PART 1 - READING ASSIGNMENT https://learning.oreilly.com/library/view/mongodb-in-action/9781617291609/?ar (Links to an external site.)

Chapter 4. Document-oriented data

Chapter 5. Constructing queries

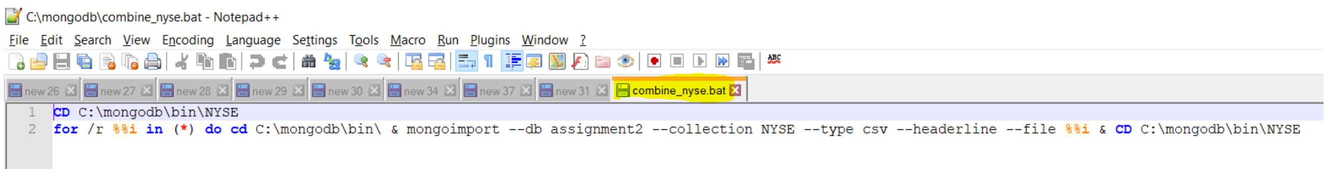
Chapter 6. Aggregation

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

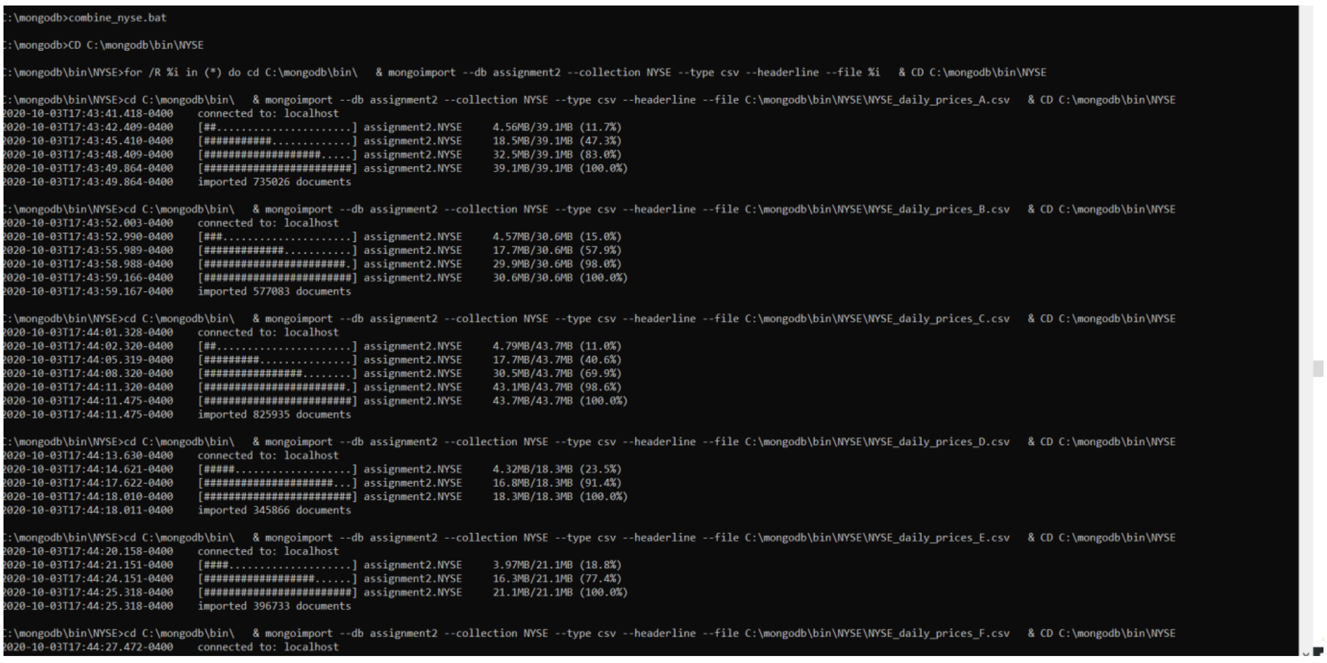
PART 2 - PROGRAMMING ASSIGNMENT

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

NYSE Dataset Link: http://msis.neu.edu/nyse/


C:\mongodb\combine_nyse.bat - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
new 26
new 27
new 28
new 29
new 30
new 34
new 37
new 31
combine_nyse.bat

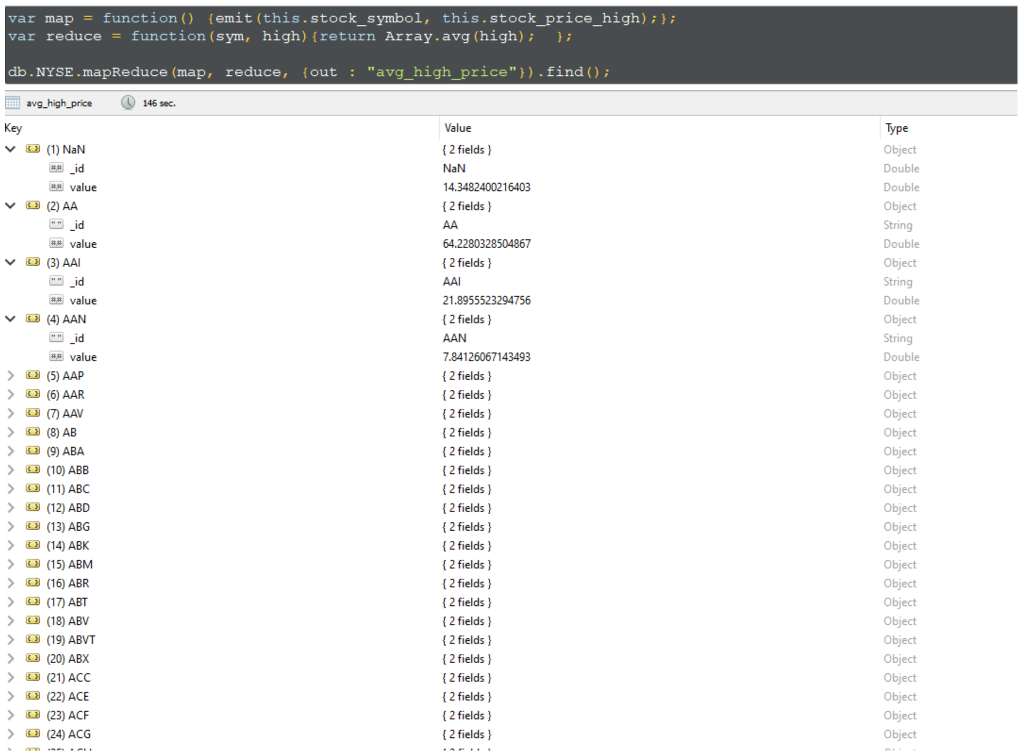
```
CD C:\mongodb\bin\NYSE
for /r %%i in (*) do cd C:\mongodb\bin\ & mongoimport --db assignment2 --collection NYSE --type csv --headerline --file %%i & CD C:\mongodb\bin\NYSE
```



```
:\mongodb>combine_nyse.bat

:\mongodb>CD C:\mongodb\bin\NYSE

:\mongodb\bin\NYSE>for /R %i in (*) do cd C:\mongodb\bin\   & mongoimport --db assignment2 --collection NYSE --type csv --headerline --file %i   & CD C:\mongodb\bin\NYSE

:\mongodb\bin\NYSE>cd C:\mongodb\bin\   & mongoimport --db assignment2 --collection NYSE --type csv --headerline --file C:\mongodb\bin\NYSE\NYSE_daily_prices_A.csv   & CD C:\mongodb\bin\NYSE
2020-10-03T17:43:41.418-0400    connected to: localhost
2020-10-03T17:43:42.409-0400    [##......................] assignment2.NYSE     4.56MB/39.1MB (11.7%)
2020-10-03T17:43:45.410-0400    [###########.............] assignment2.NYSE     18.5MB/39.1MB (47.3%)
2020-10-03T17:43:48.409-0400    [###################.....] assignment2.NYSE     32.5MB/39.1MB (83.0%)
2020-10-03T17:43:49.864-0400    [########################] assignment2.NYSE     39.1MB/39.1MB (100.0%)
2020-10-03T17:43:49.864-0400    imported 735026 documents

:\mongodb\bin\NYSE>cd C:\mongodb\bin\   & mongoimport --db assignment2 --collection NYSE --type csv --headerline --file C:\mongodb\bin\NYSE\NYSE_daily_prices_B.csv   & CD C:\mongodb\bin\NYSE
2020-10-03T17:43:52.003-0400    connected to: localhost
2020-10-03T17:43:52.990-0400    [###.....................] assignment2.NYSE     4.57MB/30.6MB (15.0%)
2020-10-03T17:43:55.989-0400    [#############...........] assignment2.NYSE     17.7MB/30.6MB (57.9%)
2020-10-03T17:43:58.988-0400    [#######################.] assignment2.NYSE     29.9MB/30.6MB (98.0%)
2020-10-03T17:43:59.166-0400    [########################] assignment2.NYSE     30.6MB/30.6MB (100.0%)
2020-10-03T17:43:59.167-0400    imported 577083 documents

:\mongodb\bin\NYSE>cd C:\mongodb\bin\   & mongoimport --db assignment2 --collection NYSE --type csv --headerline --file C:\mongodb\bin\NYSE\NYSE_daily_prices_C.csv   & CD C:\mongodb\bin\NYSE
2020-10-03T17:44:01.328-0400    connected to: localhost
2020-10-03T17:44:02.320-0400    [##......................] assignment2.NYSE     4.79MB/43.7MB (11.0%)
2020-10-03T17:44:05.319-0400    [#########...............] assignment2.NYSE     17.7MB/43.7MB (40.6%)
2020-10-03T17:44:08.320-0400    [################........] assignment2.NYSE     30.5MB/43.7MB (69.9%)
2020-10-03T17:44:11.320-0400    [#######################.] assignment2.NYSE     43.1MB/43.7MB (98.6%)
2020-10-03T17:44:11.475-0400    [########################] assignment2.NYSE     43.7MB/43.7MB (100.0%)
2020-10-03T17:44:11.475-0400    imported 825935 documents

:\mongodb\bin\NYSE>cd C:\mongodb\bin\   & mongoimport --db assignment2 --collection NYSE --type csv --headerline --file C:\mongodb\bin\NYSE\NYSE_daily_prices_D.csv   & CD C:\mongodb\bin\NYSE
2020-10-03T17:44:13.630-0400    connected to: localhost
2020-10-03T17:44:14.621-0400    [#####...................] assignment2.NYSE     4.32MB/18.3MB (23.5%)
2020-10-03T17:44:17.622-0400    [#####################...] assignment2.NYSE     16.8MB/18.3MB (91.4%)
2020-10-03T17:44:18.010-0400    [########################] assignment2.NYSE     18.3MB/18.3MB (100.0%)
2020-10-03T17:44:18.011-0400    imported 345866 documents

:\mongodb\bin\NYSE>cd C:\mongodb\bin\   & mongoimport --db assignment2 --collection NYSE --type csv --headerline --file C:\mongodb\bin\NYSE\NYSE_daily_prices_E.csv   & CD C:\mongodb\bin\NYSE
2020-10-03T17:44:20.158-0400    connected to: localhost
2020-10-03T17:44:21.151-0400    [####....................] assignment2.NYSE     3.97MB/21.1MB (18.8%)
2020-10-03T17:44:24.151-0400    [##################......] assignment2.NYSE     16.3MB/21.1MB (77.4%)
2020-10-03T17:44:25.318-0400    [########################] assignment2.NYSE     21.1MB/21.1MB (100.0%)
2020-10-03T17:44:25.318-0400    imported 396733 documents

:\mongodb\bin\NYSE>cd C:\mongodb\bin\   & mongoimport --db assignment2 --collection NYSE --type csv --headerline --file C:\mongodb\bin\NYSE\NYSE_daily_prices_F.csv   & CD C:\mongodb\bin\NYSE
2020-10-03T17:44:27.472-0400    connected to: localhost
```

PART 3 - PROGRAMMING ASSIGNMENT

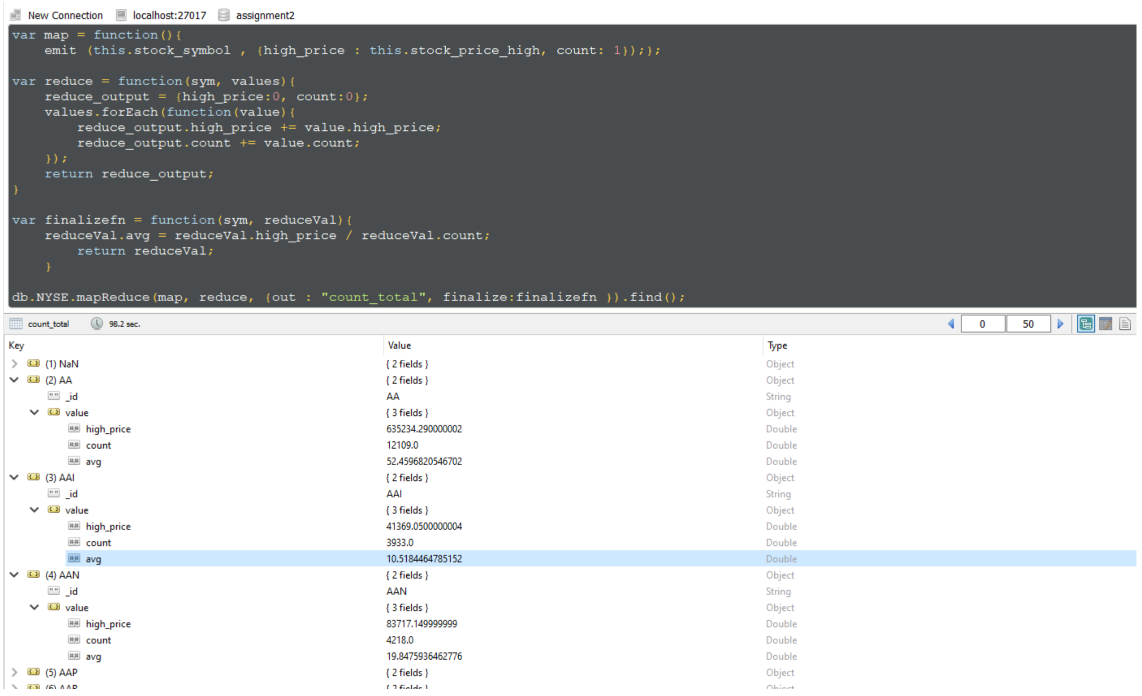
PART 3.1. Use the NYSE database to find the average price of stock_price_high values for each stock using MapReduce.



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

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

```
avg_high_price    146 sec.

Key                 Value                 Type
(1) NaN             { 2 fields }          Object
    _id             NaN                   Double
    value           14.3482400216403      Double
(2) AA              { 2 fields }          Object
    _id             AA                    String
    value           64.2280328504867      Double
(3) AAI             { 2 fields }          Object
    _id             AAI                   String
    value           21.8955523294756      Double
(4) AAN             { 2 fields }          Object
    _id             AAN                   String
    value           7.84126067143493      Double
(5) AAP             { 2 fields }          Object
(6) AAR             { 2 fields }          Object
(7) AAV             { 2 fields }          Object
(8) AB              { 2 fields }          Object
(9) ABA             { 2 fields }          Object
(10) ABB            { 2 fields }          Object
(11) ABC            { 2 fields }          Object
(12) ABD            { 2 fields }          Object
(13) ABG            { 2 fields }          Object
(14) ABK            { 2 fields }          Object
(15) ABM            { 2 fields }          Object
(16) ABR            { 2 fields }          Object
(17) ABT            { 2 fields }          Object
(18) ABV            { 2 fields }          Object
(19) ABVT           { 2 fields }          Object
(20) ABX            { 2 fields }          Object
(21) ACC            { 2 fields }          Object
(22) ACE            { 2 fields }          Object
(23) ACF            { 2 fields }          Object
(24) ACG            { 2 fields }          Object
```

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



New Connection   localhost:27017   assignment2

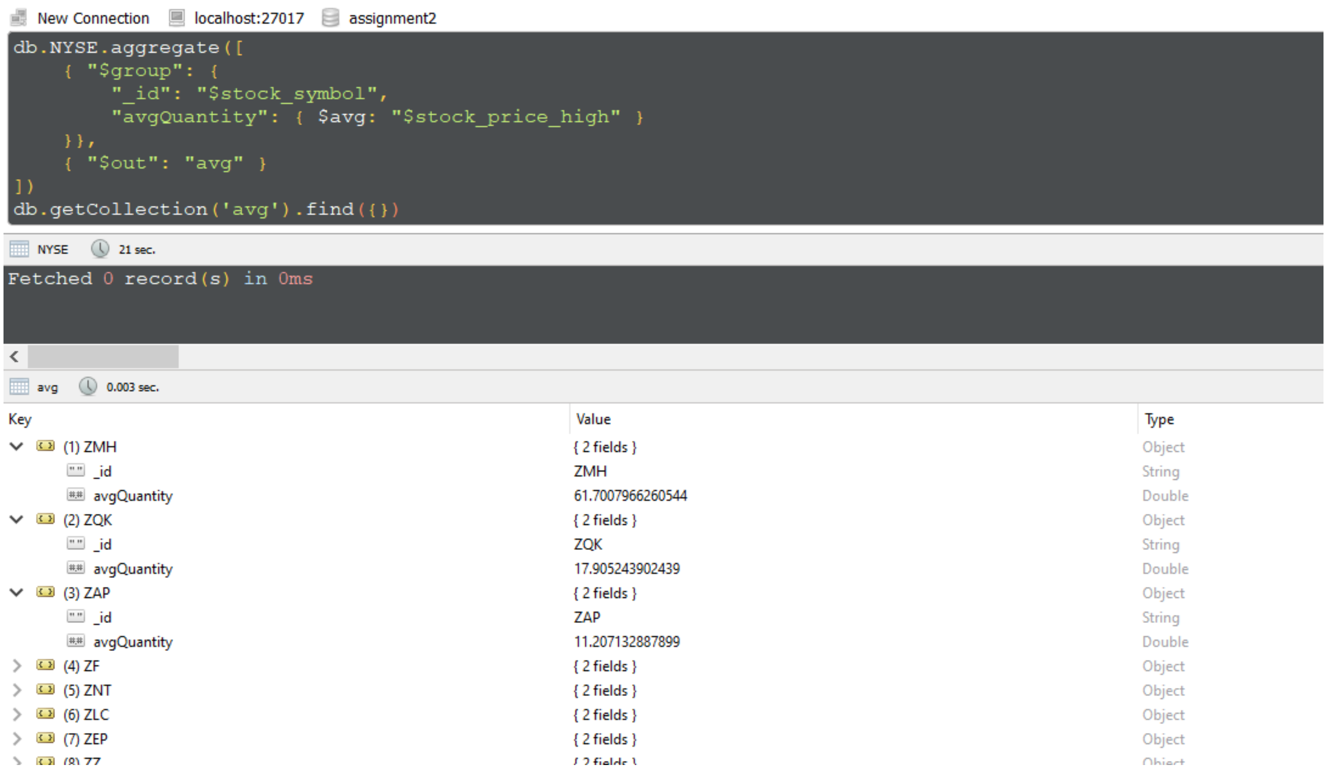
```
var map = function(){
    emit (this.stock_symbol , {high_price : this.stock_price_high, count: 1});};

var reduce = function(sym, values){
    reduce_output = {high_price:0, count:0};
    values.forEach(function(value){
        reduce_output.high_price += value.high_price;
        reduce_output.count += value.count;
    });
    return reduce_output;
}

var finalizefn = function(sym, reduceVal){
    reduceVal.avg = reduceVal.high_price / reduceVal.count;
        return reduceVal;
    }

db.NYSE.mapReduce(map, reduce, {out : "count_total", finalize:finalizefn }).find();
```

```
count_total    98.2 sec.                                   0    50

Key                   Value                  Type
(1) NaN               { 2 fields }           Object
(2) AA                { 2 fields }           Object
    _id               AA                     String
    value             { 3 fields }           Object
        high_price    635234.290000002       Double
        count         12109.0                Double
        avg           52.4596820546702       Double
(3) AAI               { 2 fields }           Object
    _id               AAI                    String
    value             { 3 fields }           Object
        high_price    41369.0500000004       Double
        count         3933.0                 Double
        avg           10.5184464785152       Double
(4) AAN               { 2 fields }           Object
    _id               AAN                    String
    value             { 3 fields }           Object
        high_price    83717.149999999        Double
        count         4218.0                 Double
        avg           19.8475936462776       Double
(5) AAP               { 2 fields }           Object
```

PART 4 - PROGRAMMING ASSIGNMENT

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

https://docs.mongodb.com/manual/reference/operator/aggregation/avg/ (Links to an external site.)



```
db.NYSE.aggregate([
    { "$group": {
        "_id": "$stock_symbol",
        "avgQuantity": { $avg: "$stock_price_high" }
    }},
    { "$out": "avg" }
])
db.getCollection('avg').find({})
```

```
Fetched 0 record(s) in 0ms
```

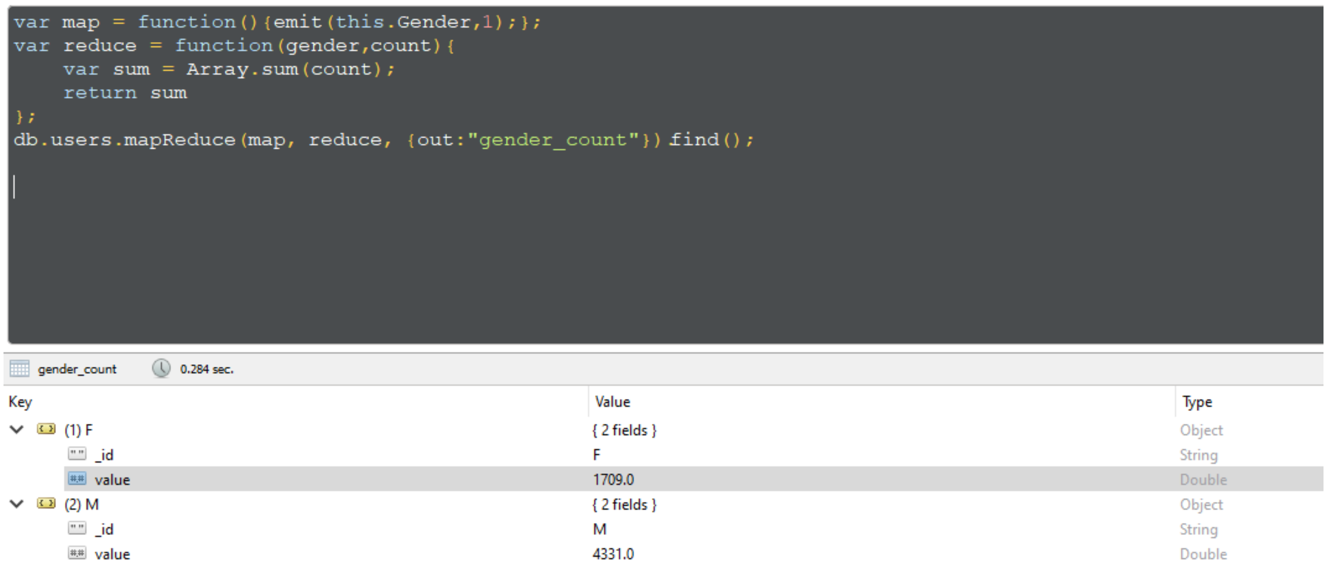
```
Key                  Value                Type
(1) ZMH              { 2 fields }         Object
    _id              ZMH                  String
    avgQuantity      61.7007966260544     Double
(2) ZQK              { 2 fields }         Object
    _id              ZQK                  String
    avgQuantity      17.905243902439      Double
(3) ZAP              { 2 fields }         Object
    _id              ZAP                  String
    avgQuantity      11.207132887899      Double
(4) ZF               { 2 fields }         Object
(5) ZNT              { 2 fields }         Object
(6) ZLC              { 2 fields }         Object
(7) ZEP              { 2 fields }         Object
```

PART 5 - PROGRAMMING ASSIGNMENT

Import the Movielens dataset into MongoDB. Refer to README about file contents and headings.

https://grouplens.org/datasets/movielens/1m/ (Links to an external site.) [you may replace :: in the dateset with comma or tab to import]

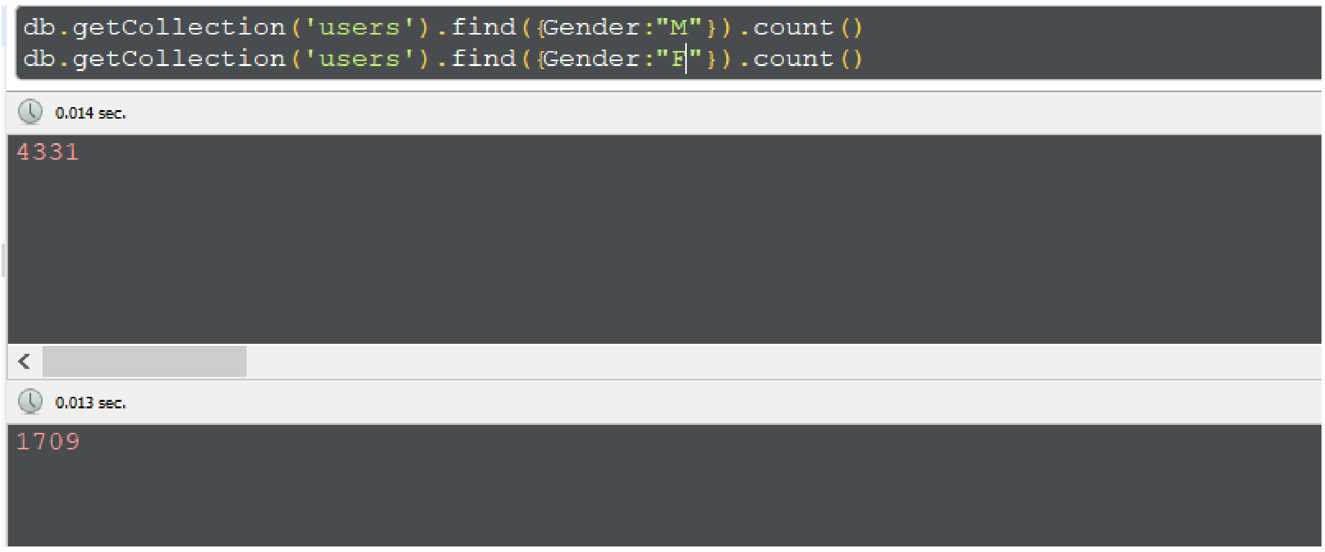
- Find the number Females and Males from the users collection using MapReduce. Do the same thing using count() to compare the results.



```
var map = function(){emit(this.Gender,1);};
var reduce = function(gender,count){
    var sum = Array.sum(count);
    return sum
};
db.users.mapReduce(map, reduce, {out:"gender_count"}).find();
```

gender_count 0.284 sec.

```
Key                 Value          Type
(1) F               { 2 fields }   Object
    _id             F              String
    value           1709.0         Double
(2) M               { 2 fields }   Object
    _id             M              String
    value           4331.0         Double
```



```
db.getCollection('users').find({Gender:"M"}).count()
db.getCollection('users').find({Gender:"F"}).count()
```

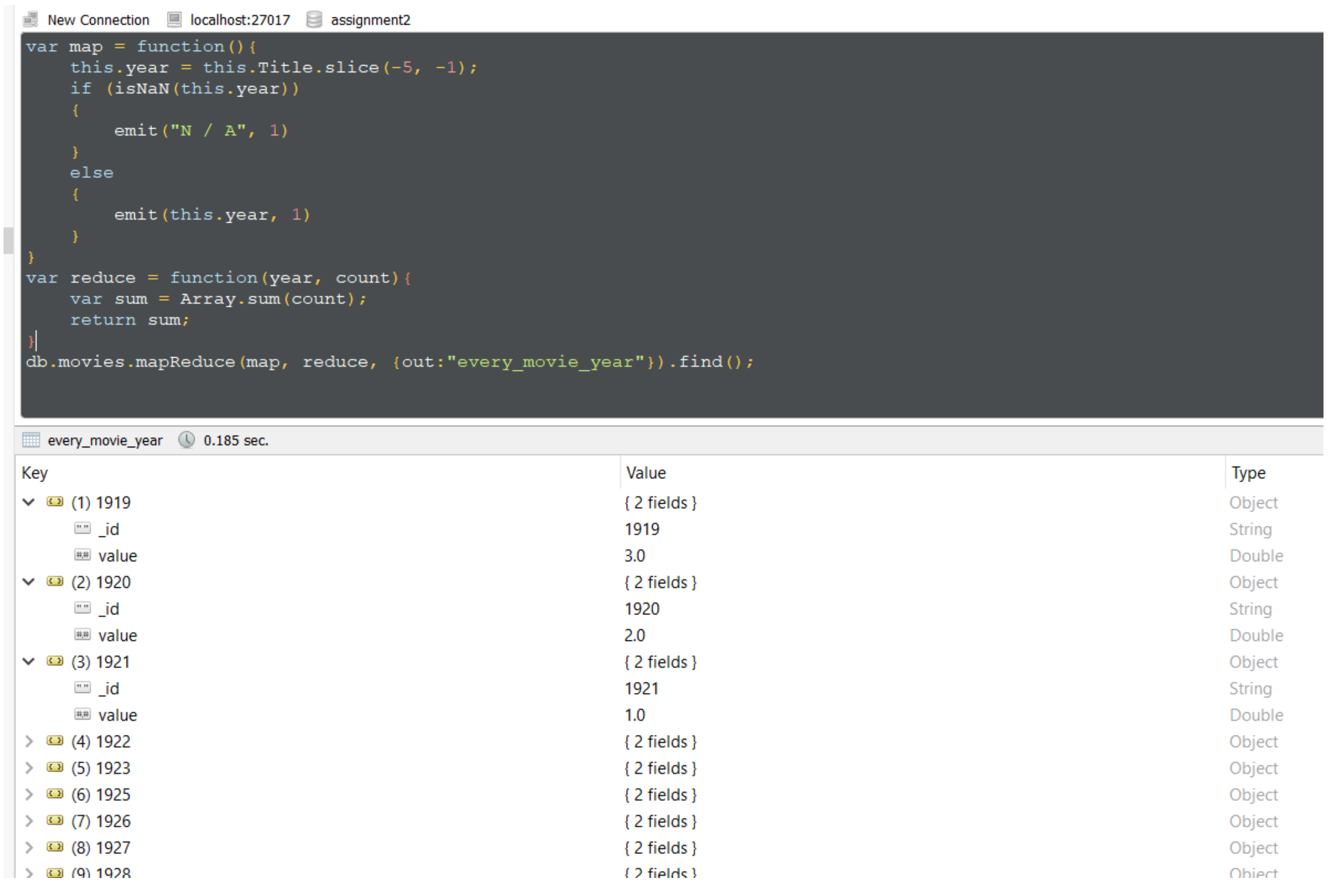
0.014 sec.

```
4331
```

0.013 sec.

```
1709
```

- Find the number of Movies per year using MapReduce



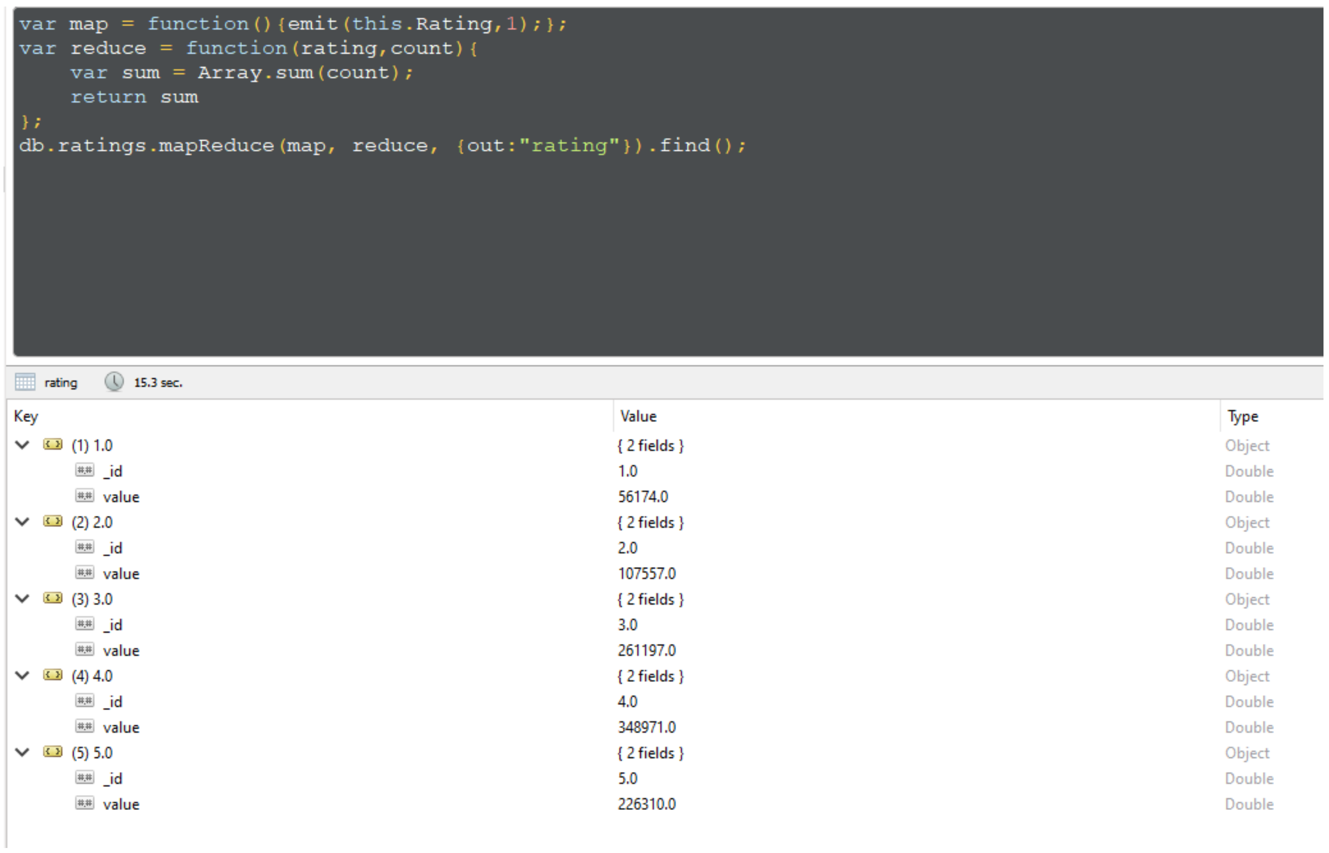
New Connection localhost:27017 assignment2

```
var map = function(){
    this.year = this.Title.slice(-5, -1);
    if (isNaN(this.year))
    {
        emit("N / A", 1)
    }
    else
    {
        emit(this.year, 1)
    }
}
var reduce = function(year, count){
    var sum = Array.sum(count);
    return sum;
}
db.movies.mapReduce(map, reduce, {out:"every_movie_year"}).find();
```

every_movie_year 0.185 sec.

```
Key                 Value          Type
(1) 1919            { 2 fields }   Object
    _id             1919           String
    value           3.0            Double
(2) 1920            { 2 fields }   Object
    _id             1920           String
    value           2.0            Double
(3) 1921            { 2 fields }   Object
    _id             1921           String
    value           1.0            Double
(4) 1922            { 2 fields }   Object
(5) 1923            { 2 fields }   Object
(6) 1925            { 2 fields }   Object
(7) 1926            { 2 fields }   Object
(8) 1927            { 2 fields }   Object
(9) 1928            { 2 fields }   Object
```

- Find the number of Movies per rating using MapReduce



```
var map = function(){emit(this.Rating,1);};
var reduce = function(rating,count){
    var sum = Array.sum(count);
    return sum
};
db.ratings.mapReduce(map, reduce, {out:"rating"}).find();
```

rating 15.3 sec.

```
Key                 Value          Type
(1) 1.0             { 2 fields }   Object
    _id             1.0            Double
    value           56174.0        Double
(2) 2.0             { 2 fields }   Object
    _id             2.0            Double
    value           107557.0       Double
(3) 3.0             { 2 fields }   Object
    _id             3.0            Double
    value           261197.0       Double
(4) 4.0             { 2 fields }   Object
    _id             4.0            Double
    value           348971.0       Double
(5) 5.0             { 2 fields }   Object
    _id             5.0            Double
    value           226310.0       Double
```