

**Name: Patel Shivam S.**

**Enroll no: 21162101019**

**Sem: 5      Batch: 51      Branch: CBA**

**Sub: Microservices**

## **Practical 7**

**AIM:** 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:

Practical 7.1: Add new customer in Customers with (name,phon\_no,order\_id).

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

Practical 7.3: only display those customer details whose order\_id=111

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

Practical 7.5: Sort the customer list alphabetically by name

Practical 7.6: Delete any record with the phone\_no "12345"

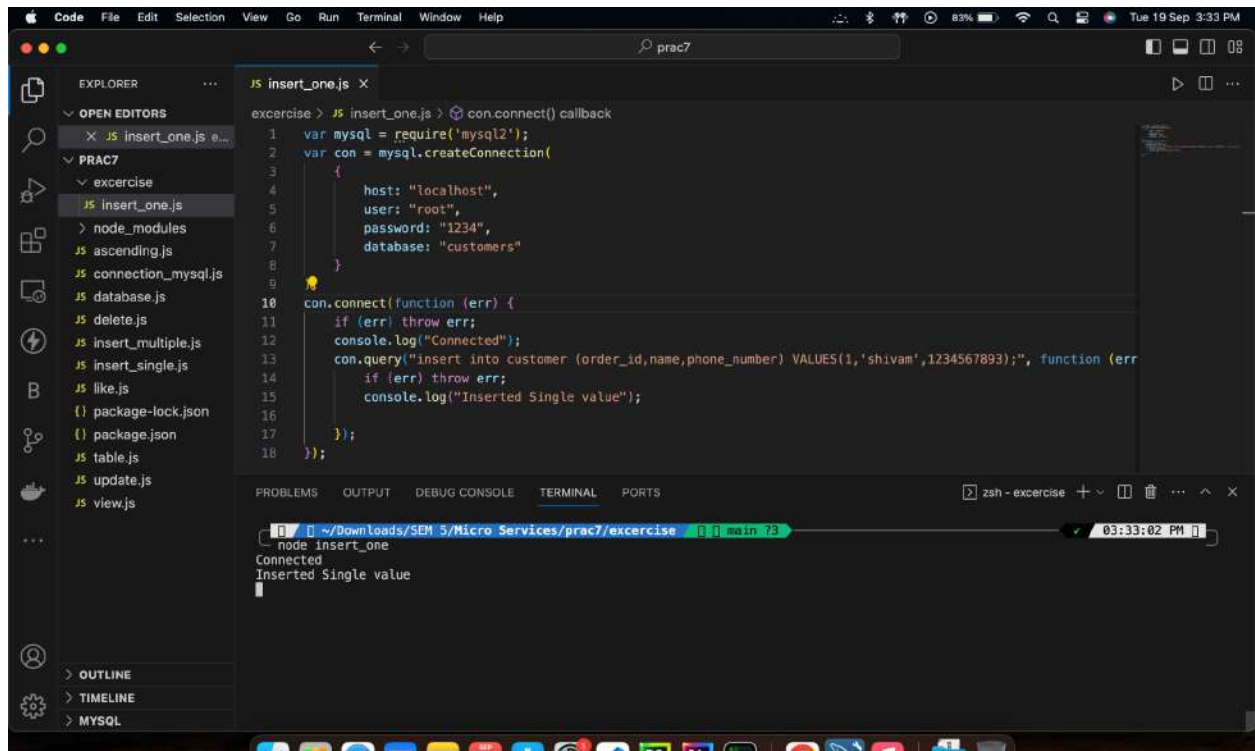
Practical 7.7: Create an API to post customer records.

**GITHUB LINK:**

**[https://github.com/Shivam3783/microservice\\_practicals/tree/main/prac7](https://github.com/Shivam3783/microservice_practicals/tree/main/prac7)**

## Practical 7.1: Add new customer in Customers with(name,phon\_no,order\_id).

### CODE & OUTPUT:

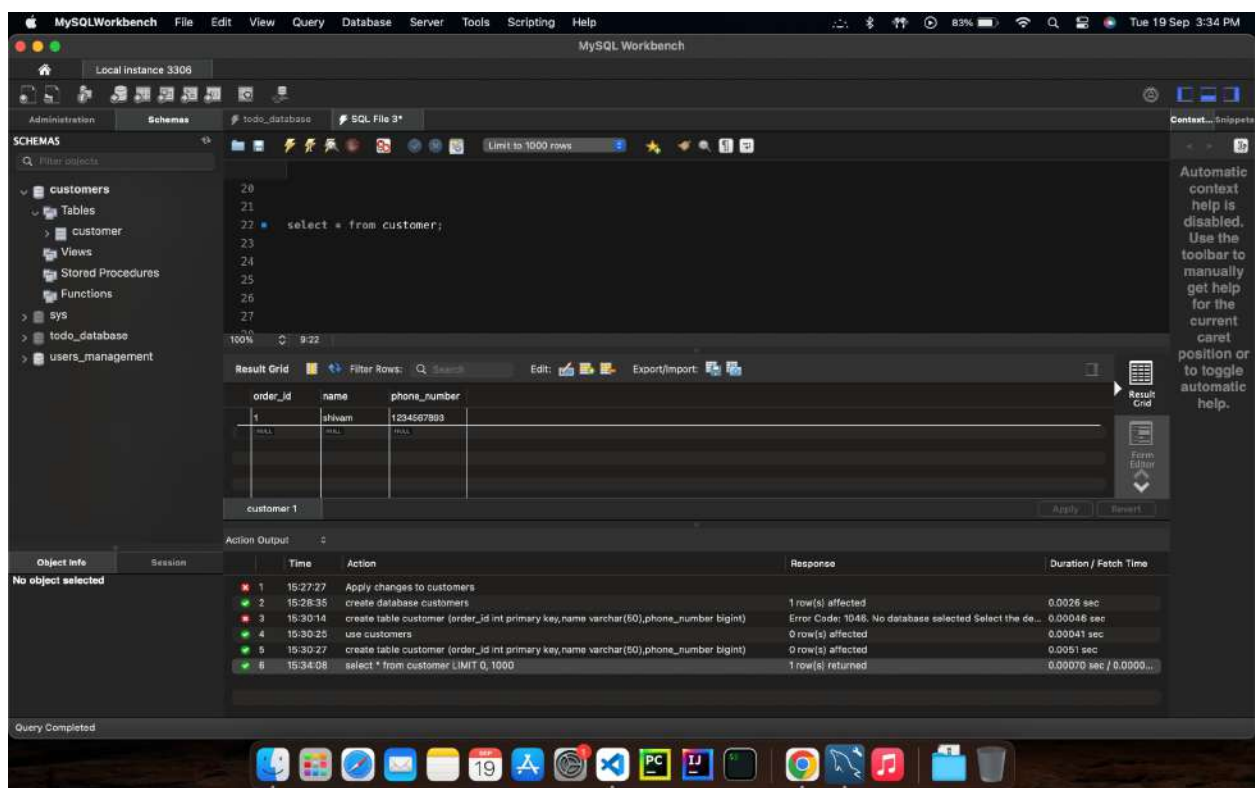


The screenshot shows the VS Code editor with a file named `insert_one.js` open. The code is as follows:

```
1 var mysql = require('mysql2');
2 var con = mysql.createConnection({
3   host: "localhost",
4   user: "root",
5   password: "1234",
6   database: "customers"
7 });
8
9
10 con.connect(function (err) {
11   if (err) throw err;
12   console.log("Connected");
13   con.query("insert into customer (order_id,name,phone_number) VALUES(1,'shivam',1234567893);", function (err) {
14     if (err) throw err;
15     console.log("Inserted Single value");
16   });
17 });
18
```

The terminal output shows the command `node insert_one` being executed, resulting in the output:

```
Connected
Inserted Single value
```

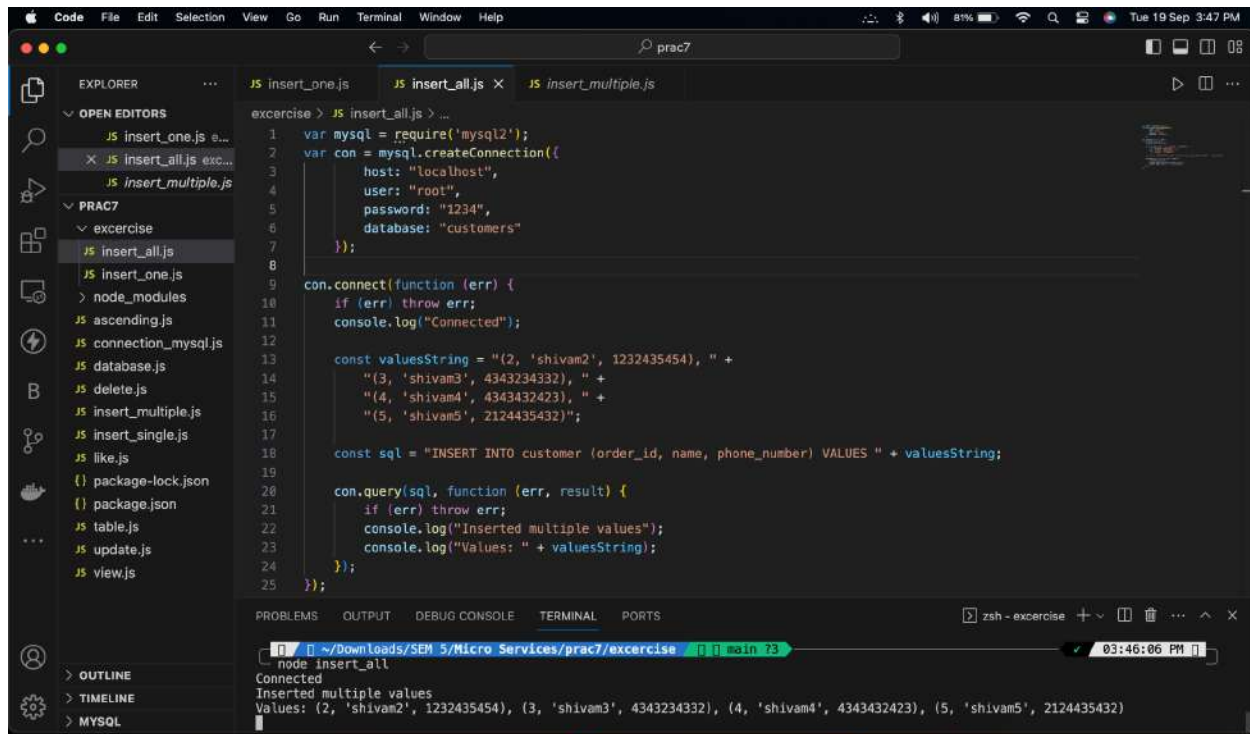


The screenshot shows the MySQL Workbench interface. The 'customers' table is selected, and the 'Action Output' log is visible at the bottom. The log shows the following actions:

Object Info	Session	Time	Action	Response	Duration / Fetch Time
No object selected		15:27:27	Apply changes to customers		
		15:28:35	create database customers	1 row(s) affected	0.0026 sec
		15:30:14	create table customer (order_id int primary key,name varchar(60),phone_number bigint)	Error Code: 1046. No database selected Select the de...	0.00046 sec
		15:30:25	use customers	0 row(s) affected	0.00041 sec
		15:30:27	create table customer (order_id int primary key,name varchar(60),phone_number bigint)	0 row(s) affected	0.0051 sec
		15:34:08	select * from customer LIMIT 0, 1000	1 row(s) returned	0.00070 sec / 0.0000...

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

### CODE & OUTPUT:

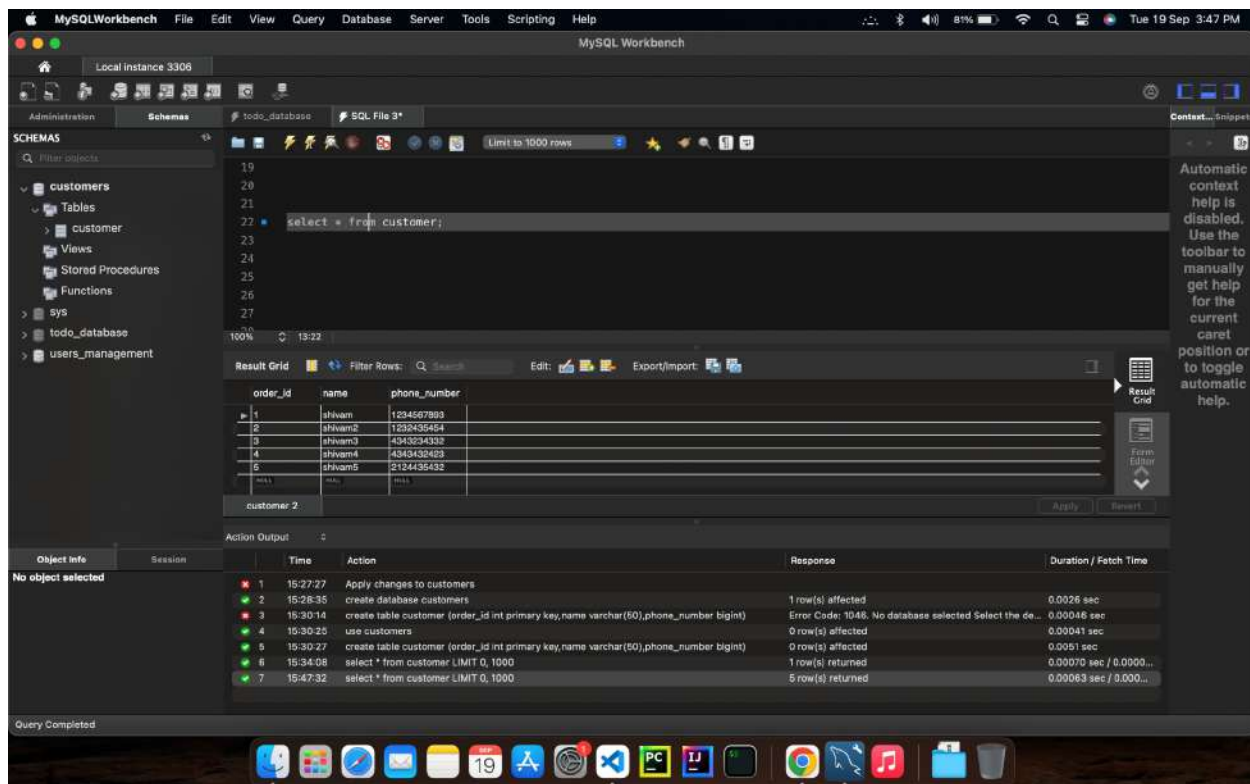


The screenshot shows the VS Code editor with a file named `insert_all.js` open. The code is as follows:

```
1 var mysql = require('mysql2');
2 var con = mysql.createConnection({
3   host: "localhost",
4   user: "root",
5   password: "1234",
6   database: "customers"
7 });
8
9 con.connect(function (err) {
10   if (err) throw err;
11   console.log("Connected");
12
13   const valuesString = "(2, 'shivam2', 1232435454), " +
14     "(3, 'shivam3', 4343234332), " +
15     "(4, 'shivam4', 4343432423), " +
16     "(5, 'shivam5', 2124435432)";
17
18   const sql = "INSERT INTO customer (order_id, name, phone_number) VALUES " + valuesString;
19
20   con.query(sql, function (err, result) {
21     if (err) throw err;
22     console.log("Inserted multiple values");
23     console.log("Values: " + valuesString);
24   });
25 });
```

The terminal output shows the following:

```
node insert_all
Connected
Inserted multiple values
Values: (2, 'shivam2', 1232435454), (3, 'shivam3', 4343234332), (4, 'shivam4', 4343432423), (5, 'shivam5', 2124435432)
```

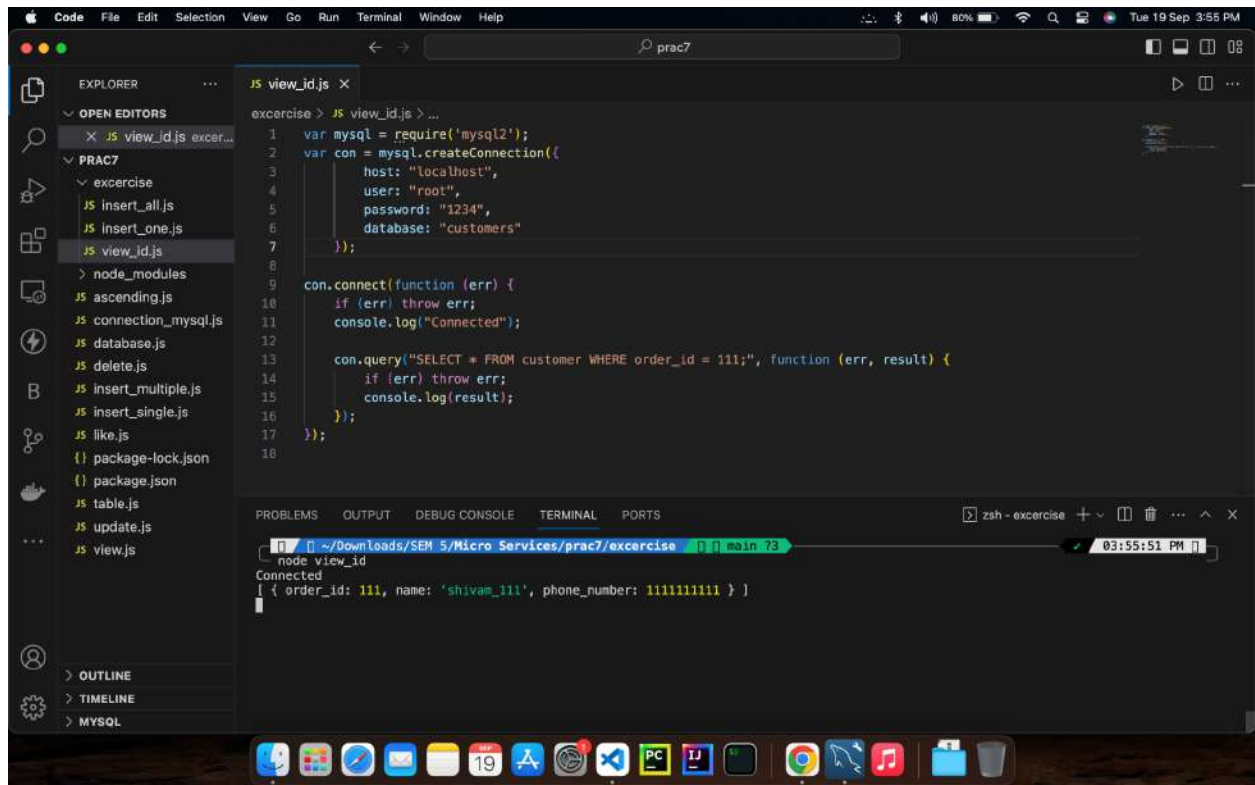


The screenshot shows the MySQL Workbench interface. The 'customers' table is selected, and the 'Action Output' log is visible at the bottom. The log shows the following actions:

Object Info	Session	Time	Action	Response	Duration / Fetch Time
No object selected					
		15:27:27	Apply changes to customers		
		15:28:35	create database customers	1 row(s) affected	0.0026 sec
		15:30:14	create table customer (order_id int primary key, name varchar(50), phone_number bigint)	Error Code: 1046. No database selected Select the de...	0.00046 sec
		15:30:25	use customers	0 row(s) affected	0.00041 sec
		15:30:27	create table customer (order_id int primary key, name varchar(50), phone_number bigint)	0 row(s) affected	0.0051 sec
		15:34:08	select * from customer LIMIT 0, 1000	1 row(s) returned	0.00070 sec / 0.0000...
		15:47:32	select * from customer LIMIT 0, 1000	5 row(s) returned	0.00063 sec / 0.0000...

## Practical 7.3: only display those customer details whose order\_id=111

### CODE & OUTPUT:

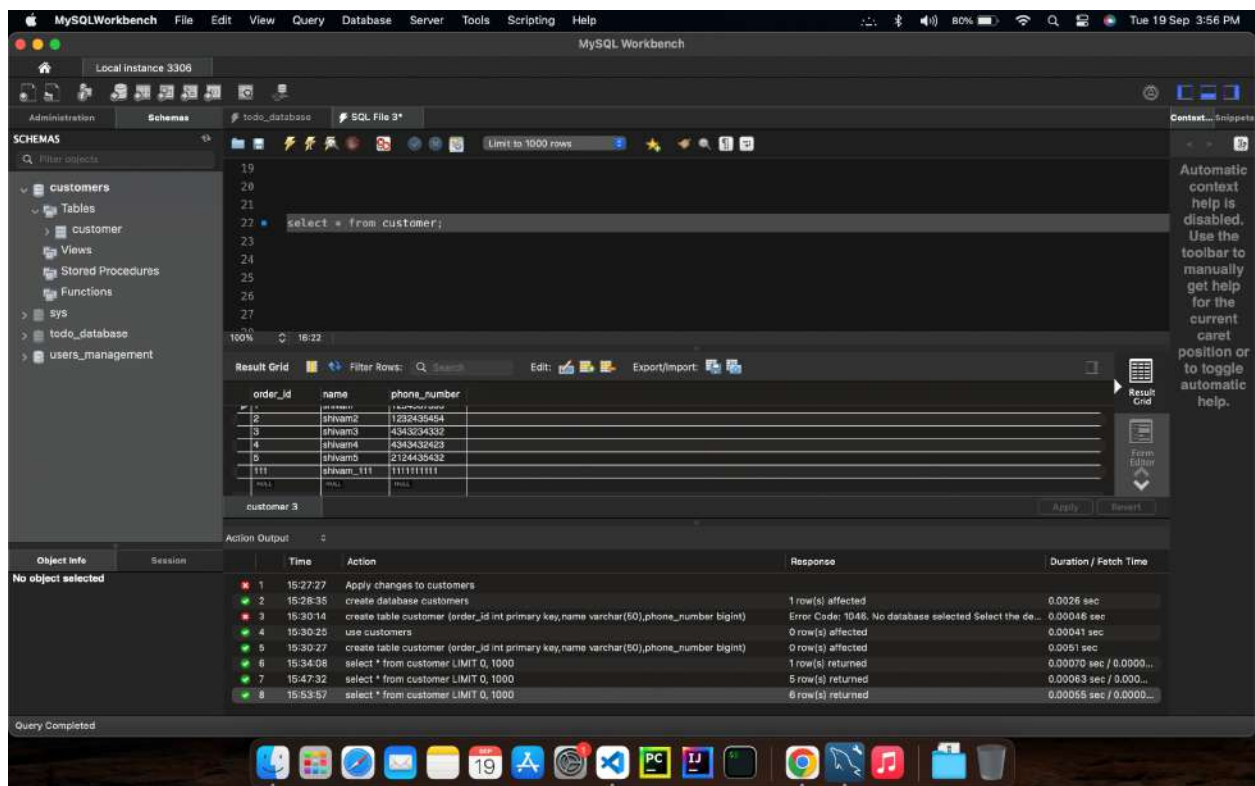


The screenshot shows the VS Code editor with a file named `view_id.js` open. The code in the file is as follows:

```
1 var mysql = require('mysql2');
2 var con = mysql.createConnection({
3   host: "localhost",
4   user: "root",
5   password: "1234",
6   database: "customers"
7 });
8
9 con.connect(function (err) {
10   if (err) throw err;
11   console.log("Connected");
12
13   con.query("SELECT * FROM customer WHERE order_id = 111;", function (err, result) {
14     if (err) throw err;
15     console.log(result);
16   });
17 });
```

The terminal output shows the command `node view_id` being executed, resulting in the following JSON output:

```
{ order_id: 111, name: 'shivam_111', phone_number: 1111111111 }
```



The screenshot shows the MySQL Workbench interface. The 'customers' table is selected, and the query `select * from customer;` is executed. The result grid shows the following data:

order_id	name	phone_number
1	shivam1	1123456789
2	shivam2	1234567890
3	shivam3	4343234321
4	shivam4	4343432423
5	shivam5	2124435432
111	shivam_111	1111111111
NULL	NULL	NULL

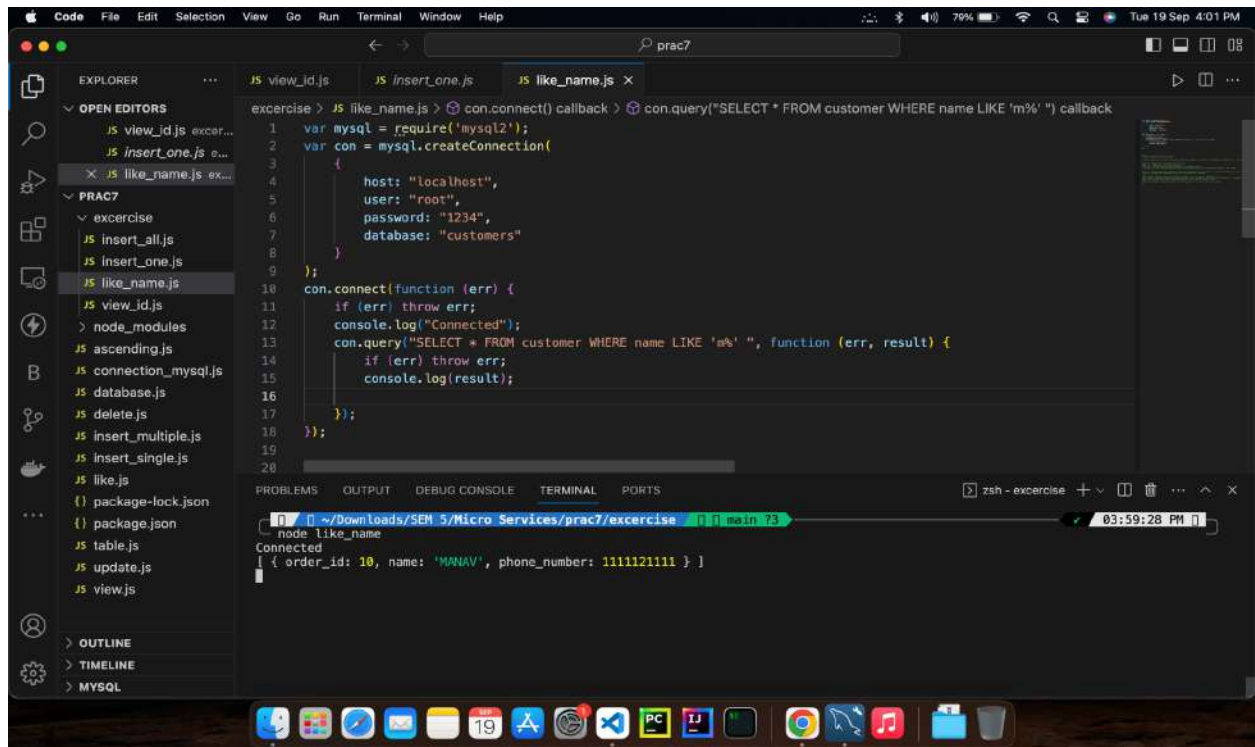
The 'Action Output' tab shows the following actions and their results:

Object Info	Session	Time	Action	Response	Duration / Fetch Time
No object selected		15:27:27	Apply changes to customers		
		15:28:35	create database customers	1 row(s) affected	0.0026 sec
		15:30:14	create table customer (order_id int primary key, name varchar(50), phone_number bigint)	Error Code: 1046. No database selected Select the de...	0.00046 sec
		15:30:25	use customers	0 row(s) affected	0.00041 sec
		15:30:27	create table customer (order_id int primary key, name varchar(50), phone_number bigint)	0 row(s) affected	0.00051 sec
		15:34:08	select * from customer LIMIT 0, 1000	1 row(s) returned	0.00070 sec / 0.0000...
		15:47:32	select * from customer LIMIT 0, 1000	5 row(s) returned	0.00063 sec / 0.0000...
		15:53:57	select * from customer LIMIT 0, 1000	6 row(s) returned	0.00055 sec / 0.0000...



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

### CODE & OUTPUT:

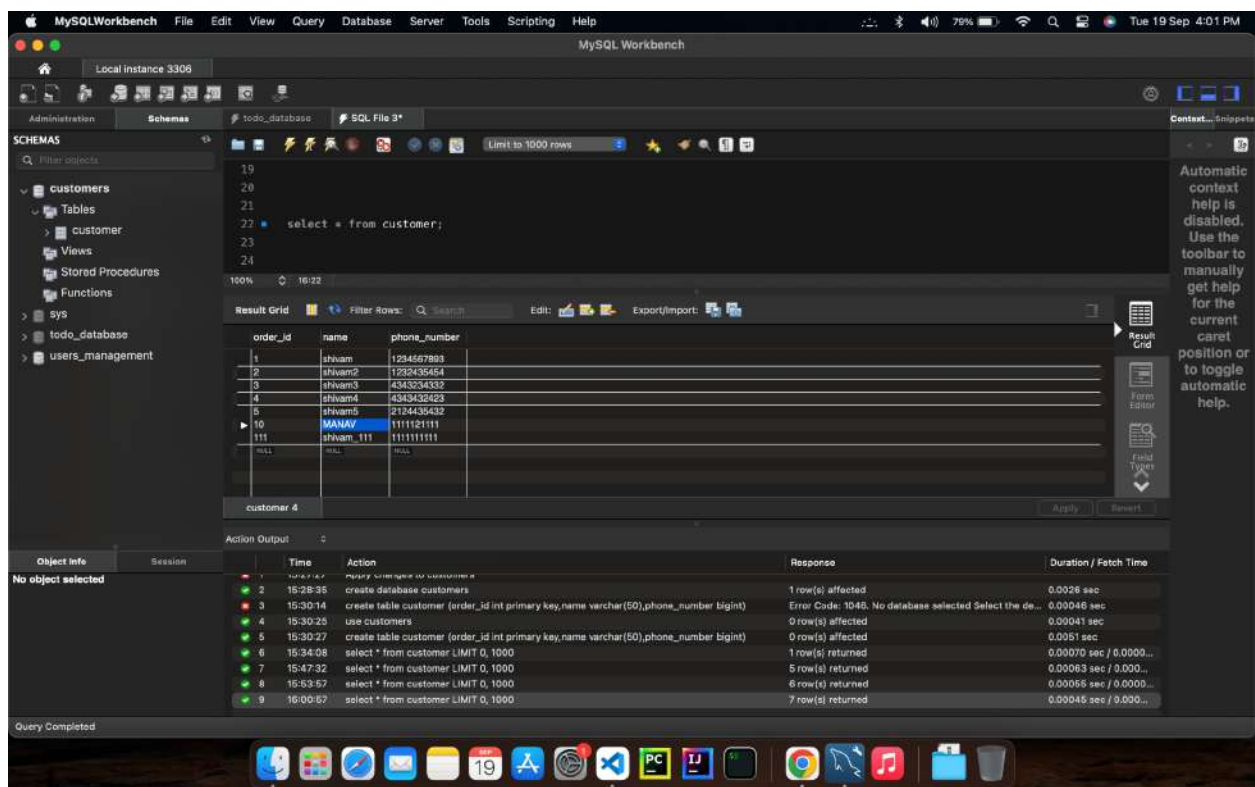


The screenshot shows the VS Code editor with a file named `like_name.js` open. The code is as follows:

```
1 var mysql = require('mysql2');
2 var con = mysql.createConnection({
3   host: "localhost",
4   user: "root",
5   password: "1234",
6   database: "customers"
7 });
8
9 con.connect(function (err) {
10   if (err) throw err;
11   console.log("Connected");
12   con.query("SELECT * FROM customer WHERE name LIKE 'm%' ", function (err, result) {
13     if (err) throw err;
14     console.log(result);
15   });
16 });
```

The terminal output shows the command `node like_name` being executed, resulting in the following JSON output:

```
{ order_id: 10, name: 'MANAV', phone_number: 1111121111 }
```



The screenshot shows the MySQL Workbench interface. The 'customers' table is selected, and the query `select * from customer;` is executed. The result grid shows the following data:

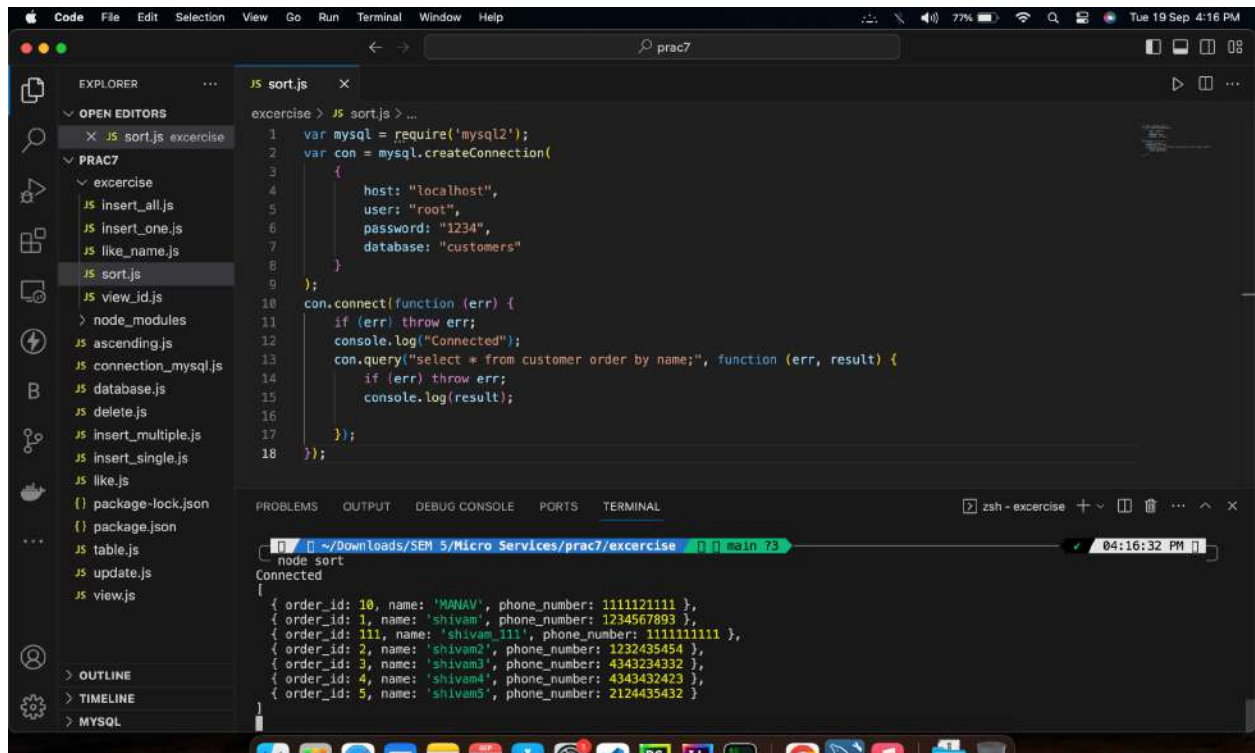
order_id	name	phone_number
1	shyam	1234567890
2	shyam2	1234567890
3	shyam3	4343234332
4	shyam4	4343234332
5	shyam5	2123434332
10	MANAV	1111121111
111	shyam_111	1111111111
NULL	NULL	NULL

The 'Action Output' tab shows the following actions and responses:

Time	Action	Response	Duration / Fetch Time
15:28:35	create database customers	1 row(s) affected	0.0026 sec
15:30:14	create table customer (order_id int primary key, name varchar(50), phone_number bigint)	Error Code: 1046. No database selected Select the de...	0.00046 sec
15:30:25	use customers	0 row(s) affected	0.00041 sec
15:30:27	create table customer (order_id int primary key, name varchar(50), phone_number bigint)	0 row(s) affected	0.00051 sec
15:34:08	select * from customer LIMIT 0, 1000	1 row(s) returned	0.00070 sec / 0.0000...
15:47:32	select * from customer LIMIT 0, 1000	5 row(s) returned	0.00063 sec / 0.0000...
15:53:57	select * from customer LIMIT 0, 1000	6 row(s) returned	0.00066 sec / 0.0000...
16:00:67	select * from customer LIMIT 0, 1000	7 row(s) returned	0.00046 sec / 0.0000...

## Practical 7.5: Sort the customer list alphabetically by name

### CODE & OUTPUT:

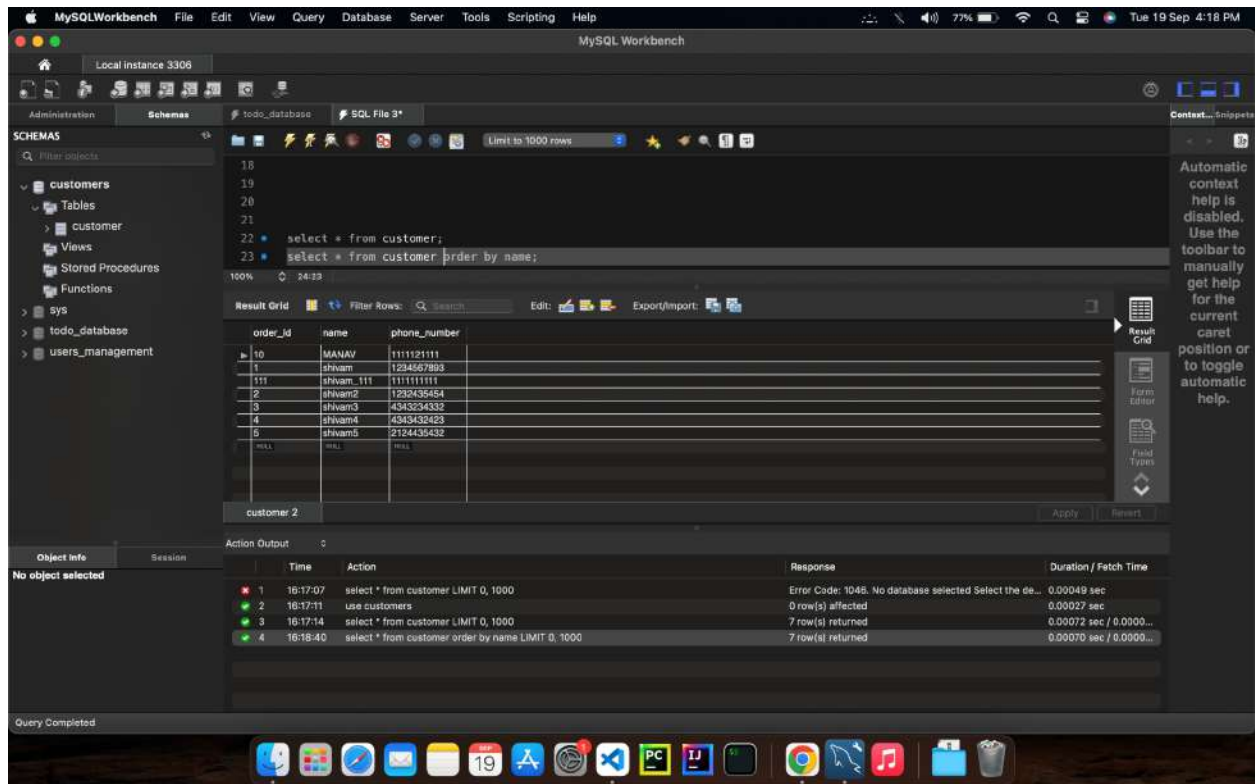


The screenshot shows the VS Code editor with a file named `sort.js` open. The code connects to a MySQL database and queries the `customer` table, sorting the results by name. The terminal output shows the successful execution of the script and the resulting data.

```
1 var mysql = require('mysql2');
2 var con = mysql.createConnection({
3   host: "localhost",
4   user: "root",
5   password: "1234",
6   database: "customers"
7 });
8
9
10 con.connect(function (err) {
11   if (err) throw err;
12   console.log("Connected");
13   con.query("select * from customer order by name;", function (err, result) {
14     if (err) throw err;
15     console.log(result);
16   });
17 });
18 });
```

Terminal Output:

```
node sort
Connected
[
  { order_id: 10, name: 'MANAV', phone_number: 1111121111 },
  { order_id: 1, name: 'shivam', phone_number: 1234567893 },
  { order_id: 111, name: 'shivam_111', phone_number: 1111111111 },
  { order_id: 2, name: 'shivam2', phone_number: 1232435454 },
  { order_id: 3, name: 'shivam3', phone_number: 4343234332 },
  { order_id: 4, name: 'shivam4', phone_number: 434323423 },
  { order_id: 5, name: 'shivam5', phone_number: 2124435432 }
]
```



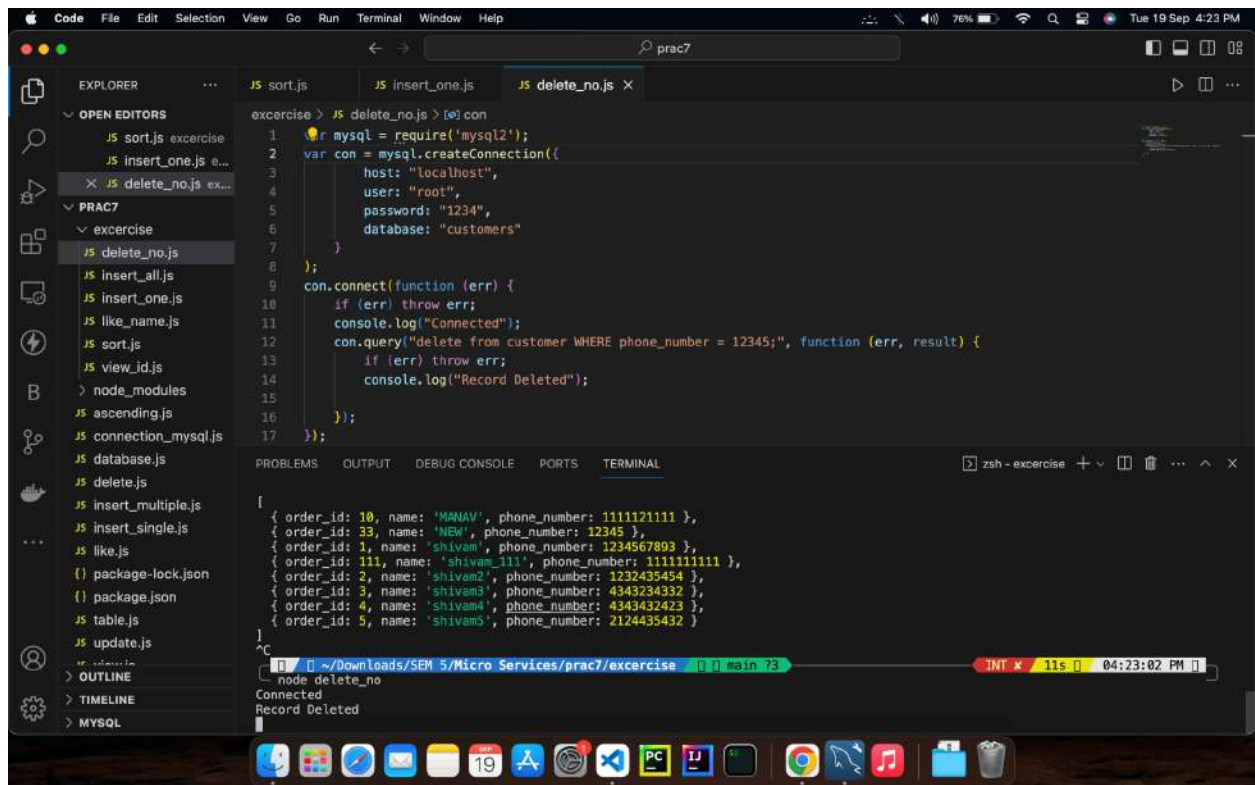
The screenshot shows the MySQL Workbench interface. The query `select * from customer order by name;` has been executed, and the results are displayed in the Result Grid. The Action Log at the bottom shows the sequence of operations performed.

order_id	name	phone_number
10	MANAV	1111121111
1	shivam	1234567893
111	shivam_111	1111111111
2	shivam2	1232435454
3	shivam3	4343234332
4	shivam4	434323423
5	shivam5	2124435432

Time	Action	Response	Duration / Fetch Time
16:17:07	select * from customer LIMIT 0, 1000	Error Code: 1048. No database selected Select the de...	0.00049 sec
16:17:11	use customers	0 row(s) affected	0.00027 sec
16:17:14	select * from customer LIMIT 0, 1000	7 row(s) returned	0.00072 sec / 0.0000...
16:18:40	select * from customer order by name LIMIT 0, 1000	7 row(s) returned	0.00070 sec / 0.0000...

## Practical 7.6: Delete any record with the phone\_no "12345"

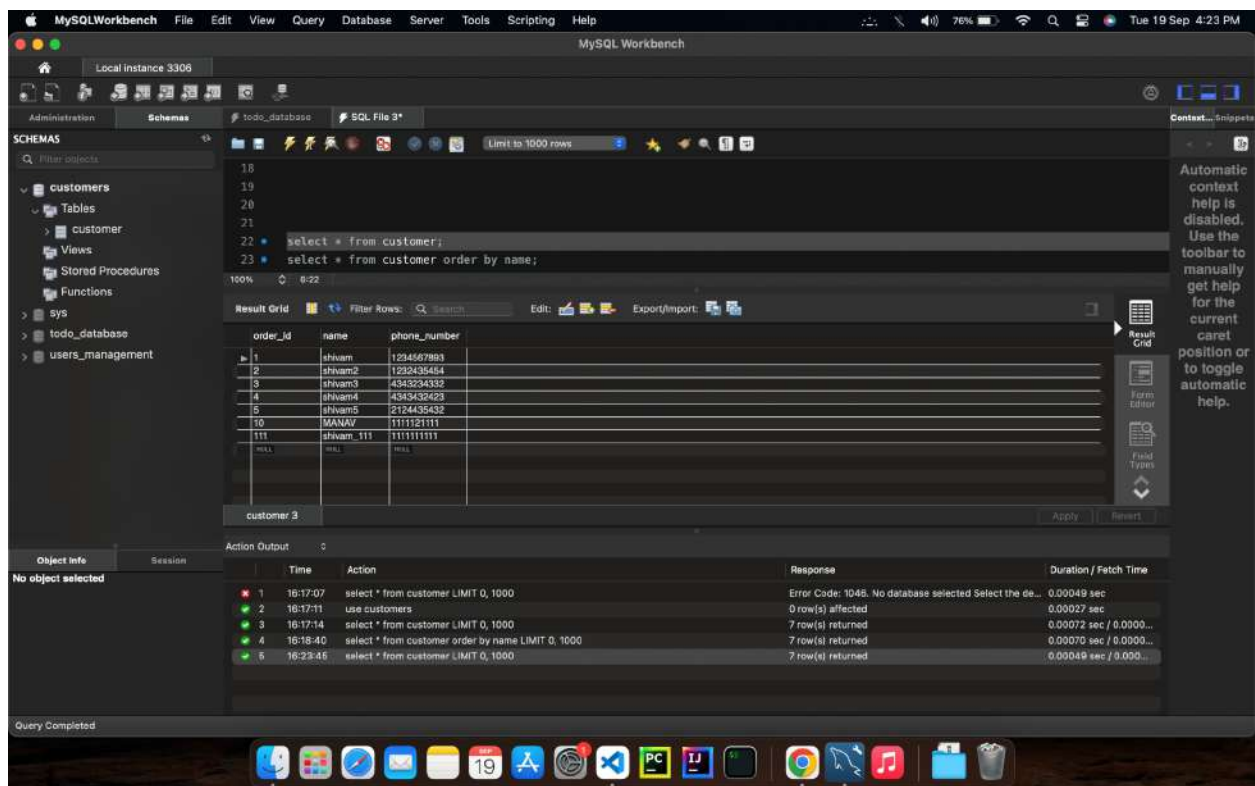
### CODE & OUTPUT:



```
exercise > JS delete_no.js > con
1  mysql = require('mysql2');
2  var con = mysql.createConnection({
3    host: "localhost",
4    user: "root",
5    password: "1234",
6    database: "customers"
7  });
8
9  con.connect(function (err) {
10   if (err) throw err;
11   console.log("Connected");
12   con.query("delete from customer WHERE phone_number = 12345;", function (err, result) {
13     if (err) throw err;
14     console.log("Record Deleted");
15   });
16 });
17 }
```

```
{ order_id: 10, name: 'MANAV', phone_number: 1111121111 },
{ order_id: 33, name: 'NEW', phone_number: 12345 },
{ order_id: 1, name: 'shivam', phone_number: 1234567893 },
{ order_id: 111, name: 'shivam_111', phone_number: 1111111111 },
{ order_id: 2, name: 'shivam2', phone_number: 1232435454 },
{ order_id: 3, name: 'shivam3', phone_number: 4343234332 },
{ order_id: 4, name: 'shivam4', phone_number: 4343432423 },
{ order_id: 5, name: 'shivam5', phone_number: 2124435432 }
```

node delete\_no  
Connected  
Record Deleted

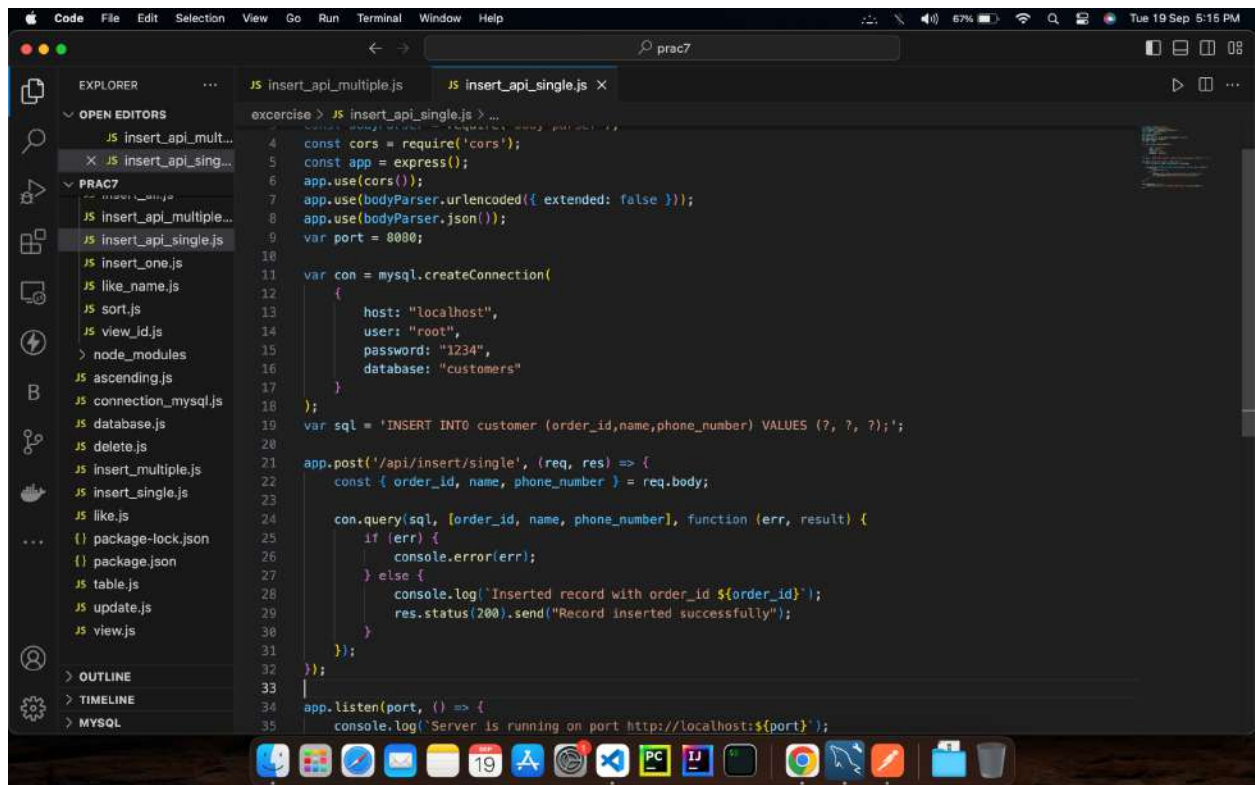


order_id	name	phone_number
1	shivam	1234567893
2	shivam2	1232435454
3	shivam3	4343234332
4	shivam4	4343432423
5	shivam5	2124435432
10	MANAV	1111121111
111	shivam_111	1111111111

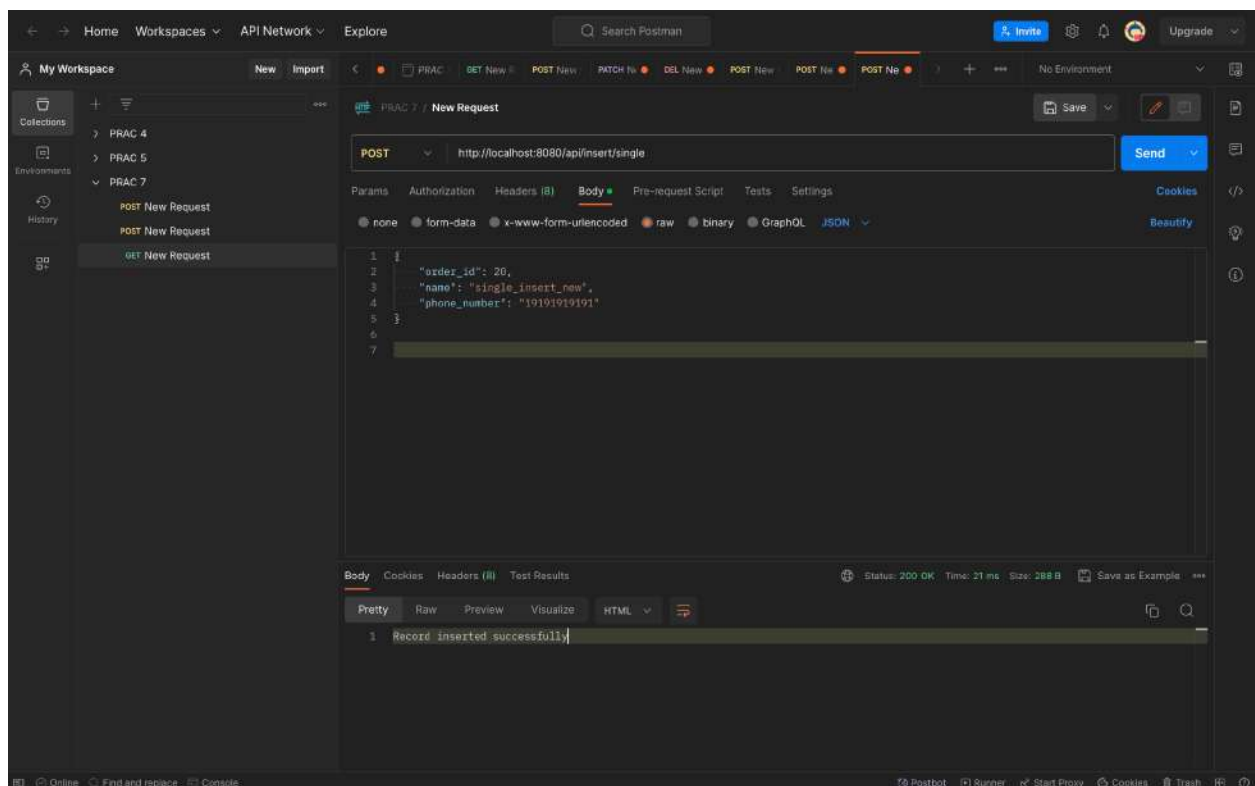
Object Info	Session	Time	Action	Response	Duration / Fetch Time
No object selected		16:17:07	select * from customer LIMIT 0, 1000	Error Code: 1048. No database selected Select the de...	0.00049 sec
		16:17:11	use customers	0 row(s) affected	0.00027 sec
		16:17:14	select * from customer LIMIT 0, 1000	7 row(s) returned	0.00072 sec / 0.0000...
		16:18:40	select * from customer order by name LIMIT 0, 1000	7 row(s) returned	0.00070 sec / 0.0000...
		16:23:45	select * from customer LIMIT 0, 1000	7 row(s) returned	0.00049 sec / 0.000...

## Practical 7.7: Create an API to post customer records.

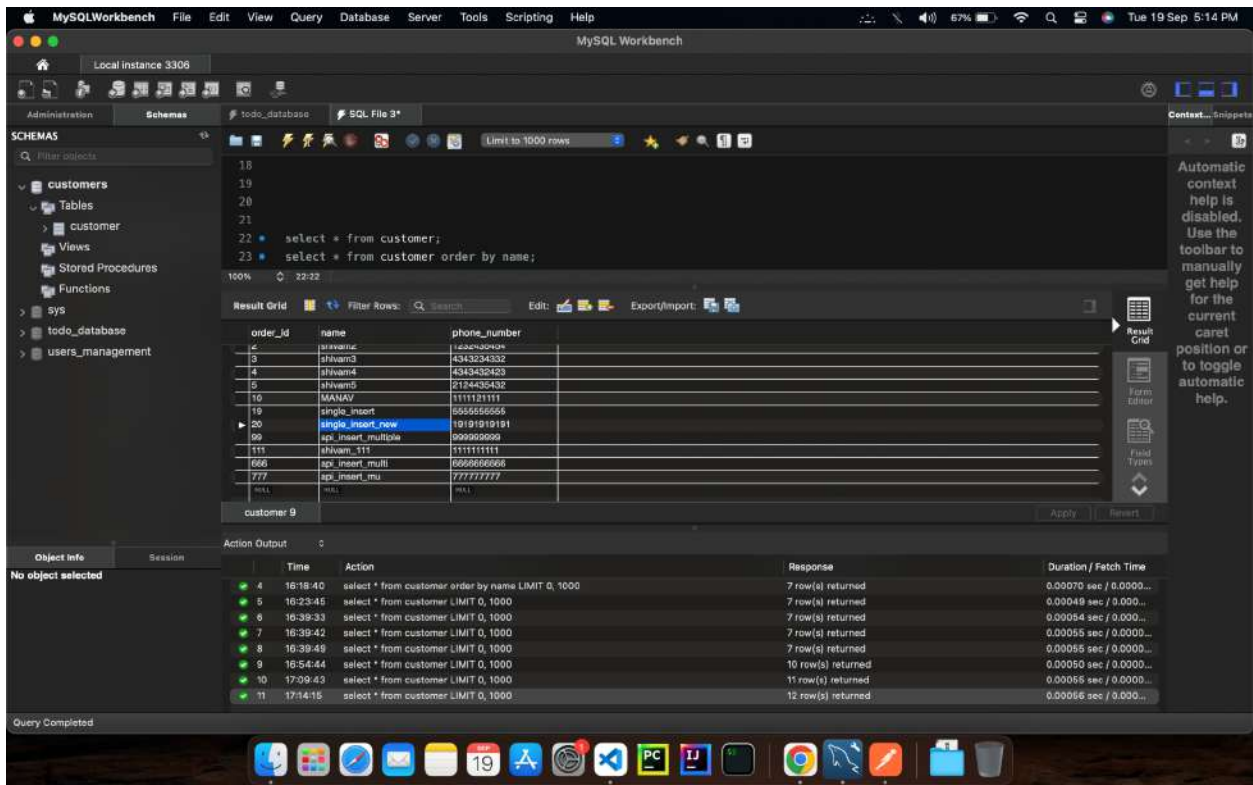
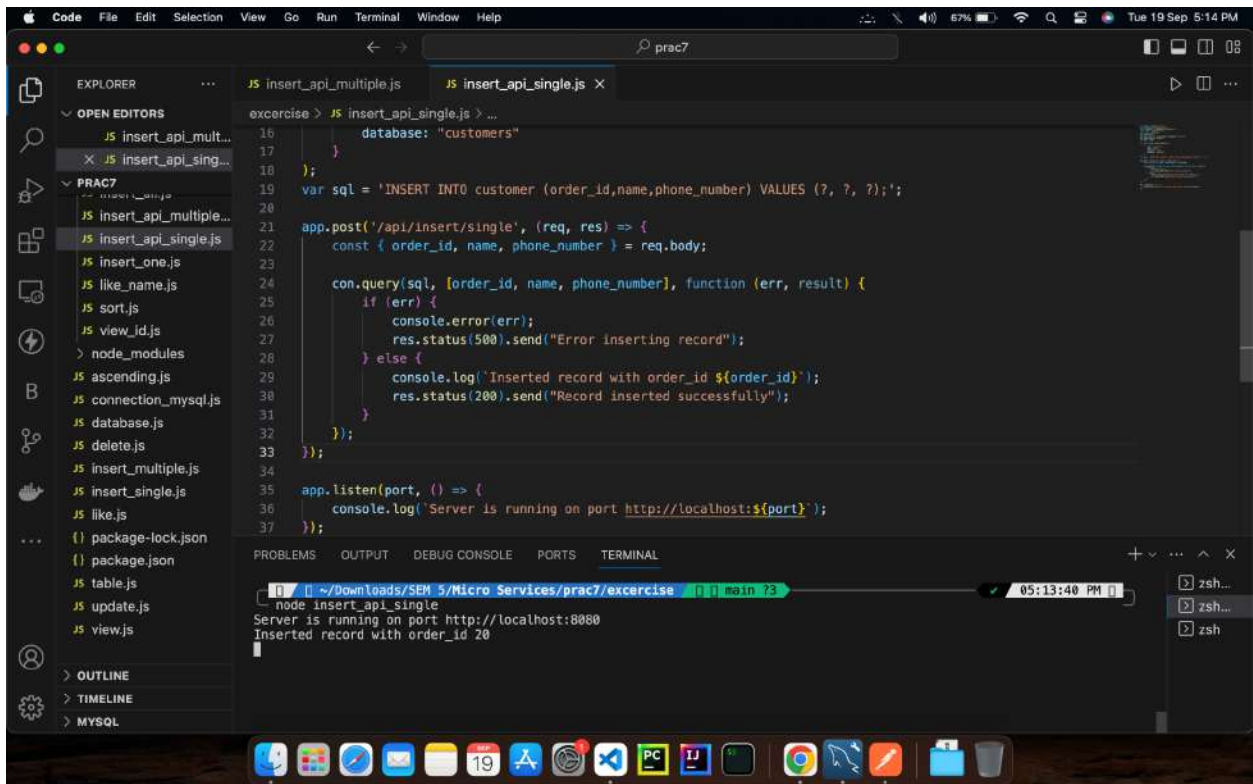
**CODE & OUTPUT: → By Single data post**



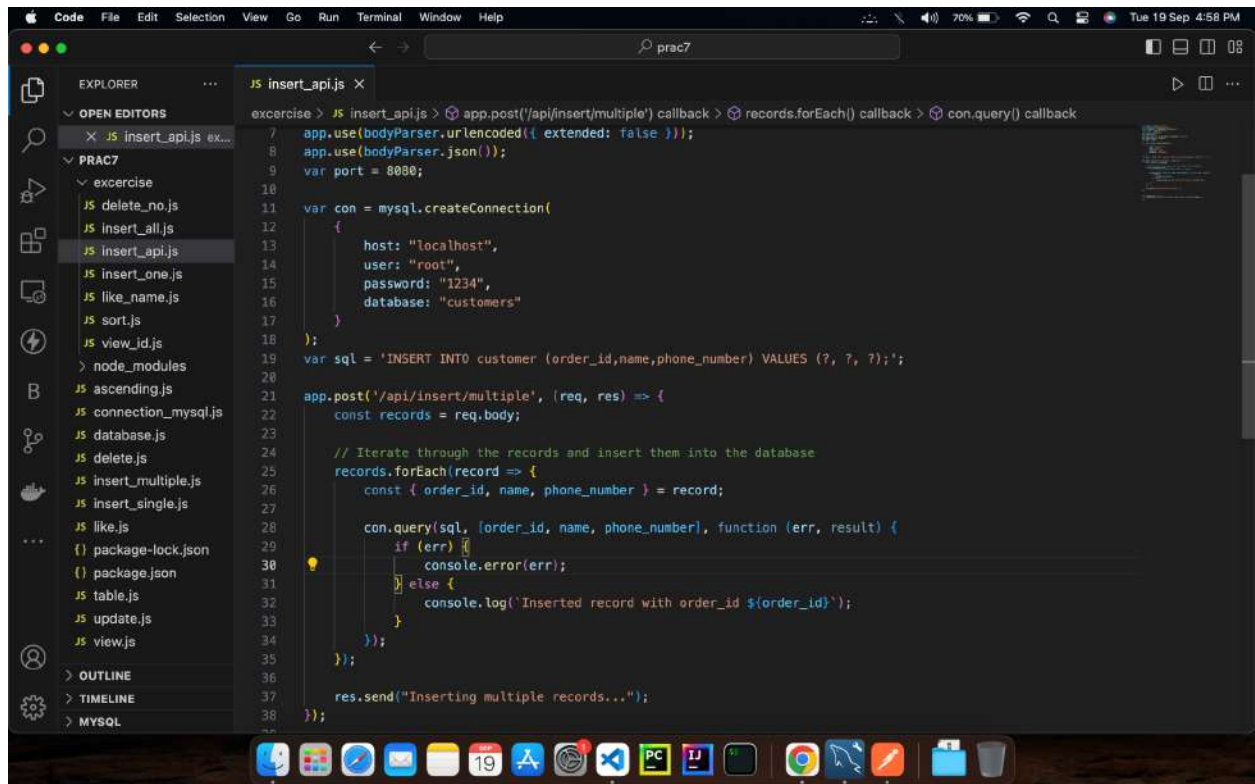
```
exercise > JS insert_api_single.js > ...
4  const cors = require('cors');
5  const app = express();
6  app.use(cors());
7  app.use(bodyParser.urlencoded({ extended: false }));
8  app.use(bodyParser.json());
9  var port = 8080;
10
11  var con = mysql.createConnection(
12    {
13      host: "localhost",
14      user: "root",
15      password: "1234",
16      database: "customers"
17    }
18  );
19  var sql = 'INSERT INTO customer (order_id,name,phone_number) VALUES (?, ?, ?)';
20
21  app.post('/api/insert/single', (req, res) => {
22    const { order_id, name, phone_number } = req.body;
23
24    con.query(sql, [order_id, name, phone_number], function (err, result) {
25      if (err) {
26        console.error(err);
27      } else {
28        console.log('Inserted record with order_id ${order_id}');
29        res.status(200).send("Record inserted successfully");
30      }
31    });
32  });
33
34  app.listen(port, () => {
35    console.log(`Server is running on port http://localhost:${port}`);
36  });
```





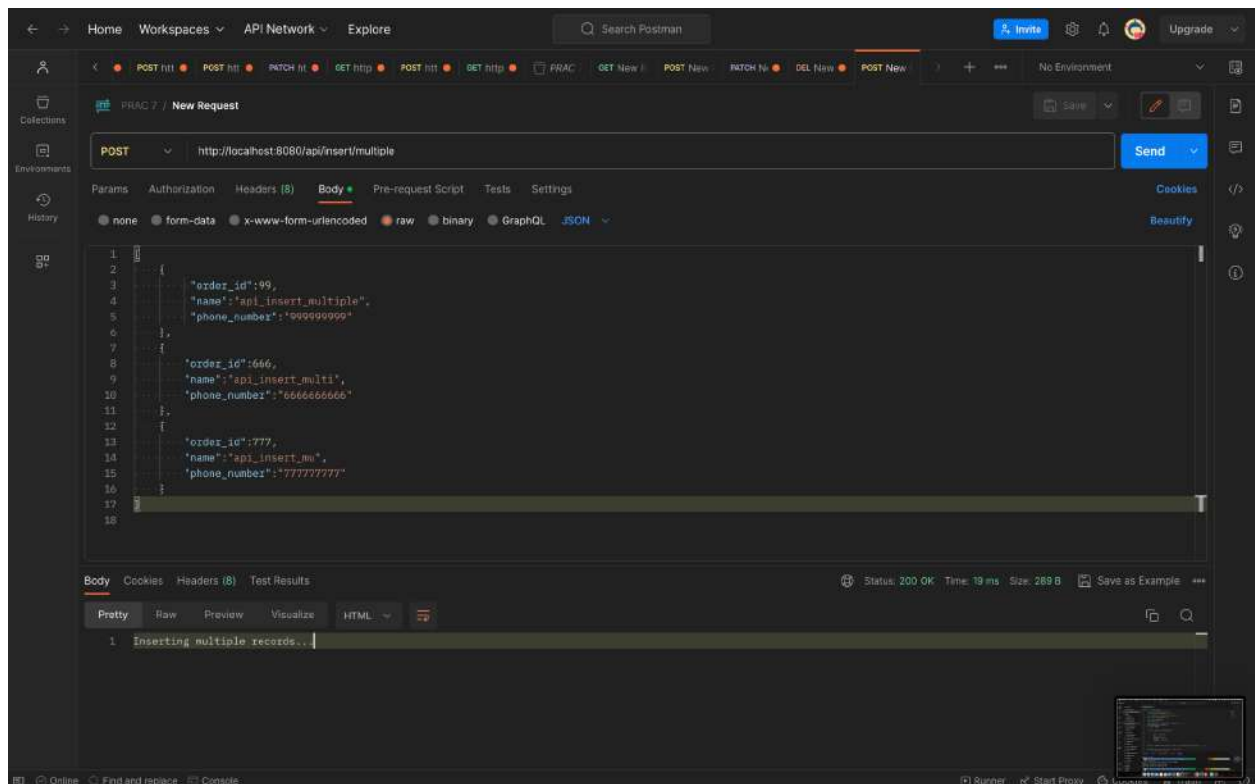


## → By Multiple data post



The screenshot shows a VS Code editor with a file explorer on the left and a code editor in the center. The file explorer shows a project named 'PRAC7' with several JavaScript files. The code editor displays the 'insert\_api.js' file, which contains the following code:

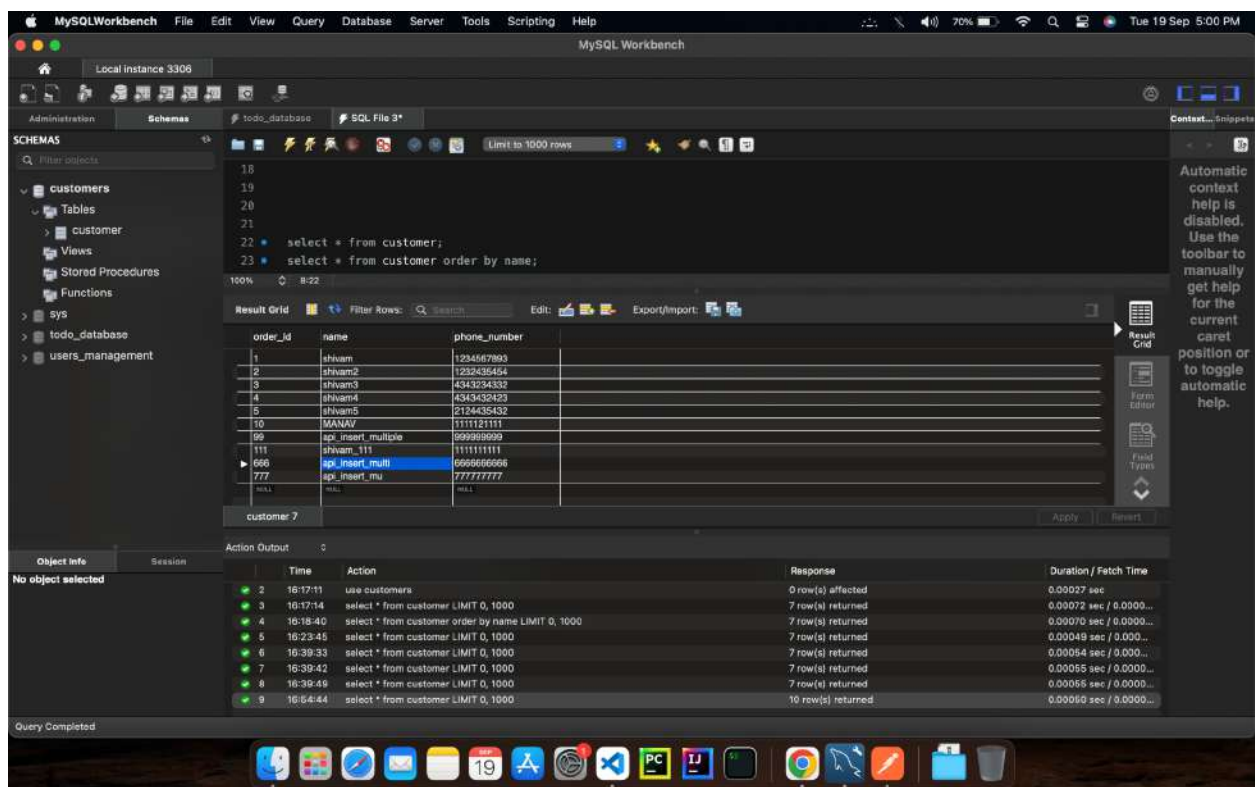
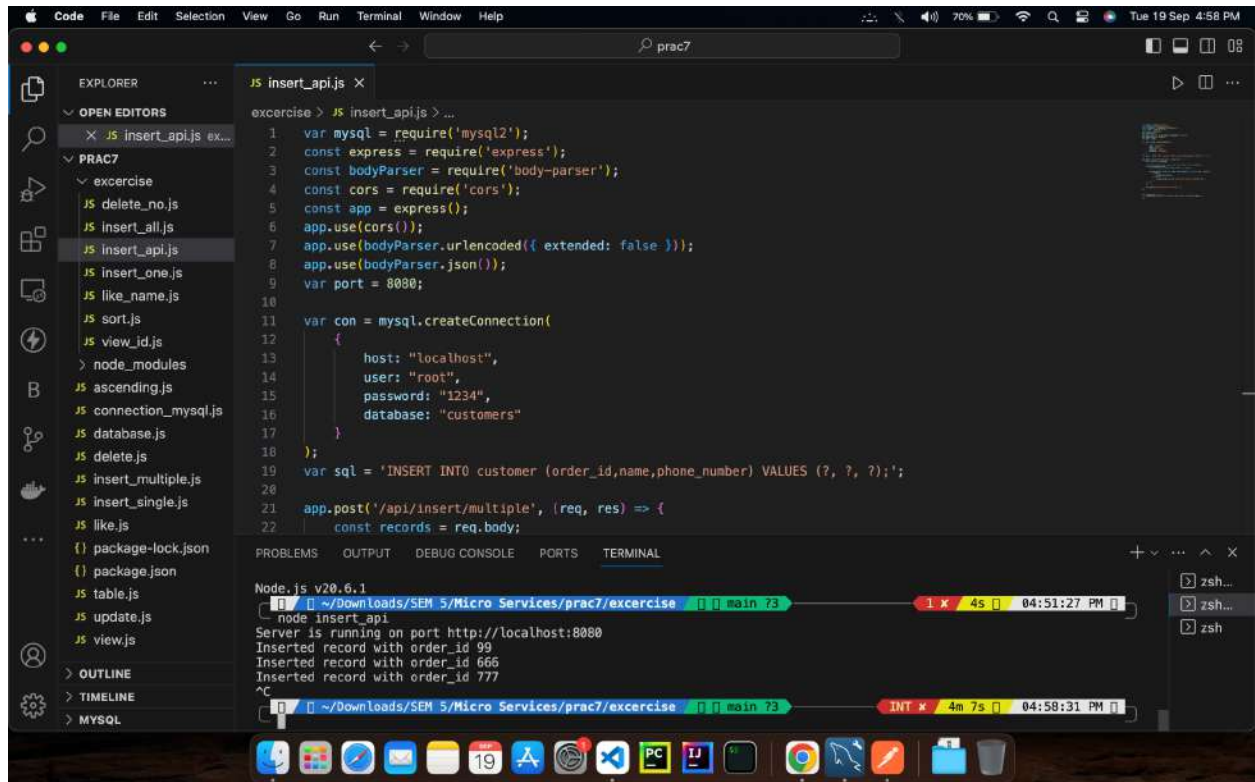
```
exercice > JS insert_api.js > app.post('/api/insert/multiple') callback > records.forEach() callback > con.query() callback
7 app.use(bodyParser.urlencoded({ extended: false }));
8 app.use(bodyParser.json());
9 var port = 8080;
10
11 var con = mysql.createConnection(
12 {
13   host: "localhost",
14   user: "root",
15   password: "1234",
16   database: "customers"
17 }
18 );
19 var sql = 'INSERT INTO customer (order_id,name,phone_number) VALUES (?, ?, ?)';
20
21 app.post('/api/insert/multiple', (req, res) => {
22   const records = req.body;
23
24   // Iterate through the records and insert them into the database
25   records.forEach(record => {
26     const { order_id, name, phone_number } = record;
27
28     con.query(sql, [order_id, name, phone_number], function (err, result) {
29       if (err) {
30         console.error(err);
31       } else {
32         console.log('Inserted record with order_id ${order_id}');
33       }
34     });
35   });
36
37   res.send("Inserting multiple records...");
38 });
```



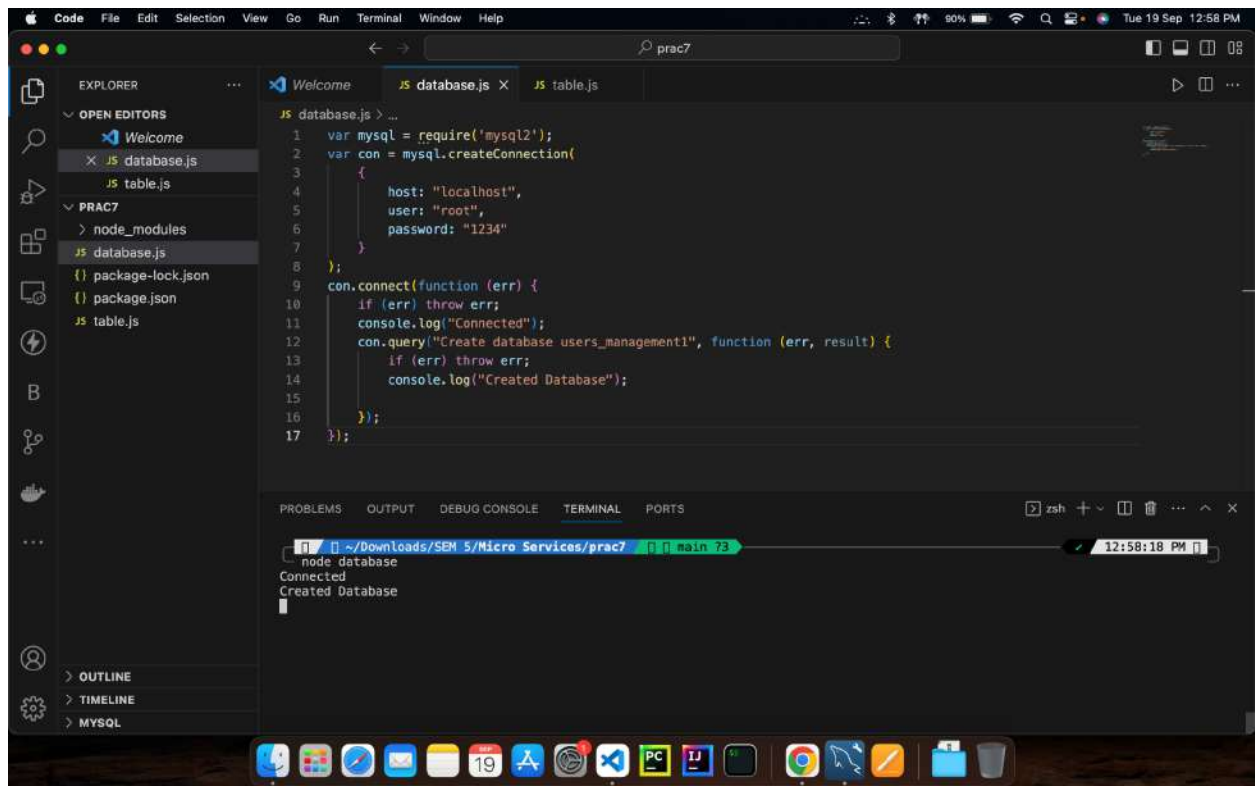
The screenshot shows the Postman application interface. A POST request is configured with the URL 'http://localhost:8080/api/insert/multiple'. The request body is set to 'raw' and contains the following JSON data:

```
1 {
2   "order_id": 99,
3   "name": "api_insert_multiple",
4   "phone_number": "999999999"
5 },
6 {
7   "order_id": 666,
8   "name": "api_insert_multi",
9   "phone_number": "666666666"
10 },
11 {
12   "order_id": 777,
13   "name": "api_insert_mn",
14   "phone_number": "777777777"
15 }
16
17
18
```

The response status is 200 OK, and the response body is 'Inserting multiple records...'. The interface also shows tabs for Params, Authorization, Headers (8), Body, Pre-request Script, Tests, and Settings.



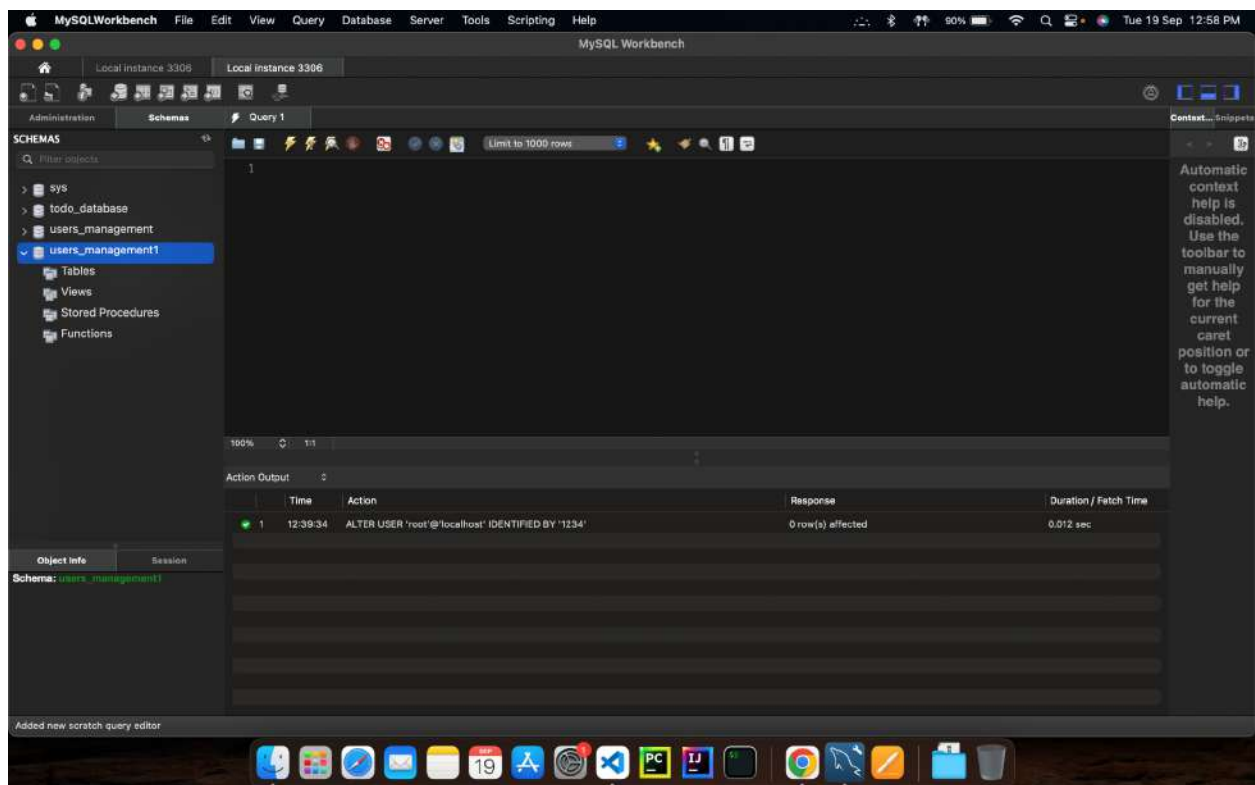
## DURING LAB:



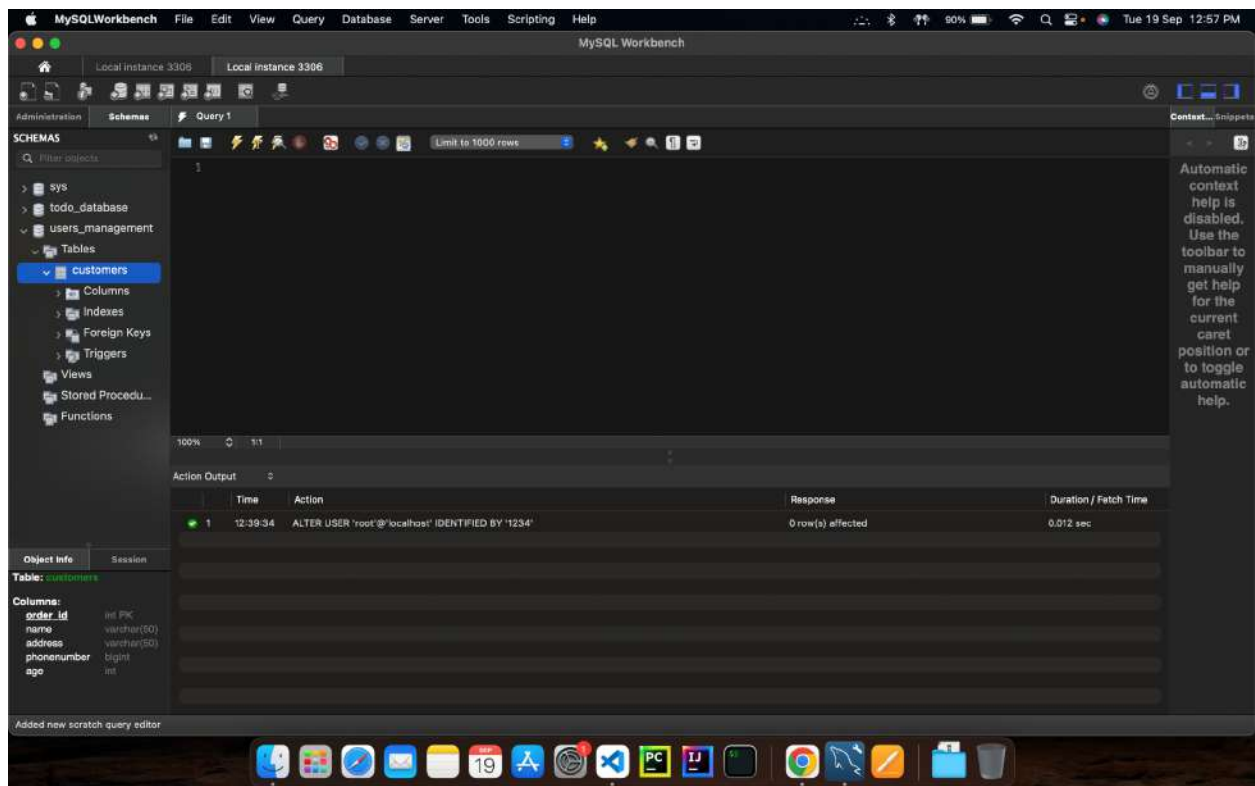
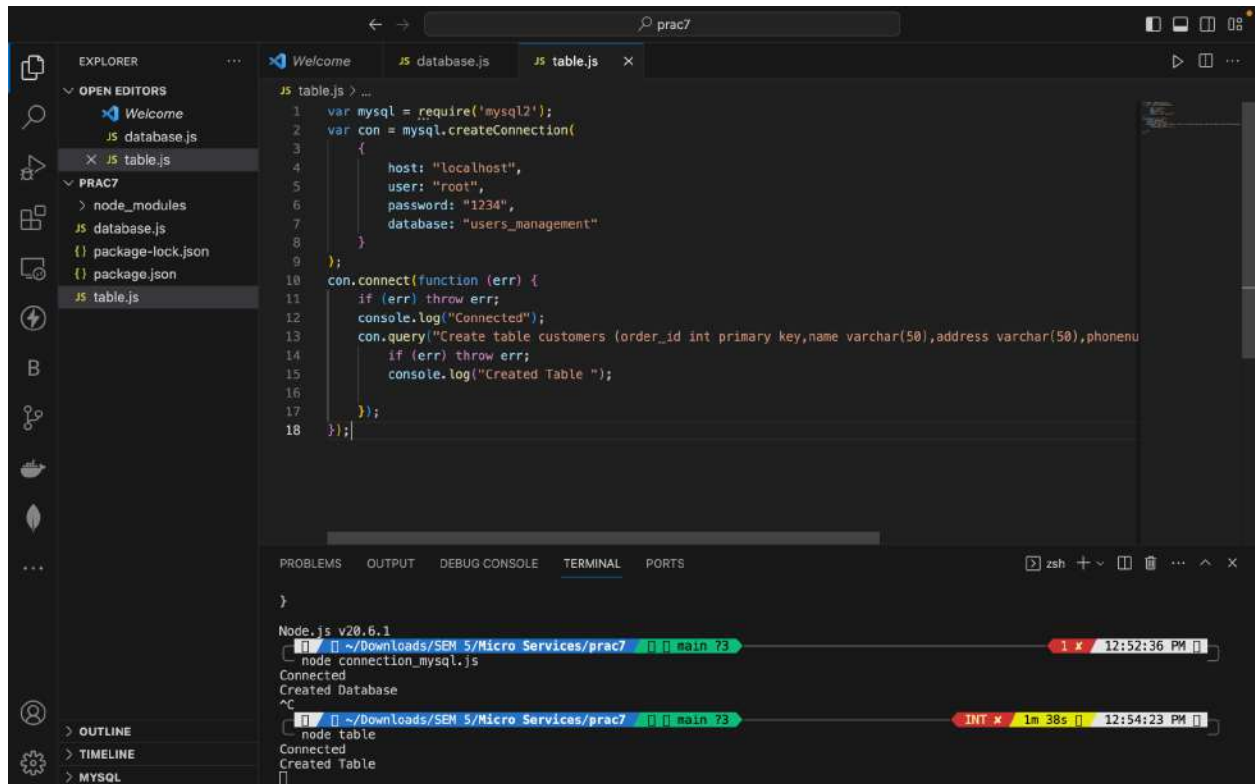
The screenshot shows the Visual Studio Code editor interface. The Explorer sidebar on the left displays the project structure for 'prac7', including 'node\_modules', 'package-lock.json', 'package.json', and 'table.js'. The main editor window shows the 'database.js' file with the following JavaScript code:

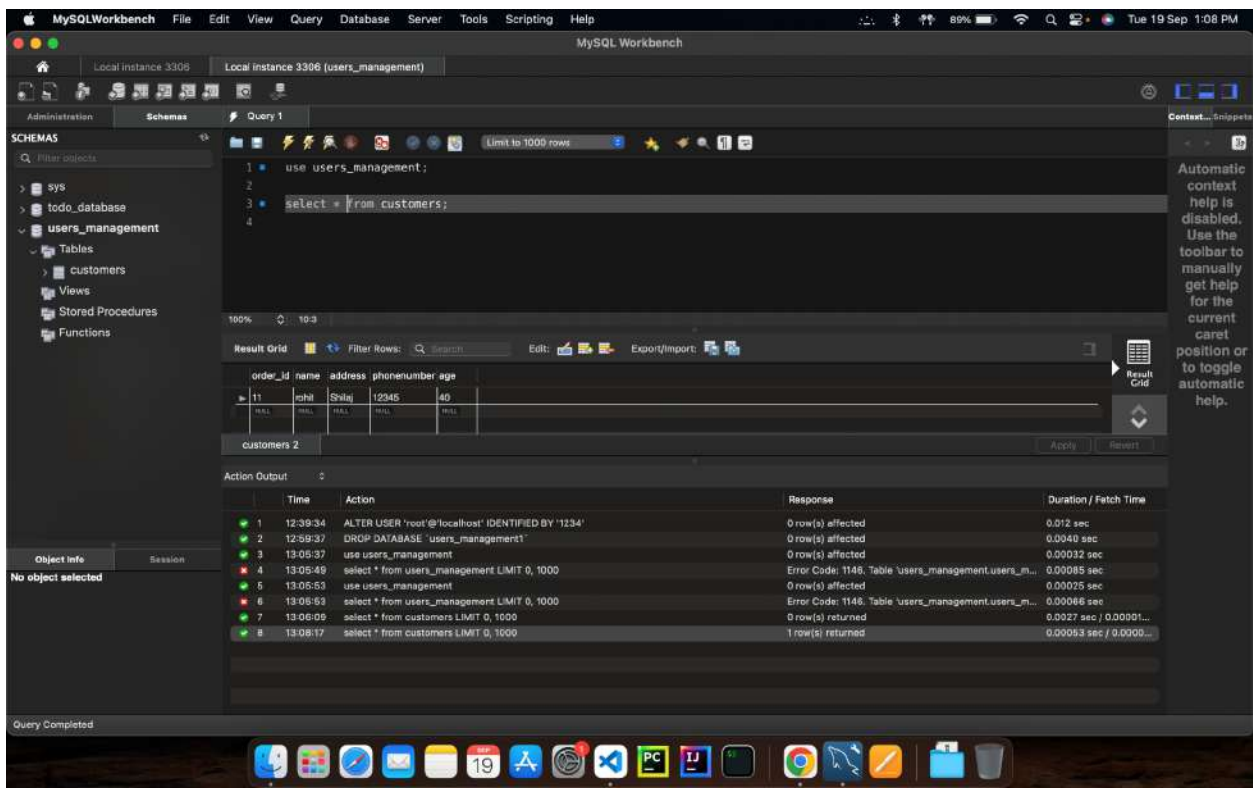
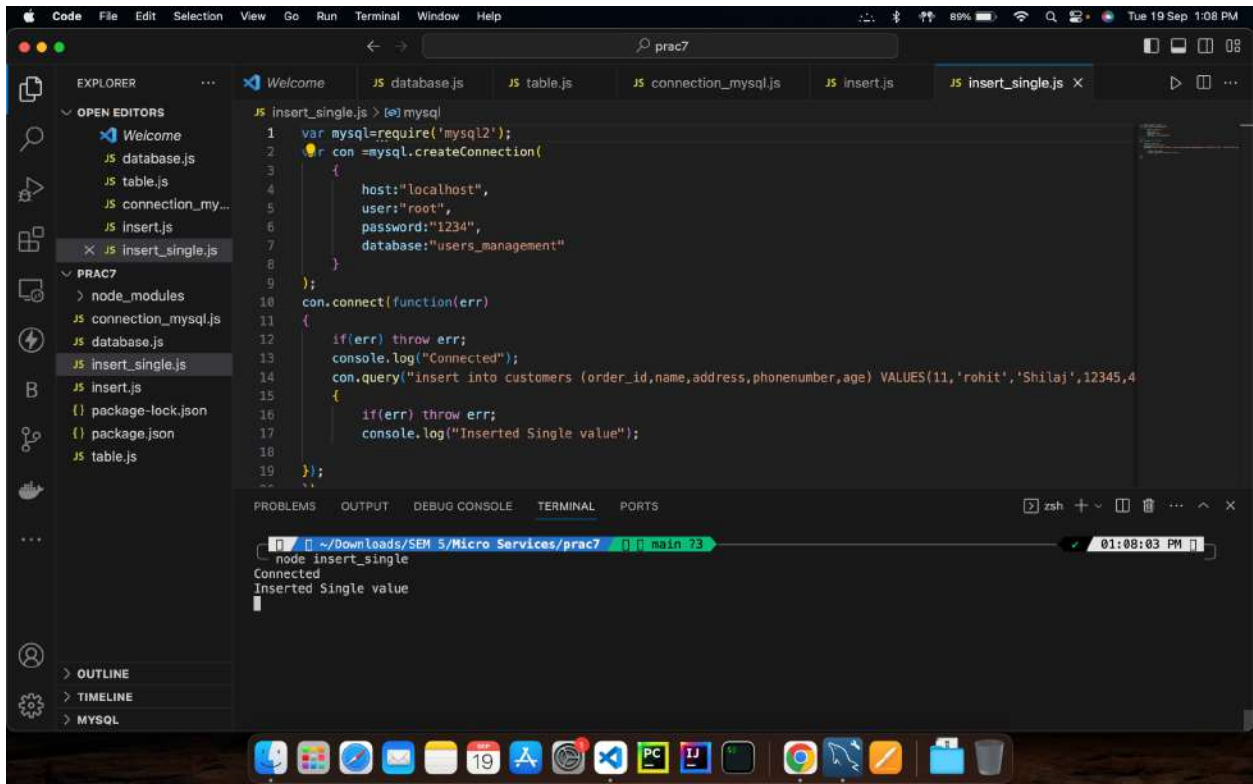
```
1 var mysql = require('mysql2');
2 var con = mysql.createConnection({
3   {
4     host: "localhost",
5     user: "root",
6     password: "1234"
7   }
8 });
9 con.connect(function (err) {
10   if (err) throw err;
11   console.log("Connected");
12   con.query("Create database users_management1", function (err, result) {
13     if (err) throw err;
14     console.log("Created Database");
15   });
16 });
17 };
```

Below the editor, the TERMINAL panel shows the output of running 'node database', displaying 'Connected' and 'Created Database'.









The screenshot shows the Visual Studio Code editor with a project named 'prac7'. The Explorer sidebar on the left lists files: 'Welcome', 'JS database.js', 'JS table.js', 'JS connection\_mysql.js', 'JS insert\_multiple.js' (selected), and 'JS insert\_single.js'. The main editor displays the code in 'insert\_multiple.js':

```
1 var mysql = require('mysql2');
2 var con = mysql.createConnection({
3   {
4     host: "localhost",
5     user: "root",
6     password: "1234",
7     database: "users_management"
8   }
9 });
10 con.connect(function (err) {
11   if (err) throw err;
12   console.log("Connected");
13   con.query("insert into customers (order_id,name,address,phonenumber,age) VALUES(2,'vikas','chandigarh',1232
14   if (err) throw err;
15   console.log("Inserted multiple values");
16 });
17 });
18 });
```

The bottom panel shows the 'TERMINAL' with the following output:

```
node insert_single
Connected
Inserted Single value
^C
node insert_multiple
Connected
Inserted multiple values
```

The terminal status bar indicates 'INT x' and '1m 22s'.

The screenshot shows the MySQL Workbench interface. The 'SCHEMAS' sidebar on the left shows the 'users\_management' database selected. The 'Query 1' editor contains the following SQL:

```
1 use users_management;
2
3 select * from customers;
4
```

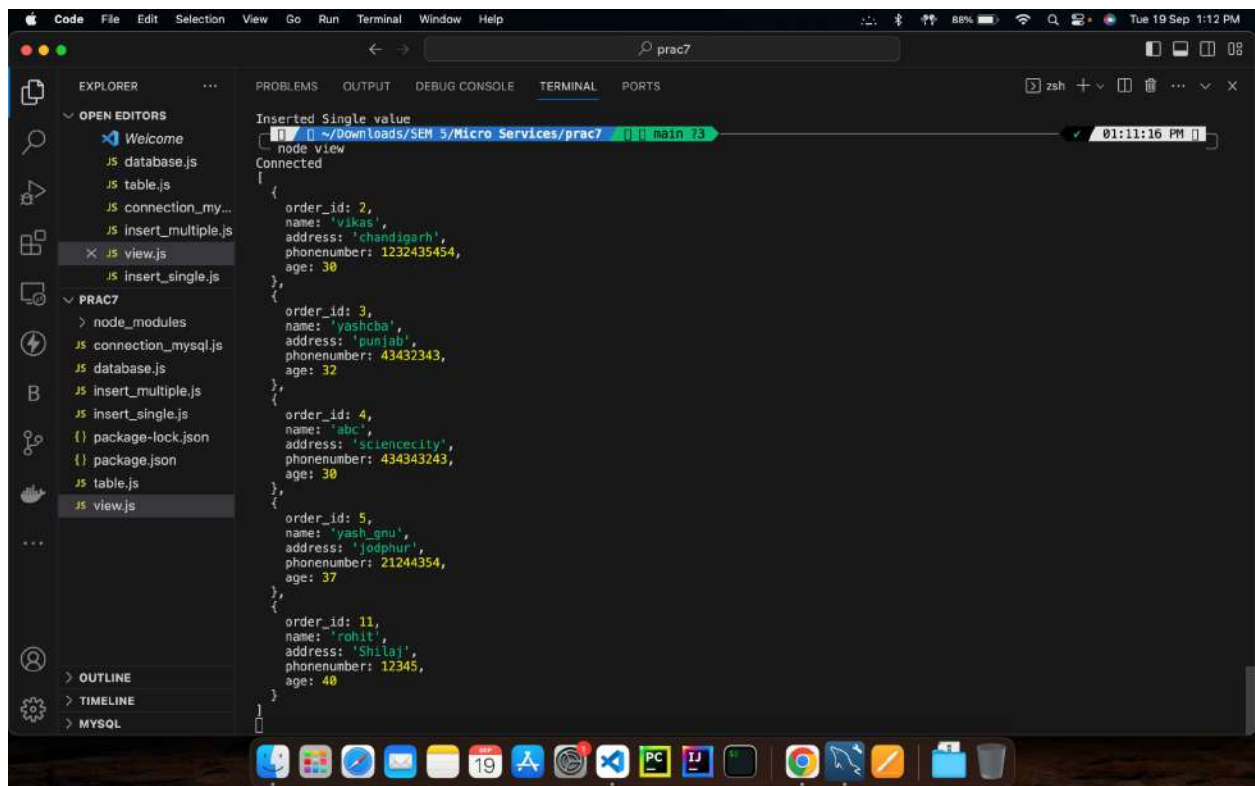
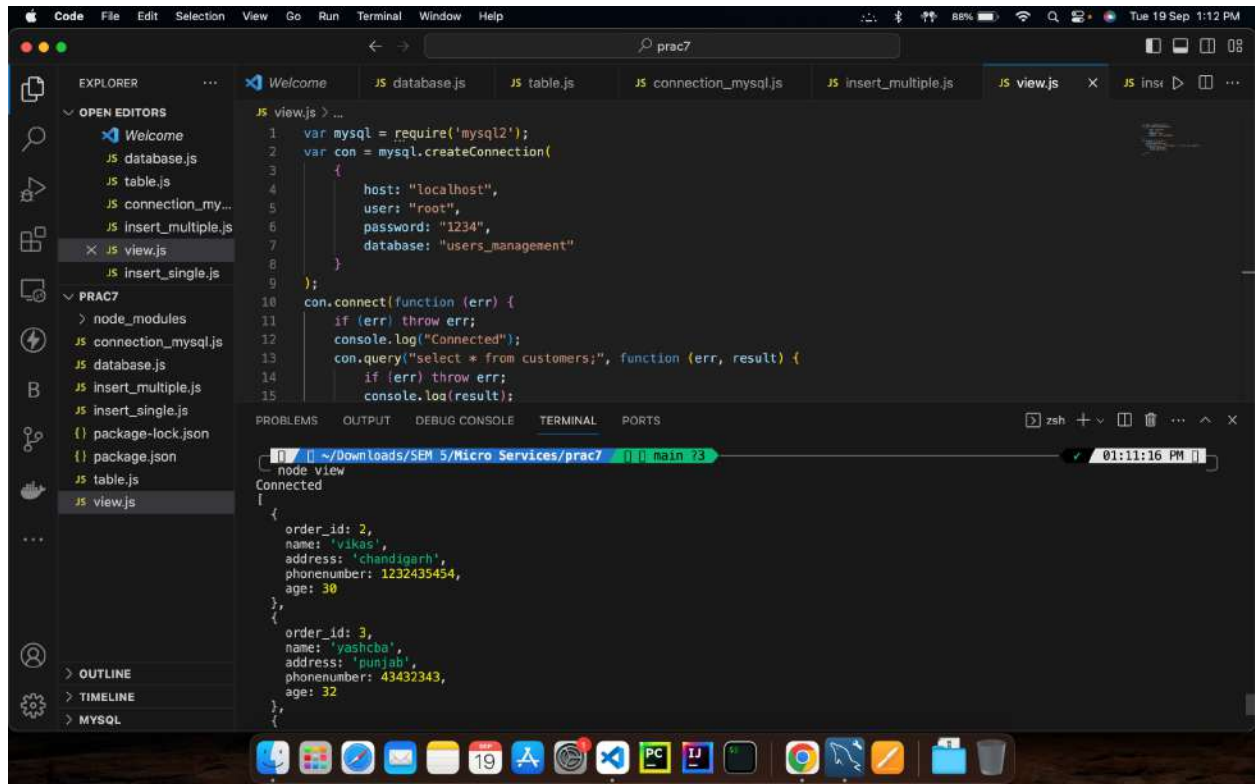
The 'Result Grid' displays the following data:

order_id	name	address	phonenumber	age
2	vikas	chandigarh	1232435454	30
3	yashba	punjab	43432343	30
4	abc	sciencecity	434343243	30
6	yash_gnu	podhpur	21244354	37
11	rohi	Shilaj	12545	40
NULL	NULL	NULL	NULL	NULL

The 'Object Info' sidebar shows 'customers 3'. The 'Action Output' sidebar shows the following log:

Time	Action	Response	Duration / Fetch Time
12:39:34	ALTER USER 'root'@'localhost' IDENTIFIED BY '1234'	0 row(s) affected	0.012 sec
12:59:37	DROP DATABASE 'users_management'	0 row(s) affected	0.0040 sec
13:05:37	use users_management	0 row(s) affected	0.00032 sec
13:05:49	select * from users_management LIMIT 0, 1000	Error Code: 1146. Table 'users_management.users_m...	0.00085 sec
13:05:53	use users_management	0 row(s) affected	0.00025 sec
13:05:53	select * from users_management LIMIT 0, 1000	Error Code: 1146. Table 'users_management.users_m...	0.00066 sec

The 'Query Completed' status is shown at the bottom.





```
JS update.js > con.connect() callback
1 var mysql=require('mysql2');
2 var con =mysql.createConnection(
3   {
4     host:"localhost",
5     user:"root",
6     password:"1234",
7     database:"users_management"
8   }
9 );
10 con.connect(function(err)
11 {
12   if(err) throw err;
13   console.log("Connected");
14   con.query("UPDATE customers SET name = 'shivam' WHERE order_id = 2;",function (err,result )
15   {
16     if(err) throw err;
17     console.log("Table Updated");
18   });
19 });
20 });
```

node update  
Connected  
Table Updated

MySQL Workbench

Local instance 3308 Local instance 3306 (users\_management)

Administration Schemas Query 1

SCHEMAS

- sys
- todo\_database
- users\_management
  - Tables
    - customers
  - Views
  - Stored Procedures
  - Functions

100% 1:3

Result Grid

order_id	name	address	phonenumber	age
2	shivam	chandigarh	1223435454	30
3	yashika	punjab	43432343	33
4	abc	sciencecity	434343243	30
6	yash_gnu	poddhar	21244354	37
11	rohi	Shilaj	12545	40
TOTAL	NULL	NULL	NULL	NULL

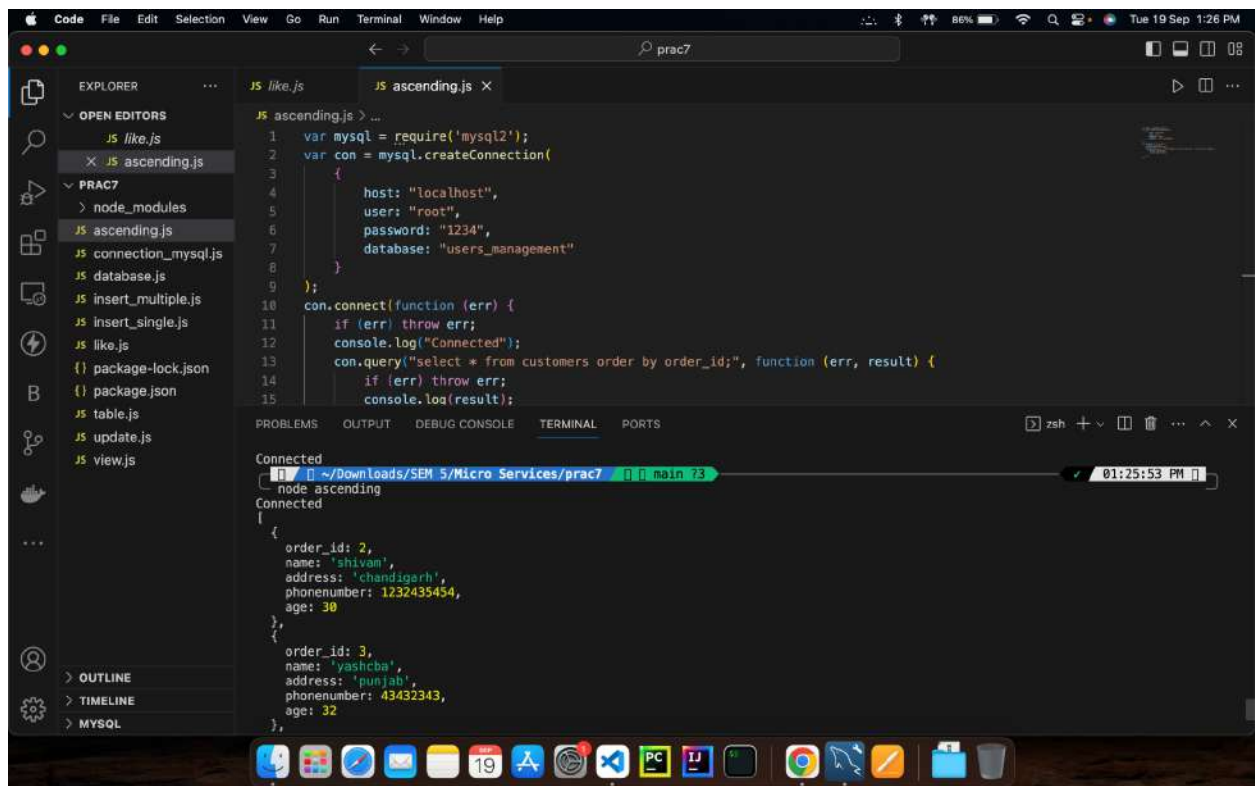
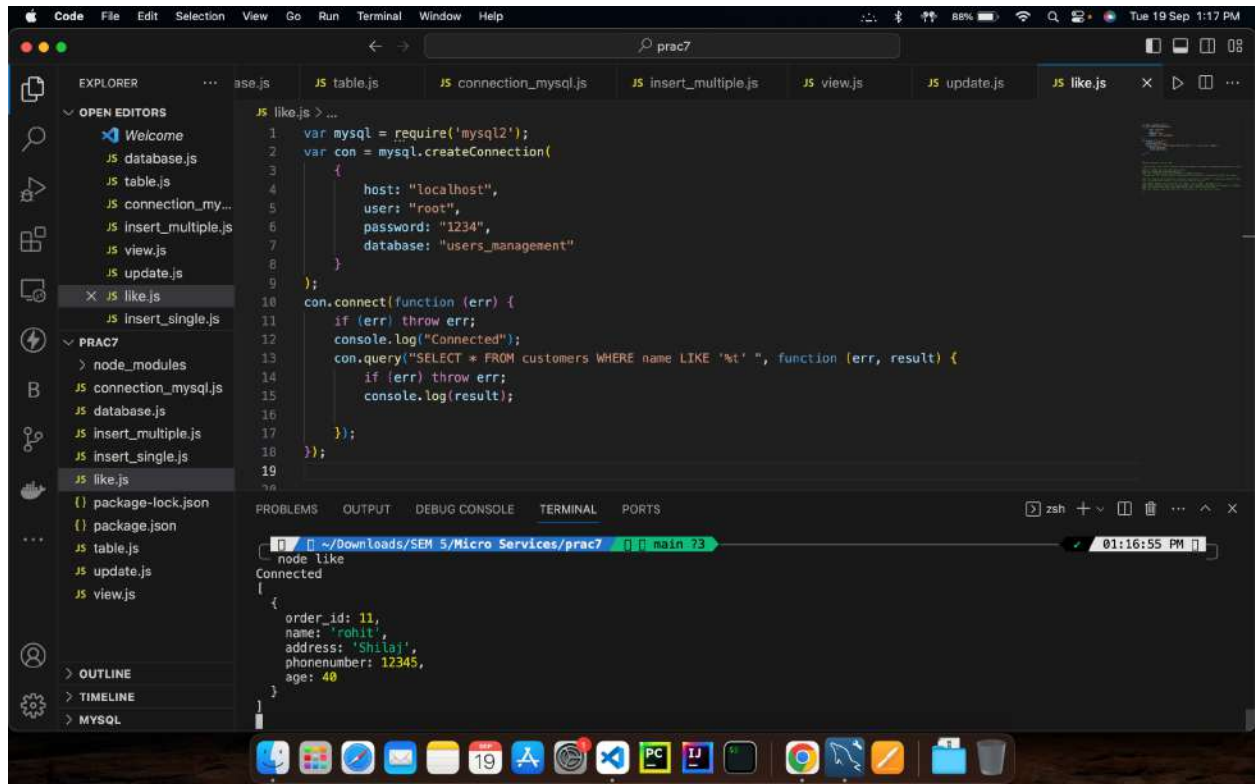
Object Info Session customers 4

No object selected

Action Output

Time	Action	Response	Duration / Fetch Time
13:05:53	select * from users_management LIMIT 0, 1000	Error Code: 1145, Table 'users_management.users_m...	0.00066 sec
13:06:09	select * from customers LIMIT 0, 1000	0 row(s) returned	0.0027 sec / 0.00001...
13:08:17	select * from customers LIMIT 0, 1000	1 row(s) returned	0.00053 sec / 0.0000...
13:09:49	select * from customers LIMIT 0, 1000	5 row(s) returned	0.00048 sec / 0.000...
13:13:55	select * from customers LIMIT 0, 1000	5 row(s) returned	0.00066 sec / 0.0000...

Query Completed



The screenshot shows the VS Code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'prac7' with several JavaScript files. The 'delete.js' file is open in the editor. The code in 'delete.js' is as follows:

```
1 var mysql = require('mysql2');
2 var con = mysql.createConnection({
3   host: "localhost",
4   user: "root",
5   password: "1234",
6   database: "users_management"
7 });
8
9 con.connect(function (err) {
10   if (err) throw err;
11   console.log("Connected");
12   con.query("delete from customers WHERE phonenum = 12345;", function (err, result) {
13     if (err) throw err;
14     console.log("Record Deleted");
15   });
16 });
```

The terminal at the bottom shows the output of the script:

```
age: 37
{
  order_id: 11,
  name: 'rohit',
  address: 'Shilaj',
  phonenum: 12345,
  age: 40
}
```

The terminal also shows the command 'node delete' and the output 'Connected' and 'Record Deleted'.

The screenshot shows the MySQL Workbench interface. The 'Query 1' tab is active, showing a query that selects all records from the 'customers' table. The query is:

```
1 use users_management;
2
3 select * from customers;
4
```

The 'Result Grid' shows the following data:

order_id	name	address	phonenum	age
2	shivam	chandigarh	1223435454	30
3	yashcha	punjab	43432343	30
4	abc	sciencecity	434343243	30
5	yash_gnu	pdphur	21244354	37
NULL	NULL	NULL	NULL	NULL

The 'Object Info' tab shows the 'customers' table with 5 rows. The 'Action Output' tab shows the results of the query execution:

Time	Action	Response	Duration / Fetch Time
13:06:09	select * from customers LIMIT 0, 1000	0 row(s) returned	0.0027 sec / 0.0001...
13:08:17	select * from customers LIMIT 0, 1000	1 row(s) returned	0.00053 sec / 0.0000...
13:09:49	select * from customers LIMIT 0, 1000	5 row(s) returned	0.00048 sec / 0.0000...
13:13:55	select * from customers LIMIT 0, 1000	5 row(s) returned	0.00055 sec / 0.0000...
13:27:25	select * from customers LIMIT 0, 1000	4 row(s) returned	0.00046 sec / 0.0000...