# Database Phase 1

https://grouplens.org/datasets/movielens/
1st database: https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset
2nd database:
https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=keywords.csv

1. Who are your team members (normally a maximum of 2 people, but with prior approval teams of 3 may be allowed if their project is sufficiently complex and the grading will be based on this expectation of greater complexity and substance).

Team members: Jacob Choi, Sanghyup Lee

2. Briefly describe your target domain (e.g. a world geographic database)

Our target domain is the user-movie interaction database taken from the MovieLens 20M dataset. The dataset represents the domain of movie ratings and tagging activities for a recommendation system, which captures over 20 million movie ratings and 465,000 tags from 138,000 users from 27,000 movies from 1995 to 2015. It includes data such as titles, genres, and IMDB ratings, which allows users to analyze user preferences and genre trends over time.

3. Give a reasonably comprehensive and representative list of the kinds of English questions you would like your system to be able to answer

(1st dataset)

a. Top 10 all time movie favorites
b. Depending on the user's view history, recommend movies accordingly
c. Average of a specific user's rating
d. Top rated movies in each year
e. Rank the users by the number of reviews and the movies watched
f. Group users with the similar genre taste in movies
g. Directors and what the average of their movie ratings are
h. Sort the ratings by a time frame and give an analysis on it
i. Top rated movies for each genre

(2nd dataset)

j. Movies with the highest ROI based on budget and revenue
k. Most frequent keywords in top-rated movies
l. Actors and actresses who appear frequently in top-rated movies
m. Average movie ratings across different budget ranges
n. Directors with the most success in terms of revenues and ratings
o. Genres with the highest revenues and ratings

4. Design and show a relational data model that you plan to use for your system, with a preliminary implementation in standard SQL data-definition-language syntax. This specification should include appropriate primary key, foreign key and domain specifications for each relation/attribute, as well as the not null constraint when appropriate. You may also find it useful, but not required, to create a few insert-into statements that populate your schema designs with representative values (both to document your choices and to exercise them. You are welcome to change and augment your design and its specification by Phase II, but any time investment now will reduce effort later.

```
CREATE TABLE movies (
        movieId INT PRIMARY KEY,
        title VARCHAR(255) NOT NULL,
        genres VARCHAR(255) NOT NULL
);

INSERT INTO movies (movieId, title, genres) VALUES
(1, 'Toy Story (1995)', 'Adventure|Animation|Children|Comedy|Fantasy'),
(2, 'Jumanji (1995)', 'Adventure|Children|Fantasy');
```

```
CREATE TABLE links (
        movieId INT PRIMARY KEY,
        imdbId VARCHAR(10),
        tmdbId INT,
        FOREIGN KEY (movieId) REFERENCES movies(movieId)
);

INSERT INTO links (movieId, imdbId, tmdbId) VALUES
(1, 'tt0114709', 862),
(2, 'tt0113497', 8844);
```

```
CREATE TABLE genome_tags (
        tagId INT PRIMARY KEY,
        tag VARCHAR(255) NOT NULL
);

INSERT INTO genome_tags (tagId, tag) VALUES
(1, 'funny'),
(2, 'action'),
(3, 'drama');
```

```
CREATE TABLE genome_scores (
        movieId INT NOT NULL,
        tagId INT NOT NULL,
        relevance FLOAT NOT NULL,
        PRIMARY KEY (movieId, tagId),
        FOREIGN KEY (movieId) REFERENCES movies(movieId),
        FOREIGN KEY (tagId) REFERENCES genome_tags(tagId)
);
```

```
INSERT INTO genome_scores (movieId, tagId, relevance) VALUES
(1, 1, 0.8),
(1, 2, 0.1),
(2, 1, 0.5),
(2, 2, 0.6);
```

```
CREATE TABLE ratings (
        userId INT NOT NULL,
        movieId INT NOT NULL,
        rating FLOAT NOT NULL,
        timestamp INT NOT NULL,
        PRIMARY KEY (userId, movieId),
        FOREIGN KEY (movieId) REFERENCES movies(movieId)
);

INSERT INTO ratings (userId, movieId, rating, timestamp) VALUES
(1, 1, 4.0, 964982703),
(1, 2, 5.0, 964982224);
```

```
CREATE TABLE tags (
        userId INT NOT NULL,
        movieId INT NOT NULL,
        tag VARCHAR(255) NOT NULL,
        timestamp INT NOT NULL,
        PRIMARY KEY (userId, movieId, tag, timestamp),
        FOREIGN KEY (movieId) REFERENCES movies(movieId)
);

INSERT INTO tags (userId, movieId, tag, timestamp) VALUES
(1, 1, 'pixar', 964982703),
(1, 2, 'fantasy', 964982224);
```

5. Optionally, you may submit a set of SQL statements that will implement a representative sample of your target queries, including some of the more interesting or challenging cases. This is primarily to get you to think about your design and how it will be exercised as well as any limitations, so focus on queries that would be useful for doing so, rather than creating trivial or non-insightful queries just to fill space

Find the titles of movies along with their genres and the most relevant tags (relevance > 0.8), sorted by movie title.

```
SELECT m.title AS MovieTitle, m.genres AS Genres, gt.tag AS Tag, gs.relevance AS Relevance
FROM movies m
JOIN genome_scores gs ON m.movieId = gs.movieId
JOIN genome_tags gt ON gs.tagId = gt.tagId
WHERE gs.relevance > 0.8
ORDER BY m.title;
```

6. Provide a plan for how you will load the database with values

To load our database with values from the MovieLens 20M dataset, we will download the underlying CSV files from Kaggle, including ratings.csv, tags.csv, movies.csv, links.csv, genome_scores.csv, and genome_tags.csv. These files will be inspected and be cleaned of any empty or null values during our pre-processing step. Then, these values will be loaded onto our database using MySQL.

7. Very briefly describe the form/type of output or result you plan to generate or any special user interface issues (e.g. views) that you plan to implement.

The planned output will include a user-friendly interface that allows students and administrators to view and interact with movie data effectively. This will involve generating web pages displaying movie catalog, personalized movie recommendations, and detailed movie information. Special user interface features will include search and filter options and a recommendation engine based on user preferences and ratings. The interface will be built using PHP to generate our web pages and handle the backend, while SQL procedures will be used to retrieve the data.

8. What are the specialized/advanced topics you plan to focus on in your database design?

Advanced SQL Topics:

- Complex Queries: Writing advanced SQL queries to generate personalized movie recommendations based on user ratings and preferences.

Optimization/Tuning:

- Query Optimization: Ensuring efficient data retrieval for high-performance search and recommendation functionality, especially with large movie datasets. This includes indexing strategies and analyzing query execution plans.
- Database Performance: Fine-tuning MySQL database settings to ensure that the system can scale and deliver quick response times as the user review grows.

# Database Phase 2

## Movie Recommendation System

The application domain focuses on a movie recommendation system built using the MovieLens 20M dataset. By analyzing user-movie interactions, including ratings, tags, and genres, the system aims to enhance user experience in the movie search process by offering tailored recommendations and detailed movie statistics.

Sanghyup Lee slee548@jhu.edu (315)
Jacob Choi echoi51@jhu.edu (415)

(3) If your project description or database design has changed since your Phase I submission, describe these changes or additions.

Since our Phase I submission, we have introduced several enhancements to the application to improve user experience, performance, and insights. These include:

- **Graph-Based Visualizations**: We added graphical representations of movie trends over time, such as the number of reviews per year, to help users understand viewing patterns and shifts in popularity.
- **Enhanced SQL Queries**: Our SQL queries have been refined for greater efficiency and reduced complexity, enabling faster responses and better handling of large datasets.
- **Caching Mechanisms**: To improve performance, we implemented caching for frequently searched terms, significantly reducing load times for repeat queries.
- **Additional Functionalities**: New features include recommendations based on genres, time-based filtering, and an option to find similar movies using both genre and tag-based similarity metrics.

(4) Briefly describe how you loaded the database with values, including the sources of your data (e.g. URL's) and pointers to any specialized code you developed for preprocessing, extracting or loading the data.

We populated our database using the MovieLens 20M dataset, a comprehensive resource for movie-related data. To automate the process, we developed a Python script tailored for this dataset. Key steps included:

1. The script generated six relational tables—`genome_scores`, `genome_tags`, `link`, `movie`, `rating`, and `tag`—following a predefined schema.
2. Before insertion, the script handled missing or malformed entries, ensuring data consistency and integrity.
3. CSV files from the MovieLens dataset were read and processed into batches. Each batch was inserted into the database using parameterized SQL queries to prevent SQL injection and ensure efficiency.
4. The script logged progress every 1,000 rows and executed commits in 5,000-row batches to minimize memory overhead and maintain transactional consistency.
5. Comprehensive error handling mechanisms were built into the script to log and skip problematic rows without halting execution.

(5) If you did not use mysql on dbase.cs.jhu.edu, describe the exact software/hardware platform you used.

We used MySQL as the database management system, integrated with Python scripts for data processing and interaction. The frontend of the project was built using HTML, CSS, and JavaScript to provide an intuitive user interface. The backend leveraged Flask to create APIs that interact with the MySQL database. The setup was executed on a local development environment, which included Python 3.8, Flask, and MySQL.

(6) Provide a brief user's guide, describing in sufficient detail how a user could run your code, especially if using a non-standard platform or interface. If the comments in your code are sufficiently detailed, a pointer to them will be adequate.

Any detail on how to run our code is included in our README file in our submission.

(7) List and very briefly describe your major/minor areas of specialization, such as (1) complex extraction of real data from online sources, (2) an interesting interface demonstrating significant accomplishment, expanded education and/or originality, (3) original and challenging data modeling (e.g. for genomic, geographic or multimedia data), (4) security, (5) transaction control, (6) natural language interfaces, or (7) datamining and knowledge discovery.

Expertise includes automating the extraction and preprocessing of large-scale datasets, demonstrated through work on the MovieLens 20M dataset. Python scripts and parameterized SQL queries were utilized to handle real-world data inconsistencies, ensuring clean and structured data for integration into relational databases. This highlights proficiency in managing complex data extraction and preprocessing tasks.

An innovative web interface was designed and implemented using HTML, CSS, and JavaScript, integrated with a Flask backend. The interface supports interactive features such as movie search, top-rated movie exploration, and visual insights into viewing trends. This work showcases originality and significant accomplishment in creating seamless user experiences while effectively connecting frontend and backend systems.

Challenging data models were developed for multimedia datasets, including relational database schemas optimized for movie and user data. The models incorporate tag-based and genre-based similarity calculations to personalize movie recommendations, enabling precise and scalable results. This demonstrates specialization in complex data modeling and knowledge discovery.

(8) Give an itemized list of your project's particular strengths or selling points, especially things that may not be obvious upon inspection of your code or output.

1. Comprehensive Functionality
    a. Offers features like movie search, top-rated movie exploration, similar movie recommendations, and graphical trends of user reviews over time, delivering an engaging and well-rounded user experience.
2. Advanced Recommendation System
    a. Implements hybrid similarity algorithms combining tag relevance, genre similarity, and user rating thresholds to produce highly personalized and accurate movie recommendations.
3. Database Index Optimization
    a. Optimized database design with carefully chosen indexes on frequently queried fields (e.g., movie titles, genres, and ratings) to significantly improve query execution speed and overall system performance.
4. Efficient SQL Queries
    a. Refined queries to minimize complexity, leverage indexed fields, and reduce response times for operations on large datasets.
5. Scalable Data Modeling
    a. Designed relational database schemas to handle large-scale data efficiently, ensuring smooth performance even with millions of records.
6. Visual Insights
    a. Introduced graph-based visualizations, such as review trends over time, to provide users with actionable insights beyond standard outputs.
7. User-Friendly Interface
    a. Features an intuitive web interface with expandable movie details, filter-based searches, and real-time feedback for better usability and engagement.
8. Secure Backend
    a. Ensures safe and reliable database interactions using parameterized SQL queries to prevent SQL injection attacks.
9. Robust Data Cleaning
    a. Performed comprehensive preprocessing and cleaning of the MovieLens dataset to ensure high-quality and consistent data for analysis.

(9) Give a brief itemization of your project's limitations and list suggested possibilities for improvement or worthwhile extension with additional time.

1. Limited Data Source
    a. Limitation: The system is currently limited to the MovieLens 20M dataset, which doesn't include external or more recent data such as current movie releases or reviews

    b. Improvement: Integrate real-time external data through APIs like the open source IMDb APIs to provide a more up-to-date recommendations based on current trends
2. Lack of Frontend Features
    a. Limitation: The frontend interface, although includes all necessary components to be a fully functional recommendation system, is relatively simple and may not be visually appealing for more advanced use cases.
    b. Improvement: Enhance the UI/UX design by adding more features like real-time movie previews or more interactive elements to create a richer user experience
3. Lack of User Feedback Mechanism
    a. Limitation: Our system currently does not have any mechanism where it can receive direct user feedback like thumbs up/down that would enhance the recommendation quality
    b. Improvement: Add user feedback mechanisms where users can rate the recommendation quality of our application

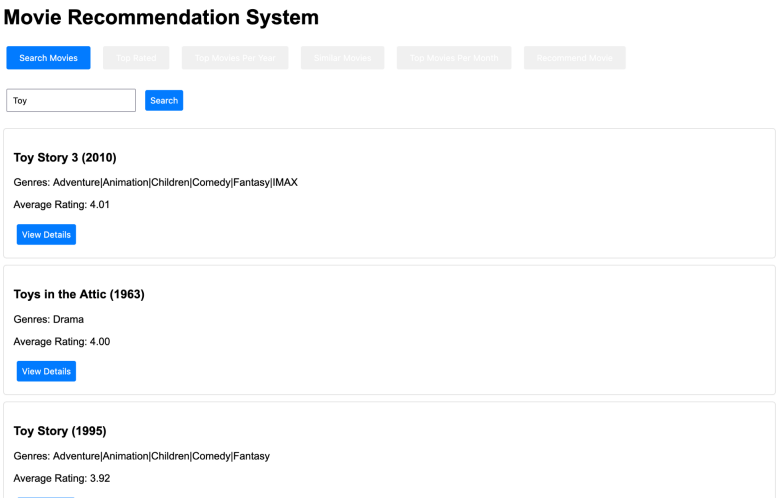
(10) If your code submission includes components that were (a) written by people not on your team or (b) written by yourself for another course, for prior research and/or employment, please very clearly explain/document which components were done elsewhere. Such outside borrowing is permitted if clearly documented, and you are permitted to build upon prior accomplishment, just like you may utilize software libraries from elsewhere, but your project will be evaluated on the original work done for this course.

Our code is original.

(11) Include a set of reasonably formatted “output” from your project, appropriate for the focus of your project, and demonstrating interesting and challenging extraction, synthesis and summarization of information in the database and supporting your specialized topics, with sufficient titles and or notes to make it clear what the output represents. The nature of the output will depend on what is the the focus of your project. Ample screenshots that show off all aspects of your project are strongly recommended to make sure all aspects of your project’s accomplishments are clear, and you may provide a live demo (via recorded video or in person/zoom) if you feel appropriate.

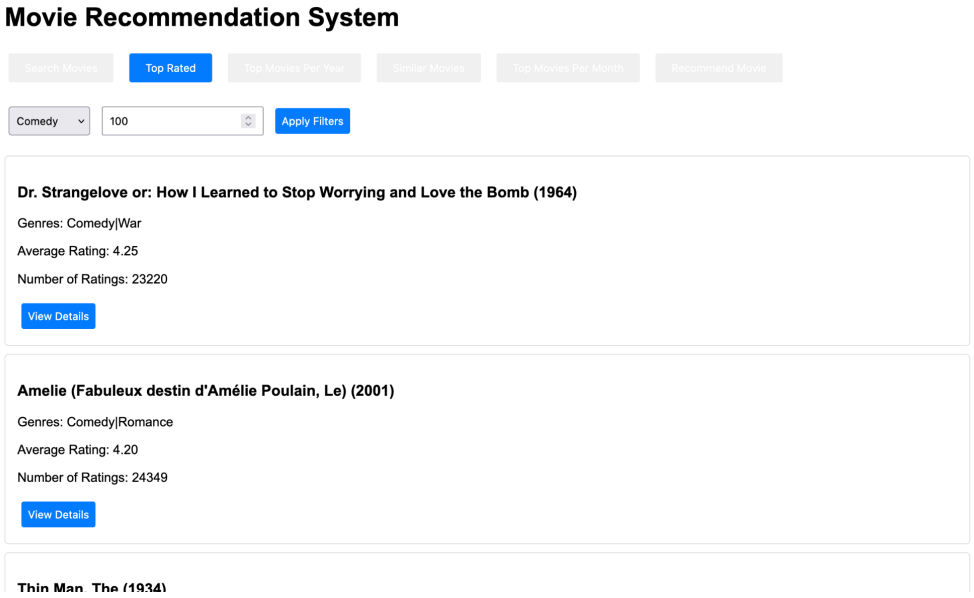
### Movie Search Results

Displays movies matching the search query, with detailed information about genres, average ratings, and the number of ratings.



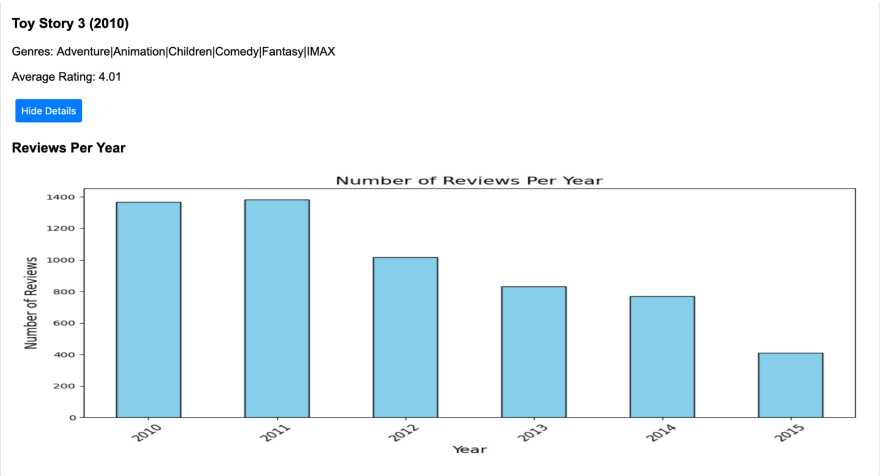

### Top-Rated Movies

Lists the highest-rated movies with genre-based filtering and minimum rating thresholds.



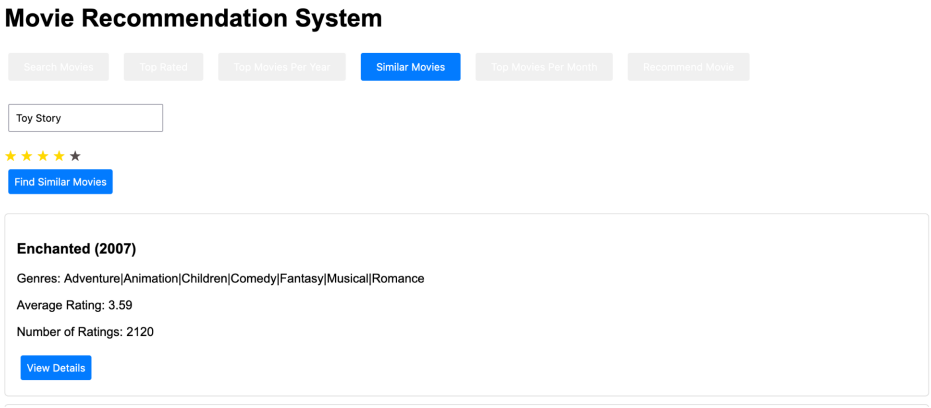

### Graphical Visualization of Review Trends

A bar chart visualizing the number of reviews for a selected movie over time.



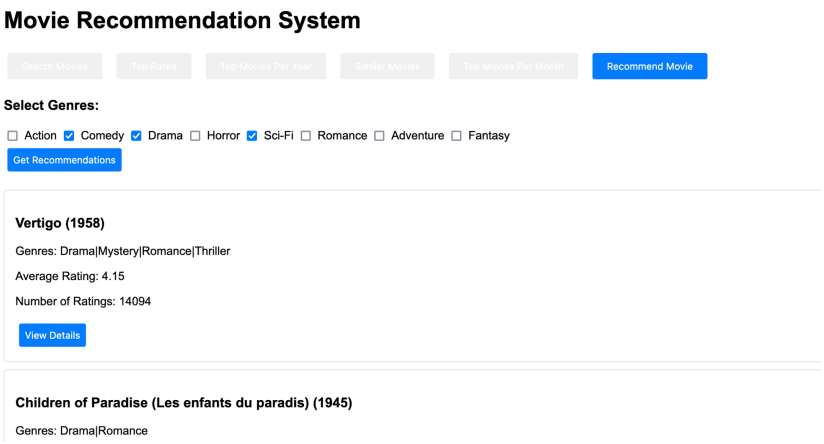

### Similar Movie Recommendations

Displays movies similar to the selected title based on genre and tag similarity, including a calculated similarity score and matching tags.



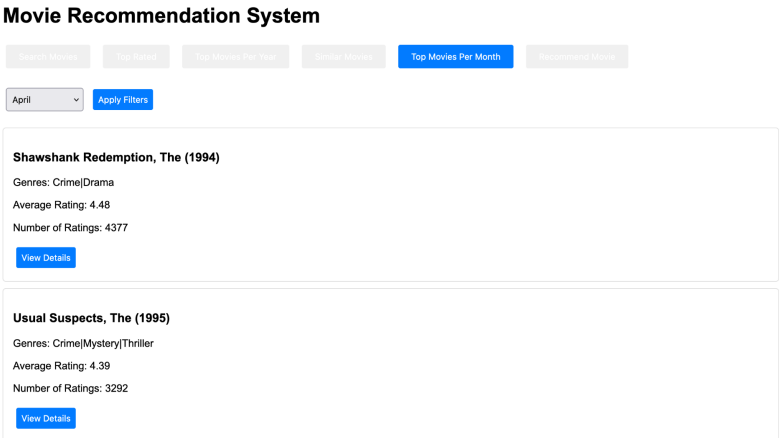

### Recommendations by Genre

Provides a curated list of movies based on user-selected genres.




**Top Movies by Season**

Highlights movies popular during specific seasons, tailored to seasonal viewing patterns.




(12) Submit a full relational table specification of your database in the SQL Database Definition Language (DDL). This specification should include both the data type of each attribute, the not null constraint when appropriate and sample data values for each attribute represented as comments. You should also specify the primary keys (e.g. primary key (ssn)) and referential constraints (e.g. foreign key (mgrssn) references employee(ssn)). Many groups included this specification in their project phase I writeup, but please update it if it has changed.

Included within section12.sql.

(13) Include a copy of all SQL code used in your system (commented appropriately), as well as any additional code you used for data acquisition and input.

Stored in Query.py and procedure.sql.

(14) You are required to submit a complete archive of all your code (SQL/stored-procedures/python/PHP/etc.) and data files, ideally in a single compressed nested directory. Your directory should be named as follows to make it easier for me to identify when I uploaded it: yourgivenname yoursurname if done individually (e.g. john doe.zip) or partner1givenname partner1surname partner2givenname partner2surname (e.g. john doe adam smith.zip)