MONGODB ASSIGNMENT

PEER LEARNING DOCUMENT

Problem Statement -

1) Create a Python application to connect to MongoDB.
2) Bulk load the JSON files in the individual MongoDB collections using Python. MongoDB collections -
    a) comments
    b) movies
    c) theaters
    d) Users
3) Create Python methods and MongoDB queries to insert new comments, movies, theatres, and users into respective MongoDB collections.
4) Create Python methods and MongoDB queries to support the below operations -
    a) ***comments*** *collection*
        i) *Find top 10 users who made the maximum number of comments*
        ii) *Find top 10 movies with most comments*
        iii) *Given a year find the total number of comments created each month in that year*
    a. ***movies*** *collection*
        i. *Find top `N` movies -*
            1. *with the highest IMDB rating*
            2. *with the highest IMDB rating in a given year*
            3. *with highest IMDB rating with number of votes > 1000*
            4. *with title matching a given pattern sorted by highest tomatoes ratings*
        ii. *Find top `N` directors -*
            1. *who created the maximum number of movies*
            2. *who created the maximum number of movies in a given year*
            3. *who created the maximum number of movies for a given genre*
        iii. *Find top `N` actors -*
            1. *who starred in the maximum number of movies*
            5. *who starred in the maximum number of movies in a given year*
            3. *who starred in the maximum number of movies for a given genre*
        iv. *Find top `N` movies for each genre with the highest IMDB rating*
    a. ***theatre*** *collection*
        i. *Top 10 cities with the maximum number of theatres*
        ii. *top 10 theatres nearby given coordinates*

Rajkeshav Kumar Jha

Approach -

Comments Collection

1. He created an aggregate pipeline to group comments by user name, count the number of comments made by each user, sort the results by the count in ascending order, and finally cap the results at 10.
2. He created a pipeline for aggregation that would group comments by movie ID, count the number of comments for each film, sort the results by count in ascending order, and then cap the results at 10.
3. The year and month were projected from the date field, the documents matching the given year were matched, the comments were grouped by month, the number of comments were counted for each month, the month and count fields were projected, the _id field was excluded, and finally the results were sorted by month in ascending order.

Movies collection

1. **Question-1**
   1.1. He developed an aggregate pipeline to first match the documents where the IMDB rating is not null, sort the documents by IMDB rating in reverse chronological order, restrict the results to N documents, and project the _id, title, and imdb.rating columns.
   1.2. He developed an aggregation pipeline to first match the documents where the IMDB rating is not null and the year matches the supplied year. Then, the documents were sorted by IMDB rating in descending order, the results were limited to N documents, and the _id, title, imdb.rating, and year fields were projected.
   1.3. The movies are filtered and arranged according to the specified criteria using MongoDB's aggregation pipeline.
   There are four stages in the pipeline. Movies with a non-empty imdb.rating, the year 2002, and more than 1000 votes in imdb.votes are filtered out by the $match stage. The filtered movies are sorted by imdb.rating using the $sort stage. The output is limited by $limit stage to n documents. Just the _id, title, imdb.rating, and imdb.votes fields are included in the documents once $project stage has been applied.
   1.4. He filtered out films where the directors field is None is the first step in the pipeline for the aggregation query. Then, using the $sum operator, it counts the number of films for each director after grouping the films according to their directors field. With the $sort operator, the outcomes are arranged in descending order according to count. The $limit operator is then used to restrict the pipeline to just returning the top n results.

2. Question-2

2.1. He constructed a pipeline for aggregation The stages of the pipeline are as follows: Compare the documents whose years correspond to the specified years. To make a unique document for each director, unwind the array of directors. Sort the documents by director, then tally the number of films each director has produced. Sort the papers according to the number of movies they include. No more than n documents may be used. Project the number of documents and the year for each.

2.2. There are various stages in the pipeline. the genres array is unwound such that each document represents a different genre. After that, match the papers with the designated genre. Sort the documents by director, then count how many movies each has produced. Using the count field, arrange the results in descending order. Lastly, only the count field should be projected to be shown in the output.

2.3. He made an aggregation pipeline to unwind the "cast" data, which contains a list of actors, then group by actor name to determine how many films each actor has been in. returned the top n actors after sorting the results in order of decreasing count.

3. Question-3

3.1. Created an aggregation pipeline that uses

- The $match stage to filter the movies based on the year.
- The $unwind stage to create a separate document for each cast member of the movie.
- groups the documents by cast member
- The $sum operator to count the number of movies each actor has starred in.
- sorts the actors based on the count in descending order

3.2. He used an aggregation pipeline to first match the documents with the specified genre, unwind the cast array, group by the cast field to count how many films each actor has appeared in, sort the actors in decreasing order based on this information, and then restrict the results to n.

3.3. He used an aggregation pipeline to first match the documents with the specified genre, unwind the cast array, group by the cast field to count how many films each actor has appeared in, sort the actors in decreasing order based on this information, and then restrict the results to n.

4. Question-4

4.1. He made use of a pipeline that matches films in that category, ranks them according to IMDB ratings, filters out films without ratings, projects the movie title and rating fields, and only displays the top N matches. The results for each genre are then printed.

Therater collection

1. He used a pipeline to first group the theatre collection by city before using the $sum aggregation operator to count the number of theatres in each city. The cities are then arranged in descending order according to the number of theatres, and the top 10 cities are then excluded using the $limit aggregation operator. Project the city, count the fields, and print the outcome.
2. He developed a function called top10theatersNear that accepts a list of 2 float values for a location's longitude and latitude. The function creates a 2dsphere index on the location. The collection of theatres' geo field is used to search for the 10 theatres that are the closest to the specified coordinates. The function displays the outcome.