4. Show All Databases

Use below command to get list of all databases.

show dbs

```
        db.version() current version of the
> show dbs
charts        0.006GB
chartts1      0.001GB
comp          0.005GB
drilldown     0.006GB
export        0.005GB
export1       0.005GB
incomp        0.005GB
local         0.000GB
page          0.007GB
relationship  0.005GB
>
```

5. Create new database

To create a new database execute the following command.

use DATABASE_NAME

```
mongo.exe
> use myTestDB
switched to db myTestDB
>
```

6. Know your current selected database

To know your current working/selected database execute the following command

db

```
> db
myTestDB
>
```

7. Drop database

To drop the database execute following command, this will drop the selected database

db.dropDatabase()

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

8. Create collection

To create the new collection execute the following commands

db.createCollection(name)

```
> db.createCollection("Employee");
{ "ok" : 1 }
>
```

9. To check collections list

To get the list of collections created execute the following command
Show collections

```
> show collections
Employee
>
```

10. Drop collection

To drop the selected collection execute the following command
db.COLLECTION_NAME.drop()

```
> show collections
Department
Employee
> db.Department.drop()
true
> show collections
Employee
>
```

11. Insert document in collection

>db.COLLECTION_NAME.insert(document)

To insert single document in selected collection execute the following command

```
> db.Employee.insert({name: 'Emp1',address: 'Pune'})
WriteResult({ "nInserted" : 1 })
> db.Employee.insert({name: 'Emp2',address: 'Mumbai'})
WriteResult({ "nInserted" : 1 })
>
```

11. Insert document in collection

>db.COLLECTION_NAME.insert(document)

To insert single document in selected collection execute the following command

```
> db.Employee.insert({name: 'Emp1',address: 'Pune'})
WriteResult({ "nInserted" : 1 })
> db.Employee.insert({name: 'Emp2',address: 'Mumbai'})
WriteResult({ "nInserted" : 1 })
>
```

To insert multiple documents in selected collection execute following command

```
> db.Employee.insertMany([{name : 'Emp1', address:'Pune'},{name: 'Emp2',address: 'Mumbai'}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5b920da763cdcea45e0ac334"),
                ObjectId("5b920da763cdcea45e0ac335")
        ]
}
>
```

12. Get collection document

To get the list documents in collection execute the following command

>db.COLLECTION_NAME.find()

```
> db.Employee.find().pretty()
{
        "_id" : ObjectId("5b920c9863cdcea45e0ac332"),
        "name" : "Emp1",
        "address" : "Pune"
}
{
        "_id" : ObjectId("5b920c9d63cdcea45e0ac333"),
        "name" : "Emp2",
        "address" : "Mumbai"
}
>
```

```
> db.student.updateOne({name: "Annu"}, {$set:{age:25}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.student.find().pretty()
{
        "_id" : ObjectId("600e9afd0cf217478ba93566"),
        "name" : "Annu",
        "age" : 25
}
{
        "_id" : ObjectId("600e9afd0cf217478ba93567"),
        "name" : "Bhannu",
        "age" : 24
}
{
        "_id" : ObjectId("600e9afd0cf217478ba93568"),
        "name" : "Bhannu",
        "age" : 24
}
>
```

```
> use gfg
switched to db gfg
> db.student.find().pretty()
{
        "_id" : ObjectId("600ebc010cf217478ba93570"),
        "name" : "aaksh",
        "age" : 15
}
{
        "_id" : ObjectId("600ebc010cf217478ba93571"),
        "name" : "nikhil",
        "age" : 18
}
{
        "_id" : ObjectId("600ebc010cf217478ba93572"),
        "name" : "vishal",
        "age" : 18
}
>
```

Update single document

```
db.student.updateMany({name: "aaksh"}, {$set
```

Here, we update the age of a student whose name is aaksh from 15 to 20 using updateMany() method.

```
> db.student.updateMany({name: "aaksh"}, {$set:{age: 20}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.student.find().pretty()
{
        "_id" : ObjectId("600ebc010cf217478ba93570"),
        "name" : "aaksh",
        "age" : 20
}
{
        "_id" : ObjectId("600ebc010cf217478ba93571"),
        "name" : "nikhil",
        "age" : 18
}
{
        "_id" : ObjectId("600ebc010cf217478ba93572"),
        "name" : "vishal",
        "age" : 18
}
>
```

```
> use gfg
switched to db gfg
> db.student.find().pretty()
{
        "_id" : ObjectId("600e9afd0cf217478ba93566"),
        "name" : "Annu",
        "age" : 20
}
{
        "_id" : ObjectId("600e9afd0cf217478ba93567"),
        "name" : "Bhannu",
        "age" : 24
}
{
        "_id" : ObjectId("600e9afd0cf217478ba93568"),
        "name" : "Bhannu",
        "age" : 24
}
```

Example 1: Update the age of the student whose name is Annu

```
db.student.updateOne({name: "Annu"}, {$set:{a
```

Here, the first parameter is the document whose value to be changed {name:"Annu"} and the second parameter is set keyword means to set(update) the following first matched key value with the older key value, i.e., from 20 to 24.

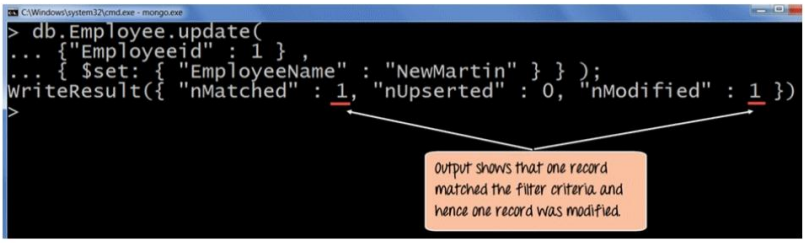
```
> db.student.updateOne({name: "Annu"}, {$set:{age:25}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.student.find().pretty()
{
        "_id" : ObjectId("600e9afd0cf217478ba93566"),
        "name" : "Annu",
        "age" : 25
}
{
        "_id" : ObjectId("600e9afd0cf217478ba93567"),
        "name" : "Bhannu",
        "age" : 24
}
{
        "_id" : ObjectId("600e9afd0cf217478ba93568"),
        "name" : "Bhannu",
        "age" : 24
}
```

8:52 43%
MongoDB Update() Do...
guru99.com

```
db.Employee.update(
{"Employeeid" : 1},
{$set: { "EmployeeName" :
"NewMartin"}});
```

If the command is executed successfully, the following Output will be shown

Output:



```
db.student.find()
```

```
> db.student.find().pretty()
{
        "_id" : ObjectId("60227eaff19652db63812e8d"),
        "name" : "Akshay",
        "age" : 18
}
{
        "_id" : ObjectId("60227eaff19652db63812e8e"),
        "name" : "Bablue",
        "age" : 17,
        "score" : {
                "math" : 230,
                "science" : 234
        }
}
{
        "_id" : ObjectId("60227eaff19652db63812e8f"),
        "name" : "Chandhan",
        "age" : 18
}
>
```

- **Find all the documents present in the collection by passing empty document:**

```
db.student.find({})
```

```
> db.student.find({})
{ "_id" : ObjectId("60227eaff19652db63812e8d"), "name" : "Akshay", "age" : 18 }
{ "_id" : ObjectId("60227eaff19652db63812e8e"), "name" : "Bablue", "age" : 17, "
score" : { "math" : 230, "science" : 234 } }
{ "_id" : ObjectId("60227eaff19652db63812e8f"), "name" : "Chandhan", "age" : 18
}
>
```

- **With empty query specification it returns the first document in the collection:**

```
db.student.findOne()
```

```
> db.student.findOne()
{
        "_id" : ObjectId("6013f10b9e34d5bfb0d50dae"),
        "name" : "Nikhil",
        "language" : "c++"
}
>
```

Limit two documents

```
db.gfg.find().limit(2)
```

Here, we only want the first two documents in the result. So, we pass 2 in the limit method.

```
> use geeksforgeeks
switched to db geeksforgeeks
> db.gfg.find().limit(2)
{ "_id" : ObjectId("6005d3158438681f01c53e7f"), "content" : "Data Structure" }
{ "_id" : ObjectId("6005d3258438681f01c53e80"), "content" : "Algorithims" }
>
```

Limit only two documents that match the given condition

```
> use geeksforgeeks
switched to db geeksforgeeks
> db.gfg.find()
{ "_id" : ObjectId("6005d3158438681f01c53e7f"), "content" : "Data Structure" }
{ "_id" : ObjectId("6005d3258438681f01c53e80"), "content" : "Algorithims" }
{ "_id" : ObjectId("6005d3318438681f01c53e81"), "content" : "Interview Preparation" }
{ "_id" : ObjectId("6005d33d8438681f01c53e82"), "content" : "FANG" }
{ "_id" : ObjectId("6009c642df8008388bd7646a"), "content" : "Competitve Programming"
{ "_id" : ObjectId("6009c66bdf8008388bd7646b"), "content" : "Development" }
{ "_id" : ObjectId("6009c8e4df8008388bd7646c"), "content" : "coding questions" }
{ "_id" : ObjectId("6009c907df8008388bd7646d"), "content" : "compiler online" }
>
```

- **Skip the first document**

```
db.gfg.find().skip(1)
```

Here, we skip the first document by passing 1 in the skip method.

```
> use geeksforgeeks
switched to db geeksforgeeks
> db.gfg.find().skip(1)
{ "_id" : ObjectId("6005d3258438681f01c53e80"), "content" : "Algorithims" }
{ "_id" : ObjectId("6005d3318438681f01c53e81"), "content" : "Interview Preparation" }
{ "_id" : ObjectId("6005d33d8438681f01c53e82"), "content" : "FANG" }
{ "_id" : ObjectId("6009c642df8008388bd7646a"), "content" : "Competitve Programming"
{ "_id" : ObjectId("6009c66bdf8008388bd7646b"), "content" : "Development" }
{ "_id" : ObjectId("6009c8e4df8008388bd7646c"), "content" : "coding questions" }
{ "_id" : ObjectId("6009c907df8008388bd7646d"), "content" : "compiler online" }
>
```

- **Skip the two documents**

```
db.gfg.find().skip(2)
```

```
> db.student.find().pretty()
{
        "_id" : ObjectId("600f1abb923681e7681ebdce"),
        "name" : "Akshay",
        "age" : 19
}
{
        "_id" : ObjectId("600f1abb923681e7681ebdcf"),
        "name" : "Bablu",
        "age" : 18
}
{
        "_id" : ObjectId("600f1abb923681e7681ebdd0"),
        "name" : "Rakesh",
        "age" : 21
}
{
        "_id" : ObjectId("600f1abb923681e7681ebdd1"),
        "name" : "Gourav",
        "age" : 20
}
>
```

- **Return all the documents in ascending order of the age:**

```
db.student.find().sort({age:1})
```

```
> db.student.find().sort({age:1})
{ "_id" : ObjectId("6015ba124dabc381f81e53ae"), "name" : "Bablue", "age" : 18 }
{ "_id" : ObjectId("6015ba124dabc381f81e53ad"), "name" : "Akshay", "age" : 19 }
{ "_id" : ObjectId("6015ba124dabc381f81e53b0"), "name" : "Gourav", "age" : 20 }
{ "_id" : ObjectId("6015ba124dabc381f81e53af"), "name" : "Rakesh", "age" : 21 }
>
```

```
>
> db.student.find().pretty()
{
        "_id" : ObjectId("6011c71f781ba1a1c1ffc5b1"),
        "name" : "Nikhil",
        "language" : "c++"
}
{
        "_id" : ObjectId("6011c71f781ba1a1c1ffc5b2"),
        "name" : "Avinash",
        "language" : "python"
}
{
        "_id" : ObjectId("6011c71f781ba1a1c1ffc5b3"),
        "name" : "Vishal",
        "language" : "python"
}
>
```

- **Create an index without option:**

```
db.student.createIndex({name:1})
```

Here, we create an ascending index on the single field (i.e., name) without options.

```
> db.student.createIndex({name:1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

```
> db.student.find().pretty()
{
        "_id" : ObjectId("601c22b805cdfa6d2ab1df17"),
        "name" : "Nikhil",
        "language" : "c++"
}
{
        "_id" : ObjectId("601c22b805cdfa6d2ab1df18"),
        "name" : "Avinash",
        "language" : "python"
}
{
        "_id" : ObjectId("601c22b805cdfa6d2ab1df19"),
        "name" : "Vishal",
        "language" : "python"
}
```

First of all we created an index on the name field using createIndex() method:

```
db.student.createIndex({name:2})
```

```
> db.student.createIndex({name:2})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "ok" : 1
}
>
```

```
>
> db.student.find().pretty()
{
        "_id" : ObjectId("6011c71f781ba1a1c1ffc5b1"),
        "name" : "Nikhil",
        "language" : "c++"
}
{
        "_id" : ObjectId("6011c71f781ba1a1c1ffc5b2"),
        "name" : "Avinash",
        "language" : "python"
}
{
        "_id" : ObjectId("6011c71f781ba1a1c1ffc5b3"),
        "name" : "Vishal",
        "language" : "python"
}
>
```

- **Return an array of documents that hold index information for the student collection:**

```
db.student.getIndexes()
```

```
> db.student.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
>
```

- **Displaying the total number of students in one section only**

```
db.students.aggregate([{$match:{sec:"B"}},{$c
```

In this example, for taking a count of the number of students in section B we first filter the documents using the **$match operator,** and then we use the **$count** accumulator to count the total number of documents that are passed after filtering from the $match.

```
> db.students.aggregate([{$match:{sec:"B"}},{$count:"Total student in sec:B"}])
{ "Total student in sec:B" : 3 }
>
```

anki — mongo — 78×49

```
> use GeeksforGeeks
switched to db GeeksforGeeks
> db.contributor.find()
{ "_id" : ObjectId("5e6f7a6692e6dfa3fc48ddbe"), "name" : "Rohit", "branch" : "
CSE", "joiningYear" : 2018, "language" : [ "C#", "Python", "Java" ], "personal
" : { "contactinfo" : 0, "state" : "Delhi", "age" : 24, "semesterMarks" : [ 70
, 73.3, 76.5, 78.6 ] }, "salary" : 1000 }
{ "_id" : ObjectId("5e7b9f0a92e6dfa3fc48ddbf"), "name" : "Amit", "branch" : "E
CE", "joiningYear" : 2017, "language" : [ "Python", "C#" ], "personal" : { "co
ntactinfo" : 234556789, "state" : "UP", "age" : 25, "semesterMarks" : [ 80, 80
.1, 98, 70 ] }, "salary" : 10000 }
{ "_id" : ObjectId("5e7b9f0a92e6dfa3fc48ddc0"), "name" : "Sumit", "branch" : "
CSE", "joiningYear" : 2017, "language" : [ "Java", "Perl" ], "personal" : { "c
ontactinfo" : 2300056789, "state" : "MP", "age" : 24, "semesterMarks" : [ 89,
80.1, 78, 71 ] } }
>
```

anki — mongo — 78×27

```
> db.contributor.find({$and: [{branch: "CSE"}, {joiningYear: 2018}]}).pretty()

{
        "_id" : ObjectId("5e6f7a6692e6dfa3fc48ddbe"),
        "name" : "Rohit",
        "branch" : "CSE",
        "joiningYear" : 2018,
        "language" : [
                "C#",
                "Python",
                "Java"
        ],
        "personal" : {
                "contactinfo" : 0,
                "state" : "Delhi",
                "age" : 24,
                "semesterMarks" : [
                        70,
                        73.3,
                        76.5,
                        78.6
                ]
        },
        "salary" : 1000
}
```

Using $and operator with multiple expressions specifying the same field :

anki — mongo — 78×46

```
> db.contributor.find({$or: [{branch: "ECE"}, {joiningYear: 2017}]}).pretty()
{
        "_id" : ObjectId("5e7b9f0a92e6dfa3fc48ddbf"),
        "name" : "Amit",
        "branch" : "ECE",
        "joiningYear" : 2017,
        "language" : [
                "Python",
                "C#"
        ],
        "personal" : {
                "contactinfo" : 234556789,
                "state" : "UP",
                "age" : 25,
                "semesterMarks" : [
                        80,
                        80.1,
                        98,
                        70
                ]
        },
        "salary" : 10000
}
{
        "_id" : ObjectId("5e7b9f0a92e6dfa3fc48ddc0"),
        "name" : "Sumit",
        "branch" : "CSE",
        "joiningYear" : 2017,
        "language" : [
                "Java",
                "Perl"
        ],
        "personal" : {
                "contactinfo" : 2300056789,
                "state" : "MP",
                "age" : 24,
                "semesterMarks" : [
                        89,
                        80.1,
                        78,
                        71
                ]
        }
}
```

Matching values in nested/embedded documen using $or operator:

anki — mongo — 78×25

```
> db.contributor.find({$nor: [{"personal.age": 24},{"personal.state": "AP"}]})
.pretty()
{
        "_id" : ObjectId("5e7b9f0a92e6dfa3fc48ddbf"),
        "name" : "Amit",
        "branch" : "ECE",
        "joiningYear" : 2017,
        "language" : [
                "Python",
                "C#"
        ],
        "personal" : {
                "contactinfo" : 234556789,
                "state" : "UP",
                "age" : 25,
                "semesterMarks" : [
                        80,
                        80.1,
                        98,
                        70
                ]
        },
        "salary" : 10000
}
```

Matching values in an array using $nor operator:

```
db.contributor.find({salary: {$not:
```

anki — mongo — 78×47

```
> db.contributor.find({salary: {$not: {$gt: 2000}}}).pretty()
{
        "_id" : ObjectId("5e6f7a6692e6dfa3fc48ddbe"),
        "name" : "Rohit",
        "branch" : "CSE",
        "joiningYear" : 2018,
        "language" : [
                "C#",
                "Python",
                "Java"
        ],
        "personal" : {
                "contactinfo" : 0,
                "state" : "Delhi",
                "age" : 24,
                "semesterMarks" : [
                        70,
                        73.3,
                        76.5,
                        78.6
                ]
        },
        "salary" : 1000
}
{
        "_id" : ObjectId("5e7b9f0a92e6dfa3fc48ddc0"),
        "name" : "Sumit",
        "branch" : "CSE",
        "joiningYear" : 2017,
        "language" : [
                "Java",
                "Perl"
        ],
        "personal" : {
                "contactinfo" : 2300056789,
                "state" : "MP",
                "age" : 24,
                "semesterMarks" : [
                        89,
                        80.1,
                        78,
                        71
                ]
        }
} _
```

Matching values in nested/embedded documents using $not operator: