Step 1:

Query 1

```
WITH top_5_customers

AS (SELECT B.customer_id, B.first_name, B.last_name, E.country, D.city, SUM(amount)

AS Total_amount_paid

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

GROUP BY B.customer_id, B.first_name, B.last_name, E.country, D.city

HAVING D.city IN('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)', 'Kurashiki',

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

ORDER BY SUM(amount) DESC  LIMIT 5)

SELECT AVG(Total_amount_paid) AS average_amount_paid_by_the_top_5_customers

FROM top_5_customers
```

Average_amount_paid_by_the_top_5_customers
107.3540000000000000

Query 2

```
WITH Top_customers

AS (SELECT B.customer_id, B.first_name, B.last_name, E.country, D.city, SUM(amount)

AS Total_amount_paid

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

GROUP BY B.customer_id, B.first_name, B.last_name, E.country, D.city

HAVING D.city IN('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulia)', 'Kurashiki',

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

ORDER BY SUM(amount) DESC LIMIT 5)

SELECT E.country, COUNT(DISTINCT A.customer_id) AS All_customer_count,
COUNT(DISTINCT Top_customers) AS Top_customer_count

FROM customer 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

LEFT JOIN Top_customers ON E.country = Top_customers.country

GROUP BY E.country

ORDER BY Top_customer_count DESC
```

Country	All_customer_count	Top_customer_count
Mexico	30	2
U.S.	36	1
Turkey	15	1
India	60	1
Etc.	Etc.	Etc.

Explanations

The first query was very simple. I started with turning my subquery into a WITH AS statement, and then I selected the value I wanted from the CTE I made.

The second query, having two subqueries instead of one, was way more difficult for me to figure out. I had no idea that the second subquery was supposed to be a part of the SELECT statement after the CTE. So I was trying to put both subqueries into the CTE, and that caused a lot of problems. I tried looking at the student examples to help me discern what I was supposed to do, but it ended up making me more confused because I wrote my first subquery with a HAVING statement that none of the student examples had. After you clarified that I was trying to give multiple aliases to the CTE, that's when the pieces started to fall into place. I kept my first subquery a part of the CTE, like in the first query of this task. Then I put the second subquery into the SELECT statement. And then finally ended it with the LEFT JOIN statement to combine the subqueries. There was one last hurdle though. I didn't know that I needed to add the DISTINCT clause to the two aggregate functions I was searching for. After realizing that the query was **FINALLY** done.

Step 2:

My gut is telling me that If you have one subquery a CTE will be more efficient, while if you have two subqueries a CTE will be less efficient.

Task 8 Query 1

Cost = 64.45..64.46

Time = 86 msec

Task 8 Query 2

Cost = 136.53..136.54

Time = 104 msec

Task 9 Query 1

Cost = 64.45..64.46

Time = 82 msec

Task 9 Query 2

Cost = 192.78..193.06

Time = 92 msec

The results didn't surprise me. The first query was very similar with and without a CTE. While the query with the CTE was faster it is important to note that the speed of a query isn't consistent. You can see this by running the query multiple times in a row. Also, like how I predicted the second query costed a lot more with the CTE.

Step 3:

I’ve already explained my struggles with converting subqueries into CTE’s. If you’re dealing with one subquery, the process is very easy. But if you have multiple subqueries, the process is a lot more complex and difficult to keep track of.