

# FULL STACK WITH MERN (JAVA) ASSIGNMENT-4

Ranga Teenamrutha  
Vignan's Nirula Institute of  
Technology and Science  
For Women (VNITSW)  
4<sup>th</sup> B-Tech (IT)  
20NN1A12B4

## ASSIGNMENT-4: Creating a Database Using MongoDB and Mongosh

### OBJECTIVE:

The objective of this assignment is to familiarize yourself with MongoDB and its command-line interface, Mongosh, and to understand how to create, manage, and query databases and collections in MongoDB.

### PROCESS

1. **DATABASE SETUP:** Create a new MongoDB database called myDatabase.



```
use myDatabase
```

➤ **Output:**

```
use myDatabase
switched to db myDatabase
db.createCollection("users")
{ ok: 1 }
```

2. **COLLECTION CREATION:** Create a collection named users within the myDatabase database.



```
db.createCollection("users")
```

➤ **Output:**

```
use myDatabase
switched to db myDatabase
db.createCollection("users")
{ ok: 1 }
```

3. **DOCUMENT INSERTION:** Insert at least three documents into the users collection, each representing a user with fields such as name, email, and age.

```
➤ db.users.insertMany([
  { name: "John Doe", email: "john@example.com", age: 25 },
  { name: "Sai", email: "sai@example.com", age: 35 },
  { name: "Alice Johnson", email: "alice@example.com", age: 30 },
],
{ name: "pavan", email: "pavan@example.com", age: 19 },
{ name: "lohi", email: "lohi@example.com", age: 20 },
{ name: "shubman gill", email: "shubman@example.com", age:
24 },
])
```

➤ **Output:**

```
db.users.insertMany([
  { name: "John Doe", email: "john@example.com", age: 25 },
  { name: "Sai", email: "sai@example.com", age: 35 },
  { name: "Alice Johnson", email: "alice@example.com", age: 30 },
  { name: "pavan", email: "pavan@example.com", age: 19 },
  { name: "lohi", email: "lohi@example.com", age: 20 },
  { name: "shubman gill", email: "shubman@example.com", age: 24 },
])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('65f95cf84594cd8c10686967'),
    '1': ObjectId('65f95cf84594cd8c10686968'),
    '2': ObjectId('65f95cf84594cd8c10686969'),
    '3': ObjectId('65f95cf84594cd8c1068696a'),
    '4': ObjectId('65f95cf84594cd8c1068696b'),
    '5': ObjectId('65f95cf84594cd8c1068696c')
  }
}
Database >
```

4. **QUERYING:** Write queries to retrieve: All users from the users collection.

```
➤ db.users.find()
```

➤ Output:

```
> db.users.find()
< {
  _id: ObjectId('65f952f64594cd8c1068695f'),
  name: 'John Doe',
  email: 'john@example.com',
  age: 25
}
{
  _id: ObjectId('65f952f64594cd8c10686960'),
  name: 'Sai',
  email: 'sai@example.com',
  age: 35
}
{
  _id: ObjectId('65f952f64594cd8c10686961'),
  name: 'Alice Johnson',
  email: 'alice@example.com',
  age: 30
}
{
  _id: ObjectId('65f953884594cd8c10686962'),
  name: 'pavan',
  email: 'pavan@example.com',
  age: 19
}
{
  _id: ObjectId('65f953884594cd8c10686963'),
  name: 'lohi',
  email: 'lohi@example.com',
  age: 20
}
{
  _id: ObjectId('65f953884594cd8c10686964'),
  name: 'shubman gill',
  email: 'shubman@example.com',
  age: 24
}
```

**5. RETRIVING USERS WHOSE AGE IS GREATER THAN 30:** Users with an age greater than or equal to 30.

➤ `db.users.find({ age: { $gte: 30 } })`

➤ **Output:**

```
db.users.find({ age: { $gte: 30 } })
{
  _id: ObjectId('65f952f64594cd8c10686960'),
  name: 'Sai',
  email: 'sai@example.com',
  age: 35
}
{
  _id: ObjectId('65f952f64594cd8c10686961'),
  name: 'Alice Johnson',
  email: 'alice@example.com',
  age: 30
}
```

**6. UPDATE OPERATION:** Update the age of a user with a specific email address.

➤ `db.users.updateOne(
 { email: "john@example.com" },
 { $set: { age: 28 } }
)`

➤ **Output:**

```
db.users.updateOne(
  { email: "john@example.com" },
  { $set: { age: 28 } }
)
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

**7. DELETION OPERATION:** Delete a user document based on a specific email address.

➤ `db.users.deleteOne({ email: "alice@example.com" })`

➤ **Output:**

```
> db.users.deleteOne({ email: "alice@example.com" })
{
  acknowledged: true,
  deletedCount: 1
}
```

**8. INDEX CREATION:** Create an index on the email field of the users collection.

➤ `db.users.createIndex({ email: 1 }, { unique: true })`

➤ **Output:**

```
> db.users.createIndex({ email: 1 }, { unique: true })

< email_1
myDatabase> |
```