Name: Chakrya Ros
Homework 6

Task 1: Basic operations in MongoDB

1. Create a database:

> use new_mongo_db

```
> use new_mongo_db
switched to db new_mongo_db
```

2. Drop a database:

> db.dropDatabase()

```
> db.dropDatabase()
{ "dropped" : "new_mongo_db", "ok" : 1 }
```

3. Creating a collection:

```
> db.createCollection("test_collection")
```

```
> db.createCollection("mycol", { capped : true, autoIndexId : true, size : 6142800, max : 10000})
```

```
> db.createCollection("test_collection")
{ "ok" : 1 }
> db.createCollection("mycol", { capped : true, autoIndexId : true, size : 6142800, max : 10000})
{
        "note" : "the autoIndexId option is deprecated and will be removed in a future release",
        "ok" : 1
}
```

4. Dropping a collection:

> db.test_collection.drop()

> db.mycol.drop()

```
> db.test_collection.drop()
true
> db.mycol.drop()
true
```

5. Insert a document:

> db.test_collection.insert({title: "Mongo Db practice", description: "This is my first MongoDB document"})

> db.mycollection.insert({title: "Mongo Db practice", description: "This is my first MongoDB document"})

```
> db.test_collection.insert({title: "Mongo Db practice", description: "This is my first MongoDB document"})
WriteResult({ "nInserted" : 1 })
> db.mycollection.insert({title: "Mongo Db practice", description: "This is my first MongoDB document"})
WriteResult({ "nInserted" : 1 })
```

6. Query a document:

> db.mycollection.find().pretty()

```
> db.test_collection.find().pretty()
{
        "_id" : ObjectId("5cb93b6c0eb979510d6879e6"),
        "title" : "Mongo Db practice",
        "description" : "This is my first MongoDB document"
}
> db.mycollection.find().pretty()
{
        "_id" : ObjectId("5cb93ba30eb979510d6879e7"),
        "title" : "Mongo Db practice",
        "description" : "This is my first MongoDB document"
}
```

7. Update a document:

> db.mycollection.update({"title": "Mongo Db practice"}, {$set: {"title":"New MongoDB practice"}})

```
> db.mycollection.update({"title": "Mongo Db practice"}, {$set: {"title":"New MongoDB practice"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

8. **Delete a Document:**

> db.mycollection.remove({status: "P"},1)

> db.mycollection.remove({status: "P"})

```
> db.mycollection.remove({status: "P"},1)
WriteResult({ "nRemoved" : 0 })
> db.mycollection.remove({status: "P"})
WriteResult({ "nRemoved" : 0 })
```

Task 2: Use real-world data set to answer the following question

(1) List the restaurants that have the string “Ice Cream” in their name. Return only the restaurant id and name. (HINT: use Regex.)

```
> db.restaurants.find({"name":{$regex: /Ice Cream/}},{"restaurant_id"
: 1, "name":1});
```

```
> db.restaurants.find({"name":{$regex: /Ice Cream/}},{"restaurant_id" : 1, "name":1});
{ "_id" : ObjectId("5cb6075204d228ec88ef2b2e"), "name" : "Taste The Tropics Ice Cream", "restaurant_id" : "40356731" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b32"), "name" : "Carvel Ice Cream", "restaurant_id" : "40360076" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b33"), "name" : "Carvel Ice Cream", "restaurant_id" : "40361322" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b41"), "name" : "Carvel Ice Cream", "restaurant_id" : "40363093" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b52"), "name" : "Carvel Ice Cream", "restaurant_id" : "40363834" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2ec2"), "name" : "Egger'S Ice Cream Parlor", "restaurant_id" : "40394644" }
{ "_id" : ObjectId("5cb6075204d228ec88ef3206"), "name" : "Carvel Ice Cream", "restaurant_id" : "40550801" }
{ "_id" : ObjectId("5cb6075204d228ec88ef3442"), "name" : "Carvel Ice Cream", "restaurant_id" : "40639953" }
{ "_id" : ObjectId("5cb6075204d228ec88ef377d"), "name" : "Carvel Ice Cream", "restaurant_id" : "40794321" }
{ "_id" : ObjectId("5cb6075204d228ec88ef37ea"), "name" : "Carvel Ice Cream", "restaurant_id" : "40814300" }
{ "_id" : ObjectId("5cb6075204d228ec88ef3831"), "name" : "Carvel Ice Cream", "restaurant_id" : "40827606" }
{ "_id" : ObjectId("5cb6075204d228ec88ef38aa"), "name" : "Brooklyn Ice Cream Factory", "restaurant_id" : "40853290" }
{ "_id" : ObjectId("5cb6075204d228ec88ef3901"), "name" : "Igloo Ice Cream Cafe", "restaurant_id" : "40865501" }
{ "_id" : ObjectId("5cb6075204d228ec88ef3ba6"), "name" : "Carvel Ice Cream", "restaurant_id" : "40959012" }
{ "_id" : ObjectId("5cb6075204d228ec88ef3bd0"), "name" : "Uncle Louie G'S Italian Ices & Ice Cream", "restaurant_id" : "40965048" }
{ "_id" : ObjectId("5cb6075204d228ec88ef3bfa"), "name" : "Carvel Ice Cream", "restaurant_id" : "40970176" }
{ "_id" : ObjectId("5cb6075204d228ec88ef3c07"), "name" : "Carvel Ice Cream", "restaurant_id" : "40972457" }
{ "_id" : ObjectId("5cb6075204d228ec88ef3c0d"), "name" : "Carvel Ice Cream", "restaurant_id" : "40973500" }
{ "_id" : ObjectId("5cb6075204d228ec88ef3cf2"), "name" : "Subway, Carvel Ice Cream", "restaurant_id" : "41001182" }
{ "_id" : ObjectId("5cb6075204d228ec88ef3f40"), "name" : "Carvel Ice Cream", "restaurant_id" : "41066646" }
```

(2) Find the names of all restaurants that serve either Italian or American cuisine and are located in the Brooklyn borough.

> db.restaurants.find({"borough":"Brooklyn", $or : [{"cuisine" :
"Italian"},{"cuisine" : "American"}]},{"name" : 1});

```
> db.restaurants.find({"borough":"Brooklyn", $or : [{"cuisine":"Italian"},{"cuisine": "American"}]},{"name": 1});
{ "_id" : ObjectId("5cb6075204d228ec88ef2b24"), "name" : "Riviera Caterer" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b2c"), "name" : "Regina Caterers" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b2f"), "name" : "C & C Catering Service" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b37"), "name" : "The Movable Feast" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b44"), "name" : "Mejlander & Mulgannon" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b50"), "name" : "Sonny'S Heros" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b5b"), "name" : "Philadelhia Grille Express" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b6f"), "name" : "Junior'S" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b71"), "name" : "Towne Cafe" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b77"), "name" : "Melody Lanes" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b7c"), "name" : "Shell Lanes" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b88"), "name" : "New Corner" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b90"), "name" : "Palace Cafe" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b94"), "name" : "Reben Luncheonette" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2b9e"), "name" : "Gargiulo'S Restaurant" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2bac"), "name" : "Fifth Avenue Bingo" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2bb4"), "name" : "Three Star Restaurant" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2bb8"), "name" : "Michael'S Restaurant" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2bb9"), "name" : "Roll-N-Roaster" }
{ "_id" : ObjectId("5cb6075204d228ec88ef2bbc"), "name" : "Brennan & Carr" }
```

(3) Return a list of boroughs ranked by the number of American restaurants in the borough. That is, for each borough, find how many restaurants serve American cuisine and print the borough and the number of such restaurants sorted descending by this number. (HINT: use the aggregate method, and use a $group and a $sum.)

```
> db.restaurants.aggregate({"$match" : {"cuisine" :
"American"}},{"$project": {"borough" : 1}}, {"$group": {"_id" :
"$borough", "number" : {$sum : 1}}}, {"$sort": {"number": -1}});
```

```
> db.restaurants.aggregate({"$match":{"cuisine" : "American"}},{"$group": {"_id": "borough","Rank number":{$sum:1}}}, {"$project": {"borough" : 1}}, {"$sort": {"Rank number": -1}})
{ "_id" : "borough" }
> db.restaurants.aggregate({"$match" : {"cuisine" : "American"}},{"$project": {"borough" : 1}}, {"$group": {"_id" : "$borough", "number" : {$sum : 1}}}, {"$sort": {"number": -1}} )
{ "_id" : "Manhattan", "number" : 3205 }
{ "_id" : "Brooklyn", "number" : 1273 }
{ "_id" : "Queens", "number" : 1040 }
{ "_id" : "Bronx", "number" : 411 }
{ "_id" : "Staten Island", "number" : 244 }
{ "_id" : "Missing", "number" : 10 }
```

(4) Find the top 5 American restaurants in Manhattan that have the highest total score. Return for each restaurant the restaurant's name and the total score. (HINT: use the aggregate method with $unwind to parse out the scores array, followed by a $group and a $sum.)

```
db.restaurants.aggregate([{$unwind: "$grades"}, {$match: {cuisine:
"American", borough: "Manhattan"}}, {$group: {_id: "$_id", name:
{$first: "$name"}, totalscore: {$sum: "$grades.score"}}}, {$sort:
{totalscore: -1}}, {$limit: 5}])
```

```
> db.restaurants.aggregate([{$unwind: "$grades"}, {$match: {cuisine: "American", borough: "Manhattan"}}, {$group: {_id: "$_id",
 name: {$first: "$name"}, totalscore: {$sum: "$grades.score"}}}, {$sort: {totalscore: -1}}, {$limit: 5}])
{ "_id" : ObjectId("5cb6075204d228ec88ef4314"), "name" : "Nios Restaurant", "totalscore" : 227 }
{ "_id" : ObjectId("5cb6075204d228ec88ef55b0"), "name" : "Amici 36", "totalscore" : 215 }
{ "_id" : ObjectId("5cb6075204d228ec88ef2c81"), "name" : "Murals On 54/Randolphs'S", "totalscore" : 202 }
{ "_id" : ObjectId("5cb6075204d228ec88ef3581"), "name" : "B.B. Kings", "totalscore" : 199 }
{ "_id" : ObjectId("5cb6075204d228ec88ef4159"), "name" : "Hai Cheng Restaurant", "totalscore" : 193 }
```

(5) Consider the area of the location field identified by the vertices [-74 , 40.5] , [-74 , 40.7] , [-73.5 , 40.5] and [-73.5 , 40.7]. Find the number of restaurants in this area that have received a grade score (at least one) more than 75. No need to sum scores. (Hint: count the restaurants whose location coordinates mathematically fall within the bounds set by the coordinates in the question.)

```
> db.restaurants.find({$and:[{grades: {$elemMatch:{score:{$gt :
75}}}},{"address.coord" :{$geoWithin :{$box:[[-
74,40.7],[73.5,40.5]]}}}]}).count()
```