

### Ex 3.8

Prepared By: Nikita Kasrekar

1. Step 1: Find the average amount paid by the top 5 customers.

```
SELECT AVG(total_amount_paid) AS average_amount_paid
FROM
(SELECT
    SUM (p.amount) AS total_amount_paid
    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
    INNER JOIN payment P ON A.customer_id = P.customer_id
WHERE C.city IN (
    SELECT
        C.city
        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
        WHERE D.country IN (
            SELECT
                D.country
                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 D.country
                ORDER BY COUNT (A.customer_id) DESC
                LIMIT 10
```

)

**GROUP BY D.country, C.city**

**ORDER BY COUNT(A.customer\_id) DESC**

## LIMIT 10

)

**GROUP BY A.customer\_id, A.first\_name, A.last\_name, D.country, C.city**

**ORDER BY total\_amount\_paid DESC**

## LIMIT 5

) AS average\_amount\_paid;

```
Rockbuster/postgres@PostgreSQL-13
```

Query Query History

```
1 SELECT AVG(total_amount_paid) AS average_amount_paid
2 FROM
3   (SELECT
4     SUM(p.amount) AS total_amount_paid
5     FROM customer A
6    INNER JOIN address B ON A.address_id = B.address_id
7    INNER JOIN city C ON B.city_id = C.city_id
8    INNER JOIN country D ON C.country_id = D.country_id
9    INNER JOIN payment P ON A.customer_id = P.customer_id
10   WHERE C.city IN (
11     SELECT
12       C.city
13     FROM customer A
14    INNER JOIN address B ON A.address_id = B.address_id
15    INNER JOIN city C ON B.city_id = C.city_id
16    INNER JOIN country D ON C.country_id = D.country_id
17   WHERE D.country IN (
18     SELECT
19       D.country
20     FROM customer A
21    INNER JOIN address B ON A.address_id = B.address_id
22    INNER JOIN city C ON B.city_id = C.city_id
23 ) ) )
```

Data Output Messages Notifications

average\_amount\_paid  
numeric

1 105.884000000000000

**Step 2: Find out how many of the top 5 customers you identified in step 1 are based within each country.**

**SELECT D.country,**

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

## COUNT(DISTINCT CASE

**WHEN (**

**SELECT SUM(p.amount)**

```
FROM payment p
WHERE p.customer_id = A.customer_id
)>
SELECT AVG (Total_sum)
FROM (
SELECT SUM (p1.amount) AS Total_sum
FROM payment p1
GROUP BY p1.customer_id
)
AS customer_totals
)
THEN A.customer_id
ELSE NULL
END) 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
GROUP BY D.country
ORDER BY top_customer_count DESC
LIMIT 10;
```

```

SELECT b.country,
       COUNT(DISTINCT a.customer_id) AS all_customer_count,
       COUNT(DISTINCT CASE
                           WHEN (
                               SELECT SUM(p.amount)
                               FROM payment p
                               WHERE p.customer_id = a.customer_id
                           ) >=
                           SELECT AVG (total_sum)
                           FROM (
                               SELECT SUM(pi.amount) AS Total_sum
                               FROM payment pi
                               GROUP BY pi.customer_id
                           )
                           AS customer_totals
                           )
                           THEN a.customer_id
                           ELSE NULL
                           END) AS top_customer_count
        FROM customer a
        INNER JOIN address b ON a.address_id = b.address_id
    
```

The screenshot shows a MySQL Workbench interface. The left sidebar contains a tree view of database objects: FTS Text, Foreign, All Functions, All Methods, Open, Procedure, Sequence, Tables, ads, add, cities, cust, film, film, film, lang, pay, rents, staff, stores, Trigger, Types, Views, Subscription, address, and Group Role. The main area has tabs for Query, Query History, and Scratch Pad. The Query tab displays the SQL code above. Below the code is a Data Output table with three rows:

country	all_customer_count	top_customer_count
India	88	26
China	63	25
United States	36	19
Total	187	70

At the bottom of the interface, it says "Total rows: 10 Query complete 00:00:00.169 CRLF Ln 26, Col 13".

### Step 3: Write 1 to 2 short paragraphs on the following:

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

**Yes, steps 1 and 2 would be very hard to do without subqueries because we first needed to isolate the top 5 customers before calculating averages or joining that list with all customers by country. Without subqueries, the query would become too long and complicated.**

- When do you think subqueries are useful?

**Subqueries are useful when we need to work with a filtered or aggregated result before using it in another calculation or join. They make complex problems easier by breaking them into smaller, clearer steps.**