INFO 7250 : ENGINEERING OF BIG DATA SYSTEMS

NAME : KAUSHAL CHAUDHARY
NUID : 001886763

ASSIGNMENT 1

PART 2 :

- Application of NoSQL Database in Web Crawling

  Summary:
  In the paper, "Application of NoSQL Database in Web Crawling", the authors talks about the applications of NoSQL databases in Web Crawling and tells us about its advantages over traditional relational databases. They starts by explaining a Web crawler and gives a brief introduction about Meteorological BBS Information collection system. They further explain in dept about the format of the data they would work on along with its characteristics which results into comparison between the NoSQL and SQL databases. They finally conclude the paper by listing advantages of NoSQL over SQL databases schemas emphasizing the performance and scalability and its use in data intensive applications.

- Comparing NoSQL MongoDB to an SQL

  Summary:
  Using the modest-sized structured database, the author talks about the quantitative and qualitative performance comparison between NoSQL MongoDB and RDBMS SQL Server. The performance parameters that were considered are insert speed, update speed and select operation speed. The authors tried to provide distinction about current practices and previously achieved results using this use case. With the help of several papers, the authors explain about their experimental setup along with the test cases and how the entire process would run to get comparative test results which we see in a in depth using graphical figures. Conclusively, the performances were measured in milliseconds which states that MongoDB performs well on the complex queries except for aggregate functions. Authors also puts emphasis on the use of MapReduce for aggregate functions as it slows down the performance.

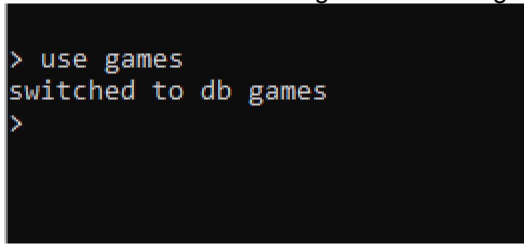
- Data Aggregation System

  Summary:
  The author talks about the implementation of a Data Aggregation System (DAS) used on CERN Large Hadron Collider with the help of Compact Muon Solenoid experiment. DAS was originated with the need to have a single interface that will query multiple heterogenous data sources which is used in the experiment without having common data structure or on API. The paper explains the architecture which consists of a web server, cache server, analytic server, data services and MongoDB at its core. Owing to the independent component structure, the paper lists scalability as one of the advantages of the DAS. In addition to this, the paper also elaborates on how MongoDB shard were used as raw to merge caches for DAS queries handling different data and solving different use cases. Finally, after looking at the benchmark results the paper states that MongoDB works better than DAS in terms of time to fetch randomly selected document.

NAME : KAUSHAL CHAUDHARY
NUID : 001886763

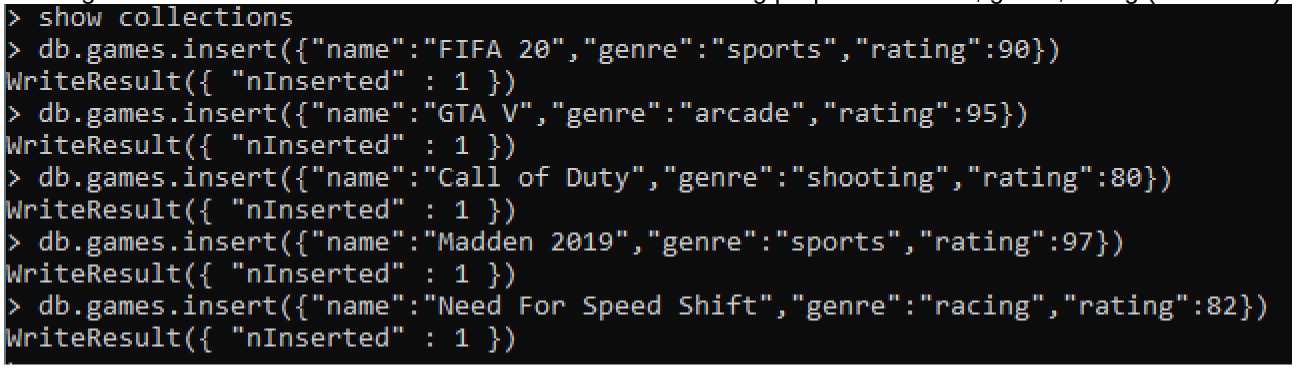
PART 3 :

1. Create a collection called ‘games’. We’re going to put some games in it.



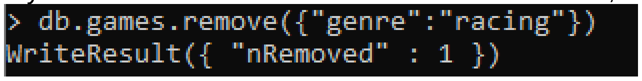
```
> use games
switched to db games
>
```

2. Add 5 games to the database. Give each document the following properties: name, genre, rating (out of 100)



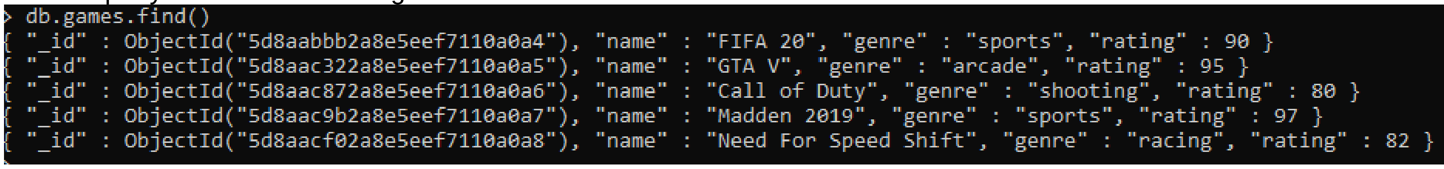
```
> show collections
> db.games.insert({"name":"FIFA 20","genre":"sports","rating":90})
WriteResult({ "nInserted" : 1 })
> db.games.insert({"name":"GTA V","genre":"arcade","rating":95})
WriteResult({ "nInserted" : 1 })
> db.games.insert({"name":"Call of Duty","genre":"shooting","rating":80})
WriteResult({ "nInserted" : 1 })
> db.games.insert({"name":"Madden 2019","genre":"sports","rating":97})
WriteResult({ "nInserted" : 1 })
> db.games.insert({"name":"Need For Speed Shift","genre":"racing","rating":82})
WriteResult({ "nInserted" : 1 })
```

3. If you make some mistakes and want to clean it out, use remove() on your collection.



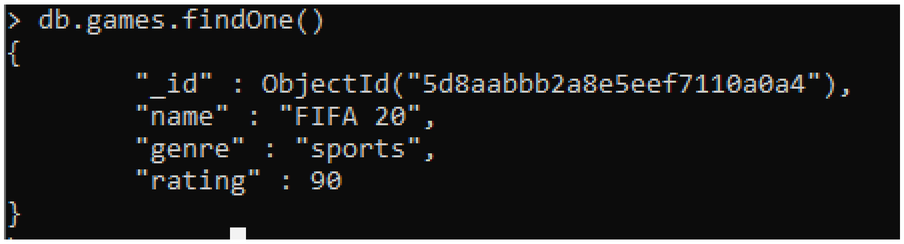
```
> db.games.remove({"genre":"racing"})
WriteResult({ "nRemoved" : 1 })
```

4. Write a query that returns all the games.



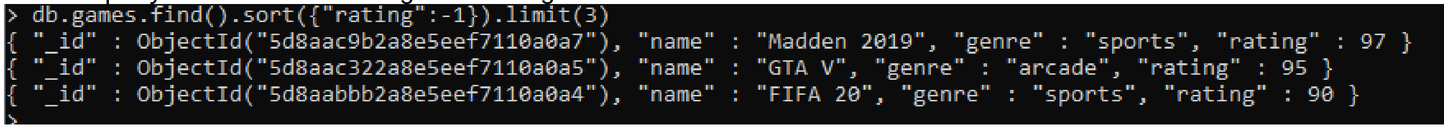
```
> db.games.find()
{ "_id" : ObjectId("5d8aabbb2a8e5eef7110a0a4"), "name" : "FIFA 20", "genre" : "sports", "rating" : 90 }
{ "_id" : ObjectId("5d8aac322a8e5eef7110a0a5"), "name" : "GTA V", "genre" : "arcade", "rating" : 95 }
{ "_id" : ObjectId("5d8aac872a8e5eef7110a0a6"), "name" : "Call of Duty", "genre" : "shooting", "rating" : 80 }
{ "_id" : ObjectId("5d8aac9b2a8e5eef7110a0a7"), "name" : "Madden 2019", "genre" : "sports", "rating" : 97 }
{ "_id" : ObjectId("5d8aacf02a8e5eef7110a0a8"), "name" : "Need For Speed Shift", "genre" : "racing", "rating" : 82 }
```

5. Write a query to find one of your games by name without using limit().Use the findOne method.
   Look how much nicer it’s formatted!



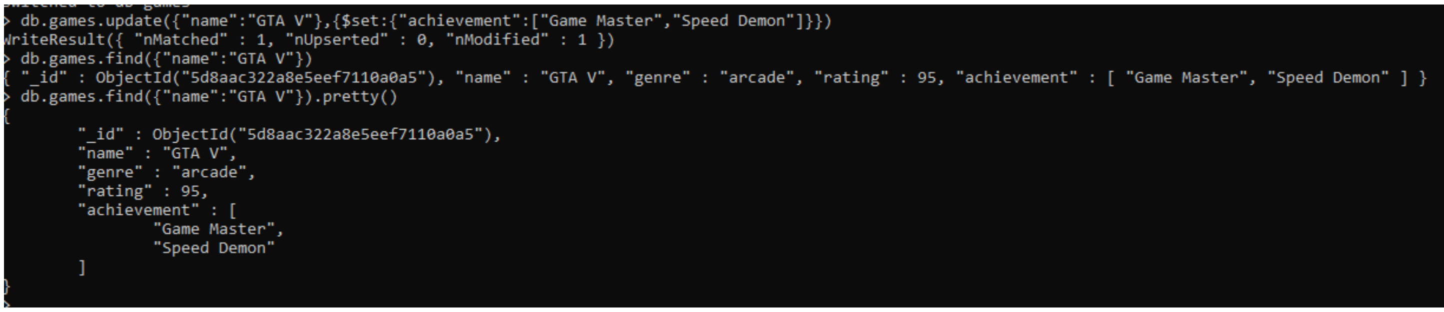
```
> db.games.findOne()
{
        "_id" : ObjectId("5d8aabbb2a8e5eef7110a0a4"),
        "name" : "FIFA 20",
        "genre" : "sports",
        "rating" : 90
}
```

6. Write a query that returns the 3 highest rated games.



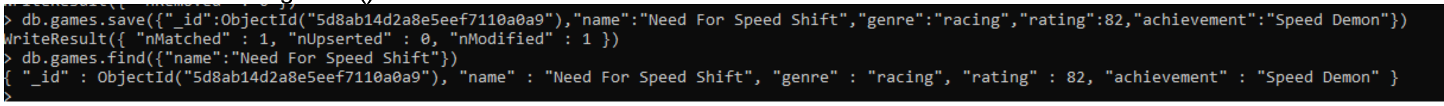
```
> db.games.find().sort({"rating":-1}).limit(3)
{ "_id" : ObjectId("5d8aac9b2a8e5eef7110a0a7"), "name" : "Madden 2019", "genre" : "sports", "rating" : 97 }
{ "_id" : ObjectId("5d8aac322a8e5eef7110a0a5"), "name" : "GTA V", "genre" : "arcade", "rating" : 95 }
{ "_id" : ObjectId("5d8aabbb2a8e5eef7110a0a4"), "name" : "FIFA 20", "genre" : "sports", "rating" : 90 }
```

7. Update your two favorite games to have two achievements called ‘Game Master’ and ‘Speed Demon’, each under a single key.Show two ways to do this.
   Do the first using update()



```
> db.games.update({"name":"GTA V"},{$set:{"achievement":["Game Master","Speed Demon"]}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.games.find({"name":"GTA V"})
{ "_id" : ObjectId("5d8aac322a8e5eef7110a0a5"), "name" : "GTA V", "genre" : "arcade", "rating" : 95, "achievement" : [ "Game Master", "Speed Demon" ] }
> db.games.find({"name":"GTA V"}).pretty()
{
        "_id" : ObjectId("5d8aac322a8e5eef7110a0a5"),
        "name" : "GTA V",
        "genre" : "arcade",
        "rating" : 95,
        "achievement" : [
                "Game Master",
                "Speed Demon"
        ]
}
```

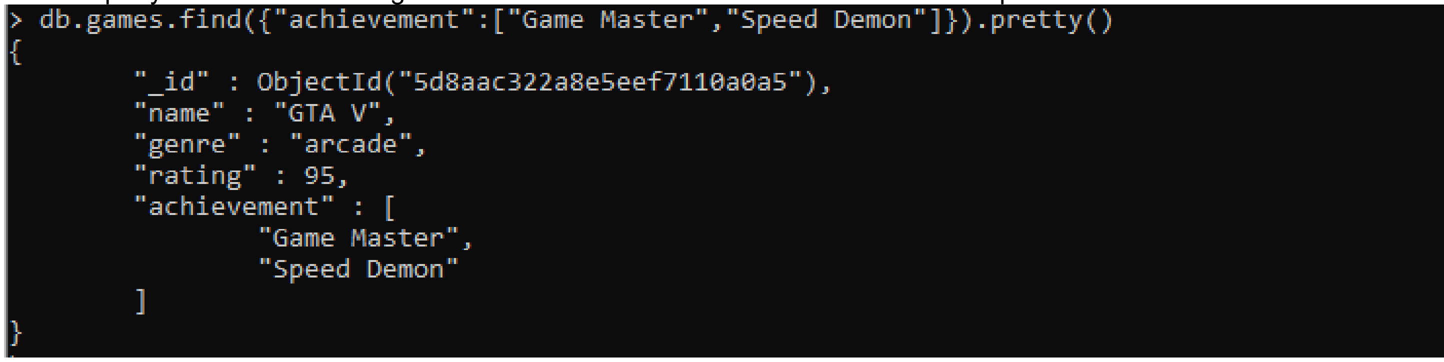
and do the second using save().



```
> db.games.save({"_id":ObjectId("5d8ab14d2a8e5eef7110a0a9"),"name":"Need For Speed Shift","genre":"racing","rating":82,"achievement":"Speed Demon"})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.games.find({"name":"Need For Speed Shift"})
{ "_id" : ObjectId("5d8ab14d2a8e5eef7110a0a9"), "name" : "Need For Speed Shift", "genre" : "racing", "rating" : 82, "achievement" : "Speed Demon" }
```

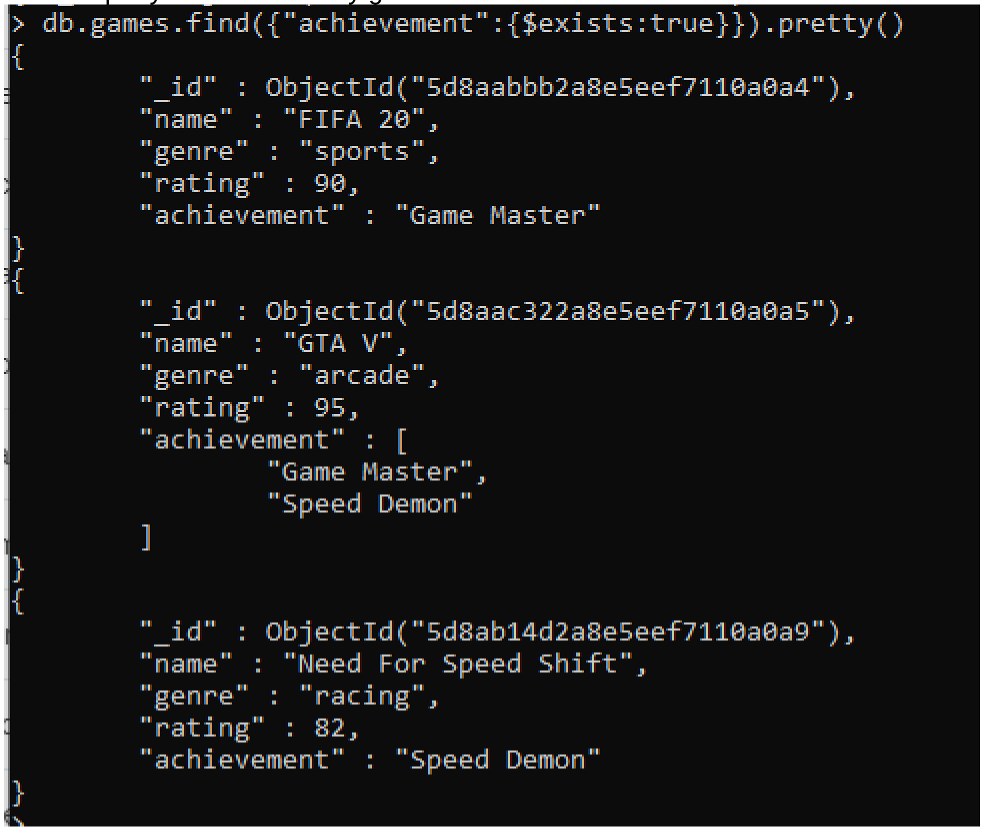
Hint: for save, you might want to query the object and store it in a variable first.

8. Write a query that returns all the games that have both the 'Game Master' and the 'Speed Demon' achievements.



```
> db.games.find({"achievement":["Game Master","Speed Demon"]}).pretty()
{
        "_id" : ObjectId("5d8aac322a8e5eef7110a0a5"),
        "name" : "GTA V",
        "genre" : "arcade",
        "rating" : 95,
        "achievement" : [
                "Game Master",
                "Speed Demon"
        ]
}
```

9. Write a query that returns only games that have achievements.

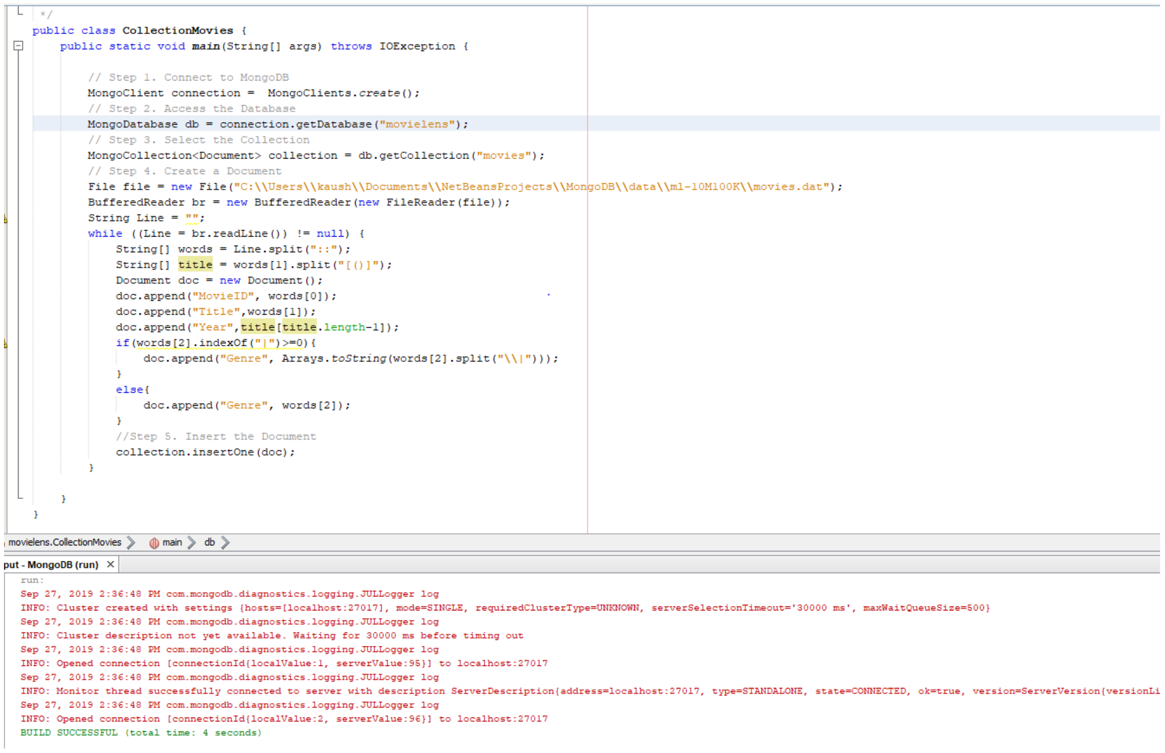


```
> db.games.find({"achievement":{$exists:true}}).pretty()
{
        "_id" : ObjectId("5d8aabbb2a8e5eef7110a0a4"),
        "name" : "FIFA 20",
        "genre" : "sports",
        "rating" : 90,
        "achievement" : "Game Master"
}
{
        "_id" : ObjectId("5d8aac322a8e5eef7110a0a5"),
        "name" : "GTA V",
        "genre" : "arcade",
        "rating" : 95,
        "achievement" : [
                "Game Master",
                "Speed Demon"
        ]
}
{
        "_id" : ObjectId("5d8ab14d2a8e5eef7110a0a9"),
        "name" : "Need For Speed Shift",
        "genre" : "racing",
        "rating" : 82,
        "achievement" : "Speed Demon"
}
```

NAME : KAUSHAL CHAUDHARY
NUID : 001886763

PART 4 :

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).
Collection Movies

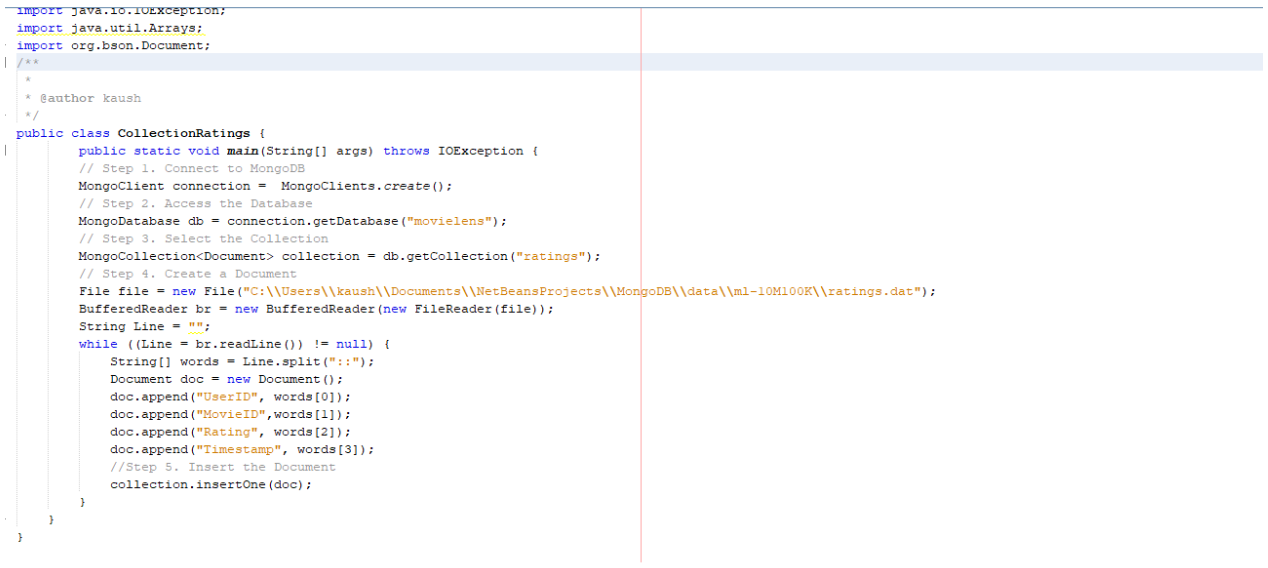


```
 */
public class CollectionMovies {
    public static void main(String[] args) throws IOException {

        // Step 1. Connect to MongoDB
        MongoClient connection =  MongoClients.create();
        // Step 2. Access the Database
        MongoDatabase db = connection.getDatabase("movielens");
        // Step 3. Select the Collection
        MongoCollection<Document> collection = db.getCollection("movies");
        // Step 4. Create a Document
        File file = new File("C:\\Users\\kaush\\Documents\\NetBeansProjects\\MongoDB\\data\\ml-10M100K\\movies.dat");
        BufferedReader br = new BufferedReader(new FileReader(file));
        String Line = "";
        while ((Line = br.readLine()) != null) {
            String[] words = Line.split("::");
            String[] title = words[1].split("[()]");
            Document doc = new Document();
            doc.append("MovieID", words[0]);
            doc.append("Title",words[1]);
            doc.append("Year",title[title.length-1]);
            if(words[2].indexOf("|")>=0){
                doc.append("Genre", Arrays.toString(words[2].split("\\|")));
            }
            else{
                doc.append("Genre", words[2]);
            }
            //Step 5. Insert the Document
            collection.insertOne(doc);
        }

    }
}
```

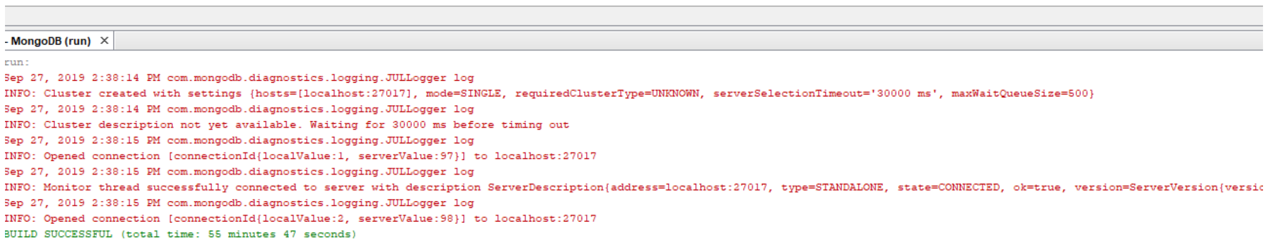
```
run:
Sep 27, 2019 2:36:48 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}
Sep 27, 2019 2:36:48 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
Sep 27, 2019 2:36:48 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:95}] to localhost:27017
Sep 27, 2019 2:36:48 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, version=ServerVersion{versionLi
Sep 27, 2019 2:36:48 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:96}] to localhost:27017
BUILD SUCCESSFUL (total time: 4 seconds)
```

NAME : KAUSHAL CHAUDHARY
NUID : 001886763

CollectionRatings



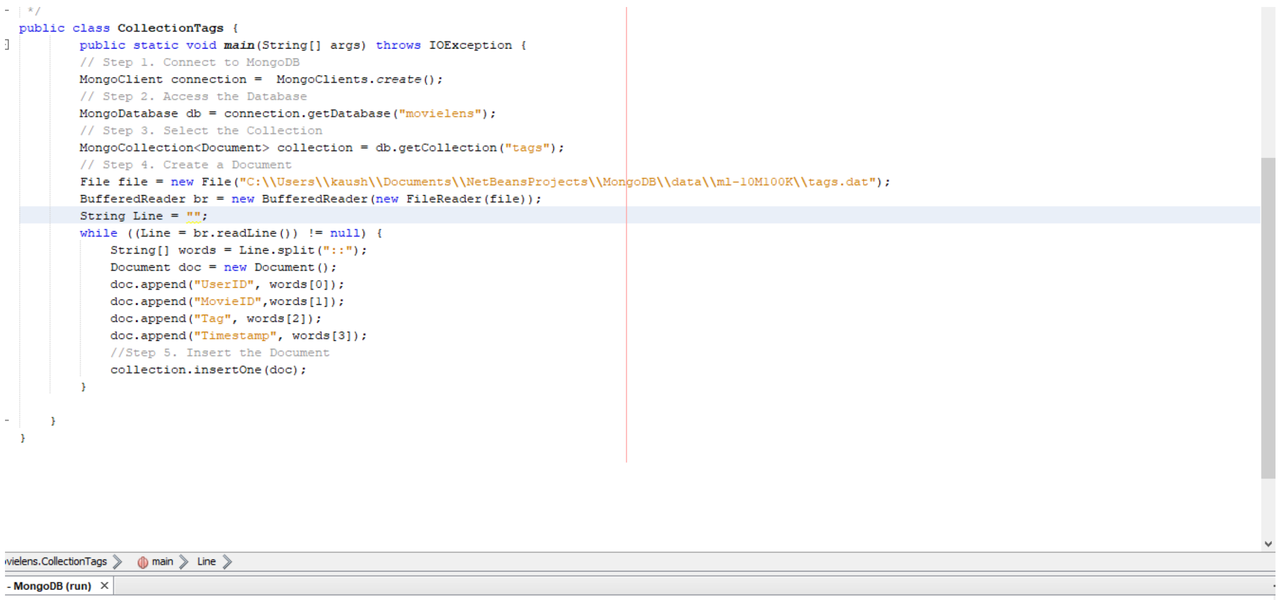
```java
import java.io.IOException;
import java.util.Arrays;
import org.bson.Document;
/**
 *
 * @author kaush
 */
public class CollectionRatings {
        public static void main(String[] args) throws IOException {
        // Step 1. Connect to MongoDB
        MongoClient connection =  MongoClients.create();
        // Step 2. Access the Database
        MongoDatabase db = connection.getDatabase("movielens");
        // Step 3. Select the Collection
        MongoCollection<Document> collection = db.getCollection("ratings");
        // Step 4. Create a Document
        File file = new File("C:\\Users\\kaush\\Documents\\NetBeansProjects\\MongoDB\\data\\ml-10M100K\\ratings.dat");
        BufferedReader br = new BufferedReader(new FileReader(file));
        String Line = "";
        while ((Line = br.readLine()) != null) {
            String[] words = Line.split("::");
            Document doc = new Document();
            doc.append("UserID", words[0]);
            doc.append("MovieID",words[1]);
            doc.append("Rating", words[2]);
            doc.append("Timestamp", words[3]);
            //Step 5. Insert the Document
            collection.insertOne(doc);
        }
    }
}
```



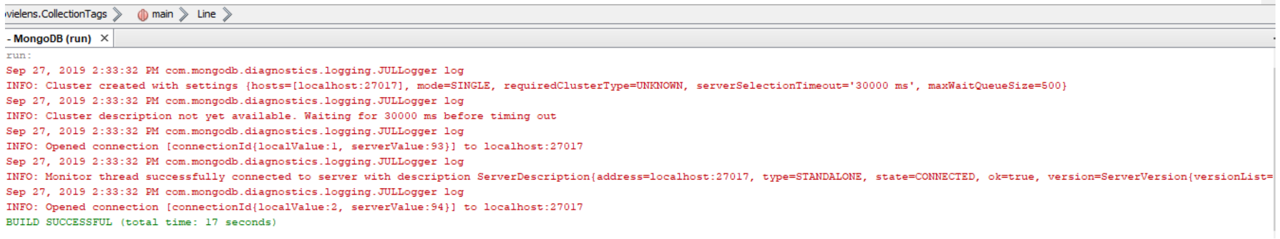
- MongoDB (run)

```
run:
Sep 27, 2019 2:38:14 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}
Sep 27, 2019 2:38:14 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
Sep 27, 2019 2:38:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:97}] to localhost:27017
Sep 27, 2019 2:38:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, version=ServerVersion{versic
Sep 27, 2019 2:38:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:98}] to localhost:27017
BUILD SUCCESSFUL (total time: 55 minutes 47 seconds)
```

CollectionTags



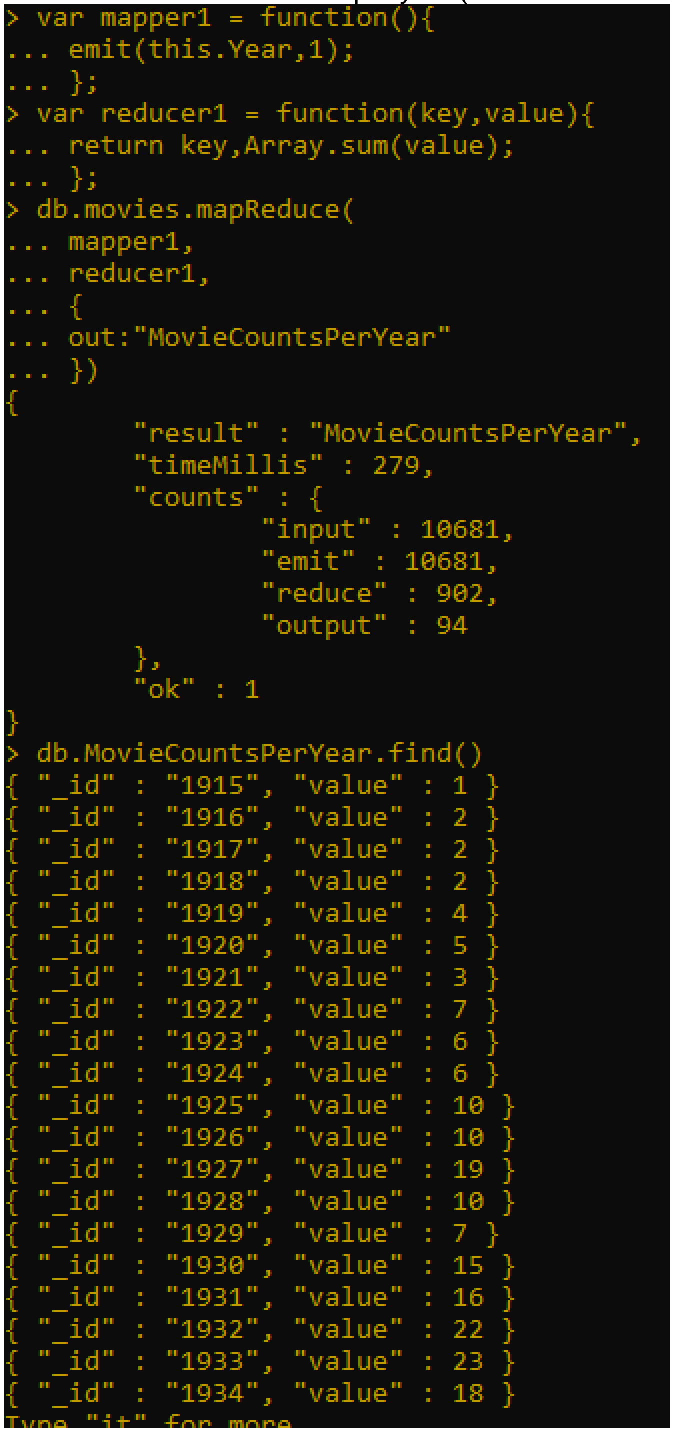
```java
 */
public class CollectionTags {
        public static void main(String[] args) throws IOException {
        // Step 1. Connect to MongoDB
        MongoClient connection =  MongoClients.create();
        // Step 2. Access the Database
        MongoDatabase db = connection.getDatabase("movielens");
        // Step 3. Select the Collection
        MongoCollection<Document> collection = db.getCollection("tags");
        // Step 4. Create a Document
        File file = new File("C:\\Users\\kaush\\Documents\\NetBeansProjects\\MongoDB\\data\\ml-10M100K\\tags.dat");
        BufferedReader br = new BufferedReader(new FileReader(file));
        String Line = "";
        while ((Line = br.readLine()) != null) {
            String[] words = Line.split("::");
            Document doc = new Document();
            doc.append("UserID", words[0]);
            doc.append("MovieID",words[1]);
            doc.append("Tag", words[2]);
            doc.append("Timestamp", words[3]);
            //Step 5. Insert the Document
            collection.insertOne(doc);
        }

    }
}
```



```
run:
Sep 27, 2019 2:33:32 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}
Sep 27, 2019 2:33:32 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
Sep 27, 2019 2:33:32 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:93}] to localhost:27017
Sep 27, 2019 2:33:32 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, version=ServerVersion{versionList=
Sep 27, 2019 2:33:32 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:94}] to localhost:27017
BUILD SUCCESSFUL (total time: 17 seconds)
```

NAME : KAUSHAL CHAUDHARY
NUID : 001886763

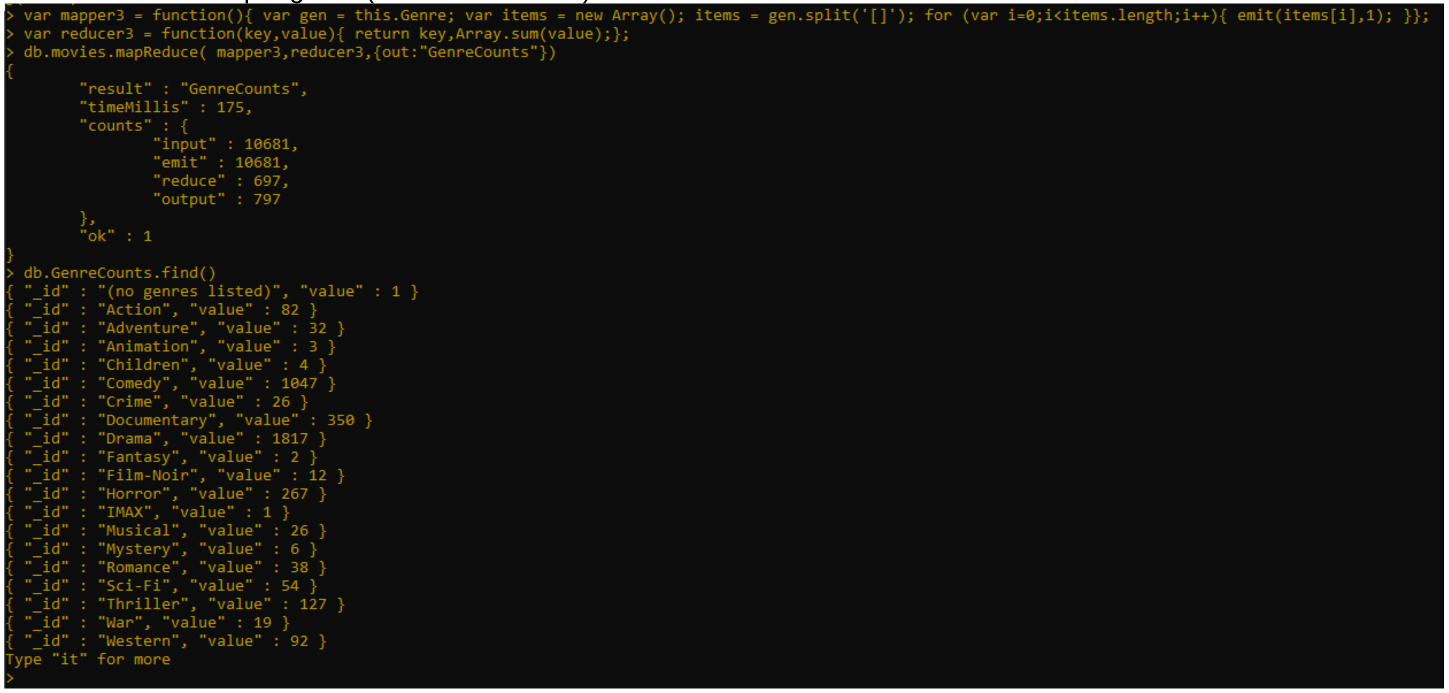
Write a MapReduce to do the followings:
- Number of Movies released per year (Movies Collection)



```
> var mapper1 = function(){
... emit(this.Year,1);
... };
> var reducer1 = function(key,value){
... return key,Array.sum(value);
... };
> db.movies.mapReduce(
... mapper1,
... reducer1,
... {
... out:"MovieCountsPerYear"
... })
{
        "result" : "MovieCountsPerYear",
        "timeMillis" : 279,
        "counts" : {
                "input" : 10681,
                "emit" : 10681,
                "reduce" : 902,
                "output" : 94
        },
        "ok" : 1
}
> db.MovieCountsPerYear.find()
{ "_id" : "1915", "value" : 1 }
{ "_id" : "1916", "value" : 2 }
{ "_id" : "1917", "value" : 2 }
{ "_id" : "1918", "value" : 2 }
{ "_id" : "1919", "value" : 4 }
{ "_id" : "1920", "value" : 5 }
{ "_id" : "1921", "value" : 3 }
{ "_id" : "1922", "value" : 7 }
{ "_id" : "1923", "value" : 6 }
{ "_id" : "1924", "value" : 6 }
{ "_id" : "1925", "value" : 10 }
{ "_id" : "1926", "value" : 10 }
{ "_id" : "1927", "value" : 19 }
{ "_id" : "1928", "value" : 10 }
{ "_id" : "1929", "value" : 7 }
{ "_id" : "1930", "value" : 15 }
{ "_id" : "1931", "value" : 16 }
{ "_id" : "1932", "value" : 22 }
{ "_id" : "1933", "value" : 23 }
{ "_id" : "1934", "value" : 18 }
Type "it" for more
```

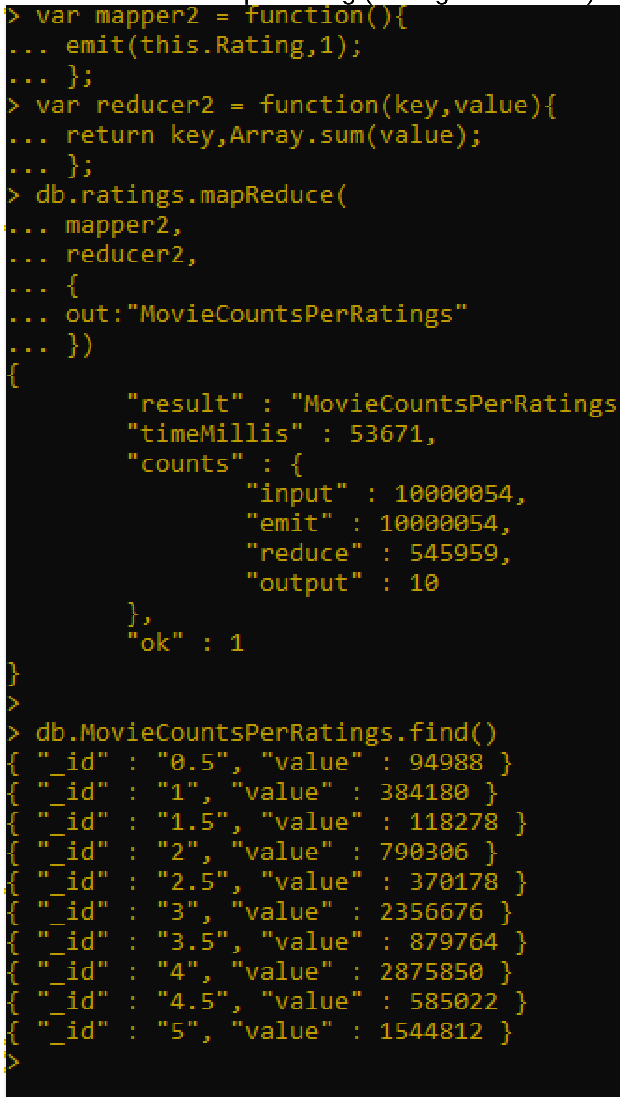
NAME : KAUSHAL CHAUDHARY
NUID : 001886763

- Number of Movies per genre (Movies Collection)



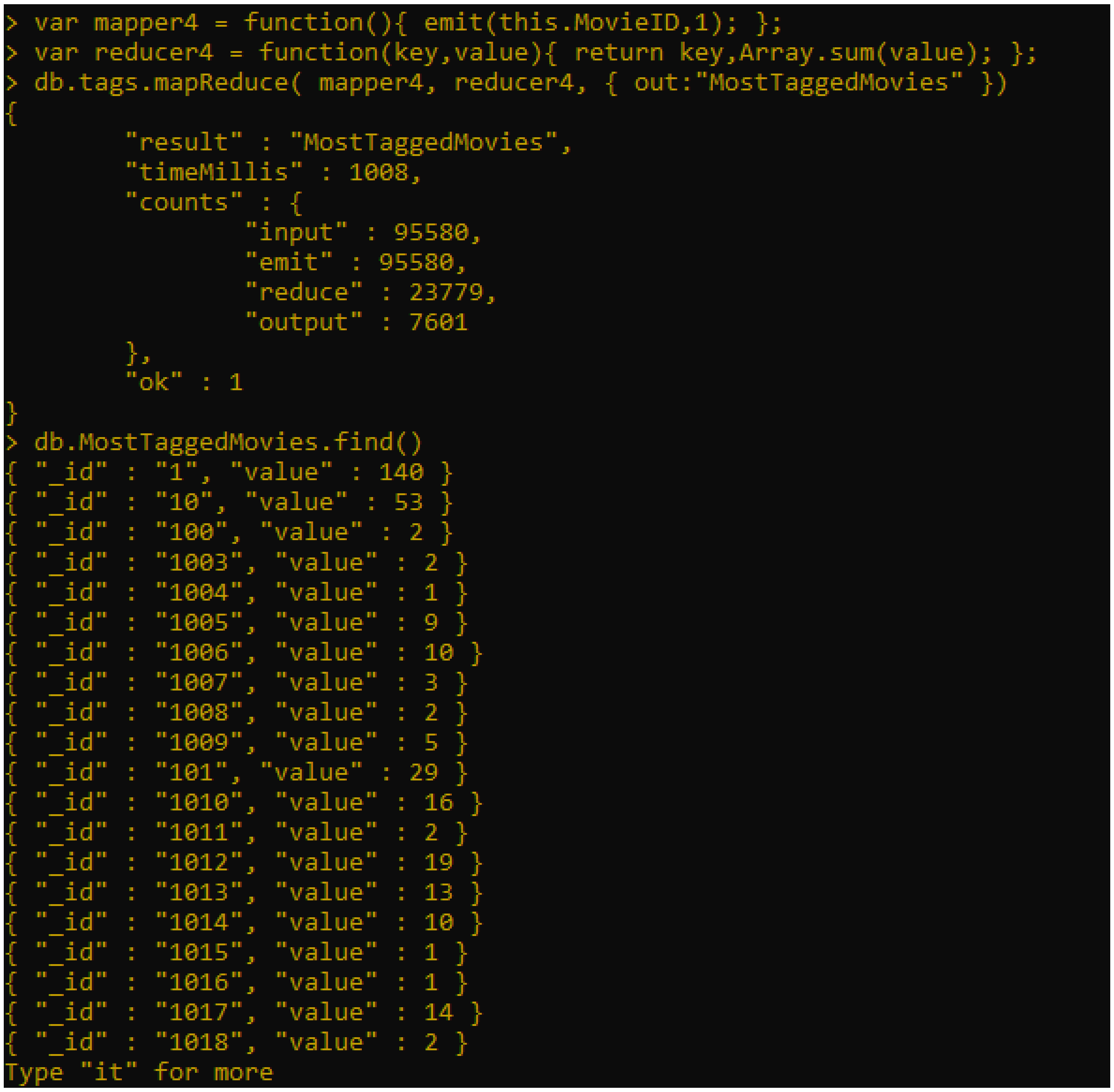
```
> var mapper3 = function(){ var gen = this.Genre; var items = new Array(); items = gen.split('[]'); for (var i=0;i<items.length;i++){ emit(items[i],1); }};
> var reducer3 = function(key,value){ return key,Array.sum(value);};
> db.movies.mapReduce( mapper3,reducer3,{out:"GenreCounts"})
{
        "result" : "GenreCounts",
        "timeMillis" : 175,
        "counts" : {
                "input" : 10681,
                "emit" : 10681,
                "reduce" : 697,
                "output" : 797
        },
        "ok" : 1
}
> db.GenreCounts.find()
{ "_id" : "(no genres listed)", "value" : 1 }
{ "_id" : "Action", "value" : 82 }
{ "_id" : "Adventure", "value" : 32 }
{ "_id" : "Animation", "value" : 3 }
{ "_id" : "Children", "value" : 4 }
{ "_id" : "Comedy", "value" : 1047 }
{ "_id" : "Crime", "value" : 26 }
{ "_id" : "Documentary", "value" : 350 }
{ "_id" : "Drama", "value" : 1817 }
{ "_id" : "Fantasy", "value" : 2 }
{ "_id" : "Film-Noir", "value" : 12 }
{ "_id" : "Horror", "value" : 267 }
{ "_id" : "IMAX", "value" : 1 }
{ "_id" : "Musical", "value" : 26 }
{ "_id" : "Mystery", "value" : 6 }
{ "_id" : "Romance", "value" : 38 }
{ "_id" : "Sci-Fi", "value" : 54 }
{ "_id" : "Thriller", "value" : 127 }
{ "_id" : "War", "value" : 19 }
{ "_id" : "Western", "value" : 92 }
Type "it" for more
>
```

- Number of Movies per rating (Ratings Collection)



```
> var mapper2 = function(){
... emit(this.Rating,1);
... };
> var reducer2 = function(key,value){
... return key,Array.sum(value);
... };
> db.ratings.mapReduce(
... mapper2,
... reducer2,
... {
... out:"MovieCountsPerRatings"
... })
{
        "result" : "MovieCountsPerRatings
        "timeMillis" : 53671,
        "counts" : {
                "input" : 10000054,
                "emit" : 10000054,
                "reduce" : 545959,
                "output" : 10
        },
        "ok" : 1
}
>
> db.MovieCountsPerRatings.find()
{ "_id" : "0.5", "value" : 94988 }
{ "_id" : "1", "value" : 384180 }
{ "_id" : "1.5", "value" : 118278 }
{ "_id" : "2", "value" : 790306 }
{ "_id" : "2.5", "value" : 370178 }
{ "_id" : "3", "value" : 2356676 }
{ "_id" : "3.5", "value" : 879764 }
{ "_id" : "4", "value" : 2875850 }
{ "_id" : "4.5", "value" : 585022 }
{ "_id" : "5", "value" : 1544812 }
>
```

- Number of times each movie was tagged (Tags Collection)

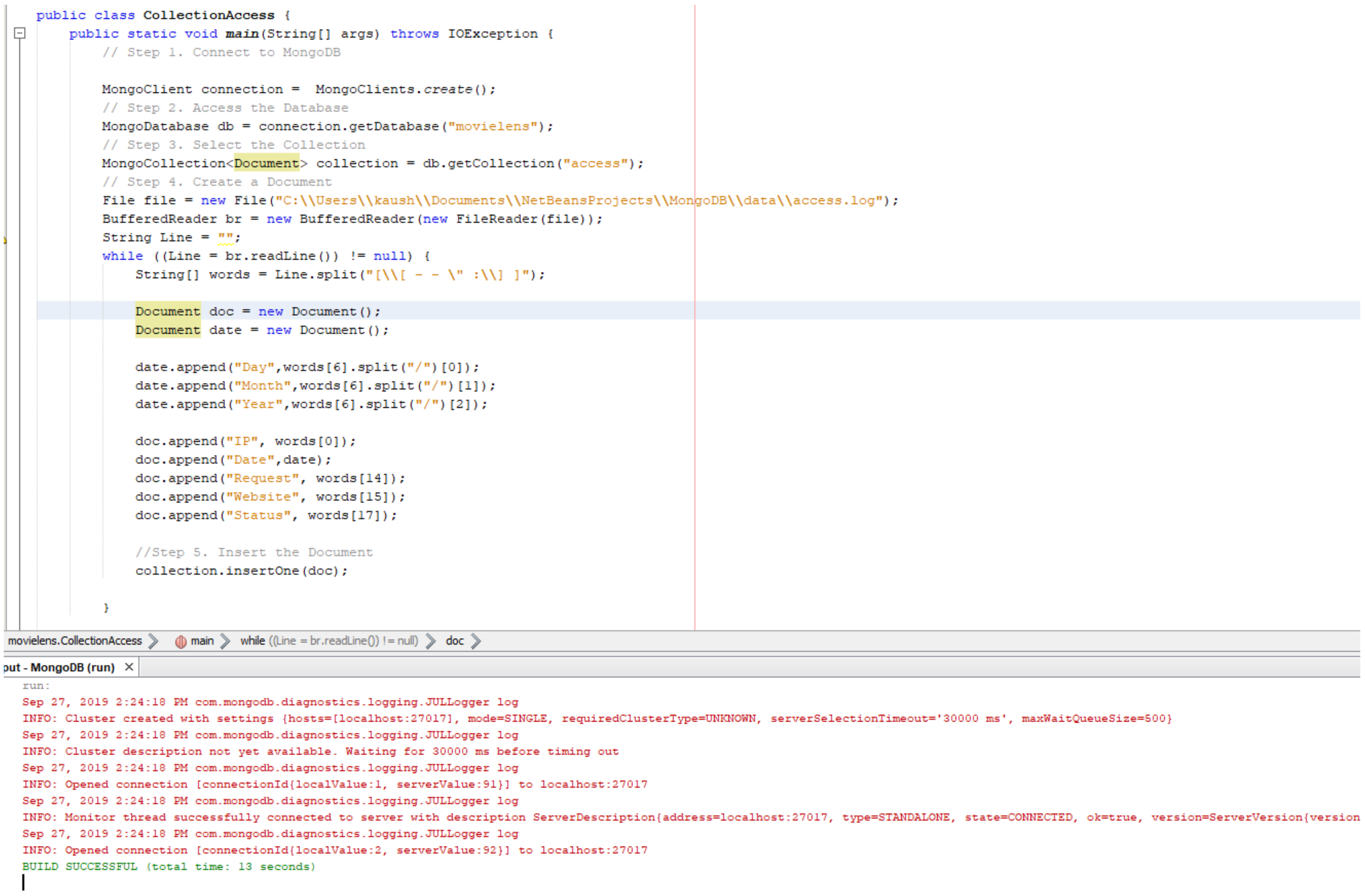


```
> var mapper4 = function(){ emit(this.MovieID,1); };
> var reducer4 = function(key,value){ return key,Array.sum(value); };
> db.tags.mapReduce( mapper4, reducer4, { out:"MostTaggedMovies" })
{
        "result" : "MostTaggedMovies",
        "timeMillis" : 1008,
        "counts" : {
                "input" : 95580,
                "emit" : 95580,
                "reduce" : 23779,
                "output" : 7601
        },
        "ok" : 1
}
> db.MostTaggedMovies.find()
{ "_id" : "1", "value" : 140 }
{ "_id" : "10", "value" : 53 }
{ "_id" : "100", "value" : 2 }
{ "_id" : "1003", "value" : 2 }
{ "_id" : "1004", "value" : 1 }
{ "_id" : "1005", "value" : 9 }
{ "_id" : "1006", "value" : 10 }
{ "_id" : "1007", "value" : 3 }
{ "_id" : "1008", "value" : 2 }
{ "_id" : "1009", "value" : 5 }
{ "_id" : "101", "value" : 29 }
{ "_id" : "1010", "value" : 16 }
{ "_id" : "1011", "value" : 2 }
{ "_id" : "1012", "value" : 19 }
{ "_id" : "1013", "value" : 13 }
{ "_id" : "1014", "value" : 10 }
{ "_id" : "1015", "value" : 1 }
{ "_id" : "1016", "value" : 1 }
{ "_id" : "1017", "value" : 14 }
{ "_id" : "1018", "value" : 2 }
Type "it" for more
```

NAME : KAUSHAL CHAUDHARY
NUID : 001886763

PART 5 :

Write a Java (could be a console app - will only run once to import the data into MongoDB) program to read the **access.log** file (attached), and insert
into access collection.



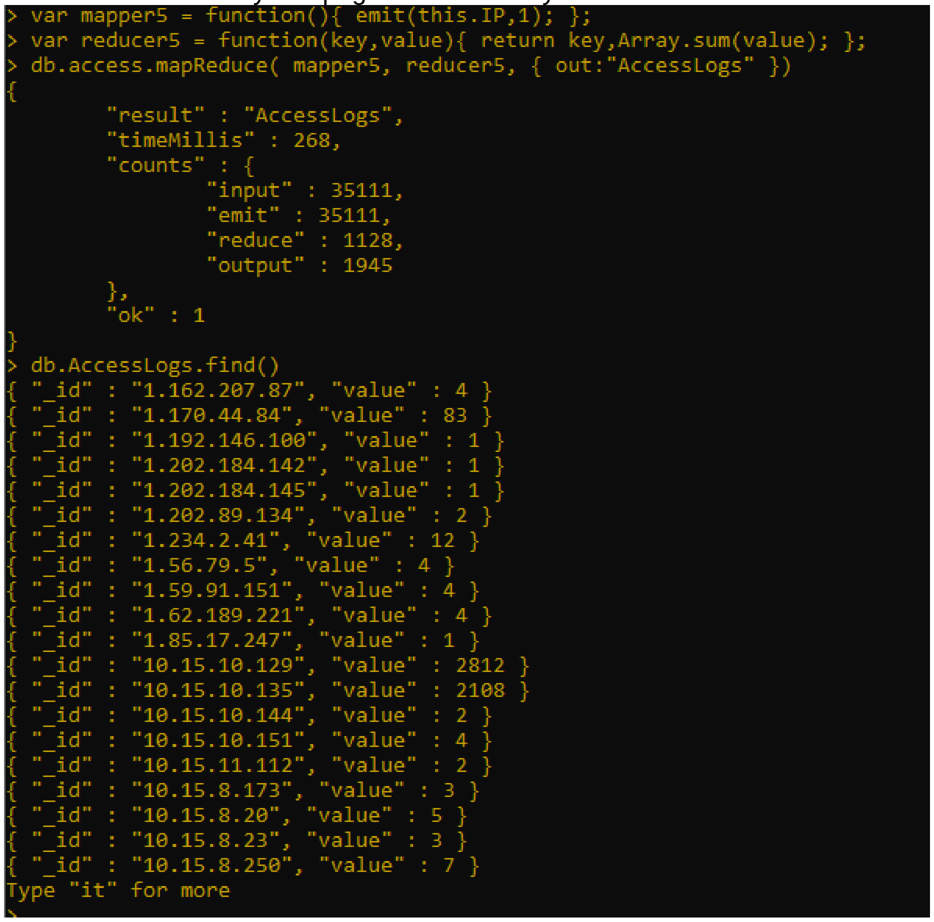
```java
public class CollectionAccess {
    public static void main(String[] args) throws IOException {
        // Step 1. Connect to MongoDB

        MongoClient connection =  MongoClients.create();
        // Step 2. Access the Database
        MongoDatabase db = connection.getDatabase("movielens");
        // Step 3. Select the Collection
        MongoCollection<Document> collection = db.getCollection("access");
        // Step 4. Create a Document
        File file = new File("C:\\Users\\kaush\\Documents\\NetBeansProjects\\MongoDB\\data\\access.log");
        BufferedReader br = new BufferedReader(new FileReader(file));
        String Line = "";
        while ((Line = br.readLine()) != null) {
            String[] words = Line.split("[\\[ - - \" :\\] ]");

            Document doc = new Document();
            Document date = new Document();

            date.append("Day",words[6].split("/")[0]);
            date.append("Month",words[6].split("/")[1]);
            date.append("Year",words[6].split("/")[2]);

            doc.append("IP", words[0]);
            doc.append("Date",date);
            doc.append("Request", words[14]);
            doc.append("Website", words[15]);
            doc.append("Status", words[17]);

            //Step 5. Insert the Document
            collection.insertOne(doc);

        }
```

movielens.CollectionAccess
main
while ((Line = br.readLine()) != null)
doc
put - MongoDB (run)

```
run:
Sep 27, 2019 2:24:18 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}
Sep 27, 2019 2:24:18 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
Sep 27, 2019 2:24:18 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:91}] to localhost:27017
Sep 27, 2019 2:24:18 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, version=ServerVersion{version
Sep 27, 2019 2:24:18 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:92}] to localhost:27017
BUILD SUCCESSFUL (total time: 13 seconds)
```

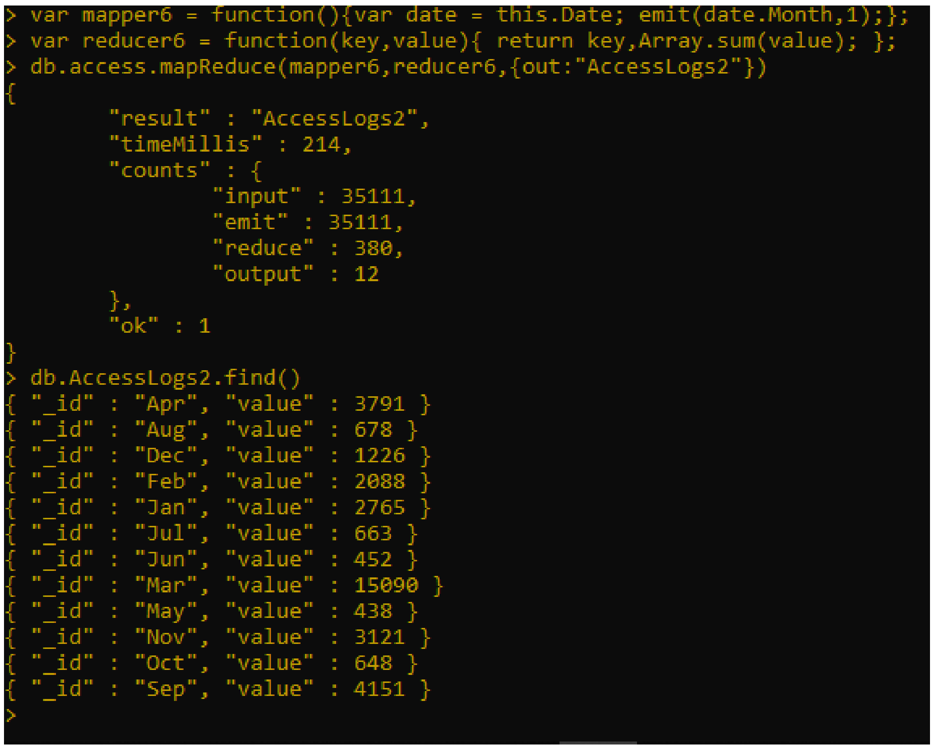
NAME : KAUSHAL CHAUDHARY
NUID : 001886763

Once the data are inserted into MongoDB, do the followings using MapReduce:
- Number of times any webpage was visited by the same IP address.



```
> var mapper5 = function(){ emit(this.IP,1); };
> var reducer5 = function(key,value){ return key,Array.sum(value); };
> db.access.mapReduce( mapper5, reducer5, { out:"AccessLogs" })
{
        "result" : "AccessLogs",
        "timeMillis" : 268,
        "counts" : {
                "input" : 35111,
                "emit" : 35111,
                "reduce" : 1128,
                "output" : 1945
        },
        "ok" : 1
}
> db.AccessLogs.find()
{ "_id" : "1.162.207.87", "value" : 4 }
{ "_id" : "1.170.44.84", "value" : 83 }
{ "_id" : "1.192.146.100", "value" : 1 }
{ "_id" : "1.202.184.142", "value" : 1 }
{ "_id" : "1.202.184.145", "value" : 1 }
{ "_id" : "1.202.89.134", "value" : 2 }
{ "_id" : "1.234.2.41", "value" : 12 }
{ "_id" : "1.56.79.5", "value" : 4 }
{ "_id" : "1.59.91.151", "value" : 4 }
{ "_id" : "1.62.189.221", "value" : 4 }
{ "_id" : "1.85.17.247", "value" : 1 }
{ "_id" : "10.15.10.129", "value" : 2812 }
{ "_id" : "10.15.10.135", "value" : 2108 }
{ "_id" : "10.15.10.144", "value" : 2 }
{ "_id" : "10.15.10.151", "value" : 4 }
{ "_id" : "10.15.11.112", "value" : 2 }
{ "_id" : "10.15.8.173", "value" : 3 }
{ "_id" : "10.15.8.20", "value" : 5 }
{ "_id" : "10.15.8.23", "value" : 3 }
{ "_id" : "10.15.8.250", "value" : 7 }
Type "it" for more
```

- Number of times any webpage was visited each month

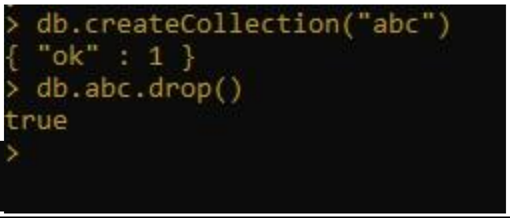


```
> var mapper6 = function(){var date = this.Date; emit(date.Month,1);};
> var reducer6 = function(key,value){ return key,Array.sum(value); };
> db.access.mapReduce(mapper6,reducer6,{out:"AccessLogs2"})
{
        "result" : "AccessLogs2",
        "timeMillis" : 214,
        "counts" : {
                "input" : 35111,
                "emit" : 35111,
                "reduce" : 380,
                "output" : 12
        },
        "ok" : 1
}
> db.AccessLogs2.find()
{ "_id" : "Apr", "value" : 3791 }
{ "_id" : "Aug", "value" : 678 }
{ "_id" : "Dec", "value" : 1226 }
{ "_id" : "Feb", "value" : 2088 }
{ "_id" : "Jan", "value" : 2765 }
{ "_id" : "Jul", "value" : 663 }
{ "_id" : "Jun", "value" : 452 }
{ "_id" : "Mar", "value" : 15090 }
{ "_id" : "May", "value" : 438 }
{ "_id" : "Nov", "value" : 3121 }
{ "_id" : "Oct", "value" : 648 }
{ "_id" : "Sep", "value" : 4151 }
>
```

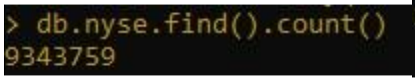
NAME : KAUSHAL CHAUDHARY
NUID : 001886763

PART 6 :

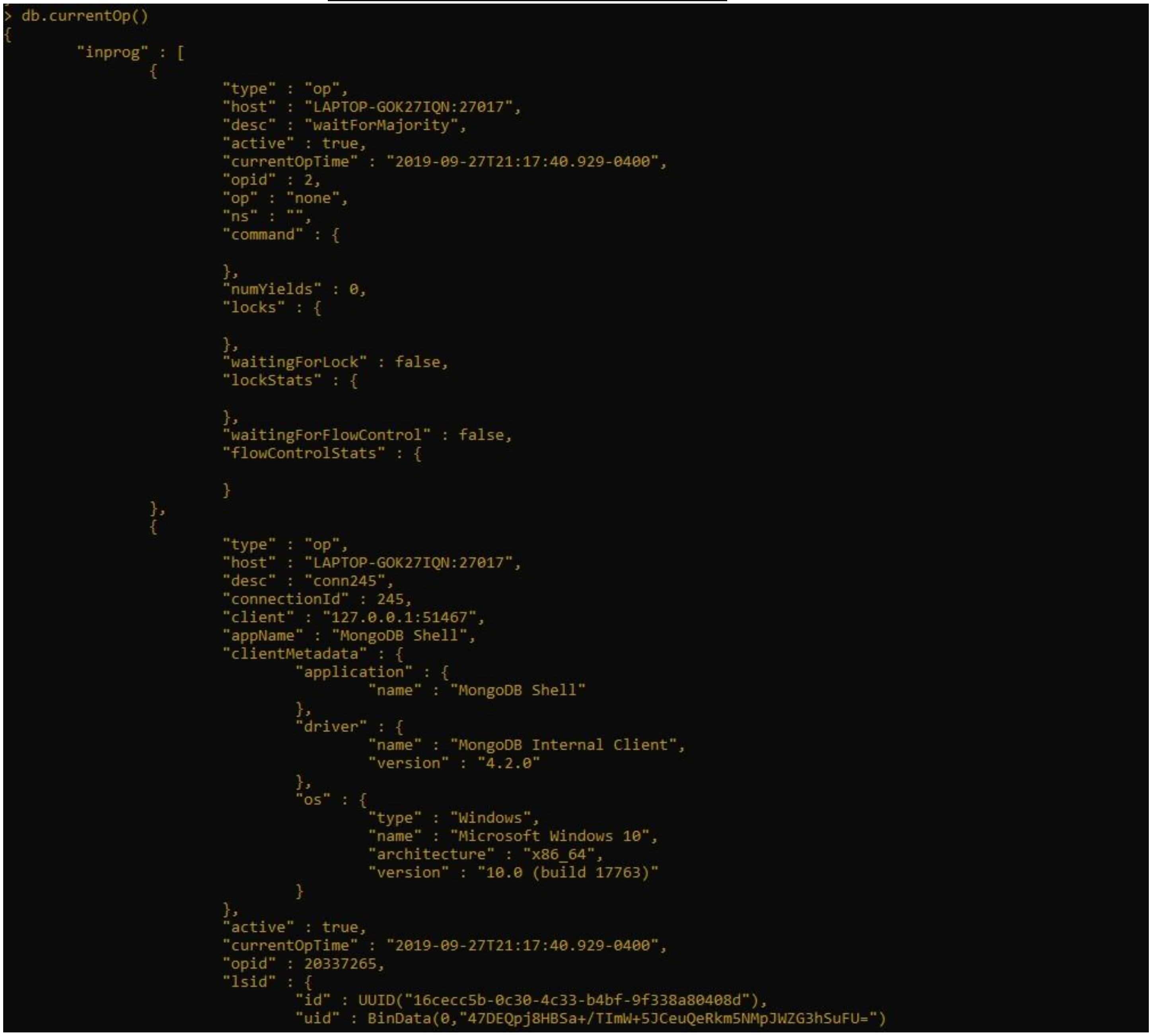
Execute 5 commands of your choice from each of the following groups, and paste the screenshots in a word document.
mongo> help [5 commands]
mongo> db.help() [5 commands]
mongo> db.mycoll.help() [10 commands]



```
> db.createCollection("abc")
{ "ok" : 1 }
> db.abc.drop()
true
>
```

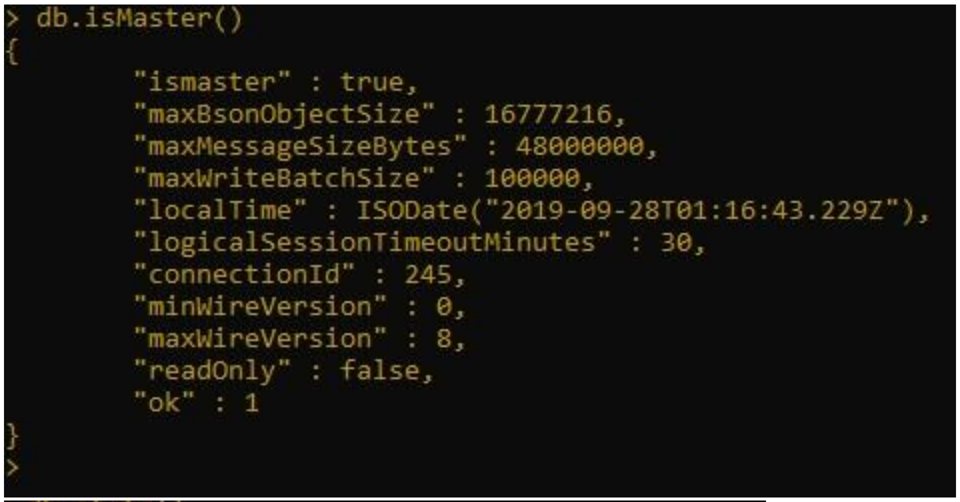


```
> db.nyse.find().count()
9343759
```

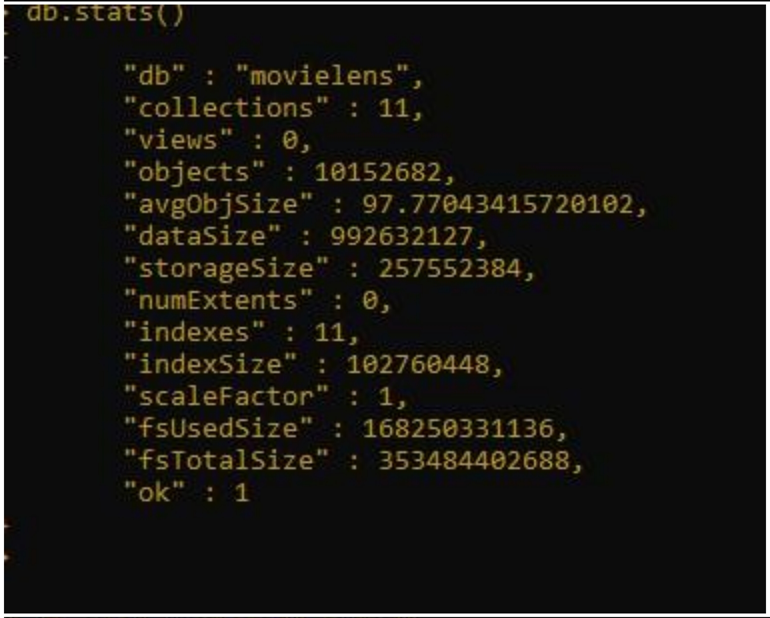


```
> db.currentOp()
{
        "inprog" : [
                {
                        "type" : "op",
                        "host" : "LAPTOP-GOK27IQN:27017",
                        "desc" : "waitForMajority",
                        "active" : true,
                        "currentOpTime" : "2019-09-27T21:17:40.929-0400",
                        "opid" : 2,
                        "op" : "none",
                        "ns" : "",
                        "command" : {

                        },
                        "numYields" : 0,
                        "locks" : {

                        },
                        "waitingForLock" : false,
                        "lockStats" : {

                        },
                        "waitingForFlowControl" : false,
                        "flowControlStats" : {

                        }
                },
                {
                        "type" : "op",
                        "host" : "LAPTOP-GOK27IQN:27017",
                        "desc" : "conn245",
                        "connectionId" : 245,
                        "client" : "127.0.0.1:51467",
                        "appName" : "MongoDB Shell",
                        "clientMetadata" : {
                                "application" : {
                                        "name" : "MongoDB Shell"
                                },
                                "driver" : {
                                        "name" : "MongoDB Internal Client",
                                        "version" : "4.2.0"
                                },
                                "os" : {
                                        "type" : "Windows",
                                        "name" : "Microsoft Windows 10",
                                        "architecture" : "x86_64",
                                        "version" : "10.0 (build 17763)"
                                }
                        },
                        "active" : true,
                        "currentOpTime" : "2019-09-27T21:17:40.929-0400",
                        "opid" : 20337265,
                        "lsid" : {
                                "id" : UUID("16cecc5b-0c30-4c33-b4bf-9f338a80408d"),
                                "uid" : BinData(0,"47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=")
```

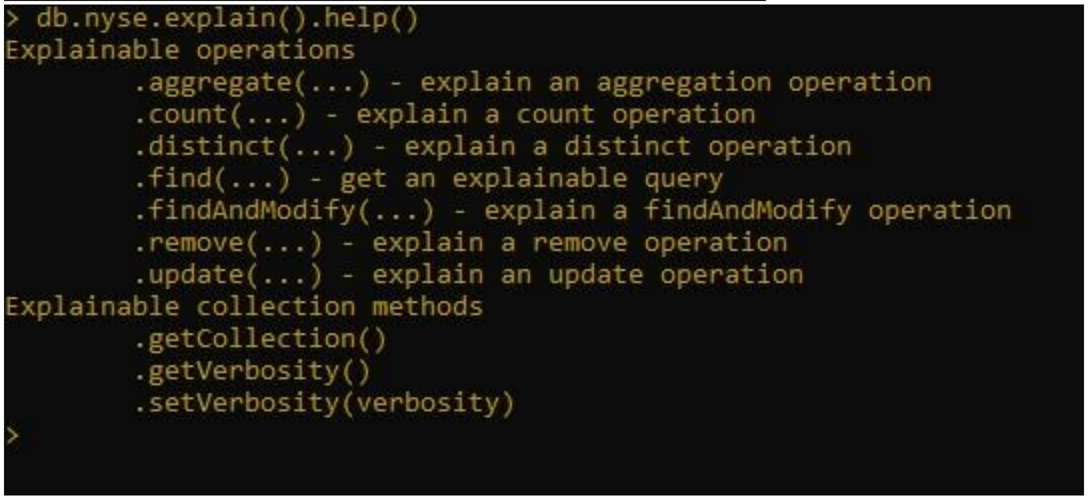
NAME : KAUSHAL CHAUDHARY
NUID : 001886763



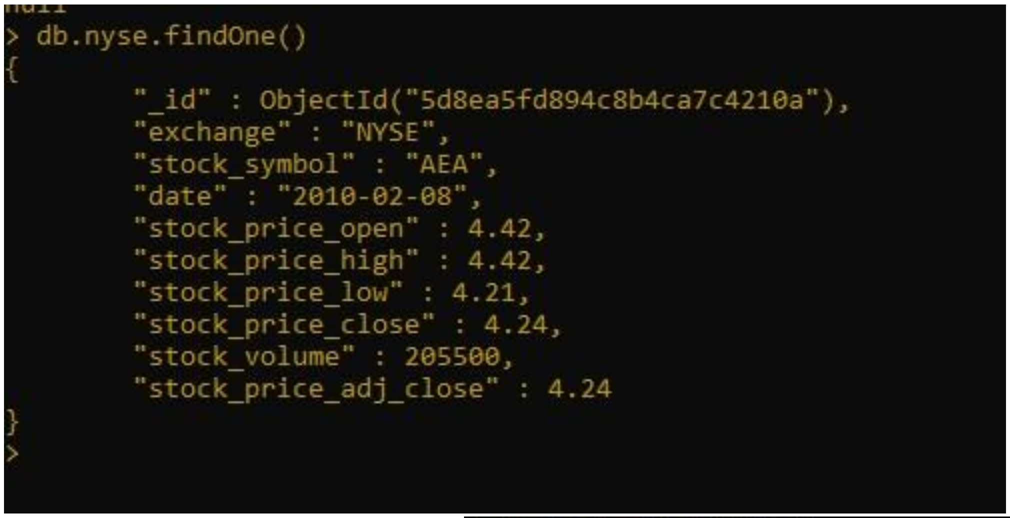
```
> db.isMaster()
{
        "ismaster" : true,
        "maxBsonObjectSize" : 16777216,
        "maxMessageSizeBytes" : 48000000,
        "maxWriteBatchSize" : 100000,
        "localTime" : ISODate("2019-09-28T01:16:43.229Z"),
        "logicalSessionTimeoutMinutes" : 30,
        "connectionId" : 245,
        "minWireVersion" : 0,
        "maxWireVersion" : 8,
        "readOnly" : false,
        "ok" : 1
}
>
```



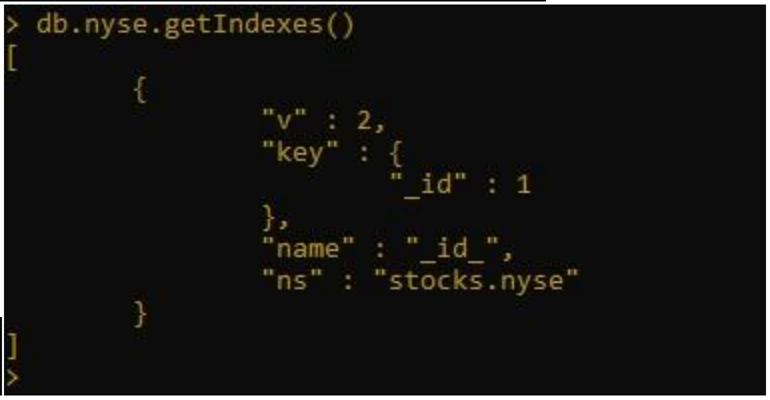
```
 db.stats()

        "db" : "movielens",
        "collections" : 11,
        "views" : 0,
        "objects" : 10152682,
        "avgObjSize" : 97.77043415720102,
        "dataSize" : 992632127,
        "storageSize" : 257552384,
        "numExtents" : 0,
        "indexes" : 11,
        "indexSize" : 102760448,
        "scaleFactor" : 1,
        "fsUsedSize" : 168250331136,
        "fsTotalSize" : 353484402688,
        "ok" : 1
```



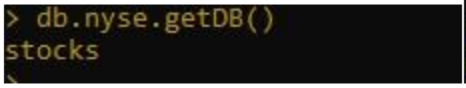
```
> db.nyse.explain().help()
Explainable operations
        .aggregate(...) - explain an aggregation operation
        .count(...) - explain a count operation
        .distinct(...) - explain a distinct operation
        .find(...) - get an explainable query
        .findAndModify(...) - explain a findAndModify operation
        .remove(...) - explain a remove operation
        .update(...) - explain an update operation
Explainable collection methods
        .getCollection()
        .getVerbosity()
        .setVerbosity(verbosity)
>
```



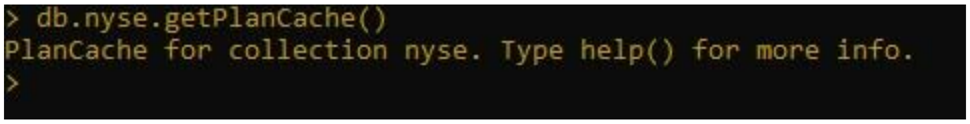
```
null
> db.nyse.findOne()
{
        "_id" : ObjectId("5d8ea5fd894c8b4ca7c4210a"),
        "exchange" : "NYSE",
        "stock_symbol" : "AEA",
        "date" : "2010-02-08",
        "stock_price_open" : 4.42,
        "stock_price_high" : 4.42,
        "stock_price_low" : 4.21,
        "stock_price_close" : 4.24,
        "stock_volume" : 205500,
        "stock_price_adj_close" : 4.24
}
>
```



```
> db.nyse.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "stocks.nyse"
        }
]
>
```



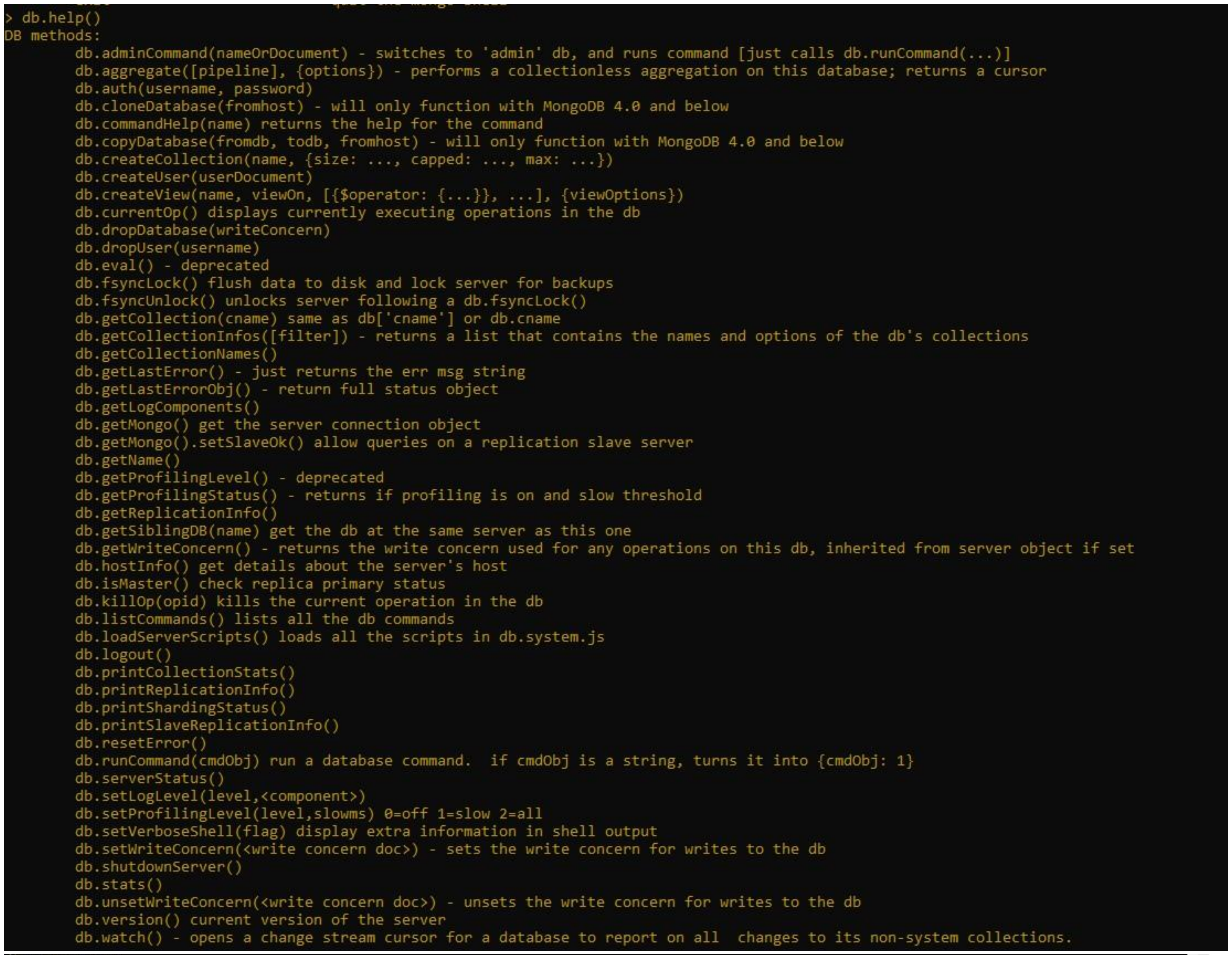
```
> db.nyse.getDB()
stocks
```



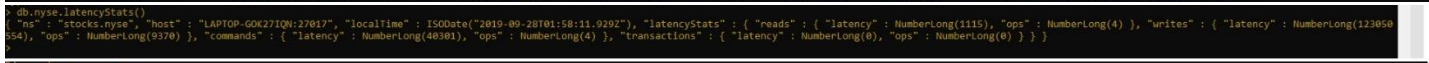
```
> db.nyse.getPlanCache()
PlanCache for collection nyse. Type help() for more info.
>
```

INFO 7250 : ENGINEERING OF BIG DATA SYSTEMS

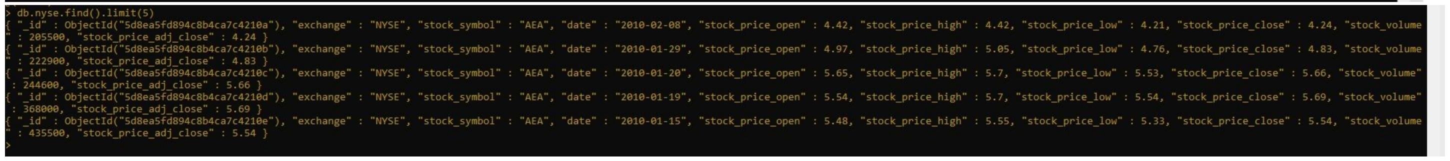
NAME : KAUSHAL CHAUDHARY
NUID : 001886763



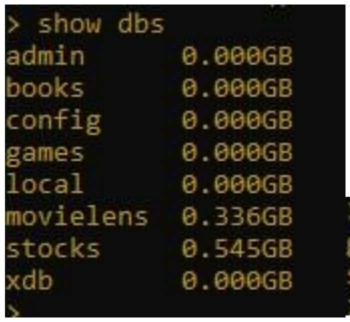
```
> db.help()
DB methods:
        db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command [just calls db.runCommand(...)]
        db.aggregate([pipeline], {options}) - performs a collectionless aggregation on this database; returns a cursor
        db.auth(username, password)
        db.cloneDatabase(fromhost) - will only function with MongoDB 4.0 and below
        db.commandHelp(name) returns the help for the command
        db.copyDatabase(fromdb, todb, fromhost) - will only function with MongoDB 4.0 and below
        db.createCollection(name, {size: ..., capped: ..., max: ...})
        db.createUser(userDocument)
        db.createView(name, viewOn, [{$operator: {...}}, ...], {viewOptions})
        db.currentOp() displays currently executing operations in the db
        db.dropDatabase(writeConcern)
        db.dropUser(username)
        db.eval() - deprecated
        db.fsyncLock() flush data to disk and lock server for backups
        db.fsyncUnlock() unlocks server following a db.fsyncLock()
        db.getCollection(cname) same as db['cname'] or db.cname
        db.getCollectionInfos([filter]) - returns a list that contains the names and options of the db's collections
        db.getCollectionNames()
        db.getLastError() - just returns the err msg string
        db.getLastErrorObj() - return full status object
        db.getLogComponents()
        db.getMongo() get the server connection object
        db.getMongo().setSlaveOk() allow queries on a replication slave server
        db.getName()
        db.getProfilingLevel() - deprecated
        db.getProfilingStatus() - returns if profiling is on and slow threshold
        db.getReplicationInfo()
        db.getSiblingDB(name) get the db at the same server as this one
        db.getWriteConcern() - returns the write concern used for any operations on this db, inherited from server object if set
        db.hostInfo() get details about the server's host
        db.isMaster() check replica primary status
        db.killOp(opid) kills the current operation in the db
        db.listCommands() lists all the db commands
        db.loadServerScripts() loads all the scripts in db.system.js
        db.logout()
        db.printCollectionStats()
        db.printReplicationInfo()
        db.printShardingStatus()
        db.printSlaveReplicationInfo()
        db.resetError()
        db.runCommand(cmdObj) run a database command.  if cmdObj is a string, turns it into {cmdObj: 1}
        db.serverStatus()
        db.setLogLevel(level,<component>)
        db.setProfilingLevel(level,slowms) 0=off 1=slow 2=all
        db.setVerboseShell(flag) display extra information in shell output
        db.setWriteConcern(<write concern doc>) - sets the write concern for writes to the db
        db.shutdownServer()
        db.stats()
        db.unsetWriteConcern(<write concern doc>) - unsets the write concern for writes to the db
        db.version() current version of the server
        db.watch() - opens a change stream cursor for a database to report on all  changes to its non-system collections.
```



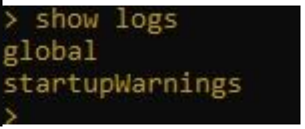
```
> db.nyse.latencyStats()
{ "ns" : "stocks.nyse", "host" : "LAPTOP-GOK27IQN:27017", "localTime" : ISODate("2019-09-28T01:58:11.929Z"), "latencyStats" : { "reads" : { "latency" : NumberLong(1115), "ops" : NumberLong(4) }, "writes" : { "latency" : NumberLong(123050
554), "ops" : NumberLong(9370) }, "commands" : { "latency" : NumberLong(40301), "ops" : NumberLong(4) }, "transactions" : { "latency" : NumberLong(0), "ops" : NumberLong(0) } } }
>
```



```
> db.nyse.find().limit(5)
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c4210a"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2010-02-08", "stock_price_open" : 4.42, "stock_price_high" : 4.42, "stock_price_low" : 4.21, "stock_price_close" : 4.24, "stock_volume
" : 205500, "stock_price_adj_close" : 4.24 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c4210b"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2010-01-29", "stock_price_open" : 4.97, "stock_price_high" : 5.05, "stock_price_low" : 4.76, "stock_price_close" : 4.83, "stock_volume
" : 222900, "stock_price_adj_close" : 4.83 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c4210c"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2010-01-20", "stock_price_open" : 5.65, "stock_price_high" : 5.7, "stock_price_low" : 5.53, "stock_price_close" : 5.66, "stock_volume"
 : 244600, "stock_price_adj_close" : 5.66 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c4210d"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2010-01-19", "stock_price_open" : 5.54, "stock_price_high" : 5.7, "stock_price_low" : 5.54, "stock_price_close" : 5.69, "stock_volume"
 : 368000, "stock_price_adj_close" : 5.69 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c4210e"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2010-01-15", "stock_price_open" : 5.48, "stock_price_high" : 5.55, "stock_price_low" : 5.33, "stock_price_close" : 5.54, "stock_volume
" : 435500, "stock_price_adj_close" : 5.54 }
>
```

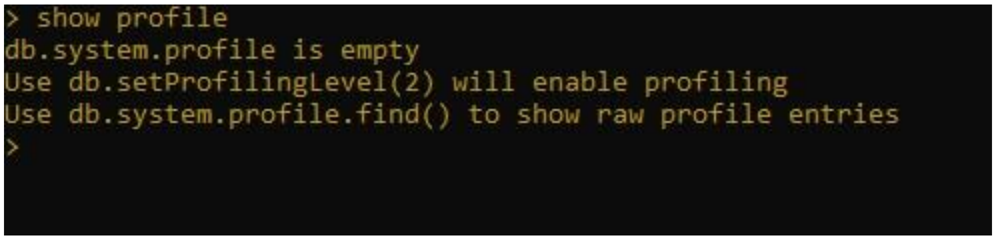


```
> show dbs
admin       0.000GB
books       0.000GB
config      0.000GB
games       0.000GB
local       0.000GB
movielens   0.336GB
stocks      0.545GB
xdb         0.000GB
>
```

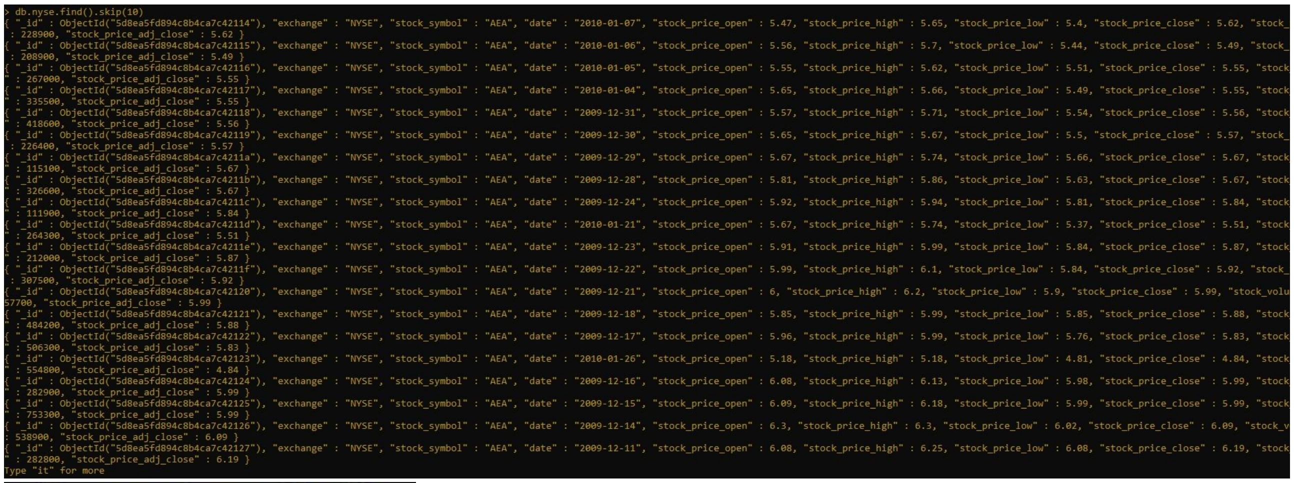


```
> show logs
global
startupWarnings
>
```

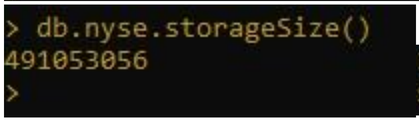
NAME : KAUSHAL CHAUDHARY
NUID : 001886763



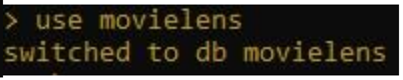
```
> show profile
db.system.profile is empty
Use db.setProfilingLevel(2) will enable profiling
Use db.system.profile.find() to show raw profile entries
>
```



```
> db.nyse.find().skip(10)
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c42114"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2010-01-07", "stock_price_open" : 5.47, "stock_price_high" : 5.65, "stock_price_low" : 5.4, "stock_price_close" : 5.62, "stock_
 : 228900, "stock_price_adj_close" : 5.62 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c42115"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2010-01-06", "stock_price_open" : 5.56, "stock_price_high" : 5.7, "stock_price_low" : 5.44, "stock_price_close" : 5.49, "stock_
 : 208900, "stock_price_adj_close" : 5.49 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c42116"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2010-01-05", "stock_price_open" : 5.55, "stock_price_high" : 5.62, "stock_price_low" : 5.51, "stock_price_close" : 5.55, "stock
" : 267000, "stock_price_adj_close" : 5.55 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c42117"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2010-01-04", "stock_price_open" : 5.65, "stock_price_high" : 5.66, "stock_price_low" : 5.49, "stock_price_close" : 5.55, "stock
" : 335500, "stock_price_adj_close" : 5.55 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c42118"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2009-12-31", "stock_price_open" : 5.57, "stock_price_high" : 5.71, "stock_price_low" : 5.54, "stock_price_close" : 5.56, "stock
" : 418600, "stock_price_adj_close" : 5.56 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c42119"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2009-12-30", "stock_price_open" : 5.65, "stock_price_high" : 5.67, "stock_price_low" : 5.5, "stock_price_close" : 5.57, "stock_
 : 226400, "stock_price_adj_close" : 5.57 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c4211a"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2009-12-29", "stock_price_open" : 5.67, "stock_price_high" : 5.74, "stock_price_low" : 5.66, "stock_price_close" : 5.67, "stock
" : 115100, "stock_price_adj_close" : 5.67 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c4211b"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2009-12-28", "stock_price_open" : 5.81, "stock_price_high" : 5.86, "stock_price_low" : 5.63, "stock_price_close" : 5.67, "stock
" : 326600, "stock_price_adj_close" : 5.67 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c4211c"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2009-12-24", "stock_price_open" : 5.92, "stock_price_high" : 5.94, "stock_price_low" : 5.81, "stock_price_close" : 5.84, "stock
" : 111900, "stock_price_adj_close" : 5.84 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c4211d"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2010-01-21", "stock_price_open" : 5.67, "stock_price_high" : 5.74, "stock_price_low" : 5.37, "stock_price_close" : 5.51, "stock
" : 264300, "stock_price_adj_close" : 5.51 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c4211e"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2009-12-23", "stock_price_open" : 5.91, "stock_price_high" : 5.99, "stock_price_low" : 5.84, "stock_price_close" : 5.87, "stock
" : 212000, "stock_price_adj_close" : 5.87 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c4211f"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2009-12-22", "stock_price_open" : 5.99, "stock_price_high" : 6.1, "stock_price_low" : 5.84, "stock_price_close" : 5.92, "stock_
 : 307500, "stock_price_adj_close" : 5.92 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c42120"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2009-12-21", "stock_price_open" : 6, "stock_price_high" : 6.2, "stock_price_low" : 5.9, "stock_price_close" : 5.99, "stock_volu
57700, "stock_price_adj_close" : 5.99 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c42121"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2009-12-18", "stock_price_open" : 5.85, "stock_price_high" : 5.99, "stock_price_low" : 5.85, "stock_price_close" : 5.88, "stock
" : 484200, "stock_price_adj_close" : 5.88 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c42122"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2009-12-17", "stock_price_open" : 5.96, "stock_price_high" : 5.99, "stock_price_low" : 5.76, "stock_price_close" : 5.83, "stock
" : 506300, "stock_price_adj_close" : 5.83 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c42123"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2010-01-26", "stock_price_open" : 5.18, "stock_price_high" : 5.18, "stock_price_low" : 4.81, "stock_price_close" : 4.84, "stock
" : 554800, "stock_price_adj_close" : 4.84 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c42124"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2009-12-16", "stock_price_open" : 6.08, "stock_price_high" : 6.13, "stock_price_low" : 5.98, "stock_price_close" : 5.99, "stock
" : 282900, "stock_price_adj_close" : 5.99 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c42125"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2009-12-15", "stock_price_open" : 6.09, "stock_price_high" : 6.18, "stock_price_low" : 5.99, "stock_price_close" : 5.99, "stock
" : 753300, "stock_price_adj_close" : 5.99 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c42126"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2009-12-14", "stock_price_open" : 6.3, "stock_price_high" : 6.3, "stock_price_low" : 6.02, "stock_price_close" : 6.09, "stock_v
: 538900, "stock_price_adj_close" : 6.09 }
{ "_id" : ObjectId("5d8ea5fd894c8b4ca7c42127"), "exchange" : "NYSE", "stock_symbol" : "AEA", "date" : "2009-12-11", "stock_price_open" : 6.08, "stock_price_high" : 6.25, "stock_price_low" : 6.08, "stock_price_close" : 6.19, "stock
" : 282800, "stock_price_adj_close" : 6.19 }
Type "it" for more
```



```
> db.nyse.storageSize()
491053056
>
```

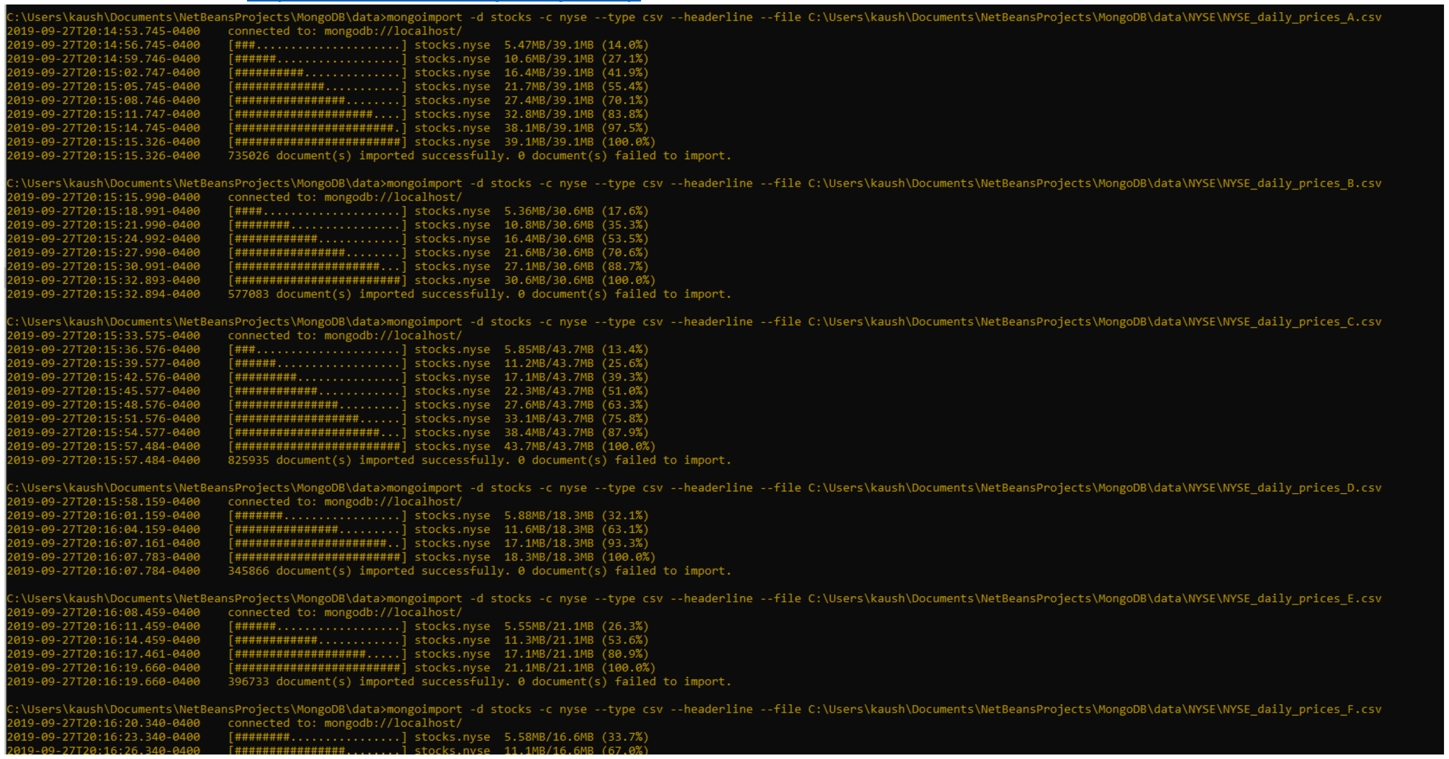


```
> use movielens
switched to db movielens
```

NAME : KAUSHAL CHAUDHARY
NUID : 001886763

PART 7 :

Write a .bat (for Windows) or .sh (for MacOS) to import the entire NYSE dataset (stocks A to Z) into MongoDB.
NYSE Dataset Link: http://msis.neu.edu/nyse/nyse.zip

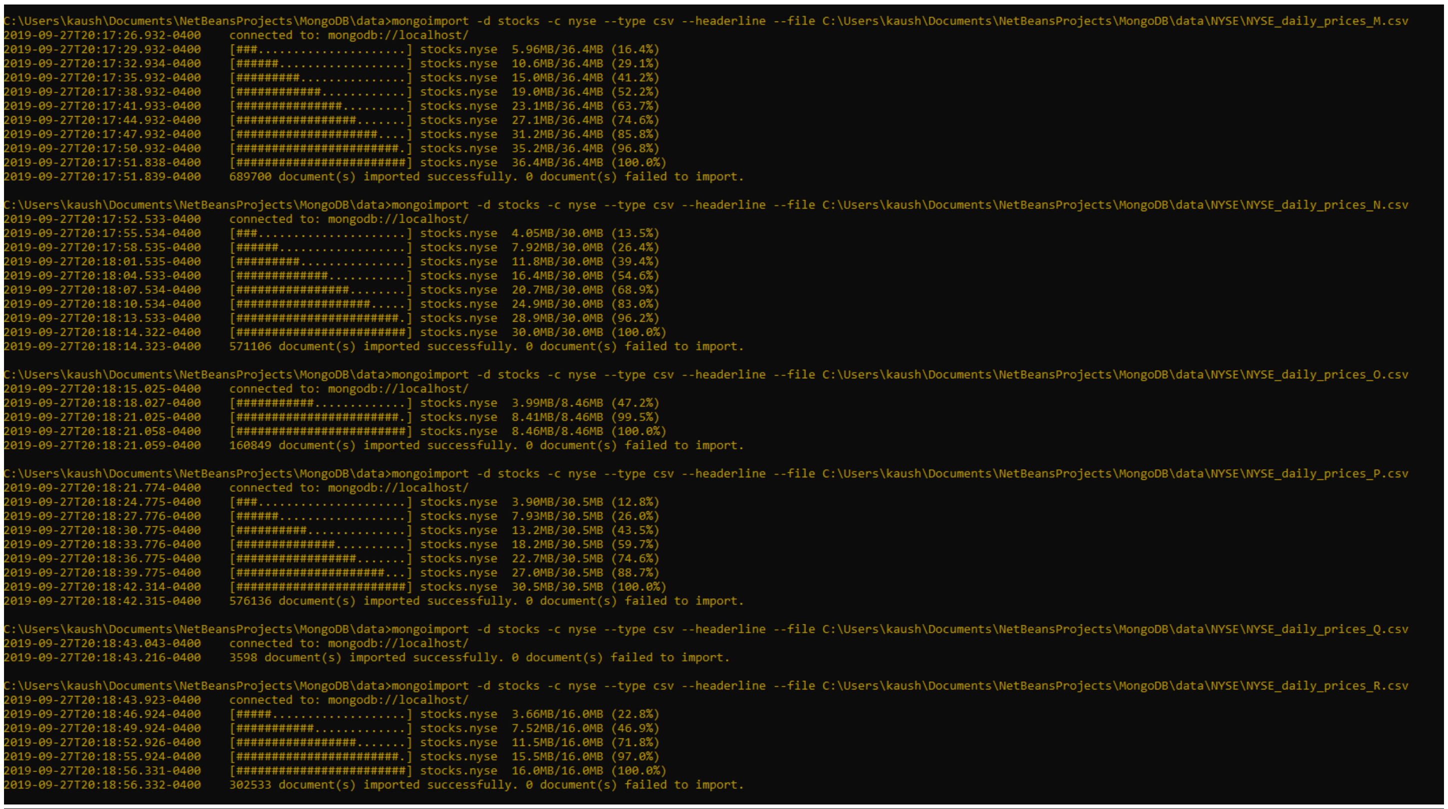


```
C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_daily_prices_A.csv
2019-09-27T20:14:53.745-0400    connected to: mongodb://localhost/
2019-09-27T20:14:56.745-0400    [###.....................] stocks.nyse  5.47MB/39.1MB (14.0%)
2019-09-27T20:14:59.746-0400    [######..................] stocks.nyse  10.6MB/39.1MB (27.1%)
2019-09-27T20:15:02.747-0400    [##########..............] stocks.nyse  16.4MB/39.1MB (41.9%)
2019-09-27T20:15:05.745-0400    [#############...........] stocks.nyse  21.7MB/39.1MB (55.4%)
2019-09-27T20:15:08.746-0400    [################........] stocks.nyse  27.4MB/39.1MB (70.1%)
2019-09-27T20:15:11.747-0400    [####################....] stocks.nyse  32.8MB/39.1MB (83.8%)
2019-09-27T20:15:14.745-0400    [#######################.] stocks.nyse  38.1MB/39.1MB (97.5%)
2019-09-27T20:15:15.326-0400    [########################] stocks.nyse  39.1MB/39.1MB (100.0%)
2019-09-27T20:15:15.326-0400    735026 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_daily_prices_B.csv
2019-09-27T20:15:15.990-0400    connected to: mongodb://localhost/
2019-09-27T20:15:18.991-0400    [####....................] stocks.nyse  5.36MB/30.6MB (17.6%)
2019-09-27T20:15:21.990-0400    [########................] stocks.nyse  10.8MB/30.6MB (35.3%)
2019-09-27T20:15:24.992-0400    [############............] stocks.nyse  16.4MB/30.6MB (53.5%)
2019-09-27T20:15:27.990-0400    [################........] stocks.nyse  21.6MB/30.6MB (70.6%)
2019-09-27T20:15:30.991-0400    [#####################...] stocks.nyse  27.1MB/30.6MB (88.7%)
2019-09-27T20:15:32.893-0400    [########################] stocks.nyse  30.6MB/30.6MB (100.0%)
2019-09-27T20:15:32.894-0400    577083 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_daily_prices_C.csv
2019-09-27T20:15:33.575-0400    connected to: mongodb://localhost/
2019-09-27T20:15:36.576-0400    [###.....................] stocks.nyse  5.85MB/43.7MB (13.4%)
2019-09-27T20:15:39.577-0400    [######..................] stocks.nyse  11.2MB/43.7MB (25.6%)
2019-09-27T20:15:42.576-0400    [#########...............] stocks.nyse  17.1MB/43.7MB (39.3%)
2019-09-27T20:15:45.577-0400    [############............] stocks.nyse  22.3MB/43.7MB (51.0%)
2019-09-27T20:15:48.576-0400    [###############.........] stocks.nyse  27.6MB/43.7MB (63.3%)
2019-09-27T20:15:51.576-0400    [##################......] stocks.nyse  33.1MB/43.7MB (75.8%)
2019-09-27T20:15:54.577-0400    [#####################...] stocks.nyse  38.4MB/43.7MB (87.9%)
2019-09-27T20:15:57.484-0400    [########################] stocks.nyse  43.7MB/43.7MB (100.0%)
2019-09-27T20:15:57.484-0400    825935 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_daily_prices_D.csv
2019-09-27T20:15:58.159-0400    connected to: mongodb://localhost/
2019-09-27T20:16:01.159-0400    [#######.................] stocks.nyse  5.88MB/18.3MB (32.1%)
2019-09-27T20:16:04.159-0400    [###############.........] stocks.nyse  11.6MB/18.3MB (63.1%)
2019-09-27T20:16:07.161-0400    [######################..] stocks.nyse  17.1MB/18.3MB (93.3%)
2019-09-27T20:16:07.783-0400    [########################] stocks.nyse  18.3MB/18.3MB (100.0%)
2019-09-27T20:16:07.784-0400    345866 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_daily_prices_E.csv
2019-09-27T20:16:08.459-0400    connected to: mongodb://localhost/
2019-09-27T20:16:11.459-0400    [######..................] stocks.nyse  5.55MB/21.1MB (26.3%)
2019-09-27T20:16:14.459-0400    [############............] stocks.nyse  11.3MB/21.1MB (53.6%)
2019-09-27T20:16:17.461-0400    [###################.....] stocks.nyse  17.1MB/21.1MB (80.9%)
2019-09-27T20:16:19.660-0400    [########################] stocks.nyse  21.1MB/21.1MB (100.0%)
2019-09-27T20:16:19.660-0400    396733 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_daily_prices_F.csv
2019-09-27T20:16:20.340-0400    connected to: mongodb://localhost/
2019-09-27T20:16:23.340-0400    [########................] stocks.nyse  5.58MB/16.6MB (33.7%)
2019-09-27T20:16:26.340-0400    [################........] stocks.nyse  11.1MB/16.6MB (67.0%)
```

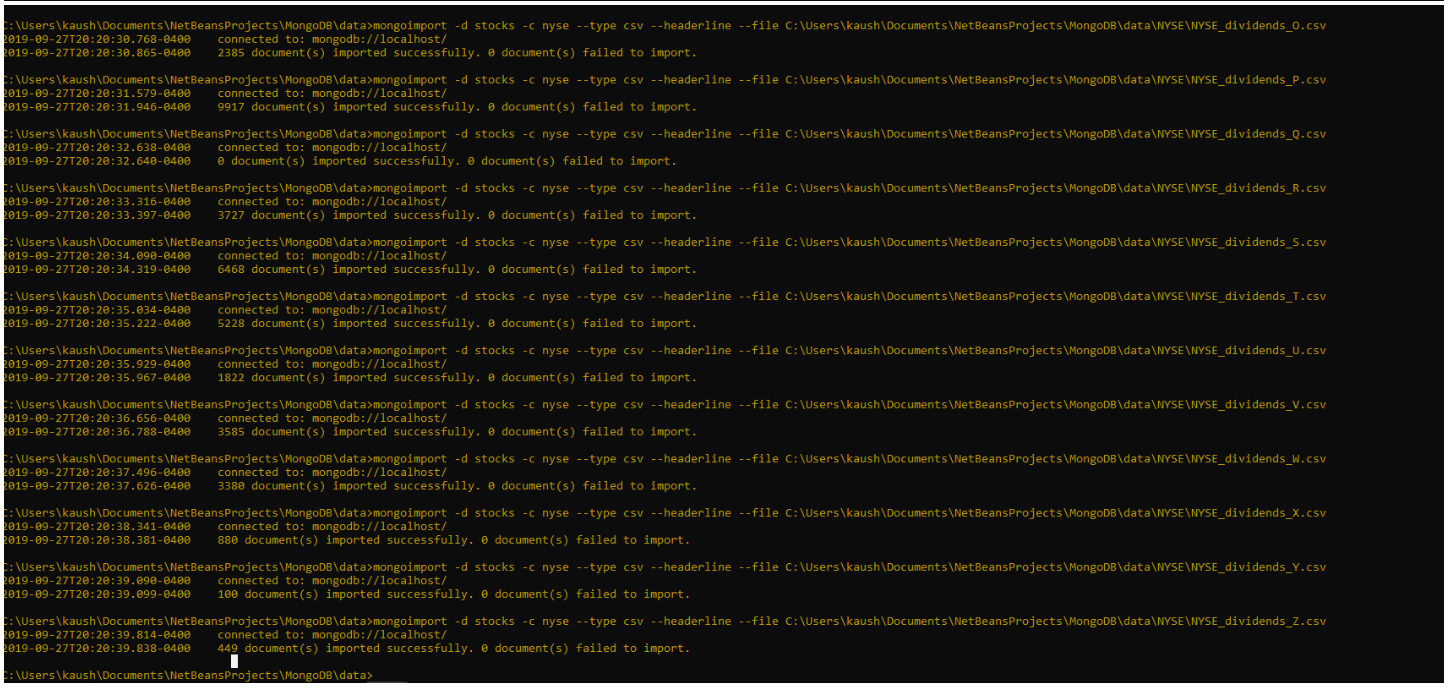
INFO 7250 : ENGINEERING OF BIG DATA SYSTEMS

NAME : KAUSHAL CHAUDHARY

NUID : 001886763



```
C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_daily_prices_M.csv
2019-09-27T20:17:26.932-0400    connected to: mongodb://localhost/
2019-09-27T20:17:29.932-0400    [###.....................] stocks.nyse  5.96MB/36.4MB (16.4%)
2019-09-27T20:17:32.934-0400    [######..................] stocks.nyse  10.6MB/36.4MB (29.1%)
2019-09-27T20:17:35.932-0400    [#########...............] stocks.nyse  15.0MB/36.4MB (41.2%)
2019-09-27T20:17:38.932-0400    [############............] stocks.nyse  19.0MB/36.4MB (52.2%)
2019-09-27T20:17:41.933-0400    [###############.........] stocks.nyse  23.1MB/36.4MB (63.7%)
2019-09-27T20:17:44.932-0400    [#################.......] stocks.nyse  27.1MB/36.4MB (74.6%)
2019-09-27T20:17:47.932-0400    [####################....] stocks.nyse  31.2MB/36.4MB (85.8%)
2019-09-27T20:17:50.932-0400    [#######################.] stocks.nyse  35.2MB/36.4MB (96.8%)
2019-09-27T20:17:51.838-0400    [########################] stocks.nyse  36.4MB/36.4MB (100.0%)
2019-09-27T20:17:51.839-0400    689700 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_daily_prices_N.csv
2019-09-27T20:17:52.533-0400    connected to: mongodb://localhost/
2019-09-27T20:17:55.534-0400    [###.....................] stocks.nyse  4.05MB/30.0MB (13.5%)
2019-09-27T20:17:58.535-0400    [######..................] stocks.nyse  7.92MB/30.0MB (26.4%)
2019-09-27T20:18:01.535-0400    [#########...............] stocks.nyse  11.8MB/30.0MB (39.4%)
2019-09-27T20:18:04.533-0400    [#############...........] stocks.nyse  16.4MB/30.0MB (54.6%)
2019-09-27T20:18:07.534-0400    [################........] stocks.nyse  20.7MB/30.0MB (68.9%)
2019-09-27T20:18:10.534-0400    [###################.....] stocks.nyse  24.9MB/30.0MB (83.0%)
2019-09-27T20:18:13.533-0400    [#######################.] stocks.nyse  28.9MB/30.0MB (96.2%)
2019-09-27T20:18:14.322-0400    [########################] stocks.nyse  30.0MB/30.0MB (100.0%)
2019-09-27T20:18:14.323-0400    571106 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_daily_prices_O.csv
2019-09-27T20:18:15.025-0400    connected to: mongodb://localhost/
2019-09-27T20:18:18.027-0400    [###########.............] stocks.nyse  3.99MB/8.46MB (47.2%)
2019-09-27T20:18:21.025-0400    [#######################.] stocks.nyse  8.41MB/8.46MB (99.5%)
2019-09-27T20:18:21.058-0400    [########################] stocks.nyse  8.46MB/8.46MB (100.0%)
2019-09-27T20:18:21.059-0400    160849 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_daily_prices_P.csv
2019-09-27T20:18:21.774-0400    connected to: mongodb://localhost/
2019-09-27T20:18:24.775-0400    [###.....................] stocks.nyse  3.90MB/30.5MB (12.8%)
2019-09-27T20:18:27.776-0400    [######..................] stocks.nyse  7.93MB/30.5MB (26.0%)
2019-09-27T20:18:30.775-0400    [##########..............] stocks.nyse  13.2MB/30.5MB (43.5%)
2019-09-27T20:18:33.776-0400    [##############..........] stocks.nyse  18.2MB/30.5MB (59.7%)
2019-09-27T20:18:36.775-0400    [#################.......] stocks.nyse  22.7MB/30.5MB (74.6%)
2019-09-27T20:18:39.775-0400    [#####################...] stocks.nyse  27.0MB/30.5MB (88.7%)
2019-09-27T20:18:42.314-0400    [########################] stocks.nyse  30.5MB/30.5MB (100.0%)
2019-09-27T20:18:42.315-0400    576136 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_daily_prices_Q.csv
2019-09-27T20:18:43.043-0400    connected to: mongodb://localhost/
2019-09-27T20:18:43.216-0400    3598 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_daily_prices_R.csv
2019-09-27T20:18:43.923-0400    connected to: mongodb://localhost/
2019-09-27T20:18:46.924-0400    [#####...................] stocks.nyse  3.66MB/16.0MB (22.8%)
2019-09-27T20:18:49.924-0400    [###########.............] stocks.nyse  7.52MB/16.0MB (46.9%)
2019-09-27T20:18:52.926-0400    [#################.......] stocks.nyse  11.5MB/16.0MB (71.8%)
2019-09-27T20:18:55.924-0400    [#######################.] stocks.nyse  15.5MB/16.0MB (97.0%)
2019-09-27T20:18:56.331-0400    [########################] stocks.nyse  16.0MB/16.0MB (100.0%)
2019-09-27T20:18:56.332-0400    302533 document(s) imported successfully. 0 document(s) failed to import.
```



```
:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_dividends_O.csv
019-09-27T20:20:30.768-0400    connected to: mongodb://localhost/
019-09-27T20:20:30.865-0400    2385 document(s) imported successfully. 0 document(s) failed to import.

:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_dividends_P.csv
019-09-27T20:20:31.579-0400    connected to: mongodb://localhost/
019-09-27T20:20:31.946-0400    9917 document(s) imported successfully. 0 document(s) failed to import.

:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_dividends_Q.csv
019-09-27T20:20:32.638-0400    connected to: mongodb://localhost/
019-09-27T20:20:32.640-0400    0 document(s) imported successfully. 0 document(s) failed to import.

:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_dividends_R.csv
019-09-27T20:20:33.316-0400    connected to: mongodb://localhost/
019-09-27T20:20:33.397-0400    3727 document(s) imported successfully. 0 document(s) failed to import.

:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_dividends_S.csv
019-09-27T20:20:34.090-0400    connected to: mongodb://localhost/
019-09-27T20:20:34.319-0400    6468 document(s) imported successfully. 0 document(s) failed to import.

:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_dividends_T.csv
019-09-27T20:20:35.034-0400    connected to: mongodb://localhost/
019-09-27T20:20:35.222-0400    5228 document(s) imported successfully. 0 document(s) failed to import.

:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_dividends_U.csv
019-09-27T20:20:35.929-0400    connected to: mongodb://localhost/
019-09-27T20:20:35.967-0400    1822 document(s) imported successfully. 0 document(s) failed to import.

:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_dividends_V.csv
019-09-27T20:20:36.656-0400    connected to: mongodb://localhost/
019-09-27T20:20:36.788-0400    3585 document(s) imported successfully. 0 document(s) failed to import.

:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_dividends_W.csv
019-09-27T20:20:37.496-0400    connected to: mongodb://localhost/
019-09-27T20:20:37.626-0400    3380 document(s) imported successfully. 0 document(s) failed to import.

:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_dividends_X.csv
019-09-27T20:20:38.341-0400    connected to: mongodb://localhost/
019-09-27T20:20:38.381-0400    880 document(s) imported successfully. 0 document(s) failed to import.

:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_dividends_Y.csv
019-09-27T20:20:39.090-0400    connected to: mongodb://localhost/
019-09-27T20:20:39.099-0400    100 document(s) imported successfully. 0 document(s) failed to import.

:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>mongoimport -d stocks -c nyse --type csv --headerline --file C:\Users\kaush\Documents\NetBeansProjects\MongoDB\data\NYSE\NYSE_dividends_Z.csv
019-09-27T20:20:39.814-0400    connected to: mongodb://localhost/
019-09-27T20:20:39.838-0400    449 document(s) imported successfully. 0 document(s) failed to import.

:\Users\kaush\Documents\NetBeansProjects\MongoDB\data>
```