

### Common Table Expressions.

#### **Step 1: Answer the business questions from steps 1 and 2 of task 3.8 using CTEs**

*Step 1- A.3.8: Find the average amount paid by the top 5 customers.*

#### **Original Subquery**

```
SELECT AVG(total_payment) AS total_amount_paid
FROM (
  SELECT B.customer_id,
         B.first_name,
         B.last_name,
         D.city,
         E.country,
         SUM(A.amount) AS total_payment
  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 D.city IN

  (SELECT D.city
   FROM customer B
        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 E.country IN

    (SELECT E.country
     FROM customer B
          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
     GROUP BY E.country
     ORDER BY COUNT(B.customer_id) DESC
     LIMIT 10)

   GROUP BY E.country, D.city
   ORDER BY COUNT(B.customer_id) DESC
   LIMIT 10)

  GROUP BY E.country, D.city, B.customer_id, B.first_name, B.last_name
  ORDER BY SUM(A.amount) DESC
  LIMIT 5)
```

#### **Output 3.8.S1.1**

total_amount_paid
-------------------

105.5540000000000000
----------------------

## CTE

WITH avg\_total\_amount\_paid\_top5 AS

```
SELECT B.customer_id,  
       B.first_name,  
       B.last_name,  
       D.city,  
       E.country,  
       SUM(A.amount) AS total_payment  
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 D.city IN  
  
(SELECT D.city  
 FROM customer B  
      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 E.country IN  
  
      (SELECT E.country  
       FROM customer B  
       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  
       GROUP BY E.country  
       ORDER BY COUNT(B.customer_id) DESC  
       LIMIT 10)  
  
      GROUP BY E.country, D.city  
      ORDER BY COUNT(B.customer_id) DESC  
      LIMIT 10)  
  
GROUP BY E.country, D.city, B.customer_id, B.first_name, B.last_name  
ORDER BY SUM(A.amount) DESC  
LIMIT 5)
```

```
SELECT  
AVG(avg_total_amount_paid_top5.total_payment) AS avg_payment  
FROM avg_total_amount_paid_top5
```

**Output:**

Output 3.9 S1

avg_payment
-------------

105.5540000000000000
----------------------

### Screenshot Output.

Object Explorer
Dashboard x Properties x SQL x Analytics x Dependencies x Processes x Untitled.sql x

- > Collations
- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > **Tables (17)**
  - > actor
  - > address
  - > category
  - > city
  - > country
  - > customer
  - > employees
  - > film
  - > film\_actor
  - > film\_category
  - > films
  - > inventory
  - > language
  - > payment
  - > rental
  - > staff
  - > store
- > Trigger Functions
- > Types
- > Views
- > Subscriptions
- > postgres
- > Login/Group Roles
- > Tablespaces

Rockbuster/postgres@PostgreSQL 17

Query Query History

```

13          INNER JOIN country E ON D.country_id = E.country_id
14 WHERE D.city IN
15
16 (SELECT D.city
17  FROM customer B
18   INNER JOIN address C ON B.address_id = C.address_id
19   INNER JOIN city D ON C.city_id = D.city_id
20   INNER JOIN country E ON D.country_ID = E.country_ID
21 WHERE E.country IN
22
23          (SELECT E.country
24   FROM customer B
25   INNER JOIN address C ON B.address_id = C.address_id
26   INNER JOIN city D ON C.city_id = D.city_id
27   INNER JOIN country E ON D.country_ID = E.country_ID
28   GROUP BY E.country
29   ORDER BY COUNT(B.customer_id) DESC
30   LIMIT 10)
31
32   GROUP BY E.country, D.city
33   ORDER BY COUNT(B.customer_id) DESC
34   LIMIT 10)
35
36   GROUP BY E.country, D.city, B.customer_id, B.first_name, B.last_name
37   ORDER BY SUM(A.amount) DESC
38   LIMIT 5)
39 SELECT
40 AVG(avg_total_amount_paid_top5.total_payment) AS avg_payment
41 FROM avg_total_amount_paid_top5
42

```

Data Output Messages Notifications

avg_payment	
numeric	
1	105.5540000000000000

Total rows: 1
Query complete 00:00:00.062

Showing rows: 1 to 1
Page No: 1 of 1

LF
Ln 18, Col 15

*Step 2-A.3.8: Find out how many of the top 5 customers you identified in step 1 are based within each country.*

### Original Subquery

```
SELECT E.country, COUNT (DISTINCT B.customer_id) AS all_customer_count,
       COUNT(DISTINCT top_5_customer.country) AS top_customer_count
FROM customer B
      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
LEFT JOIN
  (SELECT B.customer_id,
        B.first_name,
        B.last_name,
        D.city,
        E.country,
        SUM(A.amount) AS total_payment
   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 D.city IN

      (SELECT D.city
       FROM customer B
            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 E.country IN

          (SELECT E.country
           FROM customer B
                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
           GROUP BY E.country
           ORDER BY COUNT(B.customer_id) DESC
           LIMIT 10)

      GROUP BY
        E.country,
        D.city
      ORDER BY COUNT(B.customer_id) DESC
      LIMIT 10)

   GROUP BY
     E.country,
     D.city,
     B.customer_id,
     B.first_name,
     B.last_name
   ORDER BY SUM(A.amount) DESC
   LIMIT 5) AS top_5_customer ON top_5_customer.country = E.country
GROUP BY E.country
ORDER BY all_customer_count DESC
LIMIT 5;
```

Output A.3.8.S.2

country	all_customer_count	top_customer_count
India	60	1
China	53	1
United States	36	1
Japan	31	1
Mexico	30	1

**CTE**

WITH

```
top_ten_country (country) AS
    (SELECT E.country
     FROM customer B
        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
     GROUP BY E.country
     ORDER BY COUNT(B.customer_id) DESC
     LIMIT 10),
```

```
top_ten_cities(city) AS
    (SELECT D.city
     FROM customer B
        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 E.country IN

        (SELECT E.country
         FROM customer B
            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
         GROUP BY E.country
         ORDER BY COUNT(B.customer_id) DESC
         LIMIT 10)

     GROUP BY
        E.country,
        D.city
     ORDER BY COUNT(B.customer_id) DESC
     LIMIT 10),
```

```
top_5_customer (customer_id, city, customer_country, total_payment) AS
```

```
    (SELECT B.customer_id,
        D.city,
        E.country AS customer_country,
        SUM(A.amount) AS total_payment
     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
GROUP BY E.country,
        D.city,
        B.customer_id
ORDER BY SUM(A.amount) DESC LIMIT 5)

```

```

SELECT
    E.country AS countrycnt,
    COUNT (DISTINCT B.customer_id) AS all_customer_count,
    COUNT (DISTINCT top_5_customer.customer_id) AS top_customer_count
FROM customer B
    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
LEFT JOIN top_5_customer ON B.customer_id = top_5_customer.customer_id
GROUP BY E.country
ORDER BY all_customer_count DESC LIMIT 5;

```

Output A.3.9 S.2

countrycnt	all_customer_count	top_customer_count
India	60	0
China	53	0
United States	36	1
Japan	31	0
Mexico	30	0

## Screenshot Output.

The screenshot shows a PostgreSQL IDE interface. The left pane displays the Object Explorer with a tree view of the database schema, including tables like 'actor', 'address', 'category', 'city', 'country', 'customer', 'employees', 'film', 'film\_actor', 'film\_category', 'films', 'inventory', 'language', 'payment', 'rental', 'staff', 'store', and 'trigger\_functions'. The main pane shows a SQL query with two Common Table Expressions (CTEs) and a main query. The query is as follows:

```

WITH
  top_ten_country (country) AS
    (SELECT E.country
     FROM customer B
     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
     GROUP BY E.country
     ORDER BY COUNT(B.customer_id) DESC
     LIMIT 10),
  top_ten_cities(city) AS
    (SELECT D.city
     FROM customer B
     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 E.country IN
       (SELECT E.country
        FROM customer B
        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
        GROUP BY E.country
        ORDER BY COUNT(B.customer_id) DESC
        LIMIT 10))
  GROUP BY
    E.country,
    D.city
  
```

The bottom pane shows the Data Output tab with a table of results:

	countrycnt	all_customer_count	top_customer_count
1	India	60	0
2	China	53	0

1. Query 1. Write 2 to 3 sentences explaining how you approached this step,
  - a) In the first query, I began defining and locating the data I needed in my ERD.
  - b) Then I determined what was my main statement to get the information I needed.
  - c) I named the statement as “total\_amount\_paid\_top5” and I obtained from the created column “total\_payment”.
  - d) Then I selected I wanted to see the average AS avg\_payment and take it from my defined statement “ total\_amount\_paid\_top5” belonging to “total\_payment”, that was giving me the top 5 payments. So, this way I would only see the total average.
2. Query 2. Write 2 to 3 sentences explaining how you approached this step.
  - a) First I decided what I wanted to see in my table, that was already determined in the subquery.
  - b) Then I created my CTE’s
  - c) And my Main Query

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

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

It depends of the situation. If I need to write over and over my Joins, probably a CTE can be more practical to use a CTE. Actually in the second step, probably wouldn't make such a difference if I only stayed with the 3 CTE.

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

- Subquery Step 1 : Aggregate (cost=166.06..166.07 rows=1 width=32)
- CTE Step 1: Aggregate (cost=166.06..166.07 rows=1 width=32)
- Subquery Step 2: Limit (cost=268.45..268.46 rows=5 width=25)
- CTE Step 2: Limit (cost=1148.60..1148.61 rows=5 width=25)

3. To find out the actual speed of your queries, run them in pgAdmin 4. After you've run each query, a popup window will display its speed in milliseconds.

- Subquery Step 1 : 97 msec
- CTE Step 1: 60 msec
- Subquery Step 2 : 113 msec
- CTE Step 2: 63 msec

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

Is interesting to see how in the case of CTE's the speed tends to be shorter that subqueries.

Now, speaking about cost, just in Step 2 we can see a difference in optimization.

## **Step 3:**

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

Many of them. First to understand the mental map of where am I taking the information from and how does it look like. Still don't understand completely the logic behind that, but intuition helps.

Had trouble with Step 2 to get the same result with the CTE and the Subquery without success in finding the problem.