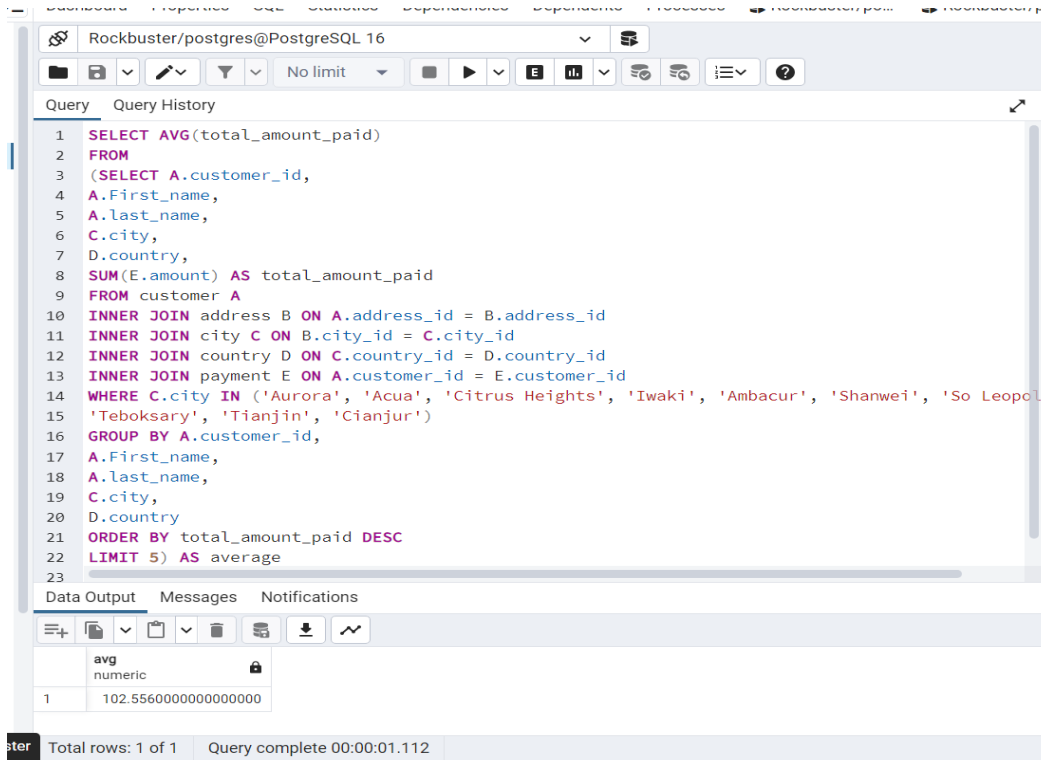


Task 3.8 Performing Subqueries

Step 1.



```

1  SELECT AVG(total_amount_paid)
2  FROM
3  (SELECT A.customer_id,
4   A.First_name,
5   A.last_name,
6   C.city,
7   D.country,
8   SUM(E.amount) AS total_amount_paid
9   FROM customer A
10  INNER JOIN address B ON A.address_id = B.address_id
11  INNER JOIN city C ON B.city_id = C.city_id
12  INNER JOIN country D ON C.country_id = D.country_id
13  INNER JOIN payment E ON A.customer_id = E.customer_id
14  WHERE C.city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambacur', 'Shanwei', 'So Leopold',
15  'Teboksary', 'Tianjin', 'Cianjur')
16  GROUP BY A.customer_id,
17  A.First_name,
18  A.last_name,
19  C.city,
20  D.country
21  ORDER BY total_amount_paid DESC
22  LIMIT 5) AS average
23

```

	avg
1	102.55600000000000

Total rows: 1 of 1 Query complete 00:00:01.112

Step 2.

```

SELECT
D.country,
COUNT(DISTINCT A.customer_id) AS all_customer_count,
COUNT(top_5_customers) 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
(SELECT
A.customer_id,
A.first_name,
A.last_name,
D.country,
C.city,
SUM(E.amount) AS total_amount_paid
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
INNER JOIN payment E ON A.customer_id = E.customer_id
WHERE C.city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambacur', 'Shanwei', 'So
Leopoldo',
'Teboksary', 'Tianjin', 'Cianjur')
GROUP BY A.customer_id,
A.First_name,
A.last_name,
C.city,
D.country
ORDER BY total_amount_paid DESC
LIMIT 5) top_5_customers ON A.customer_id = top_5_customers.customer_id
GROUP BY D.country
ORDER BY all_customer_count DESC
LIMIT 5;

```

The screenshot shows a PostgreSQL query editor interface. The top bar indicates the connection is 'Rockbuster/postgres@PostgreSQL 16'. Below the toolbar, the 'Query' tab is active, displaying a SQL query. The query is a complex join and aggregation statement. The 'Data Output' tab is also visible, showing the results of the query in a table format. The table has four columns: 'country', 'all_customer_count', and 'top_customer_count'. The results show the top 5 countries by total amount paid, ordered from highest to lowest total amount paid.

Query:

```

1 SELECT
2   D.country,
3   COUNT(DISTINCT A.customer_id) AS all_customer_count,
4   COUNT(top_5_customers) AS top_customer_count
5 FROM customer A
6 INNER JOIN address B ON A.address_id = B.address_id
7 INNER JOIN city C ON B.city_id = C.city_id
8 INNER JOIN country D ON C.country_id = D.country_id
9 LEFT JOIN
10  (SELECT
11    A.customer_id,
12    A.first_name,
13    A.last_name,
14    D.country,
15    C.city,
16    SUM(E.amount) AS total_amount_paid
17 FROM customer A
18 INNER JOIN address B ON A.address_id = B.address_id
19 INNER JOIN city C ON B.city_id = C.city_id

```

Data Output:

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	India	60	0
2	China	53	1
3	United States	36	2
4	Japan	31	1
5	Mexico	30	1

Total rows: 5 of 5 Query complete 00:00:00.949

Step 3.

Well, I think step 1 and 2 should have been done without using subqueries because we've been able to identify top 10 cities in the last task 3.7. Though subqueries are very useful in complex queries with multiple joins. I think subqueries are very useful where a query depends on the results of another query. Using only JOINS could have been less complex.