Lesson-10

1.What does FILTER(Sales, Sales[Amount] > 1000) return?

Returns:

A table that includes only the rows from the Sales table where the value in the Amount column is greater than 1000.

Key Points:

This function does not return values or a column, but a filtered table.

It's often used inside other functions (like CALCULATE, SUMX, etc.) that expect a table as an input.

Example Use:

Total Sales =CALCULATE(SUM(Sales[Amount]), FILTER(Sales, Sales[Amount] > 1000))

This example sums only the Sales[Amount] values greater than 1000.

2.Write a measure High Sales that sums Amount where Amount > 1000 using FILTER.

High Sales = CALCULATE(SUM(Sales[Amount]), FILTER(Sales, Sales[Amount] > 1000))

3.How does ALLEXCEPT(Sales, Sales[Region]) differ from ALL(Sales)?

ALL(Sales)

Removes all filters from the Sales table.

Ignores all columns and context — acts as if there's no filtering at all.

📌 Use case: When you want to calculate a total ignoring all slicers or filters.

ALLEXCEPT(Sales, Sales[Region])

Removes all filters from Sales, except for the one(s) on Sales[Region].

Keeps the filter only on the Region column.

All other filters (e.g., on Product, Date, etc.) are cleared.

📌 Use case: When you want to keep filtering by region but ignore everything else.

🔍 Example Scenario:

Suppose you have a slicer on:

Region

Product

Date

Function	What It Does
ALL(Sales)	Removes all 3 filters: Region, Product, and Date
ALLEXCEPT(Sales, Sales[Region])	Removes only Product and Date filters, keeps Region

🧠 Summary Table

Function	Keeps Filters On...	Removes Filters On...
ALL(Sales)	None	Region, Product, Date, etc.
ALLEXCEPT(Sales, Sales[Region])	Region only	Product, Date, and other columns

4.Use SWITCH to categorize Amount:

"Medium" if 500–1000

"High" if > 1000

Here's a DAX expression using SWITCH to categorize the Amount column into "Medium" and "High":

```
Amount Category =
SWITCH(
   TRUE(),
   Sales[Amount] > 1000, "High",
   Sales[Amount] >= 500 && Sales[Amount] <= 1000, "Medium",
   "Low"  -- Optional: fallback if Amount < 500
)
```

🔍 How it works:

SWITCH(TRUE(), ...) lets you use multiple conditions.

It checks each condition in order:

If Amount > 1000 → "High"

Else if between 500–1000 → "Medium"

Else → "Low" (you can remove this if not needed)

5.What is the purpose of ALLSELECTED?

Purpose of ALLSELECTED in DAX:

ALLSELECTED is used to remove filters from a column or table, but keep the filters applied by the user (e.g., slicers, filters in visuals) within the current visual or report context.

🔍 In Simple Terms:

ALLSELECTED gives you "the user's selection in the report" — even if some filters are removed inside visuals — unlike ALL, which removes all filters completely.

🧠 Why use it?

It’s especially useful when you want to:

Calculate totals or percentages that respect user selections.

Show % of selected total instead of % of grand total.

🔧 Example:

Suppose the user selects:

Region = "North" and "South"

Product = "A" and "B"

ALL(Sales[Region])

→ Ignores all filters → returns all regions (North, South, East, West)

ALLSELECTED(Sales[Region])

→ Respects the user's selection → returns only North and South

📊 Common Use Case – % of Selected Total

```
% of Selected Total =
DIVIDE(
    SUM(Sales[Amount]),
    CALCULATE(SUM(Sales[Amount]), ALLSELECTED(Sales))
)
```

This will show each row as a percentage of the filtered total, not the entire dataset.

6.Write a measure Regional Sales % showing each sale’s contribution to its region’s total (use ALLEXCEPT).

To show each sale’s contribution to its region’s total sales using a measure, you can use ALLEXCEPT to keep the filter context on the region but remove others.

DAX Measure: Regional Sales %

```
Regional Sales % =
DIVIDE(
    SUM(Sales[Amount]),
```

```
    CALCULATE(
        SUM(Sales[Amount]),
        ALLEXCEPT(Sales, Sales[Region])
    )
)
```

How It Works:

SUM(Sales[Amount]) → the amount of the current row or context.

CALCULATE(..., ALLEXCEPT(Sales, Sales[Region])) → gives the total amount for that region, ignoring filters on other columns like Product, Date, etc.

DIVIDE(...) safely calculates the percentage (avoids divide-by-zero errors).

7.Create a dynamic measure using SWITCH to toggle between SUM, AVERAGE, and COUNT of Amount.

Step 1: Create a Table for Selection

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

```
Measure Selector =
DATATABLE(
    "Measure Type", STRING,
    {
        {"Sum"},
        {"Average"},
        {"Count"}
    }
)
```

📌 This creates a static table with the options: Sum, Average, and Count.

☑ Step 2: Create a Slicer

Add a slicer to your report using the Measure Type column from the Measure Selector table.

☑ Step 3: Create the Dynamic Measure

Now create the dynamic measure:

```
Dynamic Amount Measure =
SWITCH(
```

```
    SELECTEDVALUE('Measure Selector'[Measure Type]),
    "Sum", SUM(Sales[Amount]),
    "Average", AVERAGE(Sales[Amount]),
    "Count", COUNT(Sales[Amount]),
    BLANK()
)
```

☑ Explanation

SELECTEDVALUE gets the slicer choice.

SWITCH dynamically applies the right aggregation.

BLANK() is a fallback if nothing is selected.

🧪 Example Use in a Visual

When the user selects:

Sum → Total of Amount shown.

Average → Mean of Amount values.

Count → Number of sales records.

8.Use FILTER inside CALCULATE to exclude "Furniture" sales (Products[Category] = "Furniture").

T To exclude "Furniture" sales using FILTER inside CALCULATE, you can write a measure like this:

☑ Exclude Furniture Sales:

```
Non-Furniture Sales =
CALCULATE(
    SUM(Sales[Amount]),
    FILTER(
        Products,
        Products[Category] <> "Furniture"
    )
)
```

🔍 Explanation:

CALCULATE(...) changes the filter context.

FILTER(Products, Products[Category] <> "Furniture") removes any rows where the category is "Furniture".

SUM(Sales[Amount]) calculates the total sales excluding those rows.

9.Why might ALLSELECTED behave unexpectedly in a pivot table?

Because ALLSELECTED works based on user selections and visual-level filters, and in a pivot table (especially in Excel or Power BI Matrix visuals), the way filters are applied can be indirect or implicit, leading to confusion.

Here's why it might behave unexpectedly:

1. It respects slicers and outer context

ALLSELECTED preserves slicer and outer visual filters, but removes filters from inner row/column groups.

In a pivot or matrix, it might not clear all filters, so the totals may not match your expectations.

2. It's sensitive to hierarchy levels

If you're using multiple levels (e.g., Region > Product), ALLSELECTED only includes the visible selections at the current hierarchy level.

This can make totals seem wrong if you're expecting a full unfiltered total.

3. Expecting it to behave like ALL or ALLEXCEPT

If you expect ALLSELECTED to remove all filters, like ALL, you'll get unexpected results because it does not fully clear filters.

Example:

```
% of Selected Total =
DIVIDE(
   SUM(Sales[Amount]),
   CALCULATE(SUM(Sales[Amount]), ALLSELECTED(Sales))
```

)

If you're in a pivot showing Region and Product, the denominator will reflect only the selected rows visible in the current visual, not the full dataset.

10.Write a measure that calculates total sales and ignores filters from region

To calculate total sales ignoring filters on Region, use the ALL function to remove the Region filter context.

☑ DAX Measure: Ignore Region Filters

Total Sales (Ignore Region) =

CALCULATE(

  SUM(Sales[Amount]),

  ALL(Sales[Region])

)

🔍 How it works:

SUM(Sales[Amount]) → calculates total sales.

ALL(Sales[Region]) → removes any filters applied to the Region column (from slicers, visuals, etc.).

This measure will always show the total sales across all regions, even if the user filters by a specific region in the report.

11. Optimize this measure:

High Sales = CALCULATE(SUM(Sales\[Amount]), FILTER(Sales, Sales\[Amount] > 1000)) (Hint: Replace FILTER with a Boolean filter inside CALCULATE.)

Optimized Measure:

High Sales =

CALCULATE(

  SUM(Sales[Amount]),

  Sales[Amount] > 1000

)

Why this is better:

CALCULATE can accept Boolean expressions like Sales[Amount] > 1000 directly.

This is equivalent to writing FILTER(Sales, Sales[Amount] > 1000) but runs faster and is more readable.

12.Write a measure Top 2 Products using TOPN and FILTER to show the highest-grossing products.

```
Top 2 Product = SELECTCOLUMNS(
    TOPN(
        2,
        ADDCOLUMNS(
            SUMMARIZE(Sales, Sales[ProductID]),
            "ProductName", LOOKUPVALUE(
                Products[ProductName],
                Products[ProductID], Sales[ProductID]
            ),
            "TotalSales", CALCULATE(SUM(Sales[Amount]))
        ),
        [TotalSales], DESC
    ),
    "TopProductName", [ProductName]
)
```

13.Use ALLSELECTED with no parameters to respect slicers but ignore visual-level filters.

Great! Here's how to use ALLSELECTED() with no parameters in a measure to:

Respect slicers

Ignore visual-level filters (like filters applied to a single chart or matrix)

Example Measure:

% of Total Sales (Respecting Slicers)

Sales % of Slicer Total :=

DIVIDE(

SUM(Sales[Amount]),

CALCULATE(

SUM(Sales[Amount]),

ALLSELECTED()

)

)

🔍 Behavior:

SUM(Sales[Amount]): your current row's value.

ALLSELECTED(): recalculates total sales across the context defined by slicers, but ignores visual-level filters like chart filters or drilldowns.

🧪 Example:

If a slicer limits to Region = "North" and Product = "P1" and your visual is further filtered to just January

This measure includes all North–P1 sales, even outside January.

It gives you % of total based on slicer selections, not the chart filter.

14.Debug: A SWITCH measure returns incorrect values when fields are added to a matrix visual.

When a SWITCH measure in DAX returns incorrect or unexpected values in a matrix visual, especially when more fields are added to rows or columns, it's usually caused by row context issues or ambiguous evaluation context.

☑ Common Root Cause:

You are likely using a SWITCH measure that depends on a selection, such as a slicer or a dynamic measure selector. But when you add fields to a matrix (like Region, Product, Date, etc.), the context changes, and the measure may:

Evaluate differently than expected

Return blank or default branch

Apply a value at an unexpected level of granularity

Example of a Fragile SWITCH Pattern:

```
Selected Measure :=
SWITCH(
    SELECTEDVALUE(MeasureSelector[Measure]),
    "Sum", SUM(Sales[Amount]),
    "Average", AVERAGE(Sales[Amount]),
    "Count", COUNT(Sales[Amount])
)
```

🚨 Problem:

When matrix has additional fields (e.g. Product, Region), SELECTEDVALUE(...) might return blank or unexpected value due to multiple values in context.

☑ Fix 1: Use a disconnected table for selection

Ensure MeasureSelector[Measure] is a disconnected table used only for slicers, not involved in relationships.

☑ Fix 2: Use ISINSCOPE or check granularity

If your logic needs to behave differently at different matrix levels, use:

```
SWITCH(
    TRUE(),
    ISINSCOPE(Product[ProductName]), SUM(Sales[Amount]),
    ISINSCOPE(Region[Region]), AVERAGE(Sales[Amount]),
    COUNT(Sales[Amount])
)
```

☑ Fix 3: Protect against ambiguous context with fallback logic

Use:

```
SELECTEDVALUE(MeasureSelector[Measure], "Default")
```

So if the selected value is ambiguous or missing, it doesn't break your logic.

☑ Fix 4: Test with a card first

Before adding the measure to a matrix, test it in a card visual to make sure the logic is returning as expected based on slicers.

🧪 Diagnostic Tip:

To debug further, add a temporary measure:

```
Current Context :=
SELECTEDVALUE(MeasureSelector[Measure])
```

Place it in your matrix to confirm what the context actually is when multiple fields are in play.

15.Simulate a "reset filters" button using ALL in a measure.

When a SWITCH measure in DAX returns incorrect or unexpected values in a matrix visual, especially when more fields are added to rows or columns, it’s usually caused by row context issues or ambiguous evaluation context.

☑ Common Root Cause:

You are likely using a SWITCH measure that depends on a selection, such as a slicer or a dynamic measure selector. But when you add fields to a matrix (like Region, Product, Date, etc.), the context changes, and the measure may:

Evaluate differently than expected

Return blank or default branch

Apply a value at an unexpected level of granularity

🧠 Example of a Fragile SWITCH Pattern:

```
Selected Measure :=
SWITCH(
    SELECTEDVALUE(MeasureSelector[Measure]),
    "Sum", SUM(Sales[Amount]),
    "Average", AVERAGE(Sales[Amount]),
    "Count", COUNT(Sales[Amount])
)
```

🚨 Problem:

When matrix has additional fields (e.g. Product, Region), SELECTEDVALUE(...) might return blank or unexpected value due to multiple values in context.

☑ Fix 1: Use a disconnected table for selection

Ensure MeasureSelector[Measure] is a disconnected table used only for slicers, not involved in relationships.

☑ Fix 2: Use ISINSCOPE or check granularity

If your logic needs to behave differently at different matrix levels, use:

```
SWITCH(
    TRUE(),
```

```
    ISINSCOPE(Product[ProductName]), SUM(Sales[Amount]),
    ISINSCOPE(Region[Region]), AVERAGE(Sales[Amount]),
    COUNT(Sales[Amount])
)
```

☑ Fix 3: Protect against ambiguous context with fallback logic

Use:

```
SELECTEDVALUE(MeasureSelector[Measure], "Default")
```

So if the selected value is ambiguous or missing, it doesn't break your logic.

☑ Fix 4: Test with a card first

Before adding the measure to a matrix, test it in a card visual to make sure the logic is returning as expected based on slicers.

🧪 Diagnostic Tip:

To debug further, add a temporary measure:

```
Current Context :=
SELECTEDVALUE(MeasureSelector[Measure])
```

Place it in your matrix to confirm what the context actually is when multiple fields are in play.