**IMDb Data Analysis and Management with MongoDB Compass**

In this project, I embarked on a journey to harness the power of MongoDB Compass for the structured organization and in-depth analysis of IMDb movie data. The primary objective was to create a robust database system that stores and facilitates advanced analytical operations on movie titles and their corresponding ratings. From the initial stages of database and collection creation to the intricate tasks of schema validation and data importation, the project encapsulated a comprehensive exploration of MongoDB functionalities. Furthermore, I delved deep into schema analysis, extracting meaningful insights about the characteristics of the stored data. The pinnacle of this endeavour was the advanced data aggregation tasks, where I unearthed specific trends and patterns, such as the popularity of Fantasy-Adventure movies in 2021. This project stands as a testament to my proficiency in database management and data analytics using MongoDB Compass**.**

Table of Contents

[1. Database Creation using Compass Tool 2](#_Toc143242733)

[2. Collection Creation for Data Organization 2](#_Toc143242734)

[3. Schema Validation for Data Integrity 3](#_Toc143242735)

[Schema Validation: titles.csv 3](#_Toc143242736)

[Schema Validation: ratings.csv 5](#_Toc143242737)

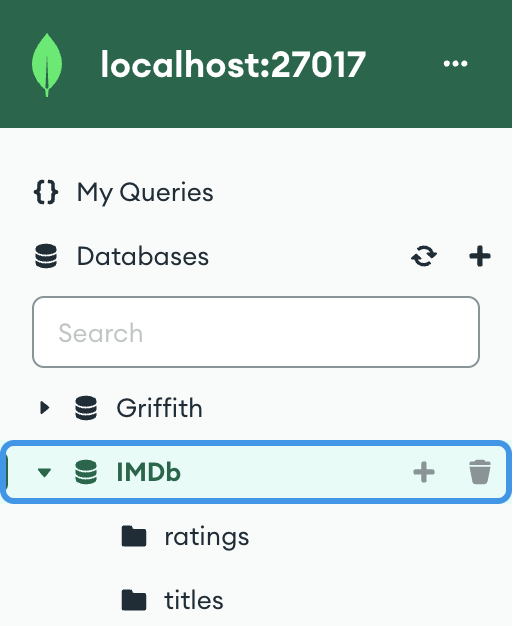
[4. Data Importation using Compass 6](#_Toc143242738)

[5. Schema Analysis and Data Characteristics Description 7](#_Toc143242739)

[6. Advanced Data Analysis using Aggregation 9](#_Toc143242740)

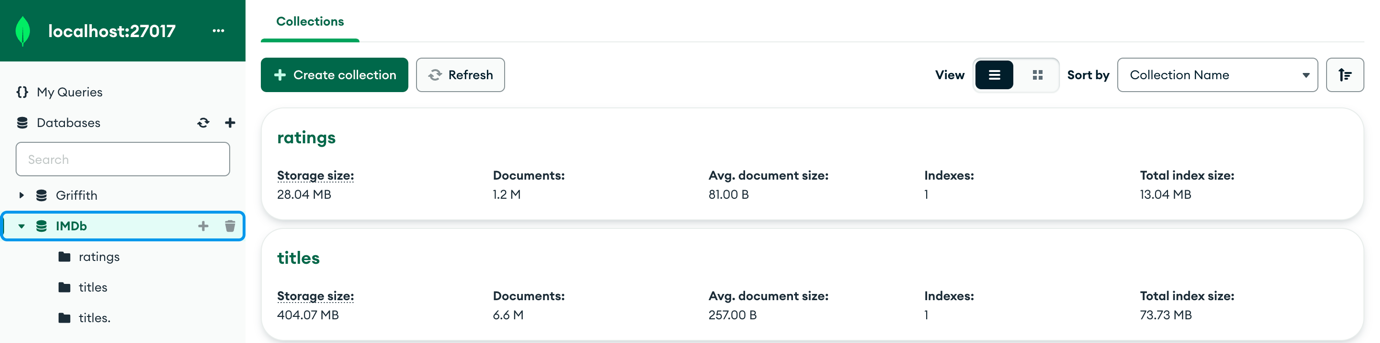
## **1. Database Creation using Compass Tool**

* Established a new database titled “imdb” utilizing the MongoDB Compass tool, demonstrating proficiency in database setup and management.



## **2. Collection Creation for Data Organization**

* Developed two distinct collections within the “imdb” database:
  + “titles”: This collection is designed to store data from the file title.basics.tsv.gz.
  + “ratings”: Specifically for data derived from title.ratings.tsv.gz, ensuring structured data storage.



## **3. Schema Validation for Data Integrity**

* Crafted comprehensive JSON Schemas for both “titles” and “ratings” collections. This was done to ensure data consistency and adherence to predefined structures, based on the descriptions provided.

## **Schema Validation: titles.csv**

Comment:

{

$jsonSchema: {

bsonType: 'object',

title: 'titles',

required: [

'tconst',

'primaryTitle'

],

properties: {

tconst: {

bsonType: 'string',

description: 'alphanumeric unique identifier of the title, required'

},

titleType: {

bsonType: 'string',

description: 'the type/format of the title (e.g., movie, short, tvseries, tvepisode, video, etc.)'

},

primaryTitle: {

bsonType: 'string',

description: 'the more popular title / the title used by the filmmakers on promotional materials at the point of release'

},

originalTitle: {

bsonType: 'string',

description: 'original title, in the original language'

},

isAdult: {

bsonType: 'int', // using int as Boolean is used for true/false and would cause error

'enum': [

0,

1

],

description: '0: non-adult title; 1: adult title'

},

startYear: {

bsonType: 'number',

description: 'represents the release year of a title. In the case of TV Series, it is the series start year, `N` if missing'

},

endYear: {

bsonType: 'number',

description: 'TV Series end year. ‘N’ for all other title types, `N` if missing'

},

runtimeMinutes: {

bsonType: 'number',

description: 'primary runtime of the title, in minutes, `N` if missing'

},

genres: {

bsonType: 'string',

description: 'includes up to three genres associated with the title',

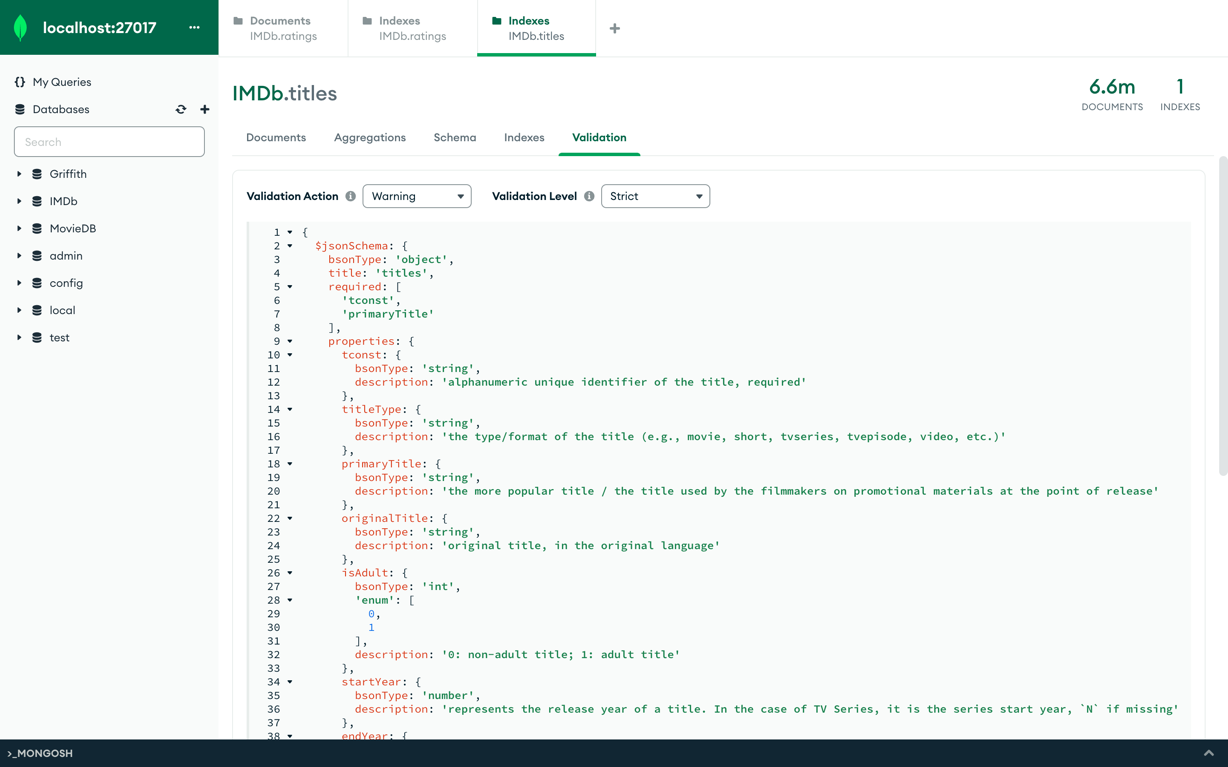
pattern: '^[^,]\*(,[^,]\*){0,2}$' // this pattern ensures that the genres are comma-separated, and there are at most two commas, meaning up to three genres

}

}

}

}



A screenshot of a computer

Description automatically generated

## **Schema Validation: ratings.csv**

{

$jsonSchema: {

bsonType: 'object',

title: 'ratings',

required: [

'tconst',

'averageRating'

],

properties: {

tconst: {

bsonType: 'string',

description: 'alphanumeric unique identifier of the title, required'

},

averageRating: {

bsonType: 'number',

description: 'weighted average of all the individual user ratings and is required'

},

numVotes: {

bsonType: 'number',

description: 'number of votes the title has received'

}

}

}

}

A screenshot of a computer

Description automatically generated

## **4. Data Importation using Compass**

* Successfully imported data into the respective collections:
  + Inserted data from titles.csv into the “titles” collection.
  + Integrated data from ratings.csv into the “ratings” collection. This showcases my capability in data migration and integration using MongoDB Compass.

Titles.csv as IMDb.titles

A screenshot of a computer

Description automatically generated

Ratings.csv as IMDb.ratings

A screenshot of a computer

Description automatically generated

## **5. Schema Analysis and Data Characteristics Description**

* Conducted an in-depth schema analysis using the “Schema” tab in Compass.
* Engaged the “Schema analysis” feature, which generated a comprehensive analysis for each column in the collections.
* Identified and described notable characteristics of the data within each column, ensuring a thorough understanding of the dataset. For more on schema analysis, I referred to the [MongoDB Compass documentation](https://docs.mongodb.com/compass/master/schema/).

In a comprehensive dataset encompassing 6.6 million files, a methodical approach was adopted using MongoDB Compass Schema to analyse a sample of 1,000 documents. This streamlined approach not only enhances efficiency but also provides a microcosm of the broader content landscape.

* **Data Integrity and Gaps:** The titles.csv file presents a challenge in data consistency. A significant number of documents are devoid of specific values, represented by the placeholder "\N". This also became evident during the schema validation process, as many documents did not pass the validation test. However, since the validation action was set to ‘warning’, we can still analyse the information provided by these documents. Particularly in the fields of genre, runtimeMinutes, startYear and endYear. This absence underscores the importance of data cleaning for a more holistic and accurate analysis.
* **Content Production Renaissance:** The data reveals a pronounced uptick in content production. A notable 29% of the sample originates from 2015-2020, with 2020 alone accounting for 18% of their subset. This starkly contrasts the 4% from the 1995-2000 era. This surge is congruent with external research, such as Klatt’s (2022) findings, which highlight the transformative role of Big Tech in the content production domain. Industry behemoths like Amazon and Netflix have emerged as primary catalysts for this boom.
* **The dominance of TV Episodes:** The sample underscores the ascendancy of TV episodes, which constitute a whopping 69%. This trend reflects the strategic direction of new-age streaming platforms, as elucidated by Shattuc (2020), which are increasingly veering towards episodic content. This preference is further mirrored in the prevalent runtimes: 30 minutes (6%), 22 minutes (6%), and 60 minutes (5%).
* **Movies - The Evergreen Contender:** Movies, while second to TV episodes, accounting for 11 % of the sample, have showcased resilience. Looking more closely at the subset of ‘movies’, we can see it has grown from a mere 0.7% in 1945 to 22% between 2015 and 2020, with 2019 being the year from this subset with the most movies at 17%. Moreover, the majority, 59%, fall within the 80-120 minute runtime bracket, with Drama emerging as the predominant genre at 22%.
* **Rise of Short Movies:** Short movies, though third in line, have carved a niche for themselves. There's been a discernible increase post-2000, with 68% clocking in under 15 minutes. This category has fostered unique genres, with “Comedy, Short”, “Short”, “Documentary, Short”, and “Drama, Short” collectively accounting for 58% of this segment. While we do not have data on the content platform, research support trends in shorter videos becoming increasingly popular (Wang, 2020).
* **Content Classification:** A vast majority, 98%, of the content is non-adult, underscoring an apparent industry inclination or perhaps a reflection of broader audience preferences.
* **Audience Ratings and Engagement Metrics:** The ratings.csv offers a window into audience preferences and feedback. Titles rated between 7 and 7.5 lead at 18%, closely trailed by those in the 7.5-8 and 8-8.5 brackets, each at 14%. However, a mere 2% clinched top ratings between 9.5 and 10. In terms of audience engagement, while one title garnered an impressive 2,425,542 votes, a staggering 99% of titles in the sample received under 100,000 votes, indicating a vast disparity in audience engagement levels.

**Conclusion:**

The insights gleaned from this analysis offer a nuanced understanding of the evolving content production landscape and audience engagement metrics. These findings can serve as a compass for stakeholders in the industry, guiding strategic decisions and content directions.

## **6. Advanced Data Analysis using Aggregation**

* Demonstrated advanced analytical skills by extracting specific insights from the data:
  + Determined the total count of movies released annually.
  + Identified the top five Fantasy-Adventure movie titles (by primaryTitle) from 2021 based on their ratings. This involved filtering by genres that included both Fantasy and Adventure.
  + Extracted the top five Fantasy-Adventure movie titles from 2021, this time ranked by the number of votes they received, again ensuring the genre criteria were met.

Find the total number of movies released each year.

In order to determine the total number of movies released each year, we can use the $sortByCount function on the startYear field. This function performs the following steps:

1. Groups the movies by their start year, with each unique year value becoming a group.

2. Counts the number of movies in each year group.

3. Sorts the results by the calculated count for each group.

As a result, we obtain a sorted list of each year and the total number of movies released in that year, which answers our original question. The use of $sortByCount simplifies the process by handling the grouping, sorting, and counting in one stage. We do not need to specify the $sum accumulator or $sort order, as $sortByCount takes care of these automatically.

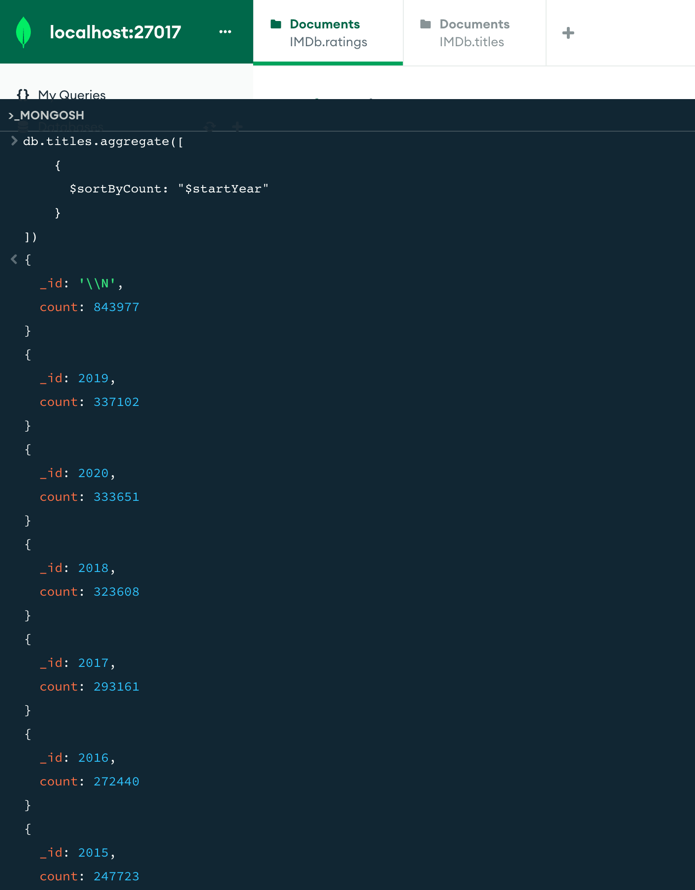
db.titles.aggregate([

{

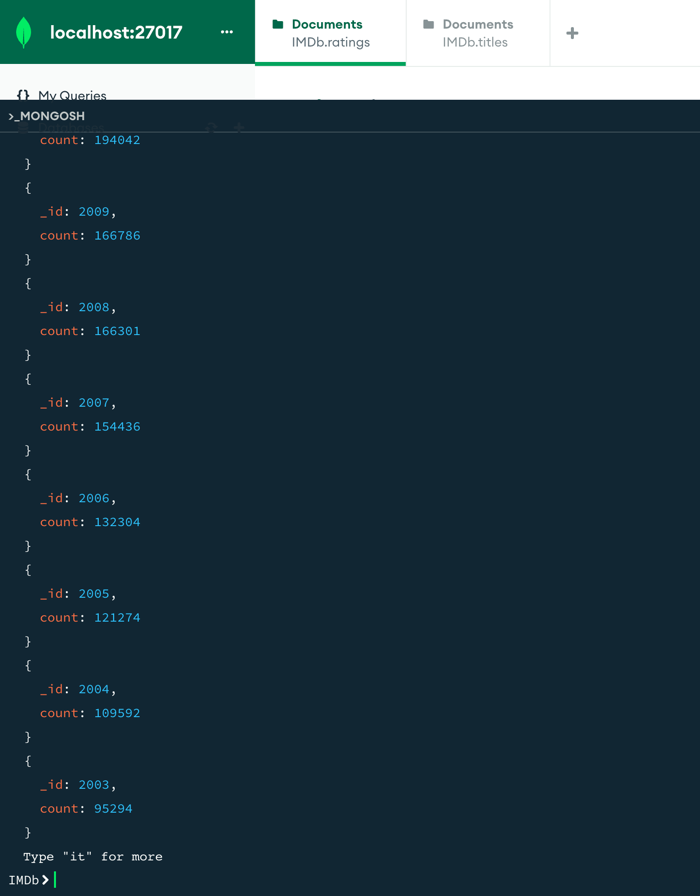
$sortByCount: "$startYear"

}

])

A screenshot of a computer program

Description automatically generated



**Find the top five Fantasy-Adventure movie titles (primaryTitle) released in 2021 according to the rating. [Hint: Genre must include both Fantasy and Adventure.]**

The provided MongoDB aggregation query first filters movies from the **titles** collection released in 2021 and belong to both "Fantasy" and "Adventure" genres using the **$match** stage and **$and** function. It then joins these movies with the **ratings** collection based on the **tconst** field using the **$lookup** stage, resulting in an array of ratings for each movie. The **$unwind** stage deconstructs this array, ensuring each rating is associated with a separate document. The movies are then sorted in descending order based on their average ratings with the **$sort** stage. Only the top 5 movies are retained using the **$limit** stage, and finally, the **$project** stage ensures that only the movie titles (**primaryTitle**) are returned in the output, excluding other fields like **\_id**.

db.titles.aggregate([

{

$match: {

"startYear": 2021,

"$and": [

{ "genres": { "$regex": "Fantasy", "$options": "i" } },

{ "genres": { "$regex": "Adventure", "$options": "i" } }

]

}

},

{

$lookup: {

from: "ratings",

localField: "tconst",

foreignField: "tconst",

as: "movieRating"

}

},

{

$unwind: "$movieRating"

},

{

$sort: {

"movieRating.averageRating": -1

}

},

{

$limit: 5

},

{

$project: {

primaryTitle: 1,

"\_id": 0

}

}

])

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated

The MongoDB aggregation query begins by filtering movies in the **titles** collection that were released in 2021 and fall under both the "Fantasy" and "Adventure" genres using the **$match** stage. It then associates these movies with their respective ratings from the **ratings** collection based on the **tconst** field through the **$lookup** stage, appending an array of ratings to each movie under the **movieRating** field. The **$unwind** stage breaks down this array, ensuring each movie is paired with a single rating in separate documents. Instead of sorting by average rating as in the previous query, this query sorts the movies in descending order based on the number of votes (**numVotes**) they received using the **$sort** stage. The top 5 movies, in terms of the number of votes, are then selected using the **$limit** stage. Finally, the **$project** stage ensures the output only displays the movie titles (**primaryTitle**), excluding other fields like **\_id**.

db.titles.aggregate([

{

$match: {

"startYear": 2021,

"$and": [

{ "genres": { "$regex": "Fantasy", "$options": "i" } },

{ "genres": { "$regex": "Adventure", "$options": "i" } }

]

}

},

{

$lookup: {

from: "ratings",

localField: "tconst",

foreignField: "tconst",

as: "movieRating"

}

},

{

$unwind: "$movieRating"

},

{

$sort: {

"movieRating.numVotes": -1

}

},

{

$limit: 5

},

{

$project: {

primaryTitle: 1,

"\_id": 0

}

}

])

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated