# EXPERIMENT NO. 7 - MongoDB

| | |
|---|---|
| **Name of Student** | **Anuprita Mhapankar** |
| **Class Roll No** | **28** |
| **D.O.P.** | **13/03/2025** |
| **D.O.S.** | **20/03/2025** |
| **Sign and Grade** | |

**AIM : To study CRUD operations in MongoDB**

## OVERVIEW OF TASKS PERFORMED :

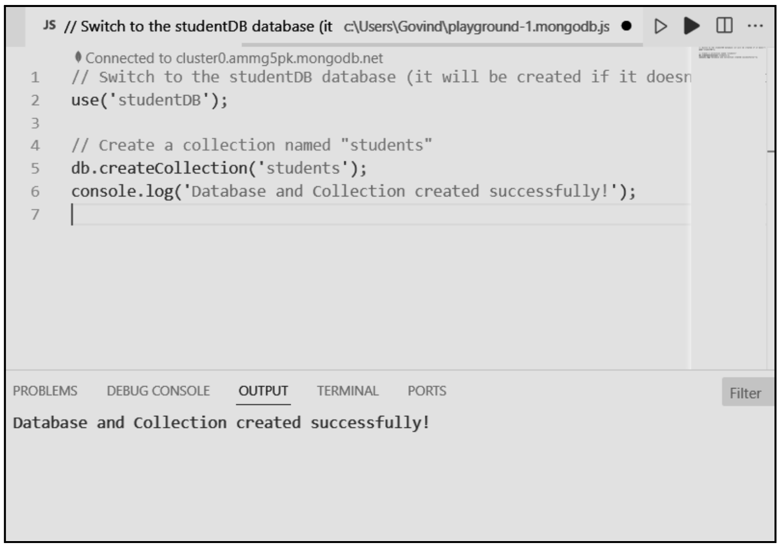
The experiment involves creating a **student database** for the IT department with fields **Name, Roll No, and Class Name**. A single student record was inserted, followed by multiple student entries at once. Queries were performed to **filter students by class**, retrieve students with a **specific roll number**, update a student's roll number, and delete a specific student's entry.

Additionally, **RESTful APIs** were implemented using **Node.js, Express, and Mongoose** to manage student data. The server was connected to **MongoDB**, and endpoints were created to **retrieve all students, get details of a student by ID, add a new student, update student details, and delete a student by ID**. The student schema included attributes **name, age, and grade** for data storage.

**GITHUB LINK -** https://github.com/Anuprita2022-26/WebX_Exp7

## OUTPUT

1. **Create a database to store student details of IT Department**

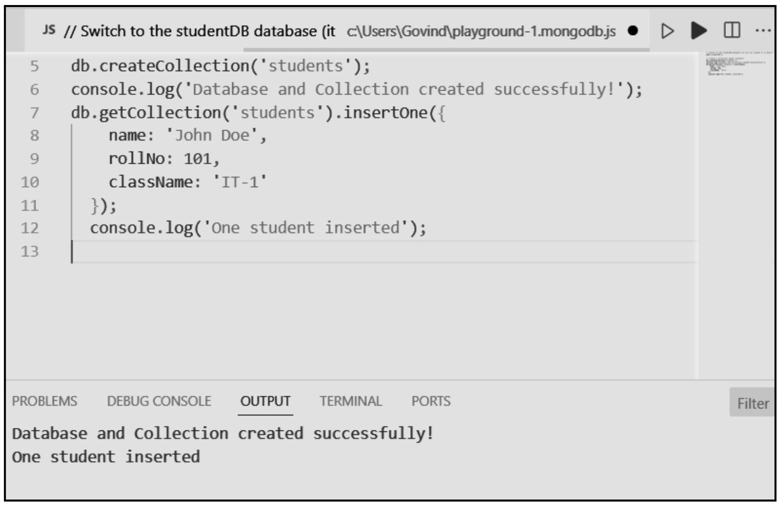


```
// Switch to the studentDB database (it   c:\Users\Govind\playground-1.mongodb.js
Connected to cluster0.ammg5pk.mongodb.net
// Switch to the studentDB database (it will be created if it doesn
use('studentDB');

// Create a collection named "students"
db.createCollection('students');
console.log('Database and Collection created successfully!');
```

```
Database and Collection created successfully!
```

## a) Insert Single & multiple student records

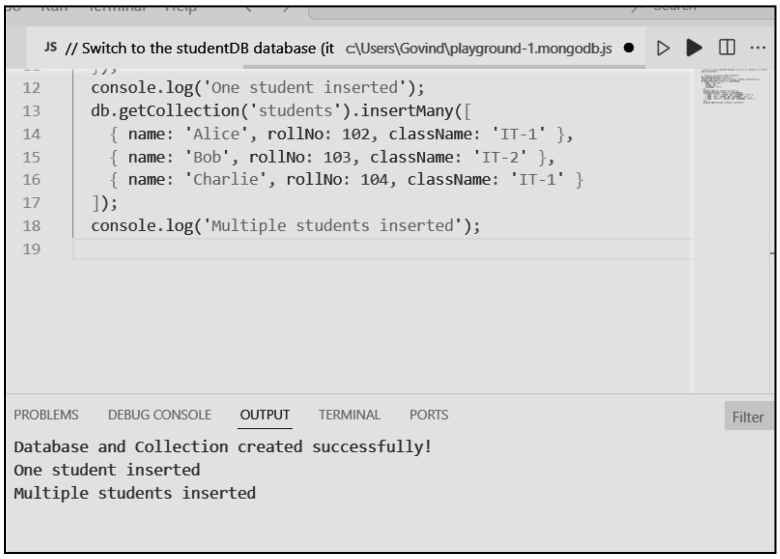


JS // Switch to the studentDB database (it c:\Users\Govind\playground-1.mongodb.js

```
db.createCollection('students');
console.log('Database and Collection created successfully!');
db.getCollection('students').insertOne({
    name: 'John Doe',
    rollNo: 101,
    className: 'IT-1'
  });
  console.log('One student inserted');
```

PROBLEMS
DEBUG CONSOLE
OUTPUT
TERMINAL
PORTS
Filter

```
Database and Collection created successfully!
One student inserted
```



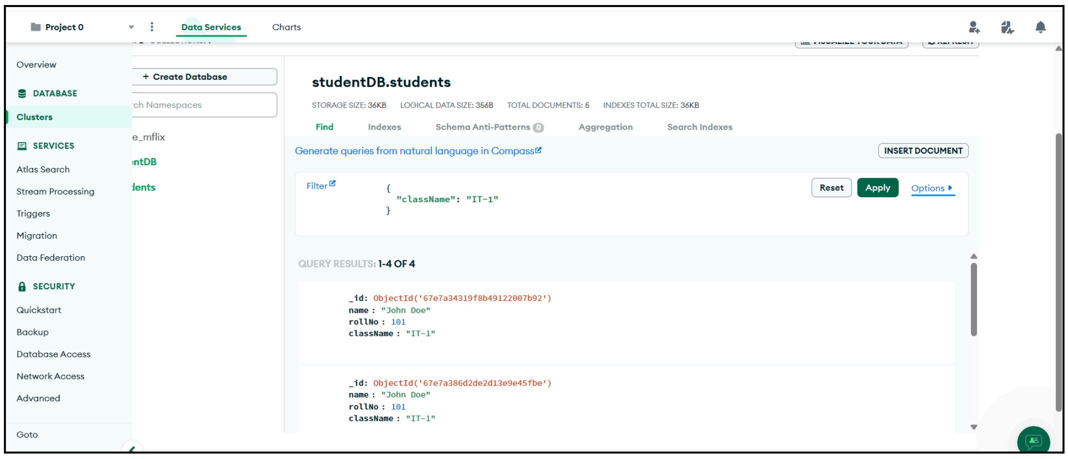
JS // Switch to the studentDB database (it c:\Users\Govind\playground-1.mongodb.js

```
  console.log('One student inserted');
  db.getCollection('students').insertMany([
    { name: 'Alice', rollNo: 102, className: 'IT-1' },
    { name: 'Bob', rollNo: 103, className: 'IT-2' },
    { name: 'Charlie', rollNo: 104, className: 'IT-1' }
  ]);
  console.log('Multiple students inserted');
```

PROBLEMS
DEBUG CONSOLE
OUTPUT
TERMINAL
PORTS
Filter

```
Database and Collection created successfully!
One student inserted
Multiple students inserted
```

## b) Display Students for a Particular Class



studentDB.students

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 356B TOTAL DOCUMENTS: 5 INDEXES TOTAL SIZE: 36KB

Filter

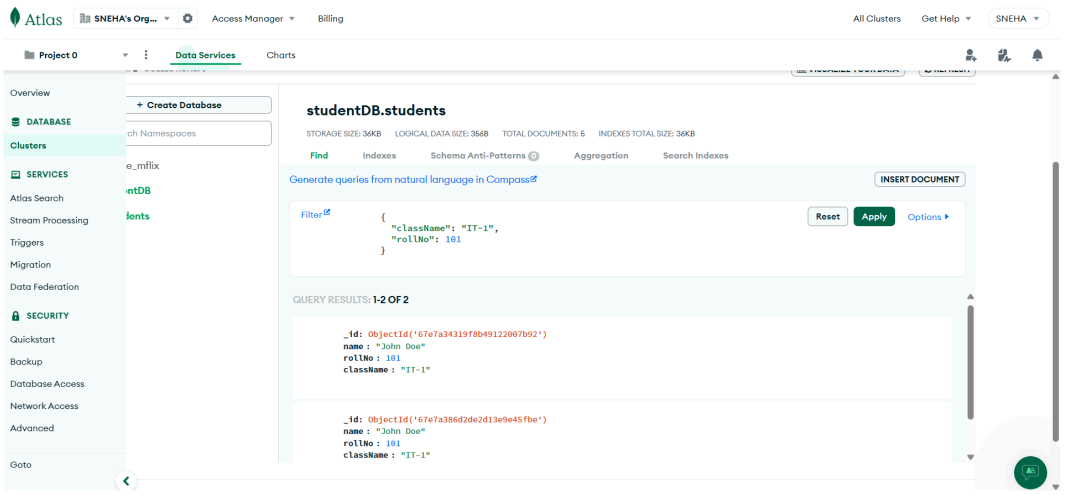
```
{
  "className": "IT-1"
}
```

QUERY RESULTS: 1-4 OF 4

```
_id: ObjectId('67e7a34319f8b49122007b92')
name : "John Doe"
rollNo : 101
className : "IT-1"

_id: ObjectId('67e7a386d2de2d13e9e45fbe')
name : "John Doe"
rollNo : 101
className : "IT-1"
```

## c) Display Student of a Specific Roll Number in a Class


studentDB.students

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 356B TOTAL DOCUMENTS: 5 INDEXES TOTAL SIZE: 36KB

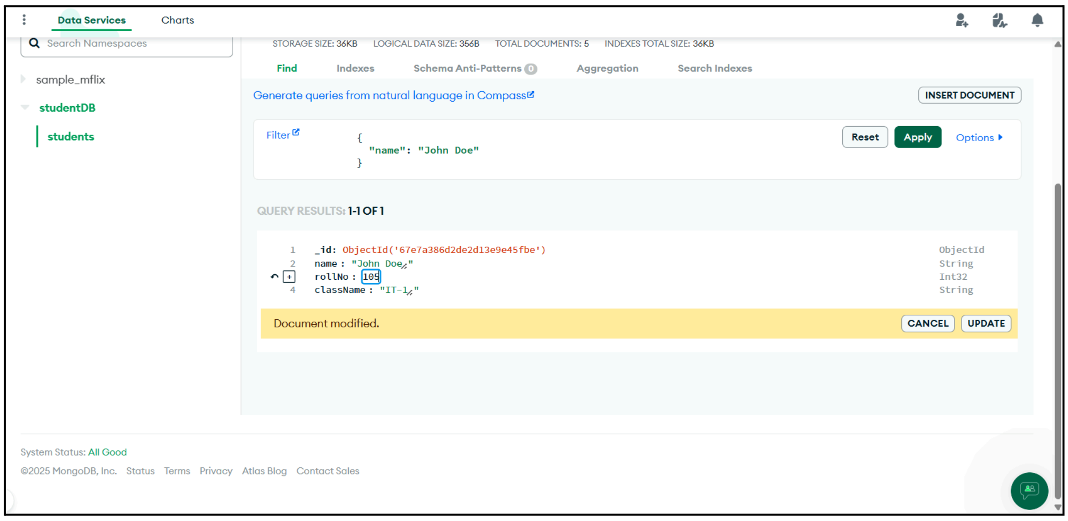
Filter
```
{
  "className": "IT-1",
  "rollNo": 101
}
```

QUERY RESULTS: 1-2 OF 2

```
_id: ObjectId('67e7a34319f8b49122007b92')
name : "John Doe"
rollNo : 101
className : "IT-1"

_id: ObjectId('67e7a386d2de2d13e9e45fbe')
name : "John Doe"
rollNo : 101
className : "IT-1"
```

## d) Change the Roll Number of a Student


STORAGE SIZE: 36KB LOGICAL DATA SIZE: 356B TOTAL DOCUMENTS: 5 INDEXES TOTAL SIZE: 36KB

Filter
```
{
  "name": "John Doe"
}
```

QUERY RESULTS: 1-1 OF 1

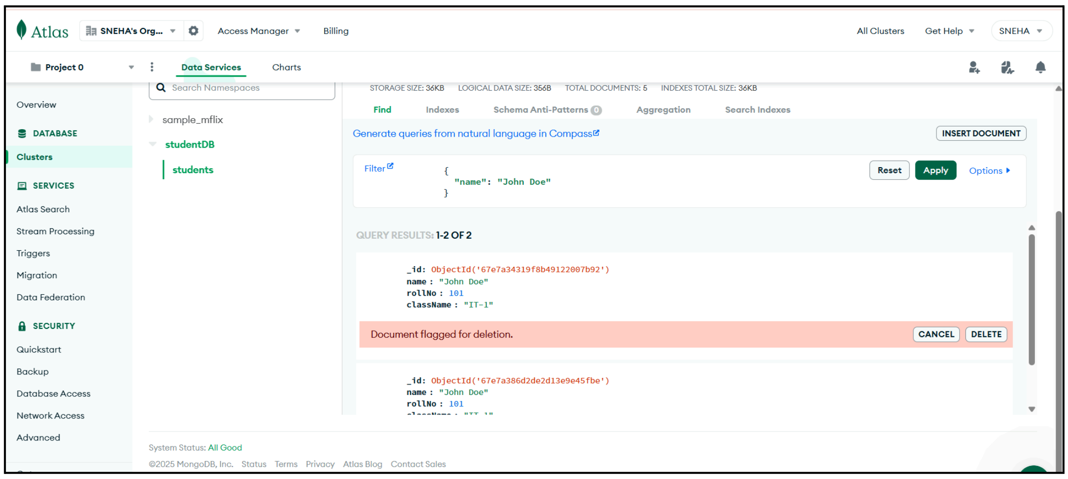
```
_id: ObjectId('67e7a386d2de2d13e9e45fbe')     ObjectId
name : "John Doe"                             String
rollNo : 105                                  Int32
className : "IT-1"                            String
```

Document modified.

System Status: All Good

©2025 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

## e) Delete Entries of a Particular Student


STORAGE SIZE: 36KB LOGICAL DATA SIZE: 356B TOTAL DOCUMENTS: 5 INDEXES TOTAL SIZE: 36KB

Filter
```
{
  "name": "John Doe"
}
```

QUERY RESULTS: 1-2 OF 2

```
_id: ObjectId('67e7a34319f8b49122007b92')
name : "John Doe"
rollNo : 101
className : "IT-1"
```

Document flagged for deletion.

```
_id: ObjectId('67e7a386d2de2d13e9e45fbe')
name : "John Doe"
rollNo : 101
```

System Status: All Good

©2025 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

## CONCLUSION

In this experiment, we successfully performed CRUD operations in **MongoDB** and implemented a **RESTful API** using **Node.js, Express, and Mongoose**. We learned how to create, read, update, and delete student records both via **MongoDB shell commands** and **API endpoints**.