

Name: Rohit Sadula

Roll no: 37

# Business Intelligence and Big Data Analytics

## Mini Project

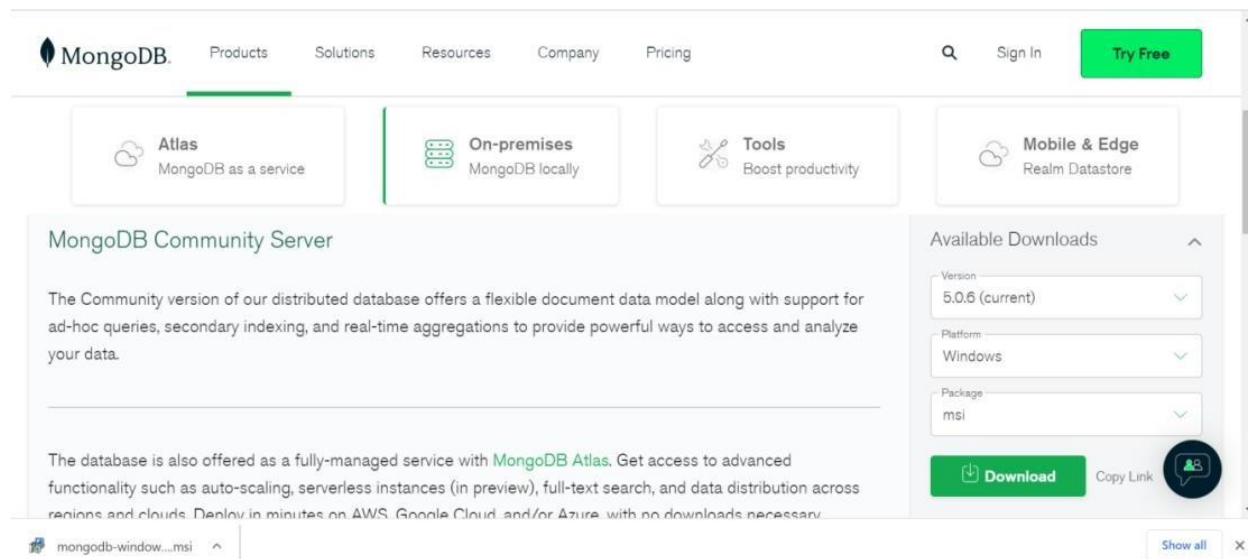
**Aim:** Executing CRUD operations in MongoDB shell.

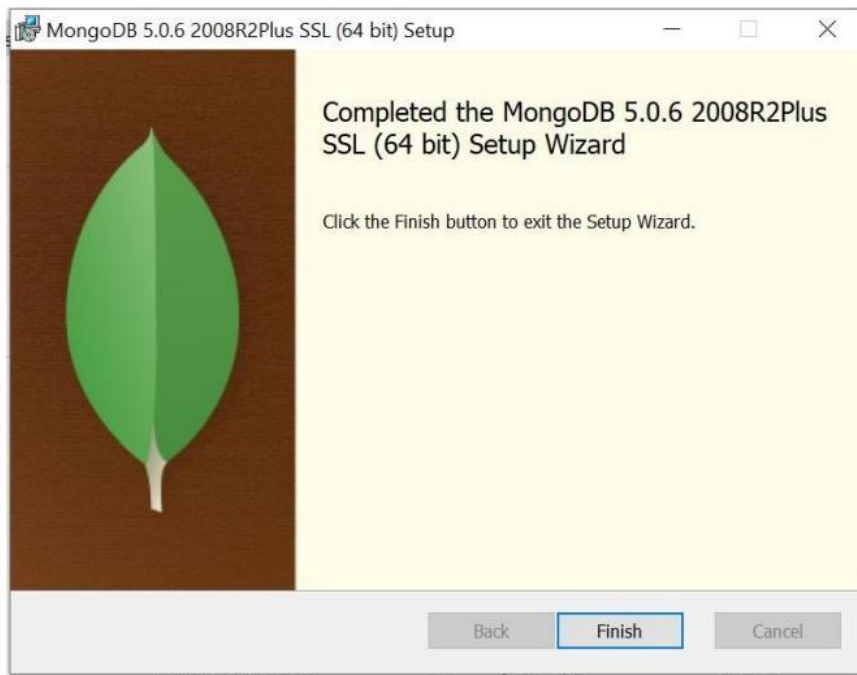
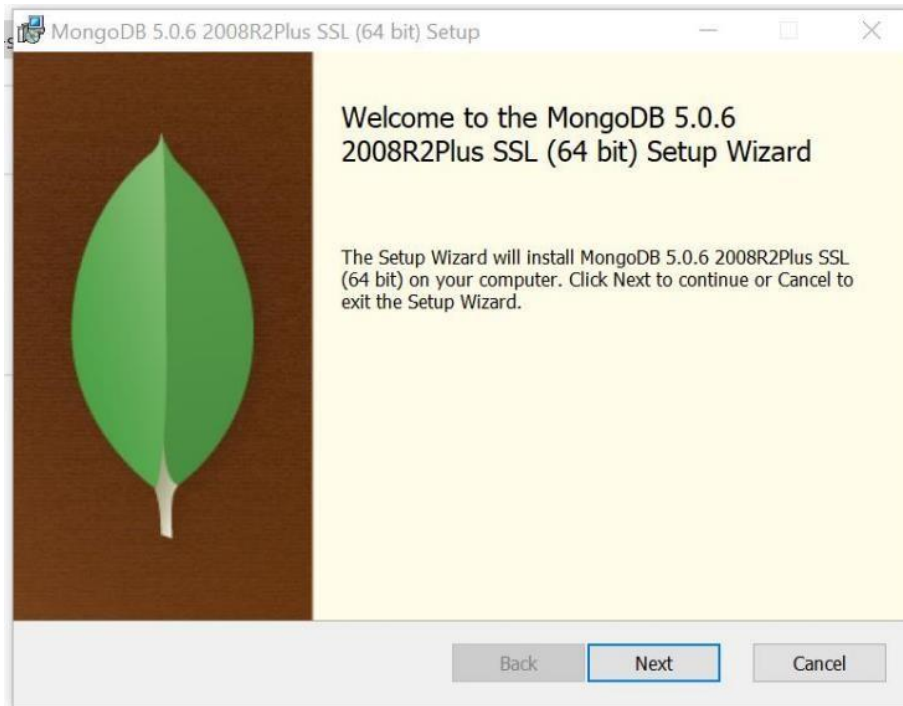
**Steps:**

**Steps to install MONGODB :**

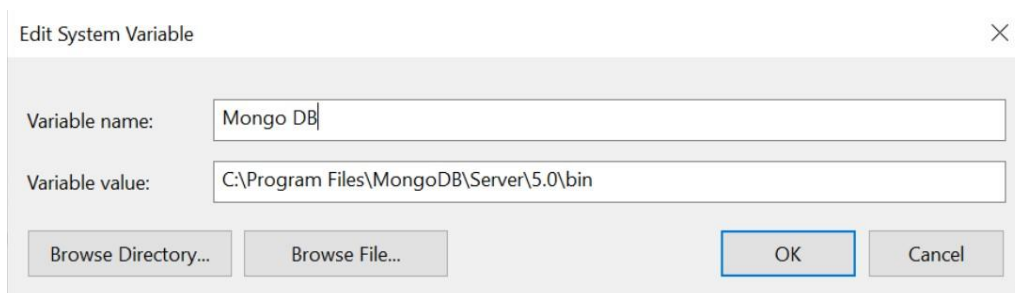
**Step1:** We need to download mongodb from <https://www.mongodb.com/> website.

**Step2:** After successfully downloading mongodb we need to install it in our system.





**Step3:** After successfully installation of the mongodb we need to set path variable of the bin file of mongodb in environment in system .eg “C:\Program Files\MongoDB\Server\5.0\bin”



**Step4:** After setting path variable we can easily use mongo command in cmd and run mongod shell.

## Steps to execute commands:

### 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("31fa4815-1ce0-4d9f-a881-cb3cb3cafe5d") }
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Warning: the "mongo" shell has been superseded by "mongosh",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:
  2022-04-01T22:20:18.136+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 bibdb
switched to db bibdb
>
```

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

### 4. We've created a database named bibdb here, but it is not displayed because it's empty, so

we need to create a collection first inside this database. To insert a document into the collection json format is followed.

```
> db.student.insert({"studname":"ram", "address":"bandra", "class":"msc cs"})
WriteResult({ "nInserted" : 1 })
>
```

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

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

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

```
> db.student.find()
{ "_id" : ObjectId("624486a3662a9f8946b9ecdc"), "studname" : "ram", "address" : "bandra", "class" : "msc cs" }
>
```

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.student.find().pretty()
{
  "_id" : ObjectId("624486a3662a9f8946b9ecdc"),
  "studname" : "ram",
  "address" : "bandra",
  "class" : "msc cs"
}
>
```

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 "demodb" with a document in it.

```
> use demodb
switched to db demodb
> db.test.insertOne({Name: 'abc'})
{
  "acknowledged" : true,
  "insertedId" : ObjectId("6249671305ccee021b594e6a")
}
>
```

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

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

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

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

If already deleted then it will show 'false' when we execute the same command:

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

**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 commands 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:

```
> use bibd
switched to db bibd
> db.student.insertOne({studname: 'Rahul', address: 'borivali', class: 'msc cs'})
{
  "acknowledged" : true,
  "insertedId" : ObjectId("6249690b05ccee021b594e6b")
}
```

```
> db.student.insertMany([{studname: 'Kapil', address: 'chembur', class: 'msc it'}, {studname: 'Roshni', address: 'kalyan', class: 'msc cs'}])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("624969c905ccee021b594e6c"),
    ObjectId("624969c905ccee021b594e6d")
  ]
}
```

16. Let us now check the database.

```
> show dbs
admin 0.000GB
bibd 0.000GB
bibdb 0.000GB
config 0.000GB
local 0.000GB
> show collections
student
>
```

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

```
> db.student.find().pretty()
{
  "_id" : ObjectId("6249690b05ccee021b594e6b"),
  "studname" : "Rahul",
  "address" : "borivali",
  "class" : "msc cs"
}
{
  "_id" : ObjectId("624969c905ccee021b594e6c"),
  "studname" : "Kapil",
  "address" : "chembur",
  "class" : "msc it"
}
{
  "_id" : ObjectId("624969c905ccee021b594e6d"),
  "studname" : "Roshni",
  "address" : "kalyan",
  "class" : "msc cs"
}
```

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 class of Kapil to MSC CS in the document.

```
> db.student.updateOne({studname:'Kapil'},{$set:{class: "msc cs"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

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

```
> db.student.find().pretty()
{
  "_id" : ObjectId("6249690b05ccee021b594e6b"),
  "studname" : "Rahul",
  "address" : "borivali",
  "class" : "msc cs"
}
{
  "_id" : ObjectId("624969c905ccee021b594e6c"),
  "studname" : "Kapil",
  "address" : "chembur",
  "class" : "msc cs"
}
{
  "_id" : ObjectId("624969c905ccee021b594e6d"),
  "studname" : "Roshni",
  "address" : "kalyan",
  "class" : "msc cs"
}
>
```

**21.** Now let's try updateMany command

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

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

```

> db.student.find().pretty()
{
  "_id" : ObjectId("6249690b05ccee021b594e6b"),
  "studname" : "Rahul",
  "address" : "borivali",
  "class" : "msc cs",
  "mobileNo" : "unknown"
}
{
  "_id" : ObjectId("624969c905ccee021b594e6c"),
  "studname" : "Kapil",
  "address" : "chembur",
  "class" : "msc cs",
  "mobileNo" : "unknown"
}
{
  "_id" : ObjectId("624969c905ccee021b594e6d"),
  "studname" : "Roshni",
  "address" : "kalyan",
  "class" : "msc cs",
  "mobileNo" : "unknown"
}
>

```

**23.** Now let's change the status of one employee.

```

> db.student.updateOne({studname:'Rahul'},{$set:{mobileNo: 9876556790}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.student.find().pretty()
{
  "_id" : ObjectId("6249690b05ccee021b594e6b"),
  "studname" : "Rahul",
  "address" : "borivali",
  "class" : "msc cs",
  "mobileNo" : 9876556790
}
{
  "_id" : ObjectId("624969c905ccee021b594e6c"),
  "studname" : "Kapil",
  "address" : "chembur",
  "class" : "msc cs",
  "mobileNo" : "unknown"
}
{
  "_id" : ObjectId("624969c905ccee021b594e6d"),
  "studname" : "Roshni",
  "address" : "kalyan",
  "class" : "msc cs",
  "mobileNo" : "unknown"
}
>

```



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

```
> db.student.find({studname: 'Rahul'}).pretty()
{
  "_id" : ObjectId("6249690b05ccee021b594e6b"),
  "studname" : "Rahul",
  "address" : "borivali",
  "class" : "msc cs",
  "mobileNo" : 9876556790
}
```

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

**26.** So now let's delete an entry from an employee using deleteOne() where mobile number is provided.

```
> db.student.deleteOne({studname:'Rahul'})
{ "acknowledged" : true, "deletedCount" : 1 }
> db.student.find().pretty()
{
  "_id" : ObjectId("624969c905ccee021b594e6c"),
  "studname" : "Kapil",
  "address" : "chembur",
  "class" : "msc cs",
  "mobileNo" : "unknown"
}
{
  "_id" : ObjectId("624969c905ccee021b594e6d"),
  "studname" : "Roshni",
  "address" : "kalyan",
  "class" : "msc cs",
  "mobileNo" : "unknown"
}
```

**27.** Now deleting users with deleteMany() operations where mobile number is unknown.

```
> db.student.deleteMany({mobileNo:'unknown'})
{ "acknowledged" : true, "deletedCount" : 2 }
> db.student.find().pretty()
>
```

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

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