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**](https://careerfoundry.com/en/course/data-immersion/exercise/joining-tables-data#task)**into the Query Tool. This will be your subquery, so give it an alias, “total\_amount\_paid,” and add parentheses around it.**

(SELECT A.customer\_id, A.first\_name, A.last\_name, C.city, D.country,  
SUM (E.amount) AS Total\_amount  
FROM customer A  
INNER JOIN address B ON B.address\_id = A.address\_id  
INNER JOIN city C ON C.city\_id = B.city\_id  
INNER JOIN country D ON D.country\_id = C.country\_id  
INNER JOIN payment E ON E.customer\_id = A.customer\_id  
WHERE C.city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')  
GROUP BY A.customer\_id, C.city, D.country  
ORDER BY Total\_amount desc  
LIMIT 5) AS total\_amount\_paid

1. **Write an outer statement to calculate the average amount paid.**

SELECT AVG(amount)   
FROM payment;

1. **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”.)**

SELECT AVG(total\_amount\_paid.total\_amount)  
FROM (SELECT A.customer\_id, A.first\_name, A.last\_name, C.city, D.country,  
SUM (E.amount) AS Total\_amount  
FROM customer A  
INNER JOIN address B ON B.address\_id = A.address\_id  
INNER JOIN city C ON C.city\_id = B.city\_id  
INNER JOIN country D ON D.country\_id = C.country\_id  
INNER JOIN payment E ON E.customer\_id = A.customer\_id  
WHERE C.city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki',  
'Ambattur', 'Shanwei', 'So Leopoldo', 'Teboksary','Tianjin', 'Cianjur')  
GROUP BY A.customer\_id, C.city, D.country  
ORDER BY Total\_amount desc  
LIMIT 5) AS total\_amount\_paid

1. **If you've done everything correctly, pgAdmin 4 will require you to add an alias after the subquery. Go ahead and call it “average”.**

**Graphical user interface, application, table

Description automatically generated**

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

**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.**

/\* A. customer B. addressC. city D. country E. payment, F.rental \*/

SELECT D.country,

COUNT (DISTINCT A.customer\_id) AS all\_customer\_count,

COUNT (DISTINCT top\_5\_customers.customer\_id) AS top\_customer\_count

FROM country D

INNER JOIN city C ON D.country\_id = C.country\_id

INNER JOIN address B ON C.city\_id = B.city\_id

INNER JOIN customer A ON B.address\_id = A.address\_id

----- 3.7

LEFT JOIN(SELECT A.customer\_id, A.first\_name, A.last\_name, C.city, D.country,

SUM (E.amount) AS total\_amount\_paid

FROM customer A

INNER JOIN address B ON B.address\_id = A.address\_id

INNER JOIN city C ON C.city\_id = B.city\_id

INNER JOIN country D ON D.country\_id = C.country\_id

INNER JOIN payment E ON E.customer\_id = A.customer\_id

WHERE C.city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki',

'Ambattur', 'Shanwei', 'So Leopoldo', 'Teboksary','Tianjin', 'Cianjur')

GROUP BY A.customer\_id, C.city, D.country

ORDER BY total\_amount\_paid desc

LIMIT 5) AS top\_5\_customers

---

ON D.country = top\_5\_customers.country

GROUP BY D.country

ORDER BY top\_customer\_count DESC

LIMIT 5

Table

Description automatically generated

**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?**

Although using subqueries can make the coding a little more complicated in terms of writing it, subqueries allow to have everything together in the same query instead of running independently each part of the process and then use the information obtained and run separately each query.   
In general, having everything ‘automated’ or working by itself without the need of having to edit the information makes the processes easier to update and reduce the mistakes.

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

It lets us to run different sub processes inside the same query, working with different operations in the same query. It is useful when you also have different tables and you want to connect and makes operations or interact with the information.

A picture containing graphical user interface

Description automatically generated