# Database Notes — By Sagar Gautam

## 🔹 Analogy (Easy way to remember)

| Concept | SQL World | MongoDB World |
|---|---|---|
| Database | PostgreSQL / MySQL | MongoDB |
| Query Language | SQL | MQL (Mongo Query Language) |
| ORM/ODM | Sequelize / Prisma | Mongoose |

So,

💡 **MongoDB** = Database
💡 **Mongoose** = Helper library (connects your Node.js app with MongoDB in an organized way)

💡 **SQL** = Language
💡 **PostgreSQL** = Database software that *uses* SQL (and adds more power).

## 1. What is Database (DB)?

- A **database** is a structured collection of data that can be easily accessed, managed, and updated.
- Example: A school database stores student names, roll numbers, and grades.

**Types of Databases:**

1. Relational Database (SQL-based)
2. Non-Relational Database (NoSQL-based)

---

## 2. What is DBMS (Database Management System)?

- **DBMS** is software that helps you **store, manage, and retrieve** data from a database.
- It acts as a bridge between users and the database.

**Examples:**

- MySQL
- PostgreSQL
- Oracle
- MongoDB

**Functions of DBMS:**

- Data storage and retrieval
- Data security and backup
- Data consistency
- User access control

---

## 3. What is SQL (Structured Query Language)?

- **SQL** is a standard language used to interact with **relational databases**.
- It is used to create, read, update, and delete data (CRUD operations).

**Common SQL Commands:**

- `CREATE` – create database or table
- `INSERT` – add data
- `SELECT` – read data
- `UPDATE` – change data
- `DELETE` – remove data

**Example:**

```
CREATE TABLE students (
  id SERIAL PRIMARY KEY,
  name VARCHAR(50),
  age INT
);

INSERT INTO students (name, age) VALUES ('Sagar', 21);
SELECT * FROM students;
```

---

## 4. What is PostgreSQL?

- **PostgreSQL** is an **open-source, advanced relational database** that uses SQL.
- It supports both **SQL and NoSQL features** (like JSON, arrays, etc.).
- It is powerful, secure, and widely used for modern web applications.

**Key Features:**

- Open-source and free
- Supports advanced data types (JSONB, UUID, ARRAY)
- Strong data integrity and constraints
- High performance and scalability

**Example:**

```
CREATE TABLE employees (
  id SERIAL PRIMARY KEY,
  name TEXT,
  salary NUMERIC(10,2),
  joined_at TIMESTAMP DEFAULT NOW()
);
```

---

## 5. What is Prisma?

- **Prisma** is an **ORM (Object Relational Mapping) tool** for Node.js and TypeScript.
- It connects your **application code** to your **database** easily.

- It works with databases like PostgreSQL, MySQL, MongoDB, etc.

**Why use Prisma?**

- Auto-generates TypeScript types
- Easy database queries with JavaScript/TypeScript
- Supports migrations and schema management

**Example (Prisma query):**

```
const allUsers = await prisma.user.findMany();
```

**Analogy:**
Prisma = a helper that lets you talk to the database using code instead of SQL.

---

## 6. What is MongoDB?

- **MongoDB** is a **NoSQL database** that stores data in **JSON-like documents**.
- It does **not use tables** or **SQL queries** like relational databases.
- It is fast and flexible, mostly used in modern web apps.

**Example Document:**

```
{
  "name": "Sagar",
  "age": 21,
  "skills": ["React", "Node.js"]
}
```

**Key Features:**

- Document-based (No tables, no rows)
- Schema-less (flexible structure)
- High performance and easy scaling

## 7. What is Mongoose?

- **Mongoose** is an **ODM (Object Data Modeling)** library for **MongoDB**.
- It helps developers interact with MongoDB in a structured way using Node.js.

**Why use Mongoose?**

- Defines schemas and models for MongoDB collections
- Adds validation and data structure
- Simplifies MongoDB operations

**Example (Mongoose):**

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

const userSchema = new mongoose.Schema({
  name: String,
  age: Number
});

const User = mongoose.model('User', userSchema);

const newUser = new User({ name: "Sagar", age: 21 });
await newUser.save();
```

**Analogy:**
👉 MongoDB = database
👉 Mongoose = helper library that organizes how you interact with MongoDB.

## 8. SQL vs NoSQL (Quick Comparison)

| Feature | SQL (Relational DB) | NoSQL (Non-Relational DB) |
|---|---|---|
| Data Storage | Tables (rows & columns) | JSON-like documents |
| Schema | Fixed | Flexible |

| | | |
|---|---|---|
| Query Language | SQL | No fixed language |
| Examples | MySQL, PostgreSQL | MongoDB, Firebase |
| Best For | Structured data | Unstructured or dynamic data |

---

## 9. PostgreSQL vs MongoDB

| Feature | PostgreSQL | MongoDB |
|---|---|---|
| Type | Relational (SQL) | Non-relational (NoSQL) |
| Structure | Tables | Documents |
| Query Language | SQL | MQL (Mongo Query Language) |
| Relationships | Supported (JOINs) | Manual or embedded |
| Best Use | Banking, structured data apps | Dynamic, large-scale web apps |

---

## 10. Mongoose vs Prisma

| Feature | Prisma | Mongoose |
|---|---|---|
| Works with | SQL & MongoDB | MongoDB only |
| Type | ORM | ODM |
| Language | TypeScript / Node.js | Node.js |
| Schema | Prisma schema file | Mongoose schema (in code) |
| Best Use | Modern TypeScript apps | Simple MongoDB apps |

---

## 11. Summary

- **Database (DB)** → Data storage.
- **DBMS** → Software to manage database.
- **SQL** → Language to query relational data.

- **PostgreSQL** → Advanced relational DB using SQL.
- **Prisma** → ORM tool for connecting apps with databases easily.
- **MongoDB** → NoSQL database for flexible JSON-like data.
- **Mongoose** → ODM library for MongoDB that adds structure.

## 12.ORM and ODM

ORM = Connects your code with **SQL databases** easily.
ODM = Connects your code with **NoSQL databases (like MongoDB)** easily.

| Feature | ORM | ODM |
|---|---|---|
| Used with | SQL Databases | NoSQL Databases |
| Example DB | PostgreSQL, MySQL | MongoDB |
| Example Tool | Prisma, Sequelize | Mongoose |
| Converts | Objects ↔ Tables | Objects ↔ Documents |