Databases & SQL for Analysts

3.9: Common Table Expressions

STEP 1

```
WITH top_countries AS (
    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
),
top_cities AS (
    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 country FROM top_countries)
    GROUP BY D.country, C.city
    ORDER BY COUNT(A.customer_id) DESC
    LIMIT 10
),
customer_totals AS (
    SELECT
        A.customer_id,
        A.first_name,
        A.last_name,
        D.country,
        C.city,
        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 city FROM top_cities)
    GROUP BY A.customer_id, A.first_name, A.last_name, D.country, C.city
    ORDER BY total_amount_paid DESC
    LIMIT 5
)
SELECT AVG(total_amount_paid) AS average_amount_paid
FROM customer_totals;
```

Query | Query History

```
1   WITH top_countries AS (
2       SELECT D.country
3       FROM customer A
4       INNER JOIN address B ON A.address_id = B.address_id
5       INNER JOIN city C ON B.city_id = C.city_id
6       INNER 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 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 (SELECT country FROM top_countries)
18      GROUP BY D.country, C.city
19      ORDER BY COUNT(A.customer_id) DESC
20      LIMIT 10
21  ),
```

Data Output | Messages | Notifications

SQL

	average_amount_paid numeric
1	105.5540000000000000

Query | Query History

```sql
WITH top_countries AS (
    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
),
top_cities AS (
    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 country FROM top_countries)
    GROUP BY D.country, C.city
    ORDER BY COUNT(A.customer_id) DESC
    LIMIT 10
),
customer_totals AS (
    SELECT
        A.customer_id,
        A.first_name,
        A.last_name,
        D.country,
        C.city,
        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 city FROM top_cities)
    GROUP BY A.customer_id, A.first_name, A.last_name, D.country, C.city
    ORDER BY total_amount_paid DESC
    LIMIT 5
)
SELECT AVG(total_amount_paid) AS average_amount_paid
FROM customer_totals;
```

Total rows: 1 | Query complete 00:00:00.062

Write 2 to 3 sentences explaining how you approached this step, for example, what you did first, second, and so on.

First, I replaced the nested subqueries with separate CTEs to make each step clearer. Then I used top_countries, top_cities, and customer_totals as CTEs to structure the logic step by step, and finally calculated the average from the last CTE for a cleaner, more readable query.

```
WITH customer_totals AS (
    SELECT
        A.customer_id,
        D.country,
        SUM(pay.amount) AS total_sum
    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
    GROUP BY A.customer_id, D.country
),
avg_total AS (
    SELECT AVG(total_sum) AS avg_sum
    FROM customer_totals
),
top_customers AS (
    SELECT ct.customer_id, ct.country
    FROM customer_totals ct
    CROSS JOIN avg_total a
    WHERE ct.total_sum > a.avg_sum
)
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 top_customers T ON A.customer_id = T.customer_id
GROUP BY D.country
ORDER BY top_customer_count DESC
LIMIT 10;
```

Query | Query History

```sql
WITH customer_totals AS (
    SELECT
        A.customer_id,
        D.country,
        SUM(pay.amount) AS total_sum
    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
    GROUP BY A.customer_id, D.country
),
avg_total AS (
    SELECT AVG(total_sum) AS avg_sum
    FROM customer_totals
),
top_customers AS (
    SELECT ct.customer_id, ct.country
    FROM customer_totals ct
    CROSS JOIN avg_total a
    WHERE ct.total_sum > a.avg_sum
)
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 top_customers T ON A.customer_id = T.customer_id
GROUP BY D.country
ORDER BY top_customer_count DESC
LIMIT 10;
```

Data Output | Messages | Notifications

SQL

country	all_customer_count	top_customer_count

Query Query History

```sql
WITH customer_totals AS (
    SELECT
        A.customer_id,
        D.country,
        SUM(pay.amount) AS total_sum
    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
    GROUP BY A.customer_id, D.country
),
avg_total AS (
    SELECT AVG(total_sum) AS avg_sum
    FROM customer_totals
),
top_customers AS (
    SELECT ct.customer_id, ct.country
    FROM customer_totals ct
    CROSS JOIN avg_total a
    WHERE ct.total_sum > a.avg_sum
)
SELECT
    D.country,
    COUNT(DISTINCT A.customer_id) AS all_customer_count,
```

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 00:00:00.104

Write 2 to 3 sentences explaining how you approached this step, for example, what you did first, second, and so on.

I moved the subquery calculating each customer's total payments into a CTE called customer_totals, then added avg_total and top_customers to filter those above average. Finally, I used these CTEs in the main query to compare all customers with top customers by country.

STEP 2:

1/Which approach do you think will perform better and why?

I think the CTE will Perform better, because perhaps it simplifies the request execution process, therefore reducing costs and time

2/Compare the costs of all the queries by creating query plans for each one.

3/The EXPLAIN command gives you an estimated cost. To find out the actual speed of your queries, run them in pgAdmin 4. After you've run each query, a popup window will display its speed in milliseconds.

2 AND 3

Query	Subquery	CTE
1	166,06/52ms	166,07/53ms
2	1723,46/54ms	2038.13/53ms

4/Did the results surprise you? Write a few sentences to explain your answer.

Query 1: Performance is practically identical between the subquery and the CTE.

Query 2: The subquery is faster than the CTE (1723.46 vs. 2038.13).

>>> I was surprised, I didn't t except that subquery will be faster than CTE

Step 3:

Write 1 to 2 paragraphs on the challenges you faced when replacing your subqueries with CTEs.

When I replaced my subqueries with CTEs, one of the main challenges was clearly identifying which parts of the query could be isolated and named as intermediate steps. At first, I felt a real cognitive overload, as I was overwhelmed by too much information to process at once, which sometimes led to confusion and disorientation when facing the many lines of code. It was important to ensure that each CTE returned exactly the necessary columns and that the aliases were consistent, otherwise the query would fail. The logic remained the same as in the subqueries, but the structure required greater rigor in defining names and dependencies.