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. This function takes two arguments: n (the number of movies to display) and pattern (the regular expression pattern to match against movie titles). The $regex operator is used to perform the pattern matching, and the $options parameter is set to "i" to make the search case-insensitive.

2. **Question-2**

    2.1. 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.

    2.2. 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.

    2.3. 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.

    2.4. 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. 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.

    3.2. 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

    3.3. 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.

Shishir Singh

Approach -

Comments Collection

1. He performed an aggregate operation on the comments collection, grouping the comments according to the user's name and keeping track of the total number of comments using the sum function. I then sorted the results according to the commentsCount and used a limit of 10 to obtain the top 10 users who posted the most comments.
2. He sorted the results based on commentsCount and used a limit of 10 to obtain the aggregate result after performing an aggregate operation on the comments collection. The comments were grouped according to movie Id, and the comments count was kept using the sum function. built a list called lst that includes the movie name and associated comment count. Fill the list by iterating over this aggregate result. I used this list to create a pandas dataframe with the fields movie name and comment count.
3. He utilised the project operation to project the year and month from the given date using the month and date operator before performing the aggregate operation on the comments collection. The records for the given year were then matched, followed by grouping on the basis of month and maintaining a count of all comments created during that month. The results were then sorted, and the month and its corresponding comment count were projected.

Movies collection

1. **Question-1**
    1.1. He used my movie collection to solve this issue. Prior to sorting the records based on IMDB ratings in descending order and using a limit of 10 to obtain the top 10 results, we must first eliminate all those entries where IMDB ratings equals ""
    1.2. He first removed all instances where the IMDB rating equals "" and the year does coincide with the specified year. Then used limit 10 to retrieve the top 10 results after sorting the records based on imdb ratings in descending order.
    1.3. The top 10 results are obtained by first fetching records with more than 1000 votes, sorting the remaining data according to IMDB ratings in descending order, then using the limit 10 formula.
    1.4. we have to find top 10 movies with title matching a given pattern sorted by highest tomatoes ratings. Used movies collection for this problem. First filter out the records using the regex operator which is used to check the pattern & then sort this filtered data on the basis of tomatoes viewer rating in descending order and use limit 10 to get the top 10 results.

2. Question-2

2.1. He performed an aggregate operation on the movie collection after using the unwind operator to unwind the directors array. The unwind operator creates a document for each element by disassembling an array field from the input documents. Then, group the results based on the names of the directors and retain a corresponding total that reflects the number of movies that director has directed. Finally, sort the results based on movieCount and use a limit of 10 to obtain the top 10 filmmakers who have produced the most films.

2.2. He used the unwind operator to unwind the directors array after performing an aggregate operation on the movie collection to filter out any entries whose year matches the inputted year. The unwind operator creates a document for each element by disassembling an array field from the input documents. Then, group the results based on the names of the directors and maintain a corresponding sum that represents the number of movies that director has directed. Finally, sort the results based on movieCount and use a limit of 10 to obtain the top 10 directors who have produced the most films in any given year.

2.3. He performed aggregation on the movie collection was carried out, requiring the directors and genre arrays to be unwound and the records to be matched with the specified genre. Then use the sum function to do a grouping operation based on the director's name while also maintaining a matching movieCount. Sort the results using this movie as a basis To determine the top 10 directors, utilise a limit of 10 and count in ascending order.

3. Question-3

3.1. He performed aggregate operations on the movie collection after using the unwind operator to unwind the cast array. Group the data based on the cast, and maintain a related movieCount to show how many films that actor appeared in. Get the top 10 actors who appeared in the most movies by using a limit of 10 and sorting the results based on this movieCount in descending order.

3.2. He performed an aggregate operation on the movie collection, where the records whose years match the specified year must first be filtered. The cast array is then unwound using the unwind operator. Group the data based on the cast, and maintain a related movieCount to show how many films that actor appeared in. Get the top 10 actors who appeared in the most movies during a given year by sorting the results based on this movieCount in descending order and using a limit of 10.

3.3. He performed aggregate operation on the movie collection was completed. To begin, we had to match the genre specified with the genres of the movie documents. Once we had the filtered results, we unwound the cast array, performed grouping based on cast, and maintained a corresponding movieCount that represented the number of movies featuring that specific cast in the genre. To get the top 10 actors who appeared in the most movies in a specific genre, sort the results based on this movieCount in descending order and use a limit of 10.

4. **Question-4**
    4.1. He had first acquired all the genres. Set the genre for this aggregate action on the movie collection, and then execute grouping based on genre. Using a loop to go through the genres that were extracted in the first section, conduct aggregate operations on each genre, making sure to first match the genre and check that the imdb rating is not empty. After that, utilise restrict 4 to get the top 4 films in that genre by sorting the results based on IMDB ratings in descending order.

Therater collection

1. He performed an aggregate operation on the theatres collection where we need to group the entries based on location.address.city and also maintain a matching theatres count indicating the number of theatres in that city. Sort the outcomes based on theatres To determine the top 10 cities with the most theatres, count in descending order and use a limit of 10.
2. In this case, he had established an index on the location field. And performed an aggregate operation on the collection of theatres. Using geonear operator that produces documents in the order of proximity to a defined place. Geonear requires us to provide the location type and coordinates. In the end, limit 10 was utilised to get the top 10 closest theatres from the provided locations.