Title: **Indexing and Advanced Indexing**

Aim: **To create index on collections for simple and geospatial retrieval**

Name: Krish Agarwal

Register Number: 21112016

Class: 4BSc DS

NoSQL Lab 7

03/05/23

SET 1

Importing the data onto MongoDB shell

```
mongostart> use employee
switched to db employee
employee> db.createCollection("restaurant_data")
{ ok: 1 }
employee> db.createCollection("neighborhood_data")
{ ok: 1 }
```

```
C:\Program Files\MongoDB\Tools\100\bin>mongoimport --db employee --collection restaurant_data --type=json --file="D:/Z/Downloads/restaurants.json"
2023-05-03T09:06:45.725+0530    connected to: mongodb://localhost/
2023-05-03T09:06:46.706+0530    25359 document(s) imported successfully. 0 document(s) failed to import.

C:\Program Files\MongoDB\Tools\100\bin>mongoimport --db employee --collection neighborhood_data --type=json --file="D:/Z/Downloads/neighborhoods.json"
2023-05-03T09:07:36.687+0530    connected to: mongodb://localhost/
2023-05-03T09:07:37.019+0530    195 document(s) imported successfully. 0 document(s) failed to import.
```

```
employee> db.restaurant_data.find(
... )
[
  {
    _id: ObjectId("55cba2476c522cafdb053ae1"),
    location: { coordinates: [ -73.8803827, 40.7643124 ], type: 'Point' },
    name: 'Brunos On The Boulevard'
  },
  {
    _id: ObjectId("55cba2476c522cafdb053adf"),
    location: { coordinates: [ -73.98241999999999, 40.579505 ], type: 'Point' },
    name: 'Riviera Caterer'
  },
  {
    _id: ObjectId("55cba2476c522cafdb053ade"),
    location: { coordinates: [ -73.961704, 40.662942 ], type: 'Point' },
    name: "Wendy'S"
  },
  {
    _id: ObjectId("55cba2476c522cafdb053ae5"),
    location: { coordinates: [ -73.9482609, 40.6408271 ], type: 'Point' },
    name: 'Taste The Tropics Ice Cream'
  },
  {
    _id: ObjectId("55cba2476c522cafdb053ae6"),
    location: { coordinates: [ -74.1377286, 40.6119572 ], type: 'Point' },
    name: 'Kosher Island'
  },
```

```
employee> db.neighborhood_data.find()
[
  {
    _id: ObjectId("55cb9c666c522cafdb053a1c"),
    geometry: {
      coordinates: [
        [
          [ -73.89138023380261, 40.86170058826304 ],
          [ -73.89106280613036, 40.86152941211661 ],
          [ -73.89106059904508, 40.86147181520911 ],
          [ -73.89115375546018, 40.86091361616668 ],
          [ -73.89174395026582, 40.86017714781636 ],
          [ -73.8922570762111, 40.85940997630237 ],
          [ -73.89299354227633, 40.85838612046538 ],
          [ -73.8944135507376, 40.85643377580147 ],
          [ -73.89477596003171, 40.8559295256393 ],
          [ -73.89549889768499, 40.854749765891384 ],
          [ -73.89560883346181, 40.85457037009787 ],
          [ -73.89644052566297, 40.85490775915559 ],
          [ -73.89709926839446, 40.855172290942875 ],
          [ -73.89831118626606, 40.8540611510514 ],
          [ -73.89863543744137, 40.8539069754361 ],
          [ -73.89816577520747, 40.85514637639308 ],
          [ -73.89877598610725, 40.8553632470964 ],
          [ -73.89967155800208, 40.8556689083972 ],
          [ -73.90060386049306, 40.855987237865136 ],
          [ -73.90077922747537, 40.856047109729 ],
          [ -73.90085477134976, 40.855937440328404 ],
          [ -73.90116590959869, 40.855495674561254 ],
          [ -73.90130126562461, 40.8553034900000 ],
```

Creating 2dsphere Index in both the collections

Syntax: db.restaurants.createIndex({ "address.coord": "2dsphere" })

db.neighborhoods.createIndex({ geometry: "2dsphere" })

```
employee> db.restaurant_data.createIndex({"address.coord": "2dsphere" })
address.coord_2dsphere
employee> db.neighborhood_data.createIndex({ geometry: "2dsphere" })
geometry_2dsphere
employee> db.restaurant_data.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  {
    v: 2,
    key: { 'address.coord': '2dsphere' },
    name: 'address.coord_2dsphere',
    '2dsphereIndexVersion': 3
  }
]
employee> db.neighborhood_data.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  {
    v: 2,
    key: { geometry: '2dsphere' },
    name: 'geometry_2dsphere',
    '2dsphereIndexVersion': 3
  }
]
```

Q1) Display the details of restaurant from the restaurants collection with location -73.856077, 40.848447.

Syntax: db.restaurant_data.findOne({'location.coordinates': [-73.856077, 40.848447]})

```
employee> db.restaurant_data.findOne({'location.coordinates': [-73.856077, 40.848447]})
{
  _id: ObjectId("55cba2476c522cafdb053add"),
  location: { coordinates: [ -73.856077, 40.848447 ], type: 'Point' },
  name: 'Morris Park Bake Shop'
}
```

Q2) Display the details of restaurant with name “Hell’s Kitchen” from neighbourhood collection.

Syntax: db.neighborhood_data.find({name: "Hell's Kitchen"})

OR

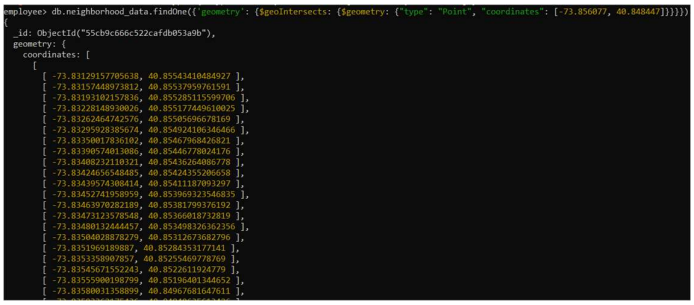
Syntax: db.neighborhood_data.aggregate([{$match: {name: "Hell's Kitchen"}}])

```
employee> db.neighborhood_data.find({name: "Hell's Kitchen"})

employee> db.neighborhood_data.aggregate([{$match: {name: "Hell's Kitchen"}}])
```

Q3) Suppose the user is located at -73.93414657 longitude and 40.82302903 latitude. To find the current neighbourhood using geoIntersects object of geometry object.

Syntax: db.neighborhood_data.findOne({'geometry': {$geoIntersects: {$geometry: {$type: "Point", "coordinates": [-73.856077, 40.848447]}}}})



Q4) Find restaurants within a specified distance of a point, using $geoWithin with $centerSphere to return results in unsorted order, or $nearSphere with $maxDistance in sorted order by distance. Using cursor variables count the restaurants within that neighborhood.

Syntax: var center = [-73.856077, 40.848447]

var radius = 5 / 6378.1

db.restaurant_data.find({"location.coordinates": {$geoWithin: {$centerSphere: [center, radius]}}})

```
employee> var center = [-73.856077, 40.848447]

employee> var radius = 5 / 6378.1

employee> db.restaurant_data.find({"location.coordinates": {$geoWithin: {$centerSphere: [center, radius]}}})
[
  {
    _id: ObjectId("55cba2476c522cafdb053add"),
    location: { coordinates: [ -73.856077, 40.848447 ], type: 'Point' },
    name: 'Morris Park Bake Shop'
  },
  {
    _id: ObjectId("55cba2476c522cafdb053ae7"),
    location: { coordinates: [ -73.8786113, 40.8502883 ], type: 'Point' },
    name: 'Wild Asia'
  },
  {
    _id: ObjectId("55cba2476c522cafdb053afc"),
    location: { coordinates: [ -73.84856870000002, 40.8903781 ], type: 'Point' },
    name: 'Carvel Ice Cream'
  },
  {
    _id: ObjectId("55cba2476c522cafdb053aff"),
    location: { coordinates: [ -73.8893654, 40.81376179999999 ], type: 'Point' },
    name: 'Happy Garden'
  },
  {
    _id: ObjectId("55cba2476c522cafdb053b18"),
    location: {
      coordinates: [ -73.81363999999999, 40.82941100000001 ],
      type: 'Point'
    },
```

Syntax: var center2 = [-73.856077, 40.848447]

db.restaurant_data.find({'location.coordinates': {$nearSphere: center2, $maxDistance: radius}}).sort({distance: 1})

```
employee> db.restaurant_data.find({'location.coordinates': {$nearSphere: center2, $maxDistance: radius}}).sort({distance: 1})
Uncaught:
MongoServerError: error processing query: ns=employee.restaurant_dataTree: GEONEAR  field=location.coordinates maxdist=0.000783933 isNearSphere=1
Sort: { distance: 1 }
Proj: {}
 planner returned error :: caused by :: unable to find index for $geoNear query
```

SET2

Inserting the data into MongoDB

```
employee> db.hotel_data.insert(
... {
... 'address': {
... 'building': '1500',
... 'geo-loc': [ -73.923973, 40.819819 ],
... 'street': 'Southern Blvd',
... 'zipcode': '10460'
... },
... 'Hotel_Name': 'Bronx Halal Palace',
... 'cuisine': 'Middle Eastern',
... 'grades': [
...
... { 'date': { '$date': '2022-05-09T00:00:00Z' }, 'grade': 'A', 'score': 12 },
... { 'date': { '$date': '2022-02-20T00:00:00Z' }, 'grade': 'A', 'score': 8 },
... { 'date': { '$date': '2021-12-01T00:00:00Z' }, 'grade': 'B', 'score': 18 },
... { 'date': { '$date': '2021-10-10T00:00:00Z' }, 'grade': 'C', 'score': 21 },
... { 'date': { '$date': '2021-08-08T00:00:00Z' }, 'grade': 'A', 'score': 11 }
... ],
... 'Owner_name': 'Abdul Malik',
... 'Hotel_id': '30100009'
... })
```

```
employee> db.hotel_data.find()
[
  {
    _id: ObjectId("645ba6b15fc20294d62ecd57"),
    address: {
      building: '100',
      'geo-loc': [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '412112'
    },
    Hotel_Name: 'Bronx Annaseva',
    cuisine: 'Indian',
    grades: [
      {
        date: { '$date': '2023-05-10T00:00:00Z' },
        grade: 'A',
        score: 2
      },
      {
        date: { '$date': '2023-05-10T00:00:00Z' },
        grade: 'A',
        score: 6
      },
      {
        date: { '$date': '2023-05-10T00:00:00Z' },
        grade: 'A',
        score: 10
      },
      {
        date: { '$date': '2023-05-10T00:00:00Z' },
        grade: 'A',
        score: 9
      },
      {
        date: { '$date': '2023-05-10T00:00:00Z' },
        grade: 'B',
        score: 14
      }
    ],
    Owner_name: 'Morris Park Bake Shop',
    Hotel_id: '30075445'
  },
```

Q5) Write a MongoDB query to find the hotels that have all grades with a score greater than 5 and cuisine as “Indian”.

Syntax: db.hotel_data.find({$and: [{cuisine: "Indian"}, {'grades.score': {$gt: 5}}]})

Q6) Create a 2d sphere index and find the hotels that are located within 100 km from the co-ordinates: -73.856077, 40.848447.

Syntax: db.hotel_data.createIndex({'address.geo-loc': "2dsphere"})

db.hotel_data.find({'address.geo-loc': {$nearSphere: {$geometry: {type: "Point", coordinates: [-73.856077, 40.848447]}, $maxDistance: 10000}}})