Databases & SQL for Analysts

3.9: Common Table Expressions

1. The Average Amount Paid by The Top 5 Customers

- **Query & answer:**

```
1. -- Find the average amount paid by the top 5 customers
2.
3. WITH average_amount_paid_cte (
4.                                customer_id, first_name, last_name,
5.                                 country,
6.                                 city)
7. AS (
8.     SELECT A.customer_id, A.first_name, A.last_name,
9.            D.country,
10.           C.city,
11.           SUM(E.amount) AS total_amount_paid
12.    FROM public.payment E
13.    RIGHT JOIN public.customer A ON E.customer_id = A.customer_id
14.    LEFT JOIN public.address B ON A.address_id = B.address_id
15.    LEFT JOIN public.city C ON B.city_id = C.city_id
16.    LEFT JOIN public.country D ON C.country_id = D.country_id
17.    WHERE C.city IN (
18.                     SELECT C.city
19.                     FROM public.city C
20.                     RIGHT JOIN public.address B
21.                                ON C.city_id = B.city_id
22.                     LEFT JOIN public.country D
23.                                ON C.country_id = D.country_id
24.                     RIGHT JOIN public.customer A
25.                                ON B.address_id = A.address_id
26.                     WHERE D.country IN (
```

```
27.                                         SELECT D.country
28.                                         FROM public.customer A
29.                                         LEFT JOIN public.address B
30.                                                 ON       A.address_id       =
   B.address_id
31.                                         LEFT JOIN public.city C
32.                                               ON B.city_id = C.city_id
33.                                         LEFT JOIN public.country D
34.                                               ON C.country_id = D.country_id
35.                                         GROUP BY country
36.                                         ORDER BY COUNT(customer_id) DESC
37.                                         LIMIT 10
38.                                         )
39.                       GROUP BY C.city, D.country
40.                       ORDER BY COUNT(customer_id) DESC
41.                       LIMIT 10
42.                       )
43.     GROUP BY A.customer_id, A.first_name, A.last_name,
44.            D.country,
45.            C.city
46.     ORDER BY total_amount_paid DESC
47.     LIMIT 5
48.     )
49. SELECT AVG (total_amount_paid) AS average_amount_paid
50. FROM average_amount_paid_cte
```

average_amount_paid
107.354

- **Process:**

As there is previously a subquery to find the average amount paid by the top 5 customers, the steps to the CTE approach are:

  - Step 1: Create the CTE name (average_amount_paid_cte)
  - Step 2: Declare the CTE with the WITH statement
  - Step 3: Include the CTE in the main query
  - Step 4: Replace the subquery with the CTE in the main query
  - Step 5: Execute the query
- **CTE vs. Subquery comparison:**

With the CTE: Total query runtime: 199 msec. 1 rows affected.

"Aggregate (cost=166.06..166.07 rows=1 width=32)"

With the Subquery: Total query runtime: 176 msec. 1 rows affected.

"Aggregate (cost=166.06..166.07 rows=1 width=32)"

The CTE took longer to run than the subquery, despite the fact that they appeared to have the same cost. This might be due to the fact that the CTE needs more resources to process than the subquery.

2. Number of The Top 5 Customers Based Within Each Country

- **Query & answer:**

```
1. -- Find out how many of the top 5 customers are based within each country
2.
3. WITH top_5_customers_cte AS (
4.          SELECT A.customer_id, A.first_name, A.last_name,
5.                 D.country,
6.                 C.city,
7.                 SUM(E.amount) AS total_amount_paid
8.          FROM public.payment E
9.          RIGHT JOIN public.customer A ON E.customer_id = A.customer_id
10.         LEFT JOIN public.address B ON A.address_id = B.address_id
11.         LEFT JOIN public.city C ON B.city_id = C.city_id
12.         LEFT JOIN public.country D ON C.country_id = D.country_id
13.         WHERE C.city IN (
14.                          SELECT C.city
15.                          FROM public.city C
16.                          RIGHT JOIN public.address B
17.                              ON C.city_id = B.city_id
18.                          LEFT JOIN public.country D
19.                              ON C.country_id = D.country_id
20.                          RIGHT JOIN public.customer A
21.                              ON B.address_id = A.address_id
22.                          WHERE D.country IN (
23.                                       SELECT D.country
24.                                              FROM public.customer A
25.                                               LEFT JOIN public.address B
26.                                                  ON A.address_id = B.address_id
27.                                     LEFT JOIN public.city C
28.                                                  ON B.city_id = C.city_id
29.                                     LEFT JOIN public.country D
30.                                                  ON C.country_id = D.country_id
31.                                     GROUP BY country
32.                                     ORDER BY COUNT(customer_id) DESC
33.                                     LIMIT 10
34.                                       )
```

```
35.                          GROUP BY C.city, D.country
36.                          ORDER BY COUNT(customer_id) DESC
37.                          LIMIT 10
38.                          )
39.         GROUP BY A.customer_id, A.first_name, A.last_name,
40.                  D.country,
41.                  C.city
42.         ORDER BY total_amount_paid DESC
43.         LIMIT 5
44.           )
45. SELECT D.country,
46.        COUNT( DISTINCT A.customer_id) AS all_customer_count,
47.        COUNT (DISTINCT top_5_customers_cte.customer_id) AS top_customer_count
48. FROM public.customer A
49. LEFT JOIN public.address B ON A.address_id = B.address_id
50. LEFT JOIN public.city C ON B.city_id = C.city_id
51. LEFT JOIN public.country D ON C.country_id = D.country_id
52. LEFT JOIN top_5_customers_cte
53.            ON D.country = top_5_customers_cte.country
54. GROUP BY D.country
55. ORDER BY top_customer_count DESC
56. LIMIT 5
```

country	all_customer_count	top_customer_count
Mexico	30	2
United States	36	1
India	60	1
Turkey	15	1
American Samoa	1	0

- **Process:**

As there is previously a subquery to find the number of the top 5 customers based within each country, the steps to the CTE approach are:

  - Step 1: Create the CTE name (top_5_customers_cte)
  - Step 2: Declare the CTE with the WITH statement
  - Step 3: Include the CTE in the main query
  - Step 4: Replace the subquery with the CTE in the main query
  - Step 5: Execute the query

- **CTE vs. Subquery comparison:**

With the CTE: Total query runtime: 113 msec. 5 rows affected.

"Limit (cost=267.83..267.84 rows=5 width=25)"

With the Subquery: Total query runtime: 158 msec. 5 rows affected.

"Limit (cost=267.83..267.84 rows=5 width=25)"

The CTE took less time to run compared to the subquery, despite the fact that they appeared to have the same cost. This might be due to the fact that the CTE is pre-defined and therefore is more efficient.

3. Reflection

- **Challenges you faced when replacing your subqueries with CTEs:**

CTEs were more complicated to create, and it took longer to comprehend the logic of the query. With the CTEs, it appeared that a more in-depth grasp of the database structure and the data it contained was necessary.