**Mongodb Peer Learning**

**Rajkeshav Kumar Jha**

**TAS129**

**Shishir’s Approach**

**Comment collections**

Q.1 Performed aggregate operation on comments collection and grouped the comments on the basis of the user's name and also maintained the comments count using sum function and then sorted the result on the basis of commentsCount and used limit 10 to get the top 10 users who made max number of comments.

Q.2 Performed aggregate operation on comments collection. Grouped the comments on the basis of the movie\_Id and also maintained the comments count using sum function and then sort the result on the basis of commentsCount and use limit 10 to get the aggregate\_result. Created a list lst which contains the movie\_name and corresponding comment count. Iterate over this aggregate\_result and populate the list. Created a pandas dataframe using this list containing 2 columns movie\_name and comment\_count

Q.3 Performed aggregate operation on comments collection, first used project operation to project the year and month from the given date using month and date operator.Then performed a match operation to match the records for the given year and then performed the grouping on the basis of month and maintained a count which indicates the total number of comments created for that month, sort the result and project the month and its corresponding comment count.

**Movies collection**

Q.4 In this problem, we have to find the top 10 movies with highest IMDB ratings. Used movies collection for this problem. First we have to discard all those records where IMDB ratings are equal to "" and then sort the records on the basis of imdb ratings in descending order and then use limit 10 tho get top 10 results.

Q.5 In this problem, we have to find the top 10 movies with highest IMDB ratings in a given year. Used movies collection for this problem. First we have to discard all those records where IMDB ratings are equal to "" and year doesn't match with the given year. Then sort the records on the basis of imdb ratings in descending order and then use limit 10 to get top 10 results.

Q.6 In this problem, we have to find the top 10 movies with the highest IMDB rating with the number of votes > 1000. Used movies collection for this problem. First filter out the records where number of votes > 1000 & then sort this filtered data on the basis of IMDB ratings in descending order and use limit 10 to get the top 10 results.

Q.7 In this problem, we have to find top 10 movies with titles matching a given pattern sorted by highest tomato 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 tomato viewer rating in descending order and use limit 10 to get the top 10 results.

Q.8 In this problem, we have to find the top 10 directors who created the maximum number of movies. Performed aggregate operation on the movies collection where first we need to unwind the directors array using unwind operator. Unwind operator deconstructs an array field from the input documents to output a document for each element. Then perform grouping on the basis of directors name and also maintain a corresponding sum which represents the movie\_count for that specific director, sort the result on the basis of movieCount and use limit 10 to get the top 10 directors who created the maximum number of movies.

Q.9 In this problem, we have to find the top 10 directors who created the maximum number of movies in a given year. Performed aggregate operation on the movies collection where first we need to filter those records whose year matches with the given year & then unwind the directors array using unwind operator. Unwind operator deconstructs an array field from the input documents to output a document for each element. Then perform grouping on the basis of directors name and also maintain a corresponding sum which represents the movie\_count for that specific director, sort the result on the basis of movieCount and use limit 10 to get the top 10 directors who created the maximum number of movies in a given year.

Q.10 In this problem, we have to find the top 10 directors who created the maximum number of movies for a given genre. Performed aggregate operation on the movies collection where we need to unwind the directors and genre array and then match the records with the given genre. Then perform grouping operations on the basis of director name and also maintain a corresponding movieCount using the sum function. Sort the result on basis of this movieCount in descending order and use limit 10 to get the top 10 directors

Q.11 In this problem, we have to find top 10 actors who starred in the maximum number of movies.Performed aggregate operation on the movies collection where first we need to unwind the cast array using unwind operator. Then group the records on basis of cast and also maintained corresponding movieCount which indicates number of movies in which that actor starred. Sort the result on basis of this movieCount in descending order and use limit 10 to get the top 10 actors who starred in max movies.

Q.12 In this problem, we have to find top 10 actors who starred in the maximum number of movies in a given year. Performed aggregate operation on the movies collection where first we need to filter the records whose year matches with the given year. Then unwind the cast array using unwind operator. Then group the records on the basis of cast and also maintain corresponding movieCount which indicates the number of movies in which that actor starred. Sort the result on basis of this movieCount in descending order and use limit 10 to get the top 10 actors who starred in max movies in given year.

Q.13 In this problem, we have to find the top 10 actors who starred in the maximum number of movies in a given genre. Performed aggregate operation on the movies collection where first we need to match the given genre with the genres of the movie documents & then once we have the filtered result, unwind the cast array, perform grouping on basis of cast and maintain corresponding movieCount which indicates the movie count for that particular cast in the given genre. Sort the result on the basis of this movieCount in descending order and use limit 10 to get the top 10 actors who starred in max movies in a given genre.

Q.14 In this problem, we have to find the top 10 movies for each genre with the highest IMDB rating. First we need to get all the genres. For this performed aggregate operation on movies collection, unwind the genre and then perform grouping on the basis of genre. Loop over the genres extracted in the first part and for each genre perform aggregate operation where we first match the genre and also ensure that imdb rating is not empty.Then sort the result on basis of IMDB ratings in descending order and use limit 4 to get the top 4 movies in that particular genre.

**Theater collection**

Q.15 In this problem, we have to find the Top 10 cities with the maximum number of theaters. Performed aggregate operation on the theaters collection where we need to group the records on the basis of location.address.city and also need to maintain corresponding theaters count indicating number of theaters in that city. Sort the result on basis of theatresCount in descending order & use limit 10 to get the top 10 cities with max number of theaters

Q.16 In this problem, we have to find the top 10 theaters nearby given coordinates. Here we need to create an index on the location field. Geonear operator requires indexing on our geolocation object. Performed aggregate operation on the theaters collection. Used geonear operator which outputs documents in order of nearest to farthest from a specified point. In geonear we have to specify the coordinates and the location type. Finally used limit 10 to get the top 10 nearest theaters from the given coordinates

**Chakradhar’s approach**

**Comment collection**

Q1) This Python function defines a MongoDB aggregation pipeline to find the top 10 users with the most comments. The pipeline consists of the following stages:

* $group stage groups documents by the value of the name field and counts the number of documents in each group using the $sum operator. The \_id field is set to the value of the name field so that the results can be grouped by username.
* $sort stage sorts the grouped results in descending order based on the count.
* $limit stage limits the number of documents in the output to 10, i.e., only the top 10 users with the most comments are included in the output.
* $project stage projects only the User and count fields in the output, and excludes the \_id field.

Q2) The function defines a MongoDB aggregation pipeline to find the top 10 movies with the most comments based on the movie\_id field in the input documents. The function then iterates through the results of the aggregation pipeline and prints the title of each movie, which is retrieved from a separate collection called moviesCollection, along with the count of comments for that movie.

Q3) The function defines a MongoDB aggregation pipeline to count the number of comments made in each month of a specific year. The function takes a year parameter and returns a list of dictionaries, each containing the month and count fields.

The pipeline consists of the following stages:

* $project: This stage extracts the year and month fields from the date field in the input documents using the $year and $month operators.
* $match: This stage filters the documents based on the year field, selecting only documents that match the specified year.
* $group: This stage groups the documents by month, and calculates the count of documents in each group using the $sum operator.
* $project: This stage selects the month and count fields from the grouped documents, and excludes the \_id field. The output documents contain only the month and count fields.
* $sort: This stage sorts the output documents in ascending order based on the month field.

**Movies collection**

Q4) The function defines a MongoDB aggregation pipeline to find the top N movies with the highest IMDB rating. The function takes the value of N.

The pipeline consists of the following stages:

* $match: This stage filters the documents by selecting only the documents where the imdb.rating field is not empty.
* $sort: This stage sorts the documents in descending order based on the imdb.rating field.
* $limit: This stage limits the output to only the top N documents based on the value entered by the user.
* $project: This stage selects the title and imdb.rating fields from the output documents, and excludes the \_id field.

Q5) The function defines a MongoDB aggregation pipeline to find the top N movies with the highest IMDB rating in a specific year. The function takes a year parameter and the value of N.

The pipeline consists of the following stages:

* $match: This stage filters the documents by selecting only the documents where the imdb.rating field is not empty and the year field matches the specified year.
* $sort: This stage sorts the documents in descending order based on the imdb.rating field.
* $limit: This stage limits the output to only the top N documents based on the value entered by the user.
* $project: This stage selects the title, imdb.rating, and year fields from the output documents, and excludes the \_id field.

### 

Q6) The function defines a MongoDB aggregation pipeline to find the top N movies with the highest IMDB rating and at least 1000 votes.

The pipeline consists of the following stages:

* $match: This stage filters the documents by selecting only the documents where the imdb.rating field is not empty and the imdb.votes field is greater than or equal to 1000.
* $sort: This stage sorts the documents in descending order based on the imdb.rating field.
* $limit: This stage limits the output to only the top N documents based on the value entered by the user.
* $project: This stage selects the title, imdb.rating, and imdb.votes fields from the output documents, and excludes the \_id field.

### 

Q7) The function defines a MongoDB aggregation pipeline to find the top N movies whose titles match a given regular expression pattern, sorted by their highest tomato viewer rating.

The pipeline consists of the following stages:

* $match: This stage filters the documents by selecting only the documents where the title field matches the regular expression pattern entered by the user.
* $sort: This stage sorts the documents in descending order based on the tomatoes.viewer.rating field.
* $limit: This stage limits the output to only the top N documents based on the value entered by the user.
* $project: This stage selects the title and tomatoes.viewer.rating fields from the output documents, and excludes the \_id field.

### 

Q8) The function defines a MongoDB aggregation pipeline to find the top N directors with the maximum number of movies in a given collection.

The pipeline consists of the following stages:

* $unwind: This stage creates a new document for each element in the directors array field of the input documents, effectively denormalizing the data.
* $group: This stage groups the documents by the directors field and calculates the total number of movies directed by each director using the $sum aggregation function.
* $sort: This stage sorts the output documents in descending order based on the noOfMovies field.
* $limit: This stage limits the output to only the top N documents based on the value entered by the user.

### 

Q9) The function builds a MongoDB aggregation pipeline to find the top N directors with the maximum number of movies in a given year.

The pipeline includes the following stages:

* $match: Filters the documents by the given year.
* $unwind: Deconstructs the directors array to create a separate document for each element in the array.
* $group: Groups the documents by the directors field and calculates the count of movies for each director using the $sum operator.
* $sort: Sorts the documents by the noOfMovies field in descending order.
* $limit: Limits the number of documents to the specified value of N.

### Q10) The function takes a genre as input and finds the top N directors with the maximum number of movies in the given genre. The pipeline for this function includes the following stages:

* $match: This stage filters the movies based on the input genre.
* $unwind: This stage splits the directors array into separate documents for each director.
* $group: This stage groups the movies based on the directors and calculates the count of movies for each director.
* $sort: This stage sorts the directors based on the number of movies they have directed in descending order.
* $limit: This stage limits the output to the top N directors based on the input value of N.

### 

Q11) This function will find the top N actors with the maximum number of movies they have acted in. The pipeline does the following:

* $unwind the cast array to get individual actors as separate documents.
* $group by the actor's name and count the number of movies they have acted in using $sum.
* $sort the documents by the count of movies in descending order.
* $limit the documents to N. The value of N is taken as input from the user.

### 

Q12) This function uses an aggregation pipeline to find the top N actors with the maximum number of movies in a given year. The pipeline first matches the documents with the given year, then it unwinds the cast array to create a document for each actor in the cast. Then it groups the documents by actor and calculates the count of movies for each actor. It then sorts the result in descending order of movie counts and limits the output to the top N actors. Finally, it returns the result as a cursor object. The value of N is taken as input from the user.

* $match: This stage filters the documents based on a given condition. In this case, it filters the documents based on the year of the movie.
* $unwind: This stage creates a new document for each element in an array. In this case, it creates a new document for each actor in the cast array.
* $group: This stage groups the documents based on a given key and applies an accumulator operation to each group. In this case, it groups the documents by the cast field and calculates the number of movies for each actor using the $sum accumulator.
* $sort: This stage sorts the documents based on a given field. In this case, it sorts the documents in descending order of the noOfMovies field.
* $limit: This stage limits the number of documents in the output to a specified number.

### 

Q13) The function aims to find the top N actors who have acted in the maximum number of movies in a given genre. It takes a genre as an input parameter and uses the aggregate() function to perform aggregation operations on the movies collection. The pipeline consists of the following stages:

* $match: This stage filters the movies collection to include only those movies that have the given genre.
* $unwind: This stage deconstructs the cast array field into individual documents, creating a new document for each actor who acted in the movie.
* $group: This stage groups the documents by the cast field and calculates the count of movies in which each actor has acted.
* $sort: This stage sorts the grouped documents in descending order based on the count of movies in which each actor has acted.
* $limit: This stage limits the output to the top N actors based on the count of movies in which each actor has acted. collection named "moviesCollection". The pipeline of the function works as follows:

Q14)

* The first stage of the pipeline is "$unwind" stage. It creates a copy of each document for every element in the "genres" array, allowing for grouping by genre later.
* The second stage groups the documents by "genres" field and creates a new document with the "\_id" field representing the genre.
* The function then loops through the list of genres and creates a pipeline for each genre.
* The pipeline for each genre starts with an "$unwind" stage that creates a copy of each document for every element in the "genres" array.
* The next stage "$match" filters the documents where the "genres" field matches the current genre.
* The next stage "$sort" sorts the matching documents by descending "imdb.rating" values.
* The next stage "$match" filters out documents with empty "imdb.rating" fields.
* The next stage "$project" outputs the "title" and "imdb.rating" fields.

Finally, the "$limit" stage limits the output to the top N movies, where N is taken as input from the user.

### **Theatre collection**

Q15) The top10CitiesMostTheaters function retrieves the top 10 cities with the most number of theaters. It does this by using the aggregate function to apply a pipeline of aggregation stages to the theatersCollection.

* The first stage is $group which groups the documents by the city in the location.address field and applies a $sum accumulator to count the number of documents per city. The result of this stage is an intermediate collection where each document has a \_id field corresponding to the city name and a cnt field with the count of documents.
* The second stage is $sort which sorts the documents in descending order based on the cnt field.
* The third and last stage is $limit which limits the number of documents in the output to 10.

### 

Q16) The function top10CitiesMostTheaters() retrieves the top 10 cities with the most number of theaters. It does so by using the aggregate() method to perform aggregation on the theaters collection.

The aggregation pipeline consists of the following stages:

* $group: This stage groups documents by the location.address.city field and calculates the count of documents in each group using the $sum operator. The result of this stage is a document for each unique city with the count of theaters in that city.
* $sort: This stage sorts the documents in descending order based on the cnt field, which is the count of theaters in each city.
* $limit: This stage limits the output to the first 10 documents, which are the cities with the highest count of theaters.