

MongoDB



mongoDB®

Session 3 – Basic MongoDB CRUD Operations

Basic MongoDB CRUD Operations

Creating a Database

- `use <database_name>` → create or select a database
- Database is created only when data is inserted
- Example:

```
>_MONGOSH  
  
> use db1  
< switched to db db1  
db1>
```

Creating a Collection

- `db.createCollection("<collection_name>")`
- Example:

```
> db.createCollection("students")  
< { ok: 1 }
```

- Collections are optional; inserting data automatically creates them

**db is a reference (object) to the current database.*

Inserting Documents

- Insert a single document:

```
> db.students.insertOne({coursename:"MongoDB",fees:10000,duration:3})  
< {  
  acknowledged: true,  
  insertedId: ObjectId('69777cb9d750925eedaa0030')  
}
```

Insert multiple documents:

```
> db.students.insertMany([
    {coursename:"Python",fees:3500,duration:4},
    {coursename:"JS",fees:6000,duration:7}
]);
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('69777e14d750925eedaa0032'),
    '1': ObjectId('69777e14d750925eedaa0033')
  }
}
```

Insert an array

```
db.students.insertOne({coursename:'SQL',fees:4000,modes:['Online','Offline']})
{
  acknowledged: true,
  insertedId: ObjectId('697f690ae7f297088d28c651')
}
```

Insert mixed-type arrays (allowed, but avoid in real apps)

```
db.students.insertOne({
  coursename:'Outsystems',fees:15000,versions:['Enterprise',10,2.4]
```

Insert an array of objects

```
db.employee.insertOne({  
  empname: 'Rajiv', designation: 'Manager',  
  experience: [  
    {company: 'Axis Bank', year: 2022},  
    {company: 'HDFC Bank', year: 2024}  
  ]  
})
```

**This replaces JOIN tables in many cases.*

Insert an embedded (nested) document

```
db.employee.insertOne({  
  empname:'Siddhant',designation:'Developer',  
  address:{  
    street:'10th Avenue',  
    city:'Pune',  
    state:'MH'  
  }  
})
```

Reading All Documents

- Find all documents

```
db.students.find()
{
  _id: ObjectId('697ccb8601797dbda5ac7539'),
  name: 'Amit',
  city: 'Mumbai'
}
{
  _id: ObjectId('697ccc5201797dbda5ac753b'),
  name: 'Manish',
  city: 'Pune',
  dob: '2021-03-04'
}
```

Reading Documents

- Find one document

```
db.students.findOne()  
{  
  _id: ObjectId('69777cb9d750925eedaa0030'),  
  coursename: 'MongoDB',  
  fees: 10000,  
  duration: 3  
}
```

- `*db.students.find().pretty()` - used in old mongosh

Reading Documents – Selected Fields

- Will show only designation. `_id` comes by default

```
db.employee.find({designation:'Developer'},{empname:1})
```

- Will show only designation. `_id` is now hidden.

```
db.employee.find({designation:'Developer'},{empname:1,_id:0})
```

Filtering documents

- By Id
- By String
- By Integer
- By Date

```
db.students.find({_id:ObjectId('697ccec68eb37f59c1f9e74')})
```

```
db.students.find({name:'Ravi'})
```

```
db.students.find({age:12})
```

```
db.students.find({dob:new Date('2026-01-30T15:31:27.034Z')})
```

or

```
db.students.find({dob:ISODate('2026-01-30T15:31:27.034Z')})
```

**Not supported by JS/Node*

Filtering in embedded documents

```
db.employee.find({'address.city': 'Pune'})
```



```
_id: ObjectId('697f70c7e7f297088d28c654'),  
empname: 'Siddhant',  
designation: 'Developer',  
address: {  
  street: '10th Avenue',  
  city: 'Pune',  
  state: 'MH'
```

Query with Filters

- Find students with fees > 5000
- Find students with fees < 5000
- Other operators
 - \$lte
 - \$gte
 - \$eq
 - \$ne

```
db.students.find({fees:{$gt:5000}})
```

```
db.students.find({fees:{$lt:5000}})
```

Logical Operators

and

```
db.students.find({$and:[{name:'Shibu'},{city:'Texas'}]}))
```

or

```
db.students.find({$or:[{name:'Shibu'},{city:'Pune'}]}))
```


Thank You
