COMMON TABLE EXPRESSION

Ann Mariya Francis

Exercise 3.9

5 Feb 2025

1.1)

WITH top_countries AS (

  SELECT D.country

  FROM customer A

  JOIN address B ON A.address_id = B.address_id

  JOIN city C ON B.city_id = C.city_id

  JOIN country D ON C.country_id = D.country_id

  GROUP BY D.country

  ORDER BY COUNT(A.customer_id) DESC

  LIMIT 10

),

top_cities AS (

  SELECT D.country, C.city

  FROM customer A

  JOIN address B ON A.address_id = B.address_id

  JOIN city C ON B.city_id = C.city_id

  JOIN country D ON C.country_id = D.country_id

  WHERE D.country IN (SELECT country FROM top_countries)

  GROUP BY D.country, C.city

  ORDER BY COUNT(A.customer_id) DESC

  LIMIT 10

),

total_amount_per_customer AS (

  SELECT

    B.customer_id,

    B.first_name,

```sql
        B.last_name,

        E.country,

        D.city,

        SUM(A.amount) AS total_amount_paid

    FROM payment A

    INNER JOIN customer B ON A.customer_id = B.customer_id

    INNER JOIN address C ON B.address_id = C.address_id

    INNER JOIN city D ON C.city_id = D.city_id

    INNER JOIN country E ON D.country_id = E.country_id

    WHERE (E.country, D.city) IN (SELECT country, city FROM top_cities)

    GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country

),

top_5_customers AS (

    SELECT customer_id, total_amount_paid

    FROM total_amount_per_customer

    ORDER BY total_amount_paid DESC

    LIMIT 5

)

SELECT AVG(total_amount_paid) AS average

FROM top_5_customers;
```

```
 1 v  WITH top_countries AS (
 2
 3        SELECT D.country
 4        FROM customer A
 5        JOIN address B ON A.address_id = B.address_id
 6        JOIN city C ON B.city_id = C.city_id
 7        JOIN country D ON C.country_id = D.country_id
 8        GROUP BY D.country
 9        ORDER BY COUNT(A.customer_id) DESC
10        LIMIT 10
11   ),
12   top_cities AS (
13
14        SELECT D.country, C.city
```

**Data Output**   Messages   Notifications

Showing rows: 1 to 1   Page No: 1

| | average<br>numeric |
|---|---|
| 1 | 105.5540000000000000 |

we first identify the **top 10 countries** with the most customers, then filter the **top 10 cities** in those countries. Next, we calculate the **total amount paid** by each customer in those locations. After that, we select the **top 5 customers** based on their total payments and finally compute the **average amount paid** by them.

1.2)

WITH top_countries AS (

   SELECT D.country

   FROM customer A

   JOIN address B ON A.address_id = B.address_id

   JOIN city C ON B.city_id = C.city_id

   JOIN country D ON C.country_id = D.country_id

   GROUP BY D.country

   ORDER BY COUNT(A.customer_id) DESC

   LIMIT 10

),

top_cities AS (

   SELECT D.country, C.city

   FROM customer A

```sql
        JOIN address B ON A.address_id = B.address_id

        JOIN city C ON B.city_id = C.city_id

        JOIN country D ON C.country_id = D.country_id

        WHERE D.country IN (SELECT country FROM top_countries)

        GROUP BY D.country, C.city

        ORDER BY COUNT(A.customer_id) DESC

        LIMIT 10

),

total_amount_per_customer AS (

    SELECT

        B.customer_id,

        B.first_name,

        B.last_name,

        E.country,

        D.city,

        SUM(A.amount) AS total_amount_paid

    FROM payment A

    INNER JOIN customer B ON A.customer_id = B.customer_id

    INNER JOIN address C ON B.address_id = C.address_id

    INNER JOIN city D ON C.city_id = D.city_id

    INNER JOIN country E ON D.country_id = E.country_id

    WHERE (E.country, D.city) IN (SELECT country, city FROM top_cities)

    GROUP BY B.customer_id, B.first_name, B.last_name, D.city, E.country

),

top_5_customers AS (

    SELECT customer_id, country

    FROM total_amount_per_customer

    ORDER BY total_amount_paid DESC

    LIMIT 5

),

country_customer_count AS (
```

```sql
    SELECT

        E.country,

        COUNT(DISTINCT B.customer_id) AS all_customer_count

    FROM customer B

    INNER JOIN address C ON B.address_id = C.address_id

    INNER JOIN city D ON C.city_id = D.city_id

    INNER JOIN country E ON D.country_id = E.country_id

    GROUP BY E.country

)

SELECT

    ccc.country,

    ccc.all_customer_count,

    COUNT(DISTINCT tc.customer_id) AS top_customer_count

FROM country_customer_count ccc

LEFT JOIN top_5_customers tc ON ccc.country = tc.country

GROUP BY ccc.country, ccc.all_customer_count

ORDER BY top_customer_count DESC;
```

```
1 ∨  WITH top_countries AS (
2          SELECT D.country
3          FROM customer A
4          JOIN address B ON A.address_id = B.address_id
5          JOIN city C ON B.city_id = C.city_id
6          JOIN country D ON C.country_id = D.country_id
7          GROUP BY D.country
8          ORDER BY COUNT(A.customer_id) DESC
9          LIMIT 10
10     ),
11     top_cities AS (
12         SELECT D.country, C.city
13         FROM customer A
14         JOIN address B ON A.address_id = B.address_id
```

Data Output   Messages   Notifications

Showing rows: 1 to 108   Page No: 1

| | country character varying (50) | all_customer_count bigint | top_customer_count bigint |
|---|---|---|---|
| 1 | Mexico | 30 | 1 |
| 2 | India | 60 | 1 |
| 3 | China | 53 | 1 |
| 4 | United States | 36 | 1 |
| 5 | Japan | 31 | 1 |
| 6 | Argentina | 13 | 0 |

we first count the **total number of customers per country**. Then, using the previous logic, we determine the **top 5 customers** and identify which country they belong to. Finally, we join this data to see how many of those top 5 customers exist in each country while also displaying the total customer count per country.

2)

CTEs generally perform better than subqueries because they help avoid repeated calculations and make queries easier to read. CTEs improve query structure and allow PostgreSQL to optimize execution more efficiently.

| Query #1 | Subquery | CTE |
|---|---|---|
| Cost | 127.72 | 127.74 |
| Execution Time (Milliseconds) | 00:00:00.105 | 00:00:00.102 |

| Query #2 | Subquery | CTE |
|---|---|---|
| Cost | 231.8 | 235.07 |
| Execution Time | 00:00:00.146 | 00:00:00.113 |

3) One challenge I faced when replacing subqueries with CTEs was making sure the query logic stayed clear and correct. Subqueries are often more compact, while CTEs require breaking the query into multiple parts, which can be tricky to organize properly.