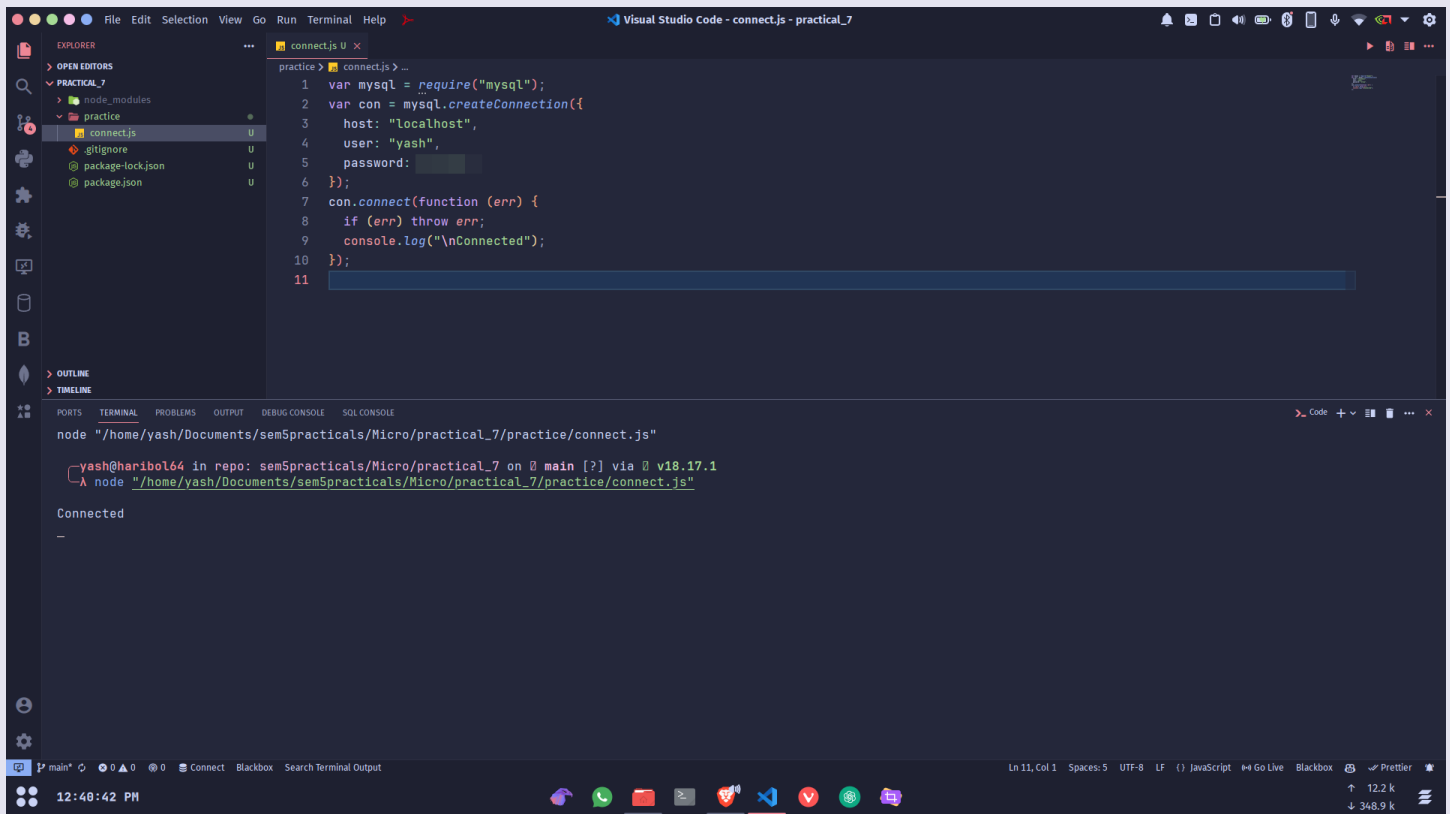


Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7

Scenario : Demonstrate the nodeJS application with MySQL Database. Anurag is selling clothes online to his customers via his website. He wants to manage his customer's records and for that, he wants to perform the following CRUD operations using MySQL and nodeJS:

Tasks, Code and Screenshots :

Initial steps : Connecting the database, creating the table, etc.

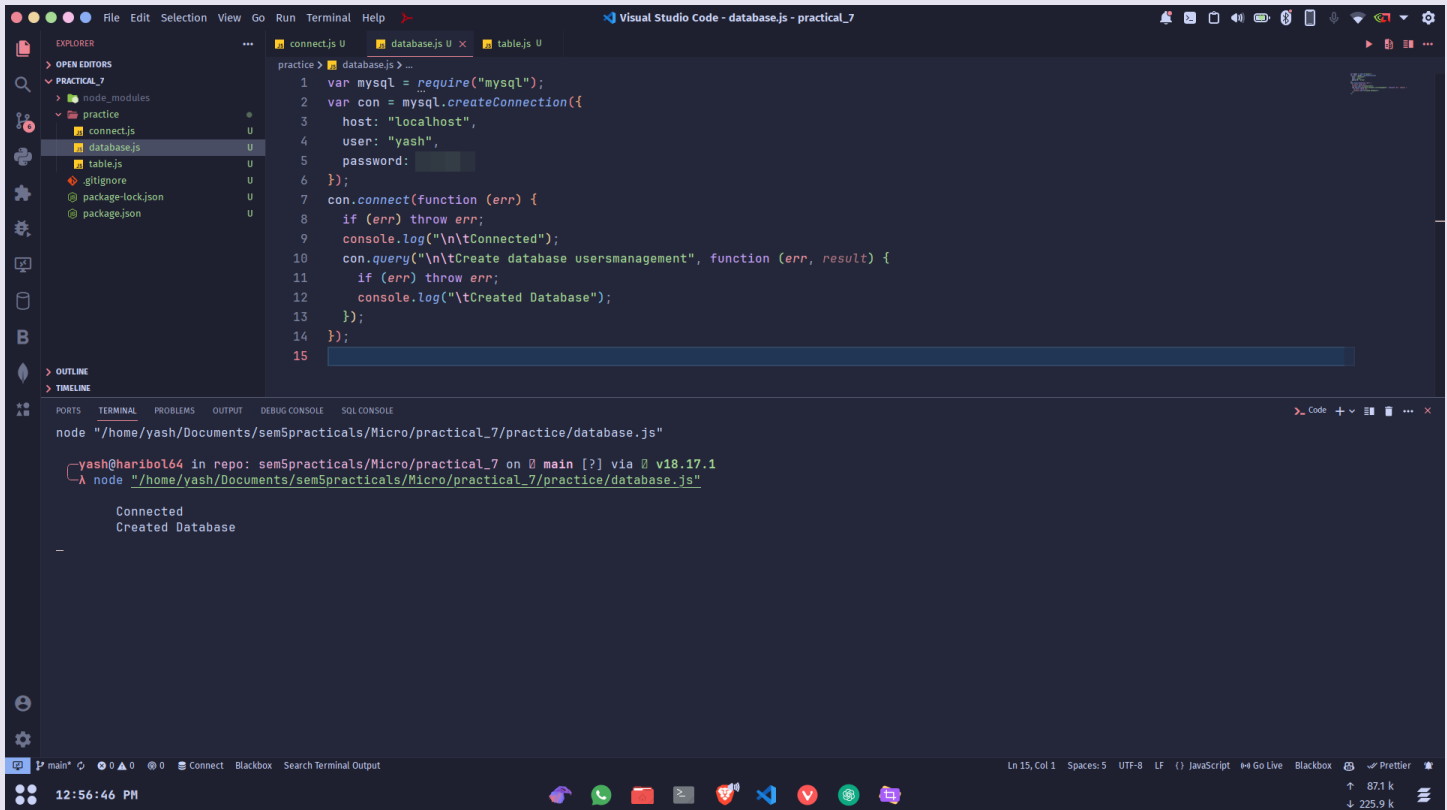


The screenshot shows the Visual Studio Code interface with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project structure with a 'practice' folder containing 'connect.js', '.gitignore', 'package-lock.json', and 'package.json'. The code editor displays the following JavaScript code in 'connect.js':

```
1 var mysql = require("mysql");
2 var con = mysql.createConnection({
3   host: "localhost",
4   user: "yash",
5   password: "yash",
6 });
7 con.connect(function (err) {
8   if (err) throw err;
9   console.log("Connected");
10 });
11
```

The terminal at the bottom shows the command to run the application: `node "/home/yash/Documents/sem5practicals/Micro/practical_7/practice/connect.js"`. The output shows the command being executed and the message "Connected" being printed to the console.

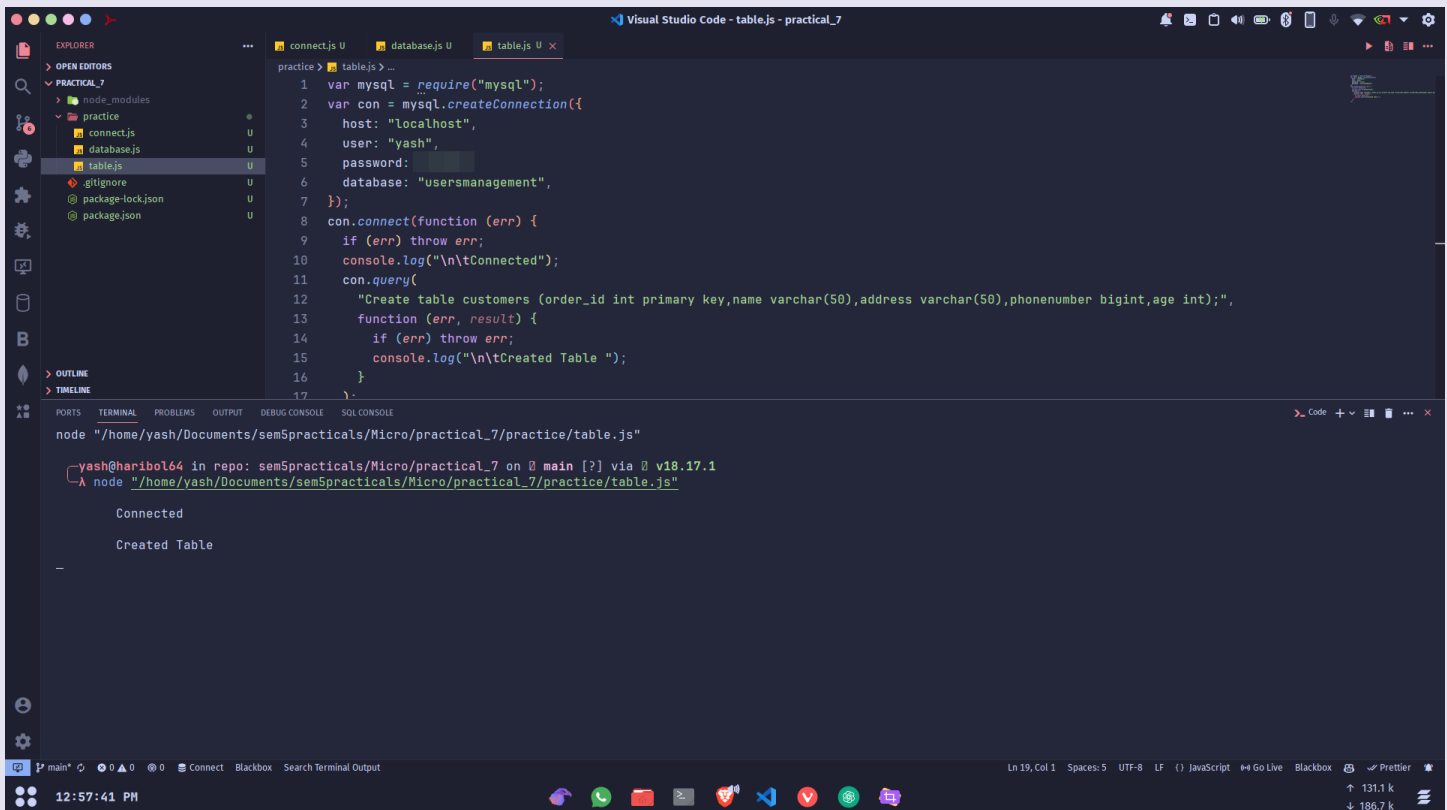
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7



The screenshot shows the Visual Studio Code editor with the file `database.js` open. The Explorer sidebar on the left shows the project structure: `node_modules`, `practice` (containing `connect.js`, `database.js`, and `table.js`), `.gitignore`, `package-lock.json`, and `package.json`. The `database.js` file contains the following code:

```
1 var mysql = require("mysql");
2 var con = mysql.createConnection({
3   host: "localhost",
4   user: "yash",
5   password: "yash",
6 });
7 con.connect(function (err) {
8   if (err) throw err;
9   console.log("\n\tConnected");
10  con.query("\n\tCreate database usersmanagement", function (err, result) {
11    if (err) throw err;
12    console.log("\tCreated Database");
13  });
14 });
15
```

The terminal at the bottom shows the command `node "/home/yash/Documents/sem5practicals/Micro/practical_7/practice/database.js"` being executed, resulting in the output: `Connected` and `Created Database`.



The screenshot shows the Visual Studio Code editor with the file `table.js` open. The Explorer sidebar on the left shows the project structure: `node_modules`, `practice` (containing `connect.js`, `database.js`, and `table.js`), `.gitignore`, `package-lock.json`, and `package.json`. The `table.js` file contains the following code:

```
1 var mysql = require("mysql");
2 var con = mysql.createConnection({
3   host: "localhost",
4   user: "yash",
5   password: "yash",
6   database: "usersmanagement",
7 });
8 con.connect(function (err) {
9   if (err) throw err;
10  console.log("\n\tConnected");
11  con.query(
12    "Create table customers (order_id int primary key,name varchar(50),address varchar(50),phonenumber bigint,age int);",
13    function (err, result) {
14      if (err) throw err;
15      console.log("\n\tCreated Table ");
16    }
17  );
18 }
```

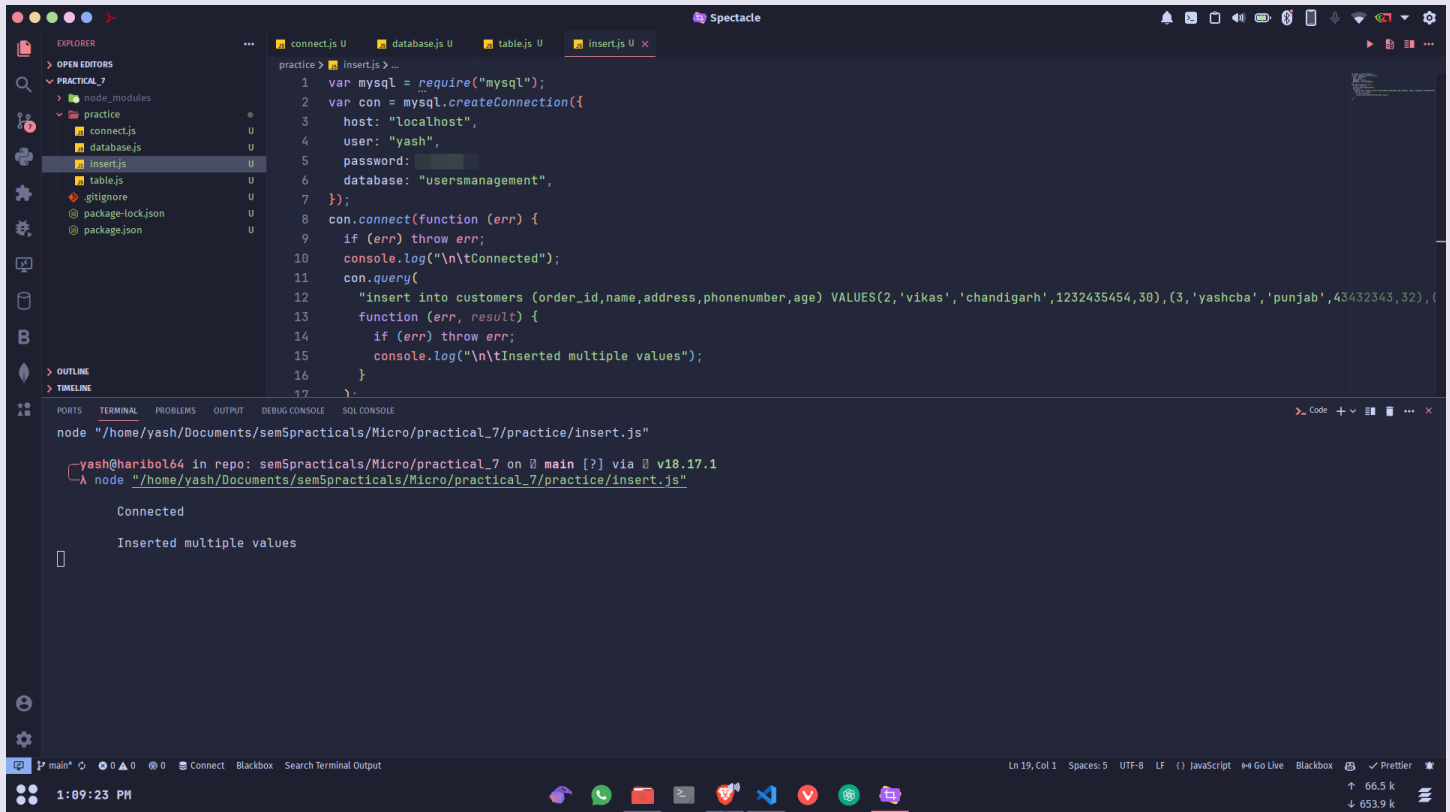
The terminal at the bottom shows the command `node "/home/yash/Documents/sem5practicals/Micro/practical_7/practice/table.js"` being executed, resulting in the output: `Connected` and `Created Table`.

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7

Practical 7.1: Add new customer in Customers with (name,phon_no,order_id).

```
var mysql = require("mysql");
var con = mysql.createConnection({
  host: "localhost",
  user: "yash",
  password: "",
  database: "usersmanagement",
});
con.connect(function (err) {
  if (err) throw err;
  console.log("\n\tConnected");
  con.query(
    "insert into customers (order_id,name,address,phonenum,age)
VALUES(2,'vikas','chandigarh',1232435454,30),(3,'yashcba','punjab',4343234
3,32),(4,'abc','sciencecity',434343243,30),(5,'yash_gnu','jodphur',2124435
4,37);",
    function (err, result) {
      if (err) throw err;
      console.log("\n\tInserted multiple values");
    }
  );
});
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7



The screenshot shows a Visual Studio Code editor window with a dark theme. The Explorer sidebar on the left shows a project structure with folders like 'node_modules', 'practice', 'connect.js', 'database.js', 'insert.js', 'table.js', and files like '.gitignore', 'package-lock.json', and 'package.json'. The 'insert.js' file is open in the editor, showing a JavaScript script that connects to a MySQL database and inserts data. The script uses the 'mysql' module and the 'mysql.createConnection' function. It sets the host to 'localhost', user to 'yash', and database to 'usersmanagement'. It then connects to the database and inserts multiple values into the 'customers' table. The terminal at the bottom shows the command to run the script and its output, which includes 'Connected' and 'Inserted multiple values'.

```
1 var mysql = require("mysql");
2 var con = mysql.createConnection({
3   host: "localhost",
4   user: "yash",
5   password: "yash",
6   database: "usersmanagement",
7 });
8 con.connect(function (err) {
9   if (err) throw err;
10  console.log("\n\tConnected");
11  con.query(
12    "insert into customers (order_id,name,address,phonenumber,age) VALUES(2,'vikas','chandigarh',1232435454,30),(3,'yashcba','punjab',43432343,32),(4,'yashcba','punjab',43432343,32)",
13    function (err, result) {
14      if (err) throw err;
15      console.log("\n\tInserted multiple values");
16    }
17  );
18 }
```

node "/home/yash/Documents/sem5practicals/Micro/practical_7/practice/insert.js"

yash@haribol64 in repo: sem5practicals/Micro/practical_7 on 0 main [?] via 0 v18.17.1
A node "/home/yash/Documents/sem5practicals/Micro/practical_7/practice/insert.js"

Connected

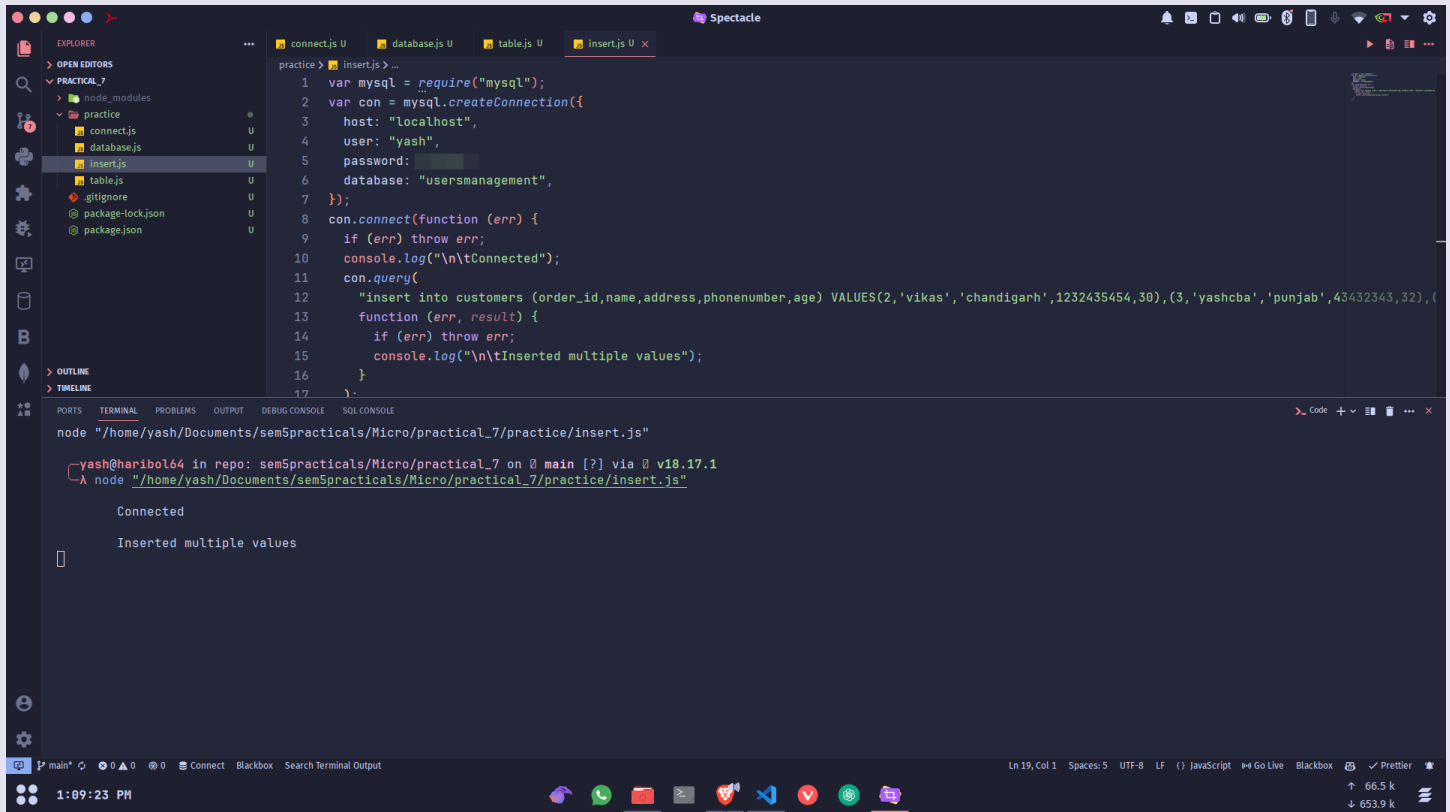
Inserted multiple values

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7

Practical 7.2: Add bulk of customers at the same time.

```
var mysql = require("mysql");
var con = mysql.createConnection({
  host: "localhost",
  user: "yash",
  password: "",
  database: "usersmanagement",
});
con.connect(function (err) {
  if (err) throw err;
  console.log("\n\tConnected");
  con.query(
    "insert into customers (order_id,name,address,phonenummer,age)
    VALUES(2,'vikas','chandigarh',1232435454,30),(3,'yashcba','punjab',4343234
    3,32),(4,'abc','sciencecity',434343243,30),(5,'yash_gnu','jodphur',2124435
    4,37);",
    function (err, result) {
      if (err) throw err;
      console.log("\n\tInserted multiple values");
    }
  );
});
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7



The screenshot shows a Visual Studio Code editor window with a dark theme. The Explorer sidebar on the left shows a project structure with folders like 'node_modules', 'practice', 'connect.js', 'database.js', 'insert.js', 'table.js', and files like '.gitignore', 'package-lock.json', and 'package.json'. The 'insert.js' file is open in the editor, showing a JavaScript script that connects to a MySQL database and inserts data. The script uses the 'mysql' module and the 'mysql.createConnection' function. It sets the host to 'localhost', user to 'yash', and password to a masked value. The database is 'usersmanagement'. It then connects to the database and runs an SQL insert statement. The terminal at the bottom shows the command to run the script and its output, which includes 'Connected' and 'Inserted multiple values'.

```
1 var mysql = require("mysql");
2 var con = mysql.createConnection({
3   host: "localhost",
4   user: "yash",
5   password: "yash",
6   database: "usersmanagement",
7 });
8 con.connect(function (err) {
9   if (err) throw err;
10  console.log("\n\tConnected");
11  con.query(
12    "insert into customers (order_id,name,address,phonenumber,age) VALUES(2,'vikas','chandigarh',1232435454,30),(3,'yashcba','punjab',43432343,32),(4,'yashcba','punjab',43432343,32)",
13    function (err, result) {
14      if (err) throw err;
15      console.log("\n\tInserted multiple values");
16    }
17  );
18 }
```

node "/home/yash/Documents/sem5practicals/Micro/practical_7/practice/insert.js"

yash@haribol64 in repo: sem5practicals/Micro/practical_7 on 0 main [?] via 0 v18.17.1
A node "/home/yash/Documents/sem5practicals/Micro/practical_7/practice/insert.js"

Connected

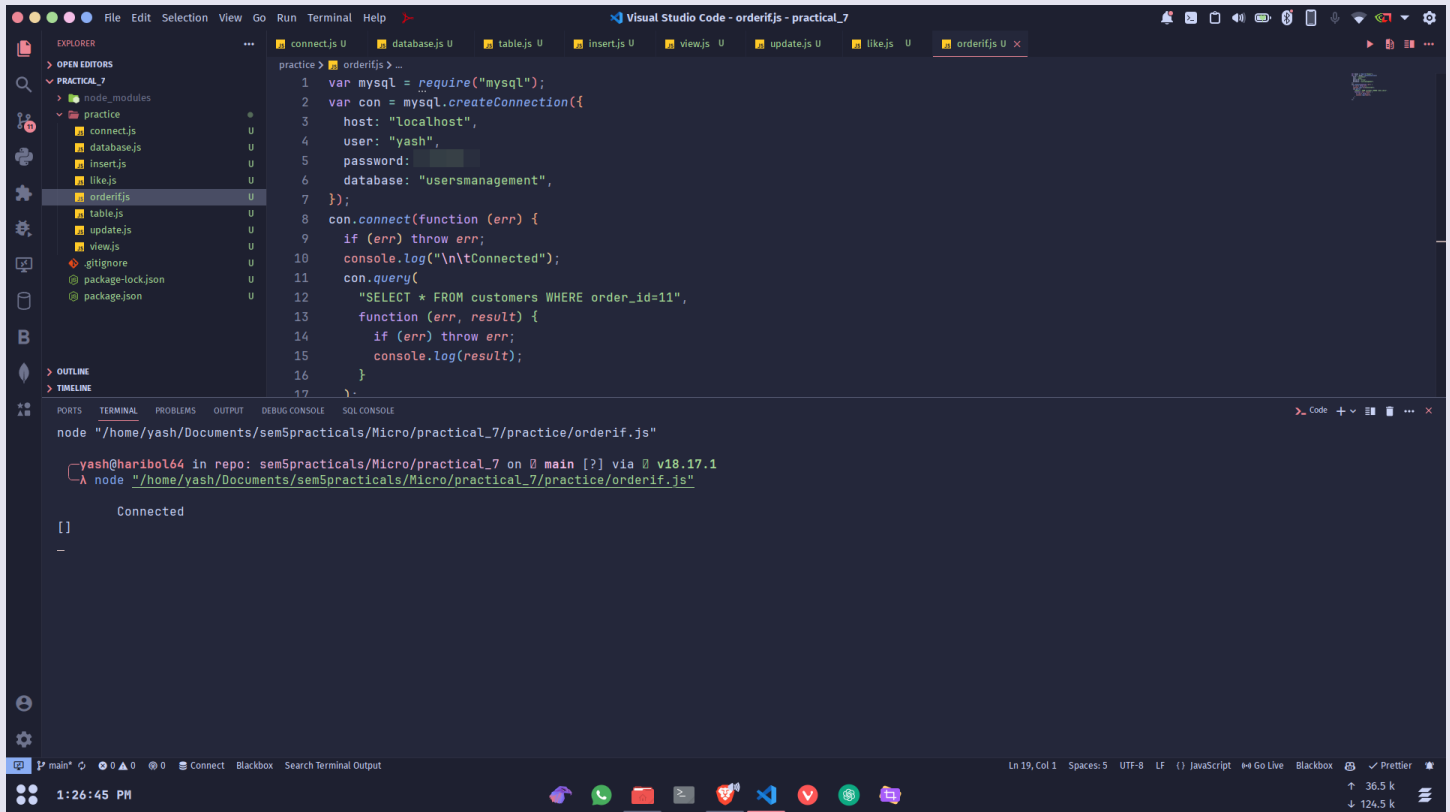
Inserted multiple values

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7

Practical 7.3: only display those customer details whose order_id=11

```
var mysql = require("mysql");
var con = mysql.createConnection({
  host: "localhost",
  user: "yash",
  password: "",
  database: "usersmanagement",
});
con.connect(function (err) {
  if (err) throw err;
  console.log("\n\tConnected");
  con.query(
    "SELECT * FROM customers WHERE order_id=11",
    function (err, result) {
      if (err) throw err;
      console.log(result);
    }
  );
});
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7



The screenshot displays the Visual Studio Code interface with a project named 'practical_7'. The Explorer sidebar on the left shows the file structure, including a 'practice' folder containing several JavaScript files. The 'orderif.js' file is selected and open in the editor. The code in the editor is a Node.js script that connects to a MySQL database and executes a query. The terminal at the bottom shows the command to run the script, which successfully connects to the database and outputs an empty array '[]'.

```
1 var mysql = require("mysql");
2 var con = mysql.createConnection({
3   host: "localhost",
4   user: "yash",
5   password: "yash",
6   database: "usersmanagement",
7 });
8 con.connect(function (err) {
9   if (err) throw err;
10  console.log("\n\tConnected");
11  con.query(
12    "SELECT * FROM customers WHERE order_id=11",
13    function (err, result) {
14      if (err) throw err;
15      console.log(result);
16    }
17  );
18 }
```

node "/home/yash/Documents/sem5practicals/Micro/practical_7/practice/orderif.js"

yash@haribol64 in repo: sem5practicals/Micro/practical_7 on 0 main [?] via 0 v18.17.1
A node "/home/yash/Documents/sem5practicals/Micro/practical_7/practice/orderif.js"

Connected

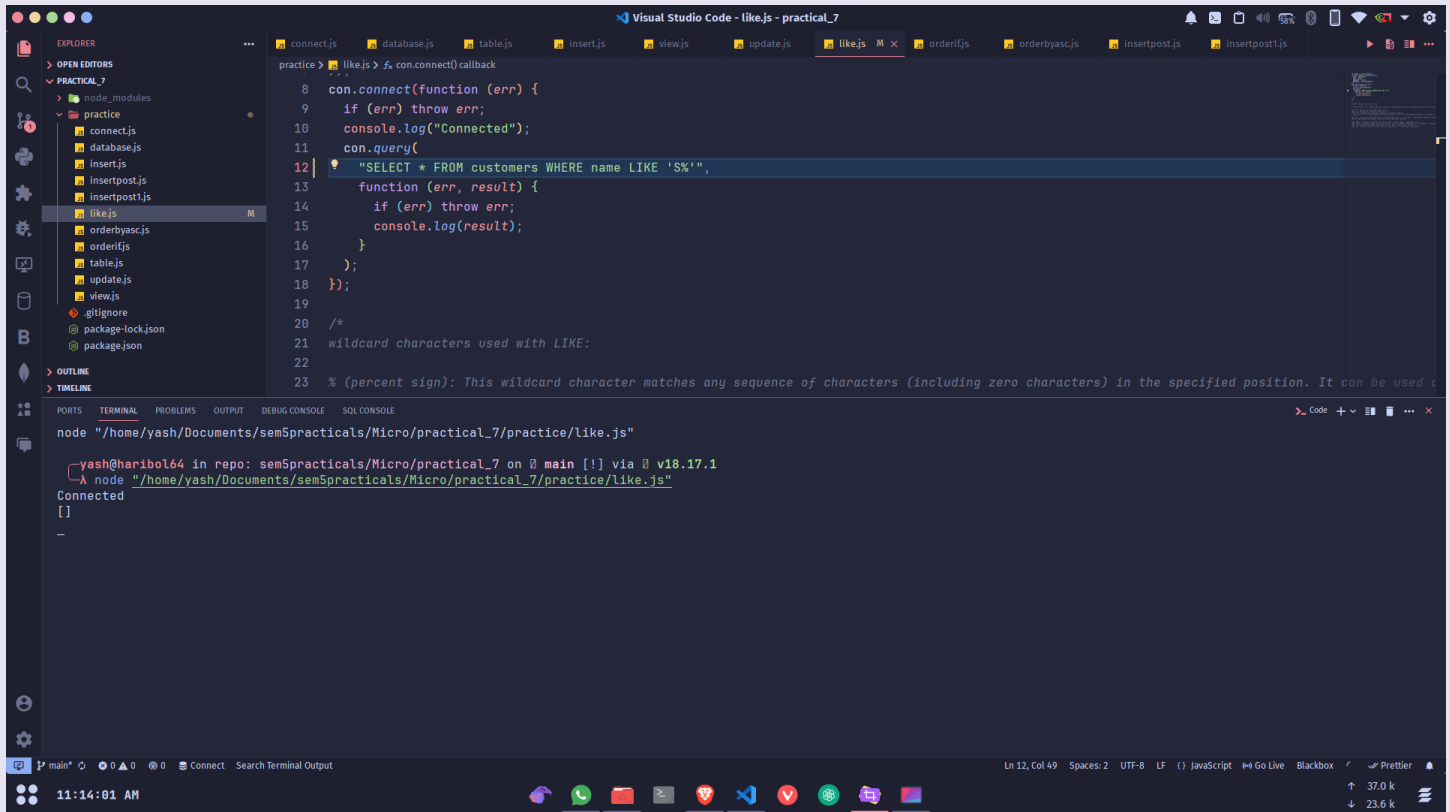
[]

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7

Practical 7.4: Select records where the address starts with the letter 'S'

```
var mysql = require("mysql");
var con = mysql.createConnection({
  host: "localhost",
  user: "yash",
  password: "",
  database: "usersmanagement",
});
con.connect(function (err) {
  if (err) throw err;
  console.log("Connected");
  con.query(
    "SELECT * FROM customers WHERE name LIKE 'S%'",
    function (err, result) {
      if (err) throw err;
      console.log(result);
    }
  );
});
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7



The screenshot displays the Visual Studio Code interface for a project named 'like.js - practical_7'. The Explorer sidebar on the left shows a file tree with folders like 'node_modules' and 'practical_7', and files such as 'connect.js', 'database.js', 'table.js', 'insert.js', 'view.js', 'update.js', 'like.js', 'orderbyasc.js', 'orderbydesc.js', 'insertpost.js', and 'insertpost1.js'. The main editor window shows the content of 'like.js'.

```
8  con.connect(function (err) {  
9    if (err) throw err;  
10   console.log("Connected");  
11   con.query(  
12     "SELECT * FROM customers WHERE name LIKE '%'",  
13     function (err, result) {  
14       if (err) throw err;  
15       console.log(result);  
16     }  
17   );  
18 });  
19  
20 /*  
21 wildcard characters used with LIKE:  
22  
23 % (percent sign): This wildcard character matches any sequence of characters (including zero characters) in the specified position. It can be used e
```

The bottom panel shows the TERMINAL with the following output:

```
node "/home/yash/Documents/sem5practicals/Micro/practical_7/practice/like.js"  
  
yash@haribol64 in repo: sem5practicals/Micro/practical_7 on 0 main [!] via 0 v18.17.1  
A node "/home/yash/Documents/sem5practicals/Micro/practical_7/practice/like.js"  
Connected  
[]  
—
```

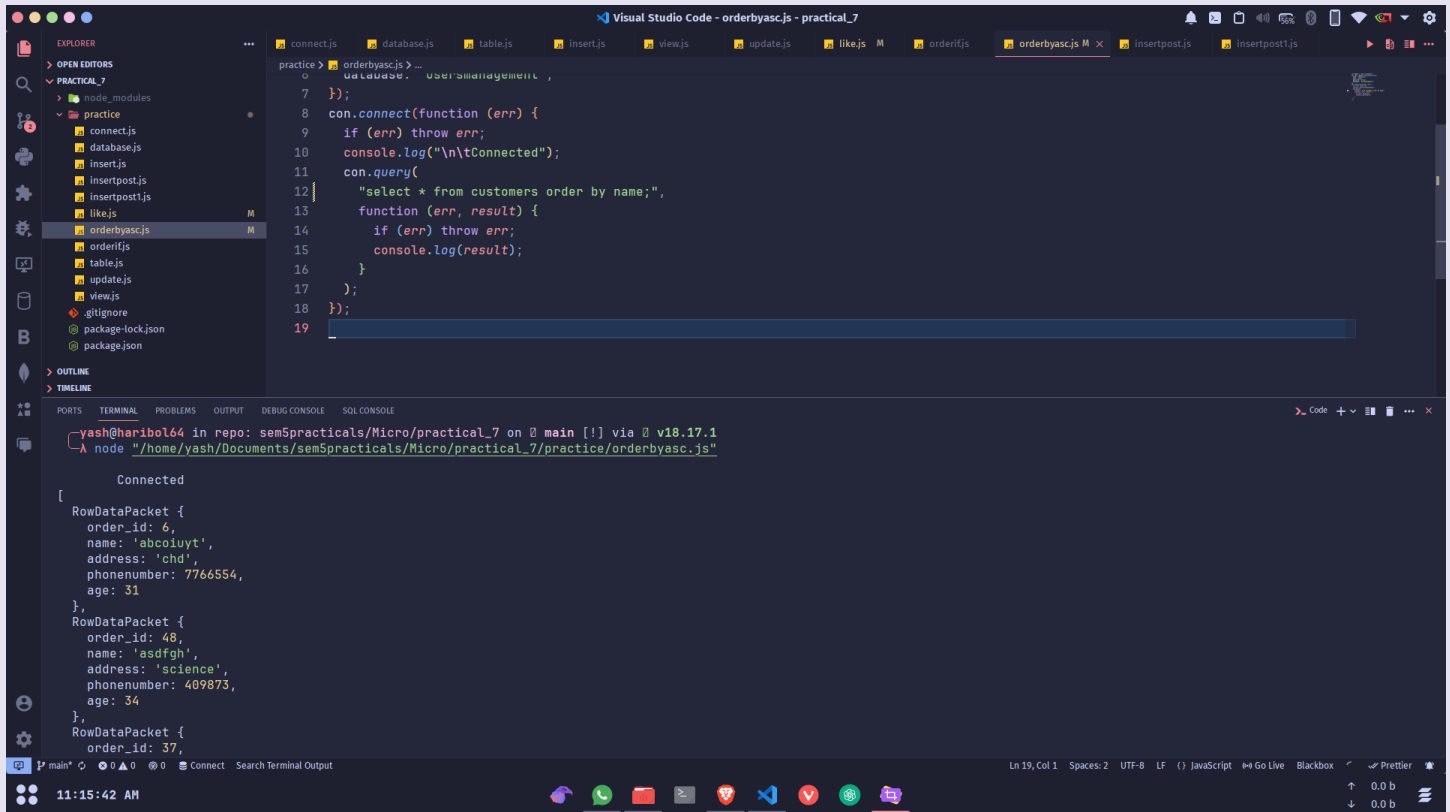
The status bar at the bottom indicates the file is 'main' on the 'main' branch, with 0 changes. It also shows the current cursor position as 'Ln 12, Col 49' and the file encoding as 'UTF-8'. The bottom right corner shows the file size as 37.0 k and the number of lines as 23.6 k.

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7

Practical 7.5: Sort the customer list alphabetically by name

```
var mysql = require("mysql");
var con = mysql.createConnection({
  host: "localhost",
  user: "yash",
  password: "",
  database: "usersmanagement",
});
con.connect(function (err) {
  if (err) throw err;
  console.log("\n\tConnected");
  con.query(
    "select * from customers order by name;",
    function (err, result) {
      if (err) throw err;
      console.log(result);
    }
  );
});
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7



The screenshot shows the Visual Studio Code editor with a project named 'practical_7'. The Explorer sidebar on the left lists the project files, including 'orderbyasc.js'. The main editor window displays the content of 'orderbyasc.js', which is a JavaScript file using the Mongoose library to connect to a database and query data. The code includes a database connection, a query to select all customers ordered by name, and a console log of the results. The terminal at the bottom shows the command to run the file and the output, which is a JSON array of customer objects.

```
practical_7 > orderbyasc.js > ...
6 database: usermanagement,
7 });
8
9 con.connect(function (err) {
10   if (err) throw err;
11   console.log("\n\tConnected");
12   con.query(
13     "select * from customers order by name;",
14     function (err, result) {
15       if (err) throw err;
16       console.log(result);
17     }
18   );
19 });
```

Connected

```
[
  RowDataPacket {
    order_id: 6,
    name: 'abcoiuyt',
    address: 'chd',
    phonenumber: 7766554,
    age: 31
  },
  RowDataPacket {
    order_id: 48,
    name: 'asdfgh',
    address: 'science',
    phonenumber: 409873,
    age: 34
  },
  RowDataPacket {
    order_id: 37,

```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7

Practical 7.6: Delete any record with the phone_no "12345"

```
var mysql = require("mysql");
var con = mysql.createConnection({
  host: "localhost",
  user: "yash",
  password: "",
  database: "usersmanagement",
});
con.connect(function (err) {
  if (err) throw err;
  console.log("\n\tConnected");
  con.query(
    "delete from customers where phonenumber=12345;",
    function (err, result) {
      if (err) throw err;
      console.log(result);
    }
  );
});
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7

The screenshot shows the Visual Studio Code editor with a project named 'practical_7'. The Explorer sidebar on the left shows the file structure, including 'delete.js' which is currently selected. The main editor area displays the following JavaScript code:

```
practical_7 > delete.js > ...
8  con.connect(function (err) {
9    if (err) throw err;
10   console.log("\n\tConnected");
11   con.query(
12     "delete from customers where phonenumber=12345;",
13     function (err, result) {
14       if (err) throw err;
15       console.log(result);
16     }
17   );
18 });
19
```

The bottom panel shows the TERMINAL output:

```
node "/home/yash/Documents/sem5practicals/Micro/practical_7/practice/delete.js"
yash@haribol64 in repo: sem5practicals/Micro/practical_7 on 0 main [!?] via 0 v18.17.1
A node "/home/yash/Documents/sem5practicals/Micro/practical_7/practice/delete.js"

Connected
OkPacket {
  fieldCount: 0,
  affectedRows: 0,
  insertId: 0,
  serverStatus: 34,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0
}
```

The status bar at the bottom indicates the file is 'delete.js' at line 19, column 1, using UTF-8 encoding and LF line endings. It also shows the file is saved and formatted with Prettier.

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7

Practical 7.7: Create an API to post customer records.

```
var mysql = require("mysql");
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");
const app = express();
app.use(cors());
// Configuring body parser middleware
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
var port = 8080;
var con = mysql.createConnection({
  host: "localhost",
  user: "yash",
  password: "",
  database: "usersmanagement",
});
var sql =
  "INSERT INTO customers (order_id,name,address,phonenummer,age) VALUES (?, ?, ?, ?, ?);";
var order, nme, add, phone, ag;

app.post("/api/store", (req, res) => {
  (order = req.body.order_id),
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7

```
(nme = req.body.name),  
(add = req.body.address),  
(phone = req.body.phonenumber),  
(ag = req.body.age);  
  
//values.push(product);  
res.send("entry done");  
con.connect(function (err) {  
  if (err) throw err;  
  console.log("\n\tConnected");  
  con.query(sql, [order, nme, add, phone, ag], function (err, result) {  
    if (err) throw err;  
    console.log("\n\tInserted multiple values");  
  });  
});  
});  
});  
app.listen(port);
```


Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 7

The image shows a development environment with Visual Studio Code and Postman. In Visual Studio Code, the file explorer on the left shows a project structure for 'practical_7' with files like 'connect.js', 'database.js', 'delete.js', 'insert.js', 'insertpost1.js', 'like.js', 'orderbyasc.js', 'orderbyid.js', 'table.js', 'update.js', 'views.js', and '.gitignore'. The main editor displays 'insertpost1.js' with the following code:

```
26 (primo = req.body.phonenumber) /
27 (ag = req.body.age);
28 //values.push(product);
29 res.send("entry done");
30 con.connect(function (err) {
31   if (err) throw err;
32   console.log("\n\tConnected");
33   con.query(sql, [order, nme, add, p
34     if (err) throw err;
35     console.log("\n\tInserted multip
36   });
37 });
38 });
39 app.listen(port);
40
```

The terminal at the bottom shows the command to run the application: `node "/home/yash/Documents/sem5practicals/Micro/practical_7/practice/insertpost1.js"`. The output indicates the application is connected and has inserted multiple values.

Postman is open to the right, showing a collection of API requests. The first request is a POST to `http://localhost:8080/api/store` with a JSON body: `{ "order_id": "108", "name": "Haribol", "address": "Vindavan", "phonenumber": "1728", "age": "21" }`. The response is a 200 OK status with a response time of 24 ms and a body of `entry done`.