IMPORTANT DAX FUNCTIONS YOU SHOULD KNOW!

ZEP ANALYTICS

INTRODUCTION

Data Analysis Expressions, commonly known as DAX, is the formula language that drives Power BI. With DAX, you can:

- Add calculated columns and measures to your model, using intuitive syntax.
- Go beyond the capabilities of traditional “grid-style” formulas, with powerful and flexible functions built specifically to work with relational data models

MEASURES

Measures are evaluated in the context of the cell evaluated in a report or in a DAX query
Some of the points related to Measures:

- Represents a single value per data model
- Computed at run time
- Dynamic results, based on filters
- Filter Context
- Not attached to any specific table

Example:

```
TotalQuantity := SUM(Sales[Quantity])
```

CALCULATED COLUMN

Calculated Column are computed at the row level within the table it belongs to

Some of the points related to Calculated Column:

- Represents a single value per row
- Computed at compile time
- Dynamic Results, based on Rows
- Row Context

Example:

Tenure_Months := Churn[Tenure]*12

IMPLICIT MEASURES

If we use a calculated column as a value/result, it creates an implicit measure.

For example:
If we have columns such as:
▹Tenure in years,
▹Monthly average usage

Goal: to create the overall average usage for that customer

```
Churn[Tenure_Months] = Churn[Tenure]*12
```

Total usage would be:

```
Churn[TotalUsage] = Churn[Tenure_Months]
* Churn[Monthly_Average_Usage]
```

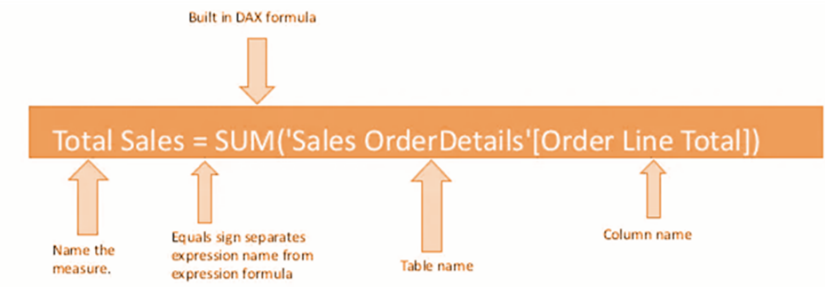
Change in the Primitive Column, i.e. Tenure, will impact the change in the Total Usage column.

SYNTAX OF DAX

DAX is great in two things that are aggregating and filtering. Aggregating means combining a group of values into a single value.

Examples: Sum, Average, Min, Max, Distinct Count

Syntax of DAX measure is shown in figure below



REVISION OF BASIC OPERATORS

Before starting with DAX let's observe some of the basic operators.

Arithmetic Operator

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

Comparison Operator

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"

REVISION OF BASIC OPERATORS

Logical Operator

Pay attention to these!

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" }

Now, let's start with DAX

BASIC DATE & TIME FUNCTIONS

DAY/MONTH/YEAR(): It returns the day of the month (1-31), month of the year (1-12) or year of the given date
Syntax: Name = DAY/MONTH/YEAR(Date)

HOUR/MINUTE/SECOND():Returns the hour (0-23), minute (0 59), or second (0 59) of a given datetime value.
Syntax: Name =HOUR/MINUTE/SECOND(DateTime)

TODAY/NOW() : Returns the current date or exact time
Syntax: Name = TODAY/NOW()

WEEKDAY/WEEKNUM() : Returns a weekday number from 1 (Sunday) to 7 (Saturday), or the week # of the year
Syntax: Name of measure/column = WEEKDAY/WEEKNUM(Date, [Return Type])

BASIC DATE & TIME FUNCTIONS

EOMONTH(): Returns the date of the last day of the month, +/-a specified number of months.
Syntax: Name of measure/Column = EOMONTH(StartDate, Months)

DATEDIFF(): Returns the difference between two dates, based on a selected interval.
Syntax: Name of measure/Column = DATEDIFF(Date1, Date2, Interval)

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.
Syntax: Name of measure/Column = IF(LogicalTest, ResultIfTrue, [ResultIfFalse])

IFERROR(): Evaluates an expression and returns a specified value if the expression returns an error, otherwise returns the expression itself.
Syntax: Name of measure/Column = IFERROR(Value, ValueIfError)

AND(): Checks whether both arguments are TRUE, and returns TRUE if both arguments are TRUE, otherwise returns FALSE.
Syntax: Name of measure/Column = AND(Logical1, Logical2)

LOGICAL FUNCTIONS

OR(): Checks whether one of the arguments is TRUE to return TRUE, and returns FALSE if both arguments are FALSE.
Syntax: Name of measure/Column = OR(Logical1, Logical2)

Note: Use the && and || operators if you want to include more than two conditions.

TEXT FUNCTIONS

LEN(): Returns the number of characters in a string
Syntax: Name of measure/Column = LEN(Text)

CONCATENATE(): Joins two text strings into one
Syntax: Name of measure/Column = CONCATENATE(Text1, Text2)

UPPER/LOWER/PROPER() : Converts letters in a string to upper/lower/proper case
Syntax: Name of measure/Column = UPPER/LOWER/PROPER(text)

SUBSTITUTE(): Replaces an instance of existing text with new text in a string.
Syntax: Name of measure/Column = SUBSTITUTE(Text, OldText, NewText, [Instance Number])

RELATED FUNCTION

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

Syntax: Name of measure/Column = RELATED (ColumnName)

RELATED works almost exactly like a VLOOKUPfunction –it uses the relationship between tables (defined by primary and foreign keys) to pull values from one table into a new column of another

BASIC MATH FUNCTIONS

SUM() : Evaluates the sum of a column
Syntax: Name of measure/Column =
SUM(ColumnName)

AVERAGE() : Returns the average (arithmetic mean) of all the numbers in a column
Syntax: Name of measure/Column =
AVERAGE(ColumnName)

MAX() : Returns the largest value in a column or between two scalar expressions
Syntax: Name of measure/Column =
MAX(ColumnName)

MIN() : Returns the smallest value in a column or between two scalar expressions
Syntax: Name of measure/Column =
MIN(ColumnName)

DIVIDE() : Performs division and returns the alternate result (or blank) if div/0
Syntax: Name of measure/Column =
DIVIDE(Numerator, Denominator, [AlternateResult])

COUNT FUNCTIONS

COUNT() : Counts the number of cells in a column that contain numbers.
Syntax: Name of measure/Column = COUNT(ColumnName)

COUNTA() : Counts the number of non-empty cells in a column (numerical and non-numerical).
Syntax: Name of measure/Column = COUNTA(ColumnName)

DISTINCTCOUNT() : Counts the number of distinct or unique values in a column.
Syntax: Name of measure/Column = DISTINCTCOUNT(ColumnName)

CALCULATE FUNCTIONS

CALCULATE(): Evaluates a given expression or formula under a set of defined filters

Syntax:
Name of the column/measure =
CALCULATE(Expression, [Filter1], [Filter2],..)

CALCULATE works just like SUMIF or COUNTIF in Excel, except it can evaluate measures based on ANY sort of calculation (not just a sum, count, etc); it may help to think of it like "CALCULATEIF"

FILTER FUNCTIONS

FILTER(): Returns a table that represents a subset of another table or expression.

Syntax:
Name of column/measure = FILTER(Table, Filter Expression)

FILTER is used to add new filter context, and can handle more complex filter expressions than CALCULATE (by referencing measures, for example) Since FILTER returns an entire table, it's almost always used as an input to other functions, like CALCULATE or SUMX

ALL FUNCTIONS

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

Syntax:
Name of column/measure = ALL(Table or ColumnName, [ColumnName1], [ColumnName2],....)

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)

ITERATOR("X") FUNCTIONS

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

Syntax:
Name of column/Measure = SUMX(Table, Expression)

Iterator can be applied to Aggregating functions.

TIME INTELLIGENCE FUNCTIONS

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

Performance To Date

Syntax: Name of column/Measure = CALCULATE(Measure, DATESYTD(Calendar[Date]))

Previous Period

Syntax: Name of column/Measure = CALCULATE(Measure, DATEADD(Calendar[Date], -1, MONTH))

Running Total

Syntax: Name of column/Measure = CALCULATE(Measure, DATESINPERIOD(Calendar[Date], MAX(Calendar[Date]), -10, DAY))

Ready to take the next steps?

So these were some of the important DAX functions which you should remember.

Remember DAX is very vast and there are many more but these are the basic and the most used ones.