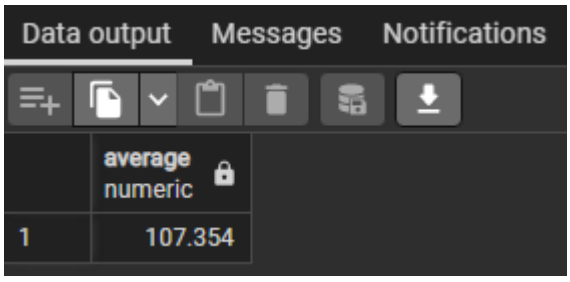
3.8 Performing Subqueries

Now your manager wants you to analyze the results of that query. The only catch is that revising your query could take quite some time, not to mention the risk of breaking it. Instead, you decide to use it as a subquery (or inner query) to answer the business questions listed below.

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

- 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.
- Write an outer statement to calculate the average amount paid.
- Add your subquery to the outer statement. It will go in either the SELECT, WHERE, or FROM clause. (Hint: When referring to the subquery in your outer statement, make sure to use the subquery’s alias, “total_amount_paid”.)
- If you've done everything correctly, pgAdmin 4 will require you to add an alias after the subquery. Go ahead and call it “average”.
- Copy-paste your queries and the final data output from pgAdmin 4 into your answers doument.

```
SELECT AVG(total_amount_paid.payment) AS average
FROM (
	SELECT SUM(pay.amount) AS payment
	FROM customer as cu
	INNER JOIN address AS ad ON cu.address_id = ad.address_id
	INNER JOIN city AS ci ON ad.city_id = ci.city_id
	INNER JOIN country AS co ON ci.country_id = co.country_id
	INNER JOIN payment AS pay ON cu.customer_id = pay.customer_id
	WHERE city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule(Dhulia)', 'Kurashiki',
			'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
	GROUP BY cu.customer_id
	ORDER BY payment DESC
	LIMIT 5
  ) AS total_amount_paid;
```

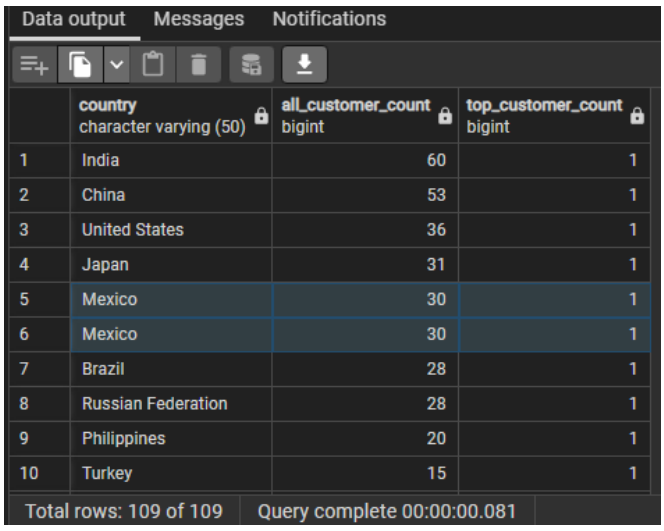


2. **Step 2: Find out how many of the top 5 customers are based within each country.**

   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

```
SELECT co.country,
        COUNT(DISTINCT cu.customer_id) AS all_customer_count,
        COUNT(DISTINCT co.country) AS top_customer_count
FROM country AS co
INNER JOIN city AS ci ON co.country_id = ci.country_id
INNER JOIN address AS ad ON ci.city_id = ad.city_id
INNER JOIN customer AS cu ON ad.address_id = cu.address_id
LEFT JOIN
    (
      SELECT cu.customer_id,
             cu.first_name,
             cu.last_name,
             ci.city,
             co.country,
             SUM(pay.amount) AS total_amount_paid
      FROM customer as cu
      INNER JOIN address AS ad ON cu.address_id = ad.address_id
      INNER JOIN city AS ci ON ad.city_id = ci.city_id
      INNER JOIN country AS co ON ci.country_id = co.country_id
      INNER JOIN payment AS pay ON cu.customer_id = pay.customer_id
      WHERE city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule(Dhulia)', 'Kurashiki',
                'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
      GROUP BY cu.customer_id, first_name, last_name, city, country
      ORDER BY total_amount_paid DESC
      LIMIT 5
    ) AS top_5_customers ON co.country = top_5_customers.country
GROUP BY co.country, top_5_customers
ORDER BY all_customer_count DESC;
```



Data output | Messages | Notifications

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	India	60	1
2	China	53	1
3	United States	36	1
4	Japan	31	1
5	Mexico	30	1
6	Mexico	30	1
7	Brazil	28	1
8	Russian Federation	28	1
9	Philippines	20	1
10	Turkey	15	1

Total rows: 109 of 109 | Query complete 00:00:00.081

3. **Step 3: Write 1 to 2 short paragraphs on the followingeach country.**

- Do you think steps 1 and 2 could be done without using subqueries?
  No. Step1 can not be done without subquery since it requires retrieving several tables plus performing operation (SUM) and then make average out of it (performing another operation). Additionally, as I discovered that aggregate function can not be nested (such as AVG(SUM()), subqueries shall be involved.
  Step 2 also needs subquery because we need to first filter out data (inside LEFT JOIN).

- When do you think subqueries are useful?

  It is useful especially to retrieve data that is constantly changing, in other words, automation and thus, avoiding hard-coded queries. From the task above, however, hard-coded query is still involved (WHERE city IN ('Aurora', 'Atlixco', …)). I think, if the city list is also changing, we should go one layer deeper on the subqueries.