# BIBD MINI PROJECT

## Aim: Executing CRUD operations in MongoDB shell.

## Steps:

1. Start The MongoDB shell.

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
MongoDB shell version v5.0.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("b74f20f2-e913-40af-a72a-4482315ba5e6") }
MongoDB server version: 5.0.6
================
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
================
---
The server generated these startup warnings when booting:
        2022-03-18T11:58:14.502+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
```

2. Check for any existing databases.

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
```

3. So, we do not have our own existing database, hence we'll create a new one.

```
> use school
switched to db school
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
>
```

4. We've created a database named school here, but it is not displayed because its empty, so we need to create a collection first inside this database. To insert document into collection json format is followed.

```
> show collections
> db.students.insertOne({Name: "Danny", Rollno: 16})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("623c4b985331ff28e5d5111b")
}
>
```

5. Here, we've created a collection in the school database named students and added a document of one student. So now if we check the databases on the system we can see the school database.

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
school  0.000GB
> _
```

6. Now, to check if the document is added in the collection we run:

```
> show collections
students
> db.students.find()
{ "_id" : ObjectId("623c4b985331ff28e5d5111b"), "Name" : "Danny", "Rollno" : 16 }
> _
```

7. So, the document we inserted earlier is shown here. If we want it in a more readable format we can use the pretty() function.

```
> db.students.find().pretty()
{
        "_id" : ObjectId("623c4b985331ff28e5d5111b"),
        "Name" : "Danny",
        "Rollno" : 16
}
>
```

8. We know how to create a database. Now let's see how to delete/drop a database. Here, I've already created another sample database "sampledb" with document in it.

```
> use sampledb
switched to db sampledb
> db.test.insertOne({Name:"abc"})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("623c4e455331ff28e5d5111c")
}
> _
```

```
> show dbs
admin     0.000GB
config    0.000GB
local     0.000GB
sampledb  0.000GB
school    0.000GB
>
```

```
> use sampledb
switched to db sampledb
> db.dropDatabase()
{ "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
school  0.000GB
>
```

9. To drop a single collection, you can do as follows:

```
> db.test.drop()
true
>
```

**10.** The basic CRUD operations include Create, Read, Update & Delete.

**11.** The Create commands are of two types “insertOne(data, options)” & “insertMany([data], options)”.

**12.** The Read command are of two types “find(filter, options)” & “findOne(filter, options)”.

**13.** The Update command are of three types “updateOne(filter, data, options)” ; “updateMany(filter, data, options)” & “replaceOne(filter, data, options)”.

**14.** The Delete command are of two types “deleteOne(filter, options)” & “deleteMany(filter, options)”.

**15.** Executing the insertOne and insertMany commands:

```
> db.employee.insertOne({name:"Ravi", surname:"Joshi", email:"ravi@gmail.com", address:{city:"Mumbai", location:"Andheri"}, hobbies:["Cricket","Music"]})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("623c7ca45331ff28e5d5111f")
}
```

```
> db.employee.insertMany([{name:"Harsh", surname:"Rai", email:"harsh@hotmail.com", address:{city:"Banglore", location:"AMK"}, hobbies:["Football"]}, {name:"Sandy", surname:"Singh", email:"sandys@yahoo.co.in", ad
dress:{city:"Pune", location: "Hinjewadi"}, hobbies:["Reading", "Traveling"]}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("623c7e7e5331ff28e5d51122"),
                ObjectId("623c7e7e5331ff28e5d51123")
        ]
}
>
```

**16.** Let us now check the database.

```
> show dbs
admin    0.000GB
company  0.000GB
config   0.000GB
local    0.000GB
school   0.000GB
> use company
switched to db company
> show collections
employee
```

**17.** Here check the records/document we have updated in the collection employee

```
> db.employee.find().pretty()
{
        "_id" : ObjectId("623c7feb5331ff28e5d51124"),
        "name" : "Ravi",
        "surname" : "Joshi",
        "email" : "ravi@gmail.com",
        "address" : {
                "city" : "Mumbai",
                "location" : "Andheri"
        },
        "hobbies" : [
                "Cricket",
                "Music"
        ]
}
{
        "_id" : ObjectId("623c7ff45331ff28e5d51125"),
        "name" : "Harsh",
        "surname" : "Rai",
        "email" : "harsh@hotmail.com",
        "address" : {
                "city" : "Banglore",
                "location" : "AMK"
        },
        "hobbies" : [
                "Football"
        ]
}
{
        "_id" : ObjectId("623c7ff45331ff28e5d51126"),
        "name" : "Sandy",
        "surname" : "Singh",
        "email" : "sandys@yahoo.co.in",
        "address" : {
                "city" : "Pune",
                "location" : "Hinjewadi"
        },
        "hobbies" : [
                "Reading",
                "Traveling"
        ]
}
>
```

**18.** Here, we've successfully executed the insertOne and insertMany commands and also Read the data in the Document.

**19.** Now let's try updating the location of Ravi to Ghatkopar in the document.

```
> db.employee.updateOne({_id : ObjectId("623c7feb5331ff28e5d51124")}, {$set: {"address.location" : "Ghatkopar"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 0 }
```

**20.**Check if the value is updated:

```
> db.employee.find().pretty()
{
        "_id" : ObjectId("623c7feb5331ff28e5d51124"),
        "name" : "Ravi",
        "surname" : "Joshi",
        "email" : "ravi@gmail.com",
        "address" : {
                "city" : "Mumbai",
                "location" : "Ghatkopar"
        },
        "hobbies" : [
                "Cricket",
                "Music"
        ]
}
{
        "_id" : ObjectId("623c7ff45331ff28e5d51125"),
        "name" : "Harsh",
        "surname" : "Rai",
        "email" : "harsh@hotmail.com",
        "address" : {
                "city" : "Banglore",
                "location" : "AMK"
        },
        "hobbies" : [
                "Football"
        ]
}
{
        "_id" : ObjectId("623c7ff45331ff28e5d51126"),
        "name" : "Sandy",
        "surname" : "Singh",
        "email" : "sandys@yahoo.co.in",
        "address" : {
                "city" : "Pune",
                "location" : "Hinjewadi"
        },
        "hobbies" : [
                "Reading",
                "Traveling"
        ]
}
> _
```

**21.** Now lets try updateMany command

```
> db.employee.updateMany({}, {$set: {relationshipStatus: "unknown"}})
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }
```

**22.** Keeping the first parameter blank means updating all the entries.

```
> db.employee.find().pretty()
{
        "_id" : ObjectId("623c7feb5331ff28e5d51124"),
        "name" : "Ravi",
        "surname" : "Joshi",
        "email" : "ravi@gmail.com",
        "address" : {
                "city" : "Mumbai",
                "location" : "Ghatkopar"
        },
        "hobbies" : [
                "Cricket",
                "Music"
        ],
        "relationshipStatus" : "unknown"
}
{
        "_id" : ObjectId("623c7ff45331ff28e5d51125"),
        "name" : "Harsh",
        "surname" : "Rai",
        "email" : "harsh@hotmail.com",
        "address" : {
                "city" : "Banglore",
                "location" : "AMK"
        },
        "hobbies" : [
                "Football"
        ],
        "relationshipStatus" : "unknown"
}
{
        "_id" : ObjectId("623c7ff45331ff28e5d51126"),
        "name" : "Sandy",
        "surname" : "Singh",
        "email" : "sandys@yahoo.co.in",
        "address" : {
                "city" : "Pune",
                "location" : "Hinjewadi"
        },
        "hobbies" : [
                "Reading",
                "Traveling"
        ],
        "relationshipStatus" : "unknown"
}
> _
```

**23.** Now let’s change status of one employee.

```
> db.employee.updateOne({_id : ObjectId("623c7ff45331ff28e5d51126")}, {$set: {"relationshipStatus" : "Married"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

```
> db.employee.find().pretty()
{
        "_id" : ObjectId("623c7feb5331ff28e5d51124"),
        "name" : "Ravi",
        "surname" : "Joshi",
        "email" : "ravi@gmail.com",
        "address" : {
                "city" : "Mumbai",
                "location" : "Ghatkopar"
        },
        "hobbies" : [
                "Cricket",
                "Music"
        ],
        "relationshipStatus" : "unknown"
}
{
        "_id" : ObjectId("623c7ff45331ff28e5d51125"),
        "name" : "Harsh",
        "surname" : "Rai",
        "email" : "harsh@hotmail.com",
        "address" : {
                "city" : "Banglore",
                "location" : "AMK"
        },
        "hobbies" : [
                "Football"
        ],
        "relationshipStatus" : "unknown"
}
{
        "_id" : ObjectId("623c7ff45331ff28e5d51126"),
        "name" : "Sandy",
        "surname" : "Singh",
        "email" : "sandys@yahoo.co.in",
        "address" : {
                "city" : "Pune",
                "location" : "Hinjewadi"
        },
        "hobbies" : [
                "Reading",
                "Traveling"
        ],
        "relationshipStatus" : "Married"
}
> _
```

**24.** Adding more data in collection and using Find() operations.

```
> db.posts.insertMany([{caption: "Post 1",body: "Something for Post 1", author: "Ravi", tags: ["chillday", "swimming"]}, {caption: "Post 2",body: "Something for Post 2", author: "Harsh", tags: ["Trekking"]}, {caption: "Post 3",body: "Something for Post 3", author: "Sandy", tags: ["SundayVibe", "Resort"]}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("623d6d6e2abed7bdcb5d8ee8"),
                ObjectId("623d6d6e2abed7bdcb5d8ee9"),
                ObjectId("623d6d6e2abed7bdcb5d8eea")
        ]
}
```

```
> db.posts.find().pretty()
{
        "_id" : ObjectId("623d6d6e2abed7bdcb5d8ee8"),
        "caption" : "Post 1",
        "body" : "Something for Post 1",
        "author" : "Ravi",
        "tags" : [
                "chillday",
                "swimming"
        ]
}
{
        "_id" : ObjectId("623d6d6e2abed7bdcb5d8ee9"),
        "caption" : "Post 2",
        "body" : "Something for Post 2",
        "author" : "Harsh",
        "tags" : [
                "Trekking"
        ]
}
{
        "_id" : ObjectId("623d6d6e2abed7bdcb5d8eea"),
        "caption" : "Post 3",
        "body" : "Something for Post 3",
        "author" : "Sandy",
        "tags" : [
                "SundayVibe",
                "Resort"
        ]
}
> _
```

**25.** Now using the Find command to find an entry with a particular tag.

```
> db.posts.find({tags: "Resort"}).pretty()
{
        "_id" : ObjectId("623d6d6e2abed7bdcb5d8eea"),
        "caption" : "Post 3",
        "body" : "Something for Post 3",
        "author" : "Sandy",
        "tags" : [
                "SundayVibe",
                "Resort"
        ]
}
>
```

**26.** To find other attributes of a particular tag we can specify the attribute using 0 or 1 as the value where 0 stands for False and 1 stands for True.

```
> db.posts.find({tags: "swimming"}, {author: 1 , _id: 0}).pretty()
{ "author" : "Ravi" }
> _
```

**27.** Here the author is set to be True and _id is set to False hence ID is hidden and only author of the tag swimming is displayed.

**28.**Now we can work on some delete operations.

**29.**So now let’s delete an entry from employee using deleteOne() where relationshipStatus is Married.

```
> db.employee.deleteOne({name: "Sandy"})
{ "acknowledged" : true, "deletedCount" : 1 }
```

```
> db.employee.find().pretty()
{
        "_id" : ObjectId("623c7feb5331ff28e5d51124"),
        "name" : "Ravi",
        "surname" : "Joshi",
        "email" : "ravi@gmail.com",
        "address" : {
                "city" : "Mumbai",
                "location" : "Ghatkopar"
        },
        "hobbies" : [
                "Cricket",
                "Music"
        ],
        "relationshipStatus" : "unknown"
}
{
        "_id" : ObjectId("623c7ff45331ff28e5d51125"),
        "name" : "Harsh",
        "surname" : "Rai",
        "email" : "harsh@hotmail.com",
        "address" : {
                "city" : "Banglore",
                "location" : "AMK"
        },
        "hobbies" : [
                "Football"
        ],
        "relationshipStatus" : "unknown"
}
> _
```

**30.** Now deleting users with deleteMany() operations where relationshipStatus is unknown.

```
> db.employee.deleteMany({relationshipStatus: "unknown"})
{ "acknowledged" : true, "deletedCount" : 2 }
> db.employee.find().pretty()
```

**31.** All records are deleted and hence we now have an empty collection.

**32.** This is all with the CRUD operations in MongoDB.