POWER BI – DAX

Ivy Professional School

Top Ranked **Data Science Education Provider since 2007**

DAX

DAX

DAX – Data analysis Expressions

a. It is formula language which drive Power BI
b. Two ways of using Dax
    i. Calculated Columns - create new column
    ii. New Measures – no new column created but they can be used in Report View for creating visualizations

Calculated Column

Allows to create new, formula based columns to the tables

There is no A1 Style, it refers to the entire column/tables

They generate new value for each row and they are visible in data view

They understand row context. They are useless for aggregation like SUM, Average etc.

Used if we want static, fixed value in each row of the table

They do not understand filtered context so we do aggregation it returns the same value in all the rows.

Measures

They are used to create new calculated **VALUES**

Refers to entire column or table

They are not visible in tables. They are used in Report View.

This is similar to Calculated field in Excel Pivot Tables

They are evaluated based on Filter context which means that they change when the field or the filters around them change.

Used when single row is not going to give us the answer. We will have to aggregate it.

Three Ways of Adding Columns & Measures

Option 1 – In Report View - from modelling tab – new measure and new column

Option 2 – Data View – right click on any column and New Measure or New Column

Option 3 – Right click on the table in Report view and select New Measure or New column

Implicit and Explicit Measures

1. Implicit measures – they are created when we drag and drop a numerical field in the value part of the visual and manually select aggregation.
   - They can only be used only in specific visualization and can’t be referred anywhere else.
2. Explicit measures – they are created using DAX functions to define calculated columns or measures
   - They are created as measures in the table and hence can be referred in other calculations.

Exercise

- Create in Sales Table
  - Total Qty (measure)
- Create in Returns Table
  - Total Return Qty (Measure)
  - % Return (Measure)
- Create in Product Table
  - ProductProfit i.e. Price-cost (calculated column)

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]*

Arithmetic Operators in Power BI

Arithmetic Operators	Meaning	Example
+	Addition	2+7
-	Subtraction	7-2
*	Multiplication	5*6
/	Division	10/5
^	Exponent	2^3

Comparison Operators in Power BI

Comparison Operators	Meaning	Example
=	Equal To	[City]=”Boston”
>	Greater than	[Quantity]>100
<	Less than	[Quantity]<100
>=	Greater than or equal to	[Unit Price]>=3
<=	Less than or equal to	[Unit Price]<=3
<>	Not equal to	[Country]<>”China”

Text/Logical Operator

Text/Logical Operator	Meaning	Example
&	Concatenates two values	[City]&" "&[State]
&&	Creates AND condition	([State]="NY" && [Quantity]>10)
|| (Double Pipe)	Creates OR condition	([State]="NY" || [State]="MA")
IN	Created OR condition based on list	'Store Lookup'[State] IN {"MA", "NY", "CT"}

Exercise

- Calculated total number of Customers (18K)
- Calculate the unique products (293)
- Average Children (1.84)

Exercise – Customer Dashboard

- Gender wise
  - Qty Bought analysis
  - Number of male and female
- Trend of customer over period of time (Month and Year)
- Top 10 Customer wrt Qty
- Number of customers based on
  - Martial Status
  - Education

Common Function Categories

- Maths and Stats
  - Basic aggregation functions as well as iterators evaluated at row-levl
- Logical
  - Used for checking conditions
- Text
  - Used to manipulate text strings or control formats of dates, time and numbers
- Filter
- Date and Time

Maths and Stats

- Common examples:
  - SUM
  - AVERAGE
  - MAX/MIN
  - DIVIDE
  - COUNT/COUNTA
  - COUNTROWS
  - DISTINCTCOUNT
- Iterator
  - SUMX
  - AVERAGEX
  - MAXX/MINX
  - RANKX
  - COUNTX

Logical Functions #1

- Common Examples:
  - IF
  - IFERROR
  - AND
  - OR
  - NOT
  - SWITCH
  - TRUE
  - FALSE

Logical Functions #2

- IF –
  - Checks if condition is met, returns one value if condition is True and another is condition is False
  - =IF(LogicalTest, ResultIfTrue, [ResultIfFalse])
- IFERROR –
  - Evaluates an expression and returns specified value if expression returns an error otherwise returns expression itself
  - =IFERROR(Value, ValueIfError) → Sales/Qty → Qty =0
  - Iferror(sales/qty,"No Qty Sold")
- AND –
  - Checks whether both arguments are True, Returns True is both are True Otherwise False
  - For more than 2 conditions && is used
  - =AND (Logical1, Logical2)
- OR –
  - Check whether one of the argument if True. Returns true is either one is True and Returns False if both arguments are False
  - For more than 2 conditons || is used
  - =OR (Logical1, Logical2)

Exercise

- In Customer Table
  - Create 'Target Customer' Column
    - Target if the Annual Income >=100000 or Children >1
- In Calendar Table
  - Go to Transformation and add a new column of day of the week
  - Then using DAX create a column of Weekday or Weekend

Text Function #1

- Common examples:
  - CONCATENATE - &
  - FORMAT
  - LEFT/RIGHT/MID
  - UPPER/LOWER
  - PROPER
  - LEN
  - SEARCH/FIND
  - REPLACE
  - REPT
  - SUBSTITUTE
  - TRIM

Text Function #2

- LEN –
  - Returns number of characters in a string
  - =LEN(Text)
- CONCATENATE –
  - Join two text strings into one
  - =CONCATENATE(Text1, Text2) → Text1 & Text2 & Text3
  - Use & to join more than 2 text string
- LEFT/MID/RIGHT –
  - Returns a number of characters from the Start/Middle/End of text string
  - =LEFT/RIGHT(Text, [Num of Char])
  - =MID(Text, Start Position, Num of Char)
- UPPER/LOWER –
  - Converts letter in a string to UPPER/lower/Proper case
  - =UPPER/LOWER(Text)

Text Function #3

- SUBSTITUTE –
  - Replaces an instance of existing text with new text in a string
  - =SUBSTITUTE(Text, OldText, NewText, [InstanceNum])
- SEARCH /FIND–
  - Returns the position where a specified string or character is found, reading left to right
  - =SEARCH(FindText, WithInText, [StartPosition], [NotFoundValue])

Exercise

- In Customer Table
  - Using Calculated Column create a column with Full name
  - Create a new column with only the username from the domain name in Email ID column

Date and Time Functions #1

- Common examples:
  - DATEDIFF
  - YEARFRAC
  - YEAR/MONTH/DAY
  - HOUR/MINUTE/SECOND
  - TODAY/NOW
  - WEEKDAY/WEEKNUM
- Time Intelligence Functions
  - DATESYTD
  - DATESQTD
  - DATESMTD
  - DATEADD
  - DATESINPERIOD

Date and Time Function #2

- DAY/MONTH/YEAR –
  - Returns day of the month (1-31), Month of the year (1-12), Year of a date
  - =DAY/MONTH/YEAR(Date)
- HOUR/MINUTE/SECOND –
  - Returns the hour (0-23), minute (0-59), second (0-59) of datetime value
  - =HOUR/MINUTE/SECOND(Datetime)
- TODAY/NOW –
  - Returns current date or exact time
  - =TODAY/NOW()
- WEEKDAY/WEEKNUM –
  - Returns weekday number from 1 (Sunday) to 7 (Saturday) or week # of year
  - =WEEKDAY/WEEKNUM (Date,[ReturnType])

Date and Time Function #3

- EOMONTH –
  - Returns 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 selected intervals
  - =DATEDIFF(Date1, Date2, Interval)

Exercise

- In Customer Table
  - Find the age of the customer
    - Datediff
    - Age = DATEDIFF(AW_Customers_Fact[BirthDate],today(),YEAR)
- In Sales Table
  - Create a column of Weekday/Weekend
    - IF, OR, Weekday
    - Weekday/Weekend = IF(wEEKDAY(AW_Sales_Data[OrderDate],2)>5,"Weekend","Weekday")

Filter Functions

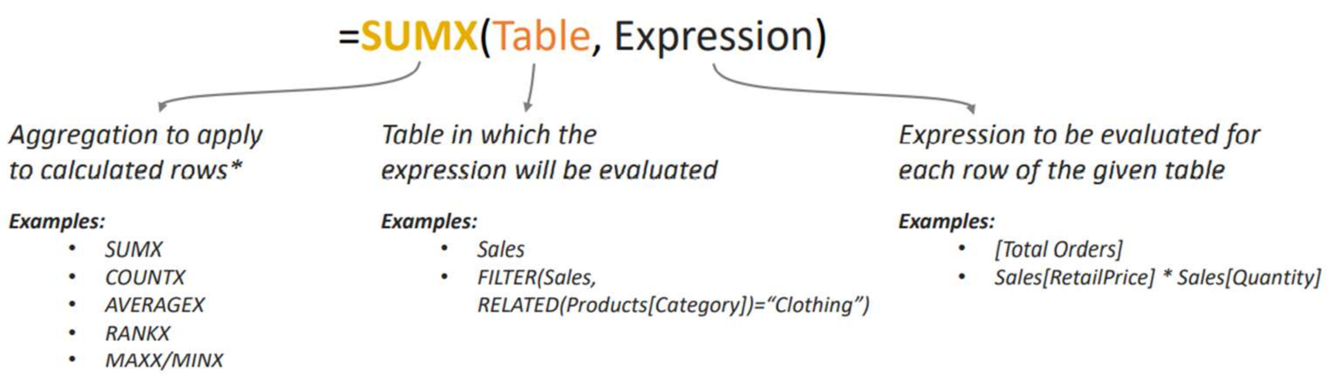
- Common examples:
  - CALCULATE
  - FILTER
  - ALL
  - RELATED
  - RELATEDTABLE
  - DISTINCT

Relate Function

- Returns related values in each row of a table based on relationships with other tables.
- It works like vlookup of excel. However, we just need to mention the column from where we want the value. The reason we don't have to give the lookup value and table array as we have already defined the relationship between the tables in data relationships.
- Since this function requires a row context it can be used in
  - Calculated Column
  - Or, Iterator function that cycles through all the rows of the table (FILTER, SUMX, MAXX etc)
- We should avoid using Related as it make redundant columns just like merging queries and increase the files size.
- We should use it within a measure like FILTER or SUMX.

Iterator Functions

- Allows to loop through the same calculation or expression on each row of table, and then apply Aggregation to the result (Sum, Average, Max etc).
- They allow referencing the columns of the table and not measures since the calculation has to be done row wise.
- `Total Rev_Measure = SUMX(AW_Sales,AW_Sales[Retail Price]*AW_Sales[OrderQuantity])`


=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")*

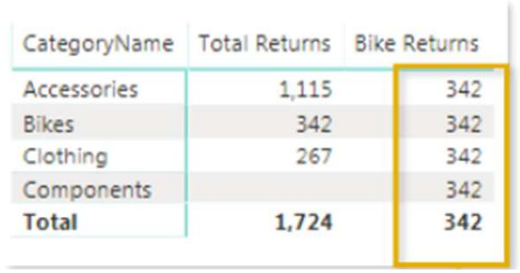
Expression to be evaluated for each row of the given table

Examples:
- *[Total Orders]*
- *Sales[RetailPrice] * Sales[Quantity]*

Calculate Function

- Calculate kind of works as Sumif, Countif etc of excel
- Calculate(measure, Filter1,Filter2…)
  - Measure could be an existing one or calculated expression for valid DAX measure
  - Filters should return only True or False
  - They have to simple and fixed value i.e. column name on left and fixed value on right
  - Can’t create filters based on measures
- Example
  - `Weekend Order = CALCULATE([Total Orders], AW_Calendar_LookUp[Weekend]="Weekend")`

Calculate Function #2



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**"

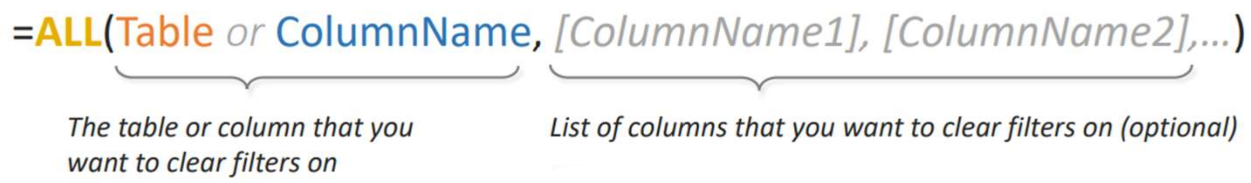
CALCULATE ***modifies*** and ***overrules*** any competing filter context!

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

All Function

- ALL remove filters on entire table or few columns of the table
- It is useful for calculating numbers like grand total
- Removes the filter when dropping in visual



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

The table or column that you want to clear filters on

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

- Since it returns column or table, it is used with functions like Calculate
- Example
  - `All Order = CALCULATE([Total Orders],ALL(AW_Sales))`

Exercise

- Total Cost
- Total Profit
- Total Quantity
- Total Sales
- Sales Amount of Female Customers
- Category wise no. of Bulk Order (where qty>1)
- Bike Return
- Total Return
- % Return
- Returned Sales Amount
- % Loss

Filter Function

- It is also known as Iterator function as it works on each row.
- It is a time taking process on larger dataset. Hence, avoid using FILTER when Calculate can do the job.
- It can handle more complex filter than CALCULATE like include measures product[Price]>[Overall Price]
- Syntax – FILTER(Table, Filter Expression)
- Filter expression should give true false as output
- It returns a table which CALCULATE Doesn't.
  - Since it return a table it is used with functions like Calculate and SumX
- `High Ticket Order = CALCULATE([Total Orders],FILTER(AW_Product_Lookup,AW_Product_Lookup[ProductPrice]>[Overall Avg Price]) )`

Exercise - Answers

- Total Cost - `Total Cost = SUMX(AW_Sales,AW_Sales[OrderQuantity]*RELATED(AW_Products_Fact[ProductCost]))`
- Total Quantity - `Total Qty = SUM(AW_Sales[OrderQuantity])`
- Total Sales - `Total Sales = SUMX(AW_Sales,AW_Sales[OrderQuantity]*RELATED(AW_Products_Fact[ProductPrice]))`
- Total Profit - `total Profit = [Total Sales]-[Total Cost]`
- Unique Products - `Unique Products = DISTINCTCOUNT(AW_Products_Fact[ProductKey])`
- Sales Amount of Female Customers - `Sales_Female = CALCULATE([Total Sales],AW_Customers_Fact[Gender]="F")`

Exercise - Answers

- Category wise no. of Bulk Order (where qty>1) - `Bulk Order = CALCULATE([Total Order],AW_Sales[OrderQuantity]>1)`
- Bike Return - `Bikes Return = CALCULATE([Total Return],AW_Product_Categories_Fact[CategoryName]="Bikes")`
- Total Return - `Total Return = SUM(AW_Returns[ReturnQuantity])`
- % Return - `% Return = [Total Return]/[Total Qty]`
- Returned Sales Amount - `Return Sales = SUMX(AW_Returns,AW_Returns[ReturnQuantity]*RELATED(AW_Products_Fact[ProductPrice]))`
- % Loss - `Loss % = AW_Returns[Return Sales]/[Total Sales]`

Time Intelligence Formula

- These allow to easily calculate common time comparison
- Performance To-Date –
  - CALCULATE(Measure, DATESYTD(Calendar[Date]))
  - Variation DATESQTD for quarter, DATESMTD for months
- Previous Period –
  - CALCULATE(Measure, DATEADD(Calendar[Date],-1, MONTH))
  - Example is for previous month
  - Instead of MONTH we can have DAY, MONTH, QUARTER, YEAR
- Running Total –
  - CALCULATE(Measure, DATESINPERIOD(Calendar[Date], MAX(Calendar[Date]),-10,DAY))
  - Example is for 10-day running total
  - Instead of MONTH we can have DAY, MONTH, QUARTER, YEAR
- To calculate Moving Average, use the running total calculation above and divide by number of intervals.

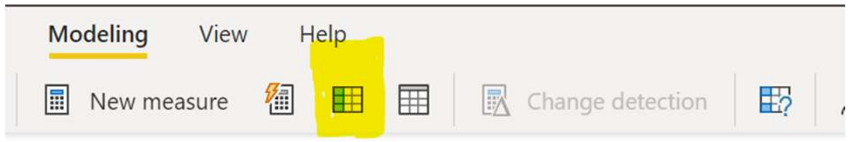
Exercise

- Calculate YTD Revenue

Create Table Using DAX

Create New Table with Values

- To create Table using DAX go to Modeling Tab
  - Click on Create table using DAX



- Syntax for creating table
  - Table with one value only → Table1 = {1}
  - Table with 4 rows → Table1 = {1,2,3,4}
  - Table with 4 columns → Table2 = {(1,2,3,4)}
  - Table with two rows and two columns → Table2 = {(1,2),(3,4)}

Creating Table from Existing Table

- CALCULATETABLE(<expression>, [<filter1>] , [<filter2> [, …]]])
  - Expression is the table from where data will be extracted
  - Filters – condition based on which data will be extracted
- Example
- CALCULATETABLE('InternetSales_USD’, 'DateTime'[CalendarYear] = 2006))

The END