1. What is a Document Database?

- A **non-relational** database that stores data as **structured documents**, usually in **JSON**.
- Designed to be **simple**, **flexible**, and **scalable**.

2. What is JSON?

- **JavaScript Object Notation** – lightweight data-interchange format.
- Built on:
    - Name/value pairs (object/dictionary/hash)
    - Ordered list of values (array/list/vector)
- Supported in all modern programming languages.

3. BSON – Binary JSON

- Binary-encoded version of JSON.
- Supports additional types (e.g., Date, Binary).
- Designed for efficient traversal, compact size, and encoding/decoding.

4. XML – A Predecessor to JSON

- Used for web content and formatting.
- Extensible tag structure (like HTML).
- Related tools:
    - **XPath** – retrieve elements
    - **XQuery** – query language for XML
    - **DTD** – defines allowed XML structure
    - **XSLT** – transforms XML into other formats

5. Why Document Databases?

- Solves **impedance mismatch** between object-oriented programming and relational models.
- Supports **inheritance** and **composition** natively.
- Self-describing document structure.
- Natural fit for apps using JSON/XML transport.

6. MongoDB – Overview

- Created in 2007 to overcome RDB limitations at web scale.
- "MongoDB" = **Humongous Database**.
- MongoDB Atlas (2016) is the cloud-based managed version.

7. MongoDB Structure

- Hierarchy:
    - **Database**
        - **Collection**
            - **Document**
- No predefined schema required.

- Each document in a collection can have different fields.

8. MongoDB vs Relational Databases (Text Version)

- **Database (RDBMS) → Database (MongoDB)**
- **Table/View → Collection**
- **Row → Document**
- **Column → Field**
- **Index → Index**
- **Join → Embedded Document**
- **Foreign Key → Reference**

9. MongoDB Features

- **Full CRUD support**
- **Indexing (primary + secondary)**
- **Replication with auto-failover**
- **Built-in load balancing**

10. MongoDB Editions

- **Atlas: Fully managed cloud service**
- **Enterprise: Subscription, self-hosted**
- **Community: Free, open-source**

11. MongoDB Tools

- **mongosh – Command-line shell**
- **MongoDB Compass – GUI**
- **PyMongo, Mongoose, etc. for code interaction**
- **Docker: Use port 27017 and provide admin user/pass**

12. Load Sample Dataset

- **Create `mflix` database in Compass**
- **Import JSON collections: `users`, `movies`, `comments`, `theaters`**
- **Dataset: https://www.dropbox.com/scl/fi/0yw17k5udo0yxu18eqsuj/mflix.zip?rlkey=zwdrzkqpz30yo48aynlye0cjx&dl=0**

13. MongoDB Queries in `mongosh`

```
// Select all from users
db.users.find()

// WHERE name = "Davos Seaworth"
db.users.find({"name": "Davos Seaworth"})

// WHERE rated IN ("PG", "PG-13")
db.movies.find({ rated: { $in: ["PG", "PG-13"] } })

// Movies from Mexico with IMDb rating >= 7
db.movies.find({
  countries: "Mexico",
  "imdb.rating": { $gte: 7 }
})

// Movies from 2010 that won >= 5 awards OR are Drama
db.movies.find({
  year: 2010,
  $or: [
    { "awards.wins": { $gte: 5 } },
    { genres: "Drama" }
  ]
})
```

```
// Count documents matching above
db.movies.countDocuments({
  year: 2010,
  $or: [
    { "awards.wins": { $gte: 5 } },
    { genres: "Drama" }
  ]
})

// Project only movie names, hide _id
db.movies.find({
  year: 2010,
  $or: [
    { "awards.wins": { $gte: 5 } },
    { genres: "Drama" }
  ]
}, { name: 1, _id: 0 })
```

14. PyMongo – Python and MongoDB

```
# Connect to MongoDB
from pymongo import MongoClient
client = MongoClient('mongodb://user_name:pw@localhost:27017')

# Select database and collection
db = client['ds4300']
collection = db['myCollection']

# Insert a single document
post = {
  "author": "Mark",
  "text": "MongoDB is Cool!",
  "tags": ["mongodb", "python"]
}
post_id = collection.insert_one(post).inserted_id
print(post_id)

# Count documents
db.collection.count_documents({})
```