Data Analytics Immersion - Task 3.8

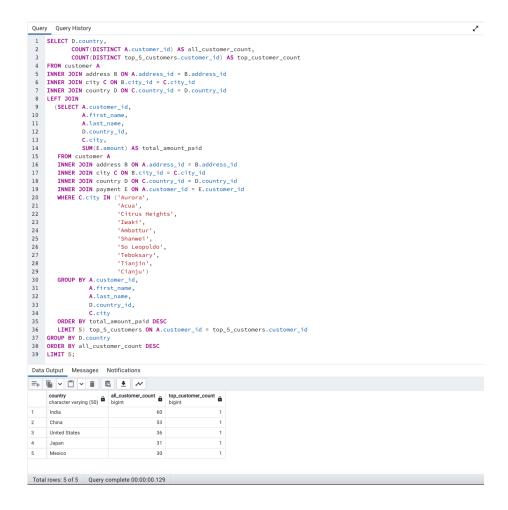
Step 1

```
Query Query History
1
   SELECT AVG(total_amount_paid) AS average
   FROM
2
3
        (SELECT
4
            customer.customer id.
5
           customer.first_name,
           customer.last_name,
 6
 7
           country.country,
 8
            city.city,
9
            SUM(payment.amount) AS total_amount_paid
10
        FROM
11
            customer
12
        JOIN
13
            payment ON customer.customer_id = payment.customer_id
14
        JOIN
15
            address ON customer.address_id = address.address_id
16
        JOIN
17
            city ON address.city_id = city.city_id
18
        JOIN
19
            country ON city.country_id = country.country_id
20
        WHERE
21
            city.city IN ('Aurora', 'London', 'Abu Dhabi', 'Adana', 'Addis Abeba')
22
        GROUP BY
           customer.customer_id, customer.first_name, customer.last_name, country.country, city.city
23
24
        ORDER BY
25
            total_amount_paid DESC
        LIMIT
26
27
            5);
Data Output Messages Notifications
   8
                         ! ~
                average
                   â
    numeric
     96.15800000000000000
```

The query is designed to calculate the average total amount paid by the top 5 customers from the specified cities:

- 1. The inner SELECT statement retrieves relevant information about customers, including their IDs, names, country, city, and the total amount they paid. It uses JOIN operations to connect tables such as 'customer', 'payment', 'address', 'city', and 'country'.
- 2. The WHERE clause filters the results to include only customers from specific cities: 'Aurora', 'London', 'Abu Dhabi', 'Adana', and 'Addis Abeba'.
- 3. The GROUP BY clause groups the results by customer ID, first name, last name, country, and city.
- 4. The ORDER BY clause arranges the results based on the total amount paid in descending order.
- 5. The LIMIT clause restricts the output to the top 5 customers.
- 6. The outer SELECT statement calculates the average of the total amount paid by the top 5 customers using the AVG function and aliases the result as 'average'.

Step 2



In the given scenario, steps 1 and 2 involve aggregating data and counting the number of customers both globally and within specific criteria, such as the top 5 cities. While it's technically possible to achieve similar results without using subqueries by employing advanced techniques like window functions or complex joins, subqueries provide a more straightforward and readable solution.

Subqueries are particularly useful when dealing with nested queries or scenarios where a result set depends on the outcome of another query. They enhance code readability, simplify complex logic, and enable a step-by-step approach to problem-solving. In this case, using a subquery allows for a clearer separation of concerns, making it easier to understand and maintain the overall query structure.