

## Data Immersion 3.8

1.

```
SELECT      AVG(total_amount_paid)      AS      average
FROM
(SELECT      B.customer_id,
              B.first_name,
              B.last_name,
              E.country,
              D.city,
              SUM(A.amount)      AS      total_amount_paid
FROM payment      A
              JOIN customer      B      ON
              A.customer_id =      B.customer_id
JOIN addressC      ON      B.address_id =      C.address_id
JOIN city      D      ON      C.city_id      =      D.city_id
JOIN country      E      ON      D.country_id =      E.country_id
WHERE      D.city IN      (SELECT      D.city
FROM      customer      B
              JOIN addressC      ON      B.address_id =      C.address_id
              JOIN city      D      ON      C.city_id      =      D.city_id
              JOIN country      E      ON      D.country_id =      E.country_id
WHERE      E.country IN      (SELECT      E.country
FROM      customer      B
              JOIN addressC      ON
B.address_id =      C.address_id
              JOIN city
D      ON      C.city_id      =      D.city_id
```

```

country      E      ON      D.country_id =      E.country_id      JOIN

GROUP      BY      E.country

COUNT(B.customer_id)      DESC      ORDER      BY

LIMIT 10)

GROUP      BY      E.country,
D.city
ORDER      BY      COUNT(B.customer_id)      DESC

LIMIT 10)
GROUP      BY      B.customer_id,

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

```

Query
Query History
Scr

```

1 SELECT AVG(total_amount_paid) AS average
2 FROM
3 (SELECT B.customer_id,
4      B.first_name,
5      B.last_name,
6      E.country,
7      D.city,
8      SUM(A.amount) AS total_amount_paid
9 FROM   payment A
10      JOIN customer B ON A.customer_id = B.customer_id
11 JOIN   address C ON B.address_id = C.address_id
12 JOIN   city D ON C.city_id = D.city_id
13 JOIN   country E ON D.country_id = E.country_id
14 WHERE  D.city IN (SELECT D.city
15                  FROM customer B
16                  JOIN address C ON B.address_id = C.address_id
17                  JOIN city D ON C.city_id = D.city_id
18                  JOIN country E ON D.country_id = E.country_id
19                  WHERE E.country IN (SELECT E.country
20                                     FROM customer B
21                                     JOIN address C ON B.address_id = C.address_id
22                                     JOIN city D ON C.city_id = D.city_id
23                                     JOIN country E ON D.country_id = E.country_id
24                                     ORDER BY COUNT(B.customer_id) DESC
25                                     GROUP BY E.country
26                                     LIMIT 10)
27                  GROUP BY E.country,
28                  D.city
29 ORDER BY COUNT(B.customer_id) DESC
30 GROUP BY B.customer_id,
31          B.first_name,
32          B.last_name,
33          E.country,
34          D.city
35 ORDER BY SUM(A.amount) DESC
36 LIMIT 5) AS total_amount_paid
37
38

```

Data Output
Messages
Notifications

Total rows: 1 of 1
Query complete 00:00:00.247

**2.**

```
SELECT
    E.country,
    COUNT(DISTINCT B.customer_id) AS all_customer_count,
    COUNT(DISTINCT top_5_customers.customer_id) AS top_customer_count
FROM
    customer B
JOIN
    address C ON B.address_id = C.address_id
JOIN
    city D ON C.city_id = D.city_id
JOIN
    country E ON D.country_id = E.country_id
LEFT JOIN (
    SELECT
        B.customer_id,
        B.first_name,
        B.last_name,
        E.country,
        D.city,
        SUM(A.amount) AS total_amount_paid
    FROM
        payment A
    JOIN
        customer B ON A.customer_id = B.customer_id
    JOIN
        address C ON B.address_id = C.address_id
    JOIN
        city D ON C.city_id = D.city_id
    JOIN
        country E ON D.country_id = E.country_id
```

```

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

```

```

ORDER BY
    COUNT(B.customer_id) DESC
LIMIT 10
)
GROUP BY
    B.customer_id, B.first_name, B.last_name, E.country, D.city
ORDER BY
    SUM(A.amount) DESC
LIMIT 5
) AS top_5_customers ON B.customer_id = top_5_customers.customer_id
GROUP BY
    E.country
ORDER BY
    all_customer_count DESC
LIMIT 10;

```

Query		Query History	
1	SELECT		
2	E.country,		
3	COUNT(DISTINCT B.customer_id) AS all_customer_count,		
4	COUNT(DISTINCT top_5_customers.customer_id) AS top_customer_count		
5	FROM		
6	customer B		
7	JOIN		
8	address C ON B.address_id = C.address_id		
9	JOIN		
10	city D ON C.city_id = D.city_id		
11	JOIN		
12	country E ON D.country_id = E.country_id		
13	LEFT JOIN (		
14	SELECT		
15	B.customer_id,		
16	B.first_name,		
17	B.last_name,		
18	E.country,		
19	D.city,		
20	SUM(A.amount) AS total_amount_paid		
21	FROM		
22	payment A		
23	JOIN		
24	customer B ON A.customer_id = B.customer_id		
Data Output		Messages	Notifications
	country	all_customer_count	top_customer_count
	character varying (50)	bigint	bigint
1	India	60	1
2	China	53	1
3	United States	36	1
4	Japan	31	1
5	Mexico	30	1
6	Brazil	28	0
7	Russian Federation	28	0
8	Philippines	20	0
9	Turkey	15	0
10	Indonesia	14	0
Total rows: 10 of 10		Query complete 00:00:00.592	

### 3.

For tasks like identifying top customers by country and calculating average payments, views can be an effective alternative to subqueries. Crafting a long subquery with multiple joins and conditions can be cumbersome and error-prone, requiring meticulous key matching and extensive code lines. In contrast, views simplify queries into shorter, readable segments that can be reused and easily adjusted. I found this query far too long and complicated, and as I have only been using PostgreSQL for 3 weeks it is hard to say much more.

Subqueries are powerful tools in SQL that provide the flexibility to perform complex calculations in a single query. They are essential when the latest, real-time data is needed, as they ensure that every part of the query is executed with the most current data. While subqueries can be less user-friendly and might slow down query performance due to their complexity, they are invaluable for creating concise, logically coherent queries that operate directly on the latest dataset without the need for intermediate steps.