Experiment No.: 05

AIM: Implementation of NoSQL (MongoDB) commands.

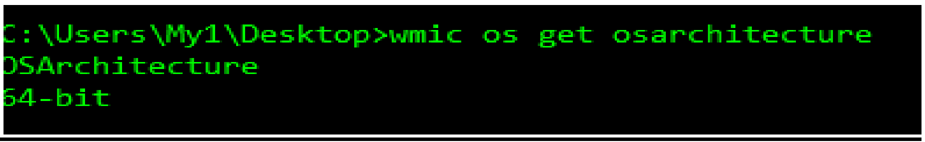
THEORY:

1. What is MongoDB?
2. Why MongoDB?
3. What is a database in MongoDB?
4. What are collections in MongoDB?
5. What are the different datatypes in MongoDB?

OUTPUT:

1. To get your windows version –

wmic os get osarchitecture

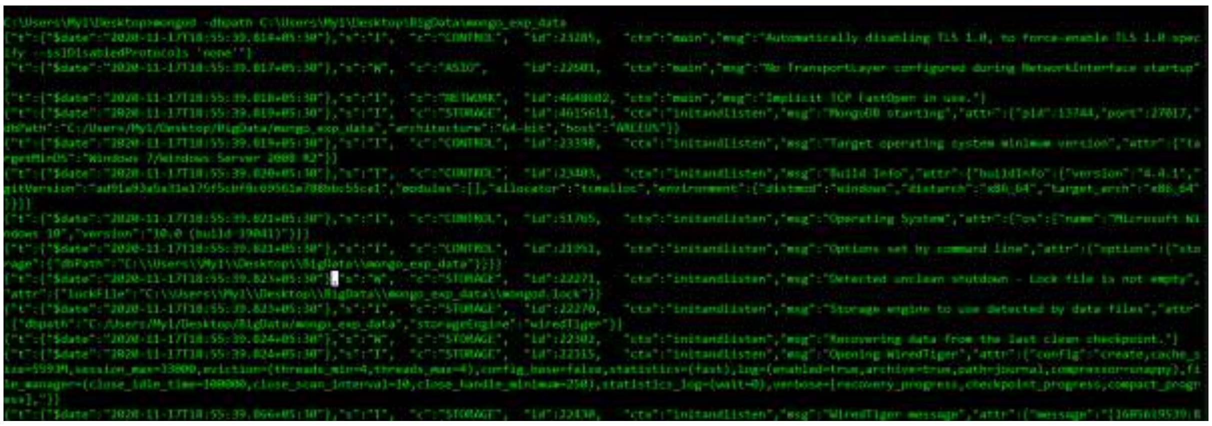


```
C:\Users\My1\Desktop>wmic os get osarchitecture
OSArchitecture
64-bit
```

2. Connecting to Server –

c:\mongodb\bin\mongod.exe –dbpath c:\mongodb\data\db

This command started the server

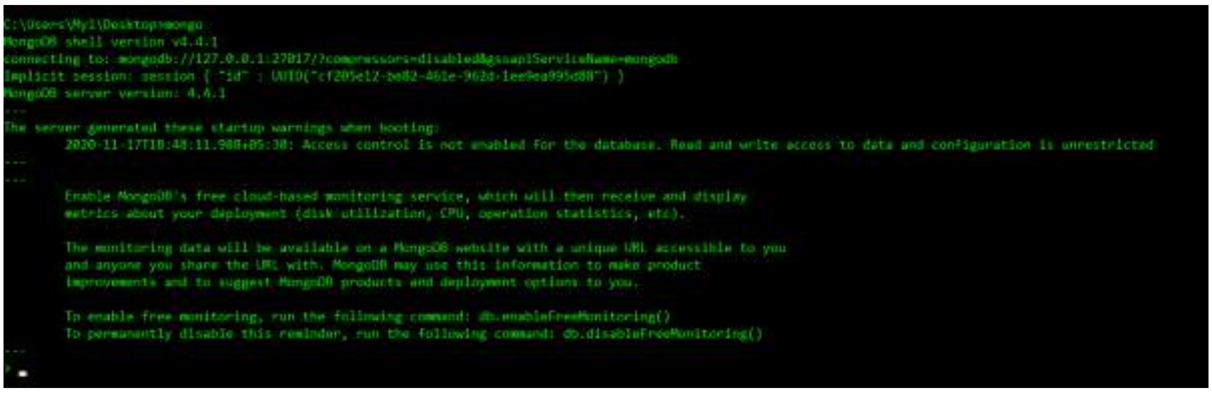


```
C:\Users\My1\Desktop>mongod -dbpath C:\Users\My1\Desktop\BigData\mongo_exp_data
{"t":{"$date":"2020-11-17T18:55:39.814+05:30"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 spec
ify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2020-11-17T18:55:39.817+05:30"},"s":"W",  "c":"ASIO",     "id":22601,   "ctx":"main","msg":"No TransportLayer configured during NetworkInterface startup"
}
{"t":{"$date":"2020-11-17T18:55:39.818+05:30"},"s":"I",  "c":"NETWORK",  "id":4648602, "ctx":"main","msg":"Implicit TCP FastOpen in use."}
{"t":{"$date":"2020-11-17T18:55:39.819+05:30"},"s":"I",  "c":"STORAGE",  "id":4615611, "ctx":"initandlisten","msg":"MongoDB starting","attr":{"pid":13744,"port":27017,"
dbPath":"C:/Users/My1/Desktop/BigData/mongo_exp_data","architecture":"64-bit","host":"ARLEUS"}}
{"t":{"$date":"2020-11-17T18:55:39.819+05:30"},"s":"I",  "c":"CONTROL",  "id":23398,   "ctx":"initandlisten","msg":"Target operating system minimum version","attr":{"ta
rgetMinOS":"Windows 7/Windows Server 2008 R2"}}
{"t":{"$date":"2020-11-17T18:55:39.820+05:30"},"s":"I",  "c":"CONTROL",  "id":23403,   "ctx":"initandlisten","msg":"Build Info","attr":{"buildInfo":{"version":"4.4.1","
gitVersion":"ad91a93a5a31e175f5cbf8c69561e788bbc55ce1","modules":[],"allocator":"tcmalloc","environment":{"distmod":"windows","distarch":"x86_64","target_arch":"x86_64"
}}}}
{"t":{"$date":"2020-11-17T18:55:39.821+05:30"},"s":"I",  "c":"CONTROL",  "id":51765,   "ctx":"initandlisten","msg":"Operating System","attr":{"os":{"name":"Microsoft Wi
ndows 10","version":"10.0 (build 19041)"}}}
{"t":{"$date":"2020-11-17T18:55:39.821+05:30"},"s":"I",  "c":"CONTROL",  "id":21951,   "ctx":"initandlisten","msg":"Options set by command line","attr":{"options":{"sto
rage":{"dbPath":"C:\\Users\\My1\\Desktop\\BigData\\mongo_exp_data"}}}}
{"t":{"$date":"2020-11-17T18:55:39.823+05:30"},"s":"W",  "c":"STORAGE",  "id":22271,   "ctx":"initandlisten","msg":"Detected unclean shutdown - Lock file is not empty",
"attr":{"lockFile":"C:\\Users\\My1\\Desktop\\BigData\\mongo_exp_data\\mongod.lock"}}
{"t":{"$date":"2020-11-17T18:55:39.823+05:30"},"s":"I",  "c":"STORAGE",  "id":22270,   "ctx":"initandlisten","msg":"Storage engine to use detected by data files","attr"
:{"dbpath":"C:/Users/My1/Desktop/BigData/mongo_exp_data","storageEngine":"wiredTiger"}}
{"t":{"$date":"2020-11-17T18:55:39.824+05:30"},"s":"W",  "c":"STORAGE",  "id":22302,   "ctx":"initandlisten","msg":"Recovering data from the last clean checkpoint."}
{"t":{"$date":"2020-11-17T18:55:39.824+05:30"},"s":"I",  "c":"STORAGE",  "id":22315,   "ctx":"initandlisten","msg":"Opening WiredTiger","attr":{"config":"create,cache_s
ize=5593M,session_max=33000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),fi
le_manager=(close_idle_time=100000,close_scan_interval=10,close_handle_minimum=250),statistics_log=(wait=0),verbose=[recovery_progress,checkpoint_progress,compact_progr
ess],"}}
{"t":{"$date":"2020-11-17T18:55:39.866+05:30"},"s":"I",  "c":"STORAGE",  "id":22430,   "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":"[1605619539:8
```

3. Connecting to Client –

c:\mongodb\bin\mongo.exe

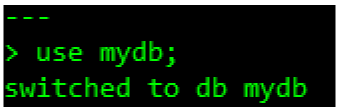
This command started the client



```
C:\Users\My1\Desktop>mongo
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("cf205e12-be82-461e-962d-1ee9ea995d88") }
MongoDB server version: 4.4.1
---
The server generated these startup warnings when booting:
        2020-11-17T18:48:11.988+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()
---
> 
```

4. Create database –

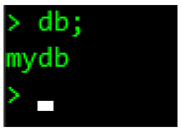
   > use mydb



```
---
> use mydb;
switched to db mydb
```

5. Confirm the existence of your database

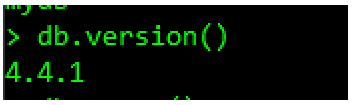
   >db;



```
> db;
mydb
>
```

6. To display the current version of MongoDB Server

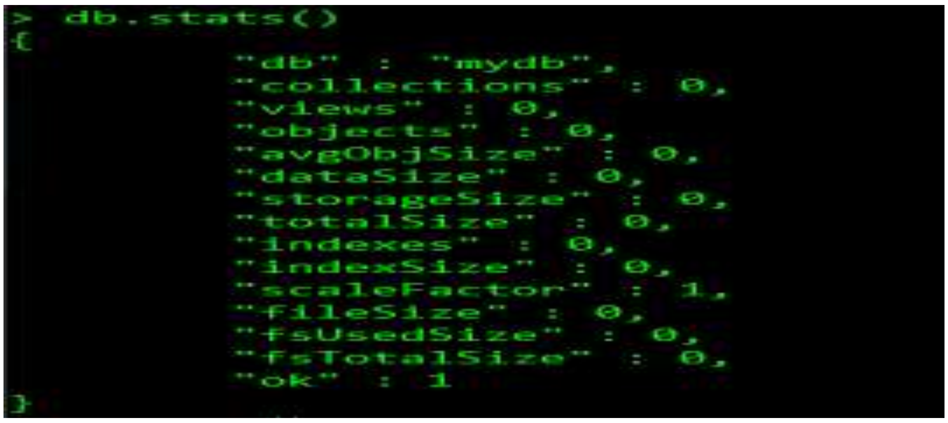
   >db.version()



```
> db.version()
4.4.1
```

7. To display the statistics that reflect the use state of the database

   >db.stats()



```
> db.stats()
{
        "db" : "mydb",
        "collections" : 0,
        "views" : 0,
        "objects" : 0,
        "avgObjSize" : 0,
        "dataSize" : 0,
        "storageSize" : 0,
        "totalSize" : 0,
        "indexes" : 0,
        "indexSize" : 0,
        "scaleFactor" : 1,
        "fileSize" : 0,
        "fsUsedSize" : 0,
        "fsTotalSize" : 0,
        "ok" : 1
}
```

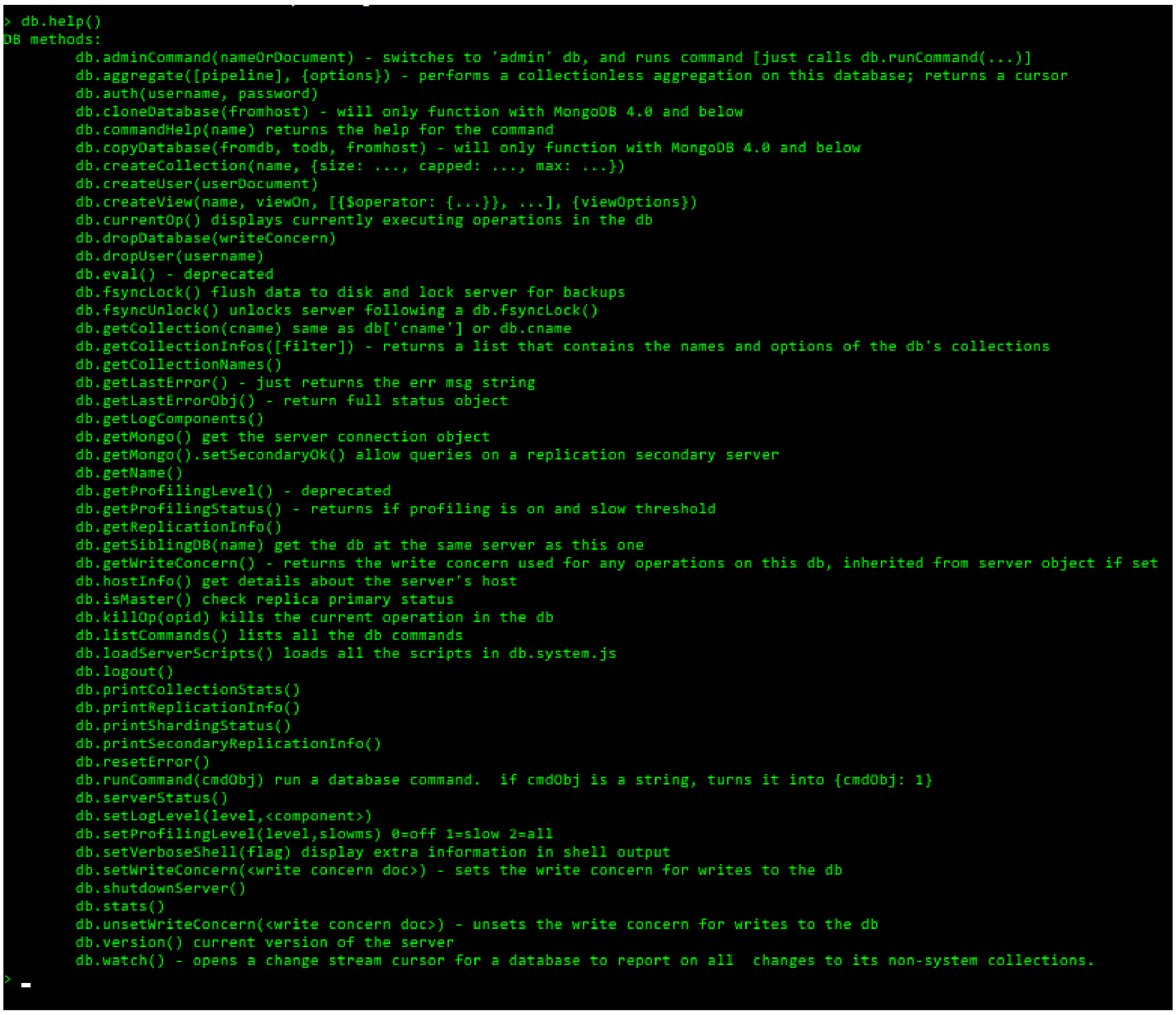
   explain what do you understand by stats

   The stats tell us about everything stored like dbs, collections, view objects, and it also tells us about the storage that is being used by the whole client. A boolean value ok tells us if system is working correctly.

8. To display the list of commands

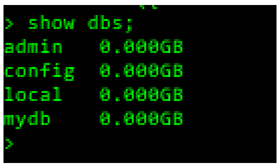
   >db.help()

   This command is used to give us all the help required for the commands used on mongo client. It tells us all DB methods that we can use.



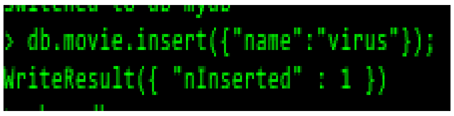
```
> db.help()
DB methods:
        db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command [just calls db.runCommand(...)]
        db.aggregate([pipeline], {options}) - performs a collectionless aggregation on this database; returns a cursor
        db.auth(username, password)
        db.cloneDatabase(fromhost) - will only function with MongoDB 4.0 and below
        db.commandHelp(name) returns the help for the command
        db.copyDatabase(fromdb, todb, fromhost) - will only function with MongoDB 4.0 and below
        db.createCollection(name, {size: ..., capped: ..., max: ...})
        db.createUser(userDocument)
        db.createView(name, viewOn, [{$operator: {...}}, ...], {viewOptions})
        db.currentOp() displays currently executing operations in the db
        db.dropDatabase(writeConcern)
        db.dropUser(username)
        db.eval() - deprecated
        db.fsyncLock() flush data to disk and lock server for backups
        db.fsyncUnlock() unlocks server following a db.fsyncLock()
        db.getCollection(cname) same as db['cname'] or db.cname
        db.getCollectionInfos([filter]) - returns a list that contains the names and options of the db's collections
        db.getCollectionNames()
        db.getLastError() - just returns the err msg string
        db.getLastErrorObj() - return full status object
        db.getLogComponents()
        db.getMongo() get the server connection object
        db.getMongo().setSecondaryOk() allow queries on a replication secondary server
        db.getName()
        db.getProfilingLevel() - deprecated
        db.getProfilingStatus() - returns if profiling is on and slow threshold
        db.getReplicationInfo()
        db.getSiblingDB(name) get the db at the same server as this one
        db.getWriteConcern() - returns the write concern used for any operations on this db, inherited from server object if set
        db.hostInfo() get details about the server's host
        db.isMaster() check replica primary status
        db.killOp(opid) kills the current operation in the db
        db.listCommands() lists all the db commands
        db.loadServerScripts() loads all the scripts in db.system.js
        db.logout()
        db.printCollectionStats()
        db.printReplicationInfo()
        db.printShardingStatus()
        db.printSecondaryReplicationInfo()
        db.resetError()
        db.runCommand(cmdObj) run a database command.  if cmdObj is a string, turns it into {cmdObj: 1}
        db.serverStatus()
        db.setLogLevel(level,<component>)
        db.setProfilingLevel(level,slowms) 0=off 1=slow 2=all
        db.setVerboseShell(flag) display extra information in shell output
        db.setWriteConcern(<write concern doc>) - sets the write concern for writes to the db
        db.shutdownServer()
        db.stats()
        db.unsetWriteConcern(<write concern doc>) - unsets the write concern for writes to the db
        db.version() current version of the server
        db.watch() - opens a change stream cursor for a database to report on all  changes to its non-system collections.
> 
```

9. <u>To get a list of all databases</u>
   >show dbs;



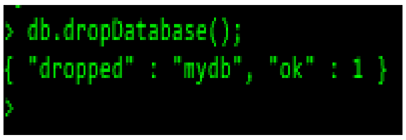
```
> show dbs;
admin    0.000GB
config   0.000GB
local    0.000GB
mydb     0.000GB
>
```

10. <u>To insert a record</u>
    db.movie.insert({"name":"Virus"});



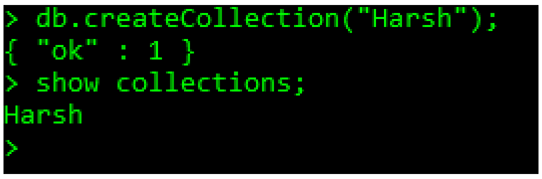
```
> db.movie.insert({"name":"virus"});
WriteResult({ "nInserted" : 1 })
```

11. <u>To drop a database</u>
    >**use mydb;**
    >db.dropDatabase();



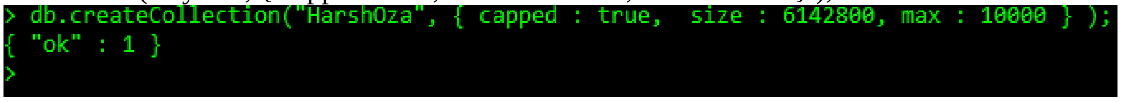
```
> db.dropDatabase();
{ "dropped" : "mydb", "ok" : 1 }
>
```

12. <u>To display the list of collections:-</u>
<u>Please mention what do you mean by collections. The general structure of collection .</u>
> Create collection
> db.createCollection("myCollection")
>**show collections**



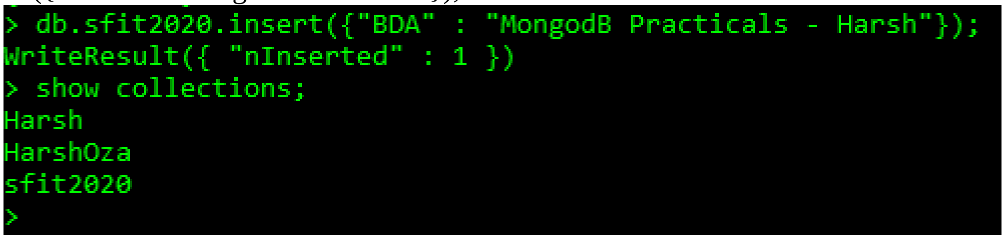
```
> db.createCollection("Harsh");
{ "ok" : 1 }
> show collections;
Harsh
>
```

13. Create collection with some options
    db.createCollection("mycol", { capped : true, size : 6142800, max : 10000 });



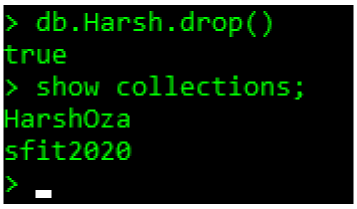
```
> db.createCollection("HarshOza", { capped : true,  size : 6142800, max : 10000 } );
{ "ok" : 1 }
>
```

14. db.sfit2020.insert({"BDA" : "MongodB Practicals"});



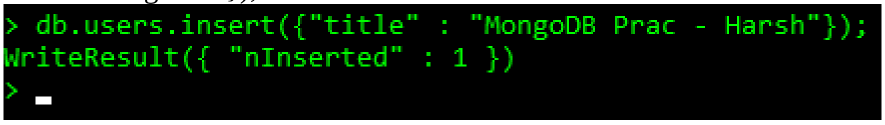
```
> db.sfit2020.insert({"BDA" : "MongodB Practicals - Harsh"});
WriteResult({ "nInserted" : 1 })
> show collections;
Harsh
HarshOza
sfit2020
>
```

15. <u>To drop a collection</u>
    db.COLLECTION_NAME.drop()



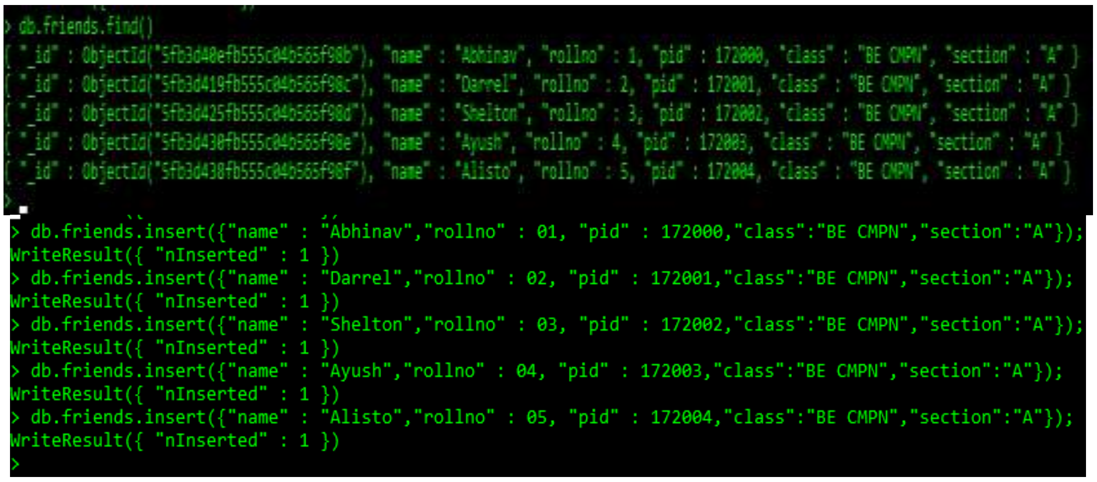
```
> db.Harsh.drop()
true
> show collections;
HarshOza
sfit2020
> _
```

16. <u>To insert data into MongodB insert() or save() method is used</u>
    db.users.insert({"title" : "MongoDB"});



```
> db.users.insert({"title" : "MongoDB Prac - Harsh"});
WriteResult({ "nInserted" : 1 })
> _
```

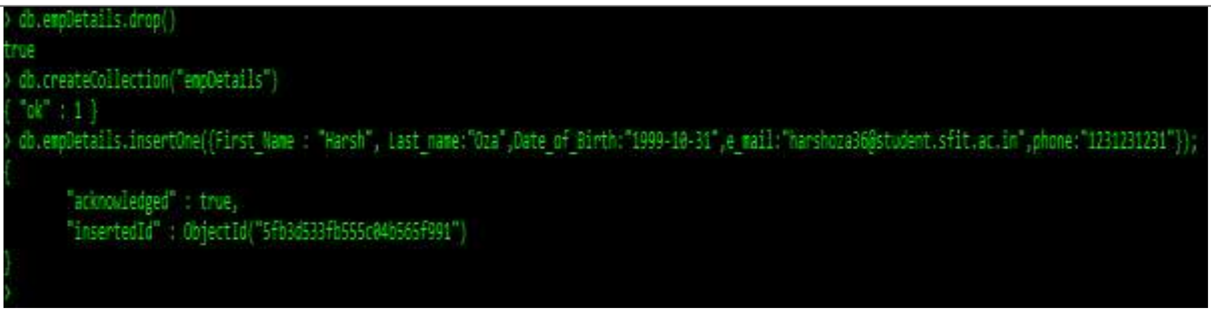
17. <u>To insert an array of documents</u>
    Try inserting 5 of ur friends details like name, rollno, pid, class, section.



```
> db.friends.find()
{ "_id" : ObjectId("5fb3d40efb555c04b565f98b"), "name" : "Abhinav", "rollno" : 1, "pid" : 172000, "class" : "BE CMPN", "section" : "A" }
{ "_id" : ObjectId("5fb3d419fb555c04b565f98c"), "name" : "Darrel", "rollno" : 2, "pid" : 172001, "class" : "BE CMPN", "section" : "A" }
{ "_id" : ObjectId("5fb3d425fb555c04b565f98d"), "name" : "Shelton", "rollno" : 3, "pid" : 172002, "class" : "BE CMPN", "section" : "A" }
{ "_id" : ObjectId("5fb3d430fb555c04b565f98e"), "name" : "Ayush", "rollno" : 4, "pid" : 172003, "class" : "BE CMPN", "section" : "A" }
{ "_id" : ObjectId("5fb3d438fb555c04b565f98f"), "name" : "Alisto", "rollno" : 5, "pid" : 172004, "class" : "BE CMPN", "section" : "A" }
>
> db.friends.insert({"name" : "Abhinav","rollno" : 01, "pid" : 172000,"class":"BE CMPN","section":"A"});
WriteResult({ "nInserted" : 1 })
> db.friends.insert({"name" : "Darrel","rollno" : 02, "pid" : 172001,"class":"BE CMPN","section":"A"});
WriteResult({ "nInserted" : 1 })
> db.friends.insert({"name" : "Shelton","rollno" : 03, "pid" : 172002,"class":"BE CMPN","section":"A"});
WriteResult({ "nInserted" : 1 })
> db.friends.insert({"name" : "Ayush","rollno" : 04, "pid" : 172003,"class":"BE CMPN","section":"A"});
WriteResult({ "nInserted" : 1 })
> db.friends.insert({"name" : "Alisto","rollno" : 05, "pid" : 172004,"class":"BE CMPN","section":"A"});
WriteResult({ "nInserted" : 1 })
>
```

18. insertOne is used to insert only one document.

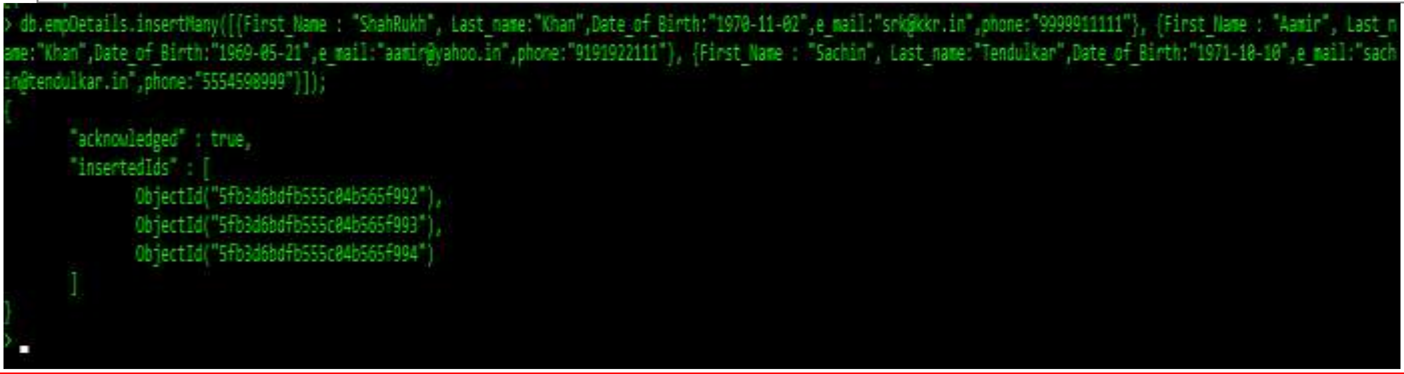
```
     db.createCollection("empDetails")
db.empDetails.insertOne(
          {
                    First_Name: "Radhika",
                    Last_Name: "Sharma",
                    Date_Of_Birth: "1995-09-26",
                e_mail: "radhika_sharma.123@gmail.com",
                phone: "9848022338"
        })
```



```
> db.empDetails.drop()
true
> db.createCollection("empDetails")
{ "ok" : 1 }
> db.empDetails.insertOne({First_Name : "Harsh", Last_name:"Oza",Date_of_Birth:"1999-10-31",e_mail:"harshoza36@student.sfit.ac.in",phone:"1231231231"});
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5fb3d533fb555c04b565f991")
}
>
```

19. To insert multiple documents insertMultiple is used.

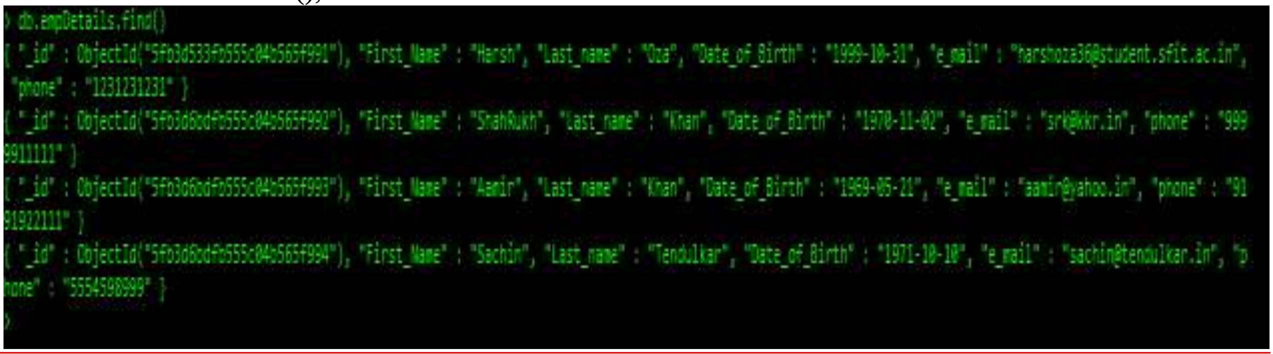
```
db.empDetails.insertMany(
      [
                {
                        First_Name: "Radhika",
                        Last_Name: "Sharma",
                        Date_Of_Birth: "1995-09-26",
                        e_mail: "radhika_sharma.123@gmail.com",
                        phone: "9000012345"
                },
                {
                        First_Name: "Rachel",
                        Last_Name: "Christopher",
                        Date_Of_Birth: "1990-02-16",
                        e_mail: "Rachel_Christopher.123@gmail.com",
                        phone: "9000054321"
                },
                {
                        First_Name: "Fathima",
                        Last_Name: "Sheik",
                        Date_Of_Birth: "1990-02-16",
                        e_mail: "Fathima_Sheik.123@gmail.com",
                        phone: "9000054321"
                }
      ]
)
```



```
> db.empDetails.insertMany([{First_Name : "ShahRukh", Last_name:"Khan",Date_of_Birth:"1970-11-02",e_mail:"srk@kkr.in",phone:"9999911111"}, {First_Name : "Aamir", Last_n
ame:"Khan",Date_of_Birth:"1969-05-21",e_mail:"aamir@yahoo.in",phone:"9191922111"}, {First_Name : "Sachin", Last_name:"Tendulkar",Date_of_Birth:"1971-10-10",e_mail:"sach
in@tendulkar.in",phone:"5554598999"}]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5fb3d6bdfb555c04b565f992"),
                ObjectId("5fb3d6bdfb555c04b565f993"),
                ObjectId("5fb3d6bdfb555c04b565f994")
        ]
}
> ▪
```

20. To query data from MongoDB collection, you need to use MongoDB's **find()** method.

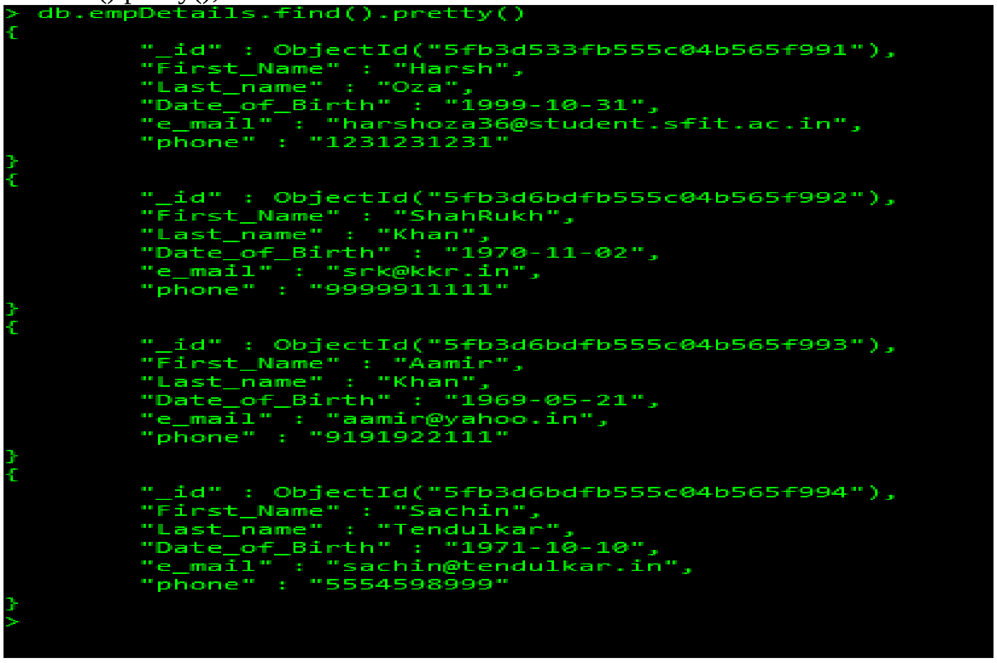
db.Students.find();



```
> db.empDetails.find()
{ "_id" : ObjectId("5fb3d533fb555c04b565f991"), "First_Name" : "Harsh", "Last_name" : "Oza", "Date_of_Birth" : "1999-10-31", "e_mail" : "harshoza36@student.sfit.ac.in",
 "phone" : "1231231231" }
{ "_id" : ObjectId("5fb3d6bdfb555c04b565f992"), "First_Name" : "ShahRukh", "Last_name" : "Khan", "Date_of_Birth" : "1970-11-02", "e_mail" : "srk@kkr.in", "phone" : "999
9911111" }
{ "_id" : ObjectId("5fb3d6bdfb555c04b565f993"), "First_Name" : "Aamir", "Last_name" : "Khan", "Date_of_Birth" : "1969-05-21", "e_mail" : "aamir@yahoo.in", "phone" : "91
91922111" }
{ "_id" : ObjectId("5fb3d6bdfb555c04b565f994"), "First_Name" : "Sachin", "Last_name" : "Tendulkar", "Date_of_Birth" : "1971-10-10", "e_mail" : "sachin@tendulkar.in", "p
hone" : "5554598999" }
>
```

21. To display the results in a formatted way, you can use pretty() method.

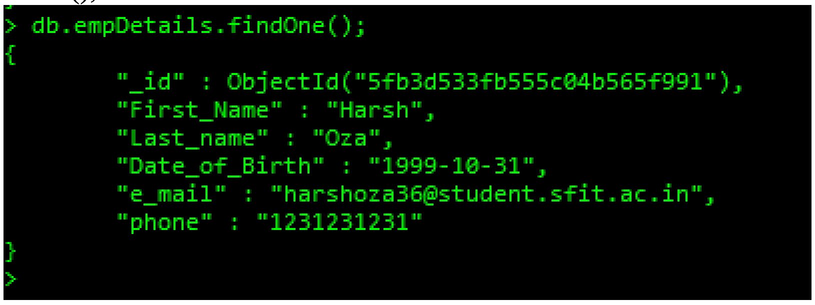
db.Students.find().pretty();



```
> db.empDetails.find().pretty()
{
        "_id" : ObjectId("5fb3d533fb555c04b565f991"),
        "First_Name" : "Harsh",
        "Last_name" : "Oza",
        "Date_of_Birth" : "1999-10-31",
        "e_mail" : "harshoza36@student.sfit.ac.in",
        "phone" : "1231231231"
}
{
        "_id" : ObjectId("5fb3d6bdfb555c04b565f992"),
        "First_Name" : "ShahRukh",
        "Last_name" : "Khan",
        "Date_of_Birth" : "1970-11-02",
        "e_mail" : "srk@kkr.in",
        "phone" : "9999911111"
}
{
        "_id" : ObjectId("5fb3d6bdfb555c04b565f993"),
        "First_Name" : "Aamir",
        "Last_name" : "Khan",
        "Date_of_Birth" : "1969-05-21",
        "e_mail" : "aamir@yahoo.in",
        "phone" : "9191922111"
}
{
        "_id" : ObjectId("5fb3d6bdfb555c04b565f994"),
        "First_Name" : "Sachin",
        "Last_name" : "Tendulkar",
        "Date_of_Birth" : "1971-10-10",
        "e_mail" : "sachin@tendulkar.in",
        "phone" : "5554598999"
}
>
```

22. To display only one record the findONe method is used.

db.empDetails.findOne();



```
> db.empDetails.findOne();
{
        "_id" : ObjectId("5fb3d533fb555c04b565f991"),
        "First_Name" : "Harsh",
        "Last_name" : "Oza",
        "Date_of_Birth" : "1999-10-31",
        "e_mail" : "harshoza36@student.sfit.ac.in",
        "phone" : "1231231231"
}
>
```

23. Different variations of where clause and its equivalent used in Mongodb

Operation	Syntax	Example	RDBMS Equivalent
Equality	{<key>:{$eg;<value>}}	db.mycol.find({"by":"tutorials point"}).pretty()	where by = 'tutorials point'
Less Than	{<key>:{$lt:<value>}}	db.mycol.find({"likes":{$lt:50}}).pretty()	where likes < 50
Less Than Equals	{<key>:{$lte:<value>}}	db.mycol.find({"likes":{$lte:50}}).pretty()	where likes <= 50
Greater Than	{<key>:{$gt:<value>}}	db.mycol.find({"likes":{$gt:50}}).pretty()	where likes > 50
Greater Than Equals	{<key>:{$gte:<value>}}	db.mycol.find({"likes":{$gte:50}}).pretty()	where likes >= 50
Not Equals	{<key>:{$ne:<value>}}	db.mycol.find({"likes":{$ne:50}}).pretty()	where likes != 50

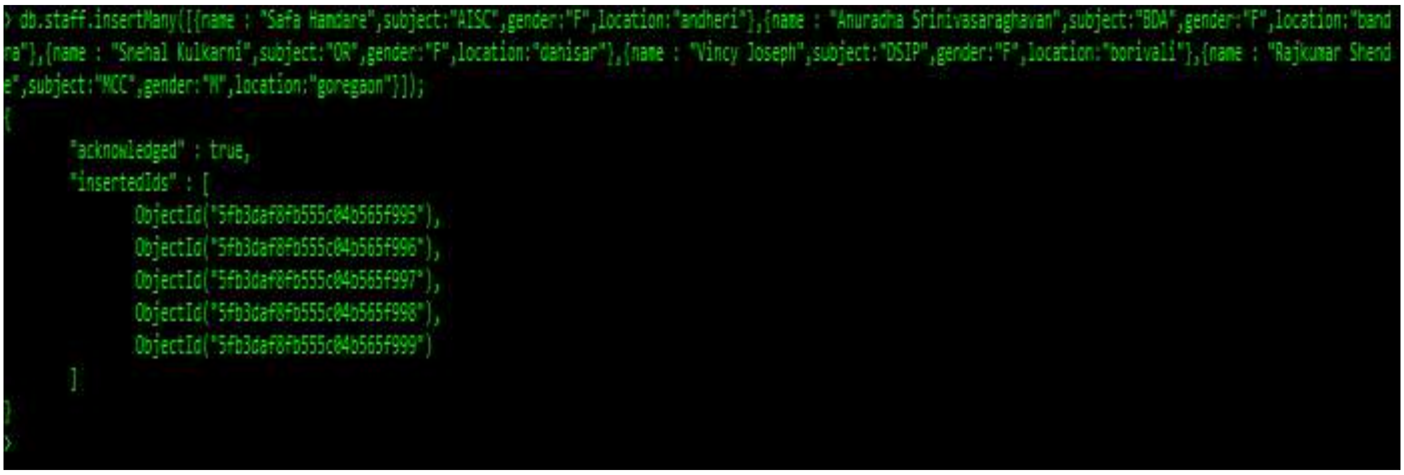
Values in an array	{<key>:{$in:[<value1>, <value2>,……<valueN>]}}	db.mycol.find({"name":{$in:["Raj", "Ram", "Raghu"]}}).pretty()	Where name matches any of the value in :["Raj", "Ram", "Raghu"]
Values not in an array	{<key>:{$nin:<value>}}	db.mycol.find({"name":{$nin:["Ramu", "Raghav"]}}).pretty()	Where name values is not in the array :["Ramu", "Raghav"] or, doesn't exist at al

Do the following

a. Create a collection of staff and include some of the staffs handling ur subjects in this semester.

b. All the particulars of the staff like name, subject handling, gender, etc. can be included and try the various querying options.

```
db.empDetails.insertMany(
     [
              {
                       First_Name: "Radhika",
                       Last_Name: "Sharma",
                       Age: "26",
                       e_mail: "radhika_sharma.123@gmail.com",
                       phone: "9000012345"
              },
              {
                       First_Name: "Rachel",
                       Last_Name: "Christopher",
                       Age: "27",
                       e_mail:"Rachel_Christopher.123@gmail.com",
                       phone: "9000054321"
              },
              {
                       First_Name: "Fathima",
                       Last_Name: "Sheik",
                       Age: "24",
                       e_mail: "Fathima_Sheik.123@gmail.com",
                       phone: "9000054321"
              }
     ]
)
```

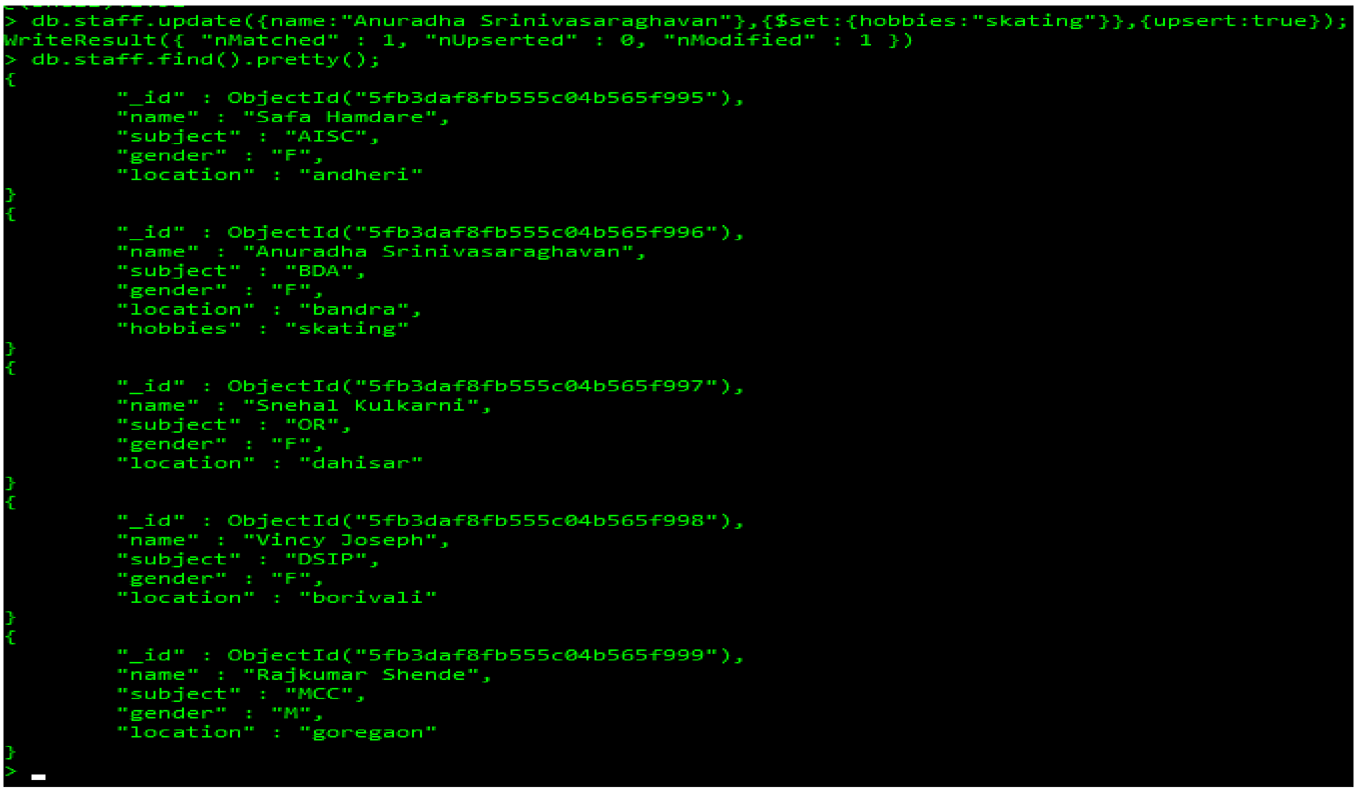


```
> db.staff.insertMany([{name : "Safa Hamdare",subject:"AISC",gender:"F",location:"andheri"},{name : "Anuradha Srinivasaraghavan",subject:"BDA",gender:"F",location:"band
ra"},{name : "Snehal Kulkarni",subject:"OR",gender:"F",location:"dahisar"},{name : "Vincy Joseph",subject:"DSIP",gender:"F",location:"borivali"},{name : "Rajkumar Shend
e",subject:"MCC",gender:"M",location:"goregaon"}]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5fb3daf8fb555c04b565f995"),
                ObjectId("5fb3daf8fb555c04b565f996"),
                ObjectId("5fb3daf8fb555c04b565f997"),
                ObjectId("5fb3daf8fb555c04b565f998"),
                ObjectId("5fb3daf8fb555c04b565f999")
        ]
}
>
```

24. Update –

> db.student.update({_id:'1',studname:'anu'},{$set:{hobbies:'skating'}},{upsert:true});

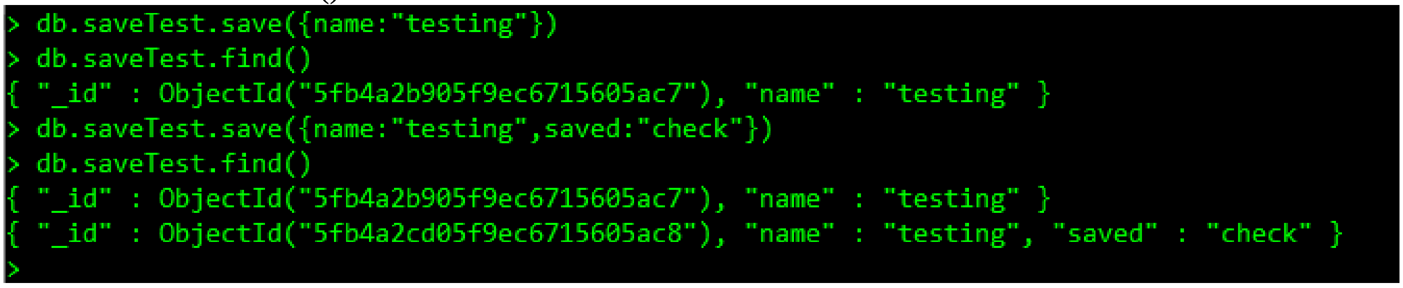
>db.student.find().pretty()



```
> db.staff.update({name:"Anuradha Srinivasaraghavan"},{$set:{hobbies:"skating"}},{upsert:true});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.staff.find().pretty();
{
        "_id" : ObjectId("5fb3daf8fb555c04b565f995"),
        "name" : "Safa Hamdare",
        "subject" : "AISC",
        "gender" : "F",
        "location" : "andheri"
}
{
        "_id" : ObjectId("5fb3daf8fb555c04b565f996"),
        "name" : "Anuradha Srinivasaraghavan",
        "subject" : "BDA",
        "gender" : "F",
        "location" : "bandra",
        "hobbies" : "skating"
}
{
        "_id" : ObjectId("5fb3daf8fb555c04b565f997"),
        "name" : "Snehal Kulkarni",
        "subject" : "OR",
        "gender" : "F",
        "location" : "dahisar"
}
{
        "_id" : ObjectId("5fb3daf8fb555c04b565f998"),
        "name" : "Vincy Joseph",
        "subject" : "DSIP",
        "gender" : "F",
        "location" : "borivali"
}
{
        "_id" : ObjectId("5fb3daf8fb555c04b565f999"),
        "name" : "Rajkumar Shende",
        "subject" : "MCC",
        "gender" : "M",
        "location" : "goregaon"
}
> _
```

25. <u>Save –</u>

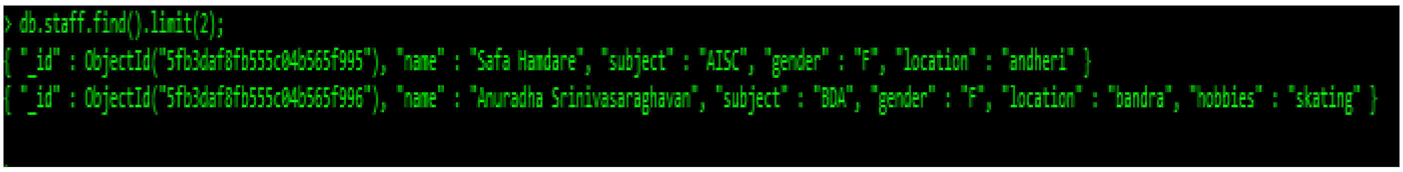
> db.student.save()



```
> db.saveTest.save({name:"testing"})
> db.saveTest.find()
{ "_id" : ObjectId("5fb4a2b905f9ec6715605ac7"), "name" : "testing" }
> db.saveTest.save({name:"testing",saved:"check"})
> db.saveTest.find()
{ "_id" : ObjectId("5fb4a2b905f9ec6715605ac7"), "name" : "testing" }
{ "_id" : ObjectId("5fb4a2cd05f9ec6715605ac8"), "name" : "testing", "saved" : "check" }
>
```

26. <u>Limit – this limits the display of output</u>

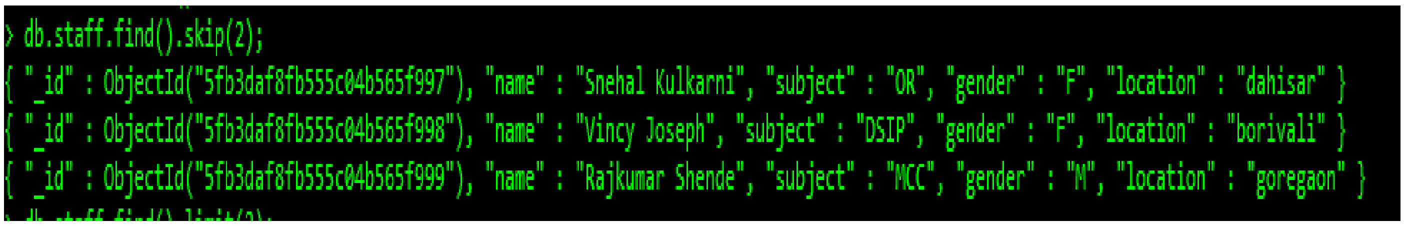
> db.student.find().limit(2)



```
> db.staff.find().limit(2);
{ "_id" : ObjectId("5fb3daf8fb555c04b565f995"), "name" : "Safa Hamdare", "subject" : "AISC", "gender" : "F", "location" : "andheri" }
{ "_id" : ObjectId("5fb3daf8fb555c04b565f996"), "name" : "Anuradha Srinivasaraghavan", "subject" : "BDA", "gender" : "F", "location" : "bandra", "hobbies" : "skating" }
```

27. <u>Skip – This skips the first n records.</u>

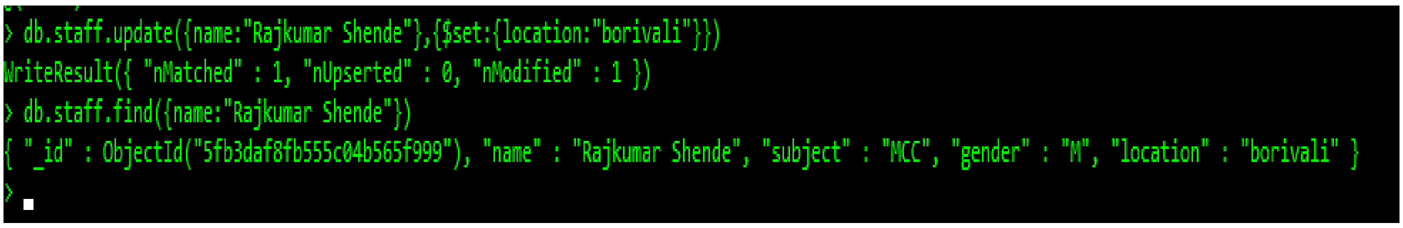
> db.student..find().skip(2)



```
> db.staff.find().skip(2);
{ "_id" : ObjectId("5fb3daf8fb555c04b565f997"), "name" : "Snehal Kulkarni", "subject" : "OR", "gender" : "F", "location" : "dahisar" }
{ "_id" : ObjectId("5fb3daf8fb555c04b565f998"), "name" : "Vincy Joseph", "subject" : "DSIP", "gender" : "F", "location" : "borivali" }
{ "_id" : ObjectId("5fb3daf8fb555c04b565f999"), "name" : "Rajkumar Shende", "subject" : "MCC", "gender" : "M", "location" : "goregaon" }
```

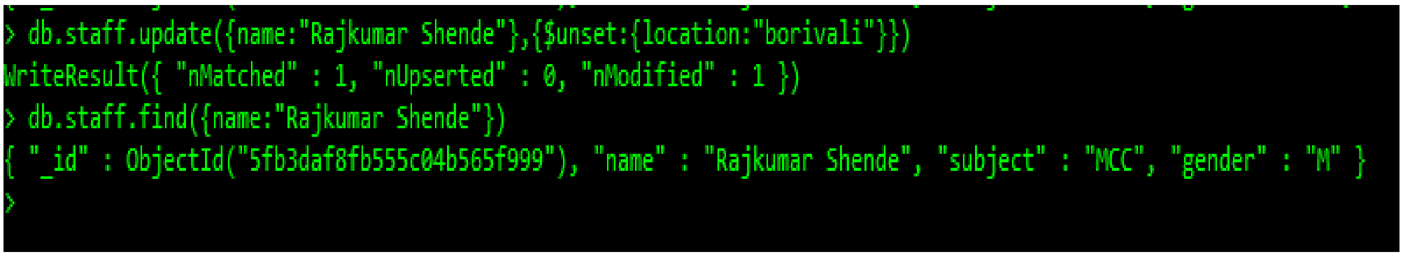
28. <u>Update to add an attribute</u>

> db.student.update({_id:'3'},{$set:{location:'borivali'}});



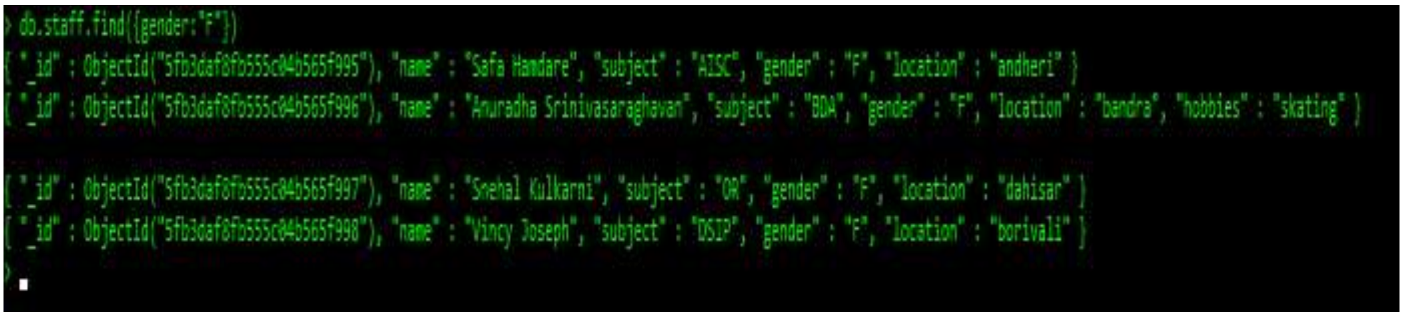
```
> db.staff.update({name:"Rajkumar Shende"},{$set:{location:"borivali"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.staff.find({name:"Rajkumar Shende"})
{ "_id" : ObjectId("5fb3daf8fb555c04b565f999"), "name" : "Rajkumar Shende", "subject" : "MCC", "gender" : "M", "location" : "borivali" }
>
```

29. <u>To remove an attribute use unset</u>
    > db.student.update({_id:'3'},{$unset:{location:'borivali'}});



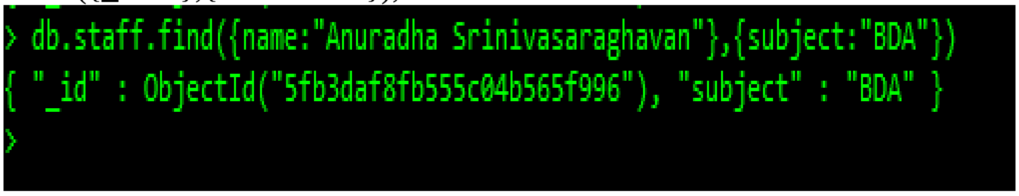
```
> db.staff.update({name:"Rajkumar Shende"},{$unset:{location:"borivali"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.staff.find({name:"Rajkumar Shende"})
{ "_id" : ObjectId("5fb3daf8fb555c04b565f999"), "name" : "Rajkumar Shende", "subject" : "MCC", "gender" : "M" }
>
```

30. <u>Finding documents based on search criteria-find method</u>
    > db.student.find({grade:'VII'});



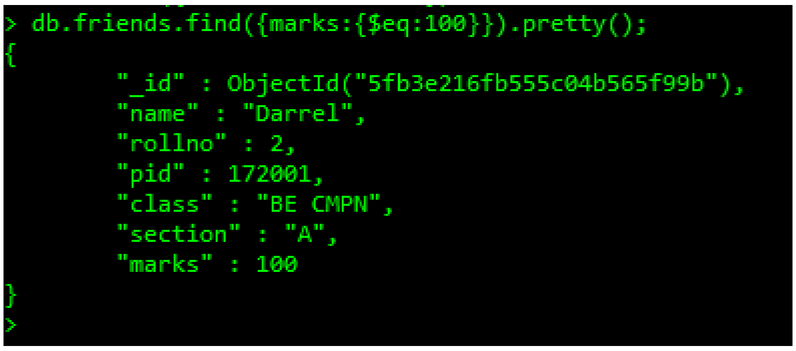
```
> db.staff.find({gender:"F"})
{ "_id" : ObjectId("5fb3daf8fb555c04b565f995"), "name" : "Safa Hamdare", "subject" : "AISC", "gender" : "F", "location" : "andheri" }
{ "_id" : ObjectId("5fb3daf8fb555c04b565f996"), "name" : "Anuradha Srinivasaraghavan", "subject" : "BDA", "gender" : "F", "location" : "bandra", "hobbies" : "skating" }

{ "_id" : ObjectId("5fb3daf8fb555c04b565f997"), "name" : "Snehal Kulkarni", "subject" : "OR", "gender" : "F", "location" : "dahisar" }
{ "_id" : ObjectId("5fb3daf8fb555c04b565f998"), "name" : "Vincy Joseph", "subject" : "DSIP", "gender" : "F", "location" : "borivali" }
>
```

31. <u>Finding projection based on selection operators</u>
    > db.student.find({_id:'1'},{studname:1});



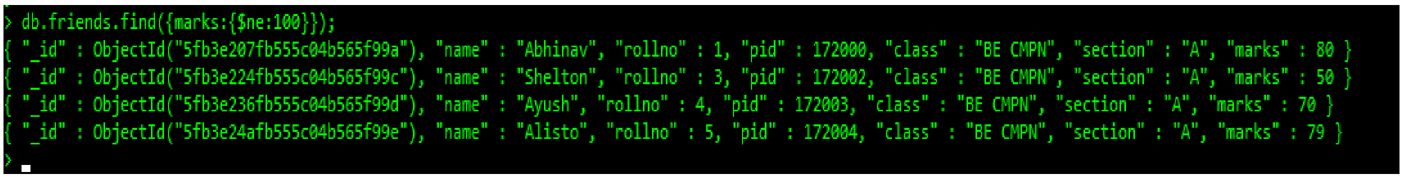
```
> db.staff.find({name:"Anuradha Srinivasaraghavan"},{subject:"BDA"})
{ "_id" : ObjectId("5fb3daf8fb555c04b565f996"), "subject" : "BDA" }
>
```

32. <u>Finding records with same matching criteria</u>. (Equivalent to "=" clause)
    > db.student.find({grade:{$eq:'VII'}}).pretty();
    $eq→equal to



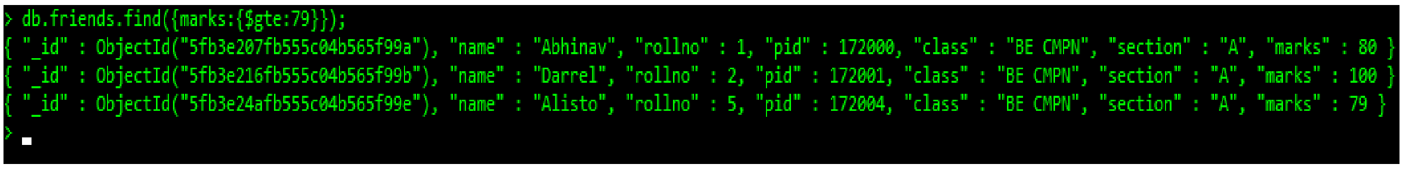
```
> db.friends.find({marks:{$eq:100}}).pretty();
{
        "_id" : ObjectId("5fb3e216fb555c04b565f99b"),
        "name" : "Darrel",
        "rollno" : 2,
        "pid" : 172001,
        "class" : "BE CMPN",
        "section" : "A",
        "marks" : 100
}
>
```

$ne→not equal to



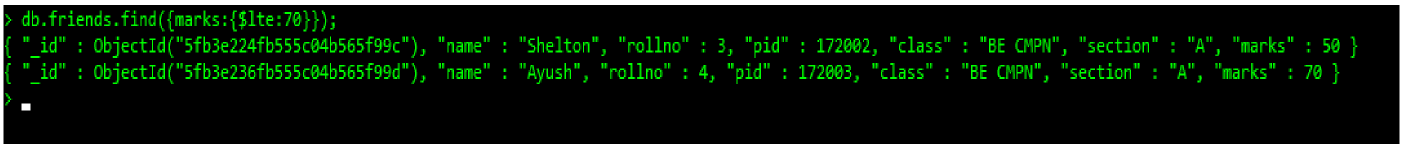
```
> db.friends.find({marks:{$ne:100}});
{ "_id" : ObjectId("5fb3e207fb555c04b565f99a"), "name" : "Abhinav", "rollno" : 1, "pid" : 172000, "class" : "BE CMPN", "section" : "A", "marks" : 80 }
{ "_id" : ObjectId("5fb3e224fb555c04b565f99c"), "name" : "Shelton", "rollno" : 3, "pid" : 172002, "class" : "BE CMPN", "section" : "A", "marks" : 50 }
{ "_id" : ObjectId("5fb3e236fb555c04b565f99d"), "name" : "Ayush", "rollno" : 4, "pid" : 172003, "class" : "BE CMPN", "section" : "A", "marks" : 70 }
{ "_id" : ObjectId("5fb3e24afb555c04b565f99e"), "name" : "Alisto", "rollno" : 5, "pid" : 172004, "class" : "BE CMPN", "section" : "A", "marks" : 79 }
>
```

$gte→greater than or equal to



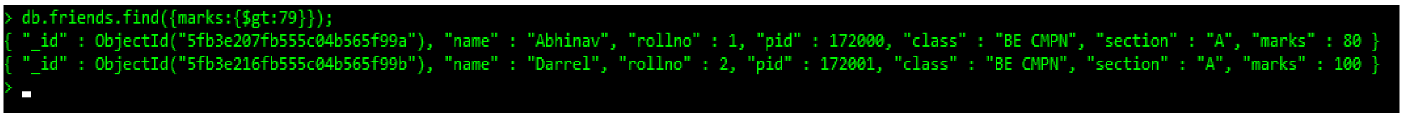
```
> db.friends.find({marks:{$gte:79}});
{ "_id" : ObjectId("5fb3e207fb555c04b565f99a"), "name" : "Abhinav", "rollno" : 1, "pid" : 172000, "class" : "BE CMPN", "section" : "A", "marks" : 80 }
{ "_id" : ObjectId("5fb3e216fb555c04b565f99b"), "name" : "Darrel", "rollno" : 2, "pid" : 172001, "class" : "BE CMPN", "section" : "A", "marks" : 100 }
{ "_id" : ObjectId("5fb3e24afb555c04b565f99e"), "name" : "Alisto", "rollno" : 5, "pid" : 172004, "class" : "BE CMPN", "section" : "A", "marks" : 79 }
>
```

$lte→less than or equal to



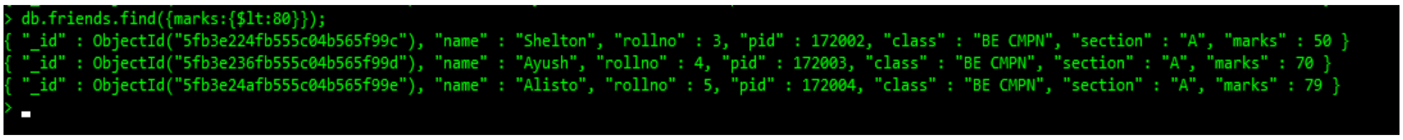
```
> db.friends.find({marks:{$lte:70}});
{ "_id" : ObjectId("5fb3e224fb555c04b565f99c"), "name" : "Shelton", "rollno" : 3, "pid" : 172002, "class" : "BE CMPN", "section" : "A", "marks" : 50 }
{ "_id" : ObjectId("5fb3e236fb555c04b565f99d"), "name" : "Ayush", "rollno" : 4, "pid" : 172003, "class" : "BE CMPN", "section" : "A", "marks" : 70 }
>
```

$gt→ greater than



```
> db.friends.find({marks:{$gt:79}});
{ "_id" : ObjectId("5fb3e207fb555c04b565f99a"), "name" : "Abhinav", "rollno" : 1, "pid" : 172000, "class" : "BE CMPN", "section" : "A", "marks" : 80 }
{ "_id" : ObjectId("5fb3e216fb555c04b565f99b"), "name" : "Darrel", "rollno" : 2, "pid" : 172001, "class" : "BE CMPN", "section" : "A", "marks" : 100 }
>
```

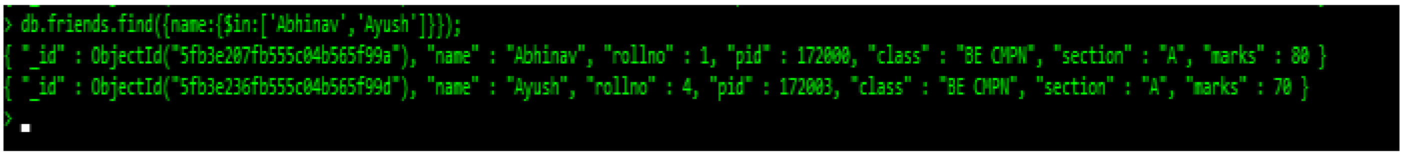
$lt→lesser than)



```
> db.friends.find({marks:{$lt:80}});
{ "_id" : ObjectId("5fb3e224fb555c04b565f99c"), "name" : "Shelton", "rollno" : 3, "pid" : 172002, "class" : "BE CMPN", "section" : "A", "marks" : 50 }
{ "_id" : ObjectId("5fb3e236fb555c04b565f99d"), "name" : "Ayush", "rollno" : 4, "pid" : 172003, "class" : "BE CMPN", "section" : "A", "marks" : 70 }
{ "_id" : ObjectId("5fb3e24afb555c04b565f99e"), "name" : "Alisto", "rollno" : 5, "pid" : 172004, "class" : "BE CMPN", "section" : "A", "marks" : 79 }
>
```

33. Finding records based on the 'IN' operator. Similar to SQL

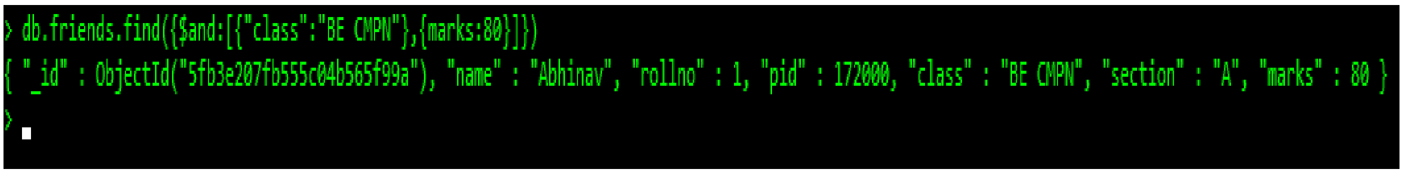
```
> db.student.find({school:{$in:['kv','ssn']}}).pretty();
```



```
> db.friends.find({name:{$in:['Abhinav','Ayush']}});
{ "_id" : ObjectId("5fb3e207fb555c04b565f99a"), "name" : "Abhinav", "rollno" : 1, "pid" : 172000, "class" : "BE CMPN", "section" : "A", "marks" : 80 }
{ "_id" : ObjectId("5fb3e236fb555c04b565f99d"), "name" : "Ayush", "rollno" : 4, "pid" : 172003, "class" : "BE CMPN", "section" : "A", "marks" : 70 }
>
```

34. Finding records based on the AND Clause

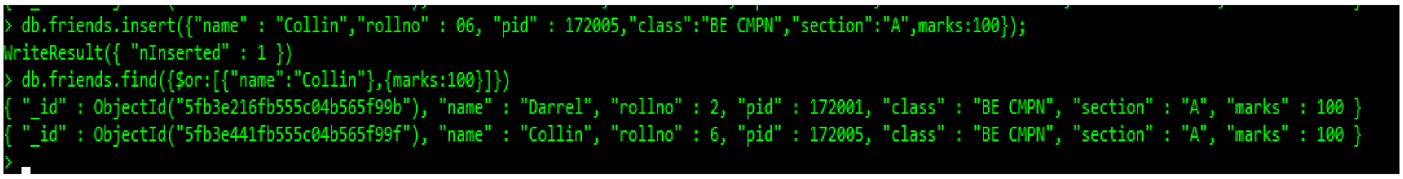
```
> db.student.find({$and:[{school:'kv'},{ grade:'VII'}]}).pretty()
```



```
> db.friends.find({$and:[{"class":"BE CMPN"},{marks:80}]})
{ "_id" : ObjectId("5fb3e207fb555c04b565f99a"), "name" : "Abhinav", "rollno" : 1, "pid" : 172000, "class" : "BE CMPN", "section" : "A", "marks" : 80 }
>
```

35. <u>Finding records based on the OR clause</u>

> db.student.find({$or:[{school:'kv'},{ grade:'VII'}]}).pretty()

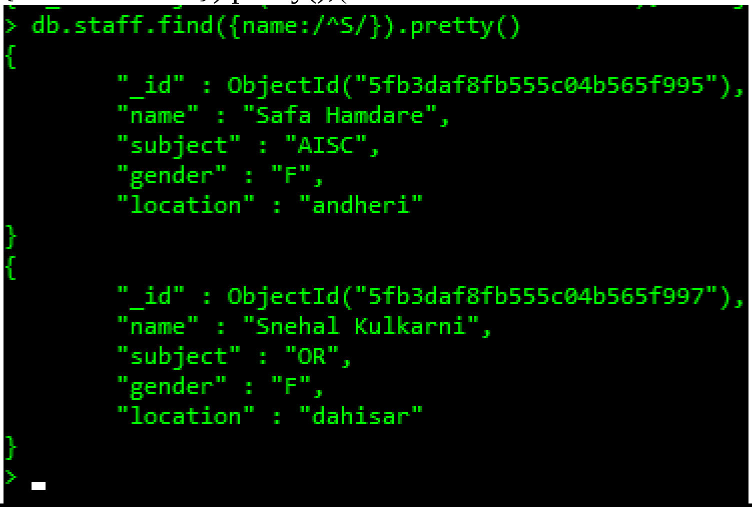


```
> db.friends.insert({"name" : "Collin","rollno" : 06, "pid" : 172005,"class":"BE CMPN","section":"A",marks:100});
WriteResult({ "nInserted" : 1 })
> db.friends.find({$or:[{"name":"Collin"},{marks:100}]})
{ "_id" : ObjectId("5fb3e216fb555c04b565f99b"), "name" : "Darrel", "rollno" : 2, "pid" : 172001, "class" : "BE CMPN", "section" : "A", "marks" : 100 }
{ "_id" : ObjectId("5fb3e441fb555c04b565f99f"), "name" : "Collin", "rollno" : 6, "pid" : 172005, "class" : "BE CMPN", "section" : "A", "marks" : 100 }
>
```

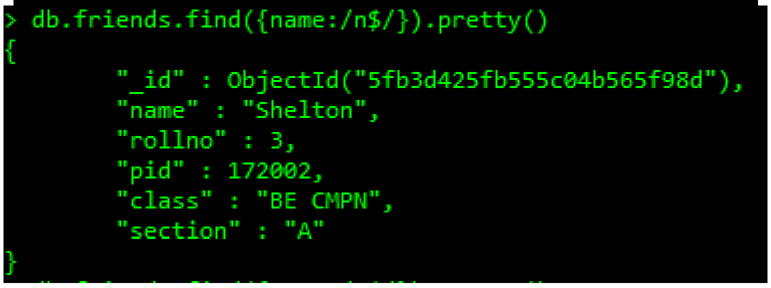
36. <u>Finding records based on matching patterns</u>

> db.student.find({studname:/^a/}).pretty();(All students whose name starts with ‘a’)

> db.student.find({studname:/u$/}).pretty();(All students whose name ends with’u’)



```
> db.staff.find({name:/^S/}).pretty()
{
        "_id" : ObjectId("5fb3daf8fb555c04b565f995"),
        "name" : "Safa Hamdare",
        "subject" : "AISC",
        "gender" : "F",
        "location" : "andheri"
}
{
        "_id" : ObjectId("5fb3daf8fb555c04b565f997"),
        "name" : "Snehal Kulkarni",
        "subject" : "OR",
        "gender" : "F",
        "location" : "dahisar"
}
>
```



```
> db.friends.find({name:/n$/}).pretty()
{
        "_id" : ObjectId("5fb3d425fb555c04b565f98d"),
        "name" : "Shelton",
        "rollno" : 3,
        "pid" : 172002,
        "class" : "BE CMPN",
        "section" : "A"
}
```