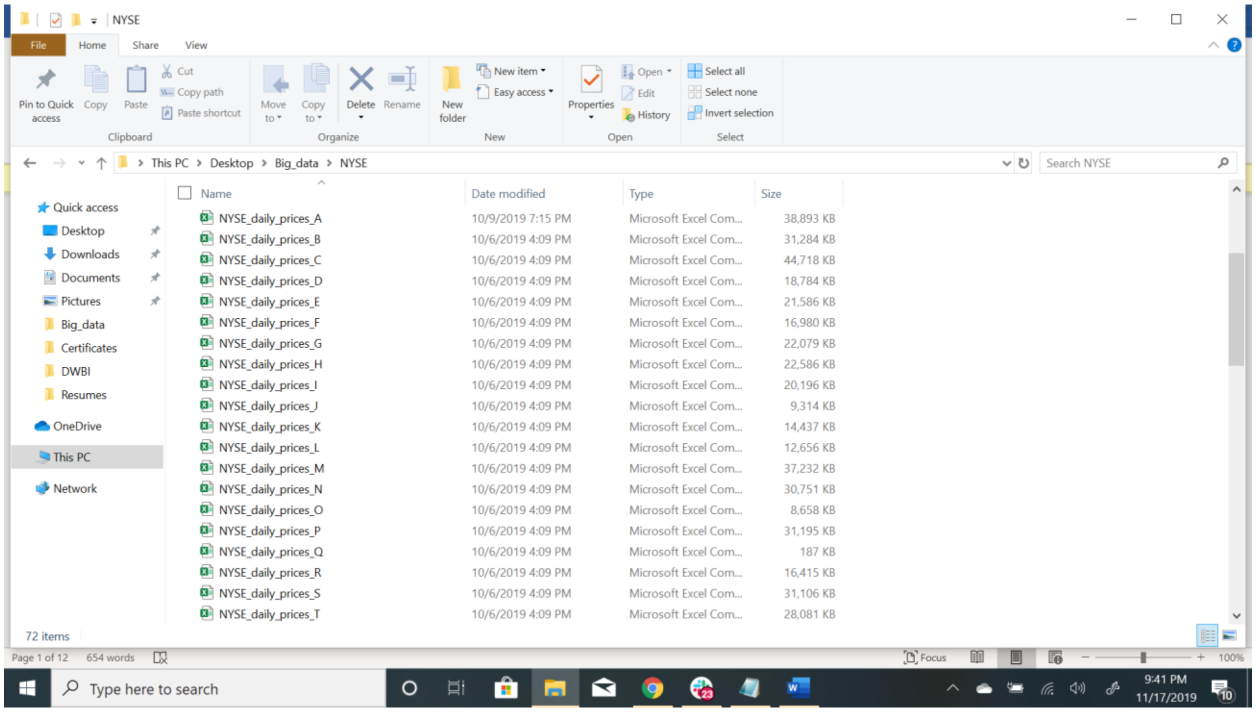
Map reduce on NYSE dataset

This NYSE dataset contains 100M rows. Screenshot of the dataset. It contains daily prices of all the stocks starts from A to Z.



Name	Date modified	Type	Size
NYSE_daily_prices_A	10/9/2019 7:15 PM	Microsoft Excel Com...	38,893 KB
NYSE_daily_prices_B	10/6/2019 4:09 PM	Microsoft Excel Com...	31,284 KB
NYSE_daily_prices_C	10/6/2019 4:09 PM	Microsoft Excel Com...	44,718 KB
NYSE_daily_prices_D	10/6/2019 4:09 PM	Microsoft Excel Com...	18,784 KB
NYSE_daily_prices_E	10/6/2019 4:09 PM	Microsoft Excel Com...	21,586 KB
NYSE_daily_prices_F	10/6/2019 4:09 PM	Microsoft Excel Com...	16,980 KB
NYSE_daily_prices_G	10/6/2019 4:09 PM	Microsoft Excel Com...	22,079 KB
NYSE_daily_prices_H	10/6/2019 4:09 PM	Microsoft Excel Com...	22,586 KB
NYSE_daily_prices_I	10/6/2019 4:09 PM	Microsoft Excel Com...	20,196 KB
NYSE_daily_prices_J	10/6/2019 4:09 PM	Microsoft Excel Com...	9,314 KB
NYSE_daily_prices_K	10/6/2019 4:09 PM	Microsoft Excel Com...	14,437 KB
NYSE_daily_prices_L	10/6/2019 4:09 PM	Microsoft Excel Com...	12,656 KB
NYSE_daily_prices_M	10/6/2019 4:09 PM	Microsoft Excel Com...	37,232 KB
NYSE_daily_prices_N	10/6/2019 4:09 PM	Microsoft Excel Com...	30,751 KB
NYSE_daily_prices_O	10/6/2019 4:09 PM	Microsoft Excel Com...	8,658 KB
NYSE_daily_prices_P	10/6/2019 4:09 PM	Microsoft Excel Com...	31,195 KB
NYSE_daily_prices_Q	10/6/2019 4:09 PM	Microsoft Excel Com...	187 KB
NYSE_daily_prices_R	10/6/2019 4:09 PM	Microsoft Excel Com...	16,415 KB
NYSE_daily_prices_S	10/6/2019 4:09 PM	Microsoft Excel Com...	31,106 KB
NYSE_daily_prices_T	10/6/2019 4:09 PM	Microsoft Excel Com...	28,081 KB

Map reduce on inserted data

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

Map Function:

```
function()
 {
      var value = this.stock_price_high;
      var key = this.stock_symbol;
      emit(key, value);
 }
```

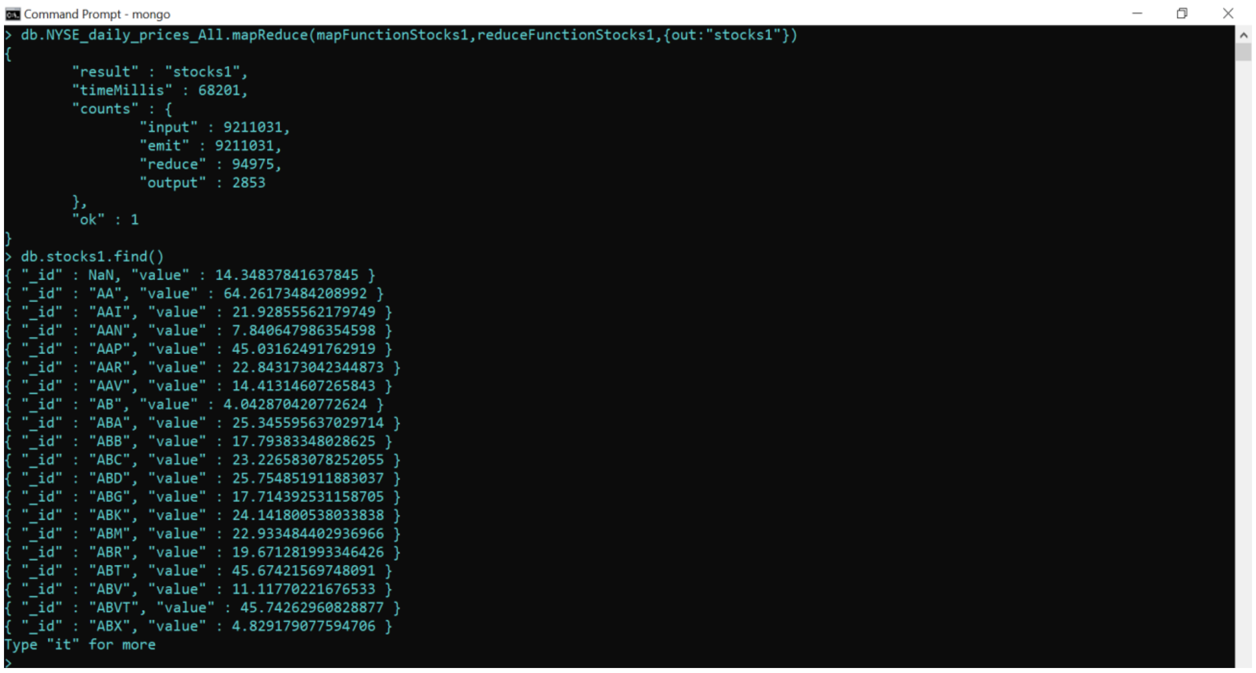
Reduce Function:

```
function(key, values)
 {
     var count= values.length;
     var price=Array.sum(values);
     Average= price/count;
 return Average;
}
```

MapReduce:

```
db.NYSE_daily_prices_All.mapReduce(mapFunctionStocks1,reduceFunctionStocks1,{out:"stocks1"})
```

Output:



```
> db.NYSE_daily_prices_All.mapReduce(mapFunctionStocks1,reduceFunctionStocks1,{out:"stocks1"})
{
        "result" : "stocks1",
        "timeMillis" : 68201,
        "counts" : {
                "input" : 9211031,
                "emit" : 9211031,
                "reduce" : 94975,
                "output" : 2853
        },
        "ok" : 1
}
> db.stocks1.find()
{ "_id" : NaN, "value" : 14.34837841637845 }
{ "_id" : "AA", "value" : 64.26173484208992 }
{ "_id" : "AAI", "value" : 21.92855562179749 }
{ "_id" : "AAN", "value" : 7.840647986354598 }
{ "_id" : "AAP", "value" : 45.03162491762919 }
{ "_id" : "AAR", "value" : 22.843173042344873 }
{ "_id" : "AAV", "value" : 14.41314607265843 }
{ "_id" : "AB", "value" : 4.042870420772624 }
{ "_id" : "ABA", "value" : 25.345595637029714 }
{ "_id" : "ABB", "value" : 17.79383348028625 }
{ "_id" : "ABC", "value" : 23.226583078252055 }
{ "_id" : "ABD", "value" : 25.754851911883037 }
{ "_id" : "ABG", "value" : 17.714392531158705 }
{ "_id" : "ABK", "value" : 24.141800538033838 }
{ "_id" : "ABM", "value" : 22.933484402936966 }
{ "_id" : "ABR", "value" : 19.671281993346426 }
{ "_id" : "ABT", "value" : 45.67421569748091 }
{ "_id" : "ABV", "value" : 11.11770221676533 }
{ "_id" : "ABVT", "value" : 45.74262960828877 }
{ "_id" : "ABX", "value" : 4.829179077594706 }
Type "it" for more
>
```

2] The average price of stocks in the above map reduce will not be correct as AVERAGE is a commutative operation but nor associative. Use a FINALIZER to find the correct average.

Map Function-

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

Reduced Function-

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

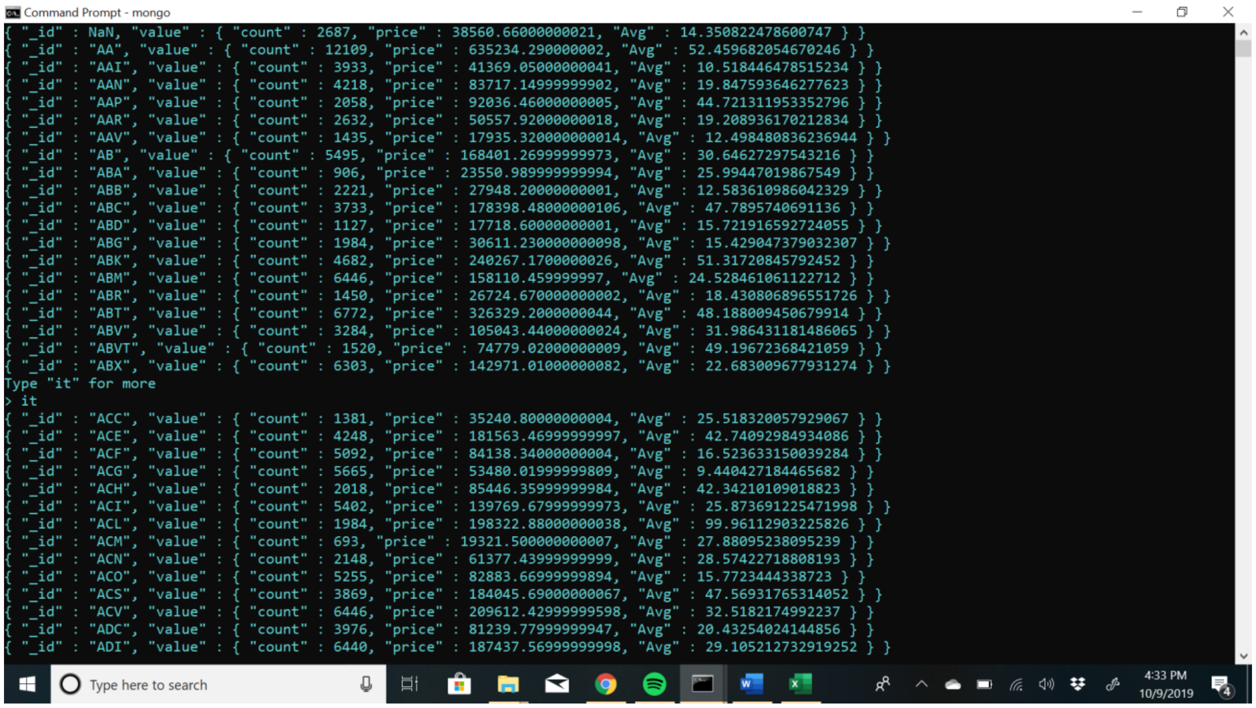
return reducedVal;
}

Finalize Function:

function (key, reducedVal) {   reducedVal.avg = reducedVal.price/reducedVal.count;
 return reducedVal; }

Map reduce Using Finalizer

db.NYSE_daily_prices_All.mapReduce(mapFunction1,reduceFunction1,{out:{merge:"avg_Stock_high_finalize"},finalize:FinalizeFunction1})



```
{ "_id" : NaN, "value" : { "count" : 2687, "price" : 38560.66000000021, "Avg" : 14.350822478600747 } }
{ "_id" : "AA", "value" : { "count" : 12109, "price" : 635234.290000002, "Avg" : 52.459682054670246 } }
{ "_id" : "AAI", "value" : { "count" : 3933, "price" : 41369.05000000041, "Avg" : 10.518446478515234 } }
{ "_id" : "AAN", "value" : { "count" : 4218, "price" : 83717.14999999902, "Avg" : 19.847593646277623 } }
{ "_id" : "AAP", "value" : { "count" : 2058, "price" : 92036.46000000005, "Avg" : 44.721311953352796 } }
{ "_id" : "AAR", "value" : { "count" : 2632, "price" : 50557.92000000018, "Avg" : 19.208936170212834 } }
{ "_id" : "AAV", "value" : { "count" : 1435, "price" : 17935.320000000014, "Avg" : 12.498480836236944 } }
{ "_id" : "AB", "value" : { "count" : 5495, "price" : 168401.26999999973, "Avg" : 30.64627297543216 } }
{ "_id" : "ABA", "value" : { "count" : 906, "price" : 23550.989999999994, "Avg" : 25.99447019867549 } }
{ "_id" : "ABB", "value" : { "count" : 2221, "price" : 27948.20000000001, "Avg" : 12.583610986042329 } }
{ "_id" : "ABC", "value" : { "count" : 3733, "price" : 178398.48000000106, "Avg" : 47.7895740691136 } }
{ "_id" : "ABD", "value" : { "count" : 1127, "price" : 17718.60000000001, "Avg" : 15.72191659272405 } }
{ "_id" : "ABG", "value" : { "count" : 1984, "price" : 30611.230000000098, "Avg" : 15.429047379032307 } }
{ "_id" : "ABK", "value" : { "count" : 4682, "price" : 240267.1700000026, "Avg" : 51.31720845792452 } }
{ "_id" : "ABM", "value" : { "count" : 6446, "price" : 158110.459999997, "Avg" : 24.528461061122712 } }
{ "_id" : "ABR", "value" : { "count" : 1450, "price" : 26724.670000000002, "Avg" : 18.430806896551726 } }
{ "_id" : "ABT", "value" : { "count" : 6772, "price" : 326329.2000000044, "Avg" : 48.188009450679914 } }
{ "_id" : "ABV", "value" : { "count" : 3284, "price" : 105043.44000000024, "Avg" : 31.986431181486065 } }
{ "_id" : "ABVT", "value" : { "count" : 1520, "price" : 74779.02000000009, "Avg" : 49.19672368421059 } }
{ "_id" : "ABX", "value" : { "count" : 6303, "price" : 142971.01000000082, "Avg" : 22.683009677931274 } }
Type "it" for more
> it
{ "_id" : "ACC", "value" : { "count" : 1381, "price" : 35240.80000000004, "Avg" : 25.51832057929067 } }
{ "_id" : "ACE", "value" : { "count" : 4248, "price" : 181563.46999999997, "Avg" : 42.7409298493408 } }
{ "_id" : "ACF", "value" : { "count" : 5092, "price" : 84138.34000000004, "Avg" : 16.523633150039284 } }
{ "_id" : "ACG", "value" : { "count" : 5665, "price" : 53480.01999999809, "Avg" : 9.440427184465682 } }
{ "_id" : "ACH", "value" : { "count" : 2018, "price" : 85446.35999999984, "Avg" : 42.3421010901882 } }
{ "_id" : "ACI", "value" : { "count" : 5402, "price" : 139769.67999999973, "Avg" : 25.873691225471998 } }
{ "_id" : "ACL", "value" : { "count" : 1984, "price" : 198322.88000000038, "Avg" : 99.96112903225826 } }
{ "_id" : "ACM", "value" : { "count" : 693, "price" : 19321.500000000007, "Avg" : 27.88095238095239 } }
{ "_id" : "ACN", "value" : { "count" : 2148, "price" : 61377.43999999999, "Avg" : 28.57422718808193 } }
{ "_id" : "ACO", "value" : { "count" : 5255, "price" : 82883.66999999894, "Avg" : 15.7723444338723 } }
{ "_id" : "ACS", "value" : { "count" : 3869, "price" : 184045.69000000067, "Avg" : 47.56931765314052 } }
{ "_id" : "ACV", "value" : { "count" : 6446, "price" : 209612.42999999598, "Avg" : 32.5182174992237 } }
{ "_id" : "ADC", "value" : { "count" : 3976, "price" : 81239.77999999947, "Avg" : 20.43254024144856 } }
{ "_id" : "ADI", "value" : { "count" : 6440, "price" : 187437.56999999998, "Avg" : 29.105212732919252 } }
```

PART 3

3] Adding a finalizer to find out the average stock price of each price of all stocks in the finalizer.

Map Function:

```
function()
 {      var value = {  count:1,
                          price_open:this.stock_price_open,
                          price_high:this.stock_price_high,
                          price_low:this.stock_price_low,
                         price_close:this.stock_price_close,      };
       var key = this.stock_symbol;
       emit(key, value);
 }
```

Reduce Function

```
function(key, values)
 {   reducedVal1 = { count:0,price_open:0,price_high:0,price_low:0,price_close:0};
for (var i= 0; i < values.length; i++)
 {  reducedVal1.count += values[i].count;
  reducedVal1.price_open += values[i].price_open
reducedVal1.price_high += values[i].price_high
reducedVal1.price_low += values[i].price_low
reducedVal1.price_close += values[i].price_close    ;
 }
return reducedVal1;
}
```

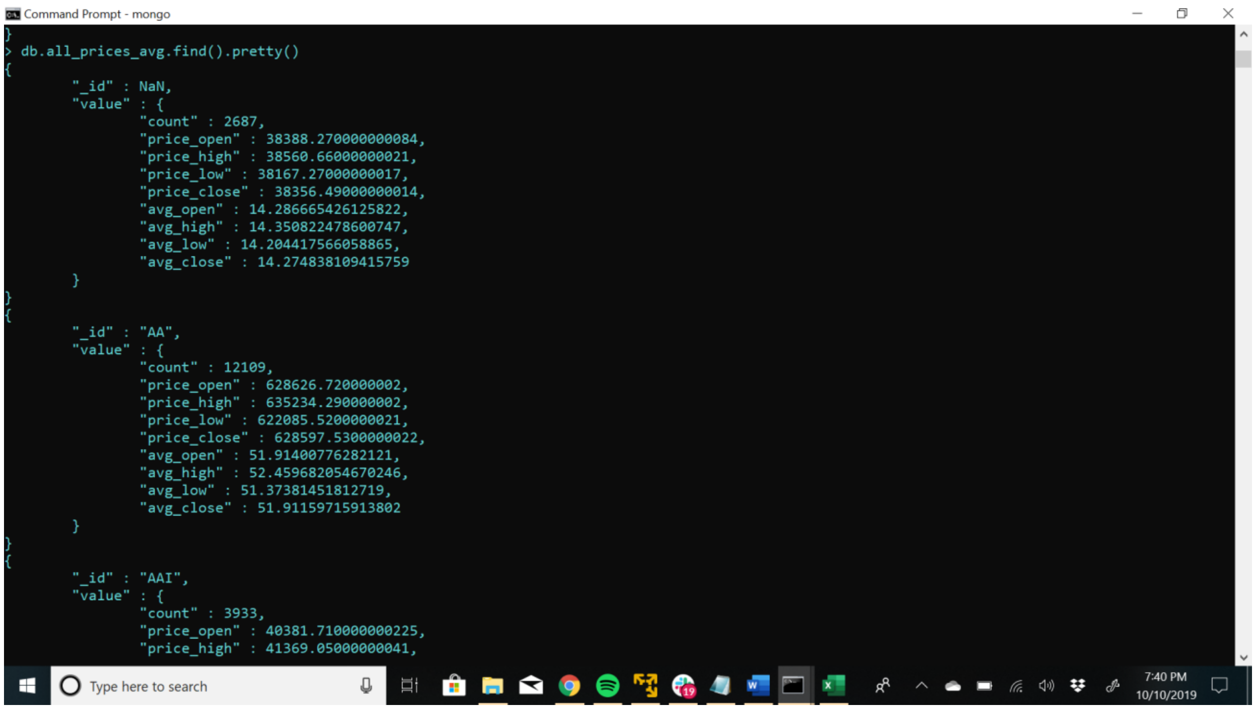
Finalize Function:

```
function (key, reducedVal1) {
reducedVal1.avg_open= reducedVal1.price_open/reducedVal1.count;
reducedVal1.avg_high= reducedVal1.price_high/reducedVal1.count;
reducedVal1.avg_low= reducedVal1.price_low/reducedVal1.count;
reducedVal1.avg_close= reducedVal1.price_close/reducedVal1.count;
 return reducedVal1; }
```

Using Finalizer

```
db.NYSE_daily_prices_All.mapReduce(mapFunctionStock1,reduceFunctionStock1,{out:{merge:"all_prices_avg"},finalize:finalizeFunctionStock1})
```

OUTPUT



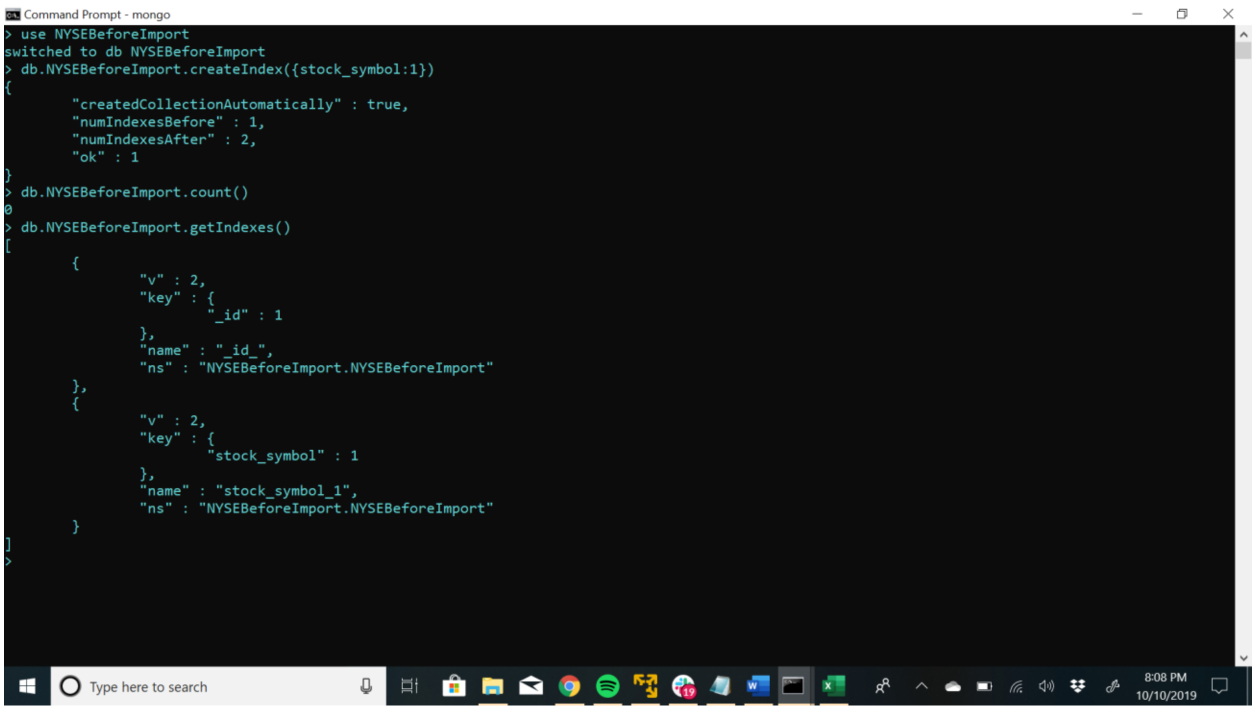
```
}
> db.all_prices_avg.find().pretty()
{
        "_id" : NaN,
        "value" : {
                "count" : 2687,
                "price_open" : 38388.270000000084,
                "price_high" : 38560.66000000021,
                "price_low" : 38167.27000000017,
                "price_close" : 38356.49000000014,
                "avg_open" : 14.286665426125822,
                "avg_high" : 14.350822478600747,
                "avg_low" : 14.204417566058865,
                "avg_close" : 14.274838109415759
        }
}
{
        "_id" : "AA",
        "value" : {
                "count" : 12109,
                "price_open" : 628626.720000002,
                "price_high" : 635234.290000002,
                "price_low" : 622085.5200000021,
                "price_close" : 628597.5300000022,
                "avg_open" : 51.91400776282121,
                "avg_high" : 52.459682054670246,
                "avg_low" : 51.37381451812719,
                "avg_close" : 51.91159715913802
        }
}
{
        "_id" : "AAI",
        "value" : {
                "count" : 3933,
                "price_open" : 40381.710000000225,
                "price_high" : 41369.05000000041,
```

MongoDB indexing Before putting into production:

use NYSEBeforeImport

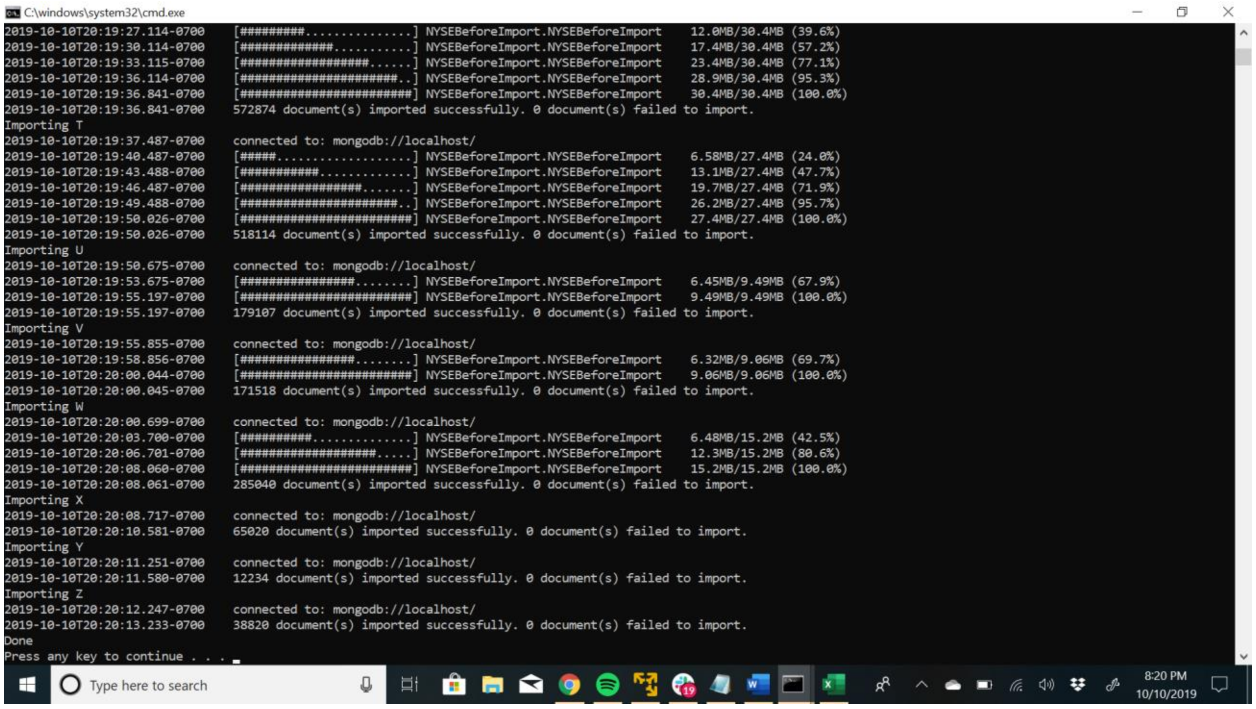
db.NYSEBeforeIndex.createIndex({stock_symbol:1})

mongoimport --db NYSEBeforeImport --collection NYSEBeforeImport --type csv --headerline --file C:\Users\SayaliGirish\Desktop\NYSE\NYSE_daily_prices.csv



```
> use NYSEBeforeImport
switched to db NYSEBeforeImport
> db.NYSEBeforeImport.createIndex({stock_symbol:1})
{
        "createdCollectionAutomatically" : true,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.NYSEBeforeImport.count()
0
> db.NYSEBeforeImport.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "NYSEBeforeImport.NYSEBeforeImport"
        },
        {
                "v" : 2,
                "key" : {
                        "stock_symbol" : 1
                },
                "name" : "stock_symbol_1",
                "ns" : "NYSEBeforeImport.NYSEBeforeImport"
        }
]
>
```

Created the index first and then imported data into collection



C:\windows\system32\cmd.exe

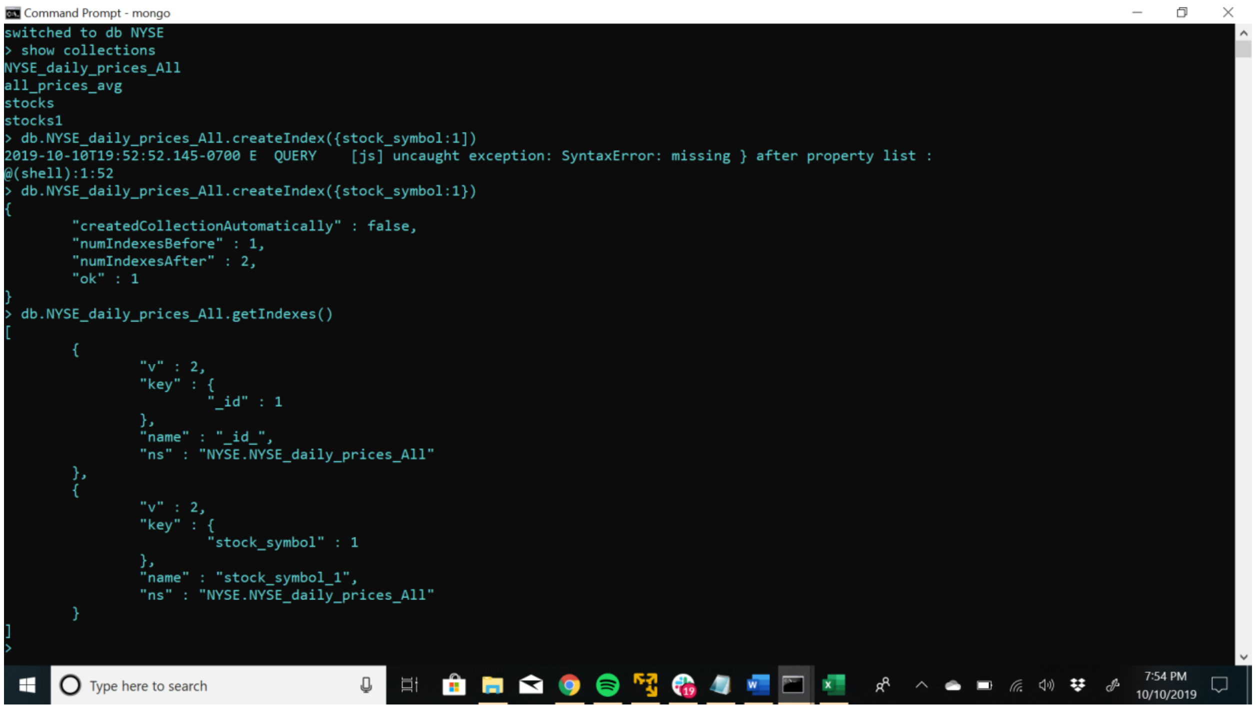
```
2019-10-10T20:19:27.114-0700    [#########...............] NYSEBeforeImport.NYSEBeforeImport    12.0MB/30.4MB (39.6%)
2019-10-10T20:19:30.114-0700    [#############...........] NYSEBeforeImport.NYSEBeforeImport    17.4MB/30.4MB (57.2%)
2019-10-10T20:19:33.115-0700    [##################......] NYSEBeforeImport.NYSEBeforeImport    23.4MB/30.4MB (77.1%)
2019-10-10T20:19:36.114-0700    [######################..] NYSEBeforeImport.NYSEBeforeImport    28.9MB/30.4MB (95.3%)
2019-10-10T20:19:36.841-0700    [########################] NYSEBeforeImport.NYSEBeforeImport    30.4MB/30.4MB (100.0%)
2019-10-10T20:19:36.841-0700    572874 document(s) imported successfully. 0 document(s) failed to import.
Importing T
2019-10-10T20:19:37.487-0700    connected to: mongodb://localhost/
2019-10-10T20:19:40.487-0700    [#####...................] NYSEBeforeImport.NYSEBeforeImport    6.58MB/27.4MB (24.0%)
2019-10-10T20:19:43.488-0700    [###########.............] NYSEBeforeImport.NYSEBeforeImport    13.1MB/27.4MB (47.7%)
2019-10-10T20:19:46.487-0700    [#################.......] NYSEBeforeImport.NYSEBeforeImport    19.7MB/27.4MB (71.9%)
2019-10-10T20:19:49.488-0700    [######################..] NYSEBeforeImport.NYSEBeforeImport    26.2MB/27.4MB (95.7%)
2019-10-10T20:19:50.026-0700    [########################] NYSEBeforeImport.NYSEBeforeImport    27.4MB/27.4MB (100.0%)
2019-10-10T20:19:50.026-0700    518114 document(s) imported successfully. 0 document(s) failed to import.
Importing U
2019-10-10T20:19:50.675-0700    connected to: mongodb://localhost/
2019-10-10T20:19:53.675-0700    [################........] NYSEBeforeImport.NYSEBeforeImport    6.45MB/9.49MB (67.9%)
2019-10-10T20:19:55.197-0700    [########################] NYSEBeforeImport.NYSEBeforeImport    9.49MB/9.49MB (100.0%)
2019-10-10T20:19:55.197-0700    179107 document(s) imported successfully. 0 document(s) failed to import.
Importing V
2019-10-10T20:19:55.855-0700    connected to: mongodb://localhost/
2019-10-10T20:19:58.856-0700    [################........] NYSEBeforeImport.NYSEBeforeImport    6.32MB/9.06MB (69.7%)
2019-10-10T20:20:00.044-0700    [########################] NYSEBeforeImport.NYSEBeforeImport    9.06MB/9.06MB (100.0%)
2019-10-10T20:20:00.045-0700    171518 document(s) imported successfully. 0 document(s) failed to import.
Importing W
2019-10-10T20:20:00.699-0700    connected to: mongodb://localhost/
2019-10-10T20:20:03.700-0700    [##########..............] NYSEBeforeImport.NYSEBeforeImport    6.48MB/15.2MB (42.5%)
2019-10-10T20:20:06.701-0700    [###################.....] NYSEBeforeImport.NYSEBeforeImport    12.3MB/15.2MB (80.6%)
2019-10-10T20:20:08.060-0700    [########################] NYSEBeforeImport.NYSEBeforeImport    15.2MB/15.2MB (100.0%)
2019-10-10T20:20:08.061-0700    285040 document(s) imported successfully. 0 document(s) failed to import.
Importing X
2019-10-10T20:20:08.717-0700    connected to: mongodb://localhost/
2019-10-10T20:20:10.581-0700    65020 document(s) imported successfully. 0 document(s) failed to import.
Importing Y
2019-10-10T20:20:11.251-0700    connected to: mongodb://localhost/
2019-10-10T20:20:11.580-0700    12234 document(s) imported successfully. 0 document(s) failed to import.
Importing Z
2019-10-10T20:20:12.247-0700    connected to: mongodb://localhost/
2019-10-10T20:20:13.233-0700    38820 document(s) imported successfully. 0 document(s) failed to import.
Done
Press any key to continue . . .
```

Type here to search

8:20 PM
10/10/2019

Insert the NYSE dataset into a new database. You may use the existing NYSE database created before.

Now, create indexes on existing data sets.



Command Prompt - mongo

```
switched to db NYSE
> show collections
NYSE_daily_prices_All
all_prices_avg
stocks
stocks1
> db.NYSE_daily_prices_All.createIndex({stock_symbol:1]})
2019-10-10T19:52:52.145-0700 E  QUERY    [js] uncaught exception: SyntaxError: missing } after property list :
@(shell):1:52
> db.NYSE_daily_prices_All.createIndex({stock_symbol:1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.NYSE_daily_prices_All.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "NYSE.NYSE_daily_prices_All"
        },
        {
                "v" : 2,
                "key" : {
                        "stock_symbol" : 1
                },
                "name" : "stock_symbol_1",
                "ns" : "NYSE.NYSE_daily_prices_All"
        }
]
>
```

Type here to search

7:54 PM
10/10/2019

1] 2numIndexesBefore: 1 indicates the number of Field values (The actual fields in the collection) which were there in the indexes before the command was run. Remember that each collection has the _id field which also counts as a Field value to the index. Since the _id index field is part of the collection when it is initially created, the value of numIndexesBefore is 1.

2] numIndexesAfter: 2 indicates the number of Field values which were there in the indexes after the command was run.

3] "ok: 1" output specifies that the operation was successful, and the new index is added to the collection.