Lesson 8

Topic: Introduction to DAX Basics & Calculated Columns vs. Measures
Prerequisites: Download DAX_Practice_Data.xlsx file

1. **What does DAX stand for?**

   DAX stands for Data Analysis Expressions.

   It's a formula language used in Microsoft tools like Power BI, Excel Power Pivot, and Analysis Services to perform calculations and data analysis on data models.

   Quick Overview:

- DAX is similar to Excel formulas but much more powerful for working with relational data.
- It allows you to create:
  - Calculated Columns
  - Measures
  - Calculated Tables

2. **Write a DAX formula to sum the Sales column.**

   To sum the Sales column in DAX, you can write the following formula as a Measure:

   DAX

   ```
   Total Sales = SUM(Sales[Sales])
   ```

   Explanation:

- Total Sales is the name of the new measure.
- SUM(Sales[Sales]) adds up all the values in the Sales column of the Sales table.

3. **What is the difference between a calculated column and a measure?**

Difference Between Calculated Column and Measure (DAX)

Calculated Column

- Added as a new column in a table.
- Uses row context (calculates row by row).
- Stored in the data model → uses more memory.
- Good for filtering, grouping, or creating new fields (e.g., Profit = Revenue - Cost).
- Example:
  Profit = Sales[Revenue] - Sales[Cost]

Measure

- A formula that calculates a value dynamically (not stored).
- Uses filter context (changes based on visuals/slicers).
- Not stored in model → better performance.
- Good for summarizing data (sum, average, etc.).

4. **Use the DIVIDE function to calculate Profit Margin (Profit/Sales).**

DAX Measure for Profit Margin:

DAX

```
Profit Margin = DIVIDE(Sales[Profit], Sales[Sales])
```

Explanation:

- Sales[Profit]: Your calculated column or field for profit (e.g., Revenue - Cost).
- Sales[Sales]: Total sales amount.
- DIVIDE(x, y) is safer than x / y — it handles division by zero and avoids errors.

5. **What does COUNTROWS() do in DAX?**

COUNTROWS() in DAX

The COUNTROWS() function counts the number of rows in a table.

Syntax:

DAX

```
COUNTROWS(table)
```

Example as a Measure:

DAX

```
Transaction Count = COUNTROWS(Data_sales)
```

Use Cases:

- Count how many sales transactions happened.
- Count number of products sold.
- Count how many rows remain after filters are applied in visuals.

With a Filter Example:

DAX

```
High Sales Count = COUNTROWS(
   FILTER(Sales, Sales[Amount] > 1000)
)
```

This counts only sales rows where the amount is greater than 1000.

6. **Create a measure: Total Profit that subtracts total cost from total sales**

DAX Measure: Total Profit

DAX

Total Profit = SUM(Data_sales[Sales]) - SUM(Data_sales[Cost])

Explanation:

- SUM(DAX_Practice_Data[Sales]): Adds all sales values.
- SUM(DAX_Practice_Data[Cost]): Adds all cost values.
- The result is the total profit across all rows.

🪄 How to Create It:

1. Go to Modeling tab in Power BI.
2. Click New measure.
3. Paste the DAX formula above.
4. Press Enter.

7. Write a measure to calculate Average Sales per Product.

DAX Measure: Average Sales per Product

If each row in your Data_sales table represents one product, you can calculate Average Sales per Product like this:

DAX

Avg Sales per Product = AVERAGE(Data_sales[Sales])

Alternative (if you want total sales ÷ number of products):

If you prefer to manually divide total sales by the number of products, use:

DAX

```
Avg Sales per Product =  DIVIDE(
   SUM(Data_sales[Sales]), COUNTROWS(Data_sales))
```

Both methods are valid — use the second one if you're doing custom logic or want more control.

8. **Use IF() to tag products as "High Profit" if Profit > 1000.**

   DAX Calculated Column: Tag Products as "High Profit"

   To tag products as "High Profit" if Profit > 1000, use this formula in a calculated column:

   DAX

   Profit Category =

   IF(Data_sales[Profit] > 1000, "High Profit", "Low Profit")

   📌 Steps to Add in Power BI:

   1. Go to the Data view.
   2. Click on New column.
   3. Paste the DAX formula above.
   4. Press Enter.

   This will create a new column (Profit Category) labeling each row as "High Profit" or "Low Profit" based on the value in the Profit column.

9. **What is a circular dependency error in a calculated column?**

   A circular dependency error happens when a calculated column refers directly or indirectly to itself, creating a loop that DAX can't resolve.

   ❌ Example (Problem):

   DAX

   ColumnA = Data_sales[ColumnB] + 100

   ColumnB = Data_sales[ColumnA] + 50

In this case:

- ColumnA depends on ColumnB
- ColumnB depends on ColumnA
  This creates a circular dependency, and Power BI will throw an error.

Error Message:

"A circular dependency was detected: Data_sales[ColumnA], Data_sales[ColumnB]..."

How to Fix It:

- Avoid mutual references between calculated columns.
- Try to restructure logic using measures instead of columns.
- Use intermediate steps that don’t depend on each other.

Summary (Note format):

- Circular dependency = formula loops back to itself.
- Caused by columns referring to each other.
- DAX can't calculate because it never reaches a final value.

10.Explain row context vs. filter context.

Row Context vs. Filter Context in DAX

Understanding these two concepts is essential for mastering DAX calculations:

Row Context

🔹 Think of it as: "One row at a time."

🔹 It applies when you create a calculated column or use iterating functions like SUMX, FILTER, etc.

Example:

DAX

Profit = Data_sales[Sales] - Data_sales[Cost]

This formula runs for each row, using the row's own Sales and Cost.

Filter Context

🔹 Think of it as: "Which data is visible right now?"

🔹 It comes from visuals (like slicers, charts) or from DAX functions like CALCULATE().

Example:

DAX

Total Sales = SUM(Data_sales[Sales])

In a report:

- If you apply a filter (e.g., ProductID = 1), DAX recalculates based on only that row's data.
- The visible data is the filter context.

11. Write a measure to calculate YTD Sales using TOTALYTD().

DAX Measure: Year-To-Date (YTD) Sales

To calculate YTD Sales using the TOTALYTD() function, use the following DAX measure:

DAX

```
YTD Sales =
TOTALYTD(
    SUM(Data_sales[Sales]),
    Data_sales[Date]
)
```

Explanation:

- SUM(Data_sales[Sales]): Total sales value.

- Data_sales[Date]: The date column used to track time progression.
- TOTALYTD() accumulates sales from the start of the year up to the current date.

12. Create a dynamic measure that switches between Sales, Profit, and Margin.

DAX: Create a Dynamic Measure to Switch Between Sales, Profit, and Margin

To do this, you'll need two steps:

◆ Step 1: Create a Slicer Table (Disconnected Table)

In Power BI, go to Modeling → New Table and enter:

DAX

CopyEdit

```
Metric Selector =
DATATABLE(
   "Metric", STRING,
   {
      {"Sales"},
      {"Profit"},
      {"Margin"}
   }
)
```

This creates a slicer list with the 3 metric options.

◆ Step 2: Create a Dynamic Measure

DAX

CopyEdit

```
Selected Metric =
VAR SelectedMetric = SELECTEDVALUE('Metric Selector'[Metric])
RETURN
SWITCH(
   SelectedMetric,
   "Sales", SUM(Data_sales[Sales]),
   "Profit", SUM(Data_sales[Sales]) - SUM(Data_sales[Cost]),
   "Margin", DIVIDE(SUM(Data_sales[Sales]) - SUM(Data_sales[Cost]), SUM(Data_sales[Sales])),
   BLANK()
)
```

How It Works:

- SELECTEDVALUE reads the user's slicer choice.
- SWITCH changes the calculation based on the selected metric.

To Use It:

1. Add a slicer to your report from the Metric Selector table.
2. Add a card or chart using the Selected Metric measure.
3. Now users can switch between Sales, Profit, and Margin dynamically!

13.Optimize a slow DAX measure using variables (VAR).

Optimize a Slow DAX Measure Using VAR

Slow DAX (Without VAR):

DAX

```
Profit Margin =
DIVIDE(
    SUM(Data_sales[Sales]) - SUM(Data_sales[Cost]),
    SUM(Data_sales[Sales])
)
```

In this example:

- SUM(Data_sales[Sales]) is calculated twice, which is inefficient.

Optimized DAX Using VAR:

```
DAX
Optimized Profit Margin =
VAR TotalSales = SUM(Data_sales[Sales])
VAR TotalCost = SUM(Data_sales[Cost])
VAR Profit = TotalSales - TotalCost
RETURN
DIVIDE(Profit, TotalSales)
```

Why This Is Better:

- Each calculation is done once and reused.
- Improves performance and readability.
- Easier to debug and maintain in large models.

Use VAR when:

- You reuse the same calculation.
- You want clean and readable logic.
- You're working with complex measures or large datasets.

14.Use CALCULATE() to override a filter

Use CALCULATE() to Override a Filter in DAX

The CALCULATE() function in DAX modifies or overrides filter context — it's one of the most powerful and flexible functions in Power BI.

🔹 Example Goal:

Show Total Sales for all products, even when a product filter is applied (e.g., from a slicer or visual).

Without CALCULATE() (Respects Filter):

DAX

```
Total Sales = SUM(Data_sales[Sales])
```

This will only sum visible rows (affected by filters/slicers).

With CALCULATE() to Override Product Filter:

DAX

```
All Product Sales =
CALCULATE(
    SUM(Data_sales[Sales]),
    REMOVEFILTERS(Data_sales[ProductID])
)
```

Explanation:

- SUM(Data_sales[Sales]): Base calculation.
- REMOVEFILTERS(Data_sales[ProductID]): Ignores any filter on ProductID, so all products are included in the total.

---

lternate Filter Override Example:

Filter to only ProductID = 1 (hardcoded):

```
DAX

Sales for Product 1 =

CALCULATE(

   SUM(Data_sales[Sales]),

   Data_sales[ProductID] = 1

)
```

15.Write a measure that returns the highest sales amount

Write a DAX Measure to Return the Highest Sales Amount

You can use the MAX() function to get the highest single sales value from the Sales column.

DAX Measure:

```
DAX

Highest Sales = MAX(Data_sales[Sales])
```

What It Does:

- Scans the Sales column in the Data_sales table.
- Returns the largest sales amount found across all rows.

Example Output:

If your sales values are: 6000, 3000, and 2000,
Highest Sales will return: 6000.