

3.9: Common Table Expressions

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

Step 1:

```
WITH Average_paid_amountCTE(customer_id, first_name, last_name, country, city,
    total_amount_paid) AS

(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)

SELECT AVG(total_amount_paid) AS average FROM Average_paid_amountCTE
```

The screenshot shows a SQL IDE interface. The top toolbar contains icons for file operations, query execution, and settings. Below the toolbar, the 'Query' tab is active, displaying a SQL query. The 'Data Output' tab is also visible, showing the result of the query.

```

1  -- CTE
2  WITH Average_paid_amountCTE(customer_id, first_name, last_name, country, city, total_amount_paid) AS
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 FROM customer A
5  INNER JOIN address B ON A.address_id = B.address_id
6  INNER JOIN city C on B.city_id = C.city_id
7  INNER JOIN country D ON C.country_id = D.country_id
8  INNER JOIN payment E ON A.customer_id = E.customer_id
9  WHERE city IN('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'Teboksary', 'Tianji',
10 'Cianjur', 'So Leopoldo')
11 GROUP BY country, city, first_name, last_name, A.customer_id
12 ORDER BY total_amount_paid DESC
13 LIMIT 5)
14
15 SELECT AVG(total_amount_paid) AS average FROM Average_paid_amountCTE

```

The 'Data Output' tab shows the result of the query:

average
105.554000000000000000

Step 2:

```

WITH top_5_customersCTE(customer_id, first_name, last_name, country, city,
total_amount_paid) AS

(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)

SELECT D.country, COUNT(A.customer_id) AS All_customer_count,
count(top_5_customersCTE) 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

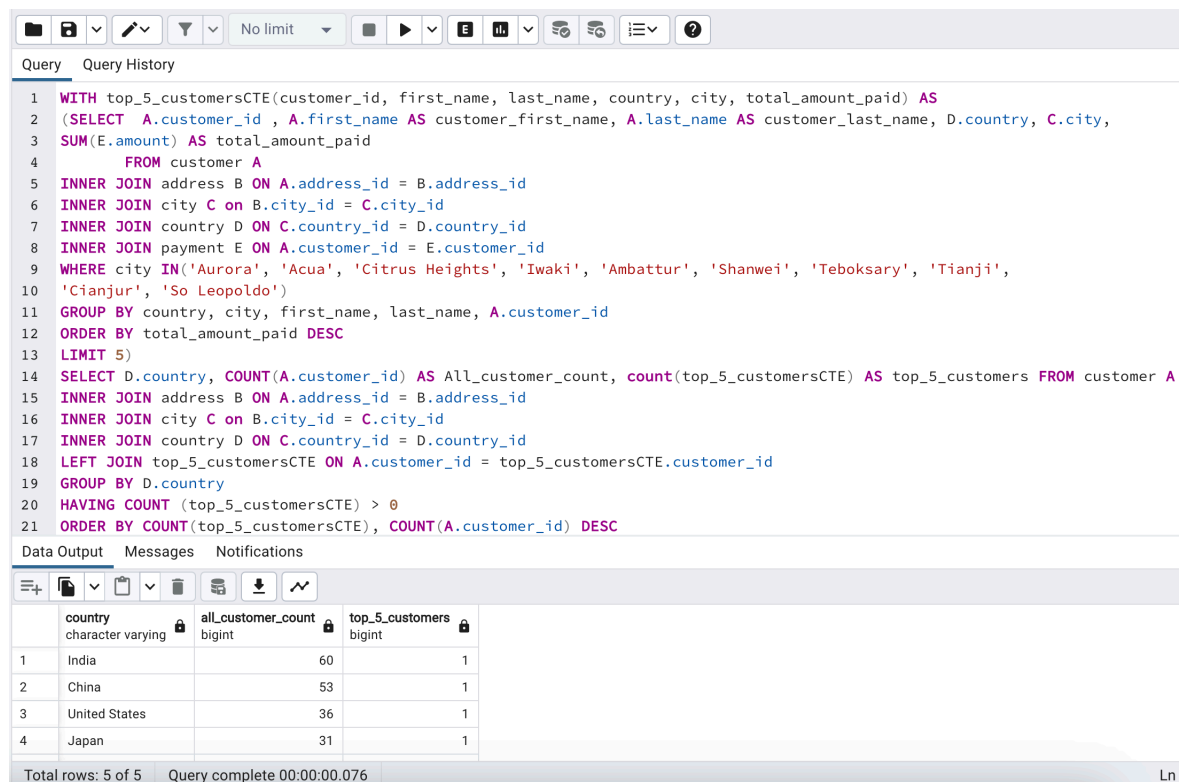
INNER JOIN country D ON C.country_id = D.country_id

LEFT JOIN top_5_customersCTE ON A.customer_id = top_5_customersCTE.customer_id

GROUP BY D.country

HAVING COUNT (top_5_customersCTE) > 0

ORDER BY COUNT(top_5_customersCTE), COUNT(A.customer_id) DESC



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 WITH top_5_customersCTE(customer_id, first_name, last_name, country, city, total_amount_paid) AS
2 (SELECT A.customer_id , A.first_name AS customer_first_name, A.last_name AS customer_last_name, D.country, C.city,
3 SUM(E.amount) AS total_amount_paid
4 FROM customer A
5 INNER JOIN address B ON A.address_id = B.address_id
6 INNER JOIN city C on B.city_id = C.city_id
7 INNER JOIN country D ON C.country_id = D.country_id
8 INNER JOIN payment E ON A.customer_id = E.customer_id
9 WHERE city IN('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'Teboksary', 'Tianji',
10 'Cianjur', 'So Leopoldo'))
11 GROUP BY country, city, first_name, last_name, A.customer_id
12 ORDER BY total_amount_paid DESC
13 LIMIT 5)
14 SELECT D.country, COUNT(A.customer_id) AS All_customer_count, count(top_5_customersCTE) AS top_5_customers FROM customer A
15 INNER JOIN address B ON A.address_id = B.address_id
16 INNER JOIN city C on B.city_id = C.city_id
17 INNER JOIN country D ON C.country_id = D.country_id
18 LEFT JOIN top_5_customersCTE ON A.customer_id = top_5_customersCTE.customer_id
19 GROUP BY D.country
20 HAVING COUNT (top_5_customersCTE) > 0
21 ORDER BY COUNT(top_5_customersCTE), COUNT(A.customer_id) DESC
```

The results are shown in a table with the following columns: country, all_customer_count, and top_5_customers.

	country	all_customer_count	top_5_customers
1	India	60	1
2	China	53	1
3	United States	36	1
4	Japan	31	1

Total rows: 5 of 5 Query complete 00:00:00.076

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

First I extracted the query from previous Exercise 3.8. Then I used the CTE table at the beginning of the query and referred to it as a separate table to join with the other tables to complete the task and gave it a name. Then added it to the last SELECT statement to get data from that table. It was very easy with the CTE table.

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

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.
-
1. I think CTE is much better than Subqueries in one case which is readability. One more is that a CTE can be used many times within a query, whereas a subquery can only be used once.
 2. This can make the query definition much shorter, but it won't necessarily result in improved performance. If we compare cost for both queries than we can see cost for both by using EXPLAIN.

Using Subqueries:

Part 1: cost=64.49

Part 2: cost=136.93

Using CTE

Part 1: cost=64.49.

Part 2: cost=136.93

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

The result did not surprised me because there was no significant difference in both the queries. CTE is just improve accessibility and readability.

Step 3: Write 1 to 2 paragraphs on the challenges you faced when replacing your subqueries with CTEs.

I didn't find such challenges while replacing subqueries, even I find it easy. Because we can call our CTE table in our main query once we create the CTE table. We do not need to create CTE table again and again which is the good part of it. The only thing is that we can keep the flow in mind while replacing the queries.