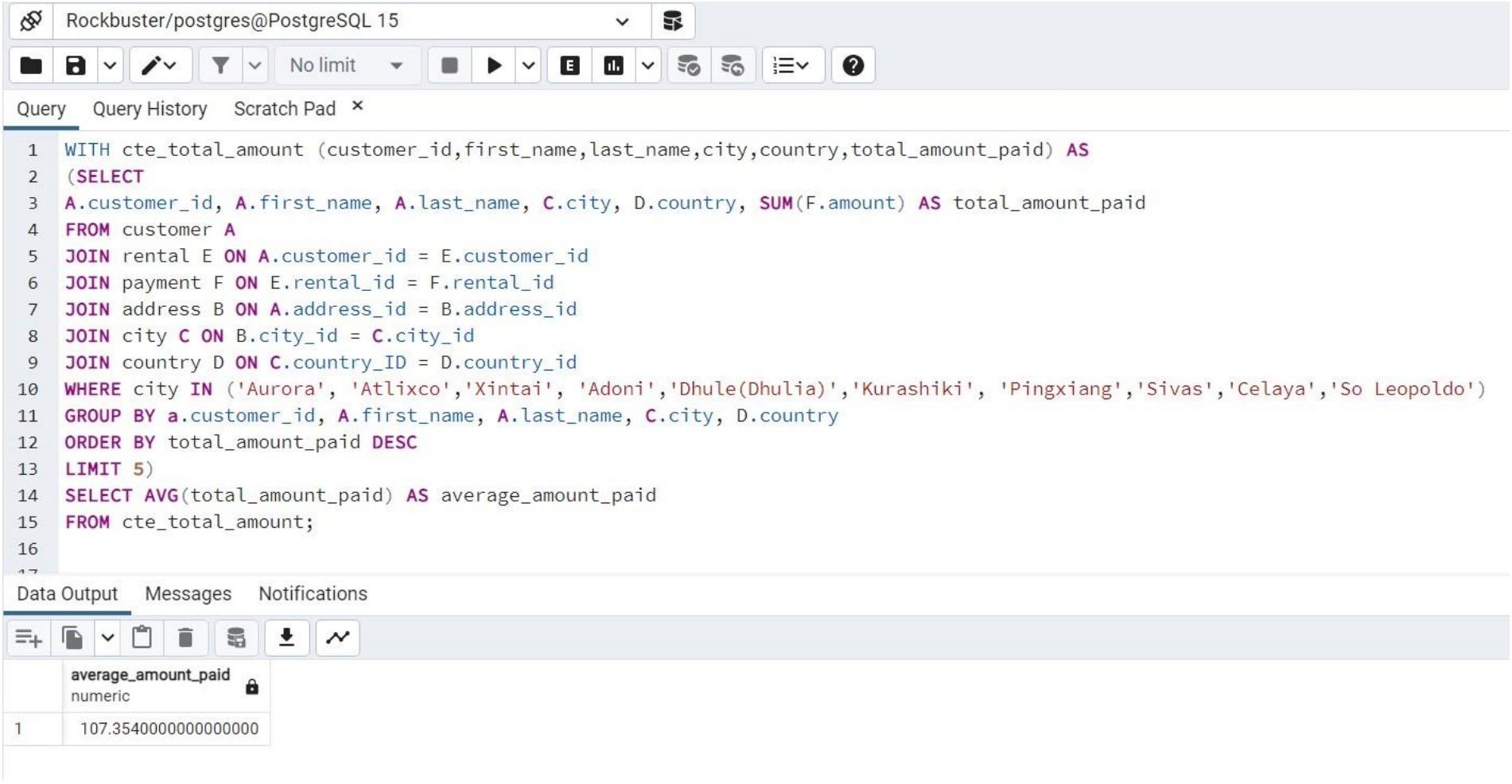
Step 1: Answer the business questions from step 1 and 2 of task 3.8 using CTE

1. **Rewrite your queries from steps 1 and 2 of task 3.8 as CTEs.**



```sql
WITH total_customer_count_cte (customer_id, first_name,last_name, city, country, Total_amount_paid) AS
(SELECT A.customer_id, A.first_name, A.last_name, C.city, D.country, SUM(F.amount) AS Total_amount_paid
FROM customer A
INNER JOIN rental E ON A.customer_id = E.customer_id
INNER JOIN payment F ON E.rental_id = F.rental_id
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
WHERE city IN ('Aurora', 'Atlixco','Xintai', 'Adoni','Dhule(Dhulia)','Kurashiki', 'Pingxiang','Sivas','Celaya','So Leopoldo')
GROUP BY A.customer_id, A.first_name, A.last_name, C.city, D.country
ORDER BY Total_amount_paid DESC LIMIT 5),
customer_count_cte AS (SELECT D.country,COUNT(DISTINCT A.customer_id) AS count_customers_cte, COUNT (DISTINCT D.country) AS top_costumers_count
FROM country D
INNER JOIN city C on D.country_id = C.country_id
INNER JOIN address B ON C.city_id = B.city_id
INNER JOIN customer A ON B.address_id = A.address_id
GROUP BY D.country)
SELECT D.country, COUNT(DISTINCT A.customer_id) AS count_customers
FROM country D
INNER JOIN city C on D.country_id = C.country_id
INNER JOIN address B ON C.city_id = B.city_id
INNER JOIN customer A ON B.address_id = A.address_id
LEFT JOIN total_customer_count_cte ON D.country=total_customer_count_cte.country
GROUP BY D.country
ORDER BY count_customers DESC LIMIT 5
```

Data Output | Messages | Notifications

	country character varying (50)	count_customers bigint
1	India	60
2	China	53
3	United States	36
4	Japan	31
5	Mexico	30

2. **Write 2 to 3 sentences explaining how you approached this step, for example, what you did first, second, and so on**
   I began by dividing the inner and outer question from exercise 3.8 for the first query. Then I wrote the WITH statement, and last I calculated the average using the new CTE table, thereby replacing the outer query. I followed the same method for the second query, but this time I needed to make two CTEs instead of one: one for the customers (count_customers) and the other for the top customers (top_costumers_count).

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

1. **Which approach do you think will perform better and why?**
   a. I believe that a CTE works better than a subquery since it is more readable and can be reused, allowing us to write less code when compared to a subquery.
2. **Compare the costs of all the queries by creating query plans for each one.**

SUBQUERY	cost=514.19…514.20 rows=1 width=32, total rows: 26, query complete 00:00:00.118
CTE	cost=514.19…514.20 rows=1 width=32, total rows: 26, query complete 00:00:00.042

3. **The EXPLAIN command gives you an *estimated* cost. To find out the actual speed of your queries, run them in pgAdmin 4. After each query has been run, a pop-up window will display its speed in milliseconds.**

Rockbuster/postgres@PostgreSQL 15

No limit

Query | Query History | Scratch Pad

```
EXPLAIN SELECT ROUND(AVG(total_amount_paid),2) AS average_amount_paid
FROM (SELECT
A.customer_id, A.first_name, A.last_name, C.city, D.country, SUM(F.amount) AS total_amount_paid
FROM customer A
JOIN rental E ON A.customer_id = E.customer_id
JOIN payment F ON E.rental_id = F.rental_id
JOIN address B ON A.address_id = B.address_id
JOIN city C ON B.city_id = C.city_id
JOIN country D ON C.country_ID = D.country_id
WHERE city IN ('Aurora', 'Atlixco','Xintai', 'Adoni','Dhule(Dhulia)','Kurashiki', 'Pingxiang','Sivas',
GROUP BY a.customer_id, A.first_name, A.last_name, C.city, D.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS total_amount_paid
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Aggregate (cost=514.19..514.20 rows=1 width=32)
2	-> Limit (cost=514.11..514.12 rows=5 width=67)
3	-> Sort (cost=514.11..514.72 rows=243 width=67)
4	Sort Key: (sum(f.amount)) DESC
5	-> HashAggregate (cost=507.04..510.07 rows=243 width=67)
6	Group Key: a.customer_id, c.city, d.country
7	-> Hash Join (cost=37.52..504.61 rows=243 width=41)
8	Hash Cond: (c.country_id = d.country_id)
9	-> Nested Loop (cost=34.07..500.49 rows=243 width=34)
10	-> Hash Join (cost=33.79..407.07 rows=267 width=32)
11	Hash Cond: (e.customer_id = a.customer_id)
12	-> Seq Scan on rental e (cost=0.00..310.44 rows=16044 width=6)
13	-> Hash (cost=33.66..33.66 rows=10 width=28)
14	-> Nested Loop (cost=14.43..33.66 rows=10 width=28)
15	-> Hash Join (cost=14.15..29.77 rows=10 width=15)
16	Hash Cond: (b.city_id = c.city_id)
17	-> Seq Scan on address b (cost=0.00..14.03 rows=603 width=6)
18	-> Hash (cost=14.03..14.03 rows=10 width=15)
19	-> Seq Scan on city c (cost=0.03..14.03 rows=10 width=15)
20	Filter: ((city)::text = ANY ('{Aurora,Atlixco,Xintai,Adoni,Dhule(Dhulia),Kurashiki,Pingxiang,Sivas,Celaya,"So Leopo
21	-> Index Scan using idx_fk_address_id on customer a (cost=0.28..0.38 rows=1 width=19)
22	Index Cond: (address_id = b.address_id)
23	-> Index Scan using idx_fk_rental_id on payment f (cost=0.29..0.34 rows=1 width=10)
24	Index Cond: (rental_id = e.rental_id)
25	-> Hash (cost=2.09..2.09 rows=109 width=13)
26	-> Seq Scan on country d (cost=0.00..2.09 rows=109 width=13)

Query | Query History | Scratch Pad

```
1  EXPLAIN WITH cte_total_amount (customer_id,first_name,last_name,city,country,total_amount_paid) AS
2  (SELECT A.customer_id, A.first_name, A.last_name, C.city, D.country, SUM(F.amount) AS total_amount_pai
3  FROM customer A
4  JOIN rental E ON A.customer_id = E.customer_id
5  JOIN payment F ON E.rental_id = F.rental_id
6  JOIN address B ON A.address_id = B.address_id
7  JOIN city C ON B.city_id = C.city_id
8  JOIN country D ON C.country_ID = D.country_id
9  WHERE city IN ('Aurora', 'Atlixco','Xintai', 'Adoni','Dhule(Dhulia)','Kurashiki', 'Pingxiang','Sivas',
10 GROUP BY a.customer_id, A.first_name, A.last_name, C.city, D.country
11 ORDER BY total_amount_paid DESC
12 LIMIT 5)
13 SELECT AVG(total_amount_paid) AS average_amount_paid
14 FROM cte_total_amount;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Aggregate (cost=514.19..514.20 rows=1 width=32)
2	-> Limit (cost=514.11..514.12 rows=5 width=67)
3	-> Sort (cost=514.11..514.72 rows=243 width=67)
4	Sort Key: (sum(f.amount)) DESC
5	-> HashAggregate (cost=507.04..510.07 rows=243 width=67)
6	Group Key: a.customer_id, c.city, d.country
7	-> Hash Join (cost=37.52..504.61 rows=243 width=41)
8	Hash Cond: (c.country_id = d.country_id)
9	-> Nested Loop (cost=34.07..500.49 rows=243 width=34)
10	-> Hash Join (cost=33.79..407.07 rows=267 width=32)
11	Hash Cond: (e.customer_id = a.customer_id)
12	-> Seq Scan on rental e (cost=0.00..310.44 rows=16044 width=6)
13	-> Hash (cost=33.66..33.66 rows=10 width=28)
14	-> Nested Loop (cost=14.43..33.66 rows=10 width=28)
15	-> Hash Join (cost=14.15..29.77 rows=10 width=15)
16	Hash Cond: (b.city_id = c.city_id)
17	-> Seq Scan on address b (cost=0.00..14.03 rows=603 width=6)
18	-> Hash (cost=14.03..14.03 rows=10 width=15)
19	-> Seq Scan on city c (cost=0.03..14.03 rows=10 width=15)
20	Filter: ((city)::text = ANY ('{Aurora,Atlixco,Xintai,Adoni,Dhule(Dhulia),Kurashiki,Pingxiang,Sivas,Celaya,"So Leopoldo"
21	-> Index Scan using idx_fk_address_id on customer a (cost=0.28..0.38 rows=1 width=19)
22	Index Cond: (address_id = b.address_id)
23	-> Index Scan using idx_fk_rental_id on payment f (cost=0.29..0.34 rows=1 width=10)
24	Index Cond: (rental_id = e.rental_id)
25	-> Hash (cost=2.09..2.09 rows=109 width=13)
26	-> Seq Scan on country d (cost=0.00..2.09 rows=109 width=13)

4. **Did the results surprise you? Write a few sentences to explain your answer.**
    a. No, the results did not surprise me: the CTE was significantly faster than the subquery in terms of milliseconds (42 versus 118), exactly as I expected.

Step 3:

Write 1 to 2 paragraphs on the challenges you faced when replacing your subqueries with CTEs.

I found it relatively simple to replace the first subquery (average amount paid) with a CTE, however the second was far more difficult and required far more than a try. In instance, I didn't realize at first that I needed to create two CTEs: one for customers and one for top customers. The answer appeared as soon as I understood it, which was a big satisfaction.