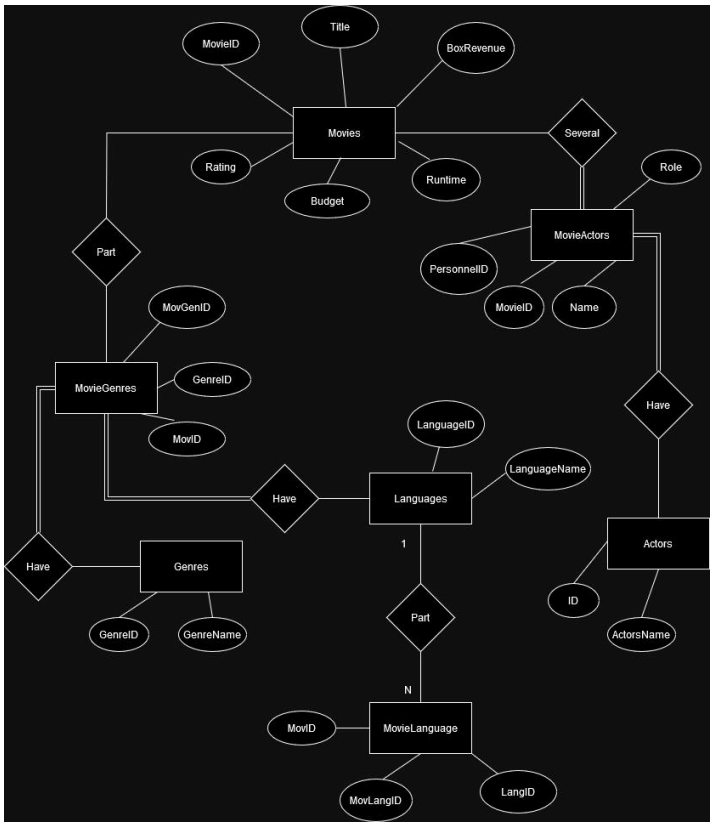
201 Database Project - 38863006

Explanation

In the domain of movie production and distribution, our entity sets encompass various aspects of the industry. The primary entity set, "Movies," serves as the core repository for movie data, featuring attributes such as title, release date, runtime, budget, box office revenue, rating, and release type. This entity relates to other sets, including "Genres," "Languages," and "MoviePersonnel," via foreign keys. For instance, "MoviePersonnel" involves individuals associated with movies, such as directors, actors, and writers, linked to "Movies" through a foreign key referencing movie IDs. Similarly, "Genres" and "Languages" are tied to "Movies" through foreign keys, making a robust relational structure within the database.

ERD:



DDL Statements:

```
CREATE TABLE IF NOT EXISTS Genres (
  ID     INT AUTO_INCREMENT PRIMARY KEY,
  GenreName VARCHAR(100)
 );


CREATE TABLE IF NOT EXISTS Actors (
  ID     INT AUTO_INCREMENT PRIMARY KEY,
  ActorName  VARCHAR(100)
 );


CREATE TABLE IF NOT EXISTS Languages (
  ID             INT AUTO_INCREMENT PRIMARY KEY,
  LanguageName VARCHAR(50)
 );


CREATE TABLE IF NOT EXISTS Movies (
  ID             INT AUTO_INCREMENT PRIMARY KEY,
  Title          VARCHAR(255),
  ReleaseDate        DATE,
  Runtime      INT,
  Budget       INT,
  BoxOfficeRevenue INT,
  Rating       INT,
  ReleaseType        VARCHAR(50),
  DIRECTOR           VARCHAR(100)
 );


CREATE TABLE IF NOT EXISTS MovieGenres (
  ID     INT AUTO_INCREMENT PRIMARY KEY,
  Movie_ID   INT,
  Genre_ID   INT,
  FOREIGN KEY (Genre_ID) REFERENCES Genres(ID),
  FOREIGN KEY (Movie_ID) REFERENCES Movies(ID)
 );
```

```
CREATE TABLE IF NOT EXISTS MovieActors (
  ID     INT AUTO_INCREMENT PRIMARY KEY,
  Movie_ID   INT,
  ActorType  enum('LEAD', 'SUPPORTING'),
  Actor_ID   INT,
  FOREIGN KEY (Actor_ID) REFERENCES Actors(ID),
  FOREIGN KEY (Movie_ID) REFERENCES Movies(ID)
 );



CREATE TABLE IF NOT EXISTS MovieLanguages (
  ID             INT AUTO_INCREMENT PRIMARY KEY,
  Movie_ID     INT,
  Language_ID  INT,
  FOREIGN KEY (Movie_ID) REFERENCES Movies(ID),
  FOREIGN KEY (Language_ID) REFERENCES Languages(ID)
 );
```

Relational Algebra and DML statements of Queries:

```
DELETE FROM Movies ORDER BY ID Asc LIMIT 1;
Movies−πID(σTRUE(Movies)⋈ID=ID(πID(σTRUE(Movies))⋉LIMIT 1))

DELETE FROM Languages ORDER BY ID Asc LIMIT 1;
Languages−πID(σTRUE(Languages)⋈ID=ID(πID(σTRUE(Languages))⋉LIMIT 1))


SELECT a.Title,count(*) No_Of_Languages
FROM  Movies a,
        MovieLanguages b
 WHERE a.ID=b.Movie_ID
 GROUP BY a.title
 HAVING count(*)>3;

π_Title, count() as No_Of_Languages ( σ_count() > 3 ( γ_Title (a ⋈_(a.ID=b.Movie_ID) b) ) )


SELECT a.Title AS Title,count(*) No_Of_Actors
FROM  Movies          a,s released in more
        MovieActors b
WHERE a.ID=b.Movie_ID
GROUP BY a.title
```

HAVING count(*)>6;
π_Title, count*() as No_Of_Actors (σ_count*() > 6 (γ_Title (a ⋈_(a.ID=b.Movie_ID) b)))