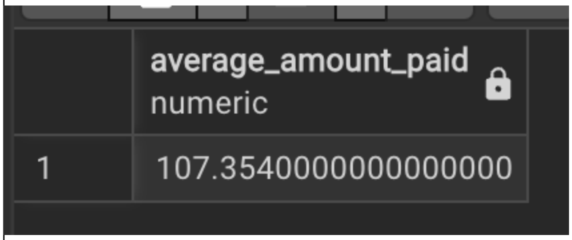
3.8: Performing Subqueries by Leo Garces

Answers 3.8:

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.
2. Write an outer statement to calculate the average amount paid.
3. 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".)
4. If you've done everything correctly, pgAdmin 4 will require you to add an alias after the subquery. Go ahead and call it "average".
5. Copy-paste your queries and the final data output from pgAdmin 4 into your answers document.

```
SELECT AVG (total_amount_paid) AS average_amount_paid
FROM
(SELECT B.customer_id,
B.first_name,
B.last_name,
D.city,
E.country,
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
('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)', 'Kurashiki',
'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
AND E.country IN
('India', 'China', 'United States', 'Japan', 'Mexico', 'Brazil',
'Russian Federation', 'Philippines', 'Turkey', 'Indonesia')
GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS average;
```



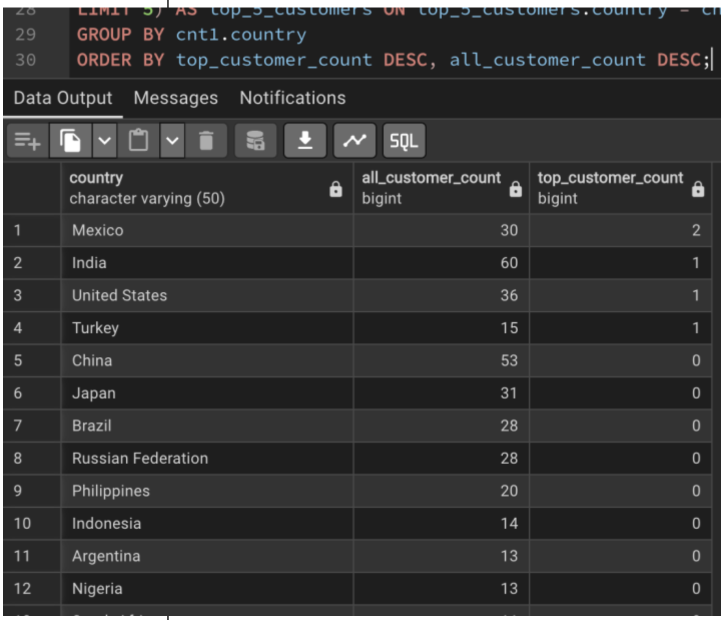
```
   average_amount_paid
   numeric
1  107.3540000000000000
```

Step 2: Find out how many of the top 5 customers you identified in step 1 are based within each country.

Your final output should include 3 columns:

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

```
SELECT cnt1.country,
COUNT(DISTINCT cust1.customer_id) AS all_customer_count,
COUNT(DISTINCT top_5_customers.customer_id) AS top_customer_count
FROM customer AS cust1
INNER JOIN address AS addr1 ON cust1.address_id = addr1.address_id
INNER JOIN city AS cty1 ON addr1.city_id = cty1.city_id
INNER JOIN country AS cnt1 ON cty1.country_id = cnt1.country_id
LEFT JOIN
(SELECT B.customer_id,
B.first_name,
B.last_name,
D.city,
E.country,
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
('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)', 'Kurashiki',
'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
AND E.country IN
('India', 'China', 'United States', 'Japan', 'Mexico', 'Brazil',
'Russian Federation', 'Philippines', 'Turkey', 'Indonesia')
GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS top_5_customers ON top_5_customers.country = cnt1.country
GROUP BY cnt1.country
ORDER BY top_customer_count DESC, all_customer_count DESC;
```



```
LIMIT 5) AS top_5_customers ON top_5_customers.country = cn
GROUP BY cnt1.country
ORDER BY top_customer_count DESC, all_customer_count DESC;
```

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	Mexico	30	2
2	India	60	1
3	United States	36	1
4	Turkey	15	1
5	China	53	0
6	Japan	31	0
7	Brazil	28	0
8	Russian Federation	28	0
9	Philippines	20	0
10	Indonesia	14	0
11	Argentina	13	0
12	Nigeria	13	0

Step 3:

1. Write 1 to 2 short paragraphs on the following:
    - Do you think steps 1 and 2 could be done without using subqueries?
    - When do you think subqueries are useful?

While it’s possible to write these queries using only filtering and joins, doing so can make the statement longer, harder to maintain, and is not considered best practice when working with data that is regularly updated. Subqueries are particularly useful when you need to run one query and use its result to influence the outcome of an outer query.