**BÁO CÁO SƠ BỘ BÀI TẬP LỚN SỐ 2 MÔN HỆ CƠ SỞ DỮ LIỆU**

**HK231 - Lớp: L0x**

**Nhóm**: SV58>

**Danh sách thành viên**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Stt** | **Họ tên** | **MSSV** | **Tỷ lệ đóng góp** | **Ký tên xác nhận** |
| **1** | **Nguyễn Đinh Bằng** | **2210298** |  |  |
| **2** | **Trương Hữu Quốc Gia Bình** | **2310314** |  |  |
| **3** | **Nguyễn Quốc Thịnh** | **2213296** |  |  |
| **4** | **Dương Công Phát** | **2212498** |  |  |
| **5** | **Nguyễn Huỳnh Đức Huy** | **2211218** |  |  |

**Chủ đề**: <chủ đề mà nhóm chọn>

**1. Thiết kế ERD cập nhật mới nhất**

A diagram of a network

Description automatically generated

**2. Trigger – Thủ tục - Hàm**

**2.1 Thủ tục INSERT / UPDATE / DELETE dữ liệu vào 1 bảng dữ liệu**

**Bảng dữ liệu: <course>**

1. Thủ tục INSERT: **InsertCourse**

Thủ tục này được sử dụng để thêm một khóa học mới vào bảng Course trong cơ sở dữ liệu.

**Usecase:** Sử dụng trong các trường hợp cần thêm một khóa học mới vào hệ thống quản lý khóa học.

**Thao tác chính:**

1. Thêm dữ liệu vào bảng Course với các giá trị được truyền vào qua các biến @p\_Course\_ID, @p\_Course\_Name, @p\_Course\_Status, @p\_Course\_Description, @p\_Course\_Price, @p\_Course\_Image, @p\_Course\_Start\_Date, @p\_Course\_End\_Date, @p\_Course\_Categories\_ID.

2. Lấy giá trị Course\_ID vừa được tạo bằng cách sử dụng SCOPE\_IDENTITY().

3. Trả về thông báo xác nhận thành công với giá trị Course\_ID vừa được tạo.

**Input:**

\* @p\_Course\_ID: ID của khóa học.

\* @p\_Course\_Name: Tên của khóa học.

\* @p\_Course\_Status: Trạng thái của khóa học.

\* @p\_Course\_Description: Mô tả khóa học.

\* @p\_Course\_Price: Giá của khóa học.

\* @p\_Course\_Image: Đường dẫn hình ảnh của khóa học.

\* @p\_Course\_Start\_Date: Ngày bắt đầu của khóa học.

\* @p\_Course\_End\_Date: Ngày kết thúc của khóa học.

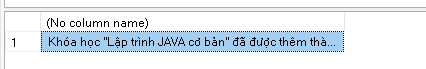
\* @p\_Course\_Categories\_ID: ID của danh mục khóa học.

**Output:**

Thông báo xác nhận thành công với giá trị Course\_ID vừa được tạo, ví dụ: "12345 Khóa học "Lập trình web" đã được thêm thành công với Course\_ID = 67890!".

**Test:**

EXEC InsertCourse 503, N'Lập trình JAVA cơ bản', 'Open', N'Khóa học giúp bạn nắm vững các kiến thức cơ bản về JAVA.', 400, 'java\_basic.jpg', '2024-01-01', '2024-06-30', 1;



1. Thủ tục UPDATE: UpdateCourse

Đây là một thủ tục lưu trữ (stored procedure) được sử dụng để cập nhật thông tin của một khóa học trong cơ sở dữ liệu. Nó thực hiện việc kiểm tra tính hợp lệ của dữ liệu đầu vào và cập nhật thông tin khóa học vào bảng Course nếu dữ liệu hợp lệ.

**Usecase**: Sử dụng trong các trường hợp cần cập nhật thông tin của một khóa học đã tồn tại trong hệ thống quản lý khóa học.

**Thao tác chính**:

1. Kiểm tra xem khóa học có tồn tại không.
2. Kiểm tra xem danh mục có tồn tại không.
3. Cập nhật thông tin khóa học nếu cả khóa học và danh mục đều tồn tại, bao gồm:
   * Cập nhật tên khóa học (Course\_Name).
   * Cập nhật trạng thái khóa học (Course\_Status).
   * Cập nhật mô tả khóa học (Course\_Description).
   * Cập nhật giá khóa học (Course\_Price).
   * Cập nhật hình ảnh khóa học (Course\_Image).
   * Cập nhật ngày bắt đầu của khóa học (Course\_Start\_Date).
   * Cập nhật ngày kết thúc của khóa học (Course\_End\_Date).
   * Cập nhật ID danh mục của khóa học (Course\_Categories\_ID).
4. Trả về thông báo xác nhận thành công.

**Input**:

* @p\_Course\_ID: ID của khóa học cần cập nhật.
* @p\_Course\_Name: Tên của khóa học.
* @p\_Course\_Status: Trạng thái của khóa học.
* @p\_Course\_Description: Mô tả khóa học.
* @p\_Course\_Price: Giá của khóa học.
* @p\_Course\_Image: Đường dẫn hình ảnh của khóa học.
* @p\_Course\_Start\_Date: Ngày bắt đầu của khóa học.
* @p\_Course\_End\_Date: Ngày kết thúc của khóa học.
* @p\_Course\_Categories\_ID: ID của danh mục khóa học.

**Output**:

* Thông báo xác nhận thành công, ví dụ: "Khóa học "Lập trình C++ cơ bản" đã được cập nhật thành công!".

**Lỗi**: Hàm sẽ trả về lỗi "Khóa học không tồn tại!" hoặc "Danh mục không tồn tại!" nếu khóa học hoặc danh mục không được tìm thấy trong cơ sở dữ liệu.

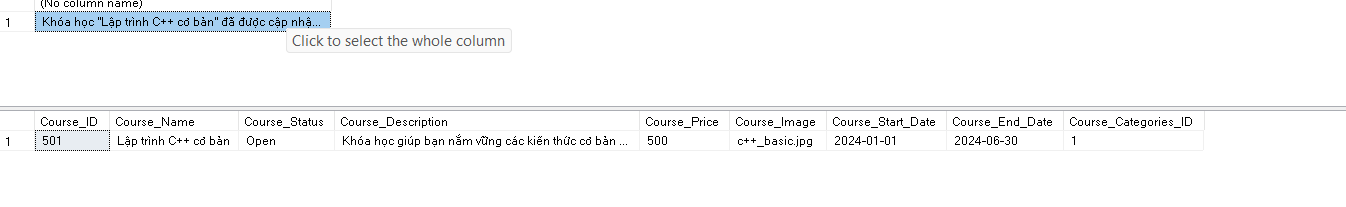
**Ví dụ**: Dòng lệnh EXEC UpdateCourse 501, 'Lập trình C++ cơ bản', 'Open', 'Khóa học giúp bạn nắm vững các kiến thức cơ bản về C++.', 500, 'c++\_basic.jpg', '2024-01-01', '2024-06-30', 1; sẽ cập nhật thông tin của khóa học có ID là 501 với các giá trị mới được cung cấp.

**TEST:**

EXEC UpdateCourse 501, N'Lập trình C++ cơ bản', 'Open', N'Khóa học giúp bạn nắm vững các kiến thức cơ bản về C++.', 500, 'c++\_basic.jpg', '2024-01-01', '2024-06-30', 1;

SELECT \* FROM Course

WHERE Course\_ID=501



1. **Thủ tục DELETE: DeleteCourse**

Đây là một thủ tục được dùng để xóa một khóa học khỏi bảng Course trong cơ sở dữ liệu. Nó thực hiện việc kiểm tra xem khóa học có tồn tại không trước khi xóa.

**Usecase**: Sử dụng trong các trường hợp cần xóa một khóa học không còn được sử dụng khỏi hệ thống quản lý khóa học.

**Thao tác chính**:

1. Kiểm tra xem khóa học có tồn tại không.
2. Xóa khóa học khỏi bảng Course nếu khóa học tồn tại.
3. Trả về thông báo xác nhận thành công.

**Input**:

* @p\_Course\_ID: ID của khóa học cần xóa.

**Output**:

* Thông báo xác nhận thành công, ví dụ: "Khóa học có Course\_ID = 501 đã được xóa thành công!".

**Lỗi**: Hàm sẽ trả về lỗi "Khóa học không tồn tại!" nếu khóa học không được tìm thấy trong cơ sở dữ liệu.

**Ví dụ**: Dòng lệnh EXEC DeleteCourse 501; sẽ xóa khóa học có ID là 501 khỏi bảng Course.\

**Test:**

EXEC DeleteCourse 501;

****

**2.2 TRIGGER**

**TRIGGER 1:**

**Mô tả Trigger:**

* **Tên trigger:** tr\_CheckCourseDates
* **Chức năng:** Trigger này được thiết kế để đảm bảo thời gian bắt đầu của khóa học luôn nhỏ hơn thời gian kết thúc của khóa học.
* **Sử dụng:** Trigger này được sử dụng trong tình huống thêm khóa học mới (insert) vào bảng Course.
* **Cách kích hoạt:** Trigger được kích hoạt (fire) *ngay trước* khi câu lệnh INSERT vào bảng Course được thực thi.
* **Thao tác chính:**
  + Kiểm tra xem có bất kỳ khóa học nào trong dữ liệu được thêm vào (inserted) có thời gian bắt đầu Course\_Start\_Date lớn hơn hoặc bằng thời gian kết thúc Course\_End\_Date hay không.
  + Nếu tìm thấy trường hợp này, trigger sẽ raise lỗi RAISERROR với thông báo "Thời gian bắt đầu phải nhỏ hơn thời gian kết thúc của khóa học.".
  + Nếu dữ liệu được thêm vào hợp lệ (thời gian bắt đầu nhỏ hơn thời gian kết thúc), trigger sẽ thực hiện việc thêm dữ liệu vào bảng Course bằng cách copy dữ liệu từ biến inserted vào bảng.
  + Cuối cùng, trigger sẽ thông báo "Khóa học đã được thêm thành công!" nếu thao tác insert thành công.

**Test: thêm vào một course có ngày bắt đầu sau ngày kết thúc**

INSERT INTO Course(Course\_ID, Course\_Name, Course\_Status, Course\_Description, Course\_Price, Course\_Image, Course\_Start\_Date, Course\_End\_Date, Course\_Categories\_ID)

VALUES

(100, N'Phân tích dữ liệu với Python', 'Open', N'Khóa học chuyên sâu về phân tích dữ liệu và trực quan hóa với Python.', 800, 'data\_analysis.jpg', '2024-09-01', '2024-08-31', 1);

****

**Trigger 2**

**Mô tả Trigger:**

* **Tên trigger:** tr\_CheckCouponValue
* **Chức năng:** Trigger này được thiết kế để kiểm tra xem thông tin của phiếu giảm giá khi được thêm vào có hợp lệ hay không.
* **Sử dụng:** Trigger này được sử dụng trong tình huống thêm phiếu giảm giá mới (insert) vào bảng Coupon.
* **Cách kích hoạt:** Trigger được kích hoạt (fire) *ngay trước* khi câu lệnh INSERT vào bảng Coupon được thực thi.
* **Thao tác chính:**
  + Kiểm tra xem có bất kỳ phiếu giảm giá nào trong dữ liệu được thêm vào (inserted) có giá trị giảm giá Coupon\_Value âm hay không. Nếu có, trigger sẽ báo lỗi "Giá trị giảm giá phải là số không âm.".
  + Kiểm tra xem có bất kỳ phiếu giảm giá nào trong dữ liệu được thêm vào (inserted) có loại giảm giá Coupon\_Type không phải là "Percent" hoặc "Fixed" hay không. Nếu có, trigger sẽ báo lỗi "Loại giảm giá phải là "Percent" hoặc "Fixed".".
  + Kiểm tra xem có bất kỳ phiếu giảm giá nào trong dữ liệu được thêm vào (inserted) có ngày bắt đầu Coupon\_Start\_Date lớn hơn hoặc bằng ngày kết thúc Coupon\_Expiry\_Date hay không. Nếu có, trigger sẽ báo lỗi "Ngày bắt đầu phải nhỏ hơn ngày kết thúc.".
  + Kiểm tra xem có bất kỳ phiếu giảm giá nào trong dữ liệu được thêm vào (inserted) có giá trị giảm giá tối đa Coupon\_Max\_Discount âm hay không. Nếu có, trigger sẽ báo lỗi "Giá trị giảm giá tối đa phải là số không âm.".
  + Nếu tất cả dữ liệu hợp lệ, trigger sẽ thêm phiếu giảm giá mới vào bảng Coupon bằng cách copy dữ liệu từ biến inserted vào bảng.
  + Cuối cùng, trigger sẽ thông báo "Phiếu giảm giá đã được thêm thành công!" nếu thao tác insert thành công.

**Test: Thêm vào một Coupon có Coupon\_Value âm**

INSERT INTO Coupon(Coupon\_ID, Coupon\_Title, Coupon\_Value, Coupon\_Type, Coupon\_Start\_Date, Coupon\_Expiry\_Date, Coupon\_Max\_Discount)

VALUE

(1, N'Giảm giá khai trương', -9, 'Percent', '2023-01-01', '2023-12-31', 30);



**Trigger 3**

**Mô tả Trigger:**

* **Tên trigger:** tr\_UpdateCoursePrice
* **Chức năng:** Trigger này được thiết kế để khi thêm một phiếu giảm giá vào khóa học, thì cập nhật giá khóa học
* **Sử dụng:** Trigger này được sử dụng trong tình huống thêm phiếu giảm giá mới (insert) vào bảng Coupon\_Course
* **Cách kích hoạt:** Trigger được kích hoạt sau khi thêm Coupon
* **Thao tác chính:**
  + Kiểm tra xem Coupon\_type là phần trăm hay giá trị cố định
  + Nếu là Percent thì trừ theo phần trăm, nếu là fixed thì trừ thẳng giá trị
  + Thông báo thành công

GO

CREATE TRIGGER tr\_UpdateCoursePrice

ON Course\_Coupon

AFTER INSERT

AS

BEGIN

UPDATE cr

SET Cr.Course\_Price =

CASE

WHEN c.Coupon\_Type = 'Percent'

THEN cr.Course\_Price \* (1 - c.Coupon\_Value/100.0)

ELSE

CASE

WHEN cr.Course\_Price > c.Coupon\_Value

THEN cr.Course\_Price - c.Coupon\_Value

ELSE 0

END

END

FROM Course cr

JOIN inserted i ON cr.Course\_ID = i.Course\_ID

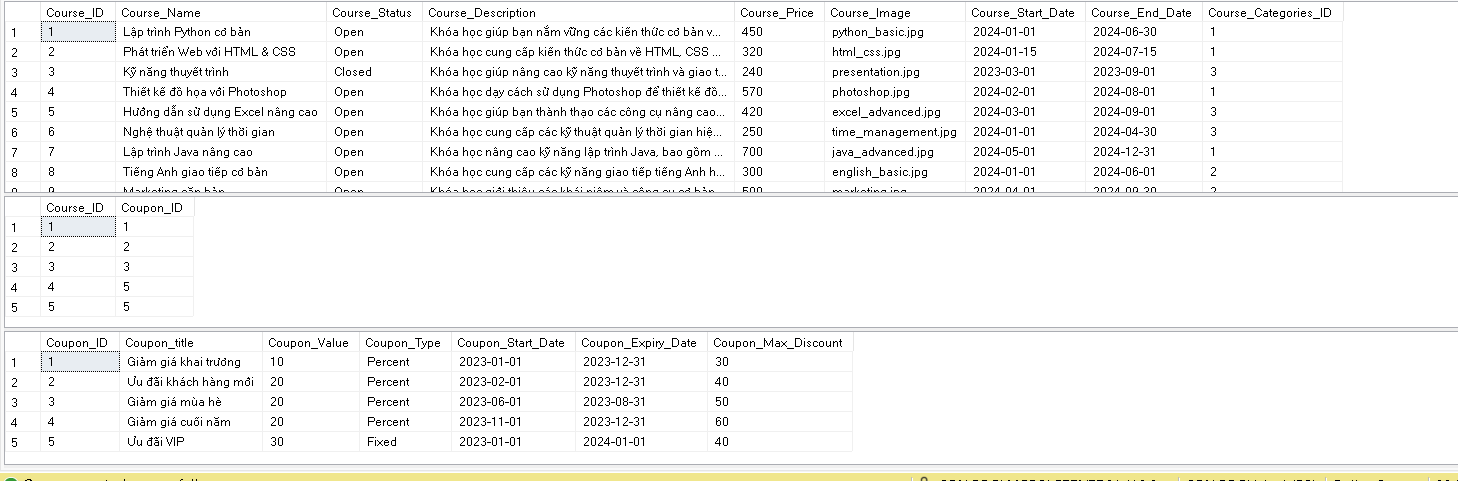
JOIN Coupon c ON c.Coupon\_ID = i.Coupon\_ID;

SELECT N'Đã cập nhật giá khóa học thành công!';

END;

**Test:**

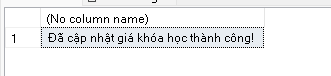
Bảng Course, Course\_Cart và Cart đang có giá trị như sau:

****

INSERT Course\_Coupon(Course\_ID,Coupon\_ID)

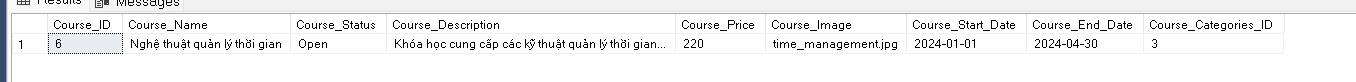
VALUES

(6,5);

****

SELECT \* FROM Course as C

WHERE C.Course\_ID=6

****

Ta thấy giá khóa học có id là 6 giảm từ 250 còn 220 do giá trị phiếu giảm giá là 30, loại là Fixed

**2.3 Thủ Tục**

**Thủ tục 1:**

Mô tả thủ tục: tính toán các giá trị tổng quan liên quan đến khóa học, để tính toán số lượng học viên, điểm đánh giá trung bình, số lượng bài học, số lượng bài tập, số lượng tài liệu, tổng thời lượng khóa học, tổng doanh thu dựa trên course\_id, thủ tục này được sử dụng khi người quản lý muốn xem lại tất cả thông số khóa học, đặc biệt số lượng học viên, điểm đánh giá trung bình và tổng doanh thu để đánh giá khóa học có vận hành hiệu quả hay không

Input: @p\_Course\_ID INT: ID của khóa học cần thống kê

Output: Bảng kết quả bao gồm các thông tin:

+ Course\_ID: ID của khóa học

+ Total\_Students: Tổng số học viên đăng ký

+ Average\_Rating: Điểm đánh giá trung bình

+ Total\_Lessons: Tổng số bài học

+ Total\_Exercises: Tổng số bài tập

+ Total\_Documents: Tổng số tài liệu

+ Total\_Duration\_Minutes: Tổng thời lượng (phút)

+ Total\_Revenue: Tổng doanh thu

GO

CREATE PROCEDURE GetCourseStatistics

@p\_Course\_ID INT

AS

BEGIN

-- Khai báo biến để lưu các thống kê

DECLARE @StudentCount INT,

@AvgRating FLOAT,

@LessonCount INT,

@ExerciseCount INT,

@DocumentCount INT,

@TotalDuration INT,

@TotalRevenue INT

-- Đếm số học viên đăng ký

SELECT @StudentCount = COUNT(Student\_ID)

FROM Course\_Student

WHERE Course\_ID = @p\_Course\_ID

-- Tính điểm đánh giá trung bình

SELECT @AvgRating = AVG(CAST(Reviews\_Score AS FLOAT))

FROM Reviews

WHERE Course\_ID = @p\_Course\_ID

-- Đếm số lượng bài học

SELECT @LessonCount = COUNT(\*)

FROM Lesson

WHERE Course\_ID = @p\_Course\_ID

-- Đếm số lượng bài tập

SELECT @ExerciseCount = COUNT(\*)

FROM Exercise

WHERE Course\_ID = @p\_Course\_ID

-- Đếm số lượng tài liệu

SELECT @DocumentCount = COUNT(\*)

FROM Document

WHERE Course\_ID = @p\_Course\_ID

-- Tính tổng thời lượng khóa học

SELECT @TotalDuration = SUM(Lesson\_Duration)

FROM Lesson

WHERE Course\_ID = @p\_Course\_ID

-- Tính tổng doanh thu

SELECT @TotalRevenue = SUM(c.Course\_Price)

FROM Course c

JOIN Course\_Order co ON c.Course\_ID = co.Course\_ID

JOIN Orders o ON co.Order\_ID = o.Orders\_ID

WHERE c.Course\_ID = @p\_Course\_ID

AND o.Orders\_Payment\_Status = 'Paid'

-- Trả về kết quả

SELECT

@p\_Course\_ID AS Course\_ID,

@StudentCount AS Total\_Students,

ISNULL(@AvgRating, 0) AS Average\_Rating,

@LessonCount AS Total\_Lessons,

@ExerciseCount AS Total\_Exercises,

@DocumentCount AS Total\_Documents,

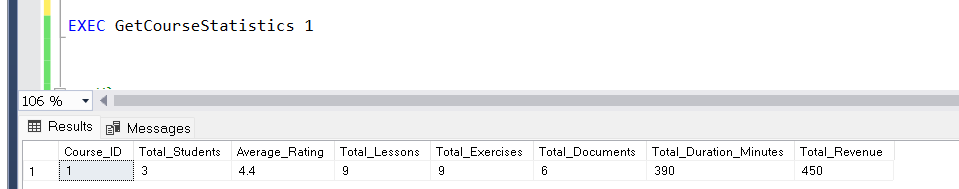
@TotalDuration AS Total\_Duration\_Minutes,

ISNULL(@TotalRevenue, 0) AS Total\_Revenue

END

Test case với course\_id =1

EXEC GetCourseStatistics 1



**Thủ tục 2:**

Mô tả thủ tục: Thủ tục này dùng để lọc và hiển thị danh sách các khóa học nằm trong một khoảng giá cụ thể. Thủ tục được sử dụng trong trường hợp học sinh muốn tìm kiếm khóa học phù hợp với ngân sách của họ. Thủ tục cho phép lọc khóa học có giá nằm giữa giá tối thiểu và tối đa, đồng thời thêm thông tin điểm đánh giá trung bình cho khóa học để học sinh cân nhắc

Input:

+ @p\_Min\_Price INT: Giá tối thiểu của khóa học (tùy chọn)

+ @p\_Max\_Price INT: Giá tối đa của khóa học (tùy chọn)

Output: Bảng kết quả bao gồm các thông tin:

+ Course\_id

+ Course\_Name

+ Course\_Price

+ Average\_rating

GO

CREATE PROCEDURE FilterCoursesByPrice

@p\_Min\_Price INT = NULL,

@p\_Max\_Price INT = NULL

AS

BEGIN

SELECT

c.Course\_ID,

c.Course\_Name,

c.Course\_Price,

AVG(CAST(r.Reviews\_Score AS FLOAT)) as Average\_Rating

FROM Course c

LEFT JOIN Reviews r ON c.Course\_ID = r.Course\_ID

WHERE

(@p\_Min\_Price IS NULL OR c.Course\_Price >= @p\_Min\_Price)

AND (@p\_Max\_Price IS NULL OR c.Course\_Price <= @p\_Max\_Price)

GROUP BY

c.Course\_ID,

c.Course\_Name,

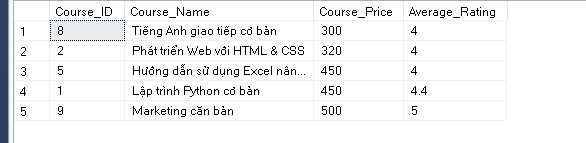
c.Course\_Price

ORDER BY c.Course\_Price ASC;

END;

Test:

EXEC FilterCoursesByPrice 300, 500;

****

**2.3 Hàm**

**Hàm 1:**

Mô tả: tính toán tổng giá trị đơn đặt hàng dựa vào ID đơn hàng

Input: @OrderID

Output: tổng giá trị đơn hàng có id là @OrderID

GO

CREATE FUNCTION CalculateOrderTotal

(

@OrderID INT

)

RETURNS DECIMAL(18,2)

AS

BEGIN

DECLARE @Total DECIMAL(18,2)

-- Tính tổng giá các khóa học trong đơn hàng

SELECT @Total = ISNULL(SUM(c.Course\_Price), 0)

FROM Orders o

JOIN Course\_Order co ON o.Orders\_ID = co.Order\_ID

JOIN Course c ON co.Course\_ID = c.Course\_ID

WHERE o.Orders\_ID = @OrderID

RETURN @Total

END;

Test:

SELECT dbo.CalculateOrderTotal(1) as "giá trị đơn hàng có id là 1";

****

**Hàm 2:**

Mô tả: Tính toán điểm giá của bài kiểm tra dựa vào số câu đúng và tổng số câu hỏi

Input:

+ @Test\_Order INT,

+ @Chapter\_Name NVARCHAR(255),

+ @Course\_ID INT

Output: Điểm của bài kiểm tra tương ứng với thứ tự bài kiểm tra, tên chương và course id

GO

CREATE FUNCTION CalculateTestScore

(

@Test\_Order INT,

@Chapter\_Name NVARCHAR(255),

@Course\_ID INT

)

RETURNS DECIMAL(4,2)

AS

BEGIN

DECLARE @Score DECIMAL(4,2)

DECLARE @TotalQuestions INT

DECLARE @CorrectAnswers INT

-- Lấy thông tin số câu hỏi và số câu đúng từ bảng Test

SELECT

@TotalQuestions = Test\_Number\_Of\_Questions,

@CorrectAnswers = Test\_Number\_Of\_Correct\_Answers

FROM Test

WHERE Test\_Order = @Test\_Order

AND Chapter\_Name = @Chapter\_Name

AND Course\_ID = @Course\_ID

-- Tính điểm theo công thức

SET @Score = CASE

WHEN @TotalQuestions > 0 THEN

CAST(@CorrectAnswers AS DECIMAL(4,2)) / @TotalQuestions \* 10

ELSE 0

END

RETURN @Score

END;

Test:

SELECT dbo.CalculateTestScore(1, N'Giới thiệu Python', 1) as "Điểm bài kiểm tra";

****