**Assignment 2**

**PART 2- Write a .bat/.sh to import the entire NYSE dataset (stocks A to Z) into MongoDB. For MacOS, there is a sample .sh script in the slides. For windows users, you may refer to the tutorial [https://ss64.com/nt/for\_d.html] or any online resource. The idea behind this is that instead of running mongoimport manually on each file, we create a script, and loop through all the files in a directory, and run mongoimport at each iteration.  
 NYSE Dataset Link:** [**http://msis.neu.edu/nyse/**](http://msis.neu.edu/nyse/)

**Answer-** We create two batch files. One for downloading data from internet and second for unzipping and saving that downloaded data in the folder.

**1. Batch File to download data from internet-**

**File Name- downloadFile.bat**

@echo off

setlocal

certutil.exe -urlcache -split -f "http://msis.neu.edu/nyse/nyse.zip" C:\temp\nyse.zip

exit /b

**2. Batch file to unzip the downloaded data and save in folder**

**File Name- unzip.bat**

@echo off

setlocal

cd /d %~dp0

Call :UnZipFile "C:\temp\unzip" "C:\Users\ankit\Documents\4th\_Sem\EnggBigData\Assignments\Assignment\_2\nyse.zip"

exit /b

:UnZipFile <ExtractTo> <newzipfile>

set vbs="%temp%\\_.vbs"

if exist %vbs% del /f /q %vbs%

>%vbs% echo Set fso = CreateObject("Scripting.FileSystemObject")

>>%vbs% echo If NOT fso.FolderExists(%1) Then

>>%vbs% echo fso.CreateFolder(%1)

>>%vbs% echo End If

>>%vbs% echo set objShell = CreateObject("Shell.Application")

>>%vbs% echo set FilesInZip=objShell.NameSpace(%2).items

>>%vbs% echo objShell.NameSpace(%1).CopyHere(FilesInZip)

>>%vbs% echo Set fso = Nothing

>>%vbs% echo Set objShell = Nothing

cscript //nologo %vbs%

if exist %vbs% del /f /q %vbs%

**Final Step: Then we create a final batch file calling above 2 batch files in it and having the command to import all the data in MongoDB**

@echo off

CALL downloadFile.bat

CALL unzip.bat

FOR %%G IN (A B C D E F G H I J K L M N O P Q R S T U V W X Y Z) DO mongoimport --db nyse --type csv --collection stocks --headerline --file C:\temp\unzip\nyse\nyse\_daily\_prices\_%%G.csv

Pause

It will save the data in mongoDB

**Database** - nyse

**Collection** – stocks

**PART 3.1. Use the NYSE database to find the average price of stock\_price\_high values for each stock using MapReduce.**

**Answer:**

**MAP Function:**

function (){

emit({stock\_name:this.stock\_symbol},{average\_price\_high:this.stock\_price\_high});

}

**REDUCE Function:**

function (key,values){

var average = 0;

var sum = 0;

for(var i = 0; i<values.length;i++){

sum += values[i].average\_price\_high;

}

average = sum/values.length;

return {average\_price\_high: average };

}

**Output:**

> db.stocks.mapReduce(map,reduce,{out:"AverageStockHigh"})

{

"result" : "AverageStockHigh",

"timeMillis" : 202847,

"counts" : {

"input" : 9242031,

"emit" : 9242031,

"reduce" : 95328,

"output" : 2853

},

"ok" : 1

}

> db.AverageStockHigh.find()

{ "\_id" : { "stock\_name" : NaN }, "value" : { "average\_price\_high" : 14.348378377373953 } }

{ "\_id" : { "stock\_name" : "AA" }, "value" : { "average\_price\_high" : 38.89387426252349 } }

{ "\_id" : { "stock\_name" : "AAI" }, "value" : { "average\_price\_high" : 21.93006691954985 } }

{ "\_id" : { "stock\_name" : "AAN" }, "value" : { "average\_price\_high" : 8.023218602241423 } }

{ "\_id" : { "stock\_name" : "AAP" }, "value" : { "average\_price\_high" : 45.02059602352953 } }

{ "\_id" : { "stock\_name" : "AAR" }, "value" : { "average\_price\_high" : 22.843163122227384 } }

{ "\_id" : { "stock\_name" : "AAV" }, "value" : { "average\_price\_high" : 14.410486322434165 } }

{ "\_id" : { "stock\_name" : "AB" }, "value" : { "average\_price\_high" : 4.039495303803753 } }

{ "\_id" : { "stock\_name" : "ABA" }, "value" : { "average\_price\_high" : 25.344888492777937 } }

{ "\_id" : { "stock\_name" : "ABB" }, "value" : { "average\_price\_high" : 17.793867275003528 } }

{ "\_id" : { "stock\_name" : "ABC" }, "value" : { "average\_price\_high" : 23.226585158085324 } }

{ "\_id" : { "stock\_name" : "ABD" }, "value" : { "average\_price\_high" : 25.712232560838924 } }

{ "\_id" : { "stock\_name" : "ABG" }, "value" : { "average\_price\_high" : 17.714399325290263 } }

{ "\_id" : { "stock\_name" : "ABK" }, "value" : { "average\_price\_high" : 24.14432490925713 } }

{ "\_id" : { "stock\_name" : "ABM" }, "value" : { "average\_price\_high" : 22.924597436891943 } }

{ "\_id" : { "stock\_name" : "ABR" }, "value" : { "average\_price\_high" : 19.681735044525503 } }

{ "\_id" : { "stock\_name" : "ABT" }, "value" : { "average\_price\_high" : 45.67433719913059 } }

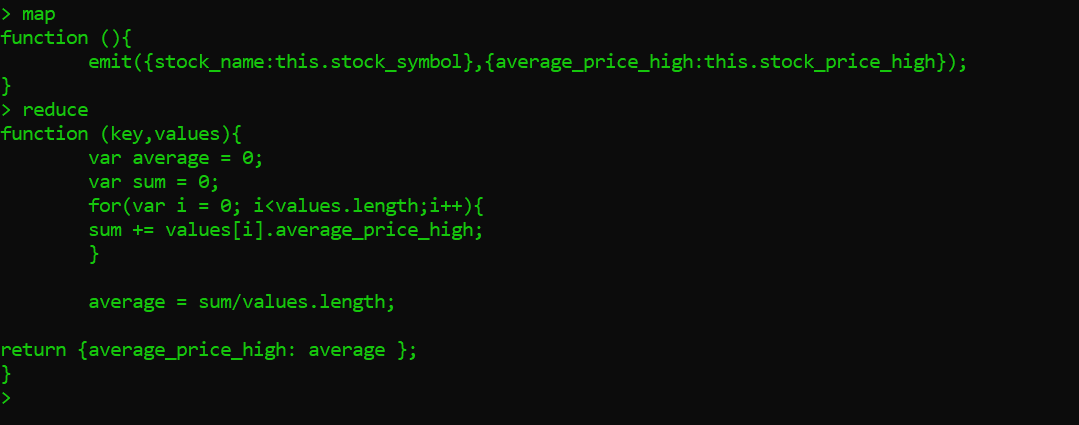
{ "\_id" : { "stock\_name" : "ABV" }, "value" : { "average\_price\_high" : 11.117702558547961 } }

{ "\_id" : { "stock\_name" : "ABVT" }, "value" : { "average\_price\_high" : 45.6995818252496 } }

{ "\_id" : { "stock\_name" : "ABX" }, "value" : { "average\_price\_high" : 4.82917907690965 } }

Type "it" for more

**SCREENSHOTS:**





**PART 3.2. Part 3.1 result will not be correct as AVERAGE is a commutative operation but nor associative. Use a FINALIZER to find the correct average. (Hint: pass sum and count from the reducer) (**[**https://docs.mongodb.com/manual/reference/method/db.collection.mapReduce/index.html**](https://docs.mongodb.com/manual/reference/method/db.collection.mapReduce/index.html)**)**

**Answer:**

**MAP Function:**

function (){

emit({stock\_name:this.stock\_symbol},{total\_price\_high:this.stock\_price\_high, stock\_count:1});

}

**REDUCE Function:**

function (key,values){

var count= 0;

var sum = 0;

for(var i = 0; i<values.length;i++){

sum += values[i].total\_price\_high;

count +=values[i].stock\_count;

}

return {total\_price\_high:sum, stock\_count:count};

}

**Finalize Function:**

function (key, reducedVal) {

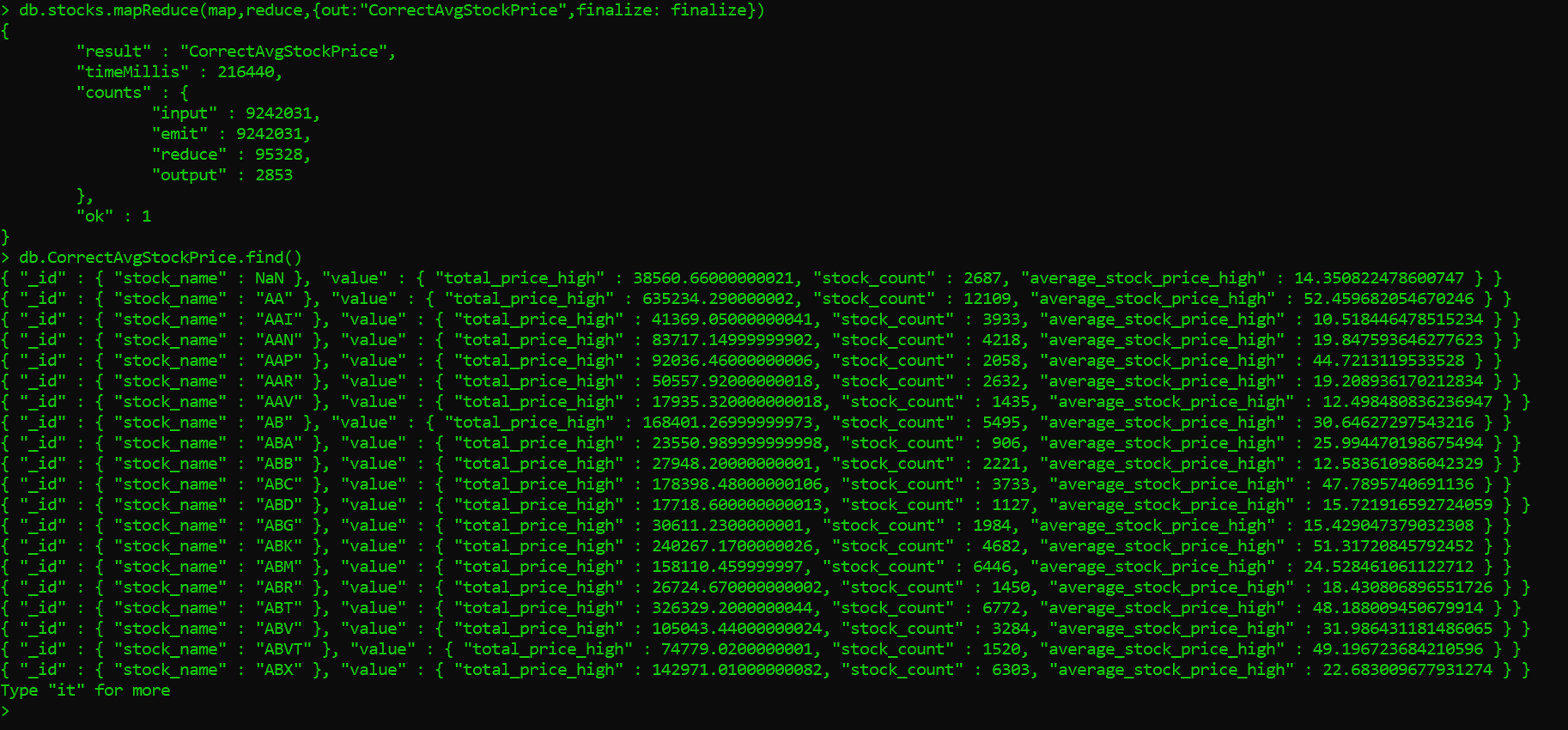
reducedVal.average\_stock\_price\_high = reducedVal.total\_price\_high/reducedVal.stock\_count;

return reducedVal;

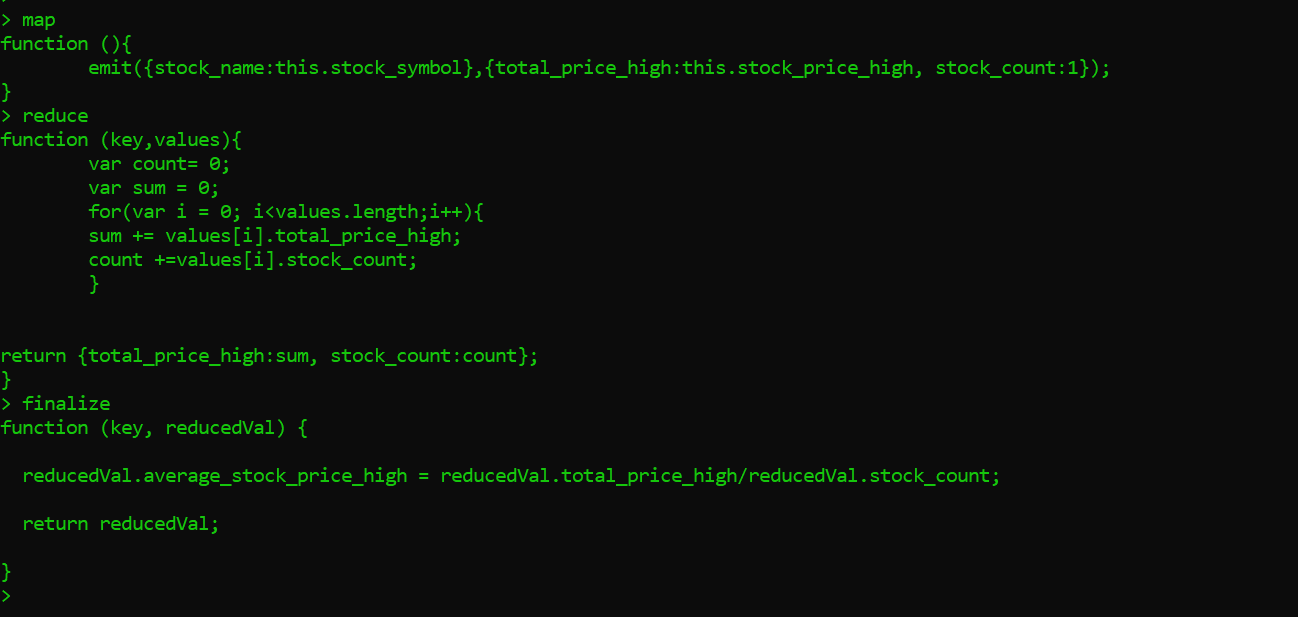
}

**Screenshots:**

**Output-**



**Function Screenshots:**

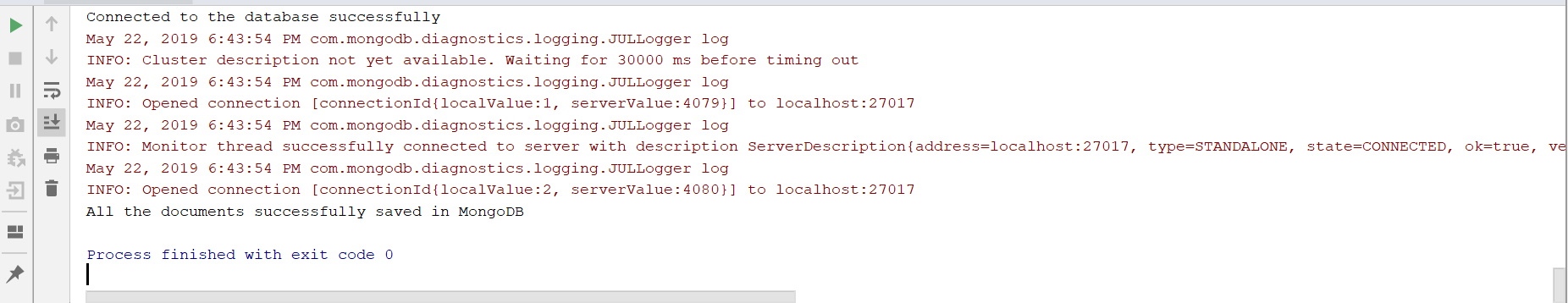


**PART 4. Write a console application (or Swing Application), to read and insert the access.log file into MongoDB. This application will only run once to insert the log file into MongoDB. Once the documents are inserted into MongoDB, perform MapReduce for each of the followings:**

**Answer:**

The java application is added in the assignment folder.

Name of Java File- LogUpload.java





1. **Number of times each IP address accessed any web page**

**MAP-**

function (){

emit({IP\_address:this.ip},{count\_of\_web\_access:1});

}

**REDUCE**

function (key,values){

var sum = 0;

for(var i = 0; i<values.length;i++){

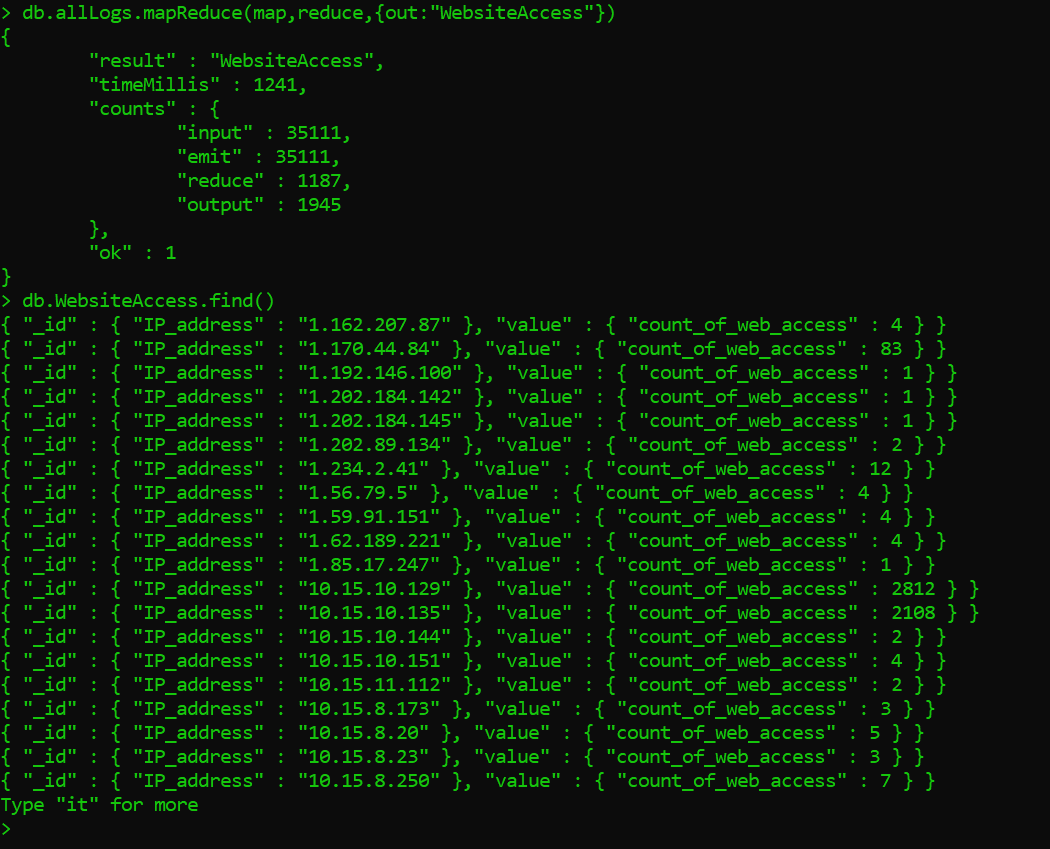
sum += values[i].count\_of\_web\_access;

}

return {count\_of\_web\_access:sum};

}

**OUTPUT**



1. **Latest access date and time from each IP address**

**MAP-**

function (){

emit({IP\_address:this.ip},{latest\_access:this.timestamp});

}

**REDUCE**

function (key, values) {

var maxTime = new Date(values[0].latest\_access);

var location = 0;

for (var i = 1; i < values.length; i++) {

var temp = new Date(values[i].latest\_access);

if (temp > maxTime) {

maxTime = temp;

location = i;

}

}

return {latest\_access:values[location].latest\_access};

}

**OUTPUT**



1. **Find the number of GET, POST, HEAD, etc. requests**

**MAP-**

function (){

emit({request\_type:this.command},{count\_of\_requests:1});

}

**REDUCE**

function (key,values){

var sum = 0;

for(var i = 0; i<values.length;i++){

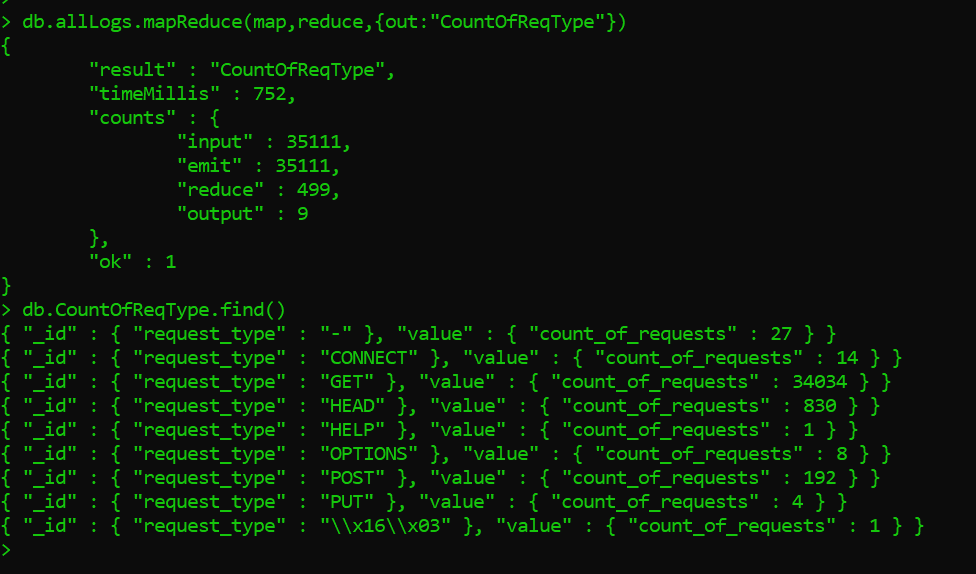
sum += values[i].count\_of\_requests;

}

return {count\_of\_requests:sum};

}

**OUTPUT**



1. **Find the number of STATUS CODES (404, 200, etc)**

**MAP-**

function (){

emit({status\_code:this.statuscode},{count\_status\_code:1});

}

**REDUCE**

function (key,values){

var sum = 0;

for(var i = 0; i<values.length;i++){

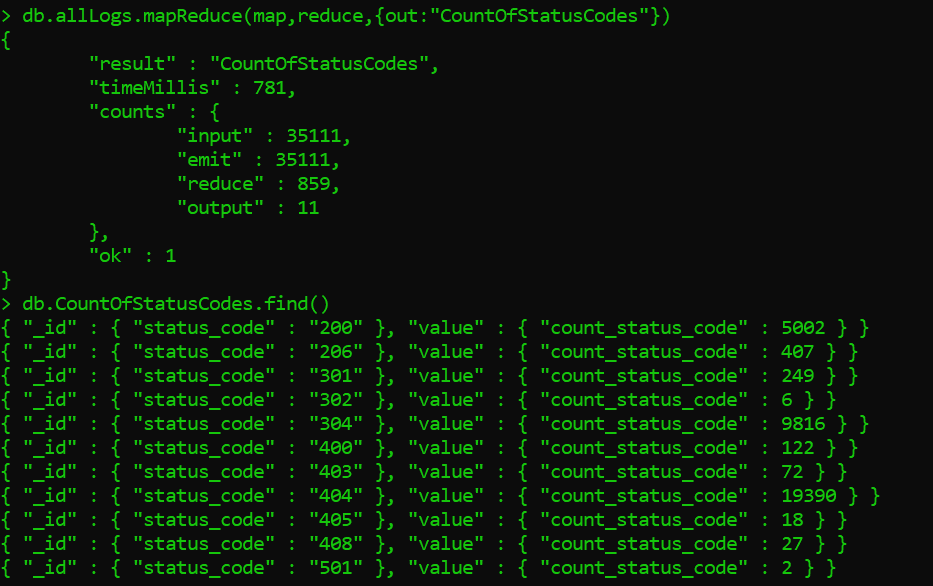
sum += values[i].count\_status\_code;

}

return {count\_status\_code:sum};

}

**OUTPUT**



**PART 4 - PROGRAMMING ASSIGNMENT**

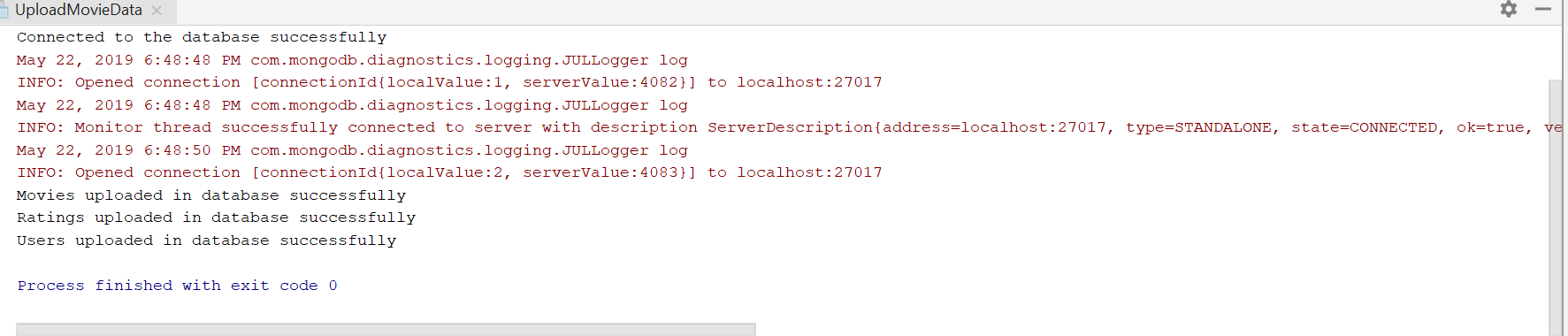
**Write a Java (could be a console app - will only run once to import the data into MongoDB) program to read the following file, and insert into 3 different collections (movies, ratings, tags).**

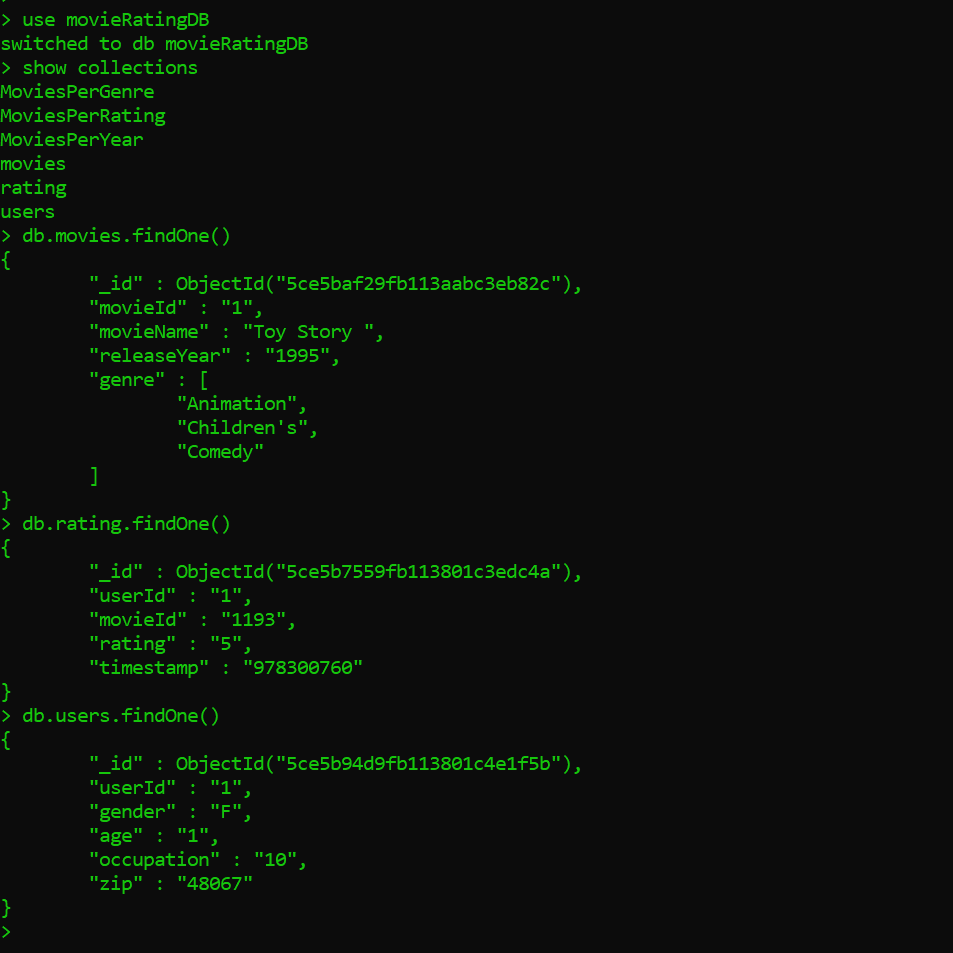
[**http://files.grouplens.org/datasets/movielens/ml-1m.zip**](http://files.grouplens.org/datasets/movielens/ml-1m.zip)

**Answer:**

The java application is added in the assignment folder.

Name of Java File- UploadMovieData.java





**Once the data are inserted into MongoDB, do the followings using MapReduce:**

1. **Number of Movies released per year (Movies Collection)**

**MAP-**

function (){

  emit({release\_year:this.releaseYear},{number\_of\_movies:1});

}

**REDUCE**

function (key,values){

var sum = 0;

values.forEach((val)=>{

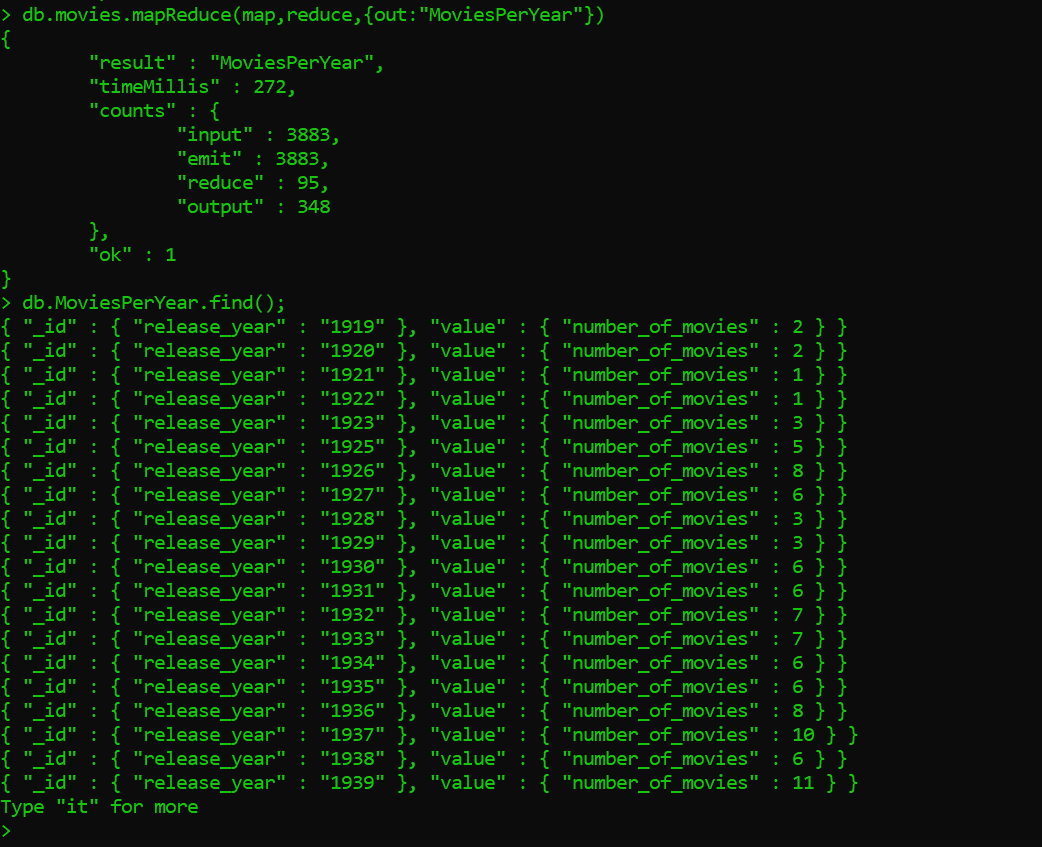
sum += val.number\_of\_movies;

});

return {number\_of\_movies:sum};

}

**OUTPUT**



1. **Number of Movies per genre (Movies Collection)**

**MAP-**

function (){

  for(var i=0;i<this.genre.length;i++){

    emit({genre:this.genre[i]},{no\_of\_movies:1});

  }

}

**REDUCE**

function (key,values){

var sum = 0;

values.forEach((val)=>{

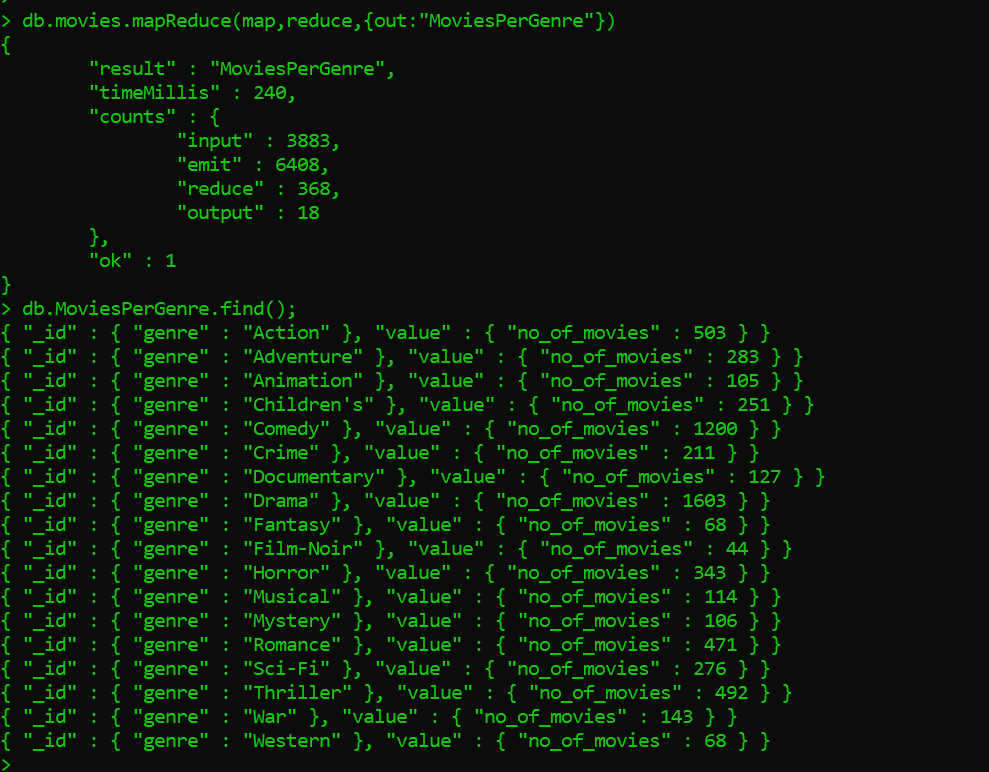
sum += val.no\_of\_movies;

});

return {no\_of\_movies:sum};

}

**OUTPUT**



1. **Number of Movies per rating (Ratings Collection)**

**MAP-**

function (){

    emit({rating:this.rating},{no\_of\_movies:1});

}

**REDUCE**

function (key,values){

var sum = 0;

values.forEach((val)=>{

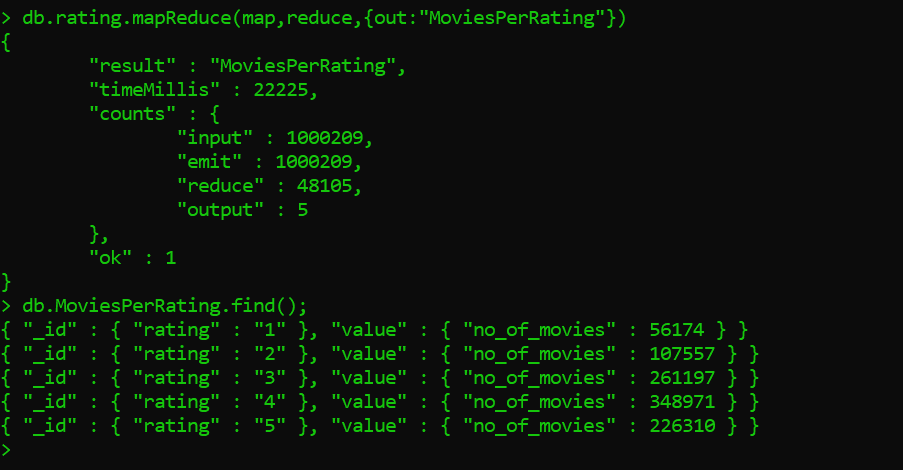
sum += val.no\_of\_movies;

});

return {no\_of\_movies:sum};

}

**OUTPUT**

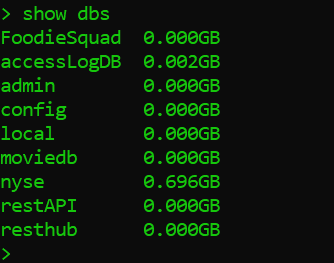


**PART 5 - PROGRAMMING ASSIGNMENT**

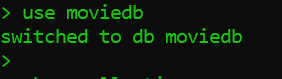
**Execute 5 commands of your choice from each of the following groups, and paste the screenshots in a word document.**

**A:  mongo> help            [5 commands]**

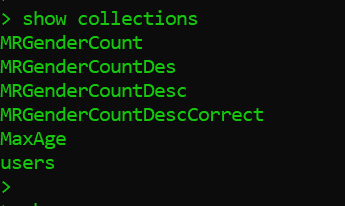
**1. show dbs**



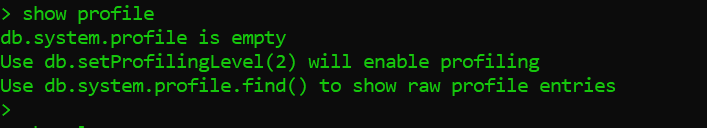
**2. use <db\_name>**



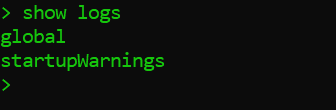
**3. Show collections**



**4. Show profile**



**5. Show logs**



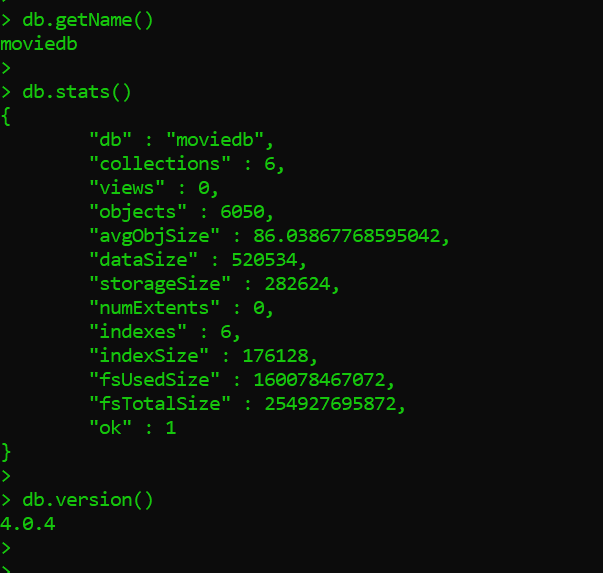
**B: mongo> db.help()        [5 commands]**

**1. db.getName()**

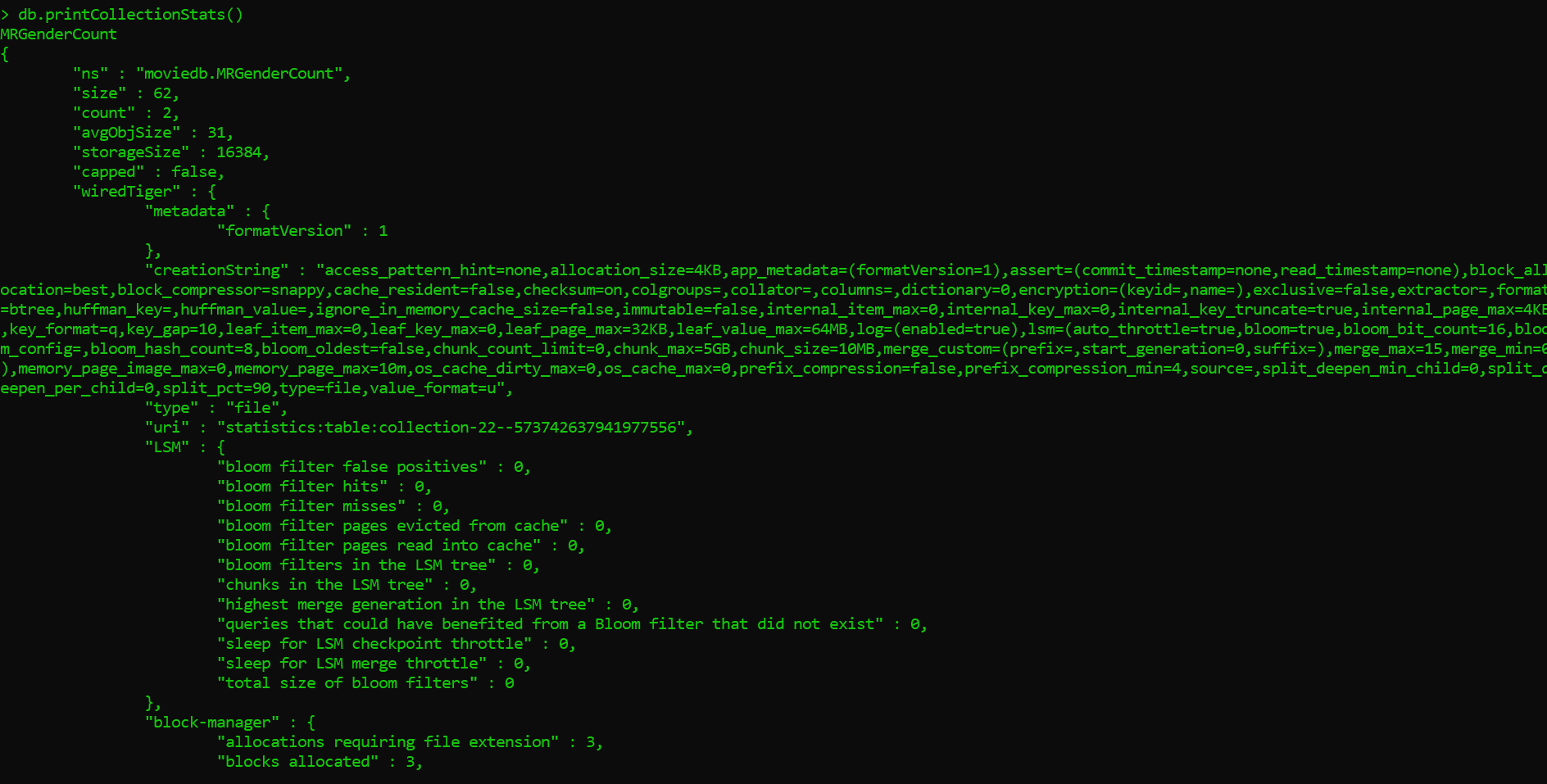
**2. db.stats()**

**3. db.version()**

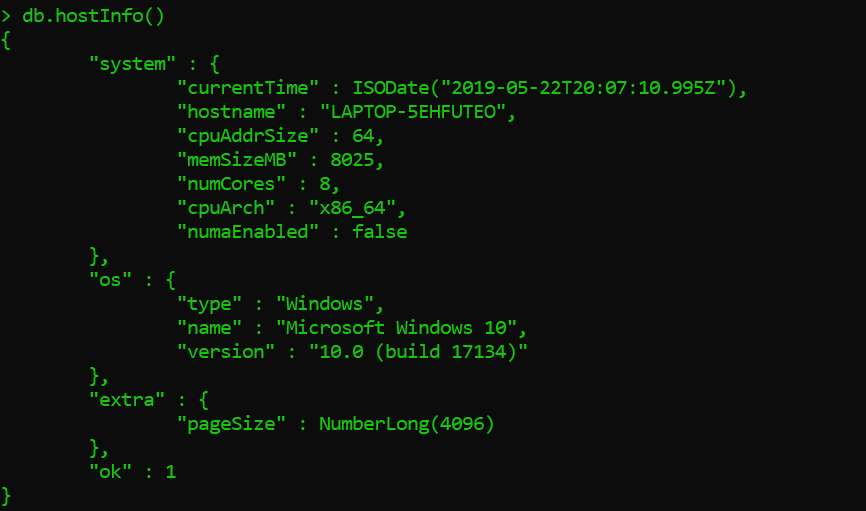
**All 3 screenshots below**



**4. db.printCollectionStats()**



**5. db.hostInfo()**

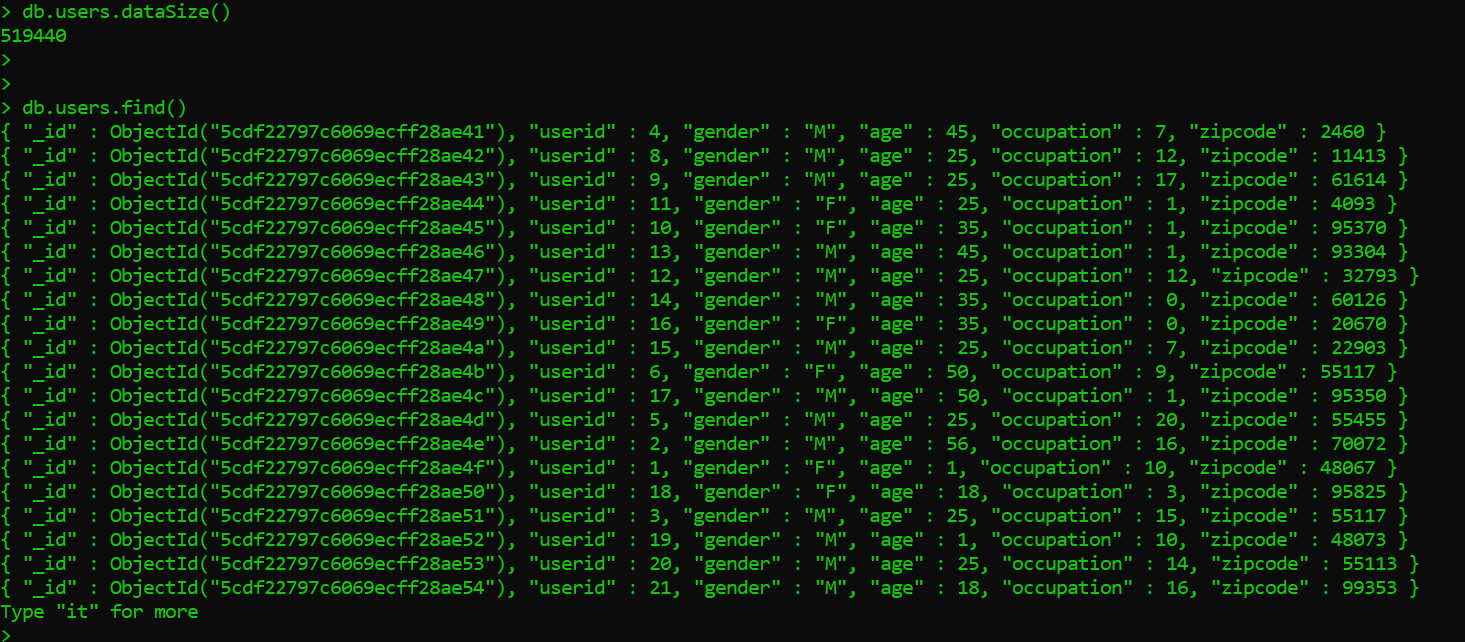


**C: mongo> db.mycoll.help()  [5 commands]**

**1. db.users.dataSize()**

**2. db.users.find()**

**All 2 screenshots below**



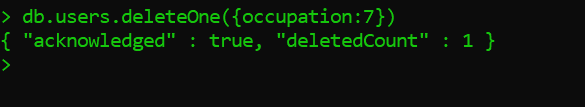
**3. db.users.findOne()**

**4. db.users.getIndexes()**

**All 2 screenshots below**

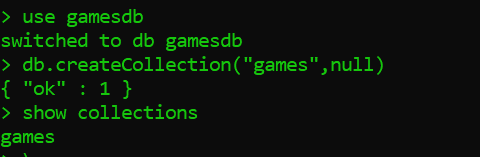


**5. db.mycoll.deleteOne( filter, <optional params> )**



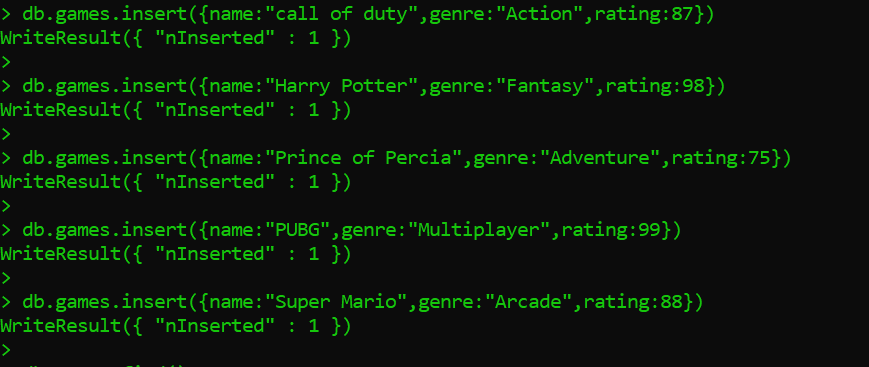
**PART 7 - PROGRAMMING ASSIGNMENT**

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

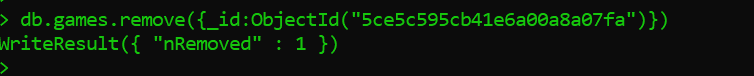


**Add 5 games to the database.**

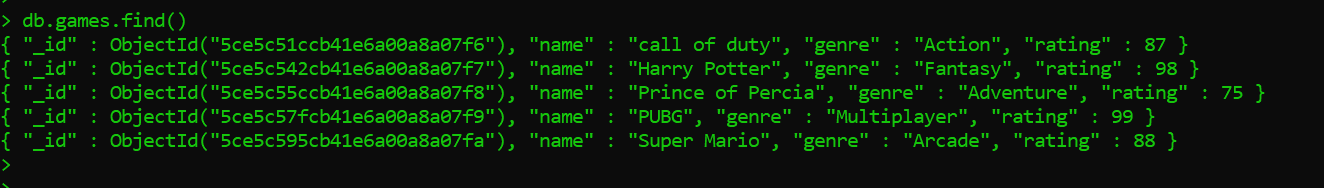
**Give each document the following properties: name, genre, rating (out of 100)**



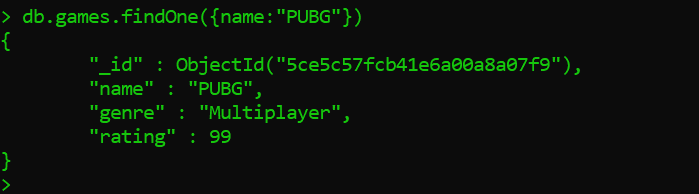
**If you make some mistakes and want to clean it out, use remove() on your collection.**



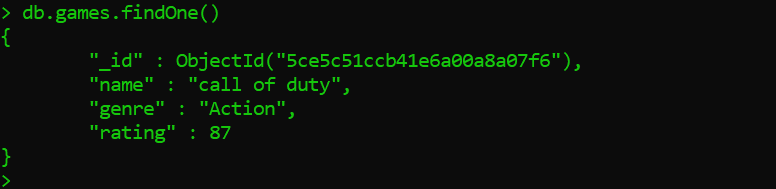
**Write a query that returns all the games.**



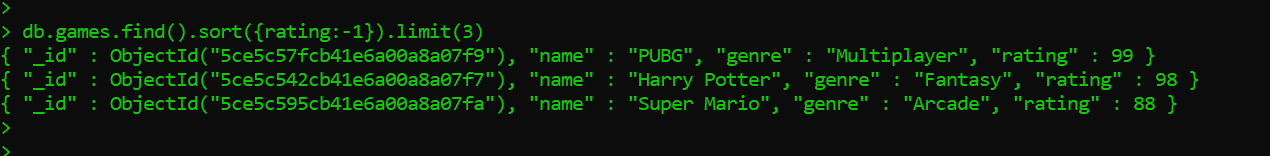
**Write a query to find one of your games by name without using limit().**



**Use the findOne method. Look how much nicer it’s formatted!**



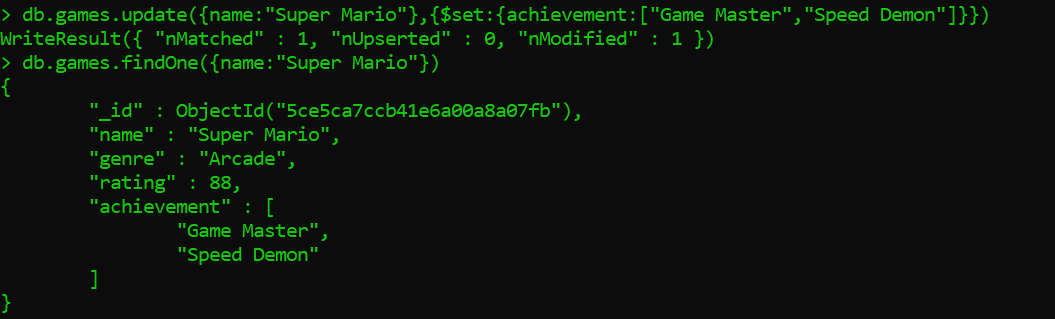
**Write a query that returns the 3 highest rated games.**



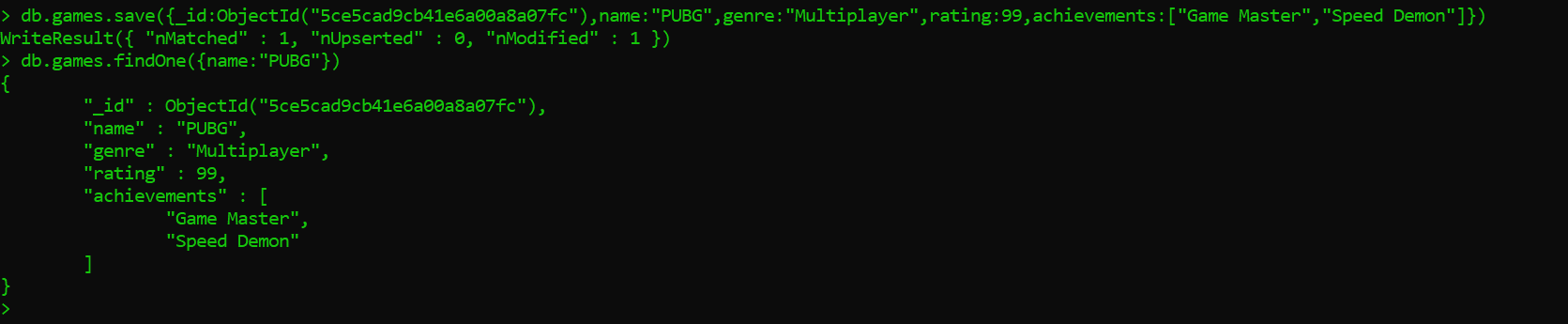
**Update your two favorite games to have two achievements called ‘Game Master’ and ‘Speed Demon’, each under a single key.**

**Show two ways to do this. Do the first using update() and do the second using save(). Hint: for save, you might want to query the object and store it in a variable first.**

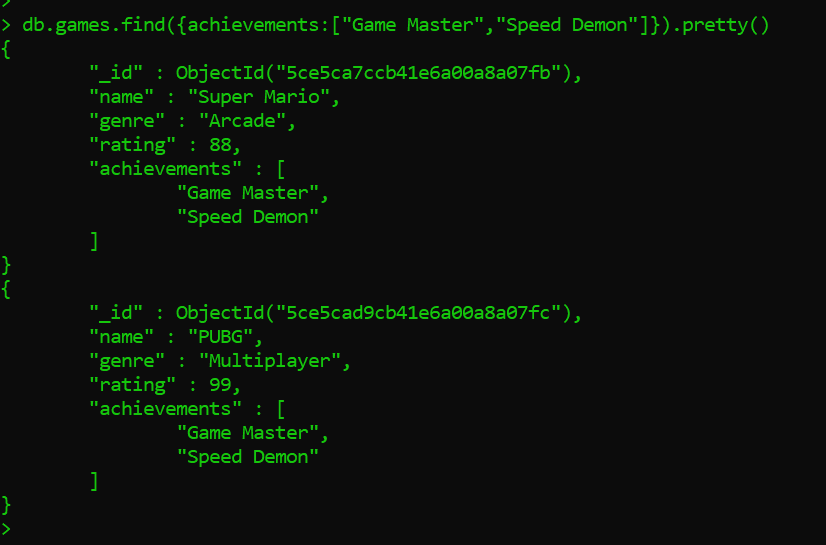
**Update()**



**Save()**



**Write a query that returns all the games that have both the ‘Game Master’ and the ‘Speed Demon’ achievements.**



**Write a query that returns only games that have achievements. Not all of your games should have achievements, obviously.**



**You could take the screenshots by pressing ALT + PRT SCRN or Snipping Tool every time you execute a command, and paste into a word document. You could then submit this document.**