Juan Ignacio Galvalisi

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.
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.

Rockbuster/postgres@PostgreSQL 14

Query Editor | Query History

```sql
SELECT
    ROUND(AVG(total_amount_paid),2) AS average
FROM
(SELECT
    A.customer_id,
    A.first_name,
    A.last_name,
    D.city,
    E.country,
    SUM(B.amount) AS total_amount_paid
FROM customer A
INNER JOIN payment B ON A.customer_id = B.customer_id
INNER JOIN address C ON A.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', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)',
'Kurashiki', 'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
GROUP BY
    A.customer_id,
    A.first_name,
    A.last_name,
    D.city,
    E.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS total_amount_paid;
```

Data Output | Explain | Messages | Notifications

	average numeric
1	107.35

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”.
8. Copy-paste your query and the data output into your “Answers 3.8” document.

Rockbuster/postgres@PostgreSQL 14

Query Editor | Query History

```sql
SELECT
    DISTINCT (A.country),
    COUNT(DISTINCT D.address_id) AS all_customer_count,
    COUNT(DISTINCT top_5_customers) AS top_customer_count
FROM country A
INNER JOIN city B ON A.country_id = B.country_id
INNER JOIN address C ON B.city_id = C.city_id
INNER JOIN customer D ON C.address_id = D.address_id
LEFT JOIN
(SELECT
    A.customer_id,
    A.first_name,
    A.last_name,
    D.city,
    E.country,
    SUM(B.amount) AS total_amount_paid
FROM customer A
INNER JOIN payment B ON A.customer_id = B.customer_id
INNER JOIN address C ON A.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
e.country IN ('India', 'China', 'United States', 'Japan', 'Mexico', 'Brazil',
              'Russian Federation', 'Philippines', 'Turkey', 'Indonesia')
AND D.city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)',
               'Kurashiki', 'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
GROUP BY
    A.customer_id,
    A.first_name,
    A.last_name,
    D.city,
    E.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS top_5_customers
ON A.country = top_5_customers.country
GROUP BY A.country
ORDER BY top_customer_count DESC
LIMIT 4;
```

Data Output | Explain | Messages | Notifications

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	Mexico	30	2
2	India	60	1
3	Turkey	15	1
4	United States	36	1

Step 3

Write 1 to 2 short paragraphs on the following:

- **Do you think steps 1 and 2 could be done without using subqueries?**

Step 1 could be done without subqueries through aggregate functions. Regarding Step 2, it is another situation because you need to carry out subqueries from different tables in order to aggregate and compare values from different tables.

- **When do you think subqueries are useful?**

They are useful because they save us time and queries. For example, you can carry out some aggregate functions in the same query to find specific information in different tables. Moreover, within the WHERE clause they are an excellent tool because we can filter our data and see what's there, even when it is constantly changing.

Of course, the use of subqueries has disadvantages, such as a drop in our query performance. For this reason, we must know when to use them and when not to optimize our queries and develop good performance.