Key MongoDB Concepts

➢ Database & Collections: Like folders in a digital library.

➢ Documents: JSON objects – flexible and nestable.

➢ CRUD Operations: Create, Read, Update, Delete.

➢ Operators: Powerful filters for smart searches.

Step	Action	Command
1	**Switch to/Create DB**	use studentDB
2	**Create Collection** (Optional, done implicitly on insert)	db.createCollection("students")
3	**Verify Database**	db
4	**Verify Collection**	show collections

```
test> use studentDB
switched to db studentDB
studentDB> db.createCollection("students")
{ ok: 1 }
studentDB> db
studentDB
studentDB> show collections
students
```

Created a new database (studentDB) and a collection (students).

2. Insert Operations: Insert one document; insert multiple documents.

Step	Action	Command
1	Insert One Document	db.students.insertOne({ name: "Arun kumar", age: 24, course: "Data Science", status: "In Progress" })
2	Insert Multiple Documents	db.students.insertMany([{ name: "ajith", age: 22, course: "MERN Stack", status: "In Progress" }, { name: "Priya", age: 25, course: "MERN Stack", status: "Completed" }, { name: "Diana ", age: 28, course: "Cybersecurity", status: "In Progress" }])
3	Check Total Count	db.students.countDocuments({})

```
studentDB> db.students.insertOne({ name: "Arun kumar", age: 24, course: "Data Science", status: "In Progress" })
{
  acknowledged: true,
  insertedId: ObjectId('693ecb0417f57114841e2621')
}
```

Used insertOne

```
studentDB> db.students.insertMany([ { name: "ajith", age: 22, course: "MERN Stack", status: "In Progress" }, { name: "Priya", age: 25, course: "MERN Stack",
 status: "Completed" }, { name: "Diana", age: 28, course: "Cybersecurity", status: "In Progress" } ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693ecb7617f57114841e2622'),
    '1': ObjectId('693ecb7617f57114841e2623'),
    '2': ObjectId('693ecb7617f57114841e2624')
  }
}
```

Used insertMany

```
studentDB> db.students.countDocuments({})
4
```

Used countDocuments to find the number of documents

3. Read Operations (Find): Fetch all documents; fetch specific data (e.g., students in “MERN Stack”).

Step	Action	Command
1	Fetch All Documents	db.students.find()
2	Fetch All (Formatted)	db.students.find().pretty()
3	Filter by Course (MERN Stack)	db.students.find({ course: "MERN Stack" })
4	Filter and Project (Show only name and age)	db.students.find({ course: "MERN Stack" }, { name: 1, age: 1, _id: 0 })

```
studentDB> db.students.find({ course: "MERN Stack" })
[
  {
    _id: ObjectId('693ecb7617f57114841e2622'),
    name: 'ajith',
    age: 22,
    course: 'MERN Stack',
    status: 'In Progress'
  },
  {
    _id: ObjectId('693ecb7617f57114841e2623'),
    name: 'Priya',
    age: 25,
    course: 'MERN Stack',
    status: 'Completed'
  }
]
```

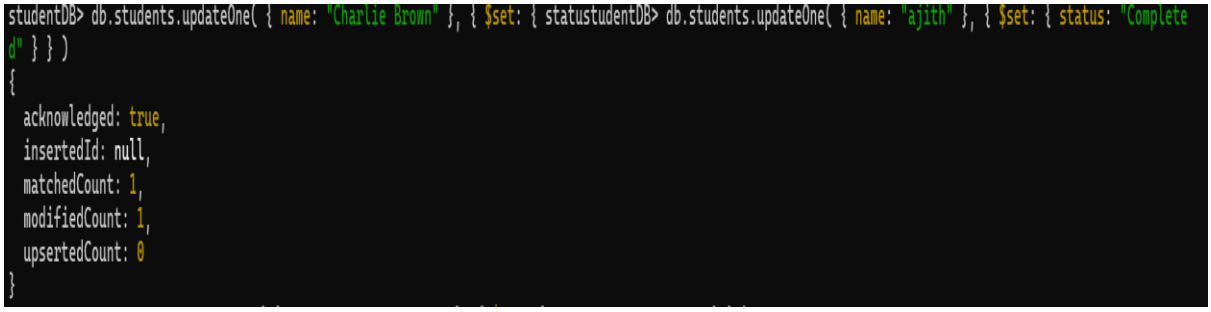
```
studentDB> db.students.find()
[
  {
    _id: ObjectId('693ecb0417f57114841e2621'),
    name: 'Arun kumar',
    age: 24,
    course: 'Data Science',
    status: 'In Progress'
  },
  {
    _id: ObjectId('693ecb7617f57114841e2622'),
    name: 'ajith',
    age: 22,
    course: 'MERN Stack',
    status: 'In Progress'
  },
  {
    _id: ObjectId('693ecb7617f57114841e2623'),
    name: 'Priya',
    age: 25,
    course: 'MERN Stack',
    status: 'Completed'
  },
  {
    _id: ObjectId('693ecb7617f57114841e2624'),
    name: 'Diana',
    age: 28,
    course: 'Cybersecurity',
    status: 'In Progress'
  }
]
```

```
studentDB>
(To exit, press Ctrl+C again or Ctrl+D or type .exit)
studentDB> db.students.find({ course: "MERN Stack" }, { name: 1, age: 1, _id: 0 })
[ { name: 'ajith', age: 22 }, { name: 'Priya', age: 25 } ]
```

```
studentDB> db.students.find().pretty()
[
  {
    _id: ObjectId('693ecb0417f57114841e2621'),
    name: 'Arun kumar',
    age: 24,
    course: 'Data Science',
    status: 'In Progress'
  },
  {
    _id: ObjectId('693ecb7617f57114841e2622'),
    name: 'ajith',
    age: 22,
    course: 'MERN Stack',
    status: 'In Progress'
  },
  {
    _id: ObjectId('693ecb7617f57114841e2623'),
    name: 'Priya',
    age: 25,
    course: 'MERN Stack',
    status: 'Completed'
  },
  {
    _id: ObjectId('693ecb7617f57114841e2624'),
    name: 'Diana',
    age: 28,
    course: 'Cybersecurity',
    status: 'In Progress'
  }
]
```

4. Update Operations: Update a single document (change status); update multiple documents.

Step	Action	Command
1	Update Single Document (Set Charlie's status to Completed)	db.students.updateOne({ name: "Charlie Brown" }, { $set: { status: "Completed" } })
2	Update Multiple Documents (Change course for all 'In Progress' students to 'Full Stack')	db.students.updateMany({ status: "In Progress" }, { $set: { course: "Full Stack" } })
3	Verify Updates	db.students.find({ course: "Full Stack" }).pretty()



```
studentDB> db.students.updateMany( { status: "In Progress" }, { $set: { course: "Full Stack" } } )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
studentDB> db.students.find({ course: "Full Stack" }).pretty()
[
  {
    _id: ObjectId('693ecb0417f57114841e2621'),
    name: 'Arun kumar',
    age: 24,
    course: 'Full Stack',
    status: 'In Progress'
  },
  {
    _id: ObjectId('693ecb7617f57114841e2624'),
    name: 'Diana',
    age: 28,
    course: 'Full Stack',
    status: 'In Progress'
  }
]
```

5. Query Operators: Practice using $gt, $lt, $in, $and, $or, $exists.

Operator	Query Condition	Command
$gt	Find students older than 24	db.students.find({ age: { $gt: 24 } })
$in	Find students in MERN Stack or Data Science	db.students.find({ course: { $in: ["MERN Stack", "Data Science"] } })
$and	Find students age > 20 AND status = In Progress	db.students.find({ $and: [{ age: { $gt: 20 } }, { status: "In Progress" }] })
$or	Find students age < 23 OR course = Cybersecurity	db.students.find({ $or: [{ age: { $lt: 23 } }, { course: "Cybersecurity" }] })
$exists	Find students where 'notes' field exists	db.students.find({ notes: { $exists: true } })
$lt	Find students younger than 25	db.students.find({ age: { $lt: 25 } })

```
studentDB> db.students.find({ age: { $gt: 24 } })
[
  {
    _id: ObjectId('693ecb7617f57114841e2624'),
    name: 'Diana',
    age: 28,
    course: 'Full Stack',
    status: 'In Progress'
  }
]
```

```
studentDB> db.students.find({ course: { $in: ["MERN Stack", "Data Science"] } })
[
  {
    _id: ObjectId('693ecb7617f57114841e2622'),
    name: 'ajith',
    age: 22,
    course: 'MERN Stack',
    status: 'Completed'
  }
]
```

```
studentDB> db.students.find({ $and: [ { age: { $gt: 20 } }, { status: "In Progress" } ] })
[
  {
    _id: ObjectId('693ecb0417f57114841e2621'),
    name: 'Arun kumar',
    age: 24,
    course: 'Full Stack',
    status: 'In Progress'
  },
  {
    _id: ObjectId('693ecb7617f57114841e2624'),
    name: 'Diana',
    age: 28,
    course: 'Full Stack',
    status: 'In Progress'
  }
]
```

```
studentDB> db.students.find({ $or: [ { age: { $lt: 23 } }, { course: "Cybersecurity" } ] })
[
  {
    _id: ObjectId('693ecb7617f57114841e2622'),
    name: 'ajith',
    age: 22,
    course: 'MERN Stack',
    status: 'Completed'
  }
]
studentDB> db.students.find({ notes: { $exists: true } })
```

```
studentDB> db.students.find({ age: { $lt: 25 } })
[
  {
    _id: ObjectId('693ecb0417f57114841e2621'),
    name: 'Arun kumar',
    age: 24,
    course: 'Full Stack',
    status: 'In Progress'
  },
  {
    _id: ObjectId('693ecb7617f57114841e2622'),
    name: 'ajith',
    age: 22,
    course: 'MERN Stack',
    status: 'Completed'
  }
]
```

6. Use Case: Library System: Create a Library System use case and apply insert, update, and search operations.

Step	Action	Command
1	Switch to New DB	use libraryDB
2	Insert Initial Books	db.books.insertMany([{ title: "The Martian", author: "Andy Weir", copies: 12, available: 10, tags: ["Sci-Fi"] }, { title: "Educated", author: "Tara Westover", copies: 8, available: 8, tags: ["Biography"] }])
3	Search Operation (Find available Sci-Fi books)	db.books.find({ tags: "Sci-Fi", available: { $gt: 0 } })
4	Update Operation (A copy of The Martian is borrowed)	db.books.updateOne({ title: "The Martian" }, { $inc: { available: -1 } })
5	Verify Update (Check available copies for The Martian)	db.books.find({ title: "The Martian" }, { title: 1, available: 1 })

```
studentDB> use libraryDB
switched to db libraryDB
libraryDB> db.books.insertMany([ { title: "The Martian", author: "Andy Weir", copies: 12, available: 10, tags: ["Sci-Fi"] }, { title: "Educated", author: "T
ara Westover", copies: 8, available: 8, tags: ["Biography"] } ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693f94dd17f57114841e2625'),
    '1': ObjectId('693f94dd17f57114841e2626')
  }
}
```

```
libraryDB> db.books.find({ tags: "Sci-Fi", available: { $gt: 0 } })
[
  {
    _id: ObjectId('693f94dd17f57114841e2625'),
    title: 'The Martian',
    author: 'Andy Weir',
    copies: 12,
    available: 10,
    tags: [ 'Sci-Fi' ]
  }
]
libraryDB> db.books.updateOne( { title: "The Martian" }, { $inc: {available: -1 } } )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
libraryDB> db.books.find({ title: "The Martian" }, { title: 1, available: 1 })
[
  {
    _id: ObjectId('693f94dd17f57114841e2625'),
    title: 'The Martian',
    available: 9
  }
]
```

7. Delete Operations: Delete one document; delete all documents.

Step	Action	Command
1	**Delete One Document** (Delete Alice Smith)	db.students.deleteOne({ name: "Priya" })
2	**Check Count After Delete**	db.students.countDocuments({})
3	**Delete All Documents**	db.students.deleteMany({})
4	**Verify Deletion**	db.students.find()