```
#show all databases
show dbs;

#create and switch to Database
use databasename;

#drop Database
db.dropDatabase();

#create Collection
db.createCollection("collection_name");

#rename Collection
db.collection_name.renameCollection("new_collection_name");

#delete/drop Collection
db.collection_name.drop();

#insert Documents in a collection
db.collection_name.insertOne({feild1:"value", feild2: value});
db.collection_name.insertMany([
	{feild1:"value", feild2: value},
	{feild1:"value", feild2: value},
	{feild1:"value", feild2: value},
	{feild1:"value", feild2: value}
]);

#display all documents in a collection
db.collection_name.findOne("maching", projections/display_nodes)
db.collection_name.find("maching", display_nodes)
db.collection_name.findOne({}) //return single documents and display all associated keys
db.collection_name.find({}) //return all documents and display all associated keys

db.collection_name.findOne({}, {model:1,make:1}) //return single documents but display only
model and make keys
db.collection_name.find({}, {model:1,make:1}) //return all documents but display only model and
make keys

db.collection_name.findOne({model:"benz"}) //return single documents with model benz
db.collection_name.find({model:"benz"}) //return all documents with model benz

db.collection_name.find({model:"benz"}, {model:1,make:1}) //return all documents with model
benz but display only model and make keys
db.collection_name.find({model:"benz", "engine.cc":2500}) //return all documents with model
benz and engine cc 2500
db.collection_name.find({model:"benz", "engine.cc":2500},{model:1,make:1})//return all
documents with model benz and engine cc 2500 but display only model and make keys

#update a document in a coolection
db.collection_name.updateOne("maching", "updateKeyValuePair");
db.collection_name.updateMany("maching", "updateKeyValuePair");
db.collection_name.updateOne({model:"benz"},{$set:{color:"red", origin:"delhi"}})// checks for model benz and adds/update the keyvalue
db.collection_name.updateMany({model:"benz"},{$set:{color:"red", origin:"delhi"}})// checks for model benz and adds/update the keyvalue

db.collection_name.updateOne({model:"benz"},{$unset:{color:"red", origin:"delhi"}})
db.collection_name.updateMany({model:"benz"},{$unset:{color:"red", origin:"delhi"}})

db.collection_name.updateOne({model:"benz"},{$push:{features:"heatedSeats"}}) //adding a new array element to a feature
db.collection_name.updateMany({model:"benz"},{$push:{features:"heatedSeats"}}) //adding a new array element to a feature

db.collection_name.updateOne({model:"benz"},{$pull:{features:"heatedSeats"}})
db.collection_name.updateMany({model:"benz"},{$pull:{features:"heatedSeats"}})

db.collection_name.updateOne({model:"benz"},{$push:{features:{$each:["voice","charging"]}}})
db.collection_name.updateMany({model:"benz"},{$push:{features:{$each:["voice","charging"]}}})

#upsert create a new doc if matching citeria is not met
db.collection_name.updateMany({model:"benz"},{$set:{color:"red", origin:"delhi"}},
{upsert:true})

#delete a document in a coolection
db.collection_name.deleteOne("maching");
db.collection_name.deleteMany("maching");

db.collection_name.deleteOne({model:"benz", "engine.cc":2500})
db.collection_name.deleteMany({model:"benz", "engine.cc":2500})

#Import and  export collections using json file
mongoimport jsonfile.json -d databasename -c collectionname
mongoimport jsonfile.json -d databasename -c collectionname --jsonArray
mongoexport

##-------------Operators------------------------

#relational operator
equal $eq
not equal $ne
less than $lt
less than equal $lte
greater than $gt
greater than equal $gte
in $in
not in $nin
db.collection_name.find({"engine.cc":{$eq:2400}})
db.collection_name.find({"engine.cc":{$ne:2400}})
db.collection_name.find({"engine.cc":{$lt:2400}})
db.collection_name.find({"engine.cc":{$lte:2400}})
db.collection_name.find({"engine.cc":{$gt:2400}})
db.collection_name.find({"engine.cc":{$gte:2400}})
db.collection_name.find({"engine.cc":{$in:[2400,3400]}})
db.collection_name.find({"engine.cc":{$nin:[2400,3400]}})
```

#logical operator
And $and
Or $or
Nor $nor

```
db.collection_name.find({
   $and:[
      {fueltype:"desiel"},
      {"engine.cc" : 3456},
      {sunroof:false}
   ]
})

db.collection_name.find({
   $or:[
      {fueltype:"desiel"},
      {"engine.cc" : 3456},
      {sunroof:false}
   ]
})

db.collection_name.find({
   $nor:[
      {fueltype:"desiel"},
      {"engine.cc" : 3456},
      {sunroof:false}
   ]
})

db.collection_name.find({

      {fueltype:{$not:"desiel"}},
      {"engine.cc" :{$not:{3456}},
      {sunroof:false}
})
```

#Element operator:
$exists
$type

```
db.collection_name.find({color:{$exists:true}});
db.collection_name.find({color:{$type:"string"}});
```

```
#Array operator
$size
$all
$elemMatch
db.collection_name.find({hobies:{$size:4}});
db.collection_name.find({hobies:{$all:["play", "read"]}});
db.collection_name.find({hobies:{$elemMatch:{"street": "456 Mt Gt","city":
"ght","zip":"345"}}});

#find number of documents
#Cursor Methods
count
db.collection_name.find({model:"benz"}).count()
sort
db.collection_name.find({model:"benz"}).sort({"engine.cc":1}) -1 for decending
limit
db.collection_name.find({model:"benz"}).limit(10) show 10 records
skip
db.collection_name.find({model:"benz"}).skip(3) skip 3 records

#-----------Aggregrate Framework

db.collection_name.aggregrate([
   //stageone
   {$match:{size:"medium"}},
   //stage2
   {$group:{_id:"$name", totalQuantity:{$sum;"$quantity"}}}
]);


# grouping Stage  =>$grouping
db.collection_name.aggregate([
{$group:
   {_id:"$maker",
   totalCars:{$sum:1}// retriew only one document
   }
}])

db.collection_name.aggregrate([
{$group:
   {_id:"$maker",
   avgPrice:{$avg:"$price"}//retrieve price value
   }
}])

#matching Stage  => $match
db.collection_name.aggregate([
{$match:
   {maker:"hyundai",
   price:{$gt:3000}
   }
}])

#projection Stage  =>   $project
db.collection_name.aggregate([
{$project:
   {maker:1,price:0,model:1}
}])

#sorting Stage  =>   $sort
db.collection_name.aggregate([
{$sort:
   {maker:1,price:-1,model:1}
}])

#limit Stage  =>   $limit
db.collection_name.aggregate([
   {$limit:1}
])

#skip Stage  =>   $skip
db.collection_name.aggregate([
   {$skip:1}
])

#sort By Count  stage =>$sortByCount
db.collection_name.aggregate([
{$sortByCount:"$maker"}
])

#unwind   stage =>$unwind
db.collection_name.aggregate([
{$unwind:"$owner"} //array or object key in a doccument
])

#filter   stage =>$filter
db.collection_name.aggregate([
{
   $project:{
    name:1,
    showValue:{
          $filter:{
          input: "$value",
          as: 'val',
          cond: {$gt:["$$val",30]}
          }
      }
   }
}
])
```

```
#string operation on Aggregrate
db.collection_name.aggregate([
{
   $project:{
    name:1,
    showValue:{$toUpper:{$concat:["$maker"," ", "$model"]}}
   }
}
])

#regexMatch
db.collection_name.aggregate([
{
   $project:{
    name:1,
    isdesiel:{$regexMatch:{
               input:"$fueltype",
               regex:"Dies",
               options: i
               }
    }
   }
}
])


#Arithematic Operation
db.collection_name.aggregate([
{
   $project:{
    name:1,
    sum:{
      $add:[2,3,4..n]
    },
    newPrice:{
      $add:["$price",1000]
    },
   }
}
])

#conditional Operator

db.collection_name.aggregate([
{
   $project:{
    name:1,
    fuelCategory:{
      $cond:{
         if:{$eq:["$fuelType","Petrol"]},
         then: "petrol",
            else: "no petrol
        }
    }
}
])

db.collection_name.aggregate([
{
    $project:{
     name:1,
     priceCategory:{
        $switch:{
            branches:[
            {case:{$lt:["$price",67890]},then:"budject"},
            {case:{$gt:["$price",67890]},then:"mid"},
            ],
           default: "unknown"
        }
    }
}
])

#Date Operator

db.collection_name.aggregate([
{
    $project:{
     name:1,
     newAddedDate:{
        $dateAdd:{
            startDate: newDate(),
            unit: "day",
            ammount:10,
        }
    }
}
])

#$out operation on Aggregrate
db.collection_name.aggregate([
{
    $project:{
     name:1,
     showValue:{$toUpper:{$concat:["$maker"," ", "$model"]}}
    },
    {$out:"hyndai_cars"}
}
])

#variables operation on Aggregrate
db.collection_name.aggregate([
{
    $project:{
     name:1,
     currentDate: "$$NOW",//system Defined

    },
    {$out:"hyndai_cars"}
}
])

# $lookup on Aggregrate
db.users.aggregate([
    {$lookup:{
        from:"orders",
        localField:"_id",
        foreignField:"user_id",
        as: "orders"
    }}
])

#validation in mongodb collections
db.createCollection("users3", {
    validator: {
        $jsonSchema: {
            bsonType: "object",
            required: ["name", "phone"],
            properties:{
                name:{
                    bsonType: "string",
                    description: "Name should be string"
                }
            }
        }
    },
    validationLevel: "strict",
    validationAction: "error"
})

#create indexes in collection documents
db.movies.find({title: 'The Ace of Hearts'}).explain("executionStats")
db.users.createIndex({ name: 1 })
db.users.dropIndex("name")


#multi datatype document

db.collection_name.insertMany([
            {
        name:"Dragon zz",
        age: 23,
        married: false,
        dob: ISODate("2000-09-08T08:00:09Z"),
        weight:56.90,
        kids: null,
        hobbies: ["music","sports"],
        address: {
                "street": "123 Mt Gt",
                "city": "mlore",
                "zip":"3456"
            }
    },
            {
        name:"Iron zz",
        age: 50,
        married: true,
        dob: ISODate("2001-09-08T08:00:09Z"),
        weight:56.90,
        kids: 3,
        hobbies: ["games","sports"],
        address: {
                "street": "456 Mt Gt",
                "city": "ght",
                "zip":"345"
            }
    },
            {
        name:"Siper zz",
        age: 34,
        married: false,
        dob: ISODate("2003-09-08T08:00:09Z"),
        weight:56.90,
        kids: null,
        hobbies: ["music","cricket"],
        address: {
                "street": "444 Mt Gt",
                "city": "chenai",
                "zip":"5677"
            }
    },
            {
        name:"flash zz",
        age: 33,
        married: true,
        dob: ISODate("2004-09-08T08:00:09Z"),
        weight:56.90,
        kids:null,
        hobbies: ["music","running"],
        address: {
                "street": "1235555Mt Gt",
                "city": "sewrt",
                "zip":"4567"
            }
        }
    ]);
    {
     "maker": "Tata",
     "model": "Nexon",
     "fuel_type": "Petrol",
     "transmission": "Automatic",
     "engine": {
                "type": "Turbocharged",
                "cc": 1199,
                "torque": "170 Nm"
      },
     "features": [
                "Touchscreen",
                "Reverse Camera",
                "Bluetooth Connectivity"
               ],
     "sunroof": false,
     "airbags": 2
     },
     {
     "maker": "Kia",
     "model": "Seltos",
     "fuel_type": "Petrol",
     "transmission": "Manual",
     "engine": {
                "type": "Turbocharged",
                "cc": 1300,
                "torque": "200 Nm"
      },
     "features": [
                "Touchscreen",
                "Reverse Camera",
                "Bluetooth Connectivity",
                "Parking"
               ],
     "sunroof": true,
     "airbags": 4
     },
     {
     "maker": "Maruthi",
     "model": "Ignus",
     "fuel_type": "Desiel",
     "transmission": "Automatic",
     "engine": {
                "type": "combusion",
                "cc": 2300,
                "torque": "340 Nm"
      },
     "features": [
                "Reverse Camera",
                "Bluetooth Connectivity",
                "Parking"
               ],
    "sunroof": true,
    "airbags": 1
    },
    {
    "maker": "Mahindra",
    "model": "xuv700",
    "fuel_type": "Desiel",
    "transmission": "Manual",
    "engine": {
                "type": "combusion",
                "cc": 3000,
                "torque": "450 Nm"
     },
    "features": [
                "Reverse Camera",
                "Bluetooth Connectivity",
                "Parking"
               ],
    "sunroof": true,
    "airbags": 6
    },
    {
    "maker": "Renault",
    "model": "Duster",
    "fuel_type": "Petrol",
    "transmission": "Manual",
    "engine": {
                "type": "combusion",
                "cc": 3000,
                "torque": "450 Nm"
     },
    "features": [
                "Reverse Camera",
                "Bluetooth Connectivity",
                "Parking"
               ],
    "sunroof": false,
    "airbags": 3
    },
    {
    "maker": "mercerdes",
    "model": "benz",
    "fuel_type": "Petrol",
    "transmission": "Manual",
    "engine": {
                "type": "combusion",
                "cc": 2400,
                "torque": "340 Nm"
     },
    "features": [
                "Reverse Camera",
                "Bluetooth Connectivity",
                "Parking"
               ],
    "sunroof": true,
    "airbags": 6
    },
```