Consultas a la Base de Datos

Consultas previas para comprobar que incluye Sakila

Para iniciar la ronda de consultas, nos ubicamos en un lienzo nuevo

USE Sakila;

Para que el sistema entienda sobre que base de datos vamos a realizar las consultas.

De esta manera trabajamos sobre la base de datos Sakila de manera directa.

Luego podemos **Verificar las tablas**

SHOW TABLES;

Realizar consultas, por ejemplo, ver los primeros 5 actores

```
SELECT * FROM actor LIMIT 5;
```

A. Nivel inicial:

1. **Listar todas las películas**:
   Nos muestra título y año de lanzamiento de todas las películas en la base de datos.

   -- 1. Listar todas las películas

   -- Esta consulta selecciona todas las películas y sus años de lanzamiento.

   -- a) Selecciona los campos 'title' y 'release_year' de la tabla 'film'.

   SELECT title, release_year FROM film;

2. **Obtener clientes activos:**
   Nos generará una lista con nombre y apellidos de los clientes que tienen en estado la indicación de activos.

   -- 2. Obtener clientes activos

   -- Esta consulta obtiene los nombres y apellidos de los clientes activos.

   -- a) Se seleccionan los campos 'first_name' y 'last_name' de la tabla 'customer'.

   -- b) El WHERE filtra aquellos clientes cuyo campo 'active' es igual a 1.

   SELECT first_name, last_name FROM customer WHERE active = 1;

3. **Mostrar las películas de categoría ‘Action’**:
   Buscará los títulos de las películas que pertenecen a la categoría pedida, en este caso Action.

   -- 3. Películas de categoría 'Action'

   -- Esta consulta busca todas las películas que pertenecen a la categoría 'Action'.

```
-- Paso a paso:

-- a) Se selecciona el título de las películas desde la tabla 'film' con el alias 'f'. Colocar ‘f’ como alias de film es como colocar film.title.

-- ‘f’, ‘fc’, ‘c’ son alias para simplificar las consultas y mejorar la legibilidad del código.

-- b) Se hace un INNER JOIN con la tabla 'film_category' para relacionar las películas con sus categorías a través del campo 'film_id'.

-- c) Se hace otro INNER JOIN con la tabla 'category' para obtener el nombre de la categoría mediante 'category_id'.

-- d) En el WHERE, se filtran las películas que tienen la categoría 'Action'.

SELECT f.title

FROM film f

INNER JOIN film_category fc ON f.film_id = fc.film_id

INNER JOIN category c ON fc.category_id = c.category_id

WHERE c.name = 'Action';

SELECT f.title FROM film f

INNER JOIN film_category fc ON f.film_id = fc.film_id

INNER JOIN category c ON fc.category_id = c.category_id

WHERE c.name = 'Action';
```

4. **Listar actores por orden alfabético**:
   Ordena los actores por apellido de forma ascendente.

```
-- 4. Listado de actores por orden alfabético

-- Esta consulta lista a los actores en orden alfabético por apellido.

-- a) Se seleccionan los nombres y apellidos de la tabla 'actor'.

-- b) Se ordenan los resultados en orden ascendente por el campo 'last_name'.

SELECT first_name, last_name FROM actor ORDER BY last_name ASC;
```

B. Nivel Intermedio

1. **Listar las películas alquiladas por clientes:**
   Muestra el nombre del cliente, el título de la película y la fecha del alquiler.

```
-- 1. Películas alquiladas por cliente

-- Esta consulta muestra las películas alquiladas por cada cliente.

-- a) Se seleccionan los nombres del cliente, el título de la película y la fecha de alquiler.
```

```
-- b) Se conectan las tablas rental, customer, inventory y film a través de sus IDs relacionados.

SELECT c.first_name, c.last_name, f.title, r.rental_date

FROM rental r

INNER JOIN customer c ON r.customer_id = c.customer_id

INNER JOIN inventory i ON r.inventory_id = i.inventory_id

INNER JOIN film f ON i.film_id = f.film_id;
```

2. **Películas sin alquiler en los últimos 3 meses:**
   Identifica las películas que no han sido alquiladas en los últimos 90 días.

```
-- 2. Películas sin alquiler en los últimos 3 meses

-- Esta consulta identifica las películas que no han sido alquiladas en los últimos 90 días.

-- a) Se utiliza LEFT JOIN para incluir todas las películas, incluso si no tienen alquileres recientes.

-- b) Se aplica una condición en el WHERE para verificar películas con alquileres nulos o anteriores a 90 días.

SELECT f.title

FROM film f

LEFT JOIN inventory i ON f.film_id = i.film_id

LEFT JOIN rental r ON i.inventory_id = r.inventory_id

WHERE r.rental_date IS NULL OR r.rental_date < NOW() - INTERVAL 90 DAY;
```

3. **Top o ranking de 5 clientes por cantidad de alquileres:**
   Muestra el listado de los 5 clientes que han alquilado más películas, incluyendo el total de alquileres.

```
-- 3. Top 5 clientes por cantidad de alquileres

-- Esta consulta muestra los 5 clientes que más películas han alquilado.

-- a) Se utiliza COUNT() para contar los alquileres por cliente.

-- b) Se agrupan los resultados por cliente y se ordenan en orden descendente.

-- c) Se limita el resultado a los 5 primeros clientes.

SELECT c.first_name, c.last_name, COUNT(r.rental_id) AS total_rentals

FROM rental r
```

```
INNER JOIN customer c ON r.customer_id = c.customer_id
GROUP BY c.customer_id
ORDER BY total_rentals DESC
LIMIT 5;
```

4. **Ingresos por categoría:**
   Calcula el total recaudado por cada categoría de película.

```
-- 4. Ingresos por categoría
-- Esta consulta calcula los ingresos totales por categoría de película.
-- a) Se suma el monto de cada pago utilizando SUM().
-- b) Se agrupan los resultados por categoría, identificando cada categoría mediante su nombre.
SELECT c.name AS category, SUM(p.amount) AS total_revenue
FROM payment p
INNER JOIN rental r ON p.rental_id = r.rental_id
INNER JOIN inventory i ON r.inventory_id = i.inventory_id
INNER JOIN film f ON i.film_id = f.film_id
INNER JOIN film_category fc ON f.film_id = fc.film_id
INNER JOIN category c ON fc.category_id = c.category_id
GROUP BY c.category_id;
```

5. **Clientes con más de 3 alquileres pendientes**
   Son los clientes que no han devuelto más de 3 películas alquiladas.

```
-- 5. Clientes con más de 3 alquileres pendientes
-- Esta consulta utiliza COUNT() para contar los alquileres pendientes (donde return_date es NULL) y HAVING para filtrar los clientes con más de 3 alquileres pendientes.
SELECT c.first_name, c.last_name, COUNT(r.rental_id) AS pending_rentals
FROM rental r
INNER JOIN customer c ON r.customer_id = c.customer_id
WHERE r.return_date IS NULL
GROUP BY c.customer_id
```

```
HAVING pending_rentals > 3;
```

6. **Comparar ingresos por ciudad**
   Compara los ingresos generados en cada tienda, mostrando la ciudad correspondiente.

```
-- 6. Comparar ingresos por ciudad

-- Esta consulta agrupa por ciudad utilizando la tabla address y calcula el total de ingresos por ciudad mediante SUM() aplicado sobre los montos de los pagos.

SELECT a.city, SUM(p.amount) AS total_revenue

FROM payment p

INNER JOIN rental r ON p.rental_id = r.rental_id

INNER JOIN customer c ON r.customer_id = c.customer_id

INNER JOIN address a ON c.address_id = a.address_id

GROUP BY a.city;
```

7. **Clientes con películas alquiladas, pendientes de devolver**
   Muestra los clientes que actualmente tienen películas alquiladas, indicando los títulos y las fechas estimadas de devolución.

```
-- 7. Clientes con películas en stock

-- Esta consulta identifica a los clientes que aún tienen películas sin devolver (return_date es NULL) y muestra los títulos y fechas estimadas de devolución.

SELECT c.first_name, c.last_name, f.title, r.return_date

FROM rental r

INNER JOIN customer c ON r.customer_id = c.customer_id

INNER JOIN inventory i ON r.inventory_id = i.inventory_id

INNER JOIN film f ON i.film_id = f.film_id

WHERE r.return_date IS NULL;
```