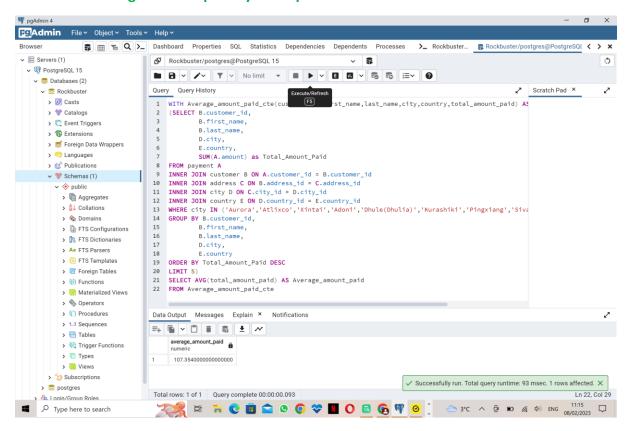
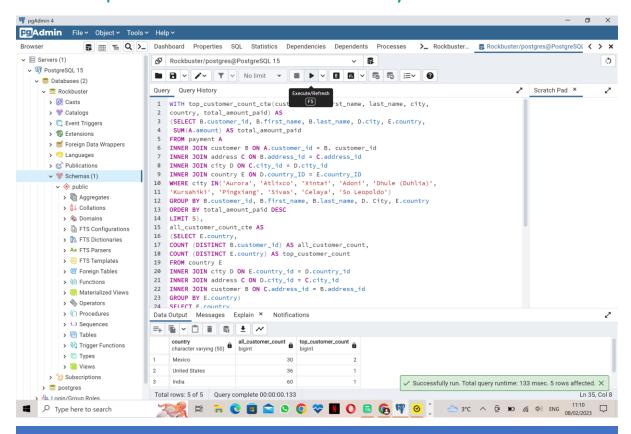
1. The average amount paid by the top 5 customers.



2. The top 5 customers are based within each country.



```
WITH top_customer_count_cte(customer_id, first_name, last_name, city,
country, total_amount_paid) 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 city IN('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Duhlia)',
'Kursahiki', 'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
GROUP BY B.customer_id, B.first_name, B.last_name, D. City, E.country
ORDER BY total_amount_paid DESC
LIMIT 5),
all_customer_count_cte AS
(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
GROUP BY E.country)
SELECT E.country,
COUNT (DISTINCT B.customer id) AS all customer count,
COUNT (DISTINCT top_customer_count_cte.customer_id) 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

top_customer_count_cte ON E.country = top_customer_count_cte.country

GROUP BY E.country

ORDER BY top_customer_count DESC

LIMIT 5

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

First step: was to identify the relevant tables from our ERD so that we could work out the pattern we want to connect.

Second Step: copying the subqueries from previous task

Third Step: replacing the subqueries with CTE's using WITH at the beginning of the query and providing AS keyword.

Fourth Step: This was an additional step for query 2 where we wrote two CTE's one for top customer count and the other for all customer count.

Fifth Step: Finally after the CTE came the main statement. (as divided by the color scheme above)

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

- 1. Which approach do you think will perform better and why?
 - a. I feel using CTE is much better as these are used right at the beginning of the query and improves readability.
- 2. The EXPLAIN command gives you an estimated cost. To find out the actual speed of your queries.

a. CTE 1: 299 msec

b. Sub query 1: 176 msec

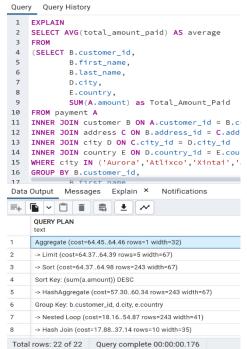
c. CTE 2: 148 msec

d. Sub query 2: 475 msec

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

SUB

SUB QUERY 1



DV 2

QUERY 2



CTE 1

```
Query Query History
 1 EXPLAIN
      WITH Average_amount_paid_cte(customer_id,first_name,last_name
      (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
10
11
     INNER JOIN city D ON C.city_id = D.city_id
     HNRER JOIN COUNTY E ON D.COUNTY_Id = D.CITY_Id

WHERE city IN ('Aurora','Atlixco','Xintai','Adoni','Dhule(Dhu
     GROUP BY B.customer id.
Data Output Messages Explain * Notifications
     OUERY PLAN
      Aggregate (cost=64.45..64.46 rows=1 width=32)
      -> Limit (cost=64.37..64.39 rows=5 width=67)
      -> Sort (cost=64.37..64.98 rows=243 width=67)
      Sort Key: (sum(a.amount)) DESC
      -> HashAggregate (cost=57.30..60.34 rows=243 width=67)
      Group Key; b.customer_id, d.city, e.country
      -> Nested Loop (cost=18.16, 54.87 rows=243 width=41)
      -> Hash Join (cost=17.88.37.14 rows=10 width=35)
Total rows: 22 of 22 Query complete 00:00:00.299
```

CTE 2

```
Query Query History
1
     WITH top_customer_count_cte(customer_id, first_
     country, total_amount_paid) AS
 4
    (SELECT B.customer_id, B.first_name, B.last_nam
      SUM(A.amount) AS total_amount_paid
     FROM payment A
     INNER JOIN customer B ON A.customer_id = B. cus
    INNER JOIN address C ON B.address_id = C.addres
    INNER JOIN city D ON C.city_id = D.city_id
10
    INNER JOIN country E ON D.country_ID = E.countr
    WHERE city IN('Aurora', 'Atlixco', 'Xintai', 'A'
'Kursahiki', 'Pingxiang', 'Sivas', 'Celaya', 'S
11
12
    GROUP BY B.customer_id, B.first_name, B.last_na
13
    ORDER BY total_amount_paid DESC
14
15
    LIMIT 5).
   all customer count cte AS
16
     (SFLECT F countr
Data Output Messages Explain ×
    QUERY PLAN
     Limit (cost=166.23..166.24 rows=5 width=25)
      -> Sort (cost=166.23..166.50 rows=109 width=25)
      Sort Key: (count(DISTINCT top_customer_count_cte.customer_id)) DESC
      -> GroupAggregate (cost=155.43..164.42 rows=109 width=25)
     Group Key: e.country
      -> Merge Left Join (cost=155.43..158.83 rows=599 width=17)
      Merge Cond: ((e.country)::text = (top_customer_count_cte.country)::text)
      -> Sort (cost=90.94..92.44 rows=599 width=13)
Total rows: 46 of 46 Query complete 00:00:00.148
```

4. Did the results surprise you? Write a few sentences to explain your answer. Yes indeed, I am. The cost from the CTEs are substantially different as well for the subqueries. Subquery 2 costed thrice as much of the CTE which clearlz shows how much better it is to use CTEs.

Step 3: Write 1 to 2 paragraphs on the challenges you faced when replacing your subqueries with CTEs.

The replacing of subquery 1 with CTE was comparatively much simple and understandable. But it was a big challenge understanding the Subquery 2 and as a result the CTE 2.