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.

Average amount paid by top 5 customers:

Query Query History

```
-- STEP 1 from task 3.8
WITH
top_10_countries AS (
    SELECT E.country
    FROM customer B
    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
    GROUP BY E.country
    ORDER BY COUNT(B.customer_id) DESC
    LIMIT 10
),
top_10_cities AS (
    SELECT D.city, E.country
    FROM customer B
    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
    INNER JOIN top_10_countries top10con ON E.country = top10con.country
    GROUP BY E.country, D.city
    ORDER BY COUNT(B.customer_id) DESC
    LIMIT 10
),
top_5_customers AS (
    SELECT A.customer_id, B.first_name, B.last_name, E.country, D.city,
           SUM(A.amount) AS total_amount
    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
    INNER JOIN top_10_cities top10cit ON D.city = top10cit.city AND E.country = top10cit.country
    GROUP BY A.customer_id, B.first_name, B.last_name, E.country, D.city
    ORDER BY total_amount DESC
    LIMIT 5
)
SELECT AVG (top_5_customers.total_amount) AS average
FROM top_5_customers;
```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1

	average numeric
1	105.5540000000000000

I began by creating the innermost subquery as a CTE to identify the top 10 countries. Then, I developed a second CTE to extract the top 10 cities within those countries, followed by a third CTE to determine the top 5 customers based on total payments. Finally, I used the 'top_5_customers' CTE to calculate the average value.

Top 5 customers within each country:

Query | Query History

```sql
-- STEP 2 from task 3.8
WITH
top_10_countries AS (                   -- top 10 countries CTE
    SELECT E.country
    FROM customer AS B
    INNER JOIN address AS C ON B.address_id = C.address_id
    INNER JOIN city AS D ON C.city_id = D.city_id
    INNER JOIN country AS E ON D.country_id = E.country_id
    GROUP BY E.country
    ORDER BY COUNT(B.customer_id) DESC
    LIMIT 10),
top_10_cities AS (                      -- top 10 cities CTE
    SELECT D.city, E.country
    FROM customer AS B
    INNER JOIN address AS C ON B.address_id = C.address_id
    INNER JOIN city AS D ON C.city_id = D.city_id
    INNER JOIN country AS E ON D.country_id = E.country_id
    INNER JOIN top_10_countries top10con ON E.country = top10con.country
    GROUP BY E.country, D.city
    ORDER BY COUNT(B.customer_id) DESC
    LIMIT 10),
top_5_customers AS (                    -- top 10 customers CTE
    SELECT A.customer_id, B.first_name, B.last_name, E.country, D.city,
           SUM(A.amount) AS total_amount
    FROM payment AS A
    INNER JOIN customer AS B ON A.customer_id = B.customer_id
    INNER JOIN address AS C on B.address_id = C.address_id
    INNER JOIN city AS D ON C.city_id = D.city_id
    INNER JOIN country AS E on D.country_id = E.country_id
    INNER JOIN top_10_cities top10cit ON D.city = top10cit.city AND E.country = top10cit.country
    GROUP BY A.customer_id, B.first_name, B.last_name, E.country, D.city
    ORDER BY total_amount DESC
    LIMIT 5)
SELECT E.country,                       -- recreating table-chain customer --> country
       COUNT(DISTINCT B.customer_id) AS all_customer_count,
       COUNT(DISTINCT top_5_customers.customer_id) AS top_customer_count
FROM customer B
INNER JOIN address AS C ON B.address_id = C.address_id
INNER JOIN city AS D ON C.city_id = D.city_id
INNER JOIN country AS E ON D.country_id = E.country_id
LEFT JOIN top_5_customers ON E.country = top_5_customers.country
GROUP BY 1
ORDER BY top_customer_count DESC,
         all_customer_count DESC
LIMIT 5;
```

Data Output | Messages | Notifications

Showing rows: 1 to 5 | Page No: 1

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	India	60	1
2	China	53	1
3	United States	36	1
4	Japan	31	1
5	Mexico	30	1

Similar to the step 1, I built the query from the bottom up, first creating the CTE for top 10 countries counting customers per country, then CTE for top 10 cities within top 10 countries by joining with top 10 countries CTE and counting customers per city. And then I built CTE for top 5 customers by joining with top 10 cities CTE and summing the amounts.

Finally, I joined the tables chain "customer → country" so it can calculate all customers and compare against the top 5 customers. The LEFT JOIN ensures that the column "country" appears in the final table despite it doesn't appear in top 5 customer CTE.

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

1. Which approach do you think will perform better and why?
   CTEs make the query more modular and easier to understand by breaking it into logical steps-for example, top to countries → top 10 cities → top 5 customers. In the subquery version queries are nested in multiple layers, making them harder to understand and modify.

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

	Subquery version		***CTE***	
Query 1: *Average amount paid by top 5 customers*	*cost: 167.29*	*time:85ms*	*cost: 127.64*	*time: 103ms*
Query 2: *Top 5 customers within each country*	*cost: 269.68*	*time:188ms*	*cost: 229.84*	*time: 124ms*

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.
4. Did the results surprise you? Write a few sentences to explain your answer.
   In the query 1 subquery version was faster (85ms vs. 103ms) but had a higher cost (167.29 vs 127.64). I was surprised by this result, specially, regarding lower cost despite longer run time for CTE version, where I had expected CTE model as winner.
   In the query 2 CTE version is clearly the winner with both factors run time and cost, as expected.

Step 3:

Write 1 to 2 paragraphs on the challenges you faced when replacing your subqueries with CTEs.
During replacing subqueries with CTEs, it was a bit tricky to figure out, how to split them into separate steps with CTEs, because subqueries are written directly inside the main query. In CTE version each CTE must be given a unique name to ensure that the joins and filters worked correctly. Furthermore, in the second query it was tricky to recreate the table-chain the tables to return the right result.
Second, CTE version was not always faster as I expected. Despite their easier readability, CTEs don't always perform the same as subqueries. Overall, CTEs helped me understand and organize queries better.

Step 4:

Save your "Answers 3.9" document as a PDF and upload it here for your tutor to review.

QUERY STEP 1:

```
WITH
top_10_countries AS (
        SELECT E.country
        FROM customer B
        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
        GROUP BY E.country
        ORDER BY COUNT(B.customer_id) DESC
        LIMIT 10
),
top_10_cities AS (
        SELECT D.city, E.country
        FROM customer B
        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
        INNER JOIN top_10_countries top10con ON E.country = top10con.country
        GROUP BY E.country, D.city
        ORDER BY COUNT(B.customer_id) DESC
        LIMIT 10
),
top_5_customers AS (
        SELECT A.customer_id, B.first_name, B.last_name, E.country, D.city,
                SUM(A.amount) AS total_amount
        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
```

```
        INNER JOIN top_10_cities top10cit ON D.city = top10cit.city AND E.country =
top10cit.country
        GROUP BY A.customer_id, B.first_name, B.last_name, E.country, D.city
        ORDER BY total_amount DESC
        LIMIT 5
)
SELECT AVG (top_5_customers.total_amount) AS average
FROM top_5_customers;
```

QUERY STEP 2

```
-- STEP 2 from task 3.8
WITH
top_10_countries AS (                                   -- top 10 countries CTE
        SELECT E.country
        FROM customer AS B
        INNER JOIN address AS C ON B.address_id = C.address_id
        INNER JOIN city AS D ON C.city_id = D.city_id
        INNER JOIN country AS E ON D.country_id = E.country_id
        GROUP BY E.country
        ORDER BY COUNT(B.customer_id) DESC
        LIMIT 10),
top_10_cities AS (                                      -- top 10 cities CTE
        SELECT D.city, E.country
        FROM customer AS B
        INNER JOIN address AS C ON B.address_id = C.address_id
        INNER JOIN city AS D ON C.city_id = D.city_id
        INNER JOIN country AS E ON D.country_id = E.country_id
        INNER JOIN top_10_countries top10con ON E.country = top10con.country
        GROUP BY E.country, D.city
        ORDER BY COUNT(B.customer_id) DESC
        LIMIT 10),
top_5_customers AS (                                    -- top 10 customers CTE
```

```
        SELECT A.customer_id, B.first_name, B.last_name, E.country, D.city,
                SUM(A.amount) AS total_amount
        FROM payment AS A
        INNER JOIN customer AS B ON A.customer_id = B.customer_id
        INNER JOIN address AS C on B.address_id = C.address_id
        INNER JOIN city AS D ON C.city_id = D.city_id
        INNER JOIN country AS E on D.country_id = E.country_id
        INNER JOIN top_10_cities top10cit ON D.city = top10cit.city AND E.country = top10cit.country
        GROUP BY A.customer_id, B.first_name, B.last_name, E.country, D.city
        ORDER BY total_amount DESC
        LIMIT 5)
SELECT E.country,                                                    -- recreating table-chain customer -->
country
                COUNT(DISTINCT B.customer_id) AS all_customer_count,
                COUNT(DISTINCT top_5_customers.customer_id) AS top_customer_count
FROM customer B
INNER JOIN address AS C ON B.address_id = C.address_id
INNER JOIN city AS D ON C.city_id = D.city_id
INNER JOIN country AS E ON D.country_id = E.country_id
LEFT JOIN top_5_customers ON E.country = top_5_customers.country
GROUP BY 1
ORDER BY top_customer_count DESC,
                all_customer_count DESC
LIMIT 5;
```