**Project : IMDb Movies Analysis using SQL**

Bolly Movies, an Indian film production company, has a successful track record of producing numerous blockbuster films. While primarily catering to the Indian audience, they have decided to venture into the global market with their upcoming project scheduled for release in 2022.

Objective:

Recognizing the value of data-driven decision-making, Bolly Movies has enlisted your expertise as a data analyst and SQL specialist. The objective of this case study is to analyse the movie dataset using SQL queries and extract valuable insights to guide Bolly Movies in planning their new project. The analysis will cover various aspects such as table exploration, movie release trends, production statistics, genre popularity, ratings analysis, crew members, and more.

**Segment 1: Database - Tables, Columns, Relationships**

* **What are the different tables in the database and how are they connected to each other in the database?**

There are six entities which are movie, genre, director\_mapping, role\_mapping, names and ratings. The attributes of movie are id, title, year, date\_published, duration, country, worldwide\_gross\_income, languages, production\_company. The attribute of genre are movie\_id and genre. The attributes of director\_mapping are movie\_id and name\_id. Role\_mapping attributes are movie\_id, name\_id ,category. Names attributes are id, name , height, date-of\_birth, known\_for\_movies. The attributes of ratings are movie\_id , avg\_rating, total\_votes, median\_ratings. The movie table is connected to all table by different relation. Movie table is connected to genre and ratings by exactally one relationship. The movie is in zero to many relationship with director\_mapping and role-mapping where id can have only one movie\_id but movie\_id can have zero to many associated id from movie table. Director\_mapping and role\_mapping are in zero to many relationship with names where id from names can have one and only relation with movie\_id but movie\_id can have zero to many relationship with id attribute of names.

* **Find the total number of rows in each table of the schema.**

CREATE DATABASE imdb;

USE imdb;

SELECT COUNT(\*) FROM director\_mapping; ---- 3867

SELECT COUNT(\*) FROM genre; ---- 14662

SELECT COUNT(\*) FROM movie; ---- 7997

SELECT COUNT(\*) FROM names; ---- 8400

SELECT COUNT(\*) FROM ratings; ---- 7997

SELECT COUNT(\*) FROM role\_mapping; --- 15615

* **Identify which columns in the movie table have null values.**

SELECT COUNT(CASE WHEN title IS NULL THEN 'id' ELSE NULL END) AS null\_title,

COUNT(CASE WHEN year IS NULL THEN 'id' ELSE NULL END) AS null\_year,

COUNT(CASE WHEN date\_published IS NULL THEN 'id' ELSE NULL END) AS null\_date\_published,

COUNT(CASE WHEN duration IS NULL THEN 'id' ELSE NULL END) AS null\_duration,

COUNT(CASE WHEN country IS NULL THEN 'id' ELSE NULL END) AS null\_country,

COUNT(CASE WHEN worlwide\_gross\_income IS NULL THEN 'id' ELSE NULL END) AS null\_worlwide\_gross\_income,

COUNT(CASE WHEN languages IS NULL THEN 'id' ELSE NULL END) AS null\_languages,

COUNT(CASE WHEN production\_company IS NULL THEN 'id' ELSE NULL END) AS null\_production\_company

FROM movie;

Segment 2: Movie Release Trends

* **Determine the total number of movies released each year and analyse the month-wise trend.**

-- Total number of movies released each year

SELECT year AS release\_year,COUNT(\*) AS total\_movies

FROM movie

GROUP BY release\_year

ORDER BY release\_year;

-- Month-wise trend of movie releases

SELECT EXTRACT(MONTH FROM date\_published) AS release\_month,COUNT(\*) AS total\_movies

FROM movie

WHERE date\_published IS NOT NULL

GROUP BY release\_month

ORDER BY release\_month;

* **Calculate the number of movies produced in the USA or India in the year 2019.**

SELECT COUNT(\*) AS num\_movies

FROM movie

WHERE (LOWER(country) LIKE '%USA%' OR LOWER(country) LIKE '%India%') AND year = 2019;

Segment 3: Production Statistics and Genre Analysis

* **Retrieve the unique list of genres present in the dataset.**

SELECT DISTINCT genre

FROM genre;

* **Identify the genre with the highest number of movies produced overall.**

SELECT genre, COUNT(\*) AS num\_movies

FROM genre

GROUP BY genre

ORDER BY num\_movies DESC

LIMIT 1;

* **Determine the count of movies that belong to only one genre.**

SELECT COUNT(\*) AS count\_single\_genre\_movies

FROM (SELECT movie\_id, COUNT(DISTINCT genre) AS genre\_count

FROM genre

GROUP BY movie\_id

HAVING genre\_count = 1) AS single\_genre\_movies;

* **Calculate the average duration of movies in each genre.**

SELECT g.genre, AVG(m.duration) AS avg\_duration

FROM genre g

INNER JOIN movie m ON g.movie\_id = m.id

WHERE m.duration IS NOT NULL

GROUP BY g.genre

ORDER BY avg\_duration DESC;

* **Find the rank of the 'thriller' genre among all genres in terms of the number of movies produced**.

SELECT \* FROM ( SELECT genre, count(movie\_id) AS movie\_count,

RANK() OVER(ORDER BY count(movie\_id) DESC) AS genre\_rank

FROM genre

GROUP BY genre) thriller\_rank

WHERE lower(genre) ='thriller';

Segment 4: Ratings Analysis and Crew Members

* **Retrieve the minimum and maximum values in each column of the ratings table (except movie\_id).**

SELECT

MIN(avg\_rating) AS min\_avg\_rating,

MAX(avg\_rating) AS max\_avg\_rating,

MIN(total\_votes) AS min\_total\_votes,

MAX(total\_votes) AS max\_total\_votes,

MIN(median\_rating) AS min\_median\_rating,

MAX(median\_rating) AS max\_median\_rating

FROM ratings;

* **Identify the top 10 movies based on average rating.**

SELECT m.title,r.avg\_rating

FROM movie AS m

JOIN ratings AS r ON m.id = r.movie\_id

ORDER BY r.avg\_rating DESC

LIMIT 10**;**

* **Summarise the ratings table based on movie counts by median ratings.**

SELECT median\_rating,COUNT(\*) AS movie\_count

FROM ratings

GROUP BY median\_rating

ORDER BY median\_rating;

* **Identify the production house that has produced the most number of hit movies (average rating > 8).**

SELECT m.production\_company,COUNT(\*) AS num\_hit\_movies

FROM movie AS m

JOIN ratings AS r ON m.id = r.movie\_id

WHERE r.avg\_rating > 8

GROUP BY m.production\_company

ORDER BY num\_hit\_movies DESC

LIMIT 1;

* **Determine the number of movies released in each genre during March 2017 in the USA with more than 1,000 votes.**

SELECT g.genre, COUNT(\*) AS num\_movies

FROM genre g

INNER JOIN movie m ON g.movie\_id = m.id

INNER JOIN ratings r ON m.id = r.movie\_id

WHERE m.year = 2017 AND country = lower(country like '%usa%') AND EXTRACT(MONTH FROM m.date\_published) = 3

AND r.total\_votes > 1000

GROUP BY g.genre

ORDER BY num\_movies DESC;

* **Retrieve movies of each genre starting with the word 'The' and having an average rating > 8.**

SELECT g.genre,m.title,r.avg\_rating

FROM genre AS g

JOIN movie AS m ON g.movie\_id = m.id

JOIN ratings AS r ON m.id = r.movie\_id

WHERE r.avg\_rating > 8 AND m.title LIKE 'The%'

ORDER BY g.genre, r.avg\_rating DESC;

Segment 5: Crew Analysis

* **Identify the columns in the names table that have null values.**

SELECT count(CASE WHEN id IS NULL THEN 'id' ELSE NULL END) AS null\_id,

count(CASE WHEN name IS NULL THEN 'name' ELSE NULL END) AS null\_name,

count(CASE WHEN height IS NULL THEN 'height' ELSE NULL END) AS null\_height,

count(CASE WHEN date\_of\_birth IS NULL THEN 'date\_of\_birth' ELSE NULL END) AS null\_date\_of\_birth,

count(CASE WHEN known\_for\_movies IS NULL THEN 'known\_for\_movies' ELSE NULL END) AS null\_known\_for\_movies

FROM names;

* **Determine the top three directors in the top three genres with movies having an average rating > 8.**

WITH TopGenres AS (SELECT g.genre

FROM genre g

JOIN movie m ON g.movie\_id = m.id

JOIN ratings r ON m.id = r.movie\_id

WHERE r.avg\_rating > 8

GROUP BY g.genre

ORDER BY COUNT(\*) DESC

LIMIT 3),

TopDirectors AS (SELECT m.title AS movie\_title,n.name AS director\_name,r.avg\_rating,g.genre

FROM movie m

JOIN genre g ON m.id = g.movie\_id

JOIN ratings r ON m.id = r.movie\_id

JOIN director\_mapping dm ON m.id = dm.movie\_id

JOIN names n ON dm.name\_id = n.id

WHERE r.avg\_rating > 8)

SELECT tg.genre AS top\_genre,td.director\_name,COUNT(\*) AS num\_movies

FROM TopGenres tg

JOIN TopDirectors td ON tg.genre = td.genre

GROUP BY tg.genre, td.director\_name

ORDER BY tg.genre, num\_movies DESC

LIMIT 3;

* **Find the top two actors whose movies have a median rating >= 8.**

WITH ActorMedians AS (SELECT nm.name AS actor\_name,r.median\_rating,

ROW\_NUMBER() OVER (ORDER BY r.median\_rating DESC) AS actor\_rank

FROM role\_mapping rm

JOIN names nm ON rm.name\_id = nm.id

JOIN ratings r ON rm.movie\_id = r.movie\_id

WHERE rm.category = 'actor' AND r.median\_rating >= 8)

SELECT actor\_name,median\_rating

FROM ActorMedians

WHERE actor\_rank <= 2;

* **Identify the top three production houses based on the number of votes received by their movies.**

SELECT m.production\_company,SUM(r.total\_votes) AS total\_votes\_received

FROM movie AS m

JOIN ratings AS r ON m.id = r.movie\_id

WHERE m.production\_company IS NOT NULL

GROUP BY m.production\_company

ORDER BY total\_votes\_received DESC

LIMIT 3;

* **Rank actors based on their average ratings in Indian movies released in India.**

WITH IndianMovies AS (SELECT m.id AS movie\_id,rm.name\_id AS actor\_id,r.avg\_rating

FROM movie m

JOIN role\_mapping rm ON m.id = rm.movie\_id

JOIN ratings r ON m.id = r.movie\_id

WHERE m.country = 'India'AND rm.category = 'actor'),

ActorRanks AS (SELECT actor\_id,AVG(avg\_rating) AS avg\_actor\_rating,

ROW\_NUMBER() OVER (ORDER BY AVG(avg\_rating) DESC) AS actor\_rank

FROM IndianMovies

GROUP BY actor\_id)

SELECT a.name AS actor\_name,ar.avg\_actor\_rating

FROM ActorRanks ar

JOIN names a ON ar.actor\_id = a.id

ORDER BY actor\_rank;

* **Identify the top five actresses in Hindi movies released in India based on their average ratings.**

WITH HindiActressRatings AS (SELECT nm.name AS actress\_name,rm.name\_id AS actress\_id,AVG(r.avg\_rating) AS avg\_rating

FROM movie m

JOIN genre g ON m.id = g.movie\_id

JOIN role\_mapping rm ON m.id = rm.movie\_id

JOIN names nm ON rm.name\_id = nm.id

JOIN ratings r ON m.id = r.movie\_id

WHERE m.country = 'India' AND m.languages LIKE '%Hindi%' AND rm.category = 'actress'

GROUP BY actress\_id, actress\_name)

SELECT actress\_name,avg\_rating

FROM HindiActressRatings

ORDER BY avg\_rating DESC

LIMIT 5;

Segment 6: Broader Understanding of Data

* **Classify thriller movies based on average ratings into different categories.**

SELECT title,avg\_rating,

CASE WHEN avg\_rating >= 8.0 THEN 'High'

WHEN avg\_rating >= 6.0 THEN 'Medium' ELSE 'Low'END AS rating\_category

FROM movie AS m

JOIN ratings AS r ON m.id = r.movie\_id

WHERE EXISTS (SELECT 1

FROM genre AS g

WHERE m.id = g.movie\_id AND g.genre = 'Thriller');

* **Analyse the genre-wise running total and moving average of the average movie duration.**

WITH GenreAverageDurations AS (SELECT g.genre,AVG(m.duration) AS avg\_duration

FROM genre g

JOIN movie m ON g.movie\_id = m.id

WHERE m.duration IS NOT NULL

GROUP BY g.genre

ORDER BY g.genre),

RankedGenres AS (SELECT genre,avg\_duration,ROW\_NUMBER() OVER (PARTITION BY genre ORDER BY genre) AS row\_num

FROM GenreAverageDurations)

SELECT rg.genre,rg.avg\_duration,SUM(rg.avg\_duration) OVER (PARTITION BY rg.genre ORDER BY rg.row\_num) AS genre\_running\_total,

AVG(rg.avg\_duration) OVER (PARTITION BY rg.genre ORDER BY rg.row\_num ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS genre\_moving\_avg

FROM RankedGenres rg;

* **Identify the five highest-grossing movies of each year that belong to the top three genres.**

WITH TopGenres AS (SELECT g.genre,COUNT(\*) AS genre\_count

FROM genre g

GROUP BY g.genre

ORDER BY genre\_count DESC

LIMIT 3),

RankedMovies AS (SELECT m.id,m.title,m.year,m.worlwide\_gross\_income,g.genre,

ROW\_NUMBER() OVER (PARTITION BY m.year, g.genre ORDER BY CAST(REPLACE(m.worlwide\_gross\_income, '$', '') AS DECIMAL) DESC) AS ranking

FROM movie m

JOIN genre g ON m.id = g.movie\_id

WHERE g.genre IN (SELECT genre FROM TopGenres))

SELECT year,genre,title,worlwide\_gross\_income

FROM RankedMovies

WHERE ranking <= 5

ORDER BY year, genre, ranking;

* **Determine the top two production houses that have produced the highest number of hits among multilingual movies.**

WITH MultilingualMovies AS (SELECT m.production\_company,r.avg\_rating

FROM movie m

JOIN ratings r ON m.id = r.movie\_id

WHERE m.languages IS NOT NULL AND r.avg\_rating >= 8),

ProductionHouseHits AS (SELECT production\_company,COUNT(\*) AS num\_hits

FROM MultilingualMovies

GROUP BY production\_company

ORDER BY num\_hits DESC)

SELECT production\_company,num\_hits

FROM ProductionHouseHits

LIMIT 2;

* **Identify the top three actresses based on the number of Super Hit movies (average rating > 8) in the drama genre**

WITH DramaActressHits AS (SELECT nm.name AS actress\_name,rm.name\_id AS actress\_id,COUNT(\*) AS num\_super\_hits

FROM role\_mapping rm

JOIN names nm ON rm.name\_id = nm.id

JOIN movie m ON rm.movie\_id = m.id

JOIN ratings r ON m.id = r.movie\_id

JOIN genre g ON m.id = g.movie\_id

WHERE rm.category = 'actress' AND r.avg\_rating > 8 AND g.genre = 'Drama'

GROUP BY actress\_id, actress\_name

ORDER BY num\_super\_hits DESC

LIMIT 3)

SELECT actress\_name,num\_super\_hits

FROM DramaActressHits;

* **Retrieve details for the top nine directors based on the number of movies, including average inter-movie duration, ratings, and more.** **WITH DirectorStats AS (SELECT nm.name AS** director\_name,COUNT(\*) AS num\_movies,AVG(m.duration) AS avg\_duration,

AVG(r.avg\_rating) AS avg\_rating

FROM director\_mapping dm

JOIN names nm ON dm.name\_id = nm.id

JOIN movie m ON dm.movie\_id = m.id

JOIN ratings r ON m.id = r.movie\_id

GROUP BY director\_name

ORDER BY num\_movies DESC

LIMIT 9)

SELECT ds.director\_name,ds.num\_movies,ds.avg\_duration AS director\_avg\_duration,ds.avg\_rating AS director\_avg\_rating,

(SELECT COUNT(\*) FROM director\_mapping WHERE name\_id = ds.director\_name) AS num\_movies\_directed,

(SELECT MAX(year) FROM movie WHERE id IN (SELECT movie\_id FROM director\_mapping WHERE name\_id = ds.director\_name)) AS most\_recent\_movie\_year

FROM DirectorStats ds;

Segment 7: Recommendations

* **Based on the analysis, provide recommendations for the types of content Bolly movies should focus on producing.**

Based on the data, drama, comedy and thriller genres seem to perform well, with a significant number of movies and high average ratings. Bolly movies can continue to produce content in these genres, focusing on storytelling and engaging plots. While drama, comedy and thriller genres are popular, Bolly movies should also experiment with other genres to diversify their content. This could include genres like romance or science fiction, depending on market trends and audience preferences. Bolly movies can consider producing more multilingual films, especially those with high-quality content. Bolly movies should prioritize producing high-quality films with engaging stories, talented casts, and strong direction. Bolly movies can support emerging directors, actors, and writers who have the potential to bring fresh and innovative content to the industry. Nurturing new talent can lead to unique and compelling movies. Moreover by analysing the success of some Indian movies in international markets, Bolly movies should collaborate with international filmmakers, actors, and studios can lead to unique and appealing content that resonates with a global audience.Regularly conduct market research to understand changing audience preferences and trends. Staying updated with audience demands can help Bolly movies tailor their content accordingly. Continuously gather and analyze audience feedback to understand what works and what doesn't. This feedback can guide future content decisions.