Sample Case Study: E-Commerce Analytics

Imagine you're working with an e-commerce platform.
You’re asked to help uncover insights related to product sales, customer behaviour, and category trends.

You’re given these three tables:

1. orders(order_id, customer_id, order_date, total_amount)
2. order_items(order_id, product_id, quantity, item_price)
3. products(product_id, category, product_name)

Step 1 – Basic Aggregation
What’s the total revenue per customer?

```
SELECT customer_id, SUM(total_amount) AS total_revenue
FROM orders
GROUP BY customer_id;
```

Step 2 – Join + Aggregation
What’s the total revenue per product category?

```
SELECT p.category, SUM(oi.quantity * oi.item_price) AS category_revenue
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.category;
```

Step 3 – Time Filtering + Business Logic
Top 3 categories by revenue in the last 3 months

```
SELECT p.category, SUM(oi.quantity * oi.item_price) AS revenue
FROM orders o
```

```
JOIN order_items oi ON o.order_id = oi.order_id
JOIN products p ON oi.product_id = p.product_id
WHERE o.order_date >= CURRENT_DATE - INTERVAL 3 MONTH
GROUP BY p.category
ORDER BY revenue DESC
LIMIT 3;
```

Step 4 – Window Functions
Rank products by revenue within each category

```
SELECT
p.category,
p.product_name,
SUM(oi.quantity * oi.item_price) AS product_revenue,
RANK() OVER (PARTITION BY p.category ORDER BY SUM(oi.quantity *
oi.item_price) DESC) AS category_rank
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.category, p.product_name;
```

Step 5 – Behavioural Insight (Window or Self-Join)
Days since a customer’s previous order

```
SELECT
  customer_id,
  order_id,
  order_date,
  LAG(order_date) OVER (PARTITION BY customer_id ORDER BY order_date) AS
previous_order_date,
  DATEDIFF(order_date, LAG(order_date) OVER (PARTITION BY customer_id ORDER
BY order_date)) AS days_since_last_order
FROM orders;
```

Step 6 – Behavioural Insight — Days since a customer’s previous order, flag customers with more than 60 days gap in order date

```
SELECT
  customer_id,
  order_id,
  order_date,
  LAG(order_date) OVER (PARTITION BY customer_id ORDER BY order_date) AS previous_order_date,
  DATEDIFF(order_date, LAG(order_date) OVER (PARTITION BY customer_id ORDER BY order_date)) AS days_since_last_order,
  CASE
        WHEN DATEDIFF(order_date, LAG(order_date) OVER (PARTITION BY customer_id ORDER BY order_date)) > 60
        THEN 'FLAGGED'
        ELSE NULL
  END AS gap_flag
FROM orders;
```