DAX

POWER BI

What is DAX?

- Data Analysis Expressions is a formula expression language used in Power BI.
- DAX formulas include functions, operators and values to perform advanced calculations and queries.
- Microsoft based it off of Excel formulas.
- It provides the ability to create programmatically generated columns, tables, measures and even for row-level security.

What are differences between Measures and Calculated Columns.?

Calculated Columns :

- For evaluating each row.
- Add a new column to an existing table.
- Calculated on data import.

Measures :

- For aggregating multiple rows.
- Results in another field that you can add to a visualization.
- Calculated at query run-time.

What are differences between Implicit and explicit measures

Implicit:

- Automatically created by Power BI and come directly from the database.

Explicit :

- Written measures.
- Reusable within other measures
- Can be given a custom name to explain its functionality.
- More flexible.

How to use variables in DAX ?

Stores the result of an expression as a named variable to be used as an argument to other measure expressions.
Used to Improve performance, Improve readability, Simplify debugging and Reduce complexity.

NEW TABLE DAX EAMPLE

Create a new empty table

```
_measure =
```

Create new table identical to an existing table

```
Ship_date = order_date
```

Create a new date table have a one date key column from and to indicated dates calculated from min and max exiting date column

```
DimCalender = CALENDAR( MIN( ‘Fact StrategyPlan’[DateKey] ),
                        MAX( ‘Fact StrategyPlan’[DateKey] ) )
```

We have a fact table and dimensions tables and you want to create a table consists of columns from all tables and then filter it and a new calculated column

Create table consists of columns from another table

```
CaliforniaPR = SUMMARIZE( Fact_Sales ,
                          Dim_Employee [Employee] ,
                          Dim_InvoiceDate[CalendarYear] ,
                          Dim_City[StateProvince] )
```

Add a filter

```
CaliforniaPR = FILTER( SUMMARIZE( Fact_Sales ,
                                  Dim_Employee [Employee] ,
                                  Dim_InvoiceDate[CalendarYear] ,
                                  Dim_City[StateProvince] ) ,
                       Dim_City[State Province] = "California" )
```

Add a calculated column

```
CaliforniaPR = ADDCOLUMNS ( FILTER( SUMMARIZE( Fact_Sales ,
                                               Dim_Employee [Employee] ,
                                               Dim_InvoiceDate[CalendarYear] ,
                                               Dim_City[StateProvince] ),
                                    Dim_City[State Province] = "California" ),
                            "Profit Ratio” , [Profit Ratio] )
```

NEW COLUMN DAX EAMPLE

Create a column that's extract the number of character from the beginning of another column

```
Login_ID = LEFT( DimCustomer[FirstName] , 2 )
```

Create a column that's extract the numbers of character from the end of a another column

```
TempPass = RIGHT( DimCustomer[Phone] , 4 )
```

Create a column that's concatenate the first 2 character from a column and another column

```
Login_ID = LEFT( DimCustomer[FirstName] , 2 ) & DimCustomer[LastName]
```

Create a column that's make the last 4 character from a column lowercase

```
TempPass = LOWER( RIGHT( DimCustomer[Phone] , 4 ) )
```

Create a column that replace a part of a string by another

```
Entityshort = SUBSTITUTE( Dim_Entity [Entity Description] , "Contoso" , "" )
```

String Functions

COMBINEVALUES	Joins two or more text strings into one text string.
CONCATENATE	Joins two text strings into one text string.
EXACT	Compares two text strings and returns TRUE if they are exactly the same, FALSE otherwise.
FIND	Returns the starting position of one text string within another text string.
LEFT	Returns the specified number of characters from the start of a text string.
LEN	Returns the number of characters in a text string.
LOWER	Converts all letters in a text string to lowercase.
MID	Returns a string of characters from the middle of a text string, given a starting position and length.
REPLACE	Replaces part of a text string, based on the number of characters you specify, with a different text string.
REPT	Repeats text a given number of times.
RIGHT	Returns the last character or characters in a text string, based on the number of characters you specify.
SEARCH	Returns the number of the character at which a specific character or text string is first found, reading left to right.
SUBSTITUTE	Replaces existing text with new text in a text string.
TRIM	Removes all spaces from text except for single spaces between words.
UPPER	Converts a text string to all uppercase letters.
VALUE	Converts a text string that represents a number to a number.

Use the relation between two tables to get the corresponding value from the other table in a new column

```
Scenario = RELATED( Dim_Scenario[ScenarioDescription] )
```

Create a column depend on a condition

```
Performance = IF( [Total_Sales] >= 50 000 , "Target Reached" , "Target Not Reached" )
```

Create a column depend on multiple conditions

```
Performance = SWITCH( TRUE,
                      [Total_Sales] < 25 000 , "Poor" ,
                      [Total_Sales] < 50 000 , "Below expectations" ,
                      [Total_Sales] < 75 000 , "Above expectations" ,
                      "Exceptional" )
```

```
DISCOUNT = SWITCH( [Clothing Type] ,
                   "T-shirt" , 0.15,
                   "Pants" , 0.20,
                   "Belts" , 0.30,
                   "Shoes" , 0.25)
```

Create a ranking column "Actual Transaction Rank " based on specific measure "Actual Transaction Amount " and check if "Product CategoryName " have a single value or more than one aggregated and if it has more than one then the value is blank instead of the ranking

```
Actual Transaction Rank =
        IF( HASONEVALUE( Dim_ProductCategory [Product CategoryName] ) = TRUE,
           RANKX( ALL( 'Dim_Product Category'[Product CategoryName] ),
                   [Actual Transaction Amount] ),
           BLANK())
```

Logical Functions

AND	Checks whether both arguments are TRUE, and returns TRUE if both arguments are TRUE.
OR	Checks whether one of the arguments is TRUE to return TRUE.
NOT	Changes FALSE to TRUE, or TRUE to FALSE.
IF	Checks a condition, and returns one value when TRUE, otherwise it returns a second value.
SWITCH	Evaluates an expression against a list of values and returns one of multiple possible result expressions.

NEW MEASURE DAX EAMPLE

Create a new measure that's count rows

```
Transaction Count = COUNTROWS( 'Fact StrategyPlan‘ )
```

Create a new measure that take the average of a column with a filter

```
Actual Average Transaction Value =
CALCULATE ( AVERAGE( 'Fact_ StrategyPlan' [Amount] ) , 'Fact_ StrategyPlan'[Scenario] = "Actual")
```

Create a new measure equal sum of a column with a all filter

```
Total Amount = CALCULATE ( SUM( 'Fact_Strategy Plan' [Amount] ) , ALL( Fact_StrategyPlan ) )
```

Aggregation Functions

AVERAGE	Returns the average of all the numbers in a column.
COUNT	Counts the number of cells in a column that contain numbers.
COUNTA	Counts the number of cells in a column that are not empty.
COUNTBLANK	Counts the number of blank cells in a column.
COUNTROWS	Counts the number of rows in the specified table, or in a table defined by an expression.
DISTINCTCOUNT	Counts the number of distinct values in a column.
MAX	Returns the largest numeric value in a column, or between two scalar expressions.
MIN	Returns the smallest numeric value in a column, or between two scalar expressions.
SUM	Adds all the numbers in a column.

Create a measure that count a distinct values of a dimension table column but with a cross filter to filter the fact table when a relation between two dimension tables is used

```
Product Category Count =
CALCULATE( DISTINCTCOUNT( Dim_Product Category [Product CategoryDescription] ) ,
             CROSSFILTER( Dim_ProductCategory [Product CategoryKey],
                          Fact_StrategyPlan[Product Categorykey],
                          BOTH ) )
```

Create a measure that sum a calculated column directly

```
Total Costs SUMX = SUMX( Fact_Orders , Fact_Orders[Sales] - Fact_Orders[Profit] )
```

Create a measure that sum a calculated column directly with a specific value filtration

```
Total Costs East SUMX = SUMX( FILTER( Fact_Orders , Fact_Orders[Region] = "East“ ),
                                      Fact_Orders[Sales] - Fact_Orders[Profit] )
```

Create a measure that rank based on a measure with a non-selecting filter

```
Total Costs RANKX = RANKX( ALL( Dim_Sales[Region] ) , [Total Costs] )
```

Aggregation Functions

AVERAGEX	Calculates the average of a set of expressions evaluated over a table.
COUNTAX	Counts nonblank results when evaluating the result of an expression over a table.
COUNTX	Counts the number of rows that contain a number or an expression that evaluates to a number, when evaluating an expression over a table.
MAXX	Evaluates an expression for each row of a table and returns the largest numeric value.
MINX	Returns the smallest numeric value that results from evaluating an expression for each row of a table.
SUMX	Returns the sum of an expression evaluated for each row in a table.

Create a measure that's calculate the change of sales between the current time and the same time last year

```
Sales Growth =

VAR
        SALESPRIORYEAR = CALCULATE( [SALES] , SAMEPERIODLASTYEAR( 'DATE‘ ) )
RETURN

[Sales] - SALESPRIORYEAR
```

Create a measure that's calculate the change in percentage of sales between the current time and the same time last year

```
Total Sales YoY%  =

VAR
  Sales_Last Year = CALCULATE( [Total Sales] , SAMEPERIODLASTYEAR( Dim_InvoiceDate[Date] ))
RETURN

DIVIDE( [Total Sales] - Sales_Last Year , Sales_Last Year)
```

Create a year running total month by month of a measure

```
Total Sales YTD = TOTALYTD( [Total Sales], Dim_InvoiceDate[Date])
```

Create a last year running total month by month of a measure

```
Total Sales YTD Last Year = TOTALYTD( [Total Sales] ,
                                      SAMEPERIODLASTYEAR( Dim_InvoiceDate [Date] ) )
```

Time Intelligence Functions

TOTALMTD	Evaluates the value of the expression for the month to date, in the current context.
TOTALQTD	Evaluates the value of the expression for the dates in the quarter to date, in the current context.
TOTALYTD	Evaluates the year-to-date value of the expression in the current context.