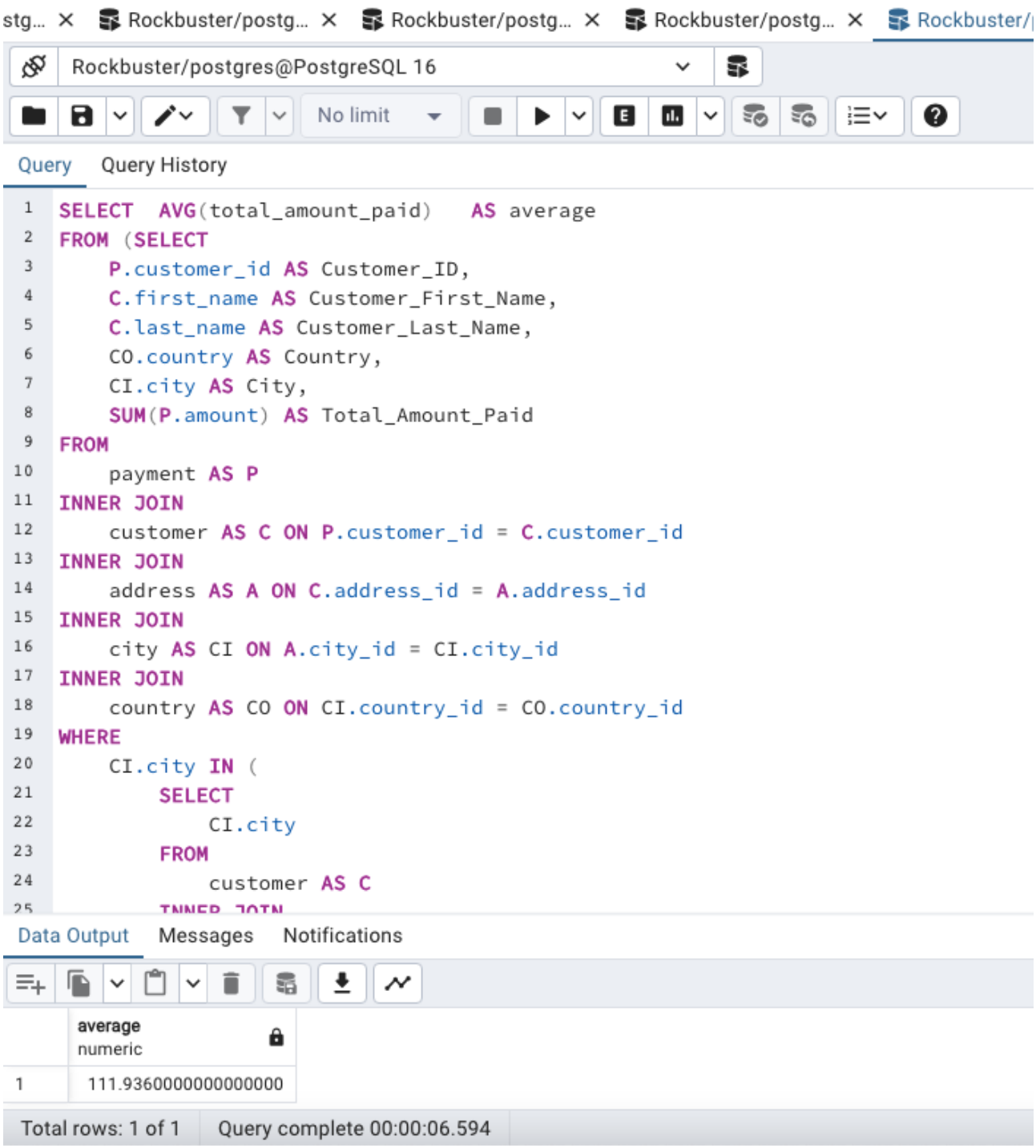
TASK 3.8

Performing Subqueries

1:

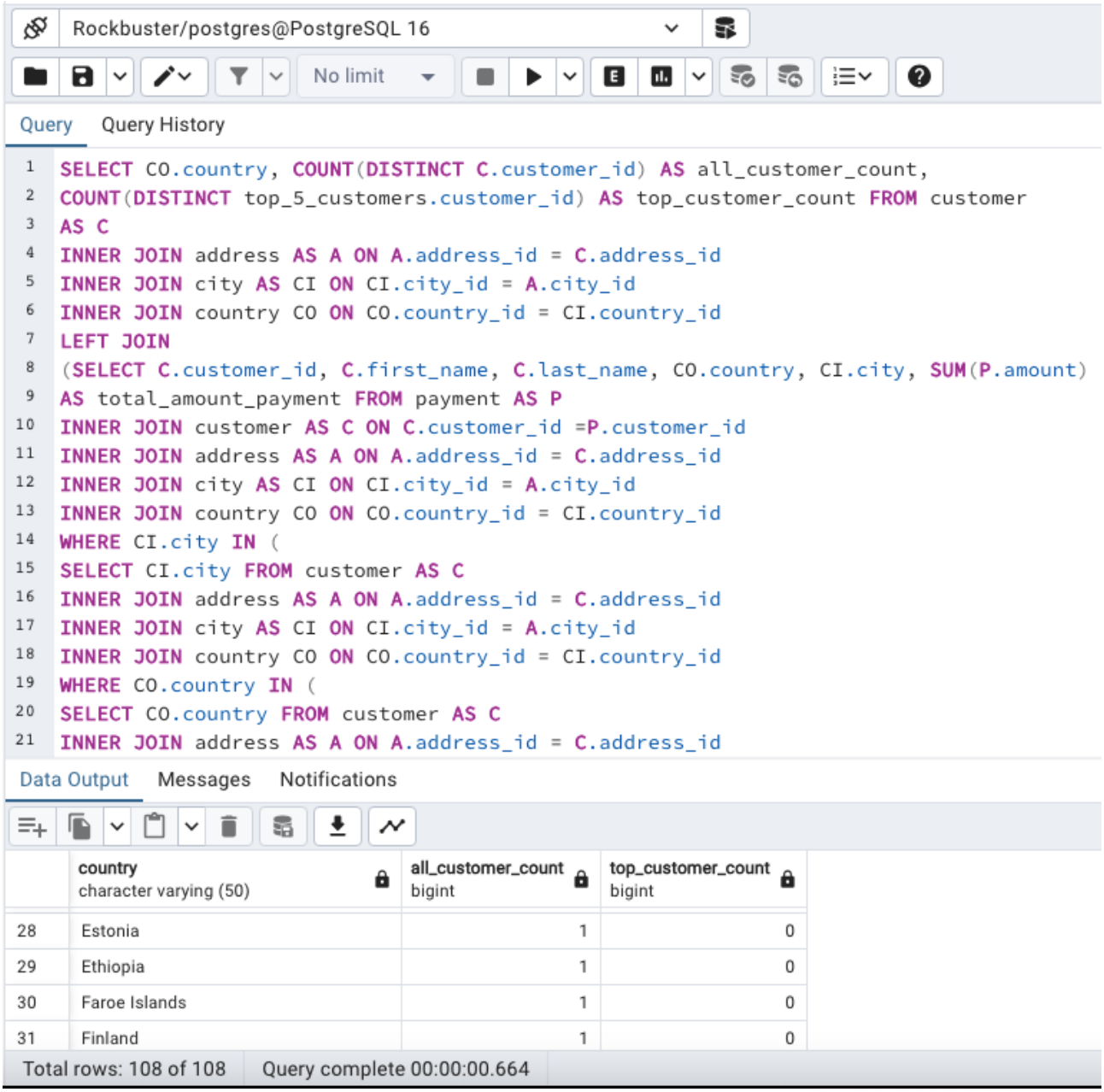
```
SELECT        AVG(total_amount_paid)      AS average
FROM (SELECT
   P.customer_id AS Customer_ID,
   C.first_name AS Customer_First_Name,
   C.last_name AS Customer_Last_Name,
   CO.country AS Country,
   CI.city AS City,
   SUM(P.amount) AS Total_Amount_Paid
FROM
   payment AS P
INNER JOIN
   customer AS C ON P.customer_id = C.customer_id
INNER JOIN
   address AS A ON C.address_id = A.address_id
INNER JOIN
   city AS CI ON A.city_id = CI.city_id
INNER JOIN
   country AS CO ON CI.country_id = CO.country_id
WHERE
   CI.city IN (
      SELECT
         CI.city
      FROM
         customer AS C
      INNER JOIN
         address AS A ON C.address_id = A.address_id
      INNER JOIN
         city AS CI ON A.city_id = CI.city_id
      INNER JOIN
         country AS CO ON CI.country_id = CO.country_id
      GROUP BY
         CI.city
      ORDER BY
         COUNT(DISTINCT C.customer_id) DESC
      LIMIT 10
   )
GROUP BY
   P.customer_id,
   C.first_name,
   C.last_name,
```

```
   CO.country,
   CI.city
ORDER BY
   Total_Amount_Paid DESC
LIMIT 5)AS total_amount_paid;
```



2:

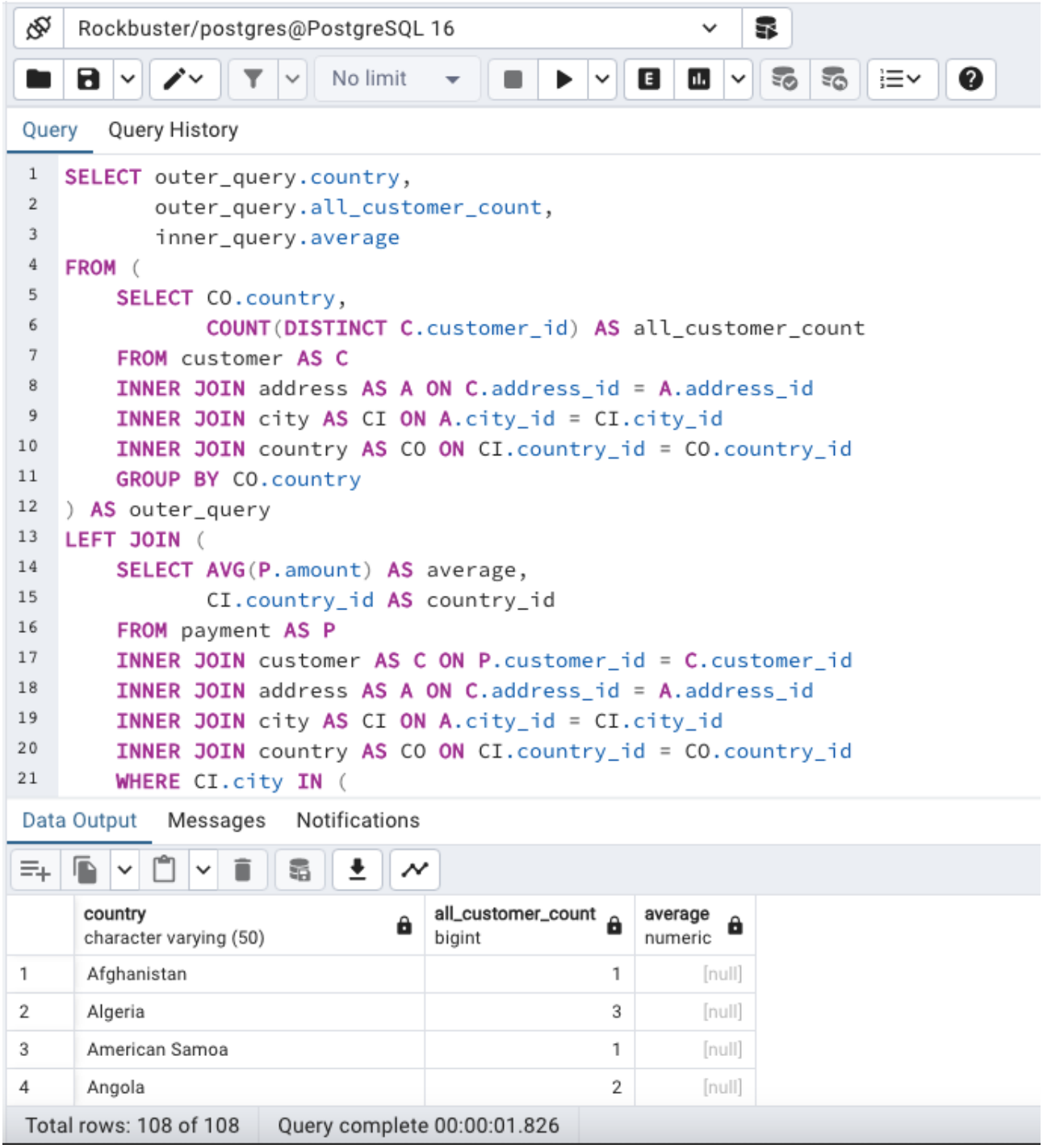
```
SELECT CO.country, COUNT(DISTINCT C.customer_id) AS all_customer_count,
COUNT(DISTINCT top_5_customers.customer_id) AS top_customer_count FROM customer
AS C
INNER JOIN address AS A ON A.address_id = C.address_id
INNER JOIN city AS CI ON CI.city_id = A.city_id
INNER JOIN country CO ON CO.country_id = CI.country_id
LEFT JOIN
(SELECT C.customer_id, C.first_name, C.last_name, CO.country, CI.city, SUM(P.amount)
AS total_amount_payment FROM payment AS P
INNER JOIN customer AS C ON C.customer_id =P.customer_id
INNER JOIN address AS A ON A.address_id = C.address_id
INNER JOIN city AS CI ON CI.city_id = A.city_id
INNER JOIN country CO ON CO.country_id = CI.country_id
WHERE CI.city IN (
SELECT CI.city FROM customer AS C
INNER JOIN address AS A ON A.address_id = C.address_id
INNER JOIN city AS CI ON CI.city_id = A.city_id
INNER JOIN country CO ON CO.country_id = CI.country_id
WHERE CO.country IN (
SELECT CO.country FROM customer AS C
INNER JOIN address AS A ON A.address_id = C.address_id
INNER JOIN city AS CI ON CI.city_id = A.city_id
INNER JOIN country CO ON CO.country_id = CI.country_id
GROUP BY CO.country
ORDER BY COUNT(C.customer_id) DESC
LIMIT 10)
GROUP BY CO.country,CI.city
ORDER BY Count(C.customer_id) DESC
LIMIT 10)
GROUP BY C.customer_id,CO.country, CI.city
ORDER BY total_amount_payment DESC
LIMIT 5) AS top_5_customers ON top_5_customers.country = CO.country
GROUP BY CO.country
ORDER BY top_customer_count DESC;
```



We start by selecting the country column from the country table and counting the number of distinct customer IDs (customer_id) for each country.
Then we join the customer, address, city, and country tables to link customer information with their respective cities and countries.
Using GROUP BY CO.country, we group the results by country.
COUNT(DISTINCT C.customer_id) counts the number of distinct customer IDs for each country.
Finally, we give the second column the alias all_customer_count for readability.

3:

```
SELECT outer_query.country,
      outer_query.all_customer_count,
      inner_query.average
FROM (
   SELECT CO.country,
         COUNT(DISTINCT C.customer_id) AS all_customer_count
   FROM customer AS C
   INNER JOIN address AS A ON C.address_id = A.address_id
   INNER JOIN city AS CI ON A.city_id = CI.city_id
   INNER JOIN country AS CO ON CI.country_id = CO.country_id
   GROUP BY CO.country
) AS outer_query
LEFT JOIN (
   SELECT AVG(P.amount) AS average,
         CI.country_id AS country_id
   FROM payment AS P
   INNER JOIN customer AS C ON P.customer_id = C.customer_id
   INNER JOIN address AS A ON C.address_id = A.address_id
   INNER JOIN city AS CI ON A.city_id = CI.city_id
   INNER JOIN country AS CO ON CI.country_id = CO.country_id
   WHERE CI.city IN (
      SELECT CI.city
      FROM customer AS C
      INNER JOIN address AS A ON C.address_id = A.address_id
      INNER JOIN city AS CI ON A.city_id = CI.city_id
      INNER JOIN country AS CO ON CI.country_id = CO.country_id
      GROUP BY CI.city
      ORDER BY COUNT(DISTINCT C.customer_id) DESC
      LIMIT 10
   )
   GROUP BY CI.country_id
   ORDER BY average DESC
   LIMIT 5
) AS inner_query ON outer_query.country = inner_query.country_id::text;
```



Subqueries are indispensable in this context. Firstly, they enable us to isolate subsets of data for analysis within a larger dataset. In the case of step 1, subqueries allow us to identify and calculate the average amount paid by the top 5 customers efficiently. Similarly, in step 2, they facilitate the segmentation of customers by country, enabling us to determine the number of customers in each country and then count how many of the top 5 customers reside in each country.

Subqueries prove to be invaluable when dealing with complex queries that require the manipulation of data subsets or when there's a need to derive insights based on the results of other queries. They enhance the modularity and readability of SQL statements by breaking down intricate problems into more manageable components. Moreover, subqueries offer versatility, as they can be employed in various parts of a SQL query,

including the SELECT, FROM, WHERE, and HAVING clauses, allowing for greater flexibility in query formulation.