**Assignment 3:** Utilize a subquery to find customers who have placed orders above the

average order value, and write a UNION query to combine two SELECT statements with

the same number of columns.

Let's break down the task into two parts: using a subquery to find customers who have placed orders above the average order value, and then combining the results using a UNION query.

**Part 1:** Using a subquery to find customers with orders above average order value

Assuming you have a **`customers`** table and an **`orders`** table, and you want to find customers who have placed orders above the average order value:

SELECT c.customer\_id, c.customer\_name, o.order\_id, o.order\_amount

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

WHERE o.order\_amount > (

SELECT AVG(order\_amount)

FROM orders

);

**- Inner Query**: **`(SELECT AVG(order\_amount) FROM orders)`**: This subquery calculates the average order amount from the **`orders`** table.

**- Main Query:**

**- `SELECT c.customer\_id, c.customer\_name, o.order\_id, o.order\_amount`**: Selects columns from both customers (**`c`**) and orders (**`o`**) tables.

**- `JOIN orders o ON c.customer\_id = o.customer\_id`**: Joins **`customers`** and **`orders`** tables based on **`customer\_id`**.

**- `WHERE o.order\_amount > (...)`**: Filters rows where the **`order\_amount`** is greater than the average order amount obtained from the subquery.

This query will return customers along with their orders where the order amount exceeds the average order amount across all orders.

**Part 2: Using UNION to combine two SELECT statements**

Next, let's combine two SELECT statements using UNION. For this example, let's say we want to combine the above query with another query that retrieves customers who have not placed any orders:

-- First SELECT statement: Customers with orders above average order value

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

WHERE o.order\_amount > (

SELECTSELECT c.customer\_id, c.customer\_name, o.order\_id, o.order\_amount

AVG(order\_amount)

FROM orders

)

-- UNION operator to combine results

UNION

-- Second SELECT statement: Customers who have not placed any orders

SELECT c.customer\_id, c.customer\_name, NULL AS order\_id, NULL AS order\_amount

FROM customers c

WHERE NOT EXISTS (

SELECT 1 FROM orders o WHERE o.customer\_id = c.customer\_id

);

**- First SELECT Statement**: Retrieves customers who have placed orders above the average order value, similar to the query in Part 1.

**- Second SELECT Statement:** Retrieves customers who have not placed any orders:

**- `NULL AS order\_id**, **NULL AS order\_amount`**: Provides placeholder columns for **`order\_id`** and **`order\_amount`** since these customers haven't placed any orders.

**- `NOT EXISTS (...)`**: Checks if there are no corresponding rows in the **`orders`** table for the customer (**`c.customer\_id`**).

**- UNION:** Combines the results of the two SELECT statements. The UNION operator requires both SELECT statements to have the same number of columns, and it removes duplicate rows by default.

**Notes:**

**-** Ensure that the column types and number of columns match between the SELECT statements when using UNION.

**-** The first SELECT statement retrieves customers who have orders above the average order value, while the second SELECT statement retrieves customers who have not placed any orders.

**-** Adjust column names and conditions **(`WHERE`** clauses) based on your actual schema and requirements.

This approach effectively combines two different queries into a single result set using the UNION operator.