MongoDB Practical Examples: MovieDB & SensorDB

This document demonstrates MongoDB usage with two real-world scenarios: a Movie database (MovieDB) and an IoT sensor database (SensorDB). It covers creating databases, inserting documents, performing CRUD operations, complex queries, sorting, and creating indexes.

1. MovieDB Example

MongoDB fits naturally for storing movie metadata because each movie can have different fields such as cast, genre, and ratings.

Create Database & Collection

```
use movieDB
db.createCollection("movies")
```

Insert Documents

```
db.movies.insertMany([
 { title: "Inception", year: 2010, genre: ["Sci-Fi","Thriller"], rating: 8.8, director: "Christopher Nolan" },
 { title: "Interstellar", year: 2014, genre: ["Sci-Fi","Drama"], rating: 8.6, director: "Christopher Nolan" },
 { title: "The Dark Knight", year: 2008, genre: ["Action","Drama"], rating: 9.0, director: "Christopher Nolan" },
 { title: "Parasite", year: 2019, genre: ["Thriller","Drama"], rating: 8.6, director: "Bong Joon-ho" }
])
```

CRUD Operations

Read:

```
db.movies.find({ director: "Christopher Nolan" })
```

Update:

```
 db.movies.updateOne({ title: "Interstellar" }, { $set: { rating: 8.7 } })
```

Delete:

```
db.movies.deleteOne({ title: "Parasite" })
```

Complex Queries

Movies released after 2010 with rating ≥ 8.7:

```
db.movies.find({ year: { $gt: 2010 }, rating: { $gte: 8.7 } })
```

Movies that are either Action OR have rating ≥ 9:

```
db.movies.find({ $or: [ { genre: "Action" }, { rating: { $gte: 9 } } ] })
```

Sorting

```
db.movies.find().sort({ rating: -1 })  // descending by rating
```

Create Index

```
db.movies.createIndex({ director: 1 })
```

2. SensorDB Example

MongoDB is ideal for time-series IoT sensor data, where readings vary in frequency and structure.

Create Database & Collection

```
use sensorDB
db.createCollection("readings")
```

Insert Documents

```
db.readings.insertMany([
  { deviceId: "sensorA", location: "Room1", temperature: 23.5, humidity: 45, timestamp: ISODate("2025-09-20T10:00:00Z") },
  { deviceId: "sensorB", location: "Room2", temperature: 24.8, humidity: 50, timestamp: ISODate("2025-09-20T10:05:00Z") },
  { deviceId: "sensorA", location: "Room1", temperature: 22.9, humidity: 47, timestamp: ISODate("2025-09-20T10:10:00Z") }
])
```

`"2025-09-20T10:00:00Z"` **= ISO 8601 format → Year-Month-Day** T **Time** Z **(UTC).**

the **Z** at the end is short for **Zulu time**, which is another name for **UTC (Coordinated Universal Time)**.

Why “Zulu”?

- In aviation and military timekeeping, each time zone is assigned a letter.
- The letter **Z** stands for the **zero** offset from the prime meridian (Greenwich).
- To avoid confusion when speaking, “Z” is pronounced **“Zulu”** (from the NATO phonetic alphabet).

What It Means

- **Z** → The time given is in **UTC**, with **no offset**.
- Equivalent to writing `+00:00` in ISO-8601 format.
- So `2025-09-20T10:00:00Z` means:

“September 20, 2025 at 10:00:00 **UTC** (Coordinated Universal Time).”

Example with Offsets

ISO String	**Meaning**
`2025-09-20T10:00:00Z`	10:00 UTC (Zulu time)
`2025-09-20T10:00:00+05:30`	10:00 **India time** (+5:30 hours ahead of UTC)
`2025-09-20T10:00:00-04:00`	10:00 Eastern Daylight Time (4 hours behind UTC)

So when MongoDB shows a date ending with **Z**, it’s storing and comparing the time in **absolute UTC**, making it timezone-independent.

CRUD Operations

Read:

```
db.readings.find({ deviceId: "sensorA" })
```

Update:

```
db.readings.updateOne({ deviceId: "sensorB" }, { $set: { humidity: 52 } })
```

Delete:

```
db.readings.deleteMany({ location: "Room2" })
```

Complex Queries

Find readings where temperature ≥ 23 AND humidity ≤ 47:

```
db.readings.find({ temperature: { $gte: 23 }, humidity: { $lte: 47 } })
```

Find readings in Room1 within a time range:

```
db.readings.find({ location: "Room1", timestamp: { $gte: ISODate("2025-09-20T10:00:00Z"), $lte: ISODate("2025-09-20T10:15:00Z") } })
```

Sorting

```
db.readings.find().sort({ timestamp: -1 })  // latest first
```

Create Index

```
db.readings.createIndex({ timestamp: -1 })
```