

BIBD MINI PROJECT

- **Start mongodb shell**

```

C:\Users\Dell>cd C:\Program Files\MongoDB\Server\5.0\bin
C:\Program Files\MongoDB\Server\5.0\bin>mongo
MongoDB shell version v5.0.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("dfeab31e-3d95-414b-8f8e-f217fbf3f797") }
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Warning: the "mongo" shell has been superseded by "mongosh",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:
  2022-04-07T09:43:27.909+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()
---

```

- **show dbs command**

```

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

```

- **use miniproject(creates a database named miniproject)**

```

> use bibdproject
switched to db bibdproject

```

We've created a database named books 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.

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

The Create commands are of two types "insertOne(data, options)" & "insertMany([data], options)".

The Read command are of two types "find(filter, options)" & "findOne(filter, options)".

The Update command are of three types "updateOne(filter, data, options)" ; "updateMany(filter, data, options)" & "replaceOne(filter, data, options)".

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

- **insertOne()**

We use the insertOne() function to insert one document into a particular collection.

```
> db.books.insertOne({"name":"ABC","city":"Mumbai","pages":230})
{
  "acknowledged" : true,
  "insertedId" : ObjectId("624fac42009c98fc4e7ef453")
}
> _
```

- **insertMany()**

We use the insertMany() function is use to insert multiple documents into a particular collection at once.

```
> db.books.insertMany([{"name":"ABC","city":"Mumbai","pages":230}, {"name":"LMN","city":"Chennai","pages":150}, {"name":"XYZ","city":"Bangalore","pages":200}])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("624fad06009c98fc4e7ef454"),
    ObjectId("624fad06009c98fc4e7ef455"),
    ObjectId("624fad06009c98fc4e7ef456")
  ]
}
```

- **show collections (lists out all the collections in a db)**

```
> show collections
books
>
```

- **find() (To find the records in a collection)**

```
> db.books.find()
{ "_id" : ObjectId("624fac42009c98fc4e7ef453"), "name" : "ABC", "city" : "Mumbai", "pages" : 230 }
{ "_id" : ObjectId("624fad06009c98fc4e7ef454"), "name" : "ABC", "city" : "Mumbai", "pages" : 230 }
{ "_id" : ObjectId("624fad06009c98fc4e7ef455"), "name" : "LMN", "city" : "Chennai", "pages" : 150 }
{ "_id" : ObjectId("624fad06009c98fc4e7ef456"), "name" : "XYZ", "city" : "Bangalore", "pages" : 200 }
```

- **Display the records of the books with city “Chennai”**

```
> db.books.find({city:"Chennai"})
{ "_id" : ObjectId("624fad06009c98fc4e7ef455"), "name" : "LMN", "city" : "Chennai", "pages" : 150 }
{ "_id" : ObjectId("624fad06009c98fc4e7ef456"), "name" : "XYZ", "city" : "Bangalore", "pages" : 200 }
```

- **pretty()** (displays the records in a more readable format)

```
> db.books.find().pretty()
{
  "_id" : ObjectId("624fac42009c98fc4e7ef453"),
  "name" : "ABC",
  "city" : "Mumbai",
  "pages" : 230
}
{
  "_id" : ObjectId("624fad06009c98fc4e7ef454"),
  "name" : "ABC",
  "city" : "Mumbai",
  "pages" : 230
}
{
  "_id" : ObjectId("624fad06009c98fc4e7ef455"),
  "name" : "LMN",
  "city" : "Chennai",
  "pages" : 150
}
{
  "_id" : ObjectId("624fad06009c98fc4e7ef456"),
  "name" : "XYZ",
  "city" : "Bangalore",
  "pages" : 200
}
```

- **updateOne()**

We use the `updateOne()` function to update one document into a particular collection

- **updateMany()**

We use the `updateMany()` function is use to update multiple documents into a particular collection at once.

```
> db.books.updateMany({},{$set:{city:"Chennai"}})
{ "acknowledged" : true, "matchedCount" : 9, "modifiedCount" : 6 }
>
```

- **deleteOne()**

We use the deleteOne() function to delete one document into a particular collection

- **deleteMany ()**

We use the deleteMany() function is use to delete multiple documents into a particular collection at once.

```
> db.books.deleteMany({city:"Chennai"})
{ "acknowledged" : true, "deletedCount" : 9 }
```

- **db.collectionname.drop() (used to delete a collection)**

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

- **db.dropDatabase() (used to delete a database)**

```
> db.dropDatabase()
{ "ok" : 1 }
>
```

- **db.collectionname.count() (used to count the documents in a collection)**

```
> db.books.count()
3
>
```

- **db.collectionname.remove()** (used to remove the documents in a collection)

```
> db.books.remove({name:"ABC"})
WriteResult({ "nRemoved" : 1 })
>
```