Experiment – 7: MongoDB

Name of Student	Arnav Santosh Sawant
Class Roll No	D15A - 52
D.O.P.	20/03/25
D.O.S.	
Sign and Grade	

1) **Aim:** To study CRUD operations in MongoDB
2) **Problem Statement:**
    A) Create a new database to storage student details of IT dept(Name, Roll no, class name) and perform the following on the database
        a) Insert one student details
        b) Insert at once multiple student details
        c) Display student for a particular class
        d) Display students of specific roll no in a class
        e) Change the roll no of a student
        f) Delete entries of particular student

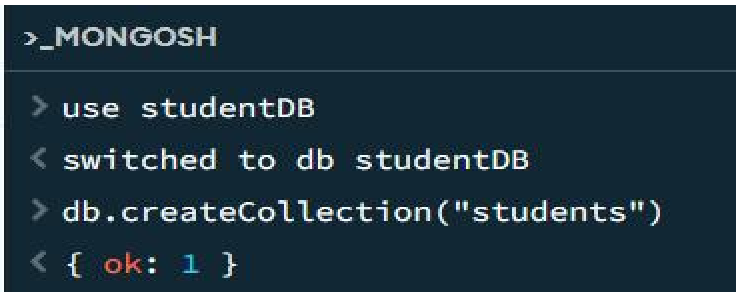
    B) Create a set of RESTful endpoints using Node.js, Express, and Mongoose for handling student data operations.
    The endpoints should support:
    - Retrieve a list of all students.
    - Retrieve details of an individual student by ID.
    - Add a new student to the database.
    - Update details of an existing student by ID.
    - Delete a student from the database by ID.

    Connect the server to MongoDB using Mongoose, and store student data with attributes: name, age, and grade.

3) **Output:**

A)Created a database called as students

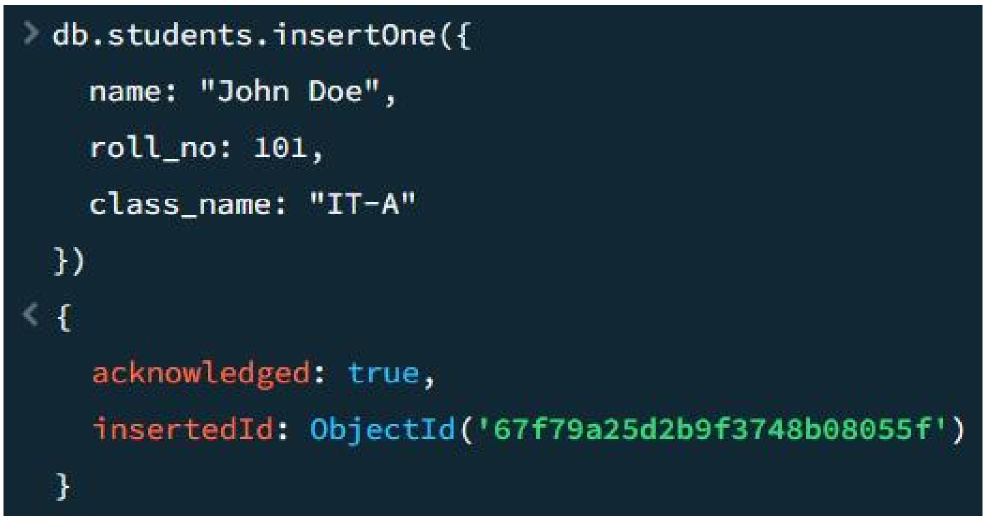
```
db.createCollection("students")
```



```
>_MONGOSH
> use studentDB
< switched to db studentDB
> db.createCollection("students")
< { ok: 1 }
```

a) Insert one student details

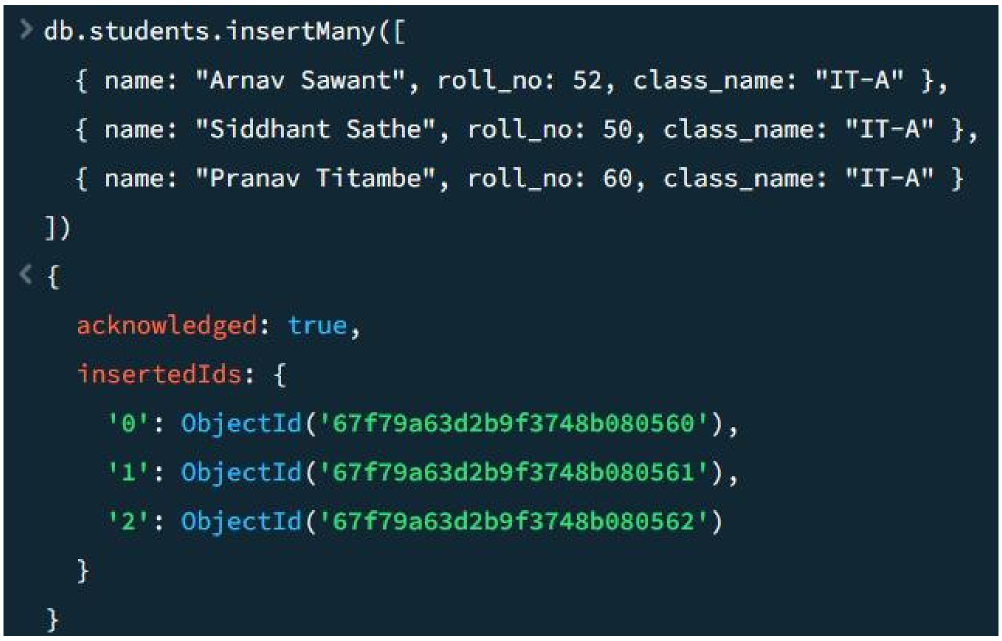
```
db.students.insertOne({
  name: "John Doe",
  roll_no: 101,
  class_name: "IT-A"
})
```



```
> db.students.insertOne({
    name: "John Doe",
    roll_no: 101,
    class_name: "IT-A"
  })
< {
    acknowledged: true,
    insertedId: ObjectId('67f79a25d2b9f3748b08055f')
  }
```

b) Insert many student details

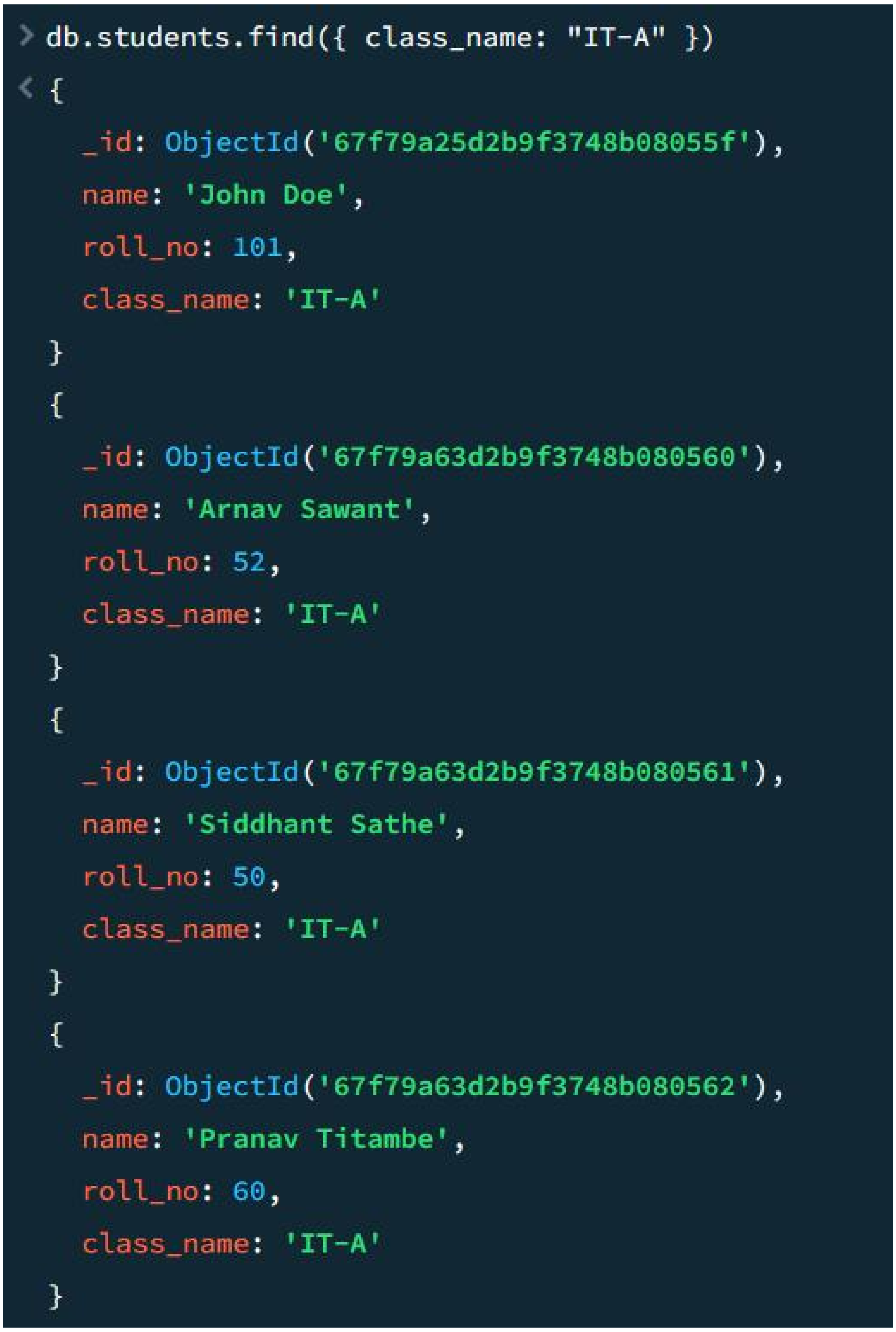
```
db.students.insertMany([
  { name: "Arnav Sawant", roll_no: 52, class_name: "IT-A" },
  { name: "Siddhant Sathe", roll_no: 50, class_name: "IT-A" },
  { name: "Pranav Titambe", roll_no: 60, class_name: "IT-A" }
])
```



```
> db.students.insertMany([
    { name: "Arnav Sawant", roll_no: 52, class_name: "IT-A" },
    { name: "Siddhant Sathe", roll_no: 50, class_name: "IT-A" },
    { name: "Pranav Titambe", roll_no: 60, class_name: "IT-A" }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('67f79a63d2b9f3748b080560'),
      '1': ObjectId('67f79a63d2b9f3748b080561'),
      '2': ObjectId('67f79a63d2b9f3748b080562')
    }
  }
```

c) Find the students based on class

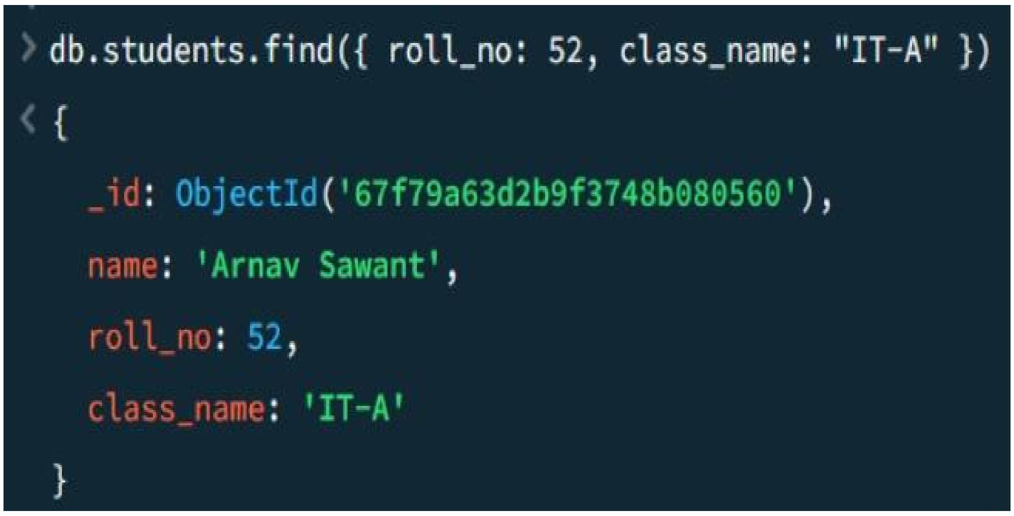
db.students.find({ class_name: "IT-A" })



```
> db.students.find({ class_name: "IT-A" })
< {
    _id: ObjectId('67f79a25d2b9f3748b08055f'),
    name: 'John Doe',
    roll_no: 101,
    class_name: 'IT-A'
  }
  {
    _id: ObjectId('67f79a63d2b9f3748b080560'),
    name: 'Arnav Sawant',
    roll_no: 52,
    class_name: 'IT-A'
  }
  {
    _id: ObjectId('67f79a63d2b9f3748b080561'),
    name: 'Siddhant Sathe',
    roll_no: 50,
    class_name: 'IT-A'
  }
  {
    _id: ObjectId('67f79a63d2b9f3748b080562'),
    name: 'Pranav Titambe',
    roll_no: 60,
    class_name: 'IT-A'
  }
```

d) Display students specific roll no in class

db.students.find({ roll_no: 52, class_name: "IT-A" })



```
> db.students.find({ roll_no: 52, class_name: "IT-A" })
< {
    _id: ObjectId('67f79a63d2b9f3748b080560'),
    name: 'Arnav Sawant',
    roll_no: 52,
    class_name: 'IT-A'
  }
```

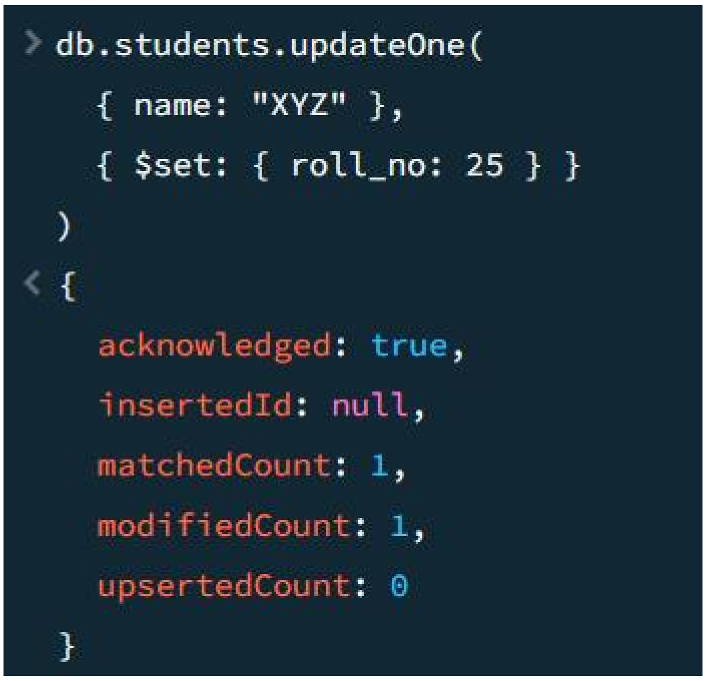
e) Change roll no of the student

db.students.updateOne(

{ name: "XYZ" },

{ $set: { roll_no: 25 } }

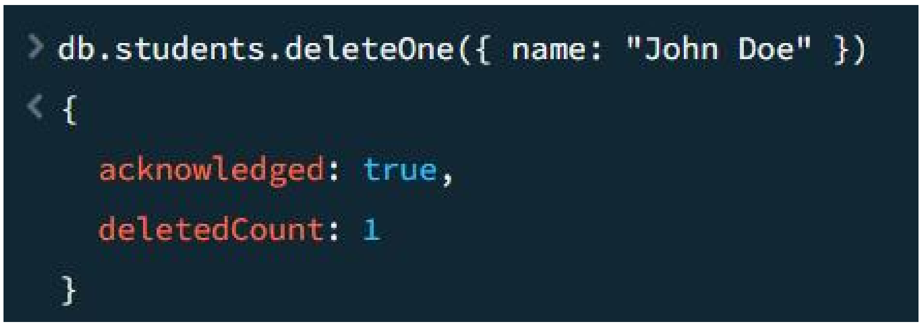
)



```
> db.students.updateOne(
    { name: "XYZ" },
    { $set: { roll_no: 25 } }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

f) Delete entries of particular student

db.students.deleteOne({ name: "John Doe" })



```
> db.students.deleteOne({ name: "John Doe" })
< {
    acknowledged: true,
    deletedCount: 1
  }
```

B) Creating a set of restful endpoints

Creating the models

models/student.js

```
const mongoose = require('mongoose');
const studentSchema = new mongoose.Schema({
  name: { type: String, required: true },
  age: { type: Number, required: true },
  grade: { type: String, required: true }
});
module.exports = mongoose.model('Student', studentSchema);
```

server.js

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const Student = require('./models/student');

const app = express();
app.use(bodyParser.json());

// Connect to MongoDB
mongoose.connect('mongodb://127.0.0.1:27017/studentDB', {
  useNewUrlParser: true,
  useUnifiedTopology: true
})
.then(() => console.log('Connected to MongoDB'))
.catch(err => console.error('MongoDB connection error:', err));

// Get all students
app.get('/students', async (req, res) => {
  const students = await Student.find();
  res.json(students);
});

// Get student by ID
app.get('/students/:id', async (req, res) => {
  try {
    const student = await Student.findById(req.params.id);
    if (!student) return res.status(404).send('Student not found');
    res.json(student);
  } catch (err) {
    res.status(400).send('Invalid ID');
  }
});

// Add new student
app.post('/students', async (req, res) => {
  try {
    const { name, age, grade } = req.body;
    const newStudent = new Student({ name, age, grade });
    await newStudent.save();
    res.status(201).json(newStudent);
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
});

// Update student by ID
app.put('/students/:id', async (req, res) => {
  try {
    const updatedStudent = await Student.findByIdAndUpdate(
      req.params.id,
      req.body,
      { new: true }
    );
    if (!updatedStudent) return res.status(404).send('Student not found');
    res.json(updatedStudent);
  } catch (err) {
    res.status(400).send('Invalid ID');
  }
});

// Delete student by ID
app.delete('/students/:id', async (req, res) => {
  try {
    const result = await Student.findByIdAndDelete(req.params.id);
    if (!result) return res.status(404).send('Student not found');
    res.send('Student deleted');
  } catch (err) {
    res.status(400).send('Invalid ID');
  }
```

```
});



// we are then starting the server

const PORT = 3000;

app.listen(PORT, () => {

  console.log(`Server running on http://localhost:${PORT}`);

});
```