Great—this is the perfect next step

Now you'll connect **Express.js + MySQL** and store real data instead of memory.

I'll give you a **simple, beginner-friendly API** for a table with:
`id, name, city, phone`

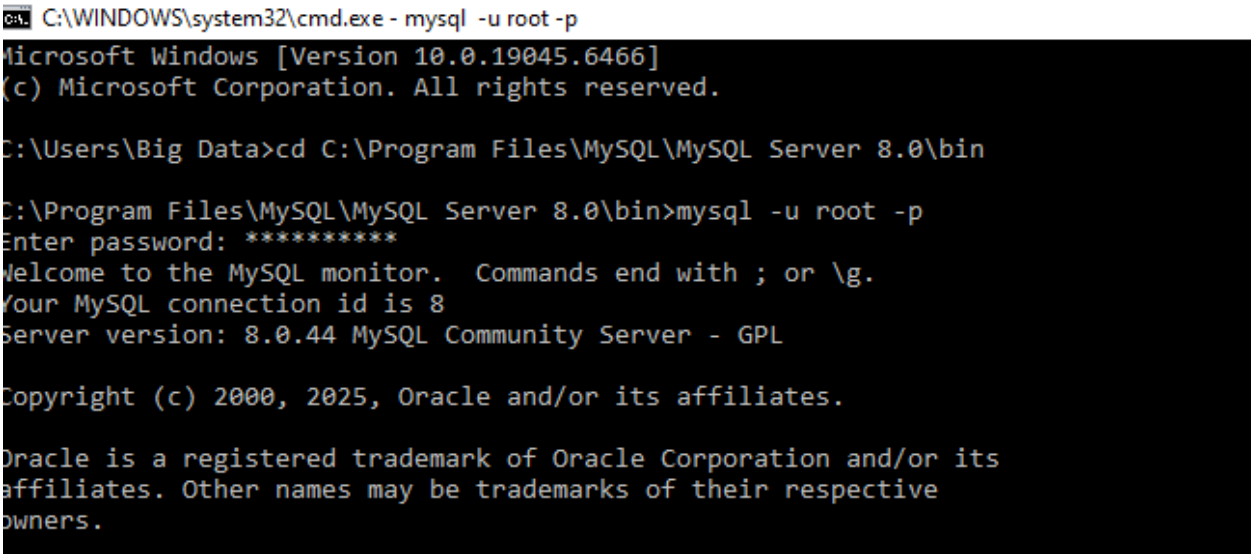
□ Step 1: Install Packages

Run in your project folder:

```
npm install express mysql2
```

□ Step 2: Create MySQL Database & Table

Open MySQL and Run:-

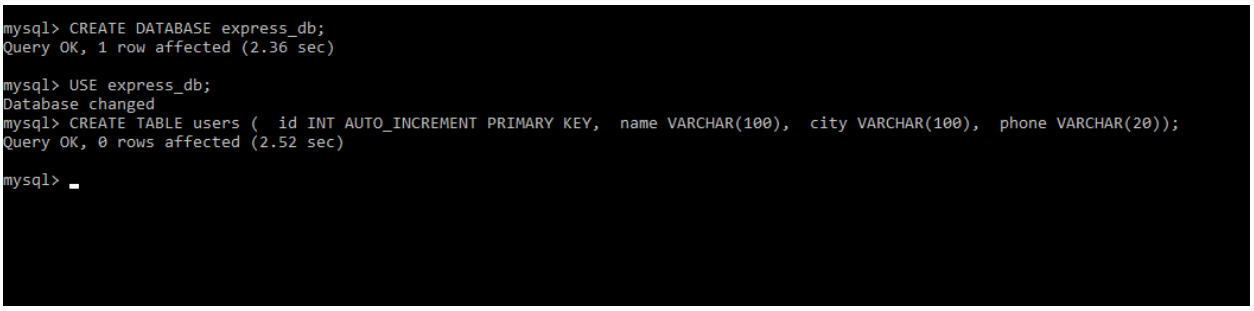


And perform following task :-

```
CREATE DATABASE express_db;

USE express_db;

CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100),
  city VARCHAR(100),
  phone VARCHAR(20)
);
```



□ Step 3: Full Express + MySQL Code

Create file: `app.js:-`

```
const express = require('express');
const mysql = require('mysql2');

const app = express();
app.use(express.json());

// ------------------
// MySQL Connection
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '', // put your MySQL password
  database: 'express_db'
});

// Connect to DB
db.connect(err => {
  if (err) {
    console.log('Database connection failed:', err);
  } else {
    console.log('Connected to MySQL');
  }
});

// ------------------
// GET all users
app.get('/users', (req, res) => {
  db.query('SELECT * FROM users', (err, results) => {
    if (err) {
      res.status(500).json({ error: err.message });
    } else {
      res.json(results);
    }
  });
});

// ------------------
// GET single user
app.get('/users/:id', (req, res) => {
  const id = req.params.id;

  db.query('SELECT * FROM users WHERE id = ?', [id], (err, results) => {
    if (err) {
      res.status(500).json({ error: err.message });
    } else if (results.length === 0) {
      res.status(404).json({ error: "User not found" });
    } else {
      res.json(results[0]);
    }
  });
});
```

```
// ------------------
// CREATE user
app.post('/users', (req, res) => {
  const { name, city, phone } = req.body;

  const sql = 'INSERT INTO users (name, city, phone) VALUES (?, ?, ?)';
  db.query(sql, [name, city, phone], (err, result) => {
    if (err) {
      res.status(500).json({ error: err.message });
    } else {
      res.json({
        message: "User created",
        id: result.insertId
      });
    }
  });
});

// ------------------
// UPDATE user
app.put('/users/:id', (req, res) => {
  const id = req.params.id;
  const { name, city, phone } = req.body;

  const sql = 'UPDATE users SET name=?, city=?, phone=? WHERE id=?';
  db.query(sql, [name, city, phone, id], (err, result) => {
    if (err) {
      res.status(500).json({ error: err.message });
    } else {
      res.json({ message: "User updated" });
    }
  });
});

// ------------------
// DELETE user
app.delete('/users/:id', (req, res) => {
  const id = req.params.id;

  db.query('DELETE FROM users WHERE id=?', [id], (err, result) => {
    if (err) {
      res.status(500).json({ error: err.message });
    } else {
      res.json({ message: "User deleted" });
    }
  });
});

// ------------------
app.listen(3000, () => {
  console.log('Server running on http://localhost:3000');
});
```

□ Test with Windows CMD (IMPORTANT)

□ Create user

```
curl -X POST http://localhost:3000/users -H "Content-Type: application/json" -d "{\"name\":\"Alice\",\"city\":\"Mumbai\",\"phone\":\"1234567890\"}"
```

□ Get all users

```
curl http://localhost:3000/users
```

□ Get single user

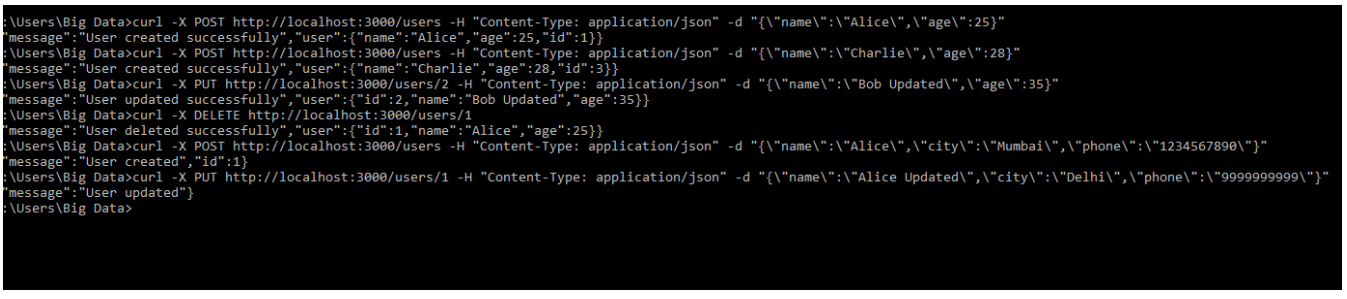
```
curl http://localhost:3000/users/1
```

□ Update user

```
curl -X PUT http://localhost:3000/users/1 -H "Content-Type: application/json" -d "{\"name\":\"Alice Updated\",\"city\":\"Delhi\",\"phone\":\"9999999999\"}"
```

□ Delete user

```
curl -X DELETE http://localhost:3000/users/1
```



Great! You're doing the right thing by understanding **line by line**. I'll explain this in **very simple beginner language**.

---

1. Import Libraries

```
const express = require('express');
```

- Loads **Express framework**.
- Used to create server and APIs.

```
const mysql = require('mysql2');
```

- Loads **MySQL library**.
- Used to connect Node.js with MySQL database.

---

2. Create App

```
const app = express();
```

- Creates your **server app**.
- You will define routes using `app`.

```
app.use(express.json());
```

- Middleware to **read JSON data from request body**.
- Required for POST/PUT.
- Without this → `req.body` will be `undefined`.

□ 3. MySQL Connection

```
const db = mysql.createConnection({
```

- Creates a **database connection object**.

```
  host: 'localhost',
```

- Database is running on your local machine.

```
  user: 'root',
```

- MySQL username (default is `root`).

```
  password: '',
```

- Your MySQL password (empty in your case).

```
  database: 'express_db'
```

- Name of your database.

```
});
```

- Ends connection configuration.

□ 4. Connect to Database

```
db.connect(err => {
```

- Tries to connect to MySQL.
- `err` will contain error if connection fails.

```
  if (err) {
    console.log('Database connection failed:', err);
  }
```

- If error → print error in console.

```
  else {
    console.log('Connected to MySQL');
  }
```

- If success → show success message.

```
});
```

- End of connection function.

□ 5. GET All Users

```
app.get('/users', (req, res) => {
```

- Handles GET request at `/users`.

```
  db.query('SELECT * FROM users', (err, results) => {
```

- Runs SQL query to fetch all users.

```
    if (err) {
      res.status(500).json({ error: err.message });
    }
```

- If error → send HTTP 500 (server error).

```
    else {
      res.json(results);
    }
```

- If success → send all users as JSON.

```
  });
});
```

- End of query and route.

6. GET Single User

```
app.get('/users/:id', (req, res) => {
```

- `:id` is dynamic URL parameter.

```
  const id = req.params.id;
```

- Extract `id` from URL.

```
  db.query('SELECT * FROM users WHERE id = ?', [id], (err, results) => {
```

- `?` is placeholder → prevents SQL injection.
- `[id]` replaces `?`.

```
    else if (results.length === 0) {
```

- If no user found.

```
      res.status(404).json({ error: "User not found" });
```

- Send 404 error.

```
    else {
      res.json(results[0]);
    }
```

- Send first user (only one result).

---

7. CREATE User (POST)

```
app.post('/users', (req, res) => {
```

- Handles POST request (create data).

```
const { name, city, phone } = req.body;
```

- Extract data from JSON body.

```
const sql = 'INSERT INTO users (name, city, phone) VALUES (?, ?, ?)';
```

- SQL query with placeholders.

```
db.query(sql, [name, city, phone], (err, result) => {
```

- Inserts values into DB.

```
    id: result.insertId
```

- Returns ID of newly inserted record.

□ 8. UPDATE User

```
app.put('/users/:id', (req, res) => {
```

- Handles update request.

```
const id = req.params.id;
```

- Get ID from URL.

```
const { name, city, phone } = req.body;
```

- Get updated data.

```
const sql = 'UPDATE users SET name=?, city=?, phone=? WHERE id=?';
```

- SQL update query.

```
db.query(sql, [name, city, phone, id], (err, result) => {
```

- Executes update.

```
  res.json({ message: "User updated" });
```

- Sends success response.

□ 9. DELETE User

```
app.delete('/users/:id', (req, res) => {
```

- Handles delete request.

```
  const id = req.params.id;
```

- Get ID.

```
  db.query('DELETE FROM users WHERE id=?', [id], (err, result) => {
```

- Deletes user from database.

```
  res.json({ message: "User deleted" });
```

- Sends confirmation.

□ 10. Start Server

```
app.listen(3000, () => {
```

- Starts server on port **3000**.

```
  console.log('Server running on http://localhost:3000');
```

- Prints message in console.

```
});
```

- End server start.

Simple Summary (Very Important)

- `express()` → create server
- `app.get()` → read data
- `app.post()` → insert data
- `app.put()` → update data
- `app.delete()` → delete data
- `req.params` → URL data
- `req.body` → JSON data
- `db.query()` → run SQL
- `?` → safe values (prevents SQL injection)