Database Design

This guide will help to understand the flow of the project.it will help to manage the below functionalities.

Project Name- University management system.

1. Database Structure (Tables)

Below are the main **tables**, their attributes.

Table	Key	Attributes
Department	**PK** departmentId	departmentName, headOfDepartment, location
Student	**PK** studentId, **FK** departmentId	firstName, lastName, dob, gender, email, phone, address, enrollmentDate
Faculty	**PK** facultyId, **FK** departmentId	firstName, lastName, email, phone, address, hireDate, role
Course	**PK** courseId, **FK** departmentId	courseName, courseCode, credits

Table	Key	Attributes
Enrollment	**PK** enrollmentId, **FK** studentId, **FK** courseId	semester, year, status
Grade	**PK** gradeId, **FK** studentId, **FK** courseId	semester, year, grade
Schedule	**PK** scheduleId, **FK** courseId, **FK** facultyId	day, startTime, endTime, location
Assignment	**PK** assignmentId, **FK** courseId	title, description, dueDate
Exam	**PK** examId, **FK** courseId	examDate, duration, type
Fee	**PK** feeId, **FK** studentId	amount, dueDate, status
Payment	**PK** paymentId, **FK** feeId	paymentDate, amount, paymentMethod, status
Event	**PK** eventId	eventName, eventDate, location, description
LibraryBook	**PK** bookId	title, author, isbn, publicationYear, availableCopies

Table	Key	Attributes
Attendance	**PK** attendanceId, **FK** studentId, **FK** courseId	date, status
Result	**PK** resultId, **FK** studentId	semester, year, gpa, status

primary (PK)

foreign keys (FK)

2. Relationships Between Tables

Relationship	Type
Department – Student:	One department has many students.
Department – Faculty:	One department has many faculty members.
Department – Course:	One department offers many courses.
Student – Enrollment – Course:	Many-to-Many resolved by Enrollment table.

Relationship	Type
Student – Grade – Course:	Many-to-Many resolved by Grade table
Course – Schedule – Faculty:	Many-to-Many resolved by Schedule table.
Course – Assignment	One-to-Many.
Course – Exam:	One-to-Many.
Student – Fee – Payment:	Student has many fees; Fee has many payments.
Student – Attendance – Course	Many-to-Many resolved by Attendance table.
Student – Result:	One-to-Many.
Event	Standalone.
LibraryBook-	Standalone (can later link to a “Loan” table).

3. ER Diagram (Text Description)

Department (departmentId PK)

|

|<-- Student (studentId PK)

| |<-- Fee (feeId PK)

| | |<-- Payment (paymentId PK)

| |<-- Enrollment (enrollmentId PK) --> Course

| |<-- Grade (gradeId PK) --> Course

| |<-- Attendance (attendanceId PK) --> Course

| |<-- Result (resultId PK)

|

|<-- Faculty (facultyId PK)

|

|<-- Course (courseId PK)

|

|<-- Assignment (assignmentId PK)

|<-- Schedule (scheduleId PK) --> Faculty

|<-- Exam (examId PK)

|

|<-- Event (eventId PK)

|

|<-- LibraryBook (bookId PK)

4.ER Diagram Explanation :-

- **Department** is the parent of Students, Faculty, and Courses.
- **Student** records store all personal + academic info.
- **Faculty** stores teachers' info; connected to **Schedule** to show who teaches which course.
- **Course** belongs to a department; linked to assignments, exams, schedules.
- **Enrollment** links students to courses (many-to-many).
- **Grade** records what grade a student got in each course per semester/year.
- **Schedule** shows when and where a course is taught and by which faculty.
- **Assignment** and **Exam** show tasks and tests for a course.
- **Fee** records each student's charges; **Payment** records actual payments made.
- **Attendance** records daily class attendance per student per course.
- **Result** is an overall GPA/academic status per semester/year for each student.
- **Event** lists university events.
- **LibraryBook** lists library's available books.

Module -	Description
Department	Stores all department details such as name, head of -department, and location. Acts as the parent table for students, faculty, and courses.
Student-	Contains personal, academic, and contact details of each student. Linked to a department and other modules like Fee, Result, and Enrollment.
Faculty-	Contains information about university staff members (professors, lecturers, etc.) including their role, hire date, and department.
Course-	Represents the academic courses offered by departments. Connected to assignments, exams, enrollments, schedules, and grades.
Enrollment-	Junction table linking students and courses, storing enrollment details such as semester, year, and status.
Grade-	Records students' grades per course for each semester and year.
Schedule-	Defines when and where courses occur, along with which faculty teaches them.

Module -	Description
Assignment	Stores assignment information for each course, including title, description, and due date.
Exam	Contains details about exams for each course, such as exam date, duration, and type.
Fee -	Stores financial details for students, including amount due, due date, and payment status.
Payment -	Tracks actual payments made against fees, including date, method, and amount.
Attendance-	Records student attendance for each class session (course + date).
Result-	Stores the student's semester and annual results, including GPA and academic standing.
Event ->	Stores details about university events, such as name, date, and description.
LibraryBook	Holds details about library books such as title, author, ->publication year, and available copies. (Future scope: link to Loan table).

Table structure

1. Department Table.

```
CREATE TABLE Department (
    departmentId INT PRIMARY KEY,
    departmentName VARCHAR(100) NOT NULL,
    headOfDepartment VARCHAR(100),
    location VARCHAR(100)
);
```

2️⃣ Student Table

```
CREATE TABLE Student (
    studentId INT PRIMARY KEY,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(50) NOT NULL,
    dob DATE,
    gender VARCHAR(10),
    email VARCHAR(100),
    phone VARCHAR(20),
    address VARCHAR(200),
    enrollmentDate DATE,
    departmentId INT,
    FOREIGN KEY (departmentId) REFERENCES Department(departmentId)
);
```

3️⃣ Faculty Table.

```
CREATE TABLE Faculty (
    facultyId INT PRIMARY KEY,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(50) NOT NULL,
    email VARCHAR(100),
    phone VARCHAR(20),
    address VARCHAR(200),
    departmentId INT,
    hireDate DATE,
```

```
    role VARCHAR(50),
    FOREIGN KEY (departmentId) REFERENCES
Department(departmentId)
);
```

4️⃣ Course Table

```
CREATE TABLE Course (
    courseId INT PRIMARY KEY,
    courseName VARCHAR(100) NOT NULL,
    courseCode VARCHAR(20),
    credits INT,
    departmentId INT,
    FOREIGN KEY (departmentId) REFERENCES
Department(departmentId)
);
```

5️⃣ Enrollment Table

```
CREATE TABLE Enrollment (
    enrollmentId INT PRIMARY KEY,
    studentId INT,
    courseId INT,
    semester VARCHAR(20),
    year INT,
```

```
    status VARCHAR(20),
    FOREIGN KEY (studentId) REFERENCES Student(studentId),
    FOREIGN KEY (courseId) REFERENCES Course(courseId)
);
```

6️⃣ Grade Table

```
CREATE TABLE Grade (
    gradeId INT PRIMARY KEY,
    studentId INT,
    courseId INT,
    semester VARCHAR(20),
    year INT,
    grade VARCHAR(5),
    FOREIGN KEY (studentId) REFERENCES Student(studentId),
    FOREIGN KEY (courseId) REFERENCES Course(courseId)
);
```

7️⃣ Schedule Table

```
CREATE TABLE Schedule (
    scheduleId INT PRIMARY KEY,
    courseId INT,
    facultyId INT,
    day VARCHAR(20),
```

```
    startTime TIME,
    endTime TIME,
    location VARCHAR(100),
    FOREIGN KEY (courseId) REFERENCES Course(courseId),
    FOREIGN KEY (facultyId) REFERENCES Faculty(facultyId)
);
```

8️⃣ Assignment Table

```
CREATE TABLE Assignment (
    assignmentId INT PRIMARY KEY,
    courseId INT,
    title VARCHAR(100),
    description TEXT,
    dueDate DATE,
    FOREIGN KEY (courseId) REFERENCES Course(courseId)
);
```

9️⃣ Exam Table

```
CREATE TABLE Exam (
    examId INT PRIMARY KEY,
    courseId INT,
```

```
    examDate DATE,
    duration INT,
    type VARCHAR(50),
    FOREIGN KEY (courseId) REFERENCES Course(courseId)
);
```

10 Fee Table

```
CREATE TABLE Fee (
    feeId INT PRIMARY KEY,
    studentId INT,
    amount DECIMAL(10,2),
    dueDate DATE,
    status VARCHAR(20),
    FOREIGN KEY (studentId) REFERENCES Student(studentId)
);
```

11 Payment Table

```
CREATE TABLE Payment (
    paymentId INT PRIMARY KEY,
    feeId INT,
    paymentDate DATE,
    amount DECIMAL(10,2),
```

```
    paymentMethod VARCHAR(50),
    status VARCHAR(20),
    FOREIGN KEY (feeId) REFERENCES Fee(feeId)
);
```

12. Attendance Table

```
CREATE TABLE Attendance (
    attendanceId INT PRIMARY KEY,
    studentId INT,
    courseId INT,
    date DATE,
    status VARCHAR(20),
    FOREIGN KEY (studentId) REFERENCES Student(studentId),
    FOREIGN KEY (courseId) REFERENCES Course(courseId)
);
```

13. Result Table

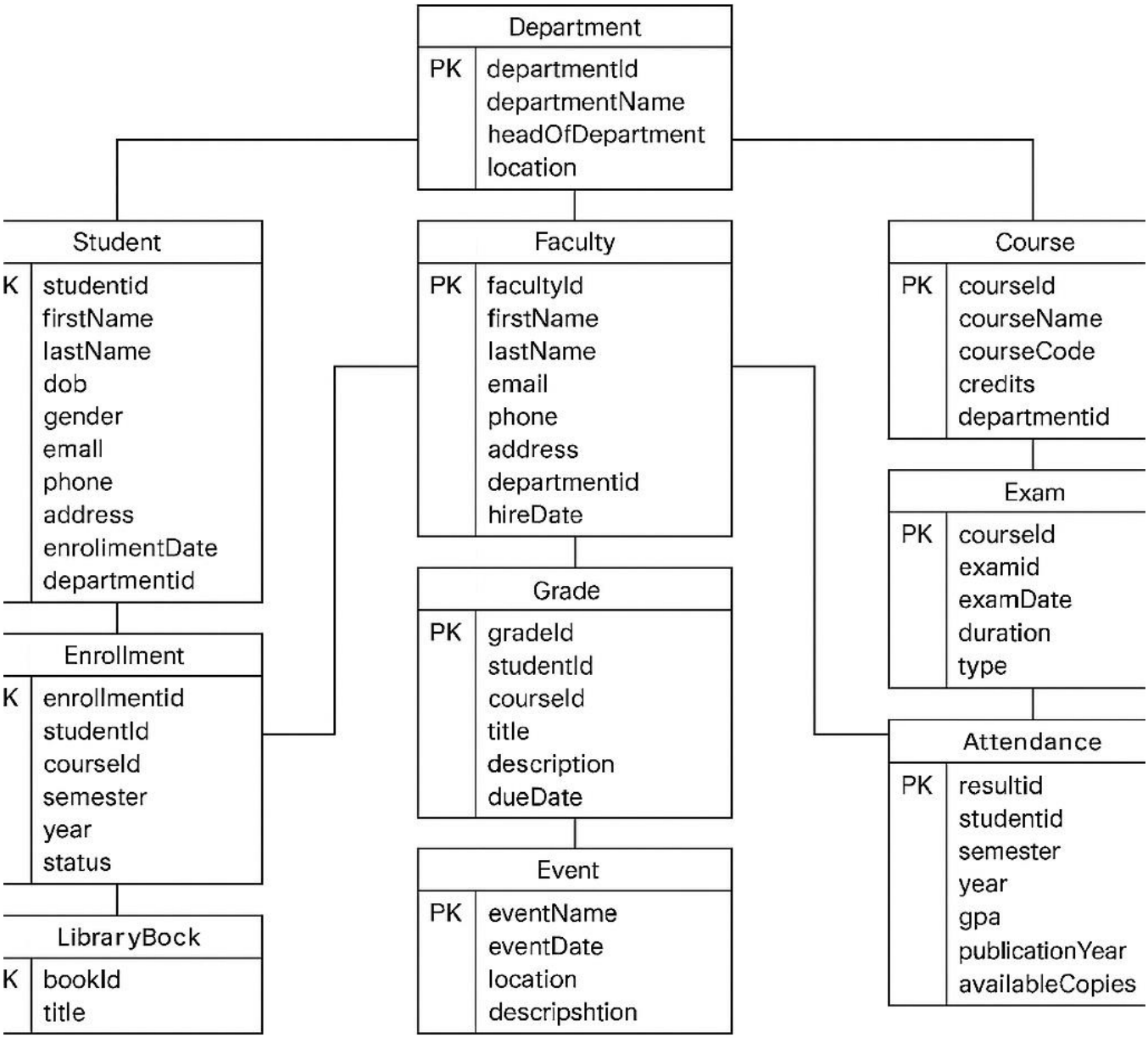
```
CREATE TABLE Result (
    resultId INT PRIMARY KEY,
    studentId INT,
    semester VARCHAR(20),
```

```
    year INT,
    gpa DECIMAL(3,2),
    status VARCHAR(20),
    FOREIGN KEY (studentId) REFERENCES Student(studentId)
);
```

14. Event Table

```
CREATE TABLE Event (
    eventId INT PRIMARY KEY,
    eventName VARCHAR(100),
    eventDate DATE,
    location VARCHAR(100),
    description TEXT
);
```

ER diagram -→


Department
PK
departmentId
departmentName
headOfDepartment
location
Student
K
studentid
firstName
lastName
dob
gender
emall
phone
address
enrolimentDate
departmentid
Faculty
PK
facultyId
firstName
lastName
email
phone
address
departmentid
hireDate
Course
PK
courseId
courseName
courseCode
credits
departmentid
Exam
PK
courseId
examid
examDate
duration
type
Enrollment
K
enrollmentid
studentId
courseId
semester
year
status
Grade
PK
gradeId
studentId
courseId
title
description
dueDate
Attendance
PK
resultid
studentid
semester
year
gpa
publicationYear
availableCopies
LibraryBock
K
bookId
title
Event
PK
eventName
eventDate
location
descripshtion