1. What is a database? Explain with an example on why should we need a database.

A **database** is an organized collection of data that is stored and accessed electronically. It allows data to be efficiently managed, retrieved, and updated. For example, consider an online retail store that needs to keep track of products, customers, orders, and inventory. A database helps in organizing this information, avoiding redundancy, maintaining data integrity, and providing quick access to data, which is crucial for the store's operations.

2. Write a short note on the file-based storage system. Explain the major challenges of a file-based storage system.

A **file-based storage system** refers to the traditional method of storing data in files within a file system. Each file stores data related to a particular application or process, and the files are typically managed manually or by the application itself.

Challenges of a File-Based Storage System:

- **Data Redundancy**: The same data might be duplicated across multiple files, leading to inconsistency and increased storage requirements.
- **Data Isolation**: Data is often scattered across various files, making it difficult to access and combine data from different sources.
- **Data Integrity**: Ensuring the accuracy and consistency of data across files can be challenging, especially when multiple copies of data exist.
- **Concurrent Access**: Managing simultaneous access to data by multiple users is complex and can lead to data corruption.
- **Scalability**: As data volume grows, managing and retrieving data efficiently becomes increasingly difficult.

3. What is DBMS? What was the need for DBMS?

A **Database Management System (DBMS)** is software that interacts with the user, applications, and the database itself to capture and analyze data. A DBMS allows users to create, read, update, and delete data in a structured and efficient manner.

Need for DBMS:

- **Data Management**: DBMS provides a centralized system for managing large volumes of data, ensuring consistency, accuracy, and security.
- **Data Security**: DBMS offers mechanisms to protect data from unauthorized access.
- **Data Integrity**: It enforces rules to maintain the accuracy and consistency of data.
- **Concurrency Control**: DBMS manages concurrent access to data, ensuring that multiple users can work with the data without conflicts.
- **Data Retrieval**: DBMS allows for complex queries to retrieve specific data efficiently.

4. Explain 5 challenges of a file-based storage system that were tackled by DBMS.

- **Data Redundancy and Inconsistency**: DBMS reduces redundancy by storing data in a centralized manner, ensuring consistency.
- **Data Isolation**: DBMS integrates data, making it accessible from a single system rather than scattered across files.
- **Data Integrity Issues**: DBMS enforces integrity constraints to ensure that data remains accurate and consistent.
- **Difficulty in Data Access**: DBMS provides advanced query languages like SQL, making data retrieval more efficient.
- **Concurrent Access Anomalies**: DBMS includes mechanisms to manage multiple users accessing the data simultaneously without causing conflicts.

5. List out the different types of classification in DBMS and explain.

DBMS Classifications:

1. **Based on Data Models**:

- **Hierarchical DBMS**: Organizes data in a tree-like structure. Example: IBM's Information Management System (IMS).
- **Network DBMS**: Uses a graph structure to allow many-to-many relationships. Example: Integrated Data Store (IDS).
- **Relational DBMS (RDBMS)**: Stores data in tables (relations) and uses SQL for querying. Example: MySQL, Oracle.
- **Object-Oriented DBMS**: Stores data as objects, similar to object-oriented programming. Example: ObjectDB.
- **Document-Based DBMS**: Stores data in document format (e.g., JSON, XML). Example: MongoDB.

2. **Based on Number of Users**:
   - **Single-User DBMS**: Supports one user at a time.
   - **Multi-User DBMS**: Supports multiple users simultaneously.
3. **Based on Database Distribution**:
   - **Centralized DBMS**: Data is stored in a single location.
   - **Distributed DBMS**: Data is distributed across multiple locations.
4. **Based on Usage**:
   - **OLTP (Online Transaction Processing)**: Manages transaction-oriented applications. Example: Banking systems.
   - **OLAP (Online Analytical Processing)**: Supports complex queries for data analysis. Example: Data warehousing systems.

6. What is the significance of Data Modelling and explain the types of Data Modelling.

Data Modelling is the process of creating a data model, which is a conceptual representation of the data structures and the relationships between them. It serves as a blueprint for designing and creating a database.

Types of Data Modelling:

1. **Conceptual Data Model**: High-level, abstract model that outlines the structure of the data, focusing on entities, attributes, and relationships.
2. **Logical Data Model**: More detailed than the conceptual model, this model defines the structure of the data elements and their relationships but is still independent of the physical implementation.
3. **Physical Data Model**: Represents the actual implementation of the data model in a specific DBMS, including tables, columns, indexes, and constraints.

7. Explain 3-schema architecture along with its advantages.

3-Schema Architecture refers to a framework for database systems that separates the database into three levels:

1. **Internal Schema**: The lowest level, which deals with the physical storage of data. It defines how data is stored and organized on physical storage devices.
2. **Conceptual Schema**: The middle level, which defines the logical structure of the entire database for a community of users. It describes what data is stored and the relationships among them.
3. **External Schema**: The highest level, which deals with individual user views. It defines how different users interact with the database, providing customized views of the data for various user needs.

Advantages:

- **Data Abstraction**: Separates the physical storage from logical data structure and user views, making it easier to manage changes.

- **Multiple Views**: Allows different user groups to have different views of the same data, tailored to their specific needs.
- **Data Independence**: Changes in the internal schema do not affect the conceptual schema or external schema, and vice versa, providing flexibility in managing data.