

Exercise 3.9

Katherine Lecce

Step 1: Answer the business questions from steps 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. WITH top_customers AS (
 3. SELECT
 4. C.customer_id,
 5. C.first_name,
 6. C.last_name,
 7. CO.country,
 8. CI.city,
 9. SUM(P.amount) AS total_amount_payment,
 10. ROW_NUMBER() OVER (PARTITION BY CO.country ORDER BY SUM(P.amount) DESC)
- AS rn
11. FROM
12. customer AS C
13. INNER JOIN
14. payment AS P ON C.customer_id = P.customer_id
15. INNER JOIN
16. address AS A ON A.address_id = C.address_id
17. INNER JOIN
18. city AS CI ON CI.city_id = A.city_id
19. INNER JOIN
20. country CO ON CO.country_id = CI.country_id
21. WHERE
22. CI.city IN (
 23. SELECT
 24. CI.city
 25. FROM
 26. customer AS C
 27. INNER JOIN
 28. address AS A ON A.address_id = C.address_id
 29. INNER JOIN
 30. city AS CI ON CI.city_id = A.city_id
 31. INNER JOIN
 32. country CO ON CO.country_id = CI.country_id
 33. WHERE
 34. CO.country IN (
 35. SELECT
 36. CO.country
 37. FROM
 38. customer AS C
 39. INNER JOIN
 40. address AS A ON A.address_id = C.address_id
 41. INNER JOIN
 42. city AS CI ON CI.city_id = A.city_id
 43. INNER JOIN
 44. country CO ON CO.country_id = CI.country_id
 45. GROUP BY

```

46.         CO.country
47.     ORDER BY
48.         COUNT(C.customer_id) DESC
49.     LIMIT 10
50. )
51. GROUP BY
52.     CO.country, Cl.city
53. ORDER BY
54.     COUNT(C.customer_id) DESC
55. LIMIT 10
56. )
57. GROUP BY
58.     C.customer_id, CO.country, Cl.city
59. )
60. SELECT
61.     AVG(total_amount_payment) AS average
62. FROM
63.     top_customers
64. WHERE
65.     rn <= 5;
66. WITH top_customers AS (
67.     SELECT
68.         C.customer_id,
69.         C.first_name,
70.         C.last_name,
71.         CO.country,
72.         Cl.city,
73.         SUM(P.amount) AS total_amount_payment
74.     FROM
75.         customer AS C
76.     INNER JOIN
77.         payment AS P ON C.customer_id = P.customer_id
78.     INNER JOIN
79.         address AS A ON A.address_id = C.address_id
80.     INNER JOIN
81.         city AS Cl ON Cl.city_id = A.city_id
82.     INNER JOIN
83.         country CO ON CO.country_id = Cl.country_id
84. WHERE
85.     Cl.city IN (
86.         SELECT
87.             Cl.city
88.         FROM
89.             customer AS C
90.         INNER JOIN
91.             address AS A ON A.address_id = C.address_id
92.         INNER JOIN
93.             city AS Cl ON Cl.city_id = A.city_id
94.         INNER JOIN

```

```

95.         country CO ON CO.country_id = Cl.country_id
96. WHERE
97.     CO.country IN (
98.         SELECT
99.             CO.country
100.        FROM
101.            customer AS C
102.        INNER JOIN
103.            address AS A ON A.address_id = C.address_id
104.        INNER JOIN
105.            city AS Cl ON Cl.city_id = A.city_id
106.        INNER JOIN
107.            country CO ON CO.country_id = Cl.country_id
108.        GROUP BY
109.            CO.country
110.        ORDER BY
111.            COUNT(C.customer_id) DESC
112.        LIMIT 10
113.    )
114.    GROUP BY
115.        CO.country, Cl.city
116.    ORDER BY
117.        COUNT(C.customer_id) DESC
118.    LIMIT 10
119. )
120. GROUP BY
121.     C.customer_id, CO.country, Cl.city
122. ORDER BY
123.     total_amount_payment DESC
124. LIMIT 5
125. )
126. SELECT
127.     CO.country,
128.     COUNT(DISTINCT C.customer_id) AS all_customer_count,
129.     COUNT(DISTINCT top_customers.customer_id) AS top_customer_count
130. FROM
131.     customer AS C
132. INNER JOIN
133.     address AS A ON A.address_id = C.address_id
134. INNER JOIN
135.     city AS Cl ON Cl.city_id = A.city_id
136. INNER JOIN
137.     country CO ON CO.country_id = Cl.country_id
138. LEFT JOIN
139.     top_customers ON top_customers.country = CO.country
140. GROUP BY
141.     CO.country
142. ORDER BY
143.     top_customer_count DESC;

```

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

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

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

Step 1: In this step, you begin by identifying the necessary data from the Rockbuster database. I utilized the data dictionary that was created earlier in this section to choose information in the "film" and "category" tables. However, these tables aren't directly connected, so we'll need to join them using "film_id" and "category_id".

Step 2: Since we can't join the "film" and "category" tables directly, we have to use the left join key to connect the "film" and "film category" tables, and then between the "film category" and "category" tables.

Step 3: Once data from step two are joined, you can name the CTE as "category_name_cte" to summarize connecting the data from step 1 and 2. You can do this by using the "WITH" clause.

Step 4: Finally, you write the main statement to query the CTE. This statement selects the rating, category name, and counts the occurrences of film IDs grouped by rating and category name. It also orders the results by rating and the count of film IDs in descending order.

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

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

I think the CTE is an expression to use in the CTE because it is easy to reuse and plug the data you need and reuse the CTE sequence in the query function. I also think it's earlier to understand, easier to write out, and less likely to get an error when using CTEs compared to subqueries. I think it's quicker to use this route instead of subqueries for more complex and longer data logic in SQL.

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

CTE query of step 1 from task 3.8:

Task 3.8:

Query 1: "Aggregate (cost=66.27..66.28 rows=1 width=32)" → 122 msec

Query 2: " -> Limit (cost=1046.35..1046.36 rows=5 width=67)" → 134 msec

Task 3.9:

Query 1: "Aggregate (cost=166.06..166.07 rows=1 width=32)" → 196 msec.

Query 2: " "Limit (cost=268.77..268.80 rows=10 width=25)" → 168 msec.

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

I am surprised that the CTE's sub queries run slower than the subqueries just because they are easier to understand. Though it does logically make sense that they might cost more just because it might be more intricate to save the paths that connect the information together into one to create the CTE. Making the process longer and more expensive to store in the database.

Step 3:

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

Since the pgAdmin 4 is a new application that is a new application that I have never used before and the query function is a new concept for pulling and organizing data both subqueries and CTEs have been both a trial and error process. I've found the more you practice it, read resources online, and understand how the application works the earlier it get to write and get the correct data output. The data dictionary is also an easy visual and Map to more easily understand where to pull the data you need.