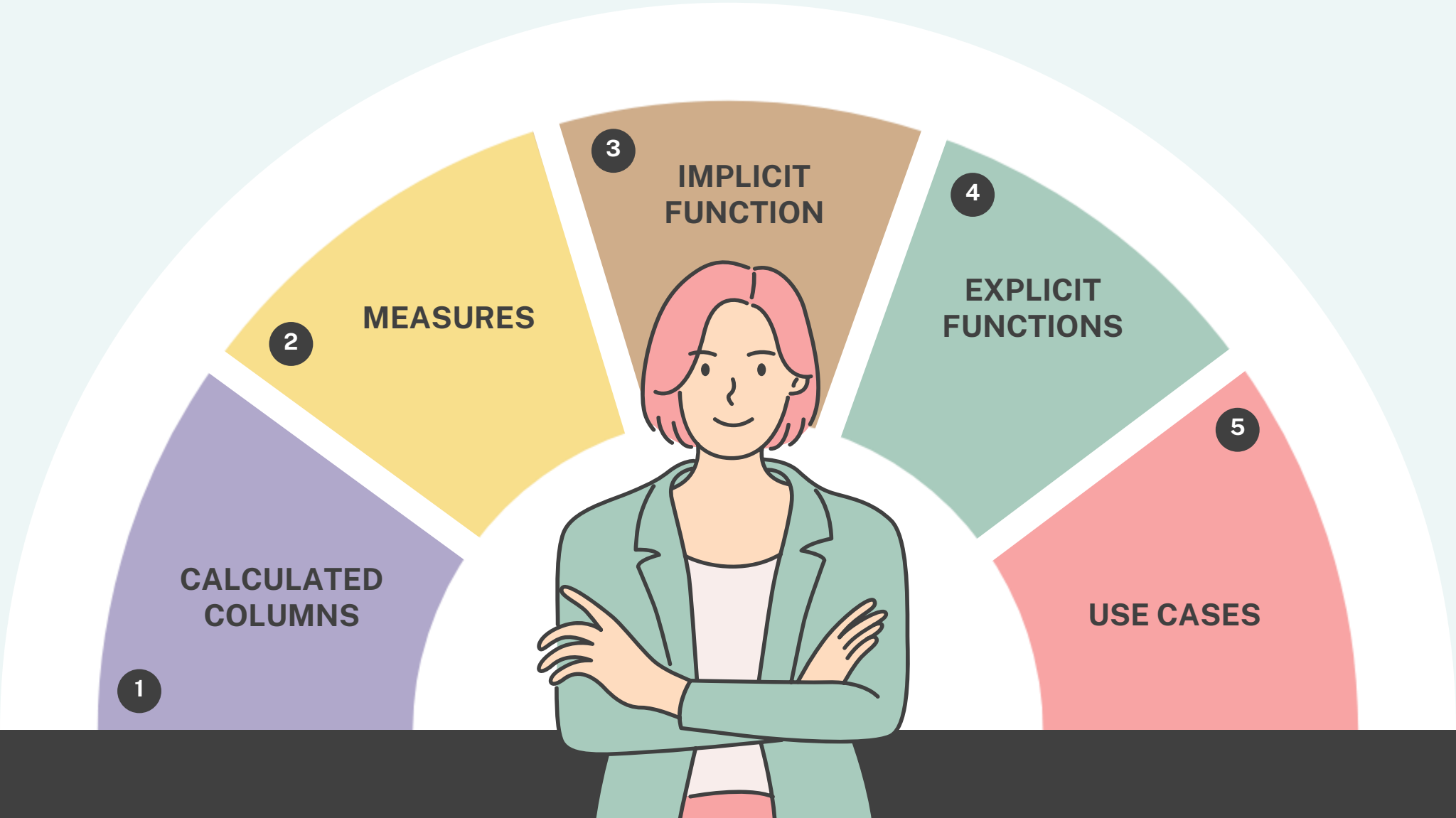
BASICS OF DAX FUNCTIONS



CALCULATED COLUMN(stored value)

Basic Theory

- A new column that is created at the row level and stored in the dataset.

From Real-Life Example:

- Think of a Calculated Column as a Pre-Written Note in Your Notebook 📖
- It's there and doesn't change dynamically.

Key Points

- **Row-Level Calculation** – Evaluates per row, storing results in a new column.
- **Stored in Data Model** – Increases file size but speeds up retrieval.
- **Used for Filtering & Relationships** – Helps in slicers, filters, and table relationships.
- **Performance Impact** – Can slow down large datasets
- **Not Affected by Slicers/Filters** – Values remain static unless data refreshes.
- **Power Query Alternative** – Use Power Query for better performance if possible.

Use Case

- When we need a **new data field** for filtering or sorting relationships.

- Example: Creating **full names** from First Name & Last Name, Extracting Year from Date, Calculating Profit Margin

Sales Table (Original TABLE)

Order Id	Product	Price	Quantity
101	Shoes	1200	1
102	Shirts	1000	2
103	Pants	1000	1
104	Dress	700	1
105	Bags	500	1
106	T-shirts	1200	3

Calculated Column Formula (DAX) : Total Sales = 'Sales'[Price] * 'Sales'[Quantity]

Order Id	Product	Price	Quantity	Total Sales
101	Shoes	1200	1	1200
102	Shirts	1000	2	2000
103	Pants	1000	1	1000
104	Dress	700	1	700
105	Bags	500	1	500
106	T-shirts	1200	3	3600

Table After Adding the DAX-F

Sales – Table Name
Price and Quantity – Column Name

Existing Columns Already there in the table

New Column Added

MEACURES(Dynamic Calculations)

Basic Theory

- A formula that calculates values dynamically based on user interactions (filters, slicers, etc.).

Key Points

- **Dynamic Calculation** – Computed at query time based on filters.
- **Efficient Performance** – Uses less storage, faster than calculated columns.
- **Context-Sensitive** – Changes based on slicers, filters, and visuals.
- **Best for Aggregations** – Ideal for totals, averages, and percentages.
- **Not Stored in Tables** – Exists only in memory when queried.
- **Used in Visuals** – Cannot be used in relationships or slicers directly.
- **Requires Aggregation Functions** – SUM, AVERAGE, COUNT, etc.

Use Case

- When we need **aggregated calculations** like total sales, average, or percentage.
- When we need **dynamic calculations** that change with filters (e.g., sales by region).
- Example: Total Revenue, Average Sales, % Growth, Running Total.

Sales Table

Products	Total Sales
Shoes	1200
Shirts	2000
Pants	1000
Dress	700
Bags	500
T-shirts	3600
Grand Total	**9000**

Only this will be shown as result for Measures

KEY DIFFERENCES

Feature	Calculated Column	Measure
Stored in Table?	Yes (takes up space)	No (calculated dynamically)
Row-Level or Aggregate?	Row-Level	Aggregate-Level
Used for Filtering?	Yes	No
Affects Performance?	More memory usage	Faster, uses less memory
Updates when data refreshes?	Yes	Yes
Example Calculation	Sales[Price]* Sales[Quantity]	SUMX(Sales, Sales[Price] * Sales[Quantity])

Implicit Measures(Auto-generated by Power BI)

Basic Theory

- These are automatically created when we drag a numeric column into a visual.

From Real-Life Example: Think of a Measure as a Calculator 

- It gives different answers depending on the input (filters, slicers, etc.)

Key Points

- **Auto-Generated by Power BI** – Created when dragging a field into a visual. Created **without** writing any DAX.
- **Not Reusable** – Cannot be referenced in other measures or calculations.
- **Limited Customization** – Only supports basic aggregations (SUM, AVERAGE, COUNT, etc.)
- **Context-Dependent** – Changes based on filters and visuals.

Use Case

- If we **drag Sales[Price] or Sales [Revenue] into a table**, Power BI will automatically **SUM it into Sum(Price) or Sum(Revenue)**
- **Uses case -** default aggregation (SUM, AVERAGE, COUNT, etc.).
- This is an **implicit measure** (SUM of Price).

Limitations:

- We **cannot modify** or use it in **complex calculations**.
- **Less Efficient** – May impact performance in complex models.

Explicit Measures(Manual Dax)

Basic Theory

- An **Explicit Measure** in DAX is a manually created measure using **DAX formulas** for precise control over calculations.

Key Points

- Defined **manually** using DAX with functions like SUM(), AVERAGE(), etc.
- **Ensures** clarity, flexibility, and optimization.
- Provides **better control** over aggregations.
- **Stored in the Model** – **Reusable** in multiple calculations, visuals and reports.
- **Context-Aware** – Adapts to slicers, filters, and visuals dynamically.

Use Case

- **Example: Explicit DAX**

Total Sales = SUMX(Sales, Sales[Price] * Sales[Quantity])

Advantages:

- Can customize calculations (e.g., filters, conditions).
- Can be used in other measures or KPIs.
- Better performance and clarity.

KEY DIFFERENCES

Feature	Implicit Measure	Explicit Measure
Created By	Power BI (Auto)	User (DAX)
Customization	No	Yes
Reusability	No	Yes
Complexity Support	Basic Aggregations	Advanced DAX Logic
Example	SUM(Price) (Auto)	SUMX(Sales, Sales[Price] * Sales[Quantity])

Why Prefer Explicit Measures?

* More **control** over calculations.

* Can be **reused** in other DAX formulas.

* Better for **performance** in large datasets.

Conclusion :

Implicit and explicit measures both belong to "Measures" in Power BI, but explicit measures are preferred for advanced analytics.

But then what is the difference between CALCULATED COLUMNS and EXPLICIT MEASURES?.....

Have you ever had this thought?
I was so confused when I first started using DAX.

Too much to take in for now, isn't it? Don't worry! I've got you covered in the upcoming topic, where we'll have exclusive clarifications on CALCULATED COLUMNS and EXPLICIT MEASURES.

Thank You For You Time!