

```

WITH average_cte AS (SELECT B. customer_id, B. first_name, B. last_name, D. city,
country, SUM(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 D. city IN
('Aurora','Tokat','Tarsus','Atlixco','Emeishan','Pontianak','Shimoga','Aparecida de
Goinia','Zalantun','Taguig') GROUP BY B. customer_id, B. first_name, B. last_name, D.
city, country
ORDER BY total_amount_paid DESC LIMIT 5) SELECT (AVG (total_amount_paid), 2) AS
average_amount_paid FROM average_cte;

```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, including the 'Rockbuster' database. The central pane shows the SQL query editor with the following query:

```

1 WITH average_cte AS (SELECT B. customer_id, B. first_name, B. last_name, D. city, country, SUM(amount) AS total_amount_paid FROM payment
2 INNER JOIN customer B ON A. customer_id = B. customer_id INNER JOIN address C ON B. address_id = C. address_id
3 INNER JOIN city D ON C. city_id = D. city_id INNER JOIN country E ON D. country_id = E. country_id
4 WHERE D. city IN ('Aurora','Tokat','Tarsus','Atlixco','Emeishan','Pontianak','Shimoga','Aparecida de Goinia','Zalantun','Taguig') GROUP BY
5 ORDER BY total_amount_paid DESC LIMIT 5) SELECT (AVG (total_amount_paid), 2) AS average_amount_paid FROM average_cte;

```

The 'Data Output' window displays the following result:

average_amount_paid
(120.322000000000000000,2)

The 'Messages' window shows the following status:

Successfully run. Total query runtime: 98 msec.  
1 rows affected.

```

WITH customers_country_cte AS (SELECT B. customer_id, B. first_name, B. last_name,
D. city, country, SUM(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 D. city IN
('Aurora','Tokat','Tarsus','Atlixco','Emeishan','Pontianak','Shimoga','Aparecida de
Goinia','Zalantun','Taguig') GROUP BY B. customer_id, B. first_name, B. last_name, D.
city, country ORDER BY total_amount_paid DESC LIMIT 5) SELECT D.country, COUNT
(DISTINCT A.customer_id) AS all_customer_count, COUNT (DISTINCT
customers_country_cte.customer_id) 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
customers_country_cte ON D.country = customers_country_cte.country GROUP BY
D.country;

```

The screenshot shows the pgAdmin 4 interface with a SQL query editor and a data output window. The query is a complex SQL statement using a Common Table Expression (CTE) to calculate the total amount paid by customer and city, and then rank the countries by total amount paid.

**Query Editor:**

```

1 WITH customers_country_cte AS (SELECT B. customer_id, B. first_name, B. last_name, D. city, country, SUM(amount) AS total_amount_paid
2 FROM payment A
3 INNER JOIN customer B ON A. customer_id = B. customer_id INNER JOIN address C ON B. address_id = C. address_id
4 INNER JOIN city D ON C. city_id = D. city_id INNER JOIN country E ON D. country_id = E. country_id
5 WHERE D. city IN ('Aurora','Tokat','Tarsus','Atlixco','Emeishan','Pontianak','Shimoga','Aparecida de Goinia','Zalantun','Taguig')
6 GROUP BY B. customer_id, B. first_name, B. last_name, D. city, country ORDER BY total_amount_paid DESC LIMIT 5) SELECT D.country,
7 COUNT (DISTINCT A.customer_id) AS all_customer_count, COUNT (DISTINCT customers_country_cte.customer_id) AS top_customer_count
8 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
9 INNER JOIN country D ON C.country_id = D.country_id LEFT JOIN customers_country_cte ON D.country = customers_country_cte.country
10 GROUP BY D.country;

```

**Data Output:**

country	all_customer_count	top_customer_count
1 Afghanistan	1	0
2 Algeria	3	0
3 American Samoa	1	0
4 Angola	2	0
5 Anguilla	1	0
6 Argentina	13	0
7 Armenia	1	0
8 Austria	3	0
9 Azerbaijan	2	0
10 Bahrain	1	0
11 Bangladesh	3	0
12 Belarus	2	0
13 Bolivia	2	0

**Messages:**

Successfully run. Total query runtime: 310 msec.  
108 rows affected.

2A. Subquery may be more straightforward than CTE.

## 2B. CTE COST AND SPEED

### EXPLAIN AND SPEED TIME FOR CTE (STEP 1 QUERY)

The screenshot shows the pgAdmin interface with a SQL query in the Query Editor. The query uses a CTE named 'average\_cte' to calculate the average amount paid per country. The EXPLAIN output is displayed in a pop-up window.

```
1 EXPLAIN WITH average_cte AS (SELECT B. customer_id, B. first_name, B. last_name, D. city, country, SUM(amount) AS total_amount_paid FROM
2 INNER JOIN customer B ON A. customer_id = B. customer_id INNER JOIN address C ON B. address_id = C. address_id
3 INNER JOIN city D ON C. city_id = D. city_id INNER JOIN country E ON D. country_id = E. country_id
4 WHERE D. city IN ('Aurora','Tokat','Tarsus','Atlitxco','Emeishan','Pontianak','Shimoga','Aparecida de Goiania','Zalantun','Taguig') GROUP BY
5 ORDER BY total_amount_paid DESC LIMIT 5) SELECT (AVG (total_amount_paid), 2) AS average_amount_paid FROM average_cte;
```

**EXPLAIN**

QUERY PLAN
Aggregate (cost=64.30..64.31 rows=1 width=32)
[-]> Limit (cost=64.22..64.23 rows=5 width=67)
[-]> Sort (cost=64.22..64.82 rows=239 width=67)
[-] Sort Key: (sum(a.amount)) DESC
[-]> HashAggregate (cost=57.26..60.25 rows=239 width=67)
[-] Group Key: b.customer_id, d.city, e.country
[-]> Nested Loop (cost=18.16..54.87 rows=239 width=41)
[-]> Hash Join (cost=17.88..37.14 rows=10 width=35)
[-] Hash Cond: (d.country_id = e.country_id)
[-]> Nested Loop (cost=14.43..33.66 rows=10 width=28)
[-]> Hash Join (cost=14.15..29.77 rows=10 width=15)
[-] Hash Cond: (c.city_id = d.city_id)

**Messages**

Successfully run. Total query runtime: 80 msec.  
22 rows affected.

✓ Successfully run. Total query runtime: 80 msec. 22 rows affected.

### ii. EXPLAIN AND SPEED TIME FOR CTE (STEP 2 QUERY)

The screenshot shows the pgAdmin interface with a SQL query in the Query Editor. The query uses a CTE named 'customers\_country\_cte' to calculate the top customer count per country. The EXPLAIN output is displayed in a pop-up window.

```
1 EXPLAIN WITH customers_country_cte AS (SELECT B. customer_id, B. first_name, B. last_name, D. city, country, SUM(amount) AS total_amount
2 FROM payment A
3 INNER JOIN customer B ON A. customer_id = B. customer_id INNER JOIN address C ON B. address_id = C. address_id
4 INNER JOIN city D ON C. city_id = D. city_id INNER JOIN country E ON D. country_id = E. country_id
5 WHERE D. city IN ('Aurora','Tokat','Tarsus','Atlitxco','Emeishan','Pontianak','Shimoga','Aparecida de Goiania','Zalantun','Taguig')
6 GROUP BY B. customer_id, B. first_name, B. last_name, D. city, country ORDER BY total_amount_paid DESC LIMIT 5) SELECT D.country,
7 COUNT (DISTINCT A.customer_id) AS all_customer_count, COUNT (DISTINCT customers_country_cte.customer_id) AS top_customer_count
8 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
9 INNER JOIN country D ON C.country_id = D.country_id LEFT JOIN customers_country_cte ON D.country = customers_country_cte.country
10 GROUP BY D.country;
```

**EXPLAIN**

QUERY PLAN
GroupAggregate (cost=155.28..164.26 rows=109 width=25)
[-] Group Key: d.country
[-]> Merge Left Join (cost=155.28..158.68 rows=599 width=17)
[-] Merge Cond: ((d.country)::text = (customers_country_cte.country)::text)
[-]> Sort (cost=90.94..92.44 rows=599 width=13)
[-] Sort Key: d.country
[-]> Hash Join (cost=43.52..63.30 rows=599 width=13)
[-] Hash Cond: (c.country_id = d.country_id)
[-]> Hash Join (cost=40.07..58.22 rows=599 width=6)
[-] Hash Cond: (b.city_id = c.city_id)
[-]> Hash Join (cost=21.57..38.14 rows=599 width=6)
[-] Hash Cond: (a.address_id = b.address_id)

**Messages**

Successfully run. Total query runtime: 100 msec.  
43 rows affected.

✓ Successfully run. Total query runtime: 100 msec. 43 rows affected.

## SUBQUERY COST AND SPEED (QUERY 1)

The screenshot shows the pgAdmin interface with the Query Editor open. The query is as follows:

```
1 EXPLAIN SELECT AVG (total_amount_paid) AS average_amount_paid FROM (SELECT B. customer_id, B. first_name, B. last_name, D. city, country
2 INNER JOIN customer B ON A. customer_id = B. customer_id INNER JOIN address C ON B. address_id = C. address_id
3 INNER JOIN city D ON C. city_id = D. city_id INNER JOIN country E ON D. country_id = E. country_id
4 WHERE D. city IN ('Aurora','Tokat','Tarsus','Atlixco','Emeishan','Pontianak','Shimoga','Aparecida de Goiania','Zalantun','Taguig') GROUP
5 ORDER BY total_amount_paid DESC LIMIT 5) AS AVERAGE
```

The Query Plan shows the following steps:

Step	Operation	Cost	Rows	Width
1	Aggregate	(cost=64.30..64.31 rows=1 width=32)	1	32
2	Limit	(cost=64.22..64.23 rows=5 width=67)	5	67
3	Sort	(cost=64.22..64.82 rows=239 width=67)	239	67
4	Sort Key	(sum(a.amount)) DESC		
5	HashAggregate	(cost=57.26..60.25 rows=239 width=67)	239	67
6	Group Key	b.customer_id, d.city, e.country		
7	Nested Loop	(cost=18.16..54.87 rows=239 width=41)	239	41
8	Hash Join	(cost=17.88..37.14 rows=10 width=35)	10	35
9	Hash Cond	(d.country_id = e.country_id)		
10	Nested Loop	(cost=14.43..33.66 rows=10 width=28)	10	28
11	Hash Join	(cost=14.15..29.77 rows=10 width=15)	10	15
12	Hash Cond	(c.city_id = d.city_id)		

Messages:

Successfully run. Total query runtime: 79 msec.  
22 rows affected.

Successfully run. Total query runtime: 79 msec. 22 rows affected.

## QUERY 2

The screenshot shows the pgAdmin interface with the Query Editor open. The query is as follows:

```
1 EXPLAIN SELECT country.country, COUNT(DISTINCT customer.customer_id) AS customer_count, COUNT(DISTINCT country.country) AS top_customer_
2 FROM (SELECT B. customer_id, B. first_name, B. last_name, D. city, country,
3 SUM(amount) AS total_amount_paid FROM payment A INNER JOIN customer B ON A. customer_id = B. customer_id INNER JOIN address
4 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
5 WHERE D. city IN ('Aurora','Tokat','Tarsus','Atlixco','Emeishan','Pontianak','Shimoga','Aparecida de Goiania','Zalantun','Taguig') GROUP
6 ORDER BY total_amount_paid DESC LIMIT 5) AS top_customer_count LEFT JOIN customer ON customer.customer_id = customer.customer_id
7 LEFT JOIN address ON customer.address_id = address.address_id LEFT JOIN city ON address.city_id = city.city_id
8 LEFT JOIN country ON city.country_id = country.country_id GROUP BY country.country ORDER BY COUNT(country.country) DESC LIMIT 5
```

The Query Plan shows the following steps:

Step	Operation	Cost	Rows	Width
1	Limit	(cost=379.79..379.81 rows=5 width=33)	5	33
2	Sort	(cost=379.79..380.07 rows=109 width=33)	109	33
3	Sort Key	(count(country.country)) DESC		
4	GroupAggregate	(cost=339.46..377.98 rows=109 width=33)	109	33
5	Group Key	country.country		
6	Sort	(cost=339.46..346.94 rows=2995 width=13)	2995	13
7	Sort Key	country.country		
8	Nested Loop Left Join	(cost=107.74..166.52 rows=2995 width=13)	2995	13
9	Limit	(cost=64.22..64.23 rows=5 width=67)	5	67
10	Sort	(cost=64.22..64.82 rows=239 width=67)	239	67
11	Sort Key	(sum(a.amount)) DESC		
12	HashAggregate	(cost=57.26..60.25 rows=239 width=67)	239	67

Messages:

Successfully run. Total query runtime: 298 msec.  
44 rows affected.

Successfully run. Total query runtime: 298 msec. 44 rows affected.

2D. Quite surprised as the CTE appears faster to run than the subquery but I presume the subquery is more straightforward than the CTE.

3. There are complexities around writing the two approaches due to the technicalities. The main challenge is the development of the subqueries (coming up with the INNER JOINS of different keys/tables) that were nested in the CTEs. I believe being able to overcome the challenges around writing the subquery itself may get the job done as I presume the subqueries are more straightforward than the CTEs.