Task 3.9 Jada Myrie

```sql
WITH average_amount_cte  AS
(SELECT SUM(A.amount) AS total_paid_amount,
        B.last_name,
        B.first_name,
        E.country,
        D.city,
        B.customer_id
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 city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur',
               'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')
GROUP BY country, city, first_name, last_name, B.customer_id
ORDER BY total_paid_amount DESC
LIMIT 5)
SELECT AVG (total_paid_amount) AS average_amount_paid
FROM average_amount_cte
```

Data Output | Messages | Notifications

	average_amount_paid numeric
1	105.5540000000000000

1. Write 2 to 3 sentences explaining how you approached this step, for example, what you did first, second, and so on.

I took the subquery from the previous exercise and entered it into pgadmin. I then began to create a WITH statement utilizing the main statement of the previous query which was the average amount. I named the CTE based on that. I then reorganized the query to CTE format and created a FROM statement at the end of the query with the CTE beginning statement.

```sql
WITH top_customer_cte AS
(SELECT SUM(A.amount) AS total_paid_amount,
        B.last_name,
        B.first_name,
        E.country,
        D.city,
        B.customer_id
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 city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei',
              'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')
GROUP BY country, city, first_name, last_name, B.customer_id
ORDER BY total_paid_amount DESC
LIMIT 5),
all_customer_count_cte AS
(SELECT D.country,
 COUNT (DISTINCT customer_id) AS all_customer_count,
 COUNT (DISTINCT country) 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
```

```sql
 GROUP BY D.country)
 SELECT D.country,
 COUNT (DISTINCT A.customer_id) AS all_customer_count,
 COUNT (DISTINCT top_customer_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_cte ON D.country = top_customer_cte.country
GROUP BY D.country
ORDER BY all_customer_count DESC
LIMIT 5
```

	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

I took the subquery from the previous exercise and entered it into pgadmin. I then began to create a WITH statement utilizing the main statement of the previous query which was the top 5 customer count in this case I named it top customer count CTE. I then reorganized the query to CTE format and created a FROM statement at the end of the query with the CTE beginning statement. I created a second CTE statement because the previous query was trying to find two counts. The second CTE was named after the count of all customers. I then formatted the query as a CTE would be formatted. I then created a left join to join the CTE to the table. Ten I ordered by the second CTE to produce the necessary results.

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

1. Which approach do you think will perform better and why?

   I believe CTE's are more readable but both the subquery and the CTE handling producing he results easily.

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 each query has been run, a pop-up window will display its speed in milliseconds.
4. Did the results surprise you? Write a few sentences to explain your answer.

Step 3:

Write 1 to 2 paragraphs on the challenges you faced when replacing your subqueries with CTEs.

	QUERY PLAN text
1	Aggregate (cost=61.33..61.34 rows=1 width=32)
2	-> Limit (cost=61.26..61.27 rows=5 width=67)
3	-> Sort (cost=61.26..61.80 rows=219 width=67)
4	Sort Key: (sum(a.amount)) DESC
5	-> HashAggregate (cost=54.88..57.62 rows=219 width=67)
6	Group Key: e.country, d.city, b.customer_id

Figure 1 First CTE

	QUERY PLAN text
1	Aggregate (cost=61.33..61.34 rows=1 width=32)
2	-> Limit (cost=61.26..61.27 rows=5 width=67)
3	-> Sort (cost=61.26..61.80 rows=219 width=67)
4	Sort Key: (sum(a.amount)) DESC
5	-> HashAggregate (cost=54.88..57.62 rows=219 width=67)
6	Group Key: e.country, d.city, b.customer_id

Figure 2 First Subquery

	QUERY PLAN text
1	Limit (cost=163.11..163.12 rows=5 width=25)
2	-> Sort (cost=163.11..163.38 rows=109 width=25)
3	Sort Key: (count(DISTINCT a.customer_id)) DESC
4	-> GroupAggregate (cost=152.31..161.30 rows=109 width=25)
5	Group Key: d.country
6	-> Merge Left Join (cost=152.31..155.71 rows=599 width=17)

Figure 3 2nd CTE

	QUERY PLAN text
1	Limit (cost=164.20..164.22 rows=5 width=25)
2	-> Sort (cost=164.20..164.48 rows=109 width=25)
3	Sort Key: (count(DISTINCT top_5_customers.*)) DESC
4	-> GroupAggregate (cost=153.41..162.39 rows=109 width=25)
5	Group Key: d.country
6	-> Merge Left Join (cost=153.41..156.81 rows=599 width=70)

Figure 4 2nd Subquery

The times varied for each execution of the CTEs and the Subqueries. I was not surprised by this. The queries and subqueries differed by less than 5 seconds.

This task was challenging. The first query was manageable as I followed the example in the notes given. The second query however, was difficult as it required a lot more formatting and two CTEs were necessary. Also while in the first one the INNER JOIN part of the query was not repeated. It was repeated in the second one which threw me off.