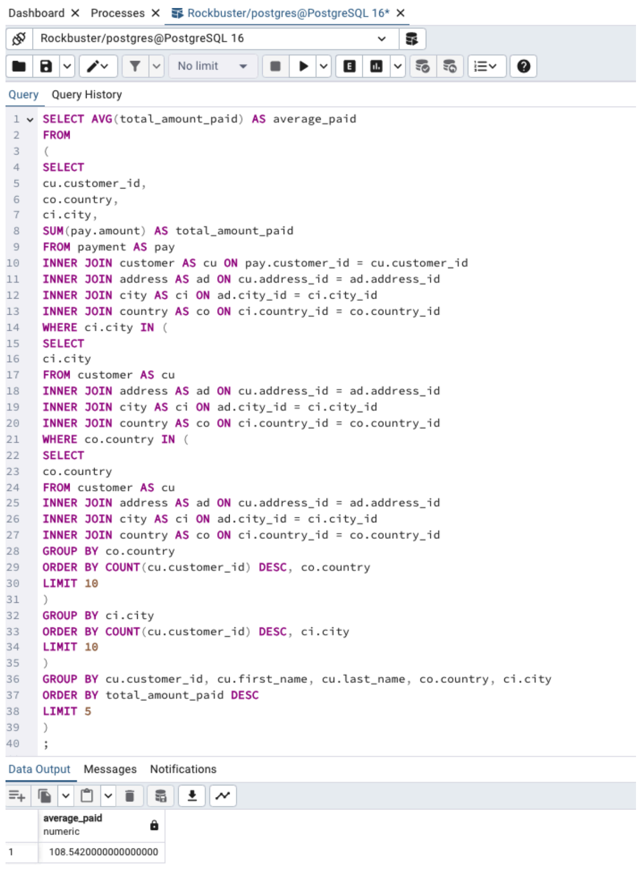
Kareemah Ashiru
Exercise 3.9

Step 1: Answer the business questions from steps 1 and 2 of task 3.8 using CTEs

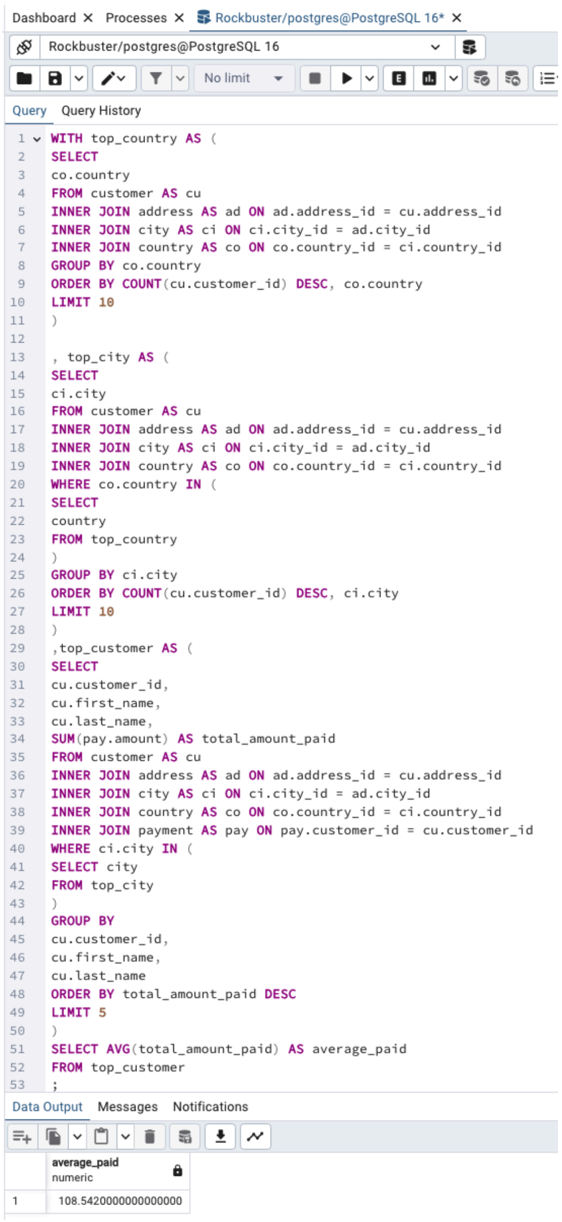
A. Step 1 from task 3.8: Subquery version



```
SELECT AVG(total_amount_paid) AS average_paid
FROM
(
SELECT
cu.customer_id,
co.country,
ci.city,
SUM(pay.amount) AS total_amount_paid
FROM payment AS pay
INNER JOIN customer AS cu ON pay.customer_id = cu.customer_id
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
WHERE ci.city IN (
SELECT
ci.city
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
WHERE co.country IN (
SELECT
co.country
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
GROUP BY co.country
ORDER BY COUNT(cu.customer_id) DESC, co.country
LIMIT 10
)
GROUP BY ci.city
ORDER BY COUNT(cu.customer_id) DESC, ci.city
LIMIT 10
)
GROUP BY cu.customer_id, cu.first_name, cu.last_name, co.country, ci.city
ORDER BY total_amount_paid DESC
LIMIT 5
)
;
```

	average_paid numeric
1	108.5420000000000000

a. Step 1 from task 3.8: CTE version

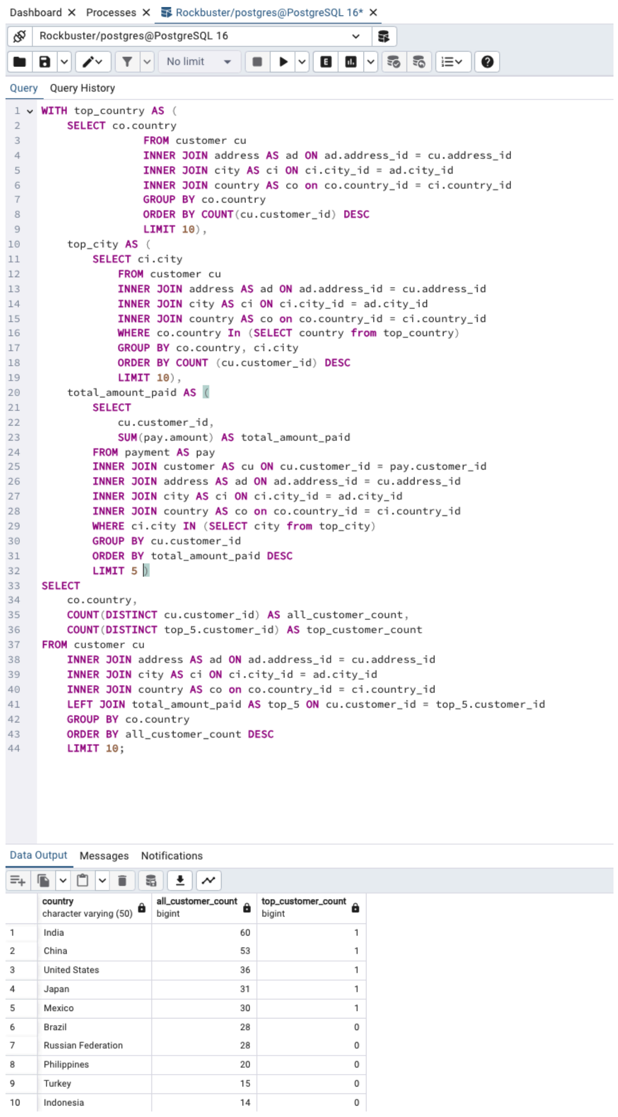


```sql
WITH top_country AS (
SELECT
co.country
FROM customer AS cu
INNER JOIN address AS ad ON ad.address_id = cu.address_id
INNER JOIN city AS ci ON ci.city_id = ad.city_id
INNER JOIN country AS co ON co.country_id = ci.country_id
GROUP BY co.country
ORDER BY COUNT(cu.customer_id) DESC, co.country
LIMIT 10
)

, top_city AS (
SELECT
ci.city
FROM customer AS cu
INNER JOIN address AS ad ON ad.address_id = cu.address_id
INNER JOIN city AS ci ON ci.city_id = ad.city_id
INNER JOIN country AS co ON co.country_id = ci.country_id
WHERE co.country IN (
SELECT
country
FROM top_country
)
GROUP BY ci.city
ORDER BY COUNT(cu.customer_id) DESC, ci.city
LIMIT 10
)
,top_customer AS (
SELECT
cu.customer_id,
cu.first_name,
cu.last_name,
SUM(pay.amount) AS total_amount_paid
FROM customer AS cu
INNER JOIN address AS ad ON ad.address_id = cu.address_id
INNER JOIN city AS ci ON ci.city_id = ad.city_id
INNER JOIN country AS co ON co.country_id = ci.country_id
INNER JOIN payment AS pay ON pay.customer_id = cu.customer_id
WHERE ci.city IN (
SELECT city
FROM top_city
)
GROUP BY
cu.customer_id,
cu.first_name,
cu.last_name
ORDER BY total_amount_paid DESC
LIMIT 5
)
SELECT AVG(total_amount_paid) AS average_paid
FROM top_customer
;
```

	average_paid numeric
1	108.5420000000000000

B. **Step 2 from task 3.8**



Dashboard ✕ Processes ✕ Rockbuster/postgres@PostgreSQL 16* ✕

Rockbuster/postgres@PostgreSQL 16

No limit

Query Query History

```
1  WITH top_country AS (
2      SELECT co.country
3                  FROM customer cu
4                  INNER JOIN address AS ad ON ad.address_id = cu.address_id
5                  INNER JOIN city AS ci ON ci.city_id = ad.city_id
6                  INNER JOIN country AS co on co.country_id = ci.country_id
7                  GROUP BY co.country
8                  ORDER BY COUNT(cu.customer_id) DESC
9                  LIMIT 10),
10     top_city AS (
11         SELECT ci.city
12             FROM customer cu
13             INNER JOIN address AS ad ON ad.address_id = cu.address_id
14             INNER JOIN city AS ci ON ci.city_id = ad.city_id
15             INNER JOIN country AS co on co.country_id = ci.country_id
16             WHERE co.country In (SELECT country from top_country)
17             GROUP BY co.country, ci.city
18             ORDER BY COUNT (cu.customer_id) DESC
19             LIMIT 10),
20     total_amount_paid AS (
21         SELECT
22             cu.customer_id,
23             SUM(pay.amount) AS total_amount_paid
24         FROM payment AS pay
25         INNER JOIN customer AS cu ON cu.customer_id = pay.customer_id
26         INNER JOIN address AS ad ON ad.address_id = cu.address_id
27         INNER JOIN city AS ci ON ci.city_id = ad.city_id
28         INNER JOIN country AS co on co.country_id = ci.country_id
29         WHERE ci.city IN (SELECT city from top_city)
30         GROUP BY cu.customer_id
31         ORDER BY total_amount_paid DESC
32         LIMIT 5 )
33 SELECT
34     co.country,
35     COUNT(DISTINCT cu.customer_id) AS all_customer_count,
36     COUNT(DISTINCT top_5.customer_id) AS top_customer_count
37 FROM customer cu
38     INNER JOIN address AS ad ON ad.address_id = cu.address_id
39     INNER JOIN city AS ci ON ci.city_id = ad.city_id
40     INNER JOIN country AS co on co.country_id = ci.country_id
41     LEFT JOIN total_amount_paid AS top_5 ON cu.customer_id = top_5.customer_id
42     GROUP BY co.country
43     ORDER BY all_customer_count DESC
44     LIMIT 10;
```

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	Brazil	28	0
7	Russian Federation	28	0
8	Philippines	20	0
9	Turkey	15	0
10	Indonesia	14	0

C. Explanation

a. First I added With top_country AS (: to help the query understand that whenever top_country is written, it consists of the queries from #2 - #11.

b. I did the same for top_city and top_customer.

c. I wrapped the whole query input with SELECT AVG(total_amount_paid) AS average_paid FROM top_customer at the end instead of at the beginning like I did for the query.

Step 2: Compare the performance of your CTEs and subqueries.

A. On one hand I like that the subquery version is shorter on the other hand the CTE version is an easier reference for anyone to understand. This is because what the query layouts like *top_country, top_city*, and *top_customer* consist of are all stated in the beginning. It reduces repetition when using the FROM command. Overall, I believe that the CTE version will perform better.

B.

a.

Query 1	**CTE**	**Subquery**
Estimated Cost	164.60	165.82
Time	131	57

b.

Query 2	**CTE**	**Subquery**
Estimated Cost	266.84	266.84
Time	70	67

C. I'm surprised by the fact that the subquery for both Query 1 and 2 cost about the same but with less time. I guess this concludes that both approaches can be efficient depending on the company's needs.

Step 3: Challenges faced when replacing subqueries with CTEs.

A. It's almost like writing the query in the reverse when replacing subqueries with CTEs which feels weird. However, I find it easier to follow than the subqueries.

B. Based on task 3.8 and 3.9, I find that CTE queries might actually be longer to write than subqueries and take more time.