

The screenshot shows the pgAdmin 4 web interface. The left sidebar displays the server hierarchy: Servers (1) > PostgreSQL 14 > Databases (2) > Rockbuster. The main pane is titled 'Rockbuster/postgres@PostgreSQL 14\*' and shows the 'Query Editor' tab. A SQL query is entered, and the 'Data Output' tab at the bottom shows the result of the query.

**SQL Query:**

```

1 WITH cte_average(customer_id, first_name, last_name, city, country, amount) AS
2 (SELECT A.customer_id,
3  B.first_name,
4  B.last_name,
5  E.country,
6  D.city,
7  SUM(A.amount) AS total_amount_paid
8 FROM payment A
9 INNER JOIN customer B ON A.customer_id = B.customer_id
10 INNER JOIN address C ON B.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 city IN ('Aurora', 'Tokat', 'Tarsus', 'Atlixco', 'Emeshan', 'Pontianak', 'Shimoga', 'Aparecida de Goiania', 'Zalatun', 'Taguig'))
14 GROUP BY A.customer_id,
15  B.first_name,
16  B.last_name,
17  E.country,
18  D.city
19 ORDER BY total_amount_paid DESC LIMIT 5)
20 SELECT AVG(amount)
21 FROM cte_average
  
```

**Query Result (Data Output):**

avg	numeric
1	120.32200000000000000000

1.b)

The screenshot shows the pgAdmin 4 interface. The Query Editor contains the following SQL query:

```

1 WITH top_customer_count_cte AS (SELECT A.customer_id,
2   A.first_name,
3   A.last_name,
4   C.city,
5   D.country_id,
6   SUM(E.amount) AS total_amount
7 FROM payment E
8 INNER JOIN customer A ON E.customer_id = A.customer_id
9 INNER JOIN address B ON A.address_id = B.address_id
10 INNER JOIN city C ON B.city_id = C.city_id
11 INNER JOIN country D ON C.country_id = D.country_id
12 WHERE C.city IN ('Aurora', 'Tokat', 'Tarsus', 'Atlixco', 'Emeishan', 'Pontianak', 'Shimoga', 'Aparecida de Goiania', 'Zalatun', 'Taguig')
13 GROUP BY A.customer_id, D.country_id, C.city
14 ORDER BY total_amount DESC
15 LIMIT 5),
16 all_customer_count_cte AS (SELECT D.country,
17   COUNT (DISTINCT A.customer_id) AS all_customer_count
18 FROM customer A
19 INNER JOIN address B ON A.address_id = B.address_id
20 INNER JOIN city C ON B.city_id = C.city_id
21 INNER JOIN country D ON C.country_id = D.country_id
22 GROUP BY D.country)
23 SELECT D.country,

```

The Data Output pane shows the following results:

country	all_customer_count	top_customer_count
1 Mexico	30	1
2 Turkey	15	1
3 China	53	1
4 Indonesia	14	1

The screenshot shows the pgAdmin 4 interface with a modified SQL query. The Query Editor contains the following SQL query:

```

14 ORDER BY total_amount DESC
15 LIMIT 5),
16 all_customer_count_cte AS (SELECT D.country,
17   COUNT (DISTINCT A.customer_id) AS all_customer_count
18 FROM customer A
19 INNER JOIN address B ON A.address_id = B.address_id
20 INNER JOIN city C ON B.city_id = C.city_id
21 INNER JOIN country D ON C.country_id = D.country_id
22 GROUP BY D.country)
23 SELECT D.country,
24   COUNT (DISTINCT A.customer_id) AS all_customer_count,
25   COUNT (DISTINCT top_customer_count_cte.customer_id) AS top_customer_count
26 FROM customer A
27 INNER JOIN address B ON A.address_id = B.address_id
28 INNER JOIN city C ON B.city_id = C.city_id
29 INNER JOIN country D ON C.country_id = D.country_id
30 LEFT JOIN top_customer_count_cte ON D.country_id = top_customer_count_cte.country_id
31 GROUP BY D.country
32 ORDER BY top_customer_count DESC
33 LIMIT 5;
34

```

The Data Output pane shows the following results:

country	all_customer_count	top_customer_count
1 Mexico	30	1
2 Turkey	15	1
3 China	53	1
4 Indonesia	14	1
5 United States	36	1

1.c) I have defined the CTE with the 'WITH' command, as well as necessary columns. I have set the name for the CTE to a descriptive name. I have wrote the main statement by inserting already the written queries from task 3.8 in parentheses after the AS command. I have adjusted order and grouping accordingly.

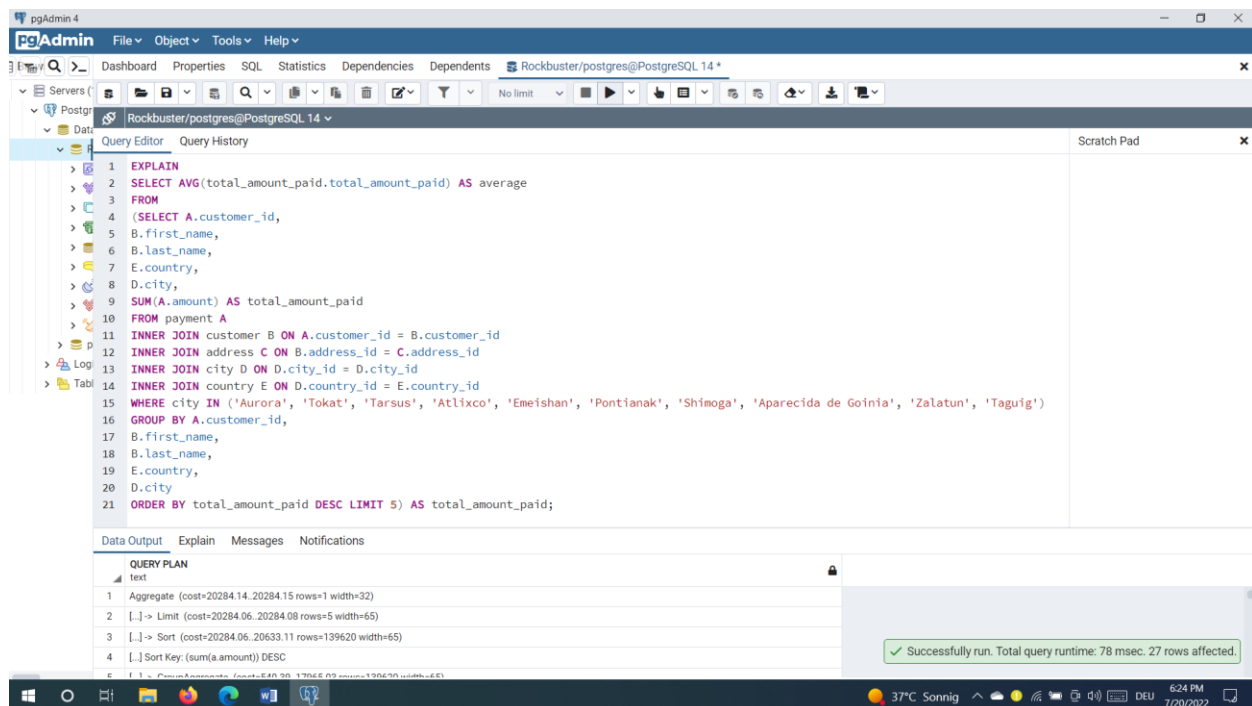
## Step 2: Compare the performance of your CTEs and subqueries.

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

I did not expect which approach would perform better.

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.

### 3.8.1



The screenshot shows the pgAdmin 4 interface. The Query Editor contains the following SQL query:

```
1 EXPLAIN
2 SELECT AVG(total_amount_paid, total_amount_paid) AS average
3 FROM
4 (SELECT A.customer_id,
5  B.first_name,
6  B.last_name,
7  E.country,
8  D.city,
9  SUM(A.amount) AS total_amount_paid
10 FROM payment A
11 INNER JOIN customer B ON A.customer_id = B.customer_id
12 INNER JOIN address C ON B.address_id = C.address_id
13 INNER JOIN city D ON D.city_id = D.city_id
14 INNER JOIN country E ON D.country_id = E.country_id
15 WHERE city IN ('Aurora', 'Tokat', 'Tarsus', 'Atlixco', 'Emeishan', 'Pontianak', 'Shimoga', 'Aparecida de Goiania', 'Zalatur', 'Taguig')
16 GROUP BY A.customer_id,
17 B.first_name,
18 B.last_name,
19 E.country,
20 D.city
21 ORDER BY total_amount_paid DESC LIMIT 5) AS total_amount_paid;
```

The Query Plan tab shows the following execution plan:

Step	Plan	Cost	Rows	Width
1	Aggregate	(cost=20284.14..20284.15 rows=1 width=32)		
2	[.]> Limit	(cost=20284.06..20284.08 rows=5 width=65)		
3	[.]> Sort	(cost=20284.06..20633.11 rows=139620 width=65)		
4	[.] Sort Key: (sum(a.amount)) DESC			

A green status bar at the bottom right indicates: "Successfully run. Total query runtime: 78 msec. 27 rows affected."

### 3.8.2

The screenshot displays the pgAdmin 4 web interface. On the left, a tree view shows the database structure: Servers (1) > PostgreSQL 14 > Databases (2) > Rockbuster. The main pane is divided into two sections: the top section contains the SQL query editor, and the bottom section shows the query plan and execution status.

**SQL Query:**

```
1 EXPLAIN
2 SELECT e.country,
3 COUNT(DISTINCT B.customer_ID) AS all_customer_count,
4 COUNT(DISTINCT top_5_customers) AS top_customer_count
5 FROM payment A
6 INNER JOIN customer B ON A.customer_ID = B.customer_ID
7 INNER JOIN address C ON B.address_id = C.address_id
8 INNER JOIN city D ON C.city_id = D.city_id
9 INNER JOIN country E ON D.country_ID = E.country_ID
10 LEFT JOIN
11 (SELECT B.customer_id, B.last_name, B.first_name, D.city, E.country,
12 SUM(amount) AS total_amount_paid
13 FROM payment A
14 INNER JOIN customer B ON A.customer_id = B.customer_id
15 INNER JOIN address C ON B.address_id = C.address_id
16 INNER JOIN city D ON C.city_id = D.city_id
17 INNER JOIN country E ON D.country_ID = E.country_ID
18 WHERE D.city IN ('Aurora', 'Tokat', 'Tarsus', 'Atlixco', 'Emeishan', 'Pontianak', 'Shimoga', 'Aparecida de Goiania', 'Zalantun', 'Taguig')
19 GROUP BY B.customer_id, B.last_name, B.first_name, D.city, E.country
20 ORDER BY total_amount_paid DESC
```

**Query Plan:**

Step	Text	Cost	Rows	Width
1	Limit	(cost=879.36..879.38 rows=5 width=25)		
2	[.]> Sort	(cost=879.36..879.64 rows=109 width=25)		
3	[.] Sort Key: (count(DISTINCT top_5_customers *)) DESC			
4	[.]> GroupAggregate	(cost=155.05..877.55 rows=109 width=25)		
5	[.] Group Key: e.country			
6	[.]> Nested Loop	(cost=155.05..771.75 rows=13962 width=72)		

**Execution Status:** Successfully run. Total query runtime: 68 msec. 49 rows affected.

### 3.9.1

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

```
1 EXPLAIN
2 WITH cte_average(customer_id, first_name, last_name, city, country, amount) AS
3 (SELECT A.customer_id,
4  B.first_name,
5  B.last_name,
6  E.country,
7  D.city,
8  SUM(A.amount) AS total_amount_paid
9  FROM payment A
10 INNER JOIN customer B ON A.customer_id = B.customer_id
11 INNER JOIN address C ON B.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 city IN ('Aurora', 'Tokat', 'Tarsus', 'Atlixco', 'Emeishan', 'Pontianak', 'Shimoga', 'Aparecida de Goiania', 'Zalatum', 'Taguig')
15 GROUP BY A.customer_id,
16  B.first_name,
17  B.last_name,
18  E.country,
19  D.city
20 ORDER BY total_amount_paid DESC LIMIT 5)
21 SELECT AVG(amount)
22 FROM cte_average
```

The query plan shows the following steps:

- 1. Aggregate (cost=64.95..64.96 rows=1 width=32)
- 2. [...] -> Limit (cost=64.87..64.89 rows=5 width=65)
- 3. [...] -> Sort (cost=64.87..65.46 rows=233 width=65)
- 4. [...] Sort Key: (sum(a.amount)) DESC

The status bar indicates: Successfully run. Total query runtime: 117 msec. 22 rows affected.

### 3.9.2

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

```
14 GROUP BY A.customer_id, D.country_id, C.city
15 ORDER BY total_amount DESC
16 LIMIT 5;
17 all_customer_count_cte AS (SELECT D.country,
18  COUNT(DISTINCT A.customer_id) AS all_customer_count
19  FROM customer A
20 INNER JOIN address B ON A.address_id = B.address_id
21 INNER JOIN city C ON B.city_id = C.city_id
22 INNER JOIN country D ON C.country_id = D.country_id
23 GROUP BY D.country)
24 SELECT D.country,
25  COUNT(DISTINCT A.customer_id) AS all_customer_count,
26  COUNT(DISTINCT top_customer_count_cte.customer_id) AS top_customer_count
27  FROM customer A
28 INNER JOIN address B ON A.address_id = B.address_id
29 INNER JOIN city C ON B.city_id = C.city_id
30 INNER JOIN country D ON C.country_id = D.country_id
31 LEFT JOIN top_customer_count_cte ON D.country_id = top_customer_count_cte.country_id
32 GROUP BY D.country
33 ORDER BY top_customer_count DESC
34 LIMIT 5;
35
```

The query plan shows the following steps:

- 1. Limit (cost=166.18..166.19 rows=5 width=25)
- 2. [...] -> Sort (cost=166.18..166.45 rows=109 width=25)
- 3. [...] Sort Key: (count(DISTINCT top\_customer\_count\_cte.customer\_id)) DESC
- 4. [...] GroupAggregate (cost=157.29..164.37 rows=109 width=25)

The status bar indicates: Successfully run. Total query runtime: 84 msec. 45 rows affected.

Table 1 contains the data about cost and speed of its queries.

**Table 1.**

QUERY	COST	SPEED (msec)
3.8.1	20284	78
3.8.2	879	68
3.9.1	64.95	117
3.9.2	166.18	84

4. Did the results surprise you? Write a few sentences to explain your answer.  
I am little surprised that COST of the Query 3.8.1 since it is significantly higher in compare with another queries.

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

For me writing the queries with CTE was harder.