**TRƯỜNG ĐẠI HỌC THỦY LỢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÀI TẬP LỚN**

**HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU**

**Đề tài:**

**Xây dựng hệ thống quản lý sinh viên**

**Nhóm thực hiện: Nhóm 3 – 63HTTT2**

**Thành viên nhóm:** Nguyễn Thị Thu Hằng – MSV: 2151163688

Hồ Thủy Tiên – MSV: 2151163732

Khổng Thị Vân – MSV: 2151163741

Hoàng Khánh Hà – MSV:

Vũ Hà Lâm – MSV: 2161163701

**Giảng viên hướng dẫn: Ths. Nguyễn Ngọc Quỳnh Châu**

**Hà Nội – 1/2025**

**LỜI MỞ ĐẦU**

Ngày nay công nghệ thông tin đã được ứng dụng rất rộng rãi và phổ biến trong tất cả các lĩnh vực của đời sống xã hội. Hầu như tất cả các cơ quan đơn vị từ lớn đến nhỏ đều cố gắng đưa công nghệ thông tin vào hỗ trợ trong công việc của mình. Trong bối cảnh công nghệ, đặc biệt là công nghệ thông tin ngày càng phát triển như vũ bão, thay đổi trong khoảng thời gian rất ngắn thì ở nước ta càng cần phải đẩy nhanh tốc độ của việc áp dụng công nghệ thông tin vào mọi lĩnh vực của cuộc sống. Công nghệ thông tin cũng là một trong những tiêu chí để đánh giá sự phát triển của một xã hội. Qua đó ta cũng thấy được tầm quan trọng trong chiến lược công nghệ thông tin ở nước ta.

Trong những năm gần đây, các chương trình quản lý không còn xa lạ với các cơ quan hành chính sự nghiệp, các doanh nghiệp …Quản lý đã xâm nhập vào lĩnh vực đời sống xã hội như: Quản lý nhân sự, quản lý lương … trong các cơ quan, quản lý chương trình đào tạo, quản lý điểm học sinh sinh viên, quản lý thư viện trong các trường đại học, cao đẳng và trung học phổ thông…

Quản lý sinh viên là một đề tài không còn mới mẻ với các quản lý bài toán. Việc đưa học vào ứng dụng để quản lý là rất hữu ích, vì chúng ta phải bỏ ra rất ít thời gian mà lại thu được hiệu quả cao, rất chính xác và tiện lợi nhanh chóng. Trong phạm vi bài kiểm tra nhóm, chúng tôi đã có thể gặp vấn đề “Quản lý sinh viên” ở trường ĐH Thủy Lợi bằng máy vi tính.

Với khoảng thời gian không nhiều, vừa phân tích thiết kế, nghiên cứu tìm hiểu khai thác ngôn ngữ mới, vừa thực hiện chương trình quả là khó khăn đối với chúng. Bởi “Quản lý sinh viên” là một chủ đề có nội dung rộng rãi, các mặt khác có khả năng hiểu được hệ thống của nhóm em vẫn còn nhiều hạn chế. Trong khoảng thời gian bùng nổ và chưa có kinh nghiệm trong thiết kế phần mền cũng như thiết kế các bối cảnh sơ đồ của chương trình vì vậy không tránh khỏi những sai sót. Rất mong nhận được sự hướng dẫn bổ sung của các thầy cô cùng lời khuyên của các bạn để chương trình được hoàn thiện hơn.

Trong suốt thời gian vừa qua chúng em cảm ơn cô Lê Thị Tú Kiên đã luôn giúp đỡ và đồng hành cùng nhóm để có một kết quả tốt nhất. Chúng em chúc cô thật nhiều sức khỏe và thành công trong công việc!

**MỤC LỤC**

[CHƯƠNG 1: THIẾT KẾ CƠ SỞ DỮ LIỆU 5](#_Toc187305355)

[**1.1** **Mô tả bài toán quản lý** 5](#_Toc187305356)

[**1.2** **Biểu đồ lớp thực thể** 5](#_Toc187305357)

[**1.3** **Ánh xạ CSDL quan hệ** 5](#_Toc187305358)

[**1.4** **Thiết kế chi tiết** 7](#_Toc187305359)

[**1.5 Nhập dữ liệu các bảng** 10](#_Toc187305360)

[CHƯƠNG 2: XÂY DỰNG CÁC ĐỐI TƯỢNG DỮ LIỆU 15](#_Toc187305361)

[**2.1 Xây dựng view** 15](#_Toc187305362)

[2.1.1. Nguyễn Thị Thu Hằng 15](#_Toc187305363)

[2.1.2: Hồ Thủy Tiên 16](#_Toc187305364)

[2.1.3 Khổng Thị Vân 18](#_Toc187305365)

[2.1.4 Vũ Hà Lâm 19](#_Toc187305366)

[2.1.5 Hoàng Khánh Hà 21](#_Toc187305367)

[**2.2 Xây dựng procedure** 22](#_Toc187305368)

[2.2.1 Nguyễn Thị Thu Hằng 22](#_Toc187305369)

[2.2.2 Hồ Thủy Tiên 24](#_Toc187305370)

[2.2.3 Khổng Thị Vân 27](#_Toc187305371)

[2.2.4 Vũ Hà Lâm 28](#_Toc187305372)

[2.2.5 Hoàng Khánh Hà 29](#_Toc187305373)

[**2.3 Xây dựng function** 30](#_Toc187305374)

[2.3.1 Nguyễn Thị Thu Hằng 30](#_Toc187305375)

[2.3.2 Hồ Thủy Tiên 31](#_Toc187305376)

[2.3.3 Khổng Thị Vân 33](#_Toc187305377)

[2.3.4 Vũ Hà Lâm 34](#_Toc187305378)

[2.3.5 Hoàng Khánh Hà 36](#_Toc187305379)

[**2.4 Xây dựng trigger** 37](#_Toc187305380)

[2.4.1: Nguyễn Thị Thu Hằng 37](#_Toc187305381)

[2.4.2: Hồ Thủy Tiên 38](#_Toc187305382)

[2.4.3: Khổng Thị Vân 40](#_Toc187305383)

[2.4.4: Vũ Hà Lâm 41](#_Toc187305384)

[2.4.5: Hoàng Khánh Hà 45](#_Toc187305385)

[**2.5. Xây dựng cursor** 47](#_Toc187305386)

[2.5.1. Nguyễn Thị Thu Hằng 47](#_Toc187305387)

[2.5.2. Hồ Thủy Tiên 48](#_Toc187305388)

[2.5.3. Khổng Thị Vân 49](#_Toc187305389)

[2.5.4. Vũ Hà Lâm 50](#_Toc187305390)

[2.5.5. Hoàng Khánh Hà 51](#_Toc187305391)

[CHƯƠNG 3: ỨNG DỤNG CÁC ĐỐI TƯỢNG VÀO PHẦN MỀM QUẢN LÝ 52](#_Toc187305392)

[**3.1 Mô tả chung chức năng phần mềm quản lý** 52](#_Toc187305393)

[**3.2 Mô tả ứng dụng một số đối tượng dữ liệu trong phần mềm** 52](#_Toc187305394)

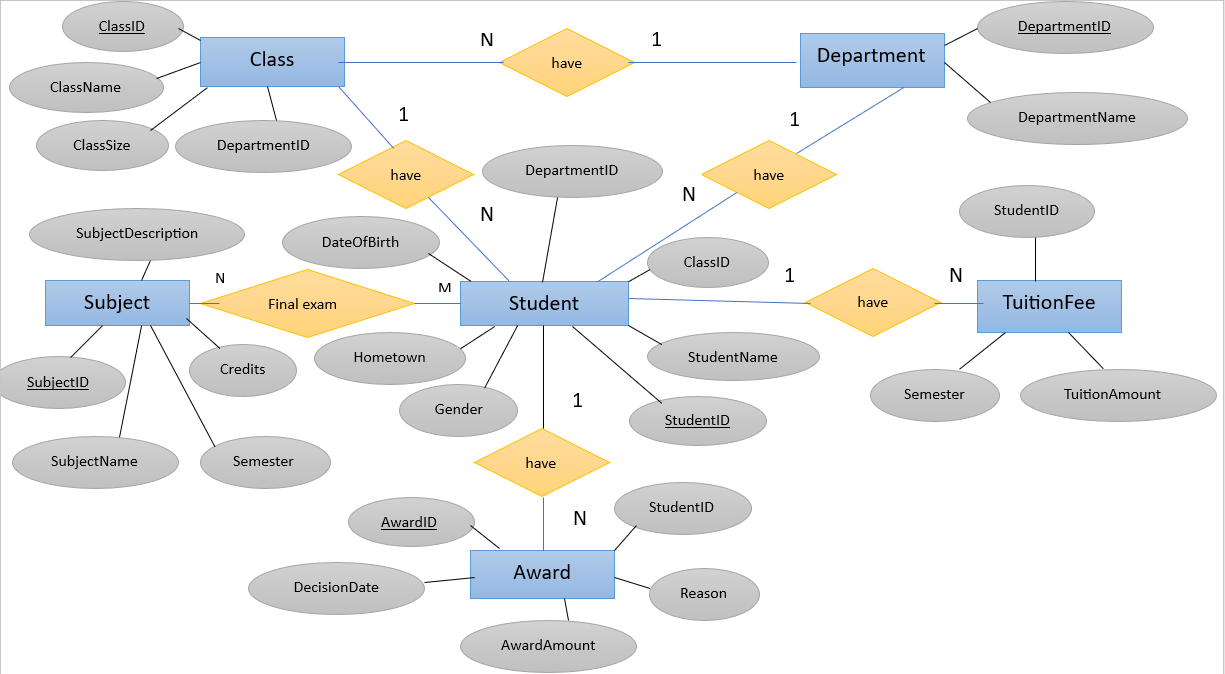
[**3.3 Demo chức năng giao diện ứng dụng** 54](#_Toc187305395)

# **CHƯƠNG 1: THIẾT KẾ CƠ SỞ DỮ LIỆU**

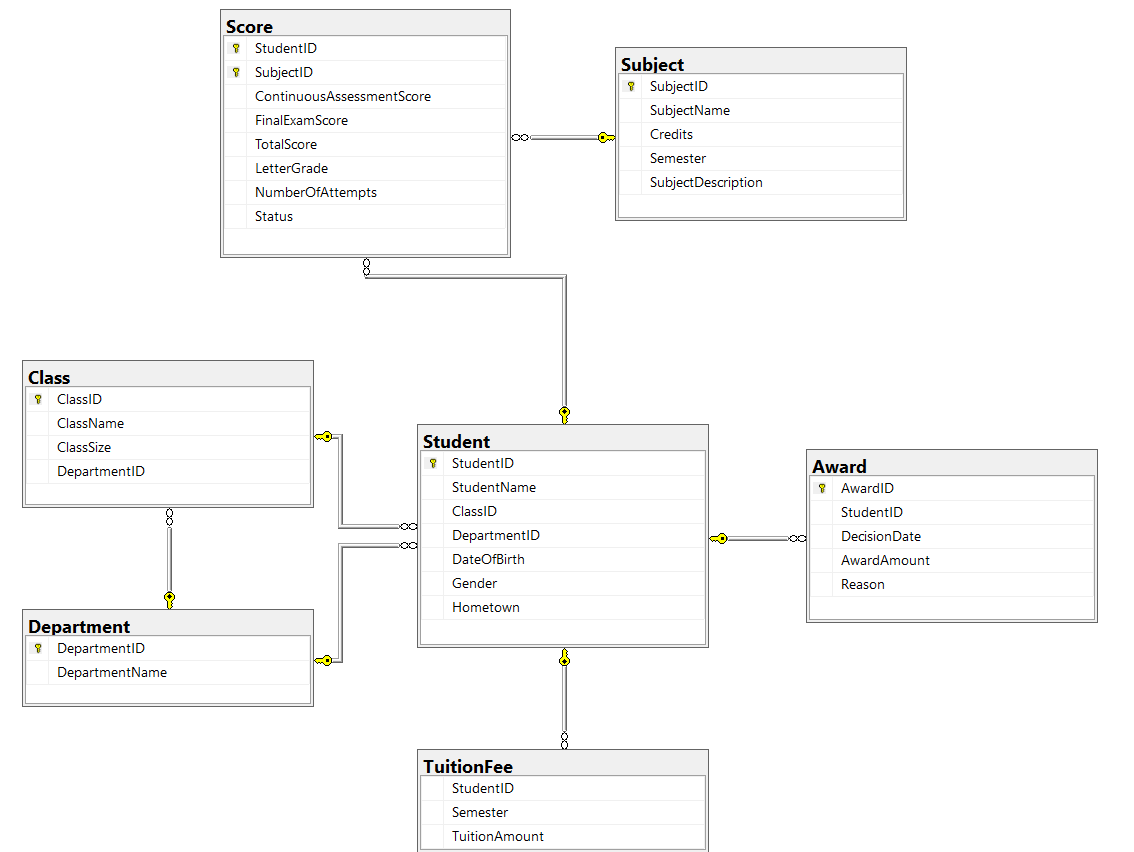
* 1. **Mô tả bài toán quản lý**

Một trường đại học đang muốn phát triển hệ thống quản lý sinh viên, không chỉ những sinh viên mới nhập học mà cả các sinh viên đang học theo học tại trường. Hệ thống này nhằm mục tiêu theo dõi và quản lý thông tin của sinh viên, từ khi nhập học đến khi tốt nghiệp. Thông tin này bao gồm các thông tin cá nhân, học tập, học phí, khen thưởng và các hoạt động đăng ký học. Sau khi nhập học xong, mỗi sinh viên đều được phát 1 tài khoản sinh viên và được xếp vào lớp và khoa như trước đó đã đăng ký học và vào hệ thống đăng ký học theo thời gian được định sẵn. Sau đó hệ thống sẽ hiển thị trạng thái đăng ký học. Ngoài ra hệ thống sẽ hiện những môn học và học phí đã đăng ký. Sau khi thi xong thì hệ thống sẽ tính điểm và cập nhật điểm trung bình môn học, trung bình học kì, đối với sinh viên khá giỏi thì sẽ được xét khen thưởng và học bổng.

* 1. **Biểu đồ lớp thực thể**

****

* 1. **Ánh xạ CSDL quan hệ**
* **Mô hình quan hệ**

****

* **Ánh xạ cơ sở dữ liệu**
* Student (**StudentID**, StudentName, ClassID, DepartmentID, DateOfBirth, Gender, Hometown)
* Department (**DepartmentID**, DepartmentName)
* Class (**ClassID**, ClassName, ClassSize, DepartmentID)
* TuitionFee (StudentID, Semester, TuitionAmout)
* Award (**AwardID**, StudentID, DecisionDate, AwardAmount, Reason)
* Subject (**SubjectID**, SubjectName, Credits, Semester, SubjectDecription)
* Score (StudentID, SubjectID, ContinousAssessmentScore, FinalExamScore, TotalScore, LetterGrade, NumberOfAttempts, Status)

* 1. **Thiết kế chi tiết**

1.4.1: Bảng Student – Thông tin về sinh viên

|  |  |  |  |
| --- | --- | --- | --- |
| Ràng buộc | Tên cột | Kiểu dữ liệu | Ghi chú |
| Khóa chính | StudentID | VARCHAR(10) | Mã sinh viên |
| Khóa ngoại | ClassID | VARCHAR(8) | Mã lớp |
| Khóa ngoại | DepartmentID | VARCHAR(8) | Mã khoa |
|  | StudentName | NVARCHAR(100) | Tên sinh viên |
|  | DateOfBirth | DATE | Ngày sinh |
|  | Gender | NVARCHAR(10) | Giới tính |
|  | Hometown | NVARCHAR(100) | Quê quán |

1.4.2: Bảng Class – Thông tin về lớp

|  |  |  |  |
| --- | --- | --- | --- |
| Ràng buộc | Tên cột | Kiểu dữ liệu | Ghi chú |
| Khóa chính | ClassID | VARCHAR(8) | Mã lớp |
| Khóa ngoại | DepartmentID | VARCHAR(8) | Mã khoa |
|  | ClassName | NVARCHAR(100) | Tên lớp |
|  | ClassSize | INT | Sĩ số |

1.4.3: Bảng Department – Thông tin về Khoa

|  |  |  |  |
| --- | --- | --- | --- |
| Ràng buộc | Tên cột | Kiểu dữ liệu | Ghi chú |
| Khóa chính | DepartmentID | VARCHAR(8) | Mã khoa |
|  | DepartmentName | NVARCHAR(100) | Tên khoa |

1.4.4: Bảng Subject – Thông tin về môn học

|  |  |  |  |
| --- | --- | --- | --- |
| Ràng buộc | Tên cột | Kiểu dữ liệu | Ghi chú |
| Khóa chính | Subject ID | VARCHAR(8) | Mã môn học |
|  | Subject Name | NVARCHAR(100) | Tên môn học |
|  | Credits | INT | Số tín chỉ |
|  | Semester | INT | Học kỳ |
|  | SubjectDescription | NVARCHAR(255) | Mô tả môn học |

1.4.5: Bảng Score – Thông tin về điểm

|  |  |  |  |
| --- | --- | --- | --- |
| Ràng buộc | Tên cột | Kiểu dữ liệu | Ghi chú |
| Khóa ngoại | StudentID | VARCHAR(10) | Mã sinh viên |
| Khóa ngoại | Subject ID | VARCHAR(8) | Mã môn học |
|  | ContinousAssessmentScore | FLOAT | Điểm quá trình |
|  | FinalExamScore | FLOAT | Điểm thi cuối kỳ |
|  | TotalScore | FLOAT | Tổng điểm |
|  | LetterGrade | NVARCHAR(2) | Điểm chữ |
|  | NumberOfAttempts | INT | Số lần thi |
|  | Status | NVARCHAR(20) | Trạng thái môn học |

1.4.6: Bảng TuitionFee – Thông tin về học phí

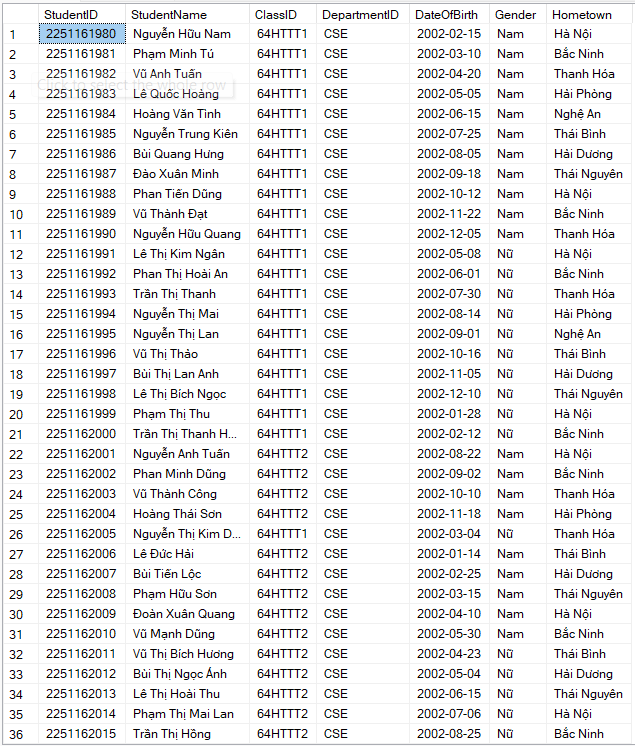
|  |  |  |  |
| --- | --- | --- | --- |
| Ràng buộc | Tên cột | Kiểu dữ liệu | Ghi chú |
| Khóa ngoại | StudentID | VARCHAR(10) | Mã sinh viên |
|  | Semester | INT | Học kỳ |
|  | TuitionAmount | FLOAT | Số tiền học phí |

1.4.7: Bảng Award – Thông tin về khen thưởng

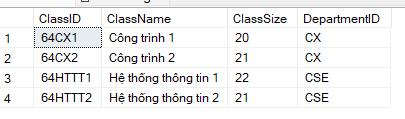
|  |  |  |  |
| --- | --- | --- | --- |
| Ràng buộc | Tên cột | Kiểu dữ liệu | Ghi chú |
| Khóa chính | AwardID | VARCHAR(8) | Mã khen thưởng |
| Khóa ngoại | StudentID | VARCHAR(10) | Mã sinh viên |
|  | DecisionDate | DATE | Ngày quyết định |
|  | AwardAmount | NVARCHAR(50) | Mức khen thưởng |
|  | Reason | NVARCHAR(255) | Lý do khen thưởng |

## **1.5 Nhập dữ liệu các bảng**

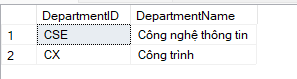
1.5.1: Bảng Student



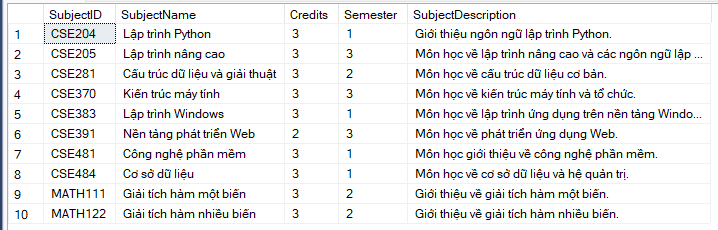
1.5.2: Bảng Class



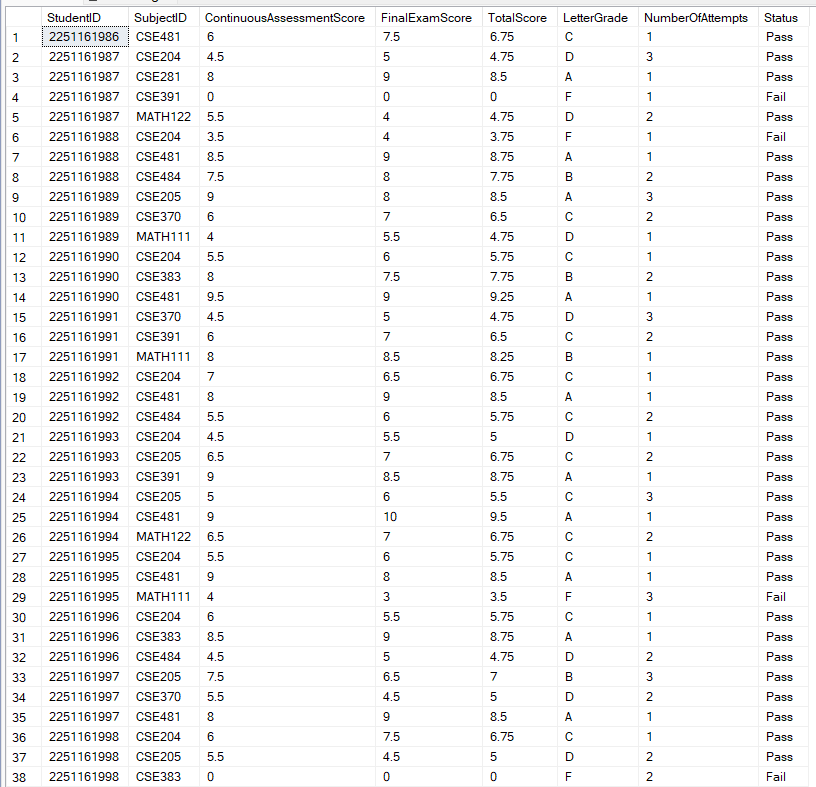
1.5.3: Bảng Department



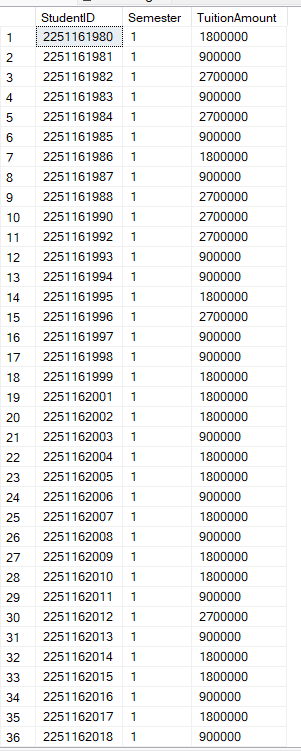
1.5.4: Bảng Subject



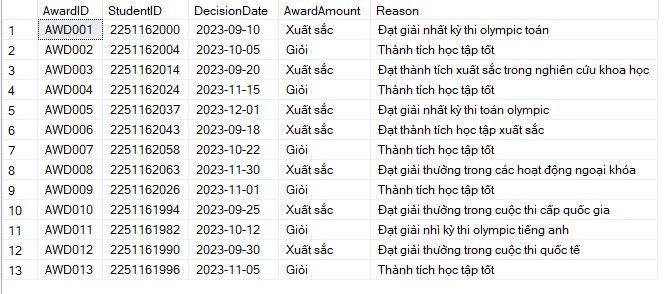
1.5.5: Bảng Score



1.5.6: Bảng TuitionFee



1.5.7: Bảng Award



# **CHƯƠNG 2: XÂY DỰNG CÁC ĐỐI TƯỢNG DỮ LIỆU**

## **2.1 Xây dựng view**

### **2.1.1. Nguyễn Thị Thu Hằng (view Scholarship, view StudentClassInfo)**

* **View scholarship**: Hiển thị danh sách sinh viên đạt học bổng và số tiền nhận được trong kỳ 1 (điều kiện đạt học bổng là điểm trung bình các môn >= 8.5 và số tín chỉ học tối thiểu là 6 tín)

|  |
| --- |
| **CREATE VIEW Scholarship**  **AS**  **SELECT**  **S.StudentID,**  **S.StudentName,**  **S.Gender,**  **AVG(SC.TotalScore) AS DiemTrungBinh,**  **SUM(SB.Credits) AS TongTinChi,**  **TF.TuitionAmount,**  **(TF.TuitionAmount + (TF.TuitionAmount \* 0.2)) AS TienHocBong**  **FROM**  **Student S**  **JOIN**  **Score SC ON S.StudentID = SC.StudentID**  **JOIN**  **[Subject] SB ON SC.SubjectID = SB.SubjectID**  **JOIN**  **TuitionFee TF ON S.StudentID = TF.StudentID**  **WHERE**  **SB.Semester =**  **AND TF.Semester = 1**  **GROUP BY**  **S.StudentID, S.StudentName, S.Gender, TF.TuitionAmount**  **HAVING**  **AVG(SC.TotalScore) >= 8.5**  **AND SUM(SB.Credits) >= 6;**  Kết quả: |

* **View StudentClassInfo**: Hiển thị thông tin sinh viên và lớp học

CREATE VIEW StudentClassInfo AS

SELECT

S.StudentID,

S.StudentName,

C.ClassName,

D.DepartmentName

FROM

Student S

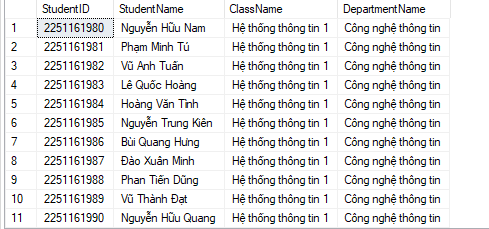
JOIN

Class C ON S.ClassID = C.ClassID

JOIN

Department D ON S.DepartmentID = D.DepartmentID;

Kết quả:



### **2.1.2: Hồ Thủy Tiên (View StudentCredits, view DepartmentStudentCount)**

* **View StudentCredits**: Đưa ra mã sinh viên, tên sinh viên, lớp và tổng số tín chỉ đã học của sinh viên lớp 64CX1

|  |
| --- |
| **CREATE VIEW StudentCredits AS**  **SELECT**  **s.StudentID,**  **s.StudentName,**  **s.ClassID,**  **SUM(sub.Credits) AS TotalCredits**  **FROM**  **Student s**  **INNER JOIN Score sc ON s.StudentID = sc.StudentID**  **INNER JOIN [Subject] sub ON sc.SubjectID = sub.SubjectID**  **WHERE**  **s.ClassID = '64CX1'**  **GROUP BY**  **s.StudentID, s.StudentName, s.ClassID** |

Kết quả:



* **View DepartmentStudentCount** : hiển thị tổng số sinh viên của từng khoa

CREATE VIEW DepartmentStudentCount AS

SELECT

D.DepartmentID,

D.DepartmentName,

COUNT(S.StudentID) AS TotalStudents

FROM

Department D

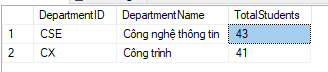
LEFT JOIN

Student S ON D.DepartmentID = S.DepartmentID

GROUP BY

D.DepartmentID, D.DepartmentName;

Kết quả:



### **2.1.3 Khổng Thị Vân (View SinhVienDiemCaoNhat, View StudentScoreInfo)**

* **View SinhVienDiemCaoNhat**: Hiển thị danh sách các sinh viên có điểm tổng cao nhất trong từng môn học.

|  |
| --- |
| **CREATE VIEW SinhVienDiemCaoNhat AS**  **SELECT**  **S.StudentID,**  **S.StudentName,**  **SB.SubjectName,**  **SC.TotalScore**  **FROM**  **Student S**  **JOIN**  **Score SC ON S.StudentID = SC.StudentID**  **JOIN**  **[Subject] SB ON SC.SubjectID = SB.SubjectID**  **JOIN**  **( SELECT**  **SubjectID,**  **MAX(TotalScore) AS MaxTotalScore**  **FROM**  **Score**  **GROUP BY**  **SubjectID**  **) MS ON SC.SubjectID = MS.SubjectID AND SC.TotalScore = MS.MaxTotalScore;** |

Kết quả:



* **View StudentScoreInfo**: Hiển thị điểm chi tiết của sinh viên

CREATE VIEW StudentScoreInfo AS

SELECT

S.StudentID,

S.StudentName,

SB.SubjectName,

SC.TotalScore,

SC.LetterGrade

FROM

Student S

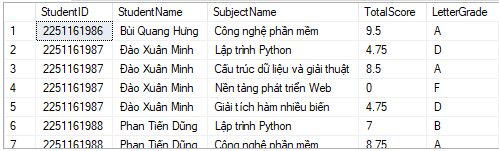
JOIN

Score SC ON S.StudentID = SC.StudentID

JOIN

[Subject] SB ON SC.SubjectID = SB.SubjectID;

Kết quả:

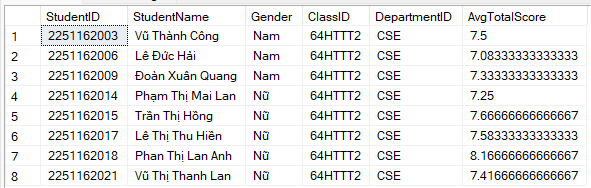


### **2.1.4 Vũ Hà Lâm (View HighAverageScore**, **View TuitionInfo)**

* **View HighAverageScore**: Hiển thị thông tin danh sách sinh viên có điểm trung bình môn học của sinh viên lớp 64HTTT2 > 7

|  |
| --- |
| **CREATE VIEW HighAverageScore AS**  **SELECT**  **st.StudentID,**  **st.StudentName,**  **st.Gender,**  **st.ClassID,**  **st.DepartmentID,**  **AVG(sc.TotalScore) AS AvgTotalScore**  **FROM Student st JOIN Score sc**  **ON st.StudentID = sc.StudentID**  **WHERE st.ClassID = '64HTTT2'**  **GROUP BY st.StudentID,**  **st.StudentName,**  **st.Gender,**  **st.ClassID,**  **st.DepartmentID**  **HAVING AVG(sc.TotalScore) > 7** |

Kết quả:



* **View TuitionInfo**: Hiển thị học phí đã đóng của từng sinh viên

CREATE VIEW TuitionInfo AS

SELECT

S.StudentID,

S.StudentName,

T.Semester,

T.TuitionAmount

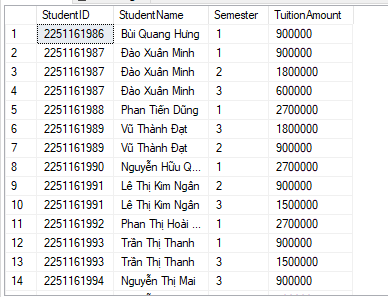
FROM

Student S

JOIN

TuitionFee T ON S.StudentID = T.StudentID;

Kết quả:



### **2.1.5 Hoàng Khánh Hà(View StudentAttemptSummaryView, View AwardedStudents)**

* **View StudentAttemptSummaryView** : Hiển thị thông tin sinh viên học môn học có mã CSE484

|  |
| --- |
| **CREATE VIEW StudentAttemptSummaryView AS**  **SELECT**  **s.StudentID,**  **st.StudentName,**  **s.SubjectID,**  **st.ClassID,**  **st.DepartmentID,**  **s.NumberOfAttempts**  **FROM Score s JOIN**  **Student st ON s.StudentID = st.StudentID**  **WHERE s.SubjectID = 'CSE484'** |

Kết quả:



* **View AwardedStudents**: Hiển thị những sinh viên được nhận thưởng

CREATE VIEW AwardedStudents AS

SELECT

S.StudentID,

S.StudentName,

A.Reason,

A.AwardAmount,

A.DecisionDate

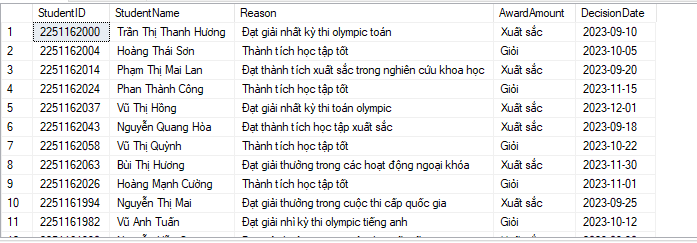
FROM

Student S

JOIN

Award A ON S.StudentID = A.StudentID;

Kết quả:



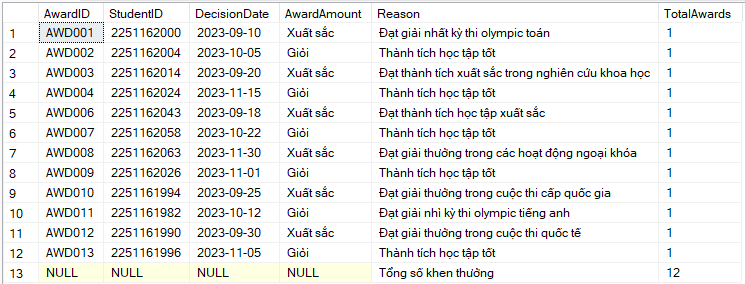
**2.2 Xây dựng procedure**

### **2.2.1 Nguyễn Thị Thu Hằng(Proc GetAwardByDate, Proc GetStudentsWithFailedSubjects)**

* **Proc GetAwardByDate**: Thống kê danh sách và tổng số khen thưởng trong một khoảng thời gian nhất định

|  |
| --- |
| **CREATE PROCEDURE GetAwardByDate**  **@StartDate DATE,**  **@EndDate DATE**  **AS**  **BEGIN**  **SELECT**  **A.AwardID,**  **A.StudentID,**  **A.DecisionDate,**  **A.AwardAmount,**  **A.Reason,**  **COUNT(A.AwardID) AS TotalAwards**  **FROM Award A**  **WHERE A.DecisionDate BETWEEN @StartDate AND @EndDate**  **GROUP BY A.AwardID, A.StudentID, A.DecisionDate, A.AwardAmount, A.Reason**  **UNION**  **SELECT**  **NULL AS AwardID,**  **NULL AS StudentID,**  **NULL AS DecisionDate,**  **NULL AS AwardAmount,**  **N'Tổng số khen thưởng' AS Reason,**  **COUNT(A.AwardID) AS TotalAwards**  **FROM Award A**  **WHERE A.DecisionDate BETWEEN @StartDate AND @EndDate;**  **END;**  **-- Thực thi procedure**  **EXEC GetAwardByDate**  **@StartDate = '2023-08-01',**  **@EndDate = '2023-11-30';** |

Kết quả:



* **Proc GetStudentsWithFailedSubjects:** trả về danh sách sinh viên kèm thông tin môn học mà họ chưa đạt yêu cầu (điểm < 4.0)

CREATE PROCEDURE GetStudentsWithFailedSubjects

AS

BEGIN

SELECT

S.StudentID,

S.StudentName,

SJ.SubjectName,

SC.TotalScore

FROM

Student S

JOIN Score SC ON S.StudentID = SC.StudentID

JOIN Subject SJ ON SC.SubjectID = SJ.SubjectID

WHERE

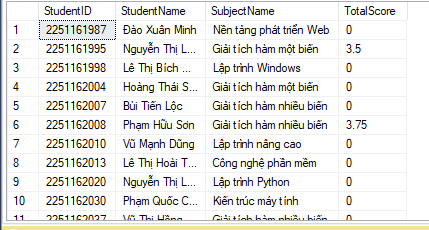
SC.TotalScore < 4.0;

END;

--Thực thi:

EXEC GetStudentsWithFailedSubjects

Kết quả:

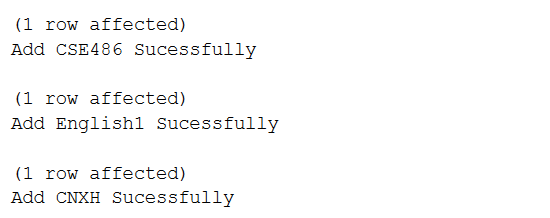


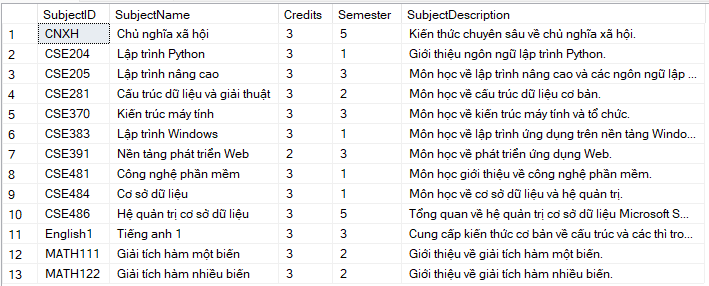
### **2.2.2 Hồ Thủy Tiên ( Proc AddSubject, Proc GetFailedStudentsInDBCourse)**

* **Proc AddSubject**: Thủ tục thêm môn học mới vào bảng môn học, kiểm tra môn học đã tồn tại chưa. Tín chỉ nằm trong khoảng 1-10, học kỳ nằm trong khoảng 1-8 và mô tả môn học không được để trống.

|  |
| --- |
| **CREATE PROC AddSubject**  **@SubjectID VARCHAR(8),**  **@SubjectName NVARCHAR(100),**  **@Credits INT,**  **@Semester INT,**  **@SubjectDescription NVARCHAR(255)**  **AS**  **BEGIN**    **IF EXISTS (SELECT 1 FROM [Subject]**  **WHERE SubjectID = @SubjectID)**  **BEGIN**  **PRINT 'Course already exists.'**  **RETURN**  **END**  **IF @Credits < 1 OR @Credits > 10**  **BEGIN**  **PRINT 'Credits must be between 1 and 10.'**  **RETURN**  **END**    **IF @Semester < 1 OR @Semester > 8**  **BEGIN**  **PRINT 'Semester must be between 1 and 8.'**  **RETURN**  **END**    **IF @SubjectDescription IS NULL OR LEN(@SubjectDescription) = 0**  **BEGIN**  **PRINT 'Subject description cannot be empty.'**  **RETURN**  **END**  **INSERT INTO [Subject] (SubjectID, SubjectName, Credits, Semester, SubjectDescription)**  **VALUES (@SubjectID, @SubjectName, @Credits, @Semester, @SubjectDescription)**  **PRINT 'Add ' + @SubjectID + ' Sucessfully'**  **END**  **GO**  **-- Thực thi**  **Exec AddSubject 'CSE486', N'Hệ quản trị cơ sở dữ liệu',3,5,N'Tổng quan về hệ quản trị cơ sở dữ liệu Microsoft SQL Server.';**  **Exec AddSubject 'English1', N'Tiếng anh 1',3,3,N'Cung cấp kiến thức cơ bản về cấu trúc và các thì trong tiếng anh.';**  **Exec AddSubject 'CNXH', N'Chủ nghĩa xã hội',3,5,N'Kiến thức chuyên sâu về chủ nghĩa xã hội.';** |

Kết quả:





* **Proc GetFailedStudentsInDBCourse**: trả về danh sách sinh viên trượt môn CNPM, gọi thủ tục đó để in ra họ tên

alter PROCEDURE GetFailedStudentsInDBCourse

AS

BEGIN

SELECT

S.StudentID,

S.StudentName

FROM

Student S

JOIN Score SC ON S.StudentID = SC.StudentID

JOIN Subject SJ ON SC.SubjectID = SJ.SubjectID

WHERE

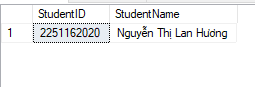
SJ.SubjectName = N'Lập trình Python' AND SC.[Status] = 'Fail';

END;

--Thực thi:

exec GetFailedStudentsInDBCourse

Kết quả



### **2.2.3 Khổng Thị Vân (Proc sp\_CapNhatDiem, Proc AddNewStudent)**

* **Proc sp\_CapNhatDiem**: Procedure cập nhật điểm sinh viên.

|  |
| --- |
| **CREATE PROCEDURE sp\_CapNhatDiem**  **@StudentID VARCHAR(10),**  **@SubjectID VARCHAR(8),**  **@ContinuousAssessmentScore FLOAT,**  **@FinalExamScore FLOAT**  **AS**  **BEGIN**  **DECLARE @TotalScore FLOAT;**  **DECLARE @LetterGrade NVARCHAR(2);**  **DECLARE @Status VARCHAR(20);**  **DECLARE @StudentExists INT;**  **SELECT @StudentExists = COUNT(\*)**  **FROM Student**  **WHERE StudentID = @StudentID;**  **BEGIN**  **PRINT 'StudentID không tồn tại';**  **RETURN;**  **end;**  **SET @TotalScore = (@ContinuousAssessmentScore \* 0.4) + (@FinalExamScore \* 0.6);**  **SET @LetterGrade = CASE**  **WHEN @TotalScore >= 8.5 AND @TotalScore <= 10 THEN 'A'**  **WHEN @TotalScore >= 7 AND @TotalScore < 8.5 THEN 'B'**  **WHEN @TotalScore >= 5.5 AND @TotalScore < 7 THEN 'C'**  **WHEN @TotalScore >= 4 AND @TotalScore < 5.5 THEN 'D'**  **ELSE 'F'**  **END;**  **SET @Status = CASE**  **WHEN @TotalScore < 4 THEN 'Fail'**  **ELSE 'Pass'**  **END;**  **UPDATE Score**  **SET ContinuousAssessmentScore = @ContinuousAssessmentScore,**  **FinalExamScore = @FinalExamScore,**  **TotalScore = @TotalScore,**  **LetterGrade = @LetterGrade,**  **[Status] = @Status**  **WHERE StudentID = @StudentID AND SubjectID = @SubjectID;**  **END;**  **GO**  **-- Thực thi**  **EXEC sp\_CapNhatDiem**  **@StudentID = '2251161985',**  **@SubjectID = 'CSE481',**  **@ContinuousAssessmentScore = 8.6,**  **@FinalExamScore = 10.0;**  **SELECT \* FROM Score** |

Kết quả:



* **Proc AddNewStudent** : Thêm sinh viên mới

CREATE PROCEDURE AddNewStudent (

@StudentID VARCHAR(10),

@StudentName NVARCHAR(100),

@ClassID VARCHAR(8),

@DepartmentID VARCHAR(8),

@DateOfBirth DATE,

@Gender NVARCHAR(10),

@Hometown NVARCHAR(100)

)

AS

BEGIN

INSERT INTO Student (StudentID, StudentName, ClassID, DepartmentID, DateOfBirth, Gender, Hometown)

VALUES (@StudentID, @StudentName, @ClassID, @DepartmentID, @DateOfBirth, @Gender, @Hometown);

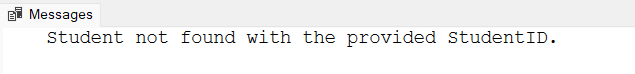
END;

### **2.2.4 Vũ Hà Lâm ( Proc UpdateStudent, Proc DeleteStudent)**

* **Proc UpdateStudent**: Procudure cập nhật tên sinh viên và ngày sinh của sinh viên

|  |
| --- |
| **CREATE PROC UpdateStudent**  **@StudentID VARCHAR(10),**  **@StudentName NVARCHAR(100),**  **@DateOfBirth DATE**  **AS**  **BEGIN**  **IF EXISTS (SELECT 1 FROM Student WHERE StudentID = @StudentID)**  **BEGIN**  **UPDATE Student**  **SET StudentName = @StudentName,**  **DateOfBirth = @DateOfBirth**  **WHERE StudentID = @StudentID;**  **PRINT 'Student information updated successfully.';**  **END**  **ELSE**  **BEGIN**  **PRINT 'Student not found with the provided StudentID.';**  **END**  **END;**  **-- Thực thi**  **EXEC UpdateStudent**  **@StudentID = '2251162900',**  **@StudentName = 'Vũ Chung Kiên',**  **@DateOfBirth = '2000-04-09'** |

Kết quả:



* **Proc DeleteStudent**: Xóa sinh viên

CREATE PROCEDURE DeleteStudent (@StudentID VARCHAR(10))

AS

BEGIN

DELETE FROM Student

WHERE StudentID = @StudentID;

END;

### **2.2.5 Hoàng Khánh Hà (Proc GetStudentInfo, Proc GetStudentsByDepartment)**

* **Proc GetStudentInfo**: Nhập mã sinh viên hiển thị thông tin sinh viên gồm tên, lớp , khoa

|  |
| --- |
| **CREATE PROC GetStudentInfo**  **@StudentID VARCHAR(10)**  **AS**  **BEGIN**    **IF EXISTS (SELECT 1 FROM Student WHERE StudentID = @StudentID)**  **BEGIN**    **SELECT**  **S.StudentName AS [Student Name],**  **C.ClassName AS [Class],**  **D.DepartmentName AS [Department]**  **FROM Student S**  **JOIN Class C ON S.ClassID = C.ClassID**  **JOIN Department D ON S.DepartmentID = D.DepartmentID**  **WHERE S.StudentID = @StudentID;**  **END**  **ELSE**  **BEGIN**  **PRINT 'Student not found with the provided StudentID.';**  **END**  **END;**  **GO**  **exec GetStudentInfo '2251162034'** |

Kết quả:



* **Proc GetStudentsByDepartment: Lấy danh sách sinh viên theo khoa**

CREATE PROCEDURE GetStudentsByDepartment1 (@DepartmentID VARCHAR(8))

AS

BEGIN

SELECT StudentID, StudentName

FROM Student

WHERE DepartmentID = @DepartmentID;

END;

## **2.3 Xây dựng function**

### **2.3.1 Nguyễn Thị Thu Hằng (func fnCalculateGPA, func TotalTuition)**

* **Func fnCalculateGPA:** tính đểm gpa của sinh viên theo mã sinh viên

|  |
| --- |
| **CREATE FUNCTION fnCalculateGPA (**  **@StudentID VARCHAR(10)**  **)**  **RETURNS FLOAT**  **AS**  **BEGIN**  **DECLARE @TotalScore FLOAT = 0, @TotalCredits INT = 0, @GPA FLOAT;**    **SELECT @TotalScore = SUM(s.TotalScore \* sub.Credits),**  **@TotalCredits = SUM(sub.Credits)**  **FROM Score s**  **JOIN [Subject] sub ON s.SubjectID = sub.SubjectID**  **WHERE s.StudentID = @StudentID;**  **IF @TotalCredits = 0**  **SET @GPA = 0;**  **ELSE**  **SET @GPA = @TotalScore / @TotalCredits;**  **RETURN @GPA;**  **END;** |

Kết quả:



* **Func TotalTuition**: Tổng học phí của sinh viên

CREATE FUNCTION TotalTuition (@StudentID VARCHAR(10))

RETURNS FLOAT

AS

BEGIN

DECLARE @Total FLOAT;

SELECT @Total = SUM(TuitionAmount)

FROM TuitionFee

WHERE StudentID = @StudentID;

RETURN @Total;

END;

print dbo.TotalTuition ('2251161981')

### **2.3.2 Hồ Thủy Tiên (Func TTSV, Func TotalCredits)**

* **Func TTSV**: Hàm đưa ra chi tiết về môn học, tín chỉ và học phí từng môn của sinh viên theo kì

|  |
| --- |
| **CREATE FUNCTION TTSV**  **(@StudentID VARCHAR(10), @Semester INT)**  **RETURNS TABLE**  **AS**  **RETURN**  **(**  **SELECT**  **s.SubjectID,**  **s.SubjectName,**  **s.Credits,**  **tf.StudentID,**  **tf.Semester,**  **(s.Credits\*300000) AS Tuition**  **FROM**  **[Subject] s**  **JOIN**  **Score sc ON s.SubjectID = sc.SubjectID**  **JOIN**  **TuitionFee tf ON sc.StudentID = tf.StudentID AND s.Semester = tf.Semester**  **WHERE**  **tf.StudentID = @StudentID AND tf.Semester = @Semester**  **);**  **GO**  **Select \* from TTSV('2251162048', 3)** |

Kết quả:



* **Func TotalCredits:** Tổng số tín chỉ của sinh viên đã học

CREATE FUNCTION TotalCredits (@StudentID VARCHAR(10))

RETURNS INT

AS

BEGIN

DECLARE @Credits INT;

SELECT @Credits = SUM(Credits)

FROM [Subject] SB

JOIN Score SC ON SB.SubjectID = SC.SubjectID

WHERE SC.StudentID = @StudentID;

RETURN @Credits;

END;

print dbo.TotalCredits ('2251161988')

Kết quả:

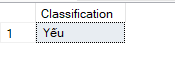


### **2.3.3 Khổng Thị Vân (Func ClassifyStudentByTotalScore, Func GetStudentCountBySubject)**

* **Func ClassifyStudentByTotalScore**: Function phân loại sinh viên theo điểm trung bình

|  |
| --- |
| **CREATE FUNCTION ClassifyStudentByTotalScore(@StudentID VARCHAR(10))**  **RETURNS NVARCHAR(20)**  **AS**  **BEGIN**  **DECLARE @TotalScore FLOAT;**  **DECLARE @Classification NVARCHAR(20);**  **SELECT @TotalScore = SUM(TotalScore \* Credits) / SUM(Credits)**  **FROM Score**  **JOIN [Subject] ON Score.SubjectID = [Subject].SubjectID**  **WHERE StudentID = @StudentID;**  **IF @TotalScore >= 8.5**  **SET @Classification = N'Giỏi';**  **ELSE IF @TotalScore >= 7.0**  **SET @Classification = N'Khá';**  **ELSE IF @TotalScore >= 5.5**  **SET @Classification = N'Trung bình';**  **ELSE**  **SET @Classification = N'Yếu';**  **RETURN @Classification;**  **END;**  **SELECT dbo.ClassifyStudentByTotalScore('2251161987') AS [Classification];** |

Kết quả:



* **Func GetStudentCountBySubject:** trả về số lượng sinh viên đã thi một môn học cụ thể

CREATE FUNCTION GetStudentCountBySubject (@SubjectID VARCHAR(8))

RETURNS INT

AS

BEGIN

DECLARE @StudentCount INT;

SELECT

@StudentCount = COUNT(DISTINCT SC.StudentID)

FROM

Score SC

WHERE

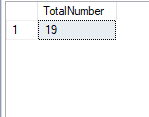
SC.SubjectID = @SubjectID;

RETURN @StudentCount;

END;

--sử dụng hàm:

select dbo.GetStudentCountBySubject('CSE484') as TotalNumber

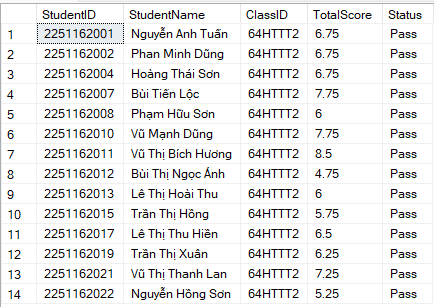


### **2.3.4 Vũ Hà Lâm( Func dbo.GetPassedStudents, Func IsStudentPassed)**

* **Func dbo.GetPassedStudents**: trả về những sinh viên đã PASS môn CSE204 của lớp 64httt2

|  |
| --- |
| **CREATE FUNCTION dbo.GetPassedStudents**  **(**  **@SubjectID VARCHAR(8),**  **@ClassID VARCHAR(8)**  **)**  **RETURNS TABLE**  **AS**  **RETURN**  **(**  **SELECT st.StudentID, st.StudentName, st.ClassID, s.TotalScore, s.Status**  **FROM Student st**  **JOIN Score s ON st.StudentID = s.StudentID**  **WHERE s.SubjectID = @SubjectID**  **AND st.ClassID = @ClassID**  **AND s.Status = 'Pass'**  **)**  **SELECT \***  **FROM dbo.GetPassedStudents('CSE204', '64HTTT2');** |

Kết quả:



* **Func IsStudentPassed** : Kiểm tra sinh viên có đỗ hay không

CREATE FUNCTION IsStudentPassed (@StudentID VARCHAR(10))

RETURNS BIT

AS

BEGIN

DECLARE @Result BIT;

IF EXISTS (SELECT \* FROM Score WHERE StudentID = @StudentID AND TotalScore < 5)

SET @Result = 0;

ELSE

SET @Result = 1;

RETURN @Result;

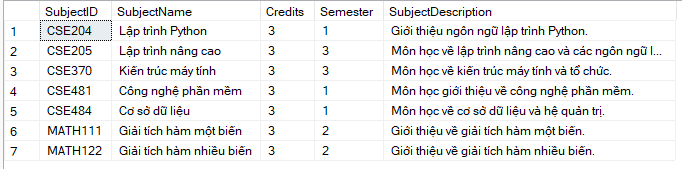
END;

### **2.3.5 Hoàng Khánh Hà( Func dbo.GetSubjectsByClass, Func GetStudentCount)**

* **Func dbo.GetSubjectsByClass:** Trả về danh sách môn học của lớp 64HTTT2

|  |
| --- |
| **CREATE FUNCTION dbo.GetSubjectsByClass**  **(**  **@ClassID VARCHAR(8)**  **)**  **RETURNS TABLE**  **AS**  **RETURN**  **(**  **SELECT DISTINCT s.SubjectID, sb.SubjectName, sb.Credits, sb.Semester, sb.SubjectDescription**  **FROM Student st**  **JOIN Score s ON st.StudentID = s.StudentID**  **JOIN [Subject] sb ON s.SubjectID = sb.SubjectID**  **WHERE st.ClassID = @ClassID**  **)**  **GO**  **SELECT \* FROM dbo.GetSubjectsByClass('64HTTT2');** |

Kết quả:



* **Func GetStudentCount :** Lấy số lượng sinh viên lớp

CREATE FUNCTION GetStudentCount (@ClassID VARCHAR(8))

RETURNS INT

AS

BEGIN

DECLARE @Count INT;

SELECT @Count = COUNT(StudentID)

FROM Student

WHERE ClassID = @ClassID;

RETURN @Count;

END;

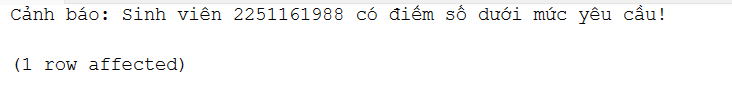
## **2.4 Xây dựng trigger**

### **2.4.1: Nguyễn Thị Thu Hằng (Trigger trgSendWarning, Trigger CheckValidScore)**

* **Trigger trgSendWarning**: Cập nhật điểm thi hết môn của sinh viên và sẽ gửi thông báo cảnh báo khi sinh viên có điểm hết môn < 4.

|  |
| --- |
| **CREATE TRIGGER trgSendWarning**  **ON Score**  **AFTER UPDATE**  **AS**  **BEGIN**  **DECLARE @StudentID VARCHAR(10);**  **DECLARE @NewScore FLOAT;**  **SELECT @StudentID = i.StudentID, @NewScore = i.TotalScore**  **FROM inserted i**  **WHERE i.TotalScore < 4.0;**  **IF @NewScore < 4.0**  **BEGIN**  **PRINT N'Cảnh báo: Sinh viên ' + @StudentID + N' có điểm số dưới mức yêu cầu!';**  **END**  **END;**  **UPDATE Score**  **SET TotalScore = 3.75**  **WHERE StudentID = '2251161988' AND SubjectID = 'CSE204';** |

Kết quả:



* **Trigger CheckValidScore**: kiểm tra điểm số hợp lệ trước khi chèn hoặc cập nhật

CREATE TRIGGER CheckValidScore

ON Score

AFTER INSERT

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE ContinuousAssessmentScore < 0 OR ContinuousAssessmentScore > 10 OR FinalExamScore < 0 OR FinalExamScore > 10)

BEGIN

PRINT'Scores must be between 0 and 10.'

ROLLBACK TRANSACTION;

END;

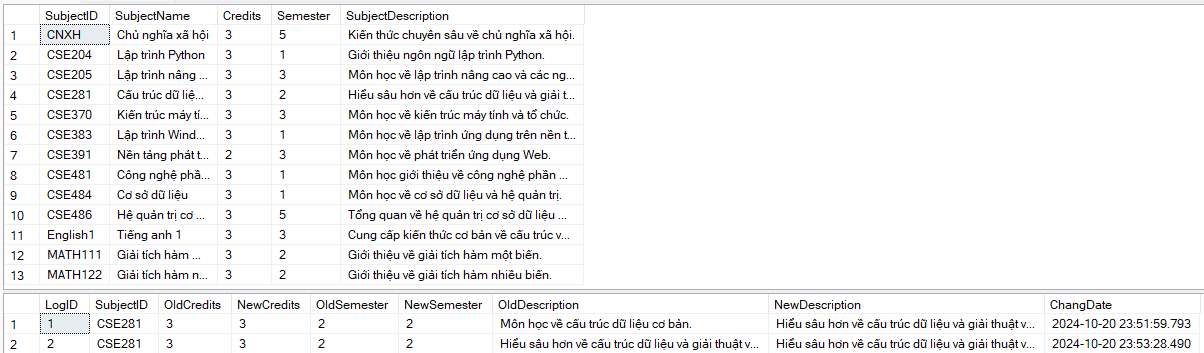
END;

### **2.4.2: Hồ Thủy Tiên (Trigger UpdateLog, Trigger UpdateClassSize)**

* **Trigger UpdateLog**: Trigger cập nhật Số tín, kì học, mô tả của các môn học

|  |
| --- |
| CREATE TABLE UpdateLog (  LogID INT IDENTITY(1,1) PRIMARY KEY,  SubjectID VARCHAR(8),  OldCredits INT,  NewCredits INT,  OldSemester INT,  NewSemester INT,  OldDescription NVARCHAR(255),  NewDescription NVARCHAR(255),  ChangDate DATETIME  );  GO  ALTER TRIGGER trgAfterUpdateSubject  ON [Subject]  AFTER UPDATE  AS  BEGIN  IF (SELECT COUNT(\*) FROM [Subject] s INNER JOIN inserted i ON s.SubjectID = i.SubjectID) > 0  BEGIN  -- Kiểm tra các điều kiện hợp lệ cho số tín chỉ, kỳ học và mô tả môn học  IF (SELECT COUNT(\*) FROM inserted i  WHERE i.Credits BETWEEN 1 AND 10  AND i.Semester BETWEEN 1 AND 8  AND i.SubjectDescription IS NOT NULL  AND i.SubjectDescription <> '') > 0  BEGIN  -- Thực hiện lưu vào bảng UpdateLog  INSERT INTO UpdateLog (SubjectID, OldCredits, NewCredits, OldSemester, NewSemester, OldDescription, NewDescription, ChangDate)  SELECT d.SubjectID, d.Credits, i.Credits, d.Semester, i.Semester, d.SubjectDescription, i.SubjectDescription, GETDATE()  FROM deleted d  JOIN inserted i ON d.SubjectID = i.SubjectID;  -- Cập nhật bảng Subject  UPDATE [Subject]  SET Credits = i.Credits,  Semester = i.Semester,  SubjectDescription = i.SubjectDescription  FROM inserted i  WHERE [Subject].SubjectID = i.SubjectID;  END  ELSE  BEGIN  RAISERROR ('Dữ liệu không hợp lệ: Số tín chỉ phải nằm trong khoảng 1-10,  kỳ học phải nằm trong khoảng 1-8 và mô tả môn học không được để trống.', 16, 1);  ROLLBACK TRAN;  END  END  ELSE  BEGIN  RAISERROR ('Môn học này không tồn tại, không thể cập nhật.', 16, 1);  ROLLBACK TRAN;  END  END;  **Select \* from UpdateLog** |

Kết quả:



* **Trigger UpdateClassSize:** cập nhật số lương sinh viên trong lớp khi thêm sinh viên

CREATE TRIGGER UpdateClassSize

ON Student

AFTER INSERT

AS

BEGIN

UPDATE Class

SET ClassSize = ClassSize + 1

FROM Class c

INNER JOIN INSERTED i ON c.ClassID = i.ClassID;

END;

### **2.4.3: Khổng Thị Vân (Trigger trg\_UpdateScoreStatus, Trigger UpdateScoreStatus)**

* **Trigger trg\_UpdateScoreStatus**: cập nhật trạng thái và điểm chữ của sinh viên trong bảng Score dựa trên tổng điểm.

|  |
| --- |
| **CREATE TRIGGER trg\_UpdateScoreStatus**  **ON Score**  **AFTER UPDATE**  **AS**  **BEGIN**  **UPDATE Score**  **SET Score.TotalScore = Score.ContinuousAssessmentScore \* 0.5 + Score.FinalExamScore \* 0.5**  **FROM inserted i**  **WHERE Score.StudentID = i.StudentID AND Score.SubjectID = i.SubjectID;**  **UPDATE Score**  **SET Score.LetterGrade = CASE**  **WHEN Score.TotalScore >= 8.5 THEN 'A'**  **WHEN Score.TotalScore >= 7.0 THEN 'B'**  **WHEN Score.TotalScore >= 5.5 THEN 'C'**  **WHEN Score.TotalScore >= 4.0 THEN 'D'**  **ELSE 'F'**  **END**  **FROM inserted i**  **WHERE Score.StudentID = i.StudentID AND Score.SubjectID = i.SubjectID;**  **UPDATE Score**  **SET Score.[Status] = CASE**  **WHEN Score.TotalScore >= 4.0 THEN 'Pass'**  **ELSE 'Fail'**  **END**  **FROM inserted i**  **WHERE Score.StudentID = i.StudentID AND Score.SubjectID = i.SubjectID;**  **END;**  **GO**  **UPDATE Score**  **SET FinalExamScore = 8.0**  **WHERE StudentID = '2251161988' AND SubjectID = 'CSE204';** |

Kết quả:



* **Trigger UpdateScoreStatus:** Trigger tự động cập nhật trạng thái điểm

CREATE TRIGGER UpdateScoreStatus

ON Score

AFTER UPDATE

AS

BEGIN

UPDATE Score

SET [Status] = CASE

WHEN inserted.TotalScore >= 5 THEN 'Passed'

ELSE 'Failed'

END

FROM inserted

WHERE Score.StudentID = inserted.StudentID AND Score.SubjectID = inserted.SubjectID;

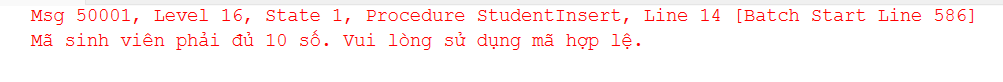
END

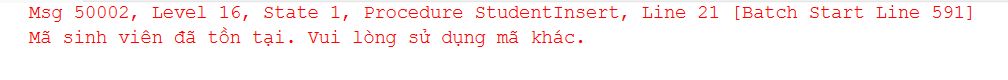
### **2.4.4: Vũ Hà Lâm (Trigger StudentInsertHistory, Trigger UpdateClassSizeOnStudentDelete)**

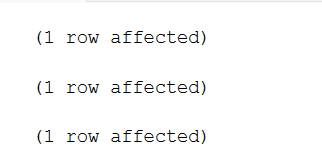
* **Trigger StudentInsertHistory:** Thêm sinh viên mới

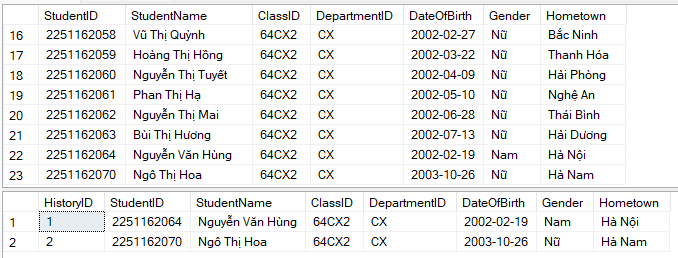
|  |
| --- |
| **CREATE TABLE StudentInsertHistory (**  **HistoryID INT IDENTITY(1,1) PRIMARY KEY, -- Mã định danh tự động**  **StudentID VARCHAR(10),**  **StudentName NVARCHAR(100),**  **ClassID VARCHAR(8),**  **DepartmentID VARCHAR(8),**  **DateOfBirth DATE,**  **Gender NVARCHAR(10),**  **Hometown NVARCHAR(100)**  **);**  **CREATE TRIGGER StudentInsert**  **ON Student**  **INSTEAD OF INSERT**  **AS**  **BEGIN**  **-- Kiểm tra xem có mã sinh viên nào không đủ 10 ký tự số**  **IF EXISTS (**  **SELECT 1**  **FROM inserted**  **WHERE LEN(StudentID) <> 10 OR StudentID NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'**  **)**  **BEGIN**  **-- Ném lỗi nếu mã sinh viên không hợp lệ**  **THROW 50001, N'Mã sinh viên phải đủ 10 số. Vui lòng sử dụng mã hợp lệ.', 1;**  **END**    **-- Kiểm tra xem có mã sinh viên nào đã tồn tại không**  **IF EXISTS (SELECT 1 FROM Student WHERE StudentID IN (SELECT StudentID FROM inserted))**  **BEGIN**  **-- Ném lỗi nếu mã sinh viên đã tồn tại**  **THROW 50002, N'Mã sinh viên đã tồn tại. Vui lòng sử dụng mã khác.', 1;**  **END**  **-- Nếu không có mã sinh viên nào trùng và mã hợp lệ, cho phép thêm bản ghi mới**  **INSERT INTO Student (StudentID, StudentName, ClassID, DepartmentID, DateOfBirth, Gender, Hometown)**  **SELECT StudentID,**  **StudentName,**  **ClassID,**  **DepartmentID,**  **DateOfBirth,**  **Gender,**  **Hometown**  **FROM inserted;**  **INSERT INTO StudentInsertHistory (StudentID, StudentName, ClassID, DepartmentID, DateOfBirth, Gender, Hometown)**  **SELECT StudentID,**  **StudentName,**  **ClassID,**  **DepartmentID,**  **DateOfBirth,**  **Gender,**  **Hometown**  **FROM inserted;**  **END;**  **--Nếu mã sinh viên không đủ 10 số**  **INSERT INTO Student (StudentID, StudentName, ClassID, DepartmentID, DateOfBirth, Gender, Hometown)**  **VALUES**  **('225116', N'Nguyễn Văn A', '64CX2', 'CX', '2002-05-19', N'Nam', N'Hà Nội');**  **-- Nếu mã sinh viên đã tồn tại**  **INSERT INTO Student (StudentID, StudentName, ClassID, DepartmentID, DateOfBirth, Gender, Hometown)**  **VALUES**  **('2251162064', N'Nguyễn Văn Hùng', '64CX2', 'CX', '2002-02-19', N'Nam', N'Hà Nội');**  **-- Nễu mã sinh viên chưa tồn tại**  **INSERT INTO Student (StudentID, StudentName, ClassID, DepartmentID, DateOfBirth, Gender, Hometown)**  **VALUES**  **('2251162070', N'Ngô Thị Hoa', '64CX2', 'CX', '2003-10-26', N'Nữ ', N'Hà Nam')**  **select \* from Student where ClassID= '64CX2'**  **select \* from StudentInsertHistory** |

Kết quả:









* **Trigger UpdateClassSizeOnStudentDelete:** Trigger cập nhật số lượng sinh viên trong lớp khi sinh viên bị xóa

CREATE TRIGGER UpdateClassSizeOnStudentDelete

ON Student

AFTER DELETE

AS

BEGIN

UPDATE Class

SET ClassSize = ClassSize - 1

WHERE ClassID = (SELECT ClassID FROM DELETED);

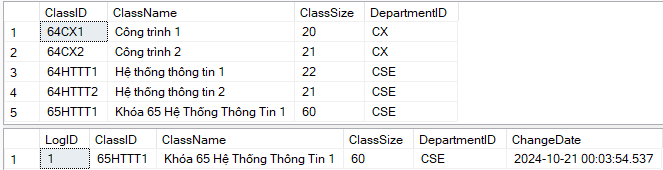
END;

### **2.4.5: Hoàng Khánh Hà (Trigger InsertClass, Trigger PreventDepartmentDeletion)**

* **Trigger InsertClass:** Thêm lớp mới

|  |
| --- |
| **CREATE TABLE InsertClass (**  **LogID INT IDENTITY(1,1) PRIMARY KEY,**  **ClassID VARCHAR(8),**  **ClassName NVARCHAR(100),**  **ClassSize INT,**  **DepartmentID VARCHAR(8),**  **ChangeDate DATETIME**  **);**  **CREATE TRIGGER trgBeforeInsertClass**  **ON Class**  **INSTEAD OF INSERT**  **AS**  **BEGIN**  **-- Kiểm tra nếu lớp học đã tồn tại**  **IF EXISTS (SELECT 1 FROM Class c JOIN inserted i ON c.ClassID = i.ClassID)**  **BEGIN**  **RAISERROR ('Lớp học này đã tồn tại, không thể thêm mới.', 16, 1);**  **ROLLBACK TRAN;**  **END**  **ELSE**  **BEGIN**  **-- Nếu lớp học chưa tồn tại, thực hiện chèn dữ liệu mới**  **INSERT INTO Class (ClassID, ClassName, ClassSize, DepartmentID)**  **SELECT ClassID, ClassName, ClassSize, DepartmentID**  **FROM inserted;**    **-- Ghi lại thông tin vào bảng ClassLog**  **INSERT INTO InsertClass(**  **ClassID,**  **ClassName,**  **ClassSize,**  **DepartmentID,**  **ChangeDate**  **)**  **SELECT**  **i.ClassID,**  **i.ClassName,**  **i.ClassSize,**  **i.DepartmentID,**  **GETDATE()**  **FROM**  **inserted i;**  **END**  **END;**  **GO**  **INSERT INTO Class(ClassID, ClassName, ClassSize, DepartmentID)**  **VALUES ('65HTTT1', N'Khóa 65 Hệ Thống Thông Tin 1', 60,'CSE')**  **Select \* from Class**  **Select \* from InsertClass** |

Kết quả:



* **Trigger PreventDepartmentDeletion:** trigger ngăn chặn việc xóa Department nếu Department đó còn có Class:

CREATE TRIGGER PreventDepartmentDeletion

ON Department

INSTEAD OF DELETE

AS

BEGIN

IF EXISTS (SELECT \* FROM Class WHERE DepartmentID IN (SELECT DepartmentID FROM deleted))

BEGIN

PRINT 'Cannot delete department with existing classes.'

ROLLBACK TRANSACTION;

END

ELSE

BEGIN

DELETE FROM Department WHERE DepartmentID IN (SELECT DepartmentID FROM deleted);

END;

END;

## **2.5. Xây dựng cursor**

### **2.5.1. Nguyễn Thị Thu Hằng**

* CURSOR 1: Hiển thị danh sách sinh viên trong mỗi lớp

DECLARE cur CURSOR FOR

SELECT ClassID FROM Class;

OPEN cur;

DECLARE @ClassID VARCHAR(8);

FETCH NEXT FROM cur INTO @ClassID;

WHILE @@FETCH\_STATUS = 0

BEGIN

SELECT StudentID, StudentName FROM Student WHERE ClassID = @ClassID;

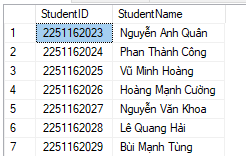
FETCH NEXT FROM cur INTO @ClassID;

END;

CLOSE cur;

DEALLOCATE cur;

Kết quả:



* CURSOR 2:hiển thị danh sách sinh viên và số môn học mà họ đã học qua (điểm >= 4).

DECLARE @StudentID VARCHAR(10);

DECLARE @StudentName NVARCHAR(100);

DECLARE @PassedSubjects INT;

DECLARE StudentCursor CURSOR FOR

SELECT StudentID, StudentName

FROM Student;

OPEN StudentCursor;

FETCH NEXT FROM StudentCursor INTO @StudentID, @StudentName;

WHILE @@FETCH\_STATUS = 0

BEGIN

SELECT @PassedSubjects = COUNT(\*)

FROM Score

WHERE StudentID = @StudentID AND TotalScore >= 4;

PRINT 'Mã SV: ' + @StudentID + ' | Tên SV: ' + @StudentName + ' | Số môn qua môn: ' + CAST(@PassedSubjects AS NVARCHAR);

FETCH NEXT FROM StudentCursor INTO @StudentID, @StudentName;

END;

CLOSE StudentCursor;

DEALLOCATE StudentCursor;

### **2.5.2. Hồ Thủy Tiên**

* Cursor 1: Hiển thị danh sách sinh viên trong mỗi khoa

DECLARE cur CURSOR FOR

SELECT DepartmentID FROM Department;

OPEN cur;

DECLARE @DepartmentID VARCHAR(8);

FETCH NEXT FROM cur INTO @DepartmentID;

WHILE @@FETCH\_STATUS = 0

BEGIN

SELECT S.StudentID, S.StudentName

FROM Student S

WHERE S.DepartmentID = @DepartmentID;

FETCH NEXT FROM cur INTO @DepartmentID;

END;

CLOSE cur;

DEALLOCATE cur;

* Cursor 2: Hiển thị danh sách sinh viên và điểm của họ trong một môn học

DECLARE cur CURSOR FOR

SELECT StudentID, SubjectID FROM Score;

OPEN cur;

DECLARE @StudentID VARCHAR(10), @SubjectID VARCHAR(8);

FETCH NEXT FROM cur INTO @StudentID, @SubjectID;

WHILE @@FETCH\_STATUS = 0

BEGIN

SELECT S.StudentName, SB.SubjectName, SC.TotalScore

FROM Score SC

JOIN Student S ON SC.StudentID = S.StudentID

JOIN [Subject] SB ON SC.SubjectID = SB.SubjectID

WHERE SC.StudentID = @StudentID AND SC.SubjectID = @SubjectID;

FETCH NEXT FROM cur INTO @StudentID, @SubjectID;

END;

CLOSE cur;

DEALLOCATE cur;

### **2.5.3. Khổng Thị Vân**

* CURSOR 1:Cập nhật học phí cho sinh viên theo kỳ

DECLARE cur CURSOR FOR

SELECT StudentID, Semester, TuitionAmount FROM TuitionFee;

OPEN cur;

DECLARE @StudentID VARCHAR(10), @Semester INT, @TuitionAmount FLOAT;

FETCH NEXT FROM cur INTO @StudentID, @Semester, @TuitionAmount;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF @Semester = 1

BEGIN

UPDATE TuitionFee

SET TuitionAmount = @TuitionAmount \* 1.1

WHERE StudentID = @StudentID AND Semester = @Semester;

END

ELSE

BEGIN

UPDATE TuitionFee

SET TuitionAmount = @TuitionAmount \* 1.05

WHERE StudentID = @StudentID AND Semester = @Semester;

END

FETCH NEXT FROM cur INTO @StudentID, @Semester, @TuitionAmount;

END;

CLOSE cur;

DEALLOCATE cur;

* CURSOR 2:hiển thị danh sách sinh viên và số lượng môn học mà họ đã đăng ký

DECLARE @StudentID NVARCHAR(20);

DECLARE @StudentName NVARCHAR(50);

DECLARE @SubjectCount INT;

DECLARE StudentSubjectCursor CURSOR FOR

SELECT

S.StudentID,

S.StudentName,

COUNT(DISTINCT SC.SubjectID) AS SubjectCount

FROM

Student S

LEFT JOIN

Score SC ON S.StudentID = SC.StudentID

GROUP BY

S.StudentID, S.StudentName;

OPEN StudentSubjectCursor;

FETCH NEXT FROM StudentSubjectCursor INTO @StudentID, @StudentName, @SubjectCount;

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT 'Mã sinh viên: ' + @StudentID + ', Tên: ' + @StudentName + ', Số lượng môn học: ' + CAST(@SubjectCount AS NVARCHAR);

FETCH NEXT FROM StudentSubjectCursor INTO @StudentID, @StudentName, @SubjectCount;

END

CLOSE StudentSubjectCursor;

DEALLOCATE StudentSubjectCursor;

### **2.5.4. Vũ Hà Lâm**

* CURSOR 1:Lấy thông tin về các sinh viên đã nhận thưởng

DECLARE cur CURSOR FOR

SELECT StudentID FROM Award;

OPEN cur;

DECLARE @StudentID VARCHAR(10);

FETCH NEXT FROM cur INTO @StudentID;

WHILE @@FETCH\_STATUS = 0

BEGIN

SELECT S.StudentName, A.AwardAmount, A.Reason

FROM Award A

JOIN Student S ON A.StudentID = S.StudentID

WHERE A.StudentID = @StudentID;

FETCH NEXT FROM cur INTO @StudentID;

END;

CLOSE cur;

DEALLOCATE cur;

### **2.5.5. Hoàng Khánh Hà**

* CURSOR 1:Cập nhật trạng thái học bổng cho sinh viên

DECLARE cur CURSOR FOR

SELECT StudentID, AwardAmount FROM Award;

OPEN cur;

DECLARE @StudentID VARCHAR(10), @AwardAmount NVARCHAR(50);

FETCH NEXT FROM cur INTO @StudentID, @AwardAmount;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF @AwardAmount = 'High'

BEGIN

UPDATE Student

SET Hometown = 'Top Scholar'

WHERE StudentID = @StudentID;

END

FETCH NEXT FROM cur INTO @StudentID, @AwardAmount;

END;

CLOSE cur;

DEALLOCATE cur;

# **CHƯƠNG 3: ỨNG DỤNG CÁC ĐỐI TƯỢNG VÀO PHẦN MỀM QUẢN LÝ**

## **3.1 Mô tả chung chức năng phần mềm quản lý**

Phần mềm quản lý sinh viên là một hệ thống toàn diện hỗ trợ các trường học và tổ chức giáo dục trong việc quản lý thông tin sinh viên một cách hiệu quả, minh bạch và thuận tiện. Chức năng chính của phần mềm bao gồm:

* **Quản lý thông tin sinh viên**: Lưu trữ và quản lý chi tiết thông tin cá nhân của từng sinh viên như mã số, họ tên, ngày sinh, giới tính, địa chỉ, khoa và lớp học. Hệ thống đảm bảo bảo mật và dễ dàng tìm kiếm, cập nhật khi cần.
* **Quản lý môn học**: Chức năng quản lý môn học cung cấp cho nhà trường công cụ để dễ dàng tạo mới, chỉnh sửa và cập nhật danh sách môn học cho từng kỳ học. Người quản trị có thể thêm thông tin chi tiết về môn học, bao gồm mã môn, tên môn học, số tín chỉ, học kỳ, và mô tả ngắn về nội dung. Chức năng này còn cho phép cập nhật các thay đổi trong cấu trúc khóa học, đáp ứng nhu cầu đào tạo thực tế. Ngoài ra, hệ thống có thể hiển thị danh sách môn học hiện có, tìm kiếm theo tiêu chí cụ thể như mã môn hoặc tên môn.
* **Quản lý điểm số và kết quả học tập**: Phần mềm lưu trữ và cập nhật điểm số, kết quả học tập của sinh viên theo từng môn, từng học kỳ. Hệ thống hỗ trợ tính toán điểm trung bình, phân loại học lực.
* **Quản lý học phí**: Tự động tính toán và theo dõi học phí dựa trên số tín chỉ và các khoản phí khác.
* **Khen thưởng và kỷ luật**: Hệ thống lưu trữ và quản lý các quyết định khen thưởng, kỷ luật của sinh viên dựa trên thành tích học tập và hoạt động ngoại khóa, tạo động lực và khuyến khích tinh thần học tập.
* **Giao diện thân thiện**: Giao diện dễ sử dụng cho cả sinh viên và nhân viên nhà trường.

Phần mềm quản lý sinh viên giúp nâng cao hiệu quả quản lý, giảm bớt gánh nặng công việc hành chính, đồng thời tạo điều kiện cho sinh viên và giảng viên tương tác dễ dàng, tiện lợi trong môi trường học tập hiện đại.

## **3.2 Mô tả ứng dụng một số đối tượng dữ liệu (view, procedure, function, trigger) trong phần mềm**

Các đối tượng dữ liệu như View, Stored Procedure, Function, và Trigger trong phần mềm quản lý sinh viên giúp thực hiện các tác vụ quản lý dữ liệu phức tạp một cách dễ dàng, dưới đây là mô tả chi tiết các chức năng của từng đối tượng:

**View - StudentAttemptSummaryView**:

* View này cung cấp một cái nhìn tổng quan về thông tin của sinh viên trong một môn học cụ thể (CSE484). Bằng cách hiển thị mã sinh viên, tên sinh viên, mã môn học, lớp học và số lần thi, nó hỗ trợ việc kiểm soát, giám sát hiệu quả kết quả học tập của sinh viên. Chức năng này đặc biệt hữu ích cho phòng đào tạo khi cần nắm bắt tình hình thi cử và mức độ khó khăn của từng môn học.

**Stored Procedure - GetAwardByDate**:

* Procedure GetAwardByDate giúp thống kê chi tiết các khen thưởng của sinh viên trong một khoảng thời gian xác định. Procedure này không chỉ liệt kê từng khen thưởng mà còn cung cấp tổng số khen thưởng trong giai đoạn đó. Điều này giúp bộ phận quản lý sinh viên dễ dàng theo dõi và báo cáo khen thưởng, đồng thời xác định giai đoạn nào có nhiều hoạt động khen thưởng, giúp nâng cao tính minh bạch trong quy trình khen thưởng.

**Function**:

* **fnCalculateGPA**: Tính toán điểm trung bình tích lũy (GPA) của sinh viên dựa trên điểm tổng kết và số tín chỉ của các môn học. Hàm này hỗ trợ việc đánh giá nhanh học lực của sinh viên theo tiêu chuẩn quốc tế.
* **TTSV**: Hàm này trả về chi tiết về môn học, tín chỉ và học phí của từng môn học mà sinh viên đăng ký trong một học kỳ cụ thể. Từ đó, hệ thống sẽ dễ dàng tổng hợp và hiển thị học phí theo học kỳ, hỗ trợ sinh viên và bộ phận tài chính trong việc kiểm tra và quản lý học phí.
* **ClassifyStudentByTotalScore**: Phân loại sinh viên theo điểm trung bình tích lũy thành các mức như "Giỏi", "Khá", "Trung bình" hoặc "Yếu". Hàm này hữu ích trong việc báo cáo, xét học bổng và khen thưởng, giúp nhà trường đưa ra quyết định phù hợp dựa trên năng lực của sinh viên.

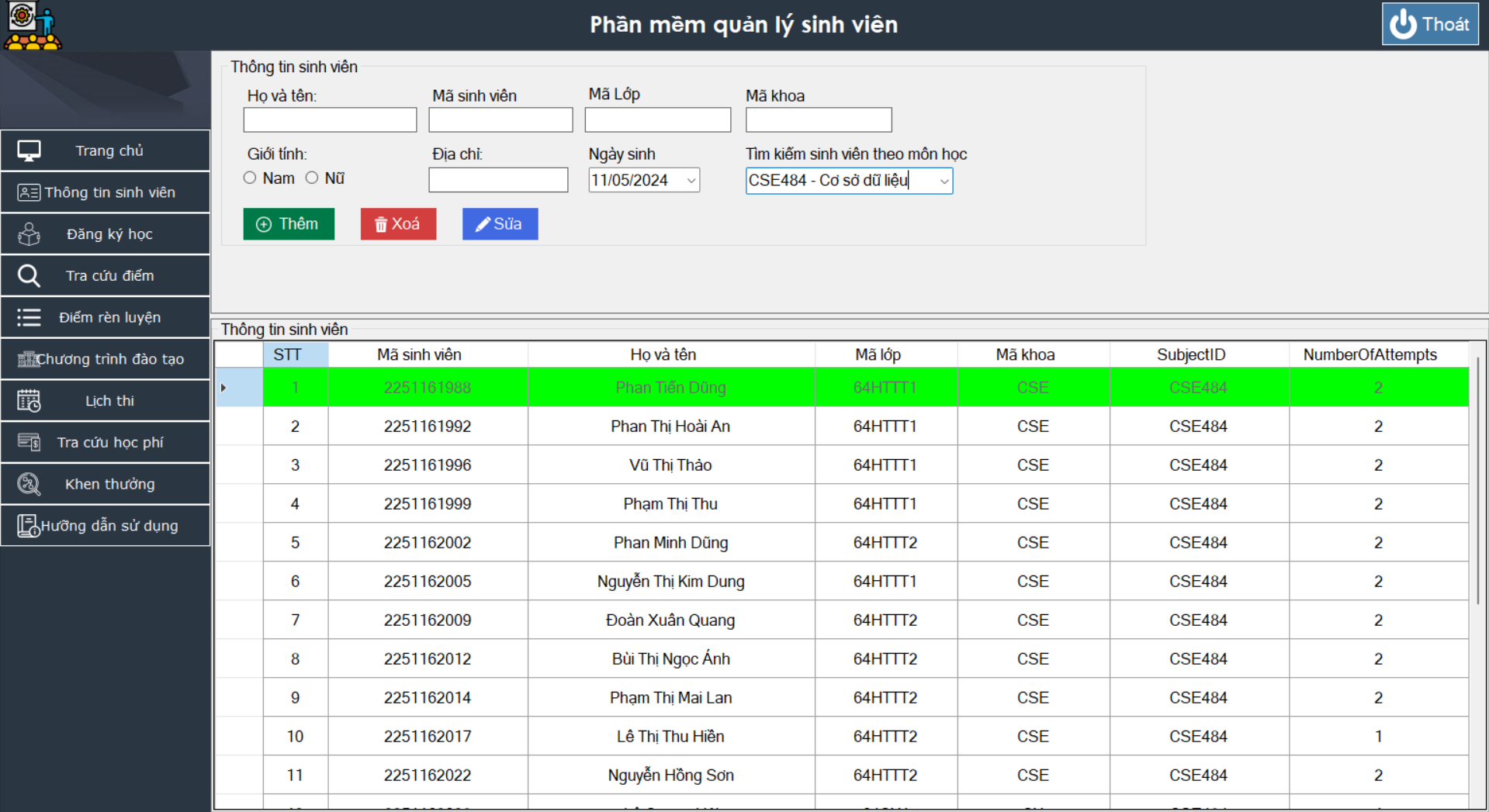
**Trigger - trg\_UpdateScoreStatus**:

* Trigger trg\_UpdateScoreStatus tự động cập nhật điểm tổng kết và trạng thái (Pass/Fail) của sinh viên khi dữ liệu về điểm thành phần được thay đổi. Trigger này đảm bảo rằng dữ liệu về điểm số của sinh viên luôn chính xác và đồng nhất. Nó cũng hỗ trợ trong việc đánh giá học lực của sinh viên một cách liên tục, giảm thiểu sai sót trong việc lưu trữ điểm số.

Các đối tượng này giúp tự động hóa và tối ưu hóa các quy trình quản lý dữ liệu, từ đó hỗ trợ hiệu quả cho công tác quản lý và ra quyết định trong phần mềm quản lý sinh viên.

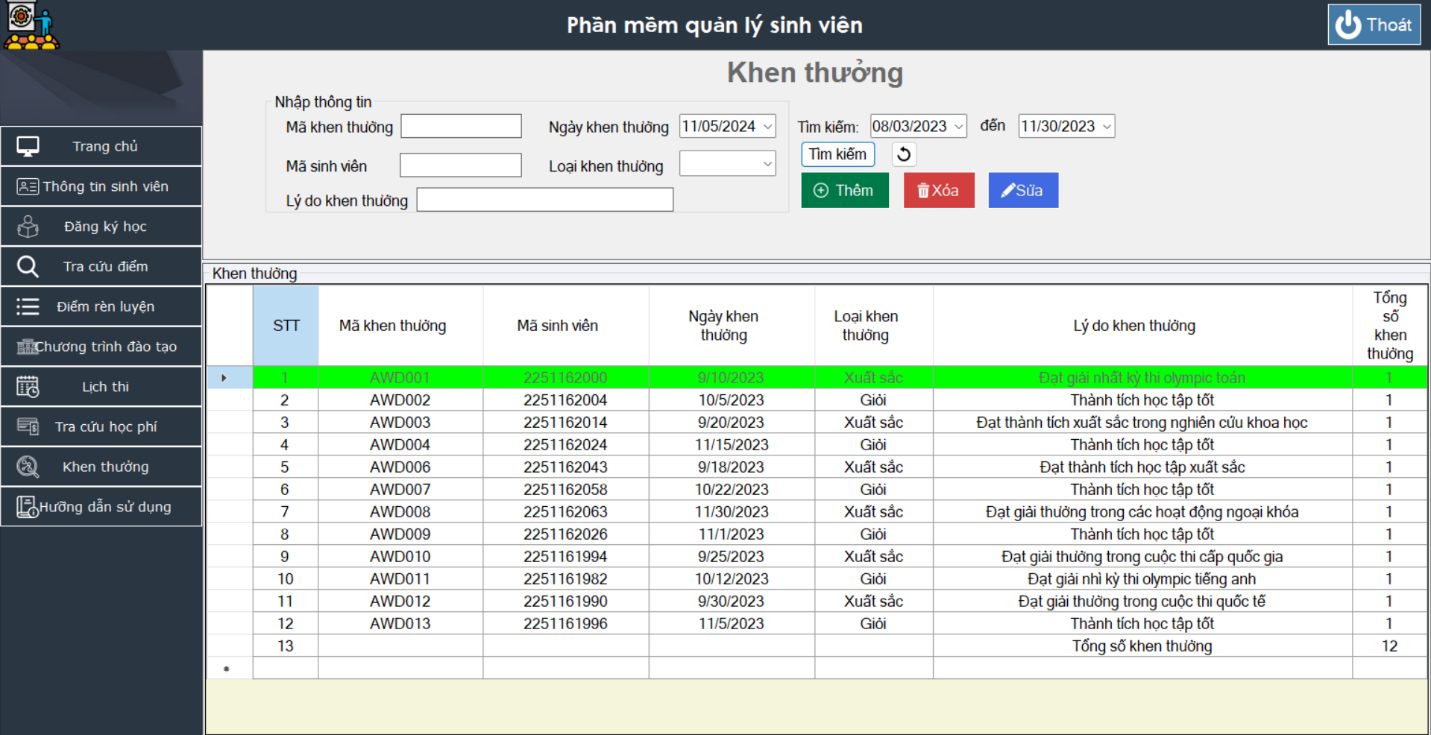
## **3.3 Demo chức năng giao diện ứng dụng**

**View - StudentAttemptSummaryView**:



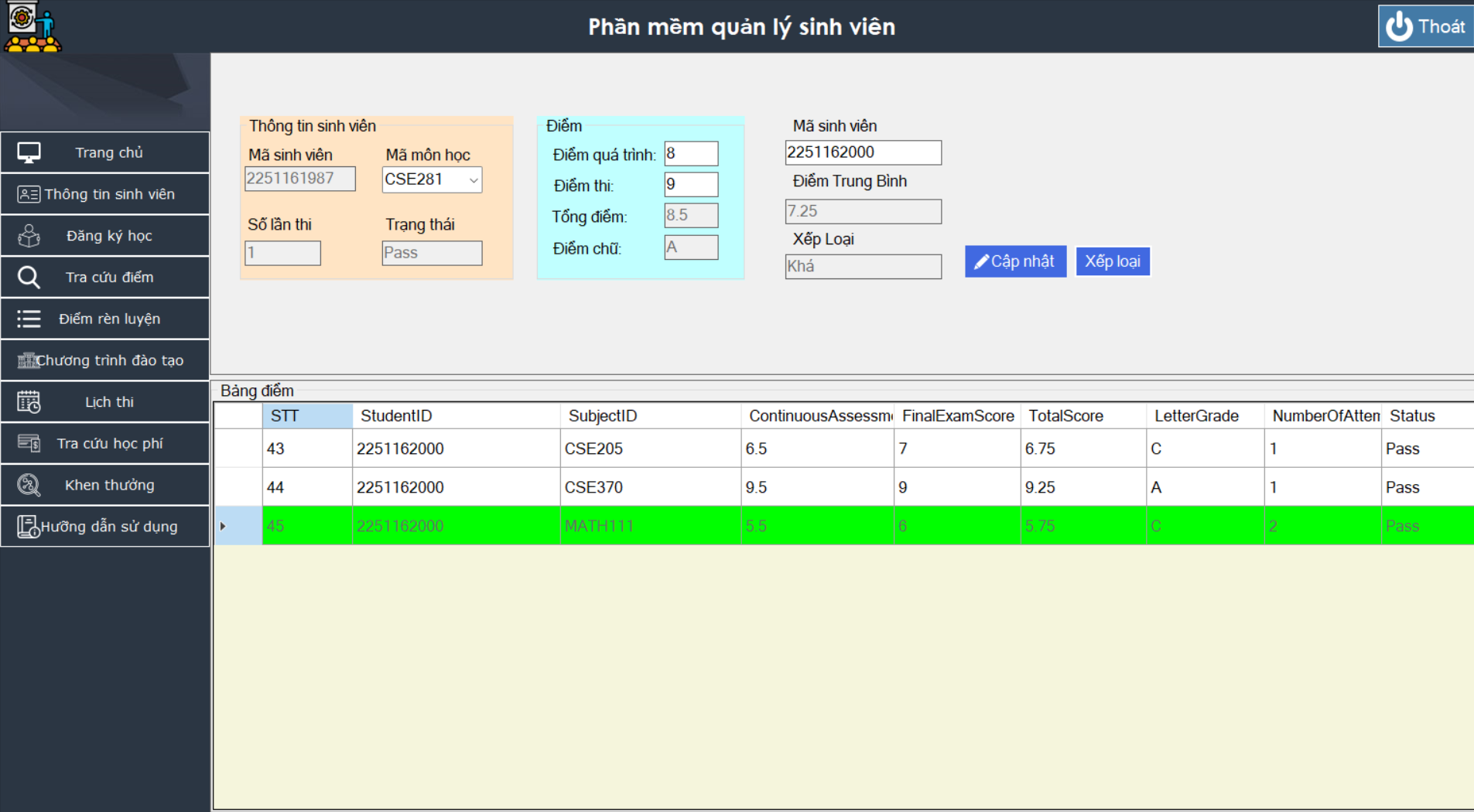
**Stored Procedure**:

* **GetAwardByDate**

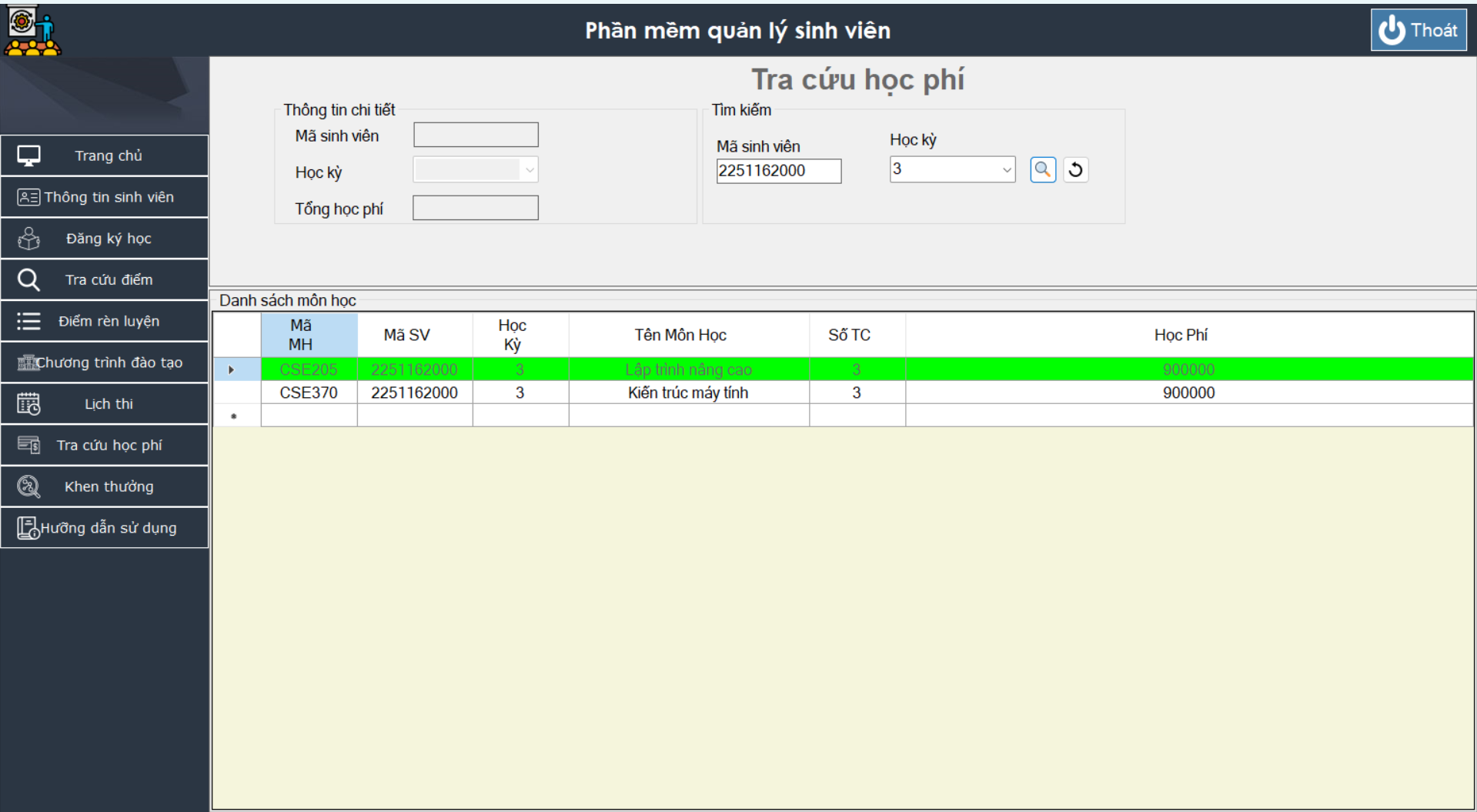


**Function**:

* **fnCalculateGPA** và **ClassifyStudentByTotalScore**



* **TTSV**:



**Trigger - trg\_UpdateScoreStatus**:

