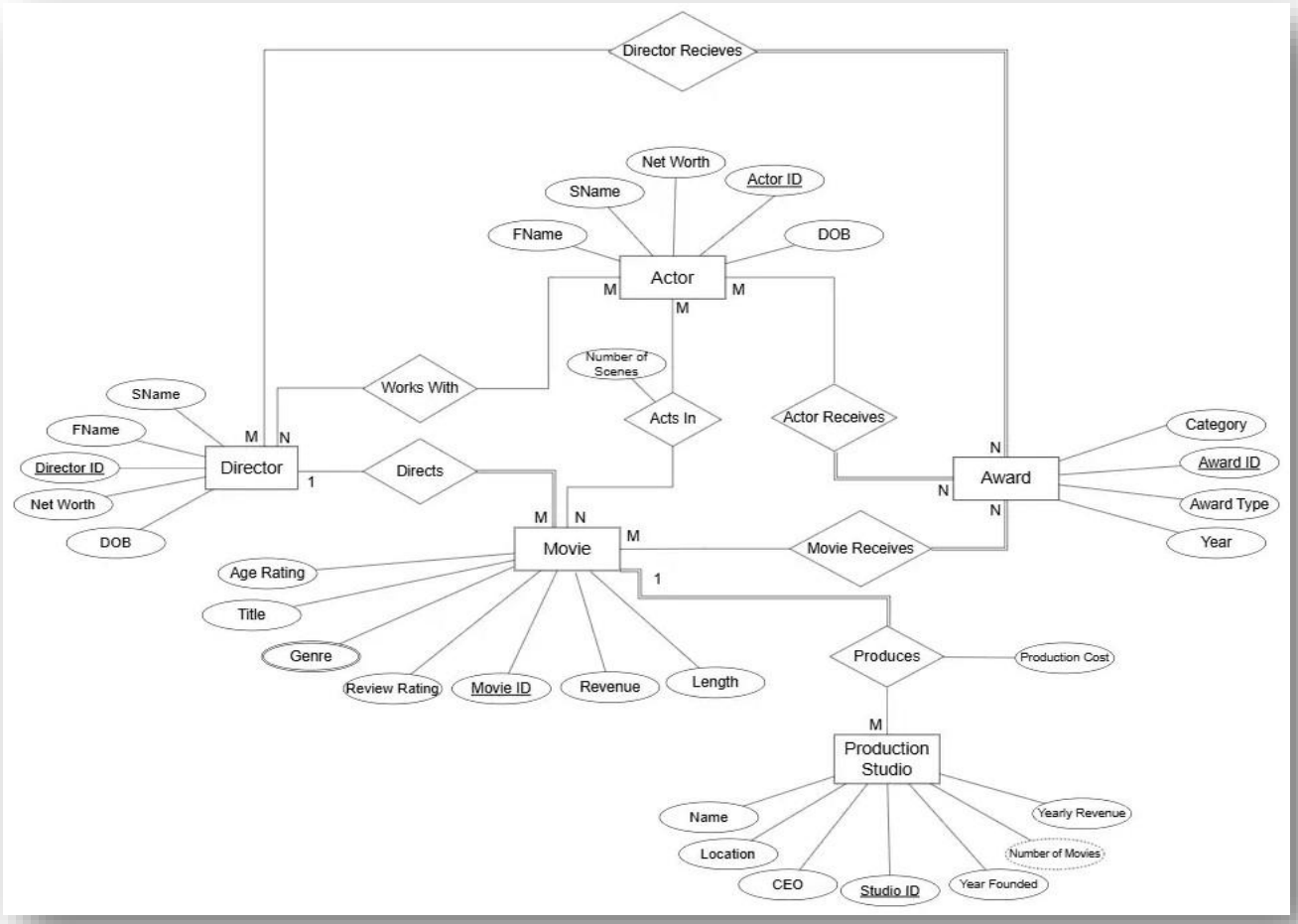
Movie Database Info Document

Designed ER Diagram for Database:



1. Mapping ER diagram into a logical table following the ER mapping steps.

Step 1: Map Strong Entities

Actor: (Actor_ID, Fname, Sname, Networth, DOB)
Director: (Director_ID, Fname, Snake, Net Worth, DOB)
Movie: (MovieID, Age Rating, Title, Genre, Review Rating, Revenue, Length)
Production Studio: (Studio_ID, Name, Location, Year Founded, Number of Movies, Yearly Revenue)
Award: (Award_ID, Category, Award Type, Year)

Step 2: Map Weak Entities

The above ER diagram does not have any weak entities.

Step 3: Map 1:1 Relationships

The above ER diagram does not have any 1:1 relationship.

Step 4: Map 1:N/N:1 Relationships

Director(Director_ID, Movie_ID, Age Rating, Title, Genre, Review Rating, Revenue, Length**)**
FK:Movie_ID

Production Studio(Studio_ID, Movie_ID, Age Rating, Title, Genre, Review Rating, Revenue, Length***)***
FK: Movie_ID

N:1 Actor(Award_ID, Actor_ID, Category, Award Type, Year**)**
FK: Actor_ID

Step 5: Map M:N Relationships

Movie Receives(Award_ID, Movie_ID**)**
FK: Award_ID and Movie_ID

Actor Receives(Award_ID, Actor_ID**)**
FK: Award_ID and Actor_ID

Director Receives(Award_ID, Director_ID**)**
FK: Award_ID and Director_ID

Works With(Director_ID, Actor_ID**)**
FK: Director_ID and Actor_ID

Produces(Studio_ID, Movie_ID)
FK: Studio_ID, Movie_ID

Acts in(Actor_ID, Movie_ID**)**
FK: Actor_ID, Movie_ID

Directs(Director_ID, Movie_ID**)**
FK: Director_ID, Movie_ID

Step 6: Map Multivalued Attributes

Genre is a multivalued attribute. Take the primary key of Movie and create another new relationship with the multivalued attribute. Movie_ID and Genre are the primary key.

Movie_Genre *(Movie_Id, Genre)*
FK: Movie_Id

Final Tables:

Director(Director_ID, SName, FName, Net Worth, DOB)
Movie(Movie_ID, Age Rating, Title, Genre, Review Rating, Revenue, Length)
Award(Award_ID, Category, Award Type, Year)
Production Studio(Studio_ID, Name, Location, CEO, Year Founded, Number of Movies, Yearly Revenue)
Actor(Actor_ID, FName, SName, Net Worth, DOB)
Director Receives(Award_ID, Director_ID**)**
Works With(Director_ID, Actor_ID**)**
Produces(Studio_ID, Movie_ID)
Acts in(Actor_ID, Movie_ID**)**
Directs(Director_ID, Movie_ID**)**
Movie Receives(Award_ID, Movie_ID**)**
Actor Receives(Award_ID, Actor_ID**)**
Movie_Genre *(Movie_Id, Genre)*

2. Normalising each table up to 3rd normal form.

1st Normal form

Entity Tables

Movie(Movie_ID, Title, Review Rating, Revenue, Length, Age Rating)
Director(Director_ID,FName, SName, Net Worth, DOB)
Actor(Actor_ID, FName, SName, Net Worth, DOB)
ProductionStudio(Studio_ID, Name, Location, CEO, Year Founded, Number of Movies, Yearly Revenue)
Award(Award_ID, Category, Award Type, Year)
Movie_Genre(Movie_ID, Genre)

Relationship Tables

Acts In(Movie_ID, Title, Actor_ID, Fname, Sname**)**
Directs(Director_ID, Fname, Sname, Movie_ID**)**
Works With(Director_ID, Actor_ID**)**
Produces(Studio_ID, Movie_ID)
Actor Receives(Award_ID, Actor_ID**)**
Director Receives(Award_ID, Director_ID**)**
Movie Receives(Award_ID, Movie_ID**)**

Changes Made:

- Removed all Multivalued attribute instances and put them into a junction/associative table (for Genre attribute of Movie table -> Movie_Genre junction table).
- Ensure atomicity among all attributes.
- Ensured all Entity Tables have a Primary Key

Explanation (Dependencies):

- At this stage, we ensure **atomicity** and remove any repeating or multivalued attributes.
- Each attribute in a table is fully functionally dependent on the primary key (but partial dependencies may still exist).
- For example, in “Movie”, all attributes are fully dependent on “Movie_ID”.

2nd Normal form

Entity Tables

Movie(Movie_ID, Title, Review Rating, Revenue, Length, Age Rating)
Director(Director_ID, FName, SName, Net Worth, DOB)
Actor(Actor_ID, FName, SName, Net Worth, DOB)
ProductionStudio(Studio_ID, Name, Location, CEO, Year Founded, Yearly Revenue)
Award(Award_ID, Category, Award Type, Year)
Movie_Genre(Movie_ID, Genre)

Relationship Tables

Acts In(Movie_ID, Actor_ID**)**
Directs(Director_ID, Movie_ID**)**
Works With(Director_ID, Actor_ID**)**
Produces(Studio_ID, Movie_ID)
Actor Receives(Award_ID, Actor_ID**)**
Movie Receives(Award_ID, Movie_ID**)**
Director Receives(Award_ID, Director_ID**)**

Changes Made:

- Removed partial dependencies by ensuring all non-key attributes are fully functionally dependent on the whole primary key.
- Removed derived attributes such as "Number of Movies" from "Production Studio".
- Removed Title, fName, sName from relationship tables like Acts In, since they depend only on part of the composite key.

Explanation (Dependencies):

- We removed partial dependencies from tables that had composite primary keys.
- For example:
    - In Acts In(Movie_ID, Actor_ID), attributes like "Title" and Actor Name were partially dependent, so they were removed.
    - In Production Studio, the "Number of Movies" attribute was a derived partial dependency and was removed to meet 2NF.
- Now, every non-key attribute is fully functionally dependent on the entire primary key.

3rd normal form

- Database is already in 3rd Normal Form after changes made for 2nd Normal Form.

Changes Made:

- None

Explanation (Dependencies):

- We confirmed that all transitive dependencies were removed.

- Example:
    - In Production Studio, “CEO” is directly dependent on “Studio_ID”, not through another non-key attribute - hence no transitive dependency.
- Every non-key attribute is non-transitively dependent on the primary key in all tables.

3. Designed SQL DDL to create tables.

Included in .sql file.

4. Created DML to insert records into created tables.

Included in .sql file.

5. Prepared Report.

Goal of The Report:

- Compare Average Movie Revenue *(from “Movie” table)* with Movie Genres *(from “Movie_Genre” table)*.

- Compare Average Movie Revenue *(from “Movie” table)* with Number of Movie Awards Won *(from “Award” table)* from “Movie Receives” relationship.
-
- Compare Movie Genres *(from “Movie_Genre” table)* with Number of Movie Awards Won *(from “Award” table)* from the “Movie_Genre” junction table.

What makes this data relevant:

- We found that genres influence the movie revenue and the profit changes which we can see on the chart.
- This information is useful for studios and investors when deciding how to market their movie according to different genres.
- Comparing average movie revenue with the number of awards won reveals whether award-winning movies are financially more successful than those with fewer awards.
- Comparing movie genres with the number of movie awards won helps identify whether specific genres have influence on the awards won. It can inform directors or writers about the genres which are more likely to succeed.

SQL Statement Used:

```sql
SELECT
    mg.Genre,
    ROUND(AVG(m.Revenue), 2) AS AvgRevenue,
    COUNT(DISTINCT mr.Award_ID) AS AwardsWon
FROM
    Movie m
JOIN
    Movie_Genre mg ON m.Movie_ID = mg.Movie_ID
LEFT JOIN
    Movie_Receives mr ON m.Movie_ID = mr.Movie_ID
GROUP BY
    mg.Genre
ORDER BY
    AvgRevenue DESC;
```

How the Query Works:

This query retrieves information about movie genres, their average revenue and award counts.

The SELECT clause:

- Tells the database the columns to display in the final result.
- "mg.Genre" displays the genre of the movie from the "Movie_Genre" table.
- "ROUND(AVG(m.Revenue), 2)" displays the average revenue from all movies in that Genre rounded to 2 decimal places from the "Movie" table.
- "COUNT(DISTINCT mr.Award_ID)" displays the number of unique (distinct) awards won by Movies in that genre from the "Awards" table.

The FROM clause:

- Sets the base table for the query as the "Movie" table (ensures no movies are excluded from the query) and aliases it to "m".

The JOIN clause:

- This is an **INNER JOIN** between Movie and Movie_Genre to associate each movie with its genres.

- At this point, each row represents a (Movie, Genre) pair.

The LEFT JOIN clause:

This joins in the Movie_Receives table, which connects movies to awards they have won.

- It's a LEFT JOIN, so if a movie hasn't won any awards, it will still appear with NULL for the award data.
- This is important so that movies with no awards still contribute to the average revenue but are counted as 0 in the awards column later.

The GROUP BY clause:

- Groups all the rows by genre.

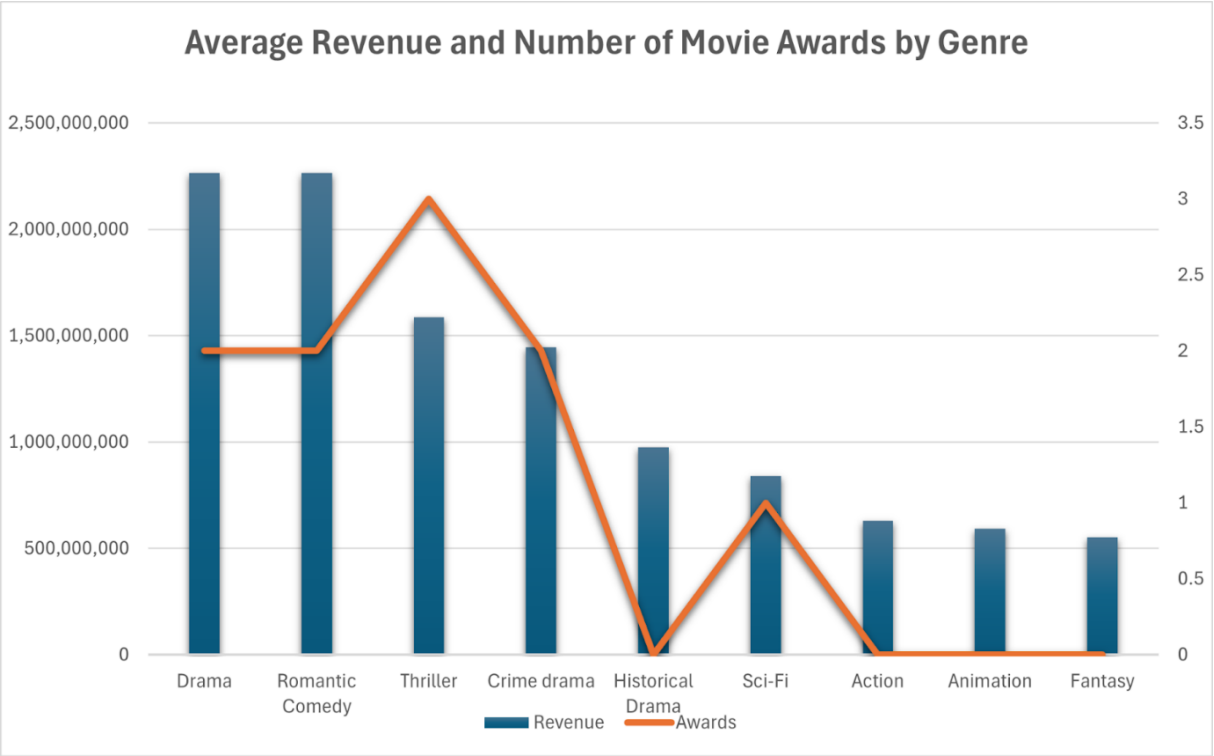
The ORDER BY clause:

- Orders all resulting rows by their average revenue, descending.

Result:

Genre	AvgRevenue	AwardsWon
Drama	2264000000.00	2
Romantic Comedy	2264000000.00	2
Thriller	1585753938.25	3
Crime drama	1444459229.00	2
Historical drama	975985123.00	0
Sci-Fi	839030630.00	1
Action	631000000.00	0
Animation	591000000.00	0
Fantasy	551000000.00	0

Generated graph:



What this data shows:

- The more popular a genre, the higher the Revenue and Awards received.
- The more awards, the higher the Revenue.
- Drama, Romantic Comedy and Thriller are the most popular genres.
- The higher the Awards won, the higher the revenu