

TASK 3.8

Performing Subqueries

1:

```
SELECT      AVG(total_amount_paid)   AS average
FROM (SELECT
  P.customer_id AS Customer_ID,
  C.first_name AS Customer_First_Name,
  C.last_name AS Customer_Last_Name,
  CO.country AS Country,
  Cl.city AS City,
  SUM(P.amount) AS Total_Amount_Paid
FROM
  payment AS P
INNER JOIN
  customer AS C ON P.customer_id = C.customer_id
INNER JOIN
  address AS A ON C.address_id = A.address_id
INNER JOIN
  city AS Cl ON A.city_id = Cl.city_id
INNER JOIN
  country AS CO ON Cl.country_id = CO.country_id
WHERE
  Cl.city IN (
    SELECT
      Cl.city
    FROM
      customer AS C
    INNER JOIN
      address AS A ON C.address_id = A.address_id
    INNER JOIN
      city AS Cl ON A.city_id = Cl.city_id
    INNER JOIN
      country AS CO ON Cl.country_id = CO.country_id
    GROUP BY
      Cl.city
    ORDER BY
      COUNT(DISTINCT C.customer_id) DESC
    LIMIT 10
  )
GROUP BY
  P.customer_id,
  C.first_name,
  C.last_name,
```

```

CO.country,
CI.city
ORDER BY
    Total_Amount_Paid DESC
LIMIT 5)AS total_amount_paid;

```

stg... X Rockbuster/postg... X Rockbuster/postg... X Rockbuster/postg... X Rockbuster/

Rockbuster/postgres@PostgreSQL 16

Query Query History

```

1 SELECT AVG(total_amount_paid) AS average
2 FROM (SELECT
3     P.customer_id AS Customer_ID,
4     C.first_name AS Customer_First_Name,
5     C.last_name AS Customer_Last_Name,
6     CO.country AS Country,
7     CI.city AS City,
8     SUM(P.amount) AS Total_Amount_Paid
9 FROM
10     payment AS P
11 INNER JOIN
12     customer AS C ON P.customer_id = C.customer_id
13 INNER JOIN
14     address AS A ON C.address_id = A.address_id
15 INNER JOIN
16     city AS CI ON A.city_id = CI.city_id
17 INNER JOIN
18     country AS CO ON CI.country_id = CO.country_id
19 WHERE
20     CI.city IN (
21         SELECT
22             CI.city
23         FROM
24             customer AS C
25         INNER JOIN

```

Data Output Messages Notifications

	average	
	numeric	
1	111.936000000000000000	

Total rows: 1 of 1 Query complete 00:00:06.594

2:

```
SELECT CO.country, COUNT(DISTINCT C.customer_id) AS all_customer_count,
COUNT(DISTINCT top_5_customers.customer_id) AS top_customer_count FROM customer
AS C
INNER JOIN address AS A ON A.address_id = C.address_id
INNER JOIN city AS CI ON CI.city_id = A.city_id
INNER JOIN country CO ON CO.country_id = CI.country_id
LEFT JOIN
(SELECT C.customer_id, C.first_name, C.last_name, CO.country, CI.city, SUM(P.amount)
AS total_amount_payment FROM payment AS P
INNER JOIN customer AS C ON C.customer_id = P.customer_id
INNER JOIN address AS A ON A.address_id = C.address_id
INNER JOIN city AS CI ON CI.city_id = A.city_id
INNER JOIN country CO ON CO.country_id = CI.country_id
WHERE CI.city IN (
SELECT CI.city FROM customer AS C
INNER JOIN address AS A ON A.address_id = C.address_id
INNER JOIN city AS CI ON CI.city_id = A.city_id
INNER JOIN country CO ON CO.country_id = CI.country_id
WHERE CO.country IN (
SELECT CO.country FROM customer AS C
INNER JOIN address AS A ON A.address_id = C.address_id
INNER JOIN city AS CI ON CI.city_id = A.city_id
INNER JOIN country CO ON CO.country_id = CI.country_id
GROUP BY CO.country
ORDER BY COUNT(C.customer_id) DESC
LIMIT 10)
GROUP BY CO.country, CI.city
ORDER BY Count(C.customer_id) DESC
LIMIT 10)
GROUP BY C.customer_id, CO.country, CI.city
ORDER BY total_amount_payment DESC
LIMIT 5) AS top_5_customers ON top_5_customers.country = CO.country
GROUP BY CO.country
ORDER BY top_customer_count DESC;
```

Rockbuster/postgres@PostgreSQL 16

Query Query History

```

1 SELECT CO.country, COUNT(DISTINCT C.customer_id) AS all_customer_count,
2 COUNT(DISTINCT top_5_customers.customer_id) AS top_customer_count FROM customer
3 AS C
4 INNER JOIN address AS A ON A.address_id = C.address_id
5 INNER JOIN city AS CI ON CI.city_id = A.city_id
6 INNER JOIN country CO ON CO.country_id = CI.country_id
7 LEFT JOIN
8 (SELECT C.customer_id, C.first_name, C.last_name, CO.country, CI.city, SUM(P.amount)
9 AS total_amount_payment FROM payment AS P
10 INNER JOIN customer AS C ON C.customer_id = P.customer_id
11 INNER JOIN address AS A ON A.address_id = C.address_id
12 INNER JOIN city AS CI ON CI.city_id = A.city_id
13 INNER JOIN country CO ON CO.country_id = CI.country_id
14 WHERE CI.city IN (
15 SELECT CI.city FROM customer AS C
16 INNER JOIN address AS A ON A.address_id = C.address_id
17 INNER JOIN city AS CI ON CI.city_id = A.city_id
18 INNER JOIN country CO ON CO.country_id = CI.country_id
19 WHERE CO.country IN (
20 SELECT CO.country FROM customer AS C
21 INNER JOIN address AS A ON A.address_id = C.address_id

```

Data Output Messages Notifications

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
28	Estonia	1	0
29	Ethiopia	1	0
30	Faroe Islands	1	0
31	Finland	1	0

Total rows: 108 of 108 Query complete 00:00:00.664

We start by selecting the country column from the country table and counting the number of distinct customer IDs (customer_id) for each country.

Then we join the customer, address, city, and country tables to link customer information with their respective cities and countries.

Using GROUP BY CO.country, we group the results by country.

COUNT(DISTINCT C.customer_id) counts the number of distinct customer IDs for each country.

Finally, we give the second column the alias all_customer_count for readability.

3:

```
SELECT outer_query.country,
       outer_query.all_customer_count,
       inner_query.average
FROM (
  SELECT CO.country,
         COUNT(DISTINCT C.customer_id) AS all_customer_count
  FROM customer AS C
  INNER JOIN address AS A ON C.address_id = A.address_id
  INNER JOIN city AS CI ON A.city_id = CI.city_id
  INNER JOIN country AS CO ON CI.country_id = CO.country_id
  GROUP BY CO.country
) AS outer_query
LEFT JOIN (
  SELECT AVG(P.amount) AS average,
         CI.country_id AS country_id
  FROM payment AS P
  INNER JOIN customer AS C ON P.customer_id = C.customer_id
  INNER JOIN address AS A ON C.address_id = A.address_id
  INNER JOIN city AS CI ON A.city_id = CI.city_id
  INNER JOIN country AS CO ON CI.country_id = CO.country_id
  WHERE CI.city IN (
    SELECT CI.city
    FROM customer AS C
    INNER JOIN address AS A ON C.address_id = A.address_id
    INNER JOIN city AS CI ON A.city_id = CI.city_id
    INNER JOIN country AS CO ON CI.country_id = CO.country_id
    GROUP BY CI.city
    ORDER BY COUNT(DISTINCT C.customer_id) DESC
    LIMIT 10
  )
  GROUP BY CI.country_id
  ORDER BY average DESC
  LIMIT 5
) AS inner_query ON outer_query.country = inner_query.country_id::text;
```

Rockbuster/postgres@PostgreSQL 16

Query Query History

```

1 SELECT outer_query.country,
2       outer_query.all_customer_count,
3       inner_query.average
4 FROM (
5     SELECT C0.country,
6           COUNT(DISTINCT C.customer_id) AS all_customer_count
7     FROM customer AS C
8    INNER JOIN address AS A ON C.address_id = A.address_id
9    INNER JOIN city AS CI ON A.city_id = CI.city_id
10   INNER JOIN country AS CO ON CI.country_id = CO.country_id
11   GROUP BY C0.country
12 ) AS outer_query
13 LEFT JOIN (
14     SELECT AVG(P.amount) AS average,
15           CI.country_id AS country_id
16     FROM payment AS P
17    INNER JOIN customer AS C ON P.customer_id = C.customer_id
18    INNER JOIN address AS A ON C.address_id = A.address_id
19    INNER JOIN city AS CI ON A.city_id = CI.city_id
20    INNER JOIN country AS CO ON CI.country_id = CO.country_id
21   WHERE CI.city IN (

```

Data Output Messages Notifications

	country character varying (50)	all_customer_count bigint	average numeric
1	Afghanistan	1	[null]
2	Algeria	3	[null]
3	American Samoa	1	[null]
4	Angola	2	[null]

Total rows: 108 of 108 Query complete 00:00:01.826

Subqueries are indispensable in this context. Firstly, they enable us to isolate subsets of data for analysis within a larger dataset. In the case of step 1, subqueries allow us to identify and calculate the average amount paid by the top 5 customers efficiently.

Subqueries prove to be invaluable when dealing with complex queries that require the manipulation of data subsets or when there's a need to derive insights based on the results of other queries. They enhance the modularity and readability of SQL statements by breaking down intricate problems into more manageable components. Moreover, subqueries offer versatility, as they can be employed in various parts of a SQL query,

including the SELECT, FROM, WHERE, and HAVING clauses, allowing for greater flexibility in query formulation.