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.

```
WITH top_customers AS (
  SELECT
    C.customer_id,
    C.first_name,
    C.last_name,
    CO.country,
    CI.city,
    SUM(P.amount) AS total_amount_payment,
    ROW_NUMBER() OVER (PARTITION BY CO.country ORDER BY SUM(P.amount) DESC) AS rn
  FROM
    customer AS C
  INNER JOIN
    payment AS P ON C.customer_id = P.customer_id
  INNER JOIN
    address AS A ON A.address_id = C.address_id
  INNER JOIN
    city AS CI ON CI.city_id = A.city_id
  INNER JOIN
    country CO ON CO.country_id = CI.country_id
  WHERE
    CI.city IN (
      SELECT
        CI.city
      FROM
        customer AS C
      INNER JOIN
        address AS A ON A.address_id = C.address_id
      INNER JOIN
        city AS CI ON CI.city_id = A.city_id
      INNER JOIN
        country CO ON CO.country_id = CI.country_id
      WHERE
        CO.country IN (
          SELECT
            CO.country
          FROM
            customer AS C
          INNER JOIN
            address AS A ON A.address_id = C.address_id
          INNER JOIN
            city AS CI ON CI.city_id = A.city_id
          INNER JOIN
            country CO ON CO.country_id = CI.country_id
          GROUP BY
```

```
                        CO.country
                    ORDER BY
                        COUNT(C.customer_id) DESC
                    LIMIT 10
                )
            GROUP BY
                CO.country, CI.city
            ORDER BY
                COUNT(C.customer_id) DESC
            LIMIT 10
        )
    GROUP BY
        C.customer_id, CO.country, CI.city
)
SELECT
    AVG(total_amount_payment) AS average
FROM
    top_customers
WHERE
    rn <= 5;
WITH top_customers AS (
    SELECT
        C.customer_id,
        C.first_name,
        C.last_name,
        CO.country,
        CI.city,
        SUM(P.amount) AS total_amount_payment
    FROM
        customer AS C
    INNER JOIN
        payment AS P ON C.customer_id = P.customer_id
    INNER JOIN
        address AS A ON A.address_id = C.address_id
    INNER JOIN
        city AS CI ON CI.city_id = A.city_id
    INNER JOIN
        country CO ON CO.country_id = CI.country_id
    WHERE
        CI.city IN (
            SELECT
                CI.city
            FROM
                customer AS C
            INNER JOIN
                address AS A ON A.address_id = C.address_id
            INNER JOIN
                city AS CI ON CI.city_id = A.city_id
            INNER JOIN
```

```sql
          country CO ON CO.country_id = CI.country_id
      WHERE
          CO.country IN (
              SELECT
                  CO.country
                    FROM
                        customer AS C
                    INNER JOIN
                        address AS A ON A.address_id = C.address_id
                    INNER JOIN
                        city AS CI ON CI.city_id = A.city_id
                    INNER JOIN
                        country CO ON CO.country_id = CI.country_id
                    GROUP BY
                        CO.country
                    ORDER BY
                  COUNT(C.customer_id) DESC
                    LIMIT 10
                )
            GROUP BY
                CO.country, CI.city
            ORDER BY
                COUNT(C.customer_id) DESC
            LIMIT 10
        )
    GROUP BY
        C.customer_id, CO.country, CI.city
    ORDER BY
        total_amount_payment DESC
    LIMIT 5
)
SELECT
    CO.country,
    COUNT(DISTINCT C.customer_id) AS all_customer_count,
    COUNT(DISTINCT top_customers.customer_id) AS top_customer_count
FROM
    customer AS C
INNER JOIN
    address AS A ON A.address_id = C.address_id
INNER JOIN
    city AS CI ON CI.city_id = A.city_id
INNER JOIN
    country CO ON CO.country_id = CI.country_id
LEFT JOIN
    top_customers ON top_customers.country = CO.country
GROUP BY
    CO.country
ORDER BY
    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 pgAmin 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.