

**Mongo.exe and MongoD.exe:**

Mongo.exe is a mongo shell command prompt to connect/transactions/scripts to mongo DB.

MongoD.exe is a mongo DB background command prompt.

**Scripts/Commands:**

* **Use dbname**

To switch to database, creates new database if there is no database with specified database name.

* **Show dbs**

List out the database names, it won’t list out if the database doesn’t have at least one document in that specific db.

* **db**

Display name of database we have been using currently.

* **db.myCollection.insert({“Name”:”Vigneshwara”})**

to insert a document into ‘myCollection’ collection.

**db.myCollection.insert(**

**[**

**{“Name”:”Vigneshwara”},**

**{“Name”:”Vigneshwara2”}**

**]**

**)**

To inert multiple documents into single collection in one query

* **db.dropDatabase()**

To delete database, here ‘db’ refers to the current working database.

* **db.createCollection(“CollectionNamehere”)**

**OR**

* **db.myCollection.insert({“Name”:”Vigneshwara”})**

To Create collection

* **show collections**

to list out all collections list

* **db.myCollection.drop()**

to delete ‘myCollection’ collection

* ***Querying collections and documents***

**List all documents in a collection**

db.myCollection.find()

**List all documents in a collection like json style**

db.myCollection.find().pretty()

to get first document in a collection

db.myCollection.findOne()

to get first document in a collection with condition name should be as “Vigneshwara”

db.myCollection.findOne({“Name”:”Vigneshwara”})

to get all documents in a collection with condition name should be as “Vigneshwara”

db.myCollection.find({

{“Name”:”Vigneshwara”}

})

to get all documents in a collection with condition age should be as “15”

db.myCollection.find({

{“Age”:”15”}

})

to get all documents in a collection with condition age should be greater than “15”

db.myCollection.find({

{“Age”:{$gt:”15”}}

})

to get all documents in a collection with condition age should be greater than or equal to “15”

db.myCollection.find({

{“Age”:{$gte:”15”}}

})

to get all documents in a collection with condition age should be less than “15”

db.myCollection.find({

{“Age”:{$lt:”15”}}

})

to get all documents in a collection with condition age should be less than or equal to “15”

db.myCollection.find({

{“Age”:{$lte:”15”}}

})

to get all documents in a collection with condition age should not be equal to “15”

db.myCollection.find({

{“Age”:{$ne:”15”}}

})

To get all documents where name and age are both meet condition

db.myCollection.find({“Name”:”Vigneshwara”,”Age”:”25”})

db.myCollection.find(

$or:[{“Name”:”Vigneshwara”},{“Age”:”15”}]

)

To get all documents which match below condition

Name as ‘Ram’ and then Name as “Vigneshwara” or age is “15”

db.myCollection.find(

{

“Name”:”Ram”,

$or:[{“Name”:”Vigneshwara”},{“Age”:”15”}]

}

)

* ***Update Documents***

**To update only one record by default first match**

db.myCollection.update(

{ “Id”:ObjectId(“DJDH8S7ISHSUYDK”)},

{$set:{“Name”:”Vigneshwara Name Updated”}}

)

**To update more than one document**

db.myCollection.update(

{ “Name”:”Vigneshwara”},

{$set:{“Name”:”Vigneshwara Name Updated”}},

{multi:true}

)

**To Update/Insert document based on existing record or not**

db.myCollection.save({ “Id”:ObjectId(“DJDH8S7ISHSUYDK”),”Name”:”Vigneshwara Update”})

* ***Delete Documents***

**To delete all documents**

db.myCollection.remove()

**To delete records based on condition**

db.myCollection.remove({ “Id”:ObjectId(“DJDH8S7ISHSUYDK”)})

**To delete record even condition matches more records**

db.myCollection.remove({ “Id”:ObjectId(“DJDH8S7ISHSUYDK”)},1)

* ***Projection In Mongo DB (Select only specific columns)***

**db.myCollection.find({},{“Name”:1})** //it specifies that get Name column(True i.e:1)

**db.myCollection.find().skip(2).limit(4);** //skips first two documents and list the 4 documents after skipping first 2 documents.

**db.myCollection.find().sort({“Name”:1});** //sort based on name in ascending order

**db.myCollection.find().sort({“Name”:-1});** //sort based on name in descending order

* ***Creating Index on collection***

Creating index on Name column. i.e: 1 indicates true here

**db.myCollection.ensureIndex({“Name”:1})**

**Aggregate results**

db.myCollection.aggregate([{$group:{\_id:”$Name”,results:{$sum:”$Age”}}}])

db.myCollection.aggregate([

$group:{

\_id:”$Geneder”,minage:{$min:”$age”}

}

])

db.myCollection.aggregate([

$group:{

\_id:”$Geneder”,maxage:{$max:”$age”}

}

])

* ***Backup and restore mongo database***

**Below command is used to backup/restore all databases in mongo**

* + mongodump **–to backup**
  + mongorestore **–to restore all databases**

**Below command is used to backup/restore specific databases in mongo**

* + mongodump --db databasename
  + mongorestore –db databasename databasepath (database path is were backup database is available )

**Ex:** **mongorestore –db test2 dump/test2**

**Below command is used to backup/restore specific collection in mongo**

* + mongodump --db databasename –collection collectionname
  + mongorestore –db databasename databasepath –collection collectionname collectionpath

**Ex:** **mongorestore –db test2 –collection myCollection dump\test2\myCollection.bson**