Day-3

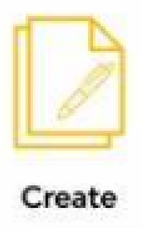
MongoDB CRUD Operations

Create database in Mongodb : use myDatabase

CRUD operations Create, Read, Update, and Delete are essential for interacting with databases. In MongoDB, CRUD operations allow users to perform various actions like inserting new documents, reading data, updating records, and deleting documents from collections. Mastering these operations is fundamental to working with MongoDB and building efficient applications.

in this we will explain each of the four core CRUD operations in MongoDB, their use cases, and provide examples to help you perform these operations effectively.

Create: Add new documents to a collection.
Read: Retrieve documents from a collection.
Update: Modify existing documents.
Delete: Remove documents from a collection.



Create Operation

Note : While creating collection there is no data type enforcement by default because it's schema less

Here are two way to create a collections

1) Create collection automatically(Commonly)
```
use myDatabase
db.employees.insertOne({ name: "Alice", position: "Developer" })

//insertMany({})
```

2) Create Collection explicitly
```
use myDatabase
db.createCollection("departments")
```

3) View collection(or view table)
```
show collections
```

Note : But if we want to create then we can
```
db.createCollection("users", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["name", "age"],
      properties: {
        name: { bsonType: "string" },
        age: { bsonType: "int", minimum: 0 }
      }
    }
  }
})
```

name is string
age is non-negative number

Read Operation

Read

db.employees.find()   // This returns all documents in the employees collection.

db.employees.find().pretty() // This will return in pretty formate

Apply a filter in read operation :

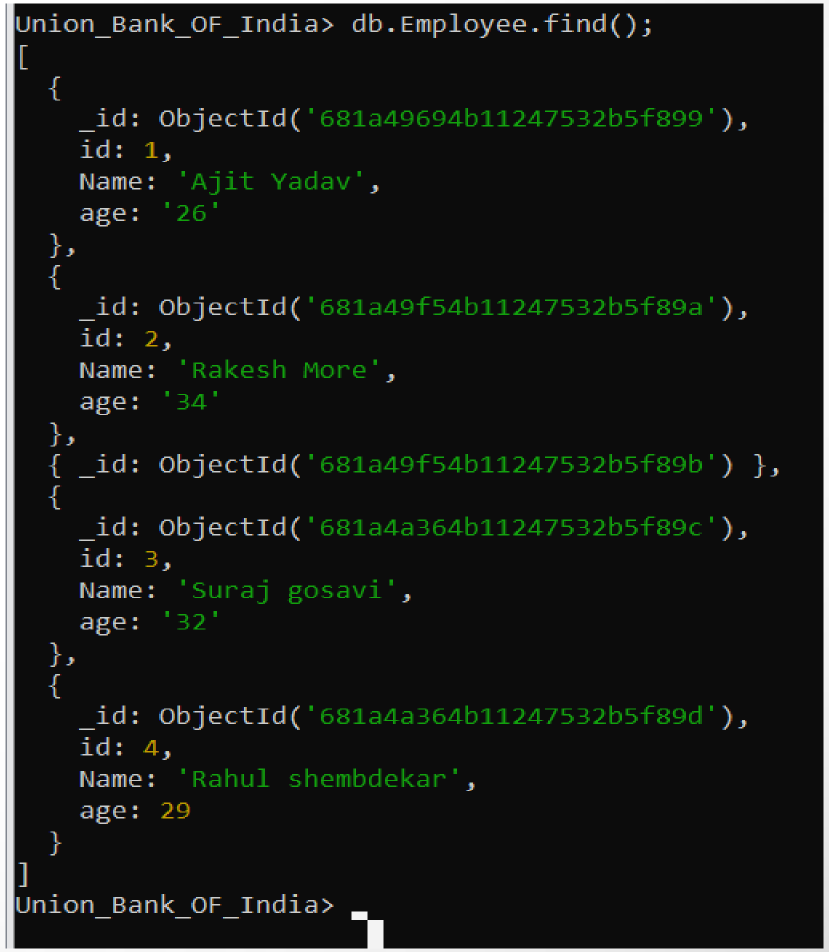
db.employees.find({ position: "Developer" })   // Returns documents where position is "Developer".

```
// Read with conditions .
// Comparison
db.employees.find({ age: { $gt: 30 } })  // age > 30

// Logical
db.employees.find({ $or: [ { age: { $lt: 25 } }, { position: "Manager" } ] })

// and operator
db.employees.find({
  $and: [
    { age: 30 },
    { department: "IT" }
  ]
})
```

Project specific fields . (project means a column name)
db.employees.find({}, { name: 1, position: 1, _id: 0 }) // Returns only name and position fields, excluding _id.

// Retrive single document (document is nothing but a column name)
db.employees.findOne({ name: "Alice" })

// Retrive count of a document
db.employees.countDocuments({ position: "Developer" })



```
Union_Bank_OF_India> db.Employee.find();
[
  {
    _id: ObjectId('681a49694b11247532b5f899'),
    id: 1,
    Name: 'Ajit Yadav',
    age: '26'
  },
  {
    _id: ObjectId('681a49f54b11247532b5f89a'),
    id: 2,
    Name: 'Rakesh More',
    age: '34'
  },
  { _id: ObjectId('681a49f54b11247532b5f89b') },
  {
    _id: ObjectId('681a4a364b11247532b5f89c'),
    id: 3,
    Name: 'Suraj gosavi',
    age: '32'
  },
  {
    _id: ObjectId('681a4a364b11247532b5f89d'),
    id: 4,
    Name: 'Rahul shembdekar',
    age: 29
  }
]
Union_Bank_OF_India>
```

Update operation : update operations are used to modify documents in a collection.

Updates only the first matching document.

```
db.employees.updateOne(
  { name: "Alice" },              // Filter
  { $set: { age: 35 } }           // Update
)
```

Updates all documents matching the filter

```
db.employees.updateMany(
  { department: "IT" },
  { $set: { location: "Head Office" } }
)
```

Replaces the entire document (except _id) with a new one.

```
db.employees.replaceOne(
  { name: "Bob" },
  { name: "Bob", age: 28, department: "Finance" }
)
```

Operator	Description
`$set`	Sets the value of a field
`$inc`	Increments a numeric field
`$unset`	Removes a field
`$rename`	Renames a field

```
db.employees.updateOne(
  { name: "Charlie" },
  { $inc: { age: 1 } }
)
```

Delete operation : delete operations are used to remove documents from a collection.

Delete

Note : U can delete one or multiple document using deleteOne() & deleteMany()

db.employees.deleteOne({ name: "Alice" }) // Removes the first document where name is "Alice".

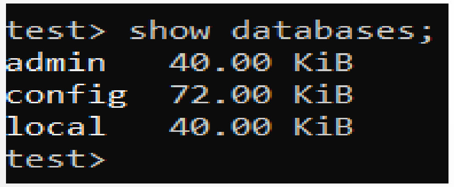
db.employees.deleteMany({ department: "HR" }) // Removes all employees in the HR department.

db.employees.deleteMany({}) // Deletes everything from the employees collection (but the collection itself still exists).

db.employees.drop() // This is remove employees connection completely

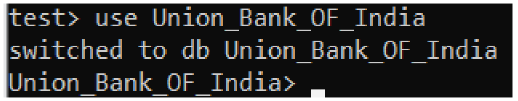
and so many methods in this u can learn from below link
https://www.geeksforgeeks.org/mongodb-tutorial/

Practicle Time :
Show the list of database :



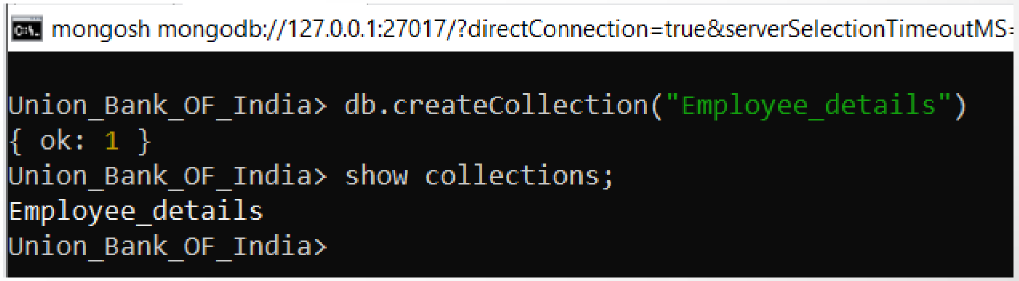
```
test> show databases;
admin    40.00 KiB
config   72.00 KiB
local    40.00 KiB
test>
```

Command to create database



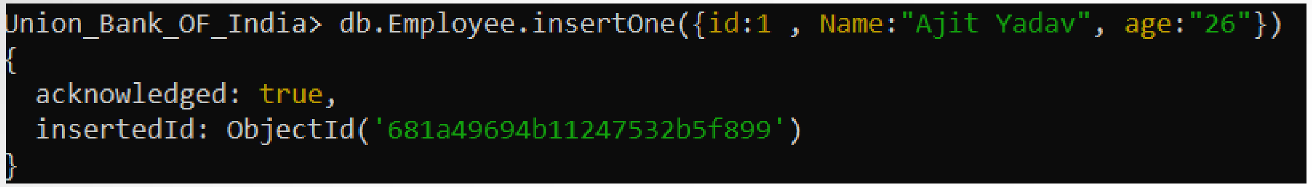
```
test> use Union_Bank_OF_India
switched to db Union_Bank_OF_India
Union_Bank_OF_India>
```

Create collection & show collection list in db



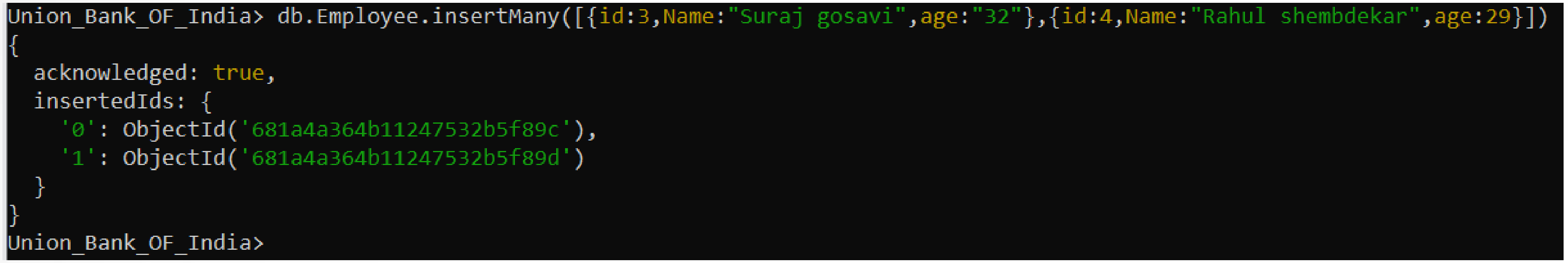
```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=

Union_Bank_OF_India> db.createCollection("Employee_details")
{ ok: 1 }
Union_Bank_OF_India> show collections;
Employee_details
Union_Bank_OF_India>
```

Insert record in a collection(table) single record



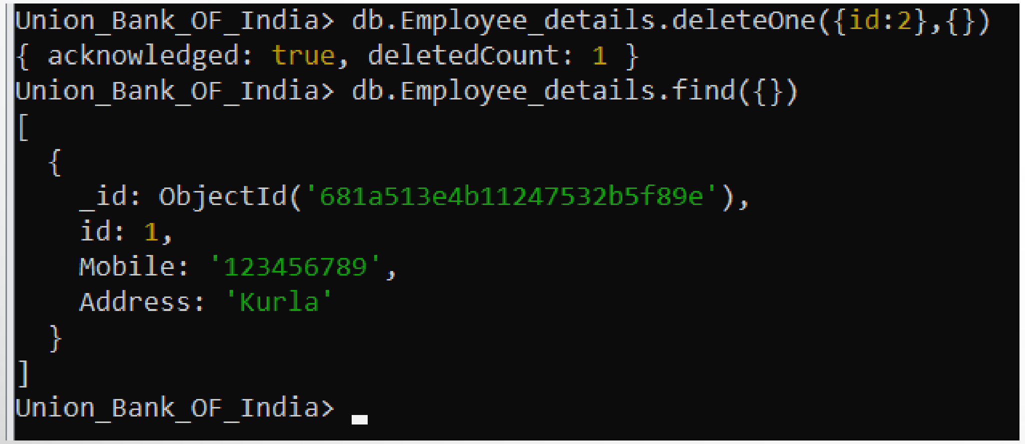
```
Union_Bank_OF_India> db.Employee.insertOne({id:1 , Name:"Ajit Yadav", age:"26"})
{
  acknowledged: true,
  insertedId: ObjectId('681a49694b11247532b5f899')
}
```

Insert more than one record at a time in a collection(table)



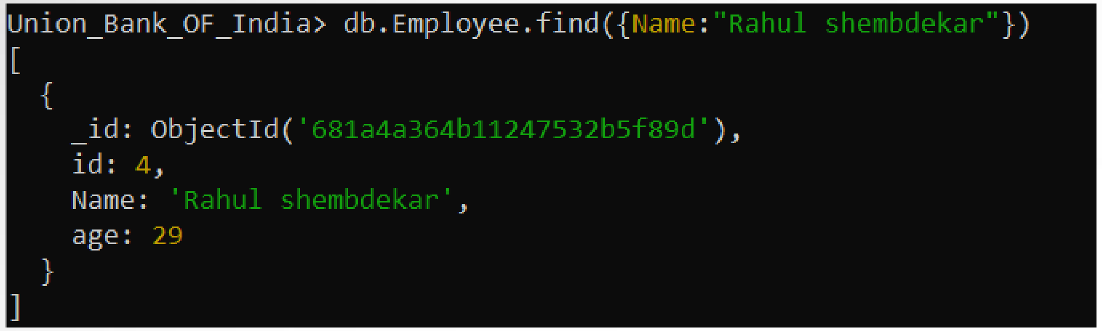
```
Union_Bank_OF_India> db.Employee.insertMany([{id:3,Name:"Suraj gosavi",age:"32"},{id:4,Name:"Rahul shembdekar",age:29}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('681a4a364b11247532b5f89c'),
    '1': ObjectId('681a4a364b11247532b5f89d')
  }
}
Union_Bank_OF_India>
```

Delete single document(table data)



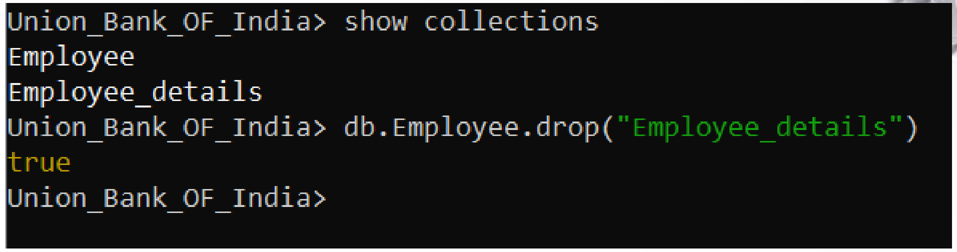
```
Union_Bank_OF_India> db.Employee_details.deleteOne({id:2},{})
{ acknowledged: true, deletedCount: 1 }
Union_Bank_OF_India> db.Employee_details.find({})
[
  {
    _id: ObjectId('681a513e4b11247532b5f89e'),
    id: 1,
    Mobile: '123456789',
    Address: 'Kurla'
  }
]
Union_Bank_OF_India>
```

This is filter query like in relation where clause ex: where name="Rahul shembdekar"



```
Union_Bank_OF_India> db.Employee.find({Name:"Rahul shembdekar"})
[
  {
    _id: ObjectId('681a4a364b11247532b5f89d'),
    id: 4,
    Name: 'Rahul shembdekar',
    age: 29
  }
]
```

Drop collection(table)



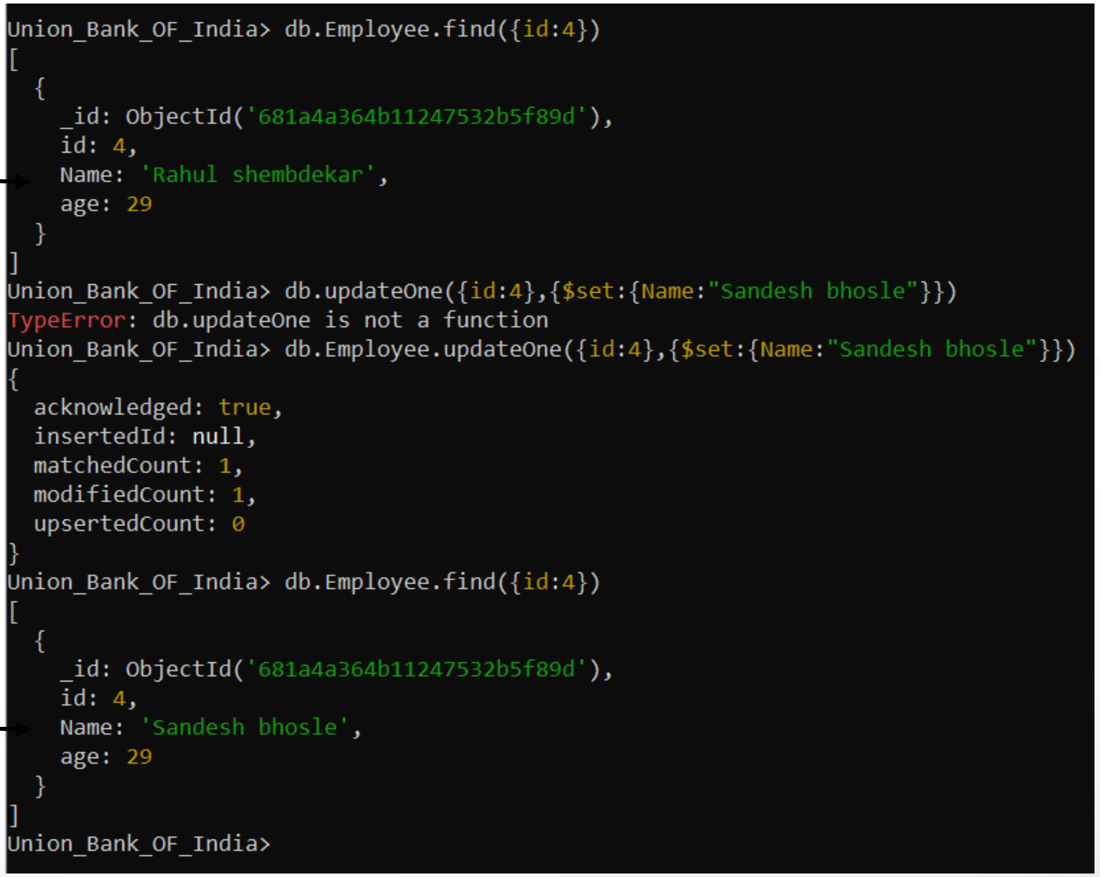
```
Union_Bank_OF_India> show collections
Employee
Employee_details
Union_Bank_OF_India> db.Employee.drop("Employee_details")
true
Union_Bank_OF_India>
```

Next we have to see update the age of the above data

Old value

Update command

Updated value



```
Union_Bank_OF_India> db.Employee.find({id:4})
[
  {
    _id: ObjectId('681a4a364b11247532b5f89d'),
    id: 4,
    Name: 'Rahul shembdekar',
    age: 29
  }
]
Union_Bank_OF_India> db.updateOne({id:4},{$set:{Name:"Sandesh bhosle"}})
TypeError: db.updateOne is not a function
Union_Bank_OF_India> db.Employee.updateOne({id:4},{$set:{Name:"Sandesh bhosle"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Union_Bank_OF_India> db.Employee.find({id:4})
[
  {
    _id: ObjectId('681a4a364b11247532b5f89d'),
    id: 4,
    Name: 'Sandesh bhosle',
    age: 29
  }
]
Union_Bank_OF_India>
```