#### **Data Immersion**

#### Exercise 3.9

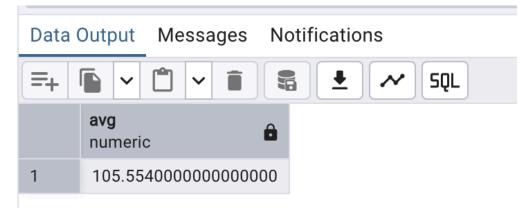
#### Common Table Expressions

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

#### Query:

```
Welcome 🕏 Rockbuster/postgr... x 🕏 Rockbuster/postgr... x 🕏 Rockbuster/postgres@PostgreSQL 17* x
-
        Rockbuster/postgres@PostgreSQL 17
3
                     ▼ ∨ No limit
                                        E
                                                          >_
    Query Query History
FFFF
    1 ➤ With Top_Ten_Countries AS
        (select cntry.country
             from customer cust
            inner join address addr on cust.address_id = addr.address_id
            inner join city city on addr.city_id = city.city_id
    6
            inner join country cntry on city.country_id = cntry.country_id
            group by cntry.country
            order by count(cust.customer_id) desc limit 10),
    8
    9 Top_Ten_Cities AS
    10 (SELECT city.city
    11
             FROM customer cust
    12
            INNER JOIN address addr ON cust.address_id = addr.address_id
    13
            INNER JOIN city city ON addr.city_id = city.city_id
    14
            INNER JOIN country cntry ON city.country_id = cntry.country_id
    15
            where cntry.country IN
    16
                (select * from Top_Ten_Countries)
    17
            group by cntry.country, city.city
            order by count(cust.customer_id) desc limit 10),
    18
    19 Top_Five_Customer AS
    20 (SELECT SUM(pay.amount) AS total_amount
    21
            FROM payment pay
    22
            INNER JOIN customer cust ON pay.customer_id = cust.customer_id
    23
            INNER JOIN address addr ON cust.address_id = addr.address_id
    24
            INNER JOIN city city ON addr.city_id = city.city_id
    25
            INNER JOIN country cntry ON city.country_id = cntry.country_id
    26
            where city.city in
    27
               ( select * from Top_Ten_Cities)
             group by cust.first_name, cust.last_name, city.city, cntry.country order by total_amount desc limit 5)
    28
    29
        select avg(total_amount)
       from Top_Five_Customer
    30
```

## **Output:**

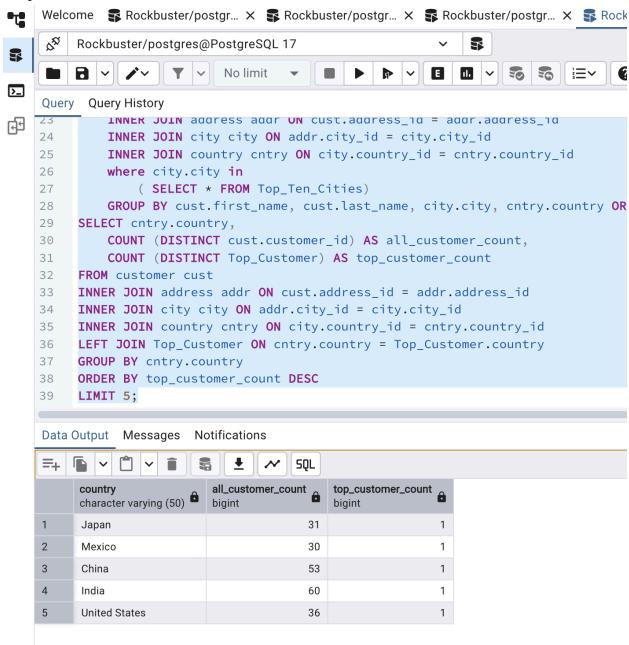


# Step 1b: How many of the top five customers identified in step one are based in each country? Query:

```
With Top Ten Countries AS
(SELECT entry.country
      from customer cust
      INNER JOIN address addr on cust.address id = addr.address id
      INNER JOIN city city on addr.city id = city.city id
      INNER JOIN country entry on city.country id = entry.country id
      GROUP BY entry.country
      ORDER BY COUNT (cust.customer id) DESC LIMIT 10),
Top Ten Cities AS
(SELECT city.city
      FROM customer cust
      INNER JOIN address addr ON cust.address id = addr.address id
      INNER JOIN city city ON addr.city id = city.city id
      INNER JOIN country entry ON city.country id = entry.country id
      WHERE entry.country IN
             (SELECT * FROM Top Ten Countries)
      GROUP BY entry.country, city.city
      ORDER BY COUNT (cust.customer id) DESC LIMIT 10),
Top Customer AS
(SELECT SUM(pay.amount) AS total amount, cntry.country
      FROM payment pay
      INNER JOIN customer cust ON pay.customer id = cust.customer id
      INNER JOIN address addr ON cust.address id = addr.address id
      INNER JOIN city city ON addr.city id = city.city id
      INNER JOIN country entry ON city.country id = entry.country id
      where city.city in
             ( SELECT * FROM Top Ten Cities)
      GROUP BY cust.first name, cust.last name, city.city, entry.country ORDER BY
total amount DESC LIMIT 5)
SELECT entry.country,
      COUNT (DISTINCT cust.customer id) AS all customer count,
      COUNT (DISTINCT Top Customer) AS top customer count
FROM customer cust
INNER JOIN address addr ON cust.address id = addr.address id
INNER JOIN city city ON addr.city id = city.city id
INNER JOIN country entry ON city.country id = entry.country id
LEFT JOIN Top Customer ON cntry.country = Top Customer.country
GROUP BY entry.country
```

ORDER BY top\_customer\_count DESC LIMIT 5;

### **Output:**



Write 2 to 3 sentences explaining how you approached this step

→ The idea was to rewrite the subqueries into CTE to find the top ten countries with the most of Rockbuster's customers. I began the query with the WITH clause and gave the name "Top Ten Countries". Then, I used AS followed by the query to make sure that I created a table with the top ten countries. I used the same process for top ten cities and top five customers.

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

- 1. Which approach do you think will perform better and why?
  - a. I think the CTE approach will perform better because of the length of the query. It's also easier to read and understand, both for myself and those who are from different industries. It also helps eliminate repetitive logic, making the query more efficient.
- 2. Compare the costs of all the queries by creating query plans for each one. 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.

Query Type:	СТЕ	Subquery
Query 1	Cost:170.21	Cost: 127.90
Query 1	Speed:71	Speed: 73
Query 2	Cost:272.60	Cost:271.90
Query 2	Speed:77	Speed:79

a.

- 3. Did the results surprise you? Write a few sentences to explain your answer.
  - a. The results are mostly similar, except for the first query. I was surprised by how much higher the cost of the CTE was compared to the subquery. In the second query, the costs are nearly the same. The execution speed is also quite similar, with the CTE being slightly faster than the subquery.

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

- One of the biggest challenges I faced when converting subqueries to CTEs was maintaining their structure while ensuring the results stayed the same. Troubleshooting and making adjustments took time since each subquery had to be adapted to the CTE format.