Task 3.9

Step 1

Step 1.1

Query | Query History

```sql
WITH total_amount_cte (customer_id, first_name, last_name, country, city, total_amount) AS
(SELECT a.customer_id,
     a.first_name,
     a.last_name,
     d.country,
     c.city,
     SUM (e.amount) AS total_amount
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
INNER JOIN payment E ON A.customer_id = E.customer_id
WHERE c.city IN ('Aurora','Tokat','Tarsus','Atlixco','Emeishan','Pontianak','Shimoga','Aparecida de Goinia','
GROUP BY a.customer_id,
         first_name,
         last_name,
         country,
         city
ORDER BY total_amount desc
LIMIT 5)
SELECT AVG (total_amount) AS average
FROM total_amount_cte
```

	average numeric
1	120.3220000000000000

Step 1.2

Query | Query History

```sql
WITH customer_full_info (customer_id, first_name, last_name, address, city_id, city, countr
(SELECT a.customer_id, a.first_name, a.last_name, b.address, b.city_id, c.city, 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),
top_5_customers (customer_id, first_name, last_name, country, city, total_amount) AS
(SELECT a.customer_id,
       a.first_name,
       a.last_name,
       d.country,
       c.city,
       sum (e.amount) as total_amount
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
INNER JOIN payment E ON A.customer_id = E.customer_id
WHERE c.city IN ('Aurora','Tokat','Tarsus','Atlixco','Emeishan','Pontianak','Shimoga','Apar
GROUP BY a.customer_id,
         first_name,
         last_name,
         country,
         city
ORDER BY total_amount desc
limit 5 )
SELECT a.country,
       COUNT(DISTINCT b.customer_id) AS top_5_customer_count,
       COUNT(DISTINCT a.customer_id) AS all_customer_count
FROM customer_full_info a
```

```
29          COUNT(DISTINCT a.customer_id) AS all_customer_count
30  FROM customer_full_info a
31  LEFT JOIN top_5_customers b ON a.customer_id = b.customer_id
32  GROUP BY a.country
33  ORDER BY top_5_customer_count desc
34  LIMIT 5
```

	country character varying (50)	top_5_customer_count bigint	all_customer_count bigint
1	Mexico	1	30
2	Turkey	1	15
3	China	1	53
4	Indonesia	1	14
5	United States	1	36

In step 1 what I did to transform the subquery into a common table expression was taking the whole subquery that I created in the last task and getting rid of the outer query and change it with the with statement to create the cte and then added what was the outer main query to the end to get the same result by consulting the cte instead of using the subquery, and on step 2 I had to do pretty much the same but dividing the subqueries in 2 different cte that I joined at the end while making my main query.

Step 2

Step 1 subquery

Messages

```
Successfully run. Total query runtime: 109 msec. 22 rows
affected.
```

Step 1 cte

Messages

```
Successfully run. Total query runtime: 54 msec. 22 rows
affected.
```

Step 2 subquery

Messages

```
Successfully run. Total query runtime: 67 msec. 46 rows
affected.
```

Step 2 cte

Messages

```
Successfully run. Total query runtime: 54 msec. 45 rows
affected.
```

In the first step there is a big difference between the query runtime of the subquery and the cte, in this case the cte is faster an more efficient to process, but in the second step the subquery and the cte doesn't have a big difference between them but still the subquery is less efficient than the cte , it was not surprising that the cte was faster than the subquery because in some cases cte are the way to go because they are usually more efficient to use in constant changing tables.

Step 3

Some of the challenges that I faced when transforming the subqueries to cte is that when typing a cte we have to take into consideration that we are creating a temporary table in which our main query is going to look for the information like in any other table, and when we are using subqueries we assign the subquery in a specific spot and it can be used just one time unless we retype it, because of this reason, cte are more useful when we have to get the information of a special table multiple times.