

Query Query History

```
1 WITH paid_cte(customer_id,customer_first_name,customer_last_name,country,
2     city,total_amount_paid)
3 AS (SELECT A.customer_id,A.first_name AS customer_first_name,A.last_name
4     AS customer_last_name,
5     D.country,C.city,SUM(amount) AS total_amount_paid
6     FROM customer A
7     INNER JOIN address B ON A.address_id=B.address_id
8     INNER JOIN city C ON B.city_id=C.city_id
9     INNER JOIN country D ON C.country_id=D.country_id
10    INNER JOIN payment E ON A.customer_id=E.customer_id
11    WHERE city IN ('Cianjur','Kuwana','Acua','Shivapuri','Iwaki','Guadalajara')
12    GROUP BY country,city,first_name,last_name,A.customer_id
13    ORDER BY SUM(amount) DESC
14    LIMIT 5)
15 SELECT customer_id,customer_first_name,customer_last_name,country,city,total_amount_paid
16 FROM paid_cte
17
```

```
1 SELECT D.country,COUNT(DISTINCT A.customer_id) AS all_customer_count,
2     COUNT(DISTINCT top_five_customer.customer_id) AS top_customer_count
3 FROM customer A
4 INNER JOIN address B ON A.address_id=B.address_id
5 INNER JOIN city C ON B.city_id=C.city_id
6 INNER JOIN country D ON C.country_id=D.country_id
7 LEFT JOIN
8 (SELECT A.customer_id,A.first_name AS customer_first_name,A.last_name AS customer_last_name,
9     D.country,C.city,SUM(amount) AS total_amount_paid
10    FROM customer A
11    INNER JOIN address B ON A.address_id=B.address_id
12    INNER JOIN city C ON B.city_id=C.city_id
13    INNER JOIN country D ON C.country_id=D.country_id
14    INNER JOIN payment E ON A.customer_id=E.customer_id
15    WHERE city IN ('Cianjur','Kuwana','Acua','Iwaki','Guadalajara','Ambattur','Jakarta','Karnal',
16        'Bhusawal')
17    GROUP BY country,city,first_name,last_name,A.customer_id
18    ORDER BY SUM(amount) DESC
19    LIMIT 5) AS top_five_customer ON D.country=top_five_customer.country
20 GROUP BY D.country
21 ORDER BY top_customer_count DESC
22
23
```

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

First of all, I defined CTE in the beginning of the query, then I referred to it as a separate table to join with the other tables to complete the task. The rest of the code were the same as in case of the previous task 3.8. It was easier for me to work with CTE as it simplifies things and makes the code more readable.

STEP 2

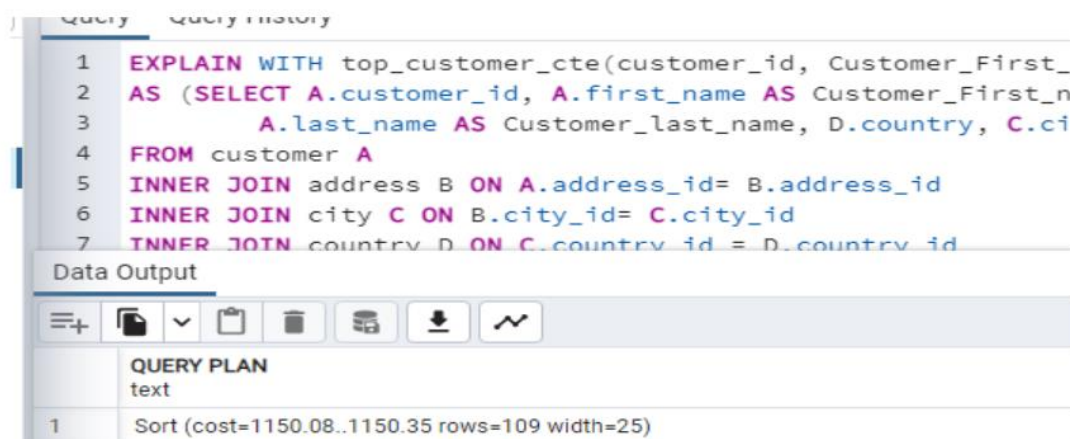
1. Which approach do you think will perform better and why?

In this context better performance means with less costs. I suppose we shouldn't expect significant differences in regard of costs, as we have almost the same number of lines in both cases

But for code with too many lines, where we should make reference to subquery several times, CTE approach can be less costly as we should define subquery in the beginning and then refer to the name without inserting the whole subquery in the utter query again and again.

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

.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



The screenshot shows the pgAdmin 4 interface. The top pane displays a SQL query using a Common Table Expression (CTE). The query is as follows:

```
1 EXPLAIN WITH top_customer_cte(customer_id, Customer_First_
2 AS (SELECT A.customer_id, A.first_name AS Customer_First_n
3      A.last_name AS Customer_last_name, D.country, C.ci
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
```

The bottom pane shows the 'Data Output' tab, which contains the 'QUERY PLAN' for the executed query. The plan is as follows:

	QUERY PLAN
1	Sort (cost=1150.08..1150.35 rows=109 width=25)

```
1 EXPLAIN SELECT D.country, COUNT( DISTINCT A.customer_id)AS all_customer_count,
2 COUNT(DISTINCT top_five_customer.customer_id) AS top_customer_count
3 FROM customer A
4 INNER JOIN address B ON A.address_id= B.address_id
5 INNER JOIN city C ON B.city_id = C.city_id
6 INNER JOIN country D ON C.country_id = D.country_id
7 LEFT JOIN
8 (SELECT A.customer_id, A.first_name AS Customer_First_name,
```

Data Output

QUERY PLAN
text

1 Sort (cost=1150.08..1150.35 rows=109 width=25)

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

It was not surprise at all, because we almost did the same things. You can see that the estimated costs of two queries: the first one with CTE and the second one with

inner queries, are the same. But with CTE case I felt more comfortable, as it seems clearer to me.

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

I didn't face any challenges to replace subqueries with CTEs. Task 3.8 was more challengeable for me, as I made a few mistakes (from typo error to other type of mistakes) and I tried a few times to correct them.

For me, definitely, CTE case is easier to work, because you have to define the imaginary table in the beginning, then refer to it rather than insert it in the middle of code