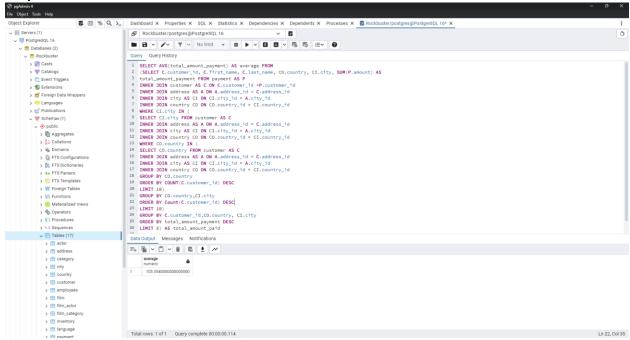
Performing Subqueries

Step 1



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 Cl.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, Cl.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 Cl.city_id = A.city_id

INNER JOIN country CO ON CO.country_id = Cl.country_id

WHERE Cl.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 Cl.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 Cl.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; 🛢 🖩 🖫 😘 Q 🚬 Dashboard x Properties x SQL x Statistics x Dependencies x Dependents x Processes x Servers (1) Ouery Ouery History

1 SELECT CO.country, COUNT(DISTINCT C.customer_id) AS all_customer_count,
2 COUNT(DISTINCT Top_S_customers_customer_id) AS top_customer_count FROM customer
3 AS C
4 INNER JOIN address AS A ON A.address_id = C.address_id
5 INNER JOIN customer AS C ON C.lcity_id = A.city_id
6 INNER JOIN customer_id, C.ffrat_name, C.last_name, Co.country_if
8 (SELECT G.customer_id, C.ffrat_name, C.last_name, Co.country_id
9 AS total_amount_payment FROM payment AS P
1 INNER JOIN customer AS C ON C.customer_id=Customer_id
1 INNER JOIN customer AS C ON C.customer_id = C.address_id
1 INNER JOIN customer AS C ON C.customer_id = C.rountry_id
1 INNER JOIN customer AS C ON C.customer_id = C.rountry_id
1 INNER JOIN customer AS C ON C.customer_id = C.rountry_id
1 INNER JOIN customer AS C
1 INNER JOIN customer AS C
2 INNER JOIN customer AS C
3 INNER JOIN customer AS C
4 INNER JOIN customer AS C
5 INNER JOIN customer AS C
6 INNER JOIN customer AS C
6 INNER JOIN customer AS C
6 INNER JOIN customer AS C
7 INNER JOIN customer AS C
7 INNER JOIN customer AS C
8 INNER JOIN c Query Query History v 🚍 Rockbuster > 🐼 Casts > 1 Extensions
> 1 Foreign Data Wrappers > 🖺 Aggregates > N FTS Dictionaries > As FTS Parsers > @ FTS Templates
> @ Foreign Tables > (ii) Functions = 6 0 0 0 > R Materialized Views country | all_customer_count | top_customer_count | bigint | top_customer_count | bigint | > 1.3 Sequences 2 India 3 China 4 United States
5 Japan
6 Argentina
7 Armenia > 🛗 city 11 Banglade 12 Belarus 13 Bolivia > III film_actor > ## film category 14 Brazil
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.