# *TABLES*

***Project: Student Management System***This document contains normalized table structures (in 3NF) with attributes, data types, and explanations of normalization decisions.

## Department Table (3NF)

CREATE TABLE Department (  
 DepartmentID INT PRIMARY KEY IDENTITY(1,1),  
 DepartmentName VARCHAR(100) NOT NULL UNIQUE  
);  
**Justification:**  
Each department has a unique ID and name. No transitive or partial dependencies exist.

## Student Table (3NF)

CREATE TABLE Student (  
 StudentID INT PRIMARY KEY IDENTITY(1,1),  
 FullName VARCHAR(100) NOT NULL,  
 Email VARCHAR(100) UNIQUE NOT NULL,  
 DateOfBirth DATE,  
 Gender VARCHAR(10),  
 DepartmentID INT,  
 FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)  
);  
**Justification:**  
Student data depends only on StudentID. Department is linked via FK; no transitive dependency.

## Course Table (3NF)

CREATE TABLE Course (  
 CourseID INT PRIMARY KEY IDENTITY(1,1),  
 CourseName VARCHAR(100) NOT NULL,  
 CreditHours INT CHECK (CreditHours > 0 AND CreditHours <= 6),  
 DepartmentID INT,  
 FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)  
);  
**Justification:**  
Each course is linked to one department; no non-key attributes depend on other non-key attributes.

## Faculty Table (3NF)

CREATE TABLE Faculty (  
 FacultyID INT PRIMARY KEY IDENTITY(1,1),  
 FullName VARCHAR(100) NOT NULL,  
 Email VARCHAR(100) UNIQUE NOT NULL,  
 DepartmentID INT,  
 FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)  
);  
**Justification:**  
Faculty info depends directly on FacultyID. Department relations are normalized.

## Enrollment Table (3NF)

CREATE TABLE Enrollment (  
 EnrollmentID INT PRIMARY KEY IDENTITY(1,1),  
 StudentID INT,  
 CourseID INT,  
 EnrollmentDate DATE DEFAULT GETDATE(),  
 FOREIGN KEY (StudentID) REFERENCES Student(StudentID),  
 FOREIGN KEY (CourseID) REFERENCES Course(CourseID),  
 CONSTRAINT UC\_StudentCourse UNIQUE(StudentID, CourseID)  
);  
**Justification:**  
No derived data or partial dependencies. Bridge table for many-to-many relationships.

## Attendance Table (3NF)

CREATE TABLE Attendance (  
 AttendanceID INT PRIMARY KEY IDENTITY(1,1),  
 StudentID INT,  
 CourseID INT,  
 Date DATE,  
 Status VARCHAR(10) CHECK (Status IN ('Present', 'Absent')),  
 FOREIGN KEY (StudentID) REFERENCES Student(StudentID),  
 FOREIGN KEY (CourseID) REFERENCES Course(CourseID)  
);

**Justification:**  
Attendance records are atomic and depend only on AttendanceID. No derived or transitive data.

## Grades Table (3NF)

CREATE TABLE Grades (  
 GradeID INT PRIMARY KEY IDENTITY(1,1),  
 StudentID INT,  
 CourseID INT,  
 Semester VARCHAR(20),  
 Grade VARCHAR(2) CHECK (Grade IN ('A', 'B', 'C', 'D', 'F')),  
 FOREIGN KEY (StudentID) REFERENCES Student(StudentID),  
 FOREIGN KEY (CourseID) REFERENCES Course(CourseID)  
);  
**Justification:**  
Grades table stores one grade per student per course per semester. Fully normalized.

## Assignments Table (3NF)

CREATE TABLE Assignments (  
 AssignmentID INT PRIMARY KEY IDENTITY(1,1),  
 CourseID INT,  
 Title VARCHAR(100),  
 DueDate DATE,  
 FOREIGN KEY (CourseID) REFERENCES Course(CourseID)  
);  
**Justification:**  
Each assignment belongs to a course. Fully depends on AssignmentID. No transitive dependency.

## Assignment\_Submissions Table (3NF)

CREATE TABLE Assignment\_Submissions (  
 SubmissionID INT PRIMARY KEY IDENTITY(1,1),  
 AssignmentID INT,  
 StudentID INT,  
 SubmissionDate DATE,  
 Marks INT CHECK (Marks >= 0 AND Marks <= 100),  
 FOREIGN KEY (AssignmentID) REFERENCES Assignments(AssignmentID),  
 FOREIGN KEY (StudentID) REFERENCES Student(StudentID)  
);

**Justification:**Each submission is linked to an assignment and student. No derived or partial dependencies.

## Exam Table (3NF)

CREATE TABLE Exam (  
 ExamID INT PRIMARY KEY IDENTITY(1,1),  
 CourseID INT,  
 ExamDate DATE,  
 Type VARCHAR(20) CHECK (Type IN ('Midterm', 'Final')),  
 FOREIGN KEY (CourseID) REFERENCES Course(CourseID)  
);  
**Justification:**  
Exam details depend on ExamID and relate directly to Course. Fully normalized.

## Exam\_Result Table (3NF)

CREATE TABLE Exam\_Result (  
 ResultID INT PRIMARY KEY IDENTITY(1,1),  
 ExamID INT,  
 StudentID INT,  
 MarksObtained INT CHECK (MarksObtained >= 0 AND MarksObtained <= 100),  
 FOREIGN KEY (ExamID) REFERENCES Exam(ExamID),  
 FOREIGN KEY (StudentID) REFERENCES Student(StudentID)  
);  
**Justification:**  
Each result is atomic and linked to a specific student and exam. No transitive dependencies.

## Fee Table (3NF)

CREATE TABLE Fee (  
 FeeID INT PRIMARY KEY IDENTITY(1,1),  
 StudentID INT,  
 Semester VARCHAR(20),  
 TotalAmount DECIMAL(10,2),  
 PaidAmount DECIMAL(10,2),  
 DueAmount AS (TotalAmount - PaidAmount) PERSISTED,  
 FOREIGN KEY (StudentID) REFERENCES Student(StudentID)  
);  
**Justification:**  
Fee info is student- and semester-specific. Computed field (DueAmount) is allowed with persistence.

## Payment Table (3NF)

CREATE TABLE Payment (  
 PaymentID INT PRIMARY KEY IDENTITY(1,1),  
 FeeID INT,  
 PaymentDate DATE DEFAULT GETDATE(),  
 AmountPaid DECIMAL(10,2) CHECK (AmountPaid > 0),  
 FOREIGN KEY (FeeID) REFERENCES Fee(FeeID)  
);  
**Justification:**  
Each payment relates to one fee entry. Fully normalized.

## Library Table (3NF)

CREATE TABLE Library (  
 BookID INT PRIMARY KEY IDENTITY(1,1),  
 Title VARCHAR(100),  
 Author VARCHAR(100),  
 ISBN VARCHAR(20) UNIQUE,  
 TotalCopies INT,  
 AvailableCopies INT  
);  
**Justification:**  
Stores atomic data for books. No dependencies between non-key attributes.

## Book\_Issues Table (3NF)

CREATE TABLE Book\_Issues (  
 IssueID INT PRIMARY KEY IDENTITY(1,1),  
 StudentID INT,  
 BookID INT,  
 IssueDate DATE,  
 ReturnDate DATE,  
 FOREIGN KEY (StudentID) REFERENCES Student(StudentID),  
 FOREIGN KEY (BookID) REFERENCES Library(BookID)  
);  
**Justification:**  
Each issue record links a student and book with dates. No derived or transitive fields.