UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

Informe de Proyecto Integrador de Saberes
Fundamentos de Base de Datos

Autores:

- Valdivieso Paucar Byron Marcelo

Octubre 2022 – Febrero 2023

Tabla de Contenidos

Extracción, transformación, carga y explotación de datos integrando Base de Datos

1. Introducción

En el proyecto actual de la asignatura Fundamentos de Base de Datos, se busca aplicar los conocimientos adquiridos a lo largo del curso con el fin de trabajar con un archivo CSV llamado "moviesdataset". Este archivo fue descargado de un repositorio de GitHub y deberá ser leído, modelado, limpiado y explotado a través del lenguaje de consulta SQL y el sistema de gestión de bases de datos relacional MySQL.

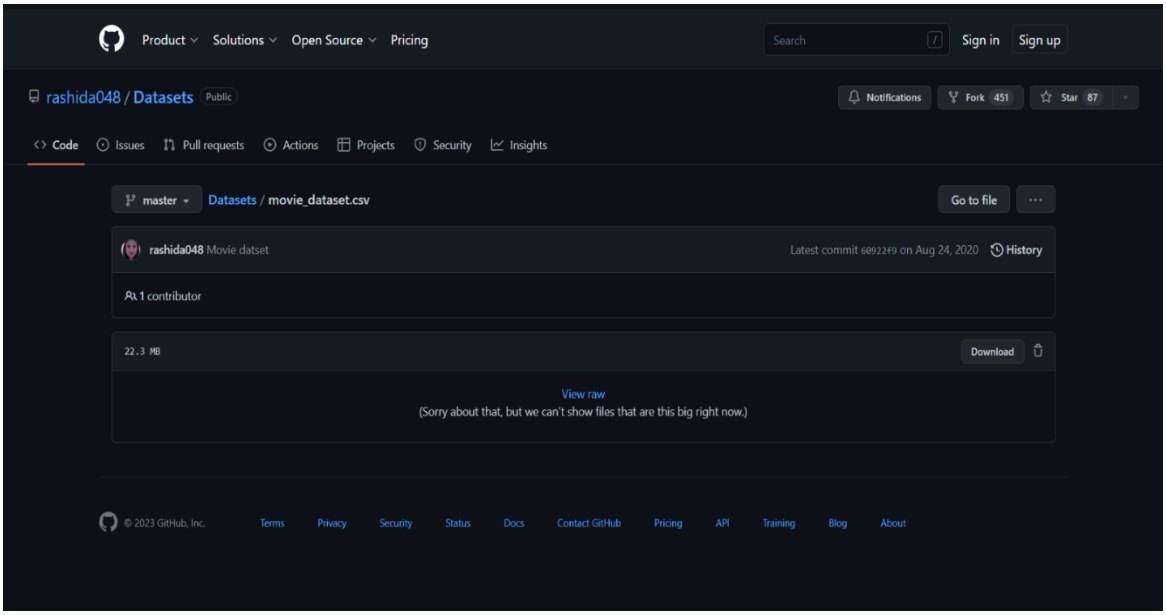
2. Desarrollo

2.1 Según el Orden según la ejecución

2.1.1Importación del archivo CSV en Excel para su análisis

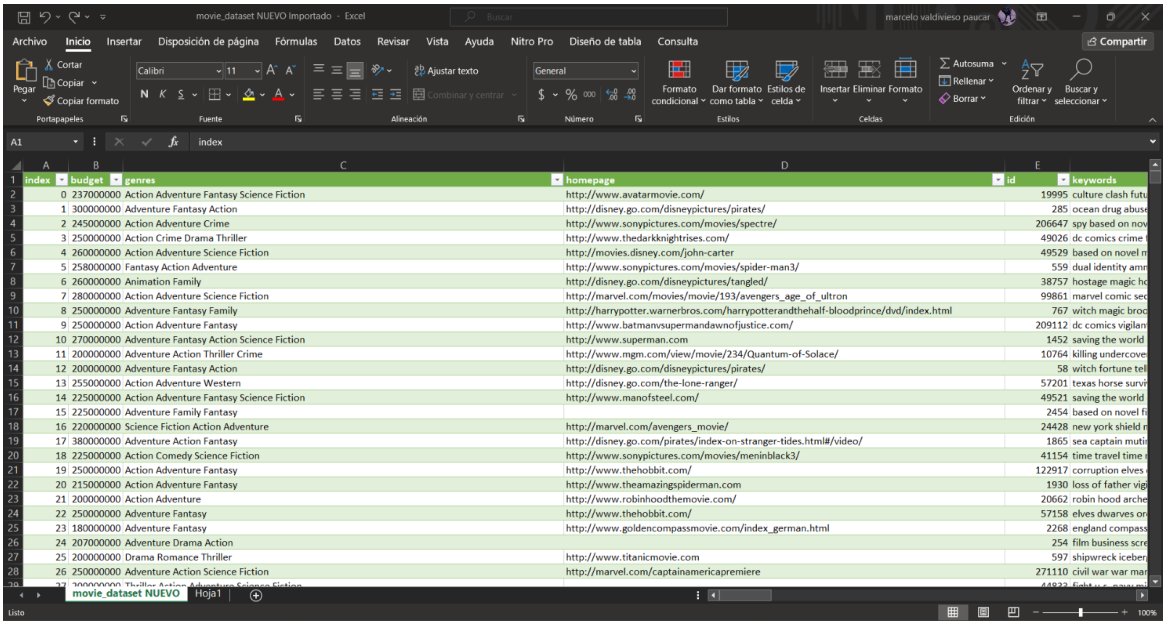
Lo primero que hicimos fue dirigirnos a la página del GitHub para descargar el archivo *csv*:

Ilustración 1.
Github de donde se obtiene el dataset



Una vez descargado los importamos en Excel para visualizar los datos de mejor manera:

Ilustración 2. Visualización de datos en Excel



A continuación, el análisis de cada columna:

Tabla 1.
Descripción de Metadatos

Nombre Columna	Descripción
budget	Sin inconvenientes, pertenece a la tabla movies, no puede ser clave primaria, contiene el presupuesto de las películas.
genres	Es una columna compuesta, pertenece a la tabla genres, posible clave primaria, contiene los géneros de las películas.
homepage	Sin inconvenientes, pertenece a la tabla movies, no puede ser clave primaria, contiene las páginas web de las películas.
id	Sin inconvenientes, pertenece a la tabla movies, posible clave primaria, contiene el identificador de cada película.
keywords	Es una columna compuesta, pertenece a la tabla keywords, posible clave primaria, contiene las palabras clave de las peliculas.
original_language	Sin inconvenientes, pertenece a la tabla movies, no puede ser clave primaria, contiene el idioma original de las películas.
original_title	Ay que hacer remplazo de caracteres de otros idiomas, pertenece a la tabla movies, no puede ser clave primaria, contiene el titulo original de las películas.
overwiev	Sin inconvenientes, pertenece a la tabla movies, no puede ser clave primaria, contiene la visión general de cada película.
popularity	Sin inconvenientes, pertenece a la tabla movies, no puede ser clave primaria, contiene la popularidad de cada película expresada por una cantidad numérica.
production_companies	Esta columna contiene jsons, pertenece a la tabla production_companies, parte de los json posible clave primaria, contiene las compañías que participaron en la producción de cada película.

production_countries	Esta columna contiene jsons, pertenece a la tabla production_countries, parte de los json posible clave primaria, contiene los países involucrados en la producción de cada película.
release_date	Sin inconvenientes, pertenece a la tabla movies, no puede ser clave primaria, contiene la fecha de estreno de las películas.
revenue	Sin inconvenientes, pertenece a la tabla movies, no puede ser clave primaria, contiene los ingresos que genero cada película.
runtime	Sin inconvenientes, pertenece a la tabla movies, no puede ser clave primaria, contiene la duración de cada película en minutos.
spoken_languages	Esta columna contiene jsons, pertenece a la tabla spoken_languages, parte de los json posible clave primaria, contiene todos los idiomas que se hablan en cada película.
status	Sin inconvenientes, pertenece a la tabla movies, no puede ser clave primaria, contiene el estado en el que se encuentra cada película.
tagline	Sin inconvenientes, pertenece a la tabla movies, no puede ser clave primaria, contiene el eslogan de cada película.
title	Sin inconvenientes, pertenece a la tabla movies, no puede ser clave primaria, contiene el título de las películas.
vote_average	Sin inconvenientes, pertenece a la tabla movies, no puede ser clave primaria, contiene el voto promedio de las películas.
vote_count	Sin inconvenientes, pertenece a la tabla movies, no puede ser clave primaria, contiene el conteo de votos de cada película.
cast	Es una columna compuesta ay que hacer remplazo de caracteres además de manejo de nombres, pertenece a la tabla cast, posible clave primaria, contiene los nombres de los actores que participaron en cada película.
crew	Esta columna contiene jsons ay que hacer remplazo de palabras y remplazo de caracteres para que los json sean válidos, pertenece a la tabla crew, parte de los json posible clave primaria, contiene todo el personal que participo en la producción de cada película.
director	Ay que hacer remplazo de caracteres, pertenece a la tabla director, no puede ser clave primaria, contiene el nombre del director de cada película.

2.1.2 Diagramas conceptual, lógico y físico

En base al análisis echo anteriormente y aplicando los conceptos de modelo entidad relación creamos los siguientes diagramas:

Diagrama Conceptual

Ilustración 3.
Modelo Conceptual

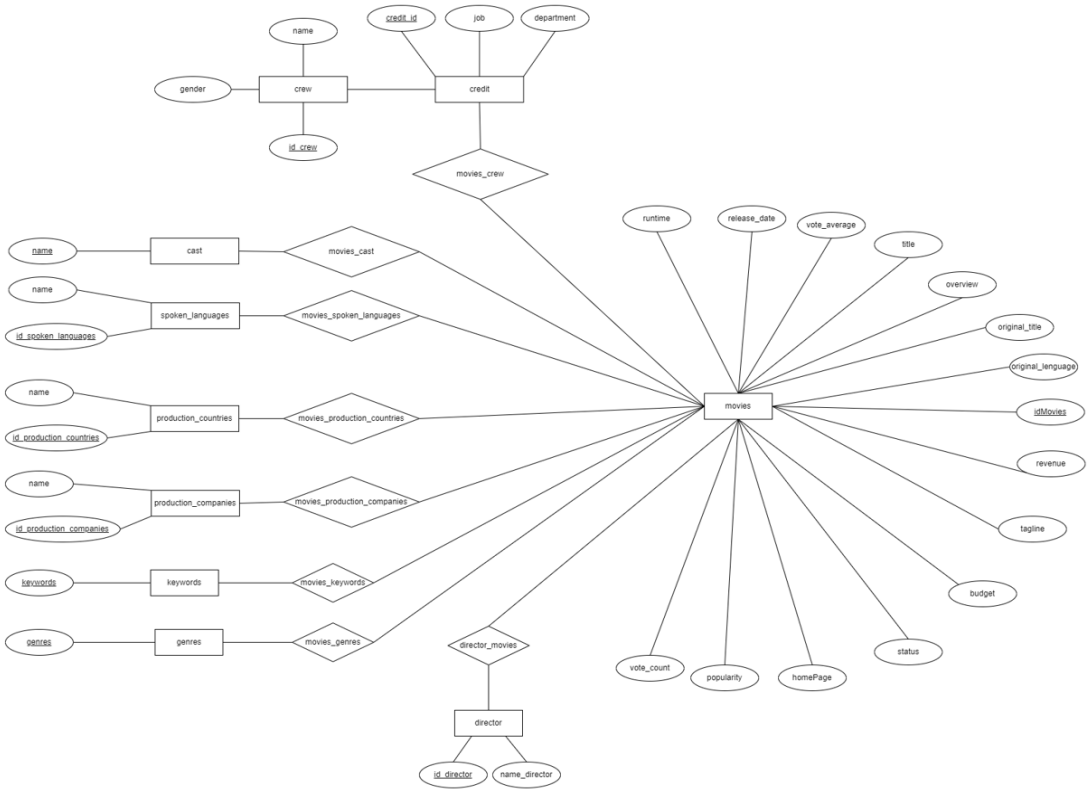


Diagrama Lógico

Ilustración 4.
Modelo Lógico

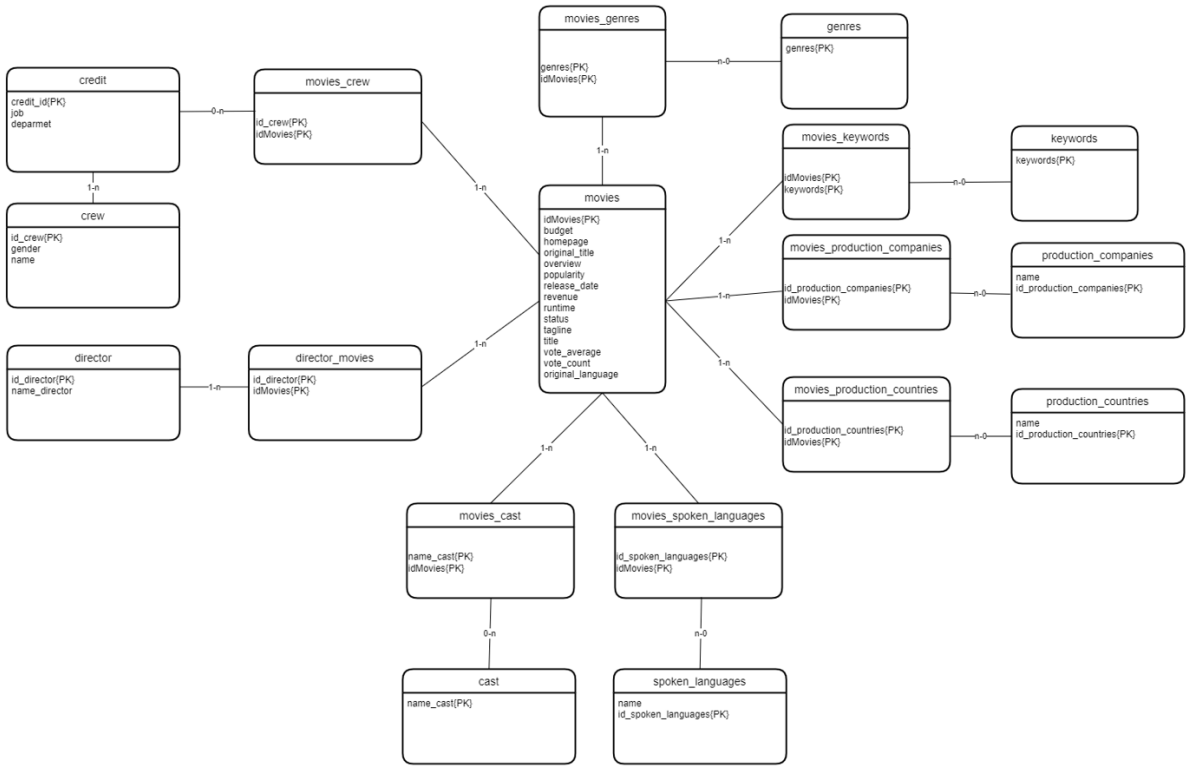
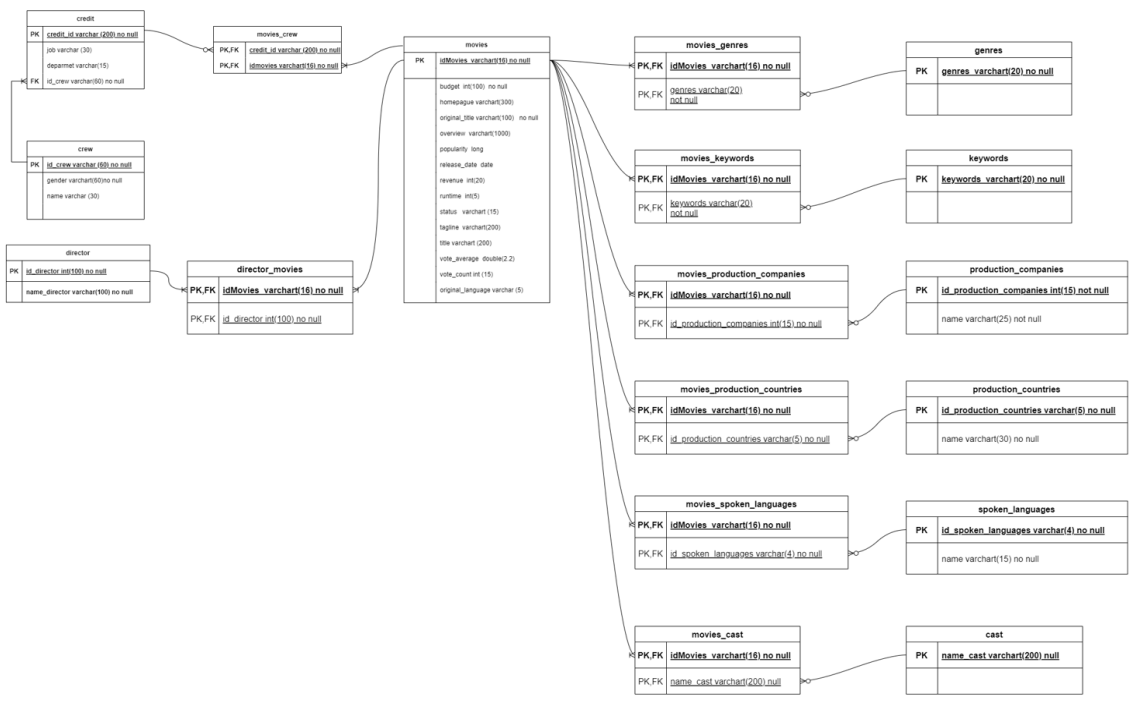


Diagrama Físico

Ilustración 5.
Modelo Físico



2.1.3 Normalización

En esta sección aplicamos las tres primeras formas normales a la tabla general y como resultado se obtuvo las mismas tablas que determinamos aplicando los conceptos del modelo entidad relación:

TABLA GENERAL:

Ilustración 6.
Tabla General

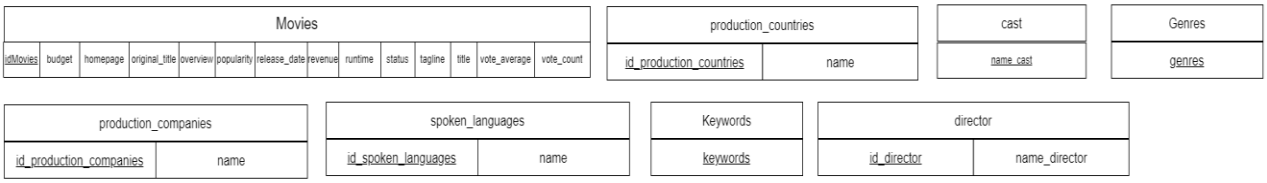
Tabla General																						
index	budget	genres	homepage	keywords	id	original_language	original_title	overview	popularity	production_companies	production_countries	release_date	revenue	runtime	spoken_languages	status	tagline	title	vote_average	vote_count	cast	crew	director

Primera forma normal

Una relación en la que la intersección de todas las filas y columnas contienen un valor y solo un valor.

- Todos los datos son atómicos.
- Las tablas contienen una clave primaria única.

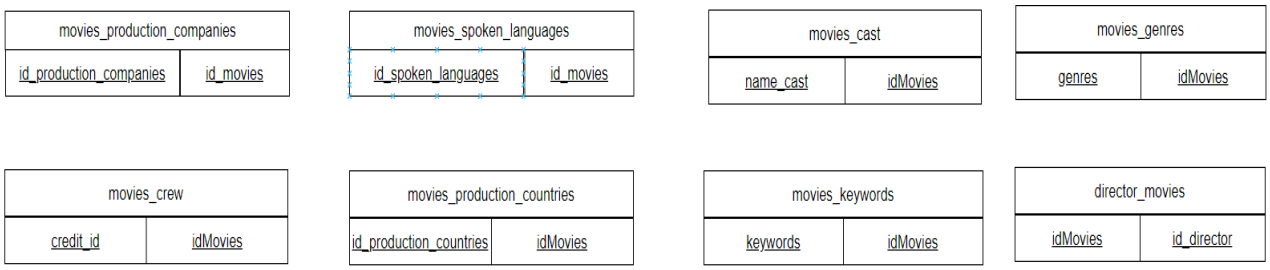
Ilustración 7.
Tablas obtenidas con la Primera Forma Normal 1



Segunda forma normal

Una relación que está en primera forma normal y en la que todo atributo que no sea de clave principal dependa funcionalmente de manera completa de la clave principal.

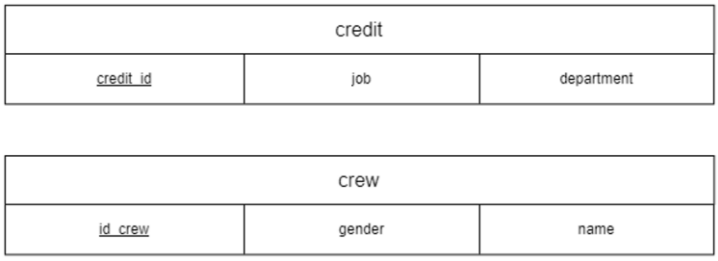
Ilustración 8.
Tablas obtenidas con la Segunda Forma Normal



Tercera forma normal

Una relación que está en primera y segunda formas normales y en la que ningún atributo que no sea clave principal dependa transitivamente de la clave principal.

Ilustración 9.
Tablas obtenidas con la Tercera Forma Normal



2.1.4 Determinación de Dependencias Funcionales

Además, también determinamos las dependencias funcionales en la tabla general e igual que paso con la normalización se generaron las mismas tablas:

Ilustración 10.
Diagrama de Dependencias Funcionales

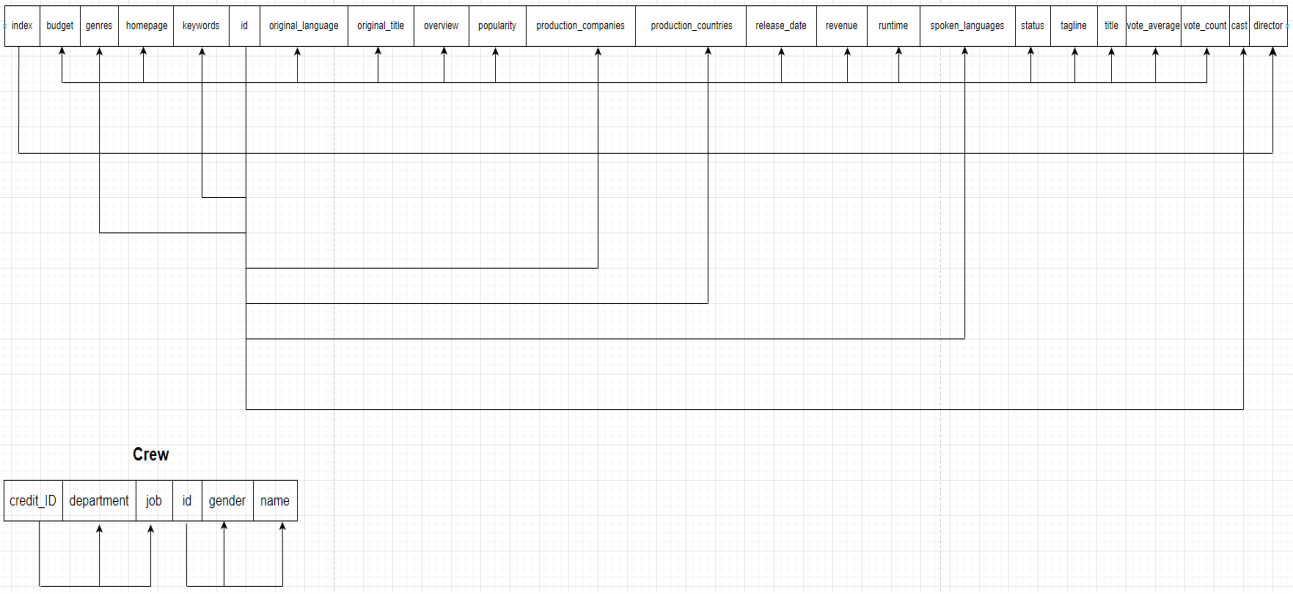


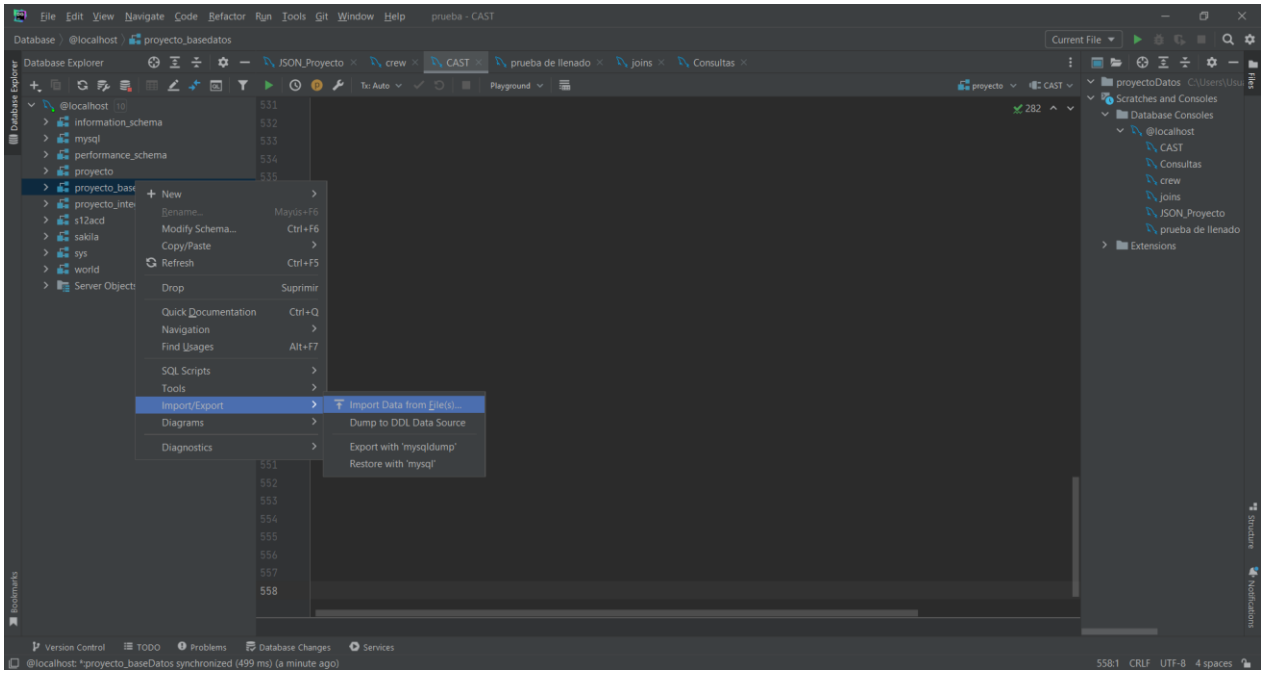
Ilustración 11.
Tablas generadas a partir de dependencias funcionales

DF1	id → {budget, homepage, original_title, overview, popularity, release_date, revenue, runtime, status, tagline, title, vote_average, vote_count, original_language} (Movies)
DF2	director → {index} (Director)
DF3	keywords → {id} (Keywords)
DF4	genres → {id} (Genres)
DF5	production_companies → {id} idMovies → {id_production_companies, name} (production_companies)
DF6	production_countries → {id} idMovies → {id_production_countries, name} (production_countries)
DF7	spoken_languages → {id} idMovies → {id_spoken_languages, name} (spoken_languages)
DF8	cast → {id, name_cast} (cast)
DF9	credit_id → {department, job} (credit)
DF10	id → {gender, name} (crew)

2.1.5 Importación del CSV

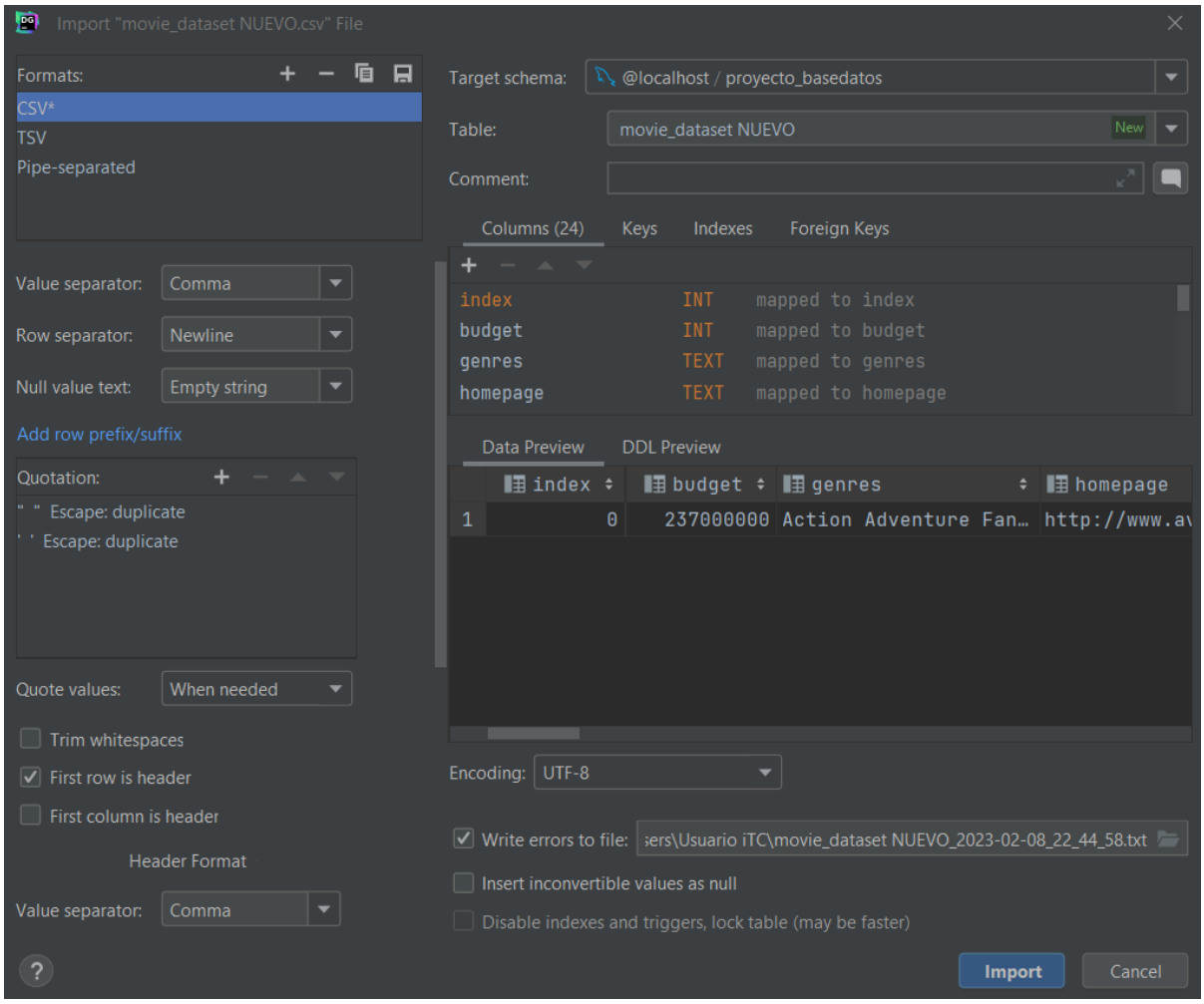
DataGrip es una herramienta desarrollada por JetBrains para manejar bases de datos. Tiene una funcionalidad que permite la importación de diferentes tipos de archivos a tablas MySQL de manera fácil. Los pasos a seguir se muestran en la siguiente imagen:

Ilustración 12.
La funcionalidad de importación de archivos en DataGrip se muestra de manera visual.



En el apartado de importación de DataGrip, se activa una herramienta que permite especificar el separador utilizado en el archivo y también identificar si la primera fila contiene los encabezados de las columnas:

Ilustración 13.
La funcionalidad de importación de archivos en DataGrip es representada en una interfaz intuitiva.



2.1.6 Limpieza de Caracteres Especiales

En muchas de las columnas de la tabla general nos encontramos con un inconveniente muchas letras están en código y no con su símbolo correspondiente lo que puede generar problemas más adelante a sí que toca intercambiar estos códigos por la letra a la que corresponden:

Limpieza Correspondiente a la columna Director

```
DROP TABLE IF EXISTS director_TMP;

CREATE TABLE director_TMP AS

SELECT id, director AS directorOriginal,

REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(director,

'\\u00e9', 'é'),

'\\u00e1', 'á'),

'\\u00f1', 'ñ'),

'\\u00f3', 'ó'),

'\\u00c0', 'À'),

'\\u00e5', 'å'),

'\\u00f6', 'ö'),

'\\u00f4', 'ô'),

'\\u00ed', 'í'),

'\\u00e7', 'ç'),

'\\u00f8', 'ø'),

'\\u0159', 'ř'),

'\\u00c5', 'Å'),

'\\u00d8', 'Ø'),

'\\u00c9', 'É'),

'\\u00e6', 'æ'),

'\\u00e4', 'ä'),

'\\u00e8', 'è'),

'\\u00ef', 'ï'),

'\\u00fb', 'û'),

'\\u00c1', 'Á'),

'\\u0161', 'š'),

'\\u0144s', 'ń'),

'\\u014ct', 'Ō'),

'\\u0142', 'ł'),

'\\u017e', 'ž')

AS directorNuevo
```

```
FROM movies_dataset;
```

Limpieza Correspondiente a la columna Crew

```
DROP TABLE IF EXISTS crew_TMP;

CREATE TABLE crew_TMP AS

SELECT id, crew AS crewOriginal,

REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REP
LACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(R
EPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(RE
PLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLA
CE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(
REPLACE(REPLACE(REPLACE(REPLACE(crew,

'\\u00e9', 'é'),

'\\u00ed', 'í'),

'\\u00e1', 'á'),

'\\u00f3', 'ó'),

'\\u00e2', 'â'),

'\\u00f6', 'ö'),

'\\u00f1', 'ñ'),

'\\u00e7', 'ç'),

'\\u00f4', 'ô'),

'\\u00e5', 'å'),

'\\u00e8', 'è'),

'\\u00eb', 'ë'),

'\\u00e3', 'ã'),

'\\u00fc', 'ü'),

'\\u00ee', 'î'),

'\\u00c9', 'É'),

'\\u00e4', 'ä'),

'\\u00fd', 'ý'),

'\\u00fa', 'ú'),

'\\u00ef', 'ï'),

'\\u00d6', 'Ö'),

'\\u00d4', 'Ô'),
```

```
'\\u00c1', 'Á'),
'\\u00f8', 'ø'),
'\\u00e6', 'æ'),
'\\u00c0', 'À'),
'\\u00e0', 'à'),
'\\u00d3', 'Ó'),
'\\u00fb', 'û'),
'\\u00f0', 'ð'),
'\\u00da', 'Ú'),
'\\u00d8', 'Ø'),
'\\u00c5', 'Å'),
'\\u00df', 'ß'),
'\\u00ea', 'ê'),
'\\u00fe', 'þ'),
'\\u00cd', 'Í'),
'\\u00ec', 'ì'),
'\\u00f5', 'õ'),
'\\u00de', 'Þ'),
'\\u00f2', 'ò'),
'\\u0159','ř'),
'\\u0144','ń'),
'\\u015f','ş'),
'\\u0161','š'),
'\\u0107','ć'),
'\\u0161','š'),
'\\u010d','č'),
'\\u0142','ł'),
'\\u015e','Ş'),
'\\u0411','Б'),
'\\u014c','Ō'),
'\\u010e','Ď'),
'\\u015e','Ş'),
'\\u010d','č'),
'\\u017e','ž'),
```

```
'\\u0107','ć'),
'\\u2019','’'),
'\\u5f20','张'),
'\\u0160','Š'),
'\\u043e','о'),
'\\u011b','ě'),
'\\u0160','Š'),
'\\u0141','Ł'),
'\\u7acb','立'),
'\\u0165','ť'),
'\\u0440','р'),
'\\u010c','Č'),
'\\u017b','Ż'),
'\\u0438','и'),
'\\u0441','с'),
'\\u0421','С'),
'\\u0442','т'),
'\\u0443','у'),
'\\u0433','г'),
'\\u0430','а'),
'\\u0446','ц'),
'\\u043a','k'),
'\\u0439','й')
AS crewNuevo
FROM movies_dataset;
```

Limpieza Correspondiente a la columna Keywords

```
DROP TABLE IF EXISTS keywords_TMP;
CREATE TABLE keywords_TMP AS
SELECT id, keywords AS keywordsOriginal,
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REP
LACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(R
EPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(RE
```

```
PLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(keywords,
'\\u00e9', 'é'),
'\\u00ed', 'í'),
'\\u00e1', 'á'),
'\\u00f3', 'ó'),
'\\u00e2', 'â'),
'\\u00f6', 'ö'),
'\\u00f1', 'ñ'),
'\\u00e7', 'ç'),
'\\u00f4', 'ô'),
'\\u00e5', 'å'),
'\\u00e8', 'è'),
'\\u00eb', 'ë'),
'\\u00e3', 'ã'),
'\\u00fc', 'ü'),
'\\u00ee', 'î'),
'\\u00c9', 'É'),
'\\u00e4', 'ä'),
'\\u00fd', 'ý'),
'\\u00fa', 'ú'),
'\\u00ef', 'ï'),
'\\u00d6', 'Ö'),
'\\u00d4', 'Ô'),
'\\u00c1', 'Á'),
'\\u00f8', 'ø'),
'\\u00e6', 'æ'),
'\\u00c0', 'À'),
'\\u00e0', 'à'),
'\\u00d3', 'Ó'),
'\\u00fb', 'û'),
'\\u00f0', 'ð'),
'\\u00da', 'Ú'),
'\\u00d8', 'Ø'),
```

'\\u00c5', 'Å'),

'\\u00df', 'ß'),

'\\u00ea', 'ê'),

'\\u00fe', 'þ'),

'\\u00cd', 'Í'),

'\\u00ec', 'ì'),

'\\u00f5', 'õ'),

'\\u00de', 'Þ'),

'\\u00f2', 'ò'),

'\\u0159','ř'),

'\\u0144','ń'),

'\\u015f','ş'),

'\\u0161','š'),

'\\u0107','ć'),

'\\u0161','š'),

'\\u010d','č'),

'\\u0142','ł'),

'\\u015e','Ş'),

'\\u0411','Б'),

'\\u014c','Ō'),

'\\u010e','Ď'),

'\\u015e','Ş'),

'\\u010d','č'),

'\\u017e','ž'),

'\\u0107','ć'),

'\\u2019','’'),

'\\u5f20','张'),

'\\u0160','Š'),

'\\u043e','o'),

'\\u011b','ě'),

'\\u0160','Š'),

'\\u0141','Ł'),

'\\u7acb','立'),

'\\u0165','ť'),

```
'\\u0440','р'),
'\\u010c','Č'),
'\\u017b','Ż'),
'\\u0438','и'),
'\\u0441','с'),
'\\u0421','С'),
'\\u0442','т'),
'\\u0443','у'),
'\\u0433','г'),
'\\u0430','а'),
'\\u0446','ц'),
'\\u043a','к'),
'\\u0439','й'),
'\\u00a0',' '),
'\\u7edd','绝'),
'\\u2013','–'),
"'",'')
AS keywordsNuevo
FROM movies_dataset;
```

Limpieza Correspondiente a la columna Original_Title

```
DROP TABLE IF EXISTS original_title_movies_TMP;
CREATE TABLE original_title_movies_TMP AS
SELECT            id,
budget,
homepage,
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REP
LACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(R
EPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(RE
PLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLA
CE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(
REPLACE(REPLACE(REPLACE(original_title,
'キュア', 'Cure'),
','دایره 'El Círculo'),
```

','بچه های آسمان 'Bacheha-Ye aseman'),

'Κυνόδοντας', 'Colmillos'),

'疯狂的石头', 'Crazy Stone'),

'七人の侍', 'Los siete samuráis'),

','جدایی نادر از سیمین 'Una separación'),

'दिल जो भी कहे', 'Dil Jo Bhi Kahey...'),

','سکر بنات 'Caramelo'),

'வாலு', 'Vaalu'),

'एबीसीडी', 'Any Body Can Dance'),

'醉拳二', 'El maestro borracho II'),

'疯狂的赛车', 'Crazy Racer'),

'올드보이', 'Oldboy: cinco días para vengarse'),

'归来', 'Regreso a casa'),

'Географ глобус пропил', 'Geograf globus propil'),

'放·逐', 'Exiled'),

'친절한 금자씨', 'Señora venganza'),

'一個好人', 'El invencible'),

'Снежная королева', 'La reina de las nieves'),

'Z 風暴', 'Tormenta Z'),

'紅番區', 'Masacre en Nueva York'),

'非常幸运', 'Mi estrella de la suerte'),

'アキラ', 'Akira'),

'실미도', 'Silmido'),

'少林足球', 'Siu lam juk kau'),

'कृष', 'Krrish'),

'三城记', 'A Tale of Three Cities'),

'まあだだよ', 'Madadayo'),

'兔侠传奇', 'Legend of a Rabbit'),

'南京!南京!', 'Ciudad de vida y muerte'),

'三枪拍案惊奇', 'San qiang pai an jing qi'),

'괴물', 'The Host'),

'十面埋伏', 'La casa de las dagas voladoras'),

'태극기 휘날리며', 'La hermandad de la guerra'),

'인천상륙작전', 'Operación oculta'),

'剑雨', 'Reino de los Asesinos'),

'新宿事件', 'La venganza del dragón'),

'Возвращение', 'El regreso'),

'刺客聶隱娘', 'La asesina'),

'해운대', 'Tidal Wave'),

'卧虎藏龙', 'El tigre y el dragón'),

'逃出生天', 'La torre del infierno'),

'風暴', 'Firestorm'),

'千と千尋の神隠し', 'El viaje de Chihiro'),

'ராமானுஜன்', 'Ramanujan'),

'三国之见龙卸甲', 'Tres Reinos: La Resurreccion del Dragon'),

'長江七號', 'Cj7: Juguete Del Espacio'),

'黃石的孩子', 'Los niños de China'),

'もののけ姫', 'La princesa Mononoke'),

'功夫', 'Kung-Fusión'),

'十月圍城', 'Bodyguards and Assassins'),

'ハウルの動く城', 'El increíble castillo vagabundo'),

'一個人的武林', 'Kung Fu Jungle'),

'キャプテンハーロック', 'Space Pirate Captain Harlock'),

'辛亥革命', '1911'),

'디워', 'Furia de Dragones'),

'崖の上のポニョ', 'Ponyo y el secreto de la sirenita'),

'一代宗師', 'El arte de la guerra'),

'滿城盡帶黃金甲', 'La maldición de la flor dorada'),

'西游记之孙悟空三打白骨精', 'The Monkey King 2'),

'天將雄師', 'Dragon Blade'),

```
'金陵十三釵', 'Las flores de la guerra'),
'シン・ゴジラ', 'Shin Godzilla'),
'좋은 놈, 나쁜 놈, 이상한 놈', 'El bueno, el malo y el raro'),
'곡성', 'The Wailing'),
'葉問3', 'Ip Man 3'),
'英雄', 'Héroe'),
'投名狀', 'Los señores de la guerra'),
'สุริโยไท', 'La leyenda de Suriyothai'),
'ต้มยำกุ้ง', 'Tom yum goong'),
'कभी अलविदा ना कहना', 'Kabhi Alvida Naa Kehna'),
'Трудно быть богом', 'Que difícil es ser Dios'),
'Солярис', 'Solaris'),
'Савва. Сердце воина', 'Savva y el dragón de fuego'),
'Ночной дозор', 'Guardianes de la noche'),
'Монгол', 'Mongol'),
'Белка и Стрелка. Звёздные собаки', 'Mascotas en el espacio')
AS original_title,
overview,
popularity,
release_date,
revenue,
runtime,
status,
tagline,
title,
vote_average,
vote_count,
original_language
FROM movies_dataset;
```

Limpieza Correspondiente a la columna Production_Companies

```
DROP TABLE IF EXISTS production_companies_TMP;

CREATE TABLE production_companies_TMP AS

SELECT id, production_companies AS production_companiesOriginal,

REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REP
LACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(R
EPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(RE
PLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLA
CE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(production_companie
s,
'\\u00e9', 'é'),
'\\u00ed', 'í'),
'\\u00e1', 'á'),
'\\u00f3', 'ó'),
'\\u00e2', 'â'),
'\\u00f6', 'ö'),
'\\u00f1', 'ñ'),
'\\u00e7', 'ç'),
'\\u00f4', 'ô'),
'\\u00e5', 'å'),
'\\u00e8', 'è'),
'\\u00eb', 'ë'),
'\\u00e3', 'ã'),
'\\u00fc', 'ü'),
'\\u00ee', 'î'),
'\\u00c9', 'É'),
'\\u00e4', 'ä'),
'\\u00fd', 'ý'),
'\\u00fa', 'ú'),
'\\u00ef', 'ï'),
'\\u00d6', 'Ö'),
'\\u00d4', 'Ô'),
'\\u00c1', 'Á'),
```

'\\u00f8', 'ø'),

'\\u00e6', 'æ'),

'\\u00c0', 'À'),

'\\u00e0', 'à'),

'\\u00d3', 'Ó'),

'\\u00fb', 'û'),

'\\u00f0', 'ð'),

'\\u00da', 'Ú'),

'\\u00d8', 'Ø'),

'\\u00c5', 'Å'),

'\\u00df', 'ß') ,

'\\u00ea', 'ê'),

'\\u00fe', 'þ'),

'\\u00cd', 'Í'),

'\\u00ec', 'ì'),

'\\u00f5', 'õ'),

'\\u00de', 'Þ'),

'\\u00f2', 'ò'),

'\\u0159','ř'),

'\\u0144','ń'),

'\\u015f','ş'),

'\\u0161','š'),

'\\u0107','ć'),

'\\u0161','š'),

'\\u010d','č'),

'\\u0142','ł'),

'\\u015e','Ş'),

'\\u0411','Б'),

'\\u014c','Ō'),

'\\u010e','Ď'),

'\\u015e','Ş'),

'\\u010d','č'),

'\\u017e','ž'),

'\\u0107','ć'),

```
'\\u2019','’'),
'\\u5f20','张'),
'\\u0160','Š'),
'\\u043e','о'),
'\\u011b','ě'),
'\\u0160','Š'),
'\\u0141','Ł'),
'\\u7acb','立'),
'\\u0165','ť'),
'\\u0440','р'),
'\\u010c','Č'),
'\\u017b','Ż'),
'\\u0438','и'),
'\\u0441','с'),
'\\u0421','С'),
'\\u0442','т'),
'\\u0443','у'),
'\\u0433','г'),
'\\u0430','а'),
'\\u0446','ц'),
'\\u043a','к'),
'\\u0439','й'),
'\\u00b0','°'),
'\\u00b2','²'),
'\\u00ce','Î')
AS production_companiesNuevo
FROM movies_dataset;
```

Limpieza Correspondiente a la columna Production_Countries

```
DROP TABLE IF EXISTS production_countries_TMP;

CREATE TABLE production_countries_TMP AS

SELECT id, production_countries AS production_countriesOriginal,

REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REP
LACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(R
EPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(RE
PLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLA
CE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(production_countrie
s,
'\\u00e9', 'é'),
'\\u00ed', 'í'),
'\\u00e1', 'á'),
'\\u00f3', 'ó'),
'\\u00e2', 'â'),
'\\u00f6', 'ö'),
'\\u00f1', 'ñ'),
'\\u00e7', 'ç'),
'\\u00f4', 'ô'),
'\\u00e5', 'å'),
'\\u00e8', 'è'),
'\\u00eb', 'ë'),
'\\u00e3', 'ã'),
'\\u00fc', 'ü'),
'\\u00ee', 'î'),
'\\u00c9', 'É'),
'\\u00e4', 'ä'),
'\\u00fd', 'ý'),
'\\u00fa', 'ú'),
'\\u00ef', 'ï'),
'\\u00d6', 'Ö'),
'\\u00d4', 'Ô'),
'\\u00c1', 'Á'),
```

'\\u00f8', 'ø'),

'\\u00e6', 'æ'),

'\\u00c0', 'À'),

'\\u00e0', 'à'),

'\\u00d3', 'Ó'),

'\\u00fb', 'û'),

'\\u00f0', 'ð'),

'\\u00da', 'Ú'),

'\\u00d8', 'Ø'),

'\\u00c5', 'Å'),

'\\u00df', 'ß') ,

'\\u00ea', 'ê'),

'\\u00fe', 'þ'),

'\\u00cd', 'Í'),

'\\u00ec', 'ì'),

'\\u00f5', 'õ'),

'\\u00de', 'Þ'),

'\\u00f2', 'ò'),

'\\u0159','ř'),

'\\u0144','ń'),

'\\u015f','ş'),

'\\u0161','š'),

'\\u0107','ć'),

'\\u0161','š'),

'\\u010d','č'),

'\\u0142','ł'),

'\\u015e','Ş'),

'\\u0411','Б'),

'\\u014c','Ō'),

'\\u010e','Ď'),

'\\u015e','Ş'),

'\\u010d','č'),

'\\u017e','ž'),

'\\u0107','ć'),

```
'\\u2019','’'),
'\\u5f20','张'),
'\\u0160','Š'),
'\\u043e','о'),
'\\u011b','ě'),
'\\u0160','Š'),
'\\u0141','Ł'),
'\\u7acb','立'),
'\\u0165','ť'),
'\\u0440','р'),
'\\u010c','Č'),
'\\u017b','Ż'),
'\\u0438','и'),
'\\u0441','с'),
'\\u0421','С'),
'\\u0442','т'),
'\\u0443','у'),
'\\u0433','г'),
'\\u0430','а'),
'\\u0446','ц'),
'\\u043a','к'),
'\\u0439','й'),
'\\u00b0','°'),
'\\u00b2','²'),
'\\u00ce','Î')
AS production_countriesNuevo
FROM movies_dataset;
```

Limpieza Correspondiente a la columna Spoken_Languages

```
DROP TABLE IF EXISTS spoken_languages_TMP;

CREATE TABLE spoken_languages_TMP AS

SELECT id, spoken_languages AS spoken_languagesOriginal,

REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REP
LACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(R
EPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(RE
PLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLA
CE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REP
LACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(R
EPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(RE
PLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLA
CE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REP
LACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(R
EPLACE(spoken_languages,

'\\u00e9', 'é'),

'\\u00ed', 'í'),

'\\u00e1', 'á'),

'\\u00f3', 'ó'),

'\\u00e2', 'â'),

'\\u00f6', 'ö'),

'\\u00f1', 'ñ'),

'\\u00e7', 'ç'),

'\\u00f4', 'ô'),

'\\u00e5', 'å'),

'\\u00e8', 'è'),

'\\u00eb', 'ë'),

'\\u00e3', 'ã'),

'\\u00fc', 'ü'),

'\\u00ee', 'î'),

'\\u00c9', 'É'),

'\\u00e4', 'ä'),
```

```
'\\u00fd', 'ý'),
'\\u00fa', 'ú'),
'\\u00ef', 'ï'),
'\\u00d6', 'Ö'),
'\\u00d4', 'Ô'),
'\\u00c1', 'Á'),
'\\u00f8', 'ø'),
'\\u00e6', 'æ'),
'\\u00c0', 'À'),
'\\u00e0', 'à'),
'\\u00d3', 'Ó'),
'\\u00fb', 'û'),
'\\u00f0', 'ð'),
'\\u00da', 'Ú'),
'\\u00d8', 'Ø'),
'\\u00c5', 'Å'),
'\\u00df', 'ß') ,
'\\u00ea', 'ê'),
'\\u00fe', 'þ'),
'\\u00cd', 'Í'),
'\\u00ec', 'ì'),
'\\u00f5', 'õ'),
'\\u00de', 'Þ'),
'\\u00f2', 'ò'),
'\\u0159','ř'),
'\\u0144','ń'),
'\\u015f','ş'),
'\\u0161','š'),
'\\u0107','ć'),
'\\u0161','š'),
'\\u010d','č'),
'\\u0142','ł'),
'\\u015e','Ş'),
'\\u0411','Б'),
```

'\\u014c','Ō'),

'\\u010e','Ď'),

'\\u015e','Ş'),

'\\u010d','č'),

'\\u017e','ž'),

'\\u0107','ć'),

'\\u2019','’'),

'\\u5f20','张'),

'\\u0160','Š'),

'\\u043e','o'),

'\\u011b','ě'),

'\\u0160','Š'),

'\\u0141','Ł'),

'\\u7acb','立'),

'\\u0165','ť'),

'\\u0440','p'),

'\\u010c','Č'),

'\\u017b','Ż'),

'\\u0438','и'),

'\\u0441','c'),

'\\u0421','C'),

'\\u0442','т'),

'\\u0443','y'),

'\\u0433','r'),

'\\u0430','a'),

'\\u0446','ц'),

'\\u043a','k'),

'\\u0439','й'),

'\\u00b0','°'),

'\\u00b2','²'),

'\\u00ce','Î'),

'\\ud55c','한'),

'\\u666e','普'),

'\\u65e5','日'),

'\\u5e7f','广'),

'\\u1ebf','ế'),

'\\u10e5','ქ'),

'\\u0e20','ภ'),

'\\u0c24','త'),

'\\u0ba4','த'),

'\\u0a2a','ਪ'),

'\\u09ac','ব'),

'\\u0939','ह'),

'\\u067e','پ'),

'\\u0641','ف'),

'\\u0627','ا'),

'\\u05e2','ע'),

'\\u049b','қ'),

'\\u0431','б'),

'\\u0423','У'),

'\\u03b5','ε'),

'\\u0103','ă'),

'\\u03bb','λ'),

'\\u0e32','า'),

'\\u093f','ि'),

'\\u0631','ر'),

'\\u0644','ل'),

'\\u05b4','ִ'),

'\\u044a','ы'),

'\\u0457','ї'),

'\\u0a70','ੰ'),

'\\u0437','з'),

'\\u0c46','ె'),

'\\u069a','ښ'),

'\\u0bae','ம'),

'\\u09be','া'),

'\\u901a','通'),

'\\u672c','本'),

'\\uad6d','国'),

'\\u5dde','州'),

'\\u1ec7','ệ'),

'\\u10d0','ა'),

'\\u03b7','η'),

'\\u0e29','ษ '),

'\\u0928','न '),

'\\u0633','س'),

'\\u062f','د'),

'\\u0639','ع'),

'\\u05d1','ב'),

'\\u043b','л'),

'\\u043d','н'),

'\\u0a1c','ਜ '),

'\\u0c32','ಲ '),

'\\u062a','ت'),

'\\u0bbf','ி'),

'\\u0982','ং'),

'\\u8bdd','话'),

'\\u8a9e','語'),

'\\uc5b4','어'),

'\\u10e0','რ'),

'\\u03bd','ν'),

'\\u0e44',' ไ'),

'\\u094d',' ्'),

'\\u06cc','ی'),

'\\u0648','و'),

'\\u0628','ب'),

'\\u05b0','ְ'),

'\\u0435','e'),

'\\u044c','ь'),

```
'\\u0a3e','ਾ '),
'\\u0c41','ు '),
'\\u0bb4','ழ '),
'\\u09b2','ল '),
'\\u5ee3','廣'),
'\\uc870','윰'),
'\\u10d7','თ'),
'\\u03b9','ι'),
'\\u0e17','ท '),
'\\u0926','द '),
'\\u064a','ي'),
'\\u05e8','ר'),
'\\u0a2c','ਬ '),
'\\u0c17','గ'),
'\\u0bcd','் '),
'\\u8a71','話'),
'\\uc120','저'),
'\\u10e3','ჳ'),
'\\u03ba','κ'),
'\\u0e22','ย'),
'\\u0940','ी'),
'\\u0629','ة'),
'\\u05d9','י'),
'\\u0a40','ਠ '),
'\\ub9d0','볐'),
'\\u10da','ღ'),
'\\u03ac','ά'),
'\\u05ea','ת'),
'\\u10d8','ი')
AS spoken_languagesNuevo
FROM movies_dataset;
```

2.1.7 Extracción correcta de datos de las columnas tipo Json (Production_Countries, Production_Companies, Spoken_Languages y Crew)

En la tabla general ay cuatro columnas que contienen Json y de estos Json toca extraer la data para poder llenar las tablas finales, otra cosa a destacar la columna Crew otro proceso que hay que realizar es la validación y correccion de cada Json, por ende, a continuación, se muestra el código usado para realizar este proceso:

Columna Crew resuelto con procedimientos

```
-- Crew Temporal Validado y la nueva tabla crew llenada DEFINITIVA
DROP TABLE IF EXISTS crew_TMP2;
CREATE TABLE crew_TMP2 AS
SELECT id,
CONVERT (
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(crewNuevo,
'"', '\''),
'{\'', '{"'),
'\': \'', '": "'),
'\', \'', '", "'),
'\': ', '": '),
', \'', ', "')
USING UTF8mb4 ) AS crew_new
FROM crew_tmp;
DROP TABLE IF EXISTS crew_TMP3;
CREATE TABLE crew_TMP3 (
id_movie VARCHAR(250),
name VARCHAR(250) ,
gender VARCHAR(250),
department VARCHAR(250),
job VARCHAR(250),
credit_id VARCHAR(250),
id VARCHAR(250)
);
-- Extraccion de la data de Json CREW
```

```
DROP PROCEDURE IF EXISTS cursor_json_crew ;

DELIMITER $$

CREATE PROCEDURE cursor_json_crew ()

BEGIN

DECLARE done INT DEFAULT FALSE ;

DECLARE i INT DEFAULT 0 ;

DECLARE jsonData json ;

DECLARE idMovie varchar(250);

DECLARE idMovie2 varchar(250);

DECLARE jsonId varchar(250) ;

DECLARE jsonLabel varchar(250) ;

DECLARE jsonLabel2 varchar(250) ;

DECLARE jsonLabel3 varchar(250) ;

DECLARE jsonLabel4 varchar(250) ;

DECLARE jsonLabel5 varchar(250) ;

DECLARE resultSTR LONGTEXT DEFAULT '';

-- Declarar el cursor

DECLARE myCursor

CURSOR FOR

SELECT JSON_EXTRACT(CONVERT(crew_new USING UTF8MB4), '$[*]') FROM crew_tmp2;

DECLARE myCursor2

CURSOR FOR

SELECT id FROM crew_tmp2;

-- Declarar el handler para NOT FOUND (esto es marcar cuando el cursor ha
llegado a su fin)

DECLARE CONTINUE HANDLER

FOR NOT FOUND SET done = TRUE ;

-- Abrir el cursor

OPEN myCursor ;

OPEN myCursor2 ;

cursorLoop: LOOP

FETCH myCursor INTO jsonData ;

FETCH myCursor2 INTO idMovie ;

SET i = 0;

-- Si alcanzo el final del cursor entonces salir del ciclo repetitivo
```

```
IF done THEN

LEAVE cursorLoop ;

END IF ;

WHILE JSON_EXTRACT(jsonData, CONCAT('$[', i, ']')) IS NOT NULL DO

SET idMovie2 = idMovie;

SET jsonId = IFNULL(JSON_EXTRACT(jsonData, CONCAT('$[', i, '].name')) , '')
;

SET jsonLabel = IFNULL(JSON_EXTRACT(jsonData, CONCAT('$[', i,'].gender')) ,
'') ;

SET     jsonLabel2     =     IFNULL(JSON_EXTRACT(jsonData,     CONCAT('$[',
i,'].department')) , '') ;

SET jsonLabel3 = IFNULL(JSON_EXTRACT(jsonData, CONCAT('$[', i,'].job')) ,
'') ;

SET     jsonLabel4     =     IFNULL(JSON_EXTRACT(jsonData,     CONCAT('$[',
i,'].credit_id')) , '') ;

SET jsonLabel5 = IFNULL(JSON_EXTRACT(jsonData, CONCAT('$[', i,'].id')) , '')
;

SET  resultSTR  =  CONCAT(resultSTR,  '  INSERT  INTO  crew_tmp3  VALUES  (',
idMovie2, ', ', jsonId, ', ', jsonLabel, ', ', jsonLabel2, ', ', jsonLabel3,
', ', jsonLabel4, ', ', jsonLabel5, '); ');

SET i = i + 1;

SET @sql_text = CONCAT('INSERT INTO crew_tmp3 VALUES (', idMovie2, ', ',
REPLACE(jsonId, '\' ', ''), ', ', jsonLabel, ', ', jsonLabel2, ', ',
jsonLabel3, ', ', jsonLabel4, ', ', jsonLabel5, '); ');

PREPARE stmt FROM @sql_text;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

END WHILE ;

END LOOP ;

SELECT jsonId, jsonLabel, resultSTR ;

CLOSE myCursor ;

END$$

DELIMITER ;

CALL cursor_json_crew ();
```

Columna Spoken_Languages resuelto con procedimientos

```
DROP TABLE IF EXISTS spoken_languages_TMPX;

CREATE TABLE spoken_languages_TMPX (

id_movie VARCHAR(250),

iso_639_1 VARCHAR(250) ,

name VARCHAR(250)

);

DROP PROCEDURE IF EXISTS cursor_json_example3 ;

DELIMITER $$

CREATE PROCEDURE cursor_json_example3 ()

BEGIN

DECLARE done INT DEFAULT FALSE ;

DECLARE i INT DEFAULT 0 ;

DECLARE i2 INT DEFAULT 0 ;

DECLARE jsonData json ;

DECLARE idMovie varchar(250);

DECLARE idMovie2 varchar(250);

DECLARE jsonId varchar(250) ;

DECLARE jsonLabel varchar(250) ;

DECLARE resultSTR LONGTEXT DEFAULT '';

-- Declarar el cursor

DECLARE myCursor

CURSOR FOR

SELECT JSON_EXTRACT(CONVERT(spoken_languagesNuevo USING UTF8MB4), '$[*]')
FROM spoken_languages_tmp;

DECLARE myCursor2

CURSOR FOR

SELECT id FROM movies_dataset;

-- Declarar el handler para NOT FOUND (esto es marcar cuando el cursor ha
llegado a su fin)

DECLARE CONTINUE HANDLER

FOR NOT FOUND SET done = TRUE ;

-- Abrir el cursor

OPEN myCursor ;

OPEN myCursor2 ;
```

```
cursorLoop: LOOP

FETCH myCursor INTO jsonData ;

FETCH myCursor2 INTO idMovie ;

SET i = 0;

-- Si alcanzo el final del cursor entonces salir del ciclo repetitivo

IF done THEN

LEAVE cursorLoop ;

END IF ;

WHILE JSON_EXTRACT(jsonData, CONCAT('$[', i, ']')) IS NOT NULL DO

SET idMovie2 = idMovie;

SET jsonId = IFNULL(JSON_EXTRACT(jsonData, CONCAT('$[', i, '].iso_639_1'))
, '') ;

SET jsonLabel = IFNULL(JSON_EXTRACT(jsonData, CONCAT('$[', i,'].name')) ,
'') ;

SET resultSTR = CONCAT(resultSTR, ' INSERT INTO spoken_languages_TMPX VALUES
(', idMovie2, ', ', jsonId, ', ', jsonLabel, '); ');

SET i = i + 1;

SET @sql_text = CONCAT('INSERT INTO spoken_languages_TMPX VALUES (',
idMovie2, ', ', REPLACE(jsonId, '\' ', ''), ', ', jsonLabel, '); ');

PREPARE stmt FROM @sql_text;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

END WHILE ;

END LOOP ;

SELECT jsonId, jsonLabel, resultSTR ;

CLOSE myCursor ;

END$$

DELIMITER ;

CALL cursor_json_example3 ();
```

Columna Production_Companies resuelto con union

```
DROP TABLE IF EXISTS production_companies_TMP2;

CREATE TABLE production_companies_TMP2 AS

SELECT * FROM (

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[0].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[0].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[1].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[1].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[2].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[2].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[3].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[3].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[4].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[4].id')) as ProductionCompanyId

FROM production_companies_tmp
```

```
UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[5].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[5].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[6].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[6].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[7].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[7].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[8].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[8].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[9].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[9].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[10].name')) as ProductionCompanyName,
```

```
JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[10].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[11].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[11].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[12].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[12].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[13].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[13].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[14].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[14].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[15].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[15].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION
```

```
SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[16].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[16].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[17].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[17].id')) as ProductionCompanyId

FROM production_companies_tmp

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[18].name')) as ProductionCompanyName,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_companiesNuevo using utf8mb4),
'$[18].id')) as ProductionCompanyId

FROM production_companies_tmp

) z_q WHERE ProductionCompanyName is not null AND ProductionCompanyId is not
null

ORDER BY z_q.id ASC ;
```

Columna Production_Countries resuelto con union

```
DROP TABLE IF EXISTS production_countries_TMP2;

CREATE TABLE production_countries_TMP2 AS

SELECT * FROM (

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_countriesNuevo using utf8mb4),
'$[0].iso_3166_1')) as ProductionCountriesId,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_countriesNuevo using utf8mb4),
'$[0].name')) as ProductionCountriesNames

FROM production_countries_TMP

UNION

SELECT id,
```

```
JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_countriesNuevo using utf8mb4),
'$[1].iso_3166_1')) as ProductionCountriesId,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_countriesNuevo using utf8mb4),
'$[1].name')) as ProductionCountriesNames

FROM production_countries_TMP

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_countriesNuevo using utf8mb4),
'$[2].iso_3166_1')) as ProductionCountriesId,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_countriesNuevo using utf8mb4),
'$[2].name')) as ProductionCountriesNames

FROM production_countries_TMP

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_countriesNuevo using utf8mb4),
'$[3].iso_3166_1')) as ProductionCountriesId,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_countriesNuevo using utf8mb4),
'$[3].name')) as ProductionCountriesNames

FROM production_countries_TMP

UNION

SELECT id,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_countriesNuevo using utf8mb4),
'$[4].iso_3166_1')) as ProductionCountriesId,

JSON_UNQUOTE(JSON_EXTRACT(CONVERT(production_countriesNuevo using utf8mb4),
'$[4].name')) as ProductionCountriesNames

FROM production_countries_TMP

) z_q WHERE ProductionCountriesId is not null AND ProductionCountriesNames
is not null

ORDER BY z_q.id ASC ;
```

2.1.8 Extracción correcta de datos de las columnas Genres y Cast

En esta sección se trata las columnas Genres y Cast que son columnas multivaluadas por esta razón lo que toca hacer es transformarlas en columnas atómicas y a continuación está el código usado para hacer esta transformación.

Columna Genres

```
DROP TABLE IF EXISTS Movies_Genres_TMP;
CREATE TABLE Movies_Genres_TMP AS
SELECT id,
genres as genresOrig ,
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REP
LACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(R
EPLACE(genres,
'Science Fiction', 'Science_Fiction'),
'Action Adventure', 'Action_Adventure'),
'Action Comedy', 'Action_Comedy'),
'Thriller Drama', 'Thriller_Drama'),
'Fantasy Adventure', 'Fantasy_Adventure'),
'Horror Comedy', 'Horror_Comedy'),
'Crime Drama', 'Crime_Drama'),
'Romance Drama', 'Romance_Drama'),
'Western Comedy', 'Western_Comedy'),
'War Drama', 'War_Drama'),
'Family Comedy','Family_Comedy'),
'Documentary Drama','Documentary_Drama'),
'Mystery Thriller','Mystery_Thriller'),
'Action Thriller','Action_Thriller'),
'Romance Comedy Drama','Romance_Comedy-Drama'),
'Fantasy Comedy.','Fantasy_Comedy.'),
'Comedy Drama','Comedy_Drama'),
'Crime Comedy','Crime_Comedy'),
'Drama Comedy','Drama_Comedy'),
```

```
'Horror Drama','Horror_Drama'),
'Romance Thriller','Romance_Thriller'),
'Fantasy Thriller','Fantasy_Thriller'),
'Action Crime','Action_Crime'),
'Fantasy Crime','Fantasy_Crime'),
'Thriller Crime','Thriller_Crime'),
'Action Drama','Action_Drama'),
'Romance Adventure','Romance_Adventure'),
'Fantasy Adventure','Fantasy_Adventure'),
'Thriller Adventure','Thriller_Adventure')
 AS genresNew
FROM movies_dataset;
--
DROP TABLE IF EXISTS Movies_Genres;
CREATE TABLE Movies_Genres AS
SELECT id,
SUBSTRING_INDEX(genresNew, ' ', 1) AS genre
FROM Movies_Genres_TMP
UNION
SELECT id,
SUBSTRING_INDEX(SUBSTRING_INDEX(genresNew, ' ', 2), ' ', -1) AS genre
FROM Movies_Genres_TMP
UNION
SELECT id,
SUBSTRING_INDEX(SUBSTRING_INDEX(genresNew, ' ', 3), ' ', -1) AS genre
FROM Movies_Genres_TMP
UNION
SELECT id,
SUBSTRING_INDEX(SUBSTRING_INDEX(genresNew, ' ', 4), ' ', -1) AS genre
FROM Movies_Genres_TMP
UNION
SELECT id,
SUBSTRING_INDEX(SUBSTRING_INDEX(genresNew, ' ', 5), ' ', -1) AS genre
FROM Movies_Genres_TMP
```

```
UNION
SELECT id,
SUBSTRING_INDEX(SUBSTRING_INDEX(genresNew, ' ', 6), ' ', -1) AS genre
FROM Movies_Genres_TMP;
```

Columna Cast

```
DROP TABLE IF EXISTS cast;
CREATE TABLE cast AS
SELECT id, cast
FROM movies_dataset;
DROP TABLE IF EXISTS cast_TMP;
CREATE TABLE cast_TMP AS
SELECT id,
cast as castOriginal ,
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REP
LACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(R
EPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(RE
PLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLA
CE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REP
LACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(R
EPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(RE
PLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLA
CE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REP
LACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(R
EPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(RE
PLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLA
CE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REP
LACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(R
EPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(RE
PLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLA
CE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REP
```

```
LACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(R
EPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(RE
PLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLA
CE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REP
LACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(R
EPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(RE
PLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(cast,

 'Jennifer Jason Leigh', 'Jennifer Jason_Leigh'),

 '\\u00ef', 'ï'),

 '\\u00e9', 'é'),

 '\\u00d3', 'Ó'),

 '\\u00f1', 'ñ'),

 '\\u00c1', 'Á'),

 '\\u00e0', 'à'),

'\\u00f3n', 'ó'),

'\\u00fcr', 'ü'),

'\\u00f4', 'ô'),

'\\u00e4t', 'ä'),

'\\u00fa', 'ú'),

'\\u00c0l', 'À'),

'\\u00fas', 'ú'),

'\\u00e0', 'à'),

'\\u00fc', 'ú'),

'\\u00f4', 'ô'),

'\\u00fa', 'ú'),

'\\u00df', 'ß'),

'\\u00ea', 'ê'),

'\\u00c9', 'É'),

'\\u00c0', 'À'),

'\\u00d8', 'Ø'),

'\\u00dc', 'Ü'),

'\\u00e9', 'é'),

'\\u00e1', 'á'),
```

```
'\\u00f1', 'ñ'),
'\\u00f3', 'ó'),
'\\u00c0', 'À'),
'\\u00e5', 'å'),
'\\u00f6', 'ö'),
'\\u00f4', 'ô'),
'\\u00ed', 'í'),
'\\u00e7', 'ç'),
'\\u00f8', 'ø'),
'\\u0159', 'ř'),
'\\u00c5', 'Å'),
'\\u00d8', 'Ø'),
'\\u00c9', 'É'),
'\\u00e6', 'æ'),
'\\u00e4', 'ä'),
'\\u00e8', 'è'),
'\\u00ef', 'ï'),
 '\\u00e1', 'á'),
 '\\u00e5rd', 'á'),
 '\\u00f6', 'ö'),
 '\\u00e8', 'è'),
 '\\u00ed', 'í'),
 '\\u016b', 'ū'),
'\\u014d', 'ō'),
 '\\u00f8', 'ø'),
 '\\u00e6', 'æ'),
 '\\u00eb', 'ë'),
 '\\u00c5', 'Å'),
 '\\u042e', 'Ю'),
'\\u043b', 'л'),
'\\u0438', 'и'),
'\\u044f', 'я'),
'\\u0421', 'С'),
'\\u043d', 'н'),
```

```
'\\u0433', 'г'),
'\\u0440', 'р'),
'\\u044c', 'ь'),
 '\\u015b', 'ś'),
 '\\u067e', 'پ'),
 '\\u0219', 'ș'),
 '\\u0110', 'Đ'),
 '\\u00fbin', 'û'),
 '\\u0101', 'ā'),
 '\\u00ee', 'î'),
 ' von ', ' von_'),
 'Frederique van der Wal', 'Frederique van_der_Wal'),
 'Tupac Amaru Shakur', 'Tupac Amaru_Shakur"'),
 ' de la ', ' de_la_'),
 ' De ', ' De_'),
 'G. W. Bailey', 'G._W. Bailey'),
 'Ana de la Reguera', 'Ana de_la_Reguera'),
 'Neil Patrick Harris', 'Neil Patrick_Harris'),
 'Jo', 'Jo_'),
 'Pelé ', 'Pelé --- '),
 ' . ', ' ._'),
 'Madonna ', 'Madonna ---- '),
 'Pink ', 'Pink ---- '),
 'Frances de la Tour', 'Frances de_la_Tour'),
 'Paz de la Huerta', 'Paz de_la_Huerta'),
 'Rodrigo de la Serna', 'Rodrigo de_la_Serna'),
 'Rapahel de la Sierra', 'Raphael de_la_Sierra'),
 'Samuel L. Jackson', 'Samuel L._Jackson'),
'Helena Bonham Carter', 'Helena Bonham_Carter'),
'Don', 'Don_'),
'LL Cool J', 'LL Cool_J'),
 'Bryce Dallas Howard', 'Bryce Dallas_Howard'),
 'David Ogden Stiers', 'David Ogden_Stiers'),
' Jr. ', '_Jr. '),
```

```
 ' Jr.', '_Jr.'),
'Catherine Zeta-Jones', 'Catherine Zeta_Jones'),
 'Martin Smith', 'Martin_Smith'),
 'Thomas Haden Church', 'Thomas Haden_Church'),
 'James Badge Dale', 'James Badge_Dale'),
 'Tommy Lee', 'Tommy_Lee'),
 'Dakota Blue', 'Dakota_Blue'),
 'Rihanna ', 'Rihanna ---- '),
 'M. Emmet', 'M._Emmet'),
 'Jo_hn Michael', 'Jo_hn_Michael'),
 'R. D.', 'R._D.'),
 'Jo_hn C.', 'Jo_hn_C.'),
 ' del ', ' del_'),
 'Baron Cohen', 'Baron_Cohen'),
 'Chloë Grace Moretz', 'Chloë_Grace Moretz'),
 'Billy Bob', 'Billy_Bob'),
 'Lara Flynn', 'Lara_Flynn'),
 'Jeffrey Dean', 'Jeffrey_Dean'),
 'William H.', 'William_H.'),
 'Tim Blake', 'Tim_Blake'),
 'D. B.', 'D._B.'),
 ' Ludacris', ' ---- Ludacris'),
 'Jay O.', 'Jay_O.'),
 'Michael J.', 'Michael_J.'),
 'Jackie Earle', 'Jackie_Earle'),
 'Jo_nny Lee', 'Jo_nny_Lee'),
 'Will Yun', 'Will_Yun'),
 'Jamie Renée', 'Jamie_Renée'),
 'Michael B.', 'Michael_B.'),
 'Dick Van', 'Dick_Van'),
 ' de ', ' de_'),
 'Mario Van', 'Mario_Van'),
 'Michael-Leon', 'Michael_Leon'),
 'Anika Noni', 'Anika_Noni'),
```

```
 'Michael Clarke', 'Michael_Clarke'),
 'Jamie Lee', 'Jamie_Lee'),
 'Van Dien', 'Van_Dien'),
 'Gordon-Levitt', 'Gordon_Levitt'),
'David Hyde', 'David_Hyde'),
 'Hadley Belle', 'Hadley_Belle'),
 'Sarah Jessica', 'Sarah_Jessica'),
 'Edward James', 'Edward_James'),
 'Seann William', 'Seann_William'),
 'Юлия Снигирь', 'Yuliya Snigir'),
 'Mary Elizabeth Winstead', 'Mary_Elizabeth Winstead'),
 'Craig T.', 'Craig_T.'),
 'Brandon T.', 'Brandon_T.'),
 'Philip Seymour', 'Philip_Seymour'),
 'Rhys Meyers', 'Rhys_Meyers'),
 'Matthew Gray', 'Matthew_Gray'),
 'Will.i.am', 'Will.i.am ----'),
 'Reyes, Jr.', 'Reyes,_Jr.'),
 ' van ', ' van_'),
 'Haley Jo_el', 'Haley_Jo_el'),
 'James Earl', 'James_Earl'),
 'F. Murray', 'F._Murray'),
 'Marcia Gay', 'Marcia_Gay'),
'Sarah Michelle', 'Sarah_Michelle'),
 ' the ', ' the_'),
 'Larry the_Cable Guy', 'Larry the_Cable_Guy'),
 'Pinkett Smith', 'Pinkett_Smith'),
 ' Nelly Borgeaud ', ' *Nelly Borgeaud '),
 ' Nelly', ' ---- Nelly'),
 'Raymond J. Barry', 'Raymond J._Barry'),
 'Dee Bradley', 'Dee_Bradley'),
 'Brooks Grant', 'Brooks_Grant'),
 'Lister Jr.', 'Lister_Jr.'),
 'Ólafur Darri', 'Ólafur_Darri'),
```

```
 'Jo_nathan Taylor', 'Jo_nathan_Taylor'),
 'T. Austin', 'T._Austin'),
 'Yun-fat', 'Yun_fat'),
'Duk Kim', 'Duk_Kim'),
 'Jason Scott', 'Jason_Scott'),
 'Lou Taylor', 'Lou_Taylor'),
 'Raven-Symoné', 'Raven Symoné'),
 'Robert Sean', 'Robert_Sean'),
 'Maï Anh Le', 'Maï_Anh Le'),
' Tao\' Petcharoen', '_Tao\'_Petcharoen'),
 'Mary Elizabeth', 'Mary_Elizabeth'),
 'William B.', 'William_B.'),
 'Neha Bikram Saluja', 'Neha_Bikram Saluja'),
 'Revathi ', 'Revathi ---- '),
 'Amitabh Bachchan', 'Amitabh *Bachchan'),
 'Jaya Bachchan', 'Jaya *Bachchan'),
 'Aishwarya Rai Bachchan', 'Aishwarya_Rai *Bachchan'),
 'Miyavi ', 'Miyavi ---- '),
 'C. Thomas', 'C._Thomas'),
 'Richard E.', 'Richard_E.'),
 'Jo_n Bon', 'Jo_n_Bon'),
 'Anna Deavere', 'Anna_Deavere'),
 'Jo_e Don_', 'Jo_e_Don_'),
 'Kristin Scott', 'Kristin_Scott'),
'Orianthi ', 'Orianthi ---- '),
'Downey Jr.', 'Downey_Jr.'),
'Sinbad ', 'Sinbad Sinbad '),
'Jennifer Love', 'Jennifer_Love'),
'Evan Rachel', 'Evan_Rachel'),
'Lisa Roberts', 'Lisa_Roberts'),
'Penelope Ann', 'Penelope_Ann'),
'Charles S.', 'Charles_S.'),
 'Richard T.', 'Richard_T.'),
'Lou Diamond', 'Lou_Diamond'),
```

```
'Jessica Brown', 'Jessica_Brown'),
'Jamie Campbell', 'Jamie_Campbell'),
'Xiao Shen', 'Xiao_Shen'),
'Eva Amurri', 'Eva_Amurri'),
'Julie Ann', 'Julie_Ann'),
'Taraji P.', 'Taraji_P.'),
'Vivica A.', 'Vivica_A.'),
'Gordon Liu', 'Gordon_Liu'),
'Chia-Hui', 'Chia_Hui'),
'Wen Yann', 'Wen_Yann'),
 'Ellen Hamilton Latzen', 'Ellen_Hamilton Latzen'),
 'William Lee Scott', 'William_Lee Scott'),
 'Jeremy James Kissner', 'Jeremy_James Kissner'),
 '"Weird Al" Yankovic', '"Weird_Al" Yankovic'),
 ' Scott Mortensen', ' *Scott Mortensen'),
 'Audrey Lynn Tennent', 'Audrey_Lynn Tennent'),
'Cher ', 'Cher ---- '),
'Courtney B.', 'Courtney_B.'),
'Charles S.', 'Charles_S.'),
'Sarah Jane', 'Sarah_Jane'),
 'John Benjamin', 'John_Benjamin'),
'Sydney Tamiia', 'Sydney_Tamiia'),
'Lisa Gay', 'Lisa_Gay'),
'Jo_ey Lauren', 'Jo_ey_Lauren'),
'K. C.', 'K._C.'),
'Maggie Elizabeth', 'Maggie_Elizabeth'),
 'Jo_hn Benjamin', 'Jo_hn_Benjamin'),
'Skye McCole', 'Skye_McCole'),
'Erik Per', 'Erik_Per'),
'Deborah Kara', 'Deborah_Kara'),
'Thomas Ian', 'Thomas_Ian'),
'Sarah Wayne', 'Sarah_Wayne'),
'Debnam-Carey', 'Debnam_Carey'),
'Gilbert R.', 'Gilbert_R.'),
```

```
'R. Lee', 'R._Lee'),
 'Lily Autumn Page', 'Lily_Autumn Page'),
 ' Douglas Rain ', ' Douglas Rain* '),
'Rain ', 'Rain ---- '),
'Cyndi Mayo', 'Cyndi_Mayo' ),
'Brían F.', 'Brían_F.' ),
 'Eddie Kaye', 'Eddie_Kaye' ),
 ' Mo\'Nique', ' ------ Mo\'Nique' ),
'Michael C.', ' Mo\'Michael_C.' ),
'Andrew Dice', 'Andrew_Dice' ),
'Russell G.', 'Russell_G.' ),
'Jean Michel', 'Jean_Michel' ),
'Anderson III', 'Anderson_III' ),
'Litefoot ', 'Litefoot ------ ' ),
'Keenen Ivory', 'Keenen_Ivory' ),
'Kevin J.', 'Kevin_J.' ),
'Catalina Sandino', 'Catalina_Sandino' ),
'David Alan', 'David_Alan' ),
'Tony Leung', 'Tony_Leung' ),
'Eminem ', 'Eminem ------ ' ),
'Thomas F.', 'Thomas_F.' ),
'T.I. ', 'T.I. ------ ' ),
' T.I.', ' ------ T.I.' ),
 ' Mo\'', ' _Mo\'' ),
 'Charles Malik', 'Charles_Malik' ),
'Lisa Ann', 'Lisa_Ann' ),
'Michael Jai', 'Michael_Jai' ),
'Raoul Max', 'Raoul_Max' ),
'Mary Kay', 'Mary_Kay' ),
'Jo_hn Carroll', 'Jo_hn_Carroll' ),
'Angelo Barra', 'Angelo_Barra' ),
'Jacqueline MacInnes', 'Jacqueline_MacInnes' ),
'Jamie Campbell', 'Jamie_Campbell' ),
'Rachael Leigh', 'Rachael_Leigh' ),
```

```
'Rufus Sewell', 'Rufus Sewell' ),
'Prabhas ', 'Prabhas ---- ' ),
'Tom Everett', 'Tom_Everett' ),
'Vicellous Reon', 'Vicellous_Reon' ),
'James Van', 'James_Van' ),
'Der Beek', 'Der_Beek' ),
'Rachael Leigh', 'Rachael_Leigh' ),
 'Philip Baker Hall', 'Philip_Baker Hall' ),
'Omar Benson Miller', 'Omar_Benson Miller' ),
'Jean-Claude Van Damme', 'Jean-Claude Van_Damme' ),
'Alexandra Maria', 'Alexandra_Maria' ),
'Philip Baker Hall', 'Philip_Baker Hall' ),
'Juan Carlos Hernández', 'Juan_Carlos Hernández' ),
'Daniel E. Smith', 'Daniel_E. Smith' ),
'J. T. Walsh', 'J._T. Walsh' ),
'Klaus Maria Brandauer', 'Klaus_Maria Brandauer' ),
'Fred Dalton Thompson', 'Fred_Dalton Thompson' ),
 'Jamyang Jamtsho Wangchuk', 'Jamyang_Jamtsho Wangchuk' ),
' Mako', ' ---- Mako' ),
'S. Epatha Merkerson', 'S._Epatha Merkerson' ),
'Eric Christian Olsen', 'Eric_Christian Olsen' ),
'Irma P. Hall', 'Irma_P. Hall' ),
'Eric Christian Olsen', 'Eric_Christian Olsen' ),
'Aaliyah ', 'Aaliyah --- ' ),
'Mary Beth Hurt', 'Mary_Beth Hurt' ),
 'Nicole Ari Parker', 'Nicole_Ari Parker' ),
'Sam J. Jo_nes', 'Sam_J. Jo_nes' ),
' Xzibit', ' ---- Xzibit' ),
'Clifton Collins Jr', 'Clifton Collins_Jr' ),
 'Maï_Anh Le', 'Maï_Anh Le*' ),
 'Charlotte Le Bon', 'Charlotte_Le* Bon' ),
 'Steve Le Marquand', 'Steve_Le* Marquandn' ),
 'Bernard Le Coq', 'Bernard_Le* Coq' ),
 'Isild Le Besco', 'Isild_Le* Besco' ),
```

```
 'James Le Gros', 'James_Le* Gros' ),
 'Le Sage', 'Le* Sage' ),
 'Paul Le Mat', 'Paul_Le* Mat' ),
 'Renée Le Calm', 'Renée_Le* Calm' ),
'Chao Li Chi', 'Chao_Li Chi' ),
'Jewel ', 'Jewel ---- ' ),
'Ki Hong Lee ', 'Ki_Hong Lee' ),
'June Diane Raphael', 'June_Diane Raphael' ),
'Daniel Hugh Kelly', 'Daniel_Hugh Kelly' ),
'Eva Marie Saint', 'Eva_Marie Saint' ),
' DMX', ' ---- DMX' ),
'Robin Atkin Downes', 'Robin_Atkin Downes' ),
'Robert Jo_hn Burke', 'Robert_Jo_hn Burke' ),
'Billy Dee Williams', 'Billy_Dee Williams' ),
' Sisqó', ' ---- Sisqó' ),
'Callum Keith Rennie', 'Callum_Keith Rennie' ),
'Billy Ray Cyrus', 'Billy_Ray Cyrus' ),
'Gael García Bernal', 'Gael_García Bernal' ),
'Adam G. Sevani', 'Adam_G. Sevani' ),
'Ki_Hong LeeKaya', 'Ki_Hong Lee Kaya' ),
'Mark Boone Junior', 'Mark_Boone Junior' ),
'María Conchita Alonso', 'María_Conchita Alonso' ),
'Lynn \'Red\' Williams', 'Lynn_\'Red\' Williams' ),
'L. Scott Caldwell', 'L._Scott Caldwell' ),
'Brian J. White', 'Brian_J. White' ),
'Doug E. Doug', 'Doug_E. Doug' ),
'Holly Robinson Peete', 'Holly_Robinson Peete' ),
'Christoph Maria Herbst', 'Christoph_Maria Herbst' ),
 'Anna Maria Horsford', 'Anna_Maria Horsford' ),
 'Amanda Redman', 'Amanda Redman*' ),
'Redman ', 'Redman --- ' ),
'Jo_nathan Ke Quan', 'Jo_nathan_Ke Quan' ),
'Bobb\'e J. Thompson', 'Bobb\'e_J. Thompson' ),
' Norfolk', ' ---- Norfolk' ),
```

```
'RZA ', 'RZA --- ' ),
'Benjamin Jay Davis', 'Benjamin_Jay Davis' ),
'Adam David Thompson', 'Adam_David Thompson' ),
'Keegan Connor Tracy', 'Keegan_Connor Tracy' ),
'Gerald R. Molen', 'Gerald_R. Molen' ),
'Michele Lamar Richards', 'Michele_Lamar Richards' ),
'Michael Kenneth Williams', 'Michael_Kenneth Williams' ),
 'Luis Fernando Peña', 'Luis_Fernando Peña' ),
'Silambarasan ', 'Silambarasan ----- ' ),
'Santhanam ', 'Santhanam -- ' ),
' Brahmanandam', ' ---- Brahmanandam' ),
'Matthew A. Brown', 'Matthew_A. Brown' ),
'Anna Claire Sneed', 'Anna_Claire Sneed' ),
' Cherki', ' ---- Cherki' ),
'Kimberly J. Brown', 'Kimberly_J. Brown' ),
'Brendan "Doogie" Milewski', 'Brendan_"Doogie" Milewski' ),
 'Jo_hn Bennett Perry', 'Jo_hn_Bennett Perry' ),
 'Christopher Walken', 'Christopher* Walken' ),
 'Dyllan Christopher', 'Dyllan Christopher*' ),
 'Christopher Lloyd', 'Christopher* Lloyd' ),
 'Christopher Meloni', 'Christopher* Meloni' ),
 'Mads Sjøgård Pettersen', 'Mads_Sjøgård Pettersen' ),
 'Tyler James Williams', 'Tyler_James Williams' ),
 'Patrick St. Esprit', 'Patrick_St. Esprit' ),
 'Christopher Evan Welch', 'Christopher*_Evan Welch' ),
 'Henry Ian Cusick', 'Henry_Ian Cusick' ),
 'Jesse Lee Soffer', 'Jesse_Lee Soffer' ),
 'Christopher Shand', 'Christopher* Shand' ),
 'Christopher Nicholas Smith', 'Christopher*_Nicholas Smith' ),
 'Christopher Jo_rdan Wallace', 'Christopher*_Jo_rdan Wallace' ),
 'Jo_hn P. Ryan', 'Jo_hn_P. Ryan' ),
 'Connor Christopher Levins', 'Connor_Christopher* Levins' ),
 'Christopher Beiring', 'Christopher* Beiring' ),
 'Brian Keith Allen', 'Brian_Keith Allen' ),
```

```
 'Christopher Darga', 'Christopher* Darga' ),
 'Harold House Moore', 'Harold_House Moore' ),
 'Christopher Toler', 'Christopher* Toler' ),
 'Christopher Robin Miller', 'Christopher*_Robin Miller' ),
 'Natalie Stephany Aguilar', 'Natalie_Stephany Aguilar' ),
 'Dragon ', 'Dragon ---- ' ),
'Don_ald L. Brooks', 'Don_ald_L. Brooks' ),
'Ice-T ', 'Ice-T ---- ' ),
'Amidou ', 'Amidou ---- ' ),
 'Marion Cotillard', '*Marion* Cotillard' ),
 'Mario_Van Peebles', '*Mario*_Van Peebles' ),
 'Marion Darlington', '*Marion* Darlington' ),
 'Marion Bailey', '*Marion* Bailey' ),
 'Marion O\'Dwyer', '*Marion* O\'Dwyer' ),
 'Marion Lambert', '*Marion* Lambert' ),
 'Melvin Van Peebles', 'Melvin_Van Peebles' ),
'Mario ', 'Mario ---- ' ),
 ' Mario', ' ---- Mario' ),
'Steve-O ', 'Steve-O ---- ' ),
'GQ ', 'GQ ---- ' ),
 ' GQ', ' ---- GQ' ),
'Luenell ', 'Luenell --- ' ),
 'Yedidia ', 'Yedidia --- ' ),
 'Machado ', 'Machado --- ' ),
'Aliya Ba Sen', 'Aliya_Ba Sen' ),
'Ne-Yo ', 'NeYo --- '),
'Astro ', 'Astro --- ' ),
 'Gabrielle Lopes Benites', 'Gabrielle_Lopes Benites' ),
'Common ', 'Common --- ' ),
'Arletty ', 'Arletty --- ' ),
'Rekha ', 'Rekha --- ' ),
 'Wilbur \'Hi-Fi\' White Leon Isaac Kennedy', 'Wilbur_\'Hi-Fi\'_White
Leon_Isaac_Kennedy' ),
'Max Elliott Slade', 'Max_Elliott Slade' ),
'J. Evan Bonifant', 'J._Evan Bonifant' ),
```

```
'Caroline Junko King', 'Caroline_Junko King' ),
'Floyd Red Crow Westerman', 'Floyd_Red Crow_Westerman' ),
'Rodney A. Grant', 'Rodney_A. Grant' ),
'Shah Rukh Khan', 'Shah_Rukh Khan' ),
'Douglas M. Griffin', 'Douglas_M. Griffin' ),
'Putu Dinda Pratika', 'Putu_Dinda Pratika' ),
'Puti Sri Candra Dewi', 'Puti_Sri Candra_Dewi' ),
'Ni Made Megahadi Pratiwi', 'Ni_Made Megahadi_Pratiwi' ),
'George H. W. Bush', 'George H._W._Bush' ),
'George W. Bush', 'George_W. Bush' ),
'Stig Frode Henriksen', 'Stig_Frode Henriksen' ),
'Jeppe Beck Laursen', 'Jeppe_Beck Laursen' ),
 'Patrick J. Adams', 'Patrick_J. Adams' ),
'Jo_hnny A. Sanchez', 'Jo_hnny_A. Sanchez' ),
 'Rufus Sewell', 'Rufus* Sewell' ),
 'Rufus Graham', 'Rufus* Graham' ),
 'Rufus ', 'Rufus ---- ' ),
 'Kevin Peter Hall', 'Kevin_Peter* Hall' ),
 'Corlandos Scott ', 'Corlandos Scott*' ),
 'Chad Michael Murray', 'Chad_Michael* Murray' ),
 'Mary Lynn Rajskub', 'Mary_Lynn* Rajskub' ),
'Jo_el David Moore', 'Jo_el_David* Moore' ),
'Elizabeth of Toro', 'Elizabeth of_Toro*' ),
'Jennifer Schwalbach Smith', 'Jennifer_Schwalbach* Smith' ),
 'Brian Van Holt', 'Brian_Van* Holt' ),
 'Steven R. McQueen', 'Steven_R. McQueen' ),
 'Jill Marie Jo_nes', 'Jill_Marie* Jo_nes' ),
 'Omar J. Dorsey', 'Omar_J. Dorsey' ),
 'Tony Lo Bianco', 'Tony_Lo* Bianco' ),
 'Anthony Michael Hall', 'Anthony_Michael* Hall' ),
 'Frank C. Turner', 'Frank_C. Turner' ),
'Chen Hu Jun', 'Chen_Hu* Jun' ),
'Mary J. Blige', 'Mary_J. Blige' ),
'Christie Lynn Smith', 'Christie_Lynn* Smith' ),
```

```
'Eddie Cheung Siu-Fai', 'Eddie_Cheung* Siu-Fai' ),
'Sean Patrick Thomas', 'Sean_Patrick* Thomas' ),
 'Eve ', 'Eve --- ' ),
'van den ', 'van_den_') AS castNuevo
FROM cast;
-- Tabla Final
DROP TABLE IF EXISTS cast_final;
CREATE TABLE cast_final AS
SELECT id,
SUBSTRING_INDEX(castNuevo, ' ', 2) AS nombres
FROM cast_tmp
UNION
SELECT id,
SUBSTRING_INDEX(SUBSTRING_INDEX(castNuevo, ' ', 4), ' ', -2) AS nombres
FROM cast_tmp
UNION
SELECT id,
SUBSTRING_INDEX(SUBSTRING_INDEX(castNuevo, ' ', 6), ' ', -2) AS nombres
FROM cast_tmp
UNION
SELECT id,
SUBSTRING_INDEX(SUBSTRING_INDEX(castNuevo, ' ', 8), ' ', -2) AS nombres
FROM cast_tmp
UNION
SELECT id,
SUBSTRING_INDEX(SUBSTRING_INDEX(castNuevo, ' ', 10), ' ', -2) AS nombres
FROM cast_tmp
UNION
SELECT id,
SUBSTRING_INDEX(SUBSTRING_INDEX(castNuevo, ' ', 12), ' ', -2) AS nombres
FROM cast_tmp
UNION
SELECT id,
SUBSTRING_INDEX(SUBSTRING_INDEX(castNuevo, ' ', 14), ' ', -2) AS nombres
```

```
FROM cast_tmp
UNION
SELECT id,
SUBSTRING_INDEX(SUBSTRING_INDEX(castNuevo, ' ', 16), ' ', -2) AS nombres
FROM cast_tmp
UNION
SELECT id,
SUBSTRING_INDEX(SUBSTRING_INDEX(castNuevo, ' ', 18), ' ', -2) AS nombres
FROM cast_tmp
UNION
SELECT id,
SUBSTRING_INDEX(SUBSTRING_INDEX(castNuevo, ' ', 20), ' ', -2) AS nombres
FROM cast_tmp
ORDER BY id;
```

2.1.9 DDL del Diagrama Físico

A continuación, se muestra el Script que contiene el DDL del Diagrama Físico:

```
DROP DATABASE IF EXISTS proyecto_integrador_DEFINITIVO;
CREATE DATABASE proyecto_integrador_DEFINITIVO CHARACTER SET utf8mb4;
USE proyecto_integrador_DEFINITIVO;
DROP TABLE IF EXISTS director;
CREATE TABLE director (
id_director int(100) NOT NULL,
name_director varchar(100) NOT NULL,
PRIMARY KEY (id_director)
);
DROP TABLE IF EXISTS movies;
CREATE TABLE movies (
idMovies varchar(16) NOT NULL,
```

```
budget int(100) NOT NULL,

homepague varchar(300),

original_title varchar(100), -- cambio NOT NULL a puede tener NULL

overview varchar(1000),

popularity long,

release_date varchar(10) , -- cambio de date a varchar(10)

revenue long, -- cambio de int a long

runtime varchar(10), -- cambio de int(5) a double / cambio de double a varchar(10)

status varchar(15) ,

tagline varchar(400), -- cambio de 200 a 400

title varchar (200),

vote_average double , -- cambio de 2,2 a 5,5 / cambio de 5,5 a double

vote_count int (15),

original_lenguague varchar (5),

PRIMARY KEY (idMovies)

);

-- -----------------------------------------------------------

DROP TABLE IF EXISTS `cast`;

CREATE TABLE cast (

id_cast INT(100) NOT NULL, -- creamos id para cast

name_cast VARCHAR(200) NOT NULL,

PRIMARY KEY (id_cast)

);

DROP TABLE IF EXISTS spoken_languages;

CREATE TABLE spoken_languages(

id_spoken_languages VARCHAR(4) NOT NULL,

name VARCHAR(50) NOT NULL, -- cambio de varchar(15) a varchar(50)
```

```
PRIMARY KEY (id_spoken_languages)

);

DROP TABLE IF EXISTS crew;

CREATE TABLE crew (

id_crew VARCHAR(60) NOT NULL,

gender VARCHAR(60) NOT NULL,

name VARCHAR(200), -- cambio de varchar(30) a varchar(50) / cambio de varchar(50)
a varchar(200)

PRIMARY KEY (id_crew)

);

DROP TABLE IF EXISTS credit;

CREATE TABLE credit (

credit VARCHAR(200) NOT NULL, -- cambio de varchar(50) a varchar(200)

job VARCHAR(200), -- cambio de varchar(30) a varchar(50) / cambio de varchar(50) a
varchar(200)

department VARCHAR(200), -- cambio de varchar(15) a varchar(50) / cambio de
varchar(50) a varchar(200)

id_crew VARCHAR(60) NOT NULL,

PRIMARY KEY (credit),

FOREIGN KEY (id_crew) REFERENCES crew (id_crew)

);

DROP TABLE IF EXISTS genres;

CREATE TABLE genres (

id_genres INT(100) NOT NULL, -- creamos id para genres

genres VARCHAR(200) NOT NULL, -- cambio de varchar(20) a varchar(200)

PRIMARY KEY (id_genres)

);

DROP TABLE IF EXISTS keywords;

CREATE TABLE keywords (
```

```sql
id_keywords INT(100) NOT NULL, -- creamos id para keywords

keywords VARCHAR(300) NOT NULL, -- cambio de varchar(20) a varchar(300)

PRIMARY KEY (id_keywords) -- intercambio clave primaria por id_keywords

);

DROP TABLE IF EXISTS production_companies;

CREATE TABLE production_companies (

id_production_companies int(15) NOT NULL,

`name` varchar(300) NOT NULL, -- cambio de varchar(25) a varchar(300)

PRIMARY KEY (id_production_companies)

);

DROP TABLE IF EXISTS production_countries;

CREATE TABLE production_countries (

id_production_countries varchar(5) NOT NULL, -- cambio de int(5) a varchar(5)

`name` varchar(30) NOT NULL,

PRIMARY KEY (id_production_countries)

);

-- -----------------------------------------------------------

DROP TABLE IF EXISTS movies_production_countries;

CREATE TABLE movies_production_countries (

idMovies varchar(16) NOT NULL,

id_production_countries varchar(5) NOT NULL,

PRIMARY KEY (idMovies,id_production_countries),

FOREIGN KEY (idMovies) REFERENCES movies (idMovies),

FOREIGN KEY (id_production_countries) REFERENCES production_countries
(id_production_countries)

);

DROP TABLE IF EXISTS movies_production_companies;

CREATE TABLE movies_production_companies(
```

```
idMovies varchar(16) NOT NULL,

id_production_companies int(15) NOT NULL,

PRIMARY KEY (idMovies,id_production_companies),

FOREIGN KEY (idMovies) REFERENCES movies (idMovies),

FOREIGN KEY (id_production_companies) REFERENCES production_companies
(id_production_companies)

);

DROP TABLE IF EXISTS movies_crew;

CREATE TABLE movies_crew (

credit VARCHAR(200) NOT NULL,

idMovies VARCHAR(16) NOT NULL,

PRIMARY KEY (credit, idMovies),

FOREIGN KEY (credit) REFERENCES credit (credit),

FOREIGN KEY (idMovies) REFERENCES movies (idMovies)

);

DROP TABLE IF EXISTS movies_genres;

CREATE TABLE movies_genres (

idMovies VARCHAR(16) NOT NULL,

id_genres INT(100) NOT NULL,

PRIMARY KEY (idMovies, id_genres),

FOREIGN KEY (idMovies) REFERENCES movies (idMovies),

FOREIGN KEY (id_genres) REFERENCES genres (id_genres)

);

DROP TABLE IF EXISTS movies_keywords;

CREATE TABLE movies_keywords (

idMovies VARCHAR(16) NOT NULL,

id_keywords INT(100) NOT NULL, -- intercambio clave primaria por id_keywords

PRIMARY KEY (idMovies, id_keywords),
```

```
FOREIGN KEY (idMovies) REFERENCES movies (idMovies),

FOREIGN KEY (id_keywords) REFERENCES keywords (id_keywords)

);

DROP TABLE IF EXISTS movies_cast;

CREATE TABLE movies_cast (

idMovies VARCHAR(16) NOT NULL,

id_cast INT(100) NOT NULL,

PRIMARY KEY (idMovies, id_cast),

FOREIGN KEY (id_cast) REFERENCES cast (id_cast),

FOREIGN KEY (idMovies) REFERENCES movies (idMovies)

);

DROP TABLE IF EXISTS movies_spoken_languages;

CREATE TABLE movies_spoken_languages (

idMovies VARCHAR(16) NOT NULL,

id_spoken_languages VARCHAR(4) NOT NULL,

PRIMARY KEY (idMovies, id_spoken_languages),

FOREIGN KEY (id_spoken_languages) REFERENCES
spoken_languages(id_spoken_languages),

FOREIGN KEY (idMovies) REFERENCES movies (idMovies)

);

DROP TABLE IF EXISTS director_movies;

CREATE TABLE director_movies (

idMovies VARCHAR(16) NOT NULL,

id_director int(100) NOT NULL,

PRIMARY KEY (idMovies, id_director),

FOREIGN KEY (id_director) REFERENCES director (id_director),

FOREIGN KEY (idMovies) REFERENCES movies (idMovies)

);
```

2.1.10 Población de las tablas

En este proceso mediante consultas Insert Into y Select se procedió a poblar las tablas que conforman nuestro modelo físico.

```
use proyecto_integrador_definitivo;
-- Poblacion de tabla movies
INSERT INTO movies (
idMovies,
budget,
homepague,
original_title,
overview,
popularity,
release_date,
revenue,
runtime,
status,
tagline,
title,
vote_average,
vote_count,
original_lenguague)
SELECT id,
budget,
homepage,
original_title,
overview,
```

```
popularity,
release_date,
revenue,
runtime,
status,
tagline,
title,
vote_average,
vote_count,
original_language
FROM proyecto.original_title_movies_tmp;
-- Poblacion de Director
use proyecto;
DROP TABLE IF EXISTS director_TMP2;
CREATE TABLE director_TMP2 AS
SELECT DISTINCT directorNuevo
FROM proyecto.director_tmp;
ALTER TABLE director_TMP2
ADD COLUMN id INT AUTO_INCREMENT PRIMARY KEY;
INSERT INTO director(
id_director,
name_director
)
SELECT id,
directorNuevo
FROM proyecto.director_tmp2;
-- Poblacion de Keywords
```

```sql
DROP TABLE IF EXISTS keywords_TMP2;

CREATE TABLE keywords_TMP2 AS

SELECT DISTINCT keywordsNuevo

FROM proyecto.keywords_tmp;

ALTER TABLE keywords_TMP2

ADD COLUMN id INT AUTO_INCREMENT PRIMARY KEY;

INSERT INTO keywords(

id_keywords,

keywords

)

SELECT id,

keywordsNuevo

FROM proyecto.keywords_tmp2;

-- Poblacion de production_companies

DROP TABLE IF EXISTS production_companies_TMP3;

CREATE TABLE production_companies_TMP3 AS

SELECT DISTINCT ProductionCompanyName, ProductionCompanyId

FROM proyecto.production_companies_tmp2;

INSERT INTO production_companies(

id_production_companies,

name

)

SELECT ProductionCompanyId,

ProductionCompanyName

FROM proyecto.production_companies_TMP3;

-- Poblacion de production_countries

DROP TABLE IF EXISTS production_countries_TMP3;
```

```sql
CREATE TABLE production_countries_TMP3 AS
SELECT DISTINCT ProductionCountriesId, ProductionCountriesNames
FROM proyecto.production_countries_TMP2;
INSERT INTO production_countries(
id_production_countries,
name
)
SELECT ProductionCountriesId,
ProductionCountriesNames
FROM proyecto.production_countries_TMP3;
-- Poblacion de spoken_languages
DROP TABLE IF EXISTS spoken_languages_tmpx2;
CREATE TABLE spoken_languages_tmpx2 AS
SELECT DISTINCT iso_639_1, name
FROM proyecto.spoken_languages_tmpx;
INSERT INTO spoken_languages(
id_spoken_languages,
name
)
SELECT iso_639_1,
name
FROM proyecto.spoken_languages_tmpx2;
-- Poblacion de genres
DROP TABLE IF EXISTS movies_genres2;
CREATE TABLE movies_genres2 AS
SELECT DISTINCT genre
FROM proyecto.movies_genres;
```

```sql
ALTER TABLE movies_genres2
ADD COLUMN id INT AUTO_INCREMENT PRIMARY KEY;
INSERT INTO genres(
genres,
id_genres
)
SELECT genre,
id
FROM proyecto.movies_genres2;
-- Poblacion de cast
DROP TABLE IF EXISTS cast_final2;
CREATE TABLE cast_final2 AS
SELECT DISTINCT nombres
FROM proyecto.cast_final;
ALTER TABLE cast_final2
ADD COLUMN id INT AUTO_INCREMENT PRIMARY KEY;
INSERT INTO `cast`(
id_cast,
name_cast
)
SELECT id,
nombres
FROM proyecto.cast_final2;
-- --------------------tablas intermedias--------------------
-- Poblacion de movies_production_companies
INSERT INTO movies_production_companies (
idMovies,
```

```
id_production_companies
)
SELECT id,
ProductionCompanyId
FROM proyecto.production_companies_tmp2;
-- Poblacion de movies_production_countries
INSERT INTO movies_production_countries (
idMovies,
id_production_countries
)
SELECT id,
ProductionCountriesId
FROM proyecto.production_countries_tmp2;
-- Poblacion de movies_spoken_languages
INSERT INTO movies_spoken_languages (
idMovies,
id_spoken_languages
)
SELECT id_movie,
iso_639_1
FROM proyecto.spoken_languages_tmpx;
-- Poblacion de movies_cast
DROP TABLE IF EXISTS movies_cast_join;
CREATE TABLE movies_cast_join AS
SELECT cast_final2.nombres, cast_final2.id AS id_actor, cast_final.id AS id_movies
FROM cast_final2
INNER JOIN cast_final
```

```
ON cast_final2.nombres = cast_final.nombres;

INSERT INTO movies_cast (

idMovies,

id_cast

)

SELECT id_movies,

id_actor

FROM proyecto.movies_cast_join;

-- Poblacion director_movies

DROP TABLE IF EXISTS director_movies_join;

CREATE TABLE director_movies_join AS

SELECT director_tmp2.directorNuevo AS nombreDirector, director_tmp2.id AS
idDirector, director_tmp.id AS idMovies

FROM director_tmp2

INNER JOIN director_tmp

ON director_tmp2.directorNuevo = director_tmp.directorNuevo;

INSERT INTO director_movies (

idMovies,

id_director

)

SELECT idMovies,

idDirector

FROM proyecto.director_movies_join;

-- Poblacion de movies_keywords

DROP TABLE IF EXISTS movies_keywords_join;

CREATE TABLE movies_keywords_join AS

SELECT keywords_tmp.id AS idMovies, keywords_tmp.keywordsNuevo , keywords_tmp2.id
AS idKeywords
```

```
FROM keywords_tmp

INNER JOIN keywords_tmp2

ON keywords_tmp.keywordsNuevo = keywords_tmp2.keywordsNuevo;

INSERT INTO movies_keywords (

idMovies,

id_keywords

)

SELECT idMovies,

idKeywords

FROM proyecto.movies_keywords_join;

-- Poblacion de movies_genres

DROP TABLE IF EXISTS movies_genres_join;

CREATE TABLE movies_genres_join AS

SELECT movies_genres2.genre, movies_genres2.id AS idGender, movies_genres.id AS
idMovies

FROM movies_genres2

INNER JOIN movies_genres

ON movies_genres2.genre = movies_genres.genre;

INSERT INTO movies_genres (

idMovies,

id_genres

)

SELECT idMovies,

idGender

FROM proyecto.movies_genres_join;

-- ---------------------------------------Correccion Crew-------------------------
--------------

-- Poblacion de Crew
```

```sql
DROP TABLE IF EXISTS crew_tmp4;

CREATE TABLE crew_tmp4 AS

SELECT DISTINCT crew_tmp3.id, crew_tmp3.gender, crew_tmp3.name

FROM crew_tmp3;

DELETE FROM crew_tmp4

WHERE id = 30711;

INSERT INTO crew(

id_crew,

gender,

name

)

SELECT id,

gender,

name

FROM proyecto.crew_tmp4;

-- Poblacion de Credit

DROP TABLE IF EXISTS crew_tmp5;

CREATE TABLE crew_tmp5 AS

SELECT DISTINCT crew_tmp3.credit_id, crew_tmp3.job, crew_tmp3.department,
crew_tmp3.id

FROM crew_tmp3;

DELETE FROM crew_tmp5

WHERE id = 30711;

INSERT INTO credit(

credit,

job,

department,

id_crew
```

```
)
SELECT credit_id,
job,
department,
id
FROM proyecto.crew_tmp5;
-- Poblacion de movies_crew
DROP TABLE IF EXISTS crew_tmp6;
CREATE TABLE crew_tmp6 AS
SELECT *
FROM crew_tmp3;
DELETE FROM crew_tmp6
WHERE id = 30711;
INSERT INTO movies_crew (
credit,
idMovies
)
SELECT credit_id,
id_movie
FROM proyecto.crew_tmp6;
```

2.1.11 Diagrama final fisico de la base de datos generado por 2 DBMS diferentes (DataGrip y DBeaver)

El diagrama físico sufrió cambios en algunos aspectos, incluyendo el tipo de datos y el rango de algunas columnas, debido a los problemas de truncado que surgieron al momento de poblar las tablas. Además, se crearon nuevas columnas como claves primarias en ciertas tablas, ya que una sola columna con datos repetidos no puede ser clave primaria:

Ilustración 14.
Diagrama generado por DBMS DataGrip

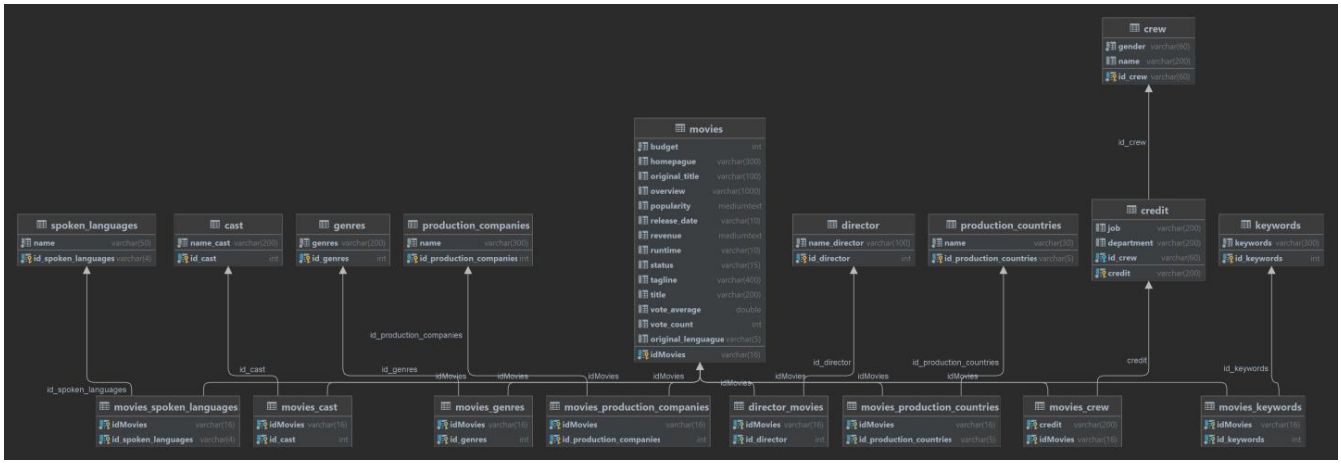
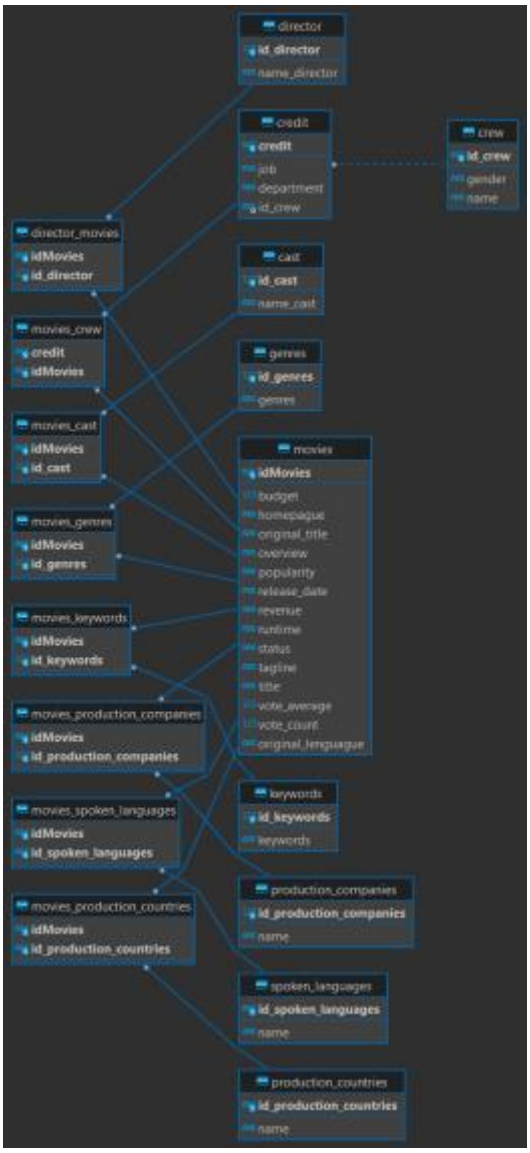


Ilustración 15.
Diagrama generado por DBMS DBeaver



2.1.12 Explotación de la data

Finalmente, tras completar con éxito la población de todas las tablas, llevamos a cabo varias consultas para evaluar el estado actual de nuestra base de datos:

```
-- Examinar el contenido de la tabla de Películas.
SELECT *
FROM movies;


-- Consultar las películas y su fecha de estreno solamente de aquellas que
tienen una duración superior a 100 minutos.
SELECT original_title, release_date
FROM movies
WHERE runtime > 100.0;


-- Mostrar las peliculas con popularidad mayor a 150
SELECT original_title, popularity
FROM movies
WHERE popularity > 150;


-- Muestra el nombre del director que dirigio la pelicula con la mayor
duracion
SELECT movies.original_title AS Peliculas, movies.runtime AS Duracion,
director.name_director AS Director
FROM movies
INNER JOIN director_movies ON director_movies.idMovies = movies.idMovies
INNER JOIN director ON director.id_director = director_movies.id_director
WHERE (runtime = (SELECT MAX(CAST(runtime AS DOUBLE)) as runtimeMax
FROM movies));


-- Muestra todos los directores que grabron peliculas en frances
SELECT director.name_director AS Director
FROM movies
INNER JOIN director_movies ON director_movies.idMovies = movies.idMovies
INNER JOIN director ON director.id_director = director_movies.id_director
WHERE (movies.original_lenguague = 'fr');


-- Muestra los generos y el nombre de la pelicula con menor popularity
SELECT movies.original_title AS Peliculas, genres.genres AS Generos
FROM movies
INNER JOIN movies_genres ON movies_genres.idMovies = movies.idMovies
INNER JOIN genres ON genres.id_genres = movies_genres.id_genres
WHERE (popularity = (SELECT MIN(CAST(popularity AS DOUBLE)) as
```

```
popularityMin
FROM movies));


-- Muestra las peliculas en las que participo Antonio Banderas
SELECT movies.original_title AS Peliculas
FROM movies
INNER JOIN movies_cast ON movies_cast.idMovies = movies.idMovies
INNER JOIN cast ON cast.id_cast = movies_cast.id_cast
WHERE (cast.name_cast = 'Antonio Banderas');


-- Muestra todos los idiomas que se hablan en la pelicula X-Men: Days of
Future Past
SELECT spoken_languages.name AS LenguajesHablados
FROM movies
INNER JOIN movies_spoken_languages ON movies_spoken_languages.idMovies =
movies.idMovies
INNER JOIN spoken_languages ON spoken_languages.id_spoken_languages =
movies_spoken_languages.id_spoken_languages
WHERE (movies.original_title = 'X-Men: Days of Future Past');


-- Muestra todas las companias que estuvieron en la produccion de la
pelicula Harry Potter and the Prisoner of Azkaban
SELECT production_companies.name AS Companias
FROM movies
INNER JOIN movies_production_companies ON
movies_production_companies.idMovies = movies.idMovies
INNER JOIN production_companies ON
production_companies.id_production_companies =
movies_production_companies.id_production_companies
WHERE (movies.original_title = 'Harry Potter and the Prisoner of
Azkaban');


-- Muestra todas las peluculas de genero Horror y presupuesto mayor a
300000
SELECT movies.original_title AS Peliculas
FROM movies
INNER JOIN movies_genres ON movies_genres.idMovies = movies.idMovies
INNER JOIN genres ON genres.id_genres = movies_genres.id_genres
WHERE (genres.genres = 'Horror' AND movies.budget = 300000);
```

3. Conclusión

En resumen, al seguir de manera sistemática el proceso del proyecto integrador de saberes, que incluyó tareas como diseñar el diagrama de la base de datos, limpiar caracteres, extraer datos de Json mediante procedimientos y completar las diferentes tablas, logramos aplicar, fortalecer y adquirir conocimientos en el ámbito de las bases de datos.