Aniruddha_Tambe_002101113_a02.md - Grip

PART 1 - READING ASSIGNMENT

https://learning.oreilly.com/library/view/mongodb-in-action/9781617291609/?ar

Chapter 4. Document-oriented data Chapter 5. Constructing queries Chapter 6. Aggregation

Note: If you cannot access the chapters, enter your neu email as @northeastern.edu instead of @husky.neu.edu

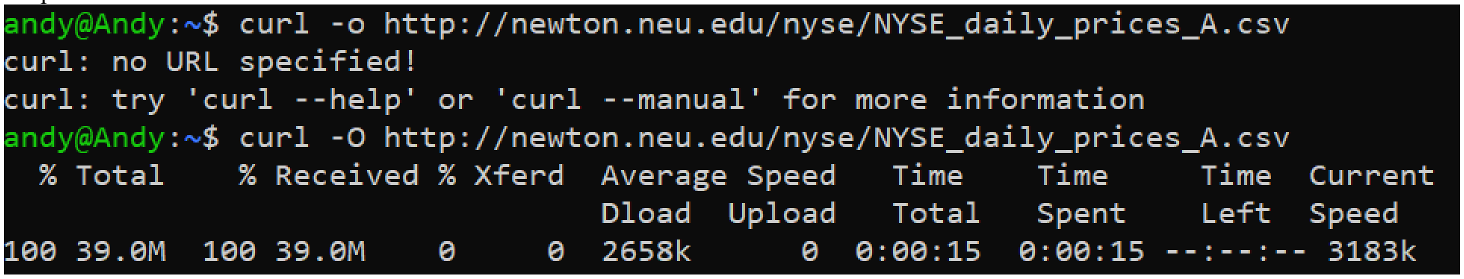
PART 2 - PROGRAMMING ASSIGNMENT

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

NYSE Dataset Link: http://newton.neu.edu/nyse/NYSE_daily_prices_A.csv

Downloading the dataset:

```
sudo apt install -y curl
curl -O http://newton.neu.edu/nyse/NYSE_daily_prices_A.csv
```

Output:


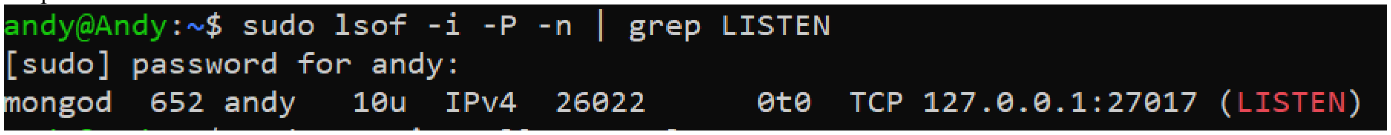
```
andy@Andy:~$ curl -o http://newton.neu.edu/nyse/NYSE_daily_prices_A.csv
curl: no URL specified!
curl: try 'curl --help' or 'curl --manual' for more information
andy@Andy:~$ curl -O http://newton.neu.edu/nyse/NYSE_daily_prices_A.csv
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 39.0M  100 39.0M    0     0  2658k      0  0:00:15  0:00:15 --:--:-- 3183k
```

Note: Before we begin, we need to make sure that mongodb is installed on Linux subsystem.Run below lines of code on a WSL:

```
sudo apt install -y mongodb
cd ..
sudo mkdir /usr/bin/data/db
sudo chmod 777 /usr/bin/data/db
/usr/bin/mongod --dbpath=/usr/bin/data/db
```

This should start the mongodb daemon-process. Run below code for checking if db is running:

```
sudo lsof -i -P -n|grep LISTEN
```

Output:


```
andy@Andy:~$ sudo lsof -i -P -n | grep LISTEN
[sudo] password for andy:
mongod  652 andy   10u  IPv4  26022      0t0  TCP 127.0.0.1:27017 (LISTEN)
```

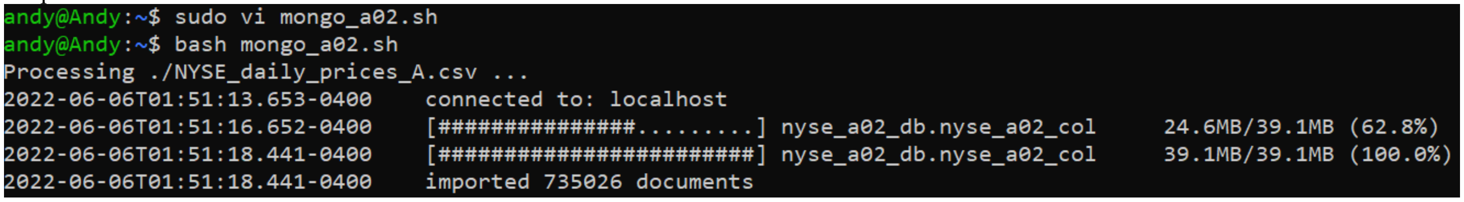
Scripting the .sh file:

```
#!/bin/bash
FILES=./NYSE_daily_prices_A.csv
for f in $FILES
do
  echo "Processing $f ..."
  # set MONGODB_HOME environment
  MONGODB_HOME=/usr/bin
  $MONGODB_HOME/mongoimport --type csv --db nyse_a02_db --collection nyse_a02_col --headerline $f
done
```

Running the bash file:

```
sudo vi mongo_a02.sh
bash mongo_a02.sh
```

Output:



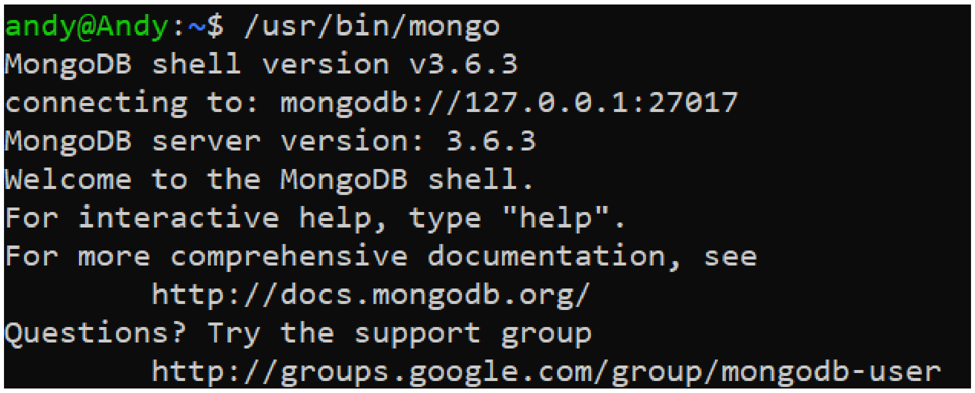
```
andy@Andy:~$ sudo vi mongo_a02.sh
andy@Andy:~$ bash mongo_a02.sh
Processing ./NYSE_daily_prices_A.csv ...
2022-06-06T01:51:13.653-0400    connected to: localhost
2022-06-06T01:51:16.652-0400    [###############.........] nyse_a02_db.nyse_a02_col     24.6MB/39.1MB (62.8%)
2022-06-06T01:51:18.441-0400    [########################] nyse_a02_db.nyse_a02_col     39.1MB/39.1MB (100.0%)
2022-06-06T01:51:18.441-0400    imported 735026 documents
```

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

Start the mongodb shell:

```
cd ~
/usr/bin/mongo
```

Output:

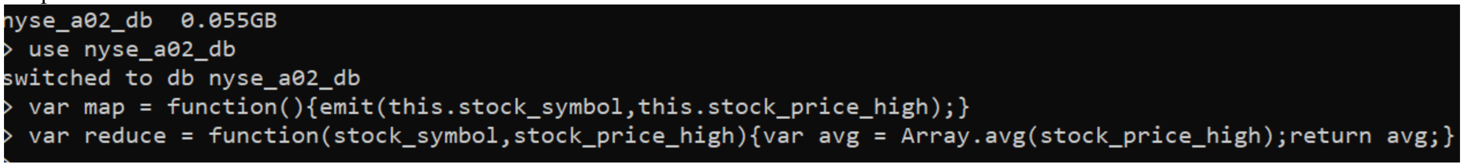


```
andy@Andy:~$ /usr/bin/mongo
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.3
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
```

Define map & reduce function:

```
use nyse_a02_db
var map = function(){emit(this.stock_symbol,this.stock_price_high);}
var reduce = function(stock_symbol,stock_price_high){var avg = Array.avg(stock_price_high);return avg;}
```

Output:

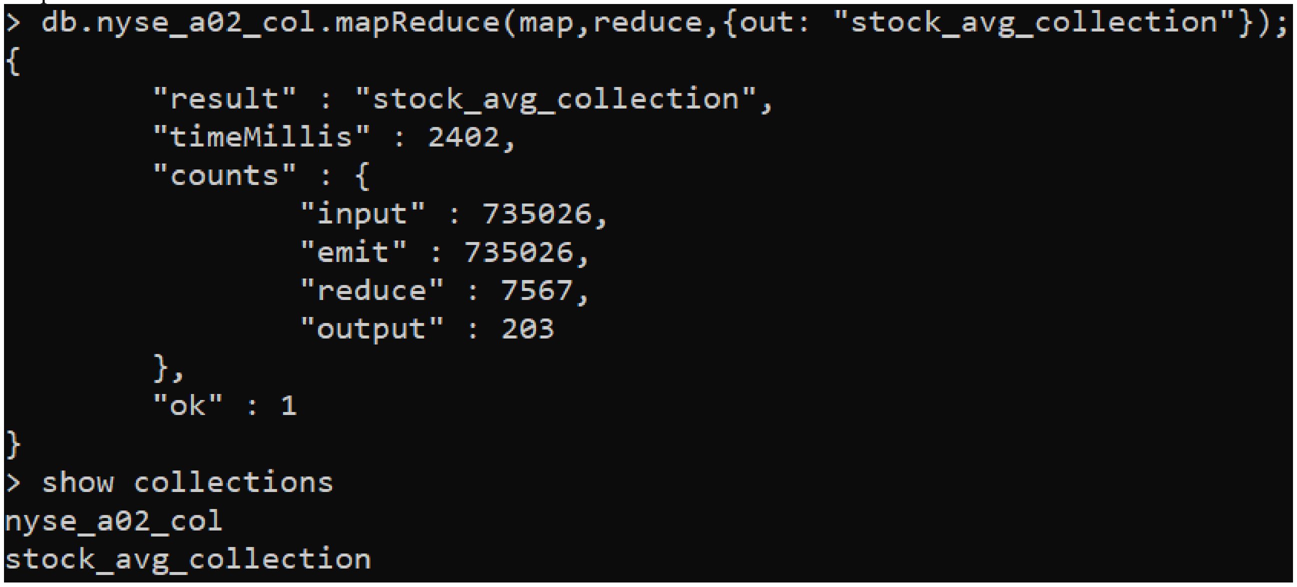


```
nyse_a02_db  0.055GB
> use nyse_a02_db
switched to db nyse_a02_db
> var map = function(){emit(this.stock_symbol,this.stock_price_high);}
> var reduce = function(stock_symbol,stock_price_high){var avg = Array.avg(stock_price_high);return avg;}
```

Run mapReduce function:

```
db.nyse_a02_col.mapReduce(map,reduce,{out: "stock_avg_collection"});
show collections
```

Output:

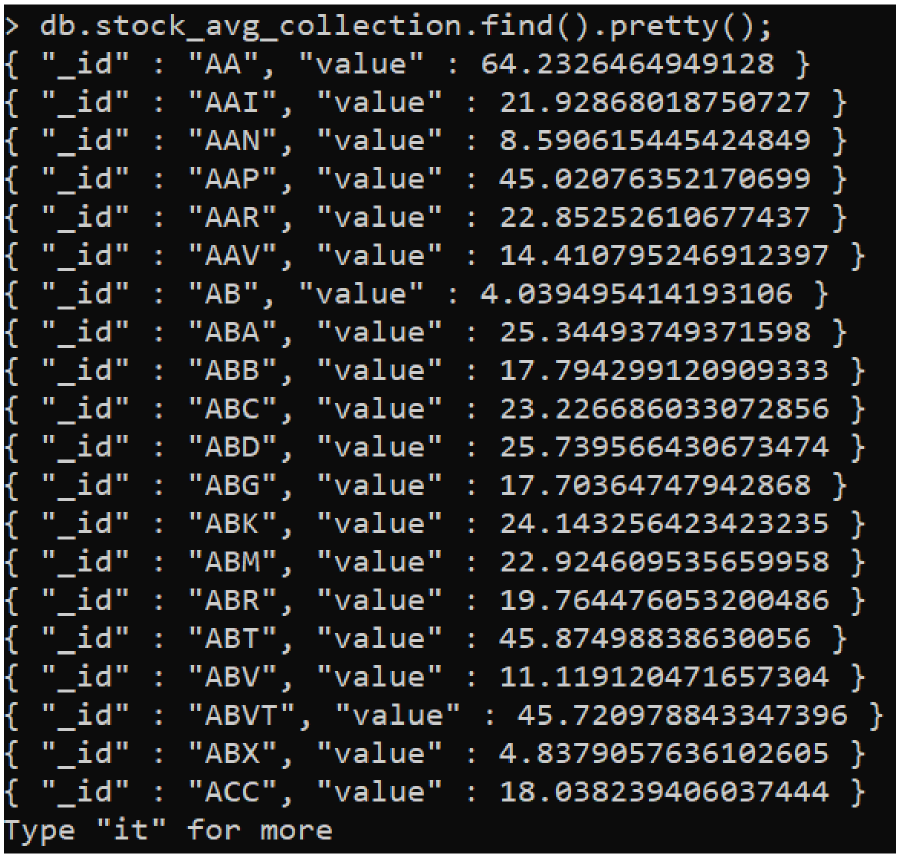


```
> db.nyse_a02_col.mapReduce(map,reduce,{out: "stock_avg_collection"});
{
        "result" : "stock_avg_collection",
        "timeMillis" : 2402,
        "counts" : {
                "input" : 735026,
                "emit" : 735026,
                "reduce" : 7567,
                "output" : 203
        },
        "ok" : 1
}
> show collections
nyse_a02_col
stock_avg_collection
```

Printing mapReduce output:

```
db.stock_avg_collection.find().pretty();
```

Output:



```
> db.stock_avg_collection.find().pretty();
{ "_id" : "AA", "value" : 64.2326464949128 }
{ "_id" : "AAI", "value" : 21.92868018750727 }
{ "_id" : "AAN", "value" : 8.590615445424849 }
{ "_id" : "AAP", "value" : 45.02076352170699 }
{ "_id" : "AAR", "value" : 22.85252610677437 }
{ "_id" : "AAV", "value" : 14.410795246912397 }
{ "_id" : "AB", "value" : 4.039495414193106 }
{ "_id" : "ABA", "value" : 25.34493749371598 }
{ "_id" : "ABB", "value" : 17.794299120909333 }
{ "_id" : "ABC", "value" : 23.226686033072856 }
{ "_id" : "ABD", "value" : 25.739566430673474 }
{ "_id" : "ABG", "value" : 17.70364747942868 }
{ "_id" : "ABK", "value" : 24.143256423423235 }
{ "_id" : "ABM", "value" : 22.924609535659958 }
{ "_id" : "ABR", "value" : 19.764476053200486 }
{ "_id" : "ABT", "value" : 45.87498838630056 }
{ "_id" : "ABV", "value" : 11.119120471657304 }
{ "_id" : "ABVT", "value" : 45.720978843347396 }
{ "_id" : "ABX", "value" : 4.8379057636102605 }
{ "_id" : "ACC", "value" : 18.038239406037444 }
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.

Redefine map,reduce & finalizer:

Map function:

```
var map_final = function(){
        emit(this.stock_symbol,{sum: this.stock_price_high,count:1});
}
```

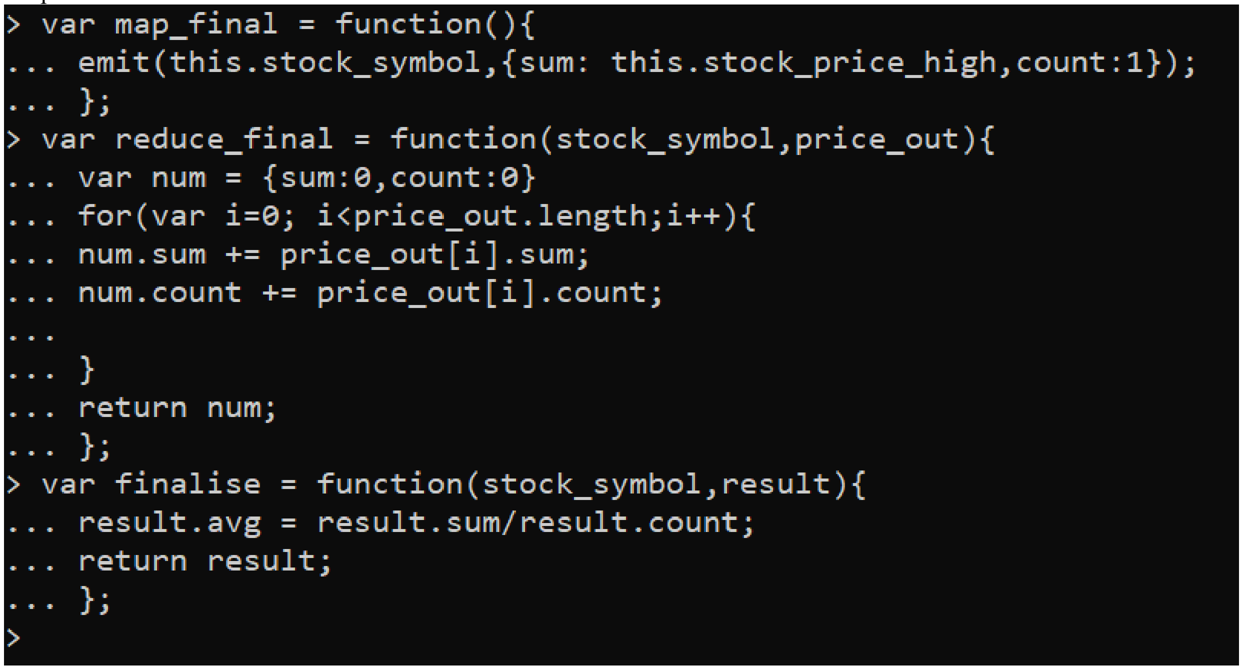
Reduce function:

```
var reduce_final = function(stock_symbol,price_out){
        var num = {sum:0,count:0}
        for(var i=0; i<price_out.length;i++){
                num.sum += price_out[i].sum;
                num.count += price_out[i].count;

        }
        return num;
};
```

Finalize function:

```
var finalise = function(stock_symbol,result){
        result.avg = result.sum/result.count;
        return result.avg;
};
```

Output:

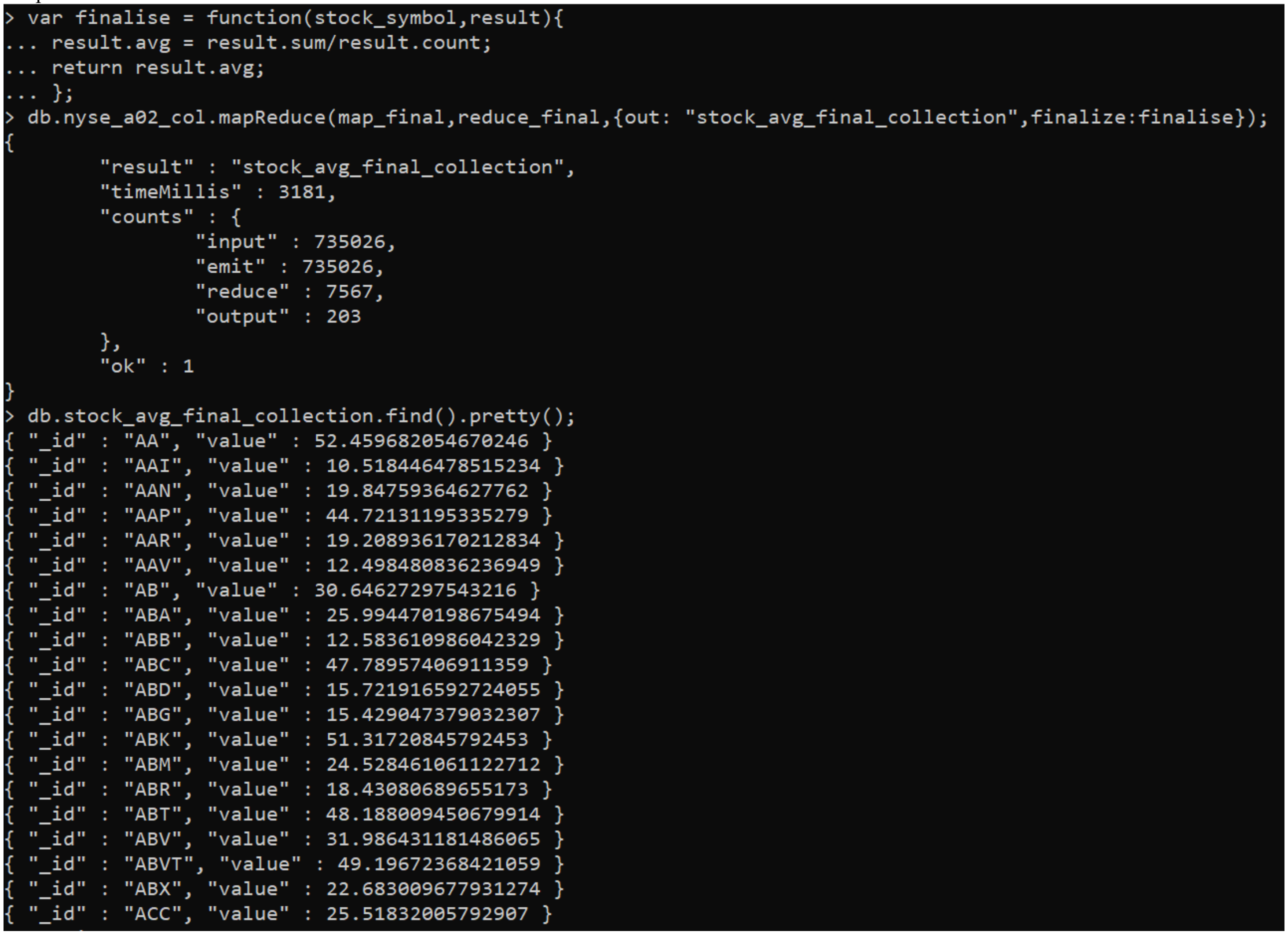


```
> var map_final = function(){
... emit(this.stock_symbol,{sum: this.stock_price_high,count:1});
... };
> var reduce_final = function(stock_symbol,price_out){
... var num = {sum:0,count:0}
... for(var i=0; i<price_out.length;i++){
... num.sum += price_out[i].sum;
... num.count += price_out[i].count;
...
... }
... return num;
... };
> var finalise = function(stock_symbol,result){
... result.avg = result.sum/result.count;
... return result;
... };
>
```

Run mapReduce with finalize:

```
db.nyse_a02_col.mapReduce(map_final,reduce_final,{out: "stock_avg_final_collection",finalize:finalise});
```

Output:



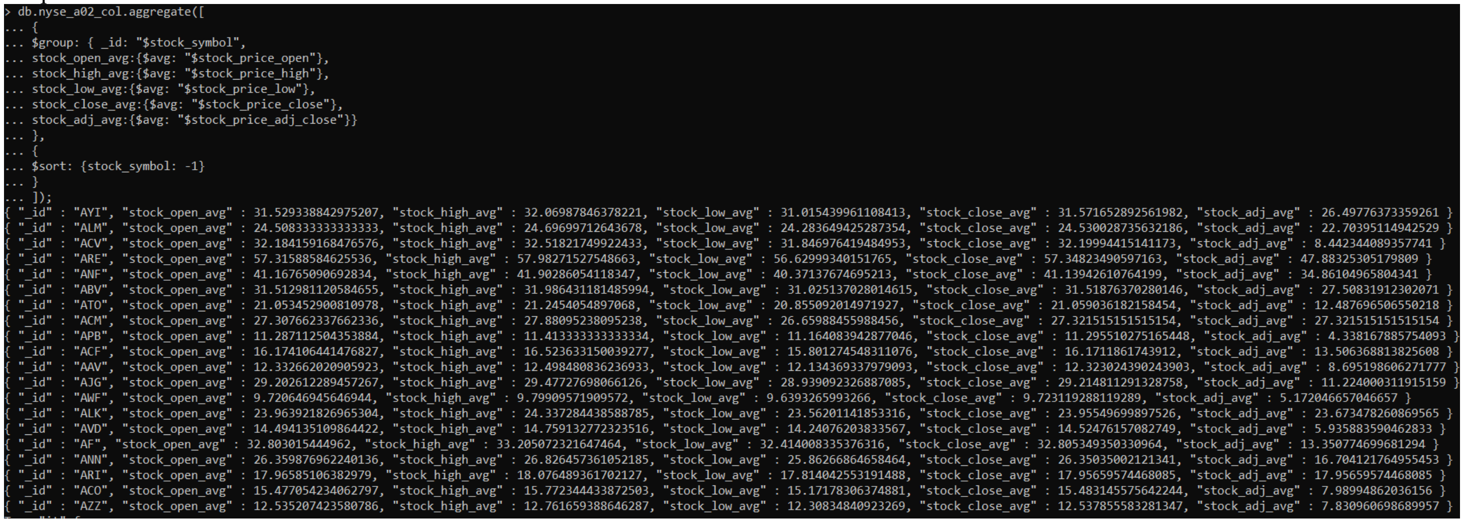
```
> var finalise = function(stock_symbol,result){
... result.avg = result.sum/result.count;
... return result.avg;
... };
> db.nyse_a02_col.mapReduce(map_final,reduce_final,{out: "stock_avg_final_collection",finalize:finalise});
{
        "result" : "stock_avg_final_collection",
        "timeMillis" : 3181,
        "counts" : {
                "input" : 735026,
                "emit" : 735026,
                "reduce" : 7567,
                "output" : 203
        },
        "ok" : 1
}
> db.stock_avg_final_collection.find().pretty();
{ "_id" : "AA", "value" : 52.459682054670246 }
{ "_id" : "AAI", "value" : 10.518446478515234 }
{ "_id" : "AAN", "value" : 19.84759364627762 }
{ "_id" : "AAP", "value" : 44.72131195335279 }
{ "_id" : "AAR", "value" : 19.208936170212834 }
{ "_id" : "AAV", "value" : 12.498480836236949 }
{ "_id" : "AB", "value" : 30.64627297543216 }
{ "_id" : "ABA", "value" : 25.994470198675494 }
{ "_id" : "ABB", "value" : 12.583610986042329 }
{ "_id" : "ABC", "value" : 47.78957406911359 }
{ "_id" : "ABD", "value" : 15.721916592724055 }
{ "_id" : "ABG", "value" : 15.429047379032307 }
{ "_id" : "ABK", "value" : 51.31720845792453 }
{ "_id" : "ABM", "value" : 24.528461061122712 }
{ "_id" : "ABR", "value" : 18.43080689655173 }
{ "_id" : "ABT", "value" : 48.188009450679914 }
{ "_id" : "ABV", "value" : 31.986431181486065 }
{ "_id" : "ABVT", "value" : 49.19672368421059 }
{ "_id" : "ABX", "value" : 22.683009677931274 }
{ "_id" : "ACC", "value" : 25.51832005792907 }
```

PART 4. Calculate the average stock price of each price of all stocks using $avg aggregation.

Using the `aggregate()` function of mongodb for calculation the average stock price:

```
db.nyse_a02_col.aggregate([
        {
                $group: { _id: "$stock_symbol",
                stock_open_avg:{$avg: "$stock_price_open"},
                stock_high_avg:{$avg: "$stock_price_high"},
                stock_low_avg:{$avg: "$stock_price_low"},
                stock_close_avg:{$avg: "$stock_price_close"},
                stock_adj_avg:{$avg: "$stock_price_adj_close"}}
        },
        {
                $sort: {stock_symbol: -1}
        }
]);
```

Output:



```
> db.nyse_a02_col.aggregate([
... {
... $group: { _id: "$stock_symbol",
... stock_open_avg:{$avg: "$stock_price_open"},
... stock_high_avg:{$avg: "$stock_price_high"},
... stock_low_avg:{$avg: "$stock_price_low"},
... stock_close_avg:{$avg: "$stock_price_close"},
... stock_adj_avg:{$avg: "$stock_price_adj_close"}}
... },
... {
... $sort: {stock_symbol: -1}
... }
... ]);
{ "_id" : "AYI", "stock_open_avg" : 31.529338842975207, "stock_high_avg" : 32.06987846378221, "stock_low_avg" : 31.015439961108413, "stock_close_avg" : 31.571652892561982, "stock_adj_avg" : 26.49776373359261 }
{ "_id" : "ALM", "stock_open_avg" : 24.508333333333333, "stock_high_avg" : 24.69699712643678, "stock_low_avg" : 24.283649425287354, "stock_close_avg" : 24.530028735632186, "stock_adj_avg" : 22.70395114942529 }
{ "_id" : "ACV", "stock_open_avg" : 32.184159168476576, "stock_high_avg" : 32.51821749922433, "stock_low_avg" : 31.846976419484953, "stock_close_avg" : 32.19994415141173, "stock_adj_avg" : 8.442344089357741 }
{ "_id" : "ARE", "stock_open_avg" : 57.31588584625536, "stock_high_avg" : 57.98271527548663, "stock_low_avg" : 56.62999340151765, "stock_close_avg" : 57.34823490597163, "stock_adj_avg" : 47.88325305179809 }
{ "_id" : "ANF", "stock_open_avg" : 41.16765090692834, "stock_high_avg" : 41.90286054118347, "stock_low_avg" : 40.37137674695213, "stock_close_avg" : 41.13942610764199, "stock_adj_avg" : 34.86104965804341 }
{ "_id" : "ABV", "stock_open_avg" : 31.512981120584655, "stock_high_avg" : 31.986431181485994, "stock_low_avg" : 31.025137028014615, "stock_close_avg" : 31.51876370280146, "stock_adj_avg" : 27.50831912302071 }
{ "_id" : "ATO", "stock_open_avg" : 21.053452900810978, "stock_high_avg" : 21.2454054897068, "stock_low_avg" : 20.855092014971927, "stock_close_avg" : 21.059036182158454, "stock_adj_avg" : 12.487696506550218 }
{ "_id" : "ACM", "stock_open_avg" : 27.307662337662336, "stock_high_avg" : 27.88095238095238, "stock_low_avg" : 26.65988455988456, "stock_close_avg" : 27.321515151515154, "stock_adj_avg" : 27.321515151515154 }
{ "_id" : "APB", "stock_open_avg" : 11.287112504353884, "stock_high_avg" : 11.413333333333334, "stock_low_avg" : 11.164083942877046, "stock_close_avg" : 11.295510275165448, "stock_adj_avg" : 4.338167885754093 }
{ "_id" : "ACF", "stock_open_avg" : 16.174106441476827, "stock_high_avg" : 16.523633150039277, "stock_low_avg" : 15.801274548311076, "stock_close_avg" : 16.1711861743912, "stock_adj_avg" : 13.506368813825608 }
{ "_id" : "AAV", "stock_open_avg" : 12.332662020905923, "stock_high_avg" : 12.498480836236933, "stock_low_avg" : 12.134369337979093, "stock_close_avg" : 12.323024390243903, "stock_adj_avg" : 8.695198606271777 }
{ "_id" : "AJG", "stock_open_avg" : 29.202612289457267, "stock_high_avg" : 29.47727698066126, "stock_low_avg" : 28.939092326887085, "stock_close_avg" : 29.214811291328758, "stock_adj_avg" : 11.224000311915159 }
{ "_id" : "AWF", "stock_open_avg" : 9.720646945646944, "stock_high_avg" : 9.79909571909572, "stock_low_avg" : 9.6393265993266, "stock_close_avg" : 9.723119288119289, "stock_adj_avg" : 5.172046657046657 }
{ "_id" : "ALK", "stock_open_avg" : 23.963921826965304, "stock_high_avg" : 24.337284438588785, "stock_low_avg" : 23.56201141853316, "stock_close_avg" : 23.95549699897526, "stock_adj_avg" : 23.673478260869565 }
{ "_id" : "AVD", "stock_open_avg" : 14.494135109864422, "stock_high_avg" : 14.759132772323516, "stock_low_avg" : 14.24076203833567, "stock_close_avg" : 14.52476157082749, "stock_adj_avg" : 5.935883590462833 }
{ "_id" : "AF", "stock_open_avg" : 32.803015444962, "stock_high_avg" : 33.205072321647464, "stock_low_avg" : 32.414008335376316, "stock_close_avg" : 32.805349350330964, "stock_adj_avg" : 13.350774699681294 }
{ "_id" : "ANN", "stock_open_avg" : 26.359876962240136, "stock_high_avg" : 26.826457361052185, "stock_low_avg" : 25.86266864658464, "stock_close_avg" : 26.35035002121341, "stock_adj_avg" : 16.704121764955453 }
{ "_id" : "ARI", "stock_open_avg" : 17.96585106382979, "stock_high_avg" : 18.076489361702127, "stock_low_avg" : 17.814042553191488, "stock_close_avg" : 17.95659574468085, "stock_adj_avg" : 17.95659574468085 }
{ "_id" : "ACO", "stock_open_avg" : 15.477054234062797, "stock_high_avg" : 15.772344433872503, "stock_low_avg" : 15.17178306374881, "stock_close_avg" : 15.483145575642244, "stock_adj_avg" : 7.98994862036156 }
{ "_id" : "AZZ", "stock_open_avg" : 12.535207423580786, "stock_high_avg" : 12.761659388646287, "stock_low_avg" : 12.30834840923269, "stock_close_avg" : 12.537855583281347, "stock_adj_avg" : 7.830960698689957 }
```

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
```

Let us add headers to each csv file

```
sed -i '1s/^/user_id,gender,age,occupation,zipcode\n/' users.csv
sed -i '1s/^/movie_id,title,genre\n/' movies.csv
sed -i '1s/^/user_id,movie_id,rating,timestamp\n/' ratings.csv
```

Importing the csv files into collections

```
cd ~
/usr/bin/mongoimport --type csv --db movielens --collection ratings --headerline ./ml-1m/ratings.csv
/usr/bin/mongoimport --type csv --db movielens --collection users --headerline ./ml-1m/users.csv
/usr/bin/mongoimport --type csv --db movielens --collection movies --headerline ./ml-1m/movies.csv
```

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

Using MapReduce:

Define a map function to emit the gender

```
var map = function(){
        emit(this.gender,{count:1});
};
```

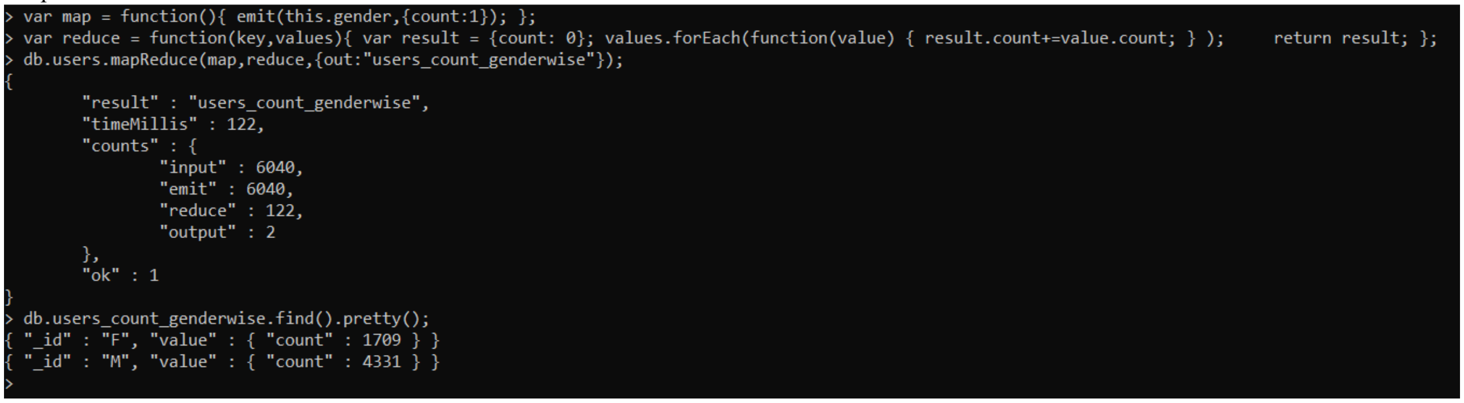
Define a reduce function to count

```
var reduce = function(key,values){
        var result = {count: 0};
        values.forEach(function(value) { result.count += value.count; } );
    return result;
};
```

Run the mapReduce command:

```
use movielens
db.users.mapReduce(map,reduce);
```

Output:

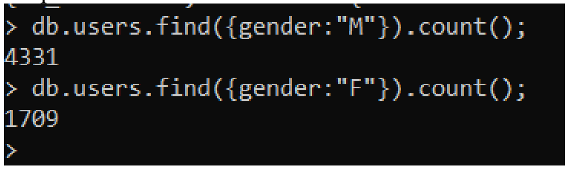


```
> var map = function(){ emit(this.gender,{count:1}); };
> var reduce = function(key,values){ var result = {count: 0}; values.forEach(function(value) { result.count+=value.count; } );     return result; };
> db.users.mapReduce(map,reduce,{out:"users_count_genderwise"});
{
        "result" : "users_count_genderwise",
        "timeMillis" : 122,
        "counts" : {
                "input" : 6040,
                "emit" : 6040,
                "reduce" : 122,
                "output" : 2
        },
        "ok" : 1
}
> db.users_count_genderwise.find().pretty();
{ "_id" : "F", "value" : { "count" : 1709 } }
{ "_id" : "M", "value" : { "count" : 4331 } }
>
```

Using count():

```
db.users.find({gender:"M"}).count();
db.users.find({gender:"F"}).count();
```

Output:



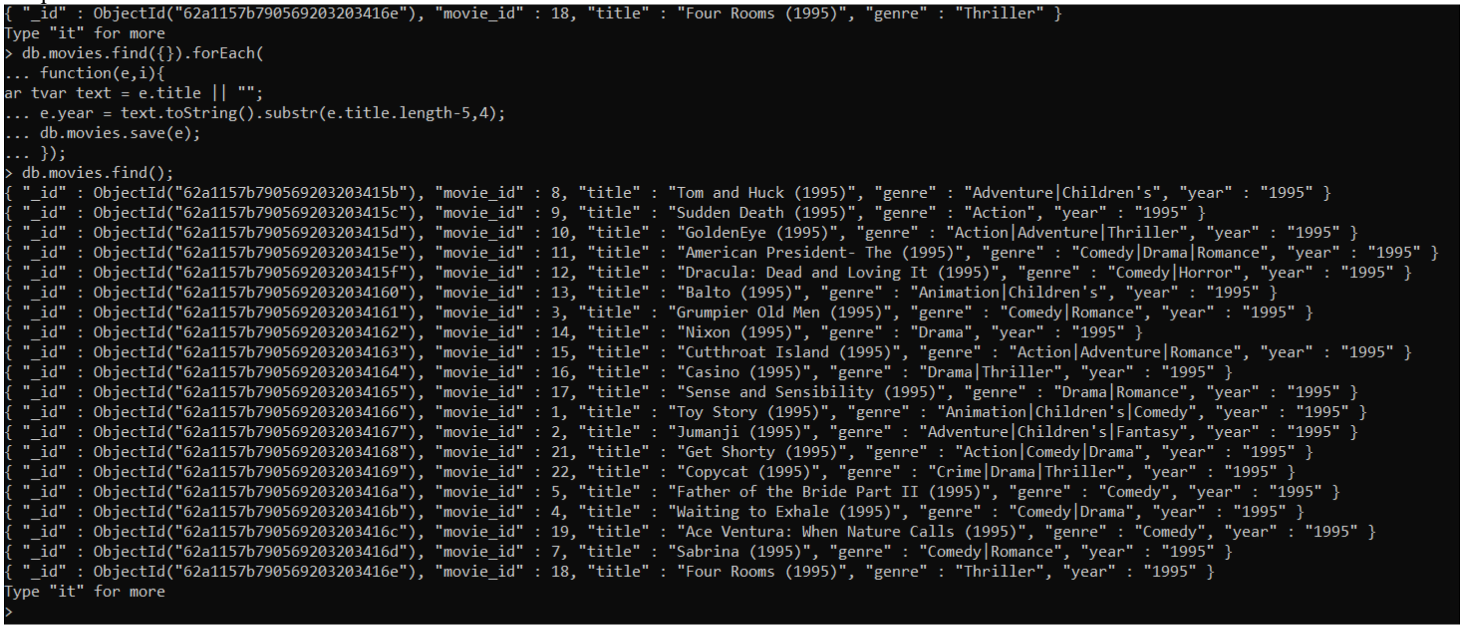
```
> db.users.find({gender:"M"}).count();
4331
> db.users.find({gender:"F"}).count();
1709
>
```

Q.Find the number of Movies per year using MapReduce

Lets add a field for adding year in the 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);
});
```

Output:



```
{ "_id" : ObjectId("62a1157b790569203203416e"), "movie_id" : 18, "title" : "Four Rooms (1995)", "genre" : "Thriller" }
Type "it" for more
> db.movies.find({}).forEach(
... function(e,i){
ar tvar text = e.title || "";
... e.year = text.toString().substr(e.title.length-5,4);
... db.movies.save(e);
... });
> db.movies.find();
{ "_id" : ObjectId("62a1157b790569203203415b"), "movie_id" : 8, "title" : "Tom and Huck (1995)", "genre" : "Adventure|Children's", "year" : "1995" }
{ "_id" : ObjectId("62a1157b790569203203415c"), "movie_id" : 9, "title" : "Sudden Death (1995)", "genre" : "Action", "year" : "1995" }
{ "_id" : ObjectId("62a1157b790569203203415d"), "movie_id" : 10, "title" : "GoldenEye (1995)", "genre" : "Action|Adventure|Thriller", "year" : "1995" }
{ "_id" : ObjectId("62a1157b790569203203415e"), "movie_id" : 11, "title" : "American President- The (1995)", "genre" : "Comedy|Drama|Romance", "year" : "1995" }
{ "_id" : ObjectId("62a1157b790569203203415f"), "movie_id" : 12, "title" : "Dracula: Dead and Loving It (1995)", "genre" : "Comedy|Horror", "year" : "1995" }
{ "_id" : ObjectId("62a1157b7905692032034160"), "movie_id" : 13, "title" : "Balto (1995)", "genre" : "Animation|Children's", "year" : "1995" }
{ "_id" : ObjectId("62a1157b7905692032034161"), "movie_id" : 3, "title" : "Grumpier Old Men (1995)", "genre" : "Comedy|Romance", "year" : "1995" }
{ "_id" : ObjectId("62a1157b7905692032034162"), "movie_id" : 14, "title" : "Nixon (1995)", "genre" : "Drama", "year" : "1995" }
{ "_id" : ObjectId("62a1157b7905692032034163"), "movie_id" : 15, "title" : "Cutthroat Island (1995)", "genre" : "Action|Adventure|Romance", "year" : "1995" }
{ "_id" : ObjectId("62a1157b7905692032034164"), "movie_id" : 16, "title" : "Casino (1995)", "genre" : "Drama|Thriller", "year" : "1995" }
{ "_id" : ObjectId("62a1157b7905692032034165"), "movie_id" : 17, "title" : "Sense and Sensibility (1995)", "genre" : "Drama|Romance", "year" : "1995" }
{ "_id" : ObjectId("62a1157b7905692032034166"), "movie_id" : 1, "title" : "Toy Story (1995)", "genre" : "Animation|Children's|Comedy", "year" : "1995" }
{ "_id" : ObjectId("62a1157b7905692032034167"), "movie_id" : 2, "title" : "Jumanji (1995)", "genre" : "Adventure|Children's|Fantasy", "year" : "1995" }
{ "_id" : ObjectId("62a1157b7905692032034168"), "movie_id" : 21, "title" : "Get Shorty (1995)", "genre" : "Action|Comedy|Drama", "year" : "1995" }
{ "_id" : ObjectId("62a1157b7905692032034169"), "movie_id" : 22, "title" : "Copycat (1995)", "genre" : "Crime|Drama|Thriller", "year" : "1995" }
{ "_id" : ObjectId("62a1157b790569203203416a"), "movie_id" : 5, "title" : "Father of the Bride Part II (1995)", "genre" : "Comedy", "year" : "1995" }
{ "_id" : ObjectId("62a1157b790569203203416b"), "movie_id" : 4, "title" : "Waiting to Exhale (1995)", "genre" : "Comedy|Drama", "year" : "1995" }
{ "_id" : ObjectId("62a1157b790569203203416c"), "movie_id" : 19, "title" : "Ace Ventura: When Nature Calls (1995)", "genre" : "Comedy", "year" : "1995" }
{ "_id" : ObjectId("62a1157b790569203203416d"), "movie_id" : 7, "title" : "Sabrina (1995)", "genre" : "Comedy|Romance", "year" : "1995" }
{ "_id" : ObjectId("62a1157b790569203203416e"), "movie_id" : 18, "title" : "Four Rooms (1995)", "genre" : "Thriller", "year" : "1995" }
Type "it" for more
>
```

Define a map function:

```
var map = function(){
        emit(this.year,{"count":1});
};
```

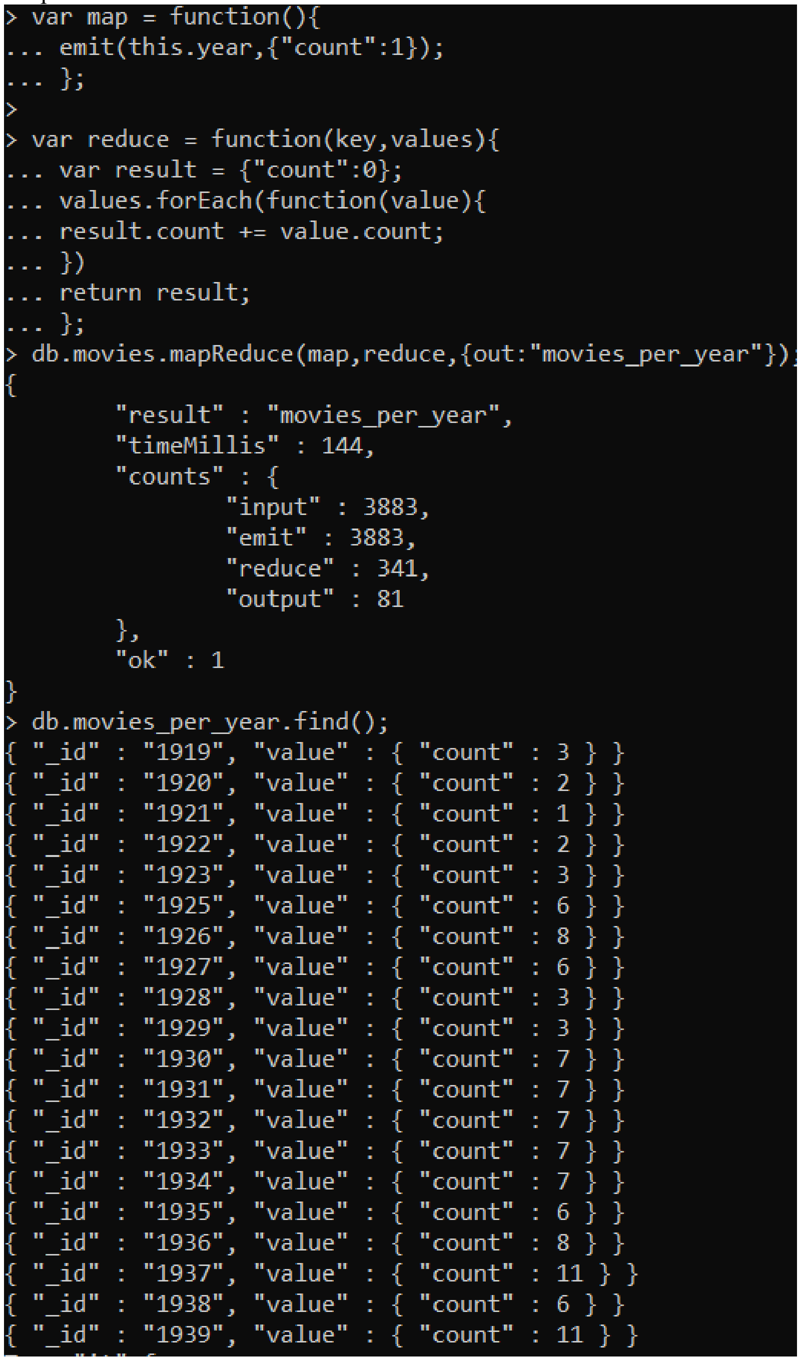
Define a reduce function:

```
var reduce = function(key,values){
        var result = {"count":0};
        values.forEach(function(value){
                result.count += value.count;
        })
        return result;
};
```

Run the mapReduce function

```
use movielens
db.movies.mapReduce(map,reduce,{out:"movies_per_year"});
```

Output:



```
> var map = function(){
... emit(this.year,{"count":1});
... };
>
> var reduce = function(key,values){
... var result = {"count":0};
... values.forEach(function(value){
... result.count += value.count;
... })
... return result;
... };
> db.movies.mapReduce(map,reduce,{out:"movies_per_year"});
{
        "result" : "movies_per_year",
        "timeMillis" : 144,
        "counts" : {
                "input" : 3883,
                "emit" : 3883,
                "reduce" : 341,
                "output" : 81
        },
        "ok" : 1
}
> db.movies_per_year.find();
{ "_id" : "1919", "value" : { "count" : 3 } }
{ "_id" : "1920", "value" : { "count" : 2 } }
{ "_id" : "1921", "value" : { "count" : 1 } }
{ "_id" : "1922", "value" : { "count" : 2 } }
{ "_id" : "1923", "value" : { "count" : 3 } }
{ "_id" : "1925", "value" : { "count" : 6 } }
{ "_id" : "1926", "value" : { "count" : 8 } }
{ "_id" : "1927", "value" : { "count" : 6 } }
{ "_id" : "1928", "value" : { "count" : 3 } }
{ "_id" : "1929", "value" : { "count" : 3 } }
{ "_id" : "1930", "value" : { "count" : 7 } }
{ "_id" : "1931", "value" : { "count" : 7 } }
{ "_id" : "1932", "value" : { "count" : 7 } }
{ "_id" : "1933", "value" : { "count" : 7 } }
{ "_id" : "1934", "value" : { "count" : 7 } }
{ "_id" : "1935", "value" : { "count" : 6 } }
{ "_id" : "1936", "value" : { "count" : 8 } }
{ "_id" : "1937", "value" : { "count" : 11 } }
{ "_id" : "1938", "value" : { "count" : 6 } }
{ "_id" : "1939", "value" : { "count" : 11 } }
```

Q.Find the number of Movies per rating using MapReduce

Define map function

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

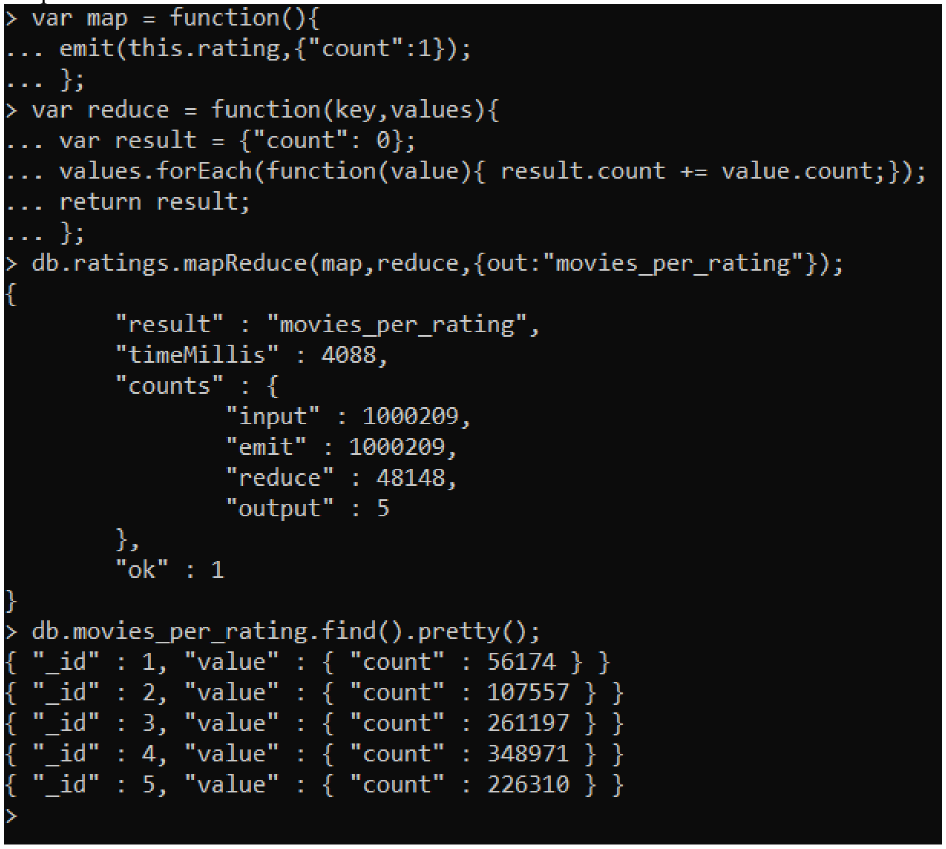
Define a reduce function

```
var reduce = function(key,values){
        var result = {"count": 0};
        values.forEach(function(value){ result.count += value.count;});
        return result;
};
```

Execute the mapReduce function

```
use movielens
db.ratings.mapReduce(map,reduce,{out:"movies_per_rating"});
```

Output:



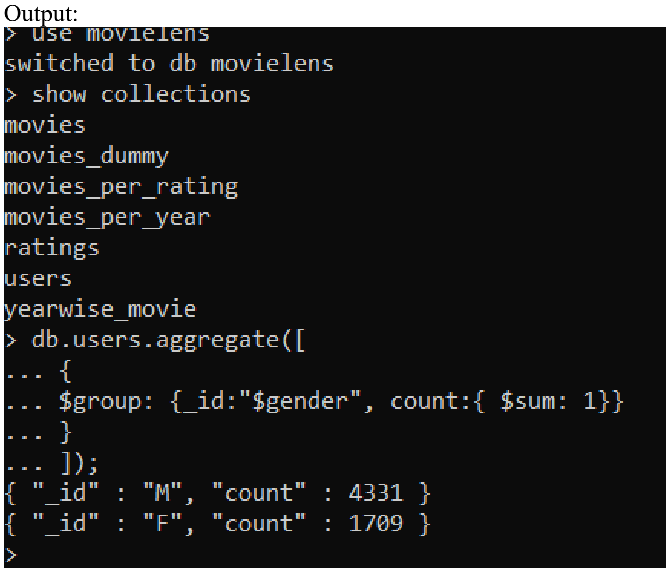
```
> var map = function(){
... emit(this.rating,{"count":1});
... };
> var reduce = function(key,values){
... var result = {"count": 0};
... values.forEach(function(value){ result.count += value.count;});
... return result;
... };
> db.ratings.mapReduce(map,reduce,{out:"movies_per_rating"});
{
        "result" : "movies_per_rating",
        "timeMillis" : 4088,
        "counts" : {
                "input" : 1000209,
                "emit" : 1000209,
                "reduce" : 48148,
                "output" : 5
        },
        "ok" : 1
}
> db.movies_per_rating.find().pretty();
{ "_id" : 1, "value" : { "count" : 56174 } }
{ "_id" : 2, "value" : { "count" : 107557 } }
{ "_id" : 3, "value" : { "count" : 261197 } }
{ "_id" : 4, "value" : { "count" : 348971 } }
{ "_id" : 5, "value" : { "count" : 226310 } }
>
```

PART 5.2 - PROGRAMMING ASSIGNMENT

Q. Find the number Females and Males from the users collection using aggregate.

```
db.users.aggregate([
        {
                $group: {_id:"$gender", count:{ $sum: 1}}
        }
]);
```

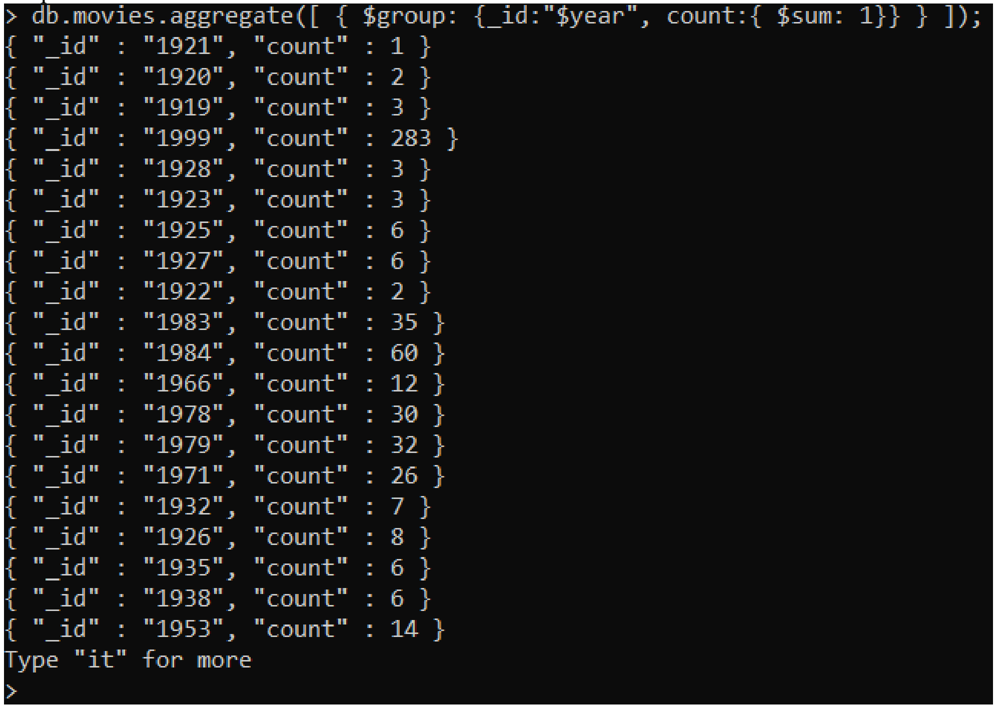
Output:


```
> use movielens
switched to db movielens
> show collections
movies
movies_dummy
movies_per_rating
movies_per_year
ratings
users
yearwise_movie
> db.users.aggregate([
... {
... $group: {_id:"$gender", count:{ $sum: 1}}
... }
... ]);
{ "_id" : "M", "count" : 4331 }
{ "_id" : "F", "count" : 1709 }
>
```

Q.Find the number of Movies per year using MapReduce

```
db.movies.aggregate([
        {
                $group: {_id:"$year", count:{ $sum: 1}}
        }
]);
```

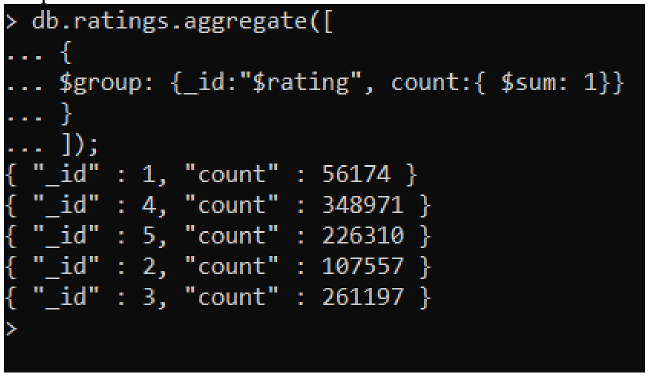
Output:


```
> db.movies.aggregate([ { $group: {_id:"$year", count:{ $sum: 1}} } ]);
{ "_id" : "1921", "count" : 1 }
{ "_id" : "1920", "count" : 2 }
{ "_id" : "1919", "count" : 3 }
{ "_id" : "1999", "count" : 283 }
{ "_id" : "1928", "count" : 3 }
{ "_id" : "1923", "count" : 3 }
{ "_id" : "1925", "count" : 6 }
{ "_id" : "1927", "count" : 6 }
{ "_id" : "1922", "count" : 2 }
{ "_id" : "1983", "count" : 35 }
{ "_id" : "1984", "count" : 60 }
{ "_id" : "1966", "count" : 12 }
{ "_id" : "1978", "count" : 30 }
{ "_id" : "1979", "count" : 32 }
{ "_id" : "1971", "count" : 26 }
{ "_id" : "1932", "count" : 7 }
{ "_id" : "1926", "count" : 8 }
{ "_id" : "1935", "count" : 6 }
{ "_id" : "1938", "count" : 6 }
{ "_id" : "1953", "count" : 14 }
Type "it" for more
>
```

Q.Find the number of Movies per rating using MapReduce

```
db.ratings.aggregate([
        {
                $group: {_id:"$rating", count:{ $sum: 1}}
        }
]);
```

Output:



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

PART 6 - PROGRAMMING ASSIGNMENT [access.log Download access.log]

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

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

Solution:

Download dataset:

```
cd ~
curl -O https://raw.githubusercontent.com/tambeani/INFO7250---Engineering-of-Big-Data-Systems/main/dataset/access.log
```

Processing the data

```
cp access.log access.csv
sed -i 's/ - - /,/g' access.csv
sed -i 's/ -/-/g' access.csv
sed -i 's/"//g' access.csv
sed -i 's/ /,/g' access.csv
```

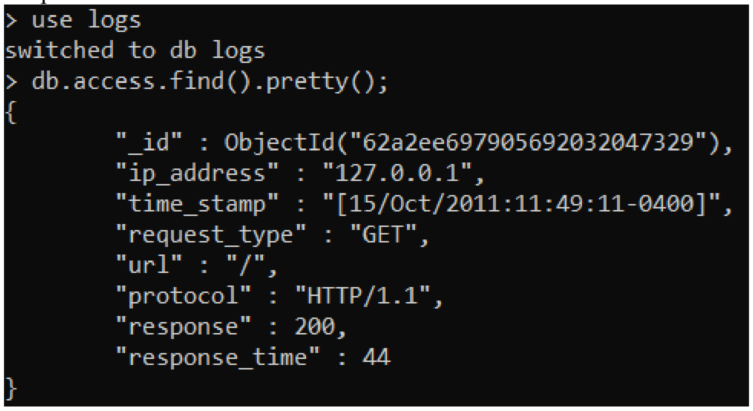
Adding headers

```
sed -i '1s/^/ip_address,time_stamp,request_type,url,protocol,response,response_time\n/' access.csv
```

Importing the dataset into mongodb:

```
/usr/bin/mongoimport --type csv --db logs --collection access --headerline ./access.csv
```

Output:



```
> use logs
switched to db logs
> db.access.find().pretty();
{
        "_id" : ObjectId("62a2ee697905692032047329"),
        "ip_address" : "127.0.0.1",
        "time_stamp" : "[15/Oct/2011:11:49:11-0400]",
        "request_type" : "GET",
        "url" : "/",
        "protocol" : "HTTP/1.1",
        "response" : 200,
        "response_time" : 44
}
```

Clone the repository

```
cd ~
git clone git@github.com:tambeani/INFO7250---Engineering-of-Big-Data-Systems.git
```

Below is the maven code for:

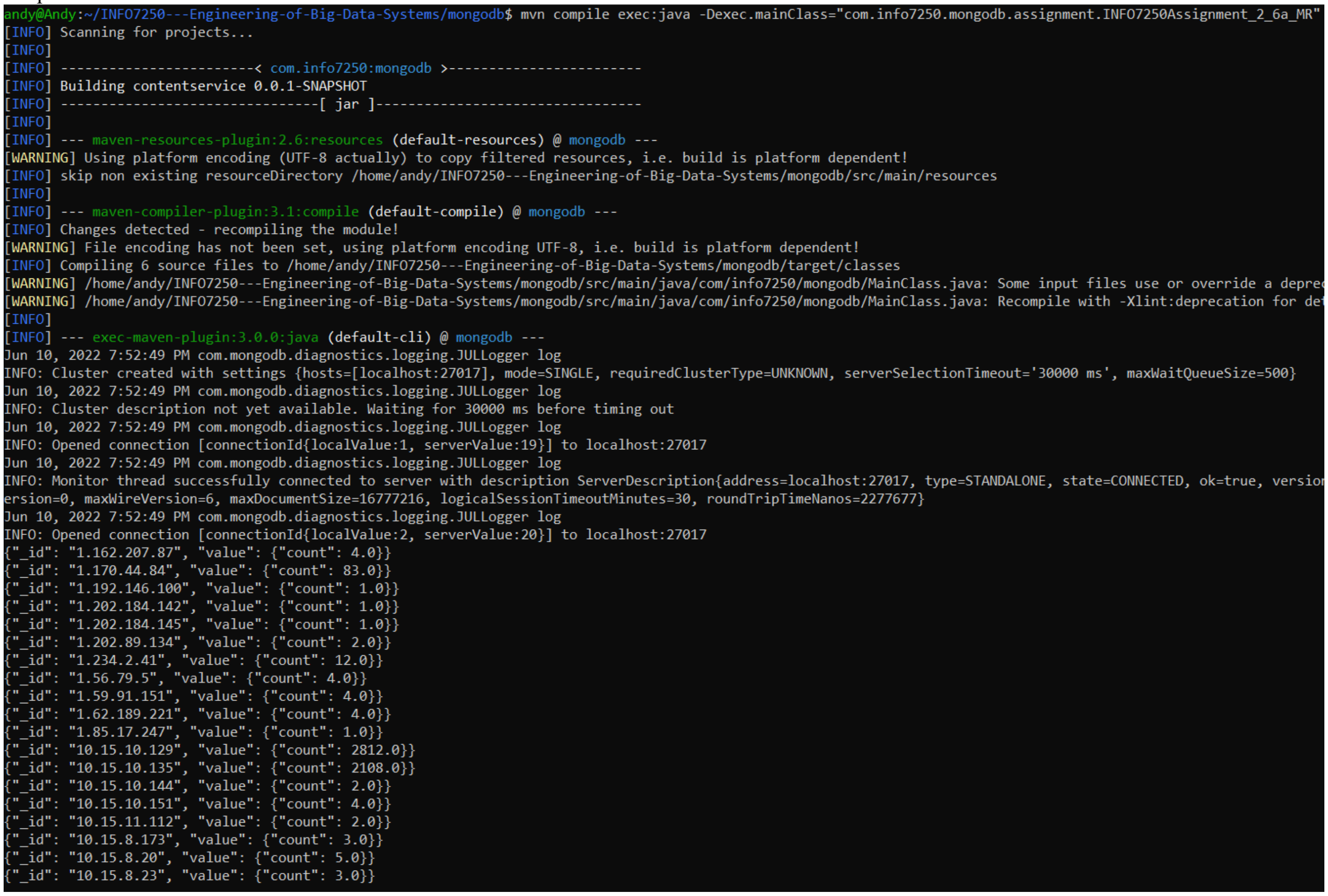
Q.Number of times any webpage was visited by the same IP address

```
package com.info7250.mongodb.assignment;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

import org.bson.Document;

import com.mongodb.Block;
import com.mongodb.MapReduceCommand;
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 com.mongodb.client.model.Accumulators;
import com.mongodb.client.model.Aggregates;
import com.mongodb.client.model.Sorts;

public class INFO7250Assignment_2_6a_MR implements Block<Document> {

        public static void main(String[] args) throws FileNotFoundException {
                // TODO Auto-generated method stub

                // Establish connection using modern client
                                MongoClient client = MongoClients.create();

                                // Connect to mongodb
                                MongoDatabase assignment_2 = client.getDatabase("logs");

                                // Create/get collections
                                MongoCollection<Document>  coll = assignment_2.getCollection("access");

                                // Define printBlock for each iterable
                                Block<Document> printBlock = new INFO7250Assignment_2_6a_MR();

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

                                // Define the map function
                                String map = "function(){"
                                                        + "emit(this.ip_address," + "{\"count\":1});"
                                                + "}";

                                // Define the reduce function
                                String reduce = "function(key,values){"
                                                + "var result = {\"count\":1};"
                                                + "values.forEach("
                                                        + "function(value){"
                                                                +"result.count += value.count;"
                                                        + "});"
                                                        +" return result;"
                                                + "}";

                                // Execute map reduce
                                MapReduceIterable<Document> result = coll.mapReduce(map,reduce);

                                // Print the map reduce result
                                for(Document doc:result) {
                                        System.out.println(doc.toJson());
                                }

                                // Close the connection
                                client.close();

        }

        public void apply(Document t) {
                // TODO Auto-generated method stub
                System.out.println(t.toJson());
        }


}
```

To run the code:

```
cd ~
cd INFO7250---Engineering-of-Big-Data-Systems/mongodb/
 mvn compile exec:java -Dexec.mainClass="com.info7250.mongodb.assignment.INFO7250Assignment_2_6a_MR"
```

Output:



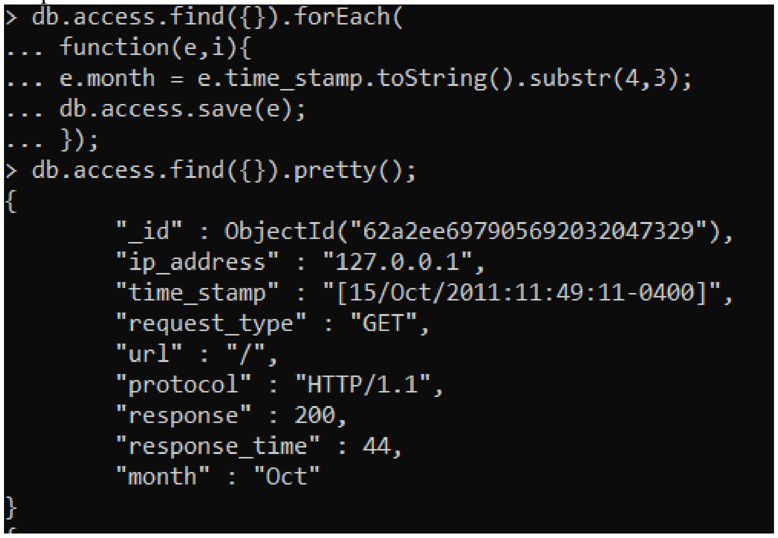
```
andy@Andy:~/INFO7250---Engineering-of-Big-Data-Systems/mongodb$ mvn compile exec:java -Dexec.mainClass="com.info7250.mongodb.assignment.INFO7250Assignment_2_6a_MR"
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------------< com.info7250:mongodb >------------------------
[INFO] Building contentservice 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ mongodb ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/andy/INFO7250---Engineering-of-Big-Data-Systems/mongodb/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ mongodb ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 6 source files to /home/andy/INFO7250---Engineering-of-Big-Data-Systems/mongodb/target/classes
[WARNING] /home/andy/INFO7250---Engineering-of-Big-Data-Systems/mongodb/src/main/java/com/info7250/mongodb/MainClass.java: Some input files use or override a depre
[WARNING] /home/andy/INFO7250---Engineering-of-Big-Data-Systems/mongodb/src/main/java/com/info7250/mongodb/MainClass.java: Recompile with -Xlint:deprecation for de
[INFO]
[INFO] --- exec-maven-plugin:3.0.0:java (default-cli) @ mongodb ---
Jun 10, 2022 7:52:49 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}
Jun 10, 2022 7:52:49 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
Jun 10, 2022 7:52:49 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:19}] to localhost:27017
Jun 10, 2022 7:52:49 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, versio
ersion=0, maxWireVersion=6, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30, roundTripTimeNanos=2277677}
Jun 10, 2022 7:52:49 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:20}] to localhost:27017
{"_id": "1.162.207.87", "value": {"count": 4.0}}
{"_id": "1.170.44.84", "value": {"count": 83.0}}
{"_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}}
{"_id": "10.15.8.20", "value": {"count": 5.0}}
{"_id": "10.15.8.23", "value": {"count": 3.0}}
```

Q.Number of times any webpage was visited each month

Let us process the `time_stamp` column to extract month data,

```
db.access.find({}).forEach(
        function(e,i){
                e.month = e.time_stamp.toString().substr(4,3);
                db.access.save(e);
});
```

Output:



```
> db.access.find({}).forEach(
... function(e,i){
... e.month = e.time_stamp.toString().substr(4,3);
... db.access.save(e);
... });
> db.access.find({}).pretty();
{
        "_id" : ObjectId("62a2ee697905692032047329"),
        "ip_address" : "127.0.0.1",
        "time_stamp" : "[15/Oct/2011:11:49:11-0400]",
        "request_type" : "GET",
        "url" : "/",
        "protocol" : "HTTP/1.1",
        "response" : 200,
        "response_time" : 44,
        "month" : "Oct"
}
```

Below is the maven code:

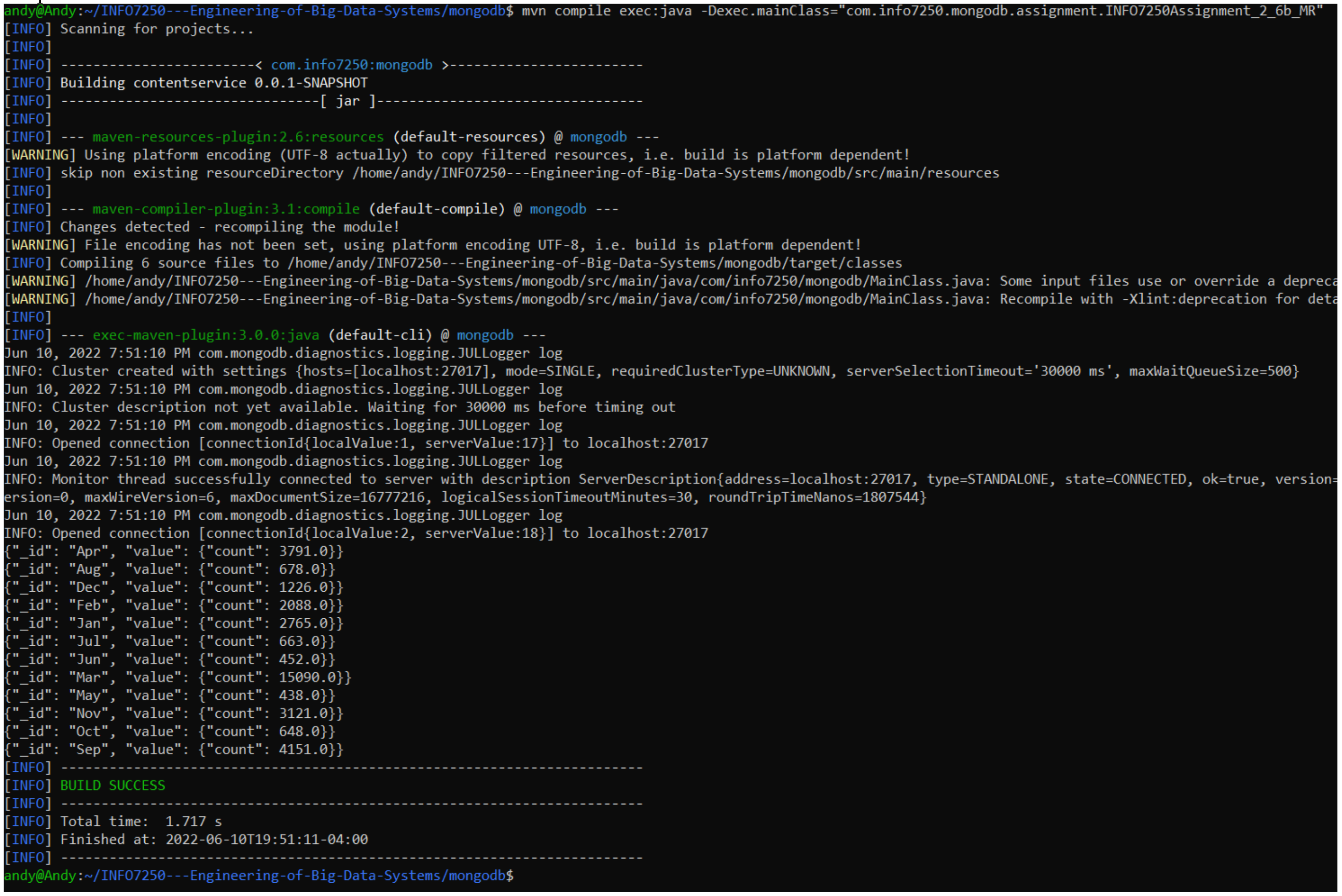
Q.Number of times any webpage was visited each month

```
package com.info7250.mongodb.assignment;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

import org.bson.Document;

import com.mongodb.Block;
import com.mongodb.MapReduceCommand;
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 com.mongodb.client.model.Accumulators;
import com.mongodb.client.model.Aggregates;
import com.mongodb.client.model.Sorts;

public class INFO7250Assignment_2_6b_MR implements Block<Document> {

        public static void main(String[] args) throws FileNotFoundException {
                // TODO Auto-generated method stub

                // Establish connection using modern client
                                MongoClient client = MongoClients.create();

                                // Connect to mongodb
                                MongoDatabase assignment_2 = client.getDatabase("logs");

                                // Create/get collections
                                MongoCollection<Document>  coll = assignment_2.getCollection("access");

                                // Define printBlock for each iterable
                                Block<Document> printBlock = new INFO7250Assignment_2_6b_MR();

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

                                // Define the map function
                                String map = "function(){"
                                                + "emit(this.month," + "{\"count\":1});"
                                        + "}";

                                // Define the reduce function
                                String reduce = "function(key,values){"
                                        + "var result = {\"count\":1};"
                                        + "values.forEach("
                                                + "function(value){"
                                                        +"result.count += value.count;"
                                                + "});"
                                                +" return result;"
                                        + "}";

                                // Execute map reduce
                                MapReduceIterable<Document> result = coll.mapReduce(map,reduce);

                                // Print the map reduce result
                                for(Document doc:result) {
                                        System.out.println(doc.toJson());
                                }

                                // Close the connection
                                client.close();

        }

        public void apply(Document t) {
                // TODO Auto-generated method stub
                System.out.println(t.toJson());
        }


}
```

To run the code:

```
cd ~
cd INFO7250---Engineering-of-Big-Data-Systems/mongodb/
mvn compile exec:java -Dexec.mainClass="com.info7250.mongodb.assignment.INFO7250Assignment_2_6b_MR"
```

Output:



```
andy@Andy:~/INFO7250---Engineering-of-Big-Data-Systems/mongodb$ mvn compile exec:java -Dexec.mainClass="com.info7250.mongodb.assignment.INFO7250Assignment_2_6b_MR"
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------------< com.info7250:mongodb >------------------------
[INFO] Building contentservice 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ mongodb ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/andy/INFO7250---Engineering-of-Big-Data-Systems/mongodb/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ mongodb ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 6 source files to /home/andy/INFO7250---Engineering-of-Big-Data-Systems/mongodb/target/classes
[WARNING] /home/andy/INFO7250---Engineering-of-Big-Data-Systems/mongodb/src/main/java/com/info7250/mongodb/MainClass.java: Some input files use or override a depreca
[WARNING] /home/andy/INFO7250---Engineering-of-Big-Data-Systems/mongodb/src/main/java/com/info7250/mongodb/MainClass.java: Recompile with -Xlint:deprecation for deta
[INFO]
[INFO] --- exec-maven-plugin:3.0.0:java (default-cli) @ mongodb ---
Jun 10, 2022 7:51:10 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}
Jun 10, 2022 7:51:10 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
Jun 10, 2022 7:51:10 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:17}] to localhost:27017
Jun 10, 2022 7:51:10 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, version=
ersion=0, maxWireVersion=6, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30, roundTripTimeNanos=1807544}
Jun 10, 2022 7:51:10 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:18}] to localhost:27017
{"_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}}
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.717 s
[INFO] Finished at: 2022-06-10T19:51:11-04:00
[INFO] ------------------------------------------------------------------------
andy@Andy:~/INFO7250---Engineering-of-Big-Data-Systems/mongodb$
```

PART 7 - PROGRAMMING ASSIGNMENT

Redo Part-6 using Aggregation Pipeline.

Below is the maven code for:

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

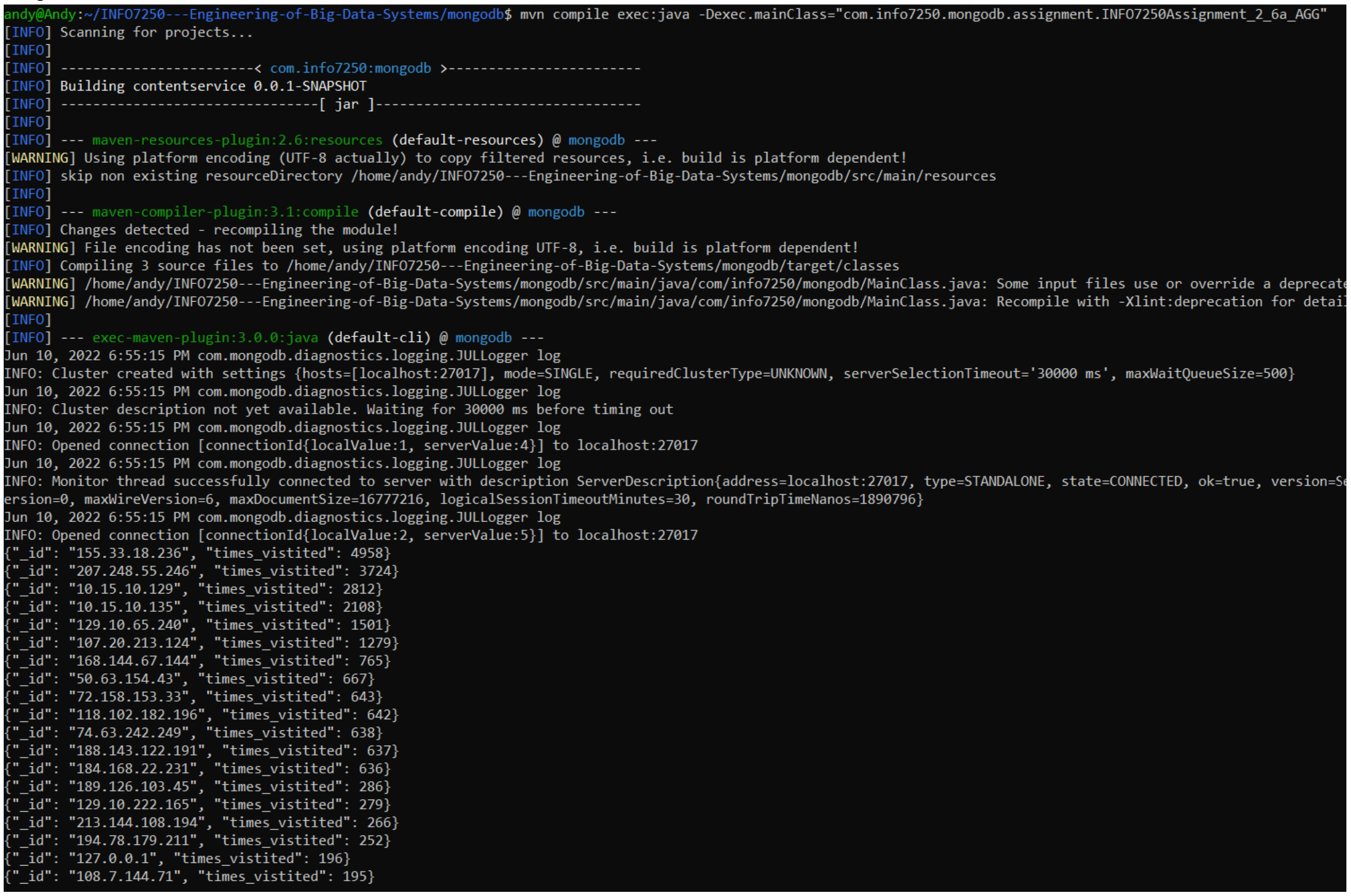
```
package com.info7250.mongodb.assignment;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

import org.bson.Document;

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;

public class INFO7250Assignment_2_6a_AGG implements Block<Document> {

        public static void main(String[] args) throws FileNotFoundException {
                // TODO Auto-generated method stub

                // Establish connection using modern client
                                MongoClient client = MongoClients.create();

                                // Connect to mongodb
                                MongoDatabase assignment_2 = client.getDatabase("logs");
```

```
                                        // Create/get collections
                                        MongoCollection<Document>  coll = assignment_2.getCollection("access");

                                        // Define printBlock for each iterable
                                        Block<Document> printBlock = new INFO7250Assignment_2_6a_AGG();

                                        // Define a pipeline for aggregation
                                        //List<Document> aggregated =
                                        coll.aggregate(
                                                        Arrays.asList(
                                                                        Aggregates.group("$ip_address",Accumulators.sum("times_vistited", 1)),
                                                                        Aggregates.sort(Sorts.descending("times_vistited"))
                                                                        )
                                                        ).forEach(printBlock);
                                                        //.into(new ArrayList<>());

                                        //lab_2.getCollection("stock_avg_collection").insertMany(aggregated);

                                        // Close the connection
                                        client.close();

        }

        public void apply(Document t) {
                // TODO Auto-generated method stub
                System.out.println(t.toJson());
        }


}
```

To run the code:

```
cd ~
cd INFO7250---Engineering-of-Big-Data-Systems/mongodb/
mvn compile exec:java -Dexec.mainClass="com.info7250.mongodb.assignment.INFO7250Assignment_2_6a_AGG"
```

Output:



```
andy@Andy:~/INFO7250---Engineering-of-Big-Data-Systems/mongodb$ mvn compile exec:java -Dexec.mainClass="com.info7250.mongodb.assignment.INFO7250Assignment_2_6a_AGG"
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------------< com.info7250:mongodb >-----------------------
[INFO] Building contentservice 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ mongodb ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/andy/INFO7250---Engineering-of-Big-Data-Systems/mongodb/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ mongodb ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 3 source files to /home/andy/INFO7250---Engineering-of-Big-Data-Systems/mongodb/target/classes
[WARNING] /home/andy/INFO7250---Engineering-of-Big-Data-Systems/mongodb/src/main/java/com/info7250/mongodb/MainClass.java: Some input files use or override a deprecate
[WARNING] /home/andy/INFO7250---Engineering-of-Big-Data-Systems/mongodb/src/main/java/com/info7250/mongodb/MainClass.java: Recompile with -Xlint:deprecation for detai
[INFO]
[INFO] --- exec-maven-plugin:3.0.0:java (default-cli) @ mongodb ---
Jun 10, 2022 6:55:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}
Jun 10, 2022 6:55:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
Jun 10, 2022 6:55:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:4}] to localhost:27017
Jun 10, 2022 6:55:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, version=Se
ersion=0, maxWireVersion=6, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30, roundTripTimeNanos=1890796}
Jun 10, 2022 6:55:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:5}] to localhost:27017
{"_id": "155.33.18.236", "times_vistited": 4958}
{"_id": "207.248.55.246", "times_vistited": 3724}
{"_id": "10.15.10.129", "times_vistited": 2812}
{"_id": "10.15.10.135", "times_vistited": 2108}
{"_id": "129.10.65.240", "times_vistited": 1501}
{"_id": "107.20.213.124", "times_vistited": 1279}
{"_id": "168.144.67.144", "times_vistited": 765}
{"_id": "50.63.154.43", "times_vistited": 667}
{"_id": "72.158.153.33", "times_vistited": 643}
{"_id": "118.102.182.196", "times_vistited": 642}
{"_id": "74.63.242.249", "times_vistited": 638}
{"_id": "188.143.122.191", "times_vistited": 637}
{"_id": "184.168.22.231", "times_vistited": 636}
{"_id": "189.126.103.45", "times_vistited": 286}
{"_id": "129.10.222.165", "times_vistited": 279}
{"_id": "213.144.108.194", "times_vistited": 266}
{"_id": "194.78.179.211", "times_vistited": 252}
{"_id": "127.0.0.1", "times_vistited": 196}
{"_id": "108.7.144.71", "times_vistited": 195}
```

Below is the maven code:

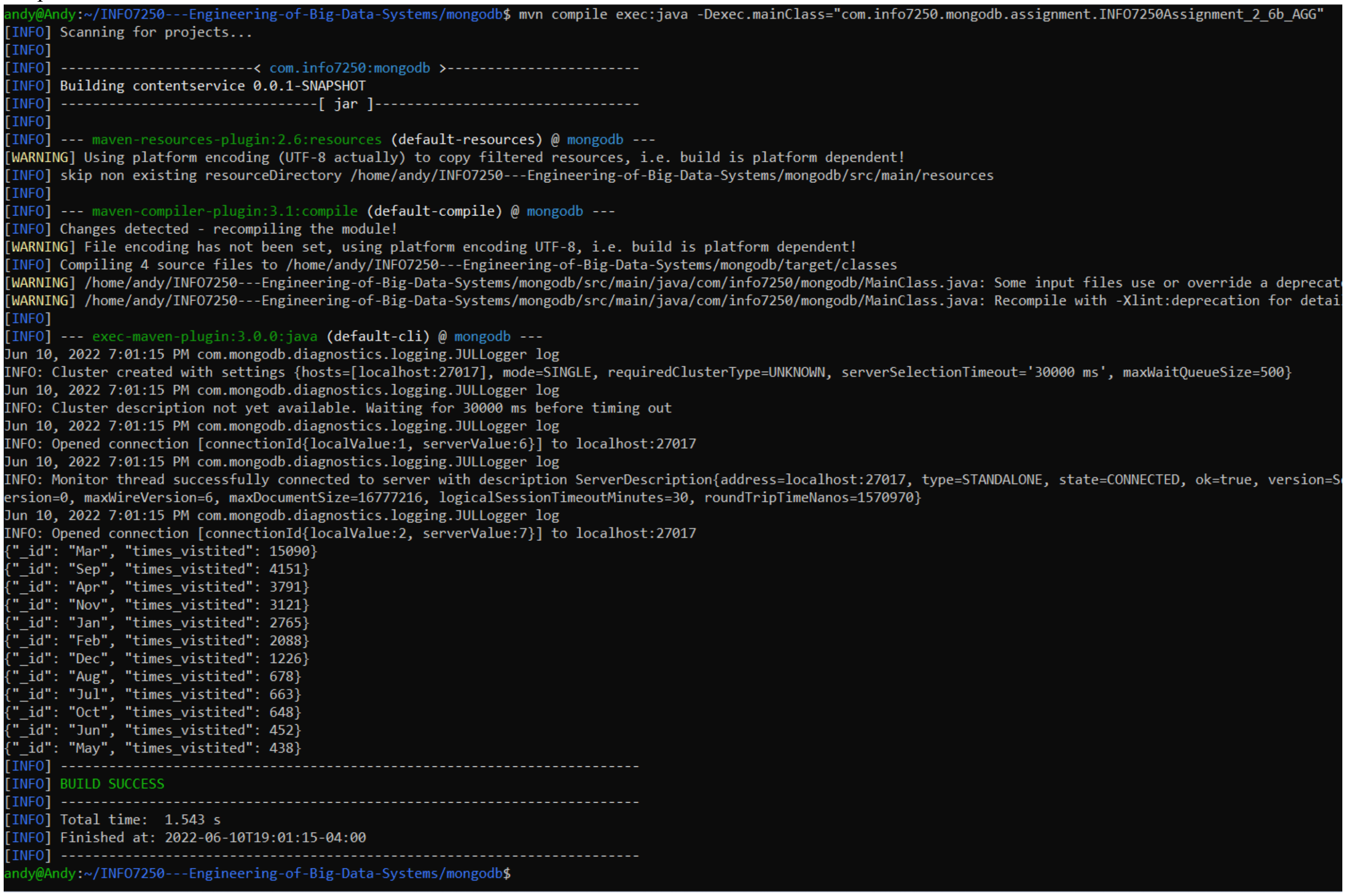
Q.Number of times any webpage was visited each month

```
package com.info7250.mongodb.assignment;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

import org.bson.Document;

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;

public class INFO7250Assignment_2_6b_AGG implements Block<Document> {

        public static void main(String[] args) throws FileNotFoundException {
                // TODO Auto-generated method stub

                // Establish connection using modern client
                                MongoClient client = MongoClients.create();

                                // Connect to mongodb
                                MongoDatabase assignment_2 = client.getDatabase("logs");

                                // Create/get collections
                                MongoCollection<Document>  coll = assignment_2.getCollection("access");

                                // Define printBlock for each iterable
                                Block<Document> printBlock = new INFO7250Assignment_2_6b_AGG();

                                // Define a pipeline for aggregation
                                //List<Document> aggregated =
                                coll.aggregate(
                                                Arrays.asList(
                                                                Aggregates.group("$month",Accumulators.sum("times_vistited", 1)),
                                                                Aggregates.sort(Sorts.descending("times_vistited"))
                                                                )
                                                ).forEach(printBlock);
                                                //.into(new ArrayList<>());

                                //lab_2.getCollection("stock_avg_collection").insertMany(aggregated);

                                // Close the connection
                                client.close();

        }

        public void apply(Document t) {
                // TODO Auto-generated method stub
                System.out.println(t.toJson());
        }


}
```

To run the code:

```
cd ~
cd INFO7250---Engineering-of-Big-Data-Systems/mongodb/
mvn compile exec:java -Dexec.mainClass="com.info7250.mongodb.assignment.INFO7250Assignment_2_6b_AGG"
```

Output:



```
andy@Andy:~/INFO7250---Engineering-of-Big-Data-Systems/mongodb$ mvn compile exec:java -Dexec.mainClass="com.info7250.mongodb.assignment.INFO7250Assignment_2_6b_AGG"
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------------< com.info7250:mongodb >------------------------
[INFO] Building contentservice 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ mongodb ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/andy/INFO7250---Engineering-of-Big-Data-Systems/mongodb/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ mongodb ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 4 source files to /home/andy/INFO7250---Engineering-of-Big-Data-Systems/mongodb/target/classes
[WARNING] /home/andy/INFO7250---Engineering-of-Big-Data-Systems/mongodb/src/main/java/com/info7250/mongodb/MainClass.java: Some input files use or override a deprecat
[WARNING] /home/andy/INFO7250---Engineering-of-Big-Data-Systems/mongodb/src/main/java/com/info7250/mongodb/MainClass.java: Recompile with -Xlint:deprecation for detai
[INFO]
[INFO] --- exec-maven-plugin:3.0.0:java (default-cli) @ mongodb ---
Jun 10, 2022 7:01:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}
Jun 10, 2022 7:01:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
Jun 10, 2022 7:01:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:6}] to localhost:27017
Jun 10, 2022 7:01:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, version=S
ersion=0, maxWireVersion=6, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30, roundTripTimeNanos=1570970}
Jun 10, 2022 7:01:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:7}] to localhost:27017
{"_id": "Mar", "times_vistited": 15090}
{"_id": "Sep", "times_vistited": 4151}
{"_id": "Apr", "times_vistited": 3791}
{"_id": "Nov", "times_vistited": 3121}
{"_id": "Jan", "times_vistited": 2765}
{"_id": "Feb", "times_vistited": 2088}
{"_id": "Dec", "times_vistited": 1226}
{"_id": "Aug", "times_vistited": 678}
{"_id": "Jul", "times_vistited": 663}
{"_id": "Oct", "times_vistited": 648}
{"_id": "Jun", "times_vistited": 452}
{"_id": "May", "times_vistited": 438}
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.543 s
[INFO] Finished at: 2022-06-10T19:01:15-04:00
[INFO] ------------------------------------------------------------------------
andy@Andy:~/INFO7250---Engineering-of-Big-Data-Systems/mongodb$
```