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/

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\ny
se.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:

```
> map
function (){
        emit({stock_name:this.stock_symbol},{average_price_high:this.stock_price_high});
}
> reduce
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 };
}
>
```

```
> 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
>
```

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)

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-

```
> db.stocks.mapReduce(map,reduce,{out:"CorrectAvgStockPrice",finalize: finalize})
{
        "result" : "CorrectAvgStockPrice",
        "timeMillis" : 216440,
        "counts" : {
                "input" : 9242031,
                "emit" : 9242031,
                "reduce" : 95328,
                "output" : 2853
        },
        "ok" : 1
}
> db.CorrectAvgStockPrice.find()
{ "_id" : { "stock_name" : NaN }, "value" : { "total_price_high" : 38560.66000000021, "stock_count" : 2687, "average_stock_price_high" : 14.350822478600747 } }
{ "_id" : { "stock_name" : "AA" }, "value" : { "total_price_high" : 635234.290000002, "stock_count" : 12109, "average_stock_price_high" : 52.459682054670246 } }
{ "_id" : { "stock_name" : "AAI" }, "value" : { "total_price_high" : 41369.05000000041, "stock_count" : 3933, "average_stock_price_high" : 10.518446478515234 } }
{ "_id" : { "stock_name" : "AAN" }, "value" : { "total_price_high" : 83717.14999999902, "stock_count" : 4218, "average_stock_price_high" : 19.847593646277623 } }
{ "_id" : { "stock_name" : "AAP" }, "value" : { "total_price_high" : 92036.46000000006, "stock_count" : 2058, "average_stock_price_high" : 44.7213119533528 } }
{ "_id" : { "stock_name" : "AAR" }, "value" : { "total_price_high" : 50557.92000000018, "stock_count" : 2632, "average_stock_price_high" : 19.208936170212834 } }
{ "_id" : { "stock_name" : "AAV" }, "value" : { "total_price_high" : 17935.320000000018, "stock_count" : 1435, "average_stock_price_high" : 12.498480836236947 } }
{ "_id" : { "stock_name" : "AB" }, "value" : { "total_price_high" : 168401.26999999973, "stock_count" : 5495, "average_stock_price_high" : 30.6462729543216 } }
{ "_id" : { "stock_name" : "ABA" }, "value" : { "total_price_high" : 23550.989999999998, "stock_count" : 906, "average_stock_price_high" : 25.994470198675494 } }
{ "_id" : { "stock_name" : "ABB" }, "value" : { "total_price_high" : 27948.20000000001, "stock_count" : 2221, "average_stock_price_high" : 12.583610986042329 } }
{ "_id" : { "stock_name" : "ABC" }, "value" : { "total_price_high" : 178398.48000000106, "stock_count" : 3733, "average_stock_price_high" : 47.7895740691136 } }
{ "_id" : { "stock_name" : "ABD" }, "value" : { "total_price_high" : 17718.600000000013, "stock_count" : 1127, "average_stock_price_high" : 15.721916592724059 } }
{ "_id" : { "stock_name" : "ABG" }, "value" : { "total_price_high" : 30611.2300000001, "stock_count" : 1984, "average_stock_price_high" : 15.429047379032308 } }
{ "_id" : { "stock_name" : "ABK" }, "value" : { "total_price_high" : 240267.1700000026, "stock_count" : 4682, "average_stock_price_high" : 51.31720845792452 } }
{ "_id" : { "stock_name" : "ABM" }, "value" : { "total_price_high" : 158110.459999997, "stock_count" : 6446, "average_stock_price_high" : 24.528461061122712 } }
{ "_id" : { "stock_name" : "ABR" }, "value" : { "total_price_high" : 26724.670000000002, "stock_count" : 1450, "average_stock_price_high" : 18.430806896551726 } }
{ "_id" : { "stock_name" : "ABT" }, "value" : { "total_price_high" : 326329.2000000044, "stock_count" : 6772, "average_stock_price_high" : 48.188009450679914 } }
{ "_id" : { "stock_name" : "ABV" }, "value" : { "total_price_high" : 105043.44000000024, "stock_count" : 3284, "average_stock_price_high" : 31.986431181486065 } }
{ "_id" : { "stock_name" : "ABVT" }, "value" : { "total_price_high" : 74779.0200000001, "stock_count" : 1520, "average_stock_price_high" : 49.196723684210596 } }
{ "_id" : { "stock_name" : "ABX" }, "value" : { "total_price_high" : 142971.01000000082, "stock_count" : 6303, "average_stock_price_high" : 22.683009677931274 } }
Type "it" for more
>
```

Function Screenshots:

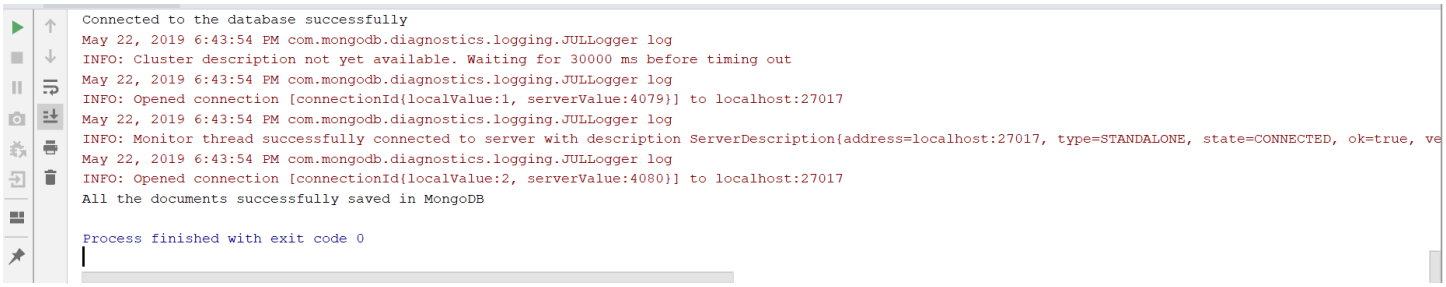
```
> map
function (){
        emit({stock_name:this.stock_symbol},{total_price_high:this.stock_price_high, stock_count:1});
}
> reduce
function (key,values){
        var count= 0;
        var sum = 0;
        for(var i = 0; i<values.length;i++){
        sum += values[i].total_price_high;
        count +=values[i].stock_count;
        }


return {total_price_high:sum, stock_count:count};
}
> finalize
function (key, reducedVal) {

  reducedVal.average_stock_price_high = reducedVal.total_price_high/reducedVal.stock_count;

  return reducedVal;

}
>
```

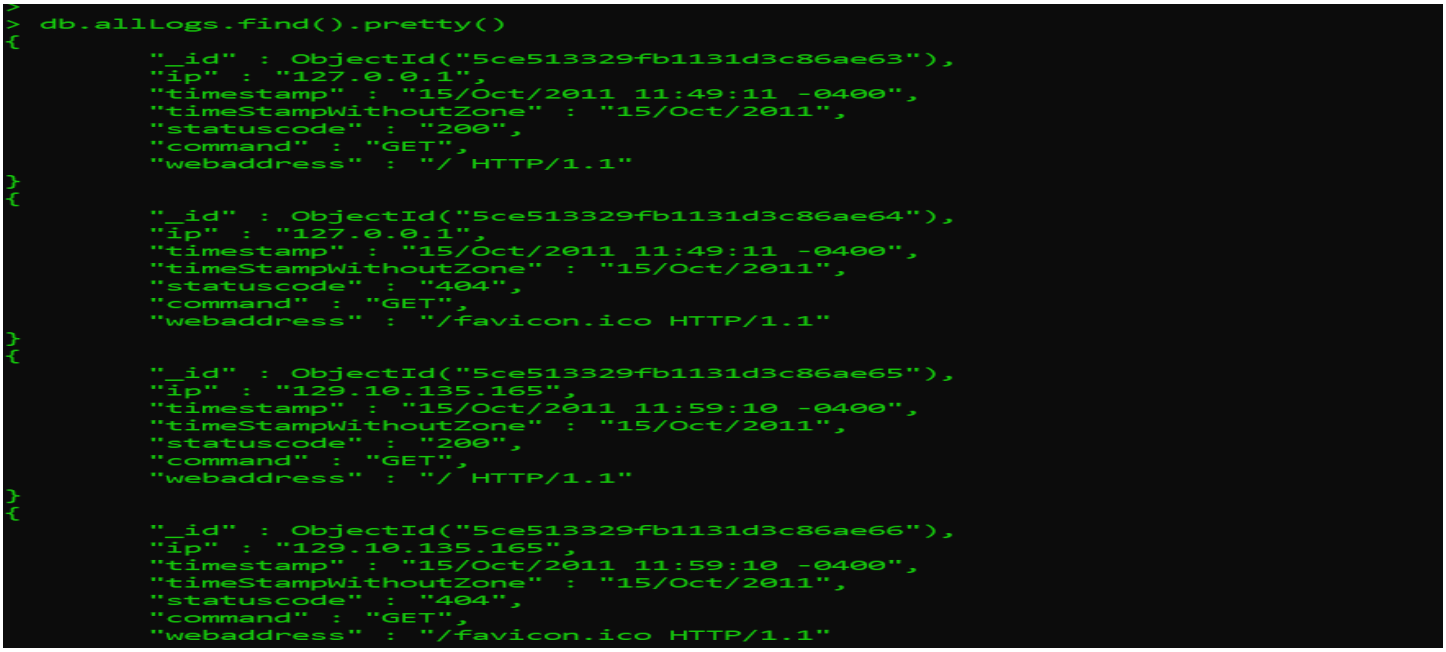
PART 4. Write a console application (or Swing Application), to read and insert the access.log file into MongoDB. This application will only run once to insert the log file into MongoDB. Once the documents are inserted into MongoDB, perform MapReduce for each of the followings:

Answer:

The java application is added in the assignment folder.

Name of Java File- LogUpload.java





a) **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

```
> db.allLogs.mapReduce(map,reduce,{out:"WebsiteAccess"})
{
        "result" : "WebsiteAccess",
        "timeMillis" : 1241,
        "counts" : {
                "input" : 35111,
                "emit" : 35111,
                "reduce" : 1187,
                "output" : 1945
        },
        "ok" : 1
}
> db.WebsiteAccess.find()
{ "_id" : { "IP_address" : "1.162.207.87" }, "value" : { "count_of_web_access" : 4 } }
{ "_id" : { "IP_address" : "1.170.44.84" }, "value" : { "count_of_web_access" : 83 } }
{ "_id" : { "IP_address" : "1.192.146.100" }, "value" : { "count_of_web_access" : 1 } }
{ "_id" : { "IP_address" : "1.202.184.142" }, "value" : { "count_of_web_access" : 1 } }
{ "_id" : { "IP_address" : "1.202.184.145" }, "value" : { "count_of_web_access" : 1 } }
{ "_id" : { "IP_address" : "1.202.89.134" }, "value" : { "count_of_web_access" : 2 } }
{ "_id" : { "IP_address" : "1.234.2.41" }, "value" : { "count_of_web_access" : 12 } }
{ "_id" : { "IP_address" : "1.56.79.5" }, "value" : { "count_of_web_access" : 4 } }
{ "_id" : { "IP_address" : "1.59.91.151" }, "value" : { "count_of_web_access" : 4 } }
{ "_id" : { "IP_address" : "1.62.189.221" }, "value" : { "count_of_web_access" : 4 } }
{ "_id" : { "IP_address" : "1.85.17.247" }, "value" : { "count_of_web_access" : 1 } }
{ "_id" : { "IP_address" : "10.15.10.129" }, "value" : { "count_of_web_access" : 2812 } }
{ "_id" : { "IP_address" : "10.15.10.135" }, "value" : { "count_of_web_access" : 2108 } }
{ "_id" : { "IP_address" : "10.15.10.144" }, "value" : { "count_of_web_access" : 2 } }
{ "_id" : { "IP_address" : "10.15.10.151" }, "value" : { "count_of_web_access" : 4 } }
{ "_id" : { "IP_address" : "10.15.11.112" }, "value" : { "count_of_web_access" : 2 } }
{ "_id" : { "IP_address" : "10.15.8.173" }, "value" : { "count_of_web_access" : 3 } }
{ "_id" : { "IP_address" : "10.15.8.20" }, "value" : { "count_of_web_access" : 5 } }
{ "_id" : { "IP_address" : "10.15.8.23" }, "value" : { "count_of_web_access" : 3 } }
{ "_id" : { "IP_address" : "10.15.8.250" }, "value" : { "count_of_web_access" : 7 } }
Type "it" for more
>
```

b) 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

```
> db.allLogs.mapReduce(map,reduce,{out:"LatestAccess"})
{
        "result" : "LatestAccess",
        "timeMillis" : 603,
        "counts" : {
                "input" : 35111,
                "emit" : 35111,
                "reduce" : 1202,
                "output" : 1945
        },
        "ok" : 1
}
> db.LatestAccess.find()
{ "_id" : { "IP_address" : "1.162.207.87" }, "value" : { "latest_access" : "28/Apr/2012 07:04:51 -0400" } }
{ "_id" : { "IP_address" : "1.170.44.84" }, "value" : { "latest_access" : "20/Mar/2012 18:29:30 -0400" } }
{ "_id" : { "IP_address" : "1.192.146.100" }, "value" : { "latest_access" : "05/Sep/2012 11:18:10 -0400" } }
{ "_id" : { "IP_address" : "1.202.184.142" }, "value" : { "latest_access" : "02/May/2012 15:45:43 -0400" } }
{ "_id" : { "IP_address" : "1.202.184.145" }, "value" : { "latest_access" : "05/Apr/2012 20:53:12 -0400" } }
{ "_id" : { "IP_address" : "1.202.89.134" }, "value" : { "latest_access" : "21/Oct/2011 20:06:49 -0400" } }
{ "_id" : { "IP_address" : "1.234.2.41" }, "value" : { "latest_access" : "27/Dec/2011 05:50:18 -0500" } }
{ "_id" : { "IP_address" : "1.56.79.5" }, "value" : { "latest_access" : "28/Apr/2012 01:36:34 -0400" } }
{ "_id" : { "IP_address" : "1.59.91.151" }, "value" : { "latest_access" : "27/Apr/2012 20:51:41 -0400" } }
{ "_id" : { "IP_address" : "1.62.189.221" }, "value" : { "latest_access" : "28/Apr/2012 03:17:25 -0400" } }
{ "_id" : { "IP_address" : "1.85.17.247" }, "value" : { "latest_access" : "26/Apr/2012 15:24:26 -0400" } }
{ "_id" : { "IP_address" : "10.15.10.129" }, "value" : { "latest_access" : "07/Mar/2013 00:50:26 -0500" } }
{ "_id" : { "IP_address" : "10.15.10.135" }, "value" : { "latest_access" : "06/Mar/2013 19:12:34 -0500" } }
{ "_id" : { "IP_address" : "10.15.10.144" }, "value" : { "latest_access" : "02/May/2013 07:10:06 -0400" } }
{ "_id" : { "IP_address" : "10.15.10.151" }, "value" : { "latest_access" : "02/May/2013 05:39:07 -0400" } }
{ "_id" : { "IP_address" : "10.15.11.112" }, "value" : { "latest_access" : "21/Mar/2013 03:43:48 -0400" } }
{ "_id" : { "IP_address" : "10.15.8.173" }, "value" : { "latest_access" : "10/Apr/2013 12:09:40 -0400" } }
{ "_id" : { "IP_address" : "10.15.8.20" }, "value" : { "latest_access" : "23/Jul/2013 14:34:10 -0400" } }
{ "_id" : { "IP_address" : "10.15.8.23" }, "value" : { "latest_access" : "29/Mar/2013 06:15:38 -0400" } }
{ "_id" : { "IP_address" : "10.15.8.250" }, "value" : { "latest_access" : "07/Feb/2013 09:13:34 -0500" } }
Type "it" for more
>
```

c) 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

```
> db.allLogs.mapReduce(map,reduce,{out:"CountOfReqType"})
{
        "result" : "CountOfReqType",
        "timeMillis" : 752,
        "counts" : {
                "input" : 35111,
                "emit" : 35111,
                "reduce" : 499,
                "output" : 9
        },
        "ok" : 1
}
> db.CountOfReqType.find()
{ "_id" : { "request_type" : "-" }, "value" : { "count_of_requests" : 27 } }
{ "_id" : { "request_type" : "CONNECT" }, "value" : { "count_of_requests" : 14 } }
{ "_id" : { "request_type" : "GET" }, "value" : { "count_of_requests" : 34034 } }
{ "_id" : { "request_type" : "HEAD" }, "value" : { "count_of_requests" : 830 } }
{ "_id" : { "request_type" : "HELP" }, "value" : { "count_of_requests" : 1 } }
{ "_id" : { "request_type" : "OPTIONS" }, "value" : { "count_of_requests" : 8 } }
{ "_id" : { "request_type" : "POST" }, "value" : { "count_of_requests" : 192 } }
{ "_id" : { "request_type" : "PUT" }, "value" : { "count_of_requests" : 4 } }
{ "_id" : { "request_type" : "\\x16\\x03" }, "value" : { "count_of_requests" : 1 } }
>
```

d) 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

```
> db.allLogs.mapReduce(map,reduce,{out:"CountOfStatusCodes"})
{
        "result" : "CountOfStatusCodes",
        "timeMillis" : 781,
        "counts" : {
                "input" : 35111,
                "emit" : 35111,
                "reduce" : 859,
                "output" : 11
        },
        "ok" : 1
}
> db.CountOfStatusCodes.find()
{ "_id" : { "status_code" : "200" }, "value" : { "count_status_code" : 5002 } }
{ "_id" : { "status_code" : "206" }, "value" : { "count_status_code" : 407 } }
{ "_id" : { "status_code" : "301" }, "value" : { "count_status_code" : 249 } }
{ "_id" : { "status_code" : "302" }, "value" : { "count_status_code" : 6 } }
{ "_id" : { "status_code" : "304" }, "value" : { "count_status_code" : 9816 } }
{ "_id" : { "status_code" : "400" }, "value" : { "count_status_code" : 122 } }
{ "_id" : { "status_code" : "403" }, "value" : { "count_status_code" : 72 } }
{ "_id" : { "status_code" : "404" }, "value" : { "count_status_code" : 19390 } }
{ "_id" : { "status_code" : "405" }, "value" : { "count_status_code" : 18 } }
{ "_id" : { "status_code" : "408" }, "value" : { "count_status_code" : 27 } }
{ "_id" : { "status_code" : "501" }, "value" : { "count_status_code" : 2 } }
```

PART 4 - PROGRAMMING ASSIGNMENT

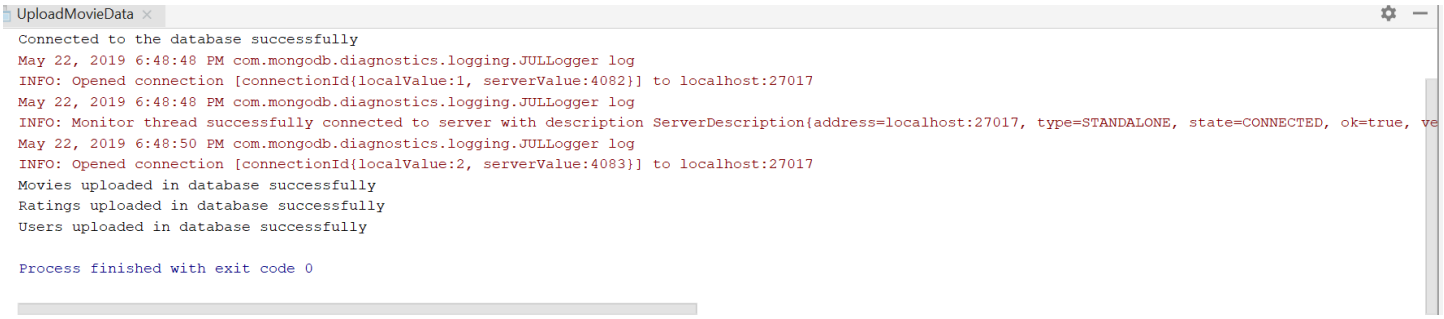
Write a Java (could be a console app - will only run once to import the data into MongoDB) program to read the following file, and insert into 3 different collections (movies, ratings, tags).

http://files.grouplens.org/datasets/movielens/ml-1m.zip

Answer:

The java application is added in the assignment folder.

Name of Java File- UploadMovieData.java




```
> use movieRatingDB
switched to db movieRatingDB
> show collections
MoviesPerGenre
MoviesPerRating
MoviesPerYear
movies
rating
users
> db.movies.findOne()
{
        "_id" : ObjectId("5ce5baf29fb113aabc3eb82c"),
        "movieId" : "1",
        "movieName" : "Toy Story ",
        "releaseYear" : "1995",
        "genre" : [
                "Animation",
                "Children's",
                "Comedy"
        ]
}
> db.rating.findOne()
{
        "_id" : ObjectId("5ce5b7559fb113801c3edc4a"),
        "userId" : "1",
        "movieId" : "1193",
        "rating" : "5",
        "timestamp" : "978300760"
}
> db.users.findOne()
{
        "_id" : ObjectId("5ce5b94d9fb113801c4e1f5b"),
        "userId" : "1",
        "gender" : "F",
        "age" : "1",
        "occupation" : "10",
        "zip" : "48067"
}
>
```

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

a) **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

```
> db.movies.mapReduce(map,reduce,{out:"MoviesPerYear"})
{
        "result" : "MoviesPerYear",
        "timeMillis" : 272,
        "counts" : {
                "input" : 3883,
                "emit" : 3883,
                "reduce" : 95,
                "output" : 348
        },
        "ok" : 1
}
> db.MoviesPerYear.find();
{ "_id" : { "release_year" : "1919" }, "value" : { "number_of_movies" : 2 } }
{ "_id" : { "release_year" : "1920" }, "value" : { "number_of_movies" : 2 } }
{ "_id" : { "release_year" : "1921" }, "value" : { "number_of_movies" : 1 } }
{ "_id" : { "release_year" : "1922" }, "value" : { "number_of_movies" : 1 } }
{ "_id" : { "release_year" : "1923" }, "value" : { "number_of_movies" : 3 } }
{ "_id" : { "release_year" : "1925" }, "value" : { "number_of_movies" : 5 } }
{ "_id" : { "release_year" : "1926" }, "value" : { "number_of_movies" : 8 } }
{ "_id" : { "release_year" : "1927" }, "value" : { "number_of_movies" : 6 } }
{ "_id" : { "release_year" : "1928" }, "value" : { "number_of_movies" : 3 } }
{ "_id" : { "release_year" : "1929" }, "value" : { "number_of_movies" : 3 } }
{ "_id" : { "release_year" : "1930" }, "value" : { "number_of_movies" : 6 } }
{ "_id" : { "release_year" : "1931" }, "value" : { "number_of_movies" : 6 } }
{ "_id" : { "release_year" : "1932" }, "value" : { "number_of_movies" : 7 } }
{ "_id" : { "release_year" : "1933" }, "value" : { "number_of_movies" : 7 } }
{ "_id" : { "release_year" : "1934" }, "value" : { "number_of_movies" : 6 } }
{ "_id" : { "release_year" : "1935" }, "value" : { "number_of_movies" : 6 } }
{ "_id" : { "release_year" : "1936" }, "value" : { "number_of_movies" : 8 } }
{ "_id" : { "release_year" : "1937" }, "value" : { "number_of_movies" : 10 } }
{ "_id" : { "release_year" : "1938" }, "value" : { "number_of_movies" : 6 } }
{ "_id" : { "release_year" : "1939" }, "value" : { "number_of_movies" : 11 } }
Type "it" for more
>
```

b) **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

```
> db.movies.mapReduce(map,reduce,{out:"MoviesPerGenre"})
{
        "result" : "MoviesPerGenre",
        "timeMillis" : 240,
        "counts" : {
                "input" : 3883,
                "emit" : 6408,
                "reduce" : 368,
                "output" : 18
        },
        "ok" : 1
}
> db.MoviesPerGenre.find();
{ "_id" : { "genre" : "Action" }, "value" : { "no_of_movies" : 503 } }
{ "_id" : { "genre" : "Adventure" }, "value" : { "no_of_movies" : 283 } }
{ "_id" : { "genre" : "Animation" }, "value" : { "no_of_movies" : 105 } }
{ "_id" : { "genre" : "Children's" }, "value" : { "no_of_movies" : 251 } }
{ "_id" : { "genre" : "Comedy" }, "value" : { "no_of_movies" : 1200 } }
{ "_id" : { "genre" : "Crime" }, "value" : { "no_of_movies" : 211 } }
{ "_id" : { "genre" : "Documentary" }, "value" : { "no_of_movies" : 127 } }
{ "_id" : { "genre" : "Drama" }, "value" : { "no_of_movies" : 1603 } }
{ "_id" : { "genre" : "Fantasy" }, "value" : { "no_of_movies" : 68 } }
{ "_id" : { "genre" : "Film-Noir" }, "value" : { "no_of_movies" : 44 } }
{ "_id" : { "genre" : "Horror" }, "value" : { "no_of_movies" : 343 } }
{ "_id" : { "genre" : "Musical" }, "value" : { "no_of_movies" : 114 } }
{ "_id" : { "genre" : "Mystery" }, "value" : { "no_of_movies" : 106 } }
{ "_id" : { "genre" : "Romance" }, "value" : { "no_of_movies" : 471 } }
{ "_id" : { "genre" : "Sci-Fi" }, "value" : { "no_of_movies" : 276 } }
{ "_id" : { "genre" : "Thriller" }, "value" : { "no_of_movies" : 492 } }
{ "_id" : { "genre" : "War" }, "value" : { "no_of_movies" : 143 } }
{ "_id" : { "genre" : "Western" }, "value" : { "no_of_movies" : 68 } }
>
```

c) **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

```
> db.rating.mapReduce(map,reduce,{out:"MoviesPerRating"})
{
        "result" : "MoviesPerRating",
        "timeMillis" : 22225,
        "counts" : {
                "input" : 1000209,
                "emit" : 1000209,
                "reduce" : 48105,
                "output" : 5
        },
        "ok" : 1
}
> db.MoviesPerRating.find();
{ "_id" : { "rating" : "1" }, "value" : { "no_of_movies" : 56174 } }
{ "_id" : { "rating" : "2" }, "value" : { "no_of_movies" : 107557 } }
{ "_id" : { "rating" : "3" }, "value" : { "no_of_movies" : 261197 } }
{ "_id" : { "rating" : "4" }, "value" : { "no_of_movies" : 348971 } }
{ "_id" : { "rating" : "5" }, "value" : { "no_of_movies" : 226310 } }
>
```

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

```
> show dbs
FoodieSquad  0.000GB
accessLogDB  0.002GB
admin        0.000GB
config       0.000GB
local        0.000GB
moviedb      0.000GB
nyse         0.696GB
restAPI      0.000GB
resthub      0.000GB
>
```

2. use <db_name>

```
> use moviedb
switched to db moviedb
>
```

3. Show collections

```
> show collections
MRGenderCount
MRGenderCountDes
MRGenderCountDesc
MRGenderCountDescCorrect
MaxAge
users
>
```

4. Show profile

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

5. Show logs

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

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

1. db.getName()

2. db.stats()

3. db.version()

All 3 screenshots below

```
> db.getName()
moviedb
>
> db.stats()
{
        "db" : "moviedb",
        "collections" : 6,
        "views" : 0,
        "objects" : 6050,
        "avgObjSize" : 86.03867768595042,
        "dataSize" : 520534,
        "storageSize" : 282624,
        "numExtents" : 0,
        "indexes" : 6,
        "indexSize" : 176128,
        "fsUsedSize" : 160078467072,
        "fsTotalSize" : 254927695872,
        "ok" : 1
}
>
> db.version()
4.0.4
>
```

4. db.printCollectionStats()


```
> db.printCollectionStats()
MRGenderCount
{
        "ns" : "moviedb.MRGenderCount",
        "size" : 62,
        "count" : 2,
        "avgObjSize" : 31,
        "storageSize" : 16384,
        "capped" : false,
        "wiredTiger" : {
                "metadata" : {
                        "formatVersion" : 1
                },
                "creationString" : "access_pattern_hint=none,allocation_size=4KB,app_metadata=(formatVersion=1),assert=(commit_timestamp=none,read_timestamp=none),block_al
ocation=best,block_compressor=snappy,cache_resident=false,checksum=on,colgroups=,collator=,columns=,dictionary=0,encryption=(keyid=,name=),exclusive=false,extractor=,format
=btree,huffman_key=,huffman_value=,ignore_in_memory_cache_size=false,immutable=false,internal_item_max=0,internal_key_max=0,internal_key_truncate=true,internal_page_max=4K
,key_format=q,key_gap=10,leaf_item_max=0,leaf_key_max=0,leaf_page_max=32KB,leaf_value_max=64MB,log=(enabled=true),lsm=(auto_throttle=true,bloom=true,bloom_bit_count=16,bloo
m_config=,bloom_hash_count=8,bloom_oldest=false,chunk_count_limit=0,chunk_max=5GB,chunk_size=10MB,merge_custom=(prefix=,start_generation=0,suffix=),merge_max=15,merge_min=8
),memory_page_image_max=0,memory_page_max=10m,os_cache_dirty_max=0,os_cache_max=0,prefix_compression=false,prefix_compression_min=4,source=,split_deepen_min_child=0,split_d
eepen_per_child=0,split_pct=90,type=file,value_format=u",
                "type" : "file",
                "uri" : "statistics:table:collection-22--573742637941977556",
                "LSM" : {
                        "bloom filter false positives" : 0,
                        "bloom filter hits" : 0,
                        "bloom filter misses" : 0,
                        "bloom filter pages evicted from cache" : 0,
                        "bloom filter pages read into cache" : 0,
                        "bloom filters in the LSM tree" : 0,
                        "chunks in the LSM tree" : 0,
                        "highest merge generation in the LSM tree" : 0,
                        "queries that could have benefited from a Bloom filter that did not exist" : 0,
                        "sleep for LSM checkpoint throttle" : 0,
                        "sleep for LSM merge throttle" : 0,
                        "total size of bloom filters" : 0
                },
                "block-manager" : {
                        "allocations requiring file extension" : 3,
                        "blocks allocated" : 3,
```


5. db.hostInfo()


```
> db.hostInfo()
{
        "system" : {
                "currentTime" : ISODate("2019-05-22T20:07:10.995Z"),
                "hostname" : "LAPTOP-5EHFUTEO",
                "cpuAddrSize" : 64,
                "memSizeMB" : 8025,
                "numCores" : 8,
                "cpuArch" : "x86_64",
                "numaEnabled" : false
        },
        "os" : {
                "type" : "Windows",
                "name" : "Microsoft Windows 10",
                "version" : "10.0 (build 17134)"
        },
        "extra" : {
                "pageSize" : NumberLong(4096)
        },
        "ok" : 1
}
```

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

1. db.users.dataSize()

2. db.users.find()

All 2 screenshots below

```
> db.users.dataSize()
519440
>
>
> db.users.find()
{ "_id" : ObjectId("5cdf22797c6069ecff28ae41"), "userid" : 4, "gender" : "M", "age" : 45, "occupation" : 7, "zipcode" : 2460 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae42"), "userid" : 8, "gender" : "M", "age" : 25, "occupation" : 12, "zipcode" : 11413 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae43"), "userid" : 9, "gender" : "M", "age" : 25, "occupation" : 17, "zipcode" : 61614 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae44"), "userid" : 11, "gender" : "F", "age" : 25, "occupation" : 1, "zipcode" : 4093 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae45"), "userid" : 10, "gender" : "F", "age" : 35, "occupation" : 1, "zipcode" : 95370 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae46"), "userid" : 13, "gender" : "M", "age" : 45, "occupation" : 1, "zipcode" : 93304 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae47"), "userid" : 12, "gender" : "M", "age" : 25, "occupation" : 12, "zipcode" : 32793 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae48"), "userid" : 14, "gender" : "M", "age" : 35, "occupation" : 0, "zipcode" : 60126 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae49"), "userid" : 16, "gender" : "F", "age" : 35, "occupation" : 0, "zipcode" : 20670 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae4a"), "userid" : 15, "gender" : "M", "age" : 25, "occupation" : 7, "zipcode" : 22903 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae4b"), "userid" : 6, "gender" : "F", "age" : 50, "occupation" : 9, "zipcode" : 55117 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae4c"), "userid" : 17, "gender" : "M", "age" : 50, "occupation" : 1, "zipcode" : 95350 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae4d"), "userid" : 5, "gender" : "M", "age" : 25, "occupation" : 20, "zipcode" : 55455 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae4e"), "userid" : 2, "gender" : "M", "age" : 56, "occupation" : 16, "zipcode" : 70072 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae4f"), "userid" : 1, "gender" : "F", "age" : 1, "occupation" : 10, "zipcode" : 48067 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae50"), "userid" : 18, "gender" : "F", "age" : 18, "occupation" : 3, "zipcode" : 95825 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae51"), "userid" : 3, "gender" : "M", "age" : 25, "occupation" : 15, "zipcode" : 55117 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae52"), "userid" : 19, "gender" : "M", "age" : 1, "occupation" : 10, "zipcode" : 48073 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae53"), "userid" : 20, "gender" : "M", "age" : 25, "occupation" : 14, "zipcode" : 55113 }
{ "_id" : ObjectId("5cdf22797c6069ecff28ae54"), "userid" : 21, "gender" : "M", "age" : 18, "occupation" : 16, "zipcode" : 99353 }
Type "it" for more
>
```

3. db.users.findOne()

4. db.users.getIndexes()

All 2 screenshots below

```
> db.users.findOne()
{
        "_id" : ObjectId("5cdf22797c6069ecff28ae41"),
        "userid" : 4,
        "gender" : "M",
        "age" : 45,
        "occupation" : 7,
        "zipcode" : 2460
}
>
>
> db.users.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "moviedb.users"
        }
]
>
```

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

```
> db.users.deleteOne({occupation:7})
{ "acknowledged" : true, "deletedCount" : 1 }
>
```

PART 7 - PROGRAMMING ASSIGNMENT

Create a collection called 'games'. We're going to put some games in it.

```
> use gamesdb
switched to db gamesdb
> db.createCollection("games",null)
{ "ok" : 1 }
> show collections
games
```

Add 5 games to the database.

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

```
> db.games.insert({name:"call of duty",genre:"Action",rating:87})
WriteResult({ "nInserted" : 1 })
>
> db.games.insert({name:"Harry Potter",genre:"Fantasy",rating:98})
WriteResult({ "nInserted" : 1 })
>
> db.games.insert({name:"Prince of Percia",genre:"Adventure",rating:75})
WriteResult({ "nInserted" : 1 })
>
> db.games.insert({name:"PUBG",genre:"Multiplayer",rating:99})
WriteResult({ "nInserted" : 1 })
>
> db.games.insert({name:"Super Mario",genre:"Arcade",rating:88})
WriteResult({ "nInserted" : 1 })
>
```

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

```
> db.games.remove({_id:ObjectId("5ce5c595cb41e6a00a8a07fa")})
WriteResult({ "nRemoved" : 1 })
>
```

Write a query that returns all the games.

```
> db.games.find()
{ "_id" : ObjectId("5ce5c51ccb41e6a00a8a07f6"), "name" : "call of duty", "genre" : "Action", "rating" : 87 }
{ "_id" : ObjectId("5ce5c542cb41e6a00a8a07f7"), "name" : "Harry Potter", "genre" : "Fantasy", "rating" : 98 }
{ "_id" : ObjectId("5ce5c55ccb41e6a00a8a07f8"), "name" : "Prince of Percia", "genre" : "Adventure", "rating" : 75 }
{ "_id" : ObjectId("5ce5c57fcb41e6a00a8a07f9"), "name" : "PUBG", "genre" : "Multiplayer", "rating" : 99 }
{ "_id" : ObjectId("5ce5c595cb41e6a00a8a07fa"), "name" : "Super Mario", "genre" : "Arcade", "rating" : 88 }
>
```

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

```
> db.games.findOne({name:"PUBG"})
{
        "_id" : ObjectId("5ce5c57fcb41e6a00a8a07f9"),
        "name" : "PUBG",
        "genre" : "Multiplayer",
        "rating" : 99
}
>
```

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

```
> db.games.findOne()
{
        "_id" : ObjectId("5ce5c51ccb41e6a00a8a07f6"),
        "name" : "call of duty",
        "genre" : "Action",
        "rating" : 87
}
>
```

Write a query that returns the 3 highest rated games.

```
>
> db.games.find().sort({rating:-1}).limit(3)
{ "_id" : ObjectId("5ce5c57fcb41e6a00a8a07f9"), "name" : "PUBG", "genre" : "Multiplayer", "rating" : 99 }
{ "_id" : ObjectId("5ce5c542cb41e6a00a8a07f7"), "name" : "Harry Potter", "genre" : "Fantasy", "rating" : 98 }
{ "_id" : ObjectId("5ce5c595cb41e6a00a8a07fa"), "name" : "Super Mario", "genre" : "Arcade", "rating" : 88 }
>
>
```

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()

```
> db.games.update({name:"Super Mario"},{$set:{achievement:["Game Master","Speed Demon"]}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.games.findOne({name:"Super Mario"})
{
        "_id" : ObjectId("5ce5ca7ccb41e6a00a8a07fb"),
        "name" : "Super Mario",
        "genre" : "Arcade",
        "rating" : 88,
        "achievement" : [
                "Game Master",
                "Speed Demon"
        ]
}
```

Save()

```
> db.games.save({_id:ObjectId("5ce5cad9cb41e6a00a8a07fc"),name:"PUBG",genre:"Multiplayer",rating:99,achievements:["Game Master","Speed Demon"]})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.games.findOne({name:"PUBG"})
{
        "_id" : ObjectId("5ce5cad9cb41e6a00a8a07fc"),
        "name" : "PUBG",
        "genre" : "Multiplayer",
        "rating" : 99,
        "achievements" : [
                "Game Master",
                "Speed Demon"
        ]
}
>
```

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

```
> db.games.find({achievements:["Game Master","Speed Demon"]}).pretty()
{
        "_id" : ObjectId("5ce5ca7ccb41e6a00a8a07fb"),
        "name" : "Super Mario",
        "genre" : "Arcade",
        "rating" : 88,
        "achievements" : [
                "Game Master",
                "Speed Demon"
        ]
}
{
        "_id" : ObjectId("5ce5cad9cb41e6a00a8a07fc"),
        "name" : "PUBG",
        "genre" : "Multiplayer",
        "rating" : 99,
        "achievements" : [
                "Game Master",
                "Speed Demon"
        ]
}
>
```

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

```
>
> db.games.find({achievements:{$exists:true}}).pretty()
{
        "_id" : ObjectId("5ce5c51ccb41e6a00a8a07f6"),
        "name" : "call of duty",
        "genre" : "Action",
        "rating" : 87,
        "achievements" : [
                "Gunslinger"
        ]
}
{
        "_id" : ObjectId("5ce5ca7ccb41e6a00a8a07fb"),
        "name" : "Super Mario",
        "genre" : "Arcade",
        "rating" : 88,
        "achievements" : [
                "Game Master",
                "Speed Demon"
        ]
}
{
        "_id" : ObjectId("5ce5cad9cb41e6a00a8a07fc"),
        "name" : "PUBG",
        "genre" : "Multiplayer",
        "rating" : 99,
        "achievements" : [
                "Game Master",
                "Speed Demon"
        ]
}
>
```

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.