Main Themes and Important Ideas/Facts:

1. **Document Databases as a Non-Relational Alternative**:
    - **Document databases** are a type of non-relational database designed to store data as structured documents, often in **JSON** format.
    - They are **simple, flexible, and scalable**, addressing the "impedance mismatch problem" between object-oriented programming and relational databases.
    - Unlike relational databases, document databases avoid rigid table schemas, offering a more natural representation of objects.
2. **JSON as the Core Data Format**:
    - **JSON (JavaScript Object Notation)** is a widely-used data interchange format:
        - **Lightweight**, **easy to read and write** for humans, and **easy to parse** for machines.
        - Built on two primary structures:
            - **Objects** (key-value pairs)
            - **Arrays** (ordered lists of values).
    - JSON's flexibility and simplicity have made it the preferred data format over older formats like **XML**.
3. **BSON (Binary JSON)**:
    - MongoDB uses **BSON (Binary JSON)**, a **binary-encoded serialization of JSON** documents. It provides several advantages:
        - Supports extended types not available in basic JSON (e.g., Date, BinaryData).
        - Efficient in terms of both space and traversal, designed for quick encoding/decoding.
4. **Motivation Behind Document Databases**:
    - Document databases like MongoDB emerged to solve challenges faced by relational databases when handling complex object-oriented systems.
    - MongoDB’s creation in 2007 stemmed from the limitations of relational databases in handling high-volume data (e.g., serving over 400,000 ads per second).
5. **MongoDB Structure and Key Concepts**:
    - **Databases**: Logical groupings of collections.
    - **Collections**: Analogous to tables in relational databases but without fixed schemas.
    - **Documents**: Stored as BSON, can vary in structure within the same collection.
    - MongoDB doesn’t require a predefined schema for documents, offering flexibility where each document in a collection can have a unique structure.

- **Comparison**:

RDBMS	MongoDB
Database	Database
Table/View	Collection

Row	Document
Column	Field
Index	Index
Join	Embedded Document
Foreign Key	Reference

6. **MongoDB Features**:
    - **Rich Query Support**: Supports CRUD operations (Create, Read, Update, Delete).
    - **Indexing**: Primary and secondary indices for efficient querying.
    - **Replication**: Automatic failover via replica sets for high availability.
    - **Horizontal Scaling**: Load balancing and sharding to distribute data across servers.
7. **MongoDB Versions**:
    - **MongoDB Atlas**: Fully managed, cloud-based MongoDB service (DBaaS).
    - **MongoDB Enterprise**: Subscription-based, self-managed enterprise version.
    - **MongoDB Community**: Free-to-use, source-available self-managed version.
8. **Interacting with MongoDB**:
    - **mongosh**: CLI tool to interact with MongoDB instances.
    - **MongoDB Compass**: GUI for working with MongoDB databases.
    - **Language Drivers**: Available for various languages, e.g., **PyMongo** for Python.
9. **Basic MongoDB Operations (mongosh & PyMongo)**:
    - Examples of **mongosh** commands:
        - `db.users.find()`: Retrieve all documents in a collection.
        - `db.users.find({"name": "Davos Seaworth"})`: Filter documents.
        - `db.movies.countDocuments({})`: Count documents in a collection.
    - Examples of **PyMongo** usage:
        - Connecting to MongoDB: `client = MongoClient()`
        - Inserting documents: `collection.insert_one(post)`
        - Counting documents: `collection.count_documents({})`