|  |  |
| --- | --- |
| **3.Encuentra los nombres de los actores que tengan un “actor\_id” entre 30 y 40.** | Utilizamos SELECT para extraer las columnas first\_name, last\_name y actor id, de la tabla actor. El WHERE filtra los resultados para que solo se seleccionen aquellos registros cuyo actor\_id esté dentro del rango especificado (de 30 a 40, ambos inclusive) mediante la expresión BETWEEN. Esto evita tener que usar múltiples comparaciones AND. |
| **4.Encuentra las películas cuyo idioma coincide con el idioma original.** | Aquí, seleccionamos los títulos de películas (title) desde la tabla film. La condición en WHERE filtra las filas donde el valor de la columna language\_id (idioma actual de la película) sea igual al valor de original\_language\_id (idioma original). Esta comparación simple = garantiza que solo se devuelvan las películas cuyo idioma original coincide con el actual. |
| **5.Ordena las películas por duración de forma ascendente.** | La consulta selecciona los títulos y las duraciones de las películas (title, length) desde la tabla film. La cláusula ORDER BY clasifica los resultados según la duración de las películas (length) en orden ascendente (ASC), de menor a mayor duración. Si no se especifica ASC o DESC, por defecto se ordena de forma ascendente. |
| **6.Encuentra el nombre y apellido de los actores que tengan ‘Allen’ en su apellido.** | Usamos la cláusula SELECT para extraer las columnas first\_name y last\_name de la tabla actor. El filtro WHERE last\_name LIKE '%Allen%' utiliza un patrón de búsqueda con comodines. El símbolo % es un comodín que coincide con cualquier secuencia de caracteres, por lo que este patrón busca cualquier apellido que contenga la subcadena "Allen" en cualquier posición. |
| **7. Encuentra la cantidad total de películas en cada clasificación de la tabla “film” y muestra la clasificación junto con el recuento.** | Se seleccionan la clasificación (rating) y el recuento de películas (COUNT(\*)) de la tabla film. COUNT(\*) cuenta todas las filas en cada grupo. La cláusula GROUP BY rating agrupa los resultados por cada valor único de la columna rating, lo que permite contar cuántas películas pertenecen a cada clasificación. |
| **7a.Suma el total de películas en todas las clasificaciones.** | La función de agregación COUNT(\*) cuenta todas las filas en la tabla film, devolviendo el total de películas. No se necesita GROUP BY porque queremos un solo valor: el recuento total de todas las películas sin agruparlas por ninguna clasificación. |
| **8.Encuentra el título de todas las películas que son ‘PG-13’ o tienen una duración mayor a 3 horas en la tabla film.** | Aquí, se seleccionan los títulos (title) de la tabla film donde se cumple una de dos condiciones: la clasificación (rating) es 'PG-13' o la duración (length) es mayor a 180 minutos (3 horas). La cláusula OR permite seleccionar películas que cumplan al menos una de las dos condiciones. |
| **8a.Recuento de películas que son ‘PG-13’ o duran más de 3 horas.** | En lugar de seleccionar títulos, usamos COUNT(\*) para contar cuántas películas cumplen con alguna de las dos condiciones: clasificación PG-13 o duración mayor a 3 horas. La cláusula WHERE con OR garantiza que al menos una de las condiciones se cumpla para que una película sea contada. |
| **9.Encuentra la variabilidad de lo que costaría reemplazar las películas.** | Se utilizan dos funciones de agregación: AVG() para calcular el costo promedio de reemplazo de las películas (replacement\_cost), y STDDEV() para calcular la desviación estándar, que mide la dispersión de los costos en relación al promedio. Ambas columnas se obtienen de la tabla film. |
| **10.Encuentra la mayor y menor duración de una película.** | Se utilizan las funciones de agregación MAX() y MIN() para obtener la mayor y menor duración (length) de las películas en la tabla film. MAX() devuelve el valor más alto y MIN() devuelve el más bajo de la columna length. |
| **11.Encuentra lo que costó el antepenúltimo alquiler ordenado por día.** | Se selecciona el costo del alquiler (rental\_rate) de la tabla rental. La cláusula ORDER BY rental\_date DESC ordena los alquileres por fecha (rental\_date) en orden descendente (del más reciente al más antiguo). LIMIT 1 asegura que solo obtendremos un resultado, y OFFSET 2 omite los dos primeros (último y penúltimo), devolviendo el antepenúltimo alquiler. |
| **12.Encuentra el título de las películas en la tabla “film” que no sean ni ‘NC17’ ni ‘G’ en cuanto a su clasificación.** | Se selecciona el título (title) de la tabla film. La cláusula WHERE con NOT IN se utiliza para excluir películas con clasificación 'NC17' y 'G'. NOT IN verifica que el valor de la columna rating no esté en el conjunto de valores proporcionados ('NC17', 'G').  lo primero es verificar qué valores son válidos para la columna rating. Puedes obtener los valores disponibles en la columna rating de la tabla film con el siguiente script: SELECT DISTINCT rating FROM film; Y encontré que la clasificación es NC-17. |
| **13.Encuentra el promedio de duración de las películas para cada clasificación de la tabla film y muestra la clasificación junto con el promedio de duración.** | Este script selecciona la clasificación (rating) y el promedio de duración (AVG(length)) de las películas. La función AVG() calcula el promedio de la columna length. Se utiliza GROUP BY rating para agrupar los resultados por cada clasificación, permitiendo así calcular el promedio de duración de las películas dentro de cada grupo de clasificación. |
| **14. Encuentra el título de todas las películas que tengan una duración mayor a 180 minutos.** | Este script selecciona el título (title) de la tabla film. La cláusula WHERE se utiliza para filtrar las películas que tienen una duración (length) superior a 180 minutos. Esto devolverá únicamente los títulos de las películas que cumplen con esta condición. |
| **15. ¿Cuánto dinero ha generado en total la empresa?** | Este script calcula el total de ingresos generados por la empresa al sumar (SUM) los valores de la columna amount en la tabla payment. El resultado se presenta con el alias total\_revenue, que representa la suma total de dinero recaudado por alquileres de películas. |
| **16. Muestra los 10 clientes con mayor valor de id.** | Este script selecciona todas las columnas (\*) de la tabla customer. La cláusula ORDER BY customer\_id DESC ordena los resultados en orden descendente según el customer\_id, asegurando que los clientes con el mayor customer\_id aparezcan primero. La cláusula LIMIT 10 restringe los resultados a los 10 primeros registros, mostrando así los 10 clientes con el mayor customer\_id. |
| **17. Encuentra el nombre y apellido de los actores que aparecen en la película con título ‘Egg Igby’.** | Este script selecciona el nombre (first\_name) y el apellido (last\_name) de los actores de la tabla actor. Se utiliza una cláusula JOIN para combinar la tabla actor con la tabla intermedia film\_actor, emparejando los actor\_id. Luego, se hace otro JOIN con la tabla film para relacionar las películas con los actores, utilizando film\_id. La cláusula WHERE filtra los resultados para que solo se incluyan aquellos actores que aparecen en la película con el título 'Egg Igby'. |
| **18. Selecciona todos los nombres de las películas únicos.** | Este script selecciona los títulos (title) de la tabla film utilizando la cláusula DISTINCT. Esto garantiza que cada título devuelto sea único, eliminando cualquier duplicado que pudiera existir en la base de datos. El resultado será una lista de títulos de películas sin repeticiones. |
| **19. Encuentra el título de las películas que son comedias y tienen una duración mayor a 180 minutos en la tabla “film”.** | El script selecciona los títulos (title) de la tabla film que cumplen con dos condiciones: tener una duración mayor a 180 minutos (f.length > 180) y pertenecer al género Comedy (a través del JOIN con las tablas film\_category y category). Esto permite filtrar las películas tanto por género como por duración. |
| **20.Encuentra las categorías de películas que tienen un promedio de duración superior a 110 minutos y muestra el nombre de la categoría junto con el promedio de duración.** | Este script selecciona el nombre de la categoría (c.name) y el promedio de duración (AVG(f.length)) de las películas en esa categoría. Se realiza un JOIN entre las tablas film, film\_category, y category para relacionar las películas con sus categorías. La cláusula GROUP BY c.name agrupa las películas por categoría, y la cláusula HAVING AVG(f.length) > 110 filtra las categorías para mostrar solo aquellas cuyo promedio de duración es mayor a 110 minutos. |
| **21.¿Cuál es la media de duración del alquiler de las películas?** | Este script calcula el promedio (AVG) de la duración del alquiler (rental\_duration) de todas las películas en la tabla film. La consulta selecciona la media de la columna rental\_duration, devolviendo el promedio como average\_rental\_duration. |
| **22. Crea una columna con el nombre y apellidos de todos los actores y**  **actrices.** | Este script selecciona el nombre completo de cada actor o actriz concatenando las columnas first\_name (nombre) y last\_name (apellido) con la función CONCAT(). El espacio (' ') asegura que los nombres y apellidos estén separados en la columna resultante llamada full\_name. |
| **23.** **Números de alquiler por día, ordenados por cantidad de alquiler de forma descendente.** | El script selecciona la fecha del alquiler (rental\_date) y cuenta cuántos alquileres ocurrieron en esa fecha usando COUNT(\*). La consulta agrupa los alquileres por fecha con GROUP BY rental\_date y luego ordena los resultados por la cantidad de alquileres (rental\_count) en orden descendente (ORDER BY rental\_count DESC). |
| **24.** **Encuentra las películas con una duración superior al promedio.** | El script selecciona los títulos (title) y la duración (length) de las películas desde la tabla film. Utiliza una subconsulta (SELECT AVG(length) FROM film) para calcular el promedio de duración de todas las películas. Luego, la cláusula WHERE length > filtra las películas cuya duración es mayor a ese promedio. |
| **25. Averigua el número de alquileres registrados por mes.** | El script extrae el mes de la fecha del alquiler (rental\_date) usando EXTRACT(MONTH FROM rental\_date). Luego agrupa los registros por mes (GROUP BY month) y cuenta la cantidad de alquileres por cada mes (COUNT(\*) AS rental\_count). Los resultados se ordenan por mes (ORDER BY month). |
| **26. Encuentra el promedio, la desviación estándar y varianza del total**  **pagado.** | El script selecciona el **promedio** (AVG()), la **desviación estándar** (STDDEV()) y la **varianza** (VAR\_SAMP/ VARIANCE) del total pagado. Estas tres funciones calculan las estadísticas de la columna amount de la tabla payment, que contiene los valores de los pagos realizados. La consulta devuelve el promedio como average\_amount, la desviación estándar como stddev\_amount, y la varianza como variance\_amount. |
| **27. ¿Qué películas se alquilan por encima del precio medio?** | El script selecciona los títulos (title) y el precio de alquiler (rental\_rate) de las películas en la tabla film. Utiliza una subconsulta (SELECT AVG(rental\_rate) FROM film) para calcular el precio medio de alquiler. Luego, la cláusula WHERE rental\_rate > filtra las películas cuyo precio de alquiler es superior a ese promedio. |
| **28. Muestra el id de los actores que hayan participado en más de 40**  **películas.** | Este script selecciona el **actor\_id** de la tabla film\_actor. Agrupa los resultados por actor\_id usando GROUP BY, y luego utiliza HAVING para filtrar los actores cuyo conteo de film\_id (es decir, la cantidad de películas en las que ha participado) es mayor que 40. |
| **29. Obtener todas las películas y, si están disponibles en el inventario, mostrar la cantidad disponible.** | En este script, se selecciona el título de cada película (f.title) de la tabla film. La subconsulta (SELECT COUNT(\*) FROM inventory i WHERE i.film\_id = f.film\_id) cuenta las filas en la tabla inventory que coinciden con el film\_id de la película actual. Esto devuelve la cantidad disponible de cada película. Si no hay coincidencias, devolverá NULL.  En este script, se selecciona el título de cada película (f.title) de la tabla film. Se utiliza un **LEFT JOIN** para unir film e inventory en función del film\_id. Luego, se cuenta la cantidad de inventory\_id para cada película en la cláusula COUNT(), que proporciona la cantidad disponible. Se agrupa por el título para asegurarse de que cada título tenga su propia fila en los resultados. |
| **30. Obtener los actores y el número de películas en las que ha actuado.** | Este script selecciona el ID del actor (a.actor\_id) y su nombre completo (CONCAT(a.first\_name, ' ', a.last\_name)), creando un nuevo alias actor\_name. Luego, cuenta cuántas películas ha actuado (COUNT(fa.film\_id) AS number\_of\_films) a partir de la tabla film\_actor. Se usa un LEFT JOIN para incluir a todos los actores, incluso aquellos que no han actuado en ninguna película. Se agrupa por actor\_id y actor\_name, y se ordena el resultado por number\_of\_films en orden descendente. |
| **31. Obtener todas las películas y mostrar los actores que han actuado en ellas, incluso si algunas películas no tienen actores asociados.** | Este script selecciona los títulos de todas las películas (f.title) de la tabla film y realiza una **LEFT JOIN** con la tabla film\_actor para obtener las relaciones entre las películas y los actores. Luego, se hace otro **LEFT JOIN** con la tabla actor para obtener los nombres completos de los actores (CONCAT(a.first\_name, ' ', a.last\_name)). Al usar **LEFT JOIN**, garantizamos que todas las películas se mostrarán, incluso si no tienen actores asociados. El resultado se ordena por el título de la película y los nombres de los actores. |
| **32. Obtener todos los actores y mostrar las películas en las que han actuado, incluso si algunos actores no han actuado en ninguna película.** | Este script selecciona los nombres completos de los actores (CONCAT(a.first\_name, ' ', a.last\_name)), y usa una **LEFT JOIN** con film\_actor para obtener las relaciones entre actores y películas. Después, hace otra **LEFT JOIN** con film para obtener los títulos de las películas (f.title). El **LEFT JOIN** asegura que todos los actores se muestren, aunque no hayan actuado en ninguna película. El resultado se ordena por el nombre del actor y, dentro de cada actor, por el título de las películas en las que ha participado. |
| **33. Obtener todas las películas que tenemos y todos los registros de**  **alquiler.** | Este script selecciona los títulos de todas las películas (f.title) de la tabla film. Se usa un **LEFT JOIN** con la tabla inventory para vincular las películas con el inventario y luego un **LEFT JOIN** con la tabla rental para obtener los registros de alquiler, incluidos rental\_id, rental\_date, return\_date y el customer\_id que realizó el alquiler. Al usar **LEFT JOIN**, garantizamos que todas las películas se muestren, incluso aquellas que no han sido alquiladas. El resultado se ordena por el título de la película y la fecha de alquiler. |
| **34. Encuentra los 5 clientes que más dinero se hayan gastado con nosotros.** | Este script selecciona el **ID del cliente** (c.customer\_id) y su **nombre completo** (CONCAT(c.first\_name, ' ', c.last\_name)). Luego, suma todos los pagos realizados por cada cliente (SUM(p.amount) AS total\_spent) a partir de la tabla payment. Usamos **JOIN** para relacionar las tablas customer y payment. El resultado se agrupa por el ID y el nombre del cliente, se ordena de mayor a menor gasto (ORDER BY total\_spent DESC), y se limita a los 5 clientes con mayor gasto (LIMIT 5). |
| **35. Selecciona todos los actores cuyo primer nombre es 'Johnny'** | Este script selecciona el **ID del actor** (actor\_id), su **primer nombre** (first\_name), y su **apellido** (last\_name) de la tabla actor. Se usa la cláusula **WHERE** para filtrar solo aquellos actores cuyo primer nombre es exactamente 'JOHNNY'. En mayúscula si o si. |
| **36. Renombra la columna “first\_name” como Nombre y “last\_name” como Apellido.** | Este script selecciona el **ID del actor** (actor\_id), renombra la columna first\_name como **Nombre** y la columna last\_name como **Apellido** en el resultado. La cláusula **AS** se utiliza para cambiar los nombres de las columnas solo en el resultado final de la consulta, sin modificar la estructura de la tabla original. |
| **37. Encuentra el ID del actor más bajo y más alto en la tabla actor.** | Este script utiliza las funciones agregadas **MIN()** y **MAX()** para encontrar el menor y el mayor valor de actor\_id en la tabla actor. El alias **AS** se usa para renombrar las columnas resultantes como actor\_id\_minimo y actor\_id\_maximo.  Si se quieren saber los nombres: Este script selecciona el **ID del actor**, su **primer nombre** (first\_name) y **apellido** (last\_name) de la tabla actor. Utiliza dos subconsultas para obtener el **ID mínimo** (MIN(actor\_id)) y el **ID máximo** (MAX(actor\_id)) de la tabla actor, y los combina con el operador **OR** para devolver los actores con esos IDs. |
| **38. Cuenta cuántos actores hay en la tabla “actor”.** | Este script utiliza la función **COUNT(\*)** para contar el número total de filas en la tabla actor, lo que corresponde al número de actores en la tabla. El alias **AS total\_actores** se usa para darle un nombre más legible a la columna de resultados. |
| **39. Selecciona todos los actores y ordénalos por apellido en orden ascendente.** | Este script selecciona el **ID del actor**, su **primer nombre** (first\_name), y su **apellido** (last\_name) de la tabla actor. Utiliza **ORDER BY last\_name ASC** para ordenar los resultados alfabéticamente por el apellido en **orden ascendente**. |
| **40. Selecciona las primeras 5 películas de la tabla “film”.** | Este script selecciona **todas las columnas** de la tabla film utilizando SELECT \*, y aplica **LIMIT 5** para restringir el resultado a las primeras 5 filas de la tabla. |
| **41. Agrupa los actores por su nombre y cuenta cuántos actores tienen el mismo nombre. ¿Cuál es el nombre más repetido?** | Este script selecciona el **primer nombre** (first\_name) de los actores y cuenta cuántos actores tienen ese nombre utilizando **COUNT(\*)**. La cláusula **GROUP BY first\_name** agrupa los resultados por nombre, y **ORDER BY cantidad DESC** ordena los grupos de mayor a menor según el número de actores. |
| **42. Encuentra todos los alquileres y los nombres de los clientes que los realizaron.** | Este script selecciona el **ID del alquiler** (rental\_id) y los **nombres** (first\_name y last\_name) de los clientes de las tablas rental y customer. Utiliza un **JOIN** para combinar ambas tablas a través del **customer\_id**, que es la clave foránea que relaciona los alquileres con los clientes. |
| **43. Muestra todos los clientes y sus alquileres si existen, incluyendo aquellos que no tienen alquileres.** | Este script selecciona el **ID del cliente** (customer\_id), su **primer nombre** (first\_name), su **apellido** (last\_name), y el **ID del alquiler** (rental\_id). Utiliza un **LEFT JOIN** para combinar las tablas customer y rental, asegurando que todos los clientes aparezcan en el resultado, incluso si no tienen alquileres asociados. |
| **44. Realiza un CROSS JOIN entre las tablas film y category. ¿Aporta valor esta consulta? ¿Por qué? Deja después de la consulta la contestación.** | **Respuesta**: El uso de un **CROSS JOIN** puede no ser muy útil en la mayoría de los contextos prácticos, especialmente si las tablas son grandes. Esto se debe a que generará un número muy elevado de combinaciones que no necesariamente tienen sentido, ya que cada película se listará con cada categoría independientemente de si corresponde a esa película o no. Este tipo de consulta puede ser útil para fines de análisis de datos o pruebas, pero no se recomienda para aplicaciones de producción donde la eficiencia y la relevancia de los datos son importantes.  En escenarios reales, es más común utilizar un **INNER JOIN** o un **LEFT JOIN** donde las relaciones entre las tablas están definidas, como unir películas a sus respectivas categorías basadas en un campo de relación (como category\_id). |
| **45. Encuentra los actores que han participado en películas de la categoría 'Action'.** | Este script selecciona los nombres (first\_name) y apellidos (last\_name) de los actores que han actuado en películas clasificadas como 'Action'. Utiliza múltiples JOIN para combinar las tablas relevantes y filtra los resultados usando la cláusula WHERE para incluir solo las categorías que son 'Action'. Se utiliza DISTINCT para asegurar que no haya nombres duplicados en el resultado. |
| **46. Encuentra todos los actores que no han participado en películas.** | Este script selecciona el ID del actor (actor\_id), el primer nombre (first\_name) y el apellido (last\_name) de la tabla actor. Se realiza un LEFT JOIN con la tabla film\_actor, que relaciona actores con películas. La cláusula WHERE fa.film\_id IS NULL filtra los resultados para incluir solo aquellos actores que no tienen ningún registro de película asociado (es decir, que no han participado en ninguna película). |
| **47. Selecciona el nombre de los actores y la cantidad de películas en las que han participado.** | Este script selecciona el **primer nombre** (first\_name) y el **apellido** (last\_name) de la tabla actor, junto con el **número total de películas** (COUNT(fa.film\_id)) en las que ha participado cada actor. Se utiliza un **LEFT JOIN** con la tabla film\_actor para asegurarse de que todos los actores se incluyan, incluso si no han participado en ninguna película. La cláusula **GROUP BY** agrupa los resultados por ID, nombre y apellido del actor. Finalmente, se ordena el resultado por el **total de películas** en orden descendente. |
| **48. Crea una vista llamada “actor\_num\_peliculas” que muestre los nombres de los actores y el número de películas en las que han participado.** | Este script crea una vista llamada actor\_num\_peliculas que incluye el primer nombre (first\_name), el apellido (last\_name) y el número total de películas (COUNT(fa.film\_id)) en las que ha participado cada actor. Se utiliza un LEFT JOIN con la tabla film\_actor para asegurar que todos los actores estén incluidos en la vista, incluso aquellos que no han participado en ninguna película. La cláusula GROUP BY agrupa los resultados por el ID, nombre y apellido del actor. |
| **49. Calcula el número total de alquileres realizados por cada cliente.** | Este script selecciona el **ID del cliente** (customer\_id), el **primer nombre** (first\_name) y el **apellido** (last\_name) de la tabla customer, junto con el **número total de alquileres** (COUNT(r.rental\_id)) que ha realizado cada cliente. Se utiliza un **LEFT JOIN** con la tabla rental para asegurarse de que todos los clientes estén incluidos, incluso si no han realizado alquileres (el conteo será 0 en ese caso). La cláusula **GROUP BY** agrupa los resultados por el ID, nombre y apellido del cliente. Finalmente, se ordena el resultado por el **total de alquileres** en orden descendente. |
| **50. Calcula la duración total de las películas en la categoría 'Action'.** | Este script calcula la **duración total** de las películas que pertenecen a la categoría **'Action'**. Utiliza la función **SUM()** para agregar la duración (length) de todas las películas. Se realizan **JOINs** entre la tabla film, la tabla intermedia film\_category y la tabla category para relacionar las películas con sus respectivas categorías. La cláusula **WHERE** filtra solo aquellas películas que están en la categoría **'Action'**. |
| **51. Crea una tabla temporal llamada “cliente\_rentas\_temporal” para almacenar el total de alquileres por cliente.** | Este script crea una tabla temporal llamada cliente\_rentas\_temporal que contiene el ID del cliente (customer\_id), el primer nombre (first\_name), el apellido (last\_name) y el total de alquileres (total\_rentals) de cada cliente. Se utiliza un LEFT JOIN para asegurar que todos los clientes estén incluidos, incluso si no han realizado alquileres (el conteo será 0 en ese caso). La cláusula GROUP BY agrupa los resultados por ID, nombre y apellido del cliente. La tabla temporal estará disponible solo durante la sesión actual. |
| **52. Crea una tabla temporal llamada “peliculas\_alquiladas” que almacene las películas que han sido alquiladas al menos 10 veces.** | Este script crea una tabla temporal llamada peliculas\_alquiladas que almacena el film\_id, el title, y la cantidad total de alquileres (total\_alquileres) de cada película. Se realiza un JOIN entre las tablas film, inventory y rental para relacionar las películas con los registros de alquiler. Luego, se agrupan los resultados por film\_id y title, y se filtran las películas que han sido alquiladas al menos 10 veces utilizando HAVING COUNT(r.rental\_id) >= 10. |
| **53. Encuentra el título de las películas que han sido alquiladas por el cliente con el nombre ‘Tammy Sanders’ y que aún no se han devuelto. Ordena los resultados alfabéticamente por título de película.** | Este script selecciona los títulos de las películas de la tabla film que han sido alquiladas por el cliente llamado 'Tammy Sanders'. Se realiza un JOIN entre las tablas film, inventory, rental, y customer para relacionar las películas con los registros de alquiler y el cliente. En la cláusula WHERE, se filtran los resultados para incluir solo aquellos alquileres donde la fecha de devolución (return\_date) es NULL, lo que indica que las películas aún no han sido devueltas. Finalmente, se ordenan los resultados alfabéticamente por el título de la película. |
| **54. Encuentra los nombres de los actores que han actuado en al menos una**  **película que pertenece a la categoría ‘Sci-Fi’. Ordena los resultados**  **alfabéticamente por apellido.** | Este script selecciona los nombres y apellidos de los actores de la tabla actor que han actuado en al menos una película de la categoría ‘Sci-Fi’. Se realiza un JOIN entre las tablas actor, film\_actor, film, film\_category y category para relacionar los actores con las películas y sus categorías. La cláusula WHERE filtra los resultados para incluir solo aquellos actores que han participado en películas de la categoría ‘Sci-Fi’. La cláusula ORDER BY asegura que los resultados estén ordenados alfabéticamente por el apellido del actor. |
| **55. Encuentra el nombre y apellido de los actores que han actuado en películas que se alquilaron después de que la película ‘Spartacus Cheaper’ se alquilara por primera vez. Ordena los resultados alfabéticamente por apellido.** | Este script selecciona los nombres y apellidos de los actores que han actuado en películas que fueron alquiladas después de que la película ‘Spartacus Cheaper’ se alquilara por primera vez. Se utiliza un JOIN entre las tablas actor, film\_actor, film, inventory y rental para relacionar a los actores con las películas y sus registros de alquiler. La subconsulta obtiene la fecha de alquiler más temprana de ‘Spartacus Cheaper’ y se utiliza en la cláusula WHERE para filtrar los resultados. Finalmente, los resultados se ordenan alfabéticamente por apellido. |
| **56. Encuentra el nombre y apellido de los actores que no han actuado en ninguna película de la categoría ‘Music’** | Este script selecciona los nombres y apellidos de los actores de la tabla actor que no han participado en ninguna película de la categoría ‘Music’. Se utiliza una subconsulta para obtener todos los actor\_id de los actores que sí han actuado en películas de esta categoría. La cláusula NOT IN se usa para filtrar a los actores que no están en ese conjunto. Esto garantiza que solo se obtengan actores que no tienen registros en películas de la categoría ‘Music’. |
| **57. Encuentra el título de todas las películas que fueron alquiladas por más de 8 días.** | Este script selecciona los títulos de las películas de la tabla film que fueron alquiladas por más de **8 días**. Se utilizan los JOIN entre las tablas film, inventory y rental para relacionar las películas con sus registros de alquiler. La cláusula WHERE utiliza EXTRACT(DAY FROM ...) para calcular el número de días entre return\_date y rental\_date, permitiendo así comparar correctamente con el valor **8**. |
| **58. Encuentra el título de todas las películas que son de la misma categoría que ‘Animation’.** | Este script selecciona los títulos de las películas de la tabla film que pertenecen a la misma categoría que **'Animation'**. Se utilizan los JOIN entre las tablas film, inventory, film\_category y category para relacionar las películas con sus categorías. La cláusula WHERE filtra los resultados para obtener solo aquellas películas cuya categoría es **'Animation'**. |
| **59. Encuentra los nombres de las películas que tienen la misma duración que la película con el título ‘Dancing Fever’. Ordena los resultados alfabéticamente por título de película.** | ste script selecciona los títulos de las películas de la tabla film que tienen la misma duración que la película 'Dancing Fever'. La subconsulta (SELECT f2.length FROM film f2 WHERE f2.title = 'DANCING FEVER') obtiene la duración de 'Dancing Fever'. Luego, se comparan las duraciones en la cláusula WHERE. Los resultados se ordenan alfabéticamente por el título de la película usando ORDER BY. |
| **60. Encuentra los nombres de los clientes que han alquilado al menos 7 películas distintas. Ordena los resultados alfabéticamente por apellido.** | Este script selecciona los nombres y apellidos de los clientes de la tabla customer. Se realiza un JOIN entre las tablas customer, rental, e inventory para relacionar los clientes con sus alquileres. Luego se agrupan los resultados por customer\_id y se utiliza HAVING para filtrar aquellos que han alquilado al menos 7 películas distintas (COUNT(DISTINCT i.film\_id) >= 7). Finalmente, los resultados se ordenan alfabéticamente por el apellido del cliente usando ORDER BY. |
| **Y si quiero saber el nombre de las peliculas** | Este script selecciona los nombres de los clientes y los títulos de las películas que han alquilado. Se utiliza un JOIN entre las tablas customer, rental, inventory, y film para relacionar los clientes con sus alquileres y las películas correspondientes. La subconsulta filtra a los clientes que han alquilado al menos 7 películas distintas. Finalmente, los resultados se ordenan alfabéticamente por el apellido del cliente y luego por el título de la película. |
| **61. Encuentra la cantidad total de películas alquiladas por categoría y muestra el nombre de la categoría junto con el recuento de alquileres.** | Este script selecciona el nombre de la categoría y el recuento de alquileres para cada categoría. Utiliza JOIN para relacionar las tablas category, film\_category, film, inventory, y rental. Agrupa los resultados por nombre de categoría y cuenta el número de alquileres. Finalmente, se ordena el resultado en orden descendente por el número de alquileres. |
| **62. Encuentra el número de películas por categoría estrenadas en 2006.** | Este script cuenta cuántas películas pertenecen a cada categoría que se estrenaron en 2006. Se utiliza un JOIN para relacionar las tablas y agrupar los resultados por nombre de categoría. Se filtra por el año directamente, sin necesidad de conversión de tipos. |
| **63. Obtén todas las combinaciones posibles de trabajadores con las tiendas que tenemos.** | Este script realiza un CROSS JOIN entre las tablas staff y store, lo que permite obtener todas las combinaciones posibles de trabajadores y tiendas. El resultado incluye los nombres de los trabajadores y el ID de las tiendas. |
| **64. Encuentra la cantidad total de películas alquiladas por cada cliente y muestra el ID del cliente, su nombre y apellido junto con la cantidad de películas alquiladas.** | Este script cuenta el número total de alquileres para cada cliente utilizando un LEFT JOIN entre las tablas customer y rental. Se agrupa por el ID del cliente y se ordena por la cantidad de alquileres en orden descendente. Esto garantiza que también se muestren clientes que no hayan realizado ningún alquiler. |