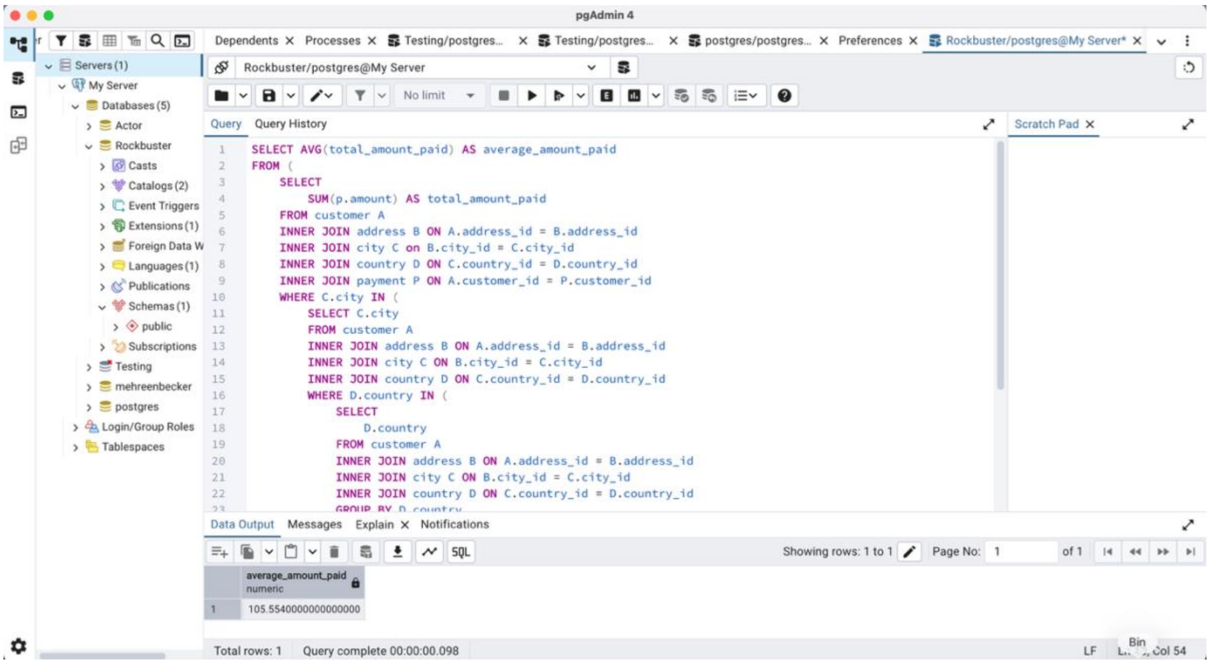
# Data Analytics Immersion
# 3.9: Common Table Expressions

*Mehreen Becker*

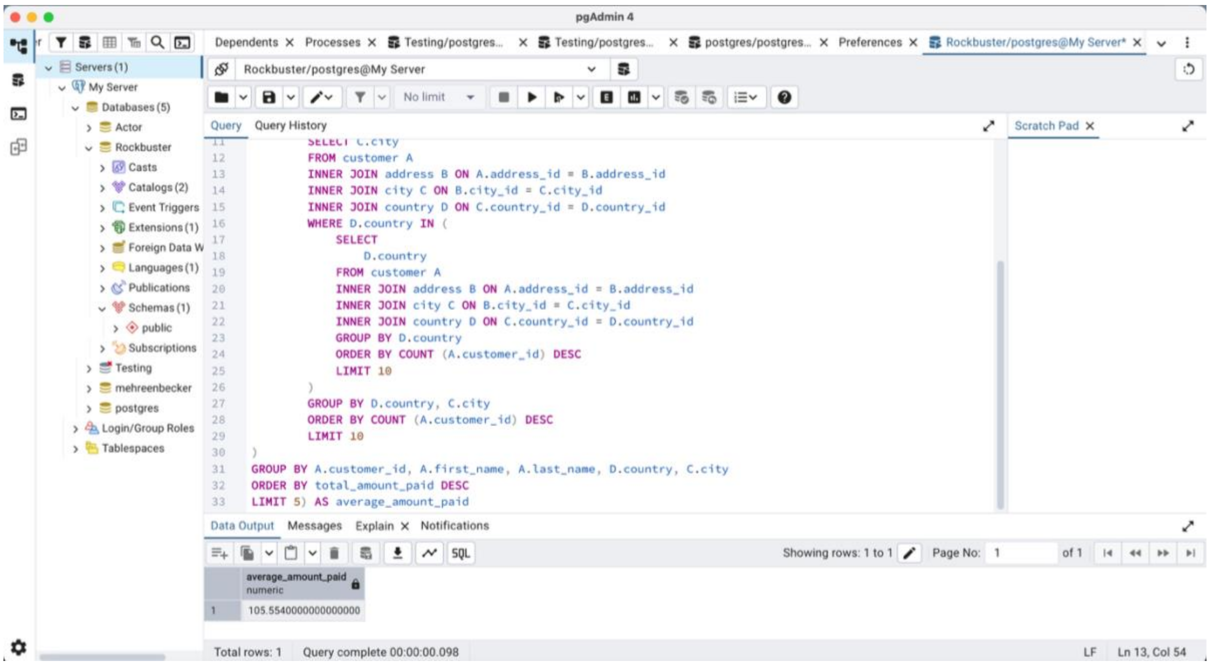
## Step 1: Answer the business questions from steps 1 and 2 of task 3.8 using CTEs

1. **Rewrite your queries from steps 1 and 2 of task 3.8 as CTEs.**
2. **Copy-paste your CTEs and their outputs into your answers document.**
3. **Write 2 to 3 sentences explaining how you approached this step, for example, what you did first, second, and so on.**



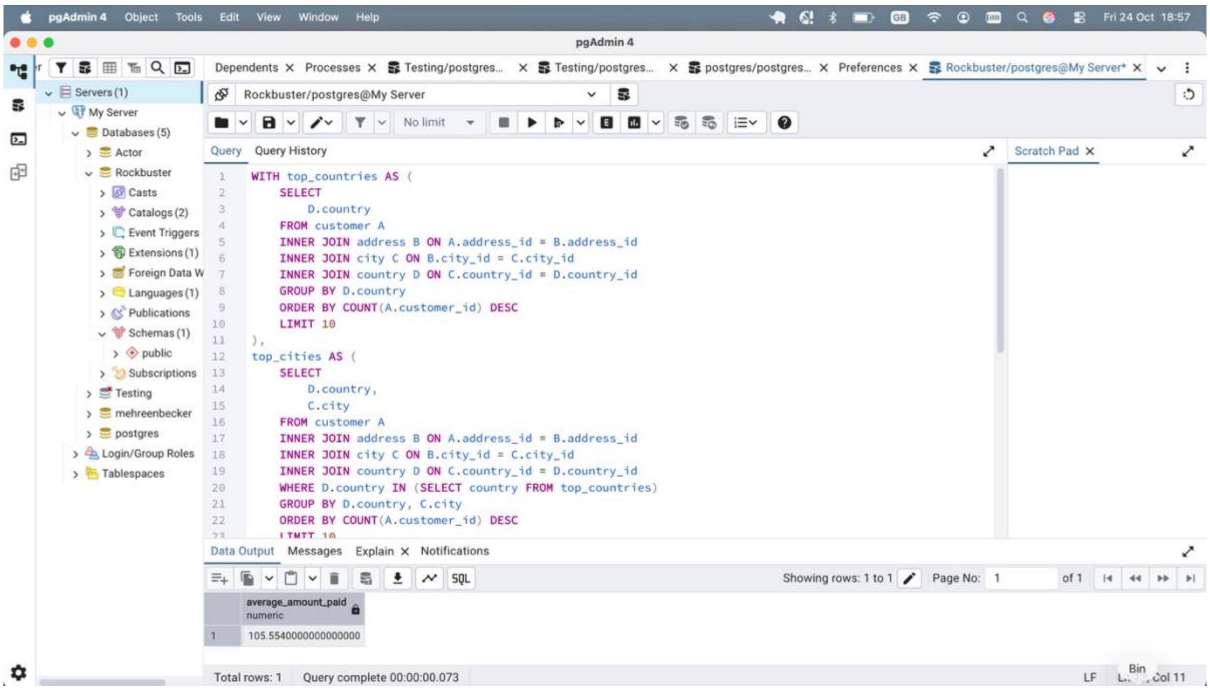
```sql
SELECT AVG(total_amount_paid) AS average_amount_paid
FROM (
    SELECT
        SUM(p.amount) AS total_amount_paid
    FROM customer A
    INNER JOIN address B ON A.address_id = B.address_id
    INNER JOIN city C on B.city_id = C.city_id
    INNER JOIN country D ON C.country_id = D.country_id
    INNER JOIN payment P ON A.customer_id = P.customer_id
    WHERE C.city IN (
        SELECT C.city
        FROM customer A
        INNER JOIN address B ON A.address_id = B.address_id
        INNER JOIN city C ON B.city_id = C.city_id
        INNER JOIN country D ON C.country_id = D.country_id
        WHERE D.country IN (
            SELECT
                D.country
            FROM customer A
            INNER JOIN address B ON A.address_id = B.address_id
            INNER JOIN city C ON B.city_id = C.city_id
            INNER JOIN country D ON C.country_id = D.country_id
            GROUP BY D.country
```

```
   average_amount_paid
   numeric
1  105.5540000000000000

Total rows: 1    Query complete 00:00:00.098
```



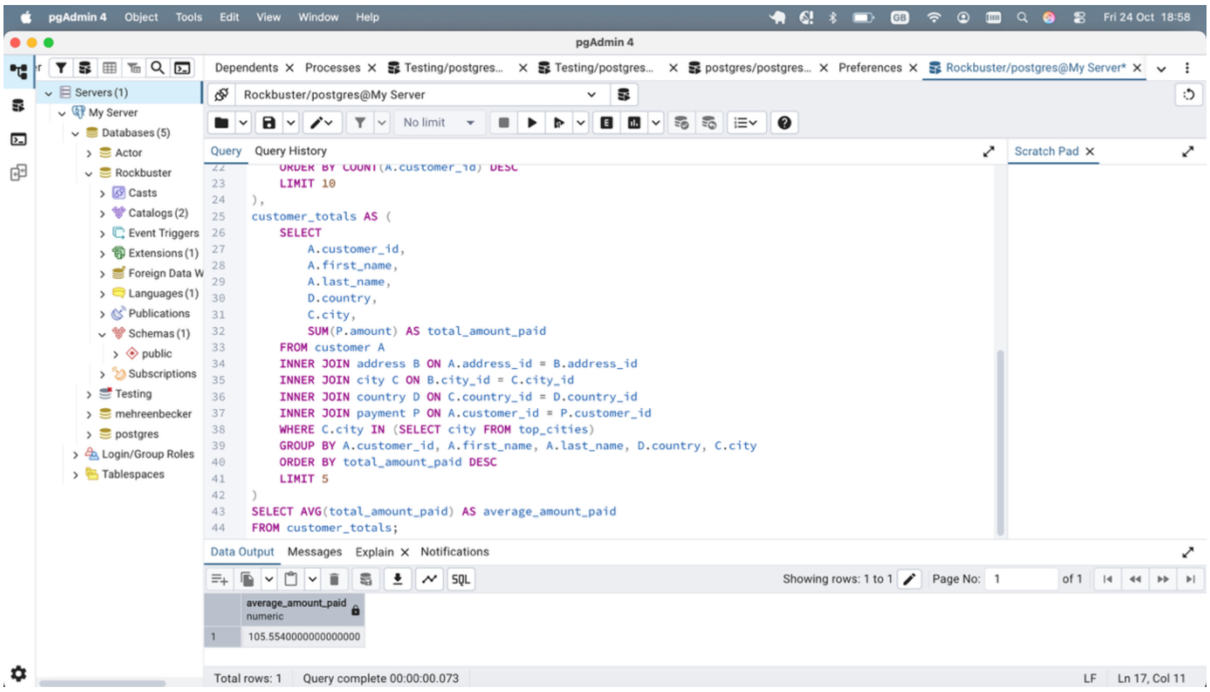
```sql
        SELECT C.city
        FROM customer A
        INNER JOIN address B ON A.address_id = B.address_id
        INNER JOIN city C ON B.city_id = C.city_id
        INNER JOIN country D ON C.country_id = D.country_id
        WHERE D.country IN (
            SELECT
                D.country
            FROM customer A
            INNER JOIN address B ON A.address_id = B.address_id
            INNER JOIN city C ON B.city_id = C.city_id
            INNER JOIN country D ON C.country_id = D.country_id
            GROUP BY D.country
            ORDER BY COUNT (A.customer_id) DESC
            LIMIT 10
        )
        GROUP BY D.country, C.city
        ORDER BY COUNT (A.customer_id) DESC
        LIMIT 10
)
GROUP BY A.customer_id, A.first_name, A.last_name, D.country, C.city
ORDER BY total_amount_paid DESC
LIMIT 5) AS average_amount_paid
```

```
   average_amount_paid
   numeric
1  105.5540000000000000

Total rows: 1    Query complete 00:00:00.098    LF    Ln 13, Col 54
```



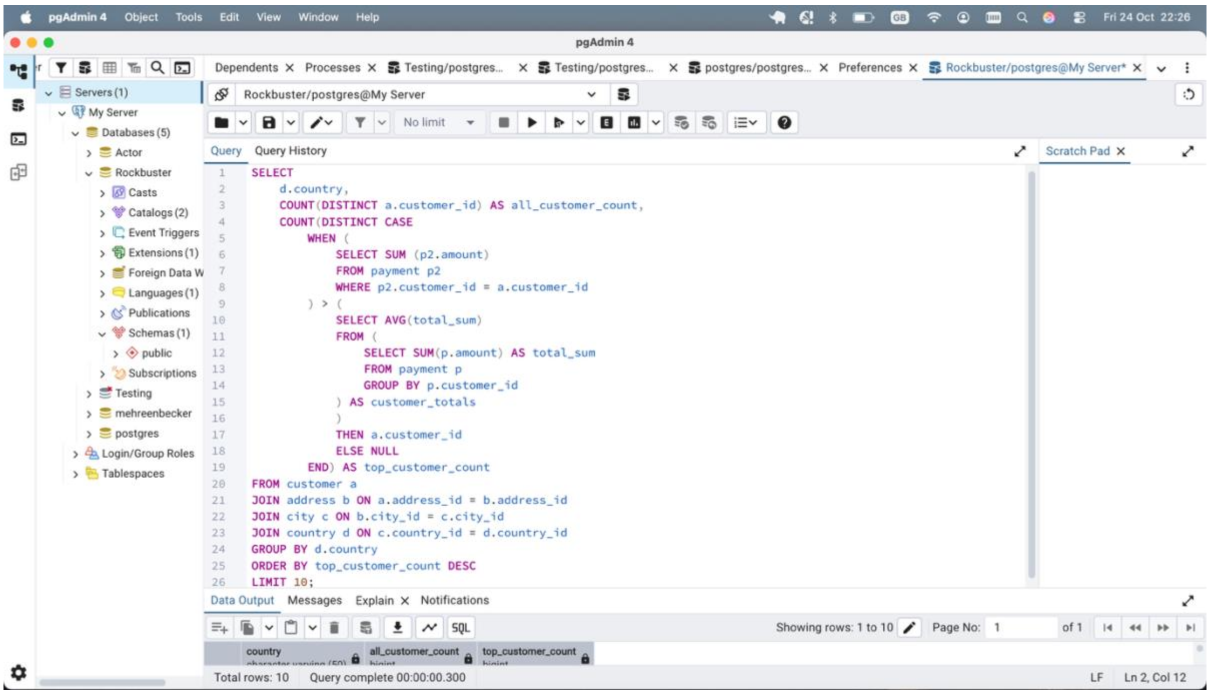
```sql
WITH top_countries AS (
    SELECT
        D.country
    FROM customer A
    INNER JOIN address B ON A.address_id = B.address_id
    INNER JOIN city C ON B.city_id = C.city_id
    INNER JOIN country D ON C.country_id = D.country_id
    GROUP BY D.country
    ORDER BY COUNT(A.customer_id) DESC
    LIMIT 10
),
top_cities AS (
    SELECT
        D.country,
        C.city
    FROM customer A
    INNER JOIN address B ON A.address_id = B.address_id
    INNER JOIN city C ON B.city_id = C.city_id
    INNER JOIN country D ON C.country_id = D.country_id
    WHERE D.country IN (SELECT country FROM top_countries)
    GROUP BY D.country, C.city
    ORDER BY COUNT(A.customer_id) DESC
    LIMIT 10
```

```
  average_amount_paid
  numeric
1 105.5540000000000000

Total rows: 1    Query complete 00:00:00.073
```



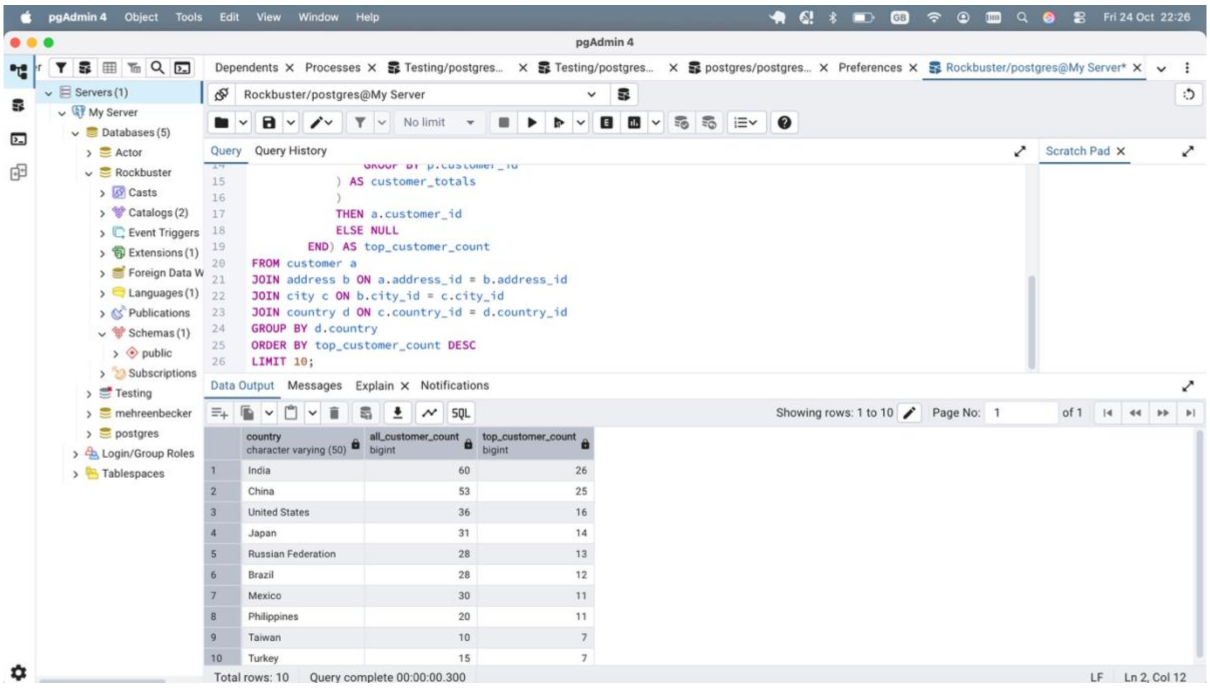
```sql
    ORDER BY COUNT(A.customer_id) DESC
    LIMIT 10
),
customer_totals AS (
    SELECT
        A.customer_id,
        A.first_name,
        A.last_name,
        D.country,
        C.city,
        SUM(P.amount) AS total_amount_paid
    FROM customer A
    INNER JOIN address B ON A.address_id = B.address_id
    INNER JOIN city C ON B.city_id = C.city_id
    INNER JOIN country D ON C.country_id = D.country_id
    INNER JOIN payment P ON A.customer_id = P.customer_id
    WHERE C.city IN (SELECT city FROM top_cities)
    GROUP BY A.customer_id, A.first_name, A.last_name, D.country, C.city
    ORDER BY total_amount_paid DESC
    LIMIT 5
)
SELECT AVG(total_amount_paid) AS average_amount_paid
FROM customer_totals;
```

```
  average_amount_paid
  numeric
1 105.5540000000000000

Total rows: 1    Query complete 00:00:00.073
```



```sql
SELECT
    d.country,
    COUNT(DISTINCT a.customer_id) AS all_customer_count,
    COUNT(DISTINCT CASE
        WHEN (
            SELECT SUM (p2.amount)
            FROM payment p2
            WHERE p2.customer_id = a.customer_id
        ) > (
            SELECT AVG(total_sum)
            FROM (
                SELECT SUM(p.amount) AS total_sum
                FROM payment p
                GROUP BY p.customer_id
            ) AS customer_totals
            )
            THEN a.customer_id
            ELSE NULL
        END) AS top_customer_count
FROM customer a
JOIN address b ON a.address_id = b.address_id
JOIN city c ON b.city_id = c.city_id
JOIN country d ON c.country_id = d.country_id
GROUP BY d.country
ORDER BY top_customer_count DESC
LIMIT 10;
```

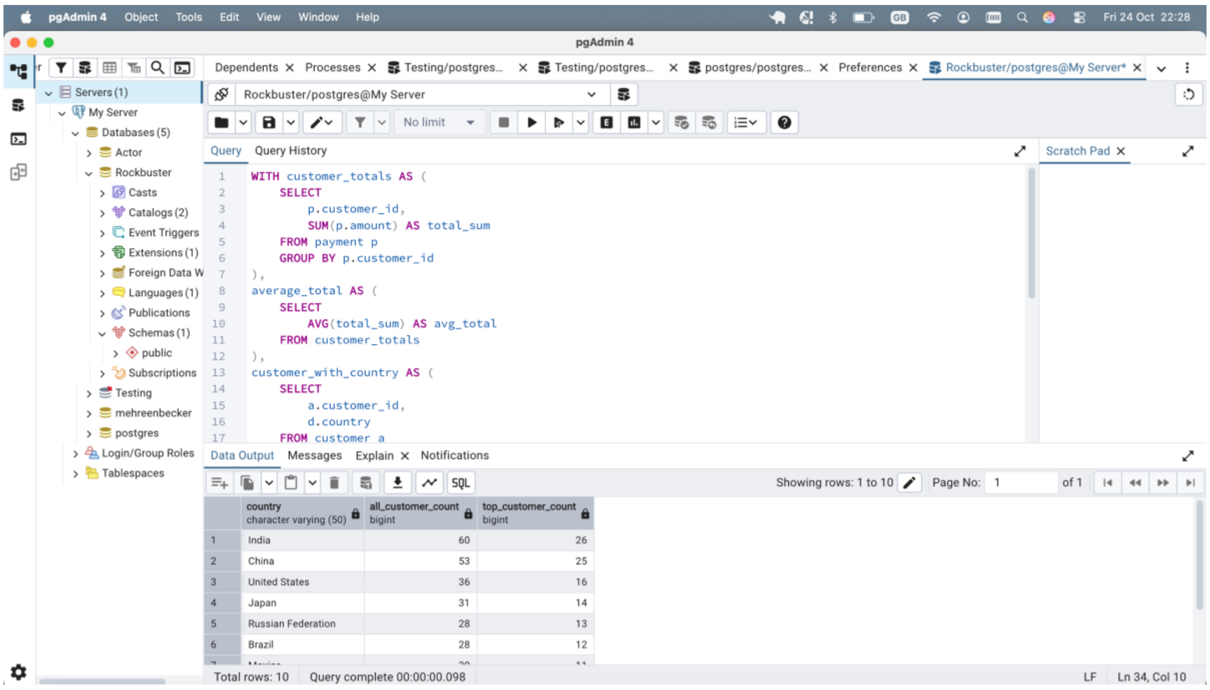
Total rows: 10    Query complete 00:00:00.300



```sql
            ) AS customer_totals
            )
            THEN a.customer_id
            ELSE NULL
        END) AS top_customer_count
FROM customer a
JOIN address b ON a.address_id = b.address_id
JOIN city c ON b.city_id = c.city_id
JOIN country d ON c.country_id = d.country_id
GROUP BY d.country
ORDER BY top_customer_count DESC
LIMIT 10;
```

| | country<br>character varying (50) | all_customer_count<br>bigint | top_customer_count<br>bigint |
|---|---|---|---|
| 1 | India | 60 | 26 |
| 2 | China | 53 | 25 |
| 3 | United States | 36 | 16 |
| 4 | Japan | 31 | 14 |
| 5 | Russian Federation | 28 | 13 |
| 6 | Brazil | 28 | 12 |
| 7 | Mexico | 30 | 11 |
| 8 | Philippines | 20 | 11 |
| 9 | Taiwan | 10 | 7 |
| 10 | Turkey | 15 | 7 |

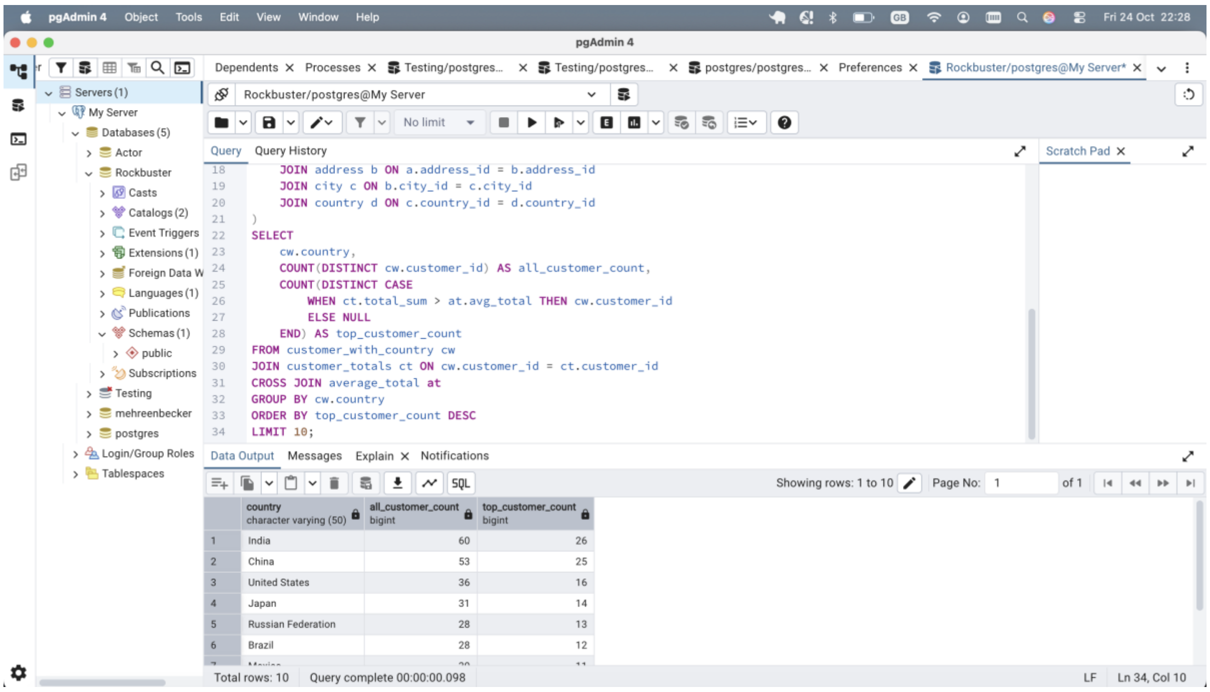
Total rows: 10    Query complete 00:00:00.300



```
WITH customer_totals AS (
    SELECT
        p.customer_id,
        SUM(p.amount) AS total_sum
    FROM payment p
    GROUP BY p.customer_id
),
average_total AS (
    SELECT
        AVG(total_sum) AS avg_total
    FROM customer_totals
),
customer_with_country AS (
    SELECT
        a.customer_id,
        d.country
    FROM customer a
```

| | country character varying (50) | all_customer_count bigint | top_customer_count bigint |
|---|---|---|---|
| 1 | India | 60 | 26 |
| 2 | China | 53 | 25 |
| 3 | United States | 36 | 16 |
| 4 | Japan | 31 | 14 |
| 5 | Russian Federation | 28 | 13 |
| 6 | Brazil | 28 | 12 |

Total rows: 10    Query complete 00:00:00.098



```
    JOIN address b ON a.address_id = b.address_id
    JOIN city c ON b.city_id = c.city_id
    JOIN country d ON c.country_id = d.country_id
)
SELECT
    cw.country,
    COUNT(DISTINCT cw.customer_id) AS all_customer_count,
    COUNT(DISTINCT CASE
        WHEN ct.total_sum > at.avg_total THEN cw.customer_id
        ELSE NULL
    END) AS top_customer_count
FROM customer_with_country cw
JOIN customer_totals ct ON cw.customer_id = ct.customer_id
CROSS JOIN average_total at
GROUP BY cw.country
ORDER BY top_customer_count DESC
LIMIT 10;
```

| | country character varying (50) | all_customer_count bigint | top_customer_count bigint |
|---|---|---|---|
| 1 | India | 60 | 26 |
| 2 | China | 53 | 25 |
| 3 | United States | 36 | 16 |
| 4 | Japan | 31 | 14 |
| 5 | Russian Federation | 28 | 13 |
| 6 | Brazil | 28 | 12 |

Total rows: 10    Query complete 00:00:00.098

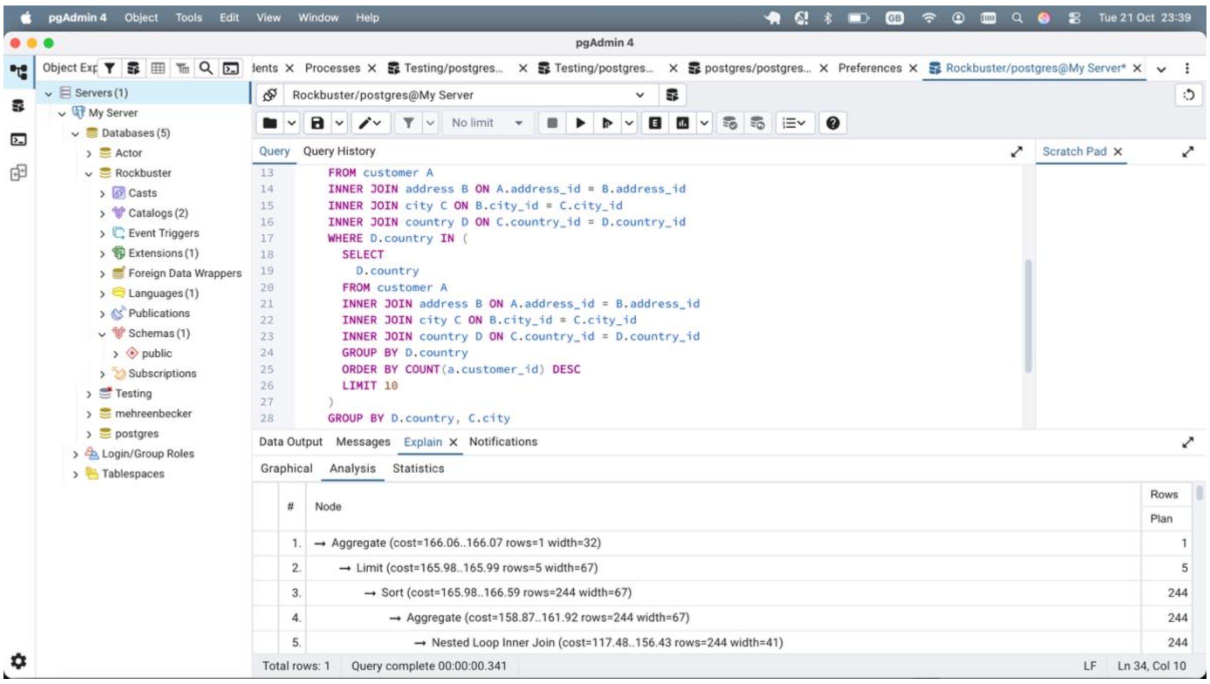
I first identified the top countries with the most customers by joining the customer, address, city and country tables and grouping them by country. Then, I narrowed the focus to the top 10 cities located within those countries, again by counting customers per city and ordering by that count. After that, I calculated information about the top 5 customers who paid the most using the payment table joined to the geographic data, and limited it to the customers from the top city. Finally I summarized the results in one query by find the average of the top 5 total payments, and in the other by comparing counts of all customers versus top customers per country.

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

1. **Which approach do you think will perform better and why?** I'm sure that both CTEs and subqueries produce the same, if not a similar, result. With my limited experience with CTEs, I feel like subqueries allow for better optimization potentially and I also felt like I had to scan fewer rows.

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

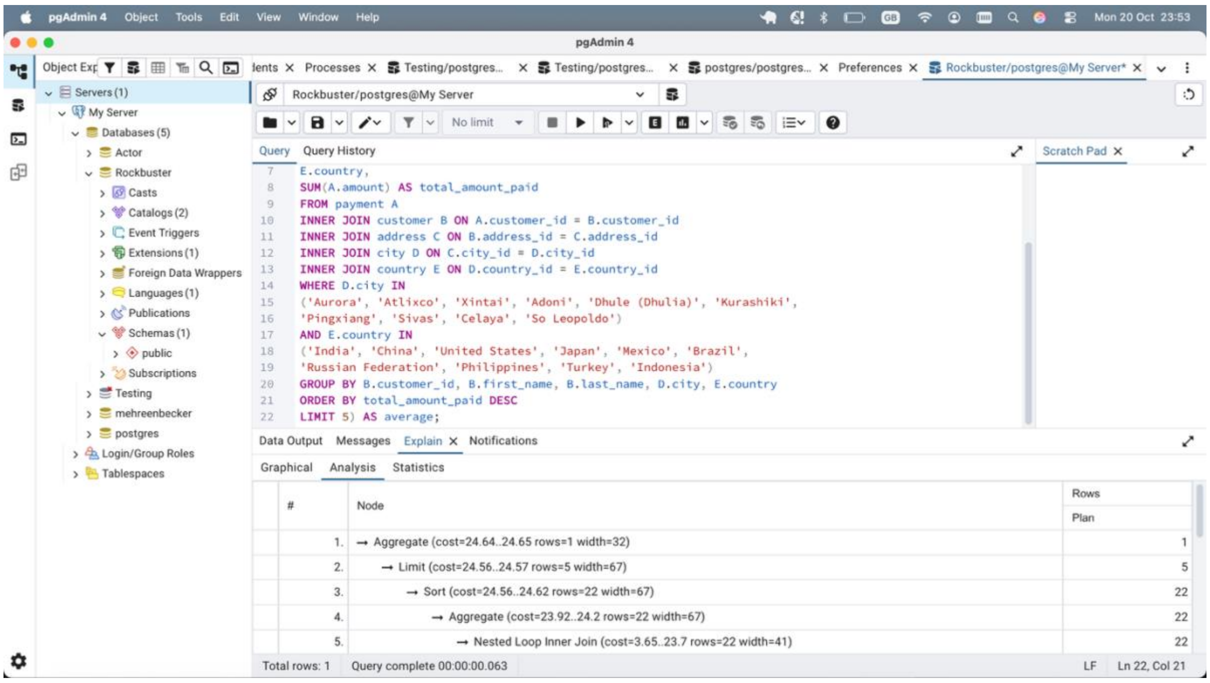
Difference in cost:



```sql
    FROM customer A
    INNER JOIN address B ON A.address_id = B.address_id
    INNER JOIN city C ON B.city_id = C.city_id
    INNER JOIN country D ON C.country_id = D.country_id
    WHERE D.country IN (
      SELECT
        D.country
      FROM customer A
      INNER JOIN address B ON A.address_id = B.address_id
      INNER JOIN city C ON B.city_id = C.city_id
      INNER JOIN country D ON C.country_id = D.country_id
      GROUP BY D.country
      ORDER BY COUNT(a.customer_id) DESC
      LIMIT 10
    )
    GROUP BY D.country, C.city
```

```
#   Node                                                                          Rows Plan
1.  → Aggregate (cost=166.06..166.07 rows=1 width=32)                                     1
2.    → Limit (cost=165.98..165.99 rows=5 width=67)                                       5
3.      → Sort (cost=165.98..166.59 rows=244 width=67)                                  244
4.        → Aggregate (cost=158.87..161.92 rows=244 width=67)                           244
5.          → Nested Loop Inner Join (cost=117.48..156.43 rows=244 width=41)            244
Total rows: 1    Query complete 00:00:00.341
```

*Figure 1: Cost (CTE)*



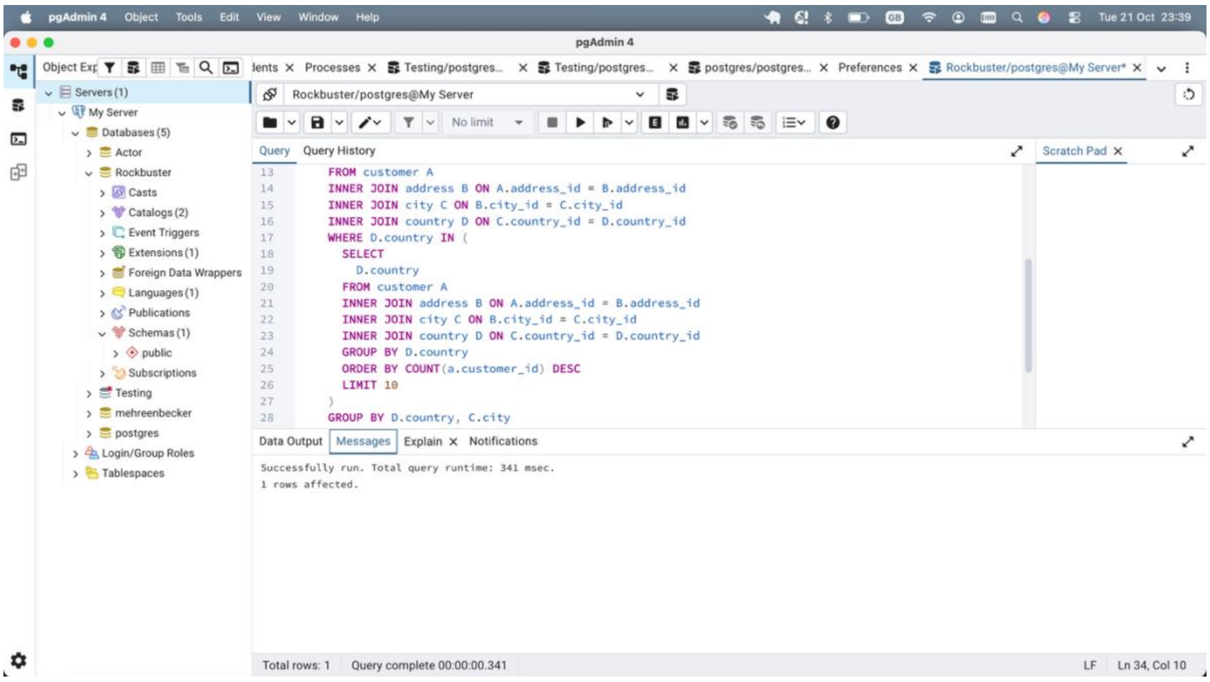
```sql
E.country,
SUM(A.amount) AS total_amount_paid
FROM payment A
INNER JOIN customer B ON A.customer_id = B.customer_id
INNER JOIN address C ON B.address_id = C.address_id
INNER JOIN city D ON C.city_id = D.city_id
INNER JOIN country E ON D.country_id = E.country_id
WHERE D.city IN
('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)', 'Kurashiki',
'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
AND E.country IN
('India', 'China', 'United States', 'Japan', 'Mexico', 'Brazil',
'Russian Federation', 'Philippines', 'Turkey', 'Indonesia')
GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS average;
```

```
#   Node                                                                      Rows Plan
1.  → Aggregate (cost=24.64..24.65 rows=1 width=32)                                   1
2.    → Limit (cost=24.56..24.57 rows=5 width=67)                                     5
3.      → Sort (cost=24.56..24.62 rows=22 width=67)                                  22
4.        → Aggregate (cost=23.92..24.2 rows=22 width=67)                            22
5.          → Nested Loop Inner Join (cost=3.65..23.7 rows=22 width=41)              22
Total rows: 1    Query complete 00:00:00.063
```

*Figure 2: Cost (Subquery)*

3. **The EXPLAIN command gives you an *estimated* cost. To find out the actual speed of your queries, run them in pgAdmin 4. After you've run each query, a popup window will display its speed in milliseconds.**

Difference in query run time:

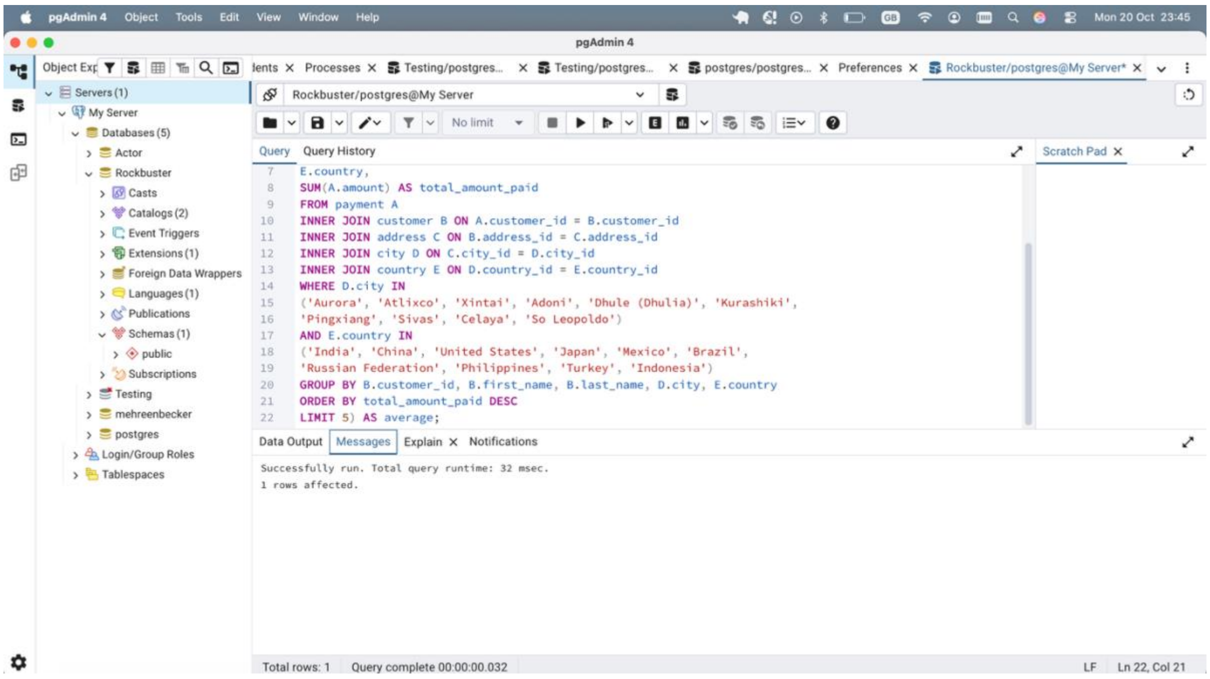


```
    FROM customer A
    INNER JOIN address B ON A.address_id = B.address_id
    INNER JOIN city C ON B.city_id = C.city_id
    INNER JOIN country D ON C.country_id = D.country_id
    WHERE D.country IN (
      SELECT
        D.country
      FROM customer A
      INNER JOIN address B ON A.address_id = B.address_id
      INNER JOIN city C ON B.city_id = C.city_id
      INNER JOIN country D ON C.country_id = D.country_id
      GROUP BY D.country
      ORDER BY COUNT(a.customer_id) DESC
      LIMIT 10
    )
    GROUP BY D.country, C.city
```

```
Successfully run. Total query runtime: 341 msec.
1 rows affected.
```

```
Total rows: 1    Query complete 00:00:00.341
```

*Figure 1: Total Query Runtime: 341 msec (CTE)*



```
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', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)', 'Kurashiki',
'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
AND E.country IN
('India', 'China', 'United States', 'Japan', 'Mexico', 'Brazil',
'Russian Federation', 'Philippines', 'Turkey', 'Indonesia')
GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS average;
```

```
Successfully run. Total query runtime: 32 msec.
1 rows affected.
```

```
Total rows: 1    Query complete 00:00:00.032
```

*Figure 2: Total Query Runtime: 32 msec (Subquery)*

4. **Did the results surprise you?** Write a few sentences to explain your answer. To be honest, no. I did think that subqueries would run faster (not that I noticed the few milisecond difference) but more because they feel more straightforward and less complex.

**Step 3: Write 1 to 2 paragraphs on the challenges you faced when replacing your subqueries with CTEs.** Since we were asked to use CTEs and not subqueries, one of the main challenges for me was to make sure that everything made sense. I did find myself feeling a little lost and confused since it's easier with CTEs to misplace filters and to also join things together. I found myself scanning rows multiple times to ensure that the string made sense and that everything was connected with each other and connected in a way that it should be.