Map reduce on 10M Movie Lens Dataset

Write a Java (could be a console app - will only run once to import the data into MongoDB) program to read the following file, and insert into 3 different collections (movies, ratings, tags).
- MovieLens 10M Stable benchmark dataset. 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users.

http://grouplens.org/datasets/movielens/

Java API to import dataset:

```
package movieapi;

import com.mongodb.client.*;
import com.mongodb.client.MongoDatabase;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.Reader;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import org.bson.Document;

public class MovieAPI {

   public static ArrayList movies = new ArrayList<String>();
   public static ArrayList ratings;
   public static ArrayList tags = new ArrayList<String>();

   public static void main(String[] args) throws FileNotFoundException {
      /*Scanner sc = new Scanner(new File("C:\\Users\\SayaliGirish\\Desktop\\ML\\movies.dat"));
       while (sc.hasNextLine()) {
       movies.add(sc.nextLine());
       }*/
      Scanner sc1 = new Scanner(new File("C:\\Users\\SayaliGirish\\Desktop\\ML\\ratings.dat"));
      int chunk = 50000;
      MongoClient connection = MongoClients.create();
      MongoDatabase db = connection.getDatabase("MovieLensAPI");
      MongoCollection<Document> collection1 = db.getCollection("movies");
      MongoCollection<Document> collection2 = db.getCollection("ratings");
      MongoCollection<Document> collection3 = db.getCollection("tags");
      List<Document> documents1 = new ArrayList<Document>();
      List<Document> documents3 = new ArrayList<Document>();
      int k = 0;
      while (sc1.hasNextLine()) {
         System.out.println("Outer loop count " + k++);
         for (int j = 0; j < chunk; j++) {
```

```
            ratings = new ArrayList<String>();
            ratings.add(sc1.nextLine());
            if (!sc1.hasNextLine()) {
               System.out.println("Returning");
            }
            for (int i = 0; i < ratings.size(); i++) {
               String s = ratings.get(i).toString();
               String[] arrayOfStrings = s.split("::");
               List<Document> documents2 = new ArrayList<Document>();
               documents2.add(new Document("userid", arrayOfStrings[0])
                     .append("movieid", arrayOfStrings[1])
                     .append("rating", arrayOfStrings[2])
                     .append("timestamp", arrayOfStrings[3]));
               collection2.insertMany(documents2);
            }
         }
      }
      Scanner sc2 = new Scanner(new File("C:\\Users\\SayaliGirish\\Desktop\\ML\\tags.dat"));
      while (sc2.hasNextLine()) {
         tags.add(sc2.nextLine());
      }
      for (int i = 0; i < movies.size(); i++) {
         String s = movies.get(i).toString();
         String[] arrayOfStrings = s.split("::");
         documents1.add(new Document("movieid", arrayOfStrings[0])
               .append("title", arrayOfStrings[1])
               .append("genere", arrayOfStrings[2]));
      }
      collection1.insertMany(documents1);

      for (int i = 0; i < tags.size(); i++) {
         String s = tags.get(i).toString();
         String[] arrayOfStrings = s.split("::");
         documents3.add(new Document("userid", arrayOfStrings[0])
               .append("movieid", arrayOfStrings[1])
               .append("tag", arrayOfStrings[2])
               .append("timestamp", arrayOfStrings[3]));
      }
      collection3.insertMany(documents3);
   }

}
```

Map reduce on inserted data

1] Number of Movies released per year (Movies Collection)

Map function- function() {
var year = this.title.match(/\(\d\d\d\d\)/);
if (year) { var key = year[0];}
var value = { count:1};
 emit(key,value); }

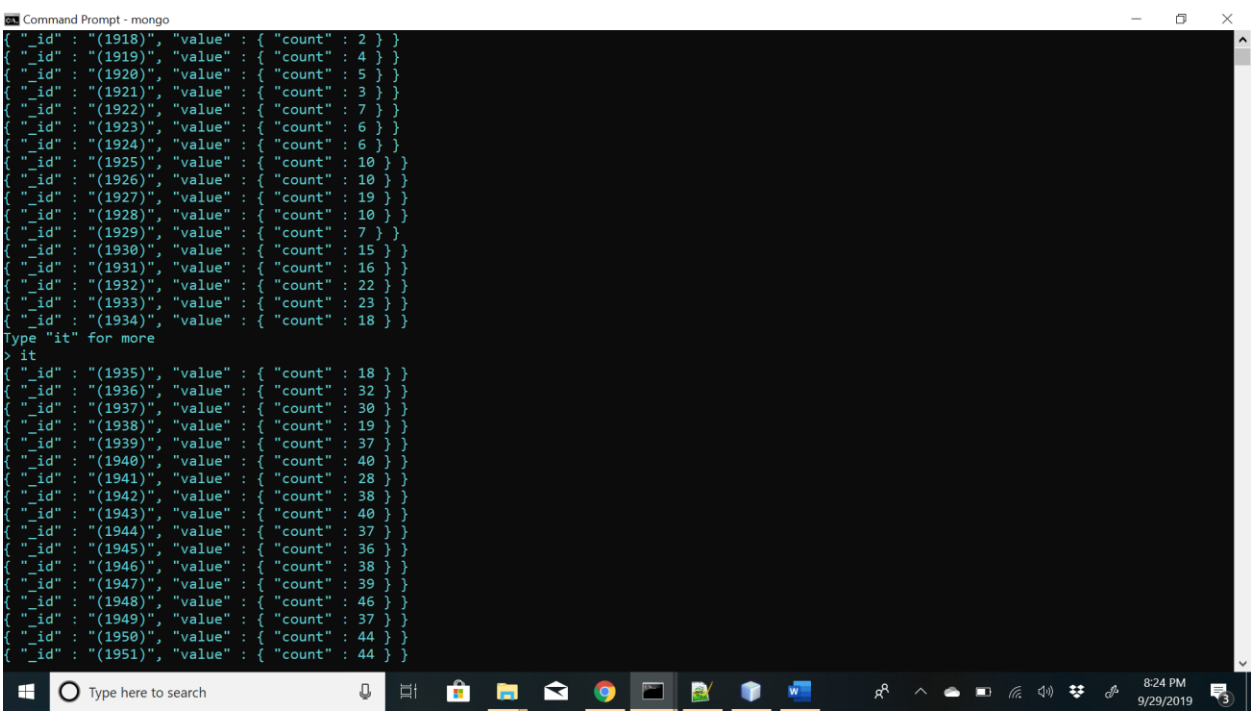
Reduce function- function(key, values)
 { counterVal = { count:0};
for (var id = 0; id < values.length; id++)
{ counterVal.count += values[id].count ; }
return counterVal; }

Mongodb command to run map reduce:

```
db.movies.mapReduce(
                mapFunction1,
                reduceFunction1,
                { out: "map_reduce_example" }
               )
```

Ps: I have not written this command again and again in the document but use this command every time when you run map reduce

Output:



2] Number of Movies per genre (Movies Collection)

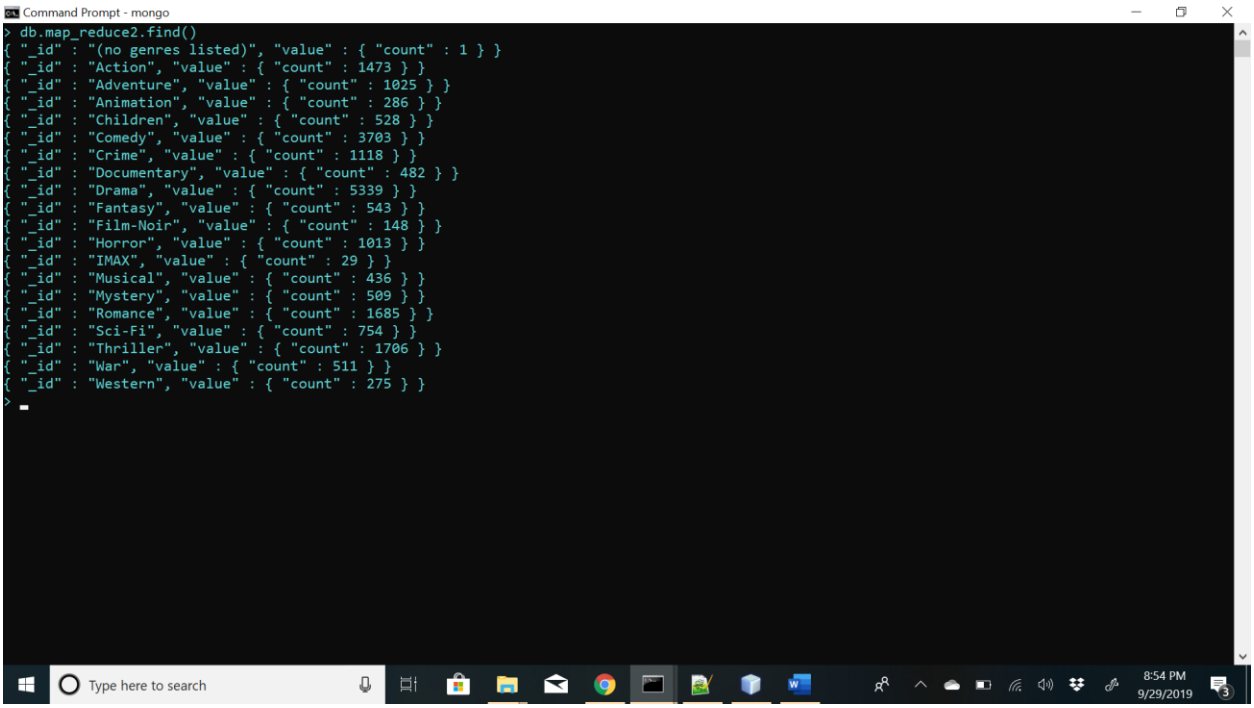
Map function-

```
function() {
var genresList = this.genres.split("|");
for (idx=0; idx<genresList.length;idx++)
 { var key=genresList[idx];
 var values = { count:1};
emit(key,values);
}  }
```

Reduce Function-

```
function(key, values) {
 counterVal = { count:0};
for (var id = 0; id < values.length; id++)
{ counterVal.count  += values[id].count ; } r
eturn counterVal; }
```

Output-



3] Number of movies per ratings (Ratings Collection)

Map Function-

```
function()
 {  var value = { count:1};
```

```
 var key = this.rating;
 emit(key, value);
 }
```

Reduce Function-

```
function(key, values)
 { counterVal = { count:0};
for (var id = 0; id < values.length; id++)
 { counterVal.count  += values[id].count ;
 }
return counterVal;  }
```

Output-

4] Number times movie tagged (Tags Collection)

Map function – function()

```
 { var value = { count:1};
 var key = this.movieid;
emit(key, value); }
```

Reduce Function- function(key, values)

```
 {  counterVal = { count:0};
 for (var id = 0; id < values.length; id++)
{ counterVal.count  += values[id].count ; }
return counterVal; }
```

Output-

Command Prompt - mongo

```
{ "_id" : "1008", "value" : { "count" : 2 } }
{ "_id" : "1009", "value" : { "count" : 5 } }
{ "_id" : "101", "value" : { "count" : 29 } }
{ "_id" : "1010", "value" : { "count" : 16 } }
{ "_id" : "1011", "value" : { "count" : 2 } }
{ "_id" : "1012", "value" : { "count" : 19 } }
{ "_id" : "1013", "value" : { "count" : 13 } }
{ "_id" : "1014", "value" : { "count" : 10 } }
{ "_id" : "1015", "value" : { "count" : 1 } }
{ "_id" : "1016", "value" : { "count" : 1 } }
{ "_id" : "1017", "value" : { "count" : 14 } }
{ "_id" : "1018", "value" : { "count" : 2 } }
Type "it" for more
> it
{ "_id" : "1019", "value" : { "count" : 6 } }
{ "_id" : "1020", "value" : { "count" : 15 } }
{ "_id" : "1021", "value" : { "count" : 3 } }
{ "_id" : "1022", "value" : { "count" : 24 } }
{ "_id" : "1023", "value" : { "count" : 6 } }
{ "_id" : "1024", "value" : { "count" : 5 } }
{ "_id" : "1025", "value" : { "count" : 31 } }
{ "_id" : "1026", "value" : { "count" : 1 } }
{ "_id" : "1027", "value" : { "count" : 36 } }
{ "_id" : "1028", "value" : { "count" : 35 } }
{ "_id" : "1029", "value" : { "count" : 28 } }
{ "_id" : "103", "value" : { "count" : 1 } }
{ "_id" : "1030", "value" : { "count" : 17 } }
{ "_id" : "1031", "value" : { "count" : 7 } }
{ "_id" : "1032", "value" : { "count" : 39 } }
{ "_id" : "1033", "value" : { "count" : 10 } }
{ "_id" : "1034", "value" : { "count" : 2 } }
{ "_id" : "1035", "value" : { "count" : 74 } }
{ "_id" : "1036", "value" : { "count" : 57 } }
{ "_id" : "1037", "value" : { "count" : 38 } }
Type "it" for more
>
```