

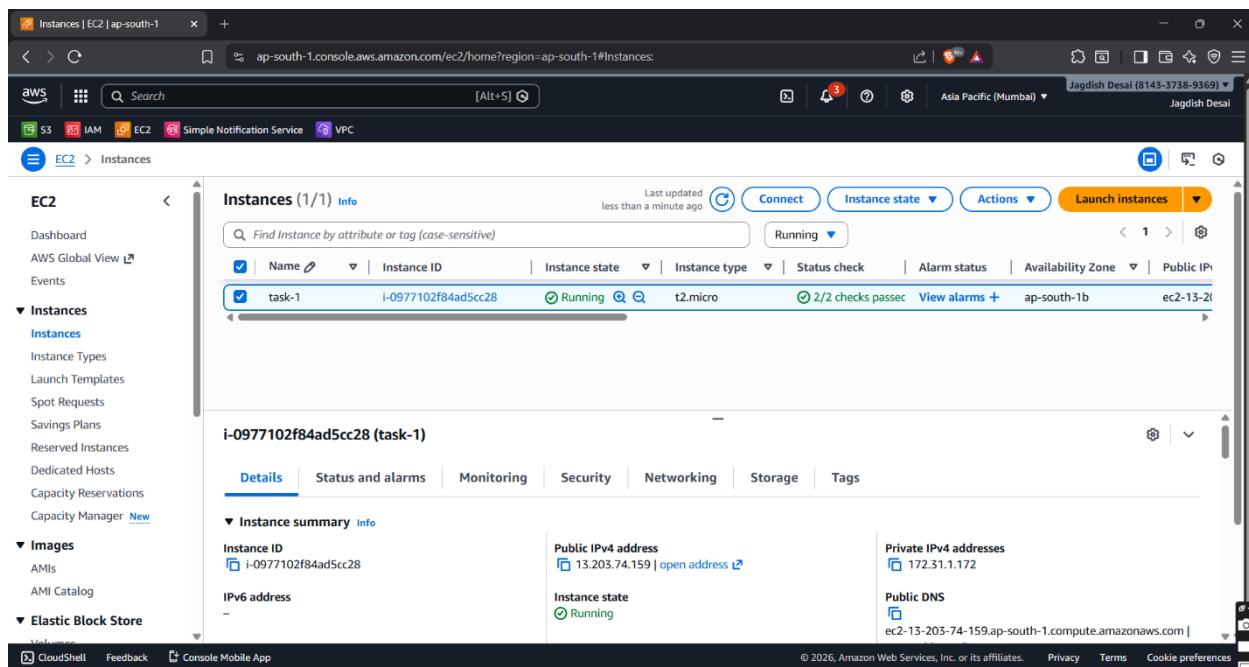
# Task 1-

## 1. Application

- Simple frontend + backend app
- Use one database (MySQL / MongoDB / Influx DB)
- App must connect to the database successfully

## Steps:

### Step 1: Created an instance task-1



### Step 2: Installing Node.js on task-1 instance

```
sudo yum install nodejs -y
```

A screenshot of the AWS EC2 Instance Connect terminal window. It shows the command 'sudo yum install nodejs -y' being run on an Amazon Linux 2023 instance. The output indicates that the package is being installed from the Amazon Linux Kernel Livepatch repository. A table at the bottom provides details for the installed packages, including their names, architectures, versions, repositories, and sizes. The transaction summary shows 6 packages installed with a total download size of 45 M and an installed size of 224 M.

## Step 3: Going to RDS and creating a mariadb database

Create database [Info](#)

Choose a database creation method

Full configuration  
You set all of the configuration options, including ones for availability, security, backups, and maintenance.

Easy create  
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type [Info](#)

Aurora (MySQL Compatible)

Aurora (PostgreSQL Compatible)

MySQL

PostgreSQL

MariaDB

Oracle

Microsoft SQL Server

IBM Db2

CloudShell Feedback [Console Mobile App](#) © 2026, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Allow port 3306 mysql port in security groups inbound rules.

After creation of database install mariadb tools on ec2 instance:

It installs MariaDB client tools on the EC2 instance

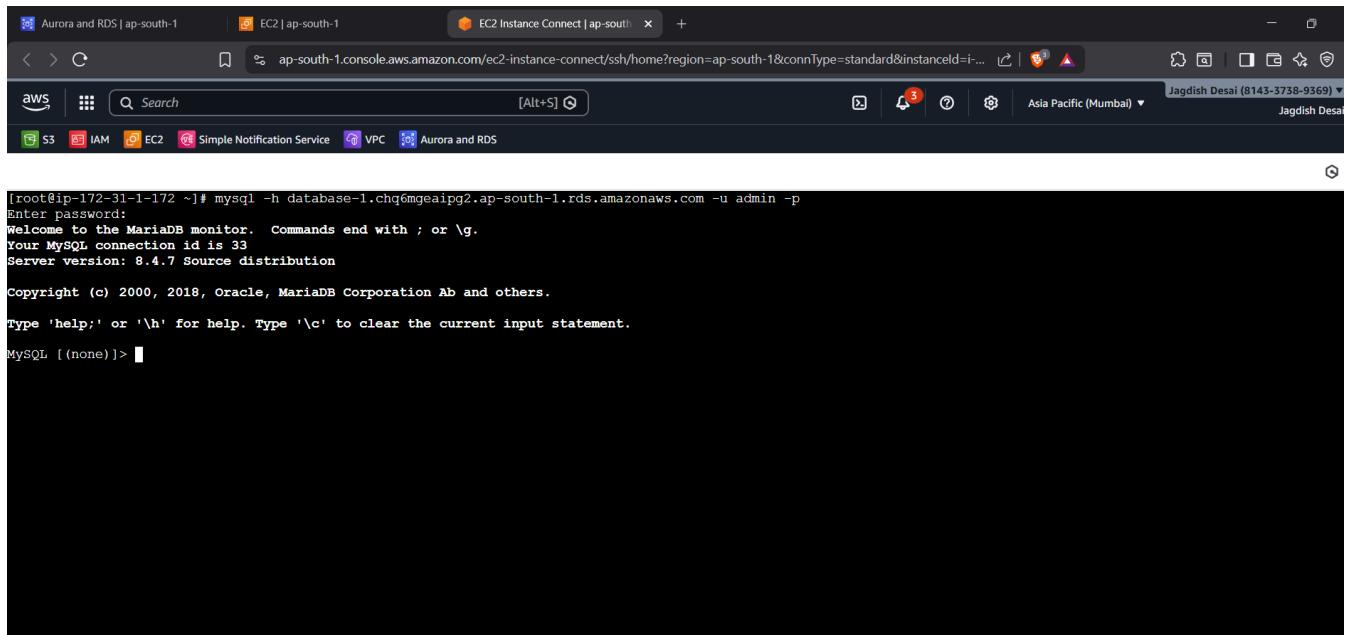
command: `yum install mariadb105* -y`

```
[ec2-user@ip-172-31-1-172 ~]$ yum install mariadb105* -y
Error: This command has to be run with superuser privileges (under the root user on most systems).
[ec2-user@ip-172-31-1-172 ~]$ sudo yum install mariadb105* -y
Last metadata expiration check: 0:29:00 ago on Sat Feb 7 07:28:10 2026.
Dependencies resolved.
=====
Package           Arch      Version       Repository   Size
=====
Installing:
mariadb105        x86_64    3:10.5.29-1.amzn2023.0.1  amazonlinux  1.5 M
mariadb105-backup  x86_64    3:10.5.29-1.amzn2023.0.1  amazonlinux  6.0 M
mariadb105-connect-engine x86_64    3:10.5.29-1.amzn2023.0.1  amazonlinux 517 k
mariadb105-cracklib-password-check x86_64    3:10.5.29-1.amzn2023.0.1  amazonlinux 13 k
mariadb105-devel   x86_64    3:10.5.29-1.amzn2023.0.1  amazonlinux 1.0 M
mariadb105-errmsg   x86_64    3:10.5.29-1.amzn2023.0.1  amazonlinux 212 k
mariadb105-gssapi-server x86_64    3:10.5.29-1.amzn2023.0.1  amazonlinux 15 k
mariadb105-ocgraph-engine x86_64    3:10.5.29-1.amzn2023.0.1  amazonlinux 75 k
mariadb105-pam     x86_64    3:10.5.29-1.amzn2023.0.1  amazonlinux 21 k
mariadb105-rocksdb-engine x86_64    3:10.5.29-1.amzn2023.0.1  amazonlinux 2.9 M
mariadb105-server-utils x86_64    3:10.5.29-1.amzn2023.0.1  amazonlinux 207 k
mariadb105-sphinx-engine x86_64    3:10.5.29-1.amzn2023.0.1  amazonlinux 58 k
mariadb105-test    x86_64    3:10.5.29-1.amzn2023.0.1  amazonlinux 14 M
=====
Installing dependencies:
Judy              x86_64    1.0.5-25.amzn2023.0.3  amazonlinux 153 k
libdatrie         x86_64    0.2.13-1.amzn2023.0.2  amazonlinux 33 k
libpq             x86_64    17.7-1.amzn2023.0.1  amazonlinux 263 k
libspfincxclient  x86_64    2.2.11-24.amzn2023.0.4  amazonlinux 133 k
libthai            x86_64    0.1.28-6.amzn2023.0.2  amazonlinux 209 k
=====
i-0977102f84ad5cc28 (task-1)
PublicIPs: 13.203.74.159 PrivateIPs: 172.31.1.172
https://ap-south-1.console.aws.amazon.com/rds/home?region=ap-south-1
© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences
```

**Then login to our database:** We use this command to connect to a MySQL database that is running on AWS RDS from your EC2/server.

## Command

```
mysql -h database-1.chq6mgeaipg2.ap-south-1.rds.amazonaws.com -u admin -p
```



The screenshot shows a terminal window within the AWS CloudShell interface. The terminal title is "Aurora and RDS | ap-south-1". The user has run the command `mysql -h database-1.chq6mgeaipg2.ap-south-1.rds.amazonaws.com -u admin -p`. The MySQL monitor prompt is visible, showing connection details and the MySQL version. The terminal also displays standard MySQL copyright and help information.

```
[root@ip-172-31-1-172 ~]# mysql -h database-1.chq6mgeaipg2.ap-south-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 33
Server version: 8.4.7 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

i-0977102f84ad5cc28 (task-1)

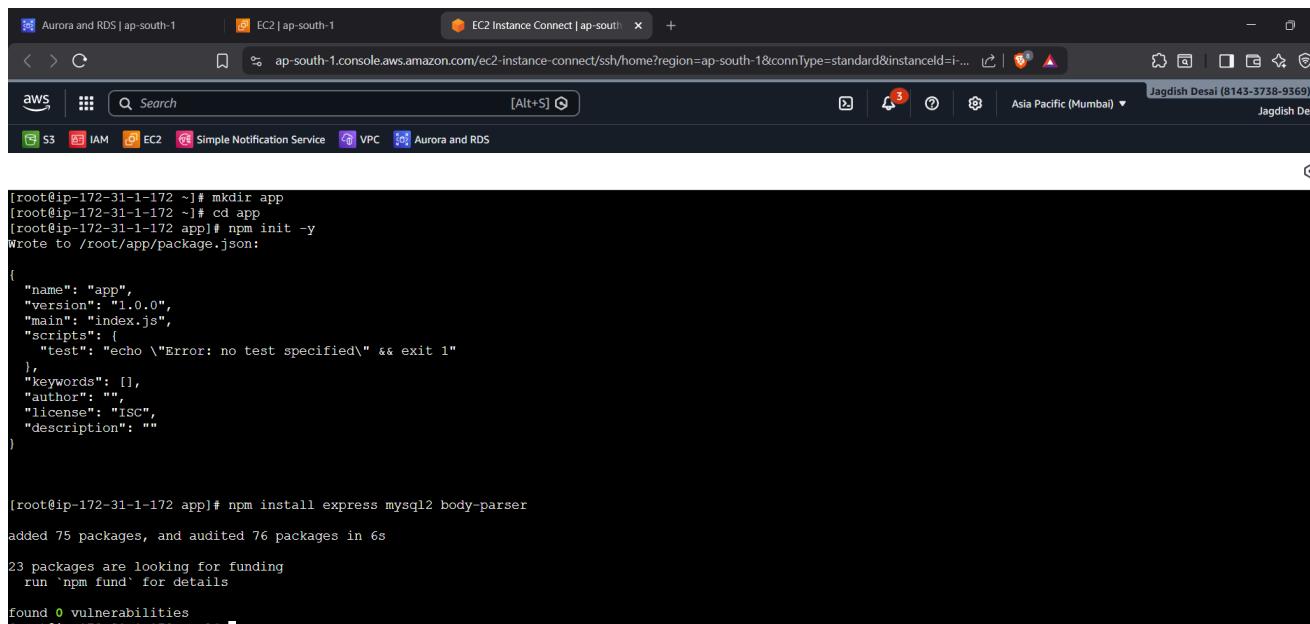
PublicIPs: 13.203.74.159 PrivateIPs: 172.31.1.172

CloudShell Feedback Console Mobile App

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

**Exit;**

**Step 4 : Then starting creating our project by creating app directory.**



The screenshot shows a terminal window within the AWS CloudShell interface. The user has run the command `mkdir app`, then changed into the `app` directory with `cd app`, and initialized a new npm project with `npm init -y`. The terminal output shows the generated `package.json` file with basic project metadata. Finally, the user runs `npm install express mysql2 body-parser`, which installs the specified dependencies.

```
[root@ip-172-31-1-172 ~]# mkdir app
[root@ip-172-31-1-172 ~]# cd app
[root@ip-172-31-1-172 app]# npm init -y
Wrote to /root/app/package.json:

{
  "name": "app",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}

[root@ip-172-31-1-172 app]# npm install express mysql2 body-parser
added 75 packages, and audited 76 packages in 6s
23 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
[root@ip-172-31-1-172 app]#
```

**Commands used:**

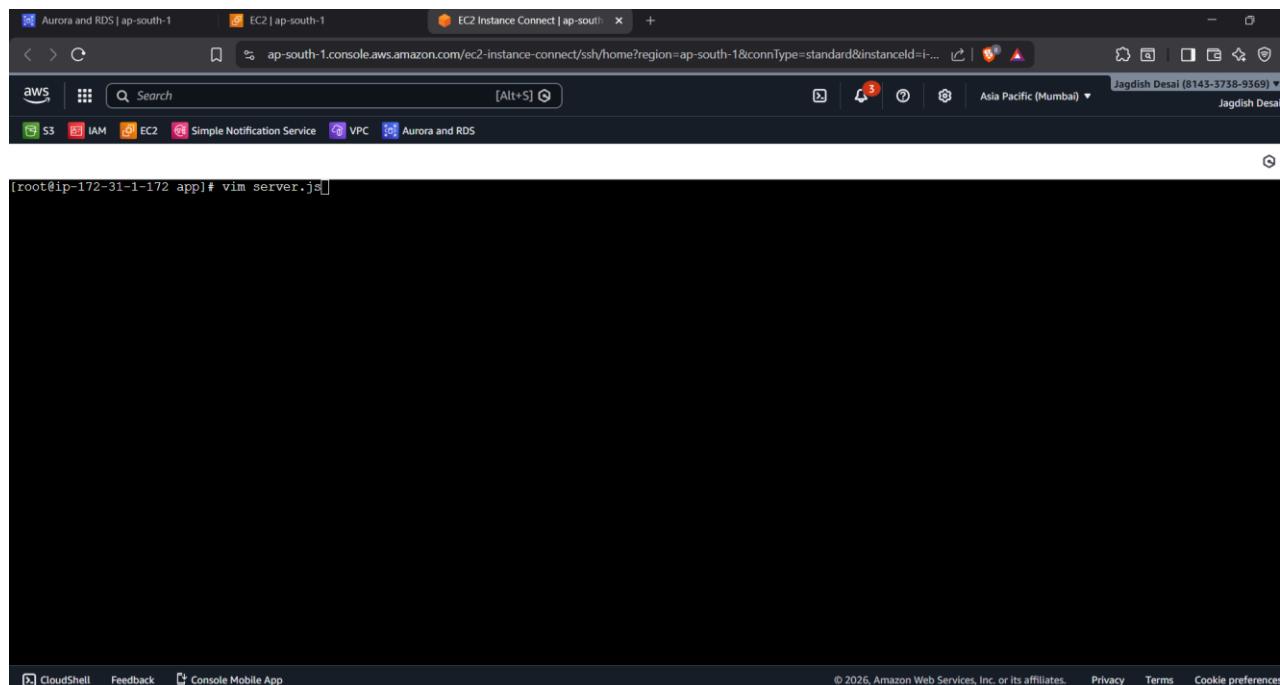
**1.Mkdir app**

**2.cd app**

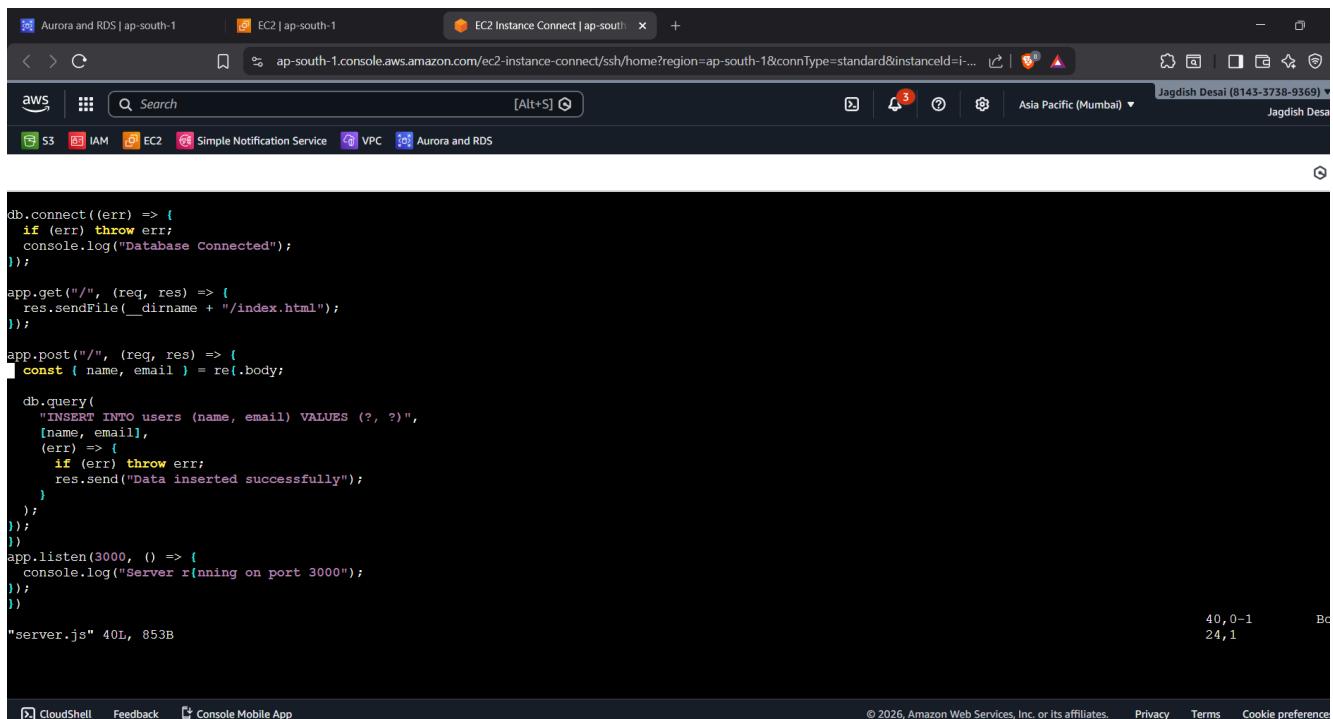
**3. npm init -y initializes a Node project instantly.**

It creates package.json with default values so you can start installing packages.

## Creating backend server.js {nodejs} file : vim server.js



```
[root@ip-172-31-1-172 app]# vim server.js
```



```
db.connect((err) => {
  if (err) throw err;
  console.log("Database Connected");
});

app.get("/", (req, res) => {
  res.sendFile(__dirname + "/index.html");
});

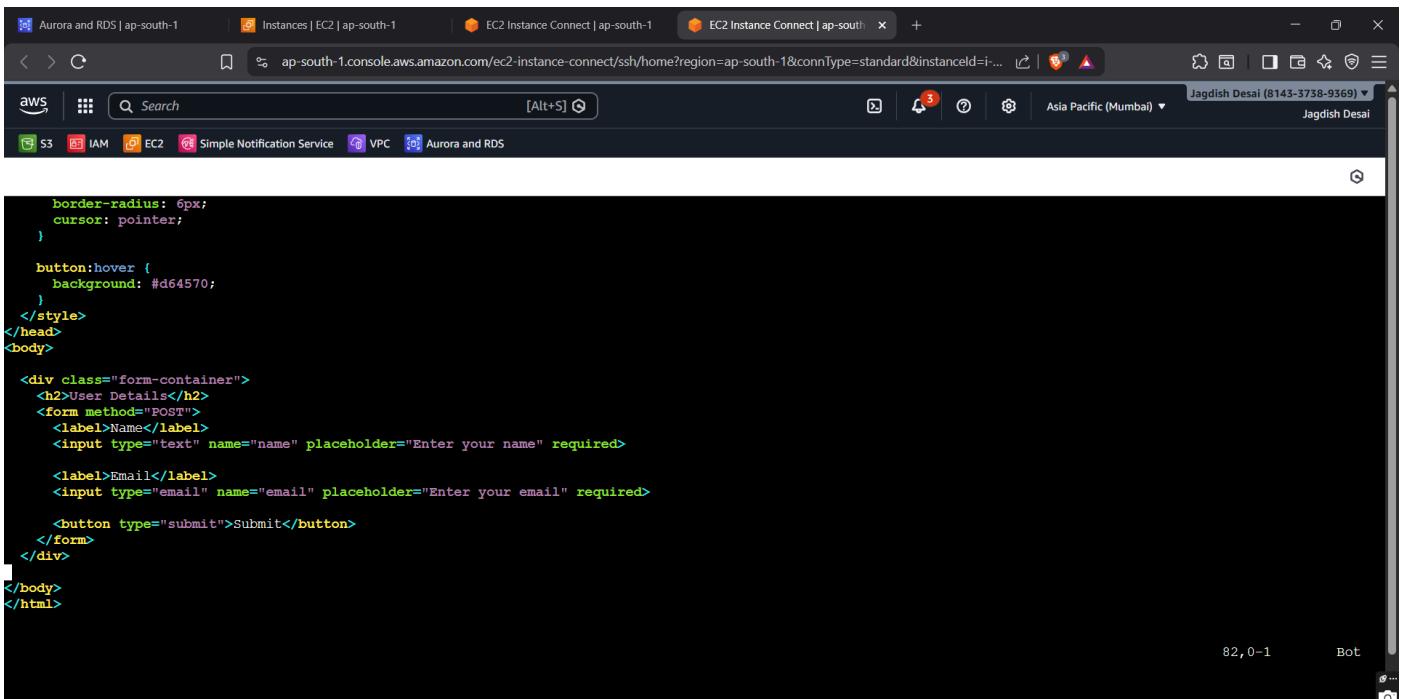
app.post("/", (req, res) => {
  const { name, email } = req.body;

  db.query(
    "INSERT INTO users (name, email) VALUES (?, ?)",
    [name, email],
    (err) => {
      if (err) throw err;
      res.send("Data inserted successfully");
    }
  );
})
app.listen(3000, () => {
  console.log("Server running on port 3000");
});
```

"server.js" 40L, 853B

40, 0-1      BC  
24, 1

## Step 5 : Creating index.html file for frontend



```
border-radius: 6px;
cursor: pointer;
}

button:hover {
background: #d64570;
}
</style>
</head>
<body>

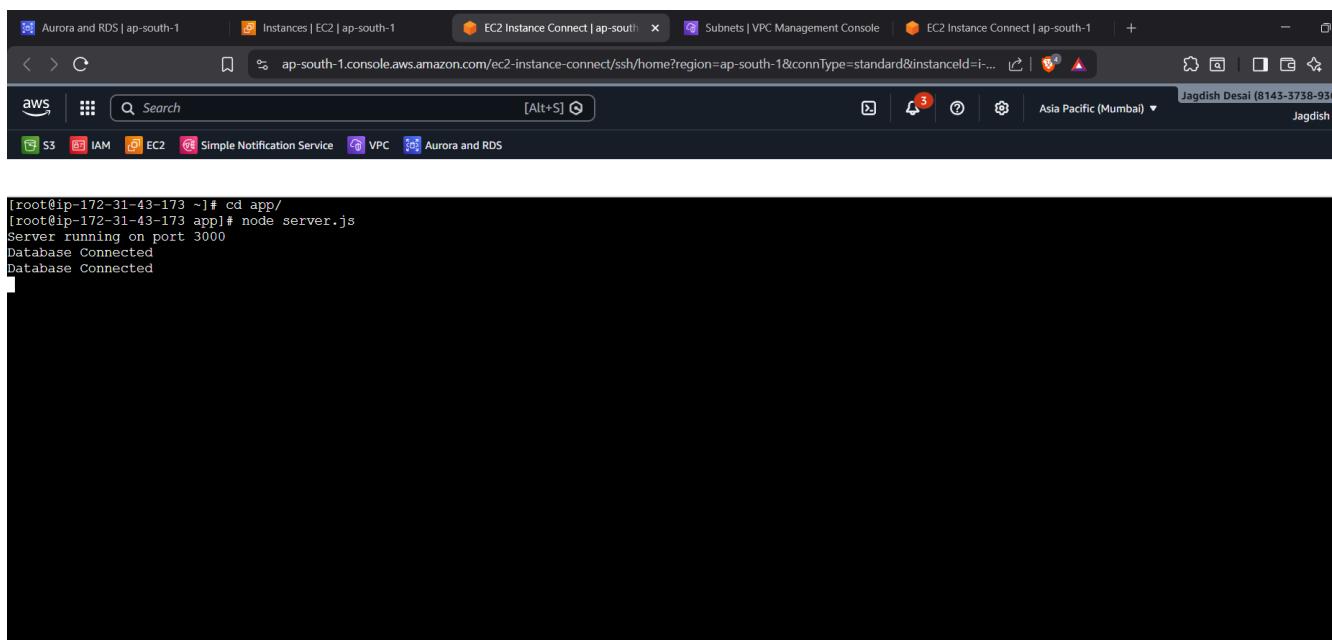
<div class="form-container">
<h2>User Details</h2>
<form method="POST">
<label>Name</label>
<input type="text" name="name" placeholder="Enter your name" required>

<label>Email</label>
<input type="email" name="email" placeholder="Enter your email" required>

<button type="submit">Submit</button>
</form>
</div>

</body>
</html>
```

## Step 6: Then Run Application using node server.js command and checking that database connection is established or not:

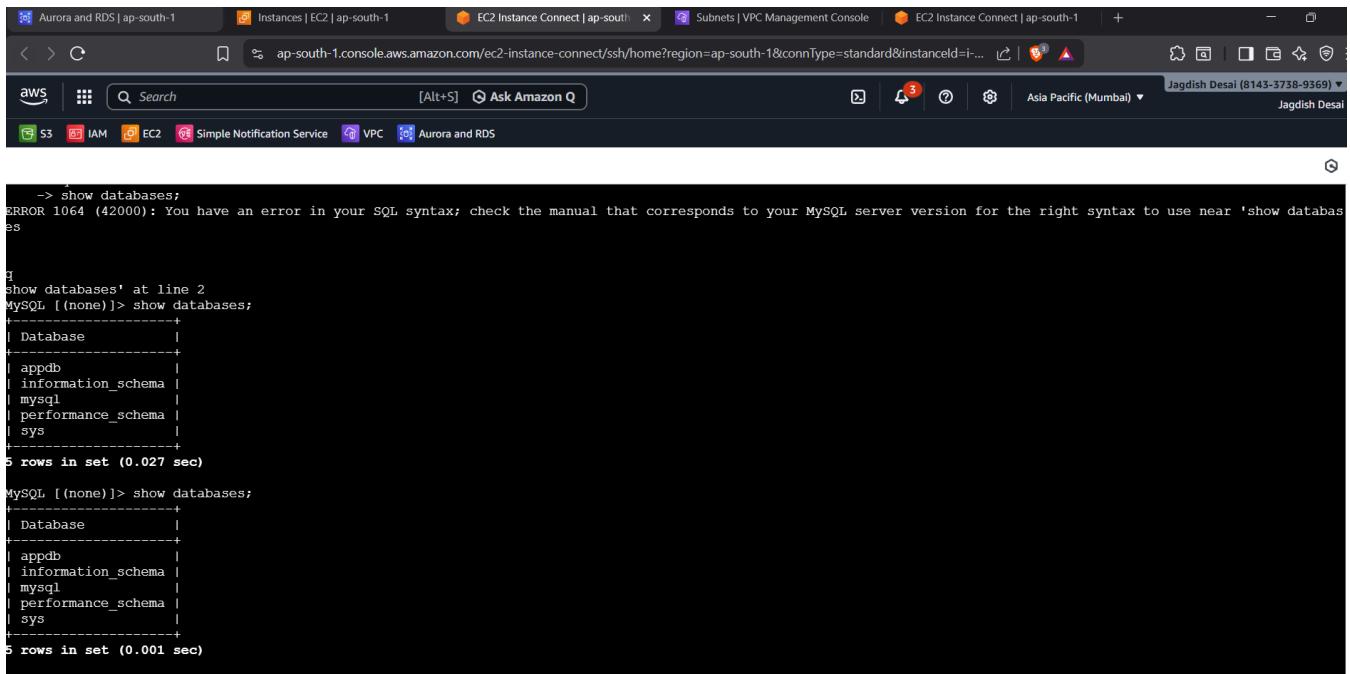


```
[root@ip-172-31-43-173 ~]# cd app/
[root@ip-172-31-43-173 app]# node server.js
Server running on port 3000
Database Connected
Database Connected
```

i-08f48e80bd1d38d2a (task-1)

## Step 7: Going to database on ec2 using endpoint and create appdb named database using command.

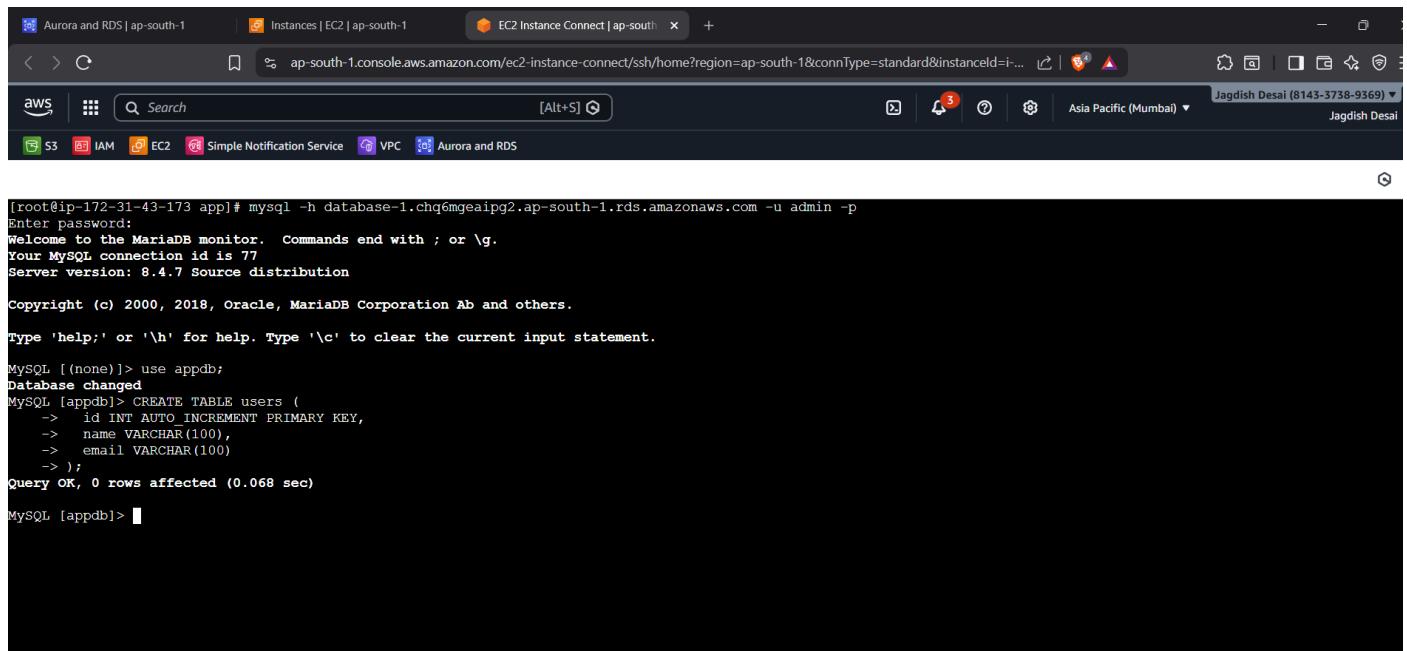
create database appdb;



```
-> show databases;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'show databases' at line 2
MySQL [(none)]> show databases;
+-----+
| Database |
+-----+
| appdb   |
| information_schema |
| mysql    |
| performance_schema |
| sys      |
+-----+
5 rows in set (0.027 sec)

MySQL [(none)]> show databases;
+-----+
| Database |
+-----+
| appdb   |
| information_schema |
| mysql    |
| performance_schema |
| sys      |
+-----+
5 rows in set (0.001 sec)
```

## Step 8: After check That Using Command use appdb; for using database after that creating table users for storing data into that table.



```
[root@ip-172-31-43-173 app]# mysql -h database-1.chq6mgeap2.ap-south-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 77
Server version: 8.4.7 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

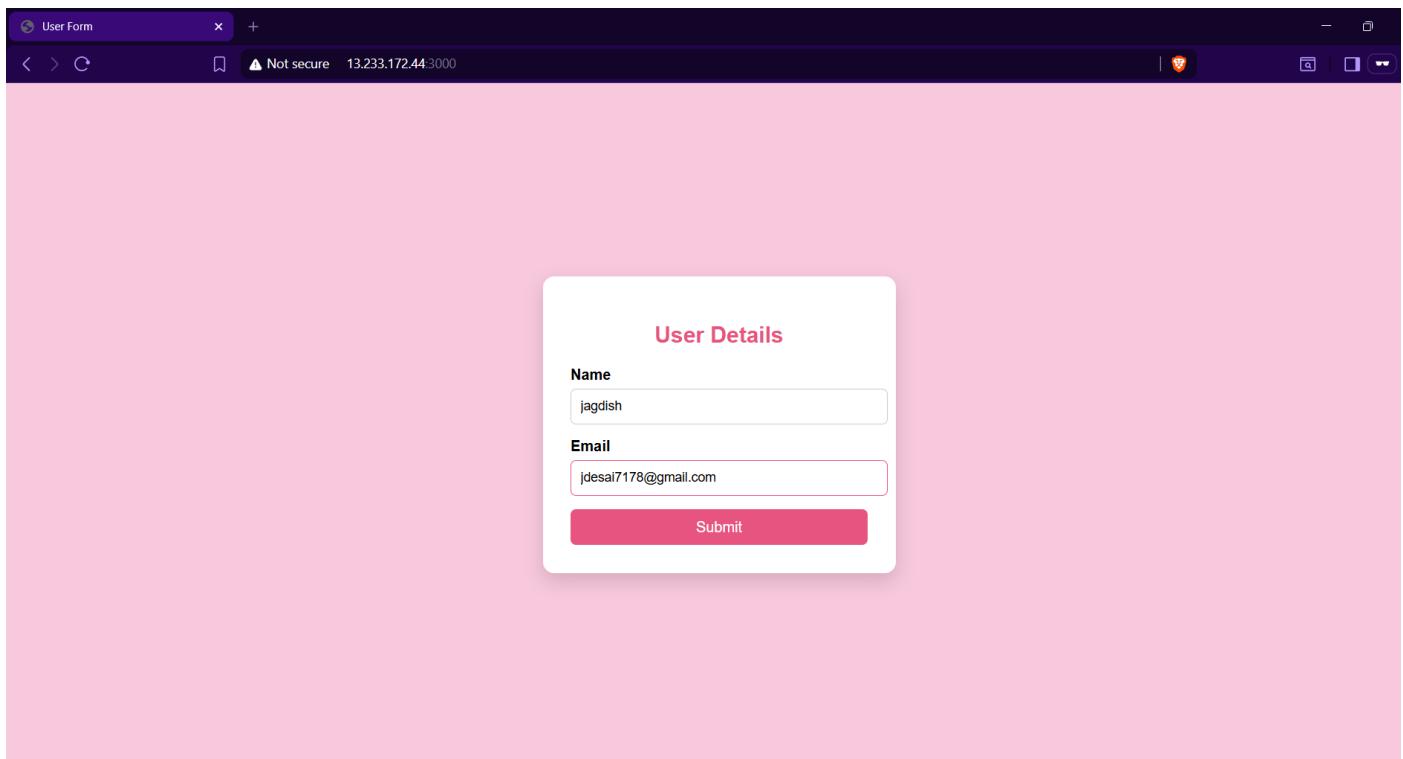
MySQL [(none)]> use appdb;
Database changed
MySQL [appdb]> CREATE TABLE users (
->     id INT AUTO_INCREMENT PRIMARY KEY,
->     name VARCHAR(100),
->     email VARCHAR(100)
-> );
Query OK, 0 rows affected (0.068 sec)

MySQL [appdb]> |
```

## **Step 9: Open browser and use public ip of task-1 instance**

**http://13.233.172.44:3000/**

### **Result:**



### **After Submitting Details form:**



**After Submitting form details it will shows data inserted successfully.**

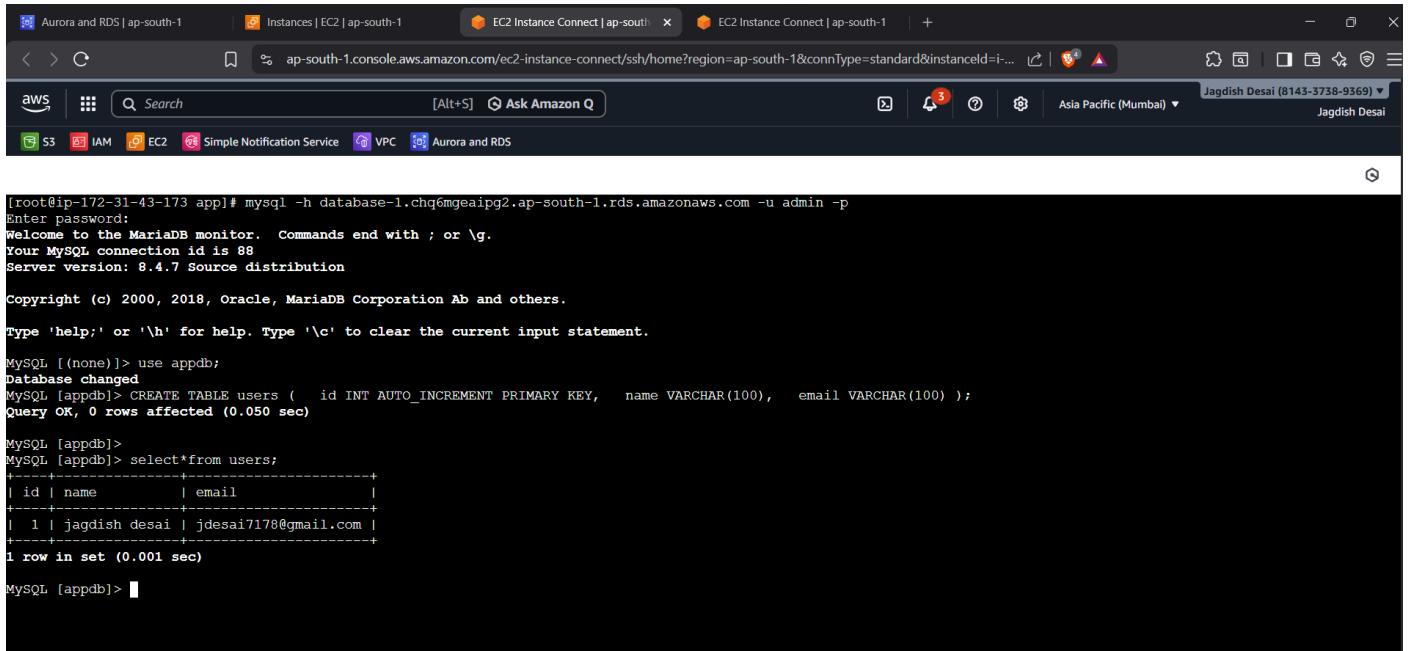
**Means our data successfully stored in appdb database users table.**

**Step 10: After Inserting data go to your instance and again check mysql database which is named appdb. By using commands**

**1.use appdb;**

**2.select\*from users; //this will show you table data which can be inserted by user**

## Step 10: appdb Database users data



The screenshot shows a browser window for EC2 Instance Connect. The address bar displays the URL: ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?region=ap-south-1&connType=standard&instanceId=i-... . The page content is a terminal window showing a MySQL session. The user is connected to the database 'appdb' and has run a 'select \* from users;' query, which returns one row of data: id 1, name jagdish desai, and email jdesai7178@gmail.com.

```
[root@ip-172-31-43-173 app]# mysql -h database-1.chq6mgeaipg2.ap-south-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 88
Server version: 8.4.7 Source distribution

copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

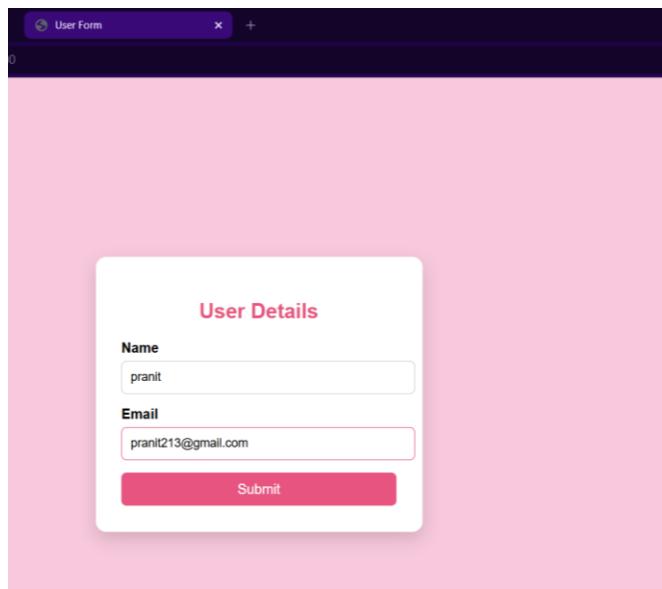
MySQL [(none)]> use appdb;
Database changed
MySQL [appdb]> CREATE TABLE users ( id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), email VARCHAR(100) );
Query OK, 0 rows affected (0.050 sec)

MySQL [appdb]>
MySQL [appdb]> select*from users;
+----+-----+-----+
| id | name | email |
+----+-----+-----+
| 1 | jagdish desai | jdesai7178@gmail.com |
+----+-----+-----+
1 row in set (0.001 sec)

MySQL [appdb]>
```

You will see the data successfully stored in appdb database users table.

**After entering some another users details:**



**Database Automatically Updated :**

```
MySQL [appdb]> select*from users;
+----+-----+-----+
| id | name      | email           |
+----+-----+-----+
| 1  | jagdish desai | jdesai7178@gmail.com |
+----+-----+-----+
1 row in set (0.001 sec)

MySQL [appdb]> select*from users;
+----+-----+-----+
| id | name      | email           |
+----+-----+-----+
| 1  | jagdish desai | jdesai7178@gmail.com |
| 2  | pranit     | pranit213@gmail.com |
+----+-----+-----+
2 rows in set (0.001 sec)
```

# **Project Workflow**

## **Node.js + Express + MySQL (AWS RDS)**

### **1. User Access**

The user opens the web application using the public IP address of the EC2 instance in a browser.

### **2. Frontend Interaction**

A simple HTML form is displayed on the browser.

The user enters their **name** and **email** and clicks the submit button.

### **3. Request Sent to Server**

When the form is submitted, the browser sends the user data to the backend server using an HTTP POST request.

### **4. Backend Processing**

The Node.js application running on the EC2 instance receives the request. Express.js processes the request and extracts the user details from the form.

### **5. Database Connection**

The backend application connects to the MySQL database hosted on AWS RDS using the RDS endpoint and credentials.

### **6. Data Storage**

The received name and email are inserted into the **users** table in the MySQL RDS database.

### **7. Response to User**

After successful insertion, the server sends a confirmation message back to the browser.

## 8. Continuous Service

The server keeps running and can handle multiple user requests without stopping.

### Overall Data Flow

User Browser → EC2 (Node.js + Express) → MySQL RDS → Response to User

### Code Used In Task:

#### Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>User Form</title>
  <style>
    body {
      margin: 0;
      padding: 0;
      height: 100vh;
      display: flex;
      justify-content: center;
      align-items: center;
      background: #f8c8dc; /* ice pink background */
      font-family: Arial, sans-serif;
    }

    .form-container {
      background: white;
      padding: 30px;
```

```
border-radius: 12px;  
width: 320px;  
box-shadow: 0 8px 20px rgba(0, 0, 0, 0.15);  
}
```

```
.form-container h2 {  
text-align: center;  
margin-bottom: 20px;  
color: #e75480;  
}
```

```
label {  
font-weight: bold;  
display: block;  
margin-bottom: 5px;  
}
```

```
input {  
width: 100%;  
padding: 10px;  
margin-bottom: 15px;  
border-radius: 6px;  
border: 1px solid #ccc;  
font-size: 14px;  
}
```

```
input:focus {  
outline: none;  
border-color: #e75480;  
}
```

```
button {  
    width: 100%;  
    padding: 10px;  
    background: #e75480;  
    border: none;  
    color: white;  
    font-size: 16px;  
    border-radius: 6px;  
    cursor: pointer;  
}  
button:hover {  
    background: #d64570;  
}  
</style>  
</head>  
<body>  
<div class="form-container">  
    <h2>User Details</h2>  
    <form method="POST">  
        <label>Name</label>  
        <input type="text" name="name" placeholder="Enter your name" required>  
  
        <label>Email</label>  
        <input type="email" name="email" placeholder="Enter your email" required>  
        <button type="submit">Submit</button>  
    </form>  
</div>  
  
</body>  
</html>
```

## Server.js

```
const express = require("express");
const mysql = require("mysql2");
const bodyParser = require("body-parser");
const app = express();
app.use(bodyParser.urlencoded({ extended: true }));
const db = mysql.createConnection({
  host: "database-1.chq6mgeaipg2.ap-south-1.rds.amazonaws.com",
  user: "admin",
  password: "shankarrao71",
  database: "appdb"
});
db.connect(err => {
  if (err) throw err;
  console.log("Database Connected");
});
app.post("/", (req, res) => {
  const { name, email } = req.body;
  db.query(
    "INSERT INTO users (name, email) VALUES (?, ?)",
    [name, email],
    () => res.send("Data inserted successfully")
  );
});
app.get("/", (req, res) => {
  res.sendFile(__dirname + "/index.html");
});
```

```

app.listen(3000, () => {
  console.log("Server running on port 3000");
});

db.connect((err) => {
  if (err) {
    console.error("DB connection failed:", err.message);
    process.exit(1);
  }
  console.log("Database Connected");
});

```

### **mysql commands I have used:**

<b>SQL Command</b>	<b>Description</b>
<b>CREATE DATABASE appdb;</b>	Creates a new database for the project
<b>USE appdb;</b>	Selects the database to work in
<b>CREATE TABLE users ( id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), email VARCHAR(100) );</b>	Creates a table to store user details
<b>SELECT * FROM users;</b>	Displays all records from the users table
<b>SHOW TABLES;</b>	Shows all tables in the database
<b>EXIT;</b>	Exits the MySQL database session