

3.9: Common Table Expressions

Alejandro Salgado

STEP 1:

- Find the average amount paid by the top 5 customers.

The screenshot shows the pgAdmin 4 interface with a SQL query editor. The query is as follows:

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

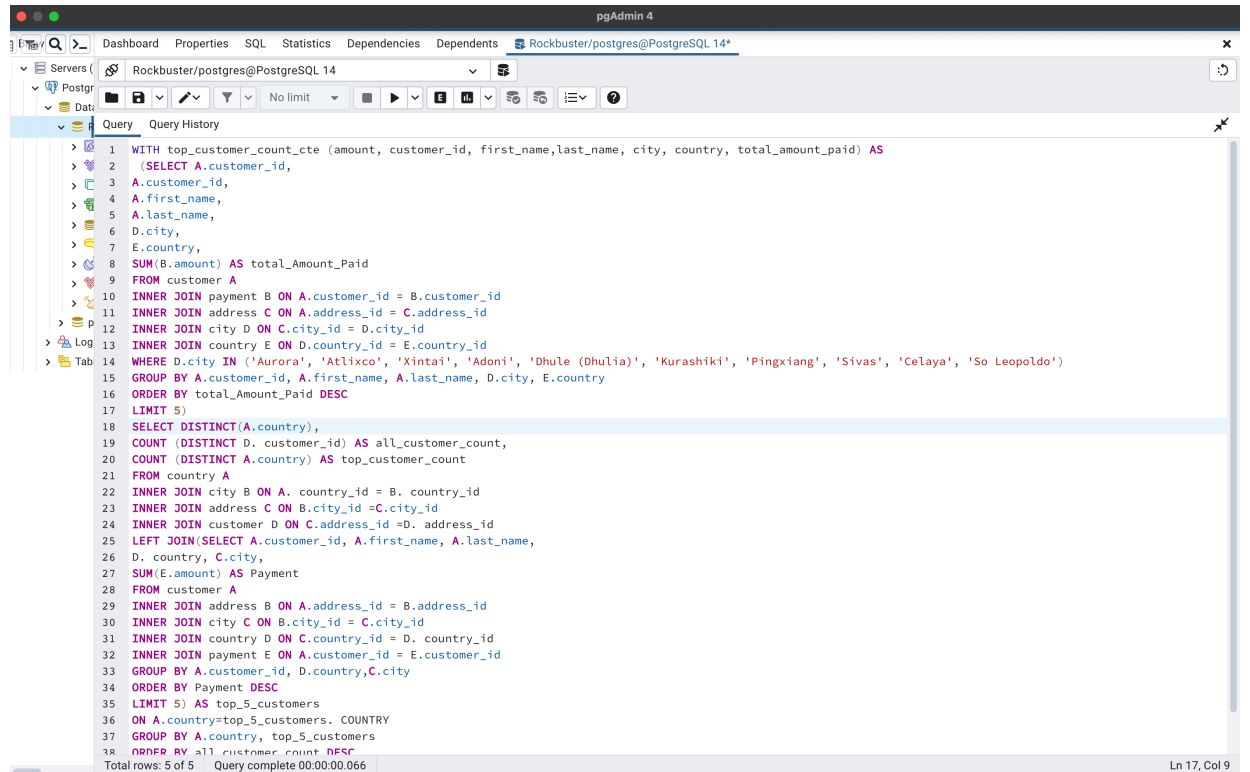
The query results are displayed in the Data output tab, showing a single row with the average amount paid:

average_total_amount_paid
107.354

Total rows: 1 of 1 Query complete 00:00:00.189 Ln 17, Col 9

STEP 2:

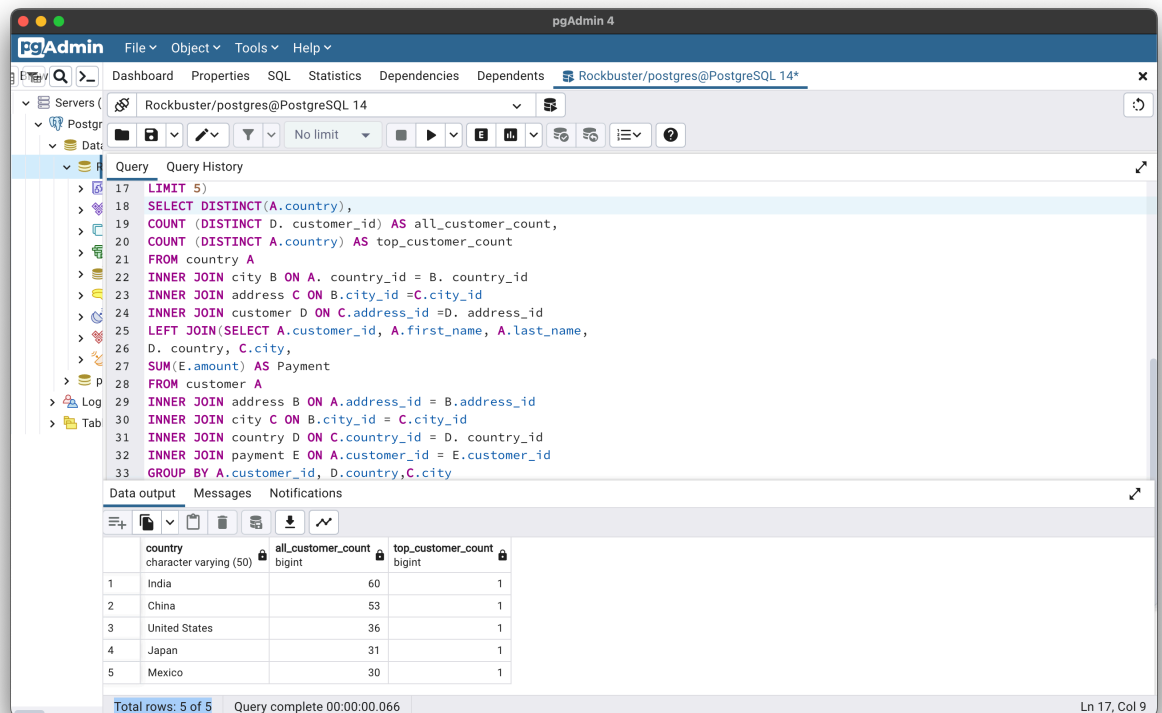
- Find the out how many of the top 5 customers are based within each country.



The screenshot shows the pgAdmin 4 interface with a SQL query editor. The query is a complex SQL statement designed to find the top 5 customers by total amount paid, grouped by country. It uses multiple CTEs and joins to aggregate data from the customer, payment, address, and city tables. The query is as follows:

```
1 WITH top_customer_count_cte (amount, customer_id, first_name, last_name, city, country, total_amount_paid) AS
2 (SELECT A.customer_id,
3  A.customer_id,
4  A.first_name,
5  A.last_name,
6  D.city,
7  E.country,
8  SUM(B.amount) AS total_Amount_Paid
9 FROM customer A
10 INNER JOIN payment B ON A.customer_id = B.customer_id
11 INNER JOIN address C ON A.address_id = C.address_id
12 INNER JOIN city D ON C.city_id = D.city_id
13 INNER JOIN country E ON D.country_id = E.country_id
14 WHERE D.city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)', 'Kurashiki', 'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
15 GROUP BY A.customer_id, A.first_name, A.last_name, D.city, E.country
16 ORDER BY total_Amount_Paid DESC
17 LIMIT 5)
18 SELECT DISTINCT(A.country),
19 COUNT (DISTINCT D. customer_id) AS all_customer_count,
20 COUNT (DISTINCT A.country) AS top_customer_count
21 FROM country A
22 INNER JOIN city B ON A. country_id = B. country_id
23 INNER JOIN address C ON B.city_id =C.city_id
24 INNER JOIN customer D ON C.address_id =D. address_id
25 LEFT JOIN(SELECT A.customer_id, A.first_name, A.last_name,
26 D. country, C.city,
27 SUM(E.amount) AS Payment
28 FROM customer A
29 INNER JOIN address B ON A.address_id = B.address_id
30 INNER JOIN city C ON B.city_id = C.city_id
31 INNER JOIN country D ON C.country_id = D. country_id
32 INNER JOIN payment E ON A.customer_id = E.customer_id
33 GROUP BY A.customer_id, D.country,C.city
34 ORDER BY Payment DESC
35 LIMIT 5) AS top_5_customers
36 ON A.country=top_5_customers. COUNTRY
37 GROUP BY A.country, top_5_customers
38 ORDER BY all_customer_count DESC
```

Total rows: 5 of 5 Query complete 00:00:00.066 Ln 17, Col 9



The screenshot shows the pgAdmin 4 interface with the same SQL query. The data output is displayed in a table format. The table has three columns: country, all_customer_count, and top_customer_count. The data is as follows:

country	all_customer_count	top_customer_count
India	60	1
China	53	1
United States	36	1
Japan	31	1
Mexico	30	1

Total rows: 5 of 5 Query complete 00:00:00.066 Ln 17, Col 9

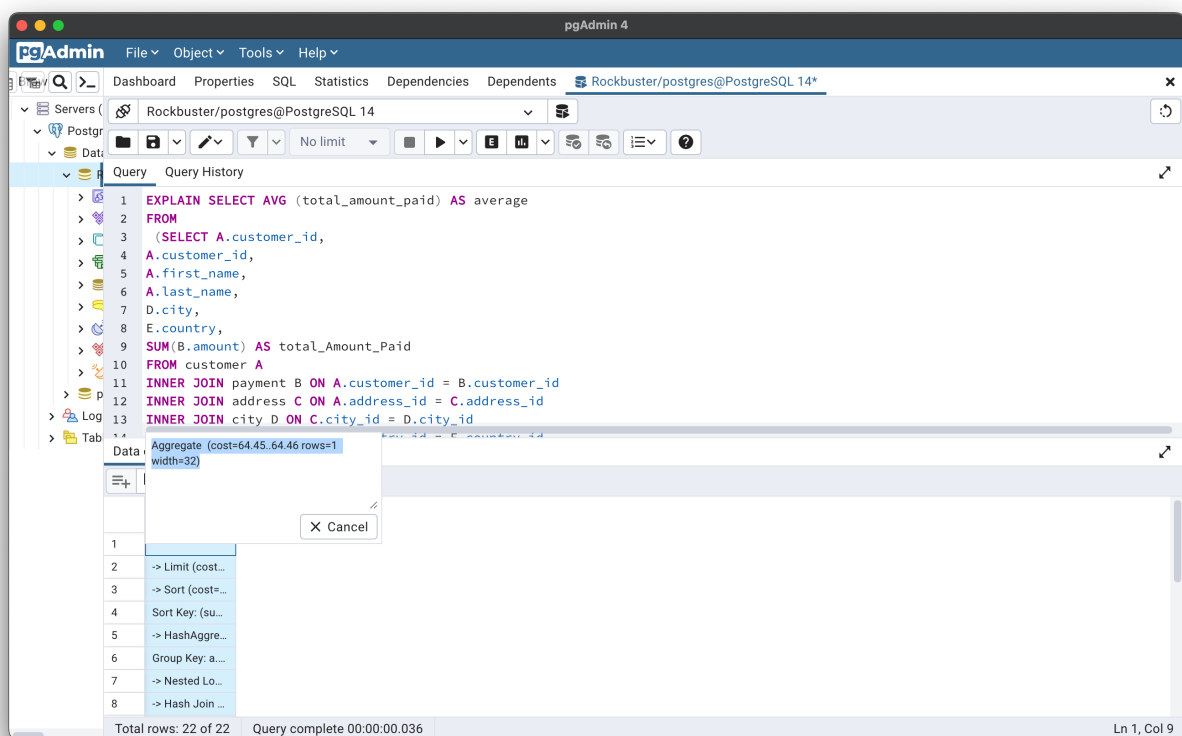
2. Write 2 to 3 sentences explaining how you approached this step, for example, what you did first, second, and so on.

- For the first cte I put in the WITH command and made a column for the average total amount paid and placed the subsequent commands. I copied and pasted the sub query I used in task 3.8 and used a SELECT command at the end to gives the average paid. For the second one I tweaked the WITH command to change to top customer count cte. I then pasted the same subquery I did from task 3.8 again and deleted a couple of lines since I was getting an error message. I was to ultimately run it and it shows the top 5 countries.

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

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



CTE:

The screenshot displays the pgAdmin 4 web interface. The top navigation bar includes 'File', 'Object', 'Tools', and 'Help'. The main toolbar contains icons for various database actions. The left sidebar shows a tree view with 'Servers' expanded, highlighting 'PostgreSQL 14'. The central pane is titled 'Query' and shows a SQL query for calculating the average total amount paid by customer. The query is as follows:

```

1 EXPLAIN WITH average_total_amount_paid_cte (amount, customer_id, first_name, last_name, city, country, total_amount_paid) AS
2 (SELECT A.customer_id,
3  A.first_name,
4  A.last_name,
5  D.city,
6  E.country,
7  SUM(B.amount) AS total_Amount_Paid
8 FROM customer A
9 INNER JOIN payment B ON A.customer_id = B.customer_id
10 INNER JOIN address C ON A.address_id = C.address_id
11 INNER JOIN city D ON C.city_id = D.city_id
12 INNER JOIN country E ON D.country_id = E.country_id
13 WHERE D.city IN ('Amsterdam', 'Antwerp', 'Brussels', 'Ghent', 'Liège', 'Luxembourg', 'Maastricht', 'Namur', 'Paris', 'Roubaix', 'Strasbourg', 'Toulouse', 'Valenciennes', 'Vervins'))
14 SELECT * FROM average_total_amount_paid_cte

```

Below the query editor, the 'Data output' tab is active, showing the 'QUERY PLAN' for the executed query. The plan details the execution steps and their costs:

Step	Operation	Cost	Rows	Width
1	Aggregate	(cost=64.45..64.46)	rows=1	width=32
2	-> Limit	(cost=64.37..64.39)	rows=5	width=71
3	-> Sort	(cost=64.37..64.98)	rows=243	width=71
4	Sort Key: (sum(b.amount)) DESC			
5	-> HashAggregate	(cost=57.30..60.34)	rows=243	width=71
6	Group Key: a.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

At the bottom of the query plan, it states: 'Total rows: 22 of 22' and 'Query complete 00:00:00.134'.

For the second task:

Subquery:

The screenshot shows the pgAdmin 4 interface with a SQL query editor and its execution plan. The query is as follows:

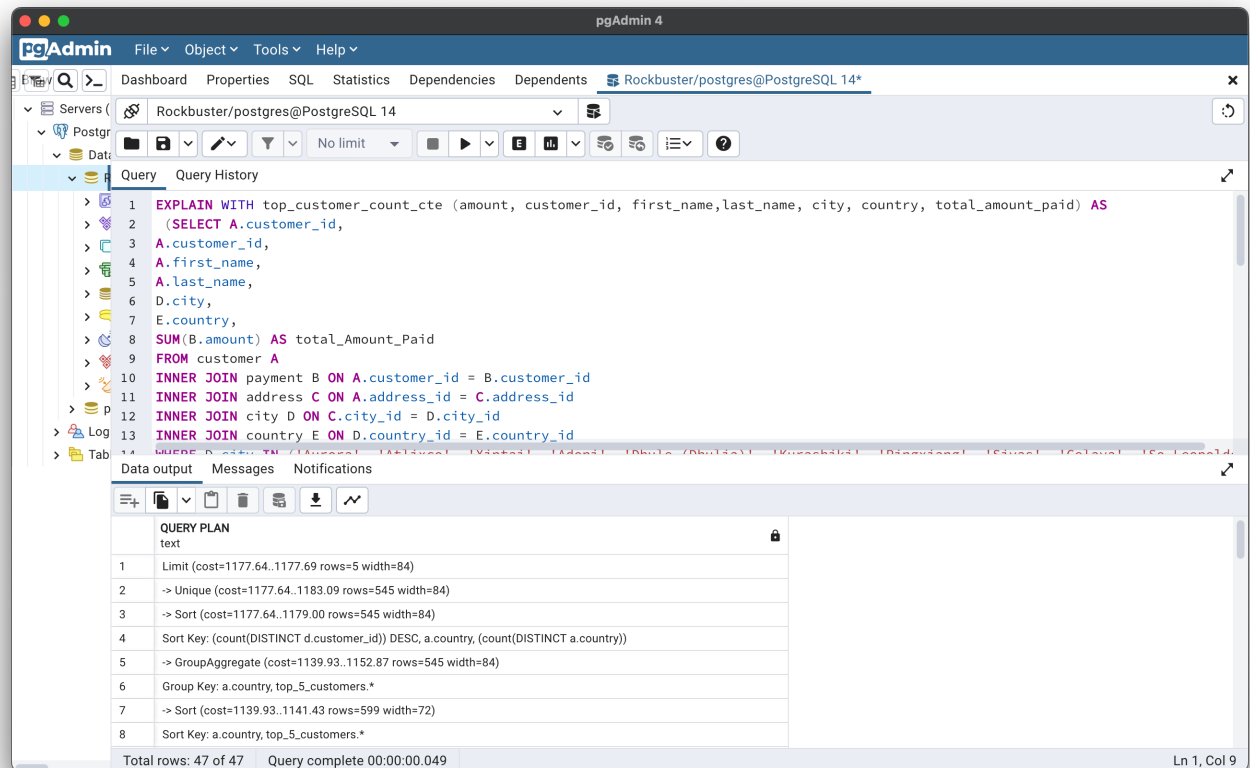
```
1 EXPLAIN SELECT DISTINCT(A.country),
2 COUNT (DISTINCT D. customer_id) AS all_customer_count,
3 COUNT (DISTINCT A.country) AS top_customer_count
4 FROM country A
5 INNER JOIN city B ON A. country_id = B. country_id
6 INNER JOIN address C ON B.city_id =C.city_id
7 INNER JOIN customer D ON C.address_id =D. address_id
8 LEFT JOIN(SELECT A.customer_id, A.first_name, A.last_name,
9 D. country, C.city,
10 SUM(E.amount) AS Payment
11 FROM customer A
12 INNER JOIN address B ON A.address_id = B.address_id
13 INNER JOIN city C ON B.city_id = C.city_id
14 INNER JOIN customer D ON C.city_id = D.city_id
```

The execution plan (QUERY PLAN) is shown below:

Step	Plan
1	Limit (cost=1177.64..1177.69 rows=5 width=84)
2	-> Unique (cost=1177.64..1183.09 rows=545 width=84)
3	-> Sort (cost=1177.64..1179.00 rows=545 width=84)
4	Sort Key: (count(DISTINCT d.customer_id)) DESC, a.country, (count(DISTINCT a.country))
5	-> GroupAggregate (cost=1139.93..1152.87 rows=545 width=84)
6	Group Key: a.country, top_5_customers.*
7	-> Sort (cost=1139.93..1141.43 rows=599 width=72)
8	Sort Key: a.country, top_5_customers.*

Total rows: 47 of 47 Query complete 00:00:00.109 Ln 1, Col 9

CTE:



Write a few sentences on your results.

- For the first one, the cost is the same. The difference is in the runtimes. The subquery one is faster. For the second one, the cost is the same as well. However, it looks like the runtime is faster on cte.

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

- For the first question I did not know how to properly use the WITH command for the cte, but after a few tries I was able to get it. I believe the second one was tougher because I thought that I had to use the WITH command twice, but I only needed it once. When I combined the two subqueries, I had to delete some rows in order for it to run properly.