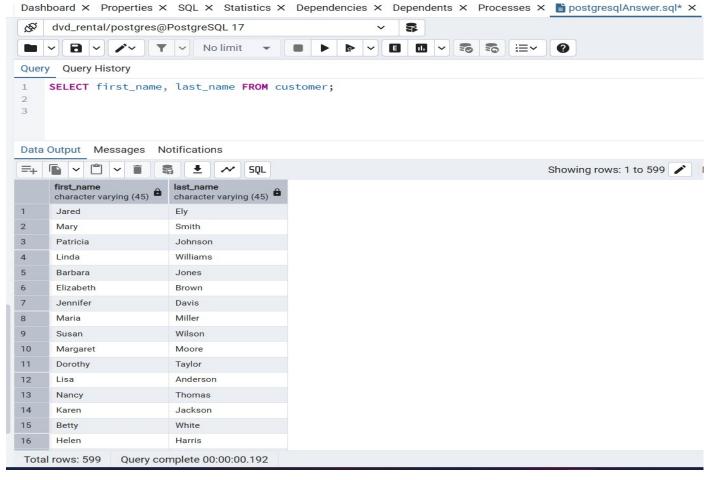
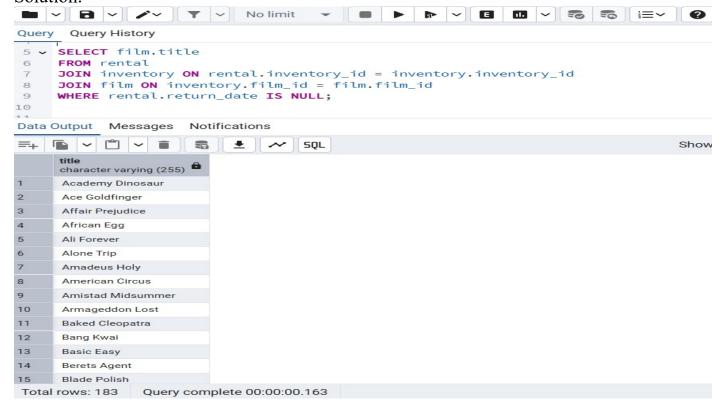
## POSTGRESQL ASSIGNMENT

#### 1. List the first name and last name of all customers.

Solution:

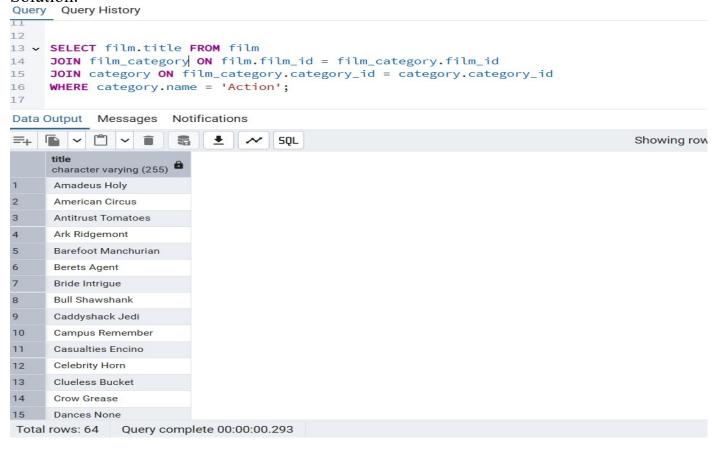


## 2. Find all the movies that are currently rented out.

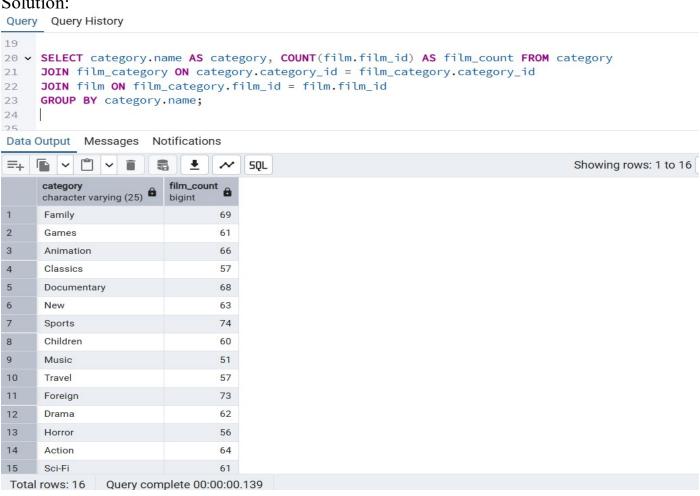


## 3. Show the titles of all movies in the 'Action' category.

#### Solution:

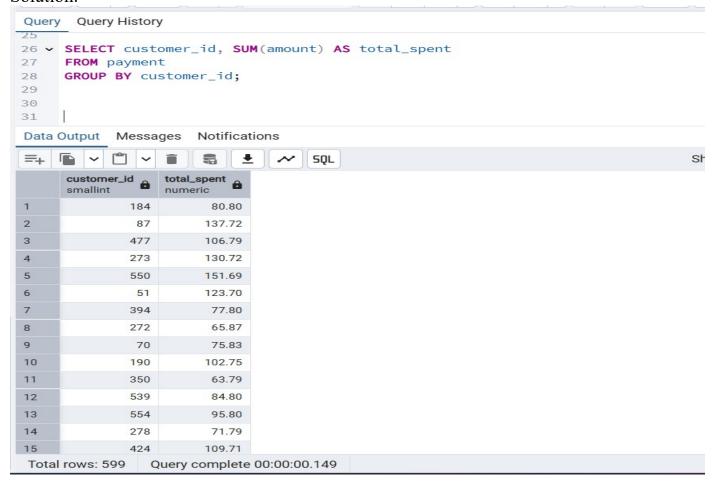


### 4. Count the number of films in each category.

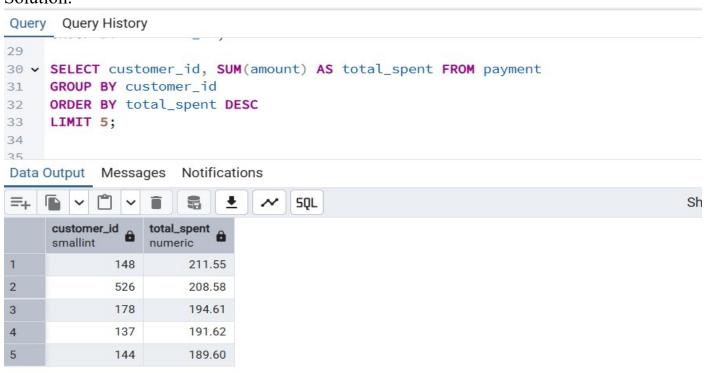


## 5. What is the total amount spent by each customer?

Solution:

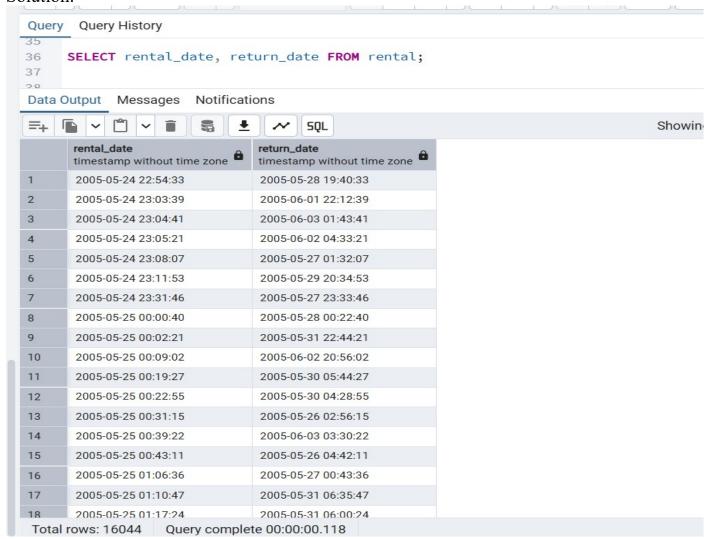


## 6. Find the top 5 customers who spent the most.

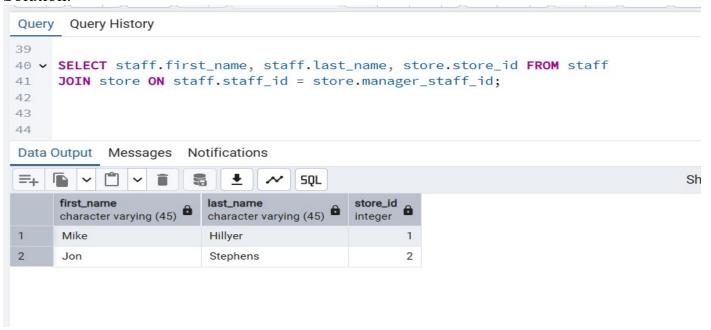


### 7. Display the rental date and return date for each rental.

Solution:

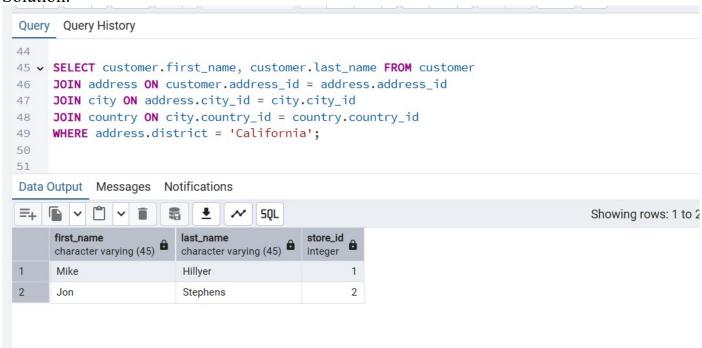


## 8. List the names of staff members and the stores they manage.

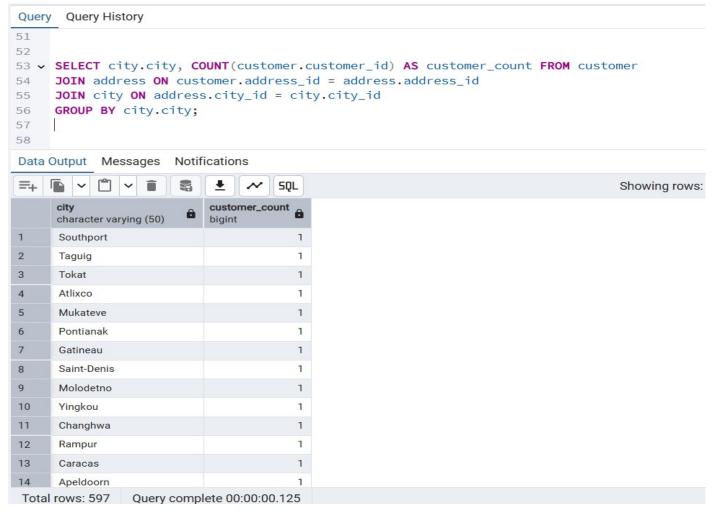


## 9. Find all customers living in 'California'.

Solution:

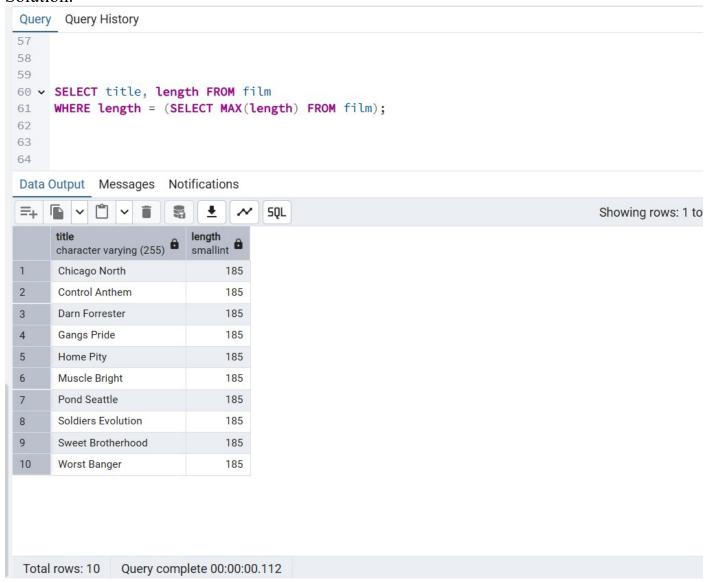


## 10. Count how many customers are from each city.

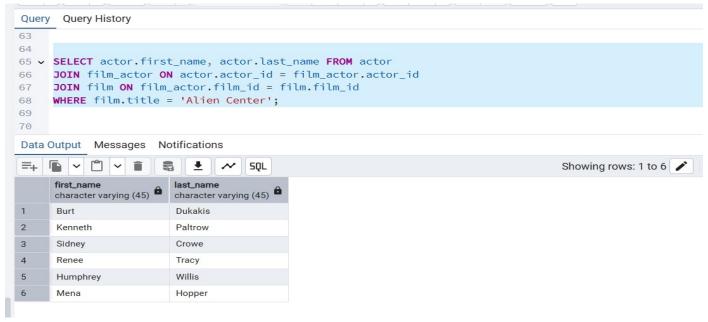


## 11. Find the film(s) with the longest duration.

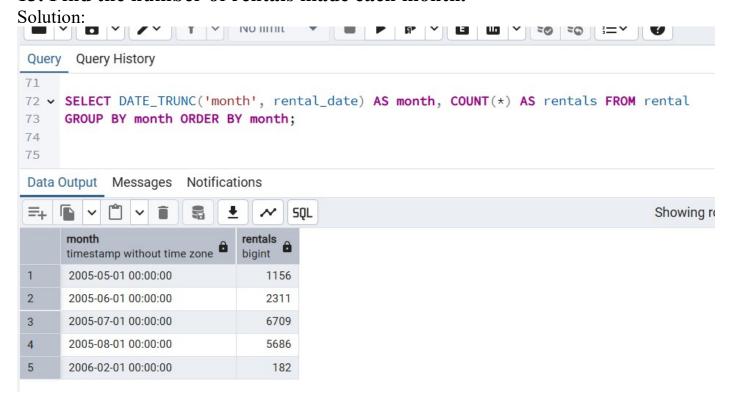
Solution:



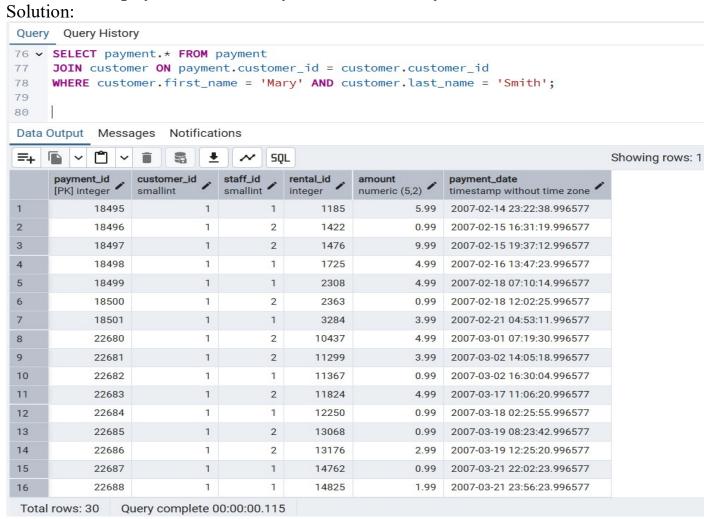
## 12. Which actors appear in the film titled 'Alien Center'?



#### 13. Find the number of rentals made each month.



## 14. Show all payments made by customer 'Mary Smith'.

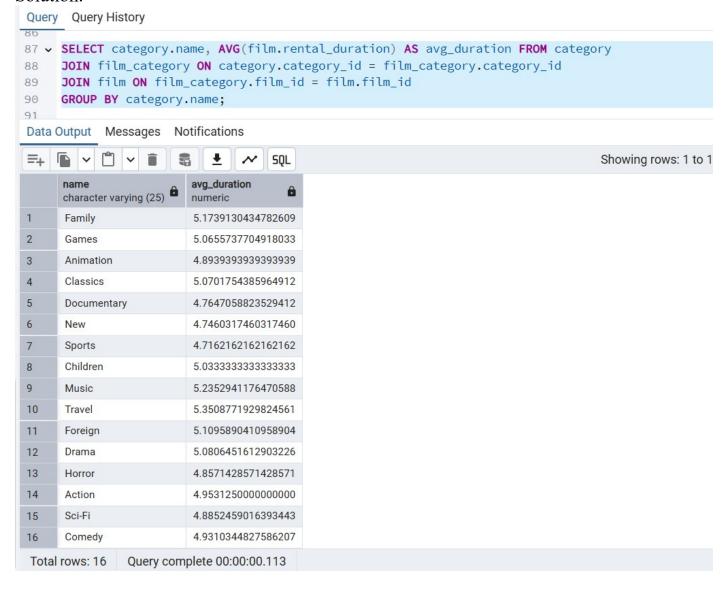


#### 15. List all films that have never been rented.

#### Solution:

```
Query Query History
81 - SELECT film.title FROM film
     JOIN inventory ON film.film_id = inventory.film_id
82
     LEFT JOIN rental ON inventory.inventory_id = rental.inventory_id
83
     WHERE rental.rental_id IS NULL;
84
85
Data Output
            Messages
                      Notifications
                                      SQL
=+
     character varying (255)
      Academy Dinosaur
```

## 16. What is the average rental duration per category?



#### 17. Which films were rented more than 50 times?

#### Solution:

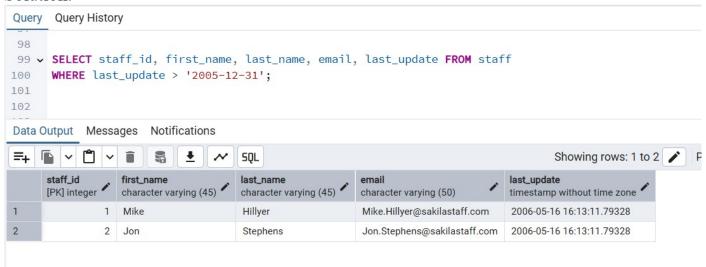
```
Query Query History

93  SELECT film.title, COUNT(*) AS rental_count FROM rental
94  JOIN inventory ON rental.inventory_id = inventory.inventory_id
95  JOIN film ON inventory.film_id = film.film_id
96  GROUP BY film.title HAVING COUNT(*) > 50;
97  Data Output Messages Notifications

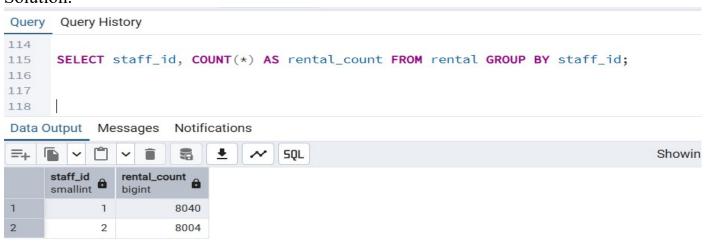
$\frac{1}{2}  \times \frac{1}{2}  \timps \frac{1}{2}  \times \frac{1}{2}  \times \frac{1}{2}  \times \
```

## 18. List all employees hired after the year 2005.

#### Solution:

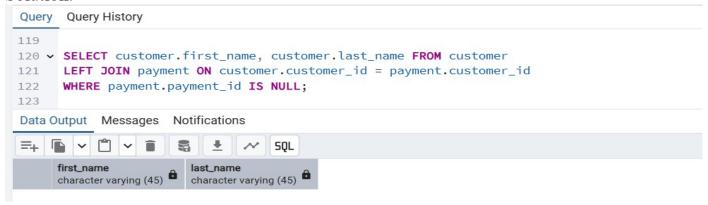


## 19. Show the number of rentals processed by each staff member.



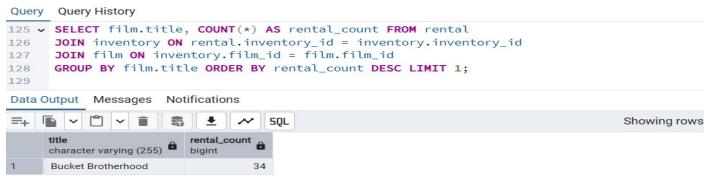
### 20. Display all customers who have not made any payments.

#### Solution:

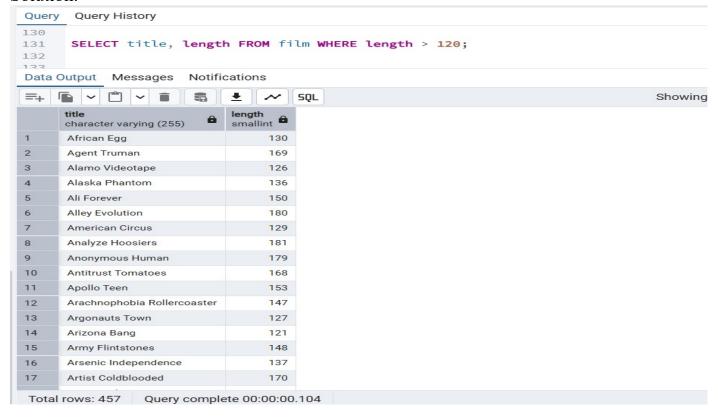


## 21. What is the most popular film (rented the most)?

#### Solution:

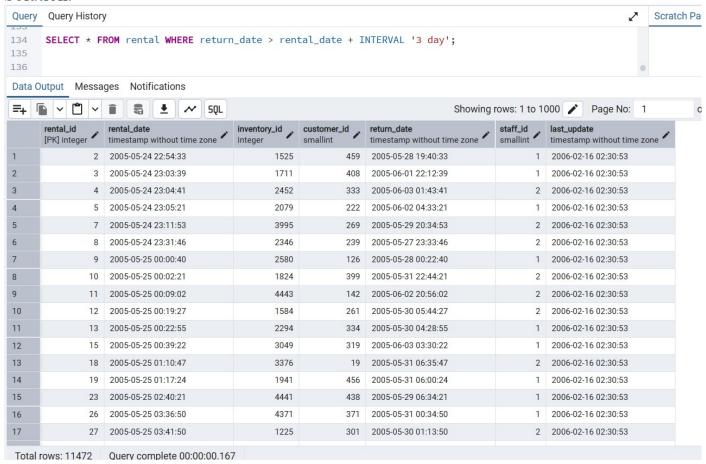


## 22. Show all films longer than 2 hours.

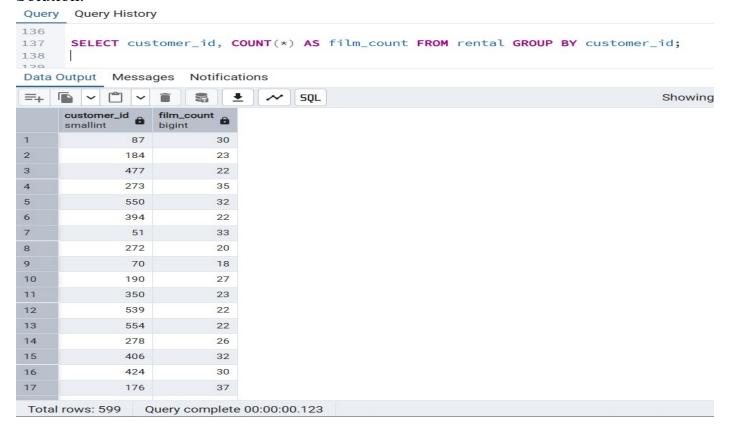


#### 23. Find all rentals that were returned late.

#### Solution:



## 24. List customers and the number of films they rented.



### 25. Write a query to show top 3 rented film categories.

#### Solution:

Query	Query History		
140 🕶	SELECT category.	name, COUNT(*) AS rental_count FROM rental	
141	JOIN inventory O	<pre>rental.inventory_id = inventory.inventory_id</pre>	
142	JOIN film_catego	ry <b>ON</b> inventory.film_id = film_category.film_id	
143	JOIN category ON	<pre>film_category.category_id = category.category_id</pre>	
144	GROUP BY category	y.name	
145	ORDER BY rental_	count DESC LIMIT 3;	
146			
Data O	utput Messages No	otifications	
<b>=</b> + ∏		₫ 🛂 ~ SQL	
	name character varying (25)	rental_count bigint	
1	Sports	1179	
2	Animation	1166	
3	Action	1112	

## 26. Create a view that shows all customer names and their payment totals.

#### Solution:

```
Query History

CREATE VIEW customer_payment_totals AS

SELECT customer.first_name, customer.last_name, SUM(payment.amount) AS total_payment FROM customer

JOIN payment ON customer.customer_id = payment.customer_id

GROUP BY customer.first_name, customer.last_name;

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 90 msec.
```

## 27. Update a customer's email address given their ID.

```
Query Query History

156
157
UPDATE customer SET email = 'newemail@example.com' WHERE customer_id = 1;
158
159
160

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 75 msec.
```

### 28. Insert a new actor into the actor table.

Solution:

```
Query Query History

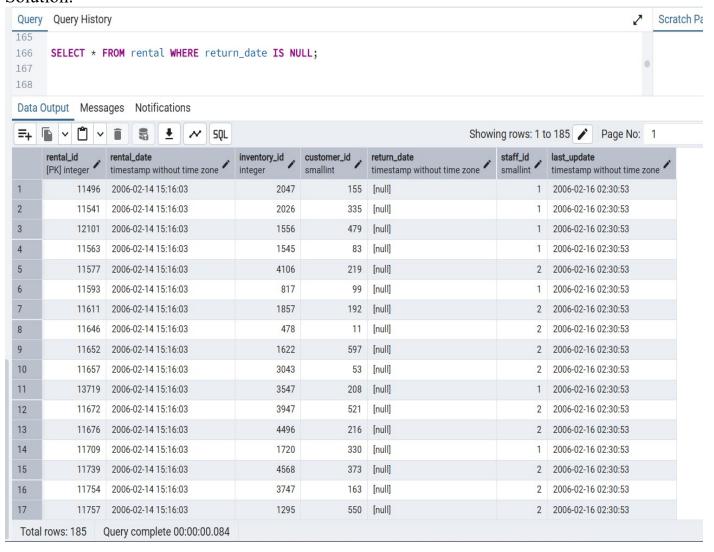
158
159
160
161
162

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 71 msec.
```

## 29. Delete all records from the rentals table where return\_date is NULL.



## 30. Add a new column 'age' to the customer table.

#### Solution:

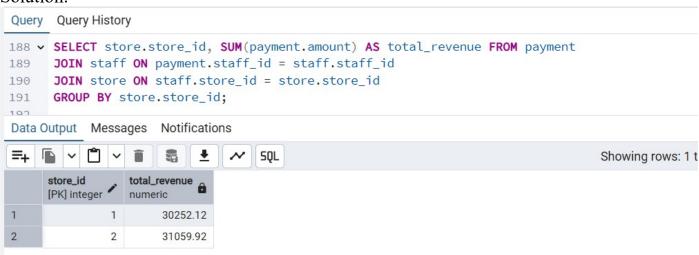


#### 31. Create an index on the 'title' column of the film table.

#### Solution:

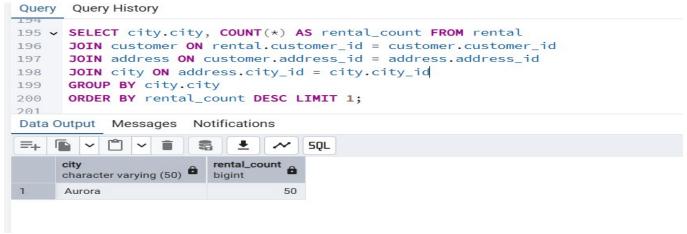


## 32. Find the total revenue generated by each store.



#### 33. What is the city with the highest number of rentals?

Solution:

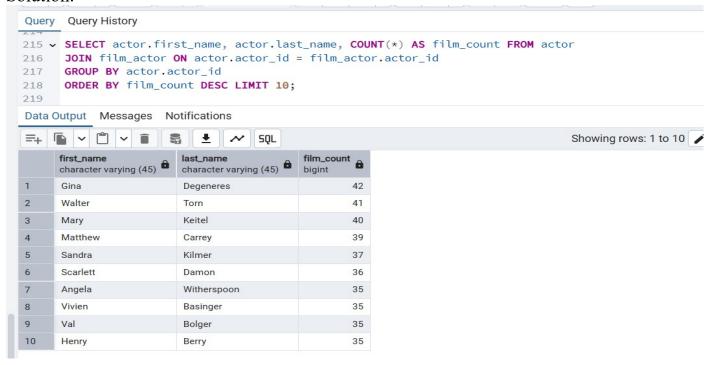


#### 34. How many films belong to more than one category?

Solution:



## 35. List the top 10 actors by number of films they appeared in.



## 36. Retrieve the email addresses of customers who rented 'Matrix Revolutions'.

#### Solution:

## 37. Create a stored function to return customer payment total given their ID.

#### Solution:

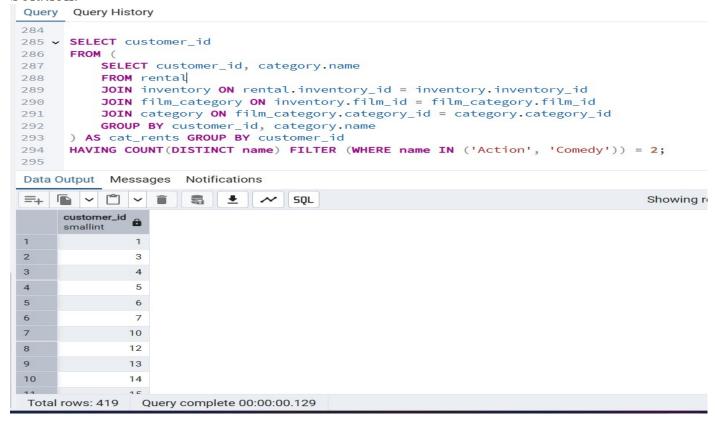
```
Query Query History
230 - CREATE OR REPLACE FUNCTION get_customer_total_payment(cust_id INT)
231
    RETURNS NUMERIC (10,2) AS $$
232 DECLARE
233
          total NUMERIC(10,2);
234 BEGIN
235
          SELECT COALESCE(SUM (amount), 0.00)
236
          INTO total
          FROM payment
237
238
          WHERE customer_id = cust_id;
239
          RETURN total;
     FND:
240
241
    $$ LANGUAGE plpgsql;
Data Output Messages Notifications
CREATE FUNCTION
Query returned successfully in 58 msec.
```

## 38. Begin a transaction that updates stock and inserts a rental record.

```
Query Query History
259
    BEGIN;
261 - UPDATE inventory
    SET last_update = NOW()
262
263
    WHERE inventory_id = 1;
264 • INSERT INTO rental (rental_date, inventory_id, customer_id, staff_id, last_update)
    VALUES (NOW(), 1, 1, 1, NOW());
265
266
     COMMIT;
267
Data Output Messages Notifications
COMMIT
Query returned successfully in 94 msec.
```

# 39. Show the customers who rented films in both 'Action' and 'Comedy' categories.

#### Solution:



### 40. Find actors who have never acted in a film.

