**Mongo DB access:**

Create a folder **C:\data\db**

Navigate to **C:\Program Files\MongoDB\Server\4.0\bin** and type **mongod**

Open a new command prompt and type **mongo**

**#Create a new database**

> use beginnersbook

switched to db beginnersbook

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

> db

beginnersbook

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

**#until collection is created, db will not be seen in show dbs**

> db.user.insert({name:'Chaitanya', age:30})

WriteResult({ "nInserted" : 1 })

> show dbs

admin 0.000GB

beginnersbook 0.000GB

config 0.000GB

local 0.000GB

**#Drop database**

> show dbs

admin 0.000GB

beginnersbook 0.000GB

config 0.000GB

local 0.000GB

> show dbs

admin 0.000GB

beginnersbook 0.000GB

config 0.000GB

local 0.000GB

> show dbs

admin 0.000GB

beginnersbook 0.000GB

config 0.000GB

local 0.000GB

> use beginnersbook

switched to db beginnersbook

> db.dropDatabase()

{ "dropped" : "beginnersbook", "ok" : 1 }

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

**# Creating collection in mongodb on the fly**

**Method 1 : Syntax : db.collection\_name.insert({key:value, key,value..})**

> use beginnersbook

switched to db beginnersbook

> db.dropDatabase()

{ "dropped" : "beginnersbook", "ok" : 1 }

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

> use begineersbookdb

switched to db begineersbookdb

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

> db

begineersbookdb

> db.begineersbook.insert({name :'Saranya', age:30, website: "beginnersbook.com"})

WriteResult({ "nInserted" : 1 })

> show dbs;

admin 0.000GB

begineersbookdb 0.000GB

config 0.000GB

local 0.000GB

**#To check if document is inserted successfully**

**Syntax : db.collection\_name.find()**

> db.begineersbook.find()

{ "\_id" : ObjectId("5ce240284b5f62a04ae886ac"), "name" : "Saranya", "age" : 30, "website" : "beginnersbook.com" }

> show collections

Begineersbook

**Method 2: Creating collection with options before inserting the documents**

**Syntax: db.createCollection(name, options)**

**Options contains 4 parameters:**

1. **capped – type Boolean. -> max entries a collections can have. Once limit reaches, it overwrites.**
2. **size -> type number -> max size of collection in bytes**
3. **max -> type number -> specifies max number of documents in a collection.**
4. **autoIndexId: type Boolean -> default is false, If true, automatically creates index \_id for each document.**

> db.createCollection("students")

{ "ok" : 1 }

> show collections

begineersbook

students

> db.createCollection("teachers", {capped : true, size : 9234456})

{ "ok" : 1 }

> show collections

begineersbook

students

teachers

**# Dropping collections**

> show collections

begineersbook

students

teachers

> db.teachers.drop()

true

> show collections

begineersbook

students

**#Inserting documents into collection**

> db.students.insert({name:'Shalya', age:32, email:'abc@gmail.com', course:[{name:'GCP', duration:30},{name:'Azure', duration:40}]})

WriteResult({ "nInserted" : 1 })

> db.students.find()

{ "\_id" : ObjectId("5ce2465d4b5f62a04ae886ad"), "name" : "Shalya", "age" : 32, "email" : "abc@gmail.com", "course" : [ { "name" : "GCP", "duration" : 30 }, { "name" : "Azure", "duration" : 40 } ] }

**Inserting multiple documents into collection by storing in a variable**

> var beginners = [ {"SID" :100, "SName":"Sara", "age":30}, {"SID":101,"SName":"Shalya","age":32},{"SID":102,"SName":"Vinay","age":1}]

> beginners

[

{

"SID" : 100,

"SName" : "Sara",

"age" : 30

},

{

"SID" : 101,

"SName" : "Shalya",

"age" : 32

},

{

"SID" : 102,

"SName" : "Vinay",

"age" : 1

}

]

> db.students.insert(beginners)

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 3,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

> db.students.find()

{ "\_id" : ObjectId("5ce2465d4b5f62a04ae886ad"), "name" : "Shalya", "age" : 32, "email" : "abc@gmail.com", "course" : [ { "name" : "GCP", "duration" : 30 }, { "name" : "Azure", "duration" : 40 } ] }

{ "\_id" : ObjectId("5ce247394b5f62a04ae886ae"), "SID" : 100, "SName" : "Sara", "age" : 30 }

{ "\_id" : ObjectId("5ce247394b5f62a04ae886af"), "SID" : 101, "SName" : "Shalya", "age" : 32 }

{ "\_id" : ObjectId("5ce247394b5f62a04ae886b0"), "SID" : 102, "SName" : "Vinay", "age" : 1 }

**# To print the output in json format for easy reading**

> db.students.find().forEach(printjson)

{

"\_id" : ObjectId("5ce2465d4b5f62a04ae886ad"),

"name" : "Shalya",

"age" : 32,

"email" : "abc@gmail.com",

"course" : [

{

"name" : "GCP",

"duration" : 30

},

{

"name" : "Azure",

"duration" : 40

}

]

}

{

"\_id" : ObjectId("5ce247394b5f62a04ae886ae"),

"SID" : 100,

"SName" : "Sara",

"age" : 30

}

{

"\_id" : ObjectId("5ce247394b5f62a04ae886af"),

"SID" : 101,

"SName" : "Shalya",

"age" : 32

}

{

"\_id" : ObjectId("5ce247394b5f62a04ae886b0"),

"SID" : 102,

"SName" : "Vinay",

"age" : 1

}

**# Querying document based on criteria**

**Syntax:**

**db.collection\_name.find({"field\_name":{$gt:criteria\_value}}).pretty() # greater than**

**db.collection\_name.find({"field\_name":{$lt:criteria\_value}}).pretty() # less than**

**db.collection\_name.find({"field\_name":{$ne:criteria\_value}}).pretty() # Not equal to**

**db.collection\_name.find({"field\_name":{$gte:criteria\_value}}).pretty() # greater than or equal to**

**db.collection\_name.find({"field\_name":{$lte:criteria\_value}}).pretty() # less than or equal to**

> db.students.find({SName:"Shalya"}).pretty()

{

"\_id" : ObjectId("5ce247394b5f62a04ae886af"),

"SID" : 101,

"SName" : "Shalya",

"age" : 32

}

**#Finding document having age>30**

> db.students.find({"age":{$gt:30}}).pretty()

{

"\_id" : ObjectId("5ce2465d4b5f62a04ae886ad"),

"name" : "Shalya",

"age" : 32,

"email" : "abc@gmail.com",

"course" : [

{

"name" : "GCP",

"duration" : 30

},

{

"name" : "Azure",

"duration" : 40

}

]

}

{

"\_id" : ObjectId("5ce247394b5f62a04ae886af"),

"SID" : 101,

"SName" : "Shalya",

"age" : 32

}

**#ne – not equal to**

> db.students.find({"name":{$ne:'Shalya'}}).pretty()

{

"\_id" : ObjectId("5ce247394b5f62a04ae886ae"),

"SID" : 100,

"SName" : "Sara",

"age" : 30

}

{

"\_id" : ObjectId("5ce247394b5f62a04ae886af"),

"SID" : 101,

"SName" : "Shalya",

"age" : 32

}

{

"\_id" : ObjectId("5ce247394b5f62a04ae886b0"),

"SID" : 102,

"SName" : "Vinay",

"age" : 1

}

**#pretty – used for formatting purpose**

**#update document in a collection**

> db.begineersbook.find().forEach(printjson)

{

"\_id" : ObjectId("5ce240284b5f62a04ae886ac"),

"name" : "Saranya",

"age" : 30,

"website" : "beginnersbook.com"

}

> db.begineersbook.update({"name":"Saranya"},{$set:{"name":"Sara"}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.begineersbook.find().forEach(printjson)

{

"\_id" : ObjectId("5ce240284b5f62a04ae886ac"),

"name" : "Sara",

"age" : 30,

"website" : "beginnersbook.com"

}

# update document generally updates 1 doucment even if multiple documents matches the criteria. In order to allow multiple documents update, need to add a parameter “multi”

> db.begineersbook.update({"name":"Saranya"},{$set:{"name":"Sara"}}, {multi:true})

WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })

**#Using save() documents can be updated:**

> db.teachers.save({"\_id":ObjectId("5ce260804b5f62a04ae886b1"), "name":"Saranya"})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.teachers.find().pretty()

{ "\_id" : ObjectId("5ce260804b5f62a04ae886b1"), "name" : "Saranya" }

{

"\_id" : ObjectId("5ce2609a4b5f62a04ae886b2"),

"name" : "Shalya",

"age" : 32

}

**# Delete document from a collection**

**Syntax: db.collection\_name.remove(delete\_criteria)**

> db.teachers.find().pretty()

{ "\_id" : ObjectId("5ce260804b5f62a04ae886b1"), "name" : "Saranya" }

{

"\_id" : ObjectId("5ce2609a4b5f62a04ae886b2"),

"name" : "Shalya",

"age" : 32

}

> db.teachers.find().pretty()

{ "\_id" : ObjectId("5ce260804b5f62a04ae886b1"), "name" : "Saranya" }

{

"\_id" : ObjectId("5ce2609a4b5f62a04ae886b2"),

"name" : "Shalya",

"age" : 32

}

> db.teachers.remove({"name":"Shalya"})

WriteResult({ "nRemoved" : 1 })

> db.teachers.find().pretty()

{ "\_id" : ObjectId("5ce260804b5f62a04ae886b1"), "name" : "Saranya" }

**##Removing document**

> db.students.remove({"age":32},1) **) # second parameter is just 1 if we want to delete first document matching the criteria.**

WriteResult({ "nRemoved" : 1 })

> db.students.find().pretty()

{

"\_id" : ObjectId("5ce247394b5f62a04ae886af"),

"SID" : 101,

"SName" : "Shalya",

"age" : 32

}

{

"\_id" : ObjectId("5ce247394b5f62a04ae886b0"),

"SID" : 102,

"SName" : "Vinay",

"age" : 1

}

**#To remove all documents:**

> db.teachers.find().pretty()

{ "\_id" : ObjectId("5ce260804b5f62a04ae886b1"), "name" : "Saranya" }

**> db.teachers.remove({})**

WriteResult({ "nRemoved" : 1 })

> db.teachers.find().pretty()

**MongoDB Projection:**

If we want to select particular field from all documents, then mongodb projection is used.

**Syntax:**

db.collection\_name.find({},{field\_key:1 or 0})

> db.students.find().pretty()

{

"\_id" : ObjectId("5ce247394b5f62a04ae886af"),

"SID" : 101,

"SName" : "Shalya",

"age" : 32

}

{

"\_id" : ObjectId("5ce247394b5f62a04ae886b0"),

"SID" : 102,

"SName" : "Vinay",

"age" : 1

}

{

"\_id" : ObjectId("5ce289294b5f62a04ae886b3"),

"SID" : 100,

"SName" : "Saranya",

"age" : 30

}

> db.students.find({}, {"\_id":0, SID:1}) **# 0 indicates, don’t select that column and 1 means select that column.**

{ "SID" : 101 }

{ "SID" : 102 }

{ "SID" : 100 }

#Other than \_id column, we can’t mix other columns and give both 0 and 1 value in the same projection

> db.students.find({}, {SID:1, SName:0})

Error: error: {

"ok" : 0,

"errmsg" : "Projection cannot have a mix of inclusion and exclusion.",

"code" : 2,

"codeName" : "BadValue"

}

In this case, SID is selected and Sname is not selected. We should only give the columsn which we want to select or the columns which we don’t want to select. But not both.

**#limit()**

> db.students.find().pretty()

{

"\_id" : ObjectId("5ce247394b5f62a04ae886af"),

"SID" : 101,

"SName" : "Shalya",

"age" : 32

}

{

"\_id" : ObjectId("5ce247394b5f62a04ae886b0"),

"SID" : 102,

"SName" : "Vinay",

"age" : 1

}

{

"\_id" : ObjectId("5ce289294b5f62a04ae886b3"),

"SID" : 100,

"SName" : "Saranya",

"age" : 30

}

> db.students.find({SID:{$gt:100}}).limit(1).pretty()

{

"\_id" : ObjectId("5ce247394b5f62a04ae886af"),

"SID" : 101,

"SName" : "Shalya",

"age" : 32

}

**#skip()**

**Used to skip the first document and print the second document.**

> db.students.find({SID:{$gt:100}}).skip(1).pretty()

{

"\_id" : ObjectId("5ce247394b5f62a04ae886b0"),

"SID" : 102,

"SName" : "Vinay",

"age" : 1

}

> db.students.find().skip(2).pretty()

{

"\_id" : ObjectId("5ce289294b5f62a04ae886b3"),

"SID" : 100,

"SName" : "Saranya",

"age" : 30

}

**#sort()**

**db.collection\_name.find().sort({field\_key:1 or -1})**

**1 is for ascending order and -1 is for descending order. The default value is 1.**

> db.students.find().pretty()

{

"\_id" : ObjectId("5ce247394b5f62a04ae886af"),

"SID" : 101,

"SName" : "Shalya",

"age" : 32

}

{

"\_id" : ObjectId("5ce247394b5f62a04ae886b0"),

"SID" : 102,

"SName" : "Vinay",

"age" : 1

}

{

"\_id" : ObjectId("5ce289294b5f62a04ae886b3"),

"SID" : 100,

"SName" : "Saranya",

"age" : 30

}

> db.students.find().sort({'SName':1}).pretty()

{

"\_id" : ObjectId("5ce289294b5f62a04ae886b3"),

"SID" : 100,

"SName" : "Saranya",

"age" : 30

}

{

"\_id" : ObjectId("5ce247394b5f62a04ae886af"),

"SID" : 101,

"SName" : "Shalya",

"age" : 32

}

{

"\_id" : ObjectId("5ce247394b5f62a04ae886b0"),

"SID" : 102,

"SName" : "Vinay",

"age" : 1

}

> db.students.find().sort({'SName':-1}).pretty()

{

"\_id" : ObjectId("5ce247394b5f62a04ae886b0"),

"SID" : 102,

"SName" : "Vinay",

"age" : 1

}

{

"\_id" : ObjectId("5ce247394b5f62a04ae886af"),

"SID" : 101,

"SName" : "Shalya",

"age" : 32

}

{

"\_id" : ObjectId("5ce289294b5f62a04ae886b3"),

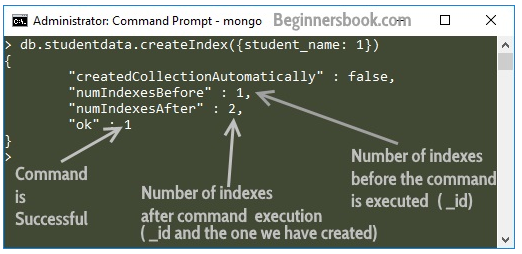
"SID" : 100,

"SName" : "Saranya",

"age" : 30

}

**#Index()**



> db.students.createIndex({SName:1})

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"ok" : 1

}

**# To view the list of indexes in a collection.**

> db.students.getIndexes()

[

{

"v" : 2,

"key" : {

"\_id" : 1

},

"name" : "\_id\_",

"ns" : "begineersbookdb.students"

},

{

"v" : 2,

"key" : {

"SName" : 1

},

"name" : "SName\_1",

"ns" : "begineersbookdb.students"

}

]

**#Drop index**

> db.students.dropIndex({SName:1})

{ "nIndexesWas" : 2, "ok" : 1 }

> db.students.getIndexes()

[

{

"v" : 2,

"key" : {

"\_id" : 1

},

"name" : "\_id\_",

"ns" : "begineersbookdb.students"

}

]

**#To drop all the indexes**

> db.students.dropIndexes()

{

"nIndexesWas" : 1,

"msg" : "non-\_id indexes dropped for collection",

"ok" : 1

}

\_id index cannot be deleted only the indexes that we created will only be deleted.