SRINIVAS UNIVERSITY
INSTITUTE OF ENGINEEERING&TECHNOLOGY
MUKKA, MANGALURU

LAB MANUAL
FOR
ADVANCED DBMS & NOSQL DATABASES
22SCS072

Prepared by

Prof. Ruksana Banu

Asst. Professor

Dept Of CSE, SUIET, Mukka

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

SRINIVAS UNIVERSITY
INSTITUTE OF ENGINEERING & TECHNOLOGY
MUKKA, MANGALURU
KARNATAKA – 574146

Program 1
Installation and configuration of MongoDB on Windows & Linux.
i. MongoDB connectivity.
ii. Read the Version and connection.

1. Installation and Configuration of MongoDB on Windows

A. Install MongoDB on Windows

Step 1: Download MongoDB Installer

- Go to: https://www.mongodb.com/try/download/community
- Choose:
  - **Edition**: Community Server
  - **OS**: Windows
  - **Package**: MSI
- Download the installer.

Step 2: Run the Installer

- Open the downloaded .msi file.
- Choose **Complete** setup (recommended).
- Ensure the checkbox **"Install MongoDB as a Service"** is selected.
- Also check **"Install MongoDB Compass"** if you want the GUI.

Step 3: Add MongoDB to Windows PATH

- MongoDB is installed in:

  C:\Program Files\MongoDB\Server\<version>\bin

  To add it to PATH:

- Open **System Properties → Environment Variables**
- Under **System variables**, find Path → Edit → Add the above MongoDB bin path.

B. Start MongoDB Service

After installation, MongoDB runs as a Windows service.

- To check if MongoDB is running:

```
net start MongoDB
```

You can also verify it using **Services** from the Start menu (services.msc).

2. MongoDB Connectivity (on Windows)

i. Using MongoDB Shell (Command Line)

Open CMD or PowerShell and run:

```
mongo
```

This opens the MongoDB shell connected to:

- **Host**: localhost
- **Port**: 27017

You can now run commands like:

```
show dbs
use test
db.testCollection.insertOne({ name: "Veeda" })
```

ii. Using MongoDB Compass (GUI Tool)

- Open Compass.
- Connection details:
  - **Hostname**: localhost
  - **Port**: 27017
- Click **Connect**

3. Read MongoDB Version and Connection Info

Check MongoDB Version

Open CMD and run:

```
mongo --eval “db.version()"
```

Or inside the shell:

```
db.version()
```

Check Connection Status

Inside the Mongo shell, run:

```
db.runCommand({ connectionStatus: 1 })
```

It returns JSON showing connection details like authenticated user, roles, and status.

Program 2

Perform MongoDB operations.
i. Create Database "Student_Data".
ii. Connect to the Database.
iii. Use the Database and checking Database connectivity.
iv. Check for the Database list using commands after creating a document in it.
v. Drop the Database.

1. Create Database 'Student_Data'
Using MongoDB Compass:
- Open MongoDB Compass and connect to mongodb://localhost:27017
- Click 'Create Database'
- Enter Database Name: Student_Data
- Enter Collection Name: students
- Click 'Create Database'
Using Command Prompt (mongosh):
> mongosh
> use Student_Data
(Note: The database is created when a collection/document is added)
2. Connect to the Database
Using MongoDB Compass:
- Open Compass and connect using mongodb://localhost:27017
- Click on the 'Student_Data' database
Using Command Prompt (mongosh):
> use Student_Data
> db
(Output: Student_Data confirms connection)
3. Use the Database and Check Connectivity
Using Compass: - After selecting the database, insert a document in 'students' collection:
{
"name": "Amit",
"age": 21,
MongoDB Installation and Basic Operations on Windows
"course": "MCA"
}
- Click 'Insert'
Using mongosh:
> use Student_Data
> db.students.insertOne({ name: "Amit", age: 21, course: "MCA" })
> db.students.find()
(Confirms connectivity and data)
4. Check Database List After Creating Document
Using Compass:
- Return to main Compass screen to see 'Student_Data' listed

Using mongosh:
> show dbs
(Student_Data will now appear because it has data)

5. Drop the Database
Using Compass:
- Select the 'Student_Data' database
- Click the trash/delete icon to drop it
Using mongosh:
> use Student_Data
> db.dropDatabase()
> show dbs
(Student_Data should no longer be listed)

Program 3

Perform the MongoDB commands – Insert, Query, Update, Delete and Projection

(Using both MongoDB Compass and Command Prompt)

1. INSERT Operation

Using MongoDB Compass

1. Open your database and select the **collection** (e.g., students).
2. Click on **"Insert Document"**.
3. Add the document like:

json
CopyEdit
{
"name": "Rahul",
"age": 22,
"course": "MCA"
}

1. Click **"Insert"**.

To insert multiple documents, use "Add Document" multiple times in the same insert window.

Using Command Prompt (mongosh)

js
CopyEdit
// Insert one document
db.students.insertOne({ name: "Rahul", age: 22, course: "MCA" })
// Insert multiple documents
db.students.insertMany([
{ name: "Asha", age: 23, course: "BCA" },
{ name: "Vikram", age: 24, course: "BSc" }
])

2. QUERY (FIND) Operation

Using MongoDB Compass

1. Go to your collection.
2. Use the **"Filter"** bar to write a query:

json
CopyEdit
{ "course": "MCA" }

1. Click **"Find"** to view results.

Using Command Prompt (mongosh)
js
CopyEdit
// Find all documents
db.students.find()
// Find documents with a condition
db.students.find({ course: "MCA" })
// Find a single document
db.students.findOne({ name: "Rahul" })

3. UPDATE Operation
Using MongoDB Compass
1. Click the **pencil icon** next to a document.

2. Edit the field(s) you want (e.g., change "age": 22 to "age": 23).

3. Click **"Update"**.

Using Command Prompt (mongosh)
js
CopyEdit
// Update one document
db.students.updateOne(
{ name: "Rahul" },
{ $set: { age: 23 } }
)
// Update many documents
db.students.updateMany(
{ course: "BCA" },
{ $set: { duration: "3 years" } }
)

4. DELETE Operation
Using MongoDB Compass
1. Click the **trash bin icon** next to the document.

2. Confirm deletion when prompted.

Using Command Prompt (mongosh)
js
CopyEdit
// Delete one document
db.students.deleteOne({ name: "Vikram" })
// Delete multiple documents
db.students.deleteMany({ course: "BCA" })

5. PROJECTION (Show Specific Fields)

Using MongoDB Compass

1. Go to the **"Project"** tab in your collection.

Enter fields to show or hide: Include fields:

json
Exclude fields:

CopyEdit
{ "name": 1, "age": 1, "_id": 0 }
json
CopyEdit
{ "course": 0 }

1. Click **"Apply"**.

Using Command Prompt (mongosh)

js
CopyEdit
// Include specific fields only
db.students.find({}, { name: 1, age: 1, _id: 0 })
// Exclude a field
db.students.find({}, { course: 0 })

Program 4

Where Clause equivalent in MongoDB.

1. **Write a Program using Where clause and include all basic commands in it.**

WHERE Clause Equivalent in MongoDB

In SQL:

sql

```
SELECT * FROM students WHERE age > 21;
```

In MongoDB:

javascript

```
db.students.find({ age: { $gt: 21 } })
```

PROGRAM: Insert, Query (WHERE), Update, Delete — in mongosh (Command Prompt)

1. Connect & Use Database

bash

```
mongosh
```

javascript

```
use Student_Data
```

2. Insert Multiple Documents

javascript

```
db.students.insertMany([
{ name: "Rahul", age: 22, course: "MCA", marks: 75 },
{ name: "Asha", age: 23, course: "BCA", marks: 80 },
{ name: "Vikram", age: 20, course: "BSc", marks: 60 },
{ name: "Sneha", age: 21, course: "MCA", marks: 85 }
])
```

3. WHERE Clause Equivalent (Query)

Javascript

```
// Find students with age > 21
db.students.find({ age: { $gt: 21 } })
// Find students in MCA
db.students.find({ course: "MCA" })
// Find students with marks between 70 and 90
db.students.find({ marks: { $gte: 70, $lte: 90 } })
```

4. Update a Document

javascript

```
// Update Sneha's course to 'MSc'
```

db.students.updateOne({ name: "Sneha" }, { $set: { course: "MSc" } })
5. Delete a Document
javascript
// Delete student named Vikram
db.students.deleteOne({ name: "Vikram" })
6. Show All Data
javascript
db.students.find()
Same Operations in MongoDB Compass
1. Create Database and Collection

Open Compass → Click **"Create Database"** Database Name: Student_Data

Collection Name: students

Click **Create**

2. Insert Documents

- Go to students collection
- Click **"Insert Document"**
- Insert:

json
{
"name": "Rahul",
"age": 22,
"course": "MCA",
"marks": 75
}

- Insert other documents similarly.

3. WHERE Clause Equivalent (Find with Filter)
In the **Filter** bar, use:

- Age > 21:

json
{ "age": { "$gt": 21 } }

- Course = MCA:

json
{ "course": "MCA" }

- Marks between 70 and 90:

json

{ "marks": { "$gte": 70, "$lte": 90 } }

Click **"Find"** to view results.

4. Update a Document

- Find the document
- Click the **pencil icon**
- Edit "course": "MCA" to "course": "MSc"
- Click **Update**

5. Delete a Document

- Find the document for Vikram
- Click **trash bin icon** to delete

6. View All Documents

- Click **Documents** tab to see all inserted data.

Program 5

Perform operations in MongoDB: AND in MongoDB, OR in MongoDB, Limit Records and Sort Records. Perform operations in MongoDB – Indexing, Advanced Indexing, Aggregation and Map Reduce.

1. **Write a program using all Arithmetic operations.**

Step 1: Setup MongoDB

Option A: MongoDB Compass (GUI)

1. **Install MongoDB Compass** from https://www.mongodb.com/try/download/compass
2. Open Compass → connect to default localhost:
3. `mongodb://localhost:27017`
4. Create a **database named `lab`** and **collection named `students`**.

Option B: MongoDB Shell (CLI)

1. Install MongoDB server and shell
2. Open terminal → Start Mongo shell:
3. `mongosh`
4. Switch to a database and create collection:
5. `use lab`
6. `db.createCollection("students")`

📥 Step 2: Insert Sample Data

Insert one or more student documents:

```
db.students.insertMany([
  { name: "Sumit", age: 22, course: "MCA", marks: 75, marks1: 75, marks2: 85 },
  { name: "Anita", age: 20, course: "BCA", marks: 65, marks1: 60, marks2: 70 },
  { name: "Raj", age: 23, course: "MCA", marks: 80, marks1: 80, marks2: 90 }
])
```

 Step 3: Run Each Query

1. AND Operator

```
db.students.find({ age: { $gt: 20 }, course: "MCA" })
```

2. OR Operator

```
db.students.find({ $or: [{ age: { $lt: 21 } }, { course: "BCA" }] })
```

3. LIMIT

```
db.students.find().limit(2)
```

4. SORT

```
db.students.find().sort({ age: 1 })          // Ascending by age
db.students.find().sort({ marks: -1 })       // Descending by marks
```

5. Create Index

```
db.students.createIndex({ name: 1 })
db.students.createIndex({ course: 1, age: -1 })
```

6. View/Drop Index

```
db.students.getIndexes()
db.students.dropIndex({ name: 1 })
```

Aggregation Queries

A. Group by course and calculate average marks

```
db.students.aggregate([
  { $group: { _id: "$course", avgMarks: { $avg: "$marks" } } }
])
```

Map-Reduce

Count students in each course:

```
var mapFunction = function() {
  emit(this.course, 1);
};

var reduceFunction = function(key, values) {
  return Array.sum(values);
};

db.students.mapReduce(mapFunction, reduceFunction, { out: "course_counts" })

// View results:
```

```
db.course_counts.find()
```

➗ Arithmetic Operations

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      total: { $add: ["$marks1", "$marks2"] },
      difference: { $subtract: ["$marks2", "$marks1"] },
      product: { $multiply: ["$marks1", "$marks2"] },
      average: { $divide: [{ $add: ["$marks1", "$marks2"] }, 2] },
      mod: { $mod: ["$marks2", "$marks1"] }
    }
  }
])
```

Program 6

Query MongoDB with Conditions on Database created with the employees details. Include columns like Employee name, salary, Location etc.

i. Find any record where the name is Tom.

ii. Find any record where the total payment amount is 40,000.

iii. Find any record where the price is greater than 40,000.

iv. Find any record where Note is null or the key itself is missing.

v. Find any record where Note exists and its value is null.

vi. Find any record where the Note key doesn't exist.

Step 1: Sample Document Structure

Assume your collection employees contains documents like:

json

CopyEdit

```
{
"Employee_name": "Tom",
"salary": 40000,
"Location": "New York",
"Note": null
}
```

MongoDB Queries with Conditions

i. Find any record where name is Tom

javascript

CopyEdit

```
db.employees.find({ Employee_name: "Tom" })
```

ii. Find any record where total payment amount is 40,000

Assuming salary is the field used for payment:

javascript

CopyEdit

```
db.employees.find({ salary: 40000 })
```

iii. Find any record where salary is greater than 40,000

javascript

CopyEdit

```
db.employees.find({ salary: { $gt: 40000 } })
```

iv. Find any record where Note is null OR key itself is missing

javascript

CopyEdit

```
db.employees.find({
$or: [
{ Note: null },
{ Note: { $exists: false } }
```

]
})
This matches documents where:
- Note: null
- **or Note field doesn't exist at all.**

v. Find any record where Note exists and value is null
javascript
CopyEdit
db.employees.find({
Note: null,
Note: { $exists: true }
})
This ensures:
- Field exists
- **Value is explicitly null**

vi. Find any record where the Note key doesn't exist
javascript
CopyEdit
db.employees.find({
Note: { $exists: false }
})
MongoDB Compass Instructions
- **Go to the employees collection.**
- **In the Filter bar, enter each query.**
- For example:

json
CopyEdit
{ "Employee_name": "Tom" }
- **Click Find.**