

**UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA**

*La Universidad Católica de Loja*

**INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN**

**Materia:**

Fundamentos de Base de Datos

**Proyecto Final:**

Ciclo de vida de bases de datos relacionales normalizada

**Estudiante:**

Francisco Gonzáles | fxgonzalez5@utpl.edu.ec

**Link del Proyecto en Github:**

https://github.com/fxgonzalez5/ProyectodeBD.git

**Profesor:**

Nelson Piedra | https://investigacion.utpl.edu.ec/nopiedra

**Fecha:**

Loja, 10 de febrero del 2022

Octubre 2021 – Febrero 2022

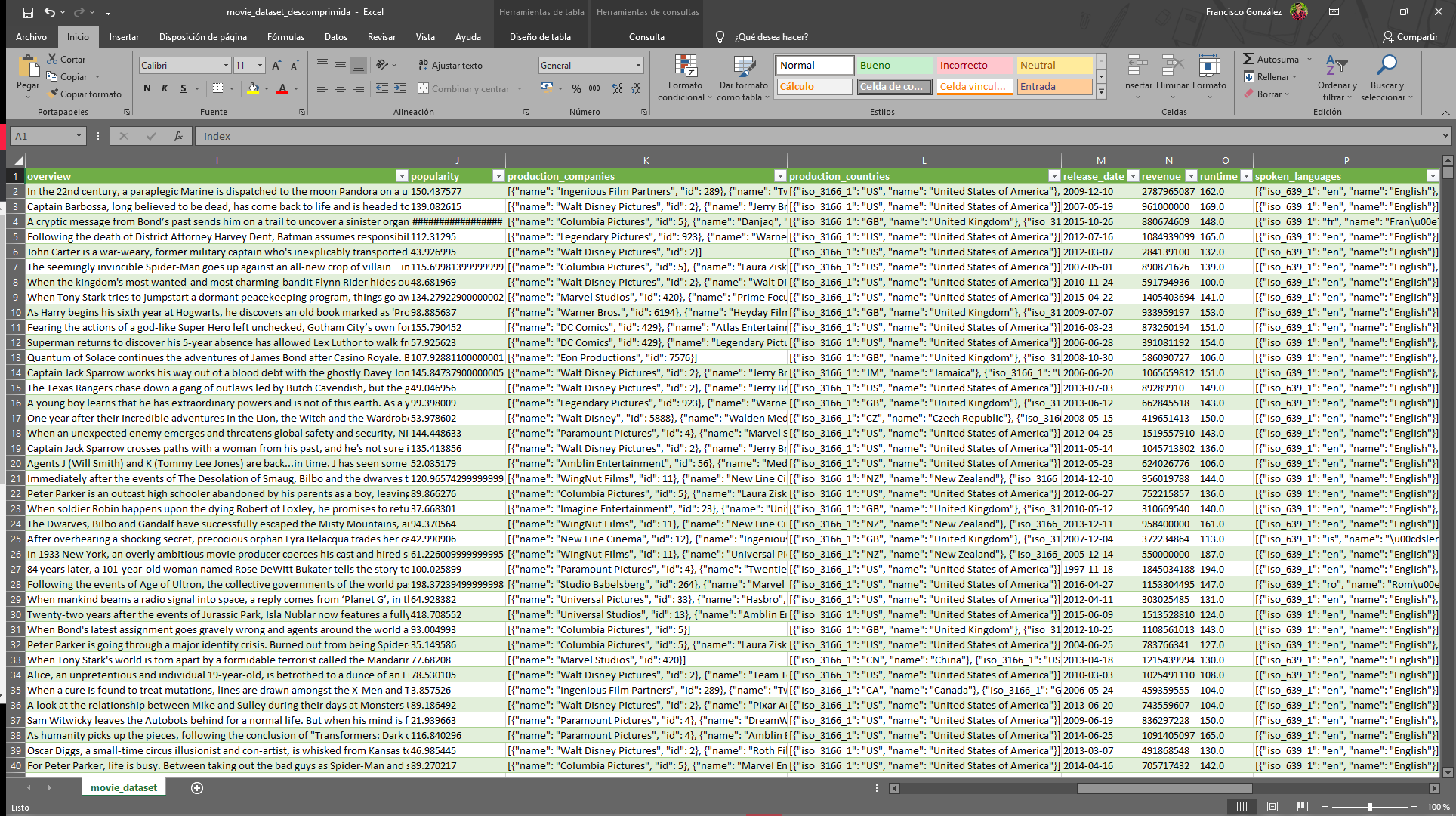
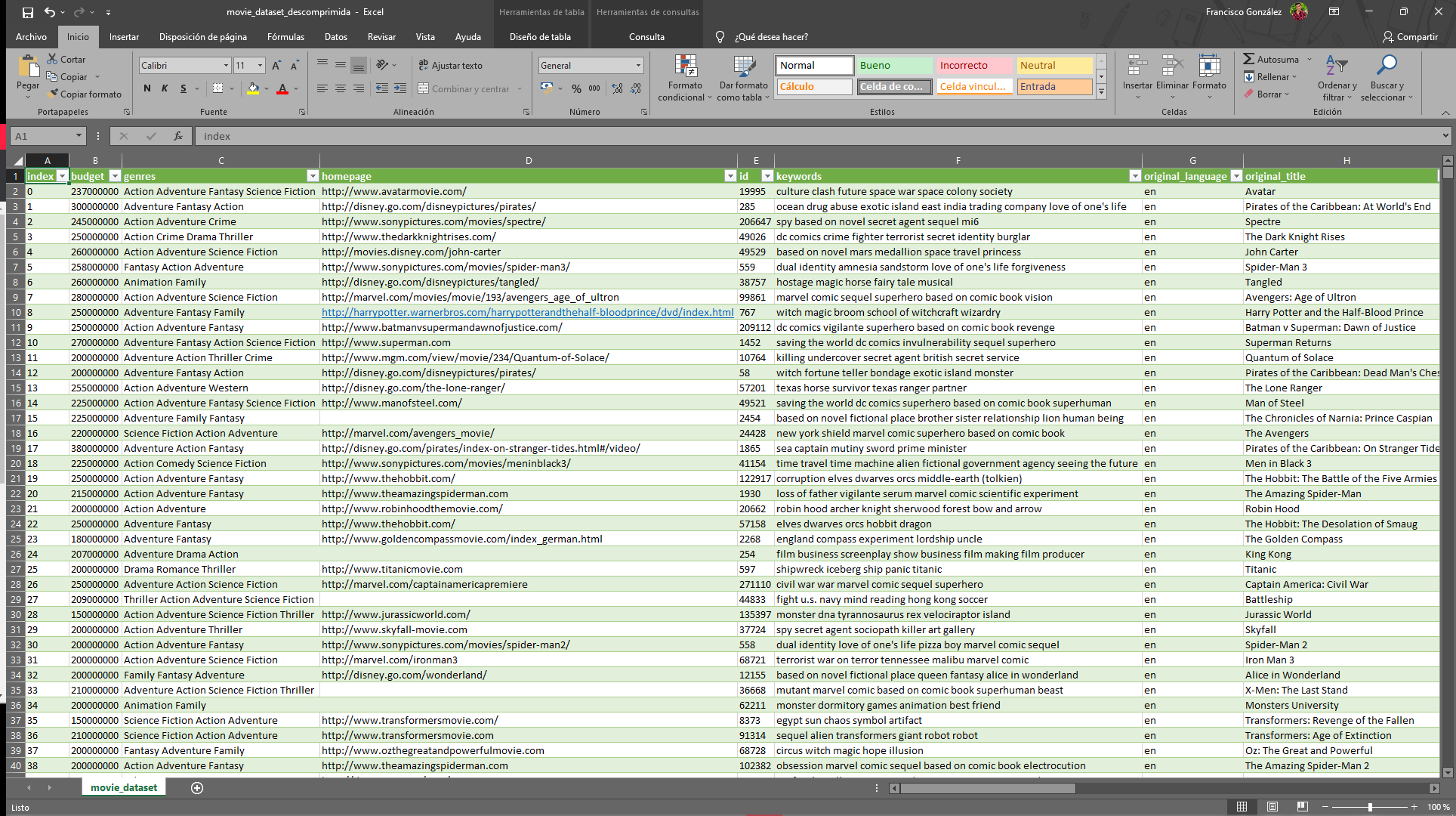
1. **Introducción**

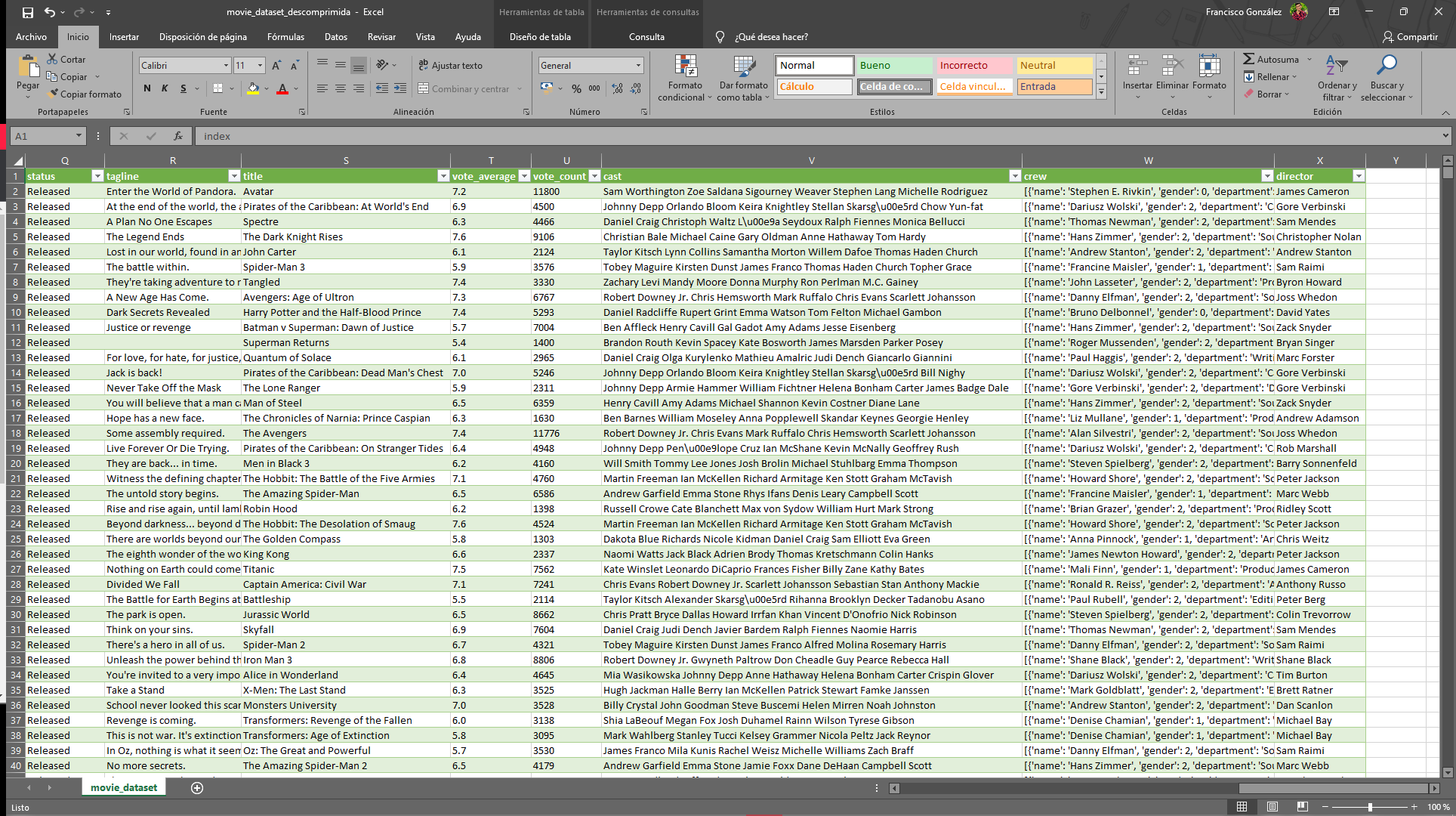
En un archivo CSV se pueden almacenar cualquier cantidad de datos, los cuales pueden ser extraídos en un documento de Excel para que sea más legible. Pero el objetivo más halla de comprender los data almacenada en el ya mencionado, es lograr transformarlo en un esquema entidad relación a partir de aplicar normalización y dependencias funcionales para el manejo exhaustivo de la data en una base de datos. Esto permite tener un mejor flujo de la manipulación de los datos que ya se tienen dentro del CSV y de los que en un futuro se requieran almacenar.

## Desarrollo

En el siguiente proyecto, disponemos de un archivo CSV que contiene datos recopilados sobre muchas películas. Analizando la data proporcionada se pudo identificar que algunos datos no estaban almacenados de manera correcta, porque al momento de convertir el CSV a una tabla en Excel, muchos caracteres no se decodificaban como la “ñ”, “í”, entre otras; también me encontré con la peculiaridad de que contábamos con campos de tipo JSON y uno de ellos como lo es el campo “crew”, la estructura no estaba de forma correcta.

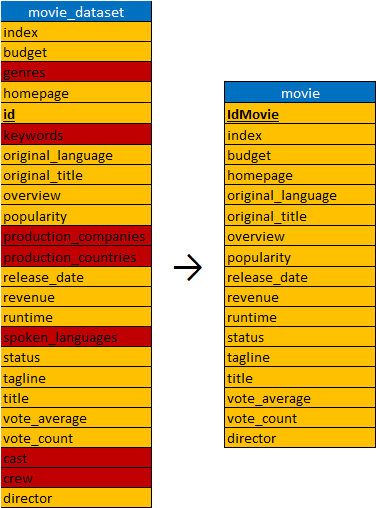
Lo primero que se realizó después del análisis de la data, es la normalización de la tabla que se extrajo en un archivo de Excel, donde la primera fila especifica los atributos o las columnas que tiene la tabla y conjuntamente a la normalización se identifican las dependencias funcionales que se requiere en cada forma normal.





Teniendo la tabla general, lo primero que se debe identificar es la claves candidatas y posteriormente determinar la clave primaria de la tabla. Seguidamente se analiza si cumple la primera forma normal (campos atómicos), para lo cual a primera vista visualizamos que “production\_campanies”, “production\_countries”, “spoken\_languages” y “crew” son campos multivaluados y a la vez compuestos por ser objetos de tipo JSON. Mientras que “genres”, “keywords” y “cast” simplemente son campos multivaluados; por tanto se la aparta de la tabla general y esto nos da como resultado la tabla final “movie” que ya cumpliría la primera forma normal.

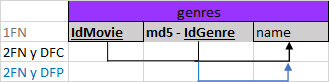
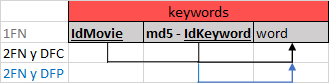
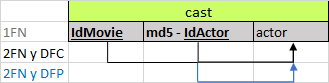


Teniendo los campos no atómicos identificados, se procede a convertirlos en tablas relacionales para que cumplan la primera forma normal; también se debe identificar las claves candidatas y se determina la o las claves primarias de cada tabla.

En el caso de los campos que simplemente eran multivaluados, se utiliza la función md5 que nos ofrece MySQL para poder generar un código uno para una cadena de caracteres, por el motivo que estos campos son considerados fuertes ya que generan una relación con cardinalidad de muchos a muchos (M:N) con la tabla “movie”.

Como siguiente paso se analiza si cumplen la segunda forma normal (exista dependencia funcional completa); si existen claves primarias compuestas como es el caso, se debe aplicar dependencia funcional parcial para lograr que cumplan la segunda forma normal y obtener las tablas finales; tener muy en cuenta que las claves primarias compuestas determinan las tablas de relación.

Como último paso de la normalización, ya que tenemos las tablas en segunda forma normal, analizamos si podría aplicarse la tercera forma normal (exista dependencia funcional transitiva); identifique la tabla “crew” donde podría poner en practica la tercera forma normal dando como resultado una tabla final más, que estaría relacionada a la tabla que surgió con la segunda forma normal mediante una clave foránea que está determinada por la clave primaria de la tabla resultante en la tercera forma normal.

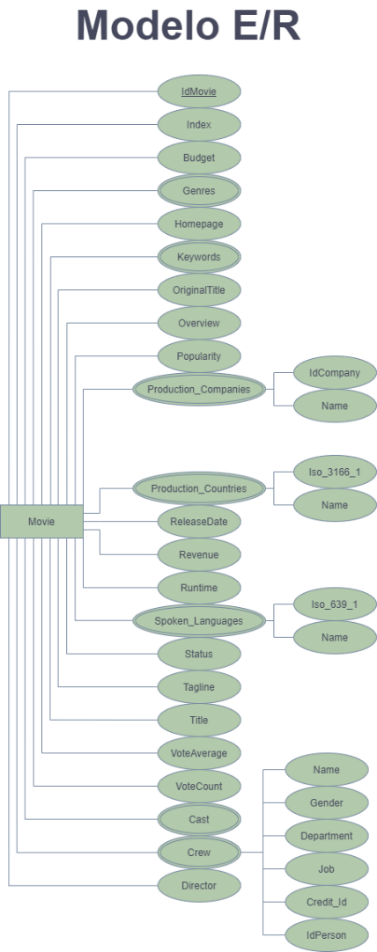


Luego se debe crear un modelado desde el enfoque relacional, donde se detallen las entidades con sus atributos y las diversas características de los mismos, como se puede observar en la siguiente tabla:

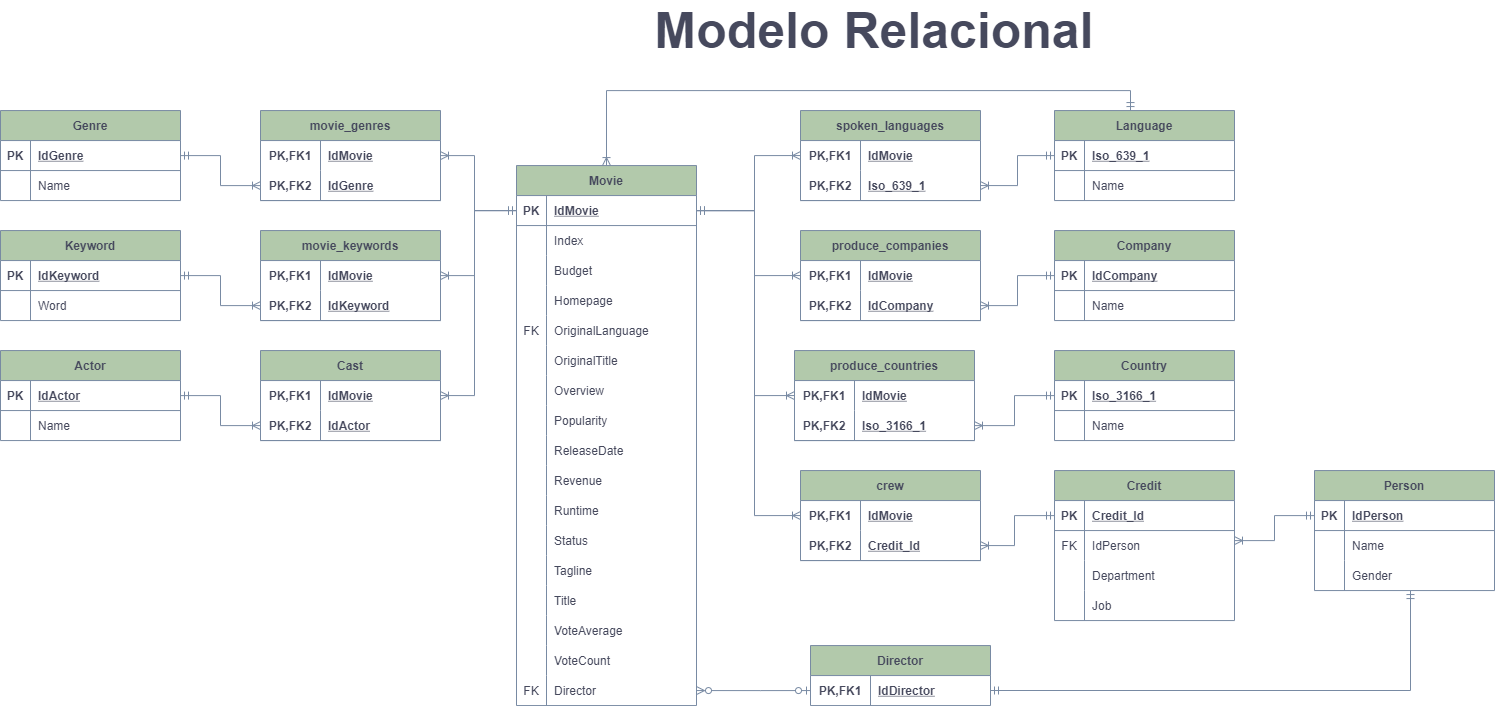
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Entity | Property | Dominio | Optional | Multivaluated | Type | Comment |
| Movie | IdMovie | Int(11) | Not Null | False | Primary Key |  |
| Movie | Index | Int(5) | Not Null | False |  |  |
| Movie | Budget | Int(12) | Not Null | False |  |  |
| Movie | Genres | Text |  | True |  |  |
| Movie | Homepage | Varchar (150) |  | False |  |  |
| Movie | Keyword | Text |  | True |  |  |
| Movie | Original\_Language | Char (2) | Not Null | False | Foreign Key |  |
| Movie | Original\_Title | Varchar (90) | Not Null | False |  |  |
| Movie | Overview | Varchar (1000) |  | False |  |  |
| Movie | Popularity | Decimal (25,21) | Not Null | False |  |  |
| Movie | Release\_Date | Date |  | False |  |  |
| Movie | Revenue | Bigint (10) | Not Null | False |  |  |
| Movie | Runtime | Decimal (4,1) |  | False |  | Tiempo de la película en minutos. |
| Movie | Status | Varchar (15) | Not Null | False |  |  |
| Movie | Tagline | Varchar (255) |  | False |  |  |
| Movie | Title | Varchar (90) | Not Null | False |  |  |
| Movie | Vote\_Average | Decimal (3,1) | Not Null | False |  |  |
| Movie | Vote\_Count | Int (10) | Not Null | False |  |  |
| Movie | Cast | Text |  | True |  |  |
| Movie | Director | Int(11) |  | False | Foreign Key |  |
| Company | IdCompany | Int(11) | Not Null | False | Primary Key |  |
| Company | Name | Varchar (85) | Not Null | False |  |  |
| Country | Iso\_3166\_1 | Char (2) | Not Null | False | Primary Key |  |
| Country | Name | Varchar (30) | Not Null | False |  |  |
| Language | Iso\_639\_1 | Char (2) | Not Null | False | Primary Key |  |
| Language | Name | Varchar (50) | Not Null | False |  |  |
| Director | IdDirector | Int(11) | Not Null | False | Primay Key, Foreign Key | Es el “id” de las personas que su cargo es director. |
| Credit | Credit\_Id | Varchar(25) | Not Null | False | Primary Key |  |
| Credit | IdPerson | Int(11) | Not Null | False | Foreign Key |  |
| Credit | Department | Varchar (25) | Not Null | False |  |  |
| Credit | Job | Varchar (75) | Not Null | False |  |  |
| Person | IdPerson | Int(11) | Not Null | False | Primary Key |  |
| Person | Name | Varchar (50) | Not Null | False |  |  |
| Person | Gender | Int (1) | Not Null | False |  | Genero del tripulante, 0 es sin especificar, 1 mujer y 2 hombre |

Basándome en las entidades que dan como resultado de aplicar la normalización, he creo conveniente crear una entidad “Director” como buena práctica del manejo de datos, para relacionarla con la entidad “Movie” en referencia al campo “director” que se encuentra de la antes mencionada.

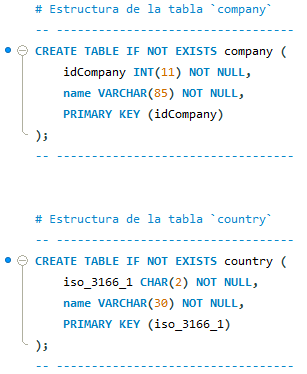
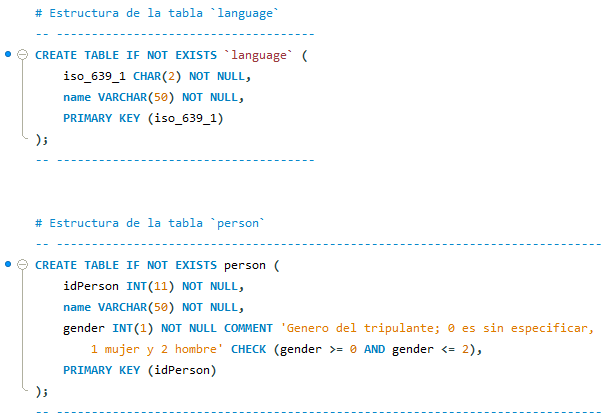
Después se determina las relaciones y cardinalidad que existe entre cada entidad en un esquema entidad-relación. Para lo cual partí de que las entidades “Company”, “Country”, “Language” y “Person” son las fuertes y todas las demás son débiles para su respectiva relación, como se puede observar en siguiente imagen:

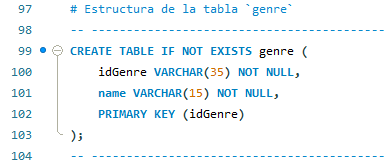
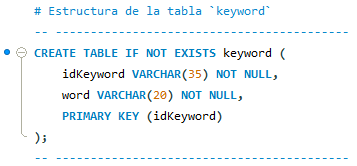


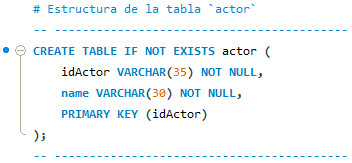
Después, diseñe el esquema relacional, tomando de referencia el modelo entidad-relación anteriormente creado, como se puede ver en la siguiente imagen:

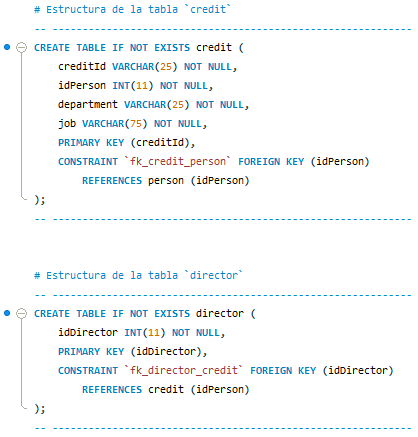
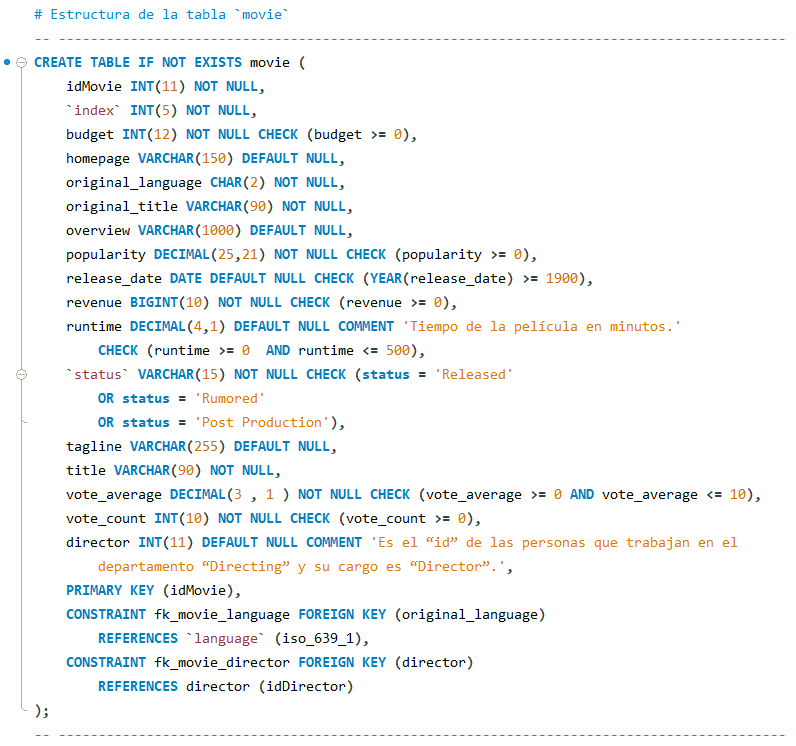


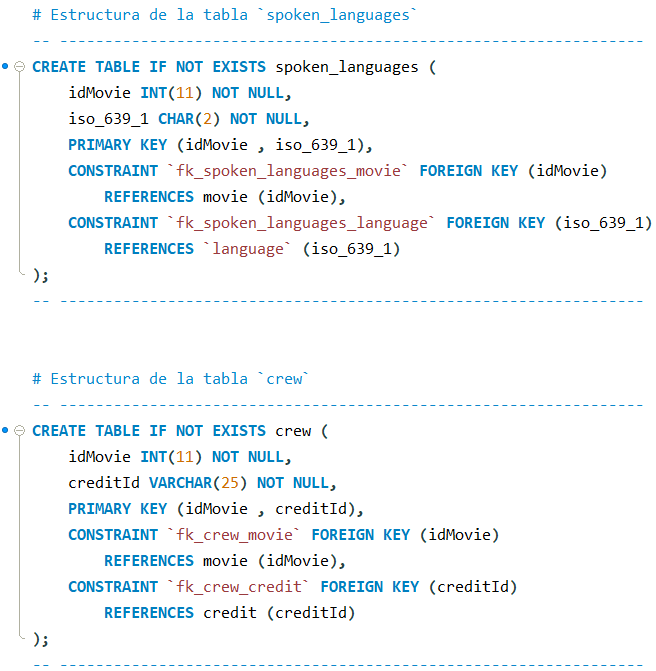
Ahora teniendo un modelo base para efectuar la prueba de que todo estaba correctamente, procedí a la crea del Lenguaje de definición de datos (DDL) para probar todo lo antes desarrollado en un gestor de base de datos. Me guie por el modelado enfocado al modelo relacional para establecer las características de los atributos de cada tabla o entidad en el DDL y en el esquema relacional para establecer las relaciones, como se puede apreciar a continuación:

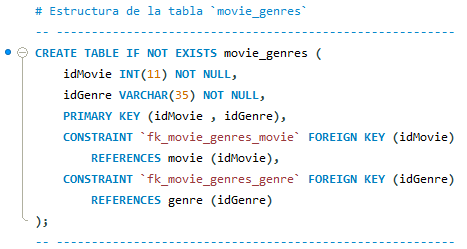
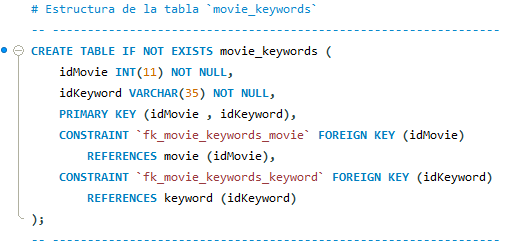
 

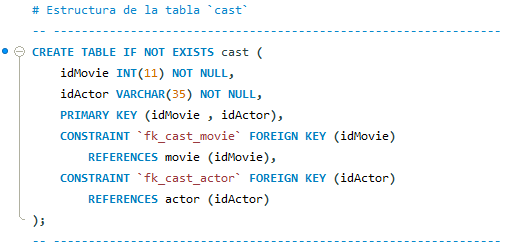
 



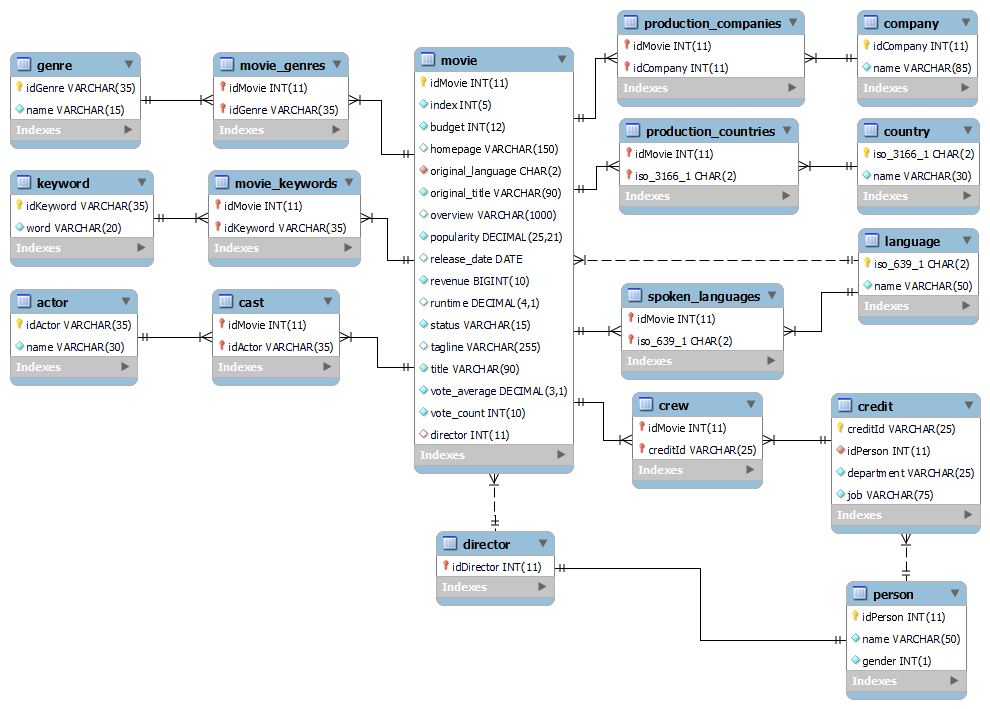
 

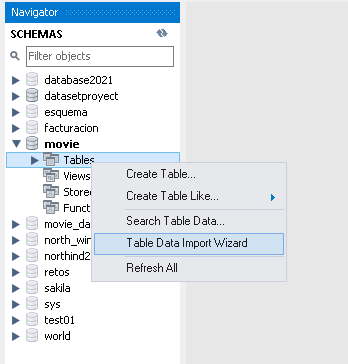
 

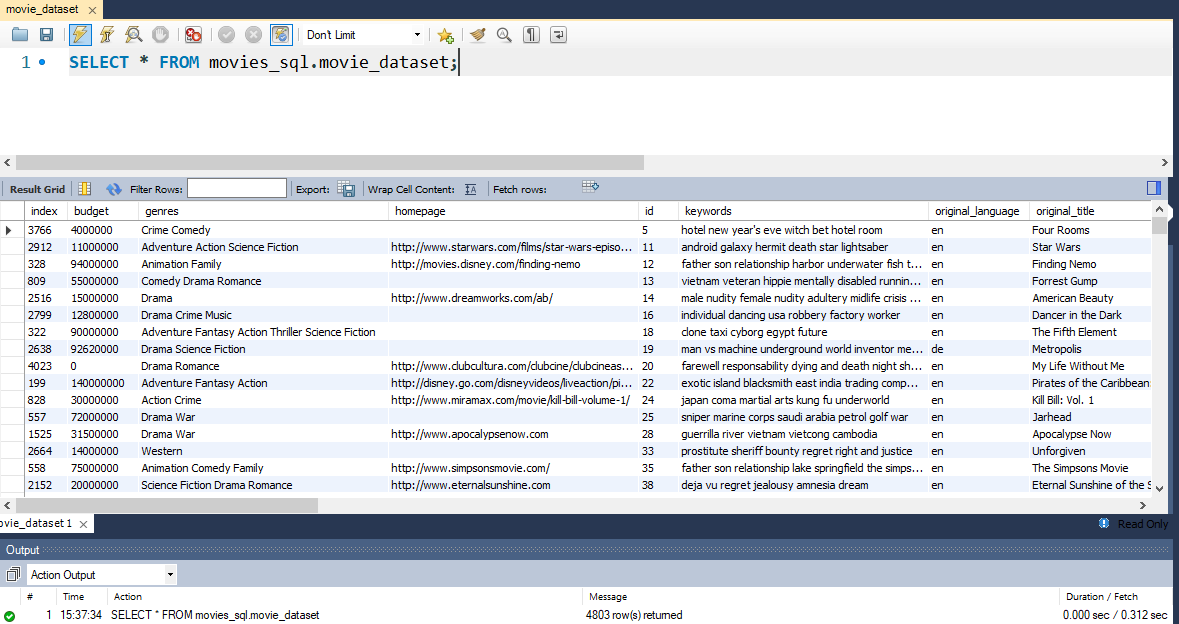


Como resultado de la creación del DDL se puede verificar que el esquema relacional creado en draw.io estaba bien, al momento de generarlo en el gestor de base de datos a partir del código, como se puede visualizar en la siguiente imagen:



Por último, faltaba cargar la data que ya disponíamos en el CSV en cada tabla, para lo cual se importo el antes mencionado en MySQL Wokbench ya que nos permite hacerlo y seguidamente procedí a crear el código para formatear la data que al inicio se identifico que estaba en mal formato, donde convierto con la función CONCAT los campos que en algunos de sus caracteres no están bien decodificados en un objeto de tipo JSON para luego aplicarles la función JSON\_EXTRACT la cual internamente decodifica las letras en base al código que encuentra en cada registro. Y en el campo “crew” se utilizó la función REPLACE para corregir el formato de tipo JSON que debe contener y también se hizo uso de la función REPLACE para cambiar algunas comillas dobles por comillas simples dentro de la data que almacena ese campo, con el fin de no tener conflictos más adelante al momento de realizar la extracción de los datos que se encuentran dentro del mismo.



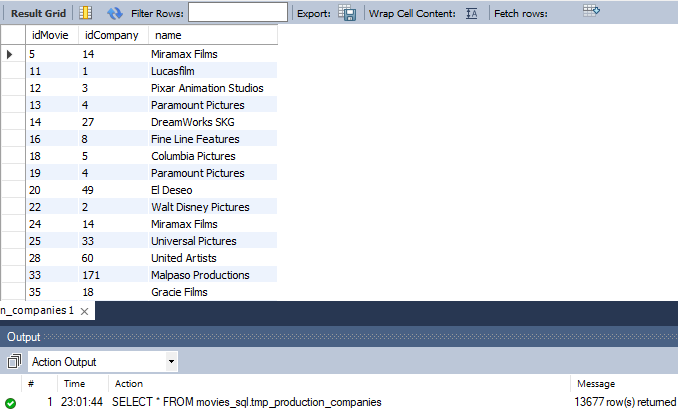
**

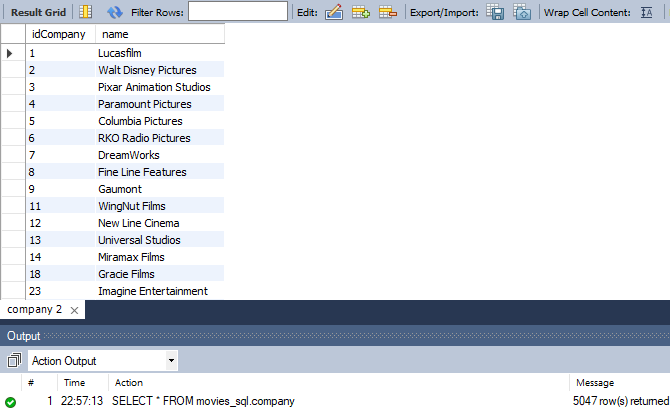


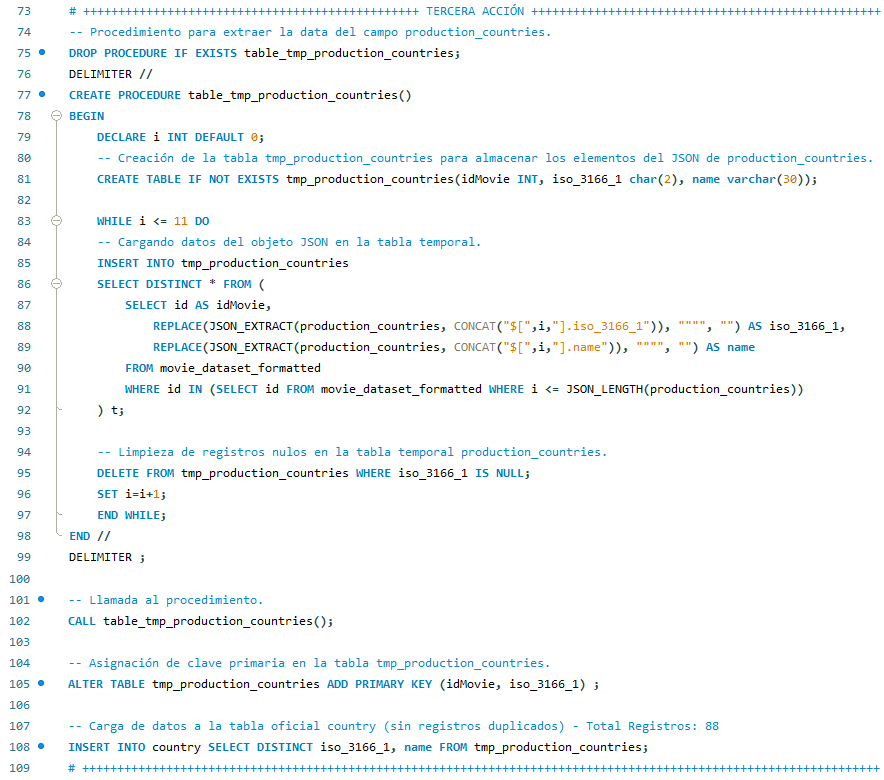
Con toda la data dentro del gestor de base de datos ya corregida, comencé a realizar procedimientos en los cuales creaba tablas temporales, las hacían referencia a los campos multivaluados y compuestos convertidos en tablas relacionales haciendo referencia a los pasos de la normalización. Además se codificaba la extracción de la data haciendo uso de la función JSON\_EXTRACT para los campos que contenían objetos de tipo JSON y para los campos que son multivaluados, cree un procedimiento parecido a los anteriores con algún cambio como lo es convertir los campos en objetos tipo JSON con la función CONCAT para después extraer los datos que contenían dentro por separado con la función JSON\_EXTRACT; y que al mismo tiempo se vaya cargando en sus tablas temporales mediante el uso de un ciclo repetitivo que recorre la data del CSV importada y corregida como tabla “movie\_dataset\_formatted” dentro del MySQL Workbench, luego codifiqué una eliminación de registros vacíos o nulos que se puedan encontrar dentro de los datos ingresados en sus tablas temporales correspondientes.

Seguidamente se alteraba las tablas temporales asignándoles las claves primarias identificadas mediante la normalización y finalmente ingresaba la data correspondiente en sus tablas finales, sacándola de las tablas temporales.

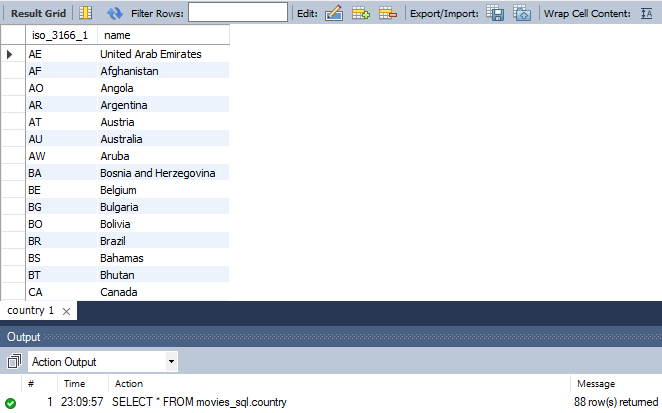


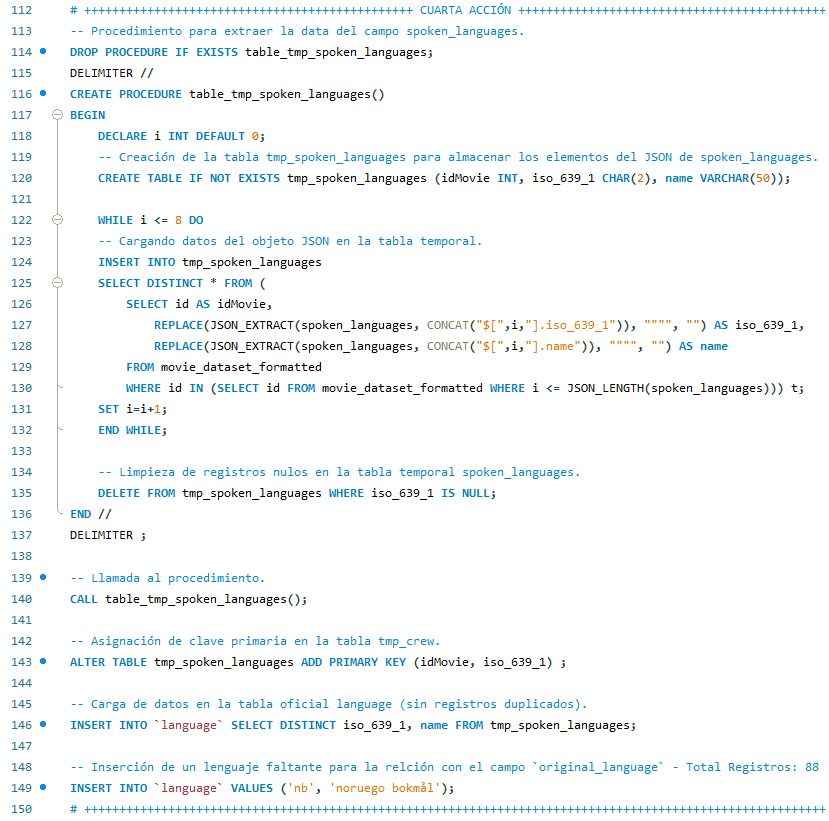


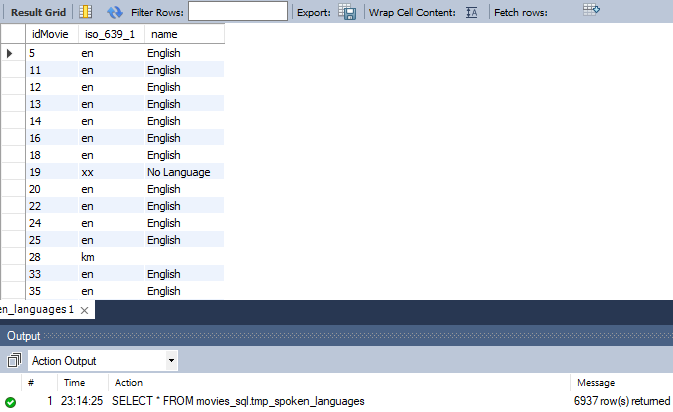


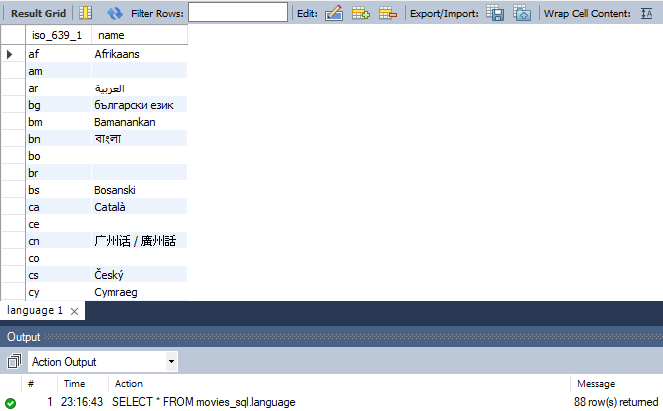




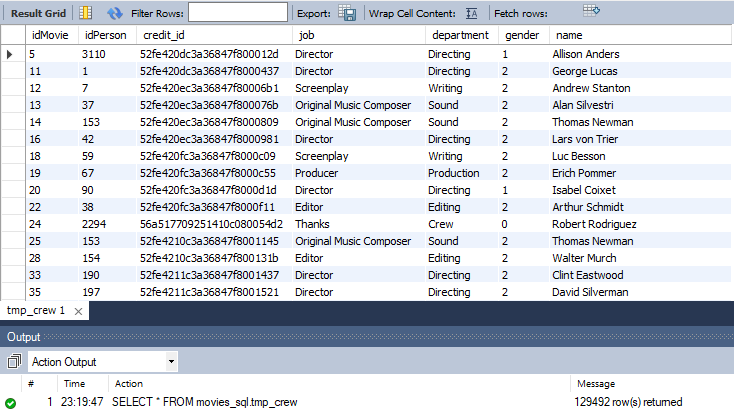


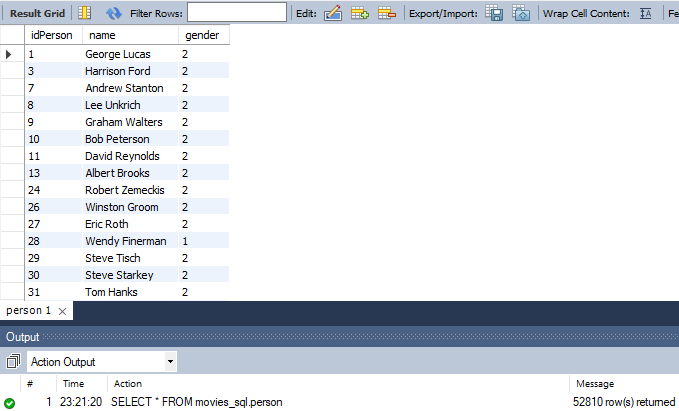


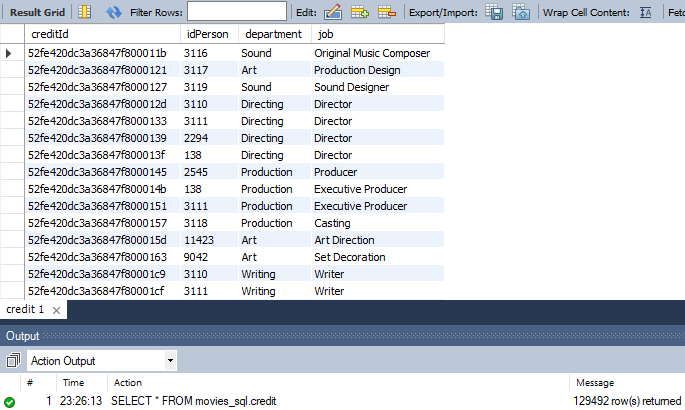


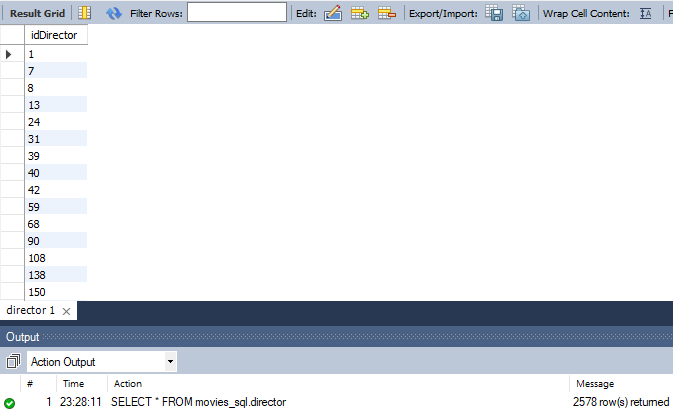
****

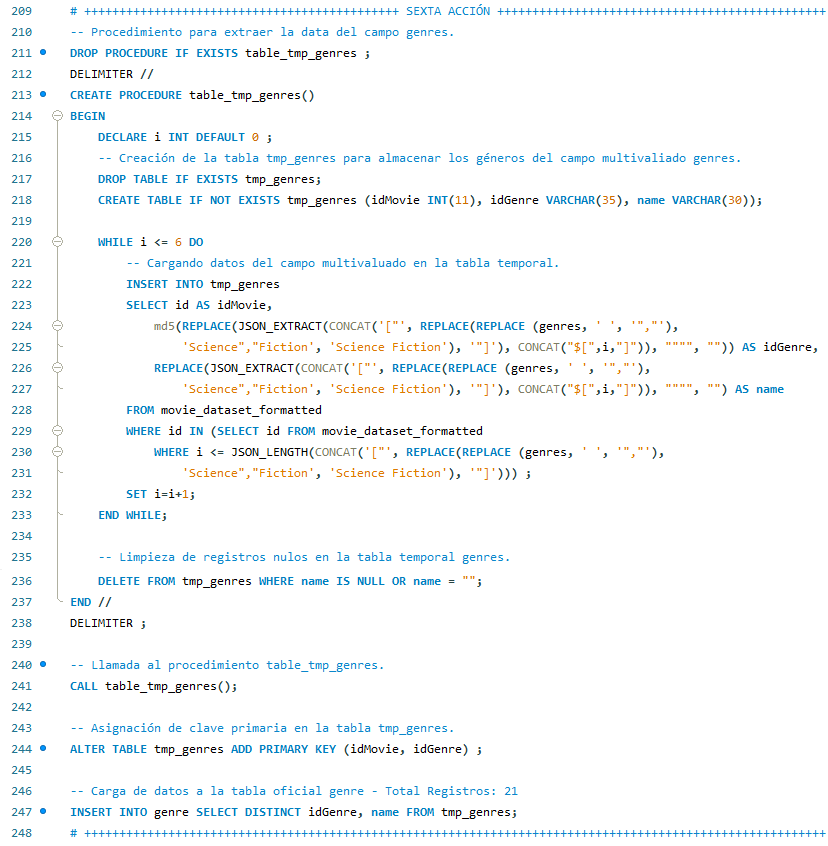


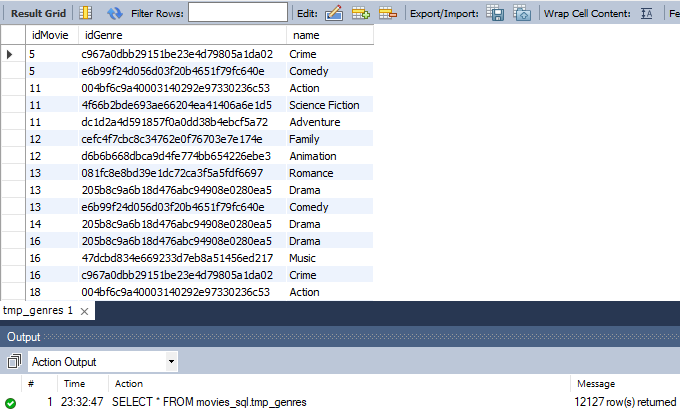


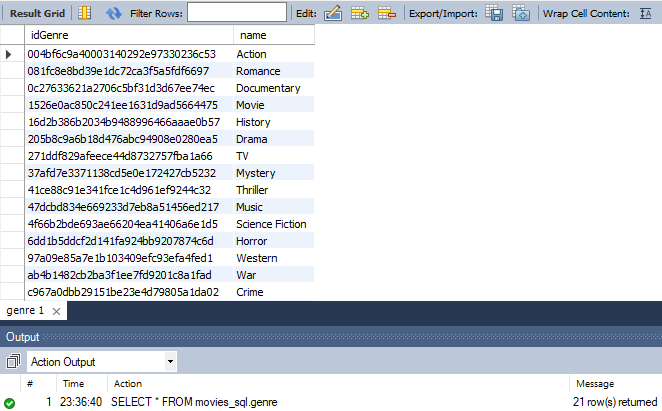


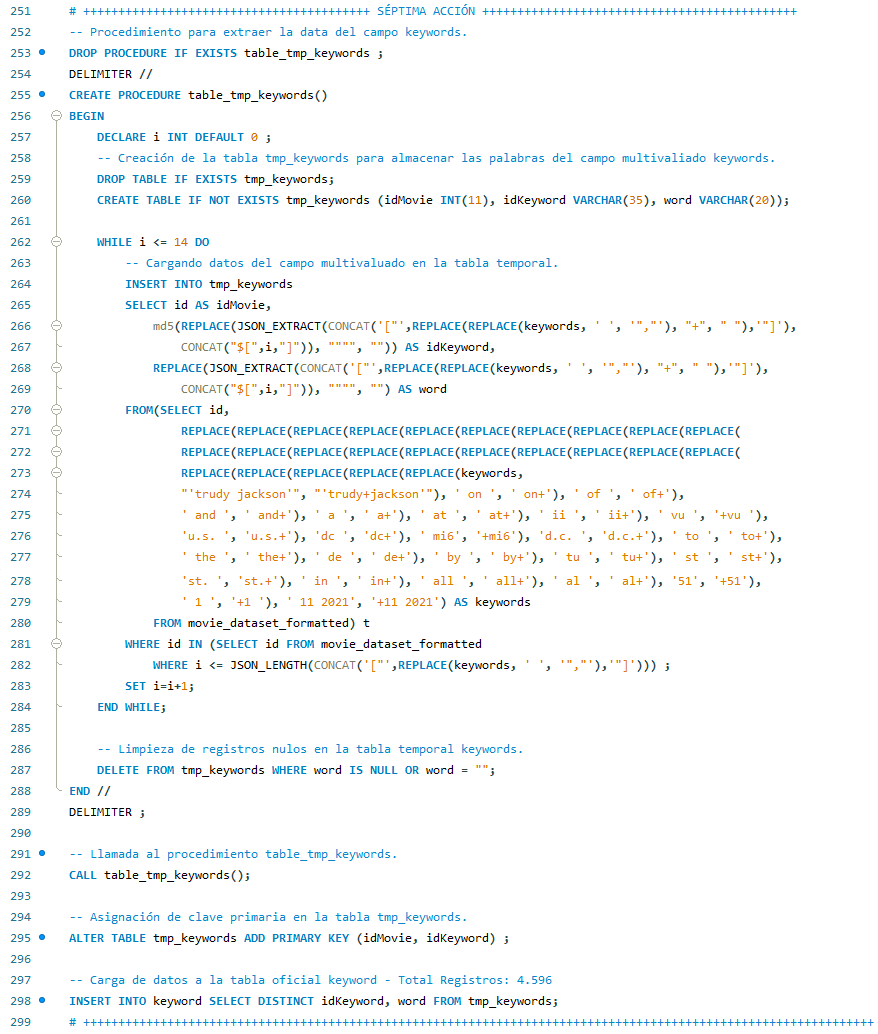


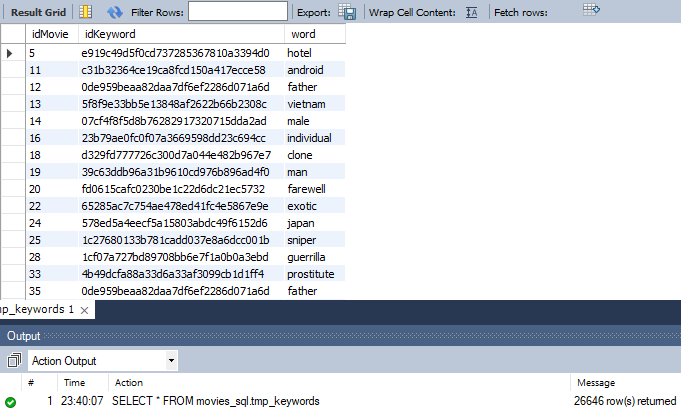


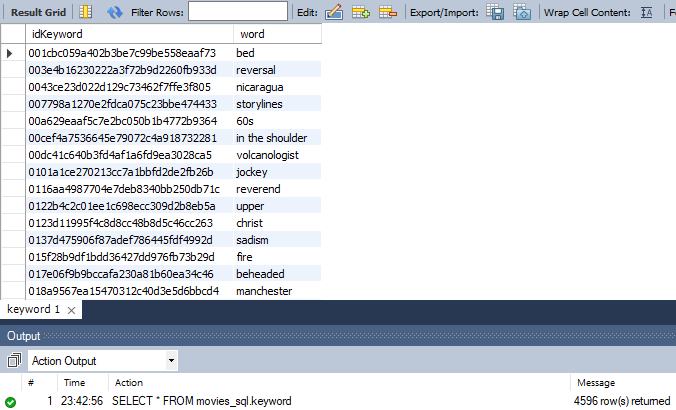




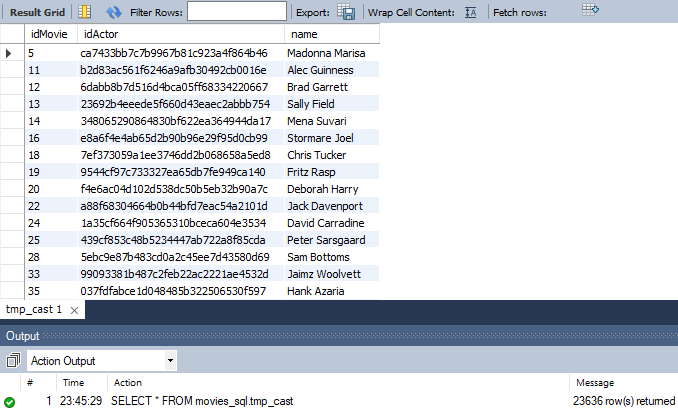


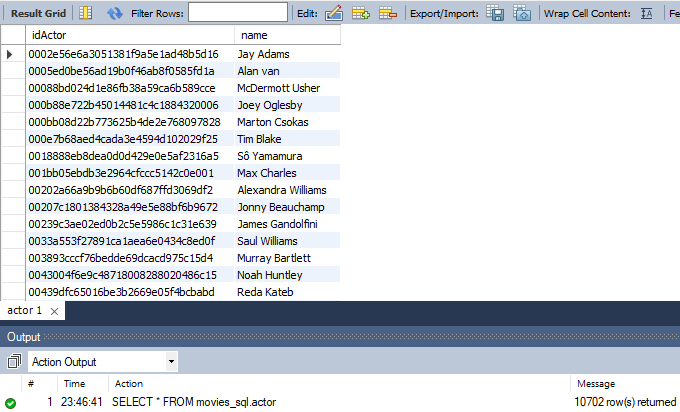


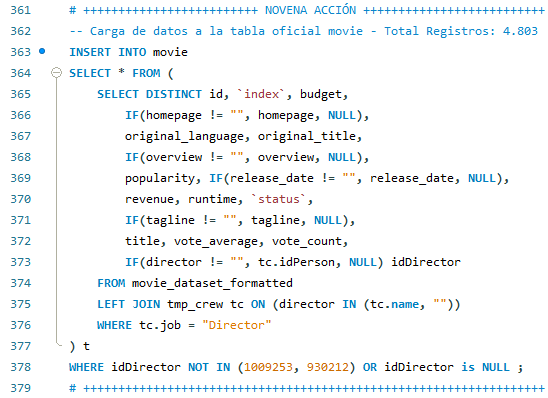


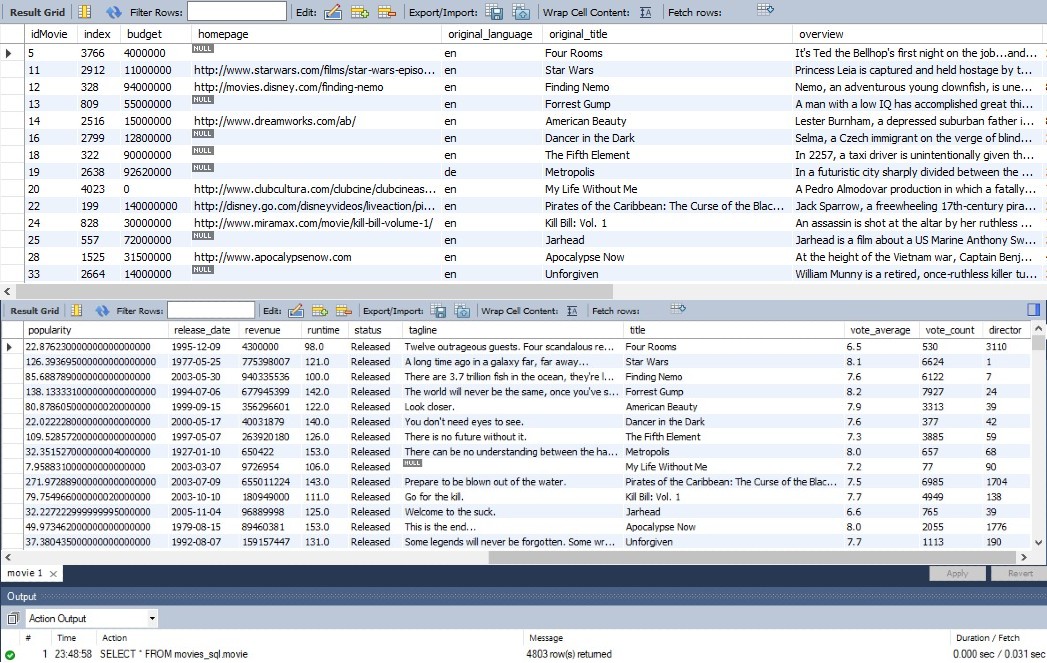


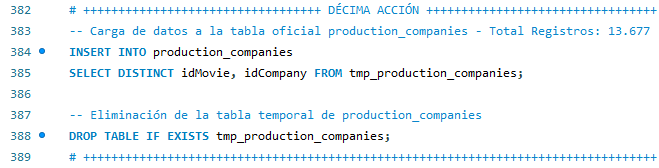


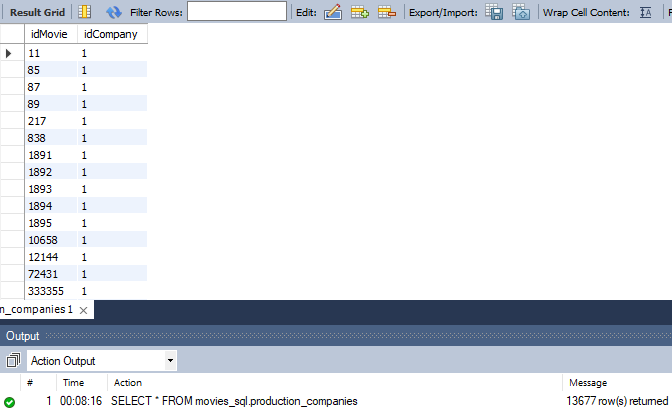


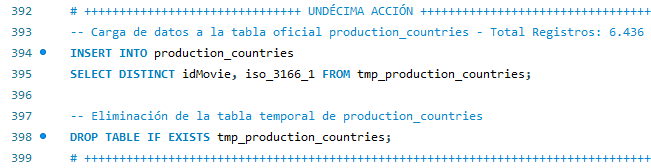


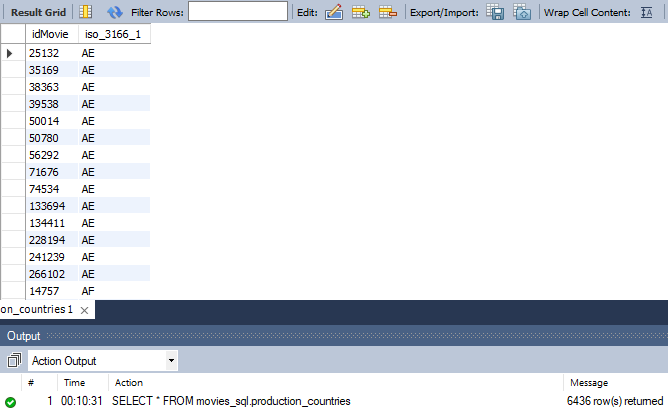


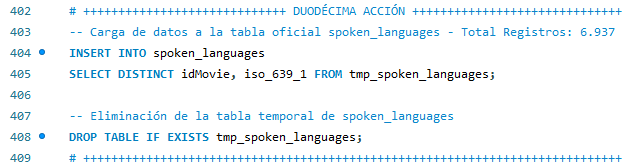


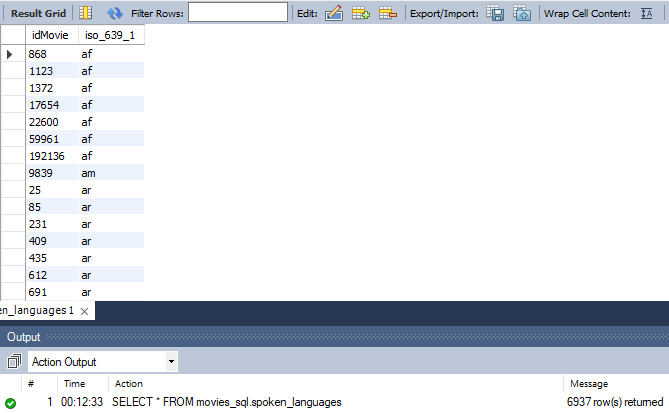


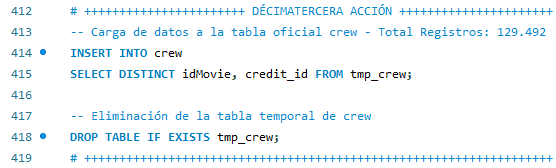


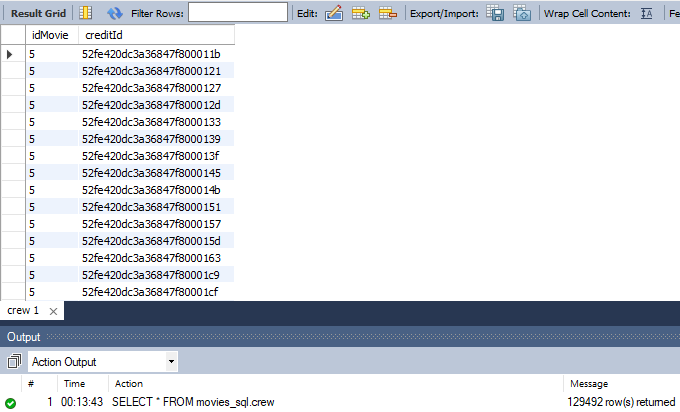


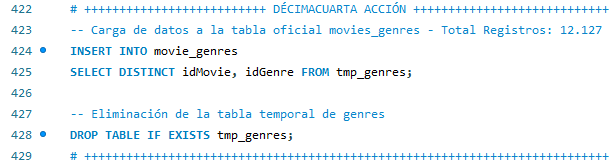


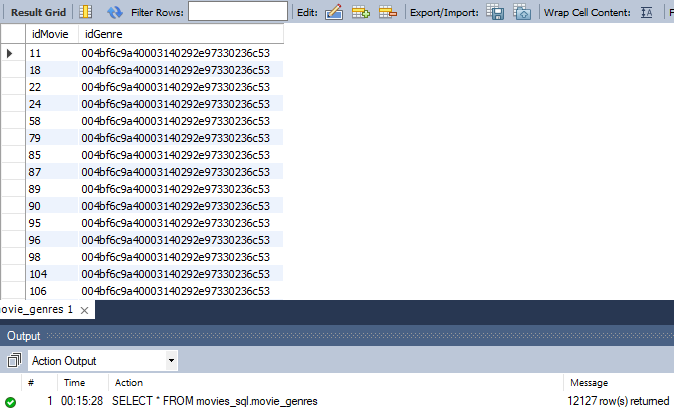


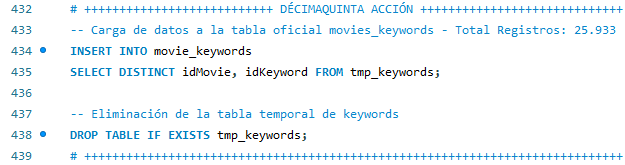


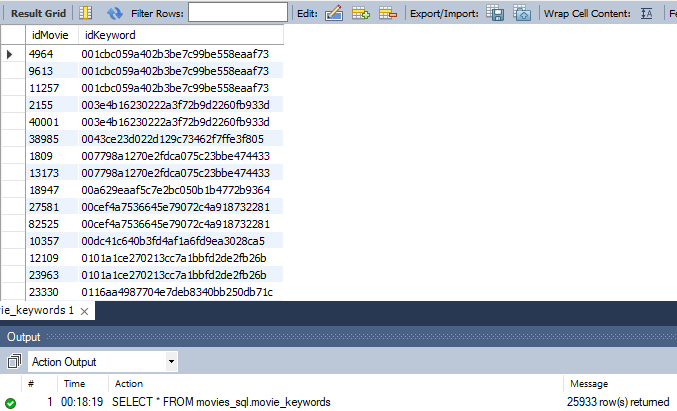


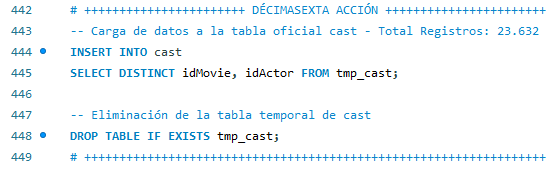


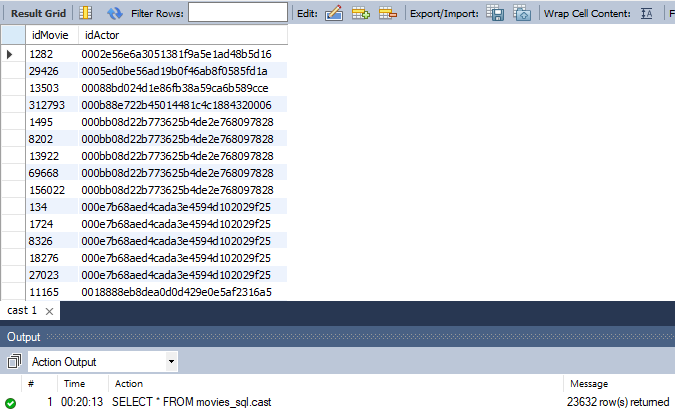












## Conclusión

En conclusión, se convierte el archivo CSV a un esquema relacional para un gestor de base de datos siguiendo una serie de pasos, como primero la normalización conjuntamente con la identificación de dependencias funcionales, luego la creación del esquema entidad-relación y el esquema relacional que nos permite tener una mejor compresión de los campos que contiene el CSV. Por último, la creación del DDL para probar toda la teoría antes realizada en un gestor de base de datos donde mediante la carga de la data se verificar que todo este correcto.

He llegado a esta solución gracias a seguir el orden correcto para la creación de un modelado para base de datos, siguiendo paso a paso la teoría para su respectiva implementación; como resultado del aprendizaje de la teoría puse en práctica todo lo aprendido y adquirí mucha experiencia en tratar muchos datos como los de tipo JSON y a determinar una solución para campos multivaluados.