

Exercise 3.8 Performing Subqueries

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.

```
(SELECT D.customer_id,  
      D.first_name,  
      D.last_name,  
      B.city,  
      C.country,  
      D.email,  
      SUM(E.amount) AS total_paid  
FROM customer D  
JOIN address A ON A.address_id = D.address_id  
JOIN city B ON A.city_id = B.city_id  
JOIN country C ON C.country_id = B.country_id  
JOIN payment E ON E.customer_id = D.customer_id  
  
WHERE B.city IN(  
  
      SELECT B.city  
      FROM customer D  
      JOIN address A ON A.address_id = D.address_id  
      JOIN city B ON A.city_id = B.city_id  
      JOIN country C ON C.country_id = B.country_id  
  
WHERE C.country IN (  
      SELECT C.country  
      FROM customer D  
      JOIN address A ON A.address_id = D.address_id  
      JOIN city B ON A.city_id = B.city_id  
      JOIN country C ON C.country_id = B.country_id  
      GROUP BY C.country  
      ORDER BY COUNT(D.customer_id) DESC  
      LIMIT 10  
      )  
GROUP BY C.country, B.city  
ORDER BY COUNT (D.customer_id) DESC  
LIMIT 10  
)  
  
GROUP BY B.city, C.country, D.last_name, D.first_name, D.email, D.customer_id  
ORDER BY SUM(E.amount)DESC  
LIMIT 5) AS total_amount_paid
```

2. Write an outer statement to calculate the average amount paid.

```
SELECT AVG(amount)
FROM payment
```

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.

```
SELECT AVG(total_paid) AS average
FROM
```

```
(SELECT D.customer_id,
       D.first_name,
       D.last_name,
       B.city,
       C.country,
       D.email,
       SUM(E.amount) AS total_paid
FROM customer D
JOIN address A ON A.address_id = D.address_id
JOIN city B ON A.city_id = B.city_id
JOIN country C ON C.country_id = B.country_id
JOIN payment E ON E.customer_id = D.customer_id
```

```
WHERE B.city IN(
```

```
    SELECT B.city
    FROM customer D
    JOIN address A ON A.address_id = D.address_id
    JOIN city B ON A.city_id = B.city_id
    JOIN country C ON C.country_id = B.country_id
```

```
WHERE C.country IN (
```

```

SELECT C.country
FROM customer D
JOIN address A ON A.address_id = D.address_id
JOIN city B ON A.city_id = B.city_id
JOIN country C ON C.country_id = B.country_id
GROUP BY C.country
ORDER BY COUNT(D.customer_id) DESC
LIMIT 10
)
GROUP BY C.country, B.city
ORDER BY COUNT (D.customer_id) DESC
LIMIT 10
)

GROUP BY B.city, C.country, D.last_name, D.first_name, D.email, D.customer_id
ORDER BY SUM(E.amount)DESC
LIMIT 5) AS total_amount_paid

```

Query	Query History
1	SELECT AVG(total_paid) AS average
2	FROM
3	
4	
5	(SELECT D.customer_id,
6	D.first_name,
7	D.last_name,
8	B.city,
9	C.country,
10	D.email,
11	SUM(E.amount) AS total_paid
12	FROM customer D
13	JOIN address A ON A.address_id = D.address_id
14	JOIN city B ON A.city_id = B.city_id
15	JOIN country C ON C.country_id = B.country_id
16	JOIN payment E ON E.customer_id = D.customer_id
17	
18	WHERE B.city IN(
19	
20	SELECT B.city

Data Output	Messages	Explain X	Notifications						
<div> <div>≡+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>🗑️</div> <div>🔍</div> <div>⬇️</div> <div>📈</div> <div>SQL</div> </div>									
<table border="1"> <thead> <tr> <th></th> <th>average</th> <th></th> </tr> </thead> <tbody> <tr> <td></td> <td>numeric</td> <td>🔒</td> </tr> </tbody> </table>		average			numeric	🔒			
	average								
	numeric	🔒							
1	105.5540000000000000								

Step 2: Find out how many of the top 5 customers you identified in step 1 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:

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.

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

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. You’ll need to add a LEFT JOIN after your outer query, followed by the subquery in parentheses.
4. Give your subquery an alias so you can refer to it in your outer query, for example, “top_5_customers”.
5. Remember to specify which columns to join the two tables on using ON. Both ON and the column names should follow the alias.
6. Count the top 5 customers for the third column using GROUP BY and COUNT (DISTINCT). Give this column the alias “top_customer_count”.
7. Copy-paste your query and the data output into your “Answers 3.8” document.

```

SELECT C.country,
       COUNT(DISTINCT A.address_id) AS all_customer_count,
       top_five_customers.total_paid
FROM country C
      JOIN city B ON C.country_id = B.country_id
      JOIN address A ON a.city_id = B.city_id
      JOIN customer D ON D.address_id = a.address_id
LEFT JOIN

```

```

(SELECT D.customer_id,
       D.first_name,
       D.last_name,
       B.city,
       C.country,
       D.email,
       SUM(E.amount) AS total_paid
FROM customer D
      JOIN address A ON A.address_id = D.address_id
      JOIN city B ON A.city_id = B.city_id
      JOIN country C ON C.country_id = B.country_id
      JOIN payment E ON E.customer_id = D.customer_id

```

```

WHERE B.city IN(

      SELECT B.city
      FROM customer D
      JOIN address A ON A.address_id = D.address_id
      JOIN city B ON A.city_id = B.city_id
      JOIN country C ON C.country_id = B.country_id

```

```

WHERE C.country IN (
      SELECT C.country
      FROM customer D
      JOIN address A ON A.address_id = D.address_id
      JOIN city B ON A.city_id = B.city_id
      JOIN country C ON C.country_id = B.country_id
      GROUP BY C.country
      ORDER BY COUNT(D.customer_id) DESC
      LIMIT 10
    )
GROUP BY C.country, B.city

```

```
ORDER BY COUNT (D.customer_id) DESC
LIMIT 10
)
```

```
GROUP BY B.city, C.country, D.last_name, D.first_name, D.email, D.customer_id
ORDER BY SUM(E.amount)DESC
LIMIT 5) AS top_five_customers
```

```
ON C.country = top_five_customers.country
```

```
GROUP BY C.country, top_five_customers.total_paid
ORDER BY all_customer_count DESC
```

1	SELECT C.country,
2	COUNT(DISTINCT A.address_id) AS all_customer_count,
3	top_five_customers.total_paid
4	FROM country C
5	JOIN city B ON C.country_id = B.country_id
6	JOIN address A ON a.city_id = B.city_id
7	JOIN customer D ON D.address_id = a.address_id
8	LEFT JOIN
9	
10	(SELECT D.customer_id,
11	D.first_name,
12	D.last_name,

Data Output	Messages	Explain X	Notifications
	country character varying (50)	all_customer_count bigint	total_paid numeric
1	India	60	111.76
2	China	53	109.71
3	United States	36	98.76
4	Japan	31	106.77
5	Mexico	30	100.77
6	Brazil	28	[null]
7	Russian Federation	28	[null]
8	Philippines	20	[null]
9	Turkey	15	[null]
10	Indonesia	14	[null]
11	Nigeria	13	[null]

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?

I think so. Although queries may appear simpler to write, they often take longer to process. Subqueries are particularly useful when you need to filter results dynamically and use them within another query. For example, you can retrieve the top five records based on one condition and then further narrow the selection to the top three based on a different criterion.