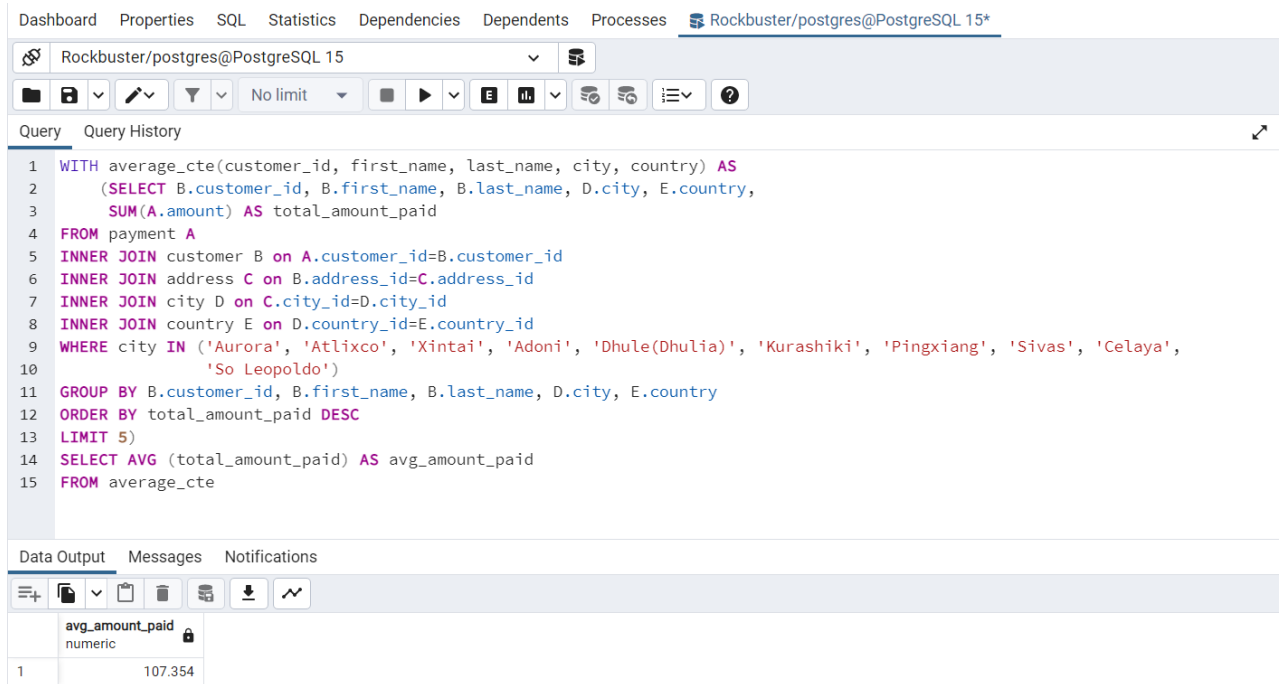


## Step 1: Answer the business questions from step 1 and 2 of task 3.8 using CTEs

- CTE- Query 1



The screenshot shows a PostgreSQL query editor interface. The top navigation bar includes 'Dashboard', 'Properties', 'SQL', 'Statistics', 'Dependencies', 'Dependents', and 'Processes'. The current connection is 'Rockbuster/postgres@PostgreSQL 15\*'. The query editor displays a SQL query using a Common Table Expression (CTE). The query is as follows:

```
1 WITH average_cte(customer_id, first_name, last_name, city, country) AS
2   (SELECT B.customer_id, B.first_name, B.last_name, D.city, E.country,
3     SUM(A.amount) AS total_amount_paid
4   FROM payment A
5   INNER JOIN customer B on A.customer_id=B.customer_id
6   INNER JOIN address C on B.address_id=C.address_id
7   INNER JOIN city D on C.city_id=D.city_id
8   INNER JOIN country E on D.country_id=E.country_id
9   WHERE city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule(Dhulia)', 'Kurashiki', 'Pingxiang', 'Sivas', 'Celaya',
10    'So Leopoldo'))
11  GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country
12  ORDER BY total_amount_paid DESC
13  LIMIT 5)
14  SELECT AVG (total_amount_paid) AS avg_amount_paid
15  FROM average_cte
```

Below the query editor, the 'Data Output' tab is active, showing the results of the query. The output is a single row with the following data:

avg_amount_paid
107.354

```
WITH average_cte(customer_id, first_name, last_name, city, country) 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(Dhulia)', 'Kurashiki', '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)
SELECT AVG (total_amount_paid) AS avg_amount_paid
FROM average_cte
```

## • CTE- Query 2

Rockbuster/postgres@PostgreSQL 15

Query Query History

```

1 WITH cte_top_customer_count (customer_id, first_name, last_name, city, country, total_amount_paid) AS
2   (SELECT B.customer_id, B.first_name, B.last_name, D.city, E.country,
3     SUM(A.amount) AS total_amount_paid
4   FROM payment A
5   INNER JOIN customer B ON A.customer_id=B.customer_id
6   INNER JOIN address C ON B.address_id=C.address_id
7   INNER JOIN city D ON C.city_id=D.city_id
8   INNER JOIN country E ON D.country_id=E.country_id
9   WHERE city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)', 'Kurashiki', 'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
10  GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country
11  ORDER BY total_amount_paid DESC
12  LIMIT 5),
13
14  cte_all_customer_count AS
15  (SELECT E.country,
16    COUNT(DISTINCT B.customer_id) AS all_customer_count,
17    COUNT(DISTINCT E.country) AS top_customer_count
18  FROM country E
19  INNER JOIN city D ON E.country_id = D.country_id
20  INNER JOIN address C ON D.city_id = C.city_id
21  INNER JOIN customer B ON C.address_id = B.address_id
22  GROUP BY E.country)
23
24 SELECT E.country,
25   COUNT(DISTINCT B.customer_id) AS all_customer_count,
26   COUNT(DISTINCT cte_top_customer_count.customer_id) AS top_customer_count
27
28 FROM country E
29  INNER JOIN city D ON E.country_id = D.country_id
30  INNER JOIN address C ON D.city_id = C.city_id
31  INNER JOIN customer B ON C.address_id = B.address_id
32  LEFT JOIN (SELECT B.customer_id, B.first_name, B.last_name, D.city, E.country,
33    SUM(A.amount) AS total_amount_paid
34  FROM payment A
35  INNER JOIN customer B ON A.customer_id=B.customer_id
36  INNER JOIN address C ON B.address_id=C.address_id
37  INNER JOIN city D ON C.city_id=D.city_id
38  INNER JOIN country E ON D.country_id=E.country_id
39  WHERE city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)', 'Kurashiki', 'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
40  GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country
41  ORDER BY total_amount_paid DESC
42  LIMIT 5) AS cte_top_customer_count
43  ON E.country=cte_top_customer_count.country
44
45 GROUP BY E.country
46 ORDER BY top_customer_count DESC
47 LIMIT 5

```

Data Output Messages Notifications

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	Mexico	30	2
2	United States	36	1
3	India	60	1
4	Turkey	15	1
5	American Samoa	1	0

WITH cte\_top\_customer\_count (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 (Dhulia)',
'Kurashiki', '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),

```

```

cte_all_customer_count 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 cte_top_customer_count.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 (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
(Dhulia)', 'Kurashiki', '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) AS cte_top_customer_count

```

```

ON E.country=cte_top_customer_count.country

```

```
GROUP BY E.country  
ORDER BY top_customer_count DESC  
LIMIT 5
```

- **Copy-paste your CTEs and their outputs into your answers document.**
- **Write 2 to 3 sentences explaining how you approached this step, for example, what you did first, second, and so on.**

As instructed in the lesson, I started my Common Table Expression with a WITH clause and the naming of the output table (cte\_top\_customer\_count and cte\_all\_customer\_count.) Please note, we were asked to produce a table that showed the countries with the top 5 customers by the number of top 5 customers in each country AND the total number of customers within each country. To do that we had to create an alias for the all\_customer\_count as well as the top\_customer\_count. In the case of the top\_customer\_count, we simply reused our query from the previous exercise. In the case of the all\_customer\_count, I looked through the database visualizer again and saw that I needed to count the number of distinct customer ids by country, hence, I had join the country and customer tables, grouping them by country, of course.z

When it came time to actually source this information, I knew that we needed to pull the countries that had the top 5 customers, meaning, we needed to pull from that same subquery we created in the previous exercise which pulled the top 5 customers, but this time, we joined it to the country table, because we were interested in pulling the data for those countries with the top 5 customers.

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

- **Which approach do you think will perform better and why?**

I believe the subquery style will perform better since the subqueries are always run first, and the CTE is basically nesting the subqueries even further to create a table which will then be referenced in the calculation of the final output. There just seem to be more mechanical steps involved for the PostGRE system when running the CTE.

- **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 each query has been run, a pop-up window will display its speed in milliseconds.**

### **Subquery 1 vs. CTE- Query 1**

<b>CTE</b>	<b>Aggregate (cost=64.45..64.46 rows=1 width=32)</b> <b>Total rows: 22 of 22</b> <b>Query complete 00:00:00.065</b>
<b>Subquery</b>	<b>Aggregate (cost=64.45..64.46 rows=1 width=32)</b> <b>Total rows: 22 of 22</b> <b>Query complete 00:00:00.103</b>

### Subquery 1

	QUERY PLAN text
1	Aggregate (cost=64.45..64.46 rows=1 width=32)
2	-> Limit (cost=64.37..64.39 rows=5 width=67)
3	-> Sort (cost=64.37..64.98 rows=243 width=67)
4	Sort Key: (sum(a.amount)) DESC
5	-> HashAggregate (cost=57.30..60.34 rows=243 width=67)
6	Group Key: b.customer_id, d.city, e.country
7	-> Nested Loop (cost=18.16..54.87 rows=243 width=41)
8	-> Hash Join (cost=17.88..37.14 rows=10 width=35)
9	Hash Cond: (d.country_id = e.country_id)
10	-> Nested Loop (cost=14.43..33.66 rows=10 width=28)
11	-> Hash Join (cost=14.15..29.77 rows=10 width=15)
12	Hash Cond: (c.city_id = d.city_id)
13	-> Seq Scan on address c (cost=0.00..14.03 rows=603 width=6)
14	-> Hash (cost=14.03..14.03 rows=10 width=15)
15	-> Seq Scan on city d (cost=0.03..14.03 rows=10 width=15)
16	Filter: ((city)::text = ANY ('{Aurora,Atlixco,Xintai,Adoni,"Dhule (Dhulia)",Kurashiki,Pingxiang,Sivas,Celaya,"So Leopoldo"}'::text[]))
17	-> Index Scan using idx_fk_address_id on customer b (cost=0.28..0.38 rows=1 width=19)
18	Index Cond: (address_id = c.address_id)
19	-> Hash (cost=2.09..2.09 rows=109 width=13)
20	-> Seq Scan on country e (cost=0.00..2.09 rows=109 width=13)
21	-> Index Scan using idx_fk_customer_id on payment a (cost=0.29..1.53 rows=24 width=8)

## CTE- Query 1

	QUERY PLAN text	
1	Aggregate (cost=64.45..64.46 rows=1 width=32)	
2	-> Limit (cost=64.37..64.39 rows=5 width=67)	
3	-> Sort (cost=64.37..64.98 rows=243 width=67)	
4	Sort Key: (sum(a.amount)) DESC	
5	-> HashAggregate (cost=57.30..60.34 rows=243 width=67)	
6	Group Key: b.customer_id, d.city, e.country	
7	-> Nested Loop (cost=18.16..54.87 rows=243 width=41)	
8	-> Hash Join (cost=17.88..37.14 rows=10 width=35)	
9	Hash Cond: (d.country_id = e.country_id)	
10	-> Nested Loop (cost=14.43..33.66 rows=10 width=28)	
11	-> Hash Join (cost=14.15..29.77 rows=10 width=15)	
12	Hash Cond: (c.city_id = d.city_id)	
13	-> Seq Scan on address c (cost=0.00..14.03 rows=603 width=6)	
14	-> Hash (cost=14.03..14.03 rows=10 width=15)	
15	-> Seq Scan on city d (cost=0.03..14.03 rows=10 width=15)	
16	Filter: ((city)::text = ANY ({'Aurora,Atlixco,Xintai,Adoni,Dhule(Dhulia),Kurashiki,Pingxiang,Sivas,Celaya,'So Leopoldo'}::text[]))	
17	-> Index Scan using idx_fk_address_id on customer b (cost=0.28..0.38 rows=1 width=19)	
18	Index Cond: (address_id = c.address_id)	
19	-> Hash (cost=2.09..2.09 rows=109 width=13)	
20	-> Seq Scan on country e (cost=0.00..2.09 rows=109 width=13)	
21	-> Index Scan using idx_fk_customer_id on payment a (cost=0.29..1.53 rows=24 width=8)	

## Subquery 2 vs. CTE- Query

<b>CTE</b>	<b>Limit (cost=166.23..166.24 rows=5 width=25)</b> <b>Total rows: 46 of 46</b> <b>Query complete 00:00:00.092</b>
<b>Subquery</b>	<b>Limit (cost=166.23..166.24 rows=5 width=25)</b> <b>Total rows: 46 of 46</b> <b>Query complete 00:00:00.090</b>

## Subquery 2

	QUERY PLAN text	
1	Limit (cost=166.23..166.24 rows=5 width=25)	
2	-> Sort (cost=166.23..166.50 rows=109 width=25)	
3	Sort Key: (count(DISTINCT top_5_customer.customer_id)) DESC	
4	-> GroupAggregate (cost=155.43..164.42 rows=109 width=25)	
5	Group Key: e.country	
6	-> Merge Left Join (cost=155.43..158.83 rows=599 width=17)	
7	Merge Cond: ((e.country)::text = (top_5_customer.country)::text)	
8	-> Sort (cost=90.94..92.44 rows=599 width=13)	
9	Sort Key: e.country	
10	-> Hash Join (cost=43.52..63.30 rows=599 width=13)	
11	Hash Cond: (d.country_id = e.country_id)	
12	-> Hash Join (cost=40.07..58.22 rows=599 width=6)	
13	Hash Cond: (c.city_id = d.city_id)	
14	-> Hash Join (cost=21.57..38.14 rows=599 width=6)	
15	Hash Cond: (b.address_id = c.address_id)	
16	-> Seq Scan on customer b (cost=0.00..14.99 rows=599 width=6)	
17	-> Hash (cost=14.03..14.03 rows=603 width=6)	
18	-> Seq Scan on address c (cost=0.00..14.03 rows=603 width=6)	
19	-> Hash (cost=11.00..11.00 rows=600 width=6)	
20	-> Seq Scan on city d (cost=0.00..11.00 rows=600 width=6)	
21	-> Hash (cost=2.09..2.09 rows=109 width=13)	

## CTE- Query 2

	QUERY PLAN text	
1	Limit (cost=166.23..166.24 rows=5 width=25)	
2	-> Sort (cost=166.23..166.50 rows=109 width=25)	
3	Sort Key: (count(DISTINCT cte_top_customer_count.customer_id)) DESC	
4	-> GroupAggregate (cost=155.43..164.42 rows=109 width=25)	
5	Group Key: e.country	
6	-> Merge Left Join (cost=155.43..158.83 rows=599 width=17)	
7	Merge Cond: ((e.country)::text = (cte_top_customer_count.country)::text)	
8	-> Sort (cost=90.94..92.44 rows=599 width=13)	
9	Sort Key: e.country	
10	-> Hash Join (cost=43.52..63.30 rows=599 width=13)	
11	Hash Cond: (d.country_id = e.country_id)	
12	-> Hash Join (cost=40.07..58.22 rows=599 width=6)	
13	Hash Cond: (c.city_id = d.city_id)	
14	-> Hash Join (cost=21.57..38.14 rows=599 width=6)	
15	Hash Cond: (b.address_id = c.address_id)	
16	-> Seq Scan on customer b (cost=0.00..14.99 rows=599 width=6)	
17	-> Hash (cost=14.03..14.03 rows=603 width=6)	
18	-> Seq Scan on address c (cost=0.00..14.03 rows=603 width=6)	
19	-> Hash (cost=11.00..11.00 rows=600 width=6)	
20	-> Seq Scan on city d (cost=0.00..11.00 rows=600 width=6)	
21	-> Hash (cost=2.09..2.09 rows=109 width=13)	

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

The costs between the subquery and CTE methods were basically identical but the actual run times were slightly faster for the subquery method when we were looking for how many of the top 5 customers were based within each country but slightly faster for the CTE method when we were looking for the average amount paid by the top 5 customers.

I was surprised that the estimated costs were the same but not as shocked by the slightly lower run time for the subquery method on the more complicated query (i.e. the top 5 customers within each country.)

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

While I do follow the general, big-picture concept of writing a CTE and a subquery, the syntax on these last few problems have just been so complicated, that I'm constantly trying to understand where my mistakes are. It's kind of hard to follow and, if I'm being honest, I need this explained to me by a person.