1. FILTER(Sales, Sales[Amount] > 1000) in DAX returns:
   - A **table** (not a single value).
   - That table contains **all the rows from Sales** where the condition Sales[Amount] > 1000 is true.
   - It **does not aggregate** or calculate anything — it just reduces the rows.

Example:

Suppose Sales table is:

SaleID	Amount
1	500
2	1200
3	2000

Then:
FILTER(Sales, Sales[Amount] > 1000) →

SaleID	Amount
2	1200
3	2000

⚠️ Important: Since FILTER() returns a **table**, you cannot display it directly in a measure. You normally pass it into another function like CALCULATE, SUMX, COUNTROWS, etc.

👉 Example usage:

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

This would return the **total of sales greater than 1000**.

3. **1. ALL(Sales)**

- Removes **all filters** from the Sales table.
- It's like looking at the entire Sales table without any slicers or filters applied.

👉 Example:
If you have filters on **Region** and **Category**, ALL(Sales) ignores both of them.

---

2. ALLEXCEPT(Sales, Sales[Region])

- Removes **all filters except the ones you specify**.
- In this case, it **keeps the Region filter**, but ignores all other filters (like Category, Product, Date, etc.).

👉 Example:
If you're filtering on **Region = "USA"** and **Category = "Electronics"**:

- ALL(Sales) → sees the full Sales table (no filters).
- ALLEXCEPT(Sales, Sales[Region]) → keeps only Region = "USA", but removes the Category filter.

🔑 **Key difference:**

- **ALL(Sales):** "Forget ALL filters."
- **ALLEXCEPT(Sales, Sales[Region]):** "Forget everything EXCEPT Region."

👉 In practice, ALLEXCEPT is often used when you want to calculate things like *% of total by Region*.

Example measure:

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

This gives each row's share of its **Region total**, instead of the **Grand total**.

5. In **DAX**, the function **ALLSELECTED()** is used to **remove filters but still respect user selections (from slicers, filters, etc.)**.

🔑 Purpose of ALLSELECTED

- It **ignores filters applied inside the visual itself** (like rows/columns in a table or matrix).
- But it **keeps slicer and report/page-level filters**.
- Most often, it's used to calculate **percentages (%) of totals** within the user's current selection.

📊 Example 1: % of Selected Total Sales

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

👉 Here:

- SUM(Sales[Amount]) → current row's sales
- ALLSELECTED(Sales) → sales total for **only what the user selected** in slicers
- Result → each row shows % of its selection total

📊 Example 2: Difference Between ALL vs ALLSELECTED

- **ALL(Sales)** → removes *all* filters (ignores slicers too).
- **ALLSELECTED(Sales)** → removes only *visual filters*, but still respects slicers.

For example:

- If a slicer selects **Year = 2023**,
  - ALL(Sales) → calculates across **all years**
  - ALLSELECTED(Sales) → calculates only within **2023**

✅ **In short:**
ALLSELECTED is very useful when you want calculations that are relative to **what the user has chosen**, not the entire dataset.

9. 🔹 **What ALLSELECTED does**

- ALLSELECTED clears filters **but only within the current user selection (slicer, filter, or pivot table selection)**.
- This means it "remembers" what the user selected, unlike ALL which clears everything.

🔹 **Why it can behave unexpectedly in a pivot table**

1. **Depends on visible selection**
   If the pivot table only shows a subset of categories, ALLSELECTED uses only those visible categories as the denominator. This can give results that look inconsistent if you expected the total across all categories.
2. **Nested levels in pivot table**
   In a pivot with hierarchies (e.g., Region → Country), ALLSELECTED can return different totals depending on which level is expanded. At Region level, it includes all selected Regions; at Country level, only those Countries under visible Regions.
3. **Interaction with slicers**
   If a slicer is applied (e.g., Year = 2024), ALLSELECTED respects that. So results may change unexpectedly when slicer values are changed or multiple slicers interact.
4. **Misunderstood denominator**
   Often ALLSELECTED is used in % of Total measures. If the pivot table shows only part of the selection, the percentage may not be against the *true* grand total but only the currently visible selection.

✅ **In short:**
ALLSELECTED is context-sensitive. In a pivot table, the "total" it returns is based on whatever rows/columns the user has chosen or expanded. That's why it sometimes looks "unexpected" if you assume it behaves like ALL.

13. Power BI'da **ALLSELECTED** ikki xil usulda ishlatiladi:

- **ALLSELECTED(Table[Column])** → faqat shu column bo'yicha slicer/selectionlarni hurmat qiladi.
- **ALLSELECTED() (parametrsiz)** → butun model bo'yicha slicerlarni hurmat qiladi, **lekin visual-level filterlarni e'tiborga olmaydi**.

🔹 **Misol**

```
Sales Respecting Slicers =
CALCULATE (
    SUM ( Sales[Amount] ),
    ALLSELECTED()
)
```

🔹 **Qanday ishlaydi?**

1. Agar siz slicer qo‘ysangiz (masalan, faqat **2024 yilni tanlasangiz**) → bu measure **faqat 2024 yil savdosini** ko‘rsatadi. ✅
2. Lekin agar siz bar chart ustida filtrlasangiz (masalan, faqat **Electronics** kategoriyani tanlab qolsangiz) → bu measure **hamma kategoriyani** hisoblaydi, chunki u **visual-level filterlarni e’tiborga olmaydi**. 🚫

🔹 **Foydali joyi**

- % of Total Sales kabi hisoblarda ishlatiladi → slicer tanlovi saqlanadi, lekin visualdagi boshqa filterlar e’tiborga olinmaydi.

14. This usually isn’t a SWITCH bug—it’s **context**. When you add fields to a matrix, the filter context changes; things like SELECTEDVALUE() may start returning **Blank** (because there are multiple values), grand totals lose row context, and branches inside SWITCH can flip unexpectedly.

Here’s a tight checklist + a robust pattern:

Why it goes wrong

1. **SELECTEDVALUE() = Blank (or different)**

- Adding rows/columns means more than one value is in scope → SELECTEDVALUE() returns Blank → SWITCH hits the wrong/else branch.
- Fix: give SELECTEDVALUE() a **default** and source it from a **disconnected selector** table (not from fields used in the matrix).

2. **Totals vs detail (row context disappears at totals)**

- At totals, AVERAGE, COUNT, etc. can mean something else than per-row logic.
- Fix: branch with ISINSCOPE() to handle totals differently.

3. **Mixed return types**

- Every SWITCH branch must return the **same type** (all numbers). If one returns text/blank mishandled, results look wrong.

4. **Overriding filters in branches**

- If some branches use ALL/ALLEXCEPT/ALLSELECTED, they can ignore slicers/visual filters; others don’t → inconsistent results.

Robust SWITCH pattern

Use a disconnected selector and guard for totals:

```
Dynamic Measure :=
VAR Choice =
   SELECTEDVALUE( 'Measure Selector'[OptionName], "Sum" )   -- default!
VAR AtProductLevel =
   ISINSCOPE( Products[ProductName] )                   -- adjust to your hierarchy
RETURN
SWITCH (
   TRUE(),
   Choice = "Sum",
      [Total Sales],                              -- e.g. SUM(Sales[Amount])

   Choice = "Average",
      IF (
         AtProductLevel,
         AVERAGE ( Sales[Amount] ),                    -- per-row level
         AVERAGEX ( VALUES ( Products[ProductName] ), [Total Sales] )  -- total logic
      ),

   Choice = "Count",
      COUNTROWS ( VALUES ( Sales[SaleID] ) ),           -- stable counting pattern
```

```
    BLANK()
)
```

Tips

- **Disconnected table** for the selector (Home → Enter Data): “Sum”, “Average”, “Count”. Use it in a slicer.
- Always give SELECTEDVALUE(..., "fallback").
- Use ISINSCOPE() to make totals/grand totals do what you expect.
- Keep all branches numeric and avoid ALL* unless you truly want to ignore filters.