>use mydb

switched to db mydb

To check your currently selected database, use the coommand **db**

>db

mydb

If you want to check your databases list, use the command **show dbs**.

>show dbs

db.movie.insert({"name":"test"})

Basic syntax of **dropDatabase()** command is as follows −

db.dropDatabase()

**Example**

First, check the list of available databases by using the command, **show dbs**.

>show dbs

local 0.78125GB

mydb 0.23012GB

test 0.23012GB

>

If you want to delete new database **<mydb>**, then **dropDatabase()** command would be as follows −

>use mydb

switched to db mydb

>db.dropDatabase()

>{ "dropped" : "mydb", "ok" : 1 }

>

Now check list of databases.

>show dbs

local 0.78125GB

test 0.23012GB

In this chapter, we will see how to create a collection using MongoDB.

## The createCollection() Method

MongoDB **db.createCollection(name, options)** is used to create collection.

### Syntax

Basic syntax of **createCollection()** command is as follows −

db.createCollection(name, options)

>use test

switched to db test

>db.createCollection("mycollection")

You can check the created collection by using the command **show collections**.

>show collections

The following example shows the syntax of **createCollection()** method with few important options −

>db.createCollection("mycol", { capped : true, autoIndexId : true, size :

6142800, max : 10000 } )

{ "ok" : 1 }

>

In MongoDB, you don't need to create collection. MongoDB creates collection automatically, when you insert some document.

>db.tt.insert({"name" : "test"})

>show collections

db.inventory.insertMany([

{ item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },

{ item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },

{ item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },

{ item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },

{ item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }

]);

Find /where

db.inventory.find( { status: "D" } )

db.inventory.find( { status: { $in: [ "A", "D" ] } } )

db.inventory.find( { status: "A", qty: { $lt: 30 } } )

db.inventory.find( { $or: [ { status: "A" }, { qty: { $lt: 30 } } ] } )

db.inventory.find( {

status: "A",

$or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]

} )

## The drop() Method

MongoDB's **db.collection.drop()** is used to drop a collection from the database.

### Syntax

Basic syntax of **drop()** command is as follows −

db.COLLECTION\_NAME.drop()

# MongoDB - Datatypes

MongoDB supports many datatypes. Some of them are −

* **String** − This is the most commonly used datatype to store the data. String in MongoDB must be UTF-8 valid.
* **Integer** − This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
* **Boolean** − This type is used to store a boolean (true/ false) value.
* **Double** − This type is used to store floating point values.
* **Min/ Max keys** − This type is used to compare a value against the lowest and highest BSON elements.
* **Arrays** − This type is used to store arrays or list or multiple values into one key.
* **Timestamp** − ctimestamp. This can be handy for recording when a document has been modified or added.
* **Object** − This datatype is used for embedded documents.
* **Null** − This type is used to store a Null value.
* **Symbol** − This datatype is used identically to a string; however, it's generally reserved for languages that use a specific symbol type.
* **Date** − This datatype is used to store the current date or time in UNIX time format. You can specify your own date time by creating object of Date and passing day, month, year into it.
* **Object ID** − This datatype is used to store the document’s ID.
* **Binary data** − This datatype is used to store binary data.
* **Code** − This datatype is used to store JavaScript code into the document.
* **Regular expression** − This datatype is used to store regular expression

## The insert() Method

To insert data into MongoDB collection, you need to use MongoDB's **insert()** or **save()** method.

### Syntax

The basic syntax of **insert()** command is as follows −

>db.COLLECTION\_NAME.insert(document)

### Example

>db.mycol.insert({

\_id: ObjectId(7df78ad8902c),

title: 'MongoDB Overview',

description: 'MongoDB is no sql database',

by: ‘test',

url: 'http://www.techvisionit.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100

})

To insert multiple documents in a single query, you can pass an array of documents in insert() command.

### Example

>db.post.insert([

{

title: 'MongoDB Overview',

description: 'MongoDB is no sql database',

by: 'test',

url: 'http://www. techvsonit.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100

},

{

title: 'NoSQL Database',

description: "NoSQL database doesn't have tables",

by: 'test',

url: 'http://www.techvsonit.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 20,

comments: [

{

user:'user1',

message: 'My first comment',

dateCreated: new Date(2013,11,10,2,35),

like: 0

}

]

}

])

## The find() Method

To query data from MongoDB collection, you need to use MongoDB's **find()** method.

### Syntax

The basic syntax of **find()** method is as follows −

>db.COLLECTION\_NAME.find()

**find()** method will display all the documents in a non-structured way.

## The pretty() Method

To display the results in a formatted way, you can use **pretty()** method.

### Syntax

>db.mycol.find().pretty()

db.mycol.find({"likes": {$gt:10}, $or: [{"by": "test"},

{"title": "MongoDB Overview"}]}).pretty()

{

"\_id": ObjectId(7df78ad8902c),

"title": "MongoDB Overview",

"description": "MongoDB is no sql database",

"by": "test",

"url": "http://www.techvisionit.com",

"tags": ["mongodb", "database", "NoSQL"],

"likes": "100"

}

Update

>db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB test’}})

>db.mycol.find()

## Remove Only One

If there are multiple records and you want to delete only the first record, then set **justOne** parameter in **remove()** method.

>db.COLLECTION\_NAME.remove(DELETION\_CRITERIA,1)

## Remove All Documents

If you don't specify deletion criteria, then MongoDB will delete whole documents from the collection. **This is equivalent of SQL's truncate command.**

>db.mycol.remove()

>db.mycol.find()

>

## The Limit() Method

To limit the records in MongoDB, you need to use **limit()** method. The method accepts one number type argument, which is the number of documents that you want to be displayed.

### Syntax

The basic syntax of **limit()** method is as follows −

>db.COLLECTION\_NAME.find().limit(NUMBER)

## MongoDB Skip() Method

Apart from limit() method, there is one more method **skip()** which also accepts number type argument and is used to skip the number of documents.

### Syntax

The basic syntax of **skip()** method is as follows −

>db.COLLECTION\_NAME.find().limit(NUMBER).skip(NUMBER)

### Example

Following example will display only the second document.

>db.mycol.find({},{"title":1,\_id:0}).limit(1).skip(1)

{"title":"NoSQL Overview"}

>

## The sort() Method

To sort documents in MongoDB, you need to use **sort()** method. The method accepts a document containing a list of fields along with their sorting order. To specify sorting order 1 and -1 are used. 1 is used for ascending order while -1 is used for descending order.

### Syntax

The basic syntax of **sort()** method is as follows −

>db.COLLECTION\_NAME.find().sort({KEY:1})

Following example will display the documents sorted by title in the descending order.

>db.mycol.find({},{"title":1,\_id:0}).sort({"title":-1})