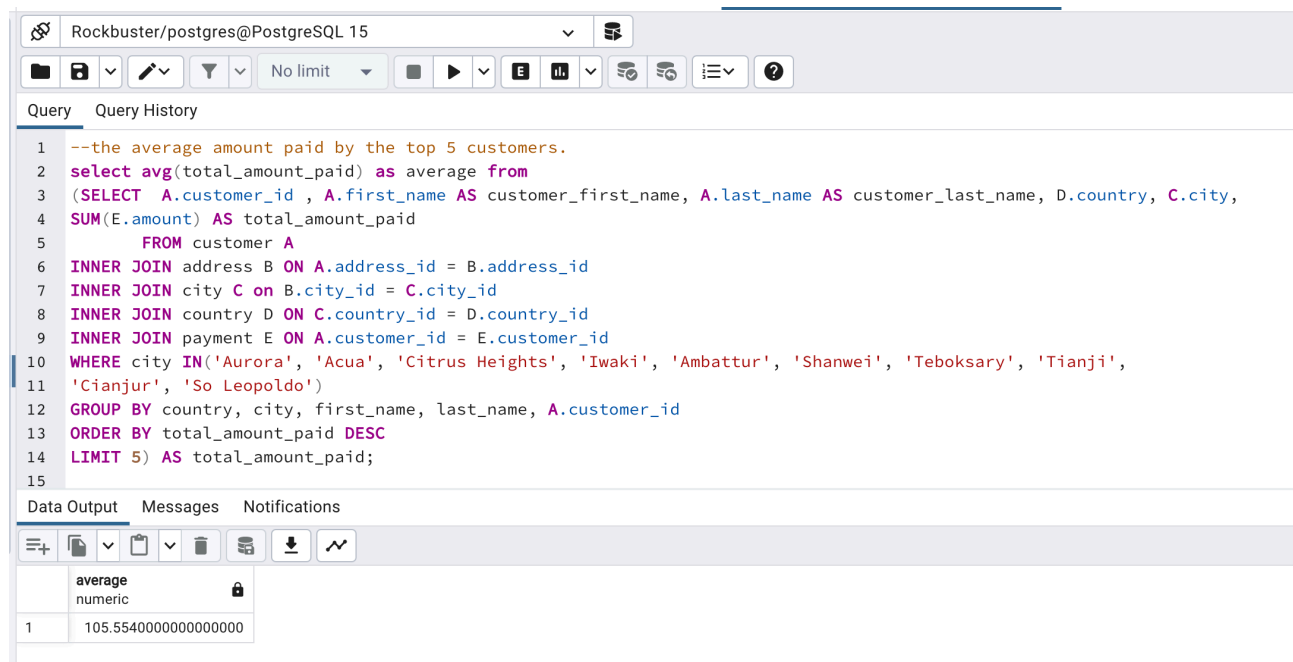


3.8: Performing Subqueries

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;
```



The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
1 --the average amount paid by the top 5 customers.
2 select avg(total_amount_paid) as average from
3 (SELECT A.customer_id , A.first_name AS customer_first_name, A.last_name AS customer_last_name, D.country, C.city,
4 SUM(E.amount) AS total_amount_paid
5 FROM customer A
6 INNER JOIN address B ON A.address_id = B.address_id
7 INNER JOIN city C on B.city_id = C.city_id
8 INNER JOIN country D ON C.country_id = D.country_id
9 INNER JOIN payment E ON A.customer_id = E.customer_id
10 WHERE city IN('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'Teboksary', 'Tianji',
11 'Cianjur', 'So Leopoldo')
12 GROUP BY country, city, first_name, last_name, A.customer_id
13 ORDER BY total_amount_paid DESC
14 LIMIT 5) AS total_amount_paid;
```

The query is executed, and the results are shown in the Data Output tab. The results are as follows:

average
105.5540000000000000

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
```

The screenshot shows a PostgreSQL query editor interface. The query is a complex SQL statement that joins multiple tables (customer, address, city, country, payment) and uses subqueries to identify the top 5 customers by total amount paid within each country. The results are displayed in a table with 5 rows and 3 columns: country, all_customer_count, and top_5_customers.

Query:

```
3 INNER JOIN city C on B.city_id = C.city_id
4 INNER JOIN country D ON C.country_id = D.country_id
5 LEFT JOIN (SELECT A.customer_id , A.first_name AS customer_first_name, A.last_name AS customer_last_name, D.country, C.city,
6 SUM(E.amount) AS total_amount_paid
7 FROM customer A
8 INNER JOIN address B ON A.address_id = B.address_id
9 INNER JOIN city C on B.city_id = C.city_id
10 INNER JOIN country D ON C.country_id = D.country_id
11 INNER JOIN payment E ON A.customer_id = E.customer_id
12 WHERE city IN('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'Teboksary', 'Tianji',
13 'Cianjur', 'So Leopoldo')
14 GROUP BY country, city, first_name, last_name, A.customer_id
15 ORDER BY total_amount_paid DESC
16 LIMIT 5) AS top_5_customers ON A.customer_id = top_5_customers.customer_id
17 GROUP BY D.country
18 HAVING COUNT(top_5_customers) > 0
19 ORDER BY COUNT(top_5_customers), COUNT(A.customer_id) DESC
20
21
```

Data Output:

	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.