JUST UNIVERSITY

Ch05
DATABASE (MONGODB&MYSQL)

Eng. Abdirizak Ibrahim Abdiwahid (Eng.Qeys)

JAMHURIYA UNIVERSITY

Department of Computer Science

WHAT IS DATABASE

- A database is **a systematic collection of data**.

They support electronic storage and manipulation of data. Databases make data management easy.

Also used for **storing, maintaining and accessing any sort of data**. They collect information on people, places or things. That information is gathered in one place so that it can be observed and analyzed. Databases can be thought of as an organized collection of information.

TYPES OF DATABASE

Here are some popular types of databases.

- Hierarchical database systems
- Cloud databases
- NoSQL databases
- Document/JSON database

Cloud database

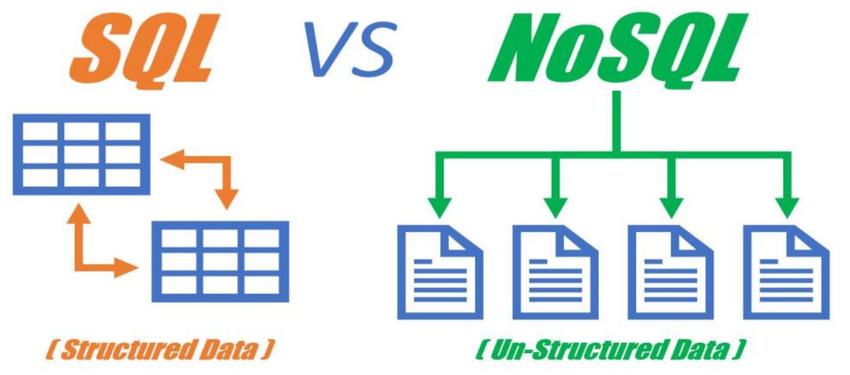
Cloud database

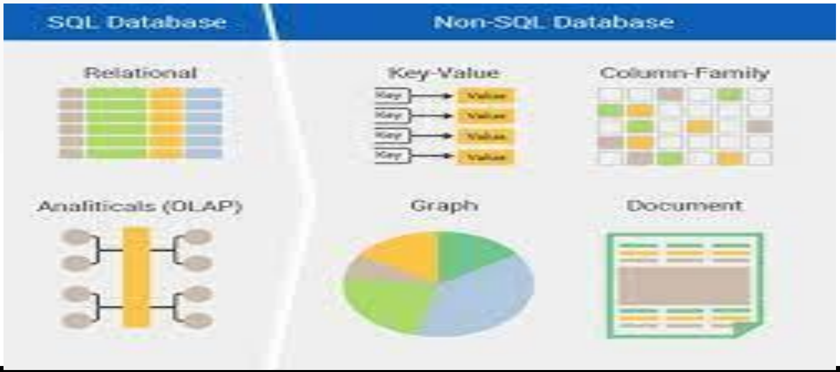
- A cloud database is one that runs over the Internet. The data is stored on a local hard drive or server, but the information is available online. This makes it easy to access your files from anywhere, as long as you have an Internet connection. To use a cloud database, users can either build one themselves or pay for a service to store their data for them. Encryption is an essential part of any cloud database, as all information needs to be protected as it is transmitted online.

SQL VS NOSQL

- SQL databases are vertically scalable, while NoSQL databases are horizontally scalable. SQL databases are table-based, while NoSQL databases are document, key-value, graph, or wide-column stores. SQL databases are better for multi-row transactions, while NoSQL is better for unstructured data like documents or JSON.

	SQL	NoSQL
Database Type	Relational Databases	Non-relational Databases / Distributed Databases
Structure	Table-based	• Key-value pairs • Document-based • Graph databases • Wide-column stores
Scalability	Designed for scaling up vertically by upgrading one expensive custom-built hardware	Designed for scaling out horizontally by using shards to distribute load across multiple commodity (inexpensive) hardware
Strength	• Great for highly structured data and don't anticipate changes to the database structure • Working with complex queries and reports	• Pairs well with fast paced, agile development teams • Data consistency and integrity is not top priority • Expecting high transaction load





MongoDB

- MongoDB is **an open source NoSQL database management program**. NoSQL is used as an alternative to traditional relational databases. NoSQL databases are quite useful for working with large sets of distributed data. MongoDB is a tool that can manage document-oriented information, store or retrieve information.

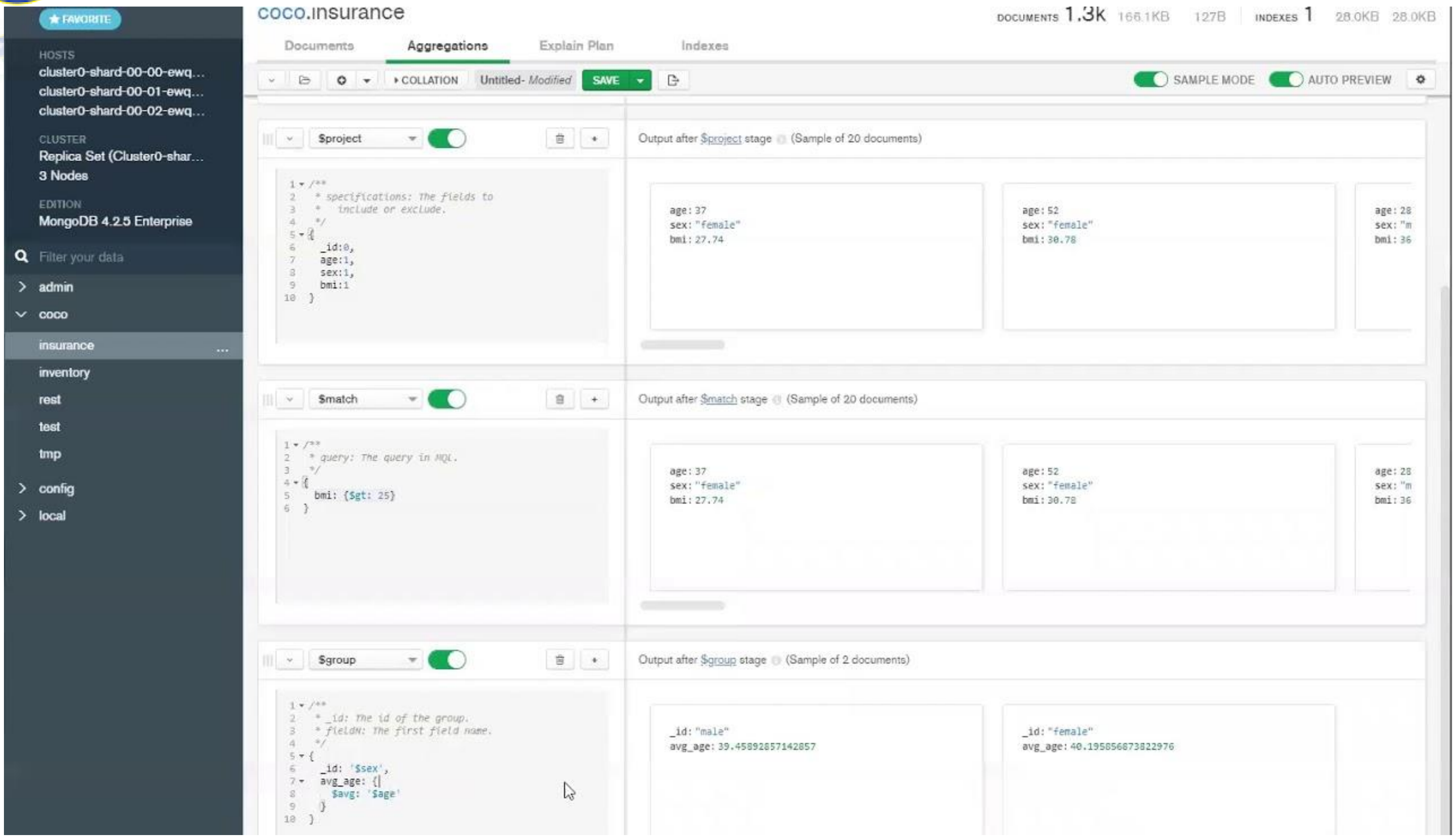
Connection:

- MONGOOSE: Mongoose is a JavaScript object-oriented programming library that creates a connection between MongoDB and the Node.js JavaScript runtime environment

```
npm install mongodb
```

MONGODB

- MongoDB shell is **an interactive JavaScript interface to MongoDB**. You can use the mongo shell to query and update data as well as perform administrative operations.
- MongoDB Compass is **a powerful GUI for querying, aggregating, and analyzing your MongoDB data in a visual environment**. Compass is free to use and source available, and can be run on macOS, Windows, and Linux.



$project

```
/**
 * specifications: The fields to
 *   include or exclude.
 */
{
  _id:0,
  age:1,
  sex:1,
  bmi:1
}
```

Output after $project stage (Sample of 20 documents)

```
age: 37
sex: "female"
bmi: 27.74
```

```
age: 52
sex: "female"
bmi: 30.78
```

```
age: 28
sex: "m
bmi: 36
```

$match

```
/**
 * query: The query in MQL.
 */
{
  bmi: {$gt: 25}
}
```

Output after $match stage (Sample of 20 documents)

```
age: 37
sex: "female"
bmi: 27.74
```

```
age: 52
sex: "female"
bmi: 30.78
```

```
age: 28
sex: "m
bmi: 36
```

$group

```
/**
 * _id: The id of the group.
 * fieldN: The first field name.
 */
{
  _id: '$sex',
  avg_age: {
    $avg: '$age'
  }
}
```

Output after $group stage (Sample of 2 documents)

```
_id: "male"
avg_age: 39.45892857142857
```

```
_id: "female"
avg_age: 40.195856873822976
```

USING MONGODB SHELL

Commands :

```
//creating database
  Use school
// showing databases created
   Show dbs
// dropping Database
db.dropDatabase()
//creating collections
  db.createCollection('ardeyda')
//showing collections created
  Show collections
```

Continue..

- // inserting data into collections

db.ardeyda.insertOne({“name”:”abdi alsow”})

// updating Data into collection

db.ardeyda.updateOne({name:”ali”},{$set:{“tell”:900}})

// Deleting Data into collection

db.ardeyda.deleteOne({name:”ali”})

Showing data into collections

db.ardeyda.find()

- // dropping collections

db.ardeyda.drop()

BASIC CONNECTING MONGODB TO NODE JS

```
• const {MongoClient}= require('mongodb');
• const url = "mongodb://localhost:27017/";
• const database = 'just';
• const client = new MongoClient(url);
•
  async function createdbs(err){
•     if(!err){
•         console.log('connected');
•     }
•     else{
•         console.log('err');
•     }
• }
• createdbs();
```

Creating Database MongoDB using Node js

-
  ```
  var MongoClient =
  require('mongodb').MongoClient;
  var url
  = "mongodb://localhost:27017/wadan";

  MongoClient.connect(url, function(err,
  db) {
    if (err) throw err;
    console.log("Database created!");
    db.close();
  });
  ```

Creating Collection MongoDB using Node js

-
  ```
  var MongoClient =
  require('mongodb').MongoClient;
  var url = "mongodb://localhost:27017/";

  MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    var dbo = db.db(“wadan");
    dbo.createcollection(“Students", function(
  err, res) {
      if (err) throw err;
      console.log("Collection created!");
  ```

-
  ```
      db.close();
    });
  });
  ```

Insert Into Collection

-
  ```
  var MongoClient = require('mongodb').MongoClient;
  var url = "mongodb://localhost:27017/";

  MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    var dbo = db.db("mydb");
    var myobj = { name: "Company Inc",
  address: "Highway 37" };
    dbo.collection("customers").insertOne(myobj, functi
  on(err, res) {
      if (err) throw err;
      console.log("1 document inserted");
      db.close();
    });
  });
  ```

Fetch data into MongoDB step 1

```
• var {MongoClient} = require('mongodb')
• var url = 'mongodb://localhost:27017/';
• var database = ‘just';

MongoClient.connect(url,function(err,db){
•     if(err) throw err;
•    var dbo = db.db(database);
•    dbo.collection(‘jamhr').find({}).toArray(f
  unction(err,rsl){
•     if(err) throw err;
•     console.log(rsl);
•    })
•
•    });
•
```

FETCH DATA INTO DATABASE USING MONGODB

```
• const {MongoClient} = require('mongodb')
• const url = 'mongodb://localhost:27017/';
• const database = 'just';
• const client = new MongoClient(url);
•
  async function creatingdb(){
•     let result = await client.connect();
•     let db = result.db(database);
•     let collection = db.collection('jamhr');
•     let response
  =await  collection.find({}).toArray();
•     console.log(response);
• }
• creatingdb();
```

Reading Data from Database

```
• const {MongoClient} = require('mongodb')
• const url = 'mongodb://localhost:27017/';
• const database = 'just';
• const client = new MongoClient(url);
•
  async function xogta(){
•     let result = await client.connect();
•     let db = result.db(database);
•     let collection = db.collection('jamhr');
•     let response =await  collection.find({name:'ali
  ahmed'}).toArray();
•     console.log(response);
• }
• xogta();
```

Delete into Collection using node js

```
• var {MongoClient} = require('mongodb')
• var url = 'mongodb://localhost:27017/';
• var database = ‘just';
•
  MongoClient.connect(url,function(err,db){
•     if(err) throw err;
•    var dbo = db.db(database);
•    var tir = {name:"asad"};
•    dbo.collection('std').deleteOne(tir,function(err){
•     if(err) throw err;
•     console.log('deleted one record');
•    });
•
•    });
•
```

Dropping collection using node

```
• var {MongoClient} = require('mongodb')
• var url = 'mongodb://localhost:27017/';
• var database = 'c192';
•
  MongoClient.connect(url,function(err,db){
•    if(err) throw err;
•   var dbo = db.db(database);
•   dbo.collection('kalas').drop(function(err){
•    if(err) throw err;
•    console.log('droped table ');
•    db.close();
•   });
•
•   });
```

Node.js MongoDB Update

```
• var MongoClient =
  require('mongodb').MongoClient;
  var url = "mongodb://127.0.0.1:27017/";

  MongoClient.connect(url, function(err, db) {
     if (err) throw err;
     var dbo = db.db("mydb");
     var myquery = { xafada: “kpp" };
     var newvalues = { $set: {name: “asad",
  xafada: “Madina" } };
     dbo.collection("customers").updateOne(myquery
  , newvalues, function(err, res) {
       if (err) throw err;
       console.log("1 document updated");
       db.close();
     });
  });
```

CONNECTING NODE JS TO MONGODB CLOUD

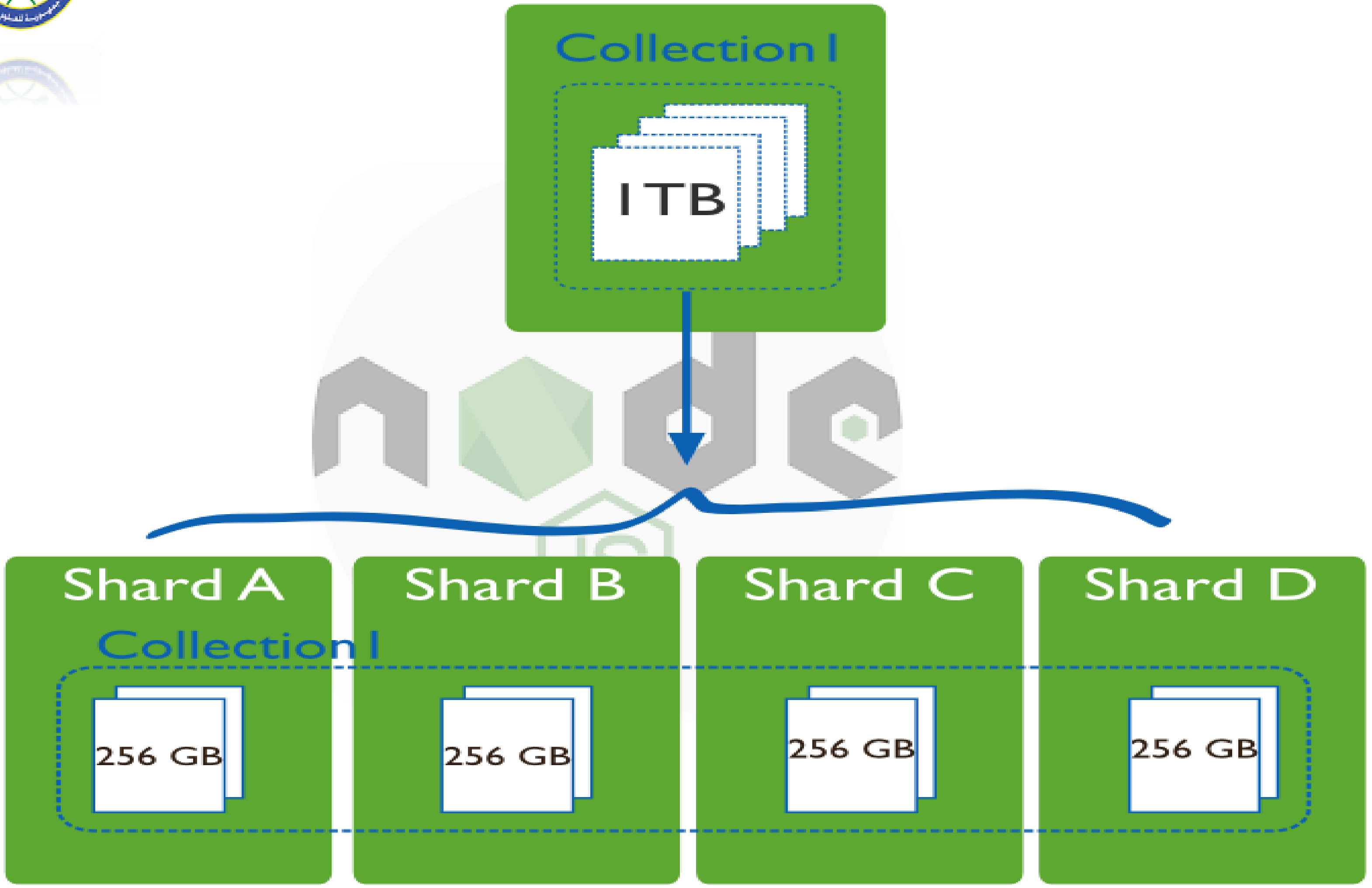
```
• const express = require('express')
• const app = express();
• const mongoose = require('mongoose');
• var url =
  'mongodb+srv://qe:j1234@qeys.vsbepjg.mongodb.net
  /?retryWrites=true&w=majority';
• mongoose.connect(url);
• async function conn(err){
•    if(err) throw err;
•    console.log('connected')
• }
• conn();
• app.listen(3000);
```

CLUSTER

A mongodb cluster is **the word usually used for sharded cluster in mongodb**. The main purposes of a sharded mongodb are: Scale reads and writes along several nodes. Each node does not handle the whole data so you can separate data along all the nodes of the shard.

A sharded cluster in MongoDB is a collection of datasets distributed across many shards (servers) in order to achieve horizontal scalability and better performance in read and write operations.

Cluster Shard



Continue….

- Sharding reduces the number of operations each shard handles. Each shard processes fewer operations as the cluster grows. As a result, a cluster can increase capacity and throughput *horizontally*.
- For example, to insert data, the application only needs to access the shard responsible for that record.
- Sharding reduces the amount of data that each server needs to store. Each shard stores less data as the cluster grows.
- For example, if a database has a 1 terabyte data set, and there are 4 shards, then each shard might hold only 256 GB of data. If there are 40 shards, then each shard might hold only 25 GB of data.

Promise

- Promise : Promises are used **to handle asynchronous operations in JavaScript**. They are easy to manage when dealing with multiple asynchronous operations where callbacks can create callback hell leading to unmanageable code
- **Benefits of Promises**
  - Improves Code Readability
  - Better handling of asynchronous operations
  - Better flow of control definition in asynchronous logic
  - Better Error Handling

MySQL

- One of the most popular databases is MySQL.

```
npm install mysql
```

BASIC CONNECTING MYSQL TO NODE JS

```
var mysql = require('mysql');
var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword“
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
});
```

Creating a Database using node.js

-
  ```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  con.query("CREATE DATABASE
mydb", function (err, result) {
    if (err) throw err;
    console.log("Database created");
  });
});
```

Creating a Table using node js

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "CREATE TABLE customers (name
VARCHAR(255), address VARCHAR(255))";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Table created");
  });
});
```

Insert Into Table

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "INSERT INTO customers (name, 
address) VALUES ('Company Inc', 'Highway 37')";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("1 record inserted");
  });
});
```

Fetching data into mysql to node js

```
ar mysql = require('mysql');


var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM customers", function (err, result, fields) {
    if (err) throw err;
    console.log(result);
  });
});
```

Deleting Mysql Data using node js

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  var sql = "DELETE FROM customers WHERE address = 
'Mountain 21'";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Number of records deleted: " + 
result.affectedRows);
  });
});
```

Drop table mysql using Node js

```
• var mysql = require('mysql');

  var con = mysql.createConnection({
    host: "localhost",
    user: "yourusername",
    password: "yourpassword",
    database: "mydb"
  });

  con.connect(function(err) {
    if (err) throw err;
    var sql = "DROP TABLE customers";
    con.query(sql, function (err, result) {
      if (err) throw err;
      console.log("Table deleted");
    });
  });
```

Update Table mysql DB using nod js

```
• var mysql = require('mysql');

  var con = mysql.createConnection({
    host: "localhost",
    user: "yourusername",
    password: "yourpassword",
    database: "mydb"
  });

  con.connect(function(err) {
    if (err) throw err;
    var sql = "UPDATE customers SET address =
  'Canyon 123' WHERE address = 'Valley 345'";
    con.query(sql, function (err, result) {
      if (err) throw err;
      console.log(result.affectedRows + "
  record(s) updated");
    });
  });
```

END