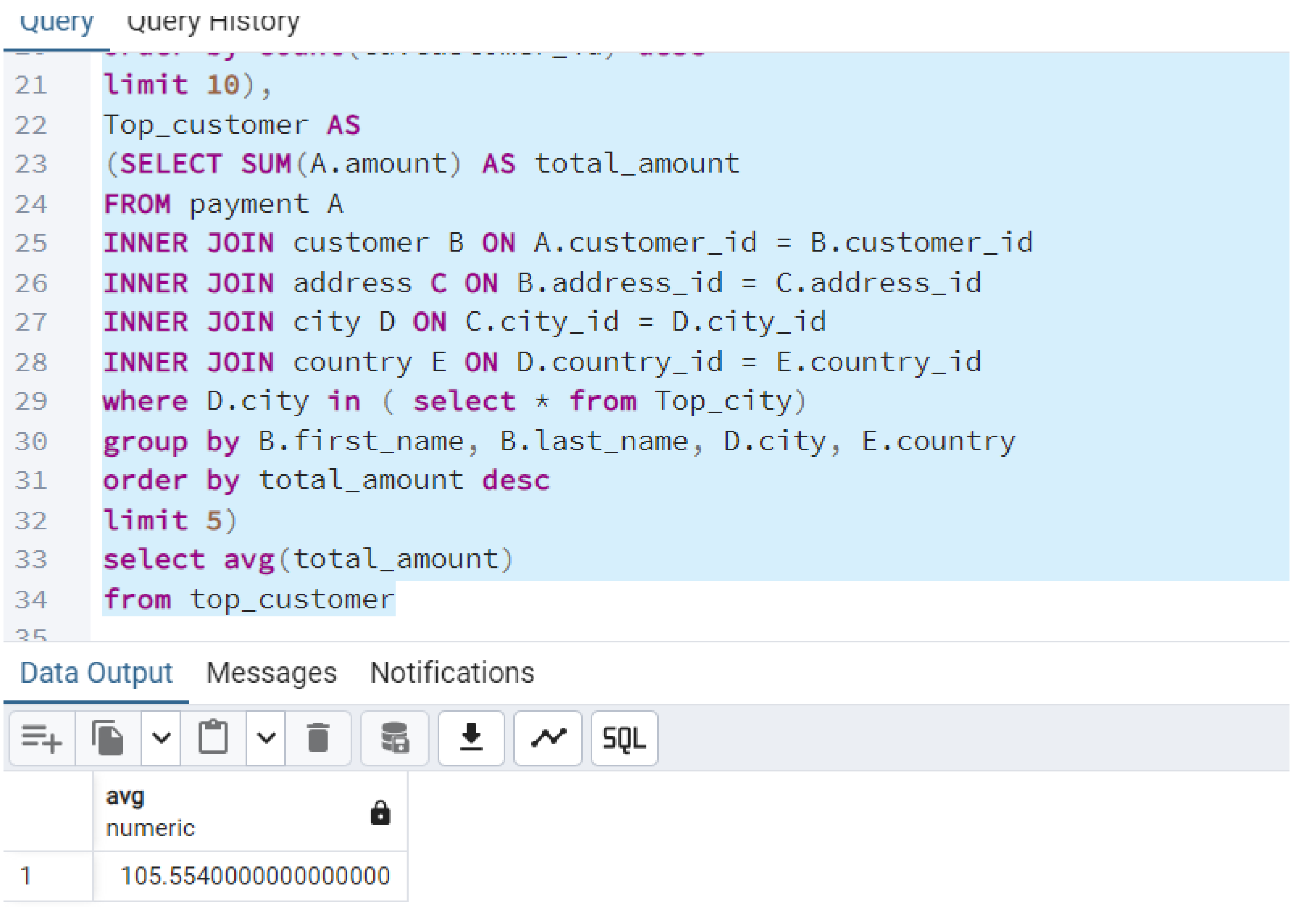
# SQL Exercise Report - Exercise 3.9

Author: Alexandru Cojocari

Task 1A - Calculating Average Total Amount from Top Customers

To calculate the average total payment amount of the top five customers residing in the ten most populated cities (from the top ten countries with the most customers), the following CTEs were defined:

1. Top_10_countries: Extracts the ten countries with the highest number of customers.
2. Top_city: Identifies the top cities based on customer count within those top countries.
3. Top_customer: Sums up the total payments from customers in those top cities and retrieves the top five based on total payment amount.



```
limit 10),
Top_customer AS
(SELECT SUM(A.amount) AS total_amount
FROM payment A
INNER JOIN customer B ON A.customer_id = B.customer_id
INNER JOIN address C ON B.address_id = C.address_id
INNER JOIN city D ON C.city_id = D.city_id
INNER JOIN country E ON D.country_id = E.country_id
where D.city in ( select * from Top_city)
group by B.first_name, B.last_name, D.city, E.country
order by total_amount desc
limit 5)
select avg(total_amount)
from top_customer
```

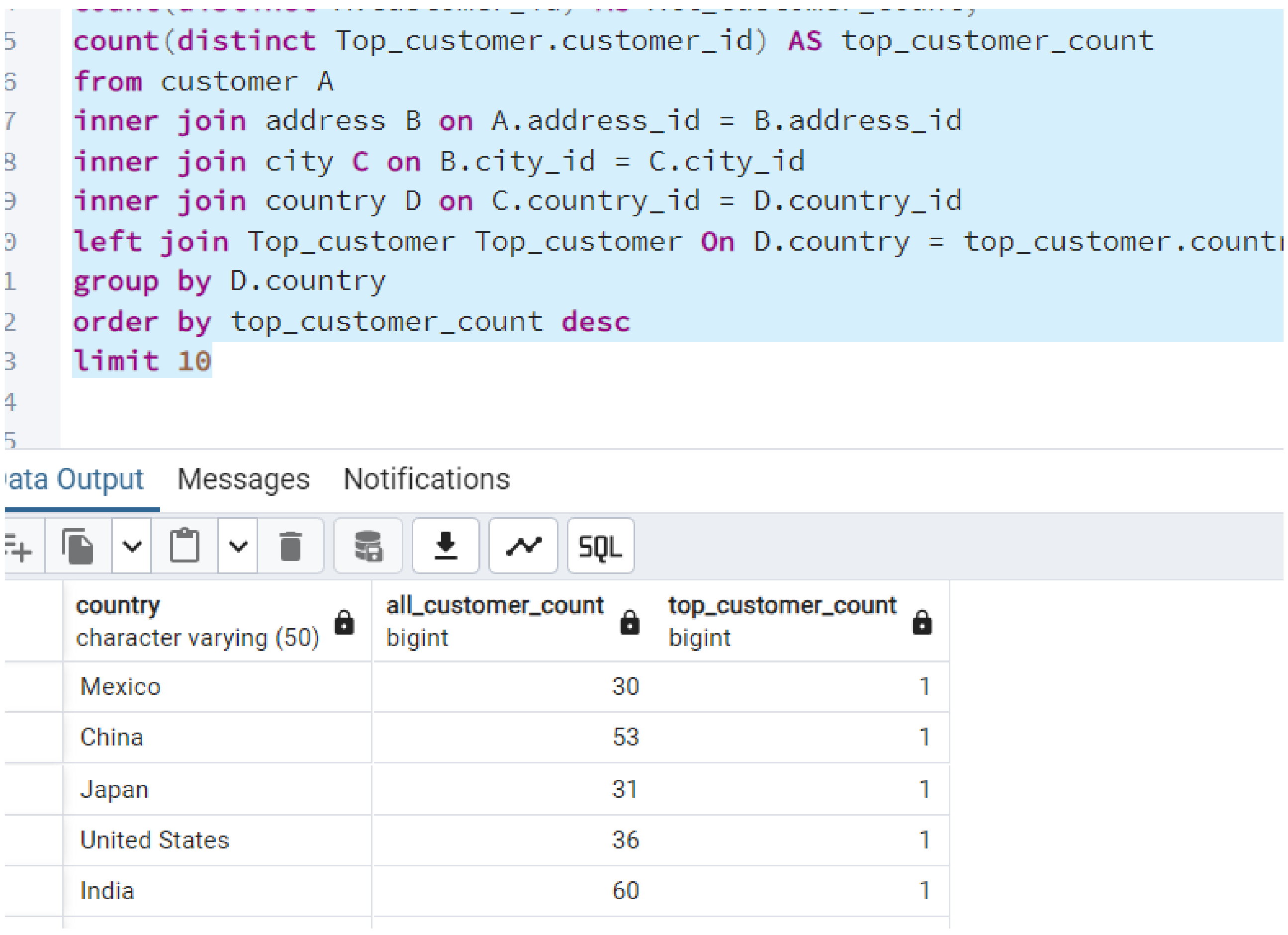
| | avg numeric |
|---|---|
| 1 | 105.5540000000000000 |

Task 1B - Counting Customers by Country and Comparing Top Performers

This task enhances the previous query to count all distinct customers in each country and compare them with the number of top customers (from the earlier result) in those same countries.

Reflection: Transitioning from Subqueries to CTEs

To refactor the original subqueries into CTEs, I began by isolating the deepest filtering layer: the top countries. Then, I created a second CTE for top cities, using the previous CTE to retain logical linkage. Lastly, a third CTE was defined for top customers. This structured method ensures clarity and modularity in the query design.



```
count(distinct Top_customer.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_customer Top_customer On D.country = top_customer.count
group by D.country
order by top_customer_count desc
limit 10
```

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

Task 2 - Performance Comparison: CTE vs. Subquery

I found that CTEs offered slightly better performance and significantly improved readability. The table below shows the execution cost and speed for both approaches:

| Query Type | CTE Cost | Subquery Cost | CTE Speed | Subquery Speed |
|---|---|---|---|---|
| Query 1 | 167 | 127.63 | 92 ms | 125 ms |

Query 2    | 268.99   | 231.70          | 107 ms     | 197 ms

Observation: While performance differences were minor, the structure and maintainability of CTEs make them favorable, especially in complex queries.

Task 3 - Challenges Faced

One of the primary challenges was adjusting the nested subqueries into standalone CTEs without altering the result. This required rethinking how data flows between each segment. I had to troubleshoot the CTE logic more frequently to ensure accurate output due to slight variations in CTE structure requirements.