031-data-wrangling-with-mongodb

April 23, 2022

3.1. Wrangling Data with MongoDB

[2]:
```
from pprint import PrettyPrinter

import pandas as pd
from IPython.display import VimeoVideo
from pymongo import MongoClient
```

[3]:
```
VimeoVideo("665412094", h="8334dfab2e", width=600)
```

[3]:
```
<IPython.lib.display.VimeoVideo at 0x7fb54447b460>
```

[4]:
```
VimeoVideo("665412135", h="dcff7ab83a", width=600)
```

[4]:
```
<IPython.lib.display.VimeoVideo at 0x7fb54447b1c0>
```

Task 3.1.1: Instantiate a `PrettyPrinter`, and assign it to the variable `pp`.

- Construct a `PrettyPrinter` instance in pprint.

[5]:
```
pp = PrettyPrinter(indent = 2)
```

1 Prepare Data

1.1 Connect

[6]:
```
VimeoVideo("665412155", h="1ca0dd03d0", width=600)
```

[6]:
```
<IPython.lib.display.VimeoVideo at 0x7fb54447ba30>
```

Task 3.1.2: Create a client that connects to the database running at `localhost` on port `27017`.

- What's a database client?
- What's a database server?
- Create a client object for a MongoDB instance.

[7]:
```
client = MongoClient(host="localhost", port=27017)
```

1.2 Explore

[8]:
```
VimeoVideo("665412176", h="6fea7c6346", width=600)
```

[8]:
```
<IPython.lib.display.VimeoVideo at 0x7fb54437fd60>
```

[9]:
```
from sys import getsizeof
my_list = [0,1,2,3,4,5] #list/ array
my_range = range(0,8_000_000) #iterator

# for i in my_list:
#     print(i)

# for i in my_range:
#     print(i)

print(getsizeof(my_list))
print(getsizeof(my_range))
```

```
152
48
```

Task 3.1.3: Print a list of the databases available on `client`.

- What's an iterator?
- List the databases of a server using PyMongo.
- Print output using pprint.

[10]:
```
db_list = list(client.list_databases())
#print(getsizeof(db_list))
pp.pprint(db_list)
```

```
[ {'empty': False, 'name': 'admin', 'sizeOnDisk': 40960},
  {'empty': False, 'name': 'air-quality', 'sizeOnDisk': 6987776},
  {'empty': False, 'name': 'config', 'sizeOnDisk': 12288},
  {'empty': False, 'name': 'local', 'sizeOnDisk': 73728}]
```

[11]:
```
VimeoVideo("665412216", h="7d4027dc33", width=600)
```

[11]:
```
<IPython.lib.display.VimeoVideo at 0x7fb542b112b0>
```

Task 3.1.4: Assign the `"air-quality"` database to the variable `db`.

- What's a MongoDB database?
- Access a database using PyMongo.

[12]:
```
db = client["air-quality"]
```

[13]:
```
VimeoVideo("665412231", h="89c546b00f", width=600)
```

[13]:
```
<IPython.lib.display.VimeoVideo at 0x7fb542b118b0>
```

Task 3.1.5: Use the `list_collections` method to print a list of the collections available in `db`.

- What's a MongoDB collection?
- List the collections in a database using PyMongo.

[14]:
```
#list(db.list_collections())[0]
for c in db.list_collections():
    print(c["name"])
```

```
lagos
system.buckets.lagos
nairobi
system.buckets.nairobi
system.views
dar-es-salaam
system.buckets.dar-es-salaam
```

[15]:
```
VimeoVideo("665412252", h="bff2abbdc0", width=600)
```

[15]:
```
<IPython.lib.display.VimeoVideo at 0x7fb542b11a30>
```

Task 3.1.6: Assign the `"nairobi"` collection in `db` to the variable name `nairobi`.

- Access a collection in a database using PyMongo.

[16]:
```
nairobi = db["nairobi"]
```

[17]:
```
VimeoVideo("665412270", h="e4a5f5c84b", width=600)
```

[17]:
```
<IPython.lib.display.VimeoVideo at 0x7fb542b32370>
```

Task 3.1.7: Use the `count_documents` method to see how many documents are in the `nairobi` collection.

- What's a MongoDB document?
- Count the documents in a collection using PyMongo.

[18]:
```
nairobi.count_documents({})
```

[18]:
```
202212
```

[19]:
```
VimeoVideo("665412279", h="c2315f3be1", width=600)
```

[19]:
```
<IPython.lib.display.VimeoVideo at 0x7fb542b326d0>
```

Task 3.1.8: Use the `find_one` method to retrieve one document from the `nairobi` collection, and assign it to the variable name `result`.

- What's metadata?

- What's semi-structured data?
- Retrieve a document from a collection using PyMongo.

[20]:
```
result = nairobi.find_one({})
pp.pprint(result)
```

```
{ 'P1': 39.67,
  '_id': ObjectId('6261a046e76424a61615daaf'),
  'metadata': { 'lat': -1.3,
                'lon': 36.785,
                'measurement': 'P1',
                'sensor_id': 57,
                'sensor_type': 'SDS011',
                'site': 29},
  'timestamp': datetime.datetime(2018, 9, 1, 0, 0, 2, 472000)}
```

[21]:
```
VimeoVideo("665412306", h="e1e913dfd1", width=600)
```

[21]:
```
<IPython.lib.display.VimeoVideo at 0x7fb542b320a0>
```

Task 3.1.9: Use the `distinct` method to determine how many sensor sites are included in the `nairobi` collection.

- Get a list of distinct values for a key among all documents using PyMongo.

[22]:
```
nairobi.distinct("metadata.site")
```

[22]:
```
[29, 6]
```

[23]:
```
VimeoVideo("665412322", h="4776c6d548", width=600)
```

[23]:
```
<IPython.lib.display.VimeoVideo at 0x7fb542b32eb0>
```

Task 3.1.10: Use the `count_documents` method to determine how many readings there are for each site in the `nairobi` collection.

- Count the documents in a collection using PyMongo.

[24]:
```
print("Documents from site 6:", nairobi.count_documents({"metadata.site":6}))
print("Documents from site 29:", nairobi.count_documents({"metadata.site":29}))
```

```
Documents from site 6: 70360
Documents from site 29: 131852
```

[25]:
```
VimeoVideo("665412344", h="d2354584cd", width=600)
```

[25]:
```
<IPython.lib.display.VimeoVideo at 0x7fb542b3d6d0>
```

Task 3.1.11: Use the `aggregate` method to determine how many readings there are for each site in the `nairobi` collection.

- Perform aggregation calculations on documents using PyMongo.

[26]:
```
result = nairobi.aggregate(
    [
        {"$group":{"_id":"$metadata.site","count":{"$count": {}}}}
    ]
)
pp.pprint(list(result))
```

```
[{'_id': 29, 'count': 131852}, {'_id': 6, 'count': 70360}]
```

[27]:
```
VimeoVideo("665412372", h="565122c9cc", width=600)
```

[27]:
```
<IPython.lib.display.VimeoVideo at 0x7fb542b3d7c0>
```

Task 3.1.12: Use the `distinct` method to determine how many types of measurements have been taken in the `nairobi` collection.

- Get a list of distinct values for a key among all documents using PyMongo.

[28]:
```
nairobi.distinct("metadata.measurement")
```

[28]:
```
['P2', 'humidity', 'temperature', 'P1']
```

[29]:
```
VimeoVideo("665412380", h="f7f7a39bb3", width=600)
```

[29]:
```
<IPython.lib.display.VimeoVideo at 0x7fb542b3d610>
```

Task 3.1.13: Use the `find` method to retrieve the PM 2.5 readings from all sites. Be sure to limit your results to 3 records only.

- Query a collection using PyMongo.

[30]:
```
result = nairobi.find({"metadata.measurement":"P2"}).limit(4)
pp.pprint(list(result))
```

```
[ { 'P2': 34.43,
    '_id': ObjectId('6261a046e76424a616165b3a'),
    'metadata': { 'lat': -1.3,
                  'lon': 36.785,
                  'measurement': 'P2',
                  'sensor_id': 57,
                  'sensor_type': 'SDS011',
                  'site': 29},
    'timestamp': datetime.datetime(2018, 9, 1, 0, 0, 2, 472000)},
  { 'P2': 30.53,
    '_id': ObjectId('6261a046e76424a616165b3b'),
    'metadata': { 'lat': -1.3,
                  'lon': 36.785,
                  'measurement': 'P2',
```

```
                'sensor_id': 57,
                'sensor_type': 'SDS011',
                'site': 29},
  'timestamp': datetime.datetime(2018, 9, 1, 0, 5, 3, 941000)},
 { 'P2': 22.8,
  '_id': ObjectId('6261a046e76424a616165b3c'),
  'metadata': { 'lat': -1.3,
                'lon': 36.785,
                'measurement': 'P2',
                'sensor_id': 57,
                'sensor_type': 'SDS011',
                'site': 29},
  'timestamp': datetime.datetime(2018, 9, 1, 0, 10, 4, 374000)},
 { 'P2': 13.3,
  '_id': ObjectId('6261a046e76424a616165b3d'),
  'metadata': { 'lat': -1.3,
                'lon': 36.785,
                'measurement': 'P2',
                'sensor_id': 57,
                'sensor_type': 'SDS011',
                'site': 29},
  'timestamp': datetime.datetime(2018, 9, 1, 0, 15, 4, 245000)}]
```

[31]:
```
VimeoVideo("665412389", h="8976ea3090", width=600)
```

[31]:
```
<IPython.lib.display.VimeoVideo at 0x7fb542b3d2b0>
```

Task 3.1.14: Use the `aggregate` method to calculate how many readings there are for each type (`"humidity"`, `"temperature"`, `"P2"`, and `"P1"`) in site 6.

- Perform aggregation calculations on documents using PyMongo.

[32]:
```
result = nairobi.aggregate(
    [
        {"$match": {"metadata.site":6}},
        {"$group":{"_id":"$metadata.measurement","count":{"$count": {}}}}
    ]
)
pp.pprint(list(result))
```

```
[ {'_id': 'P2', 'count': 18169},
  {'_id': 'humidity', 'count': 17011},
  {'_id': 'temperature', 'count': 17011},
  {'_id': 'P1', 'count': 18169}]
```

[33]:
```
VimeoVideo("665412418", h="0c4b125254", width=600)
```

[33]:
```
<IPython.lib.display.VimeoVideo at 0x7fb542b2f820>
```

Task 3.1.15: Use the `aggregate` method to calculate how many readings there are for each type (`"humidity"`, `"temperature"`, `"P2"`, and `"P1"`) in site `29`.

- Perform aggregation calculations on documents using PyMongo.

[34]:
```
result = nairobi.aggregate(
    [
        {"$match": {"metadata.site":29}},
        {"$group":{"_id":"$metadata.measurement","count":{"$count": {}}}}
    ]
)
pp.pprint(list(result))
```

```
[ {'_id': 'P2', 'count': 32907},
  {'_id': 'humidity', 'count': 33019},
  {'_id': 'temperature', 'count': 33019},
  {'_id': 'P1', 'count': 32907}]
```

1.3 Import

[35]:
```
VimeoVideo("665412437", h="7a436c7e7e", width=600)
```

[35]:
```
<IPython.lib.display.VimeoVideo at 0x7fb54437f9a0>
```

Task 3.1.16: Use the `find` method to retrieve the PM 2.5 readings from site `29`. Be sure to limit your results to 3 records only. Since we won't need the metadata for our model, use the `projection` argument to limit the results to the `"P2"` and `"timestamp"` keys only.

- Query a collection using PyMongo.

[42]:
```
result = nairobi.find(
    {"metadata.site":29, "metadata.measurement":"P2"},
    projection={"P2":1, "timestamp":1, "_id":0}
)
#pp.pprint(result.next())
```

[39]:
```
VimeoVideo("665412442", h="494636d1ea", width=600)
```

[39]:
```
<IPython.lib.display.VimeoVideo at 0x7fb542b3db80>
```

Task 3.1.17: Read records from your `result` into the DataFrame `df`. Be sure to set the index to `"timestamp"`.

- Create a DataFrame from a dictionary using pandas.

[43]:
```
df = pd.DataFrame(result).set_index("timestamp")
df.head()
```

[43]:

timestamp	P2

```
2018-09-01 00:00:02.472  34.43
2018-09-01 00:05:03.941  30.53
2018-09-01 00:10:04.374  22.80
2018-09-01 00:15:04.245  13.30
2018-09-01 00:20:04.869  16.57
```

[44]:
```
# Check your work
assert df.shape[1] == 1, f"`df` should have only one column, not {df.shape[1]}."
assert df.columns == [
    "P2"
], f"The single column in `df` should be `'P2'`, not {df.columns[0]}."
assert isinstance(df.index, pd.DatetimeIndex), "`df` should have a␣
↪`DatetimeIndex`."
```

Copyright © 2022 WorldQuant University. This content is licensed solely for personal use. Redistribution or publication of this material is strictly prohibited.