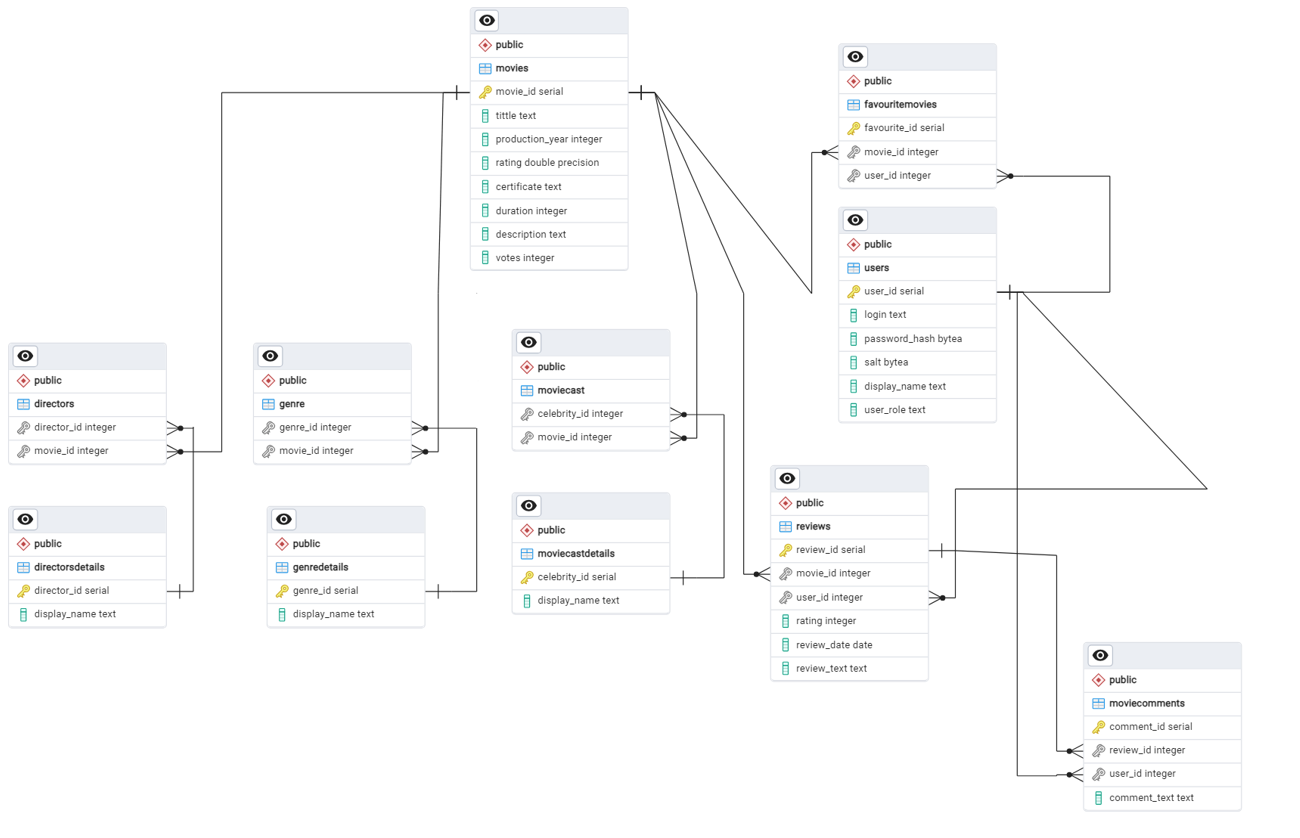
# Baza danych

Projekt wykorzystuje bazę danych PostgreSQL do przechowywania i zarządzania danymi filmów, recenzji, komentarzy, użytkowników, oraz dodatkowych szczegółów takich jak reżyserzy, obsada, i gatunki. Poniżej przedstawiono schemat bazodanowy używanej bazy:



# Tabele

Poniżej znajduje się dokładny opis każdej z tabel:

* **Movies**
  + **movie\_id:** klucz główny, numer identyfikacyjny filmu.
  + **title:** tytuł filmu.
  + **production\_year:** rok produkcji filmu.
  + **rating:** ocena filmu.
  + **certificate:** certyfikat filmu (np. PG-13).
  + **duration:** czas trwania filmu w minutach.
  + **description:** opis filmu.
  + **votes:** liczba głosów oddanych na film.
* **Reviews**
  + **review\_id:** klucz główny, numer identyfikacyjny recenzji.
  + **movie\_id:** klucz obcy, odnosi się do tabeli Movies.
  + **user\_id:** klucz obcy, odnosi się do tabeli Users.
  + **rating:** ocena przyznana w recenzji.
  + **review\_date:** data wystawienia recenzji.
  + **review\_text:** treść recenzji.
* **Users**
  + **user\_id:** klucz główny, numer identyfikacyjny użytkownika.
  + **login:** login użytkownika.
  + **password\_hash:** zahaszowane hasło użytkownika.
  + **salt:** sól do hashowania hasła.
  + **display\_name:** wyświetlana nazwa użytkownika.
  + **user\_role:** rola użytkownika w systemie (np. admin, standardowy użytkownik).
* **MovieComments**
  + **comment\_id:** klucz główny, numer identyfikacyjny komentarza.
  + **review\_id:** klucz obcy, odnosi się do tabeli Reviews.
  + **user\_id:** klucz obcy, odnosi się do tabeli Users.
  + **comment\_text:** treść komentarza.
* **FavoriteMovies**
  + **favourite\_id:** klucz główny, numer identyfikacyjny ulubionego filmu.
  + movie\_id: klucz obcy, odnosi się do tabeli Movies.
  + user\_id: klucz obcy, odnosi się do tabeli Users.
* **DirectorsDetails**
  + **director\_id:** klucz główny, numer identyfikacyjny reżysera.
  + **display\_name:** nazwa wyświetlana reżysera.
* **Directors**
  + **director\_id:** klucz obcy, odnosi się do tabeli DirectorsDetails.
  + **movie\_id:** klucz obcy, odnosi się do tabeli Movies.
* **MovieCastDetails**
  + **celebrity\_id:** klucz główny, numer identyfikacyjny osoby z obsady.
  + **display\_name:** nazwa wyświetlana osoby z obsady.
* **MovieCast**
  + **celebrity\_id:** klucz obcy, odnosi się do tabeli MovieCastDetails.
  + **movie\_id:** klucz obcy, odnosi się do tabeli Movies.
* **GenreDetails**
  + **genre\_id:** klucz główny, numer identyfikacyjny gatunku.
  + **display\_name:** nazwa wyświetlana gatunku.
* **Genre**
  + **genre\_id:** klucz obcy, odnosi się do tabeli GenreDetails.
  + **movie\_id:** klucz obcy, odnosi się do tabeli Movies.

Skrypty tworzące poszczególne tabele.

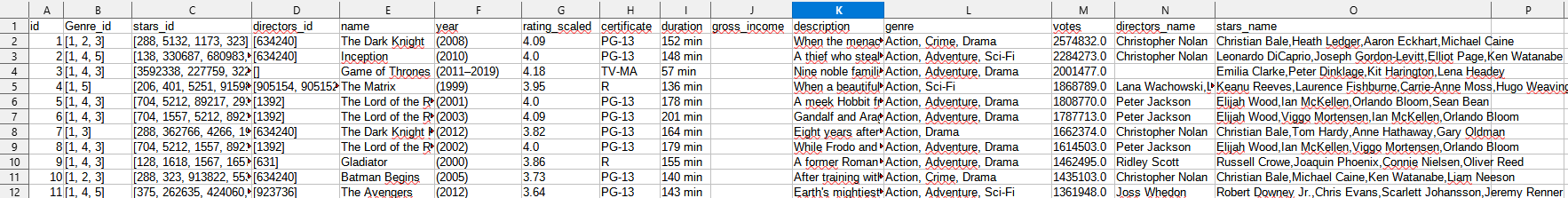
|  |  |
| --- | --- |
| CREATE TABLE Movies (      movie\_id SERIAL PRIMARY KEY,      tittle TEXT NOT NULL,      production\_year INT,      rating FLOAT,      certificate TEXT,      duration INT,      description TEXT,      votes INT  ); | CREATE TABLE Reviews (      review\_id SERIAL PRIMARY KEY,      movie\_id INT NOT NULL,      user\_id INT NOT NULL,      rating INT NOT NULL,      review\_date DATE NOT NULL,      review\_text TEXT  ); |
| CREATE TABLE Users (      user\_id SERIAL PRIMARY KEY,      login TEXT NOT NULL,      password\_hash BYTEA NOT NULL,      salt BYTEA NOT NULL,      display\_name TEXT NOT NULL,      user\_role TEXT NOT NULL  ); | CREATE TABLE MovieComments (      comment\_id SERIAL PRIMARY KEY,      review\_id INT NOT NULL,      user\_id INT NOT NULL,      comment\_text TEXT  ); |
| CREATE TABLE FavouriteMovies (      favourite\_id SERIAL PRIMARY KEY,      movie\_id INT NOT NULL,      user\_id INT NOT NULL  ); | CREATE TABLE DirectorsDetails (      director\_id SERIAL PRIMARY KEY,      display\_name TEXT NOT NULL  ); |
| CREATE TABLE Directors (      director\_id INT NOT NULL,      movie\_id INT NOT NULL  ); | CREATE TABLE MovieCastDetails (      celebrity\_id SERIAL PRIMARY KEY,      display\_name TEXT NOT NULL  ); |
| CREATE TABLE MovieCast (      celebrity\_id INT NOT NULL,      movie\_id INT NOT NULL  ); | CREATE TABLE GenreDetails (      genre\_id SERIAL PRIMARY KEY,      display\_name TEXT NOT NULL  ); |
| CREATE TABLE Genre (      genre\_id INT NOT NULL,      movie\_id INT NOT NULL  ); |  |

# Relacje

1. **Movies i GenreDetails (Relacja wiele-do-wielu):**
   * **Genre:** Ta tabela łącząca zarządza relacjami między filmami a gatunkami, umożliwiając przypisanie wielu gatunków do jednego filmu i odwrotnie. Jest to typowa relacja wiele-do-wielu realizowana przez tabelę łączącą zawierającą klucze obce odnoszące się zarówno do Movies jak i GenreDetails.
2. **Movies i DirectorsDetails (Relacja wiele-do-wielu):**
   * **Directors:** Podobnie jak w przypadku gatunków, ta tabela łącząca zarządza relacją między filmami a reżyserami, umożliwiając przypisanie wielu reżyserów do jednego filmu (często w przypadku współreżyserii) oraz przypisanie jednego reżysera do wielu filmów. Każdy rekord łączy film z reżyserem za pomocą kluczy obcych wskazujących na Movies i DirectorsDetails.
3. ****Movies i MovieCastDetails (Relacja wiele-do-wielu)**:** 
   * **MovieCast:** Ta tabela służy jako mostek dla relacji wiele-do-wielu między filmami a ich obsadą. Umożliwia to, że jeden film może mieć wielu członków obsady i odwrotnie, korzystając z kluczy obcych wskazujących na Movies i MovieCastDetails.
4. ****Reviews (Relacja jeden-do-wielu)**:** 
   * **Movies i Users:** Tabela Reviews przechowuje recenzje, które użytkownicy piszą o filmach. Każda recenzja jest powiązana z dokładnie jednym filmem i jednym użytkownikiem, co przedstawia relację jeden-do-wielu od Movies do Reviews (jeden film może mieć wiele recenzji) oraz od Users do Reviews (jeden użytkownik może napisać wiele recenzji).
5. ****MovieComments (Relacja jeden-do-wielu)**:** 
   * **Reviews i Users:** Każdy komentarz w tabeli MovieComments jest powiązany z dokładnie jedną recenzją i jednym użytkownikiem. Ta konfiguracja tworzy relację jeden-do-wielu od Reviews do MovieComments (jedna recenzja może mieć wiele komentarzy) oraz od Users do MovieComments (jeden użytkownik może zamieścić wiele komentarzy).
6. ****FavouriteMovies (Relacja wiele-do-wielu)**:** 
   * **Movies i Users:** Tabela FavouriteMovies śledzi, którzy użytkownicy dodali które filmy do ulubionych. Choć funkcjonuje to jako relacja wiele-do-wielu (użytkownik może mieć wielu ulubionych filmów i film może być ulubiony przez wielu użytkowników), jest reprezentowane bez typowej tabeli łączącej, ponieważ każdy rekord w FavouriteMovies niejako reprezentuje link między użytkownikiem a filmem.

|  |
| --- |
| ALTER TABLE MovieCast  ADD CONSTRAINT fk\_moviecast\_celebrity\_id FOREIGN KEY (celebrity\_id) REFERENCES MovieCastDetails(celebrity\_id),  ADD CONSTRAINT fk\_moviecast\_movie\_id FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id); |
| ALTER TABLE Directors  ADD CONSTRAINT fk\_directors\_director\_id FOREIGN KEY (director\_id) REFERENCES DirectorsDetails(director\_id),  ADD CONSTRAINT fk\_directors\_movie\_id FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id); |
| ALTER TABLE Genre  ADD CONSTRAINT fk\_genre\_genre\_id FOREIGN KEY (genre\_id) REFERENCES GenreDetails(genre\_id),  ADD CONSTRAINT fk\_genre\_movie\_id FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id); |
| ALTER TABLE Reviews  ADD CONSTRAINT fk\_reviews\_movie\_id FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id),  ADD CONSTRAINT fk\_reviews\_user\_id FOREIGN KEY (user\_id) REFERENCES Users(user\_id); |
| ALTER TABLE MovieComments  ADD CONSTRAINT fk\_moviecomments\_review\_id FOREIGN KEY (review\_id) REFERENCES Reviews(review\_id),  ADD CONSTRAINT fk\_moviecomments\_user\_id FOREIGN KEY (user\_id) REFERENCES Users(user\_id); |
| ALTER TABLE FavouriteMovies  ADD CONSTRAINT fk\_favouritemovies\_movie\_id FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id),  ADD CONSTRAINT fk\_favouritemovies\_user\_id FOREIGN KEY (user\_id) REFERENCES Users(user\_id); |

# Przykładowy zestaw danych

Otrzymany zestaw danych nie jest podzielony odpowiednio pod utworzone tabele. Z wykorzystaniem kilku prostych programów napisanych w Pythonie z użyciem biblioteki pandas, udało nam się przetworzyć oraz przetransformować dane w odpowiedni sposób. Python, ze swoją wszechstronnością i obszernym ekosystemem bibliotek takich jak pandas, jest idealnym narzędziem do manipulacji danymi, szczególnie w przypadkach, gdy wymagana jest elastyczność w przetwarzaniu i transformacji zestawów danych.

Przykładowe zestawy danych po obróbce:

