

1. Copy the query you wrote in step 3 of the task from [Exercise 3.7: Joining Tables of Data](#) into the Query Tool. This will be your subquery, so give it an alias, "total\_amount\_paid," and add parentheses around it.
2. Write an outer statement to calculate the average amount paid.
3. Add your subquery to the outer statement. It will go in either the SELECT, WHERE, or FROM clause. (Hint: When referring to the subquery in your outer statement, make sure to use the subquery's alias, "total\_amount\_paid".)
4. If you've done everything correctly, pgAdmin 4 will require you to add an alias after the subquery. Go ahead and call it "average".

1.

```
SELECT B.customer_id,  
B.first_name,  
B.last_name,  
D.city,  
E.country,  
SUM(A.amount) AS Total_Amount_Paid  
FROM customer B  
INNER JOIN payment A ON B.customer_id = A.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  
GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country ORDER BY Total_Amount_Paid  
DESC LIMIT 5
```

2.

The screenshot shows a PostgreSQL query editor interface. The top menu bar includes Dashboard, Properties, SQL, Statistics, Dependencies, and Dependents. The current tab is titled 'Rockbuster/postgres@PostgreSQL 15\*'. Below the menu bar, there's a toolbar with various icons for file operations, query execution, and settings. The main area displays a SQL query with line numbers 1 through 15. The query is a complex SELECT statement with multiple joins and a subquery. The output pane at the bottom shows a single row of data with the column name 'average' and a numeric value. The status bar at the bottom indicates 'Total rows: 1 of 1' and 'Query complete 00:00:00.441'.

```
1 SELECT AVG(total_amount_paid) AS average
2 FROM (SELECT B.customer_id,
3 B.first_name,
4 B.last_name,
5 D.city,
6 E.country,
7 SUM(A.amount) AS Total_Amount_Paid
8 FROM customer B
9 INNER JOIN payment A ON B.customer_id = A.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 GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country
14 ORDER BY Total_Amount_Paid DESC
15 LIMIT 5) AS total_amount_paid
```

average
199.1920000000000000

Total rows: 1 of 1 Query complete 00:00:00.441 Ln 15, Col 30

**Step 2: Find out how many of the top 5 customers are based within each country.**

Your final output should include 3 columns:

- "country"
- "all\_customer\_count" with the total number of customers in each country
- "top\_customer\_count" showing how many of the top 5 customers live in each country

The screenshot shows a PostgreSQL query editor window with the following SQL query:

```

1 SELECT top_5_customers.country,
2 COUNT (top_5_customers.country) AS top_customer_count
3 FROM (SELECT
4 A.customer_id, A.first_name, A.last_name, C.city, D.country, SUM(F.amount) AS total_paid_to_rockbuster
5 FROM customer A
6 INNER JOIN rental E ON A.customer_id = E.customer_id
7 INNER JOIN payment F ON E.rental_id = F.rental_id
8 INNER JOIN address B ON A.address_id = B.address_id
9 INNER JOIN city C ON B.city_ID = C.city_id
10 INNER JOIN country D on C.country_ID = D.country_ID
11 GROUP BY a.customer_id, A.first_name, A.last_name, C.city, D.country
12 ORDER BY total_paid_to_rockbuster DESC
13 LIMIT 5) AS top_5_customers
14 GROUP BY top_5_customers.country

```

The results are displayed in a table with the following columns: country (character varying (50)) and top\_customer\_count (bigint). The results show 5 rows of data:

country	top_customer_count
1 Belarus	1
2 Brazil	1
3 Netherlands	1
4 Runion	1

Total rows: 5 of 5 Query complete 00:00:00.534 Ln 14, Col 33

```

SELECT all_customers.country,
       all_customers.all_customer_count,
       top_customers.top_customer_count
FROM
(SELECT DISTINCT D.COUNTRY,
                COUNT (DISTINCT A.customer_id) AS all_customer_count
FROM Customer A
        INNER JOIN rental E ON A.customer_id = E.customer_id
        INNER JOIN payment F ON E.rental_id = F.rental_id
        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

```

```

GROUP BY country
ORDER BY all_customer_count DESC) AS all_customers

INNER JOIN

(SELECT top_5_customers.country, COUNT (top_5_customers.customer_id) AS top_customer_count
FROM (SELECT
        A.customer_id,
        A.first_name,
        A.last_name,
        C.city,
        D.country,
        SUM(F.amount) AS total_paid_to_rockbuster
FROM customer A
        INNER JOIN rental E ON A.customer_id = E.customer_id
        INNER JOIN payment F ON E.rental_id = F.rental_id
        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
GROUP BY a.customer_id, A.first_name, A.last_name, C.city, D.country
ORDER BY total_paid_to_rockbuster DESC
LIMIT 5)AS top_5_customers
GROUP BY top_5_customers.country) AS top_customers
ON all_customers.country = top_customers.country

```

```

1 SELECT all_customers.country,
2       all_customers.all_customer_count,
3       top_customers.top_customer_count
4 FROM
5 (SELECT DISTINCT O.COUNTRY,
6  COUNT (DISTINCT A.customer_id) AS all_customer_count
7 FROM Customer A
8      INNER JOIN rental E ON A.customer_id = E.customer_id
9      INNER JOIN payment F ON E.rental_id = F.rental_id
10     INNER JOIN address B ON A.address_id = B.address_id
11     INNER JOIN city C ON E.city_ID = C.city_id
12     INNER JOIN country D on C.country_ID = D.country_ID
13 GROUP BY country
14 ORDER BY all_customer_count DESC) AS all_customers
15 INNER JOIN
16 (SELECT top_5_customers.country, COUNT (top_5_customers.customer_id) AS top_customer_count
17 FROM (SELECT
18       A.customer_id,
19       A.first_name,
20       A.last_name,
21       C.city,
22       D.country,
23       SUM(F.amount) AS total_paid_to_rockbuster
24 FROM customer A

```

```

10     INNER JOIN address B ON A.address_id = B.address_id
11     INNER JOIN city C ON B.city_ID = C.city_id
12     INNER JOIN country D on C.country_ID = D.country_ID
13 GROUP BY country
14 ORDER BY all_customer_count DESC) AS all_customers
15 INNER JOIN
16 (SELECT top_5_customers.country, COUNT (top_5_customers.customer_id) AS top_customer_count
17 FROM (SELECT
18       A.customer_id,
19       A.first_name,
20       A.last_name,
21       C.city,

```

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	United States	36	1
2	Brazil	28	1
3	Netherlands	5	1
4	Belarus	2	1
5	Runion	1	1

- Write 1 to 2 short paragraphs on the following:
  - Do you think steps 1 and 2 could be done without using subqueries?
  - When do you think subqueries are useful?

- There's no way I could figure out how to do them without using subqueries (at least not while keeping them dynamic so that the results update if the database updates). Maybe with more practice I would eventually figure it out. Step 1 - average of top 5 customers should be the easier of the two to figure out how to do it without subqueries... but I would think that ordering and limiting for top 5 would already be the subquery, and I'm not sure how to do an "AVG" of limited results without having those limits be the subquery itself.
- I think subqueries are useful and necessary any time you want to know aggregated or limited details about anything within a database that is changing/updated on a regular basis.