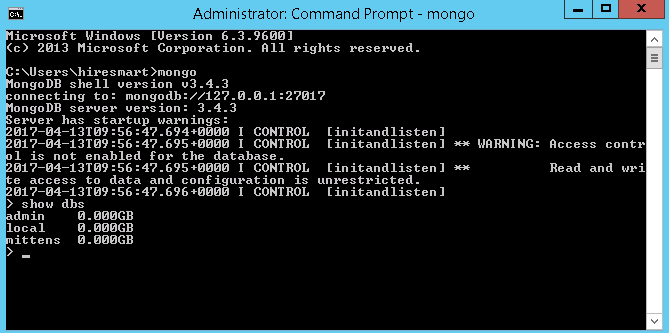
**HandsOn MongoDB**

**Recipe1:** **Connect To The MongoDB Database**

Open Command prompt and type mongod.

Open another Command prompt and type mongo after that type show dbs.



Hurray your database is up now.

Create Database: To create database in mongodb type below command in command line: use ‘databaseName’ e.g. use mittens

**Recipe2:** **Insert Operations**

Create or insert operations adds new document to a collection. If the collection does not currently exist, insert operation will create the collection.

MongoDB provides the following methods to insert document(s) into a collection:

db.collection.insertOne() : db.meows.insertOne( { text: "Hello", username: “user1” } );

db.collection.insertMany(): db.meows.insertMany( [

{ text: "Hello everyone", username: “user2” },

{ text: "Hi, This is user3", username: “user3” }, { text: "Bye", username: “user3” }] );

**Recipe3**: **Read Operation**

Read operation retrieves documents from a collection; i.e. query a collection for documents. MongoDB provides the following methods to read documents from a collection:

db.collection.find()

I.e. In Mittens database db.meows.find()

You can also specify query filters or criteria that identify the documents to return.

Exp1: db.meows.find({username:”user3”})

This operation corresponds to the following SQL statement:

SELECT \* FROM meows WHERE username = "user3"

Exp2: db.meows.find( { text: "Bye", username: “user3” } )

The operation corresponds to the following SQL statement:

SELECT \* FROM inventory WHERE text = "Bye" AND username = "user3"

**Recipe4**: **Update Operations**

Update operation modifies existing documents in a collection. MongoDB provides the following methods to update documents of a collection:

1: db.meows.updateOne( { "username" : "user1" },

{ $set: { "text" : “Hello Hello” } } );

2: db.meows.update( {" username ": " user1"}, ).

{ $set: { "text" : “Hello Hello” } } ); // Update only applied to first matching document

3: db.meows.update( {" username ": " user1"}, ).

{ $set: { "text" : “Hello Hello” } } ,

{"multi": true} );// When multi is true, the update modifies all matching documents.

db.collection.updateMany()

You can also specify criteria, or filters, that identify the documents to update. These filters use the same syntax as read operations.

**Recipe5**: **Some more exercise with update operation**

**Update a Document’s Count**

We can use the $inc operator to increment the count of an existing log document. We can increment by any number, positive or negative. If field doesn’t exist, it gets created with the value.

db.meows.update( {" username ": " user1"},

{"$inc": {"count": 1}} );

Here Count incremented by 1.

**Update a Non-existing Document**

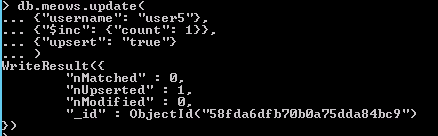
If we run the update on a meow that doesn’t exist, then nothing will happen. To handle this situation we can use upsert option. The upsert option either updates an existing document or creates a new one.

db.meows.update( {" username ": " user5"},

{"$inc": {"count": 1}},

{"upsert": true} );

Here upsert creates a document using the values from the query and update parameter.



If we run the same update again, the update will act normally and upsert won’t create another document.

C:\Users\462439\Desktop\2.PNG

**Updating a Field Name With $rename**

We can use $rename to change field names.

db.meows.update( {}, {"$rename": {"text": "meow"}}, {"multi": true} )

Here “text” is field to rename and “meow is new field name”.

**Recipe6:** **Delete Operations**

Delete operation removes document(s) from a collection. MongoDB provides the following methods to delete documents from a collection:

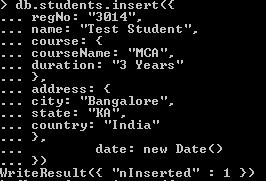
db.collection.deleteOne() : db.meows.deleteOne( { "\_id" : ObjectId("563237a41a4d68582c2509da") } );

db.collection.deleteMany() : db.meows.deleteMany( { "username" : "user3" } );

You can specify criteria, or filters, that identify the documents to remove. These filters use the same syntax as read operations.

**Recipe7:** **Let’s try with some complex schema**

Let us insert a document to a student collection. You must be connected to a database for doing any insert. It is done as follows:



Date() returns the current date as a string in the mongo shell.

new Date() returns the current date as a Date object. The mongo shell wraps the Date object with the ISODate helper. The ISODate is in UTC.

Let’s add one more Item

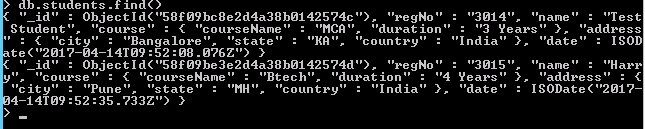


Note that an entry has been made into the collection called student.

**1:** **Querying a document from a collection (Read)**

The find() command will retrieve all the documents of the given collection.

db.collection\_name.find()

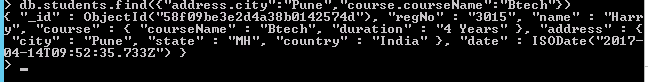


Let us retrieve the record from the student collection where the attribute regNo is 3014 and the query for the same is as shown below:

db.students.find({"regNo":"3014"})

One more exercise let us retrieve the record from the student collection with city = "Pune" and courseName = "Btech" and the query for the same is as shown below:

db.students.find({"address.city":"Pune","course.courseName":"Btech"})



**2:** **Updating a document in a collection (Update)**

In order to update specific field values of a collection in MongoDB, run the below query.

db.collection\_name.update()

Let us update the attribute name of the collection student for the document with regNo 3014.

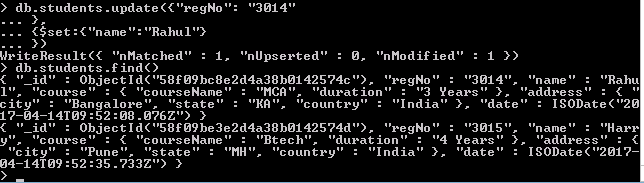
db.students.update({"regNo": "3014"

},

{$set:{ "name":"Rahul"}

})

You will see the following in the Command Prompt:

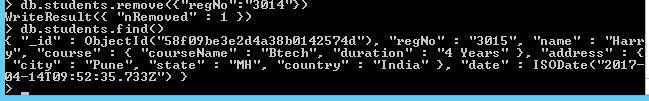


**3:** **Removing an entry from the collection (Delete)**

In order to delete an entry from a collection, run the command as shown below:

db.collection\_name.remove({"fieldname":"value"})

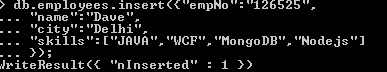
For Example:



After running the remove() method, the entry has been deleted from the student collection.

**Recipe8:** **Let’s try with Array Type Document.**

Create an Employee collection as below in mittens database

****

Similarly add some more records.

**The Dilemma of Updating an Array**

db.employees.update( {"skills": "MongoDB"}, {$set: {"skills": "Mean Stack"}} )

// It would overwrite the entire array and set it as Mean Stack

**Updating Array Values by its Location**

Since array values are treated individually, we can update a single value by specifying its location in the array using dot notation.

db.employees.update({"name": "Dave"},{$set:{"skills.1":"WPF"}})

C:\Users\462439\Desktop\2.PNG

**Updating Values without Knowing Position**

The positional operator is a placeholder that will set the proper position for the value specified in the query parameter.

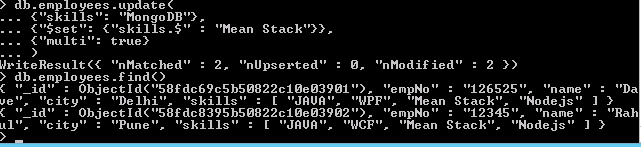
db.employees.update(

{"skills": "MongoDB"},

{"$set": {"skills.$" : "Mean Stack"}},

{"multi": true}

)



**Recipe9:** **Comparison Query Operators.**

We can use comparison query operators to match documents based on the comparison of a specified value.

Common Comparisons

**$gt** greater than

**$gte** greater than or equal to

**$lt** less than

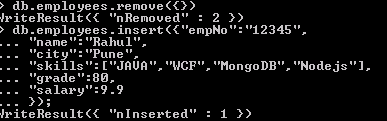
**$lte** less than or equal to

**$ne** not equal to

Let’s create new employee’s collection with some more fields.

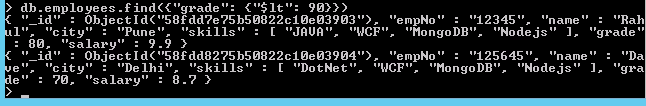
db.employees.remove({}); // It will remove all records.

Add document in employee collection with grade and salary field as below

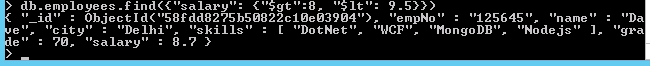


Similarly add some more records.

**Excercise1:** db.employees.find({"grade": {"$lt": 90}}) // We can match the appropriate documents by using the $lt comparison operator.



**Excercise2:** db.employees.find({"salary": {"$gt":8, "$lt": 9.5}})

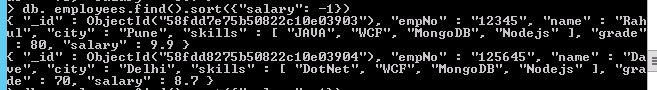


**Excercise3:** Sorting

db. employees.find().sort({"salary ": 1})// To order ascending



db. employees.find().sort({"salary ": -1})// To order descending



**Happy Coding ☺**