

Performing Subqueries

Step 1

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer displays the database structure for 'Rockbuster', including tables like 'actor', 'address', 'category', 'city', 'country', 'customer', 'employees', 'film', 'film_actor', 'film_category', 'inventory', 'language', and 'payment'. The main pane shows a SQL query with multiple subqueries and joins. The query calculates the average total amount paid by customers, grouped by country and city. The results pane at the bottom shows a single row with the average value of 105.55400000000000.

```
1 SELECT AVG(total_amount_payment) AS average FROM
2 (SELECT C.customer_id, C.first_name, C.last_name, CO.country, CI.city, SUM(P.amount) AS
3 total_amount_payment FROM payment AS P
4 INNER JOIN customer AS C ON C.customer_id = P.customer_id
5 INNER JOIN address AS A ON A.address_id = C.address_id
6 INNER JOIN city AS CI ON CI.city_id = A.city_id
7 INNER JOIN country CO ON CO.country_id = CI.country_id
8 WHERE CI.city IN (
9 SELECT CI.city FROM customer AS C
10 INNER JOIN address AS A ON A.address_id = C.address_id
11 INNER JOIN city AS CI ON CI.city_id = A.city_id
12 INNER JOIN country CO ON CO.country_id = CI.country_id
13 WHERE CO.country IN (
14 SELECT CO.country FROM customer AS C
15 INNER JOIN address AS A ON A.address_id = C.address_id
16 INNER JOIN city AS CI ON CI.city_id = A.city_id
17 INNER JOIN country CO ON CO.country_id = CI.country_id
18 GROUP BY CO.country
19 ORDER BY COUNT(C.customer_id) DESC
20 LIMIT 10)
21 GROUP BY CO.country, CI.city
22 ORDER BY Count(C.customer_id) DESC]
23 LIMIT 10)
24 GROUP BY C.customer_id, CO.country, CI.city
25 ORDER BY total_amount_payment DESC
26 LIMIT 5) AS total_amount_paid
```

average
105.55400000000000

Total rows: 1 of 1 Query complete 00:00:00.114 Ln 22, Col 35

Step 2

```
SELECT CO.country, COUNT(DISTINCT C.customer_id) AS all_customer_count,
COUNT(DISTINCT top_5_customers.customer_id) AS top_customer_count FROM customer
AS C
INNER JOIN address AS A ON A.address_id = C.address_id
INNER JOIN city AS CI ON CI.city_id = A.city_id
INNER JOIN country CO ON CO.country_id = CI.country_id
LEFT JOIN
(SELECT C.customer_id, C.first_name, C.last_name, CO.country, CI.city, SUM(P.amount)
AS total_amount_payment FROM payment AS P
INNER JOIN customer AS C ON C.customer_id=P.customer_id
INNER JOIN address AS A ON A.address_id = C.address_id
INNER JOIN city AS CI ON CI.city_id = A.city_id
INNER JOIN country CO ON CO.country_id = CI.country_id
WHERE CI.city IN (
SELECT CI.city FROM customer AS C
INNER JOIN address AS A ON A.address_id = C.address_id
INNER JOIN city AS CI ON CI.city_id = A.city_id
INNER JOIN country CO ON CO.country_id = CI.country_id
WHERE CO.country IN(
SELECT Co.country FROM customer AS C
INNER JOIN address AS A ON A.address_id=C.address_id
INNER JOIN city AS CI ON CI.city_id=A.city_id
```

```

INNER JOIN country CO ON CO.country_id=CI.country_id
GROUP BY CO.country
ORDER BY COUNT(C.customer_id) DESC
LIMIT 10)
GROUP BY CO.country,CI.city
ORDER BY Count(C.customer_id) DESC
LIMIT 10)
GROUP BY C.customer_id,CO.country, CI.city
ORDER BY total_amount_payment DESC
LIMIT 5) AS top_5_customers ON top_5_customers.country=CO.country
GROUP BY CO.country
ORDER BY top_customer_count DESC;

```

The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer showing the database structure. The main pane displays a SQL query with multiple subqueries and joins. Below the query, the 'Data Output' tab shows the results of the query as a table.

	country	all_customer_count	top_customer_count
	character varying (50)	bigint	bigint
1	Mexico	30	1
2	India	60	1
3	China	53	1
4	United States	36	1
5	Japan	31	1
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
14	Brazil	28	0

Total rows: 108 of 108 Query complete 00:00:00.108

Step 3

Do you think steps 1 and 2 could be done without using subqueries? When do you think subqueries are useful?

Honestly, my experience with Postgre is not extensive at all and very limited at this stage in the course. So, steps 1 and 2 seem a bit long and complicated, making them difficult to understand and interpret for beginners like me. I do think more seasoned analysts could find more simpler ways to produce the same results. But to obtain the most current results, subqueries are necessary. Subqueries are useful in various ways and scenarios where an analyst would need to perform complex queries or extract data from multiple tables, or where the data in the database is always changing.