Omid Soltani
Data Analytics Immersion – Task 3.9

Step 1: Answer the business questions from steps 1 and 2 of task 3.8 using CTEs

3.8.1 Find the average amount paid by the top 5 customers

Query Query History

```sql
WITH top_customers AS
  (SELECT c.customer_id,
          c.first_name,
          c.last_name,
          co.country,
          ci.city,
          SUM(p.amount) AS total_amount_paid
   FROM customer c
   JOIN address a ON c.address_id = a.address_id
   JOIN city ci ON a.city_id = ci.city_id
   JOIN country co ON ci.country_id = co.country_id
   JOIN payment p ON c.customer_id = p.customer_id
   WHERE ci.city IN ('Aurora',
                     'Acua',
                     'Citrus Heights',
                     'Iwaki',
                     'Ambattur',
                     'Shanwei',
                     'So Leopoldo',
                     'Teboksary',
                     'Tianjin',
                     'Cianju')
   GROUP BY c.customer_id,
            c.first_name,
            c.last_name,
            co.country,
            ci.city
   ORDER BY total_amount_paid DESC
   LIMIT 5)
SELECT AVG(total_amount_paid) AS average
FROM top_customers;
```

Data Output Messages Notifications

	average numeric
1	105.5540000000000000

The original query contained a subquery to find the top 5 customers based on certain criteria. The first step was to recognize this subquery and understand its purpose.

The next step involved transforming the subquery into a CTE. I named the CTE **top_customers** and placed the existing subquery logic inside it. This helped in making the code cleaner.

Then I modified the main query to reference the CTE instead of the subquery by replacing the subquery reference with the CTE name (**top_customers**) and adjusting the join conditions. To enhance readability, I used aliases for the table names in both the CTE and the main query.

Throughout the process, I ensured that the changes did not alter the logic of the original query. The CTE was correctly referenced in the main query, and the join conditions remained accurate.

Omid Soltani
Data Analytics Immersion – Task 3.9

3.8.2 Find out how many of the top 5 customers are based within each country

Query Query History

```sql
WITH top_customers AS
  (SELECT c.customer_id,
          c.first_name,
          c.last_name,
          co.country_id,
          ci.city,
          SUM(p.amount) AS total_amount_paid
   FROM customer c
   INNER JOIN address a ON c.address_id = a.address_id
   INNER JOIN city ci ON a.city_id = ci.city_id
   INNER JOIN country co ON ci.country_id = co.country_id
   INNER JOIN payment p ON c.customer_id = p.customer_id
   WHERE ci.city IN ('Aurora',
                     'Acua',
                     'Citrus Heights',
                     'Iwaki',
                     'Ambattur',
                     'Shanwei',
                     'So Leopoldo',
                     'Teboksary',
                     'Tianjin',
                     'Cianju')
   GROUP BY c.customer_id,
            c.first_name,
            c.last_name,
            co.country_id,
            ci.city
   ORDER BY total_amount_paid DESC
   LIMIT 5)
SELECT co.country,
       COUNT(DISTINCT c.customer_id) AS all_customer_count,
       COUNT(DISTINCT tc.customer_id) AS top_customer_count
FROM customer c
INNER JOIN address a ON c.address_id = a.address_id
INNER JOIN city ci ON a.city_id = ci.city_id
INNER JOIN country co ON ci.country_id = co.country_id
LEFT JOIN top_customers tc ON c.customer_id = tc.customer_id
GROUP BY co.country
ORDER BY all_customer_count DESC
LIMIT 5;
```

Data Output Messages Notifications

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

Total rows: 5 of 5 Query complete 00:00:00.059

To achieve our objective, I created a CTE named 'top_customers' to nest the subquery, retrieving necessary information from joined tables. In this subquery, I applied filters for specific cities, aggregated data for each customer, and ordered the results by the total amount paid in descending order, limiting to the top 5 customers.

In the main query, I employed INNER JOINs to establish connections between the 'customer', 'address', 'city', and 'country' tables based on their respective foreign keys. A LEFT JOIN was then performed with the top_customers' CTE on the 'customer_id' to bring in information about the top 5 customers. Using COUNT(DISTINCT), I calculated both the total number of customers ('all_customer_count') and the count of top customers ('top_customer_count') in each country. The results were grouped by country and ordered by 'all_customer_count' in descending order, with a LIMIT of 5 to retrieve the top 5 countries.

In terms of readability and consistency, I maintained clear aliasing for table names and columns throughout the query. The logical structure ensures ease of comprehension. I tested the query in a database environment to confirm correctness and performance, especially with larger datasets. This step-by-step approach facilitates clarity, testing, and potential future modifications.

Omid Soltani
Data Analytics Immersion – Task 3.9

Step 2: Compare the performance of your CTEs and subqueries.

Average Amount Paid by the Top 5 Customers

CTE

Query | Query History

```sql
EXPLAIN
WITH top_customers AS
  (SELECT c.customer_id,
          c.first_name,
          c.last_name,
          co.country,
          ci.city,
          SUM(p.amount) AS total_amount_paid
   FROM customer c
   JOIN address a ON c.address_id = a.address_id
   JOIN city ci ON a.city_id = ci.city_id
   JOIN country co ON ci.country_id = co.country_id
   JOIN payment p ON c.customer_id = p.customer_id
   WHERE ci.city IN ('Aurora',
                     'Acua',
                     'Citrus Heights',
                     'Iwaki',
                     'Ambattur',
                     'Shanwei',
                     'So Leopoldo',
                     'Teboksary',
                     'Tianjin',
                     'Cianju')
   GROUP BY c.customer_id,
            c.first_name,
            c.last_name,
            co.country,
            ci.city
   ORDER BY total_amount_paid DESC
   LIMIT 5)
SELECT AVG(total_amount_paid) AS average
FROM top_customers;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Aggregate (cost=64.49..64.50 rows=1 width=32)
2	-> Limit (cost=64.41..64.43 rows=5 width=67)
3	-> Sort (cost=64.41..65.02 rows=244 width=67)
4	Sort Key: (sum(p.amount)) DESC
5	-> HashAggregate (cost=57.31..60.36 rows=244 width=67)

Total rows: 22 of 22 | Query complete 00:00:00.254

Subquery

Query | Query History

```sql
EXPLAIN
SELECT AVG(total_amount_paid) AS average
FROM
  (SELECT c.customer_id,
          c.first_name,
          c.last_name,
          co.country,
          ci.city,
          SUM(p.amount) AS total_amount_paid
   FROM customer c
   JOIN address a ON c.address_id = a.address_id
   JOIN city ci ON a.city_id = ci.city_id
   JOIN country co ON ci.country_id = co.country_id
   JOIN payment p ON c.customer_id = p.customer_id
   WHERE ci.city IN ('Aurora',
                     'Acua',
                     'Citrus Heights',
                     'Iwaki',
                     'Ambattur',
                     'Shanwei',
                     'So Leopoldo',
                     'Teboksary',
                     'Tianjin',
                     'Cianju')
   GROUP BY c.customer_id,
            c.first_name,
            c.last_name,
            co.country,
            ci.city
   ORDER BY total_amount_paid DESC
   LIMIT 5) AS top_customers;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Aggregate (cost=64.49..64.50 rows=1 width=32)
2	-> Limit (cost=64.41..64.43 rows=5 width=67)
3	-> Sort (cost=64.41..65.02 rows=244 width=67)
4	Sort Key: (sum(p.amount)) DESC
5	-> HashAggregate (cost=57.31..60.36 rows=244 width=67)

Total rows: 22 of 22 | Query complete 00:00:00.046

Aggregate (cost=64.49..64.50 rows=1 width=32)

Omid Soltani
Data Analytics Immersion – Task 3.9

Top 5 Customers within Each Country

CTE

Query Query History

```sql
EXPLAIN
WITH top_customers AS
  (SELECT c.customer_id,
          c.first_name,
          c.last_name,
          co.country_id,
          ci.city,
          SUM(p.amount) AS total_amount_paid
   FROM customer c
   INNER JOIN address a ON c.address_id = a.address_id
   INNER JOIN city ci ON a.city_id = ci.city_id
   INNER JOIN country co ON ci.country_id = co.country_id
   INNER JOIN payment p ON c.customer_id = p.customer_id
   WHERE ci.city IN ('Aurora',
                     'Acua',
                     'Citrus Heights',
                     'Iwaki',
                     'Ambattur',
                     'Shanwei',
                     'So Leopoldo',
                     'Teboksary',
                     'Tianjin',
                     'Cianju')
   GROUP BY c.customer_id,
            c.first_name,
            c.last_name,
            co.country_id,
            ci.city
   ORDER BY total_amount_paid DESC
   LIMIT 5)
SELECT co.country,
       COUNT(DISTINCT c.customer_id) AS all_customer_count,
       COUNT(DISTINCT tc.customer_id) AS top_customer_count
FROM customer c
INNER JOIN address a ON c.address_id = a.address_id
INNER JOIN city ci ON a.city_id = ci.city_id
INNER JOIN country co ON ci.country_id = co.country_id
LEFT JOIN top_customers tc ON c.customer_id = tc.customer_id
GROUP BY co.country
ORDER BY all_customer_count DESC
LIMIT 5;
```

Data Output Messages Notifications

	QUERY PLAN text
1	Limit (cost=166.66..166.68 rows=5 width=25)
2	-> Sort (cost=166.66..166.94 rows=109 width=25)
3	Sort Key: (count(DISTINCT c.customer_id)) DESC
4	-> GroupAggregate (cost=157.77..164.85 rows=109 width=25)
5	Group Key: co.country
6	-> Sort (cost=157.77..159.27 rows=599 width=17)

Total rows: 45 of 45 Query complete 00:00:00.072

Subquery

Query Query History

```sql
EXPLAIN
SELECT co.country,
       COUNT(DISTINCT c.customer_id) AS all_customer_count,
       COUNT(DISTINCT tc.customer_id) AS top_customer_count
FROM customer c
INNER JOIN address a ON c.address_id = a.address_id
INNER JOIN city ci ON a.city_id = ci.city_id
INNER JOIN country co ON ci.country_id = co.country_id
LEFT JOIN
  (SELECT c.customer_id,
          c.first_name,
          c.last_name,
          co.country_id,
          ci.city
   FROM customer c
   INNER JOIN address a ON c.address_id = a.address_id
   INNER JOIN city ci ON a.city_id = ci.city_id
   INNER JOIN country co ON ci.country_id = co.country_id
   INNER JOIN payment p ON c.customer_id = p.customer_id
   WHERE ci.city IN ('Aurora',
                     'Acua',
                     'Citrus Heights',
                     'Iwaki',
                     'Ambattur',
                     'Shanwei',
                     'So Leopoldo',
                     'Teboksary',
                     'Tianjin',
                     'Cianju')
   GROUP BY c.customer_id,
            c.first_name,
            c.last_name,
            co.country_id,
            ci.city
   ORDER BY SUM(p.amount) DESC
   LIMIT 5) tc ON c.customer_id = tc.customer_id
GROUP BY co.country
ORDER BY all_customer_count DESC
LIMIT 5;
```

Data Output Messages Notifications

	QUERY PLAN text
1	Limit (cost=166.66..166.68 rows=5 width=25)
2	-> Sort (cost=166.66..166.94 rows=109 width=25)
3	Sort Key: (count(DISTINCT c.customer_id)) DESC
4	-> GroupAggregate (cost=157.77..164.85 rows=109 width=25)
5	Group Key: co.country
6	-> Sort (cost=157.77..159.27 rows=599 width=17)
7	Sort Key: co.country, c.customer_id

Total rows: 45 of 45 Query complete 00:00:00.091

"Limit (cost=166.66..166.68 rows=5 width=25)"

1. I initially thought that using CTEs might result in better performance compared to subqueries. However, in my testing, both approaches exhibited the same performance in terms of costs.

2. When comparing the costs of all the queries by examining the query plans, I found that the costs for CTEs and subqueries were similar. The query planner estimated comparable costs for both methods, indicating that there wasn't a significant difference in performance.

3. The results did surprise me to some extent. I anticipated that CTEs, being a more structured and readable way to organize complex queries, might have some performance benefits. However, the similarity in costs suggests that, in the specific context of these queries, the database optimizer treated CTEs and subqueries similarly in terms of execution plans and efficiency.

Step 3

Incorporating Common Table Expressions (CTEs) instead of subqueries posed some challenges. Initially, understanding the concept of CTEs required a bit of extra effort, as it was a new technique for me. Figuring out how to structure the CTE and connect it seamlessly with the main query was a bit tricky, but examples and practice helped to overcome this hurdle.

The transition from subqueries to CTEs involved adapting my mindset to a more modular approach. While subqueries are embedded within the main query, CTEs provide a cleaner and more organized way to structure complex queries. Once I grasped the idea of breaking down the task into smaller, manageable parts with a CTE, it became a valuable tool for improving code readability and maintainability. Overall, the challenges were more about adapting to a new methodology, but the benefits in terms of code structure and clarity made the learning process worthwhile.