CHAPTER 1: INTRODUCTION TO DATABASE

Lesson 1: What is a Database?

Objective:

To introduce students to the concept of databases, their importance, and real-world applications, laying the foundation for understanding data management and database systems.

---

1. Definition of a Database

A **database** is an organized collection of structured information, or data, typically stored electronically in a computer system. Databases are controlled by a Database Management System (DBMS), which allows users to create, update, and manage data efficiently.

Key Points:

- Databases are structured to enable easy access, retrieval, and management of data.
- They are essential in storing large amounts of information, organizing it in a way that makes it easy to retrieve and analyze.

Example: Consider a library that needs to keep track of books. A database would allow the library to store details such as book titles, authors, publication years, genres, and borrower information, all in one organized system.

---

2. Importance of Databases

Databases are crucial for modern-day organizations and systems. They allow for:

- **Efficient Data Management**: Databases enable systematic storage, making data easy to retrieve, update, and delete as needed.
- **Enhanced Security**: Databases provide ways to protect data through authentication and authorization protocols.
- **Scalability**: They allow organizations to handle increasing amounts of data as they grow.
- **Data Integrity**: Databases ensure the consistency and accuracy of data, minimizing redundancy and data anomalies.

Example: In a hospital, databases ensure that patient records are kept accurately, readily available to healthcare providers, and protected under privacy regulations.

---

3. Real-World Applications of Databases

Databases are used across various fields, including:

- **E-Commerce**: Storing product information, customer data, orders, and payment details.
- **Education**: Managing student records, enrollment, grades, and attendance.
- **Banking**: Recording customer transactions, balances, and personal data.
- **Healthcare**: Storing patient information, medical records, appointment schedules, and treatment history.

Examples:

- **Online Retailers** like Amazon use databases to store product inventory, track orders, and manage customer accounts.
- **Universities** use databases to manage student information, course enrollments, and faculty data.

Lesson Recap

- **A database** is an organized collection of data managed by a DBMS.
- **Databases are important** for secure, efficient data management and support various applications across industries.

In-Class Activity:

Ask students to brainstorm examples of databases they interact with daily (e.g., social media accounts, mobile banking apps) and discuss the kind of data each application might store.

Lesson 2: Components of a Database System

Objective:

To help students understand the essential components of a database system, including hardware, software, data, procedures, and users. This will provide a foundation for understanding how databases operate and how different components interact within a database environment.

---

1. Overview of Database System Components

A database system is not just the database itself but a combination of different components working together to store, manage, and retrieve data efficiently. The five main components of a database system are:

1. **Hardware**
2. **Software**

3. **Data**
4. **Procedures**
5. **Users**

---

2. Components in Detail

1. Hardware

- **Definition**: Hardware refers to the physical devices required to store and process data. This includes servers, storage devices, network devices, and any equipment that supports data handling and processing.
- **Role**: Provides the infrastructure for data storage, retrieval, and processing. Hardware forms the physical backbone of a database system.
- **Example**: In a corporate environment, servers are often used to host databases that employees can access remotely.

2. Software

- **Definition**: Software in a database system consists of the DBMS and other applications that interact with the database.
- **Role**: Manages the storage, modification, and retrieval of data. The DBMS provides tools for creating, managing, and interacting with the data, and may include additional software for data backup, security, and reporting.
- **Example**: Popular DBMS software includes MySQL, Oracle Database, Microsoft SQL Server, and PostgreSQL.

3. Data

- **Definition**: Data is the information that is stored, organized, and managed within the database. It is the core component of a database system.
- **Role**: Acts as the central resource, with other components existing to store, protect, and manage this data.

- **Example**: In a university database, data could include student names, courses, grades, and attendance records.

4. Procedures

- **Definition**: Procedures are the rules, guidelines, or instructions governing the use and operation of the database system. They are essential for ensuring that the database operates consistently and securely.
- **Role**: Define how the database should be accessed, modified, and managed. Procedures are often implemented to maintain security, efficiency, and data integrity.
- **Example**: A university might have procedures for accessing student records, allowing only authorized personnel like registrars and instructors to access specific types of data.

5. Users

- **Definition**: Users are the individuals who interact with the database. They can be categorized into various roles depending on their level of access and responsibilities.
- **Types of Users**:
    - **End Users**: Individuals who access and interact with data, typically without administrative privileges. Examples include customers viewing products on an e-commerce site.
    - **Database Administrators (DBAs)**: Individuals responsible for managing the database, including configuration, maintenance, and security.
    - **Developers**: They create applications that interact with the database, designing systems to efficiently access and manipulate data.
- **Role**: Users play a crucial role in the daily operation of the database, interacting with it to retrieve, update, and manage data as required by their specific roles.

---

3. How the Components Interact

Each component of the database system interacts with others to ensure efficient data handling and management:

- **Software** relies on **hardware** to store and retrieve data, and it enforces **procedures** to control access and modification.
- **Data** is accessed by **users** through **software**, following established **procedures**.
- **Database administrators** and **developers** are responsible for implementing and managing the software, ensuring smooth interaction among all components.

Lesson Recap:

- **Hardware** forms the physical backbone.
- **Software** manages data and interacts with hardware.
- **Data** is the primary asset stored, managed, and retrieved.
- **Procedures** guide the operations, ensuring security and consistency.
- **Users** interact with the database in different roles, such as end users, DBAs, and developers.

In-Class Activity:

Ask students to identify the components of a database system for a real-world scenario, such as a hospital management system. For example, identify the types of hardware, software, data, procedures, and users that might be involved in managing patient records.

Lesson 3: Types of Data

Objective:

To introduce students to the different types of data that can be stored and managed in a database, including structured and unstructured data, as well as various data formats. Understanding data types is essential for designing databases that can handle diverse information efficiently.

---

1. Overview of Data Types

Data is the core of any database system, and it exists in various formats and structures. The way data is structured impacts how it can be stored, retrieved, and analyzed. The two main categories of data are:

1. **Structured Data**
2. **Unstructured Data**

---

2. Structured Data

Definition:

- Structured data refers to data that is organized into a defined format or schema, usually in tables with rows and columns. This organization makes it easy to analyze, sort, and filter the data.

Characteristics:

- Organized into a format that is easily understood by database systems (like relational databases).
- Includes fields (columns) and records (rows) with specific data types.
- Highly searchable, sortable, and easy to analyze due to its consistent format.

Examples:

- **Customer data** in an e-commerce database (columns like Customer ID, Name, Email, and Order History).
- **Employee records** in a company's HR database (fields like Employee ID, Name, Position, and Salary).

Real-World Application:

- In a bank, customer data (account numbers, names, balances, and transactions) is stored as structured data, allowing bank employees to quickly retrieve and manage individual customer records.

---

3. Unstructured Data

Definition:

- Unstructured data lacks a pre-defined format or organization, making it difficult to store and analyze in traditional relational databases. It often consists of text, images, videos, or audio files.

Characteristics:

- Does not follow a specific format or organization.
- Requires specialized storage and processing solutions, such as NoSQL databases, for effective management.
- Often used for analyzing large volumes of diverse information, such as customer reviews, multimedia content, or social media posts.

Examples:

- **Emails**: Each email might have text, images, and attachments, with no standard structure.

- **Social Media Posts**: Contains text, hashtags, images, videos, and other elements without a uniform format.
- **Medical Images**: X-rays or MRIs that are stored as image files with metadata, but without structured fields.

Real-World Application:

- Social media companies, like Facebook or Twitter, store massive amounts of unstructured data to analyze user activity, preferences, and trends for marketing and content recommendation.

---

4. Semi-Structured Data

Definition:

- Semi-structured data is a hybrid form that combines elements of structured and unstructured data. It does not have a rigid schema but contains tags or markers to separate data elements, making it easier to process than purely unstructured data.

Characteristics:

- Contains organizational tags, enabling easier parsing and analysis than fully unstructured data.
- Often stored in flexible formats that can adapt to changes, such as JSON or XML.

Examples:

- **JSON and XML Files**: Commonly used in web applications to store and exchange data in a flexible format that is human-readable and machine-readable.
- **Email Headers**: Email headers include structured information like sender, recipient, and timestamp, while the email content may be unstructured.

Real-World Application:

- APIs (Application Programming Interfaces) often use JSON or XML to communicate data between applications. For instance, an e-commerce site might use an API to send product information in JSON format to display on a partner website.

---

5. Types of Data Formats

In addition to structure, data can exist in various formats that impact how it is stored, processed, and analyzed. Key data formats include:

- **Text**: Includes plain text, XML, HTML, JSON.
- **Numeric**: Whole numbers (integers) and floating-point numbers (decimals).
- **Binary**: Images, videos, and other media files stored as binary data.
- **Date and Time**: Specific data formats for dates and timestamps, essential for sorting, filtering, and calculating time-related data.

Example of Usage:

- **Images**: Stored as binary files (JPEG, PNG), often requiring specific tools or applications to interpret.
- **Dates**: A retail store database may use date formats to track order dates, shipping dates, and return periods.

Lesson Recap

- **Structured Data** is organized and easy to analyze (e.g., tables in relational databases).
- **Unstructured Data** lacks a defined structure and often includes multimedia files (e.g., social media posts).

- **Semi-Structured Data** has organizational tags but no strict schema (e.g., JSON files).
- **Different data formats** (text, numeric, binary, date) cater to specific data storage and processing needs.

In-Class Activity:

Have students categorize different types of data they encounter daily, such as personal files on their devices, social media posts, and school records. Discuss how each type of data might be stored in a database.

Homework:

Students can research examples of companies that manage large amounts of structured and unstructured data (e.g., Google, Amazon, Netflix) and briefly describe how these companies might handle different types of data.

Lesson 4: Database Systems vs File Systems

Objective:

To help students understand the differences between database systems and traditional file systems, exploring the advantages and limitations of each. This lesson will highlight why database systems are generally preferred for handling complex data requirements in modern applications.

---

1. Overview of Database Systems and File Systems

File Systems and **Database Management Systems (DBMS)** are both methods of storing, retrieving, and managing data. However, they differ significantly in terms of functionality, scalability, and efficiency.

- **File System**: A traditional method of storing and organizing files on a storage device, such as a hard drive or a USB stick. File systems provide a basic way of storing data in individual files, which can be accessed directly by users or applications.
- **Database System (DBMS)**: A software system that uses a structured approach to store and manage data. It includes tools for data manipulation, storage, retrieval, and management, typically using tables, indexes, and other constructs to optimize data handling.

---

2. Characteristics of File Systems

Definition:

- A file system is a method used by operating systems to control how data is stored and retrieved on a storage device. Each file is stored as a separate entity in directories, and users can organize files into a hierarchy of folders and subfolders.

Features:

- Simple and straightforward for storing and accessing individual files.
- Often limited in data organization and retrieval capabilities.
- Typically lacks tools for advanced data management, such as querying, indexing, and transaction handling.

Advantages:

- **Simplicity**: Easy to use for basic file storage and organization.

- **Low Overhead**: Requires minimal processing power and resources, as there is no need for complex data management tools.
- **Direct Access**: Files can be accessed directly, which may be beneficial for simple applications.

Limitations:

- **Redundancy**: Data may be duplicated across multiple files, increasing storage requirements.
- **Limited Security**: Often lacks robust security and user management features.
- **Data Inconsistency**: Changes in one file might not reflect in related files, leading to inconsistency.
- **Scalability Issues**: Not ideal for handling large volumes of data or complex data structures.

Example:

- Personal computers use file systems to manage individual documents, images, and videos stored in folders on the hard drive.

---

3. Characteristics of Database Systems (DBMS)

Definition:

- A Database Management System (DBMS) is a software system that provides a structured approach to storing, managing, and retrieving data, typically through a relational model with tables, fields, and records.

Features:

- Supports complex queries and data manipulation using SQL (Structured Query Language).

- Ensures data consistency, integrity, and security through features like ACID (Atomicity, Consistency, Isolation, Durability).
- Allows for multi-user access and concurrent data handling without compromising data integrity.

Advantages:

- **Reduced Redundancy**: DBMSs are designed to minimize data duplication through normalization.
- **Enhanced Data Security**: Allows administrators to control user access, ensuring sensitive data remains protected.
- **Data Integrity**: Built-in rules and constraints maintain data accuracy and consistency.
- **Scalability**: Database systems are scalable and can handle large volumes of data efficiently.
- **Complex Queries**: Supports advanced data retrieval and manipulation, making it ideal for complex applications.

Limitations:

- **Complexity**: Requires a certain level of technical knowledge to set up and manage.
- **Resource Intensive**: Database systems require significant processing power and memory, especially for large datasets.
- **Cost**: DBMS software and hardware infrastructure can be costly, particularly for large-scale deployments.

Example:

- An online retail store might use a database system to manage product details, customer information, and order history, enabling efficient data retrieval for a smooth customer experience.

4. Key Differences Between File Systems and Database Systems

Feature	File System	Database System (DBMS)
Data Redundancy	High redundancy, leading to inefficiency	Low redundancy due to normalization
Data Integrity	Limited control over data consistency	Strong integrity controls (ACID properties)
Security	Basic access control, low security	Advanced user management and security features
Data Retrieval	Basic file access	Advanced querying capabilities with SQL
Concurrent Access	Limited support	Supports multi-user access and transactions
Scalability	Limited scalability	Highly scalable for large datasets
Cost	Low	Moderate to high

5. Why Database Systems Are Preferred

Due to the limitations of file systems, particularly in handling complex, large-scale, and multi-user environments, database systems are generally preferred. They provide the tools needed to maintain data consistency, security, and integrity, making them suitable for use cases where data reliability is critical.

Examples of Applications Requiring Database Systems:

- Banking systems: where data accuracy, security, and multi-user access are essential.
- E-commerce websites: requiring fast data retrieval and high scalability.
- Healthcare management: where data security and consistency across multiple users are crucial.

Lesson Recap:

- **File Systems** offer basic file storage with limited security, data redundancy, and no support for complex queries or multi-user access.
- **Database Systems (DBMS)** provide structured, secure, and scalable data management with tools for complex queries, data integrity, and multi-user access.
- **DBMS** is typically preferred for complex, large-scale applications due to its advanced features for managing data.

In-Class Activity:

Ask students to compare the pros and cons of using a file system vs. a database system for a specific scenario, like a small business keeping track of customer orders. Discuss which system would be more effective as the business grows.

Homework:

Have students research and summarize a real-world scenario where switching from a file system to a database system improved an organization’s data management, detailing the benefits achieved with the new database system.

Lesson 5: Basic Terminology

Objective:

To introduce students to foundational database terminology, including tables, fields, records, queries, and relationships. Understanding these terms is essential for building and interacting with databases.

---

1. Key Terms and Definitions

In a relational database, data is organized in a structured way to facilitate easy storage, retrieval, and management. Let’s explore some fundamental terms that form the foundation of database design and usage.

---

2. Tables

Definition:

- A table is a structured collection of related data entries in rows and columns. Each table in a database typically represents a specific entity, such as "Students," "Employees," or "Orders."

Characteristics:

- Tables are made up of **columns** (fields) and **rows** (records).
- Each row represents a single record, and each column represents an attribute of the data.

Example:

- In a library database, you might have a **Books** table that includes columns for Book ID, Title, Author, Genre, and Year Published.

Book ID	Title	Author	Genre	Year Published
1	To Kill a Mockingbird	Harper Lee	Fiction	1960
2	1984	George Orwell	Dystopian	1949

---

3. Fields (Columns)

Definition:

- A field, also known as a column, is a single attribute or data type that each record in the table shares. Each field stores a specific type of data, like text, numbers, or dates.

Characteristics:

- Fields define the structure of the table and provide specific types of information about the data.
- Every field in a table has a unique name and data type (e.g., integer, text, date).

Example:

- In the **Books** table, the fields are **Book ID**, **Title**, **Author**, **Genre**, and **Year Published**. Each field represents a specific attribute of a book.

---

4. Records (Rows)

Definition:

- A record, also known as a row, represents a single, unique entry in a table. Each record contains data for all the fields in the table, representing one instance of the entity the table describes.

Characteristics:

- Each record includes a unique set of data values across all fields.
- In most tables, a **primary key** field uniquely identifies each record to prevent duplicates.

Example:

- In the **Books** table, each row represents a different book, with data filled out for each field.

Book ID	Title	Author	Genre	Year Published
1	To Kill a Mockingbird	Harper Lee	Fiction	1960

---

5. Queries

Definition:

- A query is a request for information from the database, used to retrieve, update, insert, or delete data. Queries are written in **Structured Query Language (SQL)**, which allows users to communicate with the database.

Types of Queries:

- **Select Query**: Retrieves specific data based on certain criteria (e.g., finding all books by a particular author).
- **Update Query**: Modifies existing data in the database.
- **Insert Query**: Adds new data to a table.
- **Delete Query**: Removes data from a table.

Example:

- To find all books published after 2000 in the **Books** table:

```
SELECT * FROM Books WHERE Year_Published > 2000;
```

---

6. Relationships

Definition:

- A relationship is a connection between tables in a database that defines how data in one table relates to data in another. Relationships help maintain data integrity and enable complex queries that retrieve information across multiple tables.

Types of Relationships:

- **One-to-One (1:1)**: A single record in one table is associated with a single record in another table. Example: Each **Person** has one **Passport**.
- **One-to-Many (1**

  ): A single record in one table is associated with multiple records in another table. Example: A single **Author** can write multiple **Books**.

- **Many-to-Many (M**

  ): Multiple records in one table can relate to multiple records in another table. Example: **Students** can enroll in multiple **Courses**, and each **Course** can have multiple **Students**.

Example of a One-to-Many Relationship:

- In a database with **Authors** and **Books** tables, an **Author ID** in the **Books** table could link to an **Author** record in the **Authors** table. This allows you to see which books a particular author has written.

Author ID	Author Name
1	Harper Lee
2	George Orwell

Book ID	Title	Author ID
1	To Kill a Mockingbird	1
2	1984	2

Lesson Recap

- **Tables** store data in a structured format with fields and records.
- **Fields** (columns) represent specific attributes or characteristics of the data.
- **Records** (rows) are individual data entries in a table.
- **Queries** allow users to retrieve and manipulate data.
- **Relationships** define connections between tables to represent real-world associations.

In-Class Activity:

Ask students to create a small table on paper for a **Student** database, including fields like Student ID, Name, Age, and Major. Have them discuss possible queries they could use to retrieve information, such as finding students in a specific major.

Homework:

Have students write a list of entities they would include in a **Library Database** (e.g., Books, Authors, Members) and outline possible relationships between these entities (e.g., one-to-many between **Authors** and **Books**).

Capstone Project: Creating a Simple Database to Manage Student Information in MS Access

Objective:

The capstone project aims to help students apply what they've learned about databases by creating a simple student information management system in **Microsoft Access**. This project will guide students in setting up a basic relational database, creating tables, setting up fields, and inserting data.

Project Outline:

1. **Project Setup**
    - Students will use **Microsoft Access** to create a database called **Student Management System**.
2. **Creating Tables**
    - Students will design and create the following tables:
        - **Students**: Stores personal details about each student.
        - **Courses**: Stores information about courses offered.
        - **Enrollments**: Tracks which students are enrolled in which courses, representing a many-to-many relationship between **Students** and **Courses**.
        - **Instructors** (Optional): Stores information about instructors for each course.
3. **Setting Up Fields**
    - Each table will include the following fields:

TAR JOEL **tarjoeltar10@gmail.com** **+237679303247**

Students Table:

Field Name	Data Type	Description
StudentID	AutoNumber (Primary Key)	Unique ID for each student
FirstName	Text	Student's first name
LastName	Text	Student's last name
DateOfBirth	Date/Time	Student's date of birth
Gender	Text	Student's gender
Major	Text	Student's major or area of study
EnrollmentDate	Date/Time	Date of student's enrollment

Courses Table:

Field Name	Data Type	Description
CourseID	AutoNumber (Primary Key)	Unique ID for each course
CourseName	Text	Name of the course
Department	Text	Department offering the course
Credits	Number	Credit value of the course

Enrollments Table:

Field Name	Data Type	Description
EnrollmentID	AutoNumber (Primary Key)	Unique ID for each enrollment
StudentID	Number	Foreign key linking to Students table
CourseID	Number	Foreign key linking to Courses table
EnrollmentDate	Date/Time	Date when the student enrolled

Instructors Table (Optional):

Field Name	Data Type	Description
InstructorID	AutoNumber (Primary Key)	Unique ID for each instructor
FirstName	Text	Instructor's first name

LastName	Text	Instructor's last name
Department	Text	Department the instructor belongs to

4. **Setting Up Relationships**
    - Establish relationships between the tables:
        - One-to-Many between **Students** and **Enrollments** (one student can enroll in multiple courses).
        - One-to-Many between **Courses** and **Enrollments** (one course can have multiple students enrolled).
        - (Optional) One-to-Many between **Courses** and **Instructors** if the **Instructors** table is used.
5. **Inserting Data**
    - Populate each table with sample data:
        - **Students**: Add at least 5 students with different majors and enrollment dates.
        - **Courses**: Add at least 3 courses from various departments with different credit values.
        - **Enrollments**: Register each student for at least two courses.
        - **Instructors** (Optional): Add instructors for each course.
6. **Creating Queries**
    - Students will create several queries to retrieve data, such as:
        - **List of all students enrolled in a specific course**.
        - **List of all courses taken by a specific student**.
        - **Students with enrollment dates after a specific date**.
        - **Count of students in each course**.
7. **Generating Reports**
    - Students will design a simple report showing **student enrollment details**, such as student name, course name, and enrollment date.

Project Requirements

- **Database Structure**: All tables, fields, relationships, and data types should match the outlined specifications.
- **Data Entry**: Sample data should be entered for each table to demonstrate functionality.
- **Queries**: Create and run at least three meaningful queries.
- **Report**: Generate a report that displays student enrollments.

Project Submission:

Students will submit:

1. The **MS Access database file** containing all tables, data, queries, and the report.
2. A **short write-up** (1-2 pages) explaining the steps taken, challenges faced, and how the queries were designed.

Assessment Criteria:

Criteria	**Points**
Database Structure (Tables, Fields)	20
Data Entry (Sample Data)	20
Relationships Setup	20
Queries Created	20
Report Design	20
Total	**100**

This project provides practical experience in creating, organizing, and managing data, as well as designing queries and generating reports. Through this exercise, students gain a hands-on understanding of database concepts and how they are applied in real-world scenarios.

PAST QUESTIONS

Section 1: Multiple-Choice Questions (MCQs)

1. Which of the following best defines a database?
    - a) A collection of files stored in a folder
    - b) An organized collection of related data
    - c) A software program for word processing
    - d) A device used for storing photos
    - **Answer**: b
2. What is the primary purpose of a Database Management System (DBMS)?
    - a) To manage hardware devices
    - b) To store and manage data efficiently
    - c) To manage network connections
    - d) To manage file storage in folders
    - **Answer**: b
3. Which type of data is unstructured?
    - a) An Excel spreadsheet
    - b) A JSON file
    - c) A text document with no specific format
    - d) A relational database table
    - **Answer**: c
4. Which component of a database system is responsible for defining how data is stored, retrieved, and managed?
    - a) Hardware
    - b) Software
    - c) Data
    - d) Procedures
    - **Answer**: b
5. In a relational database, what term is used for a single entry or row in a table?
    - a) Field

- b) Record
- c) Query
- d) Table
- **Answer**: b

Section 2: Short Answer Questions

1. Define a database and explain its importance in data management.
2. Differentiate between structured, semi-structured, and unstructured data, and provide an example for each.
3. Explain the purpose of each component in a database system: hardware, software, data, procedures, and users.
4. Describe the main differences between a file system and a database system, focusing on data redundancy, security, and scalability.
5. List and briefly describe the three types of relationships in a database, providing an example of each.

Section 3: Problem Solving and Practical Questions

1. **Scenario-Based Question**:
   - A university wants to store data on its students, courses, and enrollment information. Design the structure of this database by listing the tables and fields you would create. Define the relationships between these tables.
2. **Query Design**:
   - Using SQL, write a query to find all students who enrolled after a specific date (for example, January 1, 2022). Assume a **Students** table with fields such as StudentID, FirstName, LastName, and EnrollmentDate.
3. **Diagram Question**:

- Draw an Entity-Relationship (ER) diagram showing the relationships between tables in a simple library database. The database should have tables for **Books**, **Authors**, **Members**, and **Loans**.

4. **Analysis Question**:
   - A company currently uses a file system to store employee records, but it faces challenges with data duplication and security. Suggest three reasons why switching to a database system could improve its data management.
5. **Database Table Setup**:
   - Create a sample table structure for an **Employee** database with fields such as EmployeeID, Name, Position, Department, HireDate, and Salary. Describe what data type would be appropriate for each field and explain why.

Section 4: Essay Questions

1. Discuss the key advantages of using a Database Management System (DBMS) over a traditional file system. How does a DBMS improve data integrity, security, and consistency?
2. Describe the steps involved in designing a relational database, from defining entities to establishing relationships between tables. Provide examples to illustrate each step.
3. Explain the concept of data redundancy and data integrity. How does a relational database manage these issues more effectively than a file-based system?
4. Discuss the role of queries in a database system. Why are queries important, and how do they enable users to retrieve and manipulate data? Provide examples of different types of queries and their uses.
5. Explain how database relationships (one-to-one, one-to-many, many-to-many) represent real-world connections between data entities. Provide examples of each relationship type in a database setting, such as in a university, library, or hospital.