Power BI training

Power BI

Turn your data into Impact!

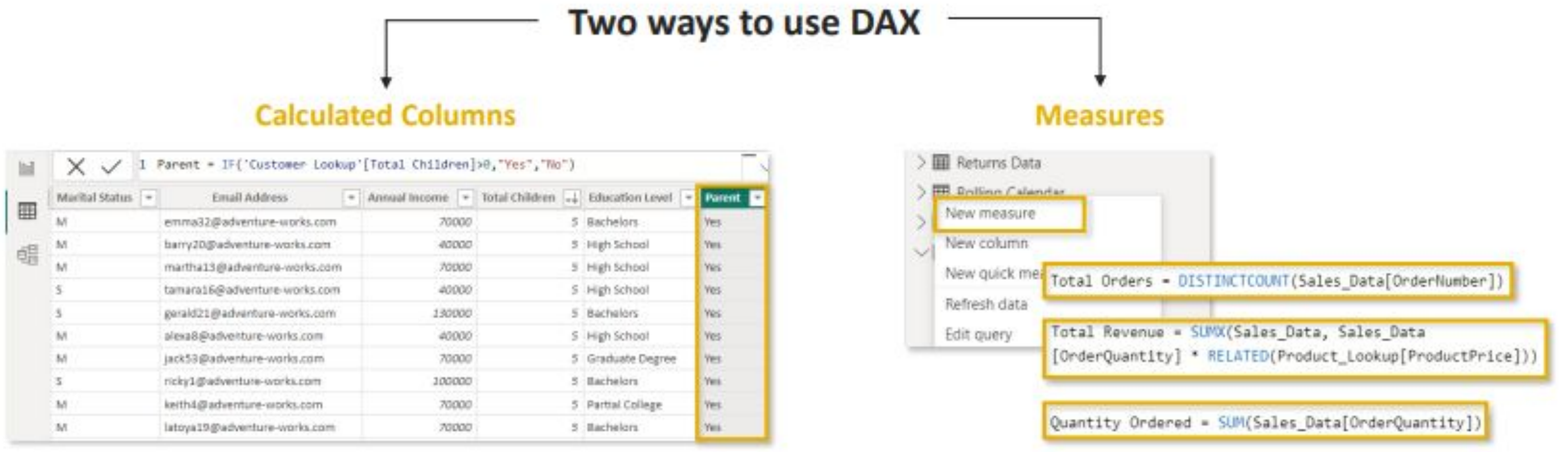
CALCULATED FIELD WITH DAX

Power BI

Power BI

DAX - DATA ANALYSIS EXPRESSION

- Formula language that drives the Power BI
- With DAX one can add - **Calculated Columns** (for filtering) and **Measures** (for aggregation) to enhance analytical capability of your data model
- Powerful and flexible functions built specifically to work with relational data models



```
1 Parent = IF('Customer Lookup'[Total Children]>0,"Yes","No")
```

Marital Status	Email Address	Annual Income	Total Children	Education Level	Parent
M	emma32@adventure-works.com	70000	5	Bachelors	Yes
M	barry20@adventure-works.com	40000	5	High School	Yes
M	martha13@adventure-works.com	70000	5	High School	Yes
S	tamara16@adventure-works.com	40000	5	High School	Yes
S	gerald21@adventure-works.com	130000	5	Bachelors	Yes
M	alexa8@adventure-works.com	40000	5	High School	Yes
M	jack53@adventure-works.com	70000	5	Graduate Degree	Yes
S	ricky1@adventure-works.com	100000	5	Bachelors	Yes
M	keith4@adventure-works.com	70000	5	Partial College	Yes
M	latoya19@adventure-works.com	70000	5	Bachelors	Yes

```
Total Orders = DISTINCTCOUNT(Sales_Data[OrderNumber])
```

```
Total Revenue = SUMX(Sales_Data, Sales_Data
[OrderQuantity] * RELATED(Product_Lookup[ProductPrice]))
```

```
Quantity Ordered = SUM(Sales_Data[OrderQuantity])
```

DAX - CALCULATED COLUMNS

Calculated Column allows you to add new, 'formula based columns' to the tables

- Calculated Columns as the name suggest refers to **creating a new column** (or entire table in case you generate more than one new columns)
- **Generate values for each row**. These values are added and are visible as a part of the tables in the Data View
- They understand the **'Row context'** i.e. they can see the information contained in each row are **great at defining properties based on information in each row**, but generally not useful for aggregated (SUM, AVG, COUNT, MIN, MIX etc) fields

As a thumb rule, use Calculated Columns for when you want to create a field having a value in every row in the table. They are typically used for 'filtering' and 'grouping' data.

DO NOT use Calculated Columns for aggregation formulas or to calculate fields for the Values area of the visualizations. We use measures for this.

DAX - MEASURES

DAX formulas that are used to generate new calculated values.

- Measures also reference to entire tables or columns i.e. they are also calculated from using entire column or table
- Unlike Calculated Columns, **Measure values are not visible within the tables** in the Data View, They can only be seen within the visualization. **They are similar to 'Calculated Fields' in an excel pivot**
- They are evaluated as per the '**Filter context'** i.e. they are recalculated when the fields or filters around them change (i.e. just like it happens in Pivot where you pull new row or column into the table or apply new filters to the report)

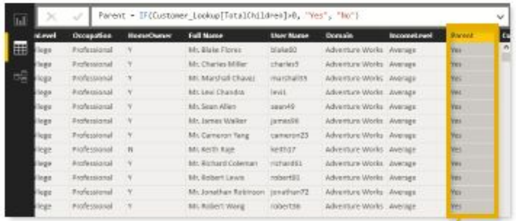
As a thumb rule, use Measures when a single row cannot give you the answer i.e. when you need to apply aggregation use Measures.

Use Measures to create numerical, calculated values that can be analysed in the 'Values' field of the report.

Power BI

DAX - CALCULATED COLUMNS vs MEASURES

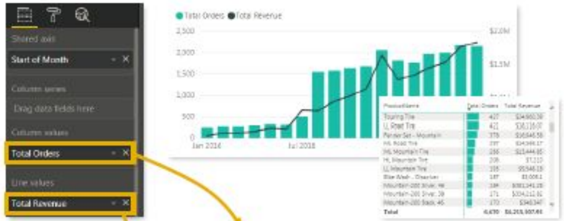
Calculated Column	Measures
Follow Row Context. Values are calculated based on the information on each row of the table	**Follow Filter Context.** Values are calculated based on any filter in the report.
Increase file size. Append calculated value to each row in the table and stores them in the model	**Does not increase file size.** Does not create new data in the table itself.
Recalculates on data source refresh or when changes are made to the component columns.	**Recalculate in response to any change to filter** within the report.
Primarily used for **filtering data** in the reports	Primarily used for **aggregating values** in the visuals



```
Parent = IF(Customer_Lookup[TotalChildren]>0, "Yes", "No")
```

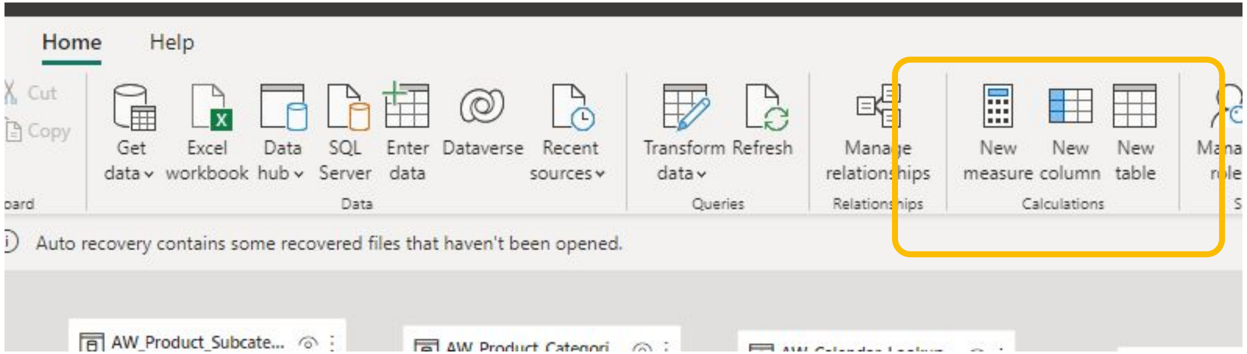
...Level	Occupation	HomeOwner	Full Name	User Name	Domain	IncomeLevel	Parent
...llege	Professional	Y	Mr. Blake Flores	blake80	Adventure Works	Average	Yes
...llege	Professional	Y	Mr. Charles Miller	charles5	Adventure Works	Average	Yes
...llege	Professional	Y	Mr. Marshall Chavez	marshall35	Adventure Works	Average	Yes
...llege	Professional	Y	Mr. Levi Chandra	levi1	Adventure Works	Average	Yes
...llege	Professional	Y	Mr. Sean Allen	sean49	Adventure Works	Average	Yes
...llege	Professional	Y	Mr. James Walker	james98	Adventure Works	Average	Yes
...llege	Professional	Y	Mr. Cameron Yang	cameron23	Adventure Works	Average	Yes
...llege	Professional	N	Mr. Keith Raje	keith17	Adventure Works	Average	Yes
...llege	Professional	Y	Mr. Richard Coleman	richard61	Adventure Works	Average	Yes
...llege	Professional	Y	Mr. Robert Lewis	robert81	Adventure Works	Average	Yes
...llege	Professional	Y	Mr. Jonathan Robinson	jonathan72	Adventure Works	Average	Yes
...llege	Professional	Y	Mr. Robert Wang	robert38	Adventure Works	Average	Yes

Calculated columns "live" in tables

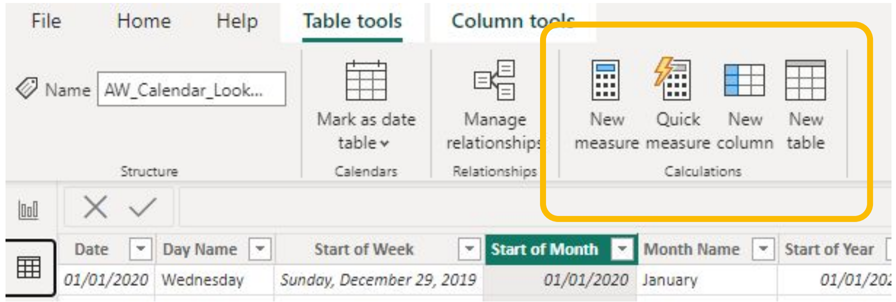


Measures "live" in visuals

ADDING MEASURES AND CALCULATED COLUMNS



Auto recovery contains some recovered files that haven't been opened.

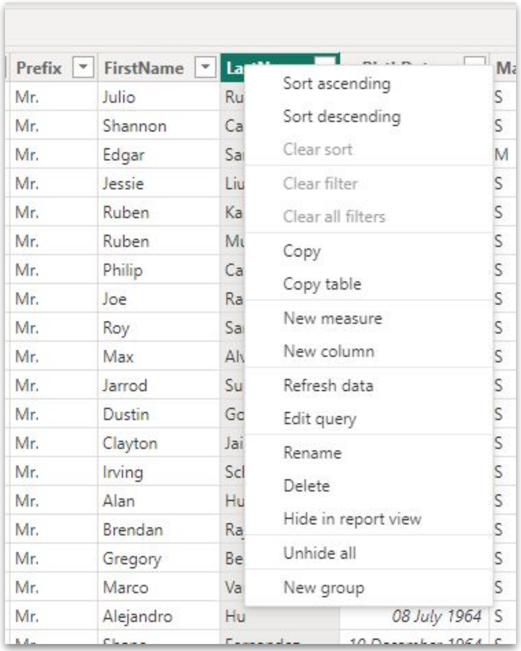


Date	Day Name	Start of Week	Start of Month	Month Name	Start of Year
01/01/2020	Wednesday	Sunday, December 29, 2019	01/01/2020	January	01/01/202

When you insert a Column or Measure from the Home tab they are assigned to whichever table is currently selected (or the first table in the field list by default).

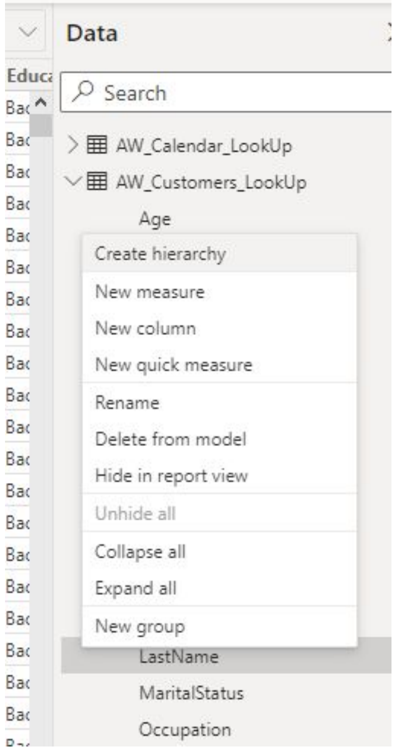
Can be reassigned to new Home Tables

ADDING MEASURES AND CALCULATED COLUMNS



Prefix	FirstName	La		Ma
Mr.	Julio	Ru		S
Mr.	Shannon	Ca		S
Mr.	Edgar	Sa		M
Mr.	Jessie	Liu		S
Mr.	Ruben	Ka		S
Mr.	Ruben	Mu		S
Mr.	Philip	Ca		S
Mr.	Joe	Ra		S
Mr.	Roy	Sa		S
Mr.	Max	Alv		S
Mr.	Jarrod	Su		S
Mr.	Dustin	Go		S
Mr.	Clayton	Jai		S
Mr.	Irving	Scl		S
Mr.	Alan	Hu		S
Mr.	Brendan	Ra		S
Mr.	Gregory	Be		S
Mr.	Marco	Va		S
Mr.	Alejandro	Hu	08 July 1964	S

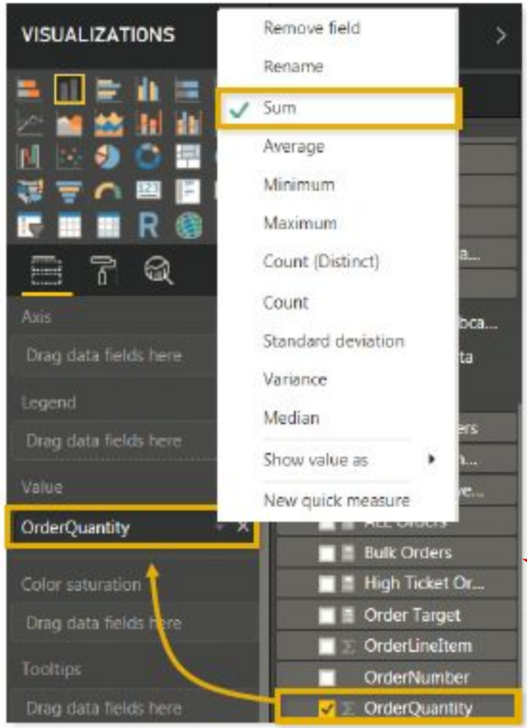
Right click on the field you want to use to create a Measure or Calculated Column.



Right click on the field you want to use in the Data pane to create a Measure or Calculated Column.

Note: When you insert a Column or Measure from the Home tab they are assigned to whichever table is currently selected (or the first table in the field list by default). Can be reassigned to new Home Tables using Properties tab (in the Data Model section).

IMPLICIT AND EXPLICIT MEASURE



Example of an implicit measure

Implicit Measures - are created when you drag a raw numerical fields (eg 'Order Quantity') into Values pane of a visual. They add aggregation (Sum, Average etc.) automatically or manually

★ **Important** - They are only **accessible within the specific visualization** in which it is created and not be used anywhere else.

Explicit Measures - are created by actually entering DAX Functions (or adding 'Quick Measures') to define Calculated Columns or measures.

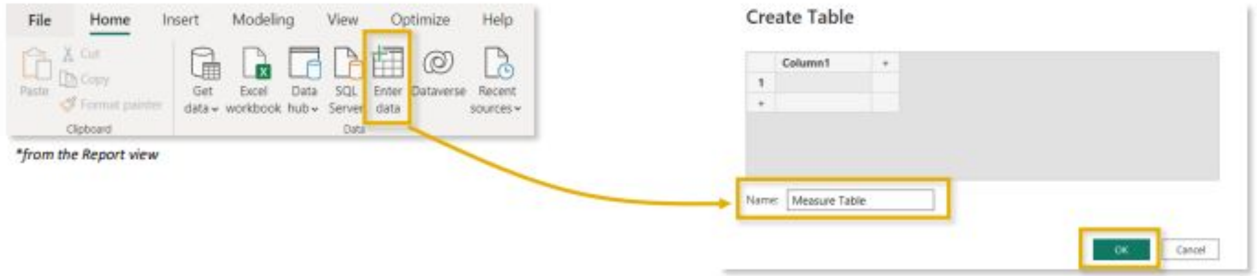
★ **Important** - They **can be used anywhere** else in the report and referenced with other DAX calculations

Power BI

MEASURES TABLES

It's a **best practice to create a dedicated table to store all your Measures.** This will help you stay organized, find measures quickly, and allow you to group related measures into folder.

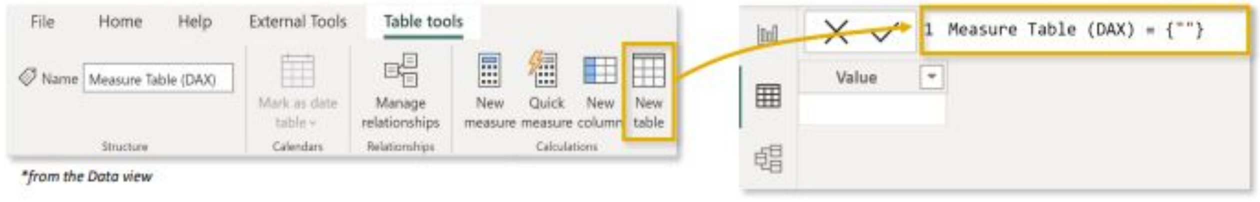
Option 1: **Enter Data** into **Power Query** (loads the table to the data model – table is visible in Power Query)



**from the Report view*

Add a table Name and **click OK** to load the table to the data model

Option 2: Create a **calculated table** using **DAX** directly in the model (table is not visible in Power Query)



```
1  Measure Table (DAX) = {""}
```

**from the Data view*

Create new table & use a table constructor { } to **add a single column**

Power BI

EXERCISE ON MEASURES TABLES

Create a 'Measure Table' using any option.

First Option

- In the Data View or Model View - click on Table Tools > New Table
- Write Calculation as - Measure Table (DAX) = {" "}

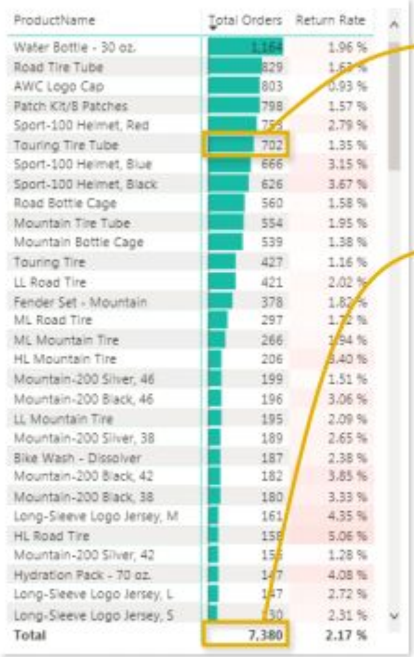
Second Option

- In the Reports View - Go to Home > Enter Data
- Change the Table Name as - Measure Table (DAX)

Note: Make sure to store all the Measures you create in the Measures Table for better accessibility. You can also move the Measures from one Table to another using Properties pane or by changing the 'Home Table' on the Menu by clicking on the Measure.

UNDERSTAND FILTER CONTEXT

Remember that measures are evaluated based on **filter context**, which means that they recalculate whenever the fields or filters around them change



ProductName	Total Orders	Return Rate
Water Bottle - 30 oz.	1,164	1.96 %
Road Tire Tube	829	1.67 %
AWC Logo Cap	803	0.93 %
Patch Kit/8 Patches	798	1.57 %
Sport-100 Helmet, Red	755	2.79 %
Touring Tire Tube	702	1.35 %
Sport-100 Helmet, Blue	666	3.15 %
Sport-100 Helmet, Black	626	3.67 %
Road Bottle Cage	560	1.58 %
Mountain Tire Tube	554	1.95 %
Mountain Bottle Cage	539	1.38 %
Touring Tire	427	1.16 %
LL Road Tire	421	2.02 %
Fender Set - Mountain	378	1.82 %
ML Road Tire	297	1.72 %
ML Mountain Tire	266	1.94 %
HL Mountain Tire	206	3.40 %
Mountain-200 Silver, 46	199	1.51 %
Mountain-200 Black, 46	196	3.06 %
LL Mountain Tire	195	2.09 %
Mountain-200 Silver, 38	189	2.65 %
Bike Wash - Dissolver	187	2.38 %
Mountain-200 Black, 42	182	3.85 %
Mountain-200 Black, 38	180	3.33 %
Long-Sleeve Logo Jersey, M	161	4.35 %
HL Road Tire	158	5.06 %
Mountain-200 Silver, 42	153	1.28 %
Hydration Pack - 70 oz.	147	4.08 %
Long-Sleeve Logo Jersey, L	147	2.72 %
Long-Sleeve Logo Jersey, S	130	2.31 %
Total	**7,380**	**2.17 %**

For this particular value in the matrix, the **Total Orders** measure is calculated based on the following filter context: *Products[**ProductName**] = "**Touring Tire Tube**"*

- This allows the measure to return the total order quantity for each product specifically (or whatever the row and column labels dictate – *years, countries, product categories, customer names*, etc)

This Total is ***not*** calculated by summing the values above; it evaluates as its own measure, with **no filter context** (*since we aren't calculating orders for a specific product*)

Each measure value in a report is like an island, and calculates according to its own filter context (even Totals and Grand Totals)

Power BI

UNDERSTAND FILTER CONTEXT

Power BI

MEASURE: Total Revenue

FILTER CONTEXT:

- *Calendar[Year]* ▪ 2016 or 2017
- *Customers[Full Name]* ▪ **Mr. Larry Munoz**

MEASURE: Total Orders

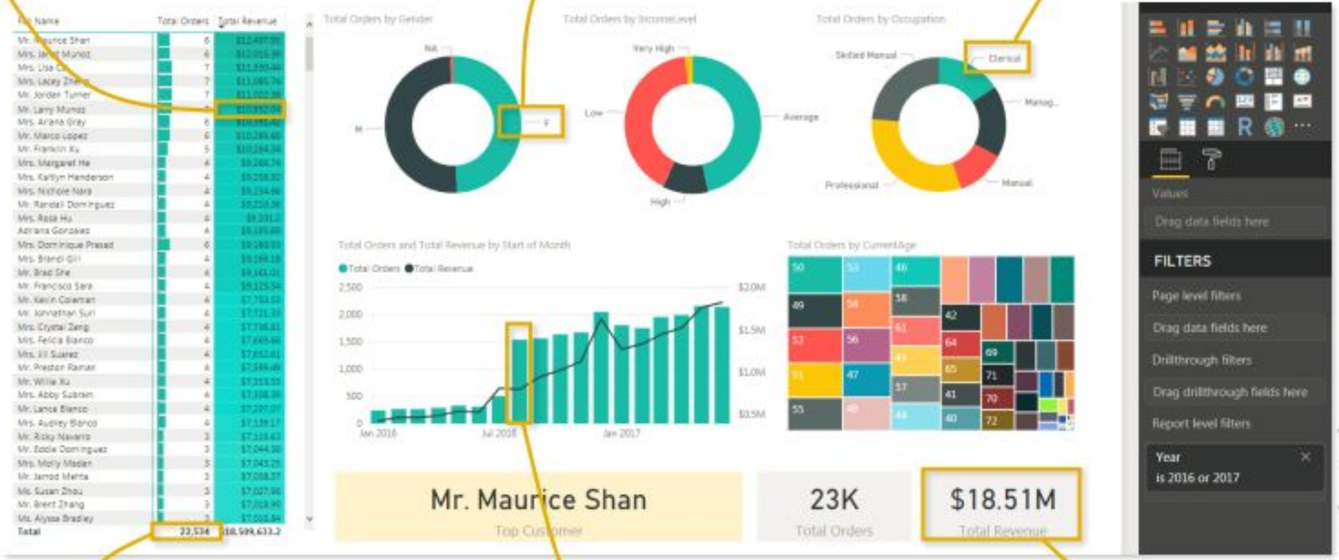
FILTER CONTEXT:

- *Calendar[Year]* ▪ 2016 or 2017
- *Customers[Gender]* ▪ F (Female)

MEASURE: Total Orders

FILTER CONTEXT:

- *Calendar[Year]* ▪ 2016 or 2017
- *Customers[Occupation]* ▪ Clerical



Full Name	Total Orders	Total Revenue
Mr. Maurice Shan	6	$12,407.98
Mrs. Janet Munoz	6	$12,015.39
Mrs. Lisa Cai	7	$11,330.44
Mrs. Lacey Zheng	7	$11,085.74
Mr. Jordan Turner	7	$11,002.38
Mr. Larry Munoz	7	$10,852.04
Mrs. Ariana Gray	6	$10,591.42
Mr. Marco Lopez	6	$10,289.60
Mr. Franklin Xu	5	$10,264.34
Mrs. Margaret He	4	$9,268.74
Mrs. Kaitlyn Henderson	4	$9,258.32
Mrs. Nichole Nara	4	$9,234.66
Mr. Randall Dominguez	4	$9,218.36
Mrs. Rosa Hu	4	$9,201.2
Adriana Gonzalez	4	$9,195.88
Mrs. Dominique Prasad	6	$9,180.03
Mrs. Brandi Gill	4	$9,166.18
Mr. Brad She	4	$9,141.01
Mr. Francisco Sara	4	$9,125.54
Mr. Kevin Coleman	4	$7,750.53
Mr. Jonathan Suri	4	$7,721.38
Mrs. Crystal Zeng	4	$7,706.81
Mrs. Felicia Blanco	4	$7,669.66
Mrs. Jill Suarez	4	$7,652.81
Mr. Preston Raman	4	$7,599.49
Mr. Willie Xu	4	$7,353.53
Mrs. Abby Subram	4	$7,308.38
Mr. Lance Blanco	4	$7,297.07
Mrs. Audrey Blanco	4	$7,139.17
Mr. Ricky Navarro	3	$7,119.63
Mr. Eddie Dominguez	3	$7,044.38
Mrs. Molly Madan	3	$7,043.25
Mr. Jarrod Mehta	3	$7,008.37
Ms. Susan Zhou	3	$7,007.98
Mr. Brent Zhang	3	$7,003.99
Ms. Alyssa Bradley	3	$7,000.84
Total	22,534	$18,509,633.2

*This is a **page-level filter**, which impact **ALL** visuals on the report page*

MEASURE: Total Orders

FILTER CONTEXT:

- *Calendar[Year]* ▪ 2016 or 2017

MEASURE: Total Orders

FILTER CONTEXT:

- *Calendar[Year]* ▪ 2016 or 2017
- *Calendar[Month]* ▪ August 2016

MEASURE: Total Revenue

FILTER CONTEXT:

- *Calendar[Year]* ▪ 2016 or 2017

EXERCISE ON DAX - CALCULATED COLUMN

1. Add a column ‘Dayoftheweek’ in ‘FactTable’. Assume week starts on Monday.

   Use WEEKDAY function.

2. Add a column ‘DateOfBirth’ in ‘Patient_Lookup’. Use DATE function.

3. Now extract the month from the date and add Column MonthName -

   Use MonthName = Patient_LookUp[DateOfBirth].[Month]

4. Calculate the age of the Patient in a separate column using function DATEDIFF.

Power BI

EXERCISE ON DAX - CALCULATED COLUMN

1. Create a calculated column in the 'Patients_LookUp' table that concatenates the provider's first name and last name.(**Hint**: Use the 'Patients_LookUp' table fields "FirstName" and "LastName").
2. Create a column 'CPTUnitConsumed' in FactTable that will show us if the quantity order was - "Single Unit" or "Multiple Units".
3. Create a calculated column 'Total_Payments' in the 'FactTable' that calculates the total payments as Insurance_Payment + Patient_Payment for each payer. (**Hint**: Use '+' symbol to add both). Try using SUM aggregation also and see the difference.
4. Create a calculated column in the 'Date' table that extracts the day of the week from the 'Date' field. **Hint:** Use the DAX function WEEKDAY().

Power BI

EXERCISE ON DAX - IMPLICIT AND EXPLICIT MEASURE

Creating an Implicit Measure

1. In the Reports View Click on the Matrix template. Bring in 'HospitalName' to rows
2. Bring in 'Insurance_Payment' (from FactTable) into the Matrix template.
3. Also bring in 'Patient_Payment' (from FactTable) into the Matrix template.

Creating an Explicit Measure

1. Right click on the 'FactTable' and click on - 'New Measure'
2. Name the Measure as 'Total_InsurancePay'. Use SUM as aggregation and bring in Insurance_Payment' to create the Measure. (**Hint:** Total_InsurancePay = SUM(FactTable[Insurance_Payment])).
3. Similarly create another Measure for 'Total_PatientPay'.
4. **Use it in the above Matrix visualization and check the results.**

Power BI

UNDERSTAND THE MECHANICS OF MEASURES (FROM ADVENTURE WORKS)

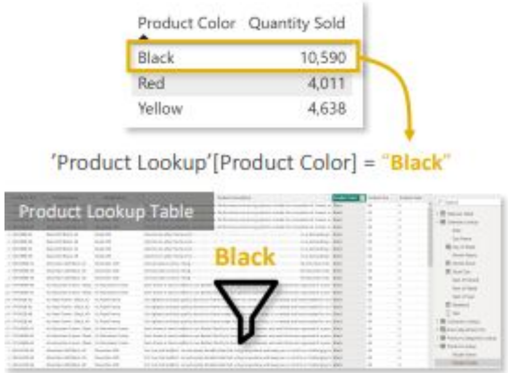
Product Color	Quantity Sold
Black	10,590
Red	4,011
Yellow	4,638

How *exactly* is this measure value calculated?

- **NOTE**: This all happens *instantly* behind the scenes, every time the filter context changes

STEP 1

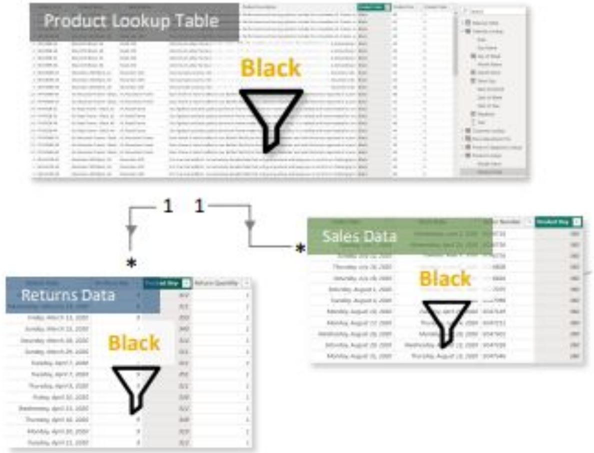
Filter context is detected & applied



'Product Lookup'[Product Color] = "Black"

STEP 2

Filters flow "downstream" to related tables



STEP 3

Measure evaluates against the filtered table

```
Quantity Sold =
SUM(
    'Sales Data'[Order Quantity]
)
```

Sum of values in the ***Order Quantity*** *column of the* ***Sales Data*** *table, filtered to rows where the product color is "Black"*

= 10,590

UNDERSTAND DAX SYNTAX

MEASURE NAME

- ***Note:*** *Measures are always surrounded in brackets (i.e.* ***[Total Quantity]****) when referenced in formulas, so spaces are OK*

Referenced
TABLE NAME

Referenced
COLUMN NAME

Total Quantity: =SUM(Transactions[quantity])

FUNCTION NAME

- Calculated columns don't always use functions, but measures do:
  - In a **Calculated Column,** =Transactions[quantity] returns the value from the quantity column in each row (*since it evaluates one row at a time*)
  - In a **Measure,** =Transactions[quantity] will return an ***error*** since Power BI doesn't know how to translate that as a single value (*you need some sort of aggregation*)

Note: *This is a "****fully qualified****" column, since it's preceeded by the table name -- table names with spaces must be surrounded by* ***single quotes:***

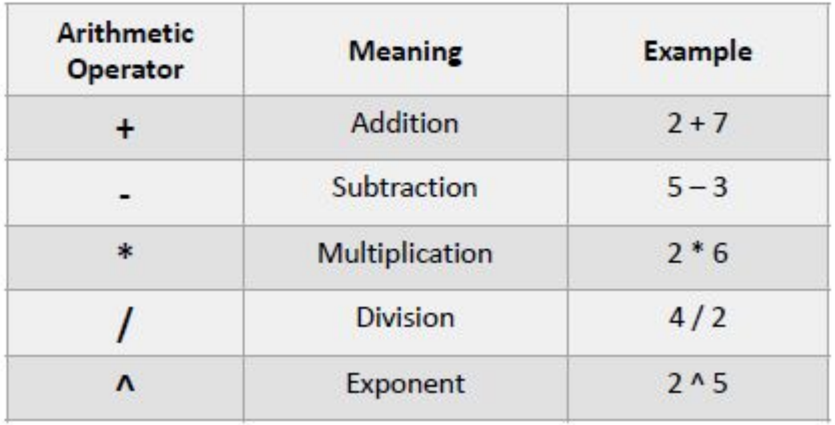
- *Without a space:* ***Transactions****[quantity]*
- *With a space:* ***'Transactions Table'****[quantity]*

Best Practice - For column references, use the fully qualified name (i.e.Table[Column]).

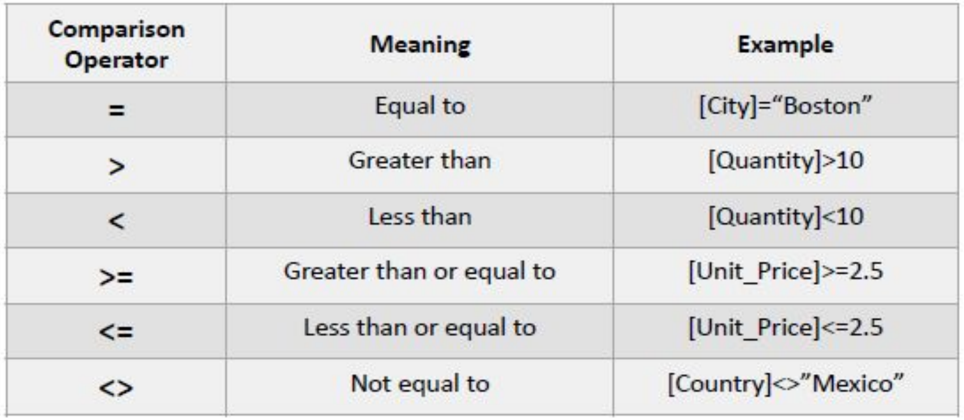
For Measures, just use Measure name

Power BI

DAX OPERATORS

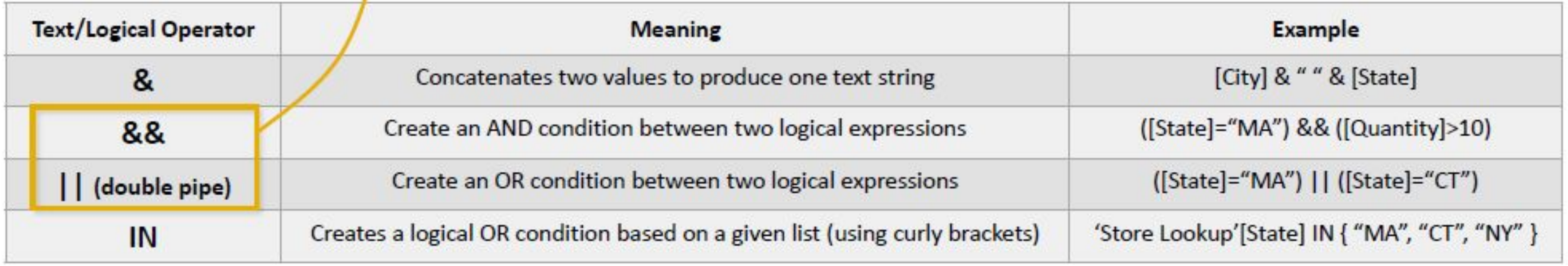


Arithmetic Operator	Meaning	Example
+	Addition	2 + 7
-	Subtraction	5 – 3
*	Multiplication	2 * 6
/	Division	4 / 2
^	Exponent	2 ^ 5



Comparison Operator	Meaning	Example
=	Equal to	[City]="Boston"
>	Greater than	[Quantity]>10
<	Less than	[Quantity]<10
>=	Greater than or equal to	[Unit_Price]>=2.5
<=	Less than or equal to	[Unit_Price]<=2.5
<>	Not equal to	[Country]<>"Mexico"

These are important



Text/Logical Operator	Meaning	Example
&	Concatenates two values to produce one text string	[City] & " " & [State]
&&	Create an AND condition between two logical expressions	([State]="MA") && ([Quantity]>10)
|| (double pipe)	Create an OR condition between two logical expressions	([State]="MA") || ([State]="CT")
IN	Creates a logical OR condition based on a given list (using curly brackets)	'Store Lookup'[State] IN { "MA", "CT", "NY" }

DAX FUNCTIONS - IMPORTANT NOTE

Power BI

Note that DAX has more than 250 functions. The plan for this section is to discuss the concepts of DAX and the most used Function types that will help you get *'up and running'* with DAX. It will also help you build a solid foundation, capability and the confidence to handle and learn any new function as and when you come across faster.

For comprehensive list of DAX Functions - https://learn.microsoft.com/en-us/dax/

MEASURE FUNCTIONS -BASIC MATHS AND STATS FUNCTIONS

SUM()	*Evaluates the sum of a column*	=SUM(ColumnName)
AVERAGE()	*Returns the average (arithmetic mean) of all the numbers in a column*	=AVERAGE(ColumnName)
MAX()	*Returns the largest value in a column or between two scalar expressions*	=MAX(ColumnName) *or* =MAX(Scalar1, *[Scalar2]*)
MIN()	*Returns the smallest value in a column or between two scalar expressions*	=MIN(ColumnName) *or* =MIN(Scalar1, *[Scalar2]*)
DIVIDE()	*Performs division and returns the alternate result (or blank) if div/0*	=DIVIDE(Numerator, Denominator, *[AlternateResult]*)

Power BI

EXERCISE ON DAX MEASURES - MATHS FUNCTIONS

1. Add a field ‘Average_InsuancePay’ using AVERAGE(FactTable[Insurance_Payment]).
2. Add a field ‘Average_PatientPay’ using AVERAGE(FactTable[Patient_Payment]).
3. Create another Measure ‘Total_Payment’. Use the Measures ‘Total_PatientPay’ and ‘Total_InsurancePay’.
4. Create a Matrix to view Total_InsurancePay, Average_InsurancePay, Total_PatientPay, Average_PatientPay and Total_Payment for - a) Hospitals, b) Payer
5. Create a Measure - ‘%InsurancePay’. Use formula as below - %InsurancePay =[Total_InsurancePayment])/[Total_Payment]

Note: Make sure to add all the Measures to the Measures Table created.

Power BI

COUNT FUNCTIONS

COUNT()	*Counts the number of cells in a column that contain numbers*	=COUNT(ColumnName)
COUNTA()	*Counts the number of non-empty cells in a column (numerical and non-numerical)*	=COUNTA(ColumnName)
DISTINCTCOUNT()	*Counts the number of distinct or unique values in a column*	=DISTINCTCOUNT(ColumnName)
COUNTROWS()	*Counts the number of rows in the specified table, or a table defined by an expression*	=COUNTROWS(Table)

Power BI

EXERCISE ON MEASURES - COUNT FUNCTIONS

1. Create a variable 'CountPatient' that count the number of Male and Female Patient per blood group. **Hint:** Use COUNT(Patient_LookUp[Patient-PK]) and create a Matrix for the same. *(Note - We can use COUNT or COUTNA also by adding a Column name and we will get the same result)*

2. Create the Matrix showing the split of Male and Female for Tobacco. Also create Matrix showing the Male and Female count for Alcohol, Exercise and Diet.

3. Show the 'Distinct Patient' for different Hospitals in the Matrix format. Use DISTINCTCOUNT Function to calculate the explicit Measure and also verify the count with implicit Measure in the Matrix.

Power BI

LOGICAL FUNCTIONS

IF	Checks if a given condition is met and returns one value if the condition is TRUE, and another if the condition is FALSE	=IF(LogicalTest, ResultIfTrue, *[ResultIfFalse]*)
IFERROR	Evaluates an expression and returns a specified value if it returns an error, otherwise returns the expression itself	=IFERROR(Value, ValueIfError)
SWITCH	Evaluates an expression against a list of values and returns one of multiple possible expressions	=SWITCH(Expression, Value1, Result1, ..., *[Else]*)
AND	Checks whether both arguments are TRUE to return TRUE, otherwise returns FALSE	=AND(Logical1, Logical2)
OR	Checks whether any argument is TRUE to return TRUE, otherwise returns FALSE	=OR(Logical1, Logical2)

***Note:** Use the **&&** and **||** operators to include more than two conditions*

Power BI

EXERCISES ON DAX COLUMNS - LOGICAL

1. Create a new calculated column ('Age>50') in the 'FactTable' table that checks if the patient age is greater than 50. Mark True or False. *(**Hint**: Use IF function and the PatientAge column).*
2. Create a calculated column in the **'Patient_LookUp'** table named **'PatientType'** that returns -
   a. 'Minor if the PatientAge is <18
   b. Adult if the PatientAge is between 18 and 75
   c. Senior if PatientAge is > 75
3. Create a column in 'Patient_LookUp' table that shows the Patient who consumes Alcohol and Tobacco both (**Hint:** Use AND logical operator)
4. Create a column 'Patient_LookUp' table that shows the Patient who consumes Alcohol, Tobacco and also follows strict Diet (**Hint:** Use && logical operator)
5. Create RegionCode Column (with codes NE,MW,S,W) using SWITCH Function.

***Note -** SWITCH can only be used where you have to equate values. It can replace multiple nested IF statements but only when the conditions are of equality. If other comparison forms (<,>,>=,<=) are used then prefer nested IF.*

Power BI

EXERCISES ON DAX COLUMNS - LOGICAL

1. Create a calculated column in the ‘Patient_LookUp’ table and check for Patients who do not consume Alcohol and do not consume Tobacco.(**Hint**: Use the AND function to create the column).
2. Create a calculated column in the ‘Hospital_LookUp’ table named "IsHospitalNameLong" that returns "Yes" if the length of the "LocationName" is greater than 10 characters, and "No" otherwise.(**Hint:** Use the IF and LEN functions to calculate the length of the "LocationName" column).

Power BI

TEXT FUNCTIONS

Note: Use the & operator as a shortcut, or to combine more than two strings!

LEN()	*Returns the number of characters in a string*	=LEN(Text)
CONCATENATE()	*Joins two text strings into one*	=CONCATENATE(Text1, Text2)
LEFT/MID/ RIGHT()	*Returns a number of characters from the start/middle/end of a text string*	=LEFT/RIGHT(Text, *[NumChars]*) =MID(Text, StartPosition, NumChars)
UPPER/LOWER/ PROPER()	*Converts letters in a string to upper/lower/proper case*	=UPPER/LOWER/PROPER(Text)
SUBSTITUTE()	*Replaces an instance of existing text with new text in a string*	=SUBSTITUTE(Text, OldText, NewText, *[InstanceNumber]*)
SEARCH()	*Returns the position where a specified string or character is found, reading left to right*	=SEARCH(FindText, WithinText, *[StartPosition], [NotFoundValue]*)

Power BI

EXERCISE ON DAX COLUMNS - TEXT

1. Create a Calculated column in the 'Patient_LookUp' table to concatenate the First name and Last name of patients. (**Hint:** Use the CONCATENATE function) Also & to create the Full Name.
2. In the 'DimDate' table use LEFT function to create a Column 'ShortDayName' using only first 3 characters.
3. Create a Calculated column in the 'Factable-2' table to convert the name of the patient to uppercase. (Hint: Use the UPPER function)
4. In the 'DimDate' table use SUBSTITUTE function to remove the 'day' from the names of the days in the 'DayName'. Create a new column.
5. Create a Calculated column in the 'Dimspeciality' table to extract the first two characters of the speciality code. (**Hint**: Use the LEFT function)
6. Create a Calculated column in the 'Dimpatients' table to concatenate the first name and last name of patients. (**Hint:** Use the CONCATENATE function)

Power BI

DATE AND TIME FUNCTIONS

DAY/MONTH/ YEAR()	*Returns the day of the month (1-31), month of the year (1-12), or year of a given date*	=DAY/MONTH/YEAR(Date)
HOUR/MINUTE/ SECOND()	*Returns the hour (0-23), minute (0-59), or second (0-59) of a given datetime value*	=HOUR/MINUTE/SECOND(Datetime)
TODAY/NOW()	*Returns the current date or exact time*	=TODAY/NOW()
WEEKDAY/ WEEKNUM()	*Returns a weekday number from 1 (Sunday) to 7 (Saturday), or the week # of the year*	=WEEKDAY/WEEKNUM(Date, *[ReturnType]*)
EOMONTH()	*Returns the date of the last day of the month, +/- a specified number of months*	=EOMONTH(StartDate, Months)
DATEDIFF()	*Returns the difference between two dates, based on a selected interval*	=DATEDIFF(Date1, Date2, Interval)

Power BI

EXERCISE ON DAX COLUMNS - TIME BASED

1. Create a Calculated column ‘DayOfWeek’ in the ‘DimDate’ table to determine the day of the week. (**Hint**: Use the DAX function WEEKDAY() and use number count starting from Monday i.e. 1 through 7 for Monday through Sunday)
2. Create a new Column ‘Weekend’. Fill ‘Weekend’ if days are 6 or 7 and ‘Weekday’.
3. Create a Calculated column in the dimdate table to determine the week number of the year. (**Hint**: Use the DAX function "WEEKNUM()" to calculate the week number based on the Date column).
4. Create a Calculated column in the ‘DimDate’ table to extract the year from the "Date" column.(**Hint**: Use the YEAR() function).
5. Create a Calculated column in the ‘DimDate’ table to determine the week number of the year.(**Hint**: Use the DAX function "WEEKNUM()" to calculate the week number based on the Date column).
6. Create a Calculated column in the ‘DimDate’ table to extract the month from the "Date" column.(**Hint**: Use the MONTH() function).

Power BI

RELATED DATA FUNCTIONS

RELATED() *Returns related values in each row of a table based on relationships with other tables*

- **RELATED()** works exactly like a **VLOOKUP function in excel**. It uses the relationships between tables (defined by Primary and Foreign Keys) to pull values from one table into the new column of another
- As it requires row context, it can only be used as a Calculated Column.
- **Note that RELATED also creates redundant data which is duplicated and against the concept of Normalization. Hence use it only when needed.**

=RELATED(ColumnName)

The column that contains the values you want to retrieve

Examples:

- *Product_Lookup[ProductName]*
- *Territory_Lookup[Country]*

Power BI

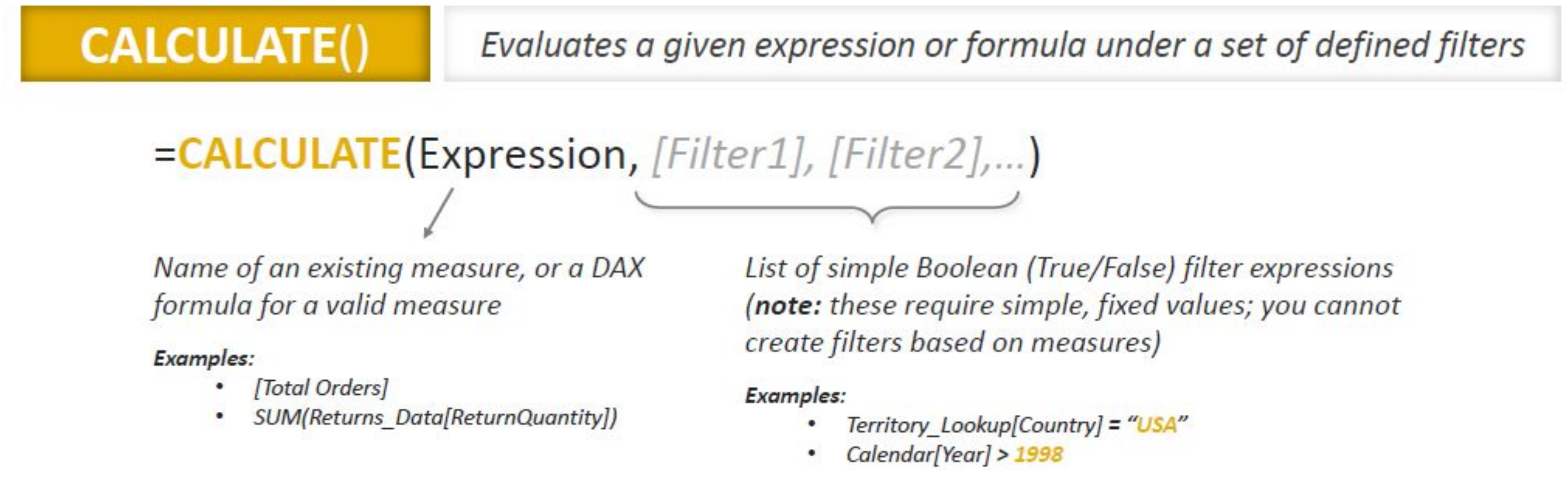
EXERCISE ON DAX - RELATED FUNCTION

1. Use RELATED function to show the 'PatientCity' and PatientGender' into FactTable.

Power BI

Power BI

ADVANCE DAX - CALCULATE FUNCTION



CALCULATE()

Evaluates a given expression or formula under a set of defined filters

=CALCULATE(Expression, *[Filter1], [Filter2],…*)

Name of an existing measure, or a DAX formula for a valid measure

Examples:

- *[Total Orders]*
- *SUM(Returns_Data[ReturnQuantity])*

*List of simple Boolean (True/False) filter expressions (**note:** these require simple, fixed values; you cannot create filters based on measures)*

Examples:

- *Territory_Lookup[Country]* **=** *"USA"*
- *Calendar[Year]* **>** *1998*

Calculate works like SUMIF, COUNTIF. AVERAGEIF in excel. It can evaluate measures based on any sort of calculation (not just SUM, COUNT etc. You can think it like "CALCULATEIF". Also remember that CALCULATE overrules the filter context.

EXERCISE ON MEASURES - CALCULATE FUNCTION

1. Create a field ‘Insurance_Tobacco’ which provides the [Total_InsuancePay] (Measure calculated as SUM(FactTable[Insuance_Payment])) only for patients consuming Tobacco. Show this in the Hospital Matrix.

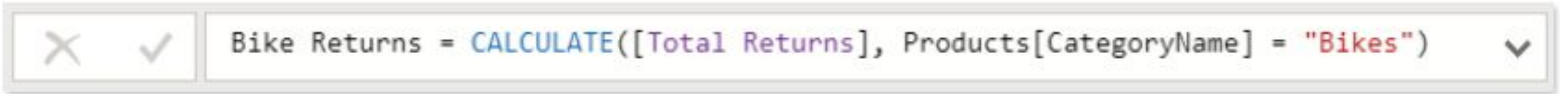
   Hint: CALCULATE([Total_InsurancePay],Patient_LookUp[Tobacco]="Yes")

2. Create a field ‘Insurance_A+’ which provides the [Total_InsuancePay] (Measure calculated as SUM(FactTable[Insuance_Payment])) only for patients with A+ Blood Group. Add this in the Hospital Matrix.
3. Create a table to show the Average Payment made by Males and Females patients of B+ blood group people to different hospitals.

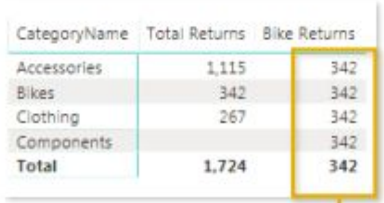
Power BI

Power BI

EXAMPLE OF CALCULATE FUNCTION (ADVENTURE WORKS)



```
Bike Returns = CALCULATE([Total Returns], Products[CategoryName] = "Bikes")
```



CategoryName	Total Returns	Bike Returns
Accessories	1,115	342
Bikes	342	342
Clothing	267	342
Components		342
Total	**1,724**	**342**

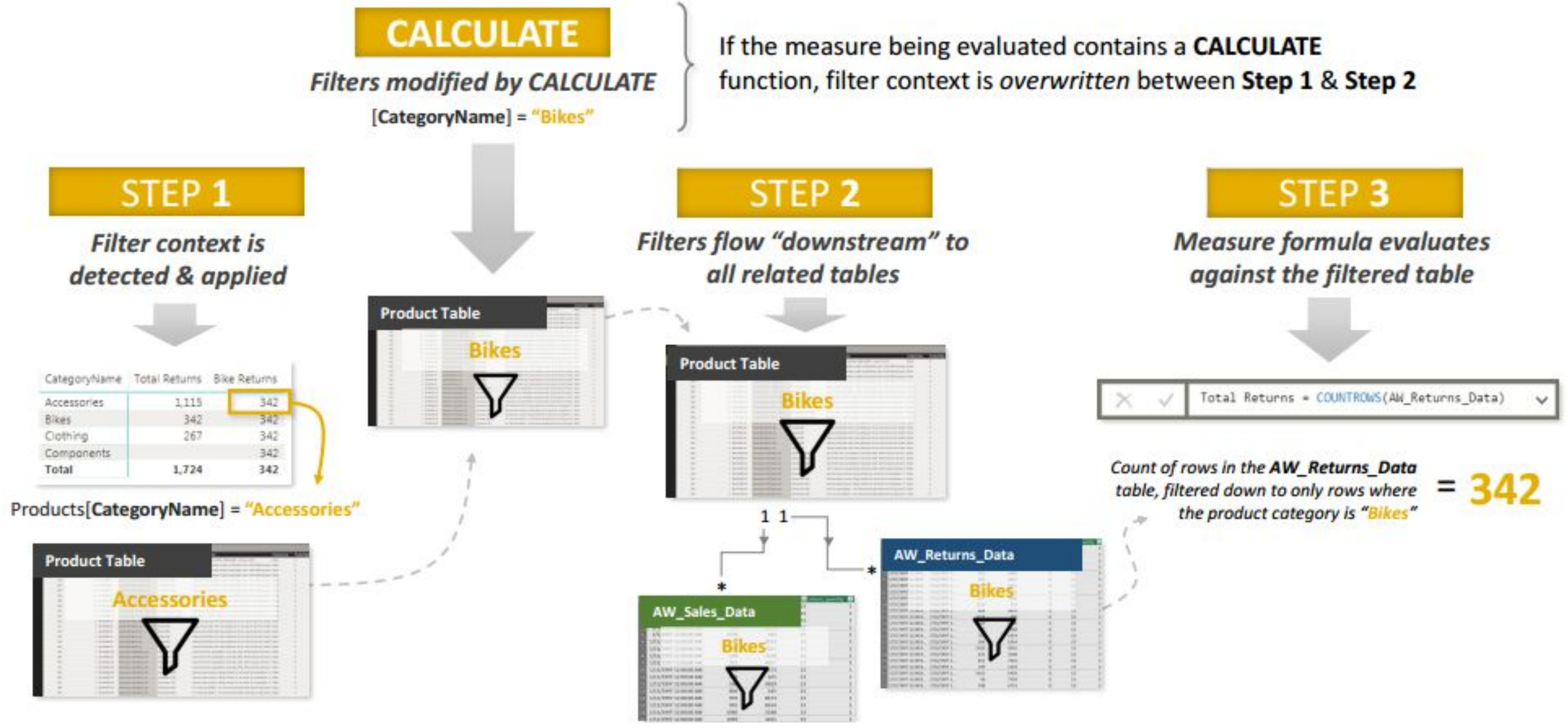
Here we've defined a new measure named "Bike Returns", which evaluates the "Total Returns" measure when the *CategoryName* in the **Products** table equals "Bikes"

*Notice that we see the repeating values when we view a matrix with **different Categories on rows**.*

*Shouldn't these cells have different filter contexts for **Accessories**, **Clothing**, **Components**, etc?*

CALCULATE modifies and overrules any competing filter context! In this example, the "Clothing" row has filter context of CategoryName = "Clothing" (defined by the row label) and CategoryName= "Bikes" (defined by the CALCULATE function). Both cannot be true at the same time, so the "Clothing" filter is overwritten and the "Bikes" filter (from CALCULATE) takes priority

EXAMPLE OF CALCULATE FUNCTION



CALCULATE

Filters modified by CALCULATE

[CategoryName] = "Bikes"

If the measure being evaluated contains a **CALCULATE** function, filter context is *overwritten* between **Step 1** & **Step 2**

STEP 1

Filter context is detected & applied

CategoryName	Total Returns	Bike Returns
Accessories	1,115	342
Bikes	342	342
Clothing	267	342
Components		342
Total	**1,724**	**342**

Products[**CategoryName**] = "Accessories"

STEP 2

Filters flow "downstream" to all related tables

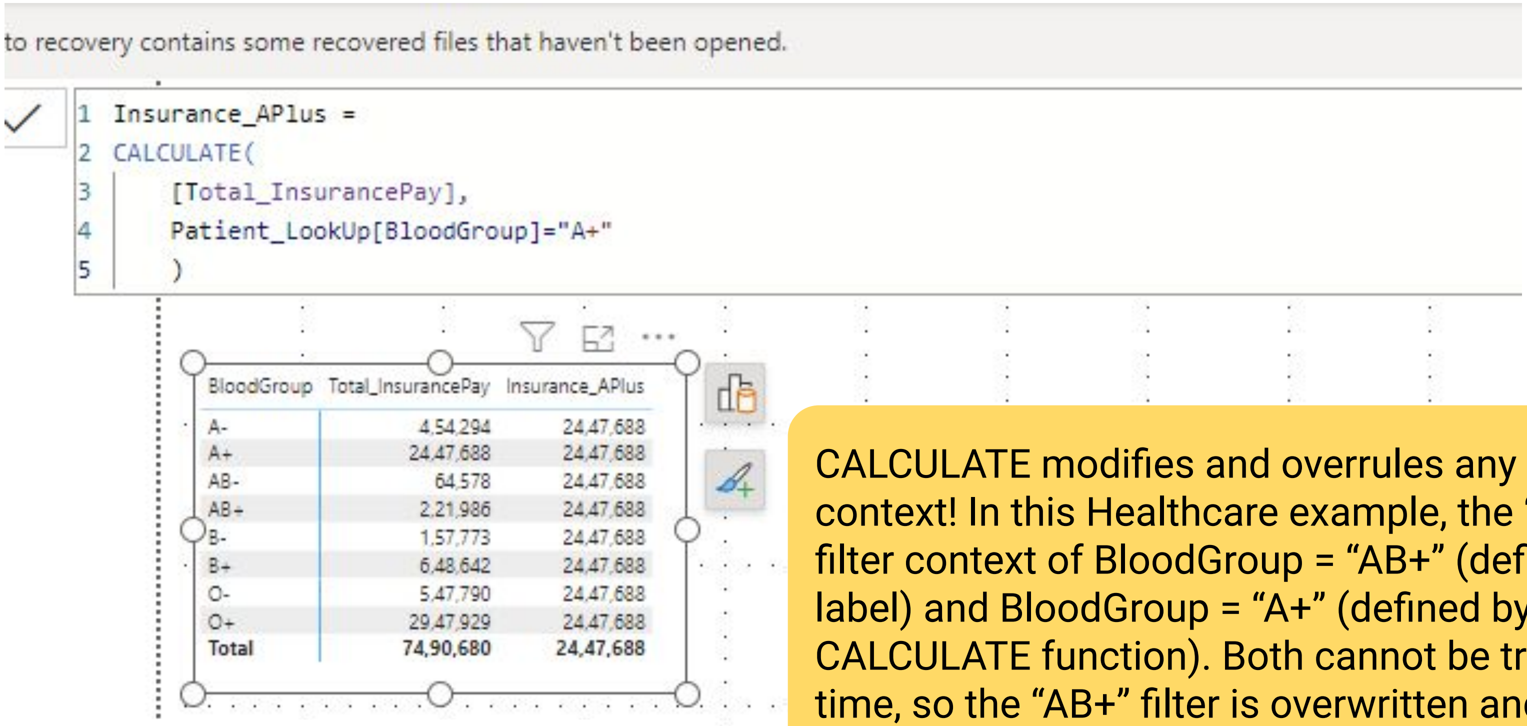
STEP 3

Measure formula evaluates against the filtered table

```
Total Returns = COUNTROWS(AW_Returns_Data)
```

*Count of rows in the **AW_Returns_Data** table, filtered down to only rows where the product category is "Bikes"* = 342

EXAMPLE OF CALCULATE FUNCTION (HEALTHCARE DATASET)



to recovery contains some recovered files that haven't been opened.

```
Insurance_APlus =
CALCULATE(
    [Total_InsurancePay],
    Patient_LookUp[BloodGroup]="A+"
    )
```

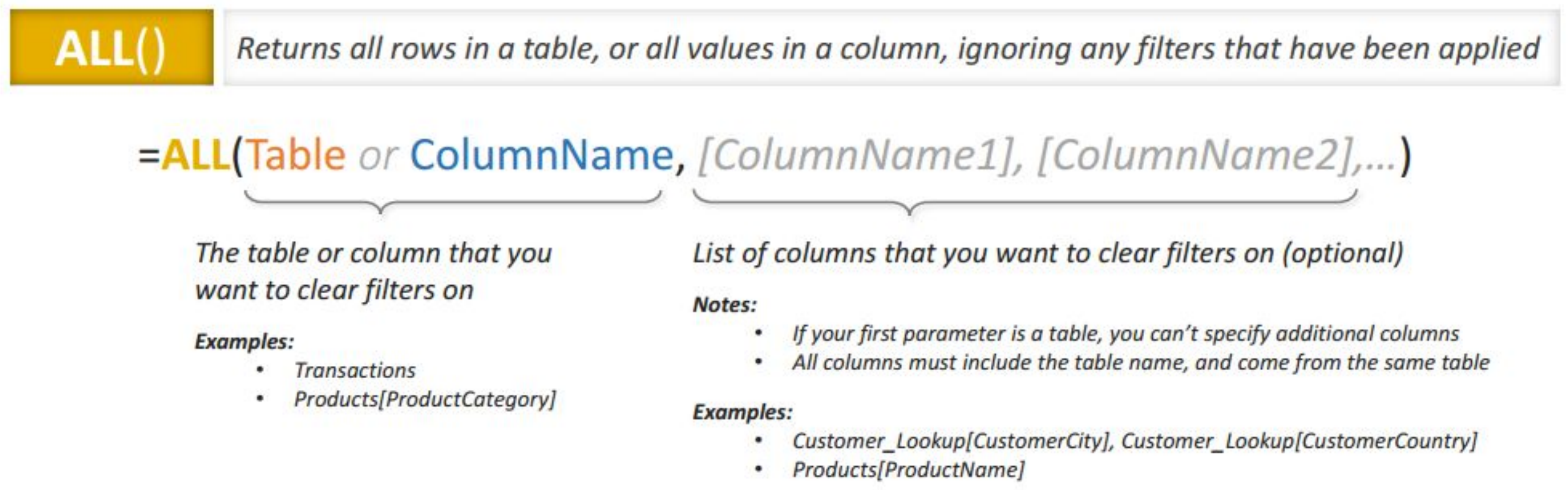
BloodGroup	Total_InsurancePay	Insurance_APlus
A-	4,54,294	24,47,688
A+	24,47,688	24,47,688
AB-	64,578	24,47,688
AB+	2,21,986	24,47,688
B-	1,57,773	24,47,688
B+	6,48,642	24,47,688
O-	5,47,790	24,47,688
O+	29,47,929	24,47,688
Total	**74,90,680**	**24,47,688**

*Notice that we see the repeating values when we view a matrix with **different Blood Groups on rows**.*

CALCULATE modifies and overrules any competing filter context! In this Healthcare example, the “AB+” row has filter context of BloodGroup = “AB+” (defined by the row label) and BloodGroup = “A+” (defined by the CALCULATE function). Both cannot be true at the same time, so the “AB+” filter is overwritten and the “A+” filter (from CALCULATE) takes priority.

Power BI

ADVANCE DAX - ALL FUNCTION



ALL() *Returns all rows in a table, or all values in a column, ignoring any filters that have been applied*

=ALL(Table *or* ColumnName, *[ColumnName1], [ColumnName2],…*)

The table or column that you want to clear filters on

Examples:

- *Transactions*
- *Products[ProductCategory]*

List of columns that you want to clear filters on (optional)

Notes:

- *If your first parameter is a table, you can't specify additional columns*
- *All columns must include the table name, and come from the same table*

Examples:

- *Customer_Lookup[CustomerCity], Customer_Lookup[CustomerCountry]*
- *Products[ProductName]*

Note - Instead of adding filter context, ALL removes it. This is often used when you need unfiltered values that won't react to changes in filter context (i.e. % of Total, where the denominator needs to remain fixed)

EXERCISE ON MEASURES - ALL FUNCTION

1. Create a field ‘FemalePatient’ that counts the Female patients. Use -

   ```
   CALCULATE([CountPatient], Patient_LookUp[PatientGender]="Female")
   ```

2. Calculate a new Measure ‘TotalFemales’ as -

   ```
   TotalFemales = CALCULATE([FemalePatient], ALL(Patient_LookUp))
   ```

3. Create a Matrix to show the % of Females for each BloodGroup.

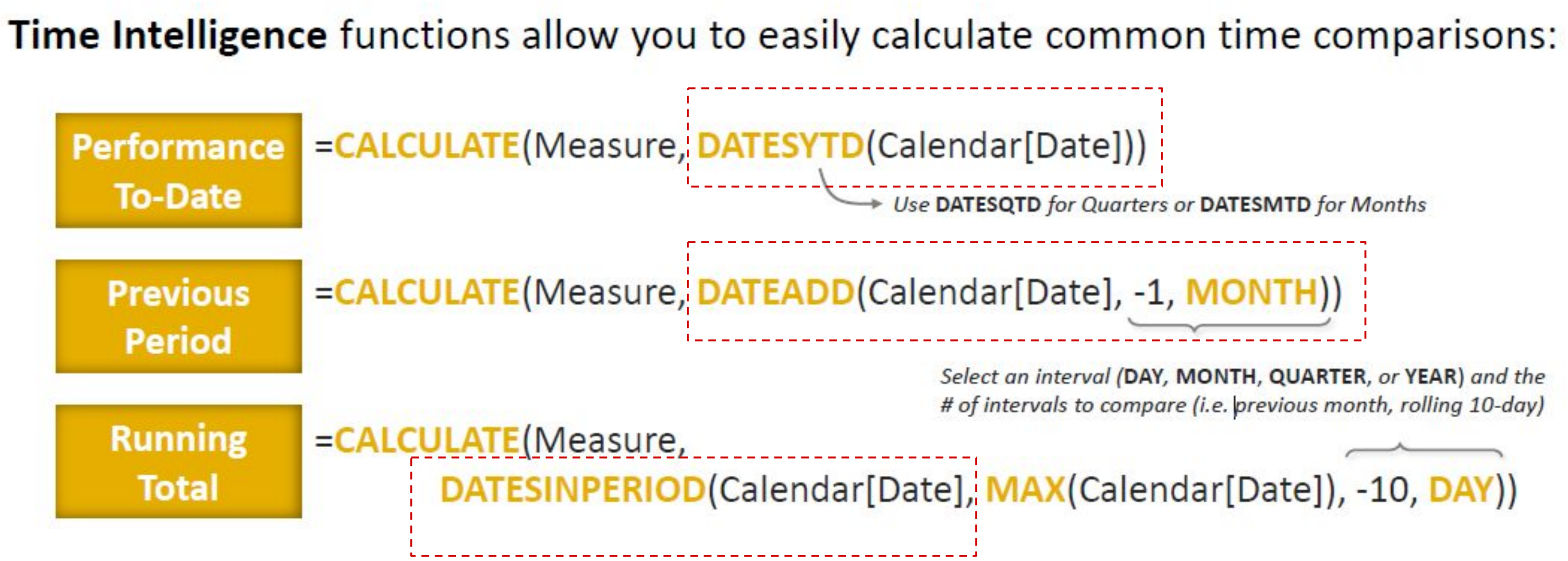
   Hint: (‘BloodGroup’ to Rows, ‘FemalePatient’ to Values and ‘TotalFemales’ also to values.

4. Create ‘%Females’ as `DIVIDE([FemalePatient], [TotalFemale])` and move it to matrix.
5. Calculate the ‘%Males’ using the same procedure.

Power BI

ADVANCE DAX - TIME INTELLIGENCE FUNCTIONS

Power BI



Time Intelligence functions allow you to easily calculate common time comparisons:

Performance To-Date

```
=CALCULATE(Measure, DATESYTD(Calendar[Date]))
```

Use **DATESQTD** *for Quarters or* **DATESMTD** *for Months*

Previous Period

```
=CALCULATE(Measure, DATEADD(Calendar[Date], -1, MONTH))
```

*Select an interval (***DAY, MONTH, QUARTER***, or* **YEAR***) and the # of intervals to compare (i.e. previous month, rolling 10-day)*

Running Total

```
=CALCULATE(Measure,
    DATESINPERIOD(Calendar[Date], MAX(Calendar[Date]), -10, DAY))
```

Moving Averages - Running Totals can be easily converted into Moving Averages by simply dividing by the number of intervals.

Power BI

EXERCISE ON MEASURES - TIME INTELLIGENCE FUNCTIONS

1. Create a field 'YTD Payment' in FactTable. Use - YTD Payment = CALCULATE([TotalPay], DATESYTD(DimDate[Date])). Add this to Matrix view along with the Start of the Month in Rows to show the YTD Payments for the years.
2. Create a field 'Previous MonthPay'. Use - PreviousMonthPay = CALCULATE([Total Pay], DATEADD(DimDate[Date], -1,MONTH)). In the similar way create 'Previous Month Insurance Payment', 'Previous Month Patient Payment' and 'Previous Month CPT Units'.
3. Create a field '10DayRollingPay'. (**Hint:** 10DayRollingPay = CALCULATE([TotalPay], DATESINPERIOD(DimDate[Date], MAX(DimDate[Date]),-10,DAY)).
4. Calculate the Moving Average for the Total Pay as - TotalPayMA = DIVIDE(CALCULATE ([TotalPay], DATESINPERIOD(DimDate[Date],MAX(DimDate[Date]),-10,DAY)),10)

ADVANCE DAX - FILTER FUNCTION

FILTER | Returns a table that represents a subset of another table or expression

=**FILTER**(Table, FilterExpression)

Table to be filtered

Examples:

- *Territory Lookup*
- *Customer Lookup*

A Boolean (True/False) filter expression to be evaluated for each row of the table

Examples:

- *'Territory Lookup'[Country]* = ***"USA"***
- *Calendar[Year]* = ***1998***
- *Products[Price]* > ***[Overall Avg Price]***

FILTER is an iterator function i.e. it works on each row of the Table and then based on the filter expression it creates a subset of the table. Since it works on each row it can be time consuming to get the result from FILTER. Also FILTER returns the entire table hence it is nested within other functions like CALCULATE.

Power BI

EXERCISE ON MEASURES - FILTER FUNCTION

1. Create a new table ‘BadDebtTable’ to understand the Bad Debts transaction using BadDebtTable = FILTER(Trancstion_LookUp, Trancstion_LookUp[AdjustmentReason] = "Bad Debt")

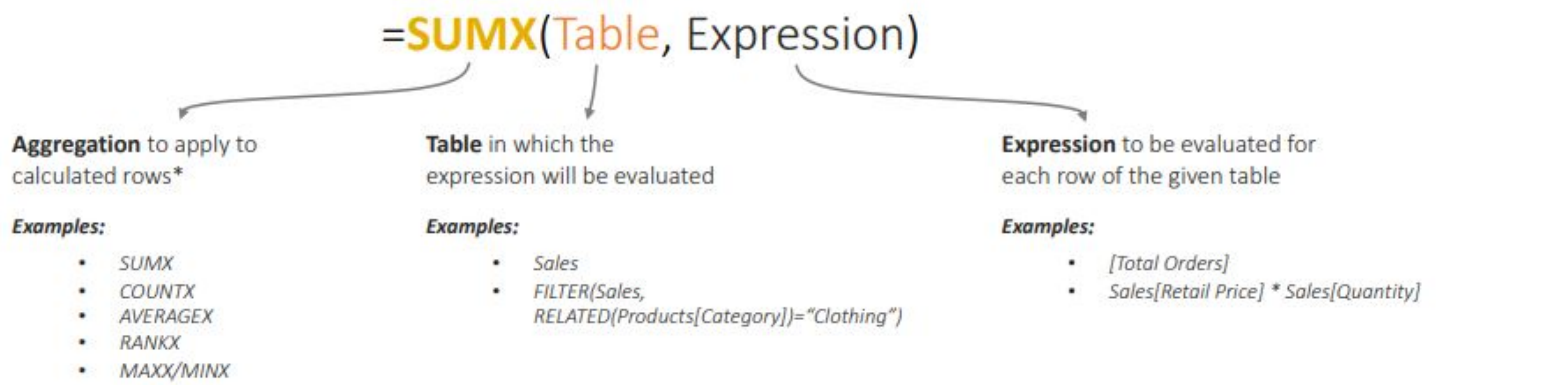
Power BI

MORE DAX CALCULATIONS FOR HEALTHCARE DATA ANALYSIS

1. Create a measure 'BadDebt' using BadDebts = CALCULATE([TotalGrossExpense], FactTable[Transaction-FK] = 55711 || FactTable[Transaction-FK] = 48429 || FactTable[Transaction-FK] = 55042)
2. Create a Measure ARGP Ratio as - ARGP Ratio = SUM(FactTable[AR])/ SUM(FactTable[Gross Expenses])
3. Create a Measure IPTP Ratio as - IPTP Ratio = SUM(FactTable[Insurance_Payment])/[TotalPay])

ADVANCE DAX - ITERATOR FUNCTIONS - SUMX

Iterator functions allow you to loop through the same expression on each row of a table, then apply some sort of aggregation to the results (SUM, MAX, etc.)



=SUMX(Table, Expression)

Aggregation to apply to calculated rows*

Examples:

- *SUMX*
- *COUNTX*
- *AVERAGEX*
- *RANKX*
- *MAXX/MINX*

Table in which the expression will be evaluated

Examples:

- *Sales*
- *FILTER(Sales, RELATED(Products[Category])="Clothing")*

Expression to be evaluated for each row of the given table

Examples:

- *[Total Orders]*
- *Sales[Retail Price] * Sales[Quantity]*

Imagine that Iterator functions add a temporary new column to a table, calculate a value in each row based on the given expression, then aggregate the values within that temporary column (similar to SUMPRODUCT in Excel).

EXERCISE ON MEASURES - ITERATOR FUNCTION

1. Create a Measure ‘Total Deficit’. Use SUMX to calculate ‘Total Deficit’. Total Deficit as Gross Expense less Total Payment.
2. Create a Measure ‘Average Deficit’. Use AVERAGEX to calculate the Average Deficit. Show the Average Deficit for the Hospitals in Matrix

ADVANCE DAX - ITERATOR FUNCTIONS - RANKX

RANKX()

Returns the ranking of a number in the list of numbers for each row in the table argument

= RANKX(Table, Expression, [Value], [Order], [Ties])

Table or DAX expression
that returns the Table

Example:
ALL(Product_Lookup[Product])

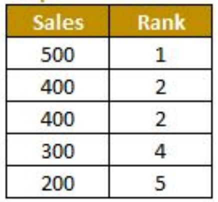
An expression that returns a scalar value, evaluated at each row of the table.

Example:
[Customer Sales],
SUM('Sales by Store' [Quantity Sold])

Optional Arguments:

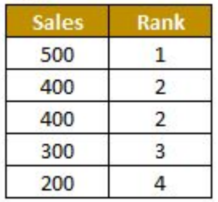
- ***Value:*** *Any DAX expression that returns a single scalar value, whose rank is to be found. By default this value is the current row.*
- ***Order:*** *specifies how to Rank. High to low or low to high. Example : ASC or DESC*
- ***Ties:*** *Determine how ties and following ranks are treated.*
  - *SKIP (default): Skips ranks after ties.*
  - *DENSE: Shows the next rank regardless of ties*

Skip



Sales	Rank
500	1
400	2
400	2
300	4
200	5

Ties



Sales	Rank
500	1
400	2
400	2
300	3
200	4

Power BI

EXERCISE ON MEASURES - ITERATOR FUNCTION

1. Create a variable 'TotalPayRank' to calculate the Rank of Hospitals based on Total Payments.

   Use: RANKX(ALL(Hospital_LookUp[HospitalName]), [Total_Payment])

SOME IMPORTANT CALCULATIONS - PERCENTAGE (%)

Power BI

Power BI

SOME IMPORTANT CALCULATIONS - PERCENTAGE (%)

SOME IMPORTANT CALCULATIONS - RUNNING TOTAL

Power BI