CST4050 - MongoDB Course Work

Group members

1) Alex Stuart - M00908955

2) Mykhailo Kaptyelov - M00915847

3) Nitin Satish - M00909311

4) Rajkumar Selvaraj - M00910461

In [1]:
```
# Importing libraries
import pymongo
from bson import json_util
```

Task2

In [2]:
```
# Connecting to mongodb atlas cluster

from pymongo.mongo_client import MongoClient
from pymongo.server_api import ServerApi
uri = "mongodb+srv://mongo:mongo@cluster0.pczztvv.mongodb.net/?retryWrites=true&w=majority"

# Create a new client and connect to the server
client = MongoClient(uri, server_api=ServerApi('1'))

# Send a ping to confirm a successful connection
try:
    client.admin.command('ping')
    print("Pinged your deployment. You successfully connected to MongoDB!")
except Exception as e:
    print(e)
```

```
Pinged your deployment. You successfully connected to MongoDB!
```

In [3]:
```
# Creating a variable named db to to point to the database named Restaurants created
# Client is the handle to our mongodb cluster

db = client.Restaurants
```

In [4]:
```
# Creating collection called Justeat by the pointer named eat

eat = db.Justeat
```

In [5]:
```
# Retrieving the contents from the json file

import json
file = open('restaurants.json')
docs = json.load(file)
```

In [6]:
```
# Uploading this json file contents to our collection

eat.insert_many(docs)
```

Out[6]:
```
<pymongo.results.InsertManyResult at 0x1ecfb99aa90>
```

In [7]:
```
# Counting number of documents
c = eat.count_documents({})
print('The number of records present in the collection is ',c)
```

```
The number of records present in the collection is  34
```

Task 3

In [8]:
```
# Python code format
# We run a loop through the documents in the collection to validate the cuisine and the address zipcode.
# We then store each retrieved value in a dictionary
items = eat.find()
d = {}

for i in items:
    if(i['cuisine']=='Chicken' and i['address']['zipcode']=='10024'):
        d['name'] = i['name']
        break
print(d)
```

```
{'name': "Harriet'S Kitchen"}
```

In [9]:
```
# Mongo code format
eat.find_one({'cuisine':'Chicken', 'address.zipcode':'10024'}, {'name':1, '_id':0})
```

Out[9]:
```
{'name': "Harriet'S Kitchen"}
```

Task 4

In [10]:
```
# Listing the boroughs and the count of grades which are A
list(eat.aggregate([{'$unwind':'$grades'}
                    ,{ '$match' : {'grades.grade':'A'} },
                    { '$group' : { '_id' : "$borough" , 'count' : { '$sum' : 1 } } }]))
```

Out[10]:
```
[{'_id': 'Manhattan', 'count': 38},
 {'_id': 'Bronx', 'count': 12},
 {'_id': 'Queens', 'count': 23},
 {'_id': 'Staten Island', 'count': 4},
 {'_id': 'Brooklyn', 'count': 58}]
```

Task 5

In [11]:
```
# Map reduce task has been carried using the pipeline technique
# The $group section groups the cuisine types for each borough
# The $sort section makes the output of the boroughs in alphabetical order
# The $project creates a count figure starting from 0, and increments everytime the corresponding cuisine in
                                                                                #the borough is found
pipeline = [
    {
        '$group': {
            '_id': {'borough': '$borough', 'cuisine': '$cuisine'},
            'count': {'$sum': 1}
        }
    },
    {
        '$sort': {'_id.borough': 1, 'count': -1}
    },
    {
        '$project': {
            '_id': 0,
            'Borough name': '$_id.borough',
            'Cuisine type': '$_id.cuisine',
            'Count': '$count'
        }
    }
]

result = eat.aggregate(pipeline)

for doc in result:
    print(doc)
```

```
{'Borough name': 'Bronx', 'Cuisine type': 'American ', 'Count': 1}
{'Borough name': 'Bronx', 'Cuisine type': 'Ice Cream, Gelato, Yogurt, Ices', 'Count': 1}
{'Borough name': 'Bronx', 'Cuisine type': 'Bakery', 'Count': 1}
{'Borough name': 'Brooklyn', 'Cuisine type': 'American ', 'Count': 5}
{'Borough name': 'Brooklyn', 'Cuisine type': 'Hamburgers', 'Count': 2}
{'Borough name': 'Brooklyn', 'Cuisine type': 'Ice Cream, Gelato, Yogurt, Ices', 'Count': 2}
{'Borough name': 'Brooklyn', 'Cuisine type': 'Delicatessen', 'Count': 2}
{'Borough name': 'Brooklyn', 'Cuisine type': 'Chinese', 'Count': 1}
{'Borough name': 'Brooklyn', 'Cuisine type': 'Caribbean', 'Count': 1}
{'Borough name': 'Brooklyn', 'Cuisine type': 'Jewish/Kosher', 'Count': 1}
{'Borough name': 'Brooklyn', 'Cuisine type': 'Donuts', 'Count': 1}
{'Borough name': 'Manhattan', 'Cuisine type': 'American ', 'Count': 5}
{'Borough name': 'Manhattan', 'Cuisine type': 'Turkish', 'Count': 1}
{'Borough name': 'Manhattan', 'Cuisine type': 'Chicken', 'Count': 1}
{'Borough name': 'Manhattan', 'Cuisine type': 'Irish', 'Count': 1}
{'Borough name': 'Manhattan', 'Cuisine type': 'Delicatessen', 'Count': 1}
{'Borough name': 'Queens', 'Cuisine type': 'Delicatessen', 'Count': 2}
{'Borough name': 'Queens', 'Cuisine type': 'American ', 'Count': 1}
{'Borough name': 'Queens', 'Cuisine type': 'Chinese', 'Count': 1}
{'Borough name': 'Queens', 'Cuisine type': 'Jewish/Kosher', 'Count': 1}
{'Borough name': 'Queens', 'Cuisine type': 'Ice Cream, Gelato, Yogurt, Ices', 'Count': 1}
{'Borough name': 'Staten Island', 'Cuisine type': 'Jewish/Kosher', 'Count': 1}
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: