Activity: Databases

Activity Task 1: University Database Schema

Define the Problem:

Design a database for a university to manage students, courses, and professors.

Steps to Complete the Task:

1. Identify the tables: Students, Courses, Professors.
2. Decide on the columns for each table.
3. Connect the tables with relationships:
    - Each professor can teach many courses, but a course is taught by one professor (One-to-Many).
    - Each student can enroll in many courses, and each course can have many students (Many-to-Many).

Solution:

1. Tables and Attributes

Students

— StudentID (PK) – unique student identifier;

— FirstName – first name;

— LastName – last name;

— Email – email address;

— DateOfBirth – date of birth;

— Major – student's major.

Professors

— ProfessorID (PK) – unique professor identifier;

— FirstName – first name;

— LastName – last name;

— Email – email address;

— Department – department name.

Courses

— CourseID (PK) – unique course identifier;

— Title – course title;

— Credits – number of credits;

— ProfessorID (FK) – reference to professor (One-to-Many: one professor → many courses).

Enrollments (junction table for Many-to-Many)

— EnrollmentID (PK) – unique enrollment record;

— StudentID (FK) – reference to student;

— CourseID (FK) – reference to course;

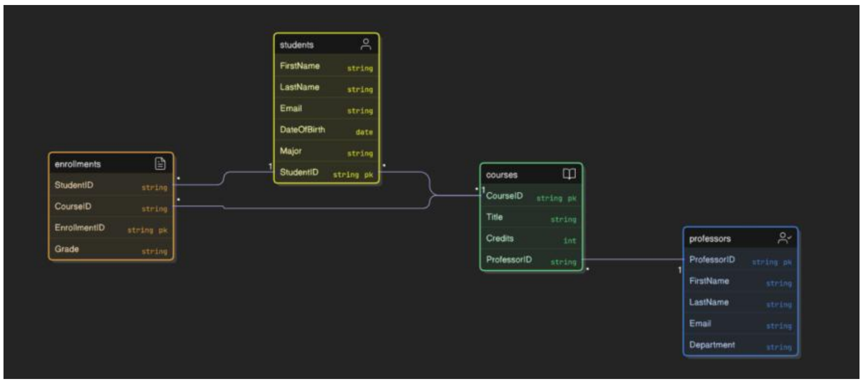
— Grade – grade (nullable until course completion).

2. Relationships

— Professors → Courses: one professor can teach many courses, but each course is taught by one professor (One-to-Many).

— Students ↔ Courses (via Enrollments): A student can enroll in many courses, and each course can have many students (Many-to-Many).

3. ER Diagram



4. SQL Script (DDL)

```sql
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Email VARCHAR(100) UNIQUE,
    DateOfBirth DATE,
    Major VARCHAR(100)
);

CREATE TABLE Professors (
    ProfessorID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Email VARCHAR(100) UNIQUE,
    Department VARCHAR(100)
);

CREATE TABLE Courses (
    CourseID INT PRIMARY KEY,
    Title VARCHAR(100),
    Credits INT,
    ProfessorID INT,
    FOREIGN KEY (ProfessorID) REFERENCES Professors(ProfessorID)
);

CREATE TABLE Enrollments (
    EnrollmentID INT PRIMARY KEY,
    StudentID INT,
    CourseID INT,
    Grade CHAR(2),
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);
```

Activity Task 2: Library Management Schema

Define the Problem:

Design a database for a library to manage members, books, and loans.

Steps to Complete the Task:

1. Identify the tables: Members, Books, Loans.
2. Decide on the columns for each table.
3. Connect the tables with relationships:
    - A loan connects a member to a book (One-to-Many).
    - A book can be borrowed many times by different members (Many-to-Many).

Solution:

1. Tables and Attributes

Members

— MemberID (PK) – unique member identifier;

— FirstName – first name;

— LastName – last name;

— Email – email address (unique);

— Phone – phone number;

— DateJoined – date when member joined.

Books

— BookID (PK) – unique book identifier;

— Title – book title;

— Author – author of the book;

— ISBN – international book number (unique);

— PublishedYear – year of publication;

— CopiesAvailable – number of available copies.

Loans (junction table for Many-to-Many)

— LoanID (PK) – unique loan identifier;

— MemberID (FK) – reference to member;

— BookID (FK) – reference to book;

— LoanDate – date when book was borrowed;

— DueDate – due date for return;

— ReturnDate – actual return date (nullable).

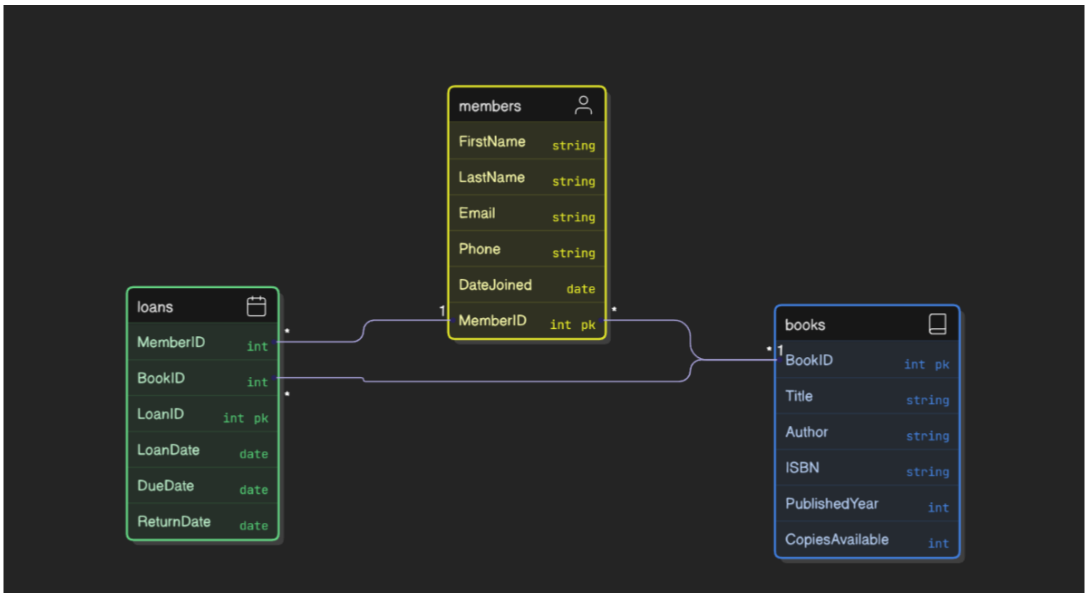
2. Relationships

— Members → Loans: One member can have many loans (One-to-Many);

— Books → Loans: One book can appear in many loans (One-to-Many);

— Members ↔ Books (via Loans): A member can borrow many books, and each book can be borrowed by many members (Many-to-Many, resolved by Loans).

3. ER Diagram



4. SQL Script (DDL)

```sql
CREATE TABLE Members (
    MemberID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Email VARCHAR(100) UNIQUE,
    Phone VARCHAR(20),
    DateJoined DATE
);

CREATE TABLE Books (
    BookID INT PRIMARY KEY,
    Title VARCHAR(150),
    Author VARCHAR(100),
    ISBN VARCHAR(20) UNIQUE,
    PublishedYear INT,
    CopiesAvailable INT
);

CREATE TABLE Loans (
    LoanID INT PRIMARY KEY,
    MemberID INT NOT NULL,
    BookID INT NOT NULL,
    LoanDate DATE NOT NULL,
    DueDate DATE NOT NULL,
    ReturnDate DATE,
    FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
    FOREIGN KEY (BookID) REFERENCES Books(BookID)
);
```