CSE 581

**Lab 18: Transactions**

*Do this against YOUR own database.*

***For the entire lab, provide all of your SQL as text, inserted at the beginning of the document.***

***For each step also provide a screenshot of execution, showing the SQL and the results.***

***Please make sure both the SQL and the screenshots are marked w/ the question number that they are answering.***

**Steps:**

1. Modify[[1]](#footnote-1) the stored procedure from lab 12, so that way it will behave as follows (*use transactions to accomplish all of this behavior, and print a message stating “Transaction was rolled back”*):

* do the original checks/behaviors as before;
* *(assuming the checks above passed, and the grade was inserted)*, after the modification, **if** all of the students in the course have a grade, **calculate the average** for all of these students. If the average is below 50, **print out a message** “”, then **roll back the change** and **then** **quit**.

1. Prove2 that if (any) one of the original checks fails, it does so via a transaction3.
2. Prove2 that the new check fails when the conditions are met, and does so via a transaction3
3. Prove[[2]](#footnote-2) that a successful data modification works correctly[[3]](#footnote-3).

CREATE PROCEDURE dbo.AssignGrade(@FacultyId AS VARCHAR(20), @CourseId AS INT, @StudentId AS VARCHAR(20), @Grade AS DECIMAL(5,2))

AS

BEGIN TRAN myTran

DECLARE @avgScore DECIMAL(5,2)

-- a) Verify the instructor is teaching the course

IF(NOT EXISTS(SELECT 1 FROM Courses WHERE CourseId = @CourseId AND Faculty = @FacultyId))

BEGIN

-- Instructor is NOT teaching the course

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

PRINT 'Transaction was rolled back'

ROLLBACK TRAN myTran

END

-- b) Verify the student is taking the course

ELSE IF(NOT EXISTS(SELECT 1 FROM CourseEnrollment WHERE CourseId = @CourseId AND StudentId = @StudentId))

BEGIN

-- Student is NOT enrolled in the course

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

PRINT 'Transaction was rolled back'

ROLLBACK TRAN myTran

END

ELSE

BEGIN

DECLARE @OldGrade AS DECIMAL(5,2)

SELECT @OldGrade = (SELECT TOP 1 FinalGrade

FROM CourseEnrollment

WHERE CourseId = @CourseId

AND StudentId = @StudentId)

-- Set the new grade, both c) and d)

UPDATE CourseEnrollment

SET FinalGrade = @Grade

WHERE CourseId = @CourseId AND StudentId = @StudentId

SET @avgScore = (SELECT AVG(C.FinalGrade) FROM CourseEnrollment C WHERE C.CourseId = @CourseId);

IF @avgScore < 50

BEGIN

PRINT 'Error: Your grade average is too low. Please check the grades to make sure you have inserted the correct values.'

PRINT 'Transaction was rolled back'

ROLLBACK TRAN myTran

END

ELSE

BEGIN

IF(@OldGrade IS NOT NULL)

BEGIN

-- c) Replacing an old grade.

PRINT 'Success, with a warning - Student’s existing grade ' + CAST(@OldGrade AS VARCHAR(10)) + ' was changed to ' + CAST(@Grade AS VARCHAR(10)) + '.'

COMMIT TRAN myTran

END

ELSE

BEGIN

-- d) No existing grade.

PRINT 'Success.'

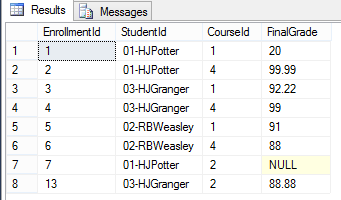
COMMIT TRAN myTran

END

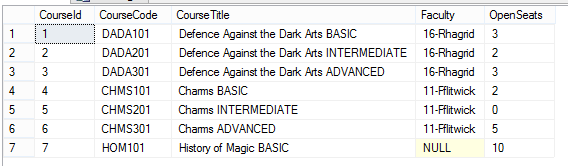
END

END;

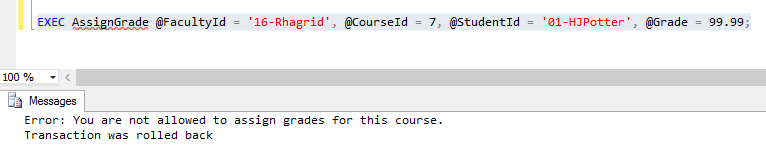
**The original table of CourseEnrollment is like:**



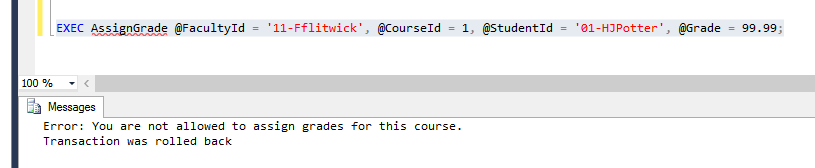
**The original table of Courses is like:**



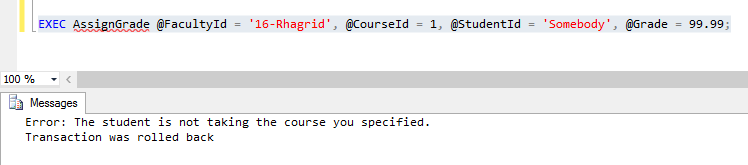
**Test case 1: when the faculty does not match (faculty is null for courseId 7):**



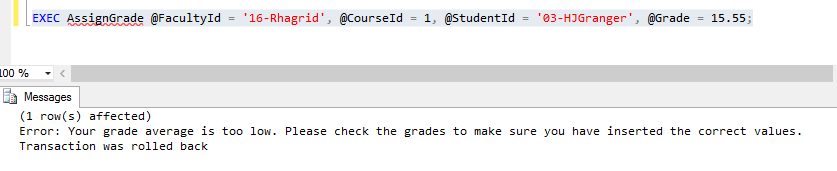
**Test case2: when the faculty does not match (wrong name of faculty):**



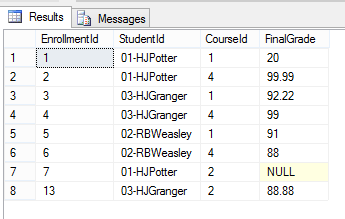
**Test case 3: when the studentId is wrong:**



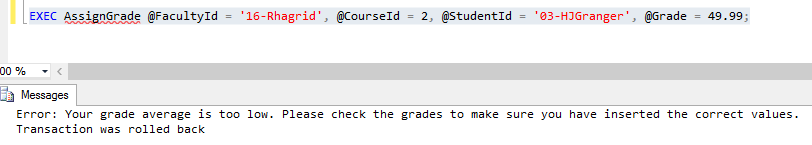
**Test case 4: when the average score of the courseId 1 is too low (<50):**



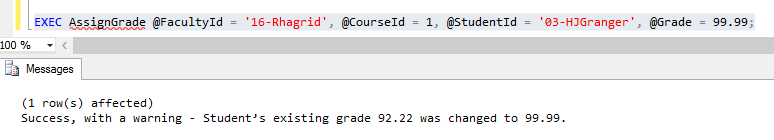
**We can see that after all the above, the data was not affected:**

****

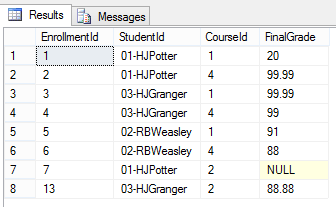
**Test cast 5: Check when there is a NULL score exist for one course, and the average is below 50**

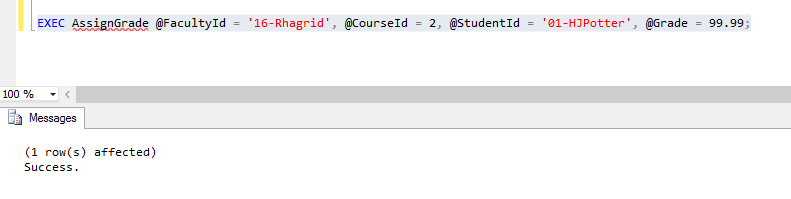


**Test case 6: Check when a score was update with the average above 50.**

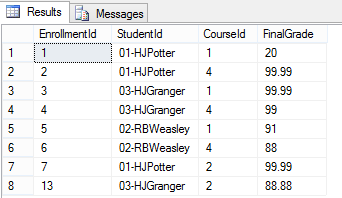


**The table was also updated correctly.**



**Test cast 7: Check when a NULL score was updated, and the average is above 50.**

**The table was also updated correctly.**



1. You can either ALTER, or DROP and CREATE [↑](#footnote-ref-1)
2. You have to prove to me that it works exactly as expected. That means you need to prove that the right messages were printed, and that the data was or was not changed, depending on the case. The only way you can prove this to me is via screenshots. [↑](#footnote-ref-2)
3. 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-3)