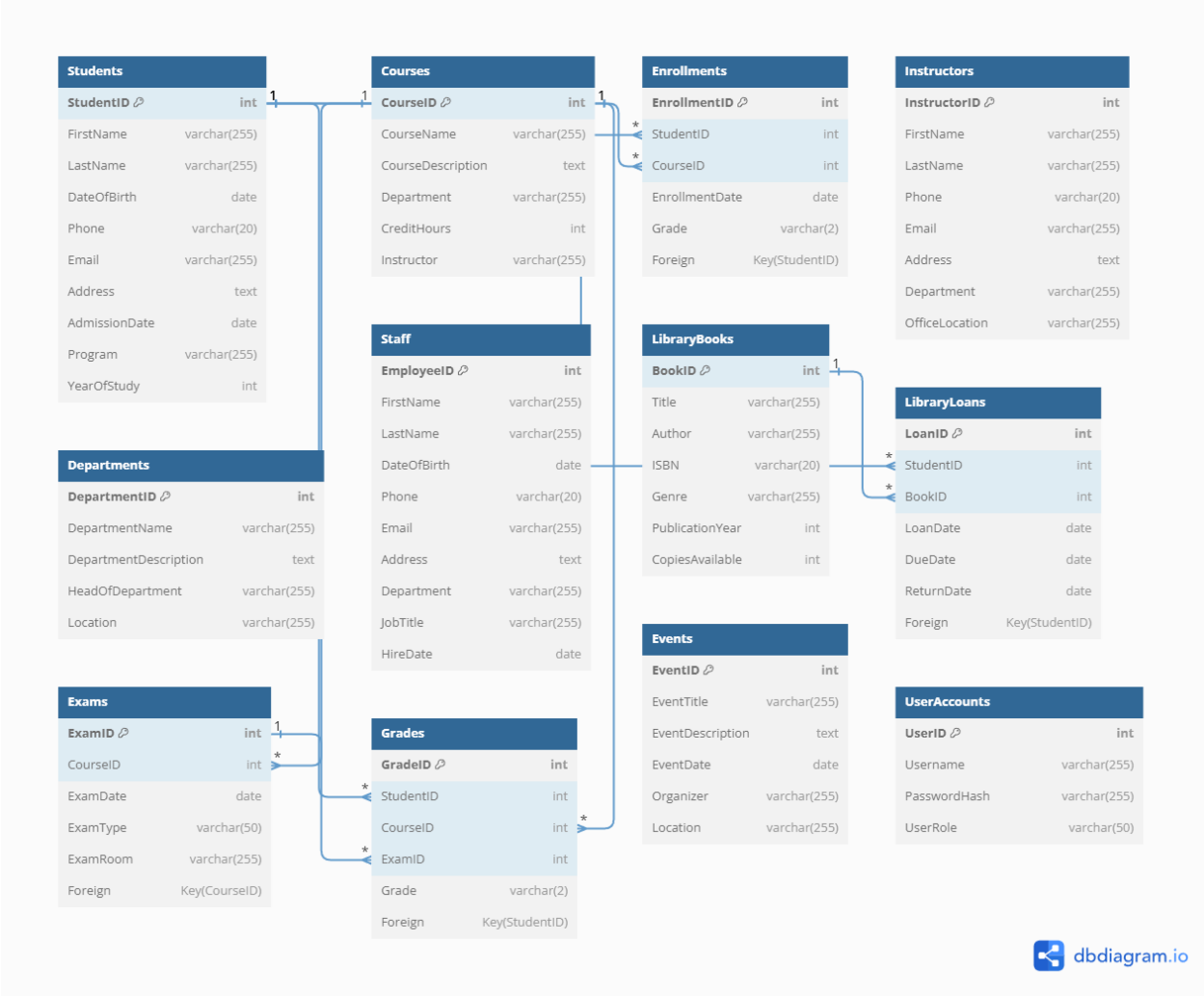
Data Model



Create all table

```
-- Create Students table
CREATE TABLE Students (
    StudentID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(255),
    LastName VARCHAR(255),
    DateOfBirth DATE,
    Phone VARCHAR(20),
    Email VARCHAR(255),
    Address TEXT,
    AdmissionDate DATE,
    Program VARCHAR(255),
    YearOfStudy INT
);
```

```
-- Create Courses table
CREATE TABLE Courses (
    CourseID INT AUTO_INCREMENT PRIMARY KEY,
    CourseName VARCHAR(255),
    CourseDescription TEXT,
    Department VARCHAR(255),
    CreditHours INT,
    Instructor VARCHAR(255)
);

-- Create Enrollments table
CREATE TABLE Enrollments (
    EnrollmentID INT AUTO_INCREMENT PRIMARY KEY,
    StudentID INT,
    CourseID INT,
    EnrollmentDate DATE,
    Grade VARCHAR(2),
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);

-- Create Instructors table
CREATE TABLE Instructors (
    InstructorID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(255),
    LastName VARCHAR(255),
    Phone VARCHAR(20),
    Email VARCHAR(255),
    Address TEXT,
    Department VARCHAR(255),
    OfficeLocation VARCHAR(255)
);

-- Create Departments/Programs table
CREATE TABLE Departments (
    DepartmentID INT AUTO_INCREMENT PRIMARY KEY,
    DepartmentName VARCHAR(255),
    DepartmentDescription TEXT,
    HeadOfDepartment VARCHAR(255),
    Location VARCHAR(255)
);

-- Create Staff/Employees table
CREATE TABLE Staff (
    EmployeeID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(255),
    LastName VARCHAR(255),
    DateOfBirth DATE,
    Phone VARCHAR(20),
    Email VARCHAR(255),
```

```
    Address TEXT,
    Department VARCHAR(255),
    JobTitle VARCHAR(255),
    HireDate DATE
);

-- Create Library Books table
CREATE TABLE LibraryBooks (
    BookID INT AUTO_INCREMENT PRIMARY KEY,
    Title VARCHAR(255),
    Author VARCHAR(255),
    ISBN VARCHAR(20),
    Genre VARCHAR(255),
    PublicationYear INT,
    CopiesAvailable INT
);

-- Create Library Loans table
CREATE TABLE LibraryLoans (
    LoanID INT AUTO_INCREMENT PRIMARY KEY,
    StudentID INT,
    BookID INT,
    LoanDate DATE,
    DueDate DATE,
    ReturnDate DATE,
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    FOREIGN KEY (BookID) REFERENCES LibraryBooks(BookID)
);

-- Create Exams table
CREATE TABLE Exams (
    ExamID INT AUTO_INCREMENT PRIMARY KEY,
    CourseID INT,
    ExamDate DATE,
    ExamType VARCHAR(50),
    ExamRoom VARCHAR(255),
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);

-- Create Grades table
CREATE TABLE Grades (
    GradeID INT AUTO_INCREMENT PRIMARY KEY,
    StudentID INT,
    CourseID INT,
    ExamID INT,
    Grade VARCHAR(2),
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID),
    FOREIGN KEY (ExamID) REFERENCES Exams(ExamID)
);
```

```
-- Create Events/Announcements table
CREATE TABLE Events (
    EventID INT AUTO_INCREMENT PRIMARY KEY,
    EventTitle VARCHAR(255),
    EventDescription TEXT,
    EventDate DATE,
    Organizer VARCHAR(255),
    Location VARCHAR(255)
);

-- Create User Accounts table for authentication
CREATE TABLE UserAccounts (
    UserID INT AUTO_INCREMENT PRIMARY KEY,
    Username VARCHAR(255),
    PasswordHash VARCHAR(255), -- Store hashed passwords
    UserRole VARCHAR(50)
);
```

Insert some value

```
-- Insert sample data into Students table
INSERT INTO Students (FirstName, LastName, DateOfBirth, Phone, Email,
Address, AdmissionDate, Program, YearOfStudy)
VALUES
    ('John', 'Doe', '1995-05-15', '+1234567890',
'john.doe@example.com', '123 Main St', '2022-09-01', 'Computer
Science', 1),
    ('Jane', 'Smith', '1998-08-22', '+9876543210',
'jane.smith@example.com', '456 Elm St', '2021-09-01', 'Biology', 2);

-- Insert sample data into Courses table
INSERT INTO Courses (CourseName, CourseDescription, Department,
CreditHours, Instructor)
VALUES
    ('Database Management', 'Introduction to database systems.',
'Computer Science', 3, 'Prof. Johnson'),
    ('Biology 101', 'Foundations of biology.', 'Biology', 4, 'Dr.
Smith');

-- Insert sample data into Enrollments table
INSERT INTO Enrollments (StudentID, CourseID, EnrollmentDate, Grade)
VALUES
    (1, 1, '2022-09-01', 'A'),
    (2, 2, '2021-09-01', 'B');

-- Insert sample data into Instructors table
INSERT INTO Instructors (FirstName, LastName, Phone, Email, Address,
Department, OfficeLocation)
VALUES
```

```
    ('Prof.', 'Johnson', '+1111111111', 'prof.j@example.com', '789 Oak
St', 'Computer Science', 'Room 101'),
    ('Dr.', 'Smith', '+2222222222', 'dr.smith@example.com', '567 Maple
St', 'Biology', 'Room 202');

-- Insert sample data into Departments table
INSERT INTO Departments (DepartmentName, DepartmentDescription,
HeadOfDepartment, Location)
VALUES
    ('Computer Science', 'Department of Computer Science', 'Prof.
Johnson', 'Science Building'),
    ('Biology', 'Department of Biology', 'Dr. Smith', 'Biology
Building');

-- Insert sample data into Staff table
INSERT INTO Staff (FirstName, LastName, DateOfBirth, Phone, Email,
Address, Department, JobTitle, HireDate)
VALUES
    ('Alice', 'Johnson', '1980-03-10', '+3333333333',
'alice.j@example.com', '101 Pine St', 'Computer Science', 'Admin',
'2010-05-15'),
    ('Bob', 'Smith', '1975-12-20', '+4444444444',
'bob.smith@example.com', '202 Cedar St', 'Biology', 'Lab Technician',
'2005-08-30');

-- Insert sample data into LibraryBooks table
INSERT INTO LibraryBooks (Title, Author, ISBN, Genre, PublicationYear,
CopiesAvailable)
VALUES
    ('Introduction to SQL', 'John Doe', '1234567890', 'Technical',
2020, 5),
    ('Biology: Concepts and Connections', 'Jane Smith', '0987654321',
'Science', 2018, 7);

-- Insert sample data into LibraryLoans table
INSERT INTO LibraryLoans (StudentID, BookID, LoanDate, DueDate,
ReturnDate)
VALUES
    (1, 1, '2022-09-15', '2022-10-15', NULL),
    (2, 2, '2022-08-20', '2022-09-20', '2022-09-15');

-- Insert sample data into Exams table
INSERT INTO Exams (CourseID, ExamDate, ExamType, ExamRoom)
VALUES
    (1, '2022-12-10', 'Final', 'Room A'),
    (2, '2022-11-15', 'Midterm', 'Room B');

-- Insert sample data into Grades table
INSERT INTO Grades (StudentID, CourseID, ExamID, Grade)
VALUES
```

```
    (1, 1, 1, 'A'),
    (2, 2, 2, 'B');

-- Insert sample data into Events table
INSERT INTO Events (EventTitle, EventDescription, EventDate,
Organizer, Location)
VALUES
    ('Career Fair', 'Meet potential employers and explore job
opportunities.', '2023-03-01', 'Career Services', 'Student Center'),
    ('Guest Lecture', 'Topic: Advances in Computer Science.', '2023-
04-15', 'Computer Science Department', 'Lecture Hall');

-- Insert sample data into UserAccounts table
INSERT INTO UserAccounts (Username, PasswordHash, UserRole)
VALUES
    ('john_doe', 'hashed_password_1', 'Student'),
    ('prof_johnson', 'hashed_password_2', 'Instructor');
```

Some Query

Student-related Queries:

1. Retrieve a list of all students in the Computer Science program.
2. Find the students who are currently enrolled in a specific course.
3. Get the total number of students in each year of study.
4. Find students who have not yet enrolled in any courses.
5. Retrieve the student with the highest grade in a particular course.

1.select * from students where Program='Computer Science';

1. select * from students where AdmissionDate= '2022-09-01' and Program ='Computer Science';

2. SELECT YEAR(AdmissionDate) AS AdmissionYear, COUNT(StudentID) AS TotalStudents FROM Students GROUP BY AdmissionYear

4.SELECT * FROM students WHERE StudentID not IN (SELECT StudentID FROM enrollments);

5.SELECT students.*, enrollments.Grade FROM students RIGHT JOIN enrollments ON students.StudentID = enrollments.StudentID WHERE enrollments.Grade = 'A';

Course-related Queries:

1. List all courses offered by the Biology department.
2. Find courses that have more than 50 enrolled students.
3. Retrieve the instructors for a specific course.
4. Get a list of courses with no enrolled students.
5. Find courses with a specific keyword in their description.

```
6.SELECT * from courses
where Department = 'Biology';
```

```
7.SELECT courses.CourseID, courses.CourseName
FROM courses
INNER JOIN enrollments ON courses.CourseID = enrollments.CourseID
GROUP BY courses.CourseID, courses.CourseName
HAVING COUNT(enrollments.StudentID) > 50;

8.SELECT courses.CourseID, courses.CourseName ,courses.Instructor FROM
courses;

9.SELECT courses.CourseID, courses.CourseName
FROM courses
LEFT JOIN enrollments ON courses.CourseID = enrollments.CourseID
WHERE enrollments.StudentID IS NULL;

10.SELECT CourseID, CourseName, Description
FROM courses
WHERE Description LIKE '%specific_keyword%';
```

Enrollment-related Queries:

1. Retrieve a student's course enrollment history.
2. Find all courses a specific student is currently enrolled in.
3. Get the total number of enrolled students in each course.
4. Find students who have dropped a particular course.
5. Retrieve the average grade for a specific course.

```
11. SELECT *FROM enrollments
WHERE enrollments.StudentID=2;

12.SELECT Courses.course_name
FROM Students
JOIN Enrollments ON Students.student_id = Enrollments.student_id
JOIN Courses ON Enrollments.course_id = Courses.course_id
WHERE Students.student_id = 2;

13.SELECT Courses.course_name, COUNT(Enrollments.student_id) AS
total_students
FROM Courses
LEFT JOIN Enrollments ON Courses.course_id = Enrollments.course_id
GROUP BY Courses.course_name;

14.SELECT Students.student_name
FROM Students
JOIN Enrollments ON Students.student_id = Enrollments.student_id
WHERE Enrollments.course_id = 3 AND Enrollments.status = 'dropped';

15.SELECT AVG(Enrollments.grade) AS average_grade
FROM Enrollments
WHERE Enrollments.course_id = 1;
```

Instructor-related Queries:

1. List all instructors in the Computer Science department.
2. Find instructors who teach more than one course.
3. Get the list of courses taught by a specific instructor.
4. Find instructors who have not taught any courses recently.
5. Retrieve the contact information for the head of a department.

```
16.SELECT *
FROM instructors
WHERE department_id = (SELECT department_id FROM departments WHERE
department_name = 'Computer Science');

17.SELECT instructor_id, COUNT(course_id) as course_count
FROM course_assignments
GROUP BY instructor_id
HAVING COUNT(course_id) > 1;

18.SELECT Courses.course_name
FROM Courses
JOIN course_assignments ON Courses.course_id =
course_assignments.course_id
WHERE course_assignments.instructor_id = 123;

19.SELECT Instructors.instructor_name
FROM Instructors
LEFT JOIN course_assignments ON Instructors.instructor_id =
course_assignments.instructor_id
WHERE course_assignments.last_taught_date < '2022-01-01' OR
course_assignments.last_taught_date IS NULL;

20.SELECT head_instructor_name, contact_info
FROM departments
JOIN instructors ON departments.head_instructor_id =
instructors.instructor_id;
```

Library-related Queries:

1. List all available books in the library.
2. Find books with a specific genre.
3. Retrieve the books that are currently checked out by a particular student.
4. Get a list of overdue books.
5. Find books by a specific author.

```
21.SELECT *
FROM Books
WHERE availability = 'available';

22. SELECT Books.title
FROM Books
JOIN Checkouts ON Books.book_id = Checkouts.book_id
```

```
WHERE Checkouts.student_id = 'Student ID';

23.SELECT Books.title
FROM Books
JOIN Checkouts ON Books.book_id = Checkouts.book_id
WHERE Checkouts.due_date < NOW() AND Checkouts.return_date IS NULL;

24.SELECT *
FROM Books
WHERE genre = 'Genre Name';

25.SELECT *
FROM Books
WHERE author = 'Author Name';
```