ALTRON | KARABINA

Advanced Power BI and DAX Masterclass

Our Credentials

ALTRON | KARABINA

Awards and more

Karabina was founded in 2001. Over 17 years we have received many awards, accolades and recognition, from our partners as well as our customers. These awards show our commitment to delivering the best solutions to our customers. In 2018, Karabina was acquired by the Altron Group and we are now known as Altron Karabina. This change allows us to expand our reach into new markets and deliver our solutions to a larger customer base.

Microsoft Partner

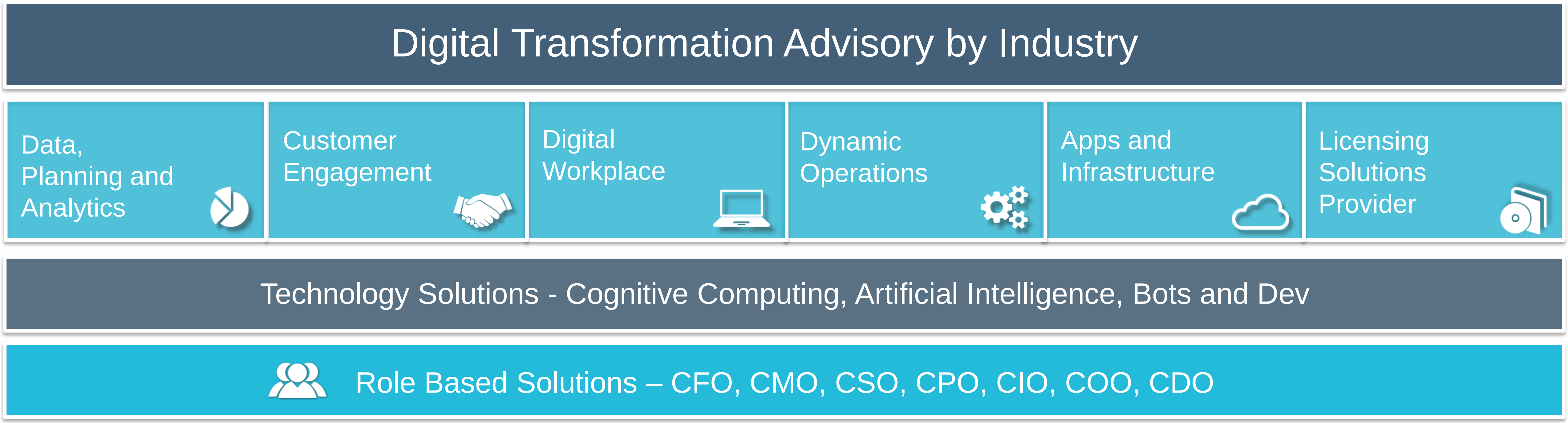
Microsoft

Gold Data Analytics
Gold Enterprise Resource Planning
Gold Cloud Customer Relationship Management
Gold Cloud Productivity
Gold Collaboration and Content
Silver Data Platform

13	13 x Microsoft Data Analytics Partner of the Year
2016	2016 Microsoft Country Partner of the Year
8	8 x Microsoft Dynamics CRM Partner of the Year

Overview of the business

ALTRON | KARABINA



Digital Transformation Advisory by Industry

Data, Planning and Analytics

Customer Engagement

Digital Workplace

Dynamic Operations

Apps and Infrastructure

Licensing Solutions Provider

Technology Solutions - Cognitive Computing, Artificial Intelligence, Bots and Dev

Role Based Solutions – CFO, CMO, CSO, CPO, CIO, COO, CDO

Microsoft Azure

Office 365

Microsoft Dynamics 365

Agenda

- Welcome and sign in:
- Day 1
  - Session 1: DAX Basics, measures, columns and table functions.
  - Lunch break
  - Session 2: Evaluation contexts, CALCULATE() and time intelligence.

- Day 2
  - Session 1: Dashboard design rules, query parameters, what-if analysis, tips and tricks for report interactivity.
  - Lunch Break
  - Session 2: Dynamic security, architecting a tabular model

Course objectives

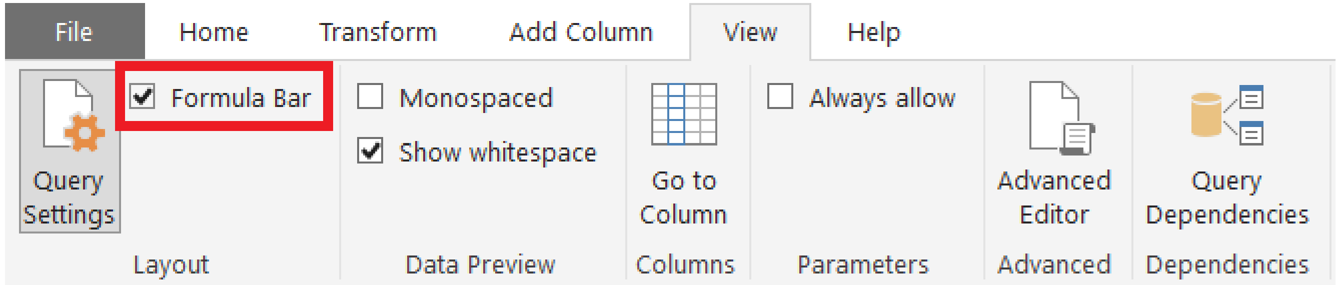
- For users to gain an understanding of the inner workings of the DAX language.
  - This is key to being able to write and debug complex measures.
- To improve the understanding of the CALCULATE() function.
- To bring together the concepts of time intelligence with the effective use of the date table.

Preface – Troubleshooting the Query Editor

- The following is outside of the scope of DAX, and will be dealt with briefly.

Query Editor Formula Bar

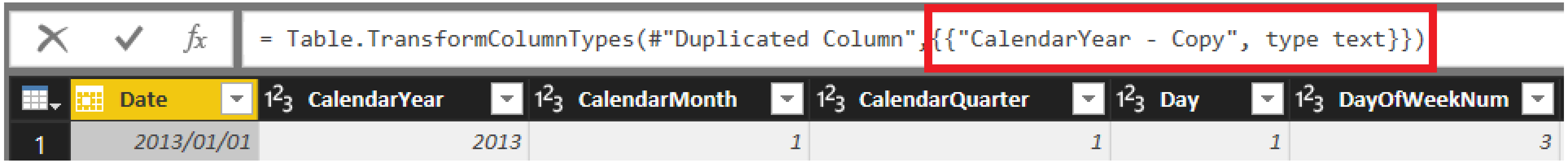
- Enable the formula bar by navigating to:
- *View* tab -> Check *Formula Bar*



Query Editor Formula Bar

- Every query step that is applied results in an M script to perform that action.

- You can edit the M directly:
  - You may want to do this if an applied step returns an error.
  - For example, you can manually edit the data type of column.

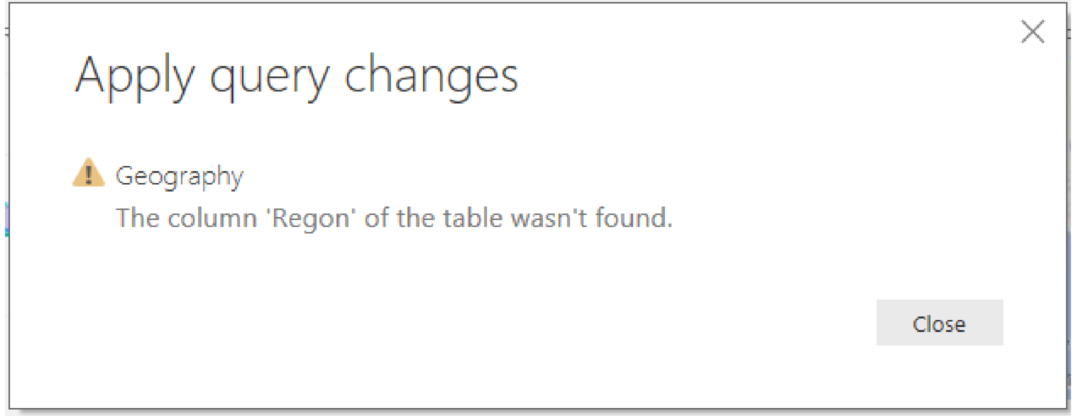


```
= Table.TransformColumnTypes(#"Duplicated Column",{{"CalendarYear - Copy", type text}})
```

	Date	CalendarYear	CalendarMonth	CalendarQuarter	Day	DayOfWeekNum
1	2013/01/01	2013	1	1	1	3

Query Editor errors and troubleshooting

- Errors most commonly occur on data load. Most likely errors:
  - Data type defined for the column is not compatible with the data in the column itself.
  - Column names in the source data may have changed.
  - The path of the source data may have changed.
  - The source of the data may have changed, from .csv to Excel, for example.
- An example of a data load error is shown below.



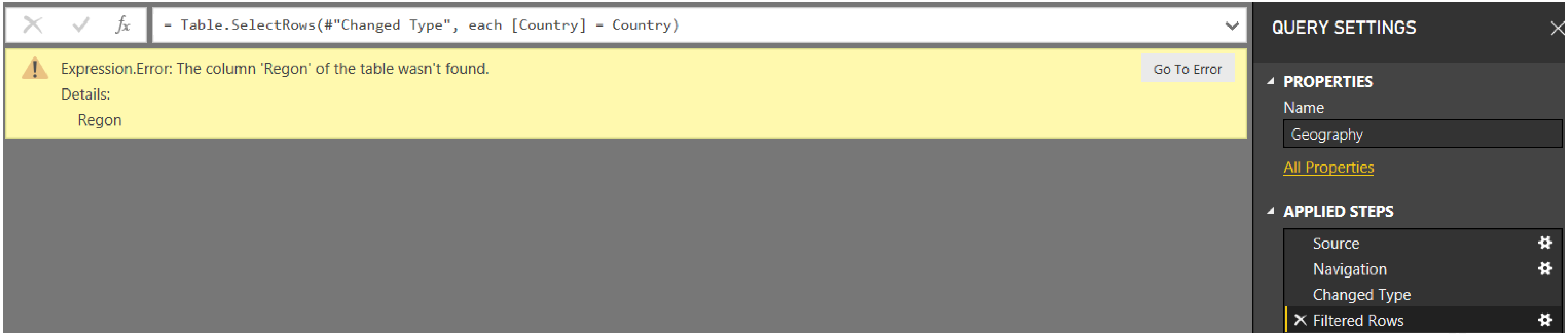
Apply query changes

⚠ Geography

The column 'Regon' of the table wasn't found.

Query Editor errors and troubleshooting

- Go to the Query Editor.
- Starting from the last step, select the step one-by-one, until the yellow error banner disappears. You have now identified the step that resulted in an error.
- In this case, the *Changed Type* step caused the error.
- In this example, the Query Editor cannot find a column in the source data called *Regon.* The column name is clearly misspelled.



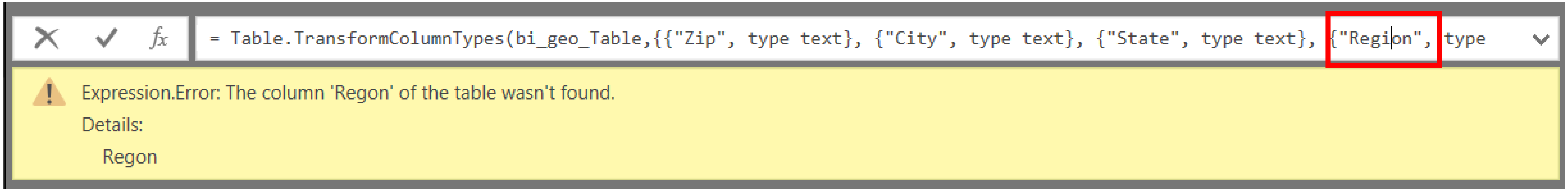
```
= Table.SelectRows(#"Changed Type", each [Country] = Country)
```

Expression.Error: The column 'Regon' of the table wasn't found.
Details:
Regon

Query Editor errors and troubleshooting

- In the Formula Bar, correct the spelling of the name of the column.

- This resolves the error.



```
= Table.TransformColumnTypes(bi_geo_Table,{{"Zip", type text}, {"City", type text}, {"State", type text}, {"Region", type
```

Expression.Error: The column 'Regon' of the table wasn't found.
Details:
Regon

0

DAX

Context of the course

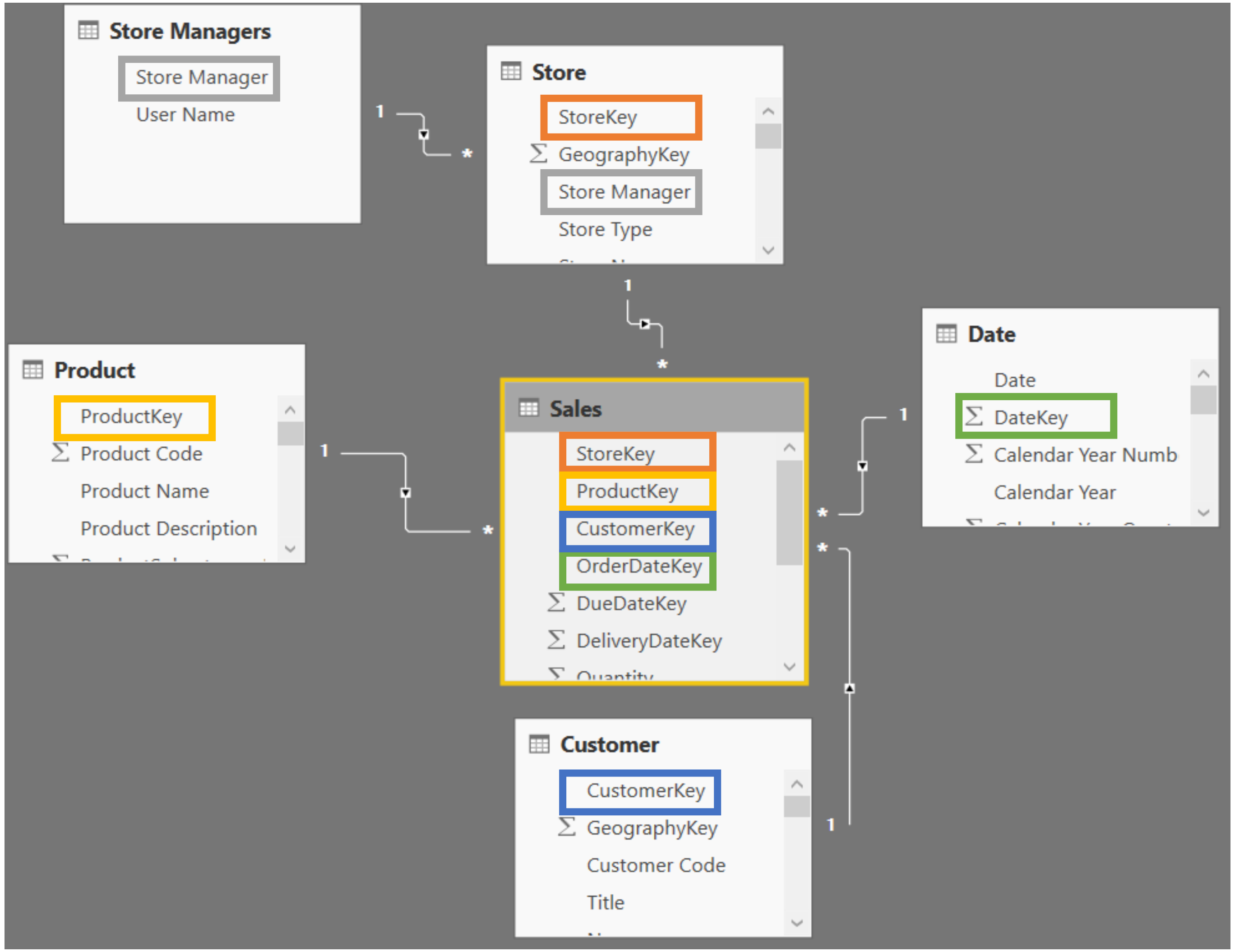
- Data
    - The data being used is excerpts from the Contoso database.
    - This is data for sales of computer hardware and software.

- There are 6 tables
    - Fact:
        - Sales
    - Dimensions:
        - Date
        - Product
        - Store
        - Store Managers
        - Customer

Context of the course – Story

- You are the Head of Business Intelligence for Contoso.
- Contoso has recently implemented a data warehouse on their sales data.
- You need to produce reports that show metrics and measures relevant for the Executive Committee.
- The data is clean, but there are several things that need to be calculated in order to produce the reports.
- Contoso has implemented Power BI as the reporting tool. All calculations must be done using DAX in Power BI.

Context of the course – Data model



Context of the course – Flow of instruction

- This is a more theory intensive course than the Power BI Masterclass.
- Each new concept will be presented, and the theory explained.
- Each theory slide will have a red bar in the top left corner.
- Each concept will have a worked example which ties back to the story.
- Worked examples have a blue bar in the top left corner.
- The expected output is a report containing business related metrics and measures.

Session 1

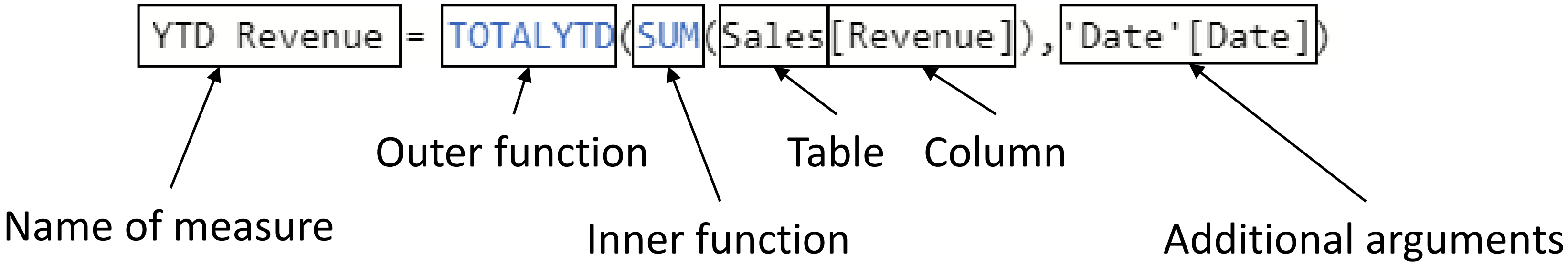
- DAX Basics.
- Measures.
- Columns.
- Table functions.

What is DAX?

- DAX (Data Analysis eXpression) is the programming language of Microsoft SQL Server Analysis Services (SSAS) and Microsoft Power Pivot for Excel.
- It was created in 2010, with the first release of PowerPivot for Excel 2010.
- Over time, DAX gained popularity in the Excel community, which uses DAX to create Power Pivot data models in Excel, and in the Business Intelligence (BI) community, which uses DAX to build models with SSAS.
- DAX is a simple language.
- Once you start to digest these concepts, you will discover that DAX is, indeed, an easy language. It just takes time to get used to.

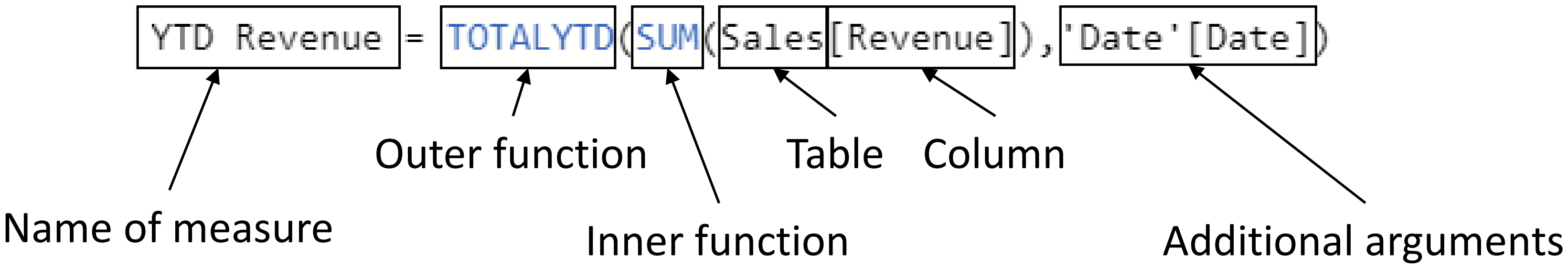
DAX Structure

- DAX is a functional language. Everything is a function call.
  - The concept of statements, loops, jumps and object orientated programming are not present in DAX.



```
YTD Revenue = TOTALYTD(SUM(Sales[Revenue]),'Date'[Date])
```

DAX Function

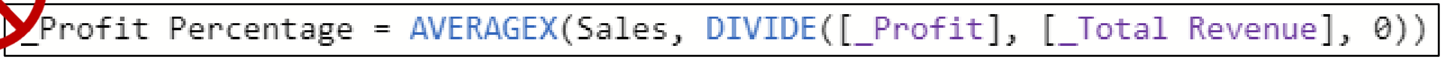


```
YTD Revenue = TOTALYTD(SUM(Sales[Revenue]),'Date'[Date])
```

- Name of measure: User friendly, everything before the ‘=’ sign.
- Inner function: DAX allows you to nest functions. Inner functions are evaluated first. Function names are given in blue text.
- Outer function: Is evaluated after inner functions. It operates on the value returned by the inner functions.
- Functions operate on columns. Values passed into functions are called *arguments*.
- Columns: Column names are always enclosed by square brackets.
- Tables: Always given in front of column names. If the table name is a reserved keyword, or the name has a space in it, the table name needs to have single quotes.

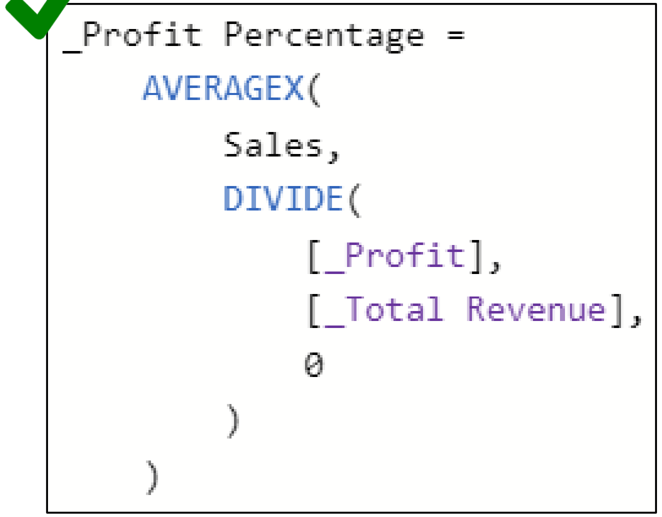
DAX Best Practice

🚫



```
_Profit Percentage = AVERAGEX(Sales, DIVIDE([_Profit], [_Total Revenue], 0))
```

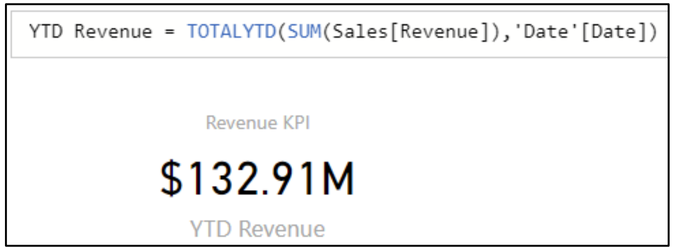
✓



```
_Profit Percentage =
    AVERAGEX(
        Sales,
        DIVIDE(
            [_Profit],
            [_Total Revenue],
            0
        )
    )
```

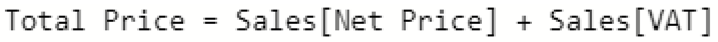
- Indent code for longer expressions.
  - Function nesting is easier to read.
  - Debugging is far easier.
- To add a new line, use Shift + Enter.
- All the arguments of a function should be on a new line with one Tab indent.

Measures vs Columns vs Tables

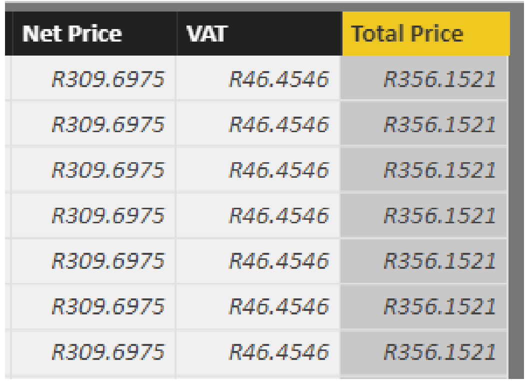


```
YTD Revenue = TOTALYTD(SUM(Sales[Revenue]),'Date'[Date])
```

- Measures return a single, scalar value.
  - These operate over all the values in a column.
  - They require an aggregator.
  - Example: Grand totals, averages, standard deviations.
- Calculated columns return a series of values which are output as a column in a table.
  - They operate over the values in a row-by-row basis.
  - They do not require an aggregator
  - Example: Calculating Total price from the VAT and Net Price columns



```
Total Price = Sales[Net Price] + Sales[VAT]
```



Net Price	VAT	Total Price
R309.6975	R46.4546	R356.1521
R309.6975	R46.4546	R356.1521
R309.6975	R46.4546	R356.1521
R309.6975	R46.4546	R356.1521
R309.6975	R46.4546	R356.1521
R309.6975	R46.4546	R356.1521
R309.6975	R46.4546	R356.1521

Optimiser bar: This bar indicates that this is the optimal way of doing things.

Measures vs Columns

- Measures are computed at query time. This means that the measure is calculated only when a visual containing that measure is created.
  - The DAX compute engine is very efficient; calculations are performed quickly.

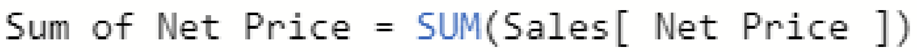
- Calculated columns are computed at refresh time. This means that the column exists as part of the data model.
  - This makes the data model larger.
  - As Power BI operates in-memory, this is less efficient.
  - It would be better to use an iterator measure instead.

- Tables are returned as the result of a Table Function.
  - Only really useful when debugging a DAX expression; to see the results of intermediate steps of a DAX expression.
  - It is far better to have a properly defined data model you are importing into Power BI.

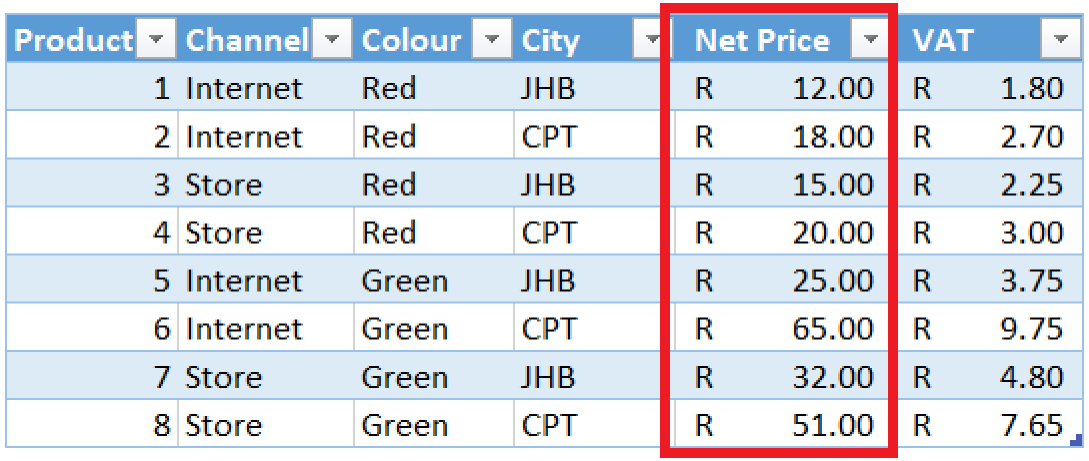
Aggregators

- Aggregators operate over all the values in a column in a single step.
- The input arguments are columns.
- They work exactly like Excel functions.
  - Examples:
    - SUM(Table[Column])
    - AVERAGE()
    - COUNT()
    - COUNTA()
    - MIN()
    - MAX()
    - MEDIAN()

Aggregators



```
Sum of Net Price = SUM(Sales[ Net Price ])
```

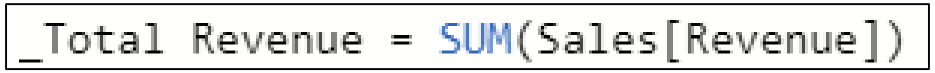


Product	Channel	Colour	City	Net Price	VAT
1	Internet	Red	JHB	R 12.00	R 1.80
2	Internet	Red	CPT	R 18.00	R 2.70
3	Store	Red	JHB	R 15.00	R 2.25
4	Store	Red	CPT	R 20.00	R 3.00
5	Internet	Green	JHB	R 25.00	R 3.75
6	Internet	Green	CPT	R 65.00	R 9.75
7	Store	Green	JHB	R 32.00	R 4.80
8	Store	Green	CPT	R 51.00	R 7.65

- Result: R238.00

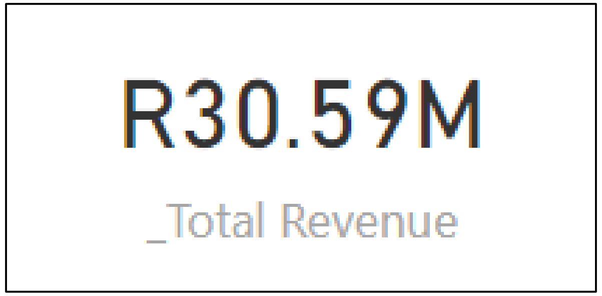
Calculating the Total Revenue – Measure

- In the Sales table, create the following measure:



```
_Total Revenue = SUM(Sales[Revenue])
```

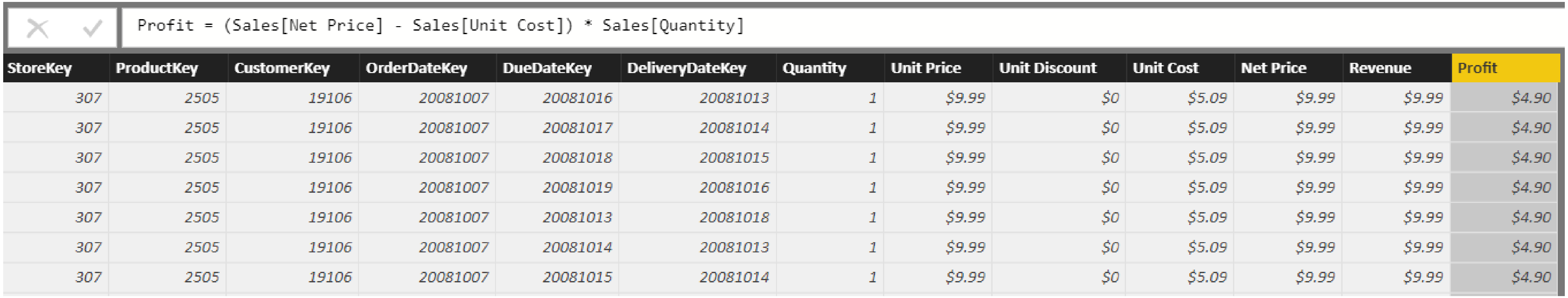
- Top tip: Precede all measures by an underscore “_”. This will ensure all measures appear at the top of the table, making them easier to find.
- Create a Card visual and populate it with the _Total Revenue measure.



R30.59M

_Total Revenue

Calculating the Profit – Calculated Column



```
Profit = (Sales[Net Price] - Sales[Unit Cost]) * Sales[Quantity]
```

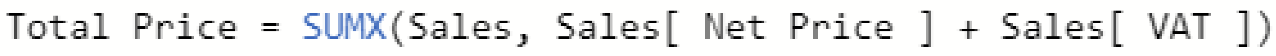
StoreKey	ProductKey	CustomerKey	OrderDateKey	DueDateKey	DeliveryDateKey	Quantity	Unit Price	Unit Discount	Unit Cost	Net Price	Revenue	Profit
307	2505	19106	20081007	20081016	20081013	1	$9.99	$0	$5.09	$9.99	$9.99	$4.90
307	2505	19106	20081007	20081017	20081014	1	$9.99	$0	$5.09	$9.99	$9.99	$4.90
307	2505	19106	20081007	20081018	20081015	1	$9.99	$0	$5.09	$9.99	$9.99	$4.90
307	2505	19106	20081007	20081019	20081016	1	$9.99	$0	$5.09	$9.99	$9.99	$4.90
307	2505	19106	20081007	20081013	20081018	1	$9.99	$0	$5.09	$9.99	$9.99	$4.90
307	2505	19106	20081007	20081014	20081013	1	$9.99	$0	$5.09	$9.99	$9.99	$4.90
307	2505	19106	20081007	20081015	20081014	1	$9.99	$0	$5.09	$9.99	$9.99	$4.90

- The problem with a calculated column is that the calculated values live in the data model, even when you don’t need it.
- To avoid the memory overhead this brings, it is far better to use an iterator.
- Additionally, knowing the profit for a single sale isn’t that useful. It is better to know an aggregate of profit, or profit percentage.

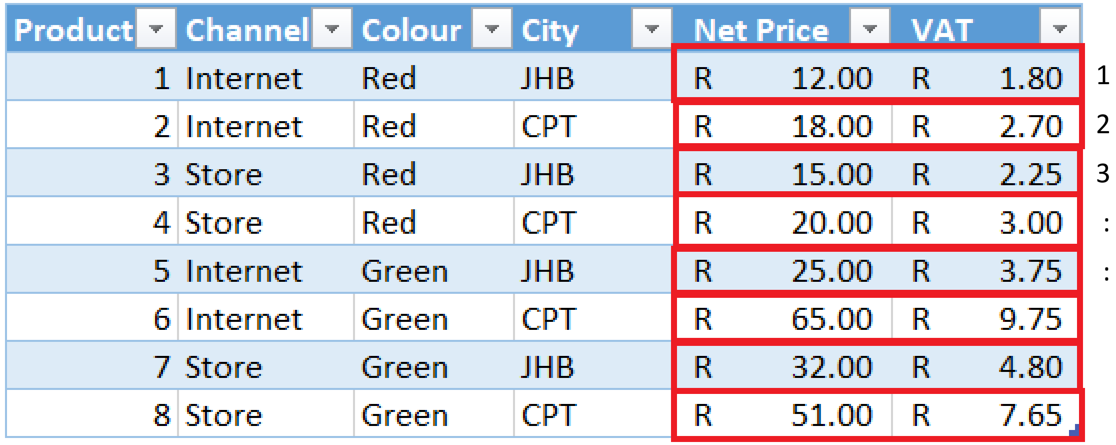
Iterators

- Iterators operate over the values in a column, row-by-row:
- All functions ending in an ‘X’ are iterators
    - Examples:
        - SUMX(Table, Expression)
        - AVERAGEX()
        - COUNTX()
        - COUNTAX()
        - MINX()
        - MAXX()
        - MEDIANX()

Iterators



```
Total Price = SUMX(Sales, Sales[ Net Price ] + Sales[ VAT ])
```



Product	Channel	Colour	City	Net Price	VAT
1	Internet	Red	JHB	R 12.00	R 1.80
2	Internet	Red	CPT	R 18.00	R 2.70
3	Store	Red	JHB	R 15.00	R 2.25
4	Store	Red	CPT	R 20.00	R 3.00
5	Internet	Green	JHB	R 25.00	R 3.75
6	Internet	Green	CPT	R 65.00	R 9.75
7	Store	Green	JHB	R 32.00	R 4.80
8	Store	Green	CPT	R 51.00	R 7.65

1. (12.00 + 1.80) +
2. (18.00 + 2.70) +
3. (15.00 + 2.25) +

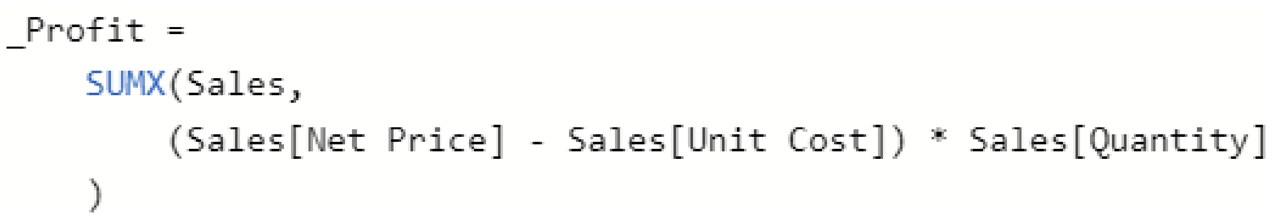
:

:

Result: R273.70

Calculating the Profit – Iterator Measure

- Create the following measure:

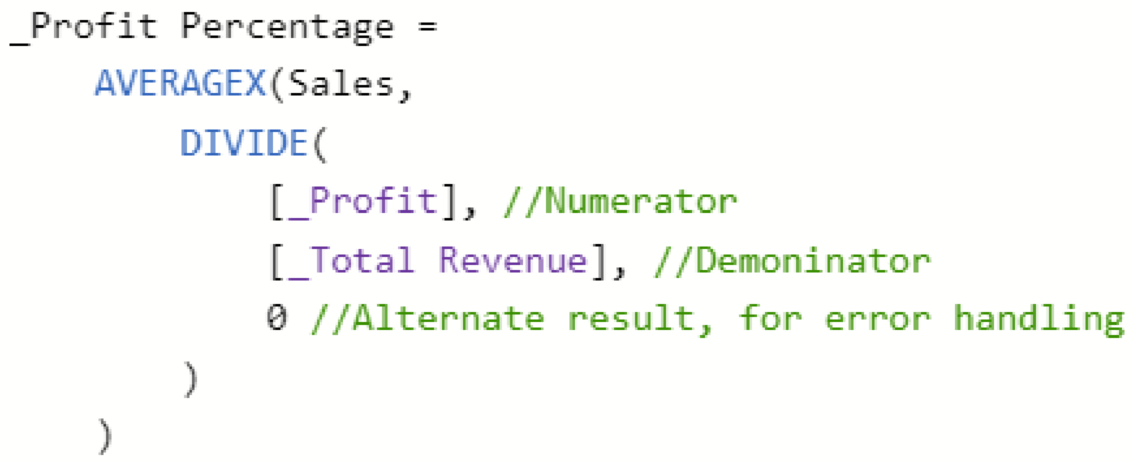


```
_Profit =
    SUMX(Sales,
        (Sales[Net Price] - Sales[Unit Cost]) * Sales[Quantity]
    )
```

- Using this measure, SUMX() is iterating over the Sales table.
- For each row it iterates over, it evaluates: `(Sales[Net Price] - Sales[Unit Cost]) * Sales[Quantity]`
- Once it is done computing the measure, only the result is stored in memory, not an entire column.
- This is far more efficient, especially with large datasets.

Calculating the Profit Percentage – Iterator Measure and DIVIDE()

- Create the following measure:



```
_Profit Percentage =
    AVERAGEX(Sales,
        DIVIDE(
            [_Profit], //Numerator
            [_Total Revenue], //Demoninator
            0 //Alternate result, for error handling
        )
    )
```

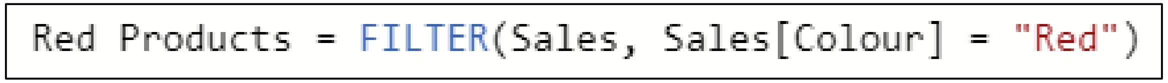
- AVERAGEX() iterates over the Sales table and computes the profit percentage.
- Use the DIVIDE() function. It is optimised and quicker than the ‘/’ operator.
- DIVIDE() has the option for an alternate result, in the case of a division by zero error.

Table Functions

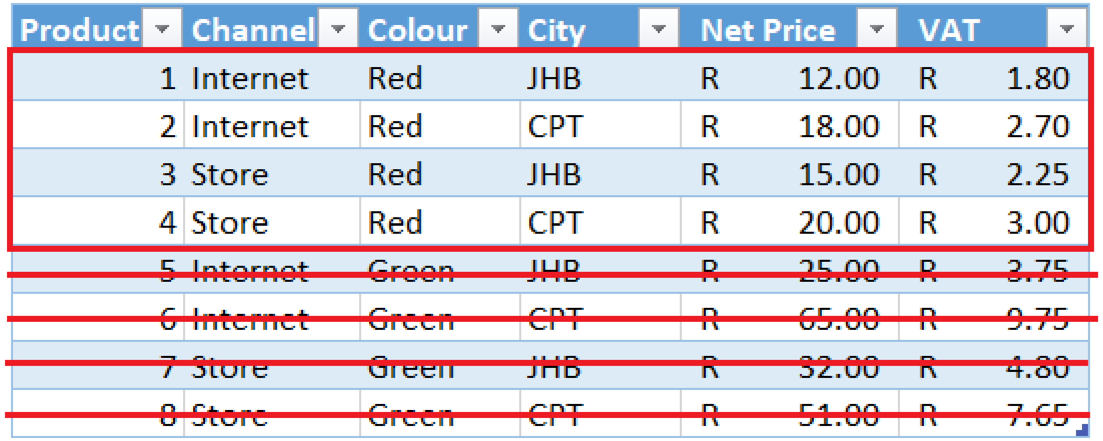
- Table functions return a table.
  - Examples:
    - FILTER()
    - ALL()

FILTER Function

- FILTER(Table, FilterExpression)
  - Iterator function.
  - It iterates over the target table, row-by-row. If the FilterExpression evaluates to TRUE for that row, that row is returned.
  - If the FilterExpression evaluates to FALSE, that row is not returned.



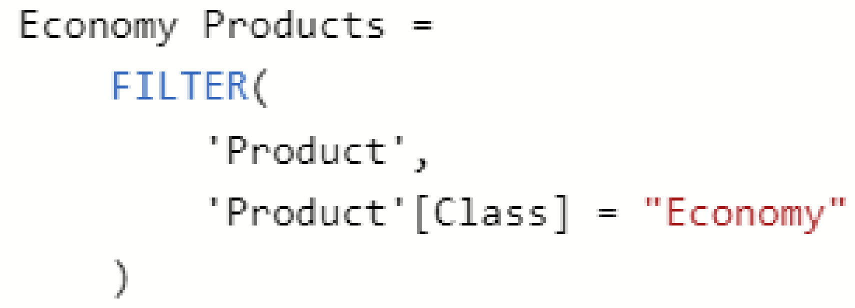
```
Red Products = FILTER(Sales, Sales[Colour] = "Red")
```



Product	Channel	Colour	City	Net Price	VAT
1	Internet	Red	JHB	R 12.00	R 1.80
2	Internet	Red	CPT	R 18.00	R 2.70
3	Store	Red	JHB	R 15.00	R 2.25
4	Store	Red	CPT	R 20.00	R 3.00
~~5~~	~~Internet~~	~~Green~~	~~JHB~~	~~R 25.00~~	~~R 3.75~~
~~6~~	~~Internet~~	~~Green~~	~~CPT~~	~~R 65.00~~	~~R 9.75~~
~~7~~	~~Store~~	~~Green~~	~~JHB~~	~~R 32.00~~	~~R 4.80~~
~~8~~	~~Store~~	~~Green~~	~~CPT~~	~~R 51.00~~	~~R 7.65~~

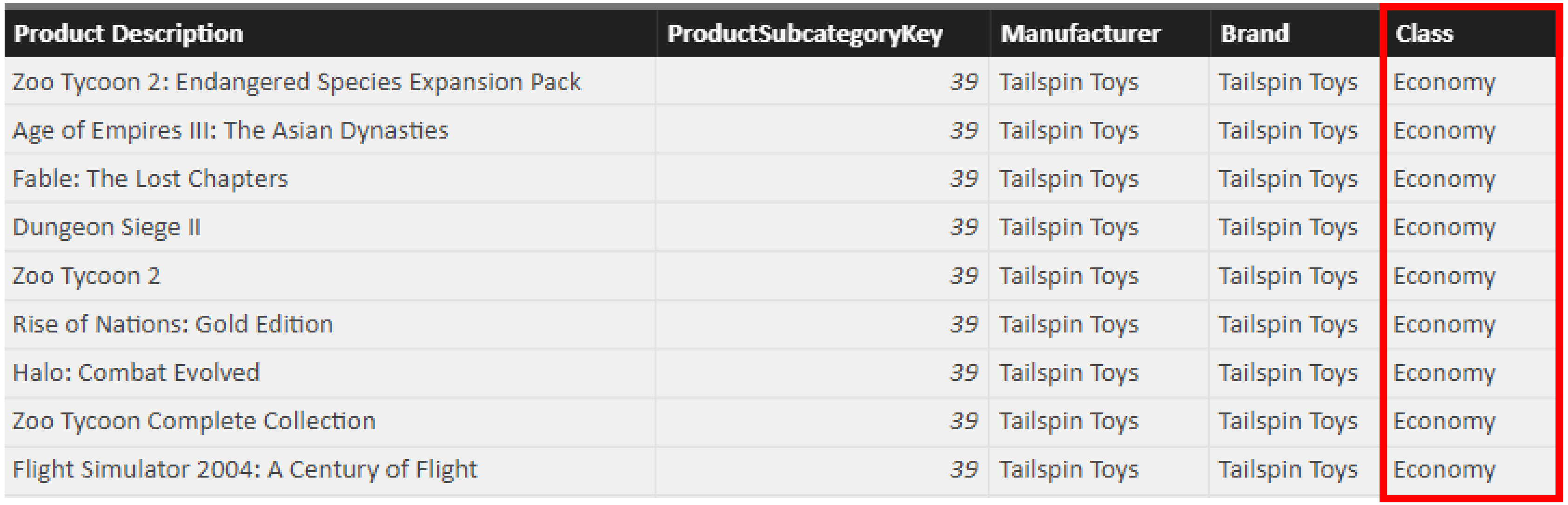
FILTER() Function – Example

- In the *Modelling* tab, select *New Table*
- Enter the following:



```
Economy Products =
    FILTER(
        'Product',
        'Product'[Class] = "Economy"
    )
```

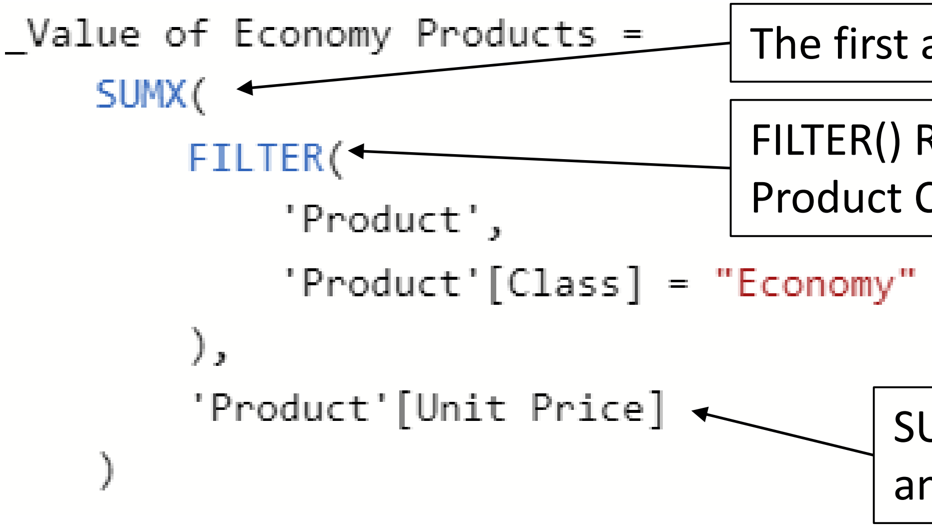
This returns a new table in the fields list which only contains Economy products.



Product Description	ProductSubcategoryKey	Manufacturer	Brand	Class
Zoo Tycoon 2: Endangered Species Expansion Pack	39	Tailspin Toys	Tailspin Toys	Economy
Age of Empires III: The Asian Dynasties	39	Tailspin Toys	Tailspin Toys	Economy
Fable: The Lost Chapters	39	Tailspin Toys	Tailspin Toys	Economy
Dungeon Siege II	39	Tailspin Toys	Tailspin Toys	Economy
Zoo Tycoon 2	39	Tailspin Toys	Tailspin Toys	Economy
Rise of Nations: Gold Edition	39	Tailspin Toys	Tailspin Toys	Economy
Halo: Combat Evolved	39	Tailspin Toys	Tailspin Toys	Economy
Zoo Tycoon Complete Collection	39	Tailspin Toys	Tailspin Toys	Economy
Flight Simulator 2004: A Century of Flight	39	Tailspin Toys	Tailspin Toys	Economy

Calculating the value of products in stock

- FILTER() is useful when using the result as an input of an iterator function, such as SUMX().
- Create the following measure in the Product table:



```
_Value of Economy Products =
    SUMX(
        FILTER(
            'Product',
            'Product'[Class] = "Economy"
        ),
        'Product'[Unit Price]
    )
```

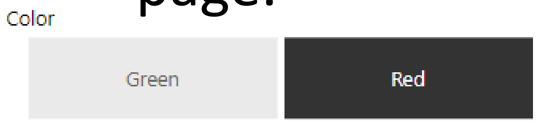
The first argument of SUMX() must be a Table

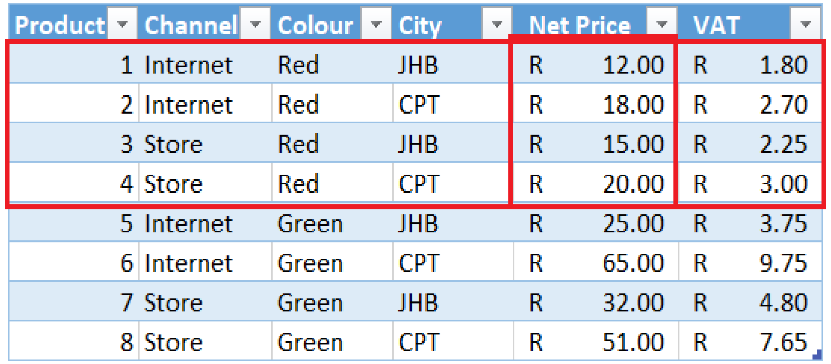
FILTER() Returns a table where the Product Class = Economy

SUMX() iterates over the table returned by FILTER(), and calculates the sum of Unit Price.

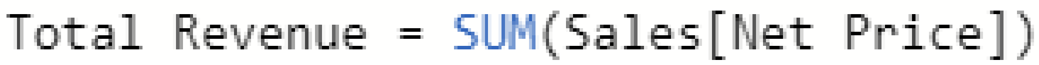
ALL() Function

- ALL(TableNameOrColumnName)
  - When operating on a table, it ignores any filters applied by any slicers on that page.


Color

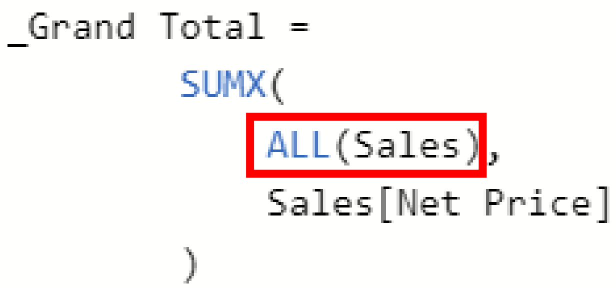


Product	Channel	Colour	City	Net Price	VAT
1	Internet	Red	JHB	R 12.00	R 1.80
2	Internet	Red	CPT	R 18.00	R 2.70
3	Store	Red	JHB	R 15.00	R 2.25
4	Store	Red	CPT	R 20.00	R 3.00
5	Internet	Green	JHB	R 25.00	R 3.75
6	Internet	Green	CPT	R 65.00	R 9.75
7	Store	Green	JHB	R 32.00	R 4.80
8	Store	Green	CPT	R 51.00	R 7.65

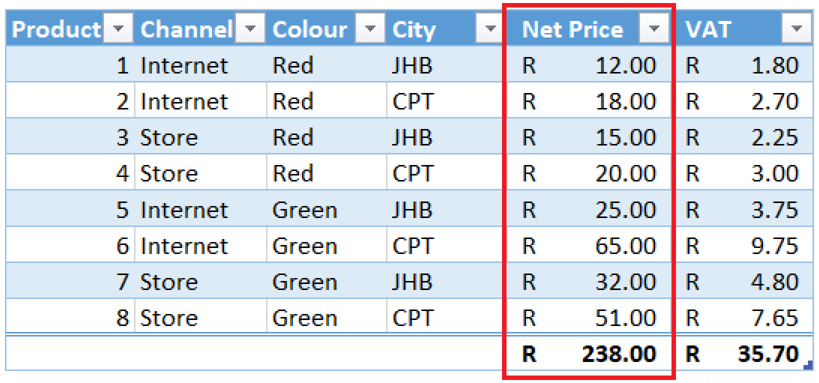


```
Total Revenue = SUM(Sales[Net Price])
```

- Page slicer set to “Red”
- Result: R65.00



```
_Grand Total =
        SUMX(
            ALL(Sales),
            Sales[Net Price]
        )
```

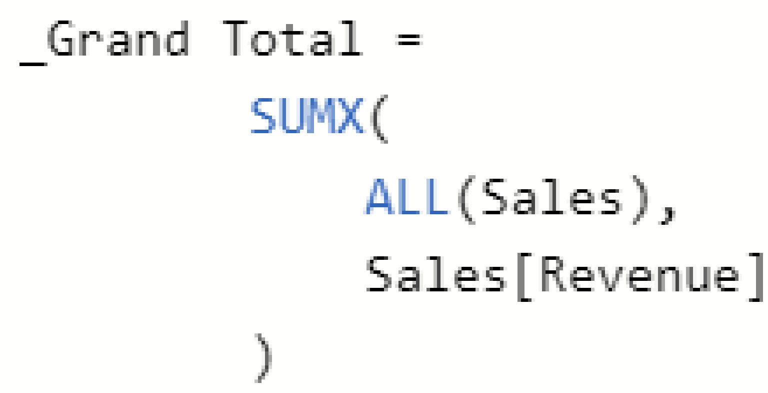


Product	Channel	Colour	City	Net Price	VAT
1	Internet	Red	JHB	R 12.00	R 1.80
2	Internet	Red	CPT	R 18.00	R 2.70
3	Store	Red	JHB	R 15.00	R 2.25
4	Store	Red	CPT	R 20.00	R 3.00
5	Internet	Green	JHB	R 25.00	R 3.75
6	Internet	Green	CPT	R 65.00	R 9.75
7	Store	Green	JHB	R 32.00	R 4.80
8	Store	Green	CPT	R 51.00	R 7.65
				R 238.00	**R 35.70**

- Result: R238.00

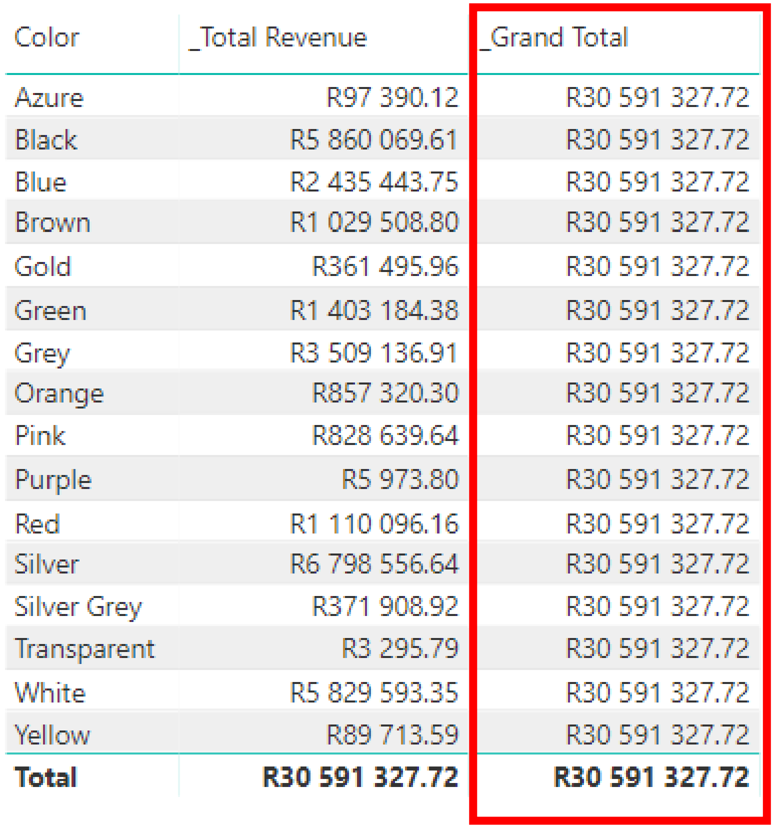
ALL() Function

- Create the following measure in the Sales table:



```
_Grand Total =
        SUMX(
            ALL(Sales),
            Sales[Revenue]
        )
```

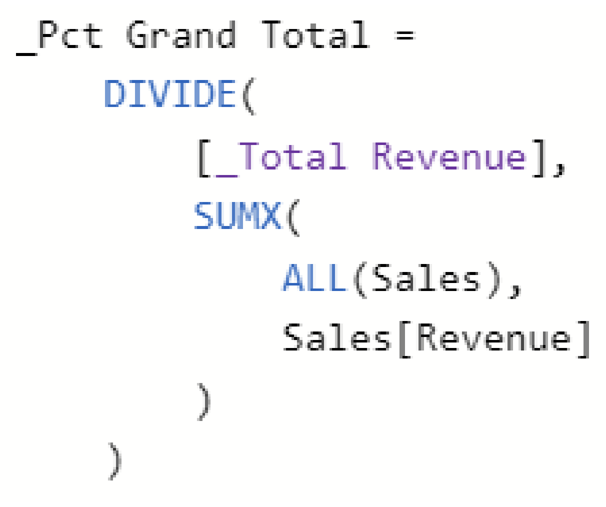
- Place this measure in a Table visual, along with Product Color and Total Revenue.
- This Grand Total measure is useful if it is used as the denominator in a % of Grand Total Calculation



Color	_Total Revenue	_Grand Total
Azure	R97 390.12	R30 591 327.72
Black	R5 860 069.61	R30 591 327.72
Blue	R2 435 443.75	R30 591 327.72
Brown	R1 029 508.80	R30 591 327.72
Gold	R361 495.96	R30 591 327.72
Green	R1 403 184.38	R30 591 327.72
Grey	R3 509 136.91	R30 591 327.72
Orange	R857 320.30	R30 591 327.72
Pink	R828 639.64	R30 591 327.72
Purple	R5 973.80	R30 591 327.72
Red	R1 110 096.16	R30 591 327.72
Silver	R6 798 556.64	R30 591 327.72
Silver Grey	R371 908.92	R30 591 327.72
Transparent	R3 295.79	R30 591 327.72
White	R5 829 593.35	R30 591 327.72
Yellow	R89 713.59	R30 591 327.72
Total	**R30 591 327.72**	**R30 591 327.72**

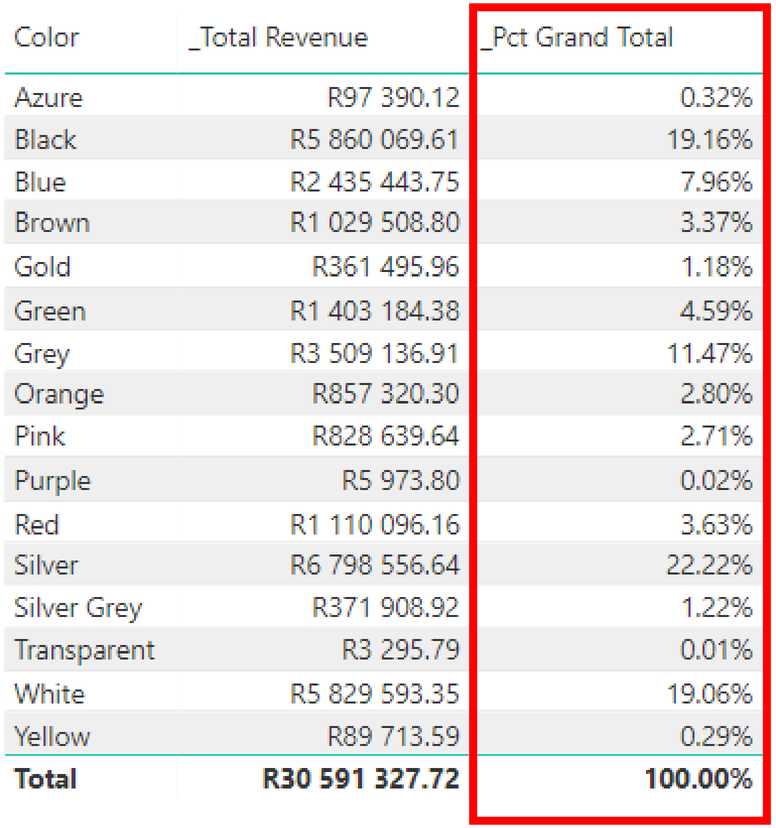
ALL() Function

- Create the following measure:



```
_Pct Grand Total =
    DIVIDE(
        [_Total Revenue],
        SUMX(
            ALL(Sales),
            Sales[Revenue]
        )
    )
```

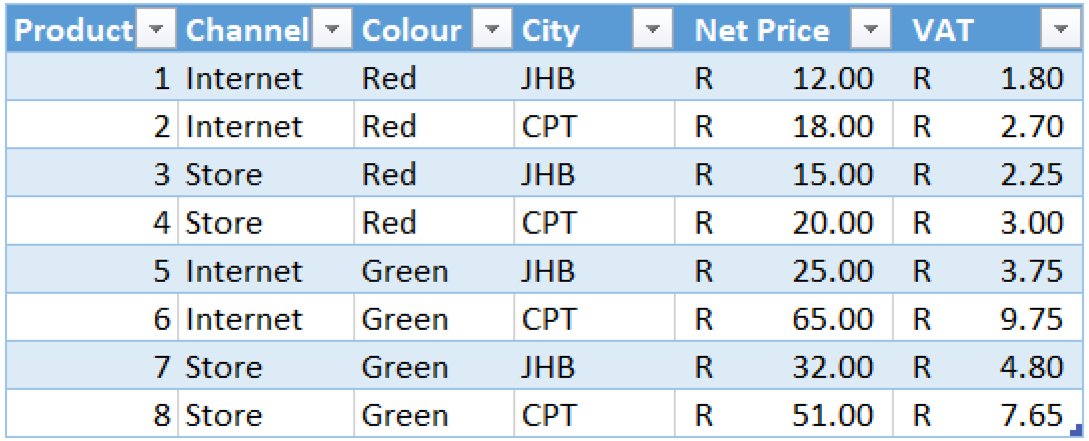
- Remove the Grand Total measure from the table, and replace it with the Pct Grand Total measure.



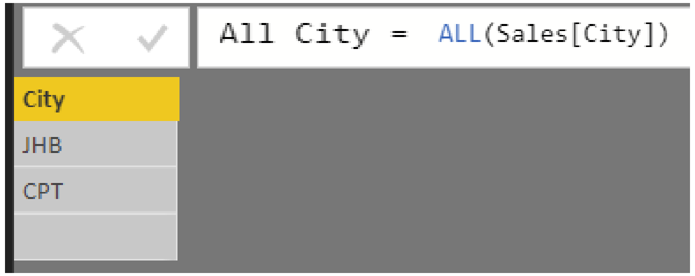
Color	_Total Revenue	_Pct Grand Total
Azure	R97 390.12	0.32%
Black	R5 860 069.61	19.16%
Blue	R2 435 443.75	7.96%
Brown	R1 029 508.80	3.37%
Gold	R361 495.96	1.18%
Green	R1 403 184.38	4.59%
Grey	R3 509 136.91	11.47%
Orange	R857 320.30	2.80%
Pink	R828 639.64	2.71%
Purple	R5 973.80	0.02%
Red	R1 110 096.16	3.63%
Silver	R6 798 556.64	22.22%
Silver Grey	R371 908.92	1.22%
Transparent	R3 295.79	0.01%
White	R5 829 593.35	19.06%
Yellow	R89 713.59	0.29%
Total	**R30 591 327.72**	**100.00%**

ALL() Function used on columns

- ALL(TableNameOrColumnName)
  - When operating on a single column, it returns a table containing one column of the unique values within that column.
  - When operating on multiple columns, it returns all the existing combinations of values of those multiple columns (NOTE: It does NOT return all the possible combinations, only those that exist).

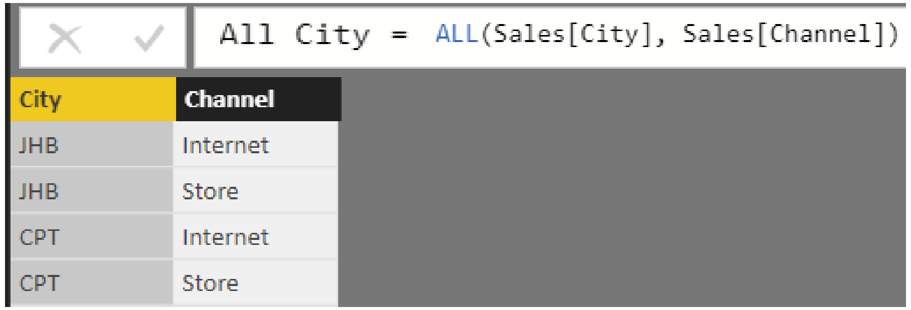


Product	Channel	Colour	City	Net Price	VAT
1	Internet	Red	JHB	R 12.00	R 1.80
2	Internet	Red	CPT	R 18.00	R 2.70
3	Store	Red	JHB	R 15.00	R 2.25
4	Store	Red	CPT	R 20.00	R 3.00
5	Internet	Green	JHB	R 25.00	R 3.75
6	Internet	Green	CPT	R 65.00	R 9.75
7	Store	Green	JHB	R 32.00	R 4.80
8	Store	Green	CPT	R 51.00	R 7.65



```
All City = ALL(Sales[City])
```

City
JHB
CPT

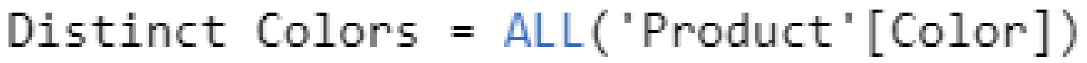


```
All City = ALL(Sales[City], Sales[Channel])
```

City	Channel
JHB	Internet
JHB	Store
CPT	Internet
CPT	Store

ALL() Columns – Example

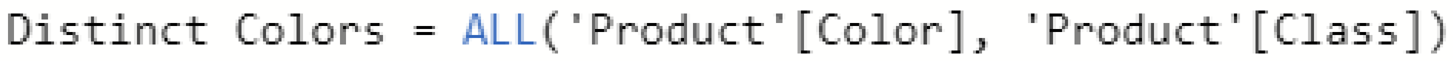
- Create a new table using the following expression:



```
Distinct Colors = ALL('Product'[Color])
```

  - Note the result. This is a list of all the distinct values in the Color column.

- Now change the expression to:



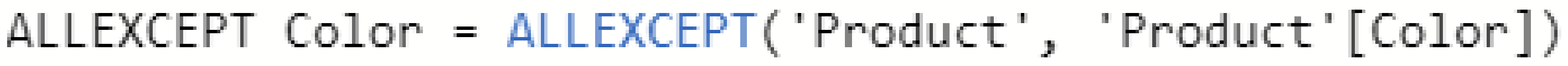
```
Distinct Colors = ALL('Product'[Color], 'Product'[Class])
```

  - This shows all of the existing combinations between Color and Class.

ALLEXCEPT()

- If you need to use all but one of the columns in a table, you can use ALLEXCEPT(Table, Table[Column])

- Create a new table using the following expression:



```
ALLEXCEPT Color = ALLEXCEPT('Product', 'Product'[Color])
```

- This returns a table with all the columns from the Product table, apart from the Color column.

Using relationships

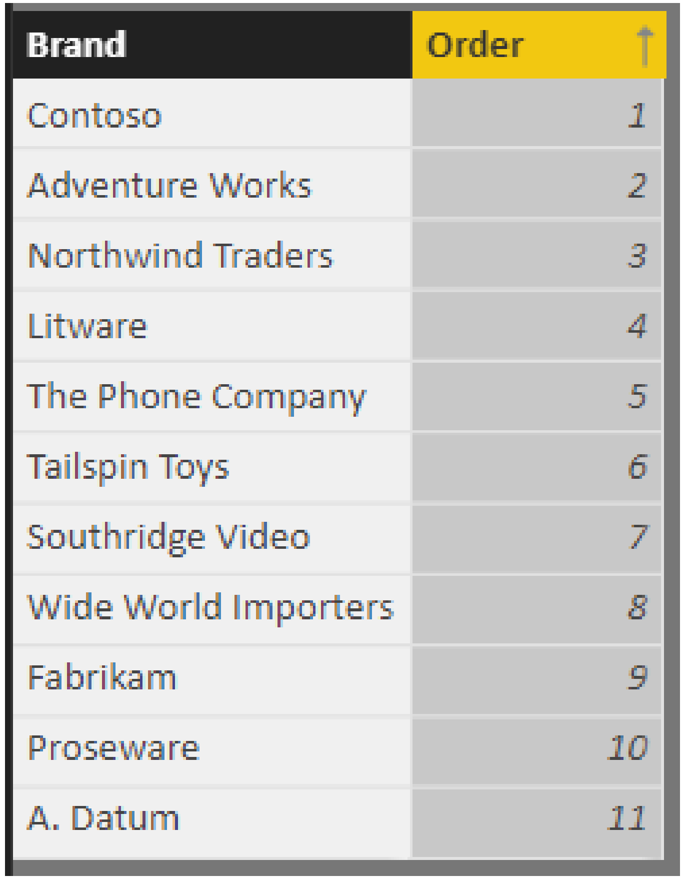
- RELATED() and RELATEDTABLE()
  - These functions allow you to reference a column in another table that has a relationship with the current table.

- RELATED()
  - Works from the Many side to the One side of the relationship

- RELATEDTABLE()
  - Works from the One side to the Many side of the relationship.

RELATED() Example – Ordering Product Brand

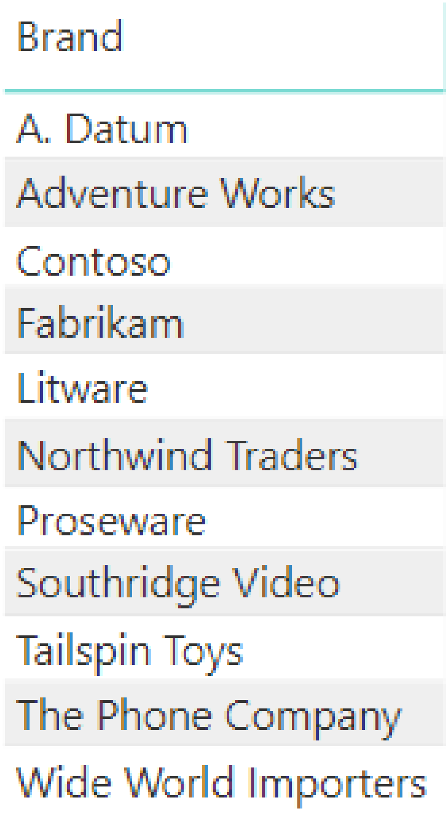
- Whenever the Product Brands are listed, the Executive Committee would like to see them in the following order:



Brand	Order ↑
Contoso	1
Adventure Works	2
Northwind Traders	3
Litware	4
The Phone Company	5
Tailspin Toys	6
Southridge Video	7
Wide World Importers	8
Fabrikam	9
Proseware	10
A. Datum	11

RELATED() Example – Ordering Product Brand

- Create a new table visual on the report and populate it with the Product Brand.
- At the moment, there is no way of ordering the Brand column to achieve what the Committee wants.

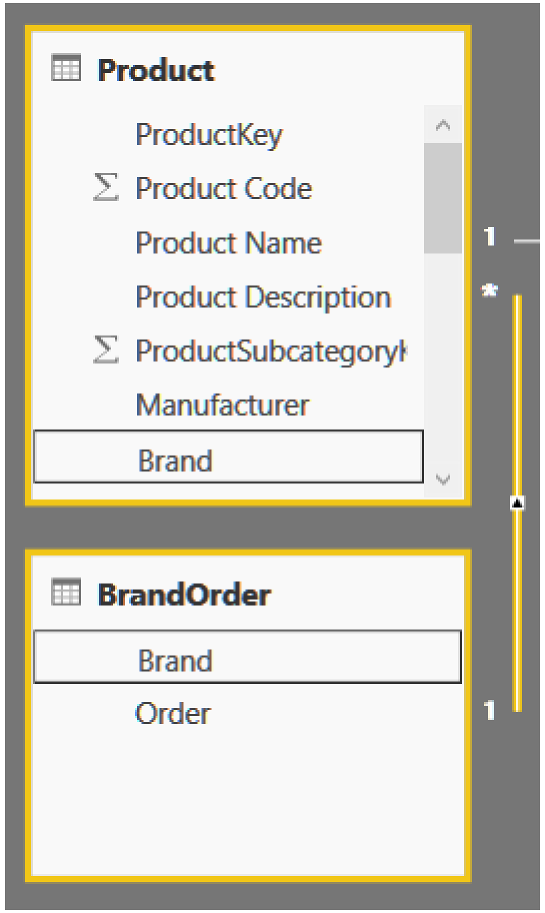


Brand
A. Datum
Adventure Works
Contoso
Fabrikam
Litware
Northwind Traders
Proseware
Southridge Video
Tailspin Toys
The Phone Company
Wide World Importers

RELATED() Example – Ordering Product Brand

- Get Data -> Text/CSV -> BrandOrder.csv.
- The BrandOrder.csv contains the correct ordering for the Brands column.
- Create a relationship:
  - Product[Brand] -> BrandOrder[Brand]
- In the Product table, create a new column with the following expression:

```
BrandOrder = RELATED(BrandOrder[Order])
```



Product

ProductKey
Σ Product Code
Product Name
Product Description
Σ ProductSubcategoryI
Manufacturer
Brand

BrandOrder

Brand
Order

RELATED() Example – Ordering Product Brand

- In the Product table, a new column, BrandOrder, is visible.

- In the Product table, select Brand, then click on:
  - *Modeling -> Sort by Column -> BrandOrder*

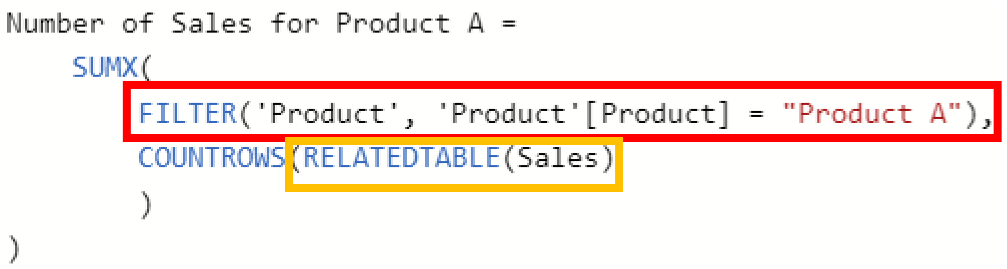
- The Brand column is sorted in the order requested by the Committee.

Using relationships

- RELATEDTABLE(Column)
  - This is a Table function. It will return a table.
  - This is commonly used as an input argument for another expression, e.g., FILTER() or SUMX().
- This works from the one side to the many side of a relationship.
- For a single product in the Product table, you can calculate the number of sales that product made.

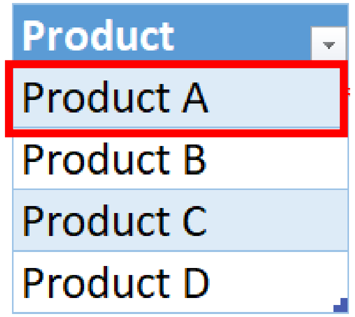
RELATEDTABLE()

- An example of RELATEDTABLE()



```
Number of Sales for Product A =
    SUMX(
        FILTER('Product', 'Product'[Product] = "Product A"),
        COUNTROWS(RELATEDTABLE(Sales)
        )
)
```

Product Table

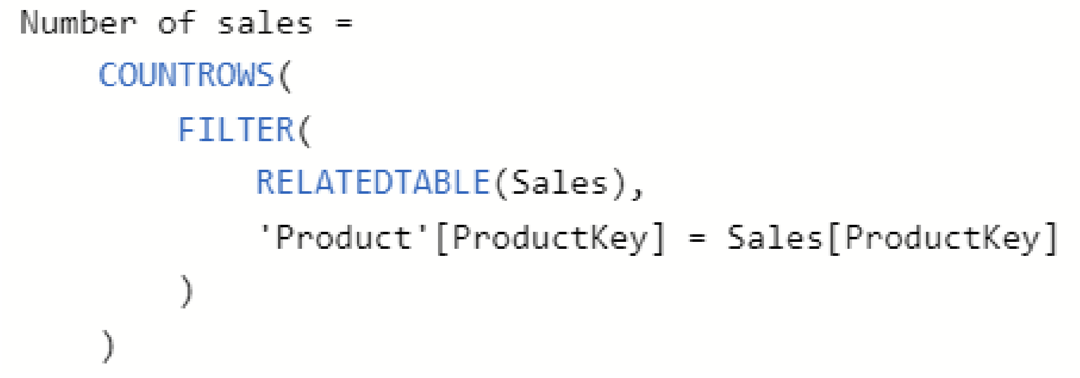


Product
Product A
Product B
Product C
Product D

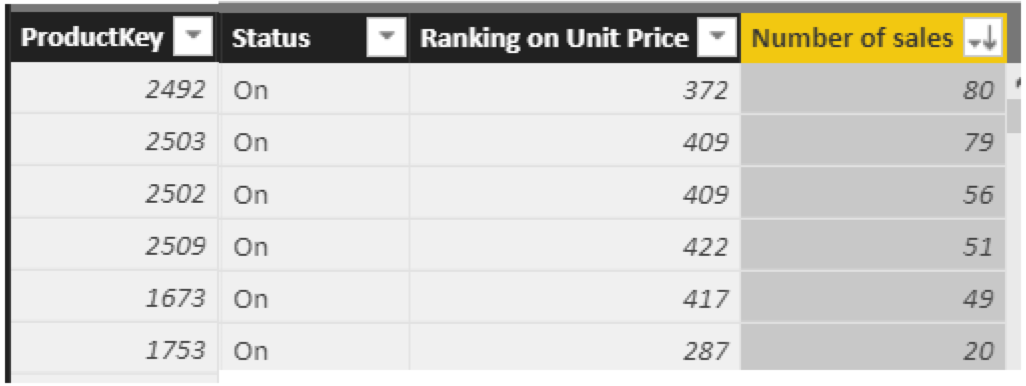
- FILTER() returns only the record where Product = *Product A*
- RELATEDTABLE() then iterates over the Sales table and returns sales for *Product A*.
- COUNTROWS() counts the number of rows returned by RELATEDTABLE().
- This gives the number of sales for *Product A.*

RELATEDTABLE()

- Create the following calculated column in the Product table:



```
Number of sales =
    COUNTROWS(
        FILTER(
            RELATEDTABLE(Sales),
            'Product'[ProductKey] = Sales[ProductKey]
        )
    )
```



ProductKey	Status	Ranking on Unit Price	Number of sales
2492	On	372	80
2503	On	409	79
2502	On	409	56
2509	On	422	51
1673	On	417	49
1753	On	287	20

- Note that ProductKey 2492 had 80 sales. Verify this in the Sales table.

Session 2

- Evaluation contexts
- CALCULATE()
- Time intelligence.

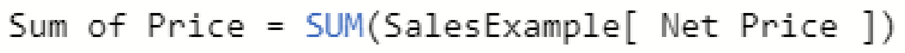
Evaluation Contexts

- DAX computes measures based on a certain *evaluation context.*

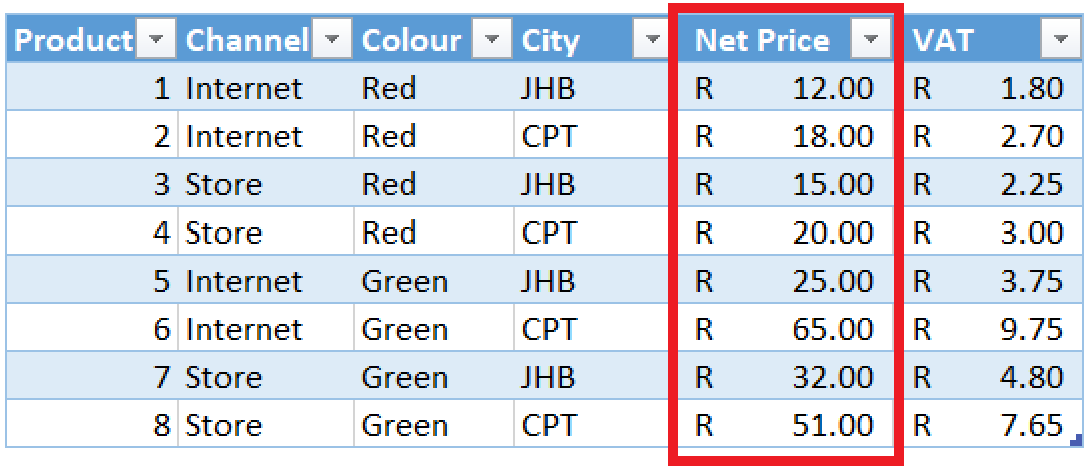
- *Filter context*
  - Filter context is set by the visual you are using, or by the slicers on the page.
  - The filter context supersedes the row context.
- R*ow context*.
  - This is set by the *current row*; either in a calculated column, or the current row being evaluated in an iterator.
  - Only exists for the time when an iterator is running.
  - Calculated columns have row contexts by default. Measures do not.

- Understanding these is fundamental to your continued progression with DAX.

Filter Context

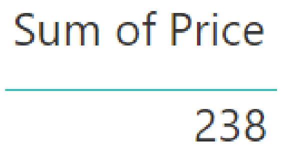


```
Sum of Price = SUM(SalesExample[ Net Price ])
```



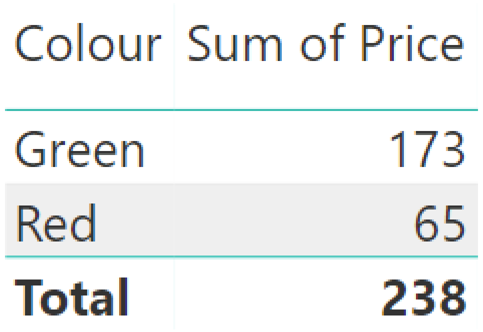
Product	Channel	Colour	City	Net Price	VAT
1	Internet	Red	JHB	R 12.00	R 1.80
2	Internet	Red	CPT	R 18.00	R 2.70
3	Store	Red	JHB	R 15.00	R 2.25
4	Store	Red	CPT	R 20.00	R 3.00
5	Internet	Green	JHB	R 25.00	R 3.75
6	Internet	Green	CPT	R 65.00	R 9.75
7	Store	Green	JHB	R 32.00	R 4.80
8	Store	Green	CPT	R 51.00	R 7.65

Table with no sub categories included

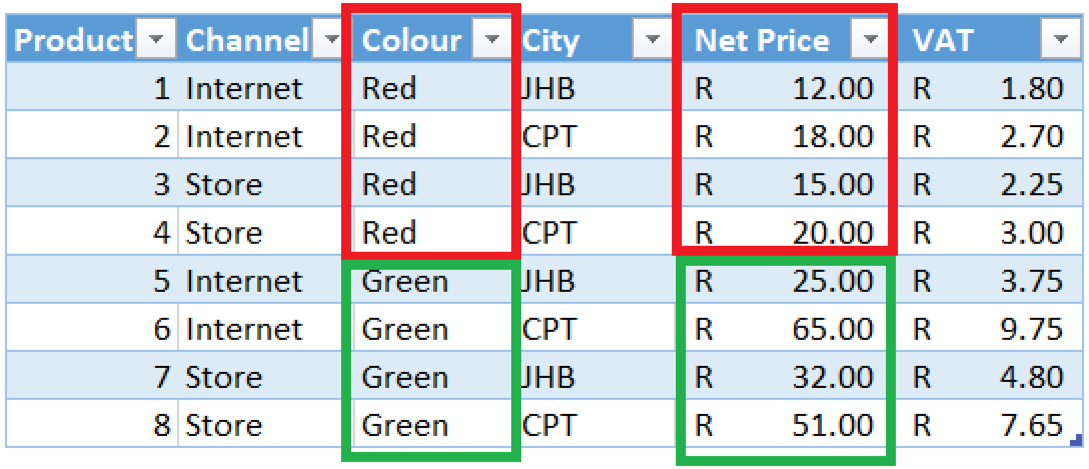


Sum of Price
238

Table with colour sub categories included



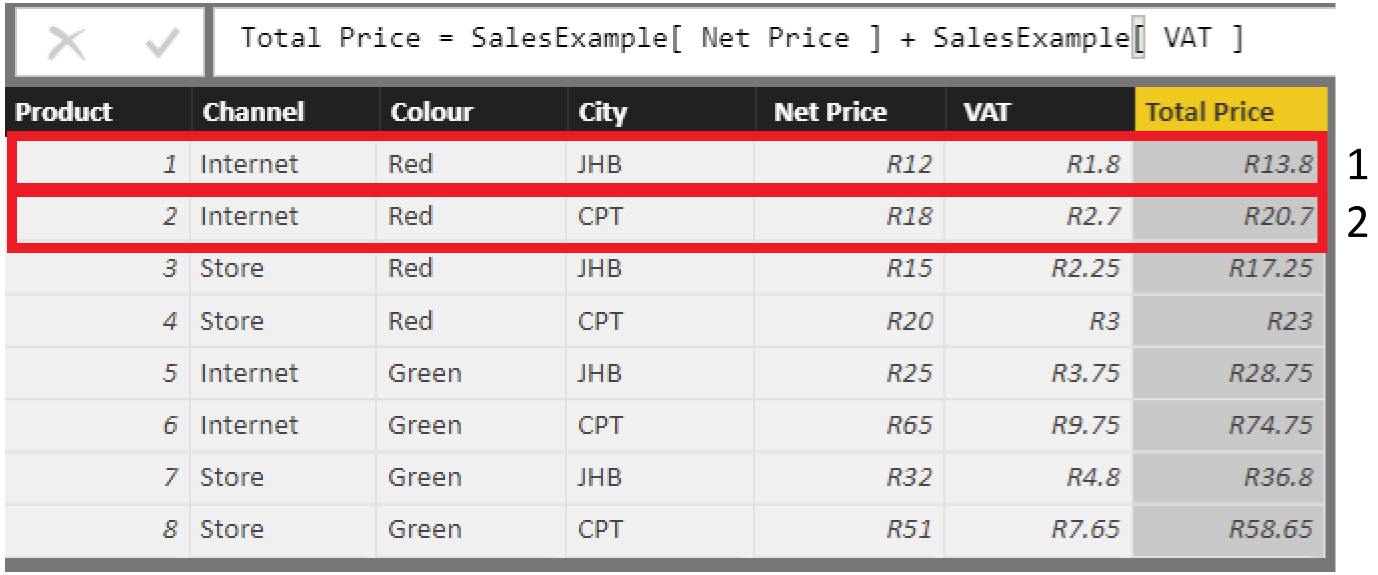
Colour	Sum of Price
Green	173
Red	65
Total	**238**



Product	Channel	Colour	City	Net Price	VAT
1	Internet	Red	JHB	R 12.00	R 1.80
2	Internet	Red	CPT	R 18.00	R 2.70
3	Store	Red	JHB	R 15.00	R 2.25
4	Store	Red	CPT	R 20.00	R 3.00
5	Internet	Green	JHB	R 25.00	R 3.75
6	Internet	Green	CPT	R 65.00	R 9.75
7	Store	Green	JHB	R 32.00	R 4.80
8	Store	Green	CPT	R 51.00	R 7.65

Row Context in Calculated Columns

- In a calculated column, the DAX engine scans the table row-by-row.
- It then computes the measure for the values available in that particular row.

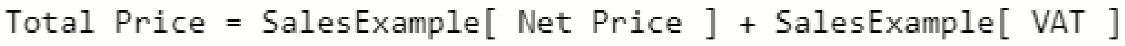


```
Total Price = SalesExample[ Net Price ] + SalesExample[ VAT ]
```

Product	Channel	Colour	City	Net Price	VAT	Total Price
1	Internet	Red	JHB	R12	R1.8	R13.8
2	Internet	Red	CPT	R18	R2.7	R20.7
3	Store	Red	JHB	R15	R2.25	R17.25
4	Store	Red	CPT	R20	R3	R23
5	Internet	Green	JHB	R25	R3.75	R28.75
6	Internet	Green	CPT	R65	R9.75	R74.75
7	Store	Green	JHB	R32	R4.8	R36.8
8	Store	Green	CPT	R51	R7.65	R58.65

Row Context

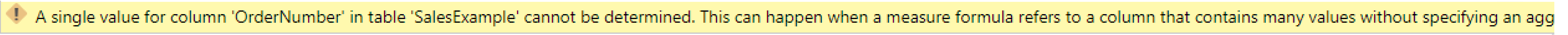
- Consider the following calculated column:



```
Total Price = SalesExample[ Net Price ] + SalesExample[ VAT ]
```

- This DAX expression works because there is a row context defined by the *current row* in the table you are iterating over.

- If you create a measure with this expression, you will get the following error:



A single value for column 'OrderNumber' in table 'SalesExample' cannot be determined. This can happen when a measure formula refers to a column that contains many values without specifying an agg

- This is basically saying that the measure does not have access to a row context.

- If you want this expression to be evaluated in a measure, you need to use it inside an iterator function, like SUMX().

- SUMX() provides a row context in which the measure can be evaluated.

Row Context with Iterators

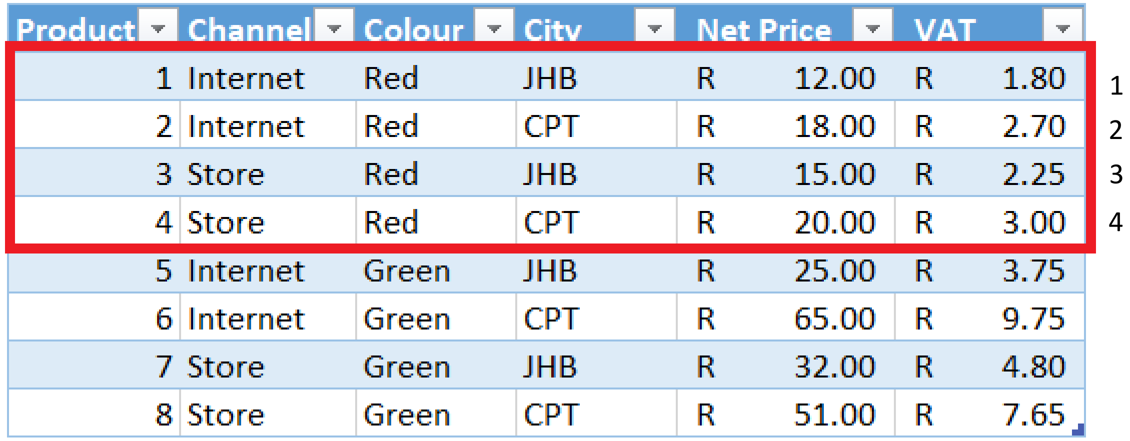
```
Total Price = SUMX(Sales, Sales[ Net Price ] + Sales[ VAT ])
```

- With a colour slicer set to “Red”.
- The table is filtered, such that the measure only sees the rows where the colour is red.
- The iterator then iterates over the rows visible to it in the current filter context.
- The iterator generates a row context for each row it iterates over in the current filtered view of the data.

1. (12.00 + 1.80) +
2. (18.00 + 2.70) +
3. (15.00 + 2.25) +
4. (20.00 + 3.00)

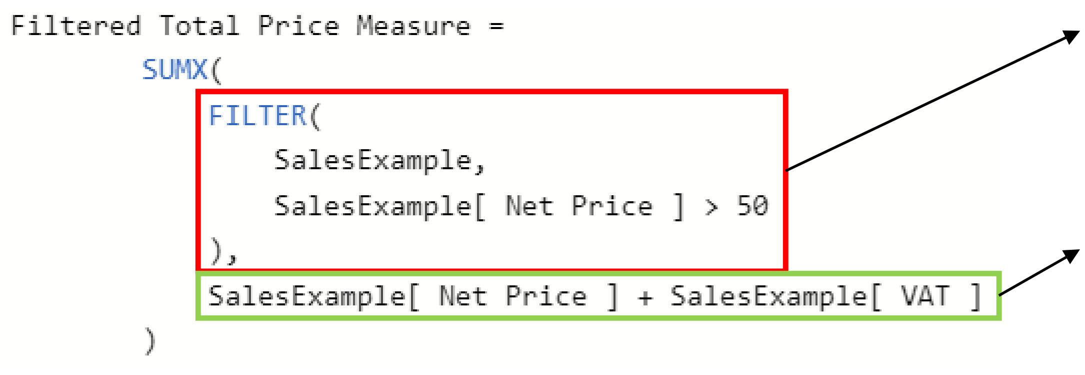
:

Result: R74.75



Product	Channel	Colour	City	Net Price	VAT
1	Internet	Red	JHB	R 12.00	R 1.80
2	Internet	Red	CPT	R 18.00	R 2.70
3	Store	Red	JHB	R 15.00	R 2.25
4	Store	Red	CPT	R 20.00	R 3.00
5	Internet	Green	JHB	R 25.00	R 3.75
6	Internet	Green	CPT	R 65.00	R 9.75
7	Store	Green	JHB	R 32.00	R 4.80
8	Store	Green	CPT	R 51.00	R 7.65

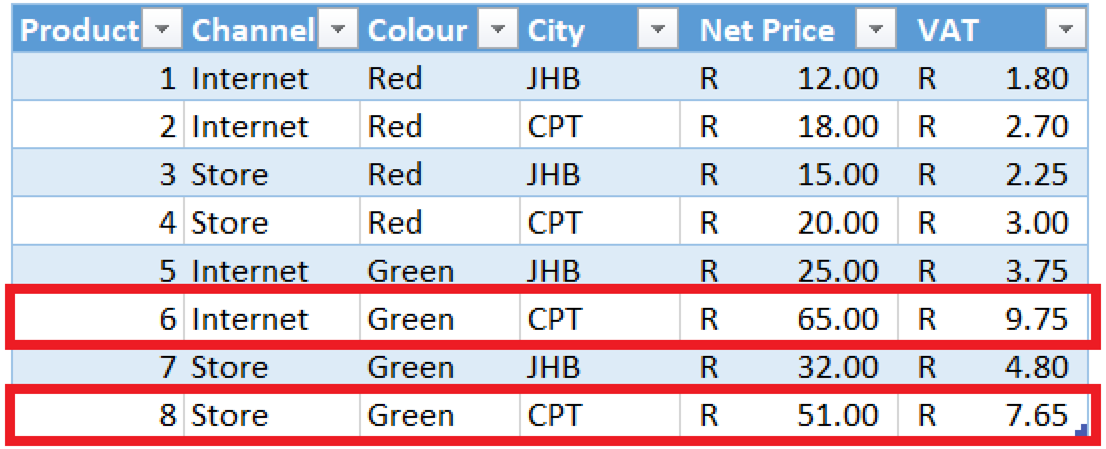
Row Context in Iterators



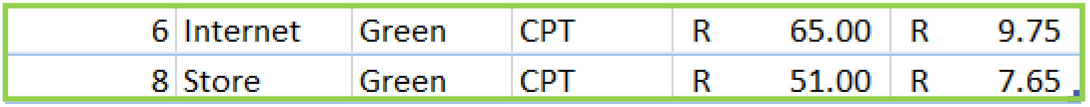
```
Filtered Total Price Measure =
        SUMX(
            FILTER(
                SalesExample,
                SalesExample[ Net Price ] > 50
            ),
            SalesExample[ Net Price ] + SalesExample[ VAT ]
        )
```

FILTER() applies a row context over the table. It iterates over the table to find the rows were the filter condition is met.

FILTER() then returns the rows to SUMX() which iterates over these rows and evaluates the expression



Product	Channel	Colour	City	Net Price	VAT
1	Internet	Red	JHB	R 12.00	R 1.80
2	Internet	Red	CPT	R 18.00	R 2.70
3	Store	Red	JHB	R 15.00	R 2.25
4	Store	Red	CPT	R 20.00	R 3.00
5	Internet	Green	JHB	R 25.00	R 3.75
6	Internet	Green	CPT	R 65.00	R 9.75
7	Store	Green	JHB	R 32.00	R 4.80
8	Store	Green	CPT	R 51.00	R 7.65



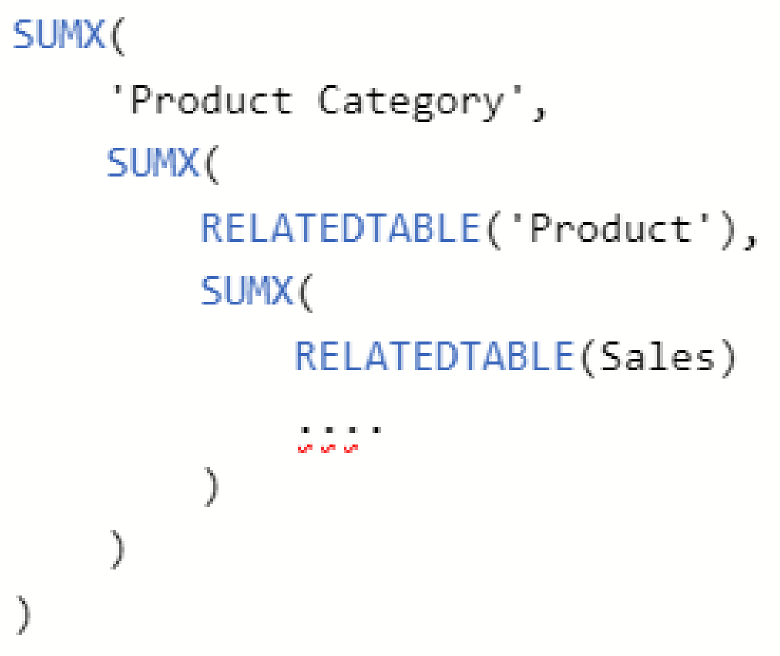
6	Internet	Green	CPT	R 65.00	R 9.75
8	Store	Green	CPT	R 51.00	R 7.65

Output: R133.40

Nested Row Contexts

- Row contexts can be nested

- Each row context can use some value of the outer, or *earlier* row context.



```
SUMX(
    'Product Category',
    SUMX(
        RELATEDTABLE('Product'),
        SUMX(
            RELATEDTABLE(Sales)
            ....
        )
    )
)
```

This measure is computing the:

Category

Products of category

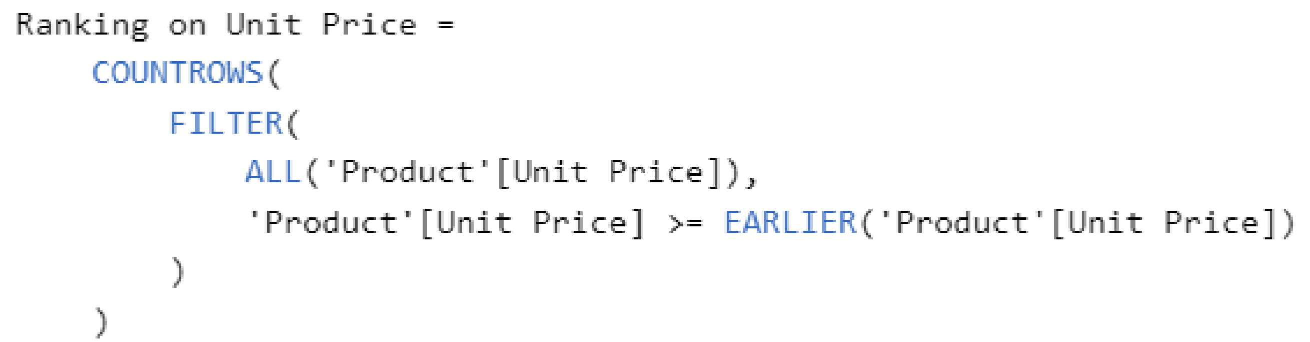
Sales of product

Calculate Ranking of Product Price

- To calculate the ranking of the prices of products, we need to use a nested row context.
- This is useful to know, as our most expensive products don’t necessarily generate the most revenue.
- To get the ranking of price we undertake the following steps:
    - We get a distinct list of product prices.
    - We compare each product’s price to every other product’s price; we count the number of products that are more expensive than the current one.
    - If there are no products that are more expensive than the current one, the current on is the most expensive.
    - If there is only one product more expensive than the current one, then the current one is the second most expensive, and so on.

Calculate Ranking of Product Price

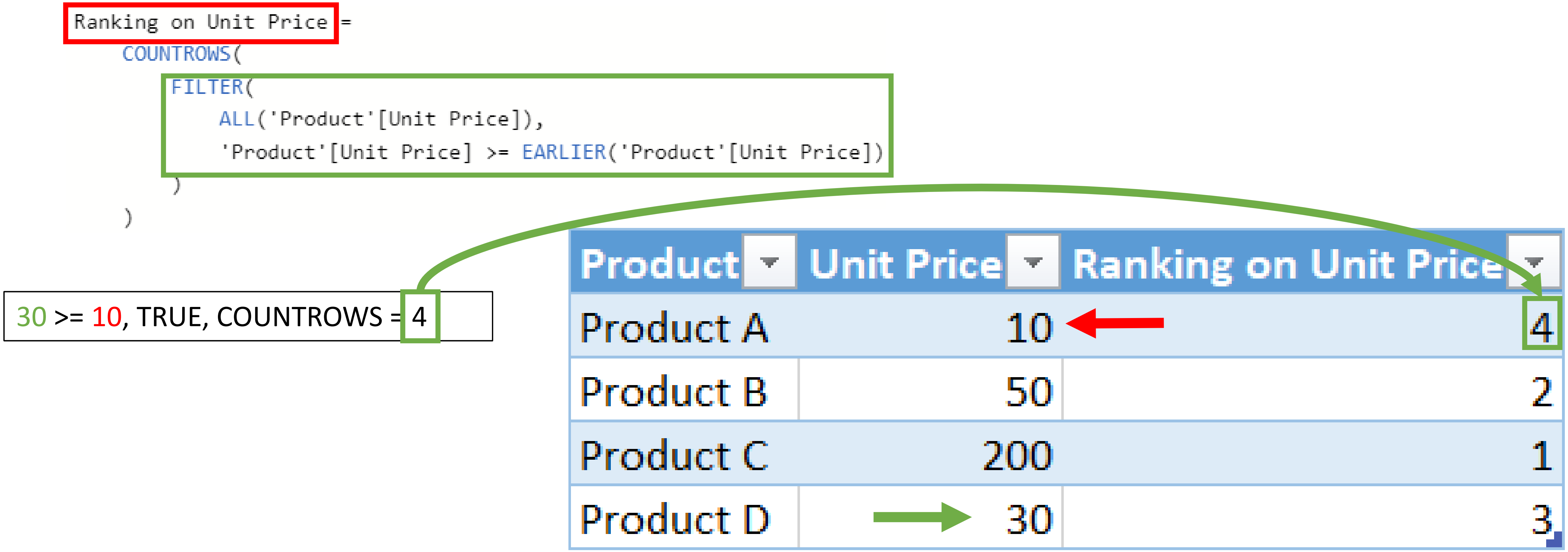
- To calculate the ranking of cost price, create the following calculated column in the Product table:



```
Ranking on Unit Price =
    COUNTROWS(
        FILTER(
            ALL('Product'[Unit Price]),
            'Product'[Unit Price] >= EARLIER('Product'[Unit Price])
        )
    )
```

- There are two row contexts here:
  - First: Created by the calculated column.
  - Second: Created by the iterator FILTER().
- For every row iterated by the calculated column, FILTER() iterates over the entire table.
- This is similar to a nested for loop.

How does the expression work?



```
Ranking on Unit Price =
    COUNTROWS(
        FILTER(
            ALL('Product'[Unit Price]),
            'Product'[Unit Price] >= EARLIER('Product'[Unit Price])
        )
    )
```

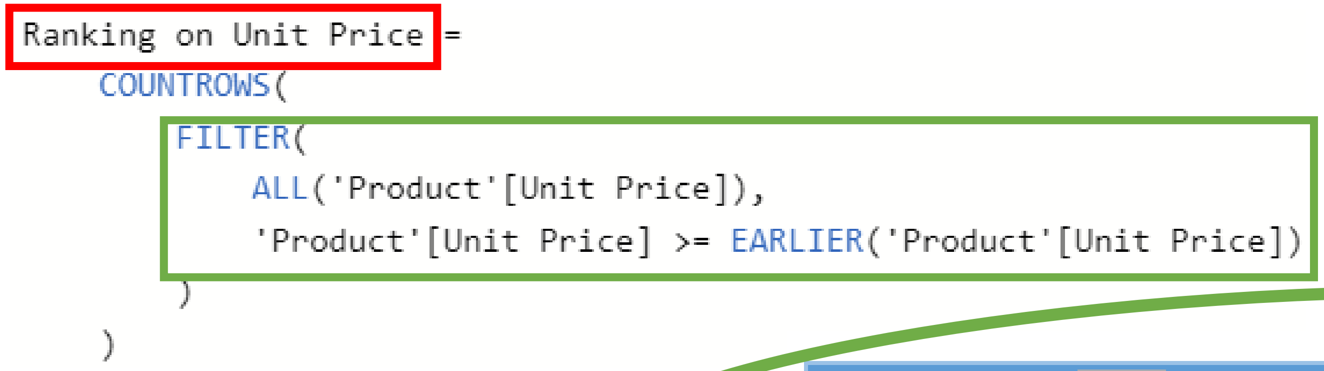
30 >= 10, TRUE, COUNTROWS = 4

Product	Unit Price	Ranking on Unit Price
Product A	10	4
Product B	50	2
Product C	200	1
Product D	30	3

RED: Row context generated by the calculated column

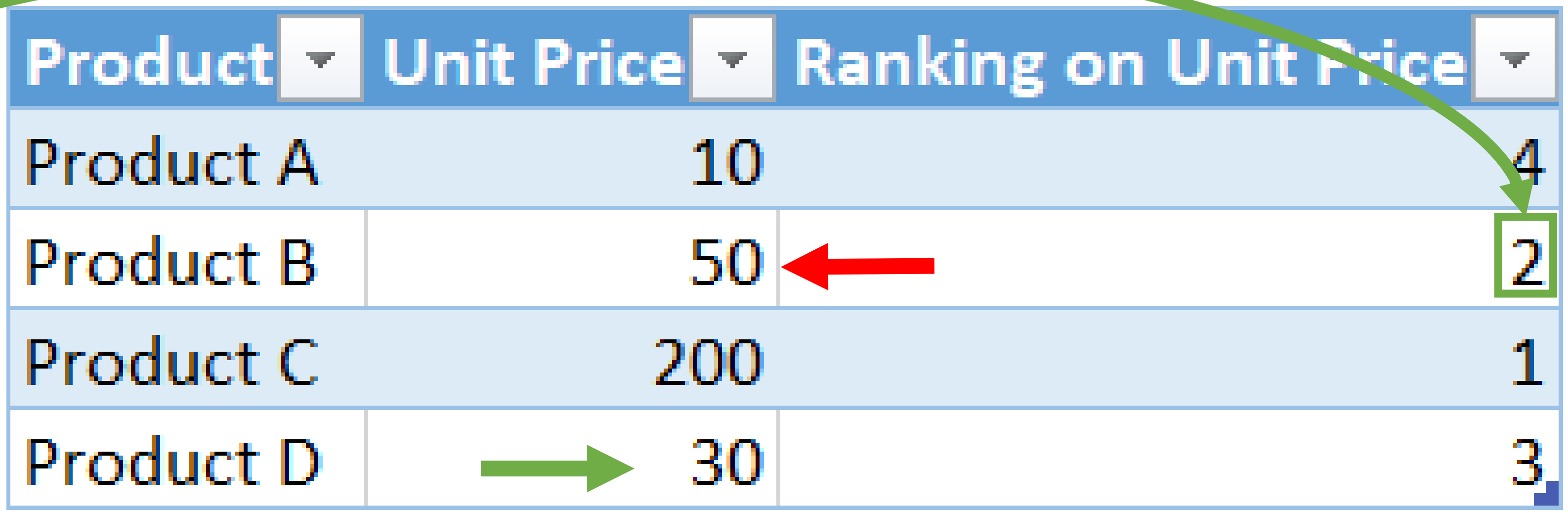
GREEN: Row context generated by FILTER() iterator function

How does the expression work?



```
Ranking on Unit Price =
    COUNTROWS(
        FILTER(
            ALL('Product'[Unit Price]),
            'Product'[Unit Price] >= EARLIER('Product'[Unit Price])
        )
    )
```

30 >= 50, FALSE, COUNTROWS = 2



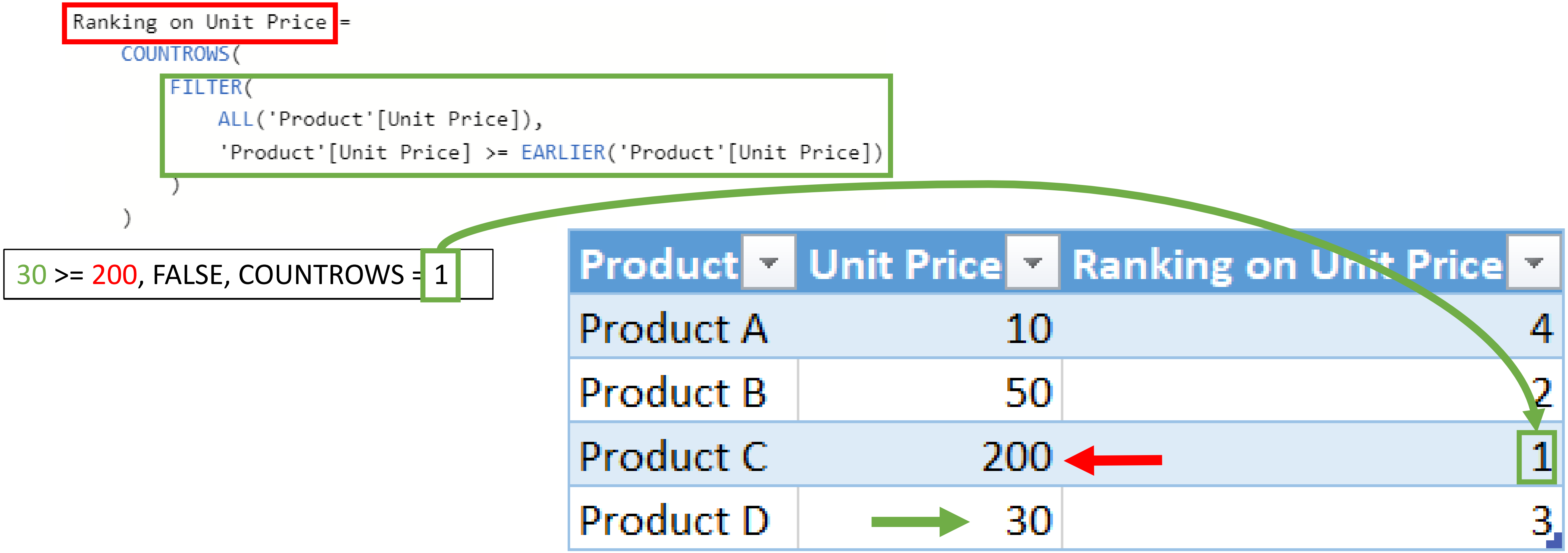
Product	Unit Price	Ranking on Unit Price
Product A	10	4
Product B	50	2
Product C	200	1
Product D	30	3

RED: Row context generated by the calculated column

GREEN: Row context generated by FILTER() iterator function

How does the expression work?

ALTRON | KARABINA



```
Ranking on Unit Price =
    COUNTROWS(
        FILTER(
            ALL('Product'[Unit Price]),
            'Product'[Unit Price] >= EARLIER('Product'[Unit Price])
        )
    )
```

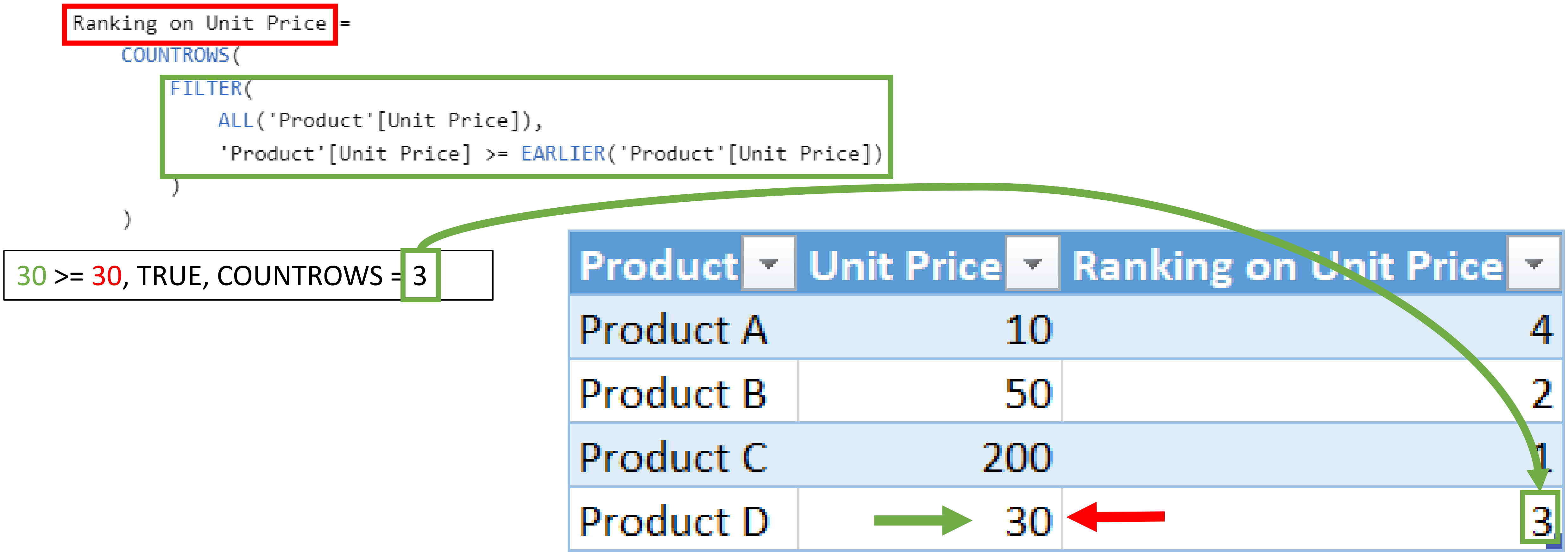
30 >= 200, FALSE, COUNTROWS = 1

Product	Unit Price	Ranking on Unit Price
Product A	10	4
Product B	50	2
Product C	200	1
Product D	30	3

RED: Row context generated by the calculated column

GREEN: Row context generated by FILTER() iterator function

How does the expression work?



```
Ranking on Unit Price =
    COUNTROWS(
        FILTER(
            ALL('Product'[Unit Price]),
            'Product'[Unit Price] >= EARLIER('Product'[Unit Price])
        )
    )
```

30 >= 30, TRUE, COUNTROWS = 3

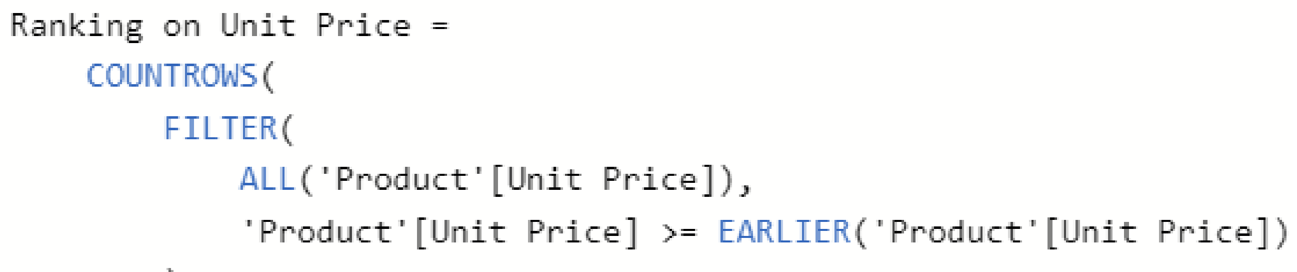
Product	Unit Price	Ranking on Unit Price
Product A	10	4
Product B	50	2
Product C	200	1
Product D	30	3

RED: Row context generated by the calculated column

GREEN: Row context generated by FILTER() iterator function

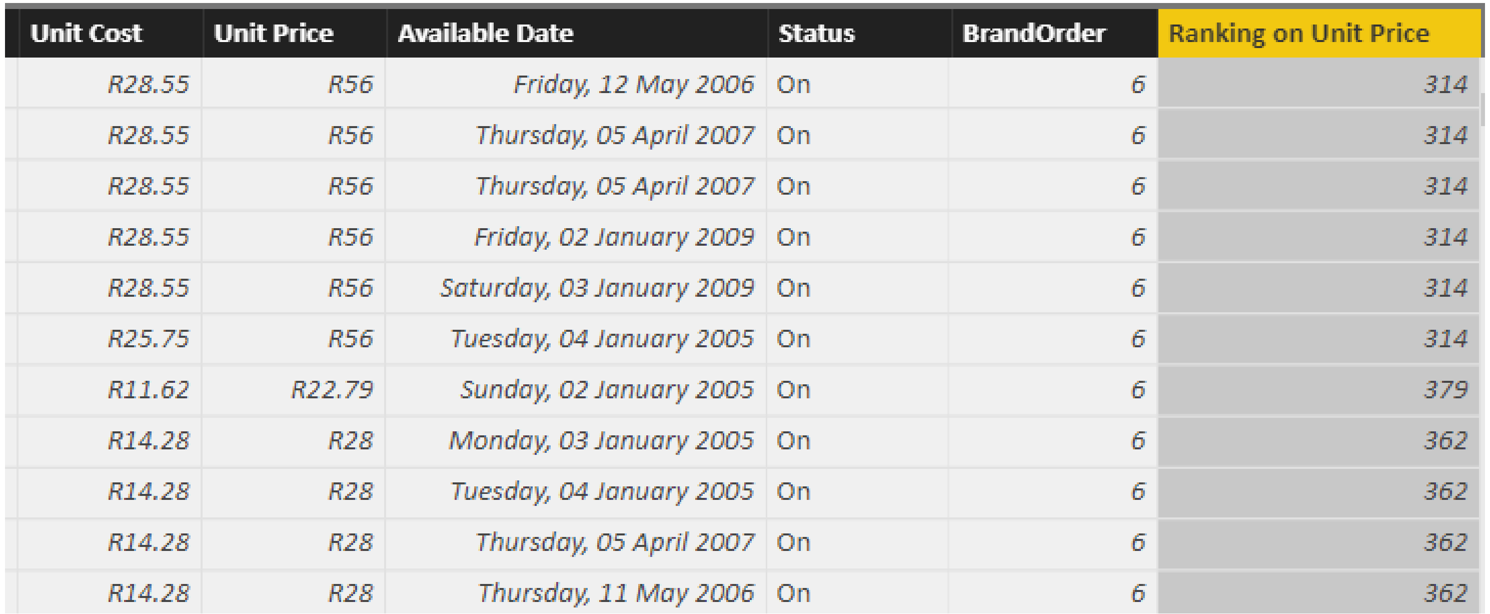
Ranking on price – Example

- Create the following calculated column in the Product table:



```
Ranking on Unit Price =
    COUNTROWS(
        FILTER(
            ALL('Product'[Unit Price]),
            'Product'[Unit Price] >= EARLIER('Product'[Unit Price])
        )
    )
```

- Note the output:



Unit Cost	Unit Price	Available Date	Status	BrandOrder	Ranking on Unit Price
R28.55	R56	Friday, 12 May 2006	On	6	314
R28.55	R56	Thursday, 05 April 2007	On	6	314
R28.55	R56	Thursday, 05 April 2007	On	6	314
R28.55	R56	Friday, 02 January 2009	On	6	314
R28.55	R56	Saturday, 03 January 2009	On	6	314
R25.75	R56	Tuesday, 04 January 2005	On	6	314
R11.62	R22.79	Sunday, 02 January 2005	On	6	379
R14.28	R28	Monday, 03 January 2005	On	6	362
R14.28	R28	Tuesday, 04 January 2005	On	6	362
R14.28	R28	Thursday, 05 April 2007	On	6	362
R14.28	R28	Thursday, 11 May 2006	On	6	362

Ranking on price – Example

- Create a Table visual on the report.
- Populate it with Ranking on Product[Ranking on Unit Price] and [_Total Revenue]
- Add data bars conditional formatting to [_Total Revenue] in the Table visual.

Another way to think about nested row contexts

- Imagine a piece of code that contains nested for loops:

```
for i = 1 to 10
    for j = 1 to 10
        if (i > j):
            return TRUE
    end
end
```

When i = 1, j will loop from 1 through 10.

i will then iterate to i = 2; j will loop from 1 through 10 again.

Another way to think about nested row contexts

- Now imagine you were only allowed to use one variable name, i.

```
for i = 1 to 10
    for i = 1 to 10
            if (i > i):
                    return TRUE
    end
end
```

- This cannot work, as there is an ambiguity as to the variable to which you are referring.

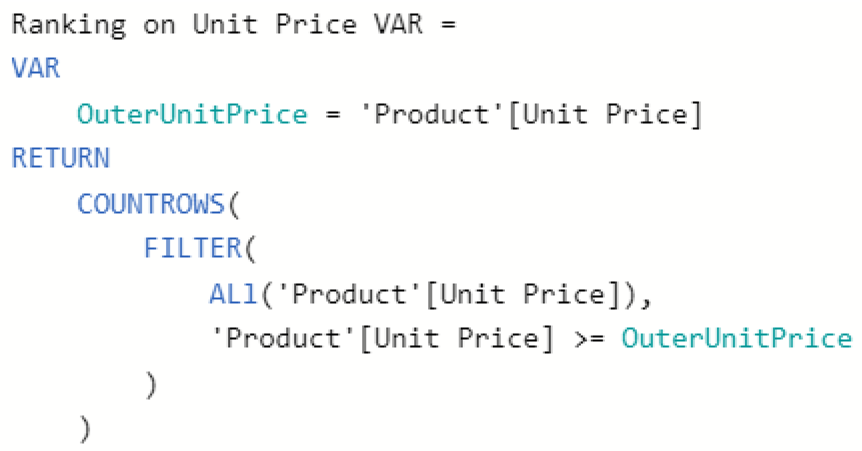
Another way to think about nested row contexts

```
for i = 1 to 10
    for i = 1 to 10
        if (i > EARLIER(i)):
            return TRUE
    end
end
```

- Now, the code is comparing the i from the inner for loop to the i from the outer for loop.
- This works as the EARLIER() function resolved the ambiguity.

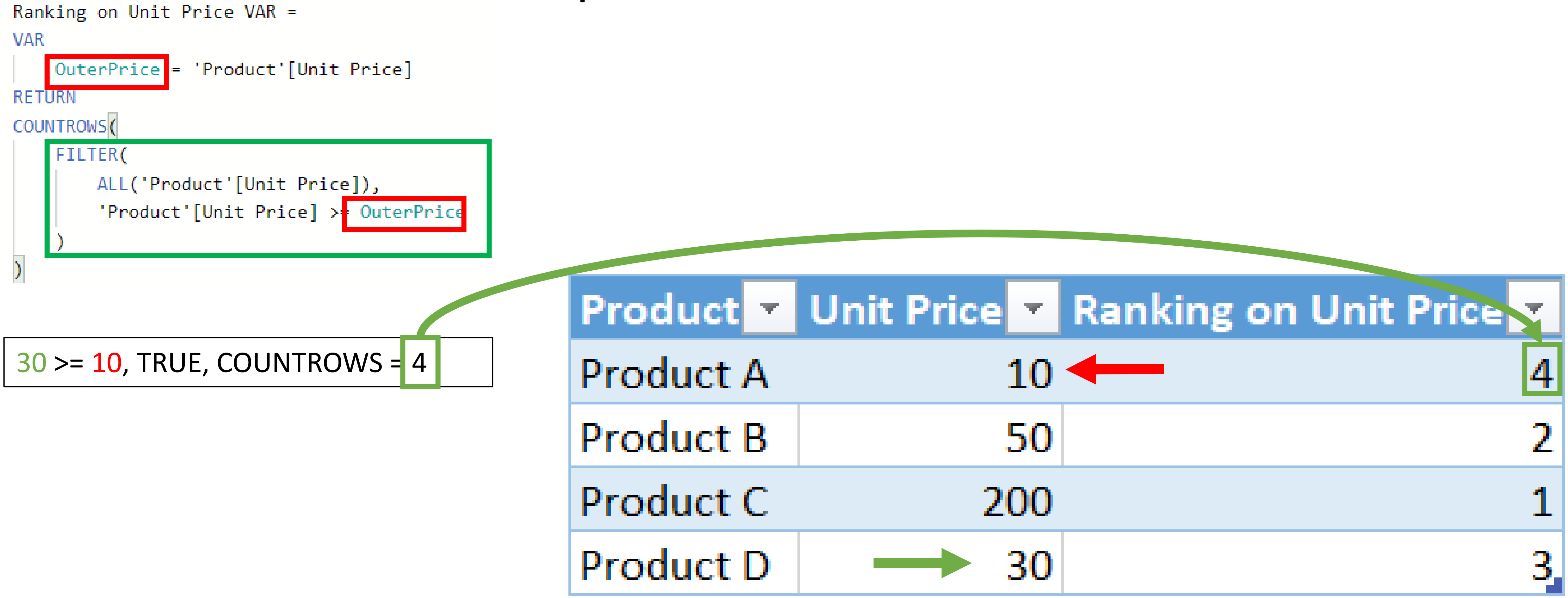
Variables

- To avoid using the EARLIER() function, you can assign and use variables.
- The variable is defined on the outer (or earlier) row context.
- You are now comparing the value found from the inner row context and comparing it against the value assigned to the variable from the outer row context.



```
Ranking on Unit Price VAR =
VAR
    OuterUnitPrice = 'Product'[Unit Price]
RETURN
    COUNTROWS(
        FILTER(
            All('Product'[Unit Price]),
            'Product'[Unit Price] >= OuterUnitPrice
        )
    )
```

How does the expression work?



```
Ranking on Unit Price VAR =
VAR
    OuterPrice = 'Product'[Unit Price]
RETURN
COUNTROWS(
    FILTER(
        ALL('Product'[Unit Price]),
        'Product'[Unit Price] >= OuterPrice
    )
)
```

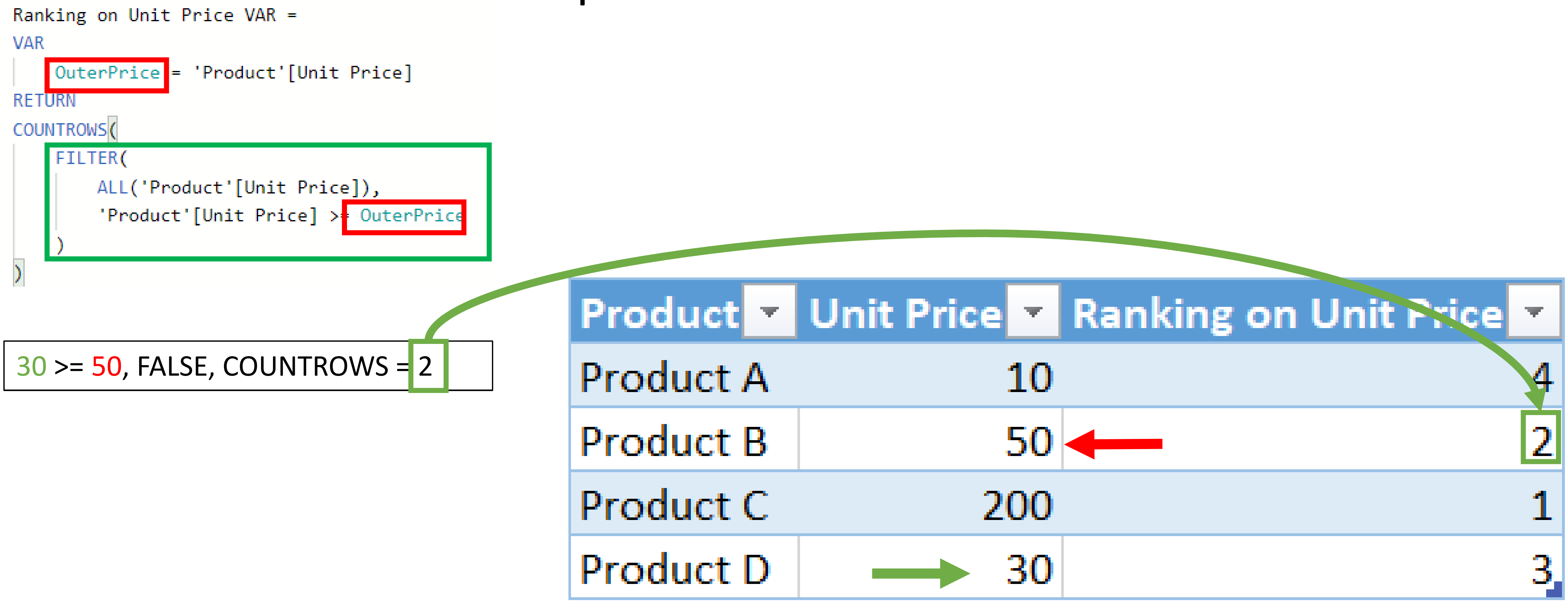
30 >= 10, TRUE, COUNTROWS = 4

Product	Unit Price	Ranking on Unit Price
Product A	10	4
Product B	50	2
Product C	200	1
Product D	30	3

RED: Row context generated by the calculated column

GREEN: Row context generated by FILTER() iterator function

How does the expression work?



```
Ranking on Unit Price VAR =
VAR
    OuterPrice = 'Product'[Unit Price]
RETURN
COUNTROWS(
    FILTER(
        ALL('Product'[Unit Price]),
        'Product'[Unit Price] >= OuterPrice
    )
)
```

30 >= 50, FALSE, COUNTROWS = 2

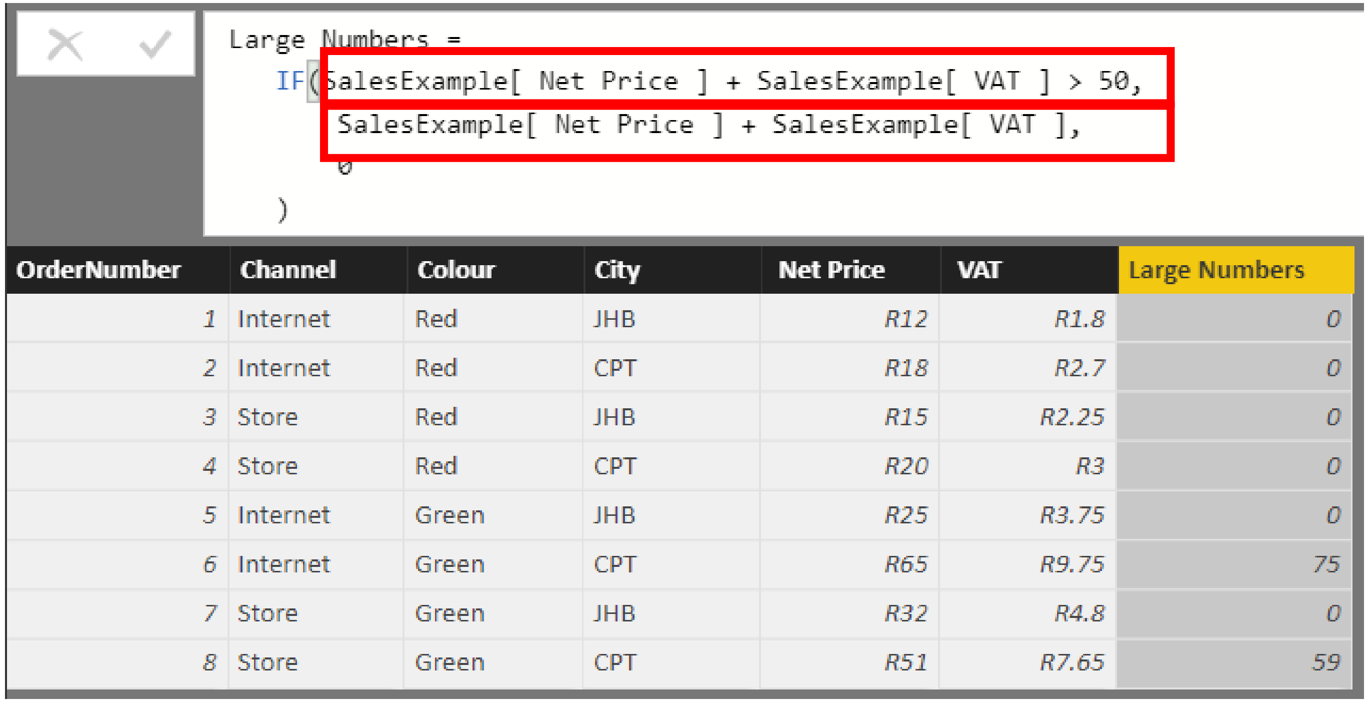
Product	Unit Price	Ranking on Unit Price
Product A	10	4
Product B	50	2
Product C	200	1
Product D	30	3

RED: Row context generated by the calculated column

GREEN: Row context generated by FILTER() iterator function

Variables to optimise code

- Variables can be used to optimise code.
- Consider the following conditional column:
  - If Net Price + VAT > 50, return that answer, otherwise return 0.
  - The calculation is being computed twice in this column:

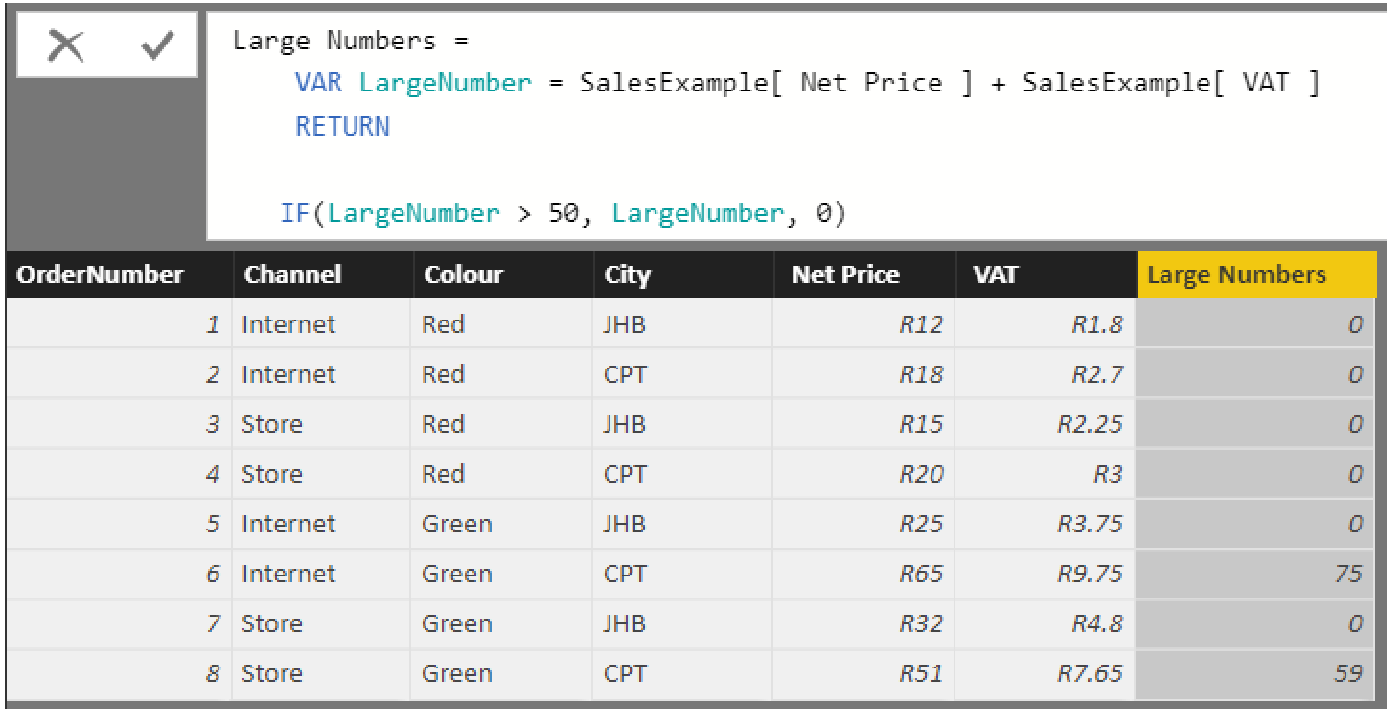


```
Large Numbers =
    IF(SalesExample[ Net Price ] + SalesExample[ VAT ] > 50,
       SalesExample[ Net Price ] + SalesExample[ VAT ],
       0
    )
```

OrderNumber	Channel	Colour	City	Net Price	VAT	Large Numbers
1	Internet	Red	JHB	R12	R1.8	0
2	Internet	Red	CPT	R18	R2.7	0
3	Store	Red	JHB	R15	R2.25	0
4	Store	Red	CPT	R20	R3	0
5	Internet	Green	JHB	R25	R3.75	0
6	Internet	Green	CPT	R65	R9.75	75
7	Store	Green	JHB	R32	R4.8	0
8	Store	Green	CPT	R51	R7.65	59

Variables to optimise code

- It is more efficient to carry out the calculation once, and save the result in a variable.
- In a large table with a complex calculation, this can save a significant amount of compute time.



```
Large Numbers =
    VAR LargeNumber = SalesExample[ Net Price ] + SalesExample[ VAT ]
    RETURN

   IF(LargeNumber > 50, LargeNumber, 0)
```

OrderNumber	Channel	Colour	City	Net Price	VAT	Large Numbers
1	Internet	Red	JHB	R12	R1.8	0
2	Internet	Red	CPT	R18	R2.7	0
3	Store	Red	JHB	R15	R2.25	0
4	Store	Red	CPT	R20	R3	0
5	Internet	Green	JHB	R25	R3.75	0
6	Internet	Green	CPT	R65	R9.75	75
7	Store	Green	JHB	R32	R4.8	0
8	Store	Green	CPT	R51	R7.65	59

Variables – Example

- The Executive Committee has promised the customers that every delivery will be delivered within 7 days.

- The Committee wants to see if it takes longer than 7 days to deliver a product from the order date.

- Create a calculated column in the Sales table that shows by how many days over the deadline was the delivery made:

- Code can be copied from the following slide.

Variables – Example

```
Long Delivery Time =
    VAR
        DeliveryDays =
            DATE(
                LEFT(Sales[DeliveryDateKey], 4),
                MID(Sales[DeliveryDateKey], 5, 2),
                RIGHT(Sales[DeliveryDateKey], 2)) -

            DATE(
                LEFT(Sales[OrderDateKey], 4),
                MID(Sales[OrderDateKey], 5, 2),
                RIGHT(Sales[OrderDateKey], 2))
    RETURN

    IF(DeliveryDays > 7, DeliveryDays - 7, 0)
```

Variables – Example

- Note the output. For every delivery that took less than the allotted time, the result is 0.

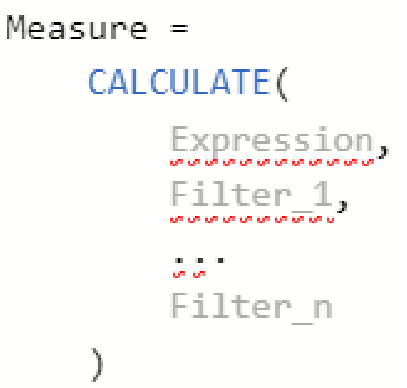
- For every deliver that took longer than 7 days, the number of days later is the output.

CALCULATE()

- CALCULATE() function is the most important function in DAX.

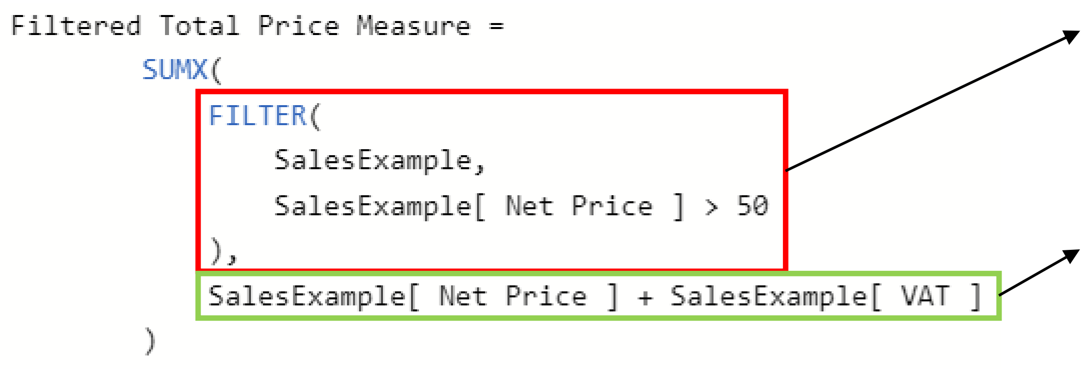
- It allows you to evaluate an expression, with the application of user defined filters.

- CALCULATE() changes the filter context against which the expression is evaluated.
    - More precisely, it creates a new filter context, based on the original one.



```
Measure =
    CALCULATE(
        Expression,
        Filter_1,
        ...
        Filter_n
    )
```

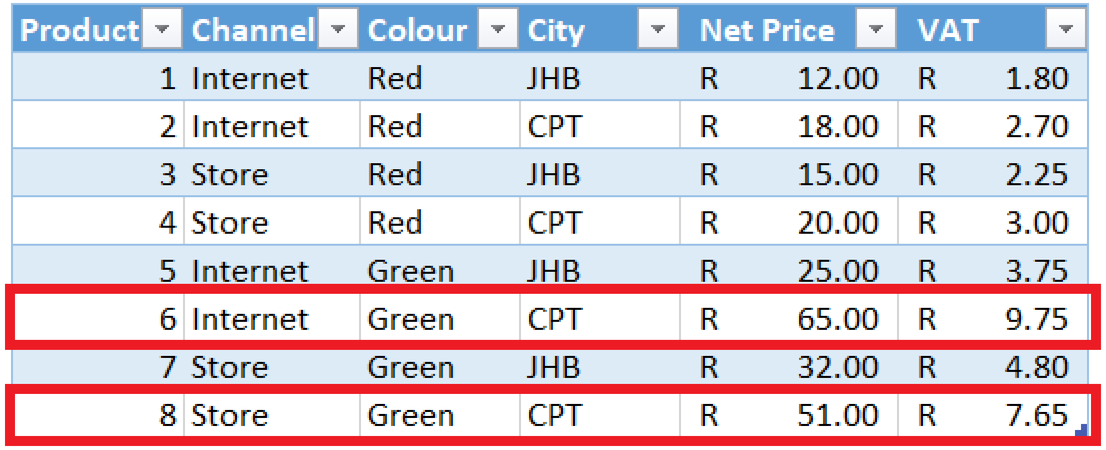
Row Context in Iterators



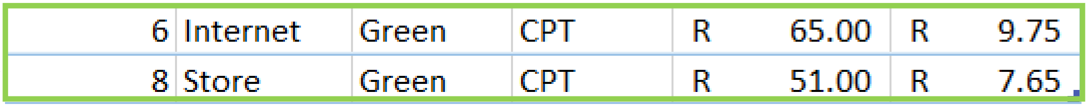
```
Filtered Total Price Measure =
        SUMX(
            FILTER(
                SalesExample,
                SalesExample[ Net Price ] > 50
            ),
            SalesExample[ Net Price ] + SalesExample[ VAT ]
        )
```

FILTER() applies a row context over the table. It iterates over the table to find the rows were the filter condition is met.

FILTER() then returns the rows to SUMX() which iterates over these rows and evaluates the expression



Product	Channel	Colour	City	Net Price	VAT
1	Internet	Red	JHB	R 12.00	R 1.80
2	Internet	Red	CPT	R 18.00	R 2.70
3	Store	Red	JHB	R 15.00	R 2.25
4	Store	Red	CPT	R 20.00	R 3.00
5	Internet	Green	JHB	R 25.00	R 3.75
6	Internet	Green	CPT	R 65.00	R 9.75
7	Store	Green	JHB	R 32.00	R 4.80
8	Store	Green	CPT	R 51.00	R 7.65

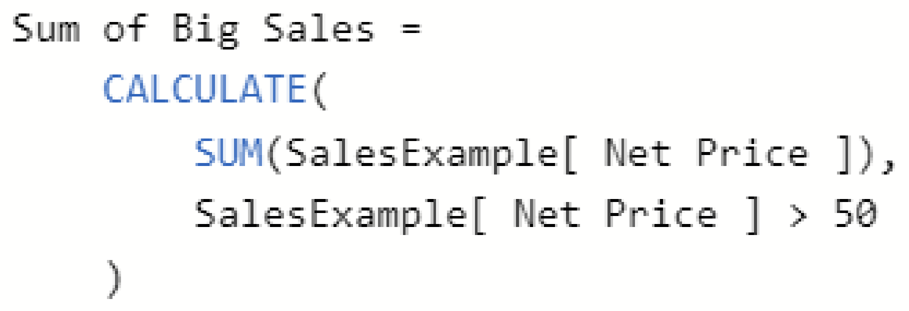


6	Internet	Green	CPT	R 65.00	R 9.75
8	Store	Green	CPT	R 51.00	R 7.65

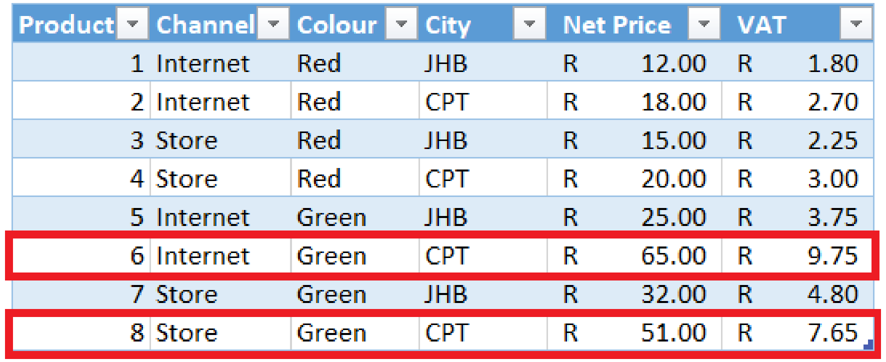
Output: R133.40

CALCULATE()

- Example



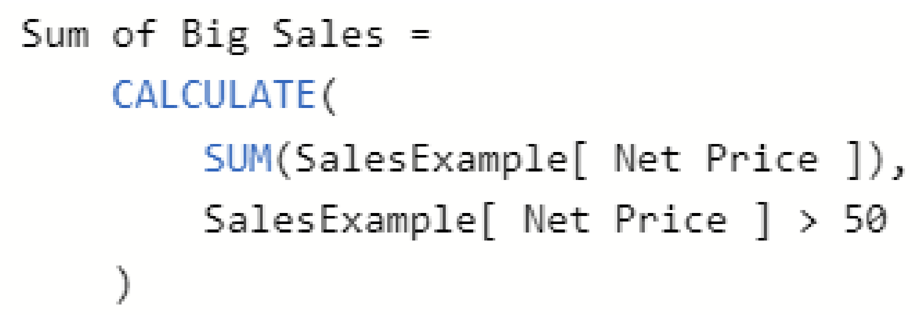
```
Sum of Big Sales =
    CALCULATE(
        SUM(SalesExample[ Net Price ]),
        SalesExample[ Net Price ] > 50
    )
```



Product	Channel	Colour	City	Net Price	VAT
1	Internet	Red	JHB	R 12.00	R 1.80
2	Internet	Red	CPT	R 18.00	R 2.70
3	Store	Red	JHB	R 15.00	R 2.25
4	Store	Red	CPT	R 20.00	R 3.00
5	Internet	Green	JHB	R 25.00	R 3.75
6	Internet	Green	CPT	R 65.00	R 9.75
7	Store	Green	JHB	R 32.00	R 4.80
8	Store	Green	CPT	R 51.00	R 7.65

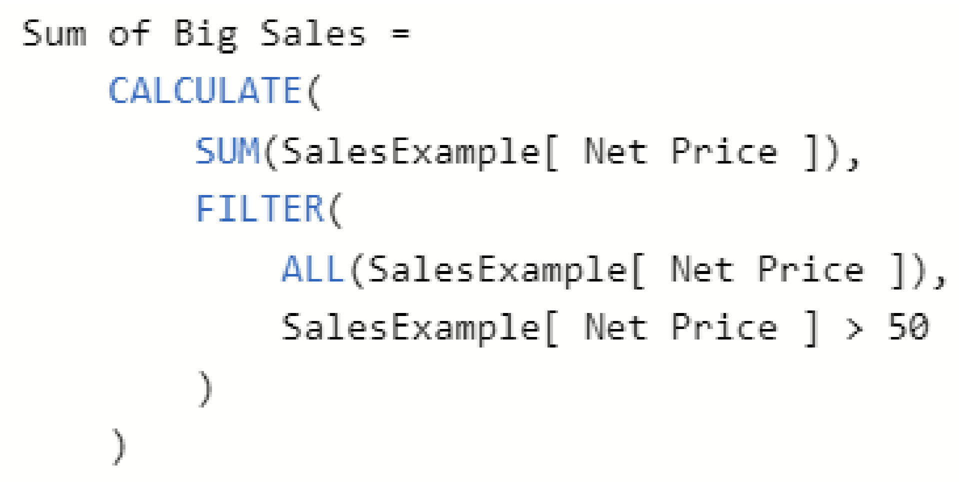
- Result = R126.00

CALCULATE()



```
Sum of Big Sales =
    CALCULATE(
        SUM(SalesExample[ Net Price ]),
        SalesExample[ Net Price ] > 50
    )
```

Is exactly equal to:



```
Sum of Big Sales =
    CALCULATE(
        SUM(SalesExample[ Net Price ]),
        FILTER(
            ALL(SalesExample[ Net Price ]),
            SalesExample[ Net Price ] > 50
        )
    )
```

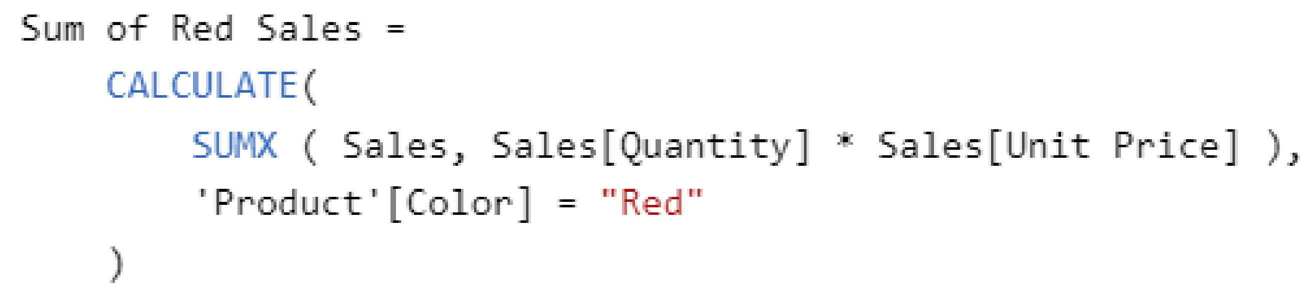
The ALL() function ignores any filter on SalesExample[Net Price].

The FILTER() function iterates over all the rows returned by ALL() and it in turn returns all the rows where SalesExample[Net Price] > 50.

CALCULATE()

- CALCULATE() modifies the filter context in which a measure is being evaluated.

- Therefore, if a filter is added on a product’s colour, for example:

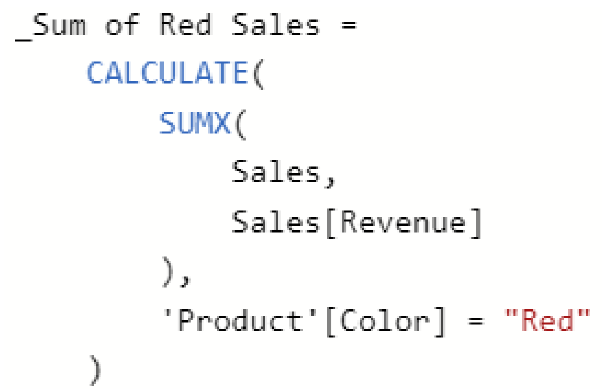


```
Sum of Red Sales =
    CALCULATE(
        SUMX ( Sales, Sales[Quantity] * Sales[Unit Price] ),
        'Product'[Color] = "Red"
    )
```

Any visual that uses the measure in which ‘color’ is a filter context, CALCULATE() will overwrite that visual’s filter context and replace it with its own.

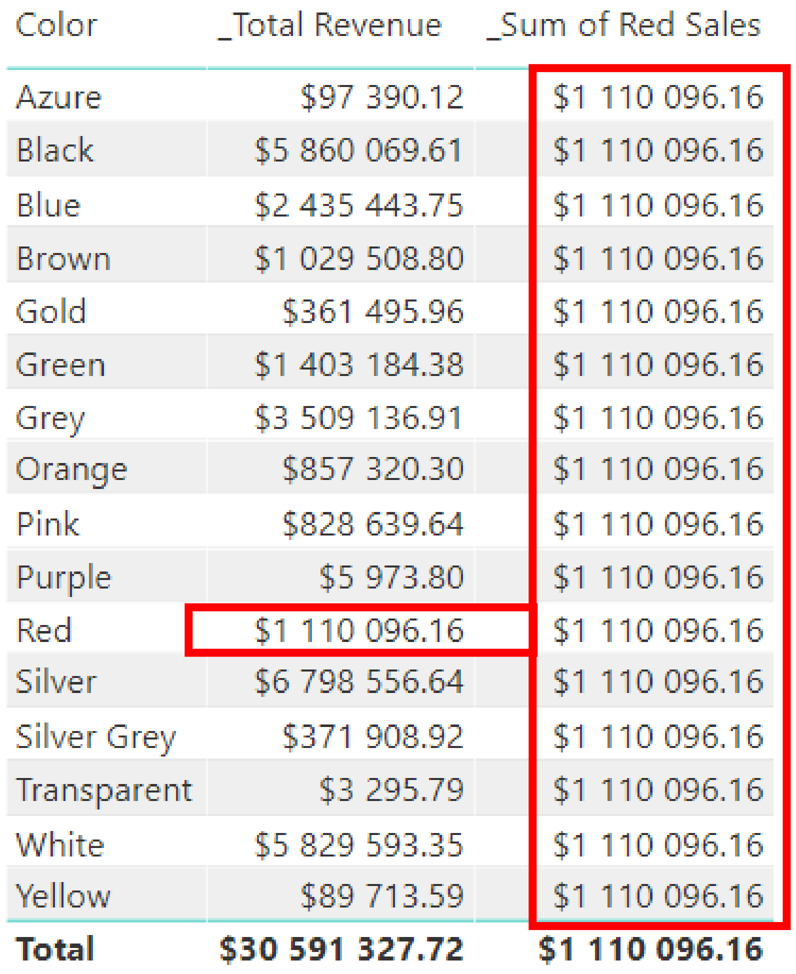
CALCULATE()

- Example of CALCULATE() changing the filter context.
  - Create the following measure:



```
_Sum of Red Sales =
    CALCULATE(
        SUMX(
            Sales,
            Sales[Revenue]
        ),
        'Product'[Color] = "Red"
    )
```

- Total sales will be evaluated in the ‘Color’ filter context given by the table.
- Red has total sales of $1,110,096.16
- Since CALCULATE() replaces the filter context on ‘Color’, anywhere where ‘Color’ appears, the result will be the sum of Red sales in all rows of the table



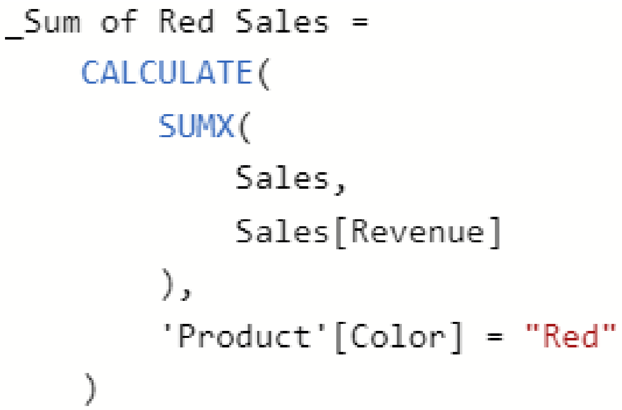
Color	_Total Revenue	_Sum of Red Sales
Azure	$97 390.12	$1 110 096.16
Black	$5 860 069.61	$1 110 096.16
Blue	$2 435 443.75	$1 110 096.16
Brown	$1 029 508.80	$1 110 096.16
Gold	$361 495.96	$1 110 096.16
Green	$1 403 184.38	$1 110 096.16
Grey	$3 509 136.91	$1 110 096.16
Orange	$857 320.30	$1 110 096.16
Pink	$828 639.64	$1 110 096.16
Purple	$5 973.80	$1 110 096.16
Red	$1 110 096.16	$1 110 096.16
Silver	$6 798 556.64	$1 110 096.16
Silver Grey	$371 908.92	$1 110 096.16
Transparent	$3 295.79	$1 110 096.16
White	$5 829 593.35	$1 110 096.16
Yellow	$89 713.59	$1 110 096.16
Total	**$30 591 327.72**	**$1 110 096.16**

CALCULATE()

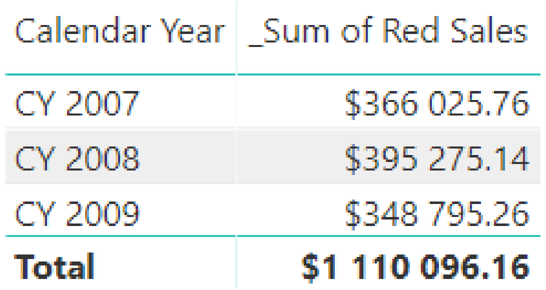
- It is important to note that CALCULATE() will only change the filter context where ‘Product’[Color] is involved.

- CALCULATE() will still respect every other filter context.

- Note, _Sum of Red Sales by Calendar Year will show a different number for each year.



```
_Sum of Red Sales =
    CALCULATE(
        SUMX(
            Sales,
            Sales[Revenue]
        ),
        'Product'[Color] = "Red"
    )
```



Calendar Year	_Sum of Red Sales
CY 2007	$366 025.76
CY 2008	$395 275.14
CY 2009	$348 795.26
Total	**$1 110 096.16**

CALCULATE()

- CALCULATE() allows you to use filters from different tables in your model.

- You can use as many filters as you need in CALCULATE().

Time Intelligence

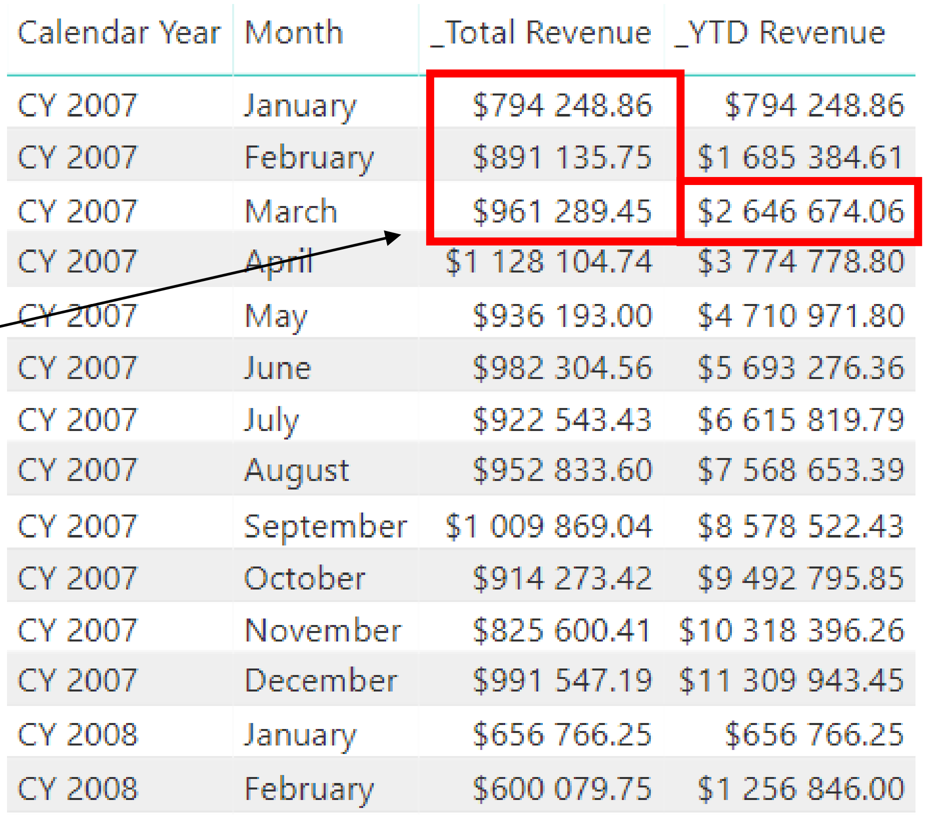
- Time intelligence functions in DAX allow you to calculate:
  - Year/Quarter/Month To Date
  - Running totals
  - Metrics based on a fact table with more than one date field.

- These functions rely on having a Date Table

- You need an understanding of the CALCULATE() function.

Aggregations over time (YTD)

- Consider the following table:
- We want to find an expression that can calculate the YTD total for each month.

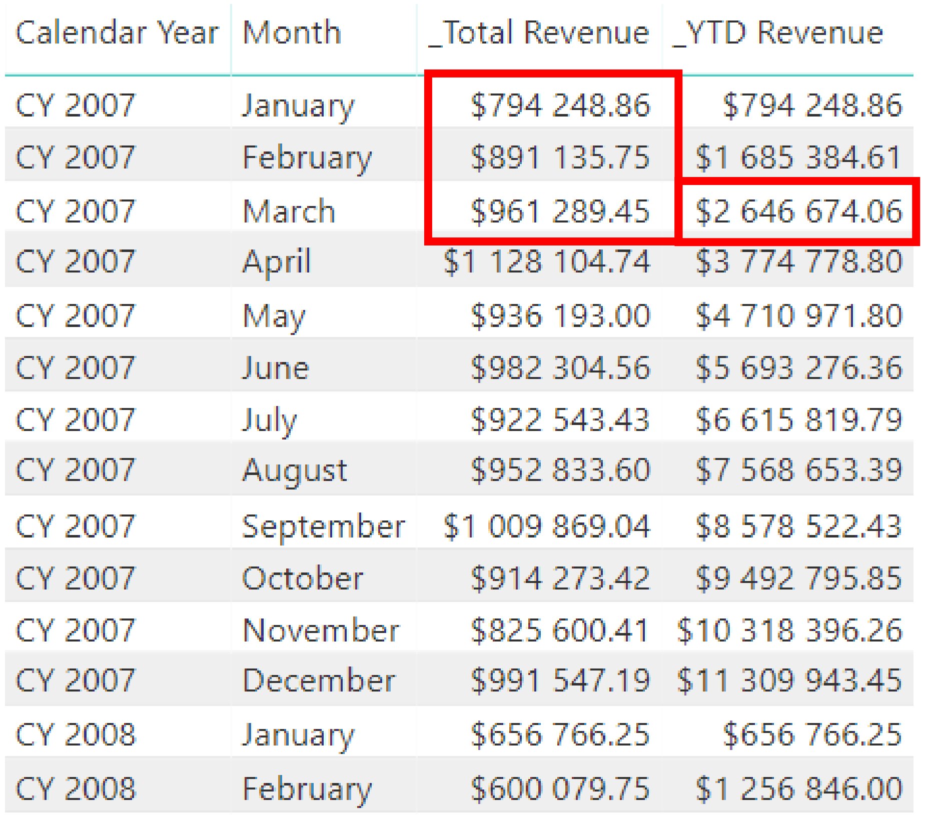


Calendar Year	Month	_Total Revenue	_YTD Revenue
CY 2007	January	$794 248.86	$794 248.86
CY 2007	February	$891 135.75	$1 685 384.61
CY 2007	March	$961 289.45	$2 646 674.06
CY 2007	April	$1 128 104.74	$3 774 778.80
CY 2007	May	$936 193.00	$4 710 971.80
CY 2007	June	$982 304.56	$5 693 276.36
CY 2007	July	$922 543.43	$6 615 819.79
CY 2007	August	$952 833.60	$7 568 653.39
CY 2007	September	$1 009 869.04	$8 578 522.43
CY 2007	October	$914 273.42	$9 492 795.85
CY 2007	November	$825 600.41	$10 318 396.26
CY 2007	December	$991 547.19	$11 309 943.45
CY 2008	January	$656 766.25	$656 766.25
CY 2008	February	$600 079.75	$1 256 846.00

Where the sum of these 3 rows equals the YTD Total for March 2007.

Aggregations over time (YTD)

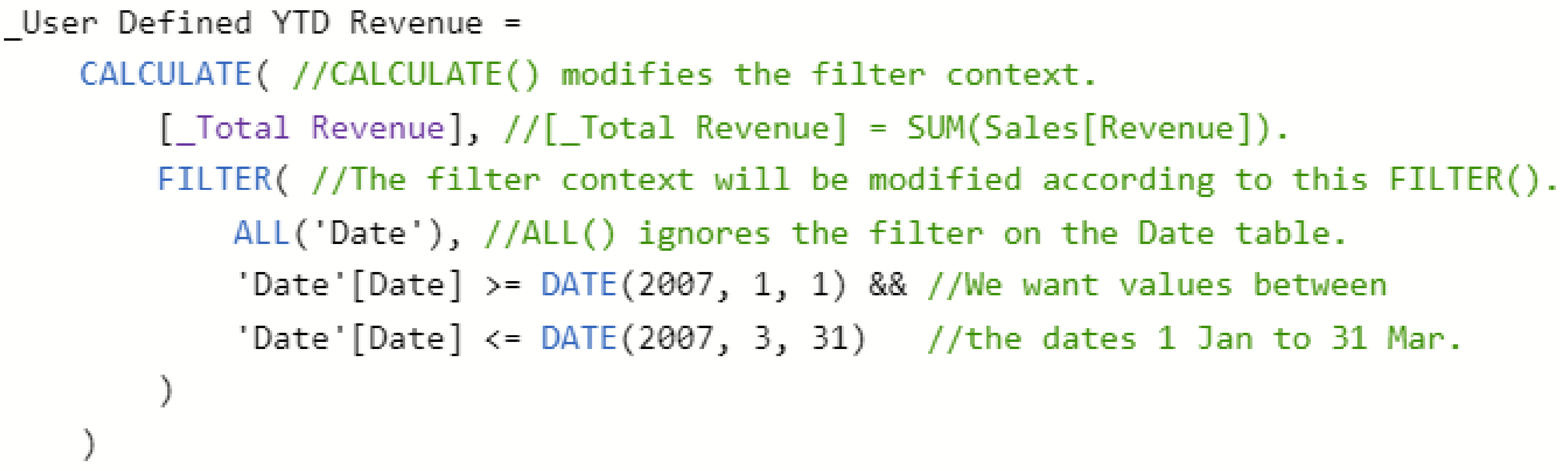
- We can’t use the *previous row* concept in DAX as it doesn’t exist.
- Our expression must therefore use all the values in the time interval 1 January to 31 March.
- At the moment, the table visual is setting the filter context such that only the days of one month are visible.
- We need to use a function that can change the filter context of the measure.



Calendar Year	Month	_Total Revenue	_YTD Revenue
CY 2007	January	$794 248.86	$794 248.86
CY 2007	February	$891 135.75	$1 685 384.61
CY 2007	March	$961 289.45	$2 646 674.06
CY 2007	April	$1 128 104.74	$3 774 778.80
CY 2007	May	$936 193.00	$4 710 971.80
CY 2007	June	$982 304.56	$5 693 276.36
CY 2007	July	$922 543.43	$6 615 819.79
CY 2007	August	$952 833.60	$7 568 653.39
CY 2007	September	$1 009 869.04	$8 578 522.43
CY 2007	October	$914 273.42	$9 492 795.85
CY 2007	November	$825 600.41	$10 318 396.26
CY 2007	December	$991 547.19	$11 309 943.45
CY 2008	January	$656 766.25	$656 766.25
CY 2008	February	$600 079.75	$1 256 846.00

Aggregations over time (YTD)

- Create the following measure in the Sales table



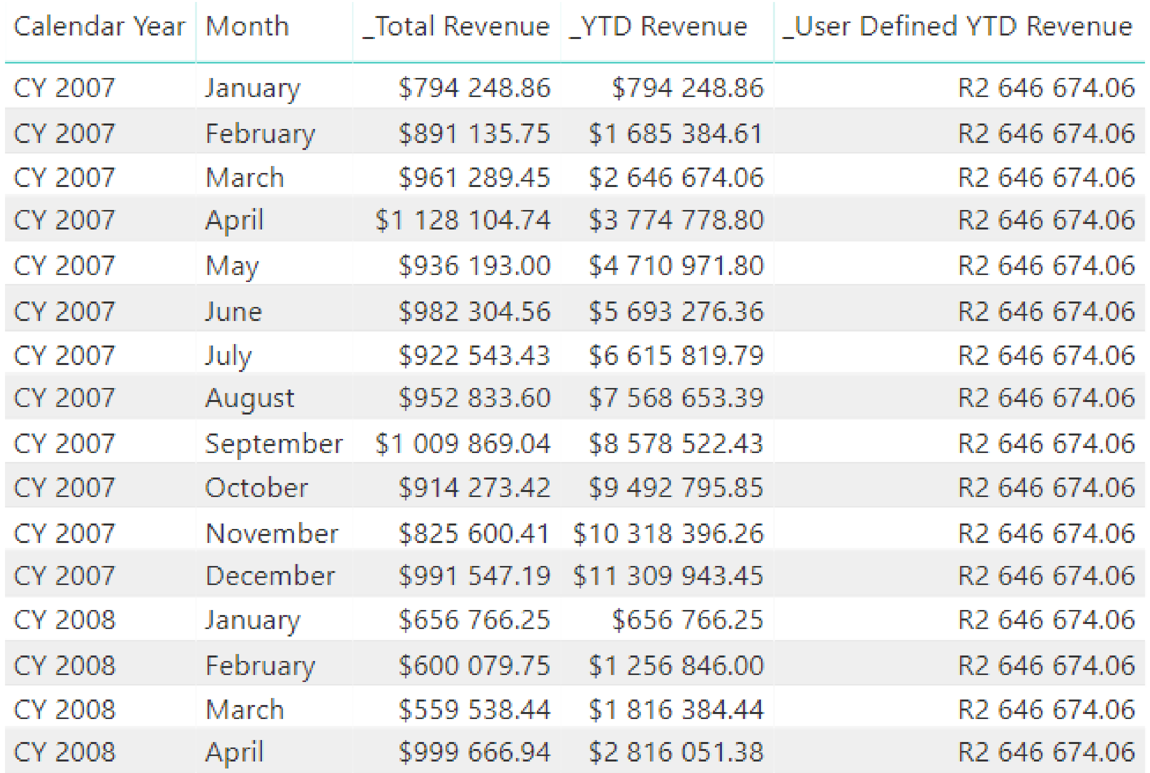
```
_User Defined YTD Revenue =
    CALCULATE( //CALCULATE() modifies the filter context.
        [_Total Revenue], //[_Total Revenue] = SUM(Sales[Revenue]).
        FILTER( //The filter context will be modified according to this FILTER().
            ALL('Date'), //ALL() ignores the filter on the Date table.
            'Date'[Date] >= DATE(2007, 1, 1) && //We want values between
            'Date'[Date] <= DATE(2007, 3, 31)   //the dates 1 Jan to 31 Mar.
        )
    )
```

Let’s see the result on the table

- All the rows of our measure contain the same value.

- This is because we have removed the filter context over the entire Date table.

- We need to modify this code to get it working properly.



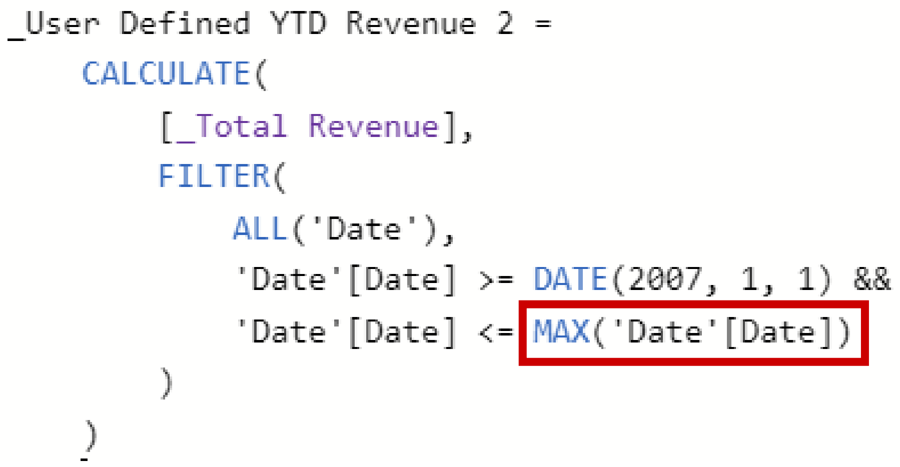
Calendar Year	Month	_Total Revenue	_YTD Revenue	_User Defined YTD Revenue
CY 2007	January	$794 248.86	$794 248.86	R2 646 674.06
CY 2007	February	$891 135.75	$1 685 384.61	R2 646 674.06
CY 2007	March	$961 289.45	$2 646 674.06	R2 646 674.06
CY 2007	April	$1 128 104.74	$3 774 778.80	R2 646 674.06
CY 2007	May	$936 193.00	$4 710 971.80	R2 646 674.06
CY 2007	June	$982 304.56	$5 693 276.36	R2 646 674.06
CY 2007	July	$922 543.43	$6 615 819.79	R2 646 674.06
CY 2007	August	$952 833.60	$7 568 653.39	R2 646 674.06
CY 2007	September	$1 009 869.04	$8 578 522.43	R2 646 674.06
CY 2007	October	$914 273.42	$9 492 795.85	R2 646 674.06
CY 2007	November	$825 600.41	$10 318 396.26	R2 646 674.06
CY 2007	December	$991 547.19	$11 309 943.45	R2 646 674.06
CY 2008	January	$656 766.25	$656 766.25	R2 646 674.06
CY 2008	February	$600 079.75	$1 256 846.00	R2 646 674.06
CY 2008	March	$559 538.44	$1 816 384.44	R2 646 674.06
CY 2008	April	$999 666.94	$2 816 051.38	R2 646 674.06

Modifications – Cumulative Sum

- We can modify the expression, such that the date is filtered dynamically to the MAX date for the filter context given by the visual.

- Create a new measure as follows:

- This creates the cumulative sum….

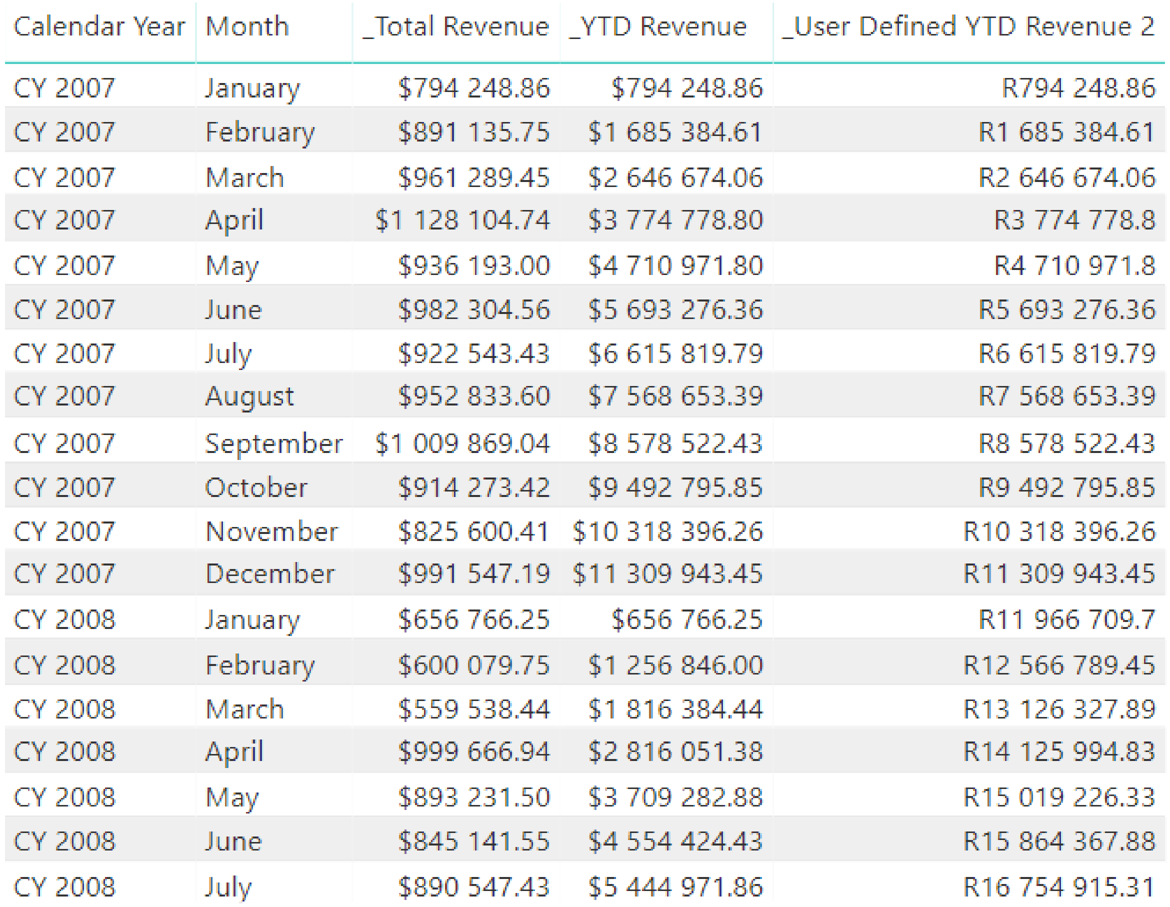


```
_User Defined YTD Revenue 2 =
    CALCULATE(
        [_Total Revenue],
        FILTER(
            ALL('Date'),
            'Date'[Date] >= DATE(2007, 1, 1) &&
            'Date'[Date] <= MAX('Date'[Date])
        )
    )
```

Let’s see the result on the table

- That’s looking better, but the values do not start over on the 1st of the next year.

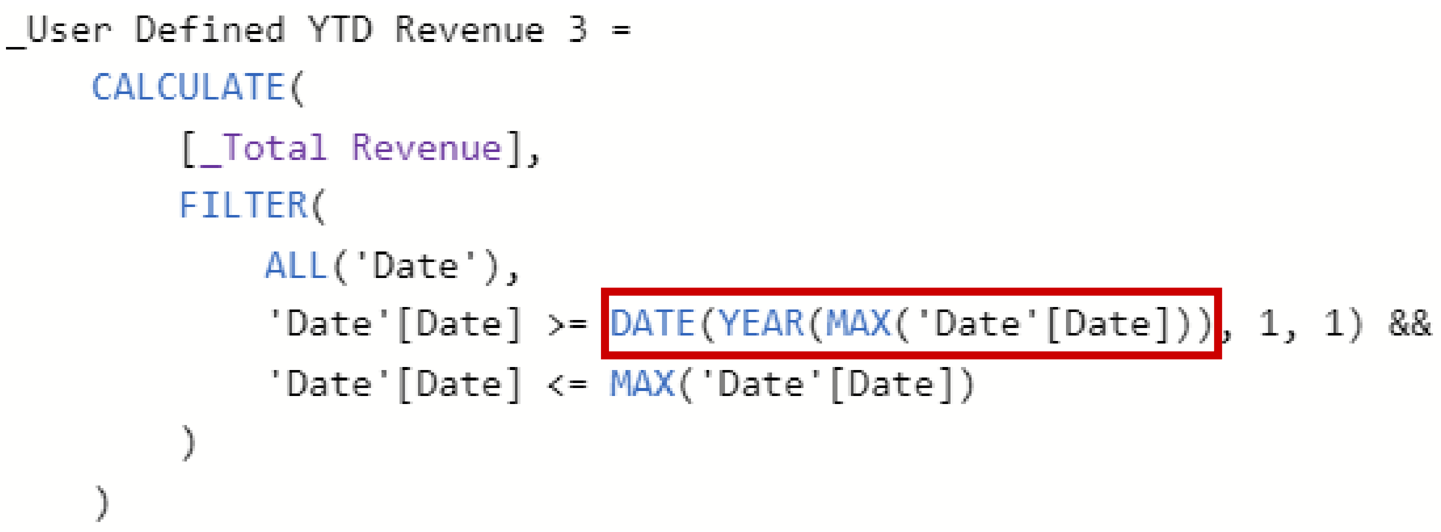
- What we have created is a cumulative sum.



Calendar Year	Month	_Total Revenue	_YTD Revenue	_User Defined YTD Revenue 2
CY 2007	January	$794 248.86	$794 248.86	R794 248.86
CY 2007	February	$891 135.75	$1 685 384.61	R1 685 384.61
CY 2007	March	$961 289.45	$2 646 674.06	R2 646 674.06
CY 2007	April	$1 128 104.74	$3 774 778.80	R3 774 778.8
CY 2007	May	$936 193.00	$4 710 971.80	R4 710 971.8
CY 2007	June	$982 304.56	$5 693 276.36	R5 693 276.36
CY 2007	July	$922 543.43	$6 615 819.79	R6 615 819.79
CY 2007	August	$952 833.60	$7 568 653.39	R7 568 653.39
CY 2007	September	$1 009 869.04	$8 578 522.43	R8 578 522.43
CY 2007	October	$914 273.42	$9 492 795.85	R9 492 795.85
CY 2007	November	$825 600.41	$10 318 396.26	R10 318 396.26
CY 2007	December	$991 547.19	$11 309 943.45	R11 309 943.45
CY 2008	January	$656 766.25	$656 766.25	R11 966 709.7
CY 2008	February	$600 079.75	$1 256 846.00	R12 566 789.45
CY 2008	March	$559 538.44	$1 816 384.44	R13 126 327.89
CY 2008	April	$999 666.94	$2 816 051.38	R14 125 994.83
CY 2008	May	$893 231.50	$3 709 282.88	R15 019 226.33
CY 2008	June	$845 141.55	$4 554 424.43	R15 864 367.88
CY 2008	July	$890 547.43	$5 444 971.86	R16 754 915.31

Modifications

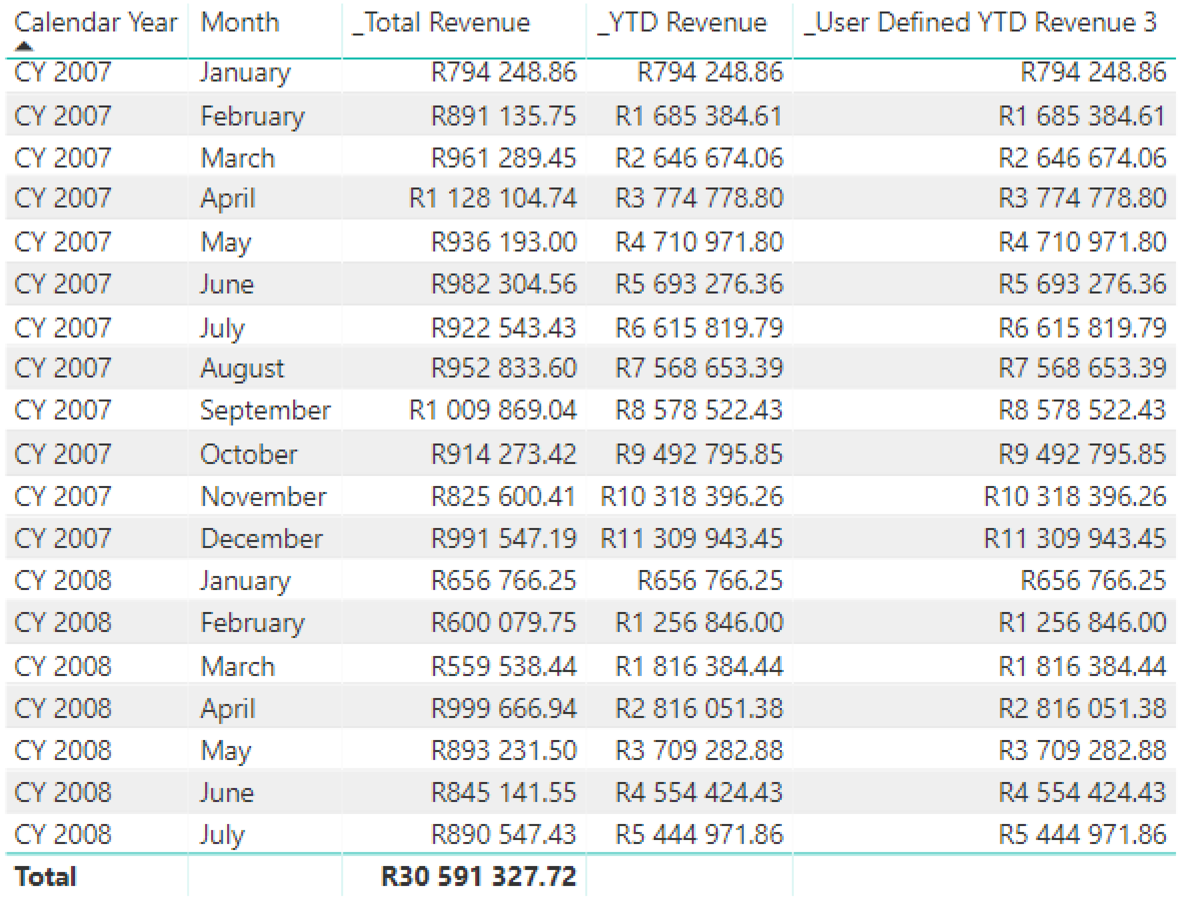
- We modify the start date to retrieve the value of the year as defined in the current filter context.



```
_User Defined YTD Revenue 3 =
    CALCULATE(
        [_Total Revenue],
        FILTER(
            ALL('Date'),
            'Date'[Date] >= DATE(YEAR(MAX('Date'[Date])), 1, 1) &&
            'Date'[Date] <= MAX('Date'[Date])
        )
    )
```

Let’s see the result on the table

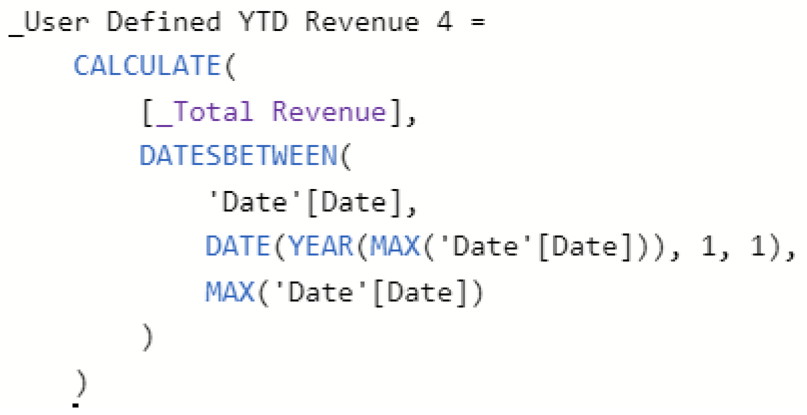
- That works as intended.



Calendar Year	Month	_Total Revenue	_YTD Revenue	_User Defined YTD Revenue 3
CY 2007	January	R794 248.86	R794 248.86	R794 248.86
CY 2007	February	R891 135.75	R1 685 384.61	R1 685 384.61
CY 2007	March	R961 289.45	R2 646 674.06	R2 646 674.06
CY 2007	April	R1 128 104.74	R3 774 778.80	R3 774 778.80
CY 2007	May	R936 193.00	R4 710 971.80	R4 710 971.80
CY 2007	June	R982 304.56	R5 693 276.36	R5 693 276.36
CY 2007	July	R922 543.43	R6 615 819.79	R6 615 819.79
CY 2007	August	R952 833.60	R7 568 653.39	R7 568 653.39
CY 2007	September	R1 009 869.04	R8 578 522.43	R8 578 522.43
CY 2007	October	R914 273.42	R9 492 795.85	R9 492 795.85
CY 2007	November	R825 600.41	R10 318 396.26	R10 318 396.26
CY 2007	December	R991 547.19	R11 309 943.45	R11 309 943.45
CY 2008	January	R656 766.25	R656 766.25	R656 766.25
CY 2008	February	R600 079.75	R1 256 846.00	R1 256 846.00
CY 2008	March	R559 538.44	R1 816 384.44	R1 816 384.44
CY 2008	April	R999 666.94	R2 816 051.38	R2 816 051.38
CY 2008	May	R893 231.50	R3 709 282.88	R3 709 282.88
CY 2008	June	R845 141.55	R4 554 424.43	R4 554 424.43
CY 2008	July	R890 547.43	R5 444 971.86	R5 444 971.86
Total		**R30 591 327.72**		

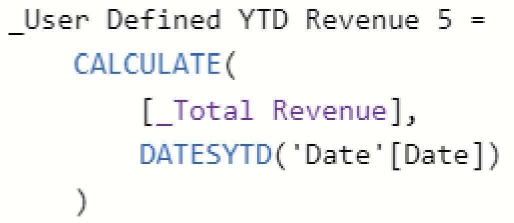
Further Modifications

- We can now replace the FILTER() function with DATESBETWEEN()



```
_User Defined YTD Revenue 4 =
    CALCULATE(
        [_Total Revenue],
        DATESBETWEEN(
            'Date'[Date],
            DATE(YEAR(MAX('Date'[Date])), 1, 1),
            MAX('Date'[Date])
        )
    )
```

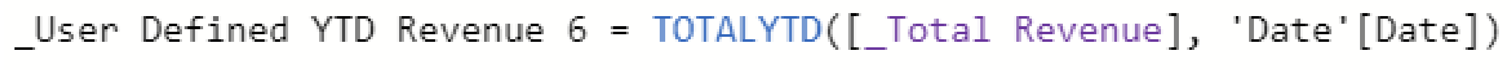
- We can simplify the code even further, by using DATESYTD:



```
_User Defined YTD Revenue 5 =
    CALCULATE(
        [_Total Revenue],
        DATESYTD('Date'[Date])
    )
```

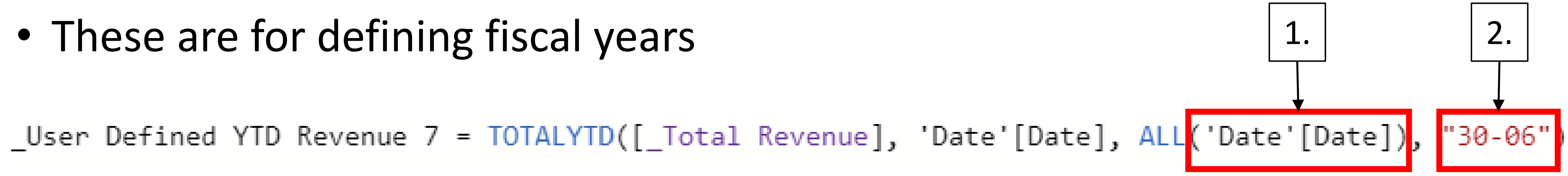
Further Modifications

- Even simpler syntax is the TOTALYTD() function



```
_User Defined YTD Revenue 6 = TOTALYTD([_Total Revenue], 'Date'[Date])
```

- Notice that TOTALYTD() and DATESYTD() have additional optional arguments.
    - These are for defining fiscal years

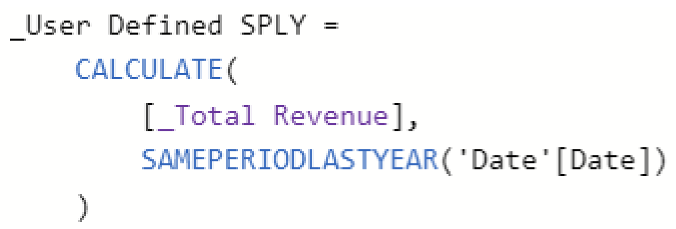


```
_User Defined YTD Revenue 7 = TOTALYTD([_Total Revenue], 'Date'[Date], ALL('Date'[Date]), "30-06")
```

1. Use the entire date table. ALL() function is very important here.
2. Define the last day of the fiscal year.

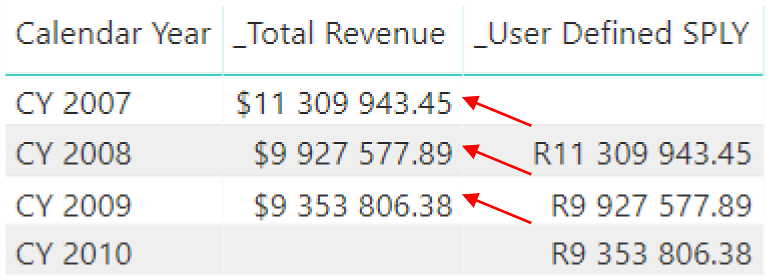
Same period last year (or month)

- Consider the following measure:



```
_User Defined SPLY =
    CALCULATE(
        [_Total Revenue],
        SAMEPERIODLASTYEAR('Date'[Date])
    )
```

- The result it returns is given:

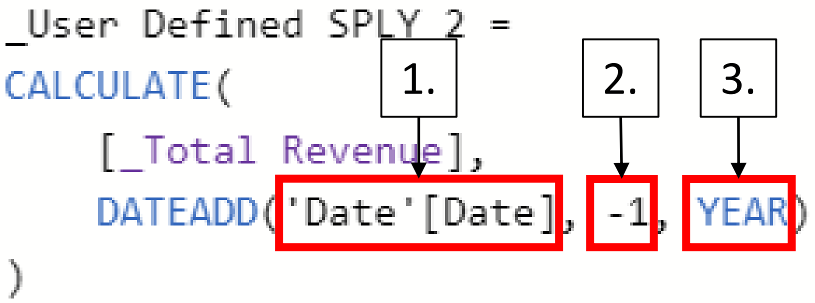


Calendar Year	_Total Revenue	_User Defined SPLY
CY 2007	$11 309 943.45	
CY 2008	$9 927 577.89	R11 309 943.45
CY 2009	$9 353 806.38	R9 927 577.89
CY 2010		R9 353 806.38

- How does SAMEPERIODLASTYEAR() work?

Same period last year (or month)

- SAMEPERIODLASTYEAR() is a different way of writing the following measure, using DATEADD:

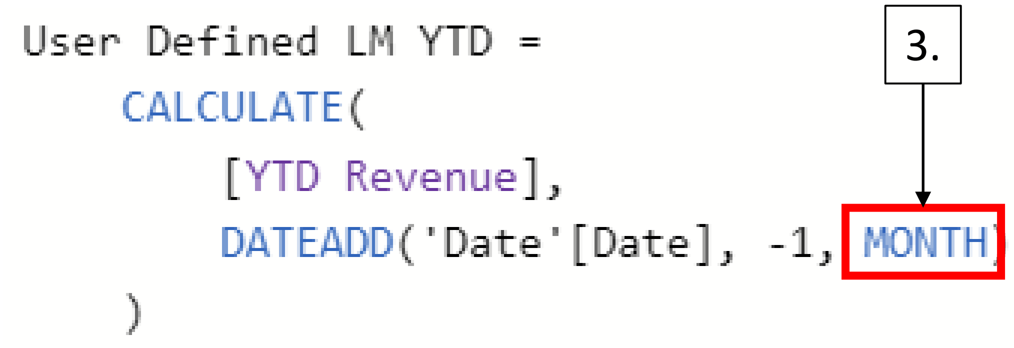


```
_User Defined SPLY 2 =
CALCULATE(
    [_Total Revenue],
    DATEADD('Date'[Date], -1, YEAR)
)
```

1. The Date column from the date table.
2. The number intervals you want to calculate.
3. The granularity.

Same period last year (or month)

- The granularity can set to YEAR, MONTH or DAY

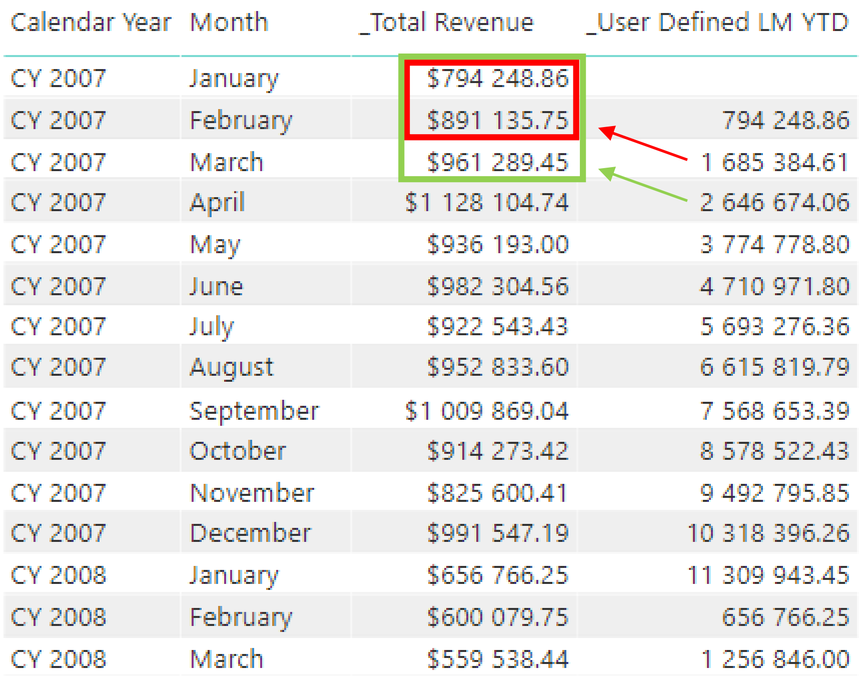


3.

```
User Defined LM YTD =
    CALCULATE(
        [YTD Revenue],
        DATEADD('Date'[Date], -1, MONTH)
    )
```

Let’s see the result on the table

- That works as intended.



Calendar Year	Month	_Total Revenue	_User Defined LM YTD
CY 2007	January	$794 248.86	
CY 2007	February	$891 135.75	794 248.86
CY 2007	March	$961 289.45	1 685 384.61
CY 2007	April	$1 128 104.74	2 646 674.06
CY 2007	May	$936 193.00	3 774 778.80
CY 2007	June	$982 304.56	4 710 971.80
CY 2007	July	$922 543.43	5 693 276.36
CY 2007	August	$952 833.60	6 615 819.79
CY 2007	September	$1 009 869.04	7 568 653.39
CY 2007	October	$914 273.42	8 578 522.43
CY 2007	November	$825 600.41	9 492 795.85
CY 2007	December	$991 547.19	10 318 396.26
CY 2008	January	$656 766.25	11 309 943.45
CY 2008	February	$600 079.75	656 766.25
CY 2008	March	$559 538.44	1 256 846.00

What if the fact table has more than one date column?

- The Sales table has three date columns
  - OrderDate
  - DeliveryDate
  - DueDate

Revenue – Order Date vs Delivery Date

- Customers who have credit with us generally place an order for their goods.

- This is captured on the system as revenue.

- Customers only pay for the goods after delivery of the goods.

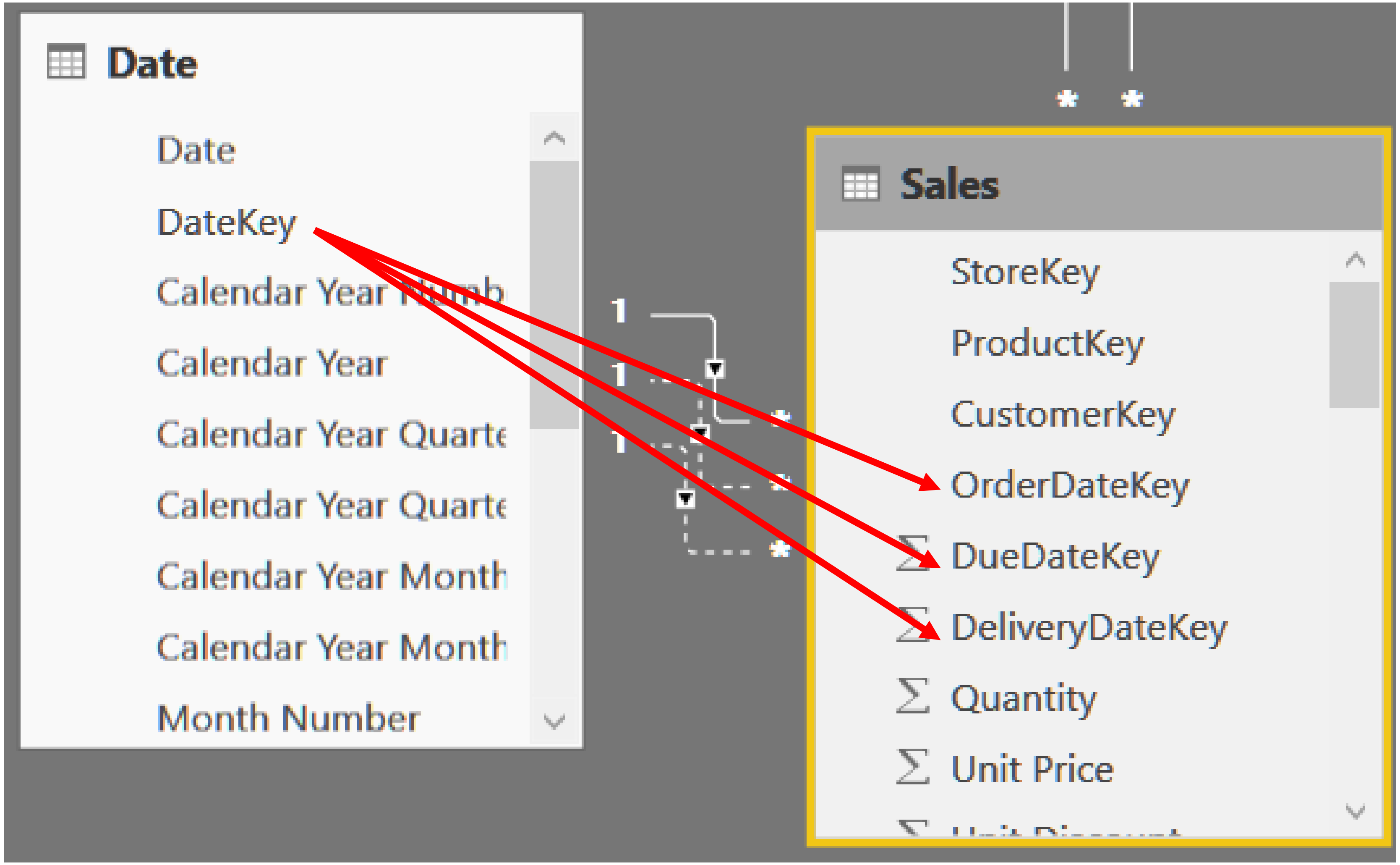
- It is therefore useful to know the amount of revenue for goods ordered, vs revenue for goods delivered.

First steps

- Go to
  File -> Options and Settings -> Options -> Data Load
- Uncheck *Auto Date/Time*

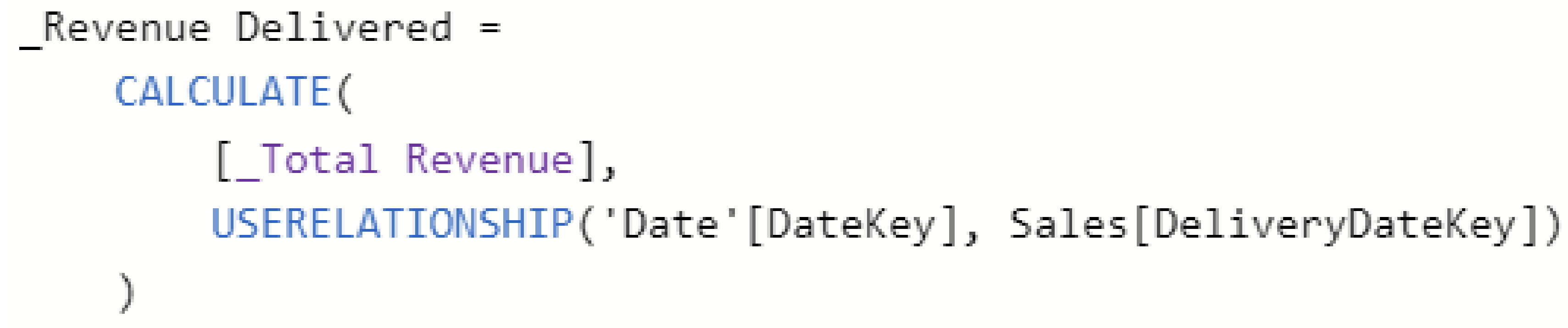
- Go to the relationship viewer
- There should already be a relationship on:
  - Sales[OrderDateKey] -> ‘Date’[DateKey]
- Create new relationships on:
  - Sales[DeliveryDateKey] -> ‘Date’[DateKey]
  - Sales[DueDateKey] -> ‘Date’[DateKey]
- Ignore the fact that the relationships are inactive.

First steps


Date
Date
DateKey
Calendar Year Numb
Calendar Year
Calendar Year Quarte
Calendar Year Quarte
Calendar Year Month
Calendar Year Month
Month Number
1
1
1
*
*
*
*
*
Sales
StoreKey
ProductKey
CustomerKey
OrderDateKey
DueDateKey
DeliveryDateKey
Quantity
Unit Price

Revenue – Order date vs Delivery Date

- Create the following measure:

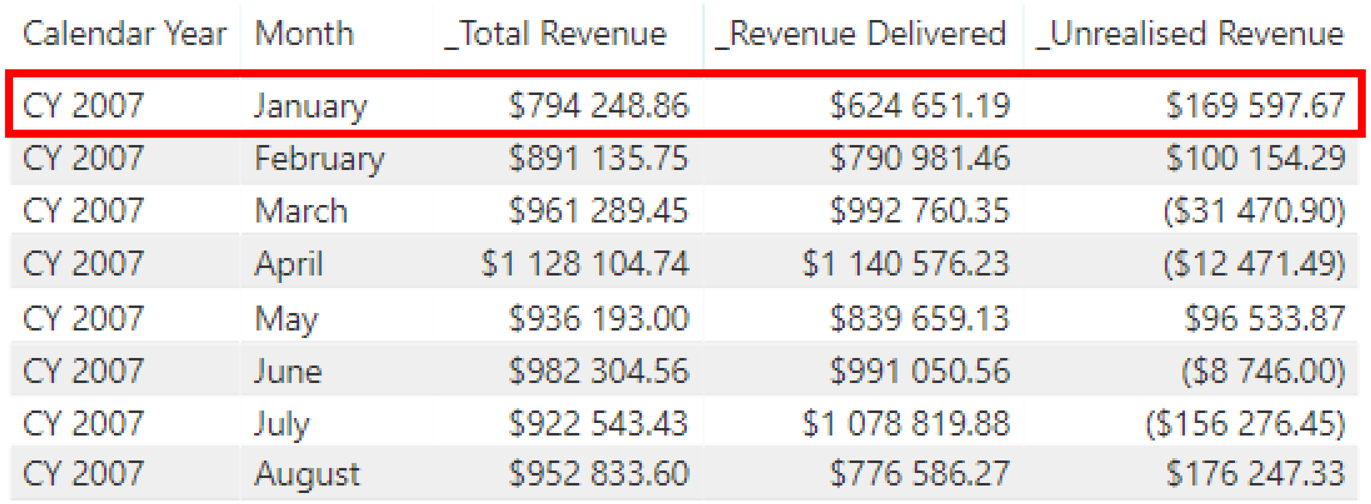


```
_Revenue Delivered =
    CALCULATE(
        [_Total Revenue],
        USERELATIONSHIP('Date'[DateKey], Sales[DeliveryDateKey])
    )
```

- Create a table visual on the report and populate it with:
  - Date[Calendar Year]
  - Date[Month]
  - [_Total Revenue]
  - [_Revenue Delivered]

Revenue – Order date vs Delivery Date

- This visual shows, for example:
  - $794,248.86 was ordered in January 2007,
  - $624 651.19 was delivered in January 2007
  - The unrealised revenue for January 2007 is $169 597.67



Calendar Year	Month	_Total Revenue	_Revenue Delivered	_Unrealised Revenue
CY 2007	January	$794 248.86	$624 651.19	$169 597.67
CY 2007	February	$891 135.75	$790 981.46	$100 154.29
CY 2007	March	$961 289.45	$992 760.35	($31 470.90)
CY 2007	April	$1 128 104.74	$1 140 576.23	($12 471.49)
CY 2007	May	$936 193.00	$839 659.13	$96 533.87
CY 2007	June	$982 304.56	$991 050.56	($8 746.00)
CY 2007	July	$922 543.43	$1 078 819.88	($156 276.45)
CY 2007	August	$952 833.60	$776 586.27	$176 247.33

Time Intelligence – Customer Growth

- The Executive Committee would like to see their client growth trends.
- The Customer table contains a Date First Purchase column.
- Getting the count of these dates, will give the client count.
- Displaying this year-on-year will give the client growth.

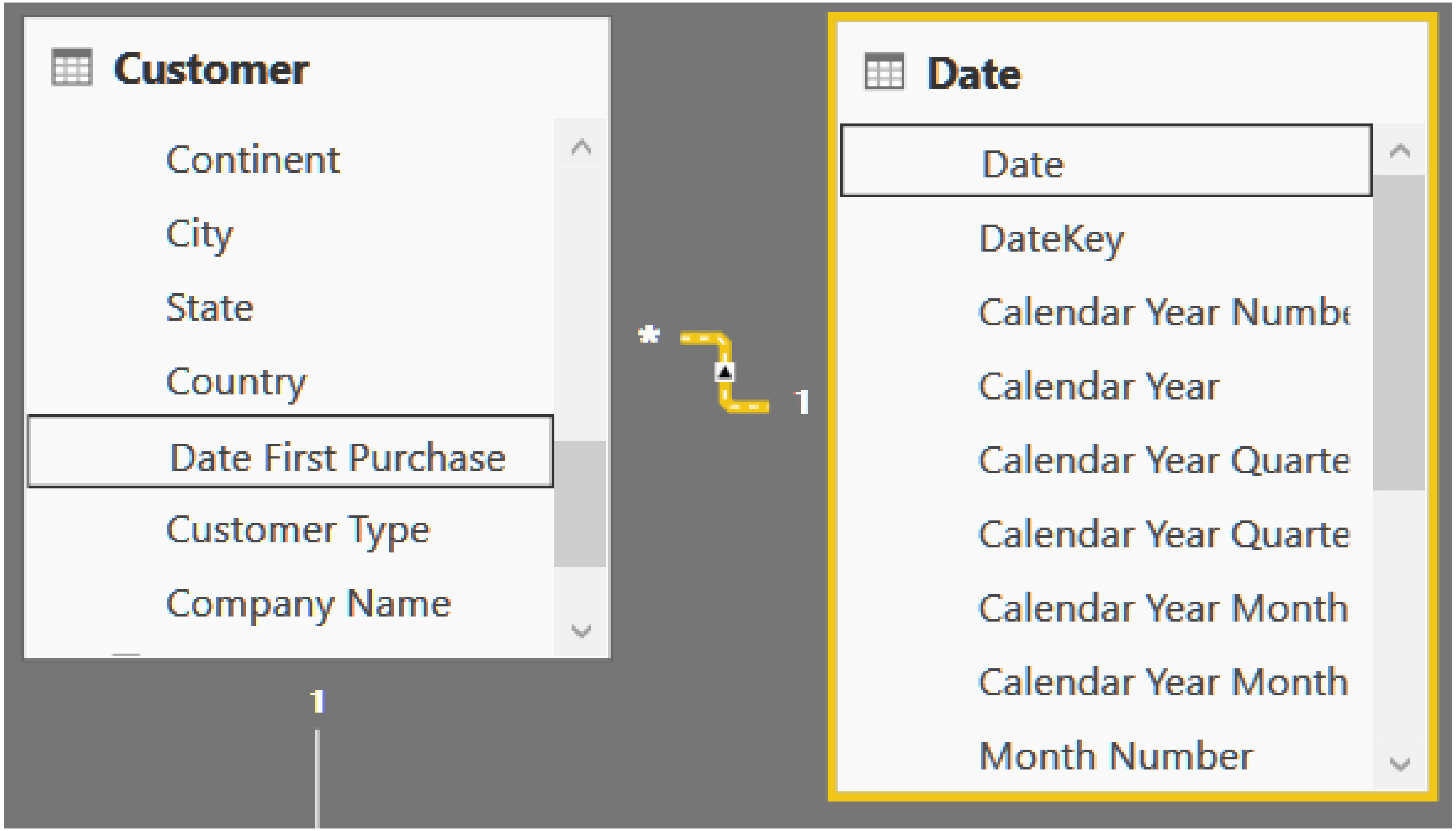
Time Intelligence – Customer Growth

- Create a relationship on:
  - Customer[Date First Purchase] -> Date[Date]

- As there is already a relationship from:
  - Customer -> Sales -> Date
  - The one just created is inactive.

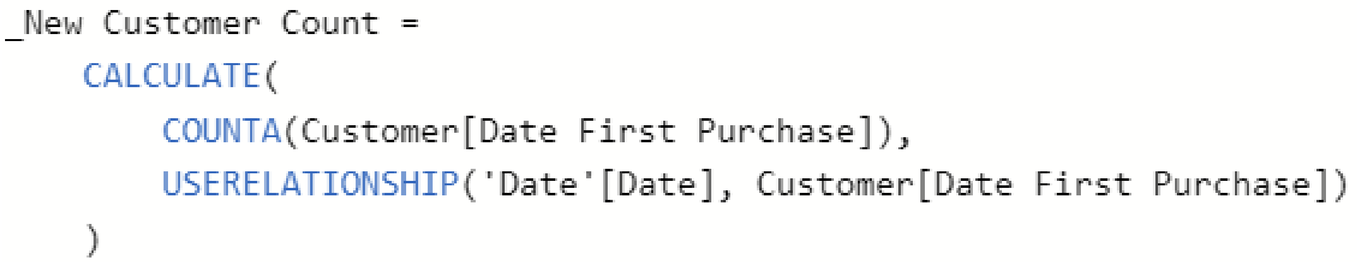
Time Intelligence – Customer Growth

ALTRON | KARABINA



Customer Growth

- Create the following measure in the Customer table:

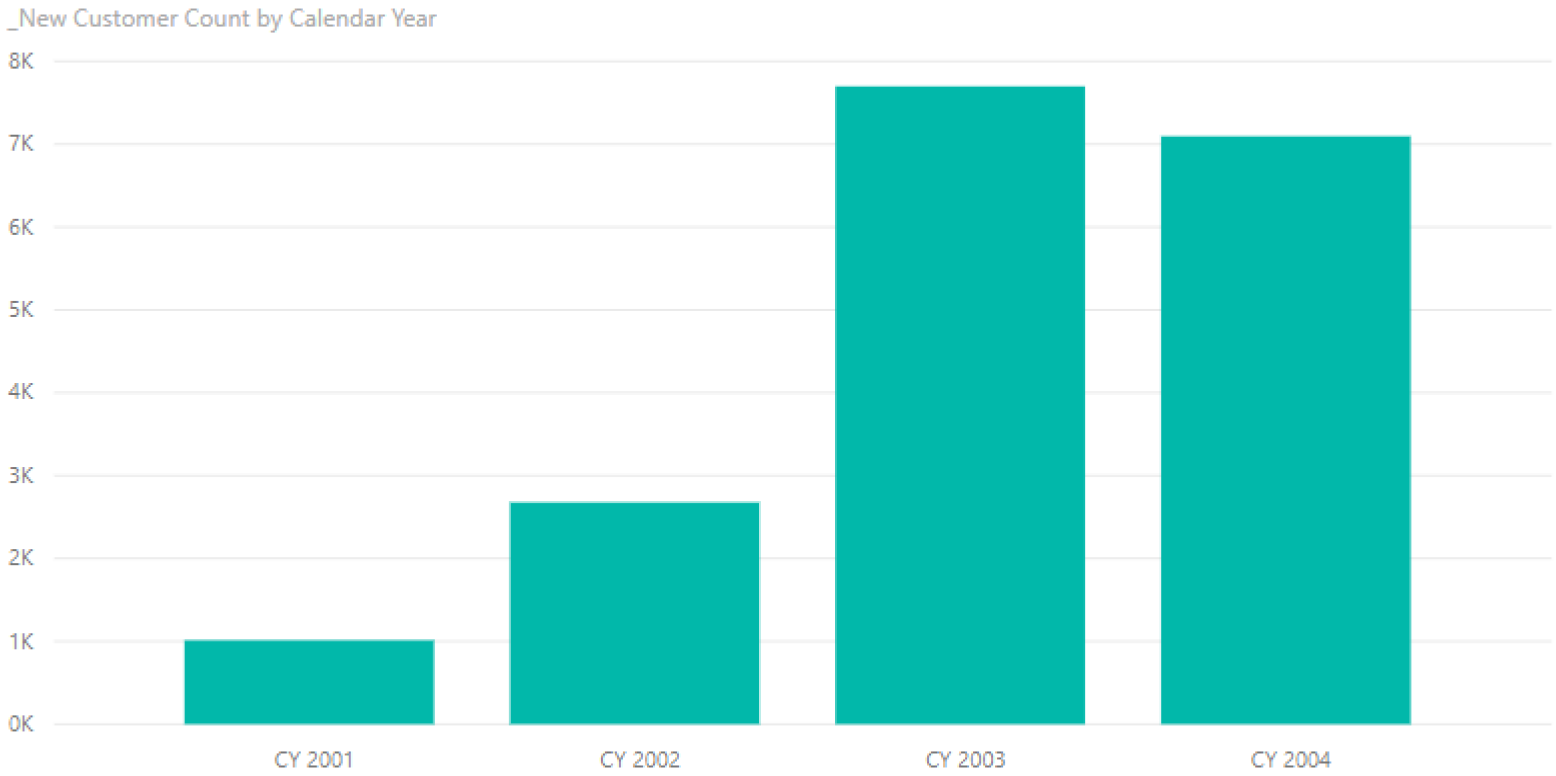


```
_New Customer Count =
    CALCULATE(
        COUNTA(Customer[Date First Purchase]),
        USERELATIONSHIP('Date'[Date], Customer[Date First Purchase])
    )
```

- This creates a measure showing the number of new customers gained each year.

Customer Growth

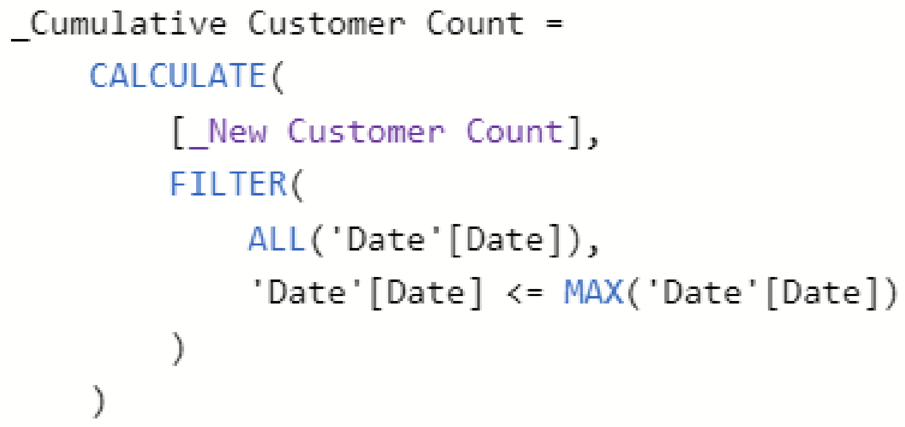
- Place this measure in a column chart with Calendar Year on the Axis.



- The graph shows that there was a drop off in new customers in 2004.

Customer Growth – Cumulative Sum

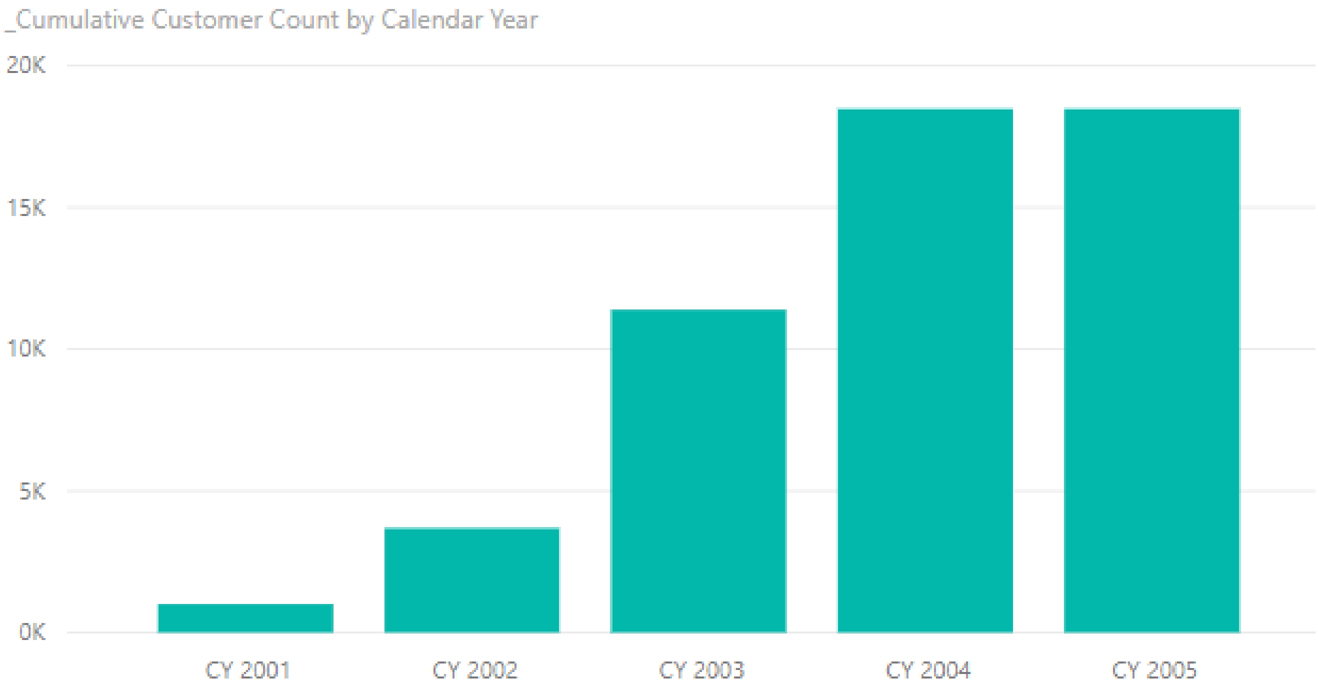
- To get a count of all the customers, use the Cumulative Sum we saw earlier.
- Create the following measure in the customer table.



```
_Cumulative Customer Count =
    CALCULATE(
        [_New Customer Count],
        FILTER(
            ALL('Date'[Date]),
            'Date'[Date] <= MAX('Date'[Date])
        )
    )
```

Customer Growth – Cumulative Sum

- Place this measure on a similar visual to get an idea of the cumulative client growth:



- The graph shows no client growth since 2004

Question and answer

ALTRON | KARABINA

0

Dashboard Design

Session 3

- Power Query Editor
    - Grouping
    - Column from examples
    - Conditional column
    - Custom column
- Creating a Date Table in DAX
- Dashboard design rules
- Types of data
- Choose the right visual for the job
- Colour schemes
- Page backgrounds
- Query parameters
- Incremental refresh
- Grouping data in the Data Model using SUMMARIZE()
- What if – Analysis
- Buttons and bookmarks
- Tooltip pages
- QnA, Clustering, Forecasting
- Dynamic Security
- Power BI Data Management Gateway
- Tabular model building

What is a dashboard?

“A dashboard is a visual display of the most important information needed to achieve one or more objectives, consolidated and arranged in a single screen, so the information can be monitored at a glance.”

- Stephen Few

Bad dashboard design

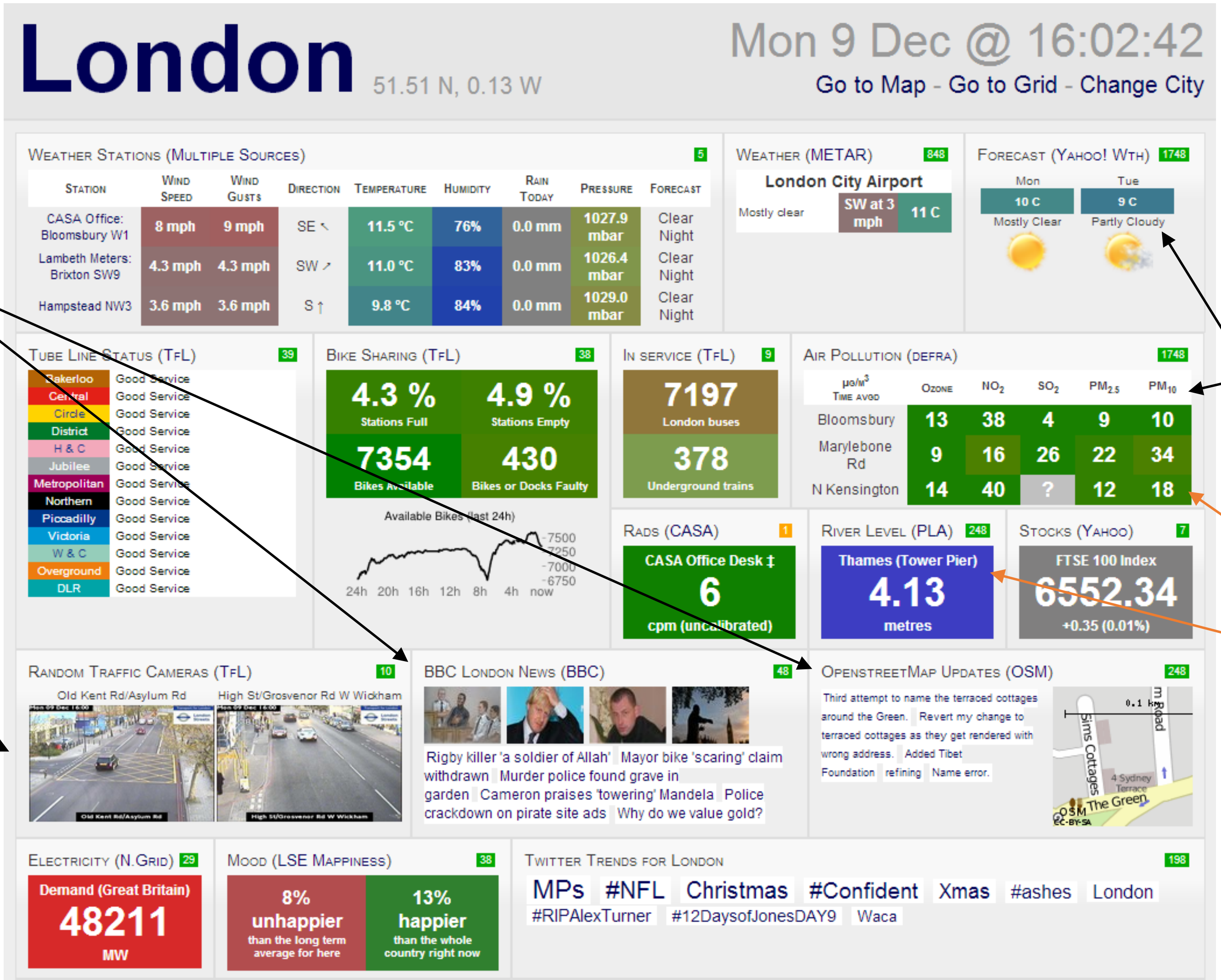
ALTRON | KARABINA

Incoherent

No logical groupings

Too much text

Useless pictures



London 51.51 N, 0.13 W

Mon 9 Dec @ 16:02:42

Go to Map - Go to Grid - Change City

WEATHER STATIONS (MULTIPLE SOURCES) 5

STATION	WIND SPEED	WIND GUSTS	DIRECTION	TEMPERATURE	HUMIDITY	RAIN TODAY	PRESSURE	FORECAST
CASA Office: Bloomsbury W1	8 mph	9 mph	SE ↖	11.5 °C	76%	0.0 mm	1027.9 mbar	Clear Night
Lambeth Meters: Brixton SW9	4.3 mph	4.3 mph	SW ↗	11.0 °C	83%	0.0 mm	1026.4 mbar	Clear Night
Hampstead NW3	3.6 mph	3.6 mph	S ↑	9.8 °C	84%	0.0 mm	1029.0 mbar	Clear Night

WEATHER (METAR) 848

London City Airport

Mostly clear | SW at 3 mph | 11 C

FORECAST (YAHOO! WTH) 1748

Mon	Tue
10 C	9 C
Mostly Clear	Partly Cloudy

TUBE LINE STATUS (TFL) 39

Bakerloo	Good Service
Central	Good Service
Circle	Good Service
District	Good Service
H & C	Good Service
Jubilee	Good Service
Metropolitan	Good Service
Northern	Good Service
Piccadilly	Good Service
Victoria	Good Service
W & C	Good Service
Overground	Good Service
DLR	Good Service

BIKE SHARING (TFL) 38

4.3 % Stations Full

4.9 % Stations Empty

7354 Bikes Available

430 Bikes or Docks Faulty

Available Bikes (last 24h)

IN SERVICE (TFL) 9

7197 London buses

378 Underground trains

AIR POLLUTION (DEFRA) 1748

µg/m³ TIME AVGD	OZONE	NO_2	SO_2	$PM_{2.5}$	PM_{10}
Bloomsbury	13	38	4	9	10
Marylebone Rd	9	16	26	22	34
N Kensington	14	40	?	12	18

RADS (CASA) 1

CASA Office Desk ‡

6

cpm (uncalibrated)

RIVER LEVEL (PLA) 248

Thames (Tower Pier)

4.13

metres

STOCKS (YAHOO) 7

FTSE 100 Index

6552.34

+0.35 (0.01%)

RANDOM TRAFFIC CAMERAS (TFL) 10

Old Kent Rd/Asylum Rd

High St/Grosvenor Rd W Wickham

BBC LONDON NEWS (BBC) 48

Rigby killer 'a soldier of Allah' Mayor bike 'scaring' claim withdrawn Murder police found grave in garden Cameron praises 'towering' Mandela Police crackdown on pirate site ads Why do we value gold?

OPENSTREETMAP UPDATES (OSM) 248

Third attempt to name the terraced cottages around the Green. Revert my change to terraced cottages as they get rendered with wrong address. Added Tibet Foundation refining Name error.

ELECTRICITY (N.GRID) 29

Demand (Great Britain)

48211

MW

MOOD (LSE MAPPINESS) 38

8% unhappier than the long term average for here

13% happier than the whole country right now

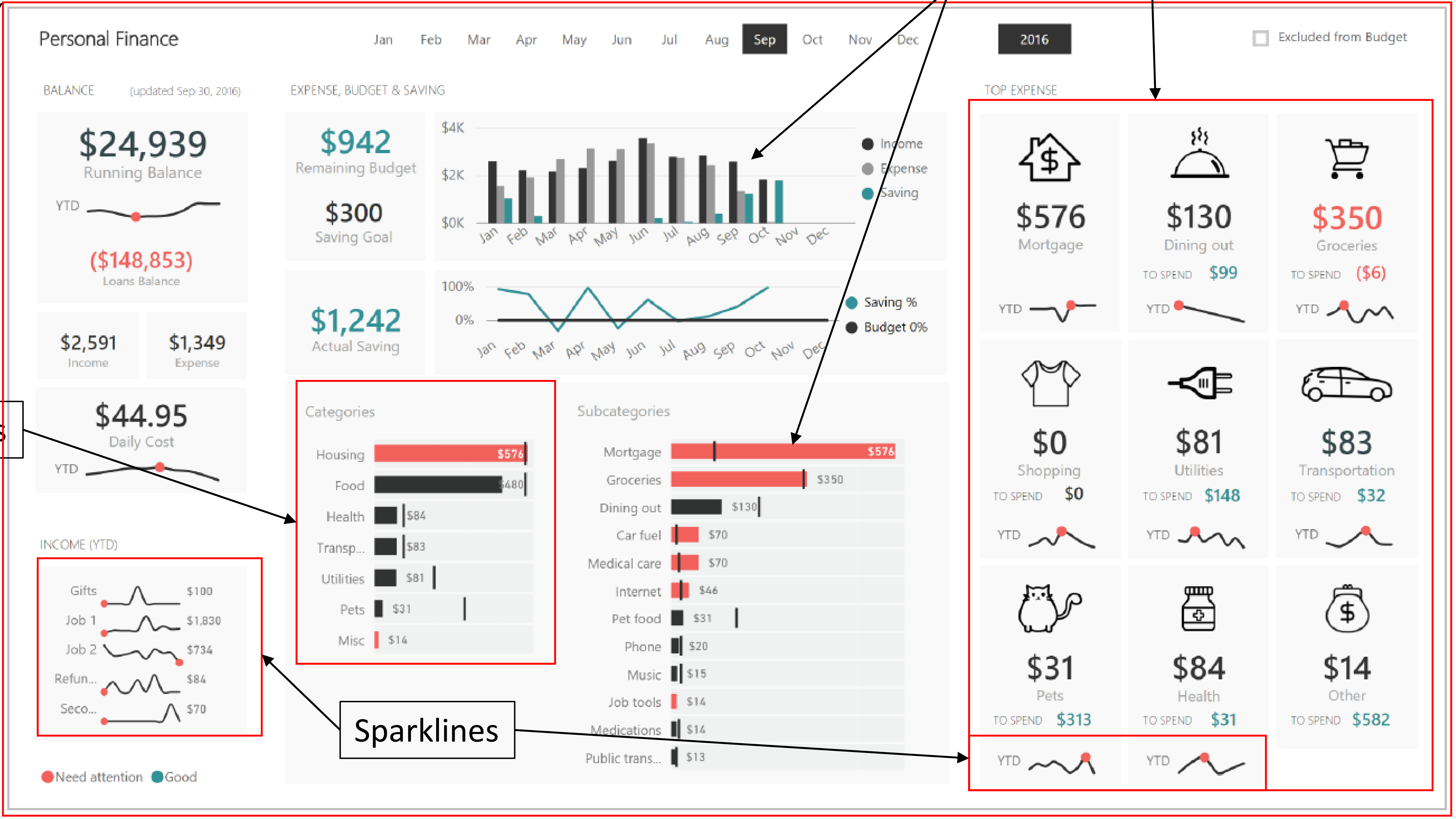
TWITTER TRENDS FOR LONDON 198

MPs #NFL Christmas #Confident Xmas #ashes London #RIPAlexTurner #12DaysofJonesDAY9 Waca

Font too small

Overwhelming colours

Good dashboard design



Clarity

Colour scheme

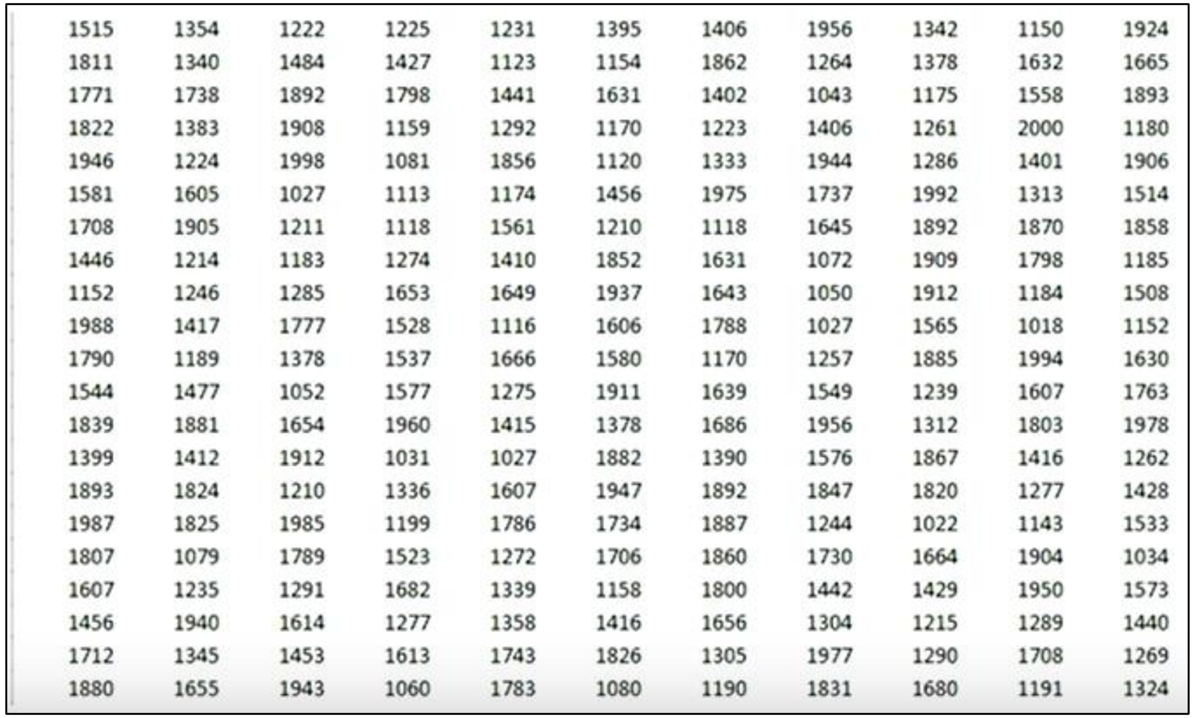
Logical grouping

Bullet charts

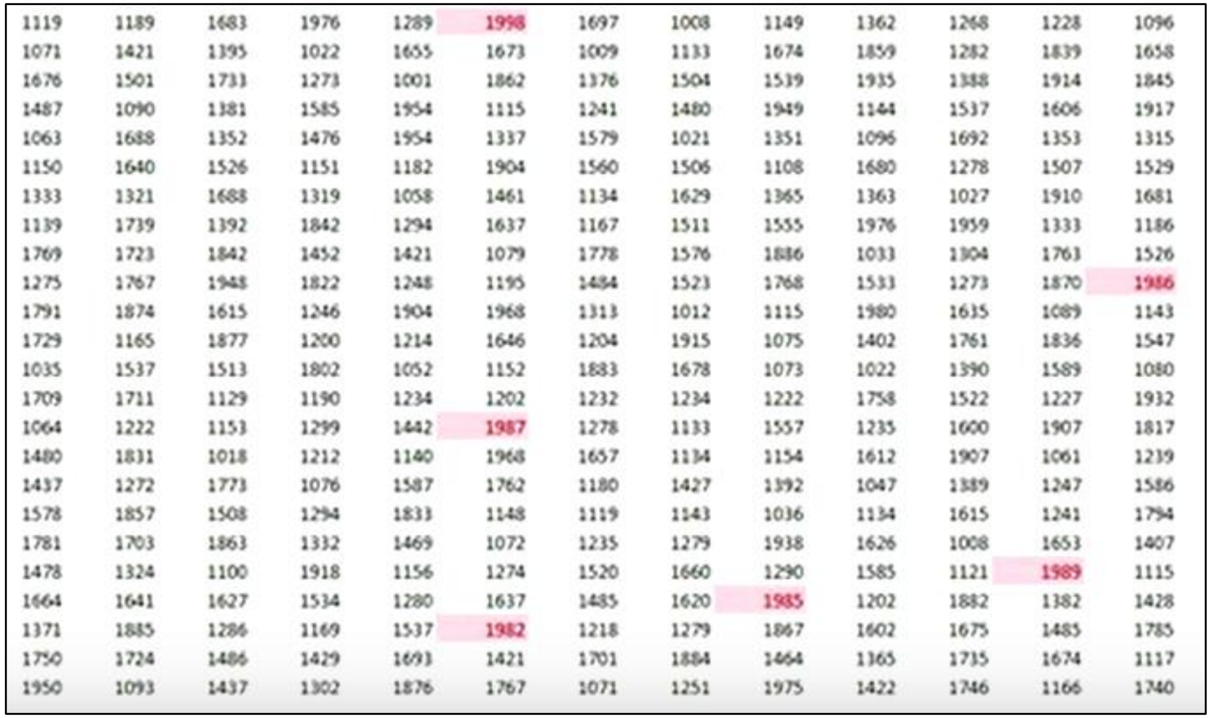
Sparklines

ALTRON | KARABINA

Importance of visualisations



1515	1354	1222	1225	1231	1395	1406	1956	1342	1150	1924
1811	1340	1484	1427	1123	1154	1862	1264	1378	1632	1665
1771	1738	1892	1798	1441	1631	1402	1043	1175	1558	1893
1822	1383	1908	1159	1292	1170	1223	1406	1261	2000	1180
1946	1224	1998	1081	1856	1120	1333	1944	1286	1401	1906
1581	1605	1027	1113	1174	1456	1975	1737	1992	1313	1514
1708	1905	1211	1118	1561	1210	1118	1645	1892	1870	1858
1446	1214	1183	1274	1410	1852	1631	1072	1909	1798	1185
1152	1246	1285	1653	1649	1937	1643	1050	1912	1184	1508
1988	1417	1777	1528	1116	1606	1788	1027	1565	1018	1152
1790	1189	1378	1537	1666	1580	1170	1257	1885	1994	1630
1544	1477	1052	1577	1275	1911	1639	1549	1239	1607	1763
1839	1881	1654	1960	1415	1378	1686	1956	1312	1803	1978
1399	1412	1912	1031	1027	1882	1390	1576	1867	1416	1262
1893	1824	1210	1336	1607	1947	1892	1847	1820	1277	1428
1987	1825	1985	1199	1786	1734	1887	1244	1022	1143	1533
1807	1079	1789	1523	1272	1706	1860	1730	1664	1904	1034
1607	1235	1291	1682	1339	1158	1800	1442	1429	1950	1573
1456	1940	1614	1277	1358	1416	1656	1304	1215	1289	1440
1712	1345	1453	1613	1743	1826	1305	1977	1290	1708	1269
1880	1655	1943	1060	1783	1080	1190	1831	1680	1191	1324

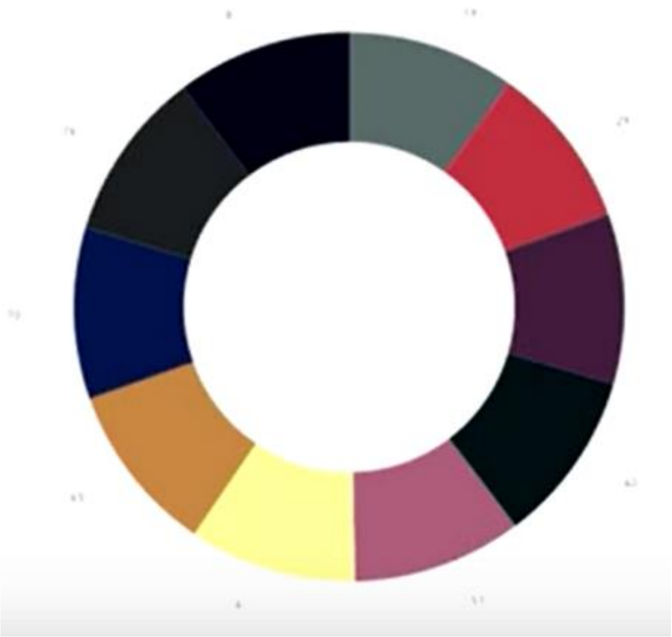


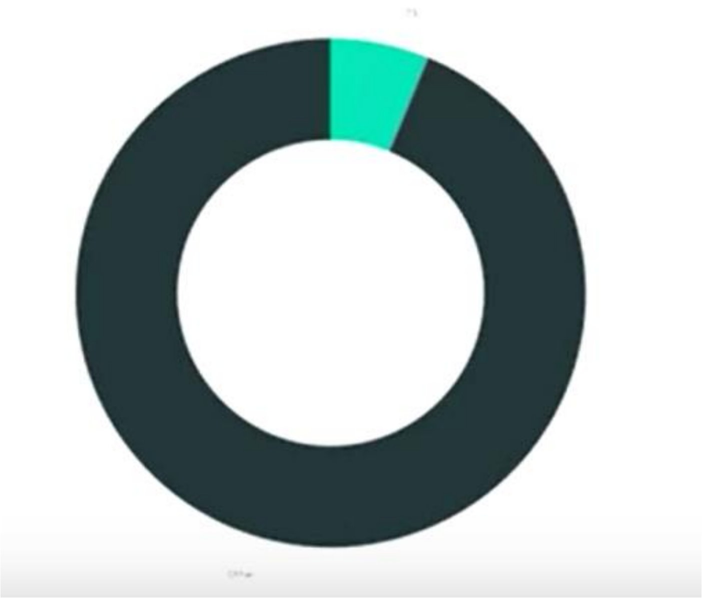
1119	1189	1683	1976	1289	1998	1697	1008	1149	1362	1268	1228	1096
1071	1421	1395	1022	1655	1673	1009	1133	1674	1859	1282	1839	1658
1676	1501	1731	1273	1001	1862	1376	1504	1539	1935	1388	1914	1845
1487	1090	1381	1585	1954	1115	1241	1480	1949	1144	1537	1606	1917
1063	1688	1352	1476	1954	1337	1579	1021	1351	1096	1692	1353	1315
1150	1640	1526	1151	1182	1904	1560	1506	1108	1680	1278	1507	1529
1333	1321	1688	1319	1058	1461	1134	1629	1365	1363	1027	1910	1681
1139	1739	1392	1842	1294	1637	1167	1511	1555	1976	1959	1333	1186
1769	1723	1842	1452	1421	1079	1778	1576	1886	1033	1304	1763	1526
1275	1767	1948	1822	1248	1195	1484	1523	1768	1533	1273	1870	1986
1791	1874	1615	1246	1904	1968	1313	1012	1115	1980	1635	1089	1143
1729	1165	1877	1200	1214	1646	1204	1915	1075	1402	1761	1836	1547
1035	1537	1513	1802	1052	1152	1883	1678	1073	1022	1390	1589	1080
1709	1711	1129	1190	1234	1202	1232	1234	1222	1758	1522	1227	1932
1064	1222	1153	1299	1442	1987	1278	1133	1557	1235	1600	1907	1817
1480	1831	1018	1212	1140	1968	1657	1134	1154	1612	1907	1061	1239
1437	1272	1773	1076	1587	1762	1180	1427	1392	1047	1389	1247	1586
1578	1857	1508	1294	1833	1148	1119	1143	1036	1134	1615	1241	1794
1781	1703	1863	1332	1469	1072	1235	1279	1938	1626	1008	1653	1407
1478	1324	1100	1918	1156	1274	1520	1660	1290	1585	1121	1989	1115
1664	1641	1627	1534	1280	1637	1485	1620	1985	1202	1882	1382	1428
1371	1885	1286	1169	1537	1982	1218	1279	1867	1602	1675	1485	1785
1750	1724	1486	1429	1693	1421	1701	1884	1464	1365	1735	1674	1117
1950	1093	1437	1302	1876	1767	1071	1251	1975	1422	1746	1166	1740

- Find the largest number in the above table…

- Adding colour coding makes it much easier to spot insights in the data; your attention is drawn to the highlighted values.

Importance of visualisations





- The type of visual is important. The pie chart on the left is useless; it does not show any insight as all the portions look similar.
- The one of the right is slightly better, but it can still be improved.

Rules for pie charts

Remove

to improve

the **pie chart** edition

Created by Darkhorse Analytics

www.**darkhorseanalytics**.com

Rules to

Designing the perfect dashboard

ALTRON | KARABINA

01

Design for a purpose

02

Keep everything at a glance

03

Maintain simplicity

04

Keep elements aligned

05

Consistency is key

06

Highlight appropriate information

07

Be clear

08

Begin from zero

09

100K

Reduce length of numbers

10

Display the context

11

Select appropriate colours

12

Design dashboards, not reports

13

Show variations

14

Don’t add noise

15

Choose the correct charts

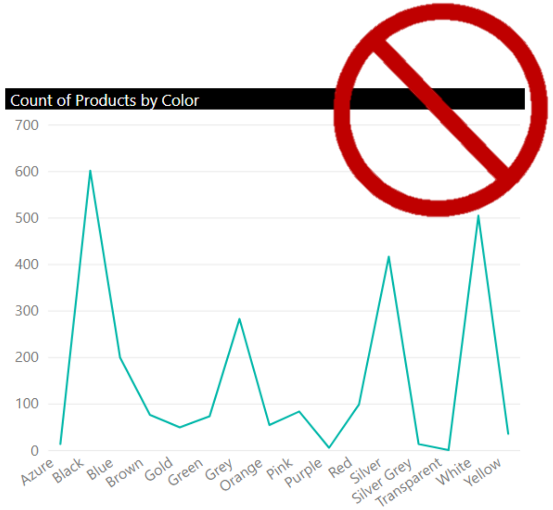
Types of data

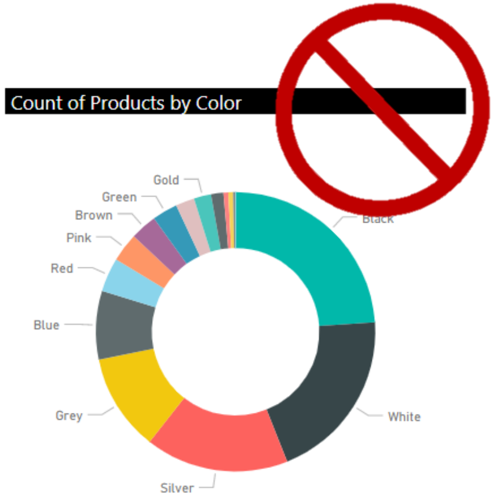
- There are two types:
  - Categorical
    - Nominal – Contains no inherent ordering, example, Red, Green, Blue
    - Ordinal – Contains an inherent ordering, example, Small, Medium, Large.
  - Numeric
    - Generally a continuous quantity.

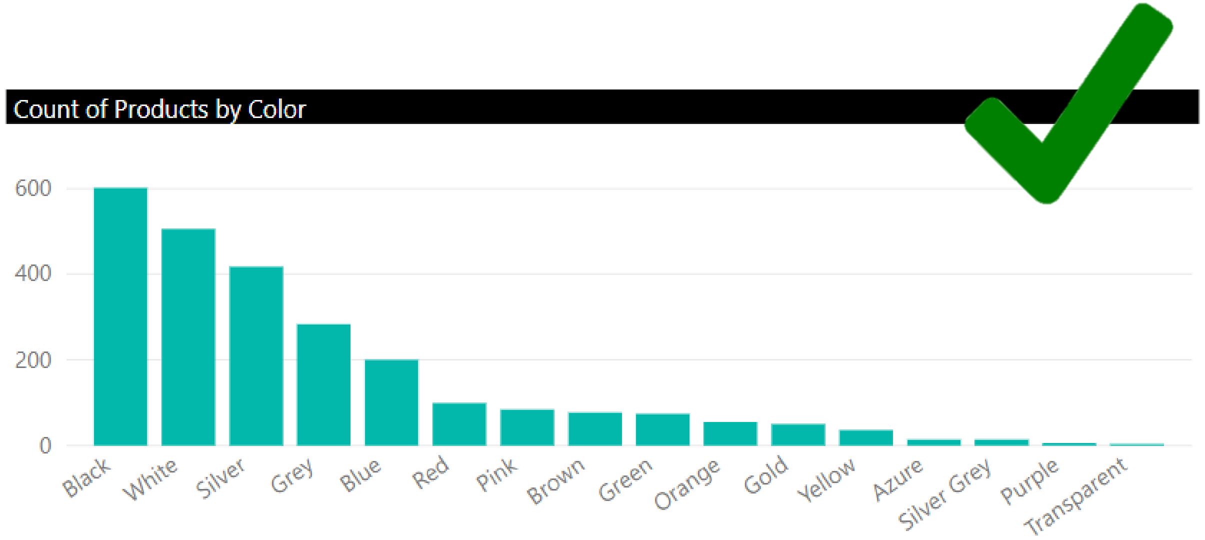
- The different types of data generally require different types of visualisations to effectively communicate their meaning.

Incorrect vs Correct visuals

- Don’t use a line chart to display categorical data. The shape of the line is meant to encode some meaning.
- Pie and doughnut charts also do not display meaning effectively.
- It is better to use a bar chart to display categorical data.







sqlbi. OKViz

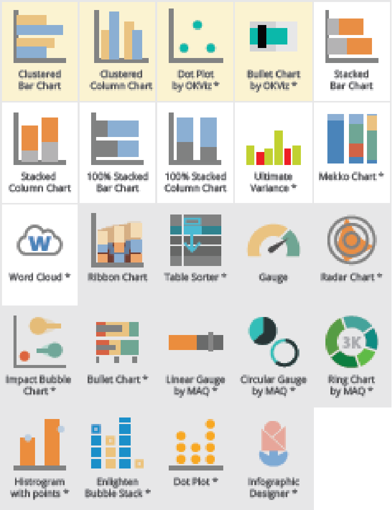
THE VISUALS REFERENCE

FOR POWER BI - SEP 2017

http://sql.bi/visual-reference

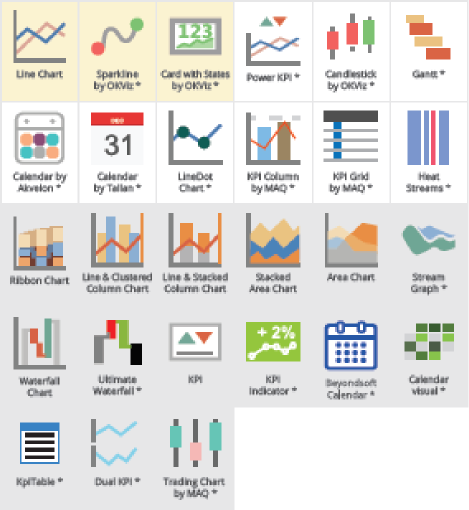
COMPARISON

Use these visuals when you want to display measures compared by its magnitude.



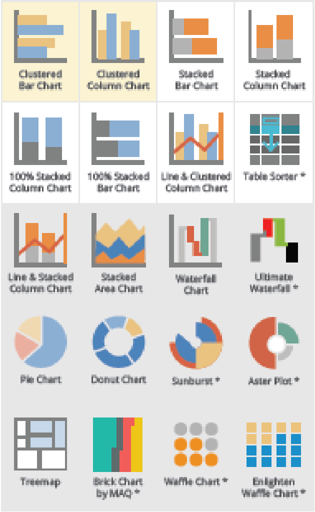
CHANGE OVER TIME

Use these visuals when you want to display the changing trend of measures.



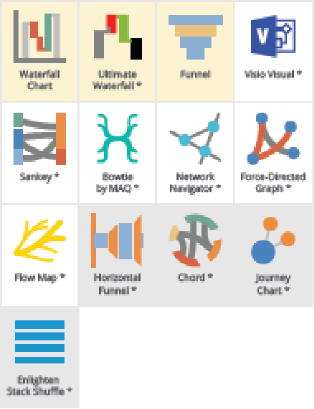
PART-TO-WHOLE

Use these visuals when you want to display parts that compose measures.



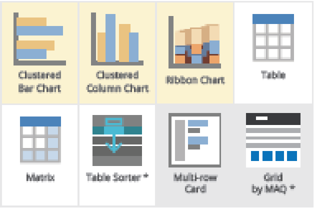
FLOW

Use these visuals when you want to display a flow or dynamic relations.



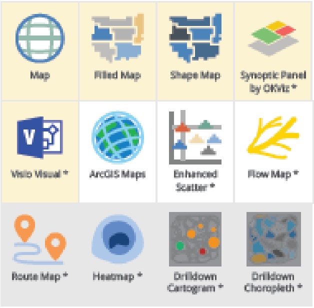
RANKING

Use these visuals when you want to display measures by its rank order.



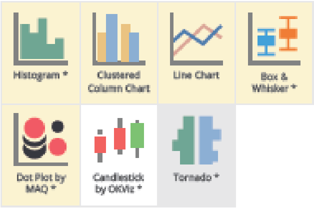
SPATIAL

Use these visuals when you want to display measures over spatial maps.



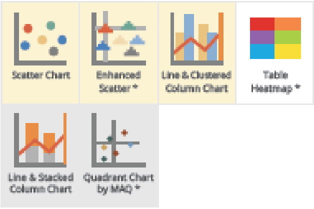
DISTRIBUTION

Use these visuals when you want to display the distribution of a measure.



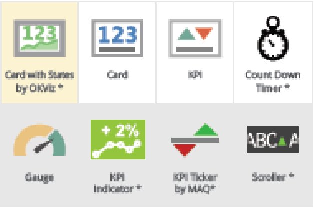
CORRELATION

Use these visuals when you want to display relations between measures.



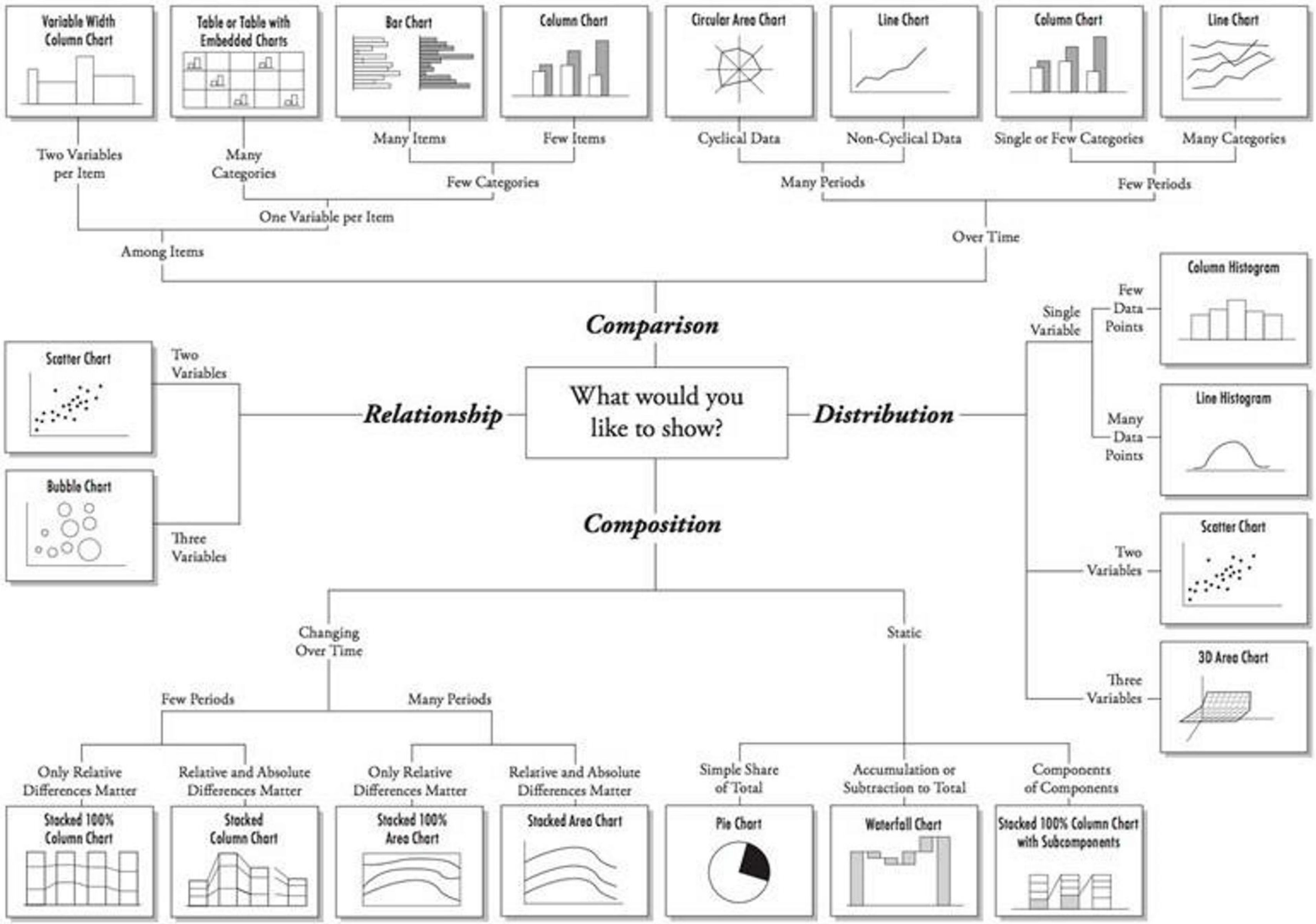
SINGLE

Use these visuals when you want to display a single value.



🟨 Preferred for Dashboards ⬜ Not Recommended (there is a Better Alternative or it is just a Clone) ★ Custom Visual

Chart Suggestions—A Thought-Starter



What would you like to show?

Comparison

- Among Items
  - Two Variables per Item
    - Variable Width Column Chart
  - One Variable per Item
    - Many Categories
      - Table or Table with Embedded Charts
    - Few Categories
      - Many Items
        - Bar Chart
      - Few Items
        - Column Chart
- Over Time
  - Many Periods
    - Cyclical Data
      - Circular Area Chart
    - Non-Cyclical Data
      - Line Chart
  - Few Periods
    - Single or Few Categories
      - Column Chart
    - Many Categories
      - Line Chart

Relationship

- Two Variables
  - Scatter Chart
- Three Variables
  - Bubble Chart

Distribution

- Single Variable
  - Few Data Points
    - Column Histogram
  - Many Data Points
    - Line Histogram
- Two Variables
  - Scatter Chart
- Three Variables
  - 3D Area Chart

Composition

- Changing Over Time
  - Few Periods
    - Only Relative Differences Matter
      - Stacked 100% Column Chart
    - Relative and Absolute Differences Matter
      - Stacked Column Chart
  - Many Periods
    - Only Relative Differences Matter
      - Stacked 100% Area Chart
    - Relative and Absolute Differences Matter
      - Stacked Area Chart
- Static
  - Simple Share of Total
    - Pie Chart
  - Accumulation or Subtraction to Total
    - Waterfall Chart
  - Components of Components
    - Stacked 100% Column Chart with Subcomponents

Colour Selection

- When setting backgrounds for report pages, choose colors that don’t overshadow the report, clash with other colors on the page, or generally hurt the eyes.
- Realize that some colors have inherent meaning. For example, red is typically interpreted as “bad”.
- One of the worst distractions in graphs involves the misuse of color. A jumble of bright colors can visually overwhelm the reader.
- Colors that are different for no reason, such as a different color per bar in a simple bar graph that contains a single set of values tempts our brains to search for a meaning for the differences.
- Best practice: Use relatively soft colors in graphs, such as lowly saturated, natural colors found in nature.
- Reserving the use of bright, dark, and highly saturated colors to make something stand out.

Colour Selection

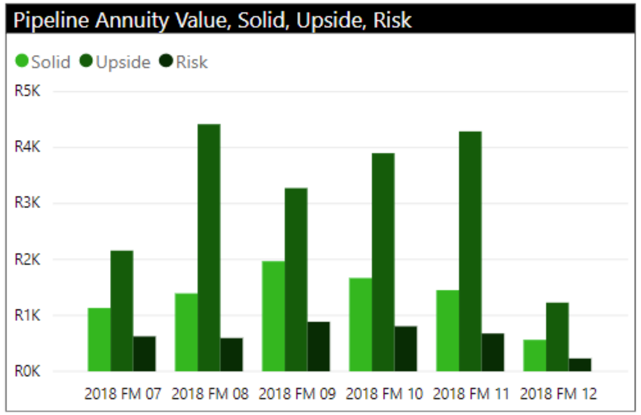
- The use of colour in the dashboard serves a purpose. Design the dashboard colour scheme to accommodate colour blindness.
- When creating the dashboard, try avoid using completely different colours in the colour palette. Try stick to different shades of the same colour.
- Even if a person cannot distinguish the colour green, for example, they are still able to distinguish different shades of the same colour.
- Use a colour blindness simulator to test your colour palette for different types of colour blindness.

http://www.color-blindness.com/coblis-color-blindness-simulator/

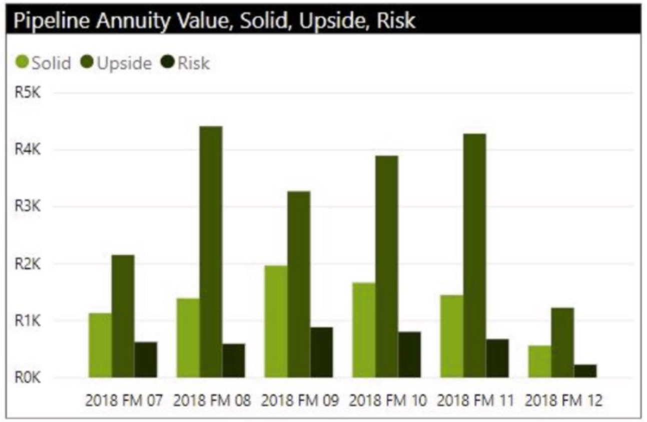
Colour Selection

ALTRON | KARABINA

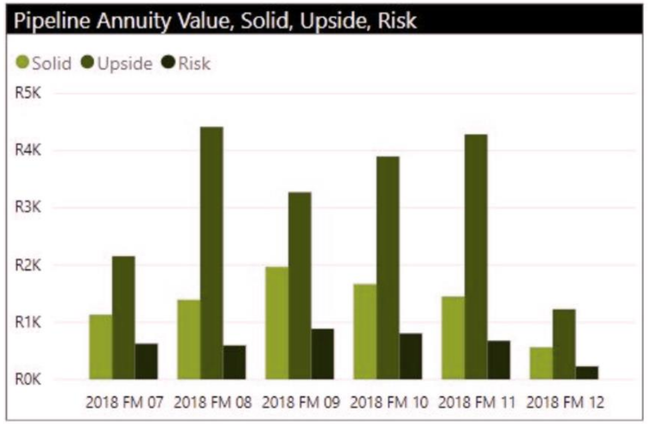
Normal vision



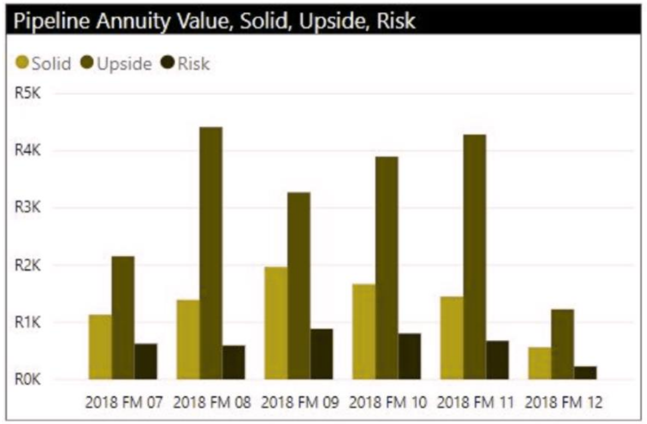
Red-weak



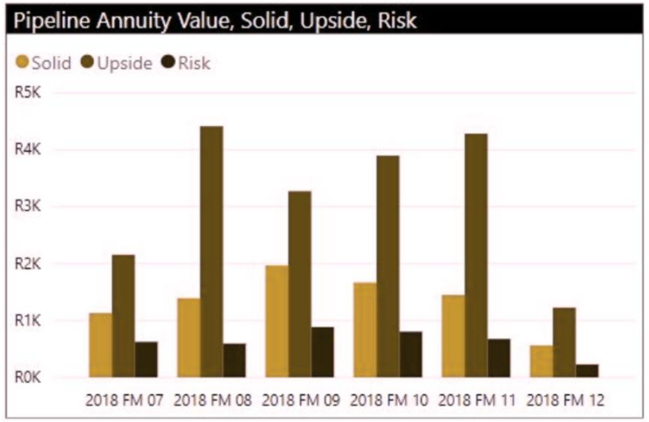
Green-weak



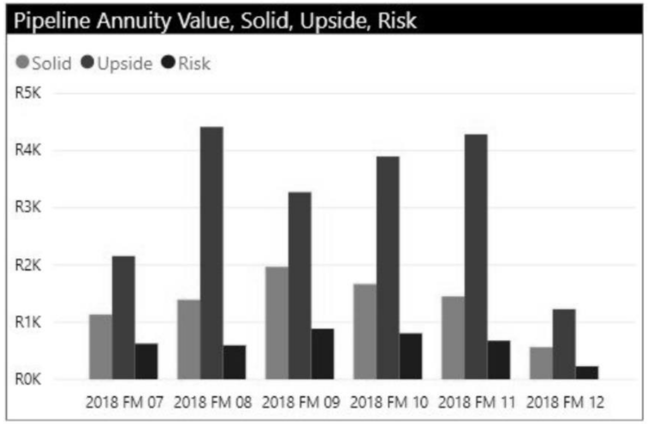
Red-Blind



Green-blind

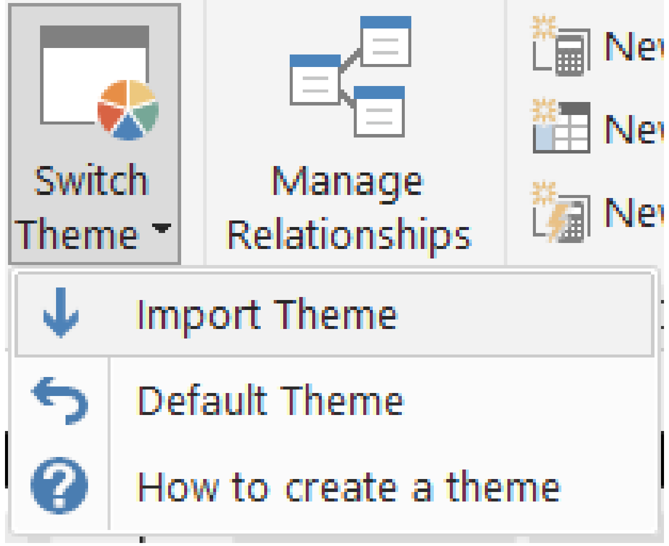


Monochromatic



Colour Selection

- Use a theme generator to create colour schemes.
- https://powerbi.tips/tools/report-theme-generator-v3/
- Download the theme as a .json file.
- Ensure Custom Themes preview feature is enabled.



The Switch Theme button is available under the Home tab

Import the theme.

Query Parameters

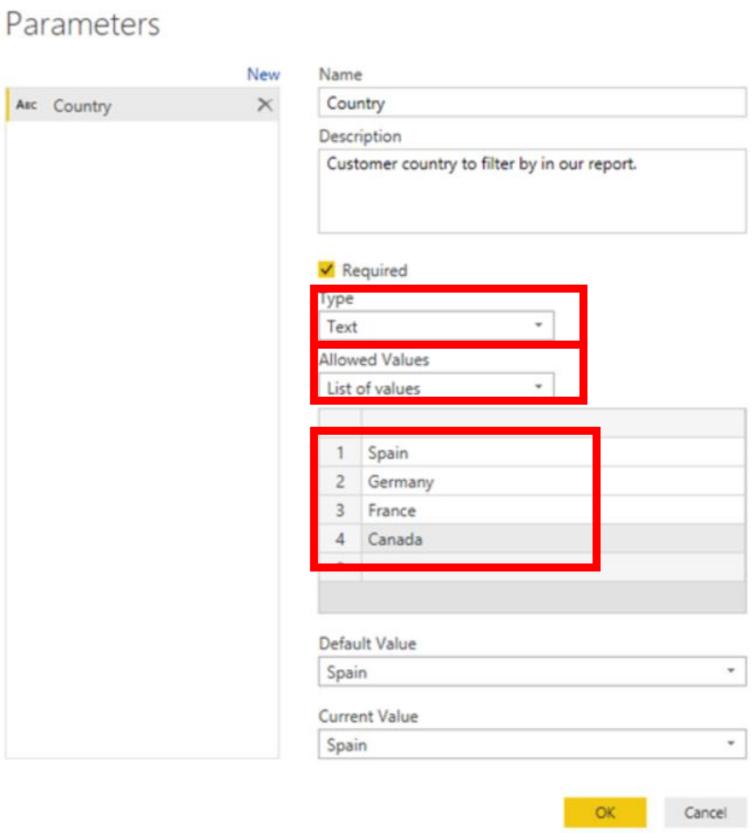
- Parameters allow for the loading of only certain data into a report.

- For example, the report creator could set a parameter on the Country column.

- When a user opens the report, they can select a certain country, and only the data for that country is imported.

Query Parameters

- Users can define new parameters by using the "Manage Parameters" dialog in the Query Editor window



Parameters

New

Country

Name: Country

Description: Customer country to filter by in our report.

☑ Required

Type: Text

Allowed Values: List of values

1	Spain
2	Germany
3	France
4	Canada

Default Value: Spain

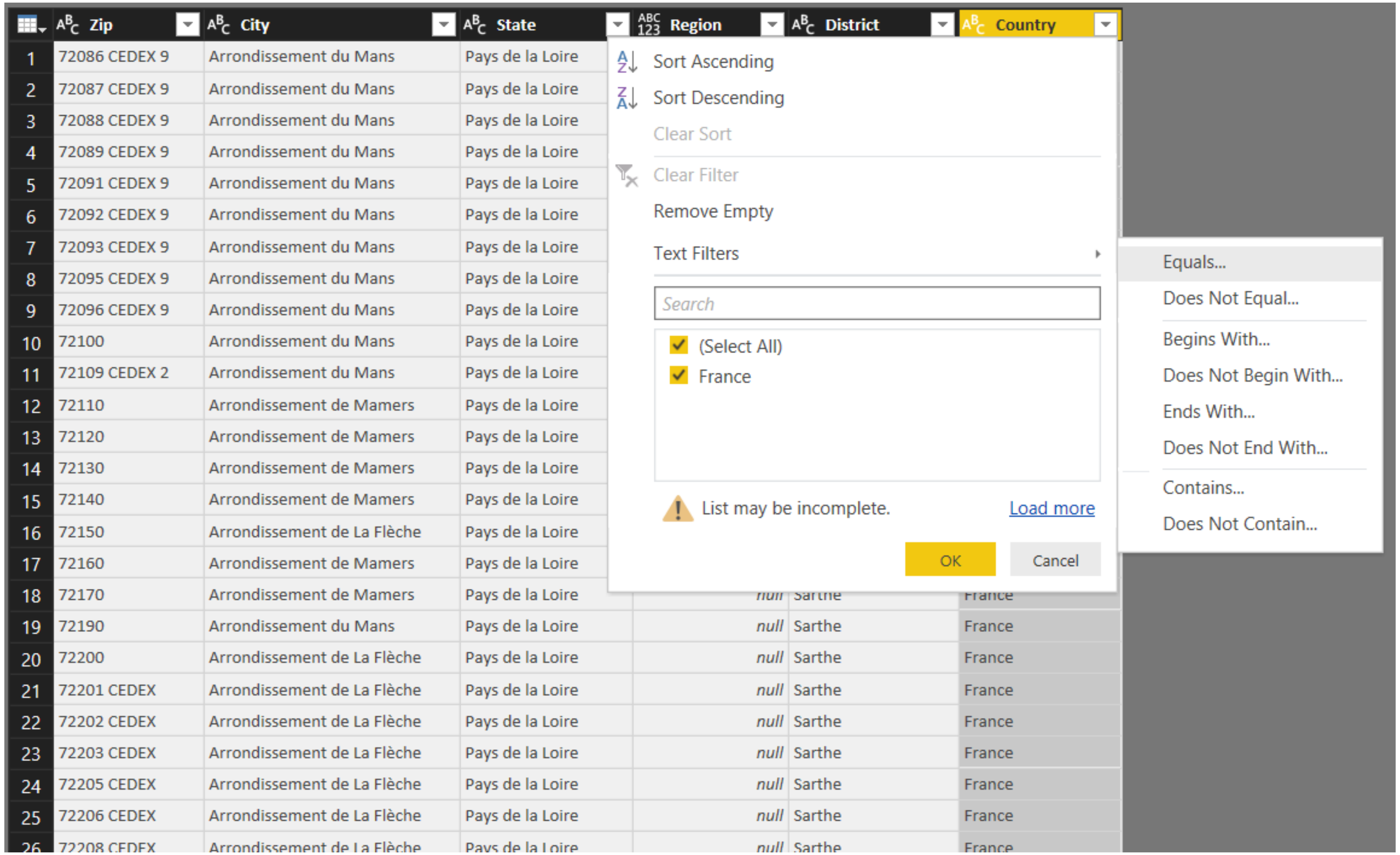
Current Value: Spain

Select the type of data the parameter requires.
If you want the user to select from a list, fill out the options in the table.

Select a default and a current value.

Query Parameters

- Go to the Geography table
- Set a filter on Country

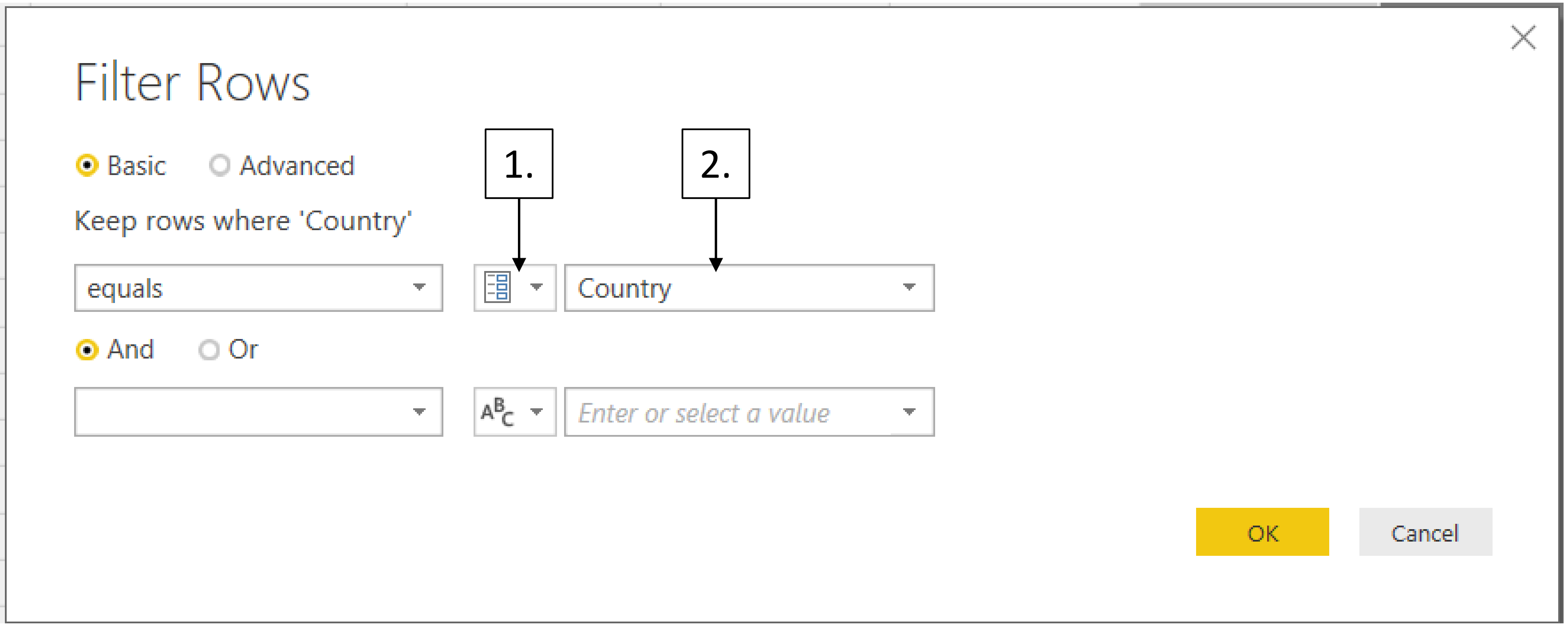


	Zip	City	State	Region	District	Country
1	72086 CEDEX 9	Arrondissement du Mans	Pays de la Loire			
2	72087 CEDEX 9	Arrondissement du Mans	Pays de la Loire			
3	72088 CEDEX 9	Arrondissement du Mans	Pays de la Loire			
4	72089 CEDEX 9	Arrondissement du Mans	Pays de la Loire			
5	72091 CEDEX 9	Arrondissement du Mans	Pays de la Loire			
6	72092 CEDEX 9	Arrondissement du Mans	Pays de la Loire			
7	72093 CEDEX 9	Arrondissement du Mans	Pays de la Loire			
8	72095 CEDEX 9	Arrondissement du Mans	Pays de la Loire			
9	72096 CEDEX 9	Arrondissement du Mans	Pays de la Loire			
10	72100	Arrondissement du Mans	Pays de la Loire			
11	72109 CEDEX 2	Arrondissement du Mans	Pays de la Loire			
12	72110	Arrondissement de Mamers	Pays de la Loire			
13	72120	Arrondissement de Mamers	Pays de la Loire			
14	72130	Arrondissement de Mamers	Pays de la Loire			
15	72140	Arrondissement de Mamers	Pays de la Loire			
16	72150	Arrondissement de La Flèche	Pays de la Loire			
17	72160	Arrondissement de Mamers	Pays de la Loire			
18	72170	Arrondissement de Mamers	Pays de la Loire	null	Sarthe	France
19	72190	Arrondissement du Mans	Pays de la Loire	null	Sarthe	France
20	72200	Arrondissement de La Flèche	Pays de la Loire	null	Sarthe	France
21	72201 CEDEX	Arrondissement de La Flèche	Pays de la Loire	null	Sarthe	France
22	72202 CEDEX	Arrondissement de La Flèche	Pays de la Loire	null	Sarthe	France
23	72203 CEDEX	Arrondissement de La Flèche	Pays de la Loire	null	Sarthe	France
24	72205 CEDEX	Arrondissement de La Flèche	Pays de la Loire	null	Sarthe	France
25	72206 CEDEX	Arrondissement de La Flèche	Pays de la Loire	null	Sarthe	France
26	72208 CEDEX	Arrondissement de La Flèche	Pays de la Loire	null	Sarthe	France

Query Parameters

- In the Filter Rows window, select Parameter in dropdown 1.

- Select the parameter defined in dropdown 2.



Filter Rows

● Basic ○ Advanced

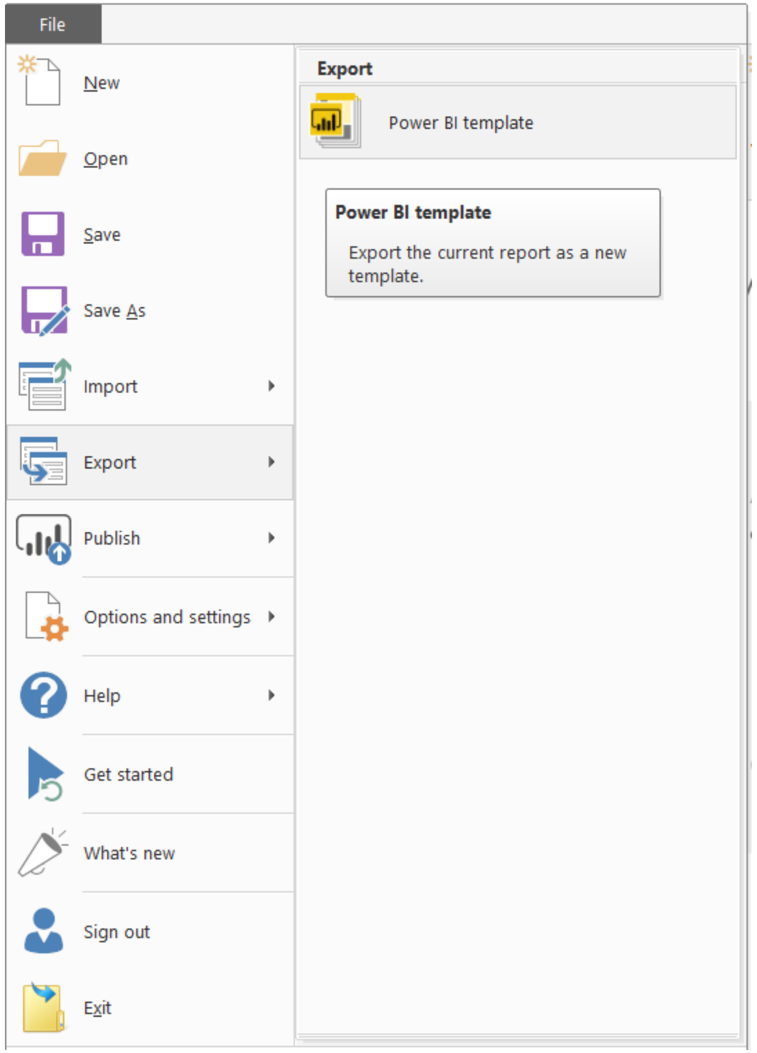
Keep rows where 'Country'

equals | Country

● And ○ Or

Enter or select a value

Query Parameters



Power BI template

Export the current report as a new template.

Export the file as a Power BI Template, and provide a description.

When a user opens the Power BI Template, they will be prompted to input the parameter.

Power BI will only import the data corresponding to the input parameter.

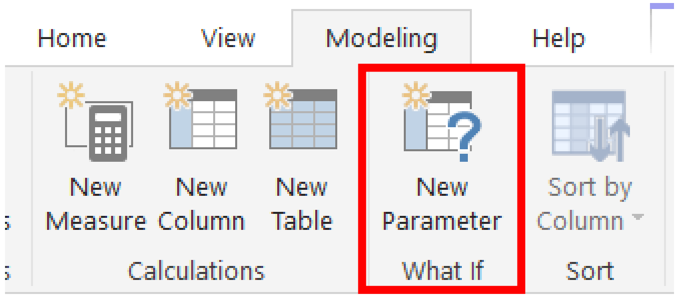
What if…- analysis

- Power BI allows you to use a parameter in calculations to determine the outcome of a ‘what if…’ scenario.

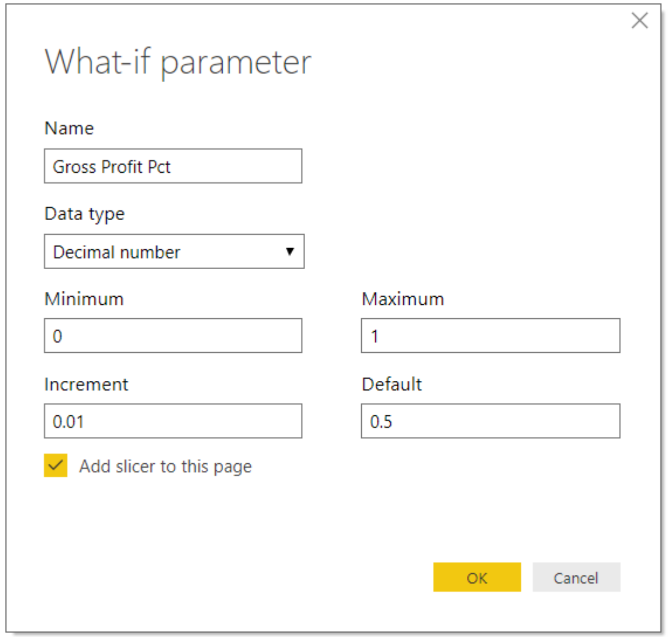
- The parameter is variable, so the user can test arrange of different possibilities and see the result in real time.

What if...- analysis, Gross profit

- On the *Modelling Tab*, select *New Parameter*.



- Set a meaning, user friendly name for the parameter.
- Select a minimum and maximum, in this case, 0 and 1.
- Set an increment value, in this case, 0.01.
- Ensure *Add slicer to this page* is checked.


What-if parameter

Name: Gross Profit Pct

Data type: Decimal number

Minimum: 0

Maximum: 1

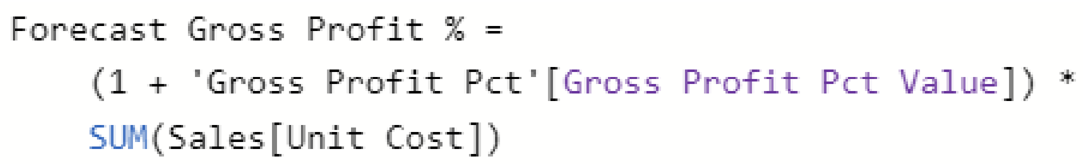
Increment: 0.01

Default: 0.5

☑ Add slicer to this page

What if…- analysis, Gross profit

- A slicer is added to the page, with a slider, allowing the user to select a value.
- Notice a new table appears in the Fields list. Expanding this table shows the column and measure which were generated.
- You can use this measure in further calculations.
- Type the following measure in the Sales table:

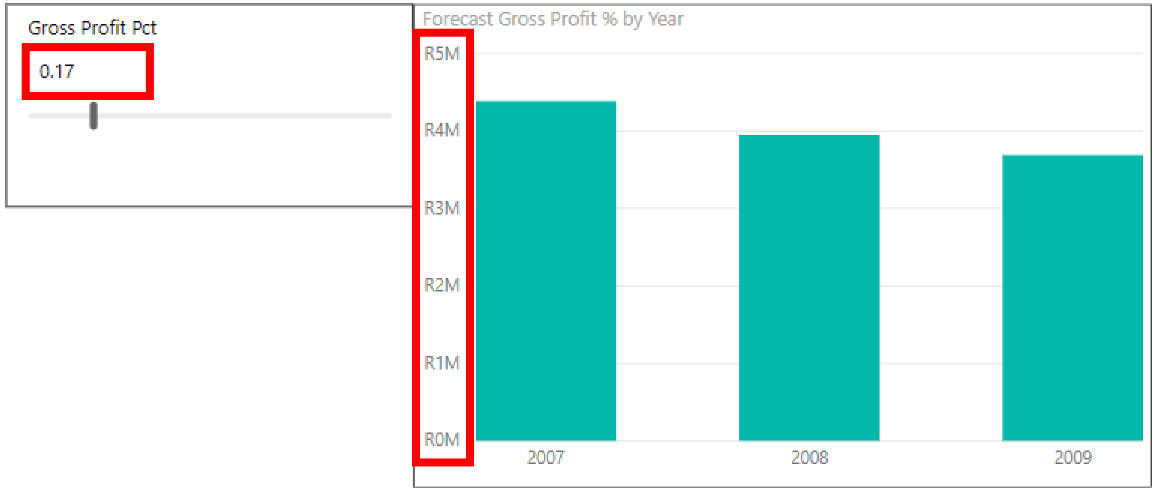


```
Forecast Gross Profit % =
    (1 + 'Gross Profit Pct'[Gross Profit Pct Value]) *
    SUM(Sales[Unit Cost])
```

What if…- analysis, Gross profit

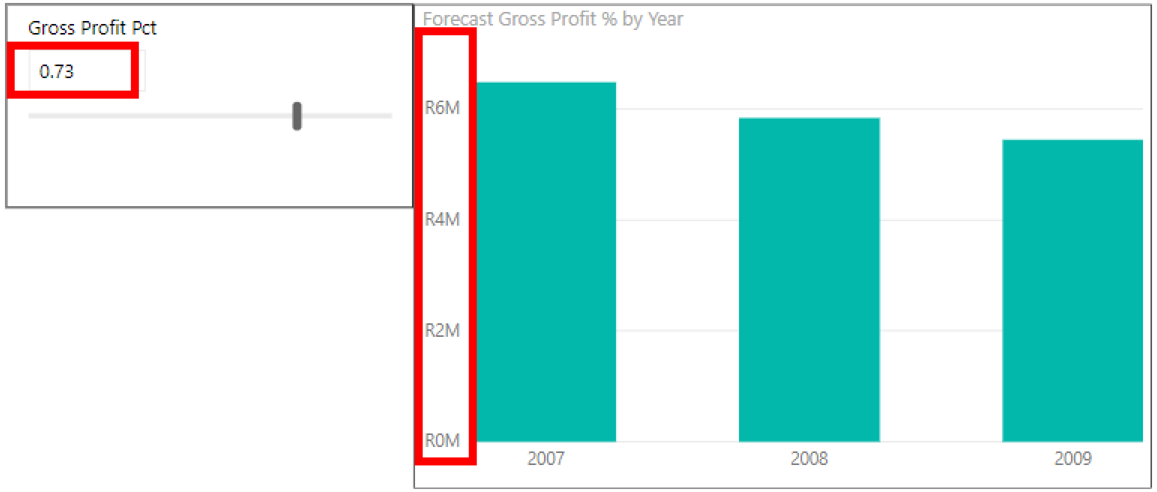
- Create a column chart showing the *Forecast Gross Profit %* by Date from the Date table.

- Change the value of the slicer and note the corresponding change to the visual.



Gross Profit Pct

0.17



Gross Profit Pct

0.73

Report Interactivity – Buttons and bookmarks

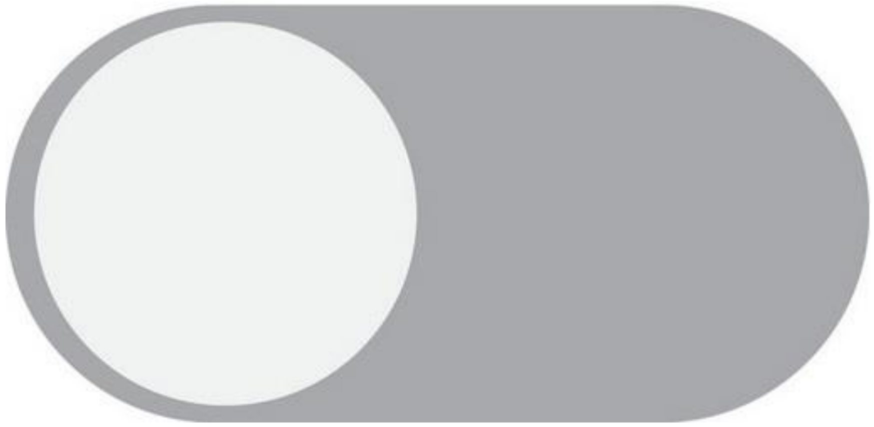
- Power BI features buttons. Any image can be used as a button and have a specific action associated with it.
  - Actions include:
    - Back: Takes you to the previous page you were on
    - Bookmark: Allows you to select a bookmark.
    - Q&A: Opens the Q&A browser to see frequently asked questions with their associated answers.

- The Selection Pane allows you to show or hide various visuals. Using this is conjunction with bookmarks and buttons will make you report very interactive.

Report Interactivity – Buttons and bookmarks

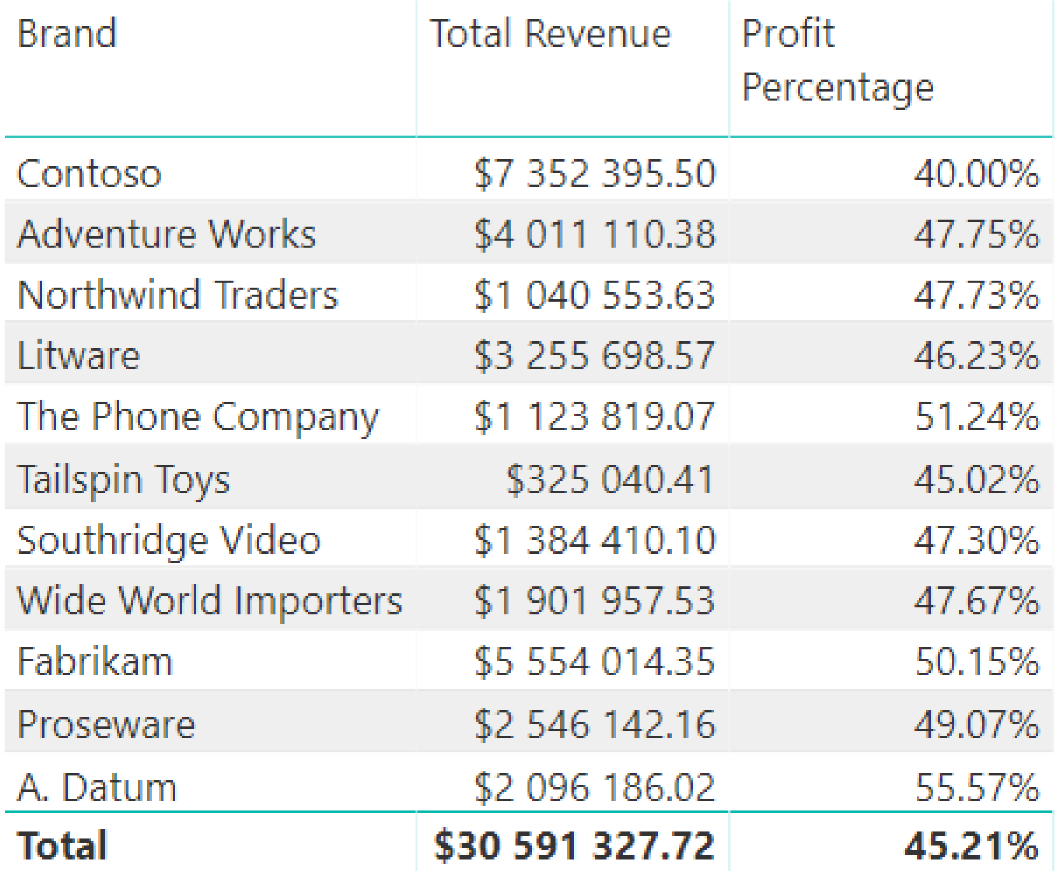
- Insert the following images:
  - Toggle 0.png
  - Toggle 180.png

- We are going to use these images to create a toggle switch to change the view of a visual from a chart to a table.



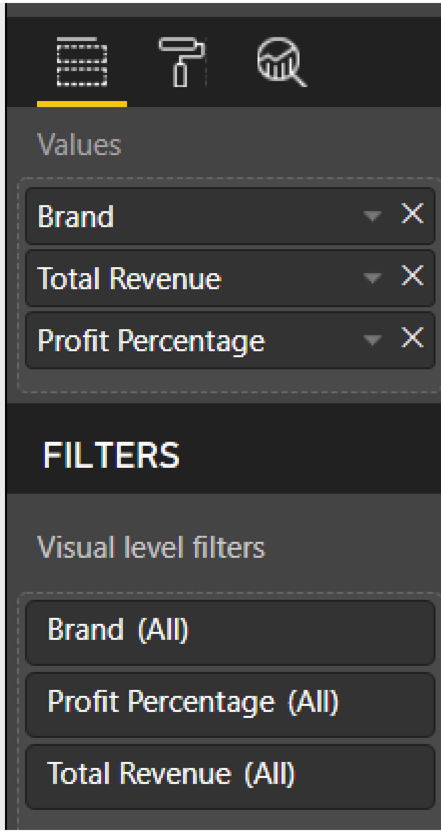
Report Interactivity – Buttons and bookmarks

- Create the following table visual



Brand	Total Revenue	Profit Percentage
Contoso	$7 352 395.50	40.00%
Adventure Works	$4 011 110.38	47.75%
Northwind Traders	$1 040 553.63	47.73%
Litware	$3 255 698.57	46.23%
The Phone Company	$1 123 819.07	51.24%
Tailspin Toys	$325 040.41	45.02%
Southridge Video	$1 384 410.10	47.30%
Wide World Importers	$1 901 957.53	47.67%
Fabrikam	$5 554 014.35	50.15%
Proseware	$2 546 142.16	49.07%
A. Datum	$2 096 186.02	55.57%
Total	**$30 591 327.72**	**45.21%**

- Copy and paste a second copy of the visual.



Values

Brand

Total Revenue

Profit Percentage

FILTERS

Visual level filters

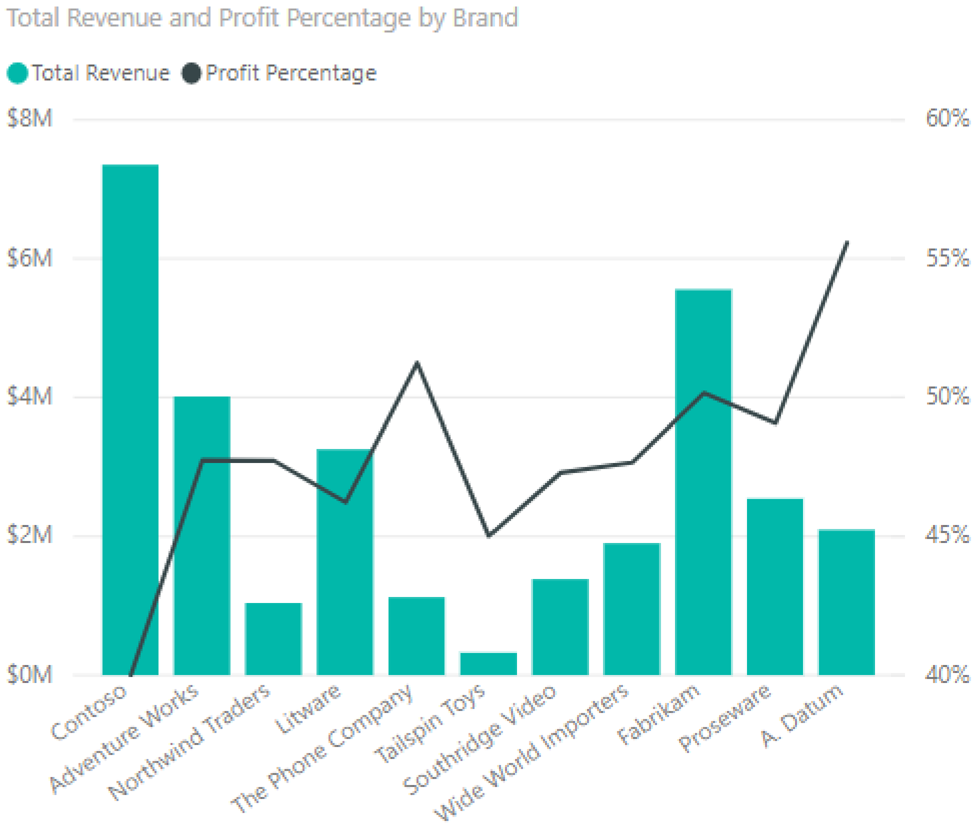
Brand (All)

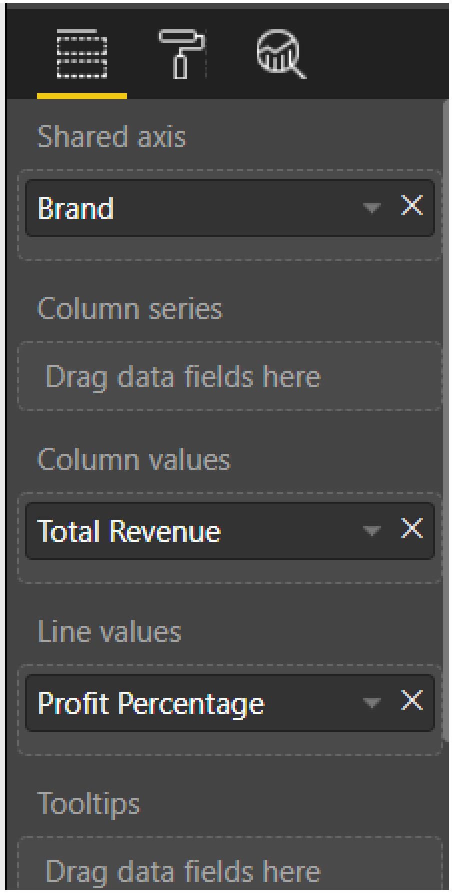
Profit Percentage (All)

Total Revenue (All)

Report Interactivity – Buttons and bookmarks

- Change the second copy to a Column and Line Combo Chart.




Shared axis

Brand

Column series

Drag data fields here

Column values

Total Revenue

Line values

Profit Percentage

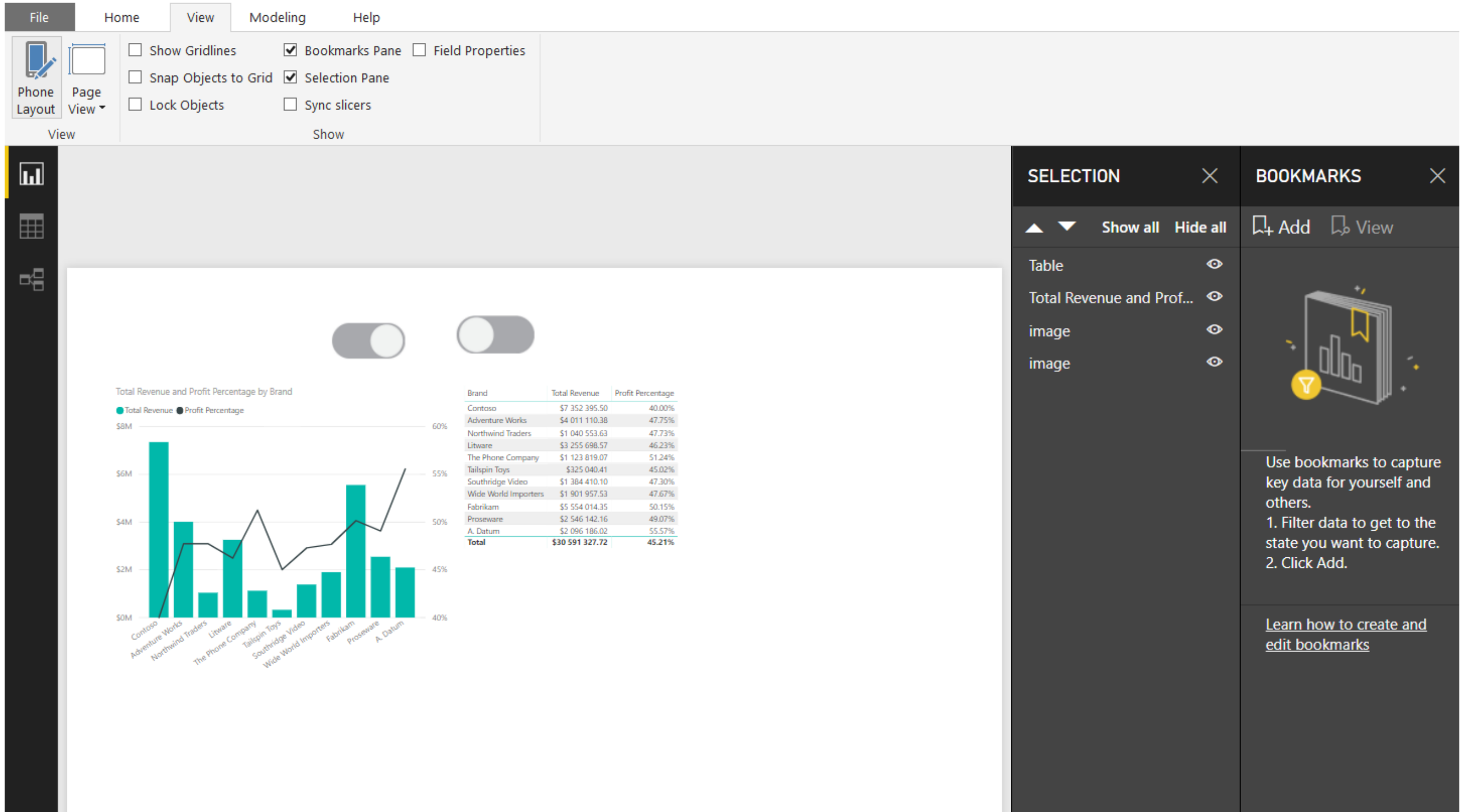
Tooltips

Drag data fields here

Report Interactivity – Buttons and bookmarks

ALTRON | KARABINA

- In the View tab, check the Bookmarks Pane and the Selection Pane.



☐ Show Gridlines
☐ Snap Objects to Grid
☐ Lock Objects
☑ Bookmarks Pane
☑ Selection Pane
☐ Sync slicers
☐ Field Properties

Table
Total Revenue and Prof...
image
image

Brand	Total Revenue	Profit Percentage
Contoso	$7 352 395.50	40.00%
Adventure Works	$4 011 110.38	47.75%
Northwind Traders	$1 040 553.63	47.73%
Litware	$3 255 698.57	46.23%
The Phone Company	$1 123 819.07	51.24%
Tailspin Toys	$325 040.41	45.02%
Southridge Video	$1 384 410.10	47.30%
Wide World Importers	$1 901 957.53	47.67%
Fabrikam	$5 554 014.35	50.15%
Proseware	$2 546 142.16	49.07%
A. Datum	$2 096 186.02	55.57%
Total	**$30 591 327.72**	**45.21%**

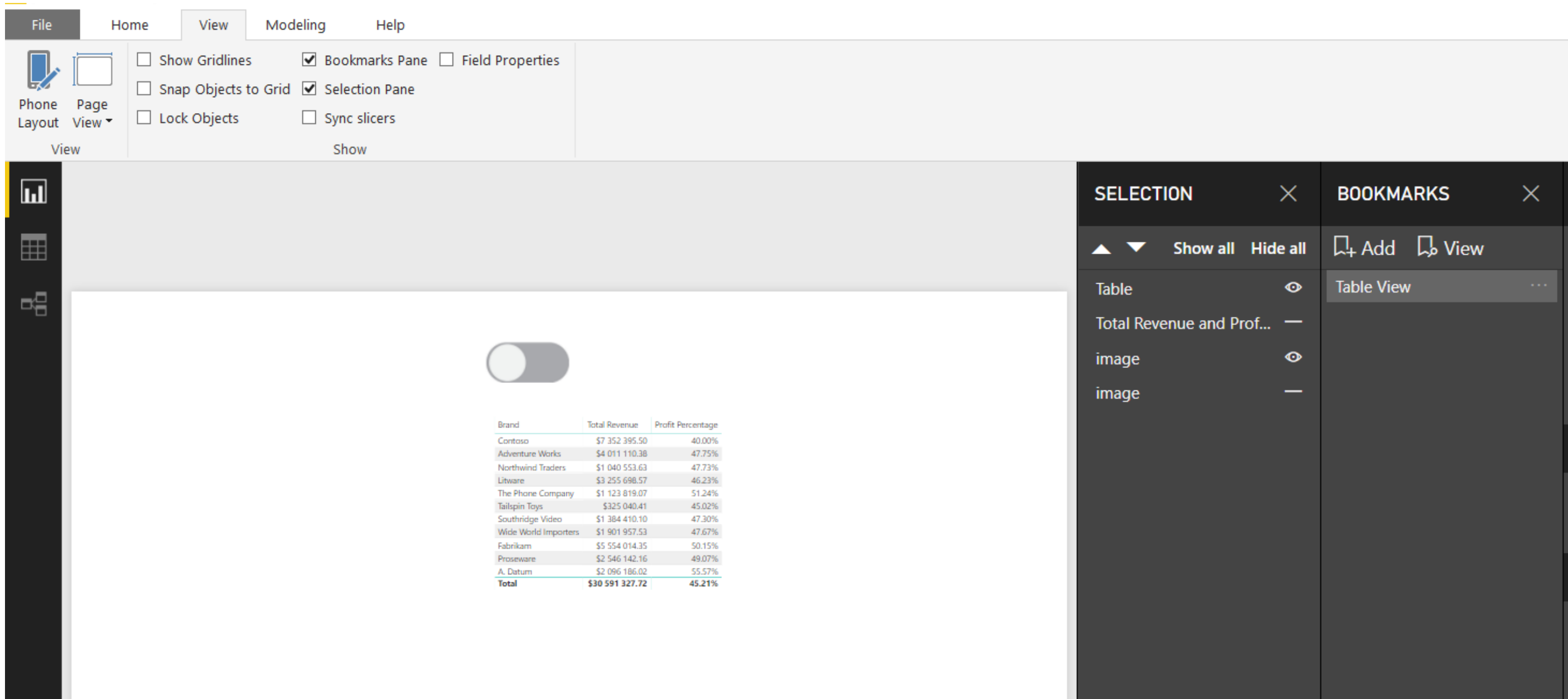
Use bookmarks to capture key data for yourself and others.
1. Filter data to get to the state you want to capture.
2. Click Add.

Learn how to create and edit bookmarks

Report Interactivity – Buttons and bookmarks

- In the Selection Pane, hide a toggle and the chart visual, such that you only have a table view.

- Save this bookmark as Table View.



□ Show Gridlines
□ Snap Objects to Grid
□ Lock Objects
☑ Bookmarks Pane
☑ Selection Pane
□ Sync slicers
□ Field Properties

SELECTION

- Table 👁
- Total Revenue and Prof... —
- image 👁
- image —

BOOKMARKS

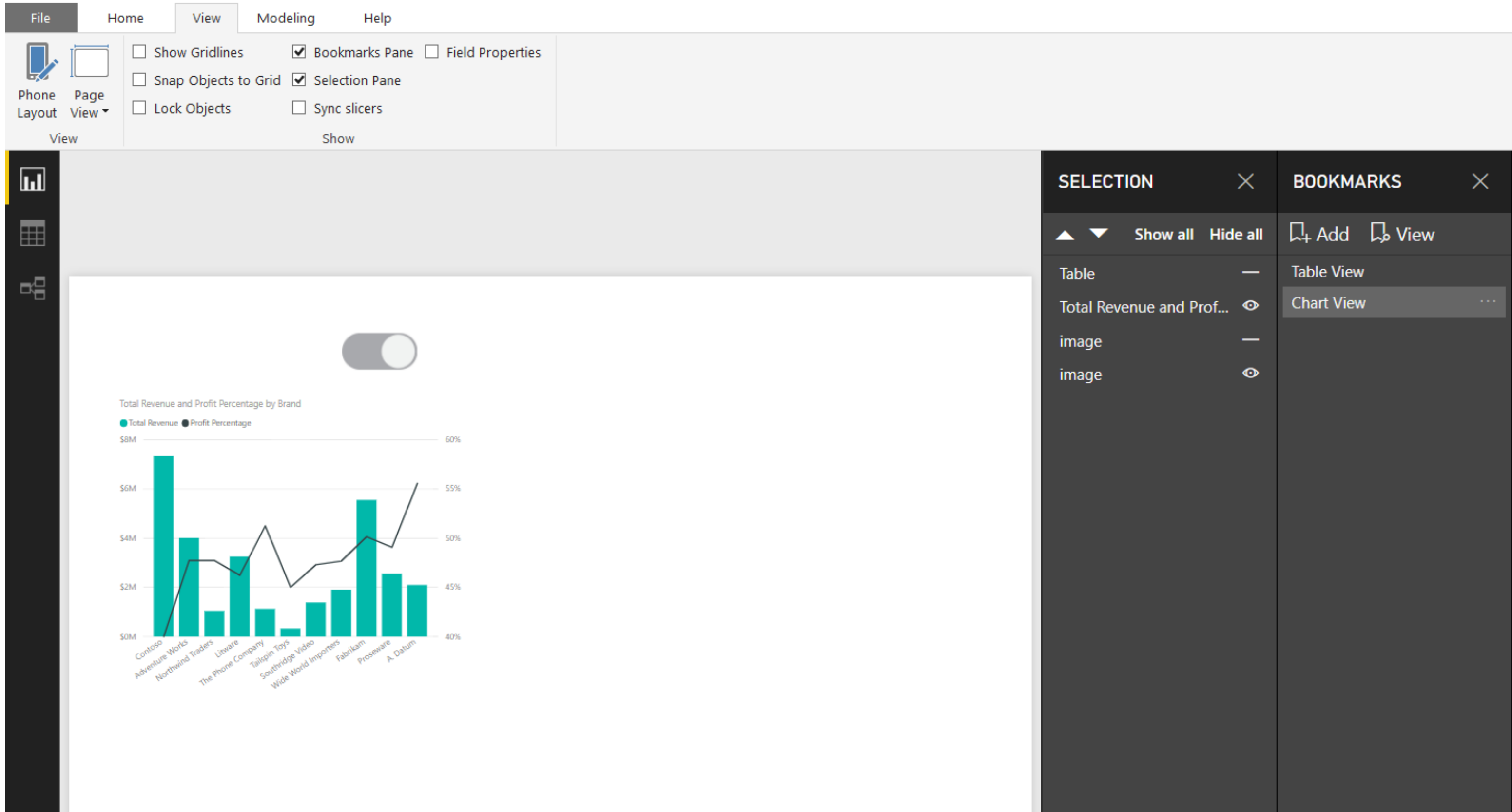
- Table View

Brand	Total Revenue	Profit Percentage
Contoso	$7 352 395.50	40.00%
Adventure Works	$4 011 110.38	47.75%
Northwind Traders	$1 040 553.63	47.73%
Litware	$3 255 698.57	46.23%
The Phone Company	$1 123 819.07	51.24%
Tailspin Toys	$325 040.41	45.02%
Southridge Video	$1 384 410.10	47.30%
Wide World Importers	$1 901 957.53	47.67%
Fabrikam	$5 554 014.35	50.15%
Proseware	$2 546 142.16	49.07%
A. Datum	$2 096 186.02	55.57%
Total	**$30 591 327.72**	**45.21%**

Report Interactivity – Buttons and bookmarks

- In the Selection Pane, hide the other toggle and the table, and show the first toggle and chart.

- Save this bookmark as Chart View.



□ Show Gridlines
□ Snap Objects to Grid
□ Lock Objects
☑ Bookmarks Pane
☑ Selection Pane
□ Sync slicers
□ Field Properties

SELECTION

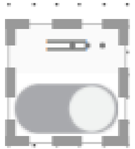
Table	—
Total Revenue and Prof...	👁
image	—
image	👁

BOOKMARKS

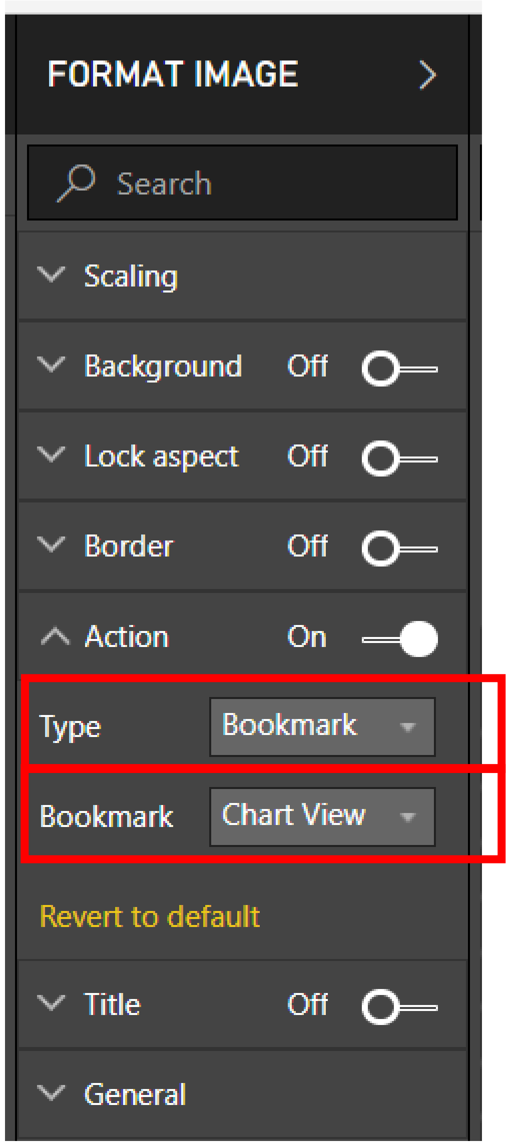
Table View
Chart View

Report Interactivity – Buttons and bookmarks

- Select the Table View bookmark.
- Select the toggle button that is visible in this bookmark.



- In the *Format Image* Pane
  - *Action -> Type: Bookmark*
  - *Bookmark -> Chart View*



FORMAT IMAGE

Search

Scaling

Background Off

Lock aspect Off

Border Off

Action On

Type: Bookmark

Bookmark: Chart View

Revert to default

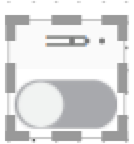
Title Off

General

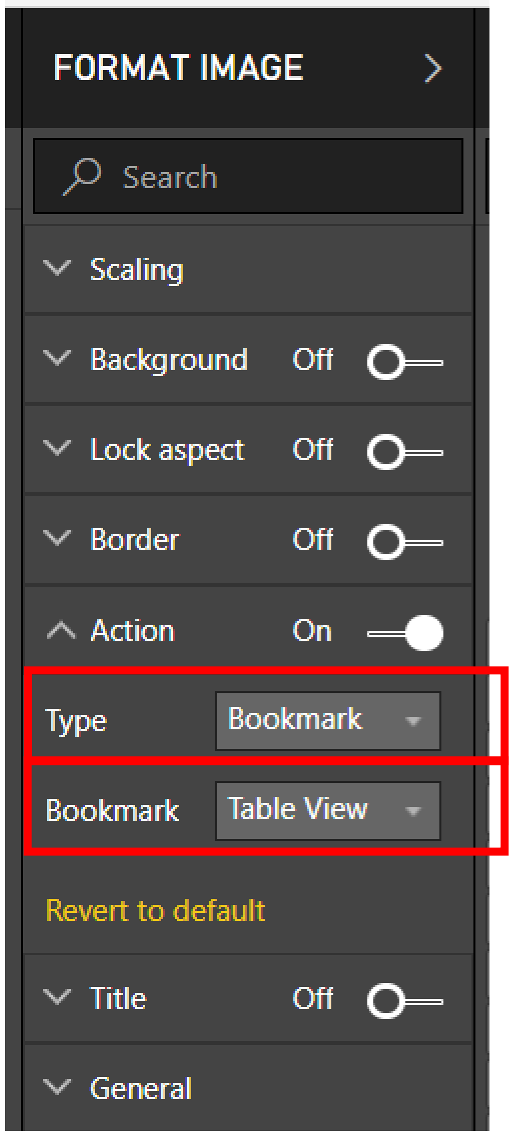
Report Interactivity – Buttons and bookmarks

- Select the Chart View bookmark.

- Select the toggle button that is visible in this bookmark.



- In the *Format Image* Pane
  - *Action -> Type: Bookmark*
  - *Bookmark -> Table View*



FORMAT IMAGE

Search

Scaling

Background Off

Lock aspect Off

Border Off

Action On

Type Bookmark

Bookmark Table View

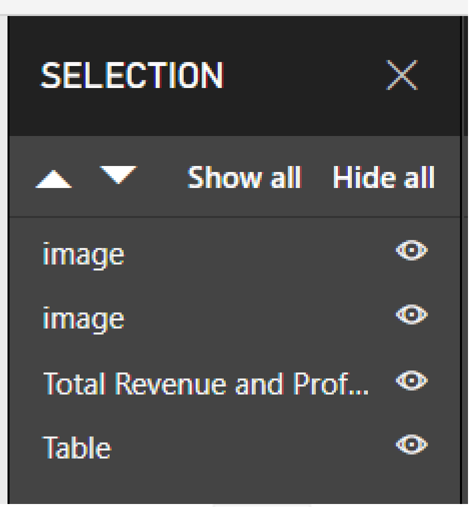
Revert to default

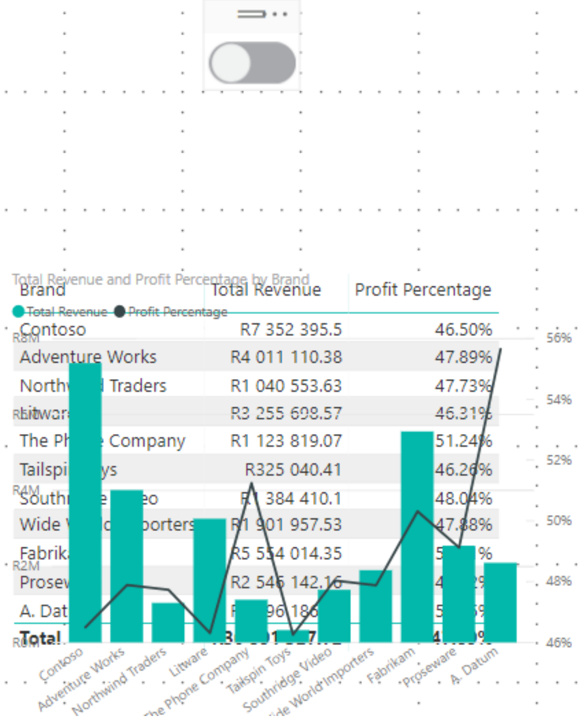
Title Off

General

Report Interactivity – Buttons and bookmarks

- In the selection pane, make all of the objects visible.
- Align the Table and Chart visuals on top of each other.
- Align the toggle switch on top of each other.
- When selecting the toggles, the view of the visual will toggle between a table and chart.




Total Revenue and Profit Percentage by Brand

Brand	Total Revenue	Profit Percentage
Contoso	R7 352 395.5	46.50%
Adventure Works	R4 011 110.38	47.89%
Northw[illegible] Traders	R1 040 553.63	47.73%
Litwar[illegible]	R3 255 608.57	46.31%
The Ph[illegible] Company	R1 123 819.07	51.24%
Tailspi[illegible]ys	R325 040.41	46.26%
South[illegible]eo	R1 384 410.1	48.04%
Wide [illegible]porters	R1 901 957.53	47.88%
Fabrik[illegible]	R5 554 014.35	[illegible]%
Prosew[illegible]	R2 546 142.16	[illegible]
A. Dat[illegible]	[illegible]96 186	[illegible]
Total	[illegible]	[illegible]

Dynamic Security

- Power BI supports dynamic Row Level Security.

- This is where a the user name for a particular user is used to filter the content of the report specifically for that user.

Dynamic Security

- The first pre-requisite is that you have a table containing all of the user’s Active Directory User Names. This table must be in the Power BI Data Model

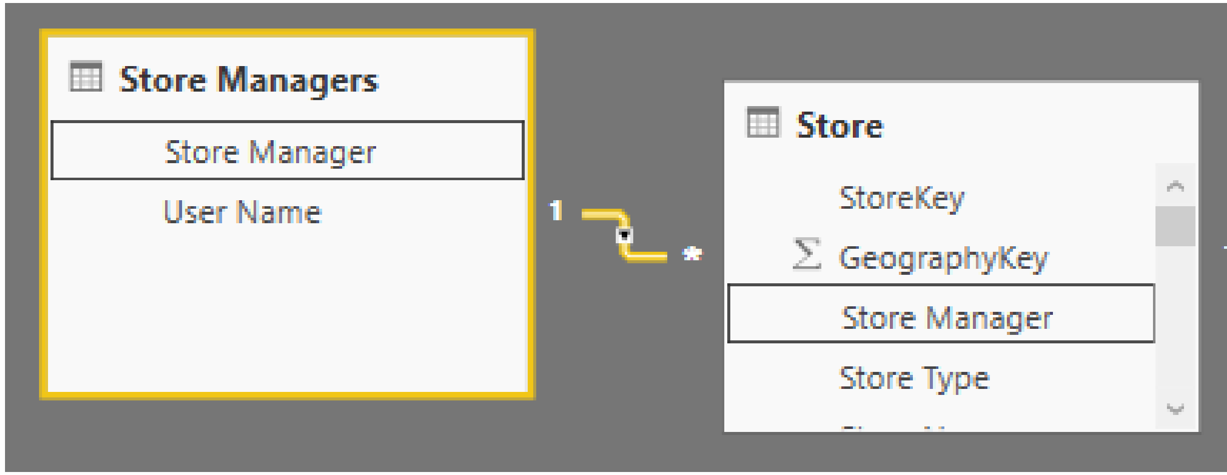
- There must be a relationship between this table and the other tables you want to filter.
    - This implies you have a foreign key in your fact table that links the table to the Users table.

- This is useful if you have a Power BI report that contains HR data.
    - Any person who logs onto app.powerbi.com will only see data pertaining to them.

Dynamic Security

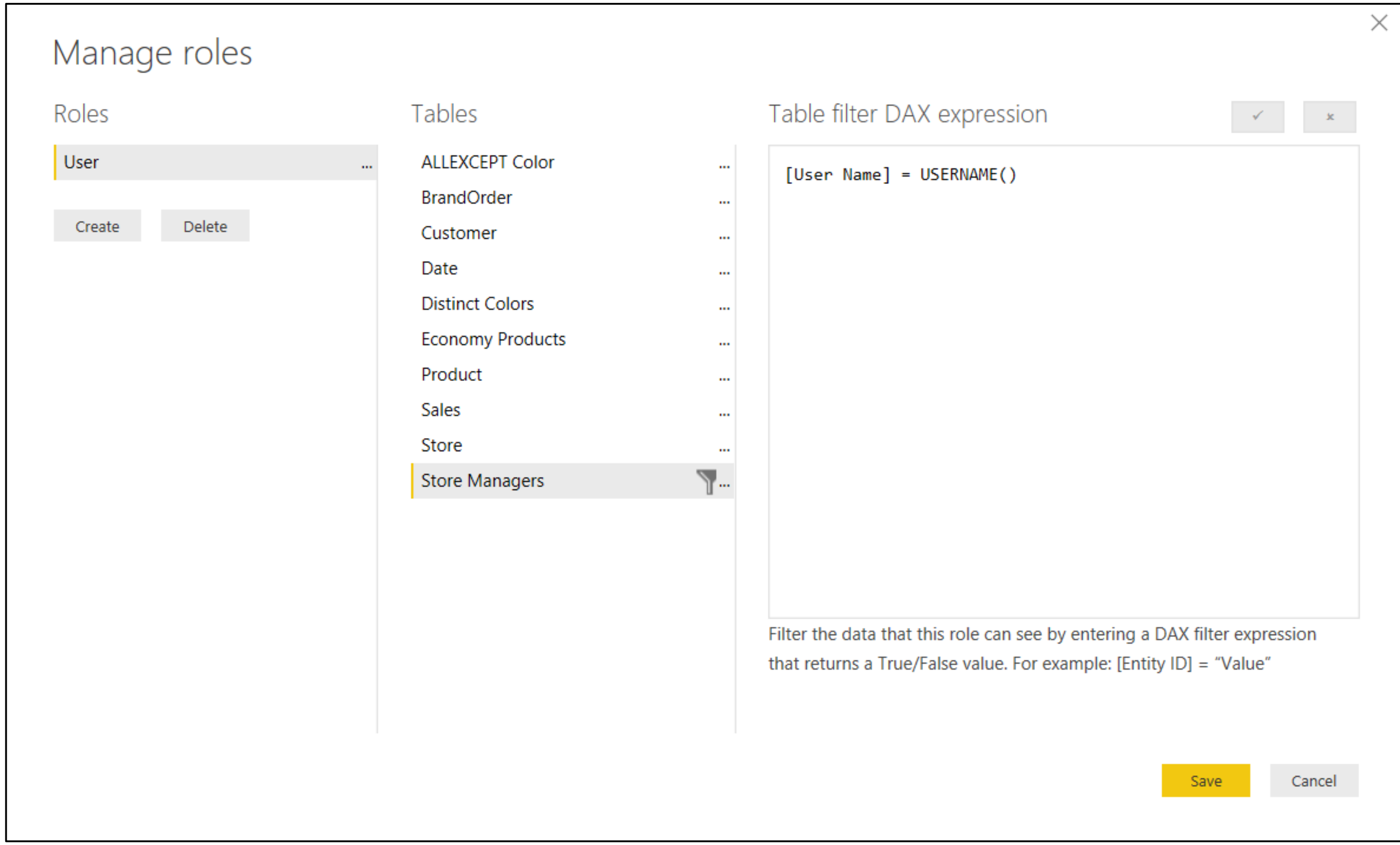
- In the example data, the Store Managers table contains a list of store manages and associated User Names.

- Ensure there is a relationship between the Store Managers and Store tables as shown.



Dynamic Security

- Under the *Modelling* tab, select *Manage Roles*.
- Create a new role called *User*.
- Under the *Store Managers* table, set a new filter on the [User Name] column.
- Set the expression to:
  - [User Name] = USERNAME()

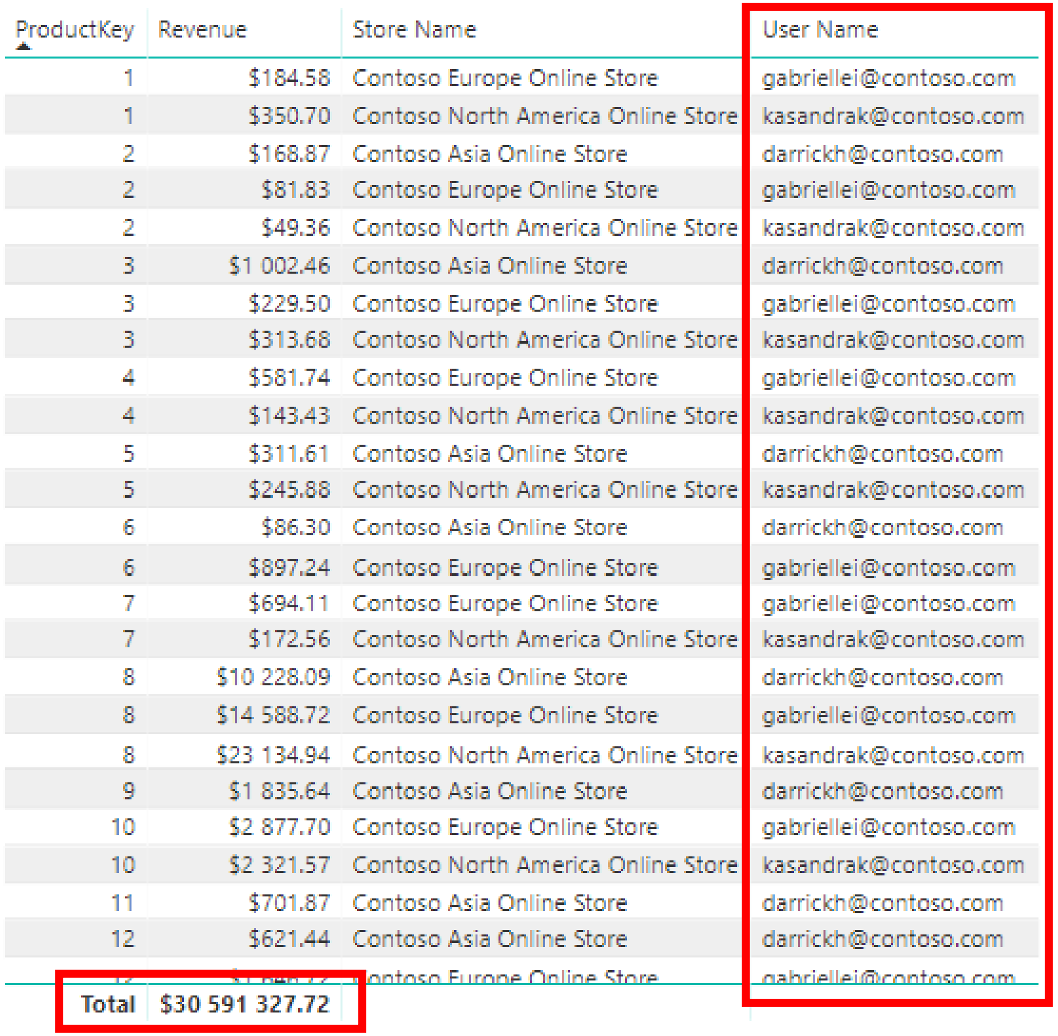


```
[User Name] = USERNAME()
```

Filter the data that this role can see by entering a DAX filter expression that returns a True/False value. For example: [Entity ID] = "Value"

Dynamic Security – Testing it out

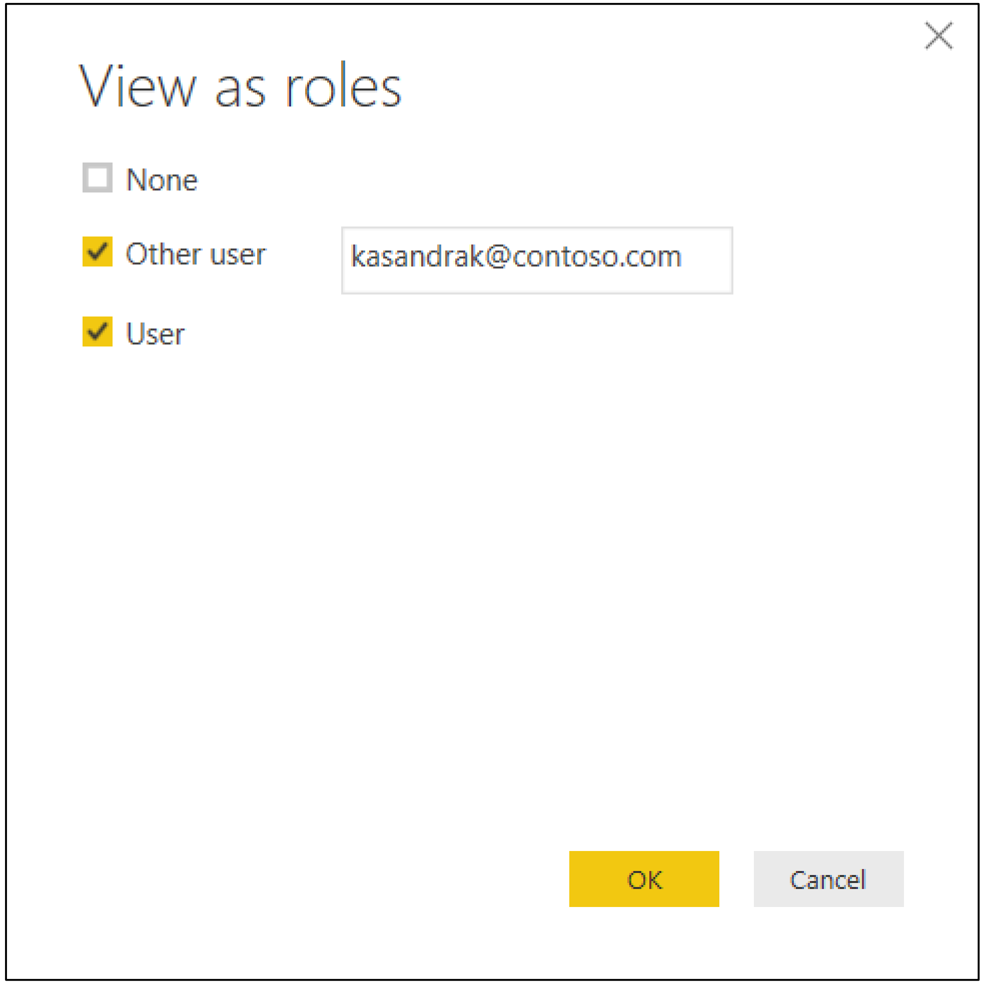
- Create a table visual containing
  Sales[Revenue]
  Store[Store Name]
  Store Managers[User Name]



ProductKey	Revenue	Store Name	User Name
1	$184.58	Contoso Europe Online Store	gabriellei@contoso.com
1	$350.70	Contoso North America Online Store	kasandrak@contoso.com
2	$168.87	Contoso Asia Online Store	darrickh@contoso.com
2	$81.83	Contoso Europe Online Store	gabriellei@contoso.com
2	$49.36	Contoso North America Online Store	kasandrak@contoso.com
3	$1 002.46	Contoso Asia Online Store	darrickh@contoso.com
3	$229.50	Contoso Europe Online Store	gabriellei@contoso.com
3	$313.68	Contoso North America Online Store	kasandrak@contoso.com
4	$581.74	Contoso Europe Online Store	gabriellei@contoso.com
4	$143.43	Contoso North America Online Store	kasandrak@contoso.com
5	$311.61	Contoso Asia Online Store	darrickh@contoso.com
5	$245.88	Contoso North America Online Store	kasandrak@contoso.com
6	$86.30	Contoso Asia Online Store	darrickh@contoso.com
6	$897.24	Contoso Europe Online Store	gabriellei@contoso.com
7	$694.11	Contoso Europe Online Store	gabriellei@contoso.com
7	$172.56	Contoso North America Online Store	kasandrak@contoso.com
8	$10 228.09	Contoso Asia Online Store	darrickh@contoso.com
8	$14 588.72	Contoso Europe Online Store	gabriellei@contoso.com
8	$23 134.94	Contoso North America Online Store	kasandrak@contoso.com
9	$1 835.64	Contoso Asia Online Store	darrickh@contoso.com
10	$2 877.70	Contoso Europe Online Store	gabriellei@contoso.com
10	$2 321.57	Contoso North America Online Store	kasandrak@contoso.com
11	$701.87	Contoso Asia Online Store	darrickh@contoso.com
12	$621.44	Contoso Asia Online Store	darrickh@contoso.com
12	[illegible]	Contoso Europe Online Store	gabriellei@contoso.com
Total	**$30 591 327.72**		

Dynamic Security – Testing it out

- To test out the role, select *View as Roles*.
- Check both the *User* and the *Other user* boxes.
- Enter the e-mail address of a user for which you want to test the role.
  - kasandrak@contoso.com



View as roles

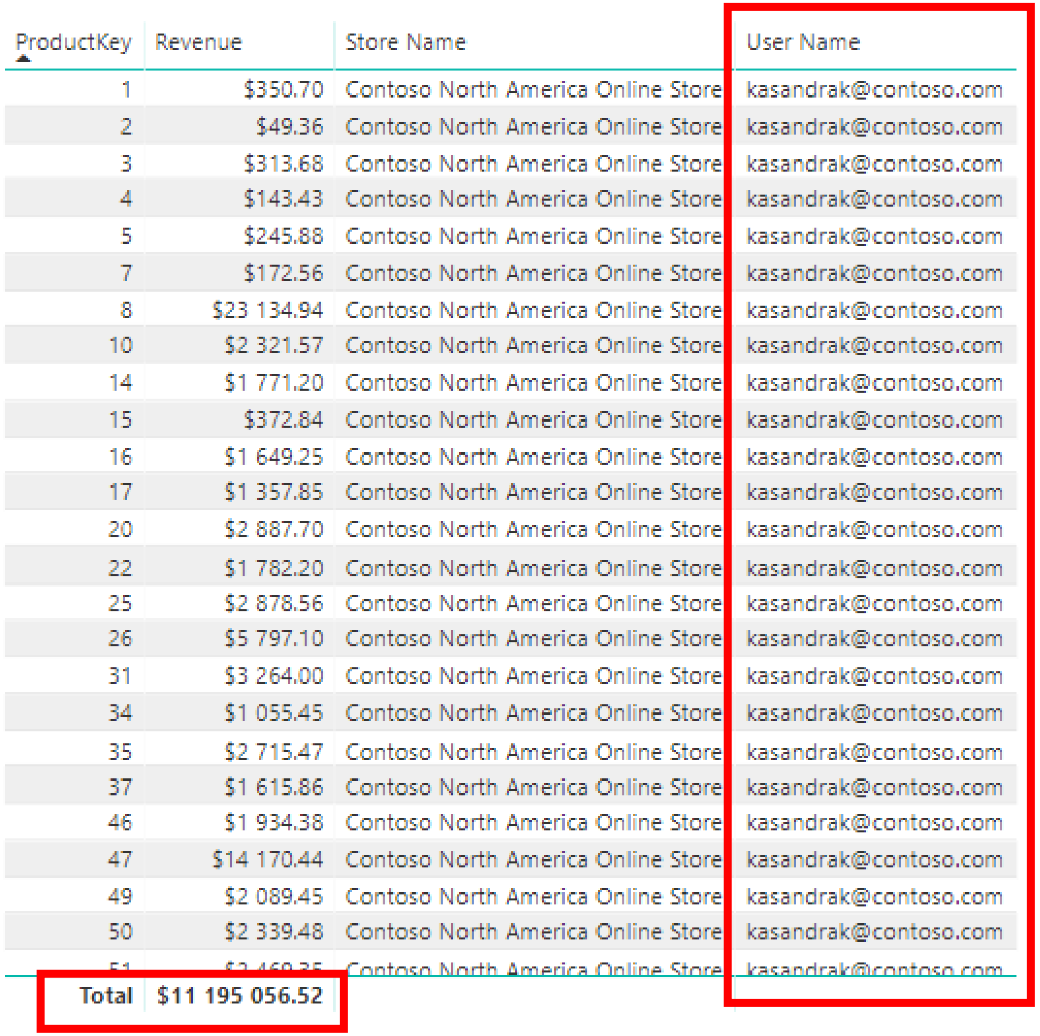
□ None

☑ Other user: kasandrak@contoso.com

☑ User

Dynamic Security – Testing it out

- Create a table visual containing
  Sales[Revenue]
  Store[Store Name]
  Store Managers[User Name]



ProductKey	Revenue	Store Name	User Name
1	$350.70	Contoso North America Online Store	kasandrak@contoso.com
2	$49.36	Contoso North America Online Store	kasandrak@contoso.com
3	$313.68	Contoso North America Online Store	kasandrak@contoso.com
4	$143.43	Contoso North America Online Store	kasandrak@contoso.com
5	$245.88	Contoso North America Online Store	kasandrak@contoso.com
7	$172.56	Contoso North America Online Store	kasandrak@contoso.com
8	$23 134.94	Contoso North America Online Store	kasandrak@contoso.com
10	$2 321.57	Contoso North America Online Store	kasandrak@contoso.com
14	$1 771.20	Contoso North America Online Store	kasandrak@contoso.com
15	$372.84	Contoso North America Online Store	kasandrak@contoso.com
16	$1 649.25	Contoso North America Online Store	kasandrak@contoso.com
17	$1 357.85	Contoso North America Online Store	kasandrak@contoso.com
20	$2 887.70	Contoso North America Online Store	kasandrak@contoso.com
22	$1 782.20	Contoso North America Online Store	kasandrak@contoso.com
25	$2 878.56	Contoso North America Online Store	kasandrak@contoso.com
26	$5 797.10	Contoso North America Online Store	kasandrak@contoso.com
31	$3 264.00	Contoso North America Online Store	kasandrak@contoso.com
34	$1 055.45	Contoso North America Online Store	kasandrak@contoso.com
35	$2 715.47	Contoso North America Online Store	kasandrak@contoso.com
37	$1 615.86	Contoso North America Online Store	kasandrak@contoso.com
46	$1 934.38	Contoso North America Online Store	kasandrak@contoso.com
47	$14 170.44	Contoso North America Online Store	kasandrak@contoso.com
49	$2 089.45	Contoso North America Online Store	kasandrak@contoso.com
50	$2 339.48	Contoso North America Online Store	kasandrak@contoso.com
51	$2 469.35	Contoso North America Online Store	kasandrak@contoso.com
Total	$11 195 056.52		

Architecting a tabular model

- You may want to upload the data model you have created to the Power BI Service, and then connect to this as a data source using Power BI or Excel.

- This enables the BI developer to create a model, metrics and measures to which anyone else can connect.

- This ensures there is one version of the data, which is controlled. The developer can rest assured that people using the model for self service BI will not inadvertently break the model.

Architecting a tabular model – Measures table

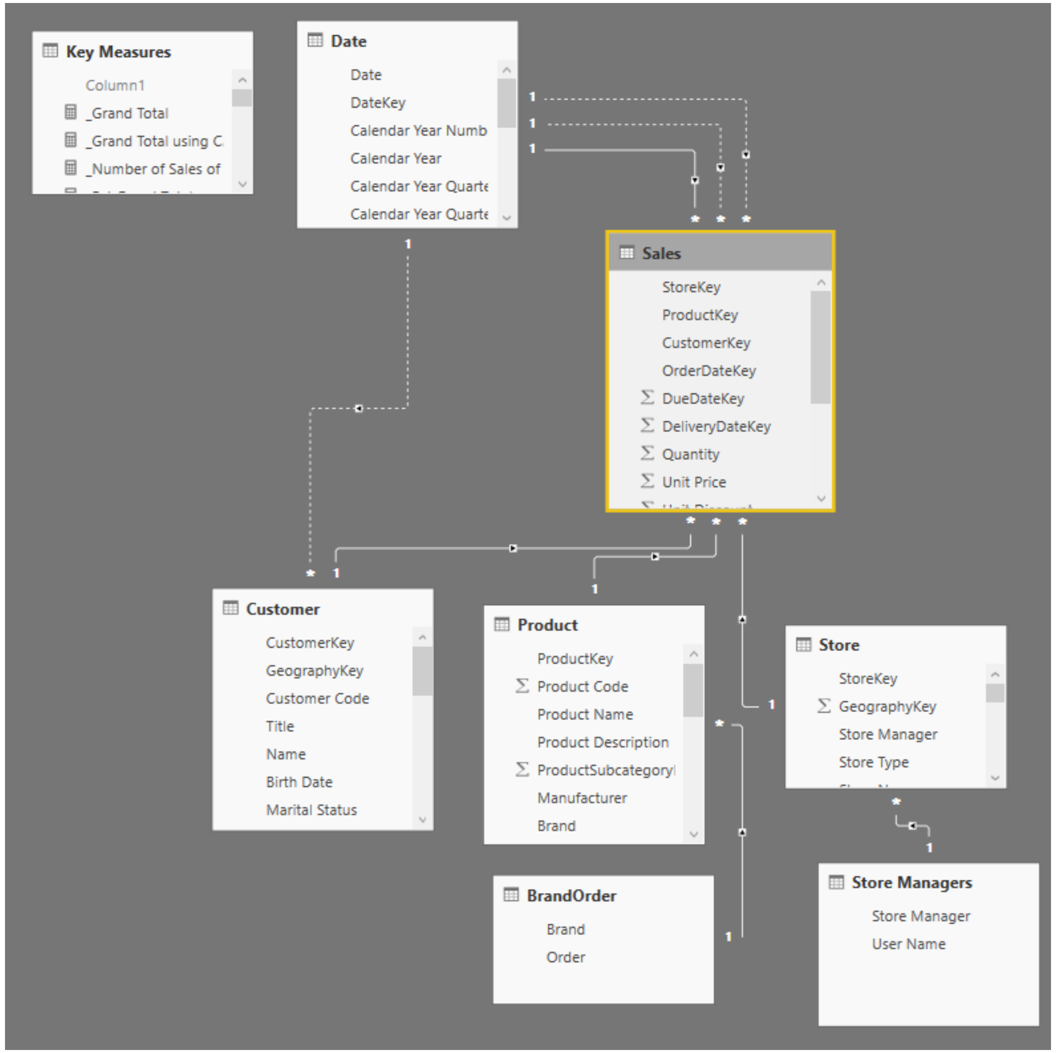
1. Insert new Table – Enter Data
2. Type “1” in the data field of column 1 and rename the table
3. Load Table
4. Create your new measure
5. Hide Column 1 in your new measure table
6. Hide and show your fields pane
7. The measure table should now appear at the top of the table list

Architecting a tabular model

ALTRON | KARABINA

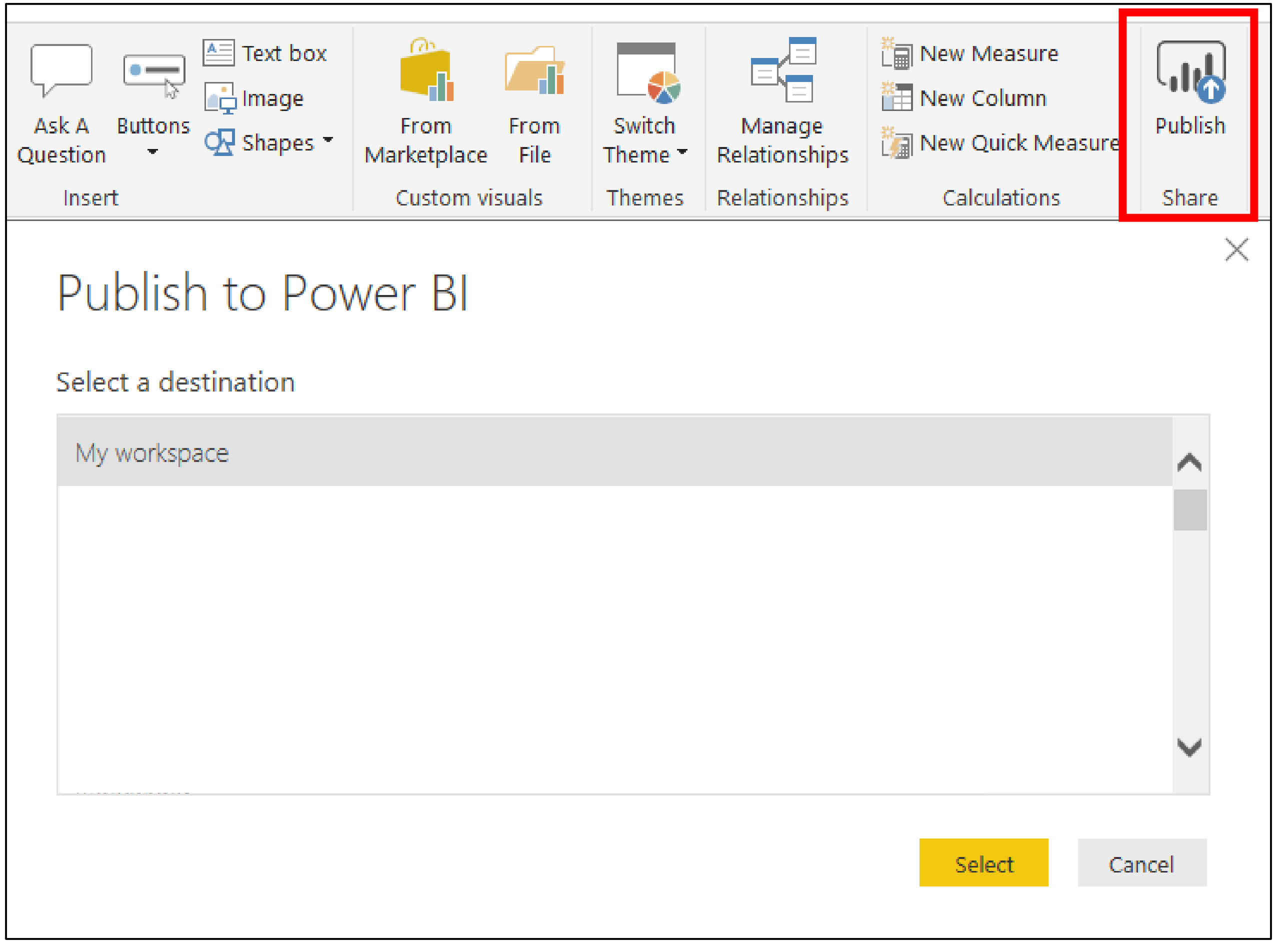
- In the model, hide all of the columns not needed for reporting. These include table key columns, columns containing nulls, or anything else you do not want to slice and dice by.

- At this point, all the relationships should be set up correctly.
  - Top Tip: In the relationship view, place the Date table at the top, the fact tables in the centre and the relationship tables below. This makes it easier to follow the topology.



Architecting a tabular model

- With all of the work on the model completed, publish the model to the Power BI Service, to the appropriate workspace.



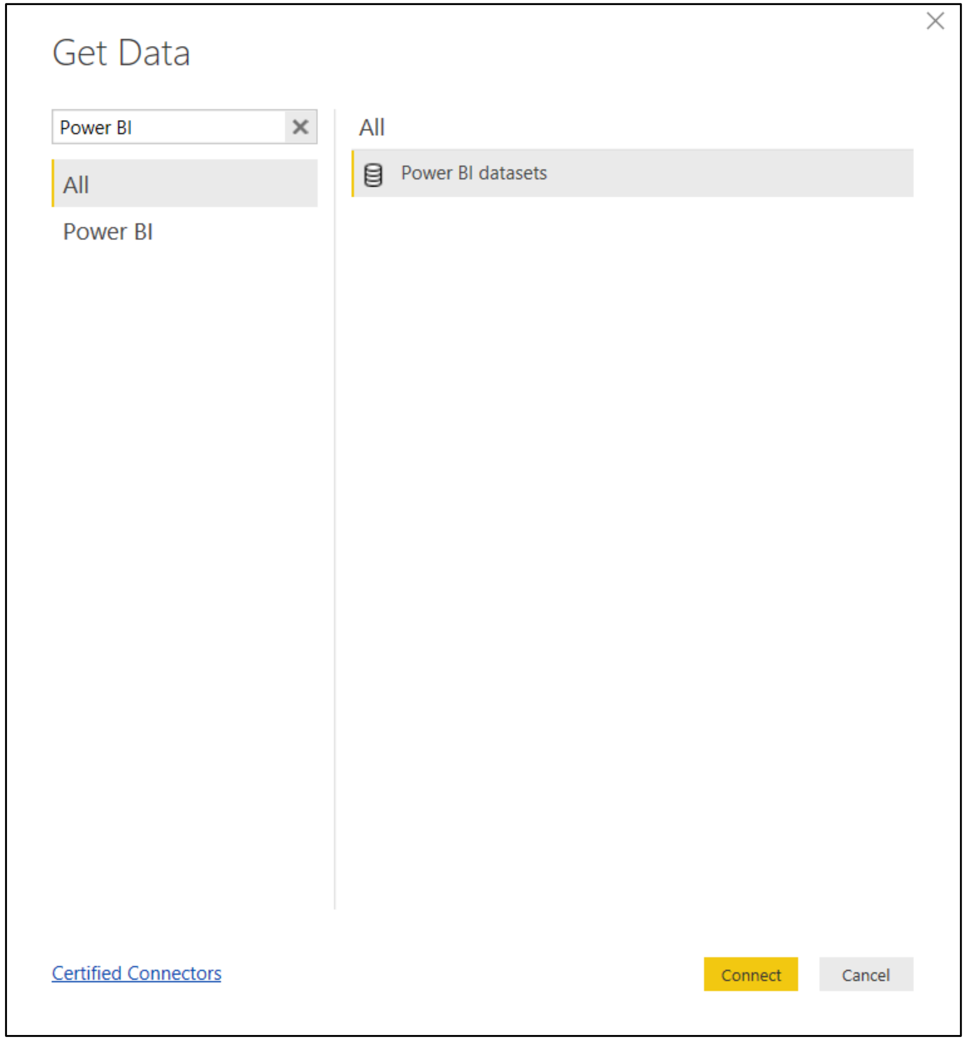
Publish to Power BI

Select a destination

My workspace

Connecting to a Power BI model

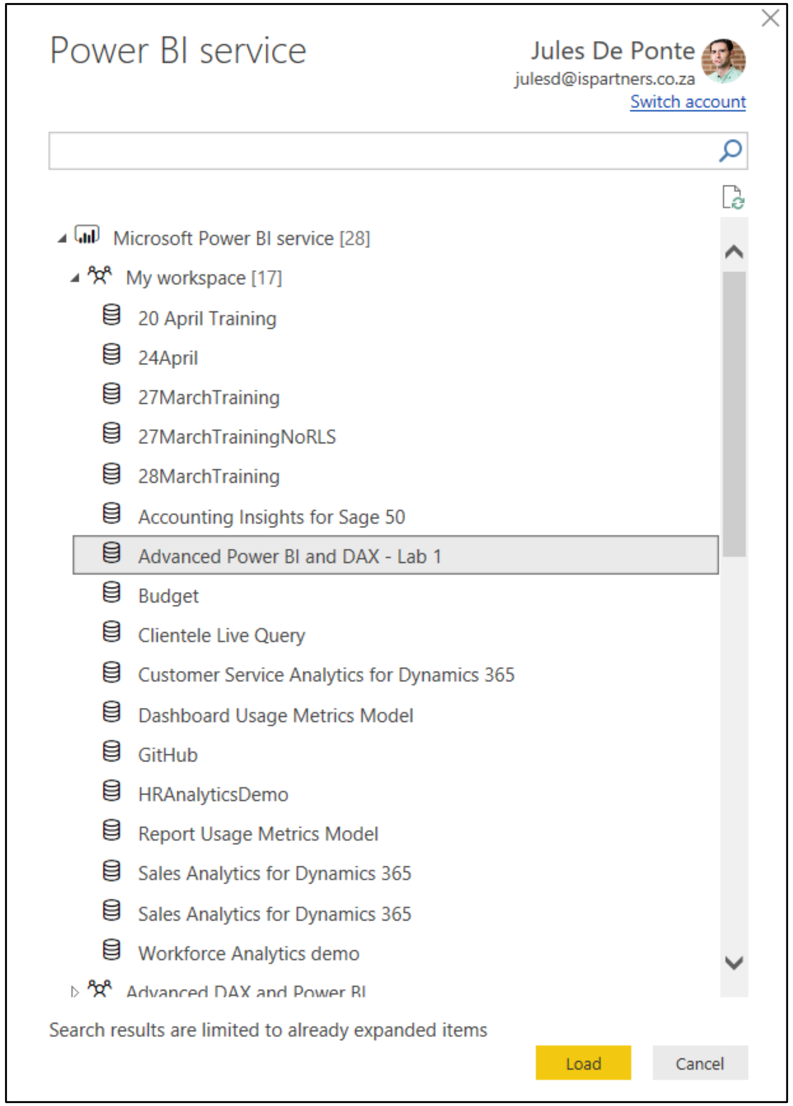
- In a new instance of Power BI Get Data -> Power BI datasets



Get Data

Connecting to a Power BI model

- In the following window, all of the workspaces to which you have access are shown.
- Select the workspace and dataset to which you want to connect.
- This creates a live connection to the Power BI Service. All of the relationships are defined and cannot be changed. You are also unable to add calculated columns. You can add measures.



Power BI service

Jules De Ponte
julesd@ispartners.co.za
Switch account

- Microsoft Power BI service [28]
  - My workspace [17]
    - 20 April Training
    - 24April
    - 27MarchTraining
    - 27MarchTrainingNoRLS
    - 28MarchTraining
    - Accounting Insights for Sage 50
    - Advanced Power BI and DAX - Lab 1
    - Budget
    - Clientele Live Query
    - Customer Service Analytics for Dynamics 365
    - Dashboard Usage Metrics Model
    - GitHub
    - HRAnalyticsDemo
    - Report Usage Metrics Model
    - Sales Analytics for Dynamics 365
    - Sales Analytics for Dynamics 365
    - Workforce Analytics demo
  - Advanced DAX and Power BI

Search results are limited to already expanded items

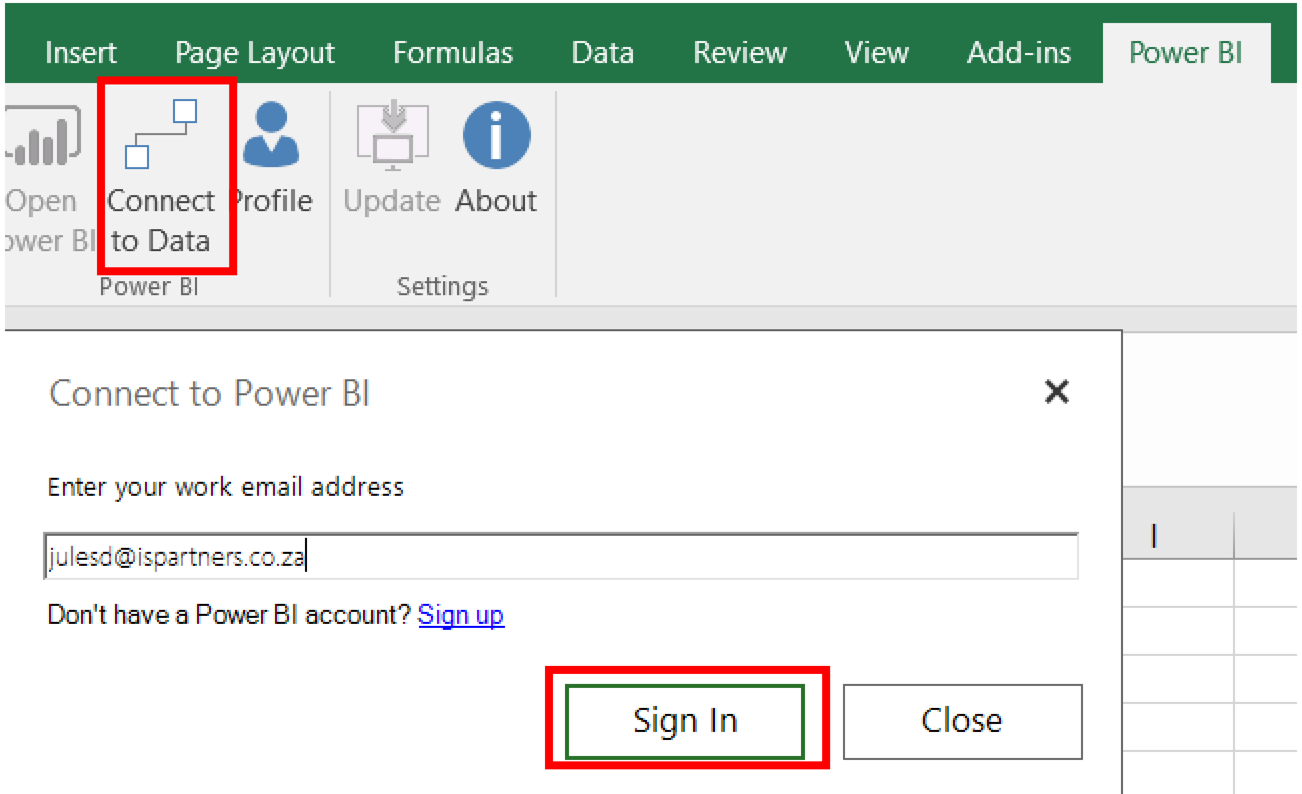
Connecting to a Power BI model in Excel

ALTRON | KARABINA

- Download the Power BI Publisher for Excel

- https://powerbi.microsoft.com/en-us/excel-dashboard-publisher/

- Install the application.
- Connect to Data
- Sign in using Office 365 Credentials.



Connect to Power BI

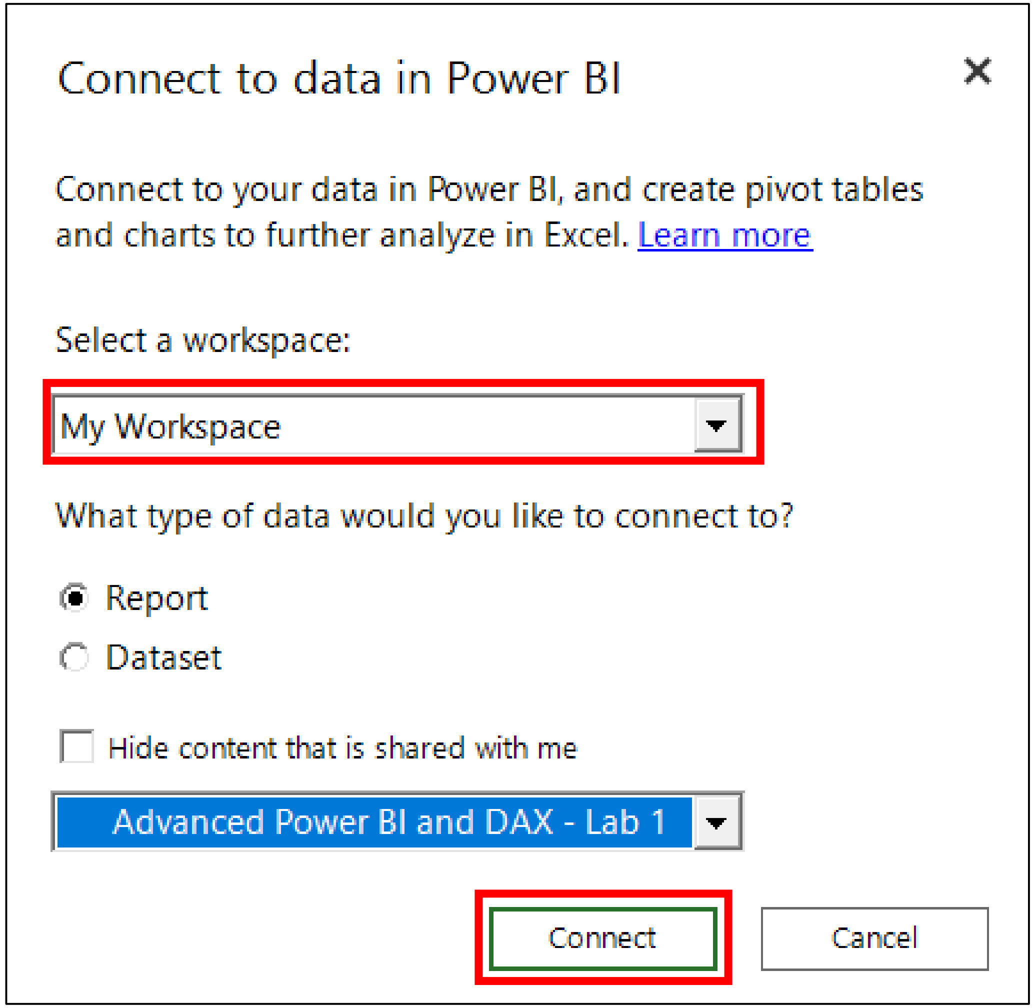
Enter your work email address

julesd@ispartners.co.za

Don't have a Power BI account? Sign up

Connecting to a Power BI model in Excel

- In the following window, select the workspace and dataset. You can now analyse the dataset in Excel.



Connect to data in Power BI

Connect to your data in Power BI, and create pivot tables and charts to further analyze in Excel. Learn more

Select a workspace: My Workspace

What type of data would you like to connect to?

● Report

○ Dataset

□ Hide content that is shared with me

Advanced Power BI and DAX - Lab 1