```
use school

db.createCollection("facutly")

db.faculty.insert({"name":"mital","age":26,"city":"rajkot"})
db.faculty.find()
db.faculty.insertMany([{"name":"aa","age":34,"city":"jamnagar"},
{"name":"bb","age":34,"city":"rajkot"}])
db.faculty.drop()
show collections
db.dropDatabase()
show dbs

db.help()
db.stats()
db.shutdownServer()

//rename collections
db.collection.renameCollection(target)db.student.renameCollection(“stud”)

//copy collections
db.collection.copyTo(target)
db.stud.copyTo(“student”)
db.student.storageSize()
show dbs
db.emp.insert({"name":"mital"})
show dbs
db.dropDatabase()
show dbs
use db
use test
db.createCollection("name")
db.test.insert({"name":"hello mital"})
db.test.drop()
show collections
use Test
db.createCollection("name")

use geet
db.createCollection('emp')
db.emp.insert({'empno':001,'empnm':'mital','salary':5000})
db.emp.insert({'empno':002, 'empnm':'sheetal','salary':6000})
db.emp.insert({'empno':003, 'empnm':'neha','salary':7000})
db.emp.find()

//greter than 6000
db.emp.find({'salary':{$gt:6000}})

//less than 6000
db.emp.find({'salary':{$lt:6000}})


// Greater than equals 6000
db.emp.find({'salary':{$gte:6000}}).pretty()

//Not equals 6000
db.emp.find({“salary”:{$ne:6000}}).pretty()

//salary less than 7000 and greater than 5000
db.emp.find({“sal”:{$lt:7000},”sal”:{$gt:5000}}).pretty()

//empnm=mital or salary =6000
db.emp.find({$or:[{'empnm':'neha'},{'salary':'6000'}]}).pretty()
```

```
//or operator
db.emp.find({$or:[{'empnm':'neha'},{'salary':'6000'}]}).pretty()

//and operator
db.emp.find({$and:[{"empnm":"mital"},{"salary":5000}]}).pretty()

// display emp whose salary is greater than 5000 and( name is mital or sheetal
db.emp.find({
      "salary": {$gt:5000},
       $and: [{"empnm": "mital"},{"empnm": "neha"}]
         }).pretty()

//and operator
db.emp.find({
      "salary": {$gt:5000},
       $and: [{"empnm": "mital"},{"salary": 5000}]
         }).pretty()

//not operator
db.stud.find({salary: {$not: {$gt: 2000}}}).pretty()



db.emp.find({'empnm':'mital'}).count()

db.emp.save({'_id':ObjectId('62edd84595da2119154effcc'),'empnm':'nehasutariya'})

db.emp.find()

//ARRAY UPDATE OPERATOR
use stud
db.createCollection("s1")
db.s1.insert(
{"no":1,"nm":"xyz",
"marks":[40,50,60]
})
db.stud.update({no:1},{$pop:{marks:1}}) //remove last element (last:1/first:-1)

db.s1.update({no:1},{$addToSet: {marks:55}}) //set marks with 55

db.s1.update({no:1},{$pull:{marks: {$in:[50,60]}}}) //remove 40,55 marks

db.s1.update({no:1},{$pullAll:{marks:[40,55]}}) //remove multiple

db.s1.update({no:1},{$push:{marks:60}})

db.s1.update({no:1},{$push:{mark: {$each: [55],$position:2}}})

use geet
db.createCollection('emp')
db.emp.insert({'empno':001,'empnm':'mital','salary':5000})
db.emp.insert({'empno':002, 'empnm':'sheetal','salary':6000})
db.emp.insert({'empno':003, 'empnm':'neha','salary':7000})
db.emp.find()


use stud
use geet
db.createCollection('emp')
db.emp.insert({'empno':001,'empnm':'mital','salary':5000})
db.emp.insert({'empno':002, 'empnm':'sheetal','salary':6000})
db.emp.insert({'empno':003, 'empnm':'neha','salary':7000})
db.emp.update({"empno":001},{$set:{"empnm":"MITAL"}})
```

```
 //db.emp.update({"empno":1},{$set:{"empnm":"MITAL"}}) //onecolumn

 //db.emp.update({"empno":1},{$set:{"empnm":"MITAL"}},{multi:true}) //whole
column
//db.emp.update({"_id" : ObjectId("66cea8a27694aa4e6e9ab9c6")},{"empno":303})

//whole document update
db.faculty.update({"_id" : ObjectId("5f2b9ebf50bfd294a6c6781e")},
{"firstname":"mitalgoswami","age":31})

//particular column update
db.faculty.update({"firstname":"mitalgoswami"},{$set:{"age":55}})
db.faculty.find()


//whole record update
db.emp.update({'_id':ObjectId('62edd84595da2119154effcc')},
{'empno':0001,'empnm':'neha','salary':5000})

//particular update
db.emp.update({'_id':ObjectId('62edd84595da2119154effcc')},{$set:
{'empno':0001}})


db.emp.update({},{$unset:{'empno':""}}) //onecolumn

 db.emp.update({},{$unset:{empno:””}},{multi:”true”}) //whole

db.emp.update({"empnm":"MITAL"},{$inc:{"salary":1000}}) //increment

db.emp.update({},{$rename:{"empnm":"ename"}},{multi:true}) //editcolumn

db.stud.find ( { $nor: [ { "empnm": { $eq:"mital" } }, { "empnm": {$eq:
"sheetal"} } ] } )
//not display mital and sheetal records

db.faculty.remove({})
db.faculty.remove({"firstname" :"mital"})
db.faculty.deleteOne({"empnm" :"mital"})
db.faculty.deleteMany({"firstname" :"mital"})
db.faculty.insertMany([{"firstname":"mital","age":25},
{"firstname":"sheetal","age":26}])
db.faculty.find().count()




 DELETE ALL DOCUMENT FROM STUD COLLECTION
> use college
> db.stud.remove({})
2. DELETE DOCUMENT FROM STUD COLLECTION WHO ARE IN MCA.
> use college
> db.stud.remove({streem:"MCA"})
3. DELETE ONE DOCUMENT WHO IS IN MSCIT.
> use college
> db.stud.remove({streem:"MSCIT"},{justOne : 1})
4. DELETE ALL DOCUMENT WHERE RLNO > 125
> use college
> db.stud.remove({ rlno: { $gt: 125 }})
//save new record
db.faculty.save([{"firstname":"mital","age":25},
{"firstname":"sheetal","age":26}])
db.faculty.find().count()
```

```
//save particular id record not insert new record.
db.faculty.save({ "_id" :
ObjectId("5f45dc8bac01ff2ad8642716"),"firstname":"mital","age":25})

use school
db.createCollection("faculty")
db.faculty.insertMany([{"firstname":"mital","age":25},
{"firstname":"sheetal","age":26}])
db.faculty.find().pretty()
db.faculty.remove({"firstname":"sheetal"})
db.faculty.find({"firstname":"mital"})

//ELEMENT QUERY OPERATOR:
db.emp.find({"empno" :{$exists : true, $in:[2,5]}}) //2 and 5 record display

db.emp.find({ "empno": { $nin: [ 20, 15 ]}}) //not match

db.emp.find({"empno" :{$type :“string”}})//ARRAY QUERY OPERATOR:


db.s1.insertMany(
   [
      { "_id" : 1, address : "2030 Martian Way", zipCode : "90698345" },
      { "_id" : 2, address: "156 Lunar Place", zipCode : 43339374 },
      { "_id" : 3, address : "2324 Pluto Place", zipCode: NumberLong(3921412) },
      { "_id" : 4, address : "55 Saturn Ring" , zipCode : NumberInt(88602117) },
      { "_id" : 5, address : "104 Venus Drive", zipCode : ["834847278",
"1893289032"]}
   ]
)
db.s1.find( { "zipCode" : { $type : 2 } } )
db.s1.find( { "zipCode" : { $type : "string" } } )

db.s1.find( { "zipCode" : { $type : 1 } } )
db.s1.find( { "zipCode" : { $type : "number" } } )

db.s1.find( { "zipCode" : { $type : 1 } } )
db.s1.find( { "zipCode" : { $type : "double" } } )

db.s1.find( { "zipcode" : { $type : [ 2 , 1 ] } } );
db.s1.find( { "zipcode" : { $type : [ "string" , "double" ] } } )


use skills
db.createCollection("sdata")
db.sdata.insert(
 [{ "Name" : "Balaji",
      "skills" : [ "Dancing", "Cooking", "Singing" ]},
 { "Name" : "Ramesh",
      "skills" : [ "Cooking", "Singing" ] },
 { "Name" : "Suresh",
      "skills" : [ "Dancing", "Singing" ] }])

db.sdata.find({skills:{$all:["Cooking","Dancing"]}})
db.sdata.find({skills:{$size:2}})




db.faculty.find({"firstname":"mital"}).limit(2)

db.faculty.find({"firstname":"mital"}).skip(2)
db.faculty.find({"firstname":"mital"}).skip(2).limit(2)
```

```
// 1 than asscending order sorting
db.faculty.find({"firstname":"mital"}).sort({"firstname":1})
// -1 than descending order sorting
db.faculty.find({"firstname":"mital"}).sort({"firstname":-1})
db.emp.find().sort({"empnm": -1})

//query Example
use geet1
db.createCollection('emp')
db.emp.insert({'empno':001,'empnm':'mital','salary':5000})
db.emp.insert({'empno':002, 'empnm':'sheetal','salary':6000})
db.emp.insert({'empno':003, 'empnm':'neha','salary':7000})
show dbs

db.emp.find({'salary':6000})
//greaterthen
db.emp.find({'salary':{$gt:5000}})
//lessthen
db.emp.find({'salary':{$lt:7000}})

//display one field empnm with no id
db.emp.find({},{empnm:1,_id:0})

//display one field empnm with id
db.emp.find({},{empnm:1})

//display two field empnm,salary
db.emp.find({},{'empnm':1,'salary':1,_id:0})

//only salary display with ascending
db.emp.find({},{"salary":1,_id:0}).sort({"salary":1})

//descending
db.emp.find({},{salary:1,_id:0}).sort({salary:-1})

//and operator
db.emp.find({'empnm':'mital','salary':5000})

//or operator
db.emp.find(
{
    $or:[{'empnm':'mital'},{'salary':6000}]
}
)

db.emp.find(
{
    'empnm':'mital',$or:[{'salary':5000},{'salary':6000}]
}
)


//remove duplicate value
//display duplicate value
db.emp.find({'empnm':'mital'},{'empnm':1,_id:0})

//not display duplicate value
db.emp.distinct('empnm',{'empnm':'mital'})

db.emp.distinct('empnm',{'salary':{$gte:5000}})

//count duplicate record
db.emp.find({'empnm':'mital'}).count()
```

```
//create index

db.emp.createIndex({empno:1})db.emp.ensureIndex({empno:1})
db.emp.getIndexes()
db.emp.dropIndex({empno:1})
```

(1) $ project - It is used to select some specific field from a collection.

(2) $ match - This is filtering operation for reduce the amount of document which
will be given as input to the next stage.

(3) $group - It is used for simple count operations where the actual aggregation done.

(4) $sort – It is used to sort the documents.

(5) $skip – It is used to skip the documents from given amount of documents.

(6) $ limit – It is used to limit the amount of documents by the given number starting from current position.

(7) $ unwind – It is used to unwind the document data using array. This operation will be undone the joint data and keep the documents individual. It will increase the amount of documents for the next stage.

```
use geet1
db.createCollection('emp')
db.emp.insert({'empno':001,'empnm':'mital','salary':5000})
db.emp.insert({'empno':002, 'empnm':'sheetal','salary':6000})
db.emp.insert({'empno':003, 'empnm':'neha','salary':7000})

//sum of all salary

db.emp.aggregate([{$group : {_id : null, salary_sum : {$sum : '$salary'}}}])

//select empnm, sum(salary) salary_sum from emp

where salary > 5000
group by firstName


 db.emp.aggregate([
  { $match: { salary : { $gt: 5000} } },
  {$group : {_id : "$empnm", salary_sum : {$sum : "$salary"}}}])

//count record whose salary is >5000 using aggregate function.

db.emp.aggregate([
  {$match:{salary:{$gt:5000}}},
  {$group:{_id:null,count:{$sum:1}}}
   ])




use geet
db.createCollection('student');
var students = [
{
      rollNo:1,
      name:"Ganesh",
      city:"Rajkot",
      state:"Gujarat",
```

```
      mobileNo:"9999999999",
      result:"PASS",
      percentage:86
},
{
      rollNo:2,
      name:"Shankar",
      city:"Ahmedabad",
      state:"Gujarat",
      mobileNo:"9999999998",
      result:"PASS",
      percentage:80
},
{
      rollNo:3,
      name:"Mahadev",
      city:"Mumbai",
      state:"Maharashtra",
      mobileNo:"88888888888",
      result:"PASS",
      percentage:89
},
{
      rollNo:4,
      name:"Rajesh",
      city:"Nashik",
      state:"Maharashtra",
      mobileNo:"9999957699",
      result:"PASS",
      percentage:56
},
{
      rollNo:5,
      name:"Samay",
      city:"Jaipur",
      state:"Rajasthan",
      mobileNo:"8888999999",
      result:"FAIL",
      percentage:33
},
];
db.student.insert(students)
db.student.find()
db.student.find({city:"Rajkot"}).pretty();

db.student.aggregate({ $match:{city:"Rajkot"}}).pretty();

db.student.find({percentage:{$lt:60}}).pretty();

db.student.aggregate({$match:{percentage:{$lt:60}}}).pretty();

db.student.find({city:{$ne:"Rajkot"}}).pretty();

db.student.aggregate({$match:{city:{$ne:"Rajkot"}}}).pretty();

db.student.count({percentage:{$lt:60}})

db.student.find({percentage:{$lt:60}}).count();

db.student.count({city:"Rajkot"});
db.student.find({city:"Rajkot"}).count();
//state vise max percentage
db.student.aggregate(
      {
```

```
            $group:{_id:"$state", minPercentage:{$min:"$percentage"}}
      }
);


//maximum persentage
db.student.aggregate({$group:{_id:"rollNo","percentage":{$max:"$percentage"}}})


db.student.aggregate({$match:{city:"Rajkot"}}).pretty();

db.student.aggregate({$match:{percentage:{$lt:60}}}).pretty();
db.student.aggregate({$match:{city:{$ne:"Rajkot"}}}).pretty();
db.student.count({city:"Rajkot"});
db.student.find({city:"Rajkot"}).count();

db.student.aggregate({$sort:{percentage:1}})

db.student.aggregate({$sort:{percentage:-1}})

db.student.aggregate({$sort:{percentage:-1}},{$skip:2})

db.student.aggregate({$sort:{percentage:-1}},{$skip:2},{$limit:2})


use geet1
db.createCollection('emp')
db.emp.insert({'empno':001,'empnm':'mital','salary':5000})
db.emp.insert({'empno':002, 'empnm':'sheetal','salary':6000})
db.emp.insert({'empno':003, 'empnm':'neha','salary':7000})

//select record
db.emp.aggregate([{$project: { _id:0, ename:1, deptno: 1, sal:1}}])

use product
db.orders.insertMany( [
   { _id: 1, cust_id: "abc1", ord_date: ISODate("2012-11-02T17:04:11.102Z"),
status: "A", amount: 50 },

   { _id: 2, cust_id: "xyz1", ord_date: ISODate("2013-10-01T17:04:11.102Z"),
status: "A", amount: 100 },

   { _id: 3, cust_id: "xyz1", ord_date: ISODate("2013-10-12T17:04:11.102Z"),
status: "D", amount: 25 },

   { _id: 4, cust_id: "xyz1", ord_date: ISODate("2013-10-11T17:04:11.102Z"),
status: "D", amount: 125 },

   { _id: 5, cust_id: "abc1", ord_date: ISODate("2013-11-12T17:04:11.102Z"),
status: "A", amount: 25 }
] )


Group by and Calculate a Sum
The following aggregation operation selects documents with status equal to "A",
groups the matching documents by the cust_id field and calculates the total for
each cust_id field from the sum of the amount field, and sorts the results by
the total field in descending order:

db.orders.aggregate( [
   { $match: { status: "A" } },
   { $group: { _id: "$cust_id", total: { $sum: "$amount" } } }] )
```

```
db.orders.explain().aggregate( [
   { $match: { status: "A" } },
   { $group: { _id: "$cust_id", total: { $sum: "$amount" } } },
   { $sort: { total: -1 } }
] )

db.users.aggregate([
    { $match: { } }
])

db.orders.aggregate([
  { $match : { cust_id : "abc1", cust_id : "xyz1" } }
]).pretty()

    db.orders.aggregate([
  { $match: { amount: { $gt: 100 } } }
])

  db.orders.aggregate([
  { $project : { _id : 0, cust_id : 1, ord_date : 1, amount : 1 } }
]).pretty()

db.orders.aggregate([
    { $match: { } }
])
    db.orders.aggregate([
  { $match: { amount: { $gt: 100 } } }
])

db.orders.aggregate([
  { $sort: { amount: 1 } }
])

db.orders.aggregate([
  { $limit: 2 }
])

db.orders.aggregate([
  { $group: { _id: "$status", averageamount: { $avg: "$amount" } } }
])

db.orders.aggregate([
    { $group: { "_id": "status" } }
])

db.orders.aggregate([
    {
        $match: {
            "status": { $in: ["A", "B"] }
        }
    }
])

db.orders.aggregate(
      {
            $group:{_id:"$status", minamount:{$min:"$amount"}}
      }
);

perator     Meaning
$count      Calculates the quantity of documents in the given group.
$max  Displays the maximum value of a document's field in the collection.
$min  Displays the minimum value of a document's field in the collection.
```

```
$avg  Displays the average value of a document's field in the collection.
$sum  Sums up the specified values of all documents in the collection.
$push Adds extra values into the array of the resulting document.
```

```
//map reduce example
use books

db.books.insert( {
          "title" : "MongoDB: The Definitive Guide",
          "published" : "2013-05-23",
          "authors": [
              { "firstName" : "BRIJESH",  "lastName" : "PATEL" }
          ],
          "categories" : [ "Databases", "NoSQL", "Programming" ],
          "publisher" : { "name" : "NACHIKETA" },
          "price" : 32.99
      } )
db.books.insert( {
    "title" : "MongoDB in Action",
    "published" : "2011-12-16",
    "authors": [
        { "firstName" : "JAYESH",  "lastName" : "PATEL" }
    ],
    "categories" : [ "Databases", "NoSQL", "Programming" ],
    "publisher" : { "name" : "JAMNADAS" },
    "price" : 30.83
} )
db.books.insert( {
    "title" : "MongoDB in Action 2",
    "published" : "2011-12-16",
    "authors": [
        { "firstName" : "JIGNESH",  "lastName" : "THANKI" }
    ],
    "categories" : [ "Databases", "NoSQL", "Programming" ],
    "publisher" : { "name" : "JAMNADAS" },
    "price" : 35.30
} )
db.books.insert( {
    "title" : "Scaling MongoDB",
    "published" : "2011-03-07",
    "authors": [
        { "firstName" : "NIKHIL",  "lastName" : "DAVE" }
    ],
    "categories" : [ "Databases", "NoSQL" ],
    "publisher" : { "name" : "NIRAV PRAKASHAN" },
    "price" : 25.30
} )
db.runCommand( {
          mapReduce: "books",
          map: function() {
              for (var index = 0; index < this.authors.length; ++index) {
                  var author = this.authors[ index ];
                  emit( author.firstName + " " + author.lastName, 1 );
              }
          },
          reduce: function(author, counters) {
              count = 0;
              for (var index = 0; index < counters.length; ++index)
      {
                  count += counters[index];
```

```
            }
            return count;
        },
        out: { inline: 1 }


//map reduce example 2
//map reduce in Mongodb
//map
//reduce
use student
db.createCollection("studentdata")
db.studentdat.insert({'name':'ram','marks':40,'age':20})
db.studentdat.insert({'name':'shyam','marks':50,'age':20})
db.studentdat.insert({'name':'swati','marks':60,'age':18})
db.studentdat.insert({'name':'rahul','marks':60,'age':21})
db.studentdat.insert({'name':'riya','marks':70,'age':21})
db.studentdat.find()
//calculate the sum of marks of all students
var mapfunction=function(){emit(this.age,this.marks)}

var reducefunction=function(key,values){return Array.sum(values)}

db.studentdata.mapReduce(mapfunction,reucefunction,{'out':'Result_mapreduce'})

db.Result_mapreduce.find()



//indexes
use student
db.createCollection('emp')
db.emp.insert({'empno':001,'empnm':'mital','salary':5000})
db.emp.insert({'empno':002, 'empnm':'sheetal','salary':6000})
db.emp.insert({'empno':003, 'empnm':'neha','salary':7000})
show collections
db.emp.find()
db.emp.createIndex( { empno: 1 })
db.emp.find( { empno: 001 } )
db.emp.getIndexes()
db.emp.dropIndexes()

//upsert()
db.emp.findAndModify({query:{empnm:"mital"},
                            update:{$set:{salary:9000}},
                            upsert:true})
```