Sec. - 4

* Data model :-
The tables are connected via relationships based on a common field.

* Database Normalization :-

Normalization is the process of organizing the tables and columns in a relational database to reduce redundancy and preserv data integrity.

⇒ Its commonly used to :-

- Eliminate redundant data to decrease table size and improve processing speed & efficiency.

- Minimize errors and anomalies from data modifications (inserting, updating or deleting records)

Fact & Dimension Tables

* Data models generally contain two types:-

  fact ("data") table, and
  dimension ("lookup") tables:-

- Fact table contain numerical values or metrics used for summarization.

- Dimension table contain descriptive attribute used for filtering or grouping.

Textual e.g.

Primary & foreign key

→ Primary key :—

Primary keys (PK) they uniquely identify each row of the table, and relate to foreign keys in fact tables.

- there is no duplicates.

→ Foreign key :- They contain multiple instances of each value, and relate to primary keys, in dimension tables.

DAX (Data Analysis expression)

Sec:- 5

→ DAX :- Data Analysis Expression (commonly known as DAX) is the formula language that drives the powerbi front-end.

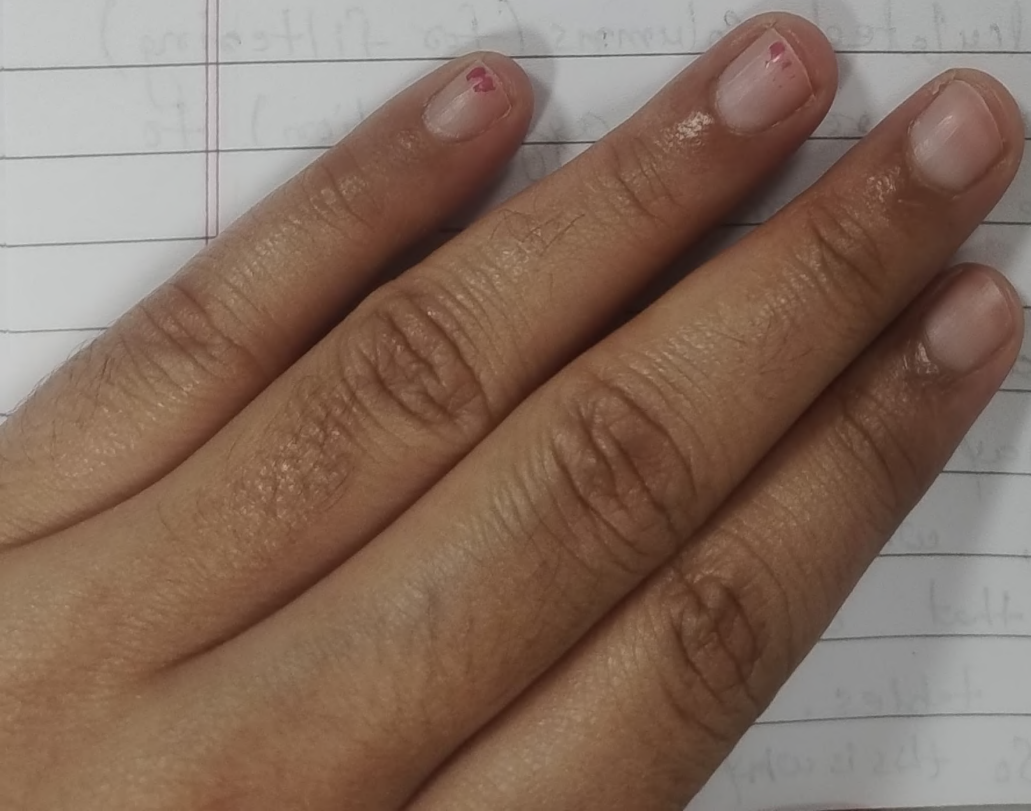
USES :-

* with Dax, you can go beyond the capabilities of traditional spreadsheet formulas, with powerful and flexible functions built specifically to work relational databases OR data models.

* Add calculated columns (for filtering) and measures (for aggregation) to enhance data model.

* we can use DAX in two ways :-
first way is to create Calculated Column, which are new column of data that actually live inside your tables.
So this is why calculated columns are used for filtering.

The Second way to use DAX is to Create measures, which are then used in visulas to show some sort of aggregation, like a sum, count, or average.



DAX code Vs M. Code

* M and DAX are two distinct functional languages used within Powerbi desktop.

- M :- is used in the power Query (backend) editor, and is designed specifically for extracting, transforming and loading data into powerbi's data model.

- DAX :- DAX is used in the Power BI front-end, and is designed Specifically for analyzing relational data models.

DAX

1. DAX is used for data modeling and creating Calculated column and measures.

2. Dax is used to create new column and measures within your data model. It can also be used to define relationship between tables and perform complex problem.

3. DAX is not designed for data transformation and should not be used for those tasks.

4. DAX is a formula language designed for Creating Custom calculations. It supports a wide range of functions and operators for performing Calculation on you data.

M. (Power Query)

1. M is used for data transformation and loading data from external resources.

2. M is used to load data from external source, transform it, and then load it into your data model.

3. M is designed for data transformation and can perform a wide range of data cleasing and shaping operations.

4. M also supports customs calculations, but it's primarily used for data transformation and should be used for these task.

* Calculated Column :- its allow you to add new, formula-based column to tables in a model.

- Calculated column refer to entire tables or columns (No A1-style cell reference)

- Calculated columns generate values for each row, which are visible within tables in the data view.

- Calculated columns understand row context; (they can see the information contained within each row in which they calculated And so they're great for defining things like properties based on information within that row).

but generally useless for aggregation

likes sum or count or avg..

* Imp: RULE :-

Use calculated columns to "stamp" static, fixed values to each row in a table (or go upstream and use the query editor).

DO NOT use calculated columns for aggregation, or try to calculate numerical vales to use in value area of a visualization.

we'll use measures for that instead.

→ Calculated columns are typically used for filtering data, giving new fields that can use to create those filters rather than creating or defining new numerical values.

DAX MEASURES

* Measures are DAX formula used to generate new calculated values.

- Like calculated columns, Measures reference entire tables or columns (no A1-style cell references)

- Unlike calculated columns, measures are not visible within tables; they can only be "seen" within a visualization like a chart or matrix (similar to a calculated field in a pivot table)

- Measures are evaluated based on filter context, which means they recalculate when the fields or filters around them change.

Report page

classmate
Date
Page

* Imp RULE :- Use measures when a single row can't give you the answer, or when you need to aggregate values across multiple rows in a table.

→ Use measures to create numerical, calculated values that can be analyzed in the "value" field of a report visual.

Implicit Vs Explicit measures

- **Implicit Measures** are created when you drag raw numerical fields into a report visual and manually select an aggregation mode (sum, min, max, count, Avergate, etc.)

- **Explicit measures** are created when you actually write a DAX formula and define a new measures that can be used within the model.

IMP :-

Implicit measures are only accesible within the **specific visualization** in which they were created, and cannot be refereced elsewhere.

Explicit measures can be used anywhere in the report, and

f_x = Calculated column

(sigma) Σ = classmate aggregated

(Little calculator symbol) = explicit measure

referenced by other DAX Calculation to create "measures trees".

Quick Measures

Quick measures automatically creates formulas based on pre-built templates or natural language prompts.

or

Quick measures are a tool that automatically creates explicit measures using pre-built templates or natural language queries.

→ Quick Measures Calculations can be used to build measures using predefined templates (weighted averages, percent difference, time intelligence, etc.)

- Quick measures Suggestions can be used to find suggested based on natural language queries.
  (i.e. " s

* Quick measures can be a great learning tool for beginners or for building more complex formulas but use them with caution;

Calculated Column Vs Measures

Calculated Columns	Measures
• Values are calculated based on information from each row of a table (row context).	• Values are calculated based on information from any filters in the reports (filter context)
• Appends static values to each row in a table and stores them in a model (which increase file size)	• Does not create new data in the tables themselves (doesn't increase file size)
• Recalculate on data source refresh or when changes are made to component columns	• Recalculate in response to any change to filters within the report. or any time the filter context changes, the measures update.

• Calculated column are primarily used for things like slicers or filters, you often use them as dimensions to change the way that you're viewing the data. or Primarily used for filtering data in reports.	• Primarily used for aggregating values in report visuals.
• It is live inside the table	• when we create measures those measures live ~~inside~~ within the visuals.

Measures Table

- Create a dedicated table to store your measures ; → (explicit measure)
  This will help you stay organized, find measures quickly, and allow you to group related measures into folders.

- there is 2 different ways to creating dedicated measures.

(i) Enter data into power Query (loads the table to the data modal - table is visible in power query)

- it is loaded into data model and it's actually visible from within the query editor.

- You're creating it in the front-end and it's actually loaded into the back as well

data view

(ii) Create a Calculated table using DAX directly in the model (table is not visible in power query)

= The table is not visible in power query editor.

Filter Context

- Measures are evaluated based on filter context, which means that they recalculated whenever the fields or filter around them change.

Common Function Categories

- MATH & STATS Function :- Function used for aggregation or iterative, row-level calculation.

→ Common Examples :-

- SUM
- AVERAGE
- MAX / MIN
- DIVIDE
- COUNT / COUNTA
- COUNTROWS
- DISTINCTCOUNT

→ Iterator Functions :-

- SUMX
- AVERAGEX
- MAXX / MINX
- RANX
- COUNTX

- LOGICAL Functions :- Function that use Conditional expressions. (IF / Then statement)

→ Common Examples :-

- IF
- IFERROR
- OR
- NOT
- SWITCH
- TRUE
- FALSE

- TEXT Functions :- Functions used to manipulate text strings or value formats.

→ Common Examples :-

- CONCATENATE
- FORMAT
- UPPER / LOWER
- SEARCH / FIND
- SUBSTITUTE
- COMBINEVALUES
- LETT / MID / RIGHT
- LEN
- REPLACE
- TRIM

- FILTER Functions :- Functions used to manipulate table and filter contexts.

Common eg:-

- CALCULATE
- ALL
- ALLSELECTED
- REMOVEFILTERS
- FILTER
- ALLEXCEPT
- KEEPFILTERS
- SELECTEDVALUE

- TABLE Functions :- Functions that Create or manipulate tables and output tables Vs scalar values

- Common Example :-
  - SUMMARIZE
  - ADDCOLUMNS
  - GENERATESERIES
  - DISTINCT
  - VALUES
  - UNION
  - INTERSECT
  - TOPN

- DATE & TIME functions :- Functions used to manipulate date & time values or handle time intelligence calculations.

Common Examples :-

- DATE
- YEARFRAC
- DAY / HOUR
- WEEKDAY
- NETWORKDAYS
- DATEDIFF
- YEAR / MONTH
- TODAY / NOW
- WEEKNUM

- Time Intelligence :-
  - DATESYTD
  - DATESMTD
  - DATEADD
  - DATESBETWEEN

- RELATIONSHIP Functions :- Functions used to manage & modify table relationship.

Common Examples :-

- RELATED
- RELATEDTABLE
- CROSSFILTER
- USERELATIONSHIP

BASIC MATH & STATS FUNCTIONS

- SUM :- Evaluate the sum of a colu
  = SUM (columnName)

- AVERAGE :- Returns the average (arithmeti
  mean) of all the numbers in a colu
  = AVERAGE (Column Name)

- MAX :- Returns the largest value in a
  column or between two scalar expressions
  = MAX (columnName Or Scalar 1, [Scalar 2])

- MIN :- Returns the smallest value in a
  Column or between two scalar expressi
  = MIN (ColumnName Or Scalar 1, [scalar 2])

- DIVIDE :- Performs division and returns
  the alternate result (or blank)
  if Div/0
  = DIVIDE (Numerator, Denominator,
  [AlternateResult])

COUNTING FUNCTIONS

- COUNT :- Counts the number of non-empty cells in a column. (excluding Boolean values)
  = COUNT (COLUMN Name)

- COUNTA :- Counts the number of non-empty cells in a column. (Including Boolean values)
  = COUNTA (column Name)

- DISTINCTCOUNT :- Counts the number of distinct value in a column.
  = DISTINCTCOUNT (column Name)

- COUNTROWS :- Counts the number of rows in the specified table, or a table defined by an expression.
  = COUNTROWS ([Table])

BASIC LOGICAL FUNCTION

- IF :- Checks if a given condition is met and returns one value if the condition is TRUE, and another if the condition is FALSE

= IF (Logical Test, Result If True, [Result If False])

- IFERROR :- Evaluates an expression and returns a specified value if it returns an error, otherwise returns the expression itself.

= IFERROR (Value, Value if Error)

- SWITCH :- Evaluates an expression against a list of values and returns one of multiple expression possible

= SWITCH (Expression, Value1, Result1, ... [Else])

THE SWITCH FUNCTION

SWITCH = Evaluates an expression against a list of values and returns one of multiple possible expressions.

= SWITCH (Expression, Value1, Result1, ..., [E

Any DAX expression that returns a single scalar value, evaluated multiples times.

eg:-

- Calendar [Month ID]
- 'Product Lookup' [category]

Value returned if the expression doesn't match any Value argume

List of values produced by the expression, each paired with a result to return for rows/cases that match

eg:-

= SWITCH (Calendar [Month ID],
1, "January"
2, "February"

NOTE :-

SWITCH (TRUE) is a common DAX pattern to replace multiple nested IF statements

TEXT FUNCTIONS

- LEN :- Returns the number of characters in a string.

  = LEN (Text)

- CONCATENATE :- Joins two text strings into one.

  = CONCATENATE (Text1, Text2)

> Note :- Use the & operator as a shortcut, or to combine more than two strings

- UPPER/LOWER :- Converts a string to upper or lower case.

  = UPPER / LOWER (Text)

- LEFT/RIGHT/MID :- Returns a number of characters from the start/middle/end of a text string.

= LEFT/RIGHT (Text, [NumChars])
= MID (Text, startPosition, NumChars)

- SUBTITUTE :- Replaces an instance of existing text with new text in a string.

= SUBTITUTE (Text, OldText, NewText, [InstanceNumber])

- SEARCH :- Returns the position where a specified string or character is found, reading left to right.

= SEARCH (FindText, WithinText, [startPosition], [NotFoundValue])

BASIC DATE & TIME FUNCTIONS

- TODAY/NOW :- Returns the current date or exact time.

= TODAY/NOW()

- DAY/MONTH/YEAR :- Returns the day of the month (1-31), month of the year (1-12), or year of a given date.

= DAY/MONTH/YEAR(Date)

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

= HOUR/MINUTE/SECOND (Datetime)

- WEEKDAY/WEEKNUM :- Returns a weekday number from 1 (Sunday) to 7 (Saturday), or the week # of the year.

= WEEKDAY/WEEKNUM (Date, [Return Type])

EOMONTH :- Returns the 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 a given interval. (day, hour, year, etc.)

= DATEDIFF (Date1, Date2, Interval)

Joining Data with RELATED

- RELATED :- Return related values in each row of a table based on relation-ship with other tables

→ RELATED works on the one side of a one to many table relationship

= RELATED (ColumnName)

↓

The Column from a related table containing the values you want to retrieve.

eg :- • 'Product Lookup' [Product Name]
• 'Territory Lookup' [Country]

NOTE :- Related works like VLOOKUP function in Excel - It uses the relationship between tables (defined by primary and foreign keys) to pull values from one table into a new column of another

Since this function requires row context, it can only be used as a Calculated Column or as part of an iterator function that cycles through all rows in a table (FILTER, SUMX, MAXX, etc.)

NOTE :- Instead of using RELATED to create extra Columns (which increase file size.), nest is within measure like FILTER or SUMX.

The CACULATE Function

- CACULATE :- Evaluates an expression in a context that is modified by filters.

= CALCULATE (Expression, [Filter1], [Filter2], ...)

Name of an exisiting measure or a DAX formula for a vaild measure.

eg :- • [Total orders]
• SUM('Returns Data'[Return Quantity])

A Boolean (True/False) expression or a Table expression that defines a filter.

NOTE :- these requires fixed values or aggregation functions that return a Scalar value. (You Cannot create filter based on measure).

eg :- • 'Territory Lookup'[country] = "USA"
• Calendar[Year] <> MAX(calender[Year])

NOTE :- Think of CALCULATE as a filter modifier ; it allows you to overrule existing report filters and "force" new filter context.

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

DAX MEASURES TOTALS

Measures total may seem incorrect or inconsistent depending on how they are calculated, because they don't simply add up the visible values in the report.

The ALL Function

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

= ALL (Table or column), [Column 2], [Column 3], ...)

The table or column that you want to clear filters on

eg:- • Transactions
• Products [category]

Additional columns that you want to clear filters on (optional)

- Cannot specify columns if your first parameter is a table
- All columns must include the table name and come from the same table.

eg:- • 'Customer Lookup' [city], 'Customer Lookup' [country]
• Products [Product Name]

The FILTER function

- FILTER :- Returns a table that represent a subset of another table or expression.

= FILTER (Table, FilterExpression)

Table to be filtered

eg:- • Territory Lookup
• Customer Lookup

A Boolean (True/False) filter expression to be evaluated for each row of the table.

eg :- 'Territory Lookup'[Country] = "USA"
• Calender[Year] = 1998
• Products[Price] > [overall Avg price]

Note :-

- Since FILTER iterates through each row in a table, it can be slow and computationally expensive; only use FILTER if a simple CALCULATE function won't get the job done!

Iterator function

- Iterator (or "x") function allow you to loop through the same expression on each row of a table, then apply some sort of aggregation to the result (SUM, MAX, etc

= SUMX(Table, Expression)

Aggregation to apply to calculated rows*

eg:-
- SUMX
- COUNTX
- AVERAGEX
- RANKX
- MAXX / MINX

Table in which the expression will be evaluated.

eg:-
- Sales
- FILTERS(Sales, RELATED(Products[Category]) = "Clothing")

Expression to be evaluated for each row of the given table

eg:-
- [Total orders]
- Sales[RetailPrice] * Sales[Quantity]