Assignment 1: MongoDB

Negin Baghbanzadeh

This assignment is based on content discussed in Module 2: Introduction to MongoDB

Learning outcomes

The purpose of this assignment is for learners to be able to:

- Familarize with JSON document syntax
- Understand basic MongoDB CRUD operations
- Understand MongoDB data pipelines to run aggregate queries

In this assignment, you will make use of the sample data provided in Module 2.

This dataset has 3 collections: Employee, Workplace and Address. You will import this data into your local MongoDB database.

Required imports for this project are given below.

```
!pip -q install "pymongo[srv]"
```

```
         |████████████████████████████████| 269 kB 5.5 MB/s
```

```
#required imports
import os
import json
import datetime
import pymongo
import pprint
import pandas as pd
import numpy as np
from pymongo import MongoClient
print('Mongo version', pymongo.__version__)
```

```
    Mongo version 4.1.1
```

We first need to connect to MongoDB Atlas Cluster using the connection string. We will use the MongoClient to connect to a local 'test' database that is running on port 27017 (this is the default port).

```
# Find connection string on MongoDB Atlas and
client = pymongo.MongoClient("mongodb+srv://neg:sep@cluster0.19zay.mongodb.net/?retryW
db = client.assignment1
db
```

```
    Database(MongoClient(host=['cluster0-shard-00-02.19zay.mongodb.net:27017', 'clus
```

After installing necessary modules proceed to import the data into your database. The following lines will download the files into your workspace.

```
# Download JSON datasets to workplace
!wget -q https://raw.githubusercontent.com/tofighi/BigData/main/datasets/work/Address.
!wget -q https://raw.githubusercontent.com/tofighi/BigData/main/datasets/work/Employee
!wget -q https://raw.githubusercontent.com/tofighi/BigData/main/datasets/work/Workplac
```

```
# Let's delete any existing collections in our database
db.workplace.drop()
db.address.drop()
db.employee.drop()

# Import our files into our three collections
with open('Employee.json') as f:
    db.employee.insert_many(json.load(f))
with open('Workplace.json') as f:
    db.workplace.insert_many(json.load(f))
with open('Address.json') as f:
    db.address.insert_many(json.load(f))
```

Question 1 (10 Marks)

The address collection contains employee from different ages and interests. Perform a simple query to list all employees that are less than or equal to 50 and like Cooking.

NOTE: the following shows the structure of an Employee document that will help you construct the query.

```
pprint.pprint(client.assignment1.employee.find_one())
```

```
    {'_id': '9f39da36-82cc-4353-ab90-d616105fa7c1',
     'address_id': 'b6c0b50a-d0e3-43bf-a2a4-8d4674c2a7e8',
     'age': 40,
     'email': 'ih@ri.ro',
     'firstname': 'Emilie',
     'interests': ['Bowling', 'Cooking', 'Golf', 'Swimming'],
      'lastname': 'Woods',
```

```
Q1_cursor = client.assignment1.employee.find({"age": {"$lte": 50}, "interests": {"$in'
pd.DataFrame(list(Q1_cursor))
```

	_id	firstname	lastname	age	email	interests	address_id
0	9f39da36-82cc-4353-ab90-d616105fa7c1	Emilie	Woods	40	ih@ri.ro	[Bowling, Cooking, Golf, Swimming]	b6c0b50a-d0e3-43bf-a2a4-8d4674c2a7e8
1	af27265e-6639-49f2-991e-193275a4111a	Thomas	Patterson	18	sug@gon.bf	[Cooking, Cricket, Tennis, Swimming, Fishing]	64fd714d-e219-4e45-888b-cc2238a8bd0b
2	00289d48-bad8-4b73-a359-a1a1f05c96e2	Sophia	Flores	22	ra@dupnejuk.nr	[Hiking, Soccer, Bowling, Rubgy, Cooking, Danc...	8a430805-00b8-40a6-bd93-c950b544a83b
3	da76e52b-b3db-4fc0-b0d6-435d1aed0cd9	Ollie	Barnett	25	ro@nemaw.et	[Cooking, Bowling, Dancing]	5d3eacc4-d1d8-459b-973c-3bc71feacf50
4	51643cd6-49bb-45d5-bd6e-717c62bb2869	James	Wilkins	27	hutfardu@vicbiri.gb	[Rubgy, Tennis, Cricket, Cooking]	3f10d9f7-57ef-40e1-a97d-d2ee53aa2c6e
5	f073a705-6546-4375-adb5-b224871776ef	Aaron	Carr	25	fekegim@lucul.tp	[Cooking]	f4e59d7e-ea40-442d-87be-106e3c46a554
6	457ef68c-9651-4925-bca0-15e246661d19	Alta	Sharp	34	jus@goal.bn	[Cricket, Cycling, Rubgy, Golf, Cooking, Dancing]	5e97658e-5809-41b8-a088-eddbd81f86a7
7	840184a3-4c4d-4b15-8813-30fca6e7827b	Delia	Douglas	36	me@wak.ne	[Cricket, Cooking, Hiking, Dancing, Tennis]	456e18ae-c2f8-443e-899d-f2b893499695
	6157dc3b-						8d162eb9-

Question 2 (10 Marks)

Insert a new Employee with the following properties:

- First Name: Jake
- Last Name: Sample
- Email: jakesample@email.com
- Age: 26
- Interest: Biking, Hiking

Also, this employee works for 'Union Planters Corp' and lives at '573 Wojhas Square, Victoria'. Verify that the insert succeeded and display the generated employees _id attribute.

HINT An Employee document references a Workplace and Address document

```
# First, we're going to find the address ID and the workplace ID of the given informat
Jake_address = client.assignment1.address.find_one({"address":{"$eq":"573 Wojhas Squar
pprint.pprint(Jake_address)
```

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

```
Jake_workplace = client.assignment1.workplace.find_one({"name":{"$eq":"Union Planters
pprint.pprint(Jake_workplace)
```

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

```
# Now that we have all the attributes that we need to add a new employee, we can call
new_employee = db.employee.insert_one({
      'address_id': Jake_address['_id'],
      'age': 26,
      'email': 'jakesample@email.com',
      'firstname': 'Jake',
      'interests': ['Biking', 'Hiking'],
      'lastname': 'Sample',
      'workplace_id': Jake_workplace['_id']
    })


# Find the employee with the new_employee.inserted_id
pd.DataFrame(list(db.employee.find({"_id": {"$eq": new_employee.inserted_id}})))
```

Question 3 (10 Marks)

Delete all employees that work for 'Great Plains Energy Inc.' and are greater than 46 years old and likes 'Tennis'. Once you delete the employees verify the number of employees deleted.

```
# We need to find the workplace's ID first
GreatPlainsEnergyInc_workplace = client.assignment1.workplace.find_one({"name":{"$eq":
pprint.pprint(GreatPlainsEnergyInc_workplace)
```

```
{'_id': 'a32bf18d-e0e5-48f2-a851-aa49c80f9460',
 'address_id': 'c35b4c7f-b7de-431d-bd60-4026490cd61c',
 'industry': 'Agriculture',
 'name': 'Great Plains Energy Inc.',
 'website': 'http://www.greatplainsenergy.com'}
```

```
delete_result = db.employee.delete_many({"workplace_id" : {"$eq":GreatPlainsEnergyInc_
                                          "age" : {"$gt": 46},
                                          "interests" : {"$in": ["Tennis"]}})
```

```
print("Delete Acknowledged:" + str(delete_result.acknowledged))
print("Delete Count:" + str(delete_result.deleted_count))
```

```
Delete Acknowledged:True
Delete Count:4
```

Question 4 (12 Marks)

Add a new field called 'industry' to all **employees** that work for 'Health Net Inc.'.

HINT All a new field to a document is like updating the document

```
# We need to find the workplace's ID first
HealthNetInc_workplace = client.assignment1.workplace.find_one({"name":{"$eq":"Health
pprint.pprint(HealthNetInc_workplace)
```

```
{'_id': 'a222385c-342c-43ea-adbc-9b487a2ee2be',
 'address_id': '1ed298fc-20ab-4750-ac38-fed1e60964af',
 'industry': 'Construction',
 'name': 'Health Net Inc.',
 'website': 'http://www.healthnetinc.com'}
```

```
# For each emplyee that it's workplace_id is the founded ID, set(add new field) the 'i
update_result = db.employee.update_many(
      { "workplace_id" : HealthNetInc_workplace['_id'] },
      { "$set": { "industry" : "Health Care" } }
);
```

```
print("Update Acknowledged:" + str(update_result.acknowledged))
print("Modified Count:" + str(update_result.modified_count))

    Update Acknowledged:True
    Modified Count:14
```

Question 5 (10 Marks)

Create an aggregate query to count the number of employees for each company and sort the output from largest employee count to lowest employee count.

NOTE you will use a pipeline to achieve the computed result. You should produce a result similar to the following table (the following table contains fake data)

	_id	count
0	[Equity Residential Properties Trust]	19
...	...	...
7	[Bell Microproducts Inc.]	6
8	[Kemet Corp.]	1

HINT you should make use of the \$lookup, \$group and \$sort pipeline operations

```
pipeline = [
    # First we add a new column called workplace to employee, to store the workplace i
    {"$lookup":
     {
        "from": "workplace",
        "localField": "workplace_id",
        "foreignField": "_id",
        "as": "workplace"
     }
    },
    # Group all the employees by their workplace and count the employees in each workp
    {"$group":
      {
        "_id": "$workplace.name",
        "count": {"$sum": 1}
      }
     },
     # Sort the count field descending
    {"$sort":
      {
         "count": -1
      }
    }
]
query_result = list(db.employee.aggregate(pipeline))
pd.DataFrame(list(query_result))
```

	_id	count
0	[Hilton Solutions]	15
1	[Health Net Inc.]	14
2	[Aetna Inc.]	13
3	[Bell Microproducts Inc.]	11
4	[Union Planters Corp]	10
5	[Equity Office Properties Trust]	10
6	[Equity Residential Properties Trust]	7
7	[Kemet Corp.]	6
8	[Xcel Bear Inc]	6
9	[Great Plains Energy Inc.]	5