SumProduct

NEWSLETTER #134 - January 2024

www.sumproduct.com | www.sumproduct.com/thought

Happy new year! Out with the old, in with the new. Well, that gets rid of me then…

Our first newsletter starts by addressing a lot of queries we received from last month's Beat the Boredom challenge, and deep dives on data validation. Is our decision valid? Judge for yourself.

Speaking of our Beat the Boredom Challenge, we provide you with our first of 2024, plus we have more tips on Charts & Dashboards, continued with our new Excel for Mac article, and then there is also Visual Basics, Power Pivot Principles, Power Query Pointers, Excel Updates and our A to Z's of both Keyboard Shortcuts and Excel functions.

Noticeable by its absence appears to be our regular take on Power BI Updates. But that isn't quite the case. Seasonal printing deadlines prohibit any news on this front this month – but we make up for it with an in-detail review of the relatively new **DAX** query view in Power BI Desktop. You can read all about it inside this month's behemoth of a newsletter.

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

Liam Bastick, Managing Director, SumProduct

Dependent Data Validation

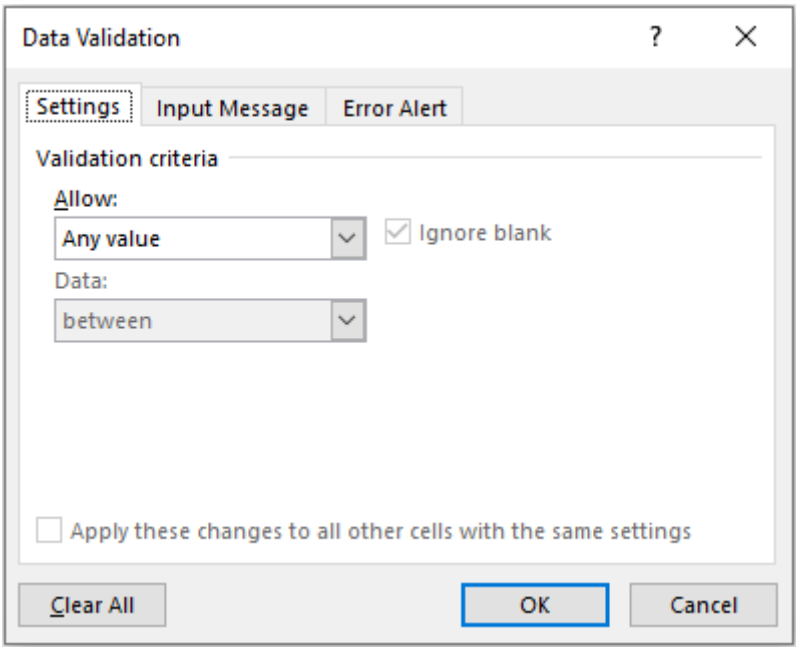
Last month's ***Beat the Boredom*** challenge certainly generated momentum in our newsletter's community, with all sorts of follow-up queries. Therefore, we thought we'd look further at dependent data validation lists as a technical article for this month.

There are several ways that Excel may control what end users mighty input into a spreadsheet, but one of the simplest and most intuitive is the use of **data validation**. I must admit that this is one of Excel's functionalities I am guilty of assuming everyone knows. But this does not appear to be the case!

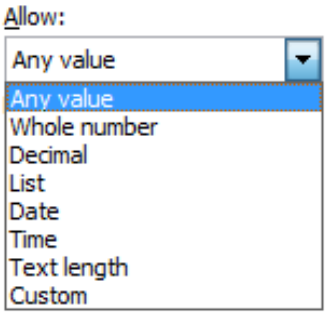
To access data validation, from any cell in Excel:

- on the Data tab of the Ribbon, go to the 'Data Tools' group and click the 'Data Validation' icon (**ALT + A + V + V**)
- alternatively, the old Excel 2003 and earlier keyboard shortcut **ALT + D + L** still works.

This brings up the following dialog box:

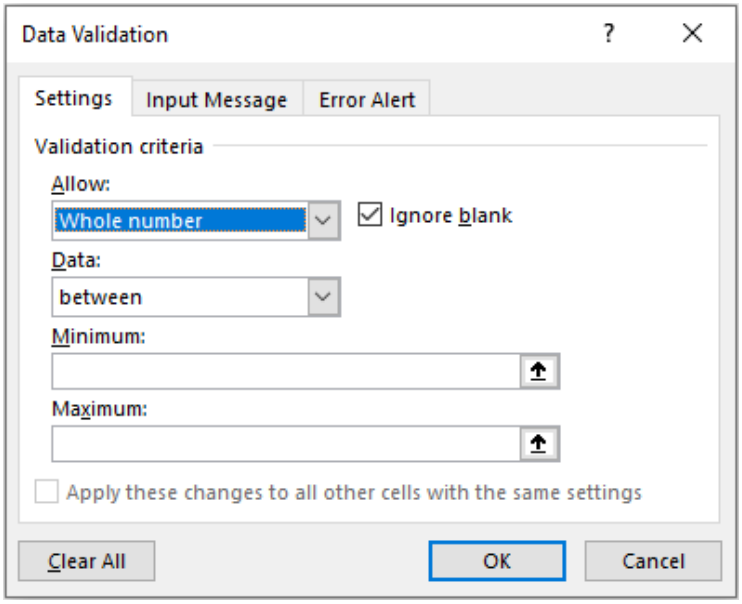


The default setting for all cells in Excel is to allow any value *(pictured)*. This can be changed by changing the selection in the 'Allow' drop down box. It may be modified to any of the following:

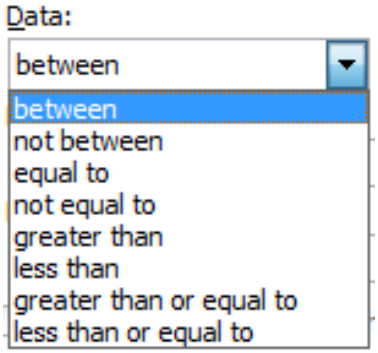


Most of these criteria do exactly what they say on the tin: by choosing 'Decimal', the input must be a number, whereas 'Whole Number' allows for integers only. However, making a selection from the 'Allow' drop down box is only the first part of the data validation process.

Once a selection has been made (for example, I will use 'Whole Number'), the dialog box will change appearance, *viz*.

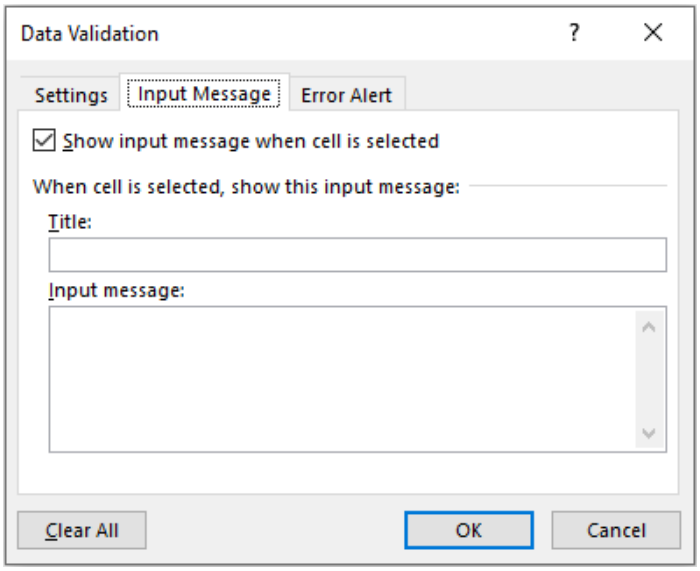


The 'Ignore blank' check box is no longer greyed out. This allows blank cells to be 'valid' regardless of the criteria selected. The remainder of the dialog box is governed by the 'Data' drop down box. There are various selections that may be made:

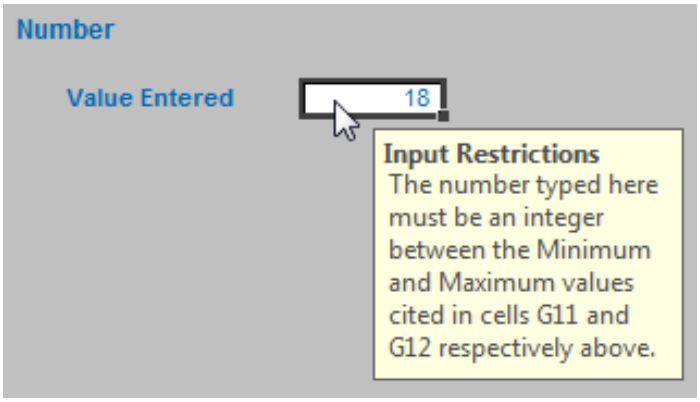


Depending upon the choice made, the box will prompt for values (*e.g.* Minimum and Maximum in the illustration above) which can be typed in, or else the values can refer to cell references directly or indirectly via range names.

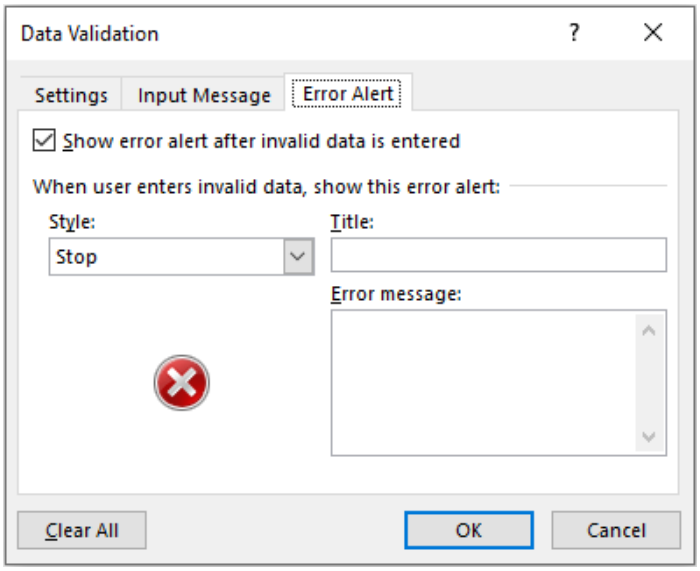
One the choices have been made, you might wish to utilise the other two tabs of the 'Data Validation' dialog box.



With the 'Show input message when cell is selected' checked, if the end user selects the data validated, when the cell in question is selected the message typed in here will appear. This can make data inputs in a model much simpler as end users are 'spoon fed' with a pop-up box detailing what to do. In the example below, the 'Input Restrictions' comment only appears when the cell is selected:



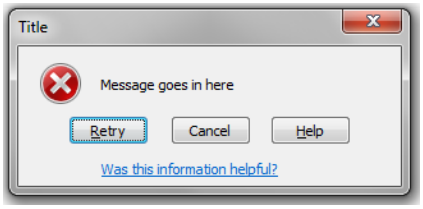
The third tab selects what to do if invalid data is entered in the cell:



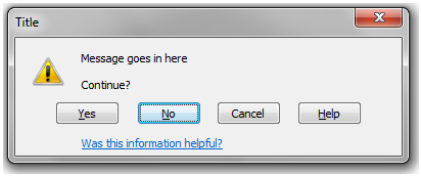
This alerts the end user when an invalid entry has been made (*e.g.* typing "dog" when a number is expected) – as long as the 'Show error alert after invalid data is entered' check box is ticked.

There are three styles available:

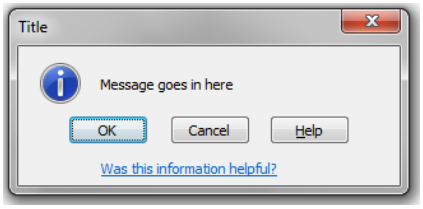
Stop



Warning



Information

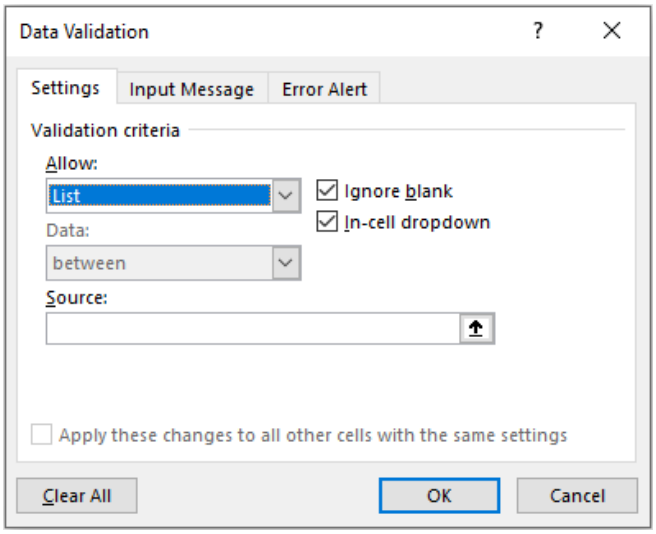


The three styles provide differing treatment of invalid data:

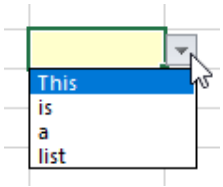
- **Stop:** the value will not be accepted and the end user will be prompted to retry
- **Warning:** the end user will be warned that the data is invalid, but be asked whether it is OK to continue
- **Information:** the end user will be advised that the data is invalid but that the data has been accepted.

If the 'Show error alert after invalid data is entered' check box is not ticked, no prompt will occur and invalid data will be accepted in the cell without any warning.

Whole Number, Decimal, Date, Time and Text Length are all relatively straightforward, albeit very similar in nature. Custom allows you to customise your criterion / criteria using a formula, but the one I wish to concentrate on here is **List**, which allows the end user to select from a list.



With 'List' selected, the dialog box prompts for a source for the list. Entries may be typed in, separated by a comma, but you can also use cell references or range names. For lists, I strongly recommend using the 'In-cell dropdown' which provides a dropdown list of valid entries once the cell has been selected.



Data validated lists are great for forcing end users to choose from a predetermined selection, and prevent typos, accidental capitalisation and additional spaces from occurring – all of which can cause dependent formulae to fail, if left unnoticed.

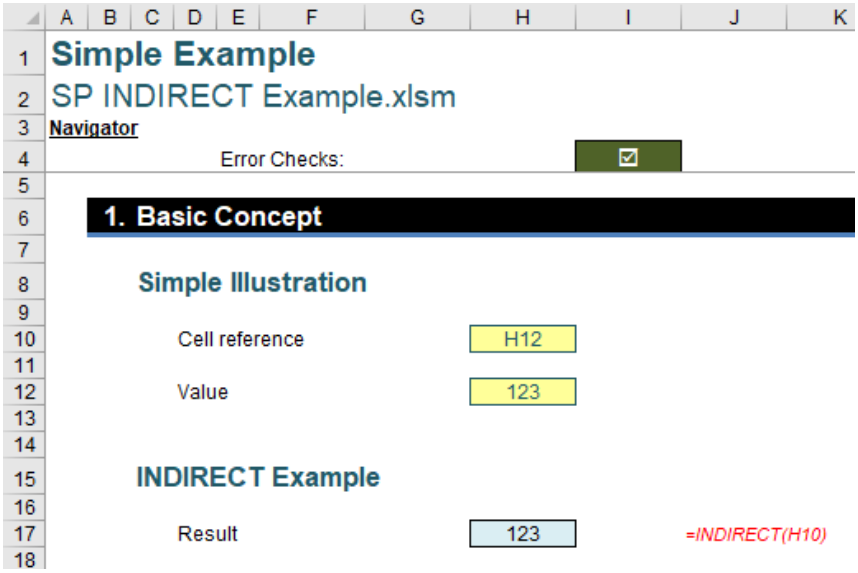
Sometimes, you wish to have one or more lists depend upon the result of another list selection, the simplest scenario being the hierarchical dependency (*i.e.* where you must select in a given order, first from one list, then another, *etc.*).

Hierarchical Data Validation Lists

This requires us to make use of Excel's often "hated" **INDIRECT** function, which allows the creation of a formula by referring to the contents of a cell, rather than the cell reference itself.

The **INDIRECT(reference_text, [a1])** function syntax has two arguments:

1. **reference_text:** this is a required reference to a cell that contains an **A1**-style reference, an **R1C1**-style reference, a name defined as a reference or a reference to a cell as a text string. If **reference_text** is not a valid cell reference, **INDIRECT** returns the *#REF!* error value. If **reference_text** refers to another workbook (an external reference), the other workbook must be open. If the source workbook is not open, **INDIRECT** again returns the *#REF!* error value
2. **[a1]:** this is optional (hence the square brackets) and represents a logical value that specifies what type of reference is contained in the cell **reference_text**. If **a1** is TRUE or omitted, **reference_text** is interpreted as an **A1**-style reference. If **a1** is FALSE, **reference_text** is interpreted as an R1C1-style reference. Essentially, **INDIRECT** works as follows:

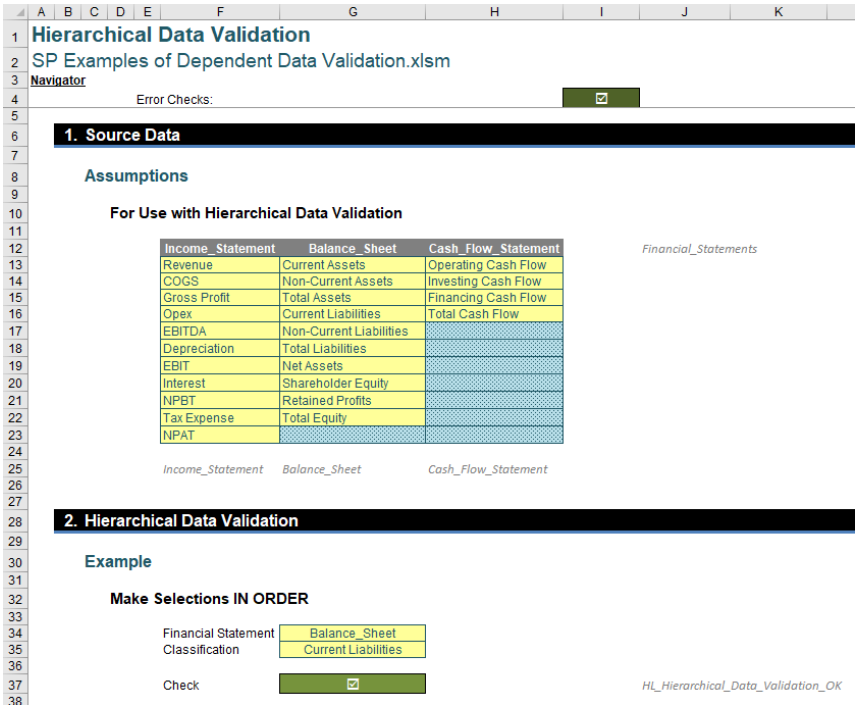


In the above example, the formula in cell **H17** (the blue cell) is given by

=INDIRECT(H10).

With only one argument in this function, **INDIRECT** assumes the **A1** cell notation (*e.g.* the cell in the third row fourth column is cell **D3**). Note that the value in cell **H10** is "H12", so this formula returns the value / contents of cell **H12**, *i.e.* 123.

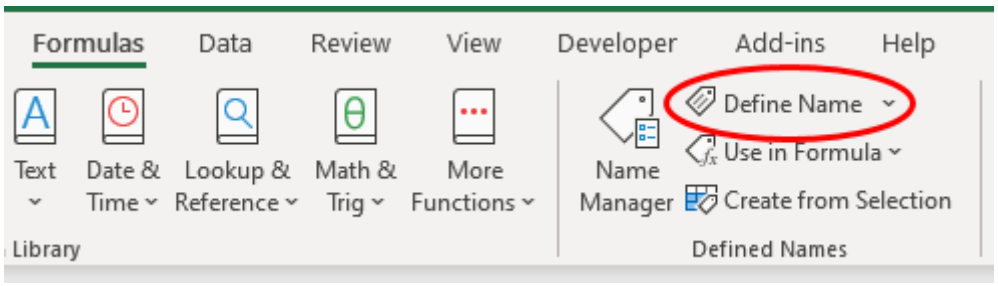
Let's now consider an hierarchical data validated list:



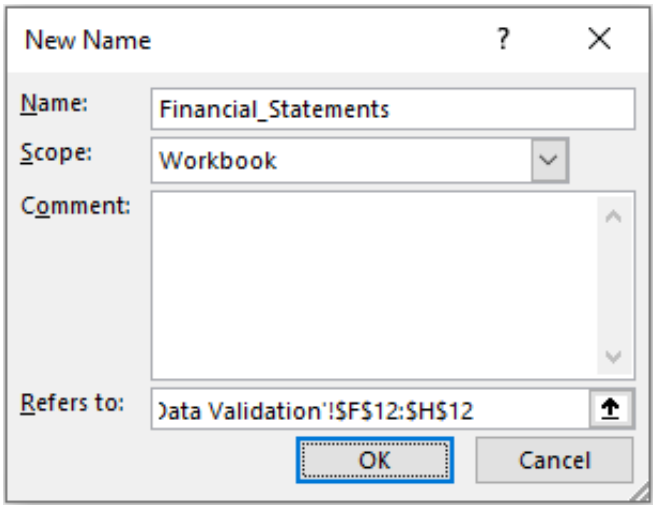
Here, in cell **G34**, I want to choose one of three financial statements, namely the Income Statement, Balance Sheet or Cash Flow Statement, and then choose a classification depending upon the selection made (*e.g.* Opex for the Income Statement, Total Assets for the Balance Sheet).

To do this, I need to set up my assumptions (in cells **F12:H23**) as follows. In row 12, I type the names of the three financial statements. However, I cannot use a space (**" "**) here (for reasons that will become apparent momentarily), so I separate the words with the underscore ("_") character instead (*e.g.* "Cash Flow Statement" becomes "Cash_Flow_Statement").

Then, I highlight cells **F12:H12** and give this range the range name **Financial_Statements** by either typing this in the 'Name Box' or else using 'Defined Name' in the Formulas tab of the Ribbon:

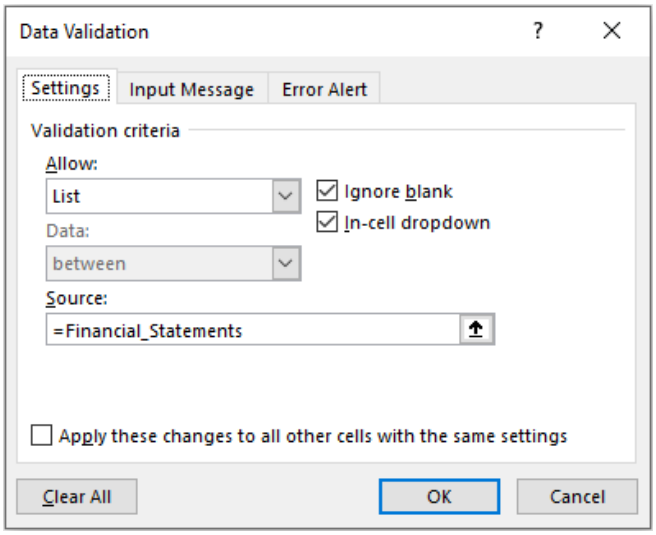


And completing the 'New Name' dialog:



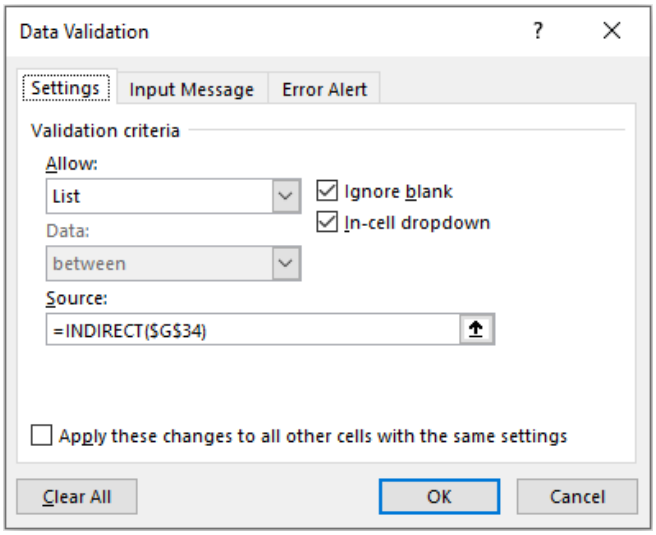
Similarly, I name the ranges **F13:F23**, **G13:G22** and **H13:H16 Income_Statement**, **Balance_Sheet** and **Cash_Flow_Statement** respectively. All we need to do now is set up the data validated lists in cells **G34** and **G35**.

For the 'Financial Statement' selector (cell **G34**), the data validation can be set as follows (**ALT + D + L**):



Here, I have referred to the source using the range name **=Financial_Statements**. Range names do not allow for spaces in the names – hence the reason for the underscored names earlier.

For the Classification selector (cell **G35**), the data validation can be set as follows (**ALT + D + L**):

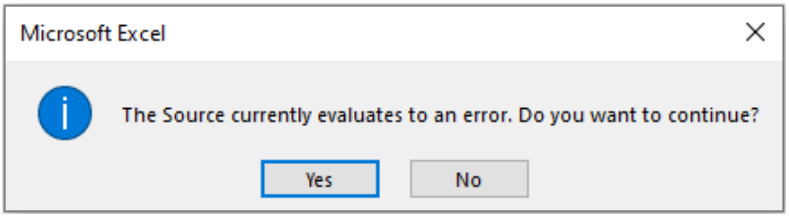


Here, in 'Source:', I have used the formula

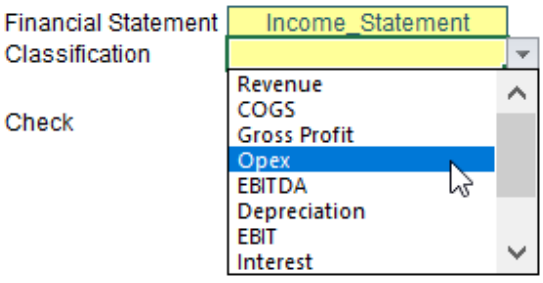
=INDIRECT(G34)

This will select the contents of cell **G34** (**Income_Statement**, **Balance_Sheet** or **Cash_Flow_Statement**), all of which have been defined as range names (hence the underscores again), and use the ranges defined to populate the list.

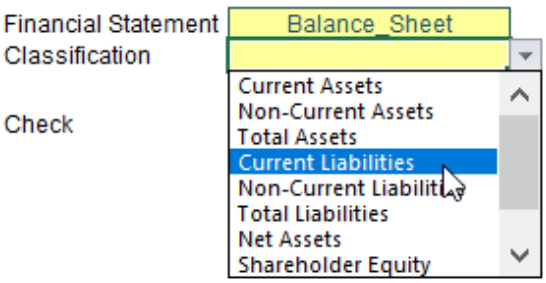
However, when you click 'OK', if the value in cell **G34** was blank when you set this data validated list up, you will encounter the following error message:



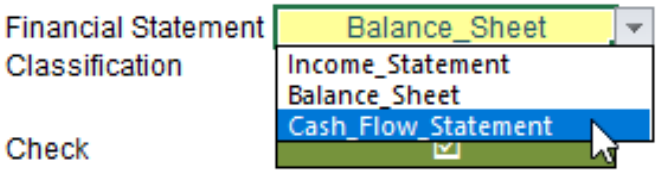
This is because Excel cannot evaluate the **INDIRECT** value of a blank cell. However, it is fine to click 'Yes' and continue. The data validated list will now work as expected, *e.g.*



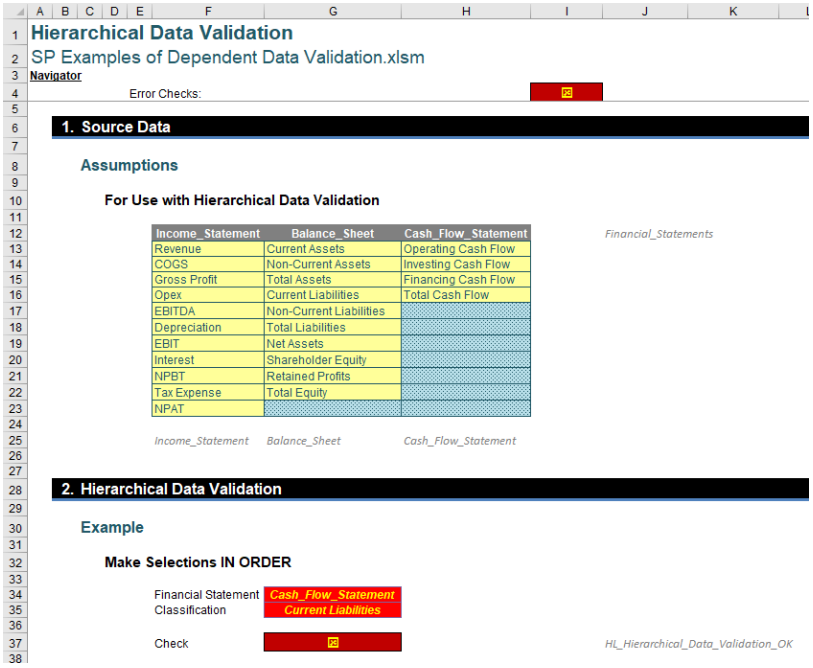
or



This type of data validated list approach is known as an "hierarchical" selection as you must select the 'Financial Statement' first. If instead I select 'Current Liabilities' in the graphic above, but then change the 'Financial Statement' to 'Cash_Flow_Statement', there is nothing to stop me:



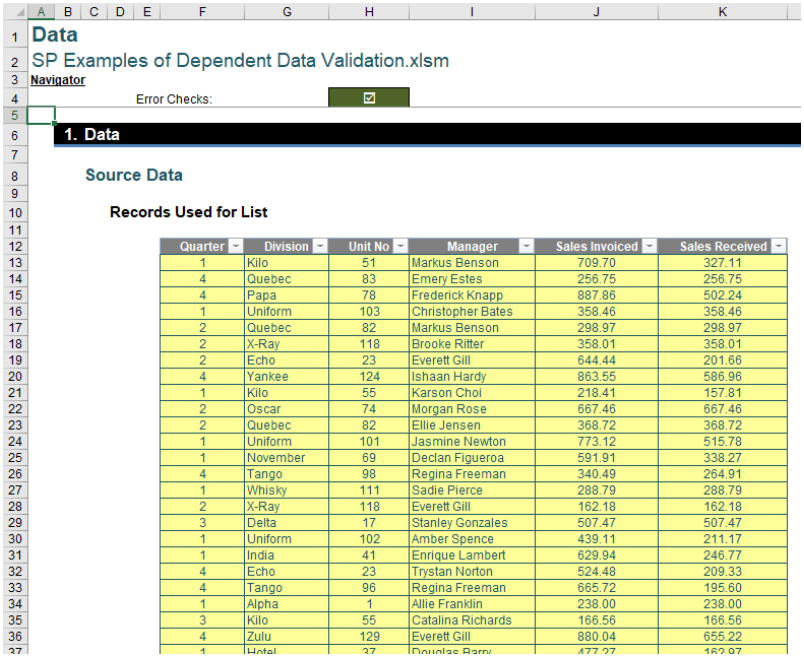
In this instance, I may wish to incorporate error checks and / or conditional formatting to highlight the problem, *viz.*



If we want to prevent this from happening, we have to get *cleverer*.

Interdependent Data Validation Lists

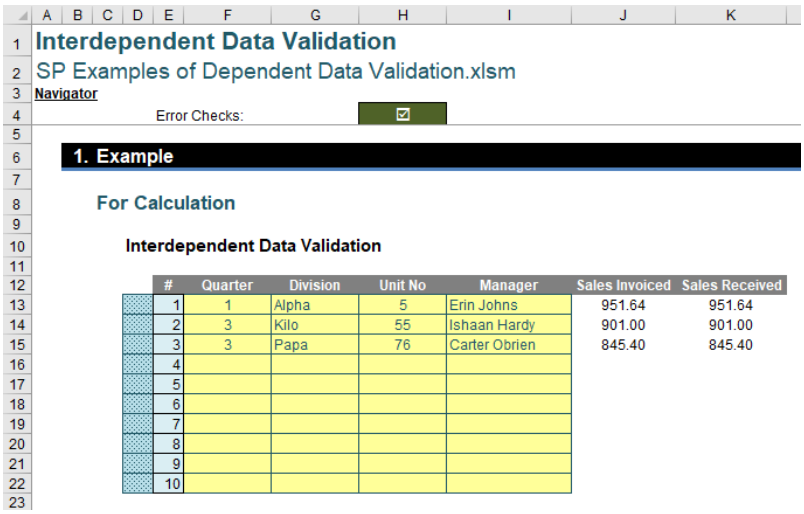
Imagine you have the following 2,000 row data table:



Quarter	Division	Unit No	Manager	Sales Invoiced	Sales Received
1	Kilo	51	Markus Benson	709.70	327.11
4	Quebec	83	Emery Estes	256.75	256.75
4	Papa	78	Frederick Knapp	887.86	502.24
1	Uniform	103	Christopher Bates	358.46	358.46
2	Quebec	82	Markus Benson	298.97	298.97
2	X-Ray	118	Brooke Ritter	358.01	358.01
2	Echo	23	Everett Gill	644.44	201.66
4	Yankee	124	Ishaan Hardy	863.55	586.96
1	Kilo	55	Karson Choi	218.41	157.81
2	Oscar	74	Morgan Rose	667.46	667.46
2	Quebec	82	Ellie Jensen	368.72	368.72
1	Uniform	101	Jasmine Newton	773.12	515.78
1	November	69	Declan Figueroa	591.91	338.27
4	Tango	98	Regina Freeman	340.49	264.91
1	Whisky	111	Sadie Pierce	288.79	288.79
2	X-Ray	118	Everett Gill	162.18	162.18
3	Delta	17	Stanley Gonzales	507.47	507.47
1	Uniform	102	Amber Spence	439.11	211.17
1	India	41	Enrique Lambert	629.94	246.77
4	Echo	23	Trystan Norton	524.48	209.33
4	Tango	96	Regina Freeman	665.72	195.60
1	Alpha	1	Allie Franklin	238.00	238.00
3	Kilo	55	Catalina Richards	166.56	166.56
4	Zulu	129	Everett Gill	880.04	655.22
1	Hotel	37	Douglas Barry	477.27	162.97

This source table has been converted to an Excel Table (using the keyboard shortcut **CTRL + T** or else **Insert -> Table**), so that the Table may automatically extend, both in numbers of rows and / or columns. This table has cunningly been called **Data**.

On a separate worksheet, I wish to summarise the total sales invoiced and received by being able to make selections regarding the **Quarter**, **Division**, **Unit No** and **Manager**.



Interdependent Data Validation

SP Examples of Dependent Data Validation.xlsm

Navigator

Error Checks: ☑

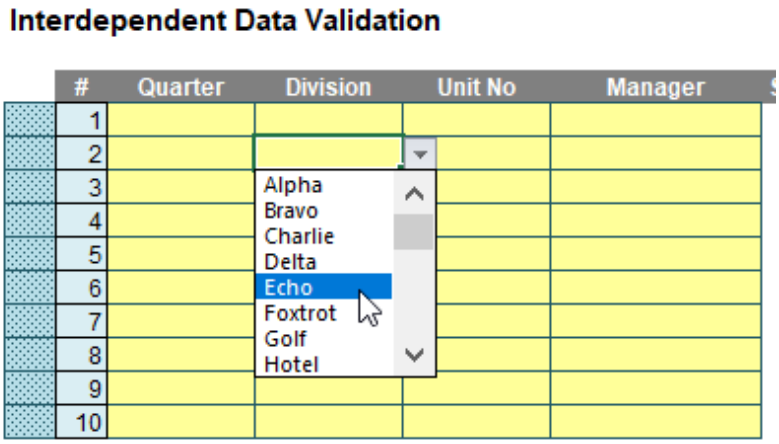
1. Example

For Calculation

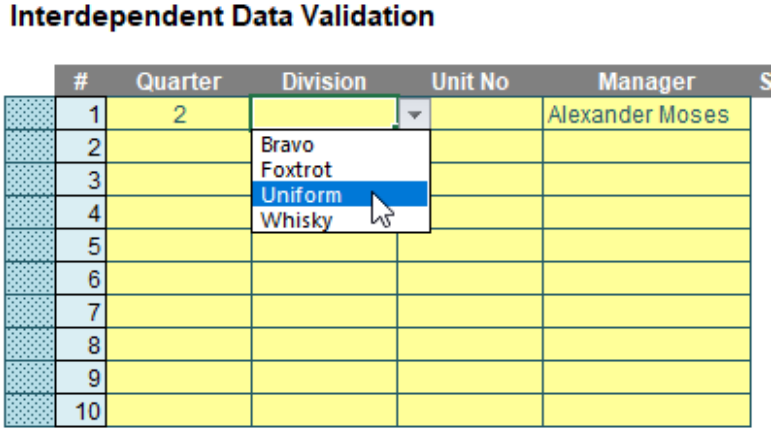
Interdependent Data Validation

#	Quarter	Division	Unit No	Manager	Sales Invoiced	Sales Received
1	1	Alpha	5	Erin Johns	951.64	951.64
2	3	Kilo	55	Ishaan Hardy	901.00	901.00
3	3	Papa	76	Carter Obrien	845.40	845.40
4						
5						
6						
7						
8						
9						
10						

Here, each assumption field (*i.e.* yellow cell) contains the appropriate data validation to make a selection:



However, it is *cleverer* than that. If I make selections for several fields (no matter which I choose, the order is irrelevant), my data validation only allows me to choose from options which exist in the **Data** Table, *e.g.*



or

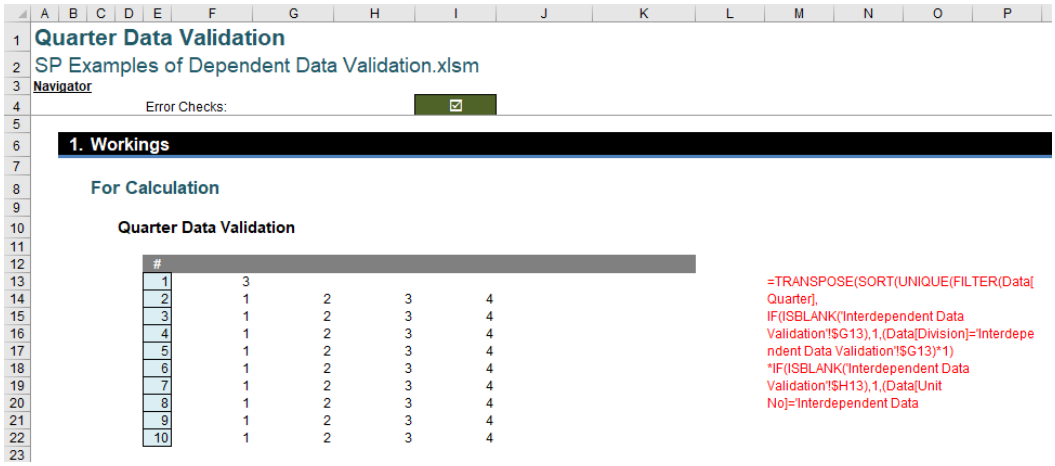
Interdependent Data Validation

	#	Quarter	Division	Unit No	Manager	Sale
	1	3	Charlie	13		
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					

Benjamin Bell
Dixie Stanton
Heidy White
Tomas Suarez

These are what is known as **interdependent data validation lists**. So how do we construct them? Well, if you don't use Excel 365, you might not like the answer…

I have created four similar workings sheets – one each for the four fields that may be selected, *i.e.* **Quarter**, **Division**, **Unit No** and **Manager**. For example, here if the 'Quarter Data Validation' workings worksheet:



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

Cell **F13** contains the following wonderful formula:

=TRANSPOSE(SORT(UNIQUE(FILTER(Data[Quarter],

IF(ISBLANK('Interdependent Data Validation'!$G13),1,(Data[Division]='Interdependent Data Validation'!$G13)*1)

***IF(ISBLANK('Interdependent Data Validation'!$H13),1,(Data[Unit No]='Interdependent Data Validation'!$H13)*1)**

***IF(ISBLANK('Interdependent Data Validation'!$I13),1,(Data[Manager]='Interdependent Data Validation'!$I13)*1)))))**

Clear, yes..?

Er, no.

Allow me to break this monster calculation up. It's not as bad as it may look on first glance. As always with larger formula, always start in the middle and work outwards. Let me first look at the segment

IF(ISBLANK('Interdependent Data Validation'!$G13),1,(Data[Division]='Interdependent Data Validation'!$G13)*1)

This formula inspects cell **G13** on the 'Interdependent Data Validation' worksheet, which is the assumption for the **Division** on the first row of the summary table. The calculation determines whether the cell is blank using **ISBLANK**:

- if it is, the entire formula returns the value one [1]
- if it isn't, it reviews the entire **Division** column of the **Data** table and creates an array which is a one [1] if the **Division** is the same value as in cell **G13** and zero [0] otherwise.

The segments

IF(ISBLANK('Interdependent Data Validation'!$H13),1,(Data[Unit No]='Interdependent Data Validation'!$H13)*1)

and

IF(ISBLANK('Interdependent Data Validation'!$I13),1,(Data[Manager]='Interdependent Data Validation'!$I13)*1)

undertake similar operations for cells **H13** and **I13** for the fields **Unit No** and **Manager** in the **Data** Table respectively.

Multiplying these three segments together results in a column vector, with a one [1] when all three elements have values in the corresponding record (horizontal row) of the **Data** Table that equal the values in row 13 of the summary table (unless blank, where all elements for that field remain included) and zero [0] otherwise.

The **FILTER** function then uses this calculated vector as the filter, *i.e.*

FILTER(Data[Quarter], Calculated_Vector)

FILTER is an Excel 365 function, known as a **dynamic array function**, that filters the field **Quarter** in the **Data** Table (the one [1] values act as the filter), so that only elements are returned where data is held for the selections of **Division**, **Unit No** and **Manager** made. This is why **Quarter** is not one of the segments in the calculation. The other fields use it to sift out all the impossible selections for **Quarter**.

Then, **UNIQUE**, another Excel 365 dynamic array function, is used so that duplicates are removed:

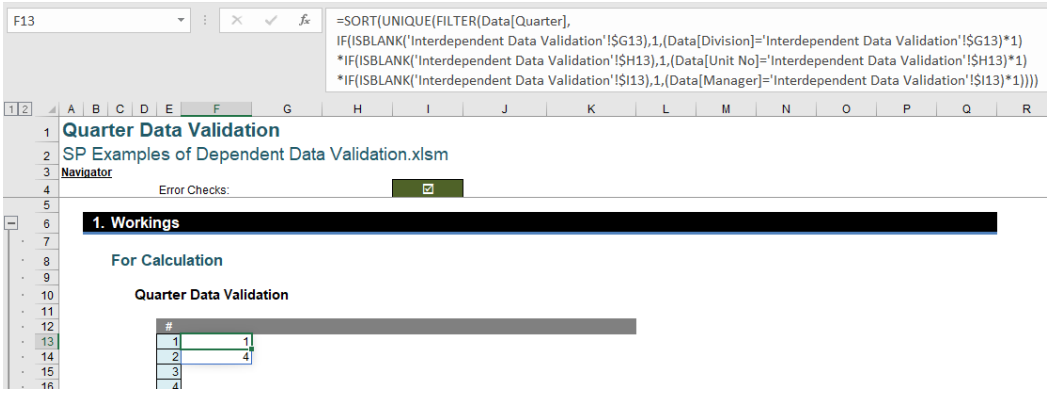
UNIQUE(Filtered_Results)

Next up is **SORT**, another Excel 365 dynamic array function, which orders the resulting list alphabetically in ascending order (*i.e.* from "A" to "Z"):

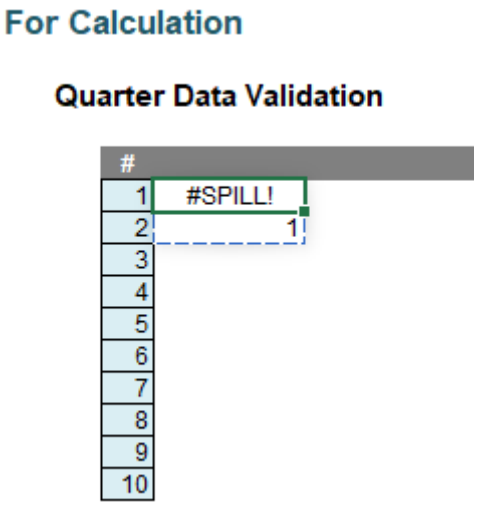
SORT(Unique_Filtered_Results)

It's always recommended that you remove duplicates first, as you then have less to sort, which helps with memory usage, calculation times, *etc.*

At this stage, for just the one formula we might get a result as follows:



The results *spill* down the column. This is what dynamic array functions do. They produce results in the form of an array which may vary in size – hence the name. This is a problem, because I will require a similar formula in the row beneath, *i.e.* cell **F14** for the second record in the summary table. However, if I enter a similar calculation in this cell, I will be met with the following *prima facie* error:



I need the result to propagate across the row, not down the column – and this is why the formula is wrapped in a **TRANSPOSE** function – which forces that to happen:

=TRANSPOSE(SORT(UNIQUE(FILTER(Data[Quarter],

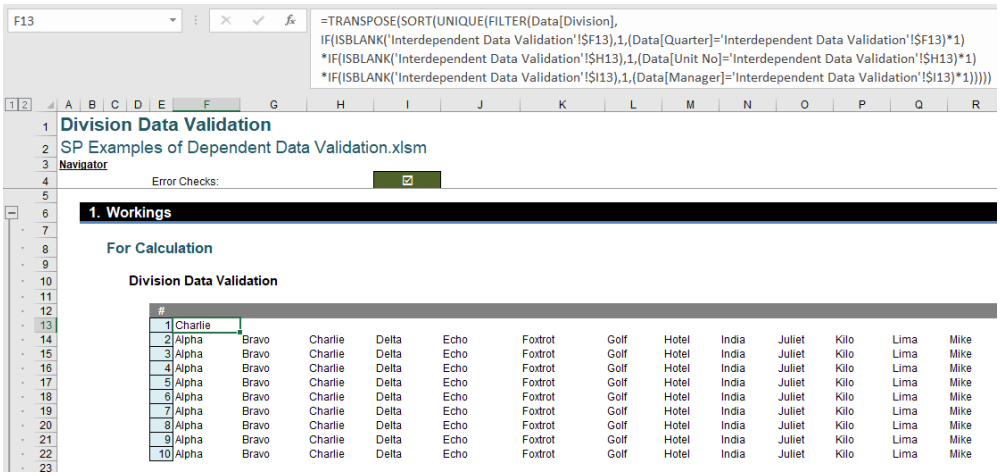
IF(ISBLANK('Interdependent Data Validation'!$G13),1,(Data[Division]='Interdependent Data Validation'!$G13)*1)

***IF(ISBLANK('Interdependent Data Validation'!$H13),1,(Data[Unit No]='Interdependent Data Validation'!$H13)*1)**

***IF(ISBLANK('Interdependent Data Validation'!$I13),1,(Data[Manager]='Interdependent Data Validation'!$I13)*1)))))**

Simple. Your PhD in Financial Modelling awaits!

This is why each field has its own workings sheet. We need similar formula for each of the four fields. For example, the 'Division Data Validation' worksheet may look as follows:



The formula is eerily similar to the one above:

=TRANSPOSE(SORT(UNIQUE(FILTER(Data[Division],

IF(ISBLANK('Interdependent Data Validation'!$F13),1,(Data[Quarter]='Interdependent Data Validation'!$F13)*1)

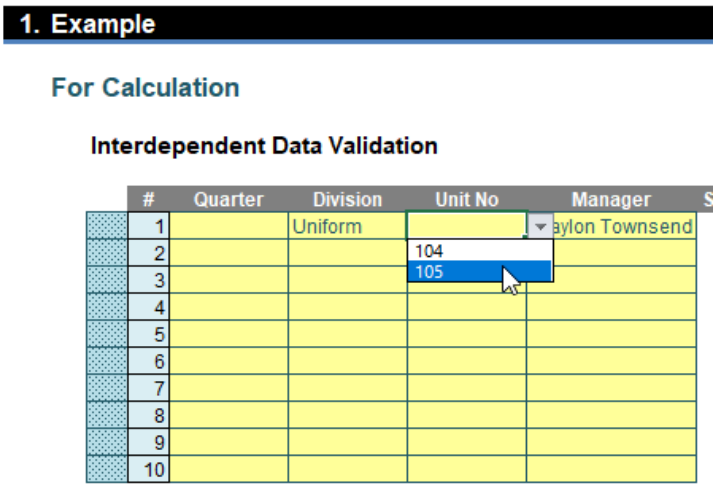
***IF(ISBLANK('Interdependent Data Validation'!$H13),1,(Data[Unit No]='Interdependent Data Validation'!$H13)*1)**

***IF(ISBLANK('Interdependent Data Validation'!$I13),1,(Data[Manager]='Interdependent Data Validation'!$I13)*1)))))**

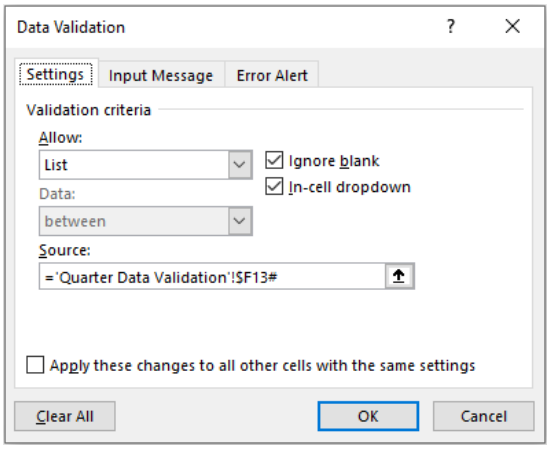
On this occasion, the **Division** field in the **Data** Table is filtered, with one of the segments now being **Quarter** (instead of **Division**).

Hence, if you require nine data validated fields you will need nine working sheets, *etc.*

All that is left to do now is create the data validation lists back on the 'Interdependent Data Validation' worksheet.



For example, the data validation list for the first record's **Quarter** (cell **F13**) is created as follows (**ALT + D + L**):



The 'Source:' here is given by the formula

='Quarter Data Validation'!$F13#

This uses the list created in cell **F13** (*i.e.* the first record's calculation) of the 'Quarter Data Validation' workings worksheet. The symbol "**#**" at the end of the formulaic reference forces Excel to refer to the entire spilled range, so that as the range expands / contracts so does the list accordingly.

Similar data validated lists may be created for **Division**, **Unit No** and **Manager** too. The result is four data validated lists that are all interconnected (*i.e.* they are "interdependent"), which allow the end user only to make selections that exist in the **Data** Table – thus avoiding the issue raised in the simpler hierarchical approach.

The Sales Invoiced and Sales Received may then be calculated simply using **SUMIFS**:

Interdependent Data Validation

#	Quarter	Division	Unit No	Manager	Sales Invoiced	Sales Received
1	3	Uniform	105	Braylon Townsend	925.73	925.73

The formula for **Sales Invoiced** is given by

=SUMIFS(Data[Sales Invoiced],Data[Quarter],$F13,Data[Division],$G13,Data[Unit No],$H13,Data[Manager],$I13)

whereas the formula for **Sales Received** is given by

=SUMIFS(Data[Sales Received],Data[Quarter],$F13,Data[Division],$G13,Data[Unit No],$H13,Data[Manager],$I13)

Easy when you know how!

Word to the Wise

For just one record for interdependent selections, it may be simpler to use Slicers connected to the **Data** Table. However, this will not work if you require a table summarising multiple selections, as in our example. The trick is always to keep things as simple as possible!

Also, it has been made clear above that I have used the dynamic array features in Excel 365 to create an interdependent data validation lists solution. That's because I am hoping slowly people are moving over to Excel 365 as Microsoft add more and more useful features – and because it is much simpler than trying to construct a solution in other versions of Excel. In other editions, this may require functions such as **AGGREGATE, COUNT**, **INDEX**, **OFFSET** and **TRANSPOSE**, plus the use of range names. This requires a lot of guile, brute force and imagination – it's just *simpler* in Excel 365. If you're not already using it, maybe it's time to take the plunge and make the switch…

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…*

Excel possess a challenge when it comes to categorical data. To identify commonality in different values within different categories can somewhat be difficult on occasion. For instance, imagine the following dataset that shows which social media channels certain people use:

	A	B	C	D	E
1	Category	Value			
2	LinkedIn	Tim		A	Hanh
3	LinkedIn	Jonathan		B	Jonathan
4	LinkedIn	Hanh			
5	LinkedIn	Nakul			
6	Discord	Tim			
7	Discord	Jonathan			
8	Discord	Nakul			
9	Instagram	Nakul			
10	Instagram	Jonathan			
11	Instagram	Kathryn			
12	Facebook	Kathryn			
13	Facebook	Liam			
14					

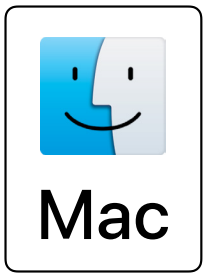
Imagine that this data set goes on to for 1,000 rows. Using this data set as a proxy, how hard it is to identify if a pair of users have a common social media channel?

The challenge is "simply" this: can you write a formula to ascertain whether a specified pair use a common social media channel. This must be done in one cell.

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

Excel for Mac

With Steve Kraynak joining the team, we thought we would exploit his knowledge and recant all about Microsoft Excel for Mac. Each month, we'll cover a different topic to help you understand how Excel for Mac is different than Excel for Windows. This month, we'll point out some important features that aren't available Excel for Mac, which should help you decide whether a Mac is right for you.

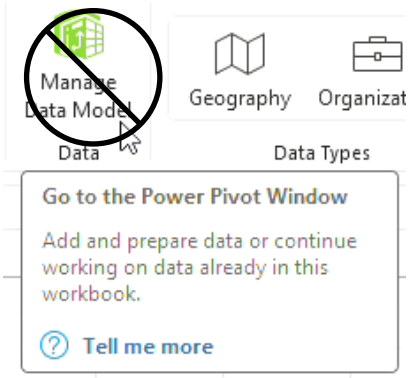


Buying a computer and deciding whether to get a Mac or a PC is a big decision, especially if you use Excel. In recent years, more and more companies offer the choice to their employees, but we've heard that some specifically consider what type of work the employee will need to. Prior to 2020, the decision was fairly clear. There were just too many deficiencies in Excel for Mac, so if you were going to use Excel to any great extent, you would need to choose a PC.

The good news is that Microsoft has been steadily reducing those deficiencies by adding more and more features in their monthly updates. However, there are still a few important differences that you should consider.

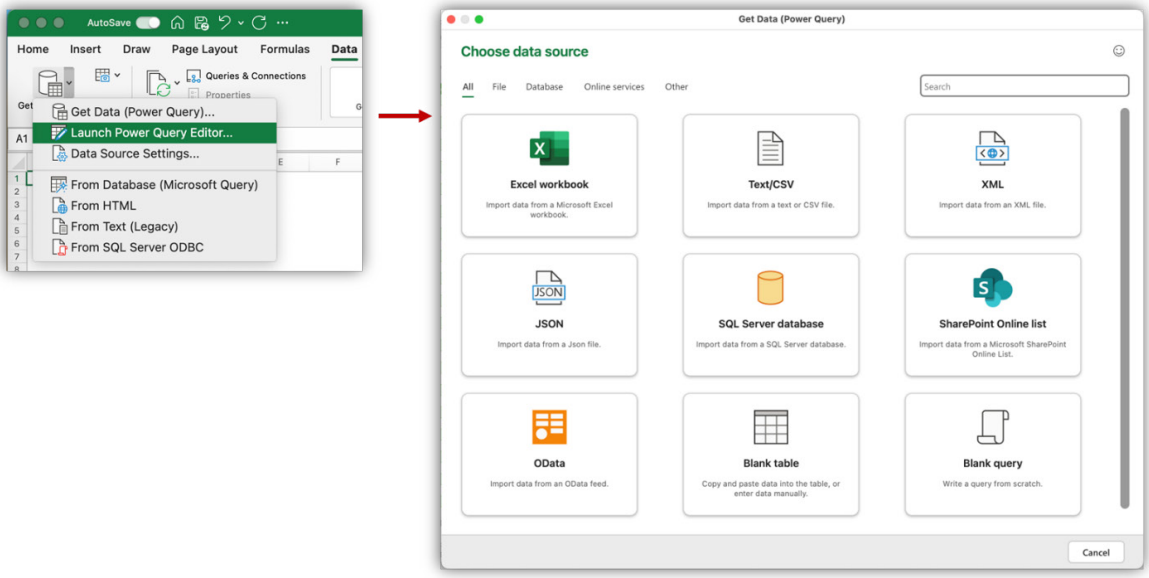
PowerPivot / Data Model

If you need to use Power Pivot (Data Model) in Excel, then you should choose a PC. Power Pivot is not available on the Mac and it likely won't be available, since it relies on some operating system API's that only exist on Windows.



Power Query

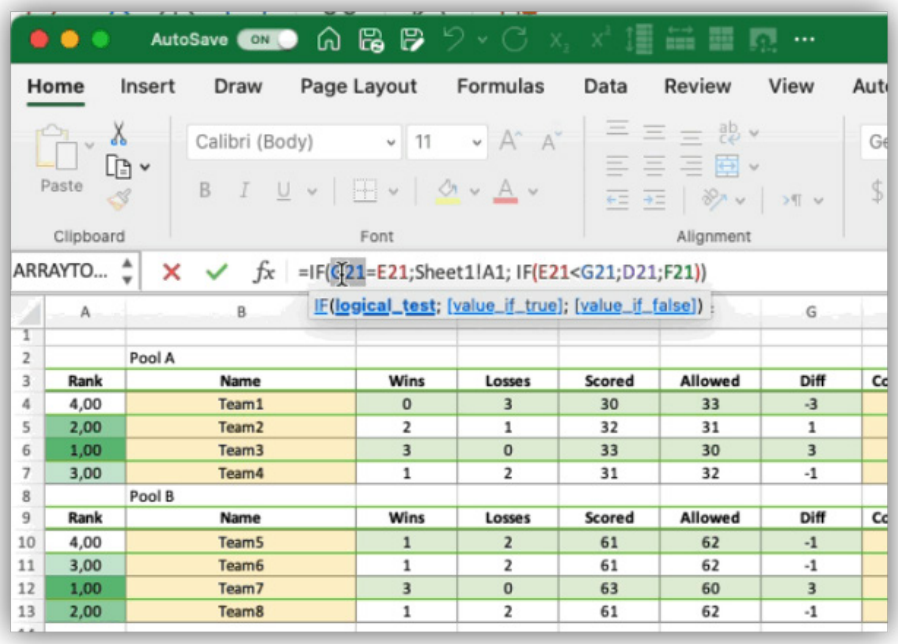
Before last year (2023 if you haven't adjusted to the new year!), if you needed Power Query, then a Mac would not be a good choice. Thankfully, Microsoft has added most of the Power Query capability to Excel for Mac, so it's less likely to be important when deciding whether to buy a Mac. As of late 2023, the most popular data sources are available, but there are still some missing. Based upon how Power Query has been progressing, it is likely that more and more data sources will be made available in monthly updates. If you use Power Query, you might want to verify that the data sources you need are available before switching to a Mac.



There are other features not available on Mac, but hopefully they're not as critical as Power Pivot and Power Query. Most of the popular features are available on Mac, so hopefully the ones mentioned here won't impact your decision. This list is not meant to be comprehensive, but hits on some of the highlights.

Features not available on Mac:

1. Evaluate Formula: although this feature is missing, Microsoft introduced the Value Preview ToolTips to help you debug your formulae. It's a nice alternative to 'Evaluate Formula'



2. **ENCODEURL()**, **FILTERXML()** and **WEBSERVICE()**: these functions rely on operating system capability on Windows, so they won't work on a Mac
3. editing UserForms in the VBA editor. If you're creating UserForms, you'll need to use a PC. They can be used on a Mac, but you can't create or edit them
4. publish to Power BI
5. Spreadsheet Inquire and Spreadsheet Compare are missing (COM add-ins don't work on a Mac)
6. PivotTable Wizard / Consolidate Ranges
7. XML Map pane / XML Tools
8. OLE: you can't insert PDF as an icon, and double-clicking on one will not open the PDF document
9. ActiveX: this is a Windows technology, so it will not be available on a Mac. Use Form Controls instead because they work on both Mac and PC.

We'll continue next month…

Charts and Dashboards

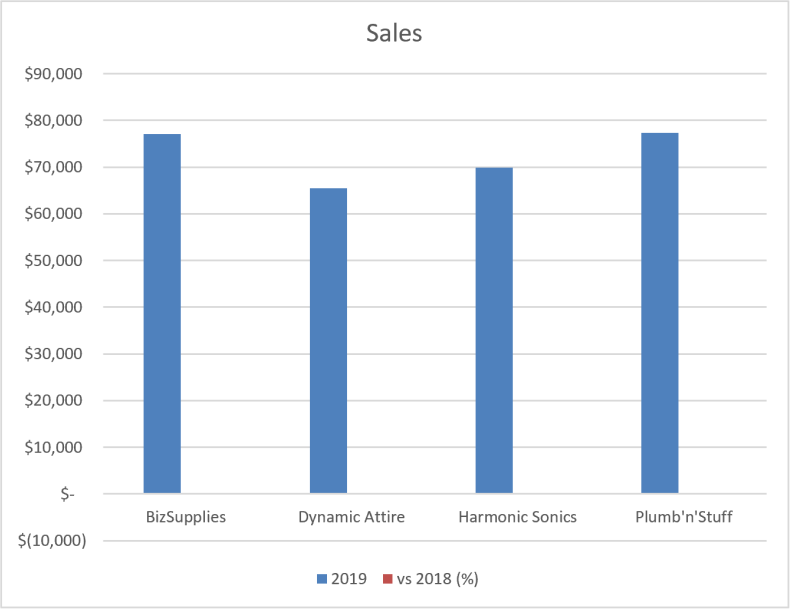
It's time to chart our progress with an introductory series into the world of creating charts and dashboards in Excel. This month, we continue our consideration of Symbol Charts in dashboards.

Last month, we introduced symbols and how to add such icons to a dashboard table:

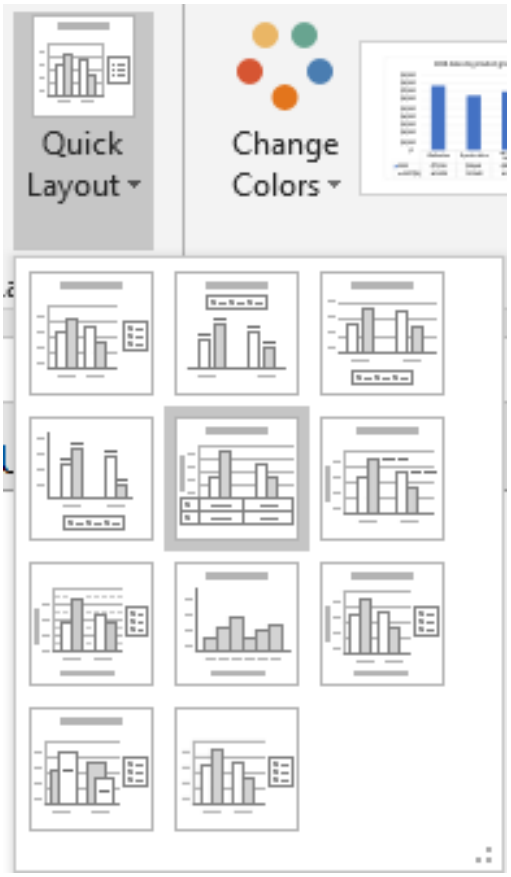
Year	2018	2019	vs 2018 (%)
BizSupplies	$ 76,813	$ 77,034	▲0.29%
Dynamic Attire	$ 65,608	$ 65,449	▼0.24%
Harmonic Sonics	$ 68,089	$ 69,948	▲2.73%
Plumb'n'Stuff	$ 76,792	$ 77,321	▲0.69%

Now, we will see how to create a chart from this data with the symbols attached.

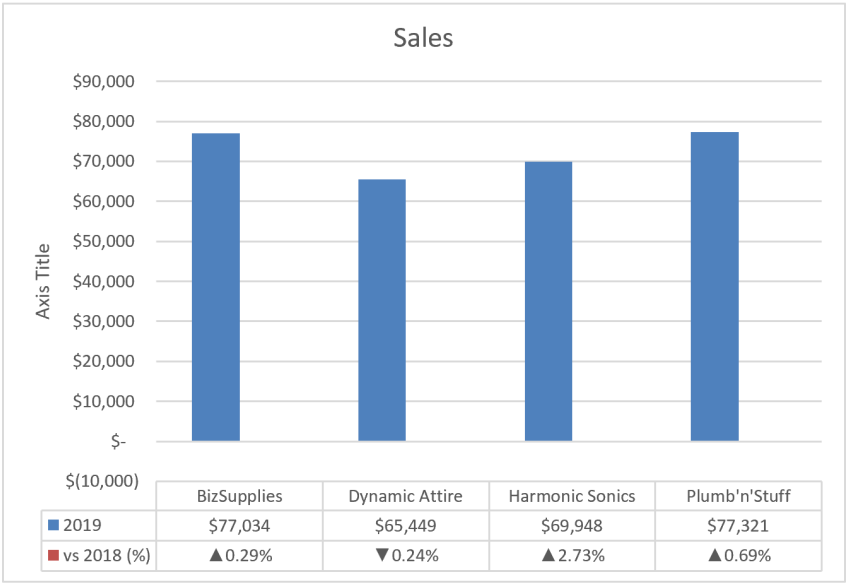
First, we choose the data area and select the Column chart. The chart initially appears like the one below. There are no icons shown here, while the '**vs 2018 (%)**' series appears to be unnecessary in the chart (because they are too small to see!).



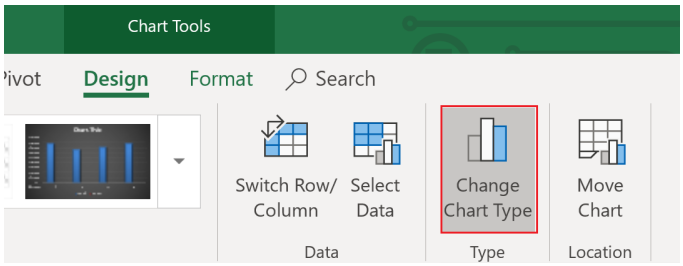
We can add the data table via **Chart Tools - > Design - > Quick Layout** to include the table with icons:



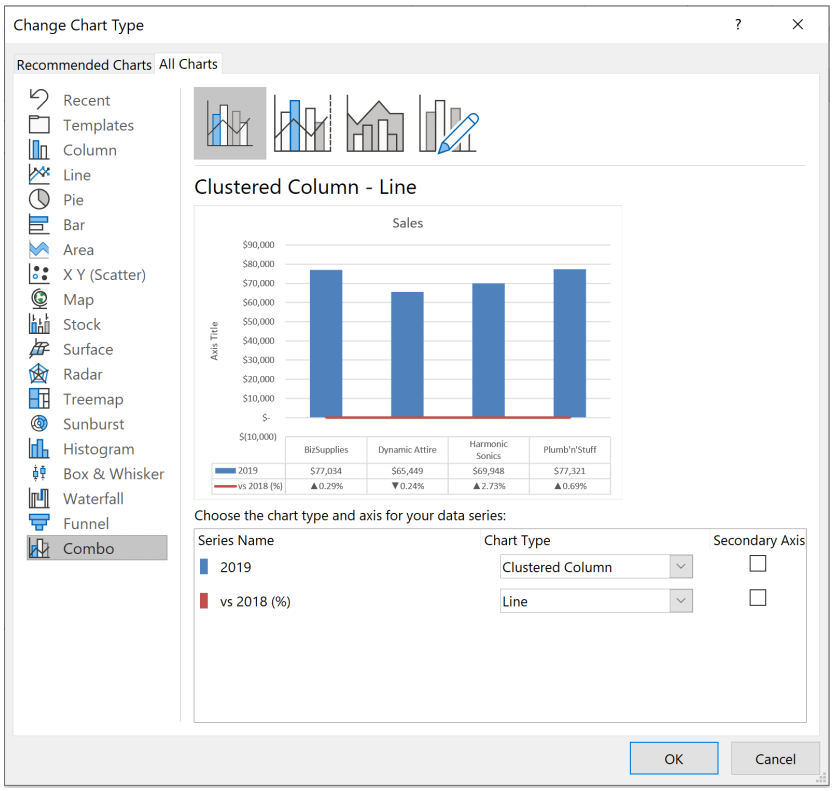
The chart now appears as below. The series '**vs 2018 (%)**' is still there, but it is still just too small to see, *viz.*



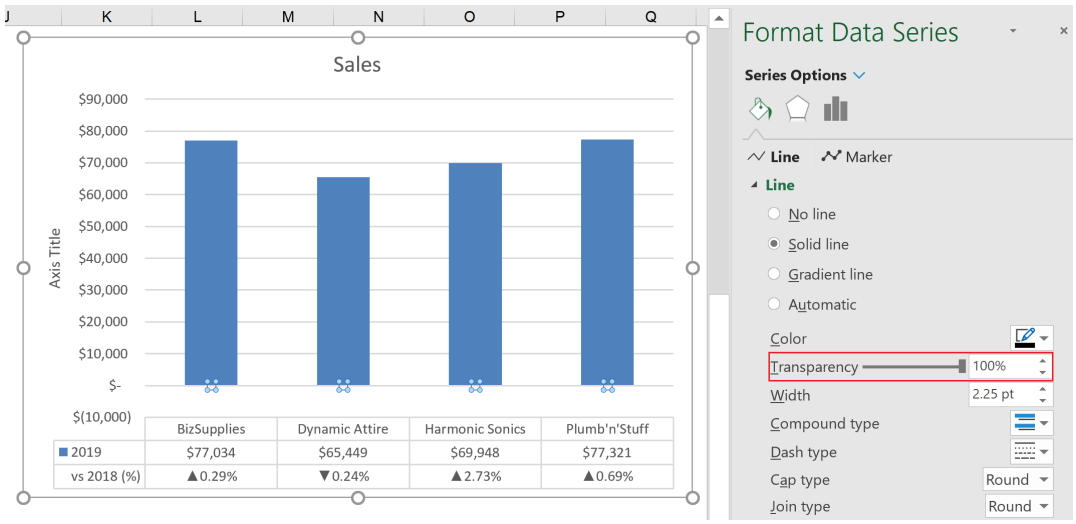
To exclude the '**vs 2018 (%)**' series from the chart while still keeping the data in the below-chart table, we can go to **Chart Tools - > Design - > Change Chart Type**:



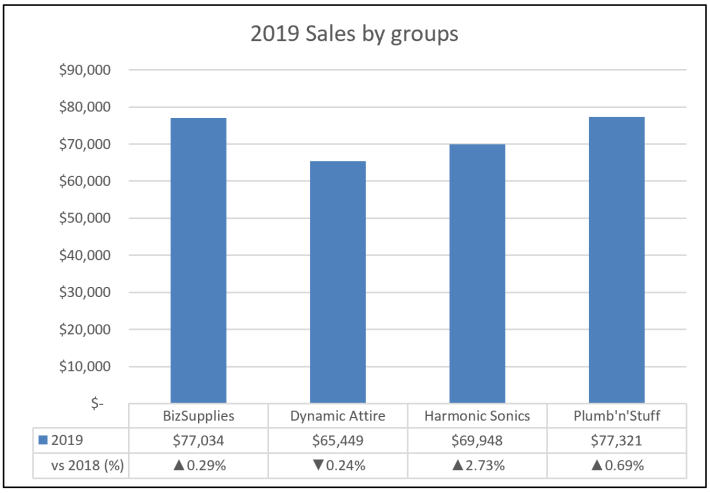
In the pop-up window's 'All Charts' tab, choose the last item, Combo. We set the chart type of '**vs 2018 (%)**' to be a Line, and finally, we can now see a line on the chart. Click OK.



To make it invisible on a chart, click on the line and go to 'Format Data Series'. In the Line section, set the Color *(sic)* to White and Transparency to 100%. The line should now be invisible:

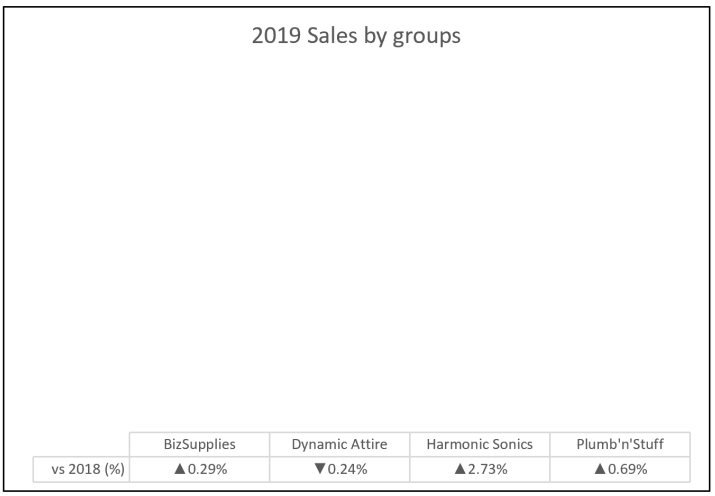


We may now modify the chart title, remove the axis title and set the vertical axis to minimum of zero [0] (since the '**vs 2018 (%)**' series includes a negative number, the initial chart is shown to $(10,000)). Then we may change the chart design as we wish. The chart is now showing both values and icons, making it easier to understand the data.

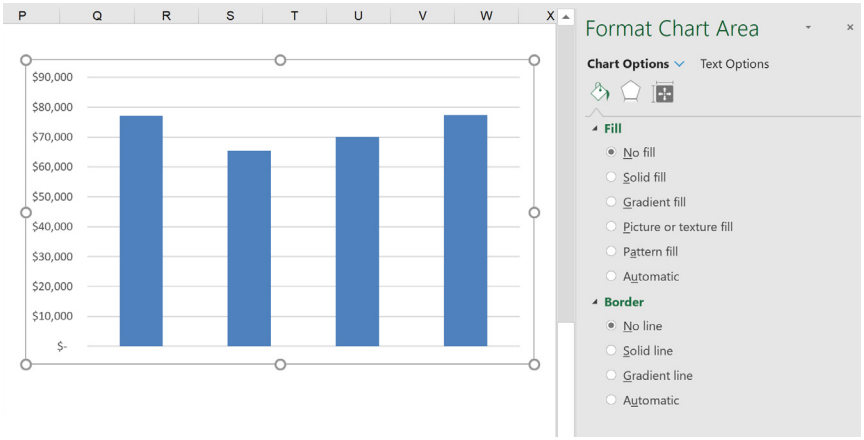


There is a trick that we can employ to remove the '**2019**' row from the associated data table. That is, we want to see the sales columns only and how sales increase / decrease compared to the year 2018.

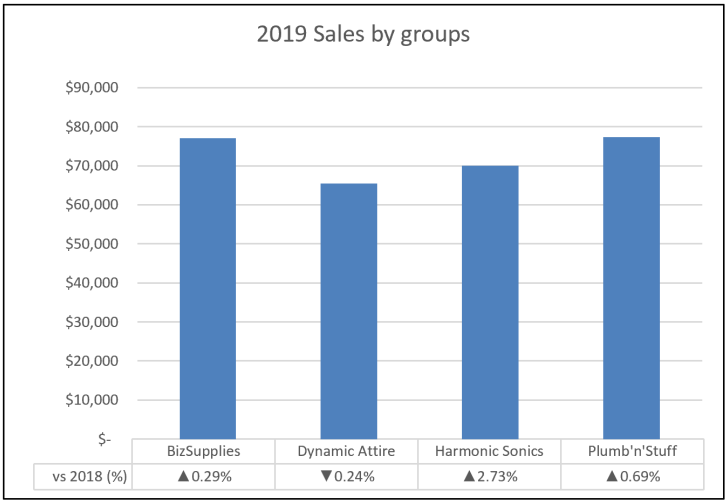
To do this, we duplicate the chart. In one of them, we remove the '**2019**' series and its axis. The chart is now empty, as illustrated below:



In the other chart, we remove the '**vs 2018 (%)**' series, as well as the horizontal axis, chart title and data tables. We also need to format the Fill and Border to be 'No fill' and 'No line' respectively. The chart should now look like this:



We now drag the column chart to fit on the empty chart with the data table, so that the columns and legend match. We now have our desired chart:



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 look at running a macro.

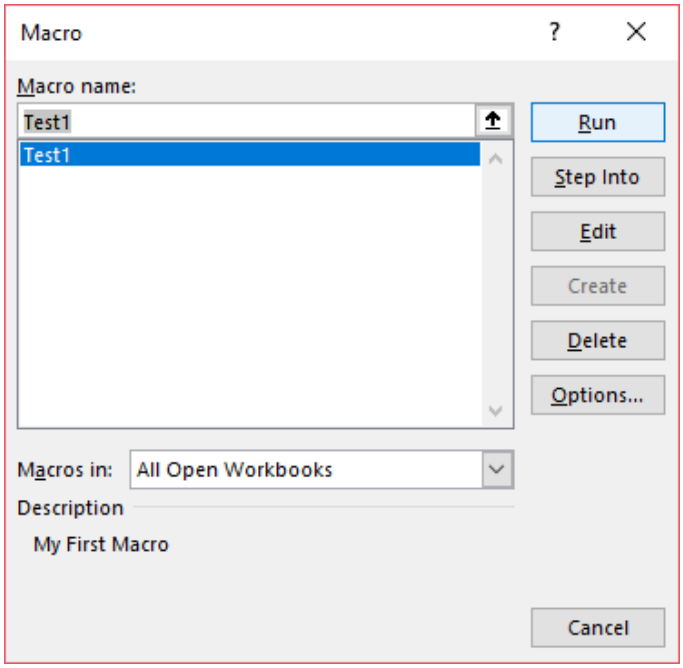
There are a few ways to run a macro:

1. From the 'Macro' dialog box
2. Running it from the VBA Editor
3. Using a predefined keyboard shortcut
4. Anchoring the macro to a form control
5. Setting it to automatically run on a specific event.

Let's consider each of these scenarios.

1. From the 'Macro' dialog box

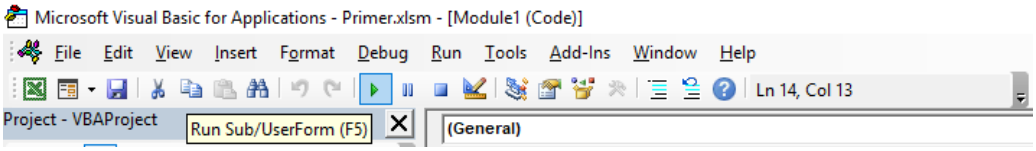
Select the intended macro from the list in the dialog (**ALT + L + PM**) and hit the 'Run' button:



It's not rocket science.

2. Running it from the VBA Editor

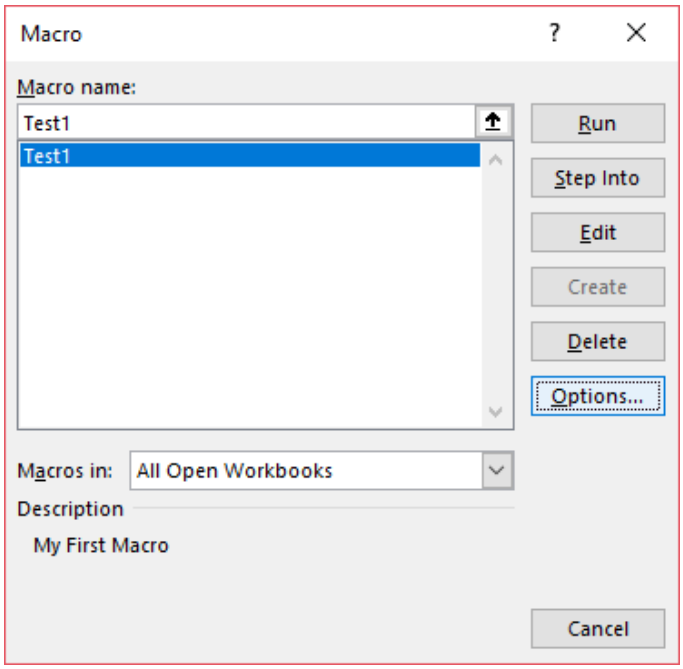
In the VBA Editor (**ALT + F11**), there is a tool bar at the top:



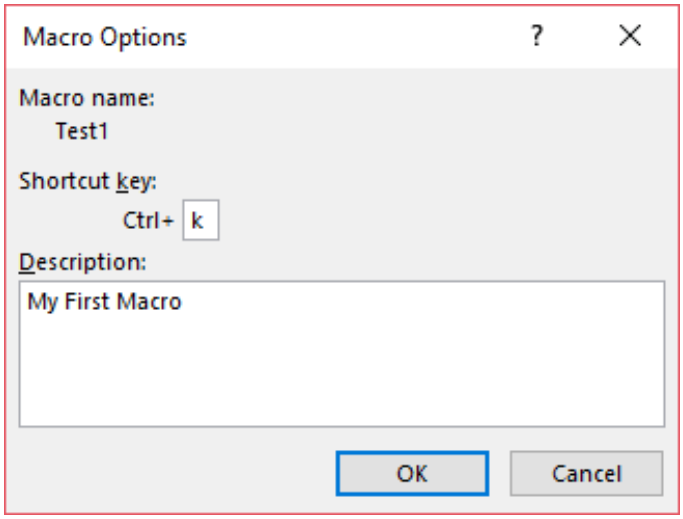
The 'Run Sub / UserForm (Shortcut: **F5**)' button looks similar to the 'Play' button on most electronic devices. Ensure the cursor is in the Code Window *within the procedure* to be run. If the cursor is not within any procedure, then the 'Macro' dialog box will pop up to prompt to select one to run.

3. Using a predefined shortcut

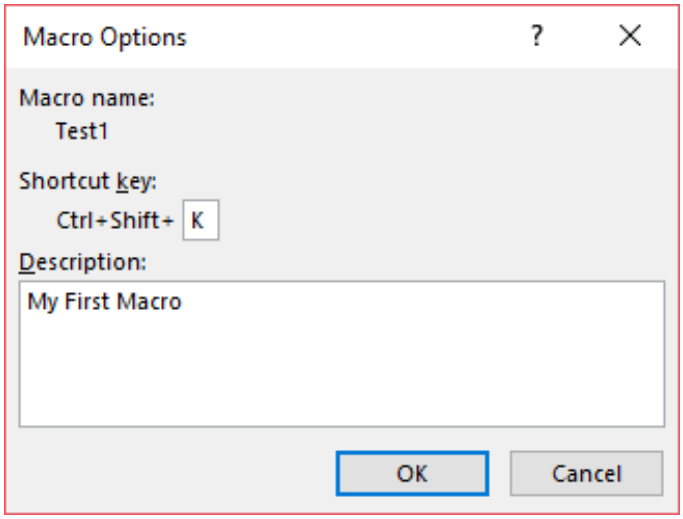
Initially when recording the macro, the option to create a shortcut key is presented. This can be set after the fact by clicking on the 'Options…' button on the 'Macro' dialog box:



The 'Macro Options' dialog box will appear:



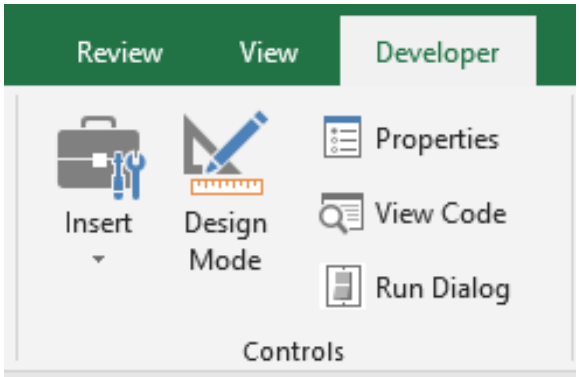
Any alphanumeric key can be used to create shortcut key. **SHIFT** can also be combined as well, by holding shift in conjunction with the key.



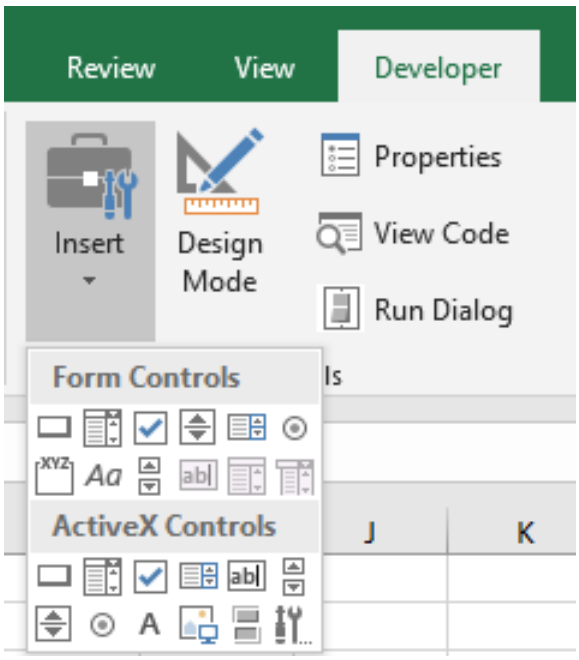
However, macro shortcut keys take precedence over Excel. This means that if an action is assigned to a keyboard shortcut that is already assigned to Excel, this keyboard shortcut replaces the Access key assignment. For example, **CTRL + C** is the keyboard shortcut for the 'Copy' command; if we assign this keyboard shortcut to a macro, Excel will run the macro instead of the 'Copy' command.

4. Anchoring the macro to a form control

In the 'Developer' tab of the Ribbon, there is the 'Controls' category:



Clicking on ‘Insert’, a drop down will appear of different things that can be included to create UserForms.

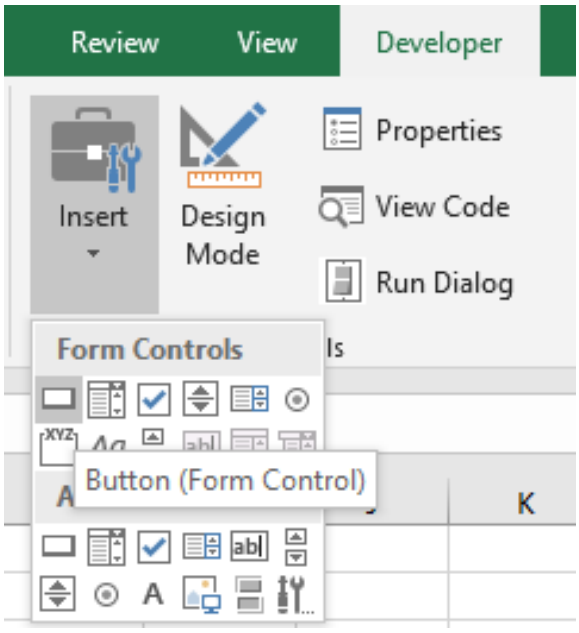


There are two types of controls: Form Controls and ActiveX controls:

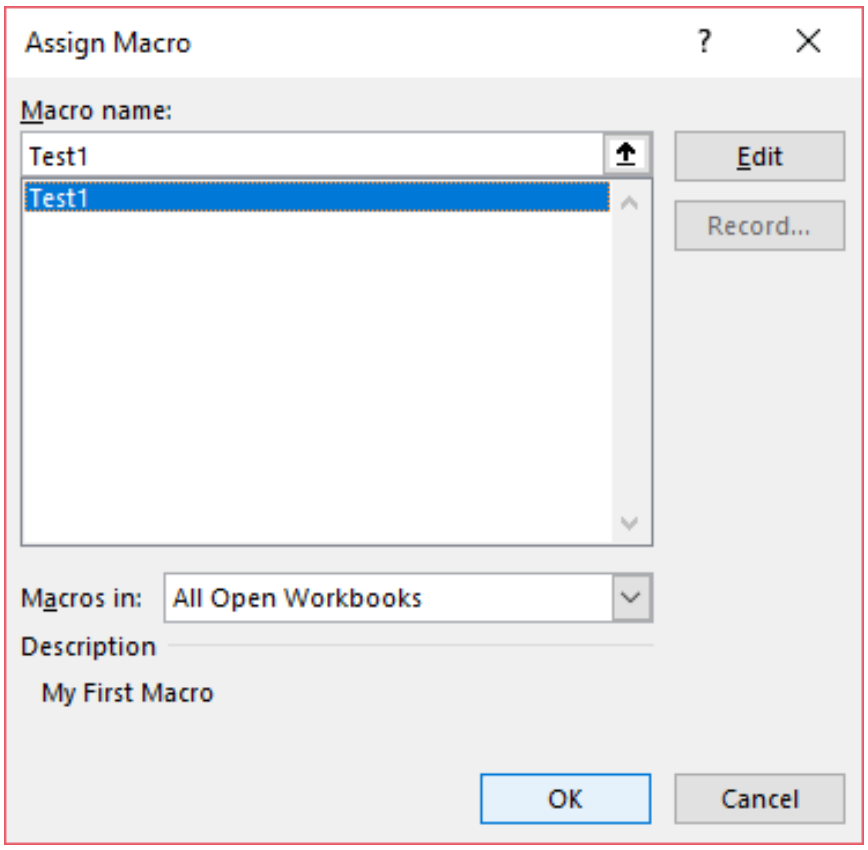
- **Form controls** are built into Excel whereas ActiveX controls are loaded separately
- **ActiveX controls** allow for more flexible design and should be used when the job just can't be done with a basic Forms control.

However, many users’ computers won't trust ActiveX by default, therefore ActiveX controls are usually disabled and may be required to be manually added to the Trust Center. One more thing to note is that ActiveX is a Microsoft-based technology and is not supported on the Mac. However, this may change very soon.

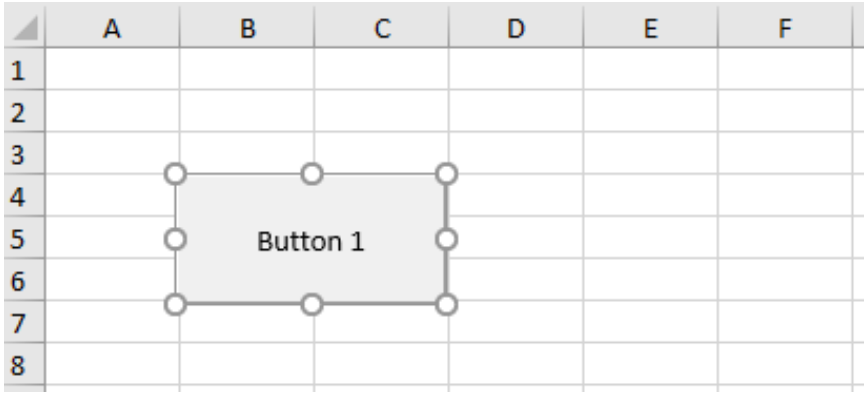
Bearing all this in mind, click on the first ‘Form Control’ button:



The cursor will turn into a cross and draw a rectangle with like when inserting shapes. An ‘Assign Macro’ dialog will appear. It will give the option for creating a “Button1_Click”, but any previously written macro can be selected.



A button will appear on the worksheet ready for use.



5. Setting it to automatically run on a specific event

Ah, that's a little more sophisticated – and we'll return to that another time (we have to keep you on tenterhooks somehow…).

Power Pivot Principles

*We continue our series on the Excel COM add-in, Power Pivot. This month, we look at the **CONCATENATE** function.*

The **CONCATENATE** function takes two text strings and joins them into one text string. It uses the following syntax to operate:

CONCATENATE(text1, text2)

Simple?

Let's move on to an example. Assuming we have the following Table with product descriptions split up into three columns arranged in a "realistic" but poor way, *viz.*

	A	B	C	D
1				
2		Column1	Column2	Column3
3		Casual	Slippers	
4		Business	Shoes	
5		Heavy	Duty	Gloves
6		Heavy	Duty	Shoes
7		Shoes	Heavy	Duty
8		Gloves	(Light)	
9		Gloves	(Heavy Duty)	
10		Shoes	Business	
11		Leather	Belt	
12				

We want to combine the data into a product description column of some sort, such as:

Product Type
Casual Slippers
Business Shoes
Heavy Duty Gloves
Heavy Duty Shoes
Shoes Heavy Duty
Gloves (Light)
Gloves (Heavy Duty)
Shoes Business
Leather Belt

Using the **CONCATENATE** function in Power Pivot:

```
=CONCATENATE(
        'Concatenate'[Column1],
                CONCATENATE(
                        'Concatenate'[Column2],'Concatenate'[Column3]
                        )
                )
```

we obtain

	Column1	Column2	Column3	Concatenated Column 1
1	Casual	Slippers		CasualSlippers
2	Business	Shoes		BusinessShoes
3	Heavy	Duty	Gloves	HeavyDutyGloves
4	Heavy	Duty	Shoes	HeavyDutyShoes
5	Shoes	Heavy	Duty	ShoesHeavyDuty
6	Gloves	(Light)		Gloves(Light)
7	Gloves	(Heavy Duty)		Gloves(Heavy Duty)
8	Shoes	Business		ShoesBusiness
9	Leather	Belt		LeatherBelt

With that example, the limitations of the **CONCATENATE** function becomes painfully apparent, it can only accept two text inputs at once! Imagine the nightmare nested **CONCATENATE** formula we would have to create if we wanted to include spaces " " in between each column.

Luckily, there is an alternative. We can use the '**&**' operator. It is the equivalent text concatenation *operator*. Using the '**&**' operator, we can concatenate the three columns like this, also including the spaces in between!

='Concatenate'[Column1]&" "&'Concatenate'[Column2]&" "&'Concatenate'[Column3]

[Concatenate... ƒx ='Concatenate'[Column1]&" "&'Concatenate'[Column2]&" "&'Concatenate'[Column3]

	Column1	Column2	Column3	Concatenated Column 1	Concatenated Column 2	Add Colum
1	Casual	Slippers		CasualSlippers	Casual Slippers	
2	Business	Shoes		BusinessShoes	Business Shoes	
3	Heavy	Duty	Gloves	HeavyDutyGloves	Heavy Duty Gloves	
4	Heavy	Duty	Shoes	HeavyDutyShoes	Heavy Duty Shoes	
5	Shoes	Heavy	Duty	ShoesHeavyDuty	Shoes Heavy Duty	
6	Gloves	(Light)		Gloves(Light)	Gloves (Light)	
7	Gloves	(Heavy Duty)		Gloves(Heavy Duty)	Gloves (Heavy Duty)	
8	Shoes	Business		ShoesBusiness	Shoes Business	
9	Leather	Belt		LeatherBelt	Leather Belt	

Of course, if you really wanted, you could combine the '**&**' operator into the **CONCATENATE** function:

```
=CONCATENATE(
        'Concatenate'[Column1]&" ",'Concatenate'[Column2]&" "&'Concatenate'[Column3]
        )
```

[Concatenate... ƒx =CONCATENATE(
'Concatenate'[Column1]&" ",'Concatenate'[Column2]&" "&'Concatenate'[Column3]
)

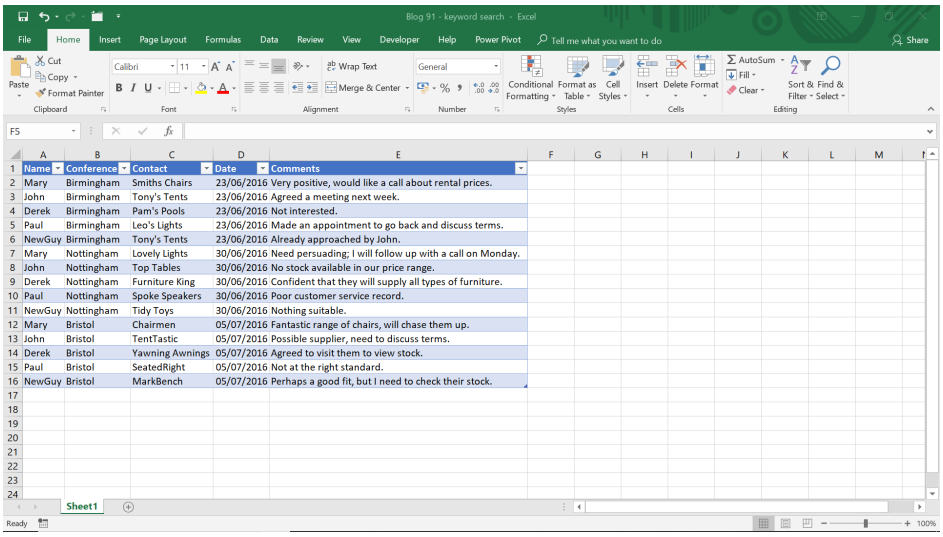
	Column1	Column2	Column3	Concatenated Column 1	Concatenated Column 2	Concatenated Column 3
1	Casual	Slippers		CasualSlippers	Casual Slippers	Casual Slippers
2	Business	Shoes		BusinessShoes	Business Shoes	Business Shoes
3	Heavy	Duty	Gloves	HeavyDutyGloves	Heavy Duty Gloves	Heavy Duty Gloves
4	Heavy	Duty	Shoes	HeavyDutyShoes	Heavy Duty Shoes	Heavy Duty Shoes
5	Shoes	Heavy	Duty	ShoesHeavyDuty	Shoes Heavy Duty	Shoes Heavy Duty
6	Gloves	(Light)		Gloves(Light)	Gloves (Light)	Gloves (Light)
7	Gloves	(Heavy Duty)		Gloves(Heavy Duty)	Gloves (Heavy Duty)	Gloves (Heavy Duty)
8	Shoes	Business		ShoesBusiness	Shoes Business	Shoes Business
9	Leather	Belt		LeatherBelt	Leather Belt	Leather Belt

But we probably wouldn't… the '**&**' operator is just much easier to use, and simpler for other users to understand.

That's it for this month; more next time.

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 look at another **List()** function to search for multiple keywords in columns of data.*



	Name	Conference	Contact	Date	Comments
2	Mary	Birmingham	Smiths Chairs	23/06/2016	Very positive, would like a call about rental prices.
3	John	Birmingham	Tony's Tents	23/06/2016	Agreed a meeting next week.
4	Derek	Birmingham	Pam's Pools	23/06/2016	Not interested.
5	Paul	Birmingham	Leo's Lights	23/06/2016	Made an appointment to go back and discuss terms.
6	NewGuy	Birmingham	Tony's Tents	23/06/2016	Already approached by John.
7	Mary	Nottingham	Lovely Lights	30/06/2016	Need persuading; I will follow up with a call on Monday.
8	John	Nottingham	Top Tables	30/06/2016	No stock available in our price range.
9	Derek	Nottingham	Furniture King	30/06/2016	Confident that they will supply all types of furniture.
10	Paul	Nottingham	Spoke Speakers	30/06/2016	Poor customer service record.
11	NewGuy	Nottingham	Tidy Toys	30/06/2016	Nothing suitable.
12	Mary	Bristol	Chairmen	05/07/2016	Fantastic range of chairs, will chase them up.
13	John	Bristol	TentTastic	05/07/2016	Possible supplier, need to discuss terms.
14	Derek	Bristol	Yawning Awnings	05/07/2016	Agreed to visit them to view stock.
15	Paul	Bristol	SeatedRight	05/07/2016	Not at the right standard.
16	NewGuy	Bristol	MarkBench	05/07/2016	Perhaps a good fit, but I need to check their stock.

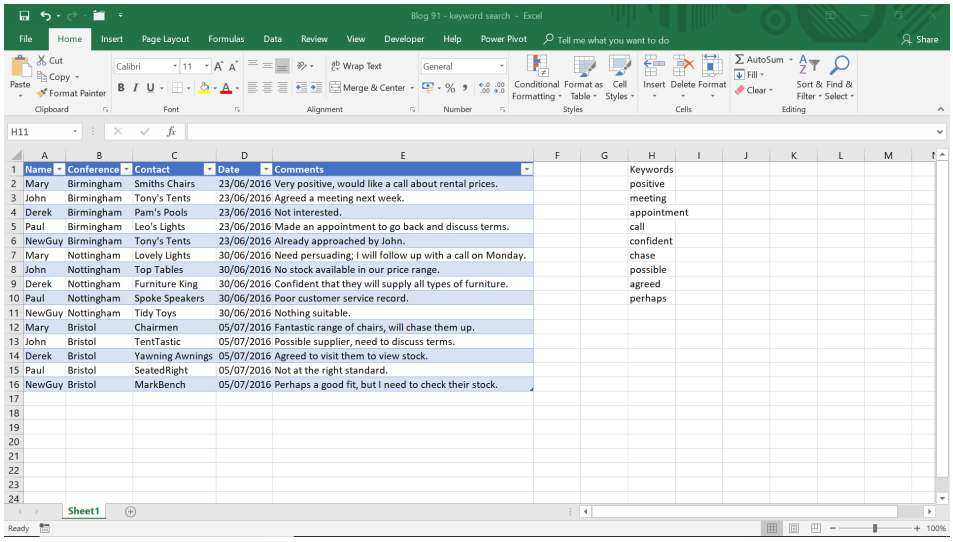
For this newsletter, let's say we have a series of comments recorded by our imaginary salespeople after attending sales conferences. We would like to analyse the comments to see where there needs to be some follow-up.

We plan to use the **M** function **List.ContainsAny()**:

List.ContainsAny(list as list**, values** as list**, optional equationCriteria** as any**)** as logical

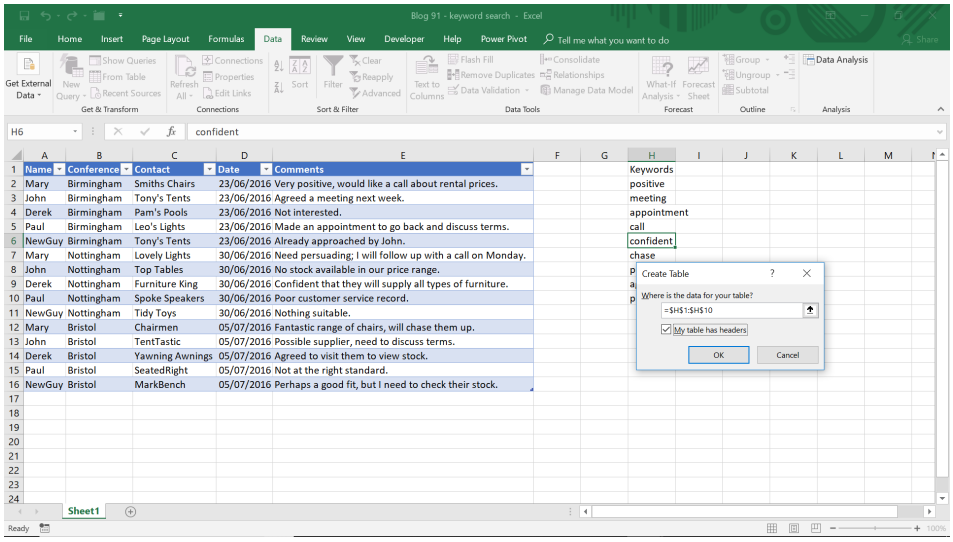
This returns true if any item in **values** is found in a **list**.

The first step is to create a list of keywords. We may either do this by creating a manual list from a blank query or we can convert a table. In this case, we will create a Table in Excel and then convert it into Power Query.

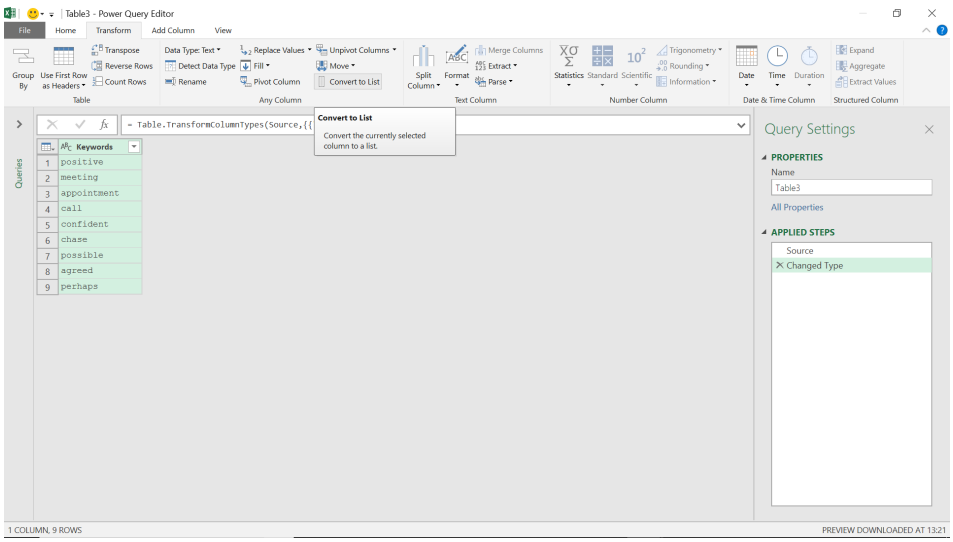


Name	Conference	Contact	Date	Comments	Keywords
Mary	Birmingham	Smiths Chairs	23/06/2016	Very positive, would like a call about rental prices.	positive
John	Birmingham	Tony's Tents	23/06/2016	Agreed a meeting next week.	meeting
Derek	Birmingham	Pam's Pools	23/06/2016	Not interested.	appointment
Paul	Birmingham	Leo's Lights	23/06/2016	Made an appointment to go back and discuss terms.	call
NewGuy	Birmingham	Tony's Tents	23/06/2016	Already approached by John.	confident
Mary	Nottingham	Lovely Lights	30/06/2016	Need persuading; I will follow up with a call on Monday.	chase
John	Nottingham	Top Tables	30/06/2016	No stock available in our price range.	possible
Derek	Nottingham	Furniture King	30/06/2016	Confident that they will supply all types of furniture.	agreed
Paul	Nottingham	Spoke Speakers	30/06/2016	Poor customer service record.	perhaps
NewGuy	Nottingham	Tidy Toys	30/06/2016	Nothing suitable.	
Mary	Bristol	Chairmen	05/07/2016	Fantastic range of chairs, will chase them up.	
John	Bristol	TentTastic	05/07/2016	Possible supplier, need to discuss terms.	
Derek	Bristol	Yawning Awnings	05/07/2016	Agreed to visit them to view stock.	
Paul	Bristol	SeatedRight	05/07/2016	Not at the right standard.	
NewGuy	Bristol	MarkBench	05/07/2016	Perhaps a good fit, but I need to check their stock.	

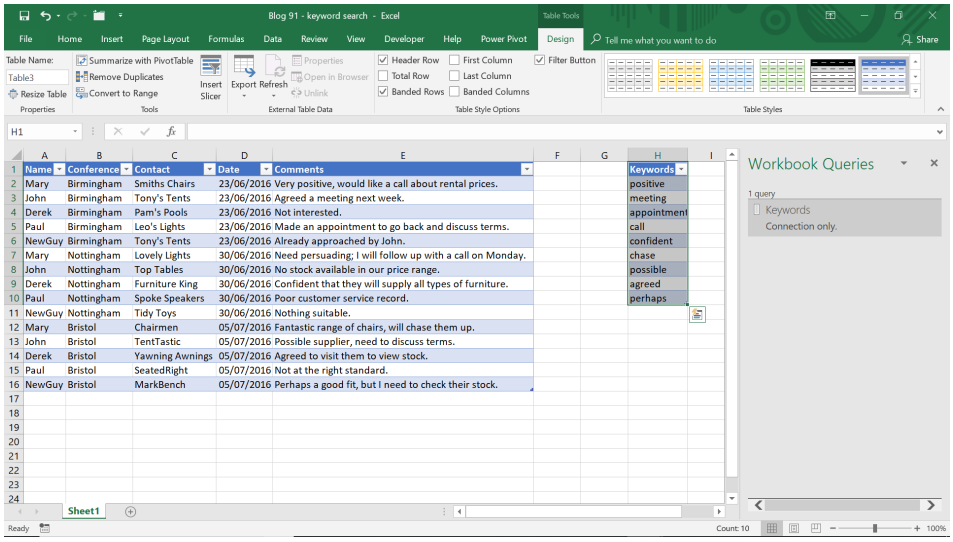
Let's choose 'From Table' in the 'Get & Transform' section on the 'Data' tab.



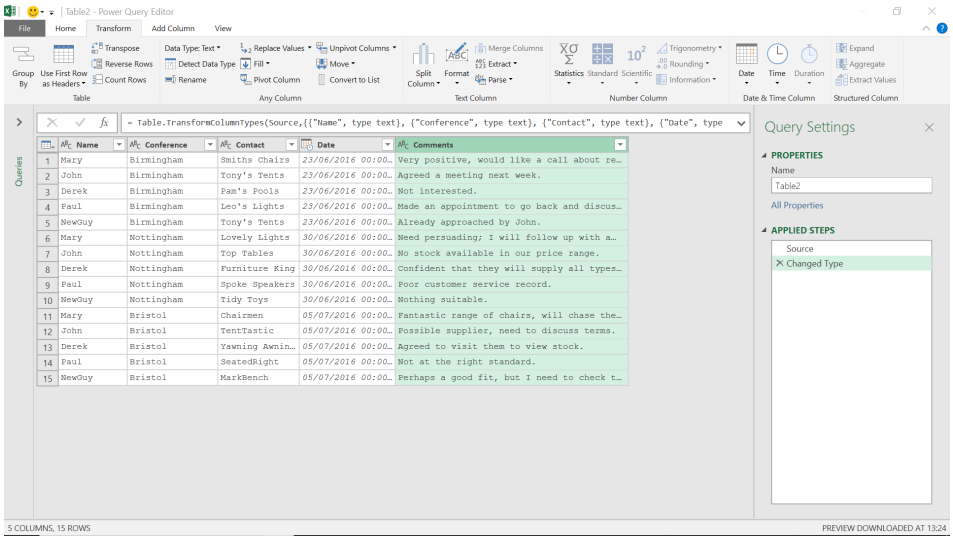
Having defined our Table, we can now convert it to a list by using the 'Convert to List' option in the 'Any Column' section on the 'Transform' tab.



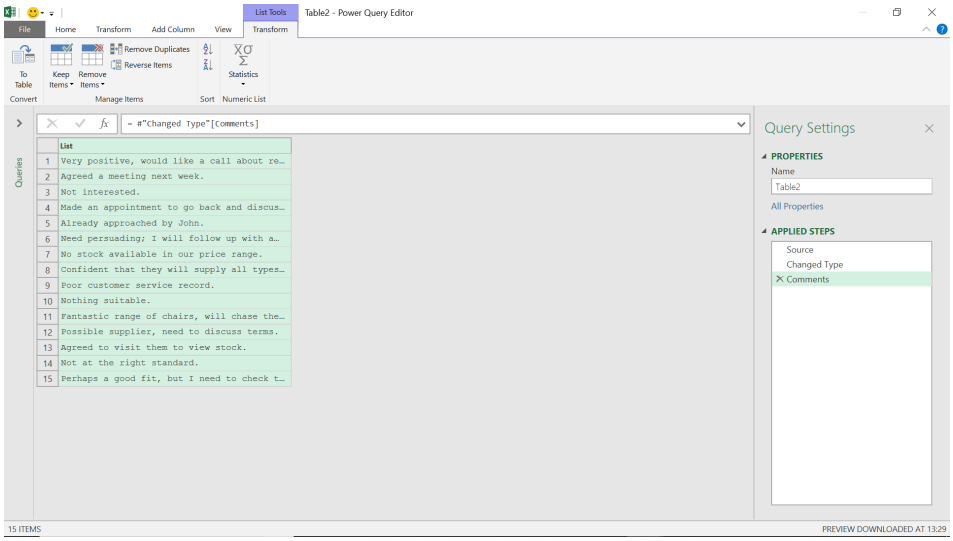
We save the list as ‘Connection Only’ from the ‘Close & Load’ dropdown on the ‘Home’ (or ‘File’) tab.



We may now create a query for the main table.



As we plan to use a **List** function, we need to convert the **Comments** column to a list. However, if we use the Graphical User Interface (GUI) functionality for this, the other columns will disappear:

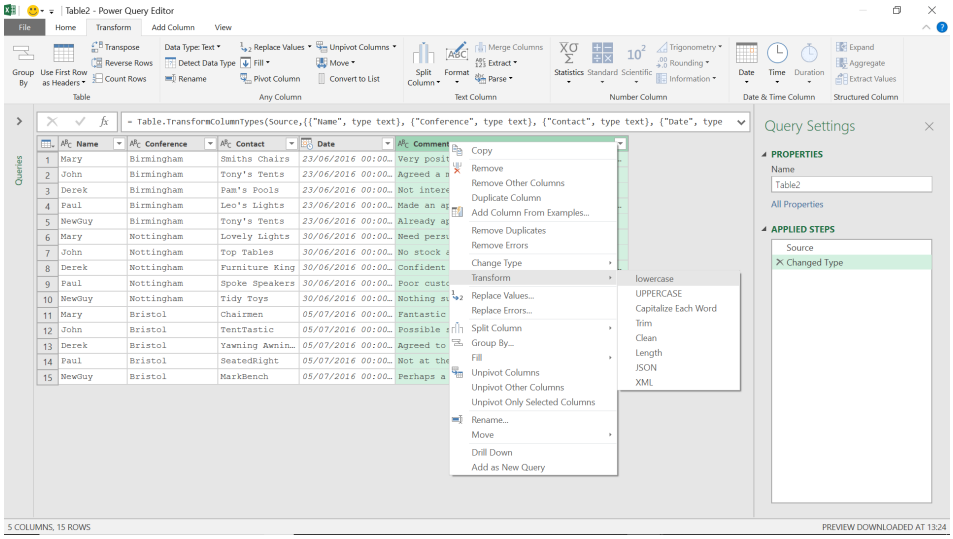


Instead, we need to create some functionality in **M** that will treat the **Comments** column as a list. Last newsletter, we looked at a function called **Text.Split()**:

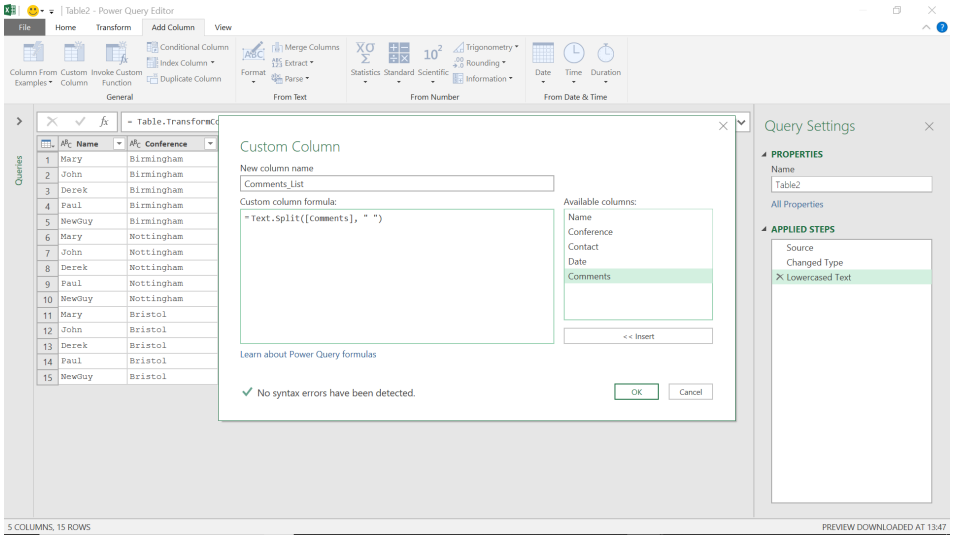
Text.Split(string as text**, separator** as text**)** as list

This returns a list containing parts of a **string** that are delimited by a **separator** text value.

Before we convert **Comments** to a list, let's make sure everything is lowercase, so that our comparison will work. We may do this by right-clicking the column and picking the appropriate 'Transform' option.



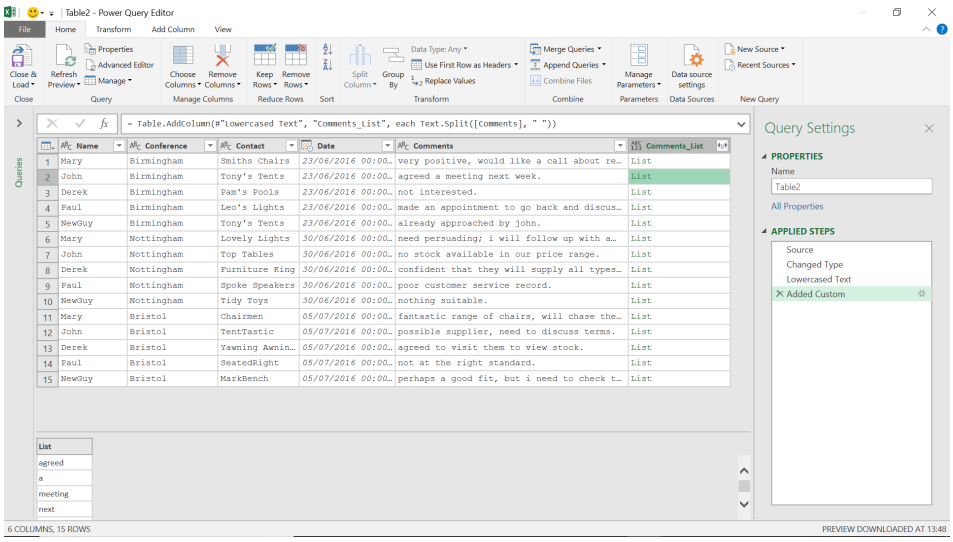
We can now create a custom column from the 'Add Column' section.



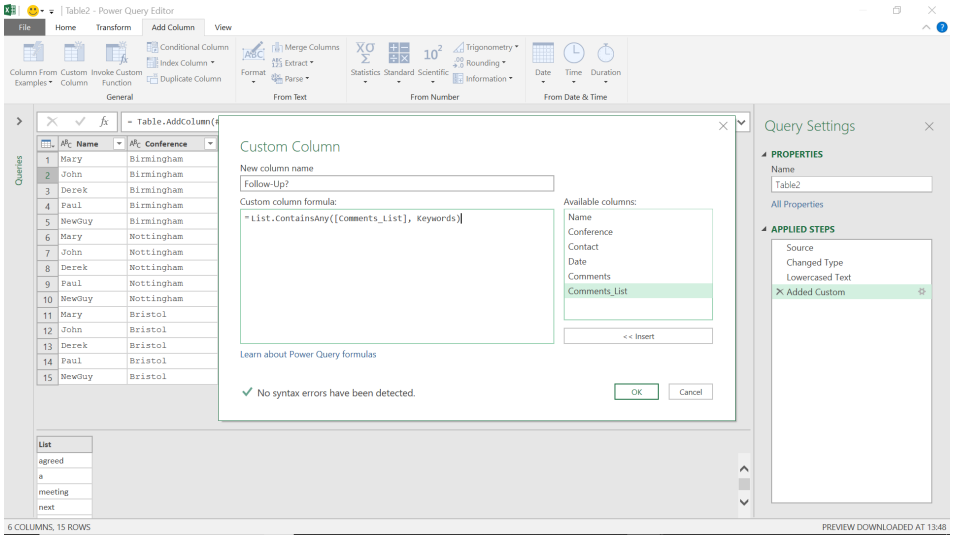
The **M** functionality used is

= Text.Split([Comments]," ")

This creates a list of words by splitting at the space.

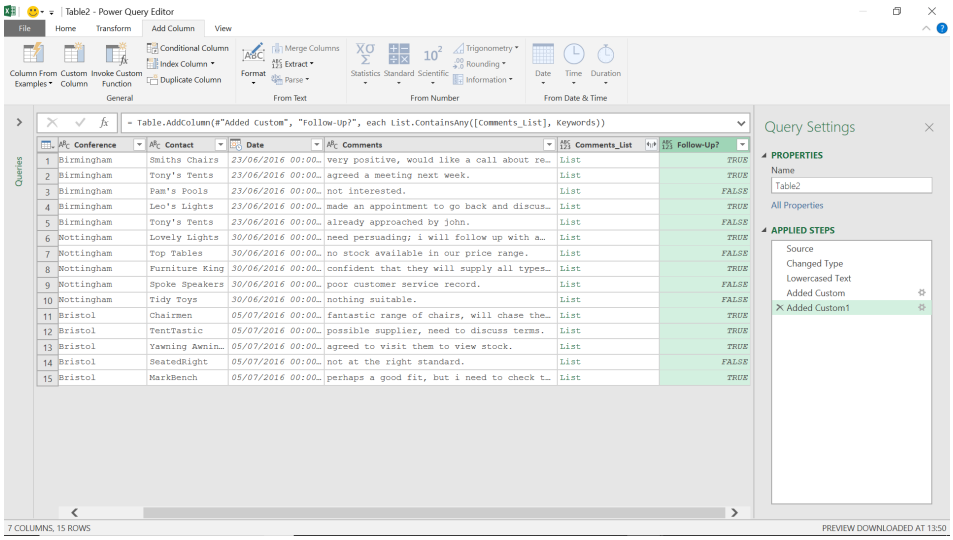


We can now create another custom column to compare values.



The **M** functionality used here is

= List.ContainsAny([Comments_List], Keywords)



We now have a **Follow-Up** column that tells us which contact needs further attention.

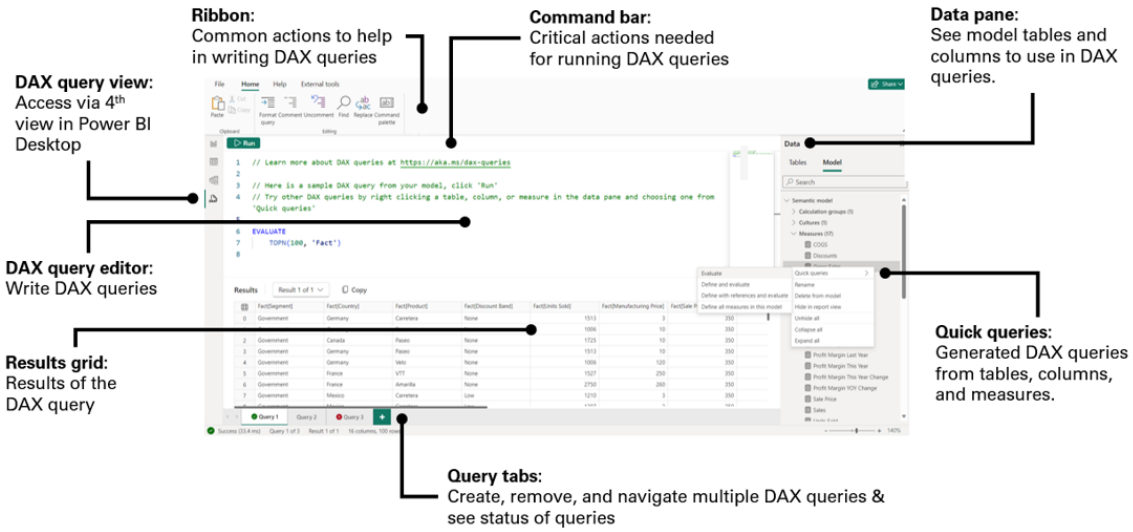
Until February.

Power BI Updates

Due to printing deadlines over the festive season (plus Microsoft staff probably wanted some time off too!), there are no new Power BI Updates reported for this month. Stand by to be overwhelmed by the February edition!

In the meantime, we thought we would reproduce Microsoft's detailed summary on the last update's addition of **DAX** query view.

This **DAX** query view provides a brand new fourth view in public Preview to Power BI Desktop.



The **DAX** query view gives you the ability to write, edit and see the results of Data Analysis eXpressions or **DAX** queries on your semantic model. This means you may now take advantage of the existing **DAX** queries syntax while working with your semantic model without leaving Power BI Desktop.

As this feature is in public Preview, to see the **DAX** query view in Power BI Desktop, you need to make sure you are using at least the November 2023 release and have initialised it via **File -> Options and Settings -> Options -> Preview features** and then clicked the check box next to 'DAX query view'.

This new way to interact with your semantic model in Power BI Desktop comes with several ways to help you be more productive with **DAX** queries:

- **Quick queries from the Data pane makes it easy to create a DAX query.** You may preview data or show summary statistics to help you understand the data without needing to create visuals or writing a **DAX** query. You can find quick queries in the context menu of tables, columns or measures in the Data pane of the **DAX** query view
- **DirectQuery model authors can use DAX query view.** You no longer need to go back to Power Query to preview the data
- **A new measure authoring workflow**. You may now see multiple measures at once in the editor, make changes, run the query to test and finally update the model all in one location
- **See the DAX query of the visuals.** If a particular Power BI visual is not showing the data you expect, you may now investigate further by looking at the **DAX** query used by the visual to get data. This may be accessed from the 'Performance Analyzer' pane
- **Create your own DAX query.** You may simply write the **DAX** query in the **DAX** query view and click run. You can even define and use measures and variables for that **DAX** query that do not exist in the model.

DAX query view, as the name suggests, allows you to create **DAX** queries. This is different to the **DAX** formulae used to create measures and calculated columns. A **DAX** query is like a SQL query in that you can use it to view data in your model.

With this borne in mind, there are two main parts to a **DAX** query:

1. **EVALUATE:** this is required and specifies what data you wish to see
2. **DEFINE:** this is optional and may specify a measure or named **DAX** formula to use in the **DAX** query. This measure may already be in the model (it might not be). If it already exists, you can make changes that only apply to this **DAX** query to try them out. You can also have the option to update the model with these measures *(see below)*.

DAX formula

```
DEFINE
    MEASURE 'Measures group'[Sales] = SUM(Fact[Sales])
    MEASURE 'Measures group'[COGS] = SUM(Fact[COGS])
    MEASURE 'Measures group'[Profit] = [Sales] - [COGS]

EVALUATE
    SUMMARIZECOLUMNS(
        'Fact'[Country],
        "Sales", 'Measures group'[Sales],
        "COGS", 'Measures group'[COGS],
        "Profit", 'Measures group'[Profit]
    )
```

DAX query

Results | Result 1 of 1 | Copy

	Fact[Country]	[Sales]	[COGS]	[Profit]
0	Canada	24887654.88	21358426	3529228.88
1	Germany	23505340.82	19824952	3680388.82
2	France	24354172.28	20573151.5	3781020.78
3	Mexico	20949352.11	18041829	2907523.11
4	United States of America	25029830.17	22034289.5	2995540.67

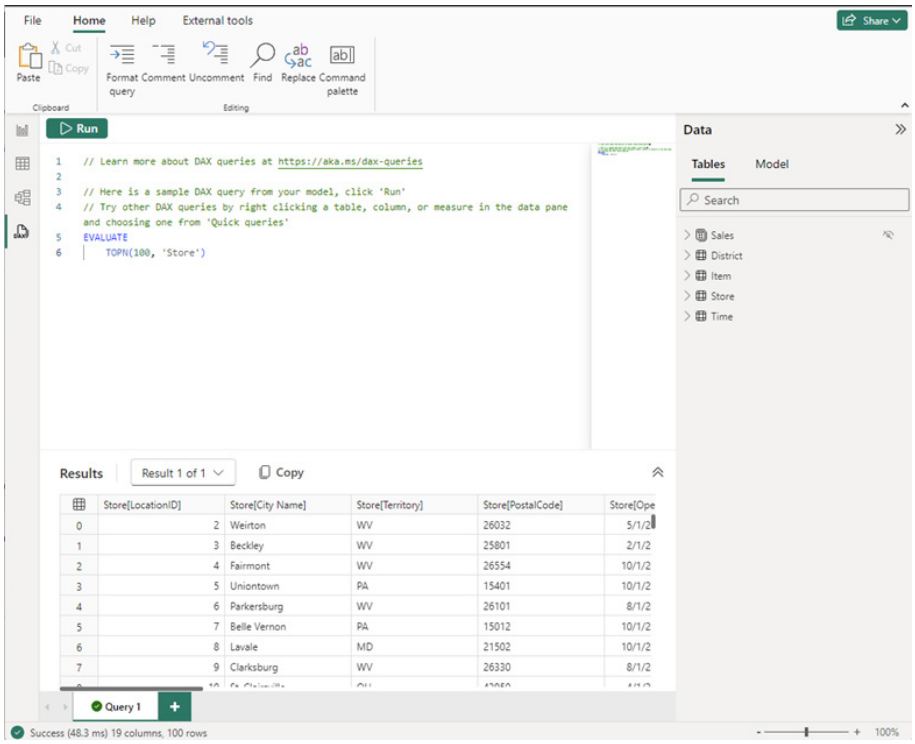
The result of running a **DAX** query is a table of data.

Some of you may already be familiar with **DAX** queries from using DAX Studio. DAX Studio is a community driven and free external tool that can also run **DAX** queries. More than just run **DAX** queries, it has plenty of features utilising **DAX** authoring / performance. It may be accessed in Power BI Desktop from the External tools Ribbon once installed.

DAX query view can generate examples for you so that you may see a **DAX** query, run it and modify it as needed. Here's an example using Quick queries.

Microsoft provide an examples, so that you may follow along. You may download the Store Sales PBIX from https://learn.microsoft.com/power-bi/create-reports/sample-datasets#updated-samples.

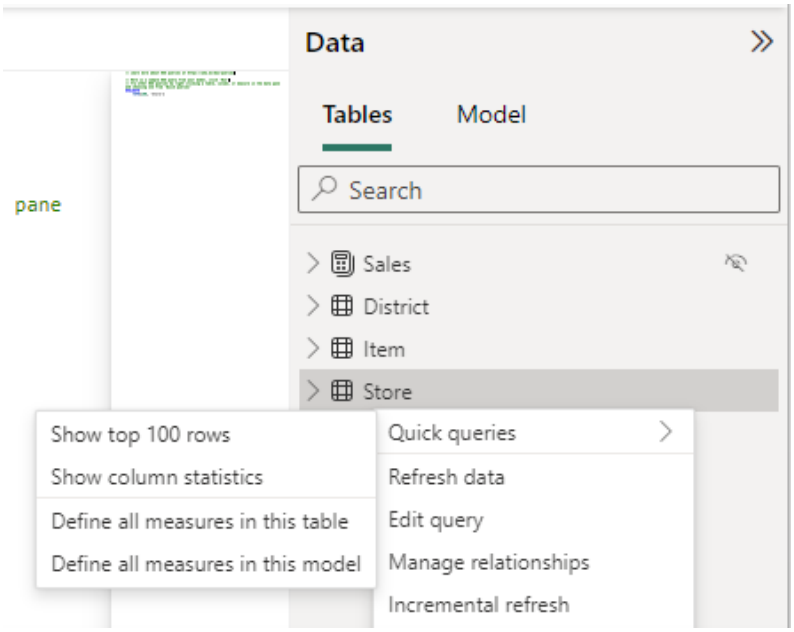
When you first click on **DAX** query view, a sample query is shown to get the top 100 rows of one of the tables in the model. In the case of **Store Sales**, this is the **Store** table.



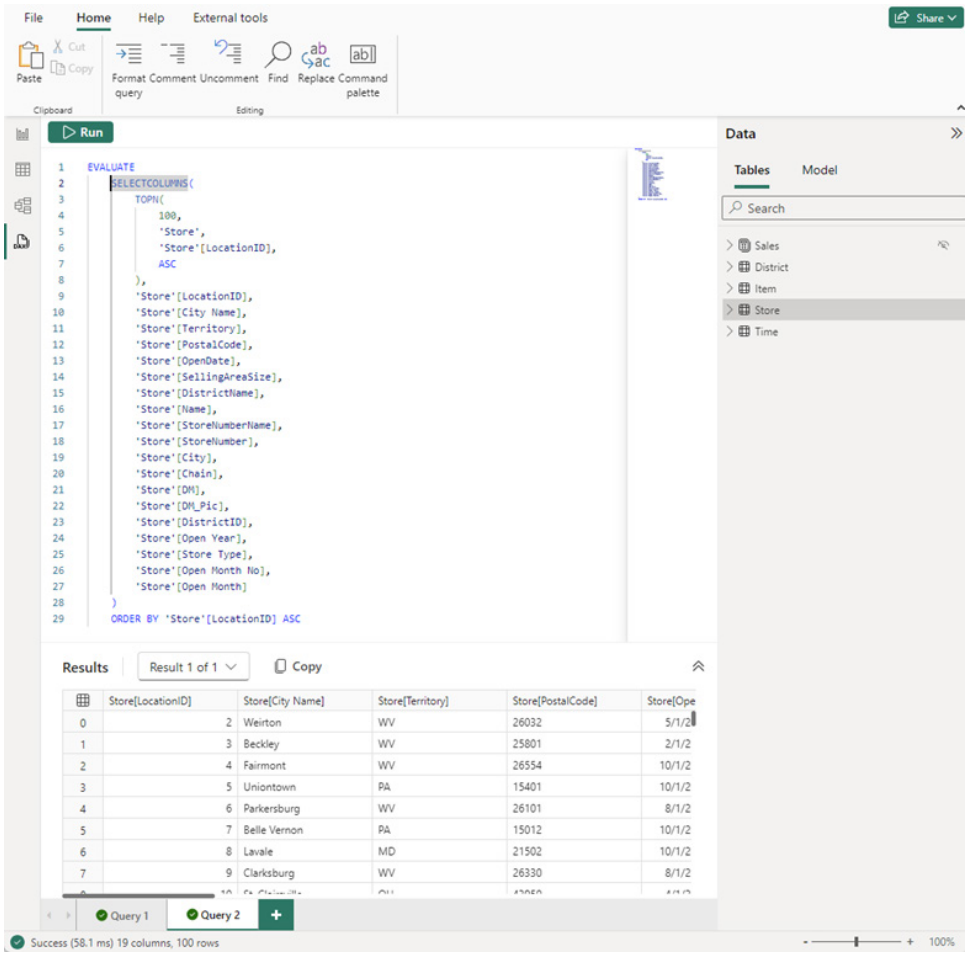
Click Run and the top 100 rows of the table are shown in the result grid. In SQL, this is the same as:

```
SELECT TOP 100 * FROM Store
```

This is great to get a preview of the data for all columns, but it's difficult to change which columns you might wish to see. Therefore, let's try 'Quick queries' instead. In the Data pane, right-click the **Store** table and from the context menu, choose **Quick queries -> Show top 100 rows**.



A new query tab will be created with a different **DAX** query showing the same data. This time, all the columns are explicitly listed. There is a **TOPN** section which also specifies which column and the order in that column to choose the top 100 rows, as well as an **ORDER BY** to specify the result order.

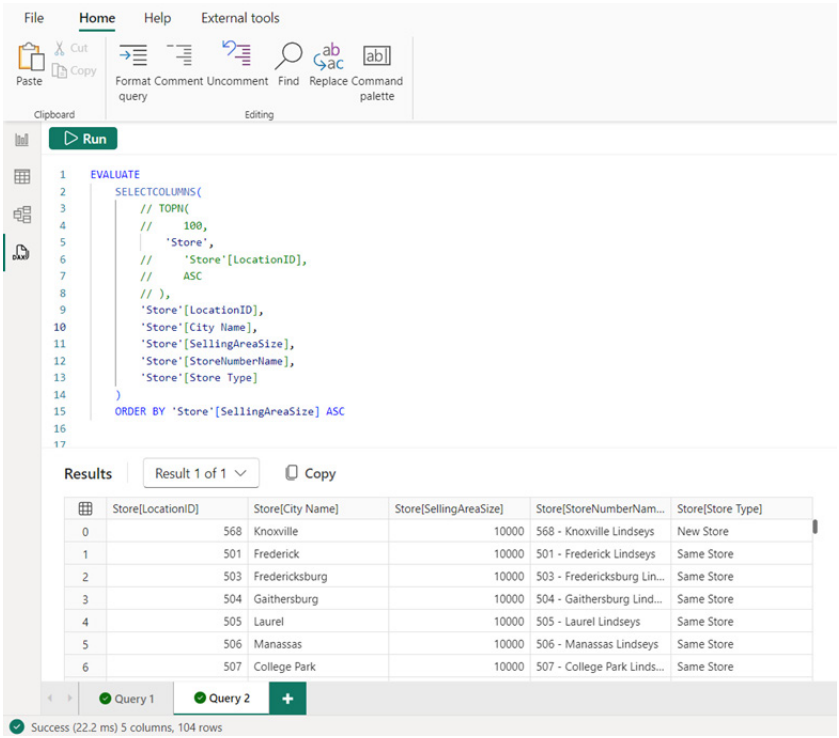


We may now see how **SELECTCOLUMNS** works to get data from the model. In SQL, this would be the same as:

```
SELECT TOP 100
store.[LocationID],
store.[City Name],
store.[Territory],
store.[PostalCode],
store.[OpenDate],
store.[SellingAreaSize],
store.[DistrictName],
store.[Name],
store.[StoreNumberName],
store.[StoreNumber],
store.[City],
store.[Chain],
store.[DM],
store.[DM_Pic],
store.[DistrictID],
store.[Open Year],
store.[Store Type],
store.[Open Month No],
store.[Open Month]
FROM
store
ORDER BY
store.[LocationID] ASC
```

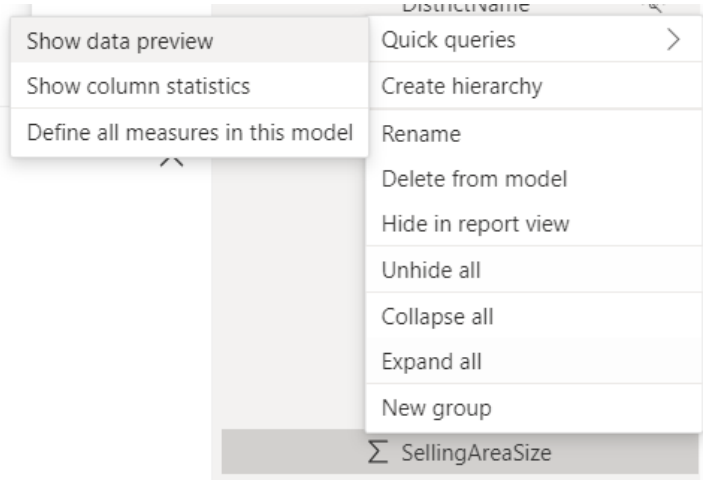
With this quick query we can remove or comment out columns we don't want to see in the result grid, adjust the number of rows, change the order by column, *etc*. **SELECTCOLUMNS** is used for this query because if you have multiple rows with thae same values, they will all show. You may change this to **SUMMARIZE** to de-duplicate the rows.

Let's just look at the **City**, **Store Name**, **Store Type** and **Selling Area Size** for each location and order by the **Selling Area Size** for all rows. To do this, comment out or remove the unwanted columns, change **TOPN** to simply refer to the table and change the column used in **ORDER BY**.

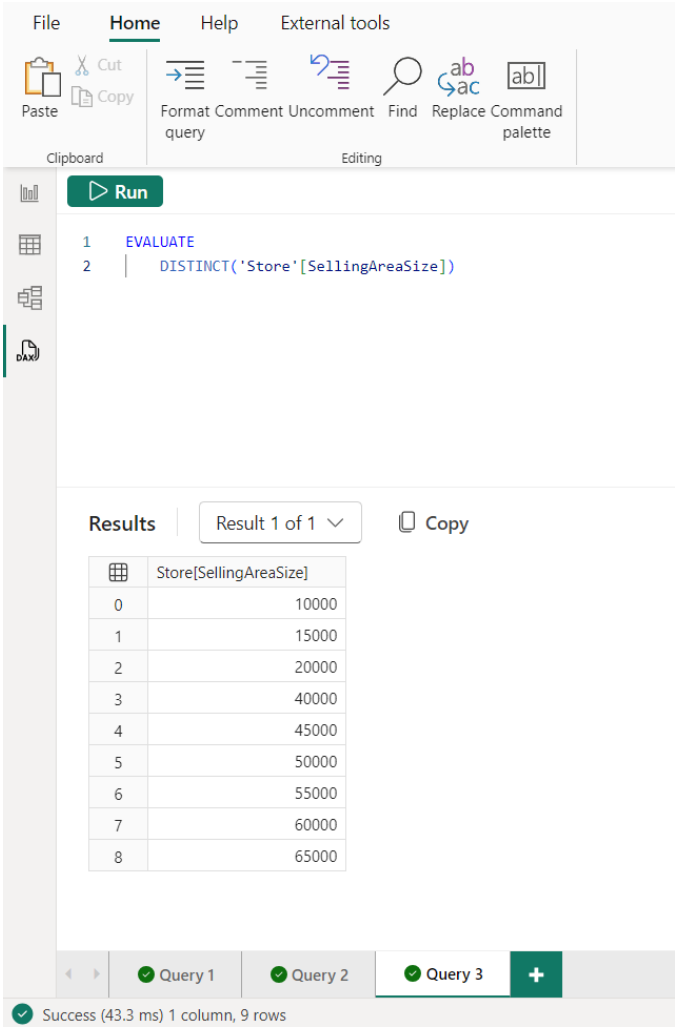


Now, we see targeted information about all 104 stores. We may even copy this and paste the results into Excel.

Now, let's say we're curious to see what the possible **Selling Area Sizes** values are. This looks like it's not an exact number, but instead a way to group stores by size. In the Data pane, right-click the **SellingAreaSize** column and from the context menu choose **Quick queries -> Show data preview**.



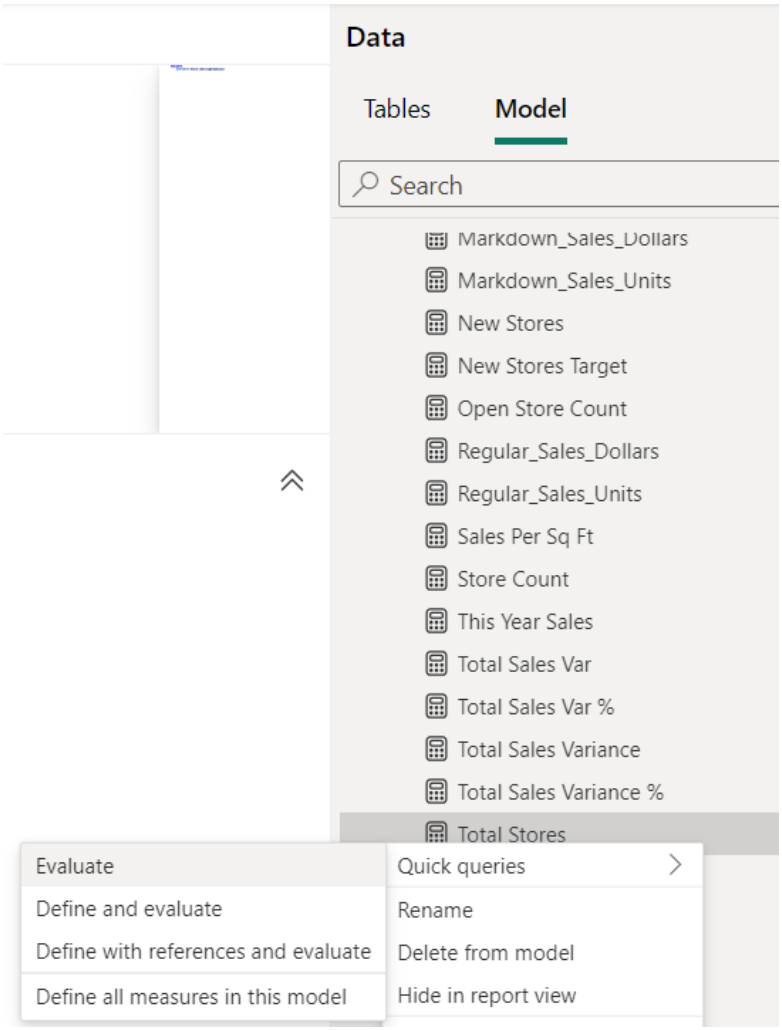
We can now see there are nine [9] values for **Selling Area Size**. As suspected, this is a way to group stores by size.



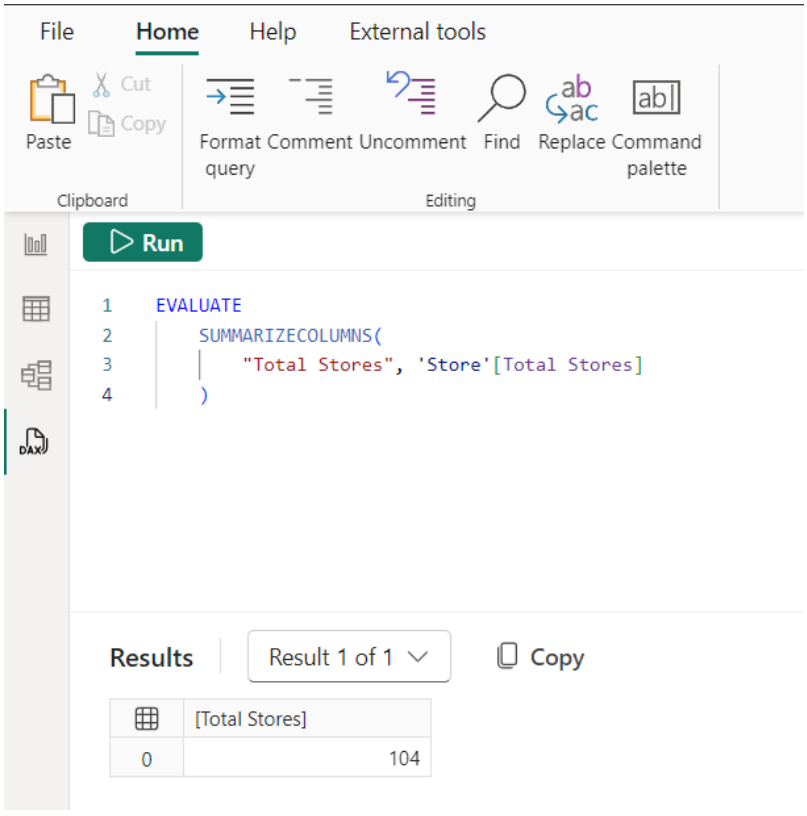
In SQL, this **DAX query** is the same as:

```
SELECT DISTINCT Store.SellingAreaSize
FROM Store
```

Let's consider how many stores we have by each **Selling Area Size**. In this data, there is a measure called **[Store Count]**, so let's see that number by using a quick query. It's easier to find all the measures in the model by changing to Model in the Data pane, or using the Search bar if you already know the name. In the Data pane, right-click the **Store Count** measure and from the context menu choose **Quick queries -> Evaluate**.



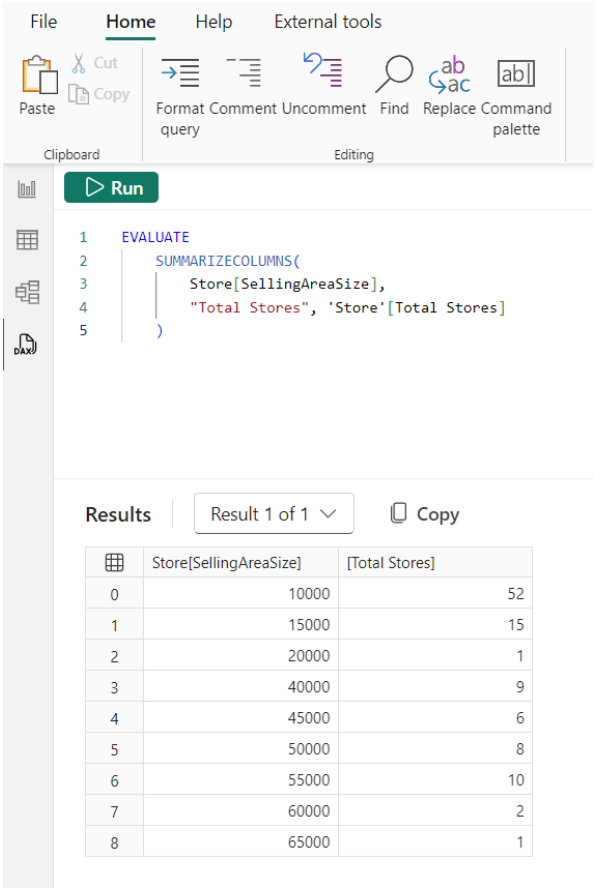
This will create a **DAX** query again in a new query tab. Yet again, we note that there are 104 stores in this data.



In SQL, there is no real equivalent to a measure in a semantic model: you have to define the aggregation in each SQL query, which is instead the same as an implicit measure in a **DAX** query. However, you can get the same result with this SQL query:

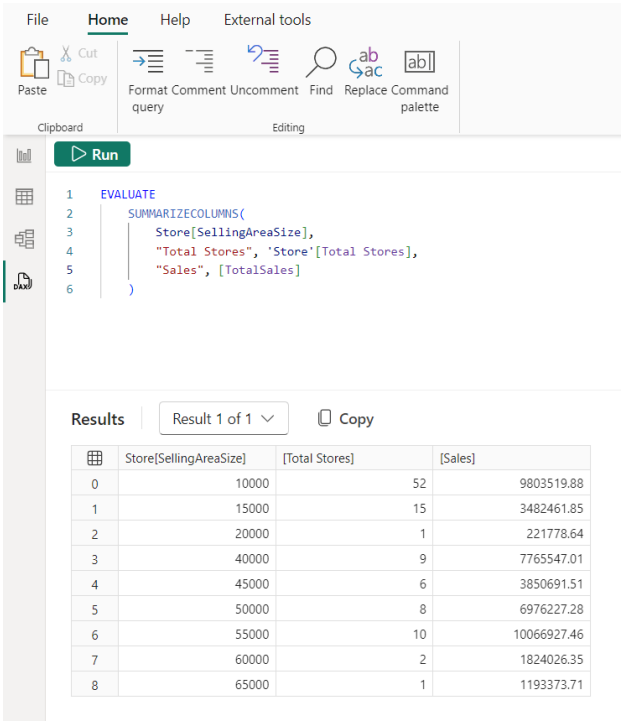
```
SELECT
    COUNT(*) AS 'Store Count'
FROM Store
```

This quick query uses **SUMMARIZECOLUMNS**, which means we can add in a group by column, such as **Selling Area Size**, to answer the question about how many stores we have for each store size.



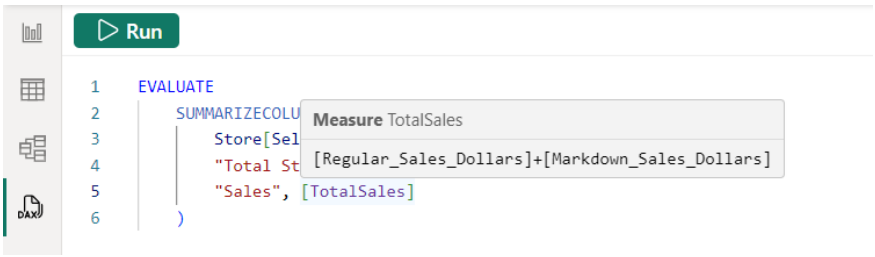
	Store[SellingAreaSize]	[Total Stores]
0	10000	52
1	15000	15
2	20000	1
3	40000	9
4	45000	6
5	50000	8
6	55000	10
7	60000	2
8	65000	1

You'll notice most of the stores are comparatively small. We can build on this query even further, not only by adding in more group by columns, but also by adding in more measures. Let's add in **Sales**.



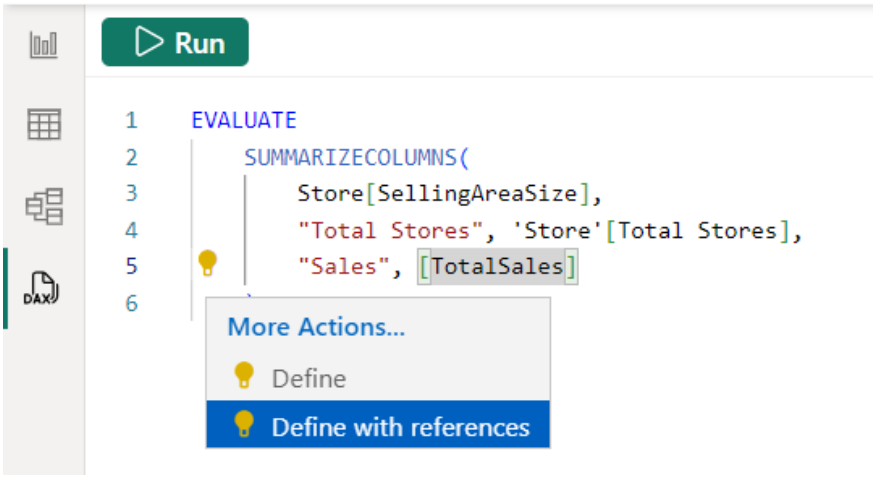
	Store[SellingAreaSize]	[Total Stores]	[Sales]
0	10000	52	9803519.88
1	15000	15	3482461.85
2	20000	1	221778.64
3	40000	9	7765547.01
4	45000	6	3850691.51
5	50000	8	6976227.28
6	55000	10	10066927.46
7	60000	2	1824026.35
8	65000	1	1193373.71

The **DAX** query view can also show the **DAX** formula of the **[TotalSales]** measure. We may hover over it to see it in an overlay:

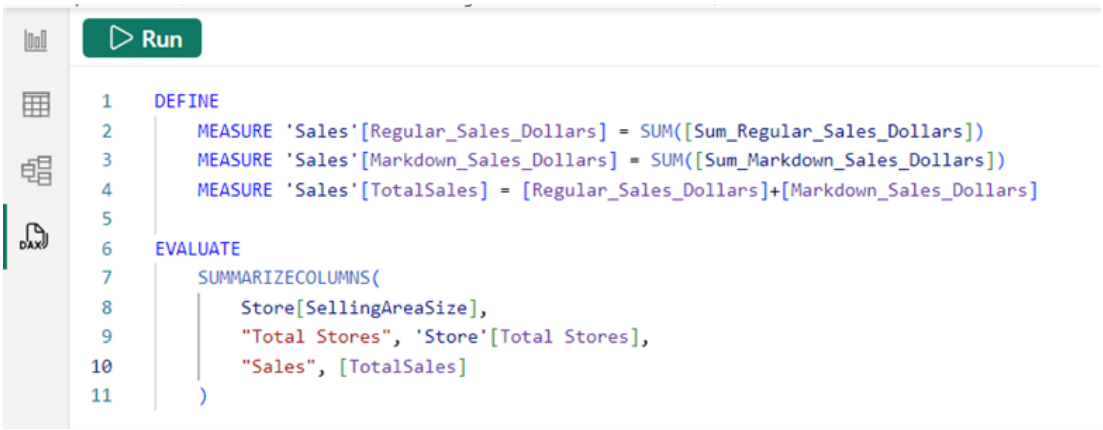


We can see that it's referencing other measures in the model. However, we can't see their **DAX** formulae in the overlay, but **DAX** query view can take advantage of the **DEFINE** syntax in **DAX** queries. We may show this measure's **DAX** formula and all referenced measures' **DAX** formulae in just a couple of clicks:

- click on the measure name, placing the cursor in the measure name on line 5. A lightbulb will appear to the left
- click on the lightbulb to see the actions available or use **CTRL + .** (period)
- click on **Define with references**.

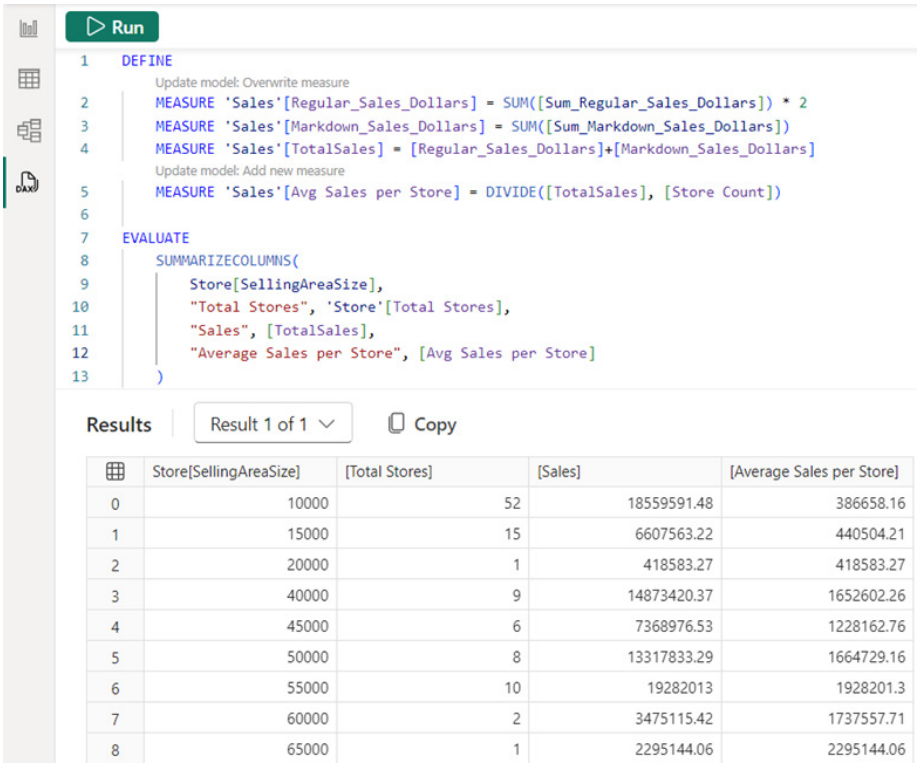


This will create the **DEFINE** block for this **DAX** query just above the **EVALUATE**. These won't be available if you already have a **DEFINE** command in the query tab.



Not only can you see the **DAX** formulae, you may even edit one or more of them. When you run the **DAX** query, they will use the modified version in the query tab over the model measure **DAX** formula. This way we can test any changes. Here, we have doubled one of the measures.

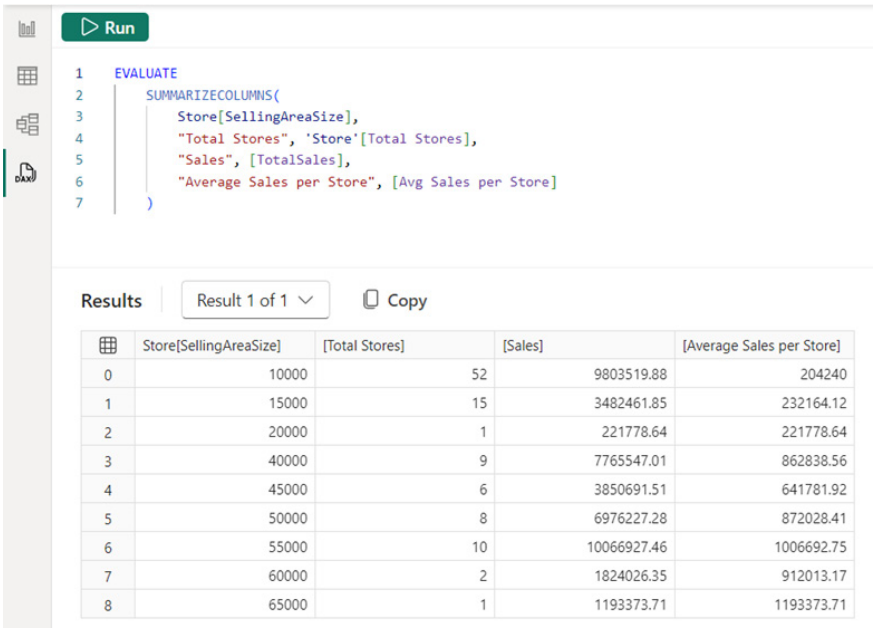
We may even add a measure to use in the **DAX** query that doesn't yet exist in the model, such as to see what the average sales per store is for each store size.



	Store[SellingAreaSize]	[Total Stores]	[Sales]	[Average Sales per Store]
0	10000	52	18559591.48	386658.16
1	15000	15	6607563.22	440504.21
2	20000	1	418583.27	418583.27
3	40000	9	14873420.37	1652602.26
4	45000	6	7368976.53	1228162.76
5	50000	8	13317833.29	1664729.16
6	55000	10	19282013	1928201.3
7	60000	2	3475115.42	1737557.71
8	65000	1	2295144.06	2295144.06

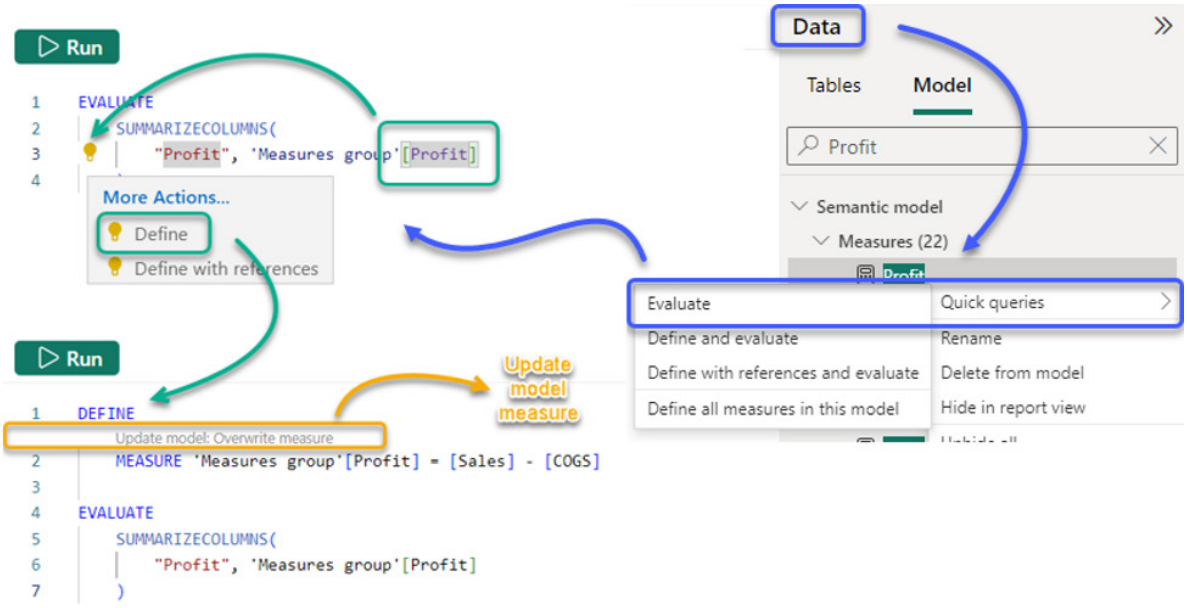
DAX query view can detect when you have changed the **DAX** formula in a measure that exists in the model, so a clickable superscript appears, called a **CodeLens**, which will update the model with the new **DAX** formula if clicked. For a measure that doesn't already exist in the model, the **CodeLens** will add this measure to the model when clicked. We don't want to keep the multiply by two [2] change, but we do want to add in the average sales per store measure. The measure is added to the model and the **CodeLens** disappears.

We may even remove the **DEFINE** block and run the query again.

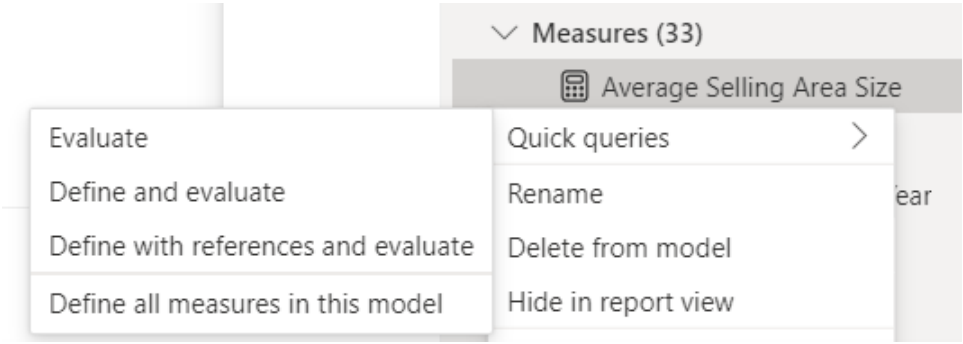


	Store[SellingAreaSize]	[Total Stores]	[Sales]	[Average Sales per Store]
0	10000	52	9803519.88	204240
1	15000	15	3482461.85	232164.12
2	20000	1	221778.64	221778.64
3	40000	9	7765547.01	862838.56
4	45000	6	3850691.51	641781.92
5	50000	8	6976227.28	872028.41
6	55000	10	10066927.46	1006692.75
7	60000	2	1824026.35	912013.17
8	65000	1	1193373.71	1193373.71

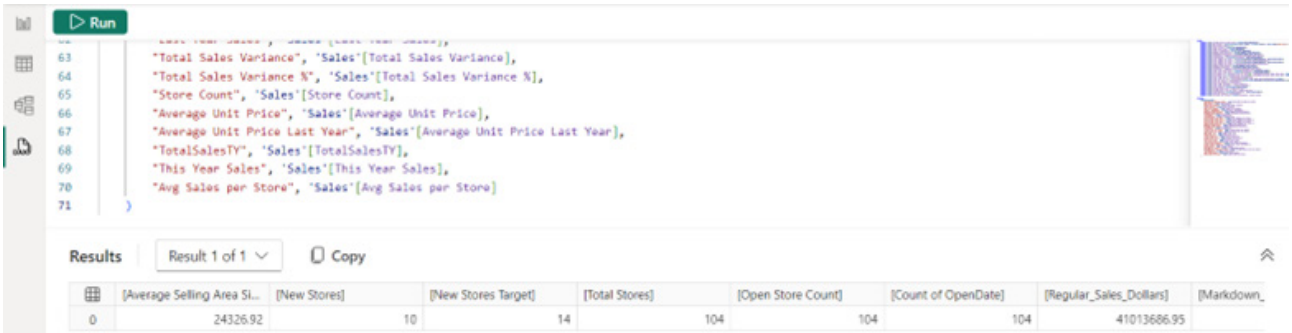
The larger selling area size of the store does show higher average sales. The measure quick queries and **CodeLens** together create a new measure authoring workflow in **DAX** query view.



The quick queries for measures also has the option to define all the measures in a table or model. In the Data pane, right-click the any measure and from the context menu choose **Quick queries -> Define all measures in this model**.



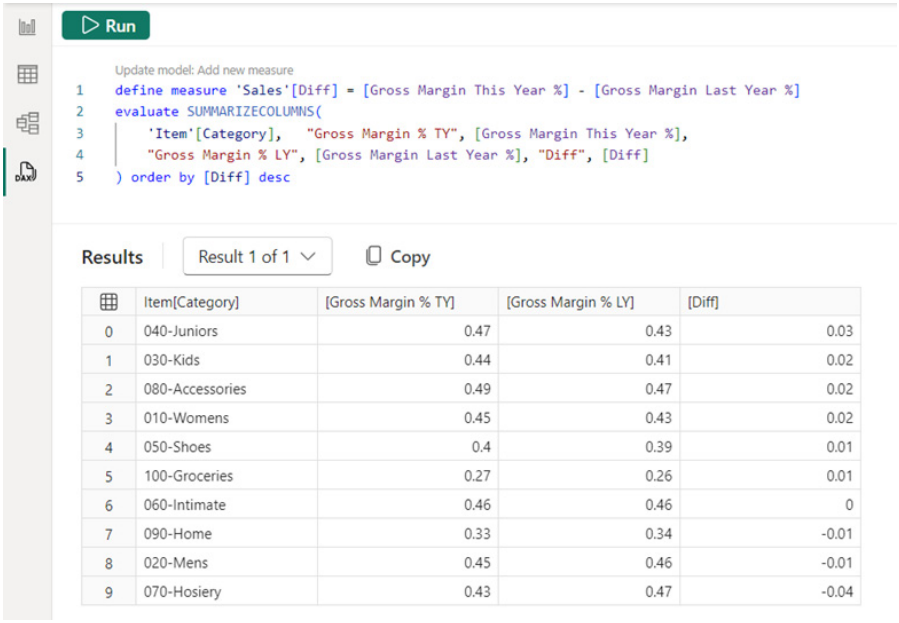
You now have a large **DAX** query that defines all the measures and creates an **EVALUATE** block to see them all at the model level.



SQL again doesn't have an equivalent to measures in the semantic model. These would all need to be aggregations in a SQL query, but **DAX** formulae can reference other measures and perform context changing (looking at last year or by a particular filter) which is more challenging to reproduce in SQL.

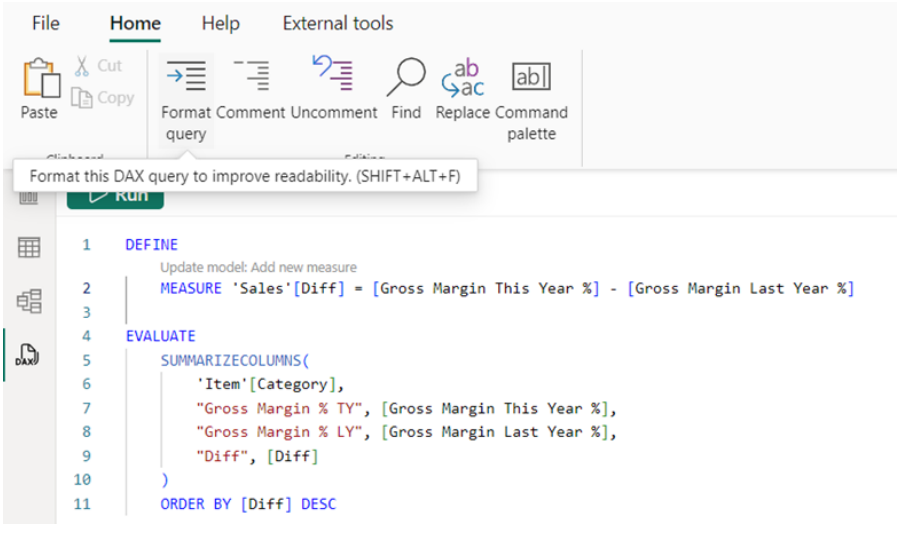
Once we have all our measures in one single query tab, we can do things such as "find". We can see how many measures use the **Selling Area Size** column, as an example. To do this, we click the 'Find' Ribbon button or use **CTRL + F**. You will note there are two [2] measures that are using that column:

Now we have a feel for how **DAX** queries work, we may create our own. To start, let's add a new query tab. We can use **SUMMARIZECOLUMNS** to see how gross margin compares across item categories and then define our own measure to show the year over year difference and order the results by categories that improved the most.

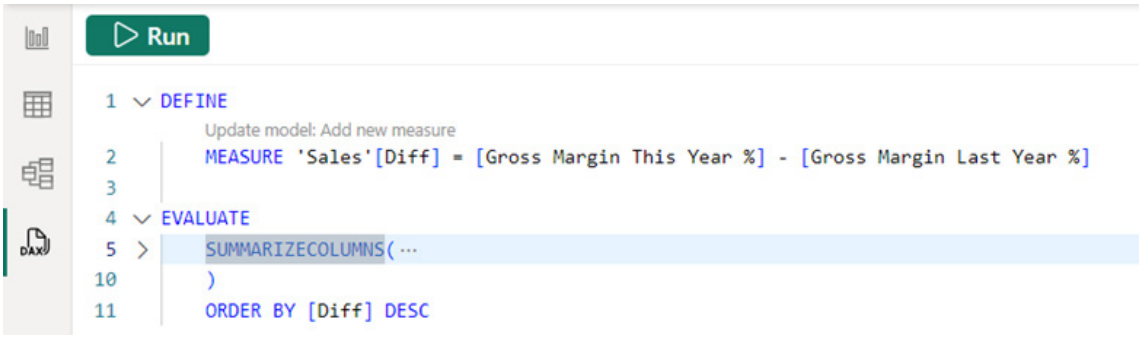


	Item[Category]	[Gross Margin % TY]	[Gross Margin % LY]	[Diff]
0	040-Juniors	0.47	0.43	0.03
1	030-Kids	0.44	0.41	0.02
2	080-Accessories	0.49	0.47	0.02
3	010-Womens	0.45	0.43	0.02
4	050-Shoes	0.4	0.39	0.01
5	100-Groceries	0.27	0.26	0.01
6	060-Intimate	0.46	0.46	0
7	090-Home	0.33	0.34	-0.01
8	020-Mens	0.45	0.46	-0.01
9	070-Hosiery	0.43	0.47	-0.04

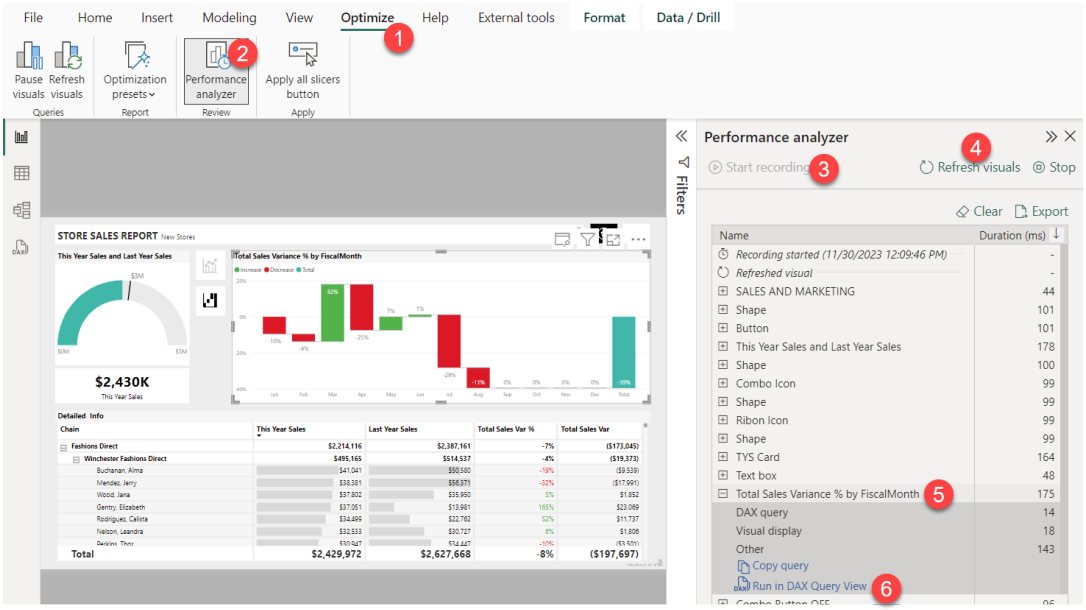
We may quickly do this analysis simply by using **DAX** queries. This **DAX** query doesn't look as pretty as the Quick queries, as we typed it by hand not paying attention to the formatting. However, we may click the 'Format query' Ribbon button or right-click and choose 'Format document' or even use **SHIFT + ALT + F** to format our **DAX** query.

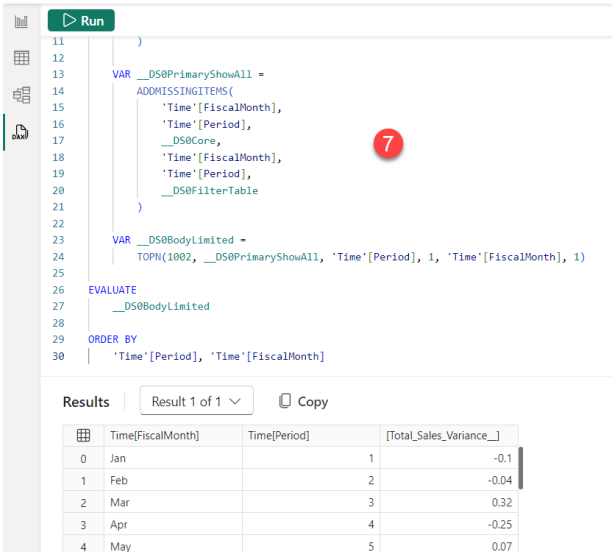


Formatting is more than just making it pretty looking and easier to read. We can collapse blocks where they are indented too.



The visuals themselves are also more than just pretty data visualisations. The report view visuals get the data from the model with a **DAX** query. It is possible to see the **DAX** query in **DAX** query view as well. In Report view, go to the 'Optimize' Ribbon and click 'Performance Analyzer'. Now, click 'Start recording' followed by 'Refresh visuals'. Finally, expand the visual title in the list and click 'Run with DAX query view'. This will move you to **DAX** query view where you see the visual **DAX** query and the results.





Power BI Updates *should* return next month.

New Features for Excel

In our first update of the new year, there are several new features and functions "officially" released. This includes the **GROUPBY** and **PIVOTBY** functions, together with Export Loop Tables, Workbook Links and Trendline Equation Formatting across the Excel media.

You can check out the full list here:

Excel for the web

- Trendline Equation Formatting

Excel for Windows

- Workbook Links
- **GROUPBY** and **PIVOTBY** functions (Insiders)
- Export Loop Tables to Excel (Insiders)

Excel for Mac

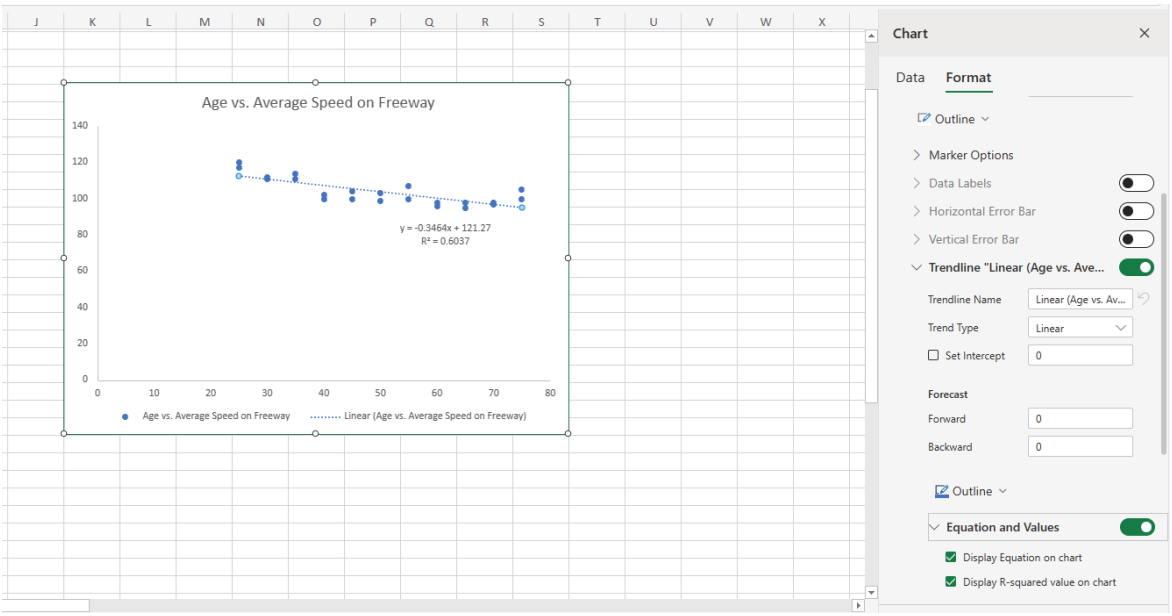
- **GROUPBY** and **PIVOTBY** functions (Insiders)
- Export Loop Tables to Excel (Insiders).

Let's get started.

Trendline Equation Formatting

In Excel for the web, additional trendline equation formatting controls are now available in the Chart Format task pane. This includes number and font / fill / outline formatting.

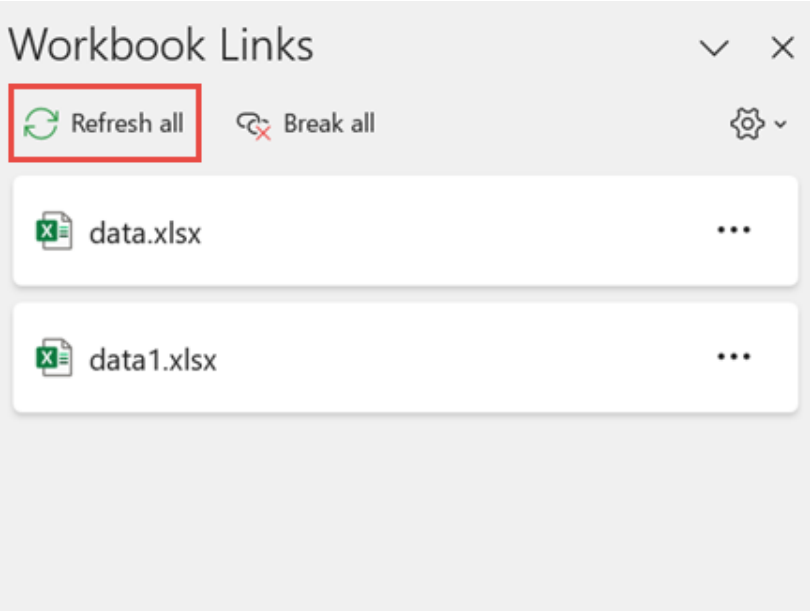
Simply plot your data points, add a Trendline (by turning it on in the pane) and then choose the appropriate options for 'Equation and Values', *viz.*



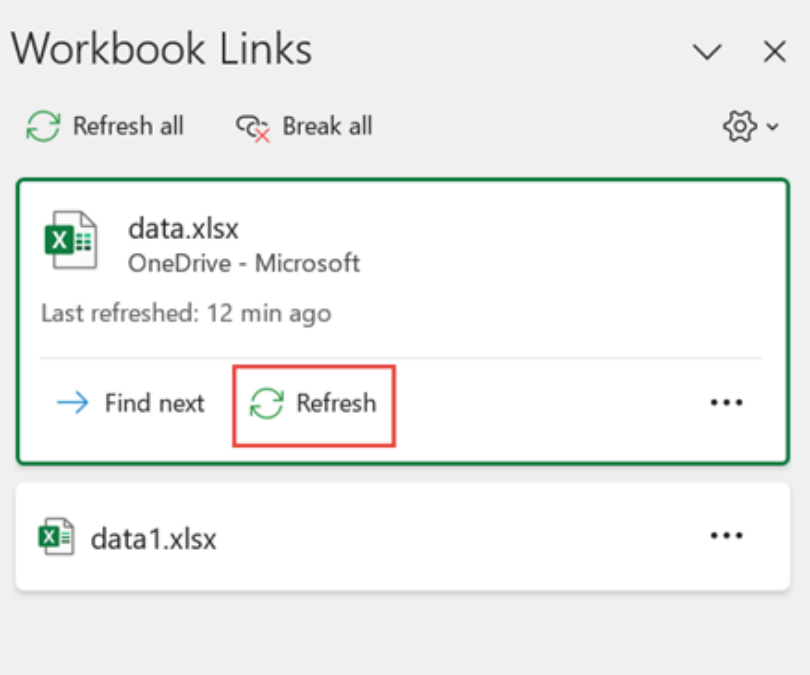
Workbook Links

In Excel for Windows, the new 'Workbook Links' task pane has been provided with the intention to improve the experience and reliability for working with linked files, including the long asked for Find feature to help you figure out where the links are used. If only we could have had that 30 years ago!

To ensure that you are retrieving the latest values from your source workbooks (*i.e.* refresh and update all workbook links), open the 'Workbook Links' pane, and then select **Data -> Queries and Connections -> Workbook Links**. Then, select 'Refresh all' at the top of the pane:



To simply refresh a specific workbook link, open the 'Workbook Links' pane, and again select **Data -> Queries and Connections -> Workbook Links**. Then, select the workbook from the list and select Refresh.



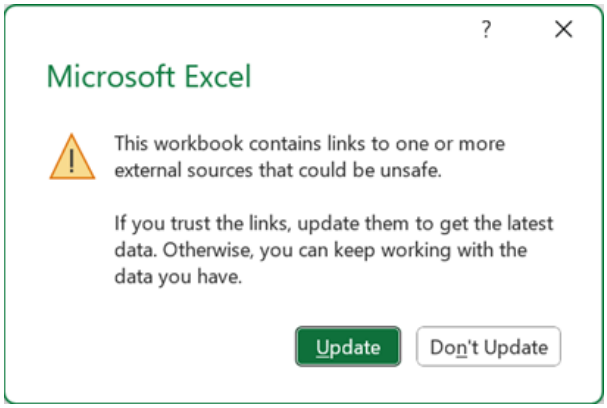
If you wish to refresh all workbook links automatically, be aware that Desktop workbooks don't have the 'Refresh Automatically' command because this operation normally occurs with automatic recalculation when opening the workbook. You can decide whether the links in a given workbook are refreshed when opening the file according to the user's setting, not refreshed when opening the file or refreshed automatically without prompting the user. This option affects all users of the workbook. If you choose not to refresh links and not to prompt, users of the workbook will not know that the data is out of date.

To open the 'Workbook Links' pane, select **Data -> Queries and Connections -> Workbook Links** (just for a change). Select and expand Refresh settings at the top right corner of the pane and select one of the following:

- **Ask to refresh:** asks the user to refresh or not when the workbook opens
- **Always refresh:** automatically refreshes all links when the workbook opens
- **Don't refresh:** doesn't refresh on open and doesn't ask the user.

When you open a destination workbook and the source workbook is not open, you may be alerted by the Trust bar to update the links. You may control whether the Trust bar alerts you, and whether to update all links when the alert does not appear. You can also choose to update only certain links if the workbook contains more than one.

To manually update all or none of the workbook links, close all source workbooks. If one source workbook is open, and others are closed, the updates will not be uniform. Then, open the destination workbook. In the 'Unsafe links' warning dialog, select 'Update'. This updates all the data links in the workbook.



If you only want to update specific links, select 'Don't Update'.

If you get a 'Security Warning' bar when you first open a workbook, it means the workbook is not yet trusted. To trust the workbook:

- select 'Enable Content'. This makes the workbook trusted for this session and updates the links
- if you get a dialog asking to make it a trusted document, select 'Yes' to avoid the 'Security Warning' bar in the future. If you select 'No', you'll get the 'Security Warning' bar the next time you open the workbook. Depending upon your settings, you may also see the 'Unsafe links' warning dialog.

Sometimes, you need to change the source workbook, a workbook link breaks or you may no longer need the workbook link.

OPEN THE SOURCE WORKBOOK

You may want to examine and inspect the source workbook first, before making significant changes. To open the 'Workbook Links' pane, select **Data -> Queries and Connections -> Workbook Links**. You can select More Commands (**...**) next to the required workbook and then select 'Open workbook'.

CHANGE THE SOURCE WORKBOOK

To change the source workbook for all references within the destination workbook:

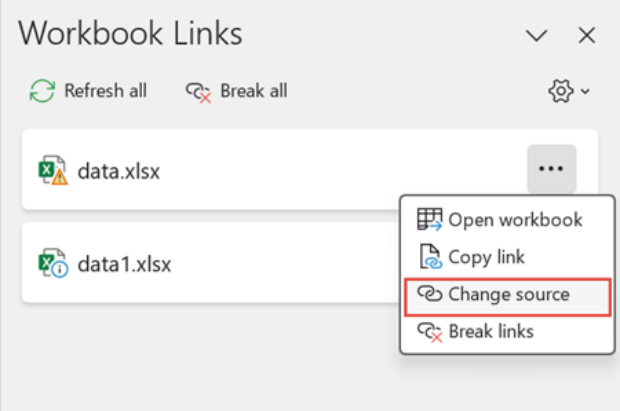
- open the 'Workbook Links' pane, select **Data -> Queries and Connections -> Workbook Links** (I have seen that tip somewhere before…)
- select More Commands (**...**) next to the required workbook and then select 'Change source'
- in the 'Change source' dialog box, under Recent, select the new source workbook. Alternatively, select Browse, and from the dialog box, open the new source file.

To change the source workbook for a particular reference within the destination workbook:

- find the workbook that you want to use as the new source for the external reference and note its location
- in the destination workbook, select the cell with the external reference that you want to change
- in the Formula bar, look for a reference to another workbook, such as **C:\Reports\[Budget.xlsx]**, and replace that reference with the location of the new source workbook.

FIX A BROKEN WORKBOOK LINK

To open the 'Workbook Links' pane, select **Data -> Queries and Connections -> Workbook Links.** Select More Commands (**...**) next to the data workbook with links you want to fix, then select 'Change source'.



In the 'Change source' dialog box, browse to the location of the file containing the linked data, and then select the new source file.

BREAK A WORKBOOK LINK

To open the 'Workbook Links pane', select **Data -> Queries and Connections -> Workbook Links**. Then, select More Commands (**...**) next to the required workbook and then select 'Break links'.

It's important to note that when you break a link to the source workbook of an workbook link, all formulae that use the value in the source workbook are converted to their current values. For example, if you break the link to the workbook link **=SUM([Budget.xls]Annual!C10:C25)**, the **SUM** formula is replaced by the calculated value, whatever that may be. Also, because this action cannot be undone, we strongly recommend that you save a version of the destination workbook as a backup before undertaking this action.

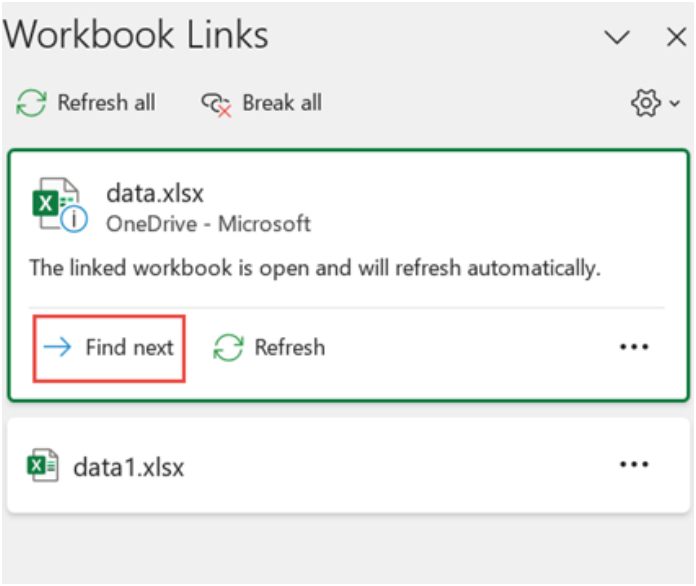
Furthermore, if you use an external data range, a parameter in the query may be using data from another workbook. You may want to check for and remove any of these type of links too.

BREAK ALL WORKBOOK LINKS

To open the 'Workbook Links' pane, select **Data -> Queries and Connections -> Workbook Links**. Then, select 'Break all' at the top of the pane.

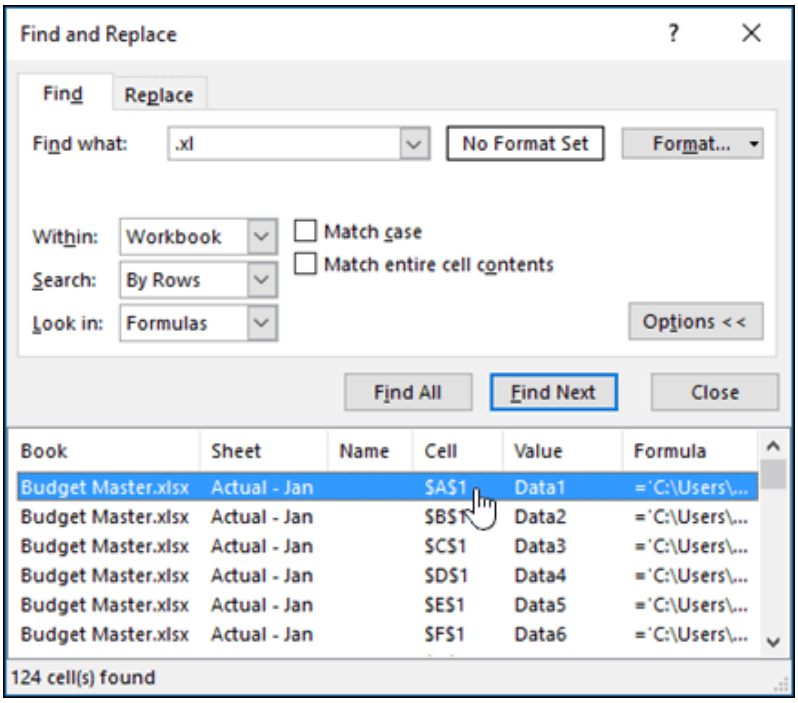
One of the big problems in Excel is finding all the workbook links contained therein. There is no automatic way to find all workbook links in a workbook. You need to look differently in formulae, defined names, objects (like text boxes or shapes), chart titles and chart data series. Indeed, one of our earliest articles was on hunting down elusive workbook links!

There may be several workbook links in a workbook. Here's how to locate the one you want. Just in case you didn't know, to open the 'Workbook Links' pane, select **Data -> Queries and Connections -> Workbook Links**. Then, select the workbook from the list and select 'Find next'.



To find workbook links used in formulae:

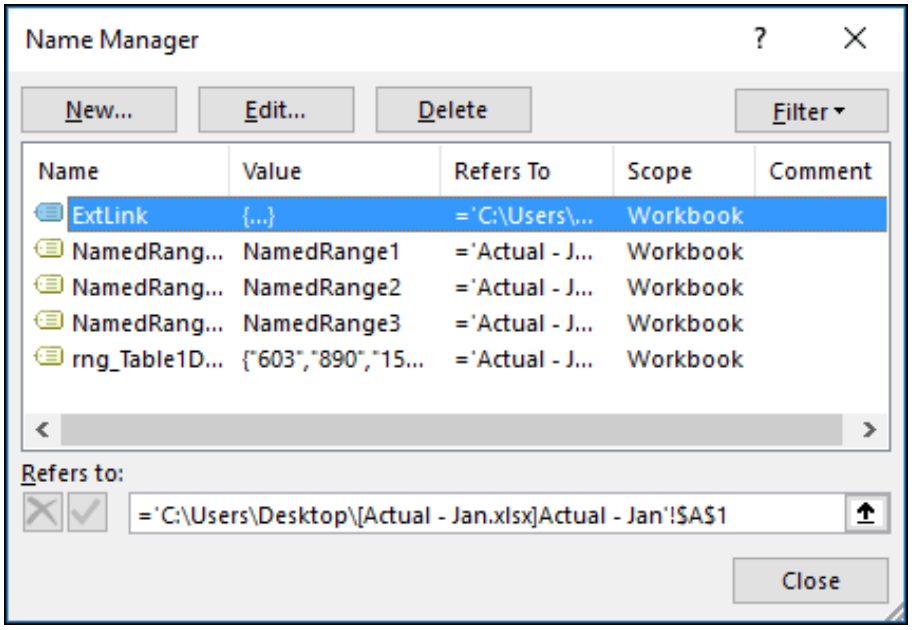
- press **CTRL + F** to launch the 'Find and Replace' dialog
- click 'Options'
- in the 'Find what' box, enter **.xl**
- in the 'Within' box, click 'Workbook'
- in the 'Look in' box, click 'Formulas'
- click 'Find All'
- in the list box that is displayed, look in the 'Formula' column for formulae that contain **.xl**. In this case, Excel found multiple instances of **Budget Master.xlsx**:



- to select the cell with a workbook link, click the cell address link for that row in the list box. As a tip, click any column header to sort the column and group all of the workbook links together.

To find workbook links used in defined names:

- on the 'Formulas' tab, in the 'Defined Names' group, click 'Name Manager'
- check each entry in the list and look in the 'Refers To' column for workbook links. Workbook links contain a reference to another workbook, such as **[Budget.xlsx]**.

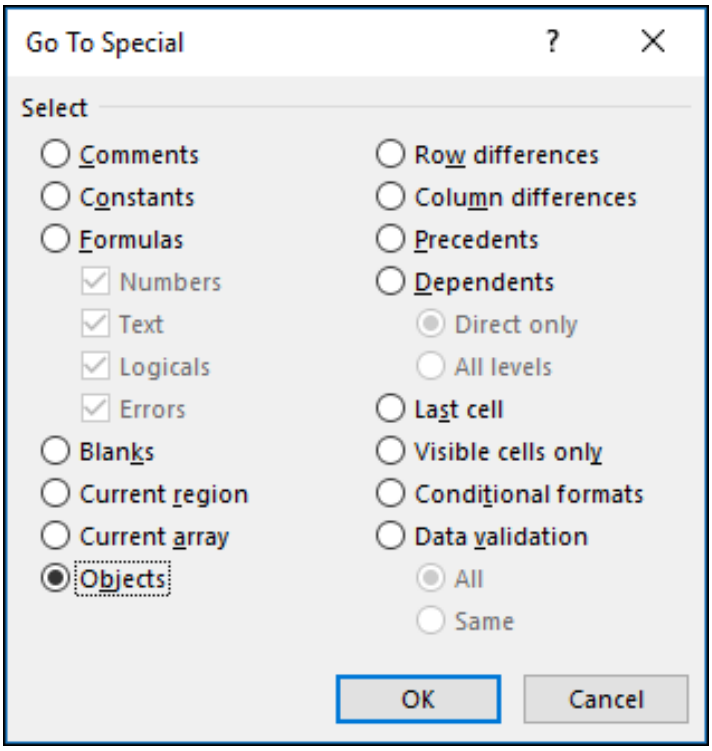


It should be noted:

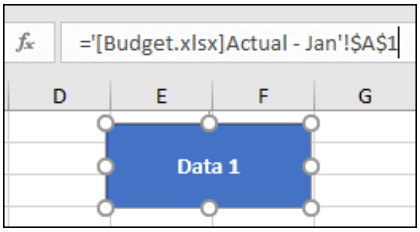
- you may click any column header to sort the column and group all of the workbook links together
- you can group multiple items with the **SHIFT** or **CTRL** keys and left-click if you want to delete multiple items at once.

To find workbook links used in objects, text boxes or shapes:

- Press **CTRL + G** (**F5** function key), the shortcut for the 'Go To' dialog, then click **Special -> Objects -> OK**. This will select all objects on the active worksheet:



- press the **TAB** key to move between each of the selected objects and then look in the Formula bar for a reference to another workbook, such as **[Budget.xlsx]**.

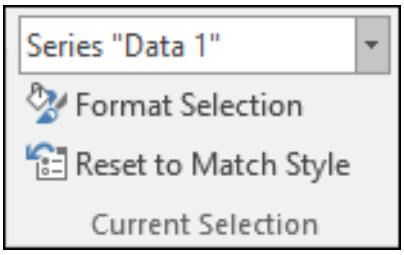


To find workbook links used in chart titles:

- click the chart title on the chart that you want to check
- In the Formula bar, look for a reference to another workbook, such as **[Budget.xls]**.

To find workbook links used in chart data series:

- select the chart that you want to check
- on the Layout tab, in the 'Current Selection' group, click the arrow next to the 'Chart Elements' box and then click the data series that you want to check

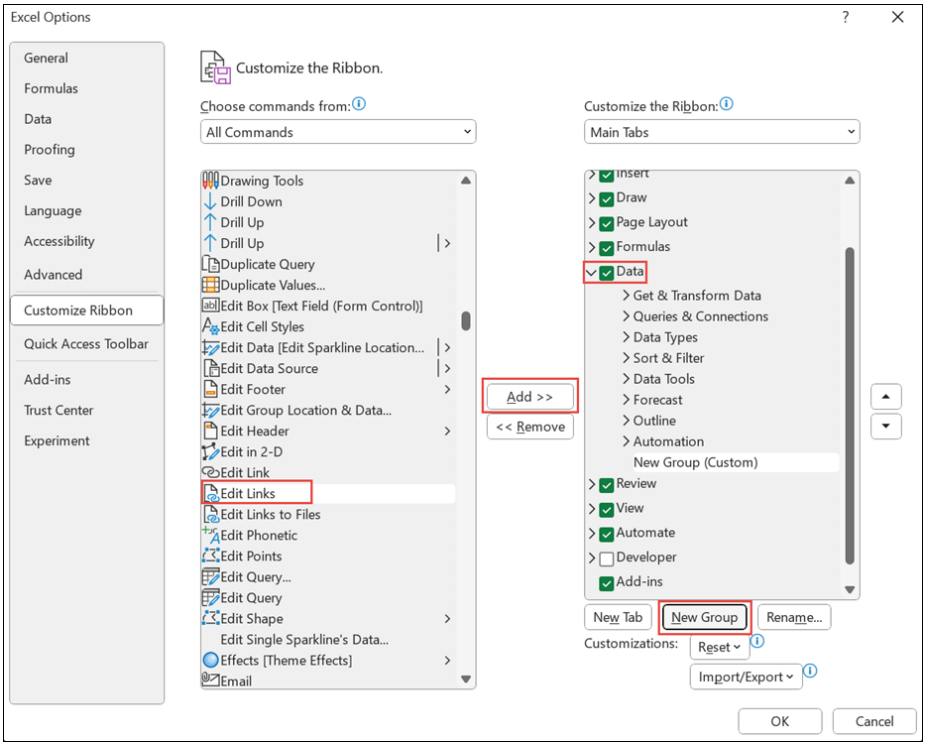


- in the Formula bar, look for a reference to another workbook, such as **[Budget.xls]** in the **SERIES** function.

USE THE LEGACY EDIT LINKS COMMAND

The legacy 'Edit Links' (**ALT + E + K**) command is replaced by this new 'Workbook Links' command. However, you can still get the old 'Edit Links' command back by adding the legacy 'Edit Links' command to your custom group in the Ribbon. However, you can't add the 'Edit Links' command to the 'Queries and Connections' group.

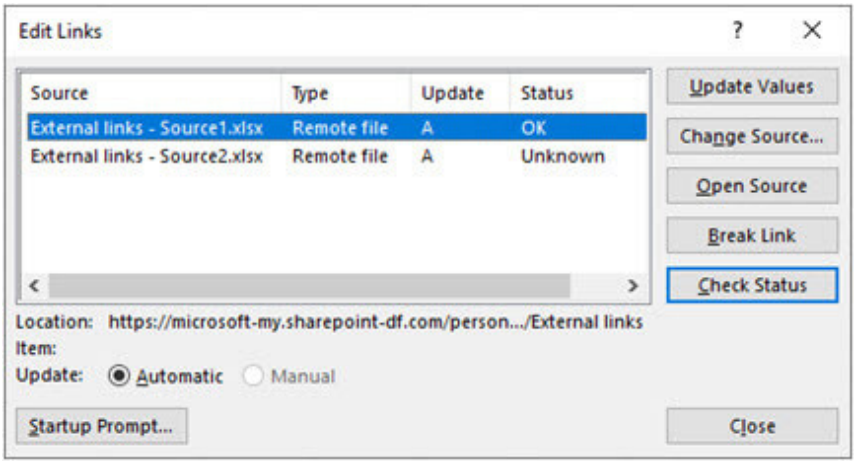
Using 'Customize the Ribbon', you need to create your custom group and only then you can add 'Edit Links' to your custom group in the Ribbon as follows:



- right-click the Ribbon and select ‘Customize the Ribbon’
- in the ‘Choose commands from:’ drop-down, select ‘All Commands’
- select ‘Edit Links’ and select the Data tab
- click ‘New Group’ and then select ‘Add’. ‘Edit Links’ is added to your customised group.

You may also add ‘Edit Links’ to the Quick Access Toolbar. The ‘Edit Links’ command is dimmed if your workbook does not contain workbook links.

When you open the ‘Edit Links’ dialog box, you're presented with several options for dealing with existing links. You can select individual workbooks with **CTRL + Click** or all of them with **CTRL + A**.



In the dialog box, there are various commands:

- **Update Values:** this will update all selected workbooks
- **Change Source:** this option is useful when you want to point existing links to another source. For instance, you might have a prior year workbook and need to point to a new workbook when a new year starts. Selecting ‘Change Source’ will launch a ‘File explorer’ dialog box, where you can browse to the new source workbook. If the workbook has more than one worksheet, you will be prompted to specify which one to link to – just click the sheet you want and click OK.

  It is possible to point a workbook back to itself by selecting it from the ‘Change Source’ dialog. This will sever any formula links to the originally linked source workbook
- **Open Source:** this will open the source workbook
- **Break Link:** when you break a link to a source, all formulae that use the source are converted to their current value. For example, the link

  =SUM([Budget.xlsx]Annual!C10:C25)

  would be converted to the sum of the values in the source workbook. Since this action cannot be undone, you may want to save a version of the file first.

  In the ‘Edit Links’ dialog box, in the Source list, click the link that you want to break. You may select individual workbooks with **CTRL + Click**, or all of them with **CTRL + A**. Then, click ‘Break Link’.

  If the link used a defined name, you may also want to delete the name. To delete a name, go to ‘Formulas’ tab, in the ‘Defined Names’ group, click ‘Name Manager’. In the ‘Name’ column, click the name that you want to delete and then click ‘Delete’. If you use an external data range, a parameter of a query may also use data from another workbook. You may want to check for and remove any of these types of links
- **Check Status:** this simply displays a notification in the ‘Edit Links’ pane whether a linked workbook is still a valid source. It should display OK, but if it doesn't then you'll need to check on the source workbook. In many cases, a source workbook may have been moved or deleted, cutting the link. If the workbook still exists, you can use the ‘Change Source’ option to relink the workbook.

Microsoft provided responses to some “Frequently Asked Questions” as part of this update:

Query	Suggested Answer
Why do I see new links in my workbook?	Previous versions of Excel hide the links in names that are not currently in use in the workbook. The links have always existed and are no longer hidden. To remove the link, select 'Break Links' for the selected workbook in the 'Workbook Links' pane.
Can I replace a single formula with its calculated value?	Yes. When you replace a formula with its value, Excel permanently removes the formula. If you accidentally replace a formula with a value and you want to restore the formula, select Home and choose Undo or press **CTRL + Z** (undo) immediately after you enter or paste the value. Press **CTRL + C** to copy the cell with the formula. Then, press **ALT + E + S + V** to paste the formula as its value, or select **Home -> Clipboard -> Paste -> Paste Values**.
What if I'm not connected to the source?	Select 'Don't Update' in the 'Unsafe links warning' dialog. Excel cannot update from a source that is not connected. For example, the source may be on a network and you may not be connected to that network.
What if I don't want my current data replaced with new data?	Select 'Don't Update' in the 'Unsafe links warning' dialog.
How can I update without it taking so long?	Select 'Don't Update' in the 'Unsafe links warning' dialog. If the data does not need to be the most current, you can save time by not updating all of the links. After opening the workbook, go to the Data tab and select 'Workbook Links' in the 'Queries & Connections' group, and then update the links only from the sources that you need.
How can I not see this prompt after someone else created the workbook?	Select 'Don't Update' in the 'Unsafe links warning' dialog and contact the workbook's owner. You can also investigate which links are in the workbook. On the Data tab, in the 'Queries & Connections' group, select 'Workbook Links'.
How can I not see the prompt again if I access the same way every time?	You can select and expand 'Refresh' settings at the top right corner of the 'Workbook Links' pane and select in a consistent way and avoid seeing the startup prompt for this workbook.
How do I not prompt for all workbooks I open, and update the links automatically?	This option affects every workbook opened on the computer. Users who open the workbook on another computer are not affected. On the File tab, select Options and choose Advanced. In the General section, clear 'Ask to update automatic links'. When this check box is cleared, the links are automatically updated and no alert is displayed.
How do I prompt in the same way for every user of this workbook?	This option affects all users of the workbook. If you choose to not update links and not to prompt, users of the workbook will not know that the data is out of date. Select and expand Refresh settings at the top right corner of the 'Workbook Link's pane and select the required option. It should be noted that you will still be notified if there are any broken links.
What if I'm using a parameter query?	A link to a parameter query cannot be updated unless the source workbook is open. 1. Select 'Don't Update' in the 'Unsafe links warning' dialog! 2. Close the destination workbook! 3. Open the source workbook! 4. Open the destination workbook! 5. Select 'Update'.
Why can't I choose 'Manual' as an update option for a specific external link?	Formula links are always set to Automatic.

GROUPBY

In Excel for Windows and Excel for Mac, the new **GROUPBY** function allows you to create a summary of your data formulaically. We know we rambled on about it last month, but for new readers and those that missed it, allow me to reiterate.

GROUPBY supports grouping along one axis and aggregating the associated values. For instance, if you had a table of sales data, you might generate a summary of sales by year, or by salesperson, or by category, or by…

In essence, it allows you to group, aggregate, sort and filter data based upon the fields you specify.

The syntax of the **GROUPBY** function is given by:

GROUPBY(row_fields, values, function, [field_headers], [total_depth], [sort_order], [filter_array])

It has the following arguments:

- **row_fields:** this is required and represents a column-oriented array or range that contains the values which are used to group rows and generate row headers. The array or range may contain multiple columns. If so, the output will have multiple row group levels
- **values:** this is also required and denotes a column-oriented array or range of the data to aggregate. The array or range may contain multiple columns. If so, the output will have multiple aggregations
- **function:** also required, this is an explicit or eta reduced lambda (*e.g.* **SUM**, **PERCENTOF**, **AVERAGE**, **COUNT**) that is used to aggregate **values**. A vector of lambdas may be provided. If so, the output will have multiple aggregations. The orientation of the vector will determine whether they are laid out row- or column-wise
- **field_headers:** this and the remaining arguments are all optional. This represents a number that specifies whether the **row_fields** and **values** have headers and whether field headers should be returned in the results. The possible values are:
    - **Missing:** Automatic
    - **0:** No
    - **1:** Yes and don't show
    - **2:** No but generate
    - **3:** Yes and show

  It should be noted that "Automatic" assumes the data contains headers based upon the **values** argument. If the first value is text and the second value is a number, then the data is assumed to have headers. Fields headers are shown if there are multiple row or column group levels
- **total_depth:** this optional argument determines whether the row headers should contain totals. The possible values are:
    - **Missing**: Automatic, with grand totals and, where possible, subtotals
    - **0:** No Totals
    - **1:** Grand Totals
    - **2:** Grand and Subtotals
    - **-1:** Grand Totals at Top
    - **-2:** Grand and Subtotals at Top

  It should be noted that for subtotals, fields must have at least two [2] columns. Numbers greater than two [2] are supported provided there are sufficient columns
- **sort_order:** again optional, this argument denotes a number indicating how rows should be sorted. Numbers correspond with the columns in **row_fields** followed by the columns in **values**. If the number is negative, the rows are sorted in descending / reverse order. A vector of numbers may be provided when sorting based upon only **row_fields**
- **filter_array:** the final optional argument, this represents a column-oriented one-dimensional array of Boolean values [1, 0] that indicate whether the corresponding row of data should be considered. It should be noted that the length of the array must match the length of **row_fields**.

To show how **GROUPBY** works, we took inspiration from Microsoft's data table:

Example Data

Table Used for Formulae

Year	Category	Item	Sales	Rating
2020	Components	Wheels	4,000	10%
2022	Components	Pedals	3,200	50%
2020	Components	Brakes	3,300	45%
2020	Clothing	Jerseys	1,100	10%
2020	Components	Saddles	500	85%
2020	Clothing	Jerseys	1,500	30%
2021	Accessories	Bike Racks	2,600	85%
2020	Bikes	Touring Bikes	1,100	30%
2022	Clothing	Tights	800	65%
2021	Clothing	Bib-Shorts	1,000	45%
2021	Accessories	Helmets	2,700	45%
2020	Clothing	Gloves	800	20%
2022	Clothing	Vests	1,100	30%
2021	Components	Brakes	1,100	100%
2022	Components	Handlebars	3,200	25%
2022	Accessories	Locks	400	55%
2021	Accessories	Tyres and Tubes	500	70%
2020	Components	Pedals	1,000	45%
2021	Accessories	Helmets	3,600	60%
2020	Bikes	Touring Bikes	200	55%
2021	Clothing	Gloves	4,000	100%
2020	Accessories	Locks	1,500	75%
2022	Bikes	Road Bikes	600	75%
2022	Clothing	Gloves	900	65%
2022	Components	Chains	100	10%
2022	Components	Chains	1,600	45%
2021	Bikes	Touring Bikes	2,400	70%

I have converted this data table into an Excel Table by selecting all the data and using **Insert -> Table** (**CTRL + T**) and calling the resultant Table **tbl**. Look, it's late as I write this and I have no imagination, OK!?

I can summarise my Table very simply using the formula

=GROUPBY(tbl[Category],tbl[Sales],SUM)

Description	Amount
Accessories	485,500
Bikes	495,800
Clothing	509,700
Components	493,200
Total	**1,984,200**

=GROUPBY(tbl[Category],tbl[Sales],SUM)

How easy is that!? Essentially, I am summing the sales (using the eta lambda **SUM**) by the **Category** field.

If you want to aggregate by more than one **row_field**, as stated above, this is possible. One way is to use **HSTACK**:

=GROUPBY(HSTACK(tbl[Year],tbl[Category]),tbl[Sales],SUM)

Year	Category	Sales
2020	Accessories	193,500
2020	Bikes	144,300
2020	Clothing	182,900
2020	Components	175,600
2021	Accessories	145,400
2021	Bikes	161,800
2021	Clothing	173,600
2021	Components	142,400
2022	Accessories	146,600
2022	Bikes	189,700
2022	Clothing	153,200
2022	Components	175,200
Total		**1,984,200**

=GROUPBY(HSTACK(tbl[Year],tbl[Category]),tbl[Sales],SUM)

This simply combines the **Year** and **Category** fields in the **tbl** Table, and then sums **Sales** across them. However, I think I prefer the **CHOOSECOLS** approach:

=GROUPBY(CHOOSECOLS(tbl,1,2),tbl[Sales],SUM)

Year	Category	Sales
2020	Accessories	193,500
2020	Bikes	144,300
2020	Clothing	182,900
2020	Components	175,600
2021	Accessories	145,400
2021	Bikes	161,800
2021	Clothing	173,600
2021	Components	142,400
2022	Accessories	146,600
2022	Bikes	189,700
2022	Clothing	153,200
2022	Components	175,200
Total		**1,984,200**

=GROUPBY(CHOOSECOLS(tbl,1,2),tbl[Sales],SUM)

Here, the idea is that I shall **SUM Sales** by columns 1 (**Year**) and 2 (**Category**) of the **tbl** Table. This might not seem as clear as the **HSTACK** alternative at first glance as you have to refer to the Table to identify what the columns are. However, stick with me. Let me make the formula more complex:

=GROUPBY(CHOOSECOLS(tbl,MATCH(F$12,tbl[#Headers],0),
MATCH(G$12,tbl[#Headers],0)),tbl[Sales],SUM)

Year	Category	Sales
2020	Accessories	193,500
2020	Bikes	144,300
2020	Clothing	182,900
2020	Components	175,600
2021	Accessories	145,400
2021	Bikes	161,800
2021	Clothing	173,600
2021	Components	142,400
2022	Accessories	146,600
2022	Bikes	189,700
2022	Clothing	153,200
2022	Components	175,200
Total		**1,984,200**

=GROUPBY(CHOOSECOLS(tbl,MATCH(F$12,tbl[#Headers],0),MATCH(G$12,tbl[#Headers],0)),tbl[Sales],SUM)

Looks horrible, yes? I have replaced the values 1 and 2 in the previous formula with

MATCH(F$12,tbl[#Headers],0)

and

MATCH(G$12,tbl[#Headers],0)

which return the positions in the **Headers** row of the Table **tbl**. Now, this may seem overkill but consider the following image:

Year	Category	Sales
2020	Accessories	193,500
2020	Bikes	144,300
2020	Clothing	182,900
2020	Components	175,600
2021	Accessories	145,400
2021	Bikes	161,800
2021	Clothing	173,600
2021	Components	142,400
2022	Accessories	146,600
2022	Bikes	189,700
2022	Clothing	153,200
2022	Components	175,200
Total		**1,984,200**

Brilliant. I have changed the background colour of the first two headers to yellow. Well no, it's a little more than that. I have used data validation dropdown lists (**ALT + D + L**) to create input headers!!

Year	Category	Sales
Year	Accessories	193,500
Category	Bikes	144,300
Item	Clothing	182,900
Sales	Components	175,600
Rating	Accessories	145,400
2021	Bikes	161,800
2021	Clothing	173,600
2021	Components	142,400
2022	Accessories	146,600
2022	Bikes	189,700
2022	Clothing	153,200
2022	Components	175,200
Total		**1,984,200**

Thus, if I change the selections, I have dynamic summarisations, such as

Category	Item	Sales
Accessories	Bike Racks	82,700
Accessories	Helmets	115,700
Accessories	Lights	64,900
Accessories	Locks	72,000
Accessories	Pumps	72,900
Accessories	Tyres and Tube	77,300
Bikes	Cargo Bikes	149,300
Bikes	Mountain Bikes	122,500
Bikes	Road Bikes	108,100
Bikes	Touring Bikes	115,900
Clothing	Bib-Shorts	52,300
Clothing	Caps	53,300
Clothing	Gloves	72,400
Clothing	Jerseys	79,000
Clothing	Shorts	67,000
Clothing	Socks	58,700
Clothing	Tights	64,700
Clothing	Vests	62,300
Components	Bottom Bracket	60,000
Components	Brakes	53,100
Components	Chains	65,000
Components	Handlebars	85,600
Components	Pedals	87,900
Components	Saddles	64,700
Components	Wheels	76,900
Total		**1,984,200**

or

Rating	Category	Sales
0.05	Accessories	20,800
0.05	Bikes	32,200
0.05	Clothing	25,800
0.05	Components	24,800
0.1	Accessories	25,800
0.1	Bikes	30,800
0.1	Clothing	28,200
0.1	Components	32,000
0.15	Accessories	32,000
0.15	Bikes	15,600
0.15	Clothing	26,500
0.15	Components	18,400
0.2	Accessories	26,600
0.2	Bikes	19,800
0.2	Clothing	17,900
0.2	Components	27,600
0.25	Accessories	22,100
0.25	Bikes	31,000
0.25	Clothing	19,500
0.25	Components	23,300
0.3	Accessories	25,000
0.3	Bikes	36,000
0.3	Clothing	24,700
0.3	Components	30,300
0.35	Accessories	26,800
0.35	Bikes	18,600
0.35	Clothing	16,200

Multiple summary statistics may be created similarly, or else you can simply connect them if the reporting fields are contiguous, *e.g.*

=GROUPBY(CHOOSECOLS(tbl,1,2),tbl[[Sales]:[Rating]],AVERAGE)

Year	Category	Sales	Rating
2020	Accessories	2,059	48%
2020	Bikes	1,659	51%
2020	Clothing	2,204	49%
2020	Components	2,116	52%
2021	Accessories	1,795	48%
2021	Bikes	1,926	49%
2021	Clothing	2,019	51%
2021	Components	2,064	54%
2022	Accessories	2,065	53%
2022	Bikes	2,062	54%
2022	Clothing	1,990	55%
2022	Components	1,947	50%
Total		**1,990**	**51%**

=GROUPBY(CHOOSECOLS(tbl,1,2),tbl[[Sales]:[Rating]],AVERAGE)

Here, **tbl[[Sales]:[Rating]]** may be used to specify the **values** as they are side by side.

Obviously, there are many more arguments to play with, but hopefully, you get the general idea, such as ranking the **Item** field in descending order by **Sales** using the formula

=GROUPBY(tbl[Item],tbl[Sales],SUM,,,-2)

Item	Sales
Cargo Bikes	149,300
Mountain Bikes	122,500
Touring Bikes	115,900
Helmets	115,700
Road Bikes	108,100
Pedals	87,900
Handlebars	85,600
Bike Racks	82,700
Jerseys	79,000
Tyres and Tubes	77,300
Wheels	76,900
Pumps	72,900
Gloves	72,400
Locks	72,000
Shorts	67,000
Chains	65,000
Lights	64,900
Tights	64,700
Saddles	64,700
Vests	62,300
Bottom Brackets	60,000
Socks	58,700
Caps	53,300
Brakes	53,100
Bib-Shorts	52,300
Total	**1,984,200**

=GROUPBY(tbl[Item],tbl[Sales],SUM,,,-2)

Indeed, the outputs summarised don't have to be numerical. A more comprehensive example summarising the **Items** field might look like this:

=GROUPBY(tbl[Category],tbl[Item],LAMBDA(x,ARRAYTOTEXT(SORT(UNIQUE(x)))))

Accessories	Bike Racks, Helmets, Lights, Locks, Pumps, Tyres and Tubes
Bikes	Cargo Bikes, Mountain Bikes, Road Bikes, Touring Bikes
Clothing	Bib-Shorts, Caps, Gloves, Jerseys, Shorts, Socks, Tights, Vests
Components	Bottom Brackets, Brakes, Chains, Handlebars, Pedals, Saddles, Wheels
Total	Bib-Shorts, Bike Racks, Bottom Brackets, Brakes, Caps, Cargo Bikes, Chains, Gloves, Handlebars, Helmets, Jerseys, Lights, Locks, Mountain Bikes, Pedals, Pumps, Road Bikes, Saddles, Shorts, Socks, Tights, Touring Bikes, Tyres and Tubes, Vests, Wheels

=GROUPBY(tbl[Category],tbl[Item],LAMBDA(x,ARRAYTOTEXT(SORT(UNIQUE(x)))))

PIVOTBY

In Excel for Windows and Excel for Mac, the **PIVOTBY** function allows you to create a summary of your data via a formula too, akin to a formulaic PivotTable. It supports grouping along two axes and aggregating the associated values. For instance, if you had a table of sales data, you might generate a summary of sales by state and year.

It should be noted that **PIVOTBY** is a function that returns an array of values that can spill to the grid. Furthermore, at this stage, not all features of a PivotTable appear to be replicable by this function.

The syntax of the **PIVOTBY** function is:

PIVOTBY(row_fields, col_fields, values, function, [field_headers], [row_total_depth], [row_sort_order], [col_total_depth], [col_sort_order], [filter_array])

It has the following arguments:

- **row_fields:** this is required and represents a column-oriented array or range that contains the values which are used to group rows and generate row headers. The array or range may contain multiple columns. If so, the output will have multiple row group levels
- **col_fields:** also required and represents a column-oriented array or range that contains the values which are used to group columns and generate column headers. The array or range may contain multiple columns. If so, the output will have multiple column group levels
- **values:** this is also required and denotes a column-oriented array or range of the data to aggregate. The array or range may contain multiple columns. If so, the output will have multiple aggregations
- **function:** also required, this is an explicit or eta reduced lambda (*e.g.* **SUM**, **PERCENTOF**, **AVERAGE**, **COUNT**) that is used to aggregate **values**. A vector of lambdas may be provided. If so, the output will have multiple aggregations. The orientation of the vector will determine whether they are laid out row- or column-wise

- **field_headers:** this and the remaining arguments are all optional. This represents a number that specifies whether the **row_fields**, **col_fields** and **values** have headers and whether field headers should be returned in the results. The possible values are:
    - **Missing:** Automatic
    - **0:** No
    - **1:** Yes and don't show
    - **2:** No but generate
    - **3:** Yes and show

  It should be noted that "Automatic" assumes the data contains headers based upon the **values** argument. If the first value is text and the second value is a number, then the data is assumed to have headers. Fields headers are shown if there are multiple row or column group levels
- **row_total_depth:** this optional argument determines whether the row headers should contain totals. The possible values are:
    - **Missing**: Automatic, with grand totals and, where possible, subtotals
    - **0:** No Totals
    - **1:** Grand Totals
    - **2:** Grand and Subtotals
    - **-1:** Grand Totals at Top
    - **-2:** Grand and Subtotals at Top

  It should be noted that for subtotals, **row_fields** must have at least two [2] columns. Numbers greater than two [2] are supported provided **row_field** has sufficient columns
- **row_sort_order:** again optional, this argument denotes a number indicating how rows should be sorted. Numbers correspond with the columns in **row_fields** followed by the columns in **values**. If the number is negative, the rows are sorted in descending / reverse order. A vector of numbers may be provided when sorting based upon only **row_fields**
- **col_total_depth:** this optional argument determines whether the column headers should contain totals. The possible values are:
    - **Missing**: Automatic, with grand totals and, where possible, subtotals
    - **0:** No Totals
    - **1:** Grand Totals
    - **2:** Grand and Subtotals
    - **-1:** Grand Totals at Top
    - **-2:** Grand and Subtotals at Top

  It should be noted that for subtotals, **col_fields** must have at least two [2] columns. Numbers greater than two [2] are supported provided **col_field** has sufficient columns
- **col_sort_order:** again optional, this argument denotes a number indicating how they should be sorted. Numbers correspond with the columns in **col_fields** followed by the columns in **values**. If the number is negative, these are sorted in descending / reverse order. A vector of numbers may be provided when sorting based upon only **col_fields**
- **filter_array:** the final optional argument, this represents a column-oriented one-dimensional array of Boolean values [1, 0] that indicate whether the corresponding row of data should be considered. It should be noted that the length of the array must match the length of **row_fields** and **col_fields**.

Similar in many ways to **GROUPBY**, **PIVOTBY** is fairly straightforward to use:

=PIVOTBY(tbl[Category],tbl[Year],tbl[Sales],AVERAGE)

	2020	2021	2022	Total
Accessories	2,059	1,795	2,065	1,974
Bikes	1,659	1,926	2,062	1,885
Clothing	2,204	2,019	1,990	2,072
Components	2,116	2,064	1,947	2,038
Total	**2,007**	**1,948**	**2,014**	**1,990**

=PIVOTBY(tbl[Category],tbl[Year],tbl[Sales],AVERAGE)

You can get more imaginative and sort in descending order by the **AVERAGE** of **Rating**, *viz.*

=PIVOTBY(tbl[Item],tbl[Year],tbl[Rating],AVERAGE,,,-2)

	2020	2021	2022	Total
Brakes	46%	61%	65%	59%
Locks	50%	58%	71%	56%
Pumps	70%	41%	56%	56%
Shorts	52%	52%	64%	55%
Vests	54%	51%	60%	55%
Touring Bikes	54%	51%	59%	55%
Pedals	58%	57%	50%	54%
Bib-Shorts	43%	48%	69%	54%
Gloves	49%	64%	52%	53%
Road Bikes	46%	54%	58%	53%
Cargo Bikes	56%	51%	51%	53%
Saddles	63%	43%	47%	52%
Tights	56%	53%	46%	51%
Handlebars	53%	52%	47%	51%
Caps	60%	39%	51%	51%
Chains	52%	49%	51%	51%
Socks	50%	45%	56%	50%
Wheels	41%	67%	45%	50%
Tyres and Tubes	46%	54%	46%	50%
Lights	47%	46%	49%	47%
Mountain Bikes	48%	42%	48%	46%
Bottom Brackets	48%	42%	45%	46%
Helmets	47%	43%	47%	45%
Bike Racks	31%	51%	54%	44%
Jerseys	27%	53%	45%	44%
Total	50%	50%	53%	51%

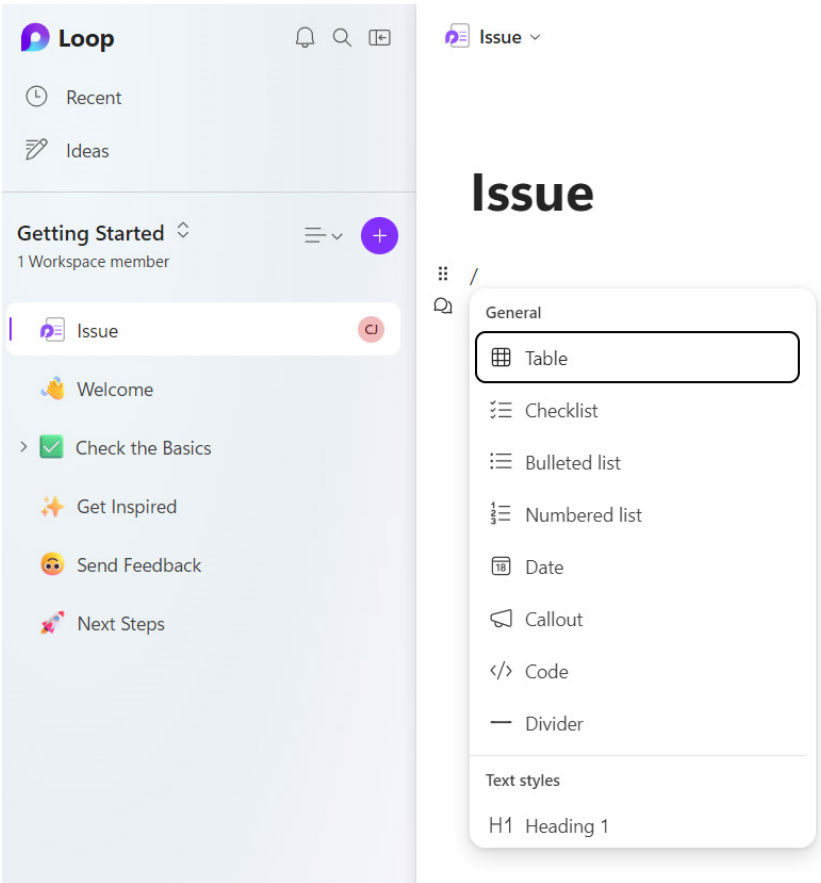
=PIVOTBY(tbl[Item],tbl[Year],tbl[Rating],AVERAGE,,,-2)

Export Loop Tables to Excel

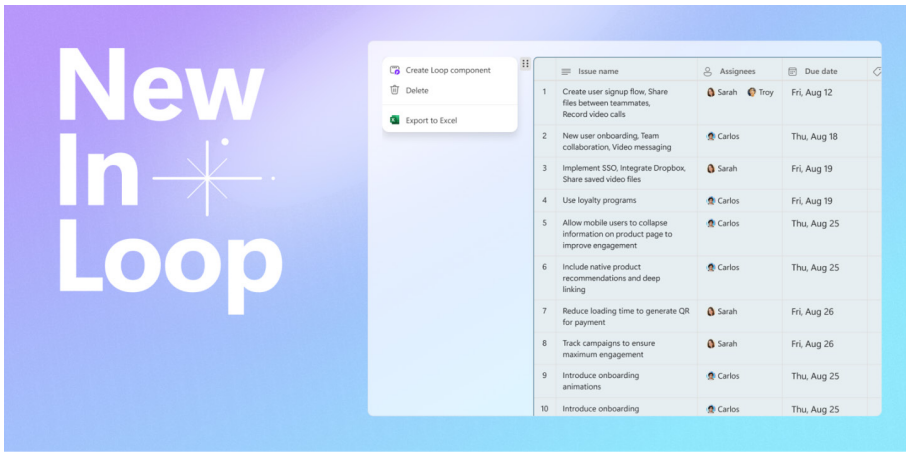
For Excel for Windows and Excel for Mac, there is a final feature this month: 'Export Loop Tables to Excel'. When you're collaborating as a team on a project, you may choose to add Loop tables to gather data and perform calculations. You can now export the tables to applications such as Excel so you can leverage its capabilities to perform any needed calculations.

To ensure compatibility, Loop's table data type is mapped to the most relevant format in Excel.

From your browser, go to loop.microsoft.com and then open an existing table or create one and add relevant information to your project.



Select the six-dot menu on the table and select ‘Export to Excel’.



After a few moments, the Excel worksheet is created in your OneDrive and in SharePoint.

To use this feature, you’ll need:

- an active Microsoft 365 subscription
- a Microsoft 365 commercial license, either E3 or E5.

This feature is available to all Microsoft 365 subscribers at https://loop.microsoft.com.

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 2308 (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

Page 2 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
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

Page 3 of 4

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

Page 4 of 4

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: MID

The **MID** function returns a specific number of characters from a text string, starting at the position you specify, based upon the number of characters you specify.

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

MID(text, start_number, number_of_characters)

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

- **text:** this is required and represents the text string that contains the characters you want to extract
- **start_number:** this is also required and specifies the position of the first character you want to extract from **text**. The first character in
- **text** has **start_number** 1, and so on
- **number_of_characters:** this argument is mandatory too and specifies the number of characters you want **MID** to return from the **text**.

It should be further noted that:

- if **start_number** is greater than the length of **text**, **MID** returns "" (empty text)
- if **start_number** is less than the length of **text**, but **start_number** plus **number_of_characters** exceeds the length of **text**, **MID** returns the characters up to the end of **text**
- if **start_number** is less than 1, **MID** returns the *#VALUE!* error value
- if **number_of_characters** is negative, **MID** returns the *#VALUE!* error value.

Please see our examples below:

	A	B	C
1	Data		
2	SumProduct		
3	31 Dec 20		
4			
5			
6	Formula	Description	Result
7	=MID(A2,1,3)	Returns three characters from the string in **A2**, starting at the first character	Sum
8	=MID(A2,4,20)	Returns 20 characters from the string in **A2**, starting at the fourth character. Since the number of characters to return (20) is greater than the length of the string from this point on (7), all characters, beginning with the fourth, are returned. No empty characters (spaces) are added to the end	Product
9	=MID(A3,4,3)	Returns three characters from the string in **A3**, starting at the fourt th character. Since the number of characters to return (3) is greater than the length of the string remaining (2), all characters, beginning with the fourth, are returned. No empty characters (spaces) are added to the end. Do note that this is a date and the underlying serial number is 44196, so this returns "96" not "Dec"	96
10			

The A to Z of Excel Functions: MIDB

The **MIDB** function returns a specific number of characters from a text string, starting at the position you specify, based upon the number of bytes you specify.

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

MIDB(text, start_number, number_of_bytes)

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

- **text:** this is required and represents the text string that contains the characters you want to extract
- **start_number:** this is also required and specifies the position of the first character you want to extract from **text**. This is based upon bytes in **text**, so that the first byte has **start_number** 1, and so on
- **number_of_characters:** this mandatory argument specifies the number of bytes you want **MIDB** to return from the **text**.

It should be further noted that:

- if **number_of_bytes** is negative, **MIDB** returns the *#VALUE!* error value
- this function may not be available in all languages. **MIDB** counts two (2) bytes per character only when a DBCS language is set as the default language (the languages that support DBCS include Japanese, Chinese (Simplified), Chinese (Traditional), and Korean
- otherwise, **MIDB** behaves the same as **MID**, counting one (1) byte per character.

For example, **=MIDB("中国香港",3,2)** is equal to "国". **MIDB** returns the second character only, because each character is counted as two (2) bytes and the second character begins at the third byte.

However, **=MID("中国香港",3,2)** is equal to **"香港"** as **MID** returns the next two (2) characters from the third character onwards, because each character is counted as one (1). **MID** returns these two characters no matter what the default language setting is on your computer.

The A to Z of Excel Functions: MIN

MIN (and **MAX**) may be familiar to many Excel users, but it is still probably worth performing a recap. **MIN** does exactly what it says on the tin – it calculates the minimum value in a range:

	D	E	F	G	H	I	J
11							
12			Values		MIN		
13			304		78	=MIN(F13:F27)	
14			128				
15			223				
16			78				
17			192				
18			420				
19			233				
20			150				
21			102				
22			460				
23			181				
24			79				
25			83				
26			101				
27			364				
28							

Modellers frequently make mistakes with **MIN**:

	Values		MIN	
35	(190)		(190)	*Doesn't treat blanks as zeros*
36	96		-	*=MIN(F36,) - does treat blanks in formula as zeros*
37	(209)		#VALUE!	*=MIN(F37,"") - empty strings are treated as text; MIN cannot evaluate text strings*

These examples highlight that you should be careful with using blanks both in the ranges and in the formulae themselves. Regular readers will notice I often omit zeros in my formulae but with **MIN** (and **MAX**) I will often take care to explicitly use zeros.

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

MIN(number1, [number2, …])

The **MIN** function has the following argument(s):

- **number1**, **number2**, …**: number1** is required, but subsequent numbers are optional
- you may have up to 255 arguments in order to find the minimum value.

It should be noted that:

- arguments may be numbers, or names, arrays or references that contain numbers
- logical values and text representations of numbers that you type directly into the list of arguments are counted
- if an argument is an array or reference, only numbers in that array or reference are used; empty cells, logical values or text in the array or reference are ignored
- if the arguments contain no numbers, **MIN** returns zero (0)
- arguments that are error values or text that cannot be translated into numbers cause errors
- if you want to include logical values and text representations of numbers in a reference as part of the calculation, use the **MINA** function instead.

There is a highly relevant finance example combining **MAX** and **MIN** functions (**MAX** is the similar function that returns the *maximum* value in a range), namely calculating the maximum dividend allowable for a particular period.

Dividends may only be paid out of what are known as distributable reserves (this is a bit of an oxymoron as dividends are also known as **distributions**). Revaluation reserves, share premium accounts, capital redemption reserves are all non-distributable. Essentially, dividends may only be paid out of the current year's Net Profit After Tax (NPAT) and the aggregation of all previous year's profits after past distributions, Retained Earnings. Dividends may not make the Balance Sheet's Total Equity become negative. This shows insolvency and this sort of distribution is illegal in most territories. Given non-distributable reserves may not become negative allow me to concentrate simply on NPAT and Retained Earnings here.

To derive the maximum dividend allowable, let me consider some scenarios. Let's imagine the following scenario:

Retained Earnings	100
NPAT	40
Maximum Dividend	**140**

It isn't rocket science that if Retained Earnings and NPAT are both positive, then the maximum dividend allowed is the sum of the two.

Retained Earnings	100
NPAT	(40)
Maximum Dividend	**60**

If NPAT is negative, but Retained Earnings are positive and exceed the NPAT figure, then the maximum dividend allowed is the net of the two figures. Should the net be negative, no dividend is allowed.

Retained Earnings	(100)
NPAT	40
Maximum Dividend	**40**

Here is the one that often surprises people. If Retained Earnings is negative but NPAT is positive, regardless of whether the net is positive or negative, the maximum dividend allowed is the NPAT amount. This may seem incomprehensible upon first thought, but it is typically dependent upon two conditions:

1. The company's auditors must sign off on it. This is to ensure the company is still seen to be a going concern (*i.e.* it can still continue to operate and trade its way out of any short-term difficulties).
2. The shareholders must vote for it. Almost as hilarious as when Members of Parliament solemnly vote for their 50% pay rise each year.

If you think about it some more, this makes sense. Remember, dividends cannot be paid if the company is insolvent. The auditors check to see whether the company can "afford" it for other reasons. But if you don't allow this scenario how would anyone ever attract share capital for a start-up company? A new company may have to provide for certain factors which may never come to fruition. A large non-current asset may have to be written off as not fit for purpose if a company's strategy changes without any cash consequence. Is it acceptable that shareholders have to wait 10 years for the Retained Earnings losses to be covered even if the business is hugely profitable in the meantime? No, and this is precisely why this is the rule in many territories.

The next scenario is more obvious:

Retained Earnings	(100)
NPAT	(40)
Maximum Dividend	-

With both a negative NPAT and Retained Earnings, there is no leeway now. These scenarios seem to suggest the following formula:

=MAX(NPAT + Retained Earnings, NPAT, 0)

This allows for the above scenarios. The check to ensure that the value is non-negative (*i.e.* the inclusion of zero in the **MAX** formula) is so that shareholders do not get asked to pay a dividend to the company. I can't imagine that would go down too well.

We are not done yet though. Let's go back to the penultimate scenario but now consider the cash position as well:

Retained Earnings	(100)
NPAT	40
Cash Available	30
Maximum Dividend	**30**

Here, the Cash Available is the total amount of cash available to pay the dividend. Technically, this includes any cash reserves built up over time, but many companies only consider the cash position for the period the dividend relates to (this is the scenario I shall be modelling in the case study shortly). This seems to suggest the formula:

=MIN(MAX(NPAT + Retained Earnings, NPAT, 0), Cash Available)

Let me just check with slightly revised numbers:

Retained Earnings	(100)
NPAT	40
Cash Available	(30)
Maximum Dividend	(30)

Here, the Cash Available is the total amount of cash available to pay the dividend. Technically, this includes any cash reserves built up over time, but many companies only consider the cash position for the period the dividend relates to (this is the scenario I shall be modelling in the case study shortly). This seems to suggest the formula:

=MIN(MAX(NPAT + Retained Earnings, NPAT, 0), Cash Available)

Let me just check with slightly revised numbers:

	E	F	G	H	I	J	K	L	M
69									
70		Retained Earnings			(100)				
71		NPAT			40				
72									
73		Cash Available			(30)				
74									
75		**Maximum Dividend**			-		*=MAX(MIN(MAX(I70+I71,I71),I73),)*		
76									

Ah yes, the wonderful **MAX(MIN(MAX))** formula. It may not be the prettiest formula in the world, but the point is, it gives the right number.

The A to Z of Excel Functions: MINA

Is there a **MINA** function in Excel..? The **MINA** function returns the largest value in a list of arguments. It has the following syntax:

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

where:

- **value1** is required and represents the first number argument for which you want to find the smallest value
- **value2, ...** are optional arguments. These are numerical arguments 2 to 255 for which you want to find the smallest value.

It should be noted that:

- arguments may be any of the following: numbers; names, arrays or references that contain numbers; text representations of numbers; logical values, such as TRUE and FALSE, in a reference
- logical values and text representations of numbers that you type directly into the list of arguments are counted
- if an argument is an array or reference, only values in that array or reference are used. Empty cells and text values in the array or reference are ignored
- arguments that are error values or text that cannot be translated into numbers cause errors
- arguments that contain TRUE evaluate as 1; arguments that contain text or FALSE evaluate as zero (0)
- if the arguments contain no values, **MINA** returns zero (0)
- if you do not want to include logical values and text representations of numbers in a reference as part of the calculation, use the **MIN** function instead.

Please consider the following example:

	A	B	C
1	Data		
2	FALSE		
3	0.2		
4	0.4		
5	0.6		
6	0.8		
7	TRUE		
8			
9			
10	Formula	Description	Result
11	=MINA(A2,A3:A6,A7)	The smallest number in the range A2:A7. Because FALSE evaluates to 0, this is the smallest value.	0.0

The A to Z of Excel Functions: MINIFS

The **MINIFS** function returns the minimum value among cells specified by a given set of conditions or criteria. It has the following syntax:

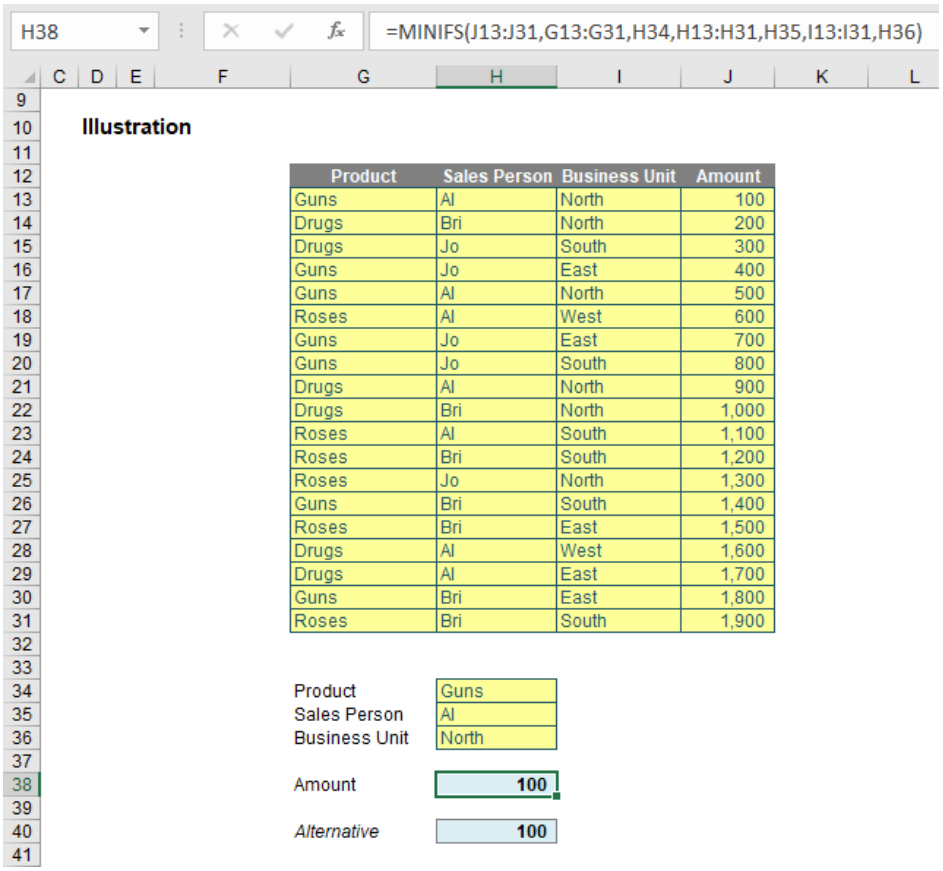
MINIFS(min_range, criterion_range1, criterion1, [criterion_range2, criterion2], ...)

where:

- **min_range** is the actual range of cells in which the minimum is to be determined
- **criterion_range1** is the set of cells to evaluate with the criterion specified
- **criterion1** is the criterion in the form of a number, expression or text that defines which cells will be evaluated as a minimum
- **criterion_range2** (onwards) and **criterion2** (onwards) are the additional ranges and their associated criteria. 126 range / criterion pairs may be specified. All ranges must have the same dimensions otherwise the function returns an *#VALUE!* error.

It should be noted that:

- the size and shape of the **min_range** and **criteria_rangeN** arguments must be the same, otherwise these functions return the *#VALUE!* error.



This example ***is*** preferable to its standard Excel counterpart:

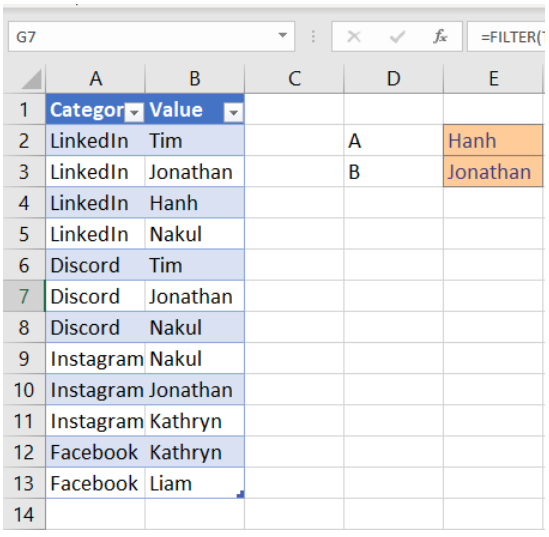
{=MIN(IF(G13:G31=H34,IF(H13:H31=H35,IF(I13:I31=H36,J13:J31))))}

Array formulae (see https://www.sumproduct.com/thought/array-of-light.html for more information) are cumbersome and not readily understood, which is why **MINIFS** may be a highly viable alternative.

More Excel Functions next month.

Beat the Boredom Suggested Solution

This month, we want to check if a pair of values are of the same category. Earlier in this newsletter, we considered the following dataset, which shows the social media channels certain made up people use:



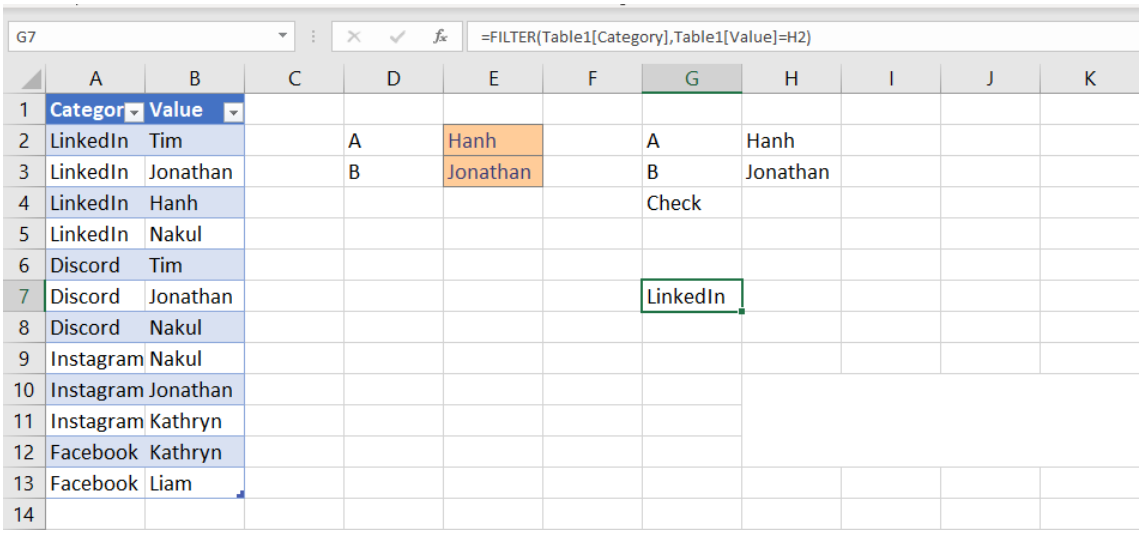
Imagine that this data set goes on to for 1,000 rows. Using this data set as a proxy, the challenge was to identify if a pair of users had a common social media channel?

The challenge was "simply" this: could you write a formula to check whether a pair use a common social media channel. This had to be coded in one cell. It wasn't our hardest ever challenge!

Suggested Solution

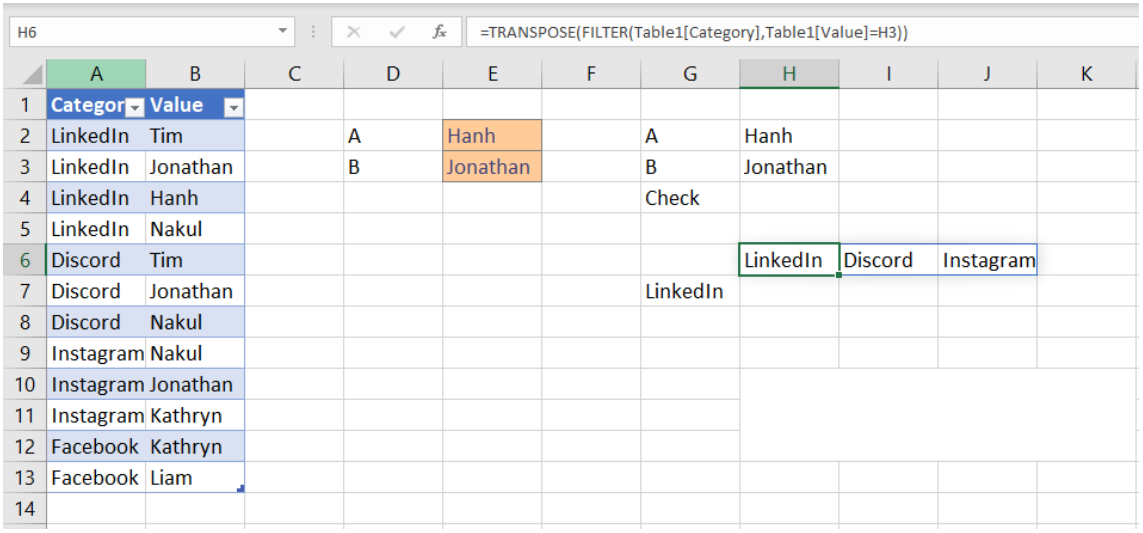
To begin, we need to identify what platforms each person uses. We will create two lists of social media channel that are to be used to compare both users. We will first start with the 'User A' (Hanh). Using the **FILTER** dynamic array function, we will create a dynamic array in **G7** using the following formula:

=FILTER(Table1[Category],Table1[Value]=H2)



As expected, the only platform 'User A' uses is LinkedIn. Now moving to 'User B' (Jonathan), let's transpose the list of social media channels to create an array in **H6** using

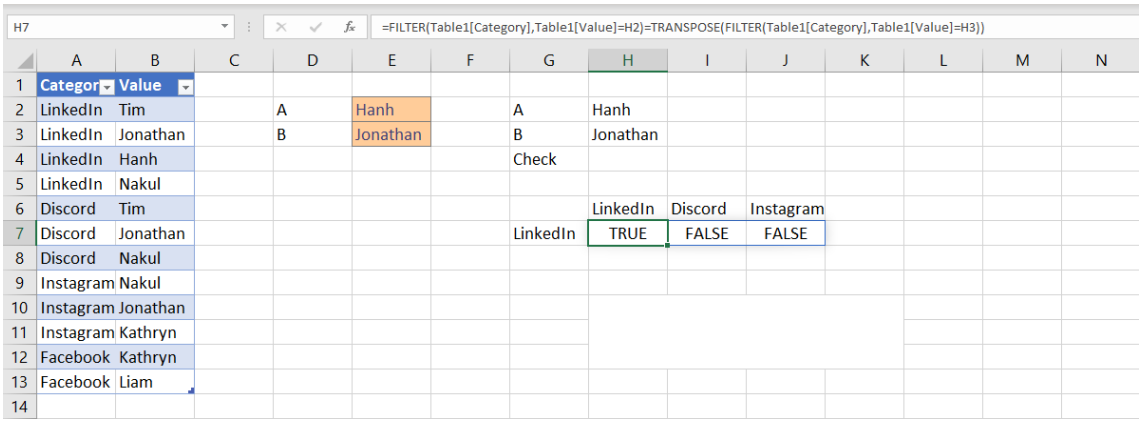
=TRANSPOSE(FILTER(Table1[Category],Table1[Value]=H3))



From the data, 'User B' uses LinkedIn, Discord, and Instagram.

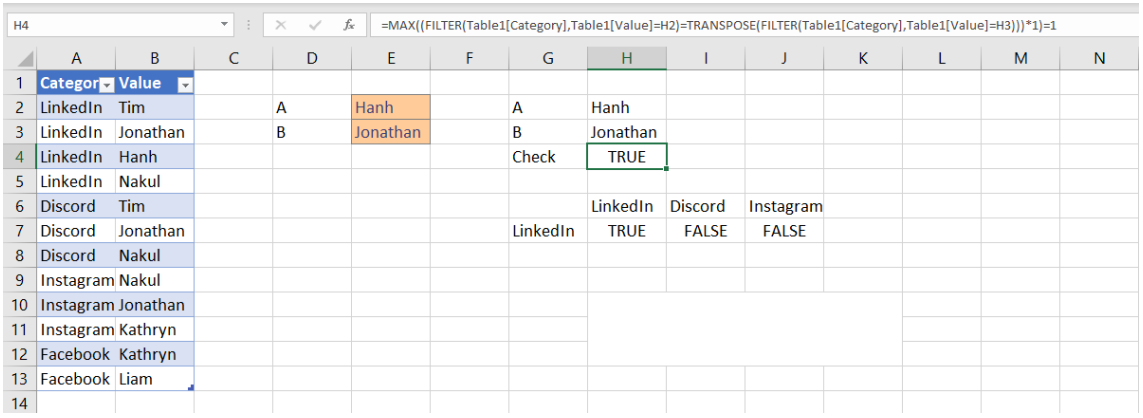
Now the question is, how do we pair up the list? The answer is simple: we just check if list A equals list B in **H7** by using the following formula:

=FILTER(Table1[Category],Table1[Value]=H2)=TRANSPOSE(FILTER(Table1[Category],Table1[Value]=H3))

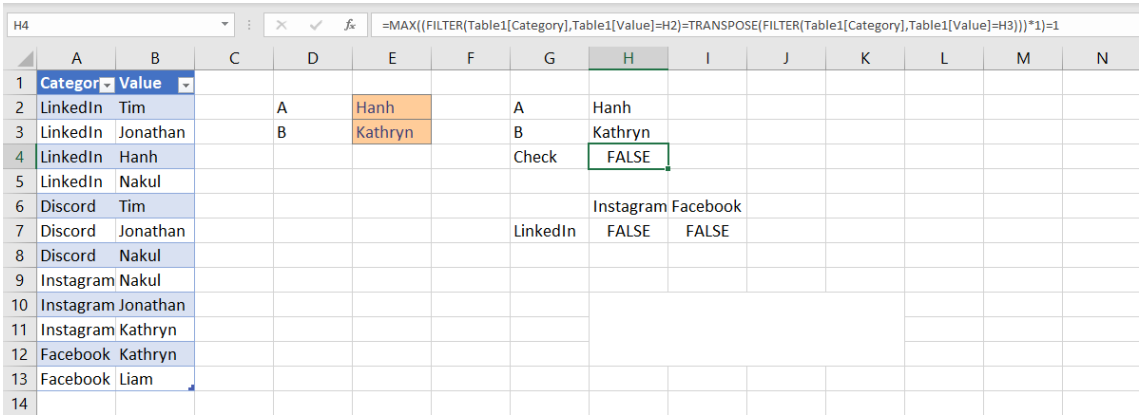


Now we finally check if the users share a social media channel in **H4** using

=MAX((FILTER(Table1[Category],Table1[Value]=H2)=TRANSPOSE(FILTER(Table1[Category],Table1[Value]=H3)))*1)=1



There it is! For verification, we will try a different user by changing cell **E3**. We get the following:



Thankfully, it works! You have a check that displays if a pair of users are active on a social media channel.

Until next month.

Upcoming SumProduct Training Courses

Location	Course	Date	Date	Duration	Duration
Sydney Australia	Excel Tips and Tricks	8 Jan 2024	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	1 Day
Sydney Australia	Financial Modelling	9 - 10 Jan 2024	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	2 Days
Sydney Australia	Power Pivot, Power Query and Power BI	11 - 12 Jan 2024	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	2 Days
Melbourne Australia	Excel Tips and Tricks	5 Feb 2024	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	1 Day
Melbourne Australia	Financial Modelling	6 - 7 Feb 2024	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	2 Days
Melbourne Australia	Power Pivot, Power Query and Power BI	8 - 9 Feb 2024	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	2 Days
Sydney Australia	Power Pivot, Power Query and Power BI	19 - 20 Feb 2024	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	2 Days
Sydney Australia	Excel Tips and Tricks	21 Feb 2024	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	1 Day
Sydney Australia	Financial Modelling	22 - 23 Feb 2024	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	2 Days
Melbourne Australia	Power Pivot, Power Query and Power BI	18 - 19 Mar 2024	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	2 Days
Melbourne Australia	Excel Tips and Tricks	20 Mar 2024	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	1 Day
Melbourne Australia	Financial Modelling	21 - 22 Mar 2024	09:00-17:00 AEDT	(-1 day) 22:00-17:00 GMT	2 Days

Key Strokes

Each newsletter, we'd like to introduce you to useful keystrokes you may or may not be aware of. We've started going through the alphabet actions. **R** me hearties, it's a pirate's list for us this month…

Keystroke	What it does
Return	Enter value and move down (depending upon Excel setting)
ALT + Return	Redo
CTRL + Return	Fill value in edited cell into all cells and do not move
SHIFT + Return	Enter value and move up (depending upon Excel setting)
CTRL + SHIFT + Return	Fill value in edited cell into all cells and do not move
Right Arrow	Move right one cell
ALT + Right Arrow	Forward (hyperlink navigation)
CTRL + Right Arrow	Select the last cell in the area right
SHIFT + Right Arrow	Extend selection right one cell
ALT + SHIFT + Right Arrow	Group
CTRL + ALT + Right Arrow	Intel chipset: turn screen -90 degrees; else: move active cell to next non-adjacent area within selection
CTRL + SHIFT + Right Arrow	Extend selection down to last cell in area right

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