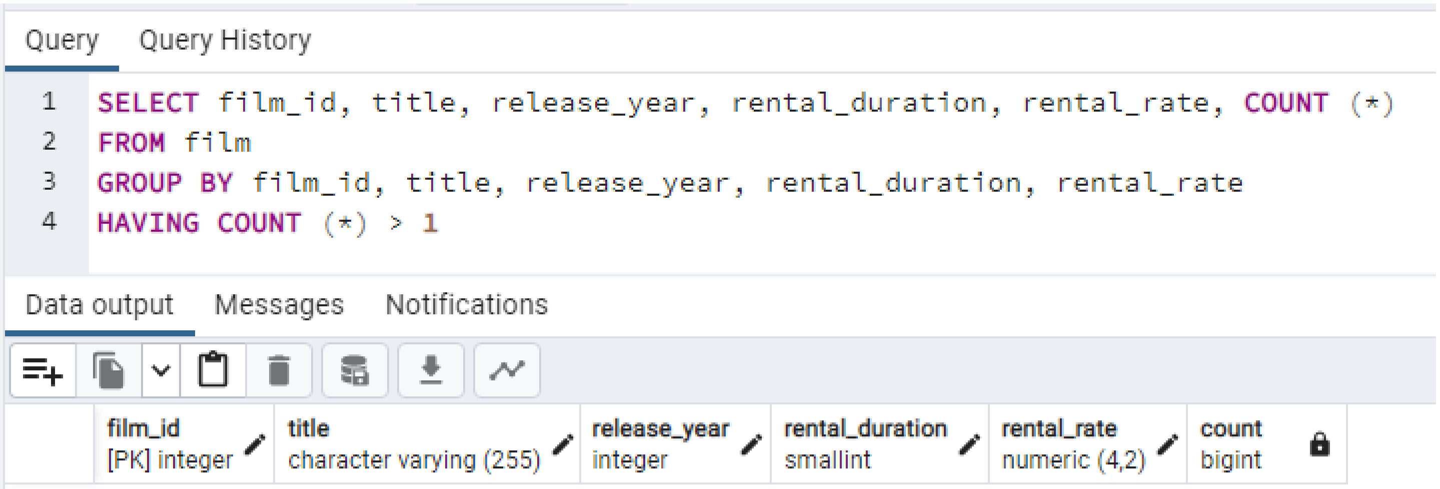
Scripts Used to Answer Business Questions

#1

Check for and clean dirty data: Find out if the film table and the customer table contain any dirty data, specifically non-uniform or duplicate data, or missing values. Next to each query write 2 to 3 sentences explaining how you would clean the data (even if the data is not dirty).

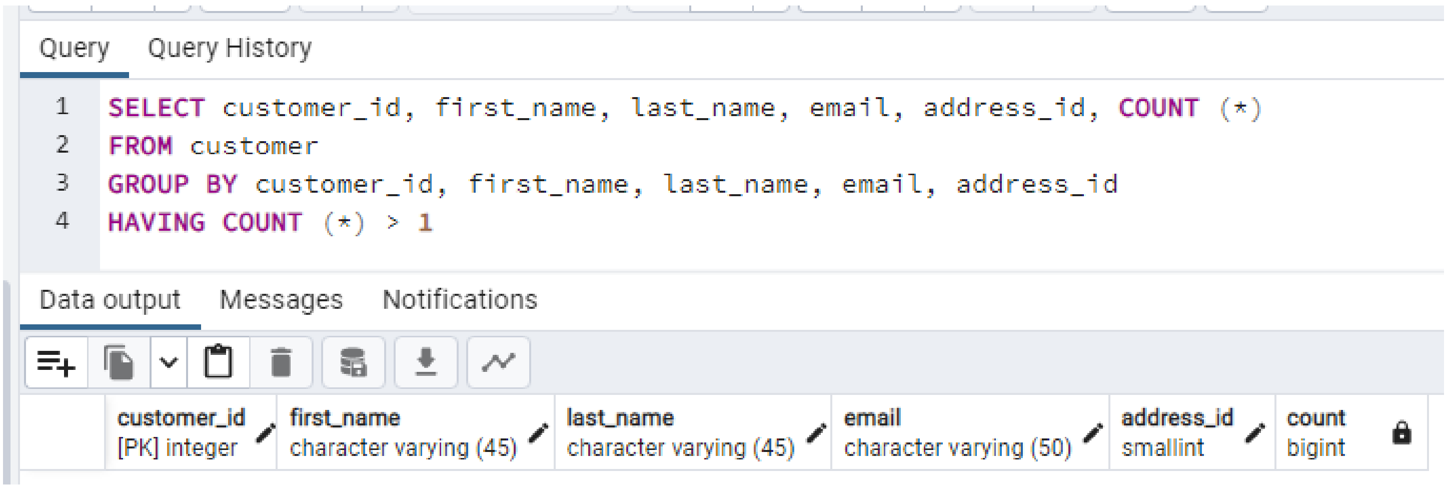
Duplicates from Film table:



```
SELECT film_id, title, release_year, rental_duration, rental_rate, COUNT (*)
FROM film
GROUP BY film_id, title, release_year, rental_duration, rental_rate
HAVING COUNT (*) > 1
```

film_id [PK] integer	title character varying (255)	release_year integer	rental_duration smallint	rental_rate numeric (4,2)	count bigint

Duplicates from Customer table:

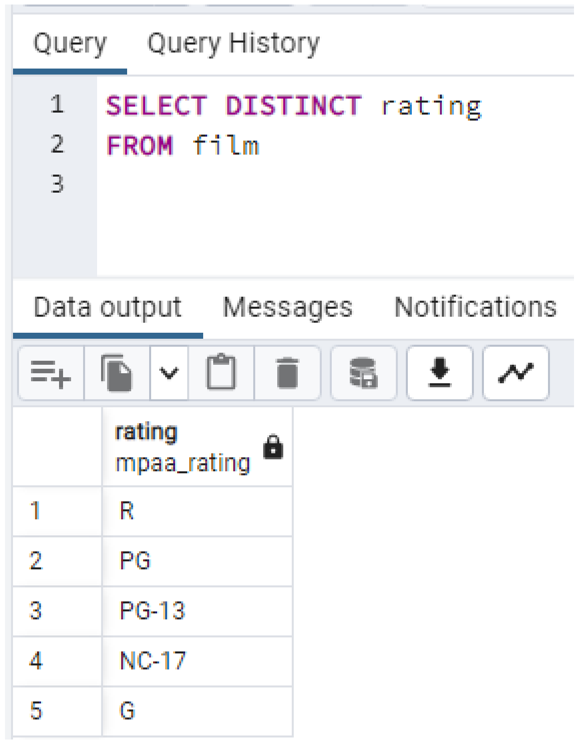


```
SELECT customer_id, first_name, last_name, email, address_id, COUNT (*)
FROM customer
GROUP BY customer_id, first_name, last_name, email, address_id
HAVING COUNT (*) > 1
```

customer_id [PK] integer	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint	count bigint

*If there were duplicates I would either create a view to only include unique records using the SELECT DISTINCT statement, or if I have the ability to delete duplicates I would do so by using the DELETE statement.

Non-uniform data from Film table:



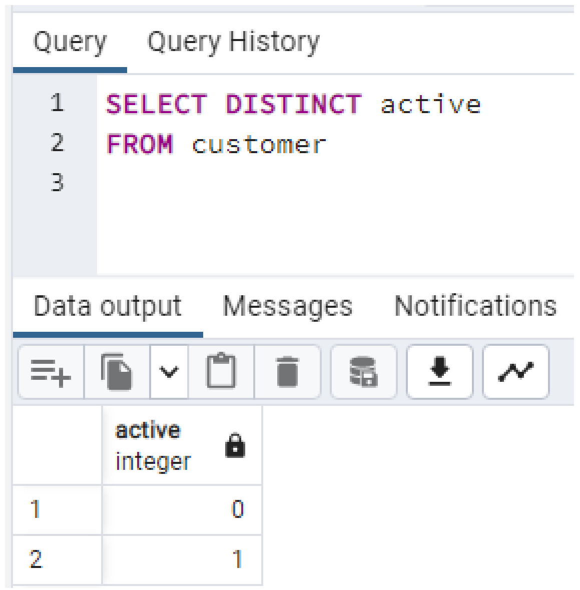
Query | Query History

```
SELECT DISTINCT rating
FROM film
```

Data output | Messages | Notifications

	rating mpaa_rating
1	R
2	PG
3	PG-13
4	NC-17
5	G

Non-uniform data from Customer table:



Query | Query History

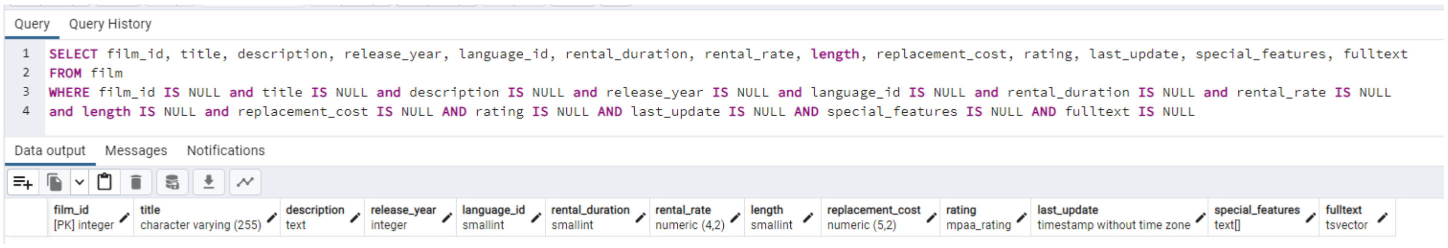
```
SELECT DISTINCT active
FROM customer
```

Data output | Messages | Notifications

	active integer
1	0
2	1

*This process has to be repeated for each column and the results analyzed to decide if the data entered is uniform. In the examples shown, the data is uniform. If this were not the case, I would use the UPDATE statement to make the data uniform.

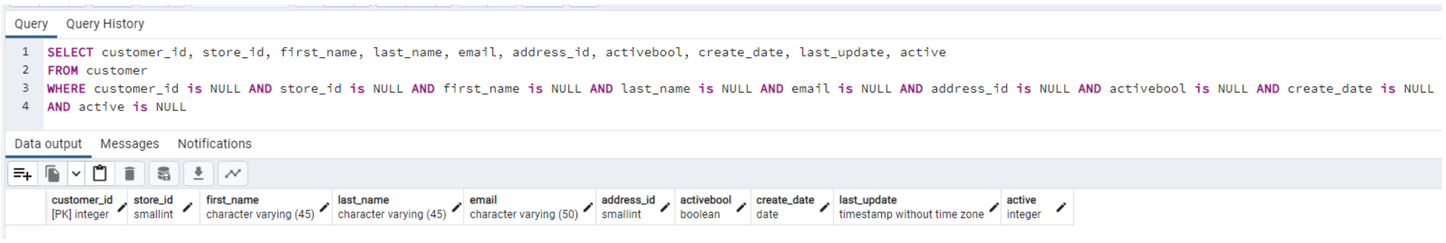
Missing values from Film table:



```
SELECT film_id, title, description, release_year, language_id, rental_duration, rental_rate, length, replacement_cost, rating, last_update, special_features, fulltext
FROM film
WHERE film_id IS NULL and title IS NULL and description IS NULL and release_year IS NULL and language_id IS NULL and rental_duration IS NULL and rental_rate IS NULL
and length IS NULL and replacement_cost IS NULL AND rating IS NULL AND last_update IS NULL AND special_features IS NULL AND fulltext IS NULL
```

film_id [PK] integer	title character varying (255)	description text	release_year integer	language_id smallint	rental_duration smallint	rental_rate numeric (4,2)	length smallint	replacement_cost numeric (5,2)	rating mpaa_rating	last_update timestamp without time zone	special_features text[]	fulltext tsvector

Missing values from Customer table:



```
SELECT customer_id, store_id, first_name, last_name, email, address_id, activebool, create_date, last_update, active
FROM customer
WHERE customer_id is NULL AND store_id is NULL AND first_name is NULL AND last_name is NULL AND email is NULL AND address_id is NULL AND activebool is NULL AND create_date is NULL
AND active is NULL
```

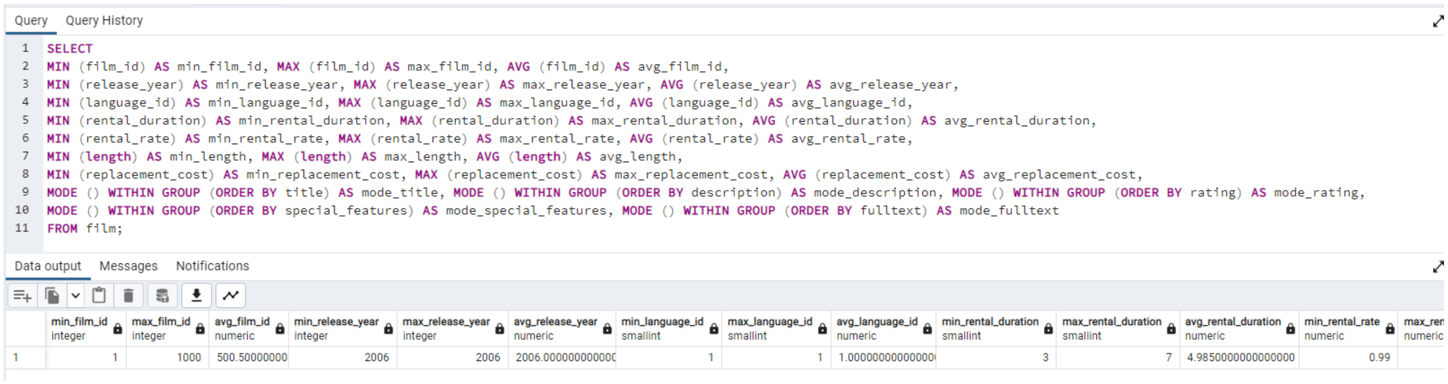
customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint	activebool boolean	create_date date	last_update timestamp without time zone	active integer

*There are no missing values in any columns in either table. If there were missing values, I could omit the column with the most missing values in my analysis. Alternatively, I could impute new values based on the average of that column using the UPDATE statement.

#2

Summarize your data: Use SQL to calculate descriptive statistics for both the film table and the customer table. For numerical columns, this means finding the minimum, maximum, and average values. For non-numerical columns, calculate the mode value.

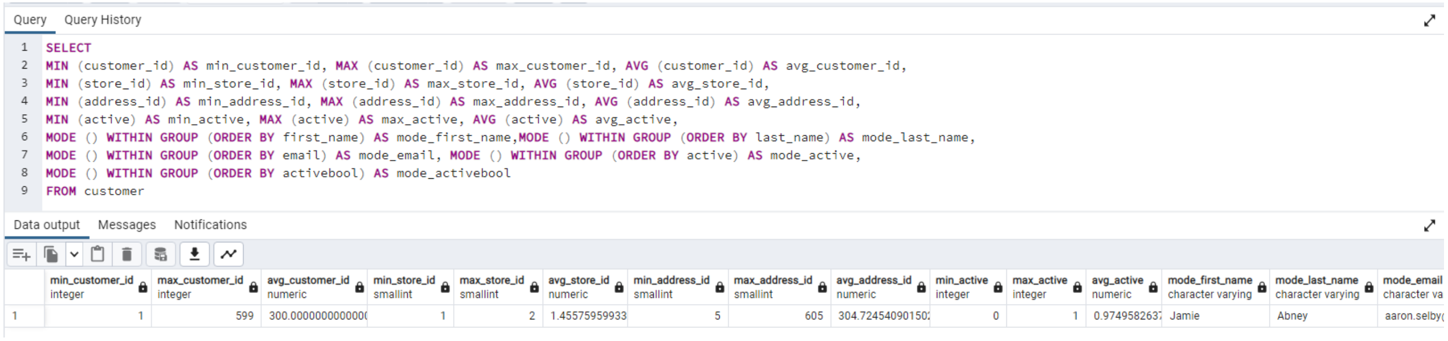
Film table:



```sql
SELECT
MIN (film_id) AS min_film_id, MAX (film_id) AS max_film_id, AVG (film_id) AS avg_film_id,
MIN (release_year) AS min_release_year, MAX (release_year) AS max_release_year, AVG (release_year) AS avg_release_year,
MIN (language_id) AS min_language_id, MAX (language_id) AS max_language_id, AVG (language_id) AS avg_language_id,
MIN (rental_duration) AS min_rental_duration, MAX (rental_duration) AS max_rental_duration, AVG (rental_duration) AS avg_rental_duration,
MIN (rental_rate) AS min_rental_rate, MAX (rental_rate) AS max_rental_rate, AVG (rental_rate) AS avg_rental_rate,
MIN (length) AS min_length, MAX (length) AS max_length, AVG (length) AS avg_length,
MIN (replacement_cost) AS min_replacement_cost, MAX (replacement_cost) AS max_replacement_cost, AVG (replacement_cost) AS avg_replacement_cost,
MODE () WITHIN GROUP (ORDER BY title) AS mode_title, MODE () WITHIN GROUP (ORDER BY description) AS mode_description, MODE () WITHIN GROUP (ORDER BY rating) AS mode_rating,
MODE () WITHIN GROUP (ORDER BY special_features) AS mode_special_features, MODE () WITHIN GROUP (ORDER BY fulltext) AS mode_fulltext
FROM film;
```

	min_film_id integer	max_film_id integer	avg_film_id numeric	min_release_year integer	max_release_year integer	avg_release_year numeric	min_language_id smallint	max_language_id smallint	avg_language_id numeric	min_rental_duration smallint	max_rental_duration smallint	avg_rental_duration numeric	min_rental_rate numeric	max_ren numeric
1	1	1000	500.50000000	2006	2006	2006.000000000000	1	1	1.000000000000000	3	7	4.9850000000000000	0.99	

Customer table:

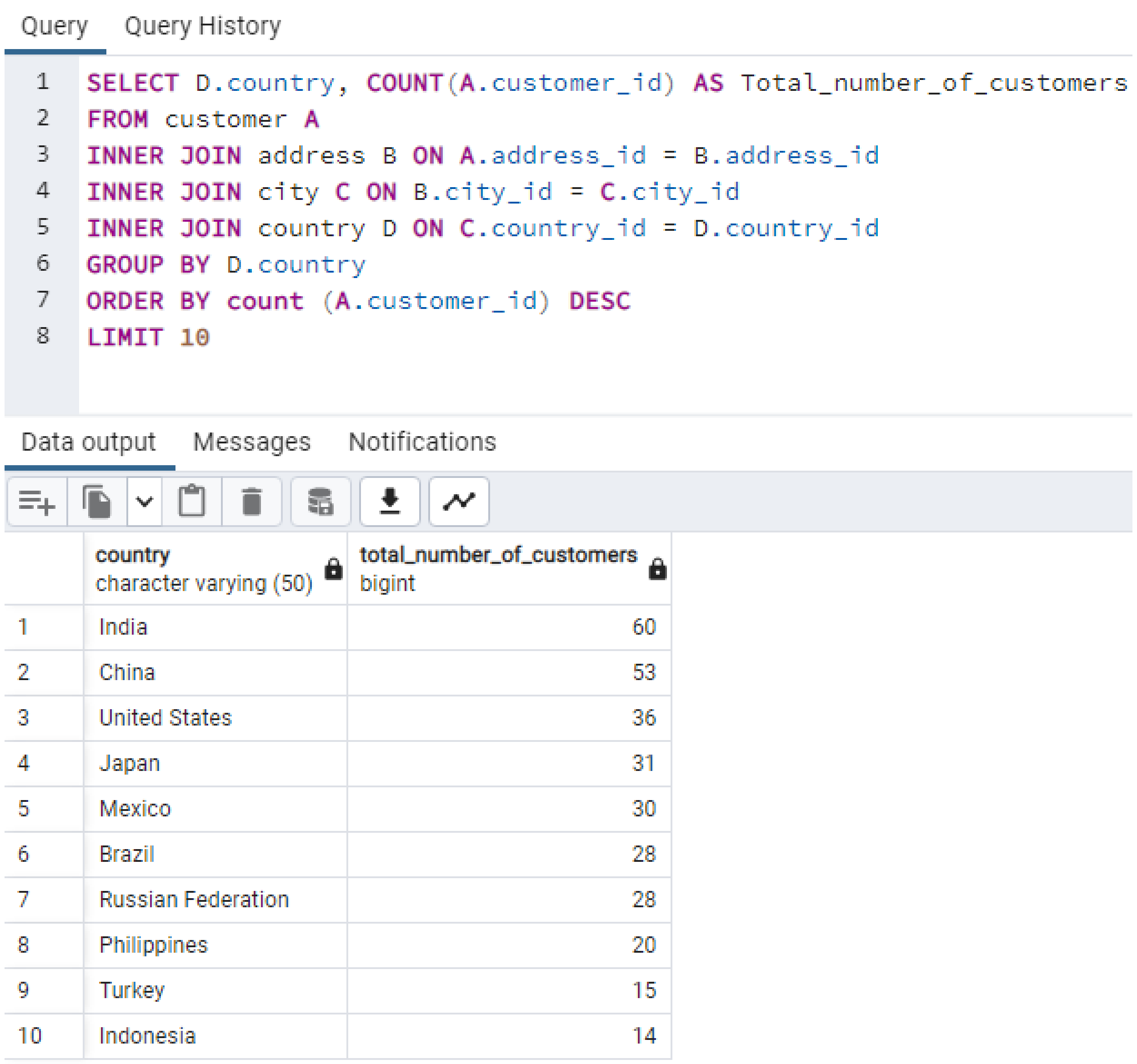


```sql
SELECT
MIN (customer_id) AS min_customer_id, MAX (customer_id) AS max_customer_id, AVG (customer_id) AS avg_customer_id,
MIN (store_id) AS min_store_id, MAX (store_id) AS max_store_id, AVG (store_id) AS avg_store_id,
MIN (address_id) AS min_address_id, MAX (address_id) AS max_address_id, AVG (address_id) AS avg_address_id,
MIN (active) AS min_active, MAX (active) AS max_active, AVG (active) AS avg_active,
MODE () WITHIN GROUP (ORDER BY first_name) AS mode_first_name,MODE () WITHIN GROUP (ORDER BY last_name) AS mode_last_name,
MODE () WITHIN GROUP (ORDER BY email) AS mode_email, MODE () WITHIN GROUP (ORDER BY active) AS mode_active,
MODE () WITHIN GROUP (ORDER BY activebool) AS mode_activebool
FROM customer
```

	min_customer_id integer	max_customer_id integer	avg_customer_id numeric	min_store_id smallint	max_store_id smallint	avg_store_id numeric	min_address_id smallint	max_address_id smallint	avg_address_id numeric	min_active integer	max_active integer	avg_active numeric	mode_first_name character varying	mode_last_name character varying	mode_email character va
1	1	599	300.000000000000	1	2	1.45575959933	5	605	304.72454090150	0	1	0.9749582637	Jamie	Abney	aaron.selby(

#3

Write a query to find the top 10 countries for Rockbuster in terms of customer numbers. Copy-paste your query and its output into your answers document. Write a few sentences on how you approached this query and why.



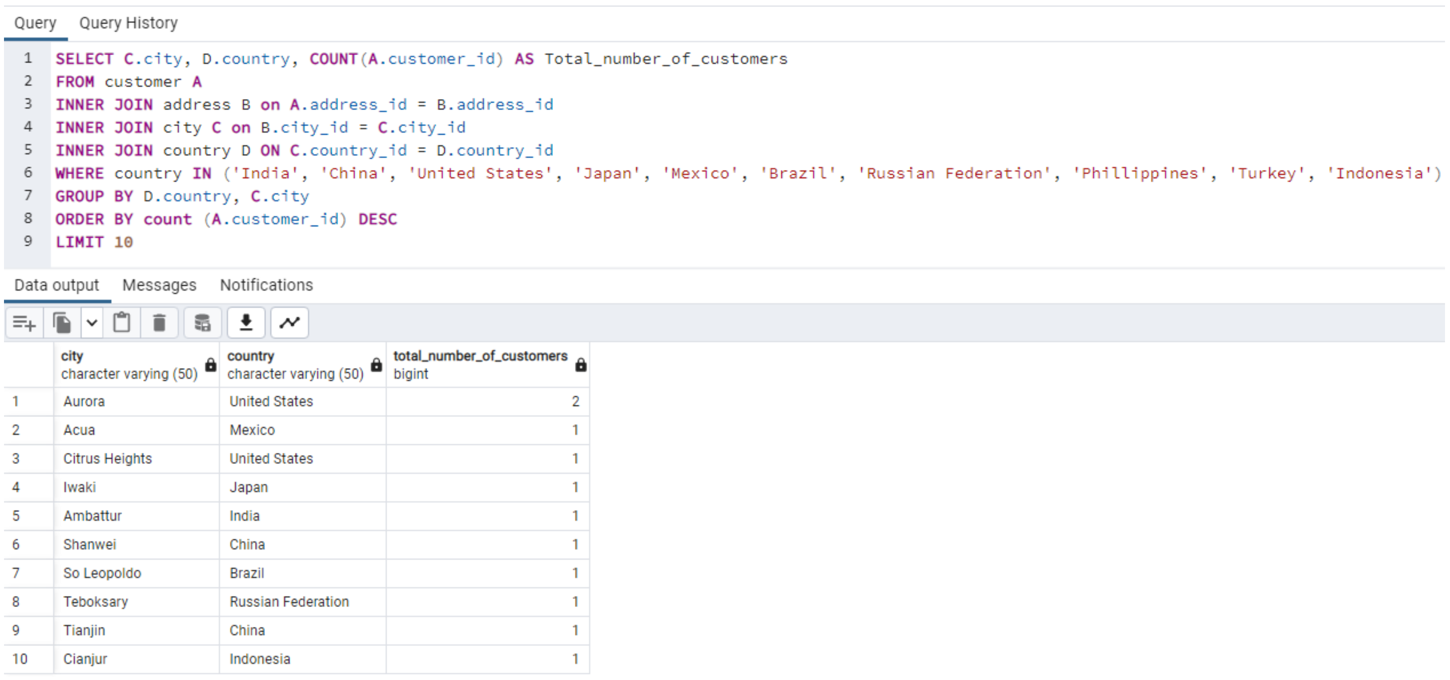
```
SELECT D.country, COUNT(A.customer_id) AS Total_number_of_customers
FROM customer A
INNER JOIN address B ON A.address_id = B.address_id
INNER JOIN city C ON B.city_id = C.city_id
INNER JOIN country D ON C.country_id = D.country_id
GROUP BY D.country
ORDER BY count (A.customer_id) DESC
LIMIT 10
```

	country character varying (50)	total_number_of_customers bigint
1	India	60
2	China	53
3	United States	36
4	Japan	31
5	Mexico	30
6	Brazil	28
7	Russian Federation	28
8	Philippines	20
9	Turkey	15
10	Indonesia	14

*For this task, I looked at the ERD to examine the relationship between the tables. I located the primary keys and the foreign keys that connected the tables I needed. I used the INNER JOIN clause to connect the Customer table to the Address table (through customer_id), the Address table to City table (through city_id), and City table to Country table (through coutnry_id). I limited the search to 10 as required and made sure to keep the statements in the correct order.

#4

Write a query to find the top 10 cities within the top 10 countries identified in step 1. Copy-paste your query and its output into your answers document. Write a short explanation of how you approached this query and why.



Query Query History

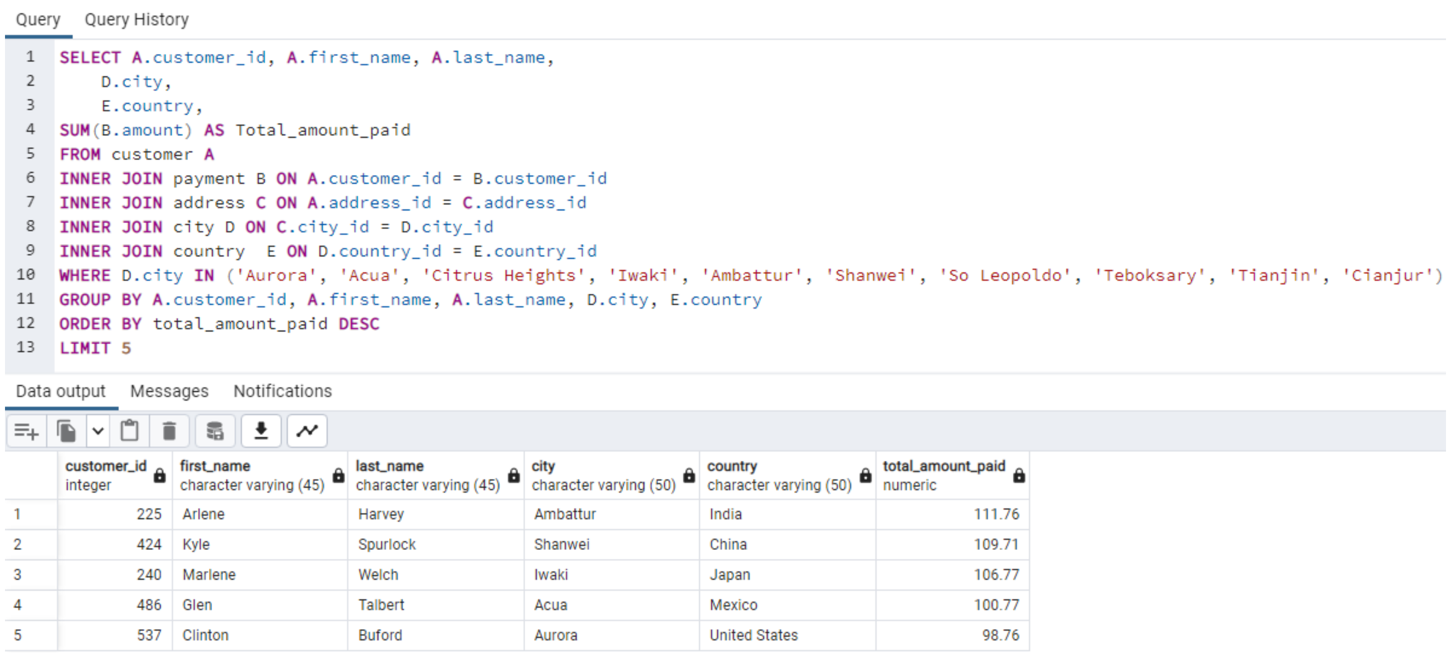
```
SELECT C.city, D.country, COUNT(A.customer_id) AS Total_number_of_customers
FROM customer A
INNER JOIN address B on A.address_id = B.address_id
INNER JOIN city C on B.city_id = C.city_id
INNER JOIN country D ON C.country_id = D.country_id
WHERE country IN ('India', 'China', 'United States', 'Japan', 'Mexico', 'Brazil', 'Russian Federation', 'Phillippines', 'Turkey', 'Indonesia')
GROUP BY D.country, C.city
ORDER BY count (A.customer_id) DESC
LIMIT 10
```

Data output Messages Notifications

	city character varying (50)	country character varying (50)	total_number_of_customers bigint
1	Aurora	United States	2
2	Acua	Mexico	1
3	Citrus Heights	United States	1
4	Iwaki	Japan	1
5	Ambattur	India	1
6	Shanwei	China	1
7	So Leopoldo	Brazil	1
8	Teboksary	Russian Federation	1
9	Tianjin	China	1
10	Cianjur	Indonesia	1

#5

Write a query to find the top 5 customers in the top 10 cities who have paid the highest total amounts to Rockbuster. Copy-paste your query and its output into your answers document.



Query Query History

```
SELECT A.customer_id, A.first_name, A.last_name,
    D.city,
    E.country,
SUM(B.amount) AS Total_amount_paid
FROM customer A
INNER JOIN payment B ON A.customer_id = B.customer_id
INNER JOIN address C ON A.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', 'Cianjur')
GROUP BY A.customer_id, A.first_name, A.last_name, D.city, E.country
ORDER BY total_amount_paid DESC
LIMIT 5
```

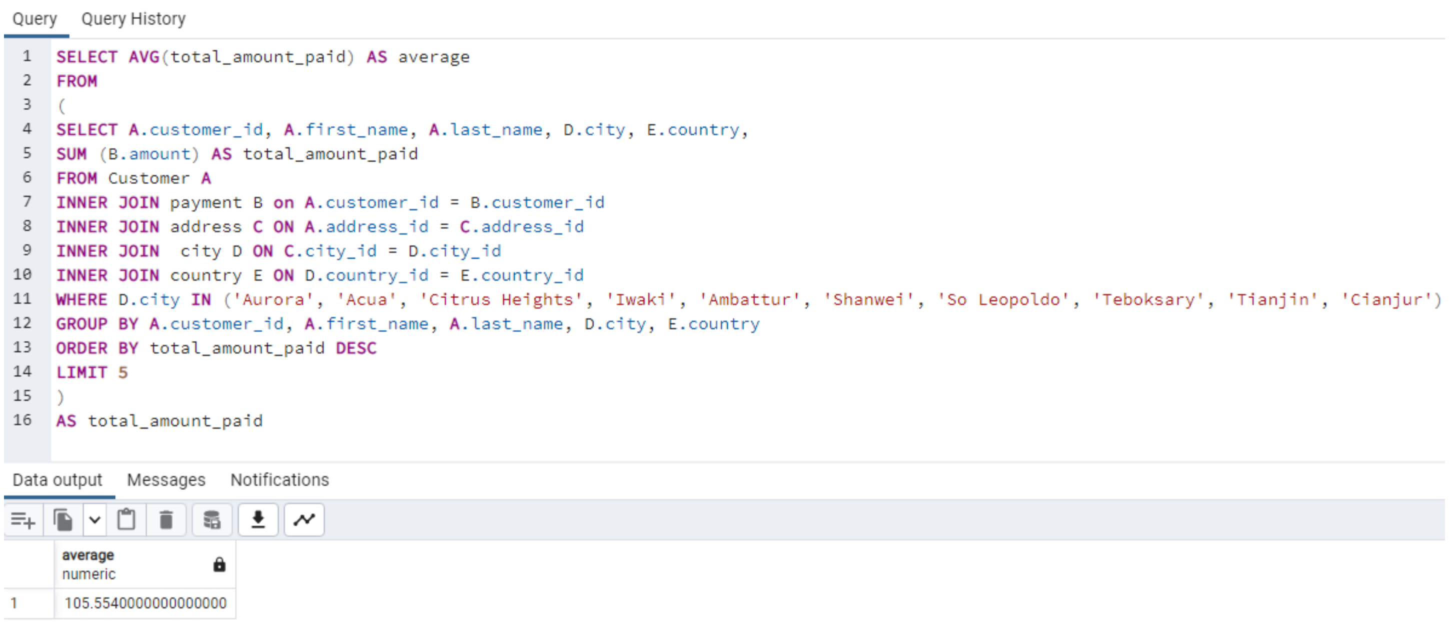
Data output Messages Notifications

	customer_id integer	first_name character varying (45)	last_name character varying (45)	city character varying (50)	country character varying (50)	total_amount_paid numeric
1	225	Arlene	Harvey	Ambattur	India	111.76
2	424	Kyle	Spurlock	Shanwei	China	109.71
3	240	Marlene	Welch	Iwaki	Japan	106.77
4	486	Glen	Talbert	Acua	Mexico	100.77
5	537	Clinton	Buford	Aurora	United States	98.76

#6

Find the average amount paid by the top 5 customers.

1. Write an outer statement to calculate the average amount paid.
2. Add your subquery to the outer statement. It will go in either the SELECT, WHERE, or FROM clause. (Hint: When referring to the subquery in your outer statement, make sure to use the subquery's alias, "total_amount_paid".)
3. If you've done everything correctly, pgAdmin 4 will require you to add an alias after the subquery. Go ahead and call it "average".
4. Copy-paste your queries and the final data output from pgAdmin 4 into your answers document.

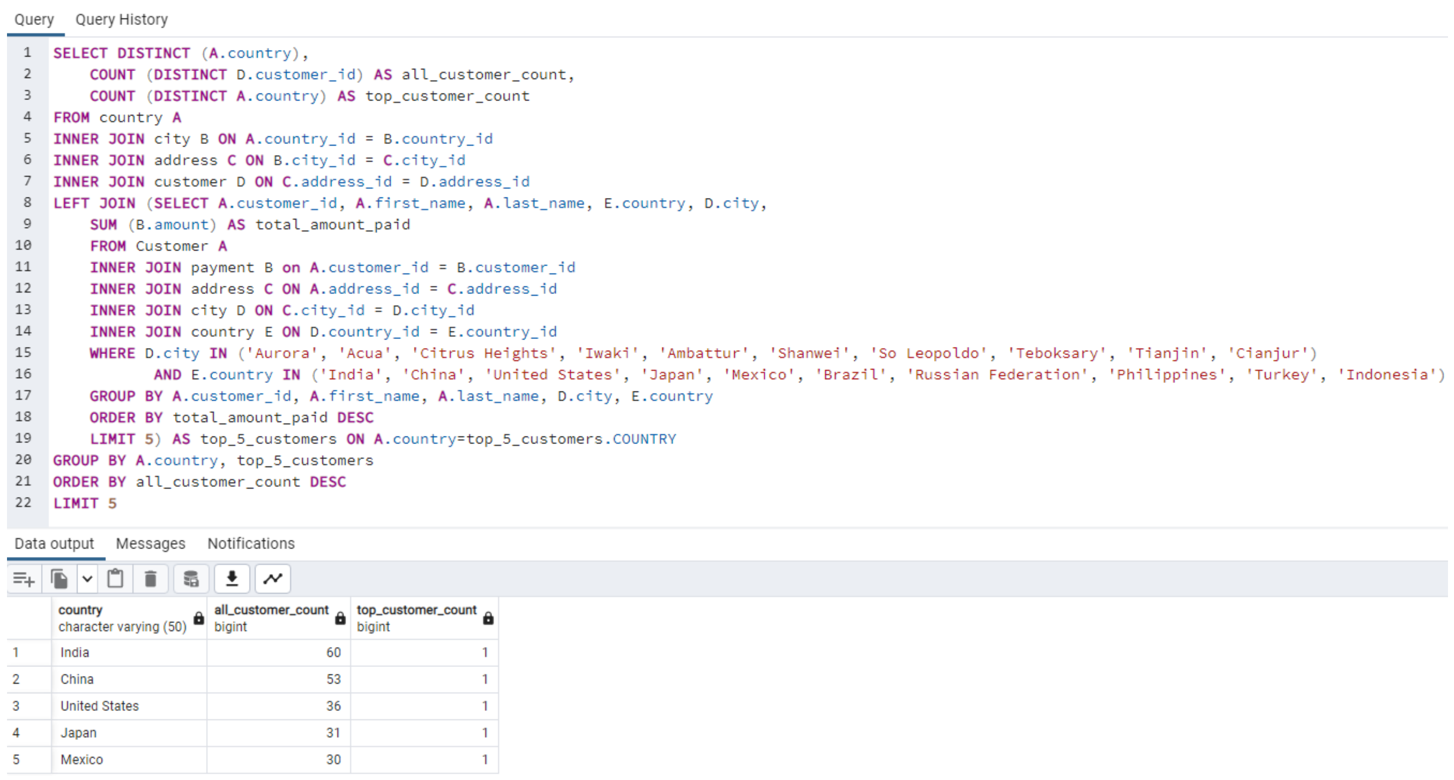


```sql
SELECT AVG(total_amount_paid) AS average
FROM
(
SELECT A.customer_id, A.first_name, A.last_name, D.city, E.country,
SUM (B.amount) AS total_amount_paid
FROM Customer A
INNER JOIN payment B on A.customer_id = B.customer_id
INNER JOIN address C ON A.address_id = C.address_id
INNER JOIN  city D ON 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', 'Cianjur')
GROUP BY A.customer_id, A.first_name, A.last_name, D.city, E.country
ORDER BY total_amount_paid DESC
LIMIT 5
)
AS total_amount_paid
```

	average numeric
1	105.5540000000000000

#7

Find out how many of the top 5 customers are based within each country. Your final output should include 3 columns: “country”, “all_customer_count” with the total number of customers in each country, “top_customer_count” showing how many of the top 5 customers live in each country.



Query   Query History

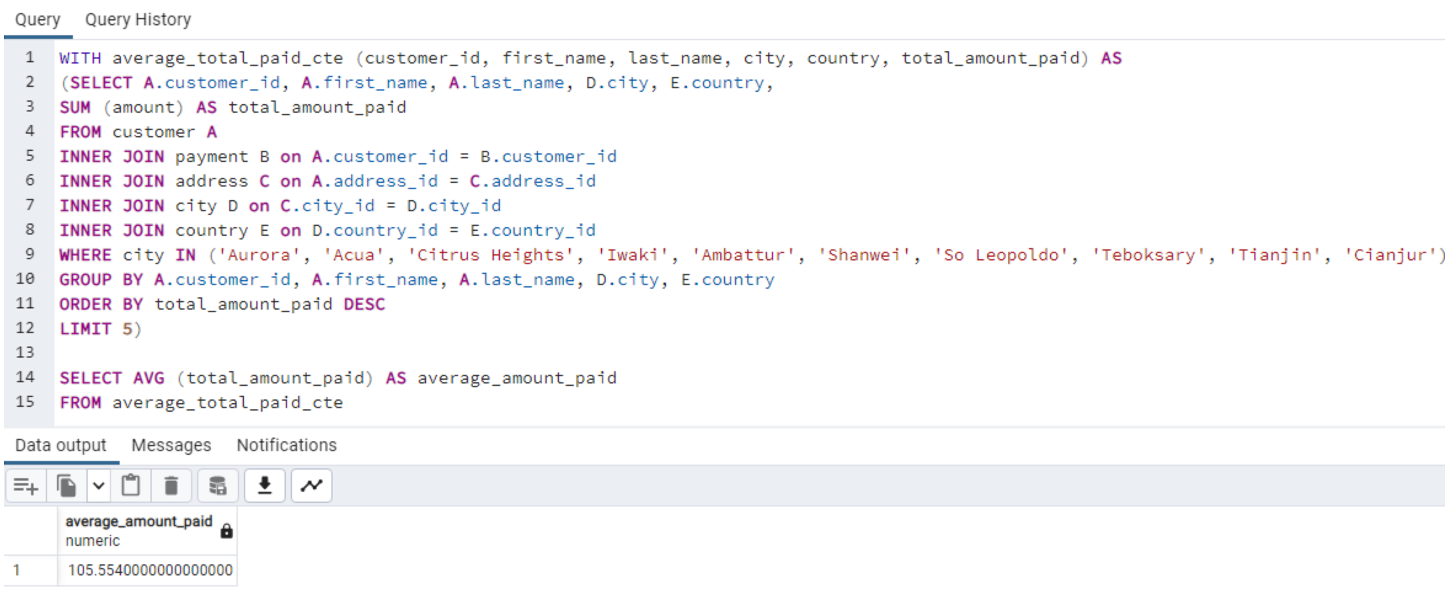
```sql
SELECT DISTINCT (A.country),
    COUNT (DISTINCT D.customer_id) AS all_customer_count,
    COUNT (DISTINCT A.country) AS top_customer_count
FROM country A
INNER JOIN city B ON A.country_id = B.country_id
INNER JOIN address C ON B.city_id = C.city_id
INNER JOIN customer D ON C.address_id = D.address_id
LEFT JOIN (SELECT A.customer_id, A.first_name, A.last_name, E.country, D.city,
    SUM (B.amount) AS total_amount_paid
    FROM Customer A
    INNER JOIN payment B on A.customer_id = B.customer_id
    INNER JOIN address C ON A.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', 'Cianjur')
          AND E.country IN ('India', 'China', 'United States', 'Japan', 'Mexico', 'Brazil', 'Russian Federation', 'Philippines', 'Turkey', 'Indonesia')
    GROUP BY A.customer_id, A.first_name, A.last_name, D.city, E.country
    ORDER BY total_amount_paid DESC
    LIMIT 5) AS top_5_customers ON A.country=top_5_customers.COUNTRY
GROUP BY A.country, top_5_customers
ORDER BY all_customer_count DESC
LIMIT 5
```

Data output   Messages   Notifications

	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

#8

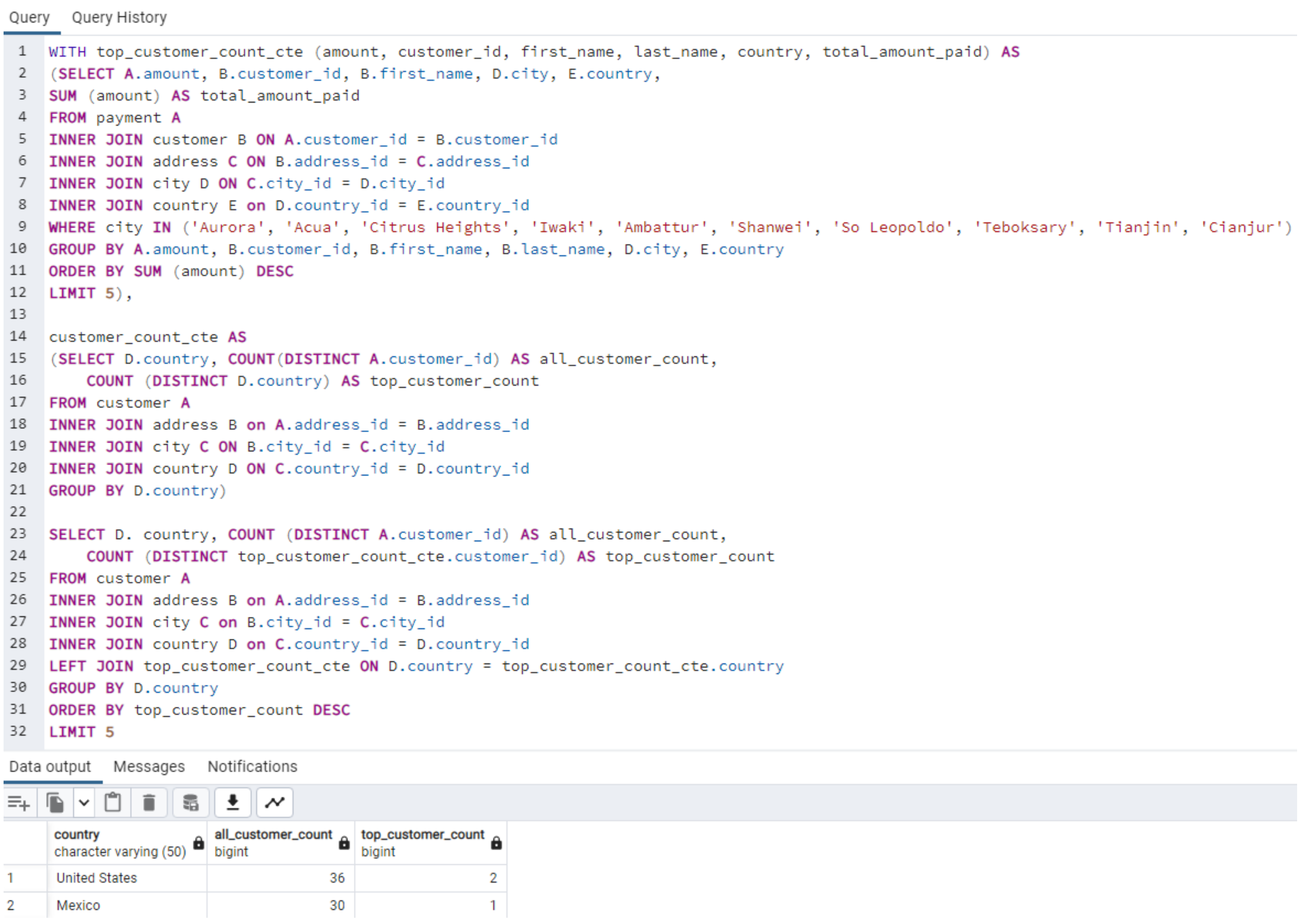
Rewrite previous queries as CTEs.



```sql
WITH average_total_paid_cte (customer_id, first_name, last_name, city, country, total_amount_paid) AS
(SELECT A.customer_id, A.first_name, A.last_name, D.city, E.country,
SUM (amount) AS total_amount_paid
FROM customer A
INNER JOIN payment B on A.customer_id = B.customer_id
INNER JOIN address C on A.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 city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')
GROUP BY A.customer_id, A.first_name, A.last_name, D.city, E.country
ORDER BY total_amount_paid DESC
LIMIT 5)

SELECT AVG (total_amount_paid) AS average_amount_paid
FROM average_total_paid_cte
```

	average_amount_paid numeric
1	105.5540000000000000



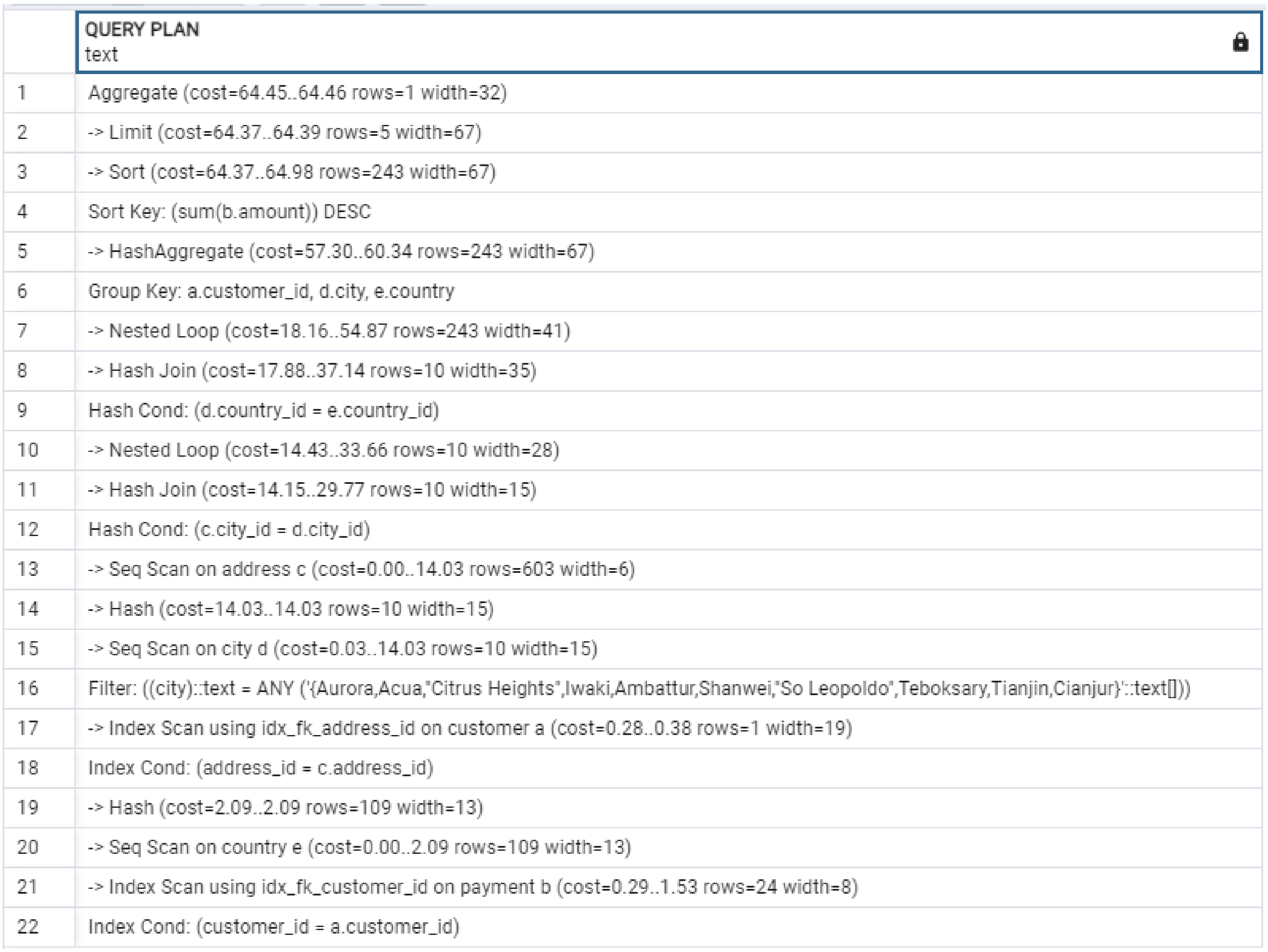
```sql
WITH top_customer_count_cte (amount, customer_id, first_name, last_name, country, total_amount_paid) AS
(SELECT A.amount, B.customer_id, B.first_name, D.city, E.country,
SUM (amount) AS total_amount_paid
FROM payment A
INNER JOIN customer B ON A.customer_id = B.customer_id
INNER JOIN address C 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
WHERE city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')
GROUP BY A.amount, B.customer_id, B.first_name, B.last_name, D.city, E.country
ORDER BY SUM (amount) DESC
LIMIT 5),

customer_count_cte AS
(SELECT D.country, COUNT(DISTINCT A.customer_id) AS all_customer_count,
    COUNT (DISTINCT D.country) 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
GROUP BY D.country)

SELECT D. country, COUNT (DISTINCT A.customer_id) AS all_customer_count,
    COUNT (DISTINCT top_customer_count_cte.customer_id) 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 top_customer_count_cte ON D.country = top_customer_count_cte.country
GROUP BY D.country
ORDER BY top_customer_count DESC
LIMIT 5
```

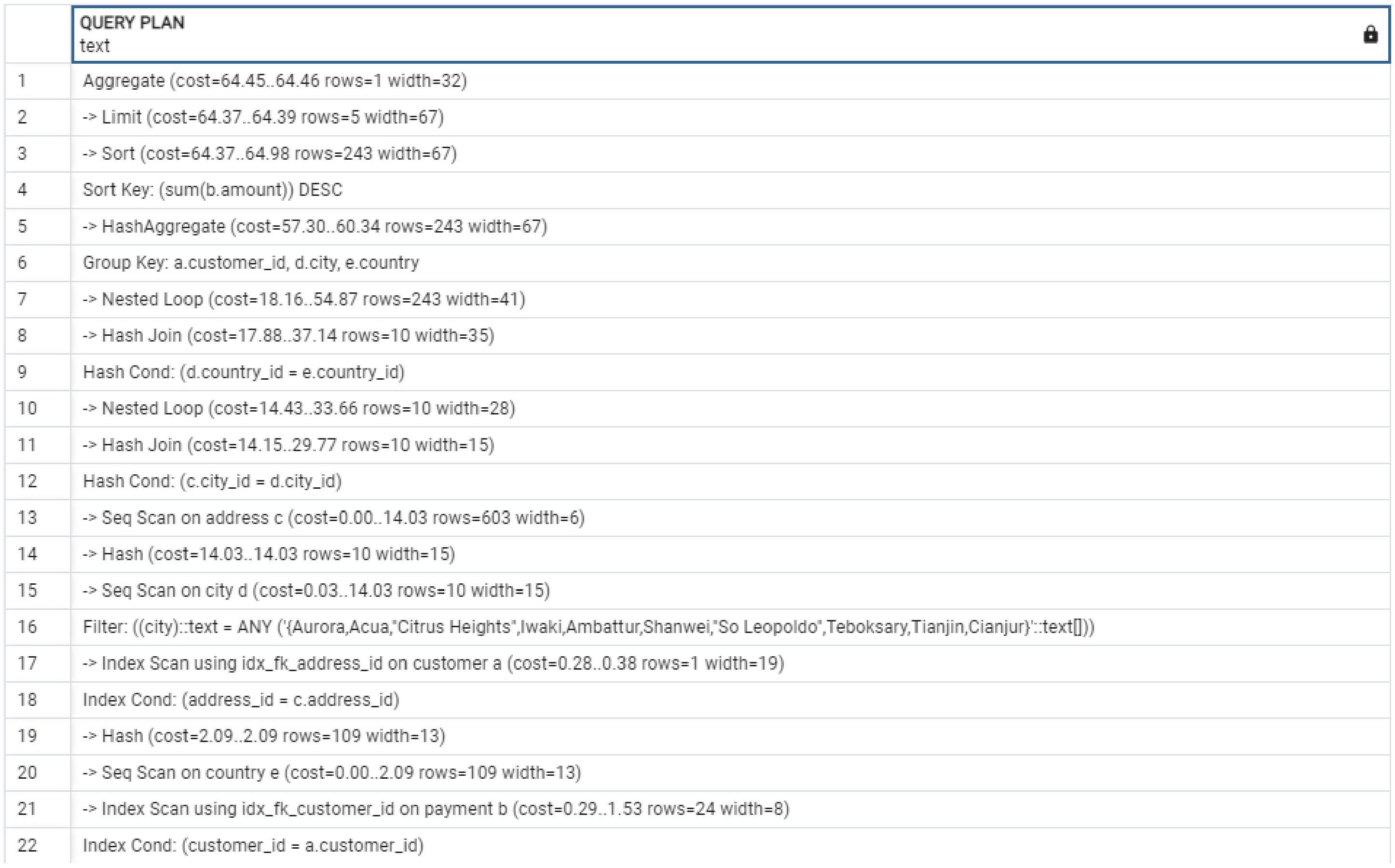
	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	United States	36	2
2	Mexico	30	1

SUBQUERY QUERY PLAN



	QUERY PLAN text
1	Aggregate (cost=64.45..64.46 rows=1 width=32)
2	-> Limit (cost=64.37..64.39 rows=5 width=67)
3	-> Sort (cost=64.37..64.98 rows=243 width=67)
4	Sort Key: (sum(b.amount)) DESC
5	-> HashAggregate (cost=57.30..60.34 rows=243 width=67)
6	Group Key: a.customer_id, d.city, e.country
7	-> Nested Loop (cost=18.16..54.87 rows=243 width=41)
8	-> Hash Join (cost=17.88..37.14 rows=10 width=35)
9	Hash Cond: (d.country_id = e.country_id)
10	-> Nested Loop (cost=14.43..33.66 rows=10 width=28)
11	-> Hash Join (cost=14.15..29.77 rows=10 width=15)
12	Hash Cond: (c.city_id = d.city_id)
13	-> Seq Scan on address c (cost=0.00..14.03 rows=603 width=6)
14	-> Hash (cost=14.03..14.03 rows=10 width=15)
15	-> Seq Scan on city d (cost=0.03..14.03 rows=10 width=15)
16	Filter: ((city)::text = ANY ('{Aurora,Acua,"Citrus Heights",Iwaki,Ambattur,Shanwei,"So Leopoldo",Teboksary,Tianjin,Cianjur}'::text[]))
17	-> Index Scan using idx_fk_address_id on customer a (cost=0.28..0.38 rows=1 width=19)
18	Index Cond: (address_id = c.address_id)
19	-> Hash (cost=2.09..2.09 rows=109 width=13)
20	-> Seq Scan on country e (cost=0.00..2.09 rows=109 width=13)
21	-> Index Scan using idx_fk_customer_id on payment b (cost=0.29..1.53 rows=24 width=8)
22	Index Cond: (customer_id = a.customer_id)

<u>CTE QUERY PLAN</u>



	QUERY PLAN text
1	Aggregate (cost=64.45..64.46 rows=1 width=32)
2	-> Limit (cost=64.37..64.39 rows=5 width=67)
3	-> Sort (cost=64.37..64.98 rows=243 width=67)
4	Sort Key: (sum(b.amount)) DESC
5	-> HashAggregate (cost=57.30..60.34 rows=243 width=67)
6	Group Key: a.customer_id, d.city, e.country
7	-> Nested Loop (cost=18.16..54.87 rows=243 width=41)
8	-> Hash Join (cost=17.88..37.14 rows=10 width=35)
9	Hash Cond: (d.country_id = e.country_id)
10	-> Nested Loop (cost=14.43..33.66 rows=10 width=28)
11	-> Hash Join (cost=14.15..29.77 rows=10 width=15)
12	Hash Cond: (c.city_id = d.city_id)
13	-> Seq Scan on address c (cost=0.00..14.03 rows=603 width=6)
14	-> Hash (cost=14.03..14.03 rows=10 width=15)
15	-> Seq Scan on city d (cost=0.03..14.03 rows=10 width=15)
16	Filter: ((city)::text = ANY ('{Aurora,Acua,"Citrus Heights",Iwaki,Ambattur,Shanwei,"So Leopoldo",Teboksary,Tianjin,Cianjur}'::text[]))
17	-> Index Scan using idx_fk_address_id on customer a (cost=0.28..0.38 rows=1 width=19)
18	Index Cond: (address_id = c.address_id)
19	-> Hash (cost=2.09..2.09 rows=109 width=13)
20	-> Seq Scan on country e (cost=0.00..2.09 rows=109 width=13)
21	-> Index Scan using idx_fk_customer_id on payment b (cost=0.29..1.53 rows=24 width=8)
22	Index Cond: (customer_id = a.customer_id)

	Subquery	CTE
Cost	1 row	1 row
Speed	00.146 sec	0.120 sec