DAX Functions

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

The `FILTER` function returns a **table**. Specifically, `FILTER(Sales, Sales[Amount] > 1000)` returns a table that is a filtered version of the 'Sales' table, containing only the rows where the value in the `Sales[Amount]` column is greater than 1000. It doesn't return a single value, but rather a new, temporary table that can be used as an argument inside another function.

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

- `ALL(Sales)` removes all filters from the `Sales` table. It returns the complete, unfiltered `Sales` table, ignoring any filters that might be applied by slicers or other visuals.
-
-
- `ALLEXCEPT(Sales, Sales[Region])` removes all filters from the `Sales` table **except** for the filters that have been applied to the `Sales[Region]` column. This function is useful when you want to calculate a value based on the total for a specific category while ignoring other filters.
-

Creating Measures

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

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

This measure uses `CALCULATE` to modify the filter context of the `SUM` function. The `FILTER` function creates a temporary table containing only the rows where `Sales[Amount]` is greater than 1000, and `CALCULATE` then performs the sum on this filtered table.

Use `SWITCH` to categorize `Amount`: "Medium" if 500–1000, "High" if > 1000.

Фрагмент кода

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

The SWITCH function evaluates the conditions in order. It checks if Sales[Amount] is greater than 1000 first. If true, it returns "High." If not, it moves to the next condition, checking if Sales[Amount] is greater than 500. If that is true, it returns "Medium." If none of the conditions are met, it returns the default value, which is "Low" in this case.

What is the purpose of ALLSELECTED?

The purpose of ALLSELECTED is to return all rows in a table while preserving all the **explicit filters applied in the visual or query**. Unlike ALL which completely ignores all filters, ALLSELECTED respects the selections made by the user in slicers or other filters. For example, if a user filters a report to show only sales for the 'West' region, ALLSELECTED will operate on the total for the 'West' region, whereas ALL would operate on the total for the entire dataset.

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

Фрагмент кода

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

This measure divides the sum of Sales[Amount] for the current filter context by the sum of Sales[Amount] for the region. ALLEXCEPT(Sales, Sales[Region]) ensures that the denominator's calculation respects the region filter but removes any other filters (like by city or product type) that might be in effect.

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

This dynamic measure requires a disconnected table with a column of values for 'Sum', 'Average', and 'Count'. The measure uses SELECTEDVALUE to detect the user's selection and then performs the corresponding calculation.

Фрагмент кода

```
Dynamic Sales Measure =
SWITCH(
    TRUE(),
    SELECTEDVALUE('User Selections'[Aggregation]) = "Sum", SUM(Sales[Amount]),
    SELECTEDVALUE('User Selections'[Aggregation]) = "Average", AVERAGE(Sales[Amount]),
    SELECTEDVALUE('User Selections'[Aggregation]) = "Count", COUNT(Sales[Amount]),
    SUM(Sales[Amount]) // Default value
)
```

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

Фрагмент кода

```
Sales Excluding Furniture =
```

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

The CALCULATE function changes the filter context for the SUM calculation. The FILTER expression removes any rows from the Products table where the Category is "Furniture," and CALCULATE then applies this modified filter context to the SUM of Sales[Amount].

Why might ALLSELECTED behave unexpectedly in a pivot table?

ALLSELECTED can behave unexpectedly in a pivot table due to the complexity of filter context. Pivot tables create a new filter context for each cell, row total, and column total. A measure using ALLSELECTED may not calculate what you expect in the totals because it accounts for the filters of the entire visual, not just the subset of data within a single cell. This can lead to a situation where a percentage measure, for example, gives a correct percentage for each row but an unexpected value for the grand total because it is calculating the total based on the entire set of data selected by the report-level filters.
Great questions—here are tight, battle-tested patterns.

1) Debug: `SWITCH` returns wrong values when you add fields to a Matrix

Cause (most common): adding rows/columns changes filter granularity, so things like `HASONEVALUE()`/`SELECTEDVALUE()` flip from single → multiple (or blank). Your `SWITCH` then hits the wrong branch (or the default).

Fix pattern: detect the *level in the matrix* with `ISINSCOPE()` and order branches from most-granular to least. Avoid branching on `HASONEVALUE()` for matrices.

```
Metric :=
SWITCH (
    TRUE(),
    -- Leaf level (e.g., Product in scope)
    ISINSCOPE ( 'Dim Product'[ProductID] ), [Leaf Calc],

    -- Higher level (e.g., Region)
    ISINSCOPE ( 'Dim Region'[Region] ), [Region Calc],

    -- Totals / grand totals
    NOT ISINSCOPE ( 'Dim Product'[ProductID] ) && NOT ISINSCOPE
( 'Dim Region'[Region] ), [Total Calc],

    BLANK()
)
```

Tips

- If you were using `SELECTEDVALUE( 'Metric'[Name] )`, make it robust:

```
VAR _metric = COALESCE ( SELECTEDVALUE('Metric'[Name]),
"Default" )
```

- In `SWITCH(TRUE(), ...)`, ensure each condition is mutually exclusive, and put the *most specific* tests first.

---

2) Simulate a "Reset filters" button with `ALL` inside a measure

Make a tiny disconnected table to act as a toggle.

Step A — helper table (Enter Data / DAX):

```
Reset Mode = DATATABLE( "Mode", STRING, { { "Use filters" },
{ "Reset" } } )
```

Put `Reset Mode[Mode]` on a single-select slicer (or a button that sets this value via bookmark selection).

Step B — use it in your measure:

```
Total Amount (Resettable) :=
VAR _mode = SELECTEDVALUE ( 'Reset Mode'[Mode], "Use filters" )
RETURN
IF (
    _mode = "Reset",
    -- ignore all filters coming from model tables
    CALCULATE (
        SUM ( Sales[Amount] ),
        ALL ( Sales ),
        ALL ( 'Dim Product' ),
        ALL ( 'Dim Region' ),
        ALL ( 'Date' )
    ),
    -- normal, respects current filters
    SUM ( Sales[Amount] )
)
```

Notes

- ALL() removes filters on the specified tables/columns (this is your “reset”).
- If you want the “reset” to **still respect slicers** but ignore only visual-level filters, swap the ALL(...) block for ALLSELECTED() with no parameters in that branch.
- On very wide models, you can replace multiple ALL()s with REMOVEFILTERS() on the same set of tables (or use it repeatedly per table).