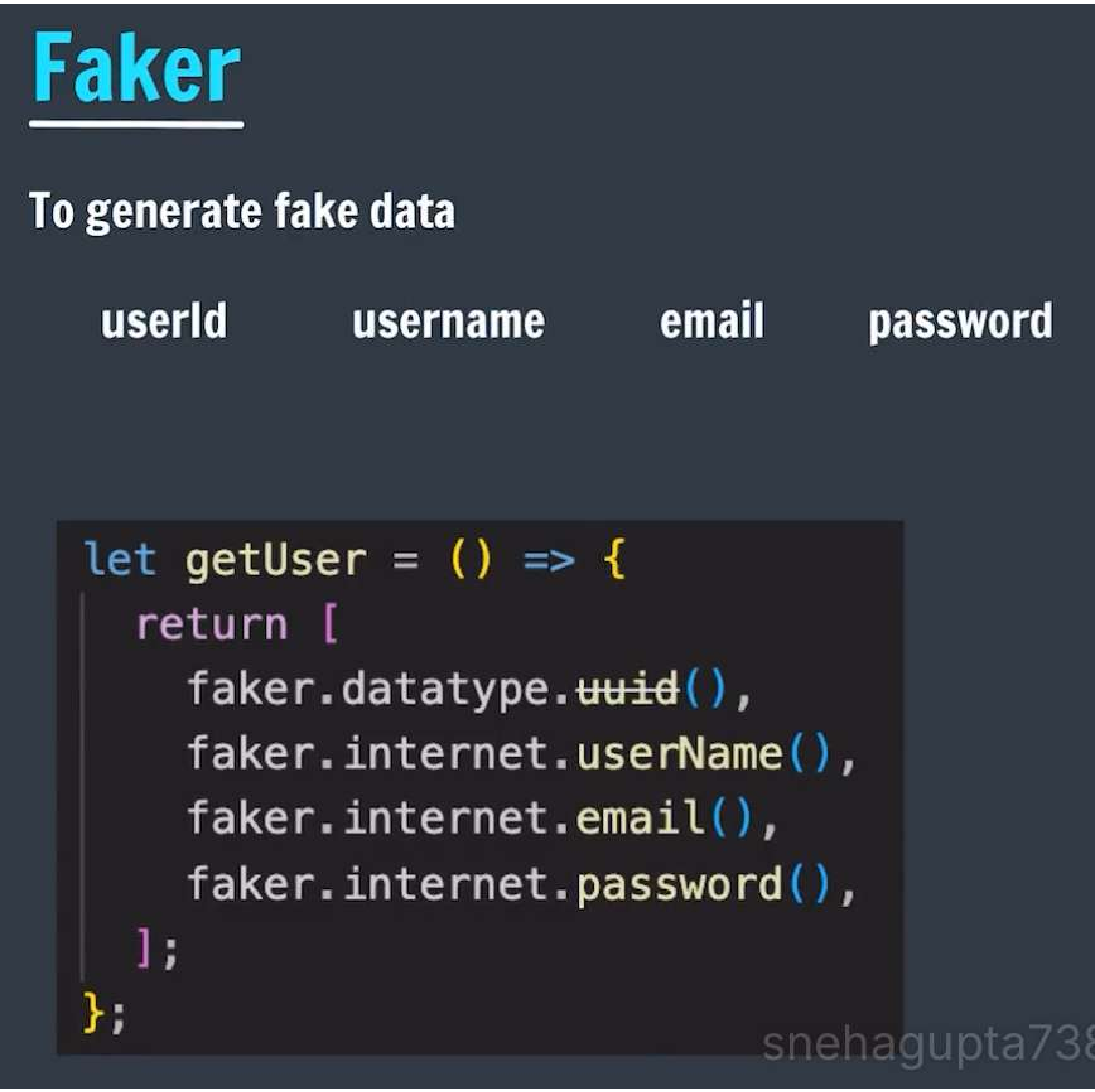
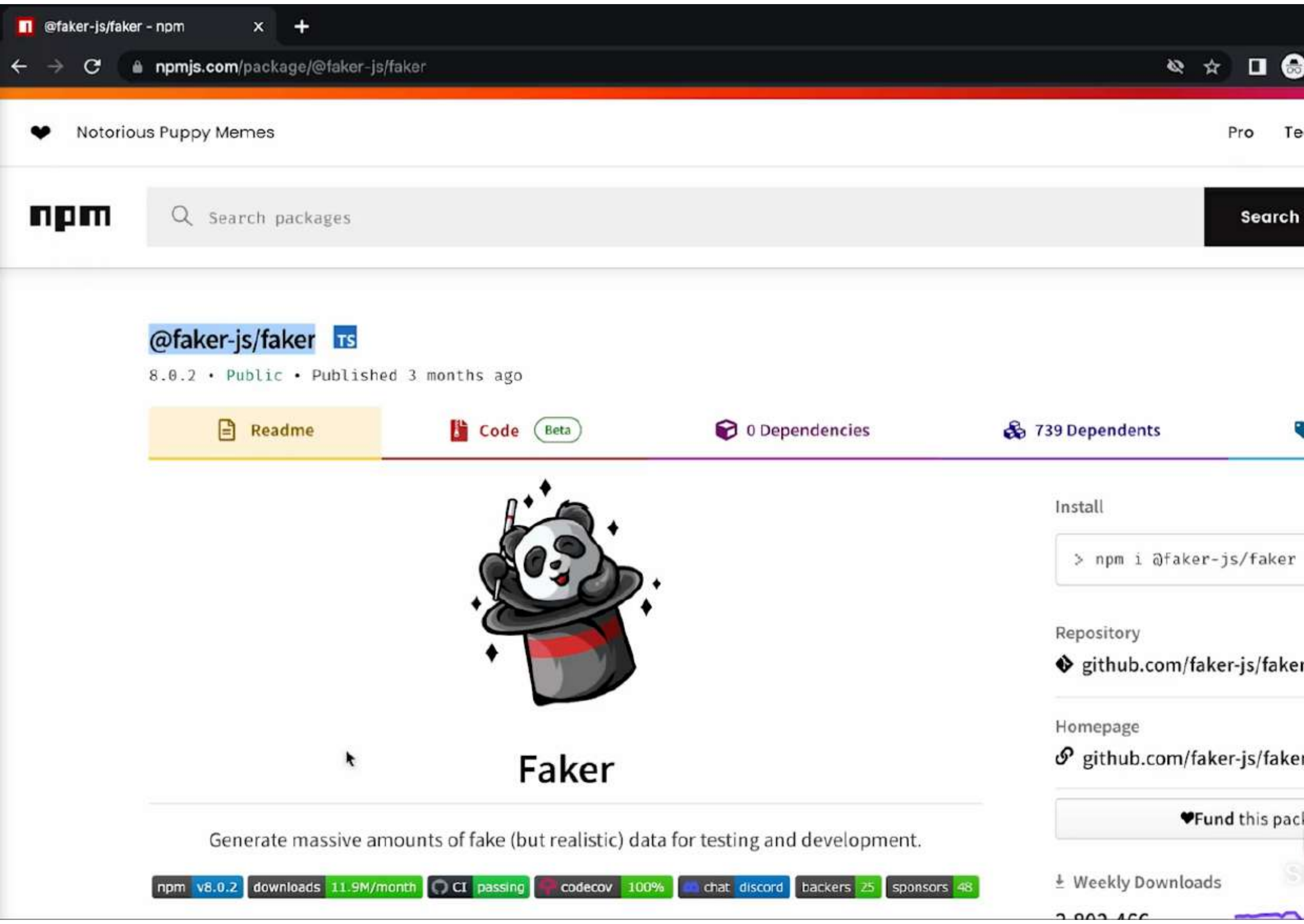


Faker

To generate fake data

userId username email password

```
let getUser = () => {
  return [
    faker.datatype.~~uuid~~(),
    faker.internet.userName(),
    faker.internet.email(),
    faker.internet.password(),
  ];
};
```



@faker-js/faker - npm

npmjs.com/package/@faker-js/faker

Notorious Puppy Memes

Pro Te

npm

Search packages

Search

@faker-js/faker TS

8.0.2 • Public • Published 3 months ago

Readme | Code Beta | 0 Dependencies | 739 Dependents

Faker

Generate massive amounts of fake (but realistic) data for testing and development.

npm v8.0.2 | downloads 11.9M/month | CI passing | codecov 100% | chat discord | backers 25 | sponsors 48

Install

```
> npm i @faker-js/faker
```

Repository

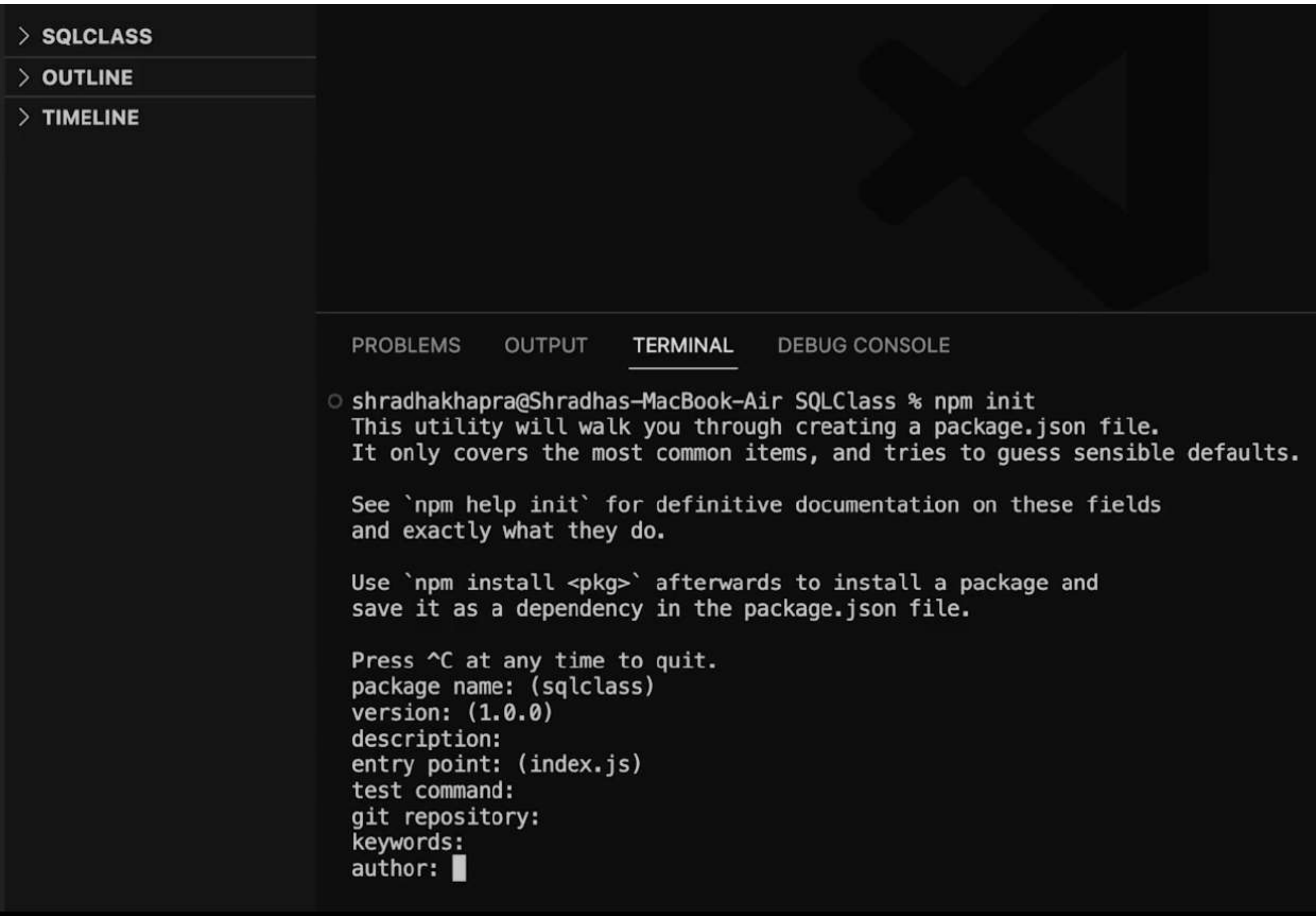
github.com/faker-js/faker

Homepage

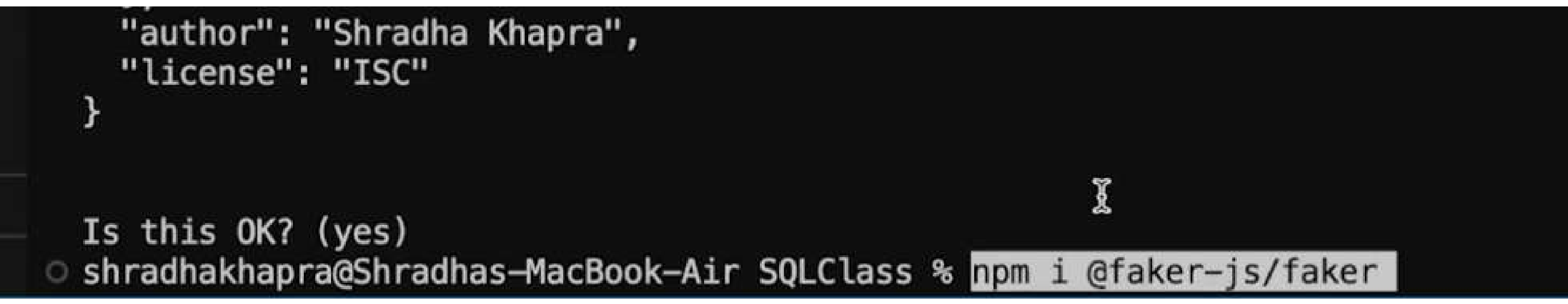
github.com/faker-js/faker

Fund this pac

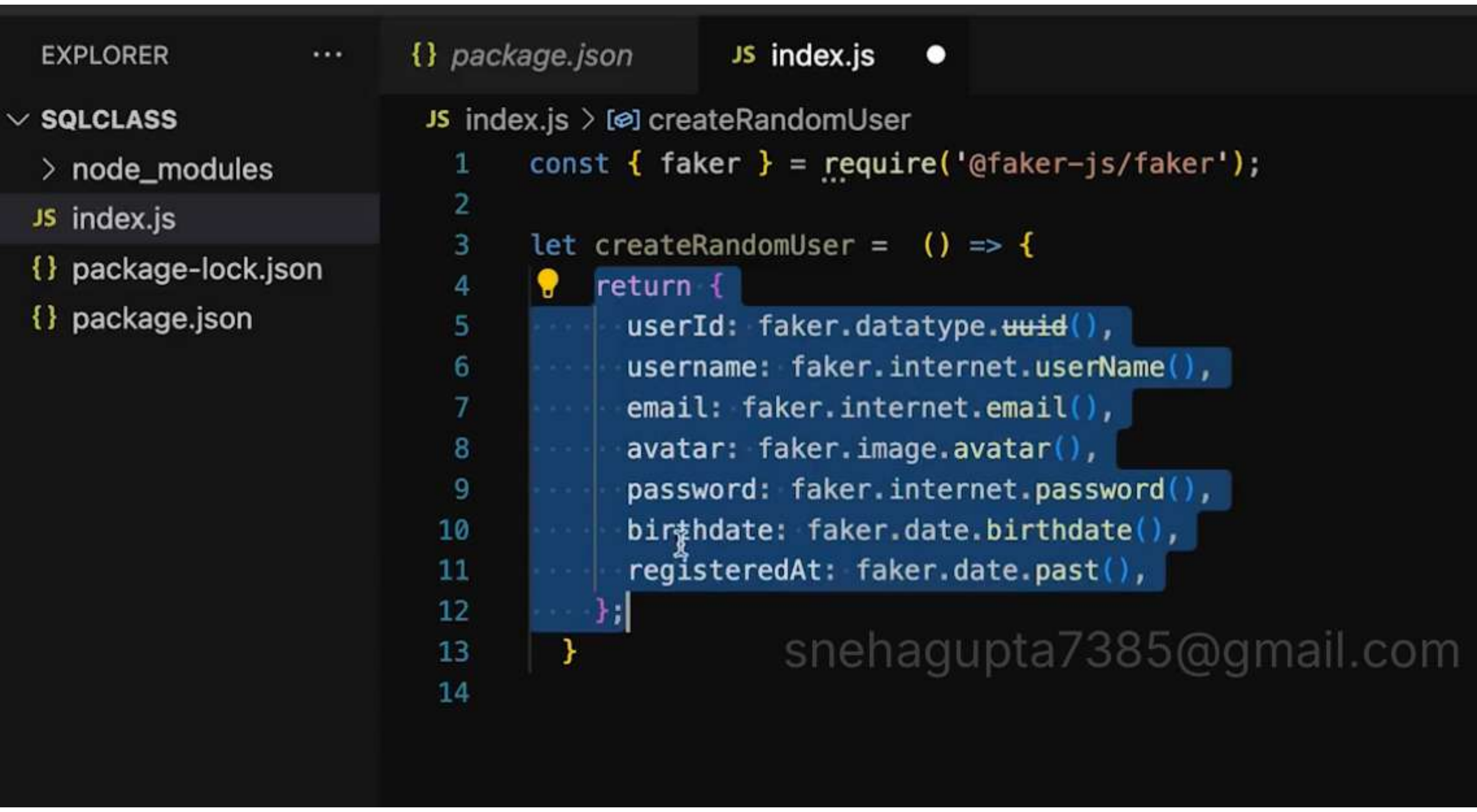
Weekly Downloads



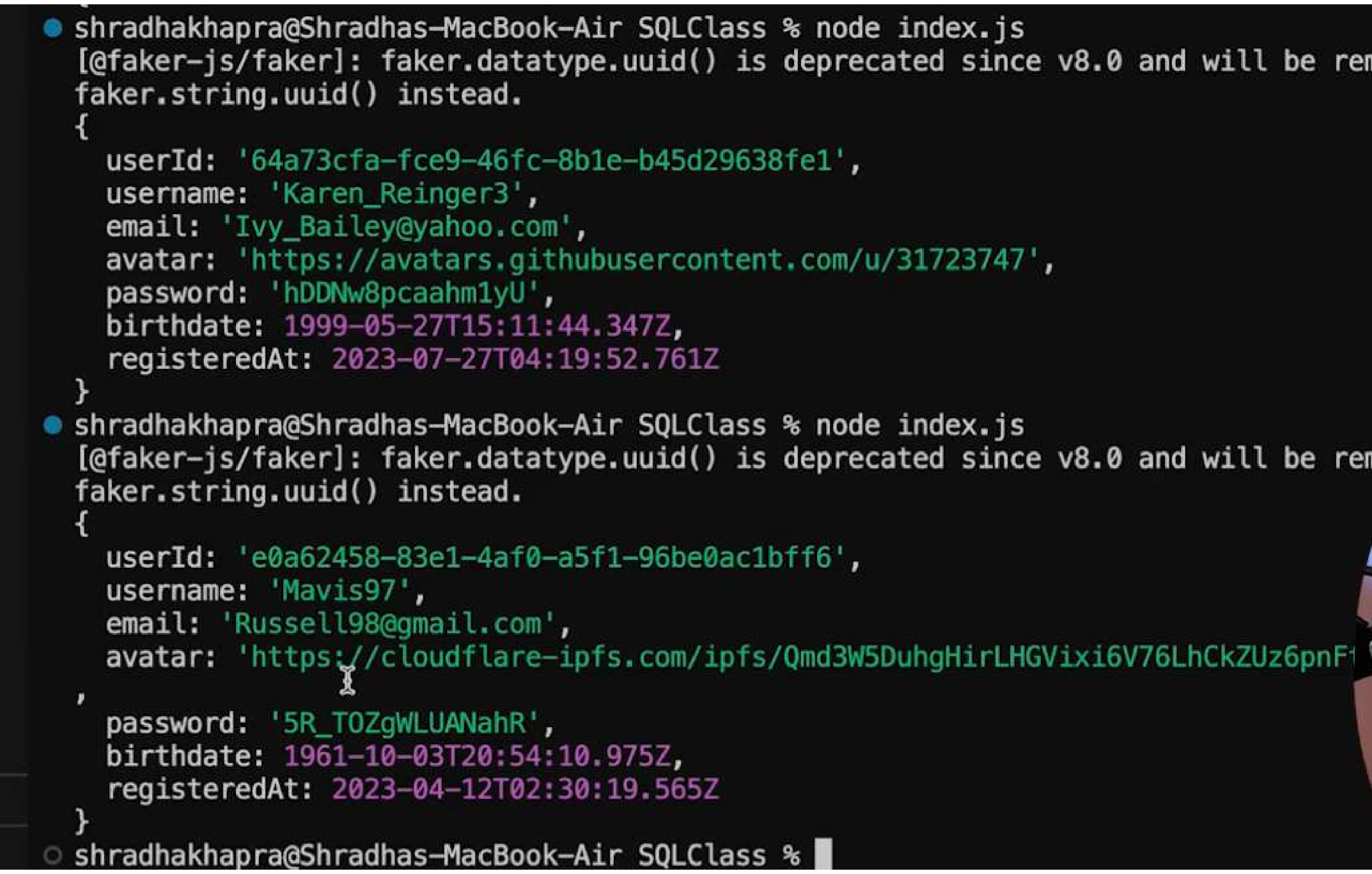
```
shradhakhapra@Shradhas-MacBook-Air SQLClass % npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (sqlclass)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author: 
```



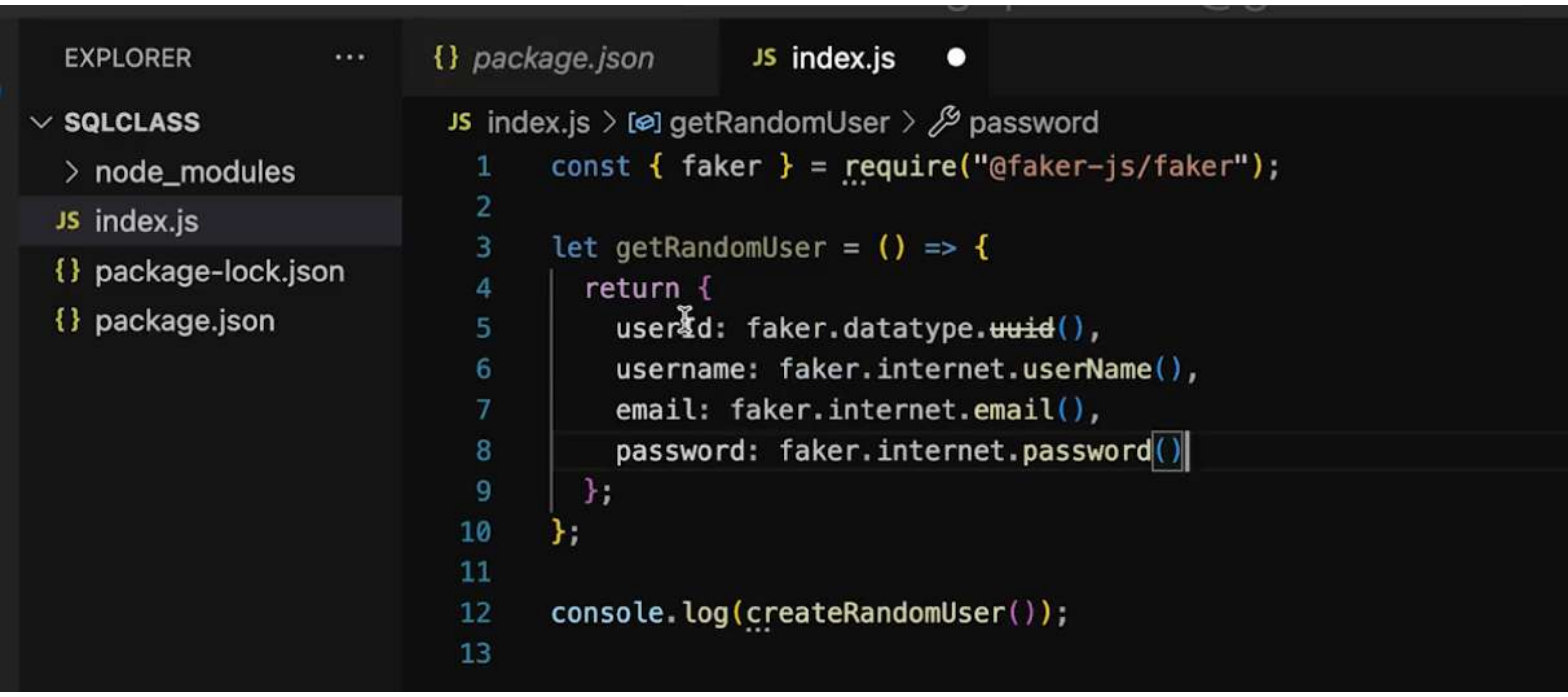
```
  "author": "Shradha Khapra",
  "license": "ISC"
}


Is this OK? (yes)
shradhakhapra@Shradhas-MacBook-Air SQLClass % npm i @faker-js/faker
```



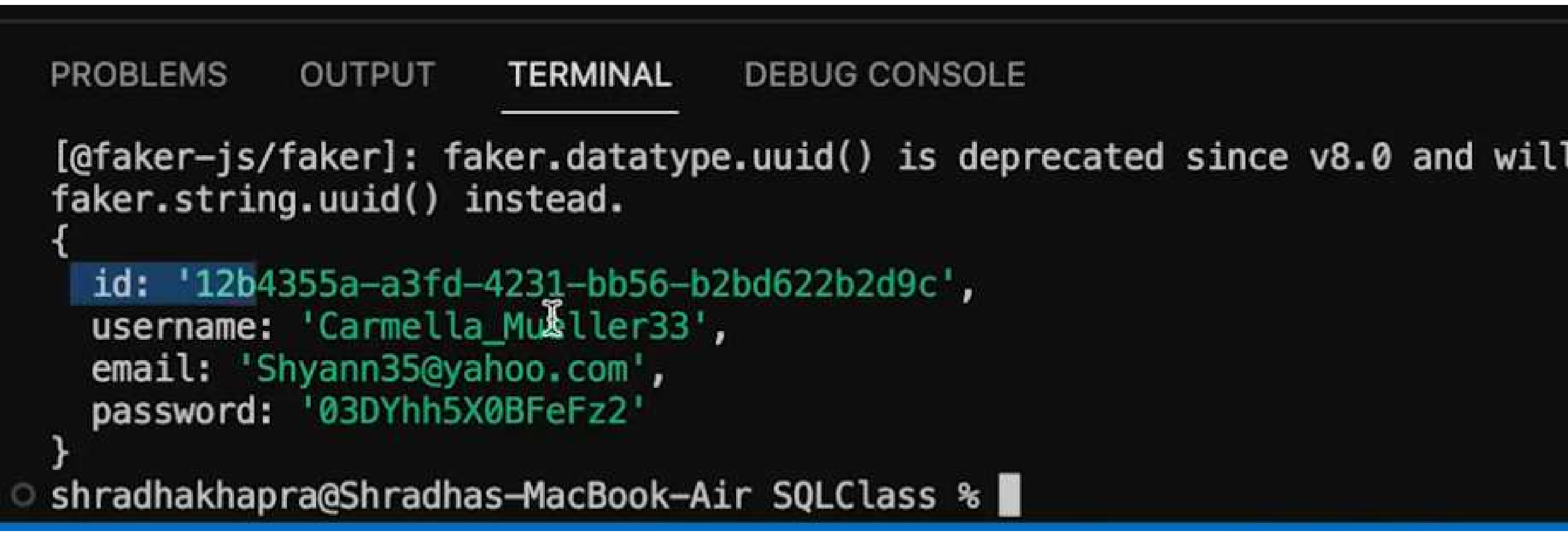
```js
const { faker } = require('@faker-js/faker');

let createRandomUser =  () => {
    return {
      userId: faker.datatype.~~uuid~~(),
      username: faker.internet.userName(),
      email: faker.internet.email(),
      avatar: faker.image.avatar(),
      password: faker.internet.password(),
      birthdate: faker.date.birthdate(),
      registeredAt: faker.date.past(),
    };
  }
```



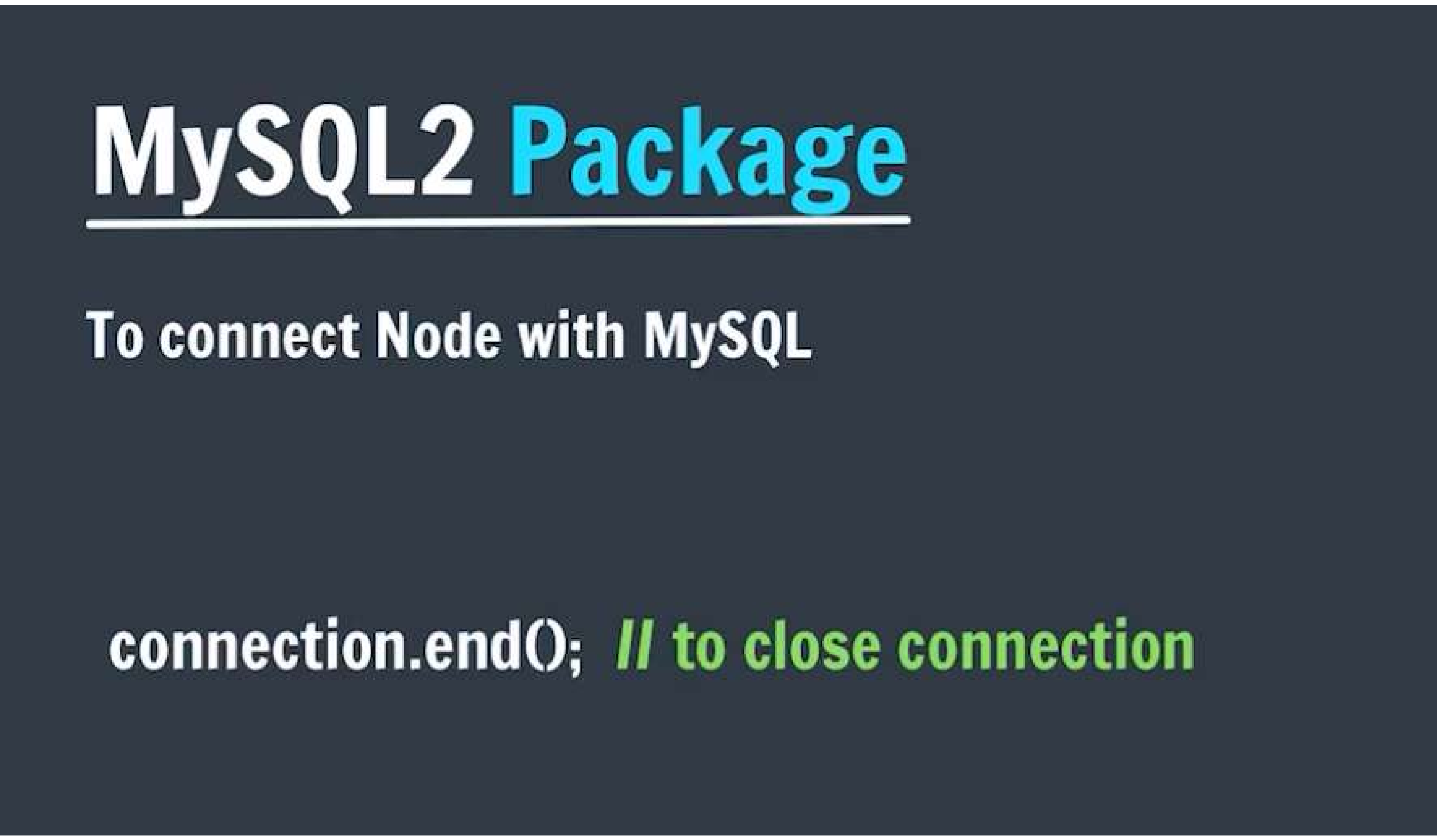
```
● shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
  [@faker-js/faker]: faker.datatype.uuid() is deprecated since v8.0 and will be rem
  faker.string.uuid() instead.
  {
    userId: '64a73cfa-fce9-46fc-8b1e-b45d29638fe1',
    username: 'Karen_Reinger3',
    email: 'Ivy_Bailey@yahoo.com',
    avatar: 'https://avatars.githubusercontent.com/u/31723747',
    password: 'hDDNw8pcaahm1yU',
    birthdate: 1999-05-27T15:11:44.347Z,
    registeredAt: 2023-07-27T04:19:52.761Z
  }
● shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
  [@faker-js/faker]: faker.datatype.uuid() is deprecated since v8.0 and will be rem
  faker.string.uuid() instead.
  {
    userId: 'e0a62458-83e1-4af0-a5f1-96be0ac1bff6',
    username: 'Mavis97',
    email: 'Russell98@gmail.com',
    avatar: 'https://cloudflare-ipfs.com/ipfs/Qmd3W5DuhgHirLHGVixi6V76LhCkZUz6pnFt
  ,
    password: '5R_TOZgWLUANahR',
    birthdate: 1961-10-03T20:54:10.975Z,
    registeredAt: 2023-04-12T02:30:19.565Z
  }
○ shradhakhapra@Shradhas-MacBook-Air SQLClass %
```



```javascript
const { faker } = require("@faker-js/faker");

let getRandomUser = () => {
  return {
    userId: faker.datatype.~~uuid~~(),
    username: faker.internet.userName(),
    email: faker.internet.email(),
    password: faker.internet.password()
  };
};

console.log(createRandomUser());
```


PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE

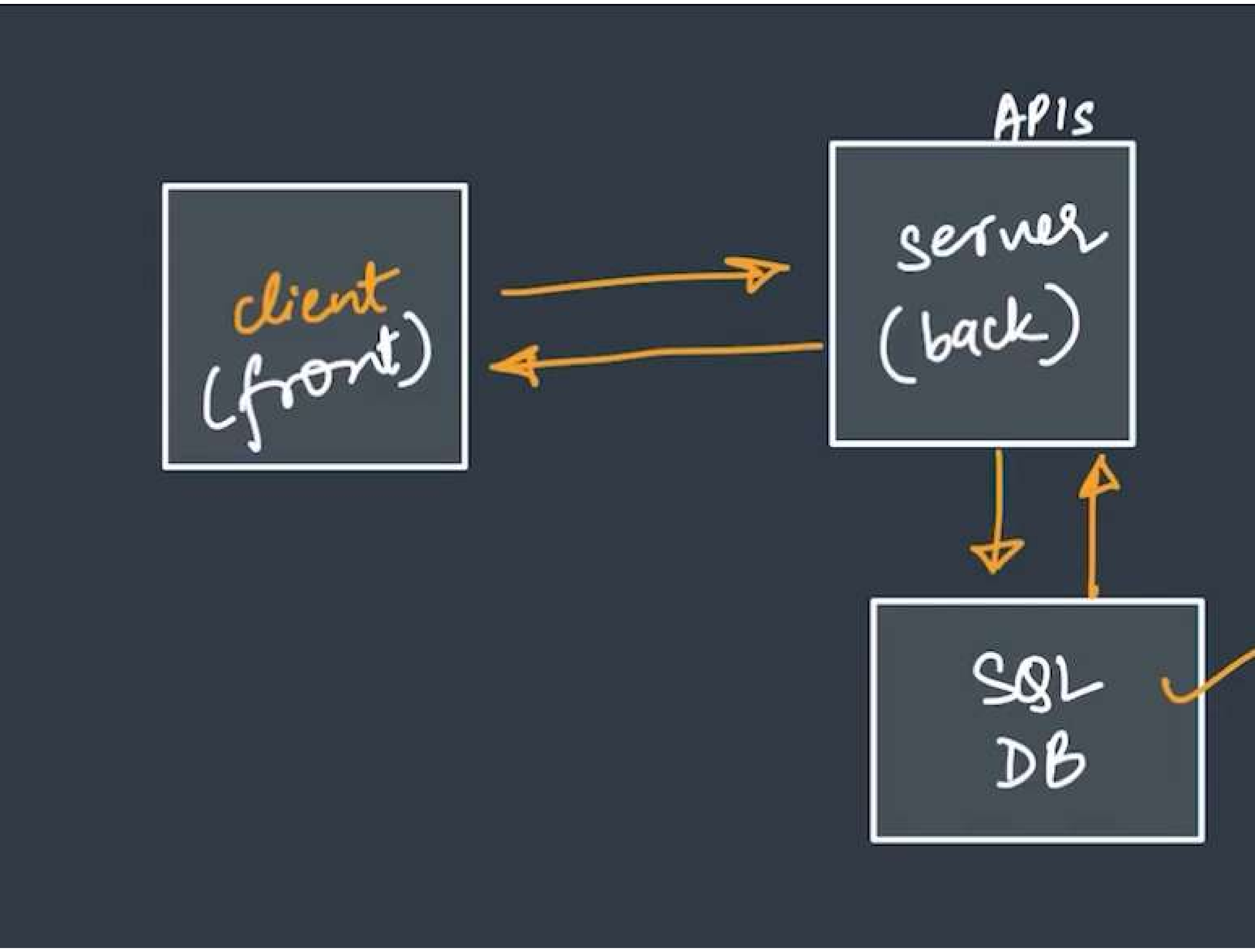
```
[@faker-js/faker]: faker.datatype.uuid() is deprecated since v8.0 and will
faker.string.uuid() instead.
{
  id: '12b4355a-a3fd-4231-bb56-b2bd622b2d9c',
  username: 'Carmella_Mueller33',
  email: 'Shyann35@yahoo.com',
  password: '03DYhh5X0BFeFz2'
}
shradhakhapra@Shradhas-MacBook-Air SQLClass %
```

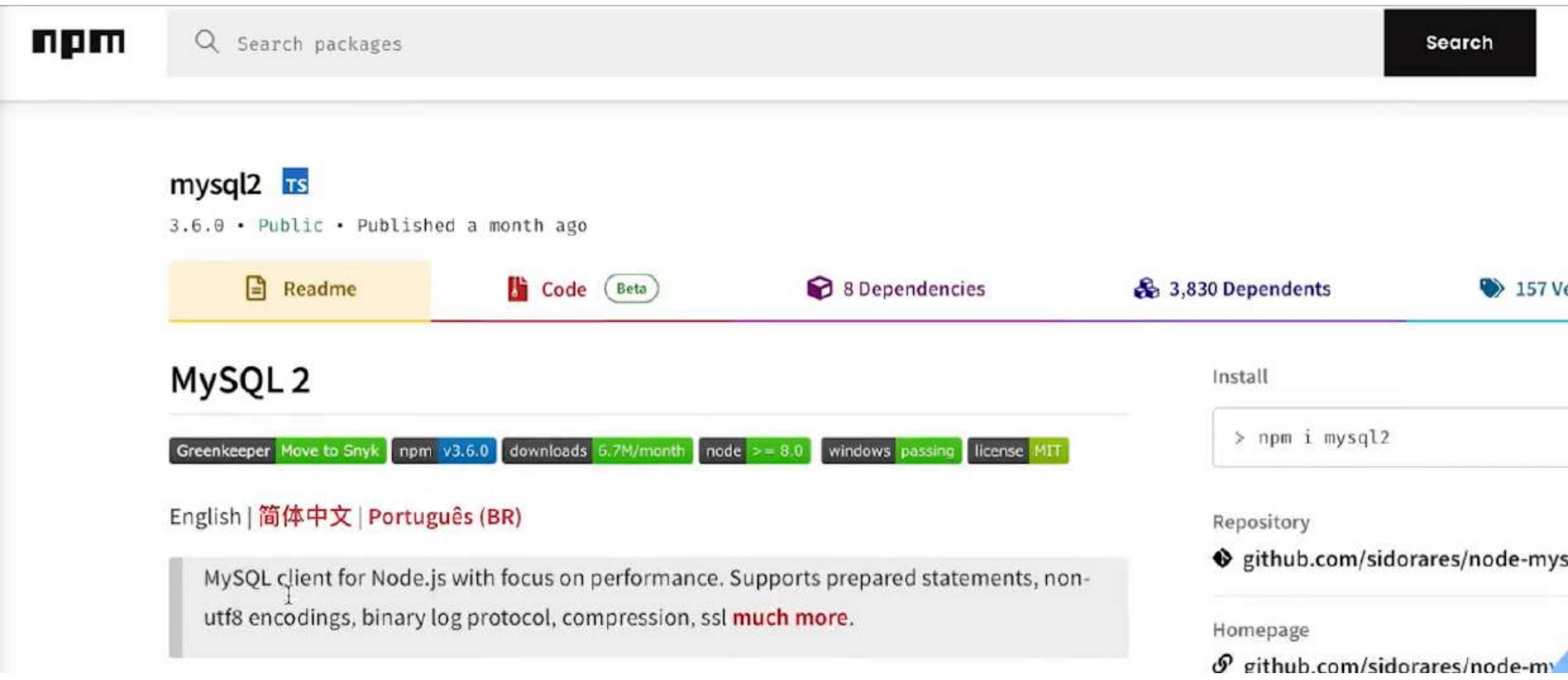


MySQL2 Package

To connect Node with MySQL

```
connection.end();  // to close connection
```


client
(front)
APIs
server
(back)
SQL
DB



mysql2 TS

3.6.0 • Public • Published a month ago

MySQL 2

English | 简体中文 | Português (BR)

MySQL client for Node.js with focus on performance. Supports prepared statements, non-utf8 encodings, binary log protocol, compression, ssl **much more**.

Install

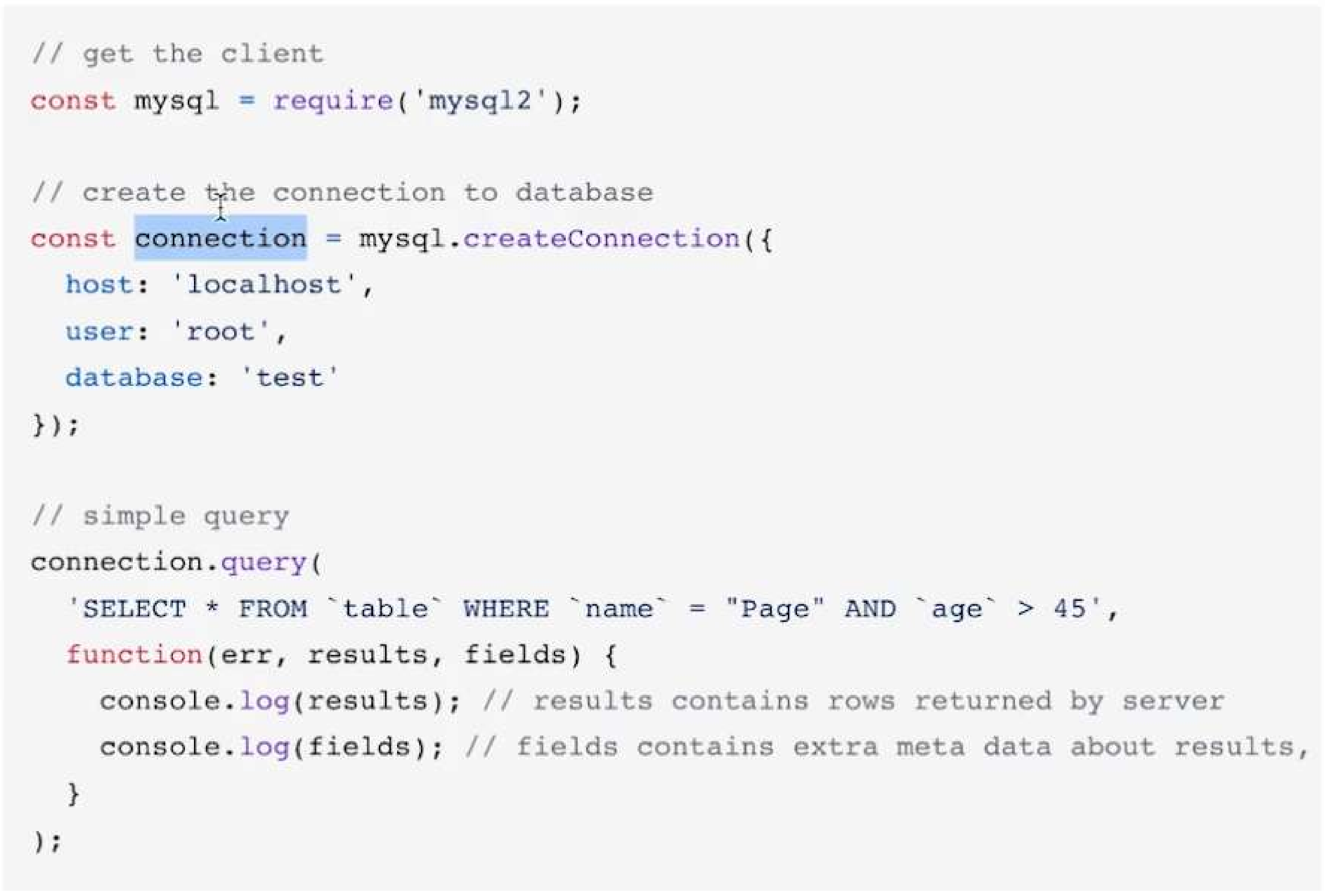
```
> npm i mysql2
```

Repository

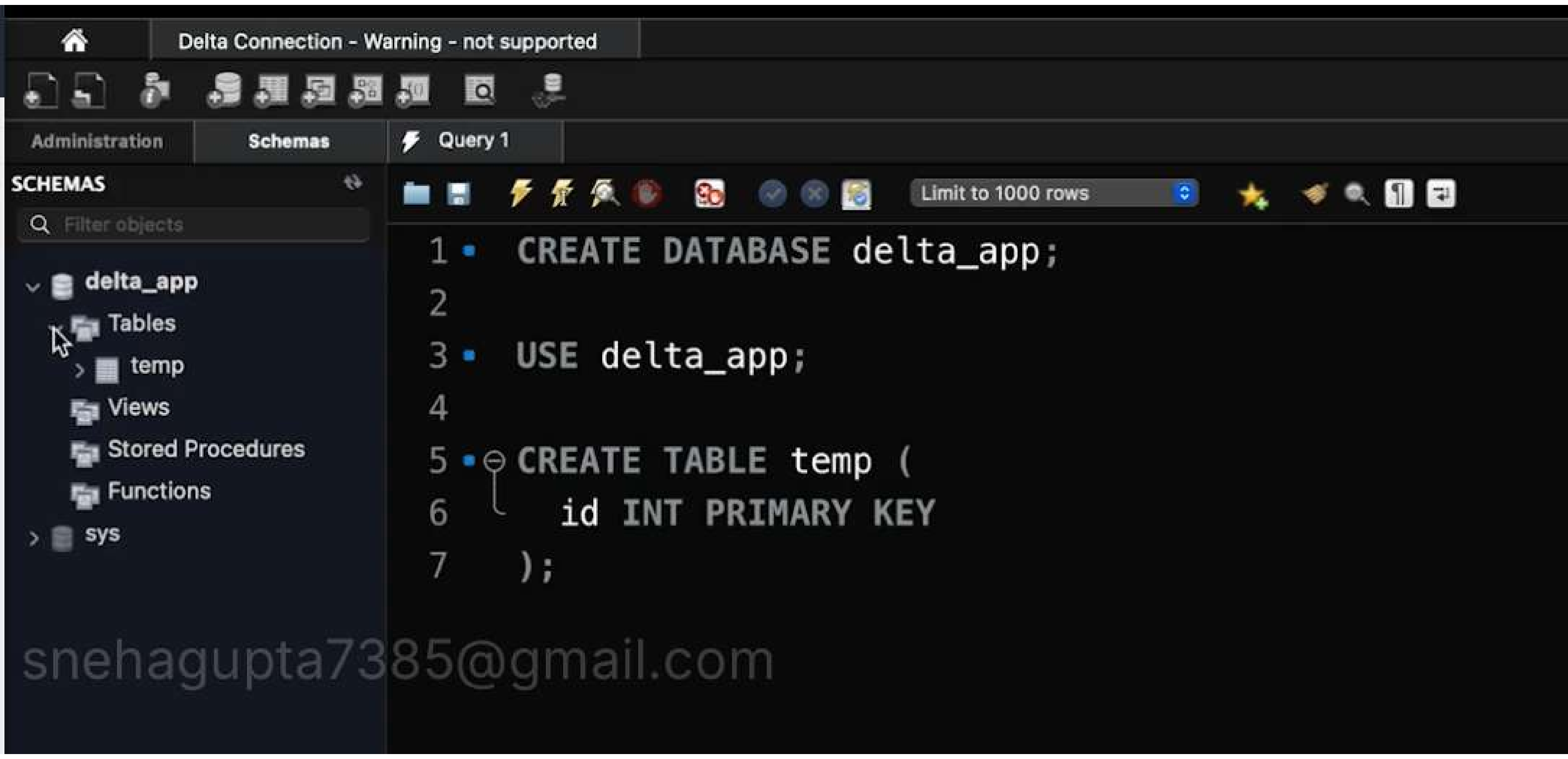
github.com/sidorares/node-mys

Homepage

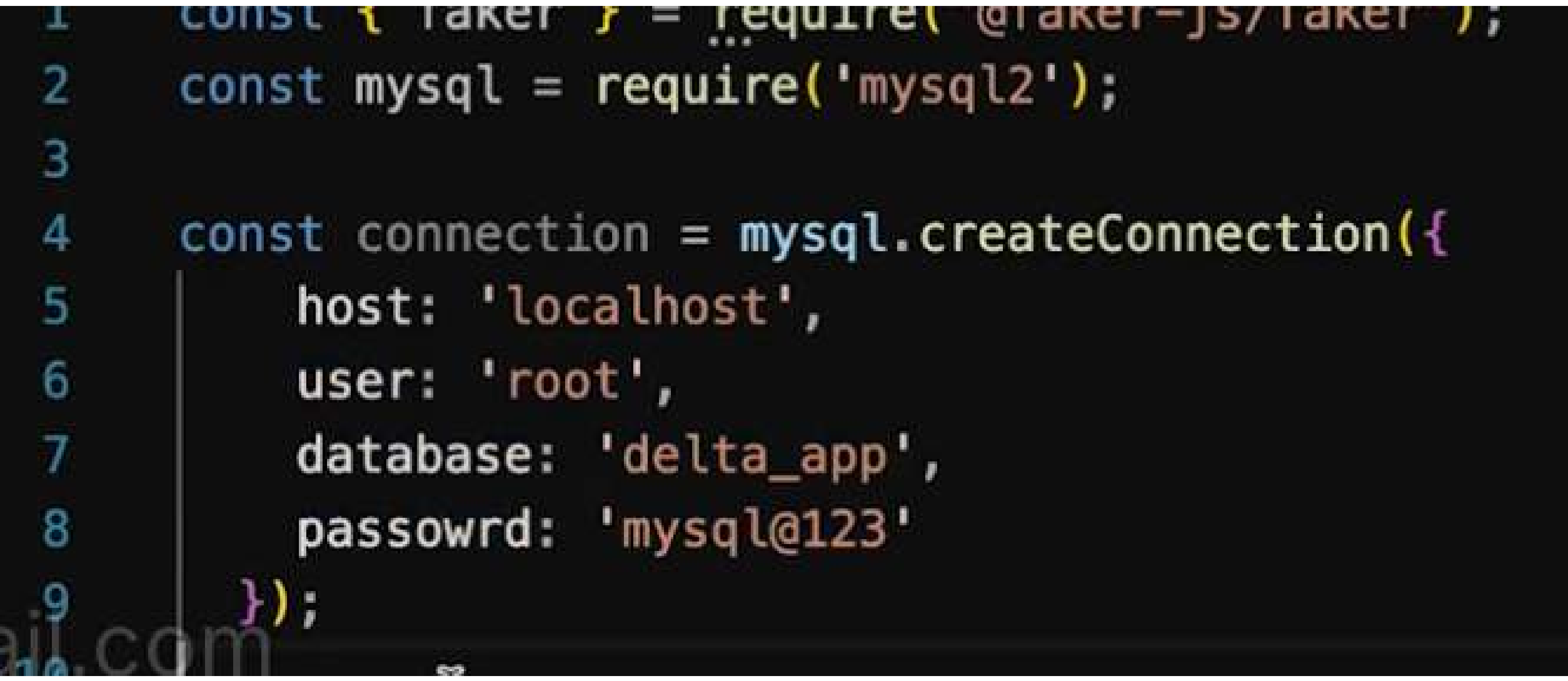
github.com/sidorares/node-my



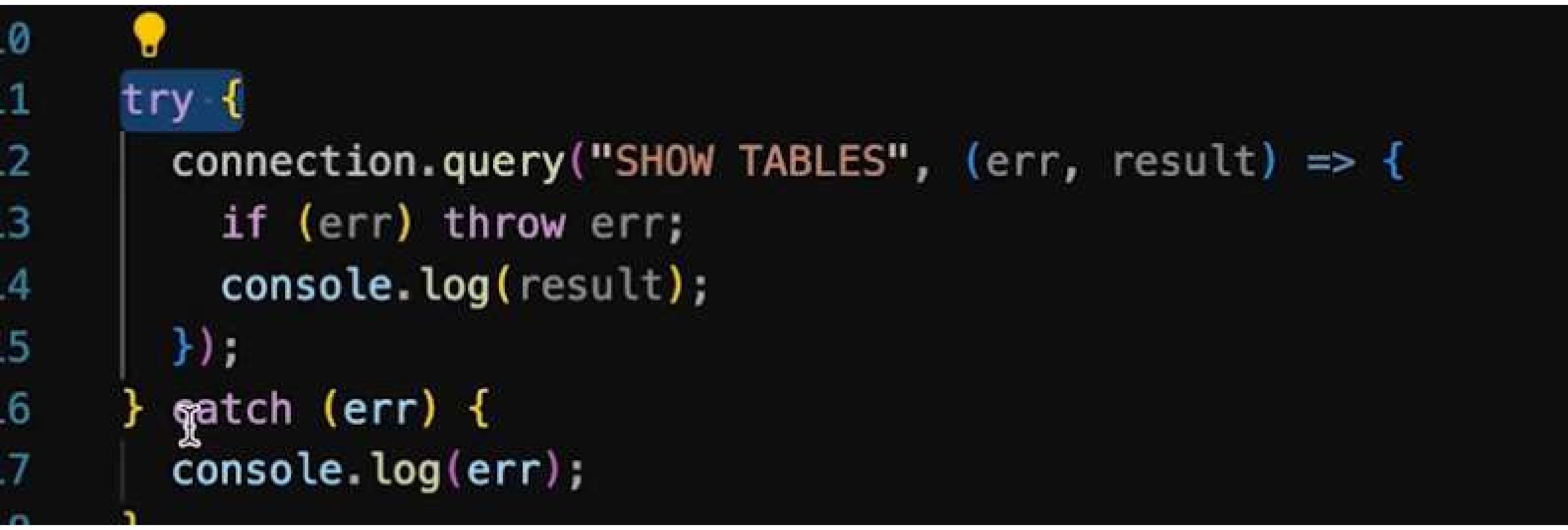
```
// get the client
const mysql = require('mysql2');

// create the connection to database
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  database: 'test'
});

// simple query
connection.query(
  'SELECT * FROM `table` WHERE `name` = "Page" AND `age` > 45',
  function(err, results, fields) {
    console.log(results); // results contains rows returned by server
    console.log(fields); // fields contains extra meta data about results,
  }
);
```



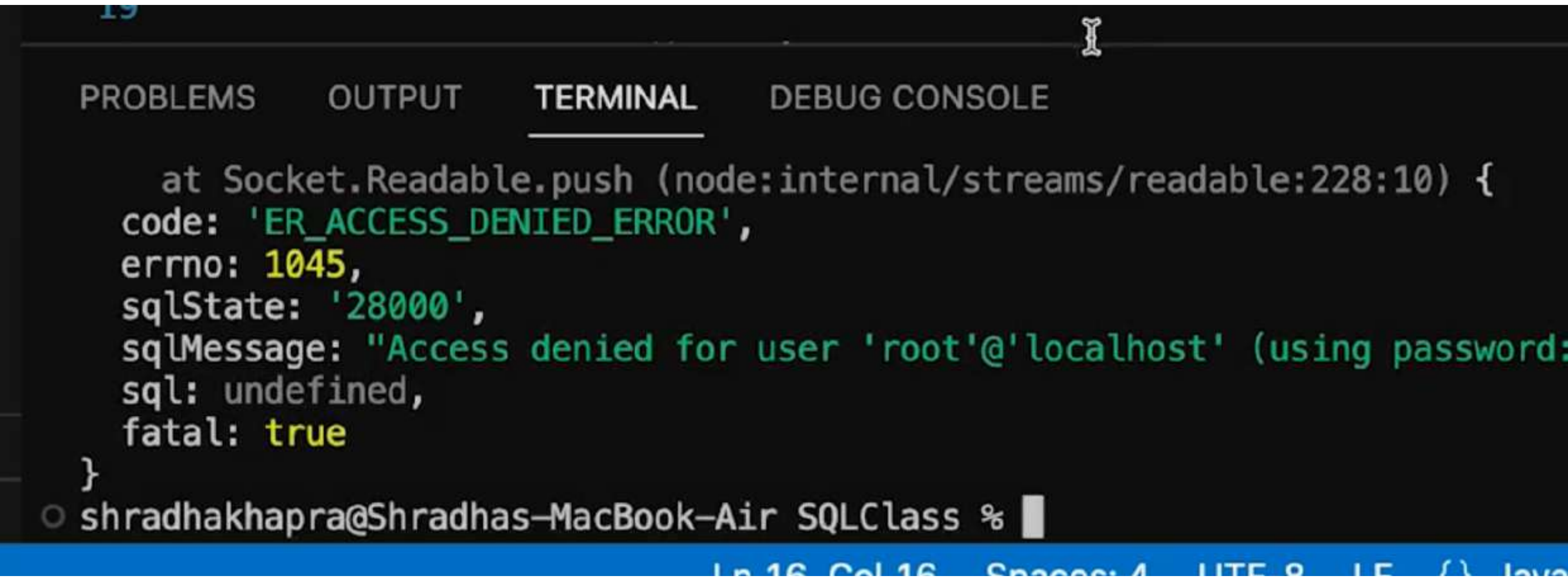
```sql
CREATE DATABASE delta_app;

USE delta_app;

CREATE TABLE temp (
  id INT PRIMARY KEY
);
```



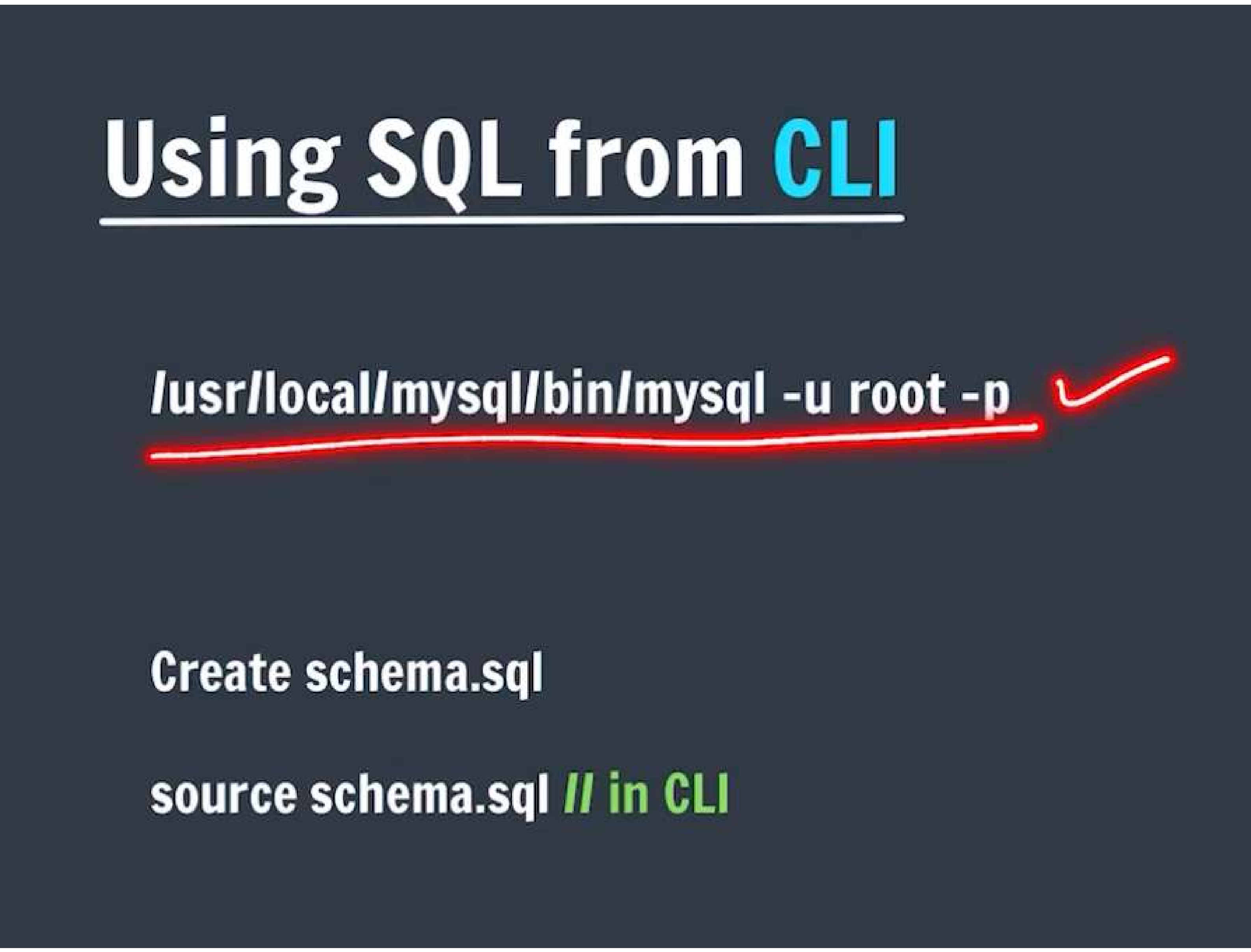
```
const { faker } = require('@faker-js/faker');
const mysql = require('mysql2');

const connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    database: 'delta_app',
    passowrd: 'mysql@123'
  });
```



```
try {
  connection.query("SHOW TABLES", (err, result) => {
    if (err) throw err;
    console.log(result);
  });
} catch (err) {
  console.log(err);
```



```
    at Socket.Readable.push (node:internal/streams/readable:228:10) {
  code: 'ER_ACCESS_DENIED_ERROR',
  errno: 1045,
  sqlState: '28000',
  sqlMessage: "Access denied for user 'root'@'localhost' (using password:
  sql: undefined,
  fatal: true
}
shradhakhapra@Shradhas-MacBook-Air SQLClass %
```

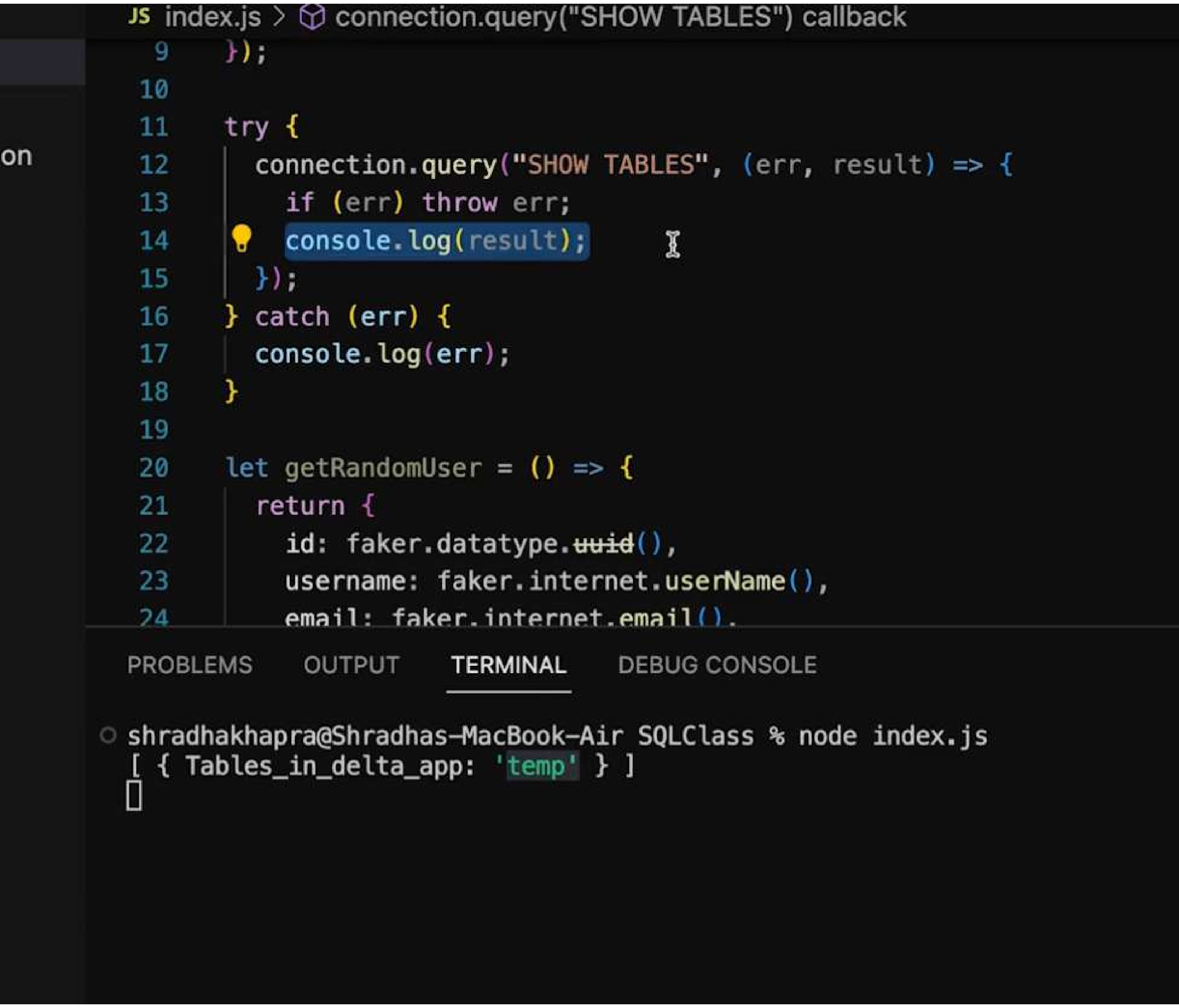


Using SQL from CLI

```
/usr/local/mysql/bin/mysql -u root -p
```

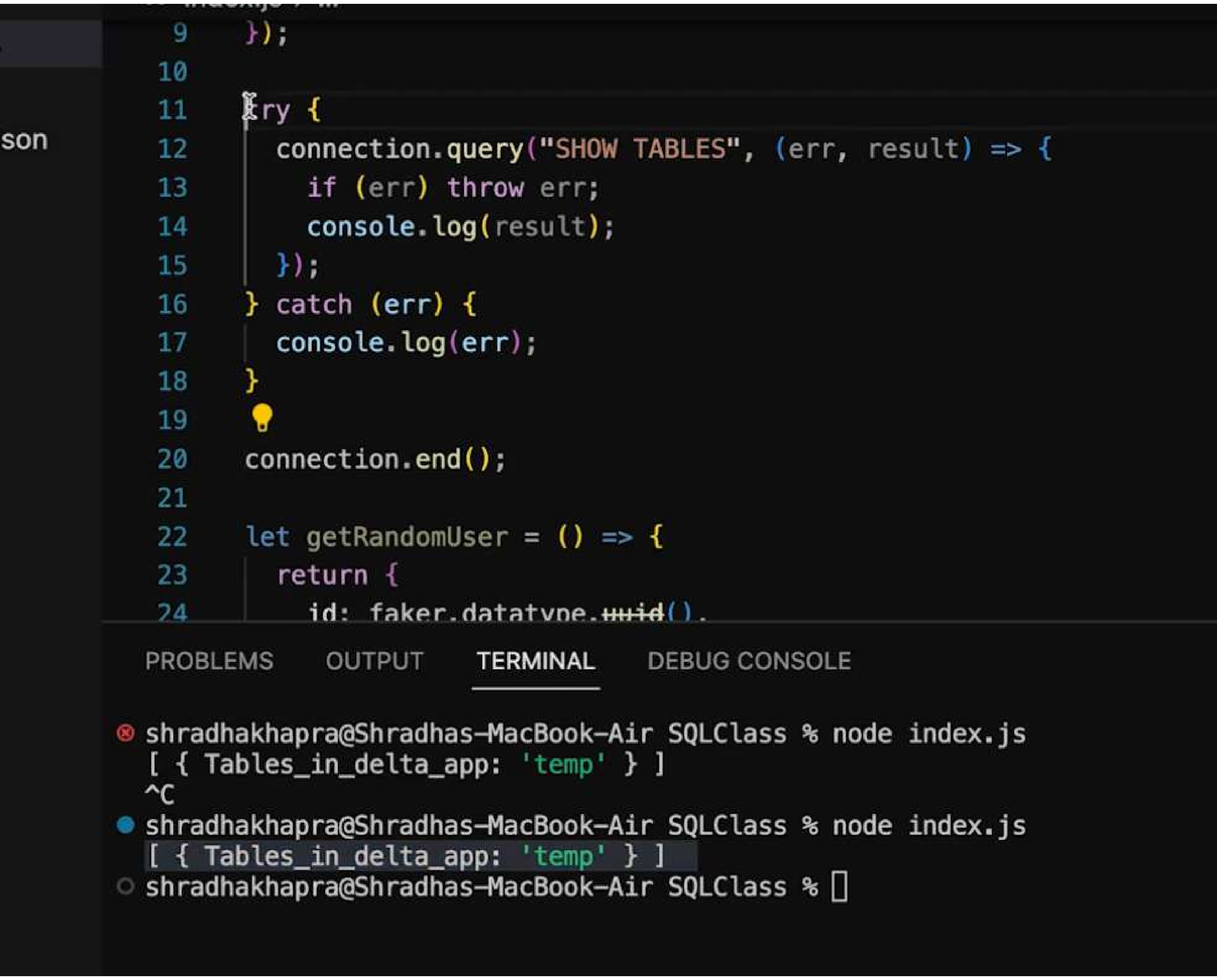
Create schema.sql

```
source schema.sql // in CLI
```



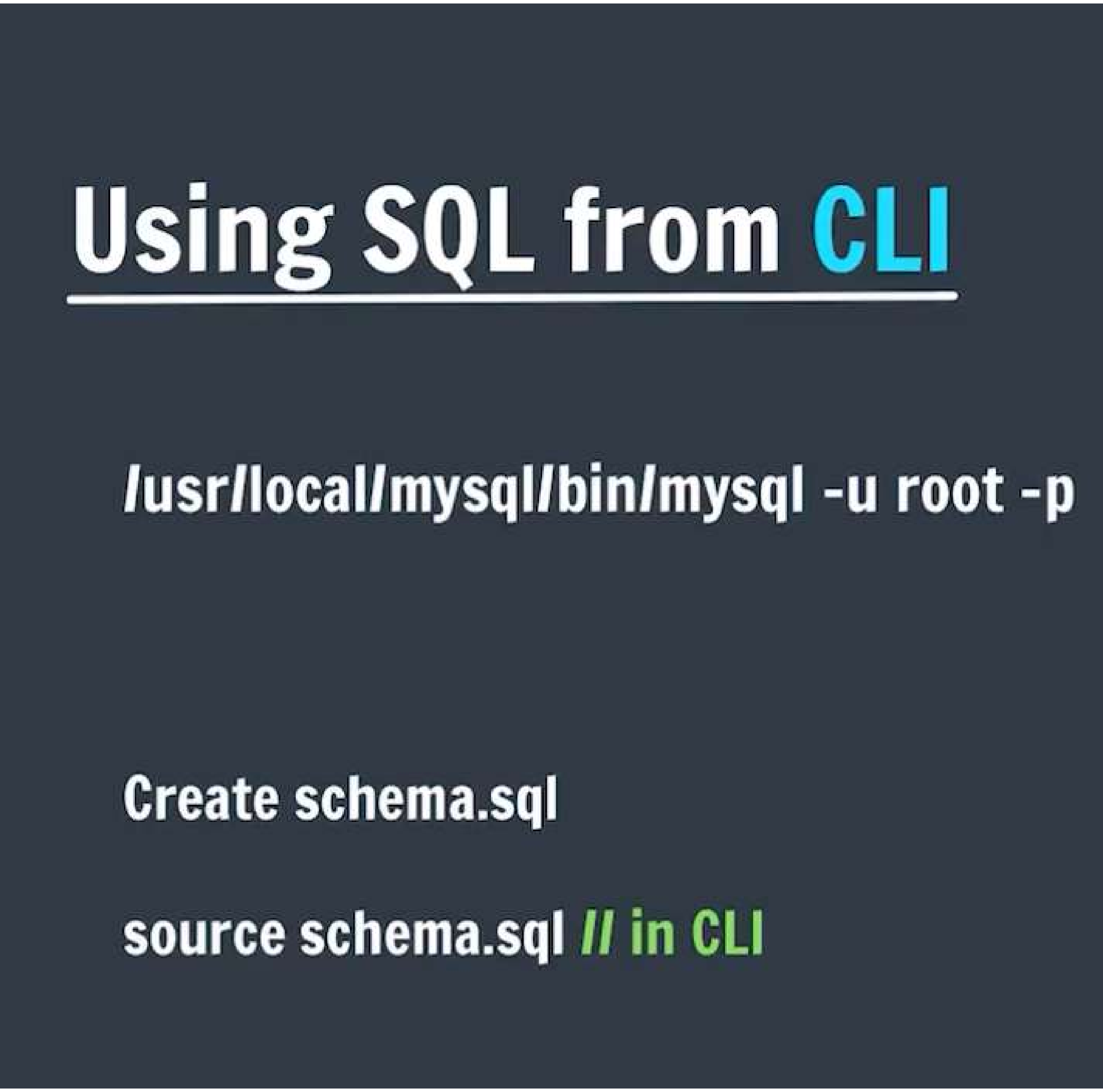
```javascript
});

try {
  connection.query("SHOW TABLES", (err, result) => {
    if (err) throw err;
    console.log(result);
  });
} catch (err) {
  console.log(err);
}

let getRandomUser = () => {
  return {
    id: faker.datatype.~~uuid~~(),
    username: faker.internet.userName(),
    email: faker.internet.email(),
```

```
shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[ { Tables_in_delta_app: 'temp' } ]
```



```
});

try {
  connection.query("SHOW TABLES", (err, result) => {
    if (err) throw err;
    console.log(result);
  });
} catch (err) {
  console.log(err);
}

connection.end();

let getRandomUser = () => {
  return {
    id: faker.datatype.~~uuid~~(),
```

```
shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[ { Tables_in_delta_app: 'temp' } ]
^C
shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[ { Tables_in_delta_app: 'temp' } ]
shradhakhapra@Shradhas-MacBook-Air SQLClass %
```

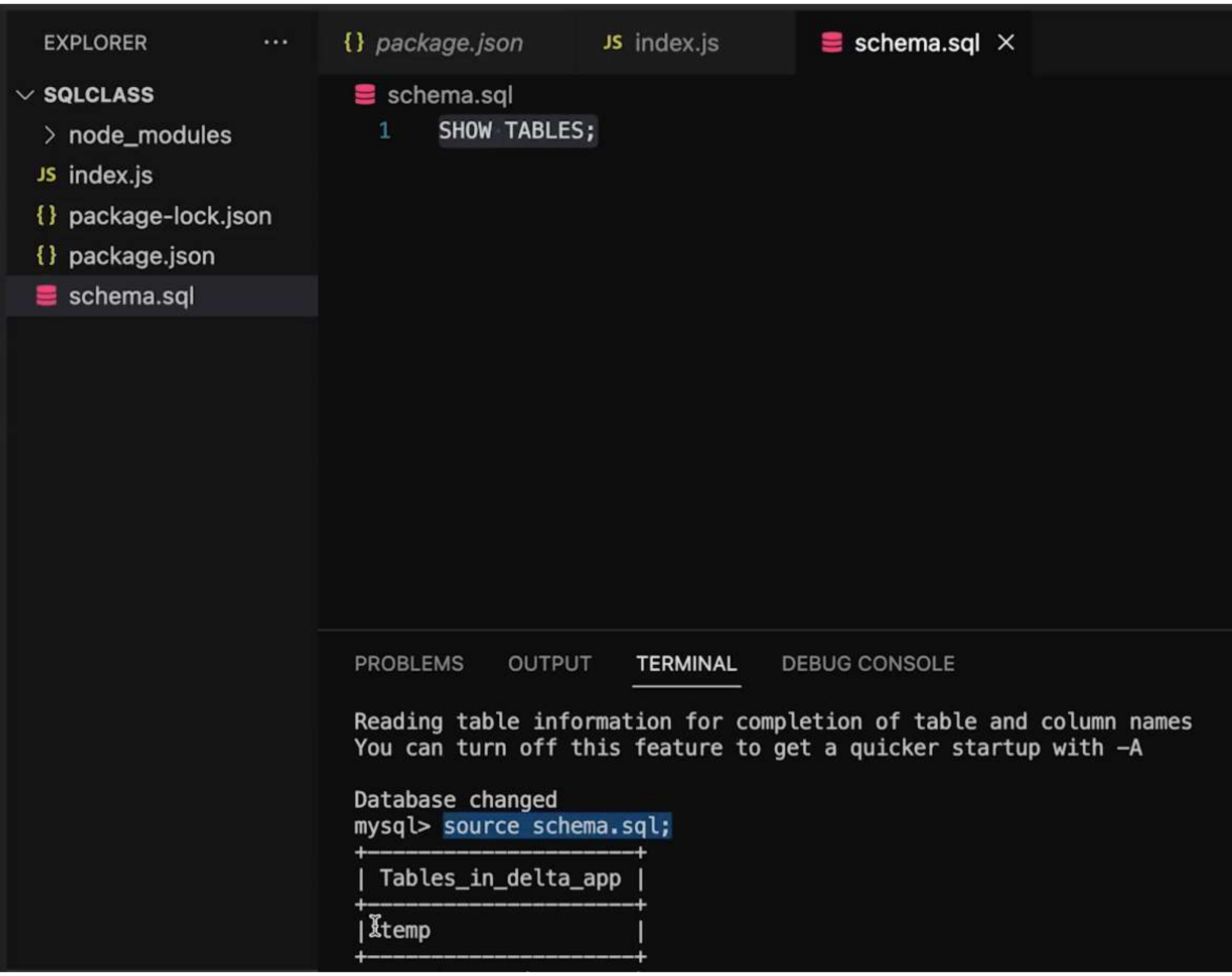


Using SQL from CLI

```
/usr/local/mysql/bin/mysql -u root -p
```

Create schema.sql

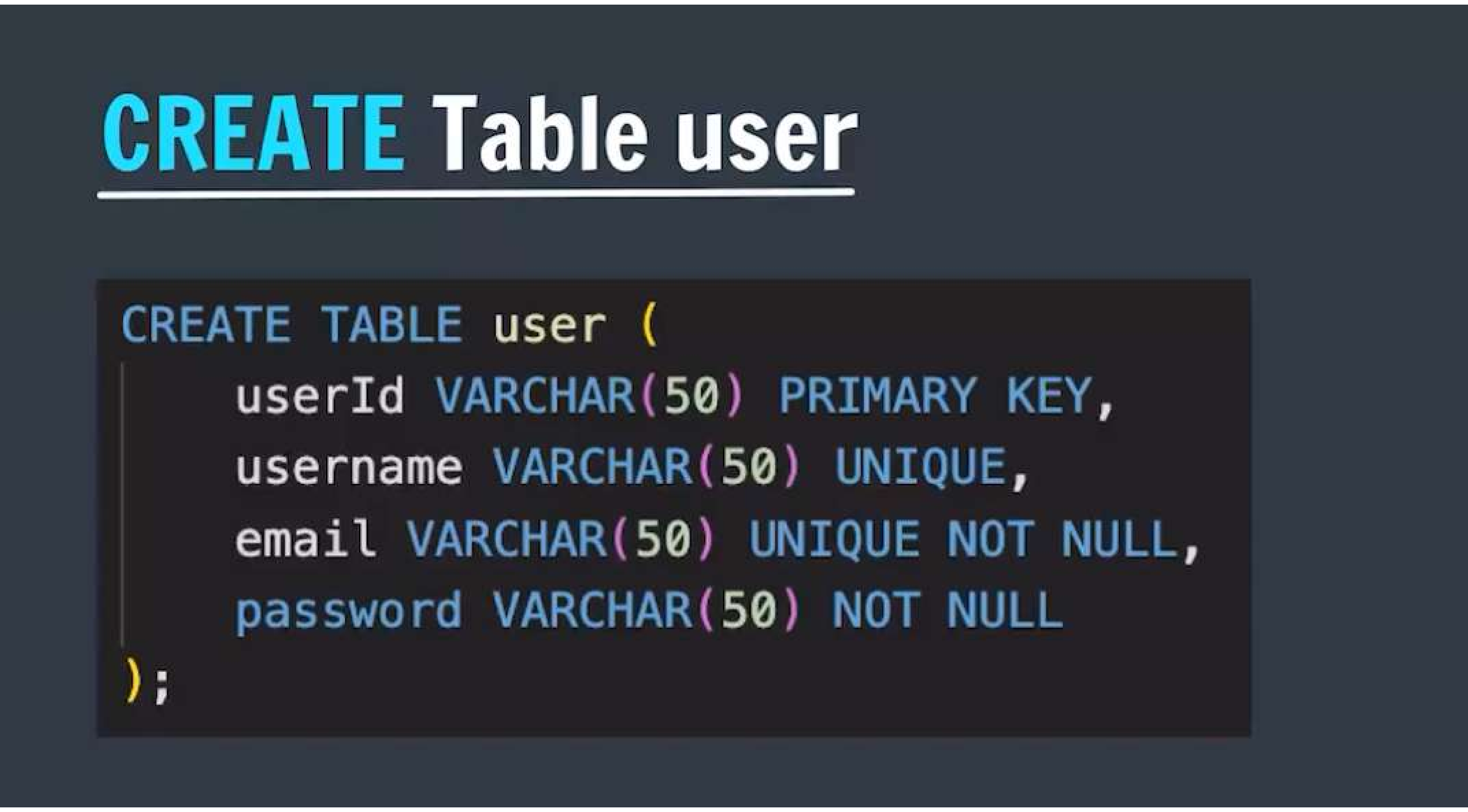
```
source schema.sql // in CLI
```



schema.sql

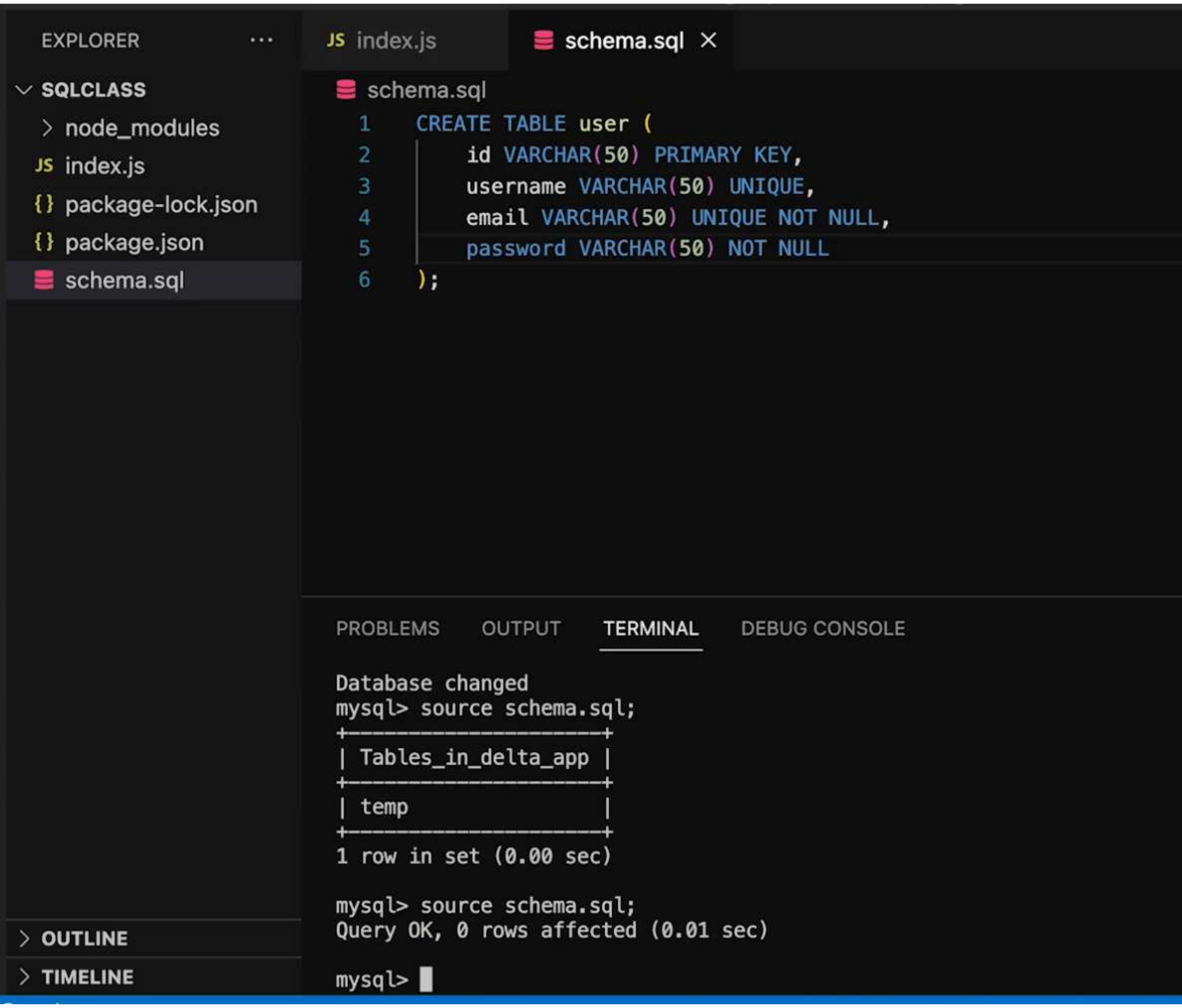
```sql
SHOW TABLES;
```

```
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> source schema.sql;
+----------------------+
| Tables_in_delta_app |
+----------------------+
| temp                 |
+----------------------+
```



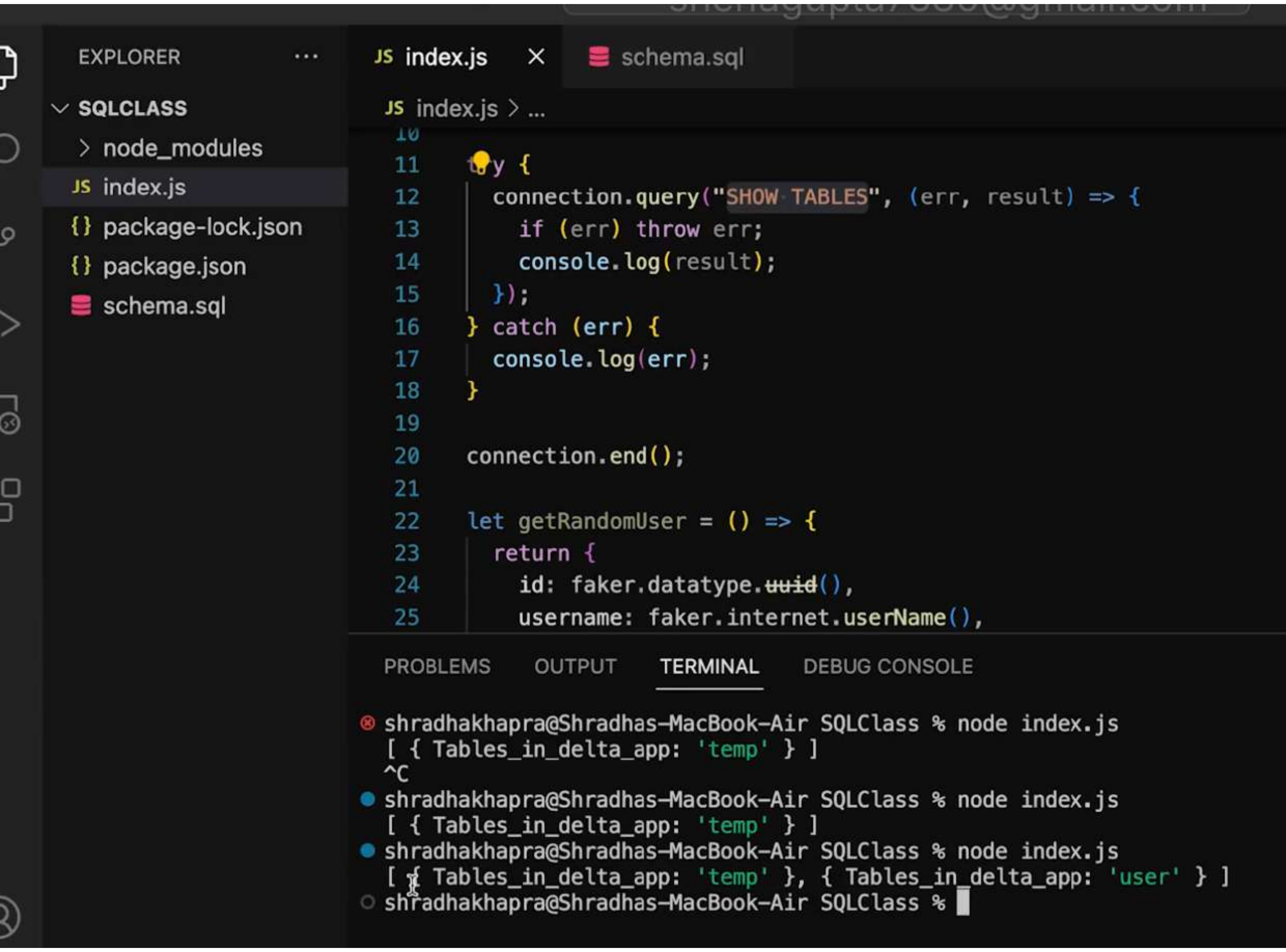
CREATE Table user

```sql
CREATE TABLE user (
    userId VARCHAR(50) PRIMARY KEY,
    username VARCHAR(50) UNIQUE,
    email VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(50) NOT NULL
);
```



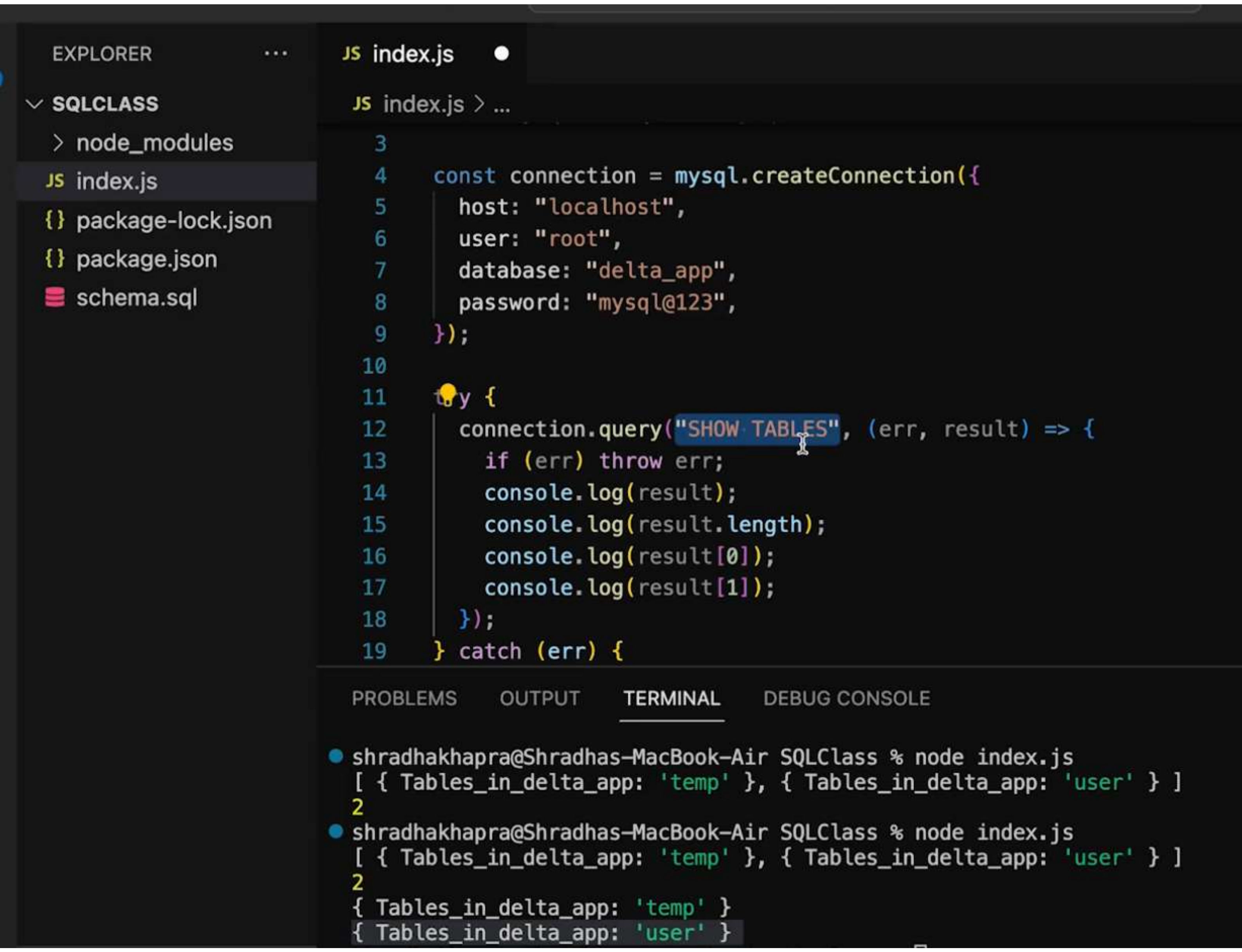
```sql
CREATE TABLE user (
    id VARCHAR(50) PRIMARY KEY,
    username VARCHAR(50) UNIQUE,
    email VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(50) NOT NULL
);
```

```
Database changed
mysql> source schema.sql;
+--------------------+
| Tables_in_delta_app |
+--------------------+
| temp               |
+--------------------+
1 row in set (0.00 sec)

mysql> source schema.sql;
Query OK, 0 rows affected (0.01 sec)

mysql> 
```



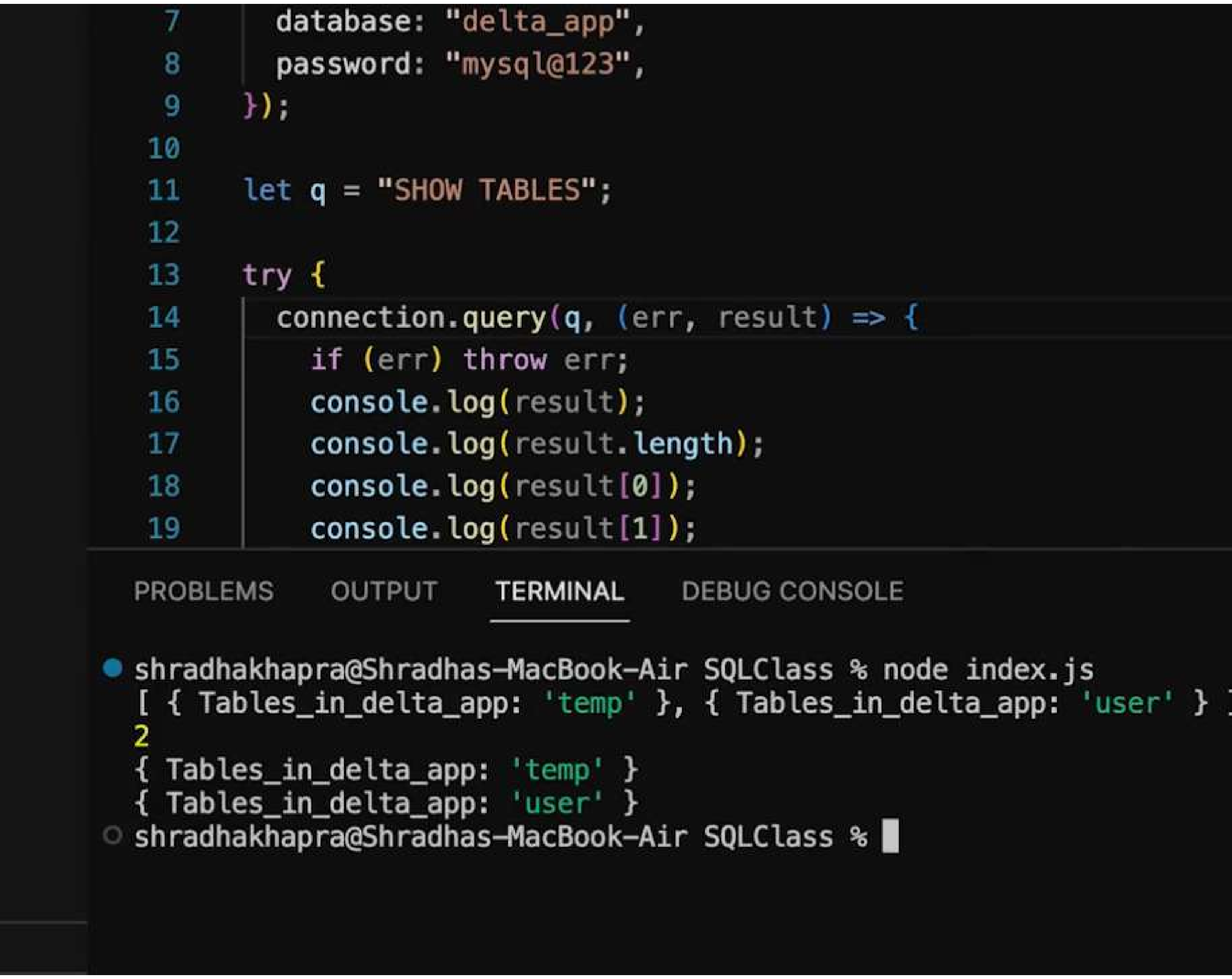
```javascript
try {
  connection.query("SHOW TABLES", (err, result) => {
    if (err) throw err;
    console.log(result);
  });
} catch (err) {
  console.log(err);
}

connection.end();

let getRandomUser = () => {
  return {
    id: faker.datatype.~~uuid~~(),
    username: faker.internet.userName(),
```

```
shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[ { Tables_in_delta_app: 'temp' } ]
^C
shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[ { Tables_in_delta_app: 'temp' } ]
shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[ { Tables_in_delta_app: 'temp' }, { Tables_in_delta_app: 'user' } ]
shradhakhapra@Shradhas-MacBook-Air SQLClass %
```



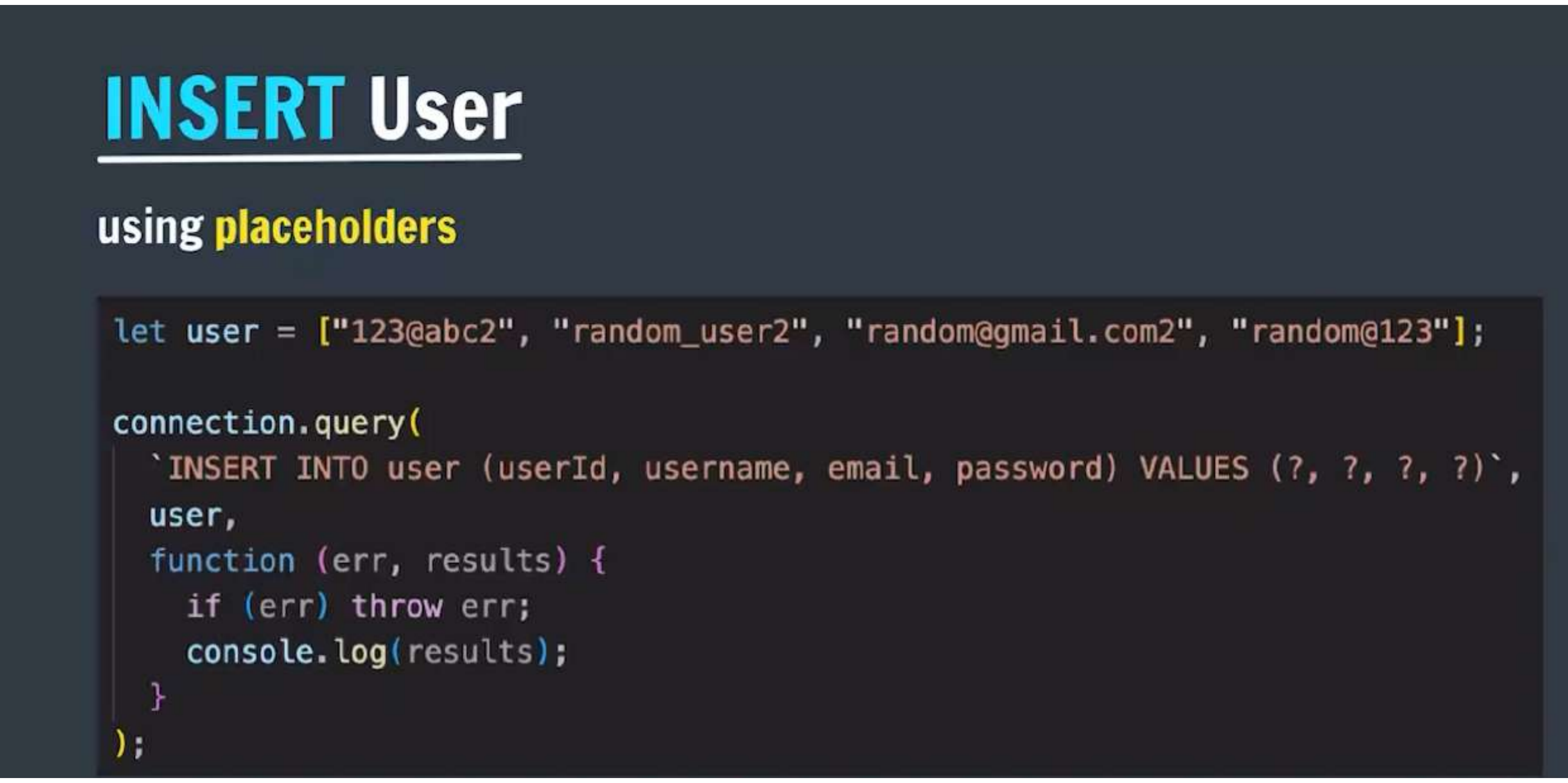
```javascript

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "delta_app",
  password: "mysql@123",
});

try {
  connection.query("SHOW TABLES", (err, result) => {
    if (err) throw err;
    console.log(result);
    console.log(result.length);
    console.log(result[0]);
    console.log(result[1]);
  });
} catch (err) {
```

```
shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[ { Tables_in_delta_app: 'temp' }, { Tables_in_delta_app: 'user' } ]
2
shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[ { Tables_in_delta_app: 'temp' }, { Tables_in_delta_app: 'user' } ]
2
{ Tables_in_delta_app: 'temp' }
{ Tables_in_delta_app: 'user' }
```



```
    database: "delta_app",
    password: "mysql@123",
});

let q = "SHOW TABLES";

try {
  connection.query(q, (err, result) => {
    if (err) throw err;
    console.log(result);
    console.log(result.length);
    console.log(result[0]);
    console.log(result[1]);
```

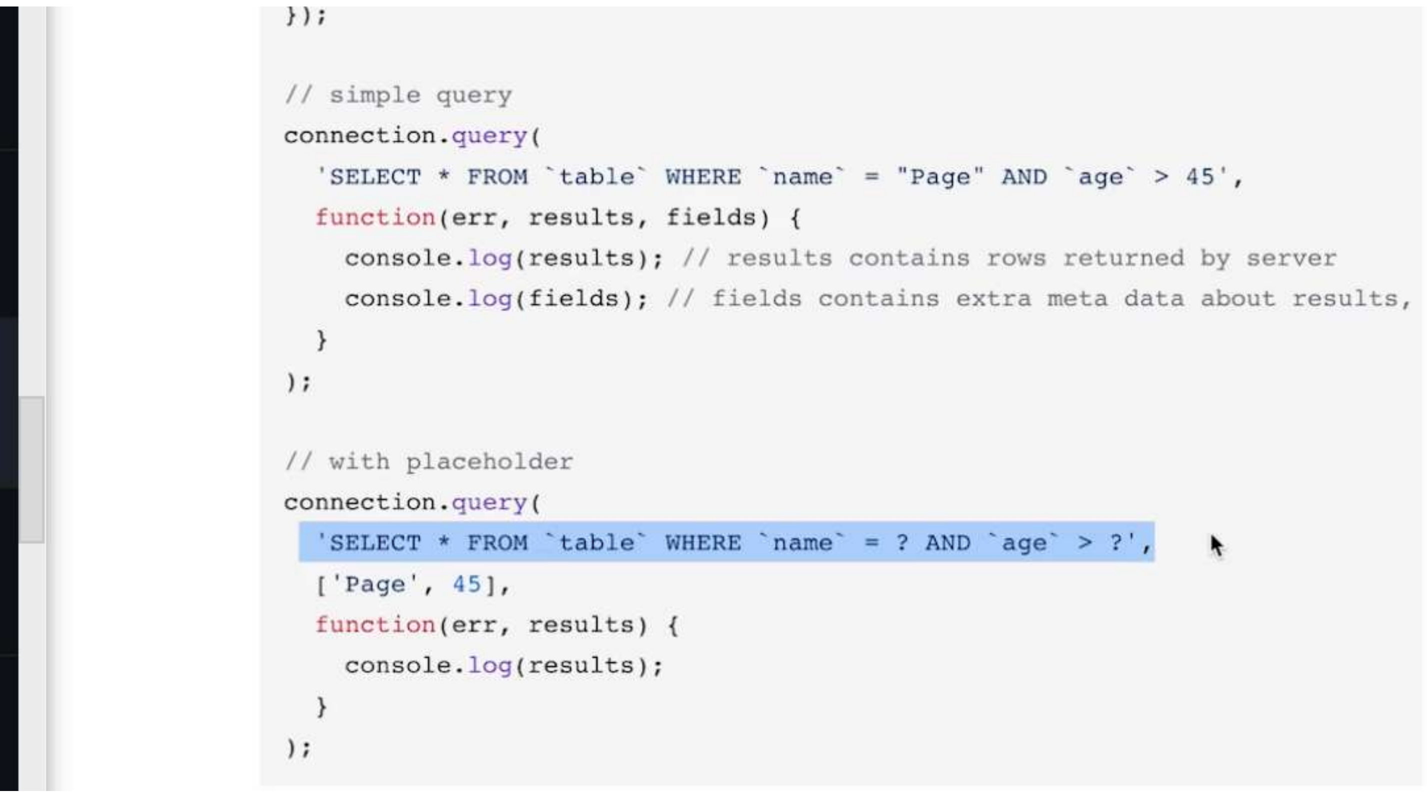
```
shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
[ { Tables_in_delta_app: 'temp' }, { Tables_in_delta_app: 'user' } ]
2
{ Tables_in_delta_app: 'temp' }
{ Tables_in_delta_app: 'user' }
shradhakhapra@Shradhas-MacBook-Air SQLClass %
```



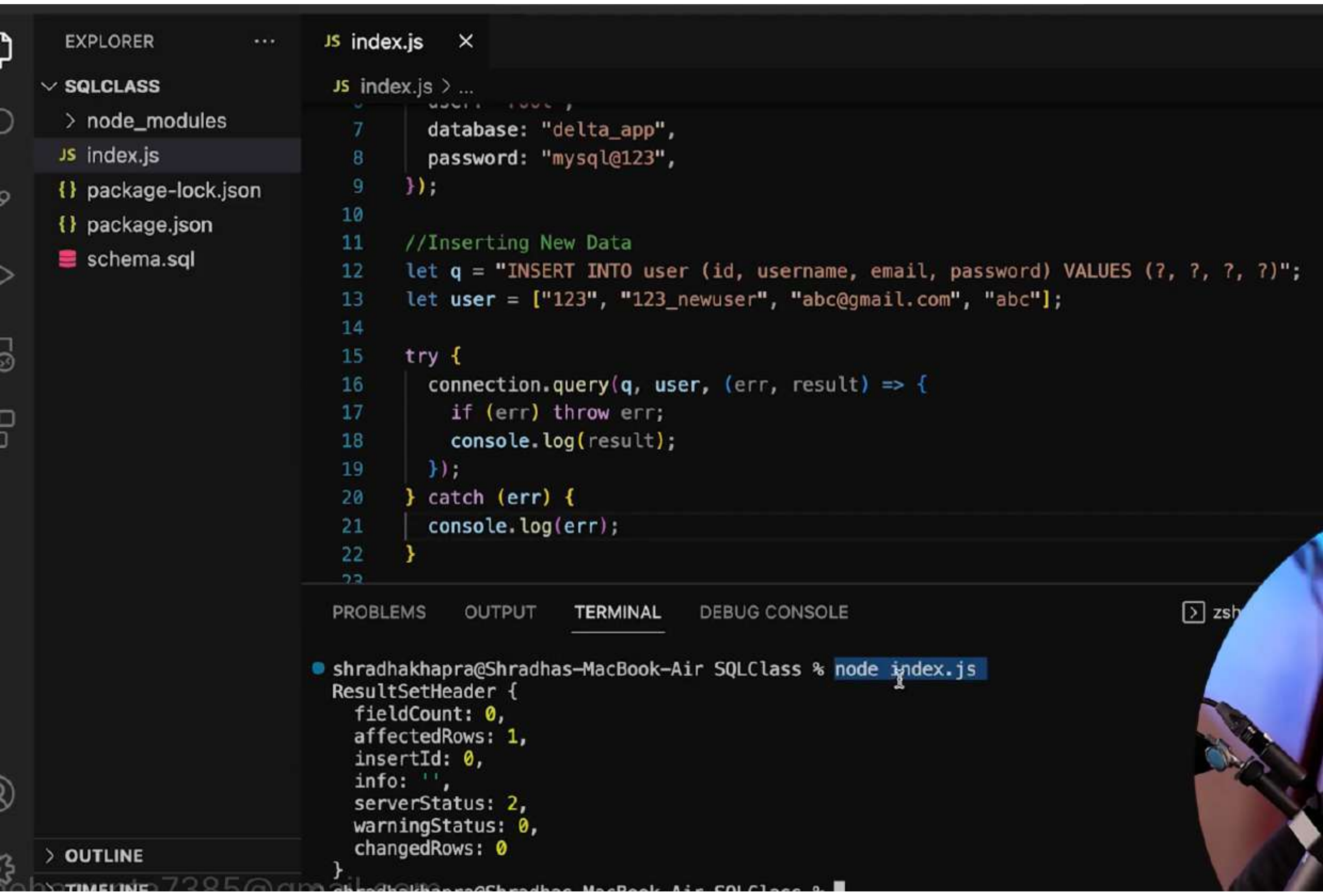
INSERT User

using placeholders

```
let user = ["123@abc2", "random_user2", "random@gmail.com2", "random@123"];

connection.query(
  `INSERT INTO user (userId, username, email, password) VALUES (?, ?, ?, ?)`,
  user,
  function (err, results) {
    if (err) throw err;
    console.log(results);
  }
);
```

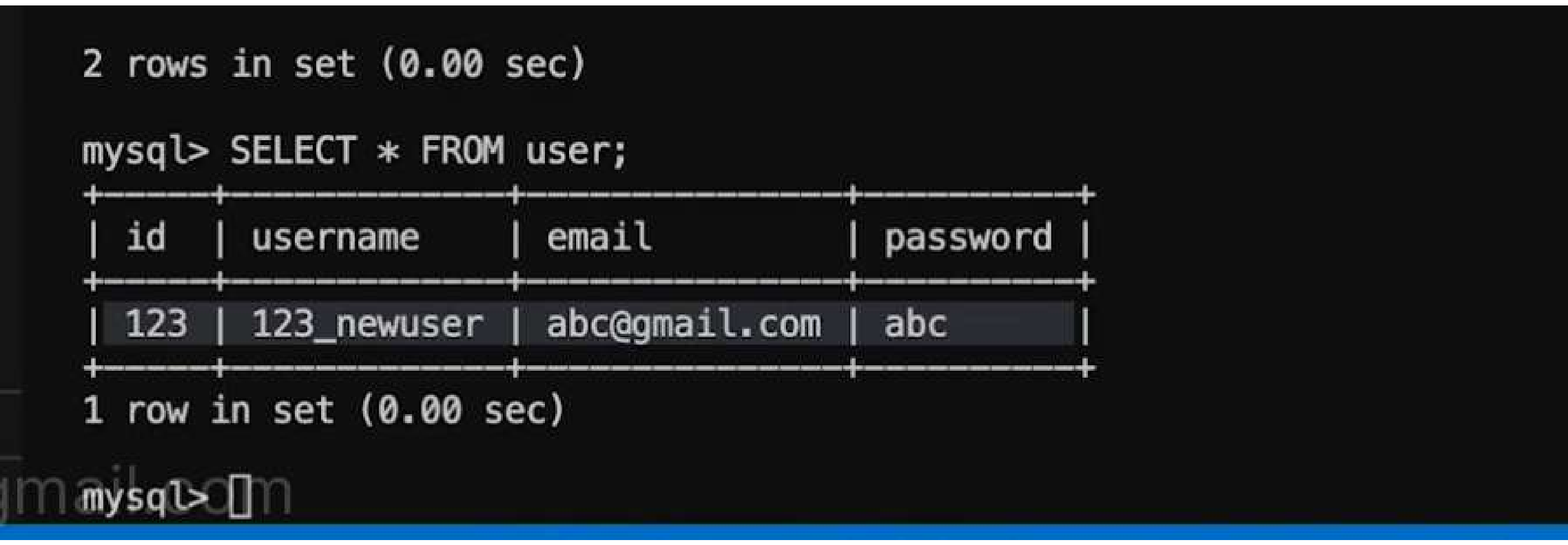


```
});

// simple query
connection.query(
  'SELECT * FROM `table` WHERE `name` = "Page" AND `age` > 45',
  function(err, results, fields) {
    console.log(results); // results contains rows returned by server
    console.log(fields); // fields contains extra meta data about results,
  }
);

// with placeholder
connection.query(
  'SELECT * FROM `table` WHERE `name` = ? AND `age` > ?',
  ['Page', 45],
  function(err, results) {
    console.log(results);
  }
);
```

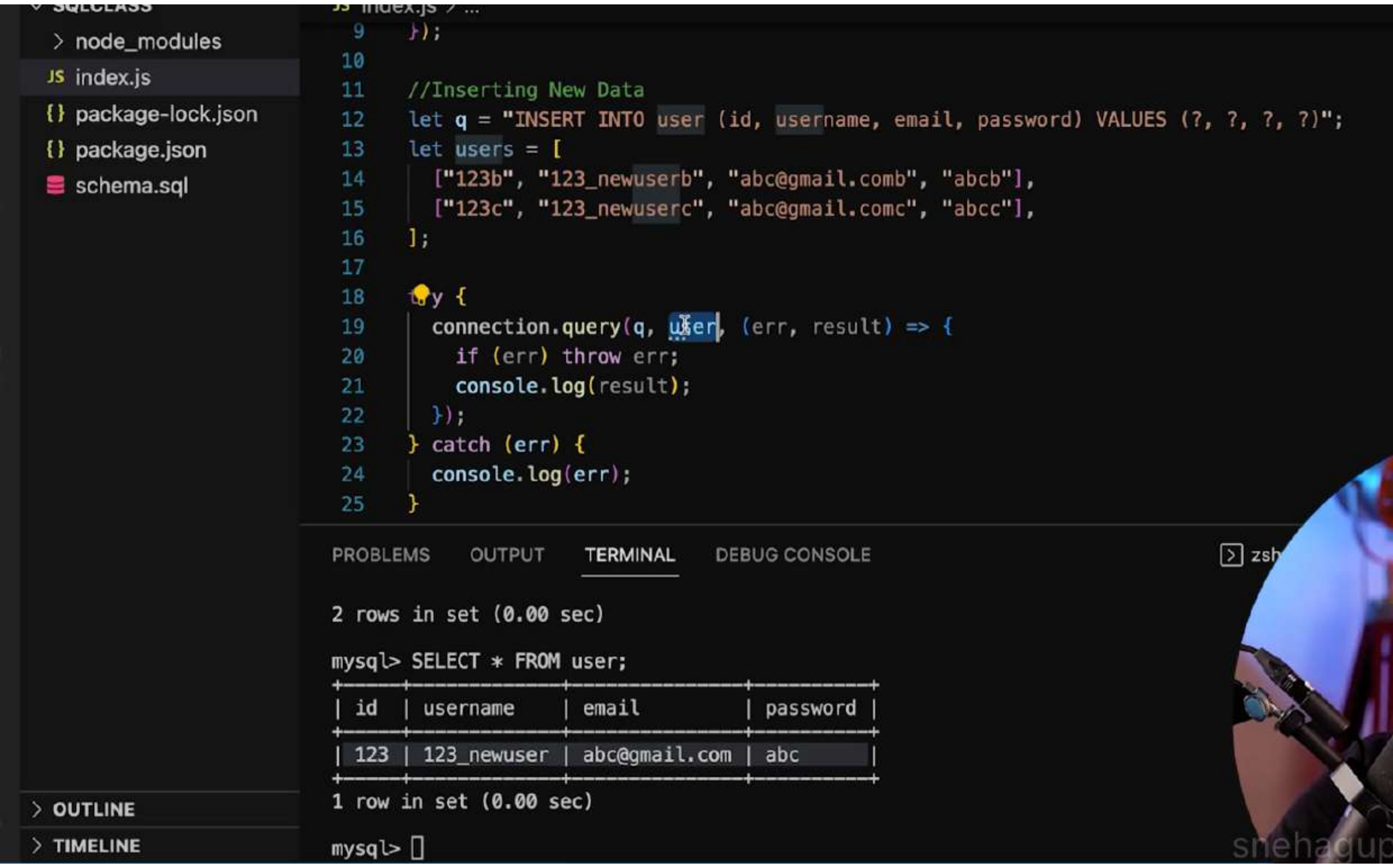


```javascript
  database: "delta_app",
  password: "mysql@123",
});

//Inserting New Data
let q = "INSERT INTO user (id, username, email, password) VALUES (?, ?, ?, ?)";
let user = ["123", "123_newuser", "abc@gmail.com", "abc"];

try {
  connection.query(q, user, (err, result) => {
    if (err) throw err;
    console.log(result);
  });
} catch (err) {
  console.log(err);
}
```

```
shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
ResultSetHeader {
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  info: '',
  serverStatus: 2,
  warningStatus: 0,
  changedRows: 0
}
shradhakhapra@Shradhas-MacBook-Air SQLClass %
```

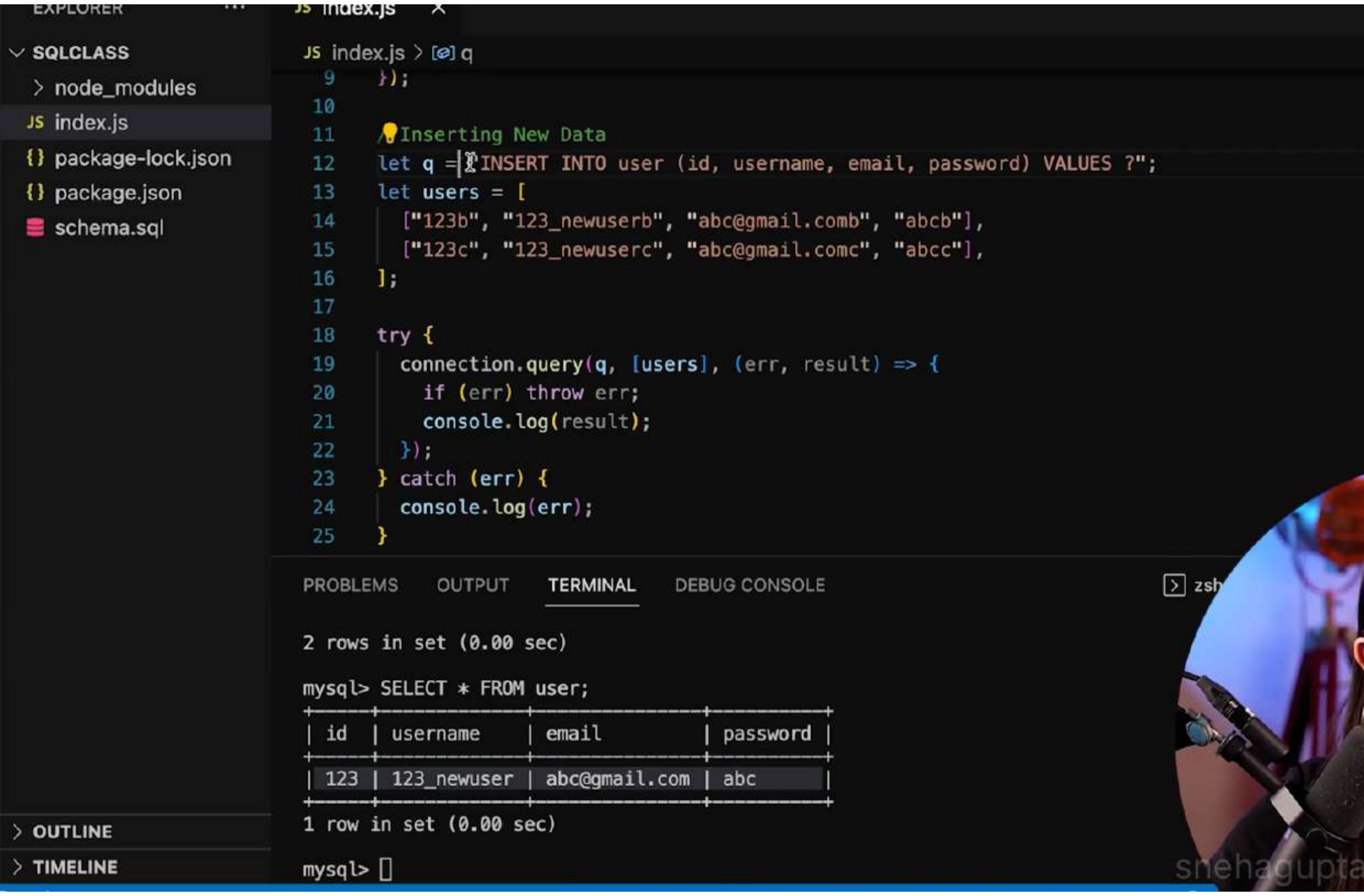


```
2 rows in set (0.00 sec)

mysql> SELECT * FROM user;
+-----+-------------+---------------+----------+
| id  | username    | email         | password |
+-----+-------------+---------------+----------+
| 123 | 123_newuser | abc@gmail.com | abc      |
+-----+-------------+---------------+----------+
1 row in set (0.00 sec)

mysql> 
```



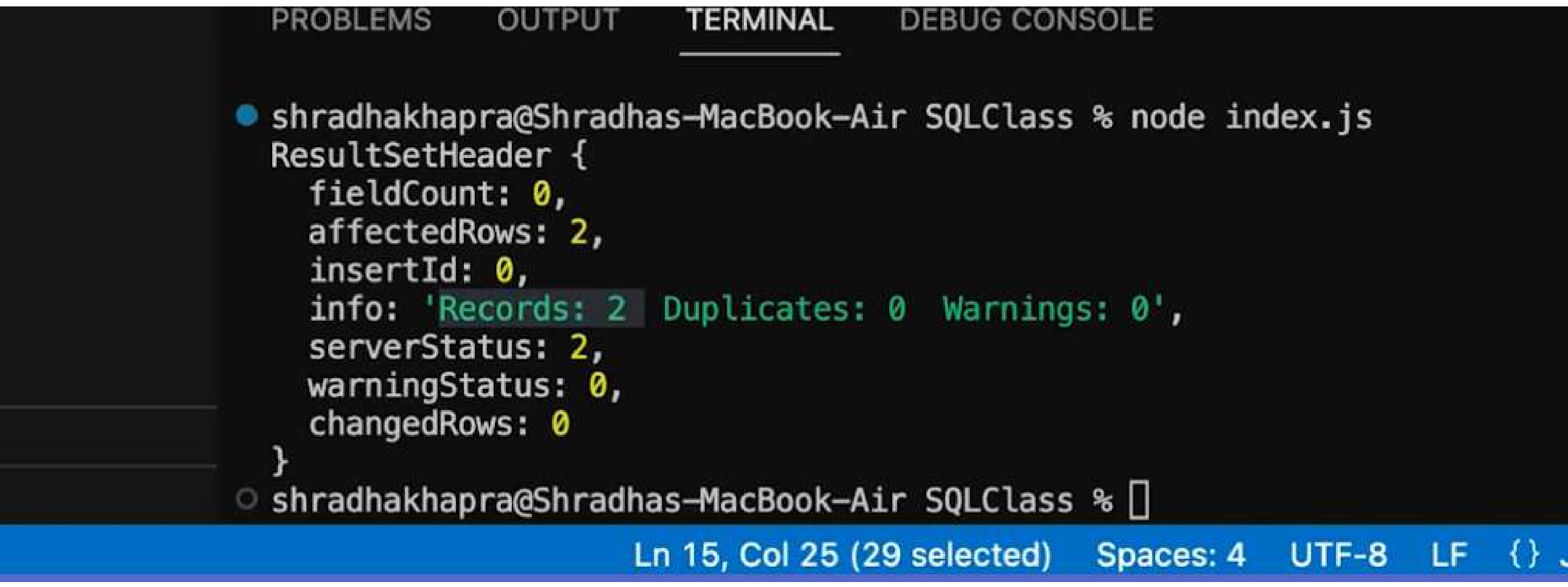
```
});

//Inserting New Data
let q = "INSERT INTO user (id, username, email, password) VALUES (?, ?, ?, ?)";
let users = [
  ["123b", "123_newuserb", "abc@gmail.comb", "abcb"],
  ["123c", "123_newuserc", "abc@gmail.comc", "abcc"],
];

try {
  connection.query(q, user, (err, result) => {
    if (err) throw err;
    console.log(result);
  });
} catch (err) {
  console.log(err);
}
```

```
2 rows in set (0.00 sec)

mysql> SELECT * FROM user;
+-----+-------------+---------------+----------+
| id  | username    | email         | password |
+-----+-------------+---------------+----------+
| 123 | 123_newuser | abc@gmail.com | abc      |
+-----+-------------+---------------+----------+
1 row in set (0.00 sec)

mysql> 
```

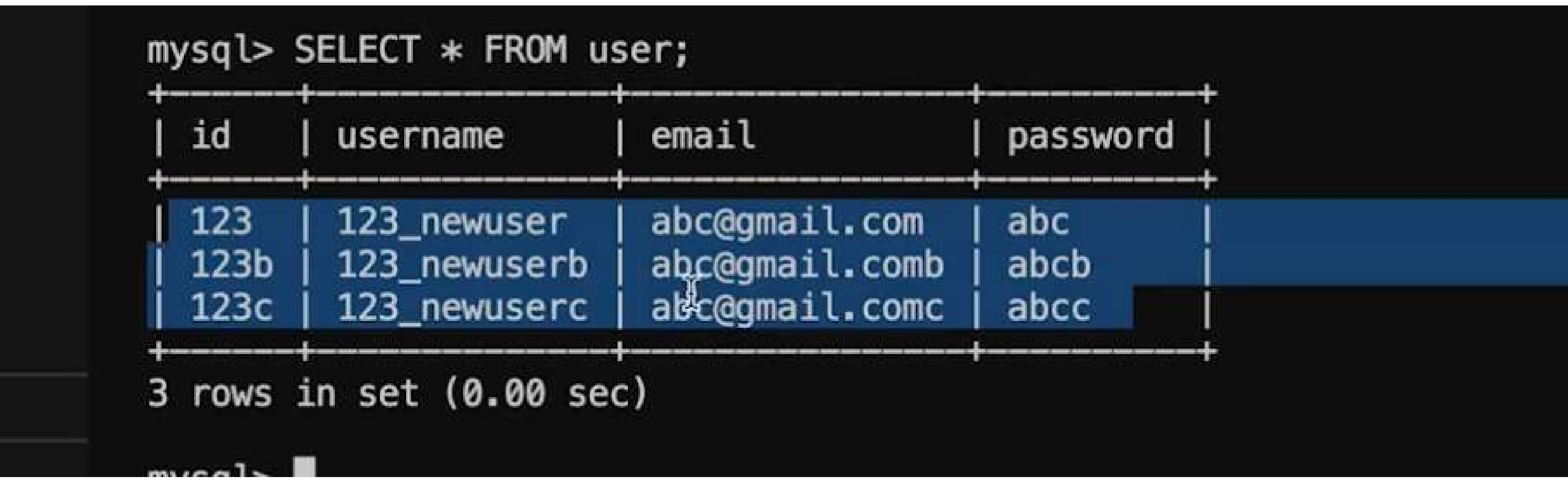


```
});

//Inserting New Data
let q = "INSERT INTO user (id, username, email, password) VALUES ?";
let users = [
  ["123b", "123_newuserb", "abc@gmail.comb", "abcb"],
  ["123c", "123_newuserc", "abc@gmail.comc", "abcc"],
];

try {
  connection.query(q, [users], (err, result) => {
    if (err) throw err;
    console.log(result);
  });
} catch (err) {
  console.log(err);
}
```

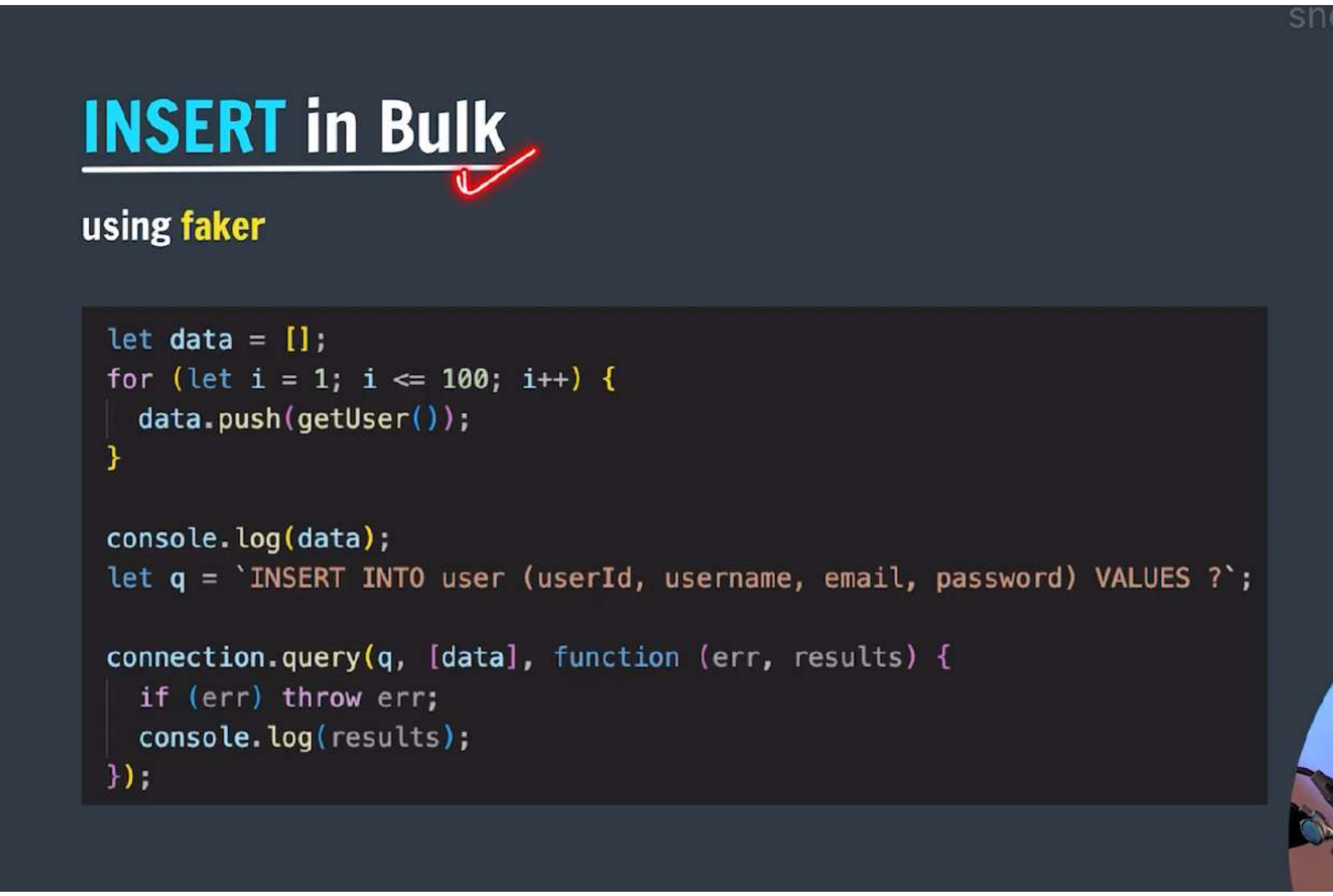
```
2 rows in set (0.00 sec)

mysql> SELECT * FROM user;
+-----+-------------+---------------+----------+
| id  | username    | email         | password |
+-----+-------------+---------------+----------+
| 123 | 123_newuser | abc@gmail.com | abc      |
+-----+-------------+---------------+----------+
1 row in set (0.00 sec)

mysql> 
```



```
shradhakhapra@Shradhas-MacBook-Air SQLClass % node index.js
ResultSetHeader {
  fieldCount: 0,
  affectedRows: 2,
  insertId: 0,
  info: 'Records: 2  Duplicates: 0  Warnings: 0',
  serverStatus: 2,
  warningStatus: 0,
  changedRows: 0
}
shradhakhapra@Shradhas-MacBook-Air SQLClass % 
```



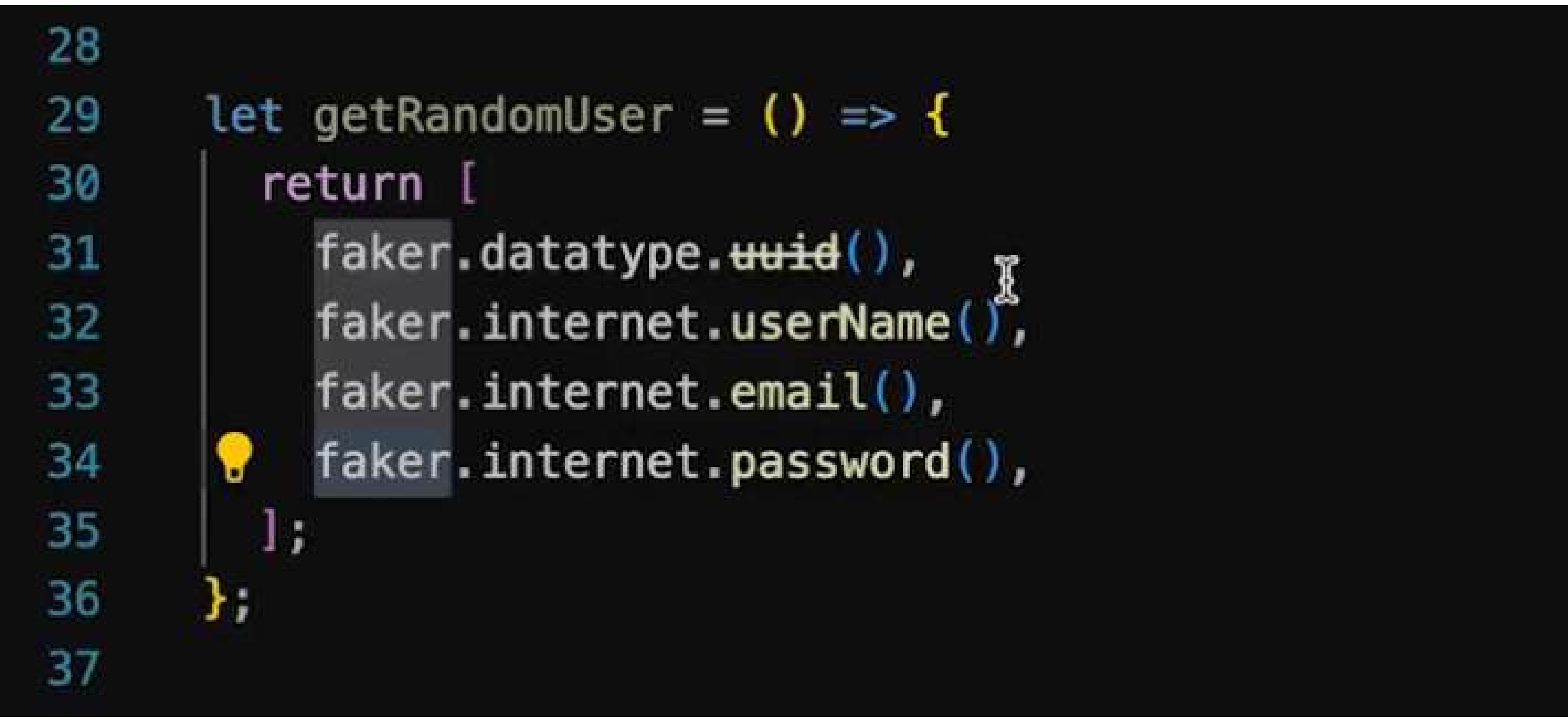
```
mysql> SELECT * FROM user;
+------+--------------+-----------------+----------+
| id   | username     | email           | password |
+------+--------------+-----------------+----------+
| 123  | 123_newuser  | abc@gmail.com   | abc      |
| 123b | 123_newuserb | abc@gmail.comb  | abcb     |
| 123c | 123_newuserc | abc@gmail.comc  | abcc     |
+------+--------------+-----------------+----------+
3 rows in set (0.00 sec)

mysql>
```



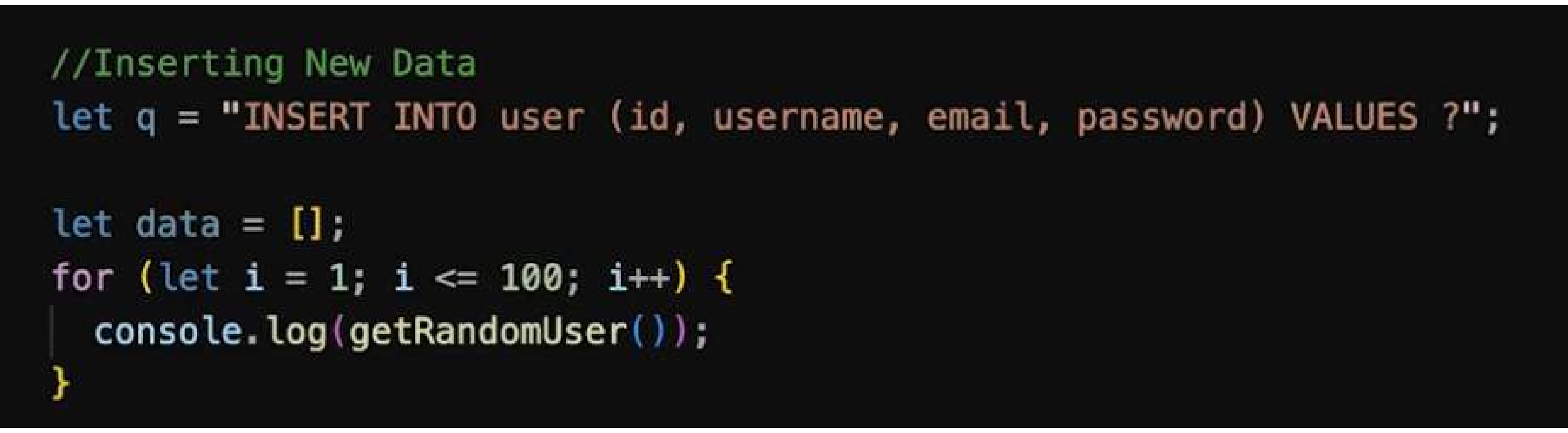
INSERT in Bulk

using **faker**

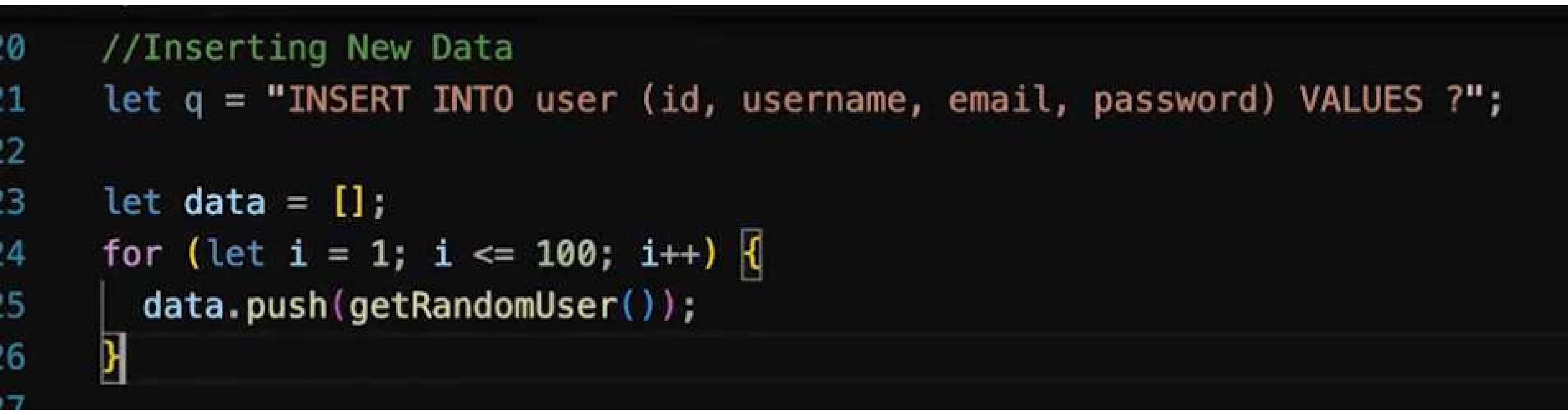
```
let data = [];
for (let i = 1; i <= 100; i++) {
  data.push(getUser());
}

console.log(data);
let q = `INSERT INTO user (userId, username, email, password) VALUES ?`;

connection.query(q, [data], function (err, results) {
  if (err) throw err;
  console.log(results);
});
```



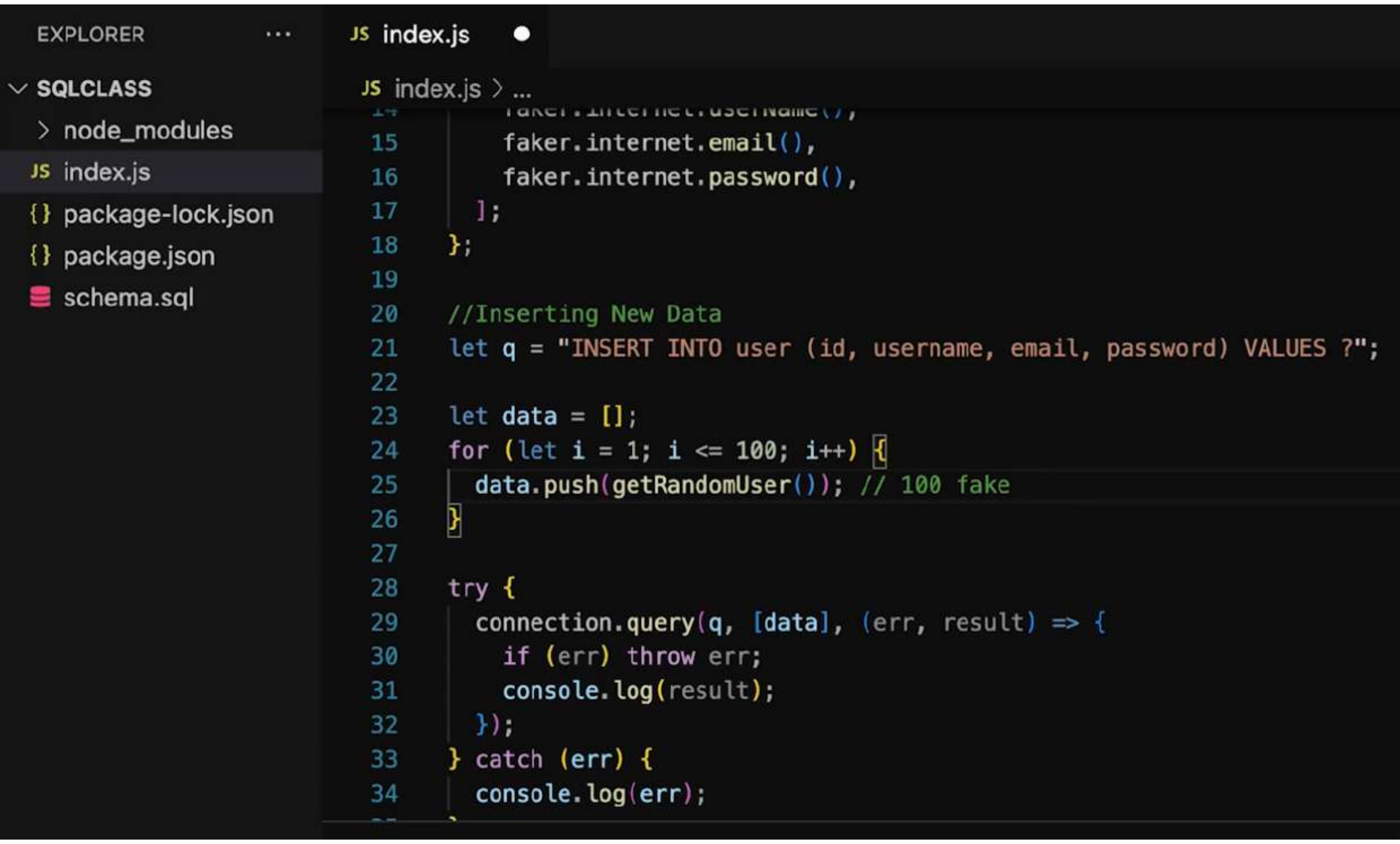
```
let getRandomUser = () => {
  return [
    faker.datatype.~~uuid~~(),
    faker.internet.userName(),
    faker.internet.email(),
    faker.internet.password(),
  ];
};
```



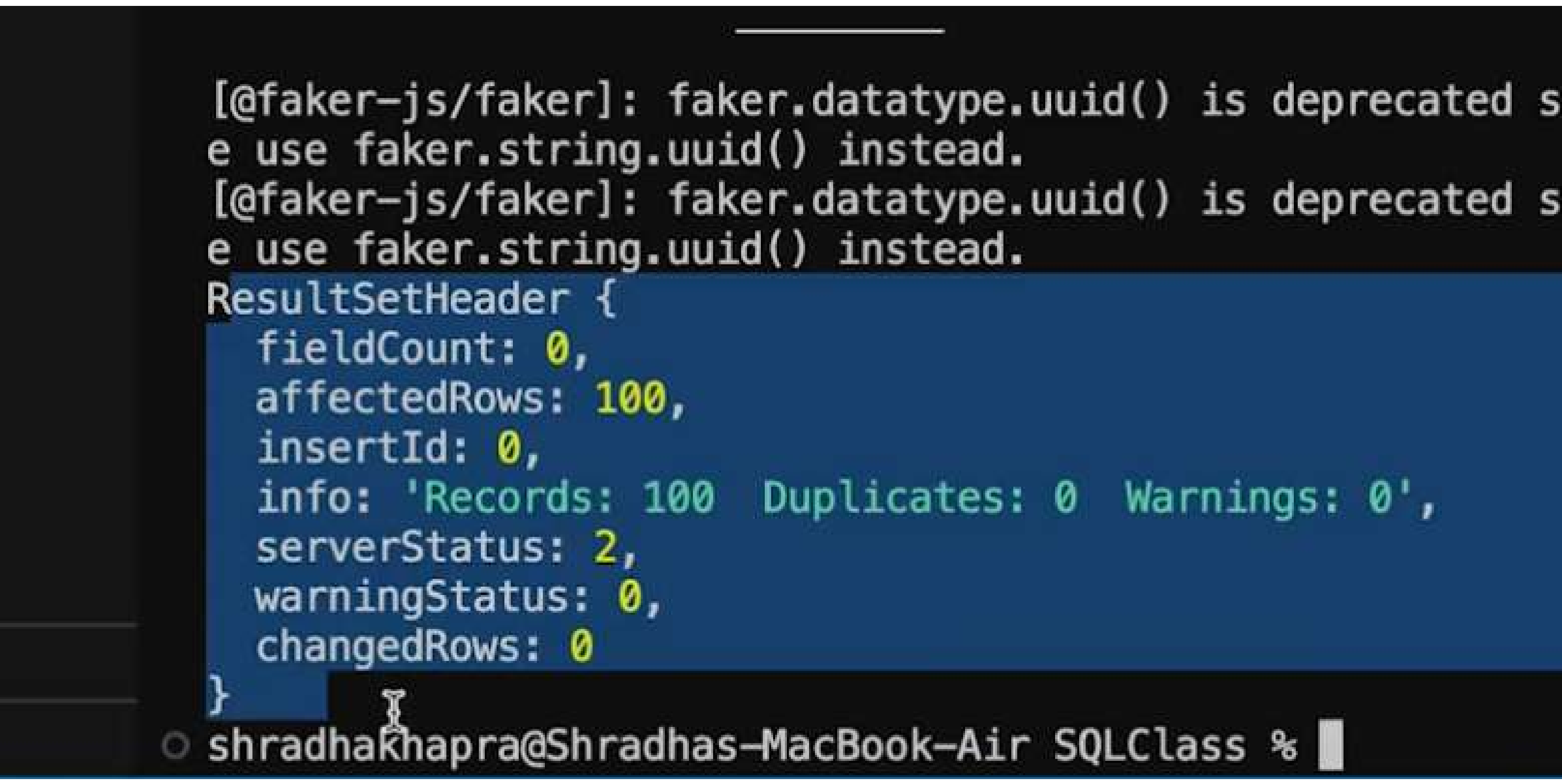
```
//Inserting New Data
let q = "INSERT INTO user (id, username, email, password) VALUES ?";

let data = [];
for (let i = 1; i <= 100; i++) {
  console.log(getRandomUser());
}
```



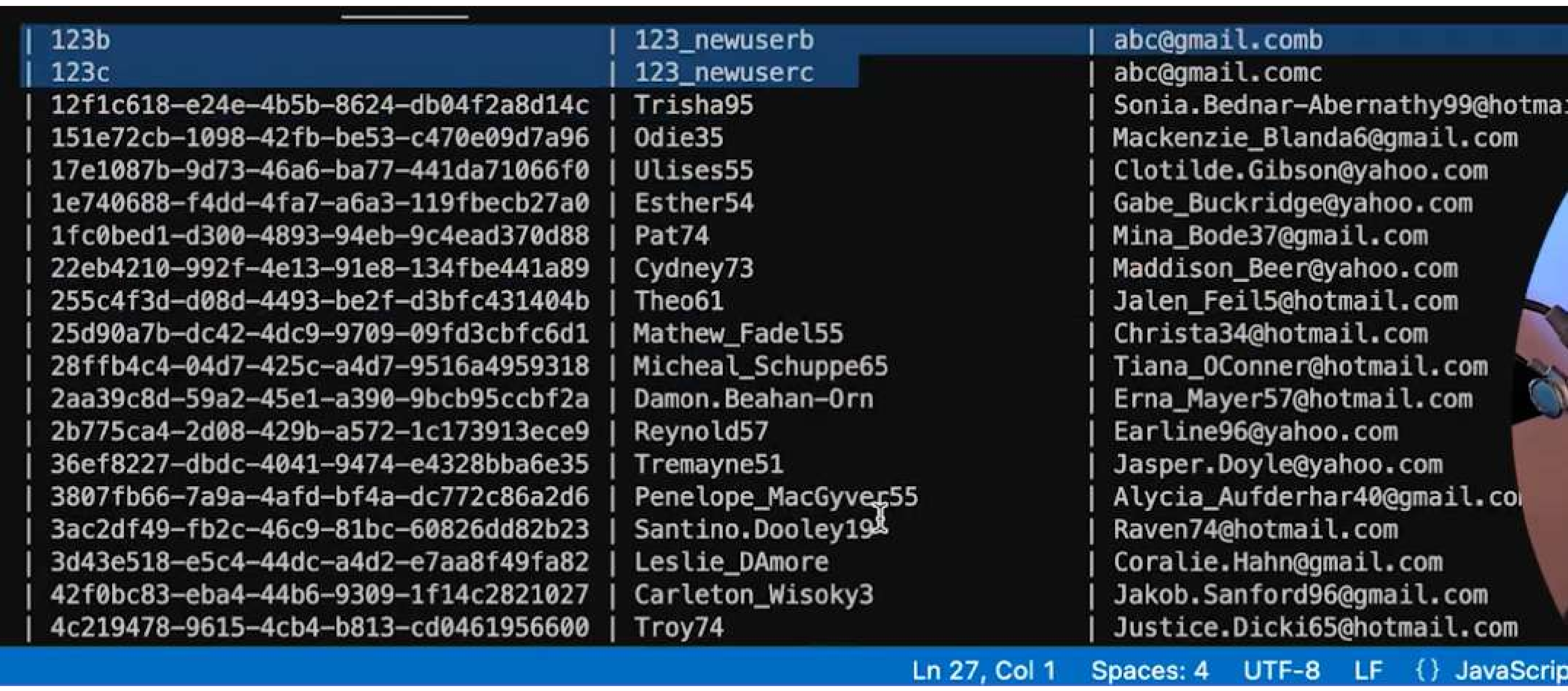
```
//Inserting New Data
let q = "INSERT INTO user (id, username, email, password) VALUES ?";

let data = [];
for (let i = 1; i <= 100; i++) {
  data.push(getRandomUser());
}
```



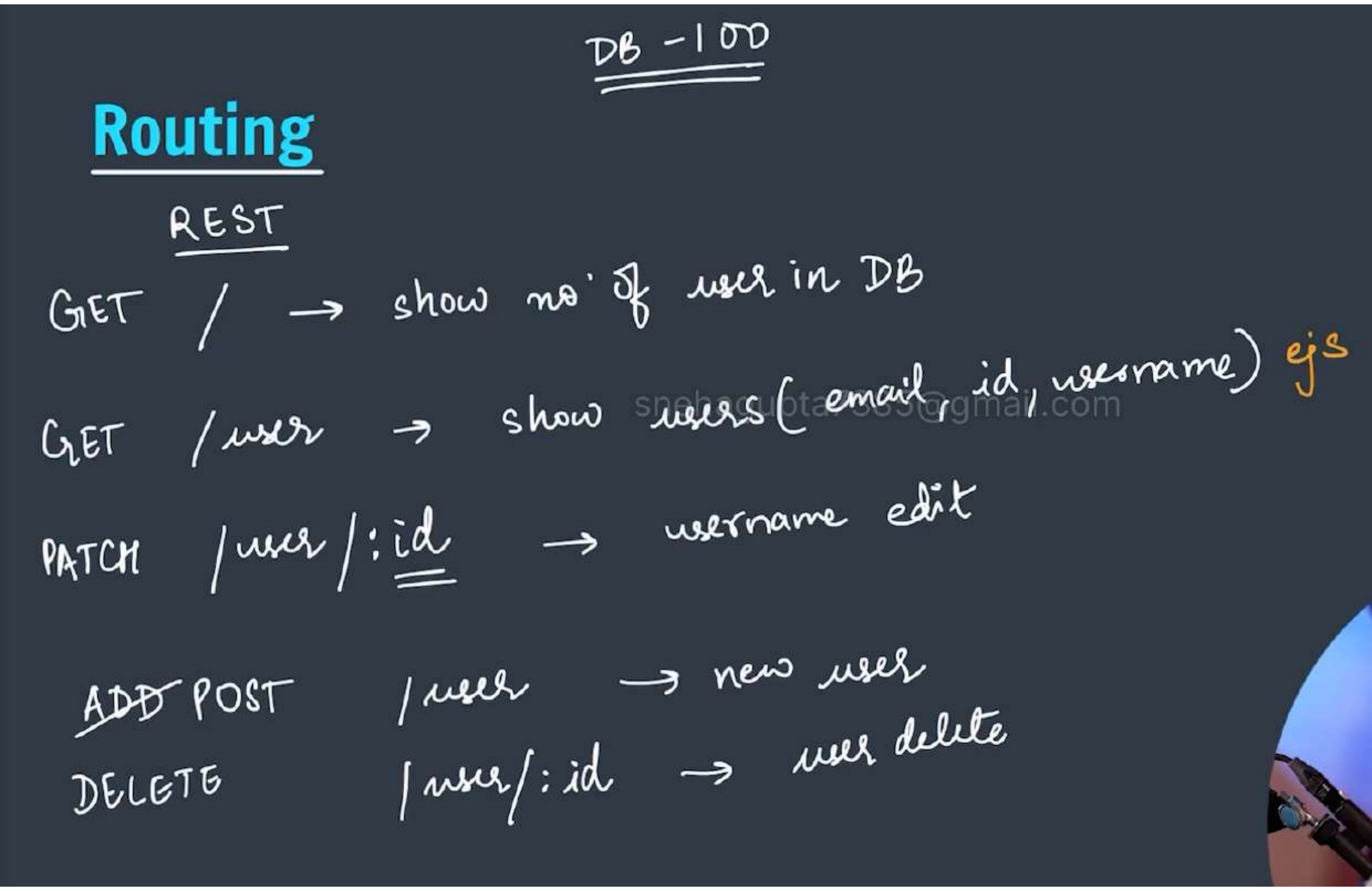
```
      faker.internet.userName(),
      faker.internet.email(),
      faker.internet.password(),
    ];
};

//Inserting New Data
let q = "INSERT INTO user (id, username, email, password) VALUES ?";

let data = [];
for (let i = 1; i <= 100; i++) {
  data.push(getRandomUser()); // 100 fake
}

try {
  connection.query(q, [data], (err, result) => {
    if (err) throw err;
    console.log(result);
  });
} catch (err) {
  console.log(err);
```



```
[@faker-js/faker]: faker.datatype.uuid() is deprecated s
e use faker.string.uuid() instead.
[@faker-js/faker]: faker.datatype.uuid() is deprecated s
e use faker.string.uuid() instead.
ResultSetHeader {
  fieldCount: 0,
  affectedRows: 100,
  insertId: 0,
  info: 'Records: 100  Duplicates: 0  Warnings: 0',
  serverStatus: 2,
  warningStatus: 0,
  changedRows: 0
}
○ shradhakhapra@Shradhas-MacBook-Air SQLClass %
```



```
| 123b                                 | 123_newuserb          | abc@gmail.comb
| 123c                                 | 123_newuserc          | abc@gmail.comc
| 12f1c618-e24e-4b5b-8624-db04f2a8d14c | Trisha95              | Sonia.Bednar-Abernathy99@hotma
| 151e72cb-1098-42fb-be53-c470e09d7a96 | Odie35                | Mackenzie_Blanda6@gmail.com
| 17e1087b-9d73-46a6-ba77-441da71066f0 | Ulises55              | Clotilde.Gibson@yahoo.com
| 1e740688-f4dd-4fa7-a6a3-119fbecb27a0 | Esther54              | Gabe_Buckridge@yahoo.com
| 1fc0bed1-d300-4893-94eb-9c4ead370d88 | Pat74                 | Mina_Bode37@gmail.com
| 22eb4210-992f-4e13-91e8-134fbe441a89 | Cydney73              | Maddison_Beer@yahoo.com
| 255c4f3d-d08d-4493-be2f-d3bfc431404b | Theo61                | Jalen_Feil5@hotmail.com
| 25d90a7b-dc42-4dc9-9709-09fd3cbfc6d1 | Mathew_Fadel55        | Christa34@hotmail.com
| 28ffb4c4-04d7-425c-a4d7-9516a4959318 | Micheal_Schuppe65     | Tiana_OConner@hotmail.com
| 2aa39c8d-59a2-45e1-a390-9bcb95ccbf2a | Damon.Beahan-Orn      | Erna_Mayer57@hotmail.com
| 2b775ca4-2d08-429b-a572-1c173913ece9 | Reynold57             | Earline96@yahoo.com
| 36ef8227-dbdc-4041-9474-e4328bba6e35 | Tremayne51            | Jasper.Doyle@yahoo.com
| 3807fb66-7a9a-4afd-bf4a-dc772c86a2d6 | Penelope_MacGyver55   | Alycia_Aufderhar40@gmail.co
| 3ac2df49-fb2c-46c9-81bc-60826dd82b23 | Santino.Dooley19      | Raven74@hotmail.com
| 3d43e518-e5c4-44dc-a4d2-e7aa8f49fa82 | Leslie_DAmore         | Coralie.Hahn@gmail.com
| 42f0bc83-eba4-44b6-9309-1f14c2821027 | Carleton_Wisoky3      | Jakob.Sanford96@gmail.com
| 4c219478-9615-4cb4-b813-cd0461956600 | Troy74                | Justice.Dicki65@hotmail.com
```



DB - 100

Routing

REST

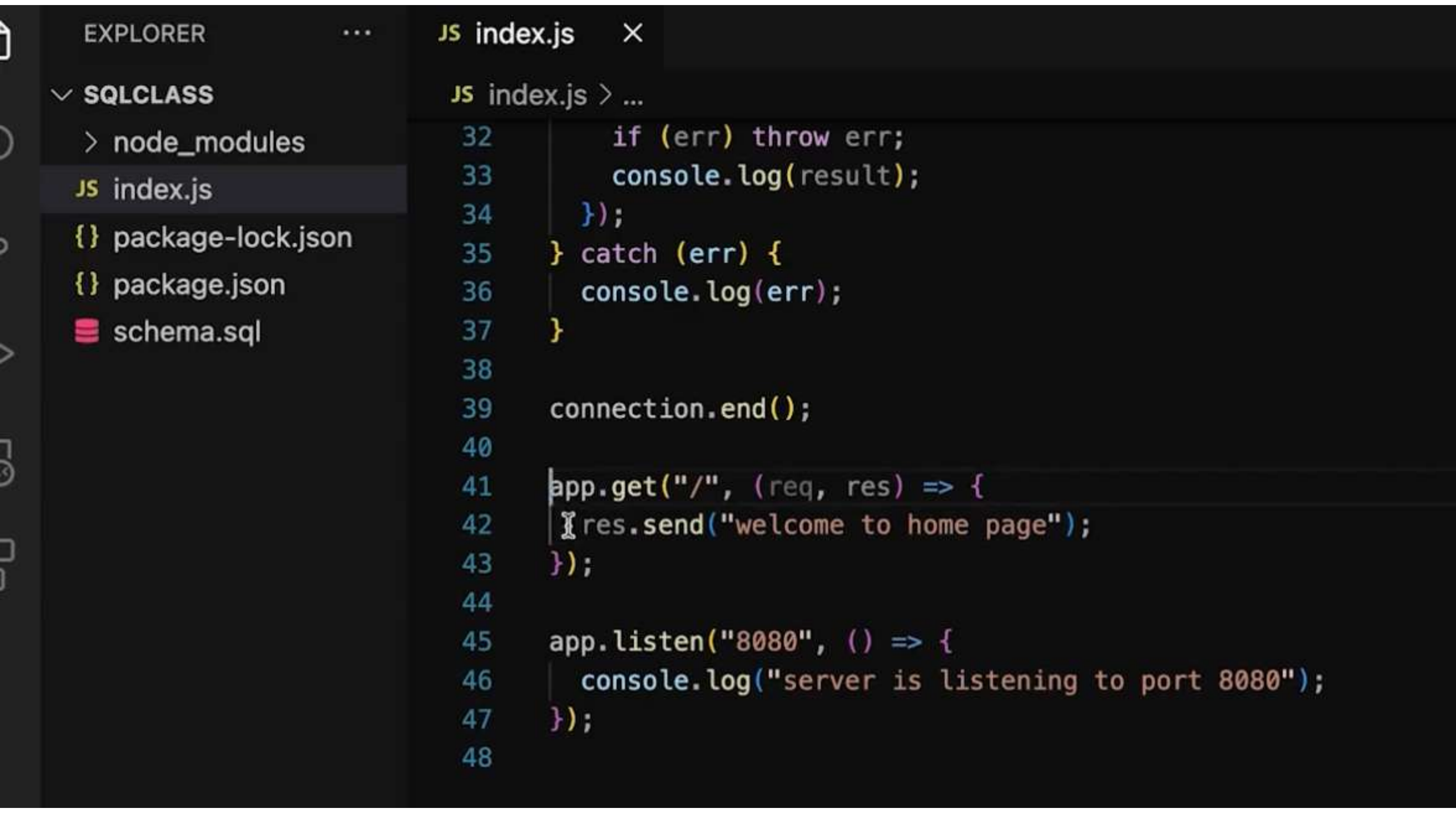
GET / → show no. of user in DB

GET /user → show users (email, id, username) ejs

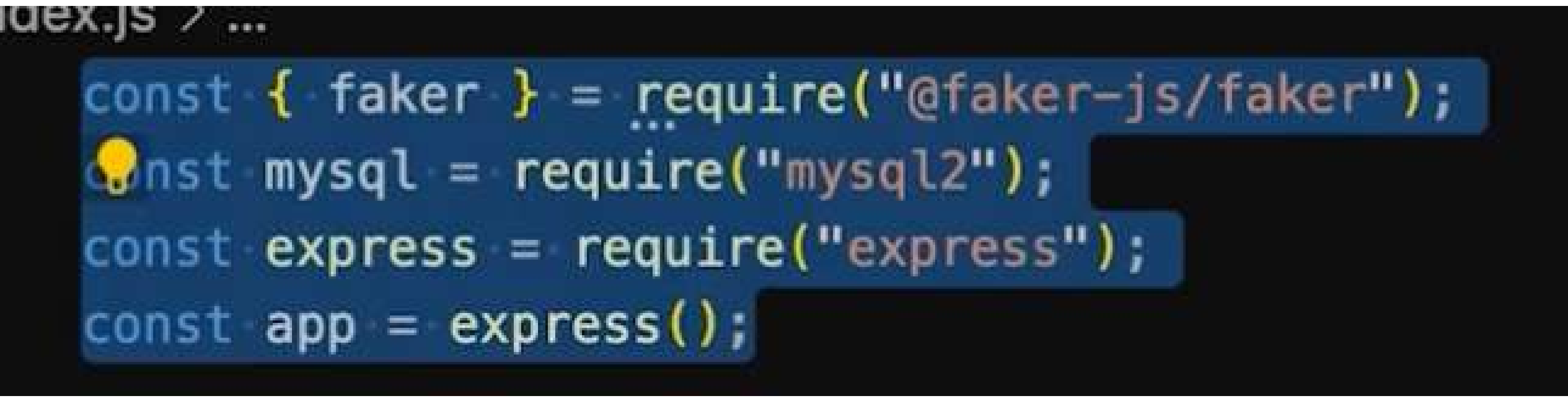
PATCH /user/:id → username edit

~~ADD~~ POST /user → new user

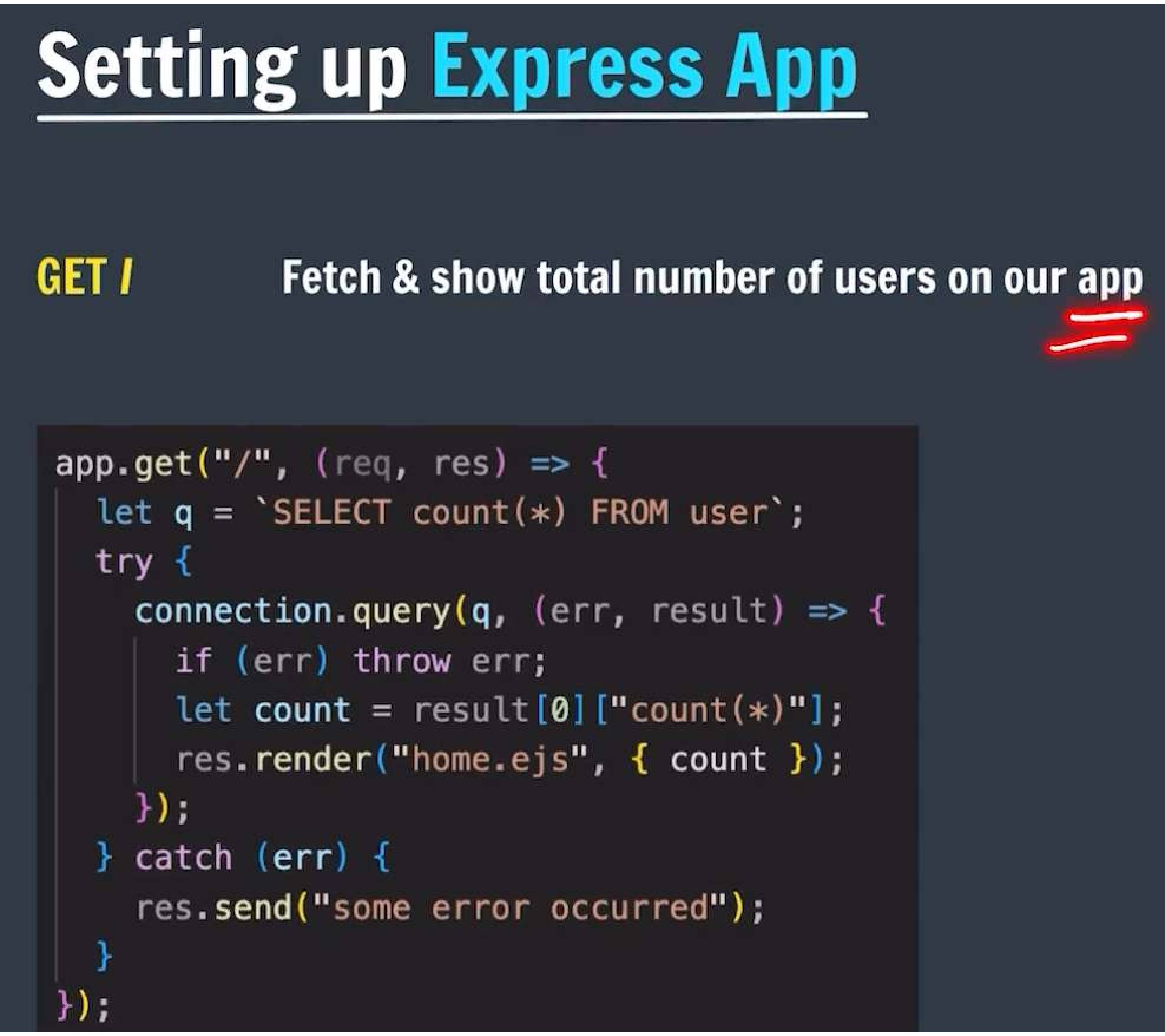
DELETE /user/:id → user delete



```
    if (err) throw err;
    console.log(result);
  });
} catch (err) {
  console.log(err);
}

connection.end();

app.get("/", (req, res) => {
  res.send("welcome to home page");
});

app.listen("8080", () => {
  console.log("server is listening to port 8080");
});

```



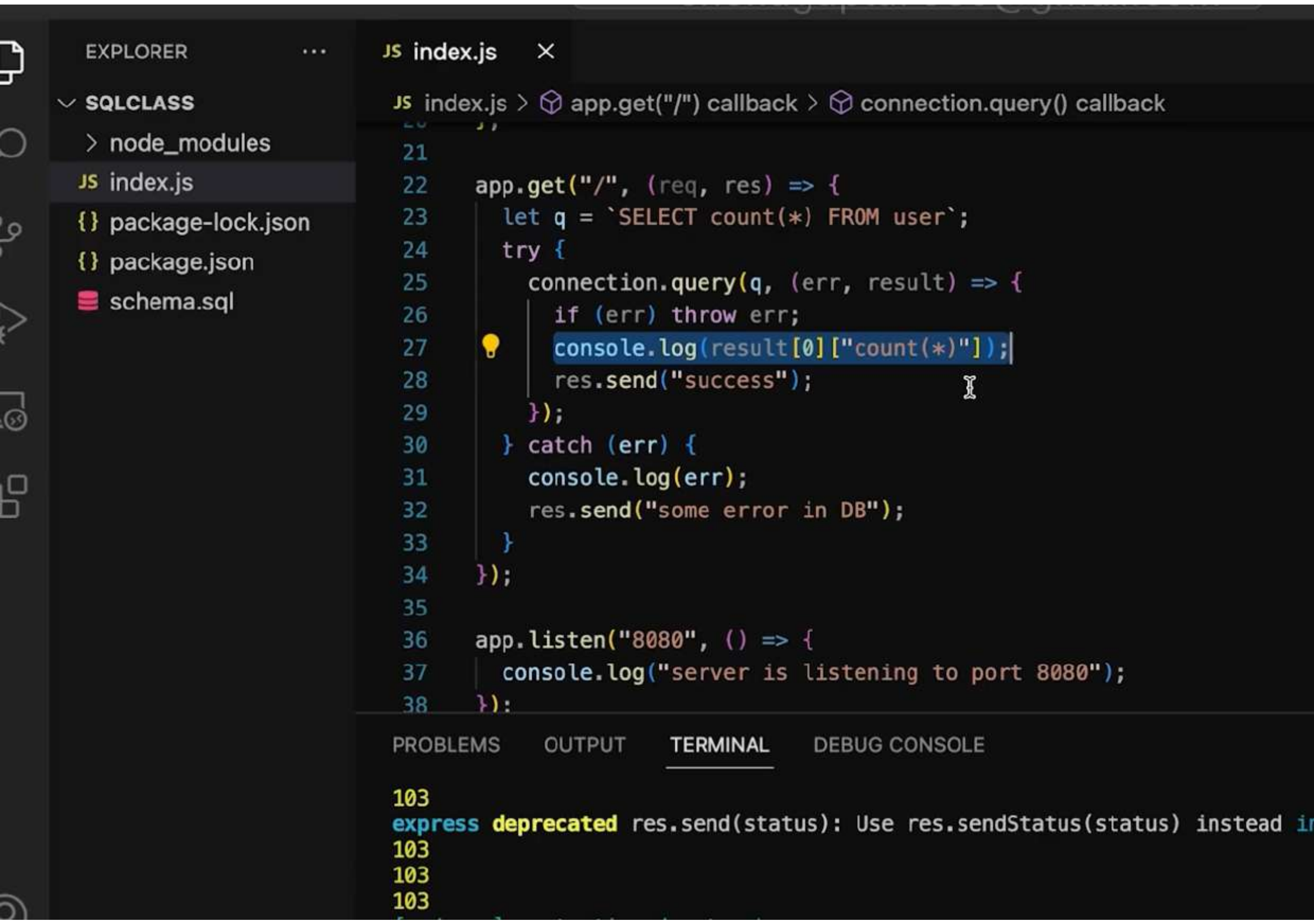
```
dex.js > ...
const { faker } = require("@faker-js/faker");
const mysql = require("mysql2");
const express = require("express");
const app = express();
```



Setting up Express App

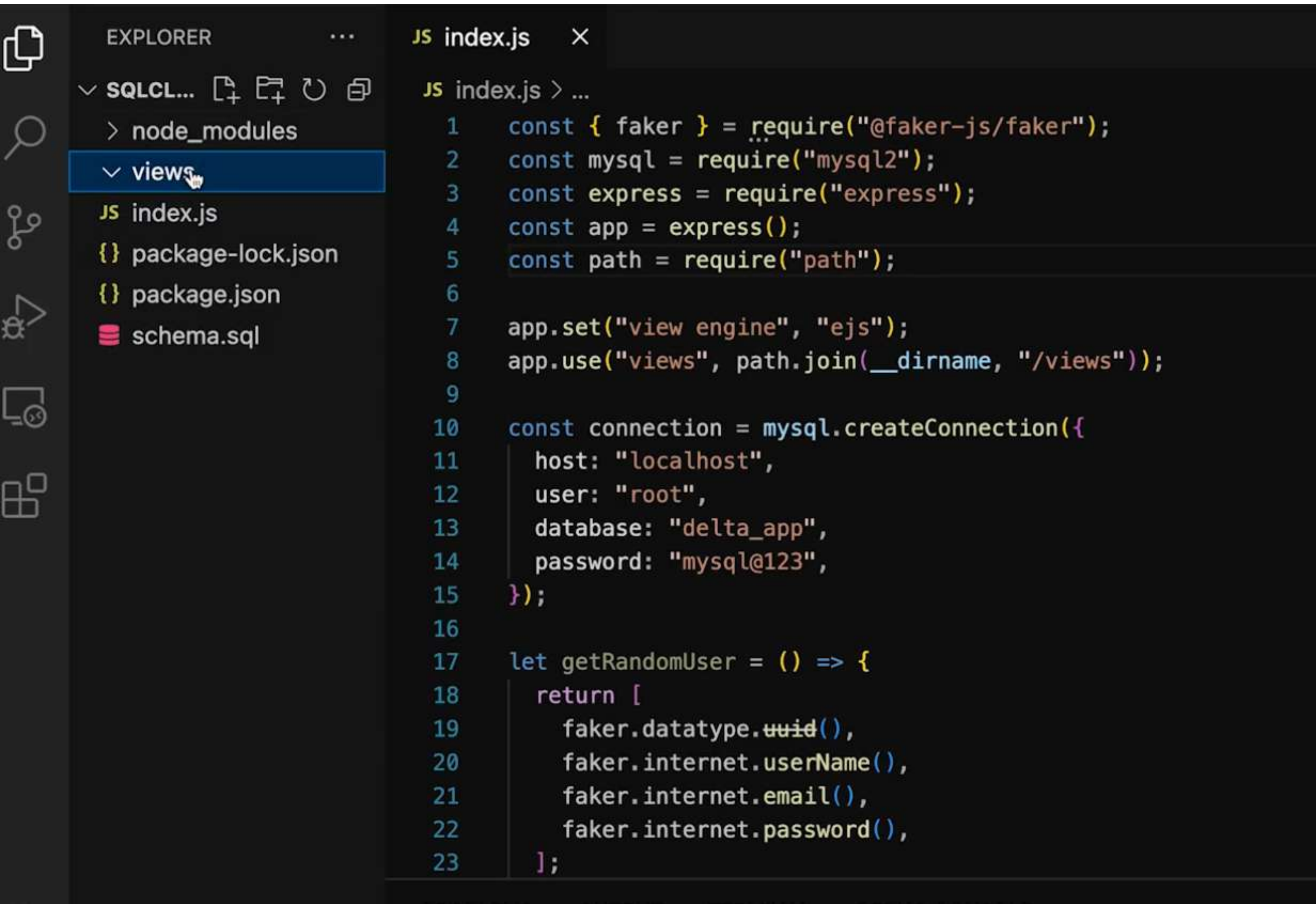
GET / Fetch & show total number of users on our app

```
app.get("/", (req, res) => {
  let q = `SELECT count(*) FROM user`;
  try {
    connection.query(q, (err, result) => {
      if (err) throw err;
      let count = result[0]["count(*)"];
      res.render("home.ejs", { count });
    });
  } catch (err) {
    res.send("some error occurred");
  }
});
```

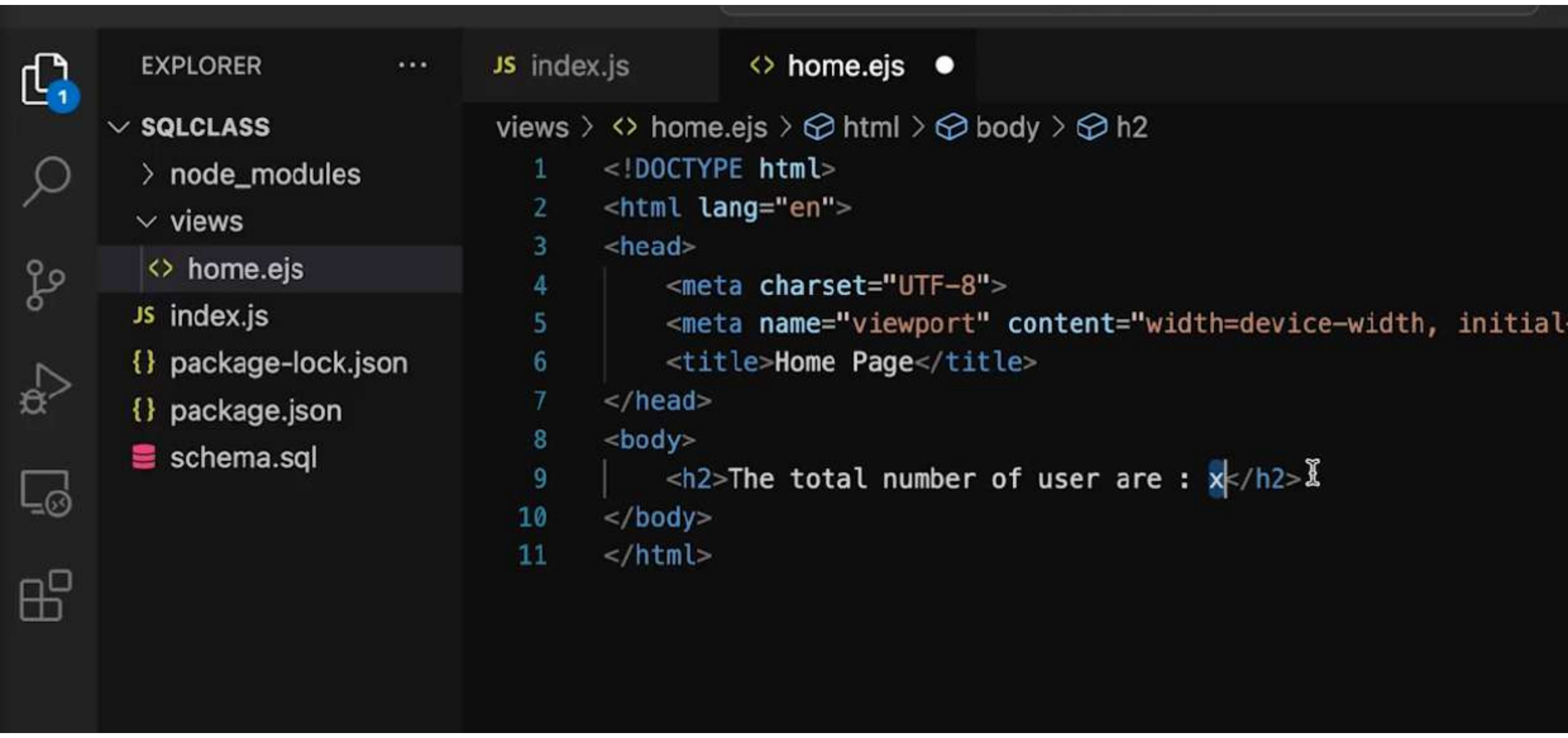


```javascript
app.get("/", (req, res) => {
  let q = `SELECT count(*) FROM user`;
  try {
    connection.query(q, (err, result) => {
      if (err) throw err;
      console.log(result[0]["count(*)"]);
      res.send("success");
    });
  } catch (err) {
    console.log(err);
    res.send("some error in DB");
  }
});

app.listen("8080", () => {
  console.log("server is listening to port 8080");
});
```

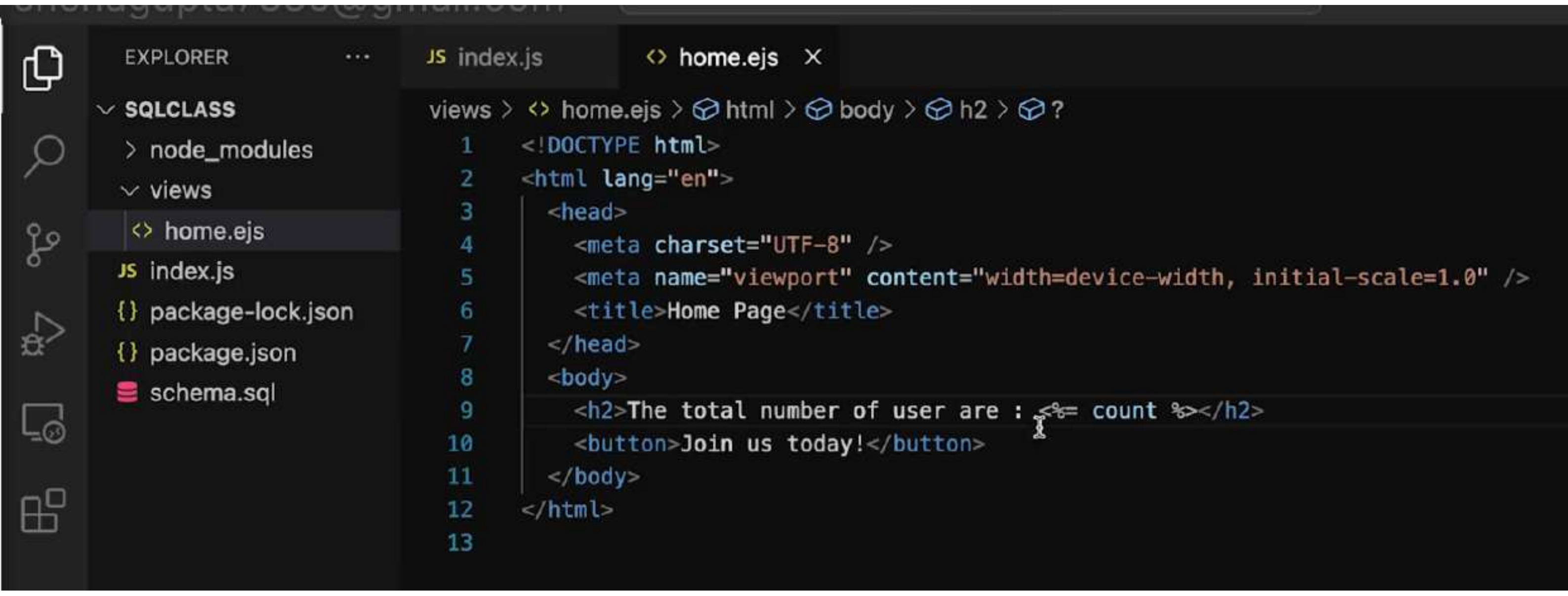
```
103
express deprecated res.send(status): Use res.sendStatus(status) instead i
103
103
103
```



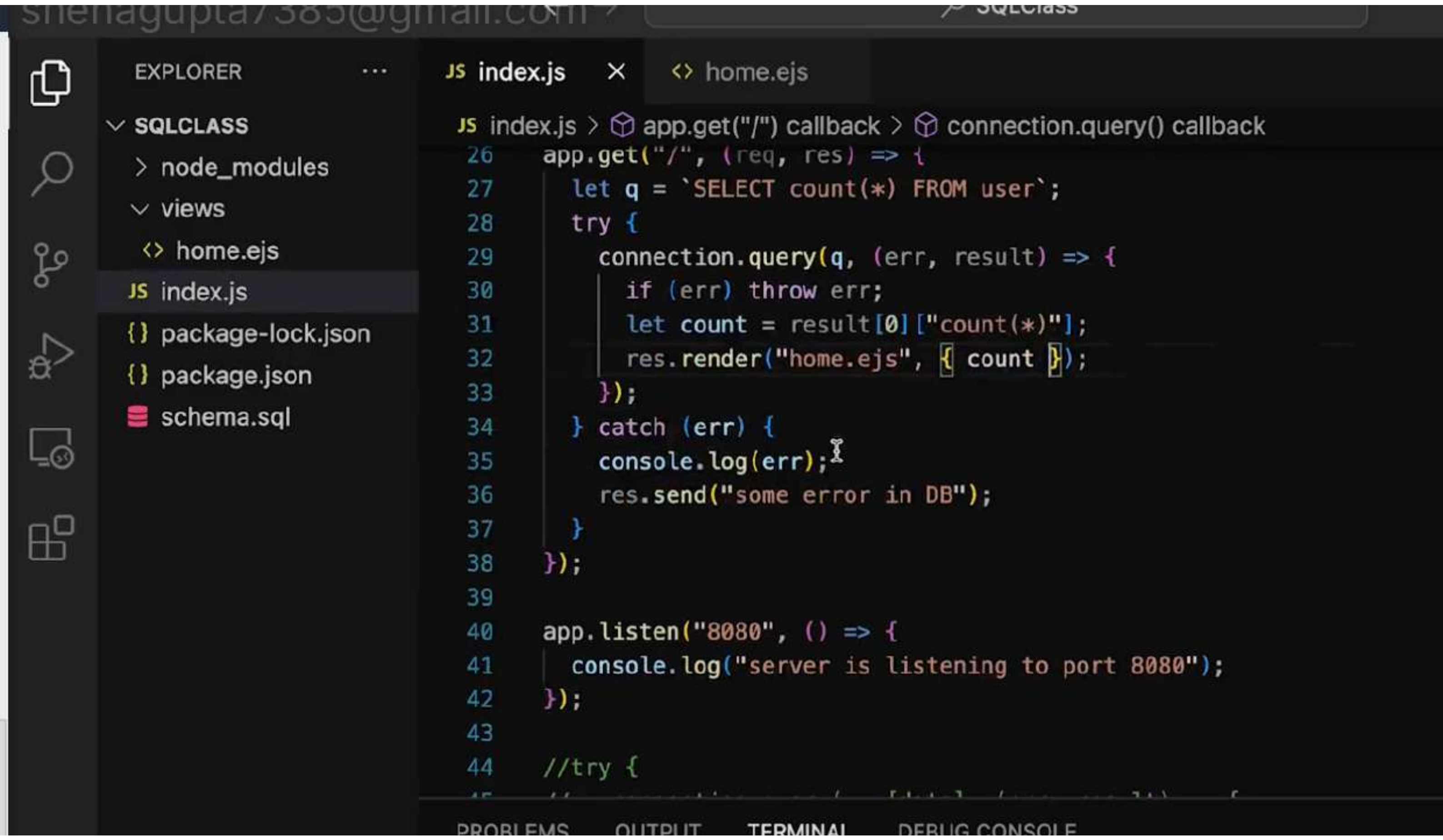
```javascript
const { faker } = require("@faker-js/faker");
const mysql = require("mysql2");
const express = require("express");
const app = express();
const path = require("path");

app.set("view engine", "ejs");
app.use("views", path.join(__dirname, "/views"));

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "delta_app",
  password: "mysql@123",
});

let getRandomUser = () => {
  return [
    faker.datatype.~~uuid~~(),
    faker.internet.userName(),
    faker.internet.email(),
    faker.internet.password(),
  ];
```



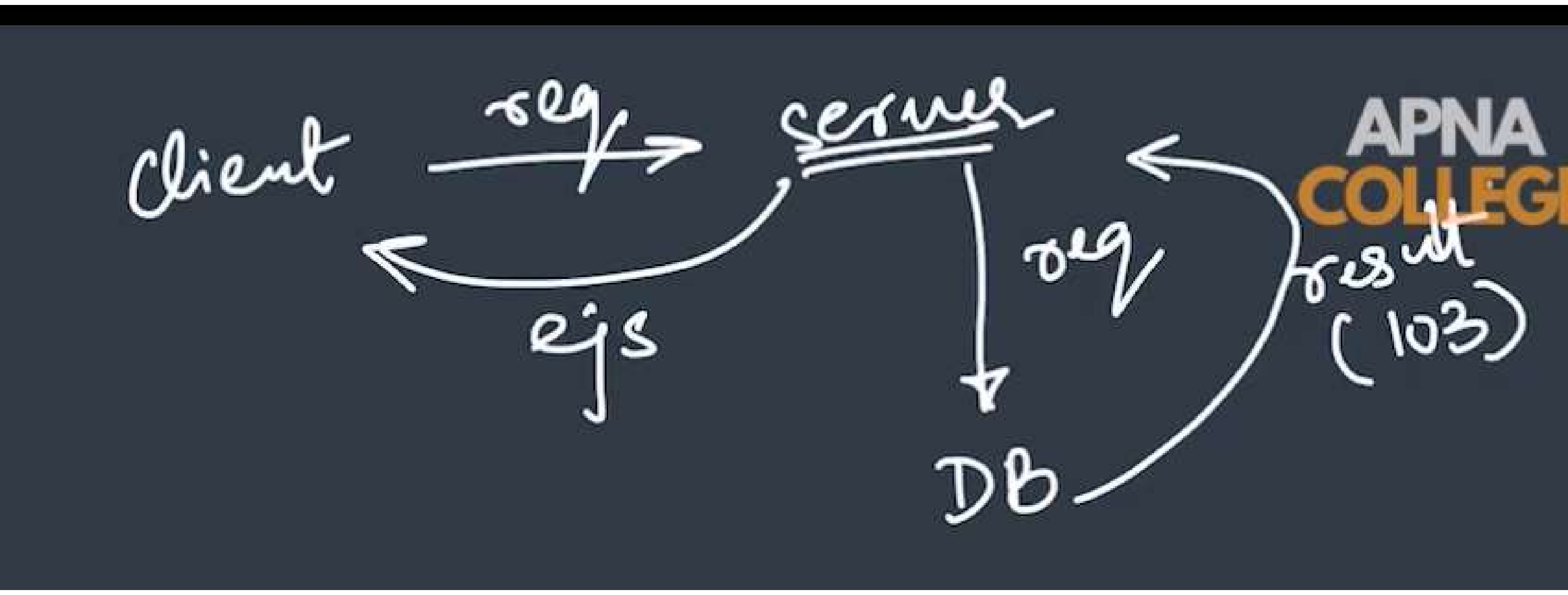
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial
    <title>Home Page</title>
</head>
<body>
    <h2>The total number of user are : x</h2>
</body>
</html>
```



```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Home Page</title>
  </head>
  <body>
    <h2>The total number of user are : <%= count %></h2>
    <button>Join us today!</button>
  </body>
</html>
```

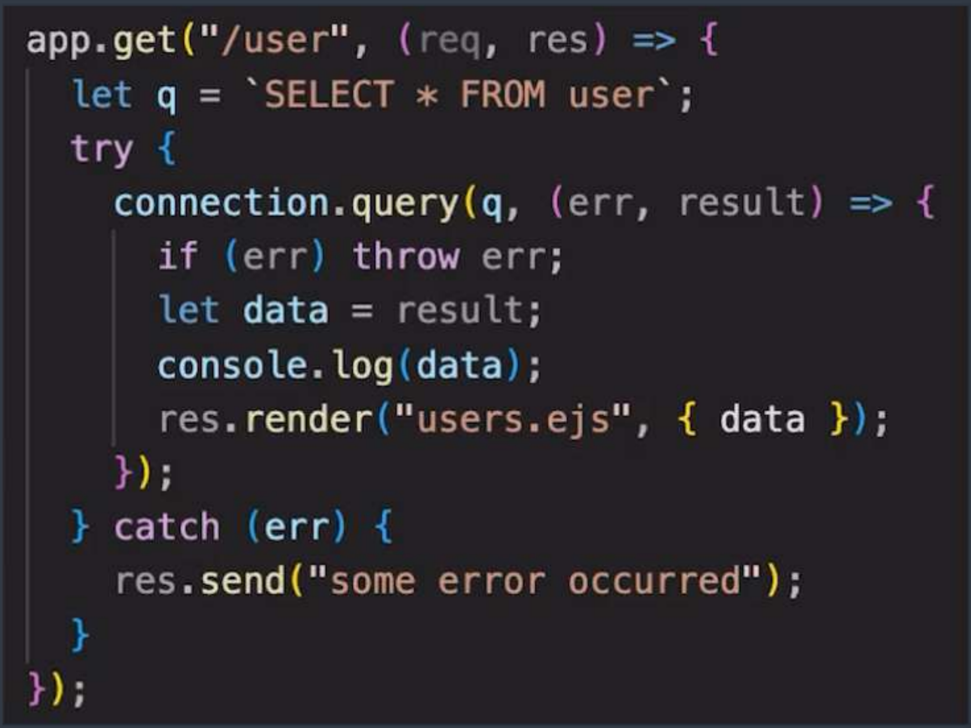


```
app.get("/", (req, res) => {
  let q = `SELECT count(*) FROM user`;
  try {
    connection.query(q, (err, result) => {
      if (err) throw err;
      let count = result[0]["count(*)"];
      res.render("home.ejs", { count });
    });
  } catch (err) {
    console.log(err);
    res.send("some error in DB");
  }
});

app.listen("8080", () => {
  console.log("server is listening to port 8080");
});

//try {
```

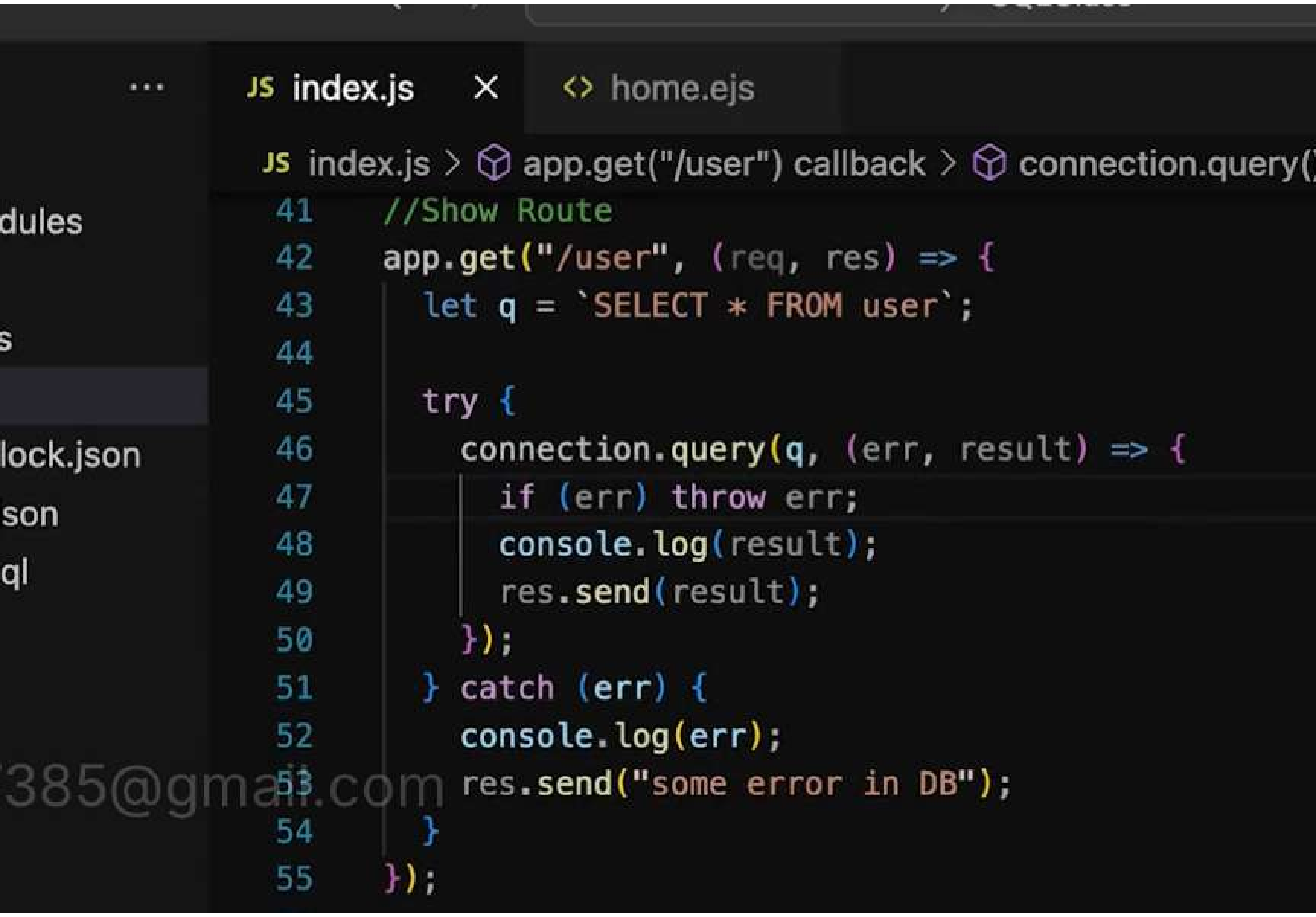

Client
req
server
ejs
req
DB
result
(103)

Creating Our Routes

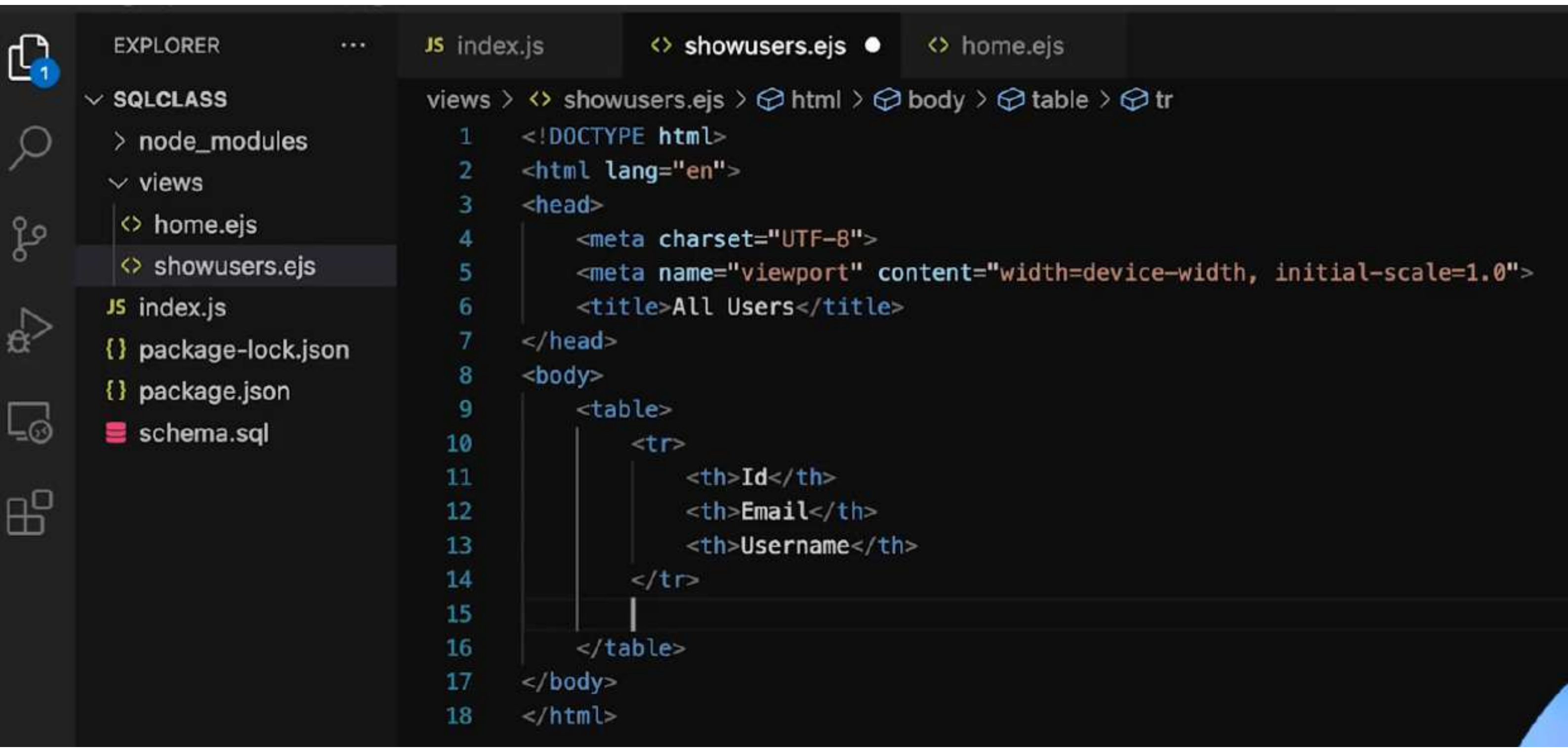
GET /user Fetch & show (userId, username, email) of all users



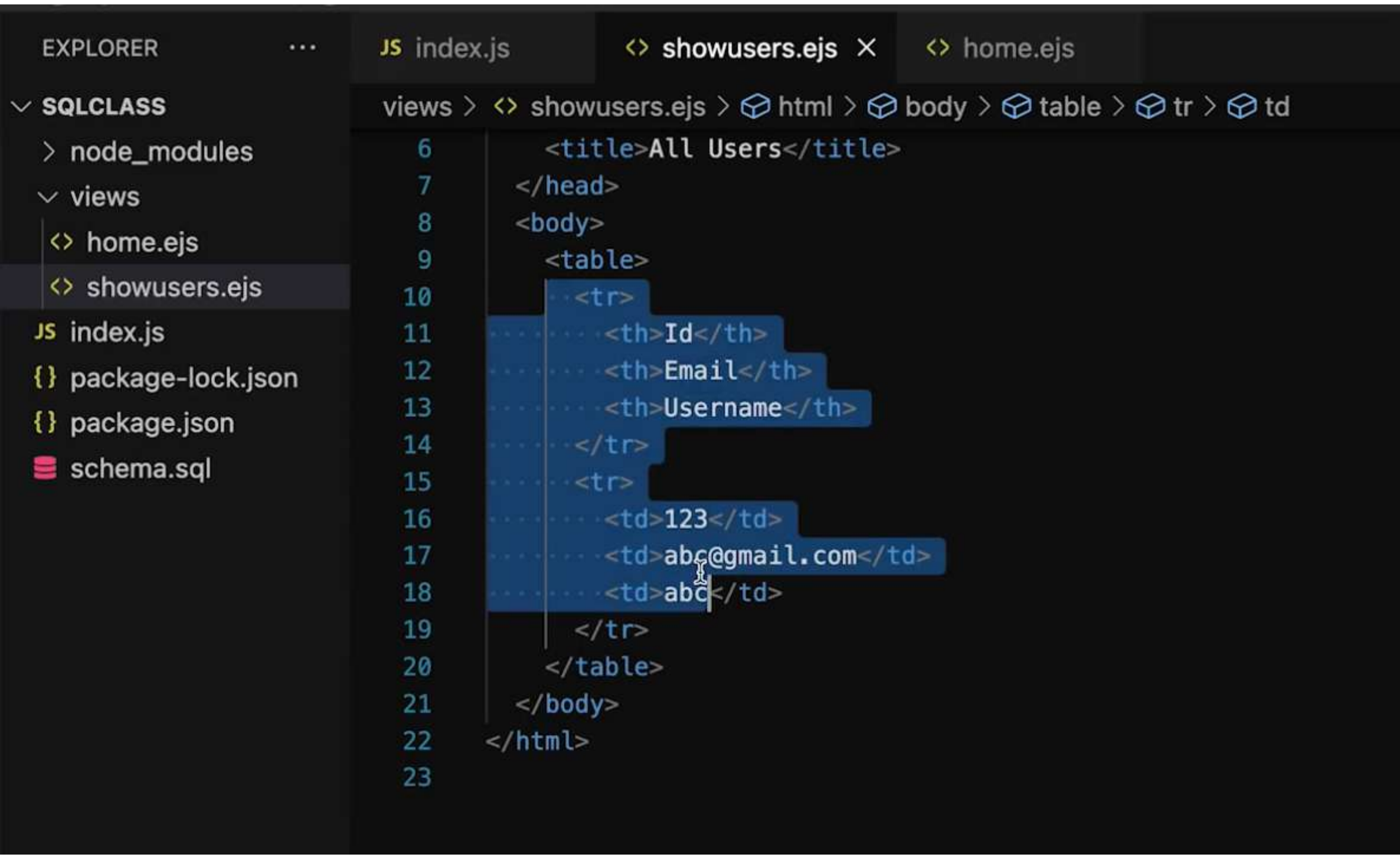
```
app.get("/user", (req, res) => {
  let q = `SELECT * FROM user`;
  try {
    connection.query(q, (err, result) => {
      if (err) throw err;
      let data = result;
      console.log(data);
      res.render("users.ejs", { data });
    });
  } catch (err) {
    res.send("some error occurred");
  }
});
```



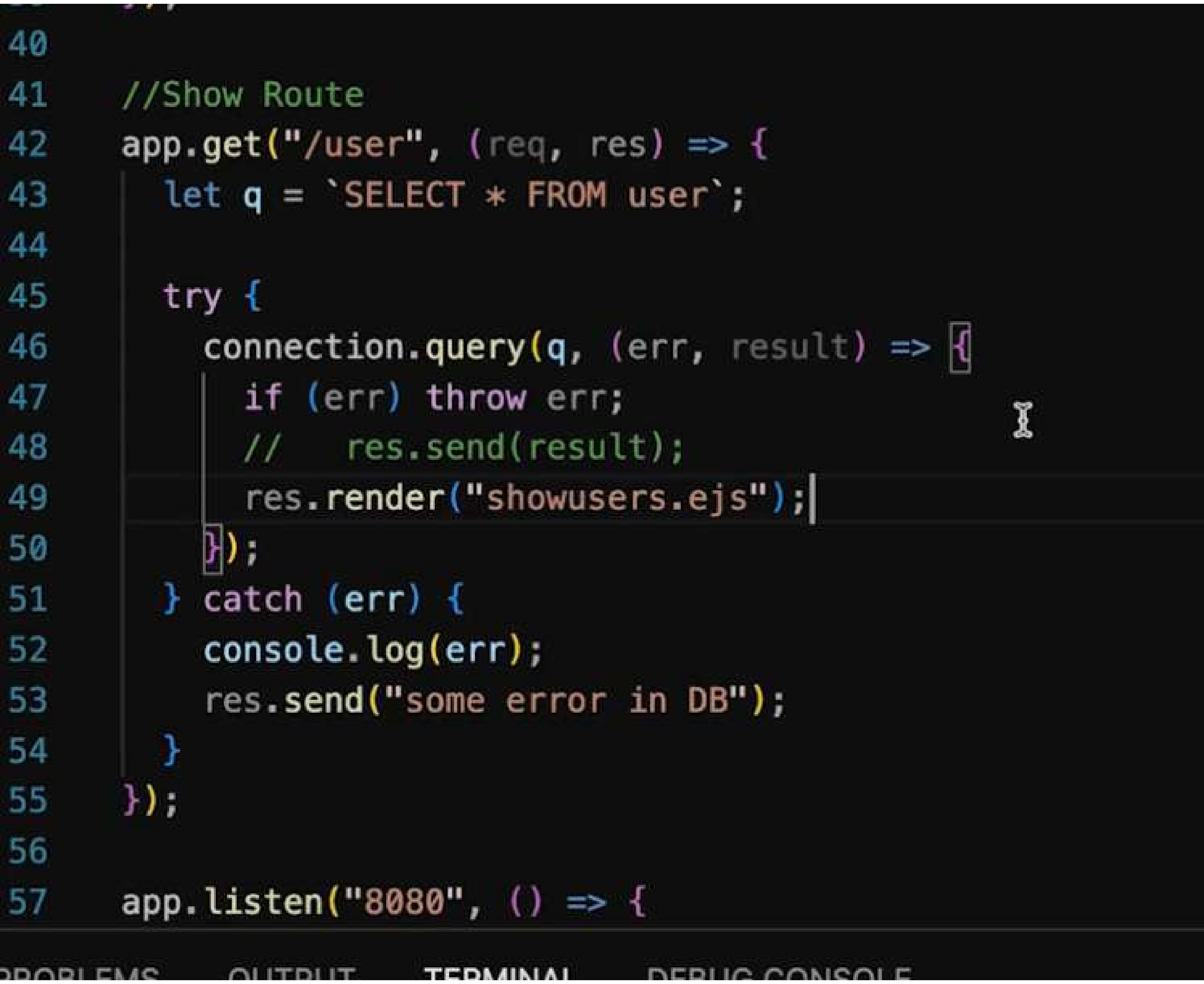
```javascript
//Show Route
app.get("/user", (req, res) => {
  let q = `SELECT * FROM user`;

  try {
    connection.query(q, (err, result) => {
      if (err) throw err;
      console.log(result);
      res.send(result);
    });
  } catch (err) {
    console.log(err);
    res.send("some error in DB");
  }
});
```



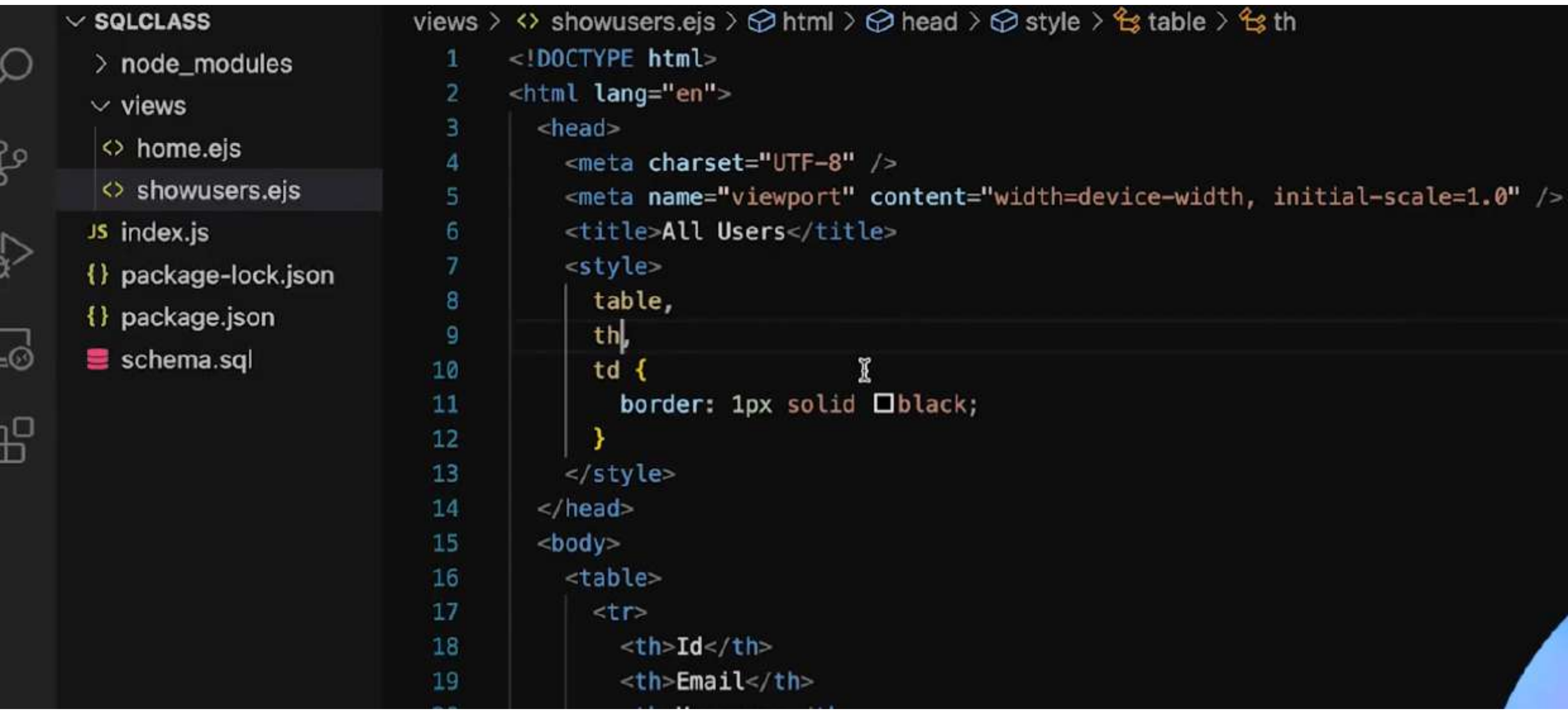
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>All Users</title>
</head>
<body>
    <table>
        <tr>
            <th>Id</th>
            <th>Email</th>
            <th>Username</th>
        </tr>

    </table>
</body>
</html>
```



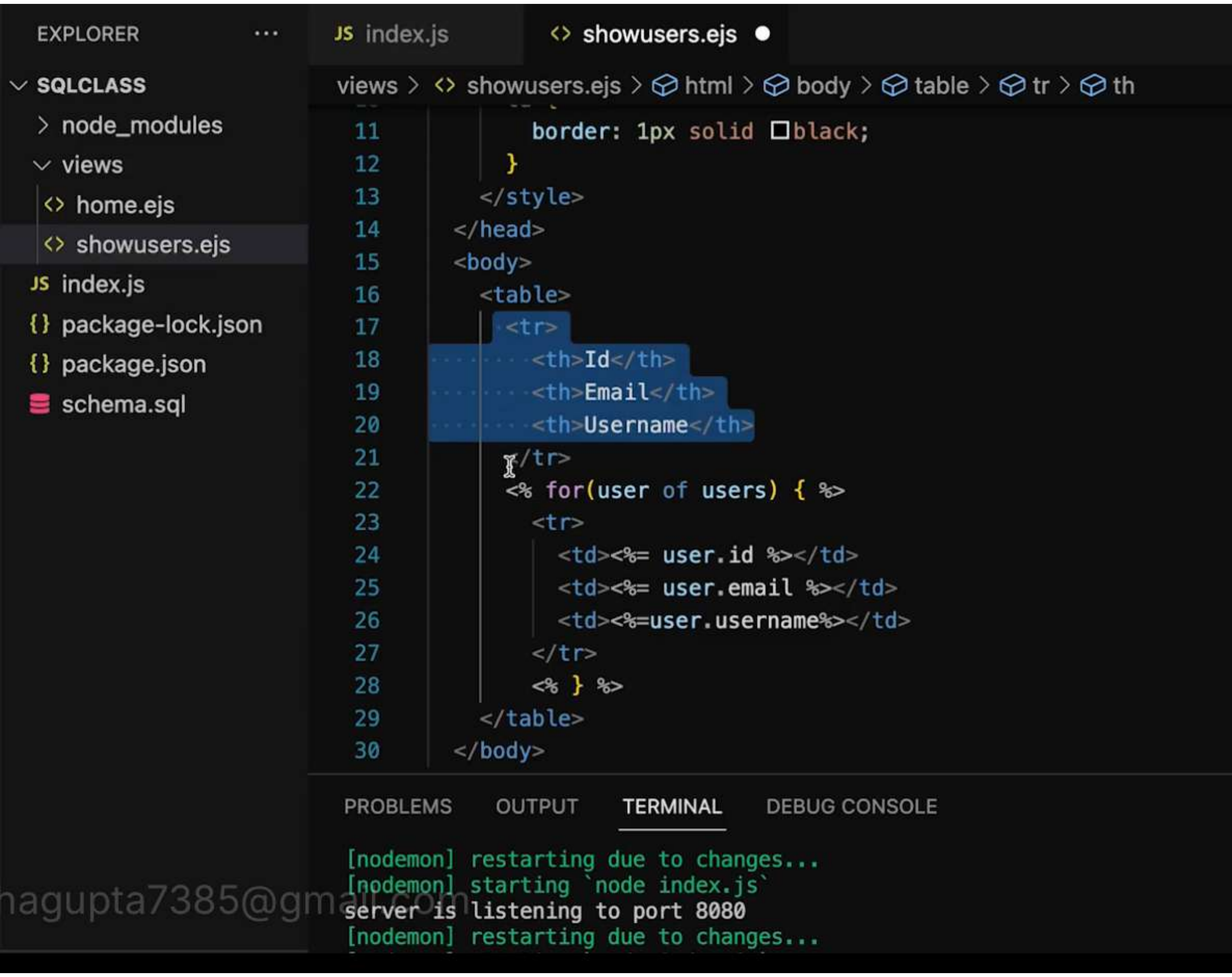
```html
    <title>All Users</title>
  </head>
  <body>
    <table>
      <tr>
        <th>Id</th>
        <th>Email</th>
        <th>Username</th>
      </tr>
      <tr>
        <td>123</td>
        <td>abc@gmail.com</td>
        <td>abc</td>
      </tr>
    </table>
  </body>
</html>
```



```
//Show Route
app.get("/user", (req, res) => {
  let q = `SELECT * FROM user`;

  try {
    connection.query(q, (err, result) => {
      if (err) throw err;
      //   res.send(result);
      res.render("showusers.ejs");
    });
  } catch (err) {
    console.log(err);
    res.send("some error in DB");
  }
});

app.listen("8080", () => {
```

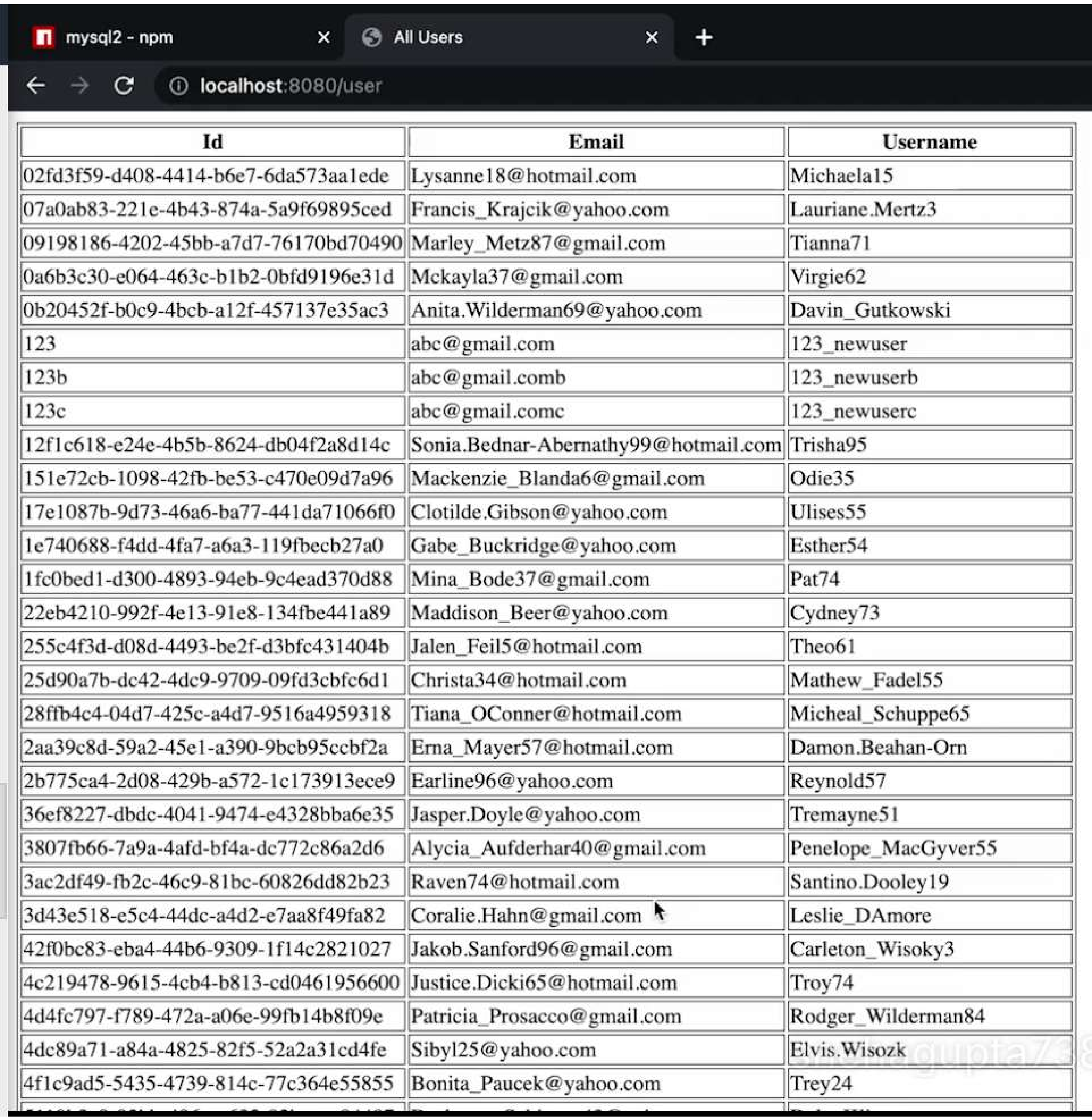


```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>All Users</title>
    <style>
      table,
      th,
      td {
        border: 1px solid black;
      }
    </style>
  </head>
  <body>
    <table>
      <tr>
        <th>Id</th>
        <th>Email</th>
```

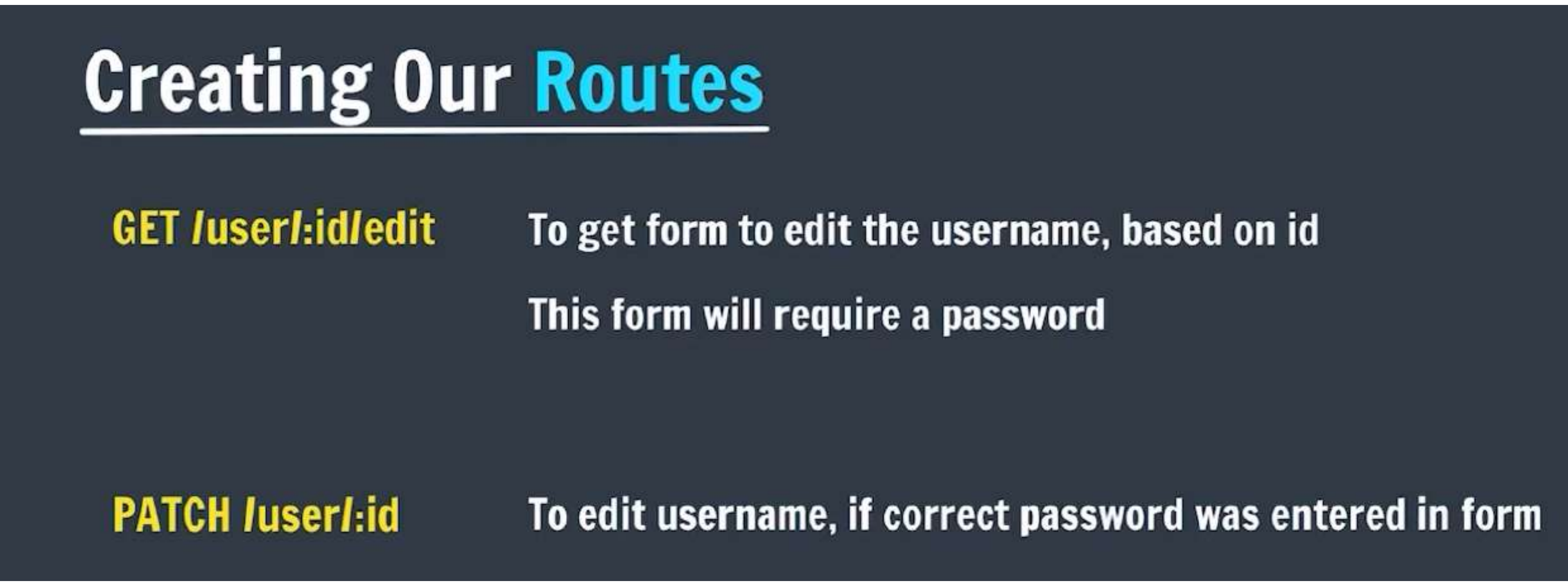


```
          border: 1px solid black;
        }
      </style>
    </head>
    <body>
      <table>
        <tr>
          <th>Id</th>
          <th>Email</th>
          <th>Username</th>
        </tr>
        <% for(user of users) { %>
          <tr>
            <td><%= user.id %></td>
            <td><%= user.email %></td>
            <td><%=user.username%></td>
          </tr>
          <% } %>
      </table>
    </body>
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
server is listening to port 8080
[nodemon] restarting due to changes...
```

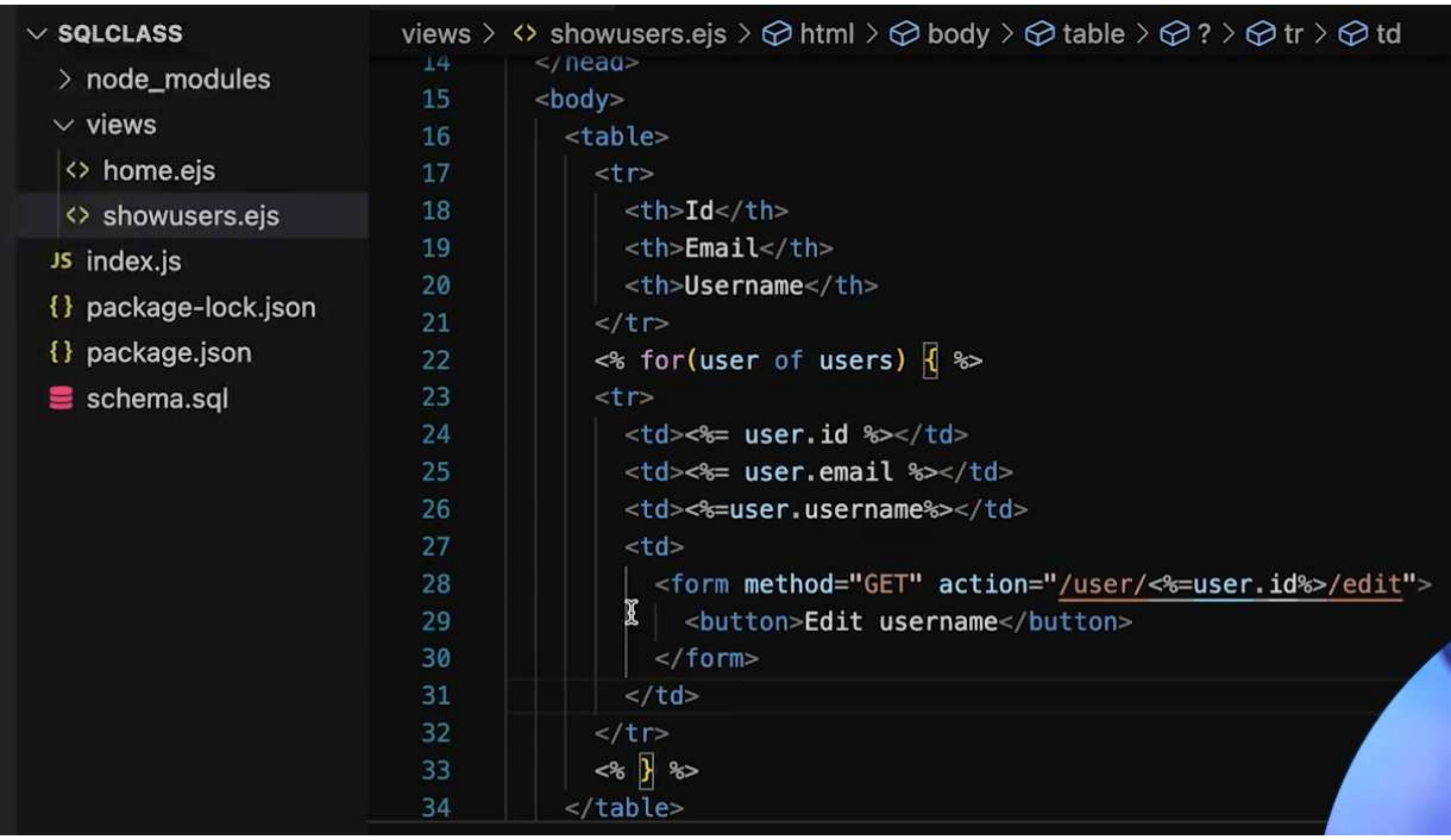


Id	Email	Username
02fd3f59-d408-4414-b6e7-6da573aa1ede	Lysanne18@hotmail.com	Michaela15
07a0ab83-221e-4b43-874a-5a9f69895ced	Francis_Krajcik@yahoo.com	Lauriane.Mertz3
09198186-4202-45bb-a7d7-76170bd70490	Marley_Metz87@gmail.com	Tianna71
0a6b3c30-e064-463c-b1b2-0bfd9196e31d	Mckayla37@gmail.com	Virgie62
0b20452f-b0c9-4bcb-a12f-457137e35ac3	Anita.Wilderman69@yahoo.com	Davin_Gutkowski
123	abc@gmail.com	123_newuser
123b	abc@gmail.comb	123_newuserb
123c	abc@gmail.comc	123_newuserc
12f1c618-e24e-4b5b-8624-db04f2a8d14c	Sonia.Bednar-Abernathy99@hotmail.com	Trisha95
151e72cb-1098-42fb-be53-c470e09d7a96	Mackenzie_Blanda6@gmail.com	Odie35
17e1087b-9d73-46a6-ba77-441da71066f0	Clotilde.Gibson@yahoo.com	Ulises55
1e740688-f4dd-4fa7-a6a3-119fbecb27a0	Gabe_Buckridge@yahoo.com	Esther54
1fc0bed1-d300-4893-94eb-9c4ead370d88	Mina_Bode37@gmail.com	Pat74
22eb4210-992f-4e13-91e8-134fbe441a89	Maddison_Beer@yahoo.com	Cydney73
255c4f3d-d08d-4493-be2f-d3bfc431404b	Jalen_Feil5@hotmail.com	Theo61
25d90a7b-dc42-4dc9-9709-09fd3cbfc6d1	Christa34@hotmail.com	Mathew_Fadel55
28ffb4c4-04d7-425c-a4d7-9516a4959318	Tiana_OConner@hotmail.com	Micheal_Schuppe65
2aa39c8d-59a2-45e1-a390-9bcb95ccbf2a	Erna_Mayer57@hotmail.com	Damon.Beahan-Orn
2b775ca4-2d08-429b-a572-1c173913ece9	Earline96@yahoo.com	Reynold57
36ef8227-dbdc-4041-9474-e4328bba6e35	Jasper.Doyle@yahoo.com	Tremayne51
3807fb66-7a9a-4afd-bf4a-dc772c86a2d6	Alycia_Aufderhar40@gmail.com	Penelope_MacGyver55
3ac2df49-fb2c-46c9-81bc-60826dd82b23	Raven74@hotmail.com	Santino.Dooley19
3d43e518-e5c4-44dc-a4d2-e7aa8f49fa82	Coralie.Hahn@gmail.com	Leslie_DAmore
42f0bc83-eba4-44b6-9309-1f14c2821027	Jakob.Sanford96@gmail.com	Carleton_Wisoky3
4c219478-9615-4cb4-b813-cd0461956600	Justice.Dicki65@hotmail.com	Troy74
4d4fc797-f789-472a-a06e-99fb14b8f09e	Patricia_Prosacco@gmail.com	Rodger_Wilderman84
4dc89a71-a84a-4825-82f5-52a2a31cd4fe	Sibyl25@yahoo.com	Elvis.Wisozk
4f1c9ad5-5435-4739-814c-77c364e55855	Bonita_Paucek@yahoo.com	Trey24

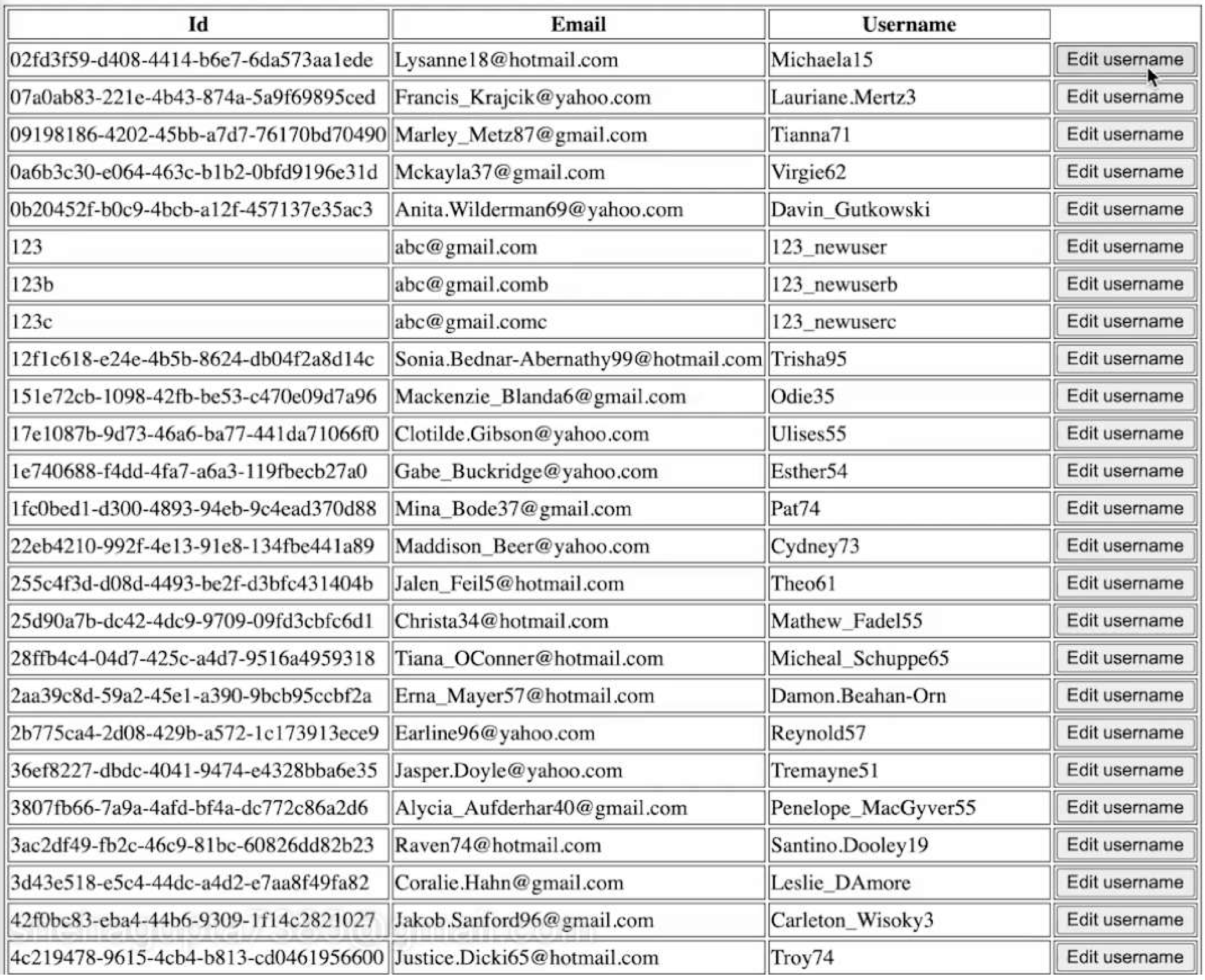


Creating Our Routes

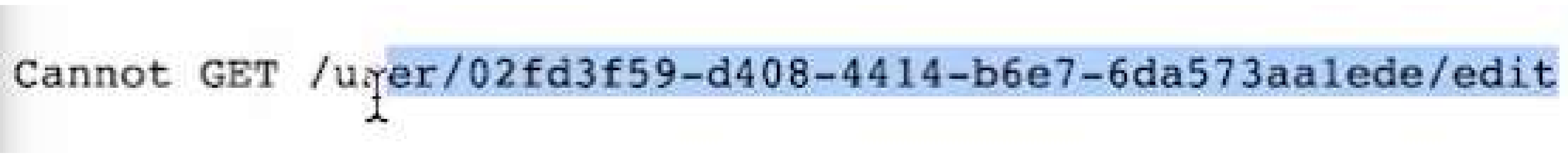
GET /user/:id/edit	To get form to edit the username, based on id This form will require a password
PATCH /user/:id	To edit username, if correct password was entered in form



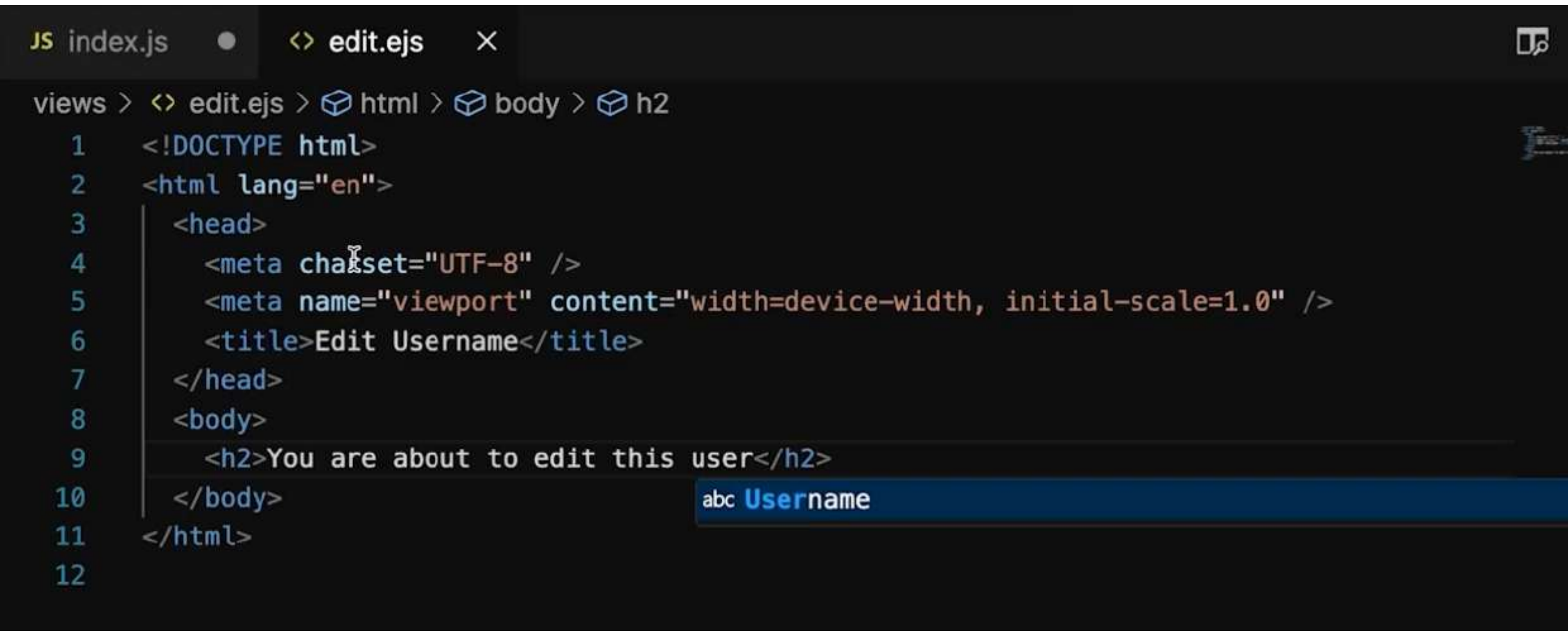
```
    </head>
    <body>
      <table>
        <tr>
          <th>Id</th>
          <th>Email</th>
          <th>Username</th>
        </tr>
        <% for(user of users) { %>
        <tr>
          <td><%= user.id %></td>
          <td><%= user.email %></td>
          <td><%=user.username%></td>
          <td>
            <form method="GET" action="/user/<%=user.id%>/edit">
              <button>Edit username</button>
            </form>
          </td>
        </tr>
        <% } %>
      </table>
```



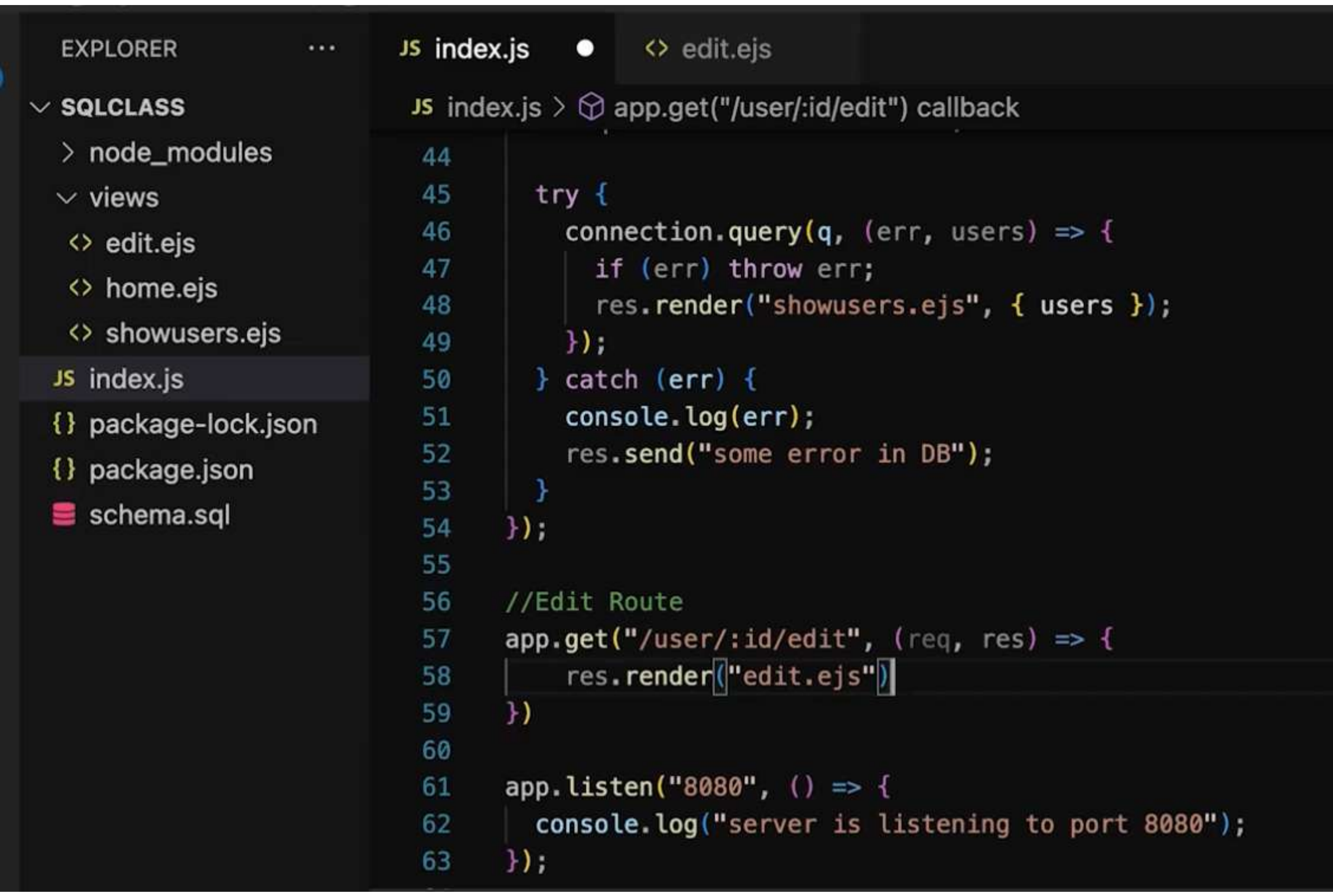
Id	Email	Username	
02fd3f59-d408-4414-b6e7-6da573aa1ede	Lysanne18@hotmail.com	Michaela15	Edit username
07a0ab83-221e-4b43-874a-5a9f69895ced	Francis_Krajcik@yahoo.com	Lauriane.Mertz3	Edit username
09198186-4202-45bb-a7d7-76170bd70490	Marley_Metz87@gmail.com	Tianna71	Edit username
0a6b3c30-e064-463c-b1b2-0bfd9196e31d	Mckayla37@gmail.com	Virgie62	Edit username
0b20452f-b0c9-4bcb-a12f-457137e35ac3	Anita.Wilderman69@yahoo.com	Davin_Gutkowski	Edit username
123	abc@gmail.com	123_newuser	Edit username
123b	abc@gmail.comb	123_newuserb	Edit username
123c	abc@gmail.comc	123_newuserc	Edit username
12f1c618-e24e-4b5b-8624-db04f2a8d14c	Sonia.Bednar-Abernathy99@hotmail.com	Trisha95	Edit username
151e72cb-1098-42fb-be53-c470e09d7a96	Mackenzie_Blanda6@gmail.com	Odie35	Edit username
17e1087b-9d73-46a6-ba77-441da71066f0	Clotilde.Gibson@yahoo.com	Ulises55	Edit username
1e740688-f4dd-4fa7-a6a3-119fbecb27a0	Gabe_Buckridge@yahoo.com	Esther54	Edit username
1fc0bed1-d300-4893-94eb-9c4ead370d88	Mina_Bode37@gmail.com	Pat74	Edit username
22eb4210-992f-4e13-91e8-134fbe441a89	Maddison_Beer@yahoo.com	Cydney73	Edit username
255c4f3d-d08d-4493-be2f-d3bfc431404b	Jalen_Feil5@hotmail.com	Theo61	Edit username
25d90a7b-dc42-4dc9-9709-09fd3cbfc6d1	Christa34@hotmail.com	Mathew_Fadel55	Edit username
28ffb4c4-04d7-425c-a4d7-9516a4959318	Tiana_OConner@hotmail.com	Micheal_Schuppe65	Edit username
2aa39c8d-59a2-45e1-a390-9bcb95ccbf2a	Erna_Mayer57@hotmail.com	Damon.Beahan-Orn	Edit username
2b775ca4-2d08-429b-a572-1c173913ece9	Earline96@yahoo.com	Reynold57	Edit username
36ef8227-dbdc-4041-9474-e4328bba6e35	Jasper.Doyle@yahoo.com	Tremayne51	Edit username
3807fb66-7a9a-4afd-bf4a-dc772c86a2d6	Alycia_Aufderhar40@gmail.com	Penelope_MacGyver55	Edit username
3ac2df49-fb2c-46c9-81bc-60826dd82b23	Raven74@hotmail.com	Santino.Dooley19	Edit username
3d43e518-e5c4-44dc-a4d2-e7aa8f49fa82	Coralie.Hahn@gmail.com	Leslie_DAmore	Edit username
42f0bc83-eba4-44b6-9309-1f14c2821027	Jakob.Sanford96@gmail.com	Carleton_Wisoky3	Edit username
4c219478-9615-4cb4-b813-cd0461956600	Justice.Dicki65@hotmail.com	Troy74	Edit username



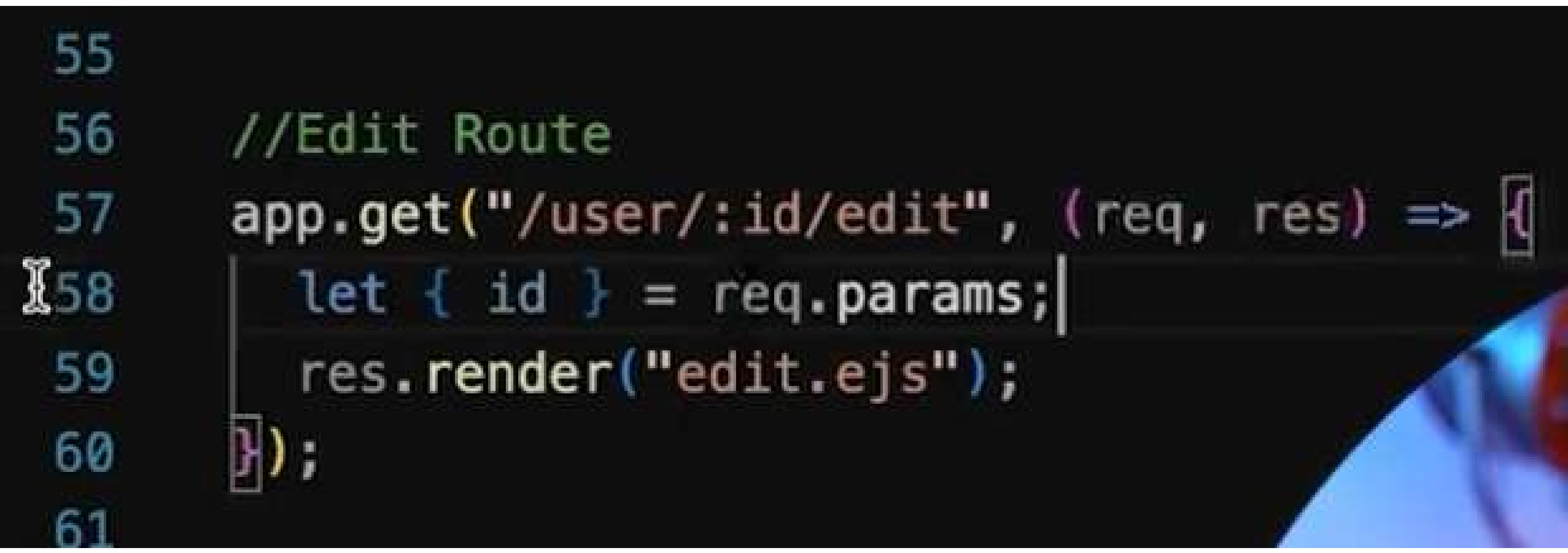
```
Cannot GET /user/02fd3f59-d408-4414-b6e7-6da573aa1ede/edit
```



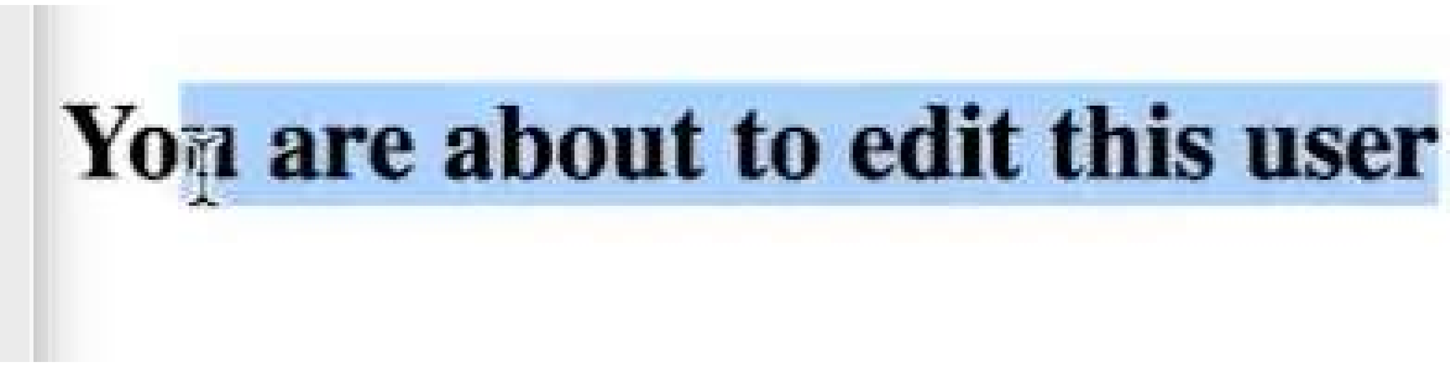
```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Edit Username</title>
  </head>
  <body>
    <h2>You are about to edit this user</h2>
  </body>
</html>

```



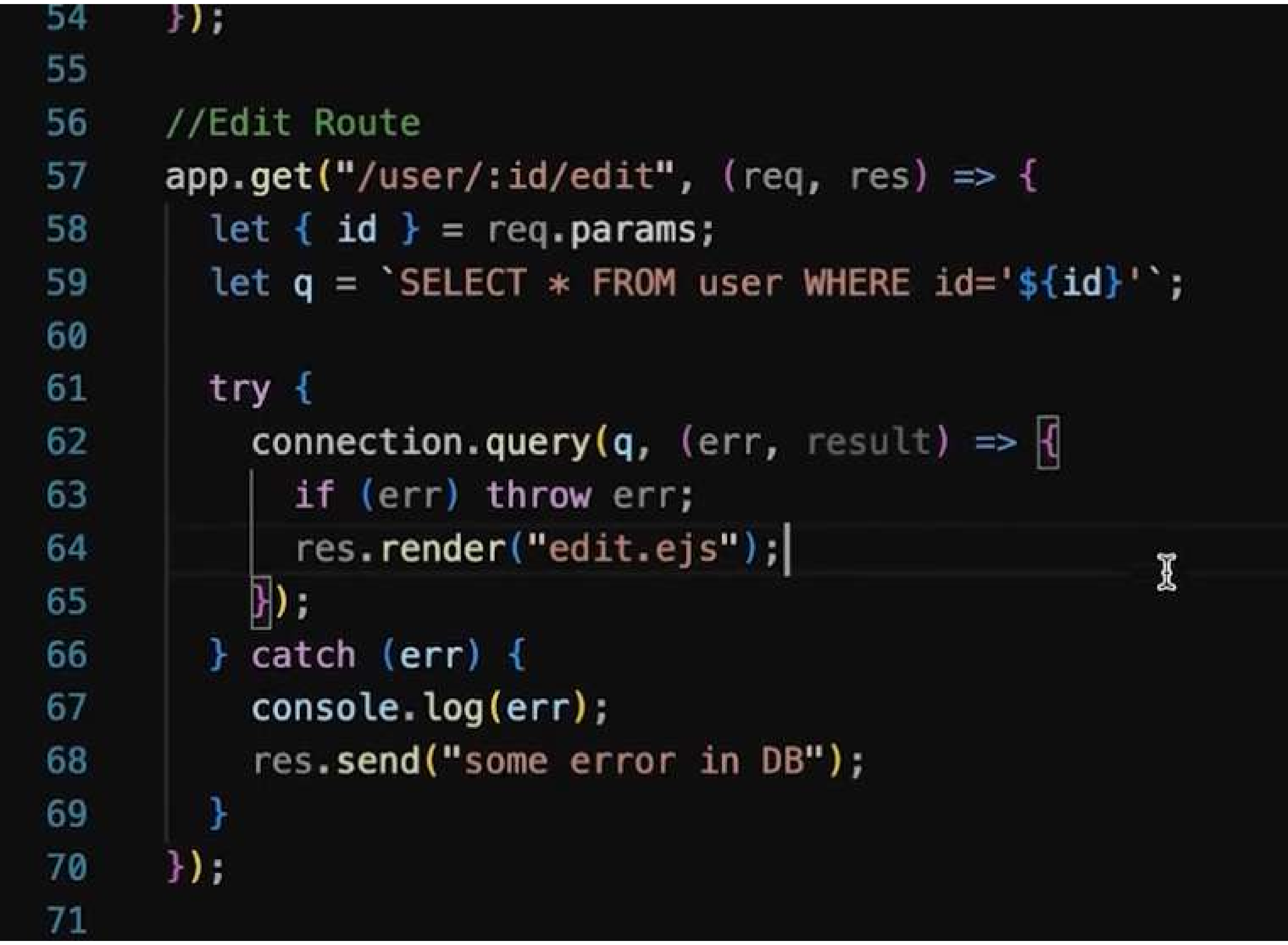
```

    try {
      connection.query(q, (err, users) => {
        if (err) throw err;
        res.render("showusers.ejs", { users });
      });
    } catch (err) {
      console.log(err);
      res.send("some error in DB");
    }
  });

  //Edit Route
  app.get("/user/:id/edit", (req, res) => {
      res.render("edit.ejs")
  })

  app.listen("8080", () => {
    console.log("server is listening to port 8080");
  });
```



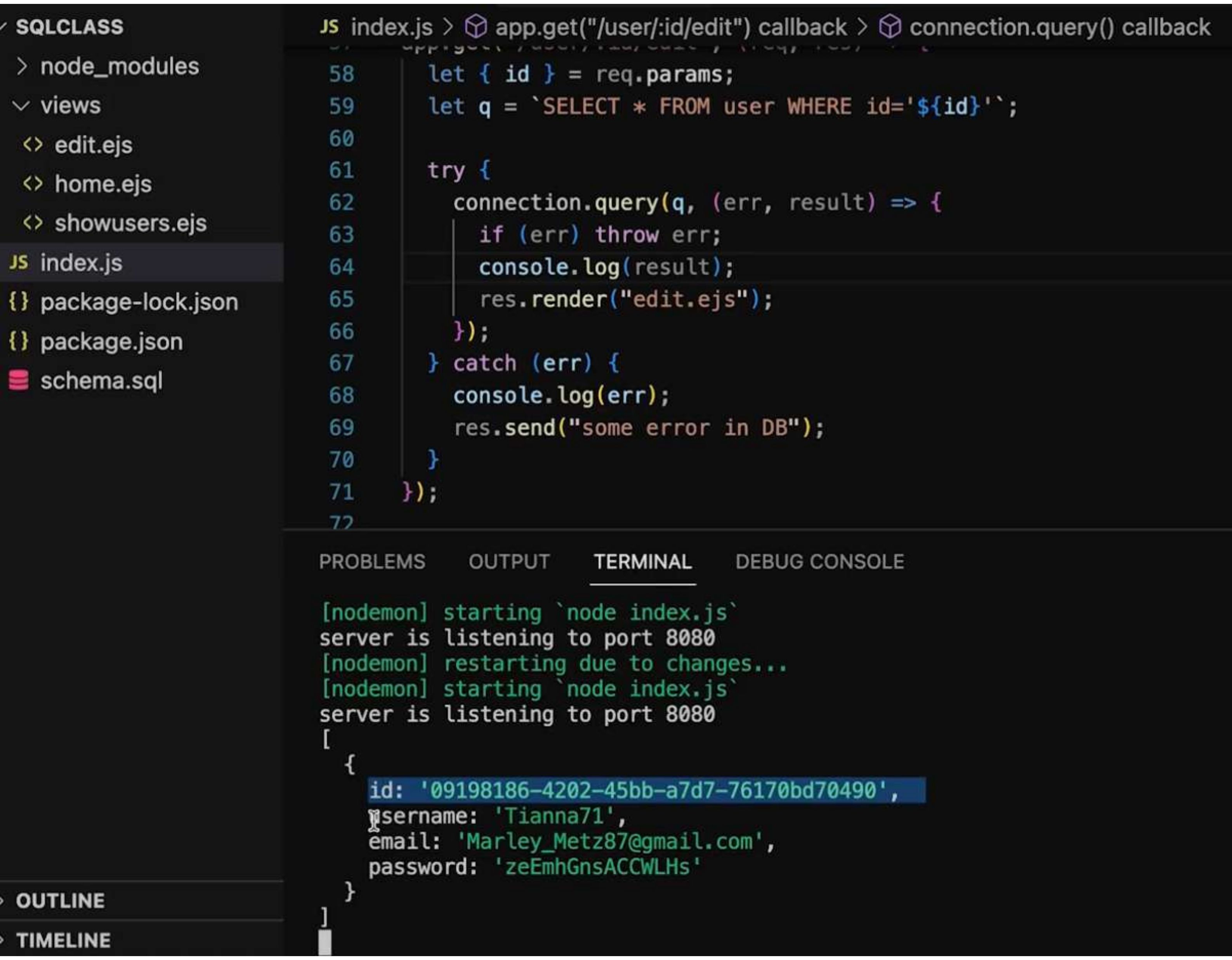
```
//Edit Route
app.get("/user/:id/edit", (req, res) => {
  let { id } = req.params;
  res.render("edit.ejs");
});
```



You are about to edit this user



```
});

//Edit Route
app.get("/user/:id/edit", (req, res) => {
  let { id } = req.params;
  let q = `SELECT * FROM user WHERE id='${id}'`;

  try {
    connection.query(q, (err, result) => {
      if (err) throw err;
      res.render("edit.ejs");
    });
  } catch (err) {
    console.log(err);
    res.send("some error in DB");
  }
});
```



SQLCLASS
- node_modules
- views
  - edit.ejs
  - home.ejs
  - showusers.ejs
- index.js
- package-lock.json
- package.json
- schema.sql

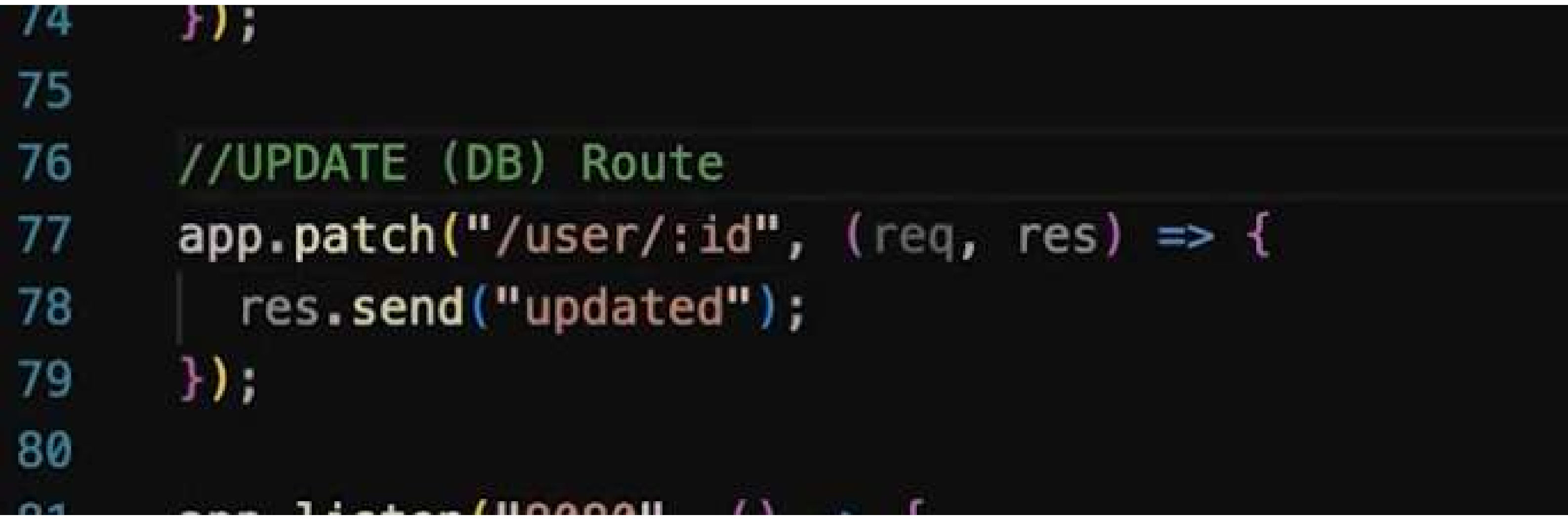
index.js > app.get("/user/:id/edit") callback > connection.query() callback

```
  let { id } = req.params;
  let q = `SELECT * FROM user WHERE id='${id}'`;

  try {
    connection.query(q, (err, result) => {
      if (err) throw err;
      console.log(result);
      res.render("edit.ejs");
    });
  } catch (err) {
    console.log(err);
    res.send("some error in DB");
  }
});
```

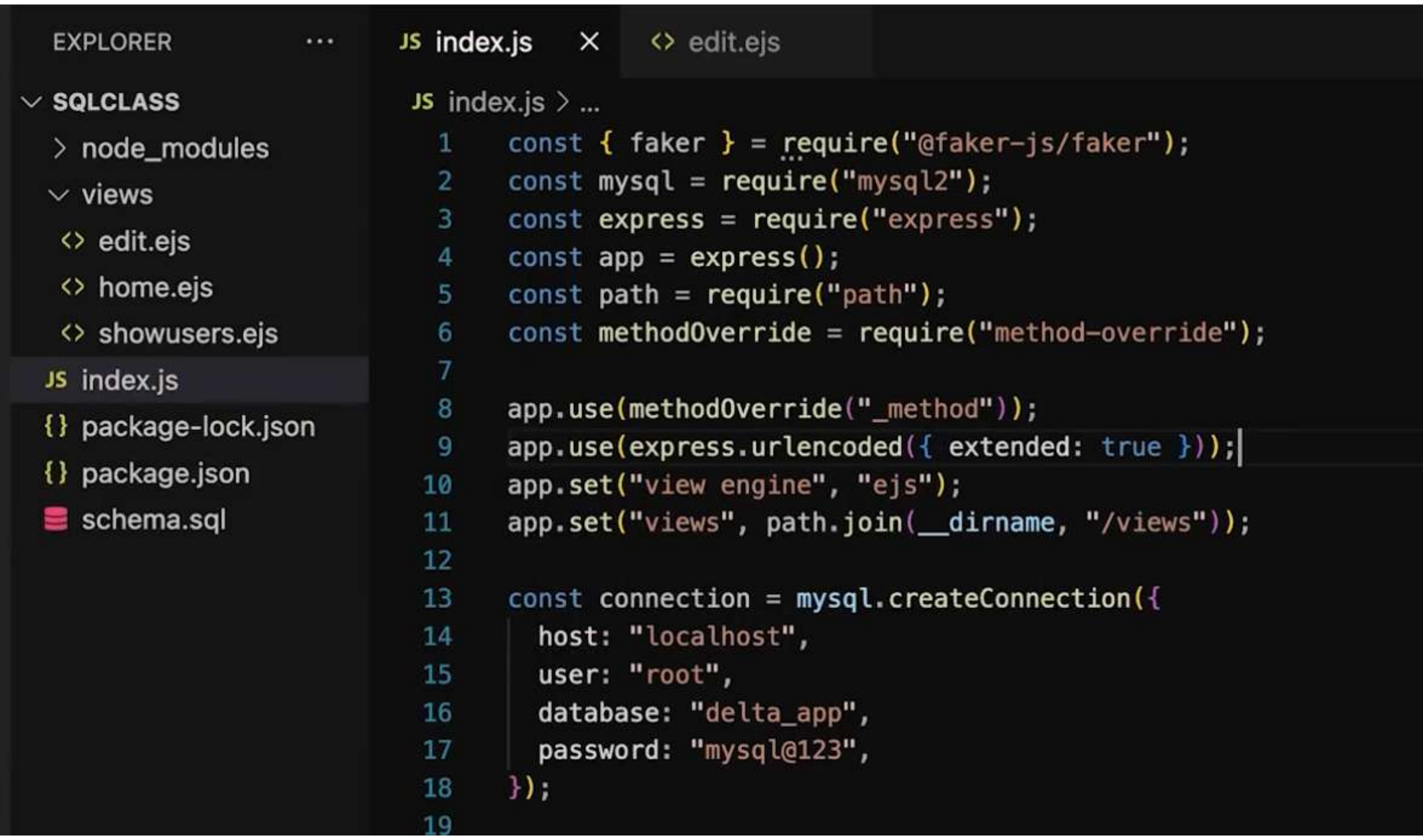
PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE

```
[nodemon] starting `node index.js`
server is listening to port 8080
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
server is listening to port 8080
[
  {
    id: '09198186-4202-45bb-a7d7-76170bd70490',
    username: 'Tianna71',
    email: 'Marley_Metz87@gmail.com',
    password: 'zeEmhGnsACCWLHs'
  }
]
```

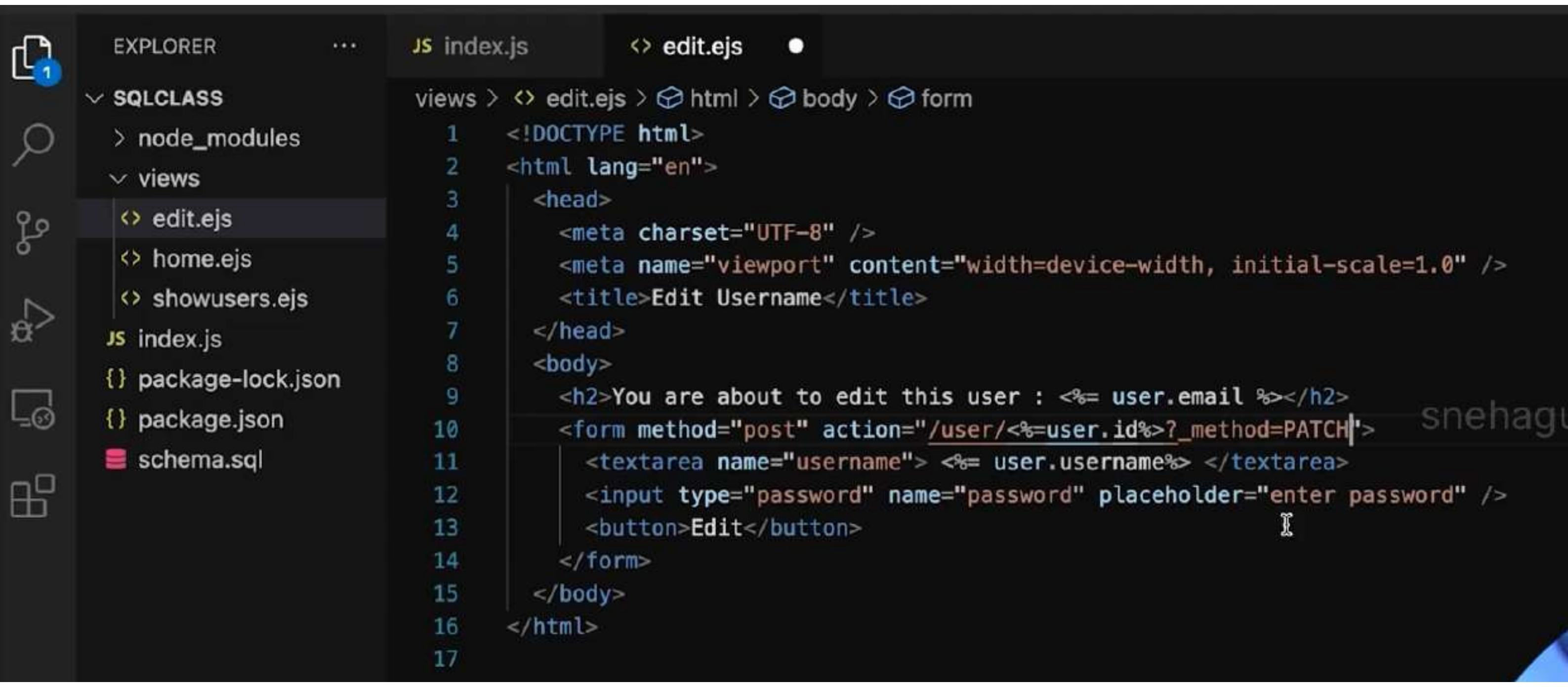
OUTLINE
TIMELINE



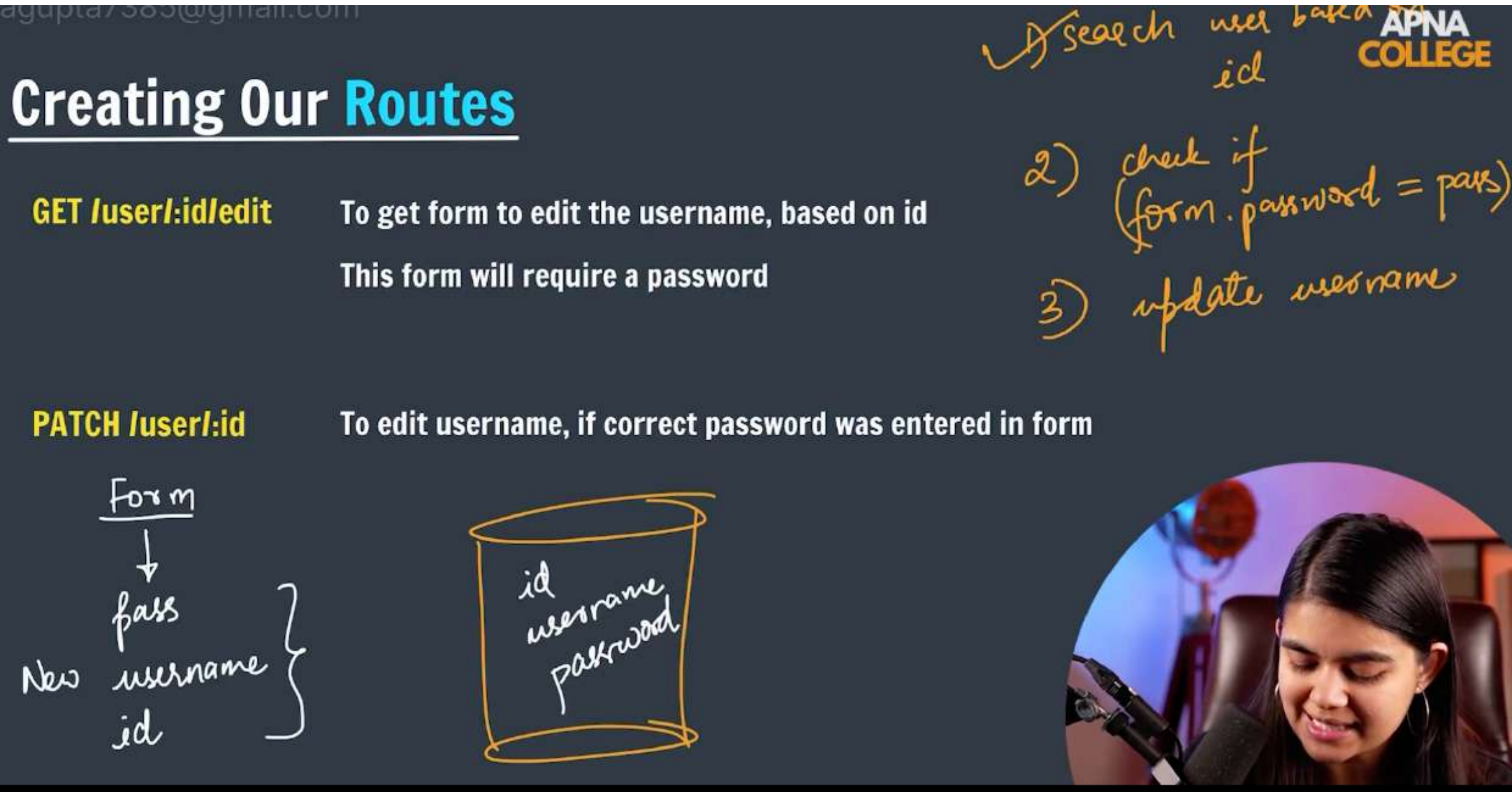
```
});

//UPDATE (DB) Route
app.patch("/user/:id", (req, res) => {
  res.send("updated");
});

app.listen("8080", () => {
```



```javascript
const { faker } = require("@faker-js/faker");
const mysql = require("mysql2");
const express = require("express");
const app = express();
const path = require("path");
const methodOverride = require("method-override");

app.use(methodOverride("_method"));
app.use(express.urlencoded({ extended: true }));
app.set("view engine", "ejs");
app.set("views", path.join(__dirname, "/views"));

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "delta_app",
  password: "mysql@123",
});

```



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Edit Username</title>
  </head>
  <body>
    <h2>You are about to edit this user : <%= user.email %></h2>
    <form method="post" action="/user/<%=user.id%>?_method=PATCH">
      <textarea name="username"> <%= user.username%> </textarea>
      <input type="password" name="password" placeholder="enter password" />
      <button>Edit</button>
    </form>
  </body>
</html>
```



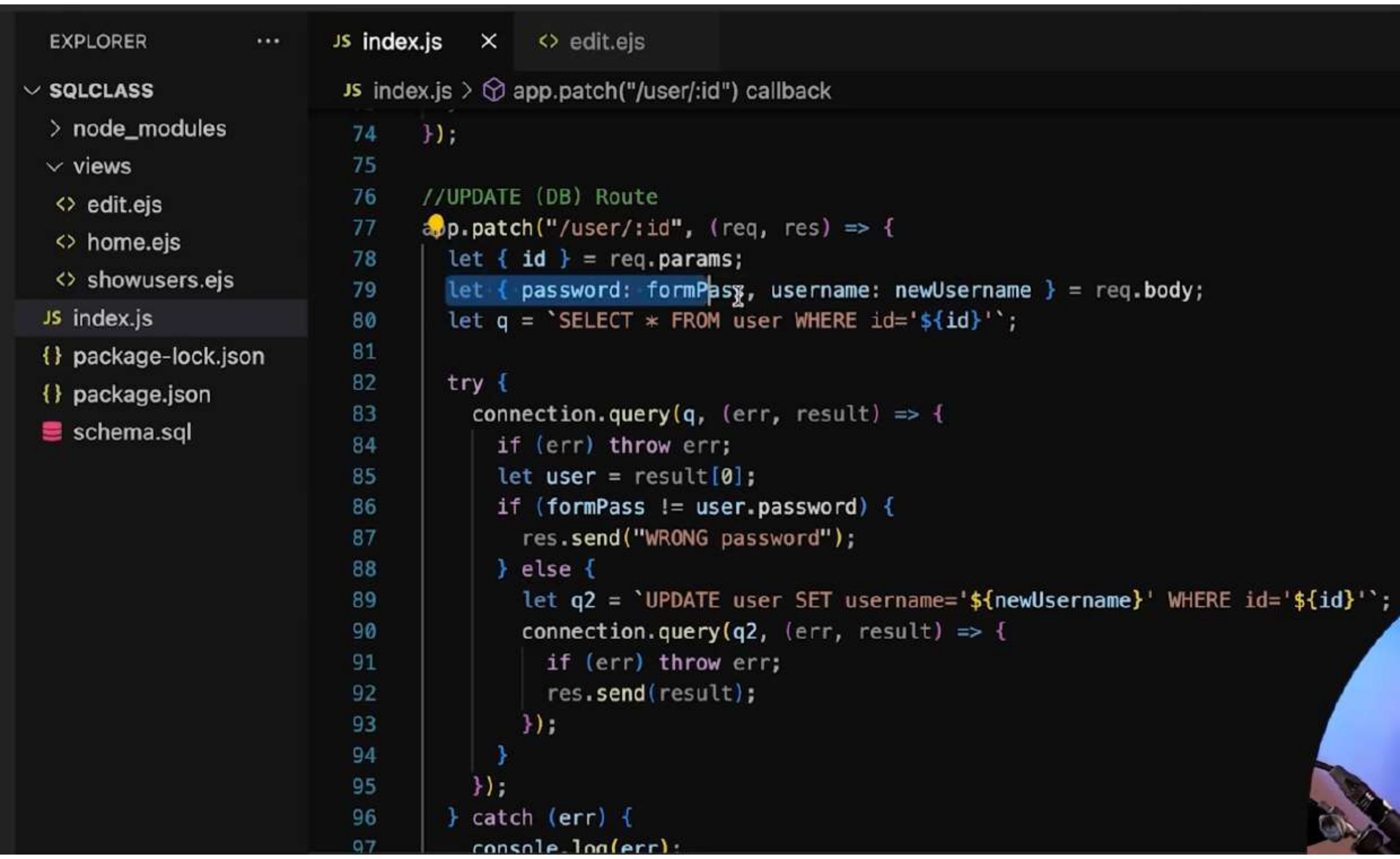
Creating Our Routes

GET /user/:id/edit To get form to edit the username, based on id

This form will require a password

PATCH /user/:id To edit username, if correct password was entered in form

1) search user based on id
2) check if (form.password = pass)
3) update username



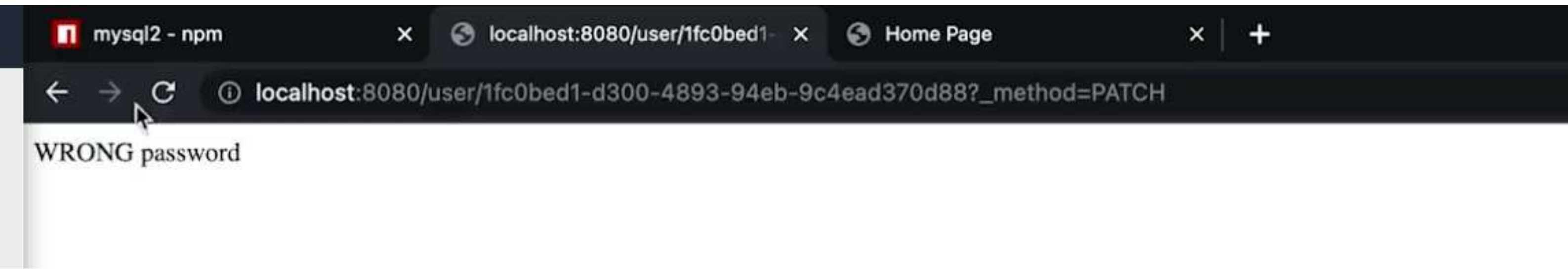
EXPLORER ···

SQLCLASS
- node_modules
- views
  - edit.ejs
  - home.ejs
  - showusers.ejs
- index.js
- package-lock.json
- package.json
- schema.sql

index.js × | edit.ejs

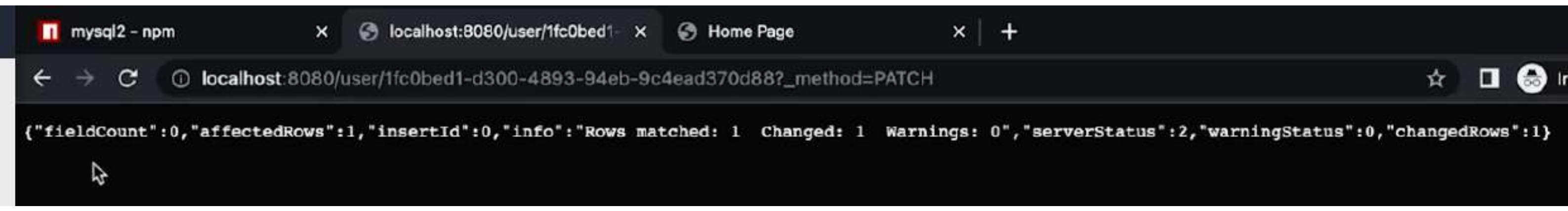
index.js > app.patch("/user/:id") callback

```
});

//UPDATE (DB) Route
app.patch("/user/:id", (req, res) => {
  let { id } = req.params;
  let { password: formPass, username: newUsername } = req.body;
  let q = `SELECT * FROM user WHERE id='${id}'`;

  try {
    connection.query(q, (err, result) => {
      if (err) throw err;
      let user = result[0];
      if (formPass != user.password) {
        res.send("WRONG password");
      } else {
        let q2 = `UPDATE user SET username='${newUsername}' WHERE id='${id}'`;
        connection.query(q2, (err, result) => {
          if (err) throw err;
          res.send(result);
        });
      }
    });
  } catch (err) {
    console.log(err);
```



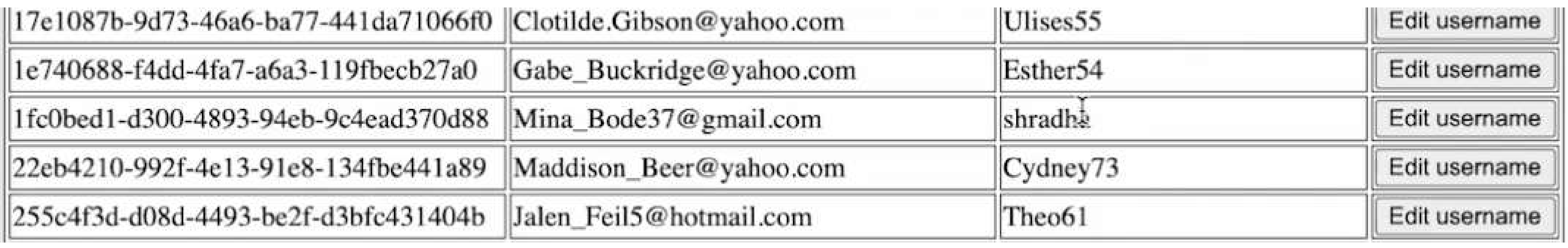
```
localhost:8080/user/1fc0bed1-d300-4893-94eb-9c4ead370d88?_method=PATCH
```

WRONG password

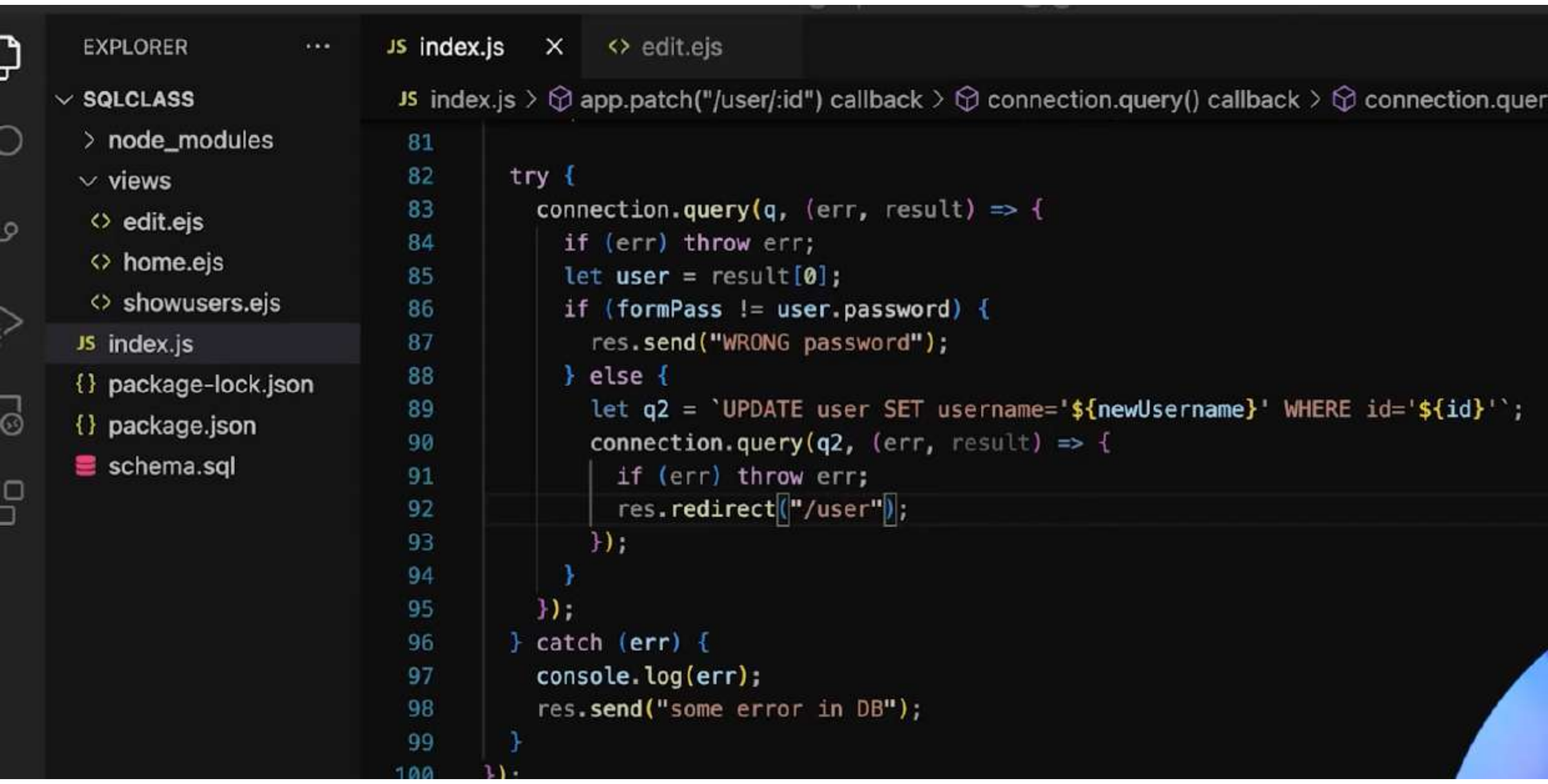


```
localhost:8080/user/1fc0bed1-d300-4893-94eb-9c4ead370d88?_method=PATCH
```

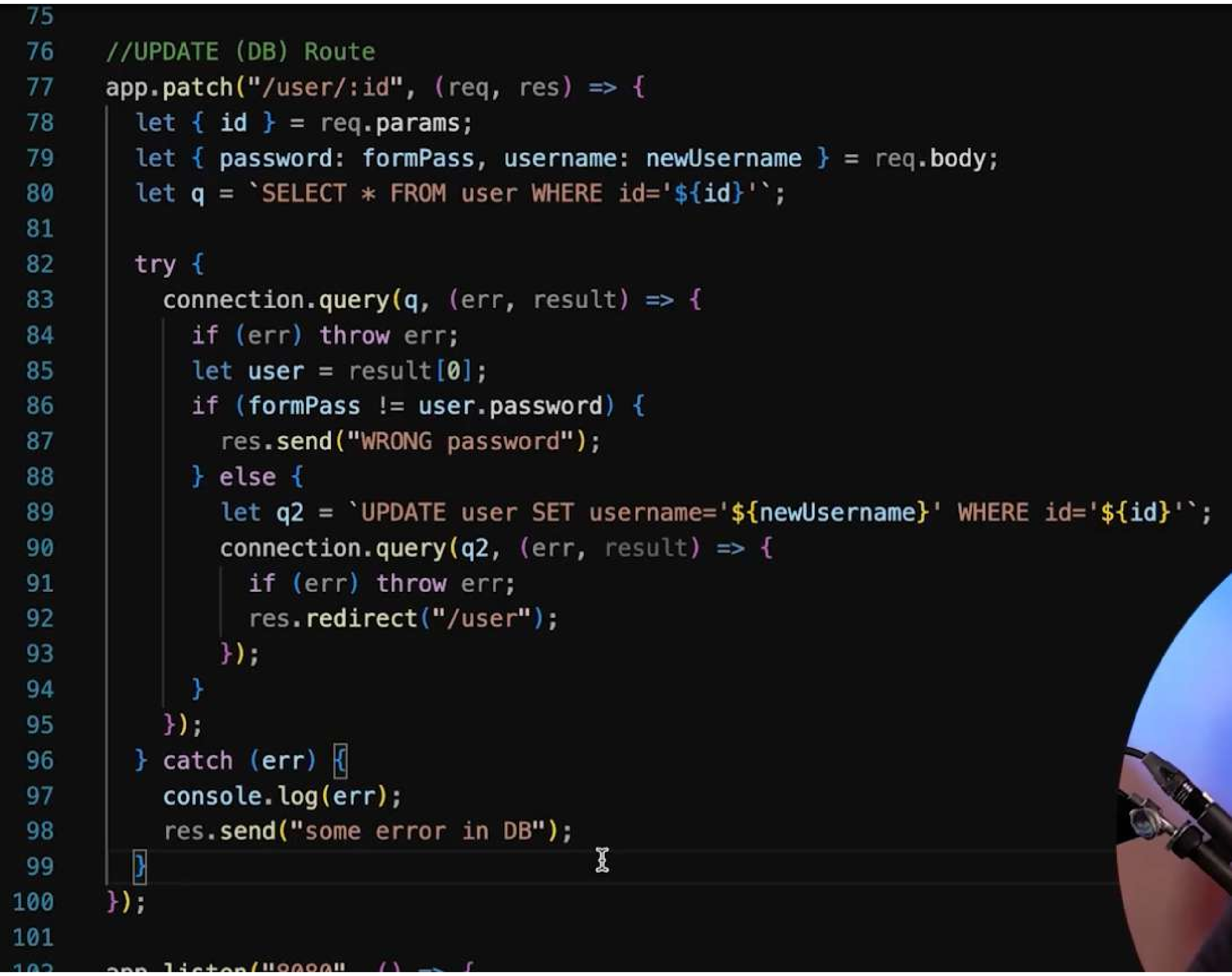
```
{"fieldCount":0,"affectedRows":1,"insertId":0,"info":"Rows matched: 1  Changed: 1  Warnings: 0","serverStatus":2,"warningStatus":0,"changedRows":1}
```



17e1087b-9d73-46a6-ba77-441da71066f0	Clotilde.Gibson@yahoo.com	Ulises55	Edit username
1e740688-f4dd-4fa7-a6a3-119fbecb27a0	Gabe_Buckridge@yahoo.com	Esther54	Edit username
1fc0bed1-d300-4893-94eb-9c4ead370d88	Mina_Bode37@gmail.com	shradhe	Edit username
22eb4210-992f-4e13-91e8-134fbe441a89	Maddison_Beer@yahoo.com	Cydney73	Edit username
255c4f3d-d08d-4493-be2f-d3bfc431404b	Jalen_Feil5@hotmail.com	Theo61	Edit username



```
    try {
      connection.query(q, (err, result) => {
        if (err) throw err;
        let user = result[0];
        if (formPass != user.password) {
          res.send("WRONG password");
        } else {
          let q2 = `UPDATE user SET username='${newUsername}' WHERE id='${id}'`;
          connection.query(q2, (err, result) => {
            if (err) throw err;
            res.redirect("/user");
          });
        }
      });
    } catch (err) {
      console.log(err);
      res.send("some error in DB");
    }
  });
```



```
//UPDATE (DB) Route
app.patch("/user/:id", (req, res) => {
  let { id } = req.params;
  let { password: formPass, username: newUsername } = req.body;
  let q = `SELECT * FROM user WHERE id='${id}'`;

  try {
    connection.query(q, (err, result) => {
      if (err) throw err;
      let user = result[0];
      if (formPass != user.password) {
        res.send("WRONG password");
      } else {
        let q2 = `UPDATE user SET username='${newUsername}' WHERE id='${id}'`;
        connection.query(q2, (err, result) => {
          if (err) throw err;
          res.redirect("/user");
        });
      }
    });
  } catch (err) {
    console.log(err);
    res.send("some error in DB");
  }
});

app.listen("8080", () => {
```

Homework Tasks

1) Create Form to Add a new user to the Database

Create Form to Delete a user from Database if they enter correct email id & password.