Unit 9: Node JS with MySQL

Node.Js Create Connection with MySQL

We can use Node.js in database applications. Here we use MySQL as a database with Node.js.

Install MySQL on your computer.

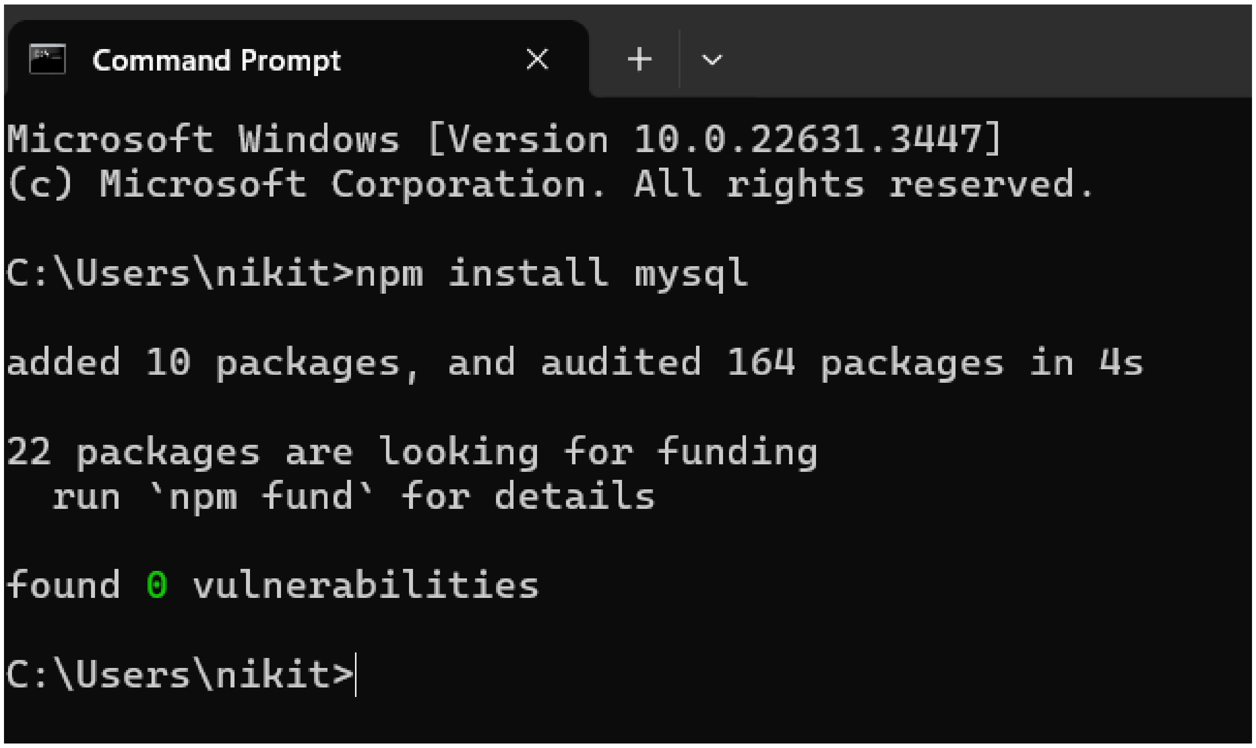
You can download it from here https://www.mysql.com/downloads/.

Once the MySQL is installed and running, you can access it by using Node.js.

Install MySQL Driver

You have to install MySQL driver to access a MySQL database with Node.js. Download MySQl module from npm.

To download and install the "mysql" module, open the Command Terminal and execute the following: npm install mysql

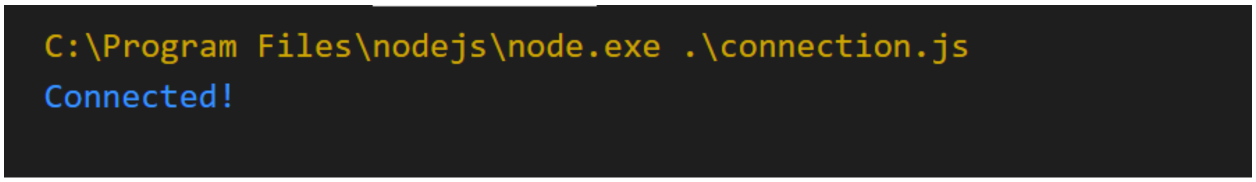


```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nikit>npm install mysql

added 10 packages, and audited 164 packages in 4s

22 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\nikit>
```

Create Connection

Create a folder named "DBexample". In that folder create a js file named "connection.js" having the following code:

```
var mysql = require('mysql');
var con = mysql.createConnection({
  host: "localhost",
  port:"3306",
  user: "root",
  password: "root"
});
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
});
```



```
C:\Program Files\nodejs\node.exe .\connection.js
Connected!
```

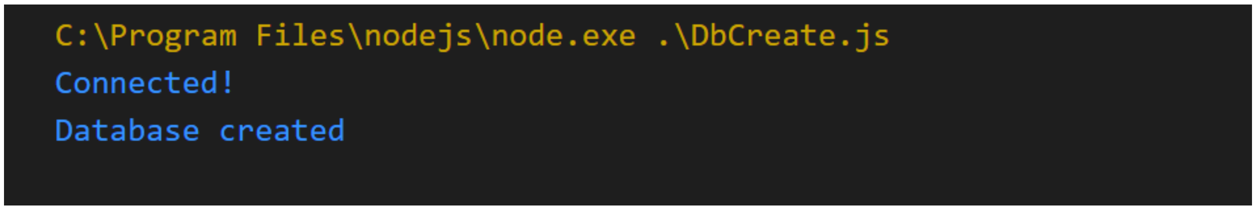
Node.js MySQL Create Database

CREATE DATABASE statement is used to create a database in MySQL.

Example

For creating a database named "Student".

```
var mysql = require('mysql');
var con = mysql.createConnection({
host: "localhost",
user: "root",
password: "root"
});
con.connect(function(err) {
if (err) throw err;
console.log("Connected!");
con.query("CREATE DATABASE Employee", function (err, result) {
if (err) throw err;
console.log("Database created");
});
});
```



```
C:\Program Files\nodejs\node.exe .\DbCreate.js
Connected!
Database created
```

Node.js MySQL Create Table

CREATE TABLE command is used to create a table in MySQL. You must make it sure that you define the name of the database when you create the connection.

Example

For creating a table named "employees".

```
var mysql = require('mysql');
var con = mysql.createConnection({
host: "localhost",
user: "root",
password: "root",
database: "Employee"
});
con.connect(function(err) {
if (err) throw err;
console.log("Connected!");
var sql = "CREATE TABLE employees (id INT, name VARCHAR(255), age INT(3), city
VARCHAR(255))";
con.query(sql, function (err, result) {
if (err) throw err;
console.log("Table created");
});
});
```

Create Table Having a Primary Key

Create Primary Key in New Table:

Let's create a new table named "employee2" having id as primary key.

Create a js file named employee2.js having the following data in DBexample folder.

```
var mysql = require('mysql');
var con = mysql.createConnection({
host: "localhost",
user: "root",
password: "root",
database: "Employee"
});
con.connect(function(err) {
if (err) throw err;
console.log("Connected!");
var sql = "CREATE TABLE employee2 (id INT PRIMARY KEY, name VARCHAR(255), age
INT(3), city VARCHAR(255))";
con.query(sql, function (err, result) {
if (err) throw err;
console.log("Table created");
});
});
```

```
C:\Program Files\nodejs\node.exe .\empoyee2.js
Connected!
Table created
```

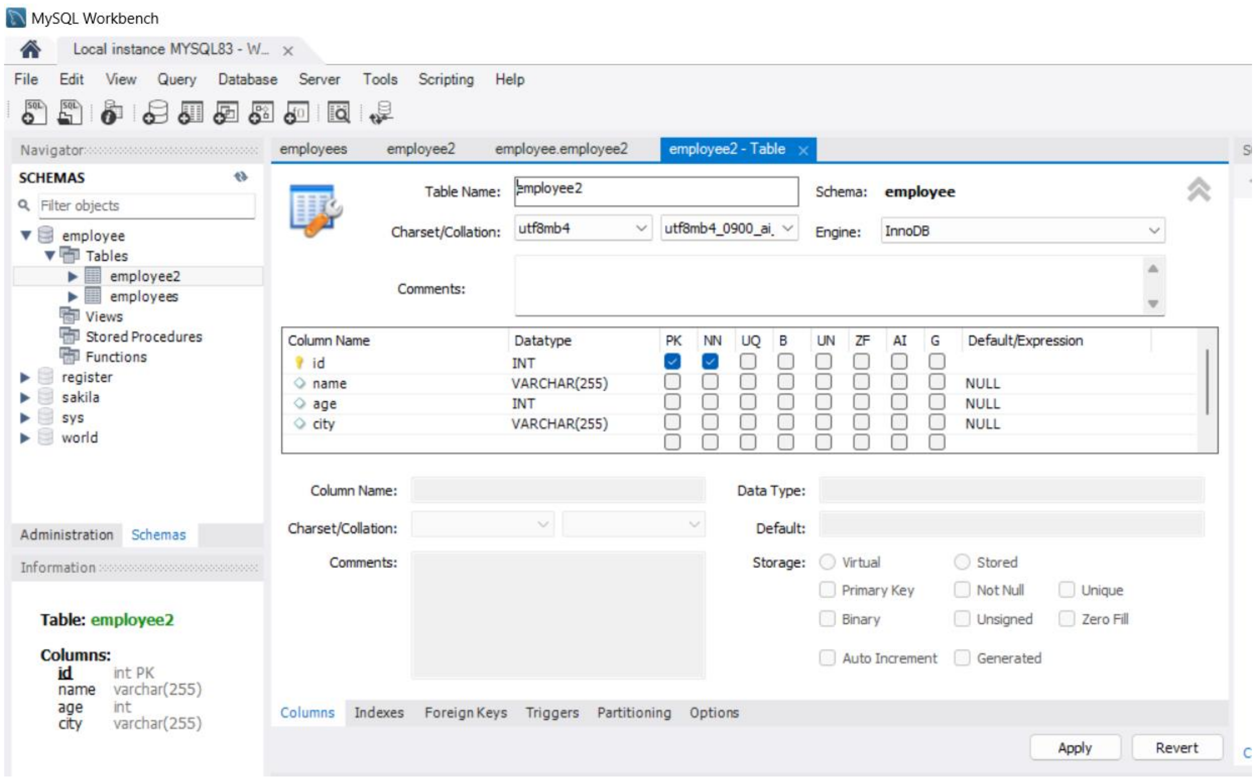

MySQL Workbench
Local instance MYSQL83 - W...
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
employee
Tables
employee2
employees
Views
Stored Procedures
Functions
register
sakila
sys
world
Administration Schemas
Information
employees employee2 employee.employee2 employee2 - Table
Columns Indexes Foreign Keys Triggers Partitioning Options
Apply Revert

Table Name: employee2 Schema: employee

Charset/Collation: utf8mb4 utf8mb4_0900_ai_ Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	✓	✓							
name	VARCHAR(255)									NULL
age	INT									NULL
city	VARCHAR(255)									NULL

Column Name: Data Type:

Charset/Collation: Default:

Comments: Storage: Virtual Stored Primary Key Not Null Unique Binary Unsigned Zero Fill Auto Increment Generated

Table: employee2

Columns:

id	int PK
name	varchar(255)
age	int
city	varchar(255)

Add columns in existing Table:

ALTER TABLE statement is used to add a column in an existing table. Take the already created table "employee2" and use a new column salary.

Replace the data of the "employee2" table with the following data:

```
var mysql = require('mysql');
var con = mysql.createConnection({
host: "localhost",
user: "root",
password: "root",
database: "Employee"
});
con.connect(function(err) {
if (err) throw err;
console.log("Connected!");
var sql = "ALTER TABLE employee2 ADD COLUMN salary INT(10)";
con.query(sql, function (err, result) {
if (err) throw err;
console.log("Table altered");
});
});
```

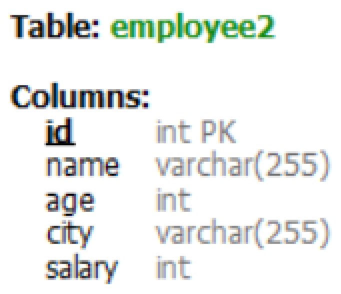


Table: employee2

Columns:

id	int PK
name	varchar(255)
age	int
city	varchar(255)
salary	int

Node.js MySQL Insert Records

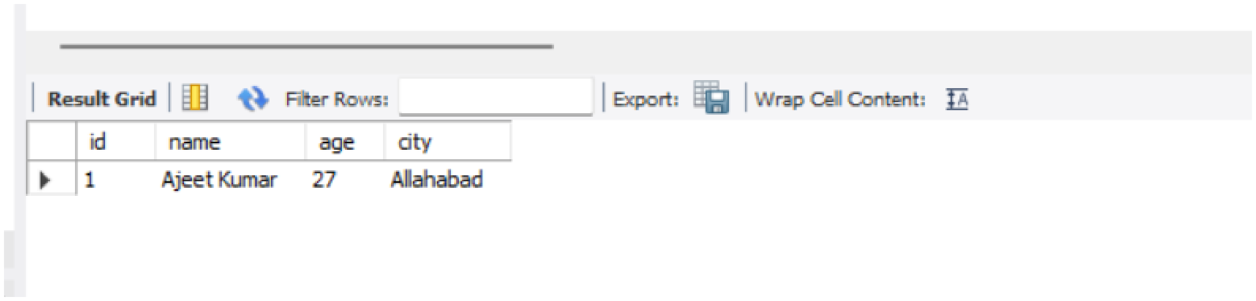
INSERT INTO statement is used to insert records in MySQL.

Example

Insert Single Record:

Insert records in "employees" table.

```
var mysql = require('mysql');
var con = mysql.createConnection({
host: "localhost",
user: "root",
password: "root",
database: "Employee"
});
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");
});
});
```

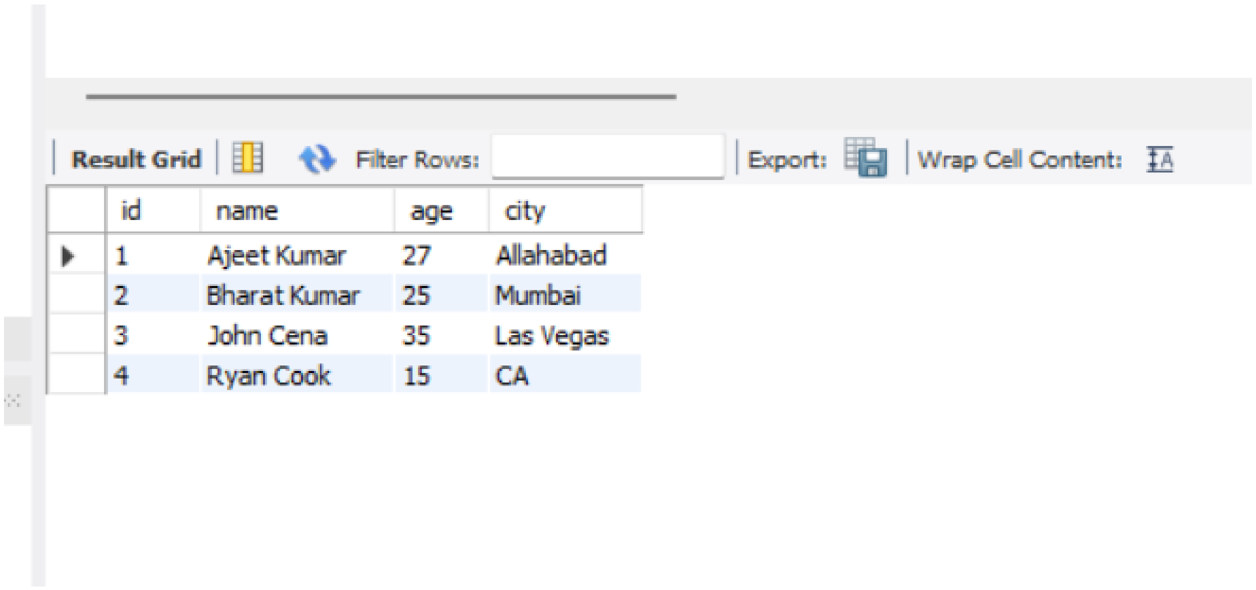


	id	name	age	city
▶	1	Ajeet Kumar	27	Allahabad

Insert Multiple Records

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

```
var mysql = require('mysql');
var con = mysql.createConnection({
host: "localhost",
user: "root",
password: "root",
database: "Employee"
});
con.connect(function(err) {
if (err) throw err;
console.log("Connected!");
var sql = "INSERT INTO employees (id, name, age, city) VALUES ?";
var values = [
['2', 'Bharat Kumar', '25', 'Mumbai'],
['3', 'John Cena', '35', 'Las Vegas'],
['4', 'Ryan Cook', '15', 'CA']
];
con.query(sql, [values], function (err, result) {
if (err) throw err;
console.log("Number of records inserted: " + result.affectedRows);
});
});
```



id	name	age	city
1	Ajeet Kumar	27	Allahabad
2	Bharat Kumar	25	Mumbai
3	John Cena	35	Las Vegas
4	Ryan Cook	15	CA

Node.js MySQL Update Records

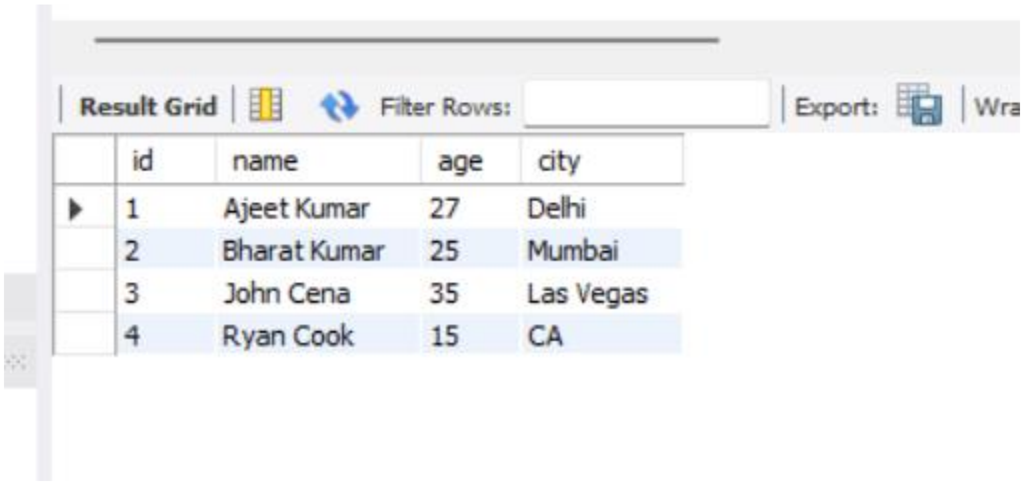
The UPDATE command is used to update records in the table.

Example

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

Create a js 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: "root",
database: "Employee"
});
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");
});
});
```



	id	name	age	city
▶	1	Ajeet Kumar	27	Delhi
	2	Bharat Kumar	25	Mumbai
	3	John Cena	35	Las Vegas
	4	Ryan Cook	15	CA

Node.js MySQL Delete Records

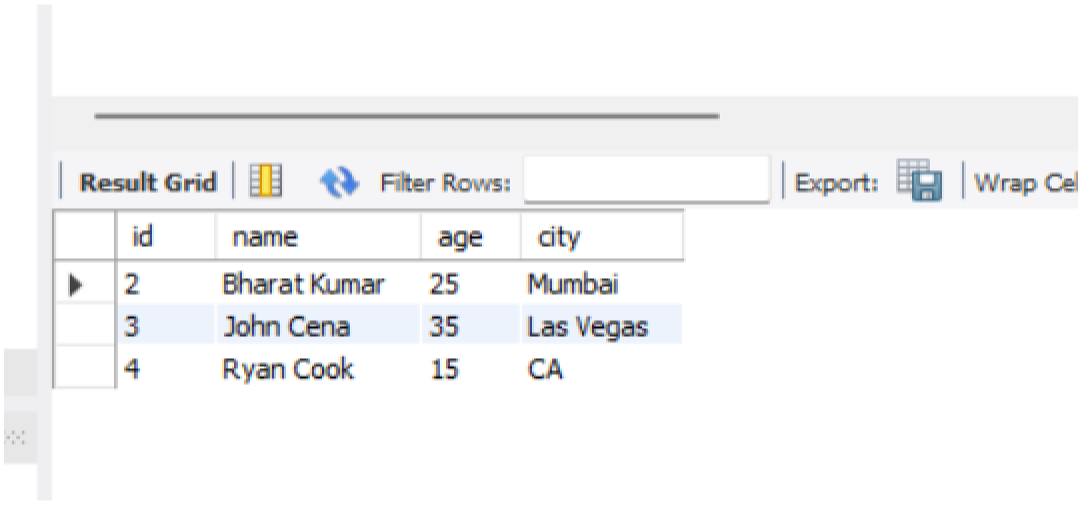
The DELETE FROM command is used to delete records from the table.

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 mysql = require('mysql');
var con = mysql.createConnection({
host: "localhost",
user: "root",
password: "root",
database: "Employee"
});
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);
});
});
```



	id	name	age	city
▶	2	Bharat Kumar	25	Mumbai
	3	John Cena	35	Las Vegas
	4	Ryan Cook	15	CA

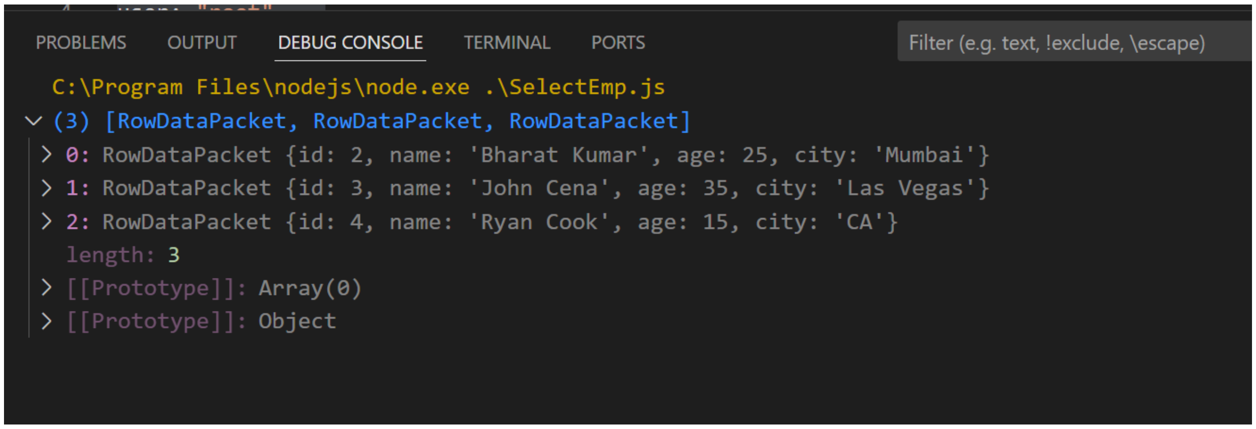
Node.js MySQL Select Records

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: "root",
database: "Employee"
});
con.connect(function(err) {
if (err) throw err;
con.query("SELECT * FROM employees", function (err, result) {
if (err) throw err;
console.log(result);
});
});
```



```
C:\Program Files\nodejs\node.exe .\SelectEmp.js
(3) [RowDataPacket, RowDataPacket, RowDataPacket]
  0: RowDataPacket {id: 2, name: 'Bharat Kumar', age: 25, city: 'Mumbai'}
  1: RowDataPacket {id: 3, name: 'John Cena', age: 35, city: 'Las Vegas'}
  2: RowDataPacket {id: 4, name: 'Ryan Cook', age: 15, city: 'CA'}
  length: 3
  [[Prototype]]: Array(0)
  [[Prototype]]: Object
```

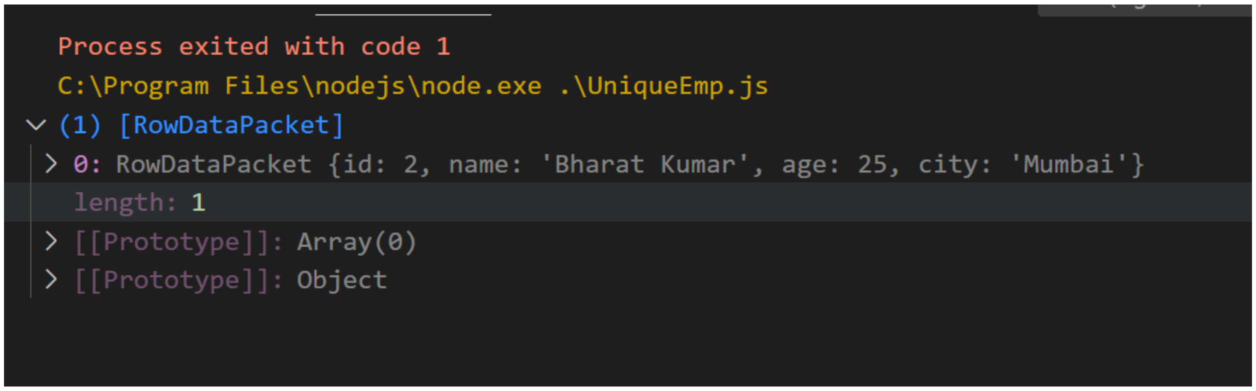
Node.js MySQL SELECT Unique Record

(WHERE Clause)

Retrieve a unique data from the table "employees".

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: "root",
database: "Employee"
});
con.connect(function(err) {
if (err) throw err;
con.query("SELECT * FROM employees WHERE id = '2'", function (err, result) {
if (err) throw err;
console.log(result);
});
});
```

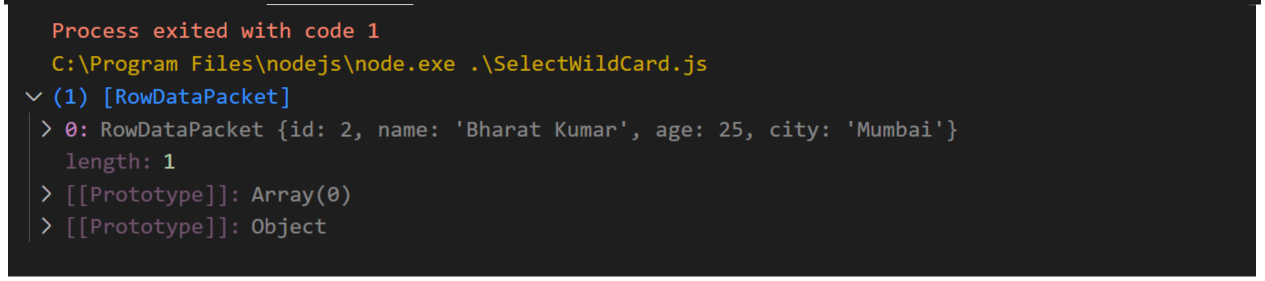


```
Process exited with code 1
C:\Program Files\nodejs\node.exe .\UniqueEmp.js
∨ (1) [RowDataPacket]
  > 0: RowDataPacket {id: 2, name: 'Bharat Kumar', age: 25, city: 'Mumbai'}
    length: 1
  > [[Prototype]]: Array(0)
  > [[Prototype]]: Object
```

Node.js MySQL Select Wildcard

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

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



```
Process exited with code 1
C:\Program Files\nodejs\node.exe .\SelectWildCard.js
∨ (1) [RowDataPacket]
  > 0: RowDataPacket {id: 2, name: 'Bharat Kumar', age: 25, city: 'Mumbai'}
    length: 1
  > [[Prototype]]: Array(0)
  > [[Prototype]]: Object
```

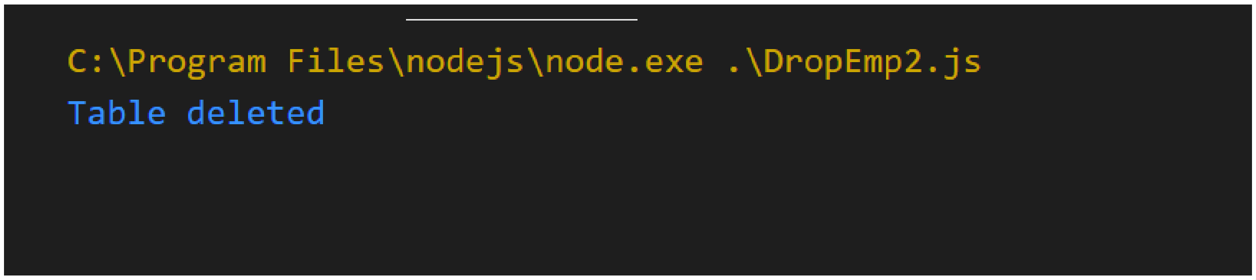
Node.js MySQL Drop Table

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

Let's 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: "root",
database: "Employee"
});
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");
});
});
```



```
C:\Program Files\nodejs\node.exe .\DropEmp2.js
Table deleted
```