

It's back to basics this month

as we spread much Excel cheer. It's all been very quiet for a while (readers may have noticed we've been concentrating on Power Query etc.) – but that all changes this month. Microsoft **SPILLS** the beans regarding what they have been doing. I know they're quite well paid there, but after these latest improvements, I think they may be asking for **arrays** (*groan – Ed.*).

But we still have the latest update news, our regular series on Power Query, VBA, Keyboard Shortcuts and the ever-continuing A to Z of Excel Functions.

Happy reading and see you next month.

Liam Bastick, Managing Director, SumProduct



Getting Arrays: Spilling the Beans on SEVEN New Excel Functions

This came in too late for October's newsletter, but boy, we make up for it in this one! September 24 is the day Excel moved on. Yes, we've had Power Pivot, Power Query / Get & Transform and Power BI, but Microsoft's "CALC" team has been busy behind the scenes rearranging the furniture.

By "furniture" I mean the "calculation engine" – it's had a complete re-write, and there are benefits general Excel users will reap for years to come. The first wave sees a new array calculation ("Dynamic Array"),

seven new functions and two new error messages. And that's just the start. There's going to be plenty more coming in the next few years.

It's all still in what Microsoft refers as "Preview" mode, *i.e.* it's not yet "Generally Available" but it is something you can try and hunt out. Presently, you need to be part of what is called the "Office Insider" programme which is an Office 365 fast track. You can register in **File -> Account -> Office Insider** in Excel's backstage area.

The screenshot shows the Microsoft Office Account page. On the left is a green navigation sidebar with options: Home, New, Open, Info, Save, Save As, Print, Share, Export, Publish, Close, Account, Feedback, and Options. The main content area is titled 'Account' and is divided into 'User Information' and 'Product Information'. Under 'User Information', there are links for 'Change photo', 'About me', 'Sign out', and 'Switch account'. Below that are dropdown menus for 'Office Background' (set to 'No Background') and 'Office Theme' (set to 'Colorful'). Under 'Connected Services', there is an 'Add a service' button. The 'Product Information' section shows the Office logo and 'Subscription Product: Microsoft Office 365'. Below this are 'Manage Account' and 'Change License' buttons. A yellow 'Office Updates' banner indicates that updates are available. The 'Office Insider' section is circled in red and shows that the user is signed up for the program, with a note that they will receive new builds of Office approximately once a week. Other sections include 'About Excel' and 'What's New'.

Even then, you're not guaranteed a ticket to the ball as only some will receive the new features as Microsoft slowly roll out these features and functions. Please don't let that put you off. These features will be with all Office 365 subscribers soon.

Let me be clear. When Office 2019 comes out, it will not include Dynamic Array functions. It's likely you will have to wait until Office 2022

(assuming such a thing will exist) as Microsoft tries to convert everyone to the annual subscription model.

So what's the big deal?

Let me begin by just looking at what a Dynamic Array is. Consider the following data:

	C	D	E	F	G	H
9						
10				Original Data		
11						
12				Shape	Colour	Sides
13				Triangle	Red	3
14				Rectangle	Amber	4
15				Circle	Green	1
16				Triangle	Red	3
17				Square	Blue	4
18				Rectangle	Blue	4
19				Rectangle	Amber	4
20				Circle	Amber	1
21				Triangle	Red	3
22				Square	Green	4
23				Circle	Blue	1
24				Square	Amber	4
25				Triangle	Blue	3
26				Circle	Green	1
27				Rectangle	Blue	4
28						

If I were to type `=F12:H27` into another cell, Excel in the past would have thought I had gone mad. I'd need to wrap it in an aggregation function such as **SUM**, **COUNT** or **MAX**, to name but a few. Otherwise, I would have to wrap it in braces using **CTRL + SHIFT + ENTER** and use it as an array formula.

But no more.

Look at what happens when I type `=F12:H27` into cell **F33**:

	C	D	E	F	G	H
30						
31				Dynamic Array Result		
32						
33				Shape	Colour	Sides
34				Triangle	Red	3
35				Rectangle	Amber	4
36				Circle	Green	1
37				Triangle	Red	3
38				Square	Blue	4
39				Rectangle	Blue	4
40				Rectangle	Amber	4
41				Circle	Amber	1
42				Triangle	Red	3
43				Square	Green	4
44				Circle	Blue	1
45				Square	Amber	4
46				Triangle	Blue	3
47				Circle	Green	1
48				Rectangle	Blue	4
49						

The formula *automatically extends* to three columns by 16 rows! It has *spilled*. Get used to the vernacular. There's a reason this article got the name it did!

Any formula that has the potential to return multiple results can be referred to as a **Dynamic Array** formula. Formulae that are currently returning multiple results, and are successfully spilling, can be referred to as **Spilled Array Formulae**.

Notice I did not have to highlight all of the cells **F33:H48**. It spilled. Also take note I formatted cell **F33** – er, that didn't spill, because presently formatting isn't propagated. This is why this is not yet Generally Available. Microsoft is still trying to work out what should and shouldn't be allowed to happen in this first release. But don't let that put you off.

And don't let this basic example put you off either. If you feel a general sense of underwhelm coming over you, it's because I haven't yet communicated how powerful this all is as my example was too basic.

However, before I carry on there is a question I do need to cover with my far too simple example: what happens if something gets in the way?

	C	D	E	F	G	H	
30							
31		Dynamic Array Result					
32							
33				#SPILL!			
34							
35							
36							
37							
38							
39							
40					I'm in the way.		
41							
42							
43							
44							
45							
46							
47							
48							
49							

In this example, in cell **G40**, I have typed in the obtrusive text, "I'm in the way". And it quite literally is. Consequently, I have generated the brand new **#SPILL!** error. The formula cannot spill, so the error message is generated accordingly.

#SPILL! Errors

#SPILL! errors are returned when a formula returns multiple results, and Excel cannot return the results to the spreadsheet. There are various reasons an **#SPILL!** error could occur:

- **spill range is not blank:** as in my example (*above*), this error occurs when one or more cells in the designated spill range are not blank and thus may not be populated.

		fx =SORT(D2:D11)	
C	D	E	F
	Unit		#SPILL!
	622		
	961		
	691		BLOCKAGE
	445		
	378		
	483		
	650		
	783		
	142		
	404		

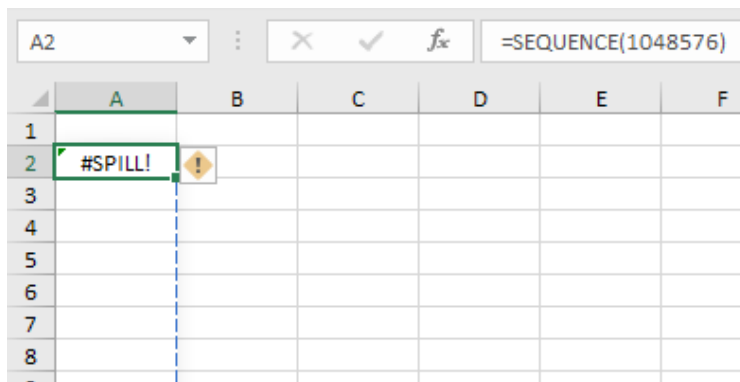
When the formula is selected, a dashed border will indicate the intended spill range. You may select the error "floatie" (believe it or not, this is what Microsoft call these things!), and choose the 'Select Obstructing Cell' option to immediately go the obstructing cell. You can then clear the error by either deleting or moving the obstructing cell's entry. As soon as the obstruction is cleared, the array formula will spill as intended

- **the range is volatile in size:** this means the size is not “set” and can vary. Excel was unable to determine the size of the spilled array because it's volatile and resizes between calculation passes. For example, the new function **SEQUENCE(x)** (explained in detail below) generates a list of **x** numbers increasing by 1 from 1 to **x**. That's fine, but the following formula will trigger this **#SPILL!** error:

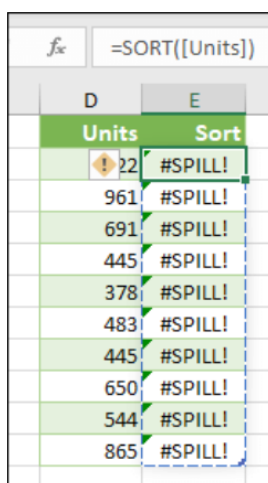
=SEQUENCE(RANDBETWEEN(1,1000)).

Dynamic array resizes may trigger additional calculation passes to ensure the spreadsheet is fully calculated. If the size of the array continues to change during these additional passes and does not stabilise, Excel will resolve the dynamic array as **#SPILL!** This error type is generally associated with the use of **RAND**, **RANDARRAY** and **RANDBETWEEN** functions. Other volatile functions such as **OFFSET**, **INDIRECT** and **TODAY** do not return different values on every calculation pass so tend not to generate this error

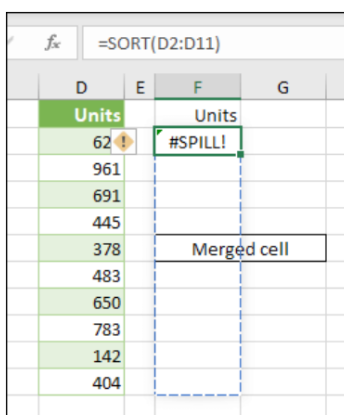
- **extends beyond the worksheet's edge:** in this situation, the spilled array formula you are attempting to enter will extend beyond the worksheet's range. You should try again with a smaller range or array. For example, moving the following formula to cell **A1** will resolve the error, and the formula will spill correctly



- **Table formula:** as I will explain shortly, Tables and Dynamic Arrays are not yet best friends. Spilled array formulae aren't supported in Excel Tables (generated by **CTRL + T**). Try moving your formula out of the Table, or go to **Table Tools -> Convert to range**



- **out of memory:** I have forgotten what this one means. Sorry, I couldn't resist that. The spilled array formula you are attempting to enter has caused Excel to run out of memory. You should try referencing a smaller array or range
- **spill into merged cells:** spilled array formulae cannot spill into merged cells. You will need to un-merge the cells in question or else move the formula to another range that doesn't intersect with merged cells.



When the formula is selected, a dashed border will indicate the intended spill range. You can again select that wonderfully named error floatie and choose the 'Select Obstructing Cell' option to immediately go the obstructing cell. As soon as the merged cells are cleared, the array formula will spill as intended

- **unrecognised / fallback error:** the "catch all" variant. Excel doesn't recognise, or cannot reconcile, the cause of this error. Here, you should make sure your formula contains all of the required arguments for your scenario.

Returning to Dynamic Arrays

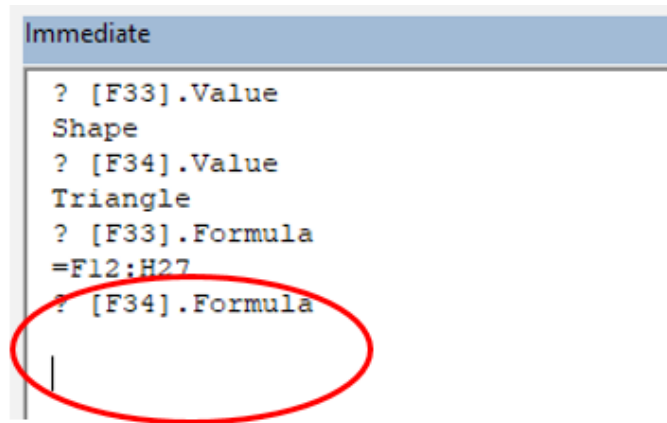
Now that we have considered what happens if you block a Dynamic Array, let me now turn my attention to what happens if you *don't*. You get the following:

Shape	Colour	Sides
Triangle	Red	3
Rectangle	Amber	4
Circle	Green	1
Triangle	Red	3
Square	Blue	4
Rectangle	Blue	4
Rectangle	Amber	4
Circle	Amber	1
Triangle	Red	3
Square	Green	4
Circle	Blue	1
Square	Amber	4
Triangle	Blue	3
Circle	Green	1
Rectangle	Blue	4

Do you see I am not having to anchor cells (*i.e.* use dollar [\$] signs)? The formula just *spills*. Let me be clear. If I select cell **F34**, I get the following:

Shape	Colour	Sides
Triangle	Red	3
Rectangle	Amber	4
Circle	Green	1
Triangle	Red	3
Square	Blue	4
Rectangle	Blue	4
Rectangle	Amber	4
Circle	Amber	1
Triangle	Red	3
Square	Green	4
Circle	Blue	1
Square	Amber	4
Triangle	Blue	3
Circle	Green	1
Rectangle	Blue	4

Here's a first. Check out the formula in the formula bar. It's *greyed out*. Ever seen that before? Effectively, cell **F34** contains the value 'Triangle' but it does not actually contain an "Excel" formula in the usual sense. To demonstrate this, let me show you the VBA Immediate Window:



But, to quote Bill Jelen, similar to Schrodinger's Cat, if you select cells **F33:H48** and use 'Go To Special' (**F5 -> Special**), and then select 'Formulas', cells **F33:H48** are shown as formula cells. You can even copy and paste them as values. Ladies and gentlemen, welcome to The Twilight Zone (cue eerie music).

I mentioned in the *#SPILL!* errors section that you cannot use Dynamic Arrays in a Table, but Dynamic Arrays may refer to a Table, viz.

Table Data				Result		
Shape	Colour	Sides		Shape	Colour	Sides
Triangle	Red	3		Triangle	Red	3
Rectangle	Amber	4		Rectangle	Amber	4
Circle	Green	1		Circle	Green	1
Triangle	Red	3		Triangle	Red	3
Square	Blue	4		Square	Blue	4
Rectangle	Blue	4		Rectangle	Blue	4
Rectangle	Amber	4		Rectangle	Amber	4
Circle	Amber	1		Circle	Amber	1
Triangle	Red	3		Triangle	Red	3
Square	Green	4		Square	Green	4
Circle	Blue	1		Circle	Blue	1
Square	Amber	4		Square	Amber	4
Triangle	Blue	3		Triangle	Blue	3
Circle	Green	1		Circle	Green	1
Rectangle	Blue	4		Rectangle	Blue	4

In this above illustration, cells **F57:H72** have been converted into a Table (**CTRL + T**), with the Table named **Basic_Array_Example**. In cell **L57**, I have simply typed '=' and then highlighted the entire Table. It was all replicated.

The advantage of linking a Dynamic Array to a Table is clear:

Table Data					Result			
Shape	Colour	Sides	First Letter		Shape	Colour	Sides	First Letter
Triangle	Red	3	T		Triangle	Red	3	T
Rectangle	Amber	4	R		Rectangle	Amber	4	R
Circle	Green	1	C		Circle	Green	1	C
Triangle	Red	3	T		Triangle	Red	3	T
Square	Blue	4	S		Square	Blue	4	S
Rectangle	Blue	4	R		Rectangle	Blue	4	R
Rectangle	Amber	4	R		Rectangle	Amber	4	R
Circle	Amber	1	C		Circle	Amber	1	C
Triangle	Red	3	T		Triangle	Red	3	T
Square	Green	4	S		Square	Green	4	S
Circle	Blue	1	C		Circle	Blue	1	C
Square	Amber	4	S		Square	Amber	4	S
Triangle	Blue	3	T		Triangle	Blue	3	T
Circle	Green	1	C		Circle	Green	1	C
Rectangle	Blue	4	R		Rectangle	Blue	4	R
Pineapple	Purple	117	P		Pineapple	Purple	117	P

I can add rows and / or columns and the Dynamic Array will update automatically. Do note that this does not breach the #SPILL! range is volatile in size error. This is because the range size will not vary on every calculation pass.

Talking of varying sizes, it's clear to see one potential issue with Dynamic Arrays. If we are not referring to a Table, what happens if the source data changes dimensions? This may be why you should refer to a Table for safety.

However, once you have a Dynamic Array, referring to it is simple using what is known as the **Spilled Range Operator**. For example, if I want to refer to the Dynamic Array in the previous examples, it initially had a range of L57:N72. However, once I had added a row and column to the Table, this resized to L57:O73. I can easily refer to this array, whatever its size as follows. In its initial state:

The screenshot shows an Excel formula bar containing '=L57#'. Below it, a table is displayed with the following data:

Shape	Colour	Sides
Triangle	Red	3
Rectangle	Amber	4
Circle	Green	1
Triangle	Red	3
Square	Blue	4
Rectangle	Blue	4
Rectangle	Amber	4
Circle	Amber	1
Triangle	Red	3
Square	Green	4
Circle	Blue	1
Square	Amber	4
Triangle	Blue	3
Circle	Green	1
Rectangle	Blue	4

The formula =L57# allows for variations – you simply type in the top left-hand cell reference (i.e. the cell with the non-greyed out formula) and add '#', known as the Spilled Range Operator. Simple!

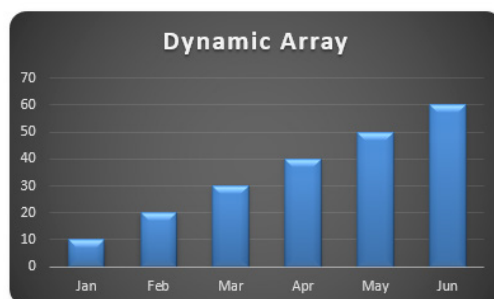
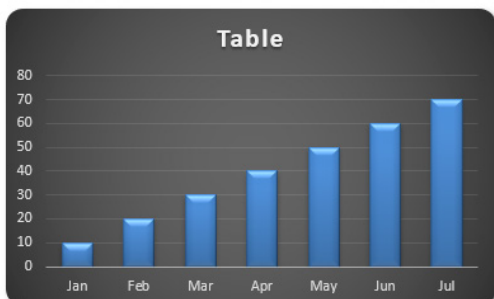
It's not all peaches and cream though. Whilst Dynamic Arrays and Tables share some similarities, they are very different beasts. This couldn't be clearer than when you create charts:

Table

Month	Sales
Jan	10
Feb	20
Mar	30
Apr	40
May	50
Jun	60
Jul	70

Dynamic Array

Month	Sales
Jan	10
Feb	20
Mar	30
Apr	40
May	50
Jun	60
Jul	70



Here, I created two charts when I only had the data up to June. Then, I added the data for July. The chart on the left referencing the Table source data updated instantly. However, the chart on the right still only displayed up to June even though the Dynamic Array had updated.

Conclusion: use Tables, not Dynamic Arrays, as your references for dynamic charts.

Implicit Intersection Implications

It may be an alliteration and sound like something you can get arrested for, but Dynamic Arrays do come at a price. There aren't many users out there who used them, but there are some – and hence there will be some legacy calculations affected.

In the past, if you entered `=A$1:A$10` anywhere in rows 1 through 10, the formula would return only the value from that row. In fact, a

spreadsheet our company is presently auditing relies on this behaviour. However, in the brave new world of Office 365 (albeit selected Insider recipients for the time being), typing this formula would create a Spilled Array Formula. To protect existing formulae, we need a new – if not instantly breathtaking – function...

SINGLE Function

Don't judge the remaining six functions on our first newbie. This one is essential to keep Excel running smoothly, but it's probably safe to say it won't set the world alight. It's like toilet roll – imagine your situation without it...

The **SINGLE** function returns a single value using logic known as implicit intersection. **SINGLE** may return a value, single cell range or an error.

The function has the following syntax:

=SINGLE(value).

The function has just one argument:

- **value:** this argument is required and represents the array to be selected.

When the supplied argument is a range, **SINGLE** will return the cell at the intersection of the row or column of the formula cell. Where there is no intersection, or more than one cell falls in the intersection, then **SINGLE** will return a **#VALUE!** error. When the supplied argument is an array, **SINGLE** returns the first item (Row 1, Column 1).

In the example below, the two **SINGLE** formulae are supplied a range, **H13:H27**, and return the values in cells **H17** and **H22** respectively.

First Name	Last Name	Points
Ivan	Idea	717
Amanda	Hugankiss	885
Artie	Detoo	976
Blake	Seven	247
Piper	Pied	978
Ivana	Tinkle	508
Artie	Chokes	300
Mike	Stand	778
Shelley	Ack	954
Blade	Runner	203
Sheikh	Spear	711
Mike	Robe	305
Daley	News	839
Hugo	There	611
Mimi	Selfish	197

I can see an argument going forward that some form of **OFFSET** (e.g. "NEXT" or "PRIOR") may be needed in due course – but no one is expecting everything to come together on Day 1.

Dynamic Arrays vs. Legacy Array Formulae

Prior to this new functionality, if you wanted to work with ranges in Excel, you used to have to build array formulae, where references would refer to ranges and be entered as **CTRL + SHIFT + ENTER** formulae. The main differences are as follows:

- Dynamic Array formulae may spill outside the cell bounds where the formula is entered. The Dynamic Array formula technically only exists in the cell in the top left-hand corner of the spilled range (as shown earlier), whereas with a legacy **CTRL + SHIFT + ENTER** formula, the formula would need to be entered in the entire range
- Dynamic Arrays will automatically resize as data is added or removed from the source range. **CTRL + SHIFT + ENTER** array formulae will truncate the return area if it's too small, or return **#N/A** errors if too large
- Dynamic Array formulae will calculate in a 1 x 1 context
- Any new formulae that return more than one result will automatically spill. There's simply no need to press **CTRL + SHIFT + ENTER**
- According to Microsoft, **CTRL + SHIFT + ENTER** array formulae are only retained for backwards compatibility reasons. Going forward, you should use Dynamic Array formulae instead
- Dynamic Array formulae may be easily modified by changing the source cell, whereas **CTRL + SHIFT + ENTER** array formulae require that the entire range be edited simultaneously

- Column and row insertion / deletion is prohibited in an active **CRL + SHIFT + ENTER** Array formula range. You first need to delete any existing array formulas that are in the way.

Everybody clear? I think we are finally good to start introducing the other new functions...

SORT Function

I am not going to do these alphabetically – let me show then in an order that makes sense (well, to me, anyway).

The **SORT** function sorts the contents of a range or array:

=SORT(array, [sort_index], [sort_order], [by_column]).

It has four arguments:

- **array**: this is required and represents the range that is required to be sorted
- **sort_index**: this is optional and refers to the position of the row or the column in the selected **array** (e.g. second row, third column). 99 times out of 98 you will be defining the column, but to select a row you will need to use this argument in conjunction with the fourth argument, **by_column**. And be careful, it's a little counter-intuitive! The default value is 1
- **sort_order**: this is also optional. The choices for **sort_order** are 1 for ascending (default) or -1 for descending. It should be noted that you might not want to hold your breath waiting for 'Sort by Color' (sic), 'Sort by Formula' or 'Sort by Custom List' using this function
- **by_column**: this final argument is also optional. Most people want to sort rows of data, so they will want the value to be FALSE (which is the default value if not specified). Should you be booking your mental health check, you may wish to use TRUE to sort by column in certain instances.

This is a function people have been crying out for, for *years*. Enterprising spreadsheets gurus have developed array formulae and user-defined functions that have replicated this functionality, but you don't need it anymore! **SORT** is coming to a theatre near you very soon.

To show you how devilishly simple it is, consider the following data:

	C	D	E	F	G	H
9						
10				Original Data		
11						
12				First Name	Last Name	Points
13				Ivan	Idea	717
14				Amanda	Hugankiss	885
15				Artie	Detoo	976
16				Blake	Seven	508
17				Piper	Pied	978
18				Ivana	Tinkle	508
19				Artie	Chokes	300
20				Mike	Stand	778
21				Shelley	Ack	954
22				Blade	Runner	203
23				Sheikh	Spear	711
24				Mike	Robe	305
25				Daley	News	839
26				Hugo	There	611
27				Mimi	Selfish	197
28						

Sorting the 'Points' column in order is easy as this:

	C	D	E	F	G	H	I
29							
30				Sorted Points			
31							
32				197			
33				203			
34				300			
35				305			
36				508			
37				508			
38				611			
39				711			
40				717			
41				778			
42				839			
43				885			
44				954			
45				976			
46				978			
47							

All you have to do is type **=SORT(H13:H27)** into cell **F32**. That's it! Note that the duplicates are repeated; there is no cull. If you want it in descending order, simply specify the requirement in the formula:

The screenshot shows the Excel formula bar with the formula `=SORT(H13:H27,-1)` entered in cell F51. Below the formula bar, the spreadsheet shows a column of sorted points in descending order, starting with 978 and ending with 197.

Row	Points
51	978
52	976
53	954
54	885
55	839
56	778
57	717
58	711
59	611
60	508
61	508
62	305
63	300
64	203
65	197

This formula is only slightly more sophisticated, in that the **sort_order** (third argument) needs to be specified as -1 to switch the sort to descending:

=SORT(H13:H27,-1).

You probably won't want the points displayed on their own:

The screenshot shows the Excel formula bar with the formula `=SORT(F13:H27,3,-1)` entered in cell F70. Below the formula bar, the spreadsheet shows a full table of sorted points in descending order, with names and points.

Row	Name	Points
70	Piper	978
71	Artie	976
72	Shelley	954
73	Amanda	885
74	Daley	839
75	Mike	778
76	Ivan	717
77	Sheikh	711
78	Hugo	611
79	Blake	508
80	Ivana	508
81	Mike	305
82	Artie	300
83	Blade	203
84	Mimi	197

Now all of these arguments start to make more sense. **SORT(F13:H27,3,-1)** produces the whole array (**array** is **F13:H27**), sorts it on the third (**sort_index** **3**) column in descending (**sort_order** **-1**) order. Blake and Ivana tie on 508 points, but Blake appears first as he was first in the original (source) table.

So far, I have only performed the one **SORT**. You can have more than one though:

The screenshot shows the Excel formula bar with the formula `=SORT(F13:G27,{1,2},{1,-1})` entered in cell F89. Below the formula bar, the spreadsheet shows a two-stage sort of names and points, with names sorted in ascending order and points in descending order.

Row	Name	Points
89	Amanda	Hugankiss
90	Artie	Detoo
91	Artie	Chokes
92	Blade	Runner
93	Blake	Seven
94	Daley	News
95	Hugo	There
96	Ivan	Idea
97	Ivana	Tinkle
98	Mike	Stand
99	Mike	Robe
100	Mimi	Selfish
101	Piper	Pied
102	Sheikh	Spear
103	Shelley	Ack

Here, I have created a second (two-level) **SORT**. Here, you need to create what is known as an array constant for the second and third arguments (you just type the braces in – don't use **CTRL + SHIFT + ENTER**):

=SORT(F13:G27,{1;2},{1;-1}).

This will sort on column 1 ('First Name') first, then sort on column 2 ('Last Name') next. This will be in ascending order (1) for the first column and descending order (-1) for the latter. It's not as straightforward a formula entry as most Excel modellers are used to, but it's relatively straightforward once you have committed it to erm, um, what do you call it, memory.

My final example of **SORT** is not something that is limited to this new function, but it does show how things fit together. From all that has been written above, it appears you can only get one value (using **SINGLE**) or all of them (using Dynamic Arrays). That's not true as this illustration clearly demonstrates:

	C	D	E	F	G	H	I
105							
106							
107							
108				Piper	Pied	978	
109				Artie	Detoo	976	
110				Shelley	Ack	954	
111							

Only the top three have spilled in this example. How? Well, I cheated. I highlighted cells **F108:H110** first, then typed in the formula

=SORT(F13:H27,3,-1)

and then pressed **CTRL + SHIFT + ENTER** (thus generating the { and } braces). This restricted the spill to the range stipulated. Cool. Other than making sure no one can delete or insert any rows by creating an array formula such as **{=1}** across the restricted area, these appear to be the only two used of **CTRL + SHIFT + ENTER** now.

SORT is really useful then, but what if you want to sort on a field you don't want displayed in the results..?

SORTBY Function

The **SORTBY** function sorts the contents of a range or array based on the values in a corresponding range or array, which does not need to be displayed. The syntax is as follows:

=SORTBY(array, by_array1, [sort_order1], [by_array2], [sort_order2], ...).

It has several arguments:

- **array**: this is required and represents the range that is required to be sorted
- **by_array1**: this is the first range that array will be sorted on and is required
- **sort_order1, sort_order2, ...**: these are optional. The choices for each sort_order are 1 for ascending (default) or -1 for descending
- **by_array2, ...**: these arguments are also optional. These represent the second and subsequent ranges that **array** will be sorted on.

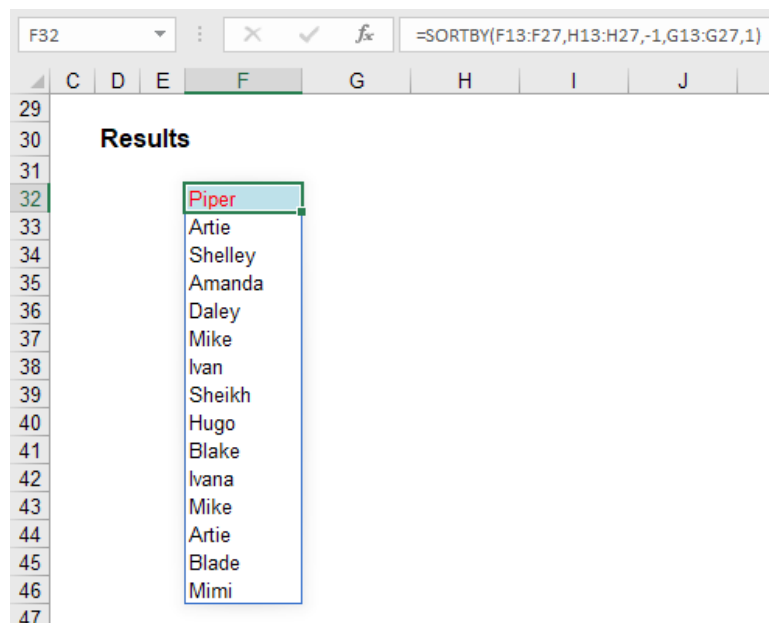
There are some important considerations to note:

- the **by_array** arguments must either be one row high or one column wide
- all of the **by_array** arguments must be the same size and contain the same number of rows as **array** if sorting on rows, or the same number of columns as array if sorting on columns
- if the sort order argument is not 1 or -1, the formula will result in a **#VALUE!** error.

It's pretty simple to use. Consider the following source data once more:

	C	D	E	F	G	H
9						
10				Original Data		
11						
12				First Name	Last Name	Points
13				Ivan	Idea	717
14				Amanda	Hugankiss	885
15				Artie	Detoo	976
16				Blake	Seven	508
17				Piper	Pied	978
18				Ivana	Tinkle	508
19				Artie	Chokes	300
20				Mike	Stand	778
21				Shelley	Ack	954
22				Blade	Runner	203
23				Sheikh	Spear	711
24				Mike	Robe	305
25				Daley	News	839
26				Hugo	There	611
27				Mimi	Selfish	197
28						

I can use **SORTBY** as follows:



Here, using the formula

=SORTBY(F13:F27,H13:H27,-1,G13:G27,1)

I have sorted the 'First Name' field (**F13:F27**) on the 'Points' column (**H13:H27**) in descending (-1) order and then used the second sort on 'Last Name' (**G13:G27**) in ascending (1) order. No need for those pesky array references in multiple sorts with the **SORT** function (*as detailed above*).

FILTER Function

The **FILTER** function will accept an array, allow you to filter a range of data based upon criteria you define and return the results to a spill range.

The syntax of **FILTER** is as follows:

=FILTER(array, include, [if_empty]).

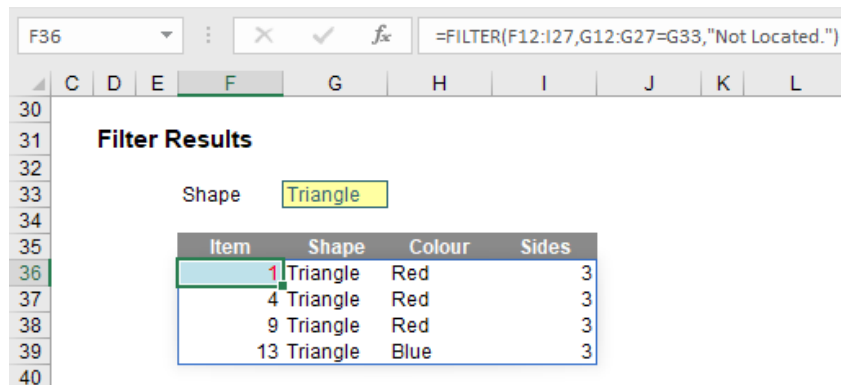
It has three arguments:

- **array:** this is required and represents the range that is to be filtered
- **include:** this is also required. This specifies the condition(s) that must be met
- **if_empty:** this argument is optional. This is what will be returned if no data meets the criterion / criteria specified in the **include** argument. It's generally a good idea to at least use "" here.

For example, consider the following source data:

	C	D	E	F	G	H	I
9							
10				Original Data			
11							
12							
13				Item	Shape	Colour	Sides
14				1	Triangle	Red	3
15				2	Rectangle	Amber	4
16				3	Circle	Green	1
17				4	Triangle	Red	3
18				5	Square	Blue	4
19				6	Rectangle	Blue	4
20				7	Rectangle	Amber	4
21				8	Circle	Amber	1
22				9	Triangle	Red	3
23				10	Square	Green	4
24				11	Circle	Blue	1
25				12	Square	Amber	4
26				13	Triangle	Blue	3
27				14	Circle	Green	1
28				15	Rectangle	Blue	4

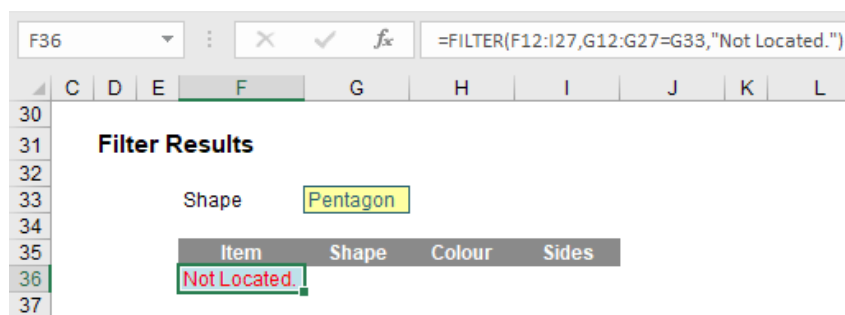
To begin with, I will perform a simple **FILTER**:



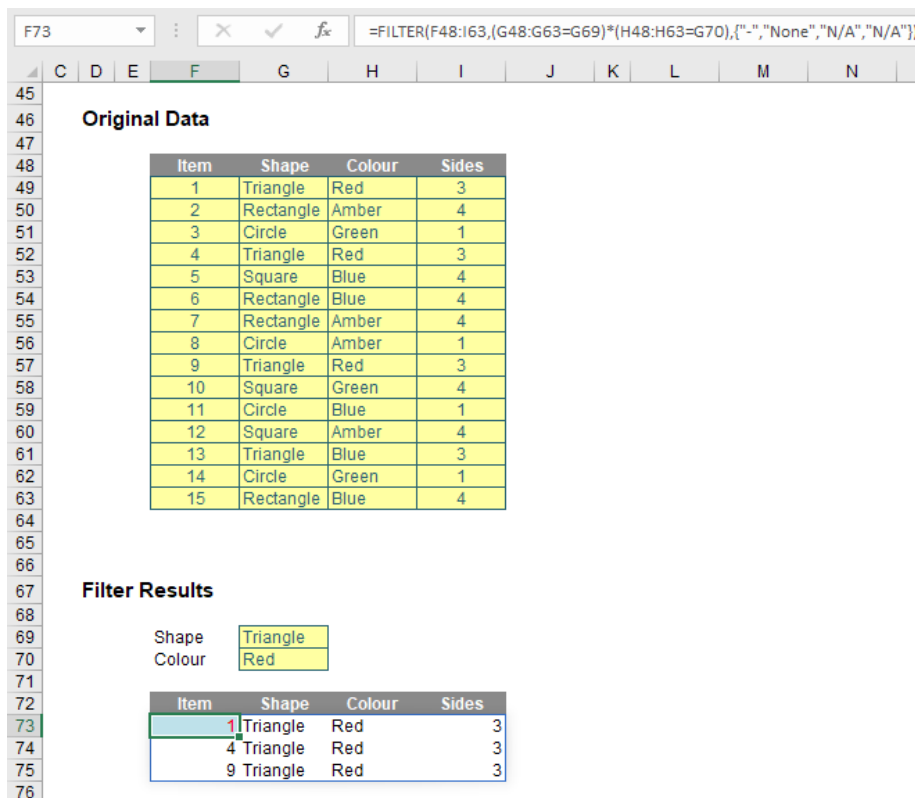
Here, in cell **F36**, I have created the formula

=FILTER(F12:I27,G12:G27=G33,"Not Located.")

F12:I27 is my source **array** and I wish only to **include** shapes (column **G12:G27**) that are 'Triangles' (specified by cell **G33**). If there are no such shapes, then **"Not Located."** is returned instead. To show this, I will change the shape as follows:



That is about as basic as it gets. I can get cleverer. Consider the following example:



I have repeated the source **array** (cells **F48:I63**) for clarity. The formula

=FILTER(F48:I63,(G48:G63=G69)*(H48:H63=G70),{"-","None","N/A","N/A"})

looks horrible to begin with, but it's not quite as bad as it appears upon further scrutiny. The include argument,

(G48:G63=G69)*(H48:H63=G70)

contains two conditions. Firstly, **G48:G63=G69** means that the 'Shape' (column **G48:G63**) has to be a 'Triangle' (cell **G69**) and that the 'Colour' (column **H48:H63**) has to be 'Red' (cell **G70**). The multiplication operator (*) is used to denote **AND**. The Excel function **AND** cannot be used with arrays – this is nothing special to Dynamic Arrays; **AND** does not work with **CTRL + SHIFT + ENTER** formulae either. This syntax is similar to how you would create **AND** criteria with the **SUMPRODUCT** function, for example.

The final argument is similar to the syntax in **SORT**: {"-","None","N/A","N/A"}. Braces (typed in!) are used to create an array argument that specifies what should be written in each column should there be no record that meets both criteria, e.g.

See? Not as bad as you might first think.

My final example is very similar:

Once you realise I have simply repeated referencing for clarity, the formula

=FILTER(F84:I99,(G84:G99=G105)+(H84:H99=G106),{"-","None","N/A","N/A"})

is nothing more than the **OR** equivalent of the previous example, with '+' replacing '*' to switch from ensuring both conditions are met to only one condition being met. As at the time of writing, **XOR** is not catered for, but I am sure some clever person will create an equivalent in due course (if Microsoft doesn't beat them to it), necessity being the mother of invention and all that jazz.

Interlude: the #CALC! Error

I mentioned there were two new error messages. I have only referred to **#SPILL!** so far. There is another, lurking in the background (I say "in the background" as at the time of writing, Microsoft hasn't written any documentation on it!).

Sometimes, as you explore how you can combine Excel functions with each other you get error messages (e.g. more often than not trying **FILTER(FILTER(...** will generate an **#VALUE!** error). When you start playing with these new array functions, you might stumble upon **#CALC!** This is a new one.

To add to the myriad of error messages such **#REF!**, **#DIV/0!**, **#VALUE!**, **#BROWN** and **#PIPE**, let's introduce **#CALC!** – which probably means something like, "Excel cannot currently figure out the answer presently, but might be able to in a future release, no promises though". I look forward to the documentation in due course though to fathom its real meaning (probably something like, "Help! Abandon ship!").

Let's move on.

UNIQUE Function

The hilarious thing about **UNIQUE** is that it does two things (!). It details distinct items (i.e. provides each value that occurs with no repetition) and also it can return values which occur once and only once in a referred range. I understand that Excel users may welcome the former use with open arms and that database developers may be very interested in the latter. I still think there should have been two functions though. Otherwise, let's just extend the **AGGREGATE** function to do just *everything* (it almost does now) and be done with it!

The **UNIQUE** function has the following syntax:

=UNIQUE(array, [by_column], [occurs_once]).

It has three arguments:

- **array:** this is required and represents the range or array from which to return unique values
- **by_column:** this argument is optional. This is a logical value (TRUE / FALSE) indicating how to compare. If you wish to compare by row, the argument should be FALSE or omitted (since this is the default). To compare by column, you will need to select TRUE
- **occurs_once:** this argument is also optional. This requires a logical value too:
 - o **TRUE:** only return unique values that occur once
 - o **FALSE:** include all distinct values (default if omitted).

It's probably clearer with some examples. Let's give it a go. As always, I need source data:

	C	D	E	F	G	H	I
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30							
31							
32							
33							
34							
35							
36							
37							
38							
39							
40							
41							
42							

Original Data			
Store	Salesperson	Section	Manager
North	Alice	White Goods	Zack
North	Barbara	Groceries	Zack
North	Charlie	White Goods	Zack
North	Dion	Computers	Yvonne
North	Echo	Insurance	Xander
North	Fred	Bedding	Winnie
North	George	Audio Video	Yvonne
North	Helen	Furniture	Winnie
North	Iris	White Goods	Zack
North	Jack	Furniture	Winnie
North	Karla	Groceries	Zack
East	Lindsay	Insurance	Xander
East	Barbara	Groceries	Zack
East	Iris	White Goods	Zack
East	Michael	Computers	Yvonne
East	Fred	Bedding	Winnie
East	Dion	Computers	Yvonne
South	Nancy	Audio Video	Yvonne
South	Oprah	Furniture	Winnie
South	Helen	Furniture	Winnie
South	Alice	White Goods	Zack
South	Pete	Groceries	Zack
West	Karla	Groceries	Zack
West	Pete	Groceries	Zack
West	Charlie	White Goods	Zack
West	Dion	Computers	Yvonne
West	George	Audio Video	Yvonne
West	Nancy	Audio Video	Yvonne
West	Michael	Computers	Yvonne

Time for the most basic illustration:

	J	K	L	M	N	O
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						
29						

Results			
Store	Salesperson	Section	Manager
North	Alice	White Goods	Zack
East	Barbara	Groceries	Yvonne
South	Charlie	Computers	Xander
West	Dion	Insurance	Winnie
	Echo	Bedding	
	Fred	Audio Video	
	George	Furniture	
	Helen		
	Iris		
	Jack		
	Karla		
	Lindsay		
	Michael		
	Nancy		
	Oprah		
	Pete		

In cell L13, I have simply typed

=UNIQUE(F13:F41).

No optional arguments; everything in default. If I have made an error, it's going to be my default. This has simply listed each store that appears; if "North" and "North " (extra space) were there, then both would appear. **UNIQUE** is not case sensitive though and each entry would appear as it first occurs reading down the range **F13:F41**. The other columns contain similar formulae and **UNIQUE** looks like it takes seconds to learn. Presently, there's an in-joke going around the Excel Most Valuable Professionals (MVPs) that array expert Mike Girvin is going to be choked as he dedicated *an entire chapter* in one of his books to creating that list with an array formula! Sorry Mike. Excel is fun!

It's just as simple if you want to see unique records for two (or more) columns, viz.

Section	Manager
White Goods	Zack
Groceries	Zack
Computers	Yvonne
Insurance	Xander
Bedding	Winnie
Audio Video	Yvonne
Furniture	Winnie

You can see **UNIQUE** is sort of crying out for **SORT**, but we'll get to that shortly.

As mentioned earlier, it's not the only way of using **UNIQUE** (no, having a unique use would be just what "they" were expecting, whoever "they" are...). You can use it to determine values that only occur once:

Store	Salesperson	Section	Manager
North	Alice	White Goods	Zack
North	Barbara	Groceries	Zack
North	Charlie	White Goods	Zack
North	Dion	Computers	Yvonne
North	Echo	Insurance	Xander
North	Fred	Bedding	Winnie
North	George	Audio Video	Yvonne
North	Helen	Furniture	Winnie
North	Iris	White Goods	Zack
North	Jack	Furniture	Winnie
North	Karla	Groceries	Zack
East	Lindsay	Insurance	Xander
East	Barbara	Groceries	Zack
East	Iris	White Goods	Zack
East	Michael	Computers	Yvonne
East	Fred	Bedding	Winnie
East	Dion	Computers	Yvonne
South	Nancy	Audio Video	Yvonne
South	Oprah	Furniture	Winnie
South	Helen	Furniture	Winnie
South	Alice	White Goods	Zack
South	Pete	Groceries	Zack
West	Karla	Groceries	Zack
West	Pete	Groceries	Zack
West	Charlie	White Goods	Zack
West	Dion	Computers	Yvonne
West	George	Audio Video	Yvonne
West	Nancy	Audio Video	Yvonne
West	Michael	Computers	Yvonne

Salesperson
Echo
Jack
Lindsay
Oprah

Here, the formula in cell L56,

=UNIQUE(G56:G84,0,1)

uses the non-default value of 1 for the optional **occurs_once** (third) argument. This means it identifies the salespeople who only occur once in cells **G56:G84**. Brilliant; I can die content knowing now.

The real power starts coming when you start playing with Excel's existing functions and features, together with these new functions. Take this comprehensive example:

L111 =SORT(UNIQUE(FILTER(F93:I122,IF(M108="OR",{H93:H122=M105}+(I93:I122=M106),{H93:H122=M105}*(I93:I122=M106)),{"N/A","-","-","-"})),{1;2;3;4},{1;1;1})

Original Data				Lookup Data	
Store	Salesperson	Section	Manager	Section	Manager
North	Alice	White Goods	Zack	Audio Video	Winnie
North	Barbara	Groceries	Zack	Bedding	Xander
North	Charlie	White Goods	Zack	Computers	Yvonne
North	Dion	Computers	Yvonne	Furniture	Zack
North	Echo	Insurance	Xander	Groceries	
North	Fred	Bedding	Winnie	Insurance	
North	George	Audio Video	Yvonne	White Goods	
North	Helen	Furniture	Winnie		
North	Iris	White Goods	Zack		
North	Jack	Furniture	Winnie		
North	Karla	Groceries	Zack		
East	Lindsay	Insurance	Xander		
East	Barbara	Groceries	Zack		
East	Iris	White Goods	Zack		
East	Michael	Computers	Yvonne		
East	Fred	Bedding	Winnie		
East	Dion	Computers	Yvonne		
South	Nancy	Audio Video	Yvonne		
South	Oprah	Furniture	Winnie		
South	Helen	Furniture	Winnie		
South	Alice	White Goods	Zack		
South	Pete	Groceries	Zack		
West	Karla	Groceries	Zack		
West	Pete	Groceries	Zack		
West	Charlie	White Goods	Zack		
West	Dion	Computers	Yvonne		
West	George	Audio Video	Yvonne		
West	Nancy	Audio Video	Yvonne		
West	Michael	Computers	Yvonne		

Section	Manager
Audio Video	Winnie
Bedding	Xander
Computers	Yvonne
Furniture	Zack
Groceries	
Insurance	
White Goods	

Filtered Summary

Section: **Computers**
 Manager: **Winnie**
 AND / OR: **OR**

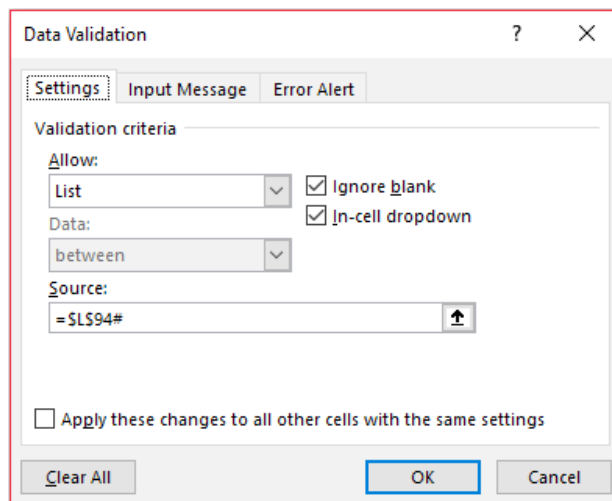
Store	Salesperson	Section	Manager
East	Dion	Computers	Yvonne
East	Fred	Bedding	Winnie
East	Michael	Computers	Yvonne
North	Dion	Computers	Yvonne
North	Fred	Bedding	Winnie
North	Helen	Furniture	Winnie
North	Jack	Furniture	Winnie
South	Helen	Furniture	Winnie
South	Oprah	Furniture	Winnie
West	Dion	Computers	Yvonne
West	Michael	Computers	Yvonne

Let me step you through *some* of this. The formulae in cells L94 and M94 use **UNIQUE** in a similar manner to my first example, to generate the list of distinct values in the 'Section' and 'Manager' fields. However, did you notice they have been sorted? That's because I used the formula

=SORT(UNIQUE(H94:H122))

in cell L94, for example. Honestly, I think **UNIQUE** should have another argument for sorting (ascending / descending / none [default]). Watch Microsoft ignore that suggestion.

But then I did something really cool. Cells M105 and M106 use data validation (**ALT + D + L**) to generate a list from the 'Lookup Data' section. That requires taking a closer look:



Do you see the source for the data validation in cell M105? **= \$L\$84#** - so elegant! This takes the 'Section' list and automatically makes the drop-down list the required length! People create all sorts of tricks using **OFFSET**, dynamic range names and the like to achieve a similar effect. No more. **= \$L\$84#** (with the '#', the Spilled Range Operator) is all that is needed. That's my favourite thing in all of these new functions and features. I'm impressed – and I'm easily impressed.

The 'AND / OR' dropdown is a bit of an anti-climax after that, but the final formula that generates the final table, namely

=SORT(UNIQUE(FILTER(F93:I122,IF(M108="OR",{H93:H122=M105}+(I93:I122=M106),{H93:H122=M105}*(I93:I122=M106)),{"N/A","-","-","-"})),{1;2;3;4},{1;1;1})

is rather fun. I am not going to go through it though – as every aspect of this formula is simply a re-hash of an earlier point (assuming you know the **IF** function!). See if you can work your way through it for yourself.

SEQUENCE Function

The penultimate function is **SEQUENCE**. This function allows you to generate a list of sequential numbers in an array, such as 1, 2, 3, 4. It doesn't sound particularly exciting, but again, it really ramps up when combined with other functions and features. The syntax is given by:

=SEQUENCE(rows, [columns], [start], [step]).

It has four arguments:

- **rows**: this argument is required and specifies how many **rows** the results should spill over
- **columns**: this argument is optional and specifies how many **columns** (surprise, surprise) the results should spill over. If omitted, the default value is 1
- **start**: this argument is also optional. This specifies what number the **SEQUENCE** should **start** from. If omitted, the default value is 1
- **step**: this final argument is also optional. This specifies the amount each number in the **SEQUENCE** should increase (the "step"). It may be positive, negative or zero. If omitted, the default value is 937,444. Wait, I'm kidding; it's 1. They're very unimaginative down in Redmond.

Therefore, **SEQUENCE** can be as simple as **SIMPLE(x)**, which will generate a list of numbers in a column 1, 2, 3, ..., **x**. Therefore, be mindful not to create a formula where **x** may be volatile and generate alternative values each time it is calculated, e.g. **=SEQUENCE(RANDBETWEEN(10,99))** as this will generate the **#SPILL!** range is volatile in size error.

A vanilla example is rather bland:

The screenshot shows the Excel formula bar with the formula **=SEQUENCE(H12,H13,H14,H15)**. Below the formula bar, a table displays the inputs and outputs of the function.

Inputs	
No. of Rows	3
No. of Columns	4
Start	5
Step	6

Outputs			
5	11	17	23
29	35	41	47
53	59	65	71

Do you see how **SEQUENCE** propagates across the row first and then down to the next row, just like reading a book? I wonder how that might work in alternative languages of Excel where users read right to left (it has to be the same or there would be chaos when workbooks were shared!).

Some of my peers had fun combining it with the **ROMAN** function:

The screenshot shows the Excel formula bar with the formula **=ROMAN(SEQUENCE(10,10,1,1))**. Below the formula bar, a table displays the Roman numerals generated by the function.

Roman Numerals									
I	II	III	IV	V	VI	VII	VIII	IX	X
XI	XII	XIII	XIV	XV	XVI	XVII	XVIII	XIX	XX
XXI	XXII	XXIII	XXIV	XXV	XXVI	XXVII	XXVIII	XXIX	XXX
XXXI	XXXII	XXXIII	XXXIV	XXXV	XXXVI	XXXVII	XXXVIII	XXXIX	XL
XLI	XLII	XLIII	XLIV	XLV	XLVI	XLVII	XLVIII	XLIX	L
LI	LII	LIII	LIV	LV	LVI	LVII	LVIII	LIX	LX
LXI	LXII	LXIII	LXIV	LXV	LXVI	LXVII	LXVIII	LXIX	LXX
LXXI	LXXII	LXXIII	LXXIV	LXXV	LXXVI	LXXVII	LXXVIII	LXXIX	LXXX
LXXXI	LXXXII	LXXXIII	LXXXIV	LXXXV	LXXXVI	LXXXVII	LXXXVIII	LXXXIX	XC
XCI	XCII	XCIII	XCIV	XCV	XCVI	XCVII	XCVIII	XCIX	C

To my mind though, my favourite simple illustration is creating a monthly calendar. A little magic with the **DATE** and **WEEKDAY** functions combined with some conditional formatting and suddenly you have:

Dates

Month	Sep
Year	2018

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1 Sep 18
2 Sep 18	3 Sep 18	4 Sep 18	5 Sep 18	6 Sep 18	7 Sep 18	8 Sep 18
9 Sep 18	10 Sep 18	11 Sep 18	12 Sep 18	13 Sep 18	14 Sep 18	15 Sep 18
16 Sep 18	17 Sep 18	18 Sep 18	19 Sep 18	20 Sep 18	21 Sep 18	22 Sep 18
23 Sep 18	24 Sep 18	25 Sep 18	26 Sep 18	27 Sep 18	28 Sep 18	29 Sep 18
30 Sep 18						

As I mentioned above, **SEQUENCE** is arguably more powerful when included in a more complex formula. For example:

	1	2	3	4	5	6	7	8	9	10	11	12	Total	SEQUENCE
1	\$ 937.50	\$ 931.30	\$ 925.08	\$ 918.83	\$ 912.56	\$ 906.26	\$ 899.95	\$ 893.61	\$ 887.24	\$ 880.85	\$ 874.44	\$ 868.00	\$10,835.61	\$10,835.61
2	\$ 861.54	\$ 855.06	\$ 848.55	\$ 842.01	\$ 835.45	\$ 828.87	\$ 822.26	\$ 815.63	\$ 808.97	\$ 802.29	\$ 795.58	\$ 788.85	\$ 9,905.06	\$ 9,905.06
3	\$ 782.09	\$ 775.31	\$ 768.50	\$ 761.66	\$ 754.81	\$ 747.92	\$ 741.01	\$ 734.07	\$ 727.11	\$ 720.12	\$ 713.10	\$ 706.06	\$ 8,931.75	\$ 8,931.75
4	\$ 698.99	\$ 691.90	\$ 684.78	\$ 677.63	\$ 670.45	\$ 663.25	\$ 656.02	\$ 648.77	\$ 641.48	\$ 634.17	\$ 626.83	\$ 619.47	\$ 7,913.74	\$ 7,913.74
5	\$ 612.08	\$ 604.65	\$ 597.21	\$ 589.73	\$ 582.22	\$ 574.69	\$ 567.13	\$ 559.54	\$ 551.92	\$ 544.28	\$ 536.60	\$ 528.90	\$ 6,848.95	\$ 6,848.95
6	\$ 521.17	\$ 513.40	\$ 505.61	\$ 497.79	\$ 489.94	\$ 482.06	\$ 474.16	\$ 466.22	\$ 458.25	\$ 450.25	\$ 442.23	\$ 434.17	\$ 5,735.26	\$ 5,735.26
7	\$ 426.08	\$ 417.96	\$ 409.81	\$ 401.63	\$ 393.42	\$ 385.18	\$ 376.91	\$ 368.61	\$ 360.27	\$ 351.91	\$ 343.51	\$ 335.08	\$ 4,570.39	\$ 4,570.39
8	\$ 326.63	\$ 318.13	\$ 309.61	\$ 301.06	\$ 292.47	\$ 283.85	\$ 275.20	\$ 266.51	\$ 257.80	\$ 249.05	\$ 240.27	\$ 231.45	\$ 3,352.02	\$ 3,352.02
9	\$ 222.60	\$ 213.72	\$ 204.81	\$ 195.86	\$ 186.88	\$ 177.86	\$ 168.81	\$ 159.73	\$ 150.61	\$ 141.46	\$ 132.28	\$ 123.05	\$ 2,077.67	\$ 2,077.67
10	\$ 113.80	\$ 104.51	\$ 95.19	\$ 85.83	\$ 76.43	\$ 67.00	\$ 57.54	\$ 48.04	\$ 38.50	\$ 28.93	\$ 19.32	\$ 9.68	\$ 744.78	\$ 744.78

In this instance, I have created a grid using the Excel **IPMT** function to determine the amount of interest to be paid in each monthly instalment. Cells **G62:R71** calculate each monthly amount and column **T** sums these amounts to calculate the annual interest payment, a figure which is non-trivial to compute. The whole table may be replaced by the formula in cell **V62**:

=IF(\$F62="", "", -SUM(IPMT(Annual_Interest_Rate/Months_in_Year, SEQUENCE(1, Months_in_Year, (\$F62-1)*Months_in_Year+1, 1), Borrowing_Term*Months_in_Year, Amount_Borrowed))).

I am not going to explain this and let me tell you why. Our company, SumProduct builds and reviews financial models for a living. We see terrible modelling practices established day-in, day-out. We proactively try to discourage these traits by emphasising that complex formulae should be stepped out and made transparent. Here, that can be done using the original table. I don't want people using **SEQUENCE**, Dynamic Arrays or other spilled formulae to wrap up complicated calculations into an opaque Pandora's Box. Yes, calculation times may be slower. Live with it. Sometimes you need to see the scenery to appreciate the beauty. I'm just a little fearful that people will embrace these functions a little too readily and the Road to Excel Hell beckons shortly. Sorry to be a miserable git.

On an upbeat note, I put a formula in cell **G61** which is simple:

=TRANSPOSE(SEQUENCE(Months_in_Year)).

Yes, I am using **TRANSPOSE** without **CTRL + SHIFT + ENTER**. We are in new territory here...

RANDARRAY Function

And so, to the final function for now: **RANDARRAY**. The **RANDARRAY** function returns an array of random numbers between 0 and 1. It's not clear from Microsoft, but it's my belief this is analogous to the pre-existing **RAND** function, which generates a number greater than or equal to zero and strictly less than one. It is noted though that **RANDARRAY** is different from the **RAND** function insofar that **RAND** does not return an array.

The function has the following syntax:

=RANDARRAY([rows], [columns]).

The function has two arguments, both optional:

- **rows**: this specifies how many rows the results should spill over. If omitted, the default value is 1
- **columns**: this specifies how many columns the results should spill over. If omitted, the default value is also 1.

Therefore, **RANDARRAY()** behaves like **RAND()**.

Again, I'll start with a basic demonstration:

Inputs			
No. of Rows	3		
No. of Columns	4		
Outputs			
0.524976	0.683739	0.256505	0.243437
0.367763	0.497963	0.994841	0.411733
0.412296	0.059395	0.741392	0.51649

This can be useful for generating "seed" values for simulations analysis, for example. Anecdotal evidence suggests using **RANDARRAY** in a simulations formula rather than using a macro or Data Table may generate calculation speed efficiencies of greater than 50%. The jury is still out on this one – but it's definitely worth exploring.

There is no **RANDBETWEENARRAY** function, but you can construct it yourself:

The screenshot shows an Excel spreadsheet with the following data:

Inputs	
No. of Rows	5
No. of Columns	8
Start Number	1
End Number	10

Outputs	
6	2 1 3 8 9 1 2
6	10 8 3 1 6 4 2
8	7 5 7 3 4 2 10
6	4 1 5 5 10 4 2
3	9 6 1 1 10 10 9

Here, in cell **F44**, I have used the formula

=ROUNDDOWN(RANDARRAY(H36,H37)*(H39-H38+1),0)+INT(H38)

to generate my equivalent **RANDBETWEEN** function. Since **RANDARRAY** generates a random number greater than or equal to zero and strictly less than one, **ROUNDDOWN(RANDARRAY(H36,H37)*(H39-H38+1),0)** generates a random integer (all equally likely) between zero and nine (this is 9, because the number generated is a whole number greater than or equal to zero but strictly less than 10 [10 – 1 + 1]). Since this is added to 1 (**INT(H38)**), the random whole number will be an integer between 1 and 10. Simple, if you have a Maths PhD.

For a final example, imagine you are a schoolteacher and you have 10 five-year-old children:

The screenshot shows an Excel spreadsheet with the following data:

Names of Children in Class	
	Children
	Alex
	Bubba
	Carlo
	Diana
	Erica
	Felix
	Grace
	Horace
	Isla
	Jules

For each of the next 10 weeks, you have topics you want one of them to present on:

The screenshot shows an Excel spreadsheet with the following data:

Presentation Topics	
	Subjects
	Algebra for the Hard of Hearing
	Chinese Alphabet and Dyslexia
	Drug Testing for Beginners
	Einstein's Field Equations using Tensor Analysis
	Methods of Modern Torture
	My Favourite Toy
	Newtonian Mechanics and M-Theory
	Non-Linear Diophantine Approximation Theory
	Rules of Cricket on an Airfield
	Why Imaginary Friends are for Psychopaths

I can use **RANDARRAY** in tandem with **SORTBY** to determine a presentation order for the term:

G93 X ✓ fx =SORTBY(F63:F72,RANDARRAY(COUNTA(F63:F72)))

	C	D	E	F	G	H	I	J	K	L	
89											
90					Presentation Order						
91											
92					Week	Child	Topic				
93					1	Diana	Non-Linear Diophantine Approximation Theory				
94					2	Isla	Drug Testing for Beginners				
95					3	Grace	Einstein's Field Equations using Tensor Analysis				
96					4	Carlo	Newtonian Mechanics and M-Theory				
97					5	Bubba	Algebra for the Hard of Hearing				
98					6	Alex	Why Imaginary Friends are for Psychopaths				
99					7	Felix	My Favourite Toy				
100					8	Horace	Methods of Modern Torture				
101					9	Jules	Chinese Alphabet and Dyslexia				
102					10	Erica	Rules of Cricket on an Airfield				
103											

Oh dear. I do hope Diana has prepared well or it could all end in tears. She could try swapping with Horace, I suppose. On a serious note, the formula

=SORTBY(F63:F72,RANDARRAY(COUNTA(F63:F72)))

sorts the 'Child' order randomly (and a similar formula is used for 'Topic' too). In a past life, as an independent expert, I once had to attest that drug testing was being performed entirely randomly, *i.e.* free from any material bias. **SORTBY(RANDARRAY)** dries up that well for me once and for all.

Death of Data Tables and PivotTables?

I near the end of this rather long article on an interesting note or two. There are some significant ramifications for Excel, once these functions and features roll out and become Generally Available (this does assume the "final" versions of everything highlighted here do not change drastically).

Let me explain.

I begin with a two-dimensional Data Table (**ALT + D + T**) with an old favourite for this sort of thing, calculating monthly payments on various loan amounts over various durations.

G24 X ✓ fx {=TABLE(H12,H13)}

	C	D	E	F	G	H	I	J	K	L	
9											
10					Interest Payments						
11											
12					Loan Amount	\$ 20,000					
13					Term (yrs)	3					
14					Annual Rate (%)	4.50%					
15					Monthly Payment	\$ 594.94					
16											
17											
18					Data Table						
19											
20					Data Table Switch	On					
21											
22											
23						\$ 10,000	\$ 20,000	\$ 30,000	\$ 40,000	\$ 50,000	\$ 60,000
24					1	\$ 853.79	\$ 1,707.57	\$ 2,561.36	\$ 3,415.14	\$ 4,268.93	\$ 5,122.71
25					2	\$ 436.48	\$ 872.96	\$ 1,309.43	\$ 1,745.91	\$ 2,182.39	\$ 2,618.87
26					3	\$ 297.47	\$ 594.94	\$ 892.41	\$ 1,189.88	\$ 1,487.35	\$ 1,784.82
27					4	\$ 228.03	\$ 456.07	\$ 684.10	\$ 912.14	\$ 1,140.17	\$ 1,368.21
28					5	\$ 186.43	\$ 372.86	\$ 559.29	\$ 745.72	\$ 932.15	\$ 1,118.58
29					6	\$ 158.74	\$ 317.48	\$ 476.22	\$ 634.96	\$ 793.70	\$ 952.44
30											

I have no plans to go through Data Tables here, suffice to say they are a great tool for "what-if?" analysis, albeit they can consume vast quantities of memory. This summary table shows how the monthly instalments would vary for different terms (in years) and different amounts borrowed.

Now, take a look at using three Dynamic Array formulae:

G39 X ✓ fx =PMT(Loan_Rate/Months_in_Year,F39#*Months_in_Year,G38#)

	C	D	E	F	G	H	I	J	K	L	
35											
36					Interest Payments Summary Table						
37											
38						\$ 10,000	\$ 20,000	\$ 30,000	\$ 40,000	\$ 50,000	\$ 60,000
39					1	\$ 853.79	\$ 1,707.57	\$ 2,561.36	\$ 3,415.14	\$ 4,268.93	\$ 5,122.71
40					2	\$ 436.48	\$ 872.96	\$ 1,309.43	\$ 1,745.91	\$ 2,182.39	\$ 2,618.87
41					3	\$ 297.47	\$ 594.94	\$ 892.41	\$ 1,189.88	\$ 1,487.35	\$ 1,784.82
42					4	\$ 228.03	\$ 456.07	\$ 684.10	\$ 912.14	\$ 1,140.17	\$ 1,368.21
43					5	\$ 186.43	\$ 372.86	\$ 559.29	\$ 745.72	\$ 932.15	\$ 1,118.58
44					6	\$ 158.74	\$ 317.48	\$ 476.22	\$ 634.96	\$ 793.70	\$ 952.44
45											

Can you spot the difference? In the second table, I have highlighted three cells:

- G38 contains the formula =SEQUENCE(1,6,10000,10000)
- F39 contains the formula =SEQUENCE(6)
- G39 contains the formula =-PMT(Loan_Rate/Months_in_Year,F39#*Months_in_Year,G38#). See how using the Spilled Range Operator ('#') makes all the difference?

That's it! Now I am not saying all Data Tables may be replaced by Dynamic Array formulae, but can you see the future? And guess what, it doesn't stop there. Let me replicate one feature in Excel many of us are familiar with: the PivotTable...

In this illustration, I have created a 1,200-record Table (CTRL + T):

Football Club	Month	Month No	Pts Achieved
Nottingham Forest	January	6	3
Sheffield Wednesday	January	6	3
Ipswich Town	December	5	0
Millwall	March	8	1
Nottingham Forest	September	2	3
Bristol City	February	7	1
Bristol City	March	8	3
Queens Park Rangers	August	1	3
Leeds United	March	8	1
Preston North End	April	9	1
Sheff Wed	January	6	0

1206	Brentford	September	2	0
1207	Aston Villa	March	8	0
1208	Sheffield Wednesday	September	2	3
1209	Brentford	November	4	3
1210	Ipswich Town	March	8	1
1211	Ipswich Town	August	1	3
1212	Birmingham City	November	4	1

It's all made up randomly generated data, and you will just have to guess who I support. The important thing to note is I have created a Table, called **Football_Data**, so I may add records and the Table will extend automatically.

Next, I created a "Pseudo PivotTable":

M13 =SUMIFS(Football_Data[Pts Achieved],Football_Data[Football Club],L13#,Football_Data[Month],M12#)

	August	September	October	November	December	January	February	March	April	May
Aston Villa	13	1	10	7	5	1	4	1	4	10
Birmingham City	8	5	11	11	10	7	4	-	1	6
Blackburn Rovers	14	2	11	17	4	6	2	10	11	6
Bolton Wanderers	3	7	9	3	7	1	6	1	7	3
Brentford	4	12	7	16	2	1	26	12	4	6
Bristol City	7	-	13	5	8	17	12	4	6	13
Derby County	6	11	19	7	9	10	10	10	15	17
Hull City	6	1	9	5	7	-	4	5	6	7
Ipswich Town	5	8	11	10	9	15	1	10	7	8
Leeds United	6	2	3	2	2	3	13	3	3	5
Middlesbrough	8	9	7	7	3	9	2	9	8	4
Millwall	12	4	6	3	10	9	6	13	9	5
Norwich City	14	12	3	12	11	6	9	12	4	6
Nottingham Forest	4	3	8	-	7	12	4	7	13	2
Preston North End	16	4	4	11	2	5	7	12	7	5
Queens Park Rangers	6	6	8	6	7	4	8	16	2	5
Reading	3	12	6	6	8	28	1	2	6	3
Rotherham United	3	9	13	5	11	6	2	9	1	3
Sheffield United	3	4	1	4	9	3	3	3	3	7
Sheffield Wednesday	7	6	3	7	-	14	8	18	7	10
Stoke City	6	6	6	8	3	10	11	7	5	8
Swansea City	6	5	14	3	1	8	3	1	4	9
West Bromwich Albion	6	1	3	7	1	18	7	4	10	4
Wigan Athletic	3	7	-	9	11	9	7	6	7	5

This was created using three Dynamic Array formulae (again, highlighted):

- **M12** contains the formula `=TRANSPOSE(UNIQUE(SORTBY(Football_Data[Month],Football_Data[Month No])))`, which sorts the months into the required order
- **L13** contains the formula `=SORT(UNIQUE(Football_Data[Football Club]))`, which simply sorts the clubs into alphabetical order
- **M13** contains the formula `=SUMIFS(Football_Data[Pts Achieved],Football_Data[Football Club],L13#,Football_Data[Month],M12#)`, which spills out the points earned each month using a standard SUMIFS formula and the Spilled Range Operator ('#').

Think about it. I have created a formulaic PivotTable which calculates no discernibly slower than the real thing. However, the source data may be extended, values may change and I don't need to hit 'Refresh'. Is this the end for PivotTables?

It's easy to get carried away. Dynamic Array formulae make league tables a breeze:

League Table

	P	W	D	L	Pts
Derby County	48	33	15	-	114
Brentford	67	22	24	21	90
Norwich City	56	27	8	21	89
Bristol City	56	21	22	13	85
Ipswich Town	68	18	30	20	84
Blackburn Rovers	56	22	17	17	83
Sheffield Wednesday	56	19	23	14	80
Millwall	54	22	11	21	77
Reading	49	21	12	16	75
Preston North End	57	18	19	20	73
Stoke City	52	19	13	20	70
Queens Park Rangers	40	18	14	8	68
Middlesbrough	47	17	15	15	66
Wigan Athletic	50	16	16	18	64
Birmingham City	52	16	15	21	63
Rotherham United	49	14	20	15	62
West Bromwich Albion	44	17	10	17	61
Nottingham Forest	54	14	18	22	60
Aston Villa	41	15	11	15	56
Swansea City	44	15	9	20	54
Hull City	39	13	11	15	50
Bolton Wanderers	37	12	11	14	47
Leeds United	43	8	18	17	42
Sheffield United	41	10	10	21	40

However, rather than get side tracked, I'd rather stay "on track" with PivotTables and finish this section unpivoting the PivotTable we have just created (the references have changed as they are on a different worksheet in my example):

	1	2	3	4	5	6	7	8	9	10
	August	September	October	November	December	January	February	March	April	May
Aston Villa	13	1	10	7	5	1	4	1	4	10
Birmingham City	8	5	11	11	10	7	4	-	1	6
Blackburn Rovers	14	2	11	17	4	6	2	10	11	6
Bolton Wanderers	3	7	9	3	7	1	6	1	7	3
Brentford	4	12	7	16	2	1	26	12	4	6
Bristol City	7	-	13	5	8	17	12	4	6	13
Derby County	6	11	19	7	9	10	10	15	17	
Hull City	6	1	9	5	7	-	4	5	6	7
Ipswich Town	5	8	11	10	9	15	1	10	7	8
Leeds United	6	2	3	2	2	3	13	3	3	5
Middlesbrough	8	9	7	7	3	9	2	9	8	4
Millwall	12	4	6	3	10	9	6	13	9	5
Norwich City	14	12	3	12	11	6	9	12	4	6
Nottingham Forest	4	3	8	-	7	12	4	7	13	2
Preston North End	16	4	4	11	2	5	7	12	7	5
Queens Park Rangers	6	6	8	6	7	4	8	16	2	5
Reading	3	12	6	6	8	28	1	2	6	3
Rotherham United	3	9	13	5	11	6	2	9	1	3
Sheffield United	3	4	1	4	9	3	3	3	3	7
Sheffield Wednesday	7	6	3	7	-	14	8	18	7	10
Stoke City	6	6	6	8	3	10	11	7	5	8
Swansea City	6	5	14	3	1	8	3	1	4	9
West Bromwich Albion	6	1	3	7	1	18	7	4	10	4
Wigan Athletic	3	7	-	9	11	9	7	6	7	5

Unpivoting can be a nightmare, but it is possible. You don't need to use Dynamic Arrays to do it, but I will to showcase them:

	Club	Month	Points
	Aston Villa	August	13
	Aston Villa	September	1
	Aston Villa	October	10
	Aston Villa	November	7
	Aston Villa	December	5
	Aston Villa	January	1
	Aston Villa	February	4
	Aston Villa	March	1
	Aston Villa	April	4
	Aston Villa	May	10
	Birmingham City	August	8
	Birmingham City	September	5
	Birmingham City	October	11
	Birmingham City	November	11
	Birmingham City	December	10
	Birmingham City	January	7
	Birmingham City	February	4
	Birmingham City	March	-
	Birmingham City	April	1

There is a hidden formula in cell E45. You can see why it is hidden – for those of you with a nervous disposition, please look away now:

=INDEX(SORT(G12#&" - "&F14:F37),ROUNDUP(SEQUENCE(COUNTA(F14:F37)
*COUNT(G12#))/COUNT(G12#),0),MOD(SEQUENCE(COUNTA(F14:F37)*COUNT(G12#))
-1,COUNT(G12#))+1).

Oh dear. That’s a horror. Rather than write 1,000 words trying to explain this, let me detail the concept instead. **SORT(G12#&" - "&F14:F37)** provides every combination of Month Number concatenated with a **Football Club**, separated by a “ – ” delimiter, e.g.

1 – Aston Villa, 2 – Aston Villa, ..., 10 – Aston Villa, 1 – Birmingham City, 2 – Birmingham City, ...

The problem is **SORT(G12#&" - "&F14:F37)** spills this into a 10-column by 24-row array. I want it as a list, so the entire rest of the formula simply forces the array down a column of 240 rows instead. **INDEX** is used to locate the next record in the array, with contrived formulae to determine the row and column numbers of the virtual grid.

SUMIFS is used to create the points total for each row, and to be honest, simpler formulae could have been used elsewhere too. But that’s my point. As I have written this article, it’s hard not to get carried away with all this and try and do everything in Dynamic Arrays. I have worked for years with Excel and been a keen advocate for keeping everything simple. Dynamic Arrays scare me that we may not help ourselves and write monsters like the formula above.

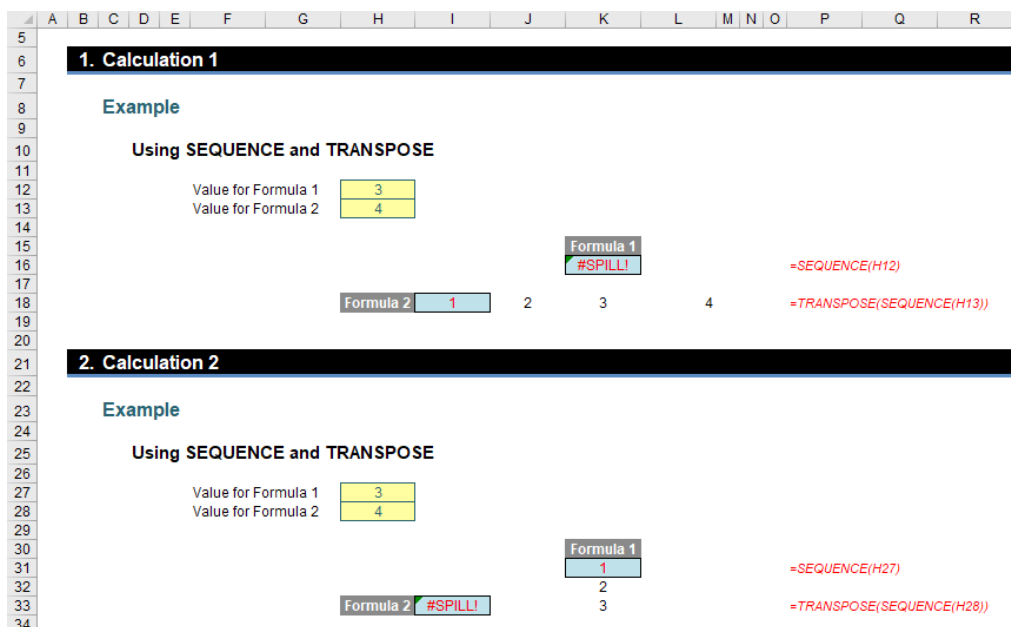
Maybe Excel’s simpler functions and features will live on after all.

Calculation Order Concern

If it feels like you have aged a year since you started reading this, you probably have. There’s a lot to get excited about and I have highlighted some of the issues too – many of which I am sure will be ironed out by the time everything becomes Generally Available. However, I am not

sure the following concern will be going away any time soon.

When I calculate something in Excel, if I use the same formula, I must get the same answer, right? Well – not necessarily. Consider the following



In the example above, Calculations 1 and 2 are identical but deliver different results (i.e. different *#SPILL!* errors). Why?

- In Calculations 1 and 2, both values for Formula 1 and Formula 2 were originally set to 1. This causes no *#SPILL!* errors
- In Calculation 1, the value for Formula 2 (cell H13) was then changed to 4 with no error
- Then, in Calculation 1, the value for Formula 1 (cell H12) was changed to 3. This caused the resultant *#SPILL!* error in cell K16
- Next, in Calculation 2, the value for Formula 1 (cell H27) was changed to 3 with no error
- Then, in Calculation 2, the value for Formula 2 (cell H28) was changed to 4. This caused the resultant *#SPILL!* error in cell I33.

I am not sure what the solution is for this problem. Technically, *#SPILL!* is working correctly, but it doesn’t seem right that two results may be generated in this instance depending upon what input I change first. The jury is out on this one

Word to the Wise

As at the time of writing, all the features, functions and error messages are beta features only. They are available to a portion of Office Insiders at this time, but don’t let that put you off. Start getting excited now! Microsoft will continue to optimise these features over the next several months. This means they *might* change. When they’re fully cooked,

Microsoft will release them out into the wild, first to all Office Insiders and then finally Office 365 subscribers (this is when a feature is known as “Generally Available”).

The future’s looking bright though.

Excel Office 365 Performance Improvements – Phase 2



Just under a year on from last year's improvement frenzy, the Microsoft team have been going gangbusters at "Phase 2". Microsoft states they continue this project "... to reinforce our commitment to fixing top impacting freezing, slow and not responding performance issues derived from Excel user feedback and usage learnings". This is what they have been up to...

- **Lookup functions: VLOOKUP, HLOOKUP, and MATCH** are three of the most used functions in Excel. If you previously used these to locate exact matches to find items in a table or range in Excel, and see it noticeably take time in seconds or minutes or more, chances are you'll now see very noticeable improvement in the speed at which you see results.

Microsoft has made them faster by more efficiently finding the item you are looking up. Essentially, they have created an index on-demand when you first search of a range of cells and then reuse it in subsequent lookups from the same range, until data changes in the lookup range.

- **Copying the entire column in a sheet** and selecting the clipboard is faster now because Microsoft optimised the large memory allocations with a more efficient data structure
- **Undoing pasted cells with conditional formatting** was slow because a paste operation was inefficient in generating an undo record for every cell with conditional formatting and with its own changed priority. It now combines them more efficiently into one undo record with changed priorities for all in it
- **Undoing inserting a new row** with a copied range with merged cells is now faster because Microsoft has optimised handling of merged cells during an undo operation
- **Deleting one or more rows with merged cells** is now more efficient because it will skip redundant calls to render the visible grid in the midst of the deletion operation
- **Viewing filter drop down** for a column with lots of cells with conditional formatting is now faster because the filter drop down algorithm has been made more efficient computing highlight, unique and duplicate values
- **Scrolling in the visible sheet** after an operation wasn't optimal, because Excel re-rendered (or at least evaluated) all rows top to bottom (including filtered rows) in the visible sheet. This can be expensive depending on the number of rows and Excel's ability to calculate for animation related rendering. The optimisation was introduced in the first wave of performance improvements and has been enhanced here
- **Scrolling, editing of cells, cell selection and filtering operations** in the grid are much faster when lots of rows are filtered / hidden because rendering is better optimised to intelligently defer

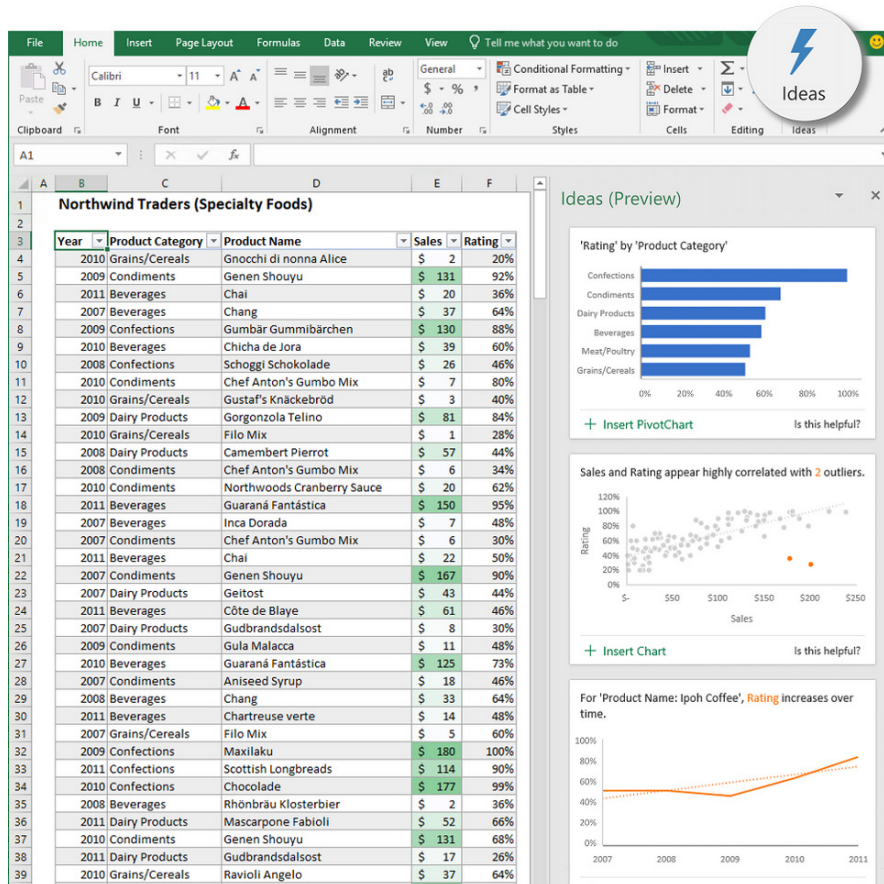
expensive rendering calls until Excel has calculated the last row in the visible range

- **Slow transitioning to the next cell after editing** a cell adjacent to a large table or range of cells with data, because after editing a cell adjacent to a large table or range of cells with data, Excel would generate a preview of all cells in that column to Flash Fill, which is on by default. Depending on the size of the adjacent table or range of cells and whether the cells contain additional metadata like conditional formatting, this can be a time consuming and expensive operation. Microsoft now limits the scope of Flash Fill preview to the visible area improving Excel's responsiveness
- When **opening any workbook** Excel used to search all existing Ribbon content to ensure every component of the Ribbon rendered correctly. This will now open faster by more intelligently searching in common cases like when the Ribbon item being searched for doesn't exist. Also, during open Excel now skips updating MRU links synchronously and instead update them asynchronously after opening. Finally, the software scopes the rendering of the grid to the visible grid area when opening workbooks with lots of wrapped text in locales like Japanese
- **Opening a local workbook when using a third party anti-virus software is faster** because Excel is now more intelligent in avoiding searching for the third party anti-virus vendor registered for scanning when a scan is determined to be redundant
- Excel now will **open simple CSV text files much faster** by employing a more efficient memory allocation
- **Flashing scrollbars and slower running VBA code (than Excel 2010) even when ScreenUpdating property is set to false:** Microsoft has stated they have addressed this by ensuring scrollbars will no longer update when the ScreenUpdating property is set to false (just like in Excel 2010). This in turn diverts CPU cycles away from redundant rendering of the scrollbars and to running user VBA code, improving code execution speed
- Programmatically **selecting a cell, a range of cells, or a sheet using VBA** is now faster because Excel avoids redundant calls for updating a document's upload status when running VBA macro code performing operations like selecting a cell, a range of cells or a sheet
- **Executing the FREQUENCY function for large data sets is faster** because Microsoft has optimised the underlying sorting algorithm for the **FREQUENCY** function for large data sets.

These updates should be noticed by anyone receiving the Office 365 Monthly Channel updates (or better) starting with version 1809 (a fine vintage!).

New Ideas: Insights Re-Badged

It's finally gone "Generally Available" – Ideas, the Artist Formerly Known as "Insights" – is now introduced in Excel to Win32 and Mac, and is apparently coming to Excel Online soon too. We take issue with this official line though as it seems to us that it's working fine and dandy in Win64 as well.



According to Bill Jelen, Ideas is the "... first icon added to the 'Home' tab in 11 years, 8 months, 23 days. It must really be something special". Well, we can soon find out as it mirrors Power BI's 'Quick Insights'.

Here at SumProduct, we agree that understanding data can be quite a challenge, especially if the volume of data is large or if it's an unfamiliar

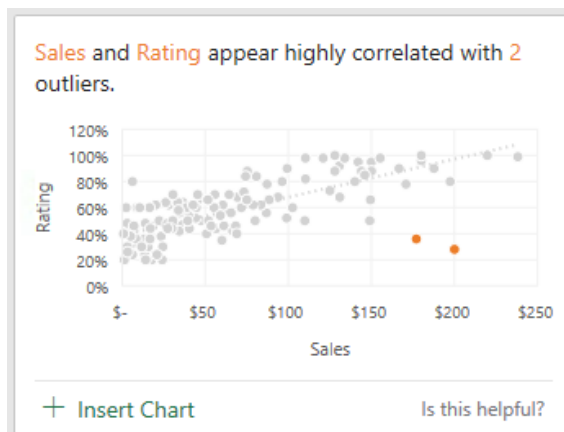
dataset. With Ideas (previously known as Insights), Excel automatically points out interesting patterns in user data, like trends and outliers. Ideas in Excel shows high-level summaries, statistically significant findings and recommended visualisations. Ideas also helps you leverage Excel PivotTables and PivotCharts (if you still need them, given our discussion on Dynamic Arrays!).

It's not just that it's been renamed and become "Generally Available". There's been some updates too:

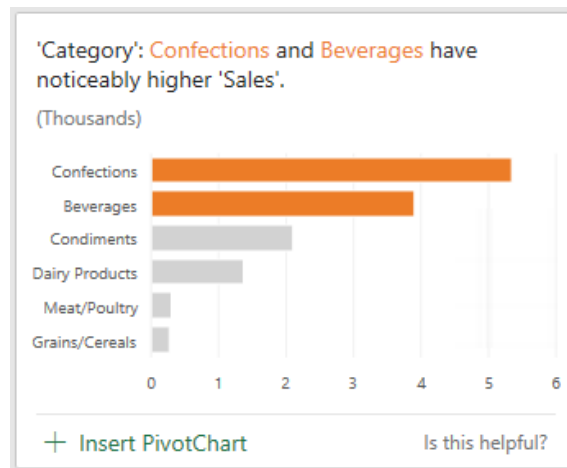
- Easier to understand chart previews
- New analysis types including scatter charts and outliers
- Updated machine learning models to make more powerful suggestions
- Now available in English, French, Spanish, German, Simplified Chinese and Japanese.

To get a feel for Ideas, here's a couple of examples:

- There is now support for scatter charts – something we think really should have been in from the beginning

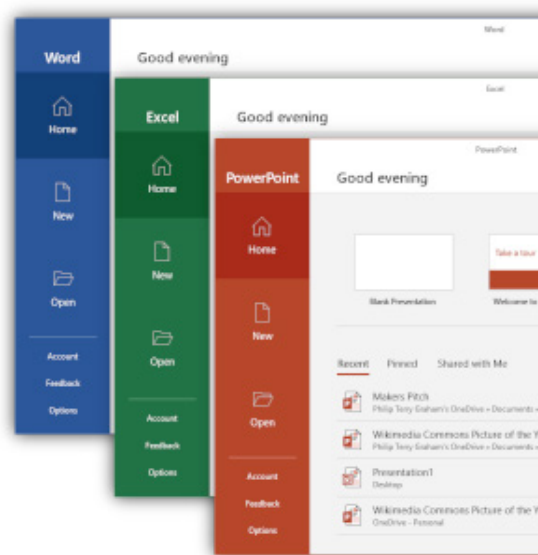


- Microsoft has added something they have confusingly called “subtitles”, to include additional supporting information to help better understand suggestions



Why not have a play..? This feature is being made available to customers on a gradual basis over several weeks. It will first be available in English, French, Spanish, German, Simplified Chinese and Japanese to Office Insider participants on Windows and Mac, and later to Office 365 subscribers. If you are an Office 365 subscriber, make sure you have the latest version of Office.

Office 2019 Commercially Available Now



Succeeding its “2016” counterpart, Microsoft Office 2019 was released for Windows 10 and on macOS on 24 September 2018. Some – but crucially, not all – features that were previously only on Office 365 products are available in this release. It’s clear what Microsoft wants you to buy, which was also identifiable with how awkward it was for the “average user” to get their mitts on a Preview release.

Whilst both include what Microsoft refers to as “classic” versions of Word, Excel, PowerPoint and Outlook (anyone remember Classic Coke?), the Windows version also includes Publisher 2019, Access 2019, Project 2019 and Visio 2019. However, as expected, Office 2019 applications will

not receive feature updates but will receive regular security and stability updates.

People who use Office 365, Office 2016, Office 2013 and Office 2010 applications can open documents created by using Office 2019 without any additional action, i.e. there is backward compatibility, although clearly new functions like **CONCAT** in Excel will not be recognised in previous editions.

There are new features slated for release. The new features that are included in the commercial Preview release, as stated by Microsoft, include:

Word

- Black theme
- Office sounds
- Learning tools captions and audio descriptions
- Text-to-speech
- Improved inking functionality
- Accessibility improvements

Excel

- Funnel charts and 2-D maps
- New Excel functions and connectors
- Publish Excel to Power BI
- Power Pivot enhancements
- Power Query enhancements

PowerPoint

- Zoom capabilities for ordering of slides within presentations
- Morph transition feature
- Insert and manage Icons, SVG and 3-D models
- Improved roaming pencil case

Outlook

- Updated contact cards [Ed.'s note: This feature requires an Exchange Online account.]
- Office 365 Groups
- @mentions
- Focused inbox
- Travel and delivery summary cards

It's not perhaps as many as people were expecting though, and worse, there are features that, at this time of writing, do appear to be missing / excluded:

Unlocks Creativity

- Editor in Word
- Tap in Word, PowerPoint, Outlook
- Designer in PowerPoint
- Researcher in Word
- Insights in Excel*
- Data Types

Built for Teamwork

- Real-time co-authoring
- @mentions in Word, Excel, PowerPoint

Integrated for Simplicity

- Shared computer licensing
- Language packs included
- FastTrack Options
- Intune integration
- Microsoft 365 Analytics*

Intelligent Security

- ATP Safe Links
- Office 365 Message Encryption*
- Office Enterprise Protection*
- Add sensitivity label in Word, Excel, PowerPoint and Outlook*.

The items with an asterisk () require an Office 365 E3 or E5 subscription.*

Perpetual licences are needed by some individuals and organisations who have confidentiality requirements, security concerns or other legitimate reasons not to move to an internet-based subscription model. Microsoft has stated:

"...Moving to the cloud is a journey with many considerations along the

way. Therefore, we remain committed to on-premises customers and plan to do (sic) additional releases post Office 2019..."

So there are further "Perpetual" releases slated, presumably on the continuing three-year cycle, but who knows for sure? It is obvious though that the push towards the subscription model has never been made clearer.

New Data Types and FIELDVALUE Function Now Generally Available (Sort Of)





We first mentioned this in May as a Preview feature, but now it's being rolled out to all users of Excel in Office 365 (in the English language only) starting back in late September.


Excel has always been great at helping people make the most of numbers. But now Excel can do even more: it can recognise real-world concepts, starting with Stocks and Geography. This new AI-powered capability turns a single, flat piece of text into an interactive property (known as an “entity”) containing layers of “data rich” information. For instance, by converting a list of countries in a workbook to “Geography” entities, customers can weave location data into an analysis of their own data.

This will be the start of new data types over time, allowing Excel’s rows, columns, cells, logic engine and tools to be used to organise, analyse and reason over any combination of numbers and sophisticated entities. The Stocks and Geography data types are generally available today.





Here’s a brief summary of these two new data types (first detailed in our May newsletter):


Data Type 1: Stocks

	A	B	C
1	Company	Price	Change
2	 JM Smucker Co	\$ 126.04	\$ 1.19
3	 Kellogg Co	\$ 66.29	\$ (0.73)
4	 Post Holdings Inc	\$ 76.76	\$ 0.73
5	 Tyson Foods Inc	\$ 74.29	\$ 0.53
6			

In this graphic, the cells with company names in column A contain the ‘Stocks’ data type. You may tell this because they have this icon . The ‘Stocks’ data type is connected to an online source that contains more information. Columns B and C are extracting that information. Specifically, the values for price and change in price are getting extracted from the ‘Stocks’ data type in column A.

Data Type 2: Geography

	A	B	C
1	Country	Population	Gasoline price
2	 France	66,896,109	\$1.79
3	 Spain	46,443,959	\$1.63
4	 Sweden	9,903,122	\$1.82
5	 Norway	5,232,929	\$2.27
6			

In this example, column A contains cells that have the ‘Geography’ data type. This time, the  icon indicates this. This data type is connected to an online source that contains more information. Columns B and C are again extracting that information. Specifically, the values for population and gasoline price are getting extracted from the ‘Geography’ data type in column A.

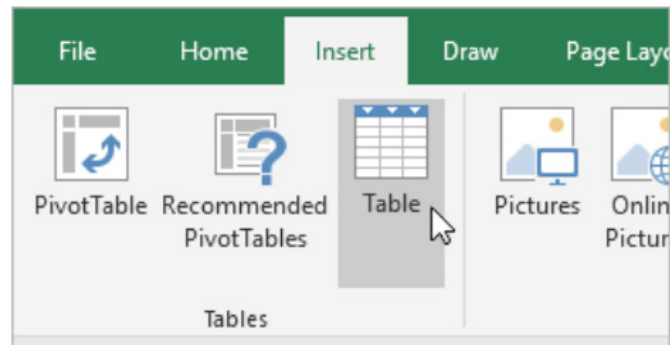
We are sure you can see how these data types might be useful. So how do you set it up? Assuming you have had your version of Excel 2016 updated, it’s as easy as, er, 1, 2, 3, 4, 5, 6...

Step 1: Type Some Text

	A	B	C	D
1	Country			
2	France			
3	Spain			
4	Sweden			
5				
6				
7				

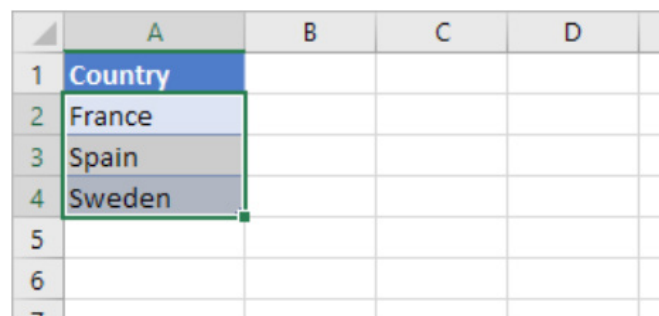
That’s right: just type some text in cells. In this example, we wanted data based on geography, so we typed country names. However, you could also have typed province names, territories, states, cities, etc. If you want stock information, similarly type company names, fund names, ticker symbols, and so on.

Step 2: Create a Table



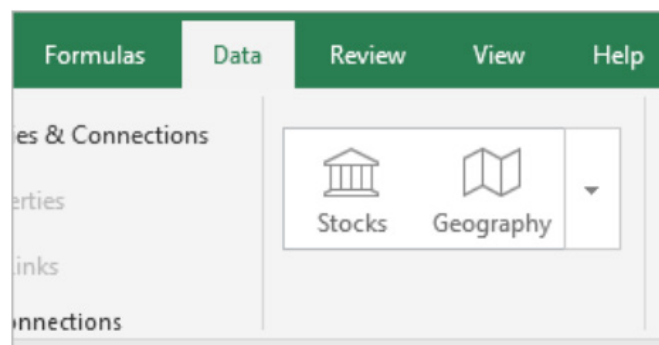
Although it's not required, it is recommended you create an Excel Table. This is so ranges may be extended readily and easily later, should you wish. Select any cell in your data and go to **Insert > Table** or use the keyboard shortcut **CTRL + T**. This will make extracting online information easier later on.

Step 3: Select Some Cells



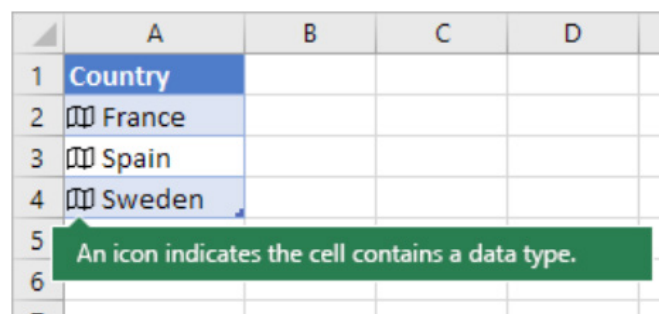
Next, select the cells that you want to convert to a data type.



Step 4: Pick a Data Type



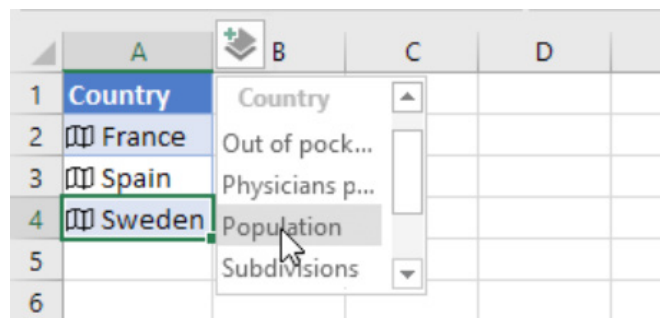
On the 'Data' tab, click either 'Stocks' or 'Geography'.


Step 5: Icons Appear





If Excel finds a match between the text in the cells and the online sources, it will convert your text to either the 'Stocks' data type or 'Geography' data type. You will know immediately if they have been converted since they will have the  icon for stocks and the  icon for geography.

Step 6: Add a Column



Click the 'Add Column' button , and then click a field name to extract more information, such as 'Population'.

If you see the  symbol instead of an icon, then Excel is having difficulty matching your text with data in Microsoft's online sources. Go back and review your data. Correct any spelling mistakes and when you press ENTER, Excel will do its best to find matching information. If this does not work, click  and a 'Selector' pane will appear. Search for data using a keyword or two, choose the data you want and then click 'Select'. That's it!

How to Write Formulae that Reference Data Types

You can use formulae that reference linked data types. This allows you to retrieve and expose more information about a specific linked data type. For example, consider the linked data type, Stocks, used in cells

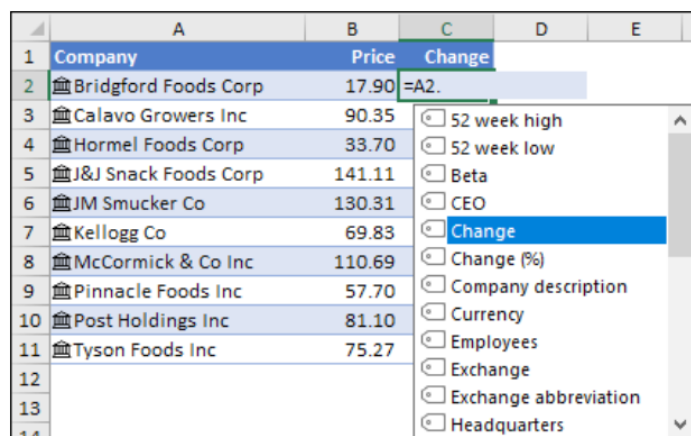
A2:A11 below. In columns **B** and **C**, there are formulae that extract more information from the 'Stocks' data type in column **A**, viz.

	A	B	C
1	Company	Price	Change
2	 Bridgford Foods Corp	17.90	-3.09
3	 Calavo Growers Inc	90.35	2.35
4	 Hormel Foods Corp	33.70	-0.06
5	 J&J Snack Foods Corp	141.11	-0.69
6	 JM Smucker Co	130.31	0.39
7	 Kellogg Co	69.83	-0.42
8	 McCormick & Co Inc	110.69	0.15
9	 Pinnacle Foods Inc	57.70	0.16
10	 Post Holdings Inc	81.10	0.26
11	 Tyson Foods Inc	75.27	-0.42

In this example, cell **B2** contains the formula **=A2.Price** and cell **C2** contains the formula **=A2.Change**. When the records are in a Table, you can use the column names in the formula instead. In this case, cell **B2** would contain the formula **=[@Company].Price** and cell **C2** would contain **=[@Company].Change**. The additional benefit is that these formulae would automatically copy down too.

Some tips for when you start playing with these two new data types:

- as soon as you type the dot operator (.) after a cell or column reference, Excel will present you with a formula AutoComplete list of fields that you can reference for that data type. Select the field you want from the list or type it if you know it



- data type field references are not case sensitive, so you can enter `=A2.Price`, or `=A2.price`
- if you select a field that has spaces in the name, Excel will automatically add brackets ([]) around the field name, e.g. `=A2.[52 Week High]`
- the **FIELDVALUE** function can also be used, but it is recommended only for creating conditional calculations based on linked data types.

This last point is a nice lead into the other new item...

The FIELDVALUE Function

You can use the **FIELDVALUE** function to retrieve field data from linked data types like the 'Stocks' or 'Geography' data types. There are easier methods for writing formulae that reference data types (*see above*), so the **FIELDVALUE** function should be used mainly for creating conditional calculations based on linked data types.

Similar to the new data types, this brand-new function is being made available to customers on a gradual basis over several days or weeks. It will first be available to Office Insider participants and later to Office 365 subscribers. If you are an Office 365 subscriber, make sure you have the latest version of Office or you may not get the update when it's your turn.

Technical details

Syntax

`=FIELDVALUE(value, field_name)`

The **FIELDVALUE** function syntax has the following arguments:

- **value**: this is the cell address, table column or named range that contains a linked data type
- **field_name**: this is the name or names of the fields you would like to extract from the linked data type.

Description

The **FIELDVALUE** function returns all matching fields(s) from the linked data type specified in the value argument. This function belongs to the **Lookup & Reference** family of functions.

Examples

In the following basic example, the formula, `=FIELDVALUE(A2,"Price")` extracts the **Price** field from the 'Stock' data type for the "internationally-renowned" *JM Smucker Co*.

	A	B	C	D	E	F
1	Company	Price	Change			
2	JM Smucker Co	130.31	0.39			
3						
4						

The next example is a more typical example for the **FIELDVALUE** function. Here, we've used the **IFERROR** function to check for errors. If there isn't a company name in cell **A2**, the **FIELDVALUE** formula returns an error, and in that case, displays nothing (""). However, if there is a company name, then we want the formula to retrieve the **Price** from the data type in **A2** with `=IFERROR(FIELDVALUE($A2,B$1),"")`.

	A	B	C	D	E	F
1	Company	Price	Change			
2	JM Smucker Co	130.31	0.39			
3						
4						

Note that the **FIELDVALUE** function allows you to reference worksheet cells for the **field_name** argument, so the above formula references cell **B1** for **Price** instead of manually entering "Price" in the formula.

Word to the Wise

If you try to retrieve data from a non-existent data type field, the **FIELDVALUE** function will return the new **#FIELD!** error. For instance, you might have entered "Prices", when the actual data type field is named "Price". Double-check your formula to make sure you've used a valid field name. If you want to display a list of field names for a record, select the cell for the record and press **CTRL + SHIFT + F2**.

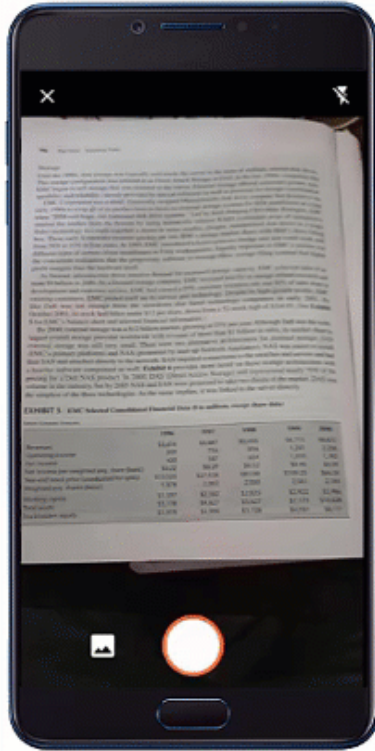
Happy playing with your new toys!

FIELDVALUE will be detailed in our blog series *A to Z of Excel Functions* in due course. You can find out about other functions in our current list [here](#).

Insert Data from Picture

There's another new feature now out – Insert Data from Picture – one of several new products to come out of Microsoft's recent advances in Artificial Intelligence. You can take a picture of a hand-drawn or printed data table with your Android device and convert that analogue information into an Excel spreadsheet with a single click. New image

recognition functionality automatically converts the picture to a fully editable table in Excel, eliminating the need for you to manually enter data. Insert Data from Picture will be available in Preview for the Excel Android app soon.



You must remember that "available in Preview," is Microsoft code for "it's not necessarily the final version of a product or feature". This feature will presumably be coming to the iOS version of Excel as well at some point, but there are no details yet.

Visual Basics

It's not all about the Power Tools, you know. We thought we'd run a new elementary series going through the rudiments of Visual Basic for Applications (VBA) as a springboard for newer users. This month, we actually look at recording a macro.

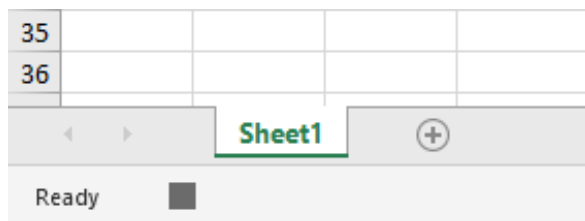
Last month we talked through the four ways of recording a macro. Let's assume we've got that initiated. After pressing 'Record Macro', the 'Record Macro' dialog box will appear. Let's start by giving the macro a name and description and press 'OK':

A screenshot of the 'Record Macro' dialog box in Excel. The dialog box has a title bar with a question mark and a close button. It contains the following fields:

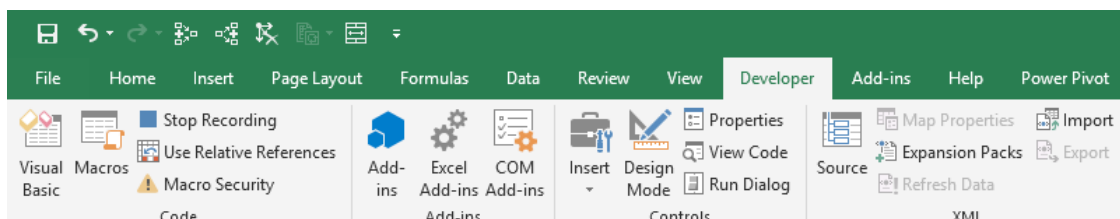
- Macro name:** A text box containing 'Test1'.
- Shortcut key:** A label 'Ctrl+' followed by an empty text box.
- Store macro in:** A dropdown menu showing 'This Workbook'.
- Description:** A text box containing 'My First Macro'.

At the bottom right, there are 'OK' and 'Cancel' buttons.

After hitting 'OK' the recording mode will start in Excel. Looking back at the status bar in Excel, the icon next to 'Ready' has changed into a stop button:



Pressing the stop button will stop the recording of the macro. 'Stop Recording' will also replace the 'Record Macro' buttons on the 'Developer' and 'View' tabs, as will the keyboard shortcut **ALT + W + M + R**.



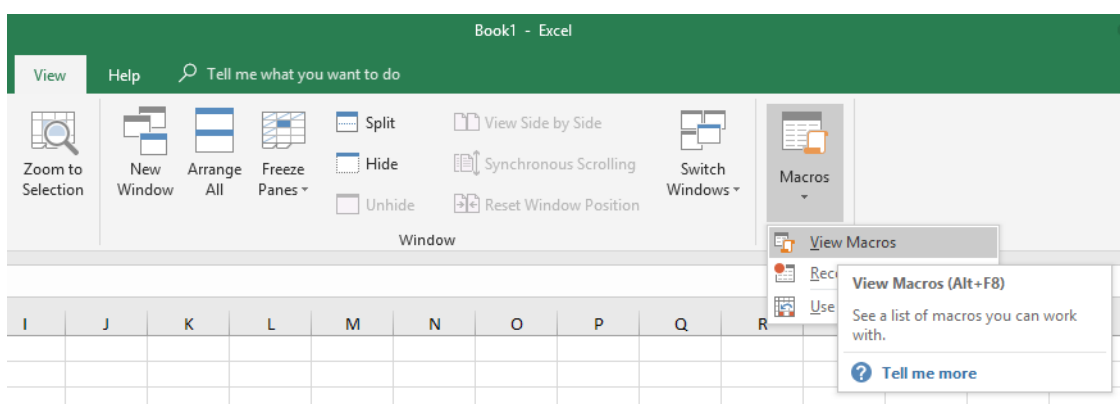
For our first example macro, let's perform the following actions

- Clicked on the 'Record Macro' button
- Fill cell **A5** red
- Copy cell **A5** to **B3**
- Type "colour" in cell **C6**
- Click on the 'Stop recording' button.

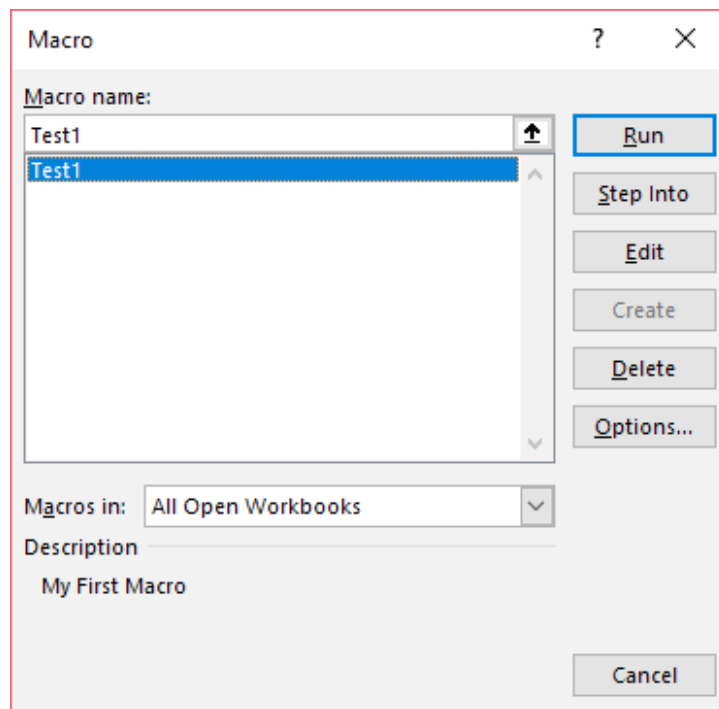
	A	B	C	D
1				
2				
3				
4				
5				
6			colour	
7				
8				

That's our first macro recorded!

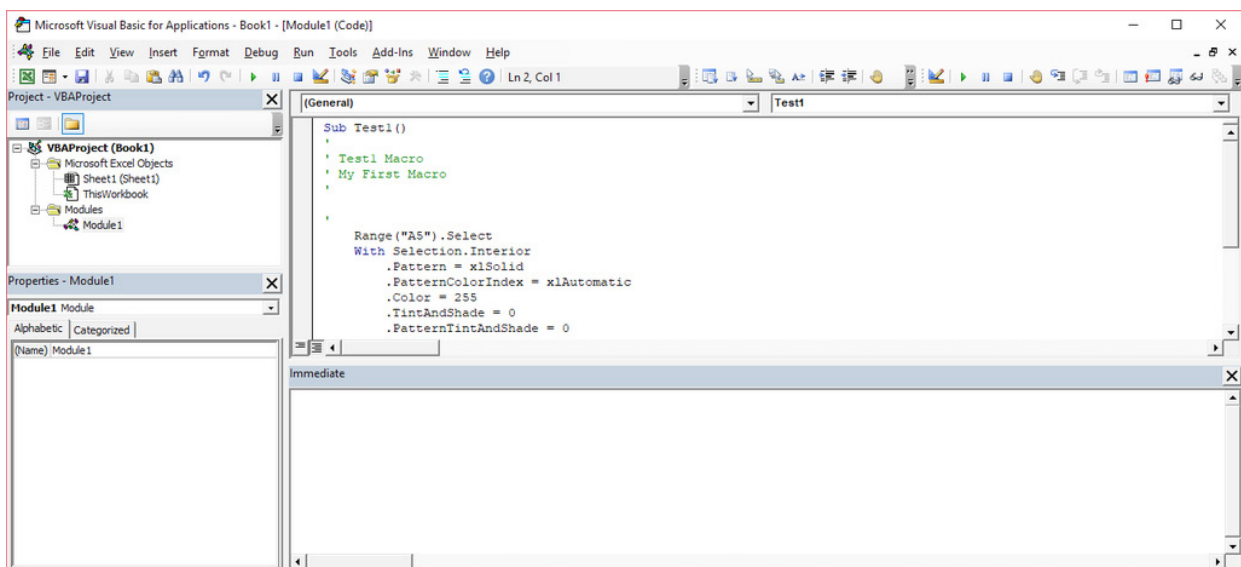
Going back to the 'Macros' button in the Ribbon and select 'View Macros':



A list of all the macros in the workbook will come up including our macro that we have just recorded.



on there we can press 'Edit' to go to the VBA Editor which will snap to the newly written macro:



We can see that we've got some text in the top right, and that's the code for our macro:

```
Sub Test1()
'
' Test1 Macro
' My First Macro
'
'
'
With Selection.Interior
    .Pattern = xlSolid
```

```

.PatternColorIndex = xlAutomatic

.Color = 255

.TintAndShade = 0

.PatternTintAndShade = 0

```

End With

Selection.Copy

Range("B3").Select

ActiveSheet.Paste

Range("C6").Select

Application.CutCopyMode = False

ActiveCell.FormulaR1C1 = "colour"

Range("C7").Select

End Sub

Recording macros is great when you are unsure how to program a specific task, making it easier to look up the relevant syntax to code up more complex procedures. However, there are certain drawbacks.

Note the comments (in green with apostrophes) – this is how you can “document” your procedures, i.e. tell other people what your code is doing. You should always try to remember to do this, especially as you become more experienced and your macros become more complex.

The inherent ‘Relative Referencing’ setting allows recording for things to occur in relation to the current cell. If the ‘Relative Referencing’ setting is

not used, it will hard code the direct cell reference in the code which might not be the intention. This setting is activated on the ‘Developer’ tab in the Ribbon.

The recording also produces extra lines indicating the movement of the screen. In the code example above, every time the mouse moved to select a cell like **Range("C6").Select**, this action was also recorded, however this isn't strictly necessary in terms of achieving the outcome of this routine.

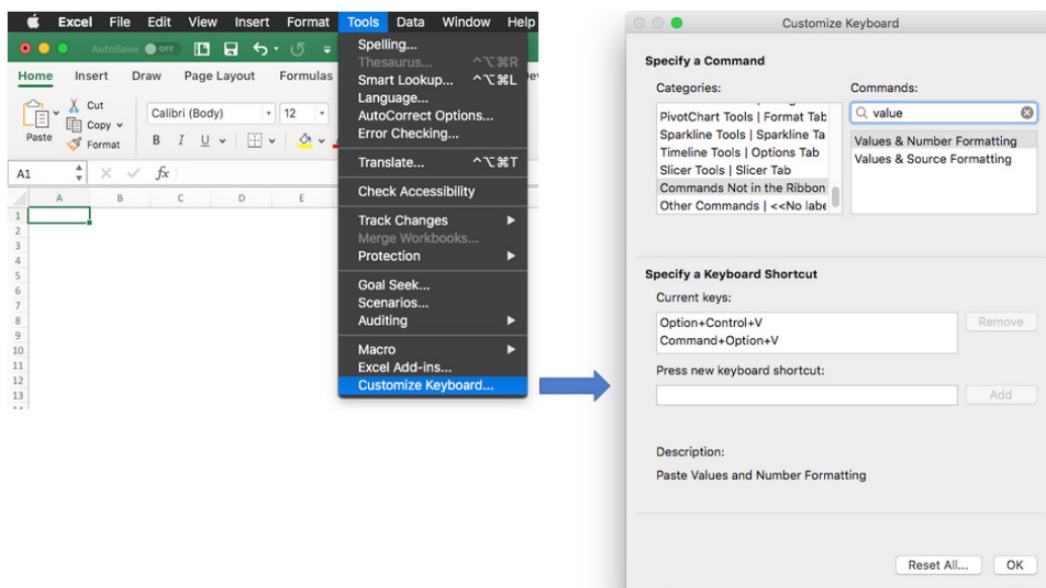
More next month.

Customise Your Keyboard Shortcuts in Excel for Mac

Slowly but surely, Excel for Mac comes into the light. Another step in the right direction has just been made with the advent of the ‘Customize Keyboard’ feature. To get this update, go to **Help > Check for Updates**, and update to version 16.18 or better.

With this feature installed, you may now assign your own key combinations to many commands within Excel. Just go to the ‘Tools’ menu and choose

‘Customize Keyboard’. Pick the category of command you’re trying to find and then search or browse for the command in the list. Select a command, press a key combination and see if it’s used already. If it’s already in use, you may want to pick a different key combination. Once chosen, just click the ‘Add’ button to assign the key combination to the selected command.



Specifically, to create a keyboard shortcut in Excel for Mac:

- on the 'Tools' menu, click 'Customize Keyboard'
- in the 'Categories' list, click a tab name
- in the 'Commands' list, click the command that you want to assign a keyboard shortcut to
- any keyboard shortcuts that are currently assigned to the selected command will appear in the 'Current keys' box
- if you prefer to use a different keyboard shortcut, add another shortcut to the list and then use it instead
- in the 'Press new keyboard shortcut' box, type a key combination that includes at least one modifier key (**⌘**, **CONTROL**, **OPTION** or **SHIFT**) and an additional key, such as **⌘ + F11**.
- if you type a keyboard shortcut that is already assigned, the action assigned to that key combination appears next to 'Currently assigned to'
- click 'Add'
- to cancel the keyboard shortcut assignment, simply press 'ESC'.

To delete a custom keyboard shortcut, note that you may only delete keyboard shortcuts that you created – you cannot delete the default keyboard shortcuts for Excel. The steps are as follows:

- on the 'Tools' menu, click 'Customize Keyboard'
- in the 'Categories' list, click a tab name
- in the 'Commands' list, click the command that you want to delete a keyboard shortcut from
- in the 'Current keys' box, click the keyboard shortcut that you want to delete and then click 'Remove'
- if the 'Remove' button appears greyed out, then the selected keyboard shortcut is a default keyboard shortcut and therefore it cannot be deleted.

If you stuff it all up, you can't do any real damage as you may simply reset all keyboard shortcuts as follows:

- on the 'Tools' menu, click 'Customize Keyboard'
- to restore keyboard shortcuts to their original state, click 'Reset All'.

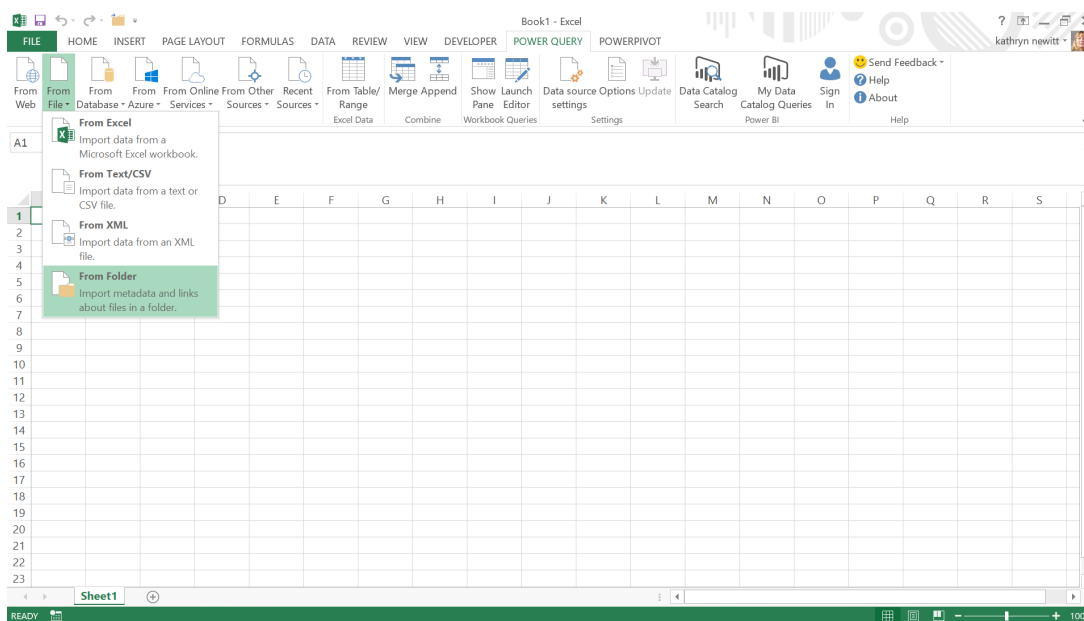
Finally, it should be noted that the keyboard shortcut descriptions refer to the US keyboard layout. Keys on other keyboard layouts might not correspond to the keys on a US keyboard. Keyboard shortcuts for laptop computers might also differ. But hey, it's getting there!

Power Query Pointers

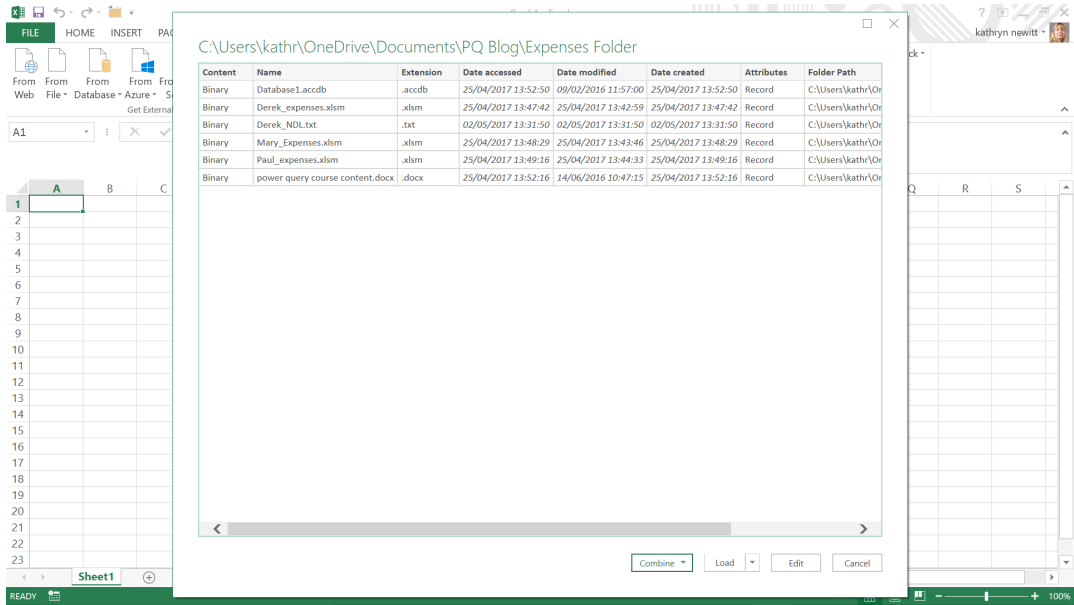
Each month we'll reproduce one of our articles on Power Query (Excel 2010 and 2013) / Get & Transform (Excel 2016) from www.sumproduct.com/blog. If you wish to read more in the meantime, simply check out our Blog section each Wednesday. This month, we look at combining data from tables in Excel worksheets in other workbooks (again), with less steps thanks to the recent Combining Files improvements.

In previous articles, we have combined data from tables in workbooks that were gathered into one folder. You may recall that using combining binaries produced an error, so we resorted to a more manual method. This time, armed with an updated version of Power Query, let's try it a different way.

We'll start as before, by getting a list of the workbooks that contain the data we wish to aggregate. To do this, let's start in a new workbook and create a blank query which is 'From Folder' as shown below:

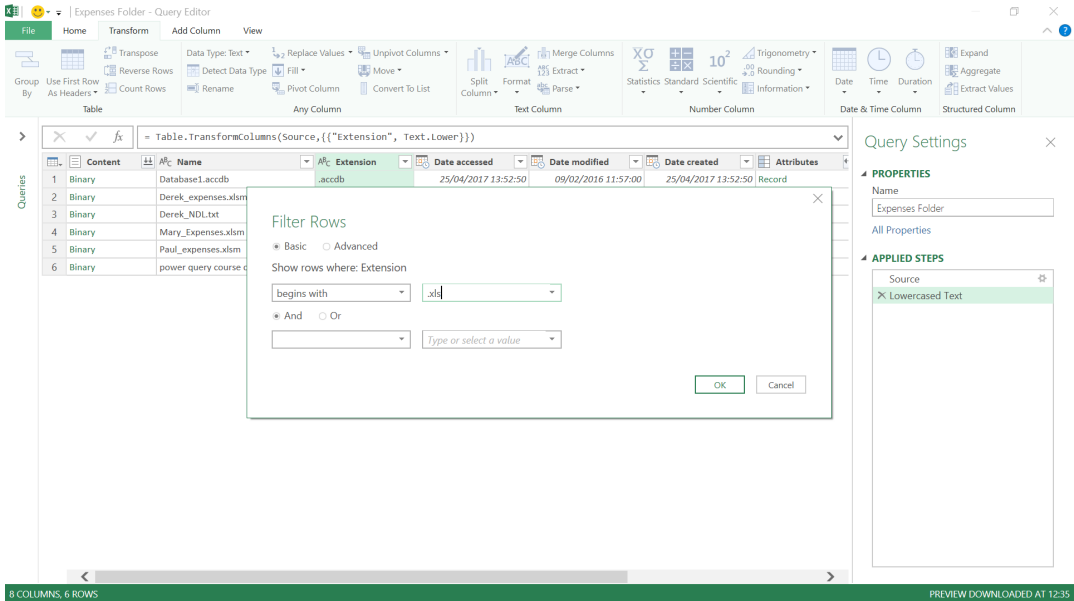


Let's create a folder with our expense worksheets and some other items lurking around.

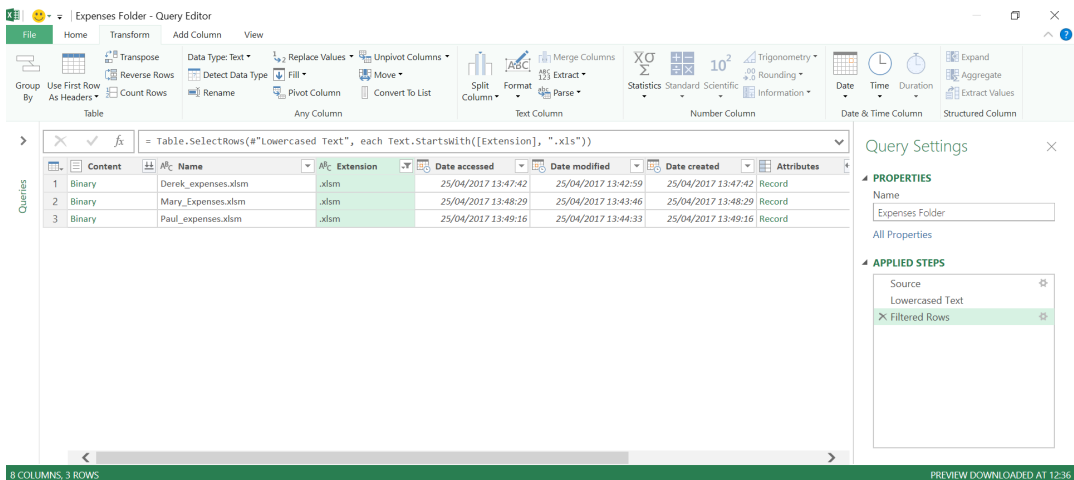


We don't want to attempt to combine at this point as some of those files are not expenses! The next step is to edit the query to filter out the other files so that we are left with the workbooks. We also want to make sure that any expense workbooks added in future would be picked

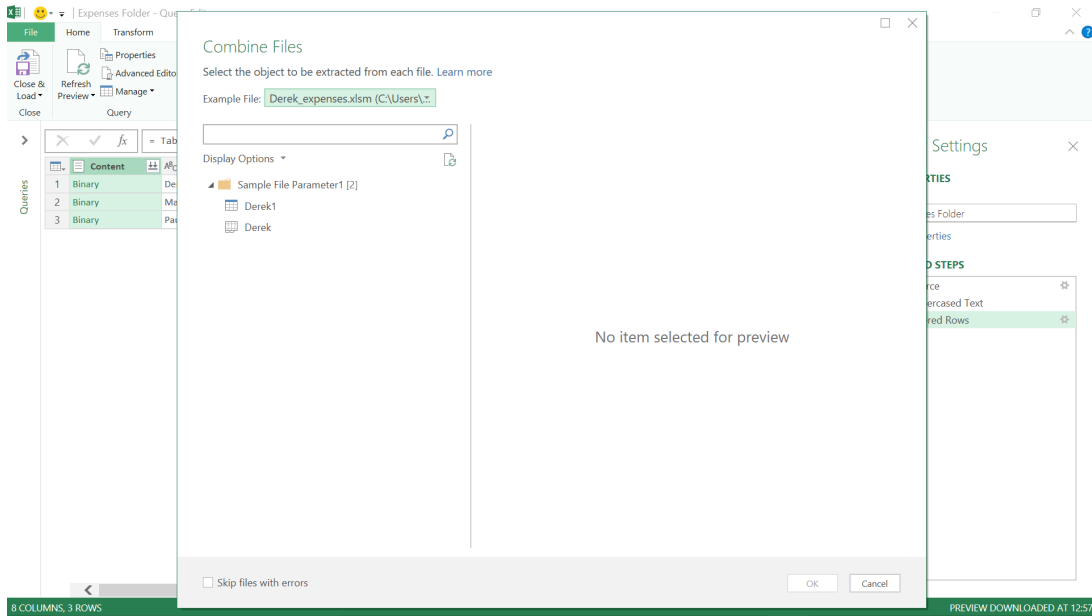
up. So that any other workbooks would also be saved regardless of the extension, let's transform the case to 'lowercase' by right-clicking on the **Extension** column and choosing the 'Transform' option. We can then filter to pick up any extensions that start with '.xls':



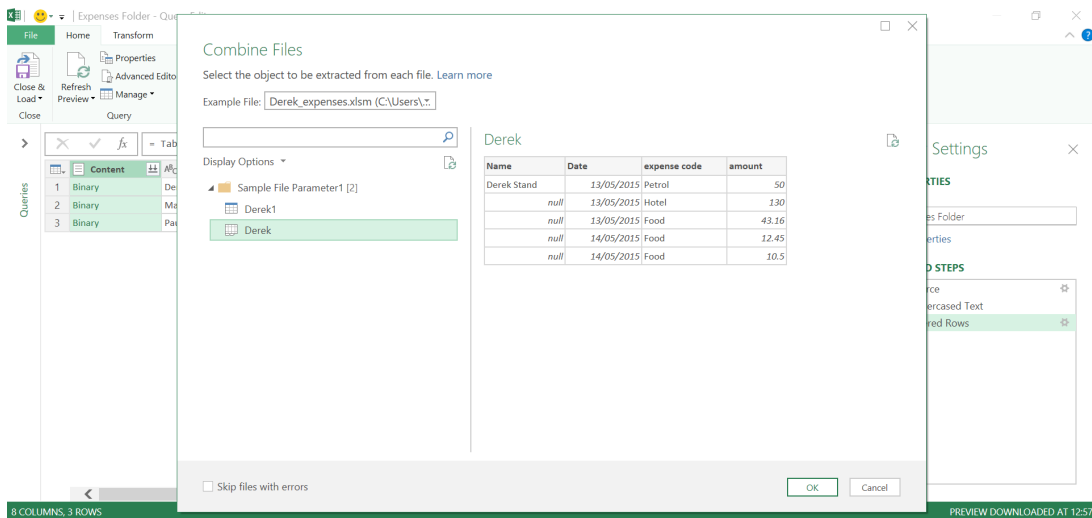
This leaves us with the files that we want.



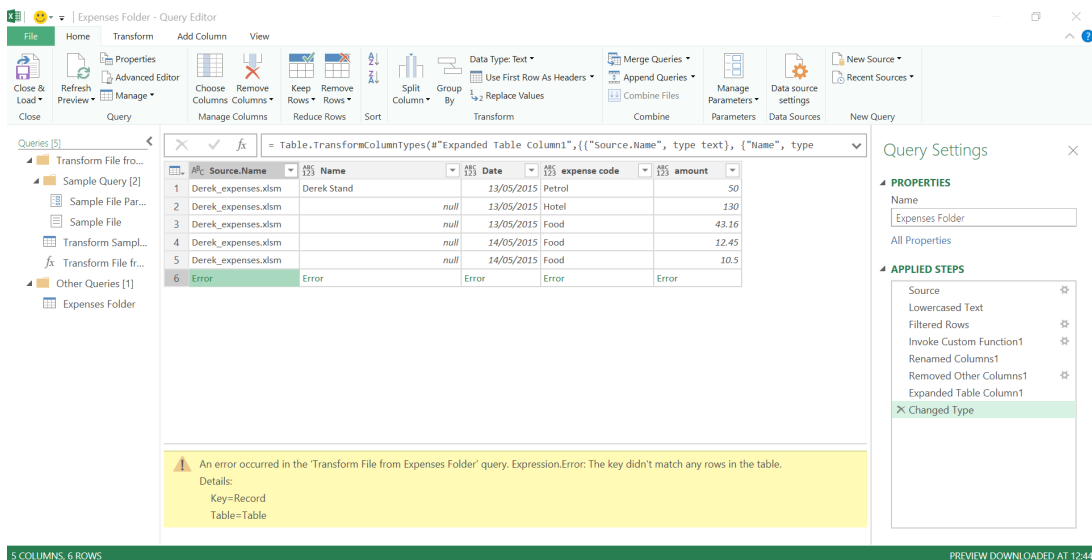
Power Query continues to be tweaked. The icon is still next to **Content**, but now instead of being called a 'Combine Binaries' button, it is a 'Combine Files' button. Pressing it begins the process, which shows us what is available in the first file:



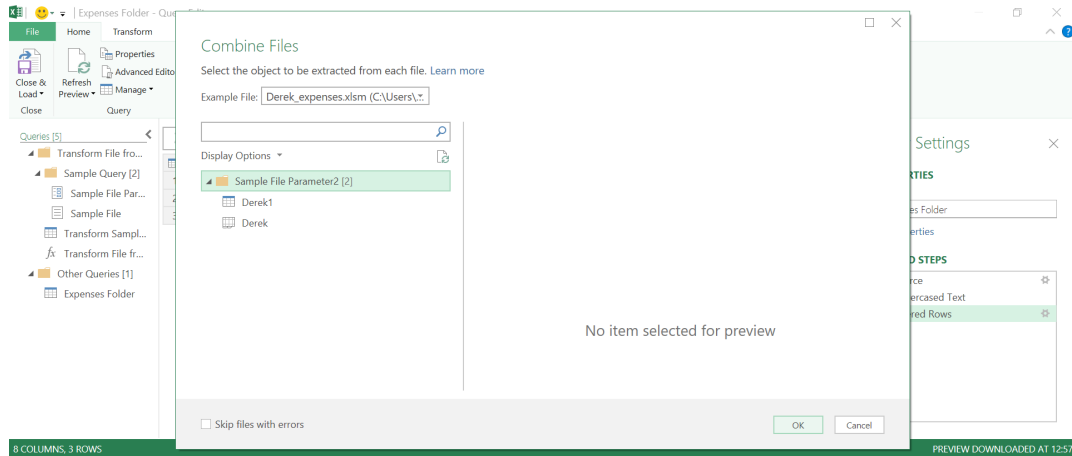
The options we have are to choose the table, the sheet or the parameters. We also have the option to 'Skip files with errors' which is a clue that the error handling has been improved now. We choose the one that looks like a sheet to see if it works now.



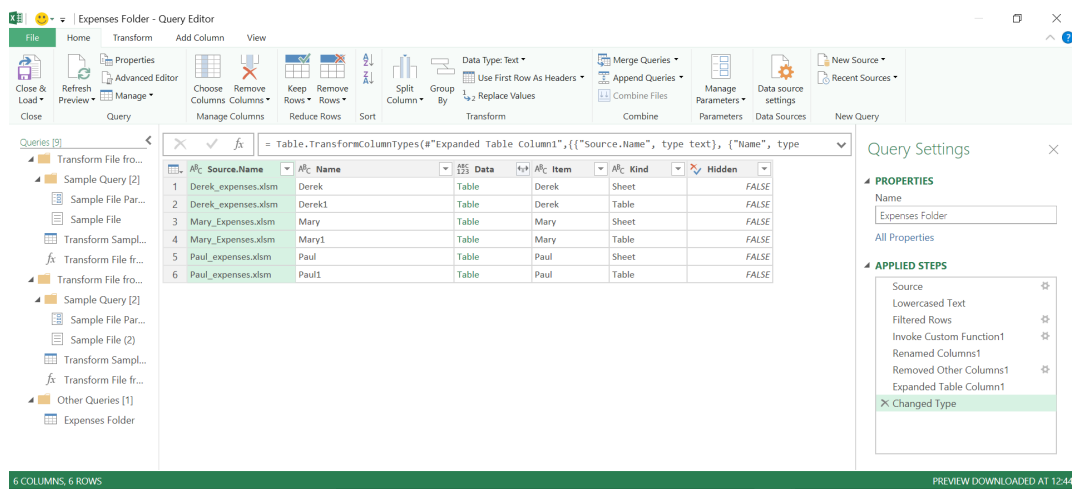
Click 'OK' to combine the sheets.



Like before, Power Query can't see how to link the data together, and a similar thing happens if we pick the table. There is however a third option, so let's delete all the steps so that we are back at 'Filtered Rows', and use the 'Combine Files' button again – this time we'll pick the sample folder option:

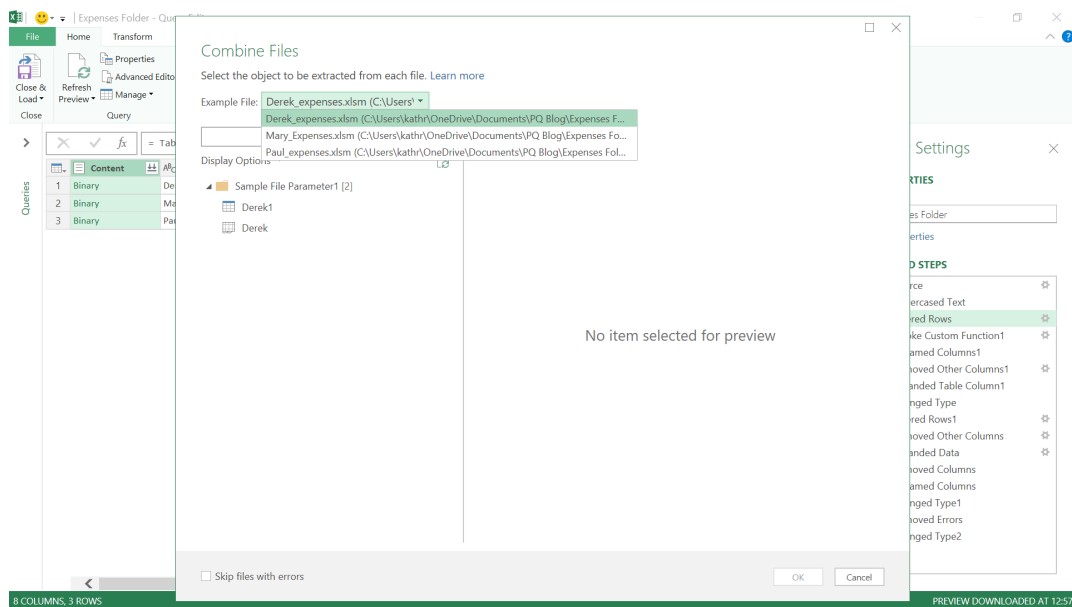


We have no option to preview anything this time, but when we press 'OK'...

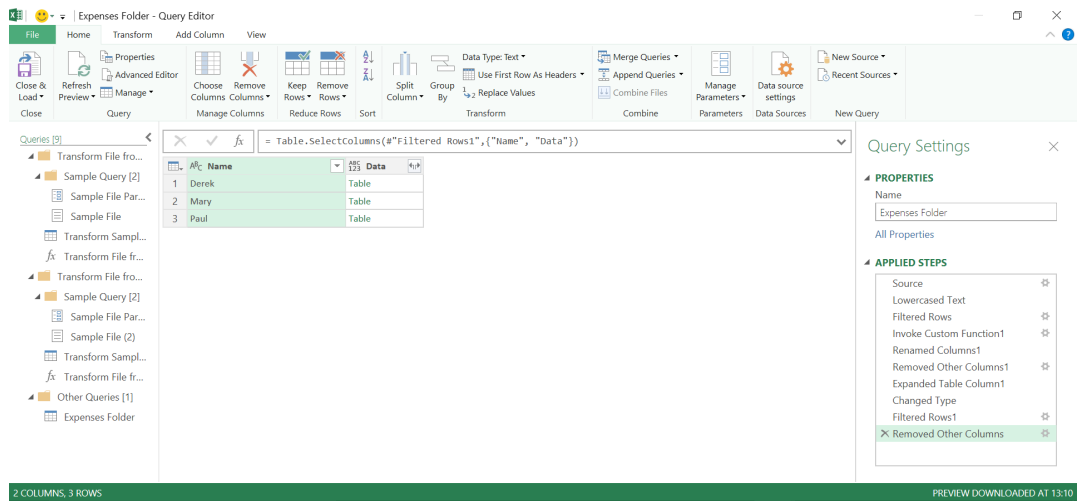


Now this looks much better! We are now ready to filter and expand the data without having to create a custom column. We also have the filename instead of the folder location as we had with our manually created column. The filename would be useful if we have to pull out information into the data.

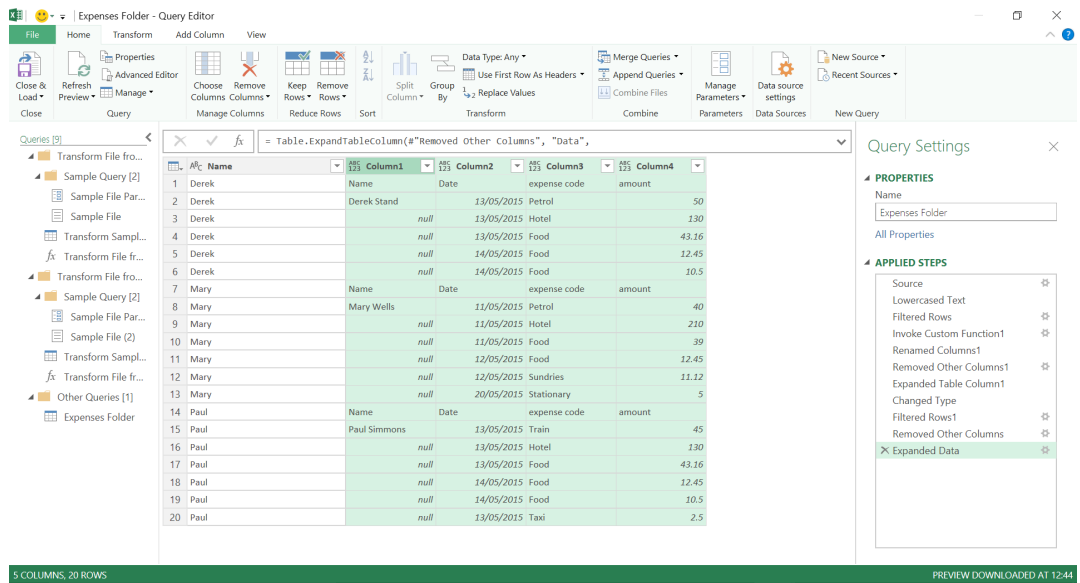
As an aside, when combining files, it can also help to select the sample file that shares as many attributes as possible with the other files, which is another way of avoiding errors. This is now possible with a new feature available on the 'Combine Files' screen.



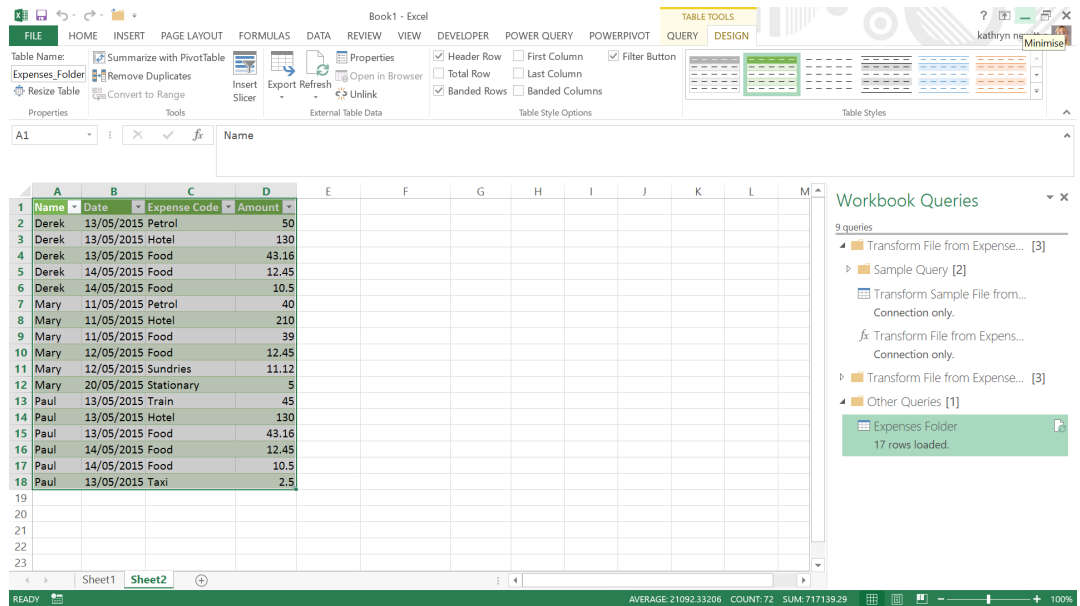
Back with my combined files, in the **Kind** column, we clearly still have some danger of duplication as there are two rows for each person; one with value 'Table' and one with value 'Sheet'. Choose to filter and keep the rows where the **Kind** column is populated with 'Sheet'. We can also get rid of most of the columns as we only want the **Name.1** and **Data** columns.



Note that Power Query is fine filtering on *Kind* even though it no longer appears. We can now expand the **Data** column:



We clearly still have some tidying up to do, but the data is all there ready to be transformed. Once we are happy we may upload the data.



So, thanks to the updates to 'Combine Binaries' (now 'Combine Files'), it is much easier to combine the data without resorting to custom columns. One thing we are not so keen on with combining data in this way is that all the queries that Power Query creates automatically are kept. As shown above, the queries from my failed attempt to combine sheets are

still there. With some housekeeping, we can tidy this up to just keep the queries that the 'Expense Folder' query references.

More next month!

Power BI Overview

Here is something we came across recently, which we thought we should share with our readers. Fellow Most Valuable Professional (MVP, Data Platform) Gilbert Quevauvilliers, and his company Fourmoo (www.fourmoo.com), has put together a brilliant graphic summary of how all of the aspects of Power BI come together. It is a complicated topic and our readers regularly express their confusion of what is what as name changes come and go regularly – often had in hand with corresponding features.

Take a look at this:

Power BI

Power BI Desktop

- Get Data from 80+ Sources
- Query Editor
- Composite Models
- Filter & Sort in Data View
- Bookmarks
- Ask a question (Q&A)
- Sync Slicers
- New Tables
- Lock Objects
- Quick Measures
- New Measures
- What If Parameter
- Buttons
- Print to PDF
- Themes
- Roles
- Python
- R Language
- New Columns
- Desktop Layout
- Phone Layout
- Selections
- Reports View
- Tables View
- Relationship View

Power BI Service

Free

- Aggregations
- Data Limit 10 GB/User
- 1 GB Dataset
- Dashboard Themes
- Ask a question about your data
- Natural Language Query (Q&A)
- Data Alerts
- Comments
- Microsoft SQL Server Tiles pinned from SSRS
- Consume live data sources (DirectQuery)
- Apps
- Consume Content that's scheduled to refresh Hourly (Can schedule 8 refreshes)
- Access Datasets in Power BI Service
- Consume streaming data 1M rows/Hour
- Access On-Premise data using Personal or On-Premise Data Gateway
- Export Data
- Control data access with Row Level Security for Users & Groups
- Persistent Filters
- Publish to Web
- Mobile Devices (iOS, Windows, Android)
- Custom Data Connectors

Pro (0 - 500 Users) (e)

- Create, View & Share Personal Dashboards & Reports
- New App Workspaces
- Usage Metrics (With User Details)
- Analyze in Excel
- Subscribe others to Dashboards & Reports (My Workspace & Apps)
- SharePoint Modern Web Parts
- Create, publish & View Organizational Content Packs

Premium (500 + Users) (e)

- Dedicated Node Capacity
- Incremental Refresh
- Geographic Distribution
- 100TB of Storage for each Capacity Provisioned
- Higher refresh rates up to 48 per day
- Noisy Neighbour Isolation

Future Release

- Pinning Datasets to Memory
- Read-Only Replicas
- Dedicated Data Refresh Nodes
- Large datasets



(e) – Estimated Users
(P) - Premium Features

We congratulate Gilbert on deciphering it all and putting it together in a highly user-friendly way. Do check out his website and find out first hand why he is a highly rated Data Platform MVP.

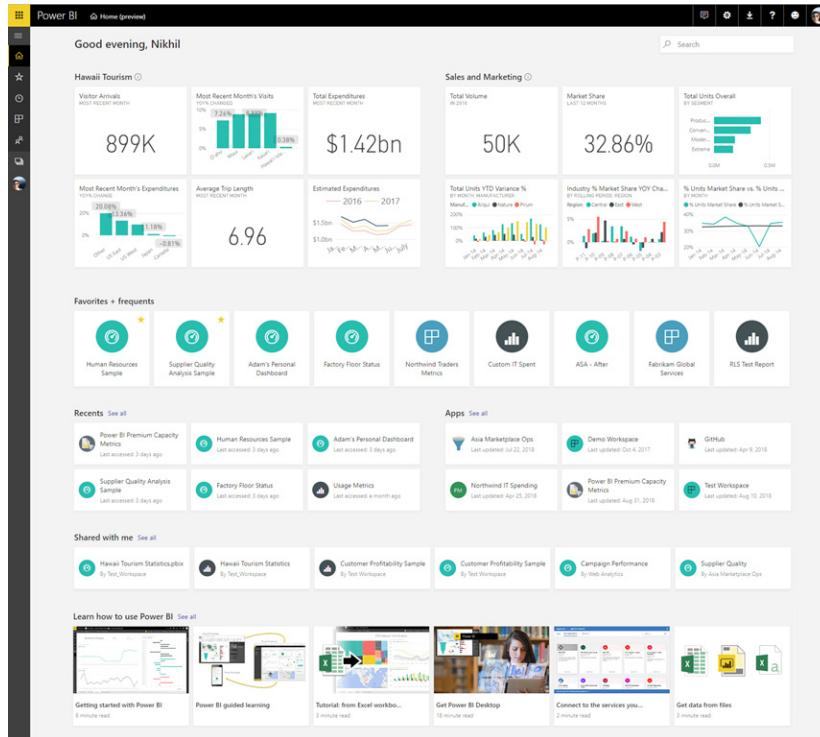
Introducing Power BI Home & Global Search

The Power BI service has assisted businesses leverage data to solve problems, share insights and make informed decisions. Users often track multiple dashboards and reports to stay on top of all their data. However, as organisations grow, it can become increasingly difficult for end users to discover and manage the Power BI content that matters to them.

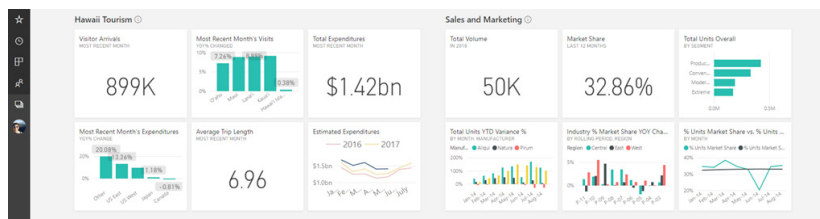
Microsoft is trying to improve the overall navigation experience by making it easier to monitor and analyse key business metrics. Whether you're distributing insights or consuming them, you need to be able to find exactly what you need when you come to the service.

Late September saw the public Preview of the Power BI Home landing page and the new Global Search feature in the Power BI service. The hope is that 'Home' will be your one-stop shop for all of your content and a quicker way to dive into the insights. The aim is 'Home' will automatically surface your most important content or let you search for it in a single page.

Here is an example of what the new homepage can look like in this Preview:



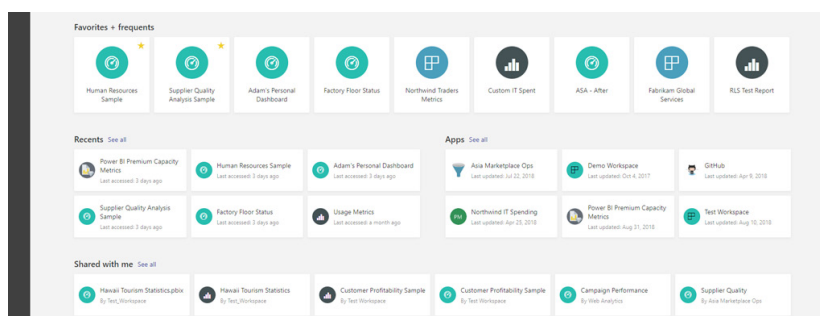
The first thing you will notice is that there are visuals pinned on top of the homepage. This update has added a dedicated section for you to monitor the metrics from your most important dashboards at a glance. From the 'Home' example (above), you can see six visuals and KPIs from the Hawaii Tourism and Sales and Marketing dashboards:



If you don't see a pinned tiles section, in this Preview, all you need to do is select a dashboard as "favorite" (sic) that you want pinned and Power BI will automatically select and pin up to six tiles from that dashboard to 'Home'. It should be noted that this current Preview only allows you to pin tiles from two "favorited" dashboards. If you have more than two that are designated as favourites, the software will automatically pick the ones that you most frequently use.

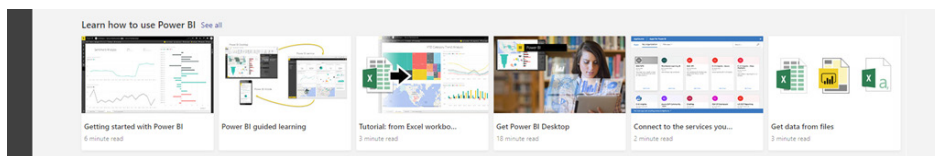
Custom visuals and live report pages are not supported for this Preview. Microsoft is evaluating this feature with a couple of other designs.

As mentioned earlier, 'Home' is focused on improving the overall navigation experience and surfacing your most important content. Right below the pinned tiles section, under 'Favorites + Frequents', you will be able to access all your "favorited" and / or most frequently used dashboards, reports, and applications.



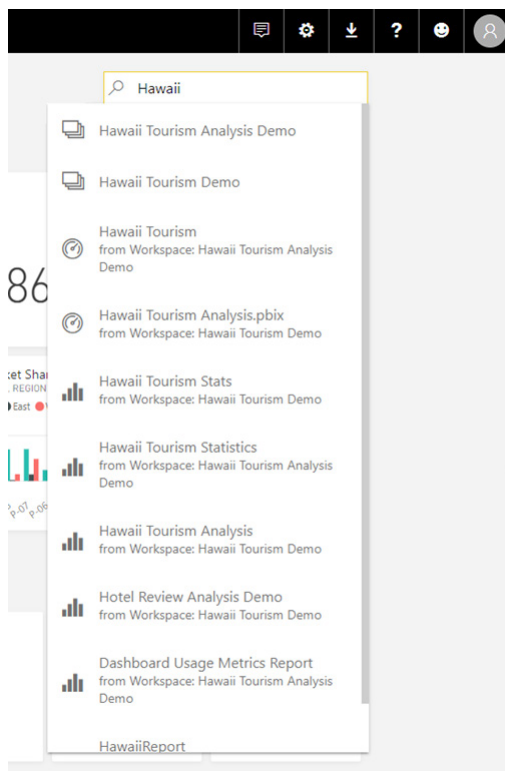
Right below that, all other content that you've accessed recently, applications that you've installed, and anything that was shared with you is surfaced in a single pane under 'Recents, Apps, and Shared with me'. You no longer need to hunt through your browser bookmarks to find that dashboard or report that you view frequently.

Next, at the bottom, there's a section added for learning resources. If you are a new user to Power BI and want to kickstart your journey, 'Home' provides various articles that will help you ramp up on both Desktop and Service. Or you could just visit the SumProduct website!



Returning to the top, there is a search field. This new feature allows you to search for all the content that you have access to within your organisation including dashboards, reports, apps and workspaces. All you have to do is type in a keyword and the overlay drop-down should list all the artefacts

that match that term. In the results, you can identify the type of artefact based on the icon and if it is a dashboard or report, you can also see its "containing folder".



It should be noted that for this Preview, Microsoft will only index once every 24 to 72 hours, so any new artefacts that you create might not show up till the next day.

There are some known issues which have already been reported:

- There have been problems navigating to content within the new workspace areas (folders) from 'Home'
- There have been reported cases of results not populating for Global Search the first time – however, Microsoft have claimed this fix should have been implemented by the time this newsletter is published!

We wait to see what's next...

October Updates for Power BI Desktop

Another month, another 21 updates. You can't say you aren't getting your money's worth (given it's free!). The latest updates are as follows:

Reporting

- Search in filter cards
- Improved accessible authoring experiences
- Performance improvements for ArcGIS Map

Modelling

- DAX editor improvements

Analytics

- Composite models and Aggregation support in the Power BI Service (Preview)
- Explain the increase for non-additive measures

Custom Visuals

- Mapbox updates: 3D extrusion on fill maps and more
- Various bar and column chart visuals by Akvelon
- 3AG Systems: column chart with small multiples
- 3AG Systems: bar chart with absolute variance
- 3AG Systems: column chart with relative variance

Data Connectivity

- Web By Example connector is now Generally Available
- SAP BW Connector implementation v2 is now Generally Available
- SAP BW Message Server Connector is now Generally Available
- Vertica connector is now Generally Available
- Dynamics NAV and Dynamics 365 Business Central connectors are now Generally Available
- New Dynamics 365 Business Central On-premises connector

Query Editing

- Data Profiling in Query Editor (Preview)
- Fuzzy Matching options for Merge Queries (Preview)

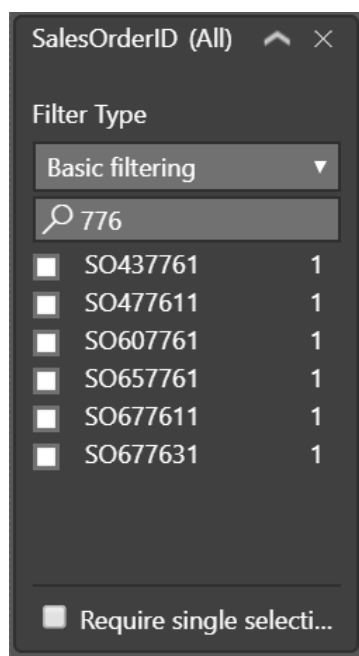
Other

- Control export data options for your reports
- Transport layer security settings.

Let's go through each new feature in turn.

Search in filter cards

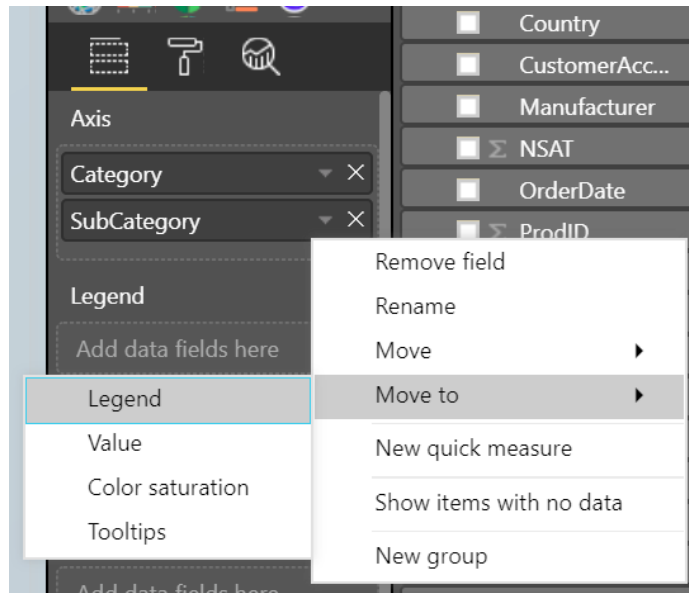
You have been able to search in slicers since since the June 2016 update and with this month's update, it's been added it into the basic filter cards as well. This functionality is heavily used in slicers, and Microsoft believes this will be an important update for the filters as well.



Improved accessible authoring experiences

As part of the continuing accessibility work in the Power BI team and across all of Microsoft, this update sees lots of improvements related to authoring and modifying reports. The field well can now be navigated using just a keyboard and interacts well with screen readers. To help

improve the usability of editing charts with screen reader and keyboard, Microsoft has also added new options to the context menu of fields to move fields up and down within a well or move to other wells.



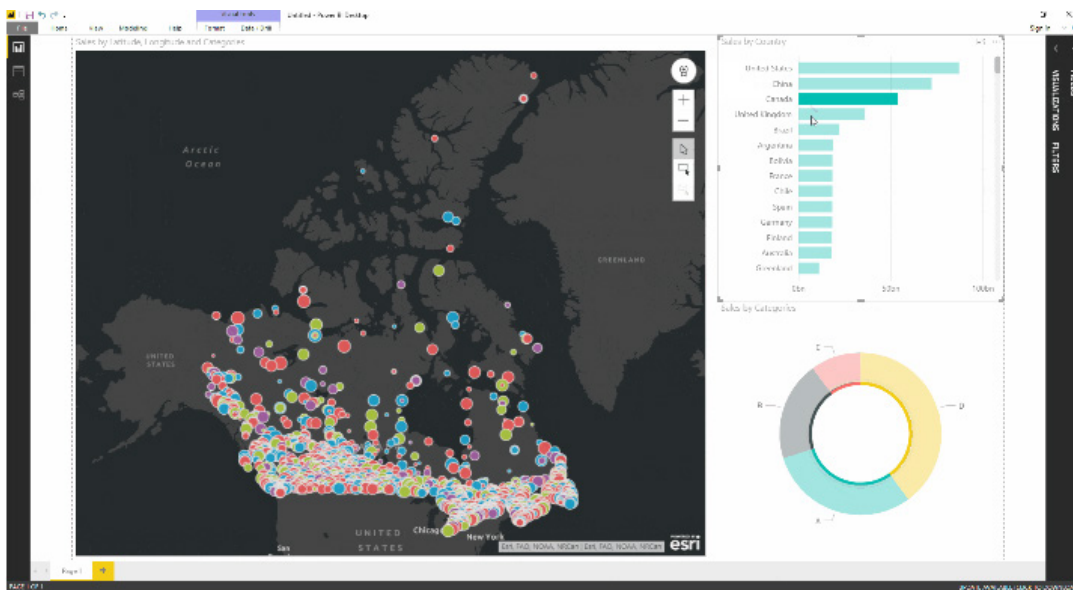
Q&A always worked with keyboard navigation, and now it also works well with screen readers too. This update is for both Power BI Desktop and the Power BI Service and applies to all the places Q&A shows up, including the

Q&A explorer and using Q&A to create visuals in reports. This is a useful way for users to create visuals quickly and easily while using assistive technology and should make report creation simpler for all.

Performance improvements for ArcGIS Map

With a recent update to ArcGIS Maps for Power BI, all maps will load up to 50% faster. Cross-highlighting, filtering, panning and zooming also have had major performance improvements. As a consequence of these performance improvements, the map can also support drawing many

more locations on the map. For latitude / longitude point locations, each map now supports 30,000 points. For standard boundaries (for example, ZIP / post codes, provinces, etc.) maps can now draw up to 15,000 areas.



DAX editor improvements

The DAX editor has been enhanced and has new keyboard shortcuts, line numbers and indent lines. The experience will be very similar to what you have for other Microsoft editors, such as VS Code.

```

1 Drug Overdose Deaths YoY% =
2 IF(
3     ISFILTERED('Overdoses'[LastDayOfMonth]),
4     ERROR("Time intelligence quick measures can only be grouped or filtered by the Power
5     BI-provided date hierarchy or primary date column."),
6     VAR __PREV_YEAR =
7         CALCULATE(
8             [DODs],
9             DATEADD(
10                'Overdoses'[LastDayOfMonth].[Date],
                -1,
            )
        )
    )

```


Some of the less well-known keyboard shortcuts you might find useful are:

ALT + ↑ / ↓	Move line up/down
SHIFT + ALT + ↓ / ↑	Copy line up/down
CTRL + ENTER	Insert line below
CTRL + SHIFT + ENTER	Insert line above
CTRL + SHIFT + \	Jump to matching bracket
CTRL +] / [Indent/outdent line
ALT + CLICK	Insert cursor
CTRL + I	Select current line
CTRL + SHIFT + L	Select all occurrences of current selection
CTRL + F2	Select all occurrences of current word

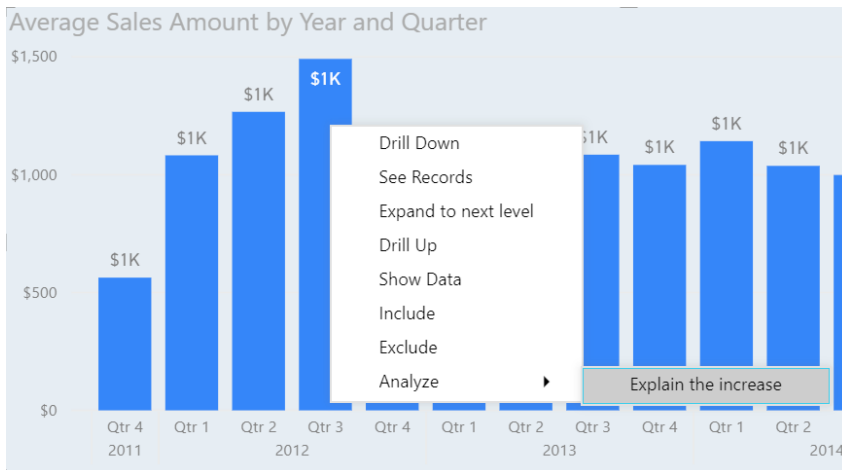
Composite models and aggregation support in the Power BI Service (Preview)

The composite models Preview has had a significant update. You may now publish files that are using composite models or aggregations to the Power BI Service, and they will work as expected there. This should assist users to start testing composite and aggregation models end to end.

Composite models have been updated in a few other ways as well. The “Blank Query” experience received an overhaul based on various user forum feedback. Apparently, this is all getting very close to becoming Generally Available.

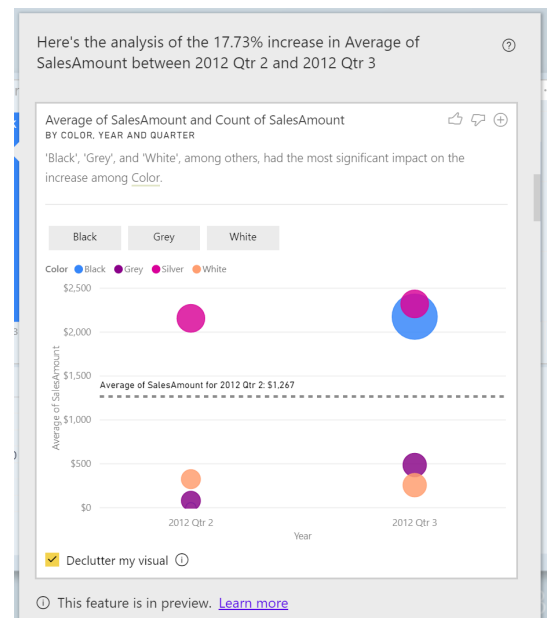
Explain the increase for non-additive measures

The ‘Explain the increase’ and ‘Explain the decrease’ insights questions are now available on all measure types, not just sums and counts. It works the same way as it did for sum and count measures, so from a visual, just right-click on a point and select **Analyze > Explain the increase**.



The visuals you get back will depend on the type of measure you are using. You will continue to get the waterfall chart showing the breakdown for additive measures (sum and count). For measures that are averages or ratios, you will see a dot plot visual, and for everything else, you will see a clustered column chart. No matter what, they will all be geared to answering the question you asked: why did this measure’s value increase or decrease?

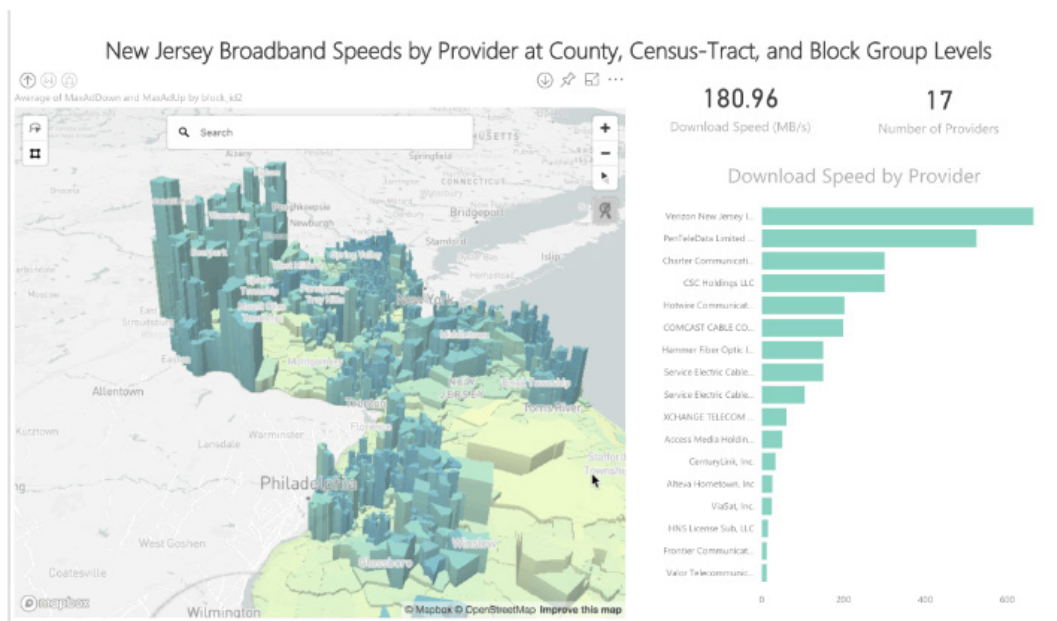
The dot plot you get for averages and ratios shows the measure plotted for two periods you selected. The periods will be on the x-axis and measure on the y-axis. The size of the bubble reflects the weight of the measure.



Mapbox updates: 3D extrusion on fill maps and more

Mapbox has provided another list of major improvements for their custom visual in this update. With the latest version of their custom visual, you can now add an extra dimension to your fill maps by using 3D extrusions.

In addition to dynamic fill colour, you can now add data-driven heights to the polygons as well. All you have to do is drop a measure into the 'Size' bucket and set the relative extrusion height.



To go along with 3D extrusions, they've added an option to set the default map pitch, which is usually needed in order to get the most context on polygon heights. There's more though:

- **Autozoom to selected shapes.** Now, whenever you filter the fill maps, the map will autozoom to the selection
- **Ability to turn off map controls.** You may also add a little more real estate on your maps by turning off the map controls for zooming and selection if they aren't required
- **Smarter tooltips.** Tooltips will now automatically display the underlying dimension's format and tooltips are now smart enough to only show if something is specified.

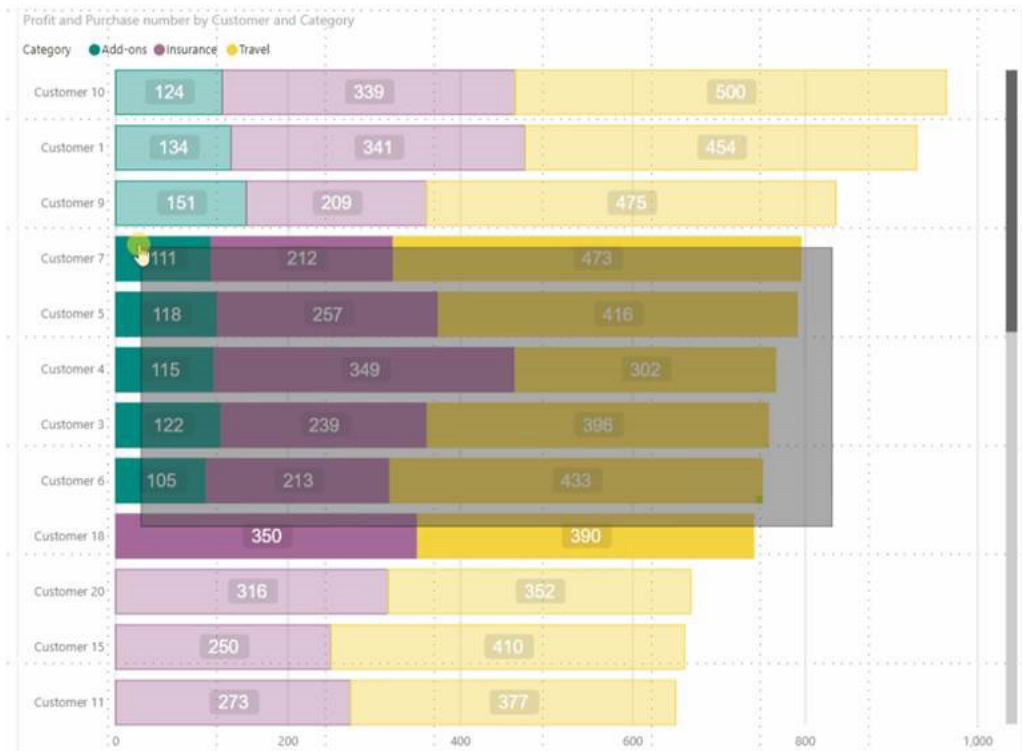


Lastly, you can now add a search bar to your map. This is very useful if you have a lot of data across a lot of locations, allowing you to jump quickly to any part of the map. All you need to do is start typing a location and Mapbox will autofill suggested locations for you. You can even optionally drop a pin at the location.

Various bar and column chart visuals by Akvelon

Akvelon recently released a set of new bar and column chart custom visuals. They are similar to Microsoft's built-in bar and column with the additional functionality supporting "rectangle lasso selection" (ride 'em

cowboy). You can click and drag to select multiple bars within rectangle area and cross-filter the rest of the report based on that selection.

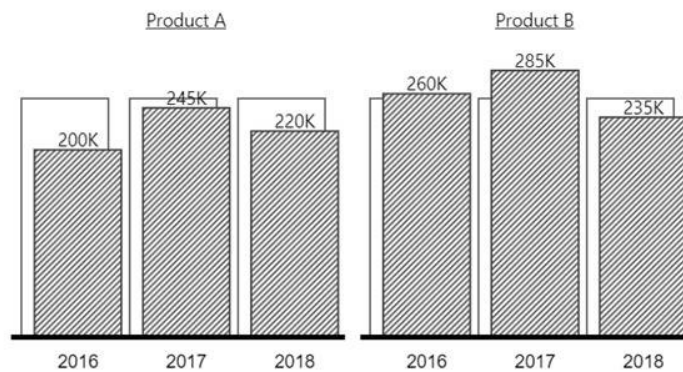


3AG Systems: column chart with small multiples

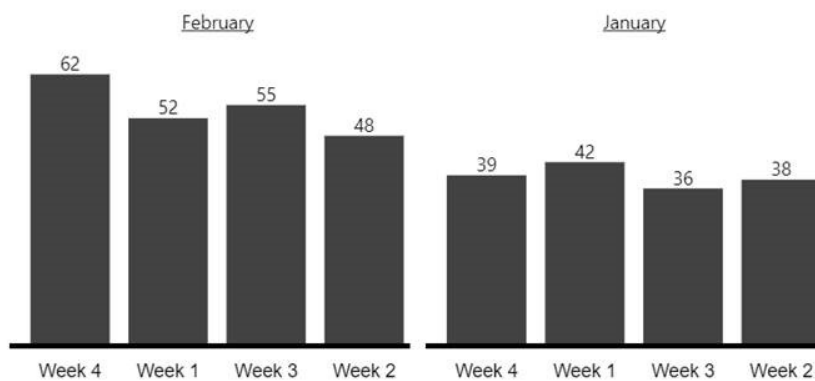
The column chart with small multiples custom visual by 3AG Systems allows you to create multiple overlapped column charts for each element in a group. It compares two scenarios, such as actual vs. forecast, and

displays overlapped column charts. From there, it will break that down by each value of a category you pick and visualise the charts next to each other.

Forecast VS Planned



While the visualisation is geared towards showing Actual, Forecast, Planned and Previous Year data across different categories (for example departments, regions, products etc.), you may also use small multiples if you only have actual values as well.

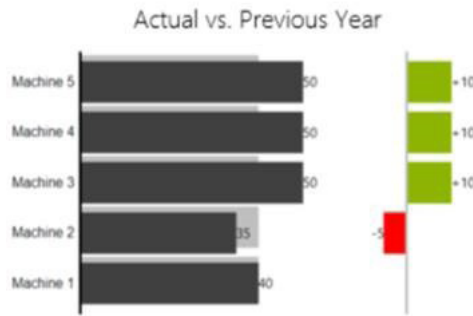


Not sure why the weeks have to be in the wrong order though...

3AG Systems: bar chart with absolute variance

The bar chart with absolute variance custom visual by 3AG Systems automatically calculates absolute variance and displays an overlapped bar chart with the variance. The visual calculates absolute variance between two scenarios and generates an overlapping bar chart that

displays two scenarios with a variance bar chart situated parallel to the chart. This visualisation is meant for comparing Actual, Forecast, Planned and Previous Year data, and works best with categorical data on the y-axis.

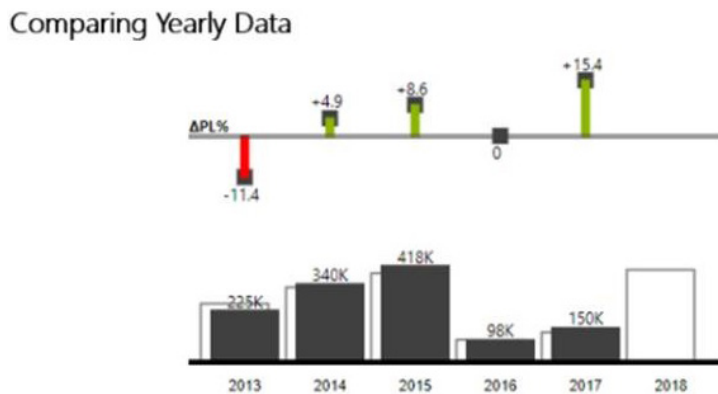


3AG Systems: column chart with relative variance

The column chart with relative variance custom visual by 3AG Systems calculates percentage change and displays an overlapped column chart with lollipop variance. It calculates relative or percentage variance between two scenarios and generates an overlapping column chart that displays the relative variance with a lollipop chart situated parallel to the chart. This visualisation is meant for comparing Actual, Forecast,

Planned and Previous Year data and works best when using a time series variable on the x-axis.

The visual has lots of formatting options including inverting the colours for red and green, resizing the data label font sizes, hiding data labels and adjusting units (to thousands-K, Millions-M, Billions-B).



Web By Example connector is now Generally Available

The Web By Example connector was released a few months ago as a Preview feature. The connector has now become Generally Available (GA). However, it should be noted that in order to be able to refresh

datasets that leverage this connector in the Power BI Service, you will need to upgrade to the October release of the On-premises data gateway.

SAP BW Connector v2 is now Generally Available

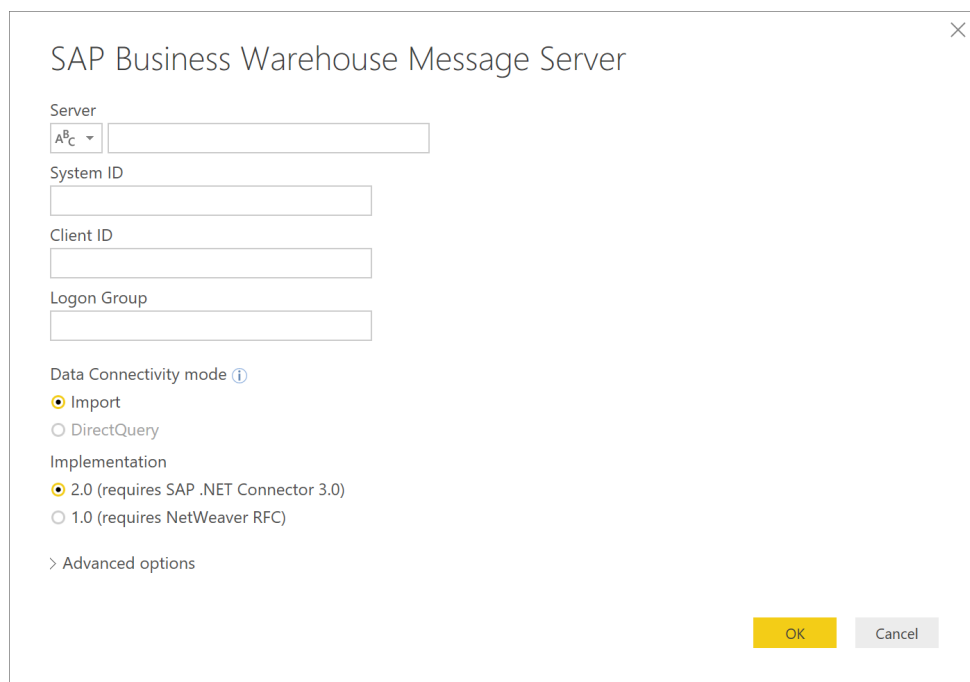
The recently added "v2" implementation of the SAP BW connector is now also GA. You can access this new implementation from the SAP BW connector dialog under the "Implementation" section. The "v2" implementation is now the default setting for new connections against SAP BW.

A screenshot of the "SAP Business Warehouse Application Server" dialog box. The dialog has a title bar with a close button (X). It contains the following fields and options:

- Server: A dropdown menu showing "APC" and an empty text box.
- System number: An empty text box.
- Client ID: An empty text box.
- Data Connectivity mode (i): Radio buttons for "Import" (selected), "DirectQuery", and "Implementation".
- Implementation: Radio buttons for "2.0 (requires SAP .NET Connector 3.0)" (selected) and "1.0 (requires NetWeaver RFC)".
- > Advanced options: A link to expand advanced options.
- Buttons: "OK" and "Cancel" buttons at the bottom right.

SAP BW Message Server Connector is now Generally Available

That's not all. The SAP BW Message Server connector is now also Generally Available.

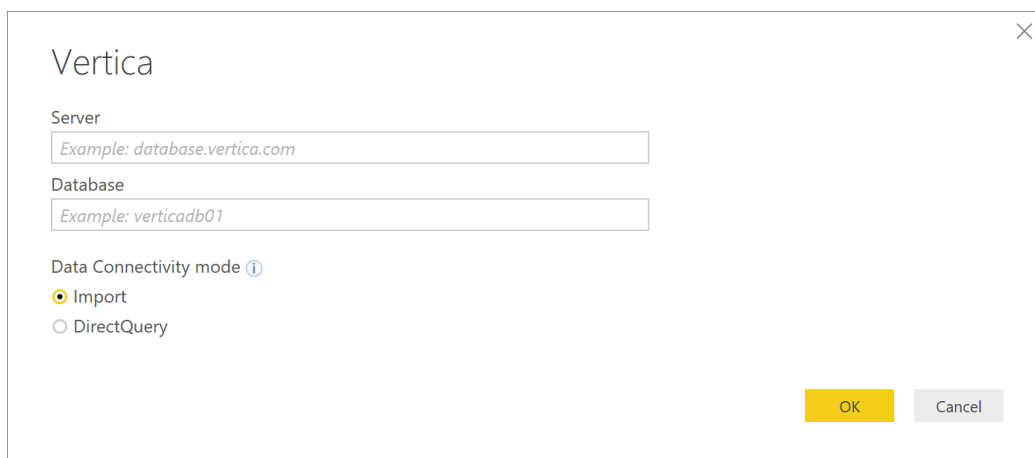


The screenshot shows a configuration dialog box titled "SAP Business Warehouse Message Server". It contains several input fields: "Server" with a dropdown menu showing "A^BC" and an empty text box; "System ID" with an empty text box; "Client ID" with an empty text box; and "Logon Group" with an empty text box. Below these fields, there are radio buttons for "Data Connectivity mode": "Import" (selected), "DirectQuery", and "Implementation": "2.0 (requires SAP .NET Connector 3.0)" (selected), "1.0 (requires NetWeaver RFC)". There is also a link for "> Advanced options". At the bottom right, there are "OK" and "Cancel" buttons.

Vertica Connector is now Generally Available

The Vertica connector is now GA, after being in Beta for a few months. The Vertica connector allows you to build Import-based or DirectQuery-based reports against Vertica databases. Please note that in order to

refresh datasets based on Vertica data in the Power BI Service you will need to upgrade to the October release of the On-premises data gateway.



The screenshot shows a configuration dialog box titled "Vertica". It contains two input fields: "Server" with the example text "Example: database.vertica.com" and "Database" with the example text "Example: verticadb01". Below these fields, there are radio buttons for "Data Connectivity mode": "Import" (selected) and "DirectQuery". At the bottom right, there are "OK" and "Cancel" buttons.

Dynamics NAV and Dynamics 365 Business Central connectors are now Generally Available

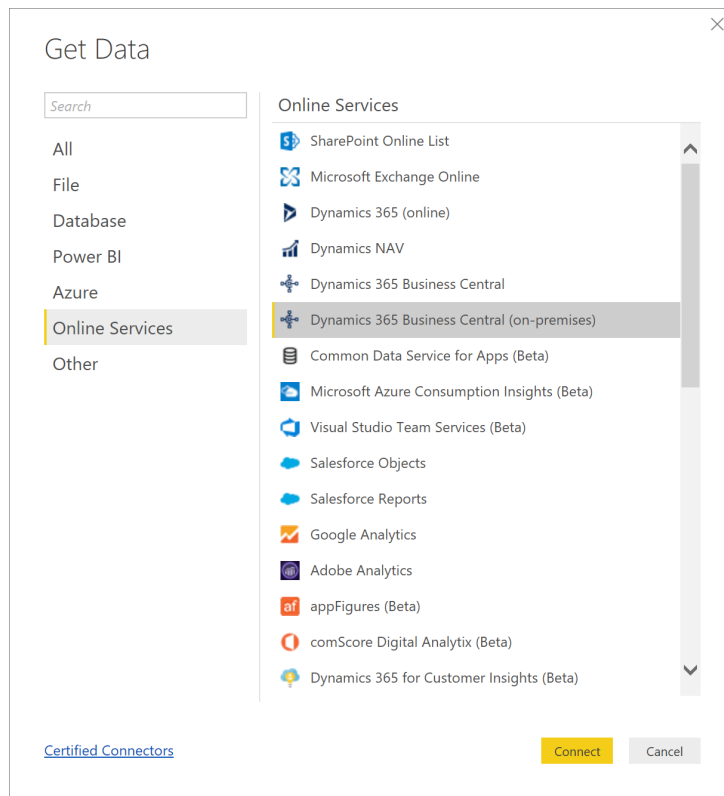
After a few months in Beta, the Dynamics NAV and Dynamics 365 Business Central connectors are also Generally Available. These connectors can be found under the 'Online Services' tab within the 'Get Data' dialog. They allow you to import data from the Dynamics NAV or Dynamics 365 Business Central accounts.

Please note that in order to be able to refresh datasets that leverage this connector in the Power BI Service, yet again you will need to upgrade to the October release of the On-premises data gateway.

New Dynamics 365 Business Central On-premises connector

In addition to all the previously mentioned connector enhancements, this update sees the release of a new connector, allowing customers to import and build reports on top of their Dynamics 365 Business Central On-premises data.

The new Dynamics 365 Business Central On-premises connector can be found under the 'Online Services' category in the 'Get Data' dialog, viz.



You know what we are going to say! Please note that in order to be able to refresh datasets that leverage this connector in the Power BI Service, you will need to upgrade to the October release of the On-premises data gateway.

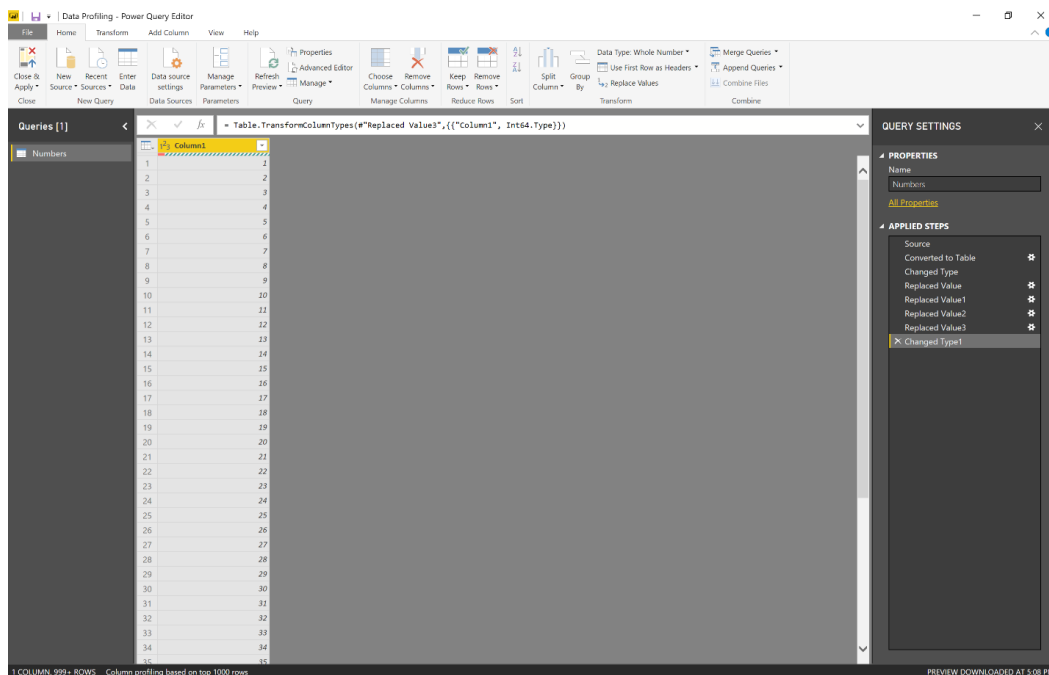
Data Profiling in Query Editor (Preview)

The Power Query Editor offers hundreds of data transformations for you to transform, filter and prepare your data for analysis. However, identifying quality issues such as errors, empty values and outliers can be quite difficult.

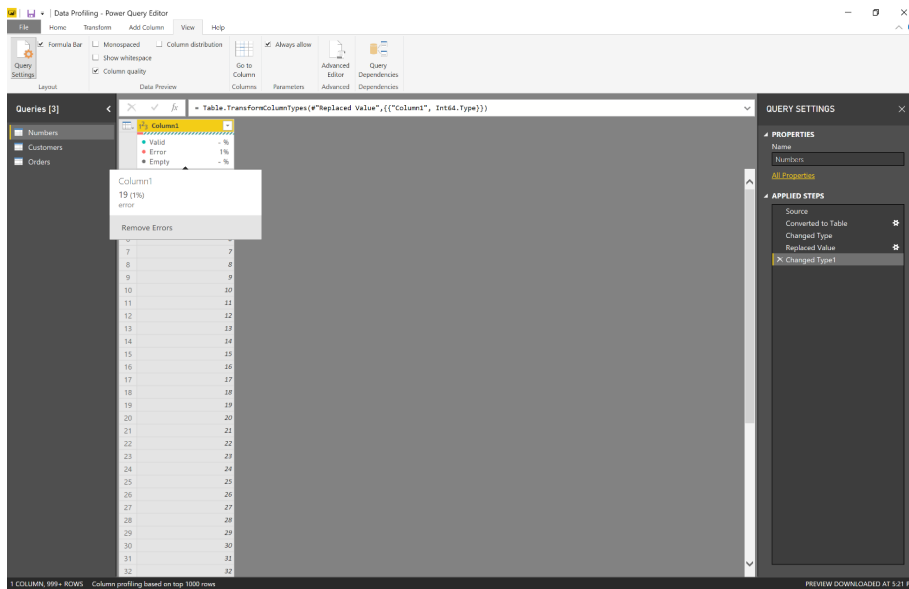
With this month's release of Power BI Desktop, Microsoft has added Data Profiling capabilities into the Power Query Editor, allowing you to

identify errors and empty values in your data previews. You can enable 'Data Profiling' from the 'Preview features' tab in the **File -> Options** dialog.

After enabling the feature, you should be able to see the quality bar below the column headers indicating whether there were any error values found or not.

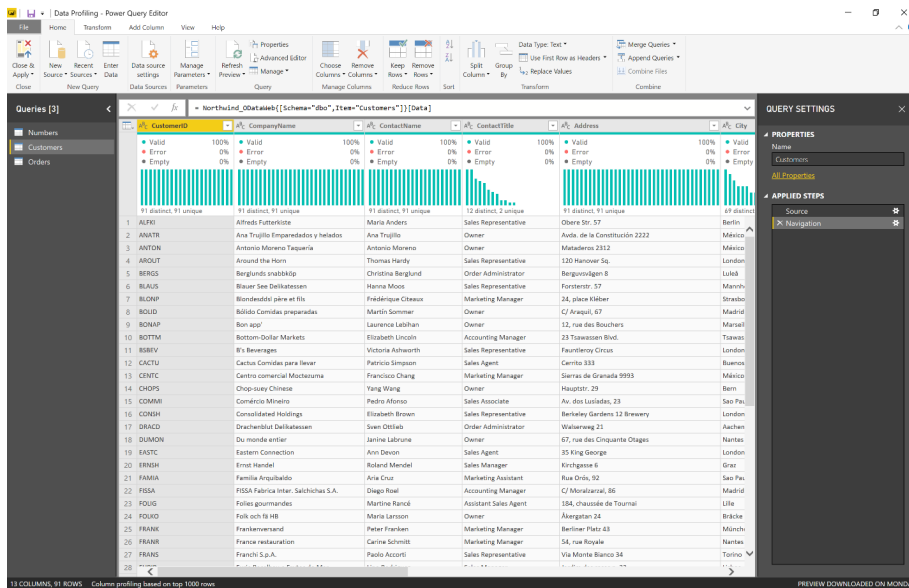


Note that you can also enable further 'Data Profiling' information by turning on the 'Column quality' option under the 'View' tab. This allows you to see counts for errors, valid or empty values. In addition, you can take action directly from the fly-out menu to 'Remove Errors'.

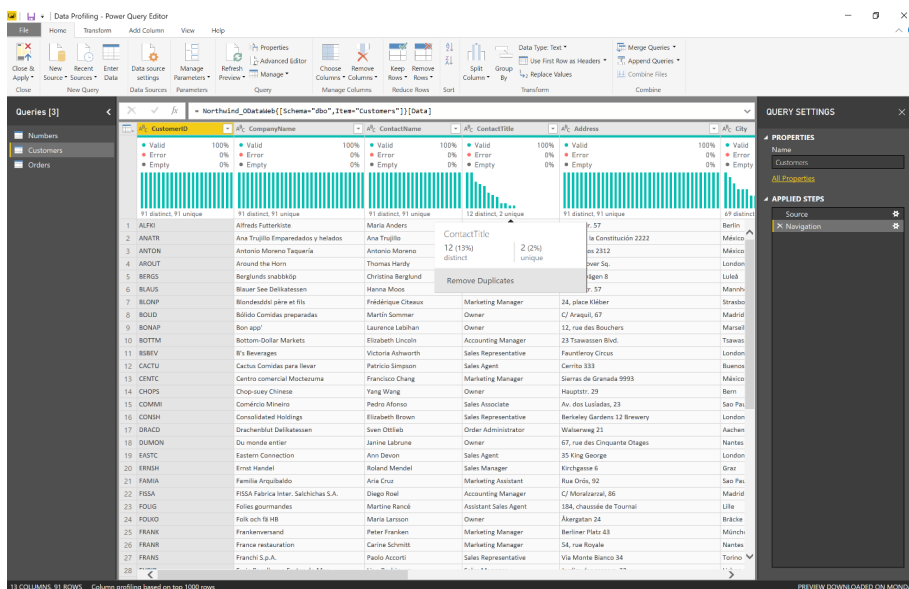


Another 'Data Profiling' capability added in this release is the 'Column Distribution' (which can also be enabled from the 'View' tab). 'Column Distribution' allows you to get a sense for the overall distribution of values within a column in your data previews, including the count of

distinct values (total number of different values found in a given column) and unique values (total number of values that only appear once in a given column).



You can also take action directly from this view, in order to 'Remove Duplicates'.



Fuzzy Matching options for Merge Queries (Preview)

'Merge Queries' allows you to combine data from multiple tables within the Power Query Editor. In this update, the option has been added to compare values in the columns to match by using Fuzzy Matching logic, in addition to the existing "exact match" option.

You can enable 'Fuzzy Merge' from the 'Preview' features list in the 'Options' dialog. Once you have done this, you'll be able to access the new 'Fuzzy Matching' options from within the 'Merge Queries' dialog. Once selecting the 'Fuzzy Matching' option, you can also (optionally) further tweak the fuzzy matching settings. This includes configuring:

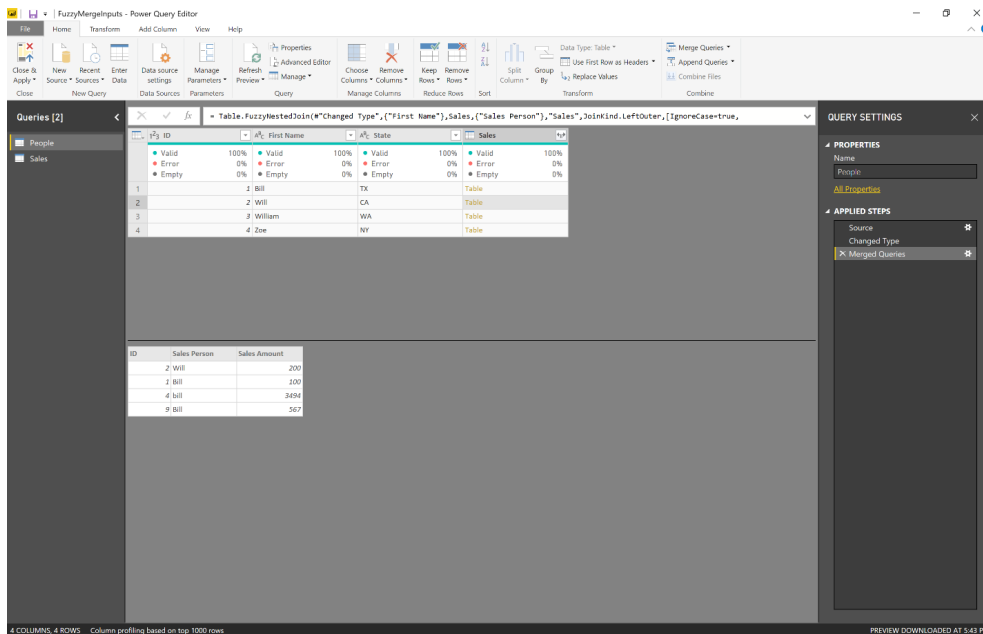
- **Similarity Threshold:** this option indicates how similar two values need to be in order to match. The minimum value of 0.00 will cause all values to match each other and the maximum value of 1.00 will only allow exact matches. The default is 0.80
- **Ignore case:** this option indicates whether text values should be compared in a case sensitive or insensitive setting. The default behaviour is case insensitive (ignore casing)
- **Ignore spaces:** this option indicates whether text values should be compared taking into account spaces or not, such as "Some value" vs. "Somevalue". The default behaviour is to ignore spaces
- **Maximum number of matches:** this option controls the maximum number of matching rows that will be returned for each input row. For example, if you only want to find one matching row for each input row, specify a value of 1. The default behaviour in this instance is to return all matches
- **Transformation table:** this option allows users to specify another query that holds a mapping table, so that some values can be auto-mapped as part of the matching logic. For example, defining a two-column table with a "From" and "To" text columns with values "Microsoft" and "MSFT" will make these two values be considered the same (similarity score of 1.00) by the matching logic.

The screenshot shows the 'Merge' dialog box with the following configuration:

- Table 1: People (ID, First Name, State)
- Table 2: Sales (ID, Sales Person, Sales Amount)
- Join Kind: Left Outer (all from first, matching from second)
- Use fuzzy matching to compare the merge
- Fuzzy merge options
 - Similarity threshold (optional): []
 - Ignore case
 - Ignore spaces
 - Maximum number of matches (optional): []
 - Transformation table (optional): []

Information: The selection has matched 3 out of the first 4 rows.

This will allow non-exact matches to be performed as part of 'Merge', like the ones showed in the following example (matching "Will" with multiple variations: Bill, bill, Will, will, etc.)



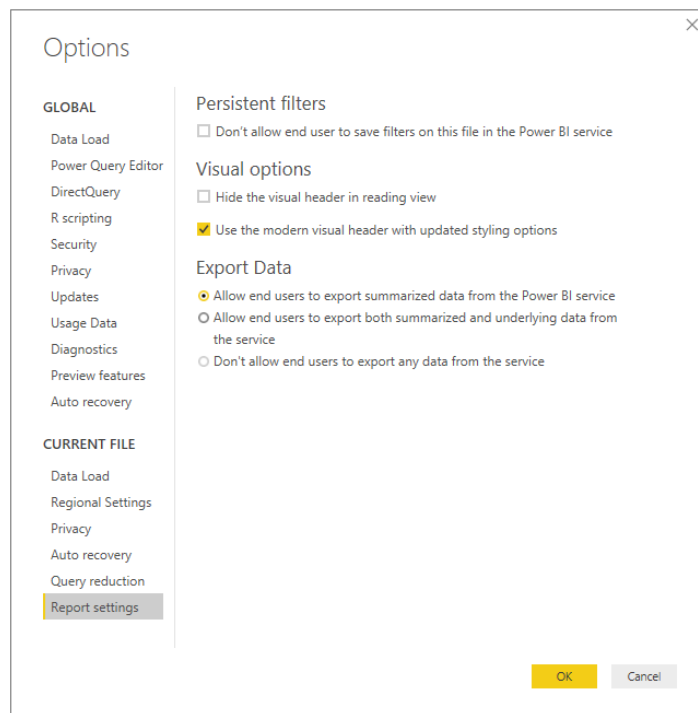
And guess what? Please note that in order to be able to refresh datasets that leverage this connector in the Power BI Service, you will need to upgrade to the October release of the On-premises data gateway.

Control over export data options for your reports

You now have the flexibility to control what types of data export options are available for your consumers when the report is published to the Power BI Service. You can choose to:

- allow exporting only of summarised data (this will be the new default for reports)
- allow exporting of summarised and underlying data (this is the default for all old reports)
- don't allow exporting of any data.

You can set this in Power BI Desktop through the options dialog under **Report settings**.



You can also update this setting in the Power BI Service through the 'Report Settings' pane.

Settings for Customer Profitabi...

Report name

Persistent filters

Don't allow end user to save filters on this report.

Visual options

Hide the visual header in reading view.

Use the modern visual header with updated styling options.

Export data

Choose the type of data you allow your end users to export.

Summarized data and underlying data ▼

Summarized data

Summarized data and underlying data

None

Transport-layer security options

Security is a priority for users and Microsoft alike, and the software giant has company-wide programs in place to ensure customers have control over the security of communications with Microsoft services. IT and network security administrators may want to force usage of more recent versions of TLS (transport layer security) for any secured communication

on their network, and Power BI Desktop now respects the Windows registry keys you use to manage this. For example, you can disable client applications from using the older TLS 1.0 by setting the following in the Windows registry:

- [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.0\Client] "Enabled"=dword:00000000
- [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.0\Client] "DisabledByDefault"=dword:00000001

Power BI Desktop will respect the registry keys specified on those pages, and only create connections using the right version of TLS. There's more information on the Windows TLS documentation sites here:

<https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/operations/manage-ssl-protocols-in-ad-fs>

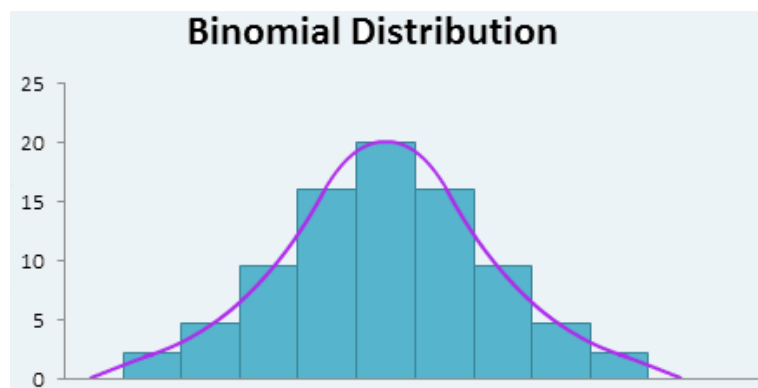
<https://docs.microsoft.com/en-us/windows-server/security/tls/tls-registry-settings>

More next month we're sure!

The A to Z of Excel Functions: CRITBINOM

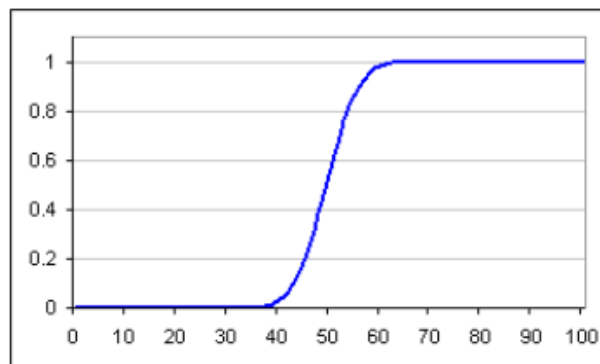
In probability theory and statistics, the binomial distribution with parameters **n** and **p** is the discrete probability distribution of the number of successes in a sequence of **n** independent success / failure experiments, each of which yields success with probability **p**. For the record, a success

/ failure experiment is also called a Bernoulli experiment or Bernoulli trial. The binomial distribution is frequently used to model the number of successes in a sample of size **n** drawn with replacement from a population of size **N**.



This function returns the smallest value for which the cumulative binomial distribution which is greater than or equal to a criterion value. This might sound like gobbledygook, but it is useful for creating independent simulations analysis in Excel (please see [Simulation Stimulation](#) for more information).

For example, the chart (*below*) shows the cumulative Binomial Distribution function for 100 tosses of a coin. This curve represents the probability that at most x heads will be thrown from the 100 tosses:



The Excel **CRITBINOM** function calculates the inverse of the curve on the right, *i.e.* the **CRITBINOM** function calculates the minimum value of x for which the cumulative Binomial Distribution function is a specified value.

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

CRITBINOM(trials, probability_s, alpha)

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

- **trials**: this is required and represents the number of Bernoulli trials
- **probability_s**: this is also required. This is the probability of a success on each trial
- **alpha**: again, required. This represents the aforementioned criterion value.

It should be further noted that:

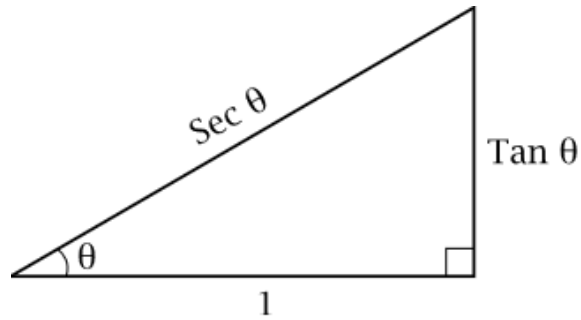
- this function has been replaced with a newer function (**BINOM.INV**) that may provide improved accuracy and whose name better reflects its usage. Although this function is still available for backward compatibility, you should consider using the new functions going forward, since this function may not be available in future versions of Excel
- if any argument is nonnumeric, **CRITBINOM** returns the **#VALUE!** error value
- If **trials** is not an integer, it is truncated
- If **trials** < 0, **CRITBINOM** returns the **#NUM!** error value
- If **probability_s** is < 0 or **probability_s** > 1, **CRITBINOM** returns the **#NUM!** error value
- If **alpha** < 0 or **alpha** > 1, **CRITBINOM** returns the **#NUM!** error value.

Please see our example below:

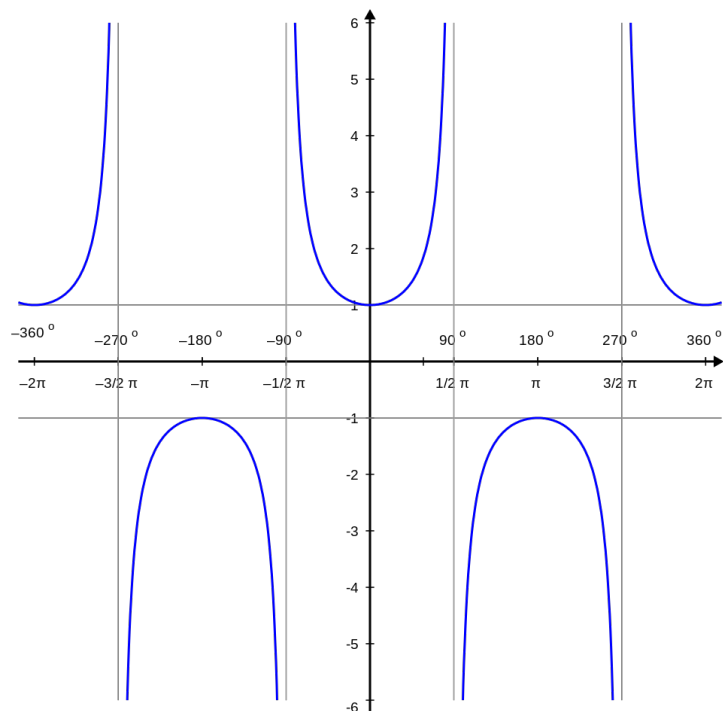
	A	B	C	D	E	F
1		Description			Value	
2		Number of Bernoulli trials			6	
3		Probability of success on each trial			50%	
4		Number of independent trials			0.75	
5						
6						
7	Description				Result	Formula
	Smallest number for which the cumulative binomial distribution is greater than or equal to a criterion value				4	=CRITBINOM(E2,E3,E4)
9						

The A to Z of Excel Functions: CSC

This function returns the cosecant of an angle specified in radians:



It has the following chart properties:



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

CSC(number)

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

- **number**: this is required.

It should be further noted that:

- the absolute value of **number** must be less than 2^{27}
- if **number** is outside its constraints, **CSC** returns the **#NUM!** error value
- if **number** is a non-numeric value, **CSC** returns the **#VALUE!** error value
- **CSC(n)** equals $1/\text{SIN}(n)$.

Please see a relevant example below:

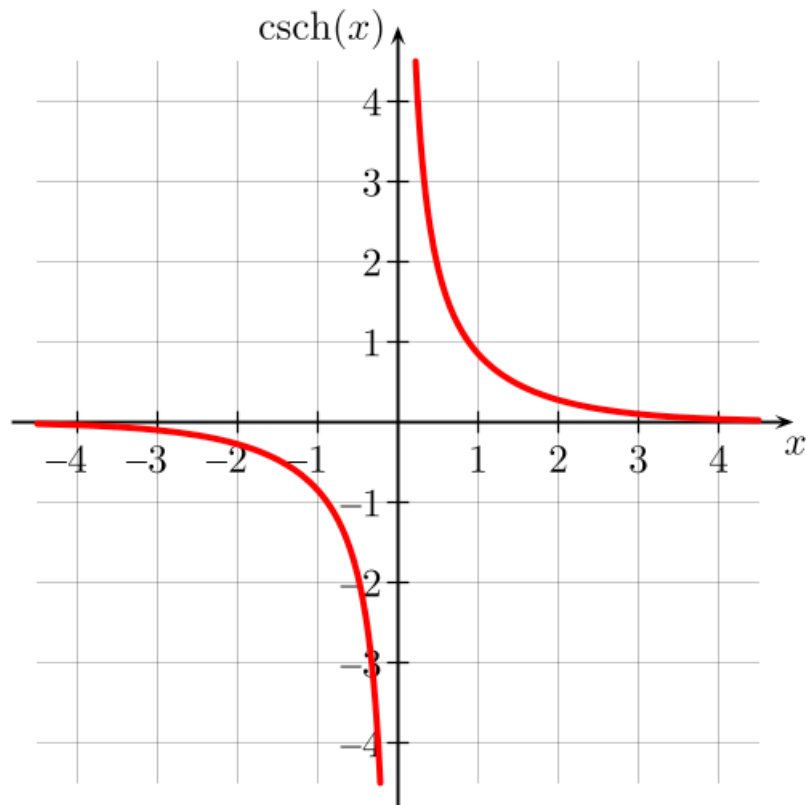
Formula	Description	Result
=CSC(15)	Returns the cosecant of 15	1.538

The A to Z of Excel Functions: CSCH

The hyperbolic cosecant function is an old mathematical function. It has a simple definition as the reciprocal to hyperbolic sine function:

$$\operatorname{csch} x = \frac{1}{\sinh x} = \frac{2}{e^x - e^{-x}}, x \neq 0$$

It has the following chart properties:



This function returns the hyperbolic cosecant of an angle specified in radians.

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

CSCH(number)

The CSCH function has the following arguments:

- **number**: this is required.

It should be further noted that:

- the absolute value of **number** must be less than 2^{27}
- if **number** is outside its constraints, **CSCH** returns the **#NUM!** error value
- if **number** is a non-numeric value, **CSCH** returns the **#VALUE!** error value.

Please see our final example for this month below:

Formula	Description	Result
<code>=CSCH(1.5)</code>	Returns the hyperbolic cosecant of 1.5	0.4696

Upcoming SumProduct Training Courses

Location	Course	Date	Duration
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 - 21 Nov 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 look at those function keys (great for all the Mac users out there, we're sure!):

Keystroke	What it does
F1	Help
F2	Toggle Select, Edit and Enter / Point modes
F3	Paste Names
F4	Redo / Edit (F2) Mode: Toggle \$ references
F5	Go To / Refresh File List
F6	Next Pane
F7	Check Spelling
F8	Extend Selection Mode
F9	Calculate Now
F10	Activate Menus
F11	Insert Chart on New (Chart) Sheet
F12	Save As

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