

TASK_9_Common Table Expressions

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

The screenshot shows the pgAdmin 4 interface with a SQL query editor. The query uses a Common Table Expression (CTE) named `cte_amount_paid` to calculate the average amount paid per country. The query is as follows:

```
1 WITH cte_amount_paid
2 AS
3 (SELECT A.customer_id, A.first_name, A.last_name, c.city, d.country, SUM(amount) AS total_amount_paid
4 FROM customer A
5 INNER JOIN address B ON A.address_id = B.address_id INNER JOIN city C ON B.city_id = C.city_id
6 INNER JOIN country D ON C.country_id = D.country_id INNER JOIN payment E ON A.customer_id = E.customer_id WHERE c.city IN(SELECT C.city
7 FROM customer A
8 INNER JOIN address B ON A.address_id = B.address_id INNER JOIN city C ON B.city_id = C.city_id
9 INNER JOIN country D ON C.country_id = D.country_id WHERE D.country IN
10 (SELECT D.country
11 FROM customer A
12 INNER JOIN address B ON A.address_id = B.address_id INNER JOIN city C ON B.city_id = C.city_id
13 INNER JOIN country D ON C.country_id = D.country_id GROUP BY country
14 ORDER BY COUNT (customer_id) DESC
15 LIMIT 10)
16 GROUP BY D.country, C.city
17 ORDER BY COUNT (customer_id) DESC
18 LIMIT 10)
19 GROUP BY A.customer_id, A.first_name, A.last_name, c.city, d.country
20 ORDER BY total_amount_paid DESC LIMIT 5)
21 SELECT ROUND(AVG(total_amount_paid),2) AS avg_amount_paid FROM cte_amount_paid;
```

The Data Output tab shows the result of the query:

avg_amount_paid
105.55

The screenshot shows the pgAdmin 4 interface with a SQL query editor. The query uses Common Table Expressions (CTEs) to calculate the top customer counts per country. The query is as follows:

```
1 WITH top_customer_count_cte
2 AS
3 (SELECT A.customer_id, A.first_name, A.last_name, c.city, d.country, SUM(amount) AS total_amount_paid
4 FROM customer A
5 INNER JOIN address B ON A.address_id = B.address_id INNER JOIN city C ON B.city_id = C.city_id
6 INNER JOIN country D ON C.country_id = D.country_id INNER JOIN payment E ON A.customer_id = E.customer_id WHERE c.city IN(SELECT C.city
7 FROM customer A
8 INNER JOIN address B ON A.address_id = B.address_id INNER JOIN city C ON B.city_id = C.city_id
9 INNER JOIN country D ON C.country_id = D.country_id WHERE D.country IN
10 (SELECT D.country
11 FROM customer A
12 INNER JOIN address B ON A.address_id = B.address_id INNER JOIN city C ON B.city_id = C.city_id
13 INNER JOIN country D ON C.country_id = D.country_id GROUP BY country
14 ORDER BY COUNT (customer_id) DESC
15 LIMIT 10)
16 GROUP BY D.country, C.city
17 ORDER BY COUNT (customer_id) DESC
18 LIMIT 10)
19 GROUP BY A.customer_id, A.first_name, A.last_name, c.city, d.country
20 ORDER BY total_amount_paid DESC
21 LIMIT 5),
22 customer_count_cte AS
23 (SELECT D.country,
24 COUNT(DISTINCT A.customer_id) AS all_customer_count, COUNT(DISTINCT D.country) AS top_customer_count
25 FROM country D
26 INNER JOIN city C ON D.country_id = C.country_id INNER JOIN address B ON C.city_id = B.city_id
27 INNER JOIN customer A ON B.address_id = A.address_id GROUP BY D.country)
28 SELECT D.country, COUNT(DISTINCT A.customer_id) AS all_customer_count,
29 COUNT(DISTINCT top_customer_count_cte.customer_id) AS top_customer_count
30 FROM customer_count_cte, customer_count_cte;
```

The Data Output tab shows the result of the query:

country	all_customer_count	top_customer_count
1 Japan	31	1
2 Mexico	30	1
3 China	53	1
4 India	60	1
5 United States	36	1

First, I started to write the query using WITH clause. Then I removed the outer query from subquery statement from Task 3.8. Then I copied the subquery and pasted for new Task.

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

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

Because of better readability, it is easier to understand the query using CTEs. For me using CTE perform better.

The screenshot shows the pgAdmin interface with a SQL query editor. The query uses CTEs to calculate the total amount paid for each customer and then ranks them by total amount paid.

```
1 EXPLAIN WITH top_customer_count_cte
2 AS
3 (SELECT A.customer_id, A.first_name, A.last_name, c.city, d.country, SUM(amount) AS total_amount_paid
4 FROM customer A
5 INNER JOIN address B ON A.address_id = B.address_id INNER JOIN city C ON B.city_id = C.city_id
6 INNER JOIN country D ON C.country_id = D.country_id INNER JOIN payment E ON A.customer_id = E.customer_id WHERE c.city IN(SELECT C.city
7 FROM customer A
8 INNER JOIN address B ON A.address_id = B.address_id INNER JOIN city C ON B.city_id = C.city_id
9 INNER JOIN country D ON C.country_id = D.country_id WHERE D.country IN
10 (SELECT D.country
11 FROM customer A
12 INNER JOIN address B ON A.address_id = B.address_id INNER JOIN city C ON B.city_id = C.city_id
13 INNER JOIN country D ON C.country_id = D.country_id GROUP BY country
14 ORDER BY COUNT (customer_id) DESC
15 LIMIT 10)
16 GROUP BY D.country, C.city
17 ORDER BY COUNT (customer_id) DESC
18 LIMIT 10)
19 GROUP BY A.customer_id, A.first_name, A.last_name, c.city, d.country
20 ORDER BY total_amount_paid DESC
21 LIMIT 5);
```

The query plan shows the execution strategy, including nested loops and hash joins.

The screenshot shows the pgAdmin interface with a SQL query editor. The query uses subqueries to calculate the average total amount paid and then ranks customers by total amount paid.

```
1 EXPLAIN SELECT AVG(total_amount_paid) AS average
2 FROM (SELECT A.customer_id, A.first_name, A.last_name, C.city, D.country, SUM(amount) AS total_amount_paid FROM customer A
3 INNER JOIN address B ON A.address_id = B.address_id
4 INNER JOIN city C ON B.city_id = C.city_id
5 INNER JOIN country D ON C.country_id = D.country_id
6 INNER JOIN payment E ON A.customer_id = E.customer_id
7 WHERE C.city IN
8 (SELECT C.city
9 FROM customer A
10 INNER JOIN address B ON A.address_id = B.address_id INNER JOIN city C ON B.city_id = C.city_id
11 INNER JOIN country D ON C.country_id = D.country_id WHERE D.country IN
12 (SELECT D.country
13 FROM customer A
14 INNER JOIN address B ON A.address_id = B.address_id INNER JOIN city C ON B.city_id = C.city_id
15 INNER JOIN country D ON C.country_id = D.country_id
16 GROUP BY D.country
17 ORDER BY COUNT(customer_id) DESC LIMIT 10)
18 GROUP BY D.country, C.city
19 ORDER BY COUNT(customer_id) DESC LIMIT 10)
20 GROUP BY A.customer_id, A.first_name, A.last_name, C.city, D.country ORDER BY sum(amount) DESC
21 LIMIT 5) AS total_amount_paid_by_top_5_customers
```

The query plan shows the execution strategy, including nested loops and hash joins.

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

It was surprising to see that, the subquery (39 msec) was faster than CTE (41msec). Even the time difference was not enormous, that was surprising for me.

Step 3:

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

In the first Task, using CTE was easy and straightforward. However, in Step 2 I had to thank about a lot for the modification. Due to complexity of Step 2, I had to dive deeper to where the columns to be extracted. Practice makes the master (; .