Name: Bhakti Varadkar

Roll No.: 36

Subject: Business Intelligence and Big Data Analytics

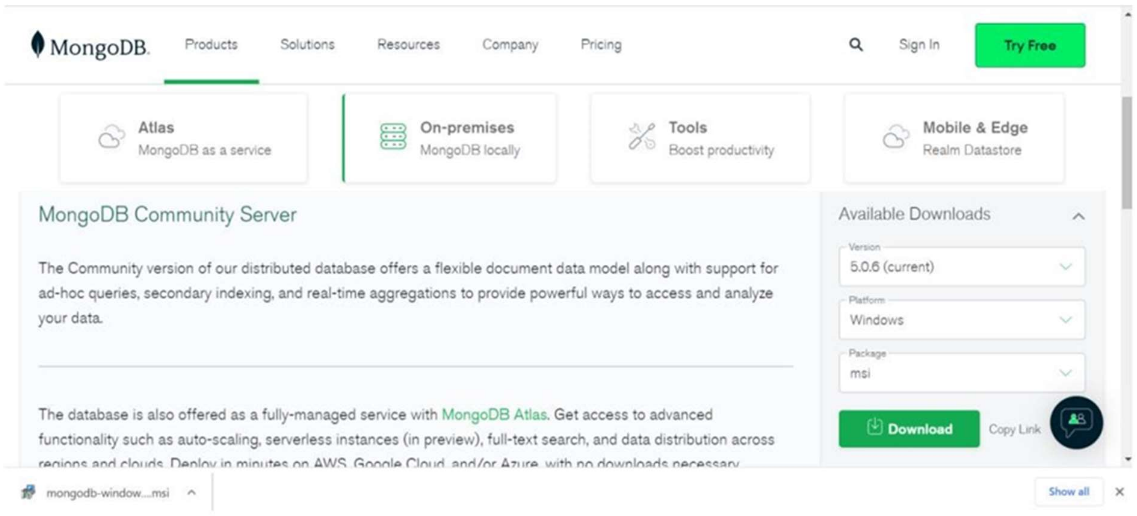
Academic Year: 2021-2022

Mini Project

Aim: Implementation of NoSQL Database - MongoDB.

❖ *Installation of MongoDB:*

1: We need to download MongoDB from https://www.mongodb.com/ website.



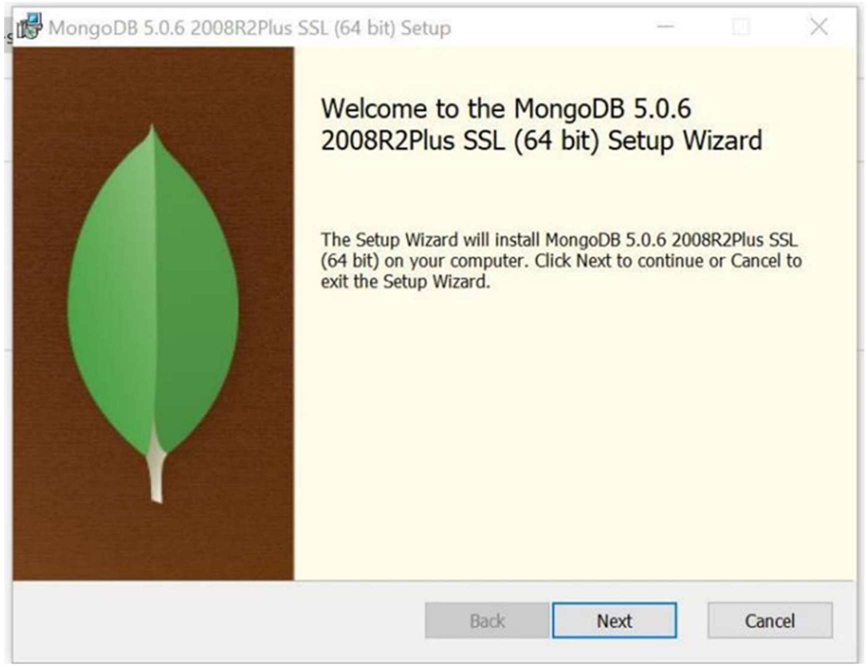
MongoDB Community Server

The Community version of our distributed database offers a flexible document data model along with support for ad-hoc queries, secondary indexing, and real-time aggregations to provide powerful ways to access and analyze your data.

The database is also offered as a fully-managed service with MongoDB Atlas. Get access to advanced functionality such as auto-scaling, serverless instances (in preview), full-text search, and data distribution across regions and clouds. Deploy in minutes on AWS, Google Cloud, and/or Azure, with no downloads necessary.

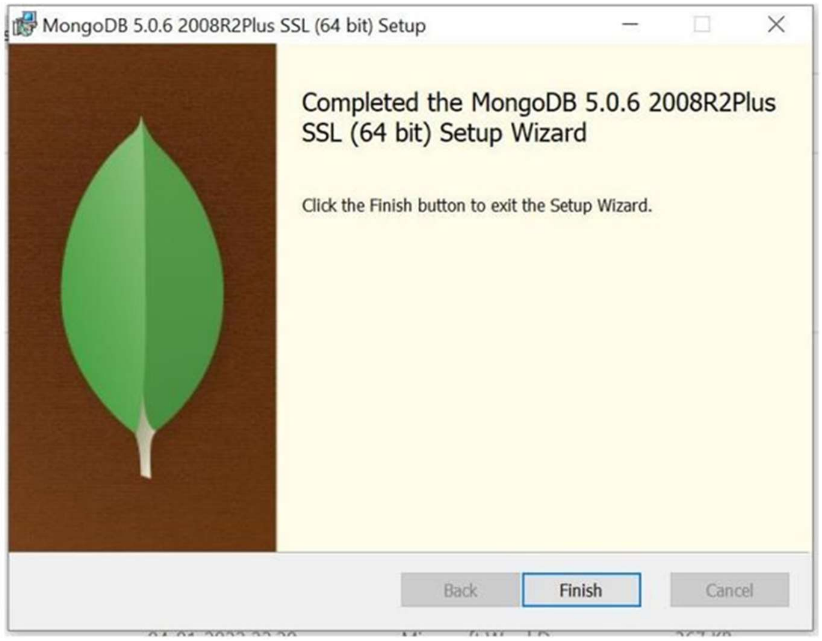
Version: 5.0.6 (current)
Platform: Windows
Package: msi

2: After successfully downloading MongoDB we need to install it in our system.



Welcome to the MongoDB 5.0.6 2008R2Plus SSL (64 bit) Setup Wizard

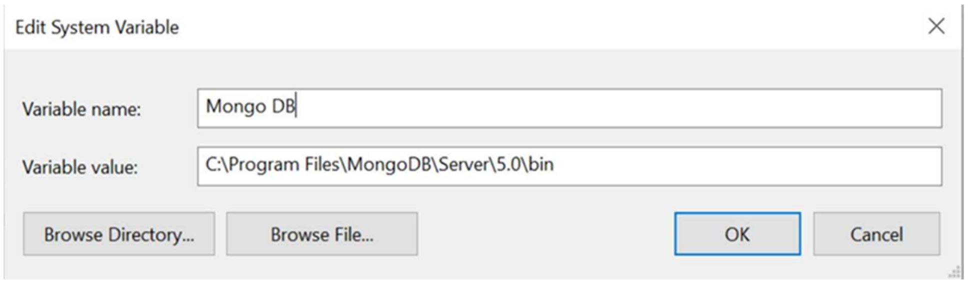
The Setup Wizard will install MongoDB 5.0.6 2008R2Plus SSL (64 bit) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.


MongoDB 5.0.6 2008R2Plus SSL (64 bit) Setup

Completed the MongoDB 5.0.6 2008R2Plus SSL (64 bit) Setup Wizard

Click the Finish button to exit the Setup Wizard.

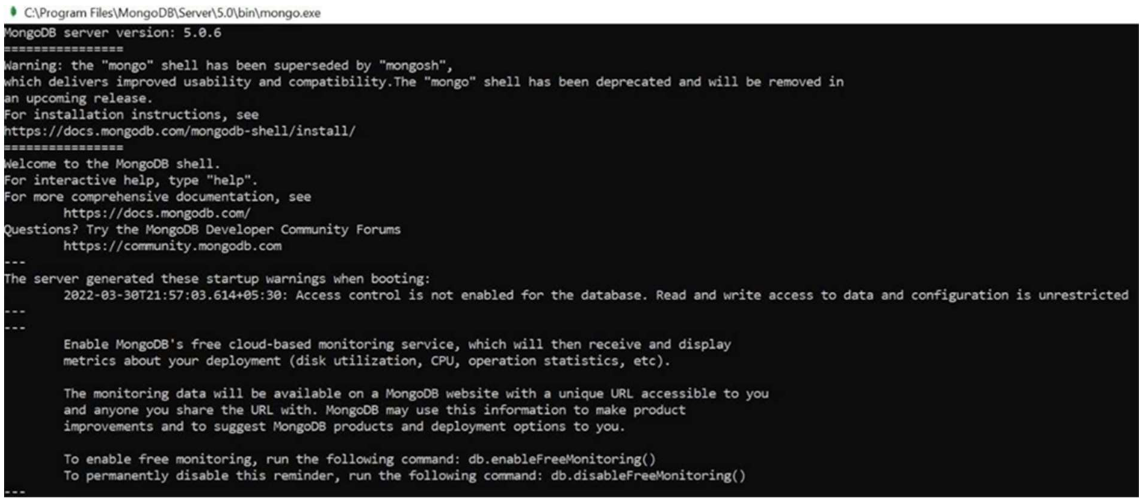
3: After successfully installation of the MongoDB we need to set path variable of the bin file of MongoDB in environment in system. E.g. "C:\Program Files\MongoDB\Server\5.0\bin"


Edit System Variable

Variable name: Mongo DB

Variable value: C:\Program Files\MongoDB\Server\5.0\bin

4: After setting path variable we can easily use mongo command in cmd and run MongoDB shell.


C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe

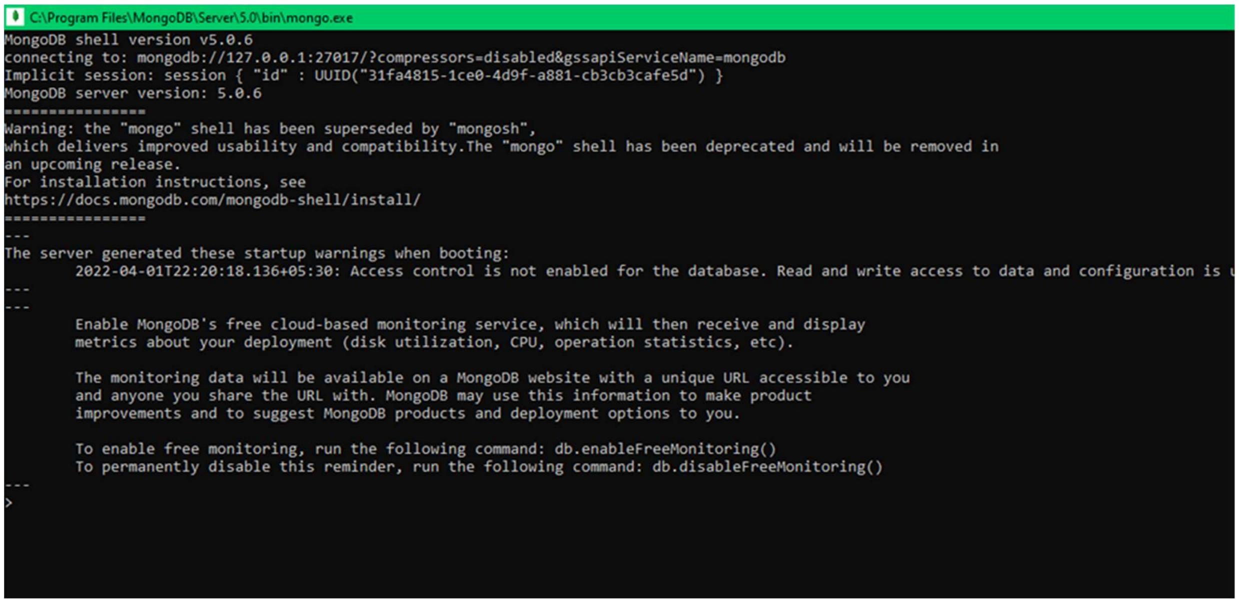
```
MongoDB server version: 5.0.6
================
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
================
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
        https://community.mongodb.com
---
The server generated these startup warnings when booting:
        2022-03-30T21:57:03.614+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()
---
```

❖ *Execution:*

1. Start the MongoDB shell.



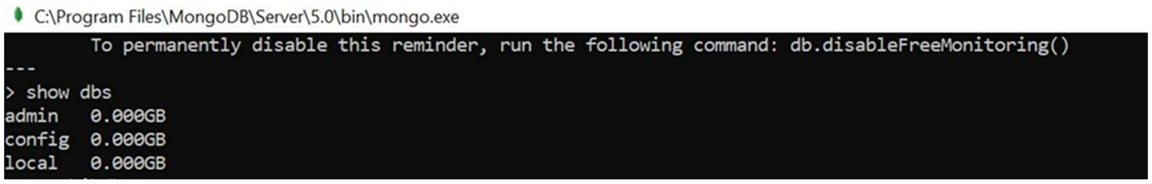
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe

```
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
---
---
        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.

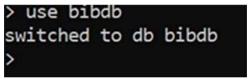


C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe

```
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
```

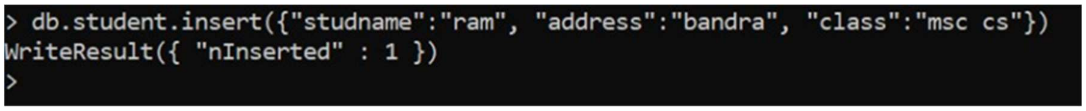
CREATE Database:

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



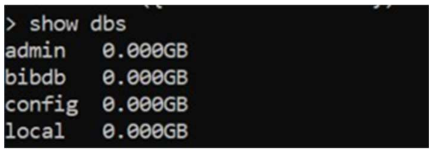
```
> use bibdb
switched to db bibdb
>
```

4. Even after creating the database it won't be 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.



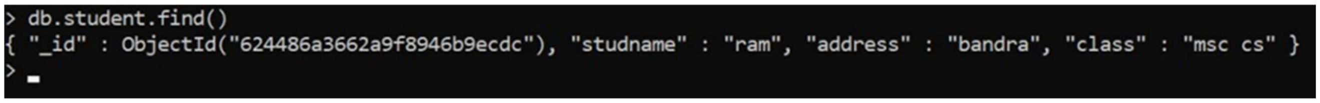
```
> db.student.insert({"studname":"ram", "address":"bandra", "class":"msc cs"})
WriteResult({ "nInserted" : 1 })
>
```

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



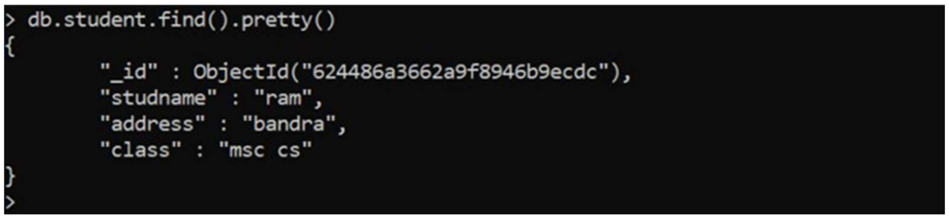
```
> show dbs
admin   0.000GB
bibdb   0.000GB
config  0.000GB
local   0.000GB
```

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" }
>
```

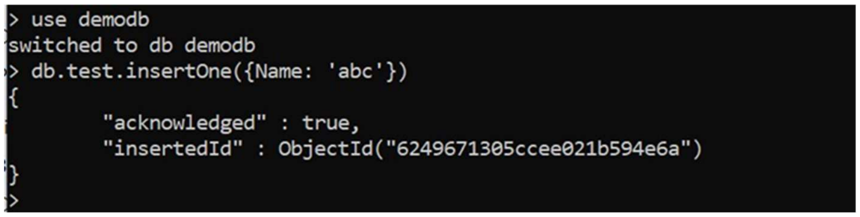
7. So, the document we inserted earlier is shown in the above result. If we want it in a more readable format we can use the pretty() function.



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

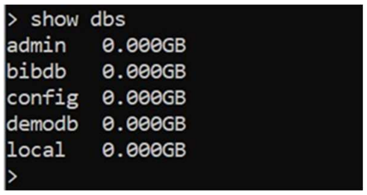
DELETE/DROP Databases:

8. We know how to create a database. Now let's see how to delete/drop a database with dropDatabase(). Here, we already have a created sample database "demodb" with a document in it.



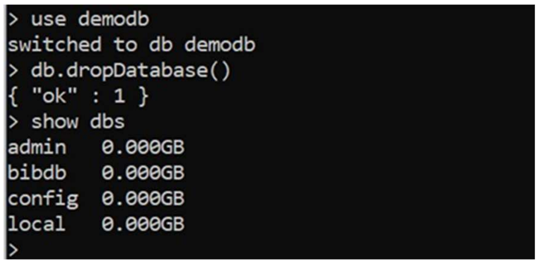
```
> use demodb
switched to db demodb
> db.test.insertOne({Name: 'abc'})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6249671305ccee021b594e6a")
}
>
```

Before:



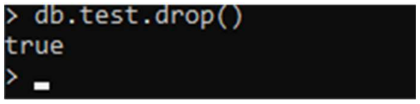
```
> show dbs
admin   0.000GB
bibdb   0.000GB
config  0.000GB
demodb  0.000GB
local   0.000GB
>
```

After:



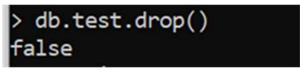
```
> use demodb
switched to db demodb
> db.dropDatabase()
{ "ok" : 1 }
> show dbs
admin   0.000GB
bibdb   0.000GB
config  0.000GB
local   0.000GB
>
```

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:



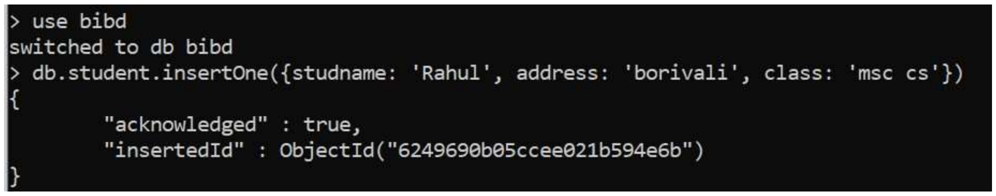
```
> db.test.drop()
false
```

The Basic CRUD operations are Create, Read, Update & Delete:

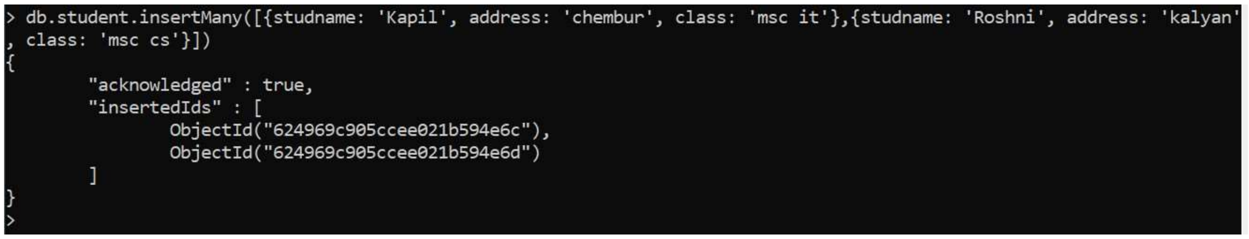
1. The ***Create*** commands are "***insertOne(data, options)"*** & "***insertMany([data], options)".***
2. The ***Read*** commands are "***find(filter, options)***" & "***findOne(filter, options)***".
3. The ***Update*** commands are "***updateOne(filter, data, options)***" ; "***updateMany(filter, data, options)***" & "***replaceOne(filter, data, options)***".
4. The ***Delete*** commands are "***deleteOne(filter, options)***" & "***deleteMany(filter, options)***".

❖ *Executing the CRUD Operations:*

- ***insertOne*** and ***insertMany*** commands:

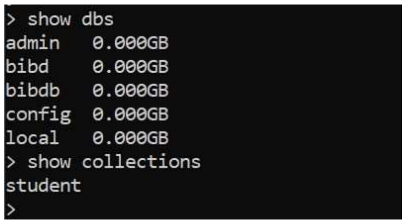


```
> use bibd
switched to db bibd
> db.student.insertOne({studname: 'Rahul', address: 'borivali', class: 'msc cs'})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6249690b05ccee021b594e6b")
}
```



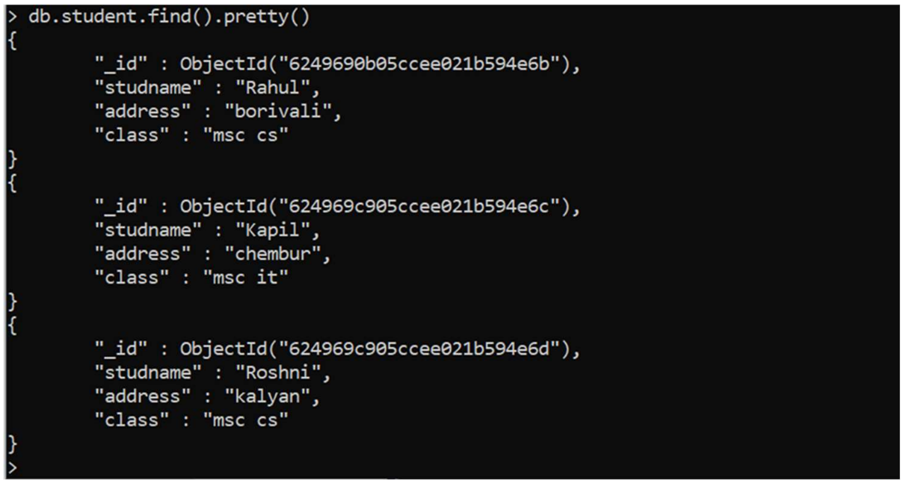
```
> db.student.insertMany([{studname: 'Kapil', address: 'chembur', class: 'msc it'},{studname: 'Roshni', address: 'kalyan'
, class: 'msc cs'}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("624969c905ccee021b594e6c"),
                ObjectId("624969c905ccee021b594e6d")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

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
>
```

Here check the records/document we have updated in the collection student

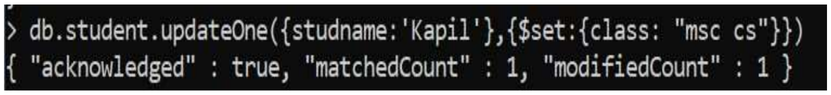


```
> db.student.find().pretty()
{
        "_id" : ObjectId("6249690b05ccee021b594e6b"),
        "studname" : "Rahul",
        "address" : "borivali",
        "class" : "msc cs"
}
{
        "_id" : ObjectId("624969c905ccee021b594e6c"),
        "studname" : "Kapil",
        "address" : "chembur",
   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

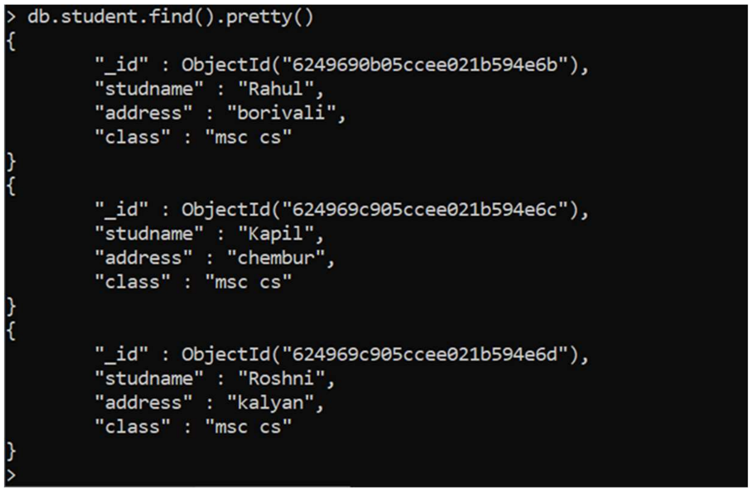
Here, we've successfully executed the insertOne and insertMany commands.

- ***updateOne*** and ***updateMany*** command:

Now let's try updating the class of Kapil to MSC CS in the document and check if the value is updated:

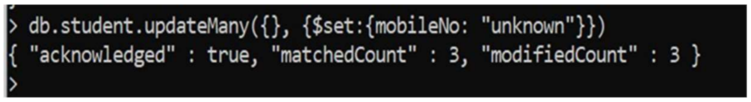


```
> db.student.updateOne({studname:'Kapil'},{$set:{class: "msc cs"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

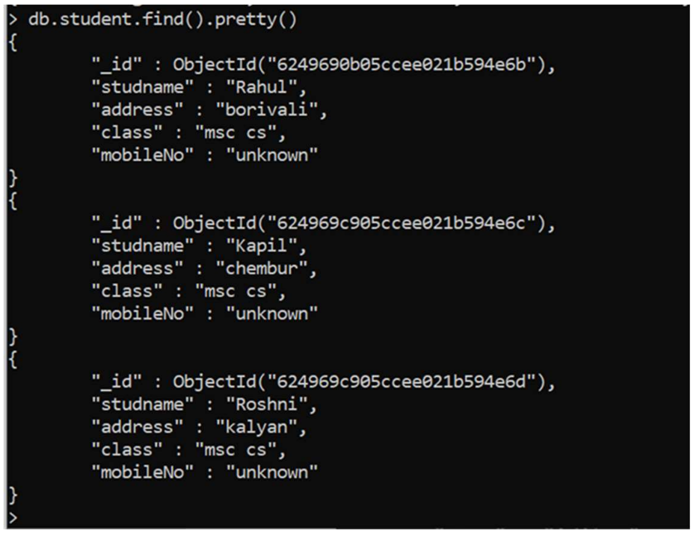


```
> db.student.find().pretty()
{
        "_id" : ObjectId("6249690b05ccee021b594e6b"),
        "studname" : "Rahul",
        "address" : "borivali",
        "class" : "msc cs"
}
{
        "_id" : ObjectId("624969c905ccee021b594e6c"),
        "studname" : "Kapil",
        "address" : "chembur",
   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

Now lets try updateMany command with keeping the first parameter blank which means updating all the entries

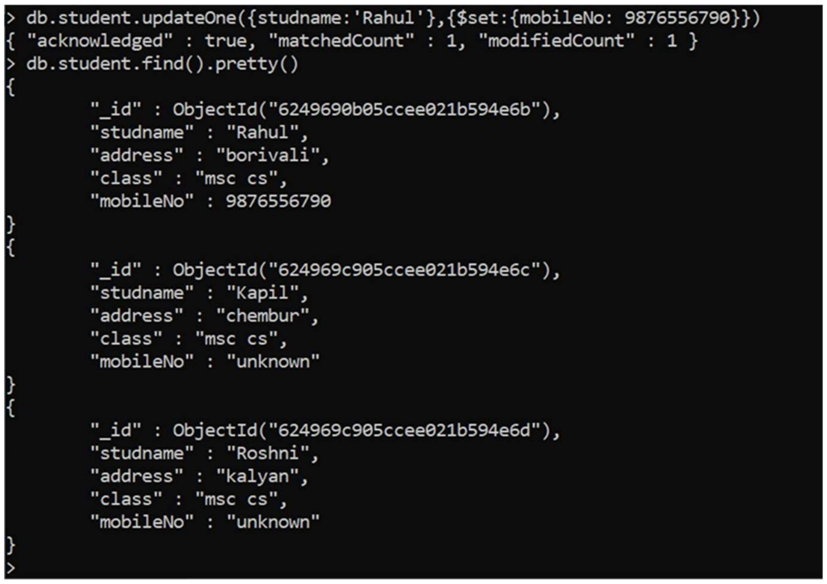


```
> db.student.updateMany({}, {$set:{mobileNo: "unknown"}})
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }
>
```



```
> 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"
}
>
```

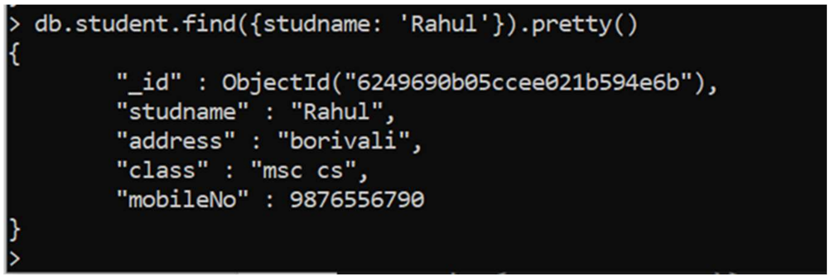
Now let's change the status of one student



```
> 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"
}
>
```

- ***find*** operation:

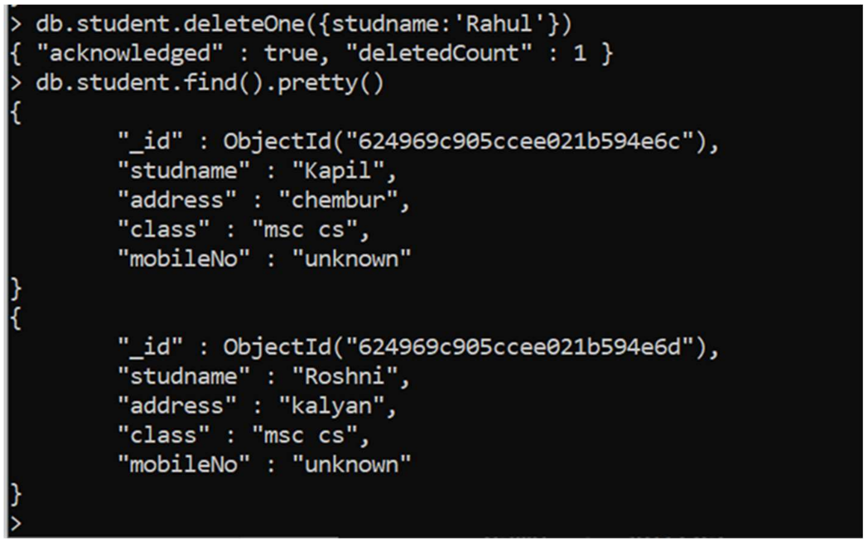
Now using the Find command to find an entry with a particular tag



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

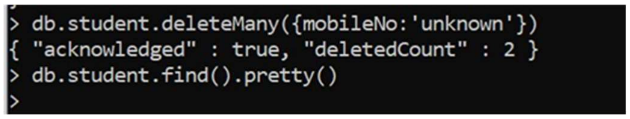
- ***deleteOne*** and ***deleteMany*** operations:

So now let's delete an entry from an employee using deleteOne().



```
> 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"
}
>
```

Now deleting users with deleteMany() operations where mobile number is unknown and hence all records are deleted and hence, we now have an empty collection.



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

These are all CRUD operations implemented in MongoDB.