**DATABASE SCHEMA DESIGN**

1. **Students**

Stores information about students.

CREATE TABLE students (

student\_id SERIAL PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

gender VARCHAR(10),

date\_of\_birth DATE,

email VARCHAR(100) UNIQUE,

phone\_number VARCHAR(20),

address TEXT,

registration\_date DATE DEFAULT CURRENT\_DATE

);

1. **Courses**

Stores information about available courses.

CREATE TABLE courses (

course\_id SERIAL PRIMARY KEY,

course\_code VARCHAR(10) UNIQUE,

course\_name VARCHAR(100),

credit\_hours INT CHECK (credit\_hours > 0),

department VARCHAR(50),

department\_id INT REFERENCES departments(department\_id)

);

1. **Lecturers**

Stores lecturer details.

CREATE TABLE lecturers (

lecturer\_id SERIAL PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(100) UNIQUE,

phone\_number VARCHAR(20),

department VARCHAR(50),

department\_id INT REFERENCES departments(department\_id),

hire\_date DATE

);

1. **Student Courses**

Junction table for many-to-many relationship between students and courses.

CREATE TABLE student\_courses (

student\_id INT REFERENCES students(student\_id),

course\_id INT REFERENCES courses(course\_id),

enrollment\_date DATE DEFAULT CURRENT\_DATE,

grade CHAR(2),

PRIMARY KEY (student\_id, course\_id)

);

1. **Lecturer\_courses**

Junction table for many-to-many relationship between lecturers and courses.

CREATE TABLE lecturer\_courses (

lecturer\_id INT REFERENCES lecturers(lecturer\_id),

course\_id INT REFERENCES courses(course\_id),

assigned\_date DATE DEFAULT CURRENT\_DATE,

PRIMARY KEY (lecturer\_id, course\_id)

);

1. **Payments**

Stores payment records of students.

CREATE TABLE payments (

payment\_id SERIAL PRIMARY KEY,

student\_id INT REFERENCES students(student\_id),

payment\_date DATE DEFAULT CURRENT\_DATE,

amount NUMERIC(10, 2),

payment\_method VARCHAR(50),

payment\_status VARCHAR(20) CHECK (payment\_status IN ('Pending', 'Completed', 'Failed'))

);

1. **Departments**

To manage different academic departments.

CREATE TABLE departments (

department\_id SERIAL PRIMARY KEY,

department\_name VARCHAR(100),

head\_of\_department INT REFERENCES lecturers(lecturer\_id)

);

**DOCUMENTED ERD (Entity Relationship Diagram)**

**ENTITY DESCRIPTIONS**

| **Entity** | **Description** |
| --- | --- |
| Students | Represents individual students |
| Courses | Represents courses offered |
| Lecturers | University teaching staff |
| Payments | Payments made by students |
| student\_courses | Tracks enrollment & grades |
| lecturer\_courses | Assigns lecturers to courses |
| Departments | Academic departments |

**RELATIONSHIPS**

| **Relationship** | **Type** | **Description** |
| --- | --- | --- |
| Student ↔ Course | Many-to-Many | Via student\_courses |
| Lecturer ↔ Course | Many-to-Many | Via lecturer\_courses |
| Student → Payment | One-to-Many | One student can have many payments |
| Course → Department | Many-to-One | Courses belong to a department |
| Lecturer → Department | Many-to-One | Lecturers work in departments |
| Department → Head (Lecturer) | One-to-One | A department has a head (lecturer) |

**ERD DIAGRAM**

