# SQL vs NoSQL Databases

**SQL (Structured Query Language) Databases:**

- Store data in structured, table-based format (rows and columns).
- Follow a fixed schema, meaning the structure must be defined before inserting data.
- Use SQL for defining and manipulating data.
- Suitable for complex queries, transactions, and relationships between data.
- Examples: MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

**NoSQL (Not Only SQL) Databases:**

- Handle unstructured, semi-structured, or flexible schema data.
- Store data in various formats — key-value, document, column-family, or graph.
- Do not require a predefined schema; the structure can vary from record to record.
- Designed for horizontal scalability and high performance in distributed systems.
- Examples: MongoDB, DynamoDB, Cassandra, Redis.

# Understanding "Not Only SQL"

NoSQL doesn’t mean the absence of SQL.
It means databases are flexible enough to handle data beyond traditional relational constraints.
They support large-scale, real-time applications where data might not fit into rigid table structures.

# Example Format for NoSQL Data

A common format used is JSON (JavaScript Object Notation):

```
text
{
  "userId": 101,
```

```
  "name": "Kiran",
  "email": "kiran@example.com",
  "skills": ["Java", "Spring Boot", "AWS"],
  "profile": {
    "role": "Software Consultant",
    "experience": "2 years"
  }
}
```

- In this example:
  - Key-value format is visible (`"name": "Kiran"`).
  - Nested structure makes it flexible, unlike fixed SQL tables.

## Key-Value or Field-Value Format

- Data can be stored as pairs like:
  `Key: Value`
  Example:
  `Name: "Kiran"`
  `Age: 22`
  `City: "Hyderabad"`
- Simple and efficient for quick lookups and caching tasks.

## AWS DynamoDB Overview

- Fully managed NoSQL key-value and document database by AWS.
- No need to manage servers – it's serverless and auto-scales with your workload.
- Ensures low-latency, high-performance operations even at scale.
- Built-in security with IAM roles, encryption at rest and in transit.
- Key features:
  - **DAX (DynamoDB Accelerator):** In-memory caching for fast reads.
  - **Streams:** Capture real-time data changes.
  - **TTL (Time-To-Live):** Automatically delete expired items.
  - **Global Tables:** Multi-region replication for high availability.

## MongoDB

- MongoDB is a document-oriented database designed for scalability and ease of development.
- Stores data in flexible JSON-like documents instead of structured tables.
- Ideal for handling semi-structured data where the schema may vary between records.
- Supports horizontal scaling and distributed data storage.

**Example document:**

```
text
{
  "Name": "mango",
  "color": "red",
  "country": "india"
}
```

Each field holds data in key-value format, allowing greater flexibility compared to relational databases.

## Amazon DocumentDB (with MongoDB Compatibility)

- Enterprise-ready document database service built by AWS for managing and scaling JSON-based workloads.
- Fully managed with built-in monitoring, automated backups, and replication.
- Compatible with open-source MongoDB APIs (Apache 2.0 MongoDB 3.6 and 4.0).

**Key points:**

- Storage scales automatically up to 64 TB with zero downtime or performance impact.
- Handles millions of requests per second with up to 15 low-latency read replicas.
- Offers 99.99% availability and replicates six copies of data across three AWS Availability Zones.
- Automatically backs up data continuously to Amazon S3, supporting point-in-time recovery for the last 35 days.
- Fully integrates with other AWS services for security, monitoring, and scalability.

# Migration Flow (Self-managed to AWS)

**Self-managed MongoDB database**
→ AWS Database Migration Service
→ **Amazon DocumentDB (in the cloud)**

This migration enables moving on-premises or self-managed MongoDB workloads to a fully managed cloud setup without downtime or data loss.

# MongoDB Database and Collections

- In MongoDB, a **database** holds one or more **collections**.
- A **collection** is a group of **documents**, similar to how a table holds rows in SQL.
- Documents are stored in JSON-like format for flexible data structure.

# Using mongosh (MongoDB Shell)

### Step 1: Create or switch to a database

```
text
```

```
use myRecordDB
```

This command creates a new database named *myRecordDB* (if it doesn't exist) or switches to it.

### Step 2: Insert a document into a collection

```
text
```

```
db.myCollection.insertOne({name: 'Apple'})
```

- Creates a collection named *myCollection* (if it doesn't already exist).
- Inserts one document with a key-value pair.

### Step 3: Create an index on a field

```
text
```

```
db.myCollection2.createIndex({values: 'data'})
```

- Creates a new collection named *myCollection2*.
- Adds an index on the field *values* to improve search/query performance.

**Step 4: Insert more detailed data**

```
text
db.MyUsers.insertOne({"Name": "Apple", "color": "Red",
"country": "India"})
```

- Inserts a new document into the *MyUsers* collection.
- Each document can hold multiple key-value pairs.