

UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

*La Universidad Católica de Loja*

Informe Preliminar

Fundamentos de Bases de Datos

**Autor**: Hermin Leonardo Chuquimarca Jaramillo

Octubre 2022 – Febrero 2023

Tabla de contenido

[1. Introducción 3](#_Toc126876252)

[2. Normalización 4](#_Toc126876253)

[3. Modelos 6](#_Toc126876254)

[4. Tabla Uno a Muchos (1 : N) 9](#_Toc126876255)

[5. Tabla Muchos a Muchos (N : N) 10](#_Toc126876256)

[6. Pasos para Realizar el Proyecto 12](#_Toc126876257)

[7. LIMPIEZA COLUMNA CREW 41](#_Toc126876258)

[8. Consultas. 62](#_Toc126876259)

[9. Conclusiones 64](#_Toc126876260)

# Introducción

Este informe actual tiene como objetivo presentar un archivo CSV que se ha importado a una base de datos mediante un sistema de administración de bases de datos MySQL, que muestra diferentes fases, incluida la inserción de datos, la limpieza de datos, la carga y, especialmente, el uso de datos incluidos. Del mismo modo, podemos distinguir los métodos utilizados para llevar a cabo todas las fases propuestas en este trabajo.

Este informe y los proyectos propuestos tienen como objetivo lograr resultados hacia la gestión y el uso eficiente de información relevante e interesante basada en conjuntos de datos de trabajo.

# Normalización

First Normal Form (1NF)

Se debe seguir una serie de pasos para normalizar, en otras palabras, para decir que nuestra tabla está en primera forma normal. Estos son:

• Eliminar los grupos repetitivos de las tablas individuales.

• Crear una tabla separada por cada grupo de datos relacionados.

• Identificar cada grupo de datos relacionados con una clave primaria.

Ejemplo:

Una tabla llamada cast con una tabla llamada movie se relacionan en varios por varios (m:n), ya que una película puede tener varios actores, así también como un mismo actor puede participar en diferentes películas; con esta relación se los identifica con una clave primaria idCast y el idMovie respectivamente.

La segunda Forma Normal (2FN)

Se debe seguir los siguientes pasos:

• Tener la 1° forma normal

• Crear tablas separadas para aquellos grupos de datos que se aplican a varios registros

• Relacionar estas tablas mediante una clave externa

Ejemplo:

La tabla llamada original title que se relaciona con la tabla keywords, IdOriginalTitle y el idKeywords respectivamente, cuentan con una clave externa que es idMovies, debido a que es la clave principal de la tabla universal que es Movies.

La tercera Forma Normal (3FN)

Se debe considerar los siguientes puntos:

• Tener la 2° forma normal

• Eliminar aquellos campos que no dependan de la clave

• Ninguna columna puede depender de una columna que no tenga una clave

• No puede haber datos derivados

Ejemplo:

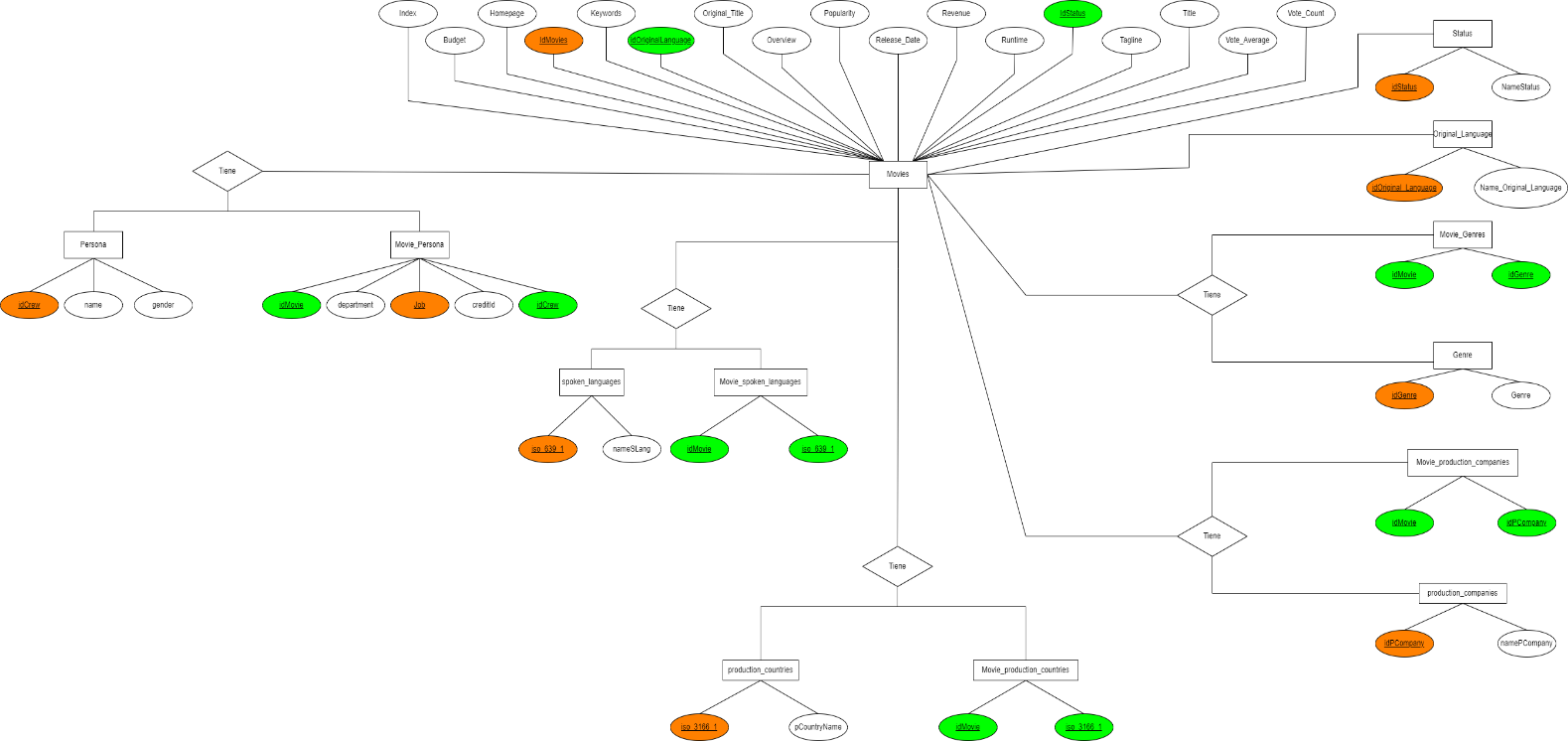
El más claro ejemplo para esta tercera forma normal es la tabla universal “Movies”, ya que es la única tabla que no depende de otras tablas.

Teniendo en base estos conceptos, procedimos a realizar los respectivos modelos Entidad/Relación, Lógico y Físico. Ya con esta conceptualización, generamos un pequeño script con la creación de tablas, determinando ya el tipo de dato que serían de los determinados atributos, establecer los tipos de llaves primarias y foráneas, entre muchas más características necesarias. Esto ya implementado en el motor de bases de datos a utilizar.

# Modelos

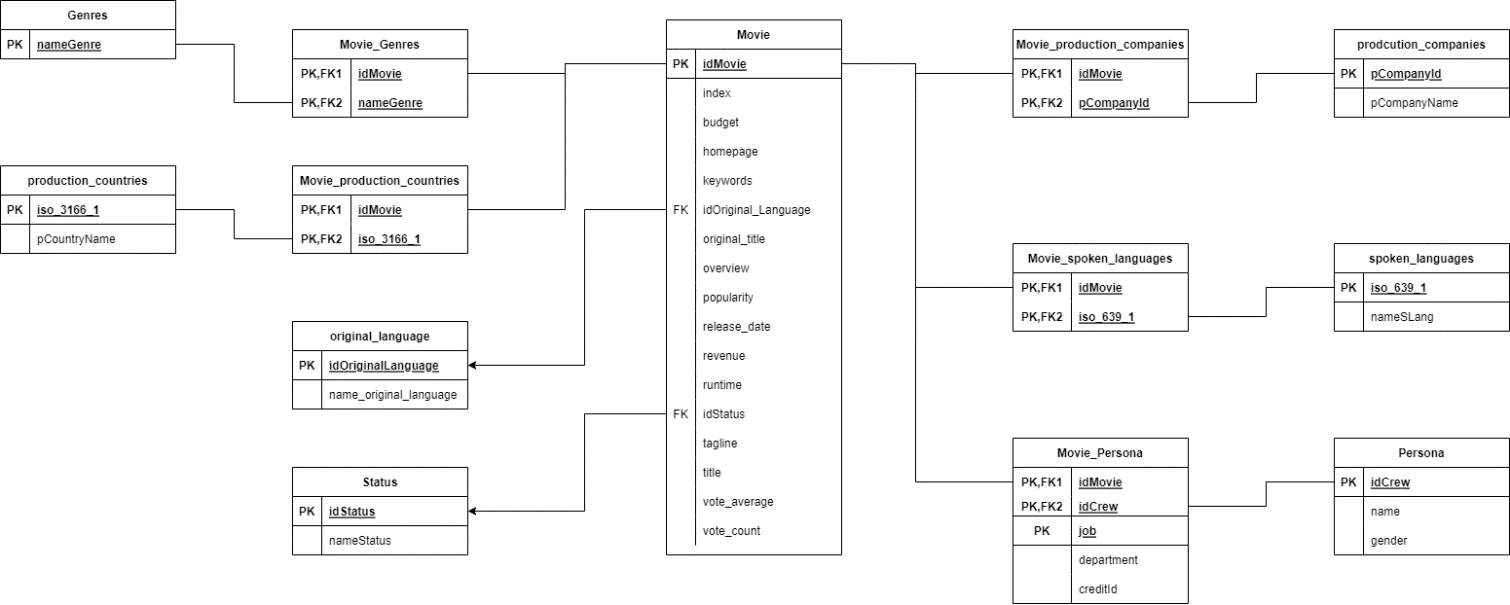
Modelo Conceptual

Después de haber establecido los datos mediante la tabla universal, normalizamos los mismos utilizando la metodología Entidad-Relación, identificando los atributos correspondientes a cada relación. Para ello, utilizamos la herramienta draw.io, lo que nos permitió crear diagramas de flujo efectivamente.



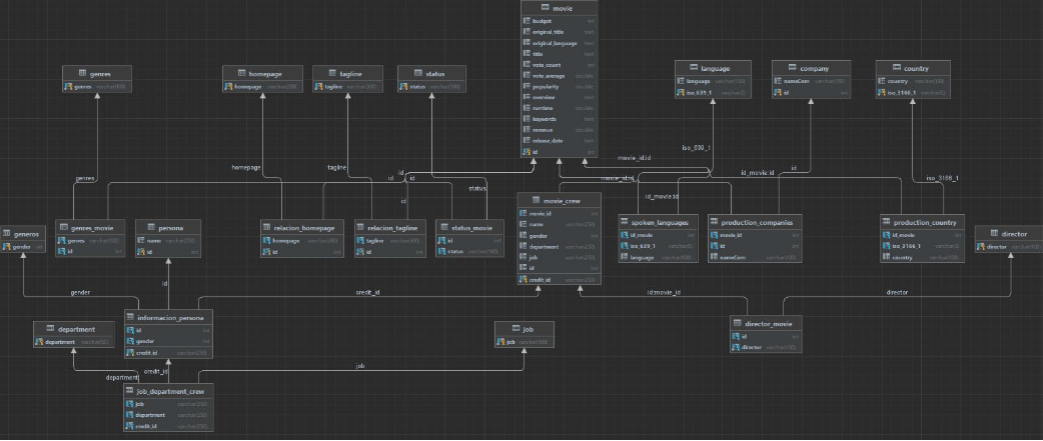
Modelo Lógico

Con el modelo conceptual ya creado, los atributos apropiados para cada tabla se seleccionan y crean prototipos en función de las entidades y relaciones identificadas. Se declara una clave principal y, opcionalmente, una clave externa en cada tabla para garantizar una base sólida para desarrollar el esquema de la base de datos.



Modelo Físico

Una vez que el modelo lógico está completo, procedemos a la fase final de construcción del modelo físico. Aquí se especifican las tablas, columnas, claves primarias y sus claves foráneas o claves foráneas y sus relaciones. Este modelo puede generar el correspondiente registro DDL.



Una vez establecidos los modelos se identificó y eliminó grupos repetitivos.

# Tabla Uno a Muchos (1 : N)

Status es una columna con tres posibles valores (released, rumored, post-production)

* Una película puede tener un Status, pero un Status puede asociarse a muchas películas.
* Para solucionar, se crea una tabla separada que se llame Status.
* La llave primaria es nameStatus.
* Como solución, utilizamos el nameStatus como llave foránea en Director que es una columna con el nombre de un único director.
* Una película puede ser dirigida por un Director, pero un Director puede dirigir muchas películas.
* Para solucionar, se crea una tabla separada que se llame Director.
* La llave primaria es directorName.

Como solución, utilizamos el directorName como llave foránea en Original\_language es una columna con el nombre del lenguaje original de la Movie.

Una película tiene un lenguaje original, pero un lenguaje original puede estar en muchas películas, para solucionar, se crea una tabla separada que se llame original\_language, la llave primaria es original\_languageName.

Como solución, utilizamos el original\_languageName como llave foránea en la tabla original\_language.

# Tabla Muchos a Muchos (N : N)

En ninguna de las 4803 entradas se repite, cada película es diferente (única), Cada Movie tiene un index y un id diferente. Como llave primaria utilizamos id el cual nombramos idMovie para no confundirlo con otros id que existan en otras columnas.

Genres contiene un String de géneros que puede contener 0 a n géneros.

* + - * Para solucionar, cada columna debe contener un solo valor.
    - En este caso tenemos una lista de géneros. Existen múltiples valores.
* La solución es crear una tabla separada que se llame Genres.
  + - * La llave primaria es genreName
      * Una Movie puede tener muchos géneros, y un género puede estar en muchas Movie.
      * La relación (muchos a muchos) se soluciona con una tabla llamada
      * Movie\_Genres utilizando la llave primaria de cada uno.
      * Keywords contiene un String de keywords que puede contener (0 a N)
      * Para solucionar, cada columna debe contener un solo valor.
    - En este caso tenemos una lista de keywords. Existen múltiples valores.
      * La solución es crear una tabla separada que se llame Keywords.
      * La llave primaria es keywordName
    - Una Movie puede tener muchos keywords, y un keyword puede aparecer en muchas Movie.
      * La relación (muchos a muchos) se soluciona con una tabla llamada
      * Movie\_Keywords utilizando la llave primaria de cada uno.

Production\_companies contiene un String de JSON que contiene un name y un id que puede contener 0 a n production\_companies.

Para solucionar, cada columna debe contener un solo valor.

* + - * En este caso tenemos una lista de production\_companies. Existen múltiples valores.
      * La solución es crear una tabla separada que se llame Production\_companies.
      * Tiene dos atributos, name e id los cuales los nombramos prodCompName y prodCompId el cual es la primary key.
      * Una Movie puede tener muchos production\_companies, y un production\_companies puede aparecer en muchas Movie.
      * La relación (muchos a muchos) se soluciona con una tabla llamada
      * Movie\_ Production\_companies utilizando la llave primaria de cada uno.
      * Production\_countries contiene un String de JSON que contiene un iso\_3166\_1 y un name que puede contener 0 a n production\_countries.
      * Para solucionar, cada columna debe contener un solo valor.
      * En este caso tenemos una lista de production\_countries. Existen múltiples valores.
      * La solución es crear una tabla separada que se llame Production\_countries. Tiene dos atributos, iso\_3166\_1 y name los cuales los nombramos, iso\_3166\_1 el cual es la primary key y prodCompName.
      * Una Movie puede tener muchos production\_countries, y un production\_countries puede aparecer en muchas Movie.
      * La relación (muchos a muchos) se soluciona con una tabla llamada
      * Movie\_ Production\_countries utilizando la llave primaria de cada uno.

Cada Movie tiene uno de los siguientes:

* + - * Production\_countries
      * Production\_companies
      * Spoken\_languages
      * Crew

Para no tener una fila con una clave principal de Película y todos los atributos de cada atributo anterior, use solo la clave principal de cada atributo anterior con película y tenga una tabla con dos claves principales. Luego puede identificar la tabla a la que pertenece y asociar el atributo específico asociado con esa clave de esta manera.

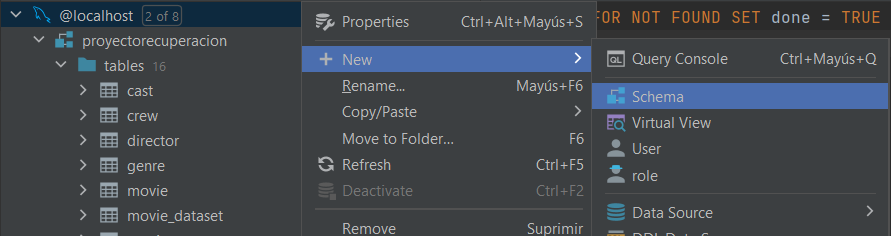
# Pasos para Realizar el Proyecto

Creación de un “Schema”.

Para la creación del “Schema” se podía realizar de dos distintas maneras, la creación automática o por sentencias SQL.

**OPCIÓN 1 – MEDIANTE LA CREACIÓN AUTOMÁTICA**

Para la creación automática del Schema dentro del lenguaje de Base de Datos MySql en DataGrip, primero se debe ubicar en la parte superior izquierda, donde se encuentra el localhost que se encuentra conectado, al señalarlo, le aparece un nuevo cuadro donde se tendrá que ir a “New” y luego en el último cuadro se va donde dice “Schema” y automáticamente se crea el schema como se puede ver en el siguiente Anexo:



**OPCIÓN 2 – MEDIANTE SENTENCIAS SQL**

Para la creación del “Schema” o base de datos usando sentencias SQL, es necesario hacer lo siguiente:



Es necesario especificarle la codificación de caracteres utf8 ya que en el archivo CSV se usa caracteres especiales y esta misma codificación los transforma.

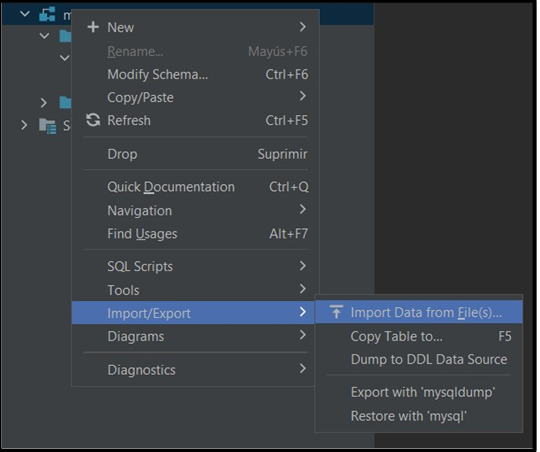
Conexión del “Schema” a la Base De Datos.

Para poderse conectar, por así decirlo, es necesario hacerlo mediante sentencias SQL, la sentencia es la siguiente:



Importación del CSV.

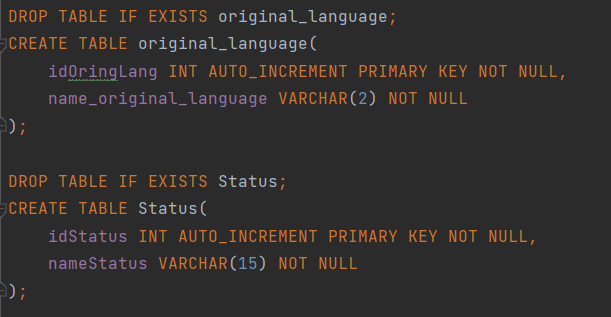
DataGrip es un IDE de JetBrains para el manejo de base de datos. Dentro de este existe una herramienta facilitadora de importación de varios tipos de archivos a tablas MySQL

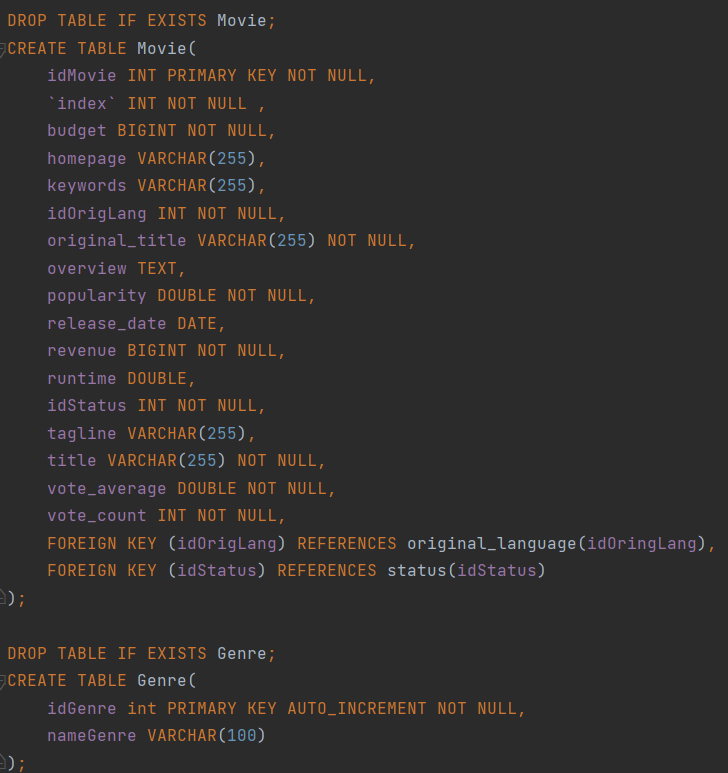


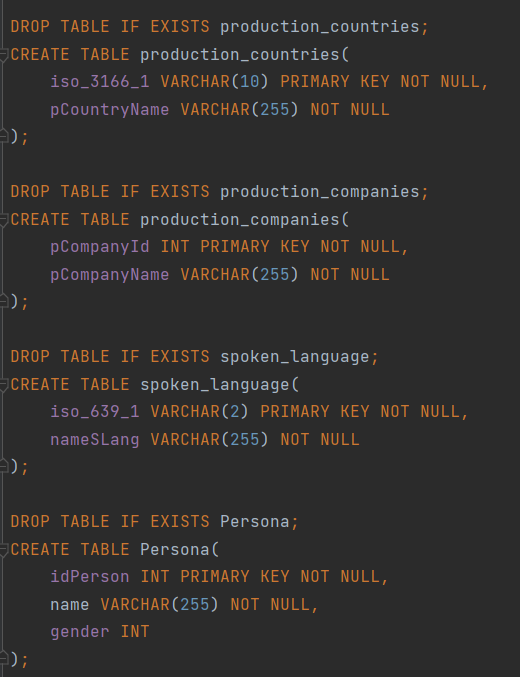
Una vez entrado a este apartado, se habilita una interfaz de importación en donde especificamos que el separador es la coma, y que la primera fila son los títulos de las columnas

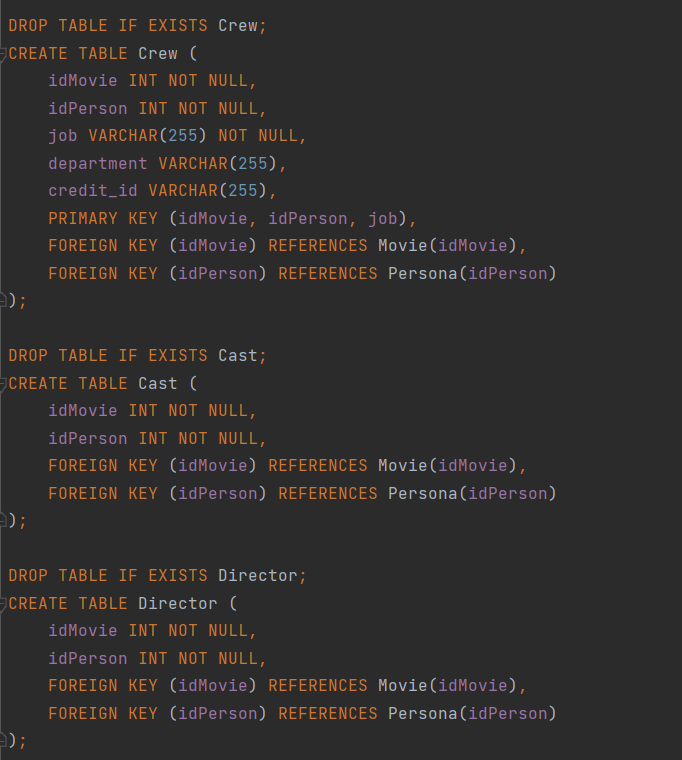


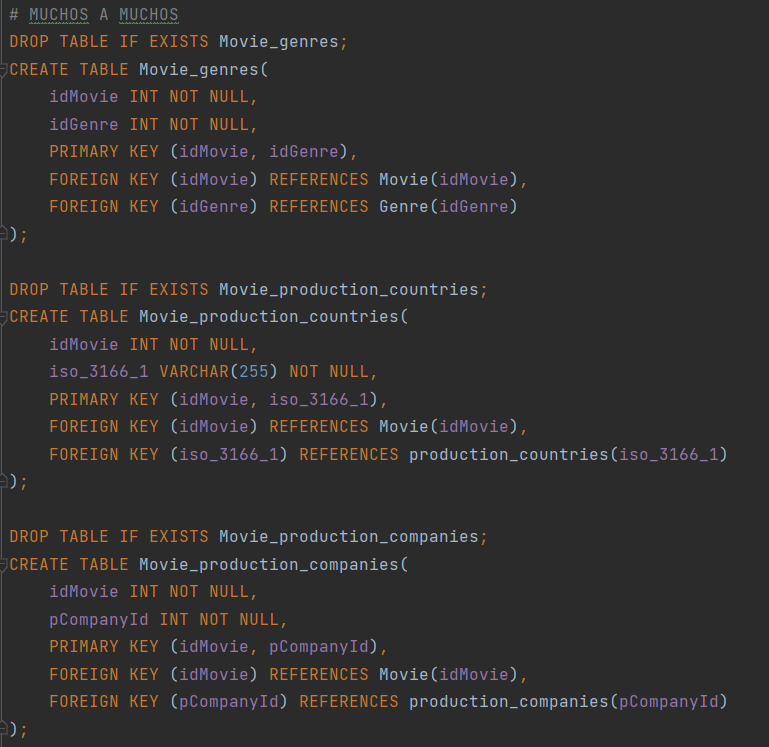
Creación de tablas en DATAGRIP

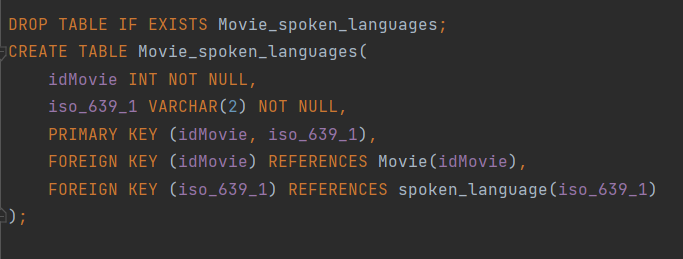












Creación de funciones que Permitan Extraer y Limpiar los Datos.

Descripción de las tablas Director, Status y original\_langauge

(Tablas uno a muchas)

Lo primero que se hace en el procedimiento almacenado en SQL es que este comienza verificando si existe un procedimiento con el nombre "TablaDirector". Si existe, se elimina. Luego, se declaran dos variables: "done" con un valor predeterminado de "FALSE" y "nameDirector" de tipo VARCHAR con una longitud máxima de 100 caracteres.

Después, se declara un cursor "CursorDirector" que selecciona los nombres únicos de los directores en la tabla "movie\_dataset". Se establece un manejador de continuación "CONTINUE HANDLER" para detectar cuando se ha alcanzado el final del cursor. Se abre el cursor y se inicia un ciclo repetitivo que recorre cada uno de los nombres de los directores en el cursor.

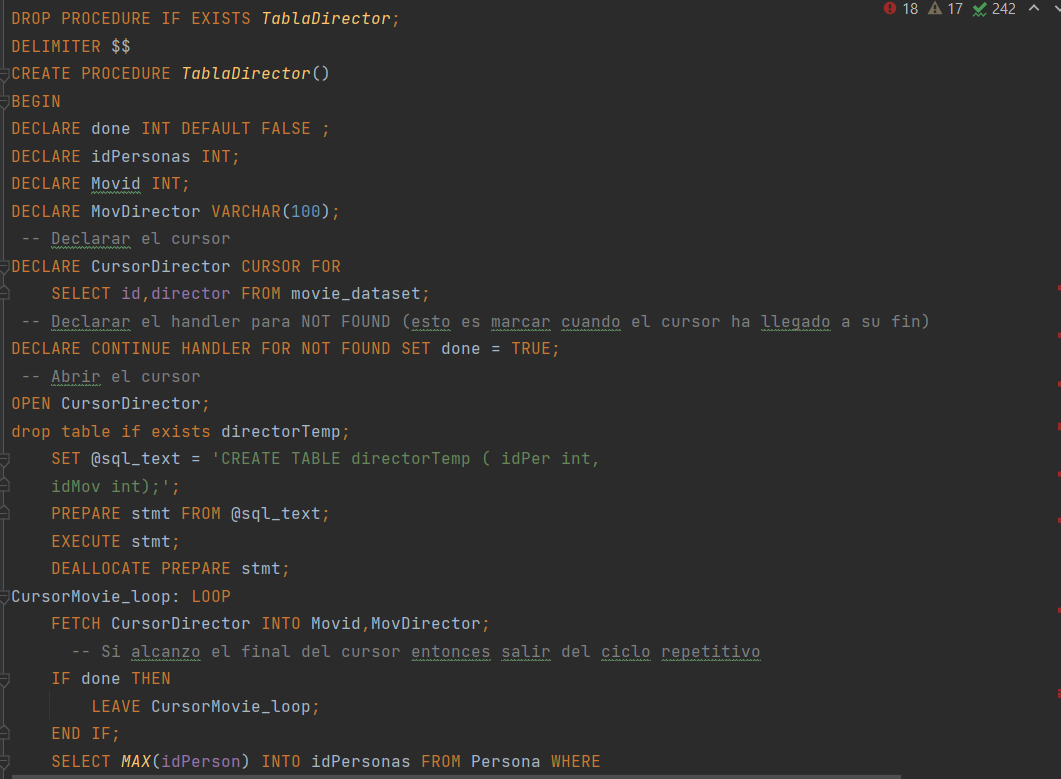
En cada iteración del ciclo, se asigna el nombre del director a la variable "nameDirector". Si se ha alcanzado el final del cursor, se sale del ciclo. Si el nombre de director es nulo, se asigna un valor vacío. Además, se tiene casos especiales en los nombres que con el Replace se puedo solucionar en la misma sentencia de código.

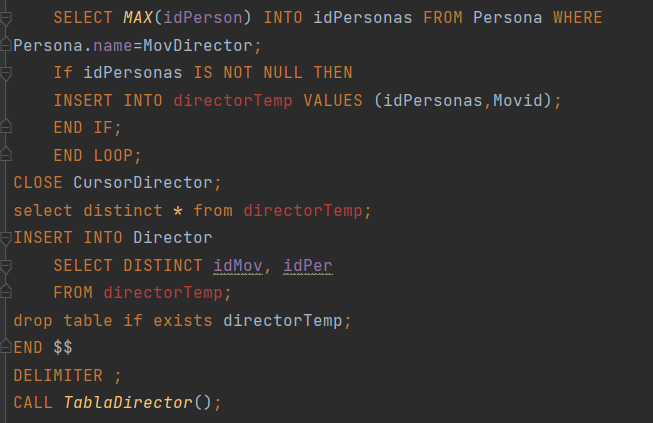
Luego, se crea una consulta dinámica que inserta el nombre del director en la tabla "director". Se prepara y ejecuta la consulta dinámica y se libera la memoria de la consulta preparada. Se repite este proceso para cada nombre de director en el cursor.

Finalmente, se cierra el cursor y se finaliza el procedimiento almacenado. En resumen, este procedimiento permite crear y llenar una tabla con los nombres únicos de los directores en una tabla de origen.

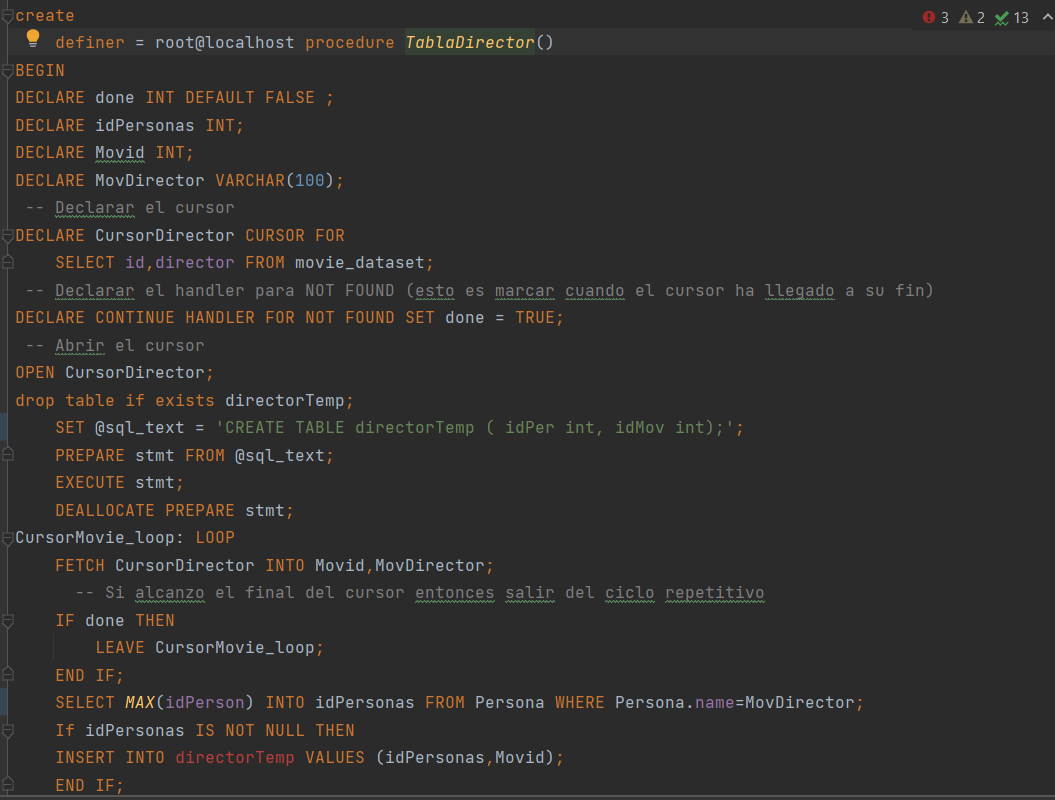
Este código nos servirá para poblar estas tres tablas.

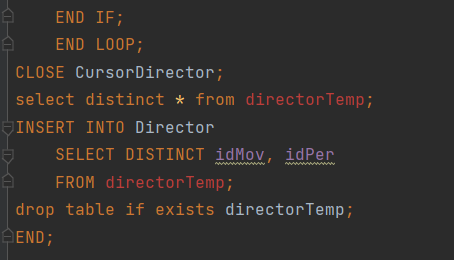
**Población Tabla “director”**



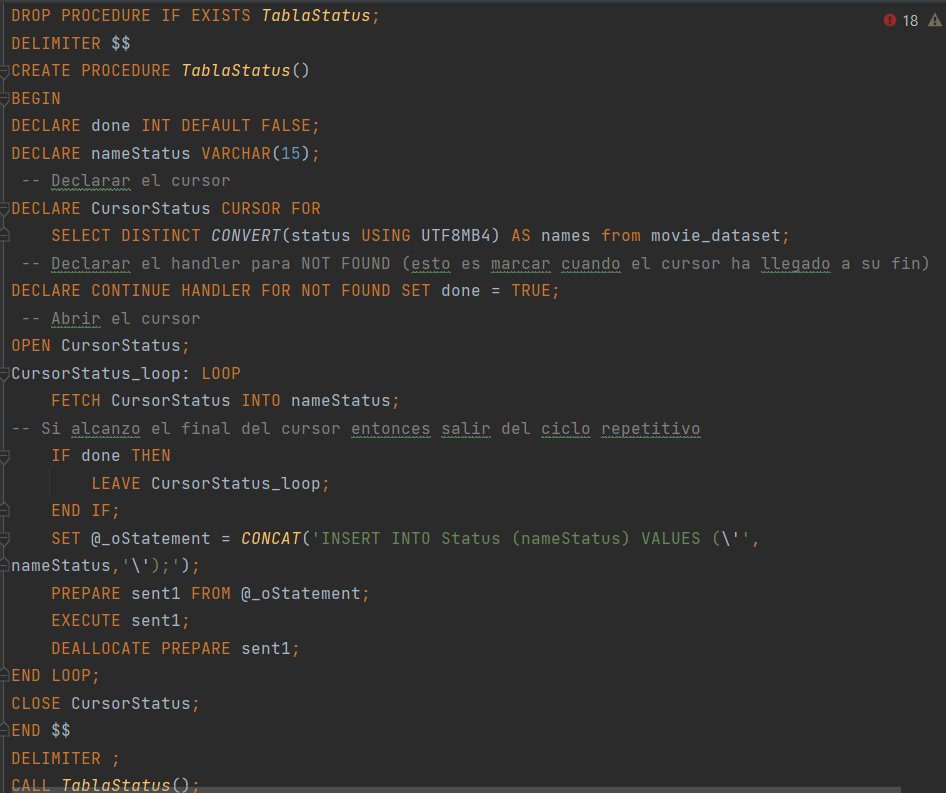


ROUTINE “TablaDirector”

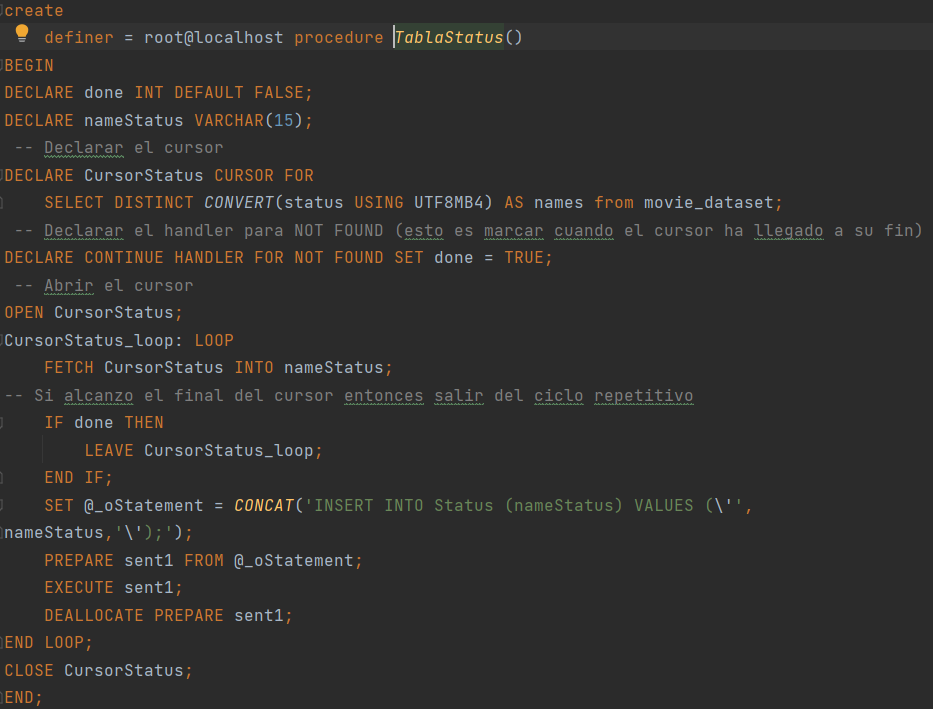
****



**Población Tabla “Status”**



ROUTINE “TablaStatus”

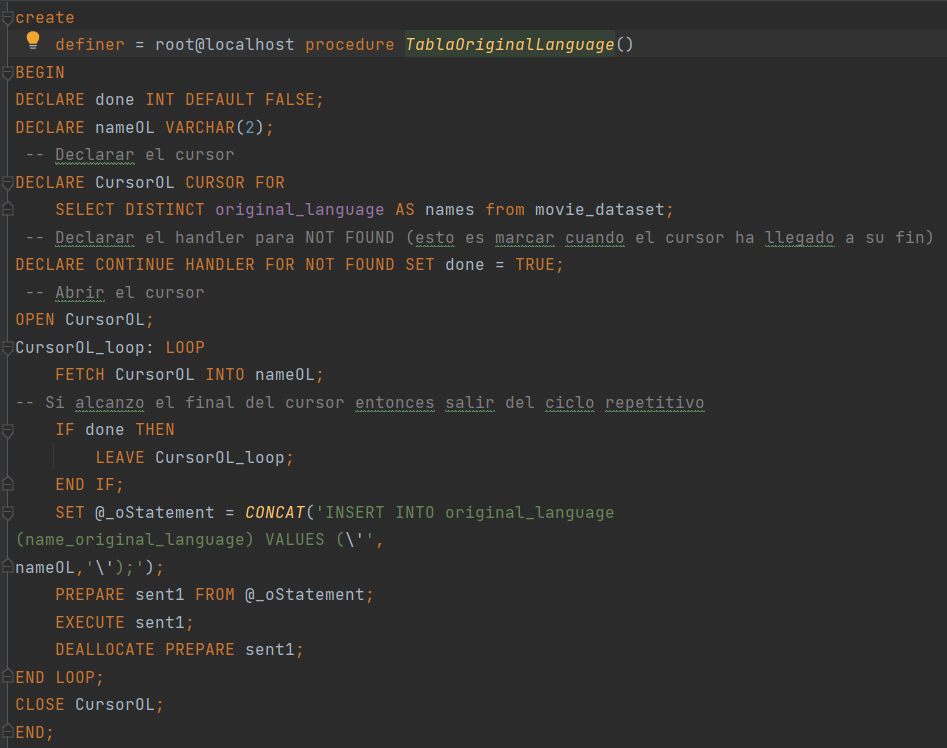


**Población Tabla “Original\_language”**





ROUTINE “TablaOriginalLanguage”



Debido a que la tabla Movie utiliza llaves foráneas, era necesario crear las tablas director, Status y original\_langauge para utilizar sus llaves primarias como foráneas en Movie. (REFERENCES)

**Población Tabla “Movie”**

El siguiente código SQL es un procedimiento almacenado que crea una tabla denominada "películas" en la base de datos. La tabla se crea a partir de los datos de un conjunto de datos denominado "movie\_dataset".

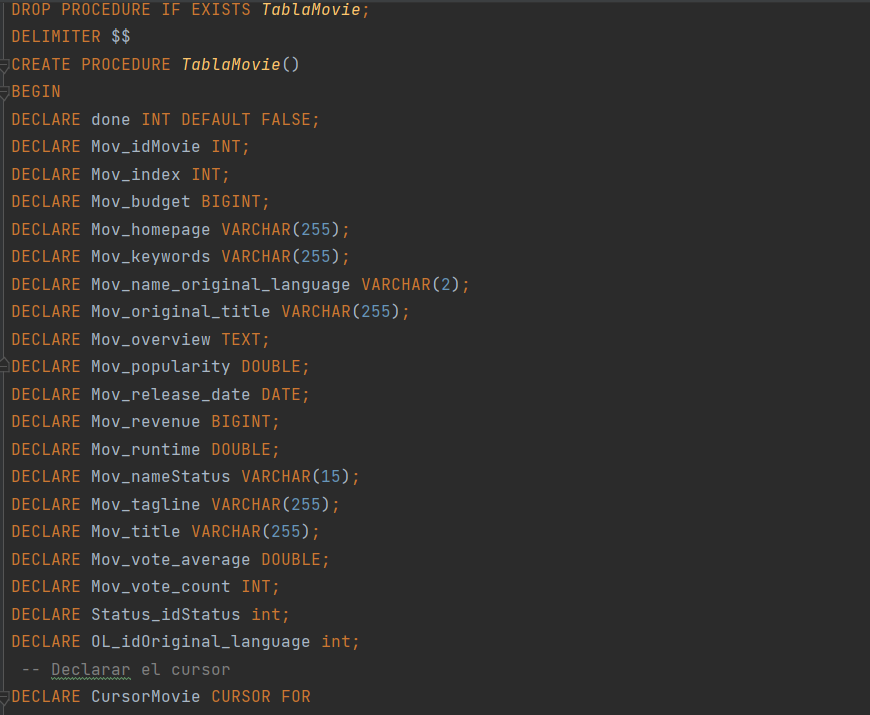
Un procedimiento almacenado primero elimina todos los procedimientos existentes con el mismo nombre. Luego se declaran varias variables para almacenar los valores de cada columna en la tabla "movie\_dataset

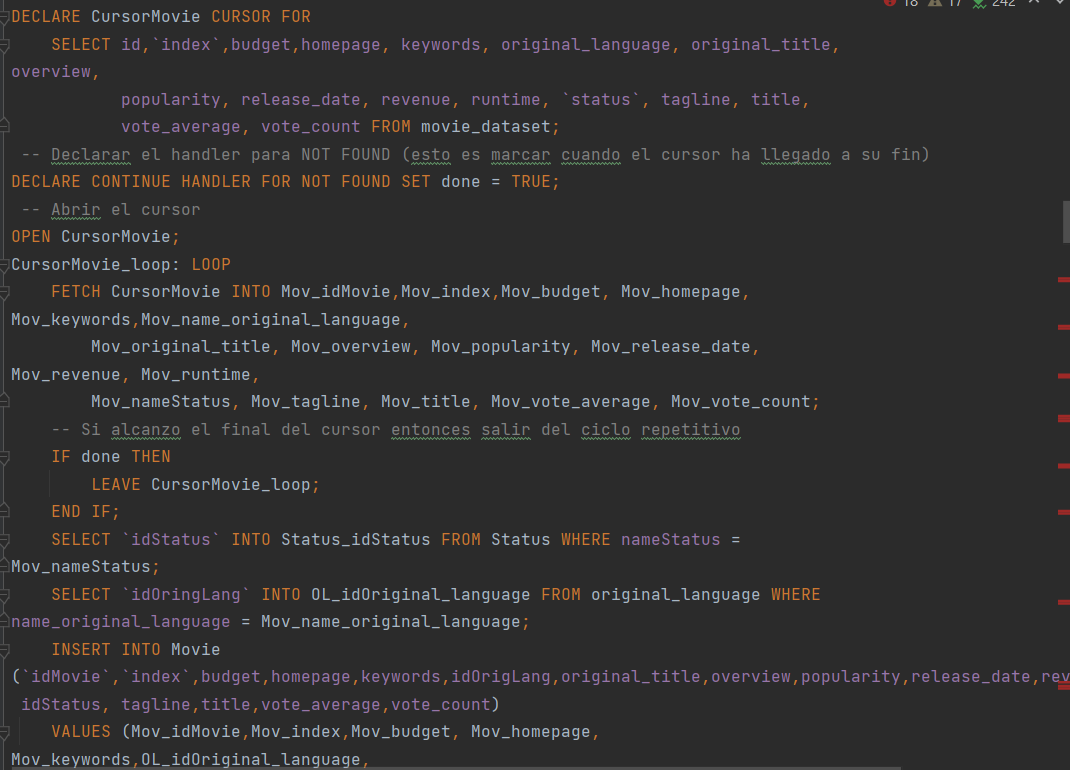
A continuación, cree un cursor llamado "CursorMovie" que seleccione datos de la tabla "movie\_dataset". Los cursores se utilizan para recorrer cada fila de una tabla y extraer valores de cada columna. Se Declara un "manejador" para manejar el caso cuando se alcance el final del puntero.

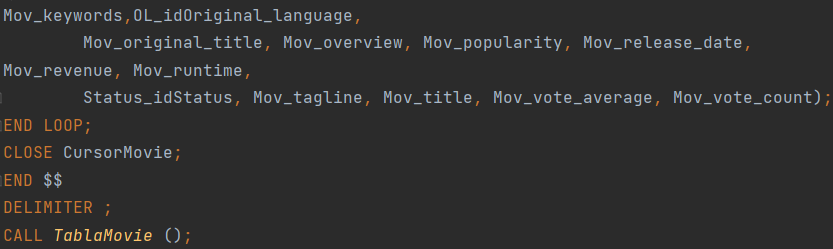
El cursor se abre y "CursorMovie\_loop" comienza a recorrer cada fila de la tabla "movie\_dataset". Dentro del ciclo, la declaración FETCH se usa para recuperar valores de la fila actual y almacenarlos en variables previamente declaradas.

Si se alcanza el final del puntero, el ciclo termina. De lo contrario, comprueba si el nombre del administrador está vacío. Si es así, se le asigna un valor de cero. Ejecuta una consulta para obtener la identificación del director a partir de su nombre y guárdela en la variable "Director\_idDirector".

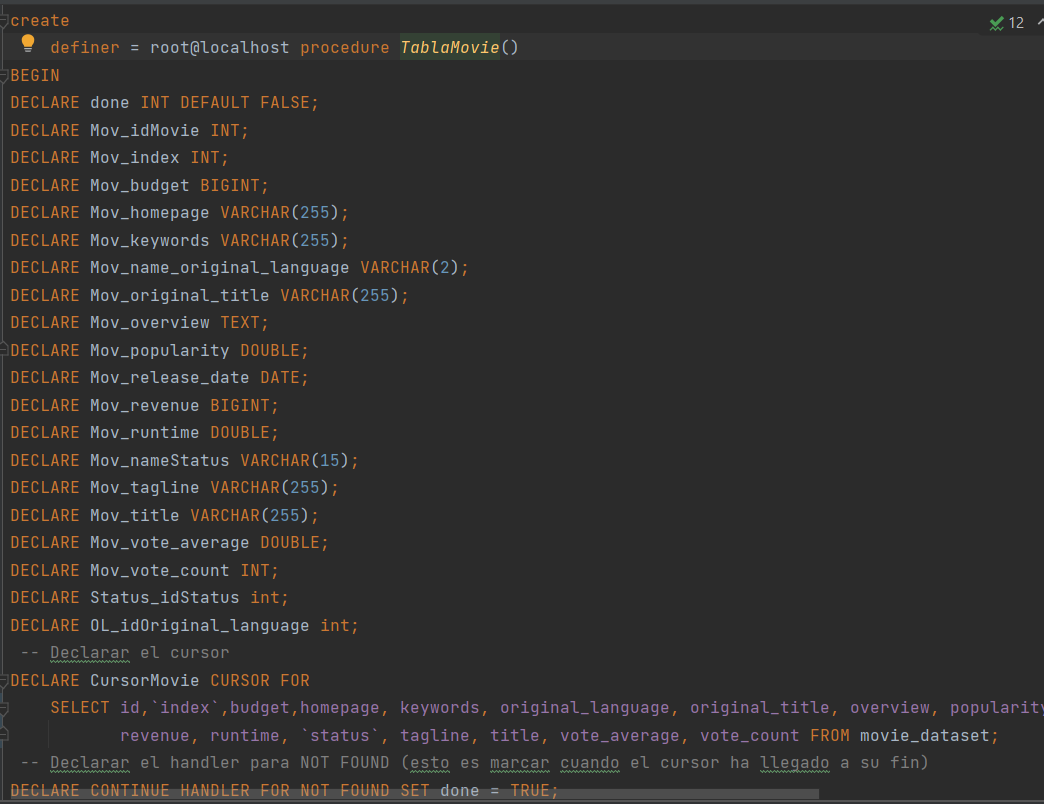
Finalmente, se inserta una nueva fila en la tabla "movies" con el valor obtenido de la tabla "movie\_dataset" y el ID del director. Cuando se completa el ciclo, el cursor se cierra y el procedimiento almacenado finaliza.

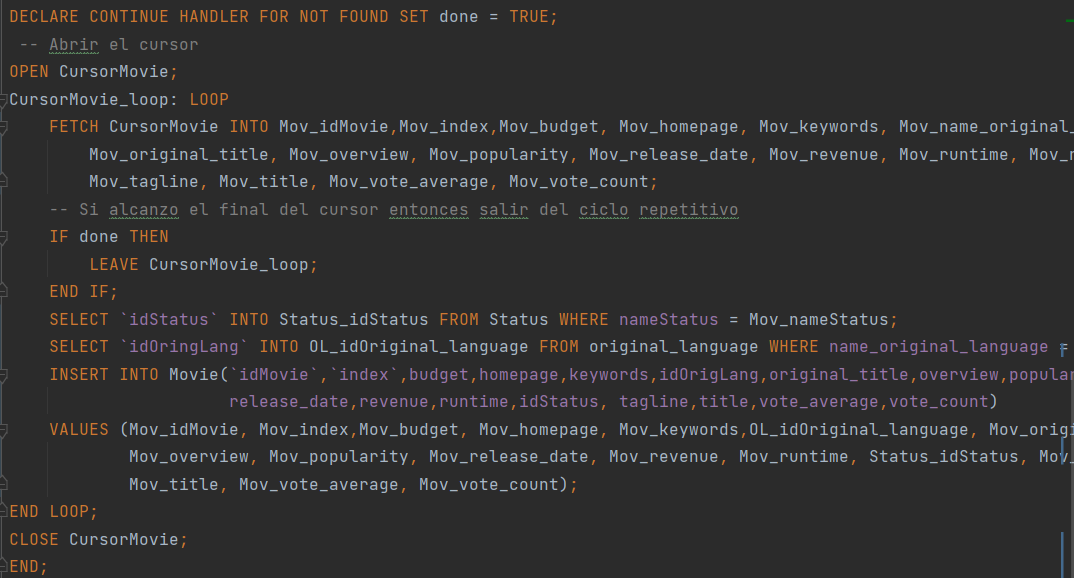






ROUTINE “TablaMovie”





Tenemos tres columnas con valores JSON, para las cuales seguiremos el mismo procedimiento para las tres que son spoken\_languages, production\_companies y production\_countries

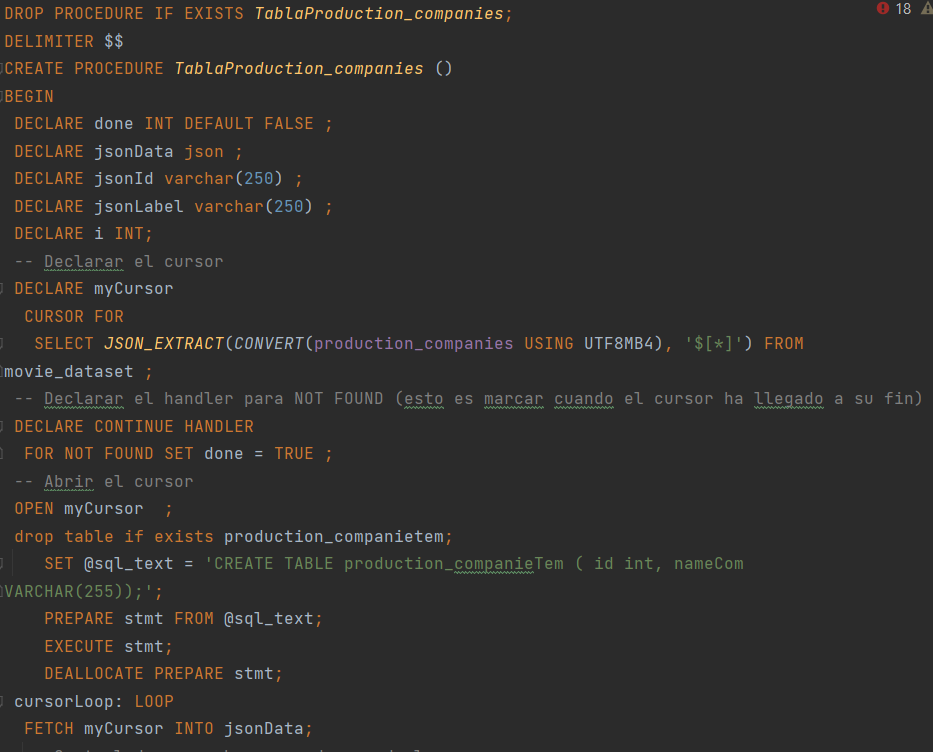
Este código es un procedimiento que se encarga de extraer información de una tabla llamada "movie\_dataset", en particular la columna "production\_companies", que contiene información en formato JSON sobre las compañías productoras de películas.

El procedimiento comienza declarando variables, un cursor que se encarga de recorrer las filas de la tabla "movie\_dataset" y un controlador de eventos "CONTINUE HANDLER" para determinar cuándo se ha alcanzado el final de los datos. Luego, se crea una tabla temporal llamada "production\_companieTem" para almacenar los datos extraídos del formato JSON.

El cursor se usa para recorrer las filas de la tabla "movie\_dataset" y, para cada fila, se extrae la información de las compañías productoras de las películas utilizando la función "JSON\_EXTRACT". Los datos extraídos se insertan en la tabla "production\_companieTem".

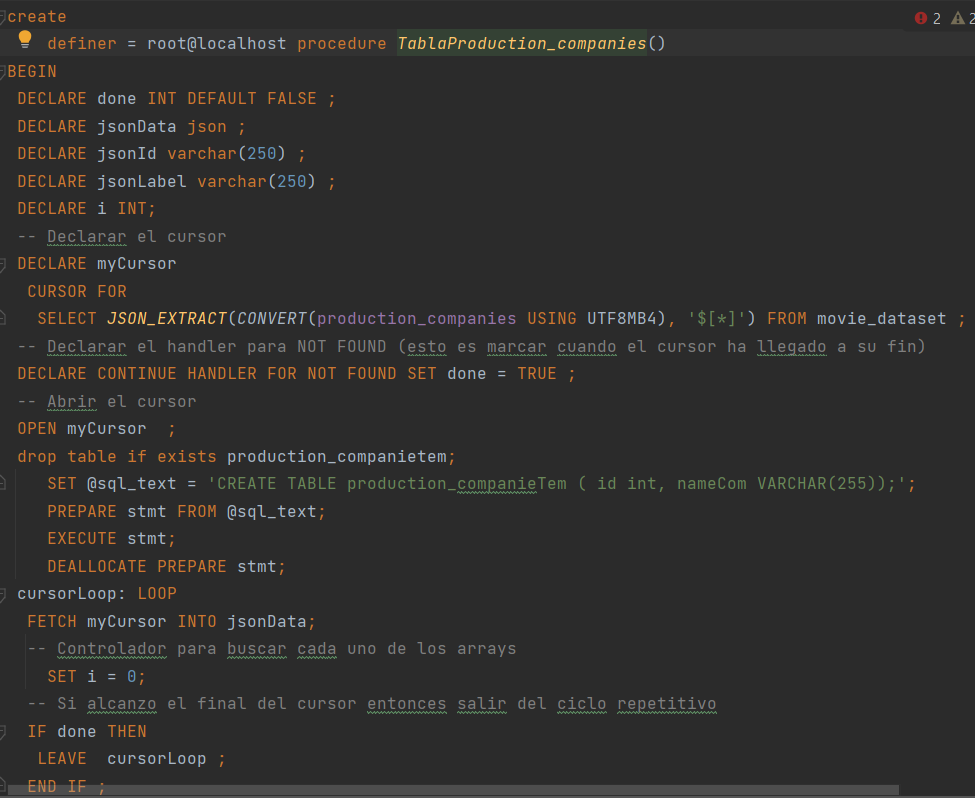
Después de que se han recorrido todas las filas, se seleccionan los datos únicos de la tabla "production\_companieTem" y se insertan en la tabla "production\_companie". Finalmente, se cierra el cursor y se elimina la tabla temporal "production\_companieTem.

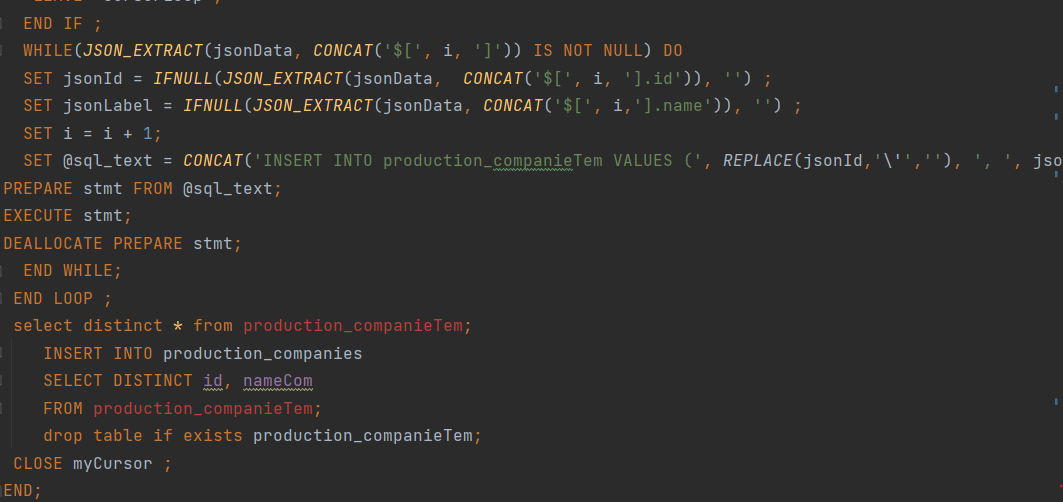
**Población Tabla “Production\_companies”**

****

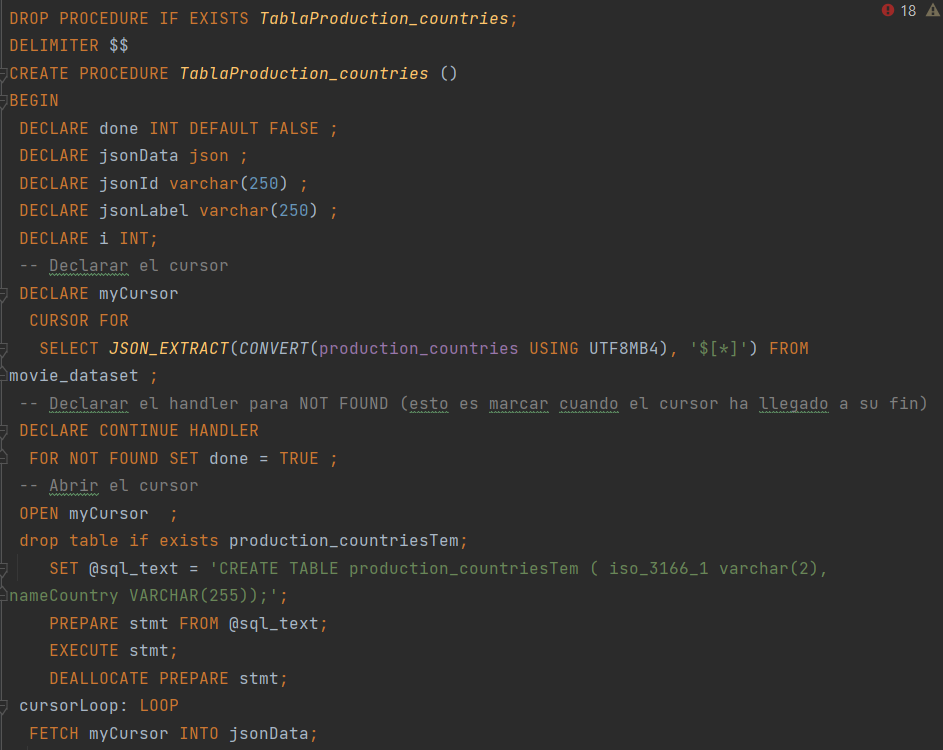
****

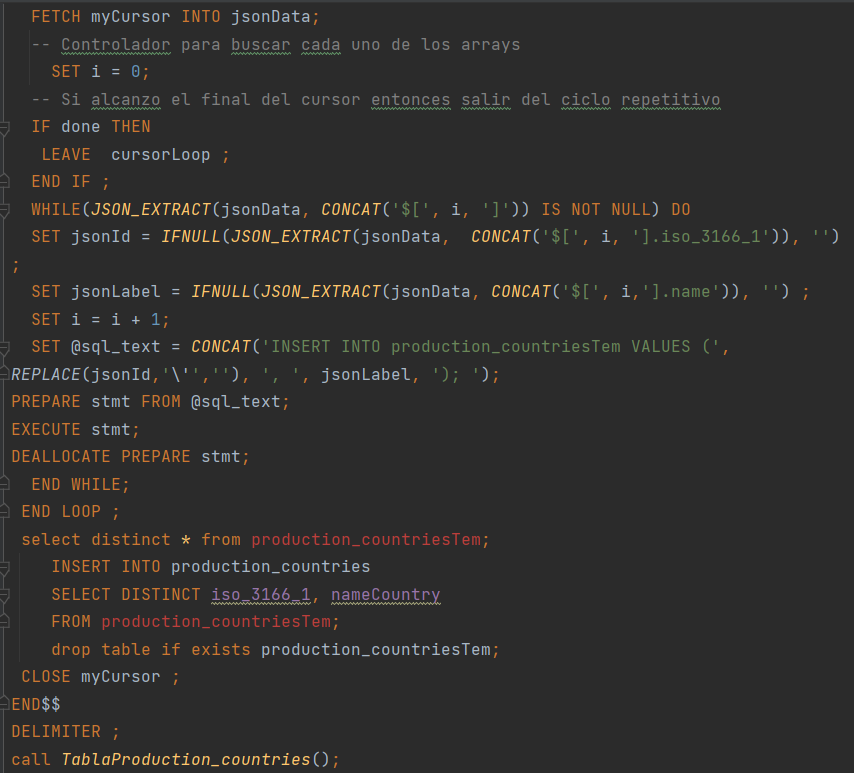
ROUTINE “TablaProduction\_companies”



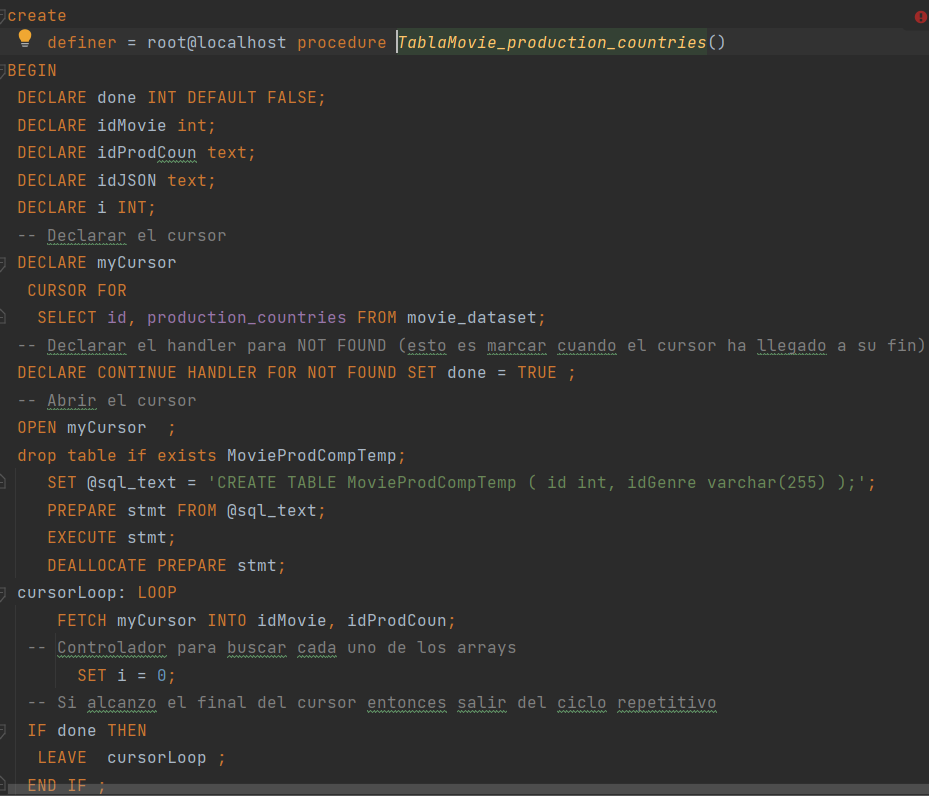


**Población Tabla “Production\_countries”**



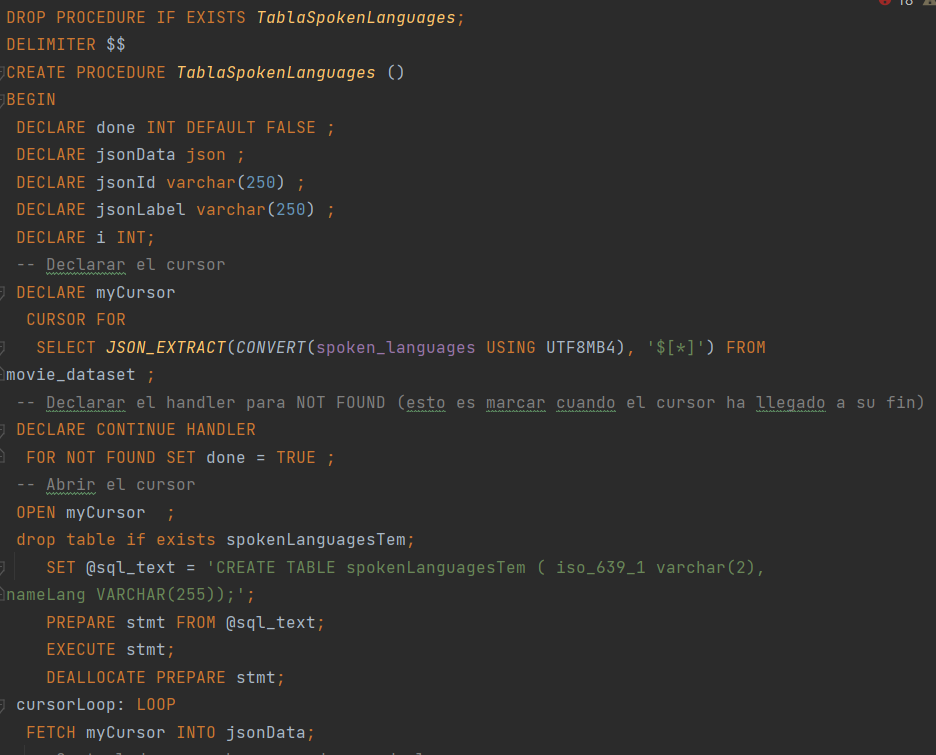


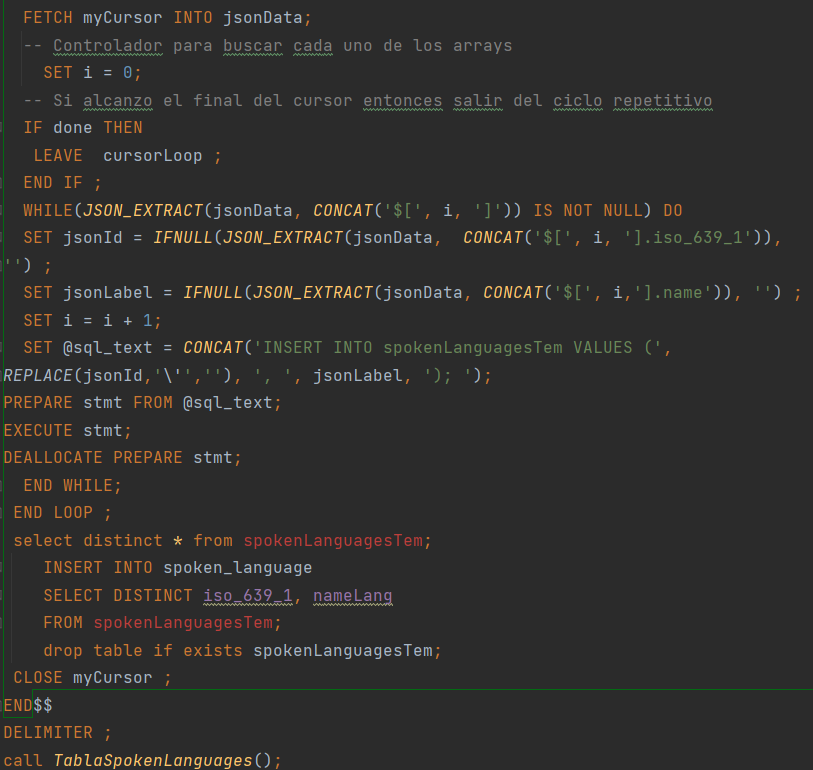
ROUTINE “TablaProduction\_countries”





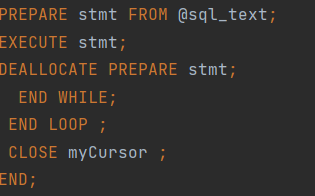
**Población Tabla “Spoken\_languages”**





ROUTINE “TablaSpokenLanguages”





# LIMPIEZA COLUMNA CREW

1. Primero, usemos IntelliJ y el lenguaje de programación Scala para leer el archivo CSV y comenzar a analizar las columnas de Crew. Encontré algunos casos en los que el valor de la clave "Nombre" contenía comillas simples.

2. Analice diferentes patrones en los datos para encontrar cuándo se usan comillas simples, como en O'Brian, o cuándo se usan comillas dobles para encerrar apodos como "D.J." I was. Las comillas son hasta \', por lo que se consideran parte de la cadena de nombre, no JSON.

3. El análisis de los casos muestra que hay casos como e', t' o d' y estos patrones coinciden con las claves 'nombre', 'departamento', 'id\_crédito' o 'id'. Por lo tanto, no era posible reemplazar el valor e', t' o d' con e\', t\' o d\', ya que provocaría un error en la clave.

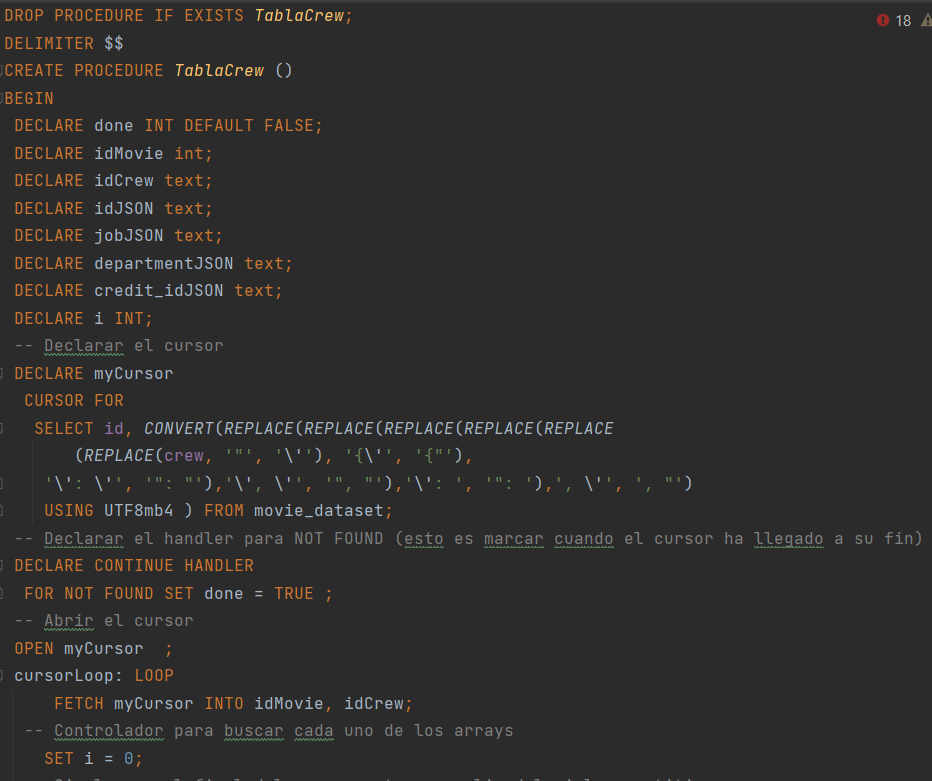
4. Para las intercalaciones 0, 1 y 2, tuve que convertir según la posición y el carácter al lado. Como tal, simplemente ponlo entre comillas.

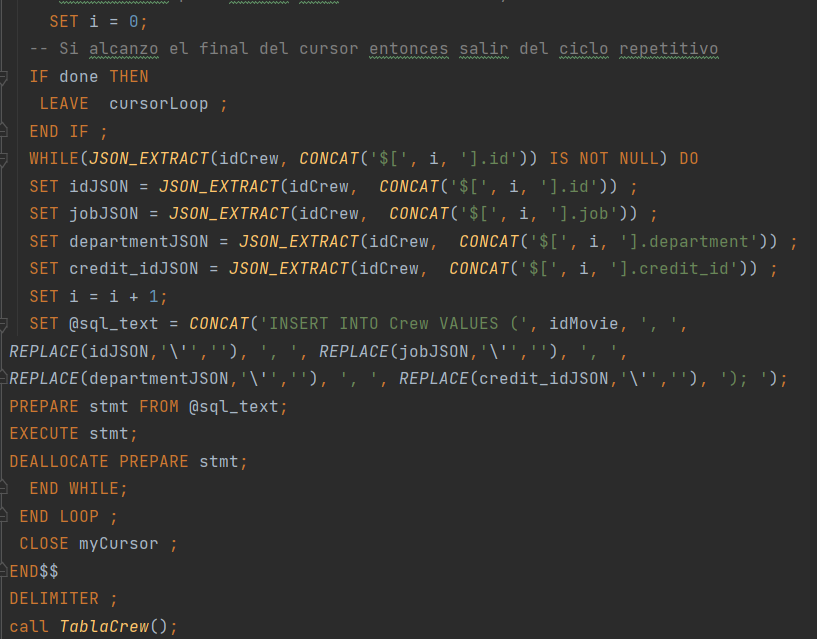
5. El valor clave de "id" también tenía que estar entre comillas simples, así que lo puse de acuerdo con el valor antes y después de "\'id\':", pero se convirtió en "\'id\':\' puse ' entre comillas antes del valor, reemplazó '},' con '\'},' y colocó una comilla simple al final del valor.

6. En el caso del último valor de “id” reemplazamos '}]' al colocar '\"}]' en su lugar. A diferencia del último reemplazo en el punto 5, el} no está seguido por un, debido a que es el último valor del arreglo. Termina con una llave] así que por eso tomamos este caso en consideración para que todos los últimos valores de key tengan comillas cerradas.

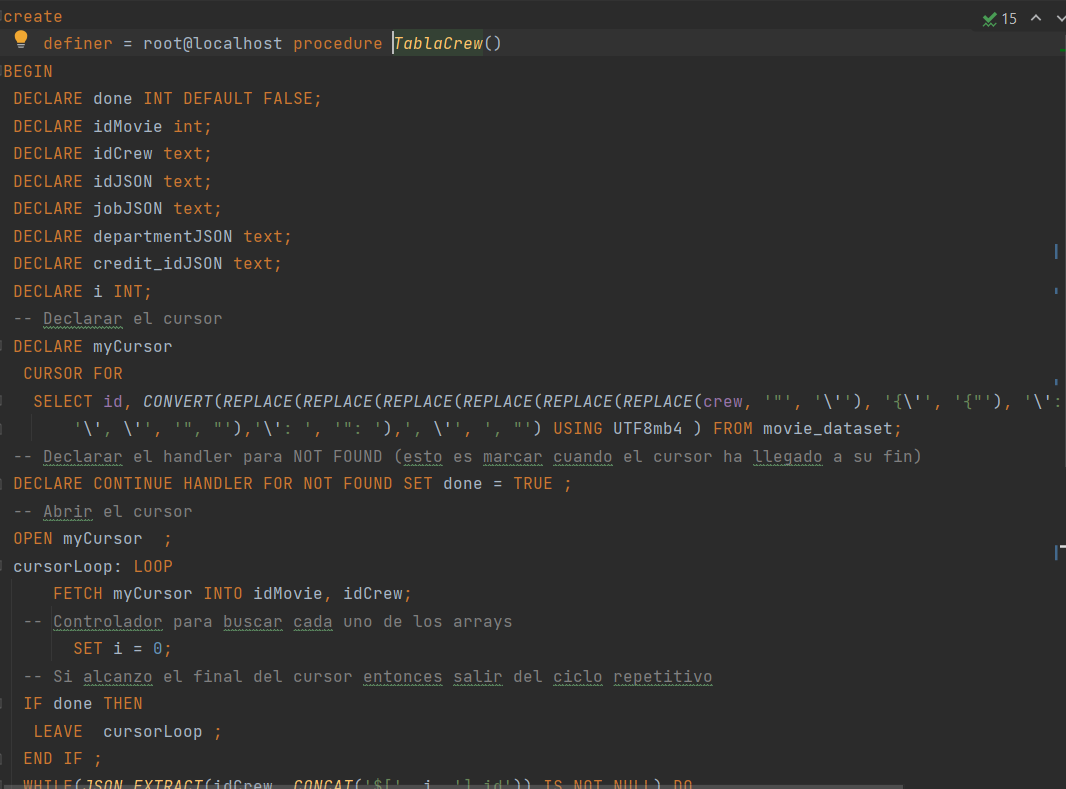
7. Tras realizar todos los REPLACE, realizamos un JSON\_EXTRACT JSON\_EXTRACT (CONVERT(\*SENTENCIA\_DE\_TODOS\_LOS REPLACE\* USING UTF8MB4), '$[\*]') para extraer todos los valores JSON.

8. Hemos realizado exitosamente la limpieza de la columna “crew”. El siguiente paso es realizar le carga de estos datos a su respectiva tabla/s.



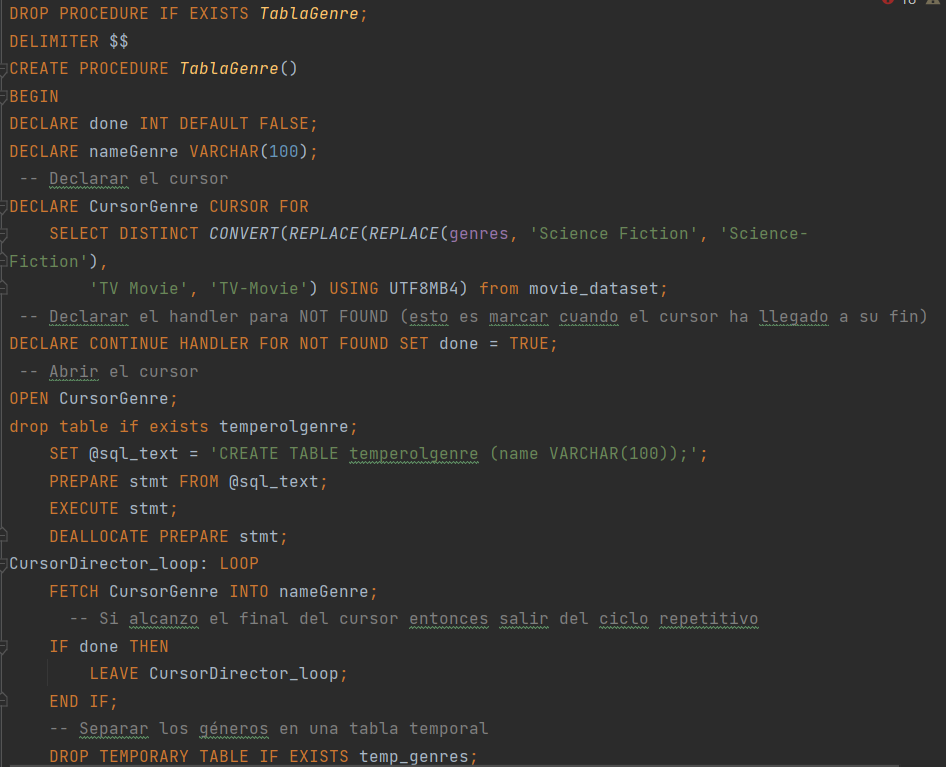


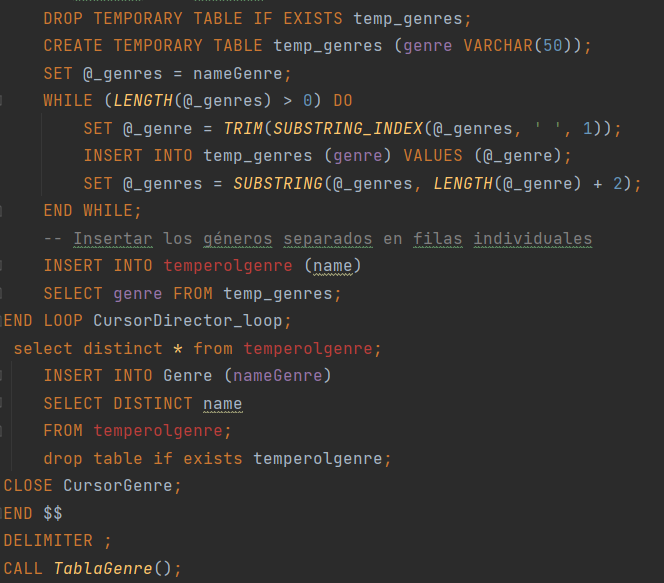
ROUTINE “TablaCrew”





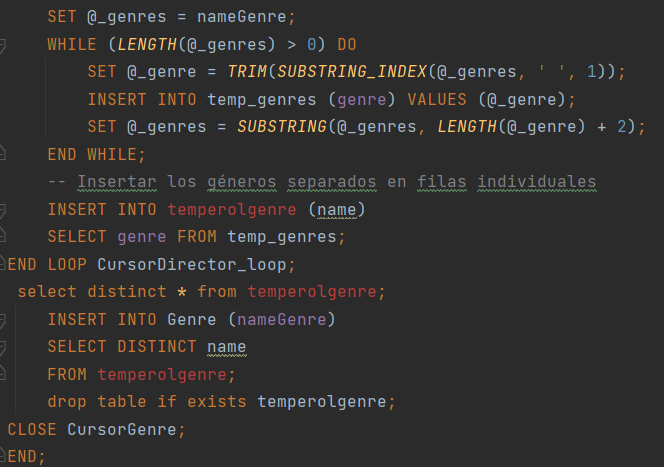
**Población “Genre”**

****

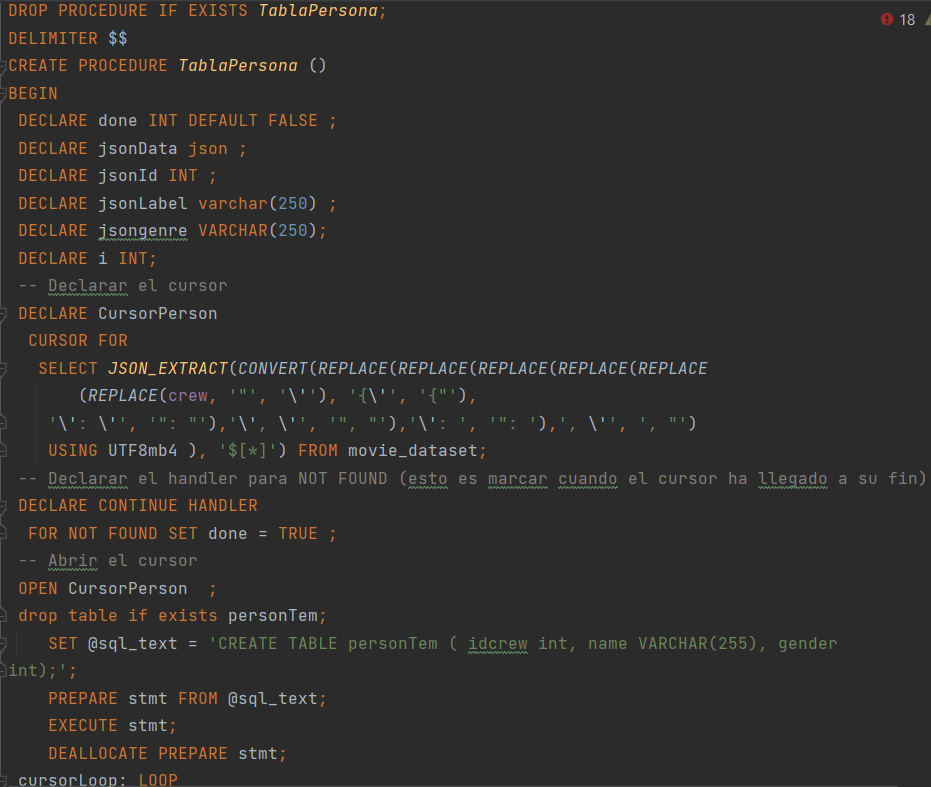
****

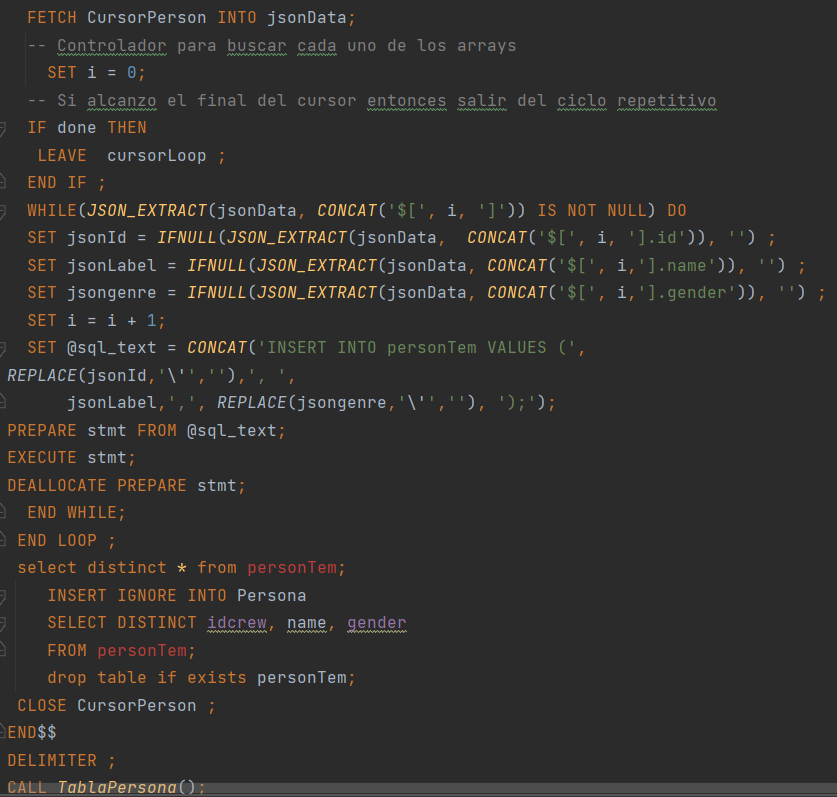
ROUTINE “TablaGenre”



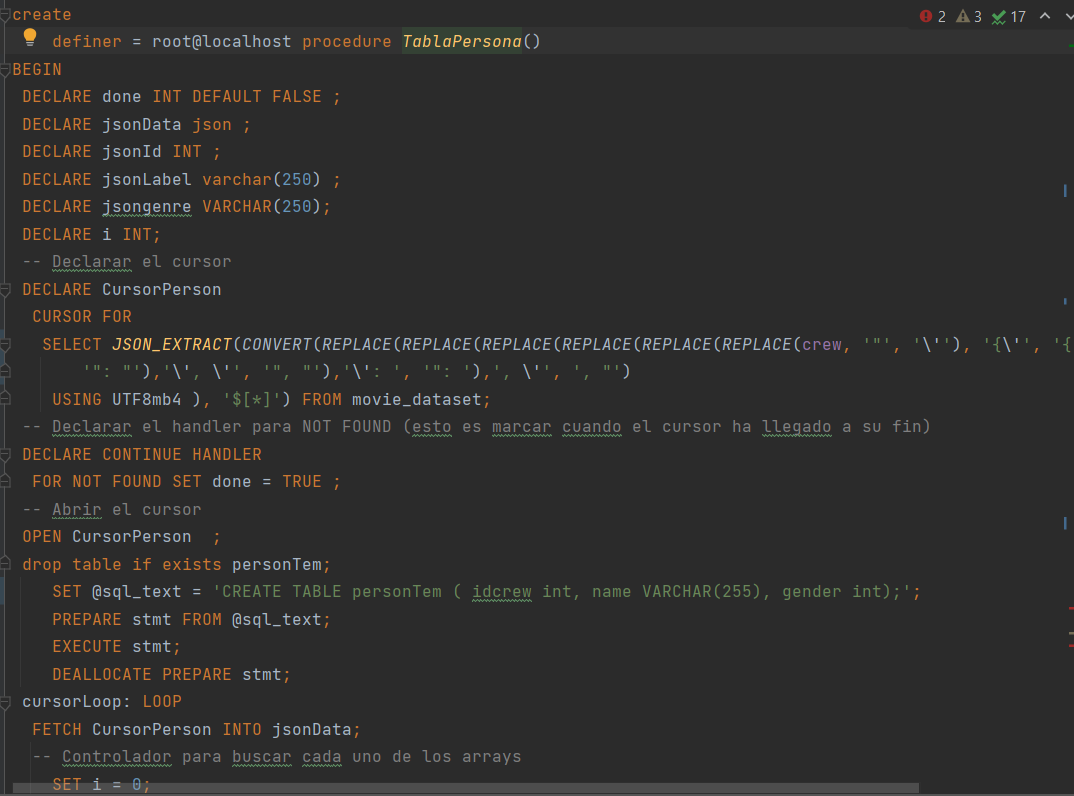


**Población “persona”**

****

****

ROUTINE “TablaPersona”

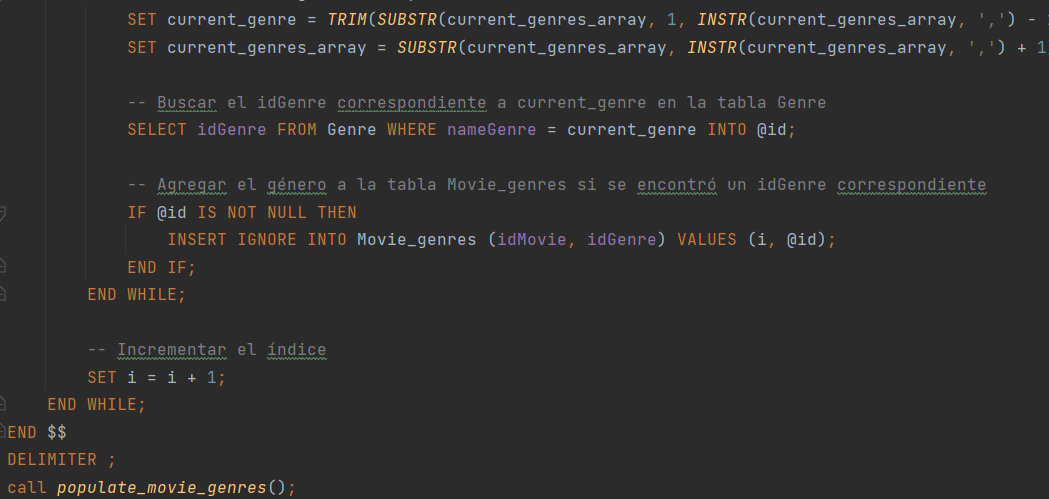




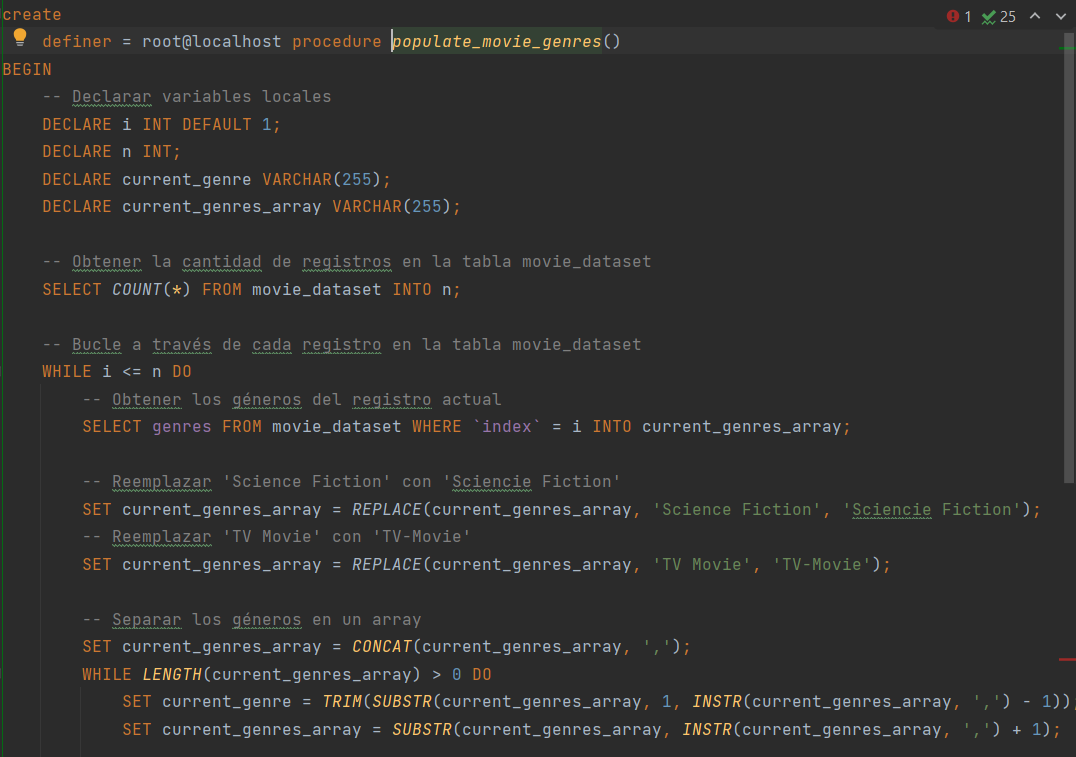
**MUCHOS A MUCHOS**

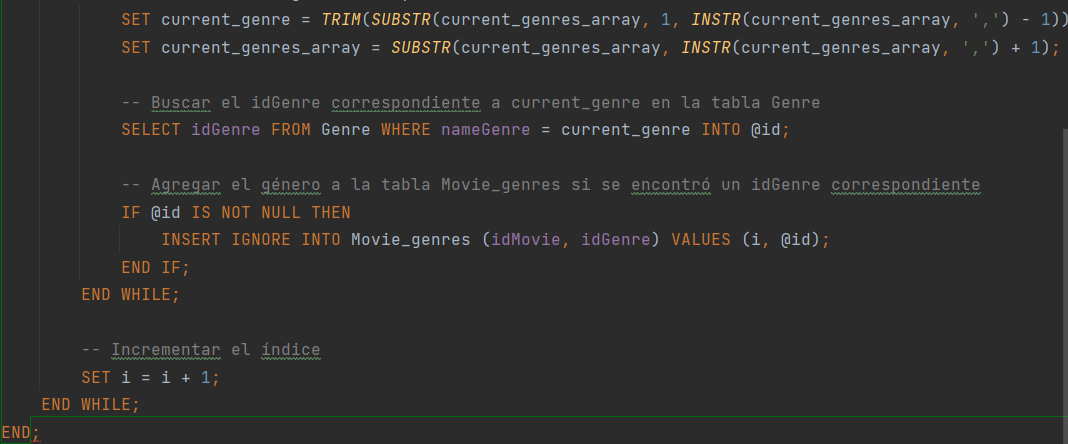
**Población “Movie – Genres”**

****

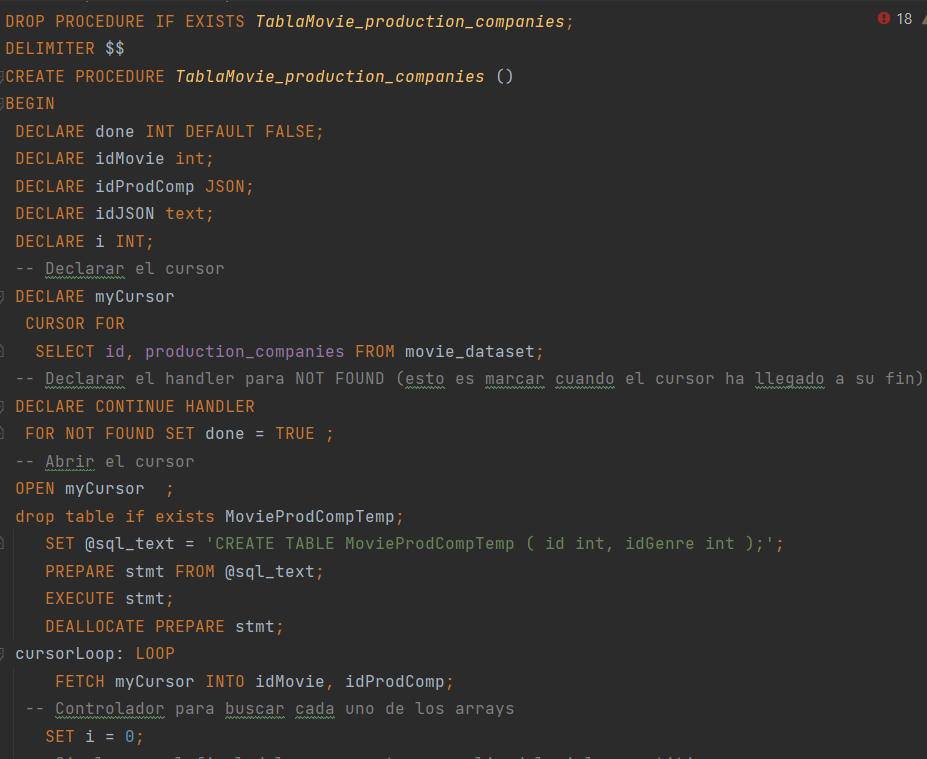
****

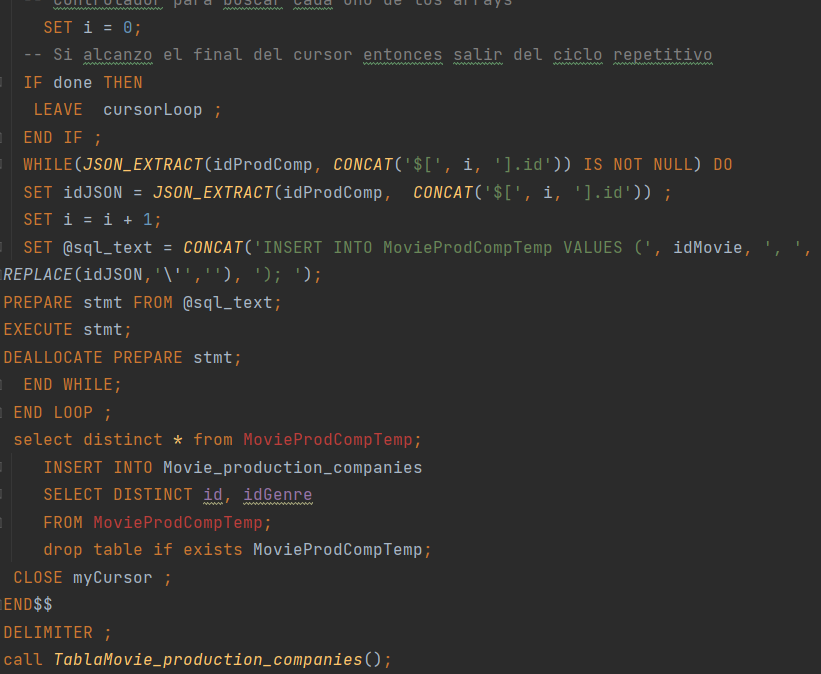
ROUTINE “populate\_movie\_genres”



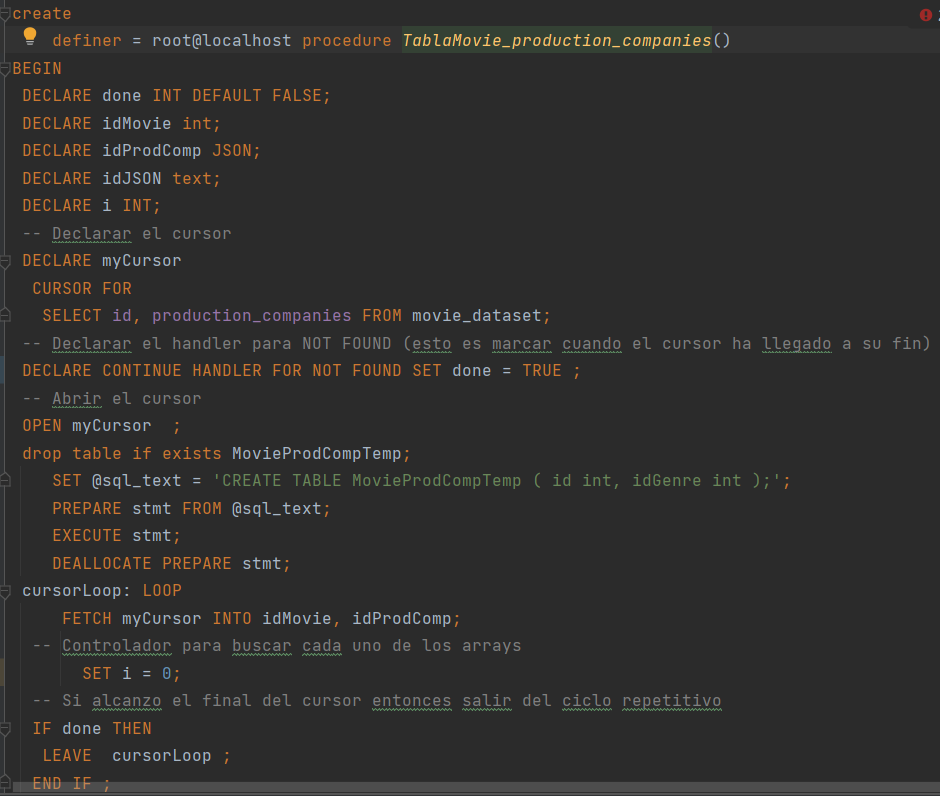


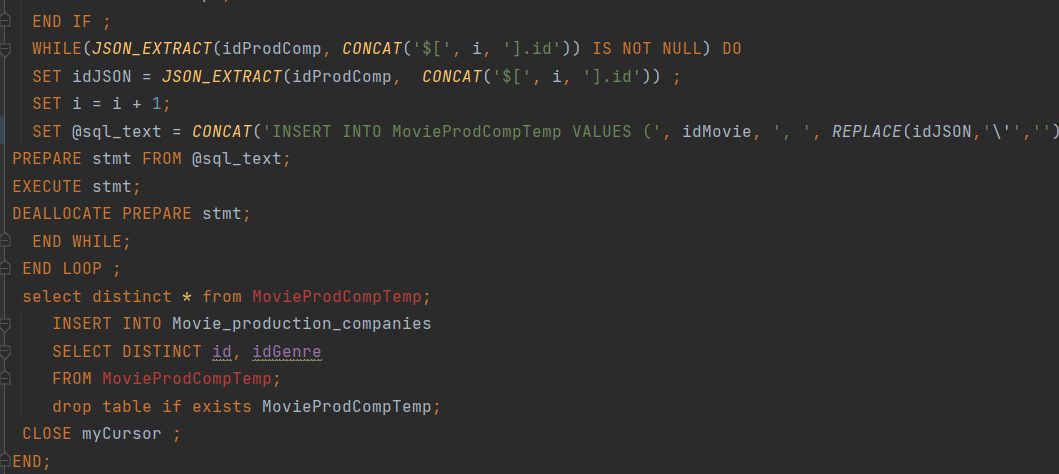
**Población “Movie – Production Companies”**

****

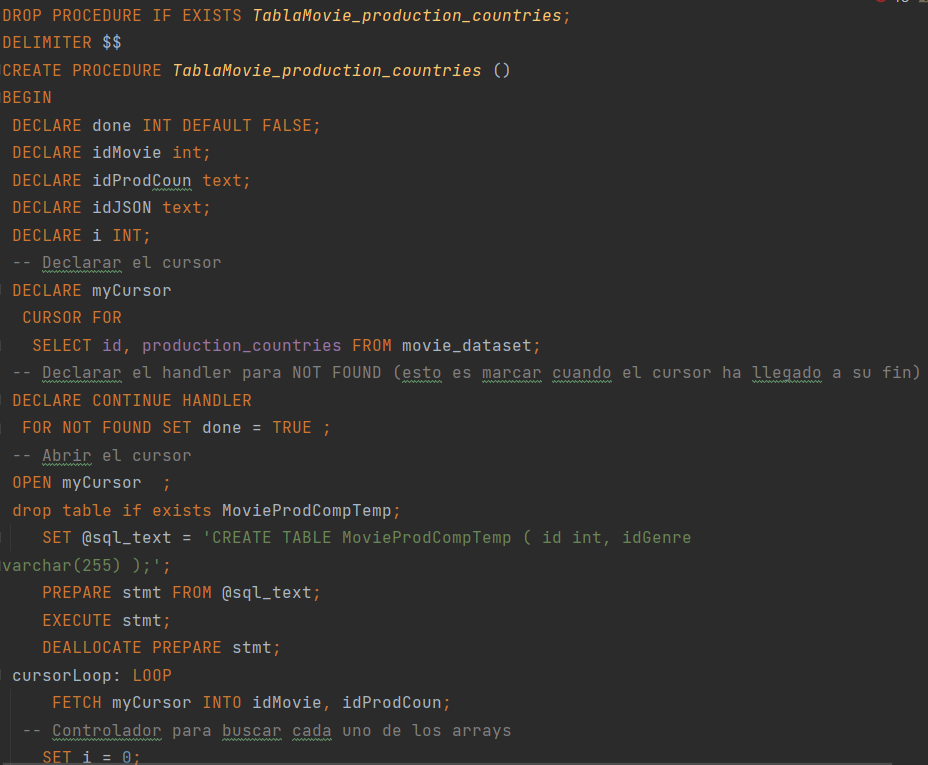
****

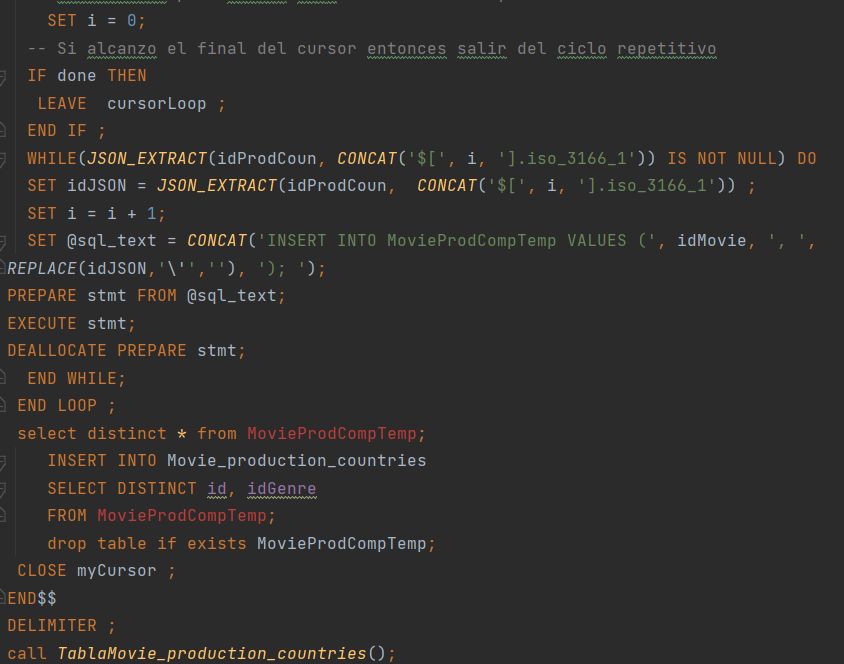
ROUTINE “TablaMovie\_production\_companies”



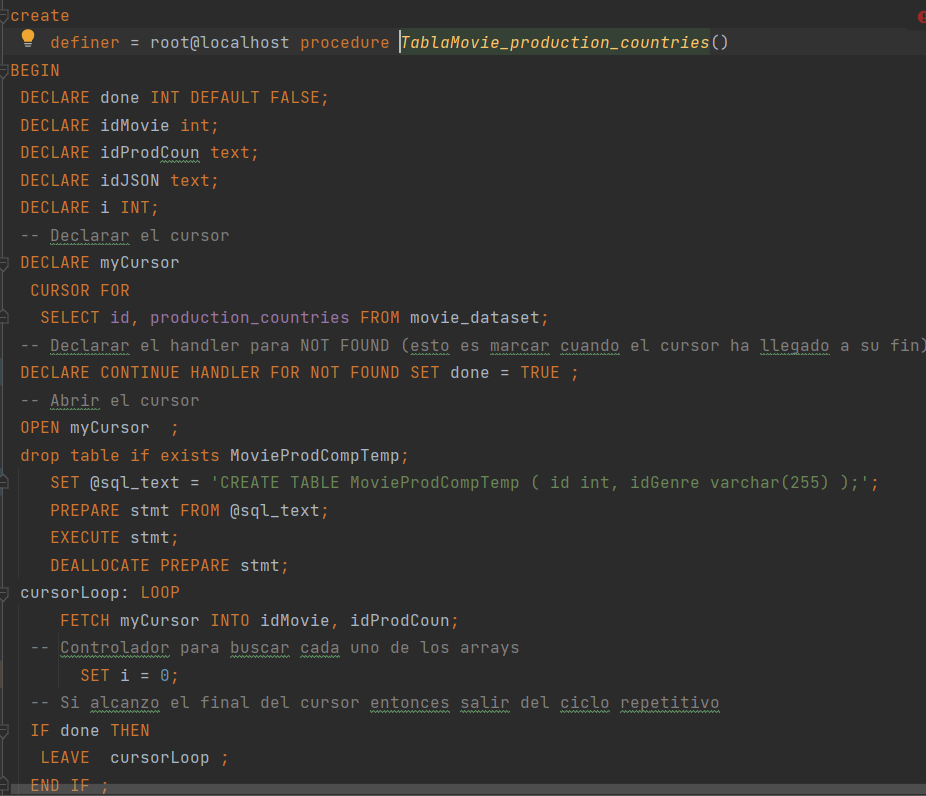


**Población “Movie – Production Countries”**





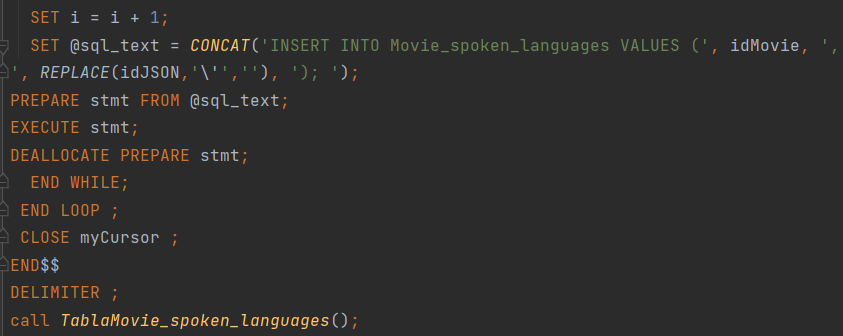
ROUTINE “TablaMovie\_production\_countries”



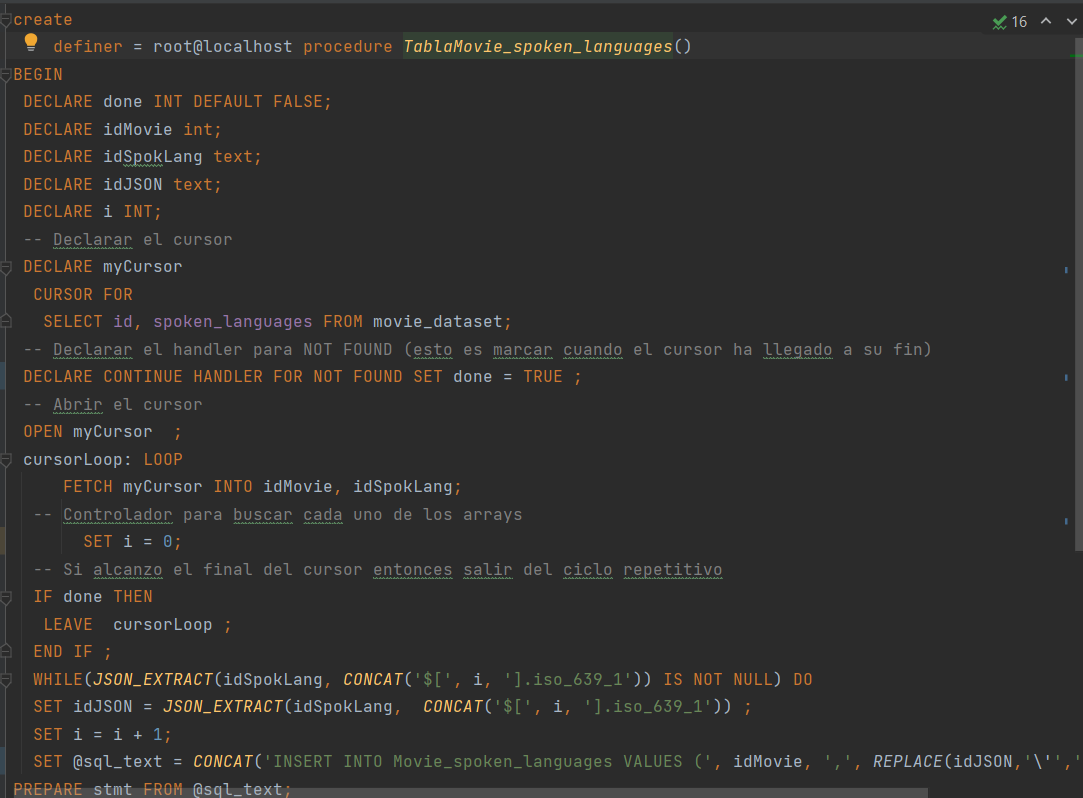


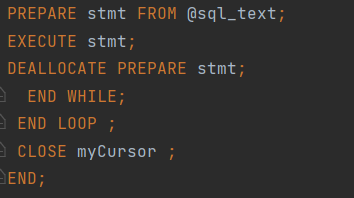
**Población “Movie – Spoken Languages**

****

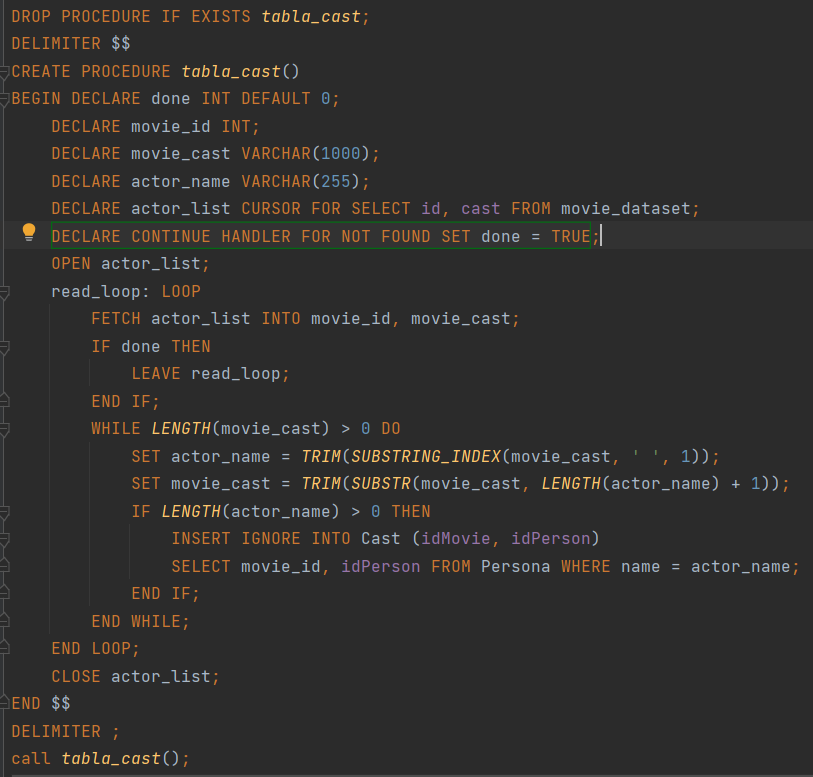
****

ROUTINE “TablaMovie\_spoken\_languages”

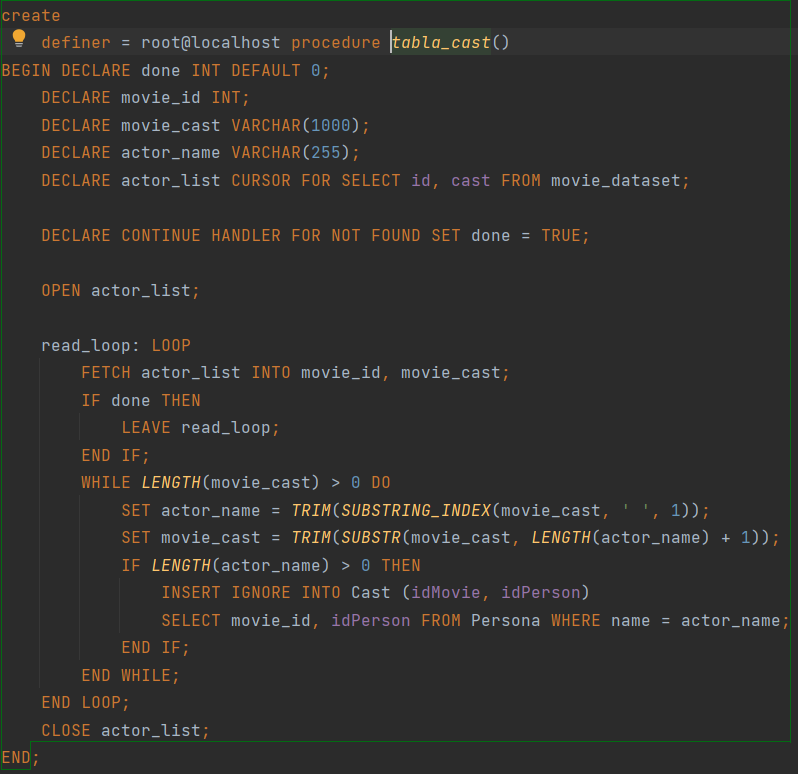




**Población “Cast”**

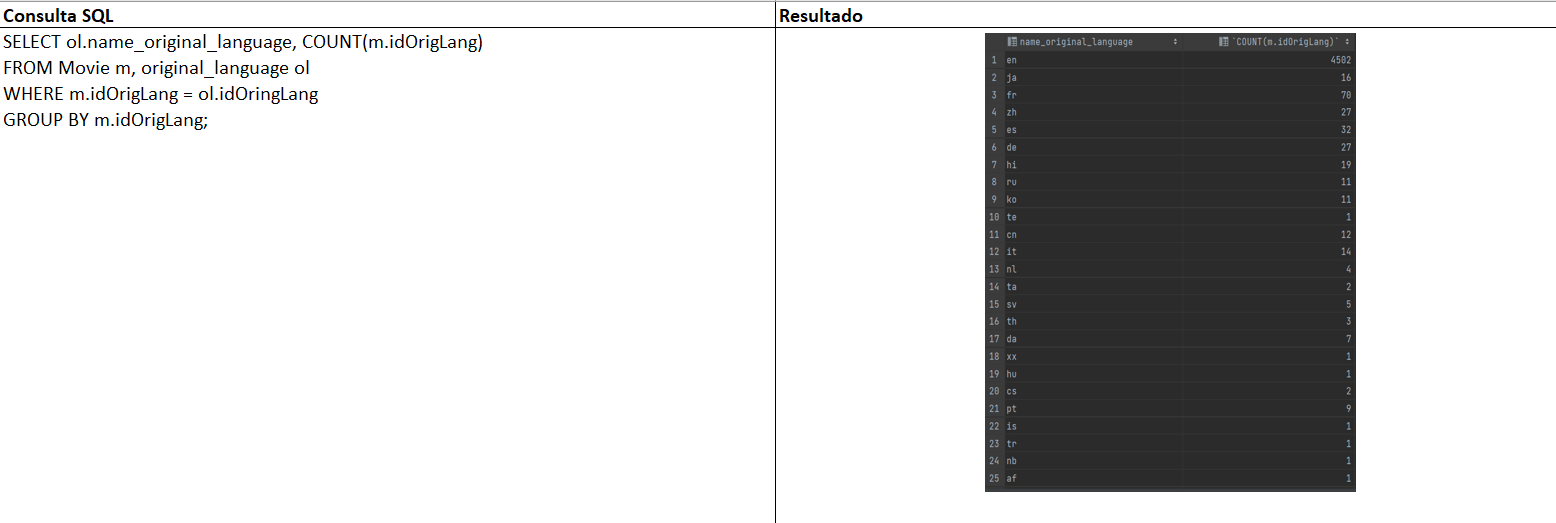


ROUTINE “tabla\_cast”

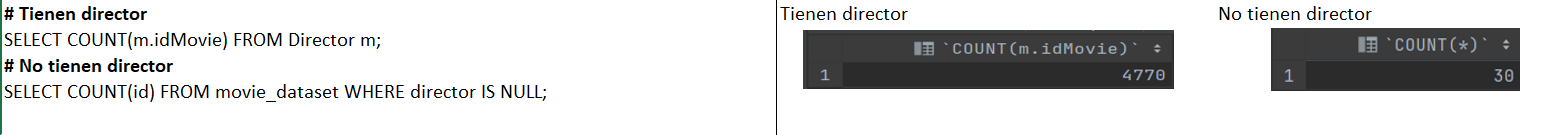


# Consultas.

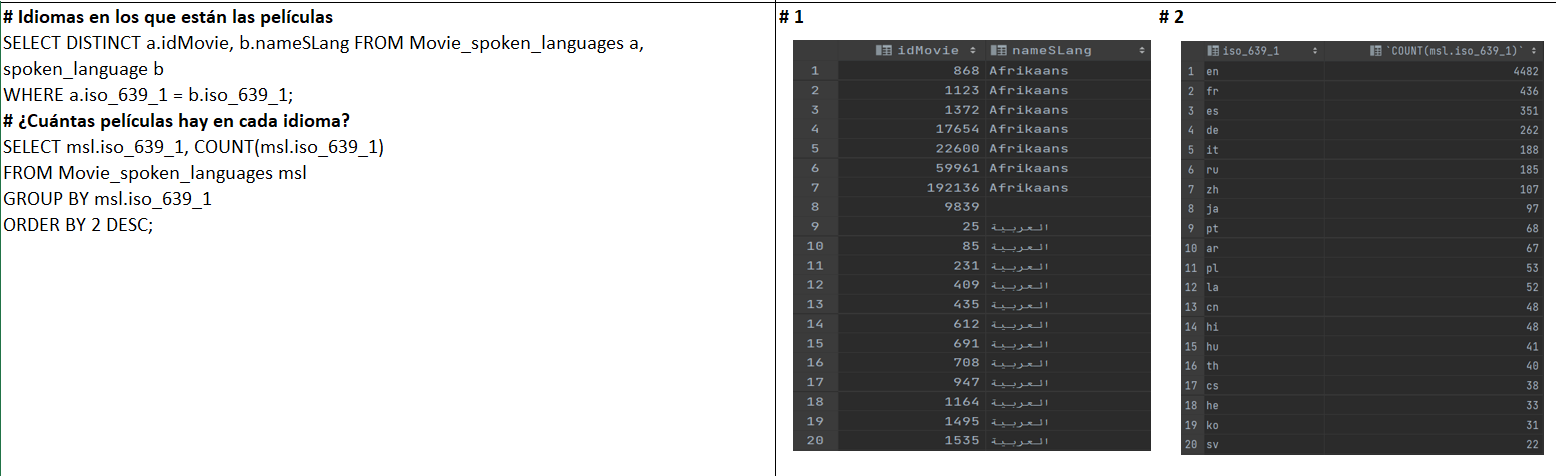
Ejemplo 1: ¿Cuántas películas por idioma tienen su tabla de películas (original language)?



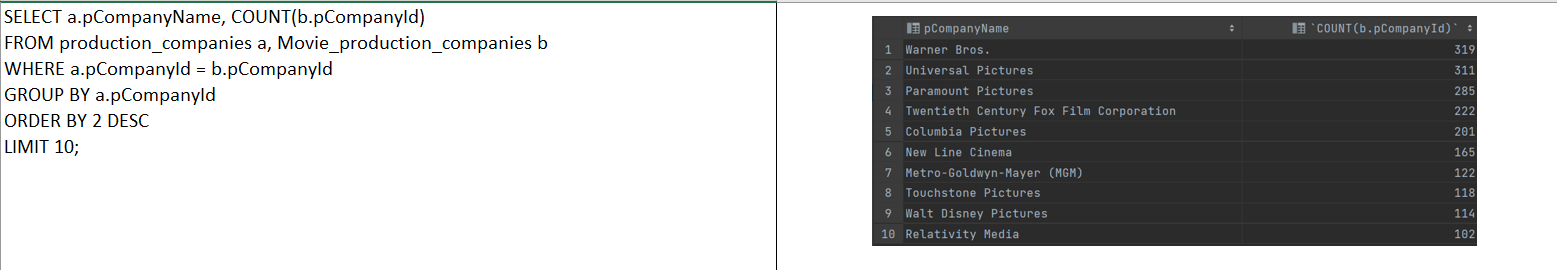
Ejemplo 2: ¿Cuántas películas tienen director y cuántas no?



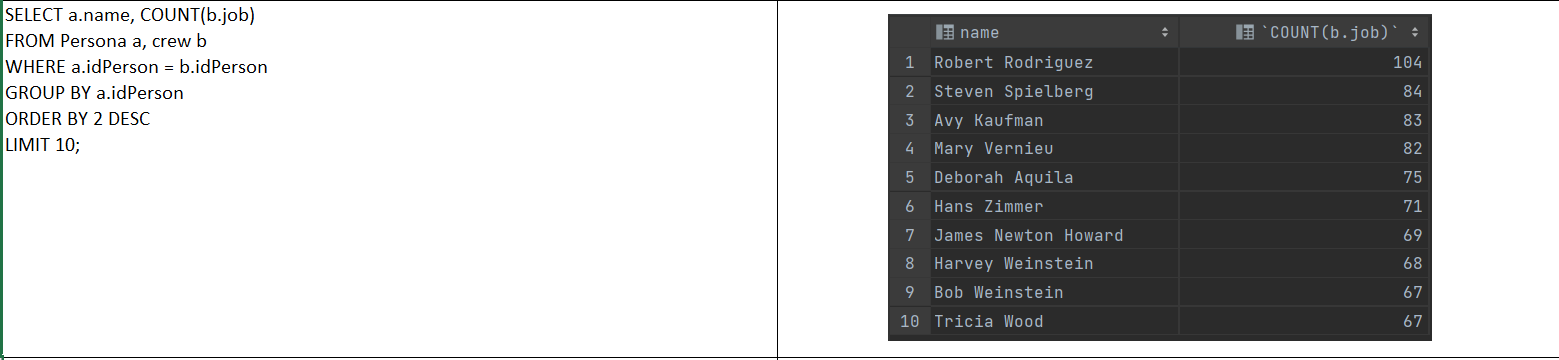
Ejemplo 3: ¿Cuáles son los idiomas en los que están las películas, y cuántas hay en cada lengua (spoken\_languages)?



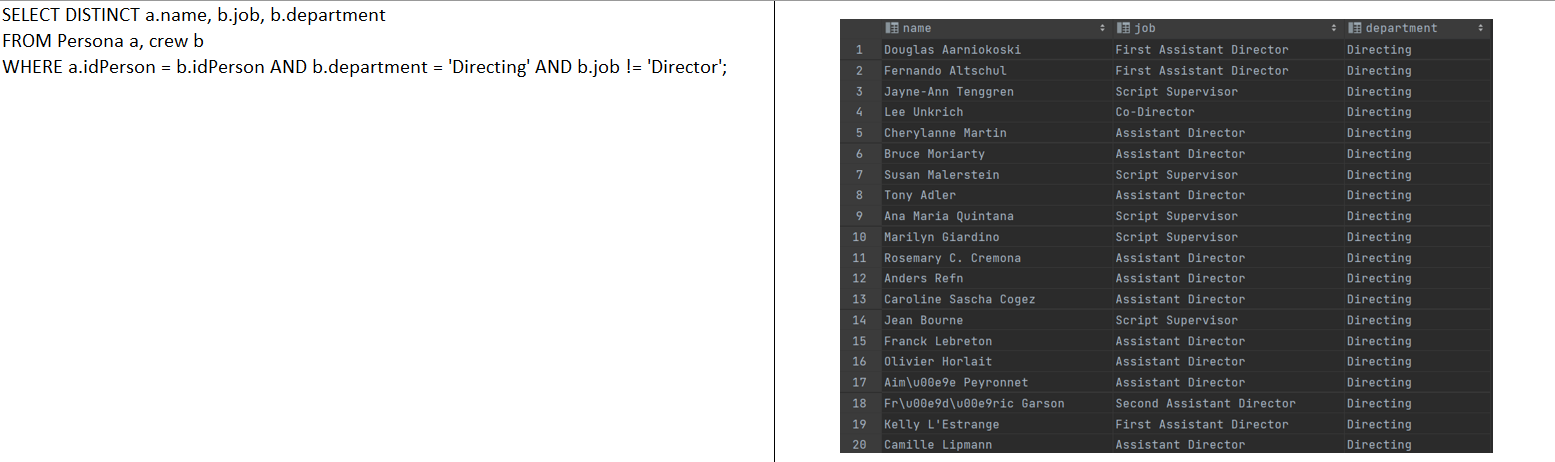
Ejemplo 4: ¿Cuáles son las compañías de producción que tienen el mayor número de películas (production companies)?



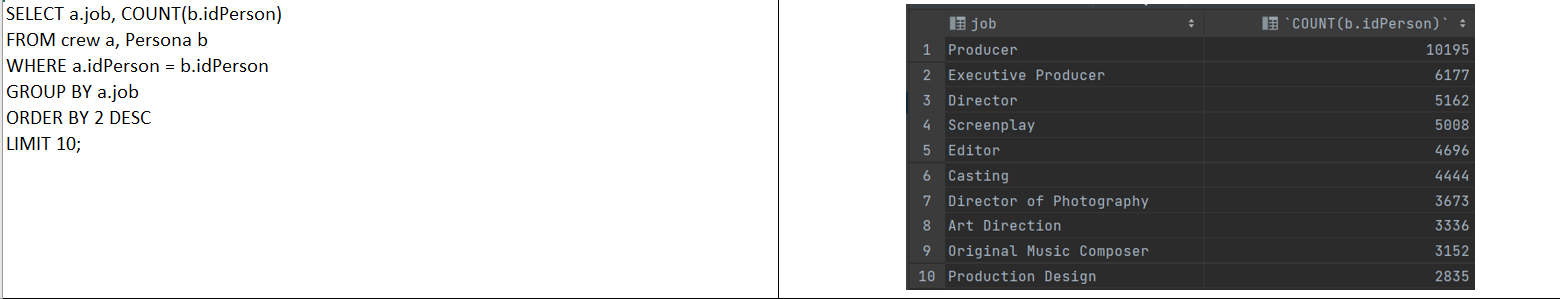
Ejemplo 5: ¿Cuáles son las personas que tienen el mayor número de Jobs (Crew)?



Ejemplo 6: ¿Cuáles son los directores que no constan como director en la Crew?



Ejemplo 7: ¿Cuáles son las jobs qué tienen el mayor número de personas (crew)?



# Conclusiones

Este proyecto se realizó bajo estándares de modelado de bases de datos como normalización, diseño conceptual y lógica. Y tras el arduo trabajo de todos los integrantes del grupo, pudimos lograr todos los objetivos planteados al inicio del desarrollo. Tuvimos que afrontar varios retos a lo largo del proyecto. Sin embargo, la guía de los profesores acompañantes y nuestra creatividad en La resolución de problemas nos permitió cumplir con los requisitos establecidos.

Todos los pasos apropiados para completar el proyecto Como resultado, hemos logrado un progreso significativo en el modelado y la toma de decisiones para las entradas de la base de datos, y hemos aprendido nuevas estrategias de programación que se pueden aplicar a situaciones reales. -casos del mundo. Desarrollar este proyecto definitivamente lo acercará un paso más a convertirse en un profesional seguro cuando trabaje en proyectos futuros.

He aprendido que los datos deben procesarse correctamente antes de cada operación para mantener la coherencia y evitar errores. Además, entendemos la importancia de las herramientas a la hora de trabajar con bases de datos. Y puedo concluir que todo lo que he visto en este ciclo es muy importante ya que se utilizó en este proyecto de integración, fíjate que fue un reto para nuestros estudiantes.