

## MongoDB Practice

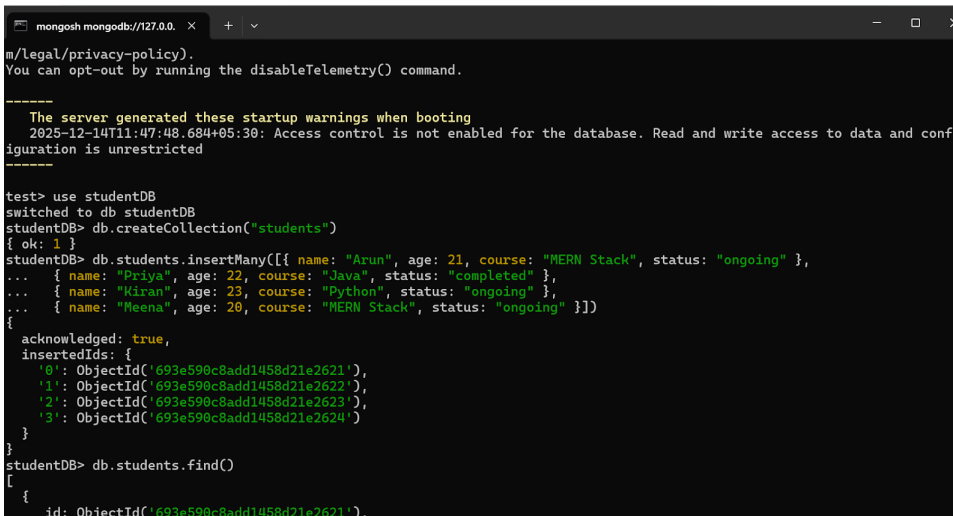
This document explains MongoDB basics, CRUD operations, query operators, array operators, and a real-world use case using a student database. It is suitable for practice, viva, and interviews.

### 1. Introduction to MongoDB

MongoDB is a NoSQL, document-oriented database that stores data in flexible, JSON-like documents. It is schema-less, scalable, and widely used in modern web applications.

### 2. Database and Collection

A database in MongoDB is a container for collections. A collection is a group of documents. Documents store data as key-value pairs similar to JSON objects.

A screenshot of a MongoDB terminal window. The window title is 'mongosh mongodb://127.0.0.1'. The terminal shows the following commands and output:

```
m/legat/privacy-policy).
You can opt-out by running the disableTelemetry() command.

-----
The server generated these startup warnings when booting
2025-12-14T11:47:48.684+05:30: Access control is not enabled for the database. Read and write access to data and conf
figuration is unrestricted
-----

test> use studentDB
switched to db studentDB
studentDB> db.createCollection("students")
{ ok: 1 }
studentDB> db.students.insertMany([
  { name: "Arun", age: 21, course: "MERN Stack", status: "ongoing" },
  { name: "Priya", age: 22, course: "Java", status: "completed" },
  { name: "Kiran", age: 23, course: "Python", status: "ongoing" },
  { name: "Meena", age: 20, course: "MERN Stack", status: "ongoing" }
])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693e590c8add1458d21e2621'),
    '1': ObjectId('693e590c8add1458d21e2622'),
    '2': ObjectId('693e590c8add1458d21e2623'),
    '3': ObjectId('693e590c8add1458d21e2624')
  }
}
studentDB> db.students.find()
[
  {
    _id: ObjectId('693e590c8add1458d21e2621'),
```

### 3. Insert Operations

Insert operations are used to add new documents into a collection. MongoDB supports inserting a single document or multiple documents at once.

## 4. Read Operations

Read operations are used to fetch data from the database. The `find()` method retrieves documents from a collection and supports filters to get specific records.

```
mongosh mongodb://127.0.0.1:27020/
> use studentDB
studentDB> db.students.find()
[
  {
    _id: ObjectId('693e590c8add1458d21e2621'),
    name: 'Arun',
    age: 21,
    course: 'MERN Stack',
    status: 'ongoing'
  },
  {
    _id: ObjectId('693e590c8add1458d21e2622'),
    name: 'Priya',
    age: 22,
    course: 'Java',
    status: 'completed'
  },
  {
    _id: ObjectId('693e590c8add1458d21e2623'),
    name: 'Kiran',
    age: 23,
    course: 'Python',
    status: 'ongoing'
  },
  {
    _id: ObjectId('693e590c8add1458d21e2624'),
    name: 'Meena',
    age: 20,
    course: 'MERN Stack',
    status: 'ongoing'
  }
]
```

## 5. Update Operations

Update operations modify existing documents. MongoDB allows updating one document or many documents using `updateOne()` and `updateMany()` methods.

```
studentDB> db.students.find({ course: "MERN Stack" })
[
  {
    _id: ObjectId('693e590c8add1458d21e2621'),
    name: 'Arun',
    age: 21,
    course: 'MERN Stack',
    status: 'ongoing'
  },
  {
    _id: ObjectId('693e590c8add1458d21e2624'),
    name: 'Meena',
    age: 20,
    course: 'MERN Stack',
    status: 'ongoing'
  }
]
studentDB> db.students.updateOne(
...   { name: "Meena" },
...   { $set: { status: "completed" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

## 6. Delete Operations

Delete operations remove documents from a collection. MongoDB supports deleting a single document or multiple documents based on conditions.

```
mongosh mongodb://127.0.0.1:27017
> use studentDB
studentDB> db.students.deleteOne({ name: "Kiran" })
{ acknowledged: true, deletedCount: 1 }
studentDB> db.students.find()
[
  {
    _id: ObjectId('693e590c8add1458d21e2621'),
    name: 'Arun',
    age: 21,
    course: 'MERN Stack',
    status: 'ongoing'
  },
  {
    _id: ObjectId('693e590c8add1458d21e2622'),
    name: 'Priya',
    age: 22,
    course: 'Java',
    status: 'completed'
  },
  {
    _id: ObjectId('693e590c8add1458d21e2624'),
    name: 'Meena',
    age: 20,
    course: 'MERN Stack',
    status: 'completed'
  }
]
```

## 7. Query Operators

Query operators are used to filter documents based on conditions. They include comparison operators like \$gt, \$lt, logical operators like \$and, \$or, and element operators like \$exists.

### Comparison Operators

```
mongosh mongodb://127.0.0.1:27017
studentDB> db.students.find({ age: { $eq: 21 } })
[
  {
    _id: ObjectId('693e590c8add1458d21e2621'),
    name: 'Arun',
    age: 21,
    course: 'MERN Stack',
    status: 'ongoing'
  }
]
studentDB> db.students.find({ status: { $ne: "completed" } })
[
  {
    _id: ObjectId('693e590c8add1458d21e2621'),
    name: 'Arun',
    age: 21,
    course: 'MERN Stack',
    status: 'ongoing'
  }
]
studentDB> db.students.find({ age: { $gt: 21 } })
[
  {
    _id: ObjectId('693e590c8add1458d21e2622'),
    name: 'Priya',
    age: 22,
    course: 'Java',
    status: 'completed'
  }
]
```

```
mongosh mongodb://127.0.0.1
]
studentDB> db.students.find({ age: { $gte: 21 } })
[
  {
    _id: ObjectId('693e590c8add1458d21e2621'),
    name: 'Arun',
    age: 21,
    course: 'MERN Stack',
    status: 'ongoing'
  },
  {
    _id: ObjectId('693e590c8add1458d21e2622'),
    name: 'Priya',
    age: 22,
    course: 'Java',
    status: 'completed'
  }
]
studentDB> db.students.find({ age: { $lt: 22 } })
[
  {
    _id: ObjectId('693e590c8add1458d21e2621'),
    name: 'Arun',
    age: 21,
    course: 'MERN Stack',
    status: 'ongoing'
  },
  {
    _id: ObjectId('693e590c8add1458d21e2624'),
    name: 'Meena',
    age: 20,
    course: 'MERN Stack',
    status: 'completed'
  }
]
```

## Logical Operators

```
mongosh mongodb://127.0.0.1
studentDB> db.students.find({
...   $and: [
...     { course: "MERN Stack" },
...     { status: "completed" }
...   ]
... })
[
  {
    _id: ObjectId('693e590c8add1458d21e2624'),
    name: 'Meena',
    age: 20,
    course: 'MERN Stack',
    status: 'completed'
  }
]
studentDB> db.students.find({
...   $or: [
...     { course: "Java" },
...     { age: { $gt: 22 } }
...   ]
... })
[
  {
    _id: ObjectId('693e590c8add1458d21e2622'),
    name: 'Priya',
    age: 22,
    course: 'Java',
    status: 'completed'
  }
]
```

```
mongosh mongodb://127.0.0.1
studentDB> db.students.find({
...   $nor: [
...     { course: "Java" },
...     { status: "completed" }
...   ]
... })
[
  {
    _id: ObjectId('693e590c8add1458d21e2621'),
    name: 'Arun',
    age: 21,
    course: 'MERN Stack',
    status: 'ongoing'
  }
]
studentDB> db.students.find({ age: { $not: { $gt: 22 } } })
[
  {
    _id: ObjectId('693e590c8add1458d21e2621'),
    name: 'Arun',
    age: 21,
    course: 'MERN Stack',
    status: 'ongoing'
  },
  {
    _id: ObjectId('693e590c8add1458d21e2622'),
    name: 'Priya',
    age: 22,
    course: 'Java',
    status: 'completed'
  },
  {
    _id: ObjectId('693e590c8add1458d21e2624'),
    name: 'Meena',
    age: 20,
    course: 'MERN Stack',
    status: 'completed'
  }
]
```

## 8. Array Operators

Array operators work with fields that contain arrays. They are useful when a document stores multiple values such as student skills.

```
studentDB> db.students.find({
...   skills: { $all: ["HTML", "CSS"] }
... })
...
[
  {
    _id: ObjectId('693e67c18add1458d21e2625'),
    name: 'Ravi',
    age: 22,
    course: 'MERN Stack',
    status: 'ongoing',
    skills: [ 'HTML', 'CSS', 'React' ]
  }
]
studentDB> db.students.find({
...   skills: { $in: ["React", "Node.js"] }
... })
...
[
  {
    _id: ObjectId('693e67c18add1458d21e2625'),
    name: 'Ravi',
    age: 22,
    course: 'MERN Stack',
    status: 'ongoing',
    skills: [ 'HTML', 'CSS', 'React' ]
  }
]
studentDB> db.students.find({
...   skills: { $size: 3 }
... })
...
[
  {
    _id: ObjectId('693e67c18add1458d21e2625'),
```

```
    age: 22,
    course: 'MERN Stack',
    status: 'ongoing',
    skills: [ 'HTML', 'CSS', 'React' ]
  }
]
studentDB> db.students.find({
...   skills: { $in: ["React", "Node.js"] }
... })
...
[
  {
    _id: ObjectId('693e67c18add1458d21e2625'),
    name: 'Ravi',
    age: 22,
    course: 'MERN Stack',
    status: 'ongoing',
    skills: [ 'HTML', 'CSS', 'React' ]
  }
]
studentDB> db.students.find({
...   skills: { $size: 3 }
... })
...
[
  {
    _id: ObjectId('693e67c18add1458d21e2625'),
    name: 'Ravi',
    age: 22,
    course: 'MERN Stack',
    status: 'ongoing',
    skills: [ 'HTML', 'CSS', 'React' ]
  }
]
studentDB> |
```

## 9. StudentDB Use Case

In the StudentDB use case, MongoDB is used to manage student details such as name, age, course, status, and skills. CRUD operations help in adding, updating, searching, and deleting student records.

```
]
studentDB> db.students.find({ course: "MERN Stack" })
[
  {
    _id: ObjectId('693e590c8add1458d21e2621'),
    name: 'Arun',
    age: 21,
    course: 'MERN Stack',
    status: 'ongoing'
  },
  {
    _id: ObjectId('693e590c8add1458d21e2624'),
    name: 'Meena',
    age: 20,
    course: 'MERN Stack',
    status: 'ongoing'
  }
]
studentDB> db.students.updateOne(
...   { name: "Meena" },
...   { $set: { status: "completed" } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}

studentDB> db.students.find({ name: { $regex: "^M" } })
[
  {
    _id: ObjectId('693e590c8add1458d21e2624'),
    name: 'Meena',
    age: 20,
    course: 'MERN Stack',
    status: 'completed'
  }
]
studentDB> db.students.find({ $expr: { $gt: ["$age", 21] } })
[
  {
    _id: ObjectId('693e590c8add1458d21e2622'),
    name: 'Priya',
    age: 22,
    course: 'Java',
    status: 'completed'
  }
]
studentDB> db.students.find(
...   { course: "MERN Stack" },
...   { name: 1, status: 1, _id: 0 }
... )
...
[
  { name: 'Arun', status: 'ongoing' },
  { name: 'Meena', status: 'completed' }
]
studentDB> |
```

## 10. Conclusion

By practicing MongoDB operations and operators, we can efficiently manage data in real-world applications. MongoDB provides flexibility, scalability, and powerful querying capabilities.