Career Foundry

Data Analytics Immersion

A3.E8

Kendra Jackson

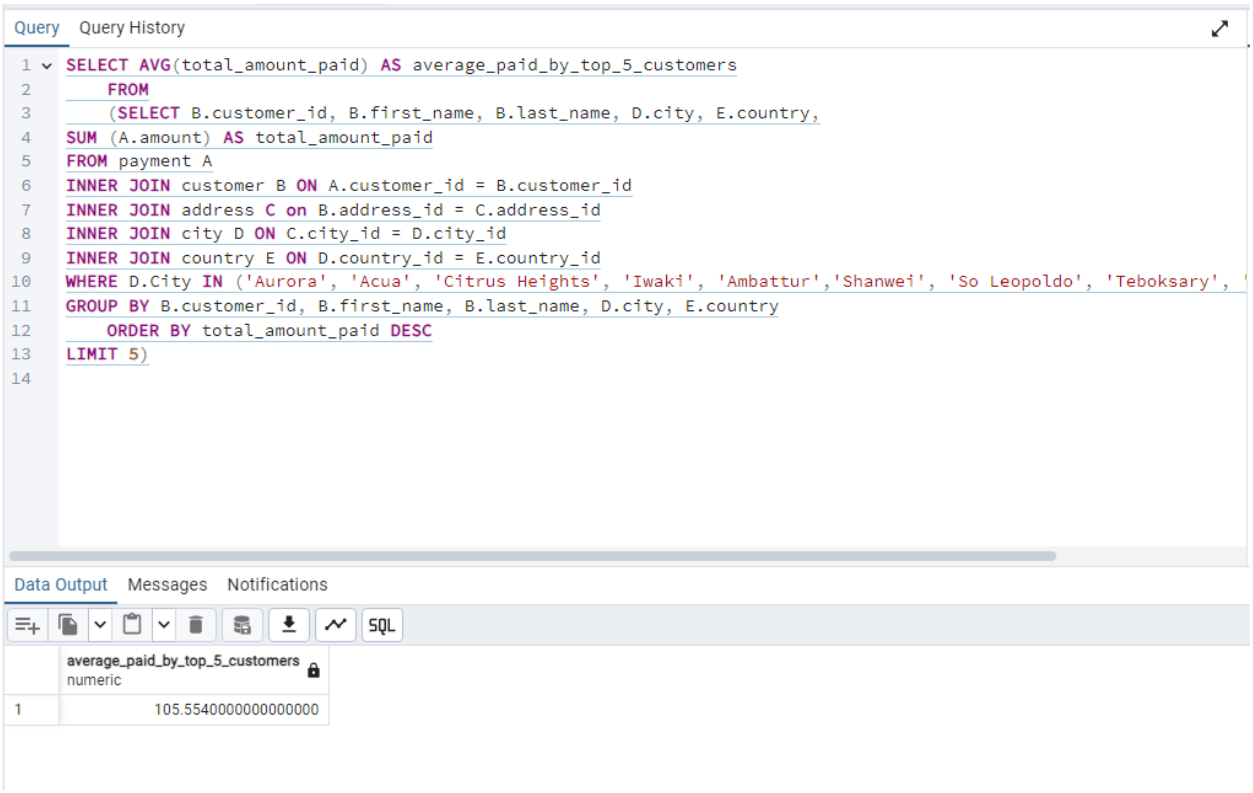
Step 1:

Inner Query:

```
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
```

Outer Query:

```
SELECT AVG(total_amount_paid) AS average_paid_by_top_5_customers

FROM (Inner Query)
```



Step 2:

Inner Query:

(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) AS top_5_customers

Outer Query

SELECT E.country,

COUNT(DISTINCT B.customer_id) AS all_customer_count

COUNT(DISTINCT E.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

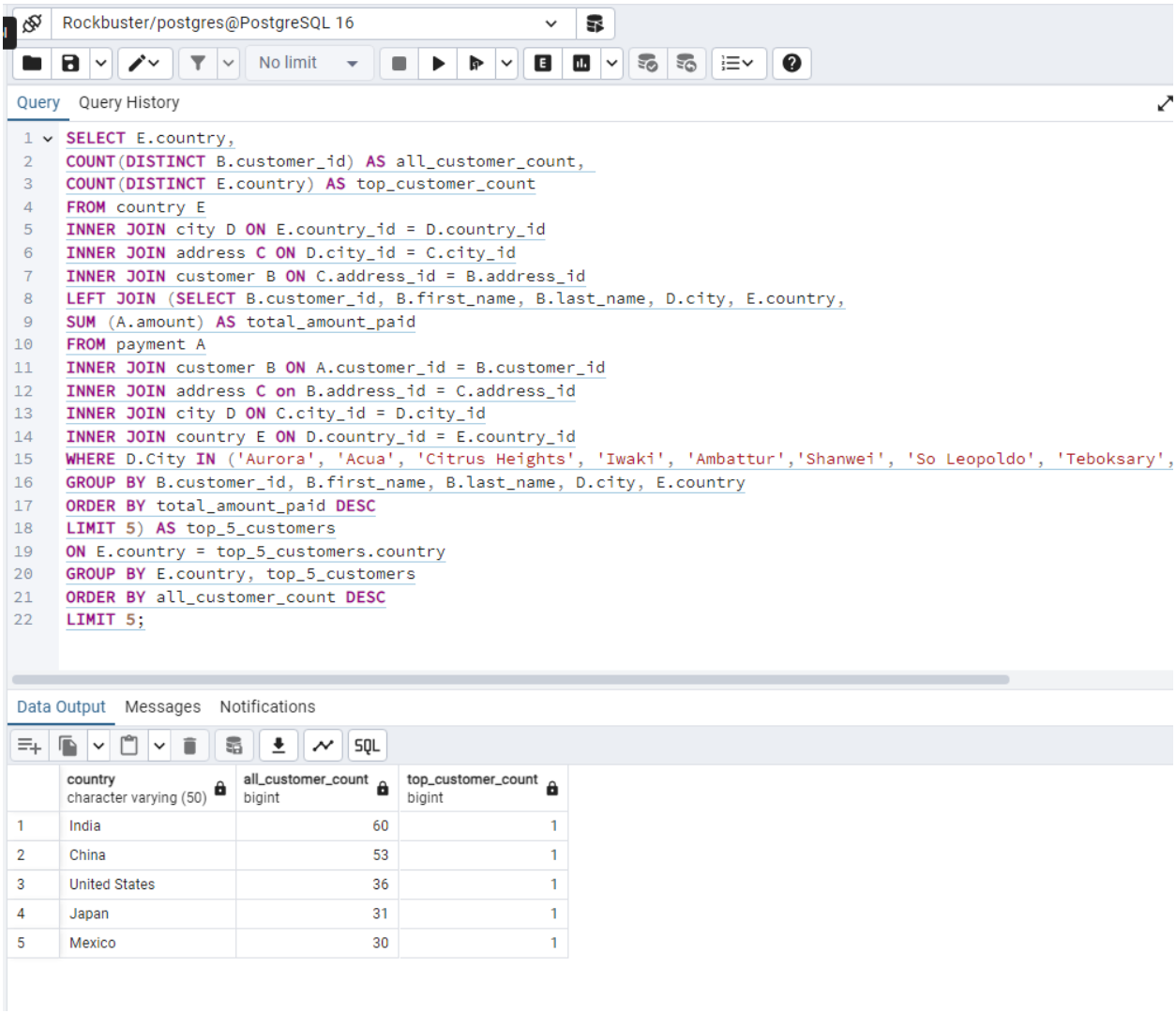
INNER JOIN customer B ON C.address_id = B.address_id

LEFT JOIN (inner statement) ON E.country = top_5_customers.country

GROUP BY E.country, top_5_customers

ORDER BY all_customer_count DESC

LIMIT 5;



Step 3:

Having limited experience with SQL, I am not quite sure if these queries could be performed without using a subquery. That being said, it feels like using these specific subqueries is a very long-winded process in order to find the data we are looking for. In this instance the queries seemed very large and were difficult to follow and read, not to mention write, due to the many tables being joined and all the aliases being used. Hopefully there is an easier way to perform these queries using a CTE function.

Potentially query 1 may be able to be performed using the WHERE or HAVING function if we had already written the “inner query” and had the results. We could take the results and query for the average of results having a person’s total.

```
SELECT AVG(amount)

FROM payment

WHERE customer_id IN ('225', '424', '240', '486', '537')
```

This, however, gives us the average per payment not the total. We could use a simpler nested query, however.

```
SELECT AVG(total_paid) AS top_customer_avg_payment

FROM (SELECT customer_id,

SUM(amount) AS total_paid

FROM payment

WHERE customer_id IN ('225', '424', '240', '486', '537')

GROUP BY customer_id)

AS customer_totals
```

As discussed in the exercise, subqueries are typically used for short code that can be fed into other queries to account for changing data. It somehow seems easier to potentially run two separate queries rather than one query with a long subquery.