CSCU9YQ

NoSQL Databases

MongoDB Report
Student Number-2626628

INDEX

Queries:

Q1. General Queries

1. Compute the average duration of movies by genre, and list the top 5 genres with the longest average duration. List the name of the genre and the average duration time sorted by time in decreasing order.

Explanation:
Strategy: I approached to get the desired results by aggregating by genre and runtime together and sorting the results by average runtime and finally limiting the total results to top5.
The aggregate query calculates aggregate values for the data in a collection. Aggregate has multiple filters. Here I have used the following filters.
group ==> group is used to club all the similar genre movies to calculate the average runtime
sort ==> for sorting the final results ascending or descending based on the filed mentioned. 1- for ascending and -1 for descending.
limit ==> restricts to limited number of results in case if we don't need the entire list of data matching the search criteria.

Query:
```
db.movies.aggregate([{"$group":{_id: "$genres", avg_runtime:{$avg:"$runtime"}}}, {"$sort":
{"avg_runtime": -1}}, {"$limit": 5}])
```

Result:

```
{ "_id" : [ "Adventure", "Drama", "Musical" ], "avg_runtime" : 224 }

{ "_id" : [ "Musical", "Drama", "Romance" ], "avg_runtime" : 197 }

{ "_id" : [ "Action", "Drama", "Musical" ], "avg_runtime" : 190 }
{ "_id" : [ "Musical", "Romance", "Drama" ], "avg_runtime" : 177 }

{ "_id" : [ "Action", "Fantasy", "Musical" ], "avg_runtime" : 170 }
```

2. List all the countries that have produced 10 or more movies with the UK. List the country names and number of movies produced in collaboration with the UK, in no particular order

Explanation:

Strategy: I have matched all the countries that have produced 10 or more movies with UK. The aggregate query calculates aggregate values for the data in a collection. Aggregate has multiple filters. Here I have used the following filters.

match ==>for filtering the data matching the condition mentioned

sort ==> for sorting the final results ascending or descending based on the filed mentioned. 1- for ascending and -1 for descending.

group ==> group is used to club all the similar genre movies to calculate the average runtime

Query:

```
db.movies.aggregate([ {"$match": {"$and":[{countries: "UK"}]}}, {"$group":{_id: "$countries", count: {$sum: 10}}},
{"$sort": {"count": -1}}])
```

Result:

```
{ "_id" : [ "UK" ], "count" : 1080 }

{ "_id" : [ "UK", "USA" ], "count" : 200 }

{ "_id" : [ "USA", "UK" ], "count" : 170 }

{ "_id" : [ "Germany", "UK" ], "count" : 30 }
{ "_id" : [ "UK", "Italy" ], "count" : 20 }

{ "_id" : [ "USA", "UK", "Germany" ], "count" : 20 }

{ "_id" : [ "USA", "Canada", "UK" ], "count" : 20 }

{ "_id" : [ "UK", "France" ], "count" : 10 }
{ "_id" : [ "USA", "Taiwan", "UK" ], "count" : 10 }

{ "_id" : [ "UK", "USA", "Canada" ], "count" : 10 }

{ "_id" : [ "USA", "UK", "Canada", "Japan" ], "count" : 10 }

{ "_id" : [ "Denmark", "Sweden", "Norway", "UK", "USA", "Germany", "Netherlands", "Israel", "Spain", "Belgium",
"Canada" ], "count" : 10 }

{ "_id" : [ "Denmark", "China", "UK" ], "count" : 10 }

{ "_id" : [ "Portugal", "UK", "Luxembourg" ], "count" : 10 }

{ "_id" : [ "France", "Italy", "UK" ], "count" : 10 }

{ "_id" : [ "UK", "USA", "France", "Sweden" ], "count" : 10 }

{ "_id" : [ "Spain", "UK", "USA" ], "count" : 10 }

{ "_id" : [ "Ireland", "UK", "USA" ], "count" : 10 }

{ "_id" : [ "UK", "Slovenia", "Serbia", "Germany", "Bosnia and Herzegovina" ], "count" : 10 }

{ "_id" : [ "UK", "Sierra Leone" ], "count" : 10 }
```

```
Type "it" for more
```

Q.2 Movies Related to Sports

1. Report the total number of movies you can find in the collection which are related to any type of sport(s). Report only the number of movies, not their details.

Explanation:
Strategy: Finding the movies with genre that matches sport, which gives total list of all movies.

The find query is just to search for the required criteria. This lists all the data in the collection matching the searched criteria. As we need only the count, I used the count method to return the total number for data matching the criteria.

Query:
db.movies.find({"genres": "Sport"}).count()

Result:

```
28
```

2. From the sport-related movies you found in part 1, list the top 3 that have received the highest IMDb 'rating'. List the movie title, the year, and both the IMDb 'rating' and the votes', sorted by the 'rating'.

Explanation:
Strategy: For this, I have first found the total sports movies and sorted by IMDB rating, limiting to the top 3 results and projecting the desired fields by IMDB rating and votes.

The aggregate query calculates aggregate values for the data in a collection. Aggregate has multiple filters. Here I have used the following filters.

match ==>for filtering the data matching the condition mentioned sort ==> for sorting the final results ascending or descending based on the filed mentioned. 1- for ascending and -1 for descending.

limit ==> restricts to limited number of results in case if we don't need the entire list of data matching the search criteria.

project ==> returns the documents with the requested fields. 1- required field, 0- non-required field.

Query:

db.movies.aggregate([{"$match": {"genres": "Sport"}},{"$sort": {"imdb.rating": -1}}, {"$limit": 3}, {"$project": {"title": 1, "_id": 0, "year": 1, "imdb.rating": 1, "imdb.votes": 1}}])

Result:

```
{ "title" : "Liverpool FC: Champions of Europe 2005", "year" : 2005, "imdb" : { "rating" : 9.5, "votes" : 410 } }

{ "title" : "WWE: DX: One Last Stand", "year" : 2011, "imdb" : { "rating" : 7.7, "votes" : 51 } }

{ "title" : "O sport, ty - mir!", "year" : 1981, "imdb" : { "rating" : 7.5, "votes" : 23 } }
```

Q3. Rating and Recommending Movies

1. Create your own field named "myRating" and compute a movie score according to your design. The idea is to combine and aggregate the information about the quality of movies existing in the collection in a single number which will be your rating.

<u>**Explanation:**</u>

Strategy: I have created my own field called my_rating and I have created my rating based on award nominations, wins, IMDB rating and votes.

The aggregate query calculates aggregate values for the data in a collection. Aggregate has multiple filters. Here I have used the following filters.

sort ==> for sorting the final results ascending or descending based on the filed mentioned. 1- for ascending and -1 for descending.

project ==> returns the documents with the requested fields. 1- required field, 0- non-required field. Here while projecting we calculate the average using $avg operator. (Here my criteria are the average of award won and nominated)

Query:

```
db.movies.aggregate([{"$project":{my_rating:{$avg:["$awards.wins", "$awards.nominations", "imdb.rating",
"imdb.votes"]},"_id":0, "title": 1}}, {"$sort": {"my_rating": -1}}])
```

Result:

```
{ "title" : "No Country for Old Men", "my_rating" : 135 }


{ "title" : "There Will Be Blood", "my_rating" : 113 }
{ "title" : "Gone Girl", "my_rating" : 107 }


{ "title" : "Beasts of the Southern Wild", "my_rating" : 105 }


{ "title" : "The Tree of Life", "my_rating" : 102 }


{ "title" : "Lost in Translation", "my_rating" : 95 }


{ "title" : "Life of Pi", "my_rating" : 95 }


{ "title" : "Little Miss Sunshine", "my_rating" : 80.5 }


{ "title" : "L.A. Confidential", "my_rating" : 79 }


{ "title" : "Good Night, and Good Luck.", "my_rating" : 73.5 }
{ "title" : "Toy Story 3", "my_rating" : 71 }

{ "title" : "Toy Story 3", "my_rating" : 71 }

{ "title" : "Shakespeare in Love", "my_rating" : 70 }

{ "title" : "An Education", "my_rating" : 60 }

{ "title" : "Goliyon Ki Rasleela Ram-Leela", "my_rating" : 59.5 }

{ "title" : "Into the Wild", "my_rating" : 56.5 }

{ "title" : "The Girl with the Dragon Tattoo", "my_rating" : 52.5 }

{ "title" : "Life Is Beautiful", "my_rating" : 51 }

{ "title" : "The Truman Show", "my_rating" : 50 }

{ "title" : "Star Trek", "my_rating" : 49.5 }

Type "it" for more
```

2. Using your newly created rating to rank movies, your task is to recommend 3 good movies according to the type of movies you enjoy watching. Your preferences could be based on any movie characteristics such as genres, actors, topics, directors, years etc.

<u>Explanation:</u>

Strategy: For this, I have matched the genre with sport and used my own field my_rating by averaging awards of wins, nominations and IMDB rating and votes using id and title. By liming with 3.

The aggregate query calculates aggregate values for the data in a collection. Aggregate has multiple filters. Here I have used the following filters.

sort ==> for sorting the final results ascending or descending based on the filed mentioned. 1- for ascending and -1 for descending.

project ==> returns the documents with the requested fields. 1- required field, 0- non-required field. Here while projecting we calculate the average using $avg operator. (Here my criteria are the average of award won and nominated).

match ==>for filtering the data matching the condition mentioned. In the same criteria above, I've added my preference as genre = Sports

Query:

```
db.movies.aggregate([{ $match: {"genres": "Sport" }}, {"$project":{my_rating:{$avg:["$awards.wins",
"$awards.nominations", "imdb.rating", "imdb.votes"]},"_id":0, "title": 1}}, {"$sort": {"my_rating": -1}}, {"$limit": 3}])
```

Result:

```
{ "title" : "Dodgeball: A True Underdog Story", "my_rating" : 3.5 }

{ "title" : "Ah Fu", "my_rating" : 2 }

{ "title" : "Ed", "my_rating" : 1.5 }
```