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] 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**.

2. Prove[2] that if (any) one of the original checks fails, it does so via a transaction[3].

3. Prove[2] that the new check fails when the conditions are met, and does so via a transaction[3]

4. Prove[2] that a successful data modification works correctly[3].

[1] You can either ALTER, or DROP and CREATE
[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.
[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.

```
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:

Results Messages

	EnrollmentId	StudentId	CourseId	FinalGrade
1	1	01-HJPotter	1	20
2	2	01-HJPotter	4	99.99
3	3	03-HJGranger	1	92.22
4	4	03-HJGranger	4	99
5	5	02-RBWeasley	1	91
6	6	02-RBWeasley	4	88
7	7	01-HJPotter	2	NULL
8	13	03-HJGranger	2	88.88

The original table of Courses is like:

	CourseId	CourseCode	CourseTitle	Faculty	OpenSeats
1	1	DADA101	Defence Against the Dark Arts BASIC	16-Rhagrid	3
2	2	DADA201	Defence Against the Dark Arts INTERMEDIATE	16-Rhagrid	2
3	3	DADA301	Defence Against the Dark Arts ADVANCED	16-Rhagrid	3
4	4	CHMS101	Charms BASIC	11-Fflitwick	2
5	5	CHMS201	Charms INTERMEDIATE	11-Fflitwick	0
6	6	CHMS301	Charms ADVANCED	11-Fflitwick	5
7	7	HOM101	History of Magic BASIC	NULL	10

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

```
EXEC AssignGrade @FacultyId = '16-Rhagrid', @CourseId = 7, @StudentId = '01-HJPotter', @Grade = 99.99;
```

100 %

Messages

```
Error: You are not allowed to assign grades for this course.
Transaction was rolled back
```

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

```
EXEC AssignGrade @FacultyId = '11-Fflitwick', @CourseId = 1, @StudentId = '01-HJPotter', @Grade = 99.99;
```

100 %

Messages

```
Error: You are not allowed to assign grades for this course.
Transaction was rolled back
```

Test case 3: when the studentId is wrong:

```
EXEC AssignGrade @FacultyId = '16-Rhagrid', @CourseId = 1, @StudentId = 'Somebody', @Grade = 99.99;
```

100 %

Messages

```
Error: The student is not taking the course you specified.
Transaction was rolled back
```

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

```
EXEC AssignGrade @FacultyId = '16-Rhagrid', @CourseId = 1, @StudentId = '03-HJGranger', @Grade = 15.55;
```

100 %

Messages

```
(1 row(s) affected)
Error: Your grade average is too low. Please check the grades to make sure you have inserted the correct values.
Transaction was rolled back
```

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

Results | Messages

	EnrollmentId	StudentId	CourseId	FinalGrade
1	1	01-HJPotter	1	20
2	2	01-HJPotter	4	99.99
3	3	03-HJGranger	1	92.22
4	4	03-HJGranger	4	99
5	5	02-RBWeasley	1	91
6	6	02-RBWeasley	4	88
7	7	01-HJPotter	2	NULL
8	13	03-HJGranger	2	88.88

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

```
EXEC AssignGrade @FacultyId = '16-Rhagrid', @CourseId = 2, @StudentId = '03-HJGranger', @Grade = 49.99;
```

00 %

Messages

```
Error: Your grade average is too low. Please check the grades to make sure you have inserted the correct values.
Transaction was rolled back
```

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

```
EXEC AssignGrade @FacultyId = '16-Rhagrid', @CourseId = 1, @StudentId = '03-HJGranger', @Grade = 99.99;
```

100 %

Messages

```
(1 row(s) affected)
Success, with a warning - Student's existing grade 92.22 was changed to 99.99.
```

The table was also updated correctly.

Results | Messages

	EnrollmentId	StudentId	CourseId	FinalGrade
1	1	01-HJPotter	1	20
2	2	01-HJPotter	4	99.99
3	3	03-HJGranger	1	99.99
4	4	03-HJGranger	4	99
5	5	02-RBWeasley	1	91
6	6	02-RBWeasley	4	88
7	7	01-HJPotter	2	NULL
8	13	03-HJGranger	2	88.88

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

```
EXEC AssignGrade @FacultyId = '16-Rhagrid', @CourseId = 2, @StudentId = '01-HJPotter', @Grade = 99.99;
```

100 %

Messages

```
(1 row(s) affected)
Success.
```

The table was also updated correctly.

Results | Messages

	EnrollmentId	StudentId	CourseId	FinalGrade
1	1	01-HJPotter	1	20
2	2	01-HJPotter	4	99.99
3	3	03-HJGranger	1	99.99
4	4	03-HJGranger	4	99
5	5	02-RBWeasley	1	91
6	6	02-RBWeasley	4	88
7	7	01-HJPotter	2	99.99
8	13	03-HJGranger	2	88.88