IMDB Movie Dataset

PROJECT BY- TEJAS JANGID

GUIDED BY- MR. SAMIR WARSOLKARSIR

Abstract:

The **IMDb Movie Dataset** refers to a comprehensive collection of information related to movies, television programs, and video games, primarily sourced from IMDb. Researchers and data scientists often use the IMDb dataset to perform tasks like sentiment analysis, recommendation systems, predictive modeling for movie success, and other entertainment industry analytics. IMDb provides public access to a subset of this data, while more detailed datasets are often curated through licensed agreements or through scraping.

The IMDB Movie dataset is a collection of movie reviews from the Internet Movie Database (IMDb), a popular online platform for film and television information. The dataset is available in gzipped, tab-separated-values (TSV) format, with headers describing each column. The data is refreshed daily and can be accessed from the IMDb website.

Tejas Sujeet Jangid

Tejasjangid15@gmail.com

Batch ID: DS T-326/ 11 To 1

Project for SQL Module

IMDB Dataset project, the focus would be on creating and managing a database to track Movie Ratings, Reviews, Genres, and Movie Titles.

Objectives:

- **Understand Genre Trends**: Analyze which movie genres are most popular over time and across different regions.
- **Evaluate Actor/Director Performance**: Assess how the involvement of specific actors or directors impacts movie success.
- **Improve Movie Recommendations**: Develop insights to enhance personalized movie recommendations based on user preferences.

Components:

- **Movie Information**: Details on movie titles, genres, runtime, release dates, and languages.
- **Ratings and Reviews**: User ratings, reviews, and critic scores for each movie.
- **Genre Classification**: Categorization of movies by genres such as Action, Drama, Comedy, etc.

Functionality

The functionality of pizza sales includes tracking and analyzing sales data to understand trends and customer preferences. It also involves managing inventory to ensure stock levels meet demand. Additionally, it supports optimizing order processing and delivery efficiency to enhance customer satisfaction.

ER diagram (Entity Relationship Diagram) for IMDB Movie Dataset

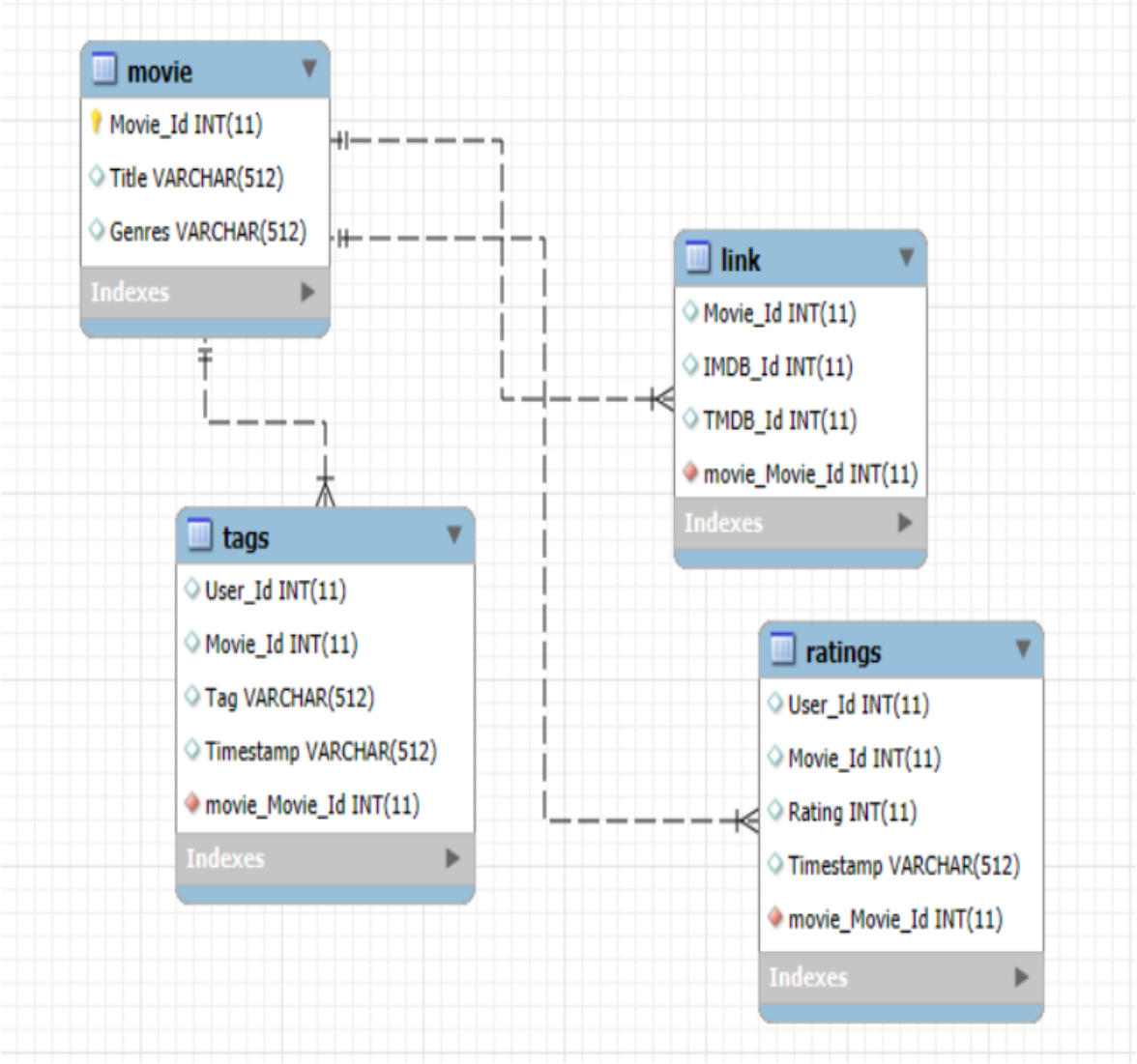

TABLE DISCRIPTION:

1. Movie:

Field	Type	Null
Title	Varchar(512)	Yes
Movie_Id	Int(11)	Yes
Genres	Varchar(512)	Yes

2. Link:

Field	Type	Null
Movie_Id	Int(11)	Yes
IMDB_Id	Int(11)	Yes
TMDB_Id	Int(11)	Yes

3. Ratings:

Field	Type	Null
User_Id	Int(11)	Yes
Movie_Id	Int(11)	Yes
Rating	Int(11)	Yes
Timestamp	Varchar(512)	Yes

4. Tags:

Field	Type	Null
User_Id	Int(11)	Yes
Movie_Id	Int(11)	Yes
Tag	Varchar(512)	Yes
Timestamp	Varchar(512)	Yes

IMDB Movie DATABASE

CREATE DATABASE:

```
create database project;

use project;
```

CREATE TABLE AND INSERTION COMMANDS:

1. CREAT TABLE Movie:

```
CREATE TABLE Movie

(Movie_Id INT,

Title VARCHAR(512),

Genres VARCHAR(512));


INSERT INTO Movie VALUES

('1', 'Toy Story (2001)', 'Adventure|Animation|Children|Comedy|Fantasy'),

('2', 'Jumanji (1995)', 'Adventure|Children|Fantasy'),

('3', 'Grumpier Old Men (1998)', 'Comedy|Romance'),

('4', 'Waiting to Exhale (1990)', 'Comedy|Drama|Romance'),

('5', 'Father of the Bride Part II (1985)', 'Comedy'),

('6', 'Heat (1966)', 'Action|Crime|Thriller'),

('7', 'Sabrina (2002)', 'Comedy|Romance'),

('8', 'Tom and Huck (2000)', 'Adventure|Children'),

('9', 'Sudden Death (1995)', 'Action'),

('10', 'GoldenEye (1995)', 'Action|Adventure|Thriller'),

('11', 'American President, The (1997)', 'Comedy|Drama|Romance'),

('12', 'Dracula: Dead and Loving It (1991)', 'Comedy|Horror'),

('13', 'Balto (1989)', 'Adventure|Animation|Children'),

('14', 'Nixon (1988)', 'Drama'),

('15', 'Cutthroat Island (2008)', 'Action|Adventure|Romance'),

('16', 'Casino (2000)', 'Crime|Drama'),

('17', 'Sense and Sensibility (2003)', 'Drama|Romance'),
```

('18', 'Four Rooms (2013)', 'Comedy'),

('19', 'Ace Ventura: When Nature Calls (1958)', 'Comedy'),

('20', 'Money Train (2014)', 'Action|Comedy|Crime|Drama|Thriller');

2. CREATE TABLE Link

CREATE TABLE Link

(Movie_Id INT,

IMDB_Id INT,

TMDB_Id INT);

INSERT INTO Link VALUES

('1', '114709', '862'),

('2', '113497', '8844'),

('3', '113228', '15602'),

('4', '114885', '31357'),

('5', '113041', '11862'),

('6', '113277', '949'),

('7', '114319', '11860'),

('8', '112302', '45325'),

('9', '114576', '9091'),

('10', '113189', '710'),

('11', '112346', '9087'),

('12', '112896', '12110'),

('13', '112453', '21032'),

('14', '113987', '10858'),

('15', '112760', '1408'),

('16', '112641', '524'),

('17', '114388', '4584'),

('18', '113101', '5'),

('19', '112281', '9273'),

('20', '113845', '11517');

3. CREATE TABLE Ratings

```
CREATE TABLE Ratings
(User_Id INT,
Movie_Id INT,
Rating  INT,
Timestamp VARCHAR(512));

INSERT INTO Ratings VALUES
('1081', '1', '4', 'November 22, 2001'),
('1097', '2', '4', 'December 15, 1995'),
('1056', '3', '4', 'December 22, 1998'),
('1053', '4', '5', 'December 22, 1990'),
('1040', '5', '5', 'December 8, 1985'),
('1028', '6', '3', 'December 15, 1966'),
('1015', '7', '5', 'December 15, 2002'),
('1003', '8', '4', 'December 22, 2000'),
('990', '9', '5', 'December 22, 1995'),
('978', '10', '5', 'November 17, 1995'),
('965', '11', '5', 'November 17, 1997'),
('953', '12', '5', 'December 22, 1991'),
('940', '13', '3', 'December 22, 1989'),
('928', '14', '5', 'December 20, 1988'),
('915', '15', '4', 'December 22, 2008'),
('903', '16', '5', 'November 22, 2000'),
('890', '17', '3', 'December 13, 2003'),
('878', '18', '3', 'December 25, 2013'),
('865', '19', '5', 'November 10, 1958'),
('853', '20', '4', 'November 22, 2014');
```

4. CREATE TABLE Tags

```
CREATE TABLE Tags
(User_Id INT,
Movie_Id INT,
Tag VARCHAR(512),
Timestamp VARCHAR(512));

INSERT INTO Tags VALUES
('1081', '1', 'Funny', 'November 22, 2001'),
('1097', '2', 'Funny', 'December 15, 1995'),
('1056', '3', 'Romantic', 'December 22, 1998'),
('1053', '4', 'Drama', 'December 22, 1990'),
('1040', '5', 'Funny', 'December 8, 1985'),
('1028', '6', 'Thriller', 'December 15, 1966'),
('1015', '7', 'Comedy', 'December 15, 2002'),
('1003', '8', 'Adventure', 'December 22, 2000'),
('990', '9', 'Action', 'December 22, 1995'),
('978', '10', 'Thriller', 'November 17, 1995'),
('965', '11', 'Drama', 'November 17, 1997'),
('953', '12', 'Horror', 'December 22, 1991'),
('940', '13', 'Children', 'December 22, 1989'),
('928', '14', 'Drama', 'December 20, 1988'),
('915', '15', 'Action', 'December 22, 2008'),
('903', '16', 'Crime', 'November 22, 2000'),
('890', '17', 'Romantic', 'December 13, 2003'),
('878', '18', 'Comedy', 'December 25, 2013'),
('865', '19', 'Funny', 'November 10, 1958'),
('853', '20', 'Drama', 'November 22, 2014');
```

QUERIES

Basic queries

1. Retrieve the total number of Movies.

   Command-

   SELECT COUNT(Movie_Id) AS Total_Movies FROM Movie;

```
select count(Movie_Id) as Total_Movies from Movie;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Total_Movies
20

2. Get all movies released in 1995.

   Command-

   SELECT * FROM Movie
   WHERE Title LIKE '%(1995)%';

```
SELECT * FROM Movie
WHERE Title LIKE '%(1995)%';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Movie_Id	Title	Genres
2	Jumanji (1995)	Adventure|Children|Fantasy
9	Sudden Death (1995)	Action
10	GoldenEye (1995)	Action|Adventure|Thriller

3. **Find all movies with the genre "Comedy".**

Command-

SELECT * FROM Movie
WHERE Genres LIKE '%Comedy%';

```
132 •   SELECT * FROM Movie
133     WHERE Genres LIKE '%Comedy%';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Movie_Id	Title	Genres
1	Toy Story (2001)	Adventure|Animation|Children|Comedy|Fantasy
3	Grumpier Old Men (1998)	Comedy|Romance
4	Waiting to Exhale (1990)	Comedy|Drama|Romance
5	Father of the Bride Part II (1985)	Comedy
7	Sabrina (2002)	Comedy|Romance
11	American President, The (1997)	Comedy|Drama|Romance
12	Dracula: Dead and Loving It (1991)	Comedy|Horror
18	Four Rooms (2013)	Comedy
19	Ace Ventura: When Nature Calls (1958)	Comedy
20	Money Train (2014)	Action|Comedy|Crime|Drama|Thriller

4. **List all movies with 'Action' genre released in 2008.**

Command-

SELECT * FROM Movie
WHERE Genres
LIKE '%Action%' AND Title LIKE '%(2008)%';

```
135 •   SELECT * FROM Movie WHERE Genres LIKE '%Action%' AND Title LIKE '%(2008)%';
136
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Movie_Id	Title	Genres
15	Cutthroat Island (2008)	Action|Adventure|Romance

5. Find the average rating for the movie 'Toy Story (2001)'.

Command-

Select AVG(Rating) AS Average_Rating FROM Ratings;

```
137 •   SELECT AVG(Rating) AS Average_Rating FROM Ratings;
138
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	Average_Rating
▶	4.3000

6. Find all users who rated the movie 'GoldenEye (1995)'.

Command-

SELECT User_Id FROM Ratings WHERE Movie_Id = 10;

```
133
134 •   SELECT User_Id FROM Ratings WHERE Movie_Id = 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	User_Id
▶	978

Ratings 2 ×

Sub-queries

1. List all movies not rated by user 1081.

   Command-

   SELECT Movie.Title
   FROM Movie
   WHERE Movie.Movie_Id NOT IN
   (SELECT Movie_Id FROM Ratings WHERE User_Id = 1081);

```
SELECT Movie.Title
FROM Movie
WHERE Movie.Movie_Id NOT IN
(SELECT Movie_Id FROM Ratings WHERE User_Id = 1081);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Title
Jumanji (1995)
Grumpier Old Men (1998)
Waiting to Exhale (1990)
Father of the Bride Part II (1985)
Heat (1966)
Sabrina (2002)
Tom and Huck (2000)
Sudden Death (1995)
GoldenEye (1995)
American President, The (1997)
Dracula: Dead and Loving It (1991)
Balto (1989)
Nixon (1988)
Cutthroat Island (2008)
Casino (2000)

2. Select all movies that have not been tagged with 'Funny' or 'Romantic'.

Command-

SELECT Movie.Title
FROM Movie
WHERE Movie.Movie_Id NOT IN
(SELECT Movie_Id FROM Tags WHERE Tag IN ('Funny', 'Romantic'));

```
146 •  SELECT Movie.Title
147    FROM Movie
148    WHERE Movie.Movie_Id NOT IN
149    (SELECT Movie_Id FROM Tags WHERE Tag IN ('Funny', 'Romantic'));
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Title
Waiting to Exhale (1990)
Heat (1966)
Sabrina (2002)
Tom and Huck (2000)
Sudden Death (1995)
GoldenEye (1995)
American President, The (1997)
Dracula: Dead and Loving It (1991)
Balto (1989)
Nixon (1988)
Cutthroat Island (2008)
Casino (2000)
Four Rooms (2013)
Money Train (2014)

Movie 9

3. Find all movies in the 'Action' genre.

Command-

SELECT Movie.Title
FROM Movie
WHERE Movie.Movie_Id IN
(SELECT Movie_Id FROM Movie WHERE Genres LIKE '%Action%');

```
SELECT Movie.Title
FROM Movie
WHERE Movie.Movie_Id IN
(SELECT Movie_Id FROM Movie WHERE Genres LIKE '%Action%');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Title
Heat (1966)
Sudden Death (1995)
GoldenEye (1995)
Cutthroat Island (2008)
Money Train (2014)

4. Find all movies that were rated before December 15, 2002.

Command-

SELECT Movie.Title
FROM Movie
WHERE Movie.Movie_Id IN (SELECT Movie_Id FROM Ratings
WHERE Timestamp < 'December 15, 2002');

```
SELECT Movie.Title
FROM Movie
WHERE Movie.Movie_Id IN
(SELECT Movie_Id FROM Ratings WHERE Timestamp < 'December 15, 2002');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Title
Jumanji (1995)
Heat (1966)
Sense and Sensibility (2003)

5. **Select all movies with an average rating greater than 4.**

Command-

```
SELECT Movie.Title, AVG(Ratings.Rating) AS Avg_Rating
FROM Movie
JOIN Ratings ON Movie.Movie_Id = Ratings.Movie_Id
GROUP BY Movie.Movie_Id HAVING AVG(Ratings.Rating) > 4;
```

```
SELECT Movie.Title, AVG(Ratings.Rating) AS Avg_Rating
FROM Movie
JOIN Ratings ON Movie.Movie_Id = Ratings.Movie_Id
GROUP BY Movie.Movie_Id HAVING AVG(Ratings.Rating) > 4;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Title	Avg_Rating
Waiting to Exhale (1990)	5.0000
Father of the Bride Part II (1985)	5.0000
Sabrina (2002)	5.0000
Sudden Death (1995)	5.0000
GoldenEye (1995)	5.0000
American President, The (1997)	5.0000
Dracula: Dead and Loving It (1991)	5.0000
Nixon (1988)	5.0000
Casino (2000)	5.0000
Ace Ventura: When Nature Calls (1958)	5.0000

JOINS

1. **Find the movie with the highest rating by user 1040.**

   COMMAND-

   SELECT Movie.Title FROM Movie
   JOIN Ratings ON Movie.Movie_Id = Ratings.Movie_Id
   WHERE Ratings.User_Id = 1040
   ORDER BY Ratings.Rating DESC;

```
SELECT Movie.Title
FROM Movie
JOIN Ratings ON Movie.Movie_Id = Ratings.Movie_Id
WHERE Ratings.User_Id = 1040
ORDER BY Ratings.Rating DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

Title
Father of the Bride Part II (1985)

2. **Which movie titles have an IMDb ID over 113000.**

COMMAND-

```
SELECT Movie.Title
FROM Movie
JOIN Link ON Movie.Movie_Id = Link.Movie_Id
WHERE Link.IMDB_Id > 113000;
```

```
173 ●   SELECT Movie.Title
174     FROM Movie
175     JOIN Link ON Movie.Movie_Id = Link.Movie_Id
176     WHERE Link.IMDB_Id > 113000;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Title
Toy Story (2001)
Jumanji (1995)
Grumpier Old Men (1998)
Waiting to Exhale (1990)
Father of the Bride Part II (1985)
Heat (1966)
Sabrina (2002)
Sudden Death (1995)
GoldenEye (1995)
Nixon (1988)
Sense and Sensibility (2003)
Four Rooms (2013)
Money Train (2014)

3. **Inner Join to get movie titles and their IMDb IDs:**

COMMAND-

```
SELECT Movie.Title, Link.IMDB_Id
FROM Movie
INNER JOIN Link ON Movie.Movie_Id = Link.Movie_Id;
```

```
178 •   SELECT Movie.Title, Link.IMDB_Id
179     FROM Movie
180     INNER JOIN Link ON Movie.Movie_Id = Link.Movie_Id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Title	IMDB_Id
Toy Story (2001)	114709
Jumanji (1995)	113497
Grumpier Old Men (1998)	113228
Waiting to Exhale (1990)	114885
Father of the Bride Part II (1985)	113041
Heat (1966)	113277
Sabrina (2002)	114319
Tom and Huck (2000)	112302
Sudden Death (1995)	114576
GoldenEye (1995)	113189
American President, The (1997)	112346
Dracula: Dead and Loving It (1991)	112896
Balto (1989)	112453
Nixon (1988)	113987
Cutthroat Island (2008)	112760

4. **Right Join to get all users and the movies they rated.**

Command-

SELECT Ratings.User_Id, Movie.Title, Ratings.Rating
FROM Ratings
RIGHT JOIN Movie ON Ratings.Movie_Id = Movie.Movie_Id;

```
182 •  SELECT Ratings.User_Id, Movie.Title, Ratings.Rating
183    FROM Ratings
184    RIGHT JOIN Movie ON Ratings.Movie_Id = Movie.Movie_Id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

User_Id	Title	Rating
1081	Toy Story (2001)	4
1097	Jumanji (1995)	4
1056	Grumpier Old Men (1998)	4
1053	Waiting to Exhale (1990)	5
1040	Father of the Bride Part II (1985)	5
1028	Heat (1966)	3
1015	Sabrina (2002)	5
1003	Tom and Huck (2000)	4
990	Sudden Death (1995)	5
978	GoldenEye (1995)	5
965	American President, The (1997)	5
953	Dracula: Dead and Loving It (1991)	5
940	Balto (1989)	3
928	Nixon (1988)	5
915	Cutthroat Island (2008)	4

Result 18

5. **Left Join to get all movies and their tags.**

Command-

SELECT Movie.Title, Tags.Tag
FROM Movie
LEFT JOIN Tags ON Movie.Movie_Id = Tags.Movie_Id;

```
SELECT Movie.Title, Tags.Tag
FROM Movie
LEFT JOIN Tags ON Movie.Movie_Id = Tags.Movie_Id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Title	Tag
Toy Story (2001)	Funny
Jumanji (1995)	Funny
Grumpier Old Men (1998)	Romantic
Waiting to Exhale (1990)	Drama
Father of the Bride Part II (1985)	Funny
Heat (1966)	Thriller
Sabrina (2002)	Comedy
Tom and Huck (2000)	Adventure
Sudden Death (1995)	Action
GoldenEye (1995)	Thriller
American President, The (1997)	Drama
Dracula: Dead and Loving It (1991)	Horror
Balto (1989)	Children
Nixon (1988)	Drama
Cutthroat Island (2008)	Action

Result 19

CONCLUSION

This database provides a flexible and scalable way to manage movie information, user interactions (ratings and tags), and external links. It supports a wide range of queries, allowing users to analyze data such as movie popularity, genre distribution, or user preferences. By effectively organizing the data with primary and foreign keys, it ensures data integrity and efficient retrieval, making it suitable for use in applications like movie recommendation systems or analytics dashboards.

Flexibility and Scalability:

The design allows for the seamless integration of additional data types or larger datasets. For example:

- Scalability: As the number of movies or users grows, the database can handle larger datasets without performance degradation. You could easily add more movies, users, or new features, such as reviews or streaming availability, without needing to fundamentally alter the structure.
- Flexibility: The relational structure ensures that adding new attributes or expanding the system, like incorporating more rating systems or genres, can be done with minimal disruption. If new genres or tags need to be added, it can be done without restructuring the core tables.

This database is a robust and scalable system designed to effectively manage movie information, user interactions (through ratings and tags), and external links to widely-used movie databases like IMDb and TMDb. By employing efficient relational database principles, this system allows for streamlined data management and analysis, offering powerful capabilities for a wide range of applications such as recommendation engines, user behavior analysis, and content management.