

MongoDB

✓ Learning outcomes

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

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]"
```

```
#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 4.6.3
```

We first need to connect to MongoDB Atlas Cluster using the connection string.

```
# Find connection string on MongoDB Atlas and
uri = "mongodb+srv://yasharkelasi1981:tmu123@cluster0.ze8liic.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0"
client = pymongo.MongoClient(uri) # Replace the connection string here between ""
db = client.assignment1

Database(MongoClient(host=['ac-7jxnftq-shard-00-01.ze8liic.mongodb.net:27017', 'ac-7jxnftq-shard-00-02.ze8liic.mongodb.net:27017', 'ac-7jxnftq-shard-00-00.ze8liic.mongodb.net:27017'], document_class=dict, tz_aware=False, connect=True, retrywrites=True, w='majority', appname='Cluster0', authsource='admin', replicaset='atlas-10wg1f-shard-0', tls=True), 'assignment1')
```

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.json
!wget -q https://raw.githubusercontent.com/tofighi/BigData/main/datasets/work/Employee.json
```

```
!wget -q https://raw.githubusercontent.com/tofighi/BigData/main/datasets/work/Workplace.json
```

```
# 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

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',
 'workplace_id': 'a32bf18d-e0e5-48f2-a851-aa49c80f9460'}
```

prompt: 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 Cookir

```
# Find all employees who are less than or equal to 50 and like Cooking.
query = {'age': {'$lte': 50}, 'interests': 'Cooking'}
results = db.employee.find(query)
```

```
# Print the results.
for result in results:
    pprint.pprint(result)
```

```

    age : 29,
    'email': 'co@mahdo.ca',
    'firstname': 'Jordan',
    'interests': ['Swimming', 'Cooking'],
    'lastname': 'Roberson',
    'workplace_id': 'cb795afb-8dc3-482f-b3a0-14229a280afa'}
{'_id': '4bc070ca-f849-4eeb-8ab2-98fe3c0861c0',
 'address_id': 'fc607f68-81c8-4ca8-8a9e-30239ccae40f',
 'age': 38,
 'email': 'pa@sodej.ck',
 'firstname': 'Francis',
 'interests': ['Dancing', 'Swimming', 'Cooking', 'Bowling', 'Cycling'],
 'lastname': 'Harris',
 'workplace_id': '2b87eb84-e5b4-4f2c-9e13-dc3ba20a7f7f'}
{'_id': '66894dba-4ff0-4545-b7cc-eb6a5bd551c5',
 'address_id': '4dcebe6e-0787-4158-9744-3d721098cec4',
 'age': 49,
 'email': 'do@womjip.by',
 'firstname': 'Clara',
 'interests': ['Soccer', 'Cooking', 'Cycling'],
 'lastname': 'Butler',
 'workplace_id': '50275ad1-8140-4e79-8818-21793e3eb0a3'}
{'_id': '581f59a2-ff4c-407e-b58f-e4f292208928',
 'address_id': '1cf24906-c700-41bb-a4c8-bba3c3ae30d9',
 'age': 33,
 'email': 'vasberet@his.nz',
 'firstname': 'Rena',
 'interests': ['Cycling', 'Swimming', 'Soccer', 'Cooking'],
 'lastname': 'Johnson',
 'workplace_id': '2b87eb84-e5b4-4f2c-9e13-dc3ba20a7f7f'}
{'_id': 'b2a2ae86-f4f6-4da5-ade6-ee1cd135daf3',
 'address_id': 'ef5a17a2-6f11-49b8-a0fd-e0bddc56f185',
 'age': 49,
 'email': 'ah@jopah.uz',
 'firstname': 'Gavin',
 'interests': ['Rubgy', 'Cooking', 'Bowling'],
 'lastname': 'Conner',
 'workplace_id': '50275ad1-8140-4e79-8818-21793e3eb0a3'}
{'_id': '2bfc0479-cb15-46e1-94fa-801404791b21',
 'address_id': '77ce0c50-afd6-47e8-86d7-b1b372e8deb2',
 'age': 37,
 'email': 'usope@tega.bh',
 'firstname': 'Peter',
 'interests': ['Cooking', 'Swimming', 'Cricket'],
 'lastname': 'Jimenez',
 'workplace_id': 'cb795afb-8dc3-482f-b3a0-14229a280afa'}

```

▼ Question 2

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

```

# Assume this finds an existing address suitable for Jake Sample
existing_address = db.address.find_one()
address_id = existing_address['_id']

new_address = {
    # Assume this is the structure for addresses in your collection
    'street': '573 Wojhas Square',
    'city': 'Victoria',
    # Include any other fields required by your Address collection schema
}

# Insert the new address
address_insert_result = db.address.insert_one(new_address)
# Retrieve the new address ID
address_id = address_insert_result.inserted_id

workplace_id = "5345fcb9-6297-4b9f-aa15-cbee8460f28f" # Directly from your provided data

new_employee = {
    'firstname': 'Jake',
    'lastname': 'Sample',
    'email': 'jakesample@email.com',
    'age': 26,
    'interests': ['Biking', 'Hiking'],
    'workplace_id': workplace_id,
    'address_id': address_id,
}

# Insert the new employee
insert_result = db.employee.insert_one(new_employee)
print('New Employee ID:', insert_result.inserted_id)

New Employee ID: 6612db93ecaec8f2da365366

# Convert the string ID to a format that MongoDB understands if necessary
from bson.objectid import ObjectId

# Assuming '6612db93ecaec8f2da365366' is the string representation of the ObjectId
employee_id = ObjectId('6612db93ecaec8f2da365366')

# Retrieve the inserted employee's information
inserted_employee = db.employee.find_one({'_id': employee_id})

# Pretty print the inserted employee's information
import pprint
pprint.pprint(inserted_employee)

{'_id': ObjectId('6612db93ecaec8f2da365366'),
 'address_id': ObjectId('6612db71ecaec8f2da365365'),

```

```

'age': 26,
'email': 'jakesample@email.com',
'firstname': 'Jake',
'interests': ['Biking', 'Hiking'],
'lastname': 'Sample',
'workplace_id': '5345fcb9-6297-4b9f-aa15-cbee8460f28f'}

```

Question 3

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.

```

workplace_id = db.workplace.find_one({'name': 'Great Plains Energy Inc.'})['_id']

delete_result = db.employee.delete_many({
    'workplace_id': workplace_id,
    'age': {'$gt': 46},
    'interests': 'Tennis'
})

print('Number of employees deleted:', delete_result.deleted_count)

```

```

Number of employees deleted: 4

```

Question 4

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

```

workplace_id = db.workplace.find_one({'name': 'Health Net Inc.'})['_id']

update_result = db.employee.update_many(
    {'workplace_id': workplace_id,
     {'$set': {'industry': 'Your Industry Here'}} # Replace 'Your Industry Here' with the actual industry
    )

print('Number of documents modified:', update_result.modified_count)

```

```

Number of documents modified: 14

```

Question 5

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 = [
    {'$lookup': {
        'from': 'workplace',
        'localField': 'workplace_id',
        'foreignField': '_id',
        'as': 'workplace'
    }},
    {'$unwind': '$workplace'},
    {'$group': {
        '_id': '$workplace.name',
        'count': {'$sum': 1}
    }},
    {'$sort': {'count': -1}}
]

query_result = 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	

