NO - SQL – MongoDB

Q-1) What is MongoDB?

Ans) MongoDB is a popular NoSQL database that stores data in a flexible, JSON like format called documents. Unlike traditional SQL database that uses tables and rows, MongoDB uses collections and documents. Which makes it more flexible and scalable, especially for handing large amounts of unstructured or semi – structured data.

➔ **Why it is Used? :-** For its flexibility, document - Oriented data model, scalability, performance and developer – friendly features.

Q-2) What is Difference between mongodb and sql?

Ans) SQL :

SQL databases are ideal for structured data, complex queries, and scenarios where data integrity is critical.

Generally better for complex queries involving multiple joins, aggregate functions, and high data integrity.

Vertical Scalability: Traditionally scales vertically by adding more resources (CPU, RAM) to the existing server.

MongoDB:

MongoDB is better suited for handling unstructured data, fast development cycles, and scenarios requiring high scalability.

Often faster for simple queries and can handle large volumes of read/write operations efficiently.

Horizontal Scalability: Designed for horizontal scaling through sharding, which distributes data across multiple servers.

Q-3) Connect your Express application to a MongoDB database using Mongoose.

Ans) ☑ Step 1: Project Setup

```
mkdir express-mongoose-app
cd express-mongoose-app
npm init -y
npm install express mongoose body-parser
```

☑ Step 2: File Structure

```
express-mongoose-app/
├── models/
│   └── User.js
├── routes/
│   └── userRoutes.js
├── server.js
```

☑ Step 3: Create User.js model

🗀 models/User.js

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

const userSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true,
    unique: true
  }
});

module.exports = mongoose.model('User', userSchema);
```

☑ Step 4: Create userRoutes.js for POST route

🗀 routes/userRoutes.js

```
const express = require('express');
const router = express.Router();
const User = require('../models/User');

// POST /api/users
router.post('/users', async (req, res) => {
  const { name, email } = req.body;

  try {
    const user = new User({ name, email });
    await user.save();
    res.status(201).json({ message: 'User saved successfully!', user });
  } catch (error) {
    res.status(400).json({ error: error.message });
  }
});

module.exports = router;
```

☑ Step 5: Set up server.js

🗎 server.js

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const userRoutes = require('./routes/userRoutes');

const app = express();
const PORT = 3000;

// Middleware
app.use(bodyParser.json());

// Routes
app.use('/api', userRoutes);

// Connect to MongoDB
mongoose.connect('mongodb://127.0.0.1:27017/mydatabase', {
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(() => {
  console.log('Connected to MongoDB');
  // Start server only after DB connects
  app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
  });
}).catch((err) => {
  console.error('Failed to connect to MongoDB', err);
});
```

☑ Step 6: Test with Postman

- URL: http://localhost:3000/api/users
- Method: POST
- Body (JSON):

```
{
  "name": "Yash Thakar",
  "email": "yash@example.com"
}
```