# MongoDB CRUD Operations

**CRUD Operations** (Create, Read, Update, and Delete) are the basic set of operations that allow users to interact with the MongoDB server.


**CRUD** is a fundamental concept in computer science, especially when dealing with databases and applications. It stands for **Create, Read, Update, and Delete**, which represent the four basic functions for managing data.

Here's a breakdown of each CRUD operation:

**Create:** This operation allows you to add new entries or records to a database. Imagine adding a new contact to your phone's address book.

**Read:** This operation lets you retrieve existing data from the database. In the address book example, this would be searching for a specific contact.

**Update:** This operation enables you to modify existing data. Continuing with the address book, this could be editing someone's phone number.

**Delete:** This operation allows you to remove unwanted data from the database. In our example, this would be deleting a contact from your address book.

## 1. Create Operations

The create or insert operations are used to insert or add new documents in the collection. If a collection does not exist, then it will create a new collection in the database.

**create** operations using the following methods provided by the MongoDB:

| Method | Description |
|---|---|
| **db.collection.insertOne()** | It is used to insert a single document in the collection. |
| **db.collection.insertMany()** | It is used to insert multiple documents in the collection. |
| **db.createCollection()** | It is used to create an empty collection. |

**Example 1:** In this example, we are inserting details of a single student in the form of document in the student collection using **db.collection.insertOne()** method.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-06-04T09:55:06.977+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use db
switched to db db
db> show collections
locations
players
SStudents
students
Students
Students_permission
db> db.Students.insertOne({
... name:"Shreya",
... age:19,
... courses:"['English','physics']",
... gpa:3.5,
... home_city:"City 3",
... blood_group:"A+",
... is_hotel_resident:true
... })
{
  acknowledged: true,
  insertedId: ObjectId('6667d21fd6b66548facdcdf6')
}
db>
```

**Example 2:** In this example, we are inserting details of the multiple students in the form of documents in the student collection using **db.collection.insertMany()** method.

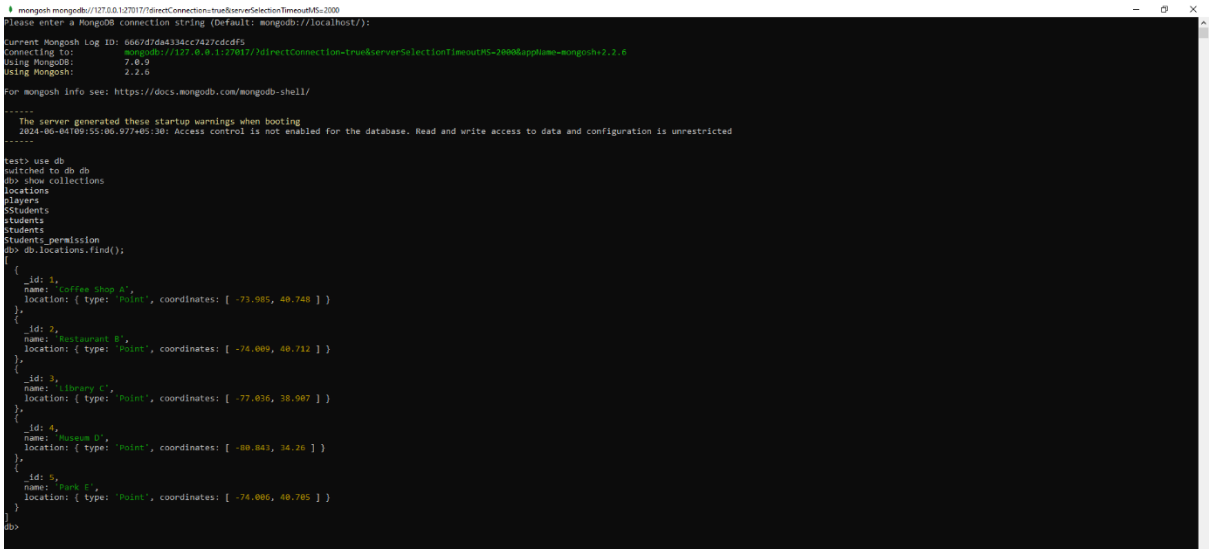
```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Please enter a MongoDB connection string (Default: mongodb://localhost/):

Current Mongosh Log ID: 66685342a096aa7506cdcdf5
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.2.6
Using MongoDB:          7.0.11
Using Mongosh:          2.2.6

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-06-11T18:28:03.490+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use db
switched to db db
db> show collections
foo
Students
db> db.foo.insertMany([{name:"shankar",age:20,branch:"cse",mode:"online"},{name:"lakshmi",age:25,branch:"ece",mode:"offline"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('666853e9a096aa7506cdcdf6'),
    '1': ObjectId('666853e9a096aa7506cdcdf7')
  }
}
db> db.foo.find()
[
  {
    _id: ObjectId('666853e9a096aa7506cdcdf6'),
    name: 'shankar',
    age: 20,
    branch: 'cse',
    mode: 'online'
  },
  {
    _id: ObjectId('666853e9a096aa7506cdcdf7'),
    name: 'lakshmi',
    age: 25,
    branch: 'ece',
    mode: 'offline'
  }
]
db>
```

## 2. Read Operations

The Read operations are used to retrieve documents from the collection, or in other words, read operations are used to query a collection for a document.

You can perform read operation using the following method provided by the MongoDB:

| Method | Description |
|---|---|
| **db.collection.find()** | It is used to retrieve documents from the collection. |




## 3. Update Operations

The update operations are used to update or modify the existing document in the collection. You can perform update operations using the following methods provided by the MongoDB:

| Method | Description |
|---|---|
| **db.collection.updateOne()** | It is used to update a single document in the collection that satisfy the given criteria. |
| **db.collection.updateMany()** | It is used to update multiple documents in the collection that satisfy the given criteria. |
| **db.collection.replaceOne()** | It is used to replace single document in the collection that satisfy the given criteria. |

**Example 1:** In this example, we are updating the age of Sumit in the student collection using **db.collection.updateOne( )** method.

```
db> db.createCollection("foo")
{ ok: 1 }
db> db.Students.insertOne({ name: "Shreya", age: 19, courses: "['English','physics']", gpa: 3.5, home_city: "City 3", blood_group: "A+", is_hotel_resident: true },)
{
  acknowledged: true,
  insertedId: ObjectId('6667dd784334cc7427cdcdf6')
}
db> db.foo.insertOne({ name: "Shreya", age: 19, courses: "['English','physics']", gpa: 3.5, home_city: "City 3", blood_group: "A+", is_hotel_resident: true },)
{
  acknowledged: true,
  insertedId: ObjectId('6667dded4334cc7427cdcdf7')
}
db> db.foo.find()
[
  {
    _id: ObjectId('6667dded4334cc7427cdcdf7'),
    name: 'Shreya',
    age: 19,
    courses: "['English','physics']",
    gpa: 3.5,
    home_city: 'City 3',
    blood_group: 'A+',
    is_hotel_resident: true
  }
]
db> db.foo.updateOne({name:"Shreya"},{$set:{age:20}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
db> db.foo.find()
[
  {
    _id: ObjectId('6667dded4334cc7427cdcdf7'),
    name: 'Shreya',
    age: 20,
    courses: "['English','physics']",
    gpa: 3.5,
    home_city: 'City 3',
    blood_group: 'A+',
    is_hotel_resident: true
  }
]
db>
```

**Example 2:** In this example, we are inserting details of the multiple students in the form of documents in the student collection using **db.collection.insertMany( )** method.

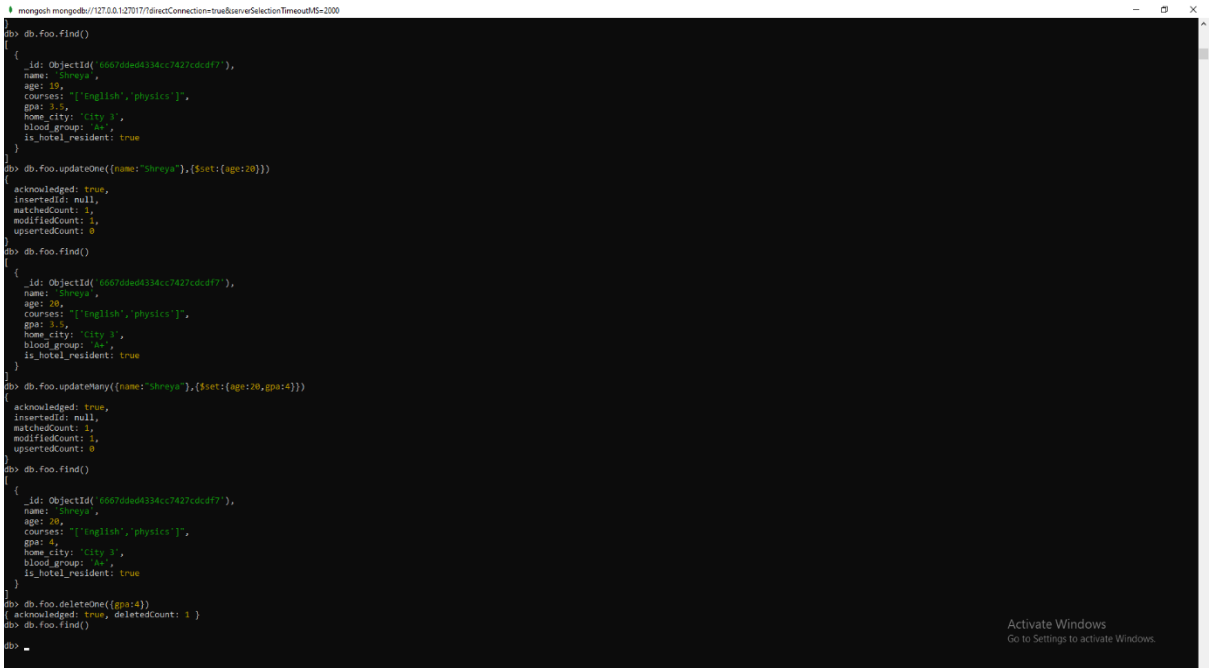
```
]
db> db.foo.updateMany({name:"Shreya"},{$set:{age:20,gpa:4}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
db> db.foo.find()
[
  {
    _id: ObjectId('6667dded4334cc7427cdcdf7'),
    name: 'Shreya',
    age: 20,
    courses: "['English','physics']",
    gpa: 4,
    home_city: 'City 3',
    blood_group: 'A+',
    is_hotel_resident: true
  }
]
db>
```

## 4. Delete Operations

The delete operation are used to delete or remove the documents from a collection. You can perform delete operations using the following methods provided by the MongoDB:

| Method | Description |
|---|---|
| **db.collection.deleteOne()** | It is used to delete a single document from the collection that satisfy the given criteria. |
| **db.collection.deleteMany()** | It is used to delete multiple documents from the collection that satisfy the given criteria. |

**Example 1:** In this example, we are deleting a document from the student collection using **db.collection.deleteOne()** method.




**Example 2:** In this example, we are deleting all the documents from the student collection using **db.collection.deleteMany( )** method.