Tatiana Ensslin

CSE 581

**Lab 12: Stored Procedures**

**Steps:**

1. Create a stored procedure. Input for the stored procedure will be FacultyId, Student Id, Course Id and a Numerical Grade.

The stored procedure will attempt to assign a grade to the student, for the class that they are enrolled in, assuming it is legal to do so. The stored procedure will do the following:

1. If the user attempting to assign the grade (FacultyId) is not the faculty teaching the course, **print a message** that says “Error: You are not allowed to assign grades for this course.”, **and quit**.
2. Assuming A passed, and if the student is not enrolled in the class, **print a message** that says “Error: The student is not taking the course you specified.”, **and quit**.
3. Assuming B passed, and if the student already has a grade for that class, **change it** **to the new grade**, and **print a message** that says “Success, with a warning - Student’s existing grade OLD\_GRADE was changed to NEW\_GRADE.” Please replace the OLD\_GRADE with the existing grade, and the NEW\_GRADE with the new grade. **Then quit**.
4. Assuming B passed and C did not execute (the student does not have a grade for the course yet), **insert the new grade** and **print a message** “Success.”, **then quit**.

CREATE PROCEDURE UpdateGrade (@FacultyId AS VARCHAR(20),

@StudentId AS VARCHAR(20),

@CourseId AS INTEGER,

@NumericalGrade AS INTEGER)

as

DECLARE @OldGrade INTEGER

IF (@FacultyId != (SELECT Faculty --A

FROM Courses

WHERE CourseID = @CourseId))

BEGIN

PRINT 'Error: You are not allowed to assign grades for this course.';

END

ELSE IF (SELECT EnrollmentId --B

FROM CourseEnrollment

WHERE CourseId = @CourseId and StudentId = @StudentId) is NULL

BEGIN

PRINT 'Error: The student is not taking the course you specified.';

END

ELSE IF (SELECT FinalGrade --C

FROM CourseEnrollment

WHERE CourseId = @CourseId and StudentId = @StudentId) is NOT NULL

BEGIN

SET @OldGrade = (SELECT FinalGrade

FROM CourseEnrollment

WHERE CourseId = @CourseId and StudentId = @StudentId)

UPDATE CourseEnrollment

SET FinalGrade = @NumericalGrade

WHERE CourseId = @CourseId and StudentId = @StudentId

PRINT ' Success, with a warning- The Students exisiting grade ' + CAST(@OldGrade AS VARCHAR) + 'was changed to' + CAST(@NumericalGrade AS VARCHAR);

END

ELSE --D

BEGIN

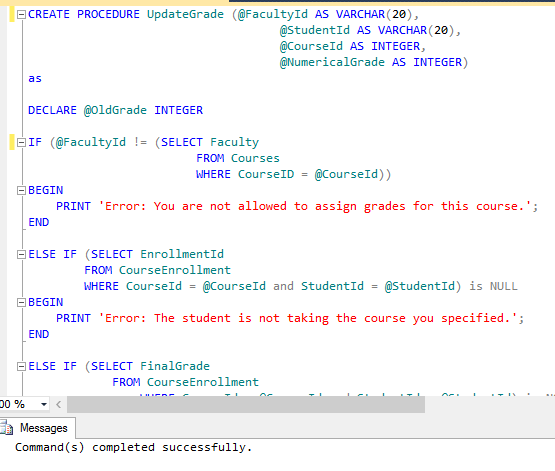
UPDATE CourseEnrollment

SET FinalGrade = @NumericalGrade

WHERE CourseId = @CourseId and StudentId = @StudentId

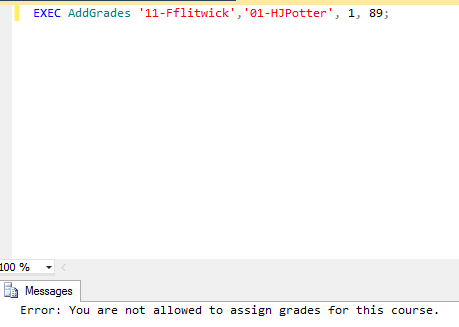
PRINT 'Success!';

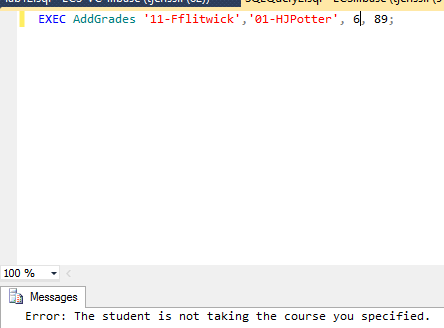
END



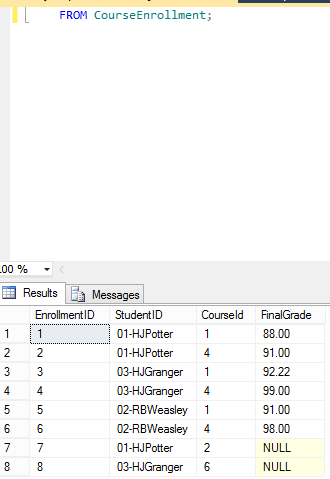
1. Run the stored procedure to prove[[1]](#footnote-1) that all 4 cases work[[2]](#footnote-2).

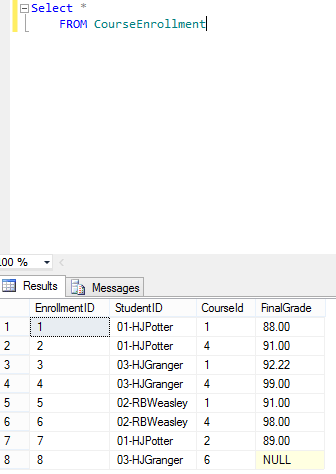
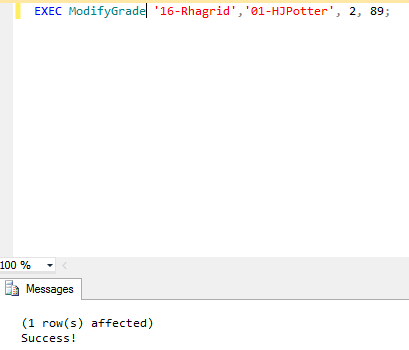
**PLEASE NOTE THAT THROUGH MY TESTING I CONSISTENTLY CREATED UPDATED AND RENAMED VERSIONS OF MY STORED PROCEDURE. THIS IS WHY DURING EXECUTION YOU WILL SEE THE NAMES OF THE SP CHANGE. PLEASE NOTE THIS IS STILL MY SCREENSHOTS AND CODE. The final version is ModifyGrade.**

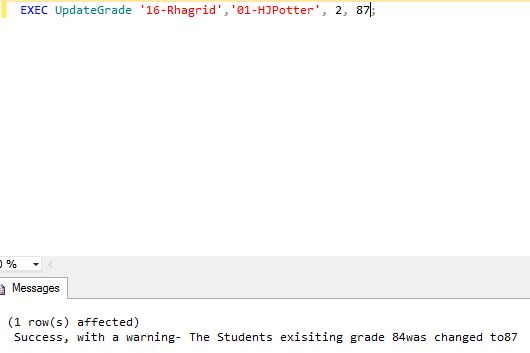
CASE A:

CASE B:

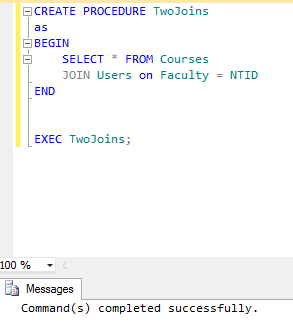
CASE C:

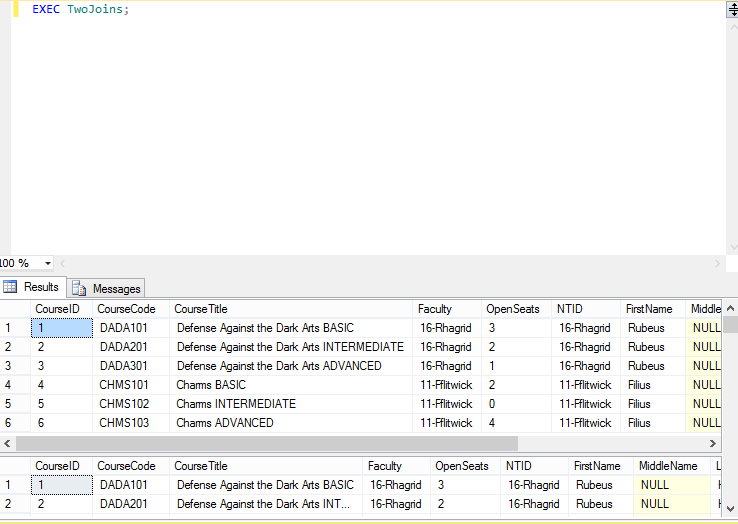


Cs

Case D: 

1. Create a SP that will return back 2 joined tables. I’ll leave which tables and which join type up to you.





1. data was or was not changed, depending on the case. The only way you can prove this to me is via screenshots. [↑](#footnote-ref-1)
2. If you need to modify the data in order to be able to execute all of the cases, feel free to do so. I do not need to see these, just make sure that the data prior to the SP’s execution is showing the valid input. [↑](#footnote-ref-2)