Advanced DAX – FILTER, CALCULATE, SWITCH, ALLSELECTED

---

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

Answer:
It returns a **table** with all rows from the `Sales` table where `Amount > 1000`.

🔍 It does **not** return a number — only a **filtered table**, often used inside `CALCULATE()` or iterators like `SUMX()`.

---

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)
)
```

✅ `CALCULATE` changes the filter context using the table returned by `FILTER()`.

---

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

Answer:

`ALL(Sales)`	`ALLEXCEPT(Sales, Sales[Region])`
Removes **all filters** on Sales	Removes all filters **except** on `Region`
Ignores slicers and visuals entirely	Keeps Region filter, removes others (e.g., Product)

---

4. Use `SWITCH` to categorize Amount:

- "Medium" if 500–1000
- "High" if > 1000
- "Low" otherwise

```
Amount Category =
SWITCH(
    TRUE(),
    Sales[Amount] > 1000, "High",
    Sales[Amount] >= 500, "Medium",
    "Low"
)
```

✅ `SWITCH(TRUE(), ...)` mimics `IF-ELSE IF` logic.

---

5. What is the purpose of `ALLSELECTED`?

Answer:
`ALLSELECTED()` **removes visual-level filters**, but **respects slicers** and **user selections**.
Useful for calculating percentages **relative to selected data**, even if some filters exist on visuals.

---

6. Write a measure `Regional Sales %` showing each sale's contribution to its region's total using `ALLEXCEPT`.

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

✅ Keeps the Region filter context, but removes other filters.

---

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

Assuming a disconnected table `MetricTable[Metric]` with values: `"SUM"`, `"AVERAGE"`, `"COUNT"`

```
Dynamic Amount Measure =
SWITCH(
    SELECTEDVALUE(MetricTable[Metric]),
    "SUM", SUM(Sales[Amount]),
    "AVERAGE", AVERAGE(Sales[Amount]),
    "COUNT", COUNT(Sales[Amount])
)
```

---

8. Use `FILTER` inside `CALCULATE` to exclude "Furniture" sales.

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

✅ Make sure `Sales` is related to `Products` by `ProductID`.

---

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

Answer:
Because `ALLSELECTED` can **capture row headers** in pivot visuals, it might include more filters than intended — especially when both slicers and visual filters apply.

✅ Use carefully or replace with `REMOVEFILTERS()` or `ALL()` depending on the context.

---

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

```
Sales All Regions =
CALCULATE(
    SUM(Sales[Amount]),
    REMOVEFILTERS(Sales[Region])
)
```

✅ `REMOVEFILTERS()` works like `ALL(Sales[Region])` but more readable.

---

11. Optimize this measure:

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

Optimized version:

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

✅ In **CALCULATE**, you can use a **Boolean expression** directly instead of wrapping with `FILTER()`.

---

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

```
Top 2 Products Sales =
CALCULATE(
    SUM(Sales[Amount]),
    TOPN(2,
        SUMMARIZE(Sales, Sales[Product], "TotalSales", SUM(Sales[Amount])),
        [TotalSales], DESC
    )
)
```

✅ You may also use this in a visual by using `ISINSCOPE` or a ranking measure.

---

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

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

✅ Works well for % of total comparisons in visuals with slicers.

---

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

Likely cause:
`SELECTEDVALUE()` may return **blank** if **multiple values** are in context (e.g., multiple rows in matrix).
✅ Use `IF(HASONEVALUE())` or default values:

```
SWITCH(
    TRUE(),
    SELECTEDVALUE(Table[Column], "Default") = "SUM", ...
)
```

---

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

```
Reset Filter Sales =
CALCULATE(
    SUM(Sales[Amount]),
    ALL(Sales)
)
```