

Databases & SQL for Analysts

3.8: Performing Subqueries

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

Query

Query History

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

Data Output

Messages

Notifications

	average_amount_paid
	numeric
1	105.5540000000000000

SELECT AVG(total_amount_paid) AS average_amount_paid
FROM (
SELECT
SUM(pay.amount) AS total_amount_paid
FROM customer A
INNER JOIN payment pay ON A.customer_id = pay.customer_id
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 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;

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

Query Query History

```
1  SELECT
2      D.country,
3      COUNT(DISTINCT A.customer_id) AS all_customer_count,
4      COUNT(DISTINCT T.customer_id) AS top_customer_count
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  LEFT JOIN (
10
11      SELECT Totals.customer_id, Totals.country
12  FROM (
13      SELECT
14          A2.customer_id,
15          D2.country,
16          SUM(pay.amount) AS total_sum
```

Data Output Messages Notifications

	country character varying (50) 🔒	all_customer_count bigint 🔒	top_customer_count bigint 🔒
1	India	60	26
2	China	53	25
3	United States	36	16
4	Japan	31	14
5	Russian Federation	28	13
6	Brazil	28	12
7	Mexico	30	11
8	Philippines	20	11
9	Taiwan	10	7
10	Turkey	15	7

Total rows: 10 Query complete 00:00:00.081

Query

Query History

```

1  SELECT
2  D.country,
3  COUNT(DISTINCT A.customer_id) AS all_customer_count,
4  COUNT(DISTINCT T.customer_id) AS top_customer_count
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  LEFT JOIN (
10
11     SELECT Totals.customer_id, Totals.country
12     FROM (
13         SELECT
14             A2.customer_id,
15             D2.country,
16             SUM(pay.amount) AS total_sum
17         FROM customer A2
18         INNER JOIN payment pay ON A2.customer_id = pay.customer_id
19         INNER JOIN address B2 ON A2.address_id = B2.address_id
20         INNER JOIN city C2 ON B2.city_id = C2.city_id
21         INNER JOIN country D2 ON C2.country_id = D2.country_id
22         GROUP BY A2.customer_id, D2.country
23     ) AS Totals
24     WHERE Totals.total_sum > (
25         SELECT AVG(total_sum)
26         FROM (
27             SELECT SUM(pay.amount) AS total_sum
28             FROM customer A3
29             INNER JOIN payment pay ON A3.customer_id = pay.customer_id
30             GROUP BY A3.customer_id
31         ) AS Customer_totals
32     )
33 ) AS T
34 ON A.customer_id = T.customer_id
35 GROUP BY D.country
36 ORDER BY top_customer_count DESC
37 LIMIT 10;
38
39

```

Total rows: 10

Query complete 00:00:00.076

SELECT
D.country,
COUNT(DISTINCT A.customer_id) AS all_customer_count,
COUNT(DISTINCT T.customer_id) 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
LEFT JOIN (
SELECT Totals.customer_id, Totals.country
FROM (
SELECT
A2.customer_id,
D2.country,
SUM(pay.amount) AS total_sum
FROM customer A2
INNER JOIN payment pay ON A2.customer_id = pay.customer_id
INNER JOIN address B2 ON A2.address_id = B2.address_id
INNER JOIN city C2 ON B2.city_id = C2.city_id
INNER JOIN country D2 ON C2.country_id = D2.country_id
GROUP BY A2.customer_id, D2.country

) AS Totals
WHERE Totals.total_sum > (
SELECT AVG(total_sum)
FROM (
SELECT SUM(pay.amount) AS total_sum
FROM customer A3
INNER JOIN payment pay ON A3.customer_id = pay.customer_id
GROUP BY A3.customer_id
) AS Customer_totals
)
) AS T
ON A.customer_id = T.customer_id
GROUP BY D.country
ORDER BY top_customer_count DESC
LIMIT 10;

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

I think we can't run steps 1 and 2 without using subqueries because they are essential to isolate intermediate results and make the logic work correctly in SQL.

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

Subqueries are useful for isolating an intermediate calculation, filtering, and simplifying complex queries.