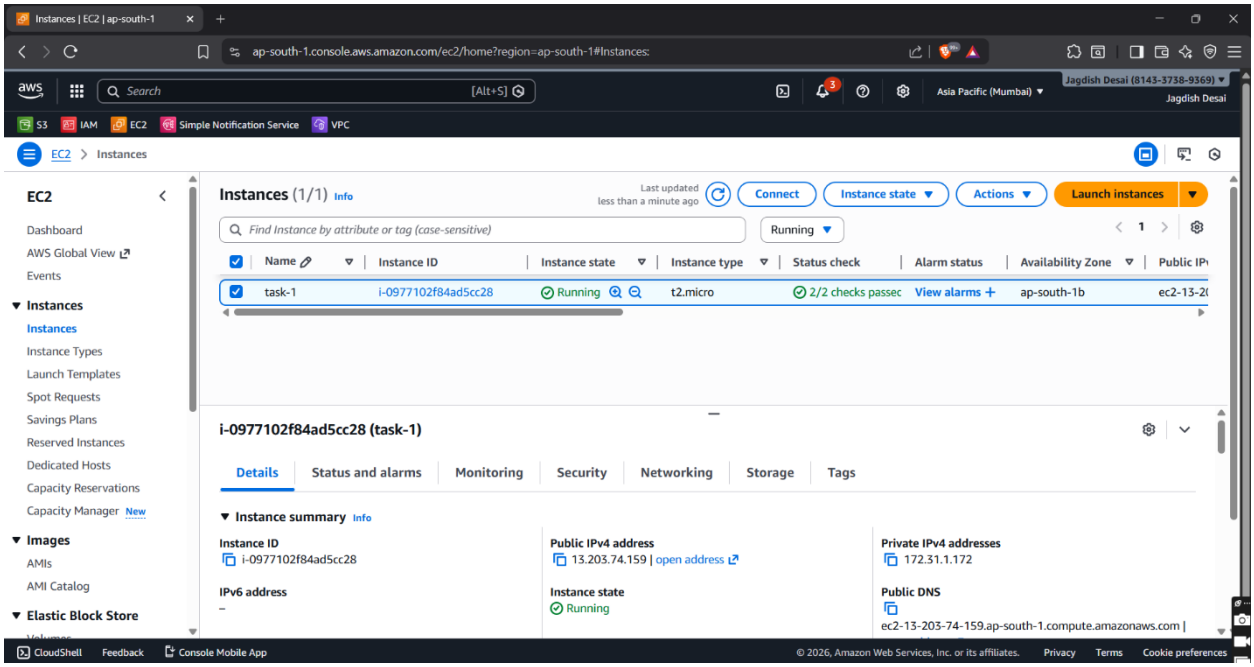
Task 1-

1. Application

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

Steps:

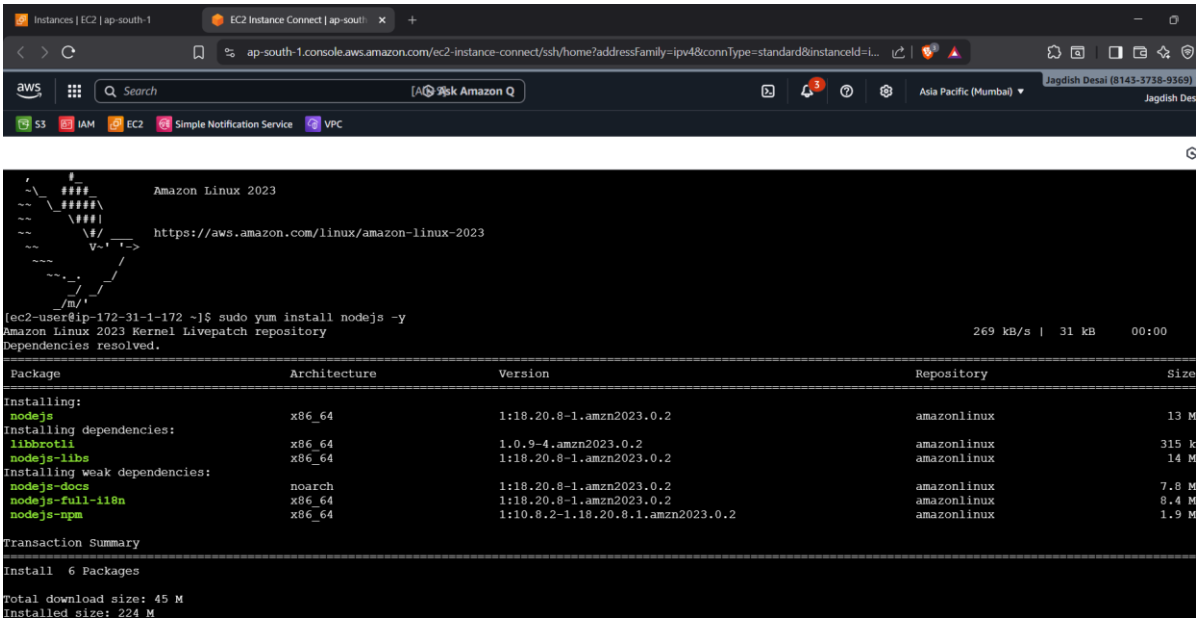
Step 1: Created an instance task-1



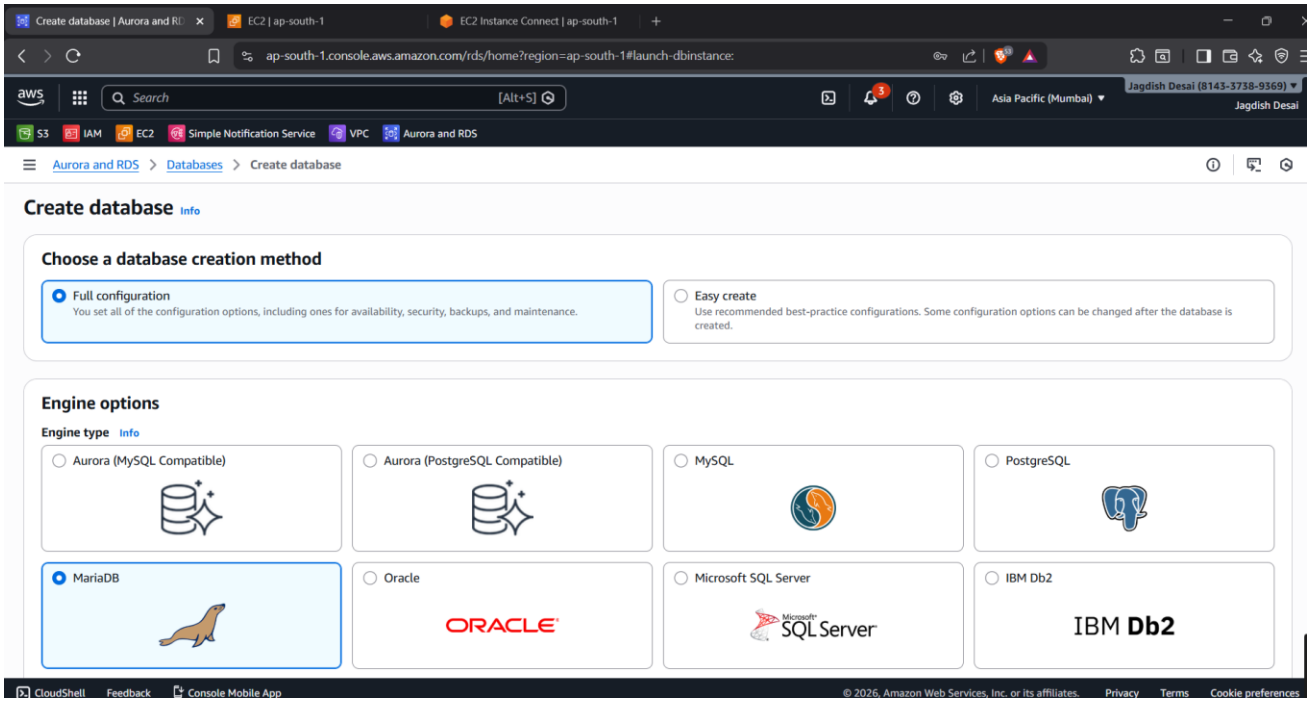
we

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

sudo yum install nodejs -y



Step 3: Going to RDS and creating a mariadb database

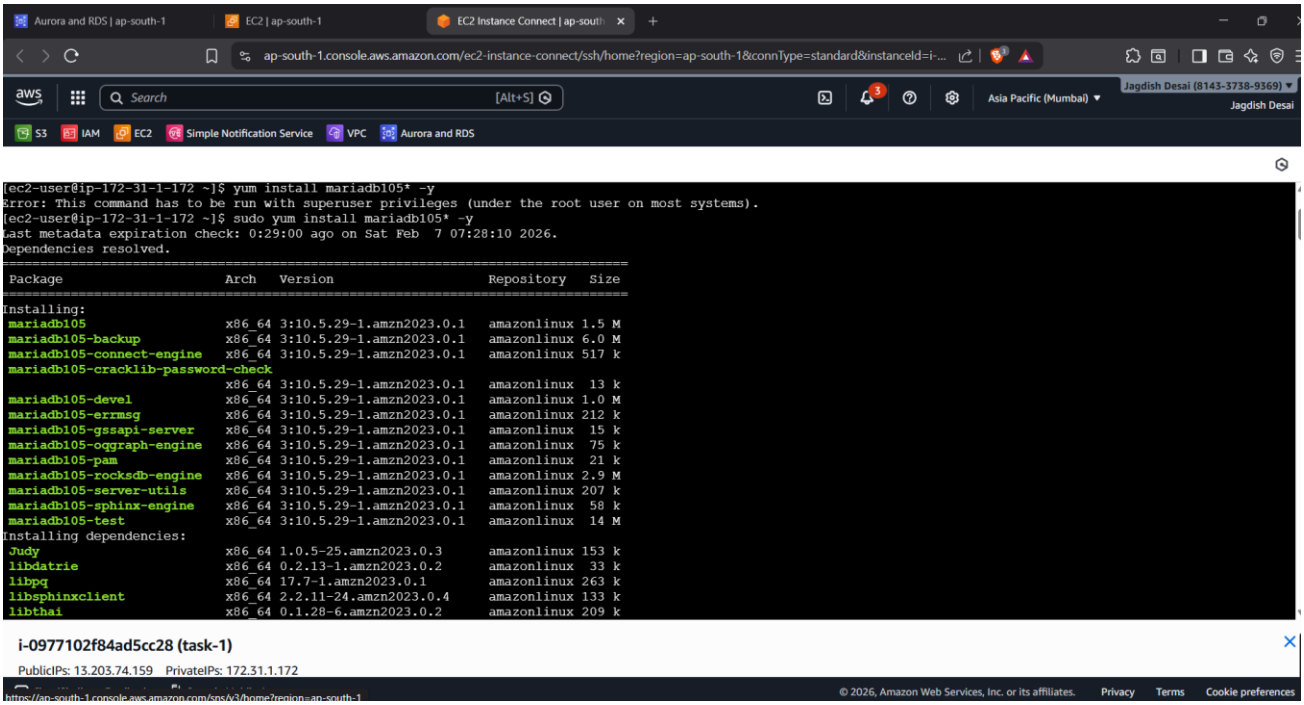


Allow port 3306 mysql port in security groups inbound rules.

After creation of database install mariadb tools on ec2 instance:

It installs MariaDB client tools on the EC2 instance

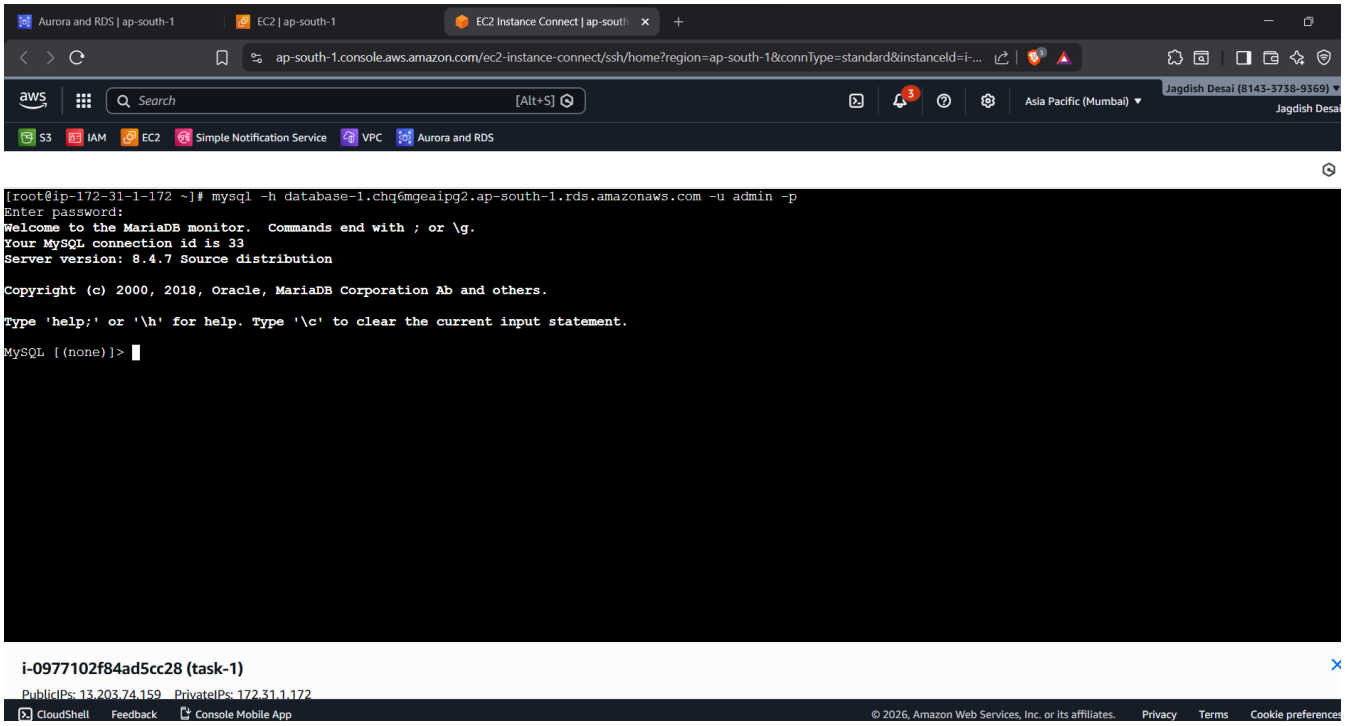
command: yum install mariadb105* -y



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.

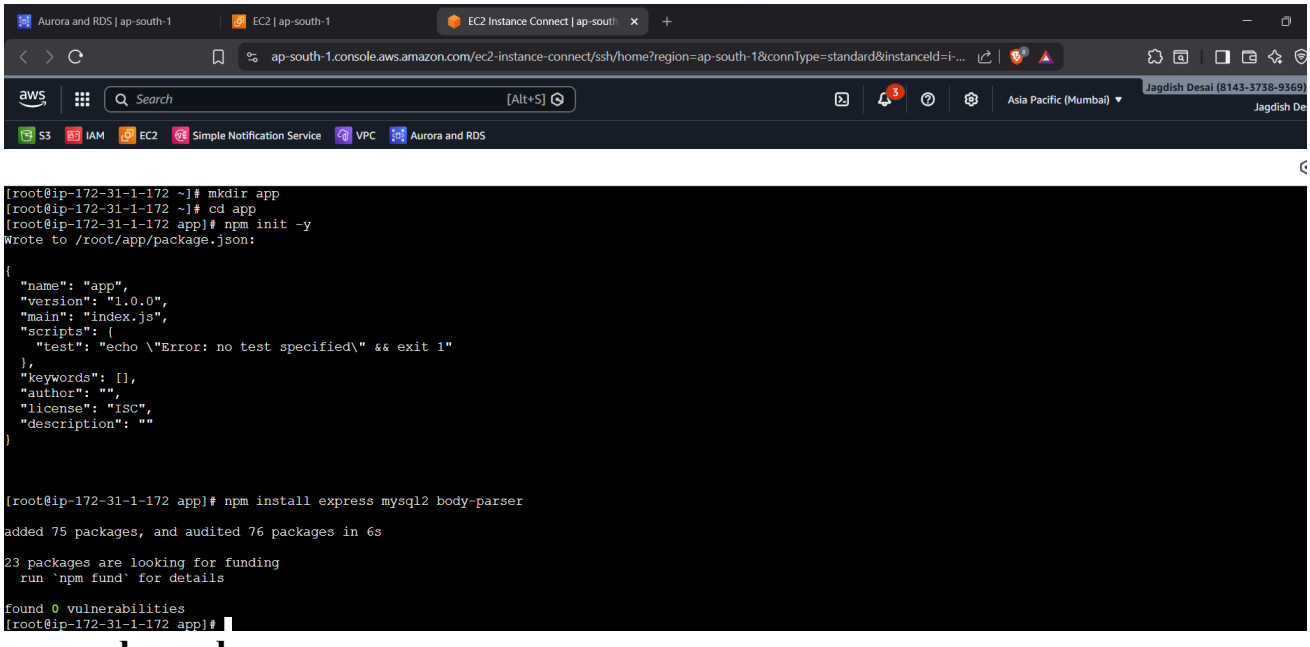
Command

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



Exit;

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



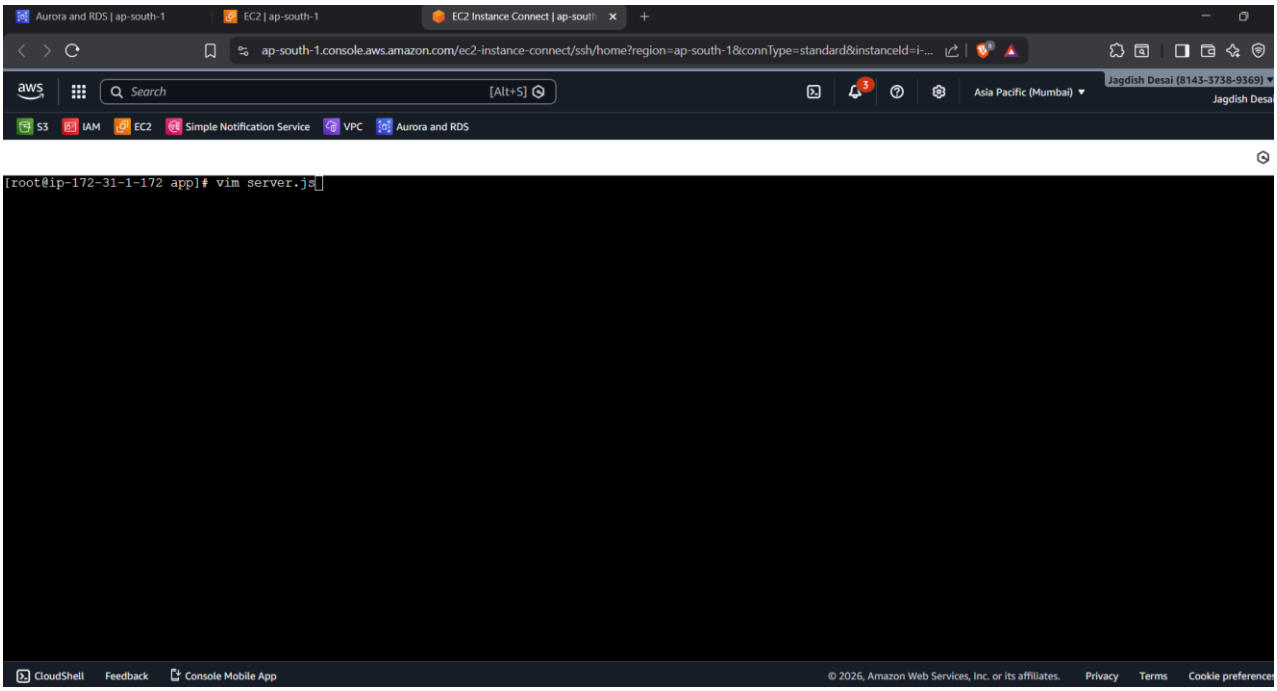
Commands used:

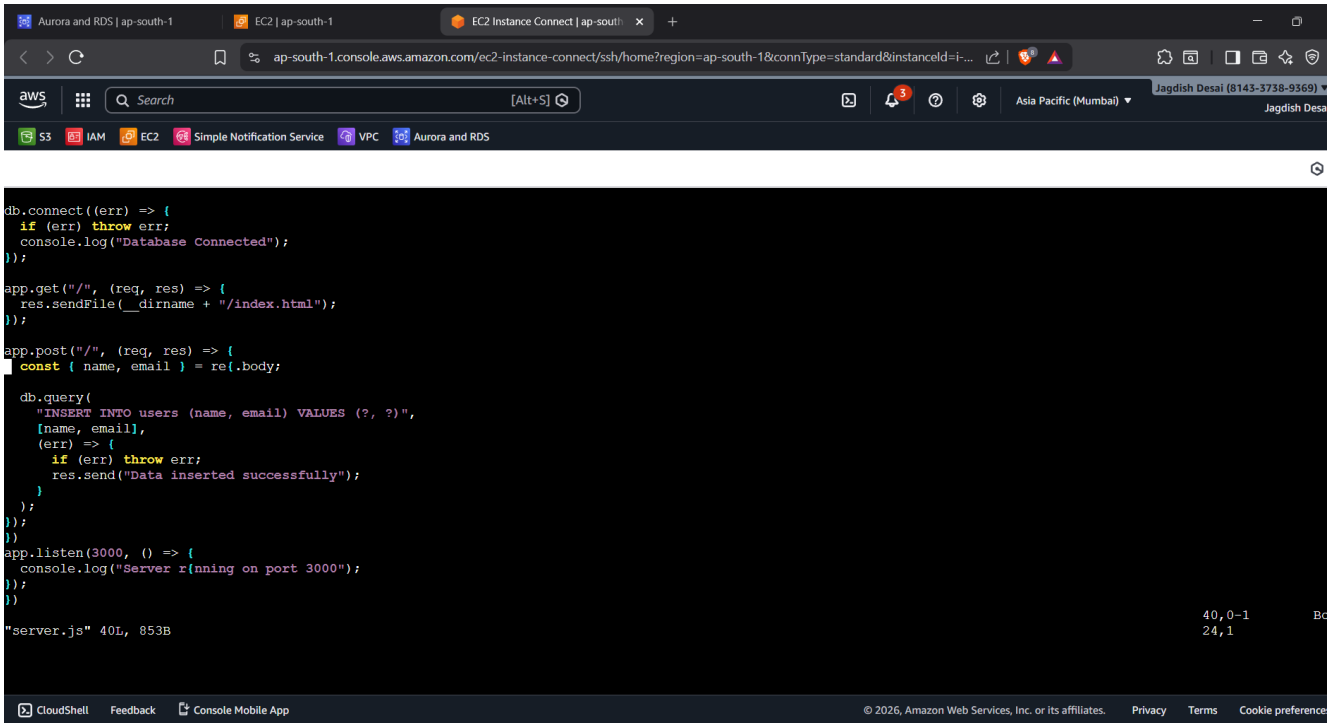
1.**Mkdir app**

2.**cd app**

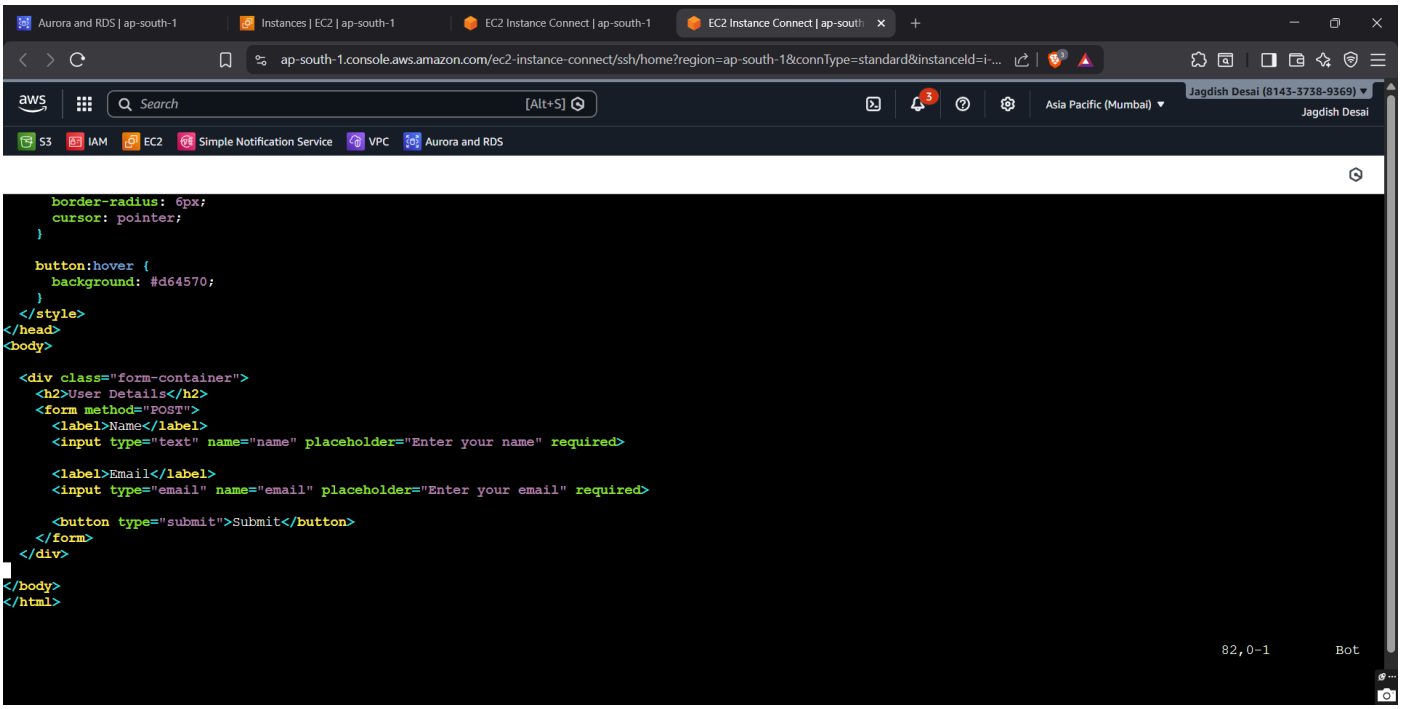
3. npm init -y initializes a Node project instantly.
It creates package.json with default values so you can start installing packages.

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

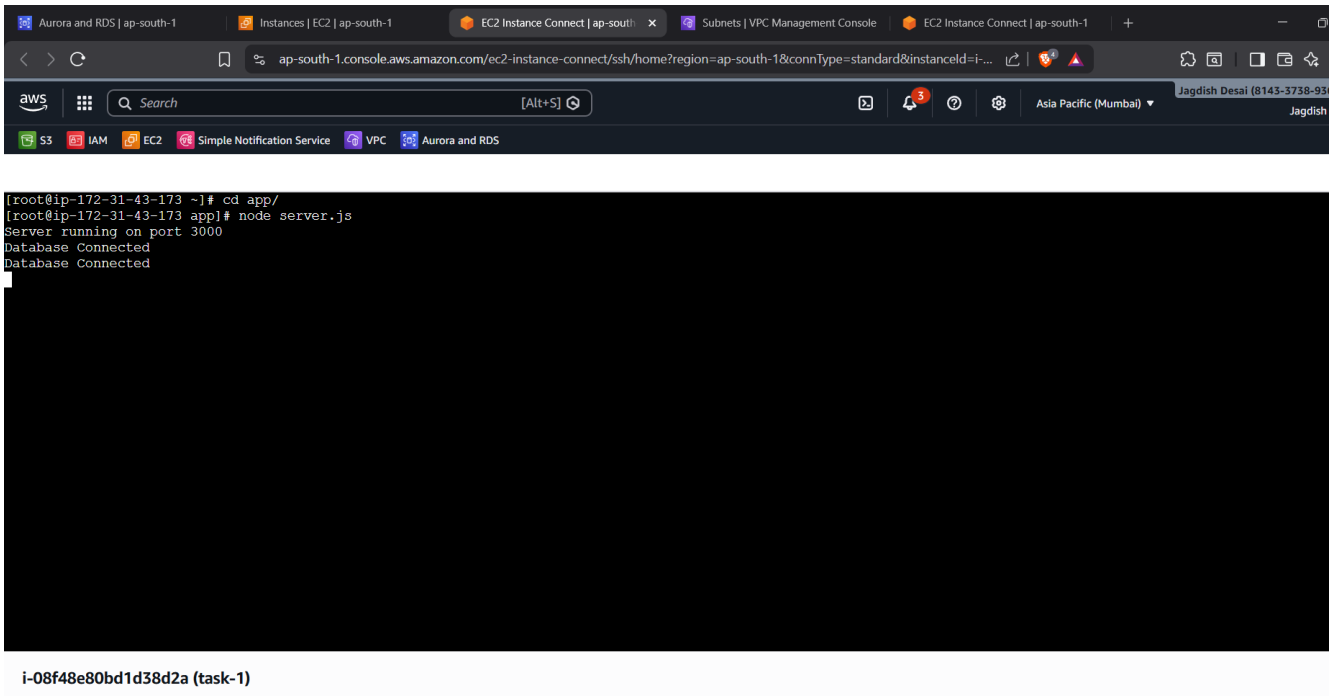




Step 5 : Creating index.html file for frontend

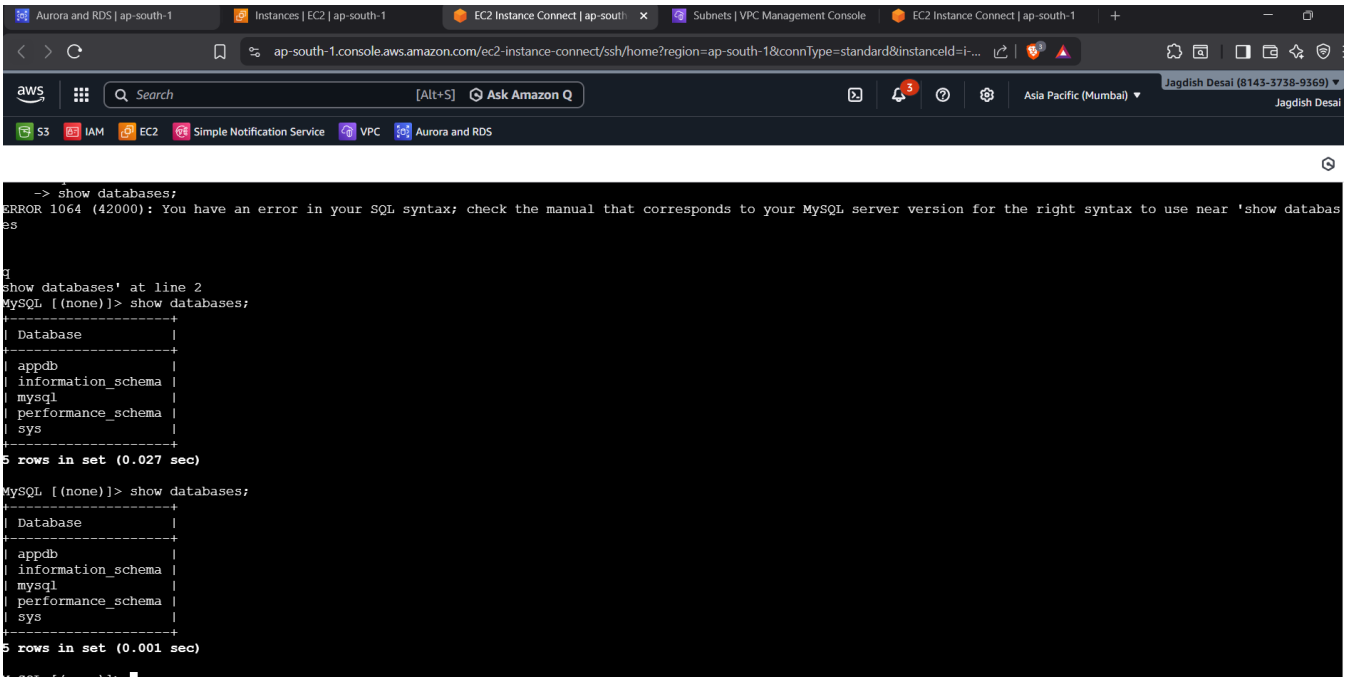


Step 6: Then Run Application using node server.js command and checking that database connection is established or not:

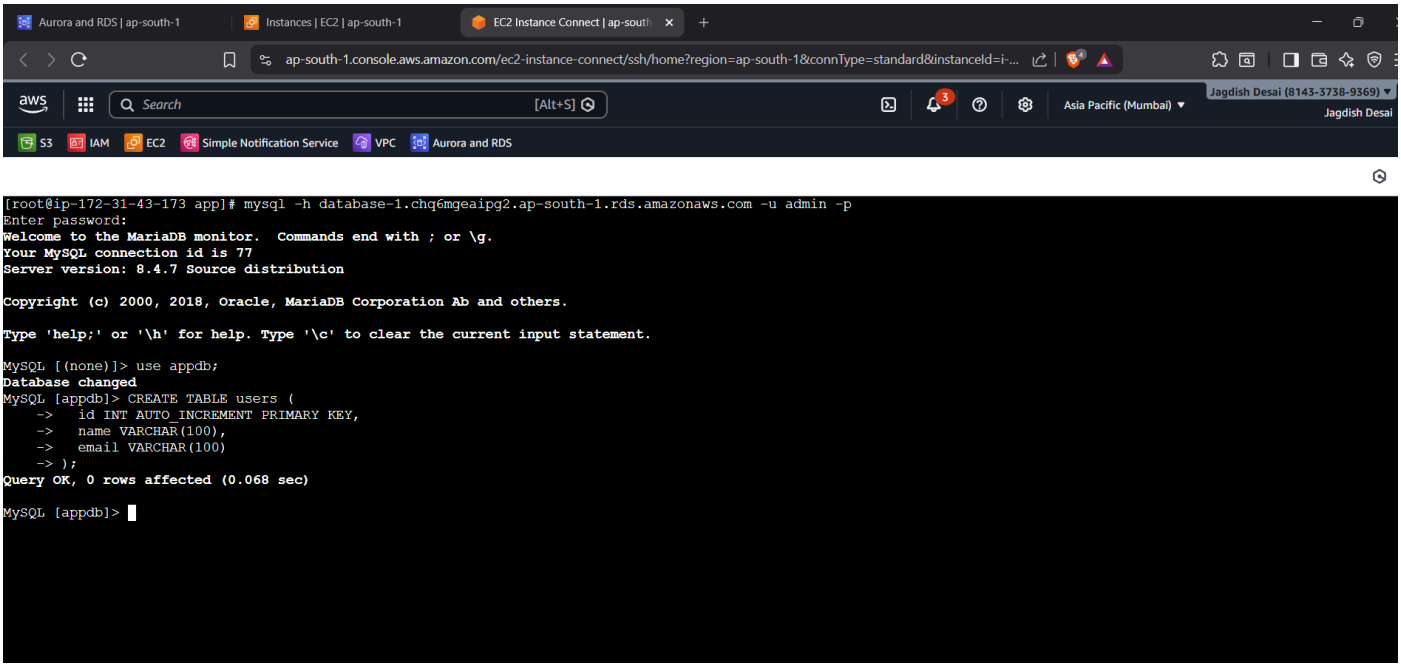


Step 7: Going to to database on ec2 using endpoint and create appdb named database using command.

create database appdb;



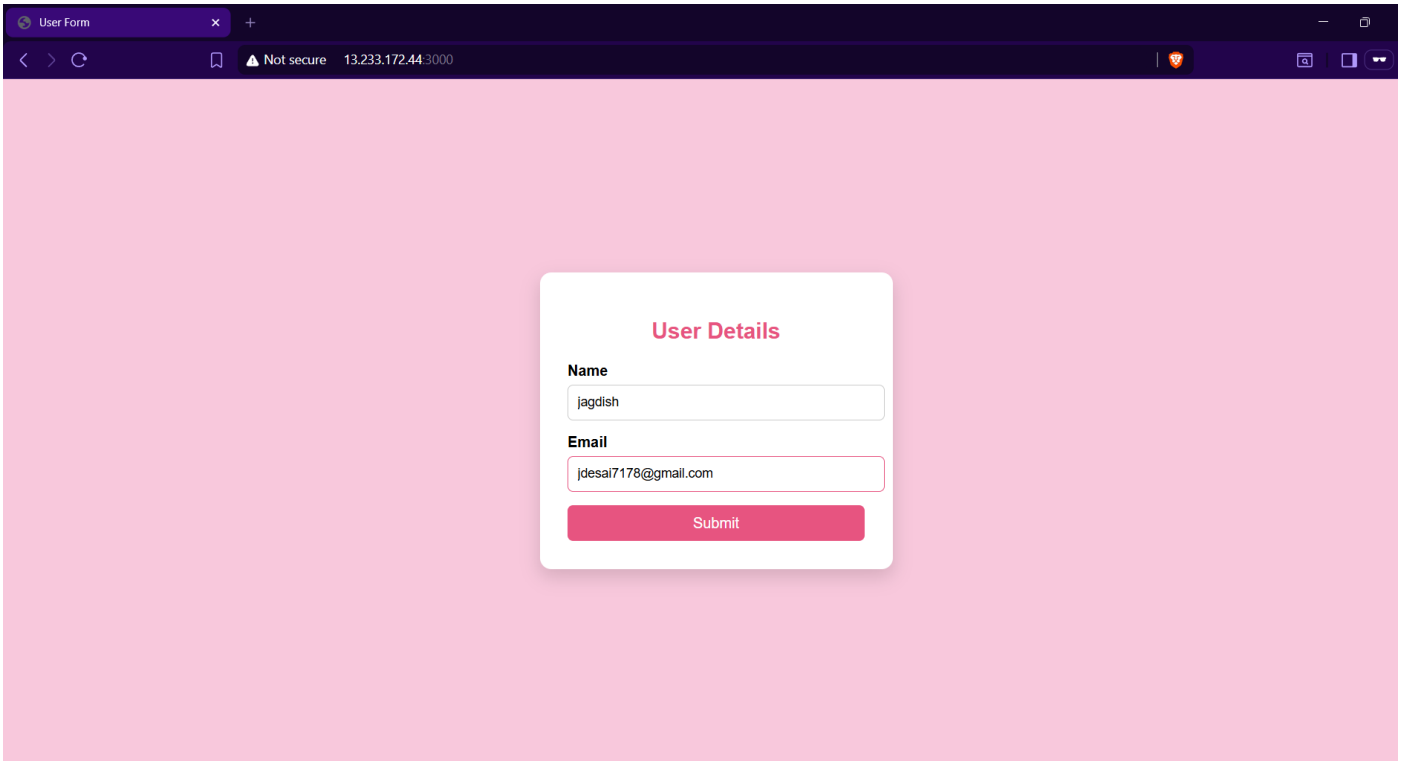
Step 8: After check That Using Command <u>use appdb;</u> for using database after that creating table users for storing data into that table.



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.

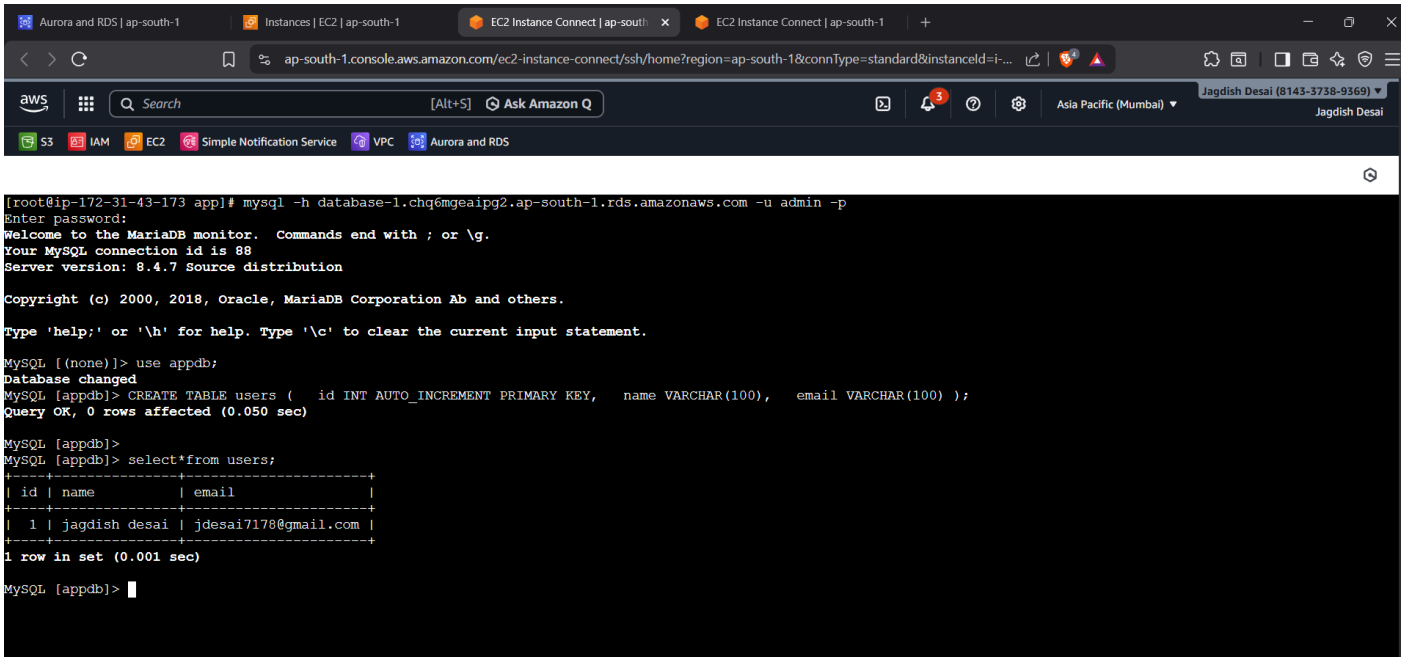
Means our data successfully stored in appdb database users table.

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

1.use appdb;

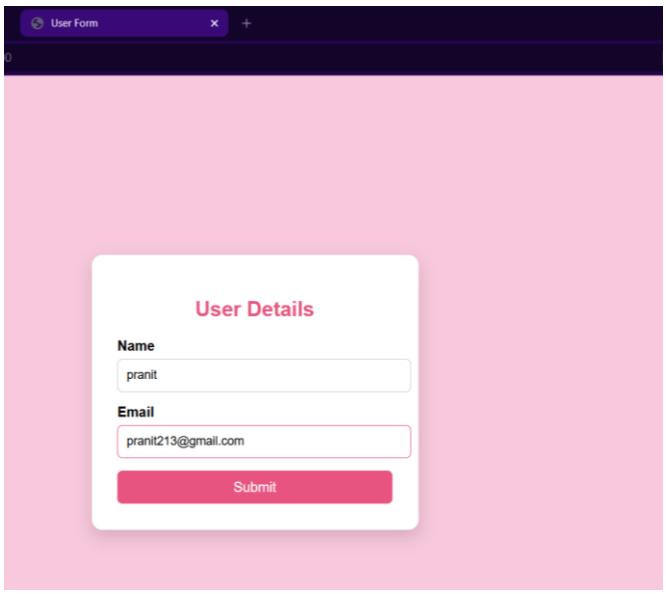
2.select*from users; //this will show you table data which can be inserted by user

Step 10: appdb Database users data



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:

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