Exercise 3.9

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

1. **Rewrite your queries from steps 1 and 2 of task 3.8 as CTEs.**
2. **Copy-paste your CTEs and their outputs into your answers document.**

3.8 Step 1: Find the average amount paid by the top 5 customers.

**WITH average\_amount\_cte (customer\_id, first\_name, last\_name, city) AS**

**(**

**SELECT**

**customer.customer\_id,**

**customer.first\_name,**

**customer.last\_name,**

**city.city,**

**SUM(payment.amount) AS total\_amount\_paid**

**FROM**

**customer**

**JOIN**

**payment ON customer.customer\_id = payment.customer\_id**

**JOIN**

**address ON customer.address\_id = address.address\_id**

**JOIN**

**city ON address.city\_id = city.city\_id**

**WHERE**

**city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule(Dhulia)',**

**'Kurashiki', 'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')**

**GROUP BY**

**customer.customer\_id,**

**customer.first\_name,**

**customer.last\_name,**

**city.city**

**ORDER BY**

**total\_amount\_paid DESC**

**LIMIT**

**5**

**)**

**SELECT AVG(total\_amount\_paid)**

**FROM average\_amount\_cte**

A screenshot of a computer program

Description automatically generated

3.8 Step 2: Find out how many of the top 5 customers you identified in step 1 are based within each country.

**WITH top\_5\_count\_cte (customer\_id, first\_name, last\_name, city, country) AS**

**(SELECT**

**customer.customer\_id,**

**customer.first\_name,**

**customer.last\_name,**

**city.city,**

**country.country,**

**SUM(payment.amount) AS total\_amount\_paid**

**FROM**

**customer**

**JOIN**

**payment ON customer.customer\_id = payment.customer\_id**

**JOIN**

**address ON customer.address\_id = address.address\_id**

**JOIN**

**city ON address.city\_id = city.city\_id**

**JOIN country ON city.country\_id = country.country\_id**

**WHERE**

**city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule(Dhulia)',**

**'Kurashiki', 'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')**

**GROUP BY**

**customer.customer\_id,**

**customer.first\_name,**

**customer.last\_name,**

**city.city,**

**country.country**

**ORDER BY**

**total\_amount\_paid DESC**

**LIMIT**

**5)**

**SELECT country.country,**

**COUNT(DISTINCT customer.customer\_id) AS all\_customer\_count,**

**COUNT(top\_5\_count\_cte) AS top\_customer\_count**

**FROM customer**

**JOIN address ON customer.address\_id = address.address\_id**

**JOIN city ON address.city\_id = city.city\_id**

**JOIN country ON city.country\_id = country.country\_id**

**LEFT JOIN top\_5\_count\_cte ON customer.customer\_id = top\_5\_count\_cte.customer\_id**

**GROUP BY country.country**

**ORDER BY all\_customer\_count DESC**

**LIMIT 10;**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

1. **Write 2 to 3 sentences explaining how you approached this step, for example, what you did first, second, and so on.**
   1. For the 1st query, I copied and pasted the queries from 3.8 into a new query. Then I took the subquery and re-wrote it as a CTE using WITH and AS. Then I added the SELECT AVG to get the average, and FROM to reference the new CTE that was created.
   2. For the 2nd query, I did the same as the first, just inserting the CTE above the main query and changed the LEFT JOIN so it was referencing the CTS instead of the original subquery.

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

1. **Which approach do you think will perform better and why?**
   1. I think the CTE will perform better since the table only has to be defined once and then the query can just reference the table that was already created.
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.**
   1. Query 1 - 3.8 : "Aggregate (cost=60.37..60.38 rows=1 width=32)" *42 msec*.
   2. Query 1 – 3.9 : "Aggregate (cost=60.37..60.38 rows=1 width=32)" *71 msec.*
   3. Query 2 – 3.8 : "Limit (cost=167.17..167.19 rows=10 width=25)" *56 msec.*
   4. Query 2 - 3.9 : "Limit (cost=167.17..167.19 rows=10 width=25)" *53 msec.*
4. **Did the results surprise you? Write a few sentences to explain your answer.**
   1. The results were somewhat surprising because the CTE query was actually slower in the first query. This shows that the CTE is now always faster and it is always a good idea to performance test your queries to decide which is the more efficient.

**Step 3:**

1. **Write 1 to 2 paragraphs on the challenges you faced when replacing your subqueries with CTEs.**
   1. The first query was fairly straightforward to replace as most of the query was within the subquery. However, the second query was much more difficult. Initially I struggled with understanding how to shift everything down to allow for the CTE while still keeping all the join statements together and making sure the end result was still the same as the original query from 3.8. However, after successfully replacing the subquery, it made much more sense and hopefully it will make any similar problems in the future easier now that I’ve seen where everything needs to go.