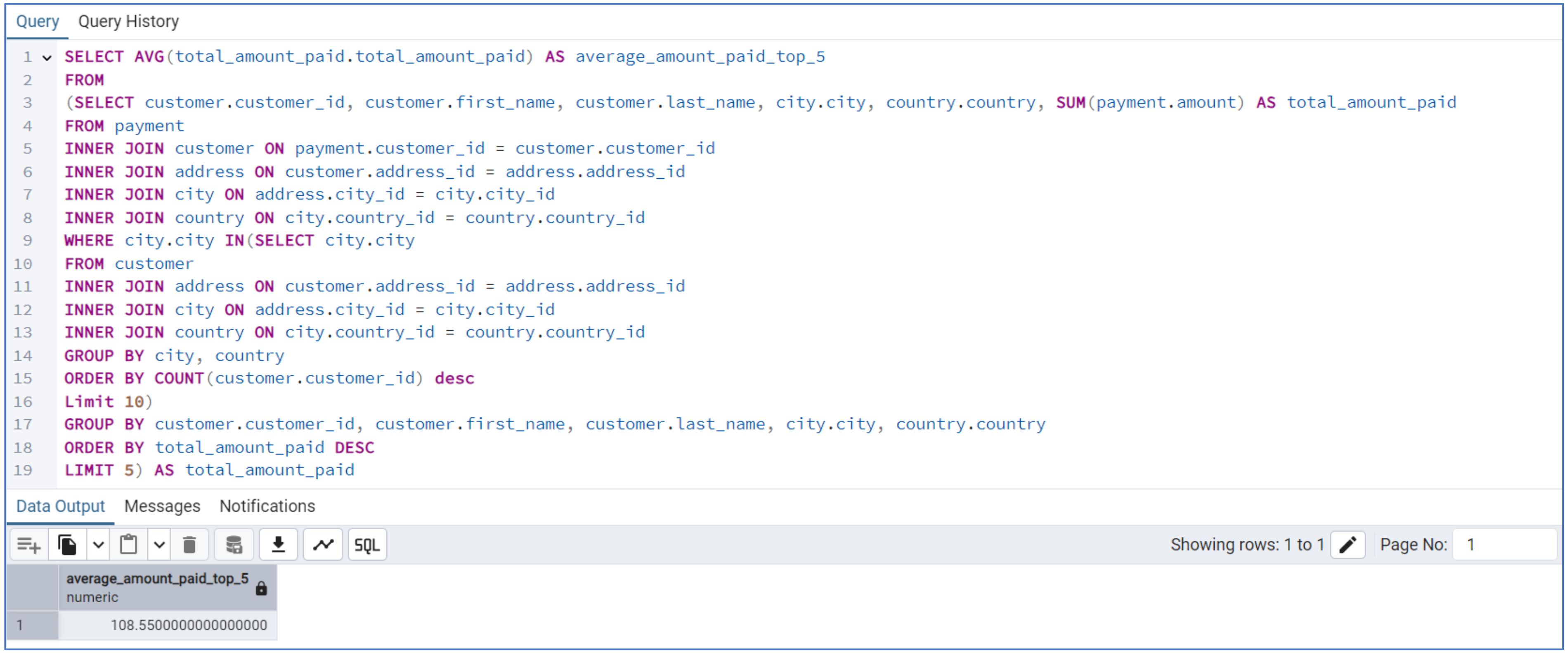
Answers 3.8

STEP 1:



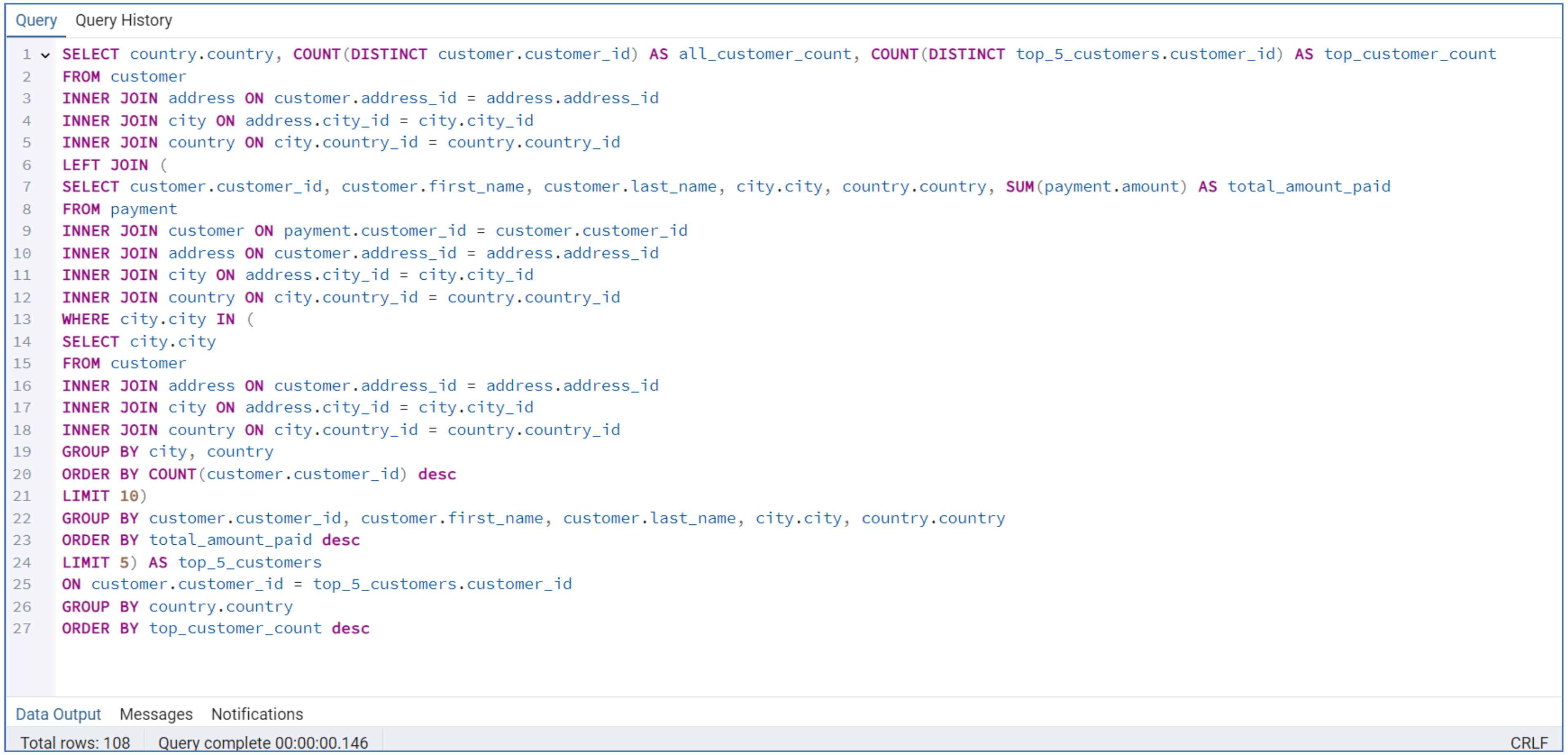
```
SELECT AVG(total_amount_paid.total_amount_paid) AS average_amount_paid_top_5
FROM
(SELECT customer.customer_id, customer.first_name, customer.last_name, city.city, country.country, SUM(payment.amount) AS total_amount_paid
FROM payment
INNER JOIN customer ON payment.customer_id = customer.customer_id
INNER JOIN address ON customer.address_id = address.address_id
INNER JOIN city ON address.city_id = city.city_id
INNER JOIN country ON city.country_id = country.country_id
WHERE city.city IN(SELECT city.city
FROM customer
INNER JOIN address ON customer.address_id = address.address_id
INNER JOIN city ON address.city_id = city.city_id
INNER JOIN country ON city.country_id = country.country_id
GROUP BY city, country
ORDER BY COUNT(customer.customer_id) desc
Limit 10)
GROUP BY customer.customer_id, customer.first_name, customer.last_name, city.city, country.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS total_amount_paid
```

	average_amount_paid_top_5 numeric
1	108.5500000000000000

CODE:

```
SELECT AVG(total_amount_paid.total_amount_paid) AS average
FROM
(SELECT customer.customer_id, customer.first_name, customer.last_name, city.city, country.country,
SUM(payment.amount) AS total_amount_paid
FROM payment
INNER JOIN customer ON payment.customer_id = customer.customer_id
INNER JOIN address ON customer.address_id = address.address_id
INNER JOIN city ON address.city_id = city.city_id
INNER JOIN country ON city.country_id = country.country_id
WHERE city.city IN
(SELECT city.city
FROM customer
INNER JOIN address ON customer.address_id = address.address_id
INNER JOIN city ON address.city_id = city.city_id
INNER JOIN country ON city.country_id = country.country_id
GROUP BY city, country
ORDER BY COUNT(customer.customer_id) DESC
LIMIT 10)
GROUP BY customer.customer_id, customer.first_name, customer.last_name, city.city,
country.country
ORDER BY total_amount_paid DESC
LIMIT 5) AS total_amount_paid
```

STEP 2:

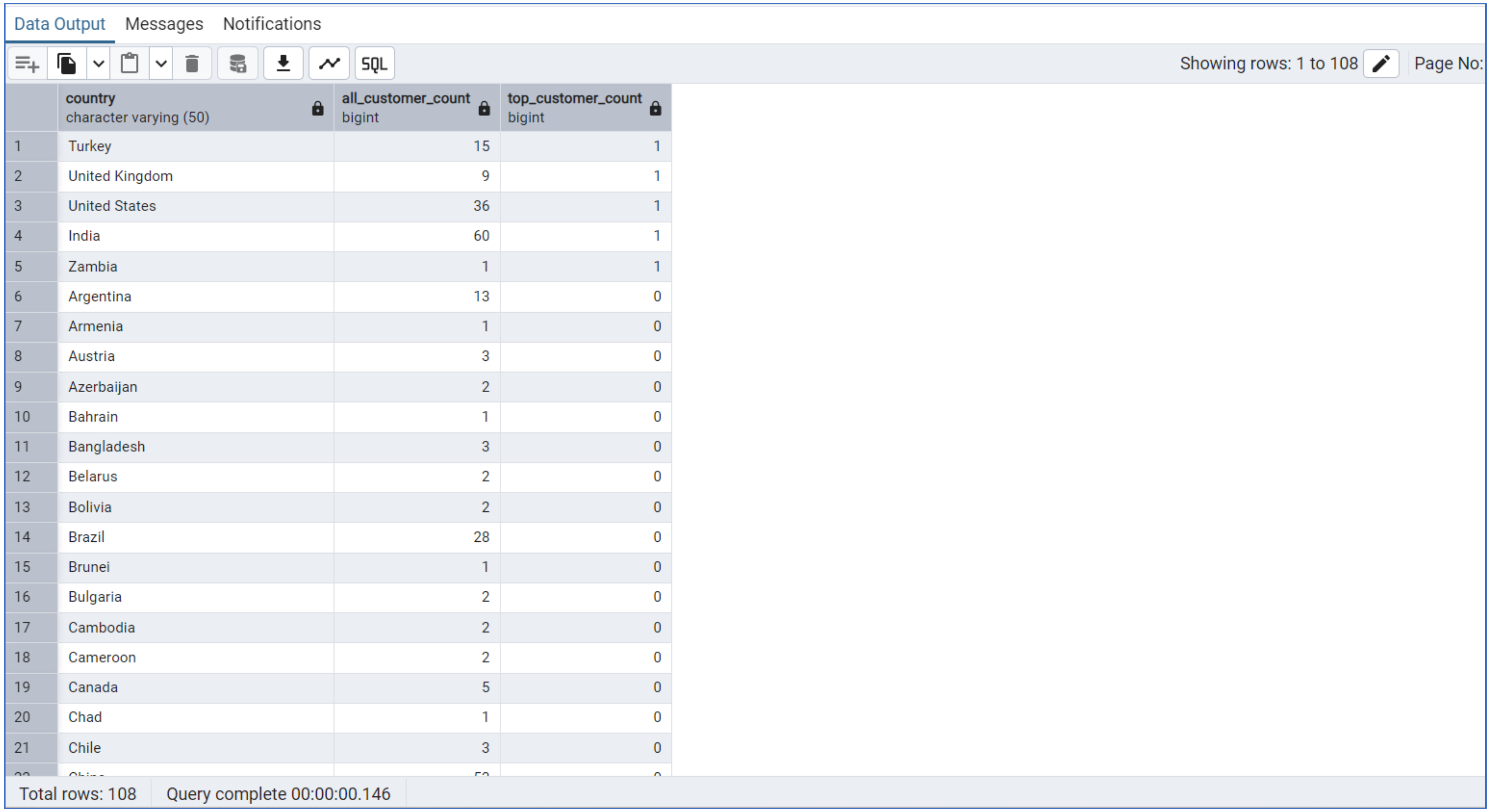


Query Query History

```
SELECT country.country, COUNT(DISTINCT customer.customer_id) AS all_customer_count, COUNT(DISTINCT top_5_customers.customer_id) AS top_customer_count
FROM customer
INNER JOIN address ON customer.address_id = address.address_id
INNER JOIN city ON address.city_id = city.city_id
INNER JOIN country ON city.country_id = country.country_id
LEFT JOIN (
SELECT customer.customer_id, customer.first_name, customer.last_name, city.city, country.country, SUM(payment.amount) AS total_amount_paid
FROM payment
INNER JOIN customer ON payment.customer_id = customer.customer_id
INNER JOIN address ON customer.address_id = address.address_id
INNER JOIN city ON address.city_id = city.city_id
INNER JOIN country ON city.country_id = country.country_id
WHERE city.city IN (
SELECT city.city
FROM customer
INNER JOIN address ON customer.address_id = address.address_id
INNER JOIN city ON address.city_id = city.city_id
INNER JOIN country ON city.country_id = country.country_id
GROUP BY city, country
ORDER BY COUNT(customer.customer_id) desc
LIMIT 10)
GROUP BY customer.customer_id, customer.first_name, customer.last_name, city.city, country.country
ORDER BY total_amount_paid desc
LIMIT 5) AS top_5_customers
ON customer.customer_id = top_5_customers.customer_id
GROUP BY country.country
ORDER BY top_customer_count desc
```

Data Output Messages Notifications

Total rows: 108 Query complete 00:00:00.146 CRLF



Showing rows: 1 to 108

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	Turkey	15	1
2	United Kingdom	9	1
3	United States	36	1
4	India	60	1
5	Zambia	1	1
6	Argentina	13	0
7	Armenia	1	0
8	Austria	3	0
9	Azerbaijan	2	0
10	Bahrain	1	0
11	Bangladesh	3	0
12	Belarus	2	0
13	Bolivia	2	0
14	Brazil	28	0
15	Brunei	1	0
16	Bulgaria	2	0
17	Cambodia	2	0
18	Cameroon	2	0
19	Canada	5	0
20	Chad	1	0
21	Chile	3	0
22	China	53	0

Total rows: 108 Query complete 00:00:00.146

CODE:

```
SELECT country.country, COUNT(DISTINCT customer.customer_id) AS all_customer_count,
COUNT(DISTINCT top_5_customers.customer_id) AS top_customer_count

FROM customer

INNER JOIN address ON customer.address_id = address.address_id

INNER JOIN city ON address.city_id = city.city_id

INNER JOIN country ON city.country_id = country.country_id

LEFT JOIN (

SELECT customer.customer_id, customer.first_name, customer.last_name, city.city, country.country,
SUM(payment.amount) AS total_amount_paid

FROM payment

INNER JOIN customer ON payment.customer_id = customer.customer_id

INNER JOIN address ON customer.address_id = address.address_id

INNER JOIN city ON address.city_id = city.city_id

INNER JOIN country ON city.country_id = country.country_id

WHERE city.city IN (

SELECT city.city

FROM customer

INNER JOIN address ON customer.address_id = address.address_id

INNER JOIN city ON address.city_id = city.city_id

INNER JOIN country ON city.country_id = country.country_id

GROUP BY city, country

ORDER BY COUNT(customer.customer_id) desc

LIMIT 10)

GROUP BY customer.customer_id, customer.first_name, customer.last_name, city.city,
country.country

ORDER BY total_amount_paid desc

LIMIT 5) AS top_5_customers

ON customer.customer_id = top_5_customers.customer_id

GROUP BY country.country

ORDER BY top_customer_count desc
```

STEP 3:

- **Do you think steps 1 and 2 could be done without using subqueries?**

The queries from step 1 and step 2 rely heavily on subqueries to filter and aggregate data before performing calculations on the results. While it might be possible to rewrite them using JOINs functions instead of subqueries, doing so could make the queries more complex and harder to read.

- **When do you think subqueries are useful?**

Subqueries are useful here because they allow us to break down the problem into smaller, more manageable parts, first identifying the top paying customers, then calculating the average, and finally analyzing their distribution across countries.

Subqueries are particularly useful when we need to filter or aggregate data before using it in a main query. They help improve readability, especially when dealing with multistep calculations like ranking customers based on payments.