Task 3.9
Data Immersion

Common Table Expressions

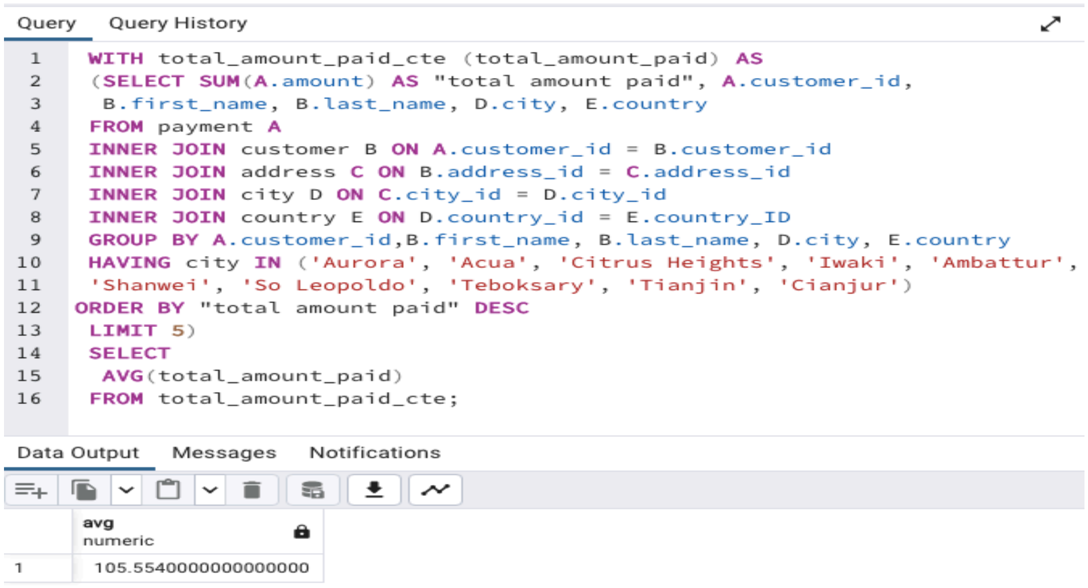
STEP 1:

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.

Average amount paid by the top 5 customers:

```
WITH total_amount_paid_cte (total_amount_paid) AS

 (SELECT SUM(A.amount) AS "total amount paid", A.customer_id,

  B.first_name, B.last_name, D.city, E.country

 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

 GROUP BY A.customer_id,B.first_name, B.last_name, D.city, E.country

 HAVING city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur',

 'Shanwei', 'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')

ORDER BY "total amount paid" DESC

 LIMIT 5)

 SELECT

  AVG(total_amount_paid)
```

FROM total_amount_paid_cte;



```
WITH total_amount_paid_cte (total_amount_paid) AS
 (SELECT SUM(A.amount) AS "total amount paid", A.customer_id,
  B.first_name, B.last_name, D.city, E.country
 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
 GROUP BY A.customer_id,B.first_name, B.last_name, D.city, E.country
 HAVING city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur',
 'Shanwei', 'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')
ORDER BY "total amount paid" DESC
 LIMIT 5)
 SELECT
  AVG(total_amount_paid)
 FROM total_amount_paid_cte;
```

```
   avg
   numeric
1  105.5540000000000000
```

Number of top 5 customers based within each country:

WITH top_5_customers AS

(SELECT A.first_name AS customer_first_name,

A.last_name AS customer_last_name,

A.customer_id,

D.country, city,

SUM(E.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 E ON A.customer_id = E.customer_id

WHERE city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'So Leopoldo', 'Teboksar','Tianjin', 'Cianjur')

GROUP BY country, city, A.customer_id, customer_first_name,

customer_last_name

ORDER BY total_amount_paid DESC

LIMIT 5)

SELECT D.country,

COUNT(A.customer_id) AS all_customer_count,

COUNT (top_5_customers) AS top_customer_count

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

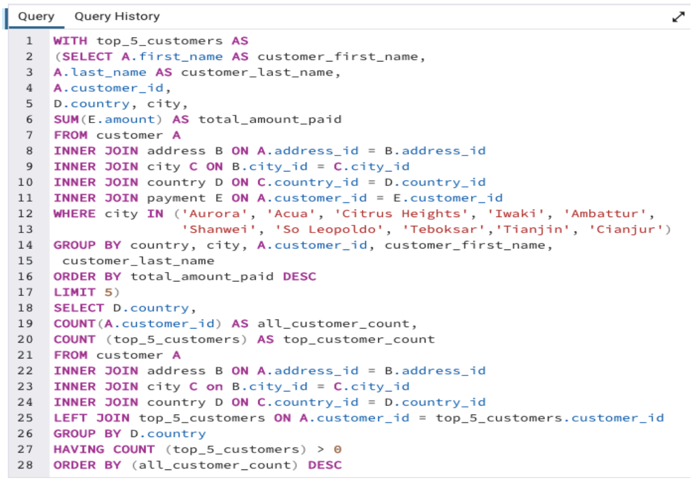
INNER JOIN country D ON C.country_id = D.country_id

LEFT JOIN top_5_customers ON A.customer_id = top_5_customers.customer_id

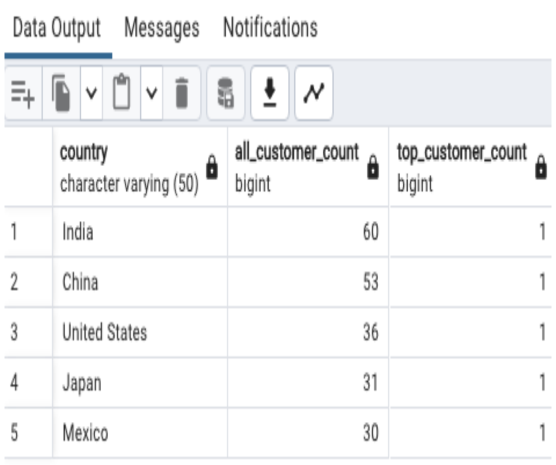
GROUP BY D.country

HAVING COUNT (top_5_customers) > 0

ORDER BY (all_customer_count) DESC



```sql
WITH top_5_customers AS
(SELECT A.first_name AS customer_first_name,
A.last_name AS customer_last_name,
A.customer_id,
D.country, city,
SUM(E.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 E ON A.customer_id = E.customer_id
WHERE city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur',
               'Shanwei', 'So Leopoldo', 'Teboksar','Tianjin', 'Cianjur')
GROUP BY country, city, A.customer_id, customer_first_name,
 customer_last_name
ORDER BY total_amount_paid DESC
LIMIT 5)
SELECT D.country,
COUNT(A.customer_id) AS all_customer_count,
COUNT (top_5_customers) AS top_customer_count
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
LEFT JOIN top_5_customers ON A.customer_id = top_5_customers.customer_id
GROUP BY D.country
HAVING COUNT (top_5_customers) > 0
ORDER BY (all_customer_count) DESC
```



	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

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

I started by utilizing common table expressions while making a table as to allow for CTE results. I also reordered the code to be more organized and therefore easier to understand. I made a reference to the CTE table in the query to gather the specific data I was looking for, with the rest of the code possessing an easy-to-understand dynamic.

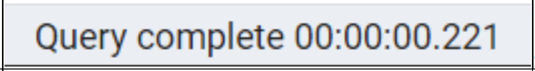
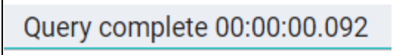
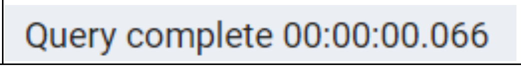
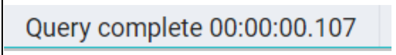
STEP 2:

1. Which approach do you think will perform better and why?

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

3. The EXPLAIN command gives you an estimated cost. To find out the actual speed of your queries, run them in pgAdmin 4. After each query has been run, a pop-up window will display its speed in milliseconds.

4. Did the results surprise you? Write a few sentences to explain your answer.

	CTE	**Subquery**
Average Paid Top 5 Customers	"Aggregate (cost=70.14..70.15 rows=1 width=32)"  Query complete 00:00:00.221	"Aggregate (cost=70.14..70.15 rows=1 width=32)"  Query complete 00:00:00.092
Top 5 Customers in each Country	"Sort (cost=142.90..142.99 rows=36 width=25)"  Query complete 00:00:00.066	"Sort (cost=142.90..142.99 rows=36 width=25)"  Query complete 00:00:00.107

As I examine the results as pertaining to the EXPLAIN function the speed was impressive, however, the CTE was faster in relation to the second step. I was surprised because I actually assumed the CTE functionality would overall be faster. This is because subqueries become less sensical to look at and read but the overall speed did surprise and impress me.

STEP 3:

Challenges I faced while replacing my subqueries with CTEs were more troubling at first but became clearer as I continued on. Firstly, I was examining the subqueries individually which made the process more difficult at first for separation purposes. This made the process more difficult and I was unable to make differentiations or separations. Once I resolved that issue, I was clearly able to decide whether components belonged together and which components needed readjusted or moved. Renaming queries was a small challenge because I would rename something and had to be absolutely sure to stick with the same name in proceedings as to not create errors in the process.