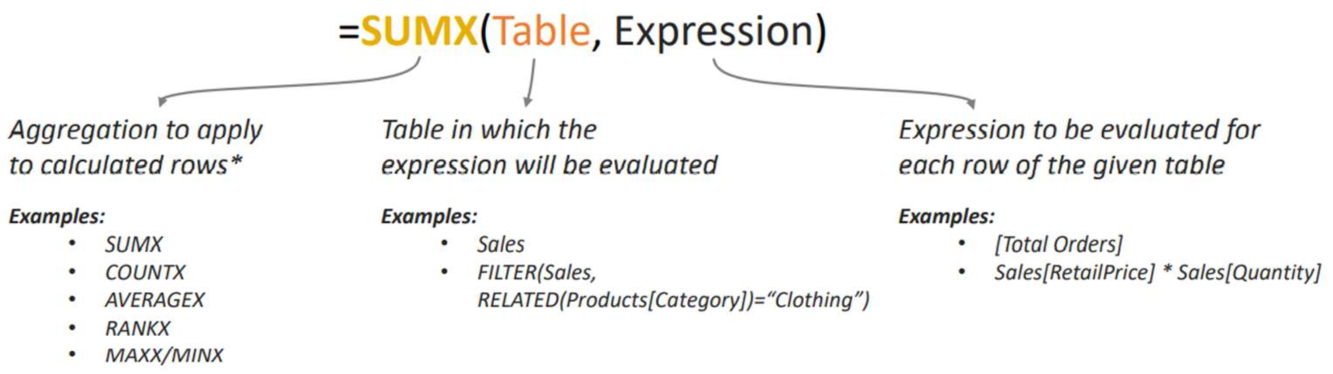
Iterator Functions

- 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

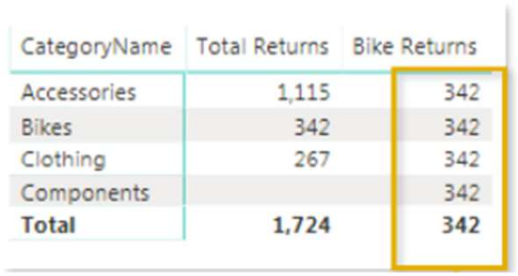
Examples:
- *[Total Orders]*
- *Sales[RetailPrice] * Sales[Quantity]*

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**"

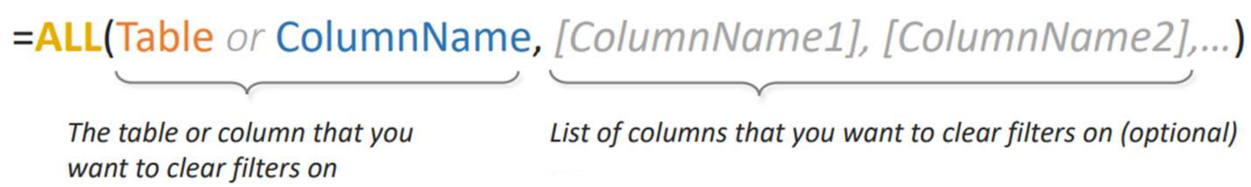
CALCULATE ***modifies*** and ***overrules*** any competing filter context!

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



$$=\text{ALL}(\underbrace{\text{Table } \textit{or} \text{ ColumnName},}_{\textit{The table or column that you want to clear filters on}}\ \underbrace{\textit{[ColumnName1], [ColumnName2],...}}_{\textit{List of columns that you want to clear filters on (optional)}})$$

- 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.

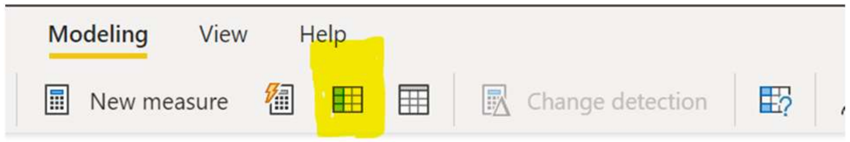
Exercise

- 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))

Professional School
ISO 9001 : 2000 Organisation

The END