NoSQL LAB MANUAL

MongoDB Installation:

1. Download MongoDB Installer

- Visit: https://www.mongodb.com/try/download/community
- Choose:
  - Version: *Latest stable*
  - Platform: *Windows*
  - Package: *MSI*

2. Run the Installer

- Double-click the .msi file.
- Choose Complete Setup.
- Make sure to check the box to install MongoDB as a Windows Service.

3. Installation of MongoDB Compass (GUI Tool) will be started

4. Set Environment Variable (if needed)

- Add C:\Program Files\MongoDB\Server\<version>\bin to your system PATH.

5. Verify Installation by Opening MongoDB Compass.

Demonstrate

Lab Program1:

1. Create an Employee collection consisting of Employee ID, Name, Age, Gender, Department and Address. Perform the following CRUD operations using MongoDB
   a. Insert details of the employees as documents.
   b. Retrieve the employee details whose age is less than or equal to 40 and display the same.
   c. Display the employees whose salary is less than 50k.
   d. Demonstrate Update operation.
   e. Remove the details of the employees belongs to any particular department.

a. Insert details of the employees as documents.

```
use employeeDB;
db.employees.insertMany([
  {
    empId: "E001",
    name: "Anjali",
    age: 35,
    gender: "Female",
    department: "HR",
    salary: 48000,
    address: { city: "Bangalore", pincode: 560001 }
  },
  {
    empId: "E002",
    name: "Rakesh",
    age: 42,
    gender: "Male",
    department: "IT",
    salary: 62000,
    address: { city: "Mysore", pincode: 570012 }
  },
  {
    empId: "E003",
    name: "Divya",
    age: 29,
    gender: "Female",
    department: "Marketing",
    salary: 45000,
    address: { city: "Chennai", pincode: 600001 }
  },
  {
    empId: "E004",
    name: "Manoj",
    age: 38,
    gender: "Male",
    department: "IT",
    salary: 72000,
    address: { city: "Hyderabad", pincode: 500001 }
  }
]);
```

b. **Retrieve the employee details whose age is less than or equal to 40 and display the same.**

```
db.employees.find({ age: { $lte: 40 } }).pretty();
```

c. **Display the employees whose salary is less than 50k.**

```
db.employees.find({ salary: { $lt: 50000 } }).pretty();
```

d. **Demonstrate Update operation.**

```
db.employees.updateOne(
  { empId: "E003" },
  { $set: { salary: 55000 } }
);
db.employees.updateMany(
  { department: "IT" },
  { $set: { department: "Tech Support" } } );
```

e. **Remove the details of the employees belongs to any particular department.**

```
db.employees.deleteMany({ department: "HR" });
    db.employees.find().pretty();
```

Demonstrate:

Lab Program2:

2. Create a university database containing collections for students, courses, departments, enrollments, and faculty. Each collection will have schema validation rules (e.g., minimum age, valid dates).Insert sample data and perform aggregation queries involving $lookup, $match, $group, $project, $sort, $unwind, and $facet to produce complex reports.

a. Switch to database
b. Create collections with validation
c. Insert sample data (10+ docs each)
d. Aggregation queries
   i. List each student with department name
   ii. Show each student's enrolled courses with course names
   iii. Number of students per department
   iv. Faculty list with their courses and departments
   v. Department summary - total students, total faculty, total courses

a. Switch to database

```
use universityDB
```

b. Create collections with validation

```
db.createCollection("students", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["studentId", "name", "age", "deptId"],
      properties: {
        studentId: { bsonType: "int" },
        name: { bsonType: "string" },
        age: { bsonType: "int", minimum: 18 },
        deptId: { bsonType: "int" }
      }
    }
  }
})
```

```
db.createCollection("courses", {
   validator: {
      $jsonSchema: {
         bsonType: "object",
         required: ["courseId", "courseName", "deptId"],
         properties: {
            courseId: { bsonType: "int" },
            courseName: { bsonType: "string" },
            deptId: { bsonType: "int" }
         }
      }
   }
})

db.createCollection("departments", {
   validator: {
      $jsonSchema: {
         bsonType: "object",
         required: ["deptId", "deptName"],
         properties: {
            deptId: { bsonType: "int" },
            deptName: { bsonType: "string" }
         }
      }
   }
})

db.createCollection("faculty", {
   validator: {
      $jsonSchema: {
         bsonType: "object",
         required: ["facultyId", "name", "deptId", "joiningDate"],
         properties: {
            facultyId: { bsonType: "int" },
            name: { bsonType: "string" },
            deptId: { bsonType: "int" },
            joiningDate: { bsonType: "date" }
         }
      }
   }
})

db.createCollection("enrollments", {
   validator: {
      $jsonSchema: {
         bsonType: "object",
            required: ["studentId", "courseId", "enrollDate"],
            properties: {
              studentId: { bsonType: "int" },
              courseId: { bsonType: "int" },
              enrollDate: { bsonType: "date" }
            }
          }
        }
      })
```

c. Insert sample data (10+ docs each)

```
// Students

db.students.insertMany([

  { studentId: 1, name: "Arjun", age: 20, deptId: 101 },

  { studentId: 2, name: "Priya", age: 21, deptId: 102 },

  { studentId: 3, name: "Ravi", age: 22, deptId: 101 },

  { studentId: 4, name: "Meena", age: 20, deptId: 103 },

  { studentId: 5, name: "Amit", age: 23, deptId: 101 },

  { studentId: 6, name: "Kiran", age: 19, deptId: 102 },

  { studentId: 7, name: "Sita", age: 20, deptId: 103 },

  { studentId: 8, name: "Raj", age: 21, deptId: 104 },

  { studentId: 9, name: "Anita", age: 22, deptId: 101 },

  { studentId: 10, name: "Vikram", age: 23, deptId: 104 }

])

// Departments

db.departments.insertMany([

  { deptId: 101, deptName: "Computer Science" },

  { deptId: 102, deptName: "Electronics" },

  { deptId: 103, deptName: "Mechanical" },

  { deptId: 104, deptName: "Civil" },

  { deptId: 105, deptName: "Architecture" },

  { deptId: 106, deptName: "Biotechnology" },

  { deptId: 107, deptName: "Mathematics" },

  { deptId: 108, deptName: "Physics" },
  { deptId: 109, deptName: "Chemistry" },
  { deptId: 110, deptName: "English" }
])
// Courses
db.courses.insertMany([
  { courseId: 1, courseName: "DBMS", deptId: 101 },
  { courseId: 2, courseName: "Data Structures", deptId: 101 },
  { courseId: 3, courseName: "Circuits", deptId: 102 },
  { courseId: 4, courseName: "Thermodynamics", deptId: 103 },
  { courseId: 5, courseName: "Surveying", deptId: 104 },
  { courseId: 6, courseName: "Microbiology", deptId: 106 },
  { courseId: 7, courseName: "Calculus", deptId: 107 },
  { courseId: 8, courseName: "Optics", deptId: 108 },
  { courseId: 9, courseName: "Organic Chemistry", deptId: 109 },
  { courseId: 10, courseName: "Literature", deptId: 110 }
])
// Faculty
db.faculty.insertMany([
  { facultyId: 1, name: "Dr. Kumar", deptId: 101, joiningDate: ISODate("2018-06-10") },
  { facultyId: 2, name: "Dr. Meera", deptId: 102, joiningDate: ISODate("2017-07-15") },
  { facultyId: 3, name: "Dr. Ramesh", deptId: 103, joiningDate: ISODate("2019-03-20") },
  { facultyId: 4, name: "Dr. Anita", deptId: 104, joiningDate: ISODate("2016-08-25") },
  { facultyId: 5, name: "Dr. Vijay", deptId: 105, joiningDate: ISODate("2020-01-12") },
  { facultyId: 6, name: "Dr. Sunita", deptId: 106, joiningDate: ISODate("2021-09-05") },
  { facultyId: 7, name: "Dr. Karthik", deptId: 107, joiningDate: ISODate("2015-11-11") },
  { facultyId: 8, name: "Dr. Rekha", deptId: 108, joiningDate: ISODate("2019-02-14") },
  { facultyId: 9, name: "Dr. Arvind", deptId: 109, joiningDate: ISODate("2020-05-18") },
  { facultyId: 10, name: "Dr. Nisha", deptId: 110, joiningDate: ISODate("2022-04-21") }
])
// Enrollments
db.enrollments.insertMany([
  { studentId: 1, courseId: 1, enrollDate: ISODate("2023-08-01") },
  { studentId: 1, courseId: 2, enrollDate: ISODate("2023-08-02") },
  { studentId: 2, courseId: 3, enrollDate: ISODate("2023-08-03") },
  { studentId: 3, courseId: 1, enrollDate: ISODate("2023-08-04") },
  { studentId: 4, courseId: 4, enrollDate: ISODate("2023-08-05") },
  { studentId: 5, courseId: 2, enrollDate: ISODate("2023-08-06") },
  { studentId: 6, courseId: 3, enrollDate: ISODate("2023-08-07") },
  { studentId: 7, courseId: 4, enrollDate: ISODate("2023-08-08") },
  { studentId: 8, courseId: 5, enrollDate: ISODate("2023-08-09") },
  { studentId: 9, courseId: 1, enrollDate: ISODate("2023-08-10") }
])
```

d. Aggregation queries

i. List each student with department name

```
db.students.aggregate([
   { $lookup: {
      from: "departments",
      localField: "deptId",
      foreignField: "deptId",
      as: "deptInfo"
   }},
   { $unwind: "$deptInfo" },
   { $project: { _id: 0, studentId: 1, name: 1, department: "$deptInfo.deptName" } }
])
```

ii. Show each student's enrolled courses with course names

```
db.students.aggregate([
   { $lookup: {
      from: "enrollments",
      localField: "studentId",
      foreignField: "studentId",
      as: "enrollments"
   }},
   { $unwind: "$enrollments" },
   { $lookup: {
      from: "courses",
      localField: "enrollments.courseId",
      foreignField: "courseId",
      as: "courseInfo"
   }},
   { $unwind: "$courseInfo" },
  { $project: { _id: 0, name: 1, course: "$courseInfo.courseName", enrollDate: "$enrollments.enrollDate" } }
])
```

iii. Number of students per department

```
db.students.aggregate([
  { $group: { _id: "$deptId", totalStudents: { $sum: 1 } } },
  { $lookup: {
    from: "departments",
    localField: "_id",
    foreignField: "deptId",
    as: "deptInfo"
  }},
  { $unwind: "$deptInfo" },
  { $project: { department: "$deptInfo.deptName", totalStudents: 1, _id: 0 } },
  { $sort: { totalStudents: -1 } }
])
```

iv. Faculty list with their courses and departments

```
db.faculty.aggregate([
  { $lookup: {
    from: "departments",
    localField: "deptId",
    foreignField: "deptId",
    as: "deptInfo"
  }},
  { $unwind: "$deptInfo" },
  { $lookup: {
    from: "courses",
    localField: "deptId",
    foreignField: "deptId",
    as: "coursesTaught"
  }},
  { $project: { name: 1, department: "$deptInfo.deptName", coursesTaught: "$coursesTaught.courseName" } }
])
```

v. Department summary - total students, total faculty, total courses

```
db.departments.aggregate([
  { $lookup: {
    from: "students",
    localField: "deptId",
    foreignField: "deptId",
    as: "studentsList"
  }},
  { $lookup: {
    from: "faculty",
    localField: "deptId",
        foreignField: "deptId",
        as: "facultyList"
    }},
    { $lookup: {
        from: "courses",
        localField: "deptId",
        foreignField: "deptId",
        as: "coursesList"
    }},
    { $project: {
        _id: 0,
        deptName: 1,
        totalStudents: { $size: "$studentsList" },
        totalFaculty: { $size: "$facultyList" },
        totalCourses: { $size: "$coursesList" }
    }}
])
```