Assignment 1: Storage and processing of data

MongoDB (help link: http://www.mongodb.org/display/DOCS/Home)

We have provided a data set which consists of 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users.

Design data structures to represent the data (you might have one document that stores all information aggregated under user object or one document that stores information about aggregate under movie object, or split into two or tree aggregations).

Your models should be able to answer the following queries below.

Create a data store. Extract, transform and load the provided data set into your data storage.

Answer the following questions.

Questions:

1)What genre is the movie CopyCat in?

-

Query:

```
db.movies.find({movieName: /.*Copycat*./},{genres:1, _id:0})
```

Result:

```
{ "genres" : [ "Crime", "Drama", "Horror", "Mystery", "Thriller" ] }
```

2)what genre has the most movies?

-

Query:

```
db.movies.aggregate([{$unwind:"$genres"},{$group:{"_id":"$genres",count:{$sum:1}}},{$sort:{count:-1}},{$limit:1}])
```

Result:

```
{ "_id" : "Drama", "count" : 5339 }
```

3)what tags did user 146 use to describe the movie "2001: A Space Odyssey”

-

Query:

```
var movie_2001_id = db.movies.findOne({movieName:/.*2001: A Space*./},{movieId:1}).movieId
db.tags.find({userId:"146",movieId: movie_2001_id},{_id: 0,tag: 1})
```

Result:

{ "tag" : "Arthur C. Clarke" }

{ "tag" : "artificial intelligence" }

{ "tag" : "based on a book" }

4)What are the top 5 movies with the highest avg rating?

-

QUERY:

```
#Parse ratings data to INTEGER (This takes a lot of time, a way to bulk parse would be more efficient)
db.ratings.find().forEach(function(data) { db.ratings.update({ "_id": data._id, "movieId": data.movieId }, { "$set": { "rating": parseInt(data.rating) } }); })

#Aggregate according to movieId, average out the sum of ratings and output the top five results to topMovies
db.ratings.aggregate([{$group:{_id:"$movieId",count:{$sum:1},totalRatings: {$sum: "$rating"},avgRating:{"$avg":{$sum:"$rating"}}}},{$sort:{avgRating:-1}},{$limit:5},{$out: "topMovies"}],{allowDiskUse: true})

#declare an array variable to store the 5 movieIds
var movIds = new Array()

#use push method to append all movieIds from topMovies into movIds array
db.topMovies.find({},{_id:1}).forEach(function(myDoc){movIds.push(myDoc._id)})

#find the corresponding movie names
db.movies.find({movieId:{$in:movIds}},{_id:0,movieName:1})
```

RESULT:

{ "movieName" : "Satan's Tango (Sátántangó) (1994)" }

{ "movieName" : "Shadows of Forgotten Ancestors (1964)" }

{ "movieName" : "Fighting Elegy (Kenka erejii) (1966)" }

{ "movieName" : "Sun Alley (Sonnenallee) (1999)" }

{ "movieName" : "Blue Light, The (Das Blaue Licht) (1932)" }

5)What is the highest avg rating possible?

-

QUERY:

```
db.topMovies.find({},{_id:0,avgRating:1}).sort({avgRating:-1}).limit(1)
```

RESULT:

{ "avgRating" : 5 }

6) Write 3 different queries of your choice to demonstrate that your data storage is working.

i)- Count of movies with average rating between 2.0 and 3.5

QUERY:

```
db.ratings.aggregate([{$group:{_id:"$movieId",count:{$sum:1},totalRatings: {$sum: "$rating"},avgRating:{"$avg":{$sum:"$rating"}}}},{$out:"movies_aggr"}],{allowDiskUse: true})
```

db.movies_aggr.count({avgRating:{$gte: 2.0, $lt: 3.5}})

RESULT:

7743

ii)- Find top 5 most popular tags

QUERY:

db.tags.aggregate([{$group:{_id:"$tag",count:{$sum:1}}},{$sort:{count:-1}},{$limit:5}])

RESULT:

{ "_id" : "Tumey's DVDs", "count" : 641 }

{ "_id" : "classic", "count" : 621 }

{ "_id" : "based on a book", "count" : 549 }

{ "_id" : "R", "count" : 518 }

{ "_id" : "less than 300 ratings", "count" : 505 }

iii)- Count number of unique user ids based on above tags

QUERY:

db.tags.aggregate([{$group:{_id:"$tag",count:{$sum:1}}},{$sort:{count:-1}},{$limit:5},{$out:"topTags"}])

var topTagIds = new Array()

db.topTags.find({},{_id:1}).forEach(function(doc){topTagIds.push(doc._id)})

db.tags.distinct("userId",{"tag":{$in:topTagIds}}).length

RESULT:

319