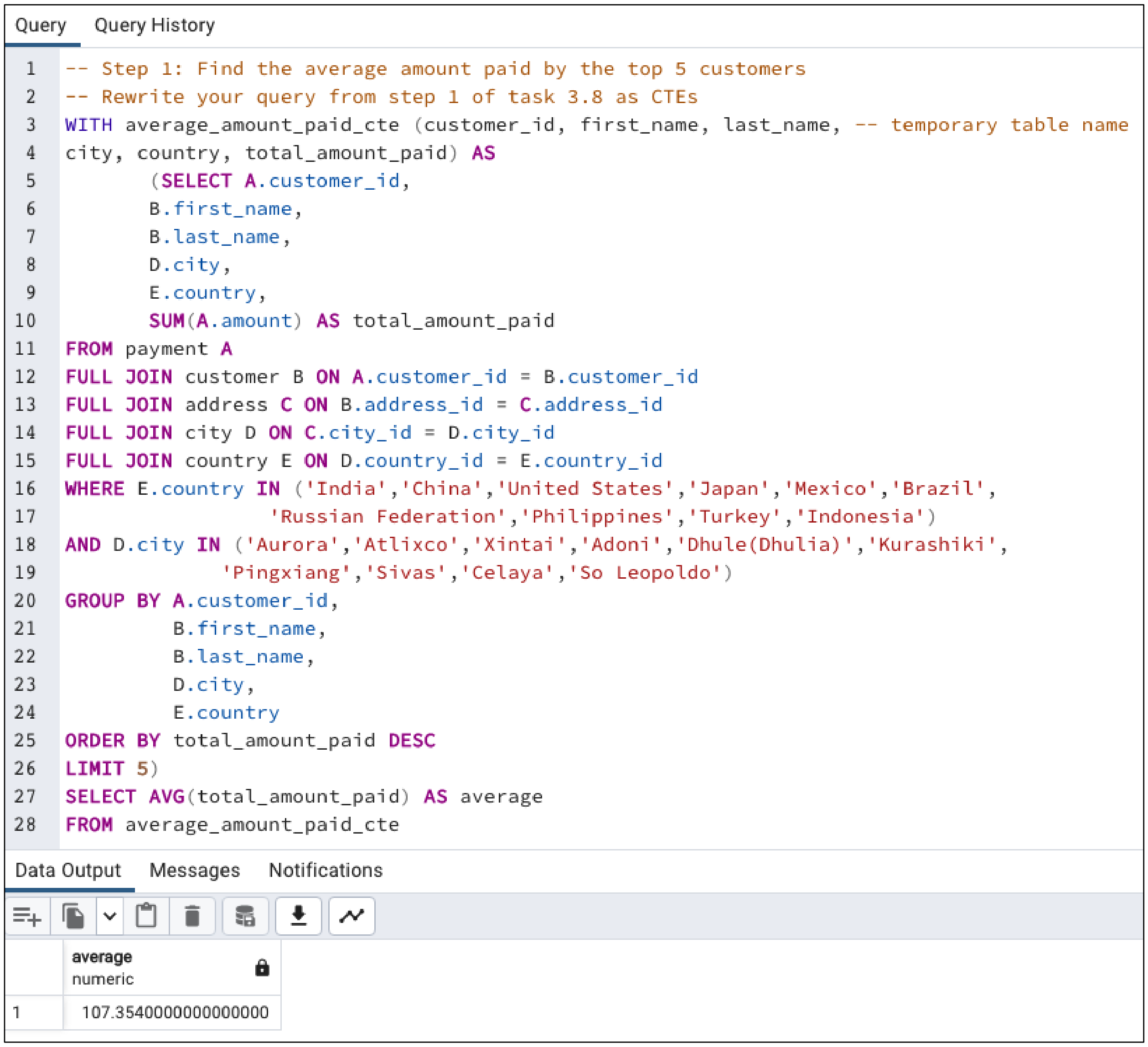
Step 1: Answer the business questions from step 1 and 2 of task 3.8 using CTEs

1. **Rewrite your queries from steps 1 and 2 of task 3.8 as CTEs.**
2. **Copy-paste your CTEs and their outputs into your answers document.**

3.8 Step 1



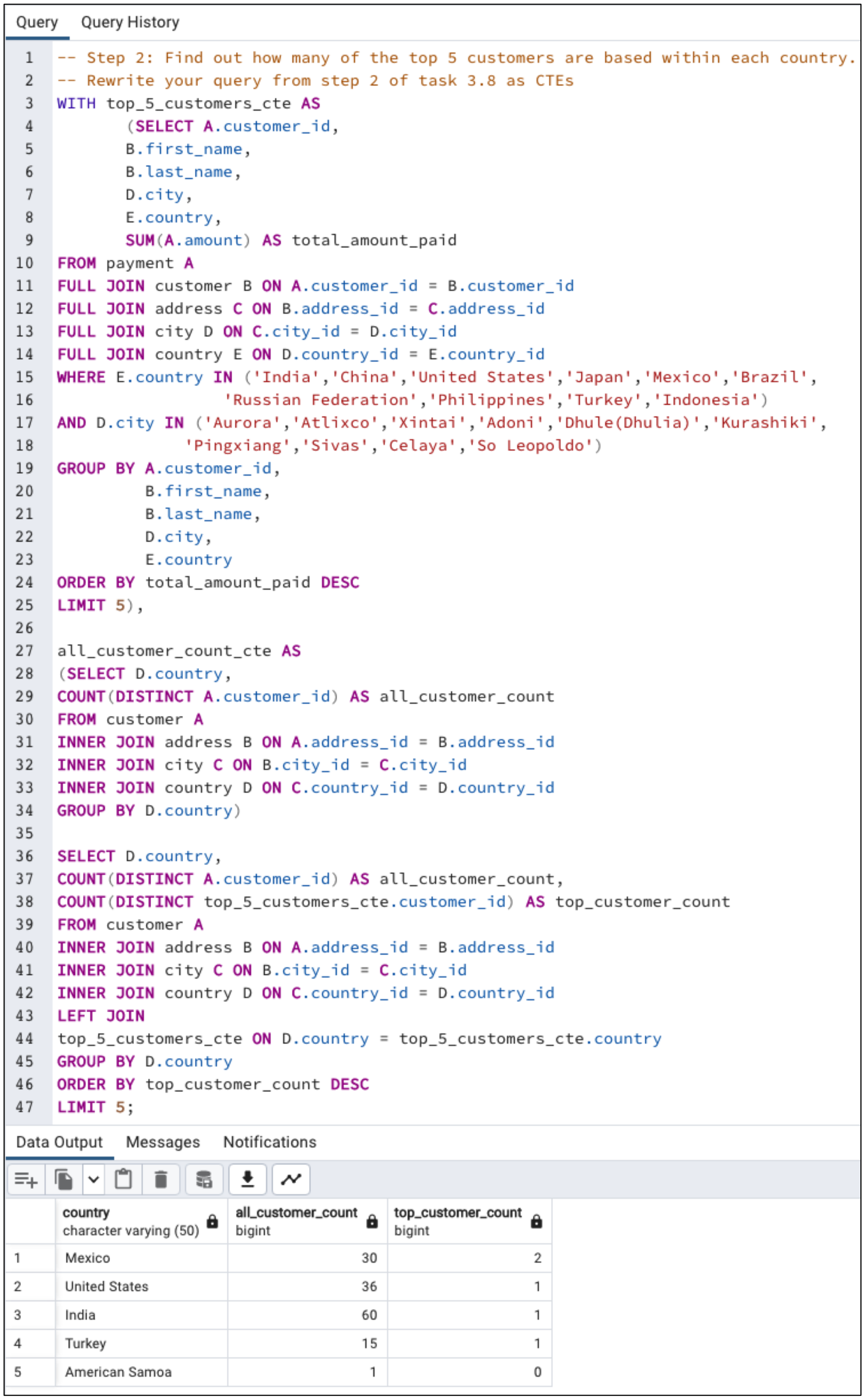
Query Query History

```sql
-- Step 1: Find the average amount paid by the top 5 customers
-- Rewrite your query from step 1 of task 3.8 as CTEs
WITH average_amount_paid_cte (customer_id, first_name, last_name, -- temporary table name
city, country, total_amount_paid) AS
       (SELECT A.customer_id,
       B.first_name,
       B.last_name,
       D.city,
       E.country,
       SUM(A.amount) AS total_amount_paid
FROM payment A
FULL JOIN customer B ON A.customer_id = B.customer_id
FULL JOIN address C ON B.address_id = C.address_id
FULL JOIN city D ON C.city_id = D.city_id
FULL JOIN country E ON D.country_id = E.country_id
WHERE E.country IN ('India','China','United States','Japan','Mexico','Brazil',
                  'Russian Federation','Philippines','Turkey','Indonesia')
AND D.city IN ('Aurora','Atlixco','Xintai','Adoni','Dhule(Dhulia)','Kurashiki',
              'Pingxiang','Sivas','Celaya','So Leopoldo')
GROUP BY A.customer_id,
         B.first_name,
         B.last_name,
         D.city,
         E.country
ORDER BY total_amount_paid DESC
LIMIT 5)
SELECT AVG(total_amount_paid) AS average
FROM average_amount_paid_cte
```

Data Output Messages Notifications

```
   average
   numeric
1  107.3540000000000000
```

3.8 Step 2



```sql
-- Step 2: Find out how many of the top 5 customers are based within each country.
-- Rewrite your query from step 2 of task 3.8 as CTEs
WITH top_5_customers_cte AS
       (SELECT A.customer_id,
       B.first_name,
       B.last_name,
       D.city,
       E.country,
       SUM(A.amount) AS total_amount_paid
FROM payment A
FULL JOIN customer B ON A.customer_id = B.customer_id
FULL JOIN address C ON B.address_id = C.address_id
FULL JOIN city D ON C.city_id = D.city_id
FULL JOIN country E ON D.country_id = E.country_id
WHERE E.country IN ('India','China','United States','Japan','Mexico','Brazil',
                 'Russian Federation','Philippines','Turkey','Indonesia')
AND D.city IN ('Aurora','Atlixco','Xintai','Adoni','Dhule(Dhulia)','Kurashiki',
             'Pingxiang','Sivas','Celaya','So Leopoldo')
GROUP BY A.customer_id,
         B.first_name,
         B.last_name,
         D.city,
         E.country
ORDER BY total_amount_paid DESC
LIMIT 5),

all_customer_count_cte AS
(SELECT D.country,
COUNT(DISTINCT A.customer_id) AS all_customer_count
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
GROUP BY D.country)

SELECT D.country,
COUNT(DISTINCT A.customer_id) AS all_customer_count,
COUNT(DISTINCT top_5_customers_cte.customer_id) AS top_customer_count
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
top_5_customers_cte ON D.country = top_5_customers_cte.country
GROUP BY D.country
ORDER BY top_customer_count DESC
LIMIT 5;
```

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	Mexico	30	2
2	United States	36	1
3	India	60	1
4	Turkey	15	1
5	American Samoa	1	0

3. **Write 2 to 3 sentences explaining how you approached this step, for example, what you did first, second, and so on.**

   <u>3.8 Step 1</u>

   The manager at Rockbuster wants to find the average amount paid by the top 5 customers. The first step is to find out which tables in the Rockbuster database hold the data we need. Based on the ERD, columns “amount” and “country” are stored in the “payment” and “country” tables. However, we need to join the “payment” table with the “customer” table, the “customer” table with the “address” table, the “address” table with the “city” table and the “city” table with the “country” table.

   For the subquery, I generated a list of customer IDs, first and last name, cities, countries, and amounts. Since it is not possible to the join the “payment” table directly with the “country” table. We used the full join to merge the “payment” table with the “customer” table, the “customer” table with the “address” table, the “address” table with the “city” table and the “city” table with the “country” table.

   Next, we want to create the CTE with the data input from the subquery. To define the CTE, we will use the WITH clause and give the CTE an alias, for this exercise, we will label it “average_amount_paid_cte” and list all the column names from the subquery along with the AS syntax followed by the query to produce the CTE in parentheses.

   Now that we have our CTE, we will write a main statement to query it. We selected the total_amount_paid column and included the average (AVG) as an aggregate function. We also given this column the alias “average.” The name of the CTE (average_amount_paid_cte) comes after the FROM clause because that is where we want to pull the data from.

   <u>3.8 Step 2</u>

   We created a CTE with data input from the subquery and labeled it “top_5_customers_cte” which included all the column names from the subquery. Now that we have out CTE, we will write a main statement to query it. Columns all_customer_count and top_customer_count from exercise 3.8 were also included in order to join the CTE and pull the correct tables to find out how many of the top 5 customer are based within each country.

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

1. **Which approach do you think will perform better and why?**

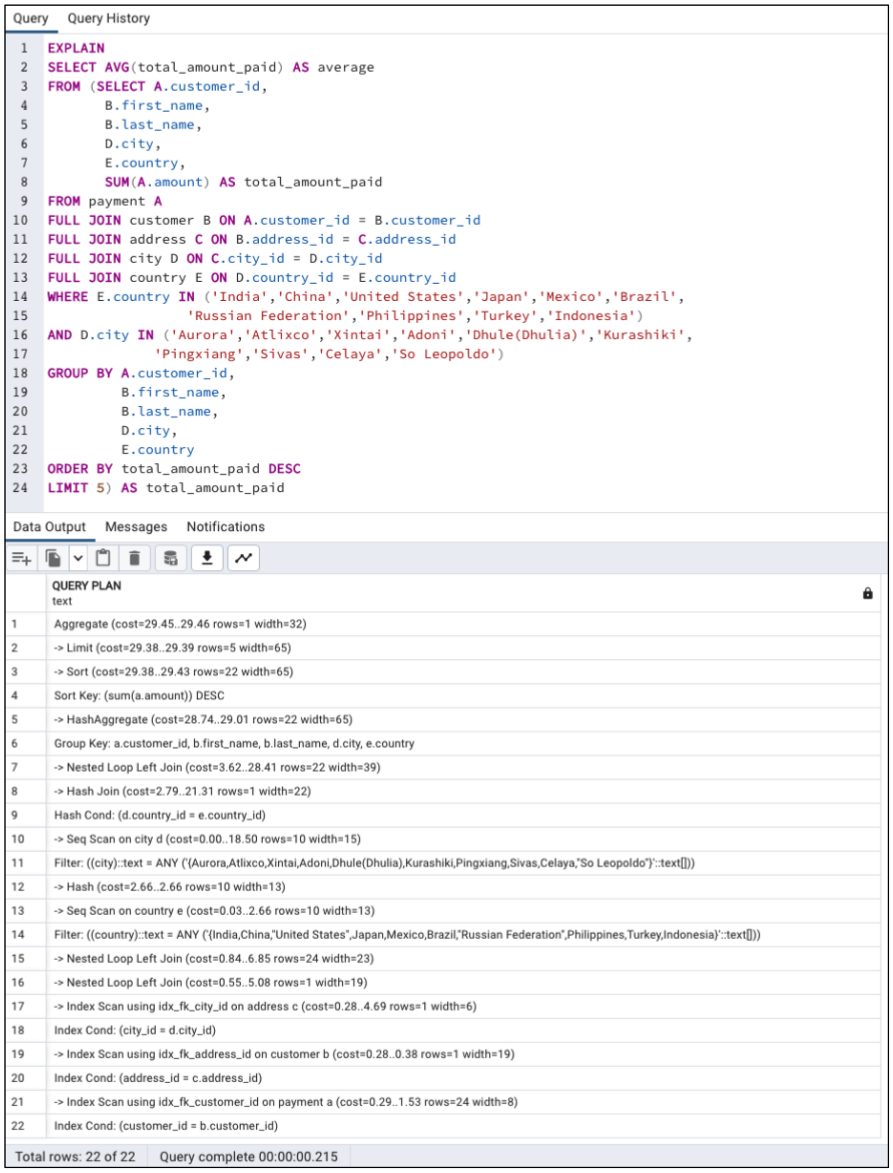
   CTE due to its ability to enhance readability and avoid repetition of subqueries. It is useable in ad-hoc-queries and has no unexpected side effects.

2. **Compare the costs of all the queries by creating query plans for each one.**

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.**

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

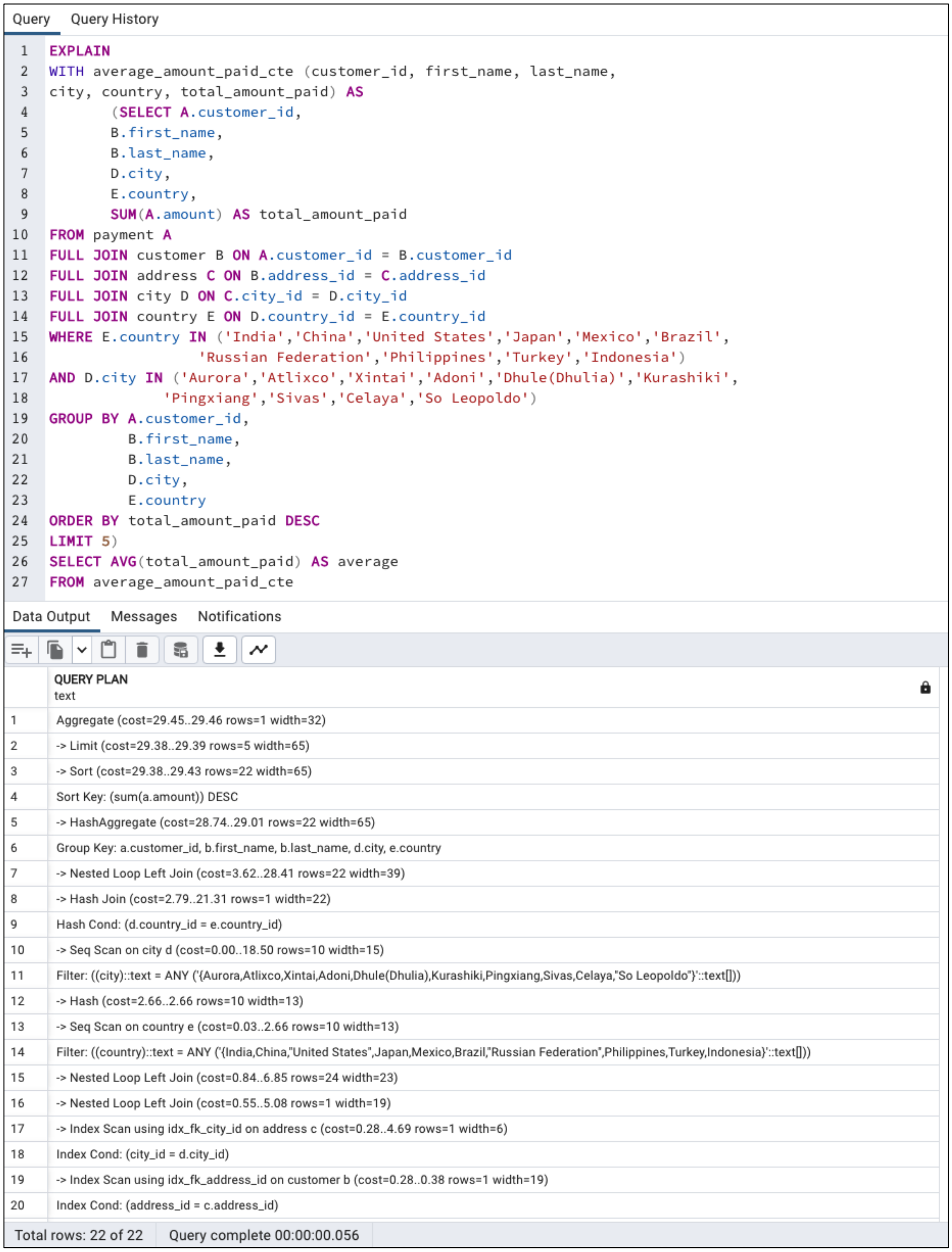
<u>Subquery</u>



```sql
EXPLAIN
SELECT AVG(total_amount_paid) AS average
FROM (SELECT A.customer_id,
       B.first_name,
       B.last_name,
       D.city,
       E.country,
       SUM(A.amount) AS total_amount_paid
FROM payment A
FULL JOIN customer B ON A.customer_id = B.customer_id
FULL JOIN address C ON B.address_id = C.address_id
FULL JOIN city D ON C.city_id = D.city_id
FULL JOIN country E ON D.country_id = E.country_id
WHERE E.country IN ('India','China','United States','Japan','Mexico','Brazil',
                 'Russian Federation','Philippines','Turkey','Indonesia')
AND D.city IN ('Aurora','Atlixco','Xintai','Adoni','Dhule(Dhulia)','Kurashiki',
             'Pingxiang','Sivas','Celaya','So Leopoldo')
GROUP BY A.customer_id,
         B.first_name,
         B.last_name,
         D.city,
         E.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS total_amount_paid
```

```
QUERY PLAN
text
1   Aggregate (cost=29.45..29.46 rows=1 width=32)
2   -> Limit (cost=29.38..29.39 rows=5 width=65)
3   -> Sort (cost=29.38..29.43 rows=22 width=65)
4   Sort Key: (sum(a.amount)) DESC
5   -> HashAggregate (cost=28.74..29.01 rows=22 width=65)
6   Group Key: a.customer_id, b.first_name, b.last_name, d.city, e.country
7   -> Nested Loop Left Join (cost=3.62..28.41 rows=22 width=39)
8   -> Hash Join (cost=2.79..21.31 rows=1 width=22)
9   Hash Cond: (d.country_id = e.country_id)
10  -> Seq Scan on city d (cost=0.00..18.50 rows=10 width=15)
11  Filter: ((city)::text = ANY ('{Aurora,Atlixco,Xintai,Adoni,Dhule(Dhulia),Kurashiki,Pingxiang,Sivas,Celaya,"So Leopoldo"}'::text[]))
12  -> Hash (cost=2.66..2.66 rows=10 width=13)
13  -> Seq Scan on country e (cost=0.03..2.66 rows=10 width=13)
14  Filter: ((country)::text = ANY ('{India,China,"United States",Japan,Mexico,Brazil,"Russian Federation",Philippines,Turkey,Indonesia}'::text[]))
15  -> Nested Loop Left Join (cost=0.84..6.85 rows=24 width=23)
16  -> Nested Loop Left Join (cost=0.55..5.08 rows=1 width=19)
17  -> Index Scan using idx_fk_city_id on address c (cost=0.28..4.69 rows=1 width=6)
18  Index Cond: (city_id = d.city_id)
19  -> Index Scan using idx_fk_address_id on customer b (cost=0.28..0.38 rows=1 width=19)
20  Index Cond: (address_id = c.address_id)
21  -> Index Scan using idx_fk_customer_id on payment a (cost=0.29..1.53 rows=24 width=8)
22  Index Cond: (customer_id = b.customer_id)
Total rows: 22 of 22    Query complete 00:00:00.215
```

CTE



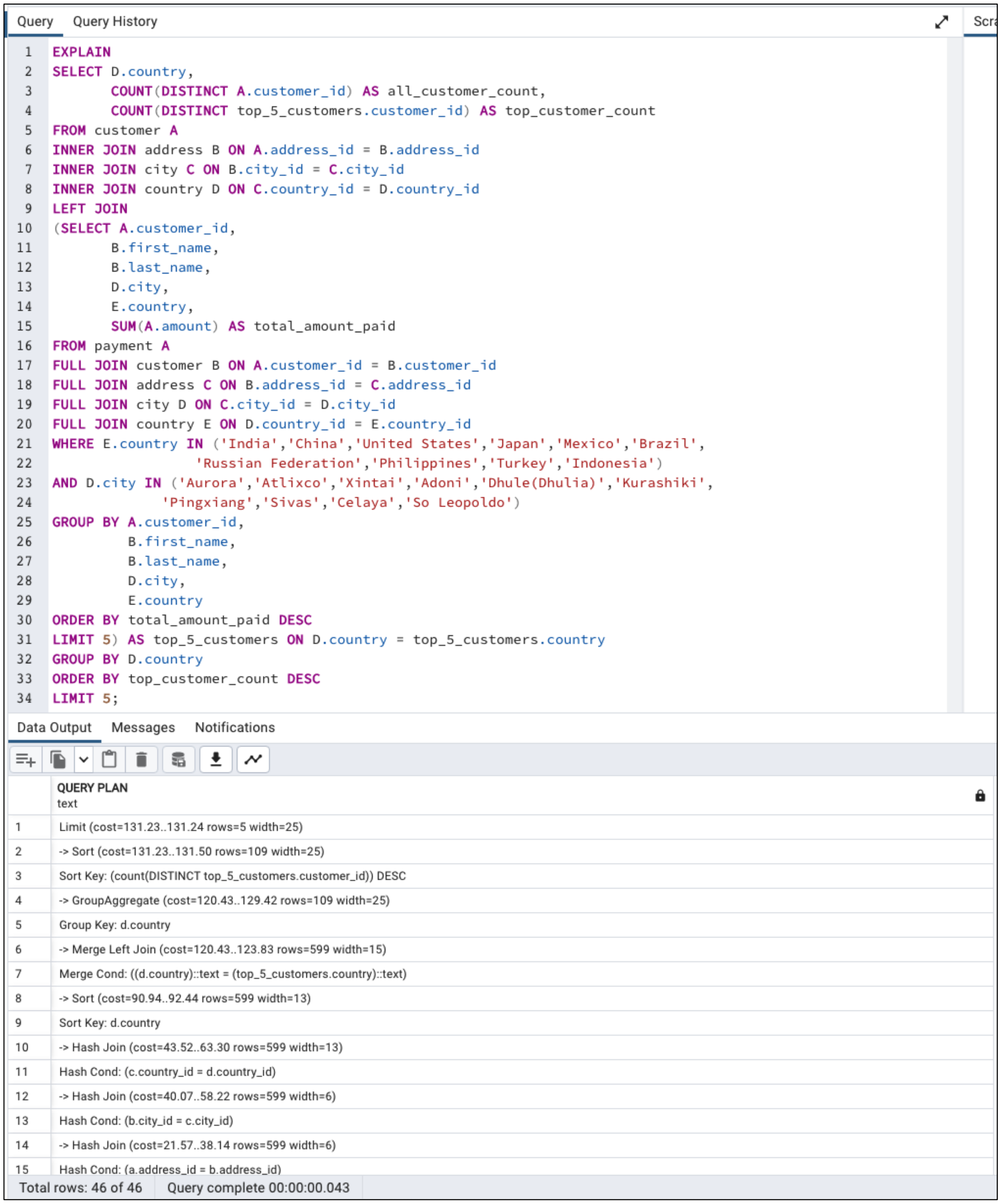
```
EXPLAIN
WITH average_amount_paid_cte (customer_id, first_name, last_name,
city, country, total_amount_paid) AS
       (SELECT A.customer_id,
       B.first_name,
       B.last_name,
       D.city,
       E.country,
       SUM(A.amount) AS total_amount_paid
FROM payment A
FULL JOIN customer B ON A.customer_id = B.customer_id
FULL JOIN address C ON B.address_id = C.address_id
FULL JOIN city D ON C.city_id = D.city_id
FULL JOIN country E ON D.country_id = E.country_id
WHERE E.country IN ('India','China','United States','Japan','Mexico','Brazil',
                 'Russian Federation','Philippines','Turkey','Indonesia')
AND D.city IN ('Aurora','Atlixco','Xintai','Adoni','Dhule(Dhulia)','Kurashiki',
             'Pingxiang','Sivas','Celaya','So Leopoldo')
GROUP BY A.customer_id,
         B.first_name,
         B.last_name,
         D.city,
         E.country
ORDER BY total_amount_paid DESC
LIMIT 5)
SELECT AVG(total_amount_paid) AS average
FROM average_amount_paid_cte
```

	QUERY PLAN text
1	Aggregate (cost=29.45..29.46 rows=1 width=32)
2	-> Limit (cost=29.38..29.39 rows=5 width=65)
3	-> Sort (cost=29.38..29.43 rows=22 width=65)
4	Sort Key: (sum(a.amount)) DESC
5	-> HashAggregate (cost=28.74..29.01 rows=22 width=65)
6	Group Key: a.customer_id, b.first_name, b.last_name, d.city, e.country
7	-> Nested Loop Left Join (cost=3.62..28.41 rows=22 width=39)
8	-> Hash Join (cost=2.79..21.31 rows=1 width=22)
9	Hash Cond: (d.country_id = e.country_id)
10	-> Seq Scan on city d (cost=0.00..18.50 rows=10 width=15)
11	Filter: ((city)::text = ANY ('{Aurora,Atlixco,Xintai,Adoni,Dhule(Dhulia),Kurashiki,Pingxiang,Sivas,Celaya,"So Leopoldo"}'::text[]))
12	-> Hash (cost=2.66..2.66 rows=10 width=13)
13	-> Seq Scan on country e (cost=0.03..2.66 rows=10 width=13)
14	Filter: ((country)::text = ANY ('{India,China,"United States",Japan,Mexico,Brazil,"Russian Federation",Philippines,Turkey,Indonesia}'::text[]))
15	-> Nested Loop Left Join (cost=0.84..6.85 rows=24 width=23)
16	-> Nested Loop Left Join (cost=0.55..5.08 rows=1 width=19)
17	-> Index Scan using idx_fk_city_id on address c (cost=0.28..4.69 rows=1 width=6)
18	Index Cond: (city_id = d.city_id)
19	-> Index Scan using idx_fk_address_id on customer b (cost=0.28..0.38 rows=1 width=19)
20	Index Cond: (address_id = c.address_id)

Total rows: 22 of 22   Query complete 00:00:00.056

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

<u>Subquery</u>


Query Query History

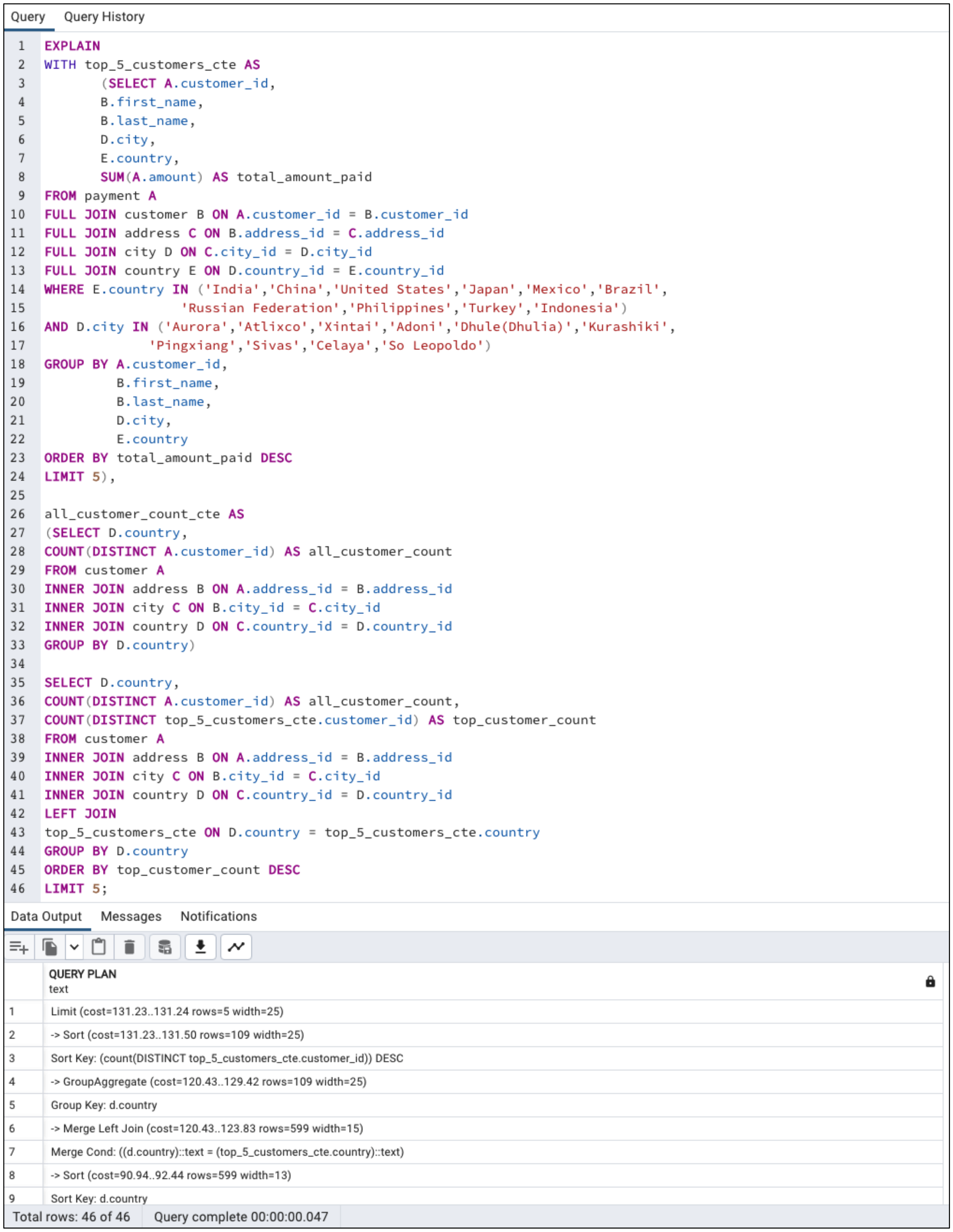
```
EXPLAIN
SELECT D.country,
       COUNT(DISTINCT A.customer_id) AS all_customer_count,
       COUNT(DISTINCT top_5_customers.customer_id) AS top_customer_count
FROM customer A
INNER JOIN address B ON A.address_id = B.address_id
INNER JOIN city C ON B.city_id = C.city_id
INNER JOIN country D ON C.country_id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

Data Output Messages Notifications

```
   QUERY PLAN
   text
1  Limit (cost=131.23..131.24 rows=5 width=25)
2  -> Sort (cost=131.23..131.50 rows=109 width=25)
3  Sort Key: (count(DISTINCT top_5_customers.customer_id)) DESC
4  -> GroupAggregate (cost=120.43..129.42 rows=109 width=25)
5  Group Key: d.country
6  -> Merge Left Join (cost=120.43..123.83 rows=599 width=15)
7  Merge Cond: ((d.country)::text = (top_5_customers.country)::text)
8  -> Sort (cost=90.94..92.44 rows=599 width=13)
9  Sort Key: d.country
10 -> Hash Join (cost=43.52..63.30 rows=599 width=13)
11 Hash Cond: (c.country_id = d.country_id)
12 -> Hash Join (cost=40.07..58.22 rows=599 width=6)
13 Hash Cond: (b.city_id = c.city_id)
14 -> Hash Join (cost=21.57..38.14 rows=599 width=6)
15 Hash Cond: (a.address_id = b.address_id)
```

Total rows: 46 of 46 Query complete 00:00:00.043

CTE



Query   Query History

```sql
EXPLAIN
WITH top_5_customers_cte AS
       (SELECT A.customer_id,
       B.first_name,
       B.last_name,
       D.city,
       E.country,
       SUM(A.amount) AS total_amount_paid
FROM payment A
FULL JOIN customer B ON A.customer_id = B.customer_id
FULL JOIN address C ON B.address_id = C.address_id
FULL JOIN city D ON C.city_id = D.city_id
FULL JOIN country E ON D.country_id = E.country_id
WHERE E.country IN ('India','China','United States','Japan','Mexico','Brazil',
                 'Russian Federation','Philippines','Turkey','Indonesia')
AND D.city IN ('Aurora','Atlixco','Xintai','Adoni','Dhule(Dhulia)','Kurashiki',
             'Pingxiang','Sivas','Celaya','So Leopoldo')
GROUP BY A.customer_id,
         B.first_name,
         B.last_name,
         D.city,
         E.country
ORDER BY total_amount_paid DESC
LIMIT 5),

all_customer_count_cte AS
(SELECT D.country,
COUNT(DISTINCT A.customer_id) AS all_customer_count
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
GROUP BY D.country)

SELECT D.country,
COUNT(DISTINCT A.customer_id) AS all_customer_count,
COUNT(DISTINCT top_5_customers_cte.customer_id) AS top_customer_count
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
top_5_customers_cte ON D.country = top_5_customers_cte.country
GROUP BY D.country
ORDER BY top_customer_count DESC
LIMIT 5;
```

Data Output   Messages   Notifications

```
   QUERY PLAN
   text
1  Limit (cost=131.23..131.24 rows=5 width=25)
2  -> Sort (cost=131.23..131.50 rows=109 width=25)
3  Sort Key: (count(DISTINCT top_5_customers_cte.customer_id)) DESC
4  -> GroupAggregate (cost=120.43..129.42 rows=109 width=25)
5  Group Key: d.country
6  -> Merge Left Join (cost=120.43..123.83 rows=599 width=15)
7  Merge Cond: ((d.country)::text = (top_5_customers_cte.country)::text)
8  -> Sort (cost=90.94..92.44 rows=599 width=13)
9  Sort Key: d.country
```

Total rows: 46 of 46   Query complete 00:00:00.047

4. **Did the results surprise you? Write a few sentences to explain your answer.**

   Yes, because since CTE's main advantage is readability and maintainability, CTE saves hundreds of lines of code. Instead of repeating a huge subquery, one can just use the alias as a variable. The CTE can serve in ad-hoc queries, which is why I would assume the projected run time of the query would be less.

Step 3:

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

The challenges I have experienced when replacing subqueries with CTEs is the unexpected results and errors. It's very easy to mistake CTE as a temporary table and assume that the visible order of steps will be identical as the actual order of execution. For example, step 2 in exercise 3.8 for subqueries and 3.9 for CTEs were complex. After numerous attempts in recreating the query and main statement, this is the kind of example that requires the capacity to "think outside the box" in order to understand the reason behind the answers.