

Assignment 4

Jiarong Ye

November 8, 2018

Import Packages

```
In [1]: import pandas as pd
import json
```

Task 1

Insert 4 vehicles from the State College Auto Database (can be the same ones you used in Redis) into the MongoDB. One easy way to do this is to create a .json file from the data and then import the .json file into a new data collection using the 'mongoimport' package as you did in HW #3 to import the restaurant dataset.

```
In [14]: cars = pd.read_csv('cars.csv',sep=',',index_col=0)
with open('cars.json', 'w') as outfile:
    cars_dict = cars.to_dict('index')
    for i in range(len(cars_dict)):
        json.dump(cars_dict[i], outfile)
```

```
In [15]: ! mongoimport --db local --collection cars --file cars.json
```

```
2018-11-06T01:22:33.494-0500      connected to: localhost
2018-11-06T01:22:34.035-0500      imported 10 documents
```

```
localhost:27017 (v3.2.19) local
1 db.cars.find({})

cars 0.014 s 10 Docs
1 /* 1 createdAt:11/6/2018, 1:22:33 AM*/
2 {
3   "_id" : ObjectId("5be13329f370999bd25887f4"),
4   "Title" : "2013 Subaru XV Crosstrek",
5   "Year" : 2013,
6   "Mileage" : 69331,
7   "Make" : "Subaru",
8   "Model" : "XV Crosstrek",
9   "Trim" : "Premium",
10  "Style" : "Station Wagon",
11  "Engine" : "4 -",
12  "Exterior Color" : "Tangerine Orange Pea",
13  "Interior Color" : "Black",
14  "VIN" : "JF2GPACXD1843307",
15  "Stock #" : "204842B",
16  "description" : "Clean CARFAX. Orange 2013 Subaru XV Crosstrek 2.0i Premium AWD 5-Speed Manual 2.0L 16V DOHC",
17 },
18
19 /* 2 createdAt:11/6/2018, 1:22:33 AM*/
20 {
21   "_id" : ObjectId("5be13329f370999bd25887f5"),
22   "Title" : "2014 Subaru Outback",
23   "Year" : 2014,
24   "Mileage" : 75655,
25   "Make" : "Subaru",
26   "Model" : "Outback",
27   "Trim" : "2.5i Premium",
28   "Style" : "Sport Utility",
29   "Engine" : "4 -",
30   "Exterior Color" : "Twilight Blue Metall",
31   "Interior Color" : "Black",
32   "VIN" : "4S4BRBCC4E3219562",
33   "Stock #" : "606614A",
34   "description" : "CARFAX One-Owner. Clean CARFAX. Blue 2014 Subaru Outback 2.5i Premium AWD 6-Speed 2.5L 4-Cyl",
35 },
36
37 /* 3 createdAt:11/6/2018, 1:22:33 AM*/
38 {
39   "_id" : ObjectId("5be13329f370999bd25887f6"),
40   "Title" : "2015 Chevrolet Malibu",
41   "Year" : 2015,
42   "Mileage" : 25757,
```

Task 2

Query your database and demonstrate that you can search on at least 2 attributes of the vehicles. Demonstrate a query on a single attribute and a query on more than one attribute.

- query on a single attribute

```
localhost:27017 (v3.2.19) local
1 db.cars.find({'Year': {'$gt': 2015}}, {'Title':1, 'Make':1, 'Year':1, 'Model':1, 'VIN':1})

cars 0.016 s 4 Docs
1 /* 1 createdAt:11/6/2018, 1:22:33 AM*/
2 - {
3   "_id" : ObjectId("5be13329f370999bd25887f9"),
4   "Title" : "2016 Chevrolet Malibu Limited",
5   "Year" : 2016,
6   "Make" : "Chevrolet",
7   "Model" : "Malibu Limited",
8   "VIN" : "1G11B5SAXG135777"
9 },
10
11 /* 2 createdAt:11/6/2018, 1:22:33 AM*/
12 - {
13   "_id" : ObjectId("5be13329f370999bd25887fa"),
14   "Title" : "2017 Subaru Legacy",
15   "Year" : 2017,
16   "Make" : "Subaru",
17   "Model" : "Legacy",
18   "VIN" : "4S3BNEN6XH3029939"
19 },
20
21 /* 3 createdAt:11/6/2018, 1:22:33 AM*/
22 - {
23   "_id" : ObjectId("5be13329f370999bd25887fb"),
24   "Title" : "2017 Subaru Outback",
25   "Year" : 2017,
26   "Make" : "Subaru",
27   "Model" : "Outback",
28   "VIN" : "4S4BSANC0H3391975"
29 },
30
31 /* 4 createdAt:11/6/2018, 1:22:33 AM*/
32 - {
33   "_id" : ObjectId("5be13329f370999bd25887fd"),
34   "Title" : "2016 Subaru Forester",
35   "Year" : 2016,
36   "Make" : "Subaru",
37   "Model" : "Forester",
38   "VIN" : "JF2SJAA3GG451169"
39 }
```

- query on more than one attribute

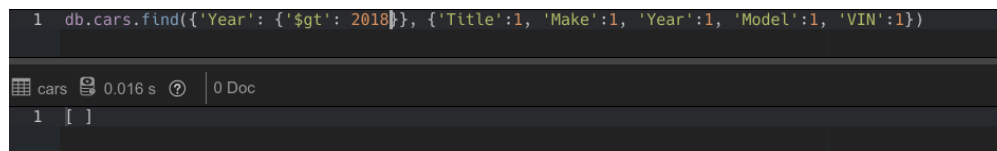
```
localhost:27017 (v3.2.19) local
1 db.cars.find({'Year': {'$gt': 2015}, 'Make': 'Subaru'}, {'Title':1, 'Make':1, 'Year':1, 'Model':1, 'VIN':1})

cars 0.014 s 3 Docs
1 /* 1 createdAt:11/6/2018, 1:22:33 AM*/
2 - {
3   "_id" : ObjectId("5be13329f370999bd25887fa"),
4   "Title" : "2017 Subaru Legacy",
5   "Year" : 2017,
6   "Make" : "Subaru",
7   "Model" : "Legacy",
8   "VIN" : "4S3BNEN6XH3029939"
9 },
10
11 /* 2 createdAt:11/6/2018, 1:22:33 AM*/
12 - {
13   "_id" : ObjectId("5be13329f370999bd25887fb"),
14   "Title" : "2017 Subaru Outback",
15   "Year" : 2017,
16   "Make" : "Subaru",
17   "Model" : "Outback",
18   "VIN" : "4S4BSANC0H3391975"
19 },
20
21 /* 3 createdAt:11/6/2018, 1:22:33 AM*/
22 - {
23   "_id" : ObjectId("5be13329f370999bd25887fd"),
24   "Title" : "2016 Subaru Forester",
25   "Year" : 2016,
26   "Make" : "Subaru",
27   "Model" : "Forester",
28   "VIN" : "JF2SJAA3GG451169"
29 }
```

Task 3

Demonstrate a query for a vehicle not in the database.

```
1 db.cars.find({'Year': {'$gt': 2018}}, {'Title':1, 'Make':1, 'Year':1, 'Model':1, 'VIN':1})
```



The screenshot shows the MongoDB Shell interface. The command entered is `1 db.cars.find({'Year': {'$gt': 2018}}, {'Title':1, 'Make':1, 'Year':1, 'Model':1, 'VIN':1})`. Below the command bar, the status bar indicates the collection is 'cars', the execution time is '0.016 s', and there are '0 Doc'uments. The results pane shows a single row with the value `1` in the first column and an empty array `[]` in the second column, representing an empty set.

It returns an empty set.

Task 4

What are differences you found between using MongoDB and Redis for loading and storing this data? What would be a role for a database like Redis in storing this vehicle information? Can you imagine challenges in using the method you used for data import into MongoDB when dealing with Big Data? How might you address any challenges in a big data environment?

With mongoDB we can import the whole dataset in the json format directly while in Redis, we have to import one key-value pair at a time. In this case, Redis acts like a dictionary in python used to store the nested dictionaries (key-value pairs).

- challenges:
 - the dataset to insert is too large
 - take up too much memory
 - it takes too long to import the whole dataset
- possible solutions to address the challenges:
 - import the data in batches
 - * `mongoimport --db local --collection cars --file cars.json --batchSize 1`
 - disable journaling to avoid further memory consumption when tracking the changes made
 - * `mongod --nojournal`
 - import the dataset in parallel with `--numInsertionWorkers`
 - * `mongoimport --db local --collection cars --file cars.json --numInsertionWorkers 8`
 - convert text data into bson format
 - use GridFS when the file exceeds the size limit of 16MB
 - * `mongofiles put cars.json`
 - if there are complex analyses on the data involved before importing, such as applying a machine learning algorithm to do classification, a possible way is to use `mongodb-spark-connector` to analyze with `spark mllib`, then import dataset from `hdfs` back to `mongodb`:

```
In [ ]: hdfs dfs -put cars.csv /user/data
        pyspark --packages org.mongodb.spark:mongo-spark-connector_2.11:2.2.1
               --packages org.mongodb:mongo-java-driver:3.8.0
               --packages org.apache.spark:spark-sql_2.11:2.3.0
               --packages com.stratio.datasources:spark-mongodb_2.11:0.12.0
               --packages org.mongodb:casbah_2.11:3.0.0
               --packages org.apache.spark:spark-catalyst_2.11:2.2.1
```

```
In [1]: from pyspark.sql import SparkSession, Row, functions
```

```
def parseInput(line):
    fields = line.split(',')
    return Row(Title = fields[1],
               Year = fields[2],
               Mileage = fields[3],
               Make = fields[4],
               Model = fields[5],
               Trim = fields[6],
               Style = fields[7],
               Engine = fields[8],
               Exterior_Color = fields[9],
               Interior_Color = fields[10],
               VIN = fields[11],
               Stock = fields[12],
               description = fields[13])

if __name__ == '__main__':
    spark = SparkSession.builder.appName('MongoDBIntegration').getOrCreate()
    lines = spark.sparkContext.textFile('hdfs://localhost:9002/user/data/cars.json')
    cars = lines.map(parseInput)
    carsDataset = spark.createDataFrame(cars)

    # write into MongoDB
    carsDataset.write \
        .format('com.stratio.datasources.mongodb') \
        .options(host="localhost:27017", database="local", collection="cars2") \
        .mode('append') \
        .save()
```

localhost:27017 (v3.2.19) local

```
1 db.cars2.find({})
```

cars2 0.013 s 10 Docs

```

1 /* 1 createdAt:11/8/2018, 9:32:39 PM*/
2 {
3   "id" : ObjectId("5be4f1c77c14805b3ab257e3"),
4   "Engine" : "4 -",
5   "Exterior_Color" : "Champagne Silver Met",
6   "Interior_Color" : "Jet Black/Titanium",
7   "Make" : "Chevrolet",
8   "Mileage" : "52582",
9   "Model" : "Malibu Limited",
10  "Stock" : "15498A",
11  "Style" : "4dr Car",
12  "Title" : "2016 Chevrolet Malibu Limited",
13  "Trim" : "LS",
14  "VIN" : "1G11B5SAXGF135777",
15  "Year" : "2016",
16  "description" : "\CARFAX One-Owner. Gold 2016 Chevrolet Malibu Limited LS FWD 6-Speed Automatic Electronic w
17 },
18
19 /* 2 createdAt:11/8/2018, 9:32:39 PM*/
20 {
21   "id" : ObjectId("5be4f1c77c14805b3ab257e4"),
22   "Engine" : "4 -",
23   "Exterior_Color" : "Ice Silver Metallic",
24   "Interior_Color" : "Black",
25   "Make" : "Subaru",
26   "Mileage" : "16994",
27   "Model" : "Forester",
28   "Stock" : "606499A",
29   "Style" : "Sport Utility",
30   "Title" : "2016 Subaru Forester",
31   "Trim" : "2.5i",
32   "VIN" : "JF2SJJAAC3GG451169",
33   "Year" : "2016",
34   "description" : "\CARFAX One-Owner. Clean CARFAX. Ice Silver Metallic 2016 Subaru Forester 2.5i AWD 6-Speed
35 },
36
37 /* 3 createdAt:11/8/2018, 9:32:39 PM*/
38 {
39   "id" : ObjectId("5be4f1c77c14805b3ab257e5"),
40   "Engine" : "8 -",
41   "Exterior_Color" : "Summit White",
42   "Interior_Color" : "Jet Black",

```