Goel, Ajay: Nu Id (001897443)

INFO 7250
Engineering Big-Data Systems
Summer Full 2019
Assignment 2

PART 1 - READING
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/

In file named as : part2_nyse.sh

```
Downloding file
Ajays-MacBook-Pro:Assignment2 ajaygoel$ curl -o /Users/ajaygoel/Documents/Neu/BigData/Assignment2/nyse_zip.zip "http://msis.neu.edu/nyse/nyse.zip"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  123M  100  123M    0     0  4700k      0  0:00:26  0:00:26 --:--:-- 5120k
Ajays-MacBook-Pro:Assignment2 ajaygoel$ if [ "$?" -gt 0 ]
> then
> echo "Error downloading file a. Exiting"
> exit
> fi
Ajays-MacBook-Pro:Assignment2 ajaygoel$ echo "Unzipping the file"
Unzipping the file
Ajays-MacBook-Pro:Assignment2 ajaygoel$ unzip nyse_zip.zip
Archive:  nyse_zip.zip
   creating: NYSE/
  inflating: NYSE/NYSE_daily_prices_0.csv
  inflating: NYSE/NYSE_daily_prices_1.csv
  inflating: NYSE/NYSE_daily_prices_2.csv
  inflating: NYSE/NYSE_daily_prices_3.csv
  inflating: NYSE/NYSE_daily_prices_4.csv
  inflating: NYSE/NYSE_daily_prices_5.csv
  inflating: NYSE/NYSE_daily_prices_6.csv
  inflating: NYSE/NYSE_daily_prices_7.csv
  inflating: NYSE/NYSE_daily_prices_8.csv
```

```
Ajays-MacBook-Pro:Assignment2 ajaygoel$
Ajays-MacBook-Pro:Assignment2 ajaygoel$ for filename in `ls NYSE | grep "NYSE_daily_prices_\([A-Z]\).csv"` ;
> do
> #echo $filename;
> mongoimport -d bigdata_Ass2 -c nyse --type csv --file NYSE/$filename --headerline
>         if [ $? -eq 0 ]; then
>         echo "$filename imported successfully!!!!"
       echo "$filename imported successfullyfor filename in `ls NYSE | grep "NYSE_daily_prices_\([A-Z]\).csv"` ; do mongoimp
ort -d bigdata_Ass2 -c nyse --type csv --file NYSE/$filename --headerline;        if [ $? -eq 0 ]; thenfor filename in `ls N
YSE | grep "NYSE_daily_prices_\([A-Z]\).csv"` ; do mongoimport -d bigdata_Ass2 -c nyse --type csv --file NYSE/$filename --hea
derline;         if [ $? -eq 0 ]; then"
>         else
>         echo "Error while importing $filename!!!"
-bash: !": event not found
>         exit 1
>         fi
> done
2019-05-19T02:29:27.809-0400    connected to: localhost
2019-05-19T02:29:30.799-0400    [#########...............] bigdata_Ass2.nyse   15.5MB/39.1MB (39.6%)
2019-05-19T02:29:33.799-0400    [####################.....] bigdata_Ass2.nyse   31.5MB/39.1MB (80.6%)
2019-05-19T02:29:35.227-0400    [########################] bigdata_Ass2.nyse   39.1MB/39.1MB (100.0%)
2019-05-19T02:29:35.227-0400    imported 735026 documents
NYSE_daily_prices_A.csv imported successfullyfor filename in NYSE_daily_prices_A.csv
NYSE_daily_prices_B.csv
```

```
NYSE_daily_prices_X.csv
NYSE_daily_prices_Y.csv
NYSE_daily_prices_Z.csv ; do mongoimport -d bigdata_Ass2 -c nyse --type csv --file NYSE/NYSE_daily_prices_Z.csv --headerline;
        if [ 0 -eq 0 ]; then
Ajays-MacBook-Pro:Assignment2 ajaygoel$ echo "All files imported sucessfully!!
```

PART 3.1. Use the NYSE database to find the average price of stock_price_high values for each stock using MapReduce.

Map Function:

```
function(){
emit({StockSymbol:this.stock_symbol},
{StockPriceHigh:this.stock_price_high);}
```

Reduce Function:

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

```
db.nyse.mapReduce(map,reduce,{out:"Average_stock_price_high"});
db.Average_stock_price_high.find()
```

```
> db.nyse.mapReduce(map,reduce,{out:"Average_stock_price_high2"});
{
        "result" : "Average_stock_price_high2",
        "timeMillis" : 52102,
        "counts" : {
                "input" : 9211031,
                "emit" : 9211031,
                "reduce" : 94976,
                "output" : 2853
        },
        "ok" : 1
}
> db.Average_stock_price_high2.find()
{ "_id" : { "StockSymbol" : NaN }, "value" : 14.348378412986753 }
{ "_id" : { "StockSymbol" : "AA" }, "value" : 64.26183293176106 }
{ "_id" : { "StockSymbol" : "AAI" }, "value" : 21.928555181348557 }
{ "_id" : { "StockSymbol" : "AAN" }, "value" : 7.965653931219254 }
{ "_id" : { "StockSymbol" : "AAP" }, "value" : 44.73758788349498 }
{ "_id" : { "StockSymbol" : "AAR" }, "value" : 22.84317259255989 }
{ "_id" : { "StockSymbol" : "AAV" }, "value" : 14.410491874957634 }
{ "_id" : { "StockSymbol" : "AB" }, "value" : 4.039483949384046 }
{ "_id" : { "StockSymbol" : "ABA" }, "value" : 25.345595706355322 }
{ "_id" : { "StockSymbol" : "ABB" }, "value" : 17.793833488671158 }
{ "_id" : { "StockSymbol" : "ABC" }, "value" : 23.2265851580881 }
{ "_id" : { "StockSymbol" : "ABD" }, "value" : 25.75419184587634 }
{ "_id" : { "StockSymbol" : "ABG" }, "value" : 17.71440889313131 }
{ "_id" : { "StockSymbol" : "ABK" }, "value" : 24.141800538219215 }
{ "_id" : { "StockSymbol" : "ABM" }, "value" : 22.924588216288598 }
{ "_id" : { "StockSymbol" : "ABR" }, "value" : 19.667373707781202 }
{ "_id" : { "StockSymbol" : "ABT" }, "value" : 45.6898341607357 }
{ "_id" : { "StockSymbol" : "ABV" }, "value" : 11.11770221676533 }
{ "_id" : { "StockSymbol" : "ABVT" }, "value" : 45.69958184091775 }
{ "_id" : { "StockSymbol" : "ABX" }, "value" : 4.8291790775756755 }
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)

Map Function:

```
var map2 = function(){
    emit({stockSymbol:this.stock_symbol},{price:this.stock_price_high,
count:1});}
```

Reduce Function:

```
var reduce2= function (key,values){
var reducedFunc = {price:0,count:0};
    values.forEach(function(val)
    {
        reducedFunc.price+=val.price;
        reducedFunc.count+=val.count;
    });
    return reducedFunc;
}
```

Finalizer function:

```
var finalizer = function(key,reducedFunc){
    reducedFunc.avg = reducedFunc.price/reducedFunc.count;
    return reducedFunc;
}
db.nyse.mapReduce(map2,reduce2,{out:"Average_stock_price_high_finalize
r2",finalize:finalizer})

db.Average_stock_price_high_finalizer2.find()
```

SCREENSHOT : NEXT PAGE

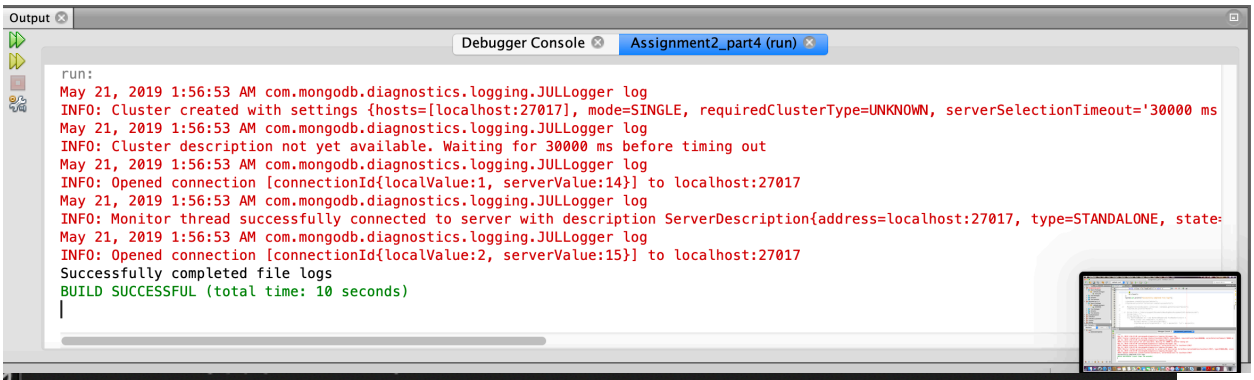
```
> var map2 = function(){
... emit(this.stock_symbol,{price:this.stock_price_high,count:1});}
>
> edit map2
> var reduce2 = function (key,values){
... var reducedFunc = {price:0,count:0};
... values.forEach(function(val)
... {
... reducedFunc.price+=val.price;
... reducedFunc.count+=val.count;
... });
... return reducedFunc;
... }
> edit reduce 2
> edit reduce2
>
>
> var finalizer = function(key,reducedFunc){
... reducedFunc.avg = reducedFunc.price/reducedFunc.count;
... return reducedFunc;
... }
>
> edit map2
> db.nyse.mapReduce(map2,reduce2,{out:"Average_stock_price_high_finalizer2",finalize:finalizer})
{
        "result" : "Average_stock_price_high_finalizer2",
        "timeMillis" : 75516,
        "counts" : {
                "input" : 9211031,
                "emit" : 9211031,
                "reduce" : 94976,
                "output" : 2853
        },
        "ok" : 1
}
> db.Average_stock_price_high_finalizer2.find()
{ "_id" : { "stockSymbol" : NaN }, "value" : { "price" : 38560.66000000021, "count" : 2687, "avg" : 14.350822478600747 } }
{ "_id" : { "stockSymbol" : "AA" }, "value" : { "price" : 635234.290000002, "count" : 12109, "avg" : 52.459682054670246 } }
{ "_id" : { "stockSymbol" : "AAI" }, "value" : { "price" : 41369.05000000041, "count" : 3933, "avg" : 10.518446478515234 } }
{ "_id" : { "stockSymbol" : "AAN" }, "value" : { "price" : 83717.14999999902, "count" : 4218, "avg" : 19.847593646277623 } }
{ "_id" : { "stockSymbol" : "AAP" }, "value" : { "price" : 92036.46000000005, "count" : 2058, "avg" : 44.721311953352796 } }
{ "_id" : { "stockSymbol" : "AAR" }, "value" : { "price" : 50557.92000000018, "count" : 2632, "avg" : 19.208936170212834 } }
{ "_id" : { "stockSymbol" : "AAV" }, "value" : { "price" : 17935.32000000001, "count" : 1435, "avg" : 12.498480836236942 } }
{ "_id" : { "stockSymbol" : "AB" }, "value" : { "price" : 168401.26999999973, "count" : 5495, "avg" : 30.64627297543216 } }
{ "_id" : { "stockSymbol" : "ABA" }, "value" : { "price" : 23550.989999999998, "count" : 906, "avg" : 25.994470198675494 } }
{ "_id" : { "stockSymbol" : "ABB" }, "value" : { "price" : 27948.20000000001, "count" : 2221, "avg" : 12.583610986042329 } }
{ "_id" : { "stockSymbol" : "ABC" }, "value" : { "price" : 178398.48000000106, "count" : 3733, "avg" : 47.7895740691136 } }
{ "_id" : { "stockSymbol" : "ABD" }, "value" : { "price" : 17718.600000000013, "count" : 1127, "avg" : 15.721916592724059 } }
{ "_id" : { "stockSymbol" : "ABG" }, "value" : { "price" : 30611.2300000001, "count" : 1984, "avg" : 15.429047379032308 } }
{ "_id" : { "stockSymbol" : "ABK" }, "value" : { "price" : 240267.1700000026, "count" : 4682, "avg" : 51.31720845792452 } }
{ "_id" : { "stockSymbol" : "ABM" }, "value" : { "price" : 158110.45999999702, "count" : 6446, "avg" : 24.528461061122716 } }
{ "_id" : { "stockSymbol" : "ABR" }, "value" : { "price" : 26724.67, "count" : 1450, "avg" : 18.430806896551722 } }
{ "_id" : { "stockSymbol" : "ABT" }, "value" : { "price" : 326329.2000000044, "count" : 6772, "avg" : 48.188009450679914 } }
{ "_id" : { "stockSymbol" : "ABV" }, "value" : { "price" : 105043.44000000024, "count" : 3284, "avg" : 31.986431181486065 } }
{ "_id" : { "stockSymbol" : "ABVT" }, "value" : { "price" : 74779.0200000001, "count" : 1520, "avg" : 49.196723684210596 } }
{ "_id" : { "stockSymbol" : "ABX" }, "value" : { "price" : 142971.01000000082, "count" : 6303, "avg" : 22.683009677931274 } }
Type "it" for more
>
```

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:

a) Number of times each IP address accessed any web page
b) Latest access date and time from each IP address
c) Find the number of GET, POST, HEAD, etc. requests
d) Find the number of STATUS CODES (404, 200, etc)

Goel, Ajay: Nu Id (001897443)

PART 4



a) Number of times each IP address accessed any web page

Map Function:

```
function (){
        emit({IPAddress:this.IpAddress},{count:1})
;}
```

Reduce Function:

```
function (key,values){
        var sum =0;
        values.forEach(function(val){
                sum+=val.count;
        });
        return {Total_count:sum};
}
db.accessLogs.mapReduce(map4a,reduce4a,{out:"IPAddress_COUNT"});
db.IPAddress_COUNT.find()
```

```
2019-05-21T02:02:42.686-0400 E QUERY    [js] SyntaxError: expected expression, got ';' @(shell):1:2
> var map4a = function(){}
> edit map4a
> var reduce4a = function(){}
> edit reduce4a
>
>
> db.accessLogs.mapReduce(map4a,reduce4a,{out:"IPAddress_COUNT"});
{
        "result" : "IPAddress_COUNT",
        "timeMillis" : 450,
        "counts" : {
                "input" : 35111,
                "emit" : 35111,
                "reduce" : 1173,
                "output" : 1945
        },
        "ok" : 1
}
>
> db.IPAddress_COUNT.find()
{ "_id" : { "IPAddress" : "1.162.207.87" }, "value" : { "Total_count" : 4 } }
{ "_id" : { "IPAddress" : "1.170.44.84" }, "value" : { "Total_count" : 83 } }
{ "_id" : { "IPAddress" : "1.192.146.100" }, "value" : { "count" : 1 } }
{ "_id" : { "IPAddress" : "1.202.184.142" }, "value" : { "count" : 1 } }
{ "_id" : { "IPAddress" : "1.202.184.145" }, "value" : { "count" : 1 } }
{ "_id" : { "IPAddress" : "1.202.89.134" }, "value" : { "Total_count" : 2 } }
{ "_id" : { "IPAddress" : "1.234.2.41" }, "value" : { "Total_count" : 12 } }
{ "_id" : { "IPAddress" : "1.56.79.5" }, "value" : { "Total_count" : 4 } }
{ "_id" : { "IPAddress" : "1.59.91.151" }, "value" : { "Total_count" : 4 } }
{ "_id" : { "IPAddress" : "1.62.189.221" }, "value" : { "Total_count" : 4 } }
{ "_id" : { "IPAddress" : "1.85.17.247" }, "value" : { "count" : 1 } }
{ "_id" : { "IPAddress" : "10.15.10.129" }, "value" : { "Total_count" : NaN } }
{ "_id" : { "IPAddress" : "10.15.10.135" }, "value" : { "Total_count" : NaN } }
{ "_id" : { "IPAddress" : "10.15.10.144" }, "value" : { "Total_count" : 2 } }
{ "_id" : { "IPAddress" : "10.15.10.151" }, "value" : { "Total_count" : 4 } }
{ "_id" : { "IPAddress" : "10.15.11.112" }, "value" : { "Total_count" : 2 } }
{ "_id" : { "IPAddress" : "10.15.8.173" }, "value" : { "Total_count" : 3 } }
{ "_id" : { "IPAddress" : "10.15.8.20" }, "value" : { "Total_count" : 5 } }
{ "_id" : { "IPAddress" : "10.15.8.23" }, "value" : { "Total_count" : 3 } }
{ "_id" : { "IPAddress" : "10.15.8.250" }, "value" : { "Total_count" : 7 } }
Type "it" for more
>
```

b) Latest access date and time from each IP address

MAP FUNCTION:

```
function (){
        emit({IPAddress:this.IpAddress},{timestamp:this.TimeStamp});
}
```

REDUCE FUNCTION:

```
function (key,values){
        var d = new Date(values[0].timestamp);
        var location =0;
        for(var i =1;i<values.length;i++)
        {
                var date = new Date(values[i].timestamp);
                if(d<date){
                        d=date;
                        location=i;
                }
        }
        return {timestamp:values[location].timestamp};
}

db.accessLogs3.mapReduce(map4b,reduce4bb,{out:"ip_latest"});
db.accessLogs3.find().pretty()
```

```
> edit map4b
> db.accessLogs3.mapReduce(map4b,reduce4bb,{out:"ip_latest"});
{
        "result" : "ip_latest",
        "timeMillis" : 536,
        "counts" : {
                "input" : 35111,
                "emit" : 35111,
                "reduce" : 1202,
                "output" : 1945
        },
        "ok" : 1
}
> db.ip_latest.find();
{ "_id" : { "IPAddress" : "1.162.207.87" }, "value" : { "timestamp" : "28/Apr/2012 07:04:51 -0400" } }
{ "_id" : { "IPAddress" : "1.170.44.84" }, "value" : { "timestamp" : "20/Mar/2012 18:29:30 -0400" } }
{ "_id" : { "IPAddress" : "1.192.146.100" }, "value" : { "timestamp" : "05/Sep/2012 11:18:10 -0400" } }
{ "_id" : { "IPAddress" : "1.202.184.142" }, "value" : { "timestamp" : "02/May/2012 15:45:43 -0400" } }
{ "_id" : { "IPAddress" : "1.202.184.145" }, "value" : { "timestamp" : "05/Apr/2012 20:53:12 -0400" } }
{ "_id" : { "IPAddress" : "1.202.89.134" }, "value" : { "timestamp" : "21/Oct/2011 20:06:49 -0400" } }
{ "_id" : { "IPAddress" : "1.234.2.41" }, "value" : { "timestamp" : "27/Dec/2011 05:50:18 -0500" } }
{ "_id" : { "IPAddress" : "1.56.79.5" }, "value" : { "timestamp" : "28/Apr/2012 01:36:34 -0400" } }
{ "_id" : { "IPAddress" : "1.59.91.151" }, "value" : { "timestamp" : "27/Apr/2012 20:51:41 -0400" } }
{ "_id" : { "IPAddress" : "1.62.189.221" }, "value" : { "timestamp" : "28/Apr/2012 03:17:25 -0400" } }
{ "_id" : { "IPAddress" : "1.85.17.247" }, "value" : { "timestamp" : "26/Apr/2012 15:24:26 -0400" } }
{ "_id" : { "IPAddress" : "10.15.10.129" }, "value" : { "timestamp" : "07/Mar/2013 00:50:26 -0500" } }
{ "_id" : { "IPAddress" : "10.15.10.135" }, "value" : { "timestamp" : "06/Mar/2013 19:12:34 -0500" } }
{ "_id" : { "IPAddress" : "10.15.10.144" }, "value" : { "timestamp" : "02/May/2013 07:10:06 -0400" } }
{ "_id" : { "IPAddress" : "10.15.10.151" }, "value" : { "timestamp" : "02/May/2013 05:39:07 -0400" } }
{ "_id" : { "IPAddress" : "10.15.11.112" }, "value" : { "timestamp" : "21/Mar/2013 03:43:48 -0400" } }
{ "_id" : { "IPAddress" : "10.15.8.173" }, "value" : { "timestamp" : "10/Apr/2013 12:09:40 -0400" } }
{ "_id" : { "IPAddress" : "10.15.8.20" }, "value" : { "timestamp" : "23/Jul/2013 14:34:10 -0400" } }
{ "_id" : { "IPAddress" : "10.15.8.23" }, "value" : { "timestamp" : "29/Mar/2013 06:15:38 -0400" } }
{ "_id" : { "IPAddress" : "10.15.8.250" }, "value" : { "timestamp" : "07/Feb/2013 09:13:34 -0500" } }
Type "it" for more
```

c) Find the number of GET, POST, HEAD, etc. requests

Map Function:

```
function (){
        emit({Requests:this.Request},1)
;}
```

Reduce Function:

```
function (key,values){
        var sum =0;
        values.forEach(function(val){
                sum+=val.count;
        });
        return sum;
}
db.accessLogs.mapReduce(map4b,reduce4b,{out:"Requests_COUNT"});
db.Requests_COUNT.find()
```

```
> edit map4b
> edit reduce4b
> db.accessLogs.mapReduce(map4b,reduce4b,{out:"Requests_COUNT"});
{
        "result" : "Requests_COUNT",
        "timeMillis" : 281,
        "counts" : {
                "input" : 35111,
                "emit" : 35111,
                "reduce" : 499,
                "output" : 9
        },
        "ok" : 1
}
> db.Requests_COUNT.find()
{ "_id" : { "Requests" : "-\"" }, "value" : 27 }
{ "_id" : { "Requests" : "CONNECT" }, "value" : 14 }
{ "_id" : { "Requests" : "GET" }, "value" : 34034 }
{ "_id" : { "Requests" : "HEAD" }, "value" : 830 }
{ "_id" : { "Requests" : "HELP\"" }, "value" : 1 }
{ "_id" : { "Requests" : "OPTIONS" }, "value" : 8 }
{ "_id" : { "Requests" : "POST" }, "value" : 192 }
{ "_id" : { "Requests" : "PUT" }, "value" : 4 }
{ "_id" : { "Requests" : "\\x16\\x03\"" }, "value" : 1 }
>
```

d) Find the number of STATUS CODES (404, 200, etc)

Map Function:

```
function (){
        emit({Code:this.Code},{count:1})
;}
```

Reduce Function:

```
function (key,values){
        var sum =0;
        values.forEach(function(val){
                sum+=val.count;
        });
        return {Total_count:sum};
}
db.accessLogs.mapReduce(map4c,reduce4c,{out:"Code_COUNT"});
db.Code_COUNT.find()
```

```
> edit map4c
> edit reduce4c
> db.accessLogs.mapReduce(map4c,reduce4c,{out:"Code_COUNT"});
{
        "result" : "Code_COUNT",
        "timeMillis" : 285,
        "counts" : {
                "input" : 35111,
                "emit" : 35111,
                "reduce" : 868,
                "output" : 14
        },
        "ok" : 1
}
> db.Code_COUNT.find()
{ "_id" : { "Code" : "200" }, "value" : 5000 }
{ "_id" : { "Code" : "206" }, "value" : 407 }
{ "_id" : { "Code" : "227" }, "value" : 2 }
{ "_id" : { "Code" : "301" }, "value" : 249 }
{ "_id" : { "Code" : "302" }, "value" : 6 }
{ "_id" : { "Code" : "304" }, "value" : 9816 }
{ "_id" : { "Code" : "400" }, "value" : 122 }
{ "_id" : { "Code" : "403" }, "value" : 72 }
{ "_id" : { "Code" : "404" }, "value" : 19356 }
{ "_id" : { "Code" : "405" }, "value" : 18 }
{ "_id" : { "Code" : "HTTP/1.0\"" }, "value" : 25 }
{ "_id" : { "Code" : "HTTP/1.1\"" }, "value" : 8 }
{ "_id" : { "Code" : "not_found" }, "value" : 29 }
{ "_id" : { "Code" : "perl" }, "value" : 1 }
>
```

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

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

a. Number of Movies released per year (Movies Collection)
b. Number of Movies per genre (Movies Collection)
c. Number of Movies per rating (Ratings Collection)

Program in Assignment2_part4

```
66
67          database.createCollection("ratings");
68          System.out.println("Collection created successfully");
69
70          MongoCollection<Document> collection2 = database.getCollection("ratings");
```

Output

Delete Project | Assignment2_part7 (run)

```
run:
May 19, 2019 6:57:45 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTim
May 19, 2019 6:57:46 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
May 19, 2019 6:57:46 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:20}] to localhost:27017
May 19, 2019 6:57:46 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=STA
May 19, 2019 6:57:46 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:21}] to localhost:27017
Collection created successfully
Successfully completed file 1 - movies
Collection created successfully
Successfully completed file 2 - ratings
Collection created successfully
Successfully completed file 3 - tags
BUILD SUCCESSFUL (total time: 1 minute 29 seconds)
```

a. Number of Movies released per year (Movies Collection)

Map Function:

```
function (){
        emit({Year:this.year},{count:1})
;}
```

Reduce Function:

```
function (key,values){
        var sum =0;
        values.forEach(function(val){
                sum+=val.count;
        });
        return {Total_movies:sum};
}
```

```
db.movies.mapReduce(map4a,reduce4a,{out:"MoviesPerYear"});
```

```
db.MoviesPerYear.find()
```

```
Type "it" for more
> edit reduce4a
> db.movies.mapReduce(map4a,reduce4a,{out:"MoviesPerYear"});
{
        "result" : "MoviesPerYear",
        "timeMillis" : 155,
        "counts" : {
                "input" : 3883,
                "emit" : 3883,
                "reduce" : 95,
                "output" : 348
        },
        "ok" : 1
}
> db.MoviesPerYear.find()
{ "_id" : { "Year" : "1919" }, "value" : { "Total_movies" : 2 } }
{ "_id" : { "Year" : "1920" }, "value" : { "Total_movies" : 2 } }
{ "_id" : { "Year" : "1921" }, "value" : { "count" : 1 } }
{ "_id" : { "Year" : "1922" }, "value" : { "count" : 1 } }
{ "_id" : { "Year" : "1923" }, "value" : { "Total_movies" : 3 } }
{ "_id" : { "Year" : "1925" }, "value" : { "Total_movies" : 5 } }
{ "_id" : { "Year" : "1926" }, "value" : { "Total_movies" : 8 } }
{ "_id" : { "Year" : "1927" }, "value" : { "Total_movies" : 6 } }
{ "_id" : { "Year" : "1928" }, "value" : { "Total_movies" : 3 } }
{ "_id" : { "Year" : "1929" }, "value" : { "Total_movies" : 3 } }
{ "_id" : { "Year" : "1930" }, "value" : { "Total_movies" : 6 } }
{ "_id" : { "Year" : "1931" }, "value" : { "Total_movies" : 6 } }
{ "_id" : { "Year" : "1932" }, "value" : { "Total_movies" : 7 } }
{ "_id" : { "Year" : "1933" }, "value" : { "Total_movies" : 7 } }
{ "_id" : { "Year" : "1934" }, "value" : { "Total_movies" : 6 } }
{ "_id" : { "Year" : "1935" }, "value" : { "Total_movies" : 6 } }
{ "_id" : { "Year" : "1936" }, "value" : { "Total_movies" : 8 } }
{ "_id" : { "Year" : "1937" }, "value" : { "Total_movies" : 10 } }
{ "_id" : { "Year" : "1938" }, "value" : { "Total_movies" : 6 } }
{ "_id" : { "Year" : "1939" }, "value" : { "Total_movies" : 11 } }
Type "it" for more
>
```

b. Number of Movies per genre (Movies Collection)

Map Function:

```
function (){
        for(var idx=0;idx<this.Genre.length;idx++) {
                var key = this.Genre[idx];
                var value = 1;
        emit(key,value);
}
}
```

Reduce Function:

```
function (key,values){
        var sum=0;
        values.forEach(function(val)
        {
                sum+=val;
        });
        return sum;
}
db.movies.mapReduce(map4b,reduce4b,{out:"Genre_count_2"});
db.Genre_count_2.find()
```

```
> edit map4b
> edit reduce4b
> edit reduce4b
> db.movies2.mapReduce(map4b,reduce4b,{out:"Genre_count_2"});
{
        "result" : "Genre_count_2",
        "timeMillis" : 148,
        "counts" : {
                "input" : 3883,
                "emit" : 6408,
                "reduce" : 368,
                "output" : 18
        },
        "ok" : 1
}
> db.Genre_count_2.find()
{ "_id" : "Action", "value" : 503 }
{ "_id" : "Adventure", "value" : 283 }
{ "_id" : "Animation", "value" : 105 }
{ "_id" : "Children's", "value" : 251 }
{ "_id" : "Comedy", "value" : 1200 }
{ "_id" : "Crime", "value" : 211 }
{ "_id" : "Documentary", "value" : 127 }
{ "_id" : "Drama", "value" : 1603 }
{ "_id" : "Fantasy", "value" : 68 }
{ "_id" : "Film-Noir", "value" : 44 }
{ "_id" : "Horror", "value" : 343 }
{ "_id" : "Musical", "value" : 114 }
{ "_id" : "Mystery", "value" : 106 }
{ "_id" : "Romance", "value" : 471 }
{ "_id" : "Sci-Fi", "value" : 276 }
{ "_id" : "Thriller", "value" : 492 }
{ "_id" : "War", "value" : 143 }
{ "_id" : "Western", "value" : 68 }
>
```

c. Number of Movies per rating (Ratings Collection)

Map Function:

```
var map4c = function (){
        emit({Ratings:this.rating},{count:1});
}
```

Reduce Function:

```
var reduce4c = function (key,values){
        var sum=0;
        values.forEach(function(val){
                sum+=val.count;
        });
        return sum;
}
```

```
db.ratings.mapReduce(map4c,reduce4c,{out:"MoviesPerRating"});
db.MoviesPerRating.find()
```

```
> db.ratings.mapReduce(map4c,reduce4c,{out:"MoviesPerRating"});
{
        "result" : "MoviesPerRating",
        "timeMillis" : 5350,
        "counts" : {
                "input" : 1000209,
                "emit" : 1000209,
                "reduce" : 48105,
                "output" : 5
        },
        "ok" : 1
}
> db.MoviesPerRating.find()
{ "_id" : { "Ratings" : "1" }, "value" : 56174 }
{ "_id" : { "Ratings" : "2" }, "value" : 107557 }
{ "_id" : { "Ratings" : "3" }, "value" : 261197 }
{ "_id" : { "Ratings" : "4" }, "value" : 348971 }
{ "_id" : { "Ratings" : "5" }, "value" : 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.

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

#help

1.Show dbs

```
> show dbs
admin          0.000GB
bigdata_1      0.001GB
bigdata_2      0.724GB
bigdata_Ass2   0.732GB
config         0.000GB
local          0.000GB
moviedb        0.000GB
>
```

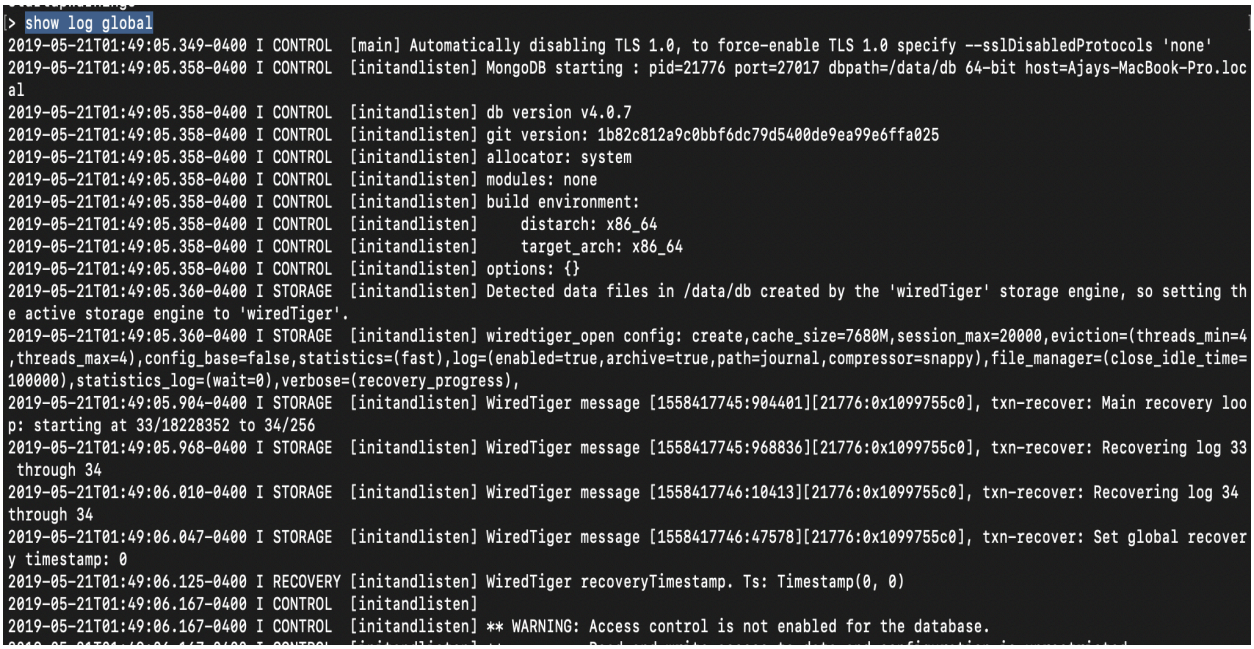
2. Show collections

```
> show collections
Average_stock_price_high
Average_stock_price_high2
Average_stock_price_high_finalizer
Average_stock_price_high_finalizer2
Code_COUNT
GenresPerYear
IPAddress_COUNT
MoviesPerRating
MoviesPerYear
RatingsPerYear
Requests_COUNT
accessLogs
movies
nyse
ratings
tags
```

3. show logs

```
> show logs
global
startupWarnings
```

4. show log global



```
> show log global
2019-05-21T01:49:05.349-0400 I CONTROL  [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2019-05-21T01:49:05.358-0400 I CONTROL  [initandlisten] MongoDB starting : pid=21776 port=27017 dbpath=/data/db 64-bit host=Ajays-MacBook-Pro.loc
al
2019-05-21T01:49:05.358-0400 I CONTROL  [initandlisten] db version v4.0.7
2019-05-21T01:49:05.358-0400 I CONTROL  [initandlisten] git version: 1b82c812a9c0bbf6dc79d5400de9ea99e6ffa025
2019-05-21T01:49:05.358-0400 I CONTROL  [initandlisten] allocator: system
2019-05-21T01:49:05.358-0400 I CONTROL  [initandlisten] modules: none
2019-05-21T01:49:05.358-0400 I CONTROL  [initandlisten] build environment:
2019-05-21T01:49:05.358-0400 I CONTROL  [initandlisten]     distarch: x86_64
2019-05-21T01:49:05.358-0400 I CONTROL  [initandlisten]     target_arch: x86_64
2019-05-21T01:49:05.358-0400 I CONTROL  [initandlisten] options: {}
2019-05-21T01:49:05.360-0400 I STORAGE  [initandlisten] Detected data files in /data/db created by the 'wiredTiger' storage engine, so setting th
e active storage engine to 'wiredTiger'.
2019-05-21T01:49:05.360-0400 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=7680M,session_max=20000,eviction=(threads_min=4
,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=
100000),statistics_log=(wait=0),verbose=(recovery_progress),
2019-05-21T01:49:05.904-0400 I STORAGE  [initandlisten] WiredTiger message [1558417745:904401][21776:0x1099755c0], txn-recover: Main recovery loo
p: starting at 33/18228352 to 34/256
2019-05-21T01:49:05.968-0400 I STORAGE  [initandlisten] WiredTiger message [1558417745:968836][21776:0x1099755c0], txn-recover: Recovering log 33
 through 34
2019-05-21T01:49:06.010-0400 I STORAGE  [initandlisten] WiredTiger message [1558417746:10413][21776:0x1099755c0], txn-recover: Recovering log 34
through 34
2019-05-21T01:49:06.047-0400 I STORAGE  [initandlisten] WiredTiger message [1558417746:47578][21776:0x1099755c0], txn-recover: Set global recover
y timestamp: 0
2019-05-21T01:49:06.125-0400 I RECOVERY [initandlisten] WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
2019-05-21T01:49:06.167-0400 I CONTROL  [initandlisten]
2019-05-21T01:49:06.167-0400 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
```

5. db.accessLogs.find().pretty()

```
> db.accessLogs.find().pretty()
{
        "_id" : ObjectId("5ce393255ea77c551a299863"),
        "IpAddress" : "127.0.0.1",
        "TimeStamp" : "15/Oct/2011:11:49:11 -0400",
        "Request" : "GET",
        "Code" : "200"
}
{
        "_id" : ObjectId("5ce393255ea77c551a299864"),
        "IpAddress" : "127.0.0.1",
        "TimeStamp" : "15/Oct/2011:11:49:11 -0400",
        "Request" : "GET",
        "Code" : "404"
}
{
        "_id" : ObjectId("5ce393255ea77c551a299865"),
        "IpAddress" : "129.10.135.165",
        "TimeStamp" : "15/Oct/2011:11:59:10 -0400",
        "Request" : "GET",
        "Code" : "200"
}
{
        "_id" : ObjectId("5ce393255ea77c551a299866"),
        "IpAddress" : "129.10.135.165",
        "TimeStamp" : "15/Oct/2011:11:59:10 -0400",
        "Request" : "GET",
        "Code" : "404"
}
{
        "_id" : ObjectId("5ce393255ea77c551a299867"),
        "IpAddress" : "129.10.65.240",
        "TimeStamp" : "15/Oct/2011:12:05:58 -0400",
        "Request" : "GET",
        "Code" : "200"
}
```

#db.help

1. db.stats()

```
> db.stats()
{
        "db" : "bigdata_Ass2",
        "collections" : 16,
        "views" : 0,
        "objects" : 10270313,
        "avgObjSize" : 222.40607253157717,
        "dataSize" : 2284179978,
        "storageSize" : 682700800,
        "numExtents" : 0,
        "indexes" : 16,
        "indexSize" : 103796736,
        "fsUsedSize" : 101410512896,
        "fsTotalSize" : 499963174912,
        "ok" : 1
}
```

2. **db.getCollectionNames()**

```
> db.getCollectionNames()
[
        "Average_stock_price_high",
        "Average_stock_price_high2",
        "Average_stock_price_high_finalizer",
        "Average_stock_price_high_finalizer2",
        "Code_COUNT",
        "GenresPerYear",
        "IPAddress_COUNT",
        "MoviesPerRating",
        "MoviesPerYear",
        "RatingsPerYear",
        "Requests_COUNT",
        "accessLogs",
        "movies",
        "nyse",
        "ratings",
        "tags"
]
```

3. db.getMongo()

```
> db.getMongo()
connection to 127.0.0.1:27017
>
```

4. **db.copyDatabase(**‘bigdata_Ass2’,‘archive_Assignment2’**)**

```
> show dbs
admin                0.000GB
archive_Assignment2  0.337GB
bigdata_1            0.001GB
bigdata_2            0.724GB
bigdata_Ass2         0.732GB
config               0.000GB
local                0.000GB
moviedb              0.000GB
> use archive_Assignment2
switched to db archive_Assignment2
> show collections
Average_stock_price_high
Average_stock_price_high2
Average_stock_price_high_finalizer
Average_stock_price_high_finalizer2
Code_COUNT
GenresPerYear
IPAddress_COUNT
MoviesPerRating
MoviesPerYear
RatingsPerYear
Requests_COUNT
accessLogs
movies
nyse
ratings
tags
>
```

5. db.getLogComponents()

```
> db.getLogComponents()
{
        "verbosity" : 0,
        "accessControl" : {
                "verbosity" : -1
        },
        "command" : {
                "verbosity" : -1
        },
        "control" : {
                "verbosity" : -1
        },
        "executor" : {
                "verbosity" : -1
        },
        "geo" : {
                "verbosity" : -1
        },
        "index" : {
                "verbosity" : -1
        },
        "network" : {
                "verbosity" : -1,
                "asio" : {
                        "verbosity" : -1
                },
                "bridge" : {
                        "verbosity" : -1
                }
```

db.mycoll.help()

1. db.GenresPerYear.copyTo("GenresPerYear_copy")

```
> db.createCollection("GenresPerYear_copy")
{ "ok" : 1 }
> db.GenresPerYear_copy.count()
301
> db.GenresPerYear.count()
301
```

2. db.GenresPerYear.getDB()

```
> db.GenresPerYear.getDB()
bigdata_Ass2
>
```

3. db.GenresPerYear_copy.drop()

```
> db.GenresPerYear_copy.drop()
true
```

4. db.movies.dataSize()

```
> db.movies.dataSize()
450833
>
```

5. db.mycoll.latencyStats() – display operation latency histograms for this collection

```
> db.movies.latencyStats().pretty()
{
        "ns" : "bigdata_Ass2.movies",
        "host" : "Ajays-MacBook-Pro.local:27017",
        "localTime" : ISODate("2019-05-21T06:50:00.567Z"),
        "latencyStats" : {
                "reads" : {
                        "latency" : NumberLong(1688),
                        "ops" : NumberLong(2)
                },
                "writes" : {
                        "latency" : NumberLong(0),
                        "ops" : NumberLong(0)
                },
                "commands" : {
                        "latency" : NumberLong(9827),
                        "ops" : NumberLong(2)
                },
                "transactions" : {
                        "latency" : NumberLong(0),
                        "ops" : NumberLong(0)
                }
        }
}
```

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.
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.

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

```
2019-05-21T18:51:50.193-0400 E QUER
@(shell):1:1
> db.createCollection("games")
{ "ok" : 1 }
>
```

INSERTING GAMES:

```
db.games.insertMany([
   { game_name: "Hitman", genre:"Shooter", rating:100 },
   { game_name: "Freedom Fighters", genre:"Adventure", rating:70 },
   { game_name: "Hitman Contracts", genre:"Shooter", rating:95 },
   { game_name: "PUBG", genre:"Arcadia", rating:80 },
   { game_name: "Call of Duty", genre:"Adventure", rating:85 },
   { game_name: "Fifa", genre:"Sports", rating:75 },
   { game_name: "Mario", genre:"Platform", rating:40 },
   { game_name: "Sudoku", genre:"Strategy", rating:87 },
   { game_name: "Clash of Titans", genre:"Action", rating:56 },
   { game_name: "Ludo", genre:"Arcade", rating:77 },
   { game_name: "God of War", genre:"Action", rating:83 },
])
```

```
> db.games.insertMany([
...     { game_name: "Hitman", genre:"Shooter", rating:100 },
...     { game_name: "Freedom Fighters", genre:"Adventure", rating:70 },
...     { game_name: "Hitman Contracts", genre:"Shooter", rating:95 },
...     { game_name: "PUBG", genre:"Arcadia", rating:80 },
...     { game_name: "Call of Duty", genre:"Adventure", rating:85 },
...     { game_name: "Fifa", genre:"Sports", rating:75 },
...     { game_name: "Mario", genre:"Platform", rating:40 },
...     { game_name: "Sudoku", genre:"Strategy", rating:87 },
...     { game_name: "Clash of Titans", genre:"Action", rating:56 },
...     { game_name: "Ludo", genre:"Arcade", rating:77 },
...     { game_name: "God of War", genre:"Action", rating:83 },
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5ce4836252fc5cfaafd33ce7"),
                ObjectId("5ce4836252fc5cfaafd33ce8"),
                ObjectId("5ce4836252fc5cfaafd33ce9"),
                ObjectId("5ce4836252fc5cfaafd33cea"),
                ObjectId("5ce4836252fc5cfaafd33ceb"),
                ObjectId("5ce4836252fc5cfaafd33cec"),
                ObjectId("5ce4836252fc5cfaafd33ced"),
                ObjectId("5ce4836252fc5cfaafd33cee"),
                ObjectId("5ce4836252fc5cfaafd33cef"),
                ObjectId("5ce4836252fc5cfaafd33cf0"),
                ObjectId("5ce4836252fc5cfaafd33cf1")
        ]
}
>
```

Write a query that returns all the games.

`db.games.find().pretty()`

```
> db.games.find().pretty()
{
        "_id" : ObjectId("5ce4836252fc5cfaafd33ce7"),
        "game_name" : "Hitman",
        "genre" : "Shooter",
        "rating" : 100
}
{
        "_id" : ObjectId("5ce4836252fc5cfaafd33ce8"),
        "game_name" : "Freedom Fighters",
        "genre" : "Adventure",
        "rating" : 70
}
{
        "_id" : ObjectId("5ce4836252fc5cfaafd33ce9"),
        "game_name" : "Hitman Contracts",
        "genre" : "Shooter",
        "rating" : 95
}
{
        "_id" : ObjectId("5ce4836252fc5cfaafd33cea"),
        "game_name" : "PUBG",
        "genre" : "Arcadia",
        "rating" : 80
}
```

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

```
db.games.find({"game_name":"Hitman"}).pretty();
```

```
> db.games.find({"game_name":"Hitman"}).pretty();
{
        "_id" : ObjectId("5ce4836252fc5cfaafd33ce7"),
        "game_name" : "Hitman",
        "genre" : "Shooter",
        "rating" : 100
}
>
```

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

```
db.games.findOne()
```

```
{
        "_id" : ObjectId("5ce4836252fc5cfaafd33ce7"),
        "game_name" : "Hitman",
        "genre" : "Shooter",
        "rating" : 100
}
>
```

Write a query that returns the 3 highest rated games.

```
db.games.find().sort({rating:-1}).pretty().limit(3)
```

```
> db.games.find().sort({rating:-1}).pretty().limit(3)
{
        "_id" : ObjectId("5ce4836252fc5cfaafd33ce7"),
        "game_name" : "Hitman",
        "genre" : "Shooter",
        "rating" : 100
}
{
        "_id" : ObjectId("5ce4836252fc5cfaafd33ce9"),
        "game_name" : "Hitman Contracts",
        "genre" : "Shooter",
        "rating" : 95
}
{
        "_id" : ObjectId("5ce4836252fc5cfaafd33cee"),
        "game_name" : "Sudoku",
        "genre" : "Strategy",
        "rating" : 87
}
```

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.

FIRST UPDATE:

```
db.games.update({game_name:"Hitman"},{ game_name: "Hitman",
genre:"Shooter", rating:100, achievements:["Game Master","Speed
Demon"] })
```

```
{
        "_id" : ObjectId("5ce4883d52fc5cfaafd33cf2"),
        "game_name" : "Hitman",
        "genre" : "Shooter",
        "rating" : 100,
        "achievements" : [
                "Game Master",
                "Speed Demon"
        ]
}
```

```
db.games.update({game_name:"Sudoku"}, { $set:{achievements:["Game
Master","Speed Demon"]}})
```

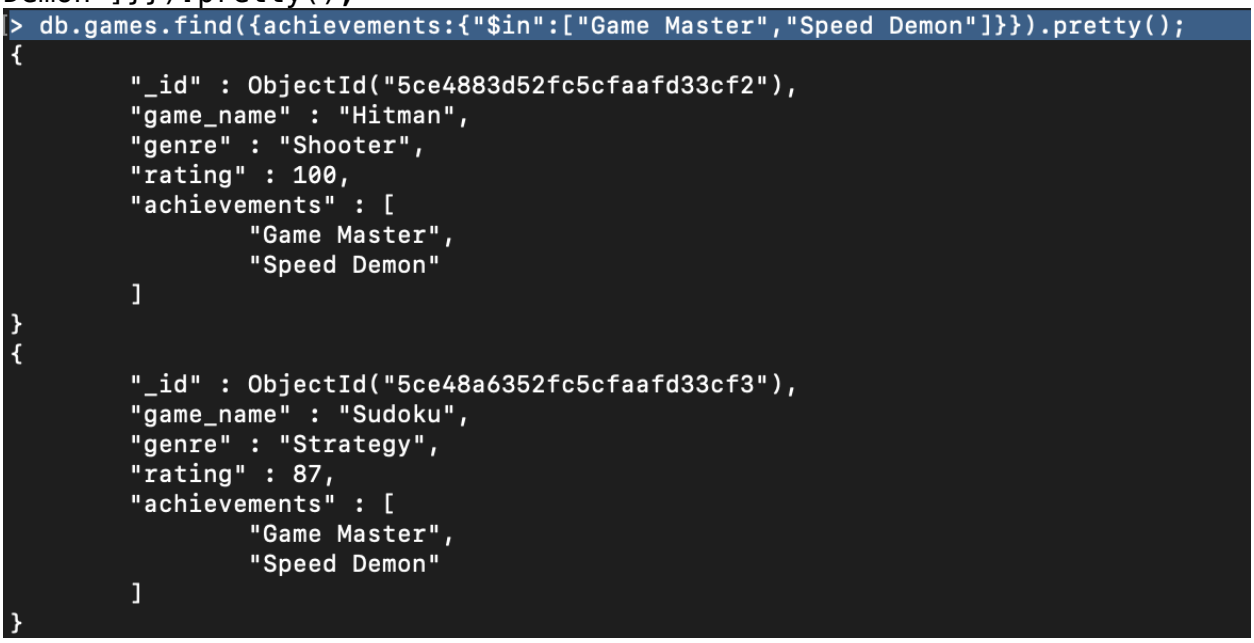
SECOND UPDATE:

```
var upd = db.games.findOne({"game_name":"Sudoku"});
db.games.save({game_name:upd.game_name,genre:upd.genre,rating:upd.rati
ng,achievements:["Game Master","Speed Demon"]})
```

```
> var upd = db.games.findOne({"game_name":"Sudoku"});
> upd
{
        "_id" : ObjectId("5ce48cec52fc5cfaafd33cf4"),
        "game_name" : "Sudoku",
        "genre" : "Strategy",
        "rating" : 87
}
> db.games.save({game_name:upd.game_name,genre:upd.genre,rating:upd.rating,achievements:["Game Master","Speed Demon"]})
WriteResult({ "nInserted" : 1 })
{
        "_id" : ObjectId("5ce4ce789a2cba6cd08f60fe"),
        "game_name" : "Sudoku",
        "genre" : "Strategy",
        "rating" : 87,
        "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:{"$in":["Game Master","Speed Demon"]}}).pretty();
```



```
> db.games.find({achievements:{"$in":["Game Master","Speed Demon"]}}).pretty();
{
        "_id" : ObjectId("5ce4883d52fc5cfaafd33cf2"),
        "game_name" : "Hitman",
        "genre" : "Shooter",
        "rating" : 100,
        "achievements" : [
                "Game Master",
                "Speed Demon"
        ]
}
{
        "_id" : ObjectId("5ce48a6352fc5cfaafd33cf3"),
        "game_name" : "Sudoku",
        "genre" : "Strategy",
        "rating" : 87,
        "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()
```

```
> db.games.find({achievements:{$exists:true}}).pretty()
{
        "_id" : ObjectId("5ce4883d52fc5cfaafd33cf2"),
        "game_name" : "Hitman",
        "genre" : "Shooter",
        "rating" : 100,
        "achievements" : [
                "Game Master",
                "Speed Demon"
        ]
}
{
        "_id" : ObjectId("5ce48a6352fc5cfaafd33cf3"),
        "game_name" : "Sudoku",
        "genre" : "Strategy",
        "rating" : 87,
        "achievements" : [
                "Game Master",
                "Speed Demon"
        ]
}
>
```