Create & Connect to Database

```
use schoolDB
```

output:

```
switched to db schoolDB
```

Create a Collection

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

output:

```
{ "ok" : 1 }
```

Insert Documents

```
db.students.insertMany([
  { name: "Alice", age: 20, grade: "A", marks: 85 },
  { name: "Bob", age: 22, grade: "B", marks: 70 },
  { name: "Charlie", age: 21, grade: "A", marks: 90 }
])
```

Output:

```
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("68d1c6bb85a31de38d0ed1b8"),
                ObjectId("68d1c6bb85a31de38d0ed1b9"),
                ObjectId("68d1c6bb85a31de38d0ed1ba")
        ]
}
```

Read Documents

```
db.students.find()
```

```
output:

{ "_id" : ObjectId("68d1c774409d4144b71858b3"), "name" : "Alice", "age" :
20, "grade" : "A", "marks" : 85 }
{ "_id" : ObjectId("68d1c774409d4144b71858b4"), "name" : "Bob", "age" : 22,
"grade" : "B", "marks" : 70 }
{ "_id" : ObjectId("68d1c774409d4144b71858b5"), "name" : "Charlie", "age" :
21, "grade" : "A", "marks" : 90 }
```

Update Documents

```
db.students.updateOne(
  { name: "Bob" },
  { $set: { marks: 75 } }
)
```

```
Output:

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

Read Using Operators

```
db.students.find(
  { name: { $eq: "Bob" } }
)
```

```
db.students.find(

 { name: "Bob" }

 )
```

```
Output:

{ "_id" : ObjectId("68d1c86b2a130b1d0593800e"), "name" : "Bob", "age" : 22,
"grade" : "B", "marks" : 75 }
```

Sorting

db.students.find().sort({ marks: 1 })

```
Output:
```

```
{ "_id" : ObjectId("68d1c93650786c67f121ab85"), "name" : "Bob", "age" : 22,
"grade" : "B", "marks" : 75 }
{ "_id" : ObjectId("68d1c93650786c67f121ab84"), "name" : "Alice", "age" :
20, "grade" : "A", "marks" : 85 }
{ "_id" : ObjectId("68d1c93650786c67f121ab86"), "name" : "Charlie", "age" :
21, "grade" : "A", "marks" : 90 }
```

db.students.find().sort({ marks: -1 })

```
Output:
```

```
{ "_id" : ObjectId("68d1c8f5be3d92e3413449b6"), "name" : "Charlie", "age" :
21, "grade" : "A", "marks" : 90 }
{ "_id" : ObjectId("68d1c8f5be3d92e3413449b4"), "name" : "Alice", "age" :
20, "grade" : "A", "marks" : 85 }
{ "_id" : ObjectId("68d1c8f5be3d92e3413449b5"), "name" : "Bob", "age" : 22,
"grade" : "B", "marks" : 75 }
```

And Condition:

```
db.students.find({
  grade: "A",
  age: { $lt: 22 }
})
```

```
Output:
```

```
{ "_id" : ObjectId("68d1ca5733e4b02ff4404d0c"), "name" : "Alice", "age" :
20, "grade" : "A", "marks" : 85 }
{ "_id" : ObjectId("68d1ca5733e4b02ff4404d0e"), "name" : "Charlie", "age" :
21, "grade" : "A", "marks" : 90 }
```

OR Condition

Students with **marks ≥ 90 OR age ≤ 20**.

```
db.students.find({
  $or: [
    { marks: { $gte: 90 } },
    { age: { $lte: 20 } }
  ]
})
```

```
Output:
```

```
{ "_id" : ObjectId("68d1cacad13a1f5d2919dfea"), "name" : "Alice", "age" :
20, "grade" : "A", "marks" : 85 }
{ "_id" : ObjectId("68d1cacad13a1f5d2919dfec"), "name" : "Charlie", "age" :
21, "grade" : "A", "marks" : 90 }
```

IN Operator

Students whose names are either Alice or Bob.

```
db.students.find({
  name: { $in: ["Alice", "Bob"] }
})
```

```
Output:

{ "_id" : ObjectId("68d1c8f5be3d92e3413449b4"), "name" : "Alice", "age" :
20, "grade" : "A", "marks" : 85 }
{ "_id" : ObjectId("68d1c8f5be3d92e3413449b5"), "name" : "Bob", "age" : 22,
"grade" : "B", "marks" : 75 }
```

Nested Logical Operators

Students with **grade A AND (marks > 80 OR age < 21)**.

```
db.students.find({
  grade: "A",
  $or: [
    { marks: { $gt: 80 } },
    { age: { $lt: 21 } }
  ]
})
```

```
Output:

{ "_id" : ObjectId("68d1c8f5be3d92e3413449b4"), "name" : "Alice", "age" :
20, "grade" : "A", "marks" : 85 }
{ "_id" : ObjectId("68d1c8f5be3d92e3413449b6"), "name" : "Charlie", "age" :
21, "grade" : "A", "marks" : 90 }
```

"Show only *name* and *marks* of students who have marks ≥ 80, and don't show _id."

```
db.students.find(
   { marks: { $gte: 80 } },
   { name: 1, marks: 1, _id: 0 }
)
```

```
Output:

{ "name" : "Alice", "marks" : 85 }
{ "name" : "Charlie", "marks" : 90 }
```

Delete Documents

```
db.students.deleteOne({ name: "Charlie" })
```

Creating Indexes:

db.students.createIndex({ name: 1 })

db.students.createIndex({ name: 1 })

output:

```
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
```

Feature	_id Index	name Index
Goal	Uniquely identifies each document.	Speeds up searches, sorting, or queries on the `name` field.
Mandatory?	✅ **Yes**. MongoDB automatically creates it for every collection.	❌ Optional. You create it when you need faster lookups on `name`.
Uniqueness	✅ Always unique (each document *must* have a unique `_id`).	Can be unique or non-unique depending on how you create it.