SumProduct

NEWSLETTER #126 - May 2023
www.sumproduct.com | www.sumproduct.com/thought

As another version of Office is consigned to the great office in the sky its legacy continues to live on and thrive unabated, with Office 365 plus partners in crime Power Pivot, Power Query and Power BI all charging along at warp speed. Office 2013 is dead; long live Office seems to be the mantra from Redmond.

So I guess we had better keep you posted with what is going on! We talk about the new feature Excel Labs, and making its debut in this newsletter is an article on how to model inventory on a Last In, First Out (LIFO) basis. This has proved controversial with various accounting editors declining to publish it, but it is a commonly requested query so we thought we would address it in this newsletter.

We also have the latest Beat the Boredom Challenge, plus Charts & Dashboards, Visual Basics, Power Pivot Principles, Power Query Pointers, Power BI Updates, Excel Updates, more Keyboard Shortcuts and the A to Z of Excel functions living it **LARGE**. Find out more below.

As always, happy reading and remember: stay safe, stay happy, stay healthy.

Liam Bastick, Managing Director, SumProduct

End of Support for Office 2013

Support for Office 2013 ended on Tuesday, 11th April (why a Tuesday?) and there will be no extension and no extended security updates. Microsoft has stated that all of your Office 2013 apps will continue to function **BUT** you could expose yourself to serious and potentially harmful security risks.

In essence, "end of support" means:

- Microsoft will no longer provide technical support, bug fixes or security fixes for Office 2013 vulnerabilities which may be subsequently reported or discovered. This includes security updates which can help protect your PC from harmful viruses, spyware and other malicious software
- you'll no longer receive Office 2013 software updates from Microsoft Update (and beware anyone offering you such updates)
- you'll no longer receive phone or chat technical support
- no further updates to support content will be provided and most online help content will be retired.

Time to upgrade!

Excel Labs

Excel Labs is an add-in that allows the folks at Redmond to release experimental ideas for you to try, and to provide them with feedback that helps Excel evolve. What is known as "The Microsoft Garage" is Microsoft's official outlet for experimental projects across the company so that teams may receive early feedback from customers and better determine product market fit. With Excel Labs, in alignment with the Garage's mission, you should expect to find very early-stage ideas that Microsoft is thinking about and wanting to evaluate with their customers. It should be noted that whilst some of these ideas may never make it to the Excel product, it's believed that having this avenue to get feedback is crucial for creating new features that transform what's possible in Excel.

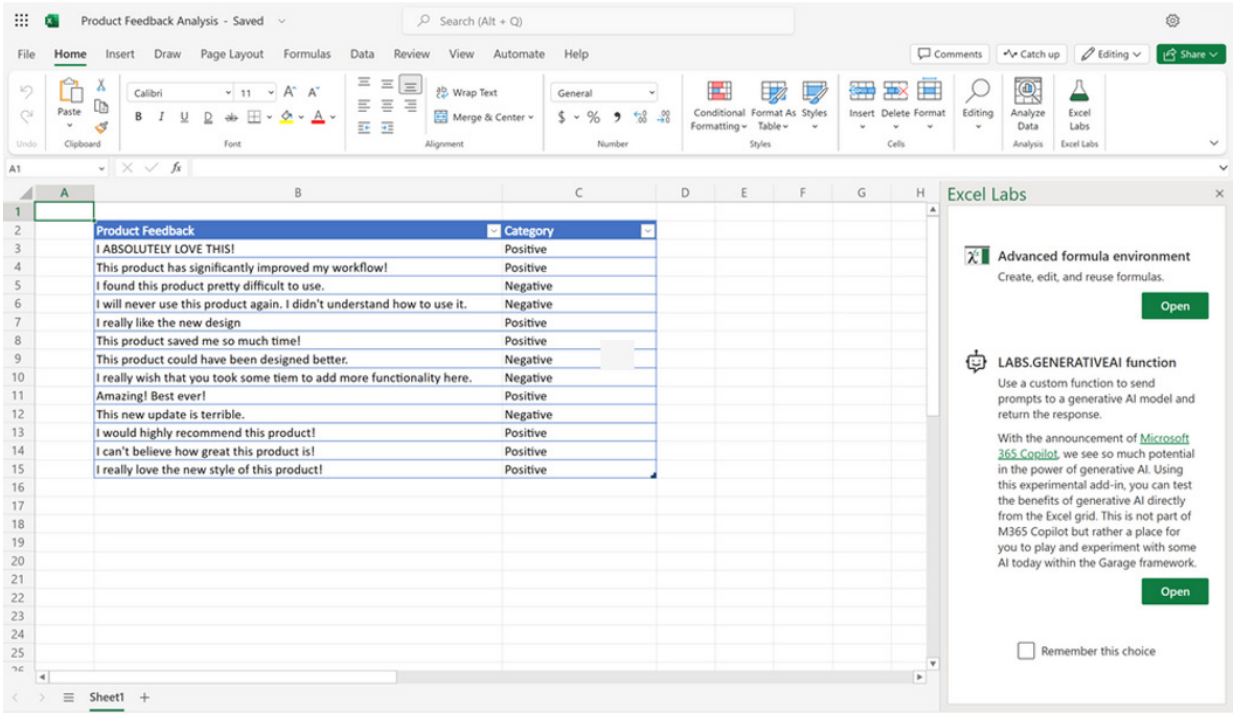
The first two experimental ideas to be released as part of Excel Labs are:

- **Advanced formula environment:** this advanced editor for complex formulae was previously released as a separate Garage project, the Advanced Formula Environment add-in. However, the AFE is now being upgraded and rolling all of its functionality into Excel Labs, so you won't need multiple add-ins installed
- **LABS.GENERATIVEAI custom function:** as you may have noticed last month with the announcement of Microsoft 365 Copilot, Microsoft sees so much potential in the power of generative AI. Therefore, with this experimental add-in, you can test the benefits of generative AI directly from the Excel grid using the LABS.GENERATIVEAI custom function.

  This function allows you to send prompts from the Excel grid to a generative AI model and then return the results from the model back to your worksheet. It should be noted that this is not part of M365 Copilot, but rather a place for you to play and experiment with generative AI today within the Garage framework.

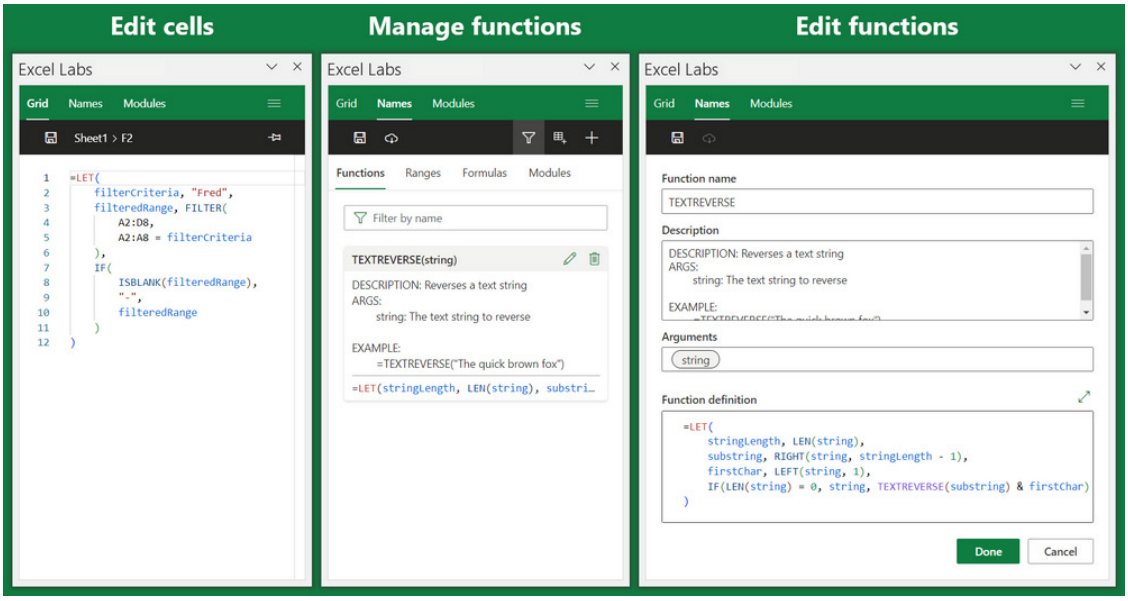
To get started, simply install the Excel Labs add-in through the Office Store. If you don't see the add-in when you type 'Excel Labs' into the Office Store search box, your version of Office may not meet the minimum system requirements.

After the add-in is installed or updated, select the 'Excel Labs' button in the Home tab. This will open the Excel Lab's feature gallery page where you can choose which feature you want to use. To get back to the feature gallery at any time, select the overflow menu in the navigation bar.



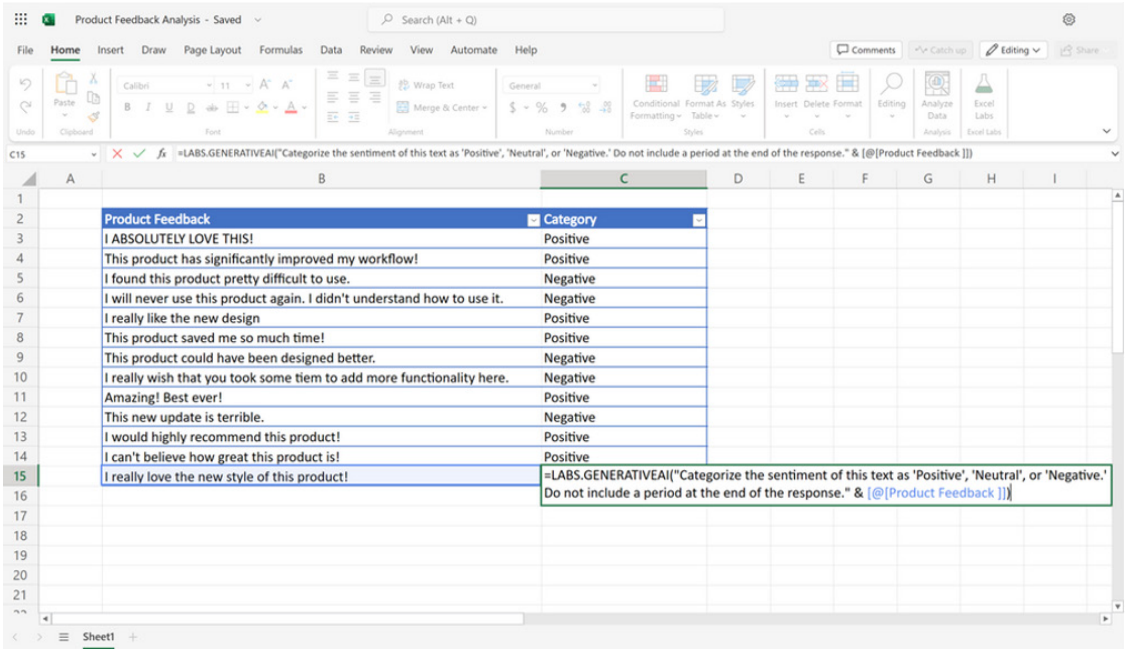
ADVANCED FORMULA ENVIRONMENT

The 'Advanced Formula Environment' feature is a tool designed to help you more easily author, edit and reuse complex formulae and **LAMBDA** functions. While the built-in Excel Name Manager lets you name and create complex formulae, the 'Advanced Formula Environment' adds to that experience by providing capabilities typically found in modern code editors, such as IntelliSense, commenting, inline errors, auto formatting and code collapse.



LABS.GENERATIVEAI CUSTOM FUNCTION

Considering the recent excitement surrounding generative AI, Microsoft is interested to see what types of Excel scenarios you have that would benefit from these AI models. Therefore, to better help you experiment, they have created the LABS.GENERATIVEAI custom function that enables you to prompt and display the response of OpenAI's large language models directly into the grid.



GENERATIVE AI

Generative AI models are large-scale language models that use machine learning to generate digital content like human-like text, code or even images. Given a prompt, the generative AI model returns a response that is generated by the model's algorithms.

The opportunities for using generative AI are virtually limitless as it can generate responses in a variety of different formats. For the LABS.GENERATIVEAI custom function, the software is specifically connecting to the OpenAI API for large language models. This includes models such as 'gpt-3.5-turbo' (the model that powers the infamous ChatGPT) and 'text-davinci-003'. Some of the most common use cases for these models include text generation, text completion, summarisation, classification, text transformation and Q&A.

For example, you can prompt it to parse out keywords in a survey response or you can ask it to analyse the sentiment of a table of tweets. In general, you are encouraged to experiment with different prompts to see how being more general or more specific impacts the responses you get. Instead of "Write a poem about Excel", try something like "Write a poem about Excel that mentions PivotTables". If you're interested, you can get started with the examples in the add-in or explore some common prompt examples on OpenAI's website.

Given the nature of large language models, it's important to understand that these models can produce nonsensical or even confident but false responses due to their predictive nature. Microsoft states that they are dedicated to building responsible AI, and the LABS.GENERATIVEAI custom function effort was created with their AI principles and the Responsible AI Standard in mind. Additionally, the OpenAI Moderation endpoint was incorporated into the feature for automatic filtering of responses.

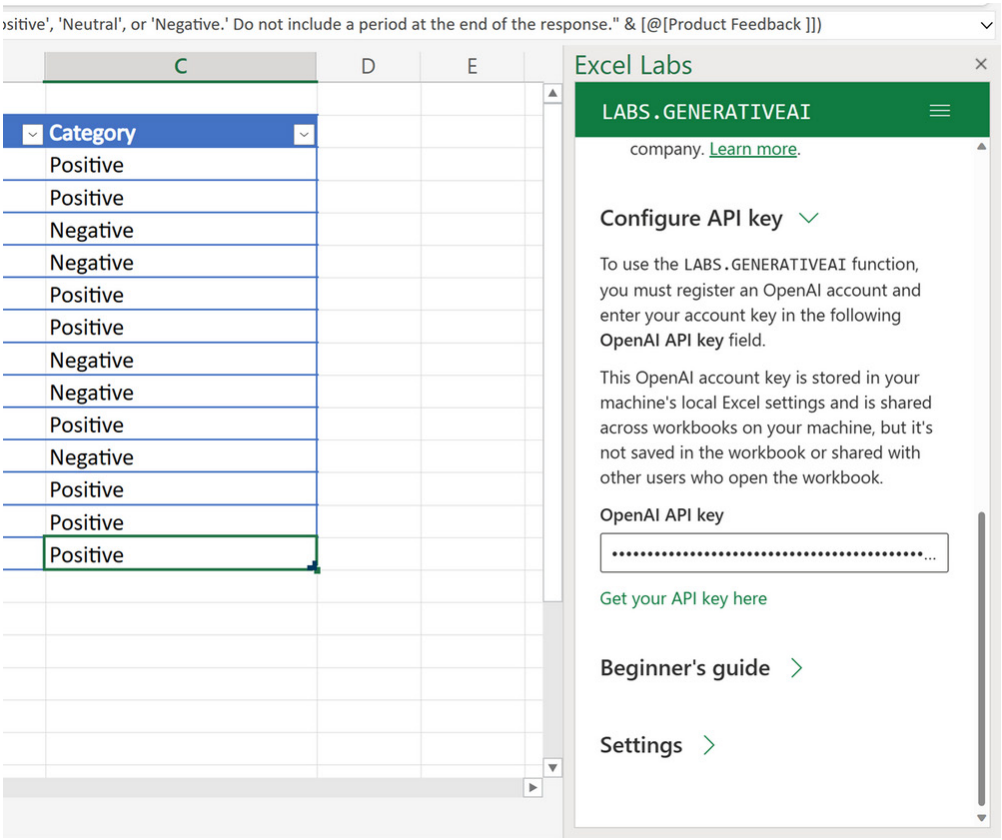
However, it is essential to remember that there are still considerations to be considered when using the AI system, such as verifying results before making decisions.

Overall, the extent to what generative AI can do is still being explored, and this is your chance to experiment with what it can do for you in the context of Excel.

USING THE LABS.GENERATIVEAI FUNCTION

When you select the LABS.GENERATIVEAI feature in the feature gallery, the first thing you need to do is add your OpenAI API key. If you don't already have an API key, you need to register for an OpenAI account and then request one.

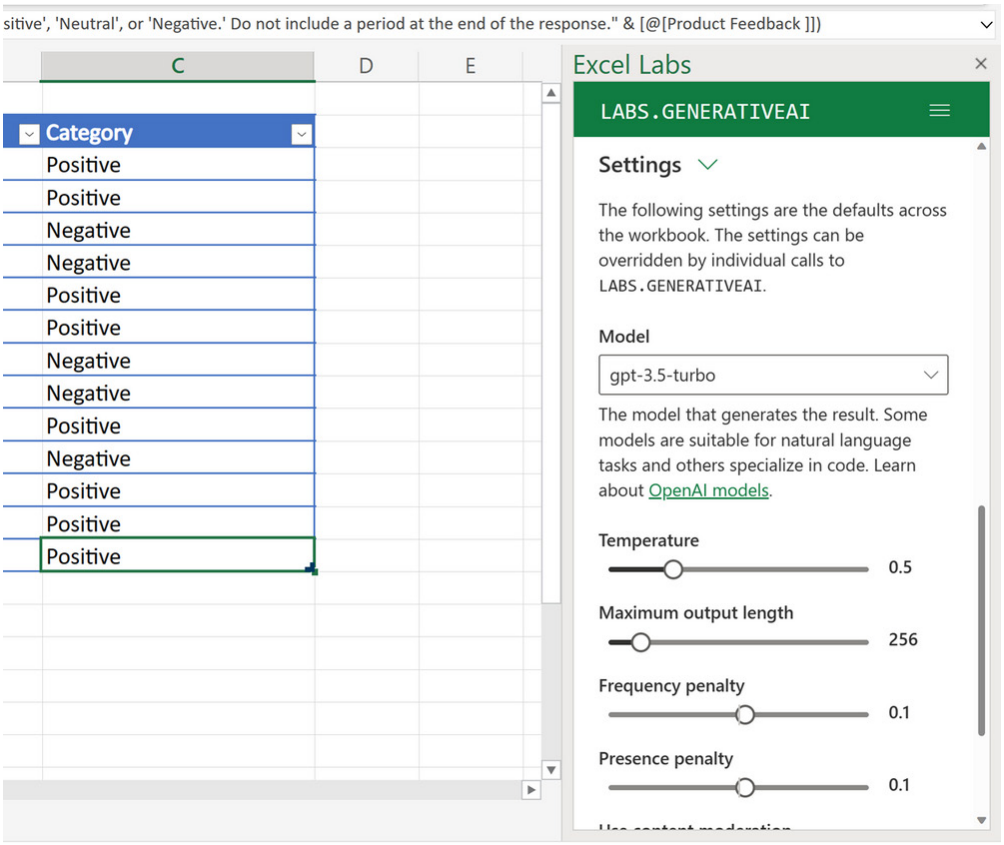
Once you have your key, add it in the task pane. You only need to do this step once.



Now, you're ready to try out the generative AI custom function. Enter =LABS.GENERATIVEAI to access the function in the grid and enter your prompt as the input to the function.

RESPONSE SETTINGS

Similar to OpenAI's Playgound, in the Excel Labs task pane, you can also adjust the settings of the model's responses by selecting the 'Generative AI model settings' button. This reveals different settings that affect the format of the responses. Two key settings you may want to experiment with are temperature and maximum output length. Temperature determines how consistent the returned responses will be. For instance, if you want to consistently get the same response from the model, you should set your temperature setting to zero [0]. The other setting you may want to play around with is the 'Maximum Output Length' setting which controls the maximum number of tokens returned in the response. Please be mindful that when you increase the number of tokens, you may hit Open AI's rate limit more quickly.



While you can adjust the settings in the task pane, you can also do so directly in the grid by using the function's optional parameters. For instance, the following function produces a response with a temperature value of one [1] and a maximum output length of 100 using the "text-curie-001" model.

=LABS.GENERATIVEAI("write me a poem about Excel", 1, 100, "text-curie-001")

Excel Labs was born from a collaboration between Excel and Microsoft Research Cambridge teams. They fundamentally believe in empowering Excel users to do more and feel that it's vital to continually expand a suite of tools that allow for experimentation and transformation of what's possible in Excel under the umbrella of Excel Labs.

Have fun playing!

LIFO Parole

Yes, we have considered modelling inventory previously, using both a simple averaging method to value the stock sold and then on a First In, First Out (FIFO) basis, which was a little bit trickier. Since then, we have been inundated with people asking me to complete the set: how do you model on a Last In, First Out (LIFO) basis? Well, we thought we would address it this newsletter…

There has been a little bit of naval gazing on this one (Pearl Harbo(u)r is my favo(u)rite) – not because I couldn't come up with an immediate solution, but that I tried to come up with a *cleverer* method. Well, truth is, dear reader, *I failed*. Time to come clean.

Before anything, let's be clear here. The LIFO method of inventory valuation is prohibited under International Financial Reporting Standards (IFRS), although it is permitted under the United States generally accepted accounting principles (US GAAP).

Why the controversy? LIFO may distort both the financial statements and financial analysis. There are several ways this can be allowed to happen. For example:

- assuming inventory costs increase over time, LIFO can understate a company's earnings for the purposes of keeping taxable income low
- the inventory valuation reported may be outdated, spoiled, expired or else obsolete
- in a liquidation scenario, earnings may be inflated artificially by selling off inventory with seemingly low carrying costs.

So do bear this all in mind before you start modelling this way! You might get "LIFO parole" if judged.

Having stepped off my sanctimonious soapbox, let's consider an example:

			1	2	3	4	5	6	7	8	9	10
Period No.	#											
Purchases	#		10	20	5	8	17	4	19	7	5	5
COGS Usage	#		8	14	7	5	9	18	5	14	13	6
Purchase Price	$ / unit		1.05	1.08	1.00	1.22	1.16	1.19	1.41	1.31	1.21	1.11

Let's jump to precisely what the problem is here. Can you provide me with the formula for deriving the costs of the inventory in Period 6? The problem is I can only reduce the requirement by four [4] given the inventory procured in the same period, but I am unsure what my stock levels are remaining for the items purchased in previous periods for different prices.

Algorithmically, I can figure it out:

- in Period 1, I purchase 10, sell eight [8], for $8.40 (8 x $1.05) and have two remaining
- in Period 2, I purchase 20 and sell 14. As I have sold less or equal to the number purchased, the LIFO cost is easy, I just calculate 14 x $1.08 which gives me $15.12. I am left with two remaining still from Period 1 and six in Period 2
- in Period 3, I purchase five [5] but sell seven [7]. Therefore, the first five sold are easy to cost (5 x $1.00) but I have to check the remainder for the previous period, which there are sufficient (six), so the remaining three may be calculated (2 x $1.08), giving me a total cost of $7.16. I am left with two still from Period 1, four from Period 2 and none remaining in Period 3
- in Period 4, I purchase eight [8] but only sell five [5], so the costing is easy again, being (5 x $1.22, which is) $6.10. This leaves remaining inventory of two from Period 1, four from Period 2 and three from Period 4
- Period 5 works along the same lines. Since the purchases (17) exceed the sales (9), my costs are trivial being (9 x $1.16, which equals) $10.44. This leaves remaining inventory of two from Period 1, four from Period 2, three from Period 4 and eight from Period 5
- finally, we get to Period 6. We only procured four [4], but we sold 18. The first four items are easy to cost (4 x $1.19), but the remaining 14 need to be sourced. Eight will come from Period 5 (8 x $1.16), three will come from Period 4 (3 x $1.22) and the remaining three from Period 2 (3 x $1.08). This will be a total cost of $20.94, leaving two remaining from Period 1 and one from Period 2.

See how awkward this is getting and the fact we are having to keep a running total of remaining inventory levels by period? It is easy when the items used is less than or equal to the items bought. The issue is when we have to raid the cupboard for older supplies. We need to know what is remaining and that depends upon what was used last period. However, that period's data depends upon what was used the period prior to that. And so on. This is an example of recursion and recursive formulae require Excel's latest *wunderkind* **LAMBDA**. Given this function is not available in all versions of Excel, I shall not be exploring that option here.

Therefore, unlike solutions for weighted average costing and First In, First Out (FIFO) methods, I shall have to fall back on what is known as a grid calculation. I am not able to construct a shortcut approach that will bypass this recursion. For those who fail to see the distinction between the other methods and the LIFO scenario, whilst it's true that both the weighted average and the FIFO methods required data from the previous period, they only needed aggregated totals; here, we have to keep tabs on what stock remains and in which period it was purchased, *i.e.* an ever-increasing array of values. Excel's in-cell calculations using "traditional, long served" functions do not appear to assist in this instance.

To be clear, not for a moment am I stating unequivocally there isn't some shortcut technique, it's just as at the time of writing, I haven't thought of one (and I have even started to get imaginative with some of the lesser known financial functions to try and concoct some Machiavellian approach). Alas, so far, it has eluded me.

Therefore, let's consider the grid approach:

	D	E	J	K	L	M	N	O	P	Q	R	S
11												
12		Period No.	1	2	3	4	5	6	7	8	9	10
13		Purchases	10	20	5	8	17	4	19	7	5	5
14		COGS Usage	8	14	7	5	9	18	5	14	13	6
15												
16		Purchase Price	1.05	1.08	1.00	1.22	1.16	1.19	1.41	1.31	1.21	1.11
17												
23												
24			1	2	3	4	5	6	7	8	9	10
25		1	2									
26		2	2	6								
27		3	2	4	-							
28		4	2	4	-	3						
29		5	2	4	-	3	8					
30		6	2	1	-	-	-	-				
31		7	2	1	-	-	-	-	14			
32		8	2	1	-	-	-	-	7	-		
33		9	2	-	-	-	-	-	-	-	-	
34		10	1	-	-	-	-	-	-	-	-	-

Do the values in rows 25:30 of the graphic look familiar? This is keeping a running total of what inventory still remains. As explained above, this is essential for working out costing and also useful for working out the age or remaining inventory, possible stockouts, *etc.*

In order to derive these numbers, I used a favourite function of mine…

Refresher on OFFSET

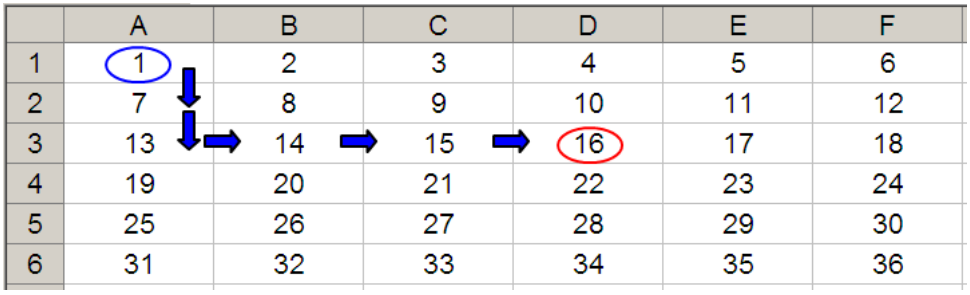
It seems every other article I bring this old chestnut up! The syntax for **OFFSET** is as follows:

OFFSET(reference, rows, columns, [height], [width])

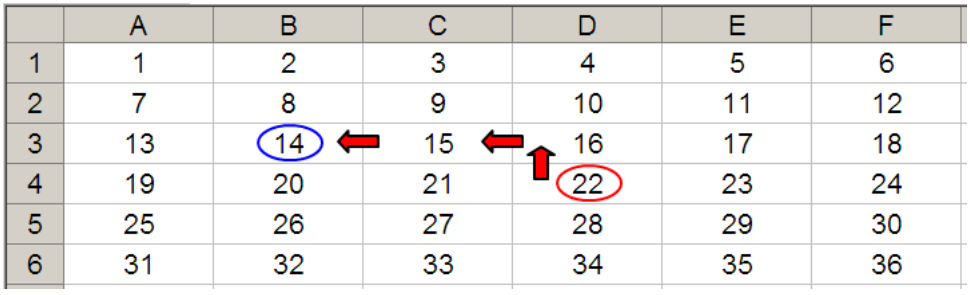
The arguments in square brackets (**height** and **width**) may be omitted from the formula, but I will need them here. In its most basic form, **OFFSET(reference, x, y)** will select a reference **x** rows down (**-x** would be **x** rows up) and **y** columns to the right (**-y** would be **y** columns to the left) of the reference **reference**. For example, consider the following grid:

	A	B	C	D	E	F
1	1	2	3	4	5	6
2	7	8	9	10	11	12
3	13	14	15	16	17	18
4	19	20	21	22	23	24
5	25	26	27	28	29	30
6	31	32	33	34	35	36

OFFSET(A1,2,3) would take us two rows down and three columns across to cell **D3**. Therefore, **OFFSET(A1,2,3)** = 16, *viz.*

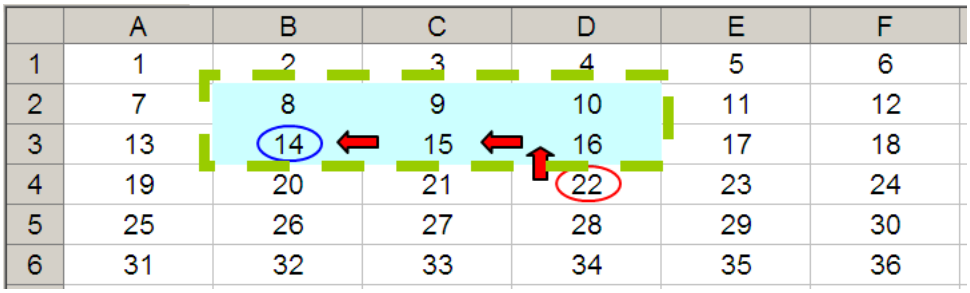


OFFSET(A1,2,3) would take us two rows down and three columns across to cell **D3**. Therefore, **OFFSET(A1,2,3)** = 16, *viz.*



OFFSET may also take advantage of the **height** and **width** arguments. If we extend the above formula to **OFFSET(D4,-1,-2,-2,3)**, it would again take us to cell **B3**, but then we would select a range based on the **height** and **width** parameters. The **height** would be two rows going up the sheet, with row 14 as the base (*i.e.* rows 13 and 14), and the **width** would be three columns going from left to right, with column **B** as the base (*i.e.* columns **B**, **C** and **D**).

Hence, **OFFSET(D4,-1,-2,-2,3)** would select the range **B2:D3**, *viz.*



Note that **OFFSET(D4,-1,-2,-2,3)** = *#VALUE!* where dynamic arrays are not recognised, since Excel cannot display a matrix in one cell, but it does recognise it. However, if after typing in **OFFSET(D4,-1,-2,-2,3)** we **press CTRL + SHIFT + ENTER**, we turn the formula into an array formula: **{OFFSET(D4,-1,-2,-2,3)}** (do not type the braces in, they will appear automatically as part of the Excel syntax). This gives a value of 8, which is the value in the top left-hand corner of the matrix, *but Excel is storing more than that*. This can be seen as follows:

- **SUM(OFFSET(D4,-1,-2,-2,3))** = 72 (*i.e.* **SUM(B2:D3)**)
- **AVERAGE(OFFSET(D4,-1,-2,-2,3))** = 12 (*i.e.* **AVERAGE(B2:D3)**).

SUM(OFFSET) and **OFFSET** will both be useful here.

Returning to LIFO

Let's revisit the grid:

		1	2	3	4	5	6	7	8	9	10
Period No.											
Purchases		10	20	5	8	17	4	19	7	5	5
COGS Usage		8	14	7	5	9	18	5	14	13	6
Purchase Price		1.05	1.08	1.00	1.22	1.16	1.19	1.41	1.31	1.21	1.11
		1	2	3	4	5	6	7	8	9	10
	1	2									
	2	2	6								
	3	2	4	-							
	4	2	4	-	3						
	5	2	4	-	3	8					
	6	2	1	-	-	-	-				
	7	2	1	-	-	-	-	14			
	8	2	1	-	-	-	-	7	-		
	9	2	-	-	-	-	-	-	-	-	
	10	1	-	-	-	-	-	-	-	-	-

```
=IF(J$24>$E25,"",IF(J$24=$E25,MAX(J$13-J$14,0),MIN(MAX(SUM(OFFSET(J24,,,$E25-J$24))+OFFSET($I$13,,$E25)-OFFSET($I$14,,$E25),0),J24)))
```

The formula in cell **J25** is

=IF(J$24>$E25,"",
IF(J$24=$E25,MAX(J$13-J$14,0),
MIN(MAX(SUM(OFFSET(J24,,,,$E25-J$24))+OFFSET($I$13,,$E25)-OFFSET($I$14,,$E25),0),J24)))

Lovely. I think this needs to be broken down!

Note that row 24 and the cells **E25:E34** both denote the period number, so for eight periods we would have an eight by eight grid, and here, we have a 10 by 10 matrix. Clearly, I cannot have any remaining inventory relating to periods later than the period presently being analysed, *e.g.* for Period 3, I could not take into account stock to be purchased in Period 4 or subsequently. This rules out half of my matrix. The formula

=IF(J$24>$E25,"", …

considers this. If the period number to be analysed is if the period of usage the formula returns empty data (**""**). So far, so good. The next period to consider is the actual period items are purchased and used (*i.e.* the leading diagonal of the grid), when the period number in the row and column are equal:

IF(J$24=$E25,MAX(J$13-J$14,0), …

In this situation, the remaining stock would be given by any excess of purchases over COGS usage, so for cell **J25** this would equate to **MAX(J$13-J$14,0)**. The restriction is so that this number may not go negative. It's a simple calculation. The issue is what follows!

MIN(MAX(SUM(OFFSET(J24,,,,$E25-J$24))+OFFSET($I$13,,$E25)-OFFSET($I$14,,$E25),0),J24)

Here, I have constructed a formula that is both replicable and scalable (this is why I have needed the **OFFSET** function).

Given the recursive nature of the calculation, perhaps starting in Period 1 is not ideal as this situation cannot occur (this calculation only comes into force when we are looking to use prior period inventory to service current year usage – in the first period, by definition, we will have no prior inventory in this example).

Therefore, it is probably more appropriate to consider the copied down formula in cell **J26** instead:

MIN(MAX(SUM(OFFSET(J25,,,,$E26-J$24))+OFFSET($I$13,,$E26)-OFFSET($I$14,,$E26),0),J25)

Here, let's first consider the difference between the purchases made in Period 2 (cell **K13**, 20) and the COGS usage (cell **K14**, 14). This would be given by the formula

K13-K14

Now the problem here is I want to copy this formula across the row so that it always references these cells:

$K13-$K14

OK, but when I copy the formula down a row I also need it to anchor to rows 13 and 14 respectively:

$K$13-$K$14

However, on the next row down I will want to reference rows **L13-L14**, *etc.* I would have to write a different formula each row, which would become cumbersome: imagine coding a 20 year monthly model. No thank you. This is why I use

OFFSET($I$13,,$E25)-OFFSET($I$14,,$E25)

This uses the cell before the first purchased amount and the first used amount and then moves one column across for every row of the table (so Period 1 will use column **J**, Period 2 will use column **K**, *etc.*). This now makes sense.

Word to the Wise

Apologies, it's not the simplest concept we have ever discussed, but this is a common problem in financial modelling. I wanted to show it for this reason – and because so many naysayers you cannot do this formulaically!

Some more advanced readers might consider using **SUMPRODUCT** and **INDEX** instead. This is because **SUMPRODUCT** cross-multiplies costs, and **INDEX** is not a so-called *volatile* function, which means it does not recalculate every time you press **ENTER**.

Given the name of my company is **SUMPRODUCT**, rest assured, dear reader, I have plenty of time for this function, but sometimes it slows down calculations, certainly compared to **SUM**. Similarly, **OFFSET** may be volatile, but it also often calculates more quickly than **INDEX**, especially given you would need multiple **INDEX** expressions. In larger models, you will find my formulae will calculate more quickly than using these alternatives – but it's not *wrong* to use them instead.

It would be a boring world if we all thought the same.

Beat the Boredom Challenge

*With many of us currently "working from home" / quarantined, there are only so Zoom / Teams calls and virtual parties you can make before you reach your (data) limit. Perhaps they should measure data allowance in blood pressure millimetres of mercury (mmHg). To try and keep our readers engaged, we will continue to reproduce some of our popular **Final Friday Fix** challenges from yesteryear in this and upcoming newsletters. One suggested solution may be found later in this newsletter. Here's this month's…*

How do you usually find and replace a character in a text string in Excel? Well, many people use the 'Find & Replace' functionality in Excel, which works fine for single replacements. However, if you have tens, hundreds or thousands of *different* items to replace this method seems very tedious and inefficient. Luckily, there are several ways to do the mass replacement in Excel more elegantly and effectively.

This month's challenge is to write a **formula** to replace multiple letters in some text strings. The replacement table (here, called **Replace**) might be as follows:

Old Character	New Character
ô	o
ê	e
ç	c
é	e
í	i
â	a
ǔ	u
ǎ	a
吳	Wu
奎	Kui
燁	Ye
雲	kumo
泥	doro
の	no
差	sa
1	Yi
2	Er
3	San
4	Si
5	Wu
6	Liu
7	Qi
8	Ba
9	Jiu
0	Ling

The result should look similar to the Table **Texts** *(below)*:

Old Text	New Text
Ngô Khôi Huê	Ngo Khoi Hue
Quân tử nhất ngôn	Quan tu nhat ngon
吳奎煒	Wu Kui Ye
520	Wu Er Ling
1314	Yi San Yi Si
雲泥の差	kumo doro no sa
Açaí da Paçoca	Acai da Pacoca
Marília Méndonça	Marilia Mendonca

As always, there are some requirements:

- the formula needs to be within just one column (no “helper” cells)
- this is a formula challenge; no Power Query / Get & Transform or VBA
- the formula should be dynamic enough to update when a similar text string is added
- only Excel 2016 functions are accepted.

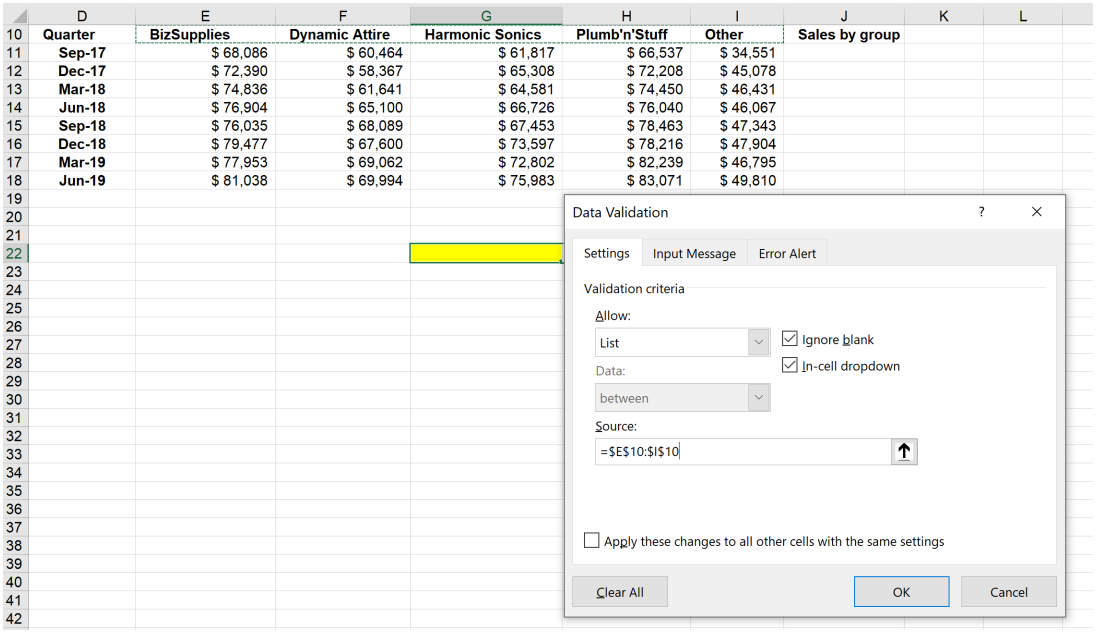
Sounds easy? Try it. One solution *just might* be found later in this newsletter – but no reading ahead!

Charts and Dashboards

It’s time to chart our progress with an introductory series into the world of creating charts and dashboards in Excel. This month, we again look at interactive charts, this time considering Data Validation in creating chart interactions.

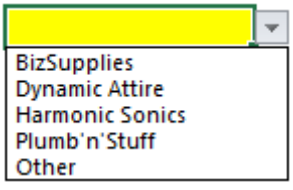
Last month, we discussed using Form Controls to create a drop-down list. This drop-down list may also be set up using Data Validation.

We will reuse the example from last time. Similar to then, we will add a ‘**Sales by group**’ column, which we will use to build a chart. Then, we will choose an empty cell in the worksheet, here the cell **G22** (highlighted in yellow), and navigate to **Data -> Data Validation**. In the pop-up ‘Data Validation’ dialog, when choosing the Source, just choose the area on which you wish to make a list; in this case, we’ll select the area from **BizSupplies** to **Other**, *i.e.* cells **E10:I10**.



	D	E	F	G	H	I	J
10	Quarter	BizSupplies	Dynamic Attire	Harmonic Sonics	Plumb'n'Stuff	Other	Sales by group
11	Sep-17	$ 68,086	$ 60,464	$ 61,817	$ 66,537	$ 34,551	
12	Dec-17	$ 72,390	$ 58,367	$ 65,308	$ 72,208	$ 45,078	
13	Mar-18	$ 74,836	$ 61,641	$ 64,581	$ 74,450	$ 46,431	
14	Jun-18	$ 76,904	$ 65,100	$ 66,726	$ 76,040	$ 46,067	
15	Sep-18	$ 76,035	$ 68,089	$ 67,453	$ 78,463	$ 47,343	
16	Dec-18	$ 79,477	$ 67,600	$ 73,597	$ 78,216	$ 47,904	
17	Mar-19	$ 77,953	$ 69,062	$ 72,802	$ 82,239	$ 46,795	
18	Jun-19	$ 81,038	$ 69,994	$ 75,983	$ 83,071	$ 49,810	

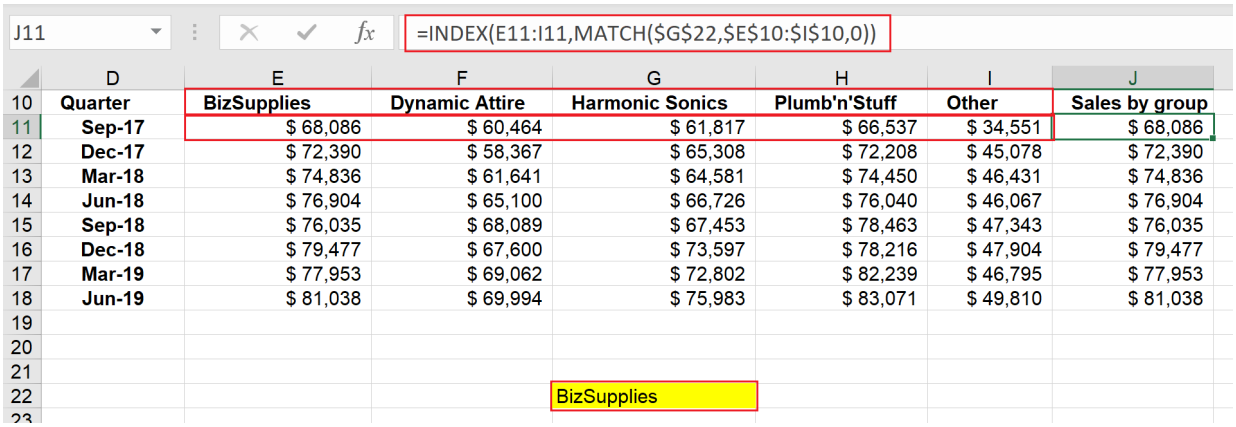
The validation cell is now active; if we click on the down arrow adjacent to the cell, it will show the list:



In the '**Sales by group**' column, we now can use the **INDEX** and **MATCH** functions to get the sales figure, based upon the group we choose from the validation list. Similarly, the '**Sales by group**' cell will pick one sales number from the sales array, if the validation cell matches with the product groups using an exact matching rule.

The formula in cell **J11** is:

=INDEX($E11:$I11,MATCH($G$22,$E$10:$I$10,0))

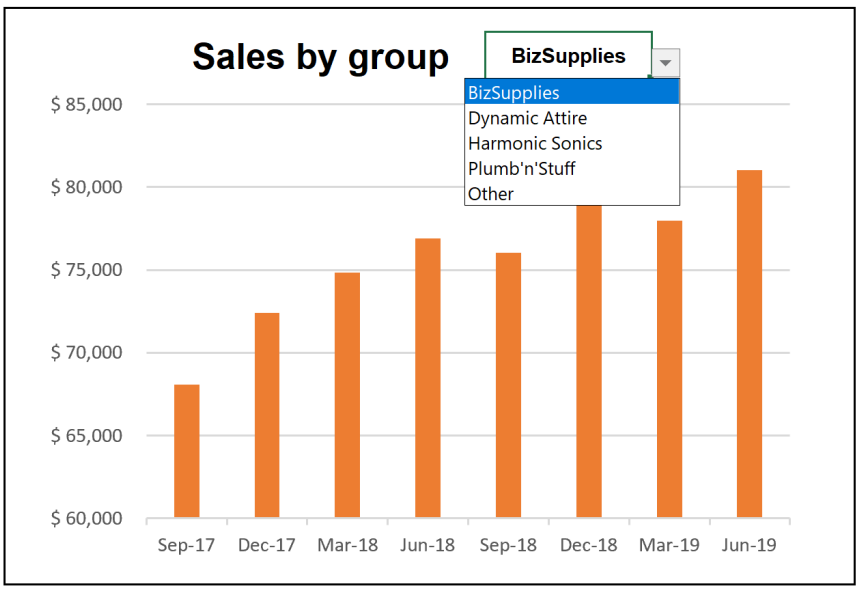


	D	E	F	G	H	I	J
10	Quarter	BizSupplies	Dynamic Attire	Harmonic Sonics	Plumb'n'Stuff	Other	Sales by group
11	Sep-17	$ 68,086	$ 60,464	$ 61,817	$ 66,537	$ 34,551	$ 68,086
12	Dec-17	$ 72,390	$ 58,367	$ 65,308	$ 72,208	$ 45,078	$ 72,390
13	Mar-18	$ 74,836	$ 61,641	$ 64,581	$ 74,450	$ 46,431	$ 74,836
14	Jun-18	$ 76,904	$ 65,100	$ 66,726	$ 76,040	$ 46,067	$ 76,904
15	Sep-18	$ 76,035	$ 68,089	$ 67,453	$ 78,463	$ 47,343	$ 76,035
16	Dec-18	$ 79,477	$ 67,600	$ 73,597	$ 78,216	$ 47,904	$ 79,477
17	Mar-19	$ 77,953	$ 69,062	$ 72,802	$ 82,239	$ 46,795	$ 77,953
18	Jun-19	$ 81,038	$ 69,994	$ 75,983	$ 83,071	$ 49,810	$ 81,038
19							
20							
21							
22				BizSupplies			

Building interactive charts from data validation requires a few steps:

- Insert a chart using '**Sales by group**' and **Quarter**
- Right-click on the horizontal axis, and choose **Format Axis Option -> Text Axis** to remove quarters with no data from the chart
- Right-click on the Chart Area, select **Format Chart Area -> Fill -> No Fill** and **Border -> No Line**
- Remove the Chart Title and type the alternative chart title on the cell next to the validation cell, then add some formatting to this title
- Drag the chart to the required position, then insert a shape with no fill to finish the chart formatting.

The chart should look like this once completed:



More next month…

Visual Basics

We thought we'd run an elementary series going through the rudiments of Visual Basic for Applications (VBA) as a springboard for newer users. This month we get in trim.

One thing a regular coder always has to consider is optimisation. Code optimisation is the process of modifying code to improve quality and efficiency. By looking doing code optimisation, programs can become more lightweight in size, consume less memory, execute more rapidly and perform fewer operations.

Let's consider this in light of the fact that the first introduction to VBA for most users is recording a macro. We're going to record a macro doing the following things in a fresh workbook:

- type "Hello World" in **A1**
- change **A1**'s fill colour to Red
- type the number one [1] in cell **A50**
- drag down the number to cell **A60**
- sum the values.

This is what the macro would record:

```
Sub Macro1()
' Macro1 Macro
    ActiveCell.FormulaR1C1 = "Hello World"
    Range("A1").Select
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .Color = 65535
        .TintAndShade = 0
        .PatternTintAndShade = 0
    End With
    ActiveWindow.SmallScroll Down:=27
    Range("A50").Select
    ActiveCell.FormulaR1C1 = "1"
    Range("A50").Select
    Selection.AutoFill Destination:=Range("A50:A60"), Type:=xlFillDefault
    Range("A50:A60").Select
    Range("A61").Select
    ActiveCell.FormulaR1C1 = "=SUM(R[-11]C:R[-1]C)"
    Range("A62").Select
End Sub
```

That's a lot of lines!

Notice what the Macro has done:

- it has recorded every specific step that was made. It's generated a **Range.Select** line whenever we moved the cursor to do an action upon a cell – it's even recorded the scroll movement made to get to cell **A50**
- When changing the cell fill colour, it's not just set the properties for the colour, but for everything else around it too.

Therefore, let's simplify our recorded macro.

Before we start, please note we will need to change the first line! It starts with **ActiveCell** because we just opened a workbook; it was set to cell **A1** and off it went. Let's reference it to **A1** explicitly because otherwise if the cursor is selecting any other cell then it will type "Hello World" there instead.

Thus, we will replace

```
ActiveCell.FormulaR1C1 = "Hello World"
```

with

```
Range("A1").FormulaR1C1 = "Hello World"
```

Next, we have our colour change statement. In order for us to be clear exactly what we are doing, we're only going to leave the **.Color** line. However, the **With** statement is unnecessary if we only have one line. Hence, we may remove that as well. Thus:

```
With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .Color = 65535
        .TintAndShade = 0
        .PatternTintAndShade = 0
    End With
```

becomes

```
Selection.Interior.Color = 65535
```

Let's remove the scroll movement altogether. There is no need for the program to change the screen, that scroll movement was an action that I had to take, not the program! Hence, we shall delete

```
ActiveWindow.SmallScroll Down:=27
```

Finally, we can remove **.Select** statements. Notice that after a **.Select** line there is a **Selection** or **ActiveCell** statement. We may simply replace those with the **Range** that was selected instead. Therefore, for example:

```
Range("A50").Select
ActiveCell.FormulaR1C1 = "1"
```

can simply become

```
Range("A50").FormulaR1C1 = "1"
```

So how will our end code look after all of these modifications?

```
Sub Macro1()
' Macro1 Macro
    Range("A1").FormulaR1C1 = "Hello World"
    Range("A1").Color = 65535
    Range("A50").FormulaR1C1 = "1"
    Range("A50").AutoFill Destination:=Range("A50:A60"), Type:=xlFillDefault
    Range("A61").FormulaR1C1 = "=SUM(R[-11]C:R[-1]C)"
End Sub
```

Wow! We've trimmed the recorded macro which generated 18 lines of code down to five [5] lines – which is more indicative of the five actions that were performed.

Recording macros is a great way to get started especially if it's a task you've done over and over again and unsure of syntax, but always look at ways to trim down the code. Each additional line of code is more operations that VBA will do and require more system resources. Code optimisation is not out of reach for the novice VBA user!

Next month, we'll look at more advanced optimisation tricks.

Power Pivot Principles

We continue our series on the Excel COM add-in, Power Pivot. This month, we discuss how to create conditional columns with multiple criteria.

This month, we discuss the different methods to create conditional custom columns in Power Pivot. Let's have a look at the following data set:

	A	B	C	D	E
1					
2		Category 1	Category 2	Category 3	Customer Name
3		Y	Y	0	Harry
4		Y	N	1	Joe
5		N	Y	1	Kat
6		N	N	1	Nicole
7		Y	Y	2	Monia
8		N	Y	0	Jeff
9		N	Y	0	Petra
10		N	N	1	Lenny
11		Y	Y	2	Gavin
12					

We have a table with three categories and a column with customer names. We are going to assume that this table has been retrieved by Power Query (otherwise known as Get & Transform) and loaded into our data model in Power Pivot.

For the purposes of this exercise, let's assume that we wish to create a conditional column that will return with the value "Included" if a customer is "Y" in Category 1 and "N" in Category 2.

We would use the following code to create our custom column:

```
=IF(AND([Category 1]="Y", [Category 2]="N"),"Included",BLANK())
```

As we can see our logic works and we get the desired result:

[Included] f_x =IF(AND([Category 1]="Y", [Category 2]="N"),"Included",BLANK())

	Category 1	Category 2	Category 3	Customer Name	Included
1	Y	Y	0	Harry	
2	Y	N	1	Joe	Included
3	N	Y	1	Kat	
4	N	N	1	Nicole	
5	Y	Y	2	Monia	
6	N	Y	0	Jeff	
7	N	Y	0	Petra	
8	N	N	1	Lenny	
9	Y	N	2	Gavin	Included

What if we want a conditional column with three conditions?

- Category1 = "Y"
- Category2 = "Y"
- Category3 > 1

Keeping in mind that the **AND** function in Power Pivot only allows for two logical statements:

```
AND(Logical 1, Logical 2)
```

We will have to use a dreaded nested **IF** formula:

```
=IF(AND([Category 1]="Y",[Category 2] = "Y"),IF([Category 3]>1,"Flag",BLANK()))
```

We achieve the intended result:

[Flag] =IF(AND([Category 1]="Y",[Category 2] = "Y"),IF([Category 3]>1,"Flag",BLANK()))

	Category 1	Category 2	Category 3	Customer Name	Included	Flag
1	Y	Y	0	Harry		
2	Y	N	1	Joe	Included	
3	N	Y	1	Kat		
4	N	N	1	Nicole		
5	Y	Y	2	Monia		Flag
6	N	Y	0	Jeff		
7	N	Y	0	Petra		
8	N	N	1	Lenny		
9	Y	N	2	Gavin	Included	

That's all well and good, but what if we have four or five criteria? The nested **IF** functions will just continue to grow and grow: is there an alternative to a nested **IF** formula?

We can use the '**&&**' operator instead. The '**&&**' operator serves as a substitute to the **AND** function so we can write the following formula:

```
=IF([Category 1]="Y" && [Category 2] = "Y" && [Category 3]>1,"Flag",BLANK())
```

[Flag1] =IF([Category 1]="Y" && [Category 2] = "Y" && [Category 3]>1,"Flag",BLANK())

	Category 1	Category 2	Category 3	Customer Name	Included	Flag	Flag1
1	Y	Y	0	Harry			
2	Y	N	1	Joe	Included		
3	N	Y	1	Kat			
4	N	N	1	Nicole			
5	Y	Y	2	Monia		Flag	Flag
6	N	Y	0	Jeff			
7	N	Y	0	Petra			
8	N	N	1	Lenny			
9	Y	N	2	Gavin	Included		

The '**&&**' operator allows us to combine several criteria into one logical test for the **IF** formula drastically simplifying the formula. That being said, the '**&&**' operator is a great way to avoid creating nasty nested **IF** formulae when creating our custom columns in Power Pivot.

That's it for this month; tune in next month when we cover the **OR** operator.

Power Query Pointers

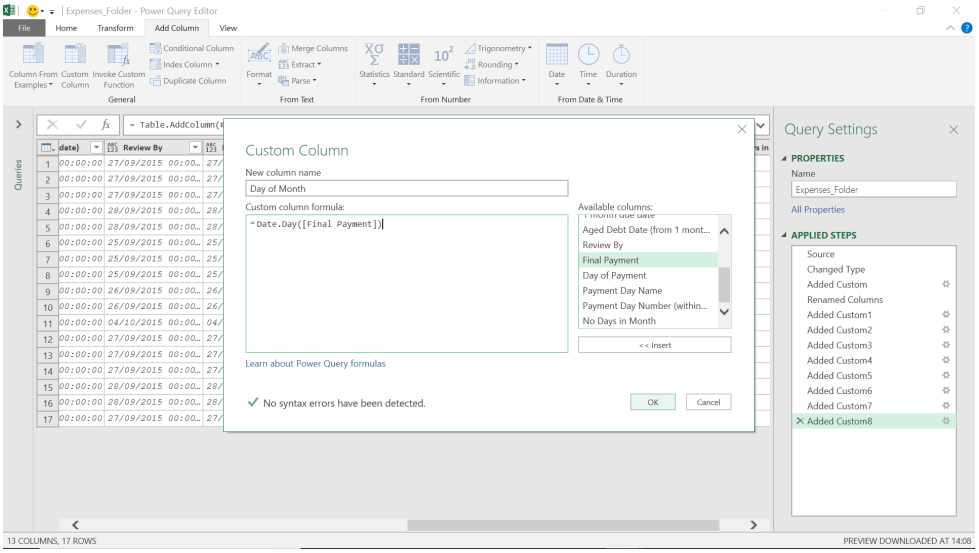
*Each month we'll reproduce one of our articles on Power Query (Excel 2010 and 2013) / Get & Transform (Office 365, Excel 2016 and 2019) from www.sumproduct.com/blog. If you wish to read more in the meantime, simply check out our Blog section each Wednesday. This month, we review some useful ways to extract data from dates using **M**.*

This month, we will take a look as some **Date() M** functions that can be used extract data from an existing date, for example, when we want to know the day or year of a particular date. As usual, we will provide an example for each function.

Date.Day

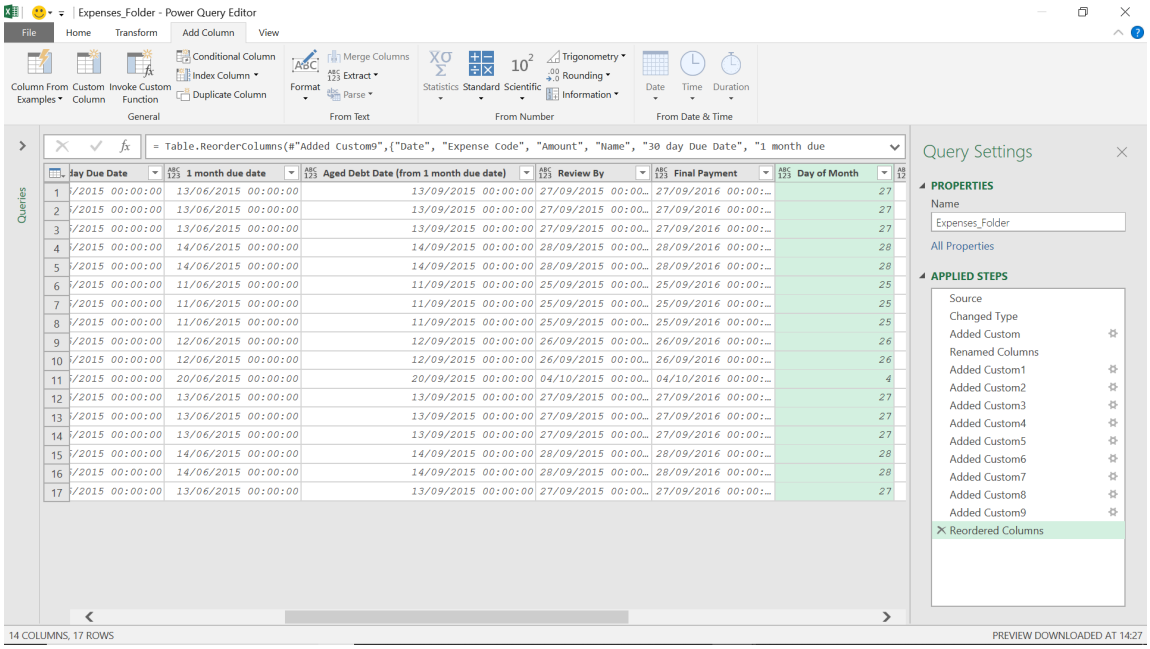
Date.Day(datetime as datetime**)** as nullable number

This returns the day for a **datetime** value. This is a simple **M** function, which will extract the day of the month:



The **M** functionality used is

= Date.Day([Final Payment])



The day of the month has now been extracted.

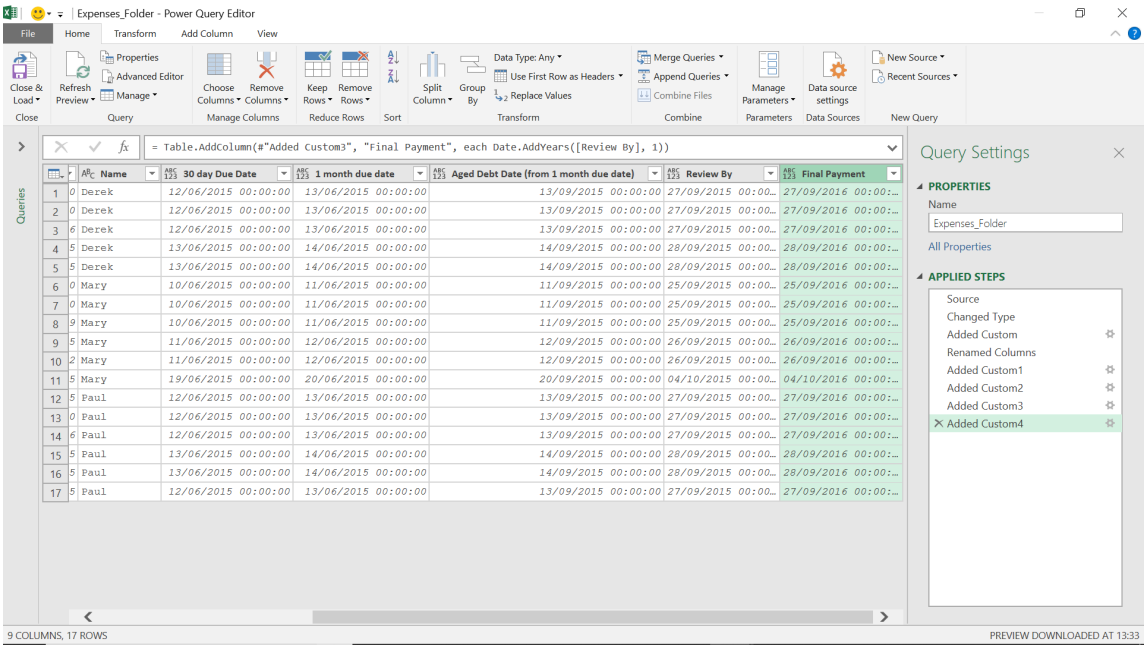
Date.DayOfWeek

Date.DayOfWeek(datetime as any**,** optional **firstDayOfWeek** as nullable number**)** as nullable number

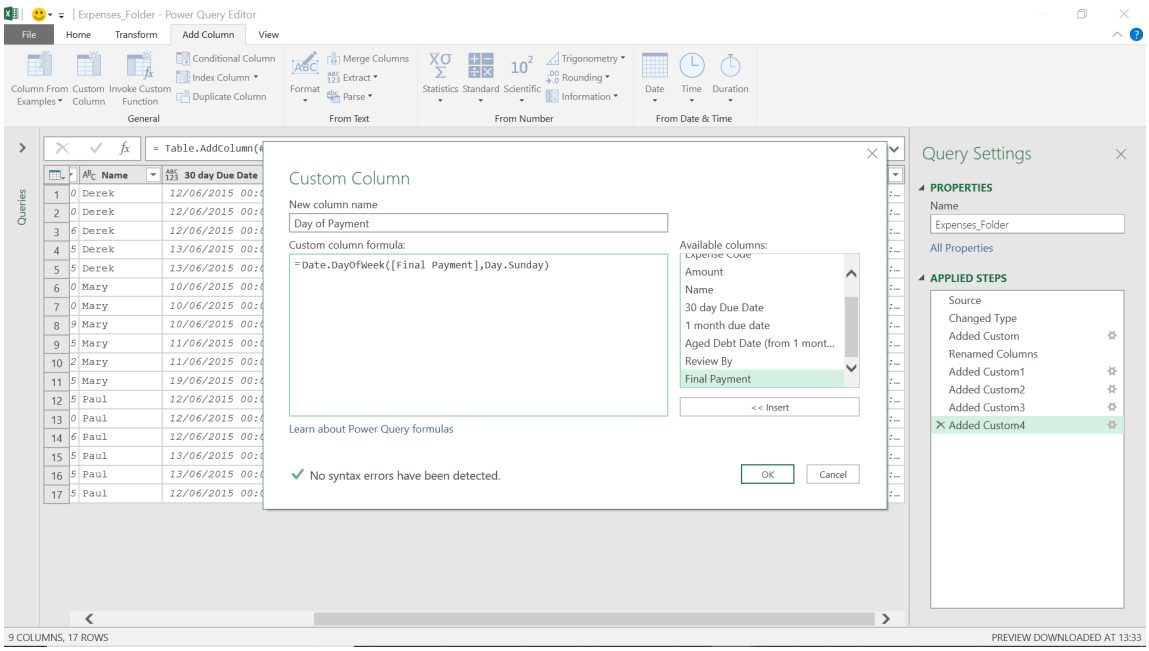
This returns a number between zero [0] and six [6], representing the day of the week in the provided **datetime** value. This function takes an optional day value, **firstDayOfWeek**, to set the first day of the week for this relative calculation. Valid values are: **Day.Sunday**, **Day.Monday**, **Day.Tuesday**, **Day.Wednesday**, **Day.Thursday**, **Day.Friday**, and **Day.Saturday**.

We have not included the Microsoft help information about the default. The help says that the default is that the week starts on a Sunday, but when we tested it, the default is Monday! If we put a second parameter in, then I can indicate that the week should start on a Sunday (we may use **Day.Sunday** as the parameter). We might like to think that the week starts on a Friday, and we can set the function to agree with us, but we can't think of a business reason to do this!

Let's use the expense data for our fictional salespeople, to find out more about the date that we will be paying them their expenses.



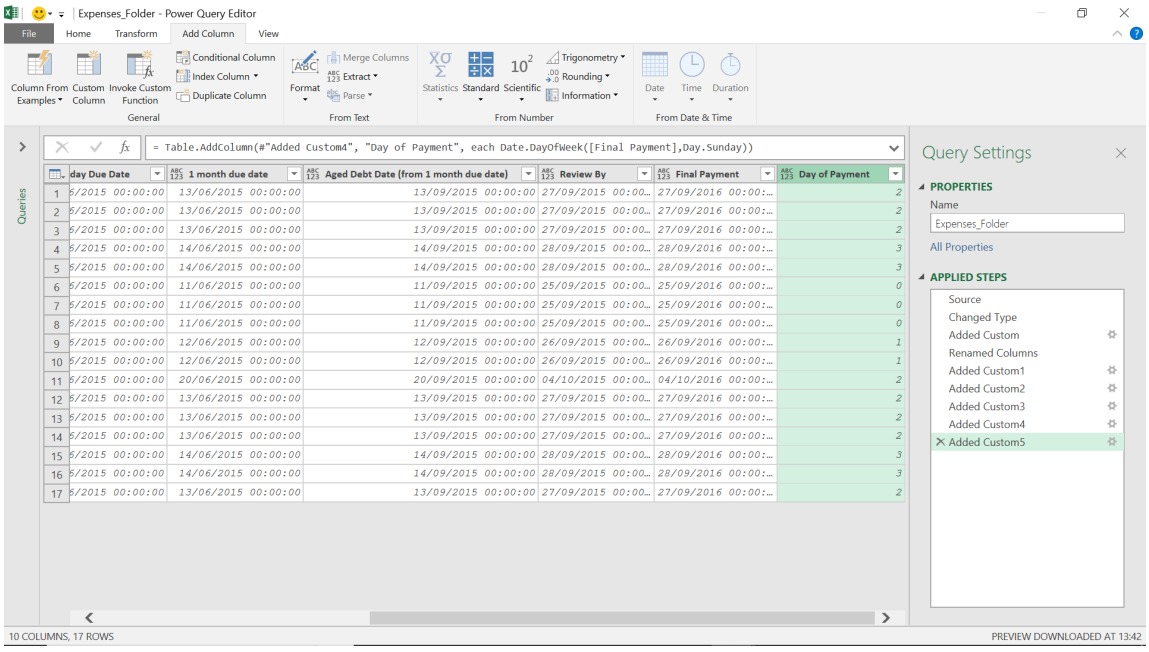
We create a new custom column to show us the day.



The **M** formula used is

= Date.DayOfWeek([Final Payment], Day.Sunday)

We have specified that the week starts on a Sunday.

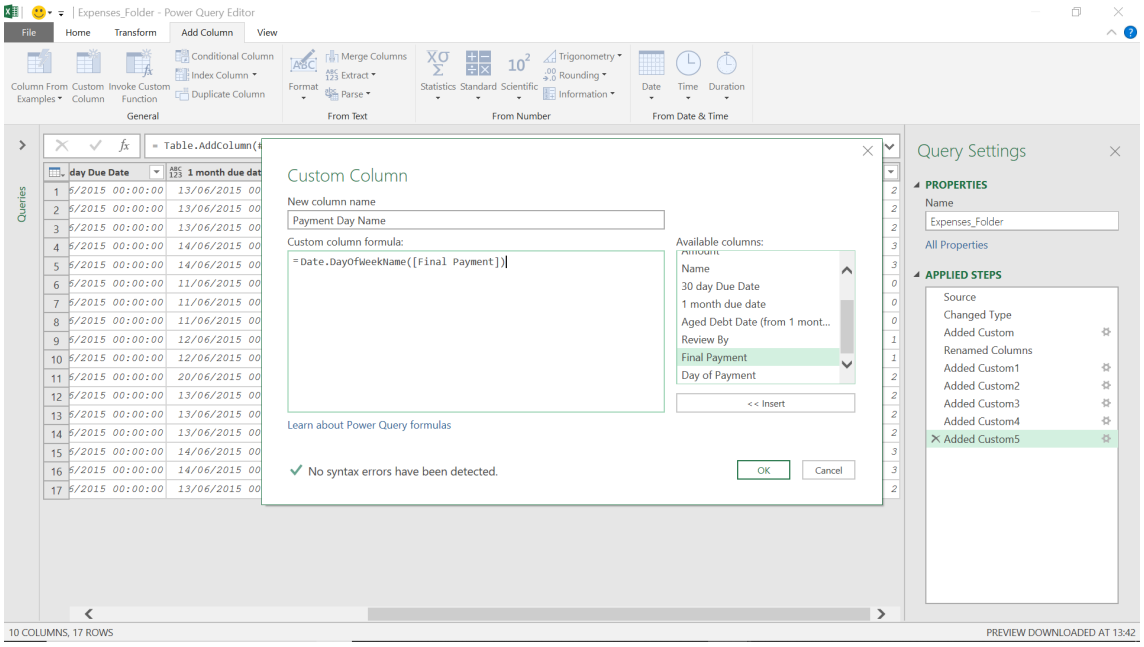


We have our day of payment represented as a number, so that zero [0] is Sunday, one [1] is Monday and so on…

Date.DayOfWeekName

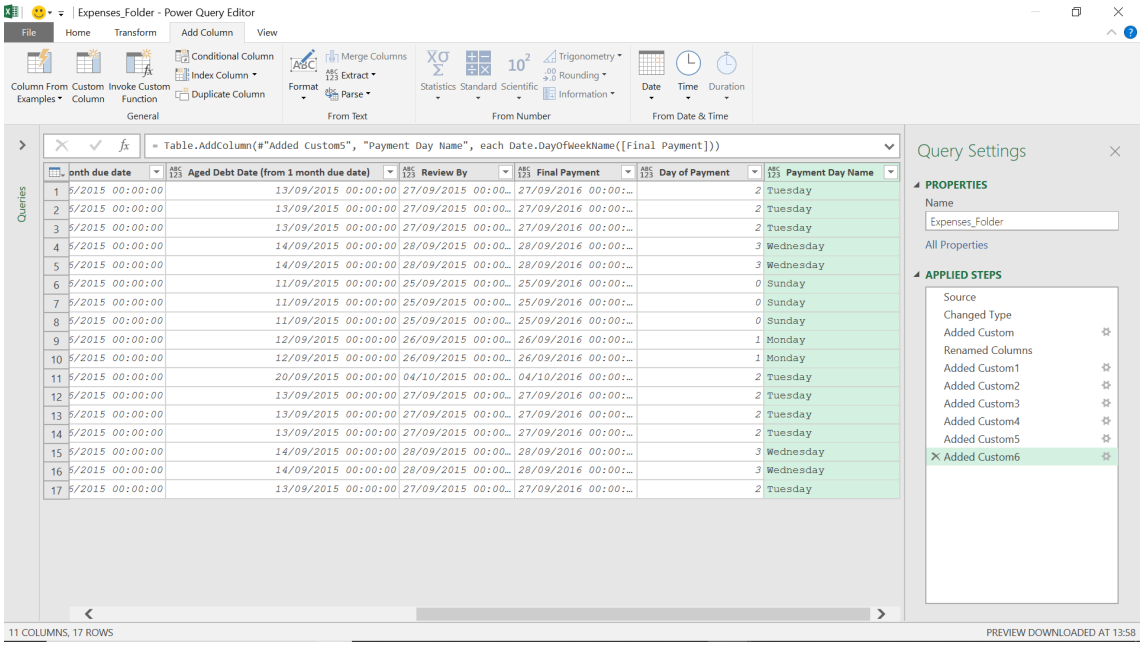
Date.DayOfWeekName(date as any**,** optional **culture** as nullable text**)**

This returns the day of the week name for the provided **date** and, optionally, a **culture**. An example of culture is 'en-US', to give me US English



The **M** functionality used is

= Date.DayOfWeekName([Final Payment])

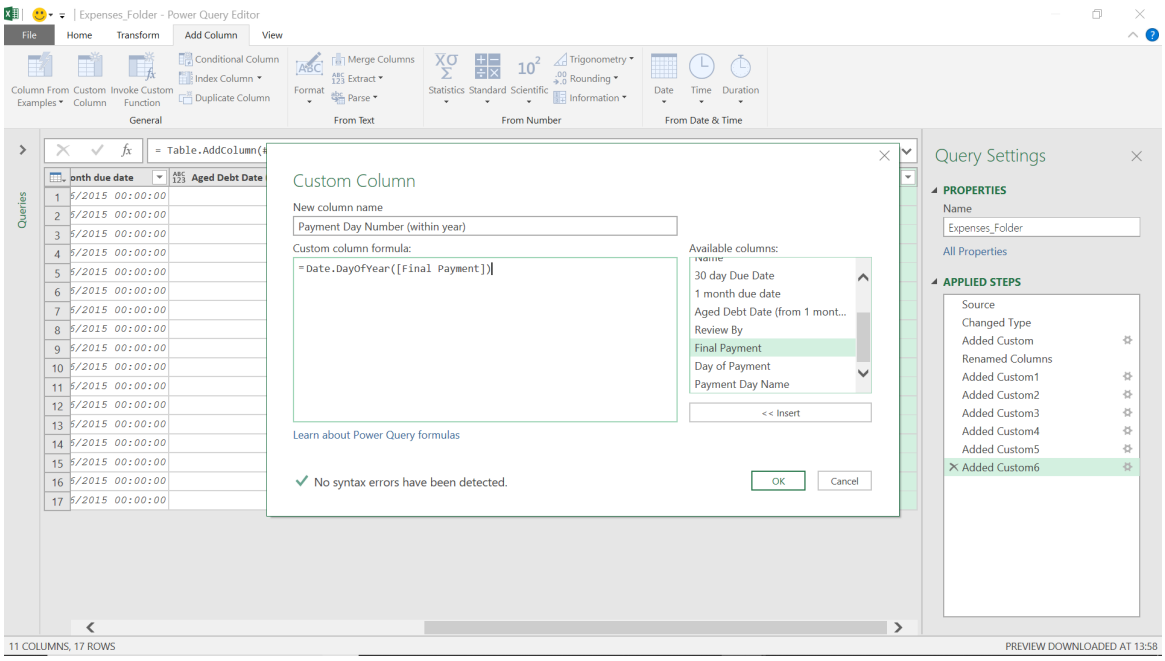


The name of the day is shown.

Date.DayOfYear

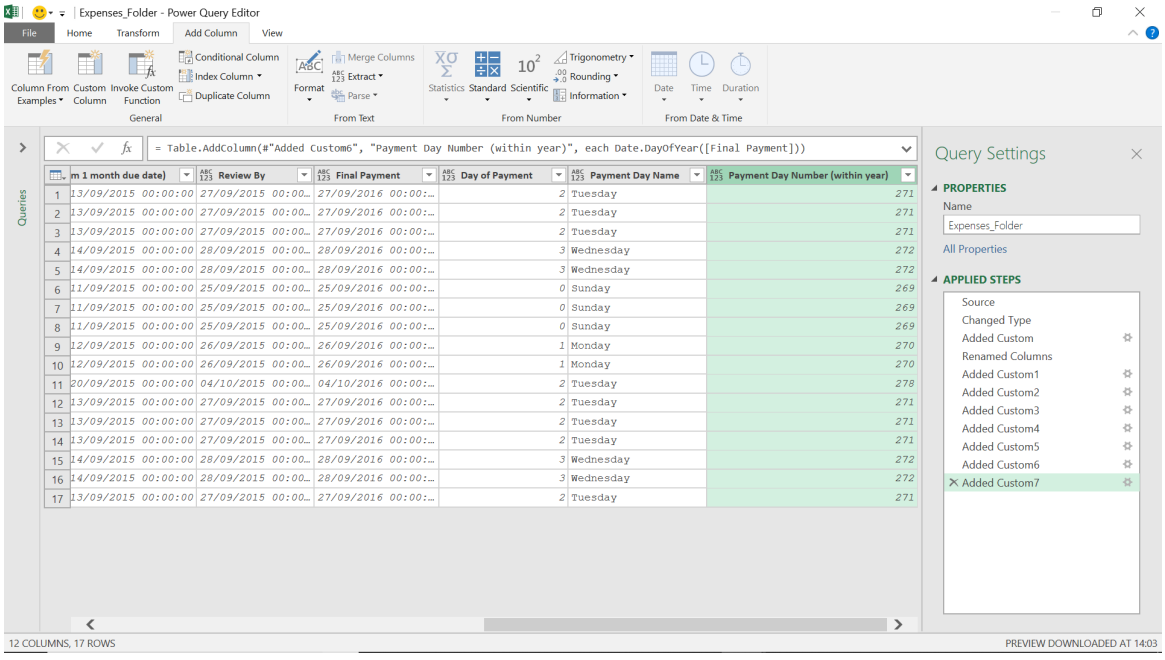
Date.DayOfYear(datetime as datetime**)** as nullable number

This returns a number that represents the day of the year from a **datetime** value. This will provide us with the day number within the year that the final payment takes place.



The **M** formula used is

= Date.DayOfYear([Final Payment])

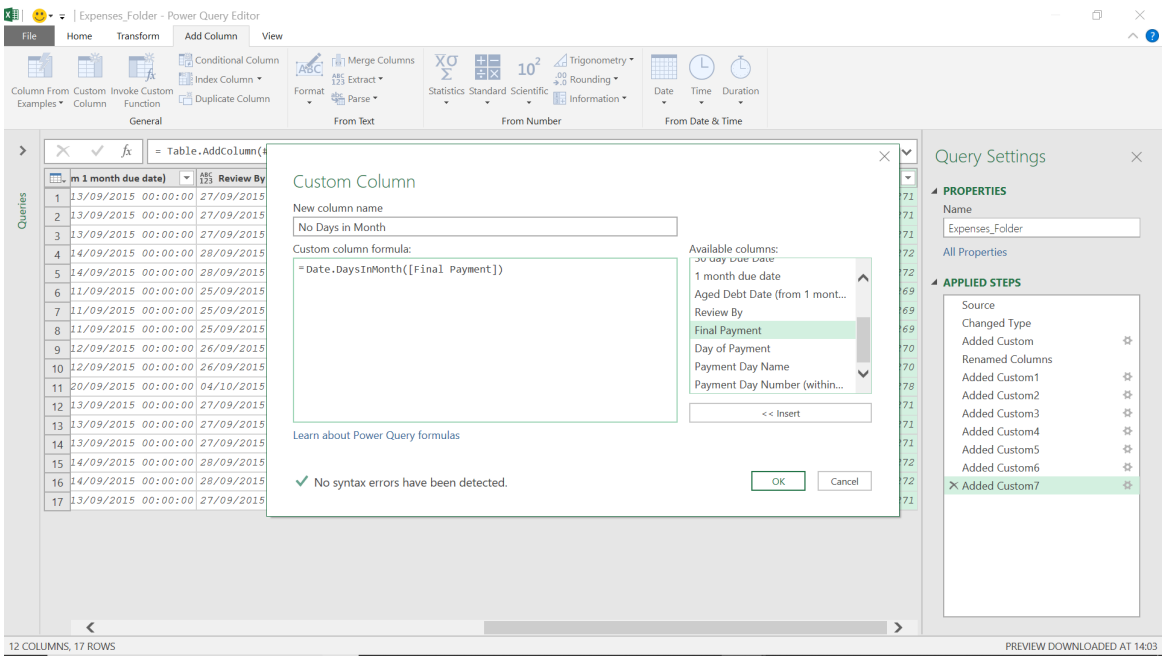


We have the day number (in terms of the year) that our payment takes place.

Date.DaysInMonth

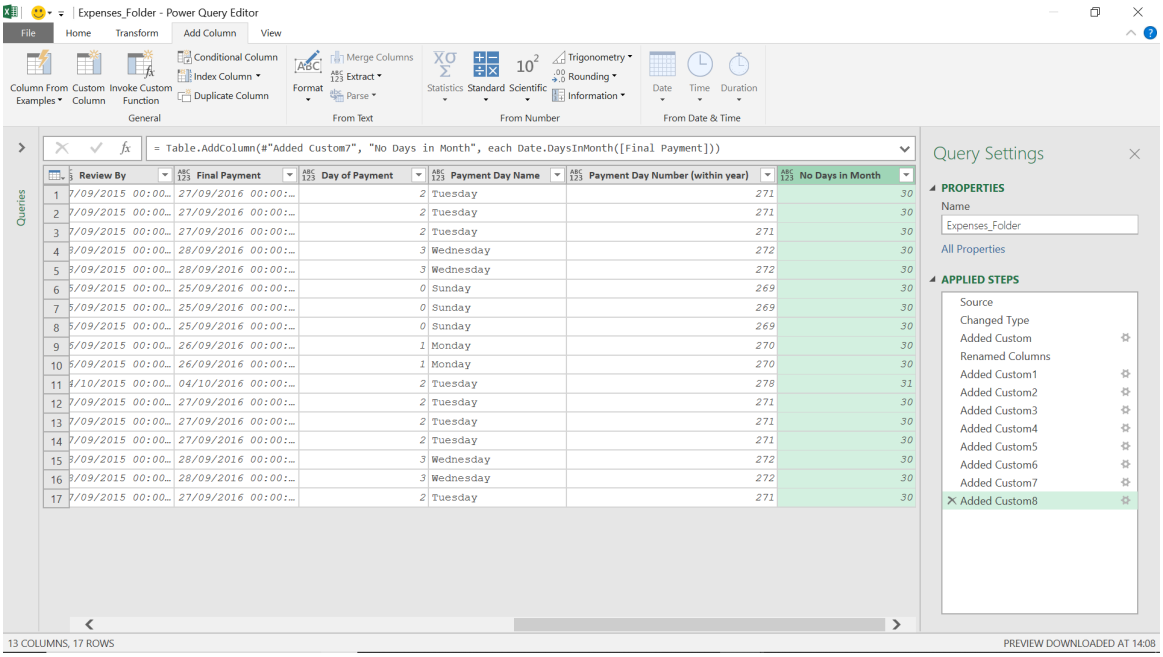
Date.DaysInMonth(datetime as datetime**)** as nullable number

This returns the number of days in the month from a **datetime** value. This function could be useful when comparing sales between months. For example, if we are comparing July and February then it makes sense to take into account the number of days involved.



The **M** functionality used is

= Date.DaysInMonth([Final Payment])



The number of days in the month of the final payment is shown. There are a number of other extractions that we may do, and we have summarised all of them in the final image at the end of this article.

Date.Year

Date.Year(datetime as datetime**)** as nullable number

This returns the year from a **datetime** value. An example **M** formula might be (used in the final image)

= Date.Year([Final Payment])

Date.Month

Date.Month(datetime as datetime**)** as nullable number

This returns the month from a **datetime** value. An example **M** formula might be (used in the final image)

= Date.Month([Final Payment])

Date.MonthName

Date.MonthName(date as any**,** optional **culture** as nullable text**)**

This returns the name of the month component for the provided **date** and, optionally, a **culture**. An example **M** formula might be (used in the final image)

= Date.MonthName([Final Payment])

Date.QuarterOfYear

Date.QuarterOfYear(datetime as datetime**)** as nullable number

This returns a number between one [1] and four [4] for the quarter of the year from a **datetime** value. An example **M** formula might be (used in the final image)

= Date.QuarterOfYear([Final Payment])

Date.WeekOfMonth

Date.WeekOfMonth(datetime as datetime**)** as nullable number

This returns a number for the count of week in the current month. An example **M** formula might be (used in the final image)

= Date.WeekofMonth([Final Payment])

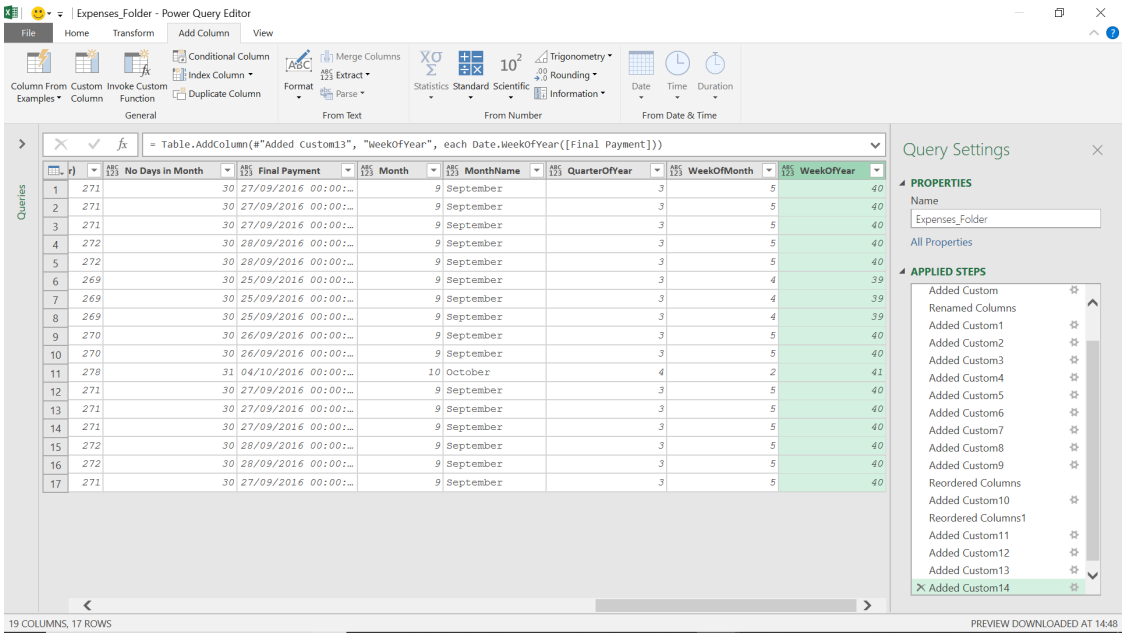
Date.WeekOfYear

Date.WeekOfYear(datetime as datetime**)** as nullable number

This returns a number for the count of week in the current year. An example **M** formula might be (used in the final image)

=Date.WeekOfYear([Final Payment])

This final screenshot shows all of the remaining functions:



More next time.

Power BI Updates

Just for a change – we have another update to communicate – hard to believe, we know!

From this month, Power BI Desktop is now fully supported on Azure Virtual Desktop (formerly Windows Virtual Desktop) and Windows 365. But that isn't all. There are also updates to the Preview feature On-object that was announced in last month's newsletter and dynamic format strings for measures. And there are also additional features in Reporting, Data Connectivity, Service, Paginated Reports, Mobile and Visualizations.

The full list is as follows:

Reporting

- Dynamic format strings for measures in Preview
- On-object interaction: updates in Preview
- New ToolTip auto-scale in Preview

Analytics

- Update to Quick Measure Suggestions in Preview

Modelling

- Composite models on Power BI Datasets and Analysis Services models now Generally Available
- Updates to the **ORDERBY** function
- New DAX functions: **RANK** and **ROWNUMBER**

Data Connectivity

- Oracle database (connector update)

Service

- New features to Deployment Pipeline:
  - View schema changes line-by-line
  - Choose whether to continue the deployment in case of a failure
- Storytelling in PowerPoint: new Style option
- Visualising views in Power Apps with Power BI Quick Report is now enabled by default

Paginated Reports

- Paginated Reports feature summary

Mobile

- Enhanced ToolTips on visuals in the Power BI Mobile app
- Hierarchies are now supported in the Power BI Mobile app

Visualisations

- Acterys Reporting Suite 3.0: reporting
- Box and Whisker Chart by MAQ Software
- Feature summary for Drill Down Graph PRO
- DualCard visual.

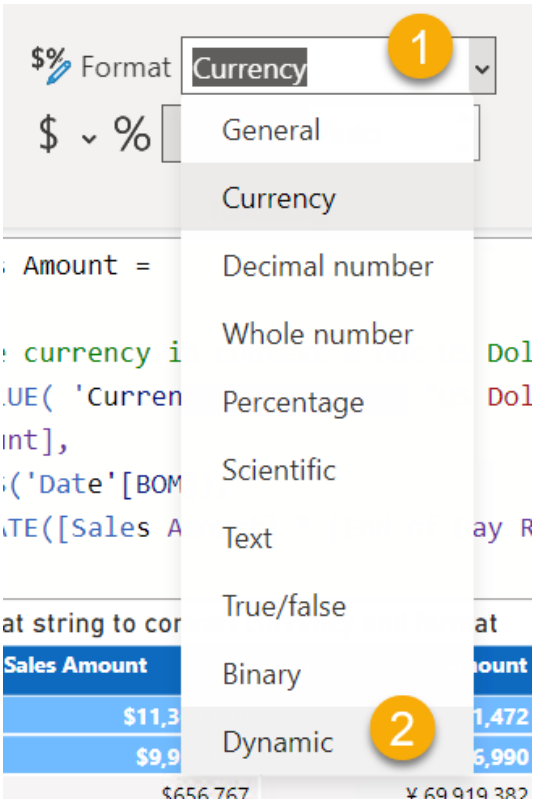
Let's look at each in turn.

Dynamic format strings for measures in Preview

Ever wanted to format a measure differently based upon a slicer selection or some other conditional way? Well now you can. This month's update begins with the public Preview of dynamic format strings for measures.

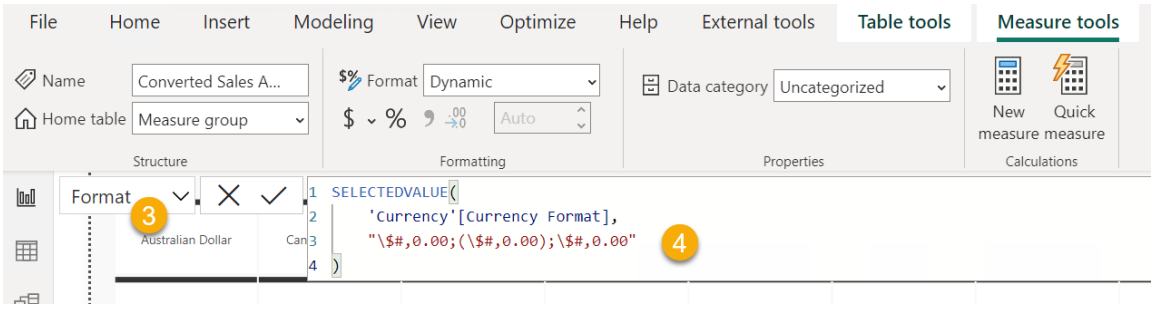
In a measure, you define the measure itself with a DAX expression, but until now you could only provide that measure a static format string such as Currency, Whole number, Decimal, *etc*. or a type in the static format string, such as "dd MMM yyyy" for "09 Mar 2023". With dynamic format strings (it doesn't sound grammatically correct to me – I hope they aren't "very hidden"!), you can create that format string also using a DAX expression. This provides you with the flexibility to adjust the format string to a variety of contexts within a report.

For example, a common scenario for this is currency conversion. If you have the currency format strings in your Currency table, you can define a DAX expression to use it. To add a dynamic format string to a measure, click the measure in the Data pane, then in the Measure tools ribbon Format dropdown choose 'Dynamic'.

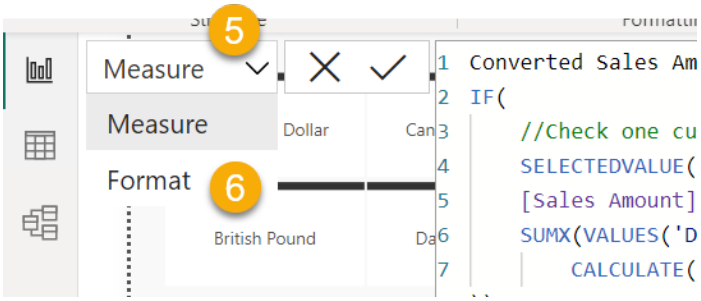


A new dropdown will appear to the left of the DAX formula bar, and it will be on 'Format'. By default, whatever static format string corresponds to the previous Format dropdown will be pre-populated to get you started, but you can also delete it and use whatever DAX expression you want for your dynamic format string. In this example, it's looking up the string from 'Currency' [Currency Format], and if that is ambiguous, then using

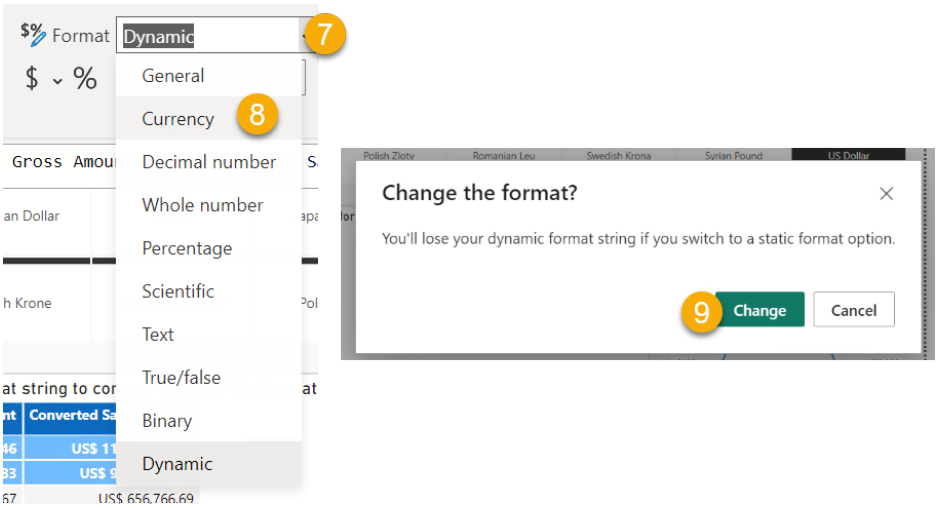
"\$#,0.00;(\$#,0.00);\$#,0.00"



If you want to get back to your measure DAX expression, you can change that left dropdown to 'Measure'.



Finally, if you want to remove the dynamic format string, go back to the Format dropdown, and choose any of the other options available. A warning dialog will appear as you will not be able to undo this action – so be careful!

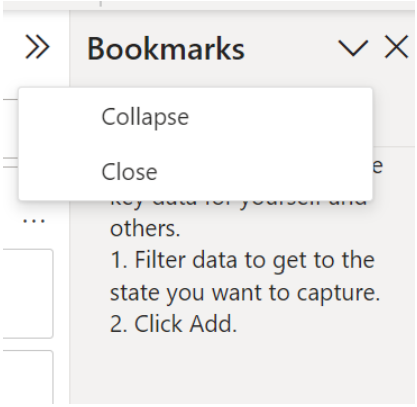


Dynamic format strings are not new, and those of you who are familiar with calculation groups in SQL Server Analysis Services, Azure Analysis Services, and / or Power BI using external tools may know calculation items (supported in some tabular models, these can significantly reduce the number of redundant measures by grouping common measure expressions as calculation items) already have dynamic format strings. These same dynamic format string DAX patterns can now be utilized in individual measures in Power BI too.

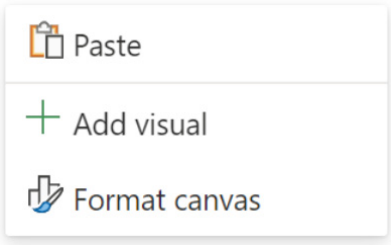
On-object interaction: updates in Preview

The new On-object interaction feature was released in Preview just recently. Whilst it is still being tinkered with (a technical term), improvements have been added in this release – albeit still in Preview:

- **Card visual** is now supported
- **Behaviour update:** mow when a user explicitly chooses a different visual type than what is suggested to them (top five [5] or full gallery) auto mode is switched off. This is to address feedback around the auto mode unexpectedly changing visuals on users after they already switched the type
- **Chevron menu** added to panes to allow for easy minimising (collapsing)



- **Paste** added to the canvas context



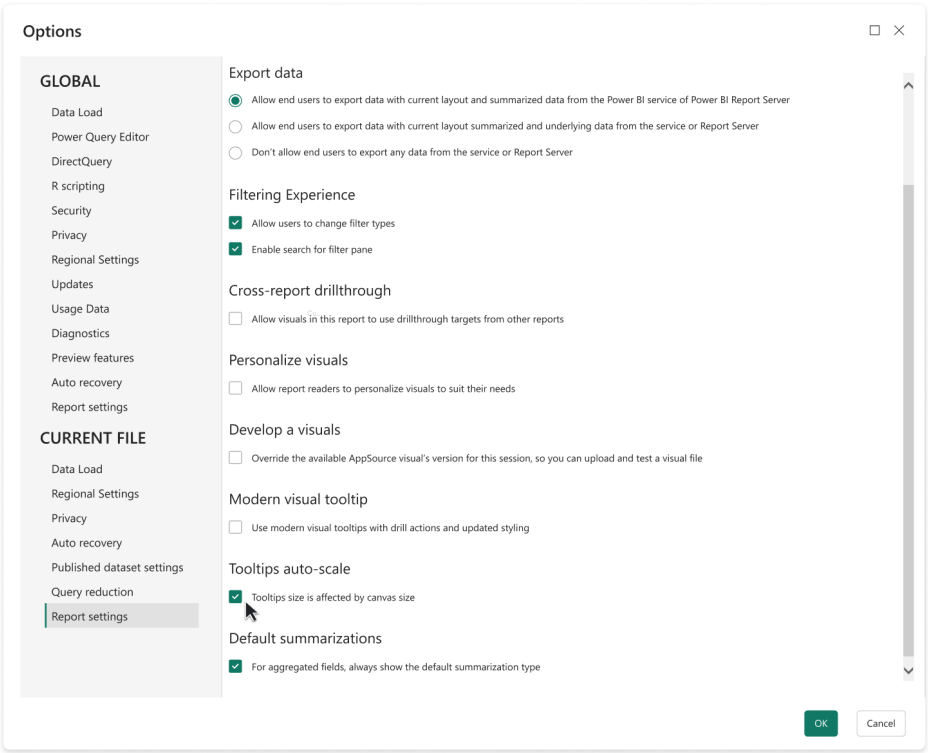
You can now drag a field to the closed build button to add a field too.

Various bug fixes were put through too:

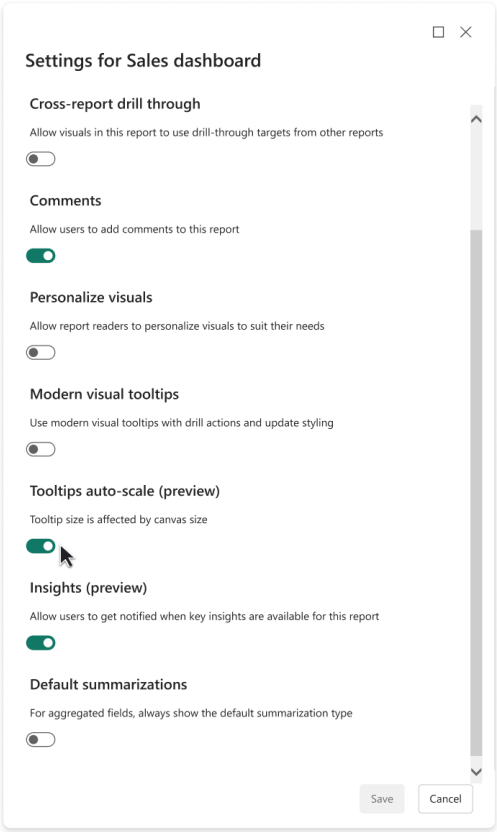
- Data pane's expansion state is now maintained whilst using the pane switcher
- if the Build menu is already open, resizing / moving the visual will reopen the menu
- Delete while in Format mode now deletes the whole visual instead of just the background
- hidden fields now shown in the Build menu's data flyout if 'view hidden' is turned on in the Data pane.

New ToolTip auto-scale in Preview

This release sees a user preference setting to allow ToolTips to auto-scale based upon the canvas' size under **Options -> Report settings -> Tooltip auto-scale**, *viz.*

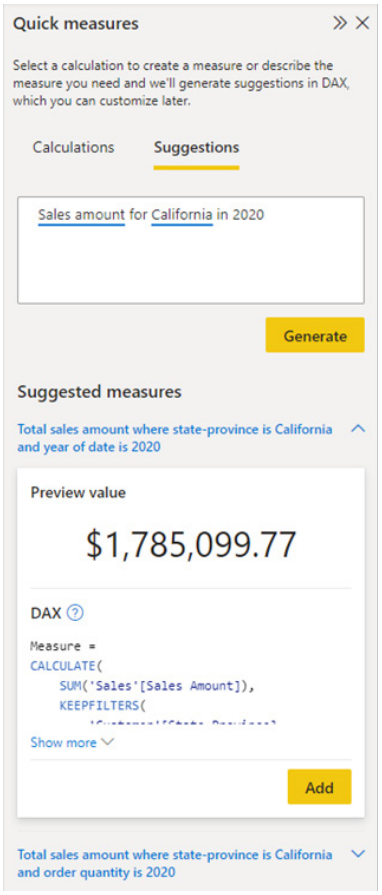


You can also update this setting in the Report settings within Power BI Service:

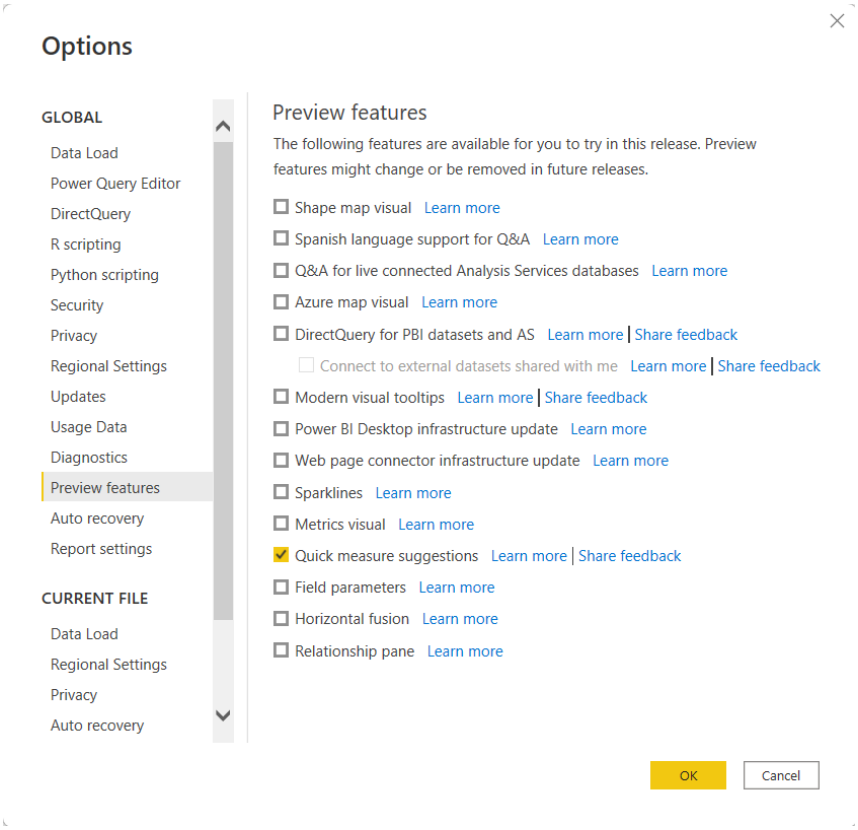


Update to Quick Measure Suggestions in Preview

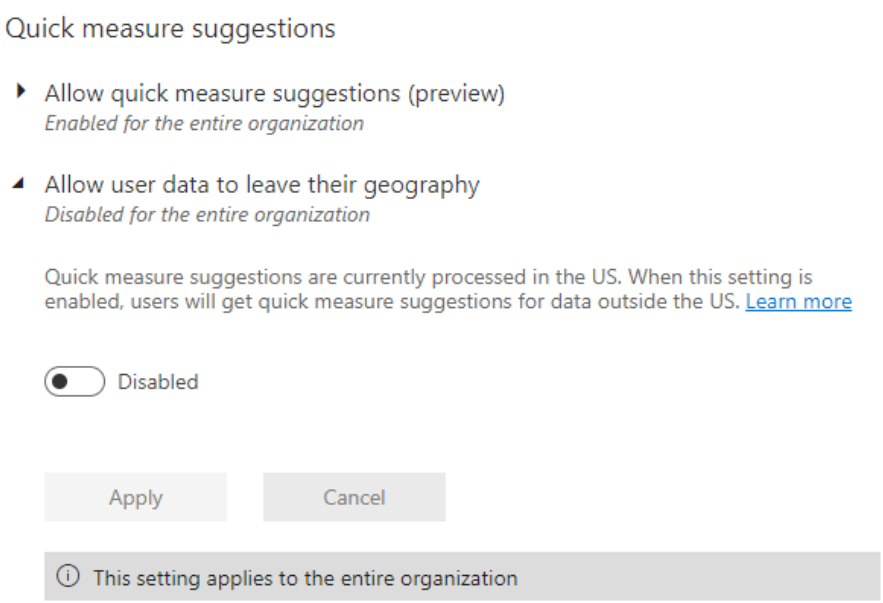
Back in October, the Power BI team released an opt-in, experimental preview of Quick Measure Suggestions, powered by Azure OpenAI, that helps users create DAX measures using natural language instead of writing DAX from scratch. With this feature, users may generate calculations and business logic for numerous different measure scenarios including time intelligence, mathematics, text operations, if conditions and much more.



With this month's release, Microsoft has removed the experimental label on the feature, and has now enabled the Preview feature by default, so you do not have to manually turn on the Preview feature in the options menu of Power BI Desktop.



The feature is powered by a machine learning model that is currently only deployed to US datacentres (East US and West US). If your data is outside the US, the feature will be disabled by default unless your tenant administrator enables 'Allow user data to leave their geography' tenant setting:



Composite models on Power BI Datasets and Analysis Services models now Generally Available

In December 2020, Power BI launched the Preview of DirectQuery for Power BI Datasets and Analysis Services. Since then, Microsoft has been working on improving this feature to get it ready for General Availability. There were various improvements made, such as:

- support for SQL Server Analysis Services Tabular models (version 2022 required)
- ensured display folders, sort by column properties and format strings persist
- introduced sub setting and the ability to connect to perspectives
- users may now view a report built using this feature with Read permissions / Viewer role instead of Build permission / Contributor role.

Models built using this feature are referred to as **composite models**. Originally known as "DirectQuery for Power BI Datasets and Analysis Services", this name is now superseded as it has served its purpose during the Preview period to be able to clearly identify that this was a Preview feature.

This update acknowledges the end of the Preview period: composite models based upon Power BI datasets and Analysis Services Tabular models are now Generally Available and fully supported on Premium, PPU and new Pro workspaces.

Microsoft will announce General Availability for existing Pro workspaces later as the Power BI team finish the backend changes required. It should be noted that as these back-end changes are happening, users might no longer need Build permissions to view a report built using this feature, but instead they can rely on just Read permissions. When this happens, you can safely take away the Build permissions for your consumers.

If you are currently using Pro workspaces and want to switch over to the Read permission model right away, you can. However, it requires you to create new workspaces and host the datasets and reports there. As a reminder, going forward you will need to enable the 'Allow XMLA endpoints and Analyze in Excel with on-premises datasets' setting enabled in your tenant to use this feature.

Furthermore, when the Preview began, any consumer of a report that leveraged a composite model based upon a Power BI dataset was required to have Build permissions or the Contributor role. Since then, Power BI has changed this for Premium and PPU workspaces: readers of reports based upon datasets in Premium or PPU workspaces just require Read permissions (or the Viewer role).

However, for most Pro workspaces Build permissions are still required in the scenario above. As mentioned above, Microsoft is still making backend changes to align the Pro workspaces with the Premium and PPU workspaces so everyone consuming these reports will just require Read permission or the Viewer role, regardless of the workspace type the data is stored in. Apparently, these changes are still underway and are taking longer than expected!

This does have an impact for us end users:

- if you only use Premium / PPU workspaces, Read permission is required to view a report
- if you use a mixture of Premium / PPU and Pro workspaces or use Pro workspace exclusively, some of your reports might currently only require Read permissions while others will still require Build. The ratio of these will shift over time to the vast majority just requiring Read as Microsoft completes the backend changes
- if you currently use Pro workspaces, some datasets will not benefit from the backend changes and will still require Build permissions, even after the back-end changes are completed. This includes datasets that:
    - use one or more unsupported sources *and / or*
    - consume a large amount of memory, which is true for about 0.03% of all datasets.

Finally (for this topic anyway!), Microsoft has stated it has seen numerous users making DirectQuery connections to unsupported sources, such as usage metrics and real-time data sets. Whilst during the Preview you might not have received an error when using these sources, you will start seeing an error going forward as "loose ends are tightened". Even if you do not see an error, you should still use this feature with supported sources. The unsupported sources are documented, so please make sure you use supported sources for your reports that rely on this feature now or trouble may be finding you out in the very near future…

Updates to the ORDERBY function

Back in December, the Power BI updates introduced new functions that make it easier to do comparison calculations. This month, these functions are being made more powerful by adding more control over the ordering of the input data: the **ORDERBY** function now accepts any DAX expression as the first parameter. Previously, it only accepted a column (field) name.

As a refresher, that update (first reported in our February newsletter) saw Microsoft introduce multiple new functions for DAX, targeted at making it easier to do comparison calculations in Power BI. The new functions were as follows:

- **INDEX** retrieves a result using absolute positioning
- **OFFSET** retrieves a result using relative positioning
- **WINDOW** retrieves a slice of results using absolute or relative positioning.

These functions also come with two helper functions called **ORDERBY** (the one that has been updated this month) and **PARTITIONBY**.

These functions will make it easier to perform calculations such as:

- comparing values against a baseline or finding another specific entry (using **INDEX**)
- comparing values against a previous value (using **OFFSET**)
- adding a running total, moving average or similar calculations that rely on selecting a range of values (using **WINDOW**).

If you are familiar with the SQL language, you will note that these functions are very similar to SQL window functions. The functions released in this update perform a calculation across a set of table rows that are in one way or another related to the current row. These functions are different from SQL window functions, because of the DAX evaluation context concept, which will determine what is the "current row". Moreover, the functions introduced will not return a value but rather a set of rows which can be used together with **CALCULATE** or an aggregation function like **SUMX** (see elsewhere in this month's newsletter) to calculate a value.

Further, it should be noted that this group of functions is not pushed to the data source, but rather they are executed in the DAX engine. Additionally, Microsoft has stated it has seen much better performance using these functions compared to existing DAX expression to achieve the same result, especially when the calculation requires sorting by non-continuous columns.

The DAX required to perform these calculations is simpler than the DAX required without them. However, while these new functions are very powerful and flexible, they still require a fair amount of complexity to make them work correctly. That is because Microsoft opted for high flexibility for these functions. However, the Powers that Be have stated that they recognise there is a need for easier to use functions that sacrifice some of the flexibility in favour of easier DAX. With this in mind, these functions now released should be seen as "a stepping stone, a building block if you will" towards Microsoft's goal to make DAX easier.

To assist what has happened to **ORDERBY**, let's revisit these functions in total.

INDEX

INDEX allows you to perform comparison calculations by retrieving a row that is in an absolute position. This will be most useful for comparing values against a certain baseline or another specific entry.

Consider the following example. Below is a table of customer names and birth dates whose last name is "Garcia":

FullName	BirthDate
Abby Garcia	12/19/1991
Abigail Garcia	7/21/1983
Adriana Garcia	12/4/1957
Alexander Garcia	11/18/1987
Alexandra Garcia	1/20/1997
Allen Garcia	5/7/1979
Alyssa Garcia	1/23/1976
Andre Garcia	8/6/1996
Andrew Garcia	4/2/1996
Anna Garcia	9/16/1979
Anthony Garcia	5/3/1993
Arthur Garcia	3/6/2000
Ashley Garcia	1/3/1984
Austin Garcia	4/20/1979
Barry Garcia	2/17/1972
Benjamin Garcia	10/1/1999
Blake Garcia	3/8/1984

Now, imagine you wanted to find the oldest customer for each last name. Therefore, for the last name "Garcia" that would be Adriana Garcia, born December 4th, 1957. You can add the following calculated column on the **DimCustomer** table to achieve this goal and return the name:

Oldest Customer of LastName = SELECTCOLUMNS(INDEX(1, DimCustomer, ORDERBY([BirthDate]), PARTITIONBY([LastName])), [FullName])

This would return the following result:

FullName	BirthDate	Oldest Customer of LastName
Abby Garcia	12/19/1991	Adriana Garcia
Abigail Garcia	7/21/1983	Adriana Garcia
Adriana Garcia	12/4/1957	Adriana Garcia
Alexander Garcia	11/18/1987	Adriana Garcia
Alexandra Garcia	1/20/1997	Adriana Garcia
Allen Garcia	5/7/1979	Adriana Garcia
Alyssa Garcia	1/23/1976	Adriana Garcia
Andre Garcia	8/6/1996	Adriana Garcia
Andrew Garcia	4/2/1996	Adriana Garcia
Anna Garcia	9/16/1979	Adriana Garcia
Anthony Garcia	5/3/1993	Adriana Garcia
Arthur Garcia	3/6/2000	Adriana Garcia
Ashley Garcia	1/3/1984	Adriana Garcia
Austin Garcia	4/20/1979	Adriana Garcia
Barry Garcia	2/17/1972	Adriana Garcia
Benjamin Garcia	10/1/1999	Adriana Garcia
Blake Garcia	3/8/1984	Adriana Garcia

In the example above, we showed only customers whose last name is “Garcia”. However, the same calculated column works on a set that has more than one last name:

FullName	BirthDate	Oldest Customer of LastName
Abby Kovár	10/20/1973	Abby Kovár
Andre Kovár	12/12/1991	Abby Kovár
Barry Kovár	12/30/1991	Abby Kovár
Donald Kovár	11/2/1973	Abby Kovár
Geoffrey Kovár	10/10/1981	Abby Kovár
Rebekah Kovár	11/10/1984	Abby Kovár
Whitney Kovár	12/30/1983	Abby Kovár
Abhijit Thakur	1/30/1976	Abhijit Thakur
Aaron Wang	3/4/1985	Adam Wang
Adam Wang	10/31/1958	Adam Wang
Aimee Wang	5/25/1982	Adam Wang
Alan Wang	11/4/1995	Adam Wang
Alejandro Wang	1/25/1983	Adam Wang

As you can see in the screenshots above, the full name of the oldest person with that last name is returned. That’s because we instructed **INDEX** to retrieve the first result when ordering by birth date, by specifying one [1]. By default, the ordering for the columns passed into **OrderBy** is ascending. If we had specified two [2], we would have retrieved the name of the second oldest person with the last name instead, and so on.

Had we specified -1 or changed the sort order we would have returned the youngest person instead:

Youngest Customer of LastName = SELECTCOLUMNS(INDEX(1, DimCustomer, ORDERBY([BirthDate], DESC), PARTITIONBY([LastName])), [FullName])

is equivalent to:

Youngest Customer of LastName = SELECTCOLUMNS(INDEX(-1, DimCustomer, ORDERBY([BirthDate]), PARTITIONBY([LastName])), [FullName])

Notice that **INDEX** relies on two other new helper functions called **ORDERBY** and **PARTITIONBY**.

ORDERBY and PARTITIONBY

These helper functions may only be used in functions that accept an **orderBy** or **partitionBy** parameter, which are the functions introduced above:

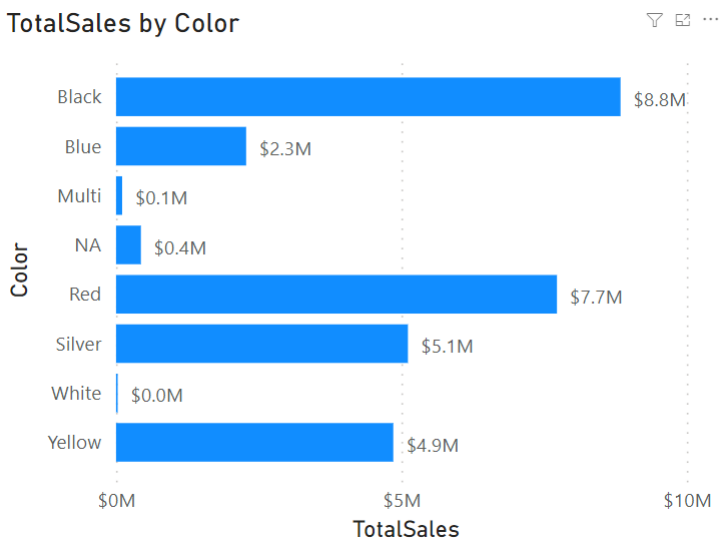
- the **PARTITIONBY** function defines the columns that will be used to partition the rows on which these functions operate
- the **ORDERBY** function defines the columns that determine the sort order within each of a window function’s partitions specified by **PARTITIONBY**.

OFFSET

OFFSET allows you to perform comparison calculations more readily by retrieving a row that is in a relative position from your current position. This will be most useful for comparing across something else than time, for example across regions, cities or products. For date comparisons, for example, comparing the sales for this quarter against the same quarter last year there are dedicated Time Intelligence functions in DAX already.

That doesn’t mean you cannot use **OFFSET** to do the same, but it is not the immediate scenario.

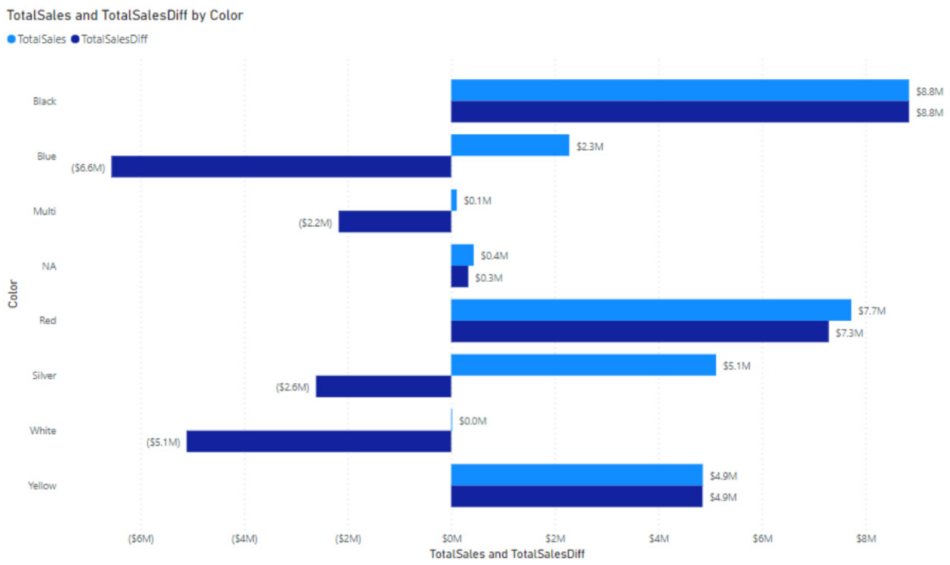
So what is the scenario for **OFFSET**? Consider the following. Here’s a Bar chart that shows total sales by product colour:



Now, let's say you wanted to compare how well each colour is doing against the colour above it in the chart. You could write a complicated DAX statement for that, or you can now use **OFFSET** to accomplish this goal more simply, *viz.*

TotalSalesDiff = IF(NOT ISBLANK([TotalSales]), [TotalSales] - CALCULATE([TotalSales], OFFSET(-1, FILTER(ALLSELECTED(DimProduct[Color]), NOT ISBLANK([TotalSales])))))

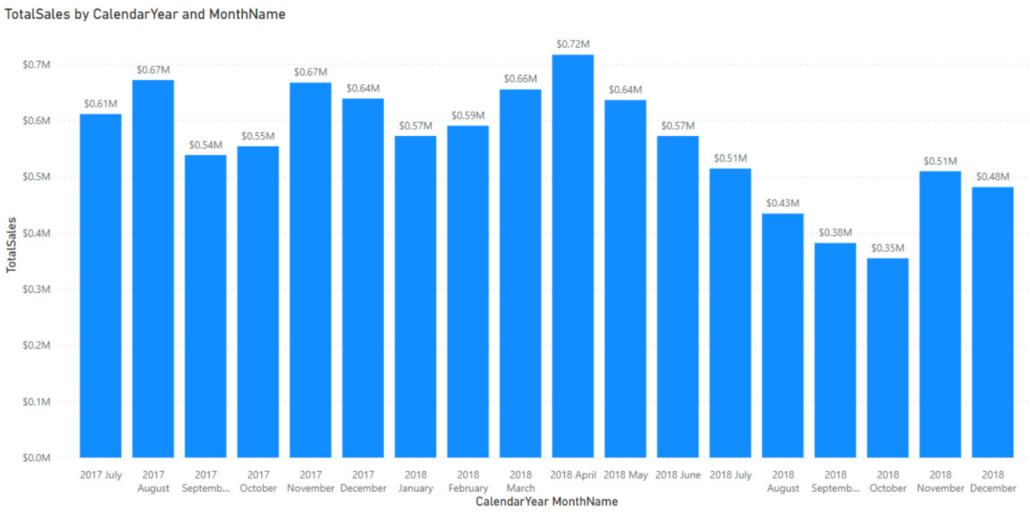
This will return the following result:



As you can see the newly added bars calculate the difference for each colour compared to the one just above it in the chart. That's because the DAX formula specified -1 for the first parameter to **OFFSET**. If we had specified -2 we would have made the comparison against the colour above each colour, but skipping the one right above it, so effectively the sales for the grey colour would have been compared against the sales for products that were black, *etc.*

WINDOW

WINDOW allows you to perform calculations that rely on ranges of results ("windows"), such as a moving average or a running sum. For example, the below Column chart shows total sales by year and month:

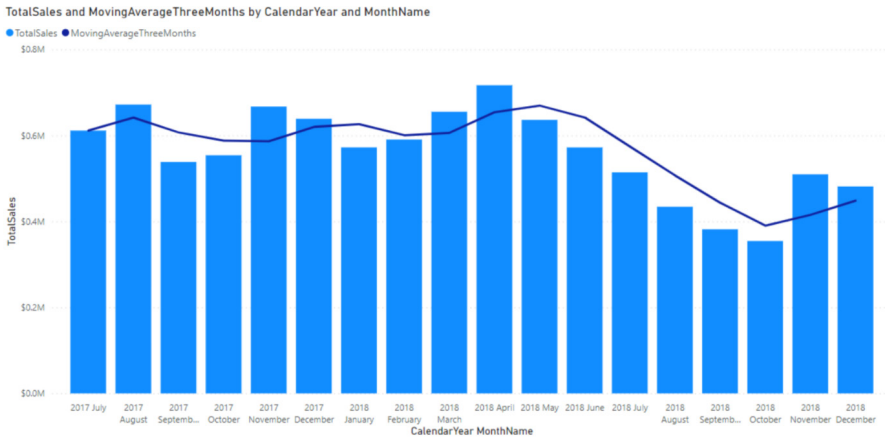


Now, let's say you wanted to add a moving average for the last three months of sales including the current. For example, for September 2017, you would expect the result to be the average sales of July, August and September in 2017 and for February 2018, we would expect the result to be the average sales for December 2017, January 2018 and February 2018.

To meet this requirement, you could write a complicated DAX statement or you can now use **WINDOW** to accomplish this goal using a simpler DAX statement:

MovingAverageThreeMonths = AVERAGEX(WINDOW(-2, 0, ALLSELECTED(DimDate[CalendarYear], DimDate[MonthName], DimDate[MonthNumberOfYear]), ORDERBY(DimDate[CalendarYear], ASC, DimDate[MonthNumberOfYear], ASC)), [TotalSales])

This will return the following result:



The newly added line correctly calculates the average sales over three months (including the current month). This release on a so-called 'relative window': the first parameter to **WINDOW** is set to -2, which means that the start of the range is set two months before to the current month (if that exists). The end of the range is inclusive and set to zero [0], which means the current month. Absolute windows are available as well, as both the start and end of the range can be defined in relative or absolute terms. Notice that **WINDOW** relies on two other new functions called **ORDERBY** and **PARTITIONBY** too.

So let's return to this month's update. Continuing from the examples above, let's figure out which customer has bought the most and return their full name using the following expression:

BiggestSpender = INDEX(1, ALLSELECTED ('DimCustomer'[FullName]), ORDERBY(CALCULATE(SUM('FactInternetSales'[SalesAmount])), DESC))

Notice how the first parameter of the **ORDERBY** function is an expression returning the sum of **SalesAmount**. This is not something that was possible before. We could have performed the same using a measure defined as:

[Total Sales] = SUM('FactInternetSales'[SalesAmount])

The **BiggestSpender** definition then changes slightly:

BiggestSpender = INDEX(1, ALLSELECTED ('DimCustomer'[FullName]),
ORDERBY([Total Sales], DESC))

New DAX functions: RANK and ROWNUMBER

This month sees two more functions added that should assist when calculating rankings: **RANK** and **ROWNUMBER** are joining the DAX ranks.

These functions return a number indicating the rank for the current context within the specified partition, sorted by the specified order. The difference between **RANK** and **ROWNUMBER** is that if there is a tie (*i.e.* two rows would get the same rank assigned) **ROWNUMBER** will return an error, whereas **RANK** will just assign the same **RANK** multiple times. You should note that returning an error is a last resort; **ROWNUMBER** will try to avoid doing that by finding the least number of additional columns required to uniquely identify every row and append these new columns to the **ORDERBY** clause. Only after it is unable to uniquely identify every row, **ROWNUMBER** will return an error.

These functions rely on the **ORDERBY** and **PARTITIONBY** functions.

In the following example, we have a list of customers and their birth dates. I have added the following measures to my model:

RankByBirthDateSkip = RANK(SKIP, ALLSELECTED(DimCustomer), ORDERBY(DimCustomer[BirthDate]))

RankByBirthDateDense = RANK(DENSE, ALLSELECTED(DimCustomer), ORDERBY(DimCustomer[BirthDate]))

RowNumberByBirthDate = ROWNUMBER(ALLSELECTED(DimCustomer), ORDERBY(DimCustomer[BirthDate]))

This is the first part of the output:

FullName	BirthDate	RankByBirthDateSkip	RankByBirthDateDense	RowNumberByBirthDate
Adriana Garcia	Wednesday, December 04, 1957	1	1	1
Christopher Garcia	Friday, May 23, 1958	2	2	2
Zachary Garcia	Saturday, August 02, 1958	3	3	3
Harold Garcia	Sunday, November 02, 1958	4	4	4
Sydney Garcia	Saturday, September 12, 1959	5	5	5
James Garcia	Monday, January 23, 1961	6	6	6
Wyatt Garcia	Sunday, October 22, 1961	7	7	7
Francisco Garcia	Saturday, December 02, 1961	8	8	8
Whitney Garcia	Sunday, March 11, 1962	9	9	9
Dalton Garcia	Friday, August 23, 1963	10	10	10
Ronald Garcia	Thursday, October 01, 1964	11	11	11

All measures here return the same result. However, for customers that share a birthday, the results are different:

FullName	BirthDate	RankByBirthDateSkip	RankByBirthDateDense	RowNumberByBirthDate
Morgan Garcia	Friday, April 23, 1976	38	38	38
Rebekah Garcia	Sunday, August 22, 1976	39	39	39
Hunter Garcia	Friday, October 22, 1976	40	40	40
Donald Garcia	Tuesday, December 14, 1976	41	41	41
Kayla Garcia	Tuesday, December 14, 1976	41	41	42
Brandon Garcia	Friday, December 17, 1976	43	42	43
Natalie Garcia	Monday, April 25, 1977	44	43	44
Jessica Garcia	Monday, May 09, 1977	45	44	45

Notice how both Donald Garcia and Kayla Garcia are both on the same date. Using **RANK** with the ties parameter set to **SKIP** (the default) assigns them a rank of 41. The same happens when using **RANK** with the ties parameter set to **DENSE**. However, notice that the next customer receives a different rank (43 when the ties parameter is set to **SKIP** and 42 when set to **DENSE**). By contrast, **ROWNUMBER** gives Donald and Kayla an unique rank (41 and 42) as it expands the **ORDERBY** clause to try to unique identify these customers and is successful in doing so.

Oracle database (connector update)

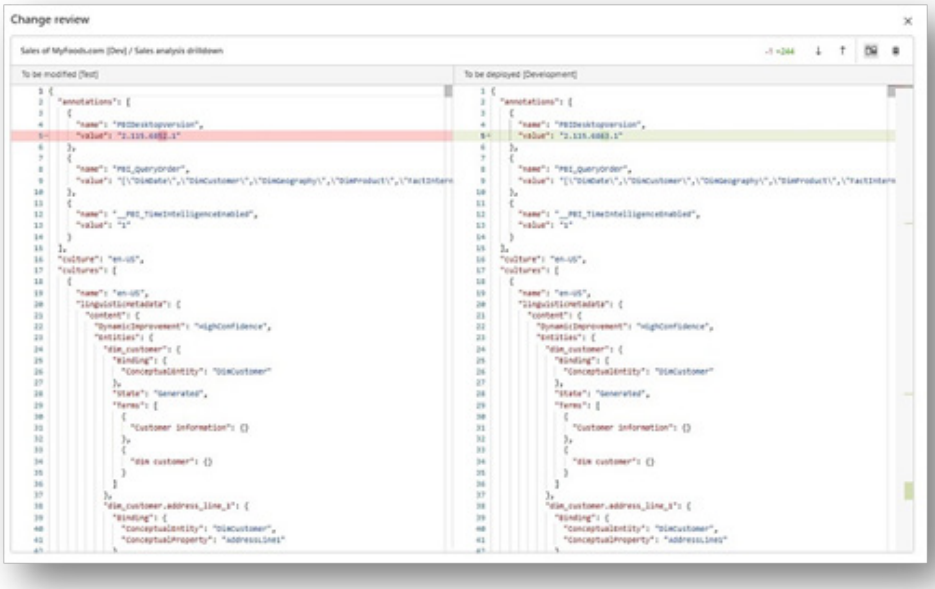
The Oracle database connector in Power BI Desktop has been updated to support Azure Active Directory-based authentication.

New features to Deployment Pipeline

Two new features this month.

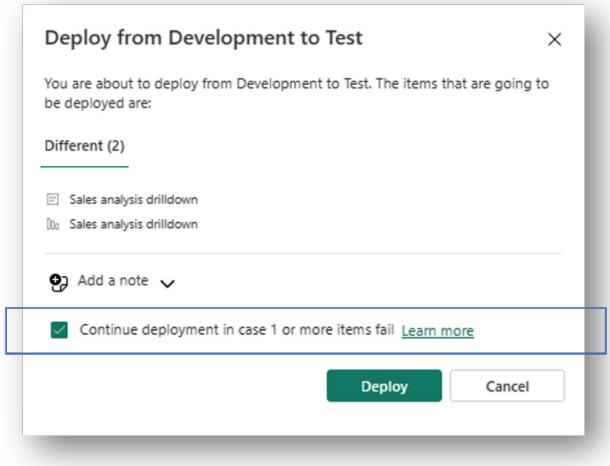
VIEW SCHEMA CHANGES LINE-BY-LINE

Using a comparison tool as part of your deployment pipeline can help to ensure you deploy changes with confidence and reduce the risk of errors and inconsistencies. For that purpose, Microsoft extended their Compare tool *(sic)* by adding the option to review an item's changed lines, as they are highlighted on its schema, presented in a dedicated pop-up window designed by industry standards for code compare.



CHOOSE WHETHER TO CONTINUE THE DEPLOYMENT IN CASE OF A FAILURE

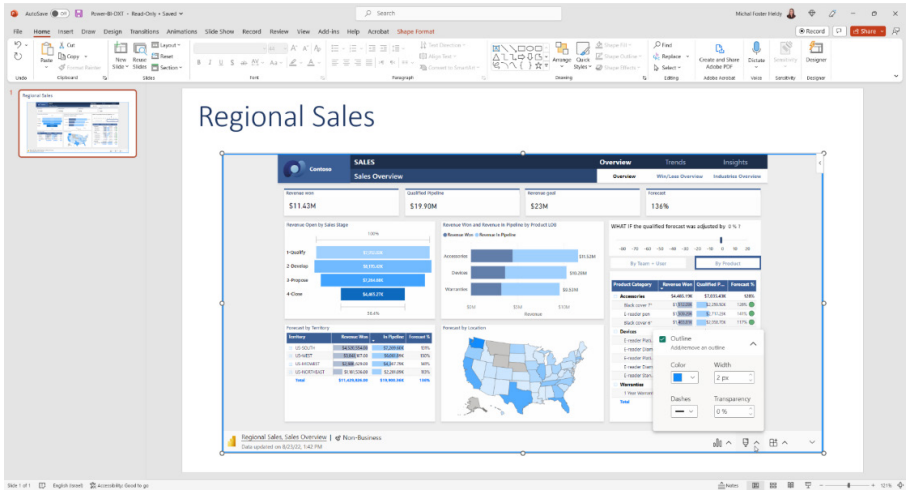
Also, as part of the ongoing efforts to enhance the deployment experience, Power BI has added the option of continuing the deployment when one item's deployment fails. Until now, by default, a deployment stopped when failed to deploy an item, so the subsequent items were not deployed too. But now, there's an option to select continuing the deployment in such case, so the failed item's downstream items will be skipped, although other subsequent items will be still deployed.



Storytelling in PowerPoint: new Style option

The Power BI Storytelling add-in for PowerPoint enables you to add Power BI reports and visuals to your PowerPoint slides, and enjoy the dynamic, interactive experience of live Power BI data inside your presentations.

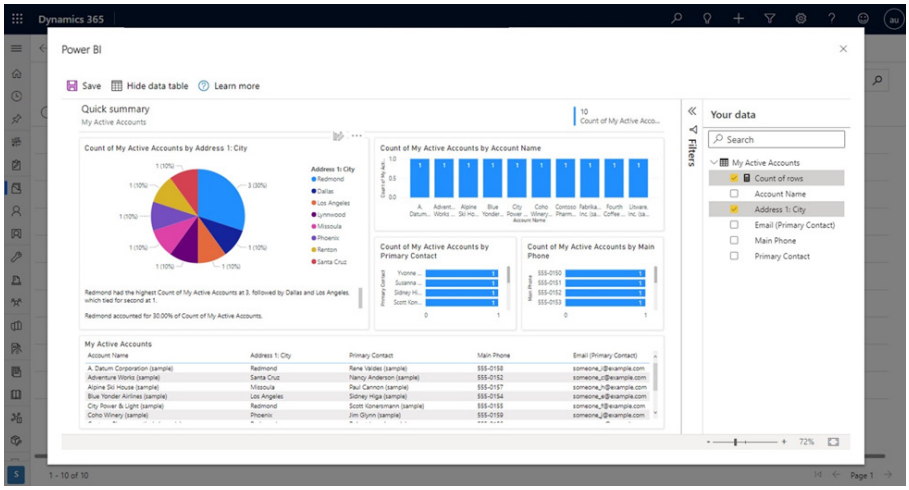
To create more attractive data-powered presentations, you now have a new styling menu item. In this update, Microsoft is introducing the first style option: outlines. You may use the outline to add a border to your add-in, making your Power BI data stand out. You can just check the Outline checkbox to get the default style or you can expand the option to customise the border to fit your slide and data.



Visualising views in Power Apps with Power BI Quick Report is now enabled by default

Power BI Quick Reports in Power Apps, which has been Generally Available as an opt-in capability to help you discover insights from your data from within business applications, is now enabled by default with 2023 Release Wave 1.

Power BI Quick Reports in Power Apps represent a seamless integration of Power Apps, Power BI and Dataverse into a single experience to enable every business user turn data into insights with just one click. Everyone can create visually appealing, meaningful, interactive reports based upon a view in a model-driven application and save the report into the Power BI Service easily.



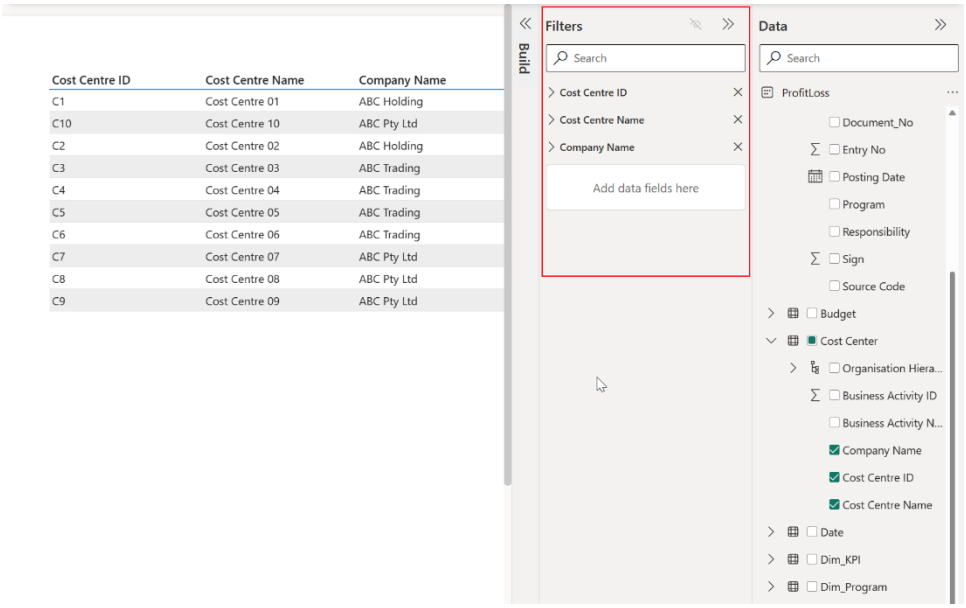
Paginated Reports feature summary

In the last update, Power BI introduced Parameter pane improvements, namely the datepicker as well as the ability to create a paginated report from a datamart. This month sees additional capabilities introduced when using paginated report web authoring.

FILTERS

You can now add filters when you create paginated reports in the Power BI Service. The software has taken the next step in our progression for web authoring to enable filtering. When you author a report in the Service, you can filter the data using the Filters pane on the canvas.

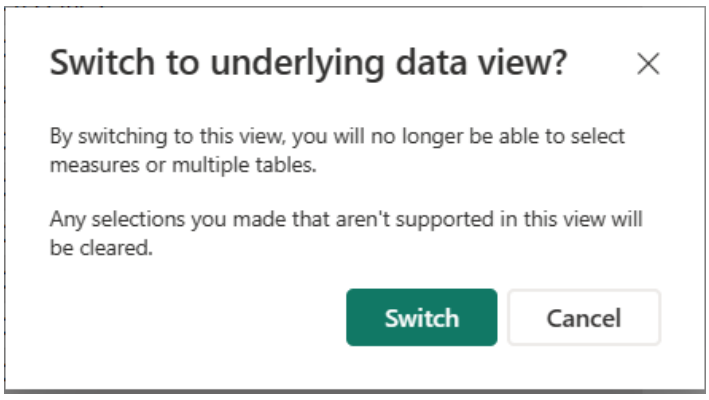
You can filter data at the Report level, which means that the filter applies to all the pages of the report.



There are two ways to add a column to the Filters pane. You can select columns from the Data pane or you may drag and drop inside the Filters pane to 'add data fields here'.

DATA PREVIEW

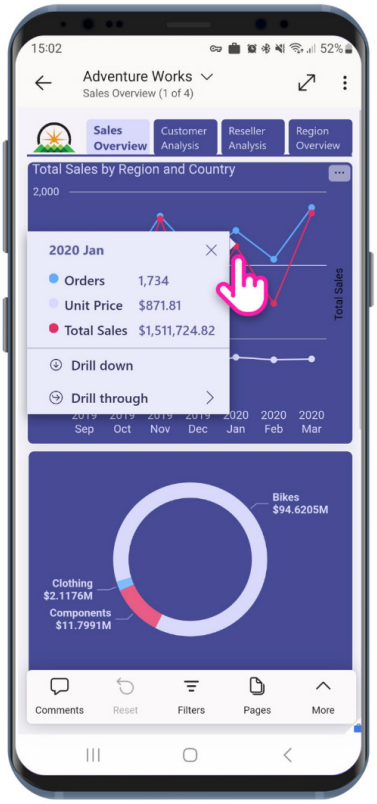
The Data preview experience has also been updated. Data preview enables you to view the underlying data of a selected table or columns from the dataset. You may also export the data to supported file formats or create a paginated report. If you want summarised data, you can select 'Create paginated report'. You may also switch from summarised to underlying data by choosing more options (…) in the 'Fields data' pane.



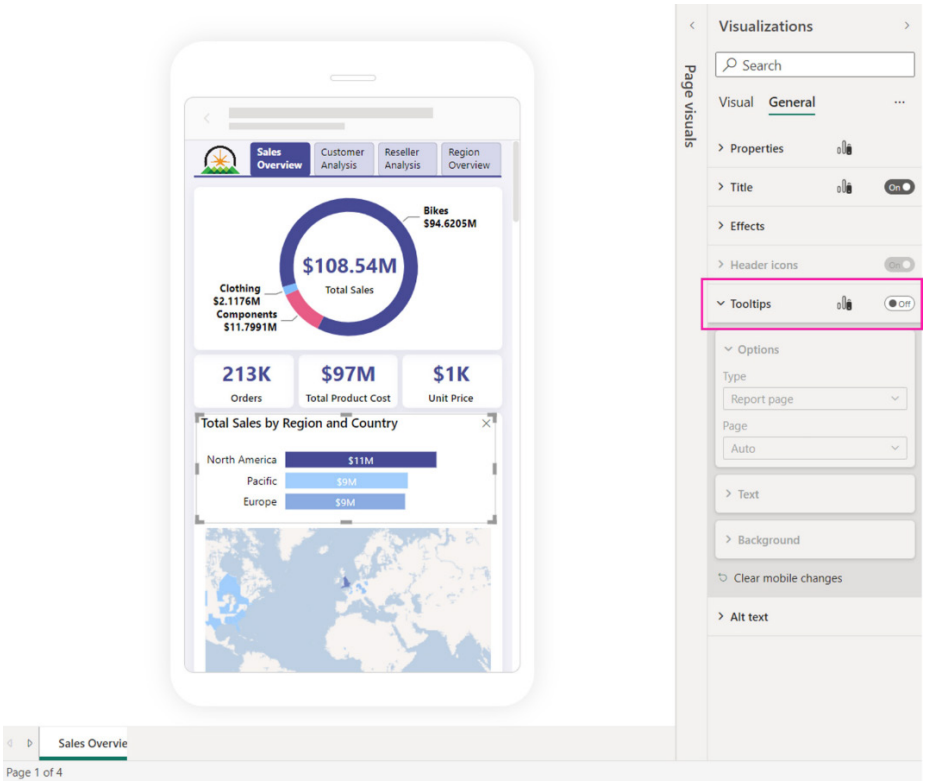
Enhanced ToolTips on visuals in the Power BI Mobile app

In this release, there's a new look and feel for ToolTips on visuals. By tapping and holding on a data point, you'll now enjoy a sleek, user-friendly ToolTip that includes:

- a pointer to help you identify the selected data point
- clear, easy-to-use actions such as drill-down and drill-through
- custom styling as configured by the report creator.

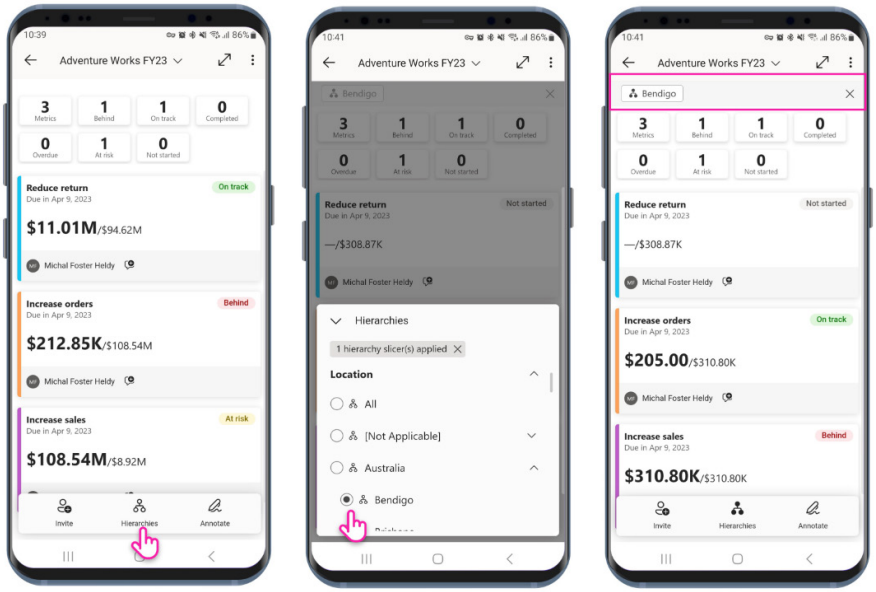


Additionally, report creators can now choose to disable a ToolTip on mobile devices by going to the 'Mobile layout visual formatting" pane in Power BI Desktop, selecting the ToolTip setting and disabling it.

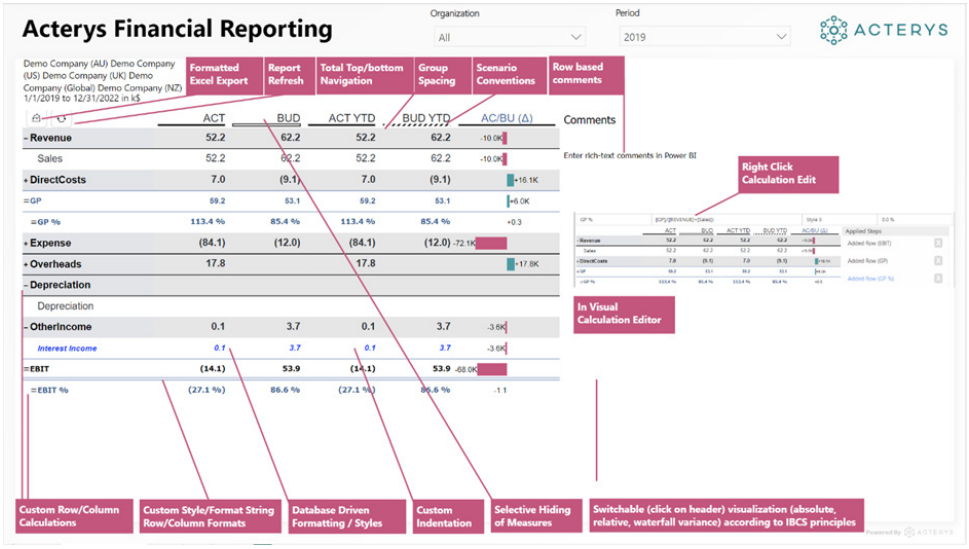


Hierarchies are now supported in the Power BI Mobile app

Hierarchies for your metrics and scorecards make it easier to drill in and check the status at different levels of your data. Hierarchies are now available in the Mobile app, where you can drill into the hierarchy to check progress and statuses, as well as do check-ins, at different levels. To view the available hierarchies and change the selection, use the Hierarchies button in the Scorecard footer.



Acterys Reporting Suite 3.0: reporting



This visual allows you to generate financial reports and visualisations according to IBCS (International Business Communication Standards) principles with extensive formatting, custom subtotals as well as combined table, chart and comment displays.

You may add your own row or column-based calculations and formatting as needed with spreadsheet ease directly in the visual avoiding the need for complex DAX. The revolutionary 'Data-Driven Styles' feature automatically applies formatting and updates of all Acterys Reporting visuals used in any report in your Power BI tenant according to the latest definition in a central style table.

Features include:

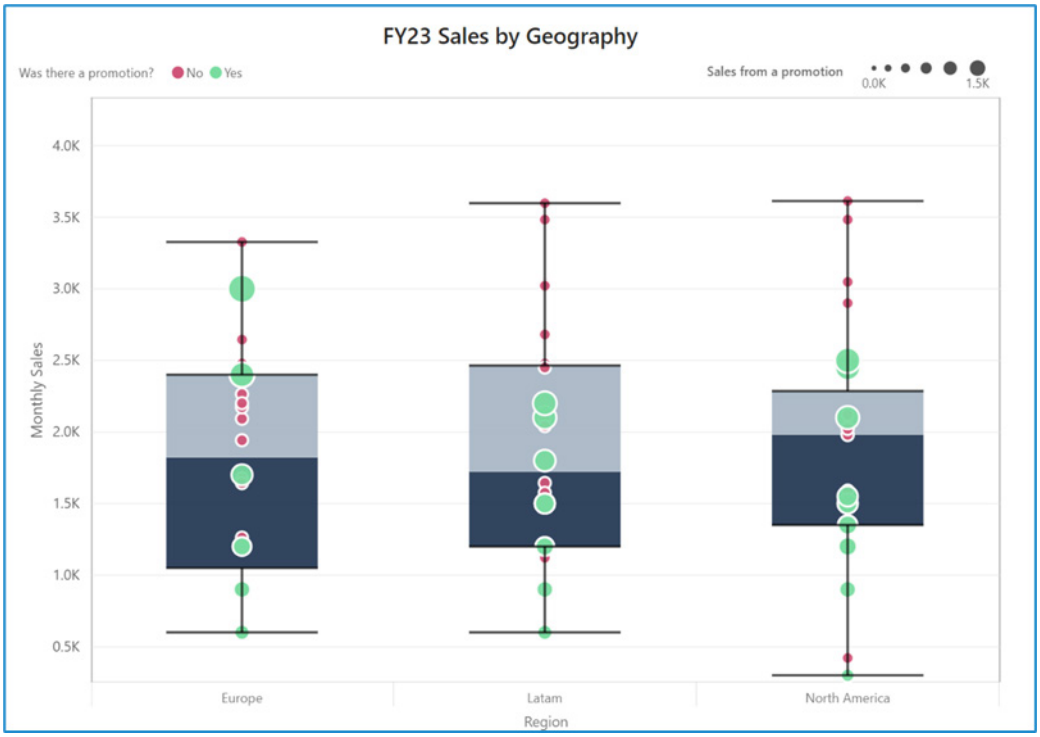
- addition of custom rows and column calculations and subtotals exactly where needed *e.g.* Gross Margin, EBIT
- financial reporting format options per row (under / over lines, (), %, scaling, *etc.*)
- data driven Styles: automatically apply and update all report formatting options (size, colour, under / over lines, *etc.*) according to the latest definition in a central style table
- switchable variance visualisation according to IBCS principles absolute / relative variance and waterfall option with option to specify comparison columns
- rich text commenting on a row or cell-level basis
- integration with all Acterys write-back, planning and business modelling features.

Benefits include:

- all aspects of financial reports produced quickly in Power BI
- instant insight: financial reports formatted according to IBCS principles
- avoids the need for complex DAX so that you should be able to run your financial reports faster.

Box and Whisker Chart by MAQ Software

This visualisation reveals patterns and outliers in your data. The Box and Whisker Chart by MAQ Software is a useful tool for analysing data distribution, identifying trends and pinpointing outliers. It displays median, quartiles, maximum, minimum, mean, standard deviation and quartile deviation, providing a deeper understanding of data, designed to help users make informed decisions.



Key business uses:

- **Sales:** compare quarter-over-quarter sales across geographic regions or market demographics
- **Project managers:** analyse work output after major process changes
- **Product developers:** compare different processes to develop the same product at an improved speed, accuracy or overall functionality.

Key features:

- clear display of broad categories (*e.g.* financial quarter or geographic region) and the specific data sets within those categories (*e.g.* individual storefronts or products)
- horizontal and vertical view based on reporting requirements
- different shapes (circle, square, rectangle) for the mean icon and other data points.

Feature summary for Drill Down Graph PRO

Drill Down Graph PRO lets you create user-friendly graphs to represent complex relationships between nodes. It's useful for both small and large network graphs and offers advanced features like cross-chart filtering and various customisation options. You can create hierarchies and explore them using this visual's intuitive interactions.

Main features include:

- **multiple layout options:** dynamic, hierarchical and radial
- **focus nodes mode:** for gradual exploration of graphs
- **customisation options:** choose colours, shapes, images and labels
- **bidirectional links:** show reciprocal relationships between nodes
- **touch device support:** explore your data on various devices.

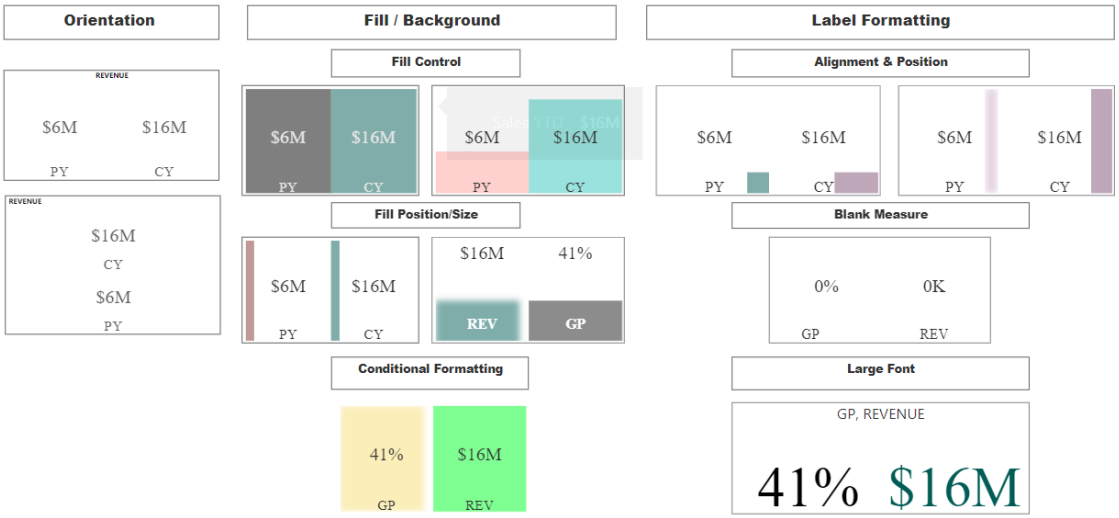
Popular use cases:

- **IT:** asset management, IT infrastructure, IoT monitoring
- **Logistics:** fleet management, stock management, parcel tracking
- **Sales & Marketing:** community detection, account management, web analytics.

DualCard visual

The DualCard visual by Inservit is useful for comparing two measures with advanced formatting options. Some key features of this visual are:

- compare two measures, the first called data and the second called comparison measure
- formatting options for data and comparison measure
- display blank measure as a static value
- support for large font sizes
- conditional formatting for fill, background, labels and categories
- card orientation and fill options.



That's it for this month: more anon.

New Features for Excel

This month's updates see the announcement of several new features across web, Windows and Mac. For web users, users may now speed up large workbooks with 'Check Performance', insert and edit formulae with 'Formula argument assistance' and enhance query organisation with drag and drop in the Queries pane. Last month's 'Block untrusted XLL add-ins' has now launched to all Windows users, whilst 'Assign a task with @mentions' is now available for Windows and Mac users.

You can check out the whole list here:

Excel for the web

- Check performance
- Formula argument assistance
- Drag and drop in Queries pane

Excel for Windows

- Block untrusted XLL add-ins
- Assign a task with @mentions

Excel for Mac

- Assign a task with @mentions.

Let's plough through.

Check Performance

Originally added to Excel for the web back in October last year, 'Check Performance' has been improved for Excel for the web, although precisely what these improvements are we aren't clear. Anyone who uses Excel regularly will be familiar with the dreaded file bloating phenomenon, where file sizes suddenly increase drastically and / or unexpectedly. At long last, the Excel Performance team has created a new capability to detect and remove unwanted size bloat and speed up such workbooks.

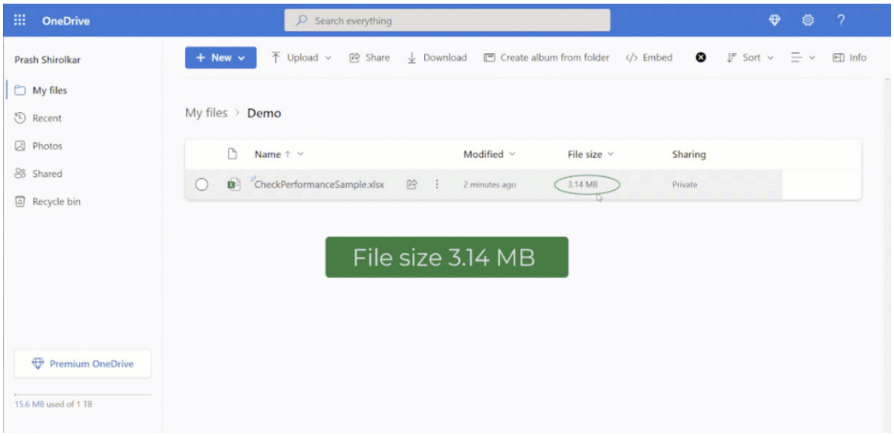
Often, a workbook can collect cells that have no data but still contain hidden information of little or no use anymore. These cells may have had data and formatting to start with, however now they do not have any data, but still take up space because they contain formatting. Consequently, too many of these cells can cause your workbook to slow down or become unresponsive. As a consequence, this update enables you to detect and remove these cells slowing down your workbooks, with 'Check Performance' – albeit presently available only in Excel for the Web.

When you open your workbook, Excel now detects whether your workbook contains too many of these unwanted formatted cells. If it does, Excel shows a "Business bar" to launch the 'Check Performance' feature. This may be launched manually from **Review -> Check Performance** too.

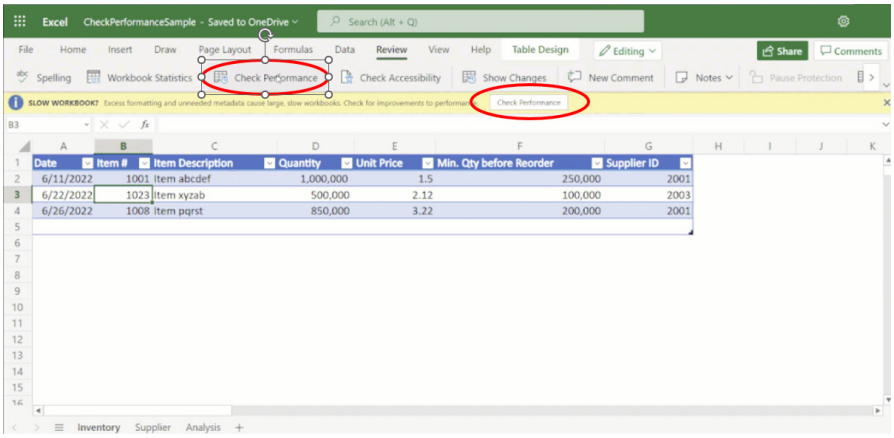
Once launched, there are two ways to remove these cells:

1. by navigating to a sheet in the task pane to review each range of these cells to optimise, and then pressing the 'Optimize Sheet' button; *or*
2. by pressing the 'Optimize All' button to remove all unwanted cells from all sheets in the workbook.

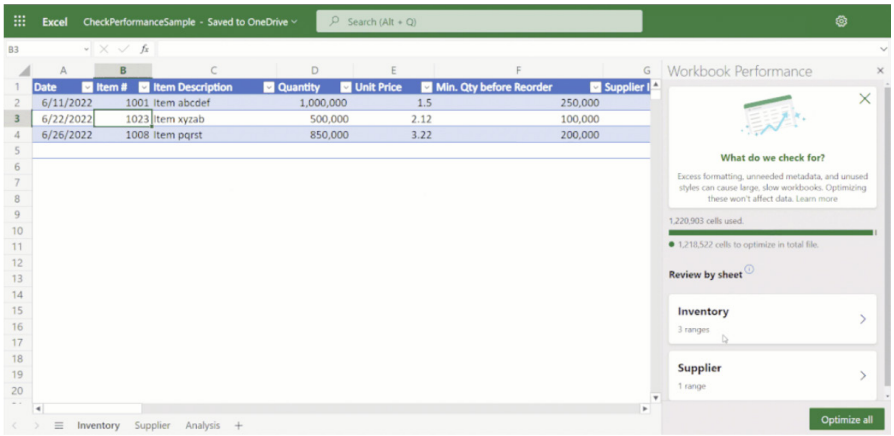
Consider the following example. Here, we have a file that has a current size of 3.14MB – not too large, but as it turns out, much larger than it *should* be.



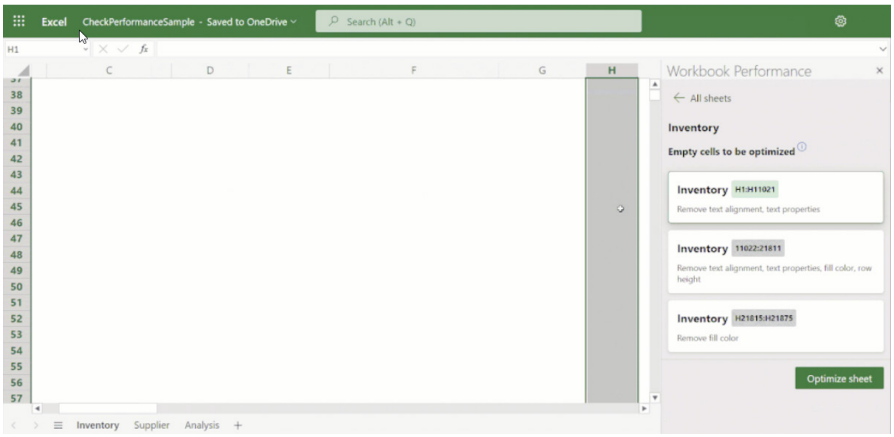
Upon opening, the Business bar (yellow bar) highlights the slow workbook and prompts you to 'Check Performance'. This may also be accessed from the Review tab, as displayed below:



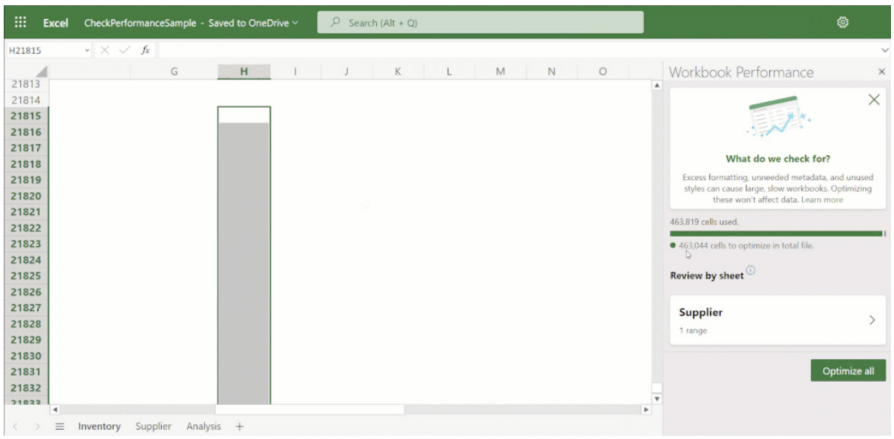
The 'Workbook Performance' pane appears and highlights two of the three worksheets may require review, citing a total of four ranges, *viz.*



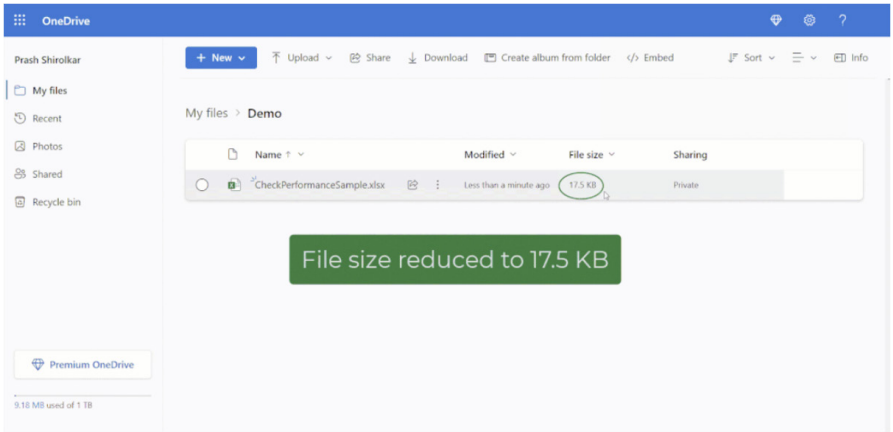
Clicking on the 'Inventory' box in the 'Workbook Performance' pane yields information on three empty cell ranges that may be optimised and prompts with an 'Optimize sheet' button (bottom right):



Similarly, the 'Supplier' sheet details one range to consider:



Following the prompts and optimising both sheets generates considerable savings in file size:



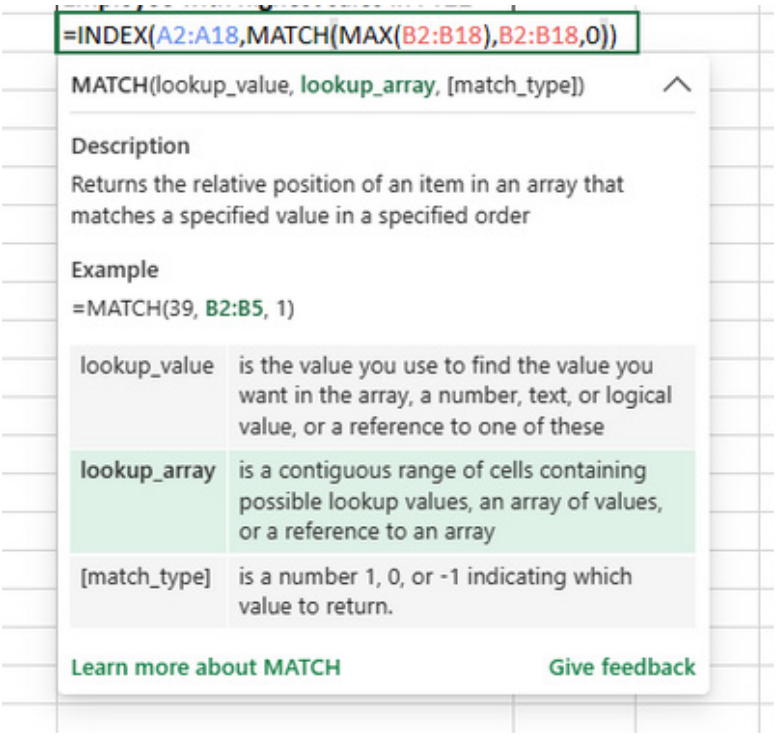
Here, 'Check Performance' has reduced a 3.14 MB file down to 17.5 KB, by detecting and removing more than a million unwanted formatted cells. Bearing this in mind, you might think, why doesn't Excel remove these cells in the background without alerting me? This is because, even though the cell has no data, removing its formatting may result in visible changes. For example, removing yellow fill from a cell may reset its fill to 'No Color', which might not be what you want. Microsoft does not want any Excel users to be surprised by visual changes by doing this in the background without alerting hence the manual interaction.

You should note that this new feature will be enabled gradually to more and more users over time as Microsoft rolls out the update and ensures it is working correctly. Therefore, if you do not see the 'Check Performance' button in the Review menu tab, then the feature may not be enabled for you yet.

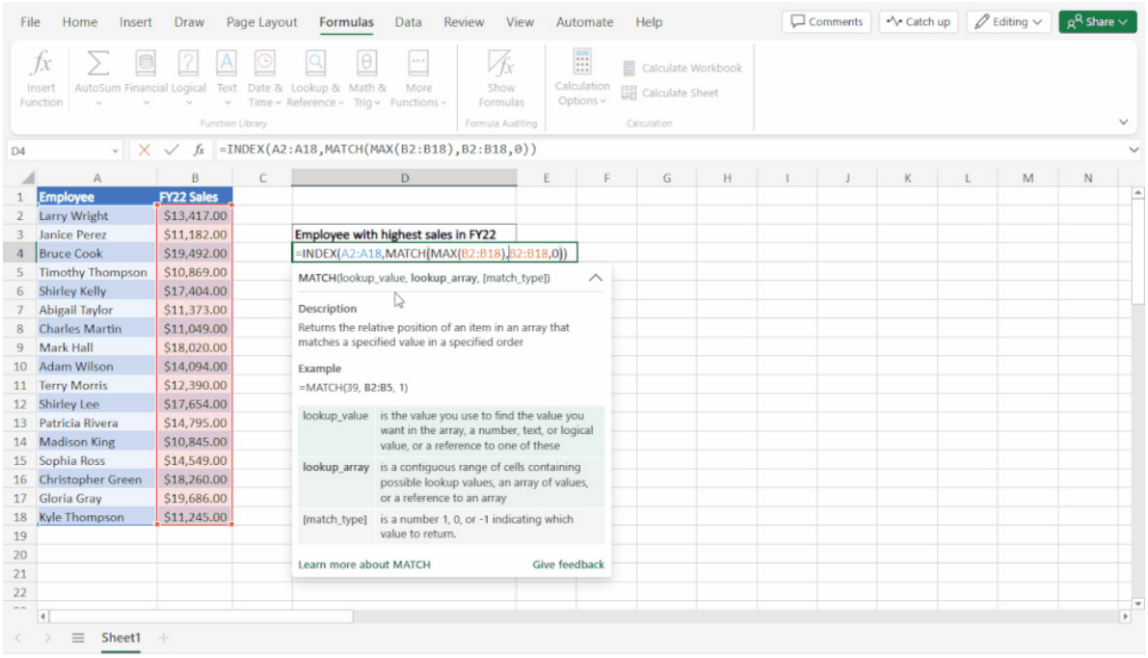
Formula argument assistance

Also for Excel for the web (only, at this stage), the 'Formula argument assistance' card accompanies you whilst writing a formula, and helps you insert or edit the arguments. You no longer need to reach out to external sources for help when you are typing your formula, as this card will help you write formulae efficiently and effectively.

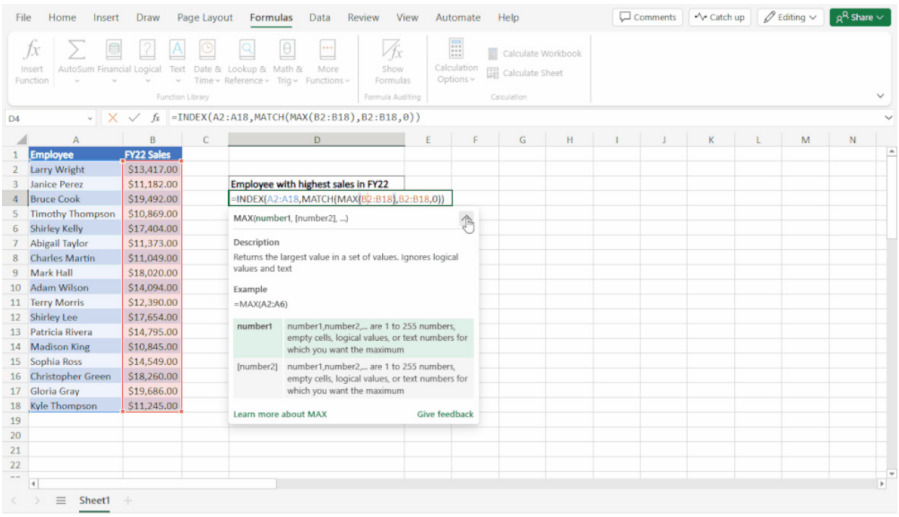
The 'Formula argument assistance' card provides descriptions of the function typed and the different arguments, as well as an example, *e.g.*

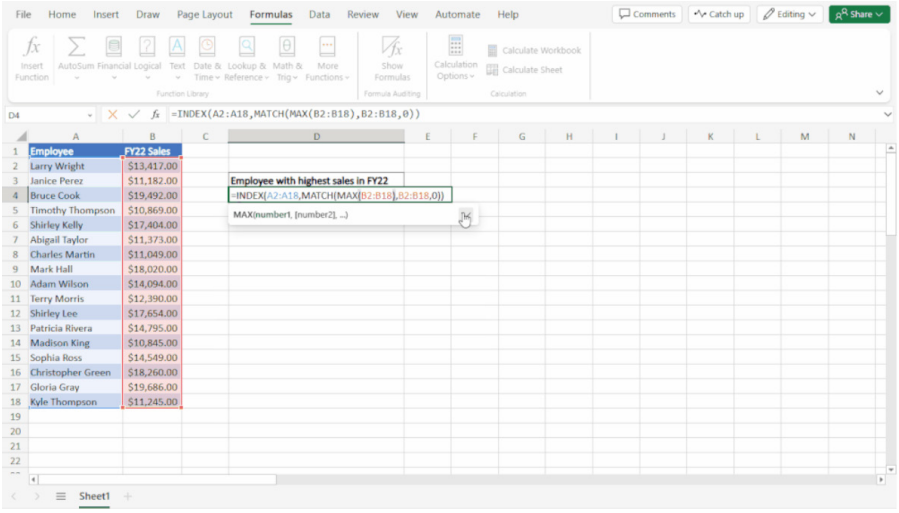


For better orientation, the relevant argument is highlighted during formula editing:

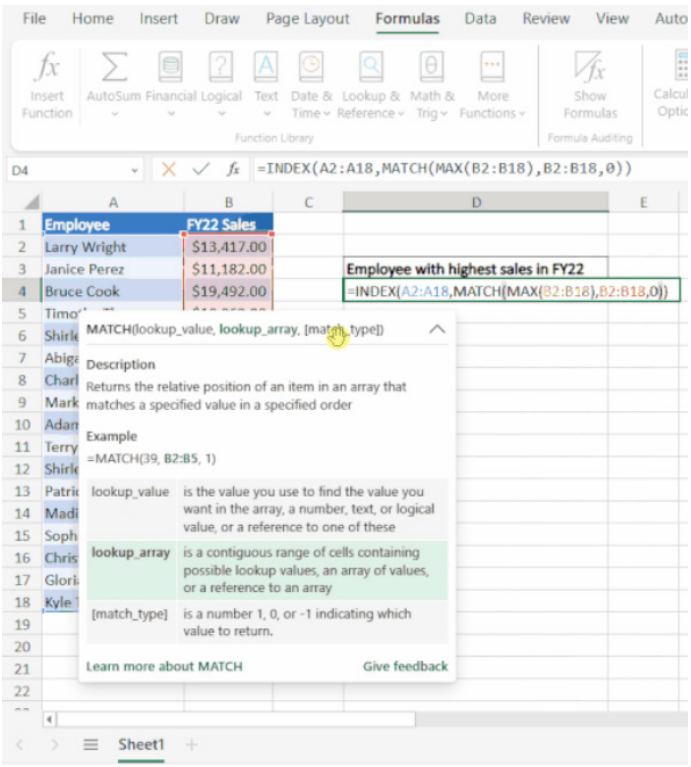


The card may be collapsed and expanded. You can collapse the card and display only the formula signature if you don't want to hide much of the grid:





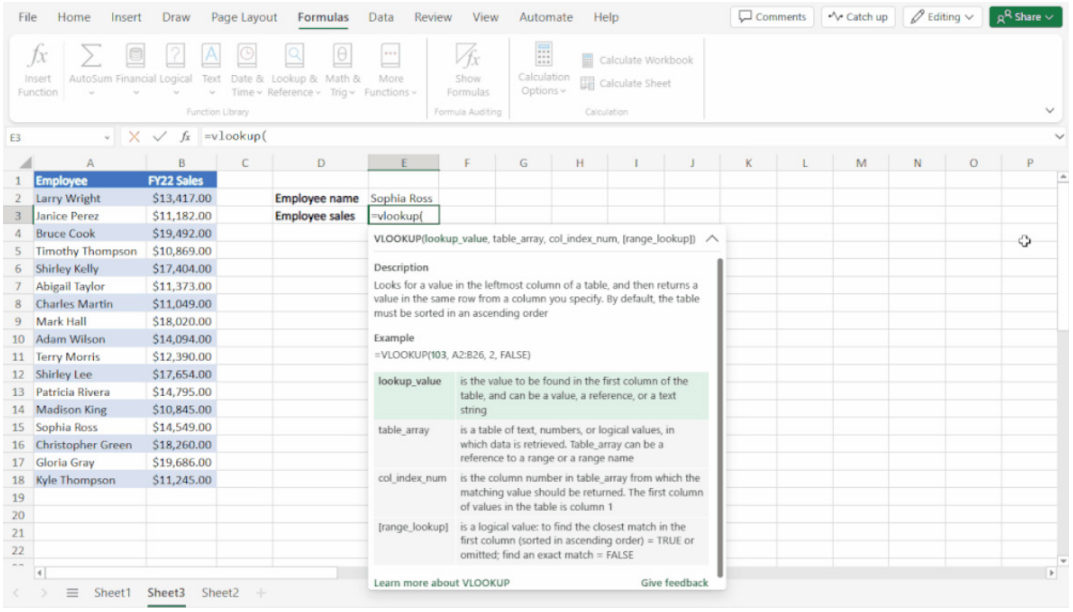
For convenience, the card may also be dragged and moved around the screen too:



To use it, start by typing a formula in a cell or in the Formula bar. After writing the formula name and the opening parenthesis (that's "bracket" to you and me), the card will appear. For example,

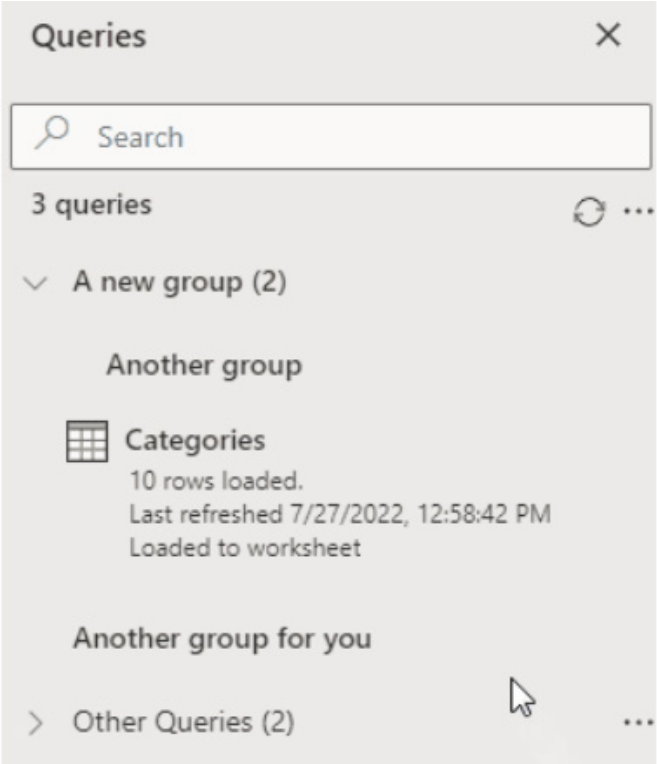
=VLOOKUP(

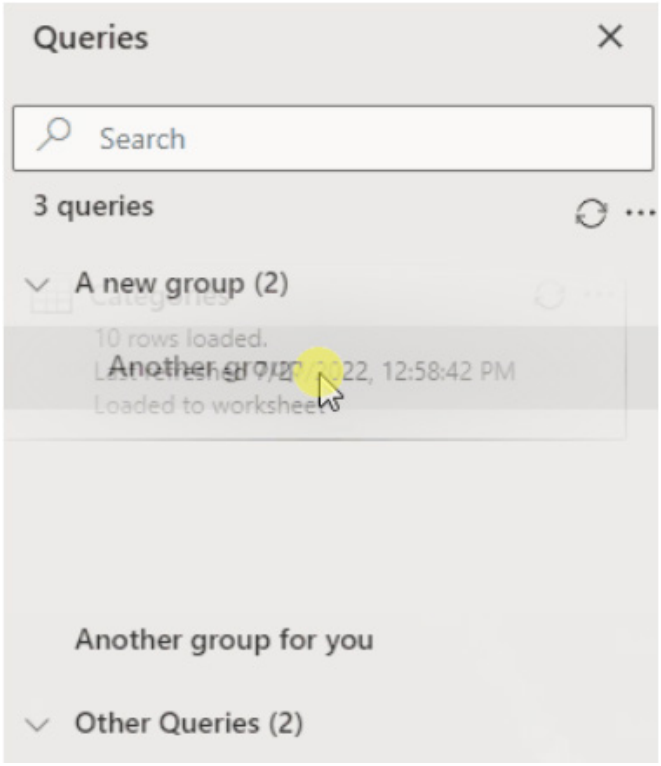
Otherwise, you may choose a function from the 'Insert function' dialog or from the dropdown menus in the Formulas tab.

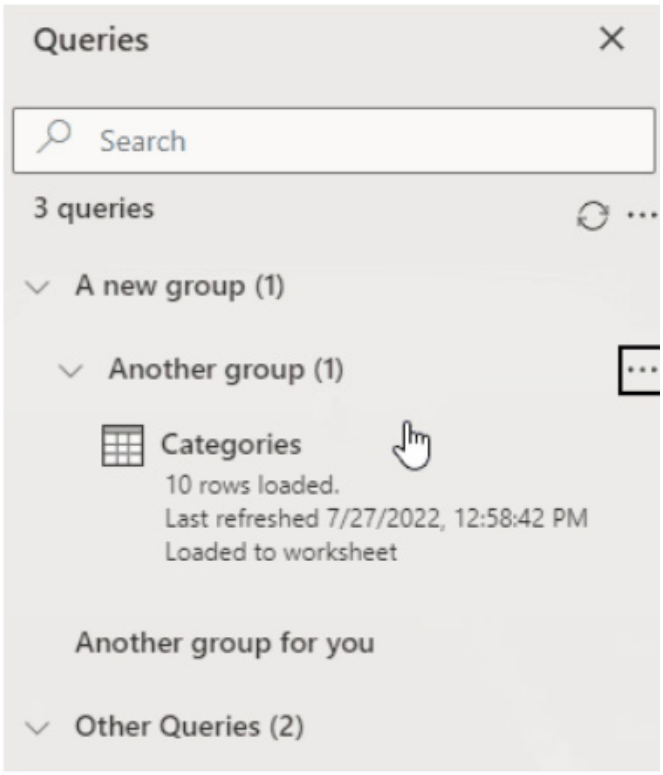


Drag and drop in Queries pane

Drag and drop is now available in the Queries pane for Excel for the web. This enhances your ability to arrange your queries, *e.g.* you may now easily sort queries or move them between folders, *viz.*







Block untrusted XLL add-ins

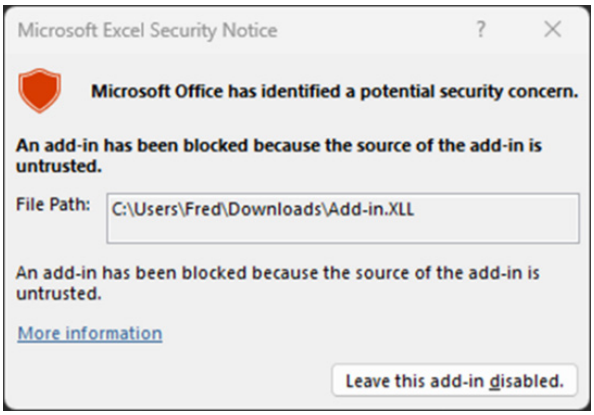
Originally in Insiders for Excel for Windows, this is now rolling out to all users on this platform.

Add-ins can add a lot of functionality to Microsoft 365, but they can also be employed by bad actors to disseminate malware. Thus, Microsoft has taken steps to help protect users by blocking XLL add-ins. Whether this is overkill, we will have to wait and see.

With this update in Excel for Windows (Insiders), you will no longer be able to enable XLL add-ins in files obtained from the internet with the click of a button, which will either be a blessing or a curse. Instead, a message will now appear, notifying you of the risk you are taking and offering a link to more information about possible workarounds.

It should be noted that this setting change only blocks XLL add-ins – not all add-ins. For example, add-ins from the Store will not be affected.

If you open an Excel file that contains an XLL add-in from an untrusted source, the following dialog will appear:



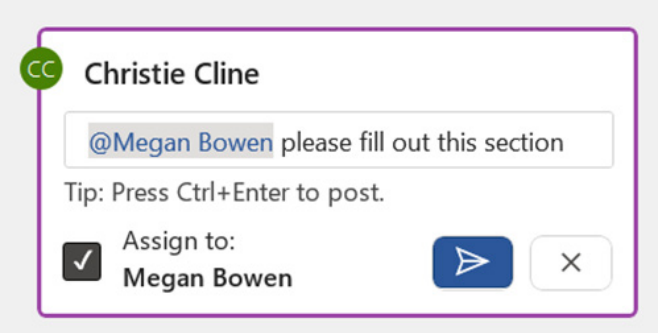
In the Microsoft Excel Security Notice message box, review the source of the add-in and click the 'Leave this add-in disabled' button, for example.

Assign a task with @mentions

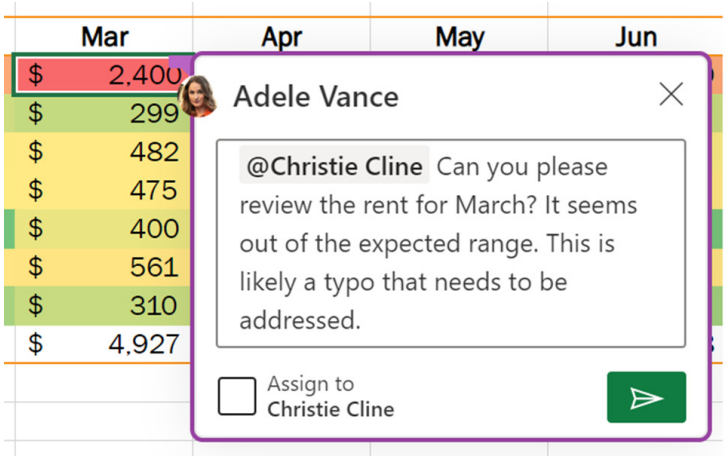
Here's one that nobody wants, coming to an Excel spreadsheet near you on Excel for Windows and Excel for Mac.

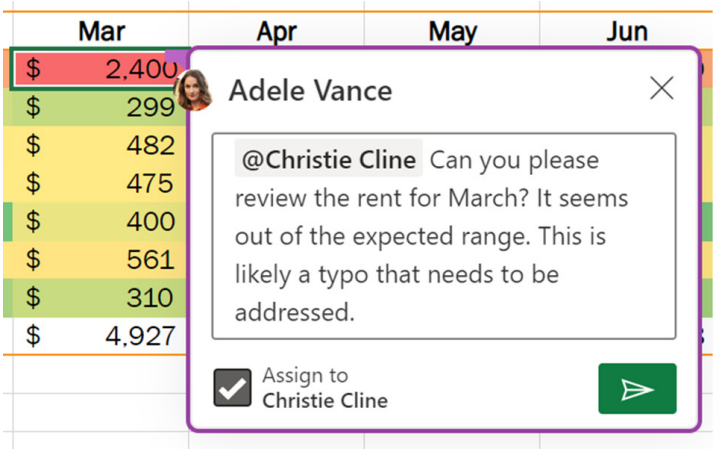
'Assign a task' allows users to collaborate more effectively with their teams by creating and assigning tasks within their Excel worksheets (and Word documents for that matter) using @mentions in comments and tagging a team member.

Once a team member is assigned to the task (by selecting the 'Assign to' check box to the comment to convert to a task and then clicking the blue arrow), they receive an email notification to let them know they have tasks to action:

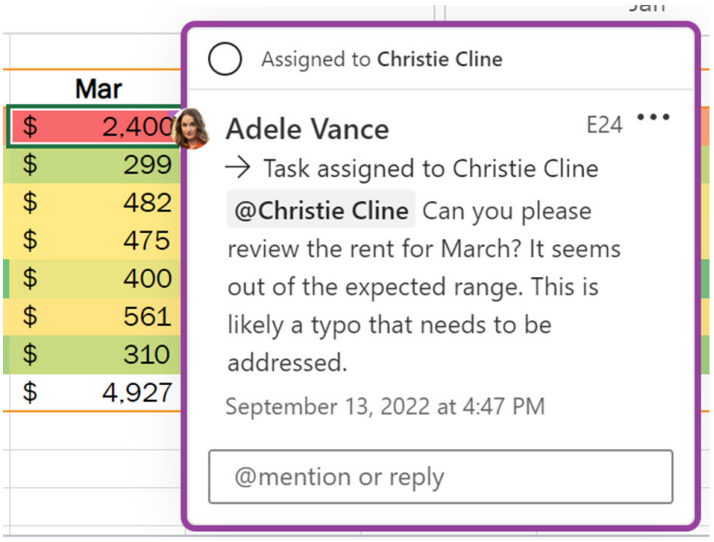


To use it, simply right-click on the cell that contains the information you want to comment on and select 'New Comment'. Then, write your comment and type '**@**' followed by the name of the team member you wish to tag:

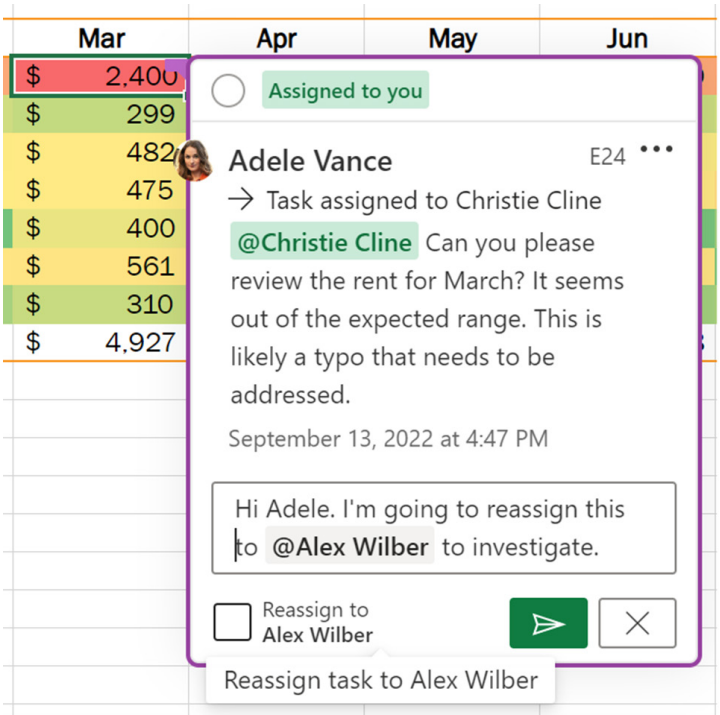




Click the green arrow pointing right or press **CTRL + ENTER** to post your comment and assign the task.

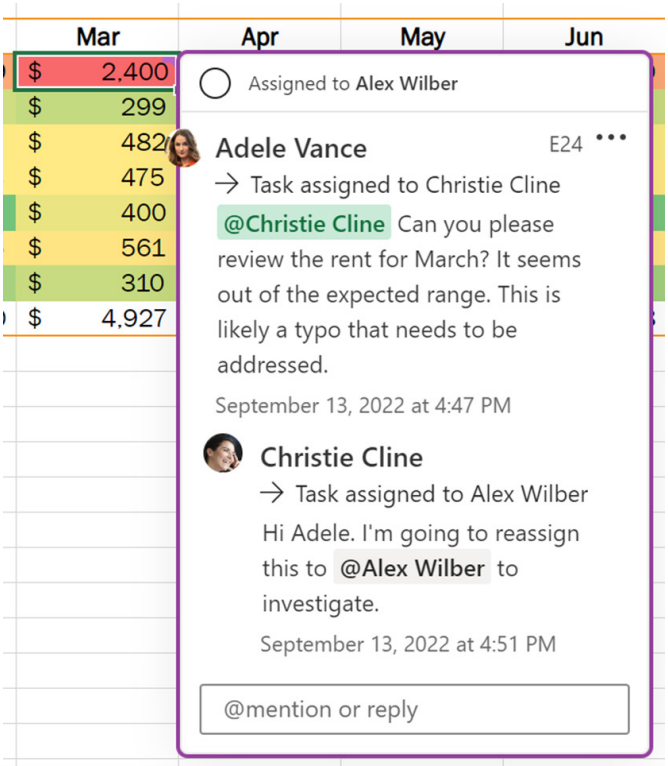


You may reassign a task by typing '**@**' followed by the name of the team member you want to reassign the task to in the response field, and then select the 'Reassign to' check box.



We imagine you can have hours of endless fun passing this parcel around the office…

Then, click the green arrow pointing right or press **CTRL + ENTER**; the person to whom you reassigned the task will be notified via email that they've been assigned the task.



To resolve a task, hover over the circle at the top of the comment and click 'Resolve thread'. Should you need to reopen a resolved task, again hover over the cell that contains the comment, and then click 'Reopen thread'.

It should be noted that to see Tasks in Excel, the Excel file must be stored in OneDrive or SharePoint, and that this feature is available to all users on the web and to Beta Channel users running:

- **Windows:** Version 2208 (Build 15504.10000) or later
- **Mac:** Version 16.66 (Build 22090700) or later.

The updated version of the grid with all the new features is fast becoming too complicated to show clearly here. Nonetheless, you can find the interactive links at aka.ms/ExcelFeaturesFlyer.

Excel Features Availability

Page 1 of 4

	Insider		Production				
Feature	Windows Find the latest Excel version for this platform	Mac Find the latest Excel version for this platform	Windows/CC Find the latest Excel version for this platform	Windows/MEC Find the latest Excel version for this platform	Windows/SA Find the latest Excel version for this platform	Mac Find the latest Excel version for this platform	Web
Block XLL Add-Ins	Version 2302 (Build 16130.20128) or later						
PivotTables: Manual Sort of Rows & Columns			Already Supported	Already Supported	Already Supported	Already Supported	February 2023
Automatic Recalculation Optimization	Version 2208 (Build 15529.10000) or later	Version 16.64 (Build 22081401) or later					
Import Data from SQL Server Database		Version 16.68 (Build 22110801) or later					
Import Data from Additional Sources						Version 16.69 (Build 23010700) or later	
Power Query Editor						Version 16.69 (Build 23010700) or later	
IMAGE function			Version 2211 (Build 15831.20190) or later	Version 2211 (Build 15831.20252) or later		Version 16.67 (Build 22102900) or later	December 2022
Check Formula with Value Preview Tooltips	Version 2302 (Build 16116.20000) or later	Version 16.70 (Build 230116) or later					
Office Scripts	Version 2212 (Build 15922.20000) or later	Version 16.68 (Build 22120101) or later					
Automate Tasks with Power Automate tab			Version 2301 (Build 15703.10000) or later			Version 16.66 (Build 22092500) or later	
PivotTable Show Details to XLO							January 2023
Excel Live in Teams							December 2022
Formula Suggestions							December 2022*
Formula by Example							December 2022*
Suggested Links							December 2022
Add search bar in queries pane							December 2022

*Starting to roll out

Features Flyer: aka.ms/ExcelFeaturesFlyer

CC: Current Channel, MEC: Monthly Enterprise Channel, SA: Semi Annual Enterprise Channel
All information is subject to change.

Excel Features Availability

Feature	Insider		Production				
	Windows Find the latest Excel version for this platform	Mac Find the latest Excel version for this platform	Windows/CC Find the latest Excel version for this platform	Windows/MEC Find the latest Excel version for this platform	Windows/SA Find the latest Excel version for this platform	Mac Find the latest Excel version for this platform	Web
Add keyboard shortcut to open PQ editor			Version 2211 (Build 15730.31883) or later				
Create nested PQ data types	Version 2211 (Build 15928.10000) or later						
Add Get Data from Dynamic Arrays	Version 2105 (Build 14014.20002) or later						
Data from picture			Version 2210 (Build 15723) or later	Version 2210 (Build 15726.20262) or later		Version 16.38 or later	December 2022
Chart Data Foils							November 2022
Show Changes			Version 2209 (Build 15703.10000) or later			Version 16.66 (Build 22092500) or later	Already Supported
New Paste Options	Version 2210 (Build 15726.20000) or later						
Quickly Find the Command you need			Version 2206 (Build 15331.20010) or later				October 2022
New DAX Functions	Version 2208 (Build 15504.10000) or later						
Navigation Pane			Version 2209 (Build 15629.10000) or later				
Smooth Scrolling			Version 2205 (Build 15225.20092) or later	Version 2208 (Build 15601.20230)	Version 2208 (Build 15601.20456) or later	Already Supported	Already Supported
Check Performance							September 2022
Share Section of Excel Workbook							September 2022
Dynamic Array Support in Charts	Version 2209 (Build 15617.10000) or later			Version 2210 (Build 15726.20262) or later			September 2022
Modern Comments			Version 2209 (Build 15427.20000) or later				
Manage Your Storage Accounts from Mac		Version 16.64 (Build 22082100) or later					

Features Flyer: aka.ms/ExcelFeaturesFlyer

CC: Current Channel, MEC: Monthly Enterprise Channel, SA: Semi Annual Enterprise Channel
All information is subject to change.

Excel Features Availability

Feature	Insider		Production				
	Windows Find the latest Excel version for this platform	Mac Find the latest Excel version for this platform	Windows/CC Find the latest Excel version for this platform	Windows/MEC Find the latest Excel version for this platform	Windows/SA Find the latest Excel version for this platform	Mac Find the latest Excel version for this platform	Web
New Excel functions			Version 2208 (Build 15427.20194) or later	Version 2208 (Build 15601.20230) or later		Version 16.64 (Build 22081401) or later	August 2022
Power Query Group operations							August 2022
Improvements to the connected Power BI experience	Version 2208 (Build 15601.20028) or later						August 2022
Add and edit rich text formatting			Already Supported	Already Supported	Already Supported	Already Supported	August 2022
Sort by color or icon from auto filter menu			Already Supported	Already Supported	Already Supported	Already Supported	August 2022
Edit files with legacy data connections			Already Supported	Already Supported	Already Supported	Already Supported	August 2022
Edit files with legacy Shared Workbook feature			Already Supported	Already Supported	Already Supported	Already Supported	August 2022
Delete chart elements							August 2022
Multiline formula bar							August 2022
Search within PivotTable Field List			Already Supported	Already Supported	Already Supported	Already Supported	July 2022
Set automatic data conversions	Version 2207 (Build 15427.20000) or later						
Natural Language Query Improvements			Version 2206 (Build 15330.20230) or later	Version 2205 (Build 15225.20356) or later		Version 16.63 (Build 22070801) or later	
Resize Conditional Formatting dialog box		Version 16.64 (Build 22070600) or later					
Sheet protection			Already Supported	Already Supported	Already Supported	Already Supported	June 2022

Features Flyer: aka.ms/ExcelFeaturesFlyer

CC: Current Channel, MEC: Monthly Enterprise Channel, SA: Semi Annual Enterprise Channel
All information is subject to change.

Excel Features Availability

Feature	Insider		Production				
	Windows Find the latest Excel version for this platform	Mac Find the latest Excel version for this platform	Windows/CC Find the latest Excel version for this platform	Windows/MEC Find the latest Excel version for this platform	Windows/SA Find the latest Excel version for this platform	Mac Find the latest Excel version for this platform	Web
Semi-select for links creation			Already Supported	Already Supported	Already Supported	Already Supported	June 2022
Add "PivotTable Connections to Slicer settings pane			Already Supported	Already Supported	Already Supported	Already Supported	June 2022
Import from local text, CSV, and XLSX files						Version 16.57 (22011100) or later	
Provide automatic alt-text suggestions on charts and PivotCharts			Version 2205 (Build 15225.20288) or later	Version 2204 (Build 15128.20280) or later		Version 16.62 (22061100) or later	
Power Query refresh for selected data sources			Already Supported	Already Supported	Already Supported	Already Supported	May 2022
Changing source file for workbook links			Already Supported	Already Supported	Already Supported	Already Supported	May 2022
Improved Recommended PivotTable experience	Version 2204 (Build 15128.10000) or later						
Faster recalc on resource constrained devices		Version 16.62 (Build 22050804) or later	Version 2204 (Build 15128.20248) or later	Version 2204 (Build 15128.20280) or later			
Faster AutoFilter				Version 2204 (Build 15128.20248) or later	Version 2208 (Build 15601.20456) or later	Version 16.61 (22050700) or later	
Dataflow connector				Version 2203 (Build 15028.20248) or later			
Dataverse connector			Version 2204 (Build 15128.20178) or later				
Improved Find dialog and Find All						Version 16.60 (220410) or later	
LAMBDA: Helper Functions			Version 2202 (Build 14931.20120) or later	Version 2202 (Build 14931.20274) or later	Version 2208 (Build 15601.20456) or later	Version 16.56 (Build 211211) or later	Already Supported

Features Flyer: aka.ms/ExcelFeaturesFlyer

CC: Current Channel, MEC: Monthly Enterprise Channel, SA: Semi Annual Enterprise Channel
All information is subject to change.

More next month, we're sure.

The A to Z of Excel Functions: LARGE

Are you living life **LARGE**? This function returns the **k**th largest value in a data set. You can use this function to select a value based on its relative standing. For example, you can use **LARGE** to return the highest, runner-up, or third-place score. Its syntax is as follows:

=LARGE(range, k)

LARGE has the following arguments:

- **array:** this is required and represents the array or range of data for which you want to determine the **k**th largest value
- **k:** this is also required. This denotes the position (from the largest) in the array or cell range of data to return.

It should be noted that:

- if **array** is empty**, LARGE** returns the *#NUM!* error value
- if **k** ≤ 0 or if **k** is greater than the number of data points, **LARGE** returns the *#NUM!* error value
- if **n** is the number of data points in a range, then **LARGE(array, 1)** returns the largest value, and **LARGE(array, n)** returns the smallest value.

k must be a positive integer less than or equal to the number of non-blank items in the **range**. For example,

	F	H	I
12	Values	Largest	
13	110	466	=LARGE(F13:F27,1)
14	438		
15	351	2nd Largest	
16	466	462	=LARGE(F13:F27,2)
17	330		
18	33	3rd Largest	
19	40	438	=LARGE(F13:F27,3)
20	347		
21	430		
22	146		
23	341		
24	122		
25	54		
26	462		
27	56		

There are opportunities to create errors using this reasonably straightforward function:

	F	G	H	I
34	Values		Largest	
35	184		#NUM!	=LARGE(F35:G35,2) - doesn't consider blanks in a range
36	(147)		#NUM!	=LARGE(F36:G36,2) - value of n is larger than the number of items in the range
37	(249)		#NUM!	=LARGE(F37:G37,0.4) - value of n must be a positive integer less than or equal to number of non-blank items in the range

Again, other than choosing an inappropriate **n** (*e.g.* choosing a negative value, too large a value or a non-integer), blank cells may again cause problems. Ensure you do not include blank cells in your given **range**.

LARGE may be used to rank numerical data in descending order using the **ROWS** function:

	F	H	
46	Values	Descending	
47	166	428	=LARGE($F$47:$F$61,ROWS($G$47:$G47))
48	370	425	
49	328	398	
50	98	392	
51	291	389	
52	392	370	
53	428	365	
54	425	328	
55	389	312	
56	54	291	
57	312	176	
58	398	166	
59	91	98	
60	176	91	
61	365	54	

LARGE may also be used to derive statistical data from a range, sometimes requiring an array formula and sometimes not (this is often a case of trial and error for the inexperienced). For example, here's two ways to calculate the sum of the top three (largest) values in the following range:

	F	H	J	K
70	Values	Top 3	Sum - Alt 1	
71	431	1	1,309	{=SUM(LARGE(F71:F85,H71:H73))}
72	39	2		
73	249	3	Sum - Alt 2	
74	199		1,309	=SUM(LARGE(F71:F85,{1,2,3}))
75	165			
76	70			
77	352			
78	425			
79	260			
80	331			
81	278			
82	386			
83	154			
84	453			
85	242			

The formulae

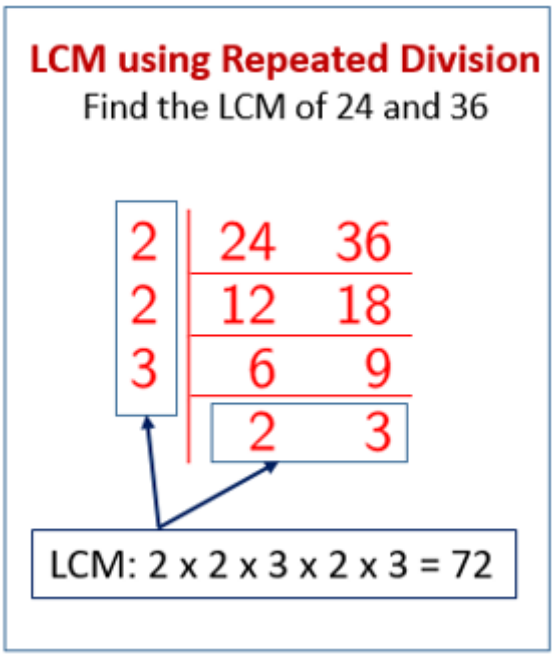
{=SUM(LARGE(F71:F85,H71:H73))} *and*

=SUM(LARGE(F71:F85,{1,2,3}))

will both sum the top three items in the list. You should note that an array formula is avoided in the second formula as **n** is specified as **{1,2,3}** – effectively creating an array of data without pressing **CTRL + SHIFT + ENTER**.

Similar formulae may be created for the sum of the bottom five, the average of the fourth, eighth and 12th largest items and so on. Try doing that with **MAX** or **MIN**!

The A to Z of Excel Functions: LCM



This function returns the least (lowest) common multiple of integers. The least common multiple is the smallest positive integer that is a multiple of all integer arguments. You can use **LCM** to find the first positive number (in ascending order) that all arguments will divide with no remainder.

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

LCM(number1, [number2], …)

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

- **number1, [number2], …: number1** is required, but subsequent numbers are optional. This can be one (1) to 255 values for which you want the least common multiple. If each number is not an integer, it is truncated.

It should be further noted that:

- if any argument is non-numeric, **LCM** returns the *#VALUE!* error value
- if any argument is less than zero, **LCM** returns the *#NUM!* error value
- if **LCM(a,b)** ≥ 2^53, **LCM** returns the *#NUM!* error value
- this function is often used with **GCD**, which determines the greatest common divisor (*i.e.* the highest number that will divide each number), in order to add fractions, for instance *(see below)*.

Please see our examples below:

Formula	Description	Result
=LCM(48,120)	Least common multiple of 48 and 120 (240 / 48 = 5 and 240 / 120 = 2)	240
=GCD(48,120)	Greatest common divisor of 48 and 120 is 24 (48 / 24 = 2 and 120 / 24 = 5)	24
=LCM(0,24)	If one or more non-negative argument is zero, LCM is zero	0
=LCM(-0.25,10)	Arguments of LCM may not be negative - even if truncated	#NUM!

Fractions:

$$\frac{3}{5} + \frac{1}{15} = \frac{9}{15} + \frac{1}{15} = \frac{10}{15} = \frac{2}{3}$$

*=(B9*F10/B10)* *=D9*H10/D10* *=F9+H9* *=I9/GCD(I9,I10)*
=LCM(B10,D10) *=F10* *=F10* *=I10/GCD(I9,I10)*

More Excel Functions next month.

Beat the Boredom Suggested Solution

The challenge this month was to find and replace multiple values at once in Excel.

The Challenge

How did you usually find and replace a character in a text string in Excel? Many people use the 'Find & Replace' functionality in Excel, which works fine for single replacements. However, if you have tens, hundreds or thousands of items to replace this method seems very tedious and inefficient. Luckily, there were several ways to do the mass replacement in Excel more elegantly and effectively.

This month's challenge was to write a **formula** to replace multiple letters in some text strings. The replacement table (here, called **Replace**) might be as follows:

Old Character	New Character
ô	o
ê	e
ç	c
é	e
í	i
â	a
ử	u
ả	a
吳	Wu
奎	Kui
燁	Ye
雲	kumo
泥	doro
の	no
差	sa
1	Yi
2	Er
3	San
4	Si
5	Wu
6	Liu
7	Qi
8	Ba
9	Jiu
0	Ling

The result should look similar to the Table **Texts** *(below)*:

Old Text	New Text
Ngô Khôi Huê	Ngo Khoi Hue
Quân tử nhất ngôn	Quan tu nhat ngon
吳奎燁	Wu Kui Ye
520	Wu Er Ling
1314	Yi San Yi Si
雲泥の差	kumo doro no sa
Açaí da Paçoca	Acai da Pacoca
Marília Méndonça	Marilia Mendonca

As always, there were some requirements:

- the formula needed to be within just one column (no "helper" cells)
- this was a formula challenge; no Power Query / Get & Transform or VBA
- the formula should be dynamic enough to update when a similar text string is added
- only Excel 2016 functions were accepted.

Suggested Solution

Before we discuss the solution, I would like to note there are several complicating factors here. Let's go through them.

Problem 1: SUBSTITUTION – a Pragmatic Approach

For a standard approach, you might think of using multiple nested **SUBSTITUTE** functions here along with **INDEX** functions. For example, if we have a single character to replace here, we will use the following formula:

=SUBSTITUTE(Texts[@Text],

INDEX(Replace[@Old Character], 1), INDEX(Replace[@New Character], 1))

In the formula above, the **Texts**, **Replace** is the name of the table. Therefore, **Texts[@Old Text], [@Old Character]**, and **[@New Character]** specify one row of column **Text**, **Old Character**, and **New Character** respectively. If we want to replace two [2] characters, we can nest the function like this:

=SUBSTITUTE(SUBSTITUTE(Texts[@Text],

INDEX(Replace[@Old Character], 1), INDEX(Replace[@New Character], 1)),

INDEX(Replace[@Old Character], 2), INDEX(Replace[@New Character], 2))

Therefore, with the same principle, if we want to replace n values, we will nest our function like this:

=SUBSTITUTE(SUBSTITUTE(…SUBSTITUTE(Texts[@Old Text],

INDEX(Replace[@Old Character], 1), INDEX(Replace[@New Character], 1)),

INDEX(Replace[@Old Character], 2), INDEX(Replace[@New Character], 2)),

…

INDEX(Replace[@Old Character], n), INDEX(Replace[@New Character], n))

This pragmatic approach might work but it makes the function hard to inspect in case of any error occurs. Besides that, it is not dynamic, if we put more characters to replace, we will have to modify the formula to adapt to the new inputs. Therefore, this approach is rejected.

Problem 2: Sequencing in the non-SEQUENCE world

When Dynamic Arrays were introduced, they introduced many useful Excel functions that come along with them, especially the **SEQUENCE** function. This function helps us generate a list of sequential numbers in an array. Hence, we can combine the **SEQUENCE** function with other functions like **MID** or **LEN** to transform the text string into individual cells in a spreadsheet with the following formula:

=MID(Texts[@Old Text], SEQUENCE(1, LEN(Texts[@Old Text])),1)

SEQUENCE(1, LEN(Texts[@Old Text])) will help us create a horizontal list of the consecutive text string from one [1] to the last number which is equal to the length of the string. For example:

	A	B	C	D	E
1	1	2	3	4	5

The **MID** function will then extract each character of a string with the starting point one by one, equal to the number list created by **SEQUENCE** above.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	N	g	ô		K	h	ô	i		H	u	ê					
2	Q	u	â	n		t	ử		n	h	ấ	t		n	g	ô	n
3	吳	奎	燁														
4	5	2	0														
5	雲	泥	の	差													
6	A	ç	a	í		d	a		P	a	ç	o	c	a			
7	M	a	r	í	l	i	a		M	é	n	d	o	n	ç	a	

However, the problem is **SEQUENCE** does not exist in Excel 2016. Hence, we must find a workaround to do sequencing in “legacy” Excel that does not have the **SEQUENCE** function. Rats.

Problem 3: Number or not Number?

One minor problem you might encounter is whether the value in the Replace table is in the number type or the text type. Therefore, you have to convert these numbers into the same type. If you do not convert them into the same type, it can cause a problem for you later on when you use any lookup or match function. The idea here is that you also have to include the conversion within the formula as well.

Problem 4: Legacy Excel World

Before Dynamic Arrays became a thing, we had previously written an array formula in what we now call a legacy **CTRL + SHIFT + ENTER** (**CSE**) array. Thus, what is the difference between those two? First, the "legacy" CSE array formulae require entering **CTRL + SHIFT + ENTER** after completing the formula. For some formulae, we have to select a range for the output and then press the combination **CTRL + SHIFT + ENTER** whereas the Dynamic Array functions do not require this method of entry. This is not an issue, but it requires us to visualise the outputs before they are generated so that we select the correct range for the outputs.

Brainstorming

To address the replacement for the **SEQUENCE** function, may we introduce you to the **INDIRECT** and **ROW** functions? Let's employ the following trick. Consider the following formula:

=ROW(INDIRECT("A1:A"&LEN(Texts[@[Old Text]])))

With respect to **"A1:A"&LEN(Texts[@[Old Text]]** this part of the formula will create a text string which is the range of the cell **A1** to one of the cells in column **A**, depending upon the length of the text string. For example, the first text string has 12 characters, so this piece of formula will return **A1:A12**.

Then we need a formula that can capture a text string here, since **ROW** requires a reference input, so it will not be able to capture a text reference here. This is where the **INDIRECT** function comes in handy since it is used to capture the text reference. Therefore, we wrap the **INDIRECT** around the text string above, then we will have the value of the first column. However, it will return an *#VALUE!* error in some versions of Excel, so what you can do here is select the cell containing your formula and a range that equal to the length of the old text string you have. Next, you press **CTRL + SHIFT + ENTER**, and voilà you generate the following:

	A
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	

After that, we wrap the **ROW** function around the **INDIRECT** function we have. Then, the formula we just create can mimic the **SEQUENCE** function which gives us the following sequence:

	A	B
1	0	1
2	0	2
3	0	3
4	0	4
5	0	5
6	0	6
7	0	7
8	0	8
9	0	9
10	0	10
11	0	11
12	0	12

Backed to suggested solutions

Now return to the example of **SEQUENCE**, **MID,** and **LEN** functions we discuss in problem two [2], we can finally replace the **SEQUENCE** function with the formula we create earlier:

=MID(Texts[@[Old Text]],ROW(INDIRECT("A1:A"&LEN(Texts[@[Old Text]]))),1)

This formula will give us the following visual:

	A	B	C
1	0	1	N
2	0	2	g
3	0	3	ô
4	0	4	
5	0	5	K
6	0	6	h
7	0	7	ô
8	0	8	i
9	0	9	
10	0	10	H
11	0	11	u
12	0	12	ê

From here we can write a lookup function to match the character in the **Texts[@[Old Text]]** with the **Replace[Old Character]** and we return these matches with the accordance **Replace[New Character]**.

Here we suggested you use the **INDEX** and **MATCH** functions. We will have the following formula for the **MATCH** function:

=MATCH(MID(Texts[@[Old Text]],ROW(INDIRECT("A1:A"&LEN(Texts[@[Old Text]]))),1), Replace[Old Character],0)

This will be able to match correctly for the first second and third text strings but the fourth test string which is a number will not match. This is the issue we discuss in problem three [3], so for it to work we must wrap our lookup array argument in the **TEXT** function to transform all the characters that have number type into text type.

=TEXT(Replace[Old Character],"0")

Therefore, our complete function will look like this:

=MATCH(MID(Texts[@[Old Text]],ROW(INDIRECT("A1:A"&LEN(Texts[@[Old Text]]))),1),TEXT(Replace[Old Character],"0"),0)

After writing the formula we will have the following visual:

	A	B	C	D
1	0	1	N	#N/A
2	0	2	g	#N/A
3	0	3	ô	1
4	0	4		#N/A
5	0	5	K	#N/A
6	0	6	h	#N/A
7	0	7	ô	1
8	0	8	i	#N/A
9	0	9		#N/A
10	0	10	H	#N/A
11	0	11	u	#N/A
12	0	12	ê	2

Then we wrap the **INDEX** function around the **MATCH** function to replace the old characters in the text string with new characters:

=INDEX(Replace[New Character],MATCH(MID(Texts[@[Old Text]],ROW(INDIRECT("A1:A"&LEN(Texts[@[Old Text]]))),1),TEXT(Replace[Old Character],"0"),0))

So, we have the following array:

	A	B	C	D	E
1	0	1	N	#N/A	#N/A
2	0	2	g	#N/A	#N/A
3	0	3	ô	1	o
4	0	4		#N/A	#N/A
5	0	5	K	#N/A	#N/A
6	0	6	h	#N/A	#N/A
7	0	7	ô	1	o
8	0	8	i	#N/A	#N/A
9	0	9		#N/A	#N/A
10	0	10	H	#N/A	#N/A
11	0	11	u	#N/A	#N/A
12	0	12	ê	2	e

We need all the places where it shows *#N/A!* to keep the original text, so we can use the **IFERROR** function here:

=IFERROR(INDEX(Replace[New Character],MATCH(MID(Texts[@[Old Text]],ROW(INDIRECT("A1:A"&LEN(Texts[@[Old Text]]))),1),TEXT(Replace[Old Character],"0"),0)),MID(Texts[@[Old Text]],ROW(INDIRECT("A1:A"&LEN(Texts[@[Old Text]]))),1))

Which will give us the following visual:

	A	B	C	D	E	F
1	0	1	N	#N/A	#N/A	N
2	0	2	g	#N/A	#N/A	g
3	0	3	ô	1	o	o
4	0	4		#N/A	#N/A	
5	0	5	K	#N/A	#N/A	K
6	0	6	h	#N/A	#N/A	h
7	0	7	ô	1	o	o
8	0	8	i	#N/A	#N/A	i
9	0	9		#N/A	#N/A	
10	0	10	H	#N/A	#N/A	H
11	0	11	u	#N/A	#N/A	u
12	0	12	ê	2	e	e

At this step, we introduced the last function that will help you merge all the arrays which is the **CONCAT** function. The **CONCAT** function is the succeeding **CONCATENATE** function and is used to merge text. Therefore, you wrap this function around all the formulas we have so far, and you will have the new text string here:

=CONCAT(IFERROR(INDEX(Replace[New Character],MATCH(MID([@[Old Text]],ROW(INDIRECT("A1:A"&LEN([@[Old Text]]))),1),TEXT(Replace[Old Character],"0"),0)),MID([@[Old Text]],ROW(INDIRECT("A1:A"&LEN([@[Old Text]]))),1))

As we know **CONCAT** function will merge all our arrays into one [1] cell so, we can safely press Ctrl + Shift + Enter to get the results of multiple replacements. Which will give you the following result table:

Old Text	New Text
Ngô Khôi Huê	Ngo Khoi Hue
Quân tử nhất ngôn	Quan tu nhat ngon
吳奎燁	Wu Kui Ye
520	Wu Er Ling
1314	Yi San Yi Si
雲泥の差	kumo doro no sa
Açaí da Paçoca	Acai da Pacoca
Marília Méndonça	Marilia Mendonca

How cool is that?

Word to the Wise

Although this solution works quite well when replacing a single **Replace[Old Character]** in the **Texts[@Old Text]** for a single or multiple **Replace[New Character]**, it cannot replace multiple characters of the **Texts[@Old Text]** with a single or multiple **Replace[Old Character]**. However, this could be a challenge for another newsletter...

Until next time.

Upcoming SumProduct Training Courses - COVID-19 update

Due to the COVID-19 pandemic that is currently spreading around the globe, we are suspending our in-person courses until further notice. However, to accommodate the new working-from-home dynamic, we are switching our public and in-house courses to an online delivery stream, presented via Microsoft Teams, with a live presenter running through the same course material, downloadable workbooks to complete the hands-on exercises during the training session, and a recording of the sessions for your use within 1 month for you to refer back to in the event of technical difficulties. To assist with the pacing and flow of the course, we will also have a moderator who will help answer questions during the course.

If you're still not sure how this will work, please contact us at training@sumproduct.com and we'll be happy to walk you through the process.

Location	Course	Date	Date	Duration	Duration
Online (Australia)	Power Pivot, Power Query and Power BI	10 - 12 May 2023	09:00-17:00 AEST	(-1 day) 23:00-17:00 GMT	3 Days
Online (Australia)	Excel Tips and Tricks	17 May 2023	09:00-17:00 AEST	(-1 day) 23:00-17:00 GMT	1 Day
Online (Australia)	Financial Modelling	18 - 19 May 2023	09:00-17:00 AEST	(-1 day) 23:00-17:00 GMT	2 Days
Online (Australia)	Power Pivot, Power Query and Power BI	7 - 9 Jun 2023	09:00-17:00 AEST	(-1 day) 23:00-17:00 GMT	3 Days
Online (Australia)	Excel Tips and Tricks	14 Jun 2023	09:00-17:00 AEST	(-1 day) 23:00-17:00 GMT	1 Day
Online (Australia)	Financial Modelling	15 - 16 Jun 2023	09:00-17:00 AEST	(-1 day) 23:00-17:00 GMT	2 Days

Key Strokes

Each newsletter, we'd like to introduce you to useful keystrokes you may or may not be aware of. This month, we said we'd look at **ALT** plus numbers – so we have. However, it's not very exciting:

Keystroke	What it does
ALT + 1	Performs first custom item on Quick Access Toolbar
ALT + 2	Performs second custom item on Quick Access Toolbar, *etc.*

There are *c.*550 keyboard shortcuts in Excel. For a comprehensive list, please download our Excel file at 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 www.sumproduct.com/training.

Sydney Address: SumProduct Pty Ltd, Suite 803, Level 8, 276 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, Ground Floor, 470 St Kilda Road, Melbourne, VIC 3004
Registered Address: SumProduct Pty Ltd, Level 14, 440 Collins Street, Melbourne, VIC 3000

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