

ANSWERS 3.8

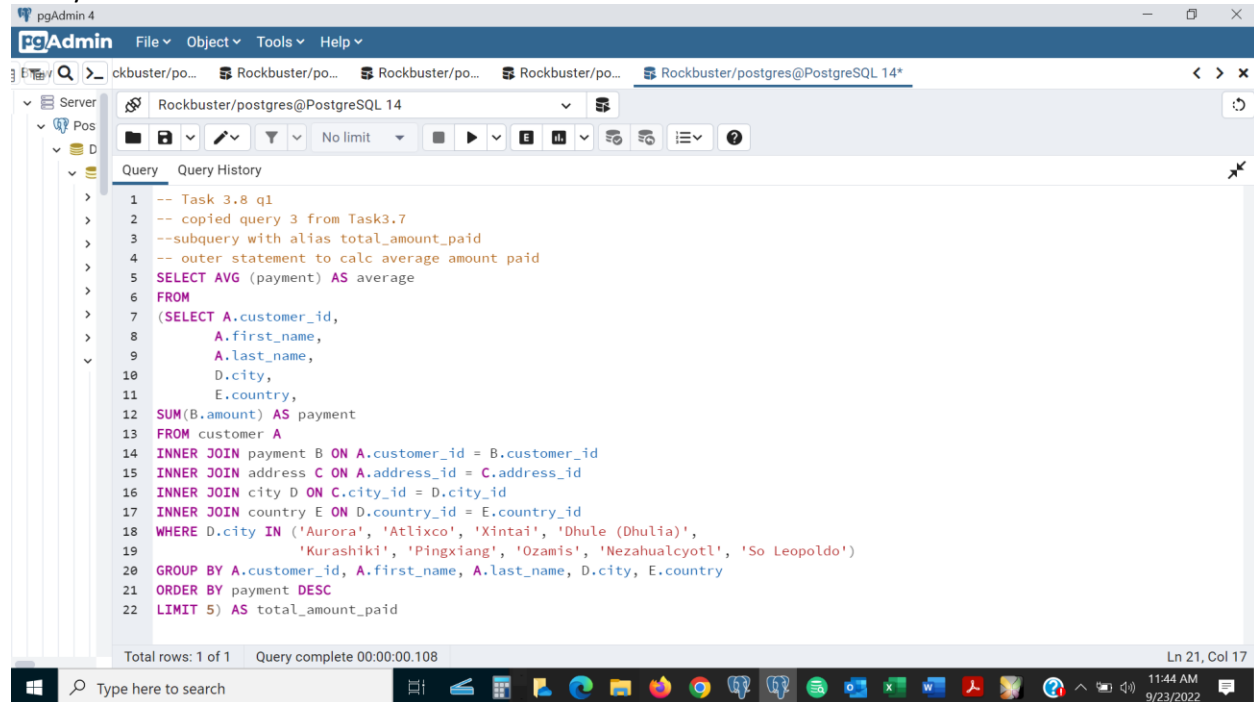
Directions

Create a new text document and call it “Answers 3.8.” You’ll be copy-pasting your queries, outputs, and written answers into this document, as you’ve done in previous tasks.

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”.
5. Copy-paste your queries and the final data output from pgAdmin 4 into your answers document.

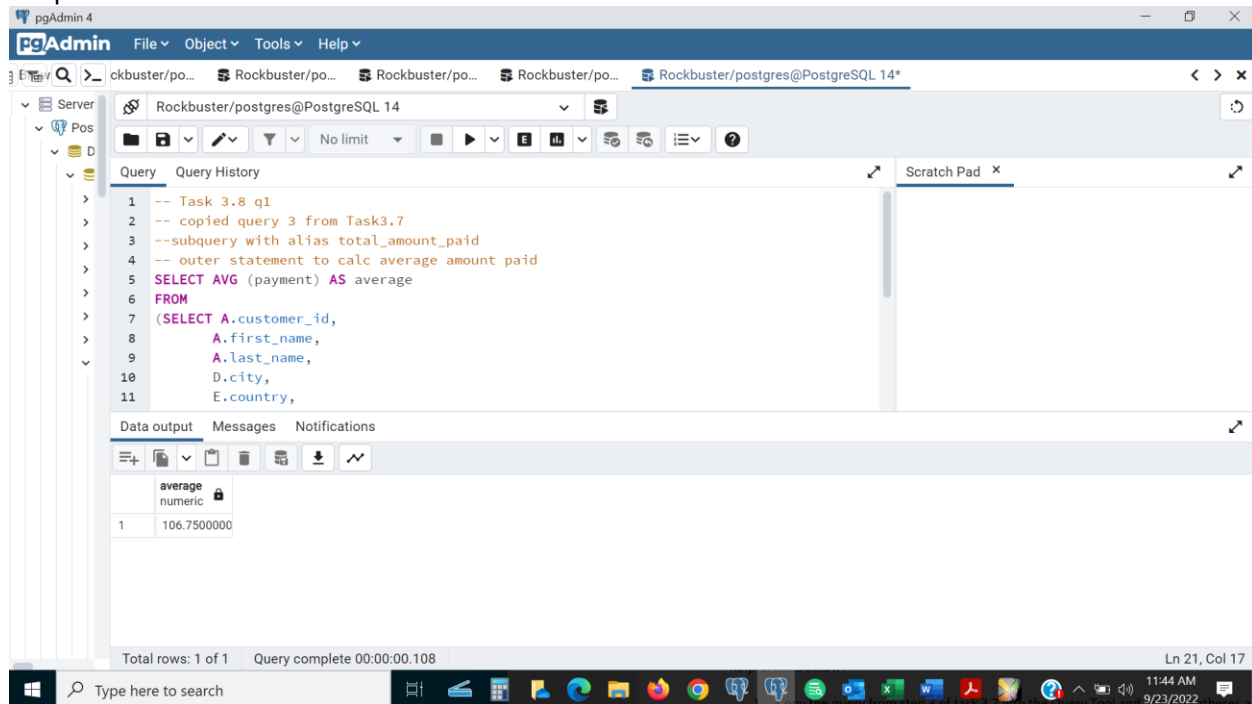
Query1



```
1 -- Task 3.8 q1
2 -- copied query 3 from Task3.7
3 --subquery with alias total_amount_paid
4 -- outer statement to calc average amount paid
5 SELECT AVG (payment) AS average
6 FROM
7 (SELECT A.customer_id,
8      A.first_name,
9      A.last_name,
10     D.city,
11     E.country,
12     SUM(B.amount) AS payment
13  FROM customer A
14  INNER JOIN payment B ON A.customer_id = B.customer_id
15  INNER JOIN address C ON A.address_id = C.address_id
16  INNER JOIN city D ON C.city_id = D.city_id
17  INNER JOIN country E ON D.country_id = E.country_id
18  WHERE D.city IN ('Aurora', 'Atlixco', 'Xintai', 'Dhule (Dhulia)',
19                  'Kurashiki', 'Pingxiang', 'Ozamis', 'Nezahualcyotl', 'So Leopoldo')
20  GROUP BY A.customer_id, A.first_name, A.last_name, D.city, E.country
21  ORDER BY payment DESC
22  LIMIT 5) AS total_amount_paid
```

Total rows: 1 of 1 Query complete 00:00:00.108 Ln 21, Col 17

Output1



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

Your final output should include 3 columns:

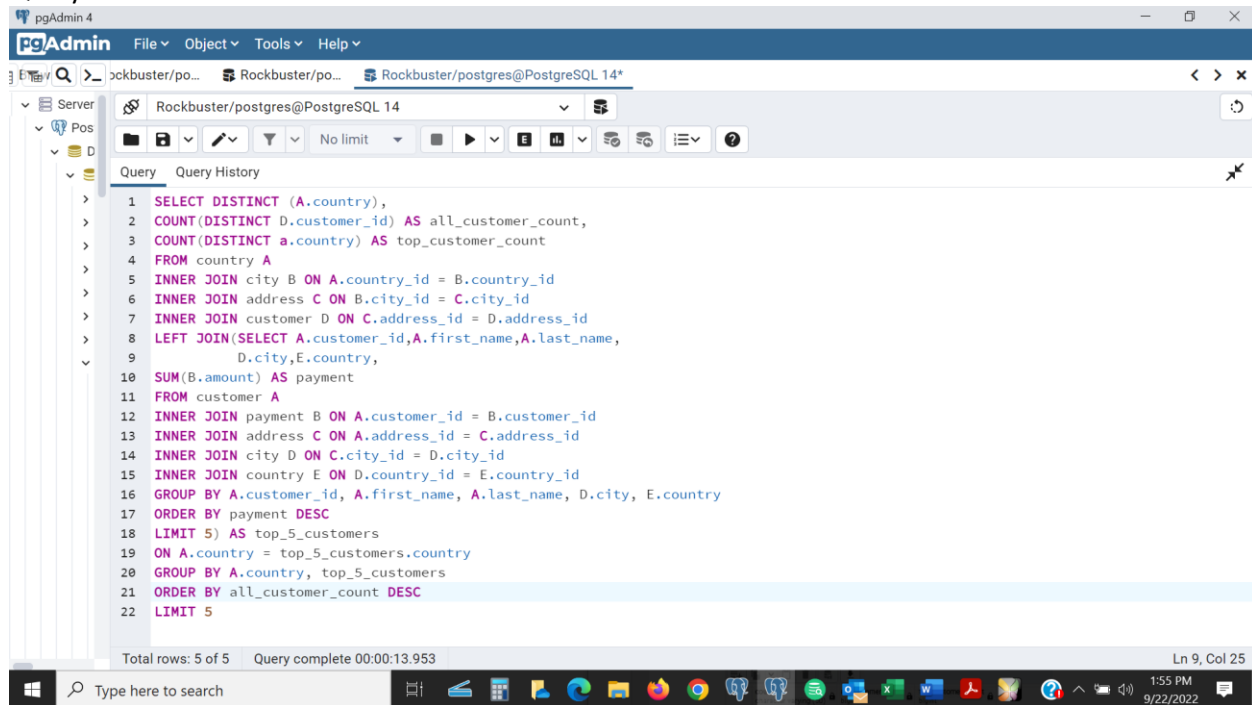
- "country"
- "all_customer_count" with the total number of customers in each country
- "top_customer_count" showing how many of the top 5 customers live in each country

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".

- Remember to specify which columns to join the two tables on using ON. Both ON and the column names should follow the alias.
- Count the top 5 customers for the third column using GROUP BY and COUNT (DISTINCT). Give this column the alias "top_customer_count".
- Copy-paste your query and the data output into your "Answers 3.8" document.

Query2



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure: Server > Pos > D > Query. The main pane shows a SQL query with 22 lines. The query is a complex join of several tables (country, city, address, customer, payment) to find the top 5 customers by payment amount, grouped by country. The status bar at the bottom indicates 'Total rows: 5 of 5' and 'Query complete 00:00:13.953'.

```
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.city,E.country,
10 SUM(B.amount) AS payment
11 FROM customer A
12 INNER JOIN payment B ON A.customer_id = B.customer_id
13 INNER JOIN address C ON A.address_id = C.address_id
14 INNER JOIN city D ON C.city_id = D.city_id
15 INNER JOIN country E ON D.country_id = E.country_id
16 GROUP BY A.customer_id, A.first_name, A.last_name, D.city, E.country
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
```

Total rows: 5 of 5 Query complete 00:00:13.953 Ln 9, Col 25

Output2

The screenshot shows the pgAdmin 4 interface with a query window open. The query is as follows:

```
12 INNER JOIN payment B ON A.customer_id = B.customer_id
13 INNER JOIN address C ON A.address_id = C.address_id
14 INNER JOIN city D ON C.city_id = D.city_id
15 INNER JOIN country E ON D.country_id = E.country_id
16 GROUP BY A.customer_id, A.first_name, A.last_name, D.city, E.country
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
```

The results are displayed in the Data output tab, showing a table with 5 rows and 3 columns:

	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 00:00:13.953 Ln 9, Col 25

Step 3:

- Write 1 to 2 short paragraphs on the following:
 - Do you think steps 1 and 2 could be done without using subqueries?
 - First query does not need to use subqueries, for we are accessing one table and could use an aggregate function. Subquery if used may cause slower response times.
 - Step 2 does require subqueries for result is being pulled from different tables, creating a new table, then evaluating results from the query against other table, and you want to make sure that the results in table are up to date.
 - When do you think subqueries are useful?
 - Subqueries are useful when analyzing results of complex queries (summarizing of data), results from tables may be changing and by need to be up to date so subqueries are one way to have current results returned.