1.Create connection:

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

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

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

Output:

"C:\Program Files\nodejs\node.exe"
C:\Users\Azizul56\IdeaProjects\nodejsquery\demo_db_connections.js

Connected!

2.Create Database:

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

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

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");
    });
});
```

Output:

A table called mydb is created.

3.Create Table:

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

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

Output
A table called customers is created

4.Inserting a new row:

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

var con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "",
    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 row inserted");
    });
});
```

Output:
one new row is added in the database.

5.Selecting for query:

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

var con = mysql.createConnection({
```

```
    host: "localhost",
    user: "root",
    password: "",
    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);
    });
});
```

Output:

"C:\Program Files\nodejs\node.exe"
C:\Users\Azizul56\IdeaProjects\nodejs_practise\nodejsquery\demo_db_connections.js

[ RowDataPacket { name: 'Company Inc', address: 'Highway 37' },

  RowDataPacket { name: 'Company Inc', address: 'Highway 37' },

  RowDataPacket { name: 'Company Inc', address: 'Highway 37' } ]

6.Field:

The third parameter of the callback function is an array containing information about each field in the result.

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

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

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

Output:

```
"C:\Program Files\nodejs\node.exe"
C:\Users\Azizul56\IdeaProjects\nodejs_practise\nodejsquery\demo_db_connections.js
[ FieldPacket {
    catalog: 'def',
    db: 'mydb',
    table: 'customers',
    orgTable: 'customers',
    name: 'name',
    orgName: 'name',
    charsetNr: 33,
    length: 765,
    type: 253,
    flags: 0,
    decimals: 0,
    default: undefined,
    zeroFill: false,
    protocol41: true },
  FieldPacket {
    catalog: 'def',
    db: 'mydb',
    table: 'customers',
    orgTable: 'customers',
    name: 'address',
    orgName: 'address',
    charsetNr: 33,
```

length: 765,

type: 253,

flags: 0,

decimals: 0,

default: undefined,

zeroFill: false,

protocol41: true } ]

7.Where operation:

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

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

con.connect(function(err) {
    if (err) throw err;
    con.query("SELECT * FROM customers WHERE address LIKE 'H%'", function (err,
result) {
        if (err) throw err;
        console.log(result);
    });
});
```

Output:

"C:\Program Files\nodejs\node.exe"
C:\Users\Azizul56\IdeaProjects\nodejs_practise\nodejsquery\demo_db_connections.js

[ RowDataPacket { name: 'Company Inc', address: 'Highway 37' },

RowDataPacket { name: 'Company Inc', address: 'Highway 37' },

RowDataPacket { name: 'Company Inc', address: 'Highway 37' } ]

**8.** Escaping Query Values:

When query values are variables provided by the user, you should escape the values.

This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database.

The MySQL module has methods to escape query values.

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

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

var adr = 'Highway 37';
var sql = 'SELECT * FROM customers WHERE address = ?';
con.query(sql,  [adr],function (err, result) {
    if (err) throw err;
    console.log(result);
});
```

Output:

"C:\Program Files\nodejs\node.exe"
C:\Users\Azizul56\IdeaProjects\nodejs_practise\nodejsquery\demo_db_connections.js

[ RowDataPacket { name: 'Company Inc', address: 'Highway 37' },

  RowDataPacket { name: 'Company Inc', address: 'Highway 37' },

  RowDataPacket { name: 'Company Inc', address: 'Highway 37' } ]

9.Order by operation:

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

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

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

Output:
"C:\Program Files\nodejs\node.exe"
C:\Users\Azizul56\IdeaProjects\nodejs_practise\nodejsquery\demo_db_connections.js

[ RowDataPacket { name: 'Company Inc', address: 'Highway 37' },

  RowDataPacket { name: 'Company Inc', address: 'Highway 37' },

  RowDataPacket { name: 'Company Inc', address: 'Highway 37' } ]

10.Delete operation:

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

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

con.connect(function(err) {
    if (err) throw err;
    var sql = "DELETE FROM customers WHERE address = 'Highway 37'";
    con.query(sql, function (err, result) {
        if (err) throw err;
        console.log("Number of records deleted: " + result.affectedRows);
    });
});
```

Output:

"C:\Program Files\nodejs\node.exe"
C:\Users\Azizul56\IdeaProjects\nodejs_practise\nodejsquery\demo_db_connections.js

Number of records deleted: 3

11.Drop 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;
    var sql = "DROP TABLE IF EXISTS customers";
    con.query(sql, function (err, result) {
        if (err) throw err;
        console.log(result);
    });
});
```

Output:
If the table exist, the result object will look like this:

```
{
  fieldCount: 0,
  affectedRows: 0,
  insertId: 0,
  serverstatus: 2,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0
}
```

If the table does not exist, the result object will look like this:

```
{
  fieldCount: 0,
  affectedRows: 0,
  insertId: 0,
  serverstatus: 2,
  warningCount: 1,
  message: '',
  protocol41: true,
  changedRows: 0
}
```

As you can see the only differnce is that the warningCount property is set to 1 if the table does not exist.

12.Update 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;
  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");
  });
});
```

**Output:**

```
Run "demo_db_update.js"

C:\Users\Your Name>node demo_db_update.js

1 record(s) updated
```

The Result Object

The result object contains information about how the query affected the table.

The result object returned from the example above looks like this:

```
{
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 34,
  warningCount: 0,
  message: '(Rows matched: 1 Changed: 1 Warnings: 0',
  protocol41: true,
  changedRows: 1
}
```

The values of the properties can be displayed like this:

```
Example

Return the number of affected rows:

console.log(result.affectedRows)
```

Which will produce this result:

```
1
```

13.Limit operation:

```
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 = "SELECT * FROM customers LIMIT 5";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

Output:

Save the code above in a file called "demo_db_limit.js" and run the file:

```
Run "demo_db_limit.js"

C:\Users\Your Name>node demo_db_limit.js
```

Which will give you this result:

```
[
  { id: 1, name: 'John', address: 'Highway 71'},
  { id: 2, name: 'Peter', address: 'Lowstreet 4'},
  { id: 3, name: 'Amy', address: 'Apple st 652'},
  { id: 4, name: 'Hannah', address: 'Mountain 21'},
  { id: 5, name: 'Michael', address: 'Valley 345'}
]
```

Start From Another Position

If you want to return five records, starting from the third record, you can use the "OFFSET" keyword:

Example

Start from position 3, and return the next 5 records:

```
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 = "SELECT * FROM customers LIMIT 5 OFFSET 2";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

Run example »

**Note:** "OFFSET 2", means starting from the third position, not the second!

Save the code above in a file called "demo_db_offset.js" and run the file:

Output:

```
Run "demo_db_offset.js"

C:\Users\Your Name>node demo_db_offset.js
```

Which will give you this result:

```
[
  { id: 3, name: 'Amy', address: 'Apple st 652'},
  { id: 4, name: 'Hannah', address: 'Mountain 21'},
  { id: 5, name: 'Michael', address: 'Valley 345'},
  { id: 6, name: 'Sandy', address: 'Ocean blvd 2'},
  { id: 7, name: 'Betty', address: 'Green Grass 1'}
]
```

Shorter Syntax

You can also use write your SQL statement like this "LIMIT 2, 5" which returns the same as the offset example above:

Example:**Start from position 3, and return the next 5 records:**

```
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 = "SELECT * FROM customers LIMIT 2, 5";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

Output:

Same as the last one.

**14.** Join Two or More Tables

You can combine rows from two or more tables, based on a related column between them, by using a JOIN statement.

Consider you have a "users" table and a "products" table:

```
users

[
  { id: 1, name: 'John', favorite_product: 154},
  { id: 2, name: 'Peter', favorite_product: 154},
  { id: 3, name: 'Amy', favorite_product: 155},
  { id: 4, name: 'Hannah', favorite_product:},
  { id: 5, name: 'Michael', favorite_product:}
]
```

products

```
[
  { id: 154, name: 'Chocolate Heaven' },
  { id: 155, name: 'Tasty Lemons' },
  { id: 156, name: 'Vanilla Dreams' }
]
```

These two tables can be combined by using users' `favorite_product` field and products' `id` field.

Example

Select records with a match in both tables:

```
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 = "SELECT users.name AS user, products.name AS favorite FROM users JOIN products ON users.favorite_product = products.id";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

Run example »

**Note:** You can use INNER JOIN instead of JOIN. They will both give you the same result.

Save the code above in a file called "demo_db_join.js" and run the file:

Run "demo_db_join.js"

```
C:\Users\Your Name>node demo_db_join.js
```

Which will give you this result:

```
[
  { user: 'John', favorite: 'Chocolate Heaven' },
  { user: 'Peter', favorite: 'Chocolate Heaven' },
  { user: 'Amy', favorite: 'Tasty Lemons' }
]
```

As you can see from the result above, only the records with a match in both tables are returned.

---

Left Join

If you want to return *all* users, no matter if they have a favorite product or not, use the LEFT JOIN statement:

Example

Select all users and their favorite product:

```
SELECT users.name AS user,
products.name AS favorite
FROM users
LEFT JOIN products ON users.favorite_product = products.id
```

Run example »

Which will give you this result:

```
[
  { user: 'John', favorite: 'Chocolate Heaven' },
  { user: 'Peter', favorite: 'Chocolate Heaven' },
  { user: 'Amy', favorite: 'Tasty Lemons' },
  { user: 'Hannah', favorite: null },
  { user: 'Michael', favorite: null }
]
```

---

Right Join

If you want to return all products, and the users who have them as their favorite, even if no user have them as their favorite, use the RIGHT JOIN statement:

Example

Select all products and the user who have them as their favorite:

```
SELECT users.name AS user,
products.name AS favorite
FROM users
RIGHT JOIN products ON users.favorite_product = products.id
```

Output:

```
[
  { user: 'John', favorite: 'Chocolate Heaven' },
  { user: 'Peter', favorite: 'Chocolate Heaven' },
  { user: 'Amy', favorite: 'Tasty Lemons' },
  { user: null, favorite: 'Vanilla Dreams' }
]
```

Reference: https://www.w3schools.com/nodejs/