Juan Ignacio Galvalisi

Exercise 3.9: Common Table Expressions (CTE)

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

1. **Rewrite your queries from steps 1 and 2 of task 3.8 as CTEs.**

<u>3.8 - Step 1</u>

Rockbuster/postgres@PostgreSQL 14

Query Editor Query History

```sql
SELECT
    ROUND(AVG(total_amount_paid),2) AS average
FROM
(SELECT
    A.customer_id,
    A.first_name,
    A.last_name,
    D.city,
    E.country,
    SUM(B.amount) AS total_amount_paid
FROM customer A
INNER JOIN payment B ON A.customer_id = B.customer_id
INNER JOIN address C ON A.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 D.city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)',
'Kurashiki', 'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
GROUP BY
    A.customer_id,
    A.first_name,
    A.last_name,
    D.city,
    E.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS total_amount_paid;
```

Data Output Explain Messages Notifications

	average numeric
1	107.35

3.8 - Step 2

Rockbuster/postgres@PostgreSQL 14

Query Editor Query History

```sql
SELECT
    DISTINCT (A.country),
    COUNT(DISTINCT D.address_id) AS all_customer_count,
    COUNT(DISTINCT top_5_customers) AS top_customer_count
FROM country A
INNER JOIN city B ON A.country_id = B.country_id
INNER JOIN address C ON B.city_id = C.city_id
INNER JOIN customer D ON C.address_id = D.address_id
LEFT JOIN
(SELECT
    A.customer_id,
    A.first_name,
    A.last_name,
    D.city,
    E.country,
    SUM(B.amount) AS total_amount_paid
FROM customer A
INNER JOIN payment B ON A.customer_id = B.customer_id
INNER JOIN address C ON A.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
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,
    A.first_name,
    A.last_name,
    D.city,
    E.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS top_5_customers
ON A.country = top_5_customers.country
GROUP BY A.country
ORDER BY top_customer_count DESC
LIMIT 4;
```

Data Output Explain Messages Notifications

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

2. Copy-paste your CTEs and their outputs into your answers document.

3.9 - Step 1

Rockbuster/postgres@PostgreSQL 14

Query Editor | Query History

```sql
WITH
final_average (customer, first_name, last_name, city, country, total_amount_paid) AS
(SELECT
    A.customer_id,
    A.first_name,
    A.last_name,
    D.city,
    E.country,
    SUM(B.amount) AS total_amount_paid
FROM customer A
INNER JOIN payment B ON A.customer_id = B.customer_id
INNER JOIN address C ON A.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 D.city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)',
'Kurashiki', 'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
GROUP BY
    A.customer_id,
    A.first_name,
    A.last_name,
    D.city,
    E.country
ORDER BY total_amount_paid DESC
LIMIT 5)
SELECT
    ROUND(AVG(total_amount_paid),2) AS average
FROM final_average
```

Data Output | Explain | Messages | Notifications

	average numeric
1	107.35

3.9 - Step 2

Rockbuster/postgres@PostgreSQL 14

Query Editor | Query History

```sql
WITH customers_top_5 (customer_id, first_name, last_name, city, country, total_amount_paid) AS
    (SELECT
        A.customer_id,
        A.first_name,
        A.last_name,
        D.city,
        E.country,
        SUM(B.amount) AS total_amount_paid
    FROM customer A
    INNER JOIN payment B ON A.customer_id = B.customer_id
    INNER JOIN address C ON A.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
    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,
        A.first_name,
        A.last_name,
        D.city,
        E.country
    ORDER BY total_amount_paid DESC
    LIMIT 5)
SELECT
    DISTINCT (A.country),
    COUNT(DISTINCT D.address_id) AS all_customer_count,
    COUNT(DISTINCT customers_top_5) AS top_customer_count
FROM country A
INNER JOIN city B ON A.country_id = B.country_id
INNER JOIN address C ON B.city_id = C.city_id
INNER JOIN customer D ON C.address_id = D.address_id
LEFT JOIN customers_top_5 ON A.country = customers_top_5.country
GROUP BY A.country
HAVING COUNT(DISTINCT customers_top_5) > 0
ORDER BY
    top_customer_count DESC,
    all_customer_count DESC;
```

Data Output | Explain | Messages | Notifications

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

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

In order to get my script more readable, I decided to get into my CTE the subquery based on the top 5 five customers who have spent the most on Rockbuster. With that in mind, I used the WITH function and named my CTE, placing it at the beginning of my query, just before the main statement.

Then, I wrote my main statement, which has the ultimate names of my three categories: country, all_customer_count, and top_customer_count regarding the requested task ("Find out how many of the top 5 customers are based within each country".) Moreover, I used the LEFT JOIN because I wanted to combine my main query with my CTE.

<u>Step 2: Compare the performance of your CTEs and subqueries</u>

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

Initially, I would assume CTE performs better because it is better organized, at least for the human eye. The way computers read this code differs from how humans read it, so both readings may vary.

However, we must check each case in detail because there would be more variables to consider to perform our queries in time and cost through the EXPLAIN + ANALYZE command.

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

QUERY	COST	POP-UP WINDOW	PLANNING TIME	EXECUTION TIME
Step 1 (3.8.) – SQ	(cost=67.91..67.92 rows=1 width=32)	39 msc. 22 rows affected	2.600 ms	3.069 ms
Step 1 (3.9.) – CTE	(cost=67.91..67.92 rows=1 width=32)	34 msc. 22 rows affected	1.487 ms	1.076 ms
Step 2 (3.8.) – SQ	(cost=130.61..130.97 rows=36 width=25)	107 msc. 47 rows affected	2.511 ms	9.225 ms
Step 2 (3.9.) – CTE	(cost=130.61..130.97 rows=36 width=25)	77 msc. 47 rows affected	3.558	3.133

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

At first, I would have thought that queries with CTEs would be more efficient than those containing subqueries. With the results in hand, it seems that this rule is fulfilled in general terms.

Regarding the cost required by each query with its characteristics obtained through the EXPLAIN command, we cannot observe differences between both pairs. Now, concerning the final execution times received automatically by PostgreSQL when a query is made, we find slight differences between both procedures. Since these are small data sets and short queries, the difference may be more significant as both the data to be analyzed and the complexity of the queries increase.

We can observe two absorbing matters if we analyze the data provided by the EXPLAIN ANALYZE command. For one thing, planning and execution times are higher when subqueries are used than when they are not. On the other hand, subqueries perform their task in a longer time than estimated, while CTEs do it in less time than estimated. Not only do they work better than expected, they even do it almost three times faster than the same query, but they include subqueries inside. In this sense, it is appropriate to ask, is it not necessary to pay more attention to the final time executed than the cost each query represents?

Step 3

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

At first, it was a bit difficult to get out of the abstraction and materialize the tables now with the new CET, but it is more than achievable if you think more carefully about the syntax. Once you know the syntax, it is easier to write your queries, and that rule applies here too.

On the other hand, it would have been a different situation if I had made the query from scratch since I had almost everything written thanks to exercise 3.8. In this sense, I had to configure the CET, join it to the main query, and execute it as general steps.