Coding challenge

SQL

Name: Rohan Vinayak Chaudhari

Batch: Data Engineering 1

<u>Question1:</u> Execute OVER and PARTITION BY Clause in SQL Queries , creating subtotals &Total Aggregations using SQL Queries.

1)Create database & using it:

Creates a database named coding_challenge_1.

Switches to using the newly created database.

```sql
/*Create a database*/
CREATE DATABASE IF NOT EXISTS coding_challenge_1;
USE coding_challenge_1;
```

2)Creating table Order:

Creates a table named orders with columns order_id, customer_id, order_date, and order_amount.

```sql
/*Create a sample table*/
CREATE TABLE orders (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    order_amount DECIMAL(10, 2)
);
```

3)Inserting data into order table

Inserts sample data into the orders table.

```
/*Insert some sample data*/
INSERT INTO orders VALUES
(1, 101, '2024-01-01', 100.00),
(2, 102, '2024-01-02', 150.00),
(3, 101, '2024-01-03', 200.00),
(4, 103, '2024-01-04', 120.00);
```

4) Execute OVER and PARTITION BY Clause in SQL Queries

Use of the OVER and PARTITION BY clauses with the SUM aggregation(window)) function to calculate the total order amount for each customer.

The query calculates the total order amount for each customer using the SUM window function with the OVER and PARTITION BY clauses.

```
/*Use OVER and PARTITION BY  to calculate the total order
amount for each customer*/
SELECT order_id,customer_id,order_amount,
    SUM(order_amount) OVER (PARTITION BY customer_id)
    AS total_amount_per_customer
FROM orders;
```

5) creating subtotals &Total Aggregations using SQL Queries.

This query is useful for obtaining a summary of total order amounts, total orders, and average order amounts, min, max amount for each customer, along with subtotals and a grand total.

GROUP BY: A clause used in SQL to arrange identical data into groups. In this query, it groups rows based on the customer_id column.

SUM(): An aggregate function that adds up the values in a specified column. Here, it calculates the total order amount for each customer.

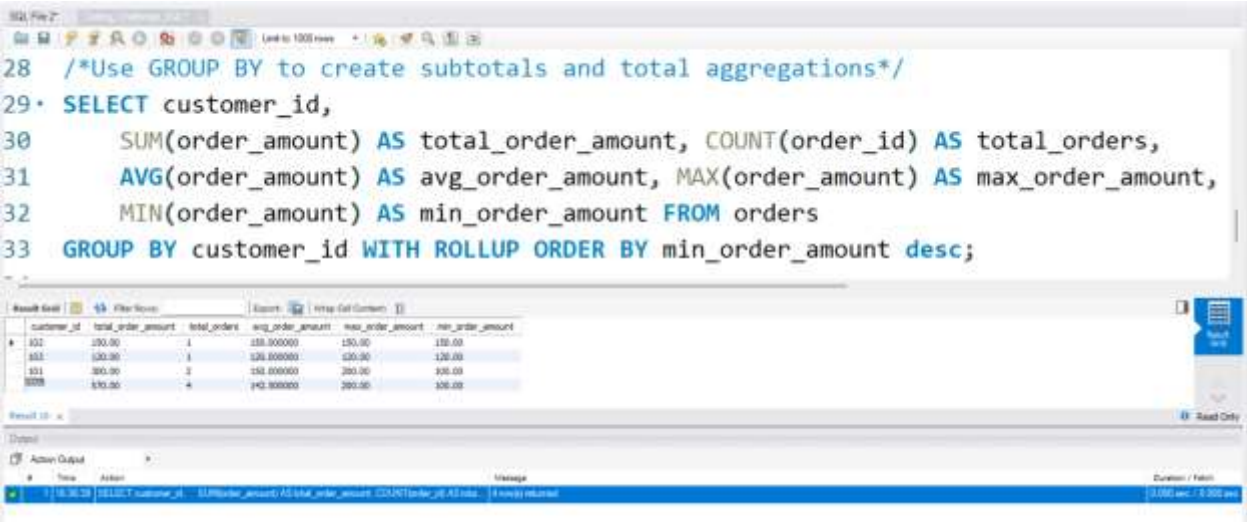
COUNT(): An aggregate function that counts the number of rows in a result set. In this query, it counts the total number of orders for each customer.

AVG(): An aggregate function that calculates the average of values in a specified column. Here, it computes the average order amount for each customer.

MAX(): An aggregate function that calculates the MAX of order value for a particular customer id. Here, it computes the average order amount for each customer.

MIN(): An aggregate function that calculates the MIN of order value for a particular customer id. Here, it computes the average order amount for each customer.

WITH ROLLUP: An extension to the GROUP BY clause that includes extra rows that represent subtotals and a grand total. The subtotals are generated for each level of grouping specified in the GROUP BY clause.



```
/*Use GROUP BY to create subtotals and total aggregations*/
SELECT customer_id,
    SUM(order_amount) AS total_order_amount, COUNT(order_id) AS total_orders,
    AVG(order_amount) AS avg_order_amount, MAX(order_amount) AS max_order_amount,
    MIN(order_amount) AS min_order_amount FROM orders
GROUP BY customer_id WITH ROLLUP ORDER BY min_order_amount desc;
```