**Task 8: Sub-queries**

**Answers 3.8**

**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:

   Outer statement was initially SELECT AVG (amount)

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”.)

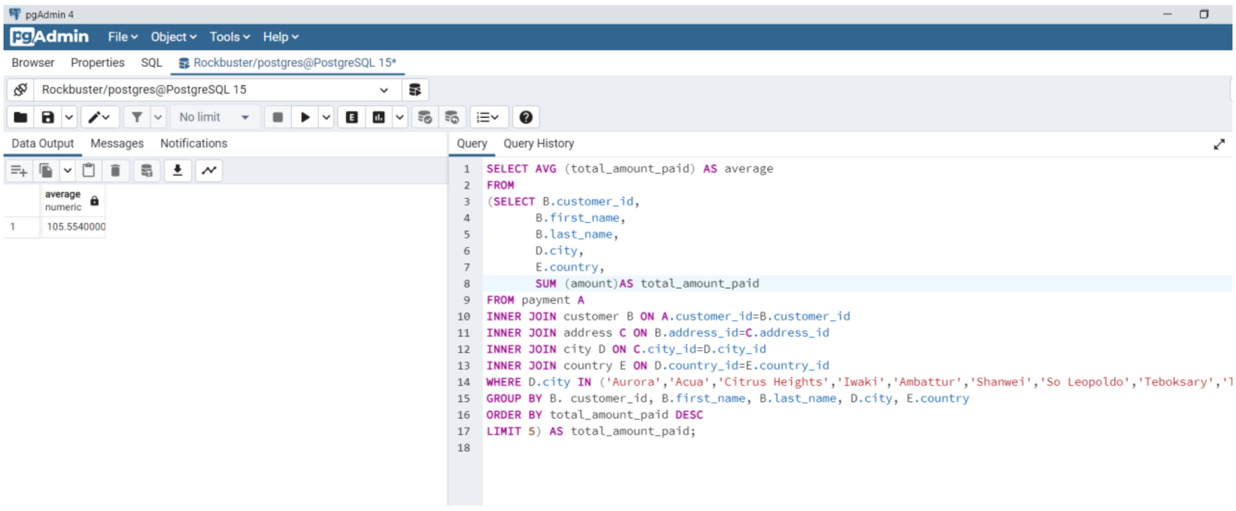
   Initial statement changed to SELECT AVG (total_amount_paid)

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”.

   Added the Alias to the outer statement

**Query and Output:**

```
SELECT AVG (total_amount_paid) AS average
FROM
(SELECT B.customer_id,
    B.first_name,
     B.last_name,
     D.city,
     E.country,
     SUM (amount)AS total_amount_paid
FROM payment A
INNER JOIN customer B ON A.customer_id=B.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
WHERE D.city IN ('Aurora','Acua','Citrus Heights','Iwaki','Ambattur','Shanwei','So
Leopoldo','Teboksary','Tianjin','Cianjur')
GROUP BY B. customer_id, B.first_name, B.last_name, D.city, E.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS total_amount_paid;
```



```
SELECT AVG (total_amount_paid) AS average
FROM
(SELECT B.customer_id,
       B.first_name,
       B.last_name,
       D.city,
       E.country,
       SUM (amount)AS total_amount_paid
FROM payment A
INNER JOIN customer B ON A.customer_id=B.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
WHERE D.city IN ('Aurora','Acua','Citrus Heights','Iwaki','Ambattur','Shanwei','So Leopoldo','Teboksary','T
GROUP BY B. customer_id, B.first_name, B.last_name, D.city, E.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS total_amount_paid;
```

```
average
numeric
1   105.5540000
```

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

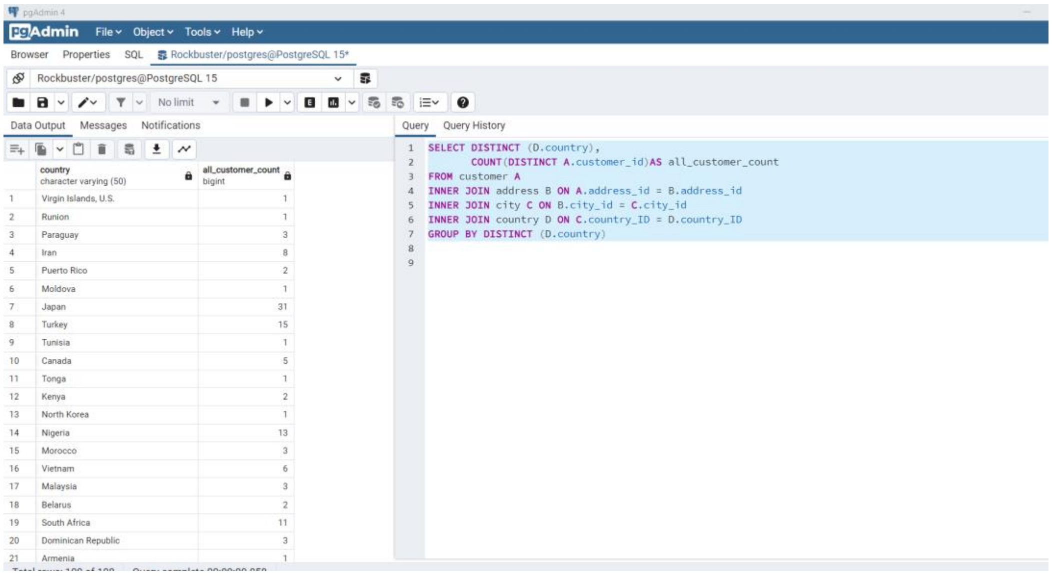
1. Copy the query from step 3 of task 3.7 into the Query Tool and add parentheses around it. This will be your inner query.

```
(SELECT B.customer_id,
       B.first_name,
         B.last_name,
         D.city,
         E.country,
         SUM (amount)AS total_amount_paid
FROM payment A
INNER JOIN customer B ON A.customer_id=B.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
WHERE D.city IN ('Aurora','Acua','Citrus Heights','Iwaki','Ambattur','Shanwei','So
Leopoldo','Teboksary','Tianjin','Cianjur')
GROUP BY B. customer_id, B.first_name, B.last_name, D.city, E.country
ORDER BY total_amount_paid DESC
LIMIT 5)
```

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.

**Outer statement:**

```
SELECT DISTINCT (D.country),
       COUNT(DISTINCT A.customer_id)AS all_customer_count
FROM customer A
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 DISTINCT (D.country)
```



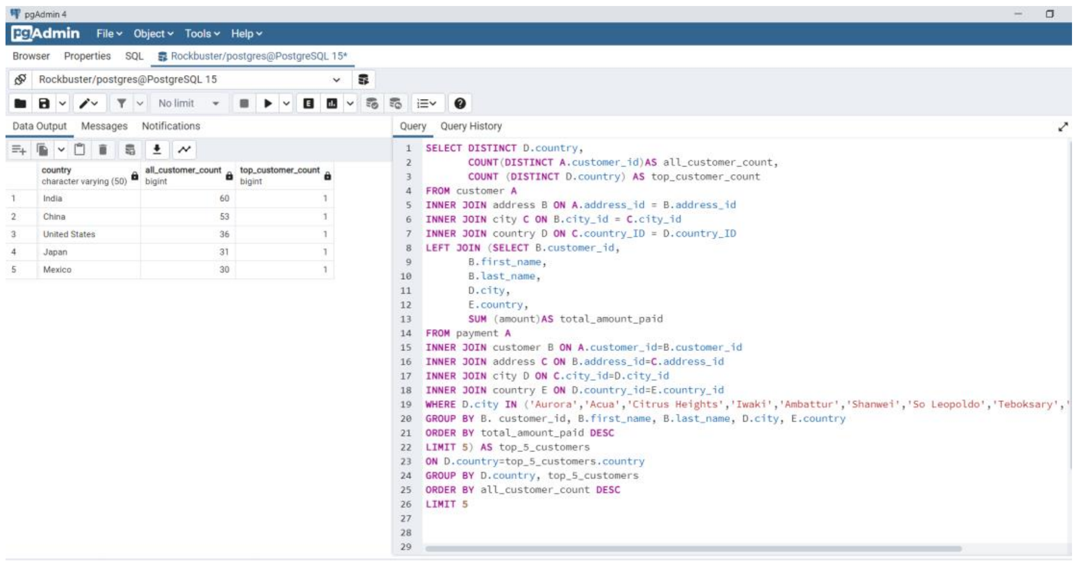
```
SELECT DISTINCT (D.country),
       COUNT(DISTINCT A.customer_id)AS all_customer_count
FROM customer A
INNER JOIN address B ON A.address_id = B.address_id
INNER JOIN city C ON B.city_id = C.city_id
INNER JOIN country D ON C.country_ID = D.country_ID
GROUP BY DISTINCT (D.country)
```

| | country<br>character varying (50) | all_customer_count<br>bigint |
|---|---|---|
| 1 | Virgin Islands, U.S. | 1 |
| 2 | Runion | 1 |
| 3 | Paraguay | 3 |
| 4 | Iran | 8 |
| 5 | Puerto Rico | 2 |
| 6 | Moldova | 1 |
| 7 | Japan | 31 |
| 8 | Turkey | 15 |
| 9 | Tunisia | 1 |
| 10 | Canada | 5 |
| 11 | Tonga | 1 |
| 12 | Kenya | 2 |
| 13 | North Korea | 1 |
| 14 | Nigeria | 13 |
| 15 | Morocco | 3 |
| 16 | Vietnam | 6 |
| 17 | Malaysia | 3 |
| 18 | Belarus | 2 |
| 19 | South Africa | 11 |
| 20 | Dominican Republic | 3 |
| 21 | Armenia | 1 |

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.

**Query and Output:**

```
SELECT DISTINCT D.country,
       COUNT(DISTINCT A.customer_id)AS all_customer_count,
              COUNT (DISTINCT D.country) AS top_customer_count
FROM customer A
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
LEFT JOIN (SELECT B.customer_id,
       B.first_name,
              B.last_name,
              D.city,
              E.country,
              SUM (amount)AS total_amount_paid
FROM payment A
INNER JOIN customer B ON A.customer_id=B.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
WHERE D.city IN ('Aurora','Acua','Citrus Heights','Iwaki','Ambattur','Shanwei','So Leopoldo','Teboksary','Tianjin','Cianjur')
GROUP BY B. customer_id, B.first_name, B.last_name, D.city, E.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS top_5_customers
ON D.country=top_5_customers.country
GROUP BY D.country, top_5_customers
ORDER BY all_customer_count DESC
LIMIT 5
```



| | 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 |

```sql
SELECT DISTINCT D.country,
       COUNT(DISTINCT A.customer_id)AS all_customer_count,
       COUNT (DISTINCT D.country) AS top_customer_count
FROM customer A
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
LEFT JOIN (SELECT B.customer_id,
       B.first_name,
       B.last_name,
       D.city,
       E.country,
       SUM (amount)AS total_amount_paid
FROM payment A
INNER JOIN customer B ON A.customer_id=B.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
WHERE D.city IN ('Aurora','Acua','Citrus Heights','Iwaki','Ambattur','Shanwei','So Leopoldo','Teboksary','
GROUP BY B. customer_id, B.first_name, B.last_name, D.city, E.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS top_5_customers
ON D.country=top_5_customers.country
GROUP BY D.country, top_5_customers
ORDER BY all_customer_count DESC
LIMIT 5
```

**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?

I think step 1 could be done without a subquery, using other functions, including HAVING with the AVERAGE function, as well as GROUP BY and ORDER BY.

But step 2, really requires retrieving data, and using diverse functions from diverse interrelated tables, and I don't think it could be done without using a subquery.

- When do you think subqueries are useful?

Subqueries are useful to analyse data from complex queries, because they enable breaking down such complexity into shorter steps, and then weaving them all logically together. With subqueries we can also take stock of queries that were used in other analyses, and integrate different queries together, for a wider analysis. They enable using isolated queries as 'lego blocks' that we put together and therefore can be very useful to perform complex analyses.