

Download Mongo DB from [www.mongodb.com](http://www.mongodb.com)

Downloaded the community version for windows, Download the complete version when prompted while installing.

Go to C:>MongoDB>Server>3.6>bin

In that “mongod” is a program that run background, which allows to establishes the connection with the database

“mongo” : executes out commands in the mongoDB

To check the successful installation of MongoDB, Go the MongoDB bin directory and execute “mongod”. Just type mongod in that location : we get

**C:\data\db\ not found.,**

So create a data and db folder in C drive :: SO that mondoDB can save all its data

After that executing **“mongod”** will open a port.

Now, open another command prompt and execute mongo> To check if its connected > type db to get **“test”** message.

**Note: Why were we writing PATH in environment variables.**

**This is to execute all the files within that URL from anywhere of the command prompt.**

**Note: To Execute MongoDB Commands, we use Mongo-Chef (MongoChef is now Studio3T)**

To Get Started with **Mongo Chef: (Studio 3T)**

Click on Connect > New Connection > Give a name > Localhost > 27017 > Save >Connect

Next >

Click on **IntelliShell >** It is like **Mongo shell** which was seen on command prompt when executed mongo application inside the bin folder.

Structure of DB in MongoDB

**Database** is made up of related data called **Collections, and** each **Collection** has an **actual data** called **Documents**

# How to Create a Database

* **use testdb**

This will create a new database with name **“testdb” is it does not exist, if it exist** it just switches to that database

* **db**

To check which is the current used database

* **show dbs**

lists all the existing databases if each of them has at least one document.

* **db.testconnection.insert({“Name”:”test”})**

The above command will insert one document inside the **testConnection** of current database.

* db.dropDatabase()

To Delete the current Database.

* db.createCollection("Test")

To Create a collection named “**Test**” inside the current database.

* show collections

To show the list of collections that exists in the current DB

* db.testcollection.drop**()**

The Above method drop the “testcollection” which is in in the current Database: It returns true if the collection is dropped / or returns false if the collection was not dropped or not present the current database.

# Insert Documents to DB

db.employees.insert**(**

**{**

"EmpNo"**:**"1"**,**

"FirstName"**:**"Andrew"**,**

"LastName"**:**"Neil"**,**

"Age"**:**"30"**,**

"Gender"**:**"Male"**,**

"Skill"**:**"MongoDB"**,**

"Phone"**:**"408-1234567"**,**

"Email"**:**"Andrew.Neil@gmail.com"**,**

"Salary"**:**"80000"

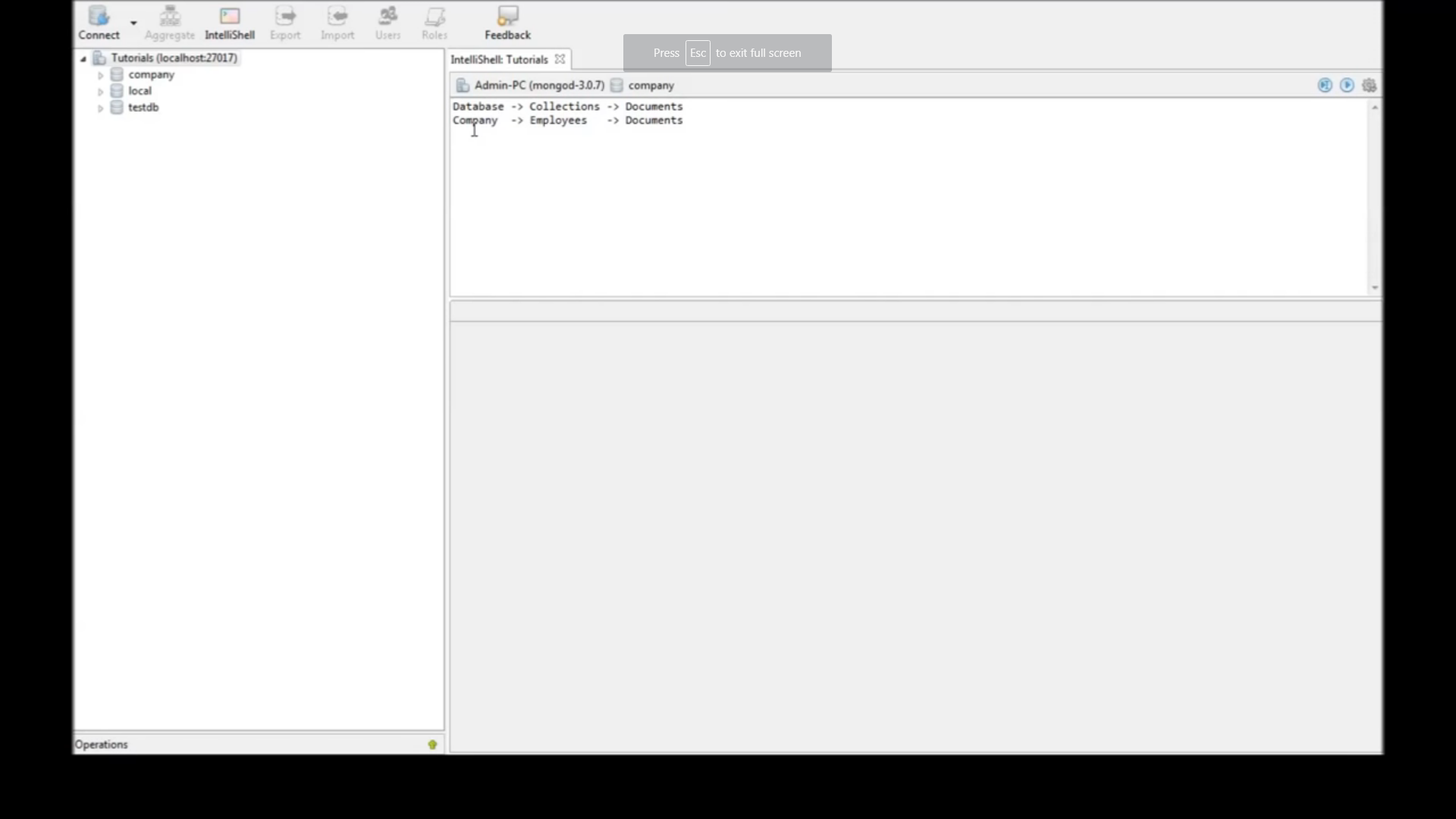
**}**

**)**

**The above code inserts one document inside the employees collection of the current db**

To insert the array of documents, use like following:

**Db.employeees.insert([{doc1},{doc2},{doc3}…….])**



# To retrieve all the documents stored in the DB: (select \*)

Db.employees.find() :

Db.employees.find().pretty()

To Retrieve first document:

db.employees.findOne**()**

To give criteria in retrieving the document

db.employees.find**({**"EmpNo" **:** "5"**})**

db.employees.find**({**"EmpNo" **:** **{**$gt **:** "30"**}})**

**$gt**: **greater than -> $gte: greater than or equal to**

**$lt: for less than🡪 $lte : less than or equal to**

**$ne: Not equal to**

# And / OR criteria

**AND Criteria**

db.employees.find**({**

"Skill”: "MongoDB"**,**"Age" **:** **{**$gt **:** "30"**}**

**})**.pretty**()**

**OR Criteria**

db.employees.find**({**

$or**:** **[{**"Skill" **:** "MongoDB"**},{**"Age" **:** **{**$gt **:** "30"**}}]**

**})**.pretty**()**

**Combination of AND / OR criteria**

**db.employees.find({**

**"Skill" : "MongoDB", $or: [{"Salary":"80000"},{"Salary":"85000"}]**

**}).pretty()**

# UPDATE Commands / it has 3 parameters

* 1st: To filter which commands needs update
* 2nd: set function to update a new value to the document
* 3rd: if Nothing performs update for only one row, even if the 1st criteria returns more than one row. To make work for multiple updates, set:: **multi:true**

To Update by Unique ID

db.employees.update**(**

**{**"\_id" **:** ObjectId**(**"5af1d7926e0e4cfe2c33ffeb"**)},**

**{**$set**:** **{**"Salary"**:**"100000"**}}**

**)**

**To Update by one of the column member:: Only the first filtered document will be updated**

db.employees.update**(**

**{**"Skill"**:**"MongoDB"**},**

**{**$set**:** **{**"Salary"**:**"135000"**}}**

**)**

**To update Multiple documents in the collection,**

db.employees.update**(**

**{**"Skill"**:**"MongoDB"**},**

**{**$set**:** **{**"Salary"**:**"150000"**}},**

**{**multi**:**true**}**

**)**

# Removing the Document from the collection

**To remove all the documents which satisfies the given criteria,**

**db.emplooyees.remove(**

**{"LastName" : "White"}**

**)**

**To delete the first matched document in the given list of selected criteria documents: Only one document is removed. ( 1 is the flag o denote one document to be removed)**

**db.emplooyees.remove(**

**{"LastName" : "White"},1**

**)**

# Selecting Documents

db.employees.find**({},{})**.pretty**()**

**the Find methods takes two arguments: 1st one for selection criteria for filtering, 2nd arguments for filtering the required fields on the document**

db.employees.find**({},{**"FirstName"**:**1**})**.pretty**()**

**The above command, filters only the FirstName along with Unique “\_Id”: Object Id**

**To get remove \_Id field, the following below command must be used. Here 1 denotes to select, and 0 denotes not to select**

db.employees.find**({},{**"FirstName"**:**1**,**"LastName"**:**1**,**"\_id"**:**0**})**.pretty**()**

# Limit, Skip and Sort

db.employees.find**()**.pretty**()**

**To Limit the First 4 Documents: This results in retrieving only first 4 Documents**

db.employees.find**({},{**"FirstName"**:**1**,** "EmpNo"**:**1**,**"\_id"**:**0**})**.pretty**()**.limit**(**4**)**

**To Skip the first four Documents and display others**

db.employees.find**({},{**"FirstName"**:**1**,** "EmpNo"**:**1**,**"\_id"**:**0**})**.pretty**()**.skip**(**4**)**

**To Skip the Fist 4 Docs and Limit the next two Docs**

db.employees.find**({},{**"FirstName"**:**1**,** "EmpNo"**:**1**,**"\_id"**:**0**})**.pretty**()**.skip**(**4**)**.limit**(**2**)**

**To Sort the Documents By (First Name) in Ascending Order, then skipping the first Document and getting the next only element : This below command for retrieving the 2nd element of the Sorted documents**

db.employees.find**({},{**"FirstName"**:**1**,** "EmpNo"**:**1**,**"\_id"**:**0**})**.pretty**()**.sort**({**"FirstName"**:**1**})**.skip**(**1**)**.limit**(**1**)**

**The Sort method if “1” Sorts in Ascending Order, “2”: Sorts in Descending Order**

# Indexing

* Their use in queries usually results in much better performance.
* They make it possible to quickly retrieve (fetch) data.
* They can be used for sorting. A post-fetch-sort operation can be eliminated.
* Unique indexes guarantee uniquely identifiable records in the database.

db.employees.ensureIndex**({**"Email"**:**1**});**

**The above commands make Email field as one of the index of the document.**

db.employees.getIndexes**();**

**The above commands get all the available index for the collection.**

db.employees.dropIndex**({**"Email"**:**1**})**

**The above command is used to drop the index for that collection. Here email is removed as index from the employees collection**

# Aggregation

Aggregations operations process data records and return computed results. **Aggregation** operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. In SQL count(\*) and with group by is an equivalent of **mongoDB aggregation**.

db.employees.aggregate**([{**$group**:** **{**\_id**:** "$Gender"**,** Total**:** **{**$sum**:** 1**}}}]);**

db.employees.aggregate**([{**$group**:** **{**\_id**:** "$Gender"**,** MaxAge**:** **{**$max**:** "$Age"**}}}]);**

db.employees.aggregate**([{**$group**:** **{**\_id**:** "$Gender"**,** MinAge**:** **{**$min**:** "$Age"**}}}]);**

# Backup and Restore in MongoDB

Make sure the Mongo bin directory is set as a PATH variable.

Go to desired folder where the backup of the db has to be created in command prompt and type below command, to back-up all the available database

* **mongodump**

To restore back all the database

* **mongorestore**

To dump individual database

* **mongodump - -db <db\_name>**

To restore individual database

* **mongorestore - -db <db\_name>**

mongodump

mongodump --db company

mongodump --db company --collection employees

mongorestore

mongorestore --db company dump/company

mongorestore --db company --collection employees dump/company/employees

# MongoDB with Node JS

* Install mongoDB driver in a specific folder by the following below command

npm install mongoDB

* Create a JS to configure the mongoDB
* var mongodb = require('mongodb');
* var MongoClient = mongodb.MongoClient;
* var url = 'mongodb://localhost:27017';
* const dbName = 'fruits';
* MongoClient.connect(url, function(err, client) {
* const db = client.db(dbName);
* if (err) {
* console.log(err);
* } else {
* console.log("Connected to ", url);
* client.close();
* }
* });
* Run the above JS file using the command below

node mongodb.js

**This is how we connect to our database using node JS**

# Inserting Document to MongoDb using Node JS

var mongodb = require('mongodb');

var MongoClient = mongodb.MongoClient;

var url = 'mongodb://localhost:27017';

const dbName = 'fruits';

MongoClient.connect(url, function(err,client) {

const db = client.db(dbName);

if(err) {

console.log(err);

} else {

console.log('Connected to ', url);

var collection = db.collection('apples');

var doc1 = {name:'red apples', coloe:'red'};

var doc2 = {name:'green apples', color:'green'};

collection.insert([doc1,doc2], function(err, res){

if (err) {

console.log(err);

} else {

console.log('%d docs inserted', res.insertedCount);

}

client.close();

});

}

});

The above code and running in Node JS will insert 2 docs inside the apples collection of fruits database.