

2025

# Data Project

LÓGICA CONSULTAS SQL

LAURA MORENO CASADO

# ÍNDICE

1. Descripción del proyecto .....	2
1.1. Descripción .....	2
1.2. Objetivos .....	2
2. Consultas SQL .....	2
3. Herramientas para realizar el poryecto .....	39

# 1. Descripción del proyecto

## 1.1. Descripción

Con este proyecto pretendemos entender y analizar la BBDD de lo que parece ser de alquileres de películas.

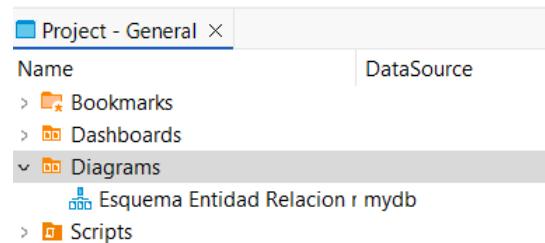
## 1.2. Objetivo

El objetivo es realizar todas las consultas planteadas a continuación.

# 2. Consultas SQL

## 2.1. Esquema de la BBDD

Para realizar el Esquema de BBDD lo hemos hecho desde el apartado de abajo “Project – General”. Para ello, hemos creado un nuevo diagrama llamado “Esquema Entidad Relación mydb”.



## 2.2. Muestra los nombres de todas las películas con una clasificación por edades de 'R'

Primero, hemos buscado la tabla donde puede estar esta información; luego nos hemos familiarizado con las columnas.

A continuación, hemos aplicado las fórmulas para poder mostrar lo que se pide en el enunciado ordenado por el id de cada película para que sigan un orden.

De manera, que ha quedado de la siguiente forma:

```
① select *  
      from film f ;  
  
② /** A continuación, observamos que la única clasificación R  
     * se encuentra en la columna Rating, por lo tanto,  
     * aplicamos dicha clasificación por esta columna.  
     */  
select film_id ,  
       title ,  
       rating  
  from film f  
 where rating = 'R'  
  order by film_id ;  
  
select film_id ,  
       title ,  
       rating  
  from film f  
 where rating = 'R'  
  order by film_id ;
```

\*\*Se puede observar en el script aportado 2.Nombres de películas con clasificación por edades de R.sql\*\*

El resultado ha sido:

Grilla	123	film_id	A-Z title	A-Z rating
1	8	AIRPORT POLLOCK	R	
2	17	ALONE TRIP	R	
3	20	AMELIE HELLFIGHTERS	R	
4	21	AMERICAN CIRCUS	R	
5	23	ANACONDA CONFESSIONS	R	
6	24	ANALYZE HOOSIERS	R	
7	30	ANYTHING SAVANNAH	R	
8	32	APOCALYPSE FLAMINGOS	R	
9	40	ARMY FLINTSTONES	R	
10	49	BADMAN DAWN	R	

### 2.3. Encuentra los nombres de los actores que tengan un “actor\_id” entre 30 y 40

Primero hemos identificado la tabla con la que trabajar; ha sido la tabla “actor”.

Y para poder realizar esta consulta, hemos recurrido al LIMIT y al OFFSET, de manera que:

```
/*Comenzamos identificando la tabla
 * Y familiarizandonos con las columnas.
 */
select *
from actor a;

/* Para esta consulta vamos a utilizar
 * el Limit y el offset de la siguiente manera
 */
select actor_id ,
       concat(first_name, ' ', last_name) as actors_names
from actor a
order by actor_id
limit 10
offset 30;
```

```
select actor_id ,
       concat(first_name, ' ', last_name) as actors_names
from actor a
order by actor_id
limit 10
offset 30;
```

\*\*Se puede observar en el script aportado 3. Actores de entre 30 y 40.sql\*\*

El resultado ha sido:

Grilla	123	actor_id	A-Z actors_names
1	31	SISSY SOBIESKI	
2	32	TIM HACKMAN	
3	33	MILLA PECK	
4	34	AUDREY OLIVIER	
5	35	JUDY DEAN	
6	36	BURT DUKAKIS	
7	37	VAL BOLGER	
8	38	TOM MCKELLEN	
9	39	GOLDIE BRODY	
10	40	JOHNNY CAGE	

## 2.4. Obtención de películas cuyo idioma coincide con el idioma original.

Para esta consulta hemos recurrido a dos tablas: film y language.

De manera que la consulta ha quedado así:

```
④ /**
 * En esta consulta nos damos cuenta
 * que con la tabla film solo no nos vale.
 */
select *
from film f;

④ -- Aquí cogemos la segunda tabla
select *
from "language" l;

④ -- La consulta queda de la siguiente manera:
select f.film_id,
       f.title,
       f.language_id,
       f.original_language_id,
       l."name"
from film f
right join "language" l
    on f.language_id = l.language_id
where f.language_id = f.original_language_id
order by f.film_id;
```

Sin embargo, esta consulta no ha generado resultados algunos, porque todos los valores de la columna de original\_language\_id son nulos, es decir, no hay valores; por lo tanto, no se genera ningún resultado.

```
select f.film_id,
       f.title,
       f.language_id,
       f.original_language_id,
       l."name"
from film f
right join "language" l
    on f.language_id = l.language_id
where f.language_id = f.original_language_id
order by f.film_id;
```

\*\*Se puede observar en el script aportado 4.Películas con idiomas coincidentes.sql\*\*

## 2.5. Ordena las películas por duración de forma ascendente.

Para esta consulta hemos recurrido de nuevo a la tabla de film.

De manera que:

```
④ /**
 * Comenzamos explorando la tabla:
 */
select *
from film f;

④ /**
 * Para ordenar la tabla por duración
 * de la película
 * vamos hacerlo de la siguiente manera:
 */
select film_id ,
       title ,
       length
from film f
order by length asc;
```

```

select film_id ,
       title ,
       length
  from film f
 order by length asc;

```

\*\*Se puede observar en el script aportado 5.Películas ordenadas por duración.sql\*\*

El resultado ha sido:

	123 film_id	A-Z title	123 length
1	504	KWAI HOMEWARD	46
2	505	LABYRINTH LEAGUE	46
3	469	IRON MOON	46
4	15	ALIEN CENTER	46
5	730	RIDGEMONT SUBMARINE	46
6	869	SUSPECTS QUILLS	47
7	398	HANOVER GALAXY	47
8	407	HAWK CHILL	47
9	393	HALLOWEEN NUTS	47
10	784	SHANGHAI TYCOON	47

2.6. Encuentra el nombre y apellido de los actores que tengan 'Allen' en su apellido.

Para esta consulta hemos recurrido a la tabla actor.

```

/*
 * Ahora cogemos la tabla actor:
 */
select *
from actor a;

/*
 * La consulta queda de la siguiente manera:
 */
select actor_id ,
       concat(first_name, ' ', last_name ) as actors_names
from actor a
where last_name = 'ALLEN'
order by actor_id ;

select actor_id ,
       concat(first_name, ' ', last_name ) as actors_names
from actor a
where last_name = 'ALLEN'
order by actor_id ;
```

\*\*Se puede observar en el script aportado 6.Nombre y apellidos de actores con Allen en su apellido.sql\*\*

El resultado ha sido:

123 actor_id	A-Z actors_names
118	CUBA ALLEN
145	KIM ALLEN
194	MERYL ALLEN

- 2.7. Encuentra la cantidad total de películas en cada clasificación de la tabla “film” y muestra la clasificación junto con el recuento.

La consulta queda de la siguiente manera:

```
④ /** Cogemos la tabla film
*/
select *
from film f;

④ --Ahora, hacemos la consulta:
select rating,
       count (film_id ) as total_films
from film f
group by rating;
```

```
select rating,
       count (film_id ) as total_films
from film f
group by rating;
```

\*\*Se puede observar en el script aportado 7.Total de películas por clasificación.sql\*\*

El resultado ha sido:

	rating	total_films
1	PG	194
2	G	178
3	PG-13	223
4	NC-17	210
5	R	195

- 2.8. Encuentra el título de todas las películas que son ‘PG-13’ o tienen una duración mayor a 3 horas en la tabla film.

Para esta consulta hemos cogido las horas y las hemos pasado a minutos, que en nuestro caso serían 360 minutos, por lo que la consulta queda de la siguiente manera:

```
④ --Cogemos la tabla film:
select *
from film f;

④ --Hacemos la consulta:
select film_id ,
       title ,
       length ,
       rating
from film f
where rating = 'PG-13' or length > 360;

select film_id ,
       title ,
       length ,
       rating
from film f
where rating = 'PG-13' or length > 360;
```

\*\*Se puede observar en el script aportado 8.Títulos de películas que son PG-13 o duración mayor a 3 horas.sql\*\*

El resultado ha sido:

	123 film_id	A-Z title	123 length	A-Z rating
1	7	AIRPLANE SIERRA	62	PG-13
2	9	ALABAMA DEVIL	114	PG-13
3	18	ALTER VICTORY	57	PG-13
4	28	ANTHEM LUKE	91	PG-13
5	33	APOLLO TEEN	153	PG-13
6	35	ARACHNOPHOBIA ROLLERCOASTER	147	PG-13
7	36	ARGONAUTS TOWN	127	PG-13
8	44	ATTACKS HATE	113	PG-13
9	45	ATTRACTION NEWTON	83	PG-13
10	48	BACKLASH UNDEFEATED	118	PG-13

## 2.9. Encuentra la variabilidad de lo que costaría reemplazar las películas.

Cogemos la tabla film, y la consulta quedaría así:

```
④ --Cogemos la tabla film:  
select *  
from film f ;  
  
④ --La consulta queda así:  
select variance(replacement_cost ) as variance  
from film f ;  
  
select variance(replacement_cost ) as variance  
from film f ;
```

\*\*Se puede observar en el script aportado 9.Variabilidad de coste de reemplazo de las películas.sql\*\*

El resultado ha sido:

	123 variance
1	36,6125765766

## 2.10. Encuentra la mayor y menor duración de una película de nuestra BBDD.

Cogemos la tabla film y la consulta quedaría así:

```
④ --Cogemos la tabla film:  
select *  
from film f ;  
  
④ --La consulta queda así:  
select min(length) as min_length,  
       max(length) as max_length  
from film f ;  
  
select min(length) as min_length,  
       max(length) as max_length  
from film f ;
```

\*\*Se puede observar en el script aportado 10.Mayor y menor duración de una película.sql\*\*

El resultado ha sido:

	min_length	max_length
1	46	185

2.11. Encuentra lo que costó el antepenúltimo alquiler ordenado por día.

Para esta consulta tenemos que coger la tabla de renta y payment.

Para ello, como el enunciado no especifica cómo tienen que estar las fechas ordenadas, si en orden ascendente o descendente, hemos recurrido a ordenar la consulta de forma descendente para así poder extraer la antepenúltima fila, en nuestro caso la tercera fila en orden descendente.

Por lo que la consulta quedaría de la siguiente manera:

```
④ --Cogemos la tabla rental:  
    select *  
    from rental r ;  
  
④ /**Nos va hacer falta también  
 * la tabla de payment  
 */  
    select *  
    from payment p ;  
  
④ --La consulta quedaría así:  
    select r.rental_id, r.rental_date, p.amount  
    from payment p  
    left join rental r  
        on r.rental_id = p.rental_id  
    order by r.rental_date desc  
    limit 1 offset 2 ;|
```

```
select r.rental_id, r.rental_date, p.amount  
from payment p  
left join rental r  
    on r.rental_id = p.rental_id  
order by r.rental_date desc  
limit 1 offset 2 ;
```

\*\*Se puede observar en el script aportado 11.Antepenúltimo alquiler ordenado por día.sql\*\*

El resultado ha sido:

	rental_id	rental_date	amount
1	11.646	2006-02-14 15:16:03.000	0,99

2.12. Encuentra el título de las películas en la tabla “film” que no sean ni ‘NC-17’ ni ‘G’ en cuanto a su clasificación.

Esta consulta quedaría de la siguiente manera:

```
④ --Cogemos la tabla film:  
    select *  
    from film f ;  
  
④ --La consulta quedaría:  
    select film_id , title , rating  
    from film f  
    where rating not in ('NC-17', 'G');
```

```

select film_id , title , rating
from film f
where rating not in ('NC-17', 'G');

```

\*\*Se puede observar en el script aportado 12.Títulos de películas que no son de la clasificación NC-17 ni G.sql\*\*

El resultado ha sido:

	123 ↗ film_id	A-Z title	A-Z rating
1	1	ACADEMY DINOSAUR	PG
2	6	AGENT TRUMAN	PG
3	7	AIRPLANE SIERRA	PG-13
4	8	AIRPORT POLLOCK	R
5	9	ALABAMA DEVIL	PG-13
6	12	ALASKA PHANTOM	PG
7	13	ALI FOREVER	PG
8	17	ALONE TRIP	R
9	18	ALTER VICTORY	PG-13
10	19	AMADEUS HOLY	PG
	--	--	--

2.13. 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.

La consulta quedaría de la siguiente manera:

```

--Cogemos la tabla film:
select *
from film f;

--La consulta quedaría así:
select rating ,
       avg(length) as length_average
from film f
group by rating ;

```

```

select rating ,
       avg(length) as length_average
from film f
group by rating ;

```

\*\*Se puede observar en el script aportado 13.Promedio duración de películas.sql\*\*

El resultado ha sido:

	A-Z rating	123 length_average
1	PG	112,0051546392
2	G	111,0505617978
3	PG-13	120,4439461883
4	NC-17	113,2285714286
5	R	118,6615384615

2.14. Encuentra el título de todas las películas que tengan una duración mayor a 180 minutos.

Cogemos la tabla film y la consulta quedaría así:

```

--Cogemos la tabla film:
select *
from film f;

--La consulta quedaría así:
select film_id , title , length
from film f
where length > 180;

```

```
select film_id , title , length
from film f
where length > 180;
```

\*\*Se puede observar en el script aportado 14.Películas duración mayor 180 minutos.sql\*\*

El resultado ha sido:

	123 film_id	A-Z title	123 length
1	24	ANALYZE HOOSIERS	181
2	50	BAKED CLEOPATRA	182
3	128	CATCH AMISTAD	183
4	141	CHICAGO NORTH	185
5	180	CONSPIRACY SPIRIT	184
6	182	CONTROL ANTHEM	185
7	198	CRYSTAL BREAKING	184
8	212	DARN FORRESTER	185
9	340	FRONTIER CABIN	183
10	349	GANGS PRIDE	185

## 2.15. ¿Cuánto dinero ha generado en total la empresa?

Cogemos la tabla payment, y la consulta quedaría así:

```
-- Cogemos la tabla payment:
select *
from payment p ;

-- La consulta quedaría así:
select sum(amount) as Total_generado
from payment p ;
```

```
select sum(amount) as Total_generado
from payment p ;
```

\*\* Se puede observar en el script aportado 15.Total generado en la empresa.sql\*\*

El resultado ha sido:

	123 total_generado
1	67.416,51

## 2.16. Muestra los 10 clientes con mayor valor de id.

Cogemos la tabla customer para ello quedando así la consulta:

```
-- Cogemos la tabla customer:
select *
from customer c ;

-- La consulta quedaría así:
select customer_id ,
       concat(first_name , ' ', last_name) as customer_name
from customer c
order by c.customer_id desc
limit 10 ;
```

```
select customer_id ,
       concat(first_name , ' ', last_name) as customer_name
from customer c
order by c.customer_id desc
limit 10 ;
```

\*\* Se puede observar en el script aportado 16.10 clientes con mayor valor id.sql\*\*

El resultado ha sido:

	customer_id	customer_name
1	599	AUSTIN CINTRON
2	598	WADE DELVALLE
3	597	FREDDIE DUGGAN
4	596	ENRIQUE FORSYTHE
5	595	TERRENCE GUNDERSON
6	594	EDUARDO HIATT
7	593	RENE MCALISTER
8	592	TERRANCE ROUSH
9	591	KENT ARSENAULT
10	590	SETH HANNON

- 2.17. Encuentra el nombre y apellido de los actores que aparecen en la película con título 'Egg Igyb'.

Para realizar esta consulta, hemos cogido 3 tablas, actor, film\_actor y film. De modo que, la consulta queda de la siguiente manera:

```
/* --Cogemos 3 tablas en este caso:  
select *  
from actor a ;  
  
select *  
from film_actor fa ;  
  
select *  
from film f ;  
  
--La consulta quedaría así:  
select concat (a.first_name, ' ', a.last_name) as actors_names ,  
       f.title  
  from film f  
inner join film_actor fa  
    on f.film_id = fa.film_id  
inner join actor a  
  on a.actor_id = fa.actor_id  
where f.title = 'EGG IGBY';|  
  
select concat (a.first_name, ' ', a.last_name) as actors_names ,  
       f.title  
  from film f  
inner join film_actor fa  
    on f.film_id = fa.film_id  
inner join actor a  
  on a.actor_id = fa.actor_id  
where f.title = 'EGG IGBY';
```

\*\* Se puede observar en el script aportado 17.Nombre y apellidos de actores en película EGG IGBY.sql\*\*

El resultado ha sido:

	actors_names	title
1	LUCILLE TRACY	EGG IGBY
2	TOM MCKELLEN	EGG IGBY
3	NATALIE HOPKINS	EGG IGBY
4	MERYL GIBSON	EGG IGBY
5	OPRAH KILMER	EGG IGBY

## 2.18. Selecciona todos los nombres de las películas únicos.

Para esta consulta utilizamos el “DISTINCT” para indicarle que solo queremos los registros únicos, y quedaría de la siguiente manera:

```
④ --Cogemos la tabla film
select *
from film f;

④ --Ahora, generamos la consulta de la siguiente forma:
select distinct title as unics_titles
from film f;

select distinct title as unics_titles
from film f;
```

\*\* Se puede observar en el script aportado 18.Nombre pelíclulas únicos.sql\*\*

El resultado ha sido:

	A-Z unics_titles
1	ITALIAN AFRICAN
2	FICTION CHRISTMAS
3	BADMAN DAWN
4	LEGALLY SECRETARY
5	PELICAN COMFORTS
6	SEARCHERS WAIT
7	STALLION SUNDANCE
8	FRONTIER CABIN
9	TERMINATOR CLUB
10	WRONG BEHAVIOR

## 2.19. Encuentra el título de las películas que son comedias y tienen una duración mayor a 180 minutos en la tabla “film”.

Para realizar esta consulta, hemos tenido que anidar 3 tablas, “film”, “film\_category” y “category”, de modo que la consulta queda de la siguiente manera:

```
④ /** Para esta consulta, vamos a necesitar
* anidar 3 tablas, film, film_category y category
* Por lo tanto, vemos primero las 3 tablas para
* ver con qué campos nos quedamos:
*/
select *
from film f;

④ select *
from film_category fc;

④ select *
from category c;

④ --Ahora, generamos la consulta:
select f.title as film_name , c."name" as category_name, f.length as film_length
from film f
inner join film_category fc
  on fc.film_id = f.film_id
inner join category c
  on fc.category_id = c.category_id
where c."name" = 'Comedy' and f.length > 180 ;|
```

```

select f.title as film_name , c."name" as category_name, f.length as film_length
from film f
inner join film_category fc
    on fc.film_id = f.film_id
    inner join category c
        on fc.category_id = c.category_id
where c."name" = 'Comedy' and f.length > 180 ;

```

\*\* Se puede observar en el script aportado 19.Películas que son comedia con duración mayor a 180 minutos.sql\*\*

El resultado ha sido:

	A-Z film_name	A-Z category_name	123 film_length
1	CONTROL ANTHEM	Comedy	185
2	SATURN NAME	Comedy	182
3	SEARCHERS WAIT	Comedy	182

- 2.20. 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.

Para esta consulta hemos tenido que anidar, de nuevo tres tablas, “film”, “film\_category” y “category”, de manera que la consulta queda de la siguiente manera:

```

/*
 * Para esta consulta, vamos a necesitar
 * anidar 3 tablas, film, film_category y category
 * Por lo tanto, vemos primero las 3 tablas para
 * ver con qué campos nos quedamos:
 */
select *
from film f;

--select *
--from film_category fc;

--select *
--from category c;

--Ahora, generaremos la consulta:
select c."name",
       avg (f.length ) as length_average
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 f.length > 180
group by c."name"
order by c."name";

select c."name",
       avg (f.length ) as length_average
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 f.length > 180


```

```
group by c."name"
order by c."name";
```

\*\* Se puede observar en el script aportado 20.Categorías de películas con promedio superior a 110 minutos de duración.sql\*\*

El resultado ha sido:

	A-Z name	length_average
1	Action	185
2	Animation	183,8
3	Classics	184
4	Comedy	183
5	Documentary	183
6	Drama	181
7	Family	183
8	Foreign	182,6
9	Games	183
10	Horror	181
11	Music	183,3333333333
12	New	181,6666666667
13	Sci-Fi	185
14	Sports	182,3333333333
15	Travel	185

## 2.21. ¿Cuál es la media de duración del alquiler de las películas?

La consulta queda así:

```
/* Cogemos la tabla film
 * para quedarnos con las columnas que
 * nos hacen falta para esta consulta:
 */
select *
from film f;

-- La consulta quedaría de la siguiente manera:
select avg (rental_duration) as media_duracion
from film f;
```

```
select avg (rental_duration) as media_duracion
from film f;
```

\*\* Se puede observar en el script aportado 21.Media de duración del alquiler de las películas.sql\*\*

El resultado ha sido:

	media_duracion
1	4,985

## 2.22. Crea una columna con el nombre y apellidos de todos los actores y actrices.

La consulta queda así:

```
-- cogemos la tabla actor:
select *
from actor a;

-- La consulta quedaría así:
select concat (first_name, ' ', last_name ) as actors_names
from actor a;
```

```
select concat (first_name, ' ', last_name ) as actors_names
from actor a;
```

\*\* Se puede observar en el script aportado 22.Columna con nombres y apellidos de todos los actores y actrices.sql\*\*

El resultado ha sido:

	A-Z actors_names
1	PENELOPE GUINNESS
2	NICK WAHLBERG
3	ED CHASE
4	JENNIFER DAVIS
5	JOHNNY LOLLOBRIGIDA
6	BETTE NICHOLSON
7	GRACE MOSTEL
8	MATTHEW JOHANSSON
9	JOE SWANK
10	CHRISTIAN GABLE

2.23. Números de alquiler por día, ordenados por cantidad de alquiler de forma descendente.

La consulta queda así:

```
-- Cogemos la tabla renta:  
select *  
from rental r;  
  
-- La consulta quedaría así:  
select rental_date,  
       count (rental_id ) as rental_number  
from rental r  
group by rental_date  
order by rental_number desc ;|
```

```
select rental_date,  
       count (rental_id ) as rental_number  
from rental r  
group by rental_date  
order by rental_number desc ;
```

\*\* Se puede observar en el script aportado 23.Números de alquiler por día.sql\*\*

El resultado ha sido:

	rental_date	rental_number
1	2006-02-14 15:16:03.000	182
2	2005-07-29 08:40:36.000	2
3	2005-08-22 04:31:50.000	2
4	2005-08-23 08:48:43.000	2
5	2005-07-29 00:14:37.000	2
6	2005-08-19 16:47:41.000	2
7	2005-07-31 09:08:03.000	2
8	2005-08-20 07:21:15.000	2
9	2005-08-01 00:08:01.000	2
10	2005-05-30 14:47:31.000	2

2.24. Encuentra las películas con una duración superior al promedio.

La consulta quedaría así:

```


-- cogemos la tabla film:
select *
from film f;

/** Primero calculamos cual es
 * la media de la duración,
 * que observamos que es 115
 * sin decimales, puesto que la columna no tiene decimales:
 */
select round(avg (length))
from film f;

--Ahora hacemos al consulta:
select title, length
from film f
where length > 115 ;

** Primero calculamos cual es
* la media de la duración,
* que observamos que es 115
* sin decimales, puesto que la columna no tiene decimales:
*/
select round(avg (length))
from film f;

--Ahora hacemos al consulta:
select title, length
from film f
where length > 115 ;

** Se puede observar en el script aportado 24.Películas con duración superior al promedio.sql**


```

El resultado ha sido:

	A-Z title	123 length
1	AFFAIR PREJUDICE	117
2	AFRICAN EGG	130
3	AGENT TRUMAN	169
4	ALAMO VIDEOTAPE	126
5	ALASKA PHANTOM	136
6	ALI FOREVER	150
7	ALLEY EVOLUTION	180
8	AMERICAN CIRCUS	129
9	ANALYZE HOOSIERS	181
10	ANONYMOUS HUMAN	179
11	ANTITRUST TOMATOES	160

## 2.25. Averigua el número de alquileres registrados por mes.

En esta consulta hemos utilizado la función extract para extraer el mes de la fecha, entonces la consulta quedaría así:

```


--Cogemos la tabla rental
select *
from rental r;

/** En esta consulta hemos utilizado
 * extract para poder extraer el mes de la fecha:
 */
select count (rental_id ) as rental_number,
       extract (month from rental_date) as month
from rental r
group by (extract (month from rental_date))
order by month;


```

```

select count (rental_id ) as rental_number,
       extract (month from rental_date) as month
  from rental r
 group by (extract (month from rental_date))
 order by month;

```

\*\* Se puede observar en el script aportado 25.Numero de alquileres por mes.sql\*\*

El resultado ha sido:

	123 rental_number	123 month
1	182	2
2	1.156	5
3	2.311	6
4	6.709	7
5	5.686	8

2.26. Encuentra el promedio, la desviación estándar y varianza del total pagado.

La consulta quedaría así:

```

--Cogemos la tabla payment:
select *
from payment p ;

--La consulta quedaría así
select avg (amount ) as average,
       stddev(amount) as std,
       variance(amount ) as variance
  from payment p ;

select avg (amount ) as average,
       stddev(amount) as std,
       variance(amount ) as variance
  from payment p ;

```

\*\* Se puede observar en el script aportado 26.Promedio, desv. estandar y varianza del total pagado.sql\*\*

El resultado ha sido:

	123 average	123 std	123 variance
1	4,2006673313	2,3629938536	5,5837399522

2.27. ¿Qué películas se alquilan por encima del precio medio?

Para esta consulta, necesitamos anidar 4 tablas hasta llegar al resultado esperado, y además, una vez, sabiendo el precio medio por la consulta anterior, la consulta en cuestión quedaría de la siguiente manera:

```

@/**En este caso tenemos que anexar 4 tablas
 * hasta llegar al resultado esperado:
 */
select *
from film;

@select *
from payment p ;

@select *
from inventory i ;

@select *
from rental r ;

--Primero vamos a ver el precio medio por el que se alquilan las películas:
select round(avg(amount)) as precio_medio
from payment p ;

--Sabiendo que el precio medio son 4, la consulta quedaría así:
select f.title, p.amount
from film f
inner join inventory i
  on f.film_id = i.film_id
inner join rental r
  on r.inventory_id = i.inventory_id
inner join payment p
  on p.rental_id = r.rental_id
where p.amount > 4

      select f.title, p.amount
      from film f
      inner join inventory i
          on f.film_id = i.film_id
      inner join rental r
          on r.inventory_id = i.inventory_id
      inner join payment p
          on p.rental_id = r.rental_id
      where p.amount > 4 ;

```

\*\* Se puede observar en el script aportado 27. Películas por encima del precio medio\*\*

El resultado ha sido:

	A-Z title	123 amount
1	MUSKeteers WAIT	5,99
2	FERRIS MOTHER	9,99
3	CLOSER BANG	4,99
4	ATTACKS HATE	4,99
5	FIRE WOLVES	5,99
6	SATURDAY LAMBS	5,99
7	SNATCH SLIPPER	4,99
8	CONFIDENTIAL INTERVIEW	4,99
9	EXPECATIONS NATURAL	7,99
10	DOORS PRESIDENT	4,99
11	LISIATI INTOUCHABLES	4,99

## 2.28. Muestra el id de los actores que hayan participado en más de 40 películas.

La consulta quedaría de la siguiente manera:

```
④ --Cogemos la tabla film_actor:  
    select *  
    from film_actor fa ;  
  
④ --La consulta quedaría así:  
    select actor_id ,  
          count (film_id) as film_numbers  
    from film_actor fa  
    group by actor_id  
    order by actor_id ;
```

```
select actor_id ,  
      count (film_id)  
from film_actor fa  
group by actor_id  
order by actor_id ;
```

\*\* Se puede observar en el script aportado 28.id actores en más de 40 películas.sql\*\*

El resultado ha sido:

	123 actor_id	123 film_numbers
1	1	19
2	2	25
3	3	22
4	4	22
5	5	29
6	6	20
7	7	30
8	8	20
9	9	25
10	10	22
11	11	25

## 2.29. Obtener todas las películas y, si están disponibles en el inventario, mostrar la cantidad disponible.

La consulta quedaría de la siguiente manera:

```
④ -- Cogeremos dos tablas en esta consulta:  
    select *  
    from film f ;  
  
④ select *  
      from inventory i ;  
  
④ --La consulta quedaría así:  
    select f.film_id, f.title, count (i.inventory_id) as inventory  
    from film f  
    inner join inventory i  
      on f.film_id = i.film_id  
    where exists(  
      select 1  
      from inventory i  
      where f.film_id = i.film_id )  
    group by f.film_id, f.title  
    order by f.film_id ;
```

```

select f.film_id, f.title, count(i.inventory_id) as inventory
from film f
inner join inventory i
    on f.film_id = i.film_id
where exists(
    select 1
    from inventory i
    where f.film_id = i.film_id )
group by f.film_id, f.title
order by f.film_id ;

```

\*\* Se puede observar en el script aportado 29.Cantidad disponible de películas en el inventario.sql\*\*

El resultado ha sido:

	123 ↗ film_id	Az title	123 inventory
1	1	ACADEMY DINOSAUR	8
2	2	ACE GOLDFINGER	3
3	3	ADAPTATION HOLES	4
4	4	AFFAIR PREJUDICE	7
5	5	AFRICAN EGG	3
6	6	AGENT TRUMAN	6
7	7	AIRPLANE SIERRA	5
8	8	AIRPORT POLLOCK	4
9	9	ALABAMA DEVIL	5
10	10	ALADDIN CALENDAR	7
11	11	ΔΙ ΔΜΩ ΒΙΝΦΟΤΑΡΡ	7

### 2.30. Obtener los actores y el número de películas en las que ha actuado.

La consulta quedaría de la siguiente manera:

```

-- Cogeremos dos tablas:
select *
from actor a;

--select *
--from film_actor fa;

--La consulta quedaría así:
select a.actor_id,
       concat(a.first_name , ' ', a.last_name) as actors_names,
       count (fa.film_id) as number_films
  from actor a
  left join film_actor fa
    on a.actor_id = fa.actor_id
 group by a.actor_id
 order by a.actor_id;

select a.actor_id,
       concat(a.first_name , ' ', a.last_name) as actors_names,
       count (fa.film_id) as number_films
  from actor a
  left join film_actor fa
    on a.actor_id = fa.actor_id
 group by a.actor_id
 order by a.actor_id;

```

\*\* Se puede observar en el script aportado 30.Actores y número de películas en las que ha actuado.sql\*\*

El resultado ha sido:

	123 ↗ actor_id	A-Z actors_names	123 ↗ number_films
1	1	PENELOPE GUINNESS	19
2	2	NICK WAHLBERG	25
3	3	ED CHASE	22
4	4	JENNIFER DAVIS	22
5	5	JOHNNY LOLLOBRIGIDA	29
6	6	BETTE NICHOLSON	20
7	7	GRACE MOSTEL	30
8	8	MATTHEW JOHANSSON	20
9	9	JOE SWANK	25
10	10	CHRISTIAN GABLE	22
11	11	ZERO CAGE	25

2.31. Obtener todas las películas y mostrar los actores que han actuado en ellas, incluso si algunas películas no tienen actores asociados.

```

/*
 * En esta consulta cogeremos 3 tablas
 * para mostrar la información mejor:
 */
select *
from film f;

-- select *
-- from film_actor fa;

-- select *
-- from actor a;

-- La consulta quedaría así:
select f.film_id,
       f.title,
       concat (a.first_name, ' ', a.last_name) as actors_name
from film f
inner join film_actor fa
  on fa.film_id = f.film_id
inner join actor a
  on fa.actor_id = a.actor_id
where concat (a.first_name, ' ', a.last_name) is not null or concat (a.first_name, ' ', a.last_name) is null
order by f.film_id ;
```

```

select f.film_id,
       f.title,
       concat (a.first_name, ' ', a.last_name) as actors_name
from film f
inner join film_actor fa
  on fa.film_id = f.film_id
inner join actor a
  on fa.actor_id = a.actor_id
where concat (a.first_name, ' ', a.last_name) is not null or concat (a.first_name, ' ', a.last_name) is null
order by f.film_id ;
```

\*\* Se puede observar en el script aportado 31.Todas las películas y actores que han actuado en ellas.sql\*\*

2.32. Obtener todos los actores y mostrar las películas en las que han actuado, incluso si algunos actores no han actuado en ninguna película.

La consulta quedaría así:

```
/* En esta consulta cogeremos 3 tablas
 * para mostrar la información mejor:
 */
select *
from film f;

select *
from film_actor fa;

select *
from actor a;

-- La consulta quedaría así:
select a.actor_id ,
       concat (a.first_name, ' ', a.last_name) as actors_name ,
       f.film_id,
       f.title
  from film f
 inner join film_actor fa
    on fa.film_id = f.film_id
 inner join actor a
    on fa.actor_id = a.actor_id
   where f.film_id is not null
 or f.film_id is null
 order by actor_id ;

select a.actor_id ,
       concat (a.first_name, ' ', a.last_name) as actors_name ,
       f.film_id,
       f.title
  from film f
 inner join film_actor fa
    on fa.film_id = f.film_id
 inner join actor a
    on fa.actor_id = a.actor_id
   where f.film_id is not null
 or f.film_id is null
 order by actor_id ;

** Se puede observar en el script aportado 32.Actores y películas en las que han actuado.sql**
```

### 2.33. Obtener todas las películas que tenemos y todos los registros de alquiler.

```
④-- Aquí cogeremos 3 tablas:  
    select *  
    from film f ;  
  
    select *  
    from inventory i ;  
  
    select *  
    from rental r ;  
  
④-- La consulta quedaría así:  
    select f.film_id, f.title , r.rental_id  
    from film f  
    inner join inventory i  
        on f.film_id = i.film_id  
    inner join rental r  
        on i.inventory_id = r.inventory_id  
    order by f.film_id ;
```

```
select f.film_id, f.title , r.rental_id  
from film f  
inner join inventory i  
    on f.film_id = i.film_id  
inner join rental r  
    on i.inventory_id = r.inventory_id  
order by f.film_id ;
```

\*\* Se puede observar en el script aportado 33.Películas y registros de alquiler.sql\*\*

### 2.34. Encuentra los 5 clientes que más dinero se hayan gastado con nosotros.

```
④--Cogeremos dos tablas:  
    select *  
    from customer c ;  
  
    select *  
    from payment p ;  
  
④--La consulta quedaría así:  
    select c.customer_id ,  
        concat (c.first_name, ' ', c.last_name) as customers_name ,  
        sum(p.amount) as total  
    from customer c  
    inner join payment p  
        on c.customer_id = p.customer_id  
    group by c.customer_id  
    order by total desc  
    limit 5 ;|
```

```
select c.customer_id ,  
    concat (c.first_name, ' ', c.last_name) as customers_name ,  
    sum(p.amount) as total  
from customer c  
inner join payment p  
    on c.customer_id = p.customer_id  
group by c.customer_id  
order by total desc  
limit 5 ;
```

\*\* Se puede observar en el script aportado 34.Los 5 clientes que más dinero han gastado.sql\*\*

El resultado ha sido:

	customer_id	customers_name	total
1	526	KARL SEAL	221,55
2	148	ELEANOR HUNT	216,54
3	144	CLARA SHAW	195,58
4	137	RHONDA KENNEDY	194,61
5	178	MARION SNYDER	194,61

2.35. Selecciona todos los actores cuyo primer nombre es 'Johnny'.

La consulta quedaría así:

```
-- Cogemos la tabla de actores:  
select *  
from actor a;  
  
-- La consulta quedaría así:  
select actor_id ,  
       concat(first_name , ' ', last_name) as actors_names  
from actor a  
where first_name = 'JOHNNY'  
order by actor_id ;  
  
select actor_id ,  
       concat(first_name , ' ', last_name) as actors_names  
from actor a  
where first_name = 'JOHNNY'  
order by actor_id ;
```

\*\* Se puede observar en el script aportado 35.Actores cuyo primer nombre es Johnny.sql\*\*

El resultado ha sido:

	actor_id	actors_names
1	5	JOHNNY LOLLOBRIGIDA
2	40	JOHNNY CAGE

2.36. Renombra la columna "first\_name" como Nombre y "last\_name" como Apellido.

La consulta sería:

```
-- Cogemos la tabla actor:  
select *  
from actor a;  
  
-- La consulta quedaría así:  
select actor_id ,  
       first_name as Nombre ,  
       last_name as Apellido ,  
       last_update  
from actor a;  
  
select actor_id ,  
       first_name as Nombre ,  
       last_name as Apellido ,  
       last_update  
from actor a;
```

\*\* Se puede observar en el script aportado 36.Columnas tabla actores renombradas.sql\*\*

2.37. Encuentra el ID del actor más bajo y más alto en la tabla actor.

```
-- La consulta quedaría así:  
select max (actor_id) as max_ID ,  
       min (actor_id ) as min_ID  
from actor a;
```

```

select max(actor_id) as max_ID ,
       min(actor_id ) as min_ID
  from actor a ;

```

\*\* Se puede observar en el script aportado 37.id más alto y más bajo de la tabla actor.sql\*\*

El resultado ha sido:

	123 max_id	123 min_id
1	200	1

2.38. Cuenta cuántos actores hay en la tabla “actor”.

--La consulta quedaría así:  

```

select count (*) as actor_numbers
from actor a ;

```

```

select count (*) as actor_numbers
from actor a ;

```

\*\* Se puede observar en el script aportado 38.Número de actores que hay en la tabla actor.sql\*\*

El resultado ha sido:

	123 actor_numbers
1	200

2.39. Selecciona todos los actores y ordénalos por apellido en orden ascendente.

--La consulta quedaría así:  

```

select *
from actor a
order by last_name asc;

```

```

select *
from actor a
order by last_name asc;

```

\*\* Se puede observar en el script aportado 39.actores ordenados por apellidos en ascendente.sql\*\*

2.40. Selecciona las primeras 5 películas de la tabla “film”.

Sería:

--La consulta quedaría así:  

```

select *
from film f
limit 5;

```

```

select *
from film f
limit 5;

```

\*\* Se puede observar en el script aportado 40.Primeras 5 películas de la tabla film.sql\*\*

El resultado ha sido:

	123 film_id	A-Z title	A-Z description	123 release_year	123 lang
> 1	1	ACADEMY DINOSAUR	A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The	2.006	1
> 2	2	ACE GOLDFINGER	A Astounding Epistle of a Database Administrator And a Explorer who must Find a	2.006	1
> 3	3	ADAPTATION HOLES	A Astounding Reflection of a Lumberjack And a Car who must Sink a Lumberjack in	2.006	1
> 4	4	AFFAIR PREJUDICE	A Fanciful Documentary of a Frisbee And a Lumberjack who must Chase a Monkey	2.006	1
> 5	5	AFRICAN EGG	A Fast-Paced Documentary of a Pastry Chef And a Dentist who must Pursue a Forer	2.006	1

2.41. Agrupa los actores por su nombre y cuenta cuántos actores tienen el mismo nombre. ¿Cuál es el nombre más repetido?

La consulta queda así:

```
--La consulta quedaría así:  
select first_name ,  
       count (actor_id ) as number  
  from actor a  
 group by first_name  
order by "number" desc;  
  
select first_name ,  
       count (actor_id ) as number  
  from actor a  
 group by first_name  
order by "number" desc;
```

\*\* Se puede observar en el script aportado 41.Agrupación de actores por su nombre y el más repetido.sql\*\*

El resultado ha sido, “Kenneth”, “Penelope” y “Julia”:

	A-Z first_name	123 number
1	KENNETH	4
2	PENELOPE	4
3	JULIA	4
4	BURT	3
5	GENE	3
6	DAN	3
7	GROUCHO	3
8	MATTHEW	3
9	MORGAN	3
10	RUSSELL	3
11	CCHRISTIAN	2

2.42. Encuentra todos los alquileres y los nombres de los clientes que los realizaron.

La consulta quedaría de la siguiente manera:

```
--Cogeremos dos tablas:  
select *  
  from customer c ;  
  
--select *  
  from rental r ;  
  
--La consulta quedaría así:  
select r.rental_id ,  
       concat (c.first_name, ' ', c.last_name) as customers  
  from rental r  
 right join customer c  
    on r.customer_id = c.customer_id  
order by r.rental_id ;
```

```
select r.rental_id ,  
       concat (c.first_name, ' ', c.last_name) as customers  
  from rental r  
 right join customer c  
    on r.customer_id = c.customer_id  
order by r.rental_id ;
```

\*\* Se puede observar en el script aportado 42.Alquileres y nombres de los clientes que lo realizaron.sql\*\*

El resultado ha sido:

	123 ▾ rental_id	A-Z customers
1	1	CHARLOTTE HUNTER
2	2	TOMMY COLLAZO
3	3	MANUEL MURRELL
4	4	ANDREW PURDY
5	5	DELORES HANSEN
6	6	NELSON CHRISTENSON
7	7	CASSANDRA WALTERS
8	8	MINNIE ROMERO
9	9	ELLEN SIMPSON
10	10	DANNY ISOM
11	11	APRII BURNS

2.43. Muestra todos los clientes y sus alquileres si existen, incluyendo aquellos que no tienen alquileres.

Sería:

```
④ --Cogeremos dos tablas:  
select *  
from customer c ;  
  
④ select *  
from rental r ;  
  
④ --La consulta quedaría así:  
select c.customer_id ,  
       rental_id  
from customer c  
left join rental r  
      on c.customer_id = r.customer_id  
where rental_id is not null  
      or rental_id is null  
order by c.customer_id;  
  
select c.customer_id ,  
       rental_id  
from customer c  
left join rental r  
      on c.customer_id = r.customer_id  
where rental_id is not null  
      or rental_id is null  
order by c.customer_id;
```

\*\* Se puede observar en el script aportado 43.Clientes y alquileres.sql\*\*

2.44. Realiza un CROSS JOIN entre las tablas film y category. ¿Aporta valor esta consulta? ¿Por qué? Deja después de la consulta la contestación.

```
④ --La consulta queda así:  
select *  
from film f  
cross join category c ;  
  
select *  
from film f  
cross join category c ;
```

\*\* Se puede observar en el script aportado 44.Cross join entre tablas film y category.sql\*\*

El resultado es una consulta sin valor añadido, puesto que lo que hace el sistema es simplemente añadir las columnas de categoría a la tabla de “film” repitiendo el id de film tantas veces como categorías haya, por lo tanto, esta consulta no nos aporta nada.

## 2.45. Encuentra los actores que han participado en películas de la categoría 'Action'.

La consulta quedaría así:

```
④-- Para esta consulta necesitamos anidar varias tablas:  
select *  
from category c ;  
  
④select *  
from film_category fc ;  
  
④select *  
from film_actor fa ;  
  
④select *  
from actor a ;  
  
④-- La consulta quedaría así:  
select a.actor_id, concat (a.first_name, ' ', a.last_name) as actors_name, c."name" as category_name  
from category c  
inner join film_category fc  
    on fc.category_id = c.category_id  
inner join film_actor fa  
    on fa.film_id = fc.film_id  
inner join actor a  
    on fa.actor_id = a.actor_id  
where c."name" = 'Action'  
order by a.actor_id;
```

```
select a.actor_id, concat (a.first_name, ' ', a.last_name) as actors_name,  
c."name" as category_name  
from category c  
inner join film_category fc  
    on fc.category_id = c.category_id  
inner join film_actor fa  
    on fa.film_id = fc.film_id  
inner join actor a  
    on fa.actor_id = a.actor_id  
where c."name" = 'Action'  
order by a.actor_id;
```

\*\* Se puede observar en el script aportado 45. Actores que han participado en la categoría Action.sql\*\*

## 2.46. Encuentra todos los actores que no han participado en películas.

La consulta quedaría así:

```
④-- Para esta consulta necesitamos anidar varias tablas:  
select *  
from film_category fc ;  
  
④select *  
from film_actor fa ;  
  
④select *  
from actor a ;  
  
④-- La consulta quedaría así:  
select a.actor_id, concat (a.first_name, ' ', a.last_name) as actors_name  
from actor a  
inner join film_actor fa  
    on a.actor_id = fa.actor_id  
inner join film_category fc  
    on fc.film_id = fa.film_id  
where fa.film_id is null ;
```

```

select a.actor_id, concat (a.first_name, '', a.last_name) as actors_name
from actor a
inner join film_actor fa
    on a.actor_id = fa.actor_id
inner join film_category fc
    on fc.film_id = fa.film_id
where fa.film_id is null ;

```

\*\* Se puede observar en el script aportado 46.Actores que no han participado en películas.sql\*\*

2.47. Selecciona el nombre de los actores y la cantidad de películas en las que han participado.

La consulta quedaría así:

```

 $\circ$ -- Para esta consulta necesitamos anidar varias tablas:
 $\circ$ select *
from film_category fc ;

 $\circ$ select *
from film_actor fa ;

 $\circ$ select *
from actor a ;

 $\circ$ -- La consulta quedaría así:
select a.actor_id,
       concat (a.first_name, ' ', a.last_name) as actors_name ,
       count (fa.film_id) as films_number
from actor a
inner join film_actor fa
    on a.actor_id = fa.actor_id
inner join film_category fc
    on fc.film_id = fa.film_id
group by a.actor_id
order by a.actor_id;

```

```

select a.actor_id,
       concat (a.first_name, ' ', a.last_name) as actors_name ,
       count (fa.film_id) as films_number
from actor a
inner join film_actor fa
    on a.actor_id = fa.actor_id
inner join film_category fc
    on fc.film_id = fa.film_id
group by a.actor_id
order by a.actor_id;

```

\*\* Se puede observar en el script aportado 47.Actores y cantidad de películas en las que han participado.sql\*\*

2.48. Crea una vista llamada “actor\_num\_películas” que muestre los nombres de los actores y el número de películas en las que han participado.

Quedaría de la siguiente manera:

```


    -- Para esta consulta necesitamos anidar varias tablas:
    select *
    from film_category fc ;

    select *
    from film_actor fa ;

    select *
    from actor a ;

    -- La consulta quedaría así:
    create view "actor_num_películas" as
    select a.actor_id,
        concat (a.first_name, ' ', a.last_name) as actors_name ,
        count (fa.film_id) as films_number
    from actor a
    inner join film_actor fa
        on a.actor_id = fa.actor_id
    inner join film_category fc
        on fc.film_id = fa.film_id
    group by a.actor_id
    order by a.actor_id;

create view "actor_num_películas" as
select a.actor_id,
    concat (a.first_name, ' ', a.last_name) as actors_name ,
    count (fa.film_id) as films_number
from actor a
inner join film_actor fa
    on a.actor_id = fa.actor_id
inner join film_category fc
    on fc.film_id = fa.film_id
group by a.actor_id
order by a.actor_id;


```

\*\* Se puede observar en el script aportado 48.Creación de vista llamada actor\_num\_películas.sql\*\*

## 2.49. Calcula el número total de alquileres realizados por cada cliente.

Quedaría de la siguiente manera:

```


    -- La consulta quedaría así:
    select customer_id , count (rental_id ) as rent_number
    from rental r
    group by customer_id
    order by customer_id ;

select customer_id , count (rental_id ) as rent_number
from rental r
group by customer_id
order by customer_id ;

** Se puede observar en el script aportado 49.Número total de alquileres realizado por cliente.sql**


```

## 2.50. Calcula la duración total de las películas en la categoría 'Action'.

La consulta quedaría de la siguiente manera:

```


--La consulta sería:
select f.film_id, f.title , sum (f.length) as total_length
from film f
left join film_category fc
    on fc.film_id = f.film_id
where fc.category_id = 1
group by f.film_id
order by f.film_id;


```

```

select f.film_id, f.title , sum (f.length) as total_length
from film f
left join film_category fc
    on fc.film_id = f.film_id
where fc.category_id = 1
group by f.film_id
order by f.film_id;

```

\*\* Se puede observar en el script aportado 50.Durección total de películas de Action.sql\*\*

- 2.51. Crea una tabla temporal llamada “cliente\_rentas\_temporal” para almacenar el total de alquileres por cliente.

La consulta sería:

```

④--La consulta quedaría así:
with Cliente_rentas_temporal as (
    select customer_id, count(rental_id) as Total_rent
    from rental r
    group by customer_id
    order by customer_id
)
select *
from Cliente_rentas_temporal ;

```

```

with Cliente_rentas_temporal as (
    select customer_id, count(rental_id) as Total_rent
    from rental r
    group by customer_id
    order by customer_id
)
select *
from Cliente_rentas_temporal ;

```

\*\* Se puede observar en el script aportado 51.Tabla temporal cliente\_rentas\_temporal.sql\*\*

- 2.52. Crea una tabla temporal llamada “peliculas\_alquiladas” que almacene las películas que han sido alquiladas al menos 10 veces.

La consulta sería:

```

④with peliculas_alquiladas as (
    select i.film_id , count (r.rental_id) as count_rent
    from rental r
    right join inventory i
        on r.inventory_id = i.inventory_id
    group by i.film_id
    order by i.film_id
)
select *
from peliculas_alquiladas
where count_rent > 10 ;

```

```

with peliculas_alquiladas as (
select i.film_id , count (r.rental_id) as count_rent
from rental r
right join inventory i
    on r.inventory_id = i.inventory_id

```

```

        group by i.film_id
        order by i.film_id
    )
    select *
    from peliculas_alquiladas
    where count_rent > 10 ;

```

\*\* Se puede observar en el script aportado 52.Tabla temporal peliculas\_alquiladas más de 10 veces.sql\*\*

- 2.53. 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.

La consulta sería:

```

④-- La consulta quedaría así:
select r.customer_id,
       concat(c.first_name, ' ', c.last_name) as customers ,
       f.title ,
       r.rental_date,
       r.return_date
  from rental r
 right join inventory i
    on r.inventory_id = i.inventory_id
   join film f
    on f.film_id = i.film_id
   join customer c
    on r.customer_id = c.customer_id
  where r.customer_id = 75 and r.return_date is null
  order by f.title ;

select r.customer_id,
       concat(c.first_name, ' ', c.last_name) as customers ,
       f.title ,
       r.rental_date,
       r.return_date
  from rental r
 right join inventory i
    on r.inventory_id = i.inventory_id
   join film f
    on f.film_id = i.film_id
   join customer c
    on r.customer_id = c.customer_id
  where r.customer_id = 75 and r.return_date is null
  order by f.title ;

```

\*\* Se puede observar en el script aportado 53.Peliculas alquiladas por TAMMY SANDERS sin devolver.sql\*\*

El resultado ha sido:

	customer_id	customers	title	rental_date	return_date
1	75	TAMMY SANDERS	LUST LOCK	2006-02-14 15:16:03.000	[NULL]
2	75	TAMMY SANDERS	SLEEPY JAPANESE	2006-02-14 15:16:03.000	[NULL]
3	75	TAMMY SANDERS	TROUBLE DATE	2006-02-14 15:16:03.000	[NULL]

- 2.54. 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.

La consulta quedaría:

```
/* --En esta consulta se anidan 3 tablas:  
select a.first_name, a.last_name , c."name", count (fa.film_id) as film_number  
from film_actor fa  
join film_category fc  
    on fa.film_id = fc.film_id  
join category c  
    on fc.category_id = c.category_id  
join actor a  
    on fa.actor_id = a.actor_id  
where c."name" = 'Sci-Fi'  
group by a.first_name, a.last_name , c."name"  
order by a.last_name ;
```

```
select a.first_name, a.last_name , c."name", count (fa.film_id) as film_number  
from film_actor fa  
join film_category fc  
    on fa.film_id = fc.film_id  
join category c  
    on fc.category_id = c.category_id  
join actor a  
    on fa.actor_id = a.actor_id  
where c."name" = 'Sci-Fi'  
group by a.first_name, a.last_name , c."name"  
order by a.last_name ;
```

\*\* Se puede observar en el script aportado 54.Nombre de actores que han actuado en la categoría Sci-Fi.sql\*\*

- 2.55. 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.

La consulta quedaría así:

```
/*/*Para realizar esta consulta,  
 * vamos a ver cuando se alquiló por primera vez  
 * la película 'Spartacus Cheaper'  
 */  
select film_id  
from film f  
where title = 'SPARTACUS CHEAPER' ;  
  
/*--Ya sabemos que nuestro ID buscado de la película es el 824  
-- Ahora vamos a ver cuando se alquiló por primera vez ese ID.  
select r.inventory_id , r.rental_date  
from inventory i  
join rental r  
    on i.inventory_id = r.inventory_id  
where i.film_id = 824  
order by r.rental_date;
```

```


    --Ya sabemos que se alquiló por primera vez el 2005-07-08 06:43:42.000
    --Ahora procedemos hacer el resto de la consulta:
    select a.first_name, a.last_name, count (fa.film_id) as film_numbers
    from actor a
    join film_actor fa
        on a.actor_id = fa.actor_id
    join inventory i
        on i.film_id = fa.film_id
    join rental r
        on i.inventory_id = r.inventory_id
    where r.rental_date > '2005-07-08'
    group by a.first_name, a.last_name
    order by a.last_name;
    /*Para que la consulta quedara más completa,
     * hemos creado otra columna donde nos da la información del número
     * de películas en las que ha actuado cada actor.
    */


```

```


select a.first_name, a.last_name, count (fa.film_id) as film_numbers
from actor a
join film_actor fa
    on a.actor_id = fa.actor_id
join inventory i
    on i.film_id = fa.film_id
join rental r
    on i.inventory_id = r.inventory_id
where r.rental_date > '2005-07-08'
group by a.first_name, a.last_name
order by a.last_name;


```

\*\* Se puede observar en el script aportado 55.Nombre y apellidos de actores que han actuado en películas.sql\*\*

2.56. Encuentra el nombre y apellido de los actores que no han actuado en ninguna película de la categoría 'Music'.

La consulta sería:

```


    -- Para hacer esta consulta tenemos que anidar 4 tablas:
    select a.first_name, a.last_name, c."name"
    from category c
    join film_category fc
        on c.category_id = fc.category_id
    join film_actor fa
        on fc.film_id = fa.film_id
    join actor a
        on a.actor_id = fa.actor_id
    where c."name" <> 'Music';


```

```


select a.first_name, a.last_name, c."name"
from category c
join film_category fc
    on c.category_id = fc.category_id
join film_actor fa
    on fc.film_id = fa.film_id
join actor a
    on a.actor_id = fa.actor_id
where c."name" <> 'Music';


```

\*\* Se puede observar en el script aportado 56.Nombre y apellidos de actores que no han actuado en películas de Music.sql\*\*

2.57. Encuentra el título de todas las películas que fueron alquiladas por más de 8 días.

La consulta sería:

```
-- La consulta quedaría:  
select f.title, (r.return_date - r.rental_date) as difference  
from rental r  
join inventory i  
    on r.inventory_id = i.inventory_id  
join film f  
    on i.film_id = f.film_id  
where (r.return_date - r.rental_date) > '8 days' ;
```

```
select f.title, (r.return_date - r.rental_date) as difference  
from rental r  
join inventory i  
    on r.inventory_id = i.inventory_id  
join film f  
    on i.film_id = f.film_id  
where (r.return_date - r.rental_date) > '8 days' ;
```

\*\* Se puede observar en el script aportado 57.Título de películas alquiladas por más de 8 días.sql\*\*

El resultado sería:

	A-Z title	difference
1	LOVE SUICIDES	9 days 02:39:00
2	IDOLS SNATCHERS	8 days 05:28:00
3	WHALE BIKINI	8 days 20:47:00
4	PELICAN COMFORTS	9 days 02:51:00
5	HARRY IDAHO	8 days 03:57:00
6	RIDGEMONT SUBMARINE	8 days 02:14:00
7	ALLEY EVOLUTION	9 days 02:10:00
8	ROSES TREASURE	9 days 01:45:00
9	PATIENT SISTER	9 days 00:30:00
10	RESERVOIR ADAPTATION	8 days 22:43:00
11	METROPOLIS COMA	8 days 01:53:00

2.58. Encuentra el título de todas las películas que son de la misma categoría que 'Animation'.

La consulta sería:

```
-- La consulta quedaría:  
select f.title  
from film f  
join film_category fc  
    on f.film_id = fc.film_id  
where fc.category_id = 2 ;
```

```
select f.title  
from film f  
join film_category fc  
    on f.film_id = fc.film_id  
where fc.category_id = 2 ;
```

\*\* Se puede observar en el script aportado 58.Título de películas con la categoría Action.sql\*\*

- 2.59. 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.

La consulta quedaría:

```
④-- Vamos a localizar la dirección de la película Dancing Fever:  
select *  
from film f  
where title = 'DANCING FEVER'  
  
④--Es 144, por lo tanto la consulta quedaría:  
select title, length  
from film f  
where length =144  
order by title ;  
  
select title, length  
from film f  
where length =144  
order by title ;
```

\*\* Se puede observar en el script aportado 59.Nombre de películas con la misma duración que Dancing Fever.sql\*\*

El resultado sería:

	A-Z title	123 length
1	DANCING FEVER	144
2	FACTORY DRAGON	144
3	LAMBS CINCINATTI	144
4	PACIFIC AMISTAD	144
5	PRESIDENT BANG	144
6	STRICTLY SCARFACE	144
7	TOWERS HURRICANE	144
8	VIRTUAL SPOILERS	144

- 2.60. Encuentra los nombres de los clientes que han alquilado al menos 7 películas distintas. Ordena los resultados alfabéticamente por apellido.

La consulta sería:

```
④--La consulta quedaría:  
select c.first_name, c.last_name,  
       count (distinct (i.film_id)) as film_number  
  from customer c  
join inventory i  
    on i.store_id = c.store_id  
group by c.first_name, c.last_name  
order by c.last_name;  
  
select c.first_name, c.last_name,  
       count (distinct (i.film_id)) as film_number  
  from customer c  
join inventory i  
    on i.store_id = c.store_id
```

```

group by c.first_name, c.last_name
order by c.last_name;

```

\*\* Se puede observar en el script aportado 60.Nombres de clientes que han alquilado al menos 7 películas distintas.sql\*\*

2.61. Encuentra la cantidad total de películas alquiladas por categoría y muestra el nombre de la categoría junto con el recuento de alquileres.

La consulta quedaría:

```

--La consulta sería:
select c."name" as category_name,
       count (i.film_id) as total_film
  from inventory i
  join rental r
    on i.inventory_id = r.inventory_id
  join film_category fc
    on i.film_id = fc.film_id
  join category c
    on fc.category_id = c.category_id
   where r.rental_id > 0
  group by c."name";

```

```

select c."name" as category_name,
       count (i.film_id) as total_film
  from inventory i
  join rental r
    on i.inventory_id = r.inventory_id
  join film_category fc
    on i.film_id = fc.film_id
  join category c
    on fc.category_id = c.category_id
   where r.rental_id > 0
  group by c."name";

```

\*\* Se puede observar en el script aportado 61.Cantidad total de películas alquiladas por categoría.sql\*\*

El resultado sería:

	A-Z category_name	123 total_film
1	Sports	1.179
2	Classics	939
3	New	940
4	Family	1.096
5	Comedy	941
6	Animation	1.166
7	Travel	837
8	Music	830
9	Drama	1.060
10	Horror	846
11	Sci-Fi	1.101
12	Games	969
13	Documentary	1.050
14	Foreign	1.033
15	Action	1.112
16	Children	945

## 2.62. Encuentra el número de películas por categoría estrenadas en 2006.

La consulta quedaría:

```
--La consulta sería:  
select c."name" as category_name,  
       count (f.film_id) as total_film  
  from film f  
  join film_category fc  
    on f.film_id = fc.film_id  
  join category c  
    on c.category_id = fc.category_id  
   where f.release_year = '2006'  
   group by c."name";  
  
select c."name" as category_name,  
       count (f.film_id) as total_film  
  from film f  
  join film_category fc  
    on f.film_id = fc.film_id  
  join category c  
    on c.category_id = fc.category_id  
   where f.release_year = '2006'  
   group by c."name";
```

\*\* Se puede observar en el script aportado 62.Número de películas por categoría estrenadas en 2006.sql\*\*

El resultado ha sido:

	category_name	total_film
1	Sports	74
2	Classics	57
3	New	63
4	Family	69
5	Comedy	58
6	Animation	66
7	Travel	57
8	Music	51
9	Drama	62
10	Horror	56
11	Sci-Fi	61
12	Games	61
13	Documentary	68
14	Foreign	73
15	Action	64
16	Children	60

## 2.63. Obtén todas las combinaciones posibles de trabajadores con las tiendas que tenemos.

La consulta sería:

```
--La consulta quedaría:  
select s2.staff_id,  
       s2.first_name,  
       s2.last_name,  
       s.store_id  
  from store s  
cross join staff s2 ;
```

```

select s2.staff_id,
       s2.first_name,
       s2.last_name,
       s.store_id
  from store s
 cross join staff s2 ;

```

\*\* Se puede observar en el script aportado 63.Combinaciones posibles de trabajadores con tiendas.sql\*\*

El resultado ha sido:

	123 ↗ staff_id	A-Z first_name	A-Z last_name	123 ↘ store_id
1	1	Mike	Hillyer	1
2	2	Jon	Stephens	1
3	1	Mike	Hillyer	2
4	2	Jon	Stephens	2

2.64. 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.

La consulta sería:

```

--La consulta quedaría:
select c.customer_id,
       c.first_name,
       c.last_name,
       count (film_id) as total_film
  from rental r
 join customer c
    on r.customer_id = c.customer_id
 join inventory i
    on i.inventory_id = r.inventory_id
 group by c.customer_id, c.first_name, c.last_name;

select c.customer_id,
       c.first_name,
       c.last_name,
       count (film_id) as total_film
  from rental r
 join customer c
    on r.customer_id = c.customer_id
 join inventory i
    on i.inventory_id = r.inventory_id
 group by c.customer_id, c.first_name, c.last_name;

```

\*\* Se puede observar en el script aportado 64.Cantidad total de películas alquiladas por cliente.sql\*\*

### 3. Herramientas para realizar el proyecto

- PostgreSQL
- DBeaver