## PostgreSQL Assignment Questions - Sure Pro Education

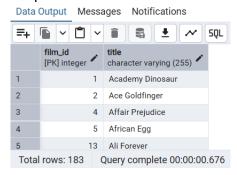
- 1. List the first name and last name of all customers.
  - → Query: select first\_name, last\_name from customer;

### Output:

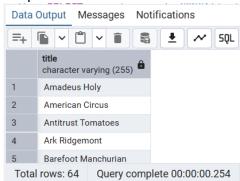


- 2. Find all the movies that are currently rented out.
  - → Query: SELECT f.film\_id, f.title
    FROM film f
    JOIN inventory i ON f.film\_id = i.film\_id
    JOIN rental r ON i.inventory\_id = r.inventory\_id
    WHERE r.return\_date IS NULL;

### Output:



- 3. Show the titles of all movies in the 'Action' category.
  - → Query: Select f.title from film f join film\_category fc on f.film\_id = fc.film\_id join category c on fc.category\_id = c.category\_id where c.name LIKE '%Action%';



- 4. Count the number of films in each category.
  - → Query: SELECT c.name AS category, COUNT(f.film\_id) AS film\_count FROM category c

    JOIN film\_category fc ON c.category\_id = fc.category\_id

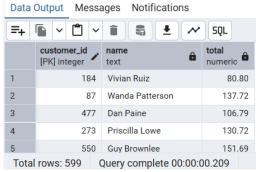
    JOIN film f ON fc.film\_id = f.film\_id

    GROUP BY c.name;

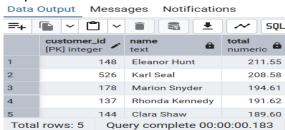


- 5. What is the total amount spent by each customer?
  - → Query: SELECT c.customer\_id, CONCAT(c.first\_name, ' ', c.last\_name) AS name, SUM(p.amount) AS total FROM customer c
    JOIN payment p ON c.customer\_id = p.customer\_id
    GROUP BY c.customer id;

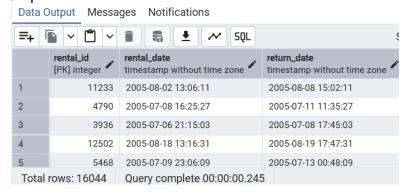
# Output:



- 6. Find the top 5 customers who spent the most.
  - → Query: SELECT c.customer\_id, CONCAT(c.first\_name, '', c.last\_name) AS name, SUM(p.amount) AS total FROM customer c JOIN payment p ON c.customer\_id = p.customer\_id GROUP BY c.customer id Order by total DESC LIMIT 5;

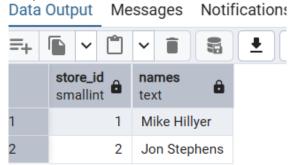


- 7. Display the rental date and return date for each rental.
  - → Query: select rental id, rental date, return date from rental group by rental id;



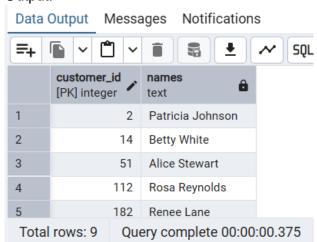
- 8. List the names of staff members and the stores they manage.
  - → Query: Select st.store\_id, CONCAT(st.first\_name, ' ', st.last\_name) as names from staff st join store s on st.store\_id = s.store\_id;

### Output:

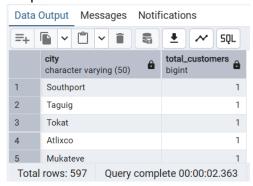


- 9. Find all customers living in 'California'.
  - → Query: SELECT customer\_id, CONCAT(first\_name, '',last\_name) as names FROM customer
    WHERE address\_id IN (
    SELECT address\_id FROM address WHERE district = 'Colifornia'):

SELECT address\_id FROM address WHERE district = 'California');

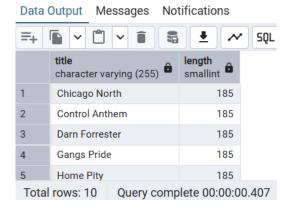


- 10. Count how many customers are from each city.
  - → Query: SELECT ci.city, COUNT(c.customer\_id) AS total\_customers FROM customer c JOIN address a ON c.address\_id = a.address\_id JOIN city ci ON a.city\_id = ci.city\_id GROUP BY ci.city;



- 11. Find the film(s) with the longest duration.
  - → Query: SELECT title, length
    FROM film
    WHERE length = (SELECT MAX(length) FROM film);

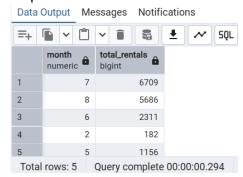
#### Output:



- 12. Which actors appear in the film titled 'Alien Center'?
  - → Query: select CONCAT(first\_name, '', last\_name) as name from actor a join film\_actor fa on a.actor\_id = fa.actor\_id join film f on fa.film\_id = f.film\_id where f.title = 'Alien Center';

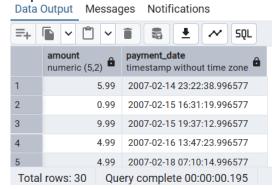


- 13. Find the number of rentals made each month.
  - → Query: SELECT EXTRACT(MONTH FROM rental\_date) AS month, COUNT(rental\_id)
    AS total\_rentals FROM rental
    GROUP BY month;

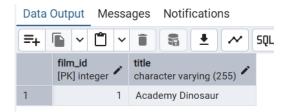


- 14. Show all payments made by customer 'Mary Smith'.
  - → Query: select amount, payment\_date from payment join customer c on payment.customer\_id = c.customer\_id where CONCAT(c.first\_name, '', c.last\_name) = 'Mary Smith';

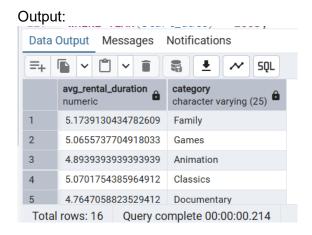
#### Output:



- 15. List all films that have never been rented.
  - → Query: SELECT f.film\_id, f.title
    FROM film f
    JOIN inventory i ON f.film\_id = i.film\_id
    LEFT JOIN rental r ON i.inventory\_id = r.inventory\_id
    WHERE r.rental id IS NULL;



- 16. What is the average rental duration per category?
  - → Query: select avg(f.rental\_duration) as avg\_rental\_duration ,c.name as category from film f join film\_category fc on fc.film\_id = f.film\_id join category c on fc.category\_id = c.category\_id group by c.name;



- 17. Which films were rented more than 50 times?
  - → Query: SELECT f.film\_id, f.title, COUNT(r.rental\_id) AS rental\_count FROM film f JOIN inventory i ON f.film\_id = i.film\_id JOIN rental r ON i.inventory\_id = r.inventory\_id GROUP BY f.film\_id, f.title HAVING COUNT(r.rental\_id) > 50;



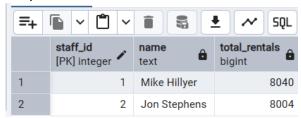
- 18. List all employees hired after the year 2005.
  - → Query: SELECT staff\_id, first\_name, last\_name, last\_update FROM staff WHERE last\_update > '2005-12-31';



- 19. Show the number of rentals processed by each staff member.
- → Query: SELECT s.staff\_id, CONCAT(s.first\_name, '', s.last\_name) as name, COUNT(r.rental\_id)
  AS total\_rentals FROM staff s

  JOIN rental r ON s.staff\_id = r.staff\_id

  GROUP BY s.staff\_id, s.first\_name, s.last\_name;



- 20. Display all customers who have not made any payments.
  - → Query: SELECT c.customer\_id, CONCAT(c.first\_name, '',c.last\_name) as name FROM customer c left JOIN payment p ON c.customer\_id = p.customer\_id WHERE p.payment id IS NULL;

### Output:



- 21. What is the most popular film (rented the most)?
  - → Query: SELECT f.film\_id, f.title, COUNT(r.rental\_id) AS rental\_count FROM film f

JOIN inventory i ON f.film\_id = i.film\_id JOIN rental r ON i.inventory\_id = r.inventory\_id GROUP BY f.film\_id, f.title order by rental\_count desc limit 1;

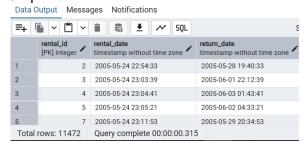
### Output:



- 22. Show all films longer than 2 hours.
  - → Query: SELECT title, length FROM film WHERE length > '120';

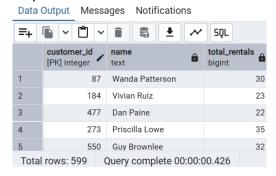


- 23. Find all rentals that were returned late.
  - → Query: SELECT rental\_id, rental\_date, return\_date FROM rental WHERE return\_date > rental\_date + INTERVAL '3 DAYs';



- 24. List customers and the number of films they rented.
  - → Query: SELECT c.customer\_id, CONCAT(c.first\_name, '', c.last\_name) AS name, COUNT(r.rental\_id) AS total\_rentals FROM customer c JOIN rental r ON c.customer\_id = r.customer\_id GROUP BY c.customer id, c.first\_name, c.last\_name;

## Output:



- 25. Write a query to show top 3 rented film categories.
  - → Query: SELECT c.name AS category, COUNT(r.rental\_id) AS total\_rentals FROM rental r

    JOIN inventory i ON r.inventory\_id = i.inventory\_id

    JOIN film f ON i.film\_id = f.film\_id

    JOIN film\_category fc ON f.film\_id = fc.film\_id

    JOIN category c ON fc.category\_id = c.category\_id

    GROUP BY c.name

    ORDER BY total\_rentals DESC

    LIMIT 3;

	category character varying (25)	total_rentals bigint
1	Sports	1179
2	Animation	1166
3	Action	1112

- 26. Create a view that shows all customer names and their payment totals.
  - → Query: CREATE VIEW Customer View AS

**SELECT** 

CONCAT(c.first\_name, ' ', c.last\_name) AS name,

SUM(p.amount) AS total payment

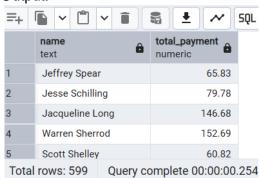
FROM customer c

JOIN payment p ON c.customer id = p.customer id

GROUP BY c.customer id, name;

select \* from Customer View;

#### Output:



- 27. Update a customer's email address given their ID.
  - → Query: UPDATE customer

SET email = 'new\_email@example.com'

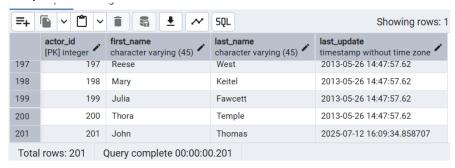
WHERE customer id = 1;

## Output:

UPDATE 1 Query returned successfully in 162 msec.

598	599	2	Austin	Cintron	austin.cintron@sakilacustomer.org
599	1	1	Mary	Smith	new_email@example.com

- 28. Insert a new actor into the actor table.
  - → Query: Insert into actor values('201', 'John', 'Thomas', now()::timestamp(2)); Output:



- 29. Delete all records from the rentals table where return\_date is NULL.
  - → Query: DELETE FROM rental

WHERE return date IS NULL

AND rental\_id NOT IN (SELECT rental\_id FROM payment);

Output: DELETE 0 Query returned successfully in 171 msec.

- 30. Add a new column 'age' to the customer table.
  - → Query: alter table customer add age int default 20 not null;

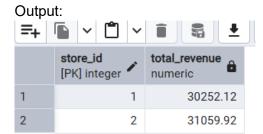
Output:											
1	of	1	I∢	4	4	<b>&gt;&gt;</b>	▶I				
nout time zone	<i>j</i> .	acti		, ·	ag int	<b>e</b> eger	•				
:49:45.738				1			20				
:49:45.738				1			20				
:49:45.738				1			20				
:49:45.738				1			20				
:49:45.738				1			20				
:49:45.738				1			20				
		CRLI	F	Ln	2,	Col	24				

- 31. Create an index on the 'title' column of the film table.
  - → Query: create index film\_title on film(title); Output: relation "film title" already exists
- 32. Find the total revenue generated by each store.
  - → Query: SELECT s.store\_id, SUM(p.amount) AS total\_revenue FROM store s

    JOIN staff st ON s.store\_id = st.store\_id

    JOIN payment p ON st.staff\_id = p.staff\_id

    GROUP BY s.store\_id;



- 33. What is the city with the highest number of rentals?
  - → Query: SELECT ci.city, COUNT(r.rental\_id) AS rental\_count FROM rental r

    JOIN customer c ON r.customer\_id = c.customer\_id

    JOIN address a ON c.address\_id = a.address\_id

    JOIN city ci ON a.city\_id = ci.city\_id

    GROUP BY ci.city

    ORDER BY rental\_count DESC

    LIMIT 1;



- 34. How many films belong to more than one category?
  - → Query: SELECT COUNT(\*) AS multi\_category\_films FROM ( SELECT film\_id FROM film\_category GROUP BY film\_id HAVING COUNT(category id) > 1 );



- 35. List the top 10 actors by number of films they appeared in.
  - → Query: SELECT

    CONCAT(a.first\_name, '', a.last\_name) AS name,

    COUNT(f.film\_id) AS no\_of\_films

    FROM actor a

    JOIN film\_actor fa ON a.actor\_id = fa.actor\_id

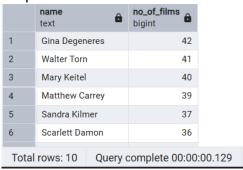
    JOIN film f ON f.film\_id = fa.film\_id

    GROUP BY a.actor\_id, a.first\_name, a.last\_name

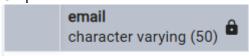
    ORDER BY no of films DESC

LIMIT 10;

# Output:



- 36. Retrieve the email addresses of customers who rented 'Matrix Revolutions'.
  - → Query: SELECT DISTINCT c.email
    FROM customer c
    JOIN rental r ON c.customer\_id = r.customer\_id
    JOIN inventory i ON r.inventory\_id = i.inventory\_id
    JOIN film f ON i.film\_id = f.film\_id
    WHERE f.title = 'Matrix Revolutions';



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

```
→ Query: CREATE FUNCTION get customer payment total(cust id INT)
         RETURNS NUMERIC AS $$
         BEGIN
          RETURN (
           SELECT SUM(amount)
           FROM payment
           WHERE customer id = cust id
          );
         END;
         $$ LANGUAGE plpgsql;
         SELECT get customer payment total(1);
   Output:
            get_customer_payment_total
            numeric
     1
                                 114.70
```

- 38. Begin a transaction that updates stock and inserts a rental record.
  - → Query: BEGIN;

Output: COMMIT

```
UPDATE inventory
SET last_update = CURRENT_TIMESTAMP
WHERE inventory_id = 1;
INSERT INTO rental (rental_date, inventory_id, customer_id, return_date, staff_id, last_update)
VALUES (CURRENT_TIMESTAMP, 1, 1, NULL, 1, CURRENT_TIMESTAMP);
COMMIT;
rollback;
```

Query returned successfully in 81 msec.

- 39. Show the customers who rented films in both 'Action' and 'Comedy' categories.
  - → Query: SELECT c.customer\_id, CONCAT(c.first\_name, ' ', c.last\_name) AS name FROM customer c

JOIN rental r ON c.customer id = r.customer id

JOIN inventory i ON r.inventory id = i.inventory id

JOIN film category fc ON i.film id = fc.film id

JOIN category cat ON fc.category id = cat.category id

WHERE cat.name IN ('Action', 'Comedy')

GROUP BY c.customer id, c.first name, c.last name

HAVING COUNT(DISTINCT cat.name) = 2;

## Output:



- 40. Find actors who have never acted in a film.
  - → Query: SELECT a.actor\_id, CONCAT(a.first\_name, '', a.last\_name) AS name FROM actor a LEFT JOIN film\_actor fa ON a.actor\_id = fa.actor\_id WHERE fa.film id IS NULL;

