# 02. Práctica SQL - DVD Rental

---

Practica a realizar en JupyterLab o notebook. Debido a las conexiones concurrente es preferible utilizar vuestra base de datos DVD Rental desde Heroku que tienen creadas. En caso de no tenerla, pueden seguir el tutorial desde material en Classroom.

1. **¿Tenemos actores en la tabla de actores que comparten el nombre completo y, en caso afirmativo, muestran esos nombres compartidos?**

Para responder a esa pregunta, primero necesitamos saber si tenemos actores en la tabla de actores que comparten sus nombres completos. Entonces comenzar contando la cantidad de actores que comparten sus nombres completos.

```
SELECT
    COUNT(DISTINCT first_name || last_name)
FROM actor;
```

```
SELECT
    DISTINCT a1.first_name, a1.last_name
FROM actor a1 JOIN actor a2
ON a1.actor_id <> a2.actor_id AND a1.first_name = a2.first_name
AND a1.last_name = a2.last_name;
```

2. **Muestra los nombres de los clientes que comparten la misma dirección (por ejemplo, esposo y esposa).**

Para responder a esta pregunta, debemos buscar más de un cliente que tenga el mismo address_id pero diferentes customer_ids. Eso significa que haremos JOIN a la tabla de clientes para sí mismo (*autounión*).

```
SELECT c1.first_name, c1.last_name, c2.first_name, c2.last_name
FROM customer c1 JOIN customer c2
ON c1.customer_id <> c2.customer_id AND c1.address_id =
c2.address_id;
```

3. **Muestra el monto total pagado por todos los clientes en la tabla de pagos.**

Utilizar aquí la función agregada **SUM()** en la columna de monto de la tabla de pagos.

```
SELECT SUM(amount)
FROM payment;
```

4. **Muestre el monto total pagado por cada cliente en la tabla de pagos.**

```
SELECT customer_id, SUM(amount)
FROM payment
GROUP BY customer_id
ORDER BY customer_id;
```

5. **¿Cuál es el pago total más alto realizado?**

```
SELECT SUM(amount) AS total_payments
FROM payment
GROUP BY customer_id
ORDER BY SUM(amount) DESC LIMIT 1;
```

**6. ¿Cuál es el nombre del cliente que realizó los pagos totales más altos?**

```
SELECT first_name, last_name
FROM customer
WHERE customer_id IN
    (SELECT customer_id
     FROM payment
     GROUP BY customer_id
     HAVING SUM(amount) =
         (SELECT SUM(amount)
          FROM payment
          GROUP BY customer_id
          ORDER BY SUM(amount) DESC LIMIT 1));
```

**7. ¿Cuáles son las películas que más se alquilaron?**

```
SELECT film_id, count(film_id)
FROM rental R JOIN inventory I
ON R.inventory_id = I.inventory_id
GROUP BY film_id
ORDER BY count(film_id) DESC;
```

```sql
SELECT count(film_id)
FROM rental R JOIN inventory I
ON R.inventory_id = I.inventory_id
GROUP BY film_id
ORDER BY count(film_id) DESC LIMIT 1;
```

```sql
SELECT title
FROM film
WHERE film_id IN
    (SELECT film_id
    FROM rental R JOIN inventory I
    ON R.inventory_id = I.inventory_id
    GROUP BY film_id
    HAVING count(film_id) =
        (SELECT count(film_id)
        FROM rental R JOIN inventory I
        ON R.inventory_id = I.inventory_id
        GROUP BY film_id
        ORDER BY count(film_id) DESC LIMIT 1));
```

**8. Qué películas se han alquilado hasta ahora.**

Debemos entender que no es necesario que ya se hayan alquilado todas las películas de la tabla de películas.

```sql
SELECT title
FROM film
WHERE film_id IN
    (SELECT distinct film_id
    FROM rental JOIN inventory
    ON rental.inventory_id = inventory.inventory_id);
```

```sql
SELECT count(title)
FROM film
WHERE film_id IN
    (SELECT distinct film_id
    FROM rental JOIN inventory
    ON rental.inventory_id = inventory.inventory_id);
```

**9. Qué películas no se han alquilado hasta ahora.**

```sql
SELECT title
FROM film
WHERE film_id NOT IN
    (SELECT distinct film_id
    FROM rental JOIN inventory
    ON rental.inventory_id = inventory.inventory_id);
```

```sql
SELECT count(title)
FROM film
WHERE film_id NOT IN
    (SELECT distinct film_id
    FROM rental JOIN inventory
    ON rental.inventory_id = inventory.inventory_id);
```

**10. Qué clientes no han alquilado ninguna película hasta ahora.**

Un cliente podría haberse registrado como cliente del lugar de alquiler de DVD, pero aún no ha comenzado a alquilar DVD.

```sql
/* solución NOT IN */
SELECT customer_id
FROM customer
WHERE customer_id NOT IN
    (SELECT distinct customer_id
    FROM rental);
```

```sql
SELECT first_name, last_name
FROM customer
WHERE customer_id NOT IN
    (SELECT DISTINCT customer_id
    FROM rental);
```

```sql
/* solución NOT EXISTS */
SELECT customer_id
FROM customer
WHERE NOT EXISTS
    (SELECT *
    FROM rental
    WHERE customer.customer_id = rental.customer_id);
```

```sql
SELECT first_name, last_name
FROM customer C
WHERE NOT EXISTS
    (SELECT DISTINCT customer_id
    FROM rental R
    WHERE C.customer_id = R.customer_id);
```

**11. Muestra cada película y la cantidad de veces que se alquila.**

```sql
SELECT film_id, COUNT(film_id)
FROM rental JOIN inventory
ON rental.inventory_id = inventory.inventory_id
GROUP BY film_id
ORDER BY film_id;
```

**12. Muestra la cantidad de películas en las que actuó cada actor.**

```sql
SELECT actor_id, count(film_id)
FROM film_actor
GROUP BY actor_id
ORDER BY actory_id;
```

```sql
SELECT first_name, last_name, COUNT(film_actor.film_id)
FROM film_actor JOIN actor
ON film_actor.actor_id = actor.actor_id
GROUP BY (first_name, last_name);
```

**13. Muestre los nombres de los actores que actuaron en más de 20 películas.**

```sql
SELECT first_name, last_name, COUNT(film_actor.film_id)
FROM film_actor JOIN actor
ON film_actor.actor_id = actor.actor_id
GROUP BY (first_name, last_name)
HAVING COUNT(film_actor.film_id) > 20;
```

**14. ¿Cuántos actores tienen 8 letras solo en sus nombres?**

```
SELECT count(first_name)
FROM actor
WHERE LENGTH(first_name) = 8;
```

**15.Para todas las películas clasificadas como "PG", muéstrame la película y la cantidad de veces que se alquiló.**

Escribir una subconsulta para seleccionar todos los film_ids que se han alquilado hasta ahora. Para eso, se necesita unir tablas de alquiler e inventario en la columna común Inventory_id.

```
SELECT inventory.film_id, COUNT(inventory.film_id)
FROM rental JOIN inventory
ON rental.inventory_id = inventory.inventory_id
JOIN film
ON inventory.film_id = film.film_id
WHERE rating = 'PG' /*comprobar "" o '' */
GROUP BY inventory.film_id
ORDER BY inventory.film_id;
```

**16. Muestra las películas que se ofrecen para alquilar en store_id 1 y no se ofrecen en store_id 2.**

```
SELECT count(film_id)
FROM inventory
WHERE store_id = 1 AND film_id NOT IN
    (SELECT film_id
    WHERE inventory
    WHERE store_id = 2);
```

**17. Muestre las películas que se ofrecen para alquilar en cualquiera de las dos tiendas 1 y 2.**

```
(SELECT film_id
FROM inventory
WHERE store_id = 1)
UNION
(SELECT film_id
FROM inventory
WHERE store_id = 2);
```

**18. Muestra los títulos de las películas que se ofrecen en ambas tiendas al mismo tiempo**

```
/* sol 1*/
SELECT title
FROM film
WHERE film_id IN
    (SELECT film_id
    FROM inventory
    WHERE store_id = 1 AND film_id IN
        (SELECT film_id
        FROM inventory
        WHERE store_id = 2));
```

```
/*sol 2*/
SELECT title
FROM film
WHERE film_id IN
    (SELECT film_id
    FROM inventory
    WHERE store_id = 1)
    INTERSET
    (SELECT film_id
        FROM inventory
        WHERE store_id = 2));
```

```
/*sol 3*/
SELECT title
FROM film
WHERE film_id IN
    (SELECT film_id
    FROM inventory I1
    WHERE store_id = 1 AND EXISTS
        (SELECT film_id
            FROM inventory I2
            WHERE store_id = 2 AND I1.film_id = I2.film_id));
```

**19. Para cada tienda, muestre el número de clientes que son miembros de esa tienda**

```sql
SELECT store_id, COUNT(customer_id)
FROM customer
GROUP BY store_id;
```

**20. Muestra el título de la película más alquilada en la tienda con store_id 1**

```sql
SELECT film_id, count(film_id)
FROM inventory
WHERE store_id = 1
GROUP BY film_id
ORDER BY count(film_id) DESC;
```

```sql
/* finalmente */
SELECT title
FROM film
WHERE film_id IN
    (SELECT film_id
    FROM rental R JOIN inventory I
    ON R.inventory_id = I.inventory_id
    WHERE store_id = 1
    GROUP BY film_id
    HAVING COUNT(film_id) =
        (SELECT COUNT(film_id)
        FROM rental R JOIN inventory I
        ON R.inventory_id = I.inventory_id
        WHERE store_id = 1
        GROUP BY film_id
        ORDER BY COUNT(film_id) DESC LIMIT 1));
```

**21. Cuántas películas aún no se ofrecen para alquilar en las tiendas. Hay dos tiendas solo 1 y 2**

```sql
(SELECT COUNT(DISTINCT film_id)
FROM
    ((SELECT film_id
     FROM inventory
     WHERE store_id = 1)
    UNION
    (SELECT film_id
    FROM inventory
    WHERE store_id = 2)) temp);
```

**22. Muestre los customer_id para aquellos clientes que alquilaron un DVD de películas más de una vez.**

Para responder a esta pregunta, primero debemos conocer los film_ids que alquiló cada cliente.

```
SELECT rental_id, rental_date, customer_id, film_id
FROM rental JOIN inventory
ON rental.inventory_id = inventory.inventory_id
```

```
/* Subset WITH */
WITH TEMP AS
(SELECT rental_id, rental_date, customer_id, film_id
FROM rental JOIN inventory
ON rental.inventory_id = inventory.inventory_id)

SELECT T1.customer_id, count(T1.film_id)
FROM TEMP T1 JOIN TEMP T2
ON T1.customer_id = T2.customer_id AND
    T1.film_id = T2.film_id AND
    T1.rental_date <> T2.rental.date
GROUP BY T1.customer_id
HAVING COUNT(T1.film_id) > 1;
```

**23. Muestra la cantidad de películas alquiladas en cada clasificación.**

Para responder a esta pregunta, primero debemos conocer los ID de película que se alquilaron (unir tablas de alquiler e inventario) y las calificaciones de cada película (unir tabla de películas).

```
SELECT film.rating, COUNT(inventory.film_id)
FROM rental JOIN inventory
ON rental.inventory_id = inventory.inventory_id
RIGHT JOIN film
ON inventory.film_id = film.film_id
GROUP BY film_rating
ORDER BY COUNT(inventory.film_id) DESC;
```

**24. Muestre el beneficio de cada una de las tiendas 1 y 2.**

Para responder a esta pregunta, necesitamos saber la cantidad pagada y el Inventory_id para cada película alquilada al unirse (tablas de alquiler y pago) y el store_id para esa transacción de alquiler (unirse a la tabla de inventario).

```
SELECT store_id, SUM(amount)
FROM payment JOIN rental
ON payment.rental_id = rental.rental_id
JOIN inventory
ON rental.inventory_id = inventory.inventory_id
GROUP BY store_id
ORDER BY count(*) DESC;
```

**25. Muestra el beneficio de cada una de las tiendas 1 y 2 seguido del beneficio total de ambas tiendas**

```
SELECT store_id, SUM(amount)
FROM payment JOIN rental
ON payment.rental_id = rental.rental_id
JOIN inventory
ON rental.inventory_id = inventory.inventory_id
GROUP BY ROLLUP (store_id)
ORDER BY count(*) DESC;
```

**26. Cuente la cantidad de actores cuyos nombres no comienzan con una "A"**

```
SELECT count(*)
FROM actor
WHERE first_name NOT LIKE 'A%';
```

**27. Busque el nombre del actor que comience con "P" seguido de (una *e* o una *a*) y luego cualquier otra letra**

```
SELECT *
FROM actor
WHERE first_name SIMILAR TO ' P(e|a)%';
```

**28. Busque nombres de clientes que comiencen con "P" seguido de 5 letras**

```sql
SELECT first_name
FROM customer
WHERE first_name SIMILAR TO 'P_____';
```

```sql
/* Sol2 */
SELECT first_name
FROM customer
WHERE first_name SIMILAR TO 'P[a-z]{5}'
```

**29. Busque actores con PaRkEr como su nombre e ignore las letras mayúsculas y minúsculas. Luego seleccione actores llamados PaRkEr y haga coincidir el caso de la letra**

```sql
SELECT *
FROM actor
WHERE first_name ~* 'PaRkEr';
```

```sql
SELECT *
FROM actor
WHERE first_name ilike 'PaRkEr';
```

```sql
SELECT *
FROM actor
WHERE first_name like 'PaRkEr';
```

```sql
SELECT *
FROM actor
WHERE first_name = 'PaRkEr';
```

**30. Busque nombres de actores que comiencen con "P" seguidos de cualquier letra de la a a la e y luego de cualquier otra letra**

```sql
SELECT *
FROM actor
WHERE first_name SIMILAR TO 'P[a-e]%';
```