Name: Urooba Gohar

Roll No: 22P-9216

Section: BSCS-6A

Database Systems Labtask 11

Class Task:

1. Create a database named SchoolDB.

Answer:

Use use SchoolDB to create and switch to the SchoolDB database:

```
test> use SchoolDB
```

This is the output:

```
switched to db SchoolDB
```

---

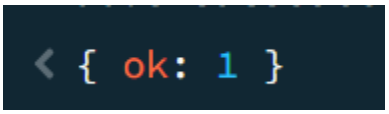
2. Create two collections:
   - o Students
   - o Courses

Answer:

Run the following query to create the Students and Courses collections:

```
SchoolDB> db.createCollection("Students")
          db.createCollection("Courses")
```

This is the output:



---

3. Insert the following documents into the Students collection:

Answer:

Run the following query to insert the documents into the Students collection:

```
SchoolDB > db.Students.insertMany([
            {
              _id: 1,
              name: "Alice",
              age: 20,
              scores: {
                math: 85,
                science: 90
              }
            },
            {
              _id: 2,
              name: "Bob",
              age: 22,
              scores: {
                math: 78,
                science: 82
              }
            },
            {
              _id: 3,
              name: "Charlie",
              age: 21,
              scores: {
                math: 92,
                science: 88
              }
            },
            {
              _id: 4,
              name: "Daisy",
              age: 23,
              scores: {
                math: 68,
                science: 74
              }
            }
          ])
```

This is the output:

---

4. Insert the following documents into the Courses collection:

Answer:

Write the following query to insert the following documents into Courses collection:

```
SchoolDB> db.Courses.insertMany([
            {
              _id: 101,
              courseName: "Mathematics",
              instructor: "Dr. Smith",
              studentsEnrolled: [1, 2, 3]
            },
            {
              _id: 102,
              courseName: "Science",
              instructor: "Dr. Adams",
              studentsEnrolled: [2, 3, 4]
            }
          ])
```

This is the output:

```
< {
    acknowledged: true,
    insertedIds: {
      '0': 101,
      '1': 102
    }
  }
```

---

5. Use findOne to retrieve:

   o A student where the math score is >=85 and the age is <22.

   o A course where the studentsEnrolled array includes 3 and the instructor is “Dr. Adams”.

Answer:

Run the following queries to retrieve the specified student and course:

```
SchoolDB> db.Students.findOne({
            "scores.math": { $gte: 85 },
            age: { $lt: 22 }
          })
```

This is the output:

```
< {
    _id: 1,
    name: 'Alice',
    age: 20,
    scores: {
      math: 85,
      science: 90
    }
  }
```

```
SchoolDB> db.Courses.findOne({
            studentsEnrolled: 3,
            instructor: "Dr. Adams"
          })
```

This is the output:

```
{
  _id: 102,
  courseName: 'Science',
  instructor: 'Dr. Adams',
  studentsEnrolled: [
    2,
    3,
    4
  ]
}
```

---

6. Use find to retrieve:

   o Students with math score >=80 and science score <90.

   o Students whose age is <23 or have a math score >=85.

   o Students with science score >=80 and (either math score <75 or age>22).

Answer:

To find students with math score >=80 and science score <90, write:

```
> db.Students.find({
    "scores.math": { $gte: 80 },
    "scores.science": { $lt: 90 }
  })
```

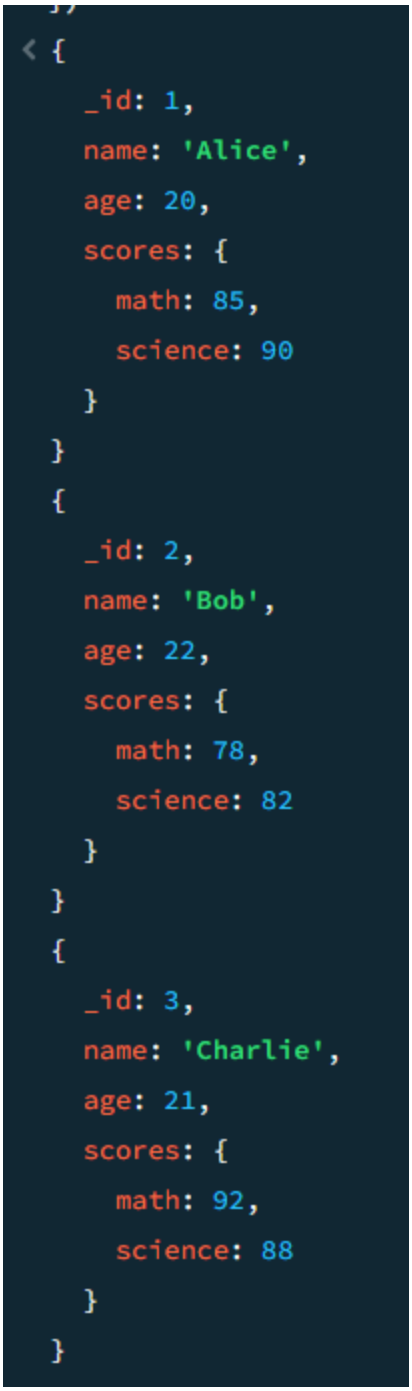
This is the output:

```
{
  _id: 3,
  name: 'Charlie',
  age: 21,
  scores: {
    math: 92,
    science: 88
  }
}
```

To find students whose age <23 or have a math score >=85, write:

```
SchoolDB> db.Students.find({
            $or: [
              { age: { $lt: 23 } },
              { "scores.math": { $gte: 85 } }
            ]
          })
```

This is the output:



Students with science score >=80 and (either math score <75 or age>22), write:

```
SchoolDB> db.Students.find({
            "scores.science": { $gte: 80 },
            $or: [
              { "scores.math": { $lt: 75 } },
              { age: { $gt: 22 } }
            ]
          })
```

Since no such students exist, there is no output.

---

7. Use updateOne to:

   o Increase the science score of the student where name is “Bob” and math score is >=75.

Answer:

To increase the science score of the student where name is “Bob” and math score is >=75, write:

```
SchoolDB> db.Students.updateOne(
            { name: "Bob", "scores.math": { $gte: 75 } },
            { $inc: { "scores.science": 1 } }
          )
```

This is the output:

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

---

8. Use updateMany to:

   o Increase the math score by 5 for students whose science score is <80 and age >22.

Answer:

To Increase the math score by 5 for students whose science score is <80 and age >22, write:

```
}
SchoolDB> db.Students.updateMany(
            { "scores.science": { $lt: 80 }, age: { $gt: 22 } },
            { $inc: { "scores.math": 5 } }
          )
```

This is the output:

```
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

---

9. Use deleteOne to:

   o Remove a student where name is “Daisy” and their science score is <80.

Answer:

To remove a student where name is “Daisy” and their science score is <80, run:

```
SchoolDB> db.Students.deleteOne({
            name: "Daisy",
            "scores.science": { $lt: 80 }
          })
```

This is the output:

```
< {
    acknowledged: true,
    deletedCount: 1
  }
```

---

10. Use deleteMany to:

    o Remove courses where the studentsEnrolled array includes 2 or the instructor is “Dr. Smith”.

Answer:

To remove courses where the studentsEnrolled array includes 2 or the instructor is “Dr. Smith”, write:

```
SchoolDB> db.Courses.deleteMany({
            $or: [
              { studentsEnrolled: 2 },
              { instructor: "Dr. Smith" }
            ]
          })
```

This is the output:

```
< {
    acknowledged: true,
    deletedCount: 2
  }
```

---

11. Drop the Students collection.

Answer:

Run db.Students.drop(); to drop the Students collection:

```
> db.Students.drop()
< true
```

---

12. Drop the Courses collection.

Answer:

Run the following to drop the Courses collection:

```
> db.Courses.drop()
< true
```

---

13. Finally, delete the SchoolDB database.

Answer:

Run db.dropDatabase(); to delete the SchoolDB database:

```
> db.dropDatabase()
< { ok: 1, dropped: 'SchoolDB' }
```

Task 1:

Write a MongoDB query to display all the documents in the collection restaurants.

Answer:

First make the database:

```
test> use restaurantDB
```

This is the output:

```
< switched to db restaurantDB
```

Now insert values in it:

```
restaurantDB> db.restaurants.insertOne({
                "address": {
                  "building": "1007",
                  "coord": [-73.856077, 40.848447],
                  "street": "Morris Park Ave",
                  "zipcode": "10462"
                },
                "borough": "Bronx",
                "cuisine": "Bakery",
                "grades": [
                  { "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },
                  { "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },
                  { "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },
                  { "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },
                  { "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }
                ],
                "name": "Morris Park Bake Shop",
                "restaurant_id": "30075445"
              });
```

This is the output:

```
< {
    acknowledged: true,
    insertedId: ObjectId('680b3e76ba6f73febcccc7e8')
  }
```

Now display all the documents in the restaurants collection:

```
> db.restaurants.find()
```

This is the output:

```
{
  _id: ObjectId('680b3e76ba6f73febcccc7e8'),
  address: {
    building: '1007',
    coord: [
      -73.856077,
      40.848447
    ],
    street: 'Morris Park Ave',
    zipcode: '10462'
  },
  borough: 'Bronx',
  cuisine: 'Bakery',
  grades: [
    {
      date: 2014-03-03T00:00:00.000Z,
      grade: 'A',
      score: 2
    },
    {
      date: 2013-09-11T00:00:00.000Z,
      grade: 'A',
      score: 6
    },
    {
      date: 2013-01-24T00:00:00.000Z,
      grade: 'A',
      score: 10
    },
    {
      date: 2011-11-23T00:00:00.000Z,
      grade: 'A',
      score: 9
    },
    {
      date: 2011-03-10T00:00:00.000Z,
      grade: 'B',
      score: 14
    }
  ],
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
```

Task 2:

Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine for all the documents in the collection restaurant.

Answer:

Run the following query to display the fields restaurant_id, name, borough and cuisine for all the documents in the collection restaurant:

```
restaurantDB > db.restaurants.find(
                {},
                {
                  restaurant_id: 1,
                  name: 1,
                  borough: 1,
                  cuisine: 1
                }
              )
```

This is the output:

```
< {
    _id: ObjectId('680b3e76ba6f73febcccc7e8'),
    borough: 'Bronx',
    cuisine: 'Bakery',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
```

Task 3:

Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine, but exclude the field _id for all the documents in the collection restaurant.

Answer:

Run the following query to display the fields restaurant_id, name, borough and cuisine, but exclude the field _id for all the documents in the collection restaurant:

```
restaurantDB> db.restaurants.find(
                {},
                {
                  restaurant_id: 1,
                  name: 1,
                  borough: 1,
                  cuisine: 1,
                  _id: 0
                }
              )
```

This is the output:

```
< {
    borough: 'Bronx',
    cuisine: 'Bakery',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
```

_____

Task 4:

Write a MongoDB query to display the fields restaurant_id, name, borough and zip code, but exclude the field _id for all the documents in the collection restaurant.

Answer:

Run the following query to display the fields restaurant_id, name, borough and zip code, but exclude the field _id for all the documents in the collection restaurant:

```
restaurantDB> db.restaurants.find(
                {},
                {
                  restaurant_id: 1,
                  name: 1,
                  borough: 1,
                  "address.zipcode": 1,
                  _id: 0
                }
              )
```

This is the output:

```
< {
    address: {
      zipcode: '10462'
    },
    borough: 'Bronx',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
```

_____

Task 5:

Write a MongoDB query to display all the restaurant which is in the borough Bronx.

Answer:

Write the following query to display all the restaurant which is in the borough Bronx:

```
restaurantDB> db.restaurants.find(
                { borough: "Bronx" }
              )
```

This is the output:

```
{
  _id: ObjectId('680b3e76ba6f73febcccc7e8'),
  address: {
    building: '1007',
    coord: [
      -73.856077,
      40.848447
    ],
    street: 'Morris Park Ave',
    zipcode: '10462'
  },
  borough: 'Bronx',
  cuisine: 'Bakery',
  grades: [
    {
      date: 2014-03-03T00:00:00.000Z,
      grade: 'A',
      score: 2
    },
    {
      date: 2013-09-11T00:00:00.000Z,
      grade: 'A',
      score: 6
    },
    {
      date: 2013-01-24T00:00:00.000Z,
      grade: 'A',
      score: 10
    },
    {
      date: 2011-11-23T00:00:00.000Z,
      grade: 'A',
      score: 9
    },
    {
      date: 2011-03-10T00:00:00.000Z,
      grade: 'B',
      score: 14
    }
  ],
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
```

_____

Task 6:

Write a MongoDB query to display the first 5 restaurant which is in the borough Bronx.

Answer:

To display the first 5 restaurant which is in the borough Bronx, run the following query:

```
restaurantDB> db.restaurants.find(
                { borough: "Bronx" }
              ).limit(5)
```

This is the output:

```
{
  _id: ObjectId('680b3e76ba6f73febcccc7e8'),
  address: {
    building: '1007',
    coord: [
      -73.856077,
      40.848447
    ],
    street: 'Morris Park Ave',
    zipcode: '10462'
  },
  borough: 'Bronx',
  cuisine: 'Bakery',
  grades: [
    {
      date: 2014-03-03T00:00:00.000Z,
      grade: 'A',
      score: 2
    },
    {
      date: 2013-09-11T00:00:00.000Z,
      grade: 'A',
      score: 6
    },
    {
      date: 2013-01-24T00:00:00.000Z,
      grade: 'A',
      score: 10
    },
    {
      date: 2011-11-23T00:00:00.000Z,
      grade: 'A',
      score: 9
    },
    {
      date: 2011-03-10T00:00:00.000Z,
      grade: 'B',
      score: 14
    }
  ],
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
```

Task 7:

Write a MongoDB query to find the restaurants who achieved a score more than 90.

Answer:

To find the restaurants who achieved a score more than 90, run the following query:

```
restaurantDB> db.restaurants.find(
                { "grades.score": { $gt: 90 } }
              )
```

As no such restaurant exists, there is no output.

Task 8:

Write a MongoDB query to find the restaurants that achieved a score, more than 80 but less than 100.

Answer:

To find the restaurants that achieved a score, more than 80 but less than 100, run the following:

```
restaurantDB> db.restaurants.find(
                { "grades.score": { $gt: 80, $lt: 100 } }
              )
```

As no such restaurant exists, there is no output.

Task 9:

Write a MongoDB query to find the restaurants which locate in latitude value less than -95.754168.

Answer:

To find the restaurants which locate in latitude value less than -95.754168, run the following:

```
restaurantDB> db.restaurants.find(
                { "address.coord.0": { $lt: -95.754168 } }
              )
```

As no such restaurant exists, there is no output.

Task 10:

Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish.

Answer:

To find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish, run the following:

```
restaurantDB> db.restaurants.find(
                {
                  "borough": "Bronx",
                  "cuisine": { $in: ["American", "Chinese"] }
                }
              )
```

As no such restaurant exists, there is no output.
