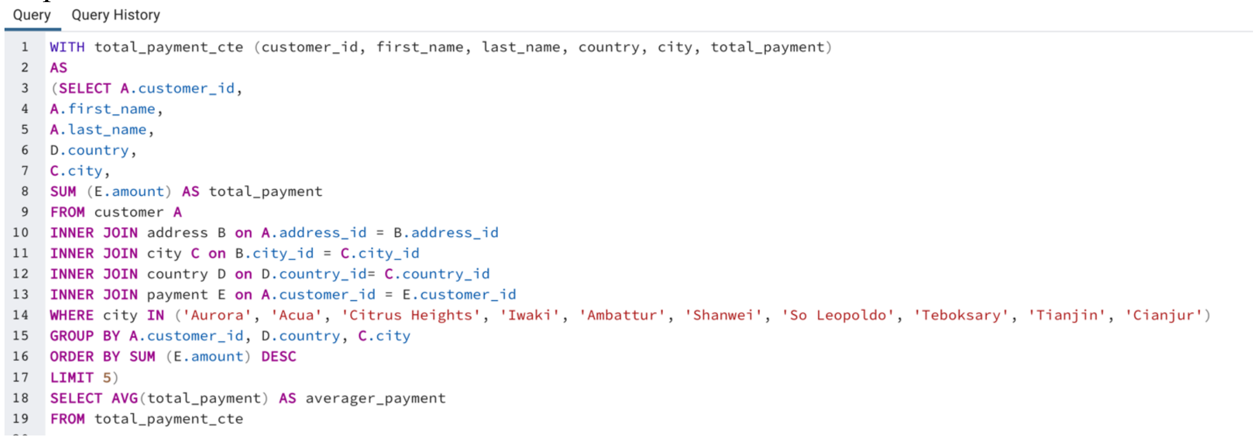
Step 1: Answer the business questions from step 1 and 2 of task 3.8 using CTEs

- Rewrite your queries from steps 1 and 2 of task 3.8 as CTEs.

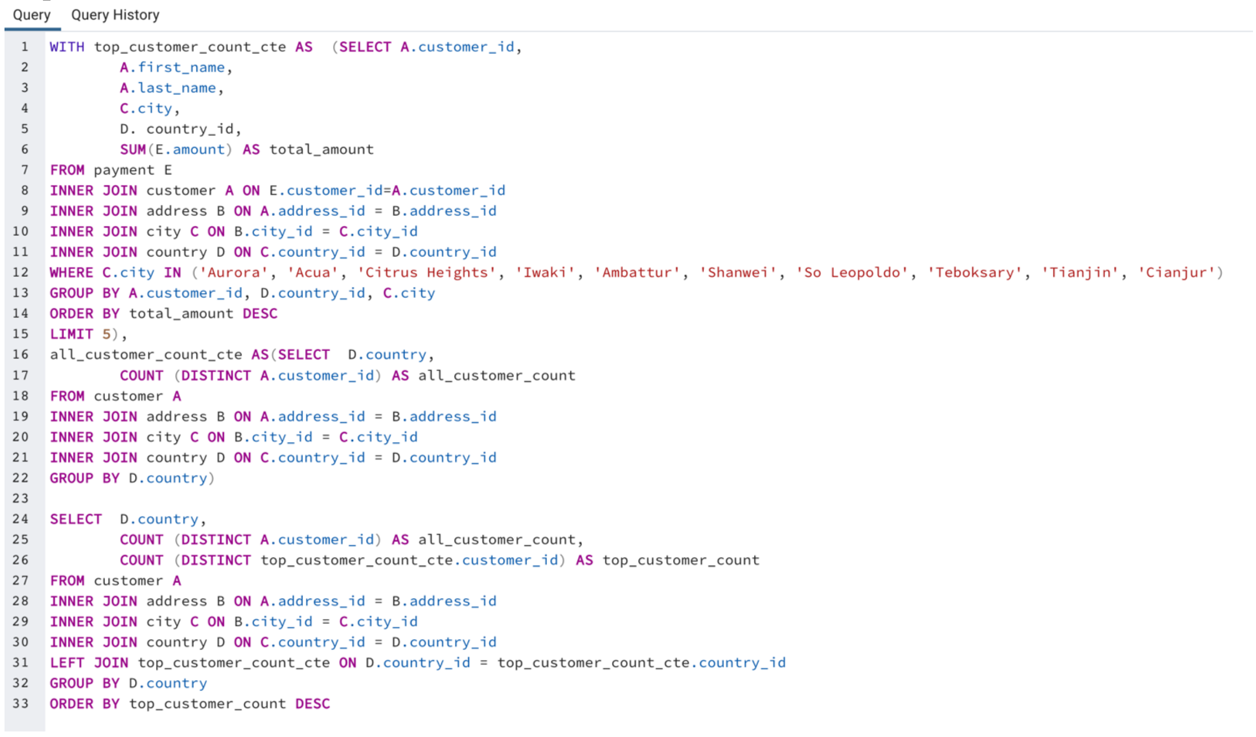
  Step 1:



Query Query History

```
WITH total_payment_cte (customer_id, first_name, last_name, country, city, total_payment)
AS
(SELECT A.customer_id,
A.first_name,
A.last_name,
D.country,
C.city,
SUM (E.amount) AS total_payment
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 D.country_id= C.country_id
INNER JOIN payment E on A.customer_id = E.customer_id
WHERE city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')
GROUP BY A.customer_id, D.country, C.city
ORDER BY SUM (E.amount) DESC
LIMIT 5)
SELECT AVG(total_payment) AS averager_payment
FROM total_payment_cte
```

  Step 2:



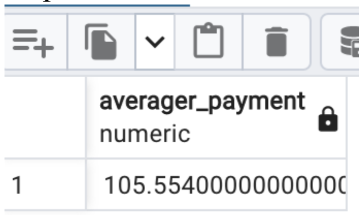
Query Query History

```
WITH top_customer_count_cte AS  (SELECT A.customer_id,
        A.first_name,
        A.last_name,
        C.city,
        D. country_id,
        SUM(E.amount) AS total_amount
FROM payment E
INNER JOIN customer A ON E.customer_id=A.customer_id
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
WHERE C.city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')
GROUP BY A.customer_id, D.country_id, C.city
ORDER BY total_amount DESC
LIMIT 5),
all_customer_count_cte AS(SELECT  D.country,
        COUNT (DISTINCT A.customer_id) AS all_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
GROUP BY D.country)

SELECT  D.country,
        COUNT (DISTINCT A.customer_id) AS all_customer_count,
        COUNT (DISTINCT top_customer_count_cte.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_count_cte ON D.country_id = top_customer_count_cte.country_id
GROUP BY D.country
ORDER BY top_customer_count DESC
```

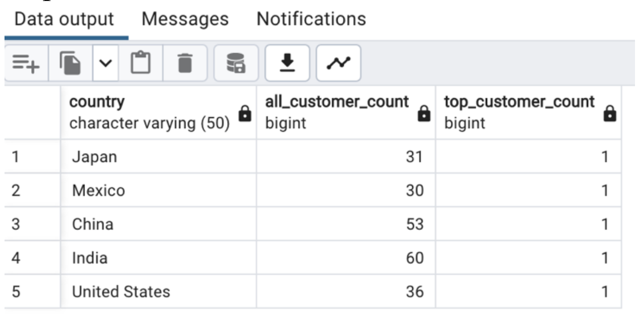
- Copy-paste your CTEs and their outputs into your answers document.

  Step 1:



	averager_payment numeric
1	105.5540000000000

Step 2:



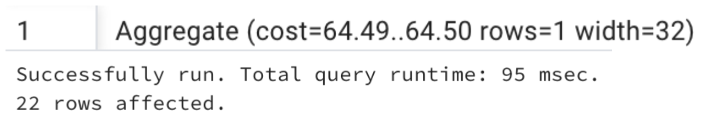
Data output | Messages | Notifications

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

- Write 2 to 3 sentences explaining how you approached this step, for example, what you did first, second, and so on.
    - As CTEs are like the VIEW, I considered it as a temporary table, so I just first create on and assign it with names at the start of the main query, and use it as I will us any other table in the database.

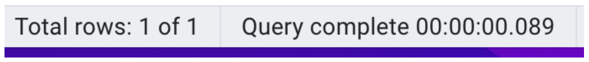
Step 2: Compare the performance of your CTEs and subqueries.

- Which approach do you think will perform better and why?
    - It really depends on the accessibility, readability and performance. Personally, I would prefer CTEs then subqueries, as it is more clear to read and more accessible than the subqueries.
- Compare the costs of all the queries by creating query plans for each one. The EXPLAIN command gives you an estimated cost. To find out the actual speed of your queries, run them in pgAdmin 4. After each query has been run, a pop-up window will display its speed in milliseconds.
    - STEP 1
        - Subquery
            - EXPLAIN:



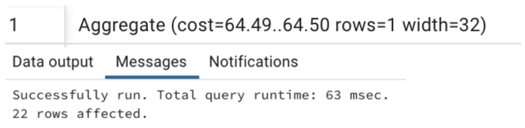
```
1   Aggregate (cost=64.49..64.50 rows=1 width=32)
Successfully run. Total query runtime: 95 msec.
22 rows affected.
```

            - ACTUAL:



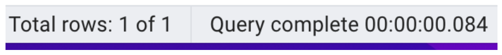
```
Total rows: 1 of 1    Query complete 00:00:00.089
```

            -
        - CTE
            - EPLAIN:



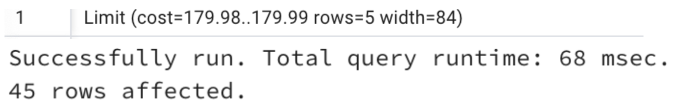
```
1   Aggregate (cost=64.49..64.50 rows=1 width=32)
Data output   Messages   Notifications
Successfully run. Total query runtime: 63 msec.
22 rows affected.
```

            - ACTUAL:



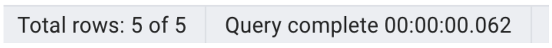
```
Total rows: 1 of 1    Query complete 00:00:00.084
```

- STEP 2
  - Subquery
    - EXPLAIN:



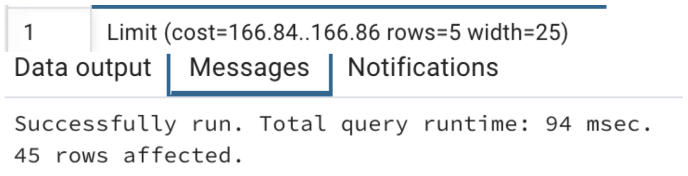
```
1   Limit (cost=179.98..179.99 rows=5 width=84)
Successfully run. Total query runtime: 68 msec.
45 rows affected.
```

    - ACTUAL:



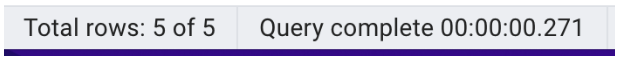
```
Total rows: 5 of 5   Query complete 00:00:00.062
```

  - CTE
    - EXPLAIN:



```
1   Limit (cost=166.84..166.86 rows=5 width=25)
Data output   Messages   Notifications
Successfully run. Total query runtime: 94 msec.
45 rows affected.
```

    - ACTUAL:



```
Total rows: 5 of 5   Query complete 00:00:00.271
```

- Did the results surprise you? Write a few sentences to explain your answer.
  - I was not very surprised with the result of Step 1, both the EXPLAIN cost and running time are similar as I guess due to the fact that both Subquery and CTE are very straight forward. But I was a little surprise by STEP2, as lots of analyst would agree that Subquery is their last resort compared to the CTE. But we can see that it is true that in EXPLAIN the cost for CTE is slightly lower than Subquery, however CTE's EXPLAIN and Actual runtime is longer than the Subquery. I guess you never know about the performance until you actually run the query or use the EXPLAIN syntax.

Step 3:

- Write 1 to 2 paragraphs on the challenges you faced when replacing your subqueries with CTEs.
  - For step 1, I did not find much challenge as it is very straight forward. The most difficult part of step 2, where the query is more complex, was defining two CTEs. After I learned how to write two CTEs for the same query, it was hard to combine them to get the output I needed.