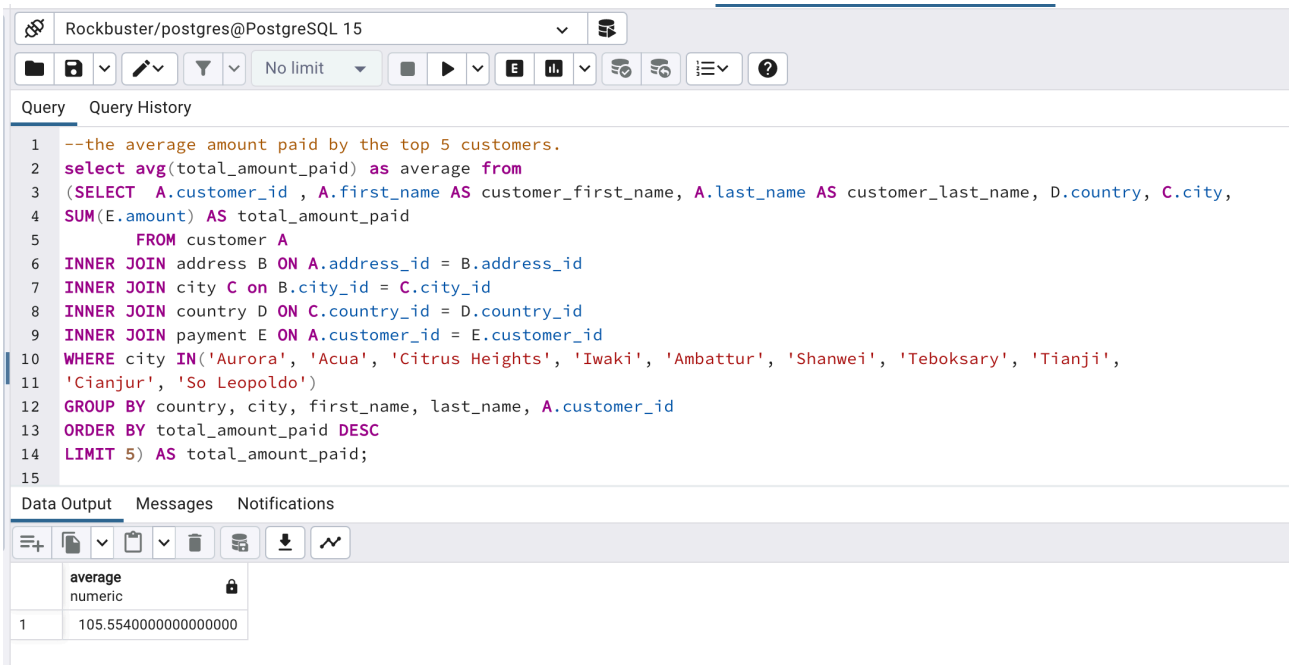
3.8: Performing Subqueries

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

```
--the average amount paid by the top 5 customers.
SELECT AVG(total_amount_paid) as average FROM
(SELECT  A.customer_id , A.first_name AS customer_first_name, A.last_name AS
customer_last_name, D.country, C.city,
SUM(E.amount) AS total_amount_paid
          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
INNER JOIN payment E ON A.customer_id = E.customer_id
WHERE city IN('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei',
'Teboksary', 'Tianji',
'Cianjur', 'So Leopoldo')
GROUP BY country, city, first_name, last_name, A.customer_id
ORDER BY total_amount_paid DESC
LIMIT 5) AS total_amount_paid;
```



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

```
--top 5 customers are based within each country.
SELECT D.country, COUNT(A.customer_id) AS All_customer_count, count(top_5_customers) AS
top_5_customers 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  A.customer_id , A.first_name AS customer_first_name, A.last_name AS
customer_last_name, D.country, C.city,
SUM(E.amount) AS total_amount_paid
            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
INNER JOIN payment E ON A.customer_id = E.customer_id
WHERE city IN('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'Teboksary',
'Tianji',
'Cianjur', 'So Leopoldo')
GROUP BY country, city, first_name, last_name, A.customer_id
ORDER BY total_amount_paid DESC
LIMIT 5) AS top_5_customers ON A.customer_id = top_5_customers.customer_id
GROUP BY D.country
HAVING COUNT(top_5_customers) > 0
ORDER BY COUNT(top_5_customers), COUNT(A.customer_id) DESC
```

Rockbuster/postgres@PostgreSQL 15

No limit

Query | Query History

```
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  A.customer_id , A.first_name AS customer_first_name, A.last_name AS customer_last_name, D.country, C.city,
SUM(E.amount) AS total_amount_paid
       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
INNER JOIN payment E ON A.customer_id = E.customer_id
WHERE city IN('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'Teboksary', 'Tianji',
'Cianjur', 'So Leopoldo')
GROUP BY country, city, first_name, last_name, A.customer_id
ORDER BY total_amount_paid DESC
LIMIT 5) AS top_5_customers ON A.customer_id = top_5_customers.customer_id
GROUP BY D.country
HAVING COUNT(top_5_customers) > 0
ORDER BY COUNT(top_5_customers), COUNT(A.customer_id) DESC
```

Data Output | Messages | Notifications

	country character varying	all_customer_count bigint	top_5_customers bigint
1	India	60	1
2	China	53	1
3	United States	36	1
4	Japan	31	1
5	Mexico	30	1

Total rows: 5 of 5 | Query complete 00:00:00.095 | Ln 20, Col 1

Step 3:

- Do you think steps 1 and 2 could be done without using subqueries?
- When do you think subqueries are useful?

- Yes, we can get this same information with JOIN query.
- When the data is continuously changing then using the subqueries will allow us to get the most recent data output from that query, whereas JOIN need to be rewritten every time the data is updated. Subqueries are useful when we need to perform operations in multiple steps. The subquery though, is useful when we have continuously changing data. when the result that we want requires more than one query and each subquery provides a subset of the table involved in the query.