```
Var sql = "ALTER TABLE employee2 ADD COLUMN salary INT(10)";
con.query(sql, function (err, result) {
if (err) throw err;
console.log("Table alerted");
});
});
```

```
root@aspire-pc: /home/user/Desktop/DBexample
root@aspire-pc:/home/user/Desktop/DBexample# node employee2.js
Connected!
Table altered
```

Nodejs My SQL Insert Records

INSERT INTO Statement is used to insert records in MySQL.

Example

Insert Single Record:

Insert records in "employees" table.

Create a js file named "insert" in DB example folder and put the following data into it:

```
Var mysql = require('mysql');
Var con = mysql.createConnection({
host: "localhost",
user: "root",
password: "12345"
database: "javatpoint"
});
con.connect(function(err){
if(err) throw err;
console.log("connected!");
Var sql = "INSERT INTO employees (id, name, age, city)
        VALUES ('1', 'Ajeet kumar', '27', 'Allahabad')";
```

```
con.query (sql, function (err, result){
if (err) throw err;
Console.log ("1 record inserted");
});
```

Now open command terminal and run the following command:

Node insert.js

```
root@aspire-pc: /home/user/Desktop/MongoDatabase
oot@aspire-pc:/home/user/Desktop/MongoDatabase# node insert.js
 record inserted
oot@aspire-pc:/home/user/Desktop/MongoDatabase#
```

Check the inserted record by using SELECT query:

SELECT * FROM employees;

```
root@aspire-pc: /home/user/Desktop/MongoDatabase
root@aspire-pc:/home/user/Desktop/MongoDatabase# node insertall.js
Number of records inserted: 4
root@aspire-pc:/home/user/Desktop/MongoDatabase#
```

The Result Object

When executing the insert() method a result object is returned. The result object contains information about the insertion.

It is looked like this:

```
root@aspire-pc: /home/user/Desktop/DBexample
^C
root@aspire-pc:/home/user/Desktop/DBexample# node select.js
[ RowDataPacket { id: 1, name: 'Ajeet Kumar', age: 27, city: 'Allahabad' },
  RowDataPacket { id: 2, name: 'Bharat Kumar', age: 25, city: 'Mumbai' },
  RowDataPacket { id: 3, name: 'John Cena', age: 35, city: 'Las Vegas' },
  RowDataPacket { id: 4, name: 'Ryan Cook', age: 15, city: 'CA' } ]
```

Node.js MySQL Update Records

The UPDATE command is used to update records in the table.

Example

Update city in "employees" table where id is 1.

Create a file named "update" in DBexample folder and put the following data into it.

```
Var mysql = require ('mysql');
Var con = mysql.createConnection({
host: "localhost",
user: "root"
password:"12345"
database:"javatpoint"
});
con.connect(function(err){
if (err) throw err;
Var sql = "UPDATE employees SET city = 'Delhi' WHERE
          city = 'Allahabad'";
con.query (sql, function (err, result) {
if (err) throw err;
console.log(result.affectedRows +"record(s)updated")
});
});
```

Now open command terminal and run the following command:

```
Node update.js
```

It will change the city of the id 1 is to Delhi which is prior Allahabad.

```
root@aspire-pc: /home/user/Desktop/DBexample
^C
root@aspire-pc:/home/user/Desktop/DBexample# node update.js
1 record(s) updated
```

Node.js My SQL Delete Records

The DELETE FROM command is used to delete records from the table.

Example

Delete employee from the table employees where city is Delhi.

Create a js file named "delete" in DBexample folder and put the following data into it:

```
Var my sql = require ('mysql');
 Var con = mysql.createconnection({
 host:"localhost"
 user: "root",
 password : "12345";
 database: "javatpoint"
});
 con.connect(function(err){
 if (err) throw err;
 Var sql = "DELETE FROM employees WHERE city 'Delhi";
 con.query(sql, function (err, result){
 if(err) throw err;
 console.log("Number of records deleted:" + result.
                affectedRows);
});
});
```

Now open command terminal and run the following command:

```
Node delete.js
```

```
root@aspire-pc: /home/user/Desktop/DBexample
root@aspire-pc:/home/user/Desktop/DBexample# node delete.js
Number of records deleted: 1
```

Node.js My SQL Select Records

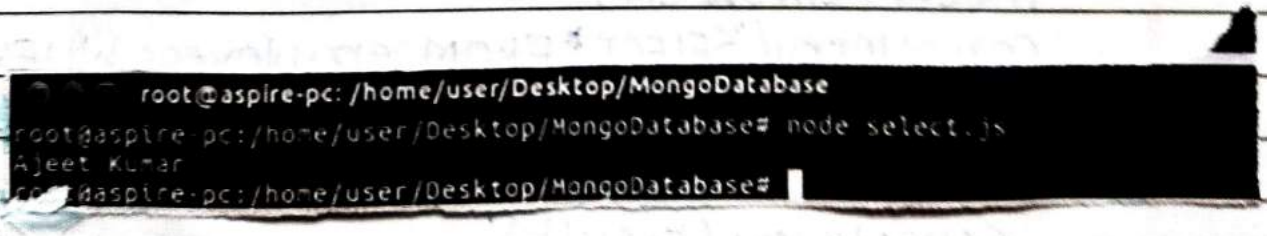
Example

Retrieve all data from the table "employees".
Create a js file named select.js having the following data in DBexample folder.

```
var mysql = require ('mysql');
var con = mysql.createConnection ({
 host: "localhost",
 user: "root",
 password: "12345",
 database: "javatpoint"
 });
 con.connect(function(err){
 if (err) throw err;
 con.query("SELECT * FROM employees", function (err, result) {
 if (err) throw err;
 console.log (result);
 });
 });
```

Now open command terminal and run the following command.

```
Node select.js
```



You can also use the statement:

```
SELECT * FROM employees;
```

```
root@aspire-pc: /home/user/Desktop/MongoDatabase
root@aspire-pc:/home/user/Desktop/MongoDatabase# node selectall.js
[ { _id: 59104e062f60cd3366ceb7da,
    name: 'Ajeet Kumar',
    age: '28',
    address: 'Delhi' },
  { _id: 59104e062f60cd3366ceb7db,
    name: 'Mahesh Sharma',
    age: '25',
    address: 'Ghaziabad' },
  { _id: 59104e062f60cd3366ceb7dc,
    name: 'Tom Moody',
    age: '31',
    address: 'CA' },
  { _id: 59104e062f60cd3366ceb7dd,
    name: 'Zahira Wasim',
    age: '19',
    address: 'Islamabad' },
  { _id: 59104e062f60cd3366ceb7de,
    name: 'Juck Ross',
    age: '45',
    address: 'London' } ]
root@aspire-pc:/home/user/Desktop/MongoDatabase#
```

Node.js MySQL SELECT Unique Record

(WHERE Clause)

Retrieve a unique data from the table "employee"
Create a js file named selectwhere.js having the following data in DBexample folder.

```
Var mysql = require ('mysql');
Var con = mysql.createConnection({
 host: "localhost",
 user : "root",
 password: "12345"
 database: "javatpoint"
 });
 con.connect(function(err) {
 if (err) throw err;
 con.querry("SELECT * FROM employess WHERE id =
            '1', function(err, result) {
 If (err) throw err;
  console.log (result);
 });
 });
```

Now open command terminal and run the

following command:

Node selectWhere.js

Node.js My SQL Select Wildcard

Retrieve a unique data by using Wildcard from the table "employees".

Create a .js file named selectwildcard.js having the following data in DBexample folder.

```
Var mysql = require('mysql');
Varcon = mysql.createConnection({
host: "localhost",
user: "root"
password: "12345"
database: "javatpoint"
});
con.connect(function(err) {
if (err) throw err.
con.query("SELECT * FROM employees WHERE city LIKE
          'A%'", function (err, result) {
if (err) throw err;
console.log (result);
});
});
```

Now open command terminal and run the following command

Node selectWildcard.js

It will retrieve the record where City start with A.

```
root@aspire-pc: /home/user/Desktop/DBexample
^C
root@aspire-pc:/home/user/Desktop/DBexample# node selectwildcard.js
[ RowDataPacket { id: 1, name: 'Ajeet Kumar', age: 27, city: 'Allahabad' } ]
```

Node.js MySQL Drop Table

The DROP TABLE command is used to delete or drop a table.

Lets drop a table named employee2.

Create a js file named "delete" in DBexample folder and put the following data into it:

```
Var mysql = require('mysql');
Var con = mysql.createConnection({
host: "localhost",
User: "root"
password: "12345",
database: "javatpoint"
});
Con.connect(function(err){
if(err) throw err;
Var sql = "DROP TABLE employee2";
Con.query(sql, function(err, result){
if(err) throw err;
console.log("Table deleted");
});
});
```

Now open command terminal and run the following command:

```
Node drop.js
```

```
root@aspire-pc: /home/user/Desktop/DBexample
^C
root@aspire-pc:/home/user/Desktop/DBexample# node drop.js
Table deleted
```

Node.js Create Connection With Mongo DB

MongoDb is a NOSQL database. It can be used with Node.js as a database to insert and retrieve data. Use the following command to start MongoDB Services:

```
Service mongodb start
```

Now, connection is created for further operations.

Node.js Mongo DB Create Database

To create a database in MongoDB. First create a Mongo Client and specify a connection URL with the correct ip address and the name of the database which you want to create.

Example

Create a folder named "MongoDatabase" as a database. Suppose you create it on Desktop. Create a file named "Createdatabase.js" within that folder and having the following code:

```
Var MongoClient = require('mongodb').MongoClient;
Var url = "mongodb://localhost:27017/MongoDatabase";
MongoClient.connect(url, function(err, db) {
if (err) throw err;
console.log("Database created!");
db.close();
});
```

Now open the command terminal and set the path where MongoDatabase exists. Now execute the following command:

Node create data base.js

```
root@aspire-pc: /home/user/Desktop/MongoDatabase
root@aspire-pc:/home/user# cd Desktop
root@aspire-pc:/home/user/Desktop# cd MongoDatabase
root@aspire-pc:/home/user/Desktop/MongoDatabase# node createdatabase.js
Database created!
root@aspire-pc:/home/user/Desktop/MongoDatabase#
```

Now database is created.

Node.js MongoDB Create Collection

MongoDB is a NoSQL database so data is stored in collection instead of table. Create Collection method is used to create a collection in MongoDB.

Example

Create a collection named "employees".

Create a js file named "employees.js", having the following data:

```
Var Mongo client = require('mongodb').MongoClient;
Var url = "mongodb://localhost:27017/Mongo Database";
Mongo Client.connect (url, function(err, db) {
if(err) throw err;
db.create Collection("employees", function(err,res) {
if(err) throwerr;
console.log("Collection is created!");
db.close();
});
});
```

Open the command terminal and run the following command:

Node employees.js

```
root@aspire-pc: /home/user/Desktop/MongoDatabase
root@aspire-pc:/home/user/Desktop/MongoDatabase# node employees.js
Collection is created!
root@aspire-pc:/home/user/Desktop/MongoDatabase#
```

Now the collection is created.

Node.js MongoDB Insert Record

The insertOne method is used to insert record in MongoDB's collection. The first argument of the insertOne method is an object which contains the name and value of each field in the record you want to insert.

Example

(Insert Single record)

Insert a record in "employees" collection.

Create a js file named "insert.js", having the following code:

```
Var MongoClient = require('mongodb').MongoClient;
Var url = "mongodb://localhost:27017/MongoDatabase";
MongoClient.connect(url, function(err,db){
if (err) throw err;
Var myobj={name:"Ajeet kumar", age:"28", address: "Delhi"};
db.collection("employees").insertone(myobj, function(err,res){
If(err) throw err;
console.log("1 record inserted");
db.close();
});
});
```

Open the command terminal and run the following command:

```
Node insert.js
```

```
root@aspire-pc: /home/user/Desktop/MongoDatabase
root@aspire-pc:/home/user/Desktop/MongoDatabase# node insert.js
1 record inserted
root@aspire-pc:/home/user/Desktop/MongoDatabase#
```

Now a record is inserted in the Collection.

Insert Multiple Records

You can insert multiple records in a collection by using insert()method. The insert()method uses array of objects which contain the data you want to insert.

Example

Insert multiple records in the collection named "employees".

Create a js file name insertall.js having the following Code:

```
Var MongoClient = require('mongodb').MongoClient;
Var url = "mongodb://localhost:27017/MongoDatabase";
MongoClient.connect(url, function(err, db) {
if (err) throw err;
Var myobj = [
{name:"Mahesh sharma", age:"25", address:"Ghaziabad"},
{name:"Tom Moody", age:"31", address:"CA"},
{name:"Zahira Wasim", age:"19", address:"Islamabad"}
];
db.collection("customers").insert(myobj, function(err, res) {
if (err) throw err;
console.log("Number of records inserted: " + res.insertedCount);
db.close();
});
});
```

Open the command terminal and run the following command:

Node insertall.js

```
root@aspire-pc: /home/user/Desktop/MongoDatabase
root@aspire-pc:/home/user/Desktop/MongoDatabase# node insertall.js
Number of records inserted: 4
root@aspire-pc:/home/user/Desktop/MongoDatabase#
```

You can see here 4 records are inserted.

Node.js MongoDB Select Record

The findOne() method is used to select a single data from a collection in MongoDB. This method returns the first record of the collection.

Example

(Select Single Record)

Select the first record from the "employees" collection.

Create a js file named "Select.js" having the following code:

```
Var http = require('http');
Var MongoClient = require('mongodb').MongoClient;
Var url = "mongodb://localhost:27017/MongoDatabase";
MongoClient.connect(url, function(err, db){
  if(err) throw err;
  db.Collection("employees").fineOne({}, function(err, result){
  if(err) throw err;
  Console.log(result.name);
  db.close();
  });
});
```

open the command terminal and run the following command:

Node Select.js

```
root@aspire-pc: /home/user/Desktop/MongoDatabase
root@aspire-pc:/home/user/Desktop/MongoDatabase# node remove.js
1 record(s) deleted
root@aspire-pc:/home/user/Desktop/MongoDatabase#
```

Select Multiple Records

The find() method is used to select all the records from collection in mongoDB.

Example

Select all records from "employees" collection.

Create a js file named "selectall.js", having the following code:

```
VarMongo client = require ('mongoDB'). mongoclient;
Var url = "mongodb://localhost:27017/MongoDatabase";
mongoclient.connect(url, function(err, db){
  if (err) throw err;
  db.collection("employees").find({}).toarray(function
                                        (err, result){
    if(err) throw err;
    console.log(result);
    db.close();
  });
});
```

Open the command terminal and run the following command:

```
Node selectall.js
```

Node.js MongoDB Filter Query

The find() method is also used to filter the result on a specific parameter. You can filter the result by using a query object.

Example

Filter the records to retrieve the specific employee whose address is "Delhi".

Create a .js file named "query1.js" having the following code:

```
Var http = require('http');
VarMongoClient = require('mongodb').MongoClient;
Varurl = "mongodb://localhost:27017/MongoDatabase";
MongoClient.connect(url,function(err, db){
if(err) throw err;
Varquery = {address: "Delhi"};
db.collection("employees").find(query).toArray(function
                                (err, result){
if(err) throw err;
console.log(result);
db.close();
});
});
```

Open the Command terminal and run the following Command

```
Node query1.js
```

```
root@aspire-pc: /home/user/Desktop/MongoDatabase
root@aspire-pc:/home/user/Desktop/MongoDatabase# node query1.js
[ { _id: 591040c52a89e8301bde229a,
    name: 'Ajeet Kumar',
    age: '28',
    address: 'Delhi' } ]
root@aspire-pc:/home/user/Desktop/MongoDatabase#
```

Node.js Mongo DB Sorting

In MongoDB, the sort() method is used for sorting the results in ascending or descending order. The sort() method uses a parameter to define the object sorting order.

Value used for sorting in ascending order:

```
[name:1]
```

Value used for Sorting in descending order:

```
[name:-1]
```

Sort in Ascending Order

Example

Sort the records in ascending order by the name
Create a js file named "Sortasc.js" having the following code:

```
Var http = require('http');
Var Mongo client = require('mongodb').mongo client
Var url = "mongodb://localhost:27017/MongoDatabase";
MongoClient.connect(url, function(err, db) {
if(err) throw err;
Var mysort = {name:1};
db.collection("employees").find().sort(mysort).toArray
                    (function(err, result) {
if(err) throw err;
Console.log(result);
db.close();
});
});
```

Open the Command terminal and run the following comma

```
Node sortasc.js
```

```
root@aspire-pc: /home/user/Desktop/MongoDatabase
root@aspire-pc:/home/user/Desktop/MongoDatabase# node sortasc.js
[ { _id: 591040c52a89e8301bde229a,
    name: 'Ajeet Kumar',
    age: '28',
    address: 'Delhi' },
  { _id: 59104e062f00cd3366ceb7dd,
    name: 'Juck Ross',
    age: '45',
    address: 'London' },
  { _id: 59104e062f00cd3366ceb7da,
    name: 'Mahesh Sharma',
    age: '25',
    address: 'Ghaziabad' },
  { _id: 59104e062f00cd3366ceb7db,
    name: 'Tom Moody',
    age: '31',
    address: 'CA' },
  { _id: 59104e062f00cd3366ceb7dc,
    name: 'Zahira Wasim',
    age: '19',
    address: 'Islamabad' } ]
root@aspire-pc:/home/user/Desktop/MongoDatabase#
```

Node.js Mongo DB Remove

In Mongo DB, you can delete records or documents by using the remove() method. The first parameter of the remove() method is a query object which specifies the document to delete.

Example

Remove the record of employee whose address is Ghaziabad.

Create a js file named "remove.js", having the following code:

```
Var http = require('http');
Var MongoClient=require('mongodb').Mongo client;
Var url = "mongodb://localhost:27017/MongoDatabase";
MongoClient.connect(url, function (err, db){
if (err) throw err;
Var myquery = {address: 'Ghaziabad'};
db.Collection("employees").remove(myquery, function(err, obj){
if (err) throw err;
Console.log(obj.result.n +"record(s) deleted");
db.close();
});
});
```

Open the command terminal and run the following command:

```
Node remove.js
```