
📜 Experiment No. 1 – MongoDB CRUD Operations

🎯 Aim:

To perform **Create, Read, Update, and Delete (CRUD)** operations using MongoDB.

🧰 Software Requirements:

- **Operating System:** Windows / Ubuntu
- **Database:** MongoDB
- **Tool:** Mongo Shell or MongoDB Compass

⚙️ Commands and Outputs

🧩 Step 1: Create / Use Database

use prexam

🧩 Step 2: Create Collection

db.createCollection("student")

🧩 Step 3: Insert Documents

Insert One Document

db.student.insertOne({ Rollno:101, name:"Samruddhi" })

Insert Many Documents

db.student.insertMany([

{ Rollno:102, name:"Sakshi" },

{ Rollno:103, name:"Rutuja" }

])

🧩 Step 4: Display All Documents

```
db.student.find().pretty()
```

🧩 Step 5: Update Operations

a) Update one field

```
db.student.updateOne({ Rollno:102 }, { $set:{ name:"Sakshi Yadav" } })
```

b) Update and increment

```
db.student.updateOne(
 { Rollno:101 },
 { $set:{ cgpa:9.21 }, $inc:{ age:1 } }
)
```

c) Decrease a value

```
db.student.updateOne(
 { Rollno:101 },
 { $inc:{ age:-1 } }
)
```

🧩 Step 6: Replace Entire Document

```
db.student.replaceOne(
 { Rollno:101 },
 {
   Rollno:101,
   name:"Samruddhi Yadav",
   age:20,
   department:"IT",
   subject:["ADBMS","OS","HCI","ML"],
   cgpa:7.91
```

}

)

🧩 Step 7: Delete Operations

Delete One

db.student.deleteOne({ Rollno:103 })

Delete Many

db.student.deleteMany({ department:"Computer" })

🧩 Step 8: Comparison Query

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

🧩 Step 9: Sorting and Limiting

Sort in descending order

db.student.find().sort({ cgpa:-1 })

Sort in ascending order

db.student.find().sort({ cgpa:1 })

Limit output

db.student.find().limit(1)

🧩 Step 10: Logical Queries

$or

db.student.find({ $or:[{ department:"IT" },{ age:20 }] })

$and

db.student.find({ $and:[{ age:{ $gt:18 } },{ cgpa:{ $lt:9.5 } }] })

$not

db.student.find({ cgpa:{ $not:{ $gt:9 } } })

🧩 Step 11: Array Query

```
db.student.find({ subject:"OS" })
```

🧩 Step 12: Regex Queries

Starts with 'S'

```
db.student.find({ name:/^S/ })
```

Ends with 'v'

```
db.student.find({ name:/v$/ })
```

Case-insensitive

```
db.student.find({ name:/^s/i })
```

🧩 Step 13: Additional Queries (Bonus)

a) Projection (Select specific fields)

```
db.student.find({}, { Rollno:1, name:1, _id:0 })
```

b) Count documents

```
db.student.countDocuments()
```

c) Distinct departments

```
db.student.distinct("department")
```

d) $in Operator

```
db.student.find({ department:{ $in:["IT","Computer"] } })
```

e) $exists Operator

```
db.student.find({ cgpa:{ $exists:true } })
```

f) Aggregation – Average CGPA

```
db.student.aggregate([
 { $group:{ _id:null, avg_cgpa:{ $avg:"$cgpa" } } }
])
```

g) CGPA between 8 and 9

```
db.student.find({ cgpa:{ $gte:8, $lte:9 } })
```

🧩 Step 14: Field Modification Examples

Unset a field

```
db.student.updateOne({ Rollno:101 }, { $unset:{ department:"" } })
```

Rename a field

```
db.student.updateOne({ Rollno:101 }, { $rename:{ "cgpa":"GPA" } })
```

Upsert (Insert if not exists)

```
db.student.updateOne(
 { Rollno:104 },
 { $set:{ name:"Neha", age:21, department:"IT", GPA:8.4 } },
 { upsert:true }
)
```

🧩 Step 15: Sample Output


```
[
 {
  "_id": ObjectId("..."),
  "Rollno": 101,
  "name": "Samruddhi Yadav",
  "age": 20,
  "department": "IT",
  "subject": ["ADBMS","OS","HCI","ML"],
  "cgpa": 9.21
 },
 {
  "_id": ObjectId("..."),
  "Rollno": 102,
  "name": "Sakshi Yadav",
```

```
  "age": 22,
  "department": "Computer",
  "subject": ["DBMS","OS","CMS","ML"],
  "cgpa": 8.91
 }
]
```