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

**HK231 - Lớp: L02**

**Nhóm**: SV58

**Danh sách thành viên**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stt | Họ tên | MSSV | Nội dung thực hiện | Ký tên xác nhận |
| 1 | Nguyễn Đinh Bằng | 2210298 | Tạo bảng, insert dữ liệu, function, procedure, trigger, backend kết nối database, web |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Chủ đề**: Trang web bán khoá học online

**1. Tạo bảng và dữ liệu mẫu**

**1.1 Các câu lệnh tạo bảng và ràng buộc**

CREATE TABLE …

…

CREATE TRIGGER … (*các trigger liên quan đến kiểm tra ràng buộc dữ liệu, nếu có*)

**1.2 Câu lệnh thêm dữ liệu**

INSERT INTO …

(*Nếu dữ liệu được nhập trực tiếp vào DBMS hoặc giao diện ứng dụng thì không cần ghi nhóm câu lệnh INSERT trên. Tuy nhiên, khuyến khích các nhóm chuẩn bị sẵn các câu lệnh INSERT để dùng khi test dữ liệu hoặc khôi phục lại dữ liệu gốc khi cần.*)

Kết quả dữ liệu của các bảng trong CSDL.

A screen shot of a computer

Description automatically generated

A close up of a screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a message box

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

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: Coupon (phiếu giảm giá)

1. Thủ tục INSERT:

**Chức năng của thủ tục**

Thủ tục InsertCoupon dùng để thêm một mã giảm giá (coupon) mới vào cơ sở dữ liệu, với các ràng buộc để đảm bảo tính hợp lệ của dữ liệu đầu vào.

**Use Case**

Thủ tục này được sử dụng trong các trường hợp sau:

* Khi cần thêm mã giảm giá cho các chiến dịch khuyến mãi.
* Đảm bảo rằng các thông tin của mã giảm giá (như tiêu đề, giá trị, loại giảm giá, ngày hiệu lực, v.v.) tuân theo các quy định về tính hợp lệ và không trùng lặp.

**Các thao tác chính**

1. **Kiểm tra dữ liệu đầu vào**:
   * Xác nhận rằng các trường bắt buộc không được để trống.
   * Kiểm tra các ràng buộc liên quan đến loại mã giảm giá (percent hoặc fixed), giá trị, và các ngày bắt đầu/kết thúc.
2. **Kiểm tra tính duy nhất**:
   * Xác nhận rằng tiêu đề mã giảm giá không bị trùng lặp.
3. **Thêm dữ liệu vào bảng**:
   * Thực hiện lệnh INSERT dữ liệu hợp lệ vào bảng Coupon trong cơ sở dữ liệu.
4. **Xử lý lỗi**:
   * Nếu có lỗi xảy ra, các thao tác sẽ được hoàn tác (rollback transaction), và thông báo lỗi sẽ được trả về.

Các ràng buộc được kiểm tra bao gồm:

* Các giá trị không được để trống.
* Loại mã giảm giá chỉ có thể là percent hoặc fixed.
* Giá trị phần trăm phải nằm trong khoảng từ 0 đến 100 (đối với loại percent).
* Giá trị giảm giá phải lớn hơn 0 (đối với loại fixed).
* Ngày bắt đầu không được ở quá khứ, và phải trước ngày hết hạn.
* Tiêu đề mã giảm giá không được trùng lặp với mã giảm giá đã tồn tại trong cơ sở dữ liệu.
* Giá trị giảm giá tối đa (nếu có) không được vượt quá giá trị giảm giá.

Input: Các thông số đầu vào của thủ tục bao gồm:

* @CouponTitle (NVARCHAR(255)): Tiêu đề mã giảm giá.
* @CouponValue (INT): Giá trị mã giảm giá (có thể là số tiền hoặc phần trăm).
* @CouponType (NVARCHAR(255)): Loại mã giảm giá (percent hoặc fixed).
* @CouponStartDate (DATE): Ngày bắt đầu hiệu lực của mã giảm giá.
* @CouponExpire (DATE): Ngày hết hạn của mã giảm giá.
* @CouponMaxDiscount (INT): Giá trị giảm giá tối đa (nếu có).
* @ErrorMessage (NVARCHAR(255), OUTPUT): Biến để lưu thông báo lỗi (nếu có).

Output:

* Nếu thành công: Thông báo "Thêm mã giảm giá thành công." được hiển thị.
* Nếu thất bại: Thông báo lỗi sẽ được trả về thông qua biến @ErrorMessage.

Câu lệnh tạo thủ tục:

*-- INSERT COUPON*

GO

CREATE PROCEDURE InsertCoupon

    @CouponTitle NVARCHAR(255),

    @CouponValue INT,

    @CouponType NVARCHAR(255),

    @CouponStartDate DATE,

    @CouponExpire DATE,

    @CouponMaxDiscount INT,

    @ErrorMessage NVARCHAR(255) OUTPUT

AS

BEGIN

    SET NOCOUNT ON;

    BEGIN TRY

*-- Validate nulls*

        IF @CouponTitle IS NULL OR @CouponValue IS NULL OR @CouponType IS NULL OR

           @CouponStartDate IS NULL OR @CouponExpire IS NULL OR @CouponMaxDiscount IS NULL

        BEGIN

            PRINT 'Lỗi: Dữ liệu không được để trống.'

            RAISERROR('Dữ liệu không được để trống.', 16, 1)

            RETURN

        END

*-- Validate title*

        IF LEN(TRIM(@CouponTitle)) = 0

        BEGIN

            PRINT 'Lỗi: Tiêu đề mã giảm giá không được để trống.'

            RAISERROR('Tiêu đề mã giảm giá không được để trống.', 16, 1)

            RETURN

        END

*-- Validate type*

        IF @CouponType NOT IN ('percent', 'fixed')

        BEGIN

            PRINT 'Lỗi: Loại mã giảm giá không hợp lệ.'

            RAISERROR('Loại mã giảm giá không hợp lệ.', 16, 1)

            RETURN

        END

*-- Validate value*

        IF @CouponValue <= 0

        BEGIN

            PRINT 'Lỗi: Giá trị mã giảm giá phải lớn hơn 0.'

            RAISERROR('Giá trị mã giảm giá phải lớn hơn 0.', 16, 1)

            RETURN

        END

*-- Validate dates*

        IF @CouponStartDate < GETDATE()

        BEGIN

            PRINT 'Lỗi: Ngày bắt đầu không được trong quá khứ.'

            RAISERROR('Ngày bắt đầu không được trong quá khứ.', 16, 1)

            RETURN

        END

        IF @CouponStartDate >= @CouponExpire

        BEGIN

            PRINT 'Lỗi: Ngày bắt đầu mã giảm giá phải trước ngày hết hạn.'

            RAISERROR('Ngày bắt đầu mã giảm giá phải trước ngày hết hạn.', 16, 1)

            RETURN

        END

*-- Validate max discount*

        IF @CouponMaxDiscount < 0

        BEGIN

            PRINT 'Lỗi: Giá trị giảm giá tối đa phải lớn hơn hoặc bằng 0.'

            RAISERROR('Giá trị giảm giá tối đa phải lớn hơn hoặc bằng 0.', 16, 1)

            RETURN

        END

*-- Validate percentage type*

        IF @CouponType = 'percent' AND (@CouponValue > 100 OR @CouponValue < 0)

        BEGIN

            PRINT 'Lỗi: Giá trị phần trăm phải nằm trong khoảng từ 0 đến 100.'

            RAISERROR('Giá trị phần trăm phải nằm trong khoảng từ 0 đến 100.', 16, 1)

            RETURN

        END

*-- Validate value type*

        IF @CouponType = 'fixed' AND @CouponValue <= 0

        BEGIN

            PRINT 'Lỗi: Giá trị giảm giá phải lớn hơn 0.'

            RAISERROR('Giá trị giảm giá phải lớn hơn 0.', 16, 1)

            RETURN

        END

*-- Check duplicate title*

        IF EXISTS (SELECT 1 FROM Coupon WHERE CouponTitle = @CouponTitle)

        BEGIN

            PRINT 'Lỗi: Tiêu đề mã giảm giá đã tồn tại.'

            RAISERROR('Tiêu đề mã giảm giá đã tồn tại.', 16, 1)

            RETURN

        END

*--nếu type = giá trị thì max discount phải <= value*

        IF @CouponType = 'fixed' AND @CouponMaxDiscount > @CouponValue

        BEGIN

            PRINT 'Lỗi: Giá trị giảm giá tối đa phải nhỏ hơn hoặc bằng giá trị giảm giá.'

            RAISERROR('Giá trị giảm giá tối đa phải nhỏ hơn hoặc bằng giá trị giảm giá.', 16, 1)

            RETURN

        END

*-- Insert with transaction*

        BEGIN TRANSACTION

            INSERT INTO Coupon (

                CouponTitle,

                CouponValue,

                CouponType,

                CouponStartDate,

                CouponExpire,

                CouponMaxDiscount

            )

            VALUES (

                @CouponTitle,

                @CouponValue,

                @CouponType,

                @CouponStartDate,

                @CouponExpire,

                @CouponMaxDiscount

            )

        COMMIT TRANSACTION

        PRINT 'Thêm mã giảm giá thành công.'

    END TRY

    BEGIN CATCH

        IF @@TRANCOUNT > 0

            ROLLBACK TRANSACTION

*-- PRINT 'Lỗi: ' + ERROR\_MESSAGE()*

        SET @ErrorMessage = ERROR\_MESSAGE()

        PRINT @ErrorMessage

        RETURN

    END CATCH

END

GO

Nhóm câu lệnh thực thi thủ tục mẫu:

DECLARE @ErrorMessage NVARCHAR(255);

EXEC InsertCoupon 'SUMMER2021', 20, 'percent', '2025-06-01', '2026-08-31', 100, @ErrorMessage OUTPUT;

Kết quả các màn hình hiển thị từ DBMS minh họa cho việc thực hiện thủ tục

A blue and white screen with red text

Description automatically generated

Trước khi thực hiện thủ tục:

A screenshot of a computer

Description automatically generated

Sau khi thực hiện thủ tục: Một coupon mới được thêm vào (SUMMER 2028, ID = 17)

A screenshot of a computer

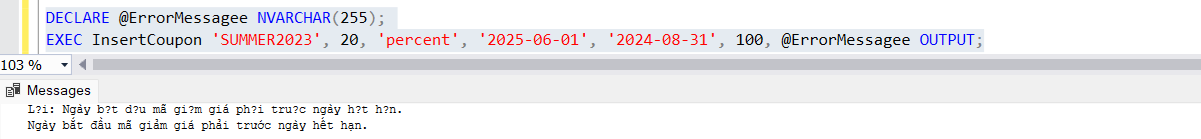
Description automatically generated

Khi ngày bắt đầu của mã giảm giá nhỏ hơn thời điểm hiện tại:

A white rectangular object with red numbers

Description automatically generated

Khi ngày bắt đầu nằm sau ngày hết hạn:



Khi tiêu đề mã giảm giá bị trùng

A blue and white rectangular object with red numbers

Description automatically generated

Khi loại mã giảm giá là “phần trăm” và giá trị mã giảm giá ngoài khoảng từ 0-100:

A white rectangular object with red text

Description automatically generated

Khi loại mã giảm giá là “giá trị” và giá trị giảm giá tối đa lớn hơn giá trị mã giảm giá:

A white text with red numbers

Description automatically generated

Khi insert một mã giảm giá có loại không phải là “fixed” hoặc “percent”:

A blue and white digital display

Description automatically generated with medium confidence

Ảnh chụp trước khi insert dữ liệu: Có 5 mã giảm giá.

A screenshot of a computer

Description automatically generated

Thực hiện insert một mã giảm giá mới: A screenshot of a computer

Description automatically generated

Sau khi đã insert: Có thêm mã giảm giá “WINTER”, tổng cộng có 6 mã giảm giá.A screenshot of a computer

Description automatically generated

Ảnh báo lỗi khi insert ngày bắt đầu trong quá khứ:

A screenshot of a computer

Description automatically generated

Ảnh báo lỗi khi ngày bắt đầu sau ngày kết thúc:

A screenshot of a computer

Description automatically generated

Ảnh báo lỗi khi loại mã giảm giá là “phần trăm” và giá trị lớn hơn 100 hoặc nhỏ hơn 0:

A screenshot of a computer

Description automatically generated

Ảnh báo lỗi khi loại mã giảm giá là “giá trị” và giá trị giảm giá tối đa lớn hơn giá trị mã giảm giá

A screenshot of a computer

Description automatically generated

1. Thủ tục UPDATE:

**Chức năng của thủ tục**

* Thủ tục UpdateCoupon được sử dụng để cập nhật thông tin của một mã giảm giá (coupon) đã tồn tại trong cơ sở dữ liệu, đồng thời đảm bảo các ràng buộc về tính hợp lệ của dữ liệu đầu vào.

**Use Case**

Thủ tục này được sử dụng trong các trường hợp:

* Khi cần thay đổi thông tin mã giảm giá, như tiêu đề, giá trị, loại giảm giá, ngày hiệu lực, hoặc giá trị giảm giá tối đa.
* Đảm bảo rằng các thông tin cập nhật không vi phạm các ràng buộc hoặc làm hỏng dữ liệu hiện có.

**Các thao tác chính**

1. **Kiểm tra tính tồn tại**:
   * Xác nhận rằng mã giảm giá cần cập nhật (CouponID) tồn tại trong cơ sở dữ liệu.
2. **Kiểm tra dữ liệu đầu vào**:
   * Ngày bắt đầu phải không nằm trong quá khứ và phải trước ngày hết hạn.
   * Giá trị giảm giá tối đa phải lớn hơn 0.
   * Loại mã giảm giá phải hợp lệ (percent hoặc fixed).
   * Đối với loại percent, giá trị phần trăm phải nằm trong khoảng từ 0 đến 100.
   * Đối với loại fixed, giá trị giảm giá phải lớn hơn 0.
3. **Cập nhật thông tin mã giảm giá**:
   * Thực hiện cập nhật các trường thông tin liên quan của mã giảm giá trong bảng Coupon.
4. **Xử lý lỗi**:
   * Nếu có lỗi xảy ra trong quá trình cập nhật, giao dịch sẽ được hoàn tác (rollback transaction), và thông báo lỗi sẽ được trả về.

***Mô tả thủ tục: UpdateCoupon***

***Chức năng của thủ tục***

*Thủ tục UpdateCoupon được sử dụng để cập nhật thông tin của một mã giảm giá (coupon) đã tồn tại trong cơ sở dữ liệu, đồng thời đảm bảo các ràng buộc về tính hợp lệ của dữ liệu đầu vào.*

***Use Case***

*Thủ tục này được sử dụng trong các trường hợp:*

* *Khi cần thay đổi thông tin mã giảm giá, như tiêu đề, giá trị, loại giảm giá, ngày hiệu lực, hoặc giá trị giảm giá tối đa.*
* *Đảm bảo rằng các thông tin cập nhật không vi phạm các ràng buộc hoặc làm hỏng dữ liệu hiện có.*

***Các thao tác chính***

1. ***Kiểm tra tính tồn tại****:*
   * *Xác nhận rằng mã giảm giá cần cập nhật (CouponID) tồn tại trong cơ sở dữ liệu.*
2. ***Kiểm tra dữ liệu đầu vào****:*
   * *Ngày bắt đầu phải không nằm trong quá khứ và phải trước ngày hết hạn.*
   * *Giá trị giảm giá tối đa phải lớn hơn 0.*
   * *Loại mã giảm giá phải hợp lệ (percent hoặc fixed).*
   * *Đối với loại percent, giá trị phần trăm phải nằm trong khoảng từ 0 đến 100.*
   * *Đối với loại fixed, giá trị giảm giá phải lớn hơn 0.*
3. ***Cập nhật thông tin mã giảm giá****:*
   * *Thực hiện cập nhật các trường thông tin liên quan của mã giảm giá trong bảng Coupon.*
4. ***Xử lý lỗi****:*
   * *Nếu có lỗi xảy ra trong quá trình cập nhật, giao dịch sẽ được hoàn tác (rollback transaction), và thông báo lỗi sẽ được trả về.*

**Input**

Các thông số đầu vào của thủ tục bao gồm:

* @CouponID (VARCHAR(20)): Mã định danh của mã giảm giá cần cập nhật.
* @CouponTitle (NVARCHAR(255)): Tiêu đề mã giảm giá mới.
* @CouponValue (INT): Giá trị mã giảm giá mới.
* @CouponType (VARCHAR(255)): Loại mã giảm giá mới (percent hoặc fixed).
* @CouponStartDate (DATE): Ngày bắt đầu hiệu lực mới của mã giảm giá.
* @CouponExpire (DATE): Ngày hết hạn mới của mã giảm giá.
* @CouponMaxDiscount (DECIMAL(10,2)): Giá trị giảm giá tối đa mới.
* @ErrorMessage (NVARCHAR(255), OUTPUT): Biến để lưu thông báo lỗi (nếu có)

**Output**

* **Kết quả trả về**:
  + Nếu thành công: Thông báo "Cập nhật mã giảm giá thành công." sẽ được hiển thị.
  + Nếu thất bại: Thông báo lỗi sẽ được trả về thông qua biến @ErrorMessage.

Các ràng buộc được kiểm tra trong thủ tục:

* Mã giảm giá cần cập nhật (CouponID) phải tồn tại trong bảng Coupon.
* Ngày bắt đầu không được ở quá khứ và phải trước ngày kết thúc.
* Loại mã giảm giá chỉ có thể là percent hoặc fixed.
* Giá trị phần trăm (percent) phải nằm trong khoảng từ 0 đến 100.
* Giá trị giảm giá (fixed) phải lớn hơn 0.
* Giá trị giảm giá tối đa phải lớn hơn 0.

Các câu lệnh tạo thủ tục:

|  |
| --- |
| *--UPDATE COUPON*  CREATE PROCEDURE UpdateCoupon      @CouponID VARCHAR(20),      @CouponTitle NVARCHAR(255),      @CouponValue INT,      @CouponType VARCHAR(255),      @CouponStartDate DATE,      @CouponExpire DATE,      @CouponMaxDiscount DECIMAL(10,2),      @ErrorMessage NVARCHAR(255) OUTPUT  AS  BEGIN      BEGIN TRY  *-- Validate input*          IF @CouponStartDate < GETDATE()          BEGIN              RAISERROR('Ngày bắt đầu không được trong quá khứ', 16, 1)              RETURN          END          IF @CouponStartDate > @CouponExpire          BEGIN              RAISERROR('Ngày bắt đầu không được lớn hơn ngày kết thúc', 16, 1)              RETURN          END            IF @CouponMaxDiscount <= 0          BEGIN              RAISERROR('Giá trị giảm giá phải lớn hơn 0', 16, 1)              RETURN          END  *--Validate coupon type*          IF @CouponType NOT IN ('percent', 'fixed')          BEGIN              RAISERROR('Loại mã giảm giá không hợp lệ', 16, 1)              RETURN          END  *--Validate coupon percentage range*          IF @CouponType = 'percent' AND (@CouponValue <= 0 OR @CouponValue > 100)          BEGIN              RAISERROR('Giá trị phần trăm phải nằm trong khoảng từ 0 đến 100', 16, 1)              RETURN          END  *--Validate coupon value*          IF @CouponType = 'fixed' AND @CouponValue <= 0          BEGIN              RAISERROR('Giá trị giảm giá phải lớn hơn 0', 16, 1)              RETURN          END  *--check if coupon exists*          IF NOT EXISTS (SELECT 1 FROM Coupon WHERE CouponID = @CouponID)          BEGIN              RAISERROR(N'Không tìm thấy mã giảm giá', 16, 1)              RETURN          END            BEGIN TRANSACTION              UPDATE Coupon              SET                  CouponTitle = @CouponTitle,                  CouponValue = @CouponValue,                  CouponType = @CouponType,                  CouponStartDate = @CouponStartDate,                  CouponExpire = @CouponExpire,                  CouponMaxDiscount = @CouponMaxDiscount              WHERE CouponID = @CouponID              IF @@ROWCOUNT = 0              BEGIN                  RAISERROR('Không tìm thấy mã giảm giá', 16, 1)                  ROLLBACK TRANSACTION                  RETURN              END          COMMIT TRANSACTION          PRINT 'Cập nhật mã giảm giá thành công.'      END TRY      BEGIN CATCH          IF @@TRANCOUNT > 0              ROLLBACK TRANSACTION          PRINT 'Lỗi: ' + ERROR\_MESSAGE()          SET @ErrorMessage = ERROR\_MESSAGE()          RETURN      END CATCH  END  GO |

Nhóm câu lệnh thực thi thủ tục mẫu:

|  |
| --- |
| *DECLARE @ErrorMessage2 NVARCHAR(255);*  *EXEC UpdateCoupon '1', 'SUMMER2022', 25, 'percent', '2025-06-01', '2026-08-31', 100, @ErrorMessage2 OUTPUT;* |

Màn hình DBMS minh hoạ việc thực hiện thủ tục: Sửa đổi mã giảm giá có ID = 1 thành “SUMMER2022”:

A screenshot of a computer

Description automatically generated

Báo lỗi khi loại mã giảm giá là “phần trăm” và giá trị ngoài khoảng 0-100:

A blue and white bar with red numbers

Description automatically generated

Khi chỉnh sửa mã giảm giá và sửa loại mã giảm giá thành một chuỗi khác ngoài “fixed” và “percent”:

A close-up of a computer screen

Description automatically generated

Khi loại mã giảm giá là “giá trị” và giá trị giảm giá tối đa lớn hơn giá trị mã giảm giá:



Minh hoạ khi kết nối với web:

Trước khi chỉnh sửa:

A screenshot of a computer

Description automatically generated

Bấm vào “Edit” và chỉnh sửa tiêu đề thành “SUMMER2025”:

A screenshot of a coupon

Description automatically generated

Nhấn “Update”: Tiêu đề của mã giảm giá đã đổi thành SUMMER2025

A screenshot of a computer

Description automatically generated

Minh hoạ báo lỗi khi chỉnh sửa mã giảm giá vi phạm các điều kiện:

A screenshot of a computer

Description automatically generated

1. Thủ tục DELETE:

**Chức năng của thủ tục**

Thủ tục DeleteCoupon được sử dụng để xóa một mã giảm giá (coupon) khỏi cơ sở dữ liệu, đồng thời xóa các dữ liệu liên quan trong bảng phụ thuộc (Edit).

**Use Case**

Thủ tục này được sử dụng trong các trường hợp:

* Khi cần hủy bỏ mã giảm giá không còn sử dụng.
* Đảm bảo dữ liệu liên quan (như các chỉnh sửa lịch sử) được xóa đồng thời để duy trì tính toàn vẹn dữ liệu.

Thủ tục đảm bảo rằng:

* Mã giảm giá chỉ được xóa khi nó tồn tại trong cơ sở dữ liệu.
* Các dữ liệu liên quan (chẳng hạn như lịch sử chỉnh sửa) cũng được xóa để đảm bảo không còn dữ liệu "mồ côi" (orphaned records).
* Sử dụng giao dịch (transaction) để đảm bảo tính toàn vẹn dữ liệu, đặc biệt khi có nhiều bảng phụ thuộc.

**Các thao tác chính**

1. **Kiểm tra tính tồn tại**:
   * Xác nhận rằng mã giảm giá cần xóa (CouponID) tồn tại trong bảng Coupon.
2. **Xóa dữ liệu liên quan**:
   * Xóa các bản ghi trong bảng phụ thuộc (Edit) có liên kết với mã giảm giá.
3. **Xóa mã giảm giá**:
   * Thực hiện xóa mã giảm giá khỏi bảng Coupon.
4. **Xử lý lỗi**:
   * Nếu có lỗi xảy ra trong quá trình xóa, giao dịch sẽ được hoàn tác (rollback transaction), và thông báo lỗi sẽ được hiển thị.

**Input**

Các thông số đầu vào của thủ tục bao gồm:

* @CouponID (VARCHAR(20)): Mã định danh của mã giảm giá cần xóa.

**Output**

* **Kết quả trả về**:
  + Nếu thành công: Thông báo "Xóa mã giảm giá thành công." sẽ được hiển thị.
  + Nếu thất bại: Thông báo lỗi sẽ được hiển thị qua cơ chế RAISERROR.

**Các bước thực hiện**

1. **Kiểm tra tính tồn tại của mã giảm giá**:
   * Nếu không tìm thấy mã giảm giá, thủ tục sẽ kết thúc và thông báo lỗi "Không tìm thấy mã giảm giá".
2. **Xóa dữ liệu liên quan trong bảng Edit**:
   * Xóa các bản ghi có liên kết với mã giảm giá thông qua cột EditCouponID.
3. **Xóa mã giảm giá trong bảng Coupon**:
   * Xóa bản ghi có mã định danh trùng với @CouponID.
   * Nếu không thể xóa, thông báo lỗi "Không thể xóa mã giảm giá" sẽ được hiển thị.
4. **Xử lý giao dịch**:
   * Nếu tất cả các bước thành công, giao dịch sẽ được hoàn tất (commit transaction).
   * Nếu xảy ra lỗi trong bất kỳ bước nào, giao dịch sẽ được hoàn tác (rollback transaction).

Các câu lệnh tạo thủ tục:

|  |
| --- |
| *--DELETE COUPON*  CREATE PROCEDURE DeleteCoupon      @CouponID VARCHAR(20)  AS  BEGIN      BEGIN TRY          BEGIN TRANSACTION  *-- Check if coupon exists*              IF NOT EXISTS (SELECT 1 FROM Coupon WHERE CouponID = @CouponID)              BEGIN                  RAISERROR('Không tìm thấy mã giảm giá', 16, 1)                  RETURN              END  *--DELETE Edit*              DELETE FROM Edit              WHERE EditCouponID = @CouponID  *-- Delete the coupon*              DELETE FROM Coupon              WHERE CouponID = @CouponID              IF @@ROWCOUNT > 0                  PRINT 'Xóa mã giảm giá thành công.'              ELSE                  RAISERROR('Không thể xóa mã giảm giá', 16, 1)          COMMIT TRANSACTION      END TRY      BEGIN CATCH          IF @@TRANCOUNT > 0              ROLLBACK TRANSACTION          PRINT 'Lỗi: ' + ERROR\_MESSAGE()      END CATCH  END  GO |

Minh hoạ việc sử dụng thủ tục:

*EXEC DeleteCoupon '1';*

Màn hình DBMS trước khi thực hiện thủ tục delete:

*A screenshot of a computer

Description automatically generated*

Sau khi thực hiện thủ tục delete với id = 7:

*A screenshot of a computer

Description automatically generated*

Nếu input là một coupon id không tồn tại:

A screenshot of a computer

Description automatically generated

Minh hoạ việc kết nối với web: Trước khi nhấn “Delete” coupon id = 6:

**A screenshot of a computer

Description automatically generated**

Sau khi nhấn vào Delete coupon id = 6, coupon đó biến mất: A screenshot of a computer

Description automatically generated

**2.2 TRIGGER**

*Lưu ý: các trigger để kiểm tra các ràng buộc dữ liệu mà có thể định nghĩa được trong câu lệnh CREATE TABLE sẽ không được tính điểm.*

*Ví dụ: kiểm tra thuộc tính MSSV có duy nhất không (Unique), kiểm tra email có đúng định dạng không (dùng REGULAR EXPRESSION hoặc Like), số điện thoại sinh viên không được để trống (Not null)*

1. **Trigger 1**

**Mô tả nhóm trigger**

Đây là nhóm gồm 2 trigger phối hợp với nhau để tự động tính toán và cập nhật tổng giá trị đơn hàng khi có bất kỳ thay đổi nào liên quan đến đơn hàng, bao gồm:

* TR\_Orders\_Calculate\_Total: Kích hoạt khi có thay đổi trong bảng Orders (thêm đơn hàng mới hoặc cập nhật mã giảm giá)
* TR\_CourseOrder\_Calculate\_Total: Kích hoạt khi có thay đổi trong bảng CourseOrder (thêm/sửa/xóa khóa học trong đơn hàng)

Nhóm trigger này được sử dụng trong các trường hợp:

* Khi tạo đơn hàng mới
* Khi thêm hoặc xóa khóa học khỏi đơn hàng
* Khi thay đổi mã giảm giá áp dụng cho đơn hàng
* Đảm bảo tổng giá trị đơn hàng luôn được cập nhật tự động và chính xác

**Các thao tác chính**

1. Thu thập thông tin đơn hàng:
   * Xác định OrderID cần cập nhật
   * Lấy thông tin về mã giảm giá (nếu có)
   * Tính tổng giá các khóa học trong đơn hàng
2. Nếu thay đổi liên quan đến đơn hàng đã được thanh toán thì sẽ báo lỗi
3. Xử lý mã giảm giá:
   * Kiểm tra tính hợp lệ và thời hạn của mã giảm giá
   * Tính toán giá trị giảm dựa trên loại giảm giá:
     + Phần trăm: Tính % trên tổng giá trị
     + Giá trị cố định: Áp dụng trực tiếp số tiền giảm
4. Áp dụng các ràng buộc giảm giá:
   * Đảm bảo giá trị giảm không vượt quá giới hạn tối đa
   * Với giảm giá cố định: không vượt quá tổng giá trị đơn hàng
   * Với giảm giá phần trăm: tính chính xác số tiền được giảm
5. Cập nhật đơn hàng:
   * Tính tổng giá trị cuối cùng sau khi áp dụng giảm giá
   * Cập nhật vào trường TotalAmount của bảng Orders

Câu lệnh tạo trigger:

|  |
| --- |
| CREATE TRIGGER TR\_Orders\_Calculate\_Total  ON Orders  AFTER UPDATE  AS  BEGIN  -- Declare variables  DECLARE @OrderID INT  DECLARE @PaymentStatus NVARCHAR(255)  -- Get OrderID and payment status from inserted record  SELECT @OrderID = OrderID, @PaymentStatus = OrderPaymentStatus  FROM inserted  -- Check if order is already paid  IF EXISTS (  SELECT 1  FROM Orders  WHERE OrderID = @OrderID  AND OrderPaymentStatus = 'paid'  )  BEGIN  RAISERROR ('Không thể thay đổi đơn hàng đã thanh toán.', 16, 1)  ROLLBACK TRANSACTION  RETURN  END  -- Declare variables  DECLARE @CouponID INT  DECLARE @SubTotal FLOAT  DECLARE @Discount FLOAT  DECLARE @FinalTotal FLOAT  DECLARE @CouponType NVARCHAR(255)  DECLARE @CouponValue INT  DECLARE @CouponMaxDiscount INT  DECLARE @CurrentDate DATE    -- Get current date  SET @CurrentDate = GETDATE()  -- Get the affected OrderID from inserted records  SELECT @OrderID = OrderID, @CouponID = CouponID  FROM inserted  -- Calculate subtotal by summing up course prices  SELECT @SubTotal = COALESCE(SUM(c.CoursePrice), 0)  FROM CourseOrder co  JOIN Course c ON co.CourseID = c.CourseID  WHERE co.OrderID = @OrderID  -- If there's a coupon, calculate discount  IF @CouponID IS NOT NULL  BEGIN  -- Get coupon details with date validation  SELECT  @CouponType = CouponType,  @CouponValue = CouponValue,  @CouponMaxDiscount = CouponMaxDiscount  FROM Coupon  WHERE CouponID = @CouponID  AND CouponStartDate <= @CurrentDate  AND CouponExpire >= @CurrentDate  -- Only calculate discount if coupon is valid (not NULL from previous query)  IF @CouponType IS NOT NULL  BEGIN  -- Calculate discount based on coupon type  IF @CouponType = 'percent'  BEGIN  SET @Discount = (@SubTotal \* @CouponValue) / 100.0  -- Check if discount exceeds max discount  IF @Discount > @CouponMaxDiscount  SET @Discount = @CouponMaxDiscount  END  ELSE -- Fixed value  BEGIN  SET @Discount = @CouponValue  -- Check if discount exceeds max discount  IF @Discount > @CouponMaxDiscount  SET @Discount = @CouponMaxDiscount  -- Check if discount exceeds subtotal  IF @Discount > @SubTotal  SET @Discount = @SubTotal  END  END  ELSE  BEGIN  SET @Discount = 0 -- Coupon is not valid due to dates  END  END  ELSE  BEGIN  SET @Discount = 0  END  -- Calculate final total  SET @FinalTotal = @SubTotal - @Discount  -- Update the TotalAmount in Orders table  UPDATE Orders  SET TotalAmount = @FinalTotal  WHERE OrderID = @OrderID  END; |

|  |
| --- |
| CREATE TRIGGER TR\_CourseOrder\_Calculate\_Total  ON CourseOrder  AFTER INSERT, UPDATE, DELETE  AS  BEGIN  IF EXISTS (  SELECT 1  FROM Orders o  WHERE o.OrderPaymentStatus = 'paid'  AND o.OrderID IN (  SELECT OrderID FROM inserted  UNION  SELECT OrderID FROM deleted  )  )  BEGIN  RAISERROR ('Không thể thay đổi đơn hàng đã thanh toán.', 16, 1)  ROLLBACK TRANSACTION  RETURN  END  DECLARE @CurrentDate DATE  SET @CurrentDate = GETDATE()  UPDATE Orders  SET TotalAmount = (  SELECT COALESCE(SUM(c.CoursePrice), 0) -  CASE  WHEN cp.CouponType = N'percent'  AND cp.CouponStartDate <= @CurrentDate  AND cp.CouponExpire >= @CurrentDate THEN  CASE  WHEN (COALESCE(SUM(c.CoursePrice), 0) \* cp.CouponValue / 100.0) > cp.CouponMaxDiscount  THEN cp.CouponMaxDiscount  ELSE (COALESCE(SUM(c.CoursePrice), 0) \* cp.CouponValue / 100.0)  END  WHEN cp.CouponType = N'value'  AND cp.CouponStartDate <= @CurrentDate  AND cp.CouponExpire >= @CurrentDate THEN  CASE  WHEN cp.CouponValue > cp.CouponMaxDiscount THEN cp.CouponMaxDiscount  WHEN cp.CouponValue > COALESCE(SUM(c.CoursePrice), 0) THEN COALESCE(SUM(c.CoursePrice), 0)  ELSE cp.CouponValue  END  ELSE 0  END  FROM CourseOrder co2  JOIN Course c ON co2.CourseID = c.CourseID  LEFT JOIN Orders o ON co2.OrderID = o.OrderID  LEFT JOIN Coupon cp ON o.CouponID = cp.CouponID  WHERE co2.OrderID = Orders.OrderID  GROUP BY co2.OrderID, cp.CouponType, cp.CouponValue, cp.CouponMaxDiscount,  cp.CouponStartDate, cp.CouponExpire  )  WHERE OrderID IN (  SELECT OrderID FROM inserted  UNION  SELECT OrderID FROM deleted  );  END; |

**Testcase:**

Tạo đơn hàng mới có OrderID là 103, lấy ngày hiện tại bằng hàm GETDATE(), tổng giá trị đơn hàng ban đầu là 0 và chưa có mã giảm giá nào

|  |
| --- |
| INSERT INTO Orders (OrderID, OrderPaymentStatus, OrderDate, StudentID,TotalAmount,CouponID)  VALUES  (103, 'unpaid', GETDATE(), 15,0,NULL); |

Kiểm tra đơn hàng có OrderID là 103

|  |
| --- |
| SELECT \*  FROM Orders  WHERE OrderID=103 |

Kết quả như sau, TotalAmmount bằng 0 vì đơn hàng này chưa có khóa học nào



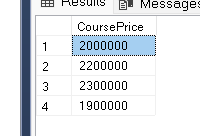
Thêm 2 khóa học có CourseID là 1 và 2 vào đơn hàng có id là 103 trong bảng CourseOrder

|  |
| --- |
| INSERT INTO CourseOrder( CourseID,OrderID)  VALUES  (1,103),  (2,103); |

Kiểm tra giá tiền của 2 khóa học có id là 1 , 2, 3 và 4

|  |
| --- |
| SELECT CoursePrice  FROM Course  WHERE CourseID IN(1,2,3,4) |

Kết quả như sau:



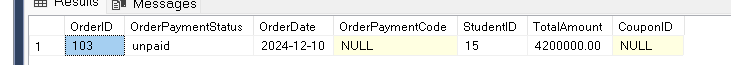
Khóa học 1 có giá là 2 000 000

Khóa học 2 có giá là 2 200 000

Khóa học 3 có giá 2 300 000

Khóa học 4 có giá 1 900 000

Ta kiểm tra lại thông tin đơn hàng có ID là 103, kết quả như sau:



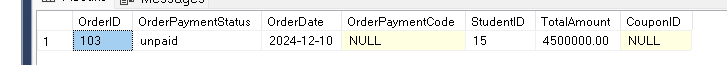
Vậy trigger TR\_CourseOrder\_Calculate\_Total đã được chạy đẻ tính toán tổng tiền khóa học khi có thay đổi trên bảng CourseOrder là 2 000 000 + 2 200 000 = 4 200 000

Thử UPDATE bảng CourseOrder có OrderID là 103 , có khóa học 1 thay thành khóa học 3

|  |
| --- |
| UPDATE CourseOrder  SET CourseID=3  WHERE CourseID=1 AND OrderID=103 |

Khi đó đơn hàng có OrderID là 103 sẽ có khóa học 2 và khóa học 3, tổng tiền là:

2 200 000 + 2 300 000 = 4 500 000



Thử DELETE bảng CourseOrder có id đơn hàng là 103, khóa học 2

|  |
| --- |
| DELETE FROM CourseOrder  WHERE CourseID=2 AND OrderID=103 |

Khi đó đơn hàng có OrderID là 103 chỉ còn khóa học 3, tổng tiền sẽ là 2 300 000



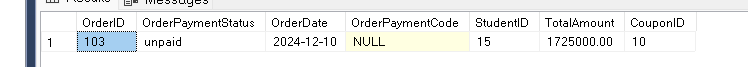
INSERT phiếu giảm giá vào:

|  |
| --- |
| INSERT INTO Coupon (CouponID,CouponTitle, CouponValue, CouponType, CouponStartDate, CouponExpire, CouponMaxDiscount)  VALUES  -- Giảm theo phần trăm  (9,N'Giảm 15% khóa học mới', 15,'percent' , '2024-01-01', '2025-04-28', 1500000),  (10,N'Giảm 25% khóa học sql', 25, 'percent', '2024-05-01', '2025-03-28', 1500000),  (11,N'Giảm 300k nhân diệp giáng sinh', 300000, 'value', '2024-01-15', '2024-12-30', 1500000); |

Thử UPDATE đơn hàng có OrderID là 103 , gắn phiếu giảm giá có CouponID là 10 cho đơn hàng

|  |
| --- |
| UPDATE Orders  SET CouponID=10  WHERE OrderID=103 |

Khi đó đơn hàng có OrderID là 103 sẽ giảm giá 25%, giá trị giảm là 2 300 000 x 25%= 575 000 , vì giá trị giảm bé hơn 1 500 000 nên giữ nguyên giá trị giảm này, khi đó giá trị đơn hàng còn : 2 300 000 – 575 000 = 1 725 000



Ta cập nhật CourseOrder để đơn hàng có OrderID là 103 có thêm khóa học 1, 2 và 4

|  |
| --- |
| INSERT INTO CourseOrder( CourseID,OrderID)  VALUES  (1,103),  (2,103),  (4,103); |

Khi đó tổng giá trị khóa học sẽ là

2 000 000 + 2 200 00 + 2 300 000 + 1 900 000 = 8 400 000

Vì áp dụng phiếu giảm giá 25%, nên giá trị giảm sẽ là 8 400 000 x 25%=2 100 000, vì giá trị giảm này lớn hơn 1 500 000 nên chỉ được giảm 1 500 000, tổng giá tiền khóa học sẽ còn:

8 400 000 – 1 500 000= 6 900 000



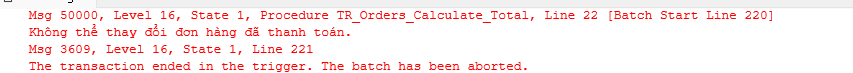
Ta thêm 1 đơn hàng có ID là 104, với trạng thái thanh toán là “paid”

|  |
| --- |
| INSERT INTO Orders (OrderID, OrderPaymentStatus, OrderDate, StudentID,TotalAmount,CouponID)  VALUES  (104, 'paid', GETDATE(), 15,0,NULL); |

Nếu ta thử gắn phiếu giảm giá cho đơn hàng này

|  |
| --- |
| UPDATE Orders  SET CouponID=10  WHERE OrderID=104 |

Sẽ in ra lỗi, vì đơn hàng đã thanh toán rồi không thay đổi được nữa



**b. Trigger 2**

**Mô tả trigger:**

Chức năng của trigger Trigger TR\_Review\_Update\_Course\_Rating được sử dụng để tự động tính toán và cập nhật điểm đánh giá trung bình của khóa học (CourseAverageRating) mỗi khi có thay đổi trong bảng Review. Trigger được kích hoạt (fire) sau khi có bất kỳ thao tác thêm, sửa, xóa đánh giá (AFTER INSERT, UPDATE, DELETE).

Trigger này được sử dụng trong các trường hợp:

* Khi học viên thêm đánh giá mới cho khóa học
* Khi học viên chỉnh sửa đánh giá đã tồn tại
* Khi đánh giá bị xóa khỏi hệ thống
* Đảm bảo điểm đánh giá trung bình của khóa học luôn được cập nhật tự động và chính xác

**Các thao tác chính**

1. Thu thập các CourseID bị ảnh hưởng:
   * Lấy danh sách CourseID từ cả các đánh giá mới/được cập nhật (inserted)
   * Lấy danh sách CourseID từ các đánh giá bị xóa/thay đổi (deleted)
   * Kết hợp (UNION) các CourseID để có danh sách đầy đủ các khóa học cần cập nhật
2. Tính toán điểm trung bình:
   * Tính trung bình cộng của tất cả ReviewScore cho mỗi khóa học
   * Sử dụng CAST để chuyển đổi điểm số thành kiểu FLOAT để có kết quả chính xác
   * Nhóm theo CourseID để tính riêng cho từng khóa học
3. Cập nhật khóa học:
   * Cập nhật giá trị CourseAverageRating trong bảng Course
   * Chỉ cập nhật các khóa học bị ảnh hưởng bởi thay đổi

Câu lệnh tạo (các) trigger:

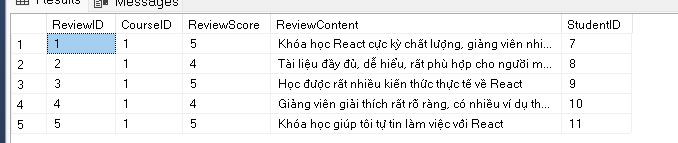
|  |
| --- |
| CREATE OR ALTER TRIGGER TR\_Review\_Update\_Course\_Rating  ON Review  AFTER INSERT, UPDATE, DELETE  AS  BEGIN  -- Update Course average rating for all affected courses  UPDATE Course  SET CourseAverageRating = (  SELECT AVG(CAST(ReviewScore AS FLOAT))  FROM Review r  WHERE r.CourseID = Course.CourseID  GROUP BY r.CourseID  )  WHERE CourseID IN (  -- Get affected CourseIDs from both inserted and deleted  SELECT CourseID FROM inserted  UNION  SELECT CourseID FROM deleted  );  END; |

**Test case**

Đầu tiên ta kiểm tra dữ liệu mẫu bảng Review, kiểm tra các đánh giá cho các khóa học có CourseID=1

|  |
| --- |
| SELECT \*  FROM Review  WHERE CourseID=1 |

Kết quả:



Vậy ban đầu Course 1 có 5 đánh giá, điểm số đánh giá lần lượt là 5, 4, 5, 4, 5

Khi thêm một đánh giá cho Course 1 bằng câu lệnh như sau:

|  |
| --- |
| INSERT INTO Review (ReviewID, CourseID, ReviewScore, ReviewContent, StudentID) VALUES  (6, 1, 1, N'Khóa học này quá mông lung', 15); |

Vậy đánh giá trung bình sau khi thêm dữ liệu vô là: (5 + 4 + 5 + 4 + 5 + 1)/6 =4

Ta kiểm tra CourseAverageRating của Course có CourseID là 1

|  |
| --- |
| SELECT CourseAverageRating  FROM Course  WHERE CourseID=1 |

Kết quả như sau:



Ta xóa đi một hàng trong Review có ReviewID là 1 và CourseID là 1

Khi đó đánh giá trung bình sẽ là : (4+5+4+5+1)/5=3.8

|  |
| --- |
| DELETE FROM Review  WHERE ReviewID=1 AND CourseID=1 |

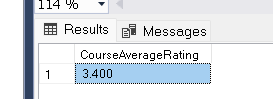
Ta kiểm tra lại CourseAverageRating của Course có CourseID là 1



Ta cập nhật Review có ReviewID là 2 và CourseID là 1, cập nhật lại ReviewScore là 2

|  |
| --- |
| UPDATE Review  SET ReviewScore=2  WHERE ReviewID=2 AND CourseID=1 |

Khi đó đánh giá trung bình sẽ là : (2+5+4+5+1)/5=3.4



**2.3 THỦ TỤC**

1. **Thủ tục 1**

**Mô tả thủ tục:** Thủ tục GetCoursesInCategoryByMinRating được sử dụng để tìm kiếm và hiển thị thông tin các khóa học trong một danh mục cụ thể, có điểm đánh giá trung bình cao hơn hoặc bằng một mức điểm cho trước.

Thủ tục này được sử dụng trong các trường hợp:

* Khi người dùng muốn tìm kiếm khóa học theo danh mục
* Khi cần lọc khóa học theo mức đánh giá tối thiểu
* Khi cần xem thống kê về các khóa học chất lượng cao trong một danh mục
* Hỗ trợ việc đề xuất khóa học cho học viên dựa trên đánh giá

**Các thao tác chính:**

1. Kiểm tra tính hợp lệ của đầu vào:
   * Rating phải nằm trong khoảng từ 1 đến 5
2. Tìm kiếm khóa học theo tiêu chí:
   * Tên danh mục chứa chuỗi tìm kiếm
   * Điểm đánh giá trung bình đạt ngưỡng tối thiểu
   * Có ít nhất một đánh giá
3. Trả về thông tin chi tiết:
   * Tên danh mục và khóa học
   * Số lượng đánh giá
   * Điểm trung bình
   * Giá và thời gian khóa học

Input:

 @CategoryName (NVARCHAR(255)): Tên danh mục cần tìm kiếm

 @MinRating (DECIMAL(10,3)): Điểm đánh giá tối thiểu

Output:

Kết quả trả về bảng gồm các cột:

* CategoryName: Tên danh mục
* CourseName: Tên khóa học
* TotalReviews: Tổng số đánh giá
* CourseAverageRating: Điểm trung bình
* CoursePrice: Giá khóa học
* CourseStartDate: Ngày bắt đầu
* CourseEndDate: Ngày kết thúc

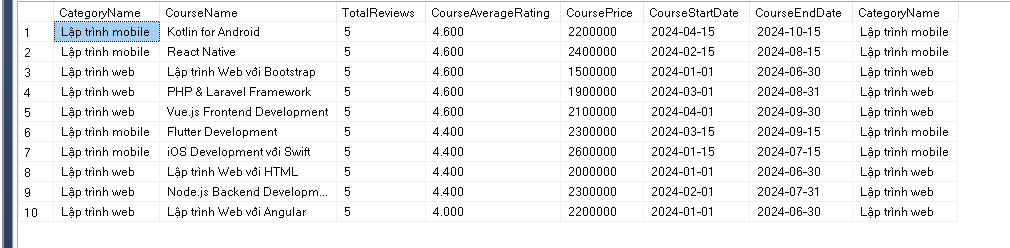
Câu lệnh tạo thủ tục:

|  |
| --- |
| CREATE PROCEDURE GetCoursesInCategoryByMinRating  @CategoryName NVARCHAR(255),  @MinRating DECIMAL(10,3)  AS  BEGIN  -- Kiểm tra tham số đầu vào  IF @MinRating < 1 OR @MinRating > 5  BEGIN  RAISERROR('Rating phải nằm trong khoảng từ 1 đến 5', 16, 1)  RETURN  END    SELECT  cat.CategoryName,  c.CourseName,  COUNT(r.ReviewID) as TotalReviews,  c.CourseAverageRating,  c.CoursePrice,  c.CourseStartDate,  c.CourseEndDate,  cat.CategoryName  FROM Course c  INNER JOIN Category cat ON c.CategoryID = cat.CategoryID  LEFT JOIN Review r ON c.CourseID = r.CourseID  WHERE  cat.CategoryName LIKE N'%' + @CategoryName + '%'  AND c.CourseAverageRating >= @MinRating  GROUP BY  cat.CategoryName,  c.CourseName,  c.CourseAverageRating,  c.CoursePrice,  c.CourseStartDate,  c.CourseEndDate,  cat.CategoryName  HAVING  COUNT(r.ReviewID) > 0 -- Thay đổi điều kiện HAVING  ORDER BY  c.CourseAverageRating DESC;  END; |

Nhóm câu lệnh thực thi thủ tục mẫu:

|  |
| --- |
| EXEC GetCoursesInCategoryByMinRating @CategoryName = N'Lập trình', @MinRating = 4.0 |

Kết quả các màn hình hiển thị từ DBMS minh họa cho việc thực hiện thủ tục



1. **Thủ tục 2**

**Mô tả thủ tục:** Thủ tục sp\_GetValidCoupons được sử dụng để lấy danh sách các mã giảm giá còn hiệu lực tại thời điểm hiện tại, kèm theo thống kê số lượng các khóa học đã áp dụng mã giảm giá đó**.** Thủ tục này được sử dụng trong các trường hợp:

* Kiểm tra các mã giảm giá còn hiệu lực
* Thống kê mức độ sử dụng của các mã giảm giá
* Quản lý và theo dõi hiệu quả của các chương trình khuyến mãi
* Hỗ trợ việc ra quyết định về gia hạn hoặc kết thúc các mã giảm giá

**Các thao tác chính:**

1. Lấy thời điểm hiện tại:
   * Sử dụng hàm GETDATE() để lấy ngày hiện tại
   * Lưu vào biến @CurrentDate
2. Kiểm tra thời hạn mã giảm giá:
   * So sánh với ngày hiện tại (@CurrentDate)
   * Lọc các mã giảm giá còn hiệu lực
3. Thống kê số lượng khóa học:
   * Đếm số khóa học đã áp dụng mỗi mã giảm giá
4. Sắp xếp kết quả:
   * Theo số lượng khóa học giảm dần
   * Theo ngày hết hạn tăng dần

**Input:** Không có tham số đầu vào

**Output:** Kết quả trả về bảng gồm các cột:

* CouponID: Mã định danh
* CouponTitle: Tiêu đề mã giảm giá
* CouponValue: Giá trị giảm giá
* CouponType: Loại giảm giá
* CouponStartDate: Ngày bắt đầu
* CouponExpire: Ngày hết hạn
* CouponMaxDiscount: Giảm giá tối đa
* TotalAffectedCourses: Số khóa học đã áp dụng

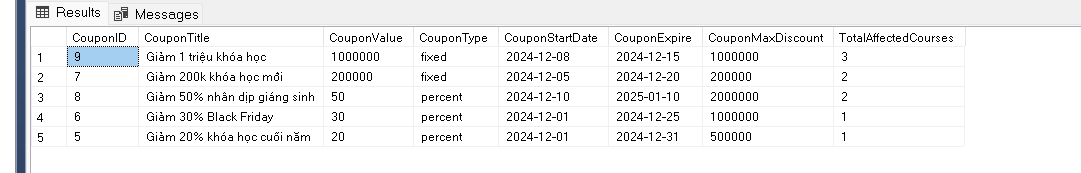
Câu lệnh tạo thủ tục:

|  |
| --- |
| GO  CREATE ALTER PROCEDURE sp\_GetValidCoupons  AS  BEGIN  DECLARE @CurrentDate DATE = GETDATE();    SELECT  c.CouponID,  c.CouponTitle,  c.CouponValue,  c.CouponType,  c.CouponStartDate,  c.CouponExpire,  c.CouponMaxDiscount,  (  SELECT COUNT(DISTINCT co.CourseID)  FROM Orders o, CourseOrder co  WHERE o.CouponID = c.CouponID  AND co.OrderID = o.OrderID  ) as TotalAffectedCourses  FROM Coupon c  WHERE c.CouponStartDate <= @CurrentDate  AND c.CouponExpire >= @CurrentDate  ORDER BY TotalAffectedCourses DESC, c.CouponExpire ASC;  END; |

Nhóm câu lệnh thực thi thủ tục mẫu:

|  |
| --- |
| EXEC sp\_GetValidCoupons; |

Kết quả các màn hình hiển thị từ DBMS minh họa cho việc thực hiện thủ tục:



**2.4 HÀM**

**a. Mô tả hàm: CalculateTotalRevenue**

**Chức năng của hàm**

Hàm CalculateTotalRevenue được sử dụng để tính tổng doanh thu từ các đơn hàng (orders) của một khóa học cụ thể, dựa trên thông tin trong bảng CourseOrder và Course.

**Use Case**

Hàm này được sử dụng trong các trường hợp:

* Khi cần tính tổng doanh thu của một khóa học cụ thể để phục vụ báo cáo doanh thu hoặc phân tích dữ liệu.
* Đảm bảo rằng doanh thu được tính toán chính xác dựa trên giá của khóa học và số lượng đơn hàng liên quan.

**Các thao tác chính**

1. **Kiểm tra tính hợp lệ**:
   * Kiểm tra xem CourseID có tồn tại trong bảng Course hay không.
   * Nếu không tồn tại, hàm trả về -1 để báo lỗi.
2. **Duyệt qua các đơn hàng**:
   * Sử dụng con trỏ (cursor) để duyệt qua từng đơn hàng liên quan đến khóa học trong bảng CourseOrder.
3. **Tính tổng doanh thu**:
   * Lấy giá của khóa học từ bảng Course và cộng dồn vào tổng doanh thu.
4. **Hoàn tất và trả về kết quả**:
   * Đóng con trỏ sau khi duyệt qua toàn bộ các đơn hàng.
   * Trả về tổng doanh thu tính được.

**Input**

Hàm nhận đầu vào là:

* @CourseID (INT): Mã định danh của khóa học cần tính tổng doanh thu.

**Output**

* **Kết quả trả về**:
  + Tổng doanh thu (INT) của các đơn hàng liên quan đến khóa học có CourseID.
  + Trả về -1 nếu CourseID không tồn tại trong bảng Course.

**Các bước thực hiện**

1. **Kiểm tra tính hợp lệ của CourseID**:
   * Nếu CourseID không tồn tại trong bảng Course, hàm sẽ trả về -1 để báo lỗi.
2. **Sử dụng con trỏ để duyệt qua đơn hàng**:
   * Lấy danh sách các OrderID liên quan đến khóa học từ bảng CourseOrder.
   * Duyệt qua từng đơn hàng bằng con trỏ.
3. **Tính tổng doanh thu**:
   * Lấy giá của khóa học từ bảng Course.
   * Cộng dồn giá trị vào biến @TotalRevenue.
4. **Hoàn tất và trả về**:
   * Đóng con trỏ và thu hồi bộ nhớ.
   * Trả về giá trị của @TotalRevenue.

**Ví dụ sử dụng**

1. **Kịch bản hợp lệ**:
   * @CourseID = 1: Khóa học tồn tại với nhiều đơn hàng.
   * Kết quả trả về là tổng doanh thu từ tất cả các đơn hàng liên quan.
2. **Kịch bản không hợp lệ**:
   * @CourseID = 999: Khóa học không tồn tại.
   * Kết quả trả về -1.

*-- Hàm tính tổng doanh thu từ các đơn hàng của một khóa học*

GO

CREATE FUNCTION CalculateTotalRevenue

(

    @CourseID INT

)

RETURNS INT

AS

BEGIN

    DECLARE @TotalRevenue INT = 0;

    DECLARE @OrderID INT;

    DECLARE @CoursePrice INT;

*-- Kiểm tra xem CourseID có tồn tại trong bảng*

    IF NOT EXISTS (SELECT 1 FROM Course WHERE CourseID = @CourseID)

    BEGIN

        RETURN -1; *-- Trả về -1 nếu CourseID không hợp lệ*

    END

*-- Khai báo con trỏ để lặp qua các đơn hàng liên quan đến khóa học*

    DECLARE OrderCursor CURSOR FOR

    SELECT OrderID FROM CourseOrder WHERE CourseID = @CourseID;

*-- Mở con trỏ để lấy OrderID đầu tiên*

    OPEN OrderCursor;

    FETCH NEXT FROM OrderCursor INTO @OrderID; *-- Lấy OrderID đầu tiên*

*-- Lặp qua các đơn hàng để tính tổng doanh thu*

    WHILE @@FETCH\_STATUS = 0 *-- Nếu còn dữ liệu*

    BEGIN

*-- Lấy giá khoá học từ bảng Course*

        SELECT @CoursePrice = CoursePrice FROM Course WHERE CourseID = @CourseID;

*-- Cộng vào tổng doanh thu*

        SET @TotalRevenue = @TotalRevenue + @CoursePrice;

*-- Lấy OrderID tiếp theo*

        FETCH NEXT FROM OrderCursor INTO @OrderID;

    END

*-- Đóng con trỏ và thu hồi bộ nhớ*

    CLOSE OrderCursor;

    DEALLOCATE OrderCursor;

*-- Trả về tổng doanh thu*

    RETURN @TotalRevenue;

END;

GO

Nhóm câu lệnh thực thi hàm:

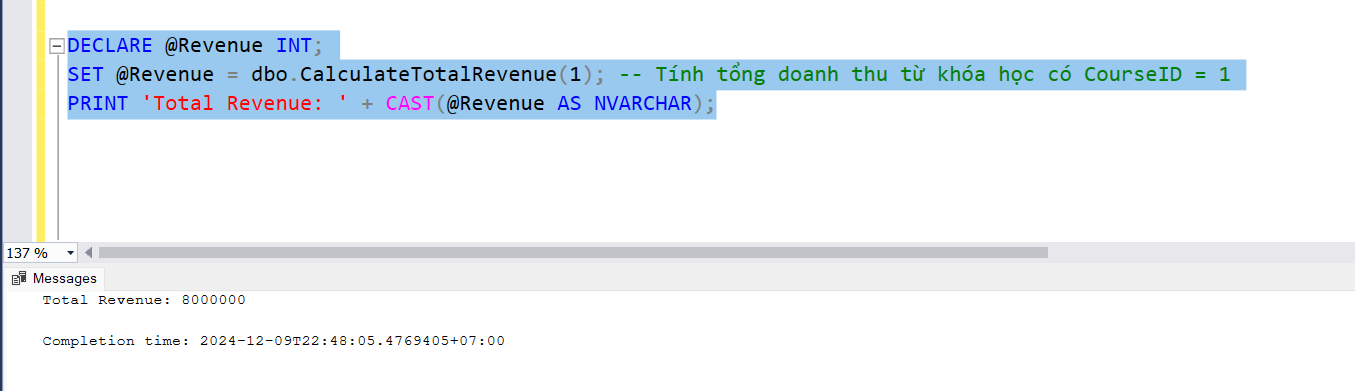
DECLARE @Revenue INT;

SET @Revenue = dbo.CalculateTotalRevenue(1); *-- Tính tổng doanh thu từ khóa học có CourseID = 1*

PRINT 'Total Revenue: ' + CAST(@Revenue AS NVARCHAR);

Kết quả các màn hình hiển thị từ DBMS minh họa cho việc thực hiện hàm

Khi course ID tồn tại:



Khi course ID không tồn tại: Trả về -1A computer screen shot of a message

Description automatically generated

b. Mô tả hàm: CalculateAverageTestScore

Chức năng của hàm

Hàm CalculateAverageTestScore được sử dụng để tính điểm trung bình của một học sinh dựa trên điểm số các bài kiểm tra mà học sinh đã thực hiện, lấy từ bảng Student\_Test.

Use Case

Hàm này được áp dụng trong các trường hợp:

* Khi cần đánh giá tổng quan về hiệu suất học tập của học sinh thông qua điểm trung bình.
* Sử dụng trong các báo cáo phân tích dữ liệu giáo dục hoặc bảng xếp hạng học sinh.

Các thao tác chính

1. Kiểm tra tính hợp lệ:
   * Xác nhận rằng StudentID có tồn tại trong bảng Student.
   * Nếu không tồn tại, hàm trả về -1 để báo lỗi.
2. Duyệt qua điểm kiểm tra:
   * Sử dụng con trỏ (cursor) để duyệt qua từng bài kiểm tra liên quan đến học sinh trong bảng Student\_Test.
3. Tính toán tổng điểm và số bài kiểm tra:
   * Cộng điểm của mỗi bài kiểm tra vào tổng điểm (@TotalScore).
   * Đếm số lượng bài kiểm tra đã duyệt (@TestCount).
4. Tính điểm trung bình:
   * Nếu số lượng bài kiểm tra lớn hơn 0, tính điểm trung bình bằng cách chia tổng điểm cho số lượng bài kiểm tra.
   * Nếu không có bài kiểm tra, trả về 0.

**Input**

Hàm nhận đầu vào là:

* @StudentID (INT): Mã định danh của học sinh cần tính điểm trung bình.

**Output**

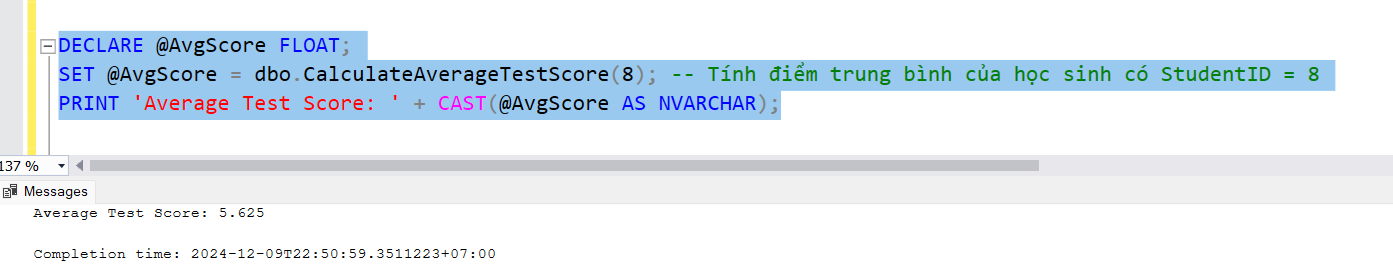
* **Kết quả trả về**:
  + Điểm trung bình (FLOAT) của các bài kiểm tra liên quan đến học sinh có StudentID.
  + Trả về -1 nếu StudentID không tồn tại trong bảng Student.
  + Trả về 0 nếu học sinh không có bài kiểm tra nào.

Các bước thực hiện

1. Kiểm tra tính hợp lệ của StudentID:
   * Nếu StudentID không tồn tại trong bảng Student, hàm sẽ trả về -1.
2. Sử dụng con trỏ để duyệt qua các bài kiểm tra:
   * Lấy danh sách các bài kiểm tra từ bảng Student\_Test liên quan đến học sinh.
   * Duyệt qua từng bài kiểm tra và cập nhật tổng điểm và số lượng bài kiểm tra.
3. Tính điểm trung bình:
   * Nếu có ít nhất một bài kiểm tra, tính điểm trung bình bằng cách chia tổng điểm cho số bài kiểm tra.
   * Nếu không có bài kiểm tra, trả về 0.
4. Hoàn tất và trả về:
   * Trả về giá trị điểm trung bình đã tính.

*Ví dụ sử dụng*

1. *Kịch bản hợp lệ:*
   * *@StudentID = 8: Học sinh có nhiều bài kiểm tra.*
   * *Kết quả trả về là điểm trung bình của tất cả các bài kiểm tra.*
2. *Kịch bản không hợp lệ:*
   * *@StudentID = 999: Học sinh không tồn tại.*
   * *Kết quả trả về -1.*
3. *Kịch bản không có bài kiểm tra:*
   * *@StudentID = 10: Học sinh tồn tại nhưng không có bài kiểm tra.*
   * *Kết quả trả về 0.*

Khi StudentID có tồn tại: Hàm trả về điểm trung bình của học sinh đó**

Khi StudentID không tồn tại: Hàm trả về -1**A screen shot of a computer

Description automatically generated**

**3. ỨNG DỤNG KẾT NỐI VỚI CSDL**

Giao diện ứng dụng: Gồm các nút bấm tương ứng với các bảng trong cơ sở dữ liệu. Khi bấm vào thì bảng dữ liệu đó sẽ hiện ra:

http://localhost:3500

A screenshot of a computer

Description automatically generated

http://localhost:3500/coupon

Giao diện khi bấm vào bảng “Coupon”: Bảng coupon hiện ra với các dữ liệu mẫu tương ứng.

Trang này gồm:

* Nút “Go Back” để trở về trang <http://localhost:3500>
* Nút “Create New Coupon” để tạo mới một mã giảm giá (Thủ tục insert trong phần 2.1)
* Thanh search để lọc ra các coupon có tiêu đề tương ứng.
* Nút sort để sắp xếp các dữ liệu trong bảng coupon theo các tiêu chí như CouponID, CouponTitle, CouponValue…

A screenshot of a coupon id

Description automatically generated

* Mỗi hàng tương ứng với coupon sẽ có thêm các nút Edit để chỉnh sửa coupon và Delete để xoá coupon tương ứng (Thủ tục sửa/xoá trong phần 2.1)
* Nút “View Coupons can be appied now” gọi thủ tục sp\_GetValidCoupons ở phần 2.3 để tìm ra các mã giảm giá có thể áp dụng ở thời điểm hiện tại.

**A screenshot of a computer

Description automatically generated**

http://localhost:3500/coupon/edit/1

Giao diện khi nhấn vào “Edit” một coupon nào đó: Khi nhập dữ liệu và nhấn “Update”, trang web sẽ dùng thủ tục ở phần 2.1 để validate và báo lỗi hoặc cập nhật (nếu dữ liệu hợp lệ).

**A screenshot of a computer

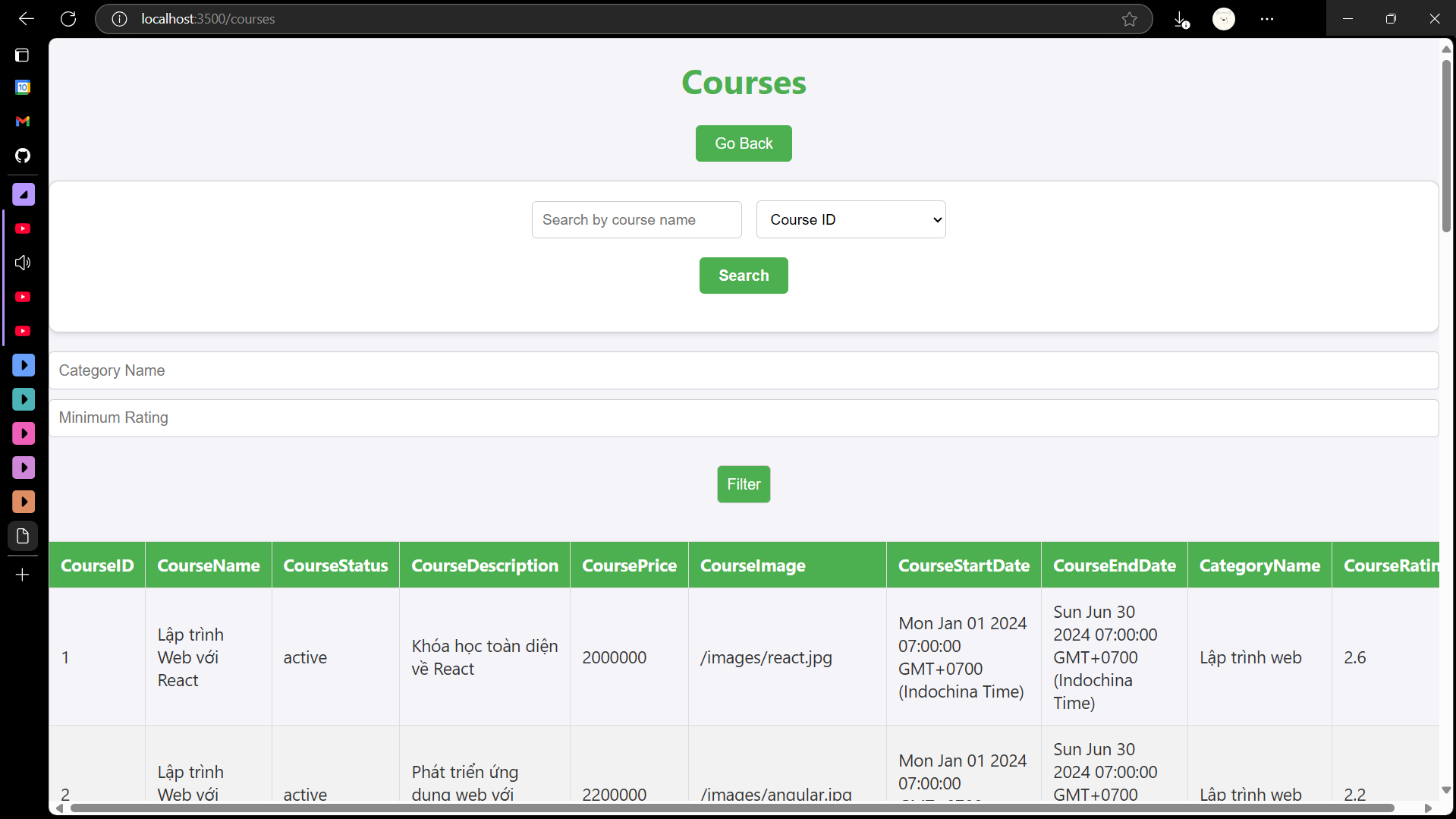
Description automatically generated**

Khi nhập dữ liệu không hợp lệ:A screenshot of a coupon

Description automatically generated

**http://localhost:3500/courses**

Giao diện khi nhấn vào bảng Course, bao gồm các nút tìm kiếm theo tên khoá học, sắp xếp, lọc (filter) theo tên danh mục và đánh giá trung bình (average rating) (các thủ tục ở phần 2.3)

****

**4. Thiết kế ERD cập nhật mới nhất**

A diagram of a football game

Description automatically generated

**A close-up of a computer code

Description automatically generated**

**Lưu ý chung:**

- Xem kỹ lại các yêu cầu trong file mô tả BTL2.

- Các câu lệnh đưa vào báo cáo phải ở dạng TEXT, không dùng ảnh chụp màn hình.

- Chỉ những nội dung được đưa vào cuốn báo cáo mới được tính điểm, những phần sửa chữa, bổ sung tại thời điểm báo cáo sẽ không tính.

- Tại buổi báo cáo, các nhóm cần thực hiện tạo sẵn các bảng dữ liệu, thêm sẵn dữ liệu, và tạo sẵn trigger, thủ tục, hàm. KHÔNG phải đến lúc báo cáo mới chạy các câu lệnh trên.

- Các nhóm cần chuẩn bị dữ liệu mẫu, câu lệnh mẫu để demo các nội dung trong buổi báo cáo.