

PROYECTO DE APLICACIÓN WEB: CINE

Base de Datos - 1º DAW (M)



Razmik Sahakyan Drachev
GRADO SUPERIOR:DAW 1
IES A.L.I.X.A.R
30/03/25

ÍNDICE

1. INTRODUCCIÓN	3
2. MODELO ENTIDAD-RELACIÓN (ER)	3
3. MODELO RELACIONAL	4
4. CARGA MASIVA.....	3
5. CONSULTAS CREADAS.....	3
6. CREACIÓN DE VISTAS.....	3
7. FUNCIONES Y PROCEDIMIENTOS.....	3
8. TRIGGERS.....	2
9. FICHEROS DE GITHUB	1
10. IP AWS.....	1
11. VALORACIÓN PERSONAL.....	1

1. INTRODUCCIÓN

Descripción general del proyecto

Este proyecto tiene como objetivo crear una base de datos que se usará en el futuro para una aplicación web relacionada con el cine. Aunque no vamos a hacer la aplicación web en este proyecto vamos a centrarnos en diseñar una base de datos que tenga toda la información importante.

La base de datos se encargará de organizar todos estos datos para que cuando se desarrolle la aplicación web se pueda entrar de forma rápida y sencilla. Esto ayudará a gestionar las películas o las salas.

El proyecto incluirá el diseño de un diagrama de Entidad-Relación (MER), que luego se convertirá en un modelo relacional en una base de datos teniendo herramientas como MySQL Workbench y DBeaver para realizar las consultas que crearemos nosotros.

Objetivos del proyecto

Los principales objetivos de este proyecto son:

- ❖ Crear el Modelo Entidad-Relación (MER) que defina cómo se organizan los datos del cine.
- ❖ Resolver el diagrama MER extendido utilizando Diagrams, añadiendo nuevas entidades y relaciones según los requisitos.
- ❖ Convertir el MER en tablas relacionales utilizando MySQL Workbench, asegurándonos de que las relaciones entre las tablas estén correctas.
- ❖ Transformar el modelo relacional y realizar el paso a tablas.
- ❖ Cargar datos de prueba en las tablas:
 - Realizar la carga masiva de datos en al menos 2 tablas con un mínimo de 1000 registros por tabla, utilizando herramientas como Kaggle o Generate Data.
 - Cargar unos 500 registros por tabla en el resto de las tablas con aplicaciones como Mockaroo para simular un cine real.
- ❖ Generar 5 consultas SQL multitabla, agrupadas y/o con subconsultas que permitan obtener datos de la base de datos del cine.
- ❖ Crear 2 vistas a partir de alguna de las consultas del apartado anterior.
- ❖ Desarrollar funciones y procedimientos:
 - 2 funciones que realicen cálculos o transformaciones útiles para el sistema.
 - 3 procedimientos, donde uno de ellos haga uso de alguna de las funciones creadas.
- ❖ Implementar 2 triggers:
 - Definir los triggers y su propósito.
 - Mostrar un ejemplo práctico de cómo se activan al realizarlos.
- ❖ Generar un script SQL para crear la base de datos y ejecutar este script en DBeaver.
- ❖ Valoración personal:
 - Reflexión sobre el trabajo realizado, el aprendizaje adquirido y posibles mejoras futuras para el proyecto.

2. MODELO ENTIDAD-RELACIÓN

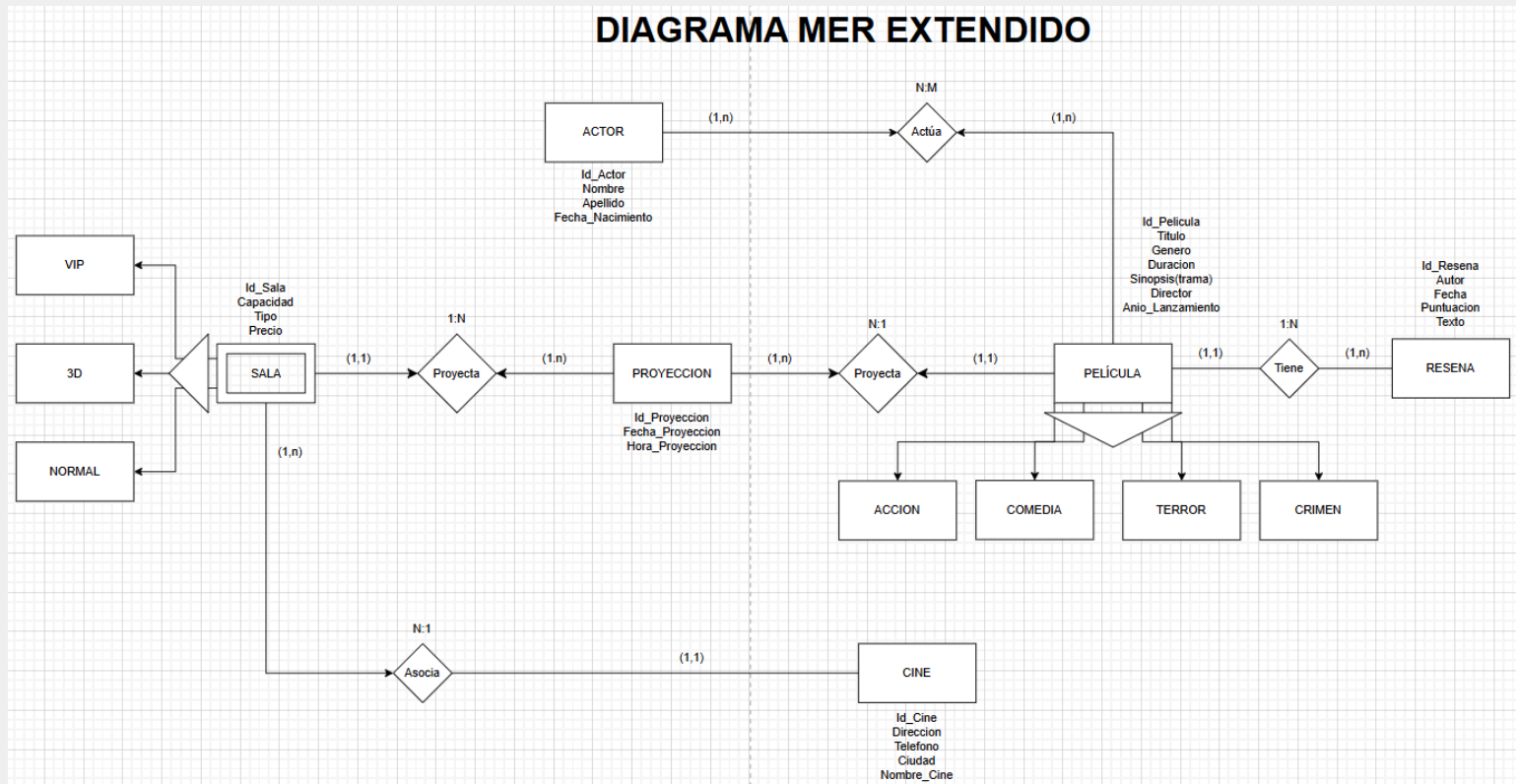
Para seguir un orden primero vamos a definir todas las entidades que encontramos en este proyecto que gira en torno al cine:

ENTIDADES

- ❖ **Cine:** Es el lugar donde se proyectan las películas para que el público las vea.
- ❖ **Película:** Es el contenido principal que se muestra en los cines. Podemos encontrar un tipo “enum” donde las películas pueden ser las siguientes:
 - Acción.
 - Comedia.
 - Terror.
 - Crimen.
- ❖ **Sala (débil):** Es el espacio dentro del cine donde se proyectan las películas. Es una entidad débil ya que no puede ser identificada de manera única por sus propios atributos, sino que depende de otra entidad para su identificación.. Además para la entidad sala tenemos un tipo “enum” que puede ser el siguiente:
 - Normal: Es la sala de cine más común. Todo el mundo puede entrar y es la opción más barata.
 - VIP: Esta sala es especial y más cara. Puede haber menos gente.
 - 3D: En esta sala las películas se ven con gafas especiales que hacen que las cosas parezcan reales.
- ❖ **Reseña:** Es una opinión que tienen los espectadores sobre la película que han visto, puede ser un texto o una valoración del 1 al 10.
- ❖ **Actor:** Es la persona que interpreta el papel de los personajes en la película.
- ❖ **Proyección:** Es el evento en el que se exhibe una película en una sala de cine en un horario específico para el público.

Vamos a seguir sobre como he relacionado cada una de las entidades del modelo entidad-relación:

- Entre actor y película tenemos una relación N:M donde se nos generaría una nueva tabla, debido a que un actor puede interpretar muchas películas y en una película puede haber muchos actores.
- Entre sala y proyección nos encontramos una relación 1:N, ya que en una sala pueden haber varias proyecciones y una proyección está solamente en esa sala.
- Entre película y proyección nos encontramos una relación N:1, porque una película puede proyectarse muchas veces y una proyección tiene una sola película.
- Entre película y reseña tenemos una relación 1:N, porque una sola película puede tener muchos comentarios y un comentario solo está en esa película.
- Entre las entidades sala y cine hay una relación N:1, esto es debido a que una sala se asocia a un solo cine y un cine tiene muchas salas.



3. MODELO RELACIONAL

Ahora vamos a pasar al modelo relacional siguiendo la lógica que hemos realizado con el modelo entidad-relación:

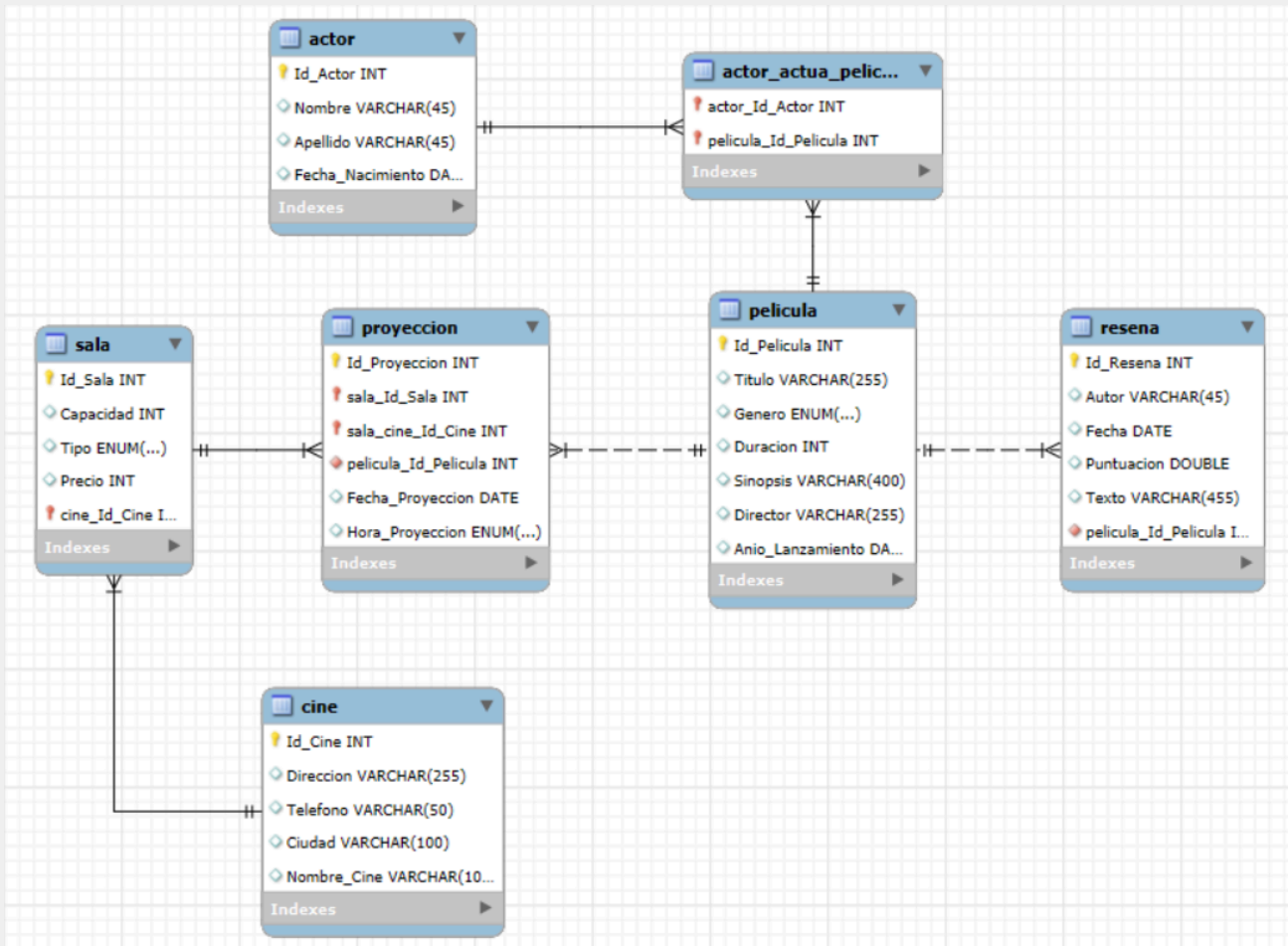
Entre las entidades actor y película como hemos dicho antes generan una tabla donde se guardan en una nueva tabla pk y fk de cada entidad (Id_Actor, Id_Pelicula). La tabla se llama "actor_actua_pelicula".

Entre las entidades sala y proyección hay una relación 1:N, por lo tanto la entidad proyección recoge el id cine de sala y el id de la sala.

Entre las entidades proyección y película hay una relación N:1, por lo tanto la entidad proyección recoge el id cine de película como fk.

Entre las entidades película y reseña hay una relación 1:N, por lo tanto la entidad reseña coge la pk de película (Id_Pelicula) como fk.

Entre las entidades sala y cine existe una relación N:1 por lo tanto la entidad sala recoge la pk de la entidad cine (Id_Cine) como fk.



4. CARGA MASIVA

Luego de hacer el modelo relacional pasamos al punto de la carga masiva de datos. Para este paso primero hay que crear las tablas dentro de esa base de datos, donde se almacenarán los datos específicos, como nombres o fechas. Cuando las tablas están listas, ya podemos realizar la carga masiva de datos, en este caso he utilizado la aplicación Mockaroo pero también se puede hacer con la aplicación de Kaggle. Usamos esta aplicación para generar datos falsos o meter información desde archivos CSV (que son los que guardan información en forma de lista, con datos separados por comas.

). Después de todo esto tenemos todo preparado para probar y trabajar con la base de datos.

Lo primero que tenemos que hacer es seleccionar la opción de “file” que se encuentra arriba a la izquierda, luego seleccionamos la opción de exportar y finalmente nos saldrán diferentes opciones pues clicamos para crear el script SQL. Si realizamos todos esos pasos nos llevará a la opción de “Forward Engineer SQL Script”:

Forward Engineer SQL Script

SQL Export Options

Filter Objects

Review SQL Script

Output SQL Script File: ...

Leave blank to view generated script but not save to a file.

SQL Options

- ☐ Generate DROP Statements Before Each CREATE Statement
- ☐ Generate DROP SCHEMA
- ☐ Sort Tables Alphabetically ⓘ
- ☐ Skip Creation of FOREIGN KEYS
- ☐ Skip creation of FK Indexes as well
- ☐ Omit Schema Qualifier in Object Names
- ☐ Generate USE statements
- ☐ Generate Separate CREATE INDEX Statements
- ☐ Add SHOW WARNINGS After Every DDL Statement
- ☐ Do Not Create Users. Only Export Privileges
- ☐ Don't create view placeholder tables.
- ☐ Generate INSERT Statements for Tables
- ☐ Disable FK checks for inserts
- ☐ Create triggers after inserts

Back Next Cancel

En el segundo apartado no cambiamos nada y en la parte final nos encontramos el apartado para generar el Script de nuestra base de datos del cine:

Forward Engineer SQL Script

Review Generated Script

Review and edit the generated script and press Finish to save.

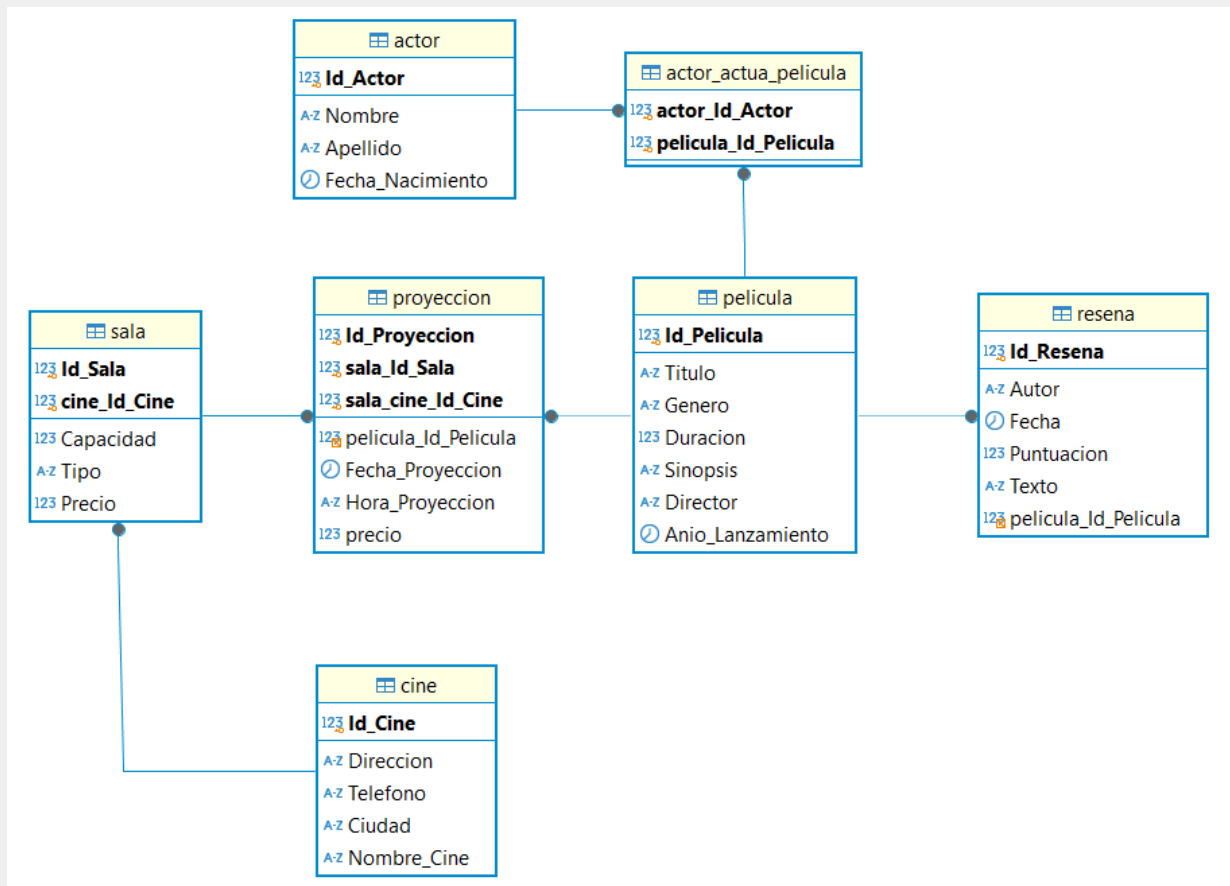
```
1  -- MySQL Script generated by MySQL Workbench
2  -- Mon Mar 31 10:54:07 2025
3  -- Model: New Model   Version: 1.0
4  -- MySQL Workbench Forward Engineering
5
6  SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
7  SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CH
8  SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT
9
10
11  -- Schema mydb
12  --
13
14  -- Schema mydb
15  --
16
17  CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
18  USE `mydb` ;
19
20
```

Save to Other File... Copy to Clipboard

Back Finish Cancel

Luego entramos en la aplicación de dbeaver, aquí copiamos y pegamos el script y lo ejecutamos en Dbeaver.

Luego de ejecutar el script en el dbeaver ya nos aparecen las entidades relacionadas con sus atributos, en este caso el nombre del proyecto por defecto se llama "mydb":



Un paso muy importante es generar datos aleatorios a esas entidades.

En este caso he usado la aplicación de mockaroo ([Mockaroo - Random Data Generator and API Mocking Tool | JSON / CSV / SQL / Excel](#)), al entrar nos aparecerán datos de una tabla de una entidad aleatoria

La interfaz de Mockaroo muestra la configuración para generar datos aleatorios. Se han seleccionado los siguientes campos:

Field Name	Type	Options
id	Row Number	blank: 0 %
first_name	First Name	blank: 0 %
last_name	Last Name	blank: 0 %
email	Email Address	blank: 0 %
gender	Gender	blank: 0 %
ip_address	IP Address v4	blank: 0 %

Se han configurado los siguientes parámetros de generación:

- # Rows: 1000
- Format: CSV
- Line Ending: Unix (LF)
- Include: ☒ header ☐ BOM

Los botones de acción son: GENERATE DATA, PREVIEW, SAVE AS..., DERIVE FROM EXAMPLE..., y MORE.

En el apartado de rows escribimos 500 (datos) o 1000 en dos tablas obligatoriamente , en el formato elegimos la opción de MySQL y en el apartado de “Line Ending” escribimos en nombre de la entidad exactamente como está escrito en el dbeaver.

Al escribir los datos en tabla de Mockaroo luego nos aparece el apartado de “Type” que básicamente es lo que queremos que sea el atributo, por ejemplo de título elegimos “Movie Title” o de año de lanzamiento “Datetime” con una fecha inicial y una fecha final:

Luego simplemente pulsaremos el botón de “Generate Data” para que genere datos aleatorios. Ejemplo de la entidad reseña:

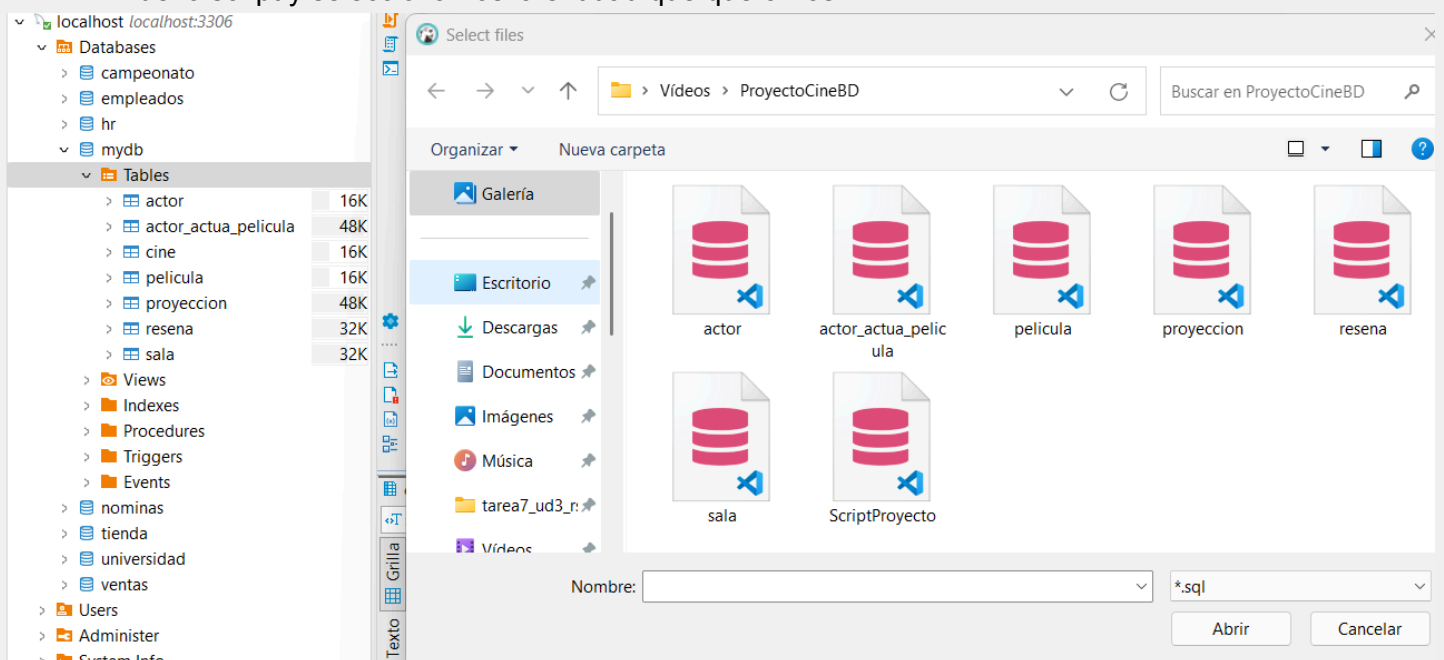
The screenshot shows the Mockaroo web application interface. The top navigation bar includes links for SCHEMAS, DATASETS, APIS, DATABASES, SCENARIOS, PROJECTS, and FUNCTIONS. The main area displays a table configuration for a table named 'pelicula'. The table has the following fields:

Field Name	Type	Options
Id_Pelicula	Row Number	blank: 0 %
Titulo	Movie Title	blank: 0 %
Genero	Custom List	Acción, Comedia, Terror, Crimen
Duracion	Number	min: 60 max: 180 decimals: 0 blank: 0 %
Puntuacion	Number	min: 1 max: 10 decimals: 1 blank: 0 %
Sinopsis	Sentences	at least 1 but no more than 1 blank: 0 %
Director	Full Name	blank: 0 %
Anio_Lanzamiento	Datetime	01/01/2000 to 01/01/2020 format: yyyy-mm-dd blank: 0 %

At the bottom, there are buttons for '+ ADD ANOTHER FIELD' and 'GENERATE FIELDS USING AI...'. Below the table configuration, there are fields for '# Rows: 500', 'Format: SQL', 'Table Name: pelicula', and a checkbox for 'include CREATE TABLE'. At the bottom right, there are buttons for 'GENERATE DATA', 'PREVIEW', 'SAVE AS...', 'DERIVE FROM EXAMPLE...', and 'MORE'.

Las descargas de tabla de cada entidad las he guardado en una carpeta que luego importamos en dbeaver.

Luego desde el dbeaver seleccionamos la opción de “Editor SQL” e importamos en un nuevo script y seleccionamos la entidad que queramos:



Un ejemplo de script de la entidad película:

```
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (1, 'Moving Out', 'Comedia', 134, 8.9, 'Maecenas ut massa quis augue luctus tincidunt.', 'Amandi Skuse', '2012-04-03');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (2, 'Devils, The', 'Terror', 150, 4.5, 'Morbi vestibulum, velit id pretium iaculis, diam erat fermentum justo, nec', 'Bo Bottle', '2010-04-13');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (3, 'Under the Skin', 'Crimen', 118, 9.0, 'In quis justo.', 'Ronica O''Crowley', '2014-01-12');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (4, 'Babyfever', 'Acción', 76, 2.7, 'Nam congue, risus semper porta volutpat, quam pede lobortis ligula, sit amet', 'Corabelle Dallander', '2003-07-13');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (5, 'Of Love and Shadows', 'Acción', 103, 1.4, 'Nam dui.', 'Marissa Saltmarshe', '2011-01-28');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (6, 'Tiny Furniture', 'Terror', 113, 8.8, 'In tempor, turpis nec euismod scelerisque, quam turpis adipiscing lorem', 'Denis Yeatman', '2018-11-16');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (7, 'Silent Night, Deadly Night', 'Acción', 86, 6.0, 'Integer ac neque.', 'Stephen Ray', '2012-03-01');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (8, 'How to Marry a Millionaire', 'Crimen', 139, 8.3, 'Vivamus vestibulum sagittis sapien.', 'Feliza Goodbourn', '2006-11-25');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (9, 'Makioka Sisters, The (Sasame-yuki)', 'Crimen', 62, 2.1, 'Cum sociis natoque penatibus et magnis dis parturient', 'Luci Wernrdley', '2009-02-03');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (10, 'Narrow Margin', 'Acción', 80, 7.4, 'Donec quis orci eget orci vehicula condimentum.', 'Saloma Koppel', '2015-08-13');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (11, 'Last Days of Mussolini (Mussolini: Ultimo atto)', 'Acción', 84, 6.3, 'Aenean auctor gravida sem.', 'Amy Lamyman', '2005-09-19');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (12, 'Vera Drake', 'Acción', 177, 7.9, 'Fusce lacus purus, aliquet at, feugiat non, pretium quis, lectus.', 'Lief Razzell', '2010-04-13');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (13, 'Drive Hard', 'Comedia', 122, 9.7, 'Nulla neque libero, convallis eget, eleifend luctus, ultricies eu, nibh.', 'Karmen Lesley', '2007-01-21');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (14, 'Novocaine', 'Comedia', 64, 2.8, 'Mauris enim leo, rhoncus sed, vestibulum sit amet, cursus id, turpis.', 'Benedetto Matusевич', '2016-10-25');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (15, 'Sniper', 'Acción', 152, 9.8, 'Integer tincidunt ante vel ipsum.', 'Blaine Lescop', '2018-03-10');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (16, 'Crew, The', 'Crimen', 82, 2.2, 'Phasellus in felis.', 'Viv Howles', '2017-06-21');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (17, 'Samurai I: Musashi Miyamoto (Miyamoto Musashi)', 'Acción', 111, 1.6, 'Duis at velit eu est congue elementum.', 'Bernardina Seefus', '2007-01-26');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (18, 'Immortals', 'Comedia', 61, 5.5, 'In congue.', 'Jeanne Mankor', '2002-03-11');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (19, 'Family, The (Famiglia, La)', 'Crimen', 84, 8.4, 'Vivamus metus arcu, adipiscing molestie, hendrerit at, vulputate', 'Lia Jelf', '2019-05-14');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (20, 'Hella W', 'Acción', 91, 6.0, 'Donec posuere metus vitae ipsum.', 'Arlyne Curthoys', '2007-08-21');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (21, 'Huhwihaji Anha (No Regret)', 'Comedia', 115, 2.0, 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.', 'Marta Symones', '2002-09-28');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (22, 'Core, The', 'Comedia', 152, 9.5, 'Nam tristique tortor eu pede.', 'Chan Dearden', '2019-11-12');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (23, 'Cropsey', 'Acción', 127, 7.2, 'Duis mattis egestas metus.', 'Jeffry Hollow', '2011-09-25');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (24, 'Ryan', 'Acción', 151, 3.2, 'Duis at velit eu est congue elementum.', 'Roda Ovid', '2008-05-21');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (25, 'Test Pilot', 'Acción', 91, 5.8, 'Nullam orci pede, venenatis non, sodales sed, tincidunt eu, felis.', 'Obed Scirman', '2006-01-20');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (26, 'Kenny', 'Acción', 68, 7.9, 'Nulla suscipit ligula in lacus.', 'Clarence Hendrikse', '2007-05-02');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (27, 'Golden Gate', 'Terror', 113, 6.6, 'Morbi ut odio.', 'Chris Luffman', '2004-02-12');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (28, 'Dark Passage', 'Terror', 175, 8.3, 'Praesent blandit lacinia erat.', 'Karlis Tissell', '2016-03-03');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (29, 'Charlie: The Life and Art of Charles Chaplin', 'Comedia', 132, 8.9, 'Morbi a ipsum.', 'Mario Mayfield', '2013-06-25');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (30, 'Halloween Tree, The', 'Terror', 147, 8.5, 'Proin at turpis a pede posuere nonummy.', 'Vivien Benthams', '2017-06-11');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (31, 'Species II', 'Terror', 137, 8.2, 'Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus', 'Karlis Tissell', '2016-03-03');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (32, 'Flipped', 'Comedia', 125, 1.6, 'Integer ac neque.', 'Fredri Willeman', '2016-04-26');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (33, 'Phil Ochs: There But for Fortune', 'Terror', 162, 7.6, 'Donec quis orci eget orci vehicula condimentum.', 'Arlyne Curthoys', '2007-08-21');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (34, 'Lola', 'Comedia', 105, 5.6, 'Suspendisse potenti.', 'Britt Gladwell', '2002-08-14');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (35, 'Single Girl, A (Fille seule, La)', 'Terror', 95, 5.1, 'Morbi non lectus.', 'Burke Dearan', '2016-04-23');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (36, 'Zero Effect', 'Acción', 75, 4.7, 'Mauris enim leo, rhoncus sed, vestibulum sit amet, cursus id, turpis.', 'Bernardina Seefus', '2007-01-26');
INSERT INTO Pelicula (id, titulo, genero, duracion, sinopsis, director, lanzamiento) VALUES (37, 'Hot to Trot', 'Comedia', 94, 7.5, 'Nunc purus.', 'Gerladina Petru', '2013-01-24');
```

Vemos que genera 500 datos aleatorios de cada uno de sus atributos.

Clicando la tabla de la entidad película y seleccionando la opción de datos , vemos con más facilidad los datos que se han generado:

123 Id_Pelicula	Az Titulo	Az Genero	123 Duracion	Az Sinopsis	Az Director	Anio_Lanzamiento
1	1 Moving Out	Comedia	134	Maecenas ut massa quis au	Amandi Skuse	2012-04-03
2	2 Devils, The	Terror	150	Morbi vestibulum, velit id p	Bo Bottle	2010-04-13
3	3 Under the Skin	Crimen	118	In quis justo.	Ronica O'Crowley	2014-01-12
4	4 Babyfever	Accion	76	Nam congue, risus semper	Corabelle Dallander	2003-07-13
5	5 Of Love and Shadows	Accion	103	Nam dui.	Marissa Saltmarshe	2011-01-28
6	6 Tiny Furniture	Terror	113	In tempor, turpis nec euism	Denis Yeatman	2018-11-16
7	7 Silent Night, Deadly Night	Accion	86	Integer ac neque.	Stephen Ray	2012-03-01
8	8 How to Marry a Millionaire	Crimen	139	Vivamus vestibulum sagittis	Feliza Goodbourn	2006-11-25
9	9 Makioka Sisters, The (Sasame	Crimen	62	Cum sociis natoque penatib	Luci Wernrdley	2009-02-03
10	10 Narrow Margin	Accion	80	Donec quis orci eget orci ve	Saloma Koppel	2015-08-13
11	11 Last Days of Mussolini (Mus	Accion	84	Aenean auctor gravida sem	Amy Lamyman	2005-09-19
12	12 Vera Drake	Accion	177	Fusce lacus purus, aliquet at	Lief Razzell	2010-04-13
13	13 Drive Hard	Comedia	122	Nulla neque libero, convalli	Karmen Lesley	2007-01-21
14	14 Novocaine	Comedia	64	Mauris enim leo, rhoncus se	Benedetto Matusевич	2016-10-25
15	15 Sniper	Accion	152	Integer tincidunt ante vel ip	Blaine Lescop	2018-03-10
16	16 Crew, The	Crimen	82	Phasellus in felis.	Viv Howles	2017-06-21
17	17 Samurai I: Musashi Miyamo	Accion	111	Duis at velit eu est congue	Bernardina Seefus	2007-01-26
18	18 Immortals	Comedia	61	In congue.	Jeanne Mankor	2002-03-11
19	19 Family, The (Famiglia, La)	Crimen	84	Vivamus metus arcu, adipis	Lia Jelf	2019-05-14
20	20 Hella W	Accion	91	Donec posuere metus vitae	Arlyne Curthoys	2007-08-21
21	21 Huhwihaji Anha (No Regret)	Comedia	115	Lorem ipsum dolor sit amet	Marta Symones	2002-09-28
22	22 Core, The	Comedia	152	Nam tristique tortor eu ped	Chan Dearden	2019-11-12
23	23 Cropsey	Accion	127	Duis mattis egestas metus.	Jeffry Hollow	2011-09-25
24	24 Ryan	Accion	151	Duis at velit eu est congue	Roda Ovid	2008-05-21
25	25 Test Pilot	Accion	91	Nullam orci pede, venenati	Obed Scirman	2006-01-20
26	26 Kenny	Accion	68	Nulla suscipit ligula in lacus	Clarence Hendrikse	2007-05-02
27	27 Golden Gate	Terror	113	Morbi ut odio.	Chris Luffman	2004-02-12
28	28 Dark Passage	Terror	175	Praesent blandit lacinia era	Karlis Tissell	2016-03-03
29	29 Charlie: The Life and Art of	Comedia	132	Morbi a ipsum.	Mario Mayfield	2013-06-25
30	30 Halloween Tree, The	Terror	147	Proin at turpis a pede posu	Vivien Benthams	2017-06-11

5. CONSULTAS CREADAS

1. Mostrar los actores con su nombre, apellido y la cantidad de películas en las que han participado. Si un actor no ha participado en ninguna película, deberá aparecer el mensaje 'NO TIENE PELÍCULAS'. Los resultados estarán ordenados de mayor a menor número de películas:

```
select a.Id_Actor as IdActor, a.Nombre as Nombre_Actor, a.Apellido as ApellidoActor,
if (count(aap.pelicula_id_pelicula)=0, 'NO HA ACTUADO', count(aap.pelicula_id_pelicula)) as NumPelículas
from actor a left join actor_actua_pelicula aap
on a.Id_Actor = aap.actor_Id_Actor
group by a.Id_Actor, a.Nombre, a.Apellido
order by NumPelículas desc;
```

RESULTADO:

	123 IdActor	A-z Nombre_Actor	A-z ApellidoActor	A-z NumPelículas ↓
1	768	Dasie	Dye	NO HA ACTUADO
2	766	Pavia	Stritton	NO HA ACTUADO
3	957	Livvy	MacCracken	NO HA ACTUADO
4	950	Eberto	Franschini	NO HA ACTUADO
5	722	Layla	Skillern	NO HA ACTUADO
6	912	Graeme	Dignam	NO HA ACTUADO
7	905	Shandee	Denisevich	NO HA ACTUADO
8	642	Chrotoem	Dellow	NO HA ACTUADO
9	819	Addison	Whereat	NO HA ACTUADO
10	3	Bentley	Adshede	8
11	264	Pete	Pauleau	7
12	32	Elvira	Cobbald	7
13	67	Clemmie	Darkott	5
14	259	Suzie	Fury	5
15	507	Meghan	Sackett	5
16	915	Clemence	Chazette	5
17	183	Andris	Tuddall	5
18	815	Shani	Goby	5
19	769	Jsandye	Fihelly	5
20	569	Eldin	Escritt	5
21	35	Albina	Crockley	5
22	112	Lucas	Arbon	4
23	891	Shandra	Sallenger	4
24	615	Moirá	Josephoff	4
25	245	James	Jeaves	4
26	452	Pancho	Brackenridge	4
27	110	Virgie	Rollings	4
28	906	Burtie	Don	4
29	193	Abran	di Rocca	4
30	396	Carmon	Castells	4
31	720	Morris	Mcshull	4

2. Mostrar la sala con la mayor capacidad por cine, mostrando el nombre del cine, el id de la sala, la capacidad de la sala, el tipo de sala y el id del cine. Además el id del cine tiene que estar ordenado ascendentemente.

```
select c.nombre_cine, s.id_sala, s.capacidad, s.tipo, c.Id_Cine
from cine c inner join sala s
on c.id_cine = s.cine_id_cine
inner join proyeccion p
on s.id_sala = p.sala_id_sala
where s.capacidad = (
select max(s2.capacidad)
from sala s2
where s2.cine_id_cine = c.id_cine
)
group by c.id_cine, c.nombre_cine, s.id_sala, s.capacidad, s.tipo, c.Id_Cine
order by c.Id_Cine asc;
```

RESULTADO:

	A-Z nombre_cine	id_sala	capacidad	tipo	Id_Cine
1	Realfire	6	197	VIP	1
2	Ntags	2	199	3D	2
3	Myworks	2	198	3D	3
4	Trunyx	1	175	VIP	4
5	Livefish	4	197	VIP	5
6	Skalith	2	155	VIP	6
7	Mudo	7	200	3D	7
8	Meetz	7	199	VIP	8
9	Zoonoodle	2	169	Normal	9
10	Yabox	2	128	VIP	10
11	Buzzster	1	200	3D	11
12	Tavu	1	183	3D	12
13	Jabbersphere	4	194	VIP	13
14	Kimia	7	194	3D	14
15	Yotz	8	192	Normal	15
16	Ainyx	2	131	3D	16
17	Ainyx	4	131	VIP	16
18	Oba	7	199	Normal	17
19	Voonyx	5	199	Normal	18
20	Browsetype	2	195	VIP	19
21	Oyoloo	10	189	VIP	20
22	Flashspan	7	178	VIP	21
23	LiveZ	7	179	Normal	22
24	Voolith	3	133	VIP	23
25	Skyndu	9	186	3D	24
26	Kwinu	6	173	VIP	25

3. Obtener la lista de cines con su nombre, ciudad y el precio promedio de sus entradas a la sala, ordenados de mayor a menor por su precio promedio:

```
select c.id_cine, c.nombre_cine, c.ciudad, avg(s.precio) as precio_promedio
from cine c inner join sala s
on c.id_cine = s.cine_id_cine
group by c.id_cine, c.nombre_cine, c.ciudad
order by precio_promedio desc;
```

RESULTADO:

	123 id_cine	A-Z nombre_cine	A-Z ciudad	123 precio_promedio
1	23	Voolith	As Sab' Biyār	50
2	10	Yabox	Damnoen Saduak	35
3	16	Ainyx	Rancabugel	30
4	6	Skalith	Santa Cruz	23,3333
5	25	Kwinu	Lebak Timur	22,1429
6	18	Voonyx	Tall Tamr	17,9
7	2	Ntags	Solna	16,8
8	22	LiveZ	Habo	16,4444
9	3	Myworks	Halmstad	15,8571
10	9	Zoonoodle	Tlogotunggal	15,4286
11	19	Browsetype	Artimet	15,1429
12	8	Meetz	Sabi	14,2
13	20	Oyoloo	Khirbat Tīn Nūr	14,1667
14	11	Buzzster	Jasionów	14,0909
15	4	Trunyx	Kushk	14
16	1	Realfire	Socorro	13,8667
17	14	Kimia	Miaotou	13
18	12	Tavu	Kashima	12
19	13	Jabbersphere	Xiangyun	11,3333
20	15	Yotz	Shuikou	10,875
21	17	Oba	Rennes	10,5
22	5	Livfish	Muyaka	9
23	7	Mudo	Beni	8,5833
24	24	Skyndu	Jinglou	8,4
25	21	Flashspan	Ljubovija	8

4. Obtener los apellidos y nombre de los actores (en una única columna y separados por espacios) que han participado en menos películas que el promedio de películas realizadas por los actores, ordenados alfabéticamente por apellido:

```

select concat(a.apellido, ' ', a.nombre) as nombre_completo, count(ap.pelicula_Id_Pelicula) as peliculas
from actor a
inner join actor_actua_pelicula ap on a.id_actor = ap.actor_id_actor
group by a.id_actor, a.apellido, a.nombre
having count(ap.pelicula_id_pelicula) < (
    select avg(peliculas_por_actor)
    from (
        select count(*) as peliculas_por_actor
        from actor_actua_pelicula
        group by actor_id_actor
    ) as peliculas_por_actor
)
order by a.apellido;

```

RESULTADO:

	A-Z nombre_completo	123 películas
1	Abeles Bethanne	1
2	Adamovitch Windham	1
3	Addinall Llywellyn	1
4	Addlestone Myron	1
5	Aiken Harrie	1
6	Aitkin Pierson	1
7	Akehurst Sonni	1
8	Albisser Bellina	1
9	Albrighton Gerek	1
10	Allcroft Berkie	1
11	Alliband Blanche	1
12	Alloway Audie	1
13	Ambresin Gregorio	1
14	Annot Antoine	1
15	Antham Shaylynn	1
16	Antonio Etty	1
17	Appleby Vaughan	1
18	Arnett Brennen	1
19	Arnson Stefanie	1
20	Ashtonhurst Murielle	1
21	Aspinal Nichol	1
22	Asple Shandee	1
23	Asple Lonee	1
24	Asplin Eveline	1
25	Ather Adolpho	1
26	Atthow Josie	1
27	Autin Keir	1
28	Avramov Torrence	1
29	Beal Lennette	1

5. Obtener el actor más mayor junto con su fecha de nacimiento y la cantidad de películas en las que ha participado. Sin usar limit:

```
select a.nombre, a.apellido, a.fecha_nacimiento, count(ap.pelicula_id_pelicula) as total_peliculas
from actor a left join actor_actua_pelicula ap
on a.id_actor = ap.actor_id_actor
group by a.id_actor, a.nombre, a.apellido, a.fecha_nacimiento
having a.fecha_nacimiento <= all (
    select a2.fecha_nacimiento
    from actor a2
)
order by total_peliculas asc;
```

RESULTADO:

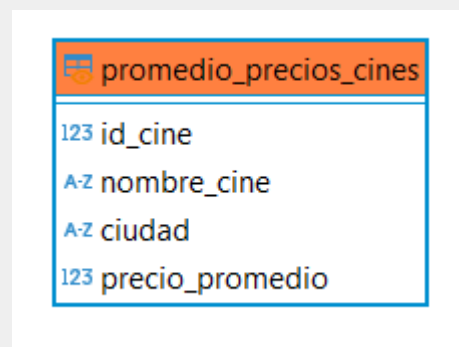
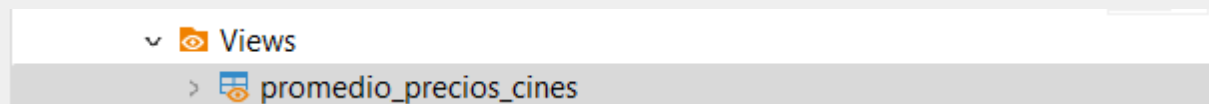
	A-Z nombre	A-Z apellido	🕒 fecha_nacimiento	123 total_peliculas
1	Godard	Learmount	1940-02-01	1

6. CREACIÓN DE VISTAS

1. Vista para mostrar promedio de las entradas a la sala, ordenados de mayor a menor por su precio promedio de cada cine, incluyendo el id del cine, el nombre, la ciudad y el precio promedio de mayor a menor:

```
create view promedio_precios_cines as
select c.id_cine, c.nombre_cine, c.ciudad, avg(s.precio) as precio_promedio
from cine c inner join sala s
on c.id_cine = s.cine_id_cine
group by c.id_cine, c.nombre_cine, c.ciudad
order by precio_promedio desc;
```

RESULTADO:

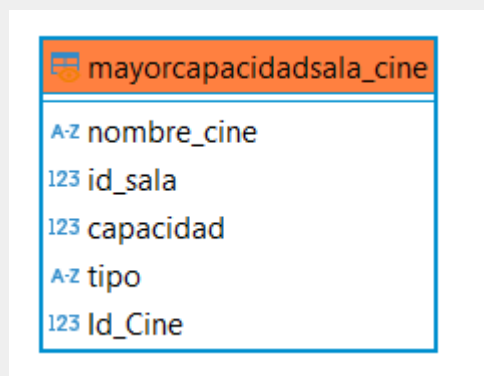
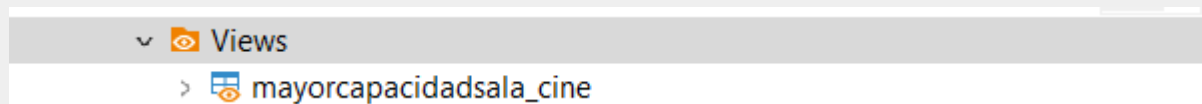


	123 id_cine	A-Z nombre_cine	A-Z ciudad	123 precio_promedio
1	23	Voolith	As Sab' Biyār	50
2	10	Yabox	Damnoen Saduak	35
3	16	Ainyx	Rancabugel	30
4	6	Skalith	Santa Cruz	23,3333
5	25	Kwinu	Lebak Timur	22,1429
6	18	Voonyx	Tall Tamr	17,9
7	2	Ntags	Solna	16,8
8	22	LiveZ	Habo	16,4444
9	3	Myworks	Halmstad	15,8571
10	9	Zoonoodle	Tlogotunggal	15,4286
11	19	Browsetype	Artimet	15,1429
12	8	Meetz	Sabi	14,2
13	20	Oyoloo	Khirbat Tīn Nūr	14,1667
14	11	Buzzster	Jasionów	14,0909
15	4	Trunyx	Kushk	14
16	1	Realfire	Socorro	13,8667
17	14	Kimia	Miaotou	13
18	12	Tavu	Kashima	12
19	13	Jabbersphere	Xiangyun	11,3333
20	15	Yotz	Shuikou	10,875
21	17	Oba	Rennes	10,5
22	5	Livefish	Muyaka	9
23	7	Mudo	Beni	8,5833
24	24	Skyndu	Jinglou	8,4
25	21	Flashspan	Ljubovija	8

2. Vista para mostrar la sala con la mayor capacidad por cine, mostrando el nombre del cine, el id de la sala, la capacidad de la sala, el tipo de sala y el id del cine. Además el id del cine tiene que estar ordenado ascendentemente.

```
create view mayorCapacidadSala_cine as
select c.nombre_cine, s.id_sala, s.capacidad, s.tipo, c.Id_Cine
from cine c inner join sala s
on c.id_cine = s.cine_id_cine
    inner join proyeccion p
    on s.id_sala = p.sala_id_sala
where s.capacidad = (
    select max(s2.capacidad)
    from sala s2
    where s2.cine_id_cine = c.id_cine
)
group by c.id_cine, c.nombre_cine, s.id_sala, s.capacidad, s.tipo, c.Id_Cine
order by c.Id_Cine asc;
```

RESULTADO:



	A-Z nombre_cine	123 id_sala	123 capacidad	A-Z tipo	123 Id_Cine
1	Realfire	6	197	VIP	1
2	Ntags	2	199	3D	2
3	Myworks	2	198	3D	3
4	Trunyx	1	175	VIP	4
5	Livefish	4	197	VIP	5
6	Skalith	2	155	VIP	6
7	Mudo	7	200	3D	7
8	Meetz	7	199	VIP	8
9	Zoonoodle	2	169	Normal	9
10	Yabox	2	128	VIP	10
11	Buzzster	1	200	3D	11
12	Tavu	1	183	3D	12
13	Jabbersphere	4	194	VIP	13
14	Kimia	7	194	3D	14
15	Yotz	8	192	Normal	15
16	Ainyx	2	131	3D	16
17	Ainyx	4	131	VIP	16
18	Oba	7	199	Normal	17
19	Voonyx	5	199	Normal	18
20	Browsetype	2	195	VIP	19
21	Oyoloo	10	189	VIP	20
22	Flashspan	7	178	VIP	21
23	LiveZ	7	179	Normal	22
24	Voolith	3	133	VIP	23
25	Skyndu	9	186	3D	24
26	Kwinu	6	173	VIP	25

7.1. FUNCIONES

1. Función que muestre cuántas veces se ha proyectado una película. Si no existe debe mostrar el numero -1. El nombre de la función se llamará "contarproyeccionesporpelicula".

```
delimiter &&
create function contarproyeccionesporpelicula(p_id_pelicula int)
returns int
deterministic
begin
    declare conteo int;
    -- verificar si la película existe
    if not exists (select 1 from pelicula where id_pelicula = p_id_pelicula) then
        return -1; -- Muestra -1 si la película no existe
    end if;

    -- contar el número de proyecciones
    set conteo = (select count(*)
                  from proyeccion
                  inner join pelicula on proyeccion.pelicula_id_pelicula = pelicula.id_pelicula
                  where pelicula.id_pelicula = p_id_pelicula);

    return ifnull(conteo, 0);
end &&
delimiter ;
```

Comprobamos que se ha creado la función en nuestra base de datos con ese nombre de función:

```
show function status where db = 'projectocine' and name = 'contarproyeccionesporpelicula';
```

Db	Name	Type	Definer	Modified	Created	Security_type	Comment	character_set_client	collation_connection	Database
projectocine	contarproyeccionesporpelicula	FUNCTION	root@localhost	2025-03-31 14:17:43	2025-03-31 14:17:43	DEFINER		utf8mb4	utf8mb4_0900_ai_ci	utf8mb3_gen

Resultado de la película 342 por ejemplo:

```
select contarproyeccionesporpelicula(342) as proyecciones;
```

proyecciones
4

2. Función que muestra cuántas reseñas con una puntuación de 7 o más ha recibido una película en sus proyecciones de las 15:00. Si la película no existe, devuelve -1. El nombre de la función se llamaría “resenasPositivasPorPeliculaHora”.

```

delimiter &&
create function resenaspositivasporpeliculahora(p_id_pelicula int)
returns int
deterministic
begin
    declare total int;
    -- Verificar si la película existe
    if not exists (select 1 from pelicula where id_pelicula = p_id_pelicula) then
        return -1; -- Muestra -1 si la película no existe
    end if;
    -- Contar reseñas positivas de las proyecciones a las 15:00
    set total = (select count(r.id_resena)
        from resena r
        inner join pelicula p on r.pelicula_id_pelicula = p.id_pelicula
        inner join proyeccion pr on p.id_pelicula = pr.pelicula_id_pelicula
        where r.puntuacion >= 7
        and p.id_pelicula = p_id_pelicula
        and pr.hora_proyeccion = '15:00');
    return ifnull(total, 0);
end &&
delimiter ;

```

Comprobamos que se ha creado la función en nuestra base de datos con ese nombre de función:

```
show function status where db = 'proyectocine' and name = 'resenaspositivasporpeliculahora';
```

n(+) 1 x

how function status where db = 'proyectocine' and

	Az Db	Az Name	Az Type	Az Definer	Modified	Created	Az Security_type	Az Comment	Az character_set_client	Az collation_
1	proyectocine	resenaspositivasporpeliculahora	FUNCTION	root@localhost	2025-03-31 17:17:04	2025-03-31 17:17:04	DEFINER		utf8mb4	utf8mb4_090

Por ejemplo la película 241 tiene dos reseñas positivas cuando se proyecta a las 15:00:

```
select resenaspositivasporpeliculahora(241) as resenas positivas;
```

ct resenaspositivas| *Enter a SQL expression to filter results (use Ctrl+Space)*

123 resenas_positivas

2

7.2. PROCEDIMIENTOS

1. Procedimiento que muestre las películas por cine. Si el cine es menor de 1 o mayor que 25 debería mostrara u mensaje de error. El nombre del procedimiento se llamaría “muestra_peliculas_por_cine”.

```
delimiter &&
create procedure muestra_peliculas_por_cine(in p_id_cine int)
begin
    declare existe_cine int default 0;
    declare cantidad_peliculas int default 0;
    declare max_cines int default 25; -- Límite máximo de cines

    -- Verificar si el ID es negativo o mayor que el límite máximo
    if p_id_cine <= 0 or p_id_cine > max_cines then
        select CONCAT('ID de cine inválido. Debe estar entre 1 y ', max_cines) as Mensaje;
    else
        -- Verificar si el cine existe
        select count(*) into existe_cine from cine where id_cine = p_id_cine;
        if existe_cine > 0 then
            -- Contar la cantidad de películas en ese cine
            select count(distinct p.id_pelicula) into cantidad_peliculas
            from pelicula p
            inner join proyeccion pr on p.id_pelicula = pr.pelicula_id_pelicula
            inner join sala s on pr.sala_id_sala = s.id_sala and pr.sala_cine_id_cine = s.cine_id_cine
            where s.cine_id_cine = p_id_cine;

            if cantidad_peliculas > 0 then
                select distinct p.titulo as Pelicula
                from pelicula p
                inner join proyeccion pr on p.id_pelicula = pr.pelicula_id_pelicula
                inner join sala s on pr.sala_id_sala = s.id_sala and pr.sala_cine_id_cine = s.cine_id_cine
                where s.cine_id_cine = p_id_cine;
            else
                select 'Sin películas' as Mensaje;
            end if;
        end if;
    end if;
end &&
delimiter ;
```

Ejemplo de cine menor que 1 y mayor que 25

call muestra_peliculas_por_cine(-3);			
Resultados 1	Estadísticas 1		
call muestra_peliculas_por_cine(-3) Enter a SQL expression to filter			
<div> <div>A-Z Mensaje</div> <table> <tr> <td>1</td><td>ID de cine inválido. Debe estar entre 1 y 25</td></tr> </table> </div>		1	ID de cine inválido. Debe estar entre 1 y 25
1	ID de cine inválido. Debe estar entre 1 y 25		

Ejemplo de cine dentro del rango

call muestra_peliculas_por_cine(2);																	
pelicula 1	Estadísticas 1																
call muestra_peliculas_por_cine(2) Enter a SQL expression to filter																	
<div> <div>A-Z Pelicula</div> <table> <tr><td>1</td><td>Billy's Hollywood</td></tr> <tr><td>2</td><td>Emitai</td></tr> <tr><td>3</td><td>Jim Jefferies: BAL</td></tr> <tr><td>4</td><td>Ambushers, The</td></tr> <tr><td>5</td><td>Man of a Thousa</td></tr> <tr><td>6</td><td>Laurel Canyon</td></tr> <tr><td>7</td><td>Tobruk</td></tr> <tr><td>8</td><td>Kansas Raiders</td></tr> </table> </div>		1	Billy's Hollywood	2	Emitai	3	Jim Jefferies: BAL	4	Ambushers, The	5	Man of a Thousa	6	Laurel Canyon	7	Tobruk	8	Kansas Raiders
1	Billy's Hollywood																
2	Emitai																
3	Jim Jefferies: BAL																
4	Ambushers, The																
5	Man of a Thousa																
6	Laurel Canyon																
7	Tobruk																
8	Kansas Raiders																

2. Procedimiento que elimine a un actor. Si no existe el actor debe mostrar un mensaje de error. Si el actor es borrado exitosamente también debe mostrar un mensaje. El nombre del procedimiento se llamaría “eliminar_actor”.

```
delimiter &&
create procedure eliminar_actor(in p_id_actor int)
begin
    declare existe_actor int default 0;
    declare tiene_peliculas int default 0;

    -- verificar si el actor existe
    select count(*) into existe_actor from actor where id_actor = p_id_actor;

    if existe_actor = 0 then
        select 'el actor no existe' as mensaje;
    else
        -- verificar si el actor está asociado a alguna película
        select count(*) into tiene_peliculas from actor_actua_pelicula where actor_id_actor = p_id_actor;

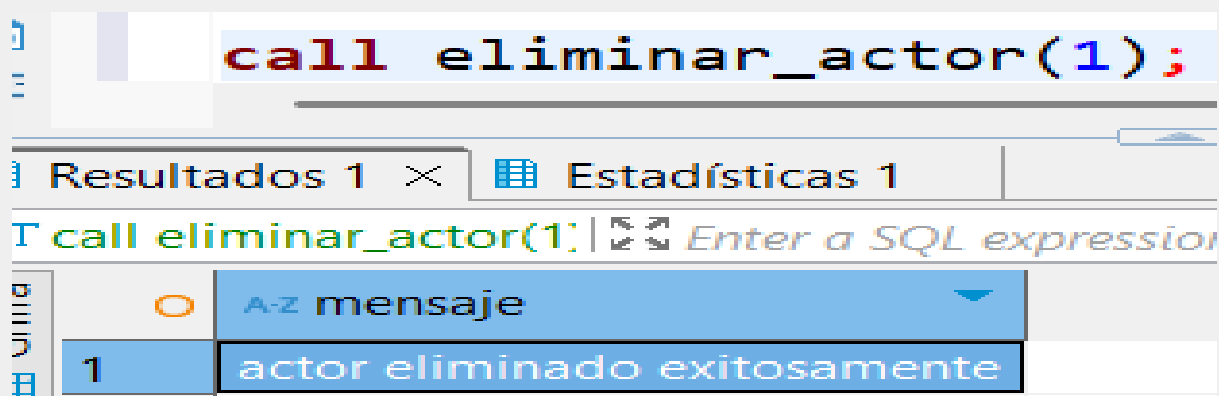
        if tiene_peliculas > 0 then
            -- eliminar relaciones del actor con películas antes de eliminarlo
            delete from actor_actua_pelicula where actor_id_actor = p_id_actor;
        end if;

        -- eliminar el actor
        delete from actor where id_actor = p_id_actor;

        -- confirmar eliminación verificando si el actor sigue existiendo
        select count(*) into existe_actor from actor where id_actor = p_id_actor;

        if existe_actor = 0 then
            select 'actor eliminado exitosamente' as mensaje;
        else
            select 'error al eliminar el actor' as mensaje;
        end if;
    end if;
end &&
delimiter ;
```

Ejemplo de actor 1 eliminado correctamente



The screenshot shows a SQL IDE interface. At the top, the command 'call eliminar_actor(1);' is entered. Below, the 'Results' pane shows a single row with the message 'actor eliminado exitosamente'.

Id	A-Z mensaje
1	actor eliminado exitosamente

Como vemos el actor 1 ha sido eliminado

actor				
Enter a SQL expression to filter results (use Ctrl+Space)				
	123 Id_Actor	A-Z Nombre	A-Z Apellido	Fecha_Nacimiento
1	2	Thomasin	Blandamere	1950-04-03
2	3	Bentley	Adshede	1979-05-17
3	4	Dynah	Spurrett	1968-01-18
4	5	Jessa	Gentery	1986-01-21
5	6	Richmond	Pinks	1950-06-18
6	7	Petronilla	Mion	1994-01-10
7	8	Gretel	Fyrth	1984-01-04
8	9	Tann	Causton	1963-02-20
9	10	Lanie	Loyley	1994-10-05
10	11	Torey	Hylden	1971-01-04
11	12	Sim	Wakeman	1947-06-20

3. Procedimiento de películas que ha hecho un actor por géneros. Si no existe el actor debe mostrar un mensaje de error. El nombre del procedimiento se llamará "contar peliculas por actor genero".

```
delimiter &&
create procedure contar_peliculas_por_actor_genero(in p_id_actor int)
begin
    declare existe_actor int default 0;

    -- Verificar si el actor existe
    select count(*) into existe_actor from actor where id_actor = p_id_actor;

    if existe_actor = 0 then
        select 'El actor no existe' as Mensaje;
    else
        -- Contar la cantidad de películas en las que ha actuado el actor por género
        select p.genero, count(distinct p.id_pelicula) as total_peliculas
        from pelicula p
        inner join actor_actua_pelicula ap on p.id_pelicula = ap.pelicula_id_pelicula
        where ap.actor_id_actor = p_id_actor
        group by p.genero;
    end if;
end &&
delimiter ;
```

Ejemplo de actor 26 (no ha participado en el género de acción)

call contar_peliculas_por_actor_genero(23);

pelicula 1	Estadísticas 1	Salida
call contar_peliculas_ Enter a SQL expression to filter results		
	A-Z genero	123 total_peliculas
1	Comedia	1
2	Terror	1
3	Crimen	1

Ejemplo de actor que no existe

call contar_peliculas_por_actor_genero(2300);

Resultados 1	Estadísticas 1	Salida
call contar_peliculas_ Enter a SQL expression to filter results		
	A-Z Mensaje	
1	El actor no existe	

8. TRIGGERS

1. Trigger que verifica que la duración de la película sea al menos de 60 minutos. El nombre del trigger se llamaría “verificar_duracion_pelicula”.

```
delimiter &&

create trigger verificar_duracion_pelicula
before insert on pelicula
for each row
begin
    if new.duracion < 60 then
        signal sqlstate '45000'
        set message_text = 'no se puede insertar la película porque no tiene más de 60 minutos.';
    end if;
end &&

delimiter ;
```

Probamos a insertar una película de menos de 60 minutos:

```
-- Intentar insertar una película con duración menor a 60 minutos
INSERT INTO pelicula (Id Pelicula, Titulo, Genero, Duracion, Sinopsis, Director, Año Lanzamiento)
VALUES (501, 'Película corta', 'Comedia', 50, 'Una comedia corta', 'John Doe', '2025-03-29');
```

adísticas 1 × Salida

SQL Error [1644] [45000]: no se puede insertar la película porque no tiene más de 60 minutos.

```
-- Intentar insertar una película con
INSERT INTO pelicula (Id Pelicula, Ti
```

2. Trigger que verifica que el actor tenga menos de 100 años. El nombre del trigger se llamaría “verificar_edad_actor”.

```
delimiter &&

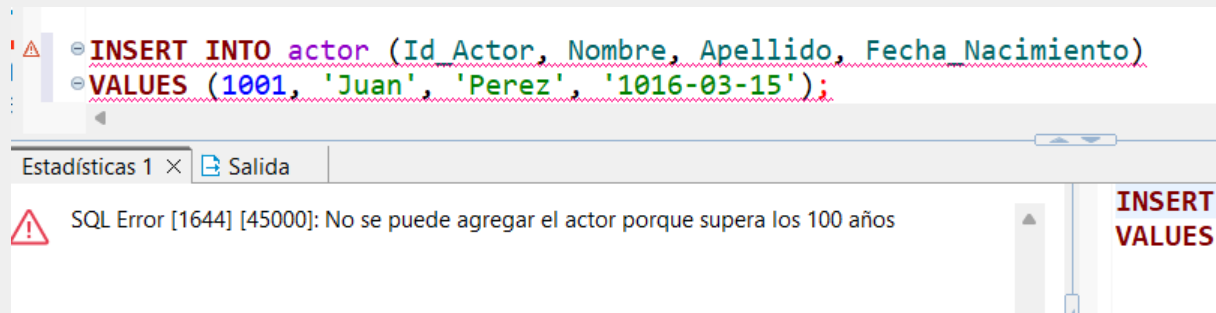
create trigger verificar_edad_actor
before insert on actor
for each row
begin
    declare edad INT;

    -- Calcular la edad del actor con base en su fecha de nacimiento
    SET edad = TIMESTAMPDIFF(YEAR, new.Fecha_Nacimiento, CURDATE());

    -- Verificar si la edad es mayor a 100 años
    if edad > 100 then
        signal sqlstate '45000'
        set message_text = 'No se puede agregar el actor porque supera los 100 años';
    end if;
end &&

delimiter ;
```

Probamos a insertar un actor de mas de 100 años:



9. FICHEROS GITHUB

Para el tema de GitHub he subido cuatro ficheros: un archivo SQL con el esquema de la base de datos (miproyecto-schema.sql), otro con los datos de la base (miproyecto-data.sql), uno más con las consultas, vistas, funciones, procedimientos y triggers (miproyecto-queries.sql) y un PDF con la documentación.

Link a mi perfil personalizado de GitHub:

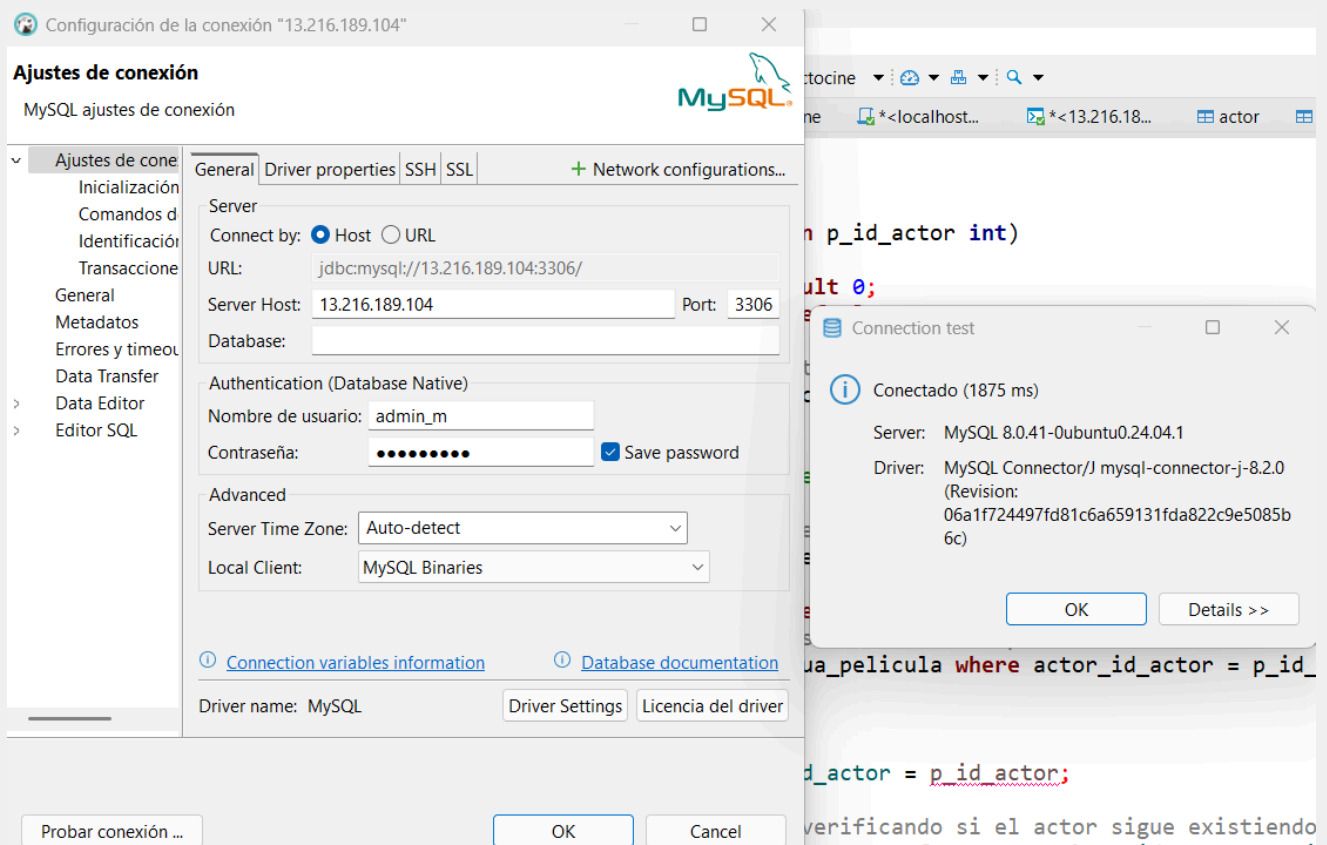
[RazmikSahakyan/RazmikBDficheros](https://github.com/RazmikSahakyan/RazmikBDficheros)

10. IP de AWS

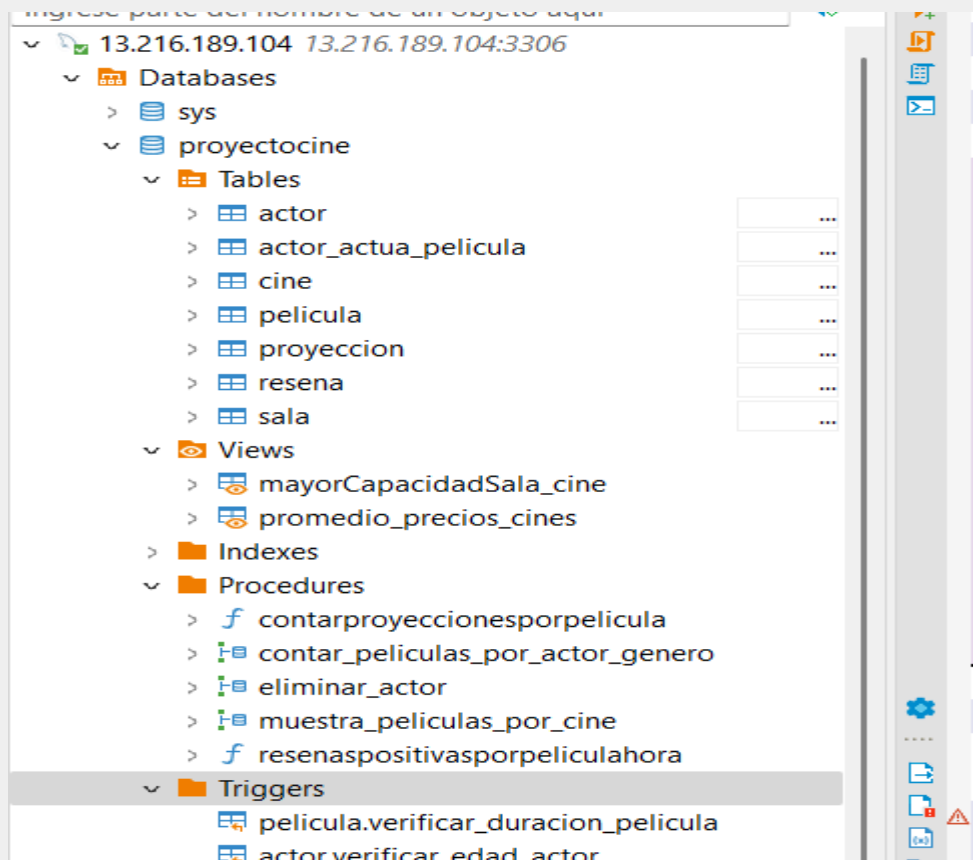
Debemos crear una instancia de EC2 en AWS, instalar el mysql-server y cargar nuestro proyecto en dicha instancia. Debemos poder realizar la presentación de tu proyecto desde el DBeaver.

Número de IP de la base de datos: 13.216.189.104

Probamos la conexión y vemos que funciona:



Como vemos esta la base de datos "proyectocine" con todas sus entidades, procedimientos, funciones y triggers:



11. VALORACIÓN PERSONAL

Este proyecto me ha ayudado a aprender más sobre cómo diseñar y crear bases de datos, especialmente en un cine donde hay varias entidades como películas, salas y reseñas que se relacionan entre ellas. Con el modelo Entidad-Relación y el paso de convertirlo en un modelo relacional he mejorado a como organizar y relacionar las tablas con claves (pk,fk). Además, al trabajar con la carga de datos y realizar consultas SQL, también he aprendido a llevar grandes cantidades de datos y a obtener respuestas de manera rápida. También he mejorado al usar subconsultas para hacer preguntas más complicadas sobre la relación entre películas, salas y reseñas, entre otras entidades.

Al crear consultas SQL, funciones, procedimientos y triggers, he aprendido a optimizar la base de datos para asegurar que los datos sean correctos y mejorar su funcionamiento. También he trabajado con herramientas como Diagrams, MySQL y AWS, lo que me ha permitido acercarme más al uso de bases de datos en el mundo real.

He mejorado mis habilidades con SQL, que es muy importante para poder crear y administrar bases de datos de mejor manera. Con la carga masiva de generar datos aleatorios para las entidades con la aplicación morackoo me ha costado al principio manejar los datos, pero poco a poco he conseguido mejorar con el uso de estos datos .

En general, este proyecto ha sido un reto interesante que me ha permitido mejorar mis habilidades en bases de datos. En el futuro, me gustaría añadir más funciones, como un sistema de reservas en línea.

