Career Foundry

Data Analytics Immersion

A3.E9

Kendra Jackson

<u>Step 1:</u>

Query 1

WITH top_customer_payments AS

(SELECT B.customer_id, B.first_name, B.last_name, D.city, E.country,

SUM (A.amount) AS total_amount_paid

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 ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')

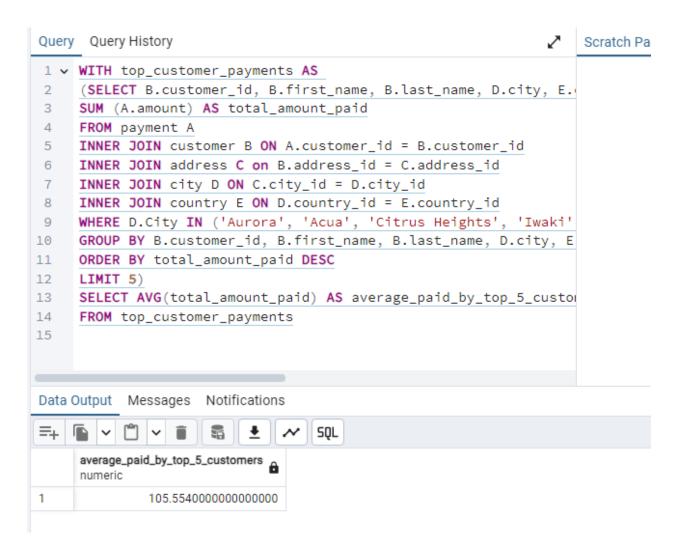
GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country

ORDER BY total_amount_paid DESC

LIMIT 5)

SELECT AVG(total_amount_paid) AS average_paid_by_top_5_customers

FROM top_customer_payments



Approach

- 1. Copy and paste query from pervious exercise
- 2. Removed the outer query and rewrote as CTE
 - Used the WITH statement and named what the table was displaying, in this top customer payments.
 - b. Changed the FROM clause to depict where the outer statement was now pulling the data from (the inner statement now named top_customer_payments)

Query 2

```
WITH top_5_customers AS (
SELECT
```

```
B.first_name,
            B.last_name,
            D.city,
            E.country,
            SUM(A.amount) AS total_amount_paid
      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 ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei',
'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')
      GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country
      ORDER BY total_amount_paid DESC
      LIMIT 5)
SELECT
      E.country,
      COUNT(DISTINCT B.customer_id) AS all_customer_count,
      COUNT(DISTINCT top_5_customers.country) AS top_customer_count
FROM Country E
INNER JOIN city D on E.country_id = D.country_id
INNER JOIN address C on D.city_id = C.city_id
```

B.customer id,

INNER JOIN customer B on C.address_id = B.address_id

LEFT JOIN top_5_customers on E.country = top_5_customers.country

GROUP BY E.country

ORDER BY all_customer_count DESC

LIMIT 5;



Approach

- 1. Copy and paste query from pervious exercise
- Removed the outer query and rewrote as CTE
 - Used the WITH statement and named what the table was displaying, in this case the top 5 customers

- b. Changed the count(distinct) statement to top_5_customers_country so
 only countries represented by the top 5 customers CTE are displayed
- c. Changed the left join so the top_5_customers CTE is added with he count of the top customers from each country

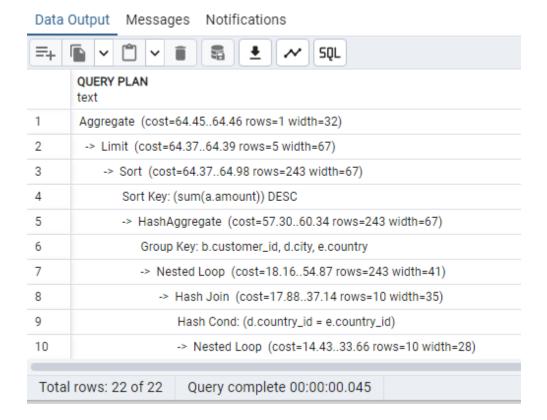
Step 2:

I am honestly not certain which approach will be better, both queries are
relatively the same length. I do however feel like the subquery may potentially
perform better, it felt like with the CTE we were just adding an extra step or two
for the database to have to go through to get our answer → create this table,
then from that select these things.

Query Plan

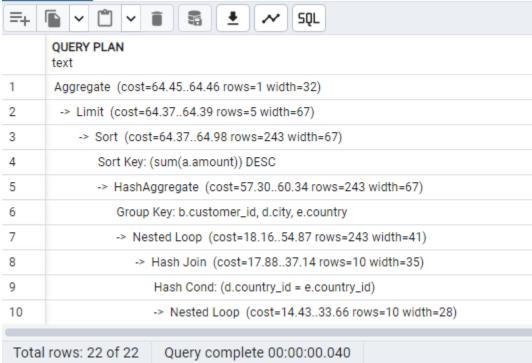
Query 1

CTE Plan



Subquery Plan

Data Output Messages Notifications



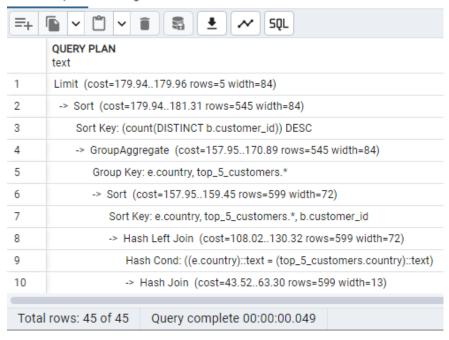
Real run times of these queries were the same as their query plans.

Results: Subquery is faster but cost is the same. The subquery is only 5 milliseconds faster. I am not surprised at this results. The subquery felt like it was adding an extra step in this instance so I was expecting the subquery to be faster.

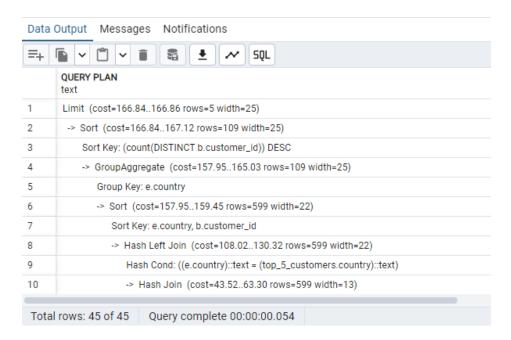
Query 2

Subquery Plan

Data Output Messages Notifications



CTE Plan



Real run times of these queries were the same as their query plans.

Results: The CTE cost was lower than the subquery however the time it took was higher, this was surprising as you would think something taking longer would equate to more cost.

I was not surprised that the subqueries were faster as they have less clauses, and lines of code to interpret.

Step 3:

For step 1, transforming the query into one with a CTE was easy to understand. You are renaming the inner query with a table name then searching within that table name for the from clause in the outer query.

Fore step 2, it was much harder. The amount of joins require accuracy and for one to be meticulous in which table they are selecting what from and joining where.

On reflection, I could have taken this exercise much differently but I also think that depends on how I wrote my previous exercises queries.

I could have written out CTEs for all steps in my query, that way they could be used in other queries if needed. An example would be if I wrote CTEs for the top country, top city and total amount paid, this would have occurred had I not already found the top 10 cities in a previous query.