🚀 **Project Title: Data-Driven Sales Insights for an E-commerce Platform**

**1. Dataset Design (4 Core Tables)**

We’ll simulate a real-world e-commerce environment:

### **1. Users**

user\_id (PK), name, city, signup\_date

### **2. Products**

product\_id (PK), name, category, price

### **3. Orders**

order\_id (PK), user\_id (FK), order\_date, total\_amount, payment\_method, status

### **4. Order\_Items**

item\_id (PK), order\_id (FK), product\_id (FK), quantity, item\_price

## 📊 2. ****Case Study Sections & Questions****

We’ll divide the project into **5 major themes**, each with real-life analytics needs.

### 🔸 A. **Sales Performance Analysis**

| **🔍 Question** | **💡 What it shows** |
| --- | --- |
| 1. Daily / Monthly / Quarterly / Yearly sales | Revenue trend tracking |
| 2. YoY & MoM growth % | Business momentum |
| 3. Top-selling products by revenue | Product demand |
| 4. Revenue by category & city | Geo/segment performance |
| 5. Peak revenue months/days | Seasonality |

### 🔸 B. **Customer Behavior & Segmentation**

| **🔍 Question** | **💡 What it shows** |
| --- | --- |
| 6. Repeat vs. one-time customers % | Loyalty |
| 7. Avg order value per user | Buyer strength |
| 8. Users who increased/decreased purchase frequency | Retention signals |
| 9. Top 5 customers by spend | High-value clients |
| 10. Inactive users (no orders in 90+ days) | Churn risk |

### 🔸 C. **Product-Level Insights**

| **🔍 Question** | **💡 What it shows** |
| --- | --- |
| 11. Inventory pressure products | Out-of-stock risks |
| 12. Products sold in most cities | Market coverage |
| 13. Products with highest return rate (if applicable) | Quality issues |
| 14. Price bands contributing most revenue | Pricing strategy |

### 🔸 D. **Order Fulfillment & Cancellations**

| **🔍 Question** | **💡 What it shows** |
| --- | --- |
| 15. % cancelled orders by category | Return/cancel hotspots |
| 16. Payment method trends | Preferred modes |
| 17. Daily/monthly order volume trend | Supply planning |
| 18. Avg order items per cart | Buyer behavior |
| 19. % orders with only 1 item | Upsell opportunity |

### 🔸 E. **📐 Percentage-Based KPIs (Interview Focus)**

| **✅ Question** | **Example SQL Concept** |
| --- | --- |
| 20. % of users who made purchases after signup |  |
| 21. % orders per payment method |  |
| 22. % revenue from top 10 products |  |
| 23. % customers who bought from multiple categories |  |
| 24. % revenue growth MoM |  |
|  |  |
|  |  |

1. **Sales Performance Analysis**
2. Daily / Monthly / Quarterly / Yearly sales

SELECT

CAST(order\_date AS DATE) AS daily,

SUM(order\_amount) AS total\_sales

FROM orders

GROUP BY CAST(order\_date AS DATE)

ORDER BY daily;

🔍What This Query Does:

Part Purpose

CAST(order\_date AS DATE) Removes time component, so you group by just the date

SUM(order\_amount) Aggregates total revenue for that date

GROUP BY Groups sales per unique day

ORDER BY Ensures the results are chronological (great for charting trends)

Bonus Tip:

If your order\_date is already of DATE datatype, CAST() isn’t needed.

But in most cases (like DATETIME or TIMESTAMP), CAST() is necessary to avoid grouping by every second

SELECT

FORMAT(order\_date, 'yyyy-MM') AS month\_year,

SUM(order\_amount) AS total\_amount

FROM orders

GROUP BY FORMAT(order\_date, 'yyyy-MM')

ORDER BY total\_amount DESC;

## Explanation:

| **Clause** | **What It Does** |
| --- | --- |
| FORMAT(order\_date, 'yyyy-MM') | Extracts **year and month** in YYYY-MM format |
| SUM(order\_amount) | Total revenue for each month |
| ORDER BY total\_amount DESC | Shows highest revenue months first |

**Yearly Sales**

Select

Year(order\_date) as year,

Sum(order\_total) as total\_amount

From orders

Group by Year(order\_date)

Order by Sum(order\_total) desc

**YoY & MoM growth %**

WITH monthly\_sales AS (

SELECT

FORMAT(order\_date, 'yyyy-MM') AS month\_year,

YEAR(order\_date) AS sales\_year,

MONTH(order\_date) AS sales\_month,

SUM(order\_amount) AS total\_sales

FROM orders

GROUP BY FORMAT(order\_date, 'yyyy-MM'), YEAR(order\_date), MONTH(order\_date)

)

, sales\_with\_growth AS (

SELECT

\*,

LAG(total\_sales, 1) OVER (ORDER BY sales\_year, sales\_month) AS prev\_month\_sales,

LAG(total\_sales, 12) OVER (ORDER BY sales\_year, sales\_month) AS prev\_year\_sales

FROM monthly\_sales

)

SELECT

month\_year,

total\_sales,

prev\_month\_sales,

prev\_year\_sales,

ROUND(

CASE

WHEN prev\_month\_sales = 0 THEN NULL

ELSE (total\_sales - prev\_month\_sales) \* 100.0 / prev\_month\_sales

END, 2

) AS MoM\_growth\_percent,

ROUND(

CASE

WHEN prev\_year\_sales = 0 THEN NULL

ELSE (total\_sales - prev\_year\_sales) \* 100.0 / prev\_year\_sales

END, 2

) AS YoY\_growth\_percent

FROM sales\_with\_growth

ORDER BY month\_year;

**Top-selling products by revenue**

SELECT

oi.product\_id,

p.name AS product\_name,

SUM(oi.quantity \* oi.item\_price) AS total\_revenue

FROM order\_items oi

JOIN products p ON oi.product\_id = p.product\_id

GROUP BY oi.product\_id, p.name

ORDER BY total\_revenue DESC;

**Revenue by category & city**

SELECT

u.city,

p.category,

SUM(oi.quantity \* oi.item\_price) AS total\_revenue

FROM orders o

JOIN users u ON o.user\_id = u.user\_id

JOIN order\_items oi ON o.order\_id = oi.order\_id

JOIN products p ON oi.product\_id = p.product\_id

WHERE o.status = 'Completed'

GROUP BY u.city, p.category

ORDER BY u.city, total\_revenue DESC;

**Peak revenue months/days**

SELECT

FORMAT(order\_date, 'yyyy-MM') AS month\_year,

SUM(order\_amount) AS total\_revenue

FROM orders

WHERE status = 'Completed'

GROUP BY FORMAT(order\_date, 'yyyy-MM')

ORDER BY total\_revenue DESC;

SELECT TOP 1

FORMAT(order\_date, 'yyyy-MM') AS peak\_month,

SUM(order\_amount) AS total\_revenue

FROM orders

WHERE status = 'Completed'

GROUP BY FORMAT(order\_date, 'yyyy-MM')

ORDER BY total\_revenue DESC

## 2. Peak Revenue Days

SELECT

CAST(order\_date AS DATE) AS order\_day,

SUM(order\_amount) AS total\_revenue

FROM orders

WHERE status = 'Completed'

GROUP BY CAST(order\_date AS DATE)

ORDER BY total\_revenue DESC;

SELECT TOP 1

CAST(order\_date AS DATE) AS peak\_day,

SUM(order\_amount) AS total\_revenue

FROM orders

WHERE status = 'Completed'

GROUP BY CAST(order\_date AS DATE)

ORDER BY total\_revenue DESC;

### B. **Customer Behavior & Segmentation**

| **Question** | **What it shows** |
| --- | --- |
| 6. Repeat vs. one-time customers % | Loyalty |
| 7. Avg order value per user | Buyer strength |
| 8. Users who increased/decreased purchase frequency | Retention signals |
| 9. Top 5 customers by spend | High-value clients |
| 10. Inactive users (no orders in 90+ days) | Churn risk |

**Repeat vs. one-time customers %**

WITH customer\_orders AS (

SELECT

user\_id,

COUNT(order\_id) AS total\_orders

FROM orders

WHERE status = 'Completed'

GROUP BY user\_id

)

, classified\_users AS (

SELECT

user\_id,

total\_orders,

CASE

WHEN total\_orders = 1 THEN 'One-Time'

ELSE 'Repeat'

END AS customer\_type

FROM customer\_orders

)

SELECT

customer\_type,

COUNT(\*) AS customer\_count,

ROUND(

COUNT(\*) \* 100.0 / (SELECT COUNT(\*) FROM customer\_orders),

2

) AS percentage\_of\_users

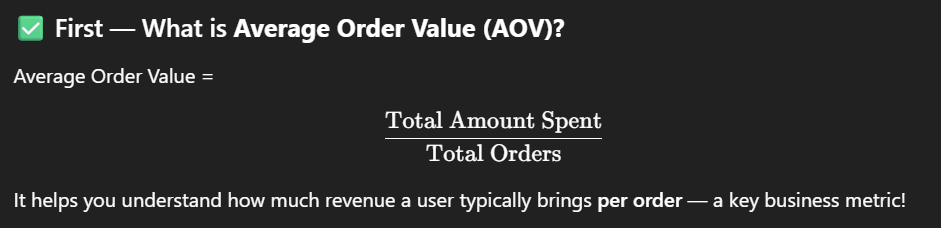
FROM classified\_users

GROUP BY customer\_type;

**Sample Output:**

| **customer\_type** | **customer\_count** | **percentage\_of\_users** |
| --- | --- | --- |
| One-Time | 1,200 | 60.00% |
| Repeat | 800 | 40.00% |

**Avg order value per user**

****

SELECT

user\_id,

COUNT(order\_id) AS total\_orders,

SUM(order\_amount) AS total\_spent,

ROUND(

SUM(order\_amount) \* 1.0 / COUNT(order\_id), 2

) AS avg\_order\_value

FROM orders

WHERE status = 'Completed'

GROUP BY user\_id

ORDER BY avg\_order\_value DESC;

. **Users who increased/decreased purchase frequency?**

-- Step 1: Monthly order count per user

WITH user\_monthly\_orders AS (

SELECT

user\_id,

FORMAT(order\_date, 'yyyy-MM') AS order\_month,

COUNT(order\_id) AS monthly\_orders

FROM orders

WHERE status = 'Completed'

GROUP BY user\_id, FORMAT(order\_date, 'yyyy-MM')

),

-- Step 2: Use LAG() to get previous month's order count

user\_order\_trend AS (

SELECT

user\_id,

order\_month,

monthly\_orders,

LAG(monthly\_orders) OVER (PARTITION BY user\_id ORDER BY order\_month) AS prev\_month\_orders

FROM user\_monthly\_orders

)

-- Step 3: Final Output – Detect Increase/Decrease

SELECT

user\_id,

order\_month,

monthly\_orders,

prev\_month\_orders,

CASE

WHEN prev\_month\_orders IS NULL THEN 'New Customer'

WHEN monthly\_orders > prev\_month\_orders THEN 'Increased'

WHEN monthly\_orders < prev\_month\_orders THEN 'Decreased'

ELSE 'No Change'

END AS purchase\_trend

FROM user\_order\_trend

ORDER BY user\_id, order\_month;

**Sample Output:**

| **user\_id** | **order\_month** | **monthly\_orders** | **prev\_month\_orders** | **purchase\_trend** |
| --- | --- | --- | --- | --- |
| 101 | 2025-05 | 2 | NULL | New Customer |
| 101 | 2025-06 | 3 | 2 | Increased |
| 101 | 2025-07 | 1 | 3 | Decreased |
| 102 | 2025-06 | 1 | NULL | New Customer |
| 102 | 2025-07 | 1 | 1 | No Change |

**Identify users who haven’t made any order in the last 90 days from today (or from the latest order date in your dataset).**

WITH last\_order\_per\_user AS (

SELECT

user\_id,

MAX(order\_date) AS last\_order\_date

FROM orders

WHERE status = 'Completed'

GROUP BY user\_id

)

SELECT

u.user\_id,

u.last\_order\_date,

DATEDIFF(DAY, u.last\_order\_date, GETDATE()) AS days\_since\_last\_order

FROM last\_order\_per\_user u

WHERE DATEDIFF(DAY, u.last\_order\_date, GETDATE()) > 90

ORDER BY days\_since\_last\_order DESC;

**Sample Output:**

| **user\_id** | **last\_order\_date** | **days\_since\_last\_order** |
| --- | --- | --- |
| 104 | 2025-03-10 | 125 |
| 203 | 2025-01-20 | 176 |

### 🔸 C. **Product-Level Insights**

| **🔍 Question** | **💡 What it shows** |
| --- | --- |
| 11. Inventory pressure products | Out-of-stock risks |
| 12. Products sold in most cities | Market coverage |
| 13. Products with highest return rate (if applicable) | Quality issues |
| 14. Price bands contributing most revenue | Pricing strategy |
| **Inventory pressure products**  Let’s dive into **“Inventory Pressure Products”** — a **real-world logistics + sales insight** used in:   * 📦 Inventory planning * 🧯 Stockout prevention * 🔁 Supply chain balancing  🎯 What Are Inventory Pressure Products? These are products that:   * Have **low available stock** * **AND** have **high demand** (high recent sales)   This combination = 🚨 **high risk of stockouts** = inventory pressure  -- Step 1: Get total stock per product  WITH current\_stock AS (  SELECT  product\_id,  SUM(stock\_quantity) AS total\_stock  FROM inventory  GROUP BY product\_id  ),  -- Step 2: Get recent demand (last 30 days)  recent\_demand AS (  SELECT  oi.product\_id,  SUM(oi.quantity) AS recent\_sales  FROM order\_items oi  JOIN orders o ON oi.order\_id = o.order\_id  WHERE o.status = 'Completed'  AND o.order\_date >= DATEADD(DAY, -30, GETDATE())  GROUP BY oi.product\_id  )  -- Step 3: Combine and identify pressure  SELECT  p.product\_id,  p.product\_name,  cs.total\_stock,  rd.recent\_sales,  ROUND(IIF(rd.recent\_sales = 0, 0, CAST(cs.total\_stock AS FLOAT) / rd.recent\_sales), 2) AS stock\_to\_sales\_ratio  FROM products p  JOIN current\_stock cs ON p.product\_id = cs.product\_id  JOIN recent\_demand rd ON p.product\_id = rd.product\_id  WHERE cs.total\_stock < rd.recent\_sales  ORDER BY stock\_to\_sales\_ratio ASC;  **What You Get:**   | **product\_id** | **product\_name** | **total\_stock** | **recent\_sales** | **stock\_to\_sales\_ratio** | | --- | --- | --- | --- | --- | | 101 | iPhone 15 | 20 | 100 | 0.20 | | 202 | Nike Shoes | 50 | 200 | 0.25 |   **Products sold in most cities**  Find products that were **ordered in the highest number of unique cities**.  SELECT  p.product\_id,  p.product\_name,  COUNT(DISTINCT u.city) AS cities\_sold\_in  FROM orders o  JOIN users u ON o.user\_id = u.user\_id  JOIN order\_items oi ON o.order\_id = oi.order\_id  JOIN products p ON oi.product\_id = p.product\_id  WHERE o.status = 'Completed'  GROUP BY p.product\_id, p.product\_name  ORDER BY cities\_sold\_in DESC;  **Products with highest return rate** Goal: For each product, calculate:   * **Total units sold** * **Total units returned** * Then compute the **return rate**:     **✅ Tables You’ll Need:**   | **Table** | **Key Columns** | **Purpose** | | --- | --- | --- | | order\_items | order\_id, product\_id, quantity | To count how many units sold | | returns | order\_id, product\_id, return\_quantity, return\_reason | To count how many were returned | | products | product\_id, product\_name | For product info |   -- Step 1: Total quantity sold per product  WITH sold\_qty AS (  SELECT  product\_id,  SUM(quantity) AS total\_sold  FROM order\_items  GROUP BY product\_id  ),  -- Step 2: Total quantity returned per product  returned\_qty AS (  SELECT  product\_id,  SUM(return\_quantity) AS total\_returned  FROM returns  GROUP BY product\_id  )  -- Step 3: Combine and calculate return rate  SELECT  p.product\_id,  p.product\_name,  s.total\_sold,  ISNULL(r.total\_returned, 0) AS total\_returned,  ROUND(  ISNULL(r.total\_returned \* 100.0 / NULLIF(s.total\_sold, 0), 0),  2  ) AS return\_rate\_percentage  FROM sold\_qty s  JOIN products p ON s.product\_id = p.product\_id  LEFT JOIN returned\_qty r ON s.product\_id = r.product\_id  ORDER BY return\_rate\_percentage DESC; |  |

### 🔸 D. **Order Fulfillment & Cancellations**

| **🔍 Question** | **💡 What it shows** |
| --- | --- |
| 15. % cancelled orders by category | Return/cancel hotspots |
| 16. Payment method trends | Preferred modes |
| 17. Daily/monthly order volume trend | Supply planning |
| 18. Avg order items per cart | Buyer behavior |
| 19. % orders with only 1 item | Upsell opportunity |

**% cancelled orders by category**

Goal:

For each product category, calculate:

* Total number of orders
* Number of cancelled orders
* Then compute:

****

**✅ Tables You’ll Need:**

| **Table** | **Key Columns** | **Purpose** |
| --- | --- | --- |
| orders | order\_id, status, order\_date | To identify cancelled orders |
| order\_items | order\_id, product\_id | Link orders to products |
| products | product\_id, category | To get product categories |

**-- Step 1: Get total and cancelled orders per category**

WITH category\_orders AS (

SELECT

p.category,

COUNT(DISTINCT o.order\_id) AS total\_orders,

COUNT(DISTINCT CASE WHEN o.status = 'Cancelled' THEN o.order\_id END) AS cancelled\_orders

FROM orders o

JOIN order\_items oi ON o.order\_id = oi.order\_id

JOIN products p ON oi.product\_id = p.product\_id

GROUP BY p.category

)

-- Step 2: Calculate cancellation %

SELECT

category,

total\_orders,

cancelled\_orders,

ROUND(

cancelled\_orders \* 100.0 / NULLIF(total\_orders, 0),

2

) AS cancellation\_percentage

FROM category\_orders

ORDER BY cancellation\_percentage DESC;

**Payment method trends**

## Goal:

Track how **usage of each payment method** changes **over time** — by:

* Daily / Monthly / Quarterly
* % share of each method per period
* **✅ Tables You'll Need:**

| **Table** | **Columns** | **Purpose** |
| --- | --- | --- |
| orders | order\_id, payment\_method, order\_date, status, order\_amount | Stores payment details |

For each month, how much each payment method was used (by count or revenue), and its **% share**.

-- Step 1: Aggregate by payment method per month

WITH monthly\_payment\_usage AS (

SELECT

FORMAT(order\_date, 'yyyy-MM') AS order\_month,

payment\_method,

COUNT(\*) AS total\_orders,

SUM(order\_amount) AS total\_amount

FROM orders

WHERE status = 'Completed'

GROUP BY FORMAT(order\_date, 'yyyy-MM'), payment\_method

),

-- Step 2: Total orders per month (for % share calc)

monthly\_totals AS (

SELECT

FORMAT(order\_date, 'yyyy-MM') AS order\_month,

COUNT(\*) AS total\_month\_orders,

SUM(order\_amount) AS total\_month\_amount

FROM orders

WHERE status = 'Completed'

GROUP BY FORMAT(order\_date, 'yyyy-MM')

)

-- Step 3: Join and calculate share

SELECT

mpu.order\_month,

mpu.payment\_method,

mpu.total\_orders,

mpu.total\_amount,

ROUND(mpu.total\_orders \* 100.0 / mt.total\_month\_orders, 2) AS order\_share\_percent,

ROUND(mpu.total\_amount \* 100.0 / mt.total\_month\_amount, 2) AS revenue\_share\_percent

FROM monthly\_payment\_usage mpu

JOIN monthly\_totals mt

ON mpu.order\_month = mt.order\_month

ORDER BY mpu.order\_month, order\_share\_percent DESC;

**Daily/monthly order volume trend**

## Goal:

Show **how many orders** were placed **each day or month**, and track the **trend over time**.

SELECT

CAST(order\_date AS DATE) AS order\_day,

COUNT(order\_id) AS total\_orders

FROM orders

WHERE status = 'Completed'

GROUP BY CAST(order\_date AS DATE)

ORDER BY order\_day;

SELECT

FORMAT(order\_date, 'yyyy-MM') AS order\_month,

COUNT(order\_id) AS total\_orders

FROM orders

WHERE status = 'Completed'

GROUP BY FORMAT(order\_date, 'yyyy-MM')

ORDER BY order\_month;

**Avg order items per cart**

## Goal:

For each completed order, count how many **items** were in the cart, then find the **average across all orders**.

****

-- Step 1: Total items per order

WITH items\_per\_order AS (

SELECT

oi.order\_id,

SUM(oi.quantity) AS total\_items

FROM order\_items oi

JOIN orders o ON oi.order\_id = o.order\_id

WHERE o.status = 'Completed'

GROUP BY oi.order\_id

)

-- Step 2: Get the average

SELECT

ROUND(AVG(CAST(total\_items AS FLOAT)), 2) AS avg\_items\_per\_cart

FROM items\_per\_order;

**% orders with only 1 item**

**“% of orders with only 1 item”** is a valuable metric that tells you:

* 🧾 How many customers make minimal purchases
* 🛒 Opportunities for **upselling** or **product bundling**
* 📉 Helps reduce shipping cost per order by encouraging larger carts

## Goal:

Calculate the **percentage of completed orders** where the **total quantity of items = 1**.

-- Step 1: Calculate total items per order

WITH order\_item\_counts AS (

SELECT

oi.order\_id,

SUM(oi.quantity) AS total\_items

FROM order\_items oi

JOIN orders o ON oi.order\_id = o.order\_id

WHERE o.status = 'Completed'

GROUP BY oi.order\_id

)

-- Step 2: Compute percentage of single-item orders

SELECT

COUNT(CASE WHEN total\_items = 1 THEN 1 END) \* 100.0 / COUNT(\*) AS one\_item\_order\_percentage

FROM order\_item\_counts;

**Explanation:**

| **Part** | **Meaning** |
| --- | --- |
| SUM(quantity) | Totals number of items per order |
| CASE WHEN total\_items = 1 THEN 1 | Flags single-item orders |
| COUNT(\*) | Total number of completed orders |
| \* 100.0 | Converts to percentage |

### 🔸 E. **📐 Percentage-Based KPIs (Interview Focus)**

| **✅ Question** | **Example SQL Concept** |
| --- | --- |
| 20. % of users who made purchases after signup |  |
| 21. % orders per payment method |  |
| 22. % revenue from top 10 products |  |
| 23. % customers who bought from multiple categories |  |
| 24. % revenue growth MoM |  |

**% of users who made purchases after signup**

**“% of users who made purchases after signup”** is a **critical retention metric** that tells you:

* 🧍‍♂️ How many signups actually become customers
* 📉 Where your funnel drops off (signup → purchase)
* 🎯 How well your onboarding process converts

## Goal:

Calculate the **percentage of users who placed at least one order** after signing up.

-- Step 1: Users who placed at least one completed order

WITH purchasing\_users AS (

SELECT DISTINCT user\_id

FROM orders

WHERE status = 'Completed'

)

-- Step 2: Calculate percentage

SELECT

ROUND(

COUNT(pu.user\_id) \* 100.0 / COUNT(u.user\_id),

2

) AS purchase\_after\_signup\_percentage

FROM users u

LEFT JOIN purchasing\_users pu ON u.user\_id = pu.user\_id;

**% orders per payment method?**

Excellent!   
**“% Orders per Payment Method”** is a classic metric for understanding **customer behavior**, **checkout preferences**, and **payment performance.** 💳

SELECT

payment\_method,

COUNT(\*) AS total\_orders,

ROUND(

COUNT(\*) \* 100.0 / (SELECT COUNT(\*) FROM orders WHERE status = 'Completed'),

2

) AS percentage\_of\_orders

FROM orders

WHERE status = 'Completed'

GROUP BY payment\_method

ORDER BY percentage\_of\_orders DESC;

22. % revenue from top 10 products

-- Step 1: Total revenue per product

WITH product\_revenue AS (

SELECT

product\_id,

SUM(order\_amount) AS revenue

FROM order\_items

GROUP BY product\_id

),

-- Step 2: Top 10 revenue products

top\_10 AS (

SELECT TOP 10 product\_id, revenue

FROM product\_revenue

ORDER BY revenue DESC

),

-- Step 3: Final percentage calculation

total AS (

SELECT SUM(revenue) AS total\_revenue FROM product\_revenue

)

SELECT

ROUND(SUM(t.revenue) \* 100.0 / total.total\_revenue, 2) AS percent\_revenue\_top\_10

FROM top\_10 t

CROSS JOIN total;

**% Customers Who Bought from Multiple Categories**

Find the percentage of customers who purchased products from **2 or more different categories**.

-- Step 1: Get customer-category combinations

WITH customer\_categories AS (

SELECT

o.customer\_id,

p.category

FROM orders o

JOIN order\_items oi ON o.order\_id = oi.order\_id

JOIN products p ON oi.product\_id = p.product\_id

GROUP BY o.customer\_id, p.category

),

-- Step 2: Count how many categories per customer

category\_count AS (

SELECT

customer\_id,

COUNT(DISTINCT category) AS category\_count

FROM customer\_categories

GROUP BY customer\_id

),

-- Step 3: Final % calculation

total\_customers AS (

SELECT COUNT(\*) AS total FROM category\_count

)

SELECT

ROUND(

COUNT(CASE WHEN category\_count > 1 THEN 1 END) \* 100.0 / total\_customers.total,

2

) AS percent\_multi\_category\_customers

FROM category\_count

CROSS JOIN total\_customers;

24. **% Revenue Growth Month-over-Month (MoM)**

Compare monthly revenue and compute **growth %** over the previous month.

-- Step 1: Monthly revenue

WITH monthly\_revenue AS (

SELECT

FORMAT(order\_date, 'yyyy-MM') AS order\_month,

SUM(order\_amount) AS total\_revenue

FROM orders

WHERE status = 'Completed'

GROUP BY FORMAT(order\_date, 'yyyy-MM')

),

-- Step 2: Add previous month's revenue using LAG()

revenue\_with\_growth AS (

SELECT

order\_month,

total\_revenue,

LAG(total\_revenue) OVER (ORDER BY order\_month) AS prev\_month\_revenue

FROM monthly\_revenue

)

-- Step 3: Calculate growth %

SELECT

order\_month,

total\_revenue,

prev\_month\_revenue,

ROUND(

(total\_revenue - prev\_month\_revenue) \* 100.0 / NULLIF(prev\_month\_revenue, 0),

2

) AS percent\_growth

FROM revenue\_with\_growth;