INFSCI 2725 DATA ANALYTICS

ASSIGNMENT 1

Group members:

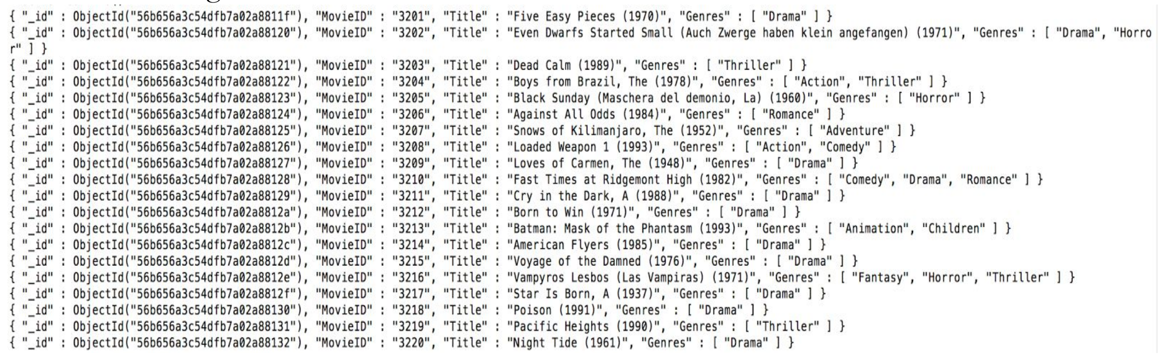
- Yumeng Lu
- Zhaoxuan Ren
- Tao Li

Programming Language: JAVA

Preparation

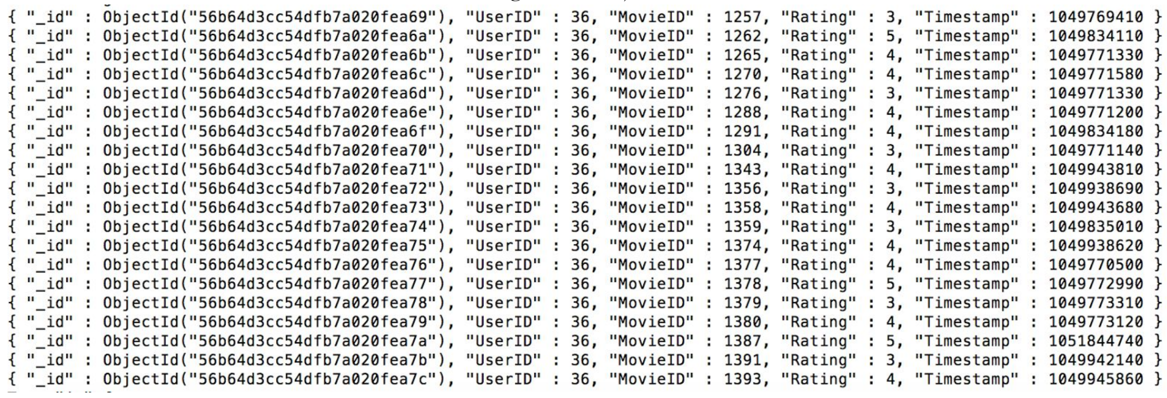
Step1: Transfer the format of the given file.

The three given files are 'dat' format which MongoDB can not read, so the first step we should do is to transfer the 'dat' format files to 'json' format which MongoDB can read.
In order to reach this goal, we used Java IO stream to complete the process.
After transferring:



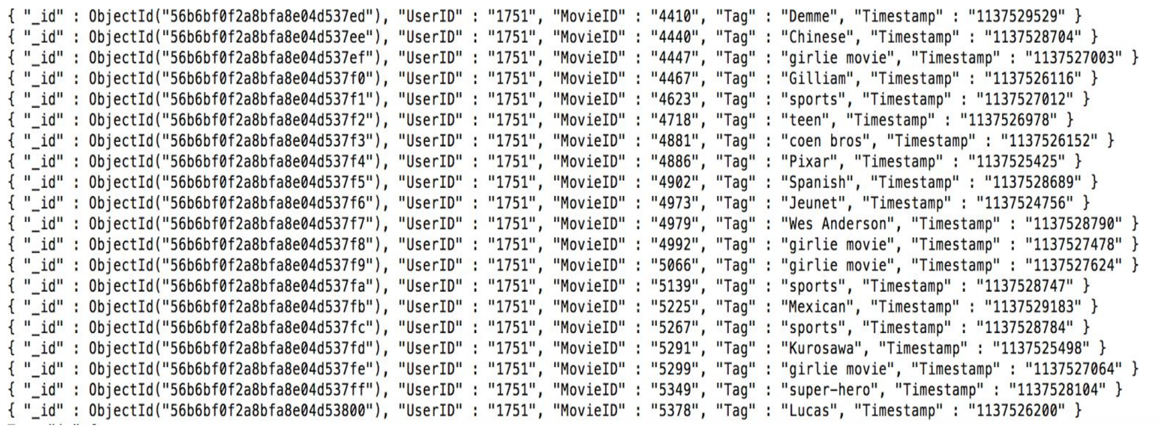
```
{ "_id" : ObjectId("56b656a3c54dfb7a02a8811f"), "MovieID" : "3201", "Title" : "Five Easy Pieces (1970)", "Genres" : [ "Drama" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a88120"), "MovieID" : "3202", "Title" : "Even Dwarfs Started Small (Auch Zwerge haben klein angefangen) (1971)", "Genres" : [ "Drama", "Horro
r" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a88121"), "MovieID" : "3203", "Title" : "Dead Calm (1989)", "Genres" : [ "Thriller" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a88122"), "MovieID" : "3204", "Title" : "Boys from Brazil, The (1978)", "Genres" : [ "Action", "Thriller" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a88123"), "MovieID" : "3205", "Title" : "Black Sunday (Maschera del demonio, La) (1960)", "Genres" : [ "Horror" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a88124"), "MovieID" : "3206", "Title" : "Against All Odds (1984)", "Genres" : [ "Romance" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a88125"), "MovieID" : "3207", "Title" : "Snows of Kilimanjaro, The (1952)", "Genres" : [ "Adventure" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a88126"), "MovieID" : "3208", "Title" : "Loaded Weapon 1 (1993)", "Genres" : [ "Action", "Comedy" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a88127"), "MovieID" : "3209", "Title" : "Loves of Carmen, The (1948)", "Genres" : [ "Drama" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a88128"), "MovieID" : "3210", "Title" : "Fast Times at Ridgemont High (1982)", "Genres" : [ "Comedy", "Drama", "Romance" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a88129"), "MovieID" : "3211", "Title" : "Cry in the Dark, A (1988)", "Genres" : [ "Drama" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a8812a"), "MovieID" : "3212", "Title" : "Born to Win (1971)", "Genres" : [ "Drama" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a8812b"), "MovieID" : "3213", "Title" : "Batman: Mask of the Phantasm (1993)", "Genres" : [ "Animation", "Children" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a8812c"), "MovieID" : "3214", "Title" : "American Flyers (1985)", "Genres" : [ "Drama" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a8812d"), "MovieID" : "3215", "Title" : "Voyage of the Damned (1976)", "Genres" : [ "Drama" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a8812e"), "MovieID" : "3216", "Title" : "Vampyros Lesbos (Las Vampiras) (1971)", "Genres" : [ "Fantasy", "Horror", "Thriller" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a8812f"), "MovieID" : "3217", "Title" : "Star Is Born, A (1937)", "Genres" : [ "Drama" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a88130"), "MovieID" : "3218", "Title" : "Poison (1991)", "Genres" : [ "Drama" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a88131"), "MovieID" : "3219", "Title" : "Pacific Heights (1990)", "Genres" : [ "Thriller" ] }
{ "_id" : ObjectId("56b656a3c54dfb7a02a88132"), "MovieID" : "3220", "Title" : "Night Tide (1961)", "Genres" : [ "Drama" ] }
```

Fig.1 movies.json



```
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea69"), "UserID" : 36, "MovieID" : 1257, "Rating" : 3, "Timestamp" : 1049769410 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea6a"), "UserID" : 36, "MovieID" : 1262, "Rating" : 5, "Timestamp" : 1049834110 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea6b"), "UserID" : 36, "MovieID" : 1265, "Rating" : 4, "Timestamp" : 1049771330 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea6c"), "UserID" : 36, "MovieID" : 1270, "Rating" : 4, "Timestamp" : 1049771580 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea6d"), "UserID" : 36, "MovieID" : 1276, "Rating" : 3, "Timestamp" : 1049771330 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea6e"), "UserID" : 36, "MovieID" : 1288, "Rating" : 4, "Timestamp" : 1049771200 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea6f"), "UserID" : 36, "MovieID" : 1291, "Rating" : 4, "Timestamp" : 1049834180 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea70"), "UserID" : 36, "MovieID" : 1304, "Rating" : 3, "Timestamp" : 1049771140 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea71"), "UserID" : 36, "MovieID" : 1343, "Rating" : 4, "Timestamp" : 1049943810 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea72"), "UserID" : 36, "MovieID" : 1356, "Rating" : 3, "Timestamp" : 1049938690 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea73"), "UserID" : 36, "MovieID" : 1358, "Rating" : 4, "Timestamp" : 1049943680 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea74"), "UserID" : 36, "MovieID" : 1359, "Rating" : 3, "Timestamp" : 1049835010 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea75"), "UserID" : 36, "MovieID" : 1374, "Rating" : 4, "Timestamp" : 1049938620 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea76"), "UserID" : 36, "MovieID" : 1377, "Rating" : 4, "Timestamp" : 1049770500 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea77"), "UserID" : 36, "MovieID" : 1378, "Rating" : 5, "Timestamp" : 1049772990 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea78"), "UserID" : 36, "MovieID" : 1379, "Rating" : 3, "Timestamp" : 1049773310 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea79"), "UserID" : 36, "MovieID" : 1380, "Rating" : 4, "Timestamp" : 1049773120 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea7a"), "UserID" : 36, "MovieID" : 1387, "Rating" : 5, "Timestamp" : 1051844740 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea7b"), "UserID" : 36, "MovieID" : 1391, "Rating" : 3, "Timestamp" : 1049942140 }
{ "_id" : ObjectId("56b64d3cc54dfb7a020fea7c"), "UserID" : 36, "MovieID" : 1393, "Rating" : 4, "Timestamp" : 1049945860 }
```

Fig.2 ratings.json

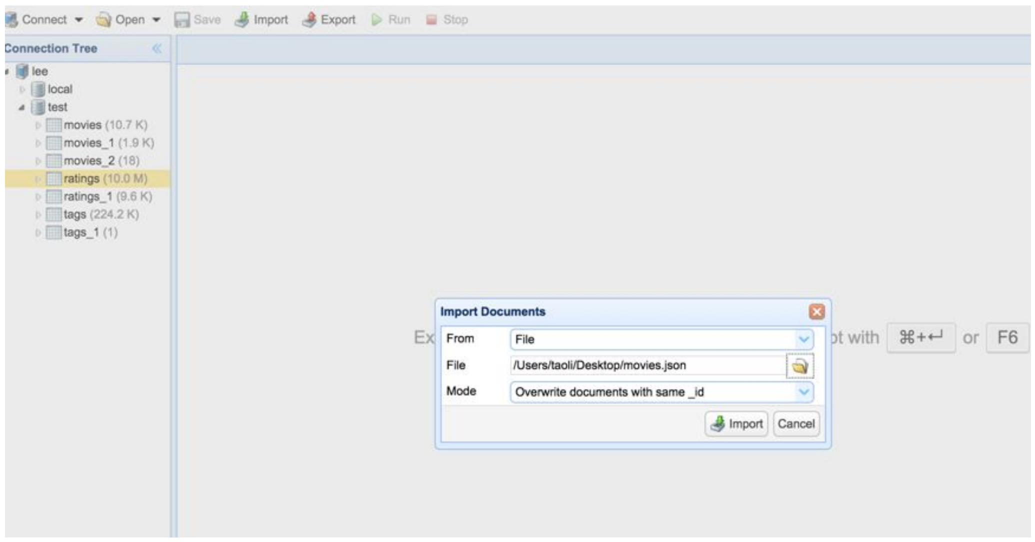


```
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537ed"), "UserID" : "1751", "MovieID" : "4410", "Tag" : "Demme", "Timestamp" : "1137529529" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537ee"), "UserID" : "1751", "MovieID" : "4440", "Tag" : "Chinese", "Timestamp" : "1137528704" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537ef"), "UserID" : "1751", "MovieID" : "4447", "Tag" : "girlie movie", "Timestamp" : "1137527003" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537f0"), "UserID" : "1751", "MovieID" : "4467", "Tag" : "Gilliam", "Timestamp" : "1137526116" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537f1"), "UserID" : "1751", "MovieID" : "4623", "Tag" : "sports", "Timestamp" : "1137527012" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537f2"), "UserID" : "1751", "MovieID" : "4718", "Tag" : "teen", "Timestamp" : "1137526978" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537f3"), "UserID" : "1751", "MovieID" : "4881", "Tag" : "coen bros", "Timestamp" : "1137526152" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537f4"), "UserID" : "1751", "MovieID" : "4886", "Tag" : "Pixar", "Timestamp" : "1137525425" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537f5"), "UserID" : "1751", "MovieID" : "4902", "Tag" : "Spanish", "Timestamp" : "1137528689" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537f6"), "UserID" : "1751", "MovieID" : "4973", "Tag" : "Jeunet", "Timestamp" : "1137524756" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537f7"), "UserID" : "1751", "MovieID" : "4979", "Tag" : "Wes Anderson", "Timestamp" : "1137528790" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537f8"), "UserID" : "1751", "MovieID" : "4992", "Tag" : "girlie movie", "Timestamp" : "1137527478" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537f9"), "UserID" : "1751", "MovieID" : "5066", "Tag" : "girlie movie", "Timestamp" : "1137527624" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537fa"), "UserID" : "1751", "MovieID" : "5139", "Tag" : "sports", "Timestamp" : "1137528747" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537fb"), "UserID" : "1751", "MovieID" : "5225", "Tag" : "Mexican", "Timestamp" : "1137529183" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537fc"), "UserID" : "1751", "MovieID" : "5267", "Tag" : "sports", "Timestamp" : "1137528784" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537fd"), "UserID" : "1751", "MovieID" : "5291", "Tag" : "Kurosawa", "Timestamp" : "1137525498" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537fe"), "UserID" : "1751", "MovieID" : "5299", "Tag" : "girlie movie", "Timestamp" : "1137527064" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d537ff"), "UserID" : "1751", "MovieID" : "5349", "Tag" : "super-hero", "Timestamp" : "1137528104" }
{ "_id" : ObjectId("56b6bf0f2a8bfa8e04d53800"), "UserID" : "1751", "MovieID" : "5378", "Tag" : "Lucas", "Timestamp" : "1137526200" }
```

Fig.3 tags.json

Step2: Import files to MongDB

In this step, we use MongoBooster to import three files



From: File
File: /Users/taoli/Desktop/movies.json
Mode: Overwrite documents with same _id

And we can also import files from Terminal.
Input the following code

```
mongoimport --host 127.0.0.1 --db test --collection movies --drop --file
/Users//Desktop/movies.json

mongoimport --host 127.0.0.1 --db test --collection ratings --drop --file
/Users/admin/Desktop/ratings.json

mongoimport --host 127.0.0.1 --db test --collection movies --drop --file
/Users/admin/Desktop/tags.json
```

Solutions

For each question, we use both methods of MongoShell and JAVA MongoDriver to solve. During our work, we find some problems about the tasks.

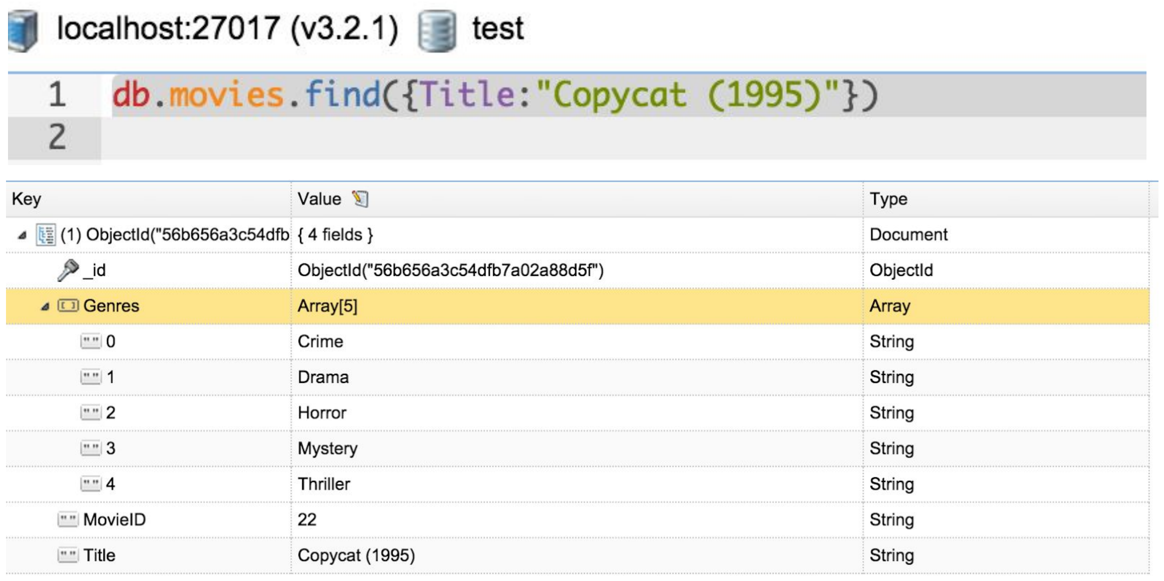
First, Q4, we consider another situation that a movie may rated highly by only a few people, and those kind of results are meaningfulness because they can not represent the majority of population.

Second, for Q5, we have some difficulty on understanding this question, the statement of this question may cause confusion. Since we have already complete question 4, obviously the possible highest average rating is 5.0, so it is a little confused about the goal of question 5.

After we discuss about this assignment, we have shown the final result below which in two versions.

Q1: What genre is the movie CopyCat in?

Query and Result


localhost:27017 (v3.2.1) test

```
db.movies.find({Title:"Copycat (1995)"})
```

Key	Value	Type
(1) ObjectId("56b656a3c54dfb	{ 4 fields }	Document
_id	ObjectId("56b656a3c54dfb7a02a88d5f")	ObjectId
Genres	Array[5]	Array
0	Crime	String
1	Drama	String
2	Horror	String
3	Mystery	String
4	Thriller	String
MovieID	22	String
Title	Copycat (1995)	String

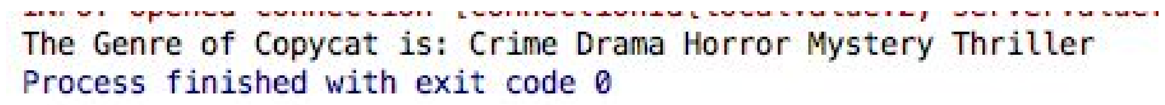
JAVA Program

```
public static void findCopyCat(DBCollection collection){
  DBCursor cursor = collection.find();
  String name;
  BasicDBObject node;
  while(cursor.hasNext()){
    node = (BasicDBObject) cursor.next();
```

```
        name = node.get("Title").toString();
        if(name.contains("Copycat")){
          String[] genre = node.get("Genres").toString().split(",");
          System.out.print("The Genre of Copycat is: ");
          for(String s:genre){
               String temp = s.substring(s.indexOf("\"")+1, s.lastIndexOf("\""));
               System.out.print(temp + " ");
          }
        }
      }

    }
```

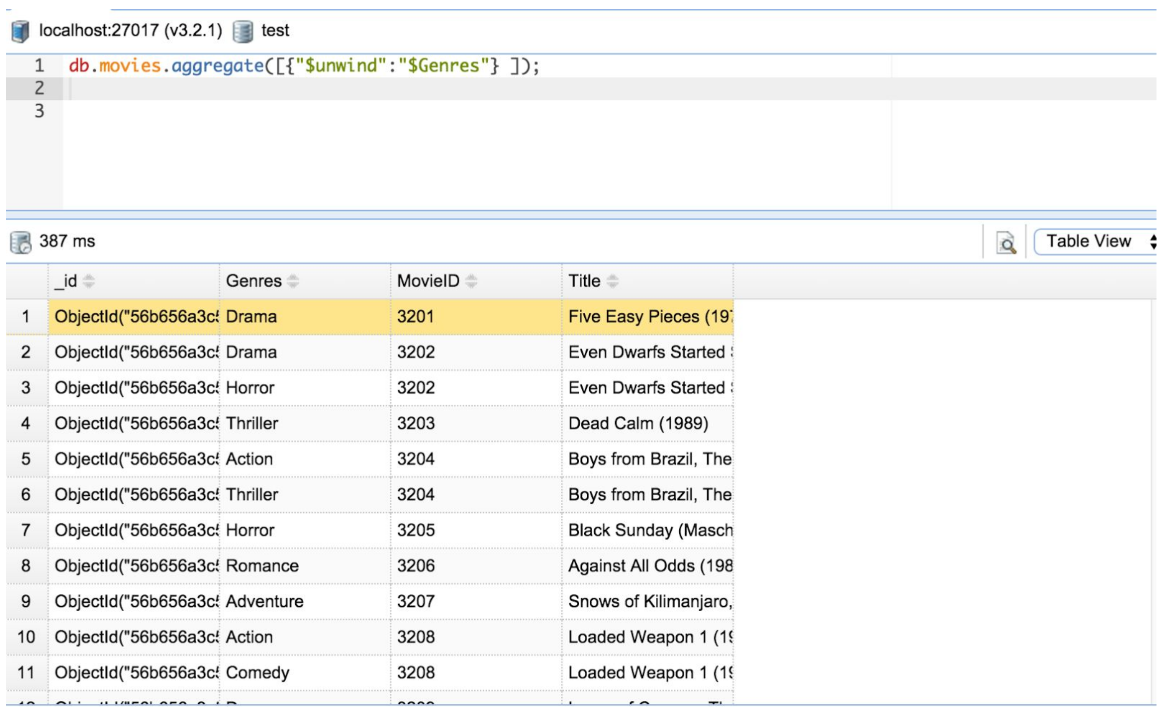
Result



```
The Genre of Copycat is: Crime Drama Horror Mystery Thriller
Process finished with exit code 0
```

Q2: What genre has the most movies?

Query and Result



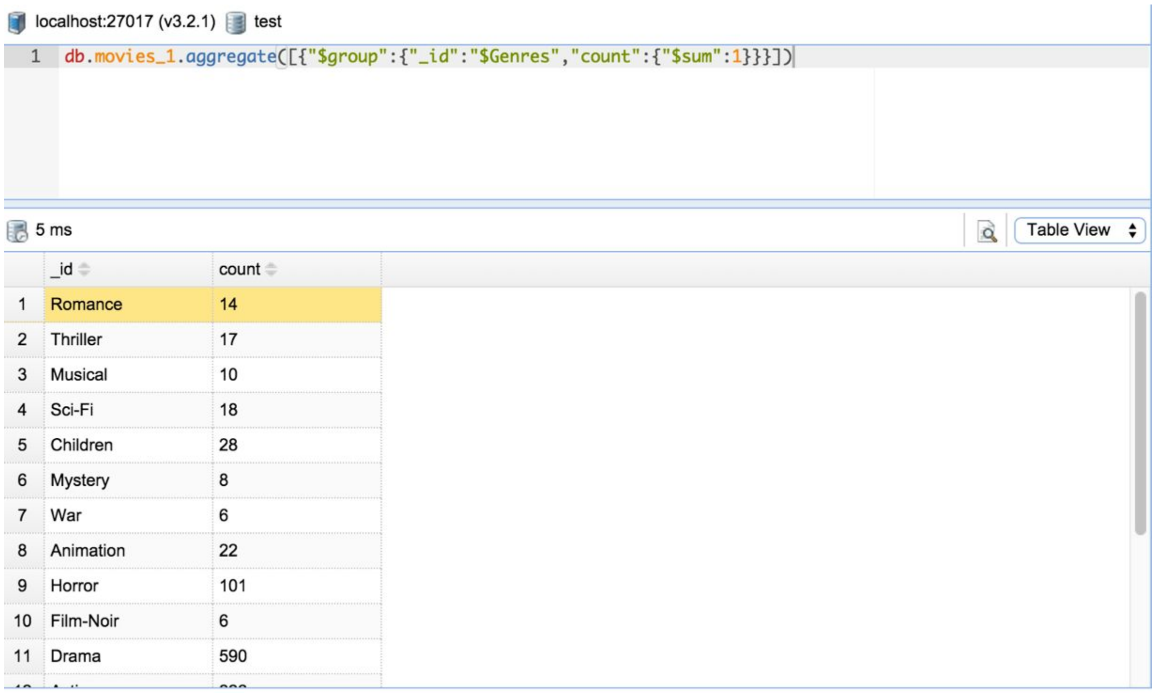
localhost:27017 (v3.2.1) test

```
db.movies.aggregate([{"$unwind":"$Genres"} ]);
```

387 ms   Table View

	_id	Genres	MovieID	Title
1	ObjectId("56b656a3c	Drama	3201	Five Easy Pieces (19
2	ObjectId("56b656a3c	Drama	3202	Even Dwarfs Started
3	ObjectId("56b656a3c	Horror	3202	Even Dwarfs Started
4	ObjectId("56b656a3c	Thriller	3203	Dead Calm (1989)
5	ObjectId("56b656a3c	Action	3204	Boys from Brazil, The
6	ObjectId("56b656a3c	Thriller	3204	Boys from Brazil, The
7	ObjectId("56b656a3c	Horror	3205	Black Sunday (Masch
8	ObjectId("56b656a3c	Romance	3206	Against All Odds (198
9	ObjectId("56b656a3c	Adventure	3207	Snows of Kilimanjaro,
10	ObjectId("56b656a3c	Action	3208	Loaded Weapon 1 (19
11	ObjectId("56b656a3c	Comedy	3208	Loaded Weapon 1 (19

The given movies.json has many records in the same Genres, so we deconstruct and make each record only has one genre. Then we store the results as movies_1 and do search in the collection



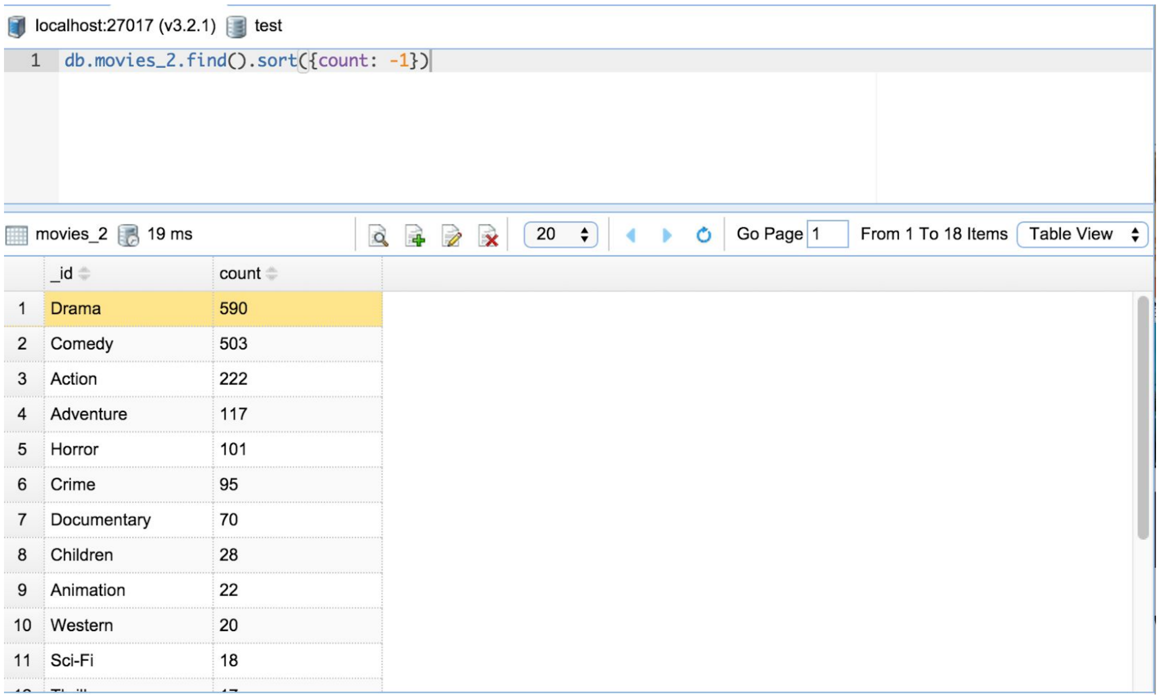
localhost:27017 (v3.2.1) test

```
db.movies_1.aggregate([{"$group":{"_id":"$Genres","count":{"$sum":1}}}])
```

5 ms Table View

	_id	count
1	Romance	14
2	Thriller	17
3	Musical	10
4	Sci-Fi	18
5	Children	28
6	Mystery	8
7	War	6
8	Animation	22
9	Horror	101
10	Film-Noir	6
11	Drama	590

We sort the result and find that Drama has the most movies



localhost:27017 (v3.2.1) test

```
db.movies_2.find().sort({count: -1})
```

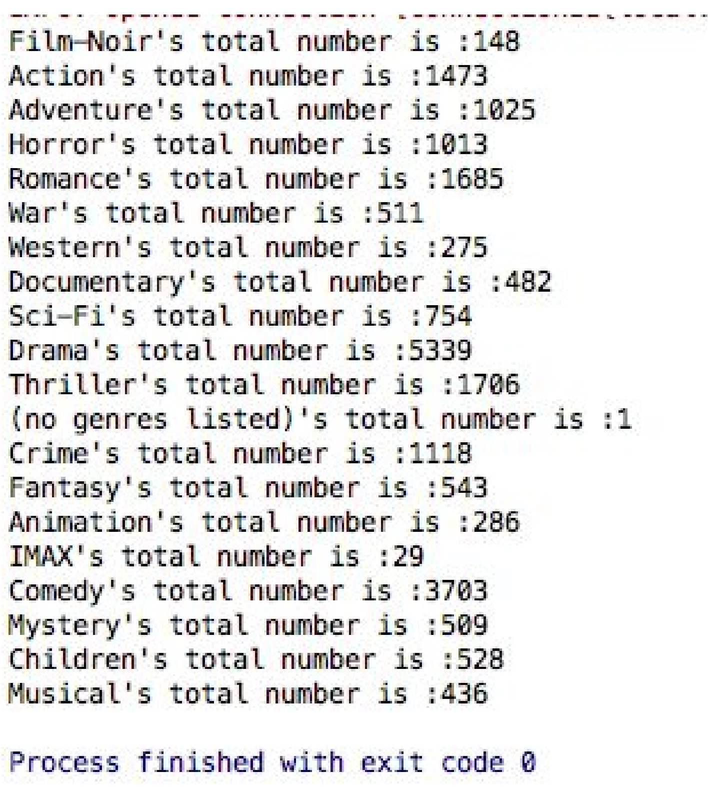
movies_2 19 ms 20 Go Page 1 From 1 To 18 Items Table View

	_id	count
1	Drama	590
2	Comedy	503
3	Action	222
4	Adventure	117
5	Horror	101
6	Crime	95
7	Documentary	70
8	Children	28
9	Animation	22
10	Western	20
11	Sci-Fi	18

JAVA Program

```
public static void CountGenres(DBCollection collection) {
  DBCursor cursor = collection.find();
  Map<String, Integer> countMap = new HashMap<String, Integer>();
  String[] genres;
  while(cursor.hasNext()) {
     genres = ((BasicDBObject) cursor.next()).get("Genres").toString().split(",");
     for(String s:genres){
          String temp = s.substring(s.indexOf("\"")+1, s.lastIndexOf("\""));
          if(temp.contains("\"")){
             continue;
          }
          else{
             if(countMap.containsKey(temp)){
                countMap.replace(temp, countMap.get(temp)+1);
             }else{
                countMap.put(temp, 1);
             }
          }
     }
  }
  for(String s: countMap.keySet()){
    System.out.println(s +"'s total number is :" + countMap.get(s));
  }
}
```

Result



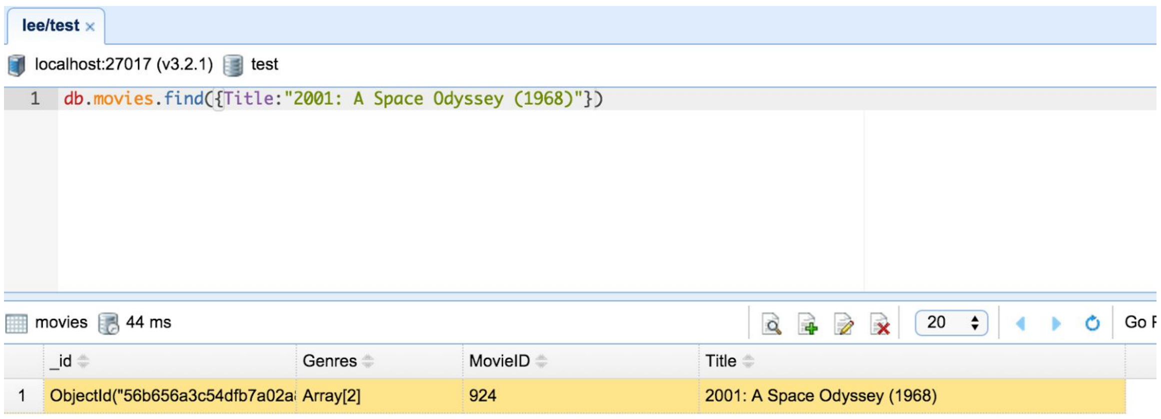
```
Film-Noir's total number is :148
Action's total number is :1473
Adventure's total number is :1025
Horror's total number is :1013
Romance's total number is :1685
War's total number is :511
Western's total number is :275
Documentary's total number is :482
Sci-Fi's total number is :754
Drama's total number is :5339
Thriller's total number is :1706
(no genres listed)'s total number is :1
Crime's total number is :1118
Fantasy's total number is :543
Animation's total number is :286
IMAX's total number is :29
Comedy's total number is :3703
Mystery's total number is :509
Children's total number is :528
Musical's total number is :436

Process finished with exit code 0
```

Q3: what tags did user 146 use to describe the movie "2001: A Space Odyssey"

Query and Result

First we find the MovieID of this movies



lee/test

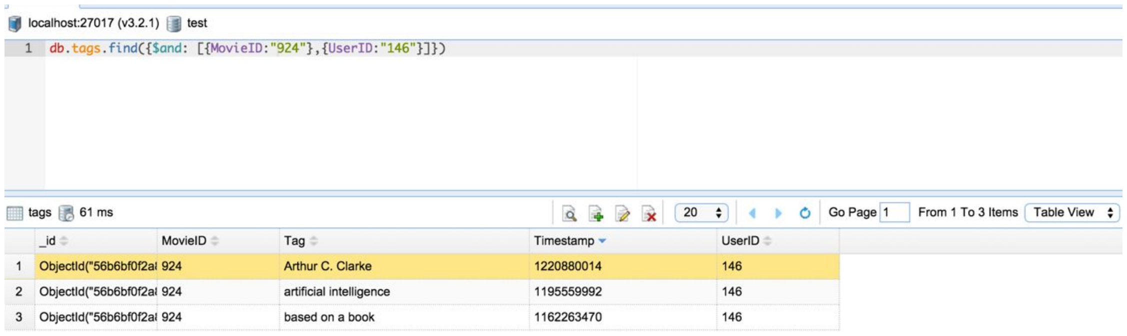
localhost:27017 (v3.2.1) test

```
db.movies.find({Title:"2001: A Space Odyssey (1968)"})
```

movies 44 ms

	_id	Genres	MovieID	Title
1	ObjectId("56b656a3c54dfb7a02a	Array[2]	924	2001: A Space Odyssey (1968)

Then, we use the MovieID to do search in the tags collection. We get the tags that user 146 used to describe the movie "2001: A Space Odyssey"



localhost:27017 (v3.2.1) test

```
db.tags.find({$and: [{MovieID:"924"},{UserID:"146"}]})
```

tags 61 ms

From 1 To 3 Items

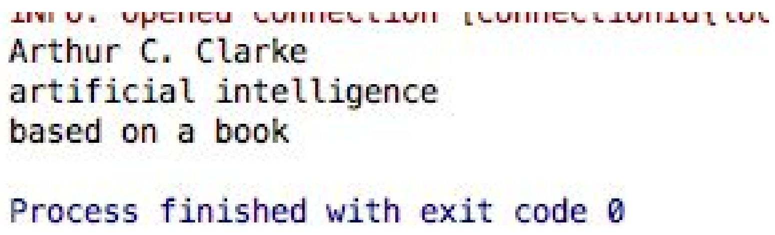
	_id	MovieID	Tag	Timestamp	UserID
1	ObjectId("56b6bf0f2a	924	Arthur C. Clarke	1220880014	146
2	ObjectId("56b6bf0f2a	924	artificial intelligence	1195559992	146
3	ObjectId("56b6bf0f2a	924	based on a book	1162263470	146

JAVA Program

```
public static void findUserTags(DBCollection collectionMovie,DBCollection collectionTag) {
   BasicDBObject queryMovieID = new BasicDBObject();
   queryMovieID.put("Title", "2001: A Space Odyssey (1968)");
   DBCursor cursorMovieID = collectionMovie.find(queryMovieID);
   String movieID = cursorMovieID.next().get("MovieID").toString();
   BasicDBObject queryTag = new BasicDBObject();
   Map<String,String> queryMap = new HashMap<String,String>();
   queryMap.put("MovieID", movieID);
   queryMap.put("UserID", "146");
   queryTag.putAll(queryMap);
   DBCursor cursorTag = collectionTag.find(queryTag);
   String tag;
```

```
  while(cursorTag.hasNext()){
    tag = cursorTag.next().get("Tag").toString();
    System.out.println(tag);
  }
}
```

Result



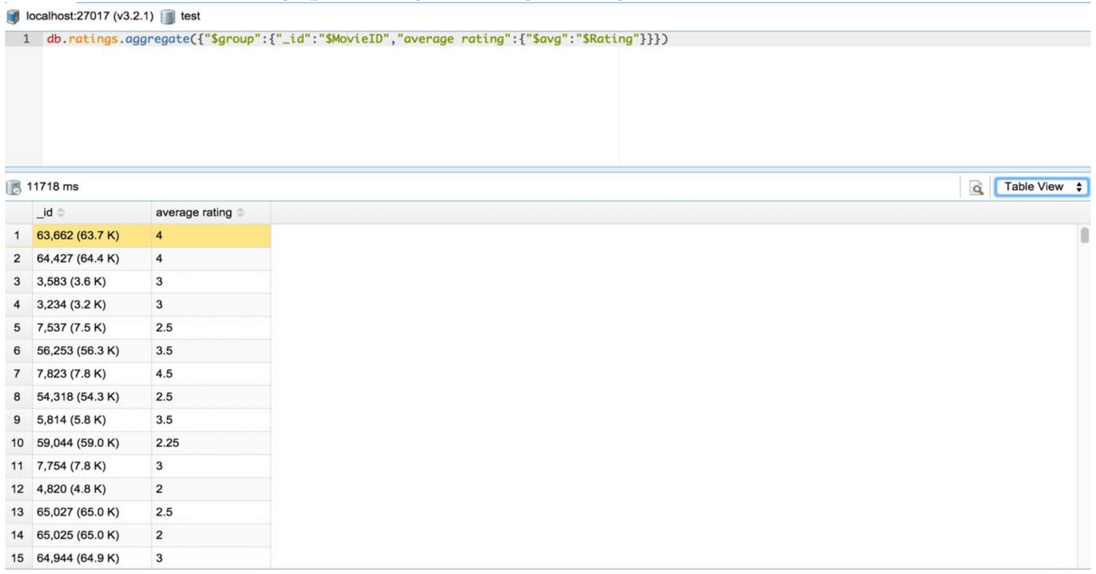
```
INFO: opened connection [connectionId{loc
Arthur C. Clarke
artificial intelligence
based on a book

Process finished with exit code 0
```

Q4: What are the top 5 movies with the highest avg rating?

Query and Result

First we use the following query to get average ratings of each movie



localhost:27017 (v3.2.1) test

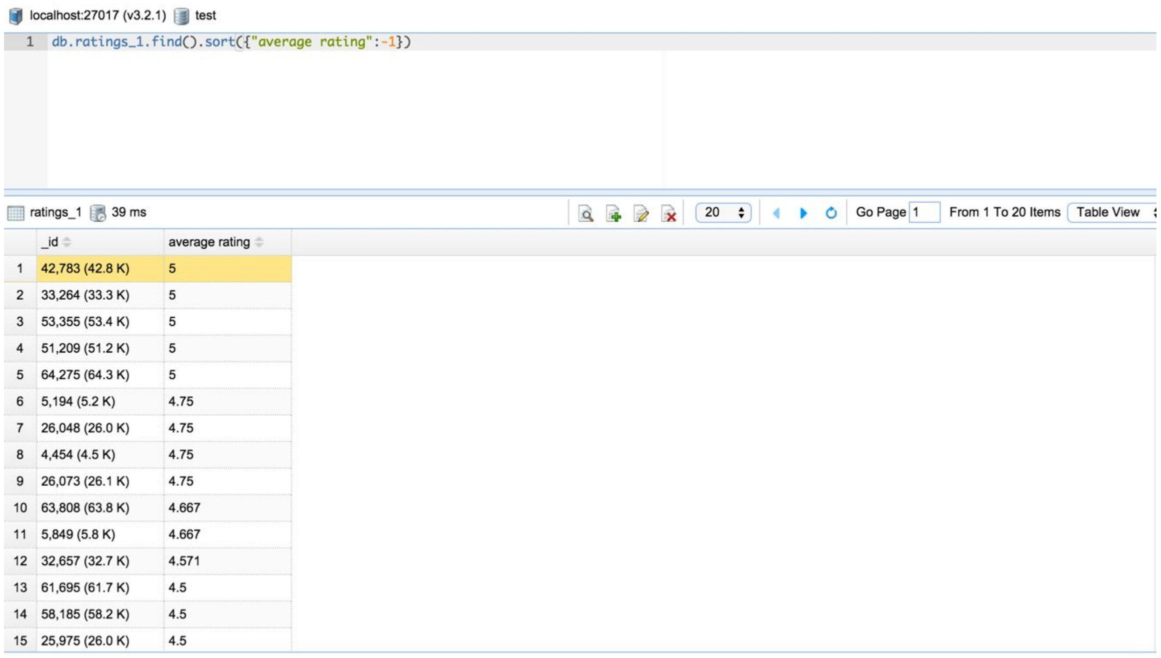
```
db.ratings.aggregate({"$group":{"_id":"$MovieID","average rating":{"$avg":"$Rating"}}})
```

11718 ms

Table View

	_id	average rating
1	63,662 (63.7 K)	4
2	64,427 (64.4 K)	4
3	3,583 (3.6 K)	3
4	3,234 (3.2 K)	3
5	7,537 (7.5 K)	2.5
6	56,253 (56.3 K)	3.5
7	7,823 (7.8 K)	4.5
8	54,318 (54.3 K)	2.5
9	5,814 (5.8 K)	3.5
10	59,044 (59.0 K)	2.25
11	7,754 (7.8 K)	3
12	4,820 (4.8 K)	2
13	65,027 (65.0 K)	2.5
14	65,025 (65.0 K)	2
15	64,944 (64.9 K)	3

Then we sort the results and it is easy to find the top 5 movies with highest avg rating.



localhost:27017 (v3.2.1) test

```
db.ratings_1.find().sort({"average rating":-1})
```

ratings_1 39 ms

20 Go Page 1 From 1 To 20 Items Table View

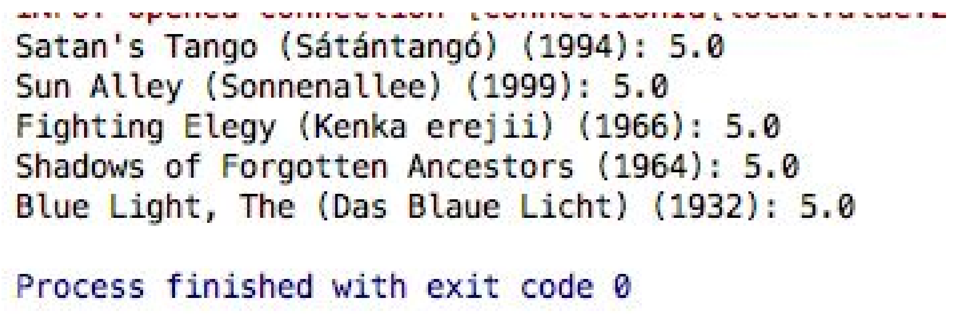
	_id	average rating
1	42,783 (42.8 K)	5
2	33,264 (33.3 K)	5
3	53,355 (53.4 K)	5
4	51,209 (51.2 K)	5
5	64,275 (64.3 K)	5
6	5,194 (5.2 K)	4.75
7	26,048 (26.0 K)	4.75
8	4,454 (4.5 K)	4.75
9	26,073 (26.1 K)	4.75
10	63,808 (63.8 K)	4.667
11	5,849 (5.8 K)	4.667
12	32,657 (32.7 K)	4.571
13	61,695 (61.7 K)	4.5
14	58,185 (58.2 K)	4.5
15	25,975 (26.0 K)	4.5

JAVA Program

```java
public static void findTopFiveMovies(DBCollection collectionRating,DBCollection collectionMovie) {
  BasicDBObject queryAveRating = new BasicDBObject();
  queryAveRating.put("$group", new BasicDBObject("_id", "$MovieID").append("average", new
BasicDBObject("$avg", "$Rating")));
  DBObject sort = new BasicDBObject("$sort", new BasicDBObject("average",-1));
  AggregationOutput output = collectionRating.aggregate(queryAveRating, sort);
  Iterator<DBObject> outputIterator = output.results().iterator();
  int i = 0;
  String[] movieIDs = new String[5];
  String[] ratings = new String[5];
  while(outputIterator.hasNext() && i<5){
    String string = outputIterator.next().toString();
    ratings[i] = string.split(",")[1].split(":")[1].trim();
    String temp = string.split(",")[0].split(":")[1].trim();
    temp = temp.substring(0, temp.length()-2);
    movieIDs[i] = temp;
    i++;
  }

  String[] name = new String[movieIDs.length];
  int j = 0;
  for(String s:movieIDs){
    BasicDBObject queryMovieTitle = new BasicDBObject();
    queryMovieTitle.put("MovieID", s);
    DBCursor cursorMovieTitle = collectionMovie.find(queryMovieTitle);
    name[j] = cursorMovieTitle.next().get("Title").toString();
    System.out.println(name[j] + ": " + ratings[j].substring(0,3));
    j++;
  }
}
```

Result



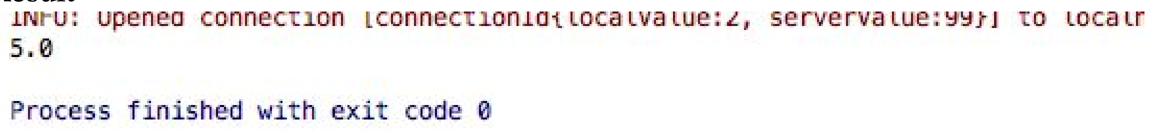
```
Satan's Tango (Sátántangó) (1994): 5.0
Sun Alley (Sonnenallee) (1999): 5.0
Fighting Elegy (Kenka erejii) (1966): 5.0
Shadows of Forgotten Ancestors (1964): 5.0
Blue Light, The (Das Blaue Licht) (1932): 5.0

Process finished with exit code 0
```

Q5: What is the highest avg rating possible?
JAVA Program

```
public static void findHighestRating(DBCollection collectionRating) {
  BasicDBObject queryAveRating = new BasicDBObject();
  queryAveRating.put("$group", new BasicDBObject("_id", "$MovieID").append("average rating",
new BasicDBObject("$avg", "$Rating")));
  DBObject sort = new BasicDBObject("$sort", new BasicDBObject("average rating",-1));
  AggregationOutput output = collectionRating.aggregate(queryAveRating, sort);
  Iterator<DBObject> outputIterator = output.results().iterator();
  String rawRating = outputIterator.next().toString().split(",")[1].split(":")[1].trim();
  String rating = rawRating.substring(0,rawRating.length()-1);
  System.out.println(rating);
}
```

Result

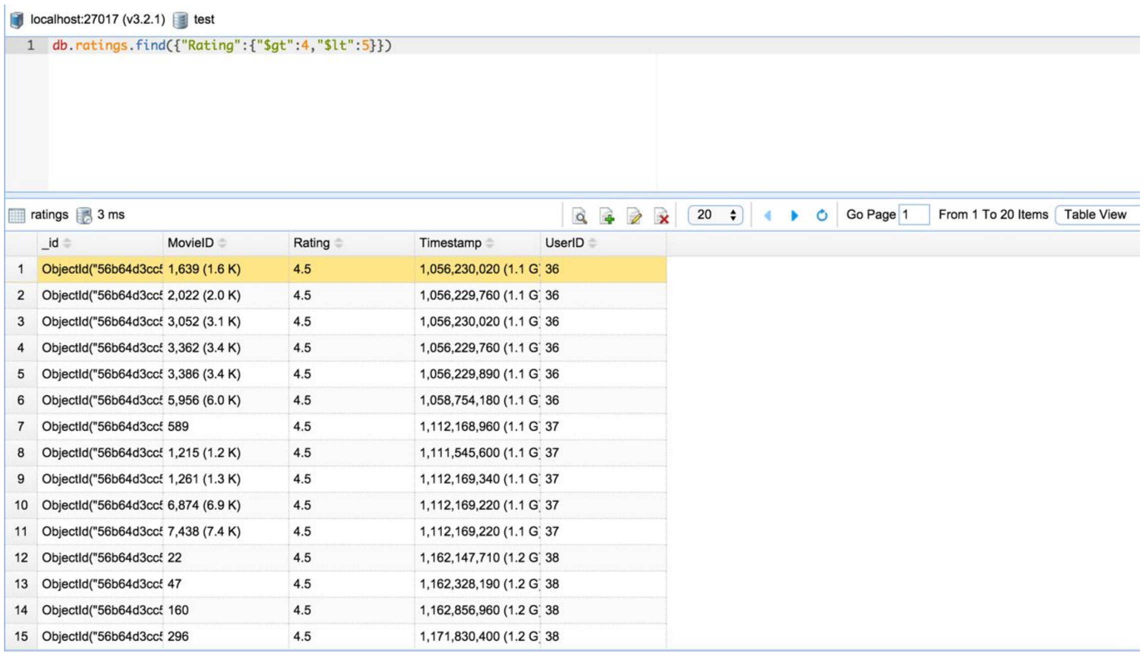


```
INFO: Opened connection [connectionId{localValue:2, serverValue:99}] to localh
5.0

Process finished with exit code 0
```

Q6: Write 3 different queries of your choice to demonstrate that your data storage is working.

(1) **The first query is to search movies with rating between 4 and 5**
Query and Result


localhost:27017 (v3.2.1) test

```
db.ratings.find({"Rating":{"$gt":4,"$lt":5}})
```

ratings 3 ms

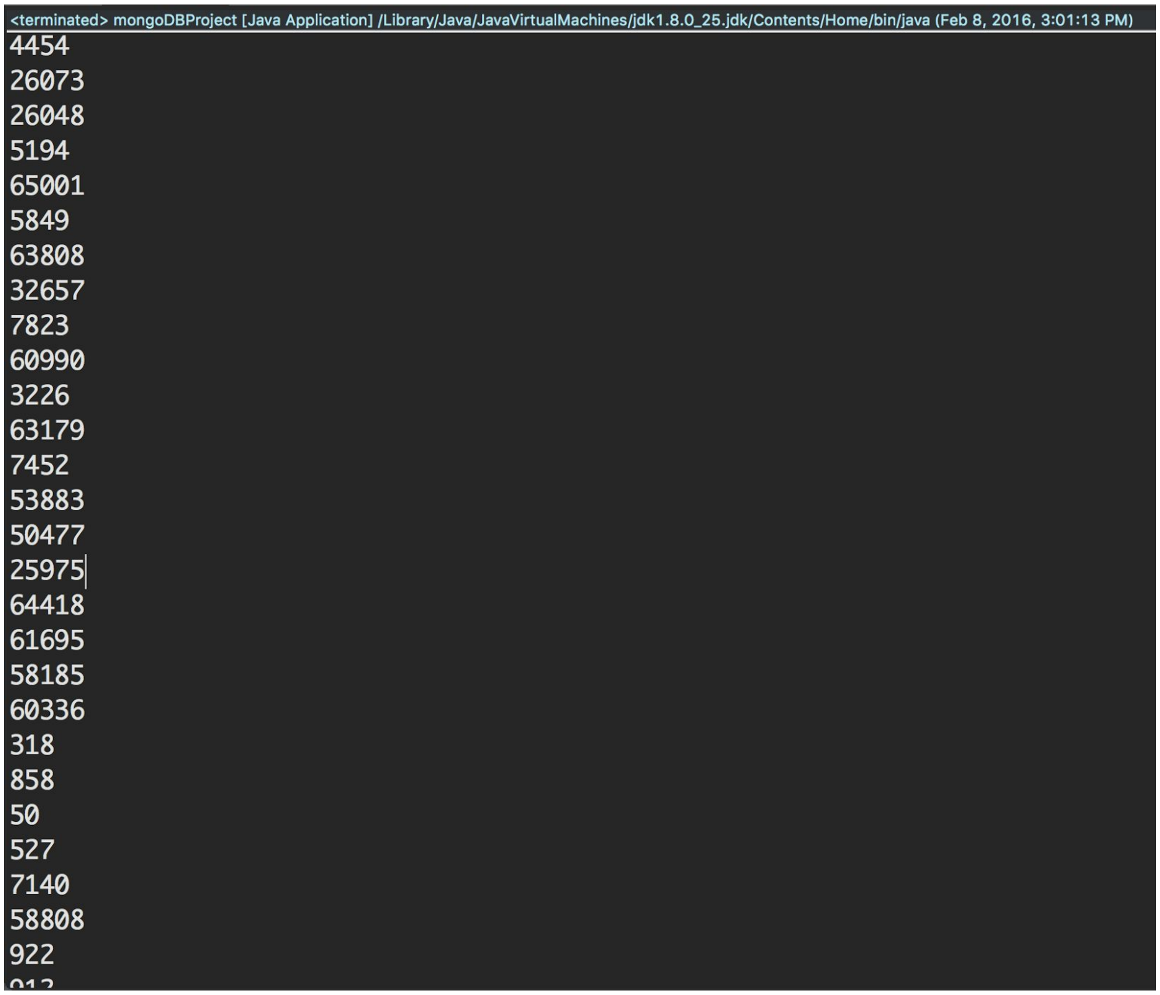
Go Page 1 From 1 To 20 Items Table View

	_id	MovieID	Rating	Timestamp	UserID
1	ObjectId("56b64d3cc5	1,639 (1.6 K)	4.5	1,056,230,020 (1.1 G	36
2	ObjectId("56b64d3cc5	2,022 (2.0 K)	4.5	1,056,229,760 (1.1 G	36
3	ObjectId("56b64d3cc5	3,052 (3.1 K)	4.5	1,056,230,020 (1.1 G	36
4	ObjectId("56b64d3cc5	3,362 (3.4 K)	4.5	1,056,229,760 (1.1 G	36
5	ObjectId("56b64d3cc5	3,386 (3.4 K)	4.5	1,056,229,890 (1.1 G	36
6	ObjectId("56b64d3cc5	5,956 (6.0 K)	4.5	1,058,754,180 (1.1 G	36
7	ObjectId("56b64d3cc5	589	4.5	1,112,168,960 (1.1 G	37
8	ObjectId("56b64d3cc5	1,215 (1.2 K)	4.5	1,111,545,600 (1.1 G	37
9	ObjectId("56b64d3cc5	1,261 (1.3 K)	4.5	1,112,169,340 (1.1 G	37
10	ObjectId("56b64d3cc5	6,874 (6.9 K)	4.5	1,112,169,220 (1.1 G	37
11	ObjectId("56b64d3cc5	7,438 (7.4 K)	4.5	1,112,169,220 (1.1 G	37
12	ObjectId("56b64d3cc5	22	4.5	1,162,147,710 (1.2 G	38
13	ObjectId("56b64d3cc5	47	4.5	1,162,328,190 (1.2 G	38
14	ObjectId("56b64d3cc5	160	4.5	1,162,856,960 (1.2 G	38
15	ObjectId("56b64d3cc5	296	4.5	1,171,830,400 (1.2 G	38

JAVA Program

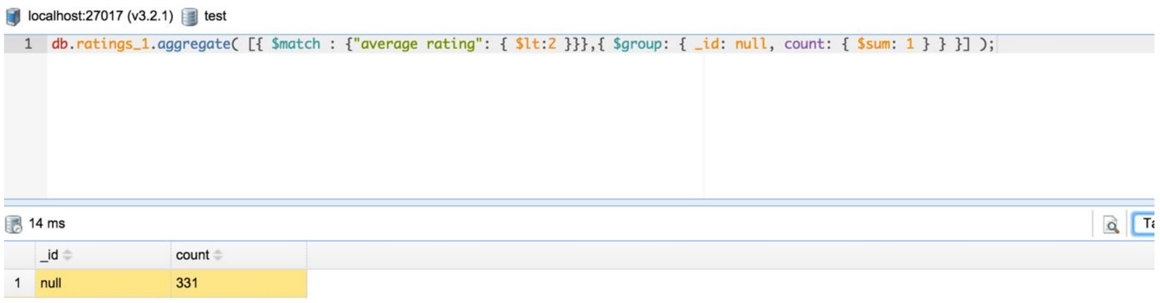
```java
public static void findRatingBetween(DBCollection collectionRating){
  BasicDBObject queryAveRating = new BasicDBObject();
  queryAveRating.put("$group", new BasicDBObject("_id", "$MovieID").append("average rating",
new BasicDBObject("$avg", "$Rating")));
  DBObject sort = new BasicDBObject("$sort", new BasicDBObject("average rating",-1));
  AggregationOutput output = collectionRating.aggregate(queryAveRating, sort);
  Iterator<DBObject> outputIterator = output.results().iterator();
  ArrayList<String> movieIDs = new ArrayList<>();
  while(outputIterator.hasNext()){
    String string = outputIterator.next().toString();
    String rawRating = string.split(",")[1].split(":")[1].trim();
    float rating = Float.parseFloat(rawRating.substring(0,rawRating.length()-1));
    if(rating<5 && rating>4){
      movieIDs.add(string.split(",")[0].split(":")[1].trim());
    }
  }
  for(int i=0;i<movieIDs.size();i++){
    System.out.println(movieIDs.get(i));
  }
}
```

Result



```
<terminated> mongoDBProject [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home/bin/java (Feb 8, 2016, 3:01:13 PM)
4454
26073
26048
5194
65001
5849
63808
32657
7823
60990
3226
63179
7452
53883
50477
25975
64418
61695
58185
60336
318
858
50
527
7140
58808
922
```

(2) **The second query is to count movies with rating under 2**
Query and Result



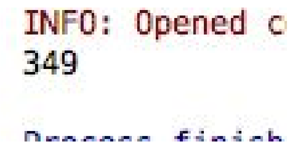
```
db.ratings_1.aggregate( [{ $match : {"average rating": { $lt:2 }}},{ $group: { _id: null, count: { $sum: 1 } } }] );
```

_id	count
null	331

JAVA Program

```
public static void findRatingUnder(DBCollection collectionRating){
  BasicDBObject queryAveRating = new BasicDBObject();
  queryAveRating.put("$group", new BasicDBObject("_id", "$MovieID").append("average
rating", new BasicDBObject("$avg", "$Rating")));
  DBObject sort = new BasicDBObject("$sort", new BasicDBObject("average rating",1));
  AggregationOutput output = collectionRating.aggregate(queryAveRating, sort);
  Iterator<DBObject> outputIterator = output.results().iterator();
  int count = 0;
  while(outputIterator.hasNext()){
    String string = outputIterator.next().toString();
    String rawRating = string.split(",")[1].split(":")[1].trim();
    float rating = Float.parseFloat(rawRating.substring(0,rawRating.length()-1));
    if(rating<2){
      count ++;
    }
  }
  System.out.println(count);
}
```

Result

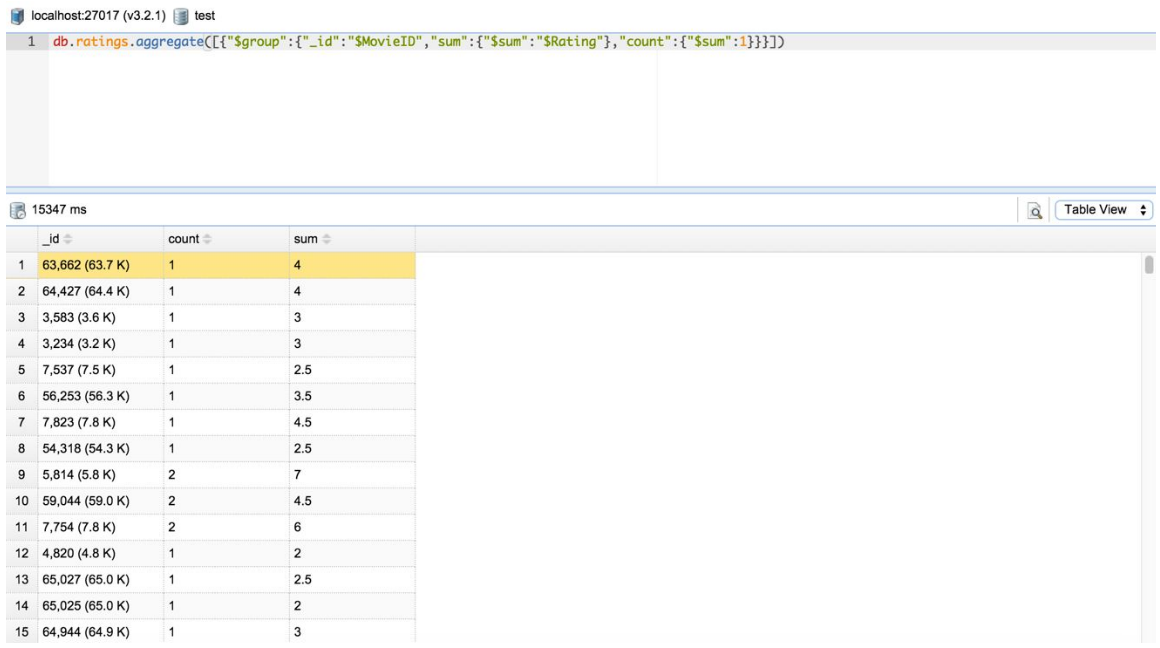


```
INFO: Opened c
349

Process finish
```

(3) The third query is to compute the total rating of each movie and the number of people who rated the movie

Query and Result



localhost:27017 (v3.2.1) test

```
db.ratings.aggregate([{"$group":{"_id":"$MovieID","sum":{"$sum":"$Rating"},"count":{"$sum":1}}}])
```

15347 ms

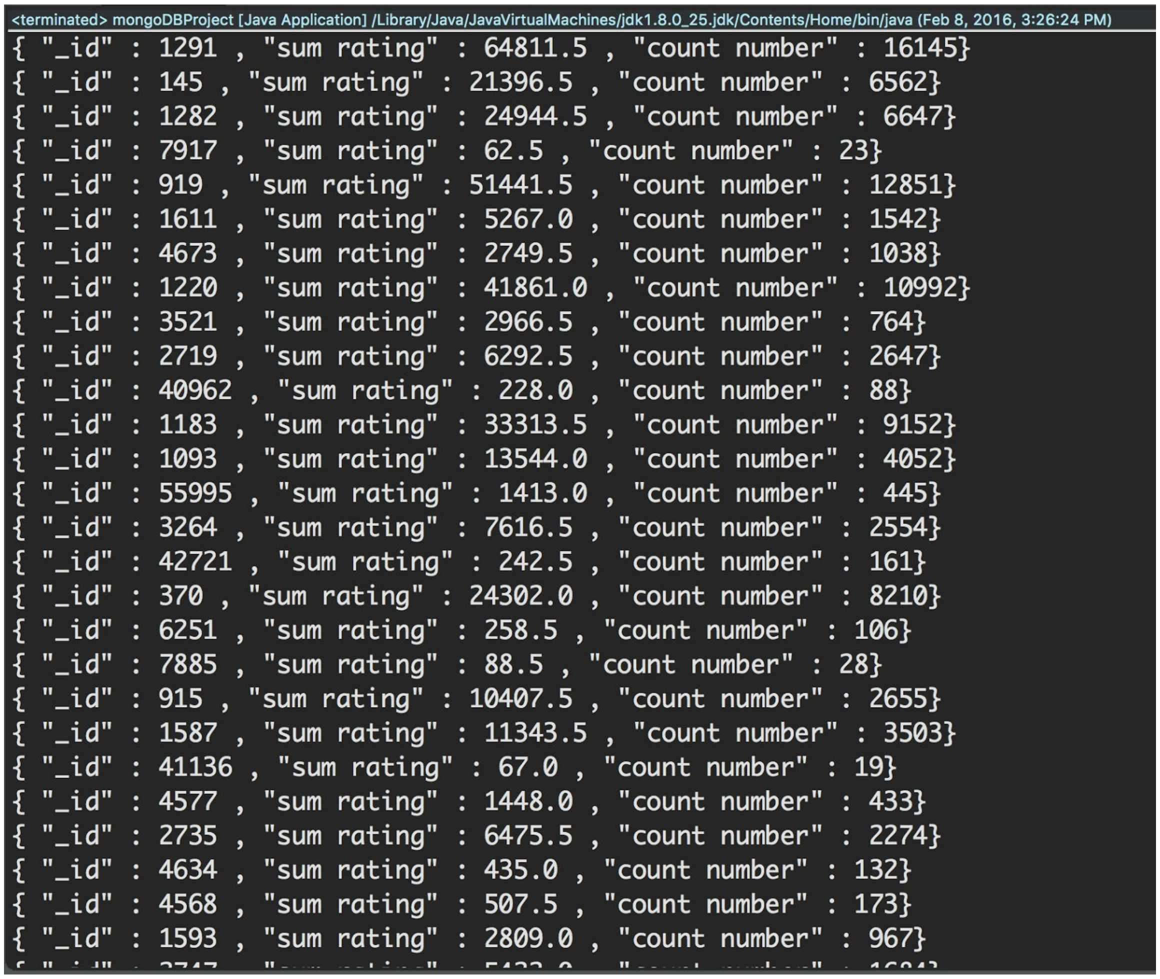
	_id	count	sum
1	63,662 (63.7 K)	1	4
2	64,427 (64.4 K)	1	4
3	3,583 (3.6 K)	1	3
4	3,234 (3.2 K)	1	3
5	7,537 (7.5 K)	1	2.5
6	56,253 (56.3 K)	1	3.5
7	7,823 (7.8 K)	1	4.5
8	54,318 (54.3 K)	1	2.5
9	5,814 (5.8 K)	2	7
10	59,044 (59.0 K)	2	4.5
11	7,754 (7.8 K)	2	6
12	4,820 (4.8 K)	1	2
13	65,027 (65.0 K)	1	2.5
14	65,025 (65.0 K)	1	2
15	64,944 (64.9 K)	1	3

JAVA Program

```
public static void findTotalRatingAndNumPeople(DBCollection collectionRating){
  BasicDBObject querySumRating = new BasicDBObject();
```

```
    querySumRating.put("$group", new BasicDBObject("_id", "$MovieID").append("sum rating",
new BasicDBObject("$sum", "$Rating")).append("count number", new BasicDBObject("$sum",
1)));
    AggregationOutput output = collectionRating.aggregate(querySumRating);
    Iterator<DBObject> outputIterator = output.results().iterator();
    while(outputIterator.hasNext()){
      System.out.println(outputIterator.next().toString());
    }
  }
}
```

Result



```
<terminated> mongoDBProject [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home/bin/java (Feb 8, 2016, 3:26:24 PM)
{ "_id" : 1291 , "sum rating" : 64811.5 , "count number" : 16145}
{ "_id" : 145 , "sum rating" : 21396.5 , "count number" : 6562}
{ "_id" : 1282 , "sum rating" : 24944.5 , "count number" : 6647}
{ "_id" : 7917 , "sum rating" : 62.5 , "count number" : 23}
{ "_id" : 919 , "sum rating" : 51441.5 , "count number" : 12851}
{ "_id" : 1611 , "sum rating" : 5267.0 , "count number" : 1542}
{ "_id" : 4673 , "sum rating" : 2749.5 , "count number" : 1038}
{ "_id" : 1220 , "sum rating" : 41861.0 , "count number" : 10992}
{ "_id" : 3521 , "sum rating" : 2966.5 , "count number" : 764}
{ "_id" : 2719 , "sum rating" : 6292.5 , "count number" : 2647}
{ "_id" : 40962 , "sum rating" : 228.0 , "count number" : 88}
{ "_id" : 1183 , "sum rating" : 33313.5 , "count number" : 9152}
{ "_id" : 1093 , "sum rating" : 13544.0 , "count number" : 4052}
{ "_id" : 55995 , "sum rating" : 1413.0 , "count number" : 445}
{ "_id" : 3264 , "sum rating" : 7616.5 , "count number" : 2554}
{ "_id" : 42721 , "sum rating" : 242.5 , "count number" : 161}
{ "_id" : 370 , "sum rating" : 24302.0 , "count number" : 8210}
{ "_id" : 6251 , "sum rating" : 258.5 , "count number" : 106}
{ "_id" : 7885 , "sum rating" : 88.5 , "count number" : 28}
{ "_id" : 915 , "sum rating" : 10407.5 , "count number" : 2655}
{ "_id" : 1587 , "sum rating" : 11343.5 , "count number" : 3503}
{ "_id" : 41136 , "sum rating" : 67.0 , "count number" : 19}
{ "_id" : 4577 , "sum rating" : 1448.0 , "count number" : 433}
{ "_id" : 2735 , "sum rating" : 6475.5 , "count number" : 2274}
{ "_id" : 4634 , "sum rating" : 435.0 , "count number" : 132}
{ "_id" : 4568 , "sum rating" : 507.5 , "count number" : 173}
{ "_id" : 1593 , "sum rating" : 2809.0 , "count number" : 967}
```