THE MOVIE DATABASE (TMDB) ANALYSIS

|  |  |  |
| --- | --- | --- |
| Nithin Keshavamurthy  *Department of Applied*  *Data Science*  *San Jose State*  *University* nithin.keshavamurthy  @sjsu.edu | Karthik Shetty  Nagaraja  *Department of Applied*  *Data Science*  *San Jose State*  *University* karthikshetty.nagaraja  @sjsu.edu | Varun Reddy Bhumi  Reddy  *Department of Applied*  *Data Science*  *San Jose State*  *University* varunreddy.bhumireddy  @sjsu.edu |

***Abstract*—This project demonstrates the transition of The Movie Database (TMDB) to a cloud-based platform, leveraging MongoDB to enhance data management and analysis. Using Python for data processing, the project involves cleaning and transferring metadata for 45,000 movies, including user ratings, into a MongoDB database. This transition underscores the scalability and accessibility benefits of a cloud-native database solution, illustrating how it can revolutionize data handling and analytics within the film industry. MongoDB's flexible schema and powerful indexing capabilities enable dynamic queries and efficient data retrieval, making it ideal for handling complex datasets like TMDB.**

***Keywords- TMDB*** ***NoSQL Database Architecture, MongoDB Migration, Production Company Data Analysis, , Denormalization Techniques, Database Scalability, Movie Database Querying***

# PROBLEM STATEMENT

Our initiative aims to uncover the intricate relationships between movie success and audience engagement within the extensive TMDB dataset, focusing on the links between a film's box office performance, critical acclaim, and key factors such as cast and genre. We will analyse trends in box office earnings, budget distributions, and the influence of production sources and genres on a film’s financial success. For this analysis, we will utilize MongoDB's powerful query capabilities. By employing MongoDB's flexible document structure and rich querying options, we can efficiently explore complex data relationships. This data-driven approach is expected to provide producers, marketers, and content creators in the film industry with actionable insights, enhancing their understanding of the elements that contribute to a movie's critical and commercial success..

# SOLUTION REQUIREMNTS

***System Requirements***

For processing queries, we require a PC with 8GB of RAM. T

## TMDB Database Management System on MongoDB

The TMDB Database Management System, transitioned to MongoDB, is designed for sophisticated analysis and management of large volumes of film metadata. Utilizing MongoDB's flexible document schema, the system efficiently organizes and connects various film-related items without the strict requirements of traditional relational databases. This structure ensures data integrity and facilitates complex relationships among data elements. Key features of this system include extensive logging capabilities, potential integration with dashboards for enhanced data visualization, and interactive user interfaces. Optimized for robust data handling and efficient querying, MongoDB's architecture leverages its performance and scalability to support comprehensive studies of the film industry.

***Data Extraction***

Data was downloaded from Kaggle ([The Movies Dataset (kaggle.com))](https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=credits.csv).

## Data Wrangling Steps

1. **Importing Libraries:** The code begins by importing necessary Python libraries for data manipulation such as NumPy and pandas, and literal\_eval from the ast module for evaluating strings containing Python literal structures.
2. **Reading CSV Files:** Several CSV files, including 'credits.csv', 'keywords.csv', 'links.csv', 'ratings\_small.csv', and 'movies\_metadata.csv', are read into pandas DataFrames.
3. **Cleaning ID Columns:** The 'id' column of the 'movies\_metadata.csv' DataFrame is converted to a numeric type, and any errors in conversion are coerced to NaN, which are then dropped from the DataFrame.
4. **Cleaning IMDb ID:** Similar to the 'id' column, the 'imdb\_id' column is stripped of its 'tt' prefix, and converted to a numeric type. Any non-numeric values are coerced to NaN.
5. **Merging DataFrames:** The 'data' DataFrame (from 'movies\_metadata.csv') is merged with 'credits' and 'keywords' on the 'id' column. It's also merged with 'links' on both 'id' (as 'movieId') and 'imdb\_id' (as 'imdbId').
6. **Dropping Duplicates:** The DataFrame undergoes a duplication check and any duplicates are removed to maintain data integrity.

Dropping Null Values: Null values are removed from specific columns such as 'cast', 'crew', 'keywords', and 'popularity' to ensure completeness of the dataset.

1. **Parsing and Cleaning Specific Columns:** Columns that contain JSON objects or arrays (like 'budget', 'genres', 'production\_companies', 'production\_countries', 'spoken\_languages', 'keywords', 'cast', and 'crew') are parsed using literal\_eval or ast.literal\_eval. They are cleaned and structured so that they contain either arrays, dictionaries, or NaN where appropriate. The cast and crew columns are trimmed to only include the first three entries, if available.
2. **Saving Cleaned Data:** Finally, the cleaned and merged data is saved to a new CSV file named 'Merged\_data\_modified.csv'.

## Denormalization

To enhance data accessibility and simplify operations within the MongoDB environment, we implemented a denormalization approach. The credits dataset, which previously separated cast and crew, was consolidated into detailed movie documents. This allows each movie entry to provide immediate access to all associated cast and crew information. Additionally, movie metadata such as collections, genres, production companies, production countries, and spoken languages were embedded directly into the movie documents. This denormalization strategy circumvents the need for multiple joins and relational mappings required in SQL databases, adopting a document-centric approach that conforms to the first normal form (1NF) by incorporating arrays and sub-documents. This method significantly improves the efficiency of read operations and query performance, ensuring a robust and scalable data model.

***System Limitations***

The presence of numerous duplicate entries presented challenges in establishing unique identifiers and maintaining data integrity. The task of parsing deeply nested JSON data to fit our schema-less database architecture placed significant demands on our processing capabilities. Constraints due to limited hardware resources impeded the ability to execute concurrent data processing tasks effectively. Moreover, MongoDB’s absence of built-in transactional support made it difficult to ensure data consistency during operations, requiring intricate solutions to achieve atomicity and coherence. These challenges necessitated creative approaches to sustain the system’s dependability and efficiency, given our technological constraints.

## End Users

This initiative supports a broad range of participants in the film sector. Enthusiasts of cinema obtain in-depth data about distinct production companies, enhancing their insights into film trends. Scholars in academia make use of the system to access a wealth of movie metadata for academic exploration. Professionals in the industry turn to our platform for vital intelligence that informs their strategic decisions and market evaluations. Furthermore, media analysts employ this tool to compile detailed analyses, expanding their understanding of film trends and audience tendencies. This tool proves crucial for various individuals who require detailed and actionable insights from the film industry.

# PROJECT REQUIRMENTS

TMDB database implementation on MongoDB provides a well-organized analysis of film-related data, highlighting the intricate relationships among various segments of the cinema industry. By leveraging MongoDB, our database maintains high data integrity and enhances operational efficiency. This is achieved through automated updates to key movie metrics, such as social media interactions and viewer ratings, which are crucial for tracking the ongoing success and popularity of films across different genres.

## Present a well-documented ER diagram with proper normalization

***Document structure, access privileges, and usage of queries and aggregation functions*.**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

*Fig 1. Document structure*

***Access Privileges:***

I. Log in to the MongoDB Atlas dashboard with administrative credentials. Navigate to the "Database Access" section under the "Security" tab to manage database users. Create a new MongoDB user, assigning appropriate roles such as "readWriteAnyDatabase" or more granular permissions as required.

II. Specify the IP Whitelist under the "Network Access" tab to define which IP addresses are allowed to connect to the database clusters. This can be set to the IP address of the remote machine specifically, or a range of IP addresses if multiple access points are needed.

III. In the MongoDB Atlas interface, ensure that the database user you created is granted access to the necessary clusters within the project. Assign or confirm roles that provide the required level of access for the user’s responsibilities.

IV. On the remote machine, install a MongoDB management tool such as MongoDB Compass or use the integrated MongoDB shell. Configure a new connection by inputting the cluster's hostname or IP address, along with the necessary port (default is usually 27017 for MongoDB). Use the credentials of the MongoDB user created for remote access.

V. Project access can be generated from Mongodb Atlas for each induvial user who needs access.

A screenshot of a computer

Description automatically generated

*Fig 3. Database Access given to each member*

A screenshot of a computer

Description automatically generated

*Fig 3. Project Access given to each member*

A screenshot of a computer

Description automatically generated

*Fig 3. Network Access given to each member*