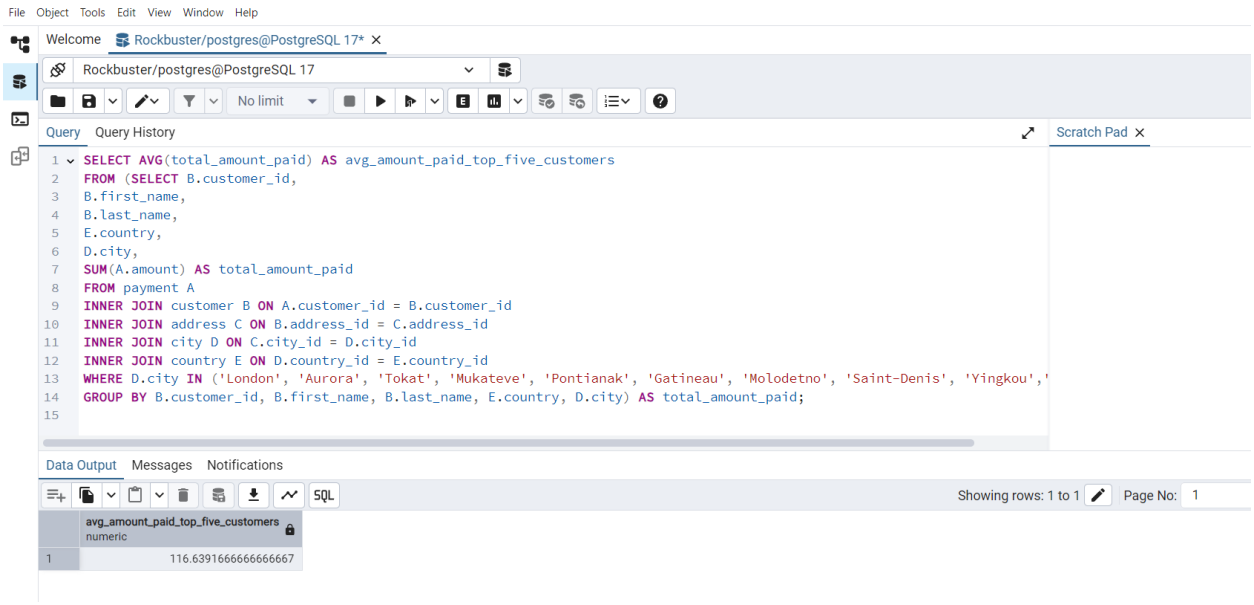
Step 1



SELECT AVG(total_amount_paid) AS avg_amount_paid_top_five_customers

FROM (SELECT B.customer_id,

B.first_name,

B.last_name,

E.country,

D.city,

SUM(A.amount) AS total_amount_paid

FROM payment A

INNER JOIN customer B ON A.customer_id = B.customer_id

INNER JOIN address C ON B.address_id = C.address_id

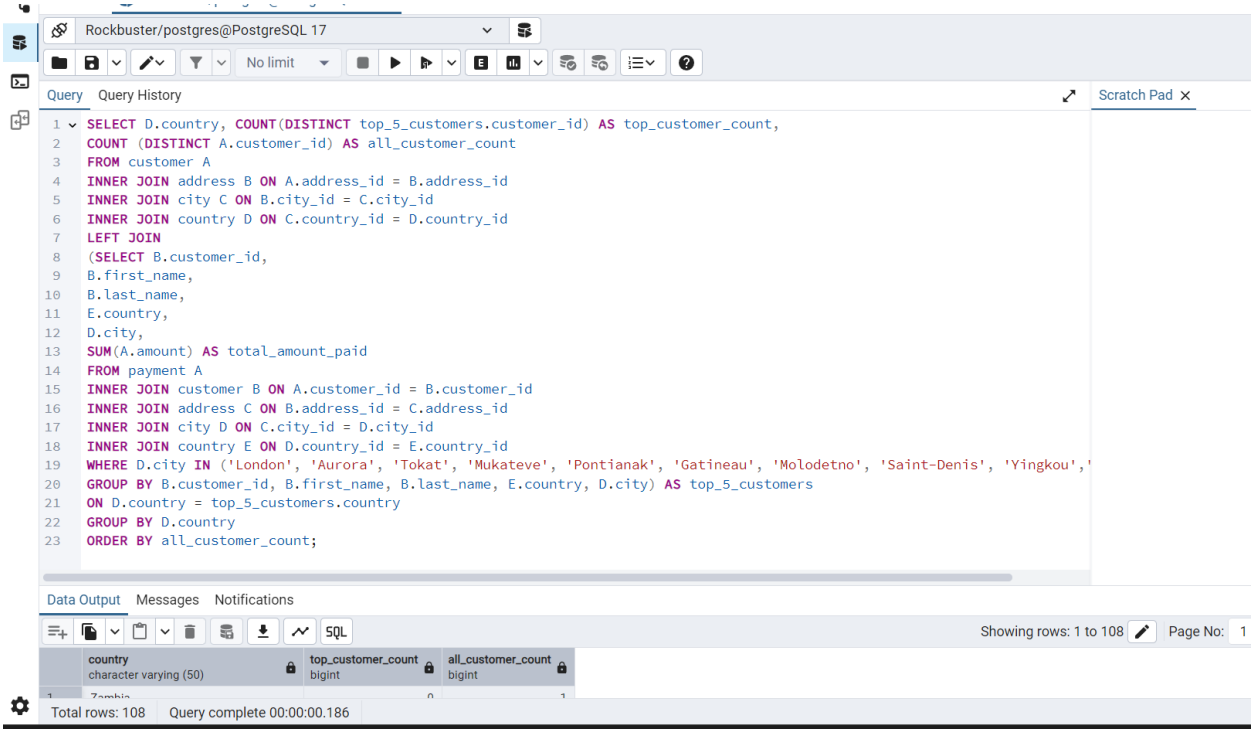
INNER JOIN city D ON C.city_id = D.city_id

INNER JOIN country E ON D.country_id = E.country_id

WHERE D.city IN ('London', 'Aurora', 'Tokat', 'Mukateve', 'Pontianak', 'Gatineau', 'Molodetno', 'Saint-Denis', 'Yingkou','Atlixco')

GROUP BY B.customer_id, B.first_name, B.last_name, E.country, D.city) AS total_amount_paid;

Step 2



SELECT D.country, COUNT(DISTINCT top_5_customers.customer_id) AS top_customer_count,

COUNT (DISTINCT A.customer_id) AS all_customer_count

FROM customer A

INNER JOIN address B ON A.address_id = B.address_id

INNER JOIN city C ON B.city_id = C.city_id

INNER JOIN country D ON C.country_id = D.country_id

LEFT JOIN

(SELECT B.customer_id,

B.first_name,

B.last_name,

E.country,

D.city,

SUM(A.amount) AS total_amount_paid

FROM payment A

INNER JOIN customer B ON A.customer_id = B.customer_id

```
INNER JOIN address C ON B.address_id = C.address_id

INNER JOIN city D ON C.city_id = D.city_id

INNER JOIN country E ON D.country_id = E.country_id

WHERE D.city IN ('London', 'Aurora', 'Tokat', 'Mukateve', 'Pontianak', 'Gatineau', 'Molodetno', 'Saint-Denis', 'Yingkou','Atlixco')

GROUP BY B.customer_id, B.first_name, B.last_name, E.country, D.city) AS top_5_customers

ON D.country = top_5_customers.country

GROUP BY D.country

ORDER BY all_customer_count;
```

Step 3

The approach can quickly become difficult to manage and understand, especially when dealing with large datasets or more advanced logic such as ranking, filtering, and aggregating. Subqueries help simplify the logic by isolating specific tasks, making the overall query easier to read, debug, and maintain.

Subqueries are especially useful when you need to perform an intermediate calculation, filter based on aggregated results or reuse a derived dataset multiple times within a larger query. They allow you to modularize your SQL code, making it clearer and more efficient by breaking down a large problem into smaller, logical steps.