Task 2 MongoDB

One advantage of NoSQL databases is that they are more flexible in the way in which they can hold data, fields with the same name are not constricted by data types as they are in SQL databases. When creating and populating the database in a table in a relational format, I used two tables, one for admissions and one for patients. When looking at the dataset and how to approach its implementation in a NoSQL format, whilst MongoDB has the ability to create something similar to a table called a 'collection', I am deciding to use a single collected and embed an appointments array as part of a patient document.

I first needed to design the way in which each document in my database will follow, after designing that, it is necessary to change the data provided so that it aligns with this design, I then need to populate the database. The data design I am going to use incorporates the use of embedded documents in order to store the admissions data inside the patients document.

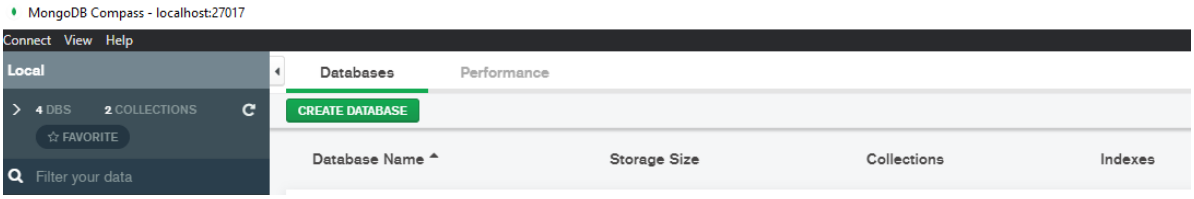
Here is the Document structure I am following for my design with an example from my dataset:

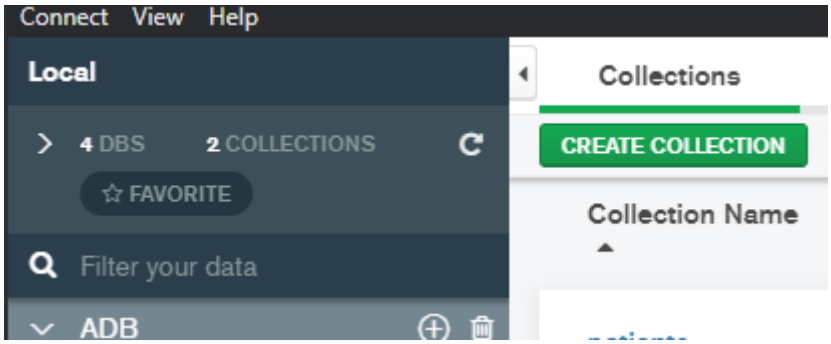
```
{
    "_ID" : Generated by MongoDB automatically
    "name": "ABC008",
    "dob": "1938-06-11",
    "sex": "F",
    "address":[{"street_address": "233 Oldstreet Road",
                "city": "Bristol",
                "postcode": "BS78 9YT",
                "telephone": "0117 9292929"
              }],
    "admissions": [{"admissionDate": ISODate('2020-11-21T00:00:00.000+00:00'),
                    "height(m)": 1.59,
                    "weight(kg)": 50,
                    "BMI": 19.78,
                    "bodyType": "Regular",
                    "smoker": "N",
                    "asthmatic": "N",
                    "njt_ngr": "Y",
                    "hypertension": "N",
                    "renalRT": "Y",
                    "ileostomyColostomy": "N",
                    "parenteralNutrition": "N"
                    "referralDecision": "N"
                  }],
}
```

I was contemplating the use of multiple collections, one for patients and one for admissions. However I decided against it as I had a concern about potentially needing a patientID field in the admissions collection in order to assist with the matching, this didn't feel very practical and was actively going against once of the advantage of NoSQL databases which is the flexibility to store data in a variety of formats. With the embedded document structure, there is no need to create another document for a new admission of a patient, it can simply be added to the existing admissions array which falls under the specified patient.

After downloading the MongoDB software and setting it up, I opened the MongoDB Compass which is their GUI software to better visualise the data I am working with after populating the database, the queries will be ran using command line however.

MongoDB Compass allows for the creation of a database and collections and was the method I used. Though it can be done through the command line using createCollection() and Use "Database name".





Populating the database

Though the MongoDB GUI has the ability to import csv and JSON files directly, the csv file was not in the format that I wanted to store the data in and if imported would just create a large document with no embedded documents inside. I could have manually formatted the data to JSON though it would have been a timely inefficient task and thus looked for ways to automated the process. I used python to automate the process making sure to outline the document structure I wanted to use. As the dates in the csv file was not in the format Mongo uses and when importing numerical values they entered the database as strings, I needed to use some functions to convert the data to the correct format I wanted.

Python File

```
CSV To Mongo - Francis Denton
Advanced Databases

Required Imports
'''
import datetime
from pymongo import MongoClient
import csv

#Connection to MongoDB Client and Specified Database
client = MongoClient(port=27017)
db = client["ADB"]

#Function to convert date to mongo format DD/MM/YYYY -> YYYY-MM-DD
def date_conversion(date):
    date = date.split('/')
    newDate = datetime.datetime.strptime(
        "{0}-{1}-{2}".format(date[2], date[1], date[0]), "%Y-%m-%d")
    return newDate

#Function to convert Numerical Values taken form csv from string to Floats (BMI, Height, Weight)
def string_to_float(value):
    Float = float(value)
    print(Float)
    return Float
```

```
def csv_to_mongo():
    #Patients Collection
    patients = db["patients"]
    with open("data.csv", 'r') as csvfile:
        line = 0
        datareader = csv.reader(csvfile)
        documents = []
        for row in datareader:
            if line > 0:
                # Construct document for the current patient
                patient_in_doc = False
                if len(documents) > 0:
                    for doc in documents:
                        if doc["name"] == row[0]:
                            patient_in_doc = True
                if not patient_in_doc:
                    doc = {
                        "name": row[0],
                        "dob": date_conversion(row[1]),
                        "sex": row[2],
                        "address": {
                            "street_address": row[3],
                            "city": row[4],
                            "postcode": row[5],
                            "phoneNo": row[6],
                        },
                        "admissions": [{
                            "admissionDate": date_conversion(row[7]),
                            "height(m)": string_to_float(row[8]),
                            "weight(kg)": string_to_float(row[9]),
                            "BMI": string_to_float(row[10]),
                            "bodyType": row[11],
                            "smoker": row[12],
                            "asthmatic": row[13],
                            "njt_ngr": row[14],
                            "hypertension": row[15],
                            "rentalRT": row[16],
                            "ileostomyColostomy": row[17],
                            "parentalNeutrition": row[18],
                            "referralDecision": row[19],
                        }]
                    }
                    documents.append(doc)
                else:
                    doc["admissions"].append({
                        "admissionDate": date_conversion(row[7]),
                        "height(m)": string_to_float(row[8]),
                        "weight(kg)": string_to_float(row[9]),
                        "BMI": string_to_float(row[10]),
                        "bodyType": row[11],
                        "smoker": row[12],
                        "asthmatic": row[13],
                        "njt_ngr": row[14],
                        "hypertension": row[15],
                        "rentalRT": row[16],
                        "ileostomyColostomy": row[17],
                        "parentalNeutrition": row[18],
                        "referralDecision": row[19],
                    })
            line += 1
        #Insert data into collection
        patients.insert_many(documents)
        print(documents)

csv_to_mongo()
```

Portion of Database after Implementation

```
_id: ObjectId("607f5e9526e759e892a5032b")
name: "ABD111"
dob: 1949-12-22T00:00:00.000+00:00
sex: "M"
> address: Object
> admissions: Array

_id: ObjectId("607f5e9526e759e892a5032c")
name: "ABC234"
dob: 2002-12-11T00:00:00.000+00:00
sex: "M"
> address: Object
> admissions: Array

_id: ObjectId("607f5e9526e759e892a5032d")
name: "ABD221"
dob: 1972-03-22T00:00:00.000+00:00
sex: "F"
> address: Object
> admissions: Array

_id: ObjectId("607f5e9526e759e892a5032e")
name: "ABD176"
dob: 1967-03-25T00:00:00.000+00:00
sex: "M"
> address: Object
> admissions: Array
```

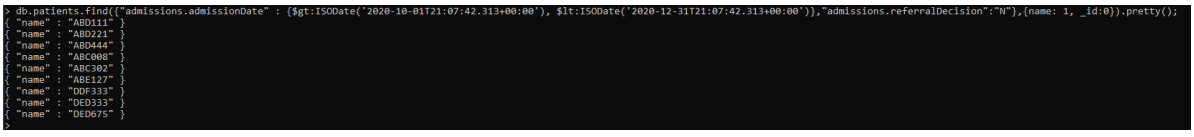

Queries

Query 1

List all the Patients that were admitted between the months of October to December 2020 (inclusive) and were not referred to a dietitian

```
db.patients.find({"admissions.admissionDate" : {$gt:ISODate('2020-10-01T21:07:42.313+00:00'), $lt:ISODate('2020-12-31T21:07:42.313+00:00')},"admissions.referralDecision":"N"},{name: 1, _id:0}).pretty();
```

Output



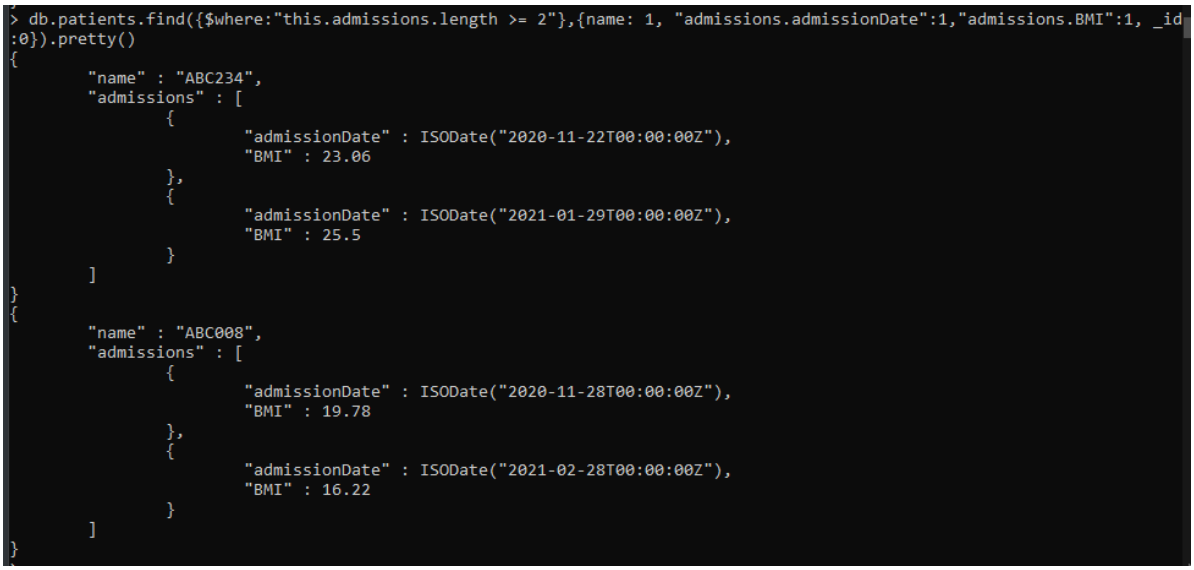
Query 2

List all the patients that have had at least two admissions to the CCU and between admissions have had a change in their BMI.

```
db.patients.find({$where:"this.admissions.length >= 2"},{name: 1, "admissions.admissionDate":1,"admissions.BMI":1, _id:0}).pretty()

'Half of the query has been implemented - Listed patients that had at least 2 admissions, but did not find patients who also had change in BMI'
```

Output

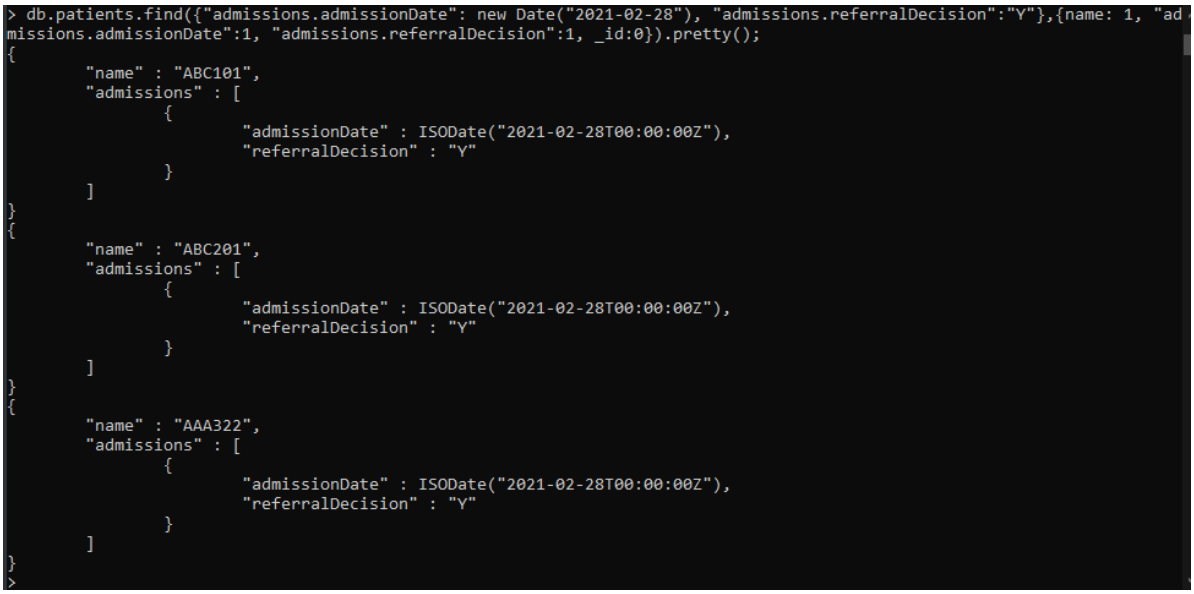


Query 3

List the patients that have been admitted in the CCU on 28/02/2021 and were referred to a dietitian in the order of priority of referral

```
db.patients.find({"admissions.admissionDate": new Date("2021-02-28"),
"admissions.referralDecision":"Y"},{name: 1, "admissions.admissionDate":1,
"admissions.referralDecision":1, _id:0}).pretty();

'Half of the query has been completed, the list of patients who were admitted to the CCU on the 28/02/2021 and were referred to a dietitian has been found. Though the sorting by order of priority has not been implemented'
```



Query 4

List all the patients that should have been referred to a dietitian, but were missed out by the system.

```
db.patients.find({$and:[{$or:[ {"admissions.BMI": {$lt: 18.5}},
{"admissions.bodyType" : "Regular" ,"admissions.BMI":{"$gte": 29}},
{"admissions.bodyType" : "Slim" , "admissions.BMI":{"$gte": 28}},
{"admissions.bodyType" : "Athletic" , "admissions.BMI":{"$gte": 30}},
{"admissions.hypertension":"Y"},{"admissions.asthmatic":"Y"},
{"admissions.smoker":"Y"},{"admissions.njt_ngr":"Y"}, {"admissions.renalRT":
"Y"}, {"admissions.ileostomyColostomy":"Y"},
{"admissions.parentalNeutrition":"Y"}]},{"admissions.referralDecision" : "N"}]},
{name: 1,_id:0}).pretty()
```

Output

List all the patients that should have been referred to a dietitian, but were missed out by the system.

Query 5

List all the patients that have slim build, normal weight condition, and were referred to a dietitian, in Descending order according to their age.

```
db.patients.find({$and: [{"admissions.bodyType":"Slim", "admissions.BMI":
{"$gte": 18.5, "$lte":25}, "admissions.referralDecision":"Y"}]},{name: 1,"dob":
1, _id:0}).sort({"dob":1}).pretty()
```

Output