






# MODULO 2 - Base de Datos | EXAMEN | Ignacio Cavallo

---

## Comentarios

Esta entrega se divide en los dos ejercicios que fueron suministrados. El pedido por el profesor en clases, más el que está en la plataforma.

Aclaraciones:

1. TODO el [script](#) está al final del documento. 
2. Se desarrolló tanto el [ejercicio del profesor](#) como también el que se encuentra en la [plataforma](#). 
3. Hice muchas consultas que no están especificadas en este README, pero si están al final. 
4. Todos los pasos en el desarrollo del esquema están en mi repositorio sobre este módulo. 
5. Al ser un documento *demasiado* extenso, me tomé la libertad de mostrar primero las consultas más avanzadas. Es por ello que JOIN está listada como 0.:thinking:
6. En el apartado de JOIN puedo demostrar lo aprendido en clase junto a otros conceptos como DATEDIFF ó UNION. :nerd\_face:
7. Espero que sea al menos entretenida esta lectura. **Muchas Gracias por su tiempo y que le sea leve!** 

## EJERCICIO 1

### Ejercicio final

Pensar en una problemática para la cual se deba diseñar una base de datos, y en base a eso realizar lo siguiente:

- Elaborar el modelo entidad - relación (incluir en el documento)
- Elaborar el modelo relacional en Workbench (incluir en el documento)
- Copiar el Script generado (incluir en el documento)
- Crear registros en la base de datos y hacer consultas utilizando todo lo visto en clases y lo que ustedes buscaron por su cuenta (mínimo 2 consultas por cada operador clausula o comando), dichas consultas deben tener la descripción, query asociada a ella y el resultado mostrado (incluir en el documento).
- Crear vistas.

## Problemática

Una de mis pasiones es el entretenimiento audio visual, por eso decidí hacer una pequeña base de datos que contenga algunas de mis películas favoritas.

Básicamente voy a implementar mi propio IMDB (*Ignacio* Movie Data Base). 😊



Por ende consideré las siguientes entidades:

- Película
- Actor
- Director
- Género
- Critica

Con estas entidades, es hora de ver las relaciones que existen entre ellas:

- Un actor puede estar en muchas películas como también muchos actores pueden estar en muchas películas.

Actor <==> Películas

- Una película puede ser dirigida por múltiples directores, como también un director dirige muchas películas.

Director <==> Películas

- El género de una película puede ser más de uno(Comedia Dramática) y todas las películas tienen género.

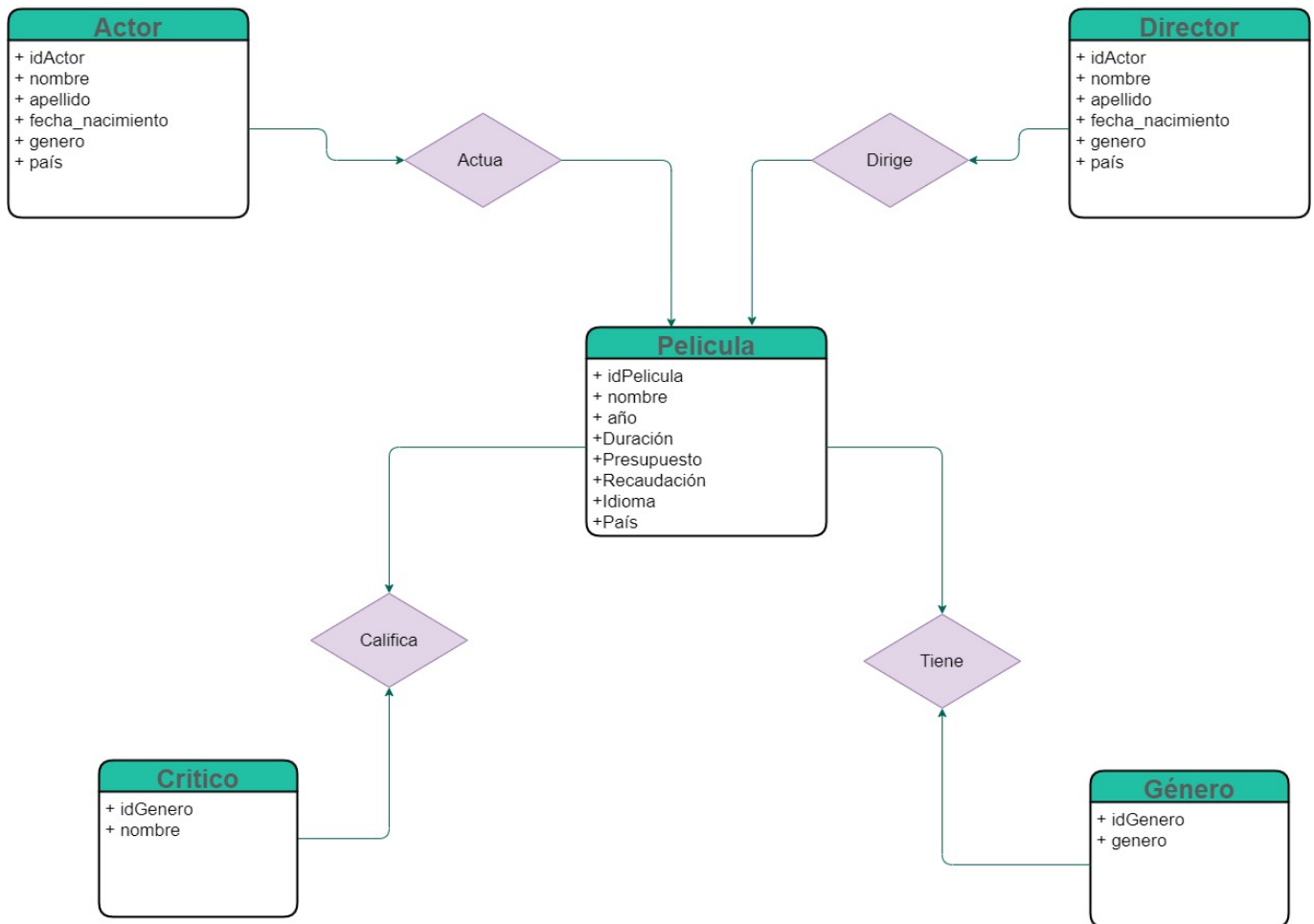
Género <==> Películas

- Por último, una película tiene criticas, y a su vez hay muchos críticos que califican a muchas películas.

Critica <==> Películas

## 1. MODELO ENTIDAD-RELACIÓN

### Películas | Ignacio CAVALLLO



Entonces, considerando lo anterior, al tener todas entidades con relaciones muchos a muchos **N:M**, se nos generaran las siguientes tablas intermedias:

Un actor que desempeña un rol en un film, se convierte en un personaje, por ende:

```
Actor ==> Personaje <== Películas
```

Una película con un director tiene una dirección:

```
Director ==> Dirección <== Películas
```

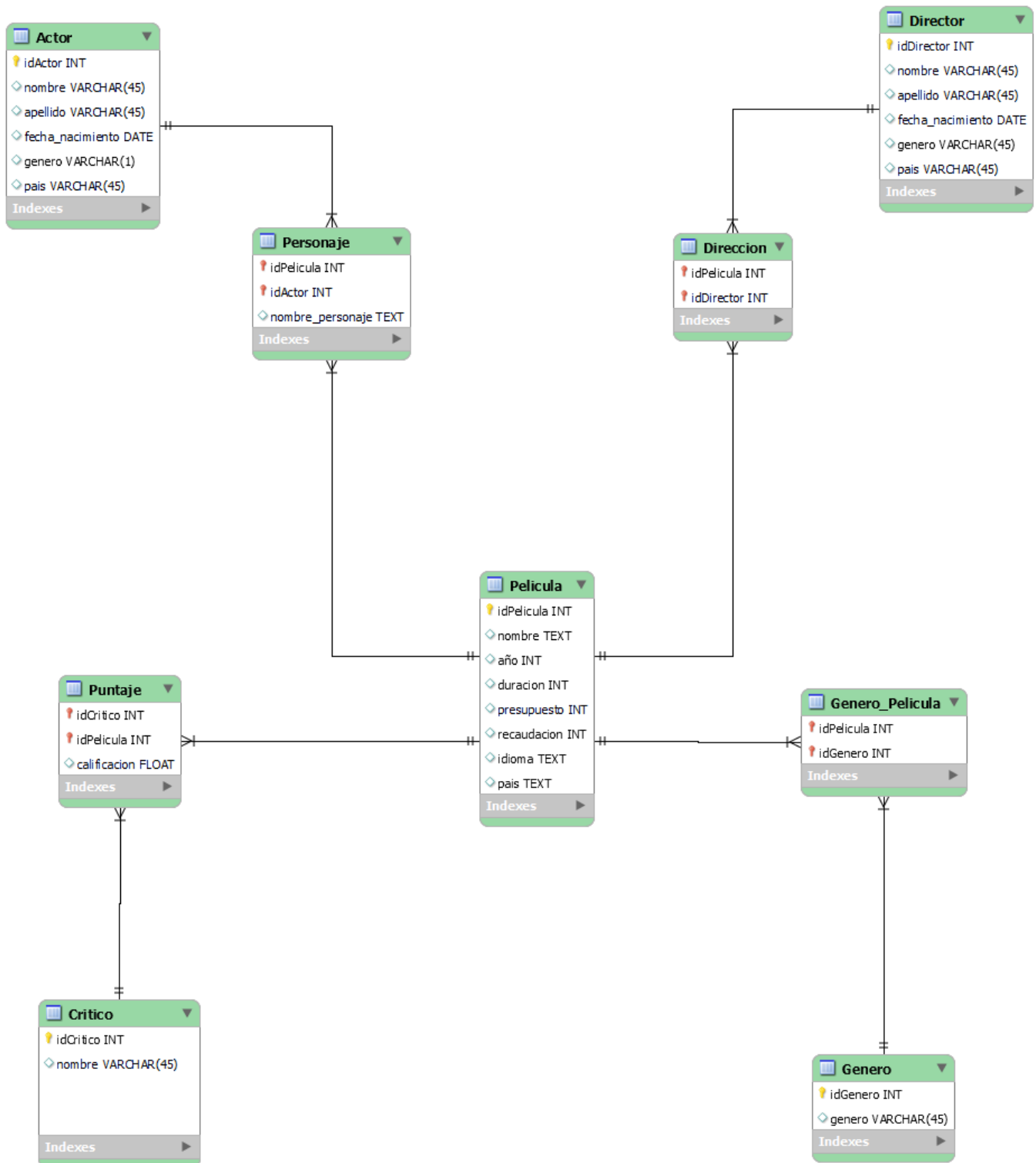
Una película tiene muchos géneros:

```
Genero ==> Género de Película <== Películas
```

Un critico al calificar una película, brinda un Puntaje:

Crítico ==> Puntaje <== Películas

## 2. MODELO RELACIONAL

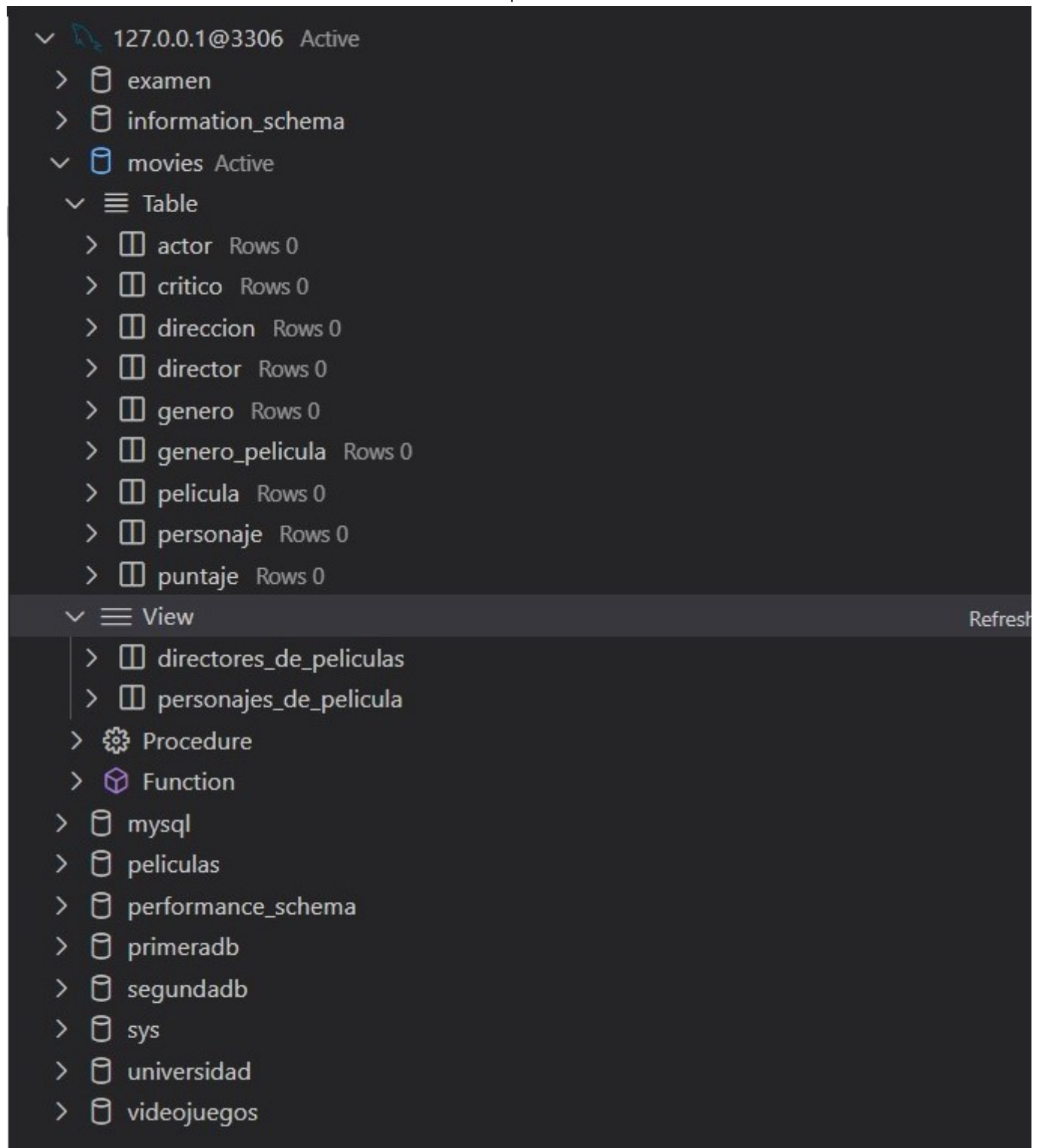


## 3. SCRIPT

El script está al final del documento!.

## 4. CONSULTAS

Con el fin de contextualizar, mostraré como está el esquema de mi base de datos.



En este apartado mostraré diferentes queries con la siguiente estructura:

**COMANDO | CLAUSULA**

**DESCRIPCION**

**QUERY** en formato de código.

**RESULTADO** en formato de imagen.

## 0. JOIN | WHERE | HAVING | GROUP BY |

Al tener tablas intermedias, estamos obligados a usar join, para poder relacionar las entidades que se definieron en el modelo. A continuación veremos varios ejemplos interesantes en los que se aplica todo lo aprendido en clase, más algunos conceptos que aprendí por mi cuenta. Por ejemplo, calcular la edad en función a una fecha, calcular la edad de un actor con respecto a la fecha de estreno, aplicar redondeo a un promedio, entre otras.

### DESCRIPCIÓN

Mostrar todas las películas de *acción*,

### QUERY

```
SELECT
    p.nombre
FROM
    pelicula p
    INNER JOIN
    genero_pelicula gp ON p.idPelicula = gp.idPelicula
    inner JOIN
    genero g ON g.idGenero = gp.idGenero

WHERE g.genero = 'Acción';
```

### RESULTADO

FROM  
pelicula p  
INNER JOIN

Input To Search Data

		* nombre int
		Filter
1		The Matrix
2		Hero
3		Memento
4		Captain fantastic
5		Inception
6		Snatch
7		Two smoking barrels
8		John Wick
9		El Señor de los Anillos
10		The Dark Knight
11		V for Vendetta
12		La casa de las dagas voladoras

## DESCRIPCIÓN

Mostrar todas las películas con sus puntajes disgregados por Crítico.  
Agregar una columna que muestre el promedio de los tres críticos

## QUERY

```
SELECT
  pelicula.nombre,
  imdb.calificacion AS IMDB,
  ff.calificacion AS Film_Affinity,
```

```

ic.calificacion AS Ignacio_Cavallo,
ROUND((imdb.calificacion + ff.calificacion + ic.calificacion)/3.0,2) as
Promedio
FROM
pelicula
INNER JOIN
puntaje imdb ON pelicula.idPelicula = imdb.idPelicula
INNER JOIN
puntaje ff ON pelicula.idPelicula = ff.idPelicula
AND ff.idCritico = 2
INNER JOIN
puntaje ic ON pelicula.idPelicula = ic.idPelicula
AND ic.idCritico = 3
GROUP BY pelicula.idPelicula;

```

## RESULTADO

ic.calificacion AS Ignacio_Cavallo,						
Input To Search Data						
Cost: 7ms < 1 > Total 17						
		* nombre int	IMDB text	Film_Affinity int	Ignacio_Cavallo int	Promedio int
		Filter	Filter	Filter	Filter	Filter
1		The Matrix	8.7	7.9	8.5	8.37
2		Hero	7.9	7.3	8.2	7.80
3		Memento	7.4	7.9	9	8.10
4		Captain fantastic	7.9	7.5	8.7	8.03
5		Inception	8.8	8	8.8	8.53
6		Snatch	8.3	7.9	9	8.40
7		Two smoking barrels	8.2	7.8	7.2	7.73
8		John Wick	7.4	6.3	6.3	6.67
9		El Señor de los Anillos	8.8	8	8.1	8.30
10		9 reinas	7.9	7.8	8.2	7.97
11		El secreto de sus ojos	8.2	8.1	8.4	8.23
12		The Dark Knight	9	8.1	9.2	8.77
13		V for Vendetta	8.1	7.5	6.6	7.40
14		La casa de las dagas voladoras	6.9	7	5.9	6.60
15		The Wolf of Wall Street	8.2	7.6	8.1	7.97
16		The Curious Case of Benjamin Button	7.8	7.2	7.9	7.63
17		What We Do In The Shadows	7.7	6.8	7.2	7.23

## DESCRIPCIÓN

Mostrar el elenco y sus personajes de *Matrix*

## QUERY








```


SELECT actor.nombre, actor.apellido, personaje.nombre_personaje
FROM
    actor
    INNER JOIN
    personaje ON actor.idActor = personaje.idActor
    JOIN
    pelicula ON pelicula.idPelicula = personaje.idPelicula
WHERE pelicula.nombre = 'The Matrix';

```

## RESULTADO

SELECT actor.nombre, actor.apellido, personaje.nombre\_personaje  
FROM  
actor  
INNER JOIN  
personaje ON actor.idActor = personaje.idActor

Input To Search Data      Cost: 1ms < 1 > Total 4

<input checked="" type="checkbox"/>		* nombre int	apellido varchar(45)	nombre_personaje varchar(45)
		Filter	Filter	Filter
1		Keanu	Reves	Neo
2		Carrie-Anne	Moss	Trinity
3		Joe	Pantoliano	Cypher
4		Hugo	Weaving	Agente Smith

## DESCRIPCIÓN

Mostrar aquellos directores que tengan *más de una* película.

## QUERY

```

SELECT
    director.nombre,
    director.apellido,
    COUNT(direccion.idDirector) AS cantidad_peliculas
FROM
    director
    INNER JOIN
    direccion ON director.idDirector = direccion.idDirector

```

```
GROUP BY director.nombre , director.apellido
HAVING cantidad_peliculas > 1;
```

## RESULTADO

```
SELECT
  director.nombre,
  director.apellido,
  COUNT(direccion.idDirector) AS cantidad_peliculas
FROM
  ..
```

Input To Search Data							Cost: 3ms	< 1 > Total
		* nombre int		apellido varchar(45)		cantidad_peliculas varchar(45)		
		Filter		Filter		Filter		
1		Lana		Wachowski		2		
2		Lily		Wachowski		2		
3		Christopher		Nolan		3		
4		Guy		Ritchie		2		

## 1. SELECT

### DESCRIPCIÓN

Mostrar todos los valores, de película, actor y director

### QUERY

```
SELECT * FROM pelicula;
SELECT * FROM actor;
SELECT * FROM director;
```

## RESULTADO

SELECT * FROM pelicula LIMIT 100									
Input To Search Data		Cost: 2ms < 1 > Total 17							
	<div><div></div><div>Q</div></div>	<div><div>* idPelicula</div><div>int</div></div>	<div><div>nombre</div><div>text</div></div>	<div><div>año</div><div>int</div></div>	<div><div>duracion</div><div>int</div></div>	<div><div>presupuesto</div><div>int</div></div>	<div><div>recaudacion</div><div>int</div></div>	<div><div>idioma</div><div>text</div></div>	<div><div>pais</div><div>text</div></div>
		Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
	1	1	The Matrix	1999	138	63000000	465000000	Ingles	USA
	2	2	Hero	2002	119	85000000	29980000	Chino	China
	3	3	Memento	2000	113	90000000	39723096	Ingles	USA
	4	4	Captain fantastic	2016	118	5000000	22000000	Ingles	USA
	5	5	Inception	2010	148	160000000	825532764	Ingles	USA
	6	6	Snatch	2000	99	10000000	83600000	Ingles	UK
	7	7	Two smoking barrels	1998	105	1400000	28000000	Ingles	UK
	8	8	John Wick	2014	114	20000000	86013268	Ingles	USA
	9	9	El seños de los anillos	2001	178	93000000	870761744	Ingles	NZ
	10	10	9 reinas	2000	115	1300000	12413888	Español	Argentina
	11	11	El secreto de sus ojos	2009	129	2000000	35079650	Español	Argentina
	12	12	The Dark Knight	2008	152	185000000	1004558444	Ingles	UK
	13	13	V for Vendetta	2005	132	54000000	425511035	Ingles	USA
	14	14	La casa de las dagas voladoras	2004	119	30000000	50000000	Chino	China
	15	15	The Wolf of Wall Street	2013	180	100000000	392000694	Ingles	USA
	16	16	The Curious Case of Benjamin Button	2008	116	150000000	333932083	Ingles	USA
	17	17	What We Do In The Shadows	2014	86	1600000	7253160	Ingles	NZ

SELECT * FROM actor							
Input To Search Data		Cost: 6ms < 1 > Total 21					
	<div><div></div><div>Q</div></div>	<div><div>* idActor</div><div>int</div></div>	<div><div>nombre</div><div>varchar(45)</div></div>	<div><div>apellido</div><div>varchar(45)</div></div>	<div><div>fecha_nacimiento</div><div>date</div></div>	<div><div>genero</div><div>varchar(1)</div></div>	<div><div>pais</div><div>varchar(45)</div></div>
		Filter	Filter	Filter	Filter	Filter	Filter
	1	1	Ricardo	Darin	1957-01-16	m	Argentina
	2	2	Gastón	Pauls	1972-01-17	m	Argentina
	3	3	Guillermo	Francella	1955-02-14	m	Argentina
	4	4	Keanu	Reves	1964-09-02	m	USA
	5	5	Carrie-Anne	Moss	1967-08-21	f	Canada
	6	6	Víggó	Mortensen	1958-10-20	m	USA
	7	7	Jet	Li	1963-04-26	m	China
	8	8	Guy	Pearce	1967-10-05	m	UK
	9	9	Joe	Pantoliano	1951-09-12	m	USA
	10	10	Leonardo	DiCaprio	1974-11-11	m	USA
	11	11	Jason	Statham	1966-07-26	m	UK
	12	12	Micael	Cane	1933-03-14	m	UK
	13	13	Hugo	Weaving	1960-04-04	m	UK
	14	14	Elijah	Wood	1981-01-28	m	USA
	15	15	Brad	Pitt	1963-12-17	m	USA
	16	16	Christian	Bale	1974-01-30	m	UK
	17	17	Ziyi	Zhang	1979-02-09	f	China
	18	18	Natalia	Dortman	1981-07-09	f	USA

main\*

Python 3.9.2 64-bit

0 5 0 8

✓ markdown | ✓ README.md

127.0.0.1 movies

tabnine

		* idDirector int	nombre varchar(45)	apellido varchar(45)	fecha_nacimiento date	genero varchar(45)	pais varchar(45)
		Filter	Filter	Filter	Filter	Filter	Filter
	1	1	Lana	Wachowski	1965-04-21	f	USA
	2	2	Lily	Wachowski	1967-12-29	f	USA
	3	3	Zhang	Yimou	1951-11-14	m	China
	4	4	Christopher	Nolan	1970-07-30	m	UK
	5	5	Matt	Ross	1970-01-03	m	USA
	6	6	Guy	Ritchie	1968-09-10	m	UK
	7	7	Chad	Stahelski	1968-09-20	m	USA
	8	8	Peter	Jackson	0000-00-00	m	NZ
	9	9	Fabian	Bielinsky	1959-02-03	m	Argentina
	10	10	Juan José	Campanella	1959-07-19	m	Argentina
	11	11	Martin	Scorsese	1942-11-17	m	USA
	12	12	David	Fincher	1962-08-28	m	USA
	13	13	Taika	Waititi	1975-08-16	m	NZ
	14	14	Jemaine	Clement	1974-01-10	m	NZ

## 2. UPDATE

Como vemos me equivoqué al cargar algunos campos, vamos a modificarlos:

### DESCRIPCIÓN

Cambiar el nombre de "El seños de los anillos" por 'El Señor de los Anillos'

### QUERY

```
SELECT * FROM pelicula
WHERE pelicula.idPelicula=9;
UPDATE pelicula
SET nombre = 'El Señor de los Anillos'
WHERE pelicula.idPelicula=9;
```

### RESULTADO

SELECT * FROM pelicula WHERE pelicula.idPelicula=9									
Input To Search Data <span>🔄</span> <span>⊕</span> <span>🗑️</span> <span>⬇️</span> <span>▶️</span> Cost: 4ms < 1 > Total 1									
	<input checked="" type="checkbox"/>	idPelicula int	nombre text	año int	duracion int	presupuesto int	recaudacion int	idioma text	pais text
		Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
<input checked="" type="checkbox"/>	1	9	El seños de los anillos	2001	178	93000000	870761744	Ingles	NZ

SELECT * FROM pelicula WHERE pelicula.idPelicula=9									
Input To Search Data <span>🔄</span> <span>⊕</span> <span>🗑️</span> <span>⬇️</span> <span>▶️</span> Cost: 1ms < 1 > Total 1									
EXECUTE SUCCESS:									
UPDATE pelicula SET nombre = 'El Señor de los Anillos' WHERE pelicula.idPelicula=9									
AffectedRows : 1									
	<input checked="" type="checkbox"/>	idPelicula int	nombre text	año int	duracion int	presupuesto int	recaudacion int	idioma text	pais text
		Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
<input checked="" type="checkbox"/>	1	9	El Señor de los Anillos	2001	178	93000000	870761744	Ingles	NZ

### 3. WHERE + CONDICIÓN

Algunas búsquedas con condiciones.

#### DESCRIPCIÓN

Buscar todos los nombres de películas que *NO* sean de Estados Unidos ó Inglaterra.

#### QUERY

```
SELECT nombre, pais FROM pelicula
WHERE pais NOT IN ('USA', 'UK');
```

#### RESULTADO

SELECT nombre, pais FROM pelicula WHERE pais NOT IN ('USA', 'UK')			
Input To Search Data			Cost: 2ms
		* nombre int	pais text
		Filter	Filter
	1	Hero	China
	2	El Señor de los Anillos	NZ
	3	9 reinas	Argentina
	4	El secreto de sus ojos	Argentina
	5	La casa de las dagas voladoras	China
	6	What We Do In The Shadows	NZ

## DESCRIPCIÓN

Mostrar nombre y apellido de todas las *ACTRICES* excluyendo a los hombres

## QUERY

```
SELECT nombre, apellido FROM actor
WHERE genero <> 'm';
```

## RESULTADO

SELECT nombre, apellido FROM actor WHERE genero <> 'm'			
Input To Search Data			Cost: 1ms
<input type="checkbox"/>		* nombre int	apellido varchar(45)
		Filter	Filter
<input type="checkbox"/>	1	Carrie-Anne	Moss
<input type="checkbox"/>	2	Ziyi	Zhang
<input type="checkbox"/>	3	Natalie	Portman
<input type="checkbox"/>	4	Cate	Blanchet

## DESCRIPCIÓN

Mostrar nombre y apellido de todos los actores cuyos nombres comiencen con 'J' y contengan al menos tres caracteres después de la j.


## QUERY


```
SELECT nombre, apellido FROM actor
WHERE nombre LIKE 'j___%';
```


## RESULTADO


SELECT nombre, apellido FROM actor  
WHERE nombre LIKE 'j\_\_%'


Input To Search Data














Cost: 1ms

<div><input type="checkbox"/></div>	<div></div>	<div><div><div>* nombre</div><div>int</div></div><div></div></div>	<div><div><div>apellido</div><div>varchar(45)</div></div><div></div></div>
		Filter	Filter
<div><input type="checkbox"/></div>	1	Jason	Statham
<div><input type="checkbox"/></div>	2	Jemaine	Clement

#### 4. GROUP BY | ORDER BY

Algunas búsquedas agrupadas y ordenadas.

##### DESCRIPCIÓN






Cuántas películas por país.

##### QUERY

```
SELECT pais, COUNT(*) AS cantidad
FROM pelicula
GROUP BY pais;
```

##### RESULTADO



SELECT pais, COUNT(*) AS cantidad FROM pelicula GROUP BY pais			
Input To Search Data		    	Cost: 2ms
		* pais int	cantidad text
		Filter	Filter
	1	USA	8
	2	China	2
	3	UK	3
	4	NZ	2
	5	Argentina	2















## DESCRIPCIÓN

Ingresos por país por año, ordenado por país

## QUERY

```
SELECT pais, año, SUM(recaudacion) as Recaudación
FROM pelicula
GROUP BY año
ORDER BY pais;
```

## RESULTADO

		* pais int	año text	Recaudación int
		Filter	Filter	Filter
	1	Argentina	2009	35079650
	2	China	2002	29980000
	3	China	2004	50000000
	4	NZ	2001	870761744
	5	UK	1998	28000000
	6	UK	2008	1338490527
	7	USA	1999	465000000
	8	USA	2000	135736984
	9	USA	2016	22000000
	10	USA	2010	825532764
	11	USA	2014	93266428
	12	USA	2005	425511035
	13	USA	2013	392000694

## 5. SUB QUERY

Algunos ejemplos de SubQuery

### DESCRIPCIÓN

Mostrar la película con la *MAYOR* recaudación

### QUERY

```
SELECT nombre
FROM pelicula
WHERE recaudacion =
(SELECT MAX(recaudacion) FROM pelicula);
```

## RESULTADO

```
SELECT nombre
FROM pelicula
WHERE recaudacion =
(SELECT MAX(recaudacion) FROM pelicula)
```

Input To Search Data

<input checked="" type="checkbox"/>	<input type="text"/>	* nombre int
		Filter
	1	The Dark Knight

## DESCRIPCIÓN

Mostrar la película con la *MENOR* recaudación

## QUERY

```
SELECT nombre
FROM pelicula
WHERE recaudacion =
(SELECT MIN(recaudacion) FROM pelicula);
```

## RESULTADO

```
SELECT nombre
FROM pelicula
WHERE recaudacion =
(SELECT MIN(recaudacion) FROM pelicula)
```

Input To Search Data

<input checked="" type="checkbox"/>	<input type="text"/>	* nombre int
		Filter
	1	What We Do In The Shadows

Algunas búsquedas con condiciones.

## DESCRIPCIÓN

Mostrar todas aquellas películas cuyo ingreso fué mayor al *promedio* de recaudaciones generales.

## QUERY

```
SELECT nombre
FROM pelicula
WHERE recaudacion >
(SELECT AVG(recaudacion) FROM pelicula);
```

## RESULTADO



The screenshot shows a SQL query execution interface. At the top, the query is displayed: `SELECT nombre FROM pelicula WHERE recaudacion > (SELECT AVG(recaudacion) FROM pelicula)`. Below the query, there is a search bar labeled "Input To Search Data" and several icons (a magnifying glass, a plus sign, a trash can, and a download arrow). The results are shown in a table with two columns: an index and the movie name. The movies listed are The Matrix, Inception, El Señor de los Anillos, The Dark Knight, V for Vendetta, The Wolf of Wall Street, and The Curious Case of Benjamin Button.

	* nombre int
	Filter
1	The Matrix
2	Inception
3	El Señor de los Anillos
4	The Dark Knight
5	V for Vendetta
6	The Wolf of Wall Street
7	The Curious Case of Benjamin Button

## 6. VIEWS

A mi parecer, a algunas de las tablas podrían tener información que a priori no parecía relevante. Por ejemplo, calcular la edad de los actores/directores, cuando filmaron la película. Por ello consideré crear las siguientes

vistas:

## DESCRIPCIÓN

Vista Personaje con su respectivo nombre, apellido, nombre del personaje y la edad del personaje.

Vista Director con su respectivo nombre, apellido, nombre del personaje y la edad del director cuando filmó la película.


## QUERY



```
CREATE VIEW Personajes_de_Pelicula AS
SELECT
    pelicula.nombre AS nombre_pelicula,
    actor.nombre,
    actor.apellido,
    personaje.nombre_personaje,
    pelicula.año - YEAR(actor.fecha_nacimiento) AS edad_personaje
FROM
    actor
    INNER JOIN
    personaje ON actor.idActor = personaje.idActor
    JOIN
    pelicula ON pelicula.idPelicula = personaje.idPelicula
ORDER BY pelicula.nombre;

CREATE VIEW directores_de_peliculas AS
SELECT
    director.nombre,
    director.apellido,
    pelicula.año - YEAR(director.fecha_nacimiento) as edad_director,
    pelicula.nombre as Título
FROM
    director
    INNER JOIN
    direccion ON director.idDirector = direccion.idDirector
    INNER JOIN
    pelicula ON pelicula.idPelicula = direccion.idPelicula;
```

## RESULTADO



		* nombre_pelicula int	nombre text	apellido int	nombre_personaje int	edad_personaje int
		Filter	Filter	Filter	Filter	Filter
	1	9 reinas	Ricardo	Darin	Marcos	43
	2	9 reinas	Gastón	Pauls	Juan	28
	3	Captain fantastic	Viggo	Mortensen	Ben	58
	4	El secreto de sus ojos	Ricardo	Darin	Bejamín Espósito	52
	5	El secreto de sus ojos	Guillermo	Francella	Pablo Sandoval	54
	6	El Señor de los Anillos	Elijah	Wood	Frodo Baggins	20
	7	El Señor de los Anillos	Viggo	Mortensen	Aragon	43
	8	El Señor de los Anillos	Hugo	Weaving	Elrond	41
	9	El Señor de los Anillos	Cate	Blanchet	Galadriel	32
	10	Hero	Jet	Li	Nameless	39
	11	Hero	Ziyi	Zhang	Moon	23
	12	Inception	Leonardo	DiCaprio	Dominick Cobb	36
	13	Inception	Micael	Cane	Miles	77
	14	John Wick	Keanu	Reves	John Wick	50
	15	La casa de las dagas voladoras	Ziyi	Zhang	Xiao Mei	25
	16	Memento	Carrie-Anne	Moss	Natalie	33
	17	Memento	Guy	Pearce	Leonard Shelby	33

		nombre varchar(45)	apellido varchar(45)	edad_director bigint	Título text
		Filter	Filter	Filter	Filter
	1	Lana	Wachowski	34	The Matrix
	2	Lana	Wachowski	40	V for Vendetta
	3	Lily	Wachowski	32	The Matrix
	4	Lily	Wachowski	35	Hero
	5	Zhang	Yimou	53	La casa de las dagas volac
	6	Christopher	Nolan	30	Memento
	7	Christopher	Nolan	40	Inception
	8	Christopher	Nolan	38	The Dark Knight
	9	Matt	Ross	46	Captain fantastic
	10	Guy	Ritchie	32	Snatch

11	Guy	Ritchie	30	Two smoking barrels
12	Chad	Stahelski	46	John Wick
13	Peter	Jackson	2001	El Señor de los Anillos
14	Fabian	Bielinsky	41	9 reinas
15	Juan José	Campanella	50	El secreto de sus ojos
16	Martin	Scorsese	71	The Wolf of Wall Street
17	David	Fincher	46	The Curious Case of Benja







## DESCRIPCIÓN

Mostrar el personaje con *MENOR EDAD* a la hora de filmar. Mostrar el personaje con *MAYOR EDAD* a la hora de filmar.

## QUERY

```
SELECT * FROM personajes_de_pelicula
WHERE edad_personaje = (SELECT MIN(edad_personaje) FROM personajes_de_pelicula)
UNION
SELECT * FROM personajes_de_pelicula
WHERE edad_personaje = (SELECT MAX(edad_personaje) FROM personajes_de_pelicula);
```

## RESULTADO

SELECT * FROM personajes_de_pelicula WHERE edad_personaje = (SELECT MIN(edad_personaje) FROM personajes_de_pelicula) UNION SELECT * FROM personajes_de_pelicula WHERE edad_personaje = (SELECT MAX(edad_personaje) FROM personajes_de_pelicula)					
Input To Search Data					
Cost: 3ms < 1 > Total 2					
<input checked="" type="checkbox"/>		nombre_pelicula text	nombre varchar(45)	apellido varchar(45)	nombre_personaje text
		Filter	Filter	Filter	Filter
1		El Señor de los Anillos	Elijah	Wood	Frodo Baggins
2		Inception	Micael	Cane	Miles

## EJERCICIO 2



## Ejercicio de evaluación Módulo 2 día 5 semana 7

Una parte importante del lenguaje de consultas de bases de datos es su aplicación a las diversas situaciones en los sistemas.

### Objetivo

Aplicar de forma conceptual la información proporcionada en el módulo 2 pudiendo realizar consultas simples a bases de datos para obtener resultados desde las tablas según los enunciados.

-

### Información proporcionada

- Tabla Productos
- Datos en tabla
- Tipo de dato de columna
- Enunciados para elaborar consulta

### Modo de resolución

Según la tabla entregada realice las siguientes consultas SQL para solicitar o realizar operaciones en los datos según los enunciados:

#### Tabla Productos

ProductoID (int)	NombreProducto (varchar)	Descripcion (int)	Precio (int)	Stock (in)
1	Camiseta	Camiseta negra simple de talla única	10	16
2	Pantalón	Pantalón largo azul tipo chino	20	24
3	Gorra	Gorra azul con el logo de los Yankees	15	32
4	Zapatillas	Zapatillas de running de color blanco y verde	35	13

#### Enunciados

- 1.- Recuperar TODOS los datos de la tabla Productos (sin condición)
- 2.- Recuperar los datos de la fila donde el nombre del producto sea "Camiseta".
- 3.- Recuperar los datos de las filas donde Stock sea menor que 20
- 4.- Recuperar los datos de la fila con ProductID 3
- 5.- Eliminar las filas de productos donde el stock sea mayor o igual a 20

Formato de entrega:

```
CREATE DATABASE examen CHARACTER SET utf8;
```

```
USE examen;
```

```
CREATE TABLE producto (  
    idproducto INT NOT NULL AUTO_INCREMENT,  
    nombre VARCHAR(45),  
    descripcion TEXT,  
    precio INT,  
    stock INT,  
    PRIMARY KEY (idproducto));
```

```
INSERT INTO producto(nombre, descripcion, precio, stock)  
VALUES('Camiseta', 'Camiseta negra simple de talla única', 10,16),  
      ('Pantalón', 'Pantalón largo azul tipo chino', 20,24),  
      ('Gorra', 'Gorra azul con el logo de los Yankees', 15, 32),  
      ('Zapatillas', 'Zapatillas de running de color blanco y verde', 35, 13);
```

```
--1.- Recuperar TODOS los datos de la tabla Productos (sin condición)
```

```
SELECT * FROM producto;
```

```
-- Recuperar los datos de la fila donde el nombre del producto sea "Camiseta".
```

```
SELECT * FROM producto WHERE nombre='Camiseta';
```

```
-- Recuperar los datos de las filas donde Stock sea menor que 20
SELECT * FROM producto WHERE stock<20;

-- Recuperar los datos de la fila con ProductoID 3
SELECT * FROM producto WHERE idproducto = 3;

-- Eliminar las filas de productos donde el stock sea mayor o igual a 20
SELECT * FROM producto WHERE stock >=20;
```

## SCRIPT

A continuación dejo el script **COMPLETO** de mi trabajo. Esto incluye:

1. Generación de Schema.
2. TODOS los datos para ser cargados en la base de datos.
3. TODAS las consultas que están este documento.
4. TODAS las consultas que hice. Es decir, las del anterior punto junto a otras que decidí no agregar para no hacer muy largo este documento.

```
-----
-----
-----

--
--                               SCRIPT
--
-----
-----
-----

-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----
-- Schema movies
-- -----

-- -----
-- Schema movies
-- -----
```

```
CREATE SCHEMA IF NOT EXISTS `movies` DEFAULT CHARACTER SET utf8 ;
USE `movies` ;
```

```
-- Table `movies`.`Actor`
```

```
CREATE TABLE IF NOT EXISTS `movies`.`Actor` (
  `idActor` INT NOT NULL AUTO_INCREMENT,
  `nombre` VARCHAR(45) NULL,
  `apellido` VARCHAR(45) NULL,
  `fecha_nacimiento` DATE NULL,
  `genero` VARCHAR(1) NULL,
  `pais` VARCHAR(45) NULL,
  PRIMARY KEY (`idActor`))
ENGINE = InnoDB;
```

```
-- Table `movies`.`Director`
```

```
CREATE TABLE IF NOT EXISTS `movies`.`Director` (
  `idDirector` INT NOT NULL AUTO_INCREMENT,
  `nombre` VARCHAR(45) NULL,
  `apellido` VARCHAR(45) NULL,
  `fecha_nacimiento` DATE NULL,
  `genero` VARCHAR(45) NULL,
  `pais` VARCHAR(45) NULL,
  PRIMARY KEY (`idDirector`))
ENGINE = InnoDB;
```

```
-- Table `movies`.`Critico`
```

```
CREATE TABLE IF NOT EXISTS `movies`.`Critico` (
  `idCritico` INT NOT NULL AUTO_INCREMENT,
  `nombre` VARCHAR(45) NULL,
  PRIMARY KEY (`idCritico`))
ENGINE = InnoDB;
```

```
-- Table `movies`.`Pelicula`
```

```
CREATE TABLE IF NOT EXISTS `movies`.`Pelicula` (
  `idPelicula` INT NOT NULL AUTO_INCREMENT,
  `nombre` TEXT NULL,
  `año` INT NULL,
  `duracion` INT NULL,
  `presupuesto` INT NULL,
  `recaudacion` INT NULL,
  `idioma` TEXT NULL,
  `pais` TEXT NULL,
  PRIMARY KEY (`idPelicula`))
```

```
ENGINE = InnoDB;
```

```
-- Table `movies`.`Genero`
```

```
CREATE TABLE IF NOT EXISTS `movies`.`Genero` (  
  `idGenero` INT NOT NULL AUTO_INCREMENT,  
  `genero` VARCHAR(45) NULL,  
  PRIMARY KEY (`idGenero`))  
ENGINE = InnoDB;
```

```
-- Table `movies`.`Personaje`
```

```
CREATE TABLE IF NOT EXISTS `movies`.`Personaje` (  
  `idPelicula` INT NOT NULL,  
  `idActor` INT NOT NULL,  
  `nombre_personaje` TEXT NULL,  
  PRIMARY KEY (`idPelicula`, `idActor`),  
  INDEX `fk_Pelicula_has_Actor_Actor1_idx` (`idActor` ASC) VISIBLE,  
  INDEX `fk_Pelicula_has_Actor_Pelicula_idx` (`idPelicula` ASC) VISIBLE,  
  CONSTRAINT `fk_Pelicula_has_Actor_Pelicula`  
    FOREIGN KEY (`idPelicula`)  
    REFERENCES `movies`.`Pelicula` (`idPelicula`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_Pelicula_has_Actor_Actor1`  
    FOREIGN KEY (`idActor`)  
    REFERENCES `movies`.`Actor` (`idActor`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
-- Table `movies`.`Direccion`
```

```
CREATE TABLE IF NOT EXISTS `movies`.`Direccion` (  
  `idPelicula` INT NOT NULL,  
  `idDirector` INT NOT NULL,  
  PRIMARY KEY (`idPelicula`, `idDirector`),  
  INDEX `fk_Pelicula_has_Director_Director1_idx` (`idDirector` ASC) VISIBLE,  
  INDEX `fk_Pelicula_has_Director_Pelicula1_idx` (`idPelicula` ASC) VISIBLE,  
  CONSTRAINT `fk_Pelicula_has_Director_Pelicula1`  
    FOREIGN KEY (`idPelicula`)  
    REFERENCES `movies`.`Pelicula` (`idPelicula`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_Pelicula_has_Director_Director1`  
    FOREIGN KEY (`idDirector`)  
    REFERENCES `movies`.`Director` (`idDirector`)  
    ON DELETE NO ACTION
```

```
ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- Table `movies`.`Puntaje`
-----
```

```
CREATE TABLE IF NOT EXISTS `movies`.`Puntaje` (
  `idCritico` INT NOT NULL,
  `idPelicula` INT NOT NULL,
  `calificacion` FLOAT NULL,
  PRIMARY KEY (`idCritico`, `idPelicula`),
  INDEX `fk_Critico_has_Pelicula_Pelicula1_idx` (`idPelicula` ASC) VISIBLE,
  INDEX `fk_Critico_has_Pelicula_Critico1_idx` (`idCritico` ASC) VISIBLE,
  CONSTRAINT `fk_Critico_has_Pelicula_Critico1`
    FOREIGN KEY (`idCritico`)
      REFERENCES `movies`.`Critico` (`idCritico`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
  CONSTRAINT `fk_Critico_has_Pelicula_Pelicula1`
    FOREIGN KEY (`idPelicula`)
      REFERENCES `movies`.`Pelicula` (`idPelicula`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- Table `movies`.`Genero_Pelicula`
-----
```

```
CREATE TABLE IF NOT EXISTS `movies`.`Genero_Pelicula` (
  `idPelicula` INT NOT NULL,
  `idGenero` INT NOT NULL,
  PRIMARY KEY (`idPelicula`, `idGenero`),
  INDEX `fk_Pelicula_has_Genero_Genero1_idx` (`idGenero` ASC) VISIBLE,
  INDEX `fk_Pelicula_has_Genero_Pelicula1_idx` (`idPelicula` ASC) VISIBLE,
  CONSTRAINT `fk_Pelicula_has_Genero_Pelicula1`
    FOREIGN KEY (`idPelicula`)
      REFERENCES `movies`.`Pelicula` (`idPelicula`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
  CONSTRAINT `fk_Pelicula_has_Genero_Genero1`
    FOREIGN KEY (`idGenero`)
      REFERENCES `movies`.`Genero` (`idGenero`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```



--Actor/Actriz

```
INSERT INTO actor(nombre, apellido, fecha_nacimiento, genero, pais)
VALUES
```

```
('Ricardo', 'Darin', '1957-01-16', 'm', 'Argentina'),
('Gastón', 'Pauls', '1972-01-17', 'm', 'Argentina'),
('Guillermo', 'Francella', '1955-02-14', 'm', 'Argentina'),
('Keanu', 'Reves', '1964-09-02', 'm', 'USA'),
('Carrie-Anne', 'Moss', '1967-08-21', 'f', 'Canada'),
('Viggo', 'Mortensen', '1958-10-20', 'm', 'USA'),
('Jet', 'Li', '1963-04-26', 'm', 'China'),
('Guy', 'Pearce', '1967-10-05', 'm', 'UK'),
('Joe', 'Pantoliano', '1951-09-12', 'm', 'USA'),
('Leonardo', 'DiCaprio', '1974-11-11', 'm', 'USA'),
('Jason', 'Statham', '1966-07-26', 'm', 'UK'),
('Micael', 'Cane', '1933-03-14', 'm', 'UK'),
('Hugo', 'Weaving', '1960-04-04', 'm', 'UK'),
('Elijah', 'Wood', '1981-01-28', 'm', 'USA'),
('Brad', 'Pitt', '1963-12-17', 'm', 'USA'),
('Christian', 'Bale', '1974-01-30', 'm', 'UK'),
('Ziyi', 'Zhang', '1979-02-09', 'f', 'China'),
('Natalie', 'Portman', '1981-07-09', 'f', 'USA'),
('Cate', 'Blanchet', '1969-05-14', 'f', 'Australia'),
('Taika', 'Waititi', '1975-08-16', 'm', 'NZ'),
('Jemaine', 'Clement', '1974-01-10', 'm', 'NZ');
```

--Criticos

```
INSERT INTO critico(nombre)
VALUES
```

```
('IMDB'),
('Film Affinity'),
('Ignacio Cavallo');
```

--Calificación por película

```
INSERT INTO puntaje(idPelicula, idCritico, calificacion)
VALUES
```

```
(1, 1, 8.7), (1, 2, 7.9), (1, 3, 8.5),
(2, 1, 7.9), (2, 2, 7.3), (2, 3, 8.2),
(3, 1, 7.4), (3, 2, 7.9), (3, 3, 9.0),
(4, 1, 7.9), (4, 2, 7.5), (4, 3, 8.7),
(5, 1, 8.8), (5, 2, 8.0), (5, 3, 8.8),
(6, 1, 8.3), (6, 2, 7.9), (6, 3, 9.0),
(7, 1, 8.2), (7, 2, 7.8), (7, 3, 7.2),
(8, 1, 7.4), (8, 2, 6.3), (8, 3, 6.3),
(9, 1, 8.8), (9, 2, 8.0), (9, 3, 8.1),
(10, 1, 7.9), (10, 2, 7.8), (10, 3, 8.2),
(11, 1, 8.2), (11, 2, 8.1), (11, 3, 8.4),
(12, 1, 9.0), (12, 2, 8.1), (12, 3, 9.2),
(13, 1, 8.1), (13, 2, 7.5), (13, 3, 6.6),
(14, 1, 6.9), (14, 2, 7.0), (14, 3, 5.9),
(15, 1, 8.2), (15, 2, 7.6), (15, 3, 8.1),
(16, 1, 7.8), (16, 2, 7.2), (16, 3, 7.9),
(17, 1, 7.7), (17, 2, 6.8), (17, 3, 7.2);
```

--Genero

INSERT INTO genero(genero)

VALUES

('Acción'),  
( 'Animación'),  
( 'Ciencia Ficción'),  
( 'Comedia'),  
( 'Drama'),  
( 'Documental'),  
( 'Suspenso'),  
( 'Terror');

-- Genero de la Pelicula

INSERT INTO genero\_pelicula(idPelicula, idGenero)

VALUES

(1, 1), (1, 3),  
(2, 1), (2, 5),  
(3, 1), (3, 7),  
(4, 1), (4, 4),  
(5, 1), (5, 3),  
(6, 1), (6, 4),  
(7, 1), (7, 4),  
(8, 1), (8, 3),  
(9, 1), (9, 3),  
(10, 5), (10, 7),  
(11, 5), (11, 7),  
(12, 1), (12, 3),  
(13, 1), (13, 3),  
(14, 1), (14, 5),  
(15, 4), (15, 5),  
(16, 3), (16, 5),  
(17, 4), (17, 6);

-- Personajes

INSERT INTO personaje(idPelicula, idActor, nombre\_personaje)

VALUES

(1, 4, 'Neo'), (1, 5, 'Trinity'), (1, 9, 'Cypher'), (1, 13, 'Agente Smith'),  
(2, 7, 'Nameless'), (2, 17, 'Moon'),  
(3, 8, 'Leonard Shelby'), (3, 5, 'Natalie'),  
(3, 9, 'Teddy Gammel'),  
(4, 6, 'Ben'),  
(5, 10, 'Dominick Cobb'), (5, 12, 'Miles'),  
(6, 11, 'Turco'), (6, 15, 'Mickey'),  
(7, 11, 'Bacon'),  
(8, 4, 'John Wick'),  
(9, 14, 'Frodo Baggins'), (9, 6, 'Aragon'), (9, 19, 'Galadriel'), (9, 13, 'Elrond'),  
(10, 1, 'Marcos'), (10, 2, 'Juan'),  
(11, 1, 'Bejamín Espósito'), (11, 3, 'Pablo Sandoval'),  
(12, 16, 'Batman'), (12, 12, 'Alfred'),  
(13, 13, 'V'), (13, 18, 'Evey Hammond'),  
(14, 17, 'Xiao Mei'),



```
(15, 10, 'Jordan Belfort'),  
(16, 15, 'Benjamin Button'), (16, 19, 'Daisy Fuller'),  
(17, 20, 'Viago'), (17, 21, 'Vladislav');
```

```
-- Dirección | película dirigida por  
INSERT INTO direccion(idPelícula, idDirector)  
VALUES  
(1, 1), (1, 2),  
(2, 2),  
(3, 4),  
(4, 5),  
(5, 4),  
(6, 6),  
(7, 6),  
(8, 7),  
(9, 8),  
(10, 9),  
(11, 10),  
(12, 4),  
(13, 1),  
(14, 3),  
(15, 11),  
(16, 12),  
(17, 13),  
(17, 14);
```

```
-----  
-----  
-----
```

```
-- -----  
-- QUERY  
-- -----
```

```
-----  
-----  
-----
```

```
SELECT * FROM película;  
SELECT * FROM actor;  
SELECT * FROM director;
```

```
SELECT nombre FROM película  
WHERE (año = 2000 AND país = 'Argentina') OR (año >= 2003 AND país = 'NZ');
```

```
SELECT * FROM película  
WHERE película.idPelícula=9;  
UPDATE película  
SET nombre = 'El Señor de los Anillos'
```

```
WHERE pelicula.idPelicula=9;
```

```
SELECT * FROM pelicula  
WHERE pelicula.idPelicula=9;
```

```
SELECT nombre, pais FROM pelicula  
WHERE pais NOT IN ('USA', 'UK');
```

```
SELECT nombre, apellido FROM actor  
WHERE genero <> 'm';  
SELECT nombre, apellido FROM actor  
WHERE nombre LIKE 'j___%';
```

```
-- -----  
-- GROUPBY + ORDER BY  
-- -----
```

```
SELECT pais, COUNT(*) AS cantidad  
FROM pelicula  
GROUP BY pais;
```

```
SELECT pais, año, SUM(recaudacion) as Recaudación  
FROM pelicula  
GROUP BY año  
ORDER BY pais ;
```

```
SELECT nombre  
FROM pelicula  
WHERE recaudacion =  
(SELECT MAX(recaudacion) FROM pelicula);
```

```
SELECT nombre  
FROM pelicula  
WHERE recaudacion =  
(SELECT MIN(recaudacion) FROM pelicula);
```

```
SELECT nombre  
FROM pelicula  
WHERE recaudacion >  
(SELECT AVG(recaudacion) FROM pelicula);
```

```
SELECT  
    p.nombre  
FROM  
    pelicula p  
    INNER JOIN  
    genero_pelicula gp ON p.idPelicula = gp.idPelicula  
    inner JOIN  
    genero g ON g.idGenero = gp.idGenero  
  
WHERE g.genero = 'Acción';
```

```

SELECT
    pelicula.nombre,
    imdb.calificacion AS IMDB,
    ff.calificacion AS Film_Affinity,
    ic.calificacion AS Ignacio_Cavallo,
    ROUND((imdb.calificacion + ff.calificacion + ic.calificacion)/3.0,2) as
Promedio
FROM
    pelicula
        INNER JOIN
    puntaje imdb ON pelicula.idPelicula = imdb.idPelicula
        INNER JOIN
    puntaje ff ON pelicula.idPelicula = ff.idPelicula
        AND ff.idCritico = 2
        INNER JOIN
    puntaje ic ON pelicula.idPelicula = ic.idPelicula
        AND ic.idCritico = 3
GROUP BY pelicula.idPelicula;

```

```

SELECT actor.nombre, actor.apellido, personaje.nombre_personaje
FROM
    actor
        INNER JOIN
    personaje ON actor.idActor = personaje.idActor
        JOIN
    pelicula ON pelicula.idPelicula = personaje.idPelicula
WHERE pelicula.nombre = 'The Matrix';

```

```

SELECT
    director.nombre,
    director.apellido,
    COUNT(direccion.idDirector) AS cantidad_peliculas
FROM
    director
        INNER JOIN
    direccion ON director.idDirector = direccion.idDirector
GROUP BY director.nombre , director.apellido
HAVING cantidad_peliculas > 1;

```

```

CREATE VIEW Personajes_de_Pelicula AS
SELECT
    pelicula.nombre AS nombre_pelicula,
    actor.nombre,
    actor.apellido,
    personaje.nombre_personaje,
    pelicula.año -YEAR(actor.fecha_nacimiento) AS edad_personaje
FROM
    actor
        INNER JOIN
    personaje ON actor.idActor = personaje.idActor
        JOIN
    pelicula ON pelicula.idPelicula = personaje.idPelicula
ORDER BY pelicula.nombre;

```

```

CREATE VIEW directores_de_peliculas AS
SELECT
    director.nombre,
    director.apellido,
    pelicula.año - YEAR(director.fecha_nacimiento) as edad_director,
    pelicula.nombre as Título
FROM
    director
    INNER JOIN
    direccion ON director.idDirector = direccion.idDirector
    INNER JOIN
    pelicula ON pelicula.idPelicula =direccion.idPelicula;

SELECT * FROM personajes_de_pelicula
WHERE edad_personaje = (SELECT MIN(edad_personaje) FROM personajes_de_pelicula)
UNION
SELECT * FROM personajes_de_pelicula
WHERE edad_personaje = (SELECT MAX(edad_personaje) FROM personajes_de_pelicula);
-- -----
--          QUERY ADICIONALES
-- -----

SELECT nombre, año FROM pelicula WHERE pais = 'Argentina';

SELECT nombre FROM pelicula
WHERE año =2000 AND pais = 'Argentina';

SELECT nombre FROM pelicula
WHERE año=2000 OR año=2009;

SELECT nombre FROM pelicula
WHERE año>2010 and pais = 'nz';

SELECT nombre FROM pelicula
WHERE pais NOT IN ('USA', 'UK');

SELECT nombre FROM pelicula
WHERE pais IN ('UK', 'NZ');
SELECT nombre FROM pelicula
ORDER BY año;

SELECT nombre FROM pelicula
WHERE pais NOT IN('USA')
ORDER BY año DESC ;

SELECT pais, SUM(recaudacion) as ingresos
FROM pelicula
GROUP BY pais
ORDER BY ingresos DESC;

```

```

-- Ingresos por país por año, ordenado por país
SELECT pais, año, SUM(recaudacion) as Recaudación
FROM pelicula
GROUP BY año
ORDER BY pais;

SELECT nombre, (recaudacion - presupuesto) as ganancias
FROM pelicula;

SELECT
    g.genero, p.nombre
FROM
    pelicula p
    INNER JOIN
    genero_pelicula gp ON p.idPelicula = gp.idPelicula
    inner JOIN
    genero g ON g.idGenero = gp.idGenero

    ORDER BY g.genero;

-- Películas de acción
SELECT
    p.nombre
FROM
    pelicula p
    INNER JOIN
    genero_pelicula gp ON p.idPelicula = gp.idPelicula
    inner JOIN
    genero g ON g.idGenero = gp.idGenero

    WHERE g.genero = 'Acción';

-- Todas las películas y sus puntajes
SELECT pelicula.nombre, puntaje.calificacion
FROM
    pelicula
    INNER JOIN
    puntaje ON pelicula.idPelicula = puntaje.idPelicula;

SELECT pelicula.nombre, puntaje.calificacion, critico.idCritico
FROM
    pelicula
    INNER JOIN
    puntaje ON pelicula.idPelicula = puntaje.idPelicula
    JOIN
    critico ON critico.idCritico = puntaje.idCritico;

SELECT pelicula.nombre, imdb.calificacion as IMDB, ff.calificacion as
Film_Affinity, ic.calificacion as Ignacio_Cavallo
FROM pelicula
INNER JOIN
puntaje imdb ON pelicula.idPelicula = imdb.idPelicula
INNER JOIN
puntaje ff ON pelicula.idPelicula = ff.idPelicula and ff.idCritico=2

```

```

INNER JOIN
puntaje ic ON pelicula.idPelicula = ic.idPelicula AND ic.idCritico=3
GROUP BY pelicula.idPelicula;

-- Pelicula con su calificación por cada crítico
SELECT
    pelicula.nombre,
    imdb.calificacion AS IMDB,
    ff.calificacion AS Film_Affinity,
    ic.calificacion AS Ignacio_Cavallo,
    ROUND((imdb.calificacion + ff.calificacion + ic.calificacion)/3.0,2) as
Promedio
FROM
    pelicula
        INNER JOIN
    puntaje imdb ON pelicula.idPelicula = imdb.idPelicula
        INNER JOIN
    puntaje ff ON pelicula.idPelicula = ff.idPelicula
        AND ff.idCritico = 2
        INNER JOIN
    puntaje ic ON pelicula.idPelicula = ic.idPelicula
        AND ic.idCritico = 3
GROUP BY pelicula.idPelicula;

-- NO OLVIDAR CAMBIAR EL SEÑOR DE LOS ANILLOS

-- LOS ACTORES POR PELICULA
SELECT pelicula.nombre as nombre_pelicula, actor.nombre, actor.apellido
FROM
    actor
        INNER JOIN
    personaje ON actor.idActor = personaje.idActor
        JOIN
    pelicula ON pelicula.idPelicula = personaje.idPelicula;

-- LOS ACTORES en una PELICULA ESPECIFICA
SELECT actor.nombre, actor.apellido, personaje.nombre_personaje
FROM
    actor
        INNER JOIN
    personaje ON actor.idActor = personaje.idActor
        JOIN
    pelicula ON pelicula.idPelicula = personaje.idPelicula
    WHERE pelicula.nombre = 'El secreto de sus ojos';

-- Todos los directores por peliculas
SELECT pelicula.nombre as Título, director.nombre, director.apellido
FROM
    director
        INNER JOIN
    direccion ON director.idDirector = direccion.idDirector

```

```

        INNER JOIN
        pelicula ON pelicula.idPelicula =direccion.idPelicula;

-- Todos los directores con más de una peli

SELECT
    director.nombre,
    director.apellido,
    COUNT(direccion.idDirector) AS cantidad_peliculas
FROM
    director
        INNER JOIN
        direccion ON director.idDirector = direccion.idDirector
GROUP BY director.nombre , director.apellido
HAVING cantidad_peliculas > 1;

-- Todas las peliculas que son mayor al promedio segun mi puntuación
SELECT
    pelicula.nombre,
    ROUND(AVG(ic.calificacion), 2) as Promedio_Ignacio

FROM
    pelicula
        INNER JOIN
        puntaje imdb ON pelicula.idPelicula = imdb.idPelicula
        INNER JOIN
        puntaje ff ON pelicula.idPelicula = ff.idPelicula
        AND ff.idCritico = 2
        INNER JOIN
        puntaje ic ON pelicula.idPelicula = ic.idPelicula
        AND ic.idCritico = 3
GROUP BY pelicula.idPelicula
HAVING Promedio_Ignacio > AVG(Promedio_Ignacio);

--SELECT DATEDIFF ( yy , '1960-07-15' , '2019-10-22' )

SELECT *, DATEDIFF (CURDATE(), fecha_nacimiento) as edad
FROM actor;

SELECT nombre, YEAR(CURDATE()) - YEAR(fecha_nacimiento) AS age FROM ACTOR;

SELECT pelicula.nombre as nombre_pelicula, actor.nombre, actor.apellido,
personaje.nombre_personaje
FROM
    actor
        INNER JOIN
        personaje ON actor.idActor = personaje.idActor
        JOIN
        pelicula ON pelicula.idPelicula = personaje.idPelicula
        ORDER BY pelicula.nombre;

SELECT

```

```

    pelicula.nombre AS nombre_pelicula,
    actor.nombre,
    actor.apellido,
    personaje.nombre_personaje,
    pelicula.año -YEAR(actor.fecha_nacimiento) AS edad_personaje
FROM
    actor
        INNER JOIN
    personaje ON actor.idActor = personaje.idActor
        JOIN
    pelicula ON pelicula.idPelicula = personaje.idPelicula
ORDER BY pelicula.nombre;

```

```

SELECT * FROM personajes_de_pelicula
WHERE edad_personaje = (SELECT MAX(edad_personaje) FROM personajes_de_pelicula)
UNION
SELECT * FROM personajes_de_pelicula
WHERE edad_personaje = (SELECT MIN(edad_personaje) FROM personajes_de_pelicula);

```

---VIEWS

```

CREATE VIEW Personajes_de_Pelicula AS
SELECT
    pelicula.nombre AS nombre_pelicula,
    actor.nombre,
    actor.apellido,
    personaje.nombre_personaje,
    pelicula.año -YEAR(actor.fecha_nacimiento) AS edad_personaje
FROM
    actor
        INNER JOIN
    personaje ON actor.idActor = personaje.idActor
        JOIN
    pelicula ON pelicula.idPelicula = personaje.idPelicula
ORDER BY pelicula.nombre;

CREATE VIEW directores_de_peliculas AS
SELECT
    director.nombre,
    director.apellido,
    pelicula.año - YEAR(director.fecha_nacimiento) as edad_director,
    pelicula.nombre as Título
FROM
    director
        INNER JOIN
    direccion ON director.idDirector = direccion.idDirector
        INNER JOIN
    pelicula ON pelicula.idPelicula =direccion.idPelicula;

```