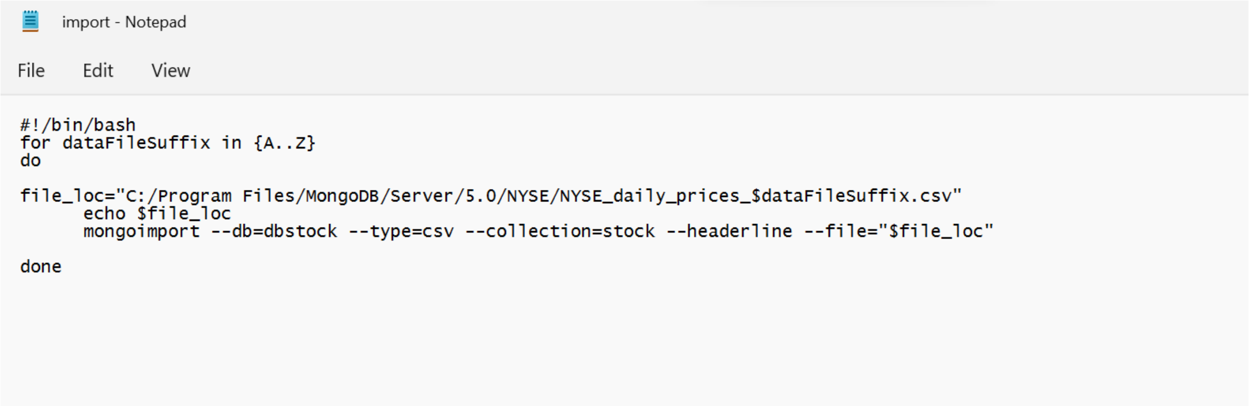
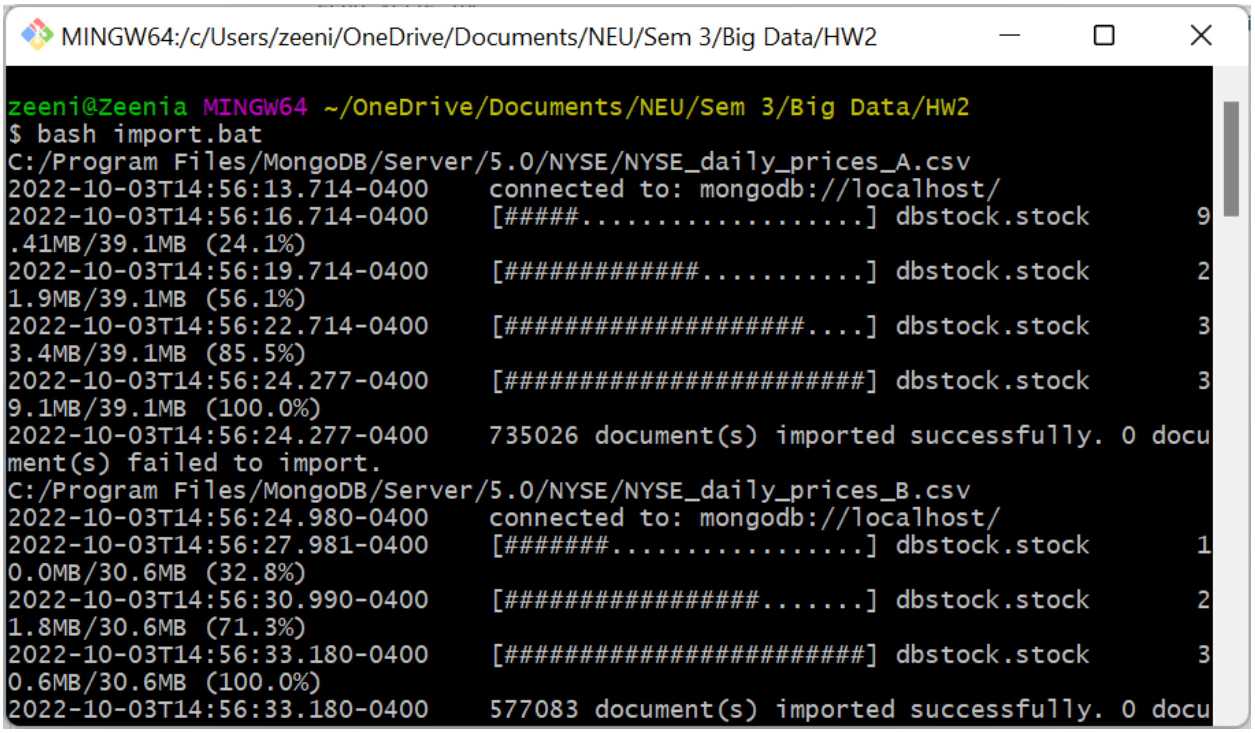
Part-2: Programming Assignment

Write a .bat/.sh to import the entire NYSE dataset (stocks A to Z) into MongoDB.



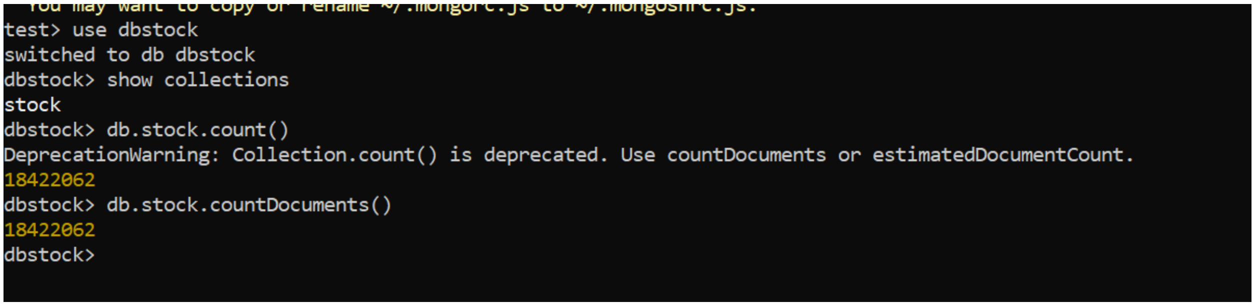
```
#!/bin/bash
for dataFileSuffix in {A..Z}
do

file_loc="C:/Program Files/MongoDB/Server/5.0/NYSE/NYSE_daily_prices_$dataFileSuffix.csv"
      echo $file_loc
      mongoimport --db=dbstock --type=csv --collection=stock --headerline --file="$file_loc"

done
```

Open git bash and execute the command: bash import.bat


MINGW64:/c/Users/zeeni/OneDrive/Documents/NEU/Sem 3/Big Data/HW2

```
zeeni@Zeenia MINGW64 ~/OneDrive/Documents/NEU/Sem 3/Big Data/HW2
$ bash import.bat
C:/Program Files/MongoDB/Server/5.0/NYSE/NYSE_daily_prices_A.csv
2022-10-03T14:56:13.714-0400    connected to: mongodb://localhost/
2022-10-03T14:56:16.714-0400    [#####..................] dbstock.stock       9
.41MB/39.1MB (24.1%)
2022-10-03T14:56:19.714-0400    [############...........] dbstock.stock       2
1.9MB/39.1MB (56.1%)
2022-10-03T14:56:22.714-0400    [###################....] dbstock.stock       3
3.4MB/39.1MB (85.5%)
2022-10-03T14:56:24.277-0400    [#######################] dbstock.stock       3
9.1MB/39.1MB (100.0%)
2022-10-03T14:56:24.277-0400    735026 document(s) imported successfully. 0 docu
ment(s) failed to import.
C:/Program Files/MongoDB/Server/5.0/NYSE/NYSE_daily_prices_B.csv
2022-10-03T14:56:24.980-0400    connected to: mongodb://localhost/
2022-10-03T14:56:27.981-0400    [#######................] dbstock.stock       1
0.0MB/30.6MB (32.8%)
2022-10-03T14:56:30.990-0400    [################.......] dbstock.stock       2
1.8MB/30.6MB (71.3%)
2022-10-03T14:56:33.180-0400    [#######################] dbstock.stock       3
0.6MB/30.6MB (100.0%)
2022-10-03T14:56:33.180-0400    577083 document(s) imported successfully. 0 docu
```

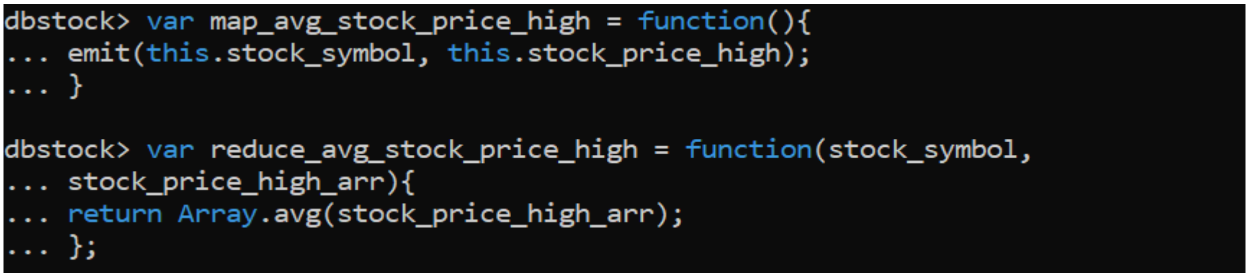
Use following commands to check the new files imported.



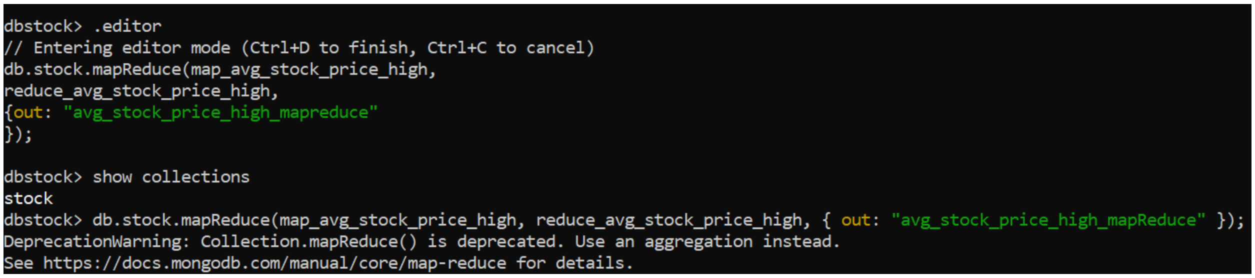
```
  You may want to copy or rename ~/.mongorc.js to ~/.mongoshrc.js.
test> use dbstock
switched to db dbstock
dbstock> show collections
stock
dbstock> db.stock.count()
DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.
18422062
dbstock> db.stock.countDocuments()
18422062
dbstock>
```

Part-3.1:

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

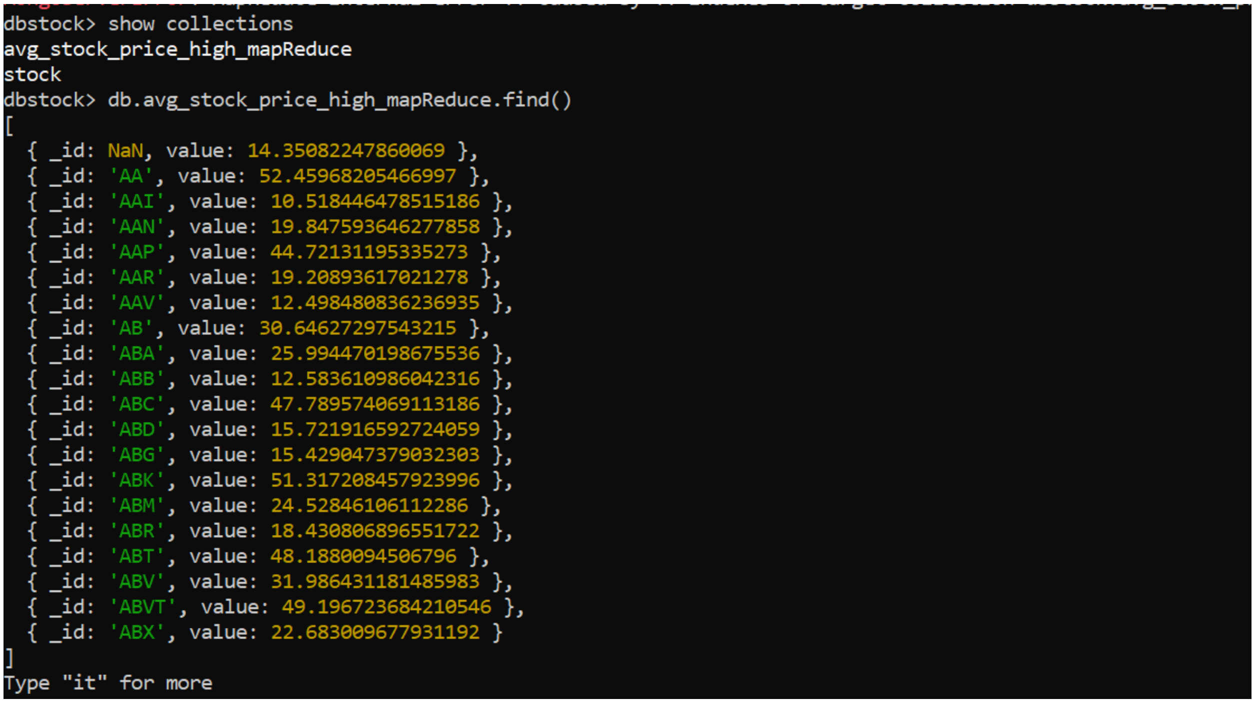


```
dbstock> var map_avg_stock_price_high = function(){
... emit(this.stock_symbol, this.stock_price_high);
... }

dbstock> var reduce_avg_stock_price_high = function(stock_symbol,
... stock_price_high_arr){
... return Array.avg(stock_price_high_arr);
... };
```



```
dbstock> .editor
// Entering editor mode (Ctrl+D to finish, Ctrl+C to cancel)
db.stock.mapReduce(map_avg_stock_price_high,
reduce_avg_stock_price_high,
{out: "avg_stock_price_high_mapreduce"
});

dbstock> show collections
stock
dbstock> db.stock.mapReduce(map_avg_stock_price_high, reduce_avg_stock_price_high, { out: "avg_stock_price_high_mapReduce" });
DeprecationWarning: Collection.mapReduce() is deprecated. Use an aggregation instead.
See https://docs.mongodb.com/manual/core/map-reduce for details.
```

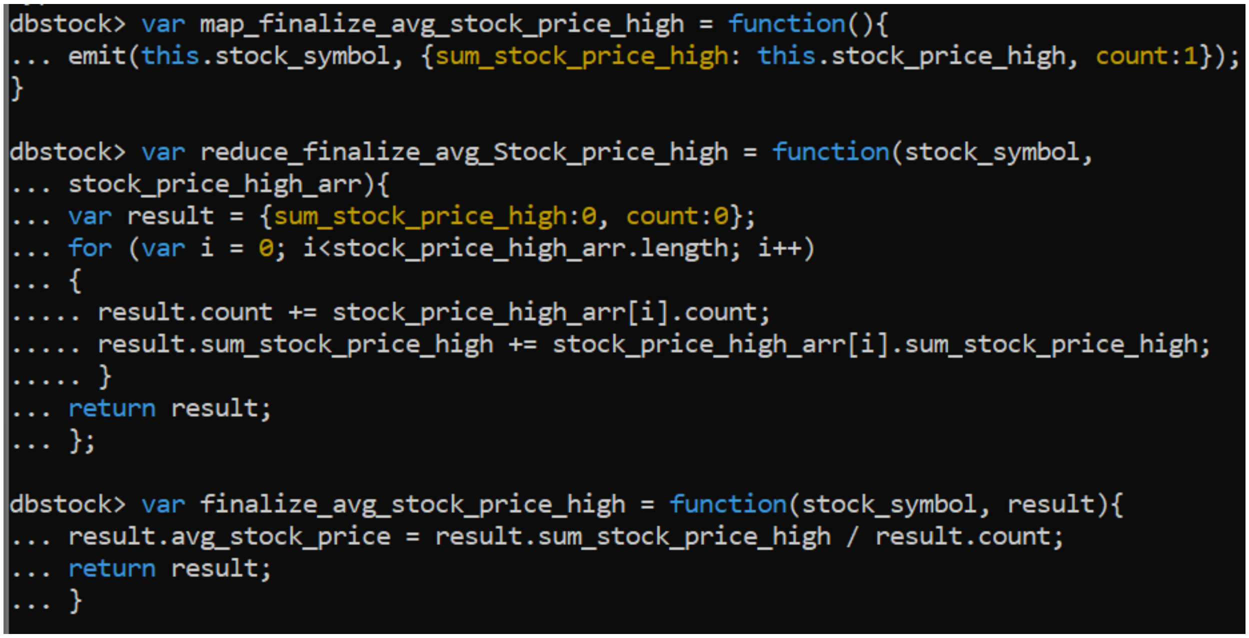
Check the result:



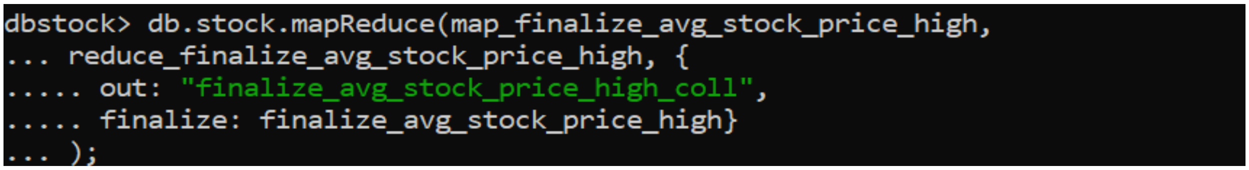
```
dbstock> show collections
avg_stock_price_high_mapReduce
stock
dbstock> db.avg_stock_price_high_mapReduce.find()
[
  { _id: NaN, value: 14.35082247860069 },
  { _id: 'AA', value: 52.45968205466997 },
  { _id: 'AAI', value: 10.518446478515186 },
  { _id: 'AAN', value: 19.847593646277858 },
  { _id: 'AAP', value: 44.72131195335273 },
  { _id: 'AAR', value: 19.20893617021278 },
  { _id: 'AAV', value: 12.498480836236935 },
  { _id: 'AB', value: 30.64627297543215 },
  { _id: 'ABA', value: 25.994470198675536 },
  { _id: 'ABB', value: 12.583610986042316 },
  { _id: 'ABC', value: 47.789574069113186 },
  { _id: 'ABD', value: 15.721916592724059 },
  { _id: 'ABG', value: 15.429047379032303 },
  { _id: 'ABK', value: 51.317208457923996 },
  { _id: 'ABM', value: 24.52846106112286 },
  { _id: 'ABR', value: 18.430806896551722 },
  { _id: 'ABT', value: 48.1880094506796 },
  { _id: 'ABV', value: 31.986431181485983 },
  { _id: 'ABVT', value: 49.196723684210546 },
  { _id: 'ABX', value: 22.683009677931192 }
]
Type "it" for more
```

Part 3.2:

Part 3.1 result will not be correct as AVERAGE is a commutative operation but not associative. Use a FINALIZER to find the correct average.

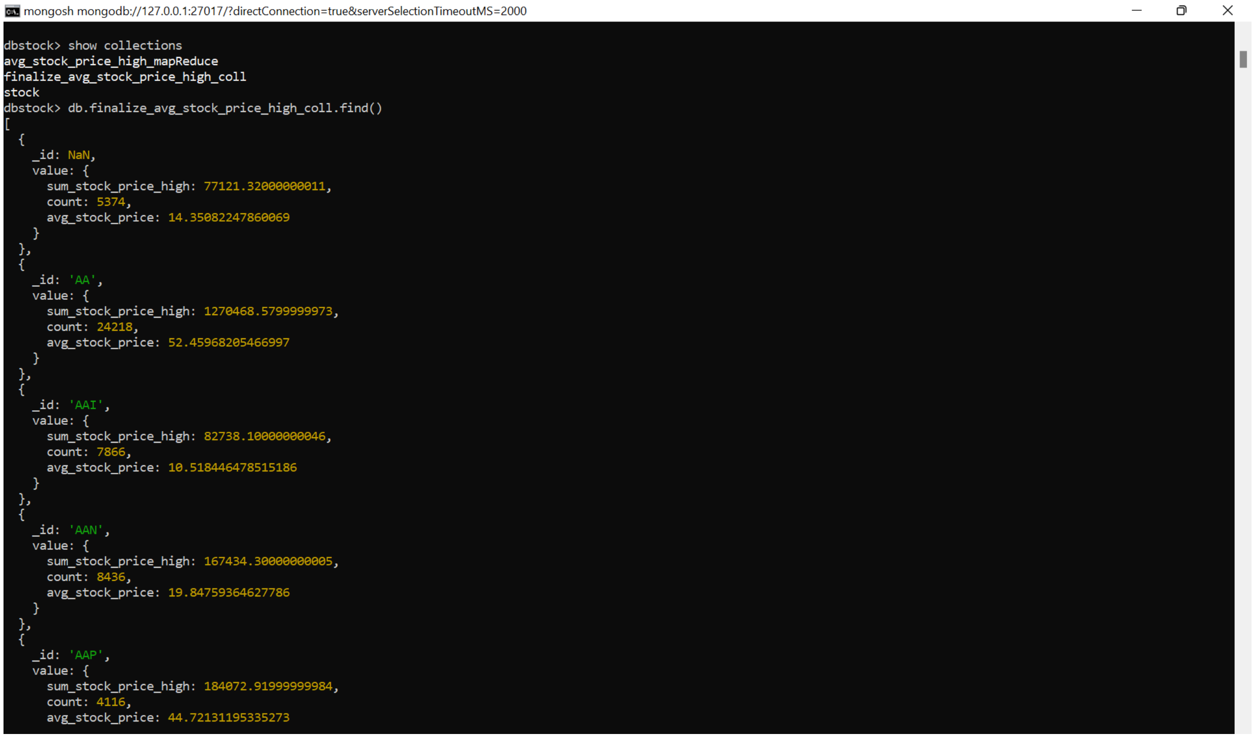


```
dbstock> var map_finalize_avg_stock_price_high = function(){
... emit(this.stock_symbol, {sum_stock_price_high: this.stock_price_high, count:1});
}

dbstock> var reduce_finalize_avg_Stock_price_high = function(stock_symbol,
... stock_price_high_arr){
... var result = {sum_stock_price_high:0, count:0};
... for (var i = 0; i<stock_price_high_arr.length; i++)
... {
..... result.count += stock_price_high_arr[i].count;
..... result.sum_stock_price_high += stock_price_high_arr[i].sum_stock_price_high;
..... }
... return result;
... };

dbstock> var finalize_avg_stock_price_high = function(stock_symbol, result){
... result.avg_stock_price = result.sum_stock_price_high / result.count;
... return result;
... }
```



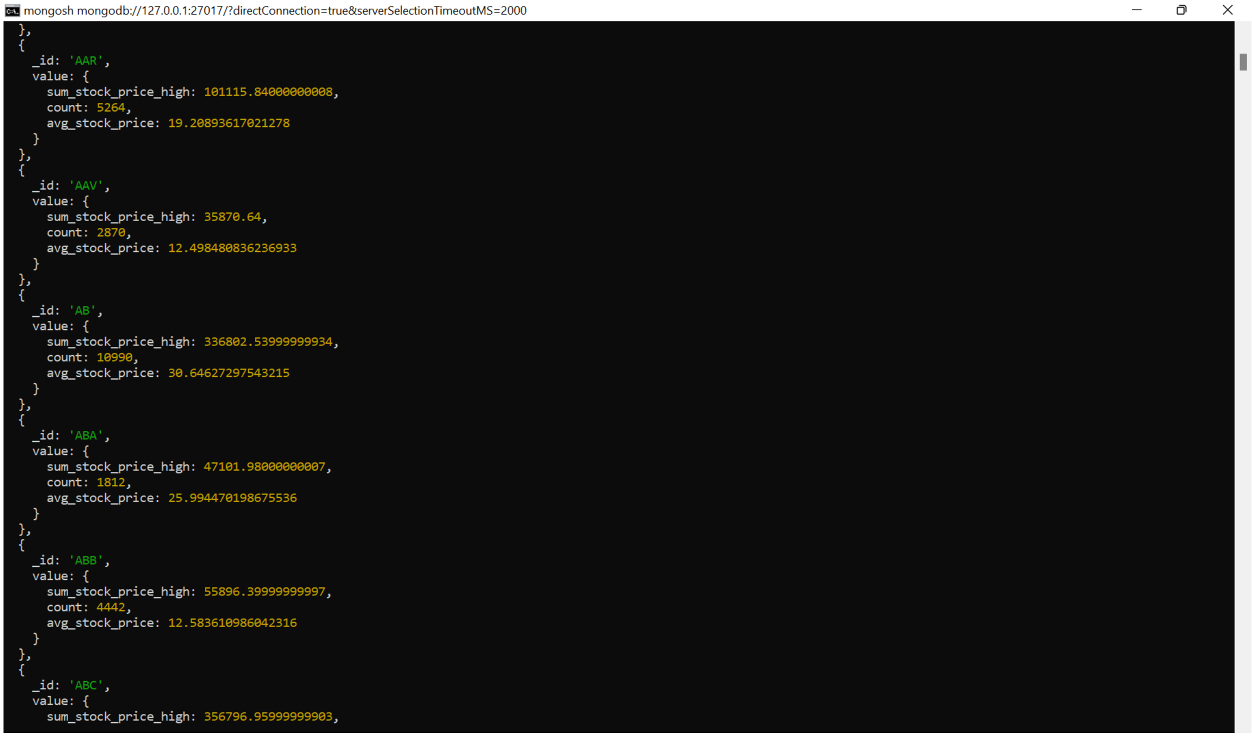
```
dbstock> db.stock.mapReduce(map_finalize_avg_stock_price_high,
... reduce_finalize_avg_stock_price_high, {
..... out: "finalize_avg_stock_price_high_coll",
..... finalize: finalize_avg_stock_price_high}
... );
```

Check the result:



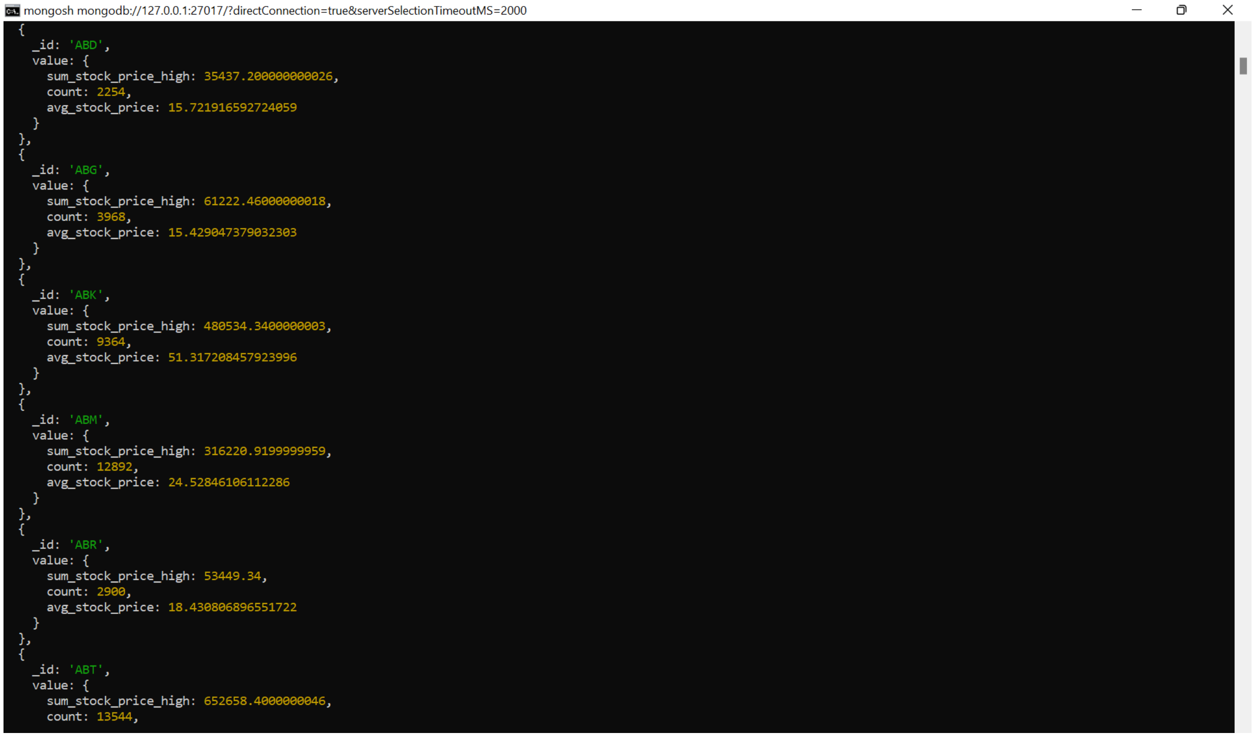
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

```
dbstock> show collections
avg_stock_price_high_mapReduce
finalize_avg_stock_price_high_coll
stock
dbstock> db.finalize_avg_stock_price_high_coll.find()
[
  {
    _id: NaN,
    value: {
      sum_stock_price_high: 77121.32000000011,
      count: 5374,
      avg_stock_price: 14.35082247860069
    }
  },
  {
    _id: 'AA',
    value: {
      sum_stock_price_high: 1270468.5799999973,
      count: 24218,
      avg_stock_price: 52.45968205466997
    }
  },
  {
    _id: 'AAI',
    value: {
      sum_stock_price_high: 82738.10000000046,
      count: 7866,
      avg_stock_price: 10.518446478515186
    }
  },
  {
    _id: 'AAN',
    value: {
      sum_stock_price_high: 167434.30000000005,
      count: 8436,
      avg_stock_price: 19.84759364627786
    }
  },
  {
    _id: 'AAP',
    value: {
      sum_stock_price_high: 184072.91999999984,
      count: 4116,
      avg_stock_price: 44.72131195335273
```



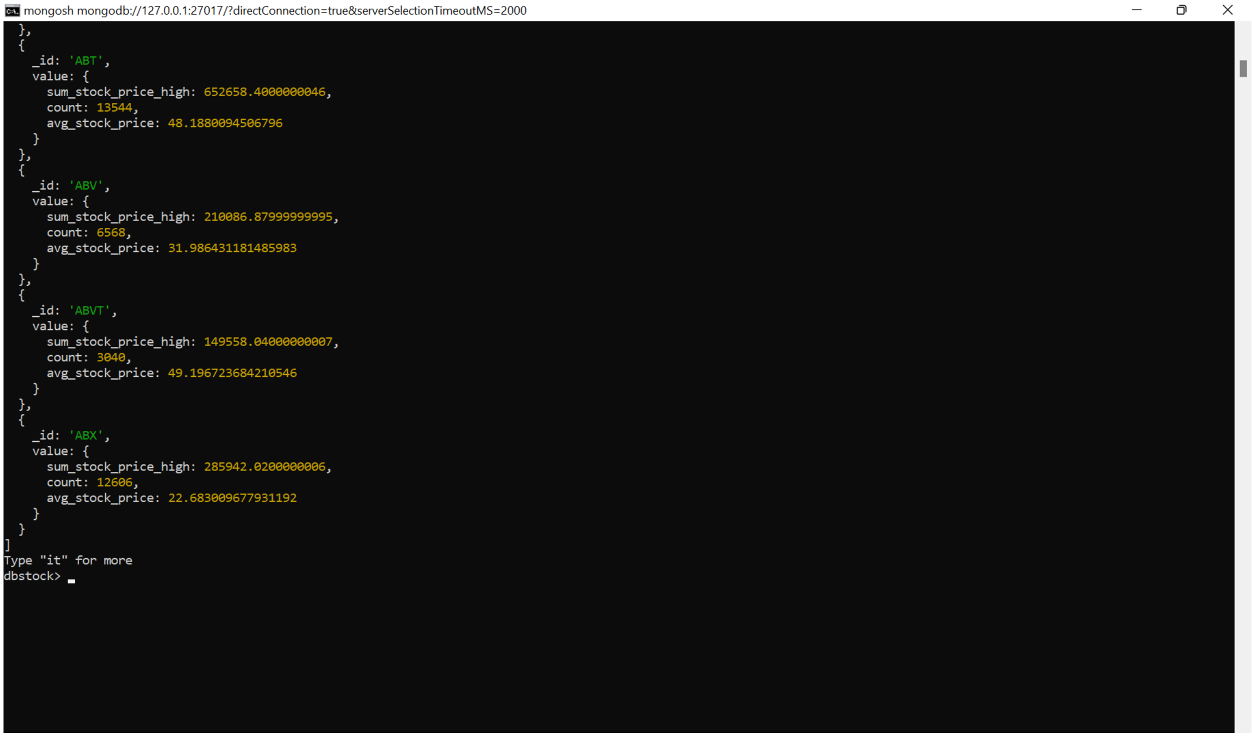
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

```
  },
  {
    _id: 'AAR',
    value: {
      sum_stock_price_high: 101115.84000000008,
      count: 5264,
      avg_stock_price: 19.20893617021278
    }
  },
  {
    _id: 'AAV',
    value: {
      sum_stock_price_high: 35870.64,
      count: 2870,
      avg_stock_price: 12.498480836236933
    }
  },
  {
    _id: 'AB',
    value: {
      sum_stock_price_high: 336802.53999999934,
      count: 10990,
      avg_stock_price: 30.64627297543215
    }
  },
  {
    _id: 'ABA',
    value: {
      sum_stock_price_high: 47101.98000000007,
      count: 1812,
      avg_stock_price: 25.994470198675536
    }
  },
  {
    _id: 'ABB',
    value: {
      sum_stock_price_high: 55896.39999999997,
      count: 4442,
      avg_stock_price: 12.583610986042316
    }
  },
  {
    _id: 'ABC',
    value: {
      sum_stock_price_high: 356796.95999999903,
```



mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

```
  {
    _id: 'ABD',
    value: {
      sum_stock_price_high: 35437.200000000026,
      count: 2254,
      avg_stock_price: 15.721916592724059
    }
  },
  {
    _id: 'ABG',
    value: {
      sum_stock_price_high: 61222.46000000018,
      count: 3968,
      avg_stock_price: 15.429047379032303
    }
  },
  {
    _id: 'ABK',
    value: {
      sum_stock_price_high: 480534.3400000003,
      count: 9364,
      avg_stock_price: 51.317208457923996
    }
  },
  {
    _id: 'ABM',
    value: {
      sum_stock_price_high: 316220.9199999959,
      count: 12892,
      avg_stock_price: 24.52846106112286
    }
  },
  {
    _id: 'ABR',
    value: {
      sum_stock_price_high: 53449.34,
      count: 2900,
      avg_stock_price: 18.430806896551722
    }
  },
  {
    _id: 'ABT',
    value: {
      sum_stock_price_high: 652658.4000000046,
      count: 13544,
```



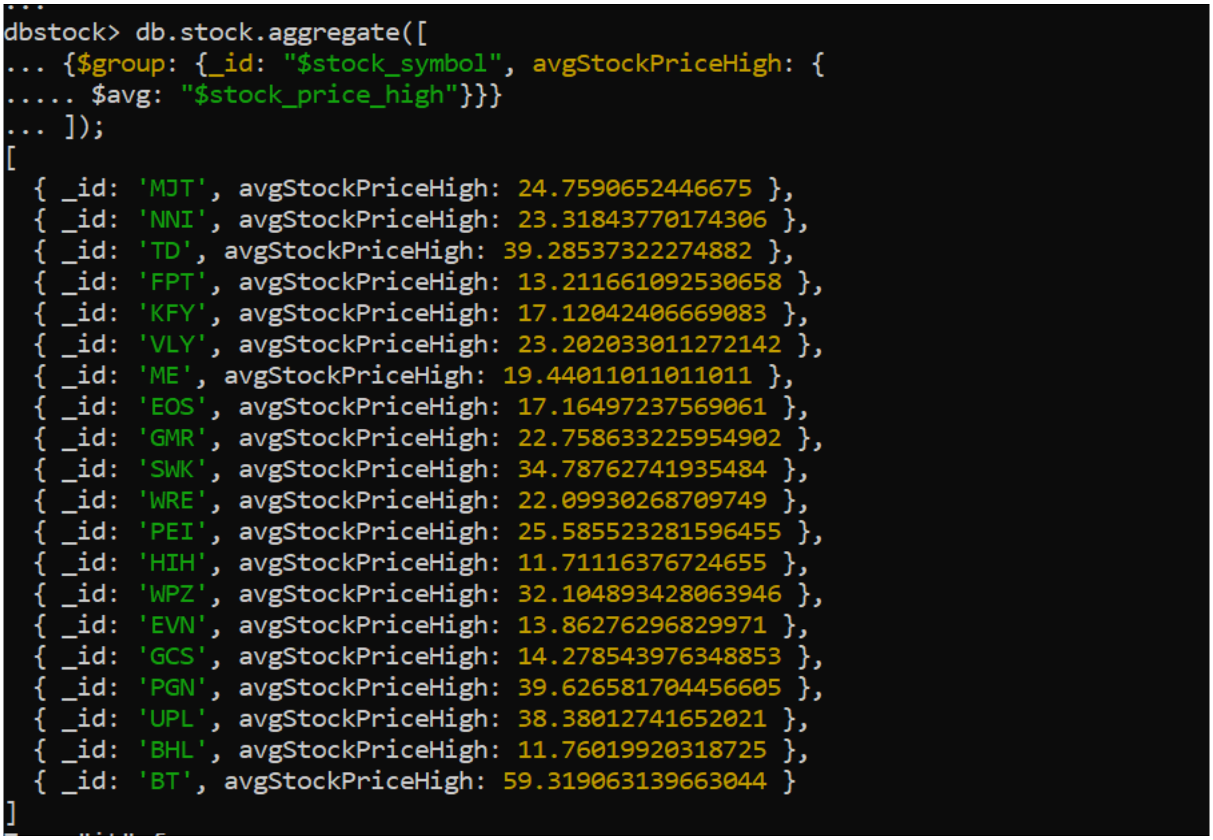
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

```
  },
  {
    _id: 'ABT',
    value: {
      sum_stock_price_high: 652658.4000000046,
      count: 13544,
      avg_stock_price: 48.1880094506796
    }
  },
  {
    _id: 'ABV',
    value: {
      sum_stock_price_high: 210086.87999999995,
      count: 6568,
      avg_stock_price: 31.986431181485983
    }
  },
  {
    _id: 'ABVT',
    value: {
      sum_stock_price_high: 149558.04000000007,
      count: 3040,
      avg_stock_price: 49.196723684210546
    }
  },
  {
    _id: 'ABX',
    value: {
      sum_stock_price_high: 285942.0200000006,
      count: 12606,
      avg_stock_price: 22.683009677931192
    }
  }
]
Type "it" for more
dbstock> 
```

Part 4:

Calculate the average stock price of each price of all stocks using $avg aggregation.

```
db.stock.aggregate([

... {$group: {_id: "$stock_symbol", avgStockPriceHigh: {

..... $avg: "$stock_price_high"}}}

... ]);
```



```
dbstock> db.stock.aggregate([
... {$group: {_id: "$stock_symbol", avgStockPriceHigh: {
..... $avg: "$stock_price_high"}}}
... ]);
[
  { _id: 'MJT', avgStockPriceHigh: 24.7590652446675 },
  { _id: 'NNI', avgStockPriceHigh: 23.31843770174306 },
  { _id: 'TD', avgStockPriceHigh: 39.28537322274882 },
  { _id: 'FPT', avgStockPriceHigh: 13.211661092530658 },
  { _id: 'KFY', avgStockPriceHigh: 17.12042406669083 },
  { _id: 'VLY', avgStockPriceHigh: 23.202033011272142 },
  { _id: 'ME', avgStockPriceHigh: 19.44011011011011 },
  { _id: 'EOS', avgStockPriceHigh: 17.16497237569061 },
  { _id: 'GMR', avgStockPriceHigh: 22.758633225954902 },
  { _id: 'SWK', avgStockPriceHigh: 34.78762741935484 },
  { _id: 'WRE', avgStockPriceHigh: 22.09930268709749 },
  { _id: 'PEI', avgStockPriceHigh: 25.585523281596455 },
  { _id: 'HIH', avgStockPriceHigh: 11.71116376724655 },
  { _id: 'WPZ', avgStockPriceHigh: 32.104893428063946 },
  { _id: 'EVN', avgStockPriceHigh: 13.86276296829971 },
  { _id: 'GCS', avgStockPriceHigh: 14.278543976348853 },
  { _id: 'PGN', avgStockPriceHigh: 39.626581704456605 },
  { _id: 'UPL', avgStockPriceHigh: 38.38012741652021 },
  { _id: 'BHL', avgStockPriceHigh: 11.76019920318725 },
  { _id: 'BT', avgStockPriceHigh: 59.31906313963044 }
]
```

Part 5.1: Programming Assignment

Import the Movielens dataset into MongoDB.

```
cd ~
curl -O https://files.grouplens.org/datasets/movielens/ml-1m.zip
sudo apt install -y unzip
unzip ml-1m.zip
cd ml-1m
cp ratings.dat ratings.csv
cp movies.dat movies.csv
cp users.dat users.csv
sed -i 's/::/,/g' ratings.csv
sed -i 's/,/-/g' movies.csv
sed -i 's/::/,/g' movies.csv
sed -i 's/::/,/g' users.csv
```

Adding headers to each csv file

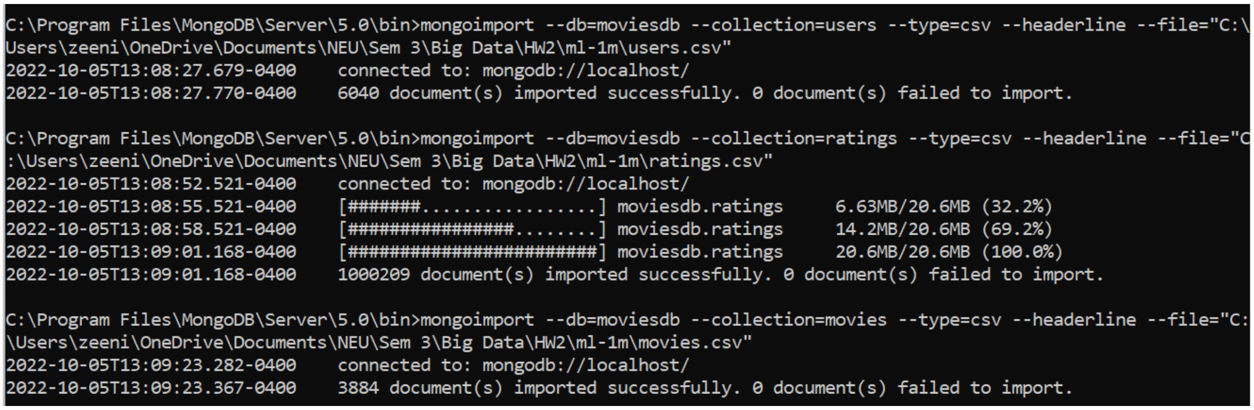
```
sed -i '1s/^/UserID,Gender,Age,Occupation,Zip-code\n/' users.csv
sed -i '1s/^/MovieID, Title ,Genres\n/' movies.csv
sed -i '1s/^/UserID,MovieID,Rating,Timestamp\n/' ratings.csv
```

Importing csv files into collections

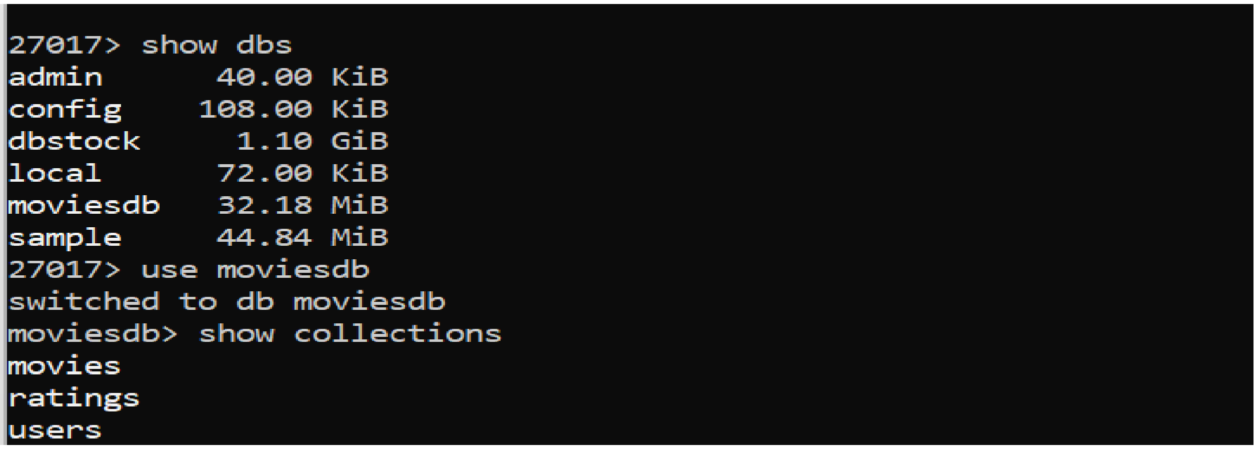
mongoimport --db=moviesdb --collection=users --type=csv --headerline --file="C:\Users\zeeni\OneDrive\Documents\NEU\Sem 3\Big Data\HW2\ml-1m\users.csv"

mongoimport --db=moviesdb --collection=ratings --type=csv --headerline --file="C:\Users\zeeni\OneDrive\Documents\NEU\Sem 3\Big Data\HW2\ml-1m\ratings.csv"

mongoimport --db=moviesdb --collection=movies --type=csv --headerline --file="C:\Users\zeeni\OneDrive\Documents\NEU\Sem 3\Big Data\HW2\ml-1m\movies.csv"

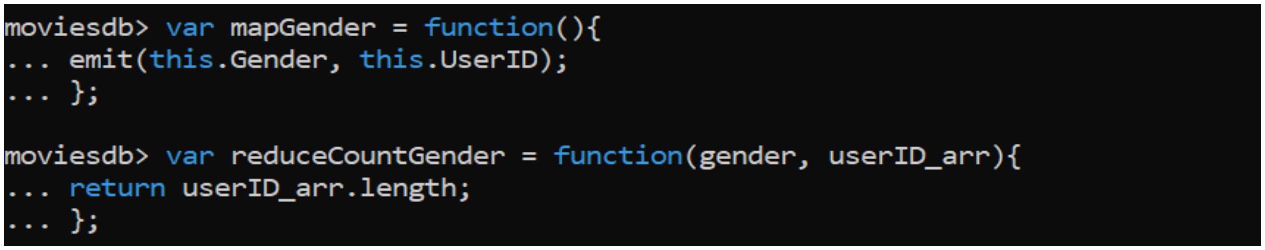


```
C:\Program Files\MongoDB\Server\5.0\bin>mongoimport --db=moviesdb --collection=users --type=csv --headerline --file="C:\
Users\zeeni\OneDrive\Documents\NEU\Sem 3\Big Data\HW2\ml-1m\users.csv"
2022-10-05T13:08:27.679-0400    connected to: mongodb://localhost/
2022-10-05T13:08:27.770-0400    6040 document(s) imported successfully. 0 document(s) failed to import.

C:\Program Files\MongoDB\Server\5.0\bin>mongoimport --db=moviesdb --collection=ratings --type=csv --headerline --file="C
:\Users\zeeni\OneDrive\Documents\NEU\Sem 3\Big Data\HW2\ml-1m\ratings.csv"
2022-10-05T13:08:52.521-0400    connected to: mongodb://localhost/
2022-10-05T13:08:55.521-0400    [#######.................] moviesdb.ratings     6.63MB/20.6MB (32.2%)
2022-10-05T13:08:58.521-0400    [################........] moviesdb.ratings     14.2MB/20.6MB (69.2%)
2022-10-05T13:09:01.168-0400    [########################] moviesdb.ratings     20.6MB/20.6MB (100.0%)
2022-10-05T13:09:01.168-0400    1000209 document(s) imported successfully. 0 document(s) failed to import.

C:\Program Files\MongoDB\Server\5.0\bin>mongoimport --db=moviesdb --collection=movies --type=csv --headerline --file="C:
\Users\zeeni\OneDrive\Documents\NEU\Sem 3\Big Data\HW2\ml-1m\movies.csv"
2022-10-05T13:09:23.282-0400    connected to: mongodb://localhost/
2022-10-05T13:09:23.367-0400    3884 document(s) imported successfully. 0 document(s) failed to import.
```

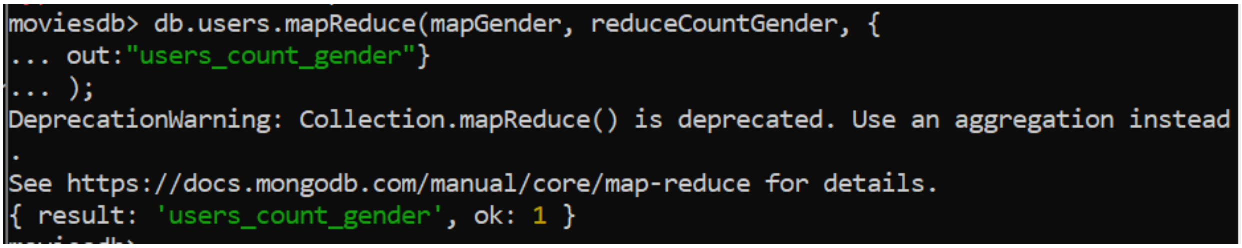


```
27017> show dbs
admin      40.00 KiB
config    108.00 KiB
dbstock     1.10 GiB
local      72.00 KiB
moviesdb   32.18 MiB
sample     44.84 MiB
27017> use moviesdb
switched to db moviesdb
moviesdb> show collections
movies
ratings
users
```

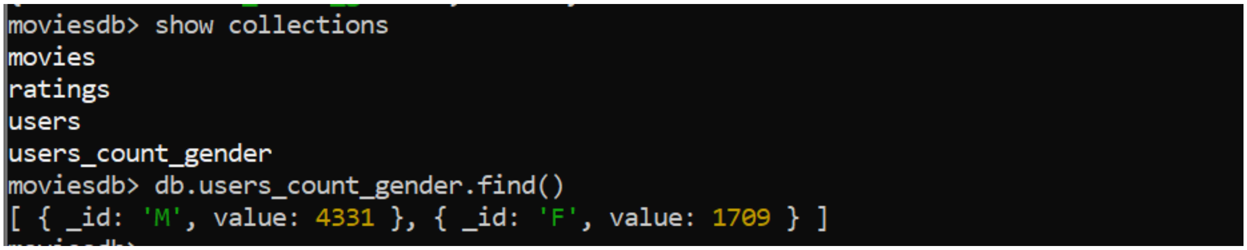
Find the number of Females and Males from the users collection using MapReduce. Do the same thing using count() to compare the results.



```
moviesdb> var mapGender = function(){
... emit(this.Gender, this.UserID);
... };

moviesdb> var reduceCountGender = function(gender, userID_arr){
... return userID_arr.length;
... };
```

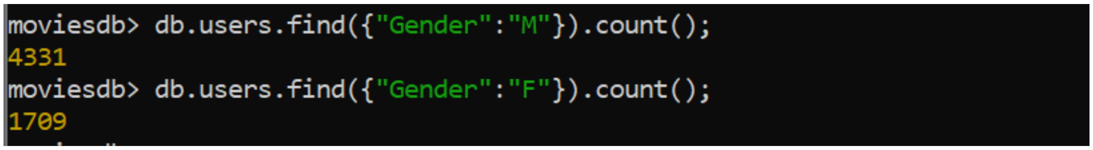


```
moviesdb> db.users.mapReduce(mapGender, reduceCountGender, {
... out:"users_count_gender"}
... );
DeprecationWarning: Collection.mapReduce() is deprecated. Use an aggregation instead
.
See https://docs.mongodb.com/manual/core/map-reduce for details.
{ result: 'users_count_gender', ok: 1 }
```



```
moviesdb> show collections
movies
ratings
users
users_count_gender
moviesdb> db.users_count_gender.find()
[ { _id: 'M', value: 4331 }, { _id: 'F', value: 1709 } ]
```

Using count()

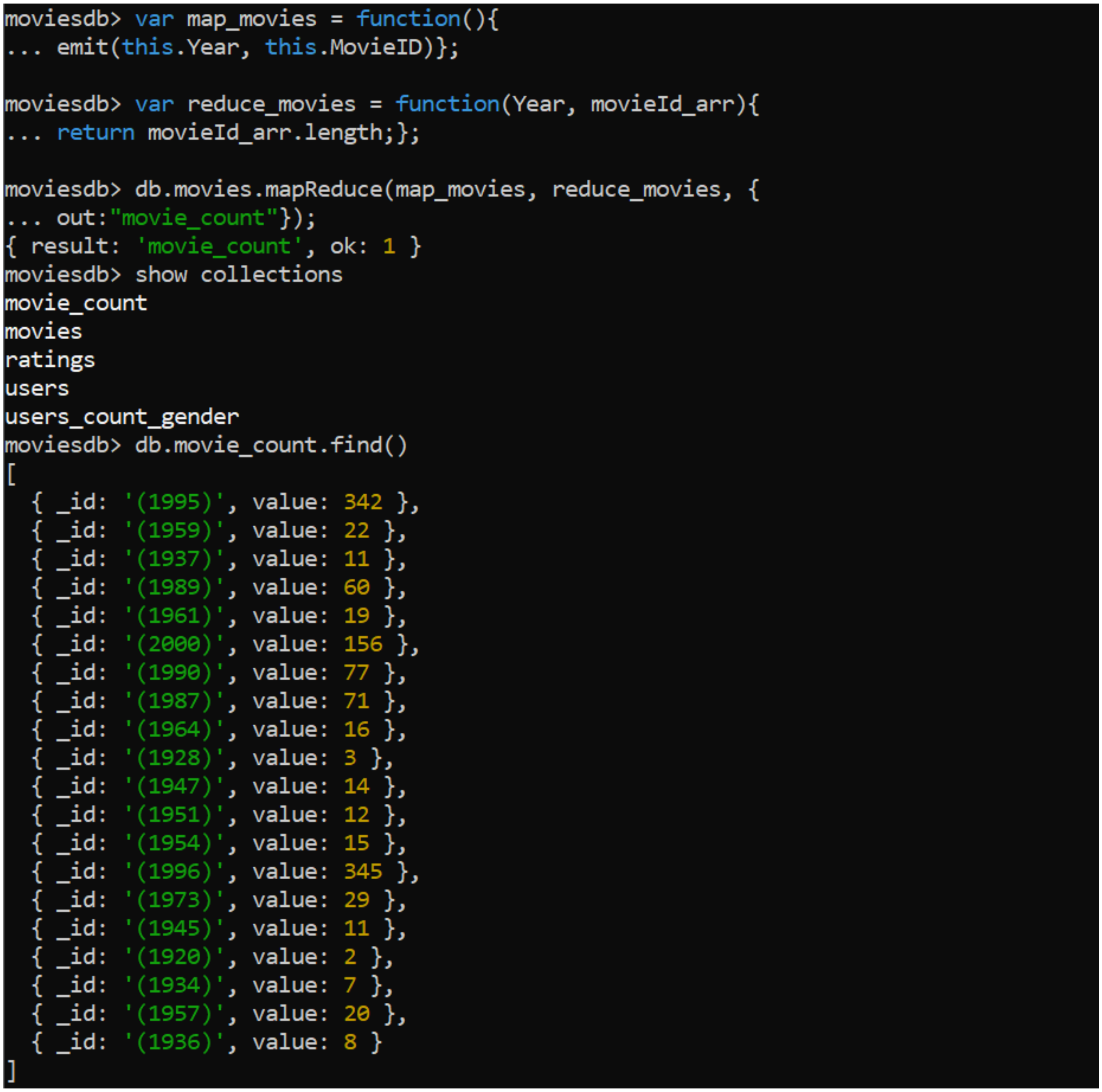


```
moviesdb> db.users.find({"Gender":"M"}).count();
4331
moviesdb> db.users.find({"Gender":"F"}).count();
1709
```

Find the number of Movies per year using MapReduce.

Adding Year field in movies collection

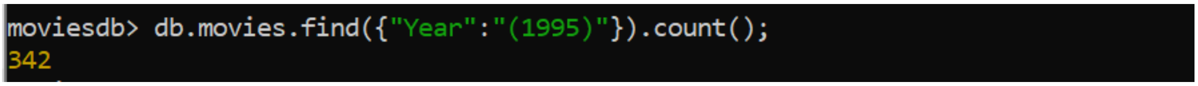
```
db.movies.find({}).forEach(
        function(e,i){
                var text = e.title || "";
                e.year = text.toString().substr(e.title.length-5,4);
                db.movies.save(e);
});
```



```
moviesdb> var map_movies = function(){
... emit(this.Year, this.MovieID)};

moviesdb> var reduce_movies = function(Year, movieId_arr){
... return movieId_arr.length;};

moviesdb> db.movies.mapReduce(map_movies, reduce_movies, {
... out:"movie_count"});
{ result: 'movie_count', ok: 1 }
moviesdb> show collections
movie_count
movies
ratings
users
users_count_gender
moviesdb> db.movie_count.find()
[
  { _id: '(1995)', value: 342 },
  { _id: '(1959)', value: 22 },
  { _id: '(1937)', value: 11 },
  { _id: '(1989)', value: 60 },
  { _id: '(1961)', value: 19 },
  { _id: '(2000)', value: 156 },
  { _id: '(1990)', value: 77 },
  { _id: '(1987)', value: 71 },
  { _id: '(1964)', value: 16 },
  { _id: '(1928)', value: 3 },
  { _id: '(1947)', value: 14 },
  { _id: '(1951)', value: 12 },
  { _id: '(1954)', value: 15 },
  { _id: '(1996)', value: 345 },
  { _id: '(1973)', value: 29 },
  { _id: '(1945)', value: 11 },
  { _id: '(1920)', value: 2 },
  { _id: '(1934)', value: 7 },
  { _id: '(1957)', value: 20 },
  { _id: '(1936)', value: 8 }
]
```

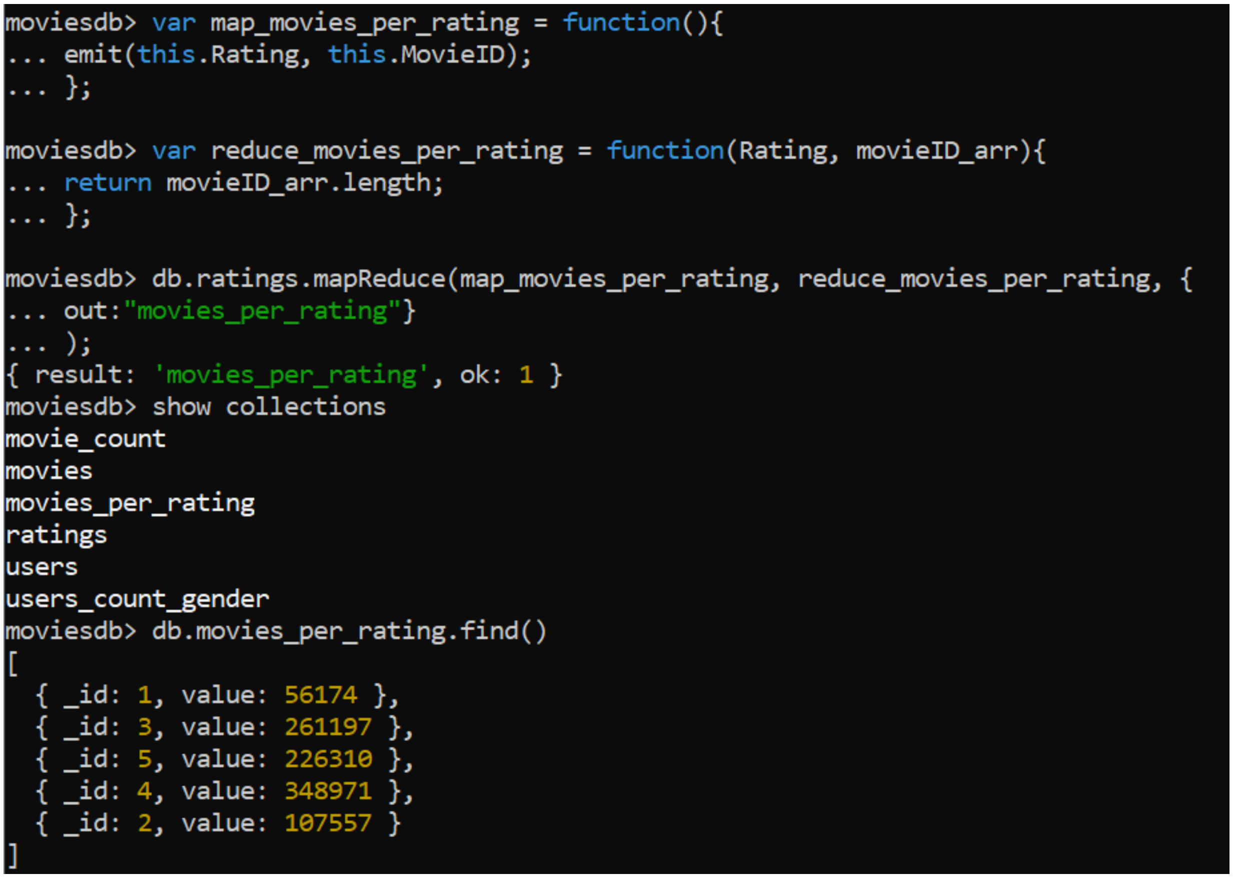
Using count()

db.movies.find({"Year":"(1995)"}).count();



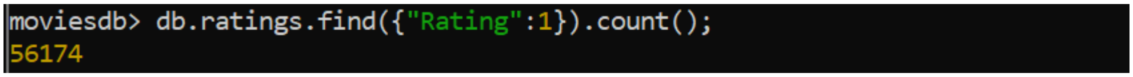
```
moviesdb> db.movies.find({"Year":"(1995)"}).count();
342
```

Find the number of Movies per rating using MapReduce.



```
moviesdb> var map_movies_per_rating = function(){
... emit(this.Rating, this.MovieID);
... };

moviesdb> var reduce_movies_per_rating = function(Rating, movieID_arr){
... return movieID_arr.length;
... };

moviesdb> db.ratings.mapReduce(map_movies_per_rating, reduce_movies_per_rating, {
... out:"movies_per_rating"}
... );
{ result: 'movies_per_rating', ok: 1 }
moviesdb> show collections
movie_count
movies
movies_per_rating
ratings
users
users_count_gender
moviesdb> db.movies_per_rating.find()
[
  { _id: 1, value: 56174 },
  { _id: 3, value: 261197 },
  { _id: 5, value: 226310 },
  { _id: 4, value: 348971 },
  { _id: 2, value: 107557 }
]
```

Using count():

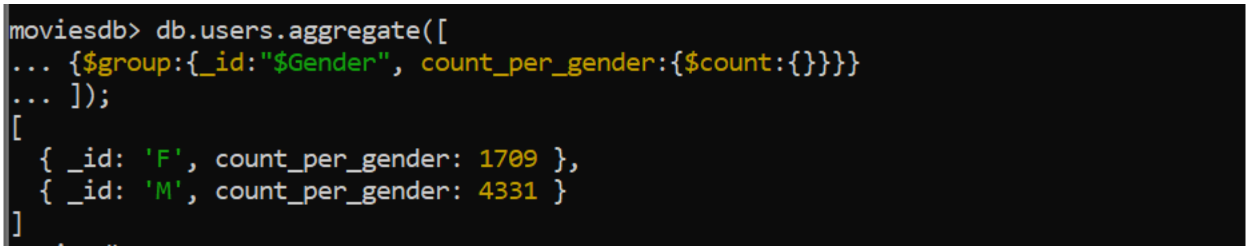


```
moviesdb> db.ratings.find({"Rating":1}).count();
56174
```

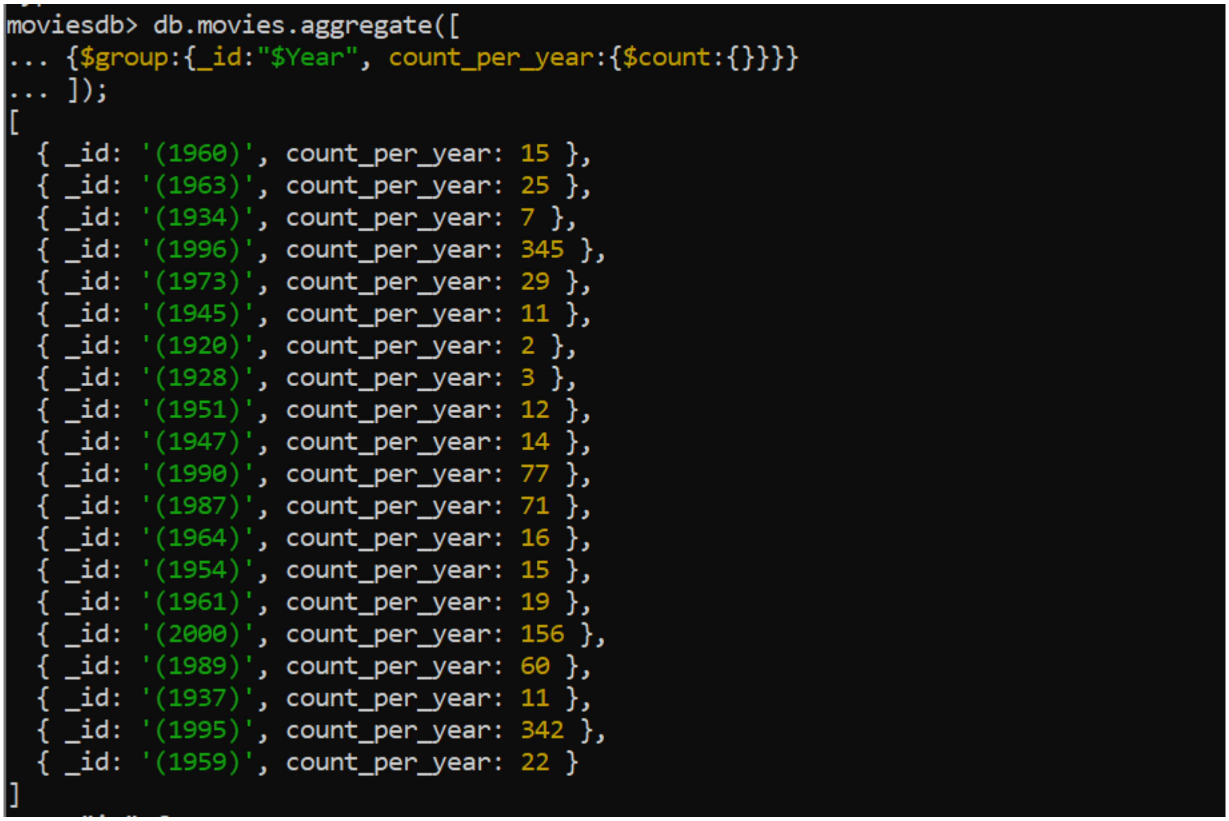
Part 5.2:

Repeat 5.1 using Aggregation Pipeline.

Find the number of Females and Males from the users collection using MapReduce.

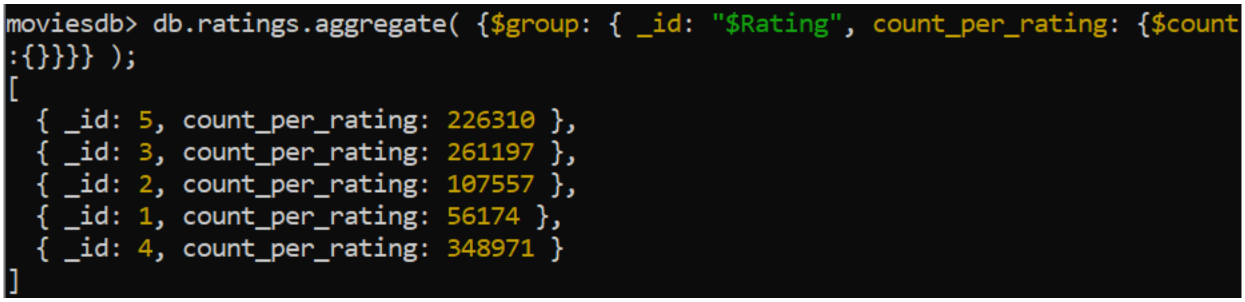


```
moviesdb> db.users.aggregate([
... {$group:{_id:"$Gender", count_per_gender:{$count:{}}}}
... ]);
[
  { _id: 'F', count_per_gender: 1709 },
  { _id: 'M', count_per_gender: 4331 }
]
```



```
moviesdb> db.movies.aggregate([
... {$group:{_id:"$Year", count_per_year:{$count:{}}}}
... ]);
[
  { _id: '(1960)', count_per_year: 15 },
  { _id: '(1963)', count_per_year: 25 },
  { _id: '(1934)', count_per_year: 7 },
  { _id: '(1996)', count_per_year: 345 },
  { _id: '(1973)', count_per_year: 29 },
  { _id: '(1945)', count_per_year: 11 },
  { _id: '(1920)', count_per_year: 2 },
  { _id: '(1928)', count_per_year: 3 },
  { _id: '(1951)', count_per_year: 12 },
  { _id: '(1947)', count_per_year: 14 },
  { _id: '(1990)', count_per_year: 77 },
  { _id: '(1987)', count_per_year: 71 },
  { _id: '(1964)', count_per_year: 16 },
  { _id: '(1954)', count_per_year: 15 },
  { _id: '(1961)', count_per_year: 19 },
  { _id: '(2000)', count_per_year: 156 },
  { _id: '(1989)', count_per_year: 60 },
  { _id: '(1937)', count_per_year: 11 },
  { _id: '(1995)', count_per_year: 342 },
  { _id: '(1959)', count_per_year: 22 }
]
```

Find the number of Movies per rating using MapReduce.



```
moviesdb> db.ratings.aggregate( {$group: { _id: "$Rating", count_per_rating: {$count
:{}}}} );
[
  { _id: 5, count_per_rating: 226310 },
  { _id: 3, count_per_rating: 261197 },
  { _id: 2, count_per_rating: 107557 },
  { _id: 1, count_per_rating: 56174 },
  { _id: 4, count_per_rating: 348971 }
]
```

Part – 6: Programming Assignment

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

Solution:

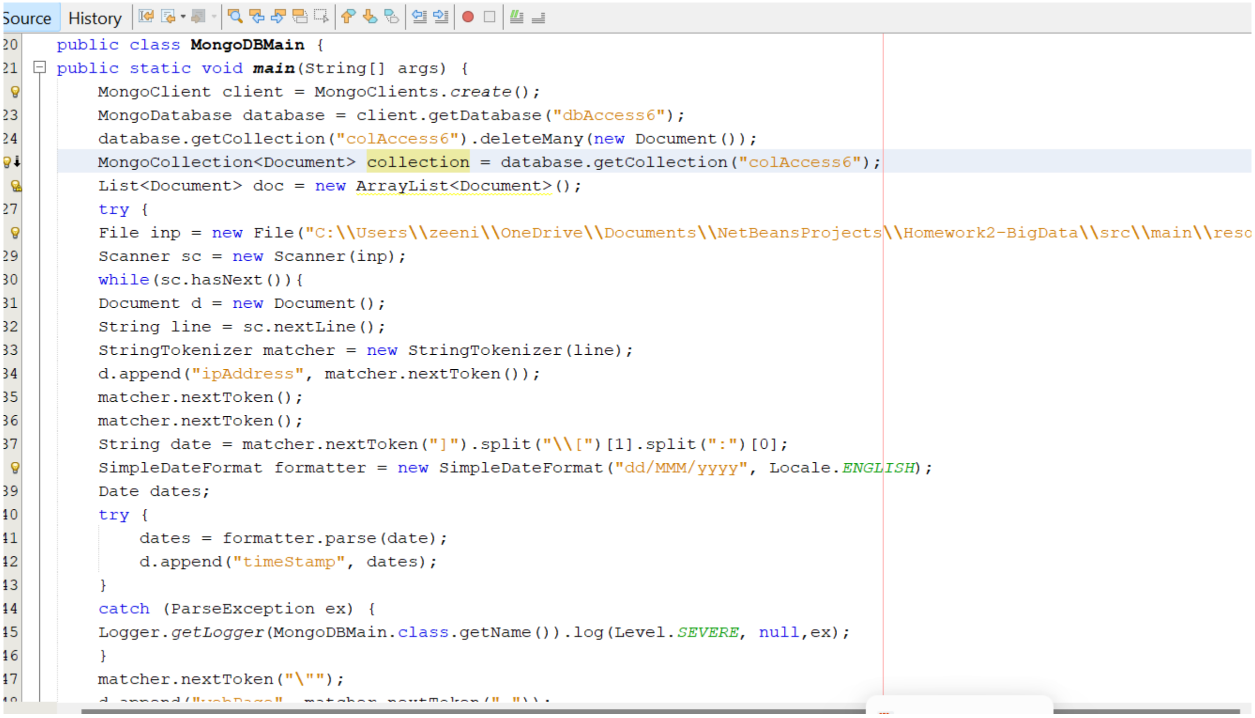
import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoCollection;

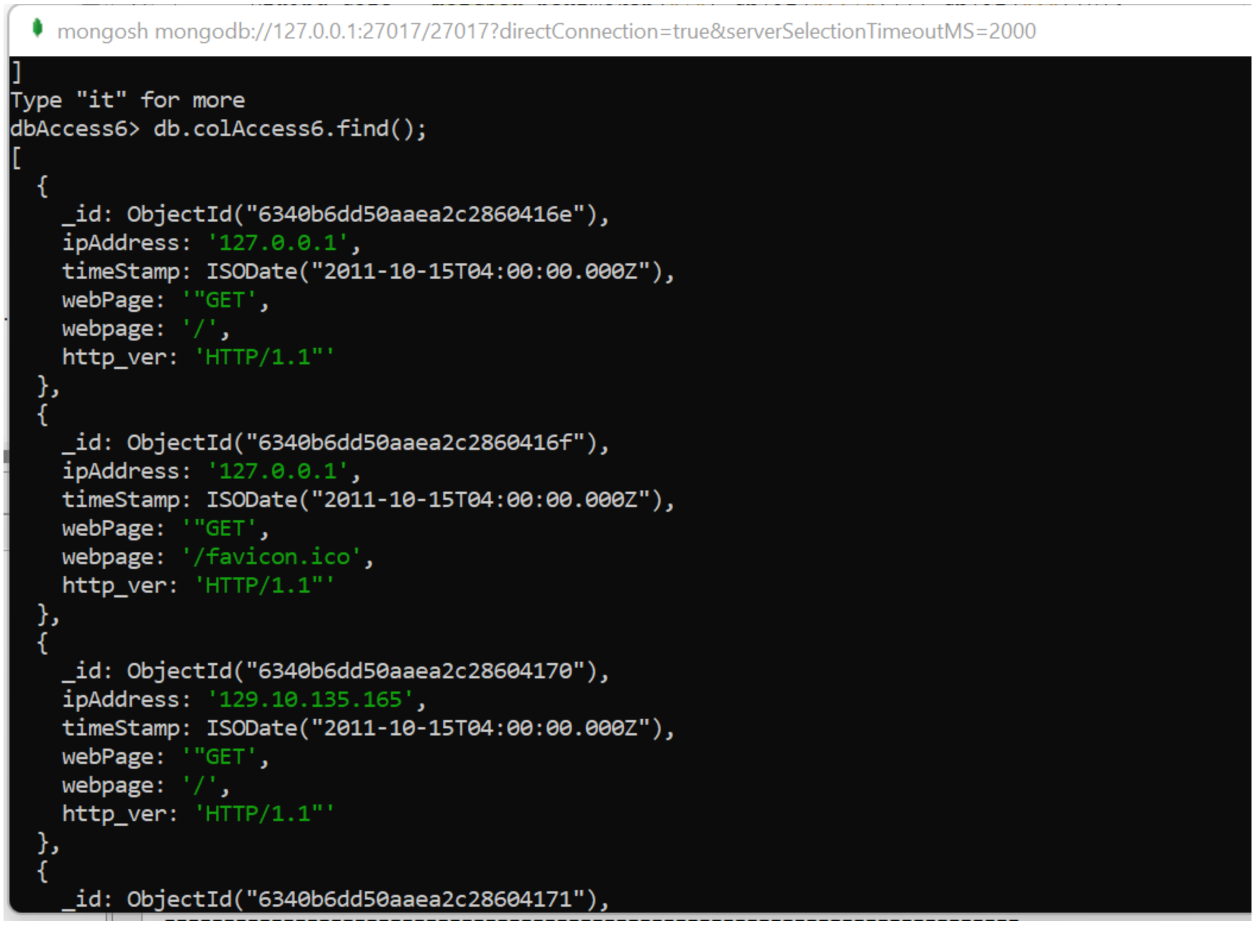
```java
import com.mongodb.client.MongoDatabase;
import java.io.File;
import java.io.FileNotFoundException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.Scanner;
import java.util.StringTokenizer;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.bson.Document;


public class MongoDBMain {
public static void main(String[] args) {
   MongoClient client = MongoClients.create();
   MongoDatabase database = client.getDatabase("dbAccess6");
   database.getCollection("colAccess6").deleteMany(new Document());
   MongoCollection<Document> collection = database.getCollection("colAccess6");
   List<Document> doc = new ArrayList<Document>();
   try {
   File inp = new File("C:\\Users\\zeeni\\OneDrive\\Documents\\NetBeansProjects\\Homework2-BigData\\src\\main\\resources\\access.log");
   Scanner sc = new Scanner(inp);
   while(sc.hasNext()){
   Document insertDoc = new Document();
   String line = sc.nextLine();
```

```
    StringTokenizer matcher = new StringTokenizer(line);
    insertDoc.append("ipAddress", matcher.nextToken());
    matcher.nextToken();
    matcher.nextToken();
    String date = matcher.nextToken("]").split("\\[")[1].split(":")[0];
    SimpleDateFormat formatter = new SimpleDateFormat("dd/MMM/yyyy", Locale.ENGLISH);
    Date dates;
    try {
        dates = formatter.parse(date);
        insertDoc.append("timeStamp", dates);
    }
    catch (ParseException ex) {
    Logger.getLogger(MongoDBMain.class.getName()).log(Level.SEVERE, null,ex);
    }
    matcher.nextToken("\"");
    insertDoc.append("call", matcher.nextToken(" "));
    insertDoc.append("webPage", matcher.nextToken());
    insertDoc.append("httpVersion", matcher.nextToken());
    doc.add(insertDoc);
    }
    collection.insertMany(doc);
    }
    catch (FileNotFoundException ex) {
        ex.printStackTrace();
        Logger.getLogger(MongoDBMain.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```



```java
public class MongoDBMain {
public static void main(String[] args) {
    MongoClient client = MongoClients.create();
    MongoDatabase database = client.getDatabase("dbAccess6");
    database.getCollection("colAccess6").deleteMany(new Document());
    MongoCollection<Document> collection = database.getCollection("colAccess6");
    List<Document> doc = new ArrayList<Document>();
    try {
    File inp = new File("C:\\Users\\zeeni\\OneDrive\\Documents\\NetBeansProjects\\Homework2-BigData\\src\\main\\reso
    Scanner sc = new Scanner(inp);
    while(sc.hasNext()){
    Document d = new Document();
    String line = sc.nextLine();
    StringTokenizer matcher = new StringTokenizer(line);
    d.append("ipAddress", matcher.nextToken());
    matcher.nextToken();
    matcher.nextToken();
    String date = matcher.nextToken("]").split("\\[")[1].split(":")[0];
    SimpleDateFormat formatter = new SimpleDateFormat("dd/MMM/yyyy", Locale.ENGLISH);
    Date dates;
    try {
        dates = formatter.parse(date);
        d.append("timeStamp", dates);
    }
    catch (ParseException ex) {
    Logger.getLogger(MongoDBMain.class.getName()).log(Level.SEVERE, null,ex);
    }
    matcher.nextToken("\"");
    d.append("webPage", matcher.nextToken(" "));
```

Checking collection in MongoDB



mongosh mongodb://127.0.0.1:27017/27017?directConnection=true&serverSelectionTimeoutMS=2000

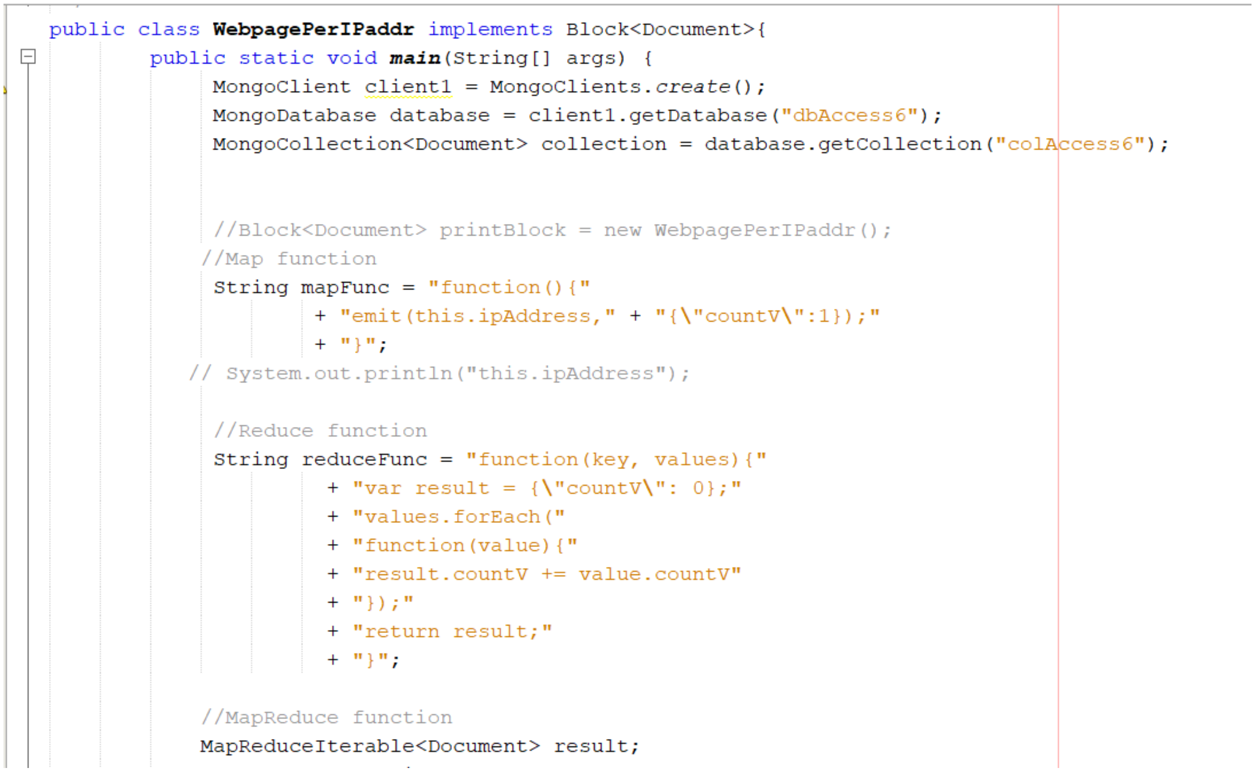
```
]
Type "it" for more
dbAccess6> db.colAccess6.find();
[
  {
    _id: ObjectId("6340b6dd50aaea2c2860416e"),
    ipAddress: '127.0.0.1',
    timeStamp: ISODate("2011-10-15T04:00:00.000Z"),
    webPage: '"GET',
    webpage: '/',
    http_ver: 'HTTP/1.1"'
  },
  {
    _id: ObjectId("6340b6dd50aaea2c2860416f"),
    ipAddress: '127.0.0.1',
    timeStamp: ISODate("2011-10-15T04:00:00.000Z"),
    webPage: '"GET',
    webpage: '/favicon.ico',
    http_ver: 'HTTP/1.1"'
  },
  {
    _id: ObjectId("6340b6dd50aaea2c28604170"),
    ipAddress: '129.10.135.165',
    timeStamp: ISODate("2011-10-15T04:00:00.000Z"),
    webPage: '"GET',
    webpage: '/',
    http_ver: 'HTTP/1.1"'
  },
  {
    _id: ObjectId("6340b6dd50aaea2c28604171"),
```

Number of times any webpage was visited by the same IP address.

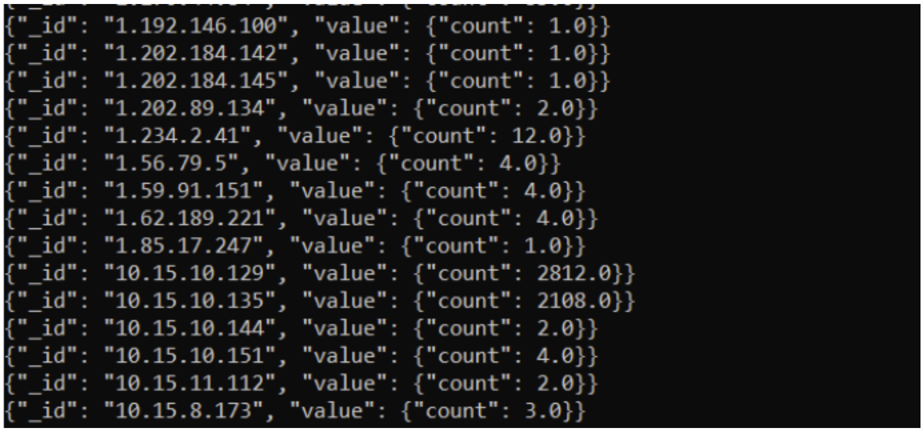
Solution:

```
import com.mongodb.Block;

import com.mongodb.client.MapReduceIterable;

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoDatabase;

import org.bson.Document;
```

```
/**
 *
 * @author zeeni
 */
public class WebpagePerIPaddr implements Block<Document>{
    public static void main(String[] args) {
       MongoClient client1 = MongoClients.create();
       MongoDatabase database = client1.getDatabase("accessdb");
       MongoCollection<Document> collection = database.getCollection("accesscol");
       //Map function
       String mapFunc = "function(){"
           + "emit(this.IPAddress," + "{\"count\":1});"
           + "}";

       //Reduce function
       String reduceFunc = "function(key, values){"
            + "var result = {\"count\": 0};"
            + "values.forEach("
            + "function(value){"
            + "result.count += value.count;"
            + "});"
            + "return result;"
            + "}";

      //MapReduce function
      MapReduceIterable<Document> result;
      result = collection.mapReduce(mapFunc, reduceFunc);
      for(Document doc:result){
            System.out.println(doc.toJson());
}
    client1.close();
   }
  public void apply(Document docs){
     System.out.println(docs.toJson());
}


}
```



```
public class WebpagePerIPaddr implements Block<Document>{
        public static void main(String[] args) {
            MongoClient client1 = MongoClients.create();
            MongoDatabase database = client1.getDatabase("dbAccess6");
            MongoCollection<Document> collection = database.getCollection("colAccess6");


            //Block<Document> printBlock = new WebpagePerIPaddr();
           //Map function
            String mapFunc = "function(){"
                    + "emit(this.ipAddress," + "{\"countV\":1});"
                    + "}";
          // System.out.println("this.ipAddress");

            //Reduce function
            String reduceFunc = "function(key, values){"
                     + "var result = {\"countV\": 0};"
                     + "values.forEach("
                     + "function(value){"
                     + "result.countV += value.countV"
                     + "});"
                     + "return result;"
                     + "}";

           //MapReduce function
           MapReduceIterable<Document> result;
```



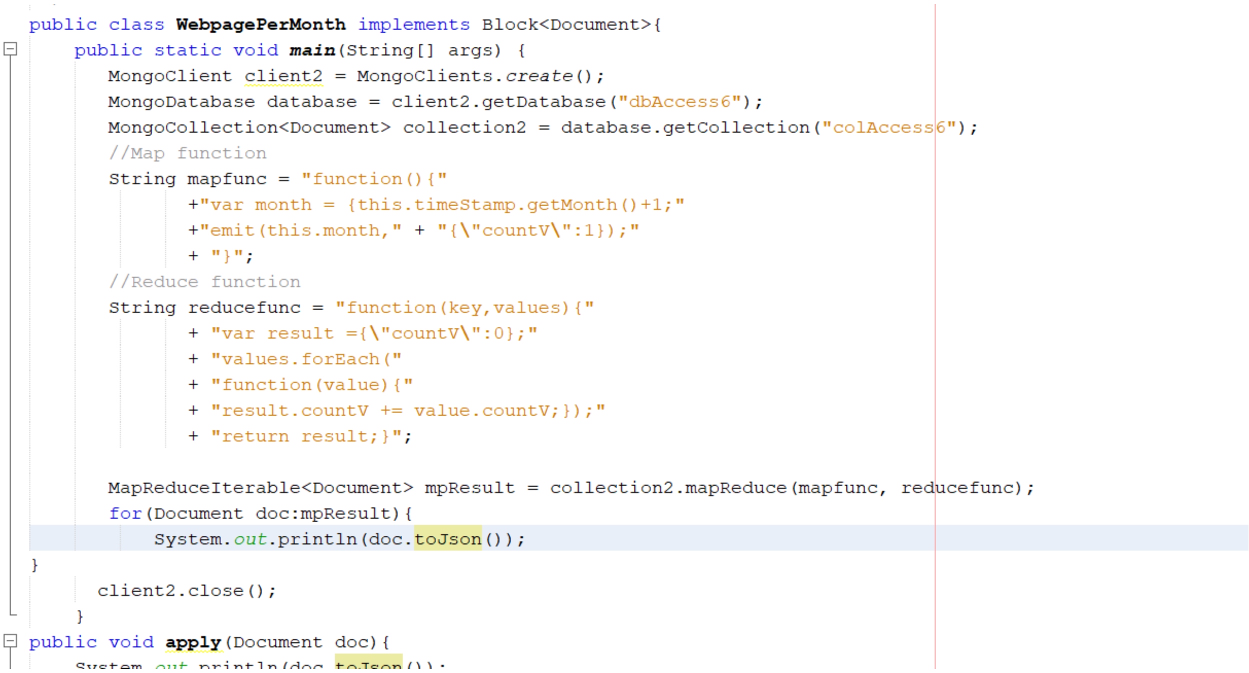
```
{"_id": "1.192.146.100", "value": {"count": 1.0}}
{"_id": "1.202.184.142", "value": {"count": 1.0}}
{"_id": "1.202.184.145", "value": {"count": 1.0}}
{"_id": "1.202.89.134", "value": {"count": 2.0}}
{"_id": "1.234.2.41", "value": {"count": 12.0}}
{"_id": "1.56.79.5", "value": {"count": 4.0}}
{"_id": "1.59.91.151", "value": {"count": 4.0}}
{"_id": "1.62.189.221", "value": {"count": 4.0}}
{"_id": "1.85.17.247", "value": {"count": 1.0}}
{"_id": "10.15.10.129", "value": {"count": 2812.0}}
{"_id": "10.15.10.135", "value": {"count": 2108.0}}
{"_id": "10.15.10.144", "value": {"count": 2.0}}
{"_id": "10.15.10.151", "value": {"count": 4.0}}
{"_id": "10.15.11.112", "value": {"count": 2.0}}
{"_id": "10.15.8.173", "value": {"count": 3.0}}
```

Number of times any webpage was visited each month.

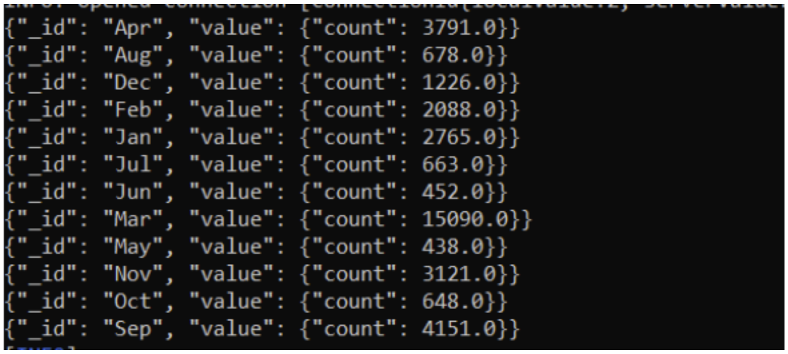
Solution:

```
import com.mongodb.Block;

import com.mongodb.client.MapReduceIterable;

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoDatabase;

import org.bson.Document;


/ *

 * Click nbfs:/ / nbhost/ SystemFileSystem/ Templates/ Licenses/ license-default.txt to change this license

 * Click nbfs:/ / nbhost/ SystemFileSystem/ Templates/ Classes/ Class.java to edit this template

 */


/ **

 *

 * @author zeeni

 */
```

```
public class WebpagePerMonth implements Block<Document>{
    public static void main(String[] args) {
       MongoClient client2 = MongoClients.create();
       MongoDatabase database = client2.getDatabase("dbAccess6");
       MongoCollection<Document> collection2 = database.getCollection("colAccess6");
       //Map function
       String mapfunc = "function(){"
              +"var month = this.timeStamp.getMonth()+1;"
              +"emit(this.month," + "{\"countV\":1});"
              + "}";
       //Reduce function
       String reducefunc = "function(key,values){"
              + "var result ={\"countV\":0};"
              + "values.forEach("
              + "function(value){"
              + "result.countV += value.countV;});"
              + "return result;}";

       MapReduceIterable<Document> mpResult = collection2.mapReduce(mapfunc, reducefunc);
       for(Document doc:mpResult){
           System.out.println(doc.toJson());
}
      client2.close();
    }
public void apply(Document doc){
    System.out.println(doc.toJson());
}
}
```



```
public class WebpagePerMonth implements Block<Document>{
    public static void main(String[] args) {
       MongoClient client2 = MongoClients.create();
       MongoDatabase database = client2.getDatabase("dbAccess6");
       MongoCollection<Document> collection2 = database.getCollection("colAccess6");
       //Map function
       String mapfunc = "function(){"
              +"var month = {this.timeStamp.getMonth()+1;"
              +"emit(this.month," + "{\"countV\":1});"
              + "}";
       //Reduce function
       String reducefunc = "function(key,values){"
              + "var result ={\"countV\":0};"
              + "values.forEach("
              + "function(value){"
              + "result.countV += value.countV;});"
              + "return result;}";

       MapReduceIterable<Document> mpResult = collection2.mapReduce(mapfunc, reducefunc);
       for(Document doc:mpResult){
           System.out.println(doc.toJson());
}
      client2.close();
    }
public void apply(Document doc){
    System.out.println(doc.toJson());
```



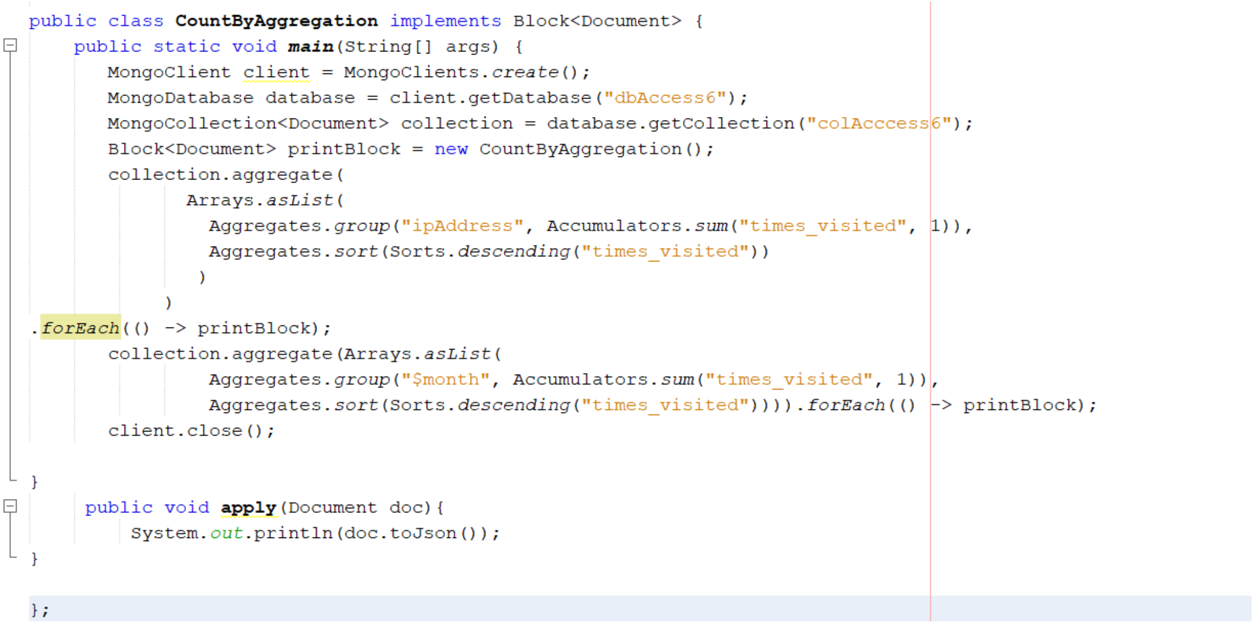
```
{"_id": "Apr", "value": {"count": 3791.0}}
{"_id": "Aug", "value": {"count": 678.0}}
{"_id": "Dec", "value": {"count": 1226.0}}
{"_id": "Feb", "value": {"count": 2088.0}}
{"_id": "Jan", "value": {"count": 2765.0}}
{"_id": "Jul", "value": {"count": 663.0}}
{"_id": "Jun", "value": {"count": 452.0}}
{"_id": "Mar", "value": {"count": 15090.0}}
{"_id": "May", "value": {"count": 438.0}}
{"_id": "Nov", "value": {"count": 3121.0}}
{"_id": "Oct", "value": {"count": 648.0}}
{"_id": "Sep", "value": {"count": 4151.0}}
```

PART 7 - PROGRAMMING ASSIGNMENT

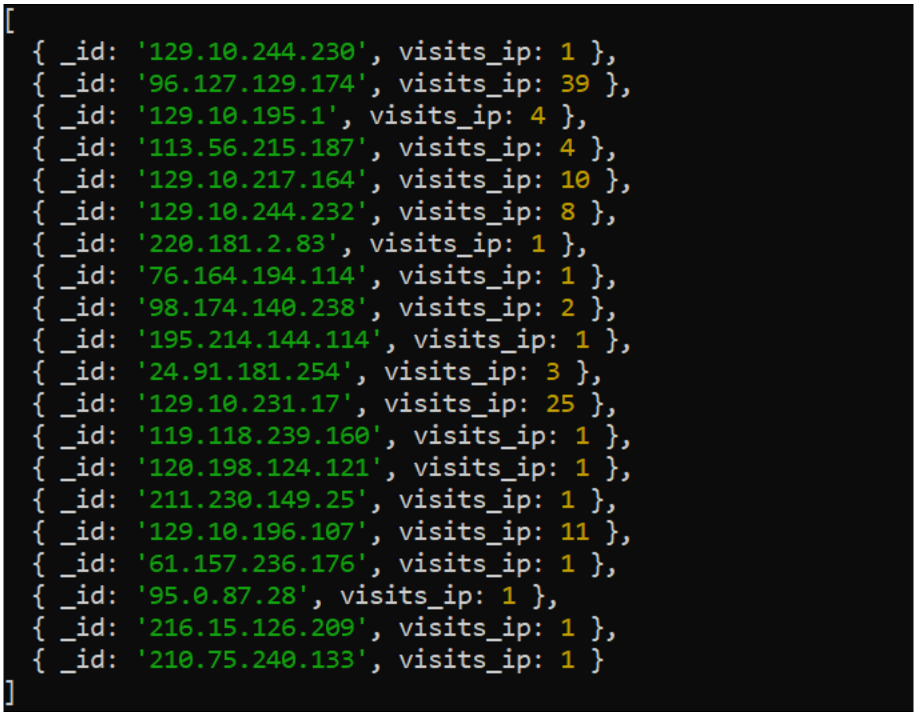
Redo Part-6 using Aggregation Pipeline.

```
import com.mongodb.Block;

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoDatabase;

import com.mongodb.client.model.Accumulators;

import com.mongodb.client.model.Aggregates;

import com.mongodb.client.model.Sorts;
import java.util.Arrays;

import java.util.function.Consumer;

import org.bson.Document;


/*

 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

 */


/**

 *

 * @author zeeni

 */

public class CountByAggregation implements Block<Document> {

  public static void main(String[] args) {

    MongoClient client = MongoClients.create();

    MongoDatabase database = client.getDatabase("dbAccess6");

    MongoCollection<Document> collection = database.getCollection("colAcccess6");

    Block<Document> printBlock = new CountByAggregation();

    collection.aggregate(

        Arrays.asList(

         Aggregates.group("ipAddress", Accumulators.sum("visits_ip", 1)),

         Aggregates.sort(Sorts.descending("visits_ip "))

         )

       )

.forEach(() -> printBlock);

    collection.aggregate(Arrays.asList(

         Aggregates.group("$month", Accumulators.sum("visits_month", 1)),

         Aggregates.sort(Sorts.descending("visits_month")))).forEach(() -> printBlock);
```

```
      client.close();


}
     public void apply(Document doc){
        System.out.println(doc.toJson());
}


};
```



```
public class CountByAggregation implements Block<Document> {
    public static void main(String[] args) {
       MongoClient client = MongoClients.create();
       MongoDatabase database = client.getDatabase("dbAccess6");
       MongoCollection<Document> collection = database.getCollection("colAcccess6");
       Block<Document> printBlock = new CountByAggregation();
       collection.aggregate(
              Arrays.asList(
                Aggregates.group("ipAddress", Accumulators.sum("times_visited", 1)),
                Aggregates.sort(Sorts.descending("times_visited"))
               )
            )
.forEach(() -> printBlock);
       collection.aggregate(Arrays.asList(
                Aggregates.group("$month", Accumulators.sum("times_visited", 1)),
                Aggregates.sort(Sorts.descending("times_visited")))).forEach(() -> printBlock);
       client.close();

}
     public void apply(Document doc){
        System.out.println(doc.toJson());
}

};
```



```
[
  { _id: '129.10.244.230', visits_ip: 1 },
  { _id: '96.127.129.174', visits_ip: 39 },
  { _id: '129.10.195.1', visits_ip: 4 },
  { _id: '113.56.215.187', visits_ip: 4 },
  { _id: '129.10.217.164', visits_ip: 10 },
  { _id: '129.10.244.232', visits_ip: 8 },
  { _id: '220.181.2.83', visits_ip: 1 },
  { _id: '76.164.194.114', visits_ip: 1 },
  { _id: '98.174.140.238', visits_ip: 2 },
  { _id: '195.214.144.114', visits_ip: 1 },
  { _id: '24.91.181.254', visits_ip: 3 },
  { _id: '129.10.231.17', visits_ip: 25 },
  { _id: '119.118.239.160', visits_ip: 1 },
  { _id: '120.198.124.121', visits_ip: 1 },
  { _id: '211.230.149.25', visits_ip: 1 },
  { _id: '129.10.196.107', visits_ip: 11 },
  { _id: '61.157.236.176', visits_ip: 1 },
  { _id: '95.0.87.28', visits_ip: 1 },
  { _id: '216.15.126.209', visits_ip: 1 },
  { _id: '210.75.240.133', visits_ip: 1 }
]
```