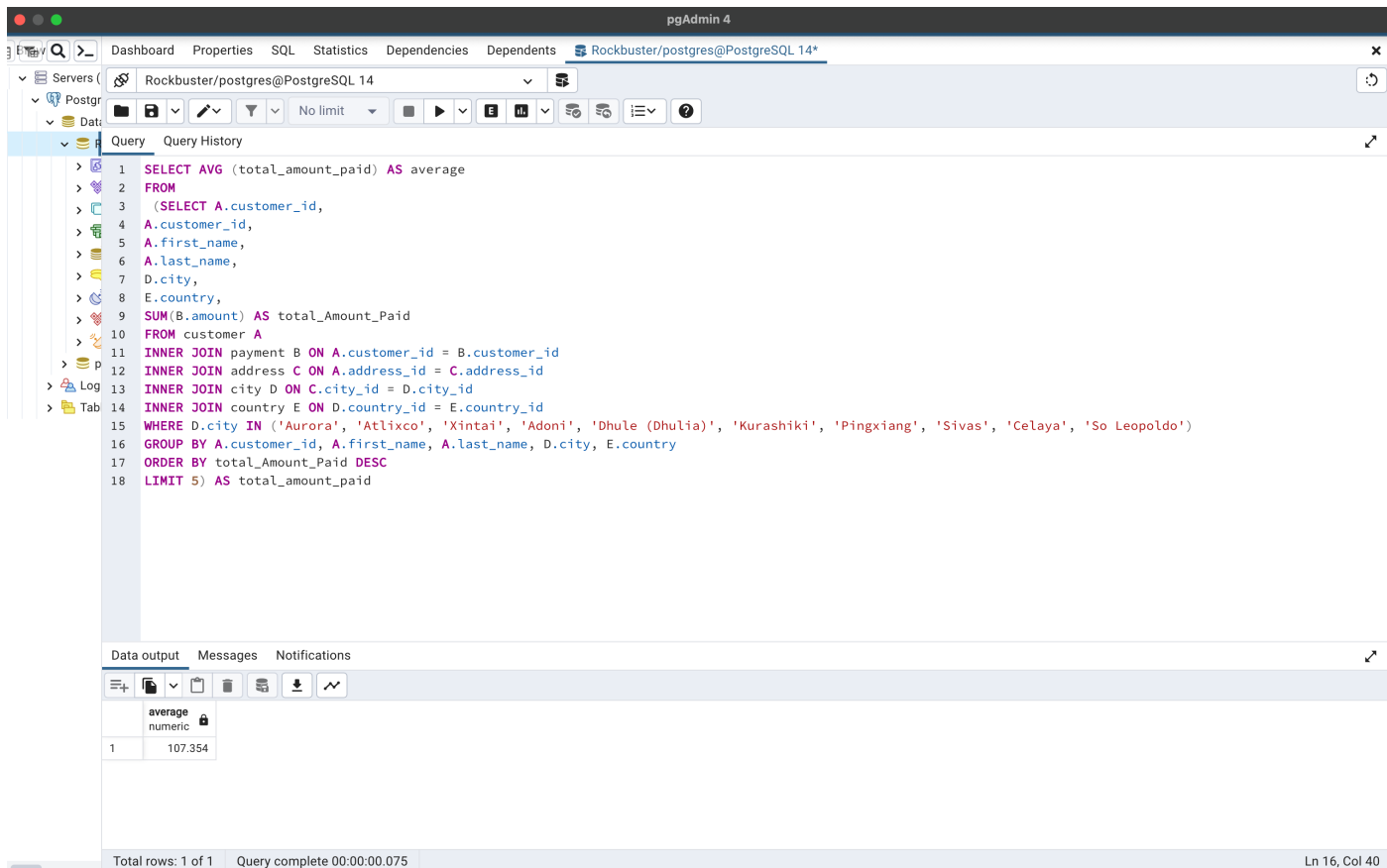


3.8: Performing Subqueries

Alejandro Salgado

Step 1: Find the average amount paid by the top 5 customers.

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”.



The screenshot shows the pgAdmin 4 interface. The top toolbar includes buttons for Dashboard, Properties, SQL, Statistics, Dependencies, and Dependents. The main window displays a SQL query in the Query editor, which is a complex query involving multiple joins and a subquery. The query is as follows:

```
1 SELECT AVG (total_amount_paid) AS average
2 FROM
3 (SELECT A.customer_id,
4 A.customer_id,
5 A.first_name,
6 A.last_name,
7 D.city,
8 E.country,
9 SUM(B.amount) AS total_Amount_Paid
10 FROM customer A
11 INNER JOIN payment B ON A.customer_id = B.customer_id
12 INNER JOIN address C ON A.address_id = C.address_id
13 INNER JOIN city D ON C.city_id = D.city_id
14 INNER JOIN country E ON D.country_id = E.country_id
15 WHERE D.city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)', 'Kurashiki', 'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
16 GROUP BY A.customer_id, A.first_name, A.last_name, D.city, E.country
17 ORDER BY total_Amount_Paid DESC
18 LIMIT 5) AS total_amount_paid
```

The bottom panel shows the Data output tab, which displays the results of the query. The results are as follows:

average	numeric
1	107.354

The bottom status bar indicates "Total rows: 1 of 1" and "Query complete 00:00:00.075". The bottom right corner shows "Ln 16, Col 40".

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

You'll notice that this step is quite difficult. We've broken down each part and provided you with some helpful hints below:

1. Copy the query from step 3 of task 3.7 into the Query Tool and add parentheses around it. This will be your inner query.
2. Write an outer statement that counts the number of customers living in each country. You'll need to refer to your entity relationship diagram or data dictionary in order to do this. The information you need is in different tables, so you'll have to use a join. To get the count for each country, use **COUNT(DISTINCT)** and **GROUP BY**. Give your second column the alias “all_customer_count” for readability.
3. Place your inner query in the outer query. Since you want to merge the entire output of the outer query with the information from your inner query, use a left join to connect the two queries on the “country” column.
4. Add a left join after your outer query, followed by the subquery in parentheses.
5. Give your subquery an alias so you can refer to it in your outer query, for example, “top_5_customers”.
6. Remember to specify which columns to join the two tables on using **ON**. Both **ON** and the column names should follow the alias.
7. Count the top 5 customers for the third column using **GROUP BY** and **COUNT (DISTINCT)**. Give this column the alias “top_customer_count”.

pgAdmin 4

Dashboard Properties SQL Statistics Dependencies Dependents Rockbuster/postgres@PostgreSQL 14*

Servers (PostgreSQL) Rockbuster/postgres@PostgreSQL 14

Query Query History

```

1 SELECT DISTINCT(A.country),
2 COUNT (DISTINCT D. customer_id) AS all_customer_count,
3 COUNT (DISTINCT A.country) AS top_customer_count
4 FROM country A
5 INNER JOIN city B ON A. country_id = B. country_id
6 INNER JOIN address C ON B.city_id =C.city_id
7 INNER JOIN customer D ON C.address_id =D. address_id
8 LEFT JOIN(SELECT A.customer_id, A.first_name, A.last_name,
9 D. country, C.city,
10 SUM(E.amount) AS Payment
11 FROM customer A
12 INNER JOIN address B ON A.address_id = B.address_id
13 INNER JOIN city C ON B.city_id = C.city_id
14 INNER JOIN country D ON C.country_id = D. country_id
15 INNER JOIN payment E ON A.customer_id = E.customer_id
16 GROUP BY A.customer_id, D.country,C.city
17 ORDER BY Payment DESC
18 LIMIT 5) AS top_5_customers
19 ON A.country=top_5_customers. COUNTRY
20 GROUP BY A.country, top_5_customers
21 ORDER BY all_customer_count DESC
22 LIMIT 5
23

```

Data output Messages Notifications

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

Total rows: 5 of 5 Query complete 00:00:00.066

Step 3:

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?

For the first task a subquery was not necessary. For the second one, it would need subqueries as it is pulling from different databases. Sub queries can be useful when comparing data from multiple databases. You can compare datapoints and make new columns which can be useful when you need to analyze results from a complex query.