Answers 3.9

Query | Query History

```
WITH avg_amount_cte (amount, customer_id, first_name, last_name, city, country) AS
(SELECT SUM(A.amount) AS total_amount_paid,
       A.customer_id,
       B.first_name,
       B.last_name,
       D.city,
       E.country
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('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'So Leopoldo', 'Teboksary', 'Tianjin'
GROUP BY A.amount, A.customer_id, B.first_name, B.last_name, D.city, E.country
ORDER BY total_amount_paid DESC
LIMIT 5)
SELECT AVG(amount) AS average
FROM avg_amount_cte

```

Data Output | Messages | Notifications

	average numeric
1	41.3220000000000000

```sql
WITH top_customer_cte (amount, customer_id, first_name, last_name, city, country) AS
(SELECT SUM(A.amount) AS total_amount_paid,
       B.customer_id,
       B.first_name,
       B.last_name,
       D.city,
       E.country
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 D.city
                FROM customer B
                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 E.country IN (SELECT E.country
                                   FROM customer B
                                   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 E.country
                                  ORDER BY COUNT(B.customer_id) DESC
                                  LIMIT 10)
               GROUP BY E.country,
                        D.city
               ORDER BY COUNT(B.customer_id) DESC
               LIMIT 10)
GROUP BY B.customer_id,
         B.first_name,
         B.last_name,
         D.city,
         E.country
ORDER BY SUM(A.amount) DESC
LIMIT 5)
SELECT E.country,
       COUNT(DISTINCT B.customer_id) AS all_customer_count,
       COUNT(DISTINCT top_customer_cte) AS top_customer_count
FROM customer B
JOIN address C ON B.address_id=C.address_id
JOIN city D ON C.city_id=D.city_id
JOIN country E ON D.country_id=E.country_id
LEFT JOIN top_customer_cte ON B.customer_id=top_customer_cte.customer_id
GROUP BY E.country
ORDER BY all_customer_count DESC
LIMIT 10
```

Data Output | Messages | Explain | Notifications

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

Explanation of Approach

- First I went back and copied the original subqueries. From there, I isolated the subquery and cut out the rest of the query. Next, I put in the WITH function and named the CTE in the top line. I kept referencing the previous subqueries to ensure I contained the necessary information.

Performance Comparison

- First one: the subquery took 77 milliseconds while the CTE took 54 milliseconds.
- Second one: the subquery took 85 milliseconds while the CTE took 95 milliseconds.
- These results were a bit surprising- at least, for the second query, as the subquery took less time than the CTE. However, we are dealing with much less data compared to databases at medium-to-large companies; in those cases, the differences would be magnified by orders of magnitude. In other words, I don't think the subquery versus CTE makes much of a difference in the Rockbuster database, but in the real world I think the advantages of the CTE would be much more apparent.

Challenges Faced When Replacing Subqueries with CTEs

The first subquery was easier for me to replace with a CTE. It was far less involved, meaning there were a lot fewer lines, so it wasn't as difficult to isolate the nested query and then rewrite the beginning and end with the correct CTE syntax.

However, the second query posed some issues. There were a lot of JOINs and more columns being pulled, so it took more time to read through line by line and ensure I kept the necessary information. I initially met some errors when trying to run the new CTE as I did not include JOINs that came outside the nested query, and I had to add those back in. Once I had the correct aliases as well, I was ready to run the query, and that time it came back correctly.