CountA :
counts the number of non-blank values in a column; regardless of data type
count the non-blank values in a column regardless of numerical or non-numerical

Syntax: COUNTA [<column>]

COUNTX:
is used to count the number of rows in a table where an expression evaluates to a non-blank value
useful when you need to apply some calculation or condition before counting the rows.

Syntax: COUNTX [<table>, <expression>]

COUNTBLANK
is used to count the number of blank (or missing) values in a column.
useful for data quality checks and identifying missing data within your dataset.

Syntax: COUNTBLANK [<column>]

DISTINCTCOUNT
used to count the number of unique non-blank values in a column.
useful for scenarios where you need to determine the number of unique items such as

unique customers, products or any other unique entries in a dataset

Syntax: DISTINCTCOUNT[<column>]

MAX:
returns the largest numerical value in a column

syntax: MAX[<column>]

MAXA:
returns the largest value in a column, but it can handle both numeric and non-numeric data types including text, logical values and blanks

Syntax: MAXA[<column>]

MAXX:
evaluates an expression for each row of a table and returns the largest numerical value.
useful when you need to apply a calculation to each row before finding the maximum value.

Syntax: MAXX[<table>, <expression>]

Ex A sales table with columns Quantity, Price and Discount
Find maximum revenue after discount.

Max. Revenue = MAXX[sales, sales[Quantity]* Sales[Price]* [1- sales[Discount]]]

Date: YOUVA

AVERAGE :
returns the arithmetic mean [average] of the the numeric values in a column

Syntax: AVERAGE [<column>]

AVERAGEA :
returns the arithmetic mean of the values in a column; treating TRUE as 1; FALSE as 0 and ignoring text except for BLANKS

Syntax: AVERAGEA [<column>]

AVERAGEX :
calculates the average of an expression evaluated for each row of a table

Syntax: AVERAGEX [<table>, <expression>]

COUNTROWS
returns the count of number of rows in a table

Syntax: COUNTROWS ([<table>])

Date:

Date and Time Functions:-

are essential for manipulating and analysing date and time data, performing calculations based on dates, and creating meaningful insights in POWER BI

CALENDAR:
generates a date table from a specified start date to a specified end date
Control: Full control over the exact range of dates by specifying the start and end dates.
Flexibility: Less flexible since you need to manually set the start and end dates.
Fiscal Year Support: Does not have built-in support for fiscal year-end specification.

Syntax: CALENDAR(<start_date>, <end_date>)

Example: CALENDAR (DATE (2024, 1, 1), DATE (2024, 12, 31))

Scenario: Used to create a custom date table, which is essential for time-based calculations and analysis.

CALENDARAUTO:
automatically generates a date table based on the minimum and maximum dates in the model.
Control: Less control over the exact range of dates as it adapts to the data in the model
Flexibility: More flexible because it automatically adjust to the date range present in the data model.

Fiscal Year Support: Allows specifying a fiscal year-end month, which adjust the date range accordingly.

Can quickly generate a date table without specifying dates manually.

Syntax: CALENDARAUTO [CfiscalYearEndMonth]]

Example: CALENDARAUTO[6]
//Adjusts for fiscal year ending in June

Scenario: Useful when the date range is not known in advance or when you want the date table to dynamically adapt to the data model.

DATE:
returns the specified date in datetime format
Used to create a date from individual year, month and day components.

Syntax: DATE[<year>, <month>, <day>]

Example: DATE [2024, 6, 30]

DATEDIFF:
returns the number of interval boundaries between two dates.
Useful for calculating the difference between two dates in terms of days, months, years etc.

Syntax: DATEDIFF [<start_date>, <end_date>, <interval>]

Example: DATEDIFF [Order[OrderDate], Delivery[DeliveryDate], DAY]

DATEVALUE:
Converts a date in the form of text to a date in datetime format.
To convert date strings to datetime values for further calculations.

Syntax: DATEVALUE [<date_text>]

Use case: Convert a date string from a CSV import to a datetime value.

Example: DATEVALUE ["2024-06-30"]

DAY:
returns the day of the month, a number from 1 to 31
Extracts the day component from a date

Syntax: DAY [<date>]
- Extracting the day part of a transaction

Example: DAY [Transaction [TransactionDate]]

MONTH:
Returns the month as a number from 1 [January] to 12 [December]
Extracts the month component from a date

Syntax: MONTH [<date>]

Extracting the month part of a sales date
Example: MONTH [Sales [SaleDate]]

YEAR:
returns the year of a date as a four-digit integer in the range 1900-9999.

Syntax: YEAR [<date>]

Extracting the year part of a hire date.
Example: YEAR [Employee [HireDate]]

QUARTER:
Returns the quarter as a number from 1 to 4

Syntax: QUARTER [<date>]

Determining the quarter in which a sale was made
Example: QUARTER [Sales [SaleDate]]

HOUR:
Return the hour as a number from 0 [12:00AM] to 23 [11:00PM]
Used to extract the hour component from a time value, which can be useful in time-based analysis.

Syntax: HOUR [<datetime>]

Use Case: Analysing the distribution of transactions across different hours of the day

Example: HOUR [Transaction [Transaction Time]]
(if transaction time is "14:35:20", this returns 14)

MINUTE:
Return the minute as a number from 0 to 59
Used to extract the minute component from a time value for detailed time analysis

Syntax: MINUTE [<datetime>]

Use Case: Determining the minute part of a timestamp for detailed logging or monitoring purposes.

Example: MINUTE [Transaction [TractionTime]]
(if transaction time is "14:35:20", this return 35)

SECOND:
Returns the seconds of a time value as a number from 0 to 59.
Used to extract the seconds component from a time value useful in precise-time-based analysis.

Syntax: SECOND[<date_time>]

Use Case: Extracting the second part of a timestamp for high precision logging.

Example: SECOND [Transaction[Transaction Time]]
(If transaction time is "14:35:20", this returns 20)

TODAY:
description: returns the current date.
Useful for calculations that require the current date such as aging analysis or daily reports.

Syntax: TODAY[]

Use Case: Calculating the number of days since a specific event.

Example: TODAY[]
(If today is June 24, 2024, this returns 2024-06-24)

NOW:
Returns the current date and time
Used for calculations that requires both the current date and time, such as real-time monitoring and updates.

Syntax: NOW[]

Use Case: Determining the exact timestamp of the latest data refresh

TIME:
Converts hours, minutes and seconds given as numbers to a time in datetime format.
Useful for constructing time values from separate hour, minute and second components.

Syntax: TIME [<hour>, <minute>, <seconds>]

Use Case: Creating a time value for a specific event

Example: TIME [14, 35, 20]

TIMEVALUE:
Converts a time in text format to a time in a datetime format
Useful for converting time strings from data imports into datetime format for analysis

Syntax: TIMEVALUE [<time_value>]

Use case: Converting a time string from a CSV import to a datetime value.

Example: TIMEVALUE ["14:35:20"]
// Returns 14:35:20

Date: YOUVA

WEEKDAY

returns the day of the week for a given date value. It can return a number from 1 to 7, where the number corresponds to a specific day of week

Syntax: WEEKDAY (<date>, <return_type>)

Example: WEEKDAY (Orders[OrderDate], 1)
(return a number from 1 (sunday) to 7 (saturday) for each date)

WEEKNUM

returns the weeknumber for a given date value. This function helps find which week of the year a particular date falls into.

Syntax: WEEKNUM (<date>, <return_type>)

Example: WEEKNUM (Orders[OrderDate], 1)
(Returns the week number for each date, assuming weeks start on sunday)

NETWORKDAYS

return the number of whole workdays between two dates, excluding weekends and holidays.
This is useful for calculating business days between two dates

Syntax: NETWORKDAYS (<start_date>, <end_date>[, <weekend>, <holidays>])

start_date : start date
end_date : end date
weekend (optional): Indicates which day are considered weekends. Default is 1 (saturday, sunday).
Holidays (optional): A column of dates to be excluded from the working days

Example -
NETWORKDAYS ('Order'[OrderDate], 'Order'[DeliveryDate], 1, Holiday)

CALENDAR	CALENDARAUTO
generates a date table from a specified start date to a specified end date.	Automatically determines the start and end dates based on the data in the model.
You have control over the exact range of dates.	More flexible as it adapts to the date range in the data model.
Does not have built-in support for fiscal year and specification	Allows specifying a fiscal year end month, which adjusts the date range accordingly.

DAX

Page No.: YOUVA
Date:

Aggregation Function –

are essential for summarizing data, performing mathematical operations and creating meaningful insights in POWER BI.

SUM

used to calculate the total sum of a single column
It adds up all the values in the specified column

Syntax: SUM [<column>]

SUMX

It is more flexible and powerful.
It is an iterator function, meaning it iterates over a table, evaluates an expression for each and then returns the sum of the results.

Syntax: SUMX [<table>, <expression>]

COUNT

is to count the non-blank numerical values in a column and excludes the blank and Null values in a column.

Syntax: COUNT [<column>]