

# POWER BI TRAINING



**Power BI**

TURN YOUR DATA INTO IMPACT!

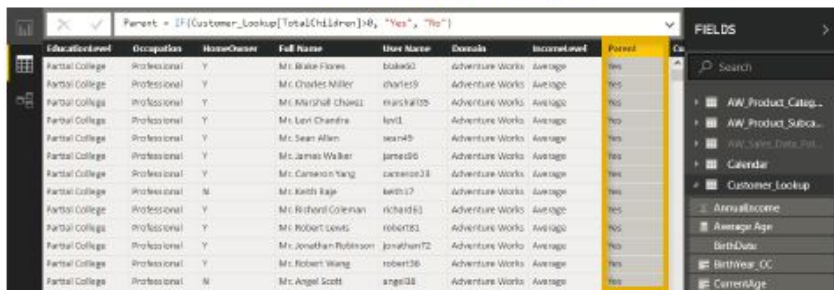
# CALCULATED FIELDS WITH DAX

# DAX - DATA ANALYSIS EXPRESSION

- Formula language that drives the Power BI
- With DAX one can add - **Calculated Columns** and **Measures** to your data model

## Two ways to use DAX

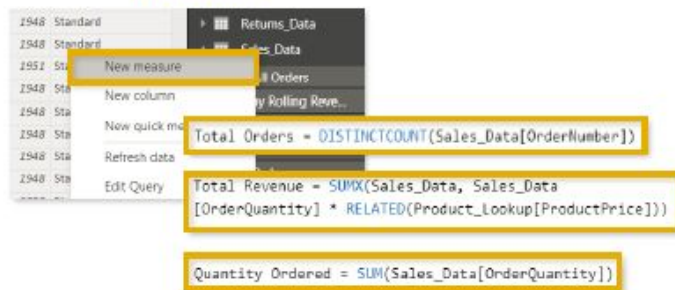
### 1) Calculated Columns



The screenshot shows the Power BI interface with a calculated column formula bar at the top: `Parent = IF(Customer_Lookup[TotalChildren]>0, "Yes", "No")`. Below the formula bar is a table with columns: EducationLevel, Occupation, HomeOwner, Full Name, User Name, Details, IncomeLevel, and Parent. The table contains 15 rows of data for Adventure Works employees.

EducationLevel	Occupation	HomeOwner	Full Name	User Name	Details	IncomeLevel	Parent
Partial College	Professional	Y	Mr. Blake Flores	blake90	Adventure Works	Average	Yes
Partial College	Professional	Y	Mr. Charles Miller	charles9	Adventure Works	Average	Yes
Partial College	Professional	Y	Mr. Marshall Chow	marshall35	Adventure Works	Average	Yes
Partial College	Professional	Y	Mr. Len Chamfrs	len1	Adventure Works	Average	Yes
Partial College	Professional	Y	Mr. Sean Allen	sean49	Adventure Works	Average	Yes
Partial College	Professional	Y	Mr. James Walker	james66	Adventure Works	Average	Yes
Partial College	Professional	Y	Mr. Cameron Yang	cameron38	Adventure Works	Average	Yes
Partial College	Professional	N	Mr. Keith Raje	keith7	Adventure Works	Average	Yes
Partial College	Professional	Y	Mr. Richard Coleman	richard60	Adventure Works	Average	Yes
Partial College	Professional	Y	Mr. Robert Lewis	robert81	Adventure Works	Average	Yes
Partial College	Professional	Y	Mr. Jonathan Robinson	jonathan72	Adventure Works	Average	Yes
Partial College	Professional	Y	Mr. Robert Wang	robert36	Adventure Works	Average	Yes
Partial College	Professional	N	Mr. Angel Scott	angel38	Adventure Works	Average	Yes

### 2) Measures




The screenshot shows the 'New measure' menu in Power BI. Below the menu, three DAX formulas are displayed in yellow boxes:

- `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)

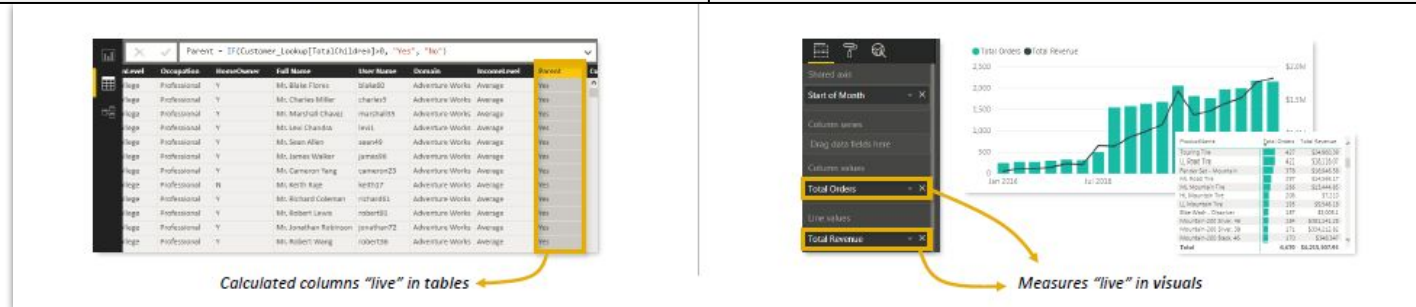


As a thumb rule, use Measures when a single row cannot give you the answer i.e. when you need to apply aggregation use Measures

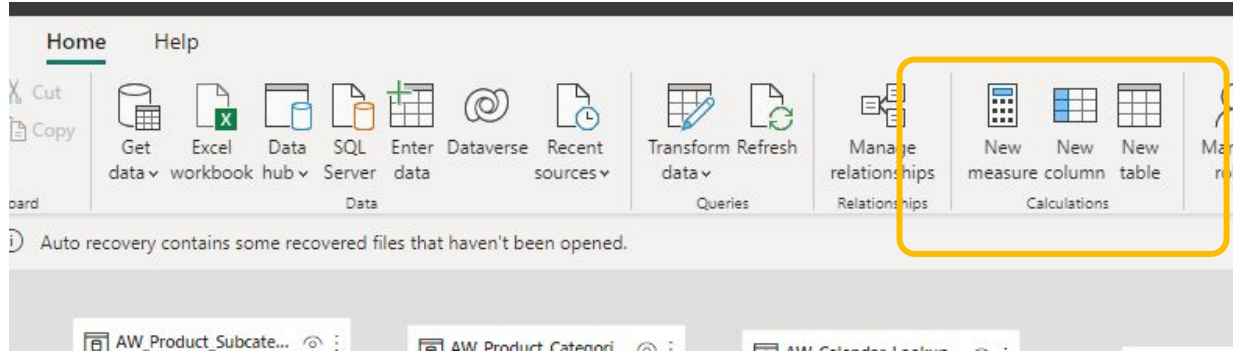
Use Measures to create numerical, calculated values that can be analysed in the 'Values' field of the report.

# DAX - MEASURES VS CALCULATED COLUMNS

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

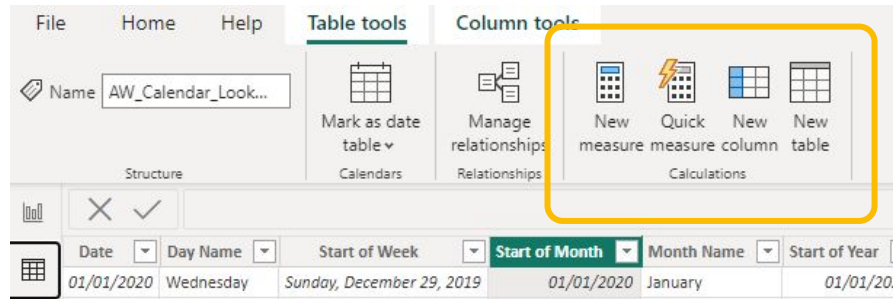


# ADDING MEASURES AND CALCULATED COLUMNS

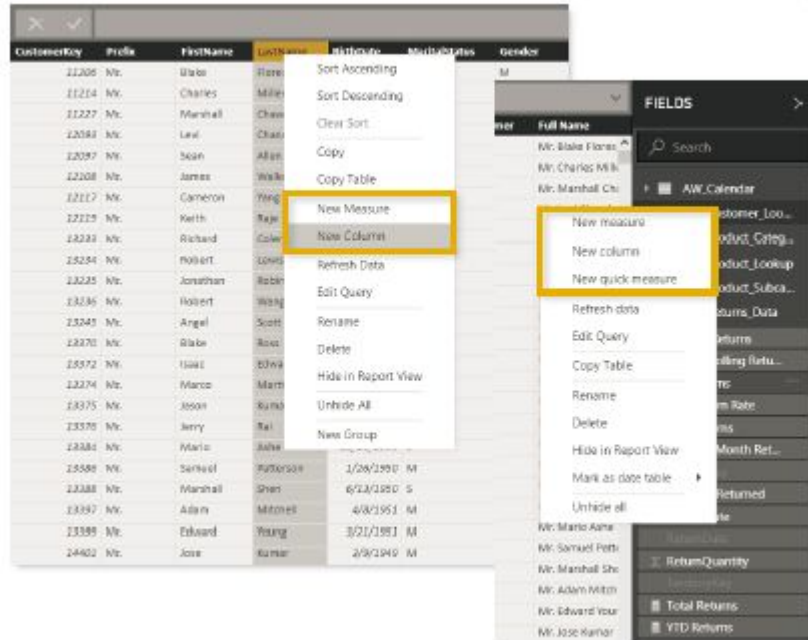


When you insert a Column or Measure from the Home tab they are assigned to whichever table is currently selected (or the first table in the field list by default).

Can be reassigned to new Home Tables



# ADDING MEASURES AND CALCULATED COLUMNS



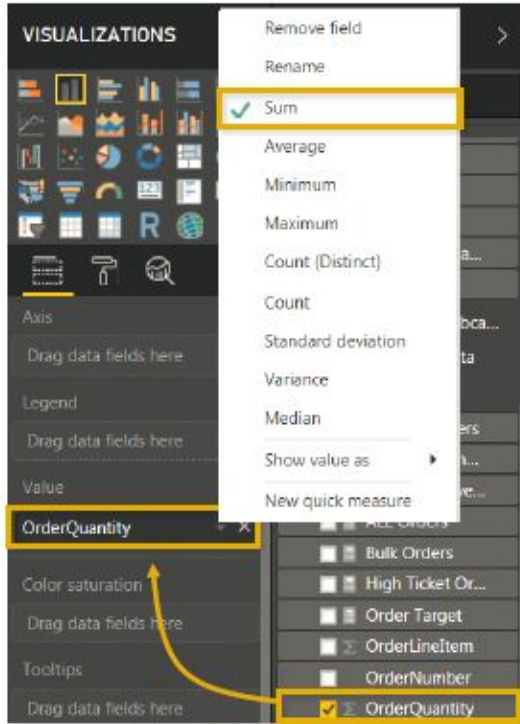
The screenshot shows a Power BI Desktop interface. On the left, a data table is displayed with columns: CustomerKey, Prefix, FirstName, LastName, Birthdate, BirthdateStatus, and Gender. The 'LastName' column is selected, and a context menu is open, showing options like 'Sort Ascending', 'Sort Descending', 'Clear Sort', 'Copy', 'Copy Table', 'New Measure', 'New Column', 'Refresh Data', 'Edit Query', 'Rename', 'Delete', 'Hide in Report View', 'Unhide All', and 'New Group'. The 'New Measure' and 'New Column' options are highlighted with a yellow box. On the right, the 'FIELDS' pane is visible, showing a list of fields including 'Full Name', 'Mr. Blake Flores', 'Mr. Charles Milk', 'Mr. Marshall Chi', 'AW.Calendar', 'Customer\_Loo...', 'Product\_Categ...', 'Product\_Lookup', 'Product\_Subca...', 'Returns\_Data', 'Returns', 'Shipping\_Retu...', 'Total Returns', and 'YTD Returns'. The 'New Measure' and 'New Column' options are also highlighted with a yellow box in the 'FIELDS' pane.

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

# 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.	1,144	1.96 %
Road Tire Tube	829	1.63 %
AWC Logo Cap	803	0.93 %
Patch KR/S Patches	798	1.57 %
Sport-100 Helmet, Red	753	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	1.32 %
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	155	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 it's own filter context (even Totals and Grand Totals)

# UNDERSTAND FILTER CONTEXT

MEASURE: Total Revenue

FILTER CONTEXT:

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

Full Name	Total Orders	Total Revenue
Mr. Maurice Shan	6	\$12,907.98
Mrs. Janet Munoz	6	\$12,907.98
Mrs. Lisa Chen	7	\$13,180.44
Ms. Lacey Zhang	7	\$13,180.74
Mr. Jordan Turner	7	\$13,200.99
Mr. Larry Munoz	8	\$16,950.24
Mrs. Ariana Gray	6	\$10,951.40
Mr. Marco Lopez	6	\$10,951.40
Mr. Francisco Ruiz	5	\$10,949.94
Mrs. Margaret Hill	4	\$8,540.76
Mrs. Kathryn Henderson	4	\$8,539.55
Mrs. Nichole Napp	4	\$8,534.40
Mr. Randal Dominguez	4	\$8,526.96
Mrs. Rishi Rao	4	\$8,509.12
Adriana Gonzalez	4	\$8,509.00
Mrs. Dominique Pineda	6	\$9,500.93
Mrs. Brenda Gill	4	\$8,189.18
Mr. Brad She	4	\$8,143.76
Mr. Francisco Sosa	4	\$8,129.94
Mr. Kevin Coleman	4	\$7,766.55
Mr. Jonathan Suli	4	\$7,721.33
Mrs. Crystal Zeng	4	\$7,716.81
Mrs. Felicia Blanco	4	\$7,646.64
Mrs. Jill Suarez	4	\$7,630.61
Mr. Preston Raman	4	\$7,549.49
Mr. Willie Ruiz	4	\$7,533.55
Mrs. Abby Eubank	4	\$7,186.38
Mr. Lance Blasco	4	\$7,207.07
Mrs. Audrey Sprad	4	\$7,283.17
Mr. Roby Navarro	3	\$7,119.43
Mr. Todd Dominguez	3	\$7,044.38
Mrs. Molly Meador	3	\$7,043.26
Mr. Jared Martin	3	\$7,038.37
Mr. Susan Shou	3	\$7,027.80
Mr. Brent Zhang	3	\$7,025.90
Mr. Alyssa Bradley	3	\$7,008.94
Total	22,514	\$4,598,433.2

MEASURE: Total Orders

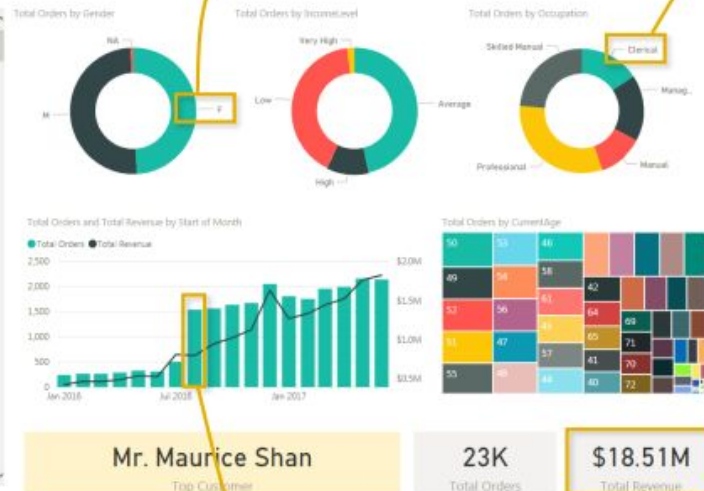
FILTER CONTEXT:

- Calendar[Year] = 2016 or 2017

MEASURE: Total Orders

FILTER CONTEXT:

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



MEASURE: Total Orders

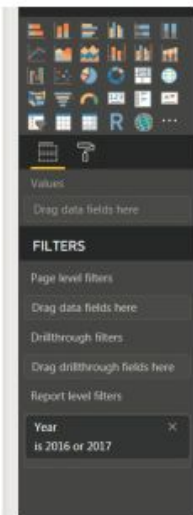
FILTER CONTEXT:

- Calendar[Year] = 2016 or 2017
- Calendar[Month] = August 2016

MEASURE: Total Orders

FILTER CONTEXT:

- Calendar[Year] = 2016 or 2017
- Customers[Occupation] = Clerical



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

MEASURE: Total Revenue

FILTER CONTEXT:

- Calendar[Year] = 2016 or 2017

# EXERCISE ON DAX - CALCULATED COLUMN

1. Add a column 'Day of week' in 'FactTable'. Assume week start on Monday. Use function - Weekday
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.

# EXERCISE ON DAX - CALCULATED COLUMN

1. Create a calculated column in the 'Patients\_LookUp' table that calculates the BMI (Body Mass Index) using the formula:  $BMI = (PatientWeight / (PatientHeight ^ 2)) * 703$ . (**Hint:** Use the "dimpatients" table fields "PatientWeight" and "PatientHeight").
2. 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").
3. Create a column 'CPTUnitType' in FactTable that will show us if the quantity order was - "Single Unit" or "Multiple Units"

# EXERCISE ON DAX - CALCULATED COLUMN ( PRACTICE)

1. Create a calculated column in the 'FactTable' that calculates the total payments (Insurance\_Payment + Patient\_Payment) for each payer. (**Hint:** Use '+' symbol to add both). Try using SUM aggregation also and see the difference.
2. 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()".

# 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.
3. Also bring in 'Patient\_Payment' (from [FactTable](#)) into the matrix.

## Creating an Explicit Measure

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

# UNDERSTAND THE MECHANICS OF MEASURES

CategoryName	Total Returns
Accessories	1115
Bikes	342
Clothing	267

**How exactly is this measure calculated?**

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

## STEP 1

Filter context is detected & applied

CategoryName	Total Returns
Accessories	1115
Bikes	342
Clothing	267

Product[CategoryName] = "Accessories"

Product Table	
Accessories	

## STEP 2

Filters flow "downstream" to all related tables

Product Table	
Accessories	

AW_Sales_Data	
Accessories	

AW>Returns_Data	
Accessories	

## 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 "Accessories" = 1,115



# UNDERSTAND DAX SYNTAX

## MEASURE NAME

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

Total Quantity: =SUM(Transactions[quantity])

Referenced  
TABLE NAME

Referenced  
COLUMN NAME

## 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 preceded 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

**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.**

# 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])



# EXERCISE ON DAX MEASURES - MATHS FUNCTIONS

1. Add a field 'Total\_InsurancePay' in 'FactTable' using  $\text{SUM}(\text{FactTable}[\text{Insurance\_Payment}])$ . Also create 'Average\_InsurancePay'. This will create new field in the Reports view and will not add column in the data.
2. Add a field 'Total\_PatientPay' in 'FactTable' using  $\text{SUM}(\text{FactTable}[\text{Patient\_Payment}])$ . Also create 'Average\_PatientPay'. This create new field in the Reports view. **(Practice)**
3. Create another Measure 'Total\_Payment' in 'FactTable' as -  $\text{SUM}(\text{FactTable}[\text{Insurance\_Payment}]) + \text{SUM}(\text{FactTable}[\text{Patient\_Payment}])$ . **Create a Matrix to view Total\_InsurancePay, Average\_InsurancePay, Total\_Payment for different - a) Hospitals, b) Payer**
4. Create a Measure - '%InsurancePay' in the 'FactTable'. Use formula as below -  $\%InsurancePay = \frac{\text{SUM}(\text{FactTable}[\text{Insurance\_Payment}])}{(\text{SUM}(\text{FactTable}[\text{Insurance\_Payment}]) + \text{SUM}(\text{FactTable}[\text{Patient\_Payment}]))}$

# 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)

# 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 COUNTA 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. (Practice - Try for other 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.



# 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)

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

## OR

Checks whether any argument is TRUE to return TRUE, otherwise returns FALSE

=**OR**(Logical1, Logical2)





# EXERCISES ON DAX COLUMNS - LOGICAL

1. Create a new calculated column in the 'FactTable-2.csv' table that checks if the patient age is greater than 50. (**Hint:** Use the 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 that shows the Patient who consumes Alcohol and Tobacco both (**Hint:** Use AND logical operator)
4. Create a column 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.

# EXERCISES ON DAX COLUMNS - LOGICAL (PRACTICE)

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).

# TEXT FUNCTIONS

**LEN()**

Returns the number of characters in a string

**CONCATENATE()**

Joins two text strings into one

**LEFT/MID/  
RIGHT()**

Returns a number of characters from the start/middle/end of a text string

**UPPER/LOWER/  
PROPER()**

Converts letters in a string to upper/lower/proper case

**SUBSTITUTE()**

Replaces an instance of existing text with new text in a string

**SEARCH()**

Returns the position where a specified string or character is found, reading left to right

=**LEN**(Text)

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

=**CONCATENATE**(Text1, Text2)

=**LEFT/RIGHT**(Text, [NumChars])

=**MID**(Text, StartPosition, NumChars)

=**UPPER/LOWER/PROPER**(Text)

=**SUBSTITUTE**(Text, OldText, NewText, [InstanceNumber])

=**SEARCH**(FindText, WithinText, [StartPosition], [NotFoundValue])

# 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.

# EXERCISE ON DAX COLUMNS - TEXT (PRACTICE)

1. Create a calculated column in the dimspeciality table to extract the first two characters of the speciality code. (**Hint:** Use the LEFT function)
2. Create a calculated column in the dimpatients table to concatenate the first name and last name of patients. (**Hint:** Use the CONCATENATE function)

# 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)

# 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).

# EXERCISE ON DAX COLUMNS - TIME BASED (PRACTICE)

1. Create a calculated column in the 'DimDate' table to extract the year from the "Date" column. (**Hint:** Use the YEAR() function).
2. 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).
3. Create a calculated column in the 'DimDate' table to extract the month from the "Date" column. (**Hint:** Use the MONTH() function).



# 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]



# EXERCISE ON DAX - RELATED FUNCTION

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

# 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

- Create a field 'Insurance\_Tobacco' which provides the [Total\_InsurancePay] (Measure calculated as SUM(FactTable[Insurance\_Payment])) only for patients consuming Tobacco. Show this in the Hospital Matrix.

**Hint:** CALCULATE([Total\_InsurancePay],Patient\_LookUp[Tobacco]="Yes")

- Create a field 'Insurance\_A+' which provides the [Total\_InsurancePay] (Measure calculated as SUM(FactTable[Insurance\_Payment])) only for patients with A+ Blood Group. Add this in the Hospital Matrix.

# 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"

Product Table
Accessories

### STEP 2

Filters flow "downstream" to all related tables

Product Table
Bikes

Product Table
Bikes

AW_Sales_Data
Bikes

AW_Returns_Data
Bikes

### 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.

```

1 Insurance_APlus =
2 CALCULATE(
3     [Total_InsurancePay],
4     Patient_LookUp[BloodGroup]="A+"
5 )

```

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.

# 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" 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.

# ADVANCE DAX - TIME INTELLIGENCE FUNCTIONS

**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.



# EXERCISE ON MEASURES - TIME INTELLIGENCE FUNCTIONS

1. Create a field 'YTD Payment' in FactTable. Use - `YTD Payment = CALCULATE([TotalPay], DATESYTD(DimDate[Date]))`. Also make sure you remove the Date Hierarchy so view the YTD payments.
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.

# 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.

# EXERCISE ON MEASURES - FILTER FUNCTION

- Create a new table 'BadDebtTable' to understand the Bad Debts transaction using `BadDebtTable = FILTER(Trancstion_LookUp, Trancstion_LookUp[AdjustmentReason] = "Bad Debt")`



# MORE DAX CALCULATIONS FOR HEALTHCARE DATA ANALYSIS

- Create a measure 'BadDebt' using  $\text{BadDebts} = \text{CALCULATE}([\text{TotalGrossExpense}], \text{FactTable}[\text{Transaction-FK}] = 55711 \parallel \text{FactTable}[\text{Transaction-FK}] = 48429 \parallel \text{FactTable}[\text{Transaction-FK}] = 55042)$
- Create a Measure ARGP Ratio as -  $\text{ARGP Ratio} = \text{SUM}(\text{FactTable}[\text{AR}]) / \text{SUM}(\text{FactTable}[\text{Gross Expenses}])$
- Create a Measure IPTP Ratio as -  $\text{IPTP Ratio} = \text{SUM}(\text{FactTable}[\text{Insurance\_Payment}]) / [\text{TotalPay}]$