### 3.9 Common Table Expressions

Step 1

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 rewritten

Rockbuster/postgres@PostgreSQL 14

Query Editor | Query History

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

Data Output | Explain | Messages | Notifications

| | avg<br>numeric |
|---|---|
| 1 | 107.3540000000000000 |

3.8 Step 2 rewritten

Rockbuster/postgres@PostgreSQL 14

Query Editor | Query History

```sql
WITH top_5_cte(customer_id, first_name, last_name, country, city, sum_amount) AS
(SELECT A.customer_id,
        B.first_name,
        B.last_name,
        E.country,
        D.city,
        SUM(A.amount) AS total_amount_paid
FROM payment A
INNER JOIN customer B ON A.customer_id = B.customer_id
INNER JOIN address C ON B.address_id = C.address_id
INNER JOIN city D ON C.city_id = D.city_id
INNER JOIN country E ON D.country_id = E.country_id
WHERE country IN ('India','China','United States','Japan','Mexico','Brazil',
                  'Russia Federation','Philippines','Turkey','Indonesia')
AND city IN ('Aurora','Atlixco','Xintai','Adoni','Dhule (Dhulla)','Kurashiki',
             'Pingxiang','Sivas','Celaya','So Leopoldo')
GROUP BY A.customer_id,
         B.first_name,
         B.last_name,
         E.country,
         D.city
ORDER BY total_amount_paid DESC LIMIT 5)
SELECT country.country,
       COUNT(DISTINCT customer.customer_id) AS all_customer_count,
       COUNT(DISTINCT country.country) AS top_customer_count
FROM top_5_cte
LEFT JOIN customer ON customer.customer_id = customer.customer_id
LEFT JOIN address ON customer.address_id = address.address_id
LEFT JOIN city ON address.city_id = city.city_id
LEFT JOIN country ON city.country_id = country.country_id
GROUP BY country.country
ORDER BY all_customer_count DESC LIMIT 5;
```

Data Output | Messages | Explain | 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 |

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

   I define the CTE with the 'WITH' clause and gave it an appropriate expression name. Then, I listed columns that will be listed in the CTE definition and used the AS keyword. I copied the query from previous exercise as then used SELECT again to display the result.

Step 2

1. Which approach do you think will perform better and why?
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.
4. Did the results surprise you? Write a few sentences to explain your answer.

In general, I was of the opinion that CTE will perform better due to its readability and the fact that it is only defined once. However the result of 'EXPLAIN' keyword indicated that that for this query both CTE and Subquery have the same cost.

## CTE Step 1

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

| | QUERY PLAN text |
|---|---|
| 1 | Aggregate (cost=29.22..29.23 rows=1 width=32) |
| 2 | [...] -> Limit (cost=29.15..29.16 rows=5 width=65) |
| 3 | [...] -> Sort (cost=29.15..29.20 rows=22 width=65) |
| 4 | [...] Sort Key: (sum(a.amount)) DESC |
| 5 | [...] -> HashAggregate (cost=28.51..28.78 rows=22 width=65) |
| 6 | [...] Group Key: a.customer_id, b.first_name, b.last_name, e.country, d.city |
| 7 | [...] -> Nested Loop (cost=3.62..28.18 rows=22 width=39) |
| 8 | [...] -> Nested Loop (cost=3.33..26.40 rows=1 width=35) |
| 9 | [...] -> Nested Loop (cost=3.06..26.01 rows=1 width=22) |
| 10 | [...] -> Hash Join (cost=2.79..21.31 rows=1 width=22) |
| 11 | [...] Hash Cond: (d.country_id = e.country_id) |
| 12 | [...] -> Seq Scan on city d (cost=0.00..18.50 rows=10 width=15) |

Successfully run. Total query runtime: 51 msec. 22 rows affected.

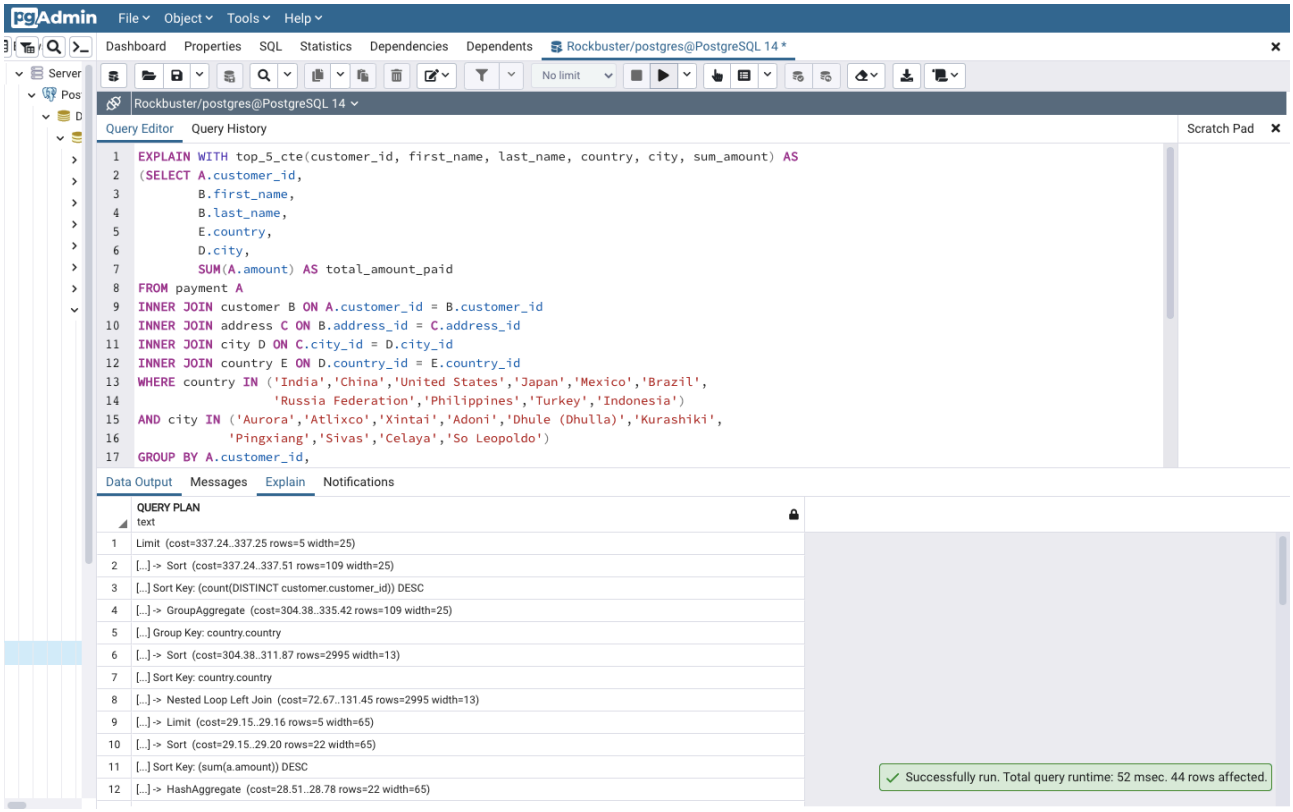
## Subquery Step 1

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

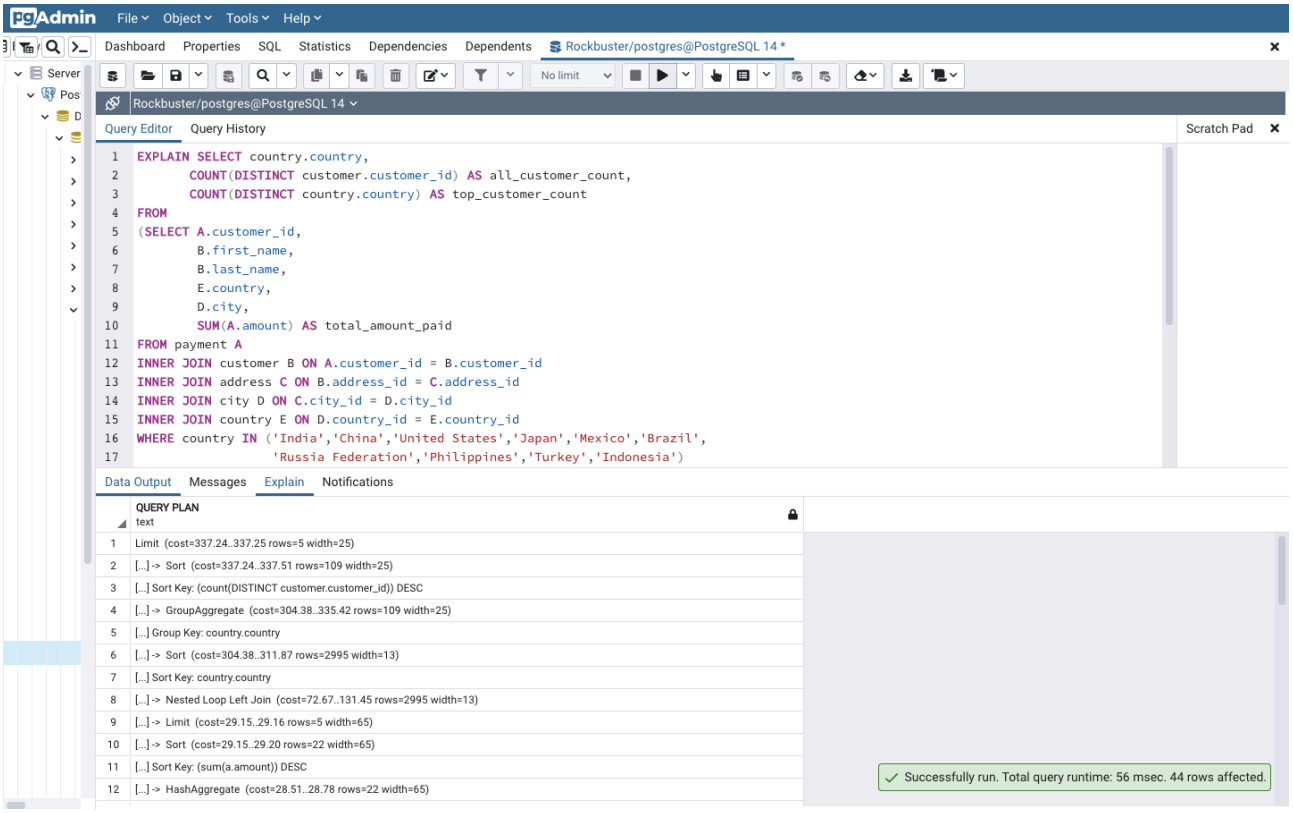
| | QUERY PLAN text |
|---|---|
| 1 | Aggregate (cost=29.22..29.23 rows=1 width=32) |
| 2 | [...] -> Limit (cost=29.15..29.16 rows=5 width=65) |
| 3 | [...] -> Sort (cost=29.15..29.20 rows=22 width=65) |
| 4 | [...] Sort Key: (sum(a.amount)) DESC |
| 5 | [...] -> HashAggregate (cost=28.51..28.78 rows=22 width=65) |
| 6 | [...] Group Key: a.customer_id, b.first_name, b.last_name, e.country, d.city |
| 7 | [...] -> Nested Loop (cost=3.62..28.18 rows=22 width=39) |
| 8 | [...] -> Nested Loop (cost=3.33..26.40 rows=1 width=35) |
| 9 | [...] -> Nested Loop (cost=3.06..26.01 rows=1 width=22) |
| 10 | [...] -> Hash Join (cost=2.79..21.31 rows=1 width=22) |
| 11 | [...] Hash Cond: (d.country_id = e.country_id) |
| 12 | [...] -> Seq Scan on city d (cost=0.00..18.50 rows=10 width=15) |

Successfully run. Total query runtime: 41 msec. 22 rows affected.

## CTE Step 2




## Subquery Step 2



Step 3

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

The challenges I faced include determine what is needed to build the CTE. I am aware that unlike subqueries, the CTE are defined at the start of query. Also, writing a totally new SELECT statement querying the temporary table created with CTE seems strange initially. However, after spending more time to read the syntax everything was quite easier to understand.