Big Data - Basic CRUD Operations

Azhar Chowdhury
Statistical Programmer|Data Scientist[SAS, Python, R]

The following meta summary is helpful to follow the problem at hand and it's solution.

Dataset A generic employee/workplace dataset that qualifies for a big data query.

Domain Standard database - JSON

Operations Standard big data CRUD operations

Components Basic Mongo CRUD operations, Mongo Database pipelines to run essenetial aggregate queries.

Problem Description Applying CRUD operations and NoSQL queries on a dataset

[25]:
```
# Set up environment/import libraries, modules, objects etc.

import json
import datetime
import pymongo
import pandas as pd
import numpy as np
from pymongo import MongoClient
print('Mongo Version: ', pymongo.__version__)
```

```
Mongo Version:  4.8.0
```

Comment: Proir to run the Mongo client object (client) and the Mongo database object (db) we Initiated Mongo Client/Server from DOS prompt (command.exe). After import all the necessary libraries we run mongo server (at dos prompt at the local container folder: mongod.exe) and the mongo client (at dos prompt at the local container folder: mongo.exe). ()Quick note: Windows power shell also performs this.)

[26]:
```
client = MongoClient("mongodb://localhost:27017/")
```

[27]:
```
db = client["database"]
```

[28]:
```
collection = db["myCollection"]
```

[29]:
```
projDatabases = db.list_collection_names()
```

[30]:
```
from IPython.display import display  # Print all databases
display(projDatabases)
```

```
['Address', 'Workplace', 'Employee']
```

[31]:
```
client = MongoClient("mongodb://localhost:27017/")
db = client["database"]
collection = db["myCollection"]

# Check the list of my Collection (Collections contain Documents)
projDatabases = db.list_collection_names() # All databases added to the␣
 ↪collection:
from IPython.display import display  # Print all databases
display(projDatabases)
```

```
['Address', 'Workplace', 'Employee']
```

[ ]:
```
import pandas as pd
with open('C://Users//azhar//Downloads//Address.json') as jFile:
    db.Address.insert_many(json.load(jFile))

with open('C://Users//azhar//Downloads//Workplace.json') as jFile:
    db.Workplace.insert_many(json.load(jFile))

with open('C://Users//azhar//Downloads//Employee.json') as jFile:
    db.Employee.insert_many(json.load(jFile))
```

The above statements yields an error: BulkWriteError: batch op errors occurred, full error: {'writeErrors': [{'index': 0, 'code': 11000, 'errmsg': 'E11000 duplicate key error collection: database.Employee index: *id* dup key: { : "9f39da36-82cc-4353-ab90-d616105fa7c1" }', 'op': {'_id': '9f39da36-82cc-4353-ab90-d616105fa7c1', 'firstname': 'Emilie', 'lastname': 'Woods', 'age': 40, 'email': 'ih@ri.ro', 'interests': ['Bowling', 'Cooking', 'Golf', 'Swimming'], 'address_id': 'b6c0b50a-d0e3-43bf-a2a4-8d4674c2a7e8', 'workplace_id': 'a32bf18d-e0e5-48f2-a851-aa49c80f9460'}}], 'writeConcernErrors': [], 'nInserted': 0, 'nUpserted': 0, 'nMatched': 0, 'nModified': 0, 'nRemoved': 0, 'upserted': []}

Comment on the error: A note about the above error (BulkWriteError): The above error occurred due to a duplicate key entry occurred at the input dataset. However, assuming that our input data is clean and hence instead of writing a catch() error and throw an exception I observed that sometimes the Python interpreter does not complain about the duplicate key. Also the results are obtained. So we moved forward safely with the analysis and did not fix the error in the input file.

[34]:
```
# To see if the first instance of the Employee document is populated/printed
import pprint
cursor = db.Employee.find_one() # first data point (record)
pprint.pp(cursor)
```

```
{'_id': '9f39da36-82cc-4353-ab90-d616105fa7c1',
 'firstname': 'Emilie',
 'lastname': 'Woods',
 'age': 40,
 'email': 'ih@ri.ro',
 'interests': ['Bowling', 'Cooking', 'Golf', 'Swimming'],
```

```
 'address_id': 'b6c0b50a-d0e3-43bf-a2a4-8d4674c2a7e8',
 'workplace_id': 'a32bf18d-e0e5-48f2-a851-aa49c80f9460'}
```

[35]:
```
# To see if the first instance of the Address document is populated/printed
import pprint
addr = db.Address.find_one()
pprint.pp(addr)
```

```
{'_id': '91b5b7b3-2309-4e8a-8247-cd66d626ef0c',
 'address': '573 Wojhas Square',
 'postalcode': 'A7D 5A3',
 'city': 'Victoria',
 'province': 'BC'}
```

[36]:
```
# To see if the first instance of the Workplace document is populated/printed
office = db.Workplace.find_one()
pprint.pp(office)
```

```
{'_id': '5345fcb9-6297-4b9f-aa15-cbee8460f28f',
 'name': 'Union Planters Corp',
 'website': 'http://www.unionplanternscorp.com',
 'industry': 'Aerospace',
 'address_id': '9949fe3b-99ec-4485-b91d-823925db7d28'}
```

At this point - the MongoDB is set up successfully and is ready to run queries or any other operations.

[37]:
```
# Query #1:
# In this query we'll find all the employees who meets the criteria - Employees␣
 ↪that are less than or equal to 50 and like cooking.

import pandas as pd
cursor = db.Employee.find({"age": {"$lte": 50}, "interests": {"$eq":␣
 ↪"Cooking"}}, {"firstname", "lastname", "age", "email", "interests"})
emps=pd.DataFrame(list(cursor))
emps
```

[37]:

	_id	firstname	lastname	age	\
0	9f39da36-82cc-4353-ab90-d616105fa7c1	Emilie	Woods	40	
1	af27265e-6639-49f2-991e-193275a4111a	Thomas	Patterson	18	
2	00289d48-bad8-4b73-a359-a1a1f05c96e2	Sophia	Flores	22	
3	da76e52b-b3db-4fc0-b0d6-435d1aed0cd9	Ollie	Barnett	25	
4	51643cd6-49bb-45d5-bd6e-717c62bb2869	James	Wilkins	27	
5	f073a705-6546-4375-adb5-b224871776ef	Aaron	Carr	25	
6	457ef68c-9651-4925-bca0-15e246661d19	Alta	Sharp	34	
7	840184a3-4c4d-4b15-8813-30fca6e7827b	Delia	Douglas	36	
8	6157dc3b-ee2d-463a-b62b-1cd4eed7d575	Dominic	Wade	48	
9	cc3e389d-be0d-467a-ba70-7c84f6504911	Myrtle	Little	36	
10	c2bf0e3f-e3de-41de-b745-c5b70571dd3a	Jordan	Roberson	29	
11	4bc070ca-f849-4eeb-8ab2-98fe3c0861c0	Francis	Harris	38	

```
12  66894dba-4ff0-4545-b7cc-eb6a5bd551c5     Clara     Butler   49
13  581f59a2-ff4c-407e-b58f-e4f292208928      Rena    Johnson   33
14  b2a2ae86-f4f6-4da5-ade6-ee1cd135daf3     Gavin     Conner   49
15  2bfc0479-cb15-46e1-94fa-801404791b21     Peter    Jimenez   37
```

	email	interests
0	ih@ri.ro	[Bowling, Cooking, Golf, Swimming]
1	sug@gon.bf	[Cooking, Cricket, Tennis, Swimming, Fishing]
2	ra@dupnejuk.nr	[Hiking, Soccer, Bowling, Rubgy, Cooking, Danc...
3	ro@nemaw.et	[Cooking, Bowling, Dancing]
4	hutfardu@vicbiri.gb	[Rubgy, Tennis, Cricket, Cooking]
5	fekegim@lucul.tp	[Cooking]
6	jus@goal.bn	[Cricket, Cycling, Rubgy, Golf, Cooking, Dancing]
7	me@wak.ne	[Cricket, Cooking, Hiking, Dancing, Tennis]
8	co@pog.nz	[Cycling, Dancing, Cooking]
9	saj@far.zm	[Cooking, Cycling, Hiking, Rubgy, Bowling, Dan...
10	co@mahdo.ca	[Swimming, Cooking]
11	pa@sodej.ck	[Dancing, Swimming, Cooking, Bowling, Cycling]
12	do@womjip.by	[Soccer, Cooking, Cycling]
13	vasberet@his.nz	[Cycling, Swimming, Soccer, Cooking]
14	ah@jopah.uz	[Rubgy, Cooking, Bowling]
15	usope@tega.bh	[Cooking, Swimming, Cricket]

[38]:
```
# #2: Mongo CRUD operation
# Add an employee to the database:

js = db.Employee.insert_one({
    "firstname": "Jake",
    "lastname": "Sample",
    "email": "jakesample@email.com",
    "interests": ["Biking", "Hiking"] ,
    "Workplace": {"_id": "5345fcb9-6297-4b9f-aa15-cbee8460f28f"},
    "Address": {"_id": "91b5b7b3-2309-4e8a-8247-cd66d626ef0c"}

})
```

The above operation run without error message. We'd like to see this document (Jake Sample) in the collection (Employee document).

[39]:
```
# Verify the addition of this employee in the document upon completion of Mongo␣
 ↪CRUD operation:
import pprint
newEmp = db.Employee.find_one({"firstname": "Jake"} )
pprint.pp(newEmp) # prints correctly and adds the document to the collection.
```

```
{'_id': ObjectId('67bf51b10421a30f8769bd05'),
 'firstname': 'Jake',
 'lastname': 'Sample',
```

```
 'email': 'jakesample@email.com',
 'interests': 'Biking, Hiking'}
```

[40]:
```
# #3: Mongo CRUD operation
# Deleting an employee based on (multiple) Query: We'll delete all employees tht␣
 ↪work for 'Great Plains Energy Inc.' and are greater than 46 years old and␣
 ↪likes 'Tennis'. We'll also didplay the number of employees deleted.

qEmps = {  # build a query
    "age": {"$gt": 46},  # Query 1: age>46
    "interests": "Tennis",  # Query 2: interests==Tennis
    "workplace_id": "5345fcb9-6297-4b9f-aa15-cbee8460f28f" # Query 3: Works for␣
 ↪Great Plains Energy Inc
}
qEmpList = db.Employee.delete_many(qEmps)
pprint.pp(qEmpList)

# Display Number of Employees Deleted:
print("Number of Employees Deleted: "  + str(qEmpList.deleted_count))
```

```
DeleteResult({'n': 0, 'ok': 1.0}, acknowledged=True)
Number of Employees Deleted: 0
```

Comment: I noticed that the query ran without any error message. The number of deleted records that the query found to be 0. The possible explanation would be there is no record for which an employee is over 46 years old and has an interest of Tennis. Hence the output of 0 records deleted.

[43]:
```
# #4: Mongo CRUD operation
# update a document
# We'll add a new field named 'Industry' in the employee coleection, to all␣
 ↪employees that work for 'Health Net Inc.' and populates the field with the␣
 ↪value 'Health Care'

newIndustry = db.Employee.update_many({"Workplace_id":␣
 ↪'1ed298fc-20ab-4750-ac38-fed1e60964af'},
                                       {"$set":{"industry": "Health Care"}})
#newIndustry.acknowledged
#ni = db.Employee.update_many({"Workplace": {'_id':␣
 ↪'1ed298fc-20ab-4750-ac38-fed1e60964af'}}, {"$set":{"industry": "Health␣
 ↪Care"}})

print("Number of Modification: " + str(newIndustry.modified_count))
```

```
Number of Modification: 0
```

Comment: For this question, I am not very confident that I was able to set the qurey - {“Workplace_id”: ‘1ed298fc-20ab-4750-ac38-fed1e60964af’}. The reason I was not able to see all the keys (dataset - field/variable) of the Employee document. I had to depend on the example of the Employee document supplied in the cell #4 in the Assignment 2 notebook.

[42]:
```
# Question 5:

pipeline= [
    { "$group": {
        "_id": "$workplace_id",
        "emp count": {"$sum": 1}
     }
    }
]
#qr = list(db.Employee.aggregate(pipeline))
import pprint
qry = list(db.Employee.aggregate(pipeline))
pprint.pp(qry)
```

```
[{'_id': None, 'emp count': 31},
 {'_id': '2b87eb84-e5b4-4f2c-9e13-dc3ba20a7f7f', 'emp count': 13},
 {'_id': '023acbac-b32a-477f-a65a-db11bd7d40f3', 'emp count': 10},
 {'_id': 'cb795afb-8dc3-482f-b3a0-14229a280afa', 'emp count': 15},
 {'_id': 'b12cd444-e65b-4bc2-8cf6-2dbe854a627b', 'emp count': 7},
 {'_id': '50275ad1-8140-4e79-8818-21793e3eb0a3', 'emp count': 6},
 {'_id': 'a222385c-342c-43ea-adbc-9b487a2ee2be', 'emp count': 14},
 {'_id': 'b2a2844d-aeab-4602-b74c-01bf3b8e9c78', 'emp count': 11},
 {'_id': 'b2903a4b-5688-4597-90ed-0f06d944bb6d', 'emp count': 6},
 {'_id': '5345fcb9-6297-4b9f-aa15-cbee8460f28f', 'emp count': 7},
 {'_id': 'a32bf18d-e0e5-48f2-a851-aa49c80f9460', 'emp count': 9}]
```

Comment: The output shows the Workplace ids that popuplates in the Employee document. I was not able to populate the name of the workplace. eg. corresponding to the '_id': '5345fcb9-6297-4b9f-aa15-cbee8460f28f' the name of the workplace: 'Union Planters Corp'.