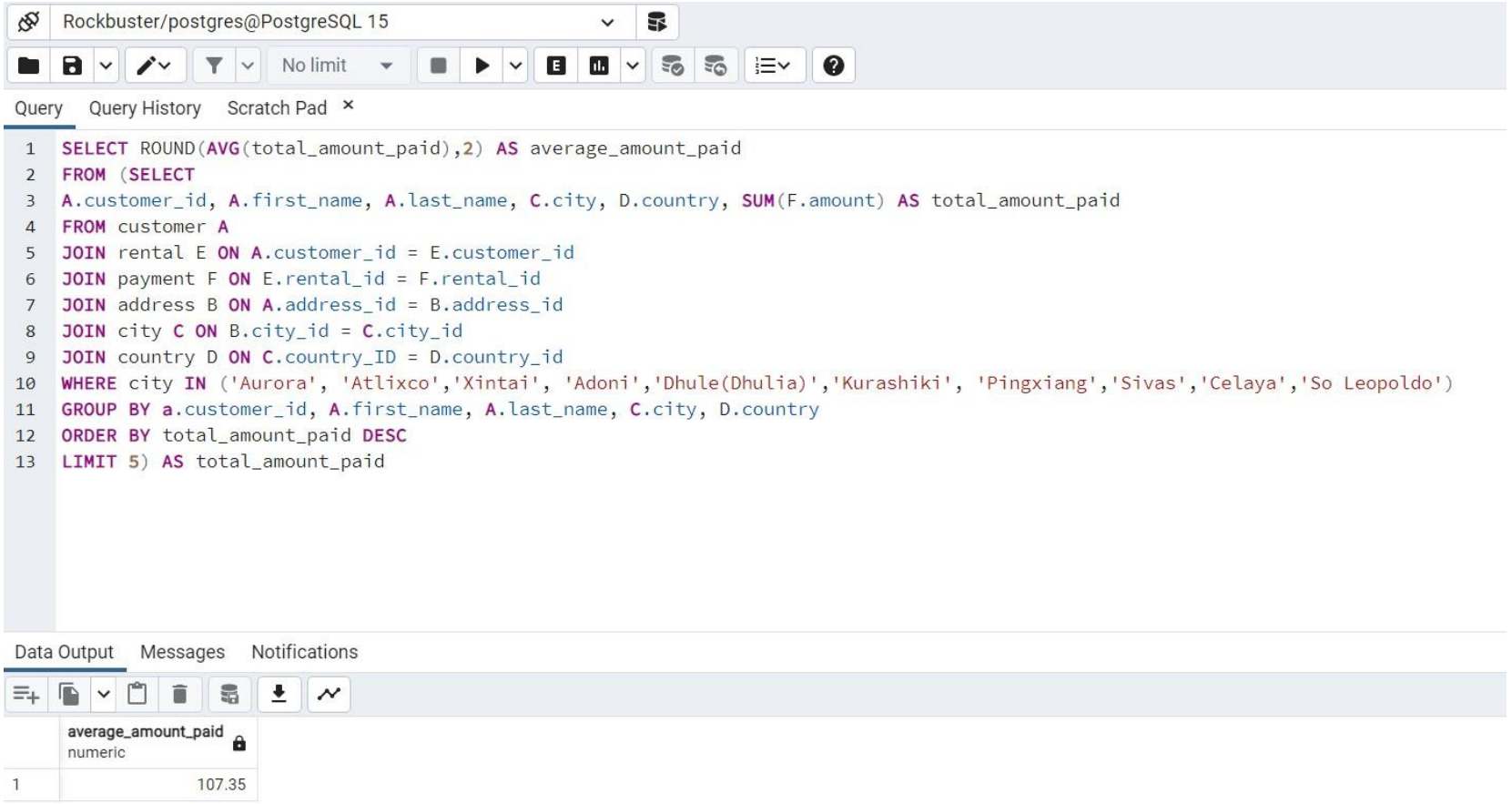
Step 1: Find the average amount paid by the top 5 customers.

1. **Copy the query you wrote in step 3 of the task from Exercise 3.7: Joining Tables of Data into the Query Tool. This will be your subquery, so give it an alias, "total_amount_paid," and add parentheses around it.**

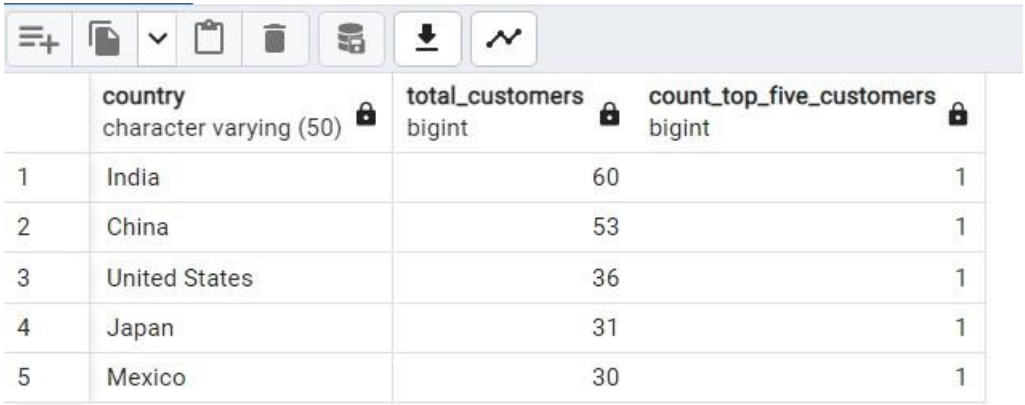


Step 2: Find out how many of the top 5 customers are based within each country.

Your final output should include 3 columns:

- “country”
- “all_customer_count” with the total number of customers in each country
- “top_customer_count” showing how many of the top 5 customers live in each country

```sql
SELECT DISTINCT(D.country),
COUNT (DISTINCT A.customer_id) as total_customers,
COUNT (DISTINCT D.country) as count_top_five_customers
FROM country D
INNER JOIN city C on D.country_id = C.country_id
INNER JOIN address B ON C.city_id = B.city_id
INNER JOIN customer A ON B.address_id = A.address_id
LEFT JOIN
(SELECT
A.customer_id, A.first_name, A.last_name, C.city, D.country, SUM(F.amount) AS total_amount_paid
FROM customer A
JOIN rental E ON A.customer_id = E.customer_id
JOIN payment F ON E.rental_id = F.rental_id
JOIN address B ON A.address_id = B.address_id
JOIN city C ON B.city_id = C.city_id
JOIN country D ON C.country_ID = D.country_id
WHERE country IN ('India', 'China','United States', 'Japan','Mexico','Brazil', 'Russian Federation','Philippines','Turkey','Indonesia')
GROUP BY a.customer_id, A.first_name, A.last_name, C.city, D.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS top_five_customers
ON D.country = top_five_customers.Country
GROUP BY D.country, top_five_customers
ORDER BY total_customers DESC
LIMIT 5;
```



	country character varying (50)	total_customers bigint	count_top_five_customers bigint
1	India	60	1
2	China	53	1
3	United States	36	1
4	Japan	31	1
5	Mexico	30	1

Step 3

Write 1 to 2 short paragraphs on the following:

- Do you think steps 1 and 2 could be done without using subqueries?
    1. Yes, I believe we could have completed steps 1 and 2 without using subqueries. We could have created multiple queries for each stage and then joined them together, but that would have required a significant amount of extra coding. The subqueries enabled us to build complex queries by simply conducting a query on the result of another query, making the syntax more straightforward.
- When do you think subqueries are useful?
    1. Subqueries can be used when data must be processed in multiple steps or when data is regularly modified. Using a subquery allows us to perform several queries in a single step and avoids the need to change the query each time the data is updated because it will run correctly every time.