1. Introduction to NoSQL

- **NoSQL** stands for “Not Only SQL.”
- It refers to a broad class of **non-relational databases** that provide flexible schemas and are designed to handle **large-scale, distributed data**.
- Unlike traditional RDBMS, NoSQL databases don’t rely on fixed tables with rows and columns.

Key Characteristics

- Schema-less data model
- High scalability and performance
- Distributed and horizontally scalable
- Support for unstructured, semi-structured, and structured data
- Designed for big data and real-time web applications

2. Need for NoSQL

RDBMS Limitations	NoSQL Advantages
Fixed schema structure	Flexible schema
Difficult horizontal scaling	Easy horizontal scaling (Sharding)
Complex JOIN operations	Denormalized data for faster access
Limited handling of unstructured data	Can handle JSON, XML, multimedia, etc.
Costly hardware requirements	Commodity hardware can be used

Use Cases:

- Big data analytics

- Real-time web and mobile apps
- IoT data storage
- Social media platforms

3. Different NoSQL Data Models

Type	Description	Examples
Key-Value Store	Data stored as key-value pairs	Redis, Riak, DynamoDB
Document Store	Stores JSON-like documents	MongoDB, CouchDB
Column-Family Store	Stores data in columns instead of rows	Cassandra, HBase
Graph Database	Stores nodes and edges for relationships	Neo4j, OrientDB

4. Introduction to MongoDB

- **MongoDB** is a **document-oriented NoSQL database** developed by MongoDB Inc.
- Stores data in **BSON (Binary JSON)** format.
- Provides high performance, availability, and scalability.

Features

- Schema-less (flexible structure)
- Supports replication and sharding
- Rich query language
- Built-in aggregation framework
- Indexing for faster search

5. MongoDB Data Types

Data Type	Description
String	Text data
Integer	Whole numbers
Boolean	True/False values
Double	Floating-point numbers
Array	List of values
Object	Embedded document
ObjectId	Unique identifier
Date	Stores date/time values
Null	Null or missing value

6. Document Data Model

MongoDB stores data as **documents** in **collections**.

Example Document:

```
{
  "_id": 1,
  "name": "Sahinur",
  "age": 23,
  "skills": ["Python", "MongoDB"],
  "address": { "city": "Guwahati", "state": "Assam" }
}
```

7. CRUD Operations

1. Create (Insert)

```
db.students.insertOne({ name: "Rahul", age: 22 });
db.students.insertMany([{ name: "Asha" }, { name: "Vikram" }]);
```

2. Read (Find)

```
db.students.find();
db.students.find({ age: { $gt: 20 } });
```

3. Update

```
db.students.updateOne({ name: "Rahul" }, { $set: { age: 23 } });
db.students.updateMany({ age: { $lt: 18 } }, { $set: { status: "minor"
} });
```

4. Delete

```
db.students.deleteOne({ name: "Asha" });
db.students.deleteMany({ status: "inactive" });
```

8. MongoDB Query Language (MQL)

Common Query Operators:

- **Comparison:** `$eq`, `$ne`, `$gt`, `$lt`, `$gte`, `$lte`
- **Logical:** `$and`, `$or`, `$not`, `$nor`
- **Element:** `$exists`, `$type`
- **Array:** `$in`, `$all`, `$size`

Example:

```
db.students.find({ $and: [{ age: { $gt: 20 } }, { city: "Delhi" }] });
```

9. Indexing in MongoDB

- **Indexes** improve the performance of read queries.

- Default index is on `_id`.

Custom index:

```
 db.students.createIndex({ name: 1 }); // 1 for ascending, -1 for
descending
```

-
- Types of Indexes:
  - Single Field Index
  - Compound Index
  - Text Index
  - Geospatial Index

10. Aggregation Framework

Used for data processing and transformation.

Example Pipeline:

```
db.orders.aggregate([
  { $match: { status: "delivered" } },
  { $group: { _id: "$customerId", totalAmount: { $sum: "$amount" } }
},
  { $sort: { totalAmount: -1 } }
]);
```

Stages:

- `$match` → Filter documents
- `$group` → Group and aggregate data
- `$sort` → Sort output
- `$project` → Select specific fields

11. Sharding in MongoDB

- **Sharding** = Horizontal partitioning of data across multiple servers.
- Helps handle **large data sets** efficiently.
- Components:
  - **Shard** → Each shard holds a portion of data.
  - **Config Server** → Stores metadata about data distribution.
  - **Query Router (mongos)** → Routes client queries to the correct shard.

12. Join Operations in MongoDB

- MongoDB doesn’t support SQL-style joins directly.
- However, you can use **`$lookup`** in the aggregation pipeline.

Example:

```
db.orders.aggregate([
  {
    $lookup: {
      from: "customers",
      localField: "customerId",
      foreignField: "_id",
      as: "customerDetails"
    }
  }
]);
```

13. Pagination in MongoDB

- Used to limit and skip results in a query.

Example:

```
db.students.find().skip(10).limit(5);
```

- **skip(n)** → Skips the first *n* documents.
- **limit(m)** → Returns *m* documents after skipping.