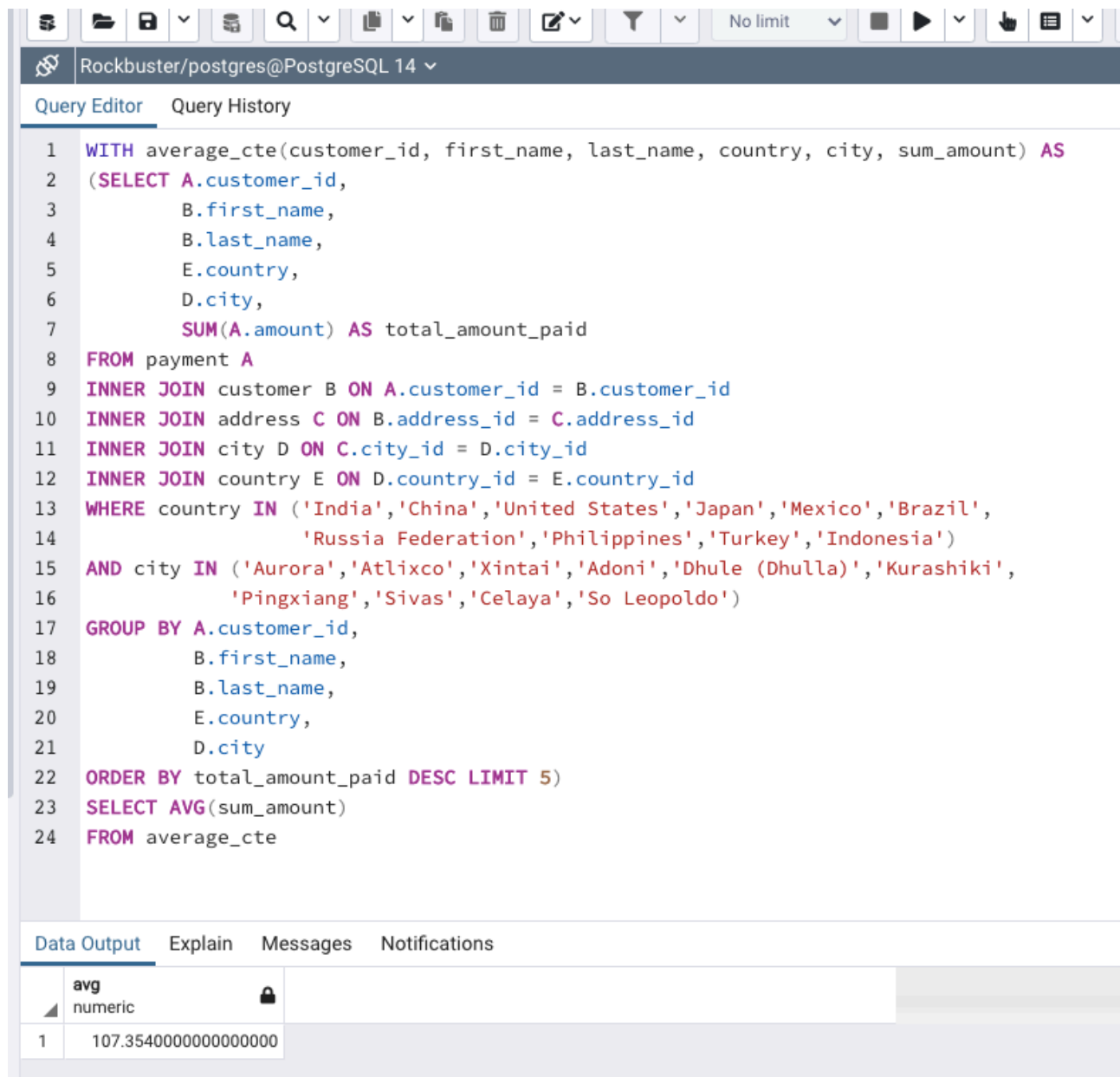


3.9 Common Table Expressions

Step 1

1. Rewrite your queries from steps 1 and 2 of task 3.8 as CTEs.
2. Copy-paste your CTEs and their outputs into your answers document.

3.8 Step 1 rewritten



The screenshot shows a PostgreSQL query editor interface. The top toolbar includes icons for file operations, search, and execution. The main area displays a SQL query using a Common Table Expression (CTE) to calculate the average amount paid by customers in specific countries and cities. The query is as follows:

```
1 WITH average_cte(customer_id, first_name, last_name, country, city, sum_amount) AS
2   (SELECT A.customer_id,
3          B.first_name,
4          B.last_name,
5          E.country,
6          D.city,
7          SUM(A.amount) AS total_amount_paid
8   FROM payment A
9   INNER JOIN customer B ON A.customer_id = B.customer_id
10  INNER JOIN address C ON B.address_id = C.address_id
11  INNER JOIN city D ON C.city_id = D.city_id
12  INNER JOIN country E ON D.country_id = E.country_id
13  WHERE country IN ('India','China','United States','Japan','Mexico','Brazil',
14                  'Russia Federation','Philippines','Turkey','Indonesia')
15  AND city IN ('Aurora','Atlixco','Xintai','Adoni','Dhule (Dhulla)','Kurashiki',
16              'Pingxiang','Sivas','Celaya','So Leopoldo')
17  GROUP BY A.customer_id,
18           B.first_name,
19           B.last_name,
20           E.country,
21           D.city
22  ORDER BY total_amount_paid DESC LIMIT 5)
23  SELECT AVG(sum_amount)
24  FROM average_cte
```

Below the query editor, the 'Data Output' tab is active, showing the result of the query. The output is a single row with the average amount paid, formatted as a numeric value.

	avg numeric
1	107.3540000000000000

3.8 Step 2 rewritten

The screenshot shows the Rockbuster PostgreSQL 14 Query Editor. The query is as follows:

```
1 WITH top_5_cte(customer_id, first_name, last_name, country, city, sum_amount) AS
2 (SELECT A.customer_id,
3      B.first_name,
4      B.last_name,
5      E.country,
6      D.city,
7      SUM(A.amount) AS total_amount_paid
8 FROM payment A
9 INNER JOIN customer B ON A.customer_id = B.customer_id
10 INNER JOIN address C ON B.address_id = C.address_id
11 INNER JOIN city D ON C.city_id = D.city_id
12 INNER JOIN country E ON D.country_id = E.country_id
13 WHERE country IN ('India','China','United States','Japan','Mexico','Brazil',
14                  'Russia Federation','Philippines','Turkey','Indonesia')
15 AND city IN ('Aurora','Atlixco','Xintai','Adoni','Dhule (Dhulla)','Kurashiki',
16              'Pingxiang','Sivas','Celaya','So Leopoldo')
17 GROUP BY A.customer_id,
18          B.first_name,
19          B.last_name,
20          E.country,
21          D.city
22 ORDER BY total_amount_paid DESC LIMIT 5)
23 SELECT country.country,
24        COUNT(DISTINCT customer.customer_id) AS all_customer_count,
25        COUNT(DISTINCT country.country) AS top_customer_count
26 FROM top_5_cte
27 LEFT JOIN customer ON customer.customer_id = customer.customer_id
28 LEFT JOIN address ON customer.address_id = address.address_id
29 LEFT JOIN city ON address.city_id = city.city_id
30 LEFT JOIN country ON city.country_id = country.country_id
31 GROUP BY country.country
32 ORDER BY all_customer_count DESC LIMIT 5;
```

The Data Output tab shows the following results:

	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

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

I define the CTE with the 'WITH' clause and gave it an appropriate expression name. Then, I listed columns that will be listed in the CTE definition and used the AS keyword. I copied the query from previous exercise as then used SELECT again to display the result.

Step 2

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

In general, I was of the opinion that CTE will perform better due to its readability and the fact that it is only defined once. However the result of 'EXPLAIN' keyword indicated that that for this query both CTE and Subquery have the same cost.

CTE Step 1

The screenshot shows the PGAdmin interface with a SQL query in the Query Editor. The query uses a Common Table Expression (CTE) named 'average_cte' to calculate the average amount paid per customer. The query is as follows:

```
1 EXPLAIN WITH average_cte(customer_id, first_name, last_name, country, city, sum_amount) AS
2 (SELECT A.customer_id,
3    B.first_name,
4    B.last_name,
5    E.country,
6    D.city,
7    SUM(A.amount) AS total_amount_paid
8 FROM payment A
9 INNER JOIN customer B ON A.customer_id = B.customer_id
10 INNER JOIN address C ON B.address_id = C.address_id
11 INNER JOIN city D ON C.city_id = D.city_id
12 INNER JOIN country E ON D.country_id = E.country_id
13 WHERE country IN ('India','China','United States','Japan','Mexico','Brazil',
14                  'Russia Federation','Philippines','Turkey','Indonesia')
15 AND city IN ('Aurora','Atlixco','Xintai','Adoni','Dhule (Dhulla)','Kurashiki',
16              'Pingxiang','Sivas','Celaya','So Leopoldo')
17 GROUP BY A.customer_id)
```

The Query Plan shows the execution strategy, including nested loops and hash joins. The final status message indicates: "Successfully run. Total query runtime: 51 msec. 22 rows affected."

Subquery Step 1

The screenshot shows the PGAdmin interface with a SQL query in the Query Editor. The query uses a subquery to calculate the average amount paid per customer. The query is as follows:

```
1 EXPLAIN SELECT AVG(total_amount_paid.total_amount_paid) AS average
2 FROM
3 (SELECT A.customer_id,
4    B.first_name,
5    B.last_name,
6    E.country,
7    D.city,
8    SUM(A.amount) AS total_amount_paid
9 FROM payment A
10 INNER JOIN customer B ON A.customer_id = B.customer_id
11 INNER JOIN address C ON B.address_id = C.address_id
12 INNER JOIN city D ON C.city_id = D.city_id
13 INNER JOIN country E ON D.country_id = E.country_id
14 WHERE country IN ('India','China','United States','Japan','Mexico','Brazil',
15                  'Russia Federation','Philippines','Turkey','Indonesia')
16 AND city IN ('Aurora','Atlixco','Xintai','Adoni','Dhule (Dhulla)','Kurashiki',
17              'Pingxiang','Sivas','Celaya','So Leopoldo'))
```

The Query Plan shows the execution strategy, including nested loops and hash joins. The final status message indicates: "Successfully run. Total query runtime: 41 msec. 22 rows affected."

CTE Step 2

The screenshot shows the pgAdmin interface with a SQL query in the Query Editor. The query uses a Common Table Expression (CTE) named `top_5_cte` to calculate the top 5 customers by total amount paid in each country. The query is as follows:

```
1 EXPLAIN WITH top_5_cte(customer_id, first_name, last_name, country, city, sum_amount) AS
2 (SELECT A.customer_id,
3    B.first_name,
4    B.last_name,
5    E.country,
6    D.city,
7    SUM(A.amount) AS total_amount_paid
8 FROM payment A
9 INNER JOIN customer B ON A.customer_id = B.customer_id
10 INNER JOIN address C ON B.address_id = C.address_id
11 INNER JOIN city D ON C.city_id = D.city_id
12 INNER JOIN country E ON D.country_id = E.country_id
13 WHERE country IN ('India','China','United States','Japan','Mexico','Brazil',
14                  'Russia Federation','Philippines','Turkey','Indonesia')
15 AND city IN ('Aurora','Atlixco','Xintai','Adoni','Dhule (Dhulla)','Kurashiki',
16              'Pingxiang','Sivas','Celaya','So Leopoldo')
17 GROUP BY A.customer_id,
```

The Query Plan shows the execution steps, including the CTE calculation and the final aggregation. The status bar indicates: "Successfully run. Total query runtime: 52 msec. 44 rows affected."

Subquery Step 2

The screenshot shows the pgAdmin interface with a SQL query in the Query Editor. The query uses a subquery to calculate the top 5 customers by total amount paid in each country. The query is as follows:

```
1 EXPLAIN SELECT country.country,
2    COUNT(DISTINCT customer.customer_id) AS all_customer_count,
3    COUNT(DISTINCT country.country) AS top_customer_count
4 FROM
5 (SELECT A.customer_id,
6    B.first_name,
7    B.last_name,
8    E.country,
9    D.city,
10   SUM(A.amount) AS total_amount_paid
11 FROM payment A
12 INNER JOIN customer B ON A.customer_id = B.customer_id
13 INNER JOIN address C ON B.address_id = C.address_id
14 INNER JOIN city D ON C.city_id = D.city_id
15 INNER JOIN country E ON D.country_id = E.country_id
16 WHERE country IN ('India','China','United States','Japan','Mexico','Brazil',
17                  'Russia Federation','Philippines','Turkey','Indonesia')
```

The Query Plan shows the execution steps, including the subquery calculation and the final aggregation. The status bar indicates: "Successfully run. Total query runtime: 56 msec. 44 rows affected."

Step 3

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

The challenges I faced include determine what is needed to build the CTE. I am aware that unlike subqueries, the CTE are defined at the start of query. Also, writing a totally new SELECT statement querying the temporary table created with CTE seems strange initially. However, after spending more time to read the syntax everything was quite easier to understand.

