has been enhanced, by leveraging SAP HANA's variable metadata regarding default values to auto-populate the variable input controls.

That's it for those updates for this month. Think we need to sit down and get our breath back. But there's more…

Latest Updates for Power BI Service and Mobile

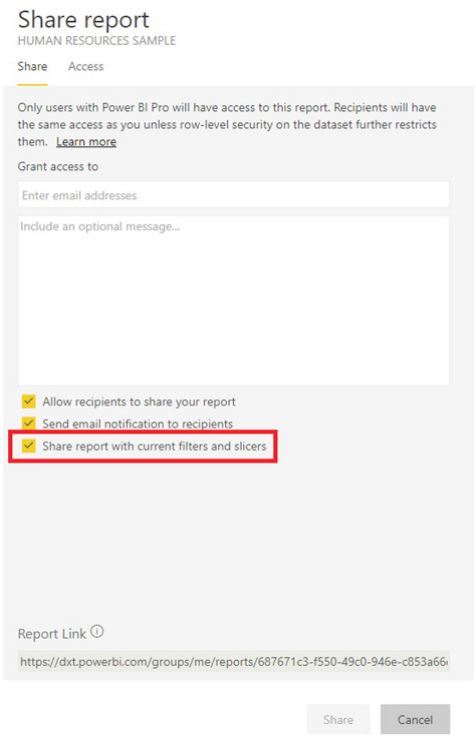
For once, the announcements for Power BI Service and Mobile beat Desktop updates. Perhaps Microsoft has been taking our criticism to heart (yeah, right). Here's this month's list:

- Sharing reports with filters and slicers
- Setting up datasets in Power BI
- Update for On-premises data gateway
- Updated report canvas on Mobile.

Let's take a look at each of these in turn.

Sharing reports with filters and slicers

Earlier this year, Microsoft made it simpler to collaborate with teams and partners by announcing report sharing in the Power BI Service. Now, this has been improved by allowing you to share reports with filters and slicers applied. In other words, you no longer have to take a screenshot or explain the steps you took to configure the report to your desired view; all you have to do is make sure "Share with current filters and slicers" checkbox is selected in the Share pane before you share.



You should note that you will only see this option if you have made changes to the published filters or slicers on the report.

Once your report has been shared successfully, your recipient will receive an email that contains an URL to the report. Clicking on the URL will take them to the report and automatically apply all the filters and slicers that were shared by you. The recipient can then bookmark this URL in the browser to retain your filter and / or slicer context for future use. It's that simple.

One other thing to note though: this feature is currently disabled for reports with custom visuals. Microsoft state this will be addressed very shortly.

Setting up datasets in Power BI

The "user experience" for gateway connections in Power BI has been simplified to set up datasets. The new interface guides you through multiple configurations that best fit with your solution.

Update for On-premises data gateway

The Power BI On-premises data gateway has again been updated. The latest version matches the recent update for Desktop and ensures compatibility with the latest updates heading for Desktop.

Updated report canvas on Mobile

The Power BI report canvas on Mobile has been made larger. Microsoft has updated the screen size to be larger, giving you more real estate for visuals on the report. In addition, the top and bottom action bars are hidden after report load, allowing users to view more of the report at once.

Short and sweet this month – more in September, no doubt!

The A to Z of Excel Functions: COUNT

Need a function you can **COUNT** on? Dracula couldn't.

This function counts the number of cells that contain numbers and counts numbers within the list of arguments. You may use the **COUNT** function to get the number of entries in a number field that is in a range or array of numbers. For example, you can enter the following formula to count the numbers in the range **A1:A20**:

=COUNT(A1:A20).

In this example, if five of the cells in the range contain numbers, the result would be 5.

The **COUNT** function employs the following syntax to operate:

COUNT(value1, [value2], ...)

The **COUNT** function has the following arguments:

- **value1:** this is required and represents the first item, cell reference or range within which you want to count numbers
- **value2:** second and subsequent arguments are optional. Up to 255 additional items, cell references or ranges may be specified to count numbers.

It should be further noted that:

- arguments that are numbers, dates or a text representation of numbers (*e.g.* a number enclosed in quotation marks, such as "1") are counted
- logical values and text representations of numbers that you type directly into the list of arguments are counted
- arguments that are error values or text that cannot be translated into numbers are not counted
- if an argument is an array or reference, only numbers in that array or reference are counted. Empty cells, logical values, text or error values in the array or reference are not counted
- if you want to count logical values, text or error values, use the **COUNTA** function instead
- if you want to count only numbers that meet certain criteria, use the **COUNTIF** or **COUNTIFS** functions depending upon how many criteria are required.

Please see our example below:

	A	B	C
1	Data		
2	28/07/2017		
3	8		
4	69.01		
5	0		
6	TRUE		
7	#DIV/0!		
8			
9			
10	Formula	Description	Result
11	=COUNT(A2:A7)	Counts the number of cells that contain numbers in cells A2:A7.	4
12	=COUNT(A6:A7)	Counts the number of cells that contain numbers in cells A6:A7.	0
13	=COUNT(A2,A4,A6,17)	Counts the number of cells that contain numbers in cells A2, A4 and A6 plus the number 17.	3
14			

The A to Z of Excel Functions: COUNTA

This is neither the Italian COUNT function nor an Excel calculator dedicated to tallying the number of times the letter "a" occurs in a spreadsheet. This function actually counts the number of cells that are not empty in a range. I suppose *COUNTNOTBLANK* is too clunky – but it's not quite right either *(see below)*.

The **COUNTA** function employs the following syntax to operate:

COUNTA(value1, [value2], ...)

The **COUNTA** function has the following arguments:

- **value1:** this is required and represents the first item, cell reference or range within which you want to count numbers
- **value2:** second and subsequent arguments are optional. Up to 255 additional items, cell references or ranges may be specified to count numbers.

It should be further noted that:

- The **COUNTA** function counts cells containing any type of information, including error values and empty text (""). For example, if the range contains a formula that returns an empty string, the **COUNTA** function counts that value. This is different behaviour to the **COUNTBLANK** function which deems empty text to be blank, *i.e.* **COUNTA(range) + COUNTBLANK(range)** does not necessarily equal the number of cells in the **range**, which many modellers assume to be an identity

- The **COUNTA** function does not count empty cells
- If you do not need to count logical values, text or error values (in other words, if you want to count only cells that contain numbers), use the **COUNT** function
- If you want to count only cells that meet certain criteria, use the **COUNTIF** or **COUNTIFS** functions depending upon how many criteria are required.

Please see the example below:

	A	B	C
1	Data	Empty Text	
2	28/07/2017		=""
3	8		
4	69.01		
5			
6	TRUE		
7	#DIV/0!		
8			
9			

	Formula	Description	Result
10	Formula	Description	Result
11	=COUNTA(A2:A7)	Counts the number of non-blank cells in the range A2:A7.	5
12	=COUNTA(A6:A7)	Counts the number of non-blank cells in the range A6:A7.	2
13	=COUNTA(B2)	Demonstrates that empty text is counted as non-blank.	1
14	=COUNTBLANK(B2)	Demonstrates that empty text is also deemed to be blank.	1
15			

The A to Z of Excel Functions: COUNTBLANK

This function counts empty cells in a specified range of cells most of the time. It does lie occasionally…

The **COUNTBLANK** function employs the following syntax to operate:

COUNTBLANK(range)

The **COUNTBLANK** function has the following argument:

- **Range:** this is required and represents the **range** from which you want to count the blank cells.

It should be further noted that:

- cells with formulas that return "" (empty text) are also counted. This is different behaviour to both the **COUNTA** and **ISBLANK** functions which deem empty text to be *non-blank*, *i.e.* **COUNTA(range) + COUNTBLANK(range)** does not necessarily equal the number of cells in the **range**, which many modellers assume to be an identity
- cells with zero values are not counted.

Please see the next example below:

	A	B	C
1	Data	Empty Text	
2	28/07/2017		=""
3	8		
4	0		
5			
6	TRUE		
7	#DIV/0!		
8			
9			

	Formula	Description	Result
10	Formula	Description	Result
11	=COUNTBLANK(A2:A7)	Counts the number of blank cells in the range A2:A7.	1
12	=COUNTBLANK(A6:A7)	Counts the number of blank cells in the range A6:A7.	0
13	=COUNTA(B2)	Demonstrates that empty text is counted as non-blank.	1
14	=COUNTBLANK(B2)	Demonstrates that empty text is also deemed to be blank.	1
15	=ISBLANK(B2)	Demonstrates that empty text is also deemed to be blank.	FALSE
16			

The A to Z of Excel Functions: COUNTIF

This function counts the number of cells that meet a particular criterion; for example, to count the number of times a particular city appears in a customer contact list.

The **COUNTIF** function employs the following syntax to operate:

COUNTIF(range,criterion)

The **COUNTIF** function has the following arguments:

- **range:** this is required and represents the **range** from which you want to count the cells meeting the specified **criterion**
- **criterion:** the condition to be met.

It should be further noted that:

- **COUNTIF** ignores upper and lower case in text strings. **Criterion** is not case sensitive, so "red" and "RED" will match the same cells
- wildcard characters are permitted. The characters, question mark (?) and asterisk (*) can be used in **criterion**. The question mark matches any single character, whereas an asterisk matches any sequence of characters. If you actually want to find a question mark or asterisk use the tilde (~) in front of the required character
- COUNTIF is pedantically precise. Therefore, make sure your data does not contain erroneous characters. In particular, when counting text values, make sure that text doesn't contain leading, excess or trailing spaces, inconsistent use of straight / curly quotation marks or non-printing characters. The **CLEAN** removes all non-printable characters and **TRIM** (a useful text manipulating function that removes excess spaces) functions can eradicate most of these issues
- range names may be used with **COUNTIF** if required
- **range** can be a range in the same worksheet, a different worksheet or even a range in another workbook. However, if you refer to a second or subsequent workbook, these workbooks must be open for **COUNTIF** to work as intended, otherwise *#VALUE!* will be returned
- the wrong value may be returned for long strings. The **COUNTIF** function returns incorrect results when you try to use it to match strings longer than 255 characters. However, there is a workaround available. You may use the **CONCATENATE** function or the concatenate (**&**) operator, *e.g.* **=COUNTIF(A1:A7,"long string"&"another long string")**
- if no value is returned where one is expected, check to see whether the **criterion** argument should be in quotation marks, *e.g.* ">=14".

Please see the example below:

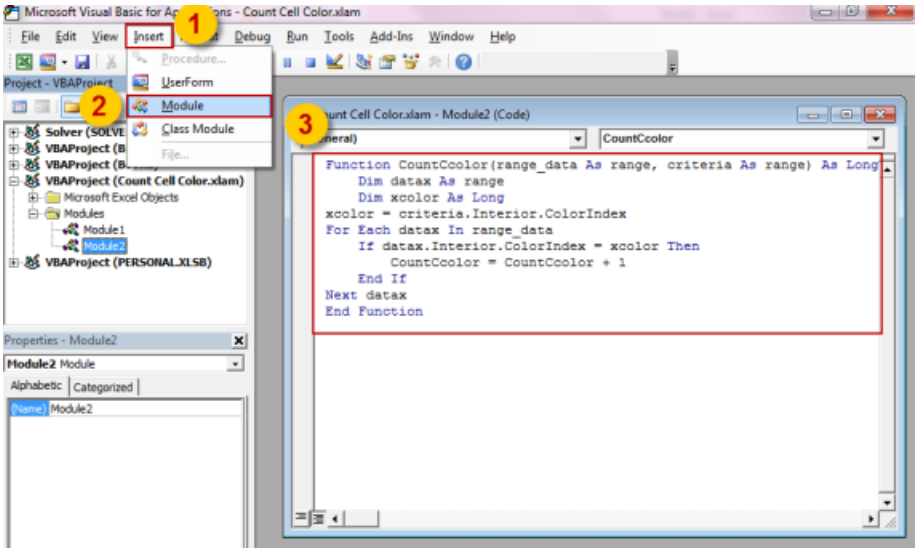
	A	B	C
1	Data	More data	
2	red	1	
3	amber	2	
4	green	4	
5	RED	8	
6	amber	16	
7	AMBER	32	
8			
9			

10	Formula	Description	Result
11	=COUNTIF(A2:A7,"red")	Counts the number of cells with "red" (not case sensitive) in cells A2:A7.	2
12	=COUNTIF(A2:A7,A4)	Counts the number of cells with "green" (the value in cell A4, not case sensitive) in cells A2:A7.	1
13	=COUNTIF(A2:A7,A2)+COUNTIF(A2:A7,A3)	Counts the number of occurrences of "red" (the value in cell A2) and amber (the value in cell A3) in cells A2:A7. You could also use the COUNTIFS function.	5
14	=COUNTIF(B2:B7,">9")	Counts the number of cells in the range B2:B7 with a value greater than 9.	2
15	=COUNTIF(B2:B7,"<>"&B4)	Counts the number of cells in the range B2:B7 with a value not equal to that in cell B4 (4). The ampersand (&) merges the comparison operator for not equal to (<>) and the value in cell B4. This reduces the formula to =COUNTIF(B2:B7,"<>4"), but makes it dynamic.	5
16	=COUNTIF(B2:B7,"<"&B7)-COUNTIF(B2:B7,"	Counts the number of cells in the range B2:B7 with a value greater than 4 but less than 32 (the value in cell B7).	2
17	=COUNTIF(A2:A7,"*")	Counts the number of cells in A2:B7 containing any text (* is the wildcard character for text).	6
18	=COUNTIF(A2:A7,"??e??")	Counts the number of cells in the range A2:A7 that have exactly five characters, but have the letter "e" in the third position of the text string ("green" is the only text that satisfies this condition).	1
19			

It should be further noted that the **COUNTIF** function will not count cells based on cell background or font colour. However, Excel supports User-Defined Functions (UDFs) using Microsoft Visual Basic for Applications (VBA). For completeness, here is an example of how you can count the number of cells with specific cell colour using VBA.

1. Open Microsoft Excel then press **ALT + F11** to show Visual Basic Editor window
2. On the 'Insert' menu, select 'Module' to create a module. Then write the following script:

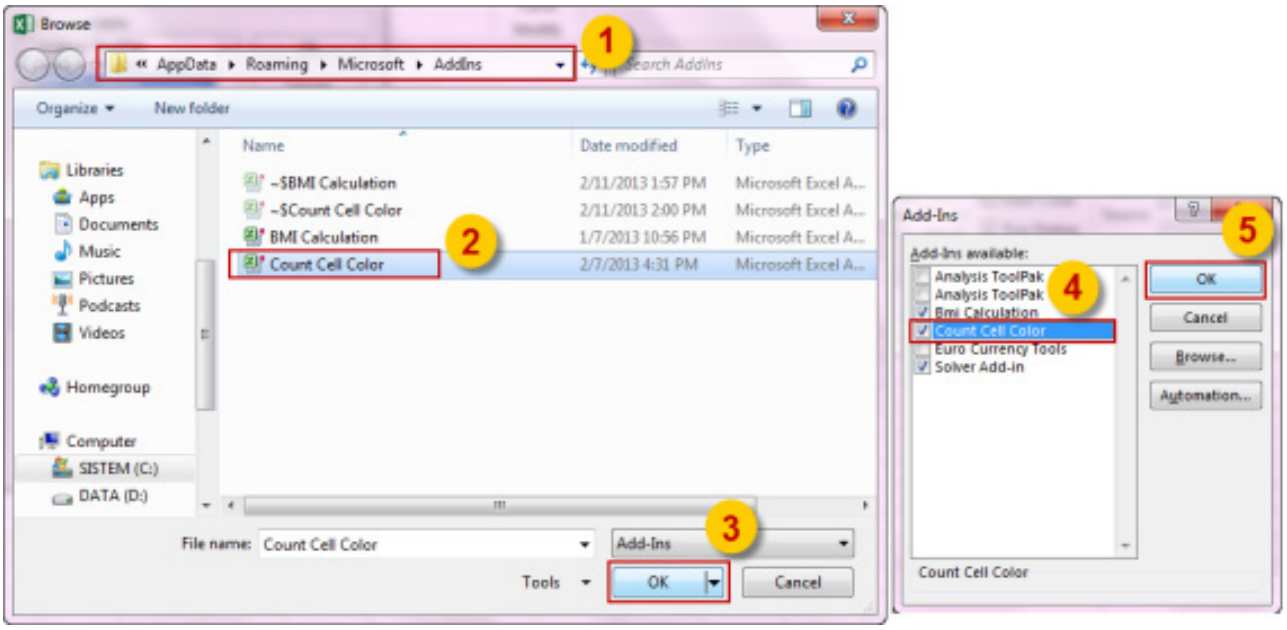
```
Function CountCcolor(range_data As range, criteria As range) As Long
    Dim datax As range
    Dim xcolor As Long
xcolor = criteria.Interior.ColorIndex
For Each datax In range_data
    If datax.Interior.ColorIndex = xcolor Then
        CountCcolor = CountCcolor + 1
    End If
Next datax
End Function
```



3. Close VBE window and return to Excel
4. Test the UDF using example data *(see example below)*
5. In cell **D3**, write the function **=CountCcolor(range_data,criteria)** (note UDFs do not automatically capitalise like built-in Excel functions)
6. In the **range_data** argument, select cells **C2:C31**
7. in **criteria** argument, select cell **F1**
8. Press **ENTER**. In cell **F2** the result is 6. It means the number of cells with this colour is six

	A	B	C	D	E	F	G	H	I	J
1	No	Name	Colour		Colour Criterion:					
2	1	Andrea			Result:	6		=CountCcolor(C2:C31,F1)		
3	2	Anna								
4	3	Bernadette								
5	4	Bill								
6	5	Cecile								
7	6	Charlene								
8	7	Debra								
9	8	Ingeborg								
10	9	Jackie								
11	10	Jak								
12	11	Jan Karel								
13	12	Jonathan								
14	13	Jonathan								
15	14	Kathryn								
16	15	Ken								
17	16	Lana								
18	17	Layla								
19	18	Liam								
20	19	Marc								
21	20	Marcus								
22	21	Michelle								
23	22	Miron								
24	23	Mynda								
25	24	Nancy								
26	25	Roger								
27	26	Stefenie								
28	27	Tim								
29	28	Tina								
30	29	Tony								
31	30	Zack								
32										

9. You can also test for another colour. Change the colour in cell **F1** as required
10. If you save this example as a workbook, say **Count Cell Color** (yes, that's right, I appear to be advocating US spellings!), save it as an **Excel Add-In (.xlam)** format
11. This way this UDF may be used in other files. Open Excel on the computer that you want to install the Add-In. Then, open the 'Add-Ins' dialog box by clicking 'Add-In' on the 'Developer' tab
12. In the 'Add-Ins' dialog box, click the 'Browse…' button so that the 'Browse' dialog box is displayed



13. Go to file location that add-In file is saved. Choose the file and then click 'Open'
14. On the 'Add-Ins' dialog box make sure that the add-in checkbox is checked. Then click 'OK'
15. The Count Cell Color UDF has now been installed and is ready to use.

The A to Z of Excel Functions: COUNTIFS

Our last function of the month (and the **COUNT** family) applies one or more criteria to cells across multiple ranges and counts the number of times all criteria are met. This is essentially the "multiple" version of **COUNTIF**.

The **COUNTIFS** function employs the following syntax to operate:

COUNTIFS(criteria_range1, criteria1, [criteria_range2, criteria2]…)

The **COUNTIFS** function has the following arguments:

- **criteria_range1:** this is required and represents the first range in which to evaluate the associated criteria
- **criteria1:** this is also required. The criteria must be in the form of a number, expression, cell reference, or text that define which cells will be counted. For example, criteria can be expressed as 32, ">32", **B4**, "apples", or "32"
- **criteria_range2, criteria2, …:** these arguments are optional but must appear in associated pairs. Up to 127 range / criteria pairs are allowed.

It should be further noted that:

- **COUNTIFS** ignores upper and lower case in text strings. **Criteria** are not case sensitive, so "red" and "RED" will match the same cells
- each range's criteria are applied one cell at a time. If all of the first cells meet their associated criteria, the count increases by 1. If all of the second cells meet their associated criteria, the count increases by 1 again, and so on until all of the cells are evaluated
- if the criteria argument is a reference to an empty cell, the **COUNTIFS** function treats the empty cell as a zero value
- wildcard characters are permitted. The characters, question mark (?) and asterisk (*) can be used in **criterion**. The question mark matches any single character, whereas an asterisk matches any sequence of characters. If you actually want to find a question mark or asterisk use the tilde (~) in front of the required character.

Please see our final example below:

	A	B	C	D
1	Date	Salesperson	Items Sold	Quota Met
2	1 Jan 20	Jo	4	Yes
3	2 Jan 20	Alex	2	No
4	3 Jan 20	Alex	6	Yes
5	4 Jan 20	Lou	1	No
6	5 Jan 20	Alex	5	Yes
7	6 Jan 20	Jo	3	No
8	7 Jan 20	Lou	6	Yes
9	8 Jan 20	Alex	7	Yes
10				
11				

	Formula	Description	Result
13	=COUNTIFS(C2:C9,">1",C2:C9,"<6")	Counts the number of sales where the items sold (cells **C2:C9**) were more than 1 but less than 6.	4
14	=COUNTIFS(A2:A9,"<="&A8,B2:B9,B3,D2:D9,"Yes")	Counts the number of sales made in the first week of the year (cells **A2:A9**) by Alex (cells **B2:B9**, with "Alex" as shown in cell **B3**) where the quota (cells **D2:D9**) was met ("Yes").	2
15	=COUNTIFS(B2:B9,B2,D2:D9,"No")+COUNTIFS(B2:B9,"Lou",D2:D9,D3)	Counts the number of sales Jo and Lou (cells B2:B9 identifying the salesperson both ways) made where quotas (cells D2:D9) were not met (referwncing "No" both ways).	2
16	=COUNTIFS(B2:B9,"*L*",C2:C9,">3")	Counts the number of sales where the salesperson (cells **B2:B9**) had the letter "L" (not case sensitive) in their name and the number of items sold (cells **C2:C9**) exceeded 3.	4
17			

Darwin	Power Pivot, Power Query and Power BI	12 Nov 2018	1 day
Sydney	Power Pivot, Power Query and Power BI	12 - 14 Nov 2018	3 day
Melbourne	Excel Tips and Tricks	19 - 21Nov 2018	3 day
Sydney	Excel Tips and Tricks	10 Dec 2018	1 day
Sydney	Financial Modelling	11 - 12 Dec 2018	2 day
Sydney	Power Pivot, Power Query and Power BI	17 - 19 Dec 2018	3 day

Key Strokes

Each newsletter, we'd like to introduce you to useful keystrokes you may or may not be aware of. This month, it's time to combine **CONTROL** with **ALT**:

Keystroke	What it does
CTRL + ALT + F1	New Macro sheet
CTRL + ALT + F2	Open
CTRL + ALT + F3	New Name
CTRL + ALT + F4	Close application
CTRL + ALT + F5	Refresh all
CTRL + ALT + F9	Initiate full recalculation
CTRL + ALT + F12	Open Thai dictionary (!)
CTRL + ALT + TAB	Indent
CTRL + ALT + L	Reapply Sort / Filter

There are over 540 keyboard shortcuts in Excel. For a comprehensive list, please download our Excel file a www.sumproduct.com/thought/keyboard-shortcuts. Also, check out our new daily **Excel Tip of the Day** feature on the www.sumproduct.com homepage.

Our Services

We have undertaken a vast array of assignments over the years, including:

- **Business planning**
- **Building three-way integrated financial statement projections**
- **Independent expert reviews**
- **Key driver analysis**
- **Model reviews / audits for internal and external purposes**
- **M&A work**
- **Model scoping**
- **Power BI, Power Query & Power Pivot**
- **Project finance**
- **Real options analysis**
- **Refinancing / restructuring**
- **Strategic modelling**
- **Valuations**
- **Working capital management**

If you require modelling assistance of any kind, please do not hesitate to contact us at contact@sumproduct.com.

Link to Others

These newsletters are not intended to be closely guarded secrets. Please feel free to forward this newsletter to anyone you think might be interested in converting to "the SumProduct way".

If you have received a forwarded newsletter and would like to receive future editions automatically, please subscribe by completing our newsletter registration process found at the foot of any www.sumproduct.com web page.

Any Questions?

If you have any tips, comments or queries for future newsletters, we'd be delighted to hear from you. Please drop us a line at newsletter@sumproduct.com.

Training

SumProduct offers a wide range of training courses, aimed at finance professionals and budding Excel experts. Courses include Excel Tricks & Tips, Financial Modelling 101, Introduction to Forecasting and M&A Modelling.

Check out our more popular courses in our training brochure:

Drop us a line at training@sumproduct.com for a copy of the brochure or download it directly from http://www.sumproduct.com/training.

Sydney Address: SumProduct Pty Ltd, Suite 52, Level 10, 88 Pitt Street, Sydney, NSW 2000
New York Address: SumProduct Pty Ltd, 48 Wall Street, New York, NY, USA 10005
London Address: SumProduct Pty Ltd, Office 7, 3537 Ludgate Hill, London, EC4M 7JN, UK
Melbourne Address: SumProduct Pty Ltd, Level 9, 440 Collins Street, Melbourne, VIC 3000
Registered Address: SumProduct Pty Ltd, Level 6, 468 St Kilda Road, Melbourne, VIC 3004

contact@sumproduct.com
www.sumproduct.com
+61 3 9020 2071