

# Sum Product

NEWSLETTER #115 - June 2022

[www.sumproduct.com](http://www.sumproduct.com) | [www.sumproduct.com/thought](http://www.sumproduct.com/thought)



*Is Excel starting to become more consistent?* Office Scripts may now be run in Excel Desktop, Power Query is more serviceable on the web, and even Excel for Mac gets improvements! Goodness, it's like it's similar software. You never know, we might get chart labels to work properly before 2098 at this rate! 😊

This month, we take to task the most Important Function in financial modelling (IF) and combine it with all the usual hangers on: there is another Beat the Boredom Challenge, plus our usual articles on Charts & Dashboards, Visual Basics, Power Pivot Principles, Power Query Pointers and Power BI Updates. We take a LOG of more IMAGINARY Excel Functions this month, and we even get DOWN and party with some Keyboard Shortcuts.

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

*Liam Bastick*, Managing Director, SumProduct



## Office Scripts Can Now Run on Windows via a Button

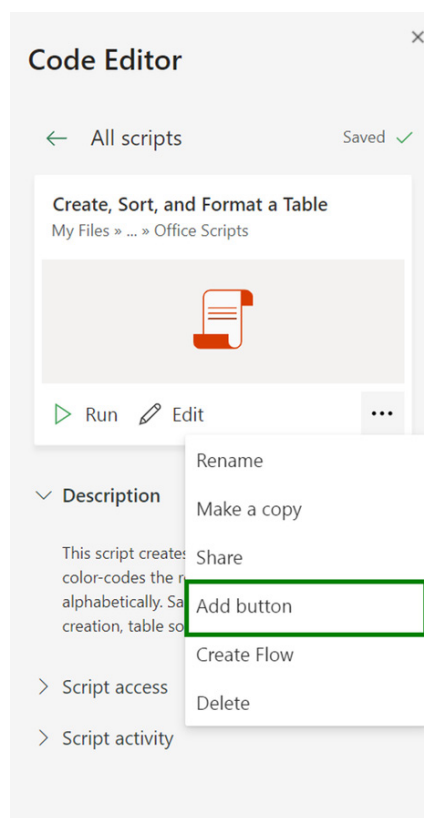
It was too late in the day to announce this in last month's newsletter, but the end of April brought new automation tools to Excel Desktop – *sort of*. Indeed, it is now possible to run Office Scripts in Excel for Windows through pressing a button within the workbook. This expands on the recently introduced Script buttons feature, meaning you can run the same button in both the browser and Windows application.

With Script buttons in Windows, Microsoft is now one step closer towards making Office Scripts an automation solution you can use to manage your workflow anytime and anywhere. Whilst you still need to use Excel on

the web to create and manage your scripts, at least it's been recognised that Excel users use a variety of platforms to complete their tasks!

To create a button in your workbook:

- From the Automate tab in Excel on the web, select a script
- Go to the More options (...) menu in either the 'Script Details', 'All Scripts' or 'Code Editor' pane
- Select the Add button.



To run the button in Windows:

- open a file containing the button in Excel on Windows
- select the button in the workbook.

The button is currently available to all E3/E5 users on the Current channel.

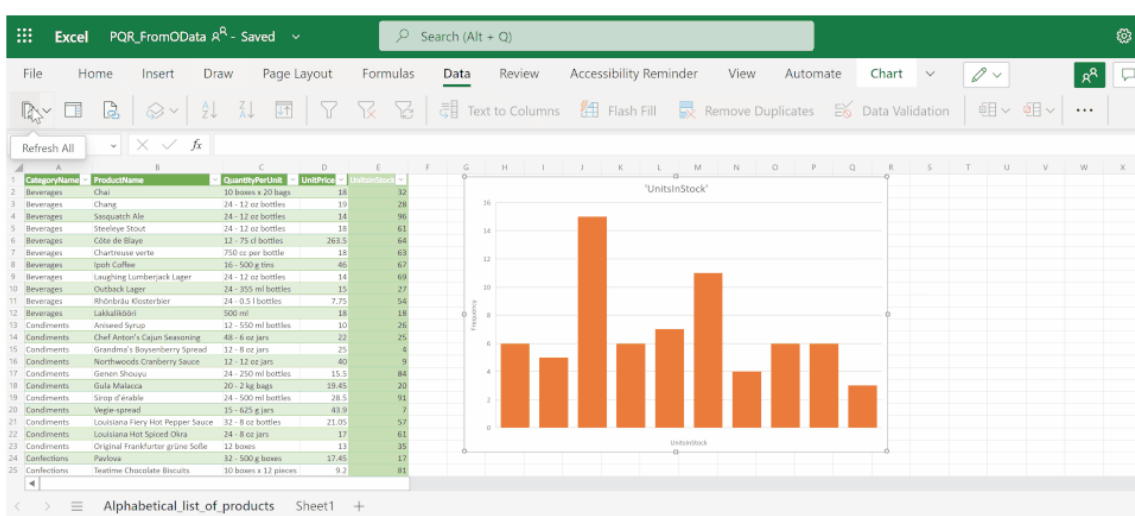
## Power Query Refresh is Now Generally Available for Selected Data Sources in Excel for the Web

Power Query Refresh is now Generally Available in Excel for the web for queries sourcing data from the current workbook and anonymous OData feeds. Slowly but surely, the full Power Query capabilities are coming to the online variant of Excel!

It should be further note that queries will refresh:

- in the background so you may continue working in the spreadsheet
- asynchronously, instead of one-by-one, enabling a faster refresh.

This new functionality is available to all users on Excel for the web and may be located on the Data tab of the Ribbon, viz.



Furthermore, you can now refresh the Power Query queries in your workbook that source data from the current workbook ('From Table / Range') and anonymous OData feeds. In addition, you may also start with a Blank Query, write some M code, and refresh it.

As stated above, the refresh happens behind the scenes so you can keep editing the workbook while refreshing.

Presently, there are two ways to refresh:

1. Select the Data tab and then choose 'Refresh All' (pictured above)
2. Open the Queries Pane and then select Refresh.

As part of this announcement, Microsoft also mentioned that future plans will include Power Query Refresh for additional data sources, automation support, and eventually, releasing the full Power Query Editor experience to Excel for the web.

We can't wait.

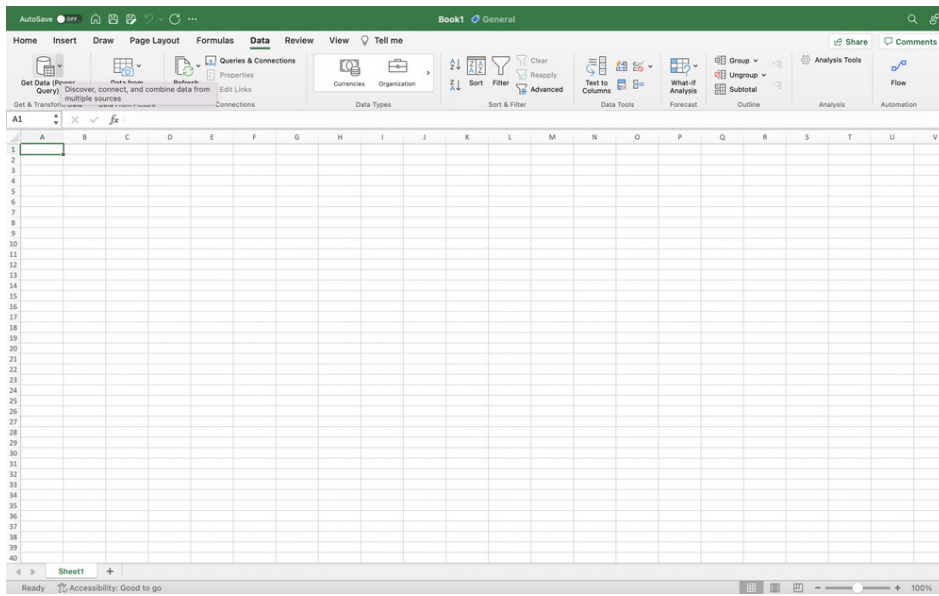
## Shaping Data with the Power Query Editor in Excel for Mac

When Power Query was first released in Excel for Mac, there was hope that it wouldn't take long for the features to mirror their Window counterparts. It's taking time, but the capabilities are improving. Beginning with the ability to refresh data two years ago, importing data was introduced this time last year, so this does seem to be an annual event! As you may imagine, Mac users have desperately requested the ability to transform data using the Query Editor, which would provide

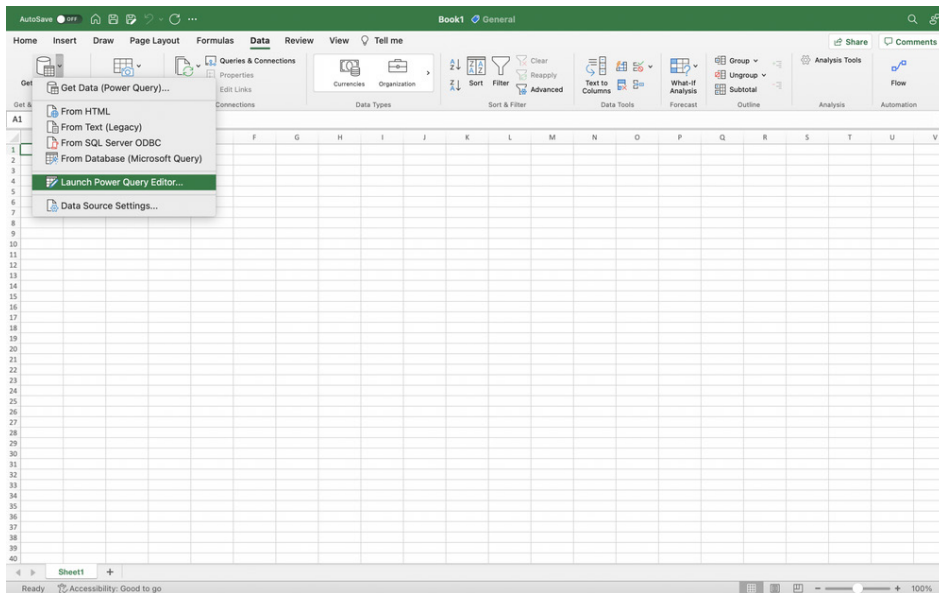
users with the full Power Query experience in Excel for Mac.

Well... it's getting there! You may now clean and shape your data with hundreds of transformations available in Power Query Editor in Excel for Mac.

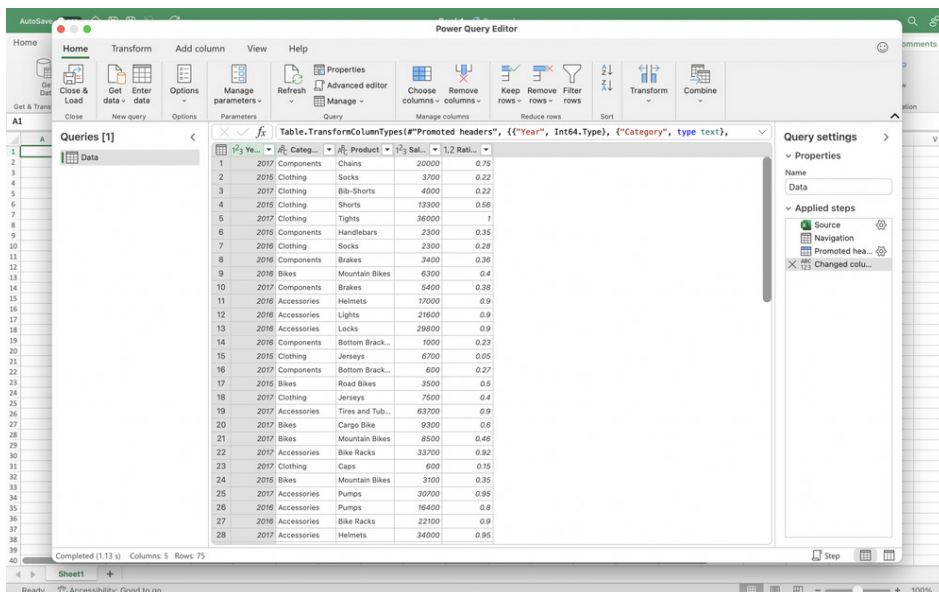
To access, on the Data tab, click the Get Data (Power Query) button.



Click Launch Power Query Editor to open the Query Editor.



You can shape and transform your data using the Query Editor similarly to Excel for Windows. When you're done, click the 'Close & Load' button on the Home tab.

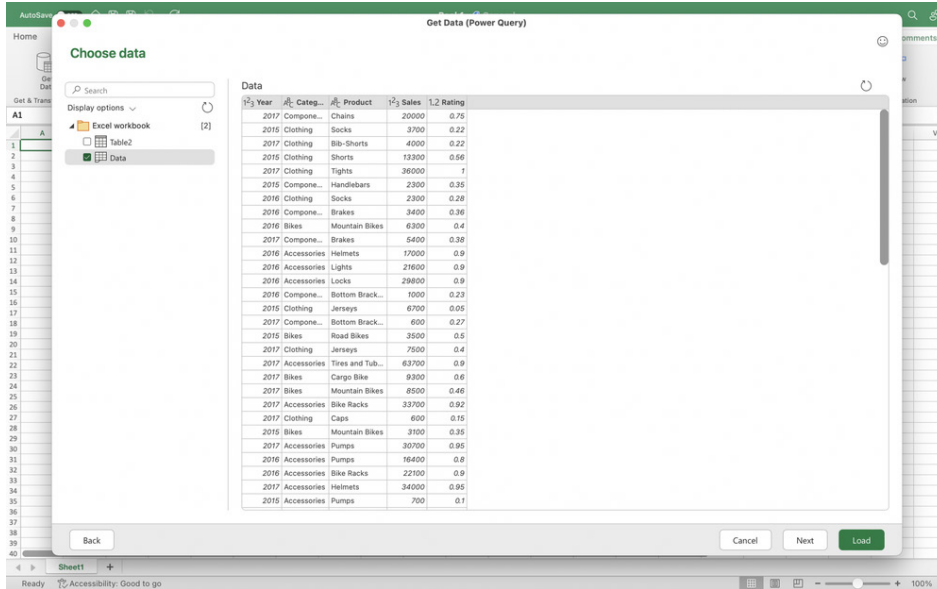


The newly imported data appears in a new sheet.

Supported data sources include:

- text, CSV, XLSX, XML and JSON files
- SharePoint, SharePoint Lists, SharePoint Folders and OData
- local tables, Tables and ranges
- Microsoft SQL Server.

You may also access the Query Editor from the data import flow by clicking the Get Data (Power Query) button, choosing a data source, and clicking the Next button



This feature is available to Beta Channel users running Version 16.61 (Build 22041701) or later.

If you have the right version of Excel for Mac and don't have it yet, don't fret. Typically, Microsoft release features over a period of time to ensure that things are working smoothly, including Insiders.

## New Power BI Known Issues Page

To keep Power BI's users more informed, Microsoft has recently launched a new Power BI Known Issues page (<https://docs.microsoft.com/en-us/power-bi/troubleshoot/known-issues/power-bi-known-issues>). This is intended to emphasise transparency and awareness (and no doubt try to discourage end users raising the same issues over and over again!).

The top-level page lists the current and recently closed known issues for Power BI.

Issue ID	Area	Title	Issue publish date	Status*
175	Consume and View	Unable to load Power BI dataset list in Excel	April 20, 2022	Open
169	Embedded, REST API or PowerShell	Try it button missing on REST API pages	April 19, 2022	Open
167	Consume and View	Matrix banded row colors appear incorrectly in Service	April 7, 2022	Open
166	Create and Author Data	DAX query fails due to parameter case sensitivity	April 7, 2022	Open
165	Refresh Data	Long running, failed or stuck dataflow in Premium Gen2	April 7, 2022	Open
164	Create and Author Data	Filter not correct after applying bookmark	April 5, 2022	Open
152	Publish Content	Uploading paginated report with same name fails	April 4, 2022	Fixed: April 5, 2022
151	Create and Author Data	Unable to open protected Desktop file	March 13, 2022	Open

\* Fixed issues are removed after 46 days.

To get more information about a specific known issue, click the Title link to open the details page for that known issue.

The screenshot shows a Microsoft Docs page for a known issue. The page title is "Known issue - Uploading paginated report with same name fails". The issue was reported on 04/19/2022 and is fixed as of April 5, 2022. The problem area is "Create and Author Data". The description states that when a paginated report with the same name as one already in the workspace is uploaded, an error occurs and the upload fails. Symptoms include the error message "Unable to upload paginated report ReportWithDisplayNameForTheModelExists" and the instruction to try again or contact support. Solutions and workarounds suggest re-uploading the report. The page also includes a "Next steps" section.

If you experience a problem with a feature, use the Known Issues page to determine whether that problem is new or is a known issue. For each known issue listed, the Power BI team will share details that help you identify whether the problem you are experiencing is the same as the selected known issue. This is done by both describing the issue and also including symptoms of the issue. If workarounds exist, they are included for you as well. When you see a known issue published on this page, Microsoft emphasises that you should be "...confident that a permanent solution is being worked on by our engineers".

If one of the issues matches a problem you're experiencing, help prioritise by clicking the 'Thumb up' button in the top right corner.

The intention is that Microsoft will update the Known Issue page often. For service-wide outages and degradation notices, note the sister website [www.support.powerbi.com](http://www.support.powerbi.com) (which for some reason appears to be down at the time of writing).

The screenshot shows a service status dashboard. It includes three main sections: "Service Status Per Region" with a legend for Good, Information, Degraded, and Outage; "Service Outage/Degradation" with a message that Power BI is running smoothly; and "Awareness" with a message that there are no updates or announcements to share at this time. A note at the bottom indicates that for Power BI Embedded service status, users should click a link.

## Recent Text and Array Functions: an Update

One of the great things about having new functions go into Beta in Excel is that Microsoft is prepared to listen and make changes before functions enter full production. Based upon user feedback, Microsoft has elected to "...enhance the signatures of the text functions..." as follows:

- added **[match\_end]** to **TEXTBEFORE** and **TEXTAFTER**. This will allow you to request a match against the end of the text
- renamed **[ignore\_case]** to **[match\_mode]** for **TEXTBEFORE** and **TEXTAFTER**
- added **[match\_mode]** to **TEXTSPLIT** to be more consistent with **TEXTBEFORE** and **TEXTAFTER**
- added **[if\_not\_found]** to **TEXTBEFORE** and **TEXTAFTER**. This expression is returned instead of #N/A when there is no delimiter match.

The updated text functions will become available to Beta channel users starting with build 15316.20000. The function documentation will be updated to reflect these changes in the coming weeks. Please note that you will need to update your **TEXTSPLIT** functions if you used the **[pad\_with]** argument, as this argument has moved from the fifth to the sixth argument. All of the other changes are backwards compatible.

Unfortunately, the announced alterations have not yet propagated as at the deadline for creating this newsletter, so I will go through the changes – with examples – in the July newsletter. To quote Arnold Schwarzenegger in a line from his upcoming musical, “I’ll be Bach”.

## Most Important Function in Financial Modelling..?

So what’s the most Important Function in Excel? Did you realise that’s what **IF** is an abbreviation for? Not surprising as I just made it up. However, there is some truth in the jest. The syntax for **IF** demonstrates just how useful this function is for financial modelling:

**=IF(logical\_test,[value\_if\_TRUE],[value\_if\_FALSE])**

This function has three arguments:

- **logical\_test**: this is the “decider”, that is, a test that results in a value of either TRUE or FALSE. Strictly speaking, the **logical\_test** tests whether something is TRUE; if not, it is FALSE
- **value\_if\_TRUE**: what to do if the **logical\_test** is TRUE. Note that you do not put square brackets around this argument. This is just the Excel syntax for saying sometimes this argument is optional. If this argument is indeed omitted, this argument will have a default value of TRUE
- **value\_if\_FALSE**: what to do if the **logical\_test** is FALSE (strictly speaking, not TRUE). If this argument is left blank, this argument will have a default value of FALSE.

This function is actually more efficient than it may look at first glance. Whilst the **logical\_test** is always evaluated, only one of the remaining two arguments is computed, depending upon whether the **logical\_test** is TRUE or FALSE.

Care should be taken with logical tests as this is the source of many, many errors in spreadsheets. Logical tests assess the criterion/criteria stipulated, no more no less. It assumes a binary universe: X and NOT(X). This isn’t always how our minds think, as I will explain with an exaggerated example.

Intrepid explorer Ivor Challenge is lost in the jungle and needs to find shelter for the night as a rainstorm beckons. Immediately ahead is a clearing with two caves. He writes a formula to determine which cave to sleep in:

**=IF(Cave 1 has a bear, sleep in Cave 2, sleep in Cave 1).**

The **logical\_test** is to check whether Cave 1 contains a bear. As it turns out, it doesn’t so he sleeps in there and is mauled to death by the lioness who was in there.

Next day, his wife, Cher Challenge, goes searching for him, gets tired and comes across the same caves and uses the same formula to determine which cave to sleep in:

**=IF(Cave 1 has a bear, sleep in Cave 2, sleep in Cave 1).**

The **logical\_test** is to check whether Cave 1 contains a bear. As it turns out, this time there is (together with some human bones) and so she sleeps in Cave 2 and is eaten by the other bear.

When using **IF** formulae, you need to train yourself to think logically like a computer. Common sense does not apply. Consider the logic function **NOT(expression)**, which is everything that is not equivalent to the **expression**. The opposite of a boy is “not a boy”: “girl” is incorrect.

Take care with inequalities in particular. The opposite of x is **greater than y** is either **x is less than or equal to y**, or **NOT(x is greater than y)**. This is a common error and it has caused embarrassing mistakes time and time again in business.

Returning to the **IF** function, let’s consider an example:

fx		=IF(Denominator=0,,Numerator/Denominator)				
D	E	F	G	H	I	
	Numerator	3				
	Denominator	-				
	Decimal	-				

In this example, the intention is to evaluate the quotient **Numerator / Denominator**. However, if the Denominator is either blank or zero, this will result in an **#DIV/0!** error. Excel has several errors that it cannot evaluate, such as **#REF!**, **#NULL**, **#N/A**, **#Brown**, **#Pipe**. OK, so one or two of these I may have made up, but prima facie errors should be avoided in Excel as they detract from the key results and cause the user to doubt the overall model integrity. Worse, in some instances these errors may contribute to Excel crashing and/or corrupting.

This is where **IF** comes in. In my example above,

**=IF(Denominator=0,,Numerator/Denominator)**

tests whether the Denominator is zero. This is the conditional formula. If so, the value is unspecified (blank) and will consequently return a value

of zero in Excel; otherwise, the quotient is calculated as intended.

This type of conditional formula is known as creating an **error trap**. Errors are “trapped” and the ‘harmless’ value of zero is returned instead. You could put “n.a” or “This is an error” as the **value\_if\_TRUE**, but you get the picture.

It is my preference not to put a zero in for the **value\_if\_TRUE**: personally, I think a formula looks clearer this way, but inexperienced end users may not understand the formula and you should consider your audience when deciding to put what may appear to be an unnecessary zero in a formula. The aim is to keep it simple **for the end user**.

An IF statement is often used to make a decision in the model:

**=IF(Decision\_Criterion=TRUE,Do\_it,Don't\_Do\_It)**

This automates a model and aids management in decision making and what-if analysis. **IF** is clearly a very powerful tool when used correctly. **IF** should never be used to look up data: there are plenty of functions out there to help with that problem, but we will discuss these later.

### Mixing Up Your IFS

As a model developer and reviewer, I must confess I remain unconvinced about this one. **If** you have ever used a formula with nested IF statements beginning with

**=IF(IF(IF...**

then maybe this next function is for you – however, if you have ever written Excel formulae like this, then maybe Excel isn't for you! There are usually better ways of writing the formula using other functions.

Most variants of Excel have the relatively new function **IFS**. The syntax for **IFS** is as follows:

**IFS(logical\_test1, value\_if\_true1, [logical\_test2, value\_if\_true2], [logical\_test3, value\_if\_true3],...)**

where:

- **logical\_test1** is a logical condition that evaluates to TRUE or FALSE
- **value\_if\_true1** is the result to be returned if **logical\_test1** evaluates to TRUE. This may be empty
- **logical\_test2** (and onwards) are further conditions that evaluate to TRUE or FALSE also
- **value\_if\_true2** (and onwards) are the respective results to be returned if the corresponding logical\_test evaluates to TRUE. Any or all may be empty.

Since functions are limited to 254 arguments (sometimes known as parameters), the new **IFS** function can contain 127 pairs of conditions and results.

One thing to note is that **IFS** is not quite the same as **IF**. With the **IF** statement, the third argument corresponds to what do if the **logical\_test** is not TRUE (that is, it is an ELSE condition). **IFS** does not have an inherent ELSE condition, but it can be easily generated. All you have to do is make the final **logical\_test** equal to a condition which is always true such as TRUE or 1=1 (*say*).

Other issues to consider:

- Whilst the **value\_if\_true** may be empty, it must not be omitted. Having an odd number of arguments in an **IFS** statement would give rise to the "You've entered too few arguments for this function" error message
- If a **logical\_test** is not actually a logical test (for example, it evaluates to something other than TRUE or FALSE, the function returns an #VALUE! error. Numbers still appear to work though: any number than zero evaluates as TRUE and zero is considered to be FALSE
- If no TRUE conditions are found, this function returns the #N/A error.

To show how it works, consider the following example.

Criteria	Yes / No	Grade
Already qualified?	No	3 Star
Work for Microsoft?	Yes	2 Star
Passed exam?	Yes	1 Star
Studying?	Yes	Student

Grade Achieved: 1 Star

Here, would-be gurus are graded based on evaluation criteria in the table, applied in a particular order:

**=IFS(H13="Yes",I13,H14="Yes",I14,H15="Yes",I15,H16="Yes",I16,TRUE,"Not a Guru")**

I think it's safe that although it is reasonably straightforward to follow, it is entirely reasonable to say it's not the prettiest, most elegant formula ever put to Excel paper. In particular, do pay heed to the final **logical\_test**: TRUE. This ensures we have an ELSE condition as discussed above.

To be fair, one similar solution using legacy Excel functions isn't any better:

**=IF(H13="Yes",I13,IF(H14="Yes",I14,IF(H15="Yes",I15,IF(H16="Yes",I16,"Not a Guru")))).**

You may note I am not supplying multiple examples of IFS formulae. This is because wherever possible you should try and replace the logic with a simpler, more accessible, logic for end users. For instance, sometimes the logic of an elongated IF or IFS formula may be condensed to

**=IF(Multiple Conditions = TRUE, Do Something, Do Something Else).**

In this situation, there is a function in Excel that can help.

My old English teacher said you should never start or finish a sentence with the word “and”. **AND** is one of several Excel logic functions (others include **NOT** [already mentioned earlier, which takes the logical opposite of an expression] and **OR**). It returns TRUE if all of its arguments evaluate to TRUE; it returns FALSE if one or more arguments evaluate to FALSE.

One common use for the **AND** function is to expand the usefulness of other functions that perform logical tests. For example, the **IF** function performs a logical test and then returns one value if the test evaluates to TRUE and another value if the test evaluates to FALSE. By using the **AND** function as the **logical\_test** argument of the **IF** function, you can test many different conditions instead of just one.

For example, imagine you are in New York on a Monday. Consider the expression

**=AND(condition1, condition2, condition3)**

where:

- **condition1** is the condition, “today is Monday”
- **condition2** is the condition, “you are in New York” *and*
- **condition3** is the condition, “this author is the best looking guy you have ever seen”.

This would clearly be FALSE as not everywhere in the world it would be Monday (that is, **condition1** would be breached)...

As alluded to above, the syntax for **AND** is as follows:

**AND(logical1, [logical2], ...)**

where:

- **logical1**: the first condition that you want to test that can evaluate to either TRUE or FALSE
- **logical2**: additional conditions that you want to test that can evaluate to either TRUE or FALSE, up to a maximum of 255 conditions. **logical2** is optional and is not needed in the syntax.

It should be noted that:

- The arguments must evaluate to logical values, such as TRUE or FALSE, or the arguments must be arrays or references that contain logical values
- If an array or reference argument contains text or empty cells, those values are ignored
- If the specified range contains no logical values, the **AND** function returns the #VALUE! error value.

To highlight how **AND** works:

	A	B	C
1	Condition 1	TRUE	
2	Condition 2	FALSE	
3	Condition 3	TRUE	
4			
5			
6	Description	Results	Formula
7	All arguments are true	FALSE	=AND(B1:B3)
8	The first and last arguments are true	TRUE	=AND(B1,B3)
9	At least one argument is false	TRUE	=NOT(AND(B1:B3))
10			

For a more practical example, consider the following summary data table:

	A	B	C	D	E	F
1	Staff ID	Works in Sales?	Sales Made	Threshold	Bonus %	Bonus Paid
2	ID 2017	yes	\$ 9,069	\$ 5,000	2.50%	\$ 227
3	ID 3102	yes	\$ 6,285	\$ 8,000	2.00%	\$ -
4	ID 3148	no	\$ 8,458			\$ -
5	ID 3321	yes	\$ 5,635	\$ 3,000	3.00%	\$ 169
6	ID 3817	no	\$ 19,973			\$ -
7	ID 5298	no	\$ 7,986			\$ -
8	ID 6774	yes	\$ 1,571	\$ 5,000	2.50%	\$ -
9	ID 8563	no	\$ 16,124			\$ -
10						
11				=IF(AND(B2="yes",C2-D2>=0),C2*E2,)		
12						



Here, we have a list of staff in column **A**, with identification of those who work in Sales (that is, eligible for a bonus) in column **B**. Details of the sales made, the threshold for getting a bonus and what rate it is paid, are detailed in columns **C**, **D** and **E** respectively. The formula in cell **F2**:

**=IF(AND(B2="yes",C2-D2>=0),C2\*E2,)**

denotes the Bonus Paid and is conditional on them working in Sales (**B2="yes"**) and that the sales made were at or above the required threshold (**C2-D2>=0**). If both conditions are TRUE, then a bonus (**C2\*E2**) is paid accordingly (putting nothing after the final comma is the equivalent of saying "else zero"). This is a prime example of **IF** and **AND** working together – and more often than not, these formulae are much easier to read than their **IF(IF... or IFS** counterparts.

The other logic function not yet mentioned, **OR**, is similar to **AND**, but only requires one condition to be TRUE. Similar to **AND**, the **OR** function may be used to expand the usefulness of other functions that perform logical tests. For example, the **IF** function performs a logical test and then returns one value if the test evaluates to TRUE and another value if the test evaluates to FALSE. By using the **OR** function as the **logical\_test** argument of the **IF** function, you can test many different conditions instead of just one.

For example, imagine you are in London on a Tuesday. Consider the expression

**=OR(condition1, condition2, condition3)**

where:

- **condition1** is the condition, "today is Tuesday"
- **condition2** is the condition, "you are in London" *and*
- **condition3** is the condition, "the Earth is flat".

This would clearly be TRUE as you are definitely in London (that is, **condition2** holds).

The syntax for **OR** is as follows:

**OR(logical1, [logical2], ...)**

where:

- **logical1**: the first condition that you want to test that can evaluate to either TRUE or FALSE
- **logical2**: additional conditions that you want to test that can evaluate to either TRUE or FALSE, up to a maximum of 255 conditions. **logical2** is optional and is not needed in the syntax.

It should be noted that:

- The arguments must evaluate to logical values, such as TRUE or FALSE, or the arguments must be arrays or references that contain logical values
- If an array or reference argument contains text or empty cells, those values are ignored
- If the specified range contains no logical values, the **OR** function returns the **#VALUE!** error value.

In summary, **OR** works as follows:

	A	B	C
1	Condition 1	TRUE	
2	Condition 2	FALSE	
3	Condition 3	FALSE	
4			
5			
6	Description	Results	Formula
7	At least one argument is true	TRUE	<b>=OR(B1:B3)</b>
8	All arguments are false	FALSE	<b>=NOT(OR(B1:B3))</b>
9	At least one argument is false	TRUE	<b>=NOT(AND(B1:B3))</b>
10			

For a more practical example, consider the following summary data table:

	A	B	C	D	E	F
1	Staff ID	Works in Sales?	Sales Made	Threshold	Bonus %	Bonus Paid
2	ID 2017	yes	\$ 9,069	\$ 5,000	2.50%	\$ 227
3	ID 3102	yes	\$ 6,285	\$ 8,000	2.00%	\$ -
4	ID 3148	no	\$ 8,458			\$ 85
5	ID 3321	yes	\$ 5,635	\$ 3,000	3.00%	\$ 169
6	ID 3817	no	\$ 19,973			\$ 200
7	ID 5298	no	\$ 7,986			\$ 80
8	ID 6774	yes	\$ 1,571	\$ 5,000	2.50%	\$ -
9	ID 8563	no	\$ 16,124			\$ 161
10						
11	<b>Non-Sales Staff</b>					
12						
13		Threshold	\$ 5,635			
14						
15		Bonus %	1.00%			
16						
17						
18	=IF(OR(AND(B2="yes",C2-D2>=0),AND(B2<>"yes",C2-\$C\$13>=0)),C2*IF(B2="yes",E2,\$C\$15),)					

Now there is a complex formula:

=IF(OR(AND(B2="yes",C2-D2>=0),AND(B2<>"yes",C2-\$C\$13>=0)),C2\*IF(B2="yes",E2,\$C\$15),)

It isn't quite as bad as it first seems. This is based on the **AND** case study from earlier, but it also allows for Non-Sales staff to participate in the bonus scheme too. The **logical\_test** in the primary **IF** statement,

OR(AND(B2="yes",C2-D2>=0),AND(B2<>"yes",C2-\$C\$13>=0))

Is essentially **OR(condition1,condition2)**. The first condition is as before for Sales staff, whereas the second,

AND(B2<>"yes",C2-\$C\$13>=0)

checks whether Non-Sales staff have exceeded the Non-Sales Staff threshold (cell **C13**). Do you see that the check for Non-Sales staff is given by **B2<>"yes"** (**B2** is not equal to "yes") rather than **B2="no"**. This takes me back to my earlier point about ensuring you develop your **logical\_test** correctly. It's a subtle point, but will ensure all staff are considered (rather than excluding staff where no entry has been made in column **B**).

The other **IF** statement,

IF(B2="yes",E2,\$C\$15)

simply ensures the correct bonus rate is applied to the sales figure.

To summarise so far, sometimes your **logical\_test** might consist of multiple criteria:

=IF(condition1=TRUE,IF(condition2=TRUE,IF(condition3=TRUE,formula,)),)

Here, this formula only gives a value of 1 if all three conditions are true. This nested **IF** statement may be avoided using the logical function **AND(Condition1,Condition2,...)** which is only TRUE if and only if all dependent arguments are TRUE,

=IF(AND(condition1,condition2,condition3),formula,)

which is actually easier to read. A similar example may be constructed for **OR** also. However, even using these logic functions, formulae may become – or simply look – complex quite quickly. There is an alternative: **flags**. In its most common form, flags are evaluated as

=(condition=TRUE)\*1

**condition=TRUE** will give rise to a value of either TRUE or FALSE. The brackets will ensure this is **condition** is evaluated first; multiplying by 1 will provide an end result of zero (if FALSE, as FALSE\*1 = 0) or one (if TRUE, TRUE\*1 = 1). I know some modellers prefer TRUEs and FALSEs everywhere, but I think 1's and 0's are easier to read (when there are lots of them) and more importantly, easier to sum when you need to know how many issues there are.

Flags make it easier to follow the tested conditions. Consider the following:

D9		=PRODUCT(D4:D7)												
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1														
2	Counter		1	2	3	4	5	6	7	8	9	10		
3														
4	Divisible by 3		-	-	1	-	-	1	-	-	1	-		=(MOD(Counter,3)=0)*1
5	Greater than 4		-	-	-	-	1	1	1	1	1	1		=(Counter>4)*1
6	Less than or equal to 9		1	1	1	1	1	1	1	1	1	1		=(Counter<=9)*1
7	Is not 6		1	1	1	1	1	-	1	1	1	1		=(Counter<>6)*1
8														
9	Product											1		=PRODUCT(D4:D7)
10														
11														

In this illustration, you might not understand what the **MOD** function does, but hopefully, you can follow each of the flags in rows 4 to 7 without being an Excel guru. Row 9, the product, simply multiplies all of the flags together (using the **PRODUCT** function allows you to add additional conditions / rows easily). This effectively produces a sophisticated **AND** flag, where all of the formulae are mercifully short.

As a brief aside, some readers ask me why I use **PRODUCT** rather than

**MIN**. That’s a good question, especially given I use **MAX** (*below*). I confess it originally was partly a preference and partly the fact that if you are modelling optimisation problems, **MIN** can give rise to non-smooth outputs (not a good thing) where **PRODUCT** does not. These days, I do tend to use **MIN** more and more.

If I wanted the flag to be a 1 as long as one of the above conditions is TRUE (that is, I wish to construct an **OR** equivalent), that is easy too:

D9		=MAX(D4:D7)												
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1														
2	Counter		1	2	3	4	5	6	7	8	9	10		
3														
4	Divisible by 3		-	-	1	-	-	1	-	-	1	-		=(MOD(Counter,3)=0)*1
5	Greater than 4		-	-	-	-	1	1	1	1	1	1		=(Counter>4)*1
6	Less than or equal to 9		1	1	1	1	1	1	1	1	1	1		=(Counter<=9)*1
7	Is not 6		1	1	1	1	1	-	1	1	1	1		=(Counter<>6)*1
8														
9	MAX		1	1	1	1	1	1	1	1	1	1		=MAX(D4:D7)
10														
11														

Flags frequently make models more transparent and this example provides a great learning point. Often, we mistakenly believe that condensing a model into fewer cells makes it more efficient and easier follow. On the contrary, it is usually better to step out a calculation. If it can be followed on a piece of paper (without access to the formula bar), then more people will follow it. If more can follow the model logic,

errors will be more easily spotted. When this occurs, a model becomes trusted and therefore is of more value in decision-making.

Ne careful though. Sometimes you just can't use flags. Let me go back to my first example in this chapter – but this time using the flag approach:

fx		=(Numerator/Denominator)*(Denominator<>0)				
D	E	F	G	H	I	
	Numerator	3				
	Denominator	-				
	Decimal	#DIV/0!				

Here, the flag does not trap the division by zero error. This is because this formula evaluates to

$$=#DIV/0! \times 0$$

which equals #DIV/0! If you need to trap an error, you must use an **IF** function.

## 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...

This month’s challenge involves the automation of grouping the rows according to different level of headings. In financial modelling, grouping the rows for each worksheet based on the headings is a time-consuming

process, it is especially the case for some financial models that have multiple worksheets. This time out, we have a worksheet with three [3] different levels of headings as shown below:

**Text**

<b>Text</b>														
<b>Text</b>	text	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula
<b>Text</b>	text	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula
<b>Text</b>	text	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula
<b>Text</b>	text	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula
<b>Text</b>	text	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula

**Text**

<b>Text</b>														
<b>Text</b>	text	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula
<b>Text</b>	text	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula

**Text**

<b>Text</b>	text	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula
-------------	------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------

After grouping at each different level, the result should be as follows.

First level:

Second level:

Third level:

We want to automate this process without manually grouping each heading. Sounds like a job for VBA...

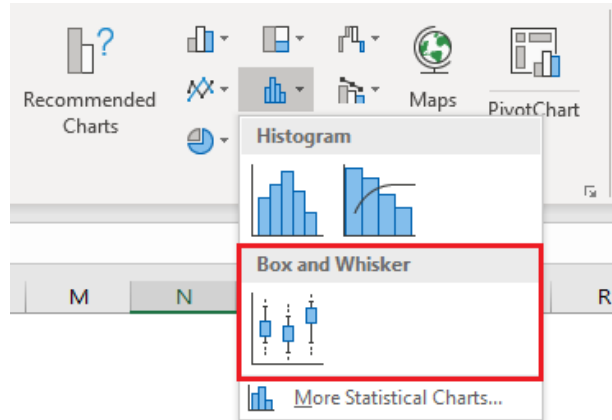
Sound easy? Try it. One solution just might be found later in this newsletter – but no reading ahead!

## Charts and Dashboards

*It's time to chart our progress with an introductory series into the world of creating charts and dashboards in Excel. This month, we look at Box and Whisker charts.*

The Box and Whisker Chart shows the distribution of data by plotting five statistical values, being the maximum, minimum, median, the upper quartile and the lower quartile. The average or mean can also be added to the chart, if required.

There is just one type of Box and Whisker chart available in Excel, which is accessible from the Statistical Chart section under Charts in the Insert tab in the Ribbon.



Previously, when we reviewed Radar charts, we graphed the average of the evaluation results that came in from 11 key people within a given example company. However, an average does not indicate anything about the level of consensus of the 11 individuals who participated in the survey, only the statistical middle point of their opinions.

To explain, if 100 people completed a single survey question about customer service and the average was 50%, you might conclude that the group were neutral overall in their opinion, and further consider implementing processes simply to improve customer service generally to try and improve the average over time. However, if we then discovered

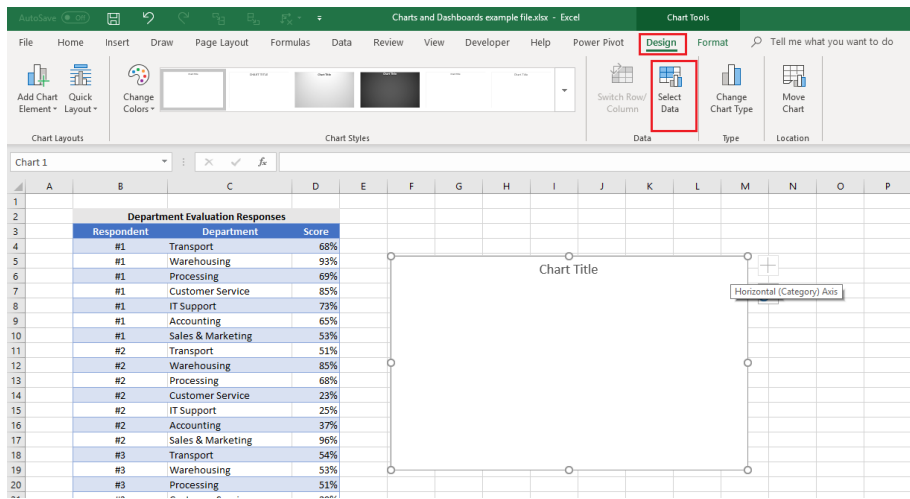
that 50 responded with 100% and the other 50 responded with 0%, then the 50% average becomes meaningless. In reality, half the group were completely satisfied and the other were completely dissatisfied, so my reaction now might be to try and access why half the group were unhappy and address that specific issue rather than improve the service generally.

Let's assume we have the individual submissions from the 11 participants from the survey. We may then map their responses and evaluate their feedback in more detail.

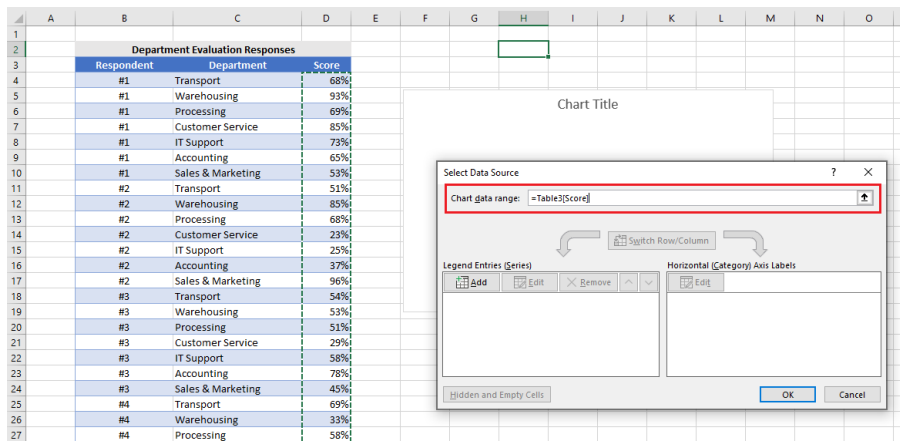
Departmental Evaluation Responses		
Respondent	Department	Score
#1	Transport	68%
#1	Warehousing	93%
#1	Processing	69%
#1	Customer Service	85%
#1	IT Support	73%
#1	Accounting	65%
#1	Sales & Marketing	53%
#2	Transport	51%
#?	Warehousing	95%
...	...	...
#10	Accounting	74%
#10	Sales & Marketing	74%
#11	Transport	55%
#11	Warehousing	93%
#11	Processing	85%
#11	Customer Service	96%
#11	IT Support	81%
#11	Accounting	63%
#11	Sales & Marketing	73%

The layout of the data table is quite simple. Two columns are required: the first contains the category, in this case the departments being evaluated, and the second column contains the values, being the individual scores from each respondent. An extra column has been added on the left, just to ensure we've included all participants' responses.

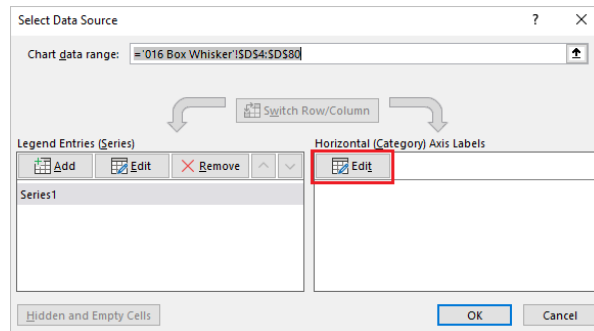
Without selecting any data, inserting a Box and Whisker Chart will merely create an empty chart box. With the chart box selected, go to the Design tab on the Ribbon and click on 'Select Data'.



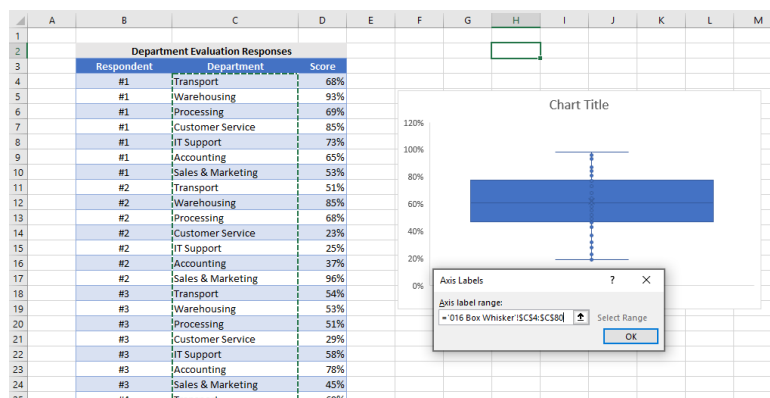
We may now provide Excel with the information needed to create the chart. The first range it needs is the data series which is the column of scores from our table:



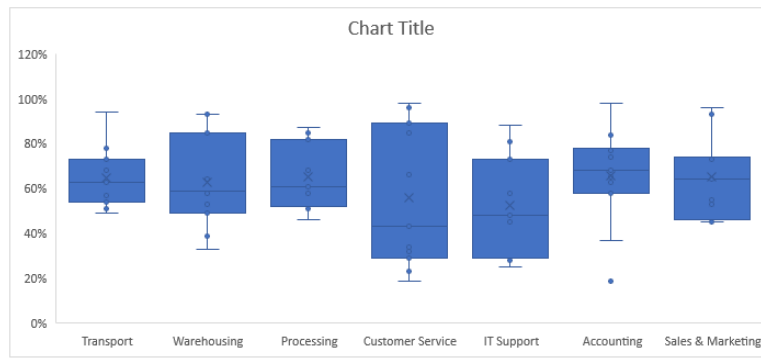
Select the data, then click on the Edit button under Horizontal (Category) Axis Labels and highlight the range under the Department column:



When selecting data, do not include the headings:



Once you click OK, Excel will collate the data automatically and prepare the chart like the one below:



It is called a Box and Whisker chart as the key elements of the chart are the coloured block called the Box, and a line that protrudes out of the top and bottom of the Box called the Whisker. The start and end points of the Whisker are the minimum and maximum values in the data. The line that goes across the Box and the cross inside the Box are the median

and mean respectively. The position and height of the Box is determined by the first quartile and third quartile. To clarify the terminology and better appreciate the information in the chart, let's focus on the results for the Transport department. The table below shows the responses from the 11 participants in the performance survey:

Respondent	Transport
#1	68%
#2	51%
#3	54%
#4	69%
#5	73%
#6	49%
#7	94%
#8	78%
#9	57%
#10	63%
#11	55%

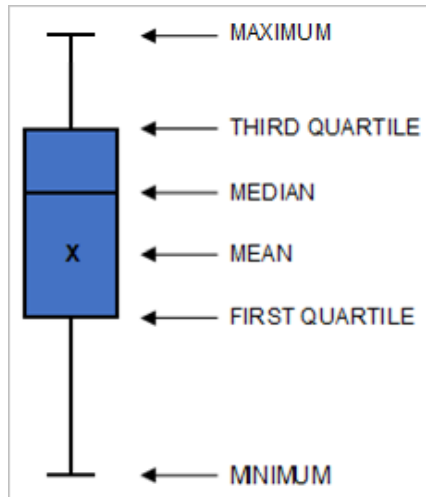
The first thing we need to do is sort the data numerically, so the scores are ranked from lowest to highest. The lowest score is the minimum, and, equally as obvious, the highest score is the maximum. The value

in the centre of the series is called the median. The mean is simply the average of the series, which is rounded to 65%.

Respondent	Transport	
#6	49%	Minimum
#2	51%	
#3	54%	
#11	55%	Median
#9	57%	
#10	63%	
#1	68%	Maximum
#4	69%	
#5	73%	
#8	78%	
#7	94%	
Mean	65%	

When it comes to quartiles, just think that I am splitting the data into quarters. If you wrote the above sorted series of number across a piece of paper all in a row equally spaced and tore the page in half precisely, you would be tearing the list where the number 63% lies. If you then took the piece then the numbers from the minimum upwards on it and

tore this in half, I would be tearing where the 54% is written. Tearing in half the piece with the maximum downwards on it would result in ripping where the 73% lies. We now have the quartile values. The first quartile is at 54%, the second quarter or the median is 63% and the third quartile is 73%.

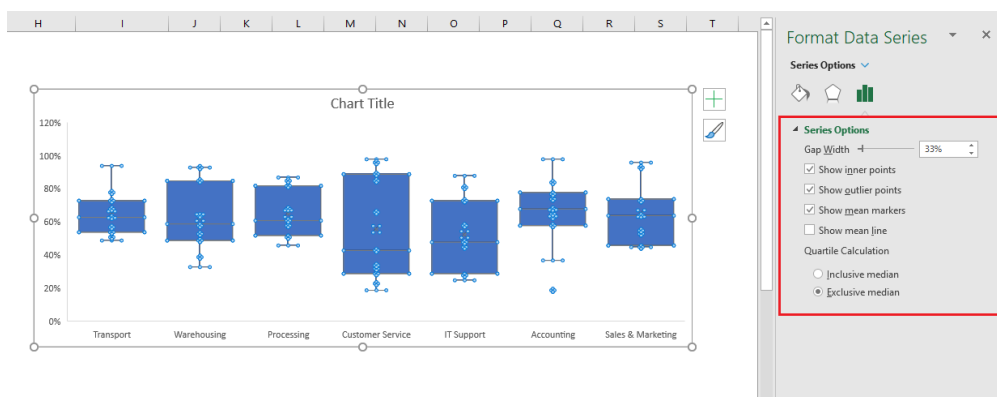


The mathematics behind the calculation of quartiles is actually much more complex than this and our example is simplified by only having 11 participants in my base. Needless to say, Excel has the mathematics built in to determine quartiles precisely.

Note that the median will always reside within or on the boundary of the Box, but the mean can reside above or below the Box but always within the range of the Whisker.

Respondent	Transport	
#6	49%	Minimum
#2	51%	
#3	54%	First Quartile
#11	55%	
#9	57%	
#10	63%	Median
#1	68%	
#4	69%	
#5	73%	Third Quartile
#8	78%	
#7	94%	Maximum
	Mean	65%

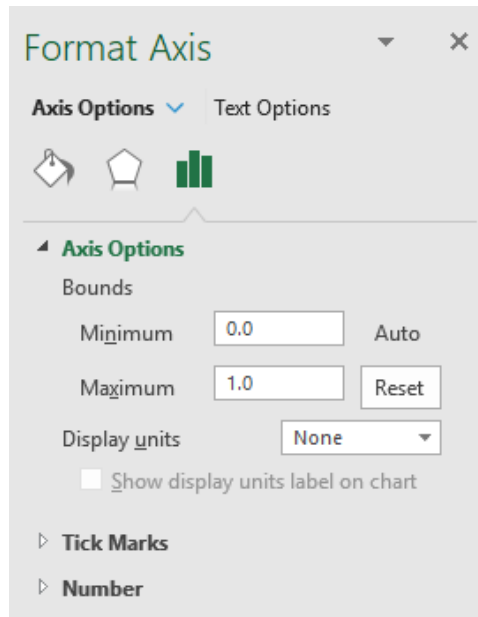
Let's remove the point elements in the initial chart, by clicking on one of the points, the 'Format Data Series' dialog will appear, in 'Series Options', untick the 'Show inner points' and 'Show outlier points':



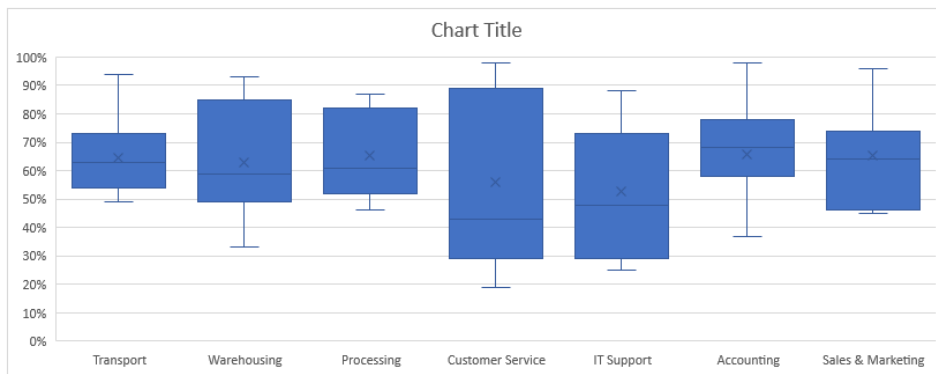


Before analysing this chart, it would be beneficial to change the vertical axis settings. The chart presently goes to 120% but obviously the maximum value is 100%, and it would be easier to read if the gridlines were every 10% instead of every 20%. Therefore, select the vertical axis by clicking on the area where the percentages are, right-click and choose

'Format Axis', then under 'Axis Options', change Maximum setting under Bounds to be 1.0 (which equates to 100% in our data). Unfortunately, you cannot specify that you want the scale to be every 10%, so if the chart is still showing every 20%, you just need to click on the border of the Chart Area and extend the height of the chart.



Let's add a grid line to the chart so that it now looks like this:

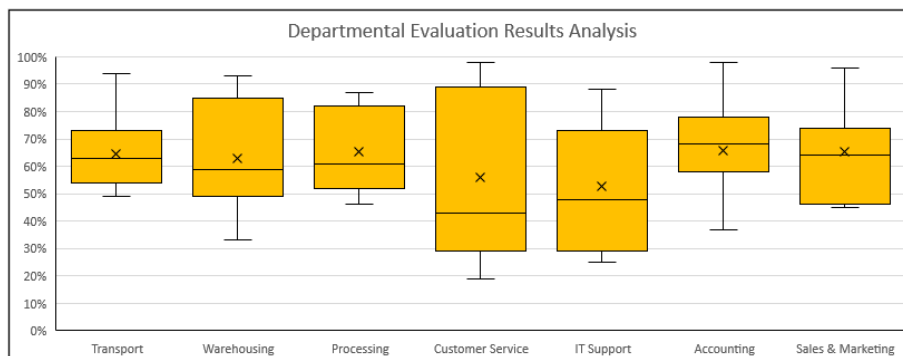


So how do you read this chart? What conclusions can you draw from the additional statistics about the data that we couldn't draw from the Radar Chart?

If we look at the results, Customer Service has a very broad range of score, with a minimum of 19% and a maximum of 98%, the first quartile of 29% and third quartile of 89%, while the median of 43% tells that half of the responses lies below 43% and half lies above 43%, which mean the evaluation is quite evenly distributed. Meanwhile, Transport has a minimum score of 49% and maximum of 94%, the mean and median are

almost identical, 63% and 65% respectively. It is obvious that the distance between the third quartile (73%) to the maximum score outweighs that of the first quartile (54%) to the minimum score, meaning 75% of the data is between 49% and 73%. This may require further analysis of the individual responses, but it could be interpreted that there is mixed feedback from the respondents as to their experiences working with Transport department.

Final touches applied, the chart is ready for presentation:



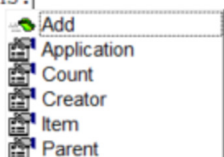
More next month...

## 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. Continuing our series about using **ListObjects** to manipulate Tables within an Excel workbook in VBA, this month we consider all the commonalities between **ListColumns** and **ListRows** on the **Item** level.

**ListColumns** can be really confusing. Using **ListColumns** references ALL the columns, but using **ListColumns(Index)** only references a specific column. Remember in VBA you may press "." and the applicable method / properties that apply to that object come up? This is true with **ListColumns**:

```
Sub Test()  
  
Dim MyTable As ListObject  
Set MyTable = Range("MyTable").ListObject  
  
MyTable.ListColumns.|  
  
End Sub
```



- **Add**: this allows you to add a column. By default it will be added at the end of the table
- **Application**: this returns the name of the application from which the object belongs to (in this case Excel)
- **Count**: returns the number of columns
- **Creator**: returns the name of the author of the object
- **Item**: this returns the column as an object. **ListColumns.Item(Index)** is identical to using **ListColumns(Index)** in terms of referencing a specific column
- **Parent**: this returns the overall owner of the object, in this case, being "MyTable".

All of these properties are identical in **ListRows**: simply interchange column in the above list with row!

Notice how there is no 'Delete' property here? This is done on a row / column indexed basis. This is because invoking 'Delete' would either delete your entire data set (on **ListRows**) or the entire table (on **ListColumns**). If that is indeed what you wish to do, then using **DataBodyRange** or the **ListObject** table itself would be more intuitive and understandable.

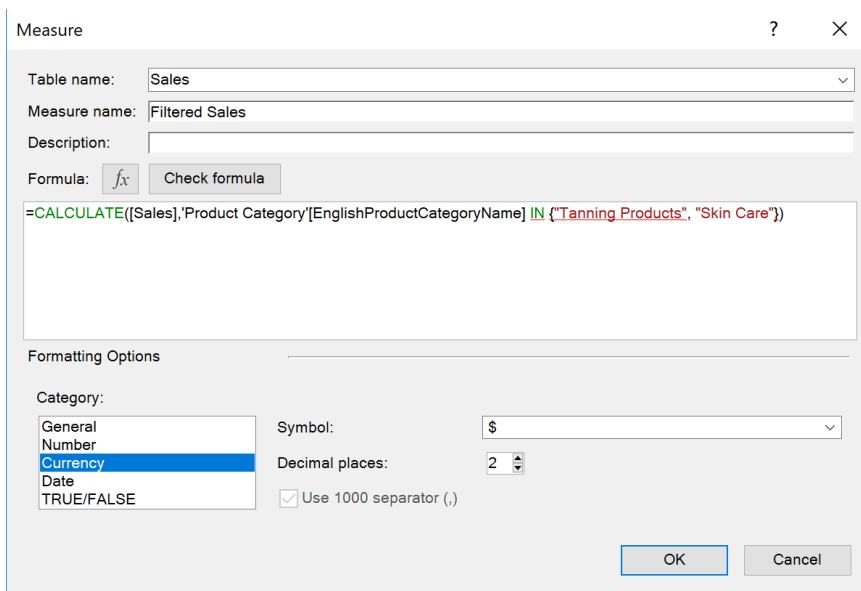
More next time.

## Power Pivot Principles

We continue our series on the Excel COM add-in, Power Pivot. This month, we get with the in-crows and look at the **IN** function.

The **IN** function is a logical function that returns with TRUE for the values in the row of data that match the input value.

For example, we can use the **IN** function together with the **CALCULATE** function to create a filtered measure that sums the sales for products that are classified as 'Tanning Products' and 'Skin Care'.



Measure

Table name: Sales

Measure name: Filtered Sales

Description:

Formula:

=CALCULATE([Sales], Product Category[EnglishProductCategoryName] IN ("Tanning Products", "Skin Care"))

Formatting Options

Category: General, Number, **Currency**, Date, TRUE/FALSE

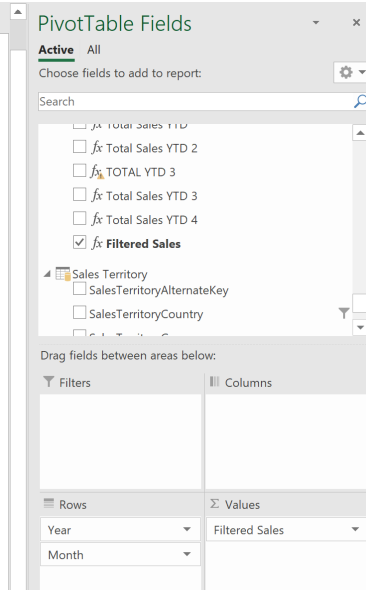
Symbol: \$

Decimal places: 2

Use 1000 separator (,)

The results are somewhat similar to filtered measures:

Year	Month	Filtered Sales
2014		\$18,554.30
2015		
	1	\$3,389.00
	2	\$2,985.30
	3	\$3,111.15
	4	\$3,842.85
	5	\$3,885.55
	6	\$4,259.05
	7	\$5,088.90
	8	\$5,884.50
	9	\$3,622.80
	10	\$4,481.15
	11	\$3,897.60
	12	\$5,846.30
2016		\$458,309.80
2017		\$603,822.85
<b>Grand Total</b>		<b>\$1,130,981.10</b>



It's more useful than you might think, and modellers often use quite convoluted formulae to derive the same result(s).

More Power Pivot Principles next month.

## 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](http://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 the **List** function to compare lists.

This isn't the first time we've considered how to compare data in different queries. Another way to do this is to extract the data to lists. Let's consider the following example:

FULL_NAME	REP_TYPE_C3	AREA_CODE	DELETE_FL
HEATHER LOWE	AMR	USA	N
SUSAN HAMPS	AMR	USA	N
SUNITA SINDA	AMR	USA	N
LUCY LEVITT	AMR	USA	N
NOREEN WESTON	AMR	USA	N
TREVOR BONN	AMR	USA	N
DAVID BROWN	ARM	USA	N
CARL LUNG	AMR	USA	N
NOREEN EAST	CRM	UK	N
BRIAN MEAD	AMR	USA	N
MARC SINDON	CRM	UK	N
DARREN PETERS	CRM	UK	N
ROLAND RIVER	AMR	USA	N
BARBARA CLARKE	CRM	UK	N
ANN CINDER	AMR	USA	N
BRENDA CORAL	AMR	USA	N
TONY MEAD	CRM	UK	N

FULL_NAME	REP_TYPE_C3	AREA_CODE	DELETE_FL
ZOE BROWN	AMR	USA	N
JOHN JAMES	AMR	USA	N
HEATHER LOWE	AMR	USA	N
SUSAN HAMPS	AMR	USA	N
SUNITA SINDA	AMR	USA	N
LUCY LEVITT	AMR	USA	N
NOREEN WESTON	AMR	USA	N
TREVOR BONN	AMR	USA	N
DAVID BROWN	ARM	USA	N
CARL LUNG	AMR	USA	N
NOREEN EAST	CRM	UK	N
BRIAN MEAD	AMR	USA	N
MARC SINDON	CRM	UK	N
DARREN PETERS	CRM	UK	N
ROLAND RIVER	AMR	USA	N
BARBARA CLARKE	CRM	UK	N
ANN CINDER	AMR	USA	N

We wish to do three things:

1. We want to see who is in Employee List 1 but not in Employee List 2
2. We want to see who is in both lists
3. We want to combine the lists.

## 1. Who is missing from Employee List 2?

To find this out, let's create two lists in Power Query and use **List.Difference** to detect entries in the first list that do not appear in the second list.

**List.Difference(list1 as list, list2 as list, optional equationCriteria as any)** as list

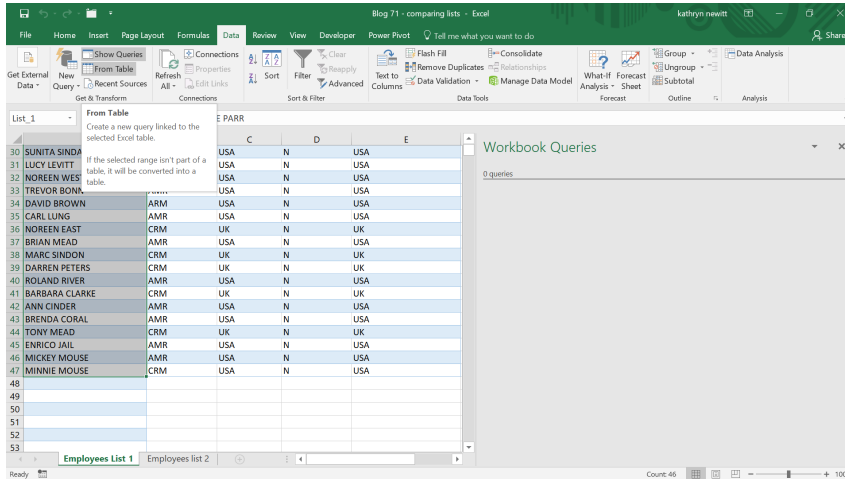
This returns the items in **list 1** that do not appear in **list 2**. Duplicate values are supported.

On the whole, this is easy to follow. You are required to enter two lists which will be compared. When it says, "duplicates are supported", this means that if 'John Smith' appears twice in **list 1**, but not at all in **list**

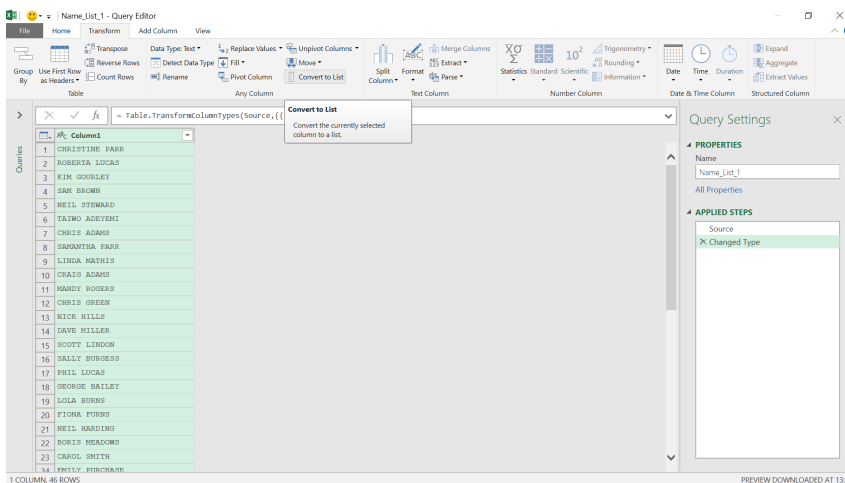
**2**, then 'John Smith' will appear twice in the results. If we don't want this, then we need to remove duplicates before comparing lists. We may either do this in the Query Editor interface or use **List.Distinct**.

The optional third parameter is **equationCriteria**, which we don't need for this example. If you wanted to ensure that upper and lower-case names were treated the same, then you could use this feature.

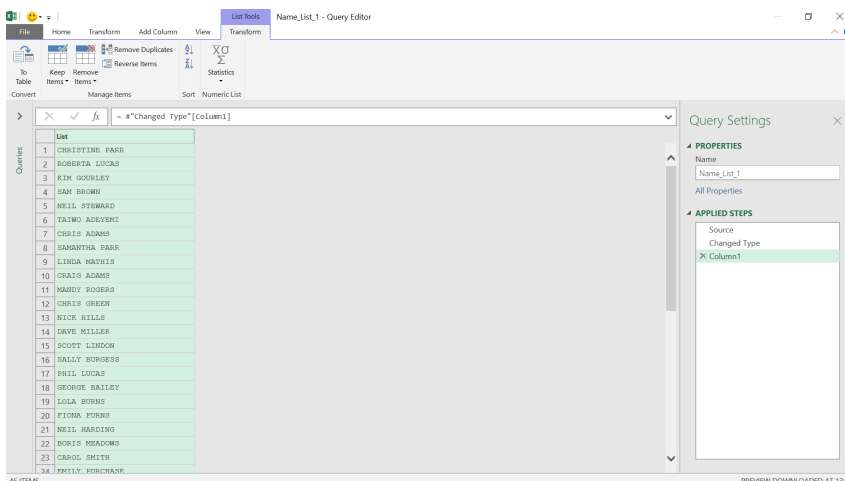
Let's begin by creating a list of names from **Employee List 1**. Our data is not already in a table in the worksheet, so a new table will be created when we choose the 'From Table' option in the 'Get and Transform' section of the 'Data' tab, with all our data in the **FULL\_NAME** column selected (but not the heading).



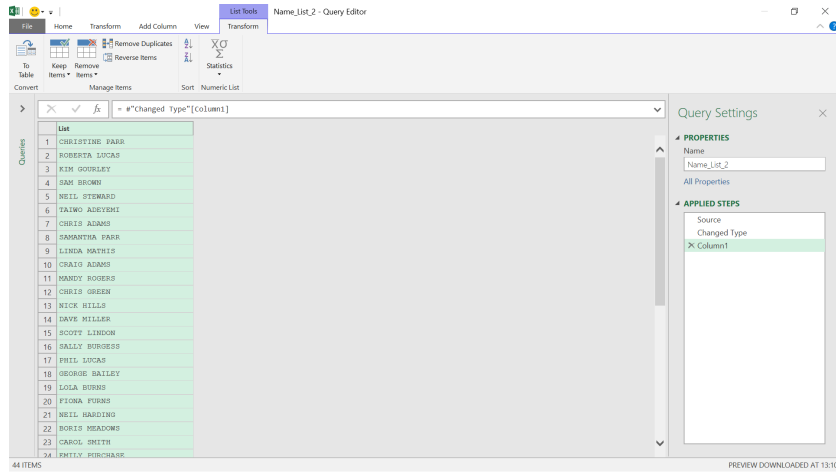
We shall save this data in a query called 'Name\_List\_1':



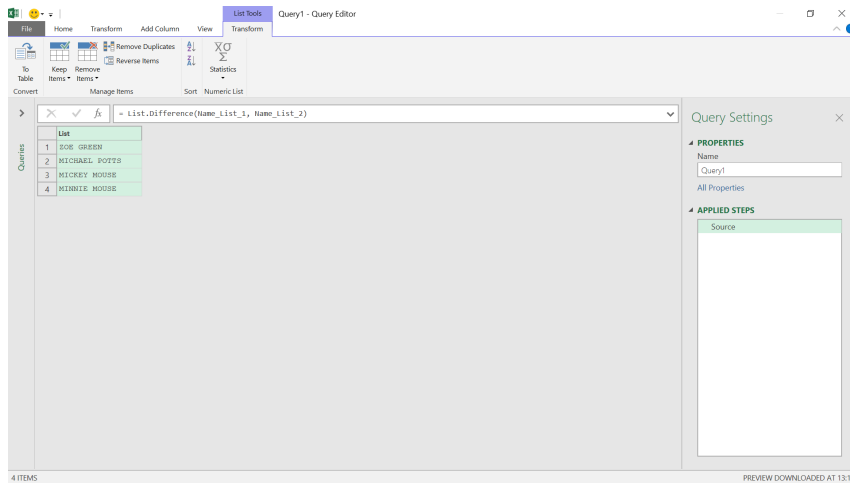
We will convert our data to a list using the option in the 'Any Column' section of the 'Transform' tab, and save it as connection only.



We may do the same for our other list, so that we have 'Name\_List\_2':



We can now create a new blank query, ready to compare our lists.



The results are presented as a list of those names that appear in 'Name\_List\_1' but not 'Name\_List\_2'. The syntax used is:

**= List.Difference(Name\_List\_1, Name\_List\_2)**

If there had not been anyone on the first list that wasn't on the second, then the answer would have been a blank list.

## 2. Who is on Both Lists?

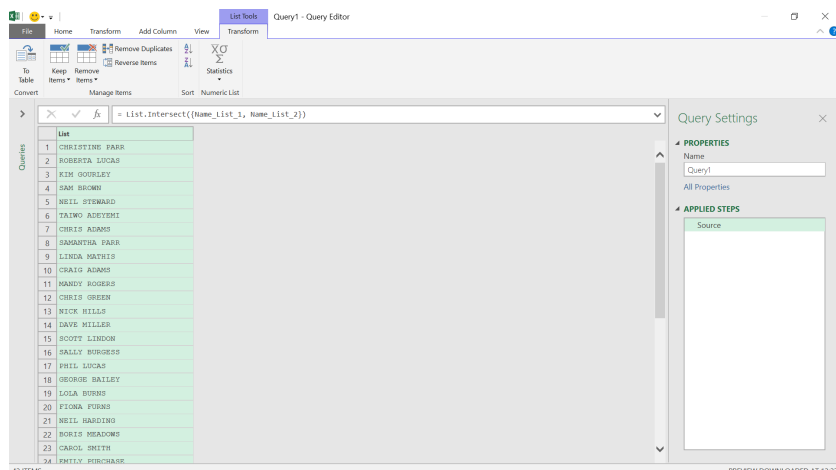
This time we will use **List.Intersect**:

**List.Intersect(list as list, optional equationCriteria as any) as list**

This returns a list from a list of lists and intersects common items in individual lists. Duplicate values are again supported.

The first thing to notice about this function is that it refers to one list in the syntax, but that in fact is a 'list of lists'. This means that you may check multiple lists, not just two. The **equationCriteria** parameter is optional again which would allow you to control what is classed as equal.

Let's begin by creating a blank query, so that I can try out this function against the two lists we have already created.



The syntax used is:

= List.Intersect({Name\_List\_1, Name\_List\_2})

You could easily extend this to intersect a list of more lists. In this case, we have a list of those employees on both lists. Each employee only appears once in my case, however, if an employee appeared on both lists twice, then they could appear here twice. If no one appeared on both lists, then the list would be empty.

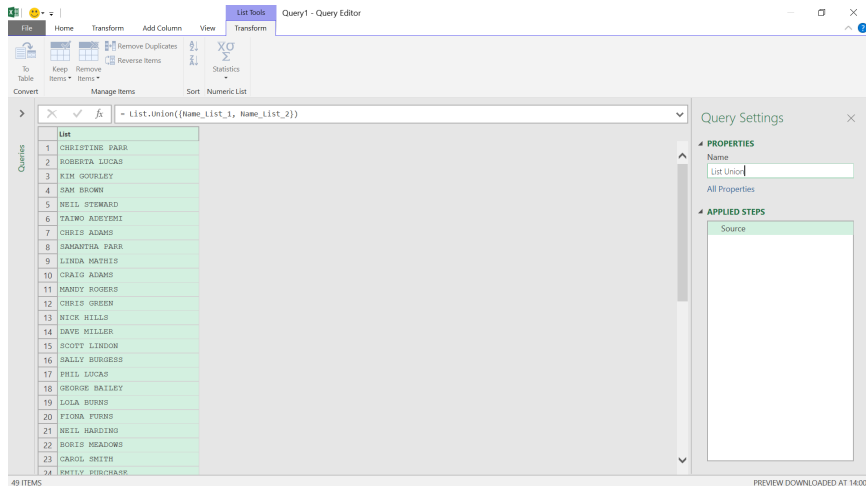
### 3. Combine My Lists into One List

For this we will use the List.Union function:

List.Union(list as list, optional equationCriteria as any) as list

This returns a list from a list of lists and creates a union of the items in the individual lists. The returned list contains all items in any input lists. Duplicate values are matched as part of the union.

This is fairly similar to the List.Intersect functionality, except that we wish to join the lists instead of finding what is common to them.



The syntax used this time is:

= List.Union({Name\_List\_1, Name\_List\_2})

which has joined the data in my lists. Each of the names appears once unless a name appears twice on both lists. You may extend this to include more lists by adding lists to the 'list of lists'.

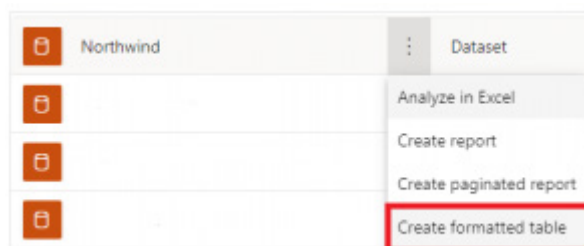
More next month!

## Exportable Formatted Data Tables in Power BI Service (Preview)

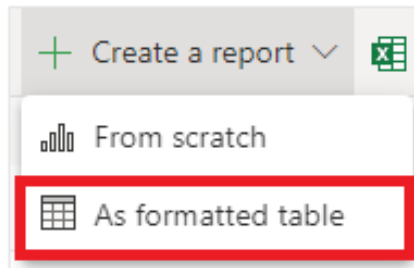
Not mentioned in the main Power BI updates (see below), late May has also seen Microsoft announce the long-anticipated public Preview of a "low-code web authoring experience" for exportable formatted data tables. For the first time, you will be able to author a simple formatted table in Power BI Service. This allows you to use Power BI datasets to edit the final output by contrast to the design first, then adopt an

approach that Report Builder requires. It can even add styling!! When you export data, the styling is preserved. Furthermore, since a paginated report is created under the covers, the 150,000 Excel and 30,000 csv row limits do not apply.

To initiate, in Data hub view, select More options (...) next to a Dataset -> Create formatted table.

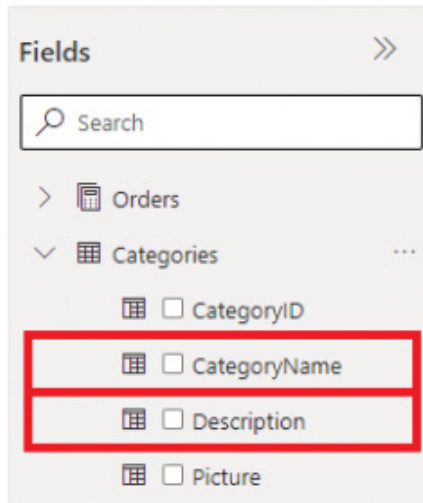


Alternatively, you may click on a dataset, and select 'As formatted table', viz.

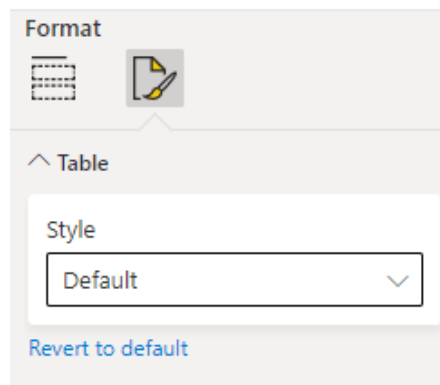


The new paginated report online editing experience, formatted table, will open. There are several associated panes which can help tailor / build your formatted table.

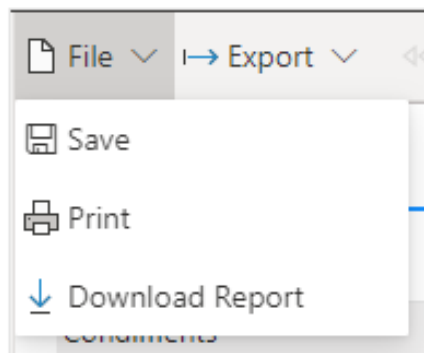
To create your table, select field names in the Fields pane on the right. This pane gives you a table-and-column based view of the chosen dataset. When you find a column that you wish to add to your table, select the column or drag it into the pane labelled 'Build'.



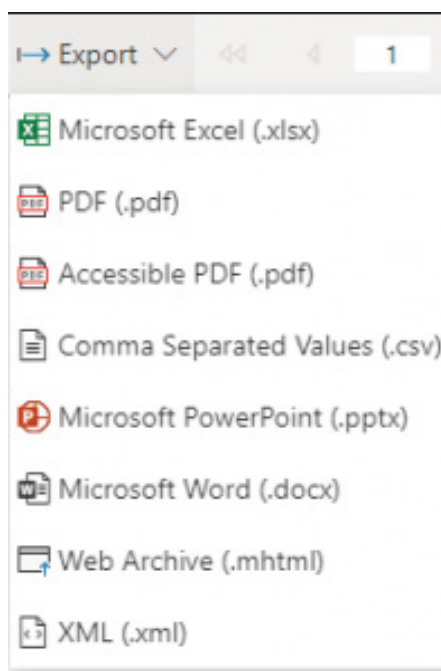
The Build pane has other customisation options for you to create a better looking table. To access these options, select the Format tab at the top of the Build pane.



On the File menu, you may save, download or print your report.



On the Export menu, you can choose the file format you prefer:



You've just created your first paginated report in the Power BI Service!

Apparently, Microsoft's plans don't stop here. The intention is to move forward, providing the ability to create formatted tables from data in Power BI visualisations and datasets with the ability to filter the data, resize columns, control fonts and colours and export whole tables of data to Excel, PDF or PowerPoint.

Watch this space.

## Power BI Updates

Talking of Power BI updates, this month's (other?) highlights include the announcement of the General Availability of Canvas Zoom, more new Format pane updates and various new Preview features. The full list is as follows:

### Reporting

- New Format pane updates in Preview
- Canvas Zoom now Generally Available
- Field parameters in Preview
- Managing composite models on Power BI datasets
- Data point rectangle select now Generally Available
- Error bars for Column and Line Combination charts in Preview
- ArcGIS for Power BI visual updates

### Data connectivity and preparation

- FactSet RMS (New Connector)
- Funnel (New Connector)
- Azure Databricks (Connector Update)
- Denodo (Connector Update)
- Google Sheets (Connector Update)
- Autodesk Construction Cloud (Connector Update)

### Service

- Managing Power BI subscriptions now easier
- Dataset hub improvements
- Deprecation of a Featured Dashboard as your default landing page
- Deprecation of the Dashboard Performance Inspector

### Mobile

- Goal updates now available in the activity feed
- Windows app: new minimum OS requirement

### Developers

- File downloads API
- Expand collapse new attributes

### Visualisations

- New visuals in AppSource
- Dumbbell Bar Chart by Nova Silva
- Drill Down Map PRO by ZoomCharts
- Charticulator now Generally Available

### Other

- Power BI Desktop infrastructure update (WebView2).

Let's now go through each in turn.

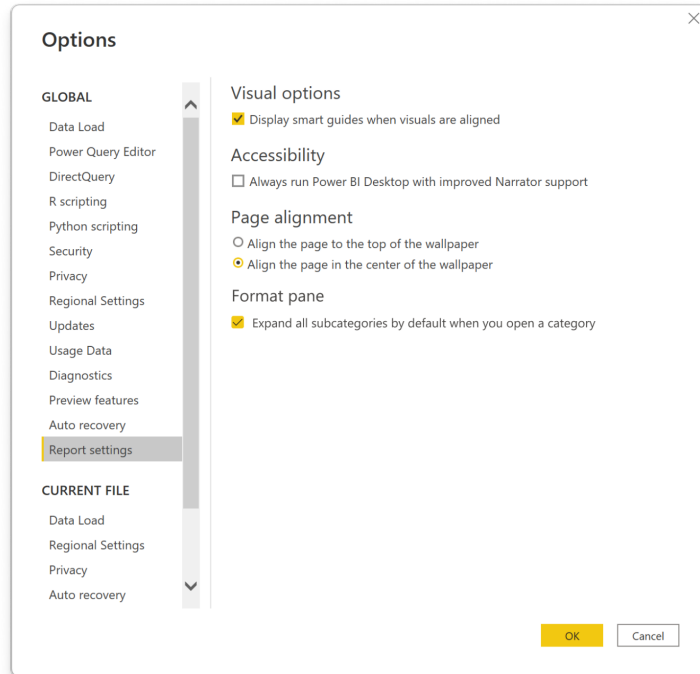


## New Format pane updates in Preview

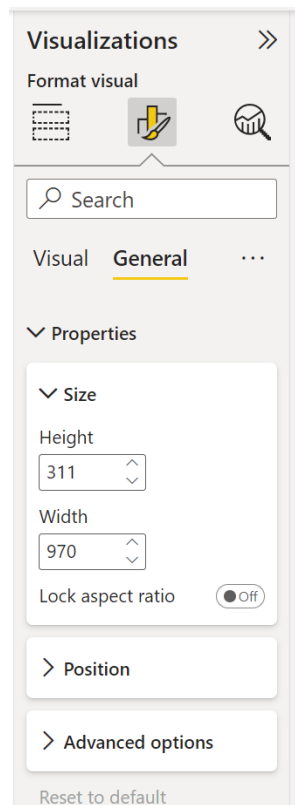
✓ New format pane [Learn more](#) | [Share feedback](#)

Apparently, the much-touted General Availability is coming out sometime later this month (June). Apparently, part of the reason for the delay has been a plethora of feedback regarding the new sub-categories requiring extra clicks. Microsoft has relented and they have now added

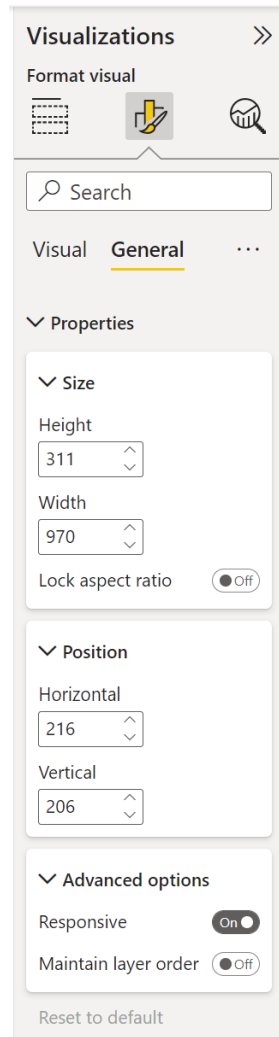
a user preference setting to allow all sub-categories to remain expanded when you open a card. This should feel familiar to those not wanting to say goodbye to the old Format pane behaviour.



The current default will remain with the setting set to "off", which will mean that only the first sub-category will be expanded by default, in order to allow for quick scanning of contents, viz.

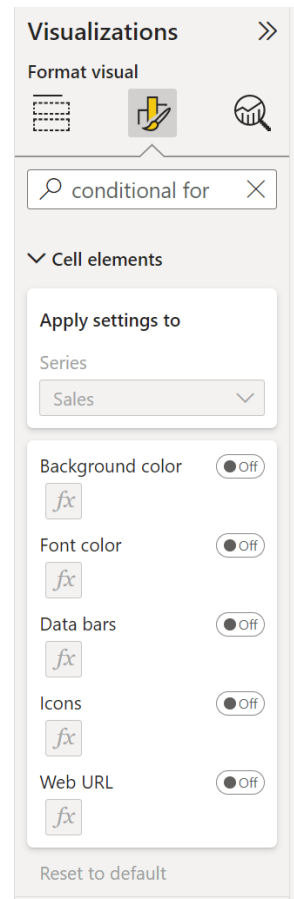


However, if end users prefer, the setting may be turned “on”, so that all subcategories will be expanded, meaning a reduction in extra clicks:



That’s not all though. Other improvements added this month include:

- Analytics pane support re-added for custom visuals
- **‘No fill’ for Title -> background, ToolTips -> background and Header icons -> Help ToolTip -> background color pickers** have been re-added
- for easier searching, Power BI has added ‘color’ back to all colour swatches
- Conditional formatting card has been re-added for Decomp Tree visual
- Matrix and Table now match on ‘conditional formatting’ search term for the ‘Cell Elements’ card. Furthermore, to assist with adjusting to the new naming conventions, Microsoft has added an alias for the renamed ‘Conditional Formatting’ card now called ‘Cell Elements’ when using search.

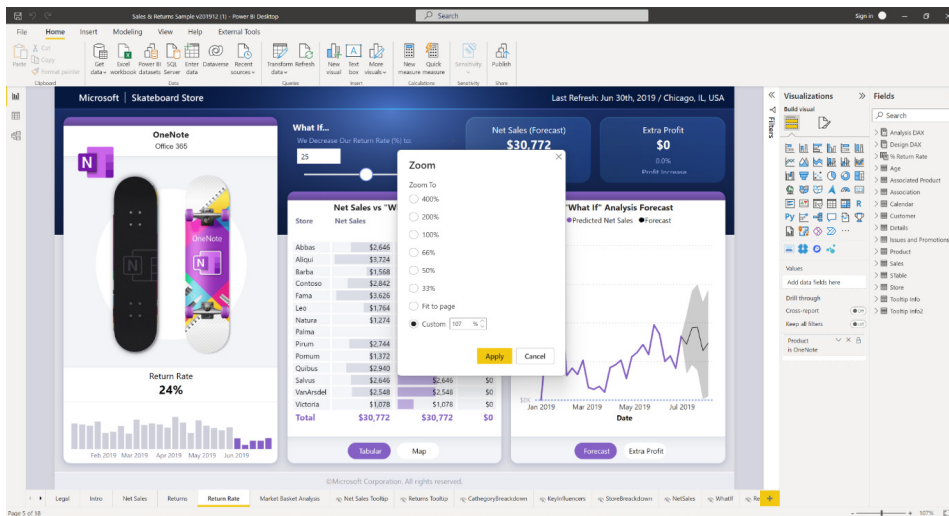
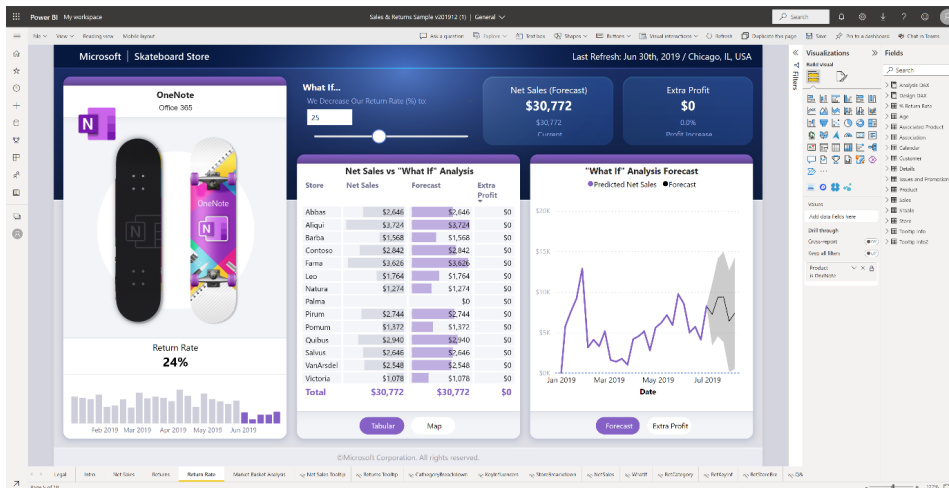
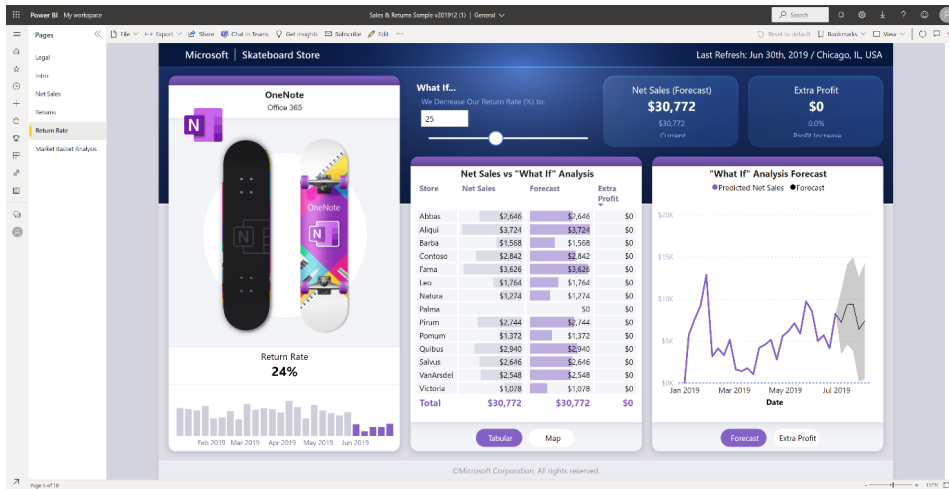


## Canvas Zoom now Generally Available

This update has added the ability to zoom on the canvas. For report readers, this is especially important for improving readability. For those creating reports, this helps magnify the canvas to make pixel perfect tweaks. Users may now drag the slider to set the zoom level or click on the zoom percentage (%) to launch the 'Zoom Level' dialog and type in a custom input. There is even a new quick 'fit to page' button to get back

to the default view. Be careful though: the zoom level is not saved with the report.

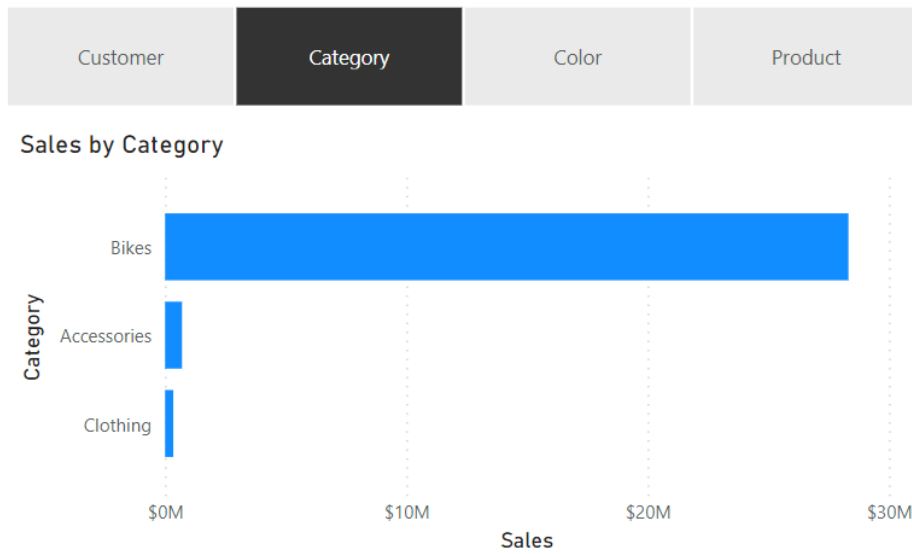
Canvas Zoom has shipped in Desktop, Service read mode and Service edit mode.



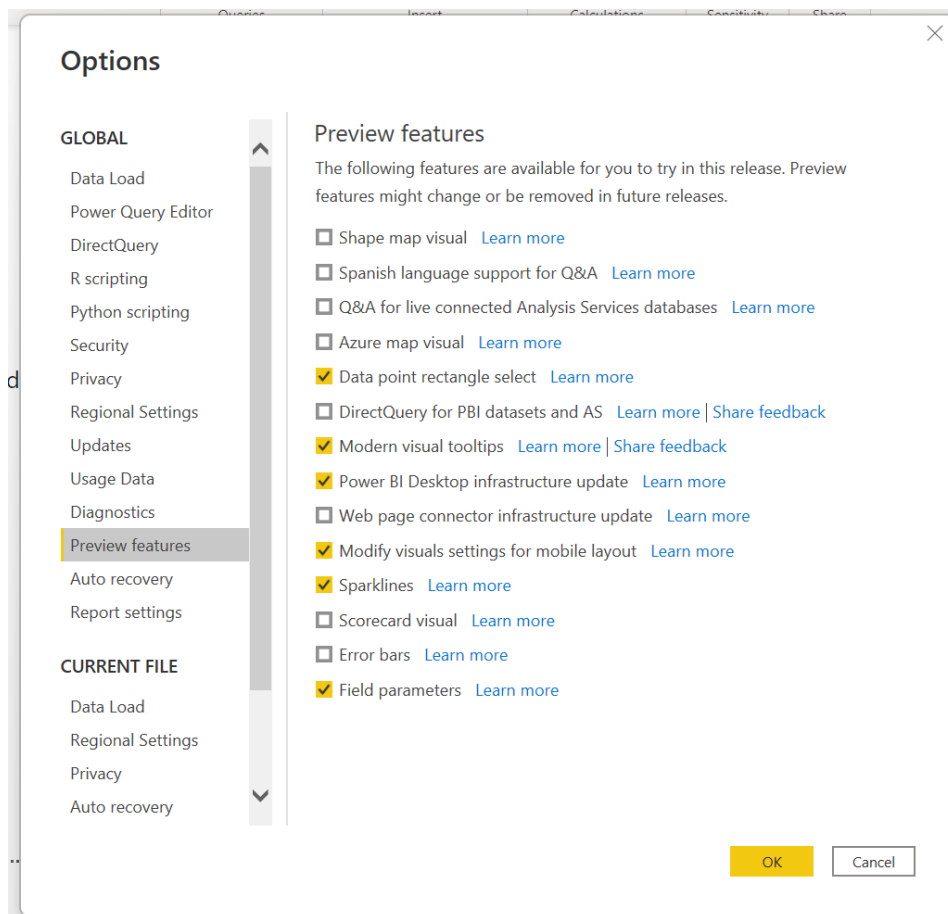
## Field parameters in Preview

This update also introduces a new Preview feature called **field parameters** that will allow users to dynamically change the measures or dimensions being analysed within a report. This feature can help your

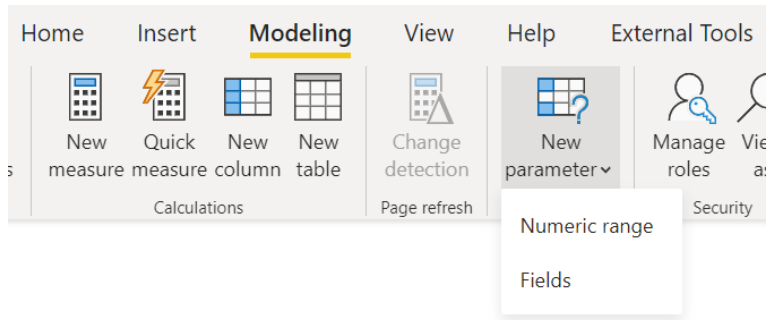
end-users explore and customise the analysis of the report by selecting the different measures or dimensions they are interested in.



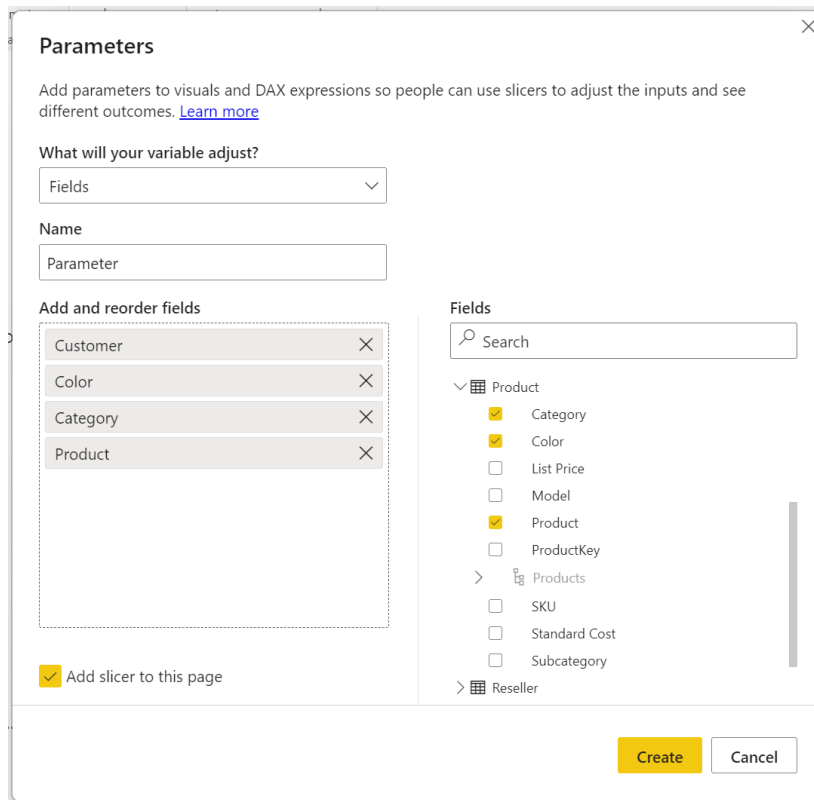
In order to create a field parameter, you will need to first enable the Preview feature called 'Field parameters':



Then, to create a new field parameter, you will need to navigate to **Modeling -> New parameter -> Fields**:



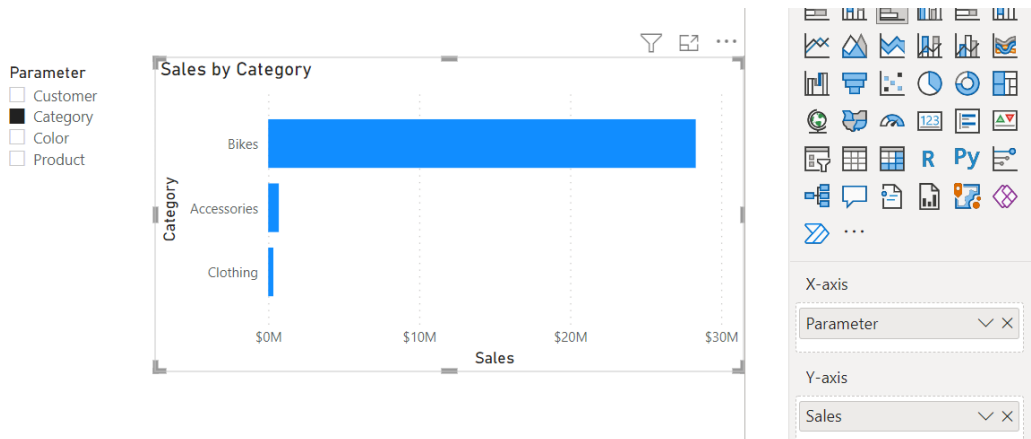
To build the parameter, you will need to provide a name for the parameter and select the fields you want to use. In this example for a field parameter, we've selected different dimensions:



In this dialog, you may drag and drop to change the ordering of the fields or double click on any of the selected fields to change the display name. You may also mix and match different measures and dimensions within the same parameter. For example, this feature can be used to

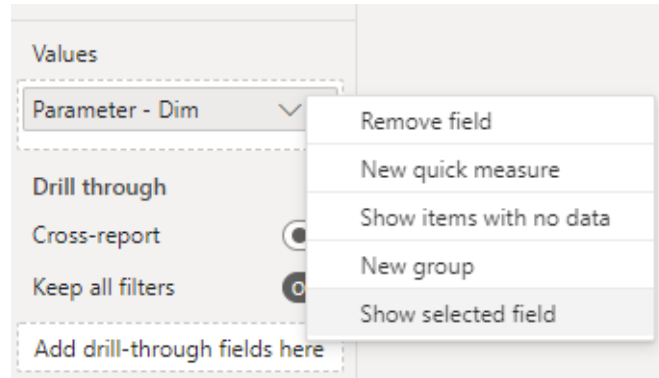
create a dynamic table, whereby the columns may be either measures or dimensions.

Once you've created a field parameter, you can now use the parameter to control the measures or dimensions used in a visual.



You may use the parameter in the field drop zones for a visual. Note that certain visual properties have restrictions on the number of fields that may be used or the types of fields that can be used.

From within the context menu, you can also change whether the field parameter is showing the values of the selected field(s) or the display names of the selected field(s) for all non-slicer visuals, viz.



Finally, if you need to edit any existing field parameters, you will need to modify the DAX directly. For example, if you want to add a new field to an existing parameter you can click **SHIFT + ENTER** to start a new entry:

```

1 Parameter = {
2     ("Customer", NAMEOF('Customer'[Customer]), 0),
3     ("Category", NAMEOF('Product'[Category]), 1),
4     ("Color", NAMEOF('Product'[Color]), 2),
5     ("Product", NAMEOF('Product'[Product]), 3)
6 }

```

You'll need to add a comma between each entry, and you will need to match the following format:

**("<display name of choice>", NAMEOF('<table name>[<field name>]), <ordinal number used for sorting>)**

There are some limitations:

- AI visuals and Q&A are not supported with the feature
- there is no way for end users to select 'none' or no fields option. This is because selecting no fields in the slicer or filter card is the same as selecting all the fields
- you cannot use implicit measures for now, meaning if you need an aggregated column as one of your fields, then you would need to create an explicit DAX measure for it
  - o example of implicit measure:  $\sum$  Sales
  - o example of explicit measure: **Measure = SUM('Table'[Sales]).**

### Managing composite models on Power BI datasets

The ability to create a composite model leveraging one or more Power BI datasets or Azure Analysis Services models has been in Preview for quite some time (since December 2020!). Recently, Microsoft announced that they would be changing the required permissions to view reports created using this feature.

Now, tenant administrators can now enable or disable DirectQuery connections to Power BI datasets in the admin portal. While this is enabled by default, disabling it will effectively stop users from publishing new composite models on Power BI datasets to the Service. As is common with these settings, you may also specify whether you want the setting to apply to specific security groups or to the whole organisation.

Allow DirectQuery connections to Power BI datasets  
*Unapplied changes*  
 DirectQuery connections allow users to make changes to existing datasets or use them to build new ones. [Learn more](#)

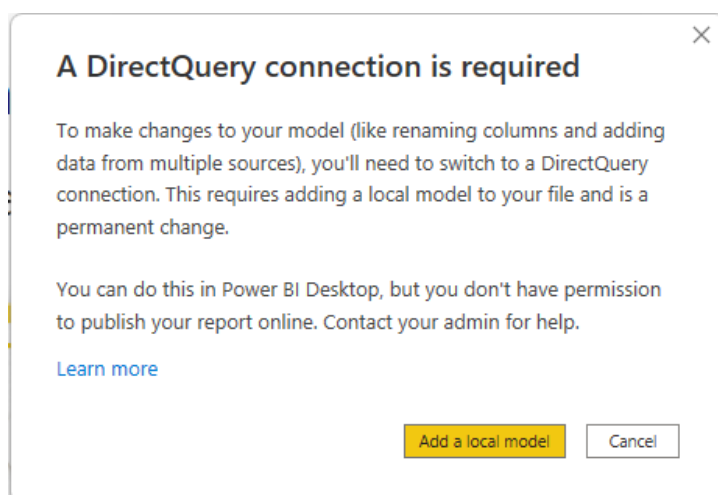
Enabled

Apply to:

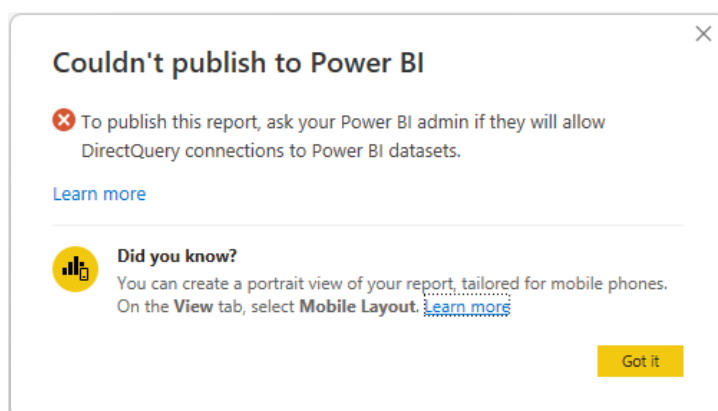
The entire organization  
 Specific security groups  
 Except specific security groups

This setting is enabled by default. If disabled, any reports that leverage a composite model on a Power BI dataset which were already published to the Service will continue to work.

If the setting is disabled, Power BI Desktop will show the following when you create a DirectQuery connection to the Power BI dataset by selecting **Make changes to this model**:



This way, you may still explore the dataset in your local Power BI Desktop environment and create the composite model. However, you will not be able to publish the report to the Service. When you publish the report and model you will see the following error message and publication will be blocked:



Note that neither live connections to Power BI datasets are influenced by the switch nor are live / DirectQuery connections to Azure Analysis Services. These will continue to work regardless of whether the switch

has been turned off. Also, any published reports that leverage a composite model on a Power BI dataset will continue to work even if the switch has been turned off after they were published.

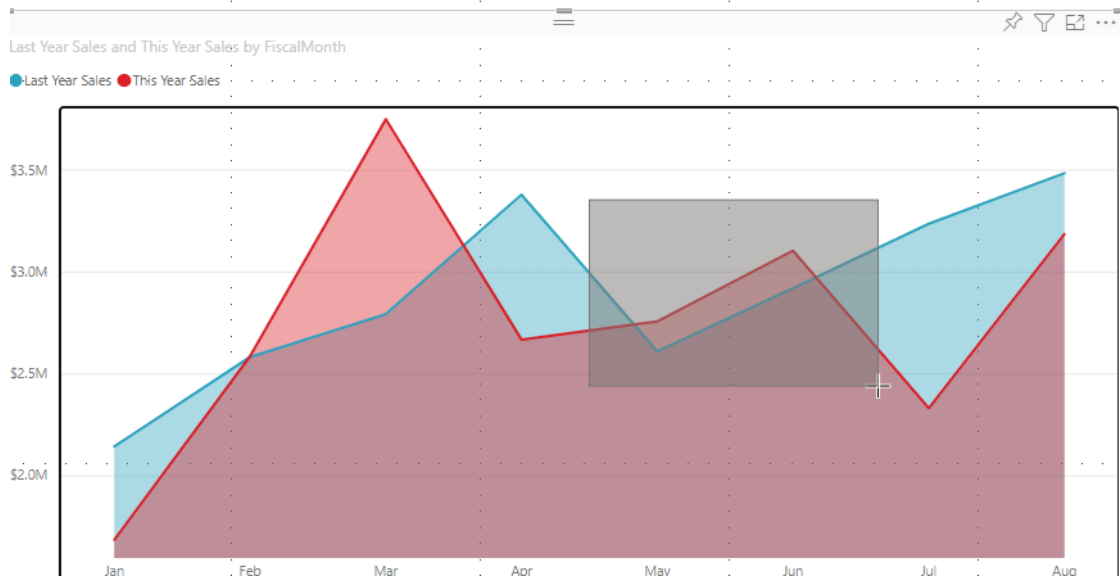
### **Data point rectangle select now Generally Available**

The data point rectangle select is now Generally Available. With this feature, you can now multi-select data points by clicking and dragging over a supported visual.

As a reminder of its operation, when editing a report, you can create a selection rectangle by holding down **CTRL** and clicking and dragging within a visual. Letting go of the mouse will select all points overlapping the selection rectangle. Your previous selections will be preserved, and already-selected data points will be unselected: it will be as though you held down the **CTRL** key and individually clicked every single point that overlaps the selection rectangle. You can also click and drag while holding down the **SHIFT** key instead. This will only add data points

to your selection without deselecting any points. As always, you can clear your current selection by clicking an empty space on the plot area (without holding down any key). Additionally, when viewing a report, you may create a selection rectangle by clicking and dragging across a visual, even when you have no other keys held down.

This update also introduces keyboard controls to help you access data point rectangle select even without a mouse. Pressing the 'S' key while focused on the plot area or a data point will enter rectangle select mode, displaying a crosshair on the visual. You can move the crosshair using the arrow keys, and speed up that movement by holding down the **SHIFT** key.



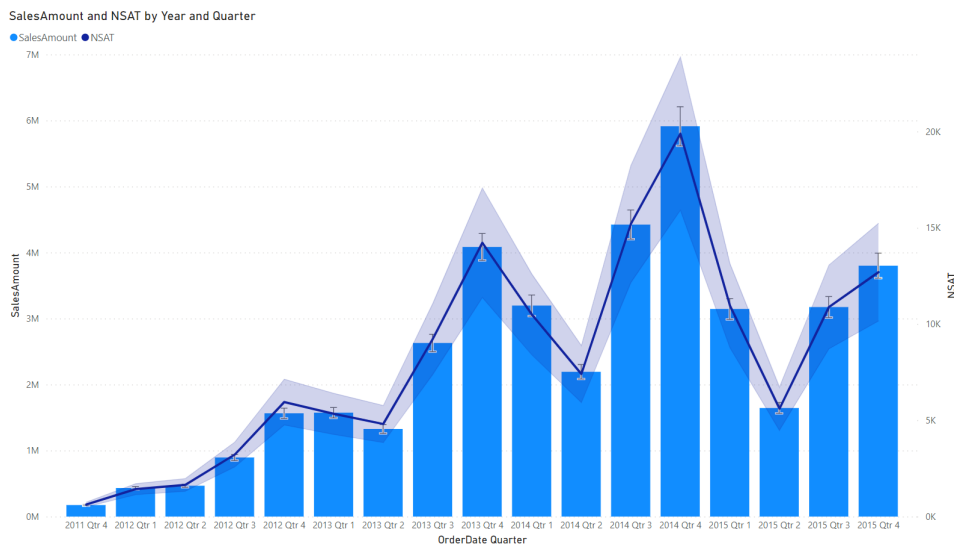
Then, when you're ready to start drawing the rectangle from your cursor's position, hold down the **SPACE** key and use the same crosshair movement controls to create the selection rectangle. The selection is completed once you let go of the **SPACE** key (you may also now clear selections by pressing **CTRL + SHIFT + C**).

Do keep in mind that data point rectangle select is available for line, area, scatter, Treemap and map visuals, and that there is a 3,500 data point limit for the number which you may select at once.

### Error bars for Column and Line Combination charts in Preview

With all the recent work on error bars, it's probably no surprise that Microsoft has brought error bars to combination charts now too. After enabling the Preview feature, you may create error bars using the same method as before: drag upper and lower bound fields into the field

wells in the error bars card of the Analytics pane. You'll find the same formatting options for the line and column portions of the visual as their respective chart types.



### ArcGIS for Power BI visual updates

The ArcGIS for Power BI visual is updated regularly to provide new features, improved speed and usability, and bug fixes. The 2022.2 update latest release, includes performance improvements and bug fixes

for a variety of features, including reference layers, infographics, geo-search and pins, sign-in and ToolTips.

### FactSet RMS (New Connector)

The FactSet Power BI Data Connector leverages the power of FactSet's IRN API to integrate research data into Power BI. This allows users the flexibility and control to customise how they consume FactSet's IRN using Power BI's data visualisations. Other data sources may also be integrated too.



## Funnel (New Connector)

Funnel makes it easier to report on your marketing performance. You can collect data from all platforms across the customer journey: advertising, analytics, social and CRM. Then, simply transform and clean your data

with recommended and customisable rules. You can then send your data to Power BI, helping you to visualise and uncover new insights.

## Azure Databricks (Connector Update)

This connector update ensures system proxy settings are now applied to Databricks connections.

## Denodo (Connector Update)

The new version adds:

- the possibility to use Value.NativeQuery in the Power Query Editor in order to manually set a native SQL query to be used
- the ability to use a connection string instead of a DSN at configuration time
- compatibility with .pbids files.

## Google Sheets (Connector Update)

The Google Sheets connector is now Generally Available and no longer in Beta.

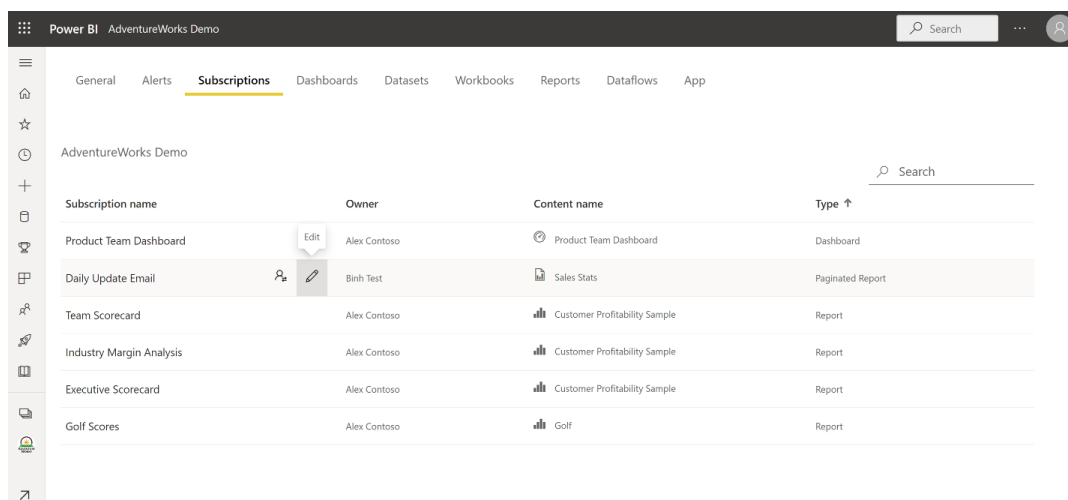
## Autodesk Construction Cloud (Connector Update)

This update includes the IsRetry option on Web.Contents as well as some response handling for 500 status codes.

## Managing Power BI subscriptions now easier

Microsoft has announced improvements to the subscription management experience that provides workspace administrators with new oversight and management capabilities over all subscriptions created against artifacts in a given workspace. If assigned the Admin role in a workspace,

you can now view all subscriptions that have been created for Power BI reports, dashboards or paginated reports in that workspace, regardless of owner. Information on the subscription name, owner, report or dashboard name, and content type is provided.



The screenshot shows the 'Subscriptions' tab in the Power BI interface. The table lists various subscriptions with columns for Subscription name, Owner, Content name, and Type. An 'Edit' button is visible for the 'Product Team Dashboard' subscription.

Subscription name	Owner	Content name	Type
Product Team Dashboard	Alex Contoso	Product Team Dashboard	Dashboard
Daily Update Email	Binh Test	Sales Stats	Paginated Report
Team Scorecard	Alex Contoso	Customer Profitability Sample	Report
Industry Margin Analysis	Alex Contoso	Customer Profitability Sample	Report
Executive Scorecard	Alex Contoso	Customer Profitability Sample	Report
Golf Scores	Alex Contoso	Golf	Report

In addition to viewing all subscriptions associated with a workspace, as a workspace administrator, you may now also make edits or take over subscriptions that are not your own.

If you are not a workspace administrator, but instead a workspace member, contributor or viewer, you will still have access to this updated

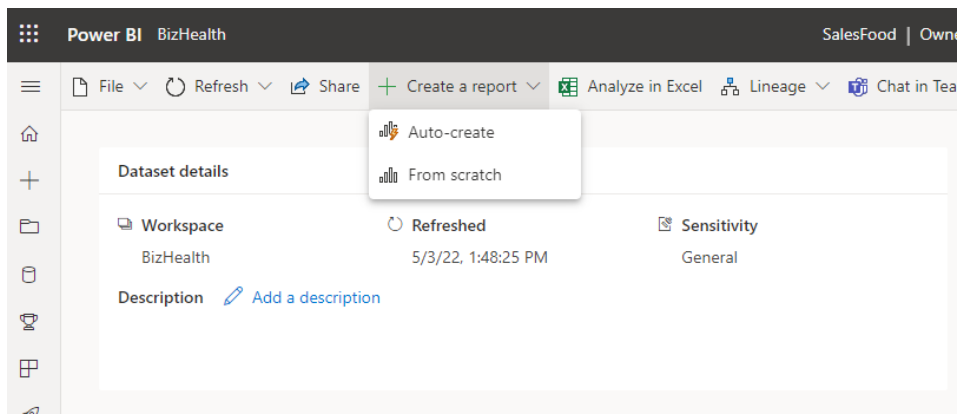
subscription management page, but will only be able to see your own subscriptions. Whether you are a workspace administrator or not, you can search through subscriptions by keyword as well as sort any of the columns alphabetically.

## Dataset hub improvements

The latest version of Power BI has updated the dataset details page to make it easier to gain insights on top of your data.

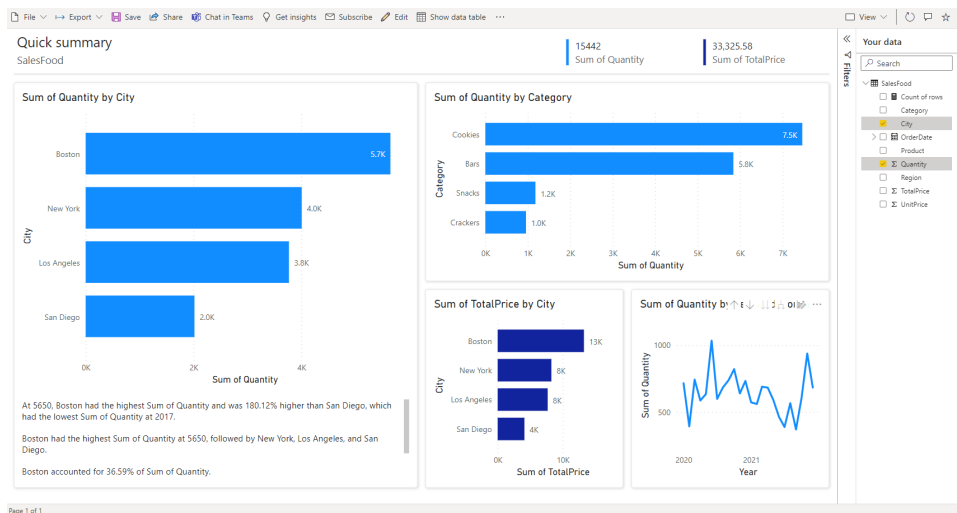
Once you discover a dataset in the datasets hub, click on the dataset to

open the dataset details page. From within the dataset details page, you can start exploring the data by using the new capabilities we have added, as described below.



You can now quickly create a report starting from an existing dataset. When you do that, Power BI autogenerates a summarised view of your data. These autogenerated visuals propel you from raw data to insights quickly. Changing the data you see in the report is straightforward. Use

the 'Your data' pane to add or remove fields from the report. Then, select and unselect fields to update what you want to measure and analyse. Power BI automatically plots meaningful charts based upon the fields you select.



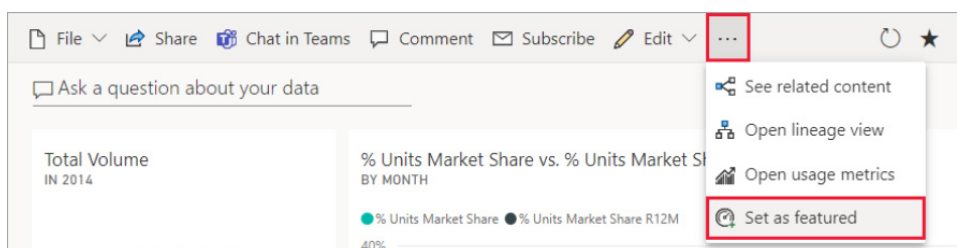
Note: this functionality is currently available only for datasets with a single table.

Furthermore, the new authoring experience for exportable formatted data tables is now available via the dataset hub.

City	Product	TotalPrice
Boston	Amorecort	1918.4
Boston	Banana	179.33
Boston	Bran	873.29
Boston	Carrot	2287.37
Boston	Chocolate Chip	1200.54
Boston	Oatmeal Raisin	3502.96
Boston	Potato Chips	344.4
Boston	Pretzels	585.9
Boston	Whole Wheat	2533.74
Los Angeles	Amorecort	483.9
Los Angeles	Bran	652.63
Los Angeles	Carrot	2233.74
Los Angeles	Chocolate Chip	1565.19
Los Angeles	Oatmeal Raisin	2076.04
Los Angeles	Potato Chips	408.84
Los Angeles	Whole Wheat	146.58

### Deprecation of a Featured Dashboard as your default landing page

The capability to add a featured dashboard as your homepage on the Power BI Service has been deprecated. This will occur between 29 April and 1 June, depending upon your geographic location.



This means that you will no longer be able to add a featured dashboard as your homepage on the Power BI Service. Previously featured dashboards will still be visible on your Power BI Homepage. However, pinned dashboards cannot be added or edited. You will still have the option to remove the featured dashboard.

As an alternative, to ensure you can find your dashboards in the future, consider “favoriting” (*sic*) your pinned dashboards through Power BI. Yuck, I hate *that* word.

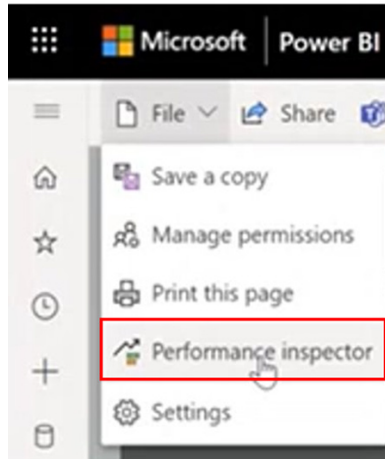
### Deprecation of the Dashboard Performance Inspector

The capability to use your Power BI dashboard’s Performance Inspector in the Power BI Service is also being deprecated between 29 April and 1 June, depending upon your geographic location.

This means that you will no longer be able to use the Performance Inspector in the Power BI Service. Therefore, the support for viewing

which areas are performing well and which areas need improvement for better performance in your Power BI dashboards will be discontinued.

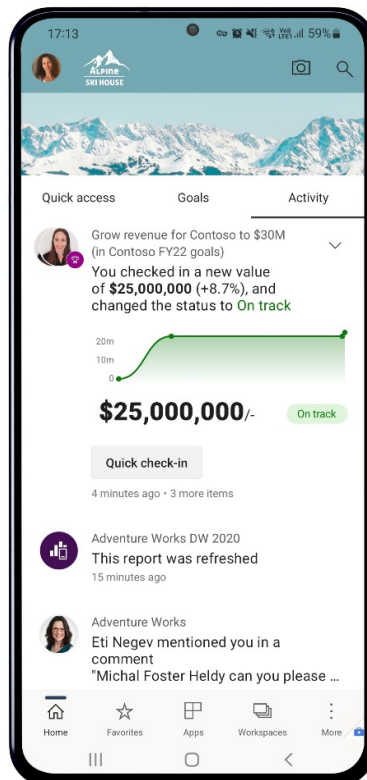
The Performance Inspector within the Power BI Service has no alternate workaround. All Microsoft advise is to “...please make sure to plan accordingly before this feature’s deprecation date”. Great.



### Goal updates now available in the activity feed

The Power BI Mobile app activity feed helps you stay up to date with all the activity and updates for your content. Starting with this release, your activity feed will also include changes and updates to goals. Now you’ll be able to more quickly understand and react to the status and

progress of your goals. Not only can you get right to the goal’s details pane inside the scorecard by tapping on the goal in the activity feed, you can also update your goals directly from the activity feed itself via the Quick check-in button.



### **Windows app: new minimum OS requirement**

In preparation for WebView2 (see below), the minimum Windows OS version required by the Power BI Windows app has changed to Windows 10 version 16299. App upgrades for Windows devices running on earlier OS versions will not be available.

### **File downloads API**

The new File downloads API enables custom visuals to export data from the visual into files. Downloads require user consent and a global admin switch to be enabled. The new API is available with the 4.5 API release.

### **Expand collapse new attributes**

The APIs are “continuously improved” (no time off for good behaviour then). With the new 4.2 API, Microsoft has added two additional attributes to the expand / collapse API for the Matrix dataview:

```
"supportsMerge": {  
  
  "type": "boolean",  
  
  "description": "Indicates that the expansion state should be updated, instead of reset,  
when the query projections change."  
  
},  
  
"restoreProjectionsOrderFromBookmark": {  
  
  "type": "boolean",  
  
  "description": "Indicates that the bookmarked expansion state should be restored even if  
the query projections order no longer matches the expansion state levels."  
  
}
```

### **New visuals in AppSource**

The following are new visuals this update:

- Razor
- Mass Filter
- Preselected Slicer
- Data Refresher
- TMap.

[Is it me or does it feel like we're having a respite..?]

### **Dumbbell Bar Chart by Nova Silva**

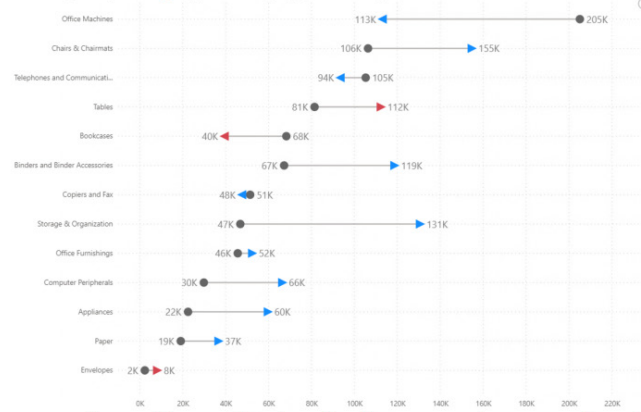
Data visualisations play a fundamental role in answering an important data question, “How does result A compare to result B?”. Typical examples of these questions are:

- How does the sales of this month compare to the sales of last month?
- What is the difference between the number of documents processed this year compared to 2021?
- How does the number of planned-visitors compare to the number of unplanned-visitors at our locations?

Key in answering these kind of questions is clearly visualising the difference between the two results. This is where the Dumbbell Bar Chart proves useful: showing both values and the difference between them.

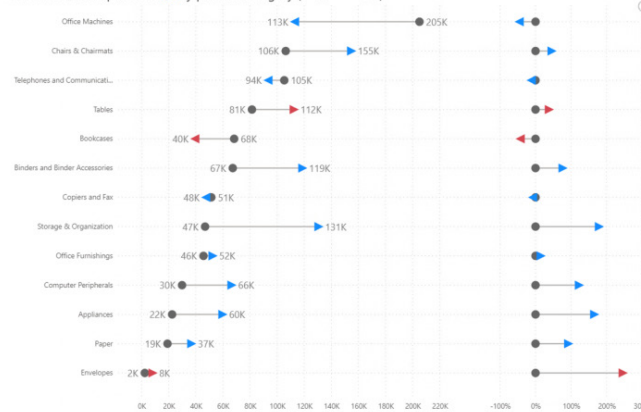
## Dumbbell Bar Chart: Compare two values...

Sales for Q1 compared to Q2 by product category (← or → = loss)



## ...and optionally include a variance chart.

Sales for Q1 compared to Q2 by product category (← or → = loss)





In the latest version, you'll find various features, including data labels and conditional formatting. These are all available in the familiar standard Power BI interface, *i.e.* there is no need to learn any new interface to configure these features.

You may try the Dumbbell Bar Chart by downloading it from the AppSource. All features are available for free to evaluate this visual within Power BI Desktop.

## Drill Down Map PRO by ZoomCharts

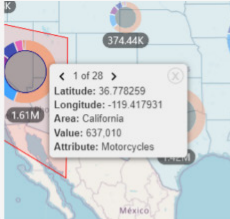
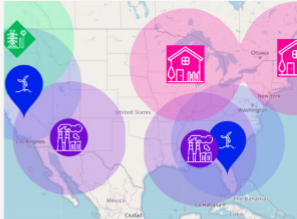
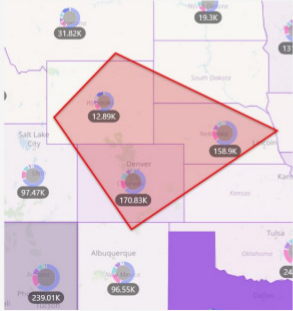
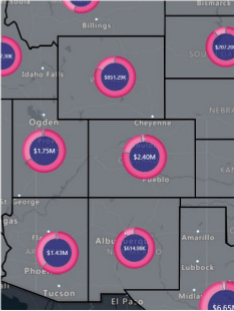
Drill Down Map PRO by ZoomCharts is a custom map visual for Power BI. You may explore location-based data in an intuitive, simple way with on-chart interactions.

### Drill Down Map PRO

- Rich customization options
- Multi-page tooltip
- Filtering with lasso tool
- Clusters as Pie or Donut charts
- User-friendly interactions

zoomcharts.com/powerbi

Features include:

- **custom shape support:** provide your own custom shapes through KML and GeoJSON files
- **lasso tool:** draw your own shapes as filters
- **node clustering capabilities:** clusters can be formatted as donut or pie charts for category display
- **various map layers let you choose from four [4] options:** Azure maps, Custom (OpenStreetMaps, Google, CartoDB etc.), Image (e.g. floor plans) and None (visualise shapes without background)
- Aura, image and custom label support.

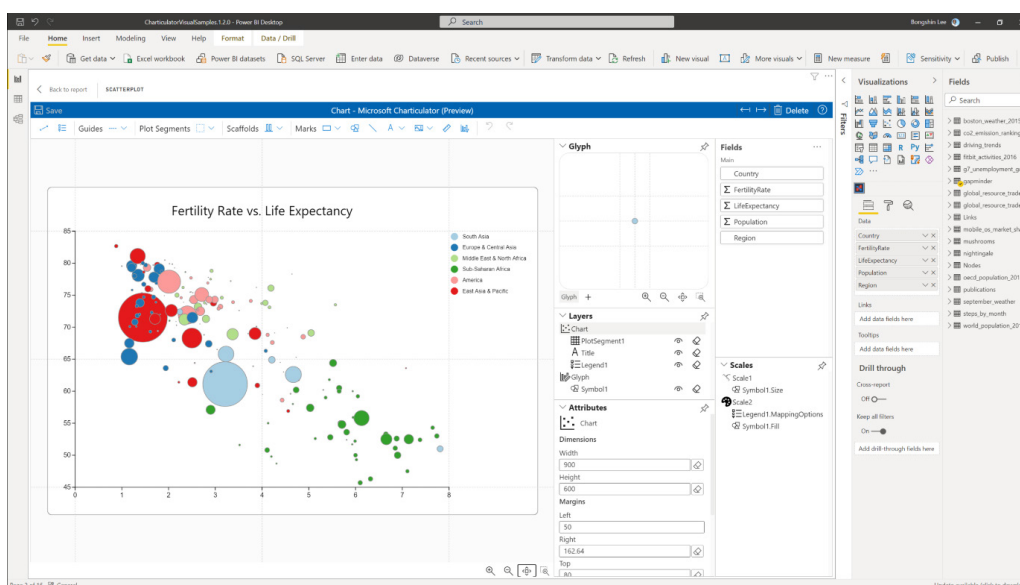
Popular use cases:

- **production:** monitoring production and equipment data by location
- **sales and marketing:** mapping sales results by region or shop location
- **public sector:** visualising environmental and sociodemographic data.

ZoomCharts Drill Down PRO Visuals are known for their interactive drilldowns, smooth animations and customisation options. All Drill Down PRO Visuals support touch input devices, interactions, custom and native ToolTips, filtering, bookmarks and context menu(s).

## Charticator now Generally Available

With the latest version of Charticator, the visual is now Generally Available.



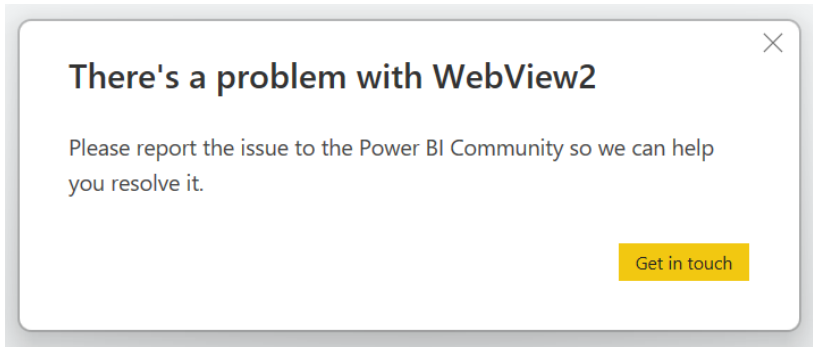
This update has made a variety of modifications and bug fixes with the visual:

- support y-axis label wrapping based on chart margins Cartesian coordinate system
- fixed an issue with text outline and background properties
- fixed an issue with the colour picker
- fixed an issue with the icons in the dropdown menu
- fixed an issue with image opening
- added the begin and end arrow type for the link line
- added orientation and rounded base properties for the Triangle mark
- added search in the Attributes panel
- fixed an issue with categorical tick data and tick format when you rebind data to the axis
- removed the first and last ticks when the baseline is not visible on the axis
- fixed an issue with the colour picker's incorrect display in Plot Segment
- fixed an issue with the nested chart when adding a new field
- fixed an issue with the axis properties in a nested chart.

## Power BI Desktop infrastructure update (WebView2)

Before releasing the switch to WebView2 as part of the infrastructure update, Microsoft has stated that it wants to be sure as many issues as possible have been solved. Therefore, they have fixed several bugs

but have asked for help. Starting with this release, if Power BI detects there is an issue with WebView2, Power BI Desktop will show you the following message:



If you see this message, please do not disregard it, but instead help them fix any outstanding issues by clicking on the 'Get in touch' button or by responding to the community forum post (<https://community.powerbi.com/>).

That's it for this month. See you in July!

## The A to Z of Excel Functions: IMLN



An imaginary number is a complex number that can be written as a real number multiplied by the imaginary unit  $i$  (sometimes denoted  $j$ ) which is defined by its property  $i^2 = -1$ . In general, the square of an imaginary number  $bi$  is  $-b^2$ . For example,  $9i$  is an imaginary number, and its square is  $-81$ . Zero is considered to be both real and imaginary.

An **imaginary** number  $bi$  can be added to a **real** number  $a$  to form a **complex number** of the form  $a + bi$ , where the real numbers  $a$  and  $b$  are called, respectively, the **real** part and the **imaginary** part of the **complex number**.

Sometimes you wish to calculate the natural logarithm of a complex number. The natural logarithm of a complex number is given by

$$\ln(x + yi) = \ln \sqrt{x^2 + y^2} + i \tan^{-1} \left( \frac{y}{x} \right)$$

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

**IMLN(number)**

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

- **number**: this is required and represents the complex number for which you wish to calculate the natural logarithm.

It should be further noted that:

- you should use **COMPLEX** to convert real and imaginary coefficients into a complex number
- **IMLN** recognises either the  $i$  or  $j$  notation
- if **number** is a value that is not in the  $x + yi$  or  $x + yj$  text format, **IMLN** returns the **#NUM!** error value
- if **number** is a logical value, **IMLN** returns the **#VALUE!** error value
- if the complex number ends in  $+i$  or  $-i$  (or  $j$ ), *i.e.* there is no coefficient between the operator and the imaginary unit, there must be no space, otherwise **IMLN** will return an **#NUM!** error.

Please see our example below:

	A	B	C
1	Formula	Description	Result
2	<code>=IMLN("3 + 4i")</code>	Natural logarithm of the complex number 3 + 4i	1.6094379124341+0.927295218001612i

## The A to Z of Excel Functions: IMLOG10



Sometimes you might wish to calculate the common logarithm of a complex number in base 10. The common logarithm (base 10) of a complex number may be calculated from the natural logarithm:

$$\log_{10}(x + yi) = (\log_{10} e) \ln(x + yi)$$

**IMLOG10** returns the common logarithm (base 10) of a complex number that is in the **x + yi** or **x + yj** text format. The function employs the following syntax to operate:

**IMLOG10(number)**

The **IMLOG10** function has the following argument:

- **number**: this is required and represents the complex number for which you wish to calculate the common logarithm (base 10).

It should be further noted that:

- you should use **COMPLEX** to convert real and imaginary coefficients into a complex number
- **IMLOG10** recognises either the i or j notation
- if **number** is a value that is not in the **x + yi** or **x + yj** text format, **IMLOG10** returns the **#NUM!** error value
- if **number** is a logical value, **IMLOG10** returns the **#VALUE!** error value
- if the complex number ends in +i or -i (or j), *i.e.* there is no coefficient between the operator and the imaginary unit, there must be no space, otherwise **IMLOG10** will return an **#NUM!** error.

Please see the example below:

	A	B	C
1	Formula	Description	Result
2	<code>=IMLOG10("3 + 4i")</code>	Common logarithm (base 10) of the complex number 3 + 4i	0.698970004336019+0.402719196273373i

## The A to Z of Excel Functions: IMLOG2





Sometimes you might wish to calculate the base-2 logarithm of a complex number. The common logarithm (base 2) of a complex number may be calculated from the natural logarithm:

$$\log_2(x + yi) = (\log_2 e) \ln(x + yi)$$

**IMLOG2** returns the binary logarithm of a complex number that is in the **x + yi** or **x + yj** text format. The function employs the following syntax to operate:

**IMLOG2(number)**

The **IMLOG2** function has the following argument:

- **number**: this is required and represents the complex number for which you wish to calculate the binary logarithm.

If you're not yet dying of boredom this month, it should be further noted that:

- you should use **COMPLEX** to convert real and imaginary coefficients into a complex number
- **IMLOG2** recognises either the **i** or **j** notation
- if **number** is a value that is not in the **x + yi** or **x + yj** text format, **IMLOG2** returns the **#NUM!** error value
- if **number** is a logical value, **IMLOG2** returns the **#VALUE!** error value
- if the complex number ends in **+i** or **-i** (or **j**), *i.e.* there is no coefficient between the operator and the imaginary unit, there must be no space, otherwise **IMLOG2** will return an **#NUM!** error.

Please see my example below:

	A	B	C
1	Formula	Description	Result
2	=IMLOG2("3 + 4i")	Binary logarithm (base 2) of the complex number 3 + 4i	2.32192809488736+1.33780421245098i
3			

More Excel Functions next month.

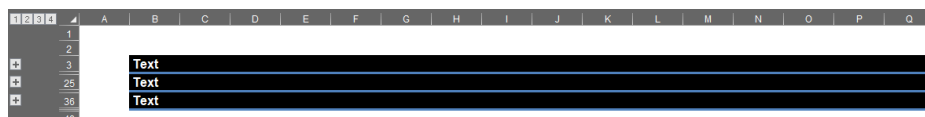
## Beat the Boredom Suggested Solution

To recap, the problem presented earlier in the newsletter was to automate the process of grouping rows based on specific headings. Essentially, we have a worksheet in the following format:

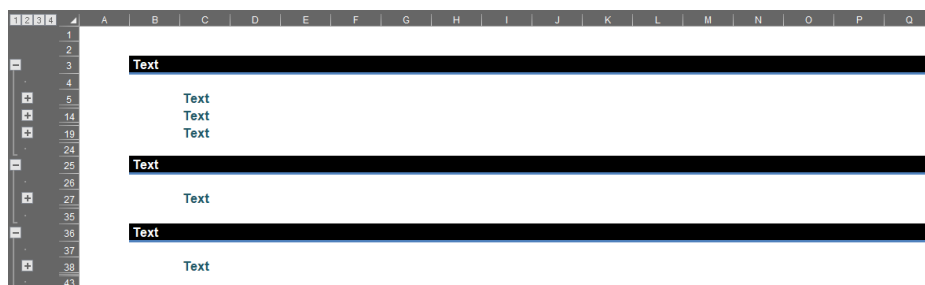
<b>Text</b>													
	Text												
	Text	text	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula
	Text	text	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula
	Text												
	Text	text	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula
	Text	text	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula
<b>Text</b>													
	Text												
	Text	text	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula
	Text	text	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula
<b>Text</b>													
	Text												
	Text	text	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula	Formula

and we want to group the worksheet as:

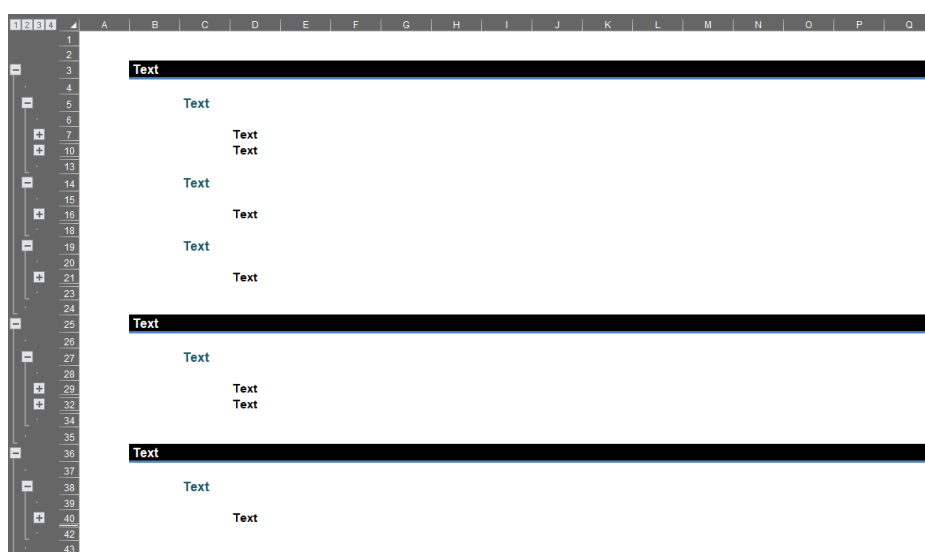
First level:



Second level:



Third level:



### Suggested Solution

One simple way here is to group all the rows with used range to the highest level and then ungroup each heading at different levels.

The first step is to define the variables for integers and ranges.

```
Dim rng As Range
```

```
Dim firstRow, lastRow as Integer
```

If there are existing groupings, we need to expand all rows to avoid potential errors in the following sections of code.

```
ActiveSheet.Outline.ShowLevels RowLevels:=8, ColumnLevels:=8
```

Next, we define the starting row and the last row of formula by using the **UsedRange** method to find the first row and use the total count method to find the last row. The values of rows are assigned to the variables we defined above.

```
firstRow = ActiveSheet.UsedRange.Row
```

```
lastRow = ActiveSheet.UsedRange.Rows (ActiveSheet.UsedRange.Rows.Count) .Row
```

Then, we need to unhide all rows (if any) and clear the existing outline for all rows between the starting row and the last row. Thus, we can remove all existing groupings and get the used range ready for the grouping pattern, as required.

```
Rows(firstRow & ":" & lastRow).EntireRow.Hidden = False
```

```
Rows(firstRow & ":" & lastRow).ClearOutline
```

Next, we group the rows between first row and last row three [3] times to the fourth [4th] level of grouping. We group all the rows at highest level of grouping and then we can start to ungroup each heading. This is more efficient than selectively grouping.

```
Rows(firstRow & ":" & lastRow).Rows.Group
```

```
Rows(firstRow & ":" & lastRow).Rows.Group
```

```
Rows(firstRow & ":" & lastRow).Rows.Group
```

For the third level of headings, we loop through column **D**. If the cell's value in column **D** is not blank, then we ungroup the row. Thus, the headings in column **D** will be grouped at third level.

```
For Each rng In Range("D" & firstRow & ":D" & lastRow)
```

```
    If Not IsEmpty(rng.Value) Then
```

```
        rng.Rows.Ungroup
```

```
    End If
```

```
Next
```

For the second level of headings, we loop through each range in column **C**. If the cell's value in column **C** is not blank, then we ungroup the current row twice and use the **Offset** function to locate the row above and below the current row and ungroup both rows. Specifically, if there is a heading in column **C**, we ungroup the row twice to make the headings grouped at second level. The reason why the rows above

the current row need to be grouped is because we need to keep a gap between the first heading and the second heading. Therefore, the grouping level at this row should be the same as the second level of headings. As for the rows beneath the current row, we need to ungroup them in order to create the gap between the second level of headings and the third level of headings.

```
For Each rng In Range("C" & firstRow & ":C" & lastRow)
```

```
    If Not IsEmpty(rng.Value) Then
```

```
        rng.Rows.Ungroup
```

```
        rng.Rows.Ungroup
```

```
        rng.Offset(1, 0).Rows.Ungroup
```

```
        rng.Offset(-1, 0).Rows.Ungroup
```

```
    End If
```

```
Next
```

For the first level of headings, we loop through the column **B**. If the cell's value in column **B** is not blank, then we ungroup the current row three times to make the headings at first level of grouping. Again, we use the **Offset** function to locate the row above and below the current row and ungroup them twice and once respectively. For the rows above the current row, we need to ungroup twice to make sure that the first levels of headings are adjacent to each other. For the rows beneath the current row, we ungroup them to keep a gap between the

first heading and the second heading. Also, we use error handling in the loop. This is because we need to use error handling in this loop so that if there is no outline in the rows defined above, the macro will not return an error and stop at a specific step in the loop. However, the error handling ensures the macro will ignore the rows without outline and continue grouping until the end of loop. Finally, the error handling is closed off with '**GoTo 0**' syntax.

```
For Each rng In Range("B" & firstRow & ":B" & lastRow)
```

```
    If Not IsEmpty(rng.Value) Then
```

```
        On Error Resume Next
```

```
        rng.Offset(-1, 0).Rows.Ungroup
```

```
        rng.Offset(-1, 0).Rows.Ungroup
```

```
        rng.Offset(1, 0).Rows.Ungroup
```

```
        rng.Rows.Ungroup
```

```
        rng.Rows.Ungroup
```

```
        rng.Rows.Ungroup
```

```
        On Error GoTo 0
```

```
    End If
```

```
Next
```

Finally, we point back to the range **A1**, it helps to reset the worksheet to the top left-hand side.

```
Range("A1").Select
```

To combine the components together we obtain:

```
Sub rowGrouping()  
  
Dim rng As Range  
Dim firstRow, lastRow As Integer  
  
'Expand all grouped rows (if any)  
ActiveSheet.Outline.ShowLevels RowLevels:=8, ColumnLevels:=8  
  
'Define first row and last row  
firstRow = ActiveSheet.UsedRange.Row  
lastRow = ActiveSheet.UsedRange.Rows(ActiveSheet.UsedRange.Rows.Count).Row  
  
'Unhide all rows (if any) and clear the outline  
Rows(firstRow & ":" & lastRow).EntireRow.Hidden = False  
Rows(firstRow & ":" & lastRow).ClearOutline  
On Error Resume Next  
  
'Group the rows from first row to last row to grouping level 4  
Rows(firstRow & ":" & lastRow).Rows.Group  
Rows(firstRow & ":" & lastRow).Rows.Group  
Rows(firstRow & ":" & lastRow).Rows.Group  
  
'Ungroup each row in column D that is not empty  
For Each rng In Range("D" & firstRow & ":" & lastRow)  
    If Not IsEmpty(rng.Value) Then  
        rng.Rows.Ungroup  
    End If  
Next  
  
'Loop through each range in column C that is not empty  
'If the range is not empty, ungroup current row twice  
'If the range is not empty, ungroup the rows above and below the current row  
For Each rng In Range("C" & firstRow & ":" & lastRow)  
    If Not IsEmpty(rng.Value) Then  
        rng.Rows.Ungroup  
        rng.Rows.Ungroup  
        rng.Offset(1, 0).Rows.Ungroup  
        rng.Offset(-1, 0).Rows.Ungroup  
    End If  
Next
```

```

'Loop through each range in column B that is not empty
'If the range is not empty, ungroup the row below current row twice
'If the range is not empty, ungroup the row above current row
'If the range is not empty, ungroup current row three times
For Each rng In Range("B" & firstRow & ":B" & lastRow)
    If Not IsEmpty(rng.Value) Then
        On Error Resume Next
        rng.Offset(-1, 0).Rows.Ungroup
        rng.Offset(-1, 0).Rows.Ungroup
        rng.Offset(1, 0).Rows.Ungroup
        rng.Rows.Ungroup
        rng.Rows.Ungroup
        rng.Rows.Ungroup
        On Error GoTo 0
    End If
Next

'Go back to range A1
Range("A1").Select

End Sub

```

This way, we may group the different level of headings automatically.

There we have it, did you come up with a better solution? Let us know at [contact@sumproduct.com](mailto:contact@sumproduct.com).

Until next time.

## Upcoming SumProduct Training Courses - COVID-19 update

Due to the COVID-19 pandemic that is currently spreading around the globe, we are suspending our in-person courses until further notice. However, to accommodate the new working-from-home dynamic, we are switching our public and in-house courses to an online delivery stream, presented via Microsoft Teams, with a live presenter running through the same course material, downloadable workbooks to complete the hands-on exercises during the training session, and a recording of the sessions for

your use within 1 month for you to refer back to in the event of technical difficulties. To assist with the pacing and flow of the course, we will also have a moderator who will help answer questions during the course.

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

Location	Course	Date	Date	Duration	Duration
Online (Australia)	Power Pivot, Power Query and Power BI	19 - 21 Jul 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	3 Days
Online (Australia)	Excel Tips and Tricks	26 Jul 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	1 Day
Online (Australia)	Financial Modelling	27 - 28 Jul 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	2 Days
Online (Australia)	Excel Tips and Tricks	29 Aug 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	1 Day

Location	Course	Date	Date	Duration	Duration
Online (Australia)	Financial Modelling	30 - 31 Aug 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	2 Days
Online (Australia)	Power Pivot, Power Query and Power BI	28 -30 Sep 2022	09:00-17:00 AEST	(-1 day) 23:00-07:00 GMT	3 Days
Online (Australia)	Excel Tips and Tricks	5 Oct 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	1 Day
Online (Australia)	Financial Modelling	6 - 7 Oct 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	2 Days
Online (Australia)	Power Pivot, Power Query and Power BI	9 - 11 Nov 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	3 Days
Online (Australia)	Excel Tips and Tricks	16 Nov 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	1 Day
Online (Australia)	Financial Modelling	17 - 18 Nov 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	2 Days
Online (Australia)	Power Pivot, Power Query and Power BI	7 - 9 Dec 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	3 Days
Online (Australia)	Excel Tips and Tricks	14 Dec 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	1 Day
Online (Australia)	Financial Modelling	15 - 16 Dec 2022	09:00-17:00 AEDT	(-1 day) 22:00-06:00 GMT	2 Days

## Key Strokes

Each newsletter, we'd like to introduce you to useful keystrokes you may or may not be aware of. This month we thought we'd get straight **DOWN** to this month's keyboard shortcuts:

Keystroke	What it does
DOWN	Move down one cell
ALT + DOWN	Open drop-down (auto-complete, filter or validation)
CTRL + DOWN	Select the last cell in the area down
SHIFT + DOWN	Extend selection down one cell
CTRL + ALT + DOWN	Intel Chipset: Invert screen (turn 180 degrees)
CTRL + SHIFT + DOWN	Extend selection down to last cell in area down

There are c.550 keyboard shortcuts in Excel. For a comprehensive list, please download our Excel file at [www.sumproduct.com/thought/keyboard-shortcuts](http://www.sumproduct.com/thought/keyboard-shortcuts). Also, check out our new daily **Excel Tip of the Day** feature on the [www.sumproduct.com](http://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](mailto: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](http://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](mailto: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](mailto:training@sumproduct.com) for a copy of the brochure or download it directly from [www.sumproduct.com/training](http://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](mailto:contact@sumproduct.com)  
[www.sumproduct.com](http://www.sumproduct.com)  
**+61 3 9020 2071**