BridgeAnalytics

March 18, 2017

1 Jupyter and MongoDB Test with Docker Compose

```python
In [17]: #Import modules
         import json
         import random
         import pprint

         #Function to connect to MongoDB Container
         def get_db():
             from pymongo import MongoClient
             client = MongoClient('datascience_db_1:27017')
             db = client.testDB
             return db

         #Function to add cracks to MongoDB
         def add_crack(db):
             for i in range (1, 10000):
                 crack = {}
                 crack['crack'] = "crack" + str(i)
                 crack['width'] = round(random.random(),2)
                 db.cracks.insert_one(crack)

         #Function to return cracks in MongoDB
         def get_cracks(db):
             return db.cracks.find()

         #Initialize DB
         db = get_db()
         print(db)
```

```
Database(MongoClient(host=['datascience_db_1:27017'], document_class=dict, tz_aware=False, conne
```

```python
In [9]: add_crack(db)
```

Iterate through all the cracks in the DB

```
In [56]: # Print all cracks
         for i in get_cracks(db):
             print(i)
```

Count number of cracks in the database

```
In [10]: db.cracks.count()
```

```
Out[10]: 9999
```

2 Invoke the MongoDB Aggregation Framework

```
In [25]: # Pipeline to compute avergage, max and min width of cracks
         pipeline = [{
             "$group": {
                 "_id": "null",
                 "min_width": {
                     "$min": "$width"
                 },
                 "avg_width": {
                     "$avg": "$width"
                 },
                 "max_width": {
                     "$max": "$width"
                 }
             }
         }]

         pprint.pprint(list((db.cracks.aggregate(pipeline))))

[{'_id': 'null',
  'avg_width': 0.503012301230123,
  'max_width': 1.0,
  'min_width': 0.0}]
```

3 Import MongoDB data into a DataFrame

```
In [27]: import pandas as pd

         cursor = db.cracks.find()
         df =  pd.DataFrame(list(cursor))
         len(df)
         print(df)

                          _id      crack  width
0     58cda73460589b000a58e3a8    crack1   0.36
1     58cda73460589b000a58e3a9    crack2   0.55
```

```
2       58cda73460589b000a58e3aa      crack3     0.89
3       58cda73460589b000a58e3ab      crack4     0.25
4       58cda73460589b000a58e3ac      crack5     0.06
5       58cda73460589b000a58e3ad      crack6     0.62
6       58cda73460589b000a58e3ae      crack7     0.11
7       58cda73460589b000a58e3af      crack8     0.26
8       58cda73460589b000a58e3b0      crack9     0.15
9       58cda73460589b000a58e3b1      crack10    0.98
10      58cda73460589b000a58e3b2      crack11    0.80
11      58cda73460589b000a58e3b3      crack12    0.91
12      58cda73460589b000a58e3b4      crack13    0.88
13      58cda73460589b000a58e3b5      crack14    0.96
14      58cda73460589b000a58e3b6      crack15    0.27
15      58cda73460589b000a58e3b7      crack16    0.08
16      58cda73460589b000a58e3b8      crack17    0.07
17      58cda73460589b000a58e3b9      crack18    0.66
18      58cda73460589b000a58e3ba      crack19    0.59
19      58cda73460589b000a58e3bb      crack20    0.98
20      58cda73460589b000a58e3bc      crack21    0.59
21      58cda73460589b000a58e3bd      crack22    0.28
22      58cda73460589b000a58e3be      crack23    0.49
23      58cda73460589b000a58e3bf      crack24    0.27
24      58cda73460589b000a58e3c0      crack25    0.21
25      58cda73460589b000a58e3c1      crack26    0.02
26      58cda73460589b000a58e3c2      crack27    0.24
27      58cda73460589b000a58e3c3      crack28    0.54
28      58cda73460589b000a58e3c4      crack29    0.73
29      58cda73460589b000a58e3c5      crack30    0.44
...                      ...           ...        ...
9969    58cda73860589b000a590a99      crack9970  0.02
9970    58cda73860589b000a590a9a      crack9971  0.33
9971    58cda73860589b000a590a9b      crack9972  0.54
9972    58cda73860589b000a590a9c      crack9973  0.42
9973    58cda73860589b000a590a9d      crack9974  0.54
9974    58cda73860589b000a590a9e      crack9975  0.84
9975    58cda73860589b000a590a9f      crack9976  0.28
9976    58cda73860589b000a590aa0      crack9977  0.98
9977    58cda73860589b000a590aa1      crack9978  0.50
9978    58cda73860589b000a590aa2      crack9979  0.54
9979    58cda73860589b000a590aa3      crack9980  0.48
9980    58cda73860589b000a590aa4      crack9981  0.68
9981    58cda73860589b000a590aa5      crack9982  0.38
9982    58cda73860589b000a590aa6      crack9983  0.51
9983    58cda73860589b000a590aa7      crack9984  0.56
9984    58cda73860589b000a590aa8      crack9985  0.52
9985    58cda73860589b000a590aa9      crack9986  0.18
9986    58cda73860589b000a590aaa      crack9987  0.89
9987    58cda73860589b000a590aab      crack9988  0.82
```

```
9988  58cda73860589b000a590aac  crack9989  0.44
9989  58cda73860589b000a590aad  crack9990  0.12
9990  58cda73860589b000a590aae  crack9991  0.35
9991  58cda73860589b000a590aaf  crack9992  0.95
9992  58cda73860589b000a590ab0  crack9993  0.85
9993  58cda73860589b000a590ab1  crack9994  0.29
9994  58cda73860589b000a590ab2  crack9995  0.88
9995  58cda73860589b000a590ab3  crack9996  0.46
9996  58cda73860589b000a590ab4  crack9997  0.22
9997  58cda73860589b000a590ab5  crack9998  0.36
9998  58cda73860589b000a590ab6  crack9999  0.33

[9999 rows x 3 columns]
```

3.1   Drop the database

```
In [7]: db.cracks.drop()

In [ ]:
```