



PROYECTO FINAL

DIPLOMATURA UNIVERSITARIA EN
BASE DE DATOS – SQL DESDE CERO
SEGUNDA COHORTE

ÁLVARO JOSÉ GUZMÁN RENGIPO - 2023

CONTENIDO

Este proyecto incluye:

1. OBJETIVO DEL PROYECTO
2. PRESENTACIÓN DEL PROYECTO
3. DIAGRAMA ENTIDAD RELACION
4. TABLAS DEL PROYECTO
5. VISTAS DEL PROYECTO
6. FUNCTIONS
7. STORED PROCEDURES
8. TRIGGERS
9. USUARIOS DEL PROYECTO – DLC
10. BACKUP
11. POWER BI
12. INFORMACION DE CONTACTO



01

OBJETIVO

BASE DE DATOS – SQL DESDE CERO

CREAR UNA BASE DE DATOS PARA ORGANIZAR Y GESTIONAR LA INFORMACIÓN RELEVANTE COMO TITULO, GENERO Y FECHAS DE ESTRENO. PARA FACILITAR SU ACCESO Y PERMITIR REALIZAR ANALISIS, OFRECER RECOMENDACIONES Y BRINDAR INFORMACIÓN PARA PLATAFOMAS DE STREAMING.

BASE DE DATOS – SQL DESDE CERO



02

PRESENTACIÓN

BASE DE DATOS – SQL DESDE CERO

PRESENTACIÓN DEL PROYECTO

Mi propuesta para el proyecto de la Base de datos es hacerla sobre la información de las películas que pueda contener alguna plataforma de streaming, como son Netflix, Disney+, HBO Max, Crunchyroll, etc. En términos generales esta base de datos va a ser una colección organizada de información sobre películas, actores, directores, géneros, estrenos, calificaciones y otros detalles relacionados con las películas. La cual puede ser utilizada por sitios web de cine, aplicaciones móviles, tiendas de video, bibliotecas de medios y otros sitios similares para proporcionar información detallada sobre las películas. Donde el usuario, a través de su cuenta, pueda visualizar cualquiera de las películas disponibles en la plataforma, conocer los nuevos estrenos, crear listas personalizadas de sus películas favoritas, conocer información de las mismas, buscarlas por su título, director, actores, compañía, genero, etc. Esta base de datos de películas va a ser una herramienta útil para cualquier persona interesada en el cine, ya que proporciona información detallada y actualizada sobre una amplia variedad de películas.



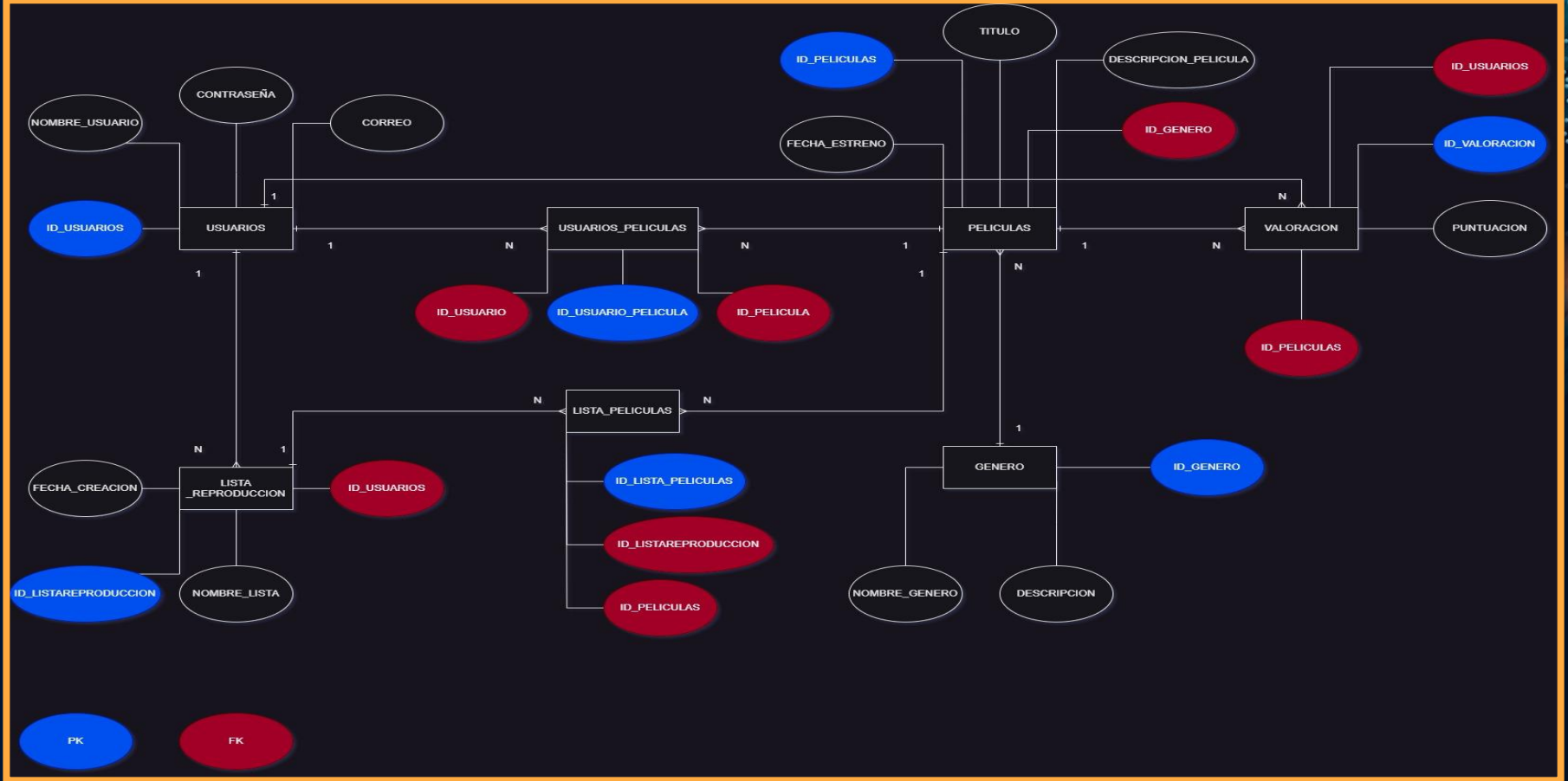


03

DIAGRAMA ENTIDAD-RELACIÓN

BASE DE DATOS – SQL DESDE CERO

DER





04

TABLAS

BASE DE DATOS – SQL DESDE CERO

TABLA USUARIOS: almacena los datos de los usuarios.

KEY	Column	Type	Not null	Unique	Len	Notes
PK	ID_usuarios	INT	NOT NULL	UNIQUE		Identificacion de los usuarios
	nombre_usuario	VARCHAR	NOT NULL		30	Nombre del usuario
	contraseña	VARCHAR	NOT NULL		50	Contraseña del usuario
	correo	VARCHAR	NOT NULL		50	Correo electronico del usuario

TABLA PELICULAS: almacena la información de las películas.

KEY	Column	Type	Not null	Unique	Len	Notes
PK	ID_peliculas	INT	NOT NULL	UNIQUE		Identificacion de peliculas
	fecha_estreno	DATE	NULL			Indica la fecha de estreno de la pelicula
	titulo	VARCHAR	NOT NULL		30	Indentifica el titulo de la pelicula
	descripcion_pelicula	TEXT	NULL		300	Indica una descripcion de la pelicula
FK	ID_genero	INT	NOT NULL			Identificacion del genero de las peliculas

TABLA USUARIOS-PELICULAS: tabla intermedia por la relación entre las tablas usuarios y películas.

KEY	Column	Type	Not null	Unique	Len	Notes
PK	ID_usuario_pelicula	INT	NOT NULL	UNIQUE		ID de Tabla Intermedia
FK	ID_usuario	INT	NOT NULL			Identificacion de los usuarios
FK	ID_peliculas	INT	NOT NULL			Identificacion de peliculas

TABLA VALORACION: contiene informacion de las valoraciones de las películas.

KEY	Column	Type	Not null	Unique	Len	Notes
PK	ID_valoracion	INT	NOT NULL	UNIQUE		Identificacion de la valoracion de peliculas
	puntuacion	INT	NOT NULL			Indica la puntuacion dada a las peliculas
FK	ID_usuarios	INT	NOT NULL			Identificacion de los usuarios
FK	ID_peliculas	INT	NOT NULL			Identificacion de peliculas

TABLA GENERO: almacena el genero de las películas.

KEY	Column	Type	Not null	Unique	Len	Notes
PK	ID_genero	INT	NOT NULL	UNIQUE		Identificacion del genero de la pelicula
	nombre_genero	TEXT	NOT NULL		50	Indica el nombre del genero de la pelicula
	descripcion	TEXT	NULL		300	Indica la descripcion de genero de la pelicula

TABLA LISTA_REPRODUCCION: almacena la lista de reproducción de los usuarios.

KEY	Column	Type	Not null	Unique	Len	Notes
PK	ID_listareproduccion	INT	NOT NULL	UNIQUE		Identificacion de la lista de reproduccion del usuario
	nombre_lista	VARCHAR	NOT NULL		50	Nombre de la lista de reproduccion
	fecha_creacion	DATE	NULL			fecha de creacion de la lista de reproduccion
FK	ID_usuarios	INT	NOT NULL			Legajo del Profesor

TABLA LISTA_PELICULAS: tabla intermedia por la relacion entre las tablas lista_reproduccion y pelicula.

KEY	Column	Type	Not null	Unique	Len	Notes
PK	ID_usuarios	INT	NOT NULL	UNIQUE		Identificacion de los usuarios
	nombre_usuario	VARCHAR	NOT NULL		30	Nombre del usuario
	contraseña	VARCHAR	NOT NULL		50	Contraseña del usuario
	correo	VARCHAR	NOT NULL		50	Correo electronico del usuario



SCRIPTS DE CREACIÓN DE LAS TABLAS Y RELACIONES FK

```
• create table Usuarios (  
    ID_usuarios int not null,  
    nombre_usuario varchar(30) not null,  
    contraseña varchar(50) not null,  
    correo varchar(50) not null,  
    primary key (ID_usuarios)  
);
```

```
• create table Peliculas(  
    ID_peliculas int not null,  
    fecha_estreno date,  
    titulo varchar(30) not null,  
    descripcion_pelicula text(300),  
    ID_genero int not null,  
    primary key (ID_peliculas)  
);
```

```
• create table Usuarios_Peliculas(  
    ID_usuario_pelicula int not null auto_increment,  
    ID_usuarios int not null,  
    ID_peliculas int not null,  
    primary key (ID_usuario_pelicula)  
);
```

```
• create table Valoracion(  
    ID_valoracion int not null,  
    puntuacion int not null,  
    ID_usuarios int not null,  
    ID_peliculas int not null,  
    primary key (ID_valoracion)  
);
```

```
• create table Genero(  
    ID_genero int not null,  
    nombre_genero text(50) not null,  
    descripcion text(300) null,  
    primary key (ID_genero)  
);
```

```
• create table Lista_Reproduccion(  
    ID_listareproduccion int not null,  
    nombre_lista varchar(50) not null,  
    fecha_creacion date,  
    ID_usuarios int not null,  
    primary key (ID_listareproduccion)  
);
```

```
• create table lista_peliculas(  
    ID_lista_peliculas int not null auto_increment,  
    ID_listareproduccion int not null,  
    ID_peliculas int not null,  
    primary key (ID_lista_peliculas)  
);
```

```
alter table peliculas add constraint fk_genero foreign key (ID_genero) references genero(ID_genero);  
alter table Usuarios_Peliculas add constraint fk_usuarios foreign key (ID_usuarios) references Usuarios(ID_usuarios);  
alter table Usuarios_Peliculas add constraint fk_peliculas foreign key (ID_peliculas) references Peliculas(ID_peliculas);  
alter table Valoracion add constraint fk_usuarios_val foreign key (ID_usuarios) references Usuarios(ID_usuarios);  
alter table Valoracion add constraint fk_peliculas_val foreign key (ID_peliculas) references Peliculas(ID_peliculas);  
alter table Lista_Reproduccion add constraint fk_usuarios_list_rep foreign key (ID_usuarios) references Usuarios(ID_usuarios);  
alter table Lista_Peliculas add constraint fk_listareproduccion_list_pel foreign key (ID_listareproduccion) references Lista_Reproduccion(ID_listareproduccion);  
alter table Lista_Peliculas add constraint fk_peliculas_list_pel foreign key (ID_peliculas) references Peliculas(ID_peliculas);
```



05

VISTAS

BASE DE DATOS – SQL DESDE CERO

MOSTRAR LA CANTIDAD DE PELICULAS POR AÑO.

- `CREATE VIEW peliculas_por_año AS
SELECT YEAR(fecha_estreno) AS año, COUNT(*) AS cantidad FROM peliculas
GROUP BY YEAR(fecha_estreno)
ORDER BY año ASC;`
- `SELECT * FROM peliculas_por_año;`

MOSTRAR SOLO EL NOMBRE Y CORREO DEL USUARIO.

- `CREATE VIEW mostrar_datos_usuario as
SELECT nombre_usuario, correo FROM usuarios;`
- `SELECT * FROM mostrar_datos_usuario;`



06

FUNCTIONS

BASE DE DATOS – SQL DESDE CERO

BUSCAR UNA PELICULA POR SU ID

```
DELIMITER //
```

- `create function fn_encontrar_pelicula (id int)`
`returns varchar(30)`
`reads sql data`
`begin`
 `declare resultado varchar(30);`
 `set resultado = (select titulo from peliculas where ID_peliculas = id);`
 `if resultado is null`
 `then`
 `set resultado = (select "ERROR! El ID no se encuentra cargado");`
 `end if;`
 `return resultado;`
`end //`

```
DELIMITER ;
```

EJEMPLO

- `select fn_encontrar_pelicula (15) as Titulo_Pelicula;`
- `select fn_encontrar_pelicula (1) as Titulo_Pelicula;`

CONTAR PELICULAS POR GENERO

```
DELIMITER //
```

- `create function fn_contar_genero (nom_genero varchar(50))`
`returns int`
`no sql`
`begin`
 `declare cantidad int;`
 `set nom_genero = concat('%', nom_genero, '%');`
 `select count(*) into cantidad from genero where nombre_genero like nom_genero;`
 `return cantidad;`
`end`
 `//`
`DELIMITER ;`

EJEMPLO

- `select fn_contar_genero('Drama') as Cantidad_Peliculas_del_Genero;`
- `select fn_contar_genero('Horror') as Cantidad_Peliculas_del_Genero;`



07

STORED PROCEDURES

BASE DE DATOS – SQL DESDE CERO

ORDENAR LA COLUMNA INGRESADA POR EL USUARIO DE LA TABLA PELICULAS.

```
DELIMITER //
```

- `CREATE PROCEDURE `sp_peli_order` (IN field CHAR(20))`

```
BEGIN
    IF field <> '' THEN
        SET @peli_order = concat('ORDER BY ', field);
    ELSE
        SET @peli_order = '';
    END IF;
    SET @clausula = concat('SELECT * FROM peliculas ', @peli_order);
    PREPARE runSQL FROM @clausula;
    EXECUTE runSQL;
    DEALLOCATE PREPARE runSQL;
END //
```

```
DELIMITER ;
```

EJEMPLO

- `call sp_peli_order ('titulo');`

INSERTAR UN NUEVO REGISTRO EN LA TABLA GÉNERO

```
DELIMITER //  
  
• CREATE PROCEDURE `SP_INSERTAR_GENERO` (IN ID_NEW INT, NOM_GEN TEXT(50), DESCR TEXT(300))  
BEGIN  
    IF NOM_GEN <> '' THEN  
        INSERT INTO genero (ID_genero,nombre_genero,descripcion)  
        VALUES (ID_NEW,NOM_GEN,DESCR);  
        SET @CLAUSULA = "SELECT * FROM genero";  
    ELSE  
        SET @CLAUSULA = "SELECT 'ERROR - DEBE AGREGAR EL NOMBRE DEL GENERO ' AS ERROR";  
    END IF;  
    PREPARE RUNSQL FROM @CLAUSULA;  
    EXECUTE RUNSQL;  
    DEALLOCATE PREPARE RUNSQL;  
END //  
DELIMITER ;
```

EJEMPLO

```
• CALL SP_INSERTAR_GENERO (31,'dinosaurios','este genero es especialmente diseñado solo para las peliculas sobre dinosaurios');
```



08

TRIGGERS

BASE DE DATOS – SQL DESDE CERO

AGREGAR NUEVOS USUARIOS.

CREACION DE LA TABLA new_usuarios PARA GUARDAR LOS CAMBIOS

```
CREATE TABLE new_usuarios (  
    ID_usuarios INT PRIMARY KEY,  
    nombre_usuario VARCHAR(30),  
    contraseña VARCHAR(50),  
    correo VARCHAR(50)  
);
```

EJEMPLO

```
INSERT INTO usuarios (ID_usuarios,nombre_usuario,contraseña,correo) VALUES (97, 'allgr043', '44444asd', 'allgr04@gmail.com');
```

```
CREATE TRIGGER `tr_agregar_new_usuarios`  
AFTER INSERT ON `usuarios`  
FOR EACH ROW -- registra fila por fila  
INSERT INTO `new_usuarios` (ID_usuarios,nombre_usuario,contraseña,correo)  
VALUES (NEW.ID_usuarios,NEW.nombre_usuario,NEW.contraseña,NEW.correo);
```

CONTROL POR LA MODIFICACIÓN DE LA FECHA EN LA TABLA PELICULAS.

CREACION DE LA TABLA log_auditoria PARA GUARDAR LOS CAMBIOS

```
CREATE TABLE `log_auditoria` (  
  `id_log` INT NOT NULL AUTO_INCREMENT,  
  `tabla` VARCHAR(45) NOT NULL,  
  `accion` VARCHAR(15) NOT NULL,  
  `mensaje` VARCHAR(400) NOT NULL,  
  `usuario` VARCHAR(45) NOT NULL,  
  `fecha` DATETIME NOT NULL,  
  PRIMARY KEY (`id_log`));
```

```
DELIMITER //  
  
CREATE TRIGGER `tr_peliculas_modificar_fecha`  
BEFORE UPDATE ON peliculas  
FOR EACH ROW  
BEGIN  
  INSERT INTO log_auditoria(tabla, accion, mensaje, usuario, fecha)  
    VALUES ('peliculas' ,  
            'Modificación' ,  
            concat('Se modificó la fecha de la Pelicula: ', NEW.ID_peliculas , ' con el TITULO: ' , NEW.titulo ,  
            ' * Antes la fecha era: ', OLD.fecha_estreno, ' * Ahora: ', NEW.fecha_estreno) ,  
            USER() ,  
            NOW() );  
END//  
DELIMITER ;  
  
DROP TRIGGER IF EXISTS tr_peliculas_modificar_fecha;
```

EJEMPLO

```
UPDATE peliculas  
SET fecha_estreno = '2023-07-07' WHERE ID_peliculas = 1;
```


CONTROL DE REGISTRO ELIMINADO EN LA TABLA NEW_USUARIOS.

```
DELIMITER //
```

- ```
CREATE TRIGGER `tr_eliminacion_new_usuarios`
AFTER DELETE ON new_usuarios
FOR EACH ROW
BEGIN
 INSERT INTO log_auditoria(tabla, accion, mensaje, usuario, fecha)
 VALUES ('NEW_Usuario' ,
 'Eliminacion' ,
 concat('Se elimino al usuario con ID: ', OLD.ID_usuarios , ' con el NOMBRE DE USUARIO: ' , OLD.nombre_usuario) ,
 USER() ,
 NOW());
END//
DELIMITER ;
```

## EJEMPLO

```
• delete from new_usuarios where ID_usuarios = 94 ;
```



09

# USUARIOS

BASE DE DATOS – SQL DESDE CERO

**ALVARO GUZMAN01:** solo tiene permiso de Lectura.

**ALVARO GUZMAN02:** tiene permisos de Lectura, Inserción y Modificación.

```
-- Mostrar Usuarios
• use mysql;
• SELECT * FROM user;

-- Creacion de los nuevos Usuarios
• create user 'alvaroguzman01'@'localhost' identified by '12345';
• create user 'alvaroguzman02'@'localhost' identified by '56789';

-- Permisos Selectivos usuario alvaroguzman01, solo Lectura
• grant select on peliculas.* to 'alvaroguzman01'@'localhost';

-- Permisos Selectivos usuario alvaroguzman02, Lectura, Insercion y Modificacion
• grant select, insert,update on peliculas.* to 'alvaroguzman02'@'localhost';

-- Mostrar permisos de los usuarios
• show grants for 'root'@'localhost';
• show grants for 'alvaroguzman01'@'localhost';
• show grants for 'alvaroguzman02'@'localhost';
```




10

# BACKUP

BASE DE DATOS – SQL DESDE CERO

# PROCESO DE BACKUP.

Administration - Data Export

BD\_PELICULAS

Data Export

Advanced Options...

Object Selection

Export Progress

Tables to Export

| Exp...                              | Schema    |
|-------------------------------------|-----------|
| <input type="checkbox"/>            | fifa      |
| <input type="checkbox"/>            | gammers   |
| <input checked="" type="checkbox"/> | películas |
| <input type="checkbox"/>            | sakila    |
| <input type="checkbox"/>            | sys       |
| <input type="checkbox"/>            | world     |

Schema Objects

| Exp...                              | Schema Objects        |
|-------------------------------------|-----------------------|
| <input checked="" type="checkbox"/> | genero                |
| <input checked="" type="checkbox"/> | lista_peliculas       |
| <input checked="" type="checkbox"/> | lista_reproduccion    |
| <input checked="" type="checkbox"/> | log_auditoria         |
| <input checked="" type="checkbox"/> | mostrar_datos_usuario |
| <input checked="" type="checkbox"/> | new_usuarios          |
| <input checked="" type="checkbox"/> | películas             |
| <input checked="" type="checkbox"/> | películas_por_año     |
| <input checked="" type="checkbox"/> | usuarios              |

Refresh

11 tables selected

Dump Structure and Data

Select Views

Select Tables

Unselect All

Objects to Export

☒ Dump Stored Procedures and Functions

☒ Dump Events

☒ Dump Triggers

Export Options

☐ Export to Dump Project Folder

C:\Users\allgu\OneDrive\Documentos\dumps\Dump20230726

Each table will be exported into a separate file. This allows a selective restore, but may be slower.

☒ Export to Self-Contained File

C:\Users\allgu\OneDrive\Documentos\dumps\FINAL\_BD\_PELICULAS.sql

All selected database objects will be exported into a single, self-contained file.

☐ Create Dump in a Single Transaction (self-contained file only)

☒ Include Create Schema

Press [Start Export] to start...

Start Export





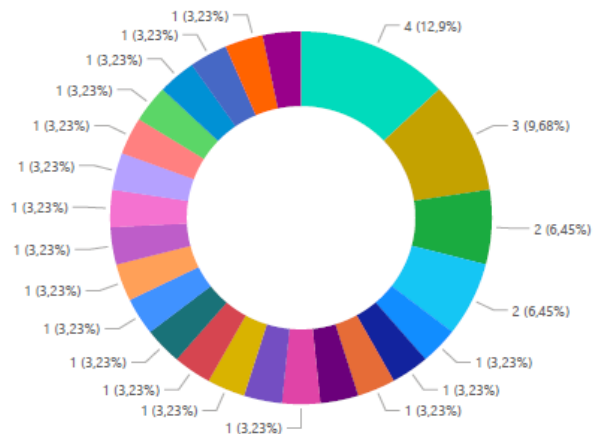
11

# POWER BI

BASE DE DATOS – SQL DESDE CERO

# EJEMPLO CON POWER BI.

## PORCENTAJES DE GENEROS DE PELICULAS



nombre\_genero

Drama

Documentary

Comedy

Comedy|Drama

Action

Action|Adventure|Animation|Children|Comedy|Fantasy

Action|Comedy

Action|Drama

Action|Horror

Adventure|Animation|Children|Comedy|Sci-Fi

Animation|Children|Comedy|Western

Animation|Sci-Fi

Children|Comedy|Drama

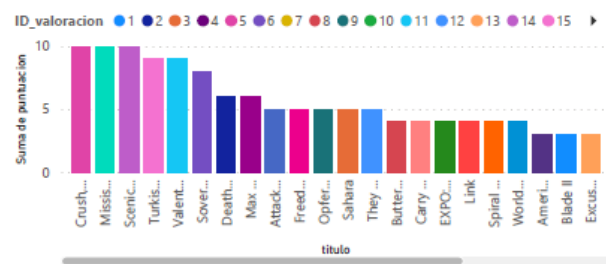
Comedy|Fantasy|Romance

Comedy|Horror

## RECuento de TITULOS POR AÑO



## PUNTUACIONES





11

# CONTACTOS

BASE DE DATOS – SQL DESDE CERO

## Información de Contacto

---

**ÁLVARO JOSÉ GUZMÁN RENGIPO**



[allguzman13@gmail.com](mailto:allguzman13@gmail.com)



+54 9 388 466-5732



[www.linkedin.com/in/alvaro-guzman-796305128](https://www.linkedin.com/in/alvaro-guzman-796305128)





# GRACIAS POR SU ATENCIÓN

---

BASE DE DATOS – SQL DESDE CERO

