1. What is row context? Give an example in a calculated column.

Row context is the concept where DAX evaluates expressions **row by row** (like in Excel).

Example (calculated column in `Sales`):

```
TotalPrice = Sales[Quantity] * Sales[UnitPrice]
```

For each row in the `Sales` table, DAX multiplies the row's `Quantity` and `UnitPrice`.

---

2. Write a measure that finds total sales

Assuming `TotalSales` is Quantity × UnitPrice:

```
Total Sales = SUMX(Sales, Sales[Quantity] * Sales[UnitPrice])
```

Or if there's a `TotalPrice` column:

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

---

3. Use RELATED to fetch the Name from the Customers table into the Sales table

Calculated column in Sales:

```
CustomerName = RELATED(Customers[Name])
```

`RELATED()` works when there's a relationship from `Sales` → `Customers`.

---

4. What does this return?

```
CALCULATE(SUM(Sales[Quantity]), Sales[Category] = "Electronics")
```

This returns the **sum of Quantity only for rows** where `Sales[Category] = "Electronics"`.

It overrides existing filters with the new one.

---

5. Explain the difference between `VAR` and `RETURN` in DAX

- `VAR` stores **intermediate values**
- `RETURN` outputs the final result

Example:

```
ElectronicsQty =
VAR Qty = SUMX(FILTER(Sales, Sales[Category] = "Electronics"),
Sales[Quantity])
RETURN Qty
```

---

6. Create a calculated column in `Sales` called `TotalPrice` using row context

```
TotalPrice = Sales[Quantity] * Sales[UnitPrice]
```

This works because calculated columns automatically use **row context**.

---

7. Write a measure `Electronics Sales` using `CALCULATE`

```
Electronics Sales =
CALCULATE(
    SUM(Sales[TotalPrice]),
    Sales[Category] = "Electronics"
)
```

---

8. Use `ALL(Sales[Category])` in a measure to show total sales ignoring category filters

```
Total Sales All Categories =
CALCULATE(
    SUM(Sales[TotalPrice]),
    ALL(Sales[Category])
)
```

---

9. Fix this error: A calculated column in Sales uses `RELATED(Customers[Region])` but returns blanks

Likely cause:

- No relationship exists between `Sales` and `Customers`, or the key values don’t match.

Fix:

- Ensure there's a valid relationship on `CustomerID`
- Check for matching keys (no data type mismatch or blank `CustomerID`s)

---

10. Why does CALCULATE override existing filters?

Because `CALCULATE`:

- **modifies filter context** using additional filters you define
- replaces filters on columns you explicitly reference

That's what makes it powerful and context-sensitive.

---

11. Write a measure that returns average `UnitPrice` of products

If `Products[UnitPrice]` exists:

```
Avg Unit Price = AVERAGE(Products[UnitPrice])
```

If `Sales[UnitPrice]` exists:

```
Avg Unit Price = AVERAGE(Sales[UnitPrice])
```

---

12. Use VAR to store a temporary table of high-quantity sales (`Quantity > 2`), then count rows

```
HighQtyCount =
VAR HighSales = FILTER(Sales, Sales[Quantity] > 2)
RETURN COUNTROWS(HighSales)
```

---

13. Write a measure `% of Category Sales`

```
% of Category Sales =
DIVIDE(
    Sales[TotalPrice],
    CALCULATE(SUM(Sales[TotalPrice]), ALLEXCEPT(Sales, Sales[Category]))
)
```

Compares each row's sale to total sales in its category.

---

14. Simulate a "remove filters" button using `ALL` in a measure

```
Sales No Filter =
CALCULATE(
    SUM(Sales[TotalPrice]),
    ALL(Sales)
)
```

Attach this measure to a **card** to show unfiltered totals regardless of slicers.

---

15. Troubleshoot: A CALCULATE measure ignores a slicer. What's the likely cause?

Possible reasons:

- The measure uses `ALL()` or `REMOVEFILTERS()` → removes slicer context.
- There's no relationship between slicer field and the fact table.
- Cross-filter direction is not set correctly (check model view).

Fix:

- Confirm a proper relationship
- Avoid removing filters unless needed
- Use `TREATAS` if slicer field is in a disconnected table