○ **Question 1**

Calculate the total sales for each store.

○ **Task:** Use aggregate functions like SUM and GROUP BY to calculate total sales from the orders and order\_items tables.

**Query:- select \* from customers join orders On customers.customer\_id=orders.customer\_id;**

**Question 2**

Aggregate Functions:

○ Question: Calculate the total sales for each store.

○ **Task**: Use aggregate functions like SUM and GROUP BY to calculate total sales from the orders and order\_items tables.

**Query:- SELECT**

**o.store\_id,**

**SUM(oi.quantity \* oi.list\_price) AS total\_sales**

**FROM**

**orders o**

**JOIN**

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

**GROUP BY**

**o.store\_id;**

**● Subqueries:**

**○ Question: Find the products that have never been ordered.**

**○ Task: Write a subquery to identify products in the products table that do not appear in the order\_items table.**

**SELECT p.product\_id, p.product\_name**

**FROM products p**

**WHERE p.product\_id NOT IN (**

**SELECT oi.product\_id**

**FROM order\_items oi**

**);**

**● Joins:**

**○ Question: Retrieve the names and email addresses of staff along with the names of their managers.**

**○ Task: Write a SQL query to join the staffs table with itself to get staff and their manager information.**

**SELECT**

**CONCAT(s.first\_name, ' ', s.last\_name) AS staff\_name,**

**s.email AS staff\_email,**

**CONCAT(m.first\_name, ' ', m.last\_name) AS manager\_name**

**FROM staffs s**

**LEFT JOIN staffs m ON s.manager\_id = m.staff\_id;**

**● Window Functions:**

**○ Question: Rank stores based on their total sales.**

**○ Task: Use window functions like ROW\_NUMBER() or RANK() to rank stores based on total sales.**

**SELECT**

**o.store\_id,**

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

**RANK() OVER (ORDER BY SUM(oi.item\_price \* oi.quantity) DESC) AS sales\_rank**

**FROM orders o**

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

**GROUP BY o.store\_id;**

**● Date Functions:**

**○ Question: Calculate the number of days each order took to ship.**

**○ Task: Use date functions to calculate the difference between shipped\_date and order\_date in the orders table.**

**SELECT**

**order\_id,**

**order\_date,**

**shipped\_date,**

**DATEDIFF(shipped\_date, order\_date) AS days\_to\_ship**

**FROM orders;**

**● Complex Joins:**

**○ Question: Retrieve all orders along with the product names and the store names.**

**○ Task: Write a SQL query to join the orders, order\_items, products, and stores tables.**

**SELECT**

**order\_id,**

**order\_status,**

**CASE**

**WHEN order\_status = 'Pending' THEN 'Awaiting Fulfillment'**

**WHEN order\_status = 'Shipped' THEN 'In Transit'**

**WHEN order\_status = 'Delivered' THEN 'Completed'**

**WHEN order\_status = 'Cancelled' THEN 'Closed'**

**ELSE 'Other'**

**END AS status\_category**

**FROM orders;**

**● Temporary Tables:**

**○ Question: Create a temporary table to store intermediate sales calculations.**

**○ Task: Use CREATE TEMPORARY TABLE to store results of a complex calculation for further analysis.**

**CREATE TEMPORARY TABLE temp\_store\_sales AS**

**SELECT**

**o.store\_id,**

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

**FROM orders o**

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

**GROUP BY o.store\_id;**

* **Case Statements:**
  + **Question:** Categorize orders based on their status.
  + **Task:** Use CASE statements to categorize orders in the orders table into different status groups.

**SELECT**

**order\_id,**

**customer\_id,**

**order\_date,**

**order\_status,**

**CASE**

**WHEN order\_status = 3 THEN 'Not Delivered'**

**WHEN order\_status = 4 THEN 'Delivered'**

**ELSE 'Unknown Status'**

**END AS order\_status\_category**

**FROM orders;**