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

**HK231 - Lớp: L0x**

**Nhóm**: <tên nhóm>

**Danh sách thành viên**:

| Stt | Họ tên | MSSV | Nội dung thực hiện | Ký tên xác nhận |
| --- | --- | --- | --- | --- |
| 1 |  |  | *Ví dụ: hàm 1, thủ tục 2, ứng dụng …* |  |
| 2 |  |  |  |  |
| *(Các vấn đề về mức đóng của các thành viên trong thực hiện BTL cần được trình bày tại buổi báo cáo. Cô không nhận email gửi riêng. Ví dụ: Bạn A không làm, Bạn B làm ít …)* | | | | | |

**Chủ đề**: <chủ đề mà nhóm chọn>

**1. Tạo bảng và dữ liệu mẫu**

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

USE [master]

GO

/\*\*\*\*\*\* Object: Database [EducationSystemDB] Script Date: 12/3/2024 5:11:00 PM \*\*\*\*\*\*/

CREATE DATABASE [EducationSystemDB]

GO

USE [EducationSystemDB]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Chapter] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Chapter](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[CourseId] [int] NOT NULL,

CONSTRAINT [PK\_Chapter] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Choice] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Choice](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Contents] [nvarchar](500) NOT NULL,

[Answer] [nvarchar](500) NOT NULL,

[QuestionId] [int] NOT NULL,

CONSTRAINT [PK\_Choice] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Classroom] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Classroom](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[SemesterCode] [nvarchar](50) NOT NULL,

[TeacherId] [int] NOT NULL,

[CourseId] [int] NOT NULL,

CONSTRAINT [PK\_Classroom] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Comment] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Comment](

[Id] [int] IDENTITY(1,1) NOT NULL,

[ForumPostId] [int] NOT NULL,

[UserId] [int] NOT NULL,

[CreatedTime] [datetime] NOT NULL,

[Contents] [nvarchar](1000) NOT NULL,

CONSTRAINT [PK\_Comment] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Course] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Course](

[Id] [int] IDENTITY(1,1) NOT NULL,

[BeforeCourseId] [int] NULL,

[Name] [nvarchar](50) NOT NULL,

[NumberOfCredits] [int] NOT NULL,

[EndDate] [date] NOT NULL,

[ForumId] [int] NULL,

CONSTRAINT [PK\_Course] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[DepartmentStats] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[DepartmentStats](

[Department] [nvarchar](50) NOT NULL,

[TotalUsers] [int] NULL,

[LastUpdated] [datetime] NULL,

PRIMARY KEY CLUSTERED

(

[Department] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Exam] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Exam](

[Id] [int] NOT NULL,

[QuestionId] [int] NOT NULL,

[Title] [nvarchar](50) NOT NULL,

[ExamDate] [date] NOT NULL,

[ExamDuration] [time](7) NOT NULL,

[Coefficient] [int] NOT NULL,

[Semester] [nvarchar](50) NOT NULL,

[ExamMethod] [nvarchar](50) NOT NULL,

[ChapterId] [int] NOT NULL,

[ClassroomId] [int] NOT NULL,

CONSTRAINT [PK\_Exam] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Exercise] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Exercise](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Description] [nvarchar](500) NOT NULL,

[MaxPoint] [int] NOT NULL,

[Coefficient] [int] NOT NULL,

[EndTime] [datetime] NOT NULL,

[ClassroomId] [int] NOT NULL,

CONSTRAINT [PK\_Exercise] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Forum] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Forum](

[Id] [int] IDENTITY(1,1) NOT NULL,

[CreatedDate] [date] NOT NULL,

[Title] [nvarchar](50) NOT NULL,

[Description] [nvarchar](500) NULL,

[CourseId] [int] NOT NULL,

CONSTRAINT [PK\_Forum] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[ForumPost] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[ForumPost](

[Id] [int] IDENTITY(1,1) NOT NULL,

[ForumId] [int] NOT NULL,

[CreatedTime] [datetime] NOT NULL,

[Contents] [nvarchar](2000) NOT NULL,

[UserId] [int] NOT NULL,

CONSTRAINT [PK\_ForumPost] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Question] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Question](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Contents] [nvarchar](500) NOT NULL,

[Answer] [nvarchar](500) NOT NULL,

[RelatedProgram] [nvarchar](1000) NOT NULL,

[Difficulty] [nvarchar](50) NOT NULL,

[Type] [nvarchar](50) NOT NULL,

[ChapterId] [int] NOT NULL,

CONSTRAINT [PK\_Question] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Schedule] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Schedule](

[Id] [int] IDENTITY(1,1) NOT NULL,

[RoomName] [nvarchar](50) NOT NULL,

[StartTime] [datetime] NOT NULL,

[EndTime] [datetime] NOT NULL,

[LessonNum] [int] NOT NULL,

[ClassroomId] [int] NOT NULL,

CONSTRAINT [PK\_Schedule] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Student] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Student](

[UserId] [int] NOT NULL,

[AdmissionDate] [date] NOT NULL,

[LearningStatus] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Student] PRIMARY KEY CLUSTERED

(

[UserId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[StudentClassroom] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[StudentClassroom](

[StudentId] [int] NOT NULL,

[ClassroomId] [int] NOT NULL,

[Point] [float] NOT NULL,

CONSTRAINT [PK\_StudentClassroom] PRIMARY KEY CLUSTERED

(

[StudentId] ASC,

[ClassroomId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[StudentExam] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[StudentExam](

[StudentId] [int] NOT NULL,

[ExamId] [int] NOT NULL,

[ChoiceId] [int] NOT NULL,

CONSTRAINT [PK\_StudentExam] PRIMARY KEY CLUSTERED

(

[StudentId] ASC,

[ExamId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[StudentExercise] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[StudentExercise](

[StudentId] [int] NOT NULL,

[ExerciseId] [int] NOT NULL,

[Point] [real] NULL,

[CreatedTime] [datetime] NOT NULL,

[Contents] [nvarchar](4000) NOT NULL,

CONSTRAINT [PK\_StudentExercise] PRIMARY KEY CLUSTERED

(

[StudentId] ASC,

[ExerciseId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[StudyMaterials] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[StudyMaterials](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Contents] [nvarchar](4000) NOT NULL,

[Type] [nvarchar](50) NOT NULL,

[ClassroomId] [int] NOT NULL,

CONSTRAINT [PK\_StudyMaterials] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Teacher] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Teacher](

[UserId] [int] NOT NULL,

[StartDate] [date] NOT NULL,

[Degree] [nvarchar](50) NOT NULL,

[WorkType] [nvarchar](50) NOT NULL,

[Achievement] [nvarchar](1000) NOT NULL,

CONSTRAINT [PK\_Teacher] PRIMARY KEY CLUSTERED

(

[UserId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Timetable] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Timetable](

[StudentId] [int] NOT NULL,

[ScheduleId] [int] NOT NULL,

CONSTRAINT [PK\_Timetable] PRIMARY KEY CLUSTERED

(

[StudentId] ASC,

[ScheduleId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[User] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[User](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Email] [varchar](50) NOT NULL,

[Password] [varchar](50) NOT NULL,

[FullName] [nvarchar](50) NOT NULL,

[IdentityNumber] [varchar](50) NULL,

[BirthDate] [date] NULL,

[Gender] [nvarchar](50) NOT NULL,

[Department] [nvarchar](50) NOT NULL,

[Status] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_User] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[UserAddress] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[UserAddress](

[UserId] [int] NOT NULL,

[HouseNum] [nvarchar](50) NOT NULL,

[Street] [nvarchar](50) NOT NULL,

[District] [nvarchar](50) NOT NULL,

[Province] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_UserAddress] PRIMARY KEY CLUSTERED

(

[UserId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[UserPhoneNumber] Script Date: 12/3/2024 5:11:01 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[UserPhoneNumber](

[UserId] [int] NOT NULL,

[PhoneNumber] [varchar](20) NOT NULL,

CONSTRAINT [PK\_UserPhoneNumber] PRIMARY KEY CLUSTERED

(

[UserId] ASC,

[PhoneNumber] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[DepartmentStats] ADD DEFAULT ((0)) FOR [TotalUsers]

GO

ALTER TABLE [dbo].[Chapter] WITH CHECK ADD CONSTRAINT [FK\_Chapter\_Course] FOREIGN KEY([CourseId])

REFERENCES [dbo].[Course] ([Id])

GO

ALTER TABLE [dbo].[Chapter] CHECK CONSTRAINT [FK\_Chapter\_Course]

GO

ALTER TABLE [dbo].[Choice] WITH CHECK ADD CONSTRAINT [FK\_Choice\_Question] FOREIGN KEY([QuestionId])

REFERENCES [dbo].[Question] ([Id])

GO

ALTER TABLE [dbo].[Choice] CHECK CONSTRAINT [FK\_Choice\_Question]

GO

ALTER TABLE [dbo].[Classroom] WITH CHECK ADD CONSTRAINT [FK\_Classroom\_Course] FOREIGN KEY([CourseId])

REFERENCES [dbo].[Course] ([Id])

GO

ALTER TABLE [dbo].[Classroom] CHECK CONSTRAINT [FK\_Classroom\_Course]

GO

ALTER TABLE [dbo].[Classroom] WITH CHECK ADD CONSTRAINT [FK\_Classroom\_Teacher] FOREIGN KEY([TeacherId])

REFERENCES [dbo].[Teacher] ([UserId])

GO

ALTER TABLE [dbo].[Classroom] CHECK CONSTRAINT [FK\_Classroom\_Teacher]

GO

ALTER TABLE [dbo].[Comment] WITH CHECK ADD CONSTRAINT [FK\_Comment\_ForumPost] FOREIGN KEY([ForumPostId])

REFERENCES [dbo].[ForumPost] ([Id])

GO

ALTER TABLE [dbo].[Comment] CHECK CONSTRAINT [FK\_Comment\_ForumPost]

GO

ALTER TABLE [dbo].[Comment] WITH CHECK ADD CONSTRAINT [FK\_Comment\_User] FOREIGN KEY([UserId])

REFERENCES [dbo].[User] ([Id])

GO

ALTER TABLE [dbo].[Comment] CHECK CONSTRAINT [FK\_Comment\_User]

GO

ALTER TABLE [dbo].[Course] WITH CHECK ADD CONSTRAINT [FK\_Course\_Course] FOREIGN KEY([BeforeCourseId])

REFERENCES [dbo].[Course] ([Id])

GO

ALTER TABLE [dbo].[Course] CHECK CONSTRAINT [FK\_Course\_Course]

GO

ALTER TABLE [dbo].[Course] WITH CHECK ADD CONSTRAINT [FK\_Course\_Forum] FOREIGN KEY([ForumId])

REFERENCES [dbo].[Forum] ([Id])

GO

ALTER TABLE [dbo].[Course] CHECK CONSTRAINT [FK\_Course\_Forum]

GO

ALTER TABLE [dbo].[Exam] WITH CHECK ADD CONSTRAINT [FK\_Exam\_Chapter] FOREIGN KEY([ChapterId])

REFERENCES [dbo].[Chapter] ([Id])

GO

ALTER TABLE [dbo].[Exam] CHECK CONSTRAINT [FK\_Exam\_Chapter]

GO

ALTER TABLE [dbo].[Exam] WITH CHECK ADD CONSTRAINT [FK\_Exam\_Classroom] FOREIGN KEY([ClassroomId])

REFERENCES [dbo].[Classroom] ([Id])

GO

ALTER TABLE [dbo].[Exam] CHECK CONSTRAINT [FK\_Exam\_Classroom]

GO

ALTER TABLE [dbo].[Exam] WITH CHECK ADD CONSTRAINT [FK\_Exam\_QuestionId] FOREIGN KEY([QuestionId])

REFERENCES [dbo].[Question] ([Id])

GO

ALTER TABLE [dbo].[Exam] CHECK CONSTRAINT [FK\_Exam\_QuestionId]

GO

ALTER TABLE [dbo].[Exercise] WITH CHECK ADD CONSTRAINT [FK\_Exercise\_Classroom] FOREIGN KEY([ClassroomId])

REFERENCES [dbo].[Classroom] ([Id])

GO

ALTER TABLE [dbo].[Exercise] CHECK CONSTRAINT [FK\_Exercise\_Classroom]

GO

ALTER TABLE [dbo].[Forum] WITH CHECK ADD CONSTRAINT [FK\_Forum\_Course] FOREIGN KEY([CourseId])

REFERENCES [dbo].[Course] ([Id])

GO

ALTER TABLE [dbo].[Forum] CHECK CONSTRAINT [FK\_Forum\_Course]

GO

ALTER TABLE [dbo].[ForumPost] WITH CHECK ADD CONSTRAINT [FK\_ForumPost\_Forum] FOREIGN KEY([ForumId])

REFERENCES [dbo].[Forum] ([Id])

GO

ALTER TABLE [dbo].[ForumPost] CHECK CONSTRAINT [FK\_ForumPost\_Forum]

GO

ALTER TABLE [dbo].[ForumPost] WITH CHECK ADD CONSTRAINT [FK\_ForumPost\_User] FOREIGN KEY([UserId])

REFERENCES [dbo].[User] ([Id])

GO

ALTER TABLE [dbo].[ForumPost] CHECK CONSTRAINT [FK\_ForumPost\_User]

GO

ALTER TABLE [dbo].[Question] WITH CHECK ADD CONSTRAINT [FK\_Question\_Chapter] FOREIGN KEY([ChapterId])

REFERENCES [dbo].[Chapter] ([Id])

GO

ALTER TABLE [dbo].[Question] CHECK CONSTRAINT [FK\_Question\_Chapter]

GO

ALTER TABLE [dbo].[Schedule] WITH CHECK ADD CONSTRAINT [FK\_Schedule\_Classroom] FOREIGN KEY([ClassroomId])

REFERENCES [dbo].[Classroom] ([Id])

GO

ALTER TABLE [dbo].[Schedule] CHECK CONSTRAINT [FK\_Schedule\_Classroom]

GO

ALTER TABLE [dbo].[Student] WITH CHECK ADD CONSTRAINT [FK\_Student\_User] FOREIGN KEY([UserId])

REFERENCES [dbo].[User] ([Id])

GO

ALTER TABLE [dbo].[Student] CHECK CONSTRAINT [FK\_Student\_User]

GO

ALTER TABLE [dbo].[StudentClassroom] WITH CHECK ADD CONSTRAINT [FK\_StudentClassroom\_Classroom] FOREIGN KEY([ClassroomId])

REFERENCES [dbo].[Classroom] ([Id])

GO

ALTER TABLE [dbo].[StudentClassroom] CHECK CONSTRAINT [FK\_StudentClassroom\_Classroom]

GO

ALTER TABLE [dbo].[StudentClassroom] WITH CHECK ADD CONSTRAINT [FK\_StudentClassroom\_Student] FOREIGN KEY([StudentId])

REFERENCES [dbo].[Student] ([UserId])

GO

ALTER TABLE [dbo].[StudentClassroom] CHECK CONSTRAINT [FK\_StudentClassroom\_Student]

GO

ALTER TABLE [dbo].[StudentExam] WITH CHECK ADD CONSTRAINT [FK\_StudentExam\_Choice] FOREIGN KEY([ChoiceId])

REFERENCES [dbo].[Choice] ([Id])

GO

ALTER TABLE [dbo].[StudentExam] CHECK CONSTRAINT [FK\_StudentExam\_Choice]

GO

ALTER TABLE [dbo].[StudentExam] WITH CHECK ADD CONSTRAINT [FK\_StudentExam\_Exam] FOREIGN KEY([ExamId])

REFERENCES [dbo].[Exam] ([Id])

GO

ALTER TABLE [dbo].[StudentExam] CHECK CONSTRAINT [FK\_StudentExam\_Exam]

GO

ALTER TABLE [dbo].[StudentExam] WITH CHECK ADD CONSTRAINT [FK\_StudentExam\_Student] FOREIGN KEY([StudentId])

REFERENCES [dbo].[Student] ([UserId])

GO

ALTER TABLE [dbo].[StudentExam] CHECK CONSTRAINT [FK\_StudentExam\_Student]

GO

ALTER TABLE [dbo].[StudentExercise] WITH CHECK ADD CONSTRAINT [FK\_StudentExercise\_Exercise] FOREIGN KEY([ExerciseId])

REFERENCES [dbo].[Exercise] ([Id])

GO

ALTER TABLE [dbo].[StudentExercise] CHECK CONSTRAINT [FK\_StudentExercise\_Exercise]

GO

ALTER TABLE [dbo].[StudentExercise] WITH CHECK ADD CONSTRAINT [FK\_StudentExercise\_Student] FOREIGN KEY([StudentId])

REFERENCES [dbo].[Student] ([UserId])

GO

ALTER TABLE [dbo].[StudentExercise] CHECK CONSTRAINT [FK\_StudentExercise\_Student]

GO

ALTER TABLE [dbo].[StudyMaterials] WITH CHECK ADD CONSTRAINT [FK\_StudyMaterials\_Classroom] FOREIGN KEY([ClassroomId])

REFERENCES [dbo].[Classroom] ([Id])

GO

ALTER TABLE [dbo].[StudyMaterials] CHECK CONSTRAINT [FK\_StudyMaterials\_Classroom]

GO

ALTER TABLE [dbo].[Teacher] WITH CHECK ADD CONSTRAINT [FK\_Teacher\_User] FOREIGN KEY([UserId])

REFERENCES [dbo].[User] ([Id])

GO

ALTER TABLE [dbo].[Teacher] CHECK CONSTRAINT [FK\_Teacher\_User]

GO

ALTER TABLE [dbo].[Timetable] WITH CHECK ADD CONSTRAINT [FK\_Timetable\_Schedule] FOREIGN KEY([ScheduleId])

REFERENCES [dbo].[Schedule] ([Id])

GO

ALTER TABLE [dbo].[Timetable] CHECK CONSTRAINT [FK\_Timetable\_Schedule]

GO

ALTER TABLE [dbo].[Timetable] WITH CHECK ADD CONSTRAINT [FK\_Timetable\_Student] FOREIGN KEY([StudentId])

REFERENCES [dbo].[Student] ([UserId])

GO

ALTER TABLE [dbo].[Timetable] CHECK CONSTRAINT [FK\_Timetable\_Student]

GO

ALTER TABLE [dbo].[UserAddress] WITH CHECK ADD CONSTRAINT [FK\_UserAddress\_User] FOREIGN KEY([UserId])

REFERENCES [dbo].[User] ([Id])

GO

ALTER TABLE [dbo].[UserAddress] CHECK CONSTRAINT [FK\_UserAddress\_User]

GO

ALTER TABLE [dbo].[UserPhoneNumber] WITH CHECK ADD CONSTRAINT [FK\_UserPhoneNumber\_User] FOREIGN KEY([UserId])

REFERENCES [dbo].[User] ([Id])

GO

ALTER TABLE [dbo].[UserPhoneNumber] CHECK CONSTRAINT [FK\_UserPhoneNumber\_User]

GO

**1.2 Câu lệnh thêm dữ liệu**

USE [EducationSystemDB]

GO

SET IDENTITY\_INSERT [dbo].[User] ON

INSERT [dbo].[User] ([Id], [Email], [Password], [FullName], [IdentityNumber], [BirthDate], [Gender], [Department], [Status]) VALUES (1, N'teacher@gmail.com', N'123', N'Teacher', N'039382759182', CAST(N'2000-04-04' AS Date), N'Nam', N'Công nghệ thông tin', N'Active')

INSERT [dbo].[User] ([Id], [Email], [Password], [FullName], [IdentityNumber], [BirthDate], [Gender], [Department], [Status]) VALUES (2, N'student@gmail.com', N'123', N'Student 1', N'039284712374', CAST(N'2003-03-15' AS Date), N'Nam', N'Công nghệ thông tin', N'Active')

INSERT [dbo].[User] ([Id], [Email], [Password], [FullName], [IdentityNumber], [BirthDate], [Gender], [Department], [Status]) VALUES (3, N'student1@gmail.com', N'123', N'Student 2', N'039231458237', CAST(N'2005-09-12' AS Date), N'Nam', N'Công nghệ thông tin', N'Active')

INSERT [dbo].[User] ([Id], [Email], [Password], [FullName], [IdentityNumber], [BirthDate], [Gender], [Department], [Status]) VALUES (4, N'student2@gmail.com', N'123', N'Student 3', N'093827421323', CAST(N'2002-12-01' AS Date), N'Nam', N'Kinh tế', N'Active')

INSERT [dbo].[User] ([Id], [Email], [Password], [FullName], [IdentityNumber], [BirthDate], [Gender], [Department], [Status]) VALUES (5, N'student3@gmail.com', N'123', N'Student 4', N'023984821392', CAST(N'2001-07-14' AS Date), N'Nam', N'Kinh tế', N'Active')

SET IDENTITY\_INSERT [dbo].[User] OFF

GO

INSERT [dbo].[Student] ([UserId], [AdmissionDate], [LearningStatus]) VALUES (2, CAST(N'2021-05-12' AS Date), N'Active')

INSERT [dbo].[Student] ([UserId], [AdmissionDate], [LearningStatus]) VALUES (3, CAST(N'2021-08-12' AS Date), N'Active')

INSERT [dbo].[Student] ([UserId], [AdmissionDate], [LearningStatus]) VALUES (4, CAST(N'2021-08-12' AS Date), N'Active')

INSERT [dbo].[Student] ([UserId], [AdmissionDate], [LearningStatus]) VALUES (5, CAST(N'2021-05-12' AS Date), N'Active')

GO

INSERT [dbo].[Teacher] ([UserId], [StartDate], [Degree], [WorkType], [Achievement]) VALUES (1, CAST(N'2020-09-12' AS Date), N'Thạc Sĩ', N'Dạy học', N'123')

GO

SET IDENTITY\_INSERT [dbo].[Course] ON

INSERT [dbo].[Course] ([Id], [BeforeCourseId], [Name], [NumberOfCredits], [EndDate], [ForumId]) VALUES (1, NULL, N'PTO152', 15, CAST(N'2024-05-15' AS Date), NULL)

INSERT [dbo].[Course] ([Id], [BeforeCourseId], [Name], [NumberOfCredits], [EndDate], [ForumId]) VALUES (2, NULL, N'AOC147', 15, CAST(N'2024-05-15' AS Date), NULL)

INSERT [dbo].[Course] ([Id], [BeforeCourseId], [Name], [NumberOfCredits], [EndDate], [ForumId]) VALUES (3, NULL, N'PCF912', 15, CAST(N'2024-05-15' AS Date), NULL)

SET IDENTITY\_INSERT [dbo].[Course] OFF

GO

SET IDENTITY\_INSERT [dbo].[Classroom] ON

INSERT [dbo].[Classroom] ([Id], [Name], [SemesterCode], [TeacherId], [CourseId]) VALUES (1, N'Winter\_DCD153', N'Spring\_HK1', 1, 1)

INSERT [dbo].[Classroom] ([Id], [Name], [SemesterCode], [TeacherId], [CourseId]) VALUES (2, N'Winter\_OBC123', N'Spring\_HK1', 1, 2)

INSERT [dbo].[Classroom] ([Id], [Name], [SemesterCode], [TeacherId], [CourseId]) VALUES (3, N'WInter\_BDA\_582', N'Spring\_HK1', 1, 3)

INSERT [dbo].[Classroom] ([Id], [Name], [SemesterCode], [TeacherId], [CourseId]) VALUES (4, N'WInter\_BDA\_531', N'Spring\_HK1', 1, 3)

SET IDENTITY\_INSERT [dbo].[Classroom] OFF

GO

SET IDENTITY\_INSERT [dbo].[Schedule] ON

INSERT [dbo].[Schedule] ([Id], [RoomName], [StartTime], [EndTime], [LessonNum], [ClassroomId]) VALUES (1, N'Phòng 202', CAST(N'2024-03-05T15:00:00.000' AS DateTime), CAST(N'2024-03-05T17:00:00.000' AS DateTime), 1, 1)

INSERT [dbo].[Schedule] ([Id], [RoomName], [StartTime], [EndTime], [LessonNum], [ClassroomId]) VALUES (3, N'Phòng 404', CAST(N'2024-03-05T17:30:00.000' AS DateTime), CAST(N'2024-03-05T19:30:00.000' AS DateTime), 2, 1)

INSERT [dbo].[Schedule] ([Id], [RoomName], [StartTime], [EndTime], [LessonNum], [ClassroomId]) VALUES (4, N'Phòng 321', CAST(N'2024-03-05T15:00:00.000' AS DateTime), CAST(N'2024-03-05T17:00:00.000' AS DateTime), 1, 3)

INSERT [dbo].[Schedule] ([Id], [RoomName], [StartTime], [EndTime], [LessonNum], [ClassroomId]) VALUES (5, N'Phòng 105', CAST(N'2024-03-05T17:30:00.000' AS DateTime), CAST(N'2024-03-05T19:30:00.000' AS DateTime), 2, 3)

INSERT [dbo].[Schedule] ([Id], [RoomName], [StartTime], [EndTime], [LessonNum], [ClassroomId]) VALUES (6, N'Phòng 024', CAST(N'2024-03-05T15:00:00.000' AS DateTime), CAST(N'2024-03-05T17:00:00.000' AS DateTime), 1, 2)

INSERT [dbo].[Schedule] ([Id], [RoomName], [StartTime], [EndTime], [LessonNum], [ClassroomId]) VALUES (7, N'Phòng 147', CAST(N'2024-03-05T17:30:00.000' AS DateTime), CAST(N'2024-03-05T19:30:00.000' AS DateTime), 2, 2)

SET IDENTITY\_INSERT [dbo].[Schedule] OFF

GO

INSERT [dbo].[Timetable] ([StudentId], [ScheduleId]) VALUES (2, 1)

INSERT [dbo].[Timetable] ([StudentId], [ScheduleId]) VALUES (2, 3)

INSERT [dbo].[Timetable] ([StudentId], [ScheduleId]) VALUES (2, 4)

INSERT [dbo].[Timetable] ([StudentId], [ScheduleId]) VALUES (2, 5)

INSERT [dbo].[Timetable] ([StudentId], [ScheduleId]) VALUES (2, 6)

INSERT [dbo].[Timetable] ([StudentId], [ScheduleId]) VALUES (3, 1)

INSERT [dbo].[Timetable] ([StudentId], [ScheduleId]) VALUES (3, 3)

INSERT [dbo].[Timetable] ([StudentId], [ScheduleId]) VALUES (3, 6)

INSERT [dbo].[Timetable] ([StudentId], [ScheduleId]) VALUES (3, 7)

INSERT [dbo].[Timetable] ([StudentId], [ScheduleId]) VALUES (4, 4)

INSERT [dbo].[Timetable] ([StudentId], [ScheduleId]) VALUES (4, 5)

INSERT [dbo].[Timetable] ([StudentId], [ScheduleId]) VALUES (5, 4)

INSERT [dbo].[Timetable] ([StudentId], [ScheduleId]) VALUES (5, 5)

GO

-- Thêm dữ liệu mẫu cho bảng Chapter

INSERT INTO [dbo].[Chapter] ([Name], [CourseId]) VALUES

(N'Chương 1: Cơ bản', 1),

(N'Chương 2: Nâng cao', 1),

(N'Chương 1: Nhập môn', 2),

(N'Chương 2: Chuyên sâu', 2),

(N'Chương 1: Cơ sở', 3),

(N'Chương 2: Ứng dụng', 3);

-- Thêm câu hỏi cho các chương

INSERT INTO [dbo].[Question] ([Contents], [Answer], [RelatedProgram], [Difficulty], [Type], [ChapterId]) VALUES

(N'Câu hỏi 1 Chương 1', N'Đáp án 1', N'Program 1', N'Dễ', N'Multiple Choice', 1),

(N'Câu hỏi 2 Chương 1', N'Đáp án 2', N'Program 1', N'Trung bình', N'Multiple Choice', 1),

(N'Câu hỏi 1 Chương 2', N'Đáp án 3', N'Program 2', N'Khó', N'Multiple Choice', 2),

(N'Câu hỏi 1 Chương 3', N'Đáp án 4', N'Program 2', N'Trung bình', N'Multiple Choice', 3);

-- Thêm lựa chọn cho câu hỏi

INSERT INTO [dbo].[Choice] ([Contents], [Answer], [QuestionId]) VALUES

(N'Lựa chọn A', N'A', 1),

(N'Lựa chọn B', N'B', 1),

(N'Lựa chọn C', N'C', 1),

(N'Lựa chọn A', N'A', 2),

(N'Lựa chọn B', N'B', 2);

-- Thêm bài kiểm tra

INSERT INTO [dbo].[Exam] ([Id], [QuestionId], [Title], [ExamDate], [ExamDuration], [Coefficient], [Semester], [ExamMethod], [ChapterId], [ClassroomId]) VALUES

(1, 1, N'Kiểm tra giữa kỳ 1', '2024-04-15', '01:30:00', 2, N'Spring\_HK1', N'Online', 1, 1),

(2, 2, N'Kiểm tra cuối kỳ 1', '2024-05-30', '02:00:00', 3, N'Spring\_HK1', N'Offline', 2, 1),

(3, 3, N'Kiểm tra giữa kỳ 2', '2024-04-16', '01:30:00', 2, N'Spring\_HK1', N'Online', 3, 2);

-- Thêm bài tập

INSERT INTO [dbo].[Exercise] ([Name], [Description], [MaxPoint], [Coefficient], [EndTime], [ClassroomId]) VALUES

(N'Bài tập 1', N'Mô tả bài tập 1', 10, 1, '2024-03-20 23:59:59', 1),

(N'Bài tập 2', N'Mô tả bài tập 2', 10, 2, '2024-04-10 23:59:59', 1),

(N'Bài tập 3', N'Mô tả bài tập 3', 10, 1, '2024-03-25 23:59:59', 2);

-- Thêm dữ liệu sinh viên làm bài tập

INSERT INTO [dbo].[StudentExercise] ([StudentId], [ExerciseId], [Point], [CreatedTime], [Contents]) VALUES

(2, 1, 8.5, '2024-03-19 10:00:00', N'Bài làm của sinh viên 1'),

(2, 2, 9.0, '2024-04-09 15:30:00', N'Bài làm của sinh viên 1'),

(3, 1, 7.5, '2024-03-18 14:20:00', N'Bài làm của sinh viên 2'),

(3, 2, 8.0, '2024-04-08 16:45:00', N'Bài làm của sinh viên 2');

-- Thêm dữ liệu sinh viên làm bài kiểm tra

INSERT INTO [dbo].[StudentExam] ([StudentId], [ExamId], [ChoiceId]) VALUES

(2, 1, 1),

(2, 2, 4),

(3, 1, 2),

(3, 2, 5);

-- Thêm điểm tổng kết môn cho sinh viên

INSERT INTO [dbo].[StudentClassroom] ([StudentId], [ClassroomId], [Point]) VALUES

(2, 1, 8.5),

(2, 2, 7.5),

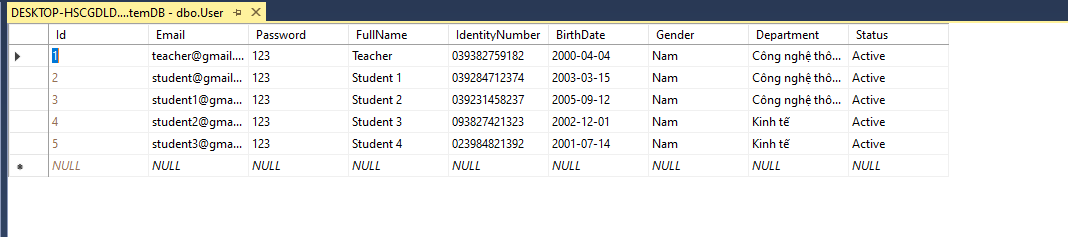
(3, 1, 8.0),

(3, 2, 7.0),

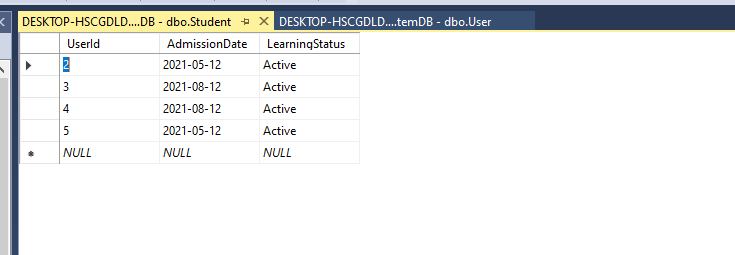
(4, 3, 8.2),

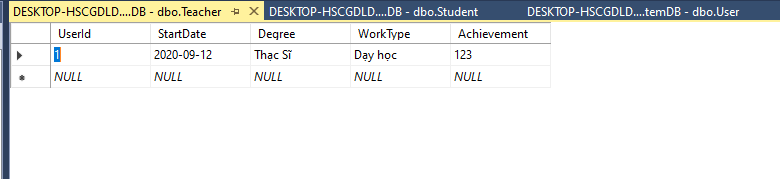
(5, 3, 7.8);

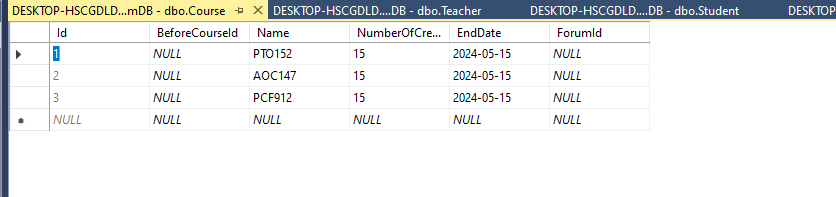
Kết quả dữ liệu của các bảng trong CSDL.

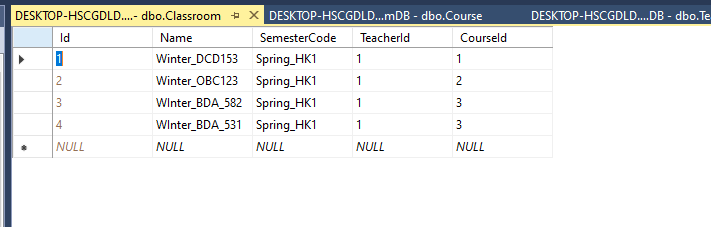
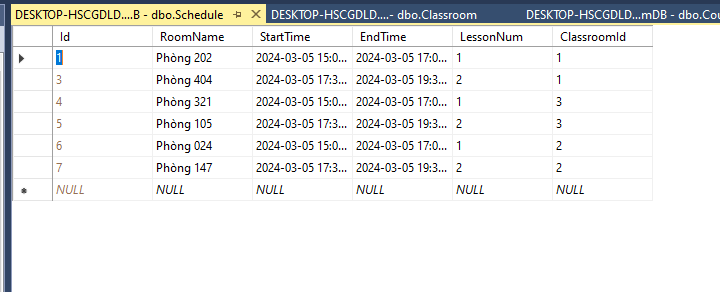
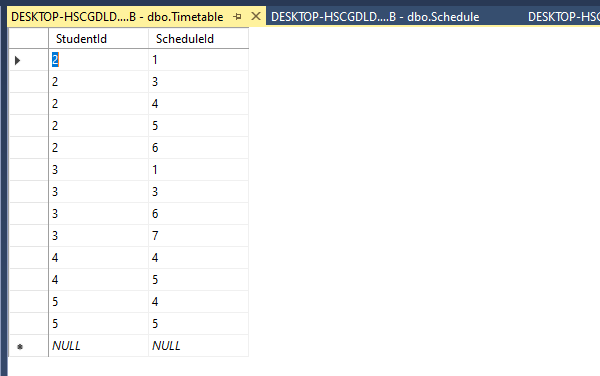
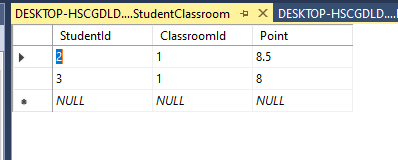
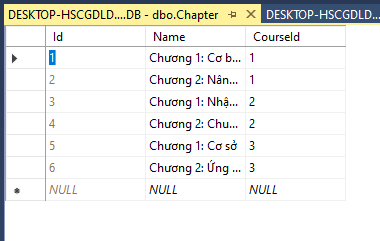
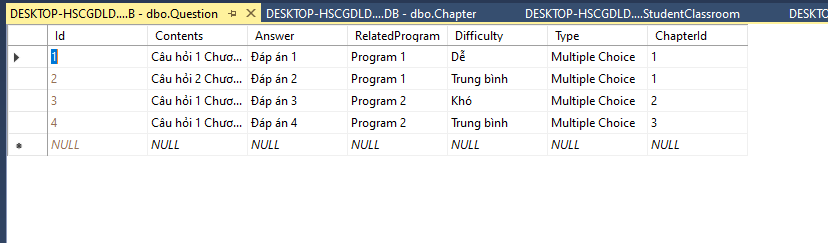
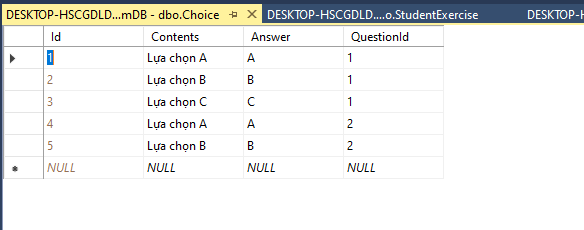
USER:  


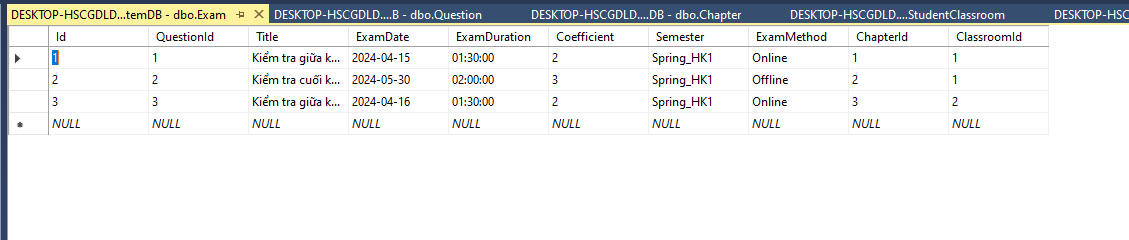
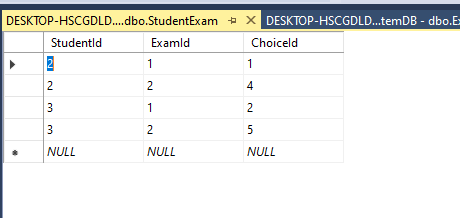
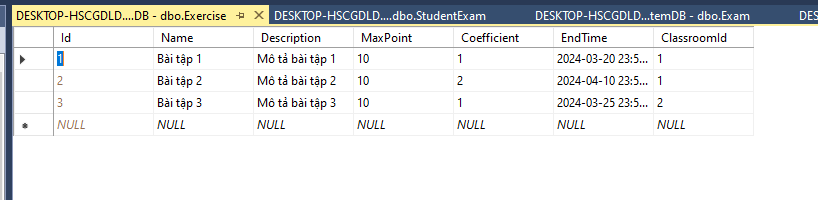
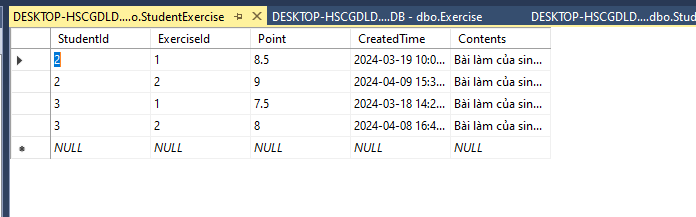
Student:



Teacher:

Course:  


Classroom:  
  
Schedule:  
  
Timetable:  
  
StudentClassroom:  
  
Chapter:  
  
Question:  
Choice:  


Exam:  
  
StudentExam:  
  
Exercise:  
  
StudentExercise:  


**2. Trigger – Thủ tục - Hàm**

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

Bảng dữ liệu: Student, User

1. Thủ tục INSERT:

Mô tả thủ tục: Thêm thông tin học sinh và tài khoản người dùng, kiểm tra check validate giá trị nhập

– Kiểm tra validation các giá trị nhập

-- Kiểm tra Email có tồn tại trong bảng User chưa

-- Thêm mới vào bảng User

-- Lấy UserId vừa được thêm

-- Thêm mới vào bảng Student

-- Thông báo kết quả

Input:

@Email NVARCHAR(50),

@Password NVARCHAR(50),

@FullName NVARCHAR(50),

@IdentityNumber NVARCHAR(50) = NULL,

@BirthDate DATE = NULL,

@Gender NVARCHAR(50),

@Department NVARCHAR(50),

@Status NVARCHAR(50),

@AdmissionDate DATE,

@LearningStatus NVARCHAR(50),

@ErrorMessage NVARCHAR(MAX) OUTPUT

Output:   
Thông báo lỗi validate:  
Lỗi: Email không hợp lệ. Độ dài phải từ 1 đến 50 ký tự

Lỗi: Email không hợp lệ. Phải có định dạng hợp lệ (vd: example@domain.com).  
Lỗi: Password không hợp lệ. Độ dài phải từ 1 đến 50 ký tự.

Lỗi: Identity Number không hợp lệ. Độ dài không được quá 50 ký tự.

Câu lệnh tạo thủ tục:

USE [EducationSystemDB]

GO

CREATE PROCEDURE InsertStudentAndUser

@Email NVARCHAR(50),

@Password NVARCHAR(50),

@FullName NVARCHAR(50),

@IdentityNumber NVARCHAR(50) = NULL,

@BirthDate DATE = NULL,

@Gender NVARCHAR(50),

@Department NVARCHAR(50),

@Status NVARCHAR(50),

@AdmissionDate DATE,

@LearningStatus NVARCHAR(50),

@ErrorMessage NVARCHAR(MAX) OUTPUT

AS

BEGIN

BEGIN TRY

DECLARE @UserId INT;

-- Kiểm tra độ dài Email

IF @Email IS NULL OR LEN(@Email) = 0 OR LEN(@Email) > 50

BEGIN

SET @ErrorMessage = N'Lỗi: Email không hợp lệ. Độ dài phải từ 1 đến 50 ký tự.';

RETURN;

END

-- Kiểm tra tính hợp lệ của Email

IF NOT (@Email LIKE '%@%.%')

BEGIN

SET @ErrorMessage = N'Lỗi: Email không hợp lệ. Phải có định dạng hợp lệ (vd: example@domain.com).';

RETURN;

END

-- Kiểm tra độ dài Password

IF @Password IS NULL OR LEN(@Password) = 0 OR LEN(@Password) > 50

BEGIN

SET @ErrorMessage = N'Lỗi: Password không hợp lệ. Độ dài phải từ 1 đến 50 ký tự.';

RETURN;

END

-- Kiểm tra độ dài Fullname

IF @FullName IS NULL OR LEN(@FullName) = 0 OR LEN(@FullName) > 50

BEGIN

SET @ErrorMessage = N'Lỗi: Fullname không hợp lệ. Độ dài phải từ 1 đến 50 ký tự.';

RETURN;

END

-- Kiểm tra độ dài Identity Number

IF @IdentityNumber IS NOT NULL AND LEN(@IdentityNumber) > 50

BEGIN

SET @ErrorMessage = N'Lỗi: Identity Number không hợp lệ. Độ dài không được quá 50 ký tự.';

RETURN;

END

-- Kiểm tra độ dài Gender

IF @Gender IS NULL OR LEN(@Gender) = 0 OR LEN(@Gender) > 50

BEGIN

SET @ErrorMessage = N'Lỗi: Gender không hợp lệ. Độ dài phải từ 1 đến 50 ký tự.';

RETURN;

END

-- Kiểm tra độ dài Department

IF @Department IS NULL OR LEN(@Department) = 0 OR LEN(@Department) > 50

BEGIN

SET @ErrorMessage = N'Lỗi: Department không hợp lệ. Độ dài phải từ 1 đến 50 ký tự.';

RETURN;

END

-- Check không empty Admission Date

IF @AdmissionDate IS NULL

BEGIN

SET @ErrorMessage = N'Lỗi: Admission Date không được để trống.';

RETURN;

END

-- Kiểm tra độ dài LearningStatus

IF @LearningStatus IS NULL OR LEN(@LearningStatus) = 0 OR LEN(@LearningStatus) > 50

BEGIN

SET @ErrorMessage = N'Lỗi: Learning Status không hợp lệ. Độ dài phải từ 1 đến 50 ký tự.';

RETURN;

END

-- Kiểm tra độ dài Status

IF LEN(@Status) = 0 OR LEN(@Status) > 50

BEGIN

SET @ErrorMessage = N'Lỗi: Status không hợp lệ. Độ dài phải từ 1 đến 50 ký tự.';

RETURN;

END

-- Kiểm tra Email có tồn tại trong bảng User chưa

IF EXISTS (SELECT 1 FROM [User] WHERE Email = @Email)

BEGIN

SET @ErrorMessage = N'Lỗi: Email đã tồn tại trong bảng User.';

RETURN;

END

-- Thêm mới vào bảng User

INSERT INTO [User] (Email, Password, FullName, IdentityNumber, BirthDate, Gender, Department, Status)

VALUES (@Email, @Password, @FullName, @IdentityNumber, @BirthDate, @Gender, @Department, @Status);

-- Lấy UserId vừa được thêm

SET @UserId = SCOPE\_IDENTITY();

-- Thêm mới vào bảng Student

INSERT INTO [Student] (UserId, AdmissionDate, LearningStatus)

VALUES (@UserId, @AdmissionDate, @LearningStatus);

-- Trả về thông báo thành công

SET @ErrorMessage = N'Thêm User và Student thành công!';

END TRY

BEGIN CATCH

-- Xử lý lỗi hệ thống

SET @ErrorMessage = ERROR\_MESSAGE();

END CATCH

END;

Nhóm câu lệnh thực thi thủ tục mẫu:

USE [EducationSystemDB]

GO

DECLARE @ErrorMessage NVARCHAR(MAX);

EXEC InsertStudentAndUser

@Email = 'example@student.com',

@Password = 'password123',

@FullName = 'Nguyen Van A',

@IdentityNumber = '123456789',

@BirthDate = '2000-01-01',

@Gender = 'Nam',

@Department = 'Công nghệ thông tin',

@Status = 'Active',

@AdmissionDate = '2024-11-25',

@LearningStatus = 'Active',

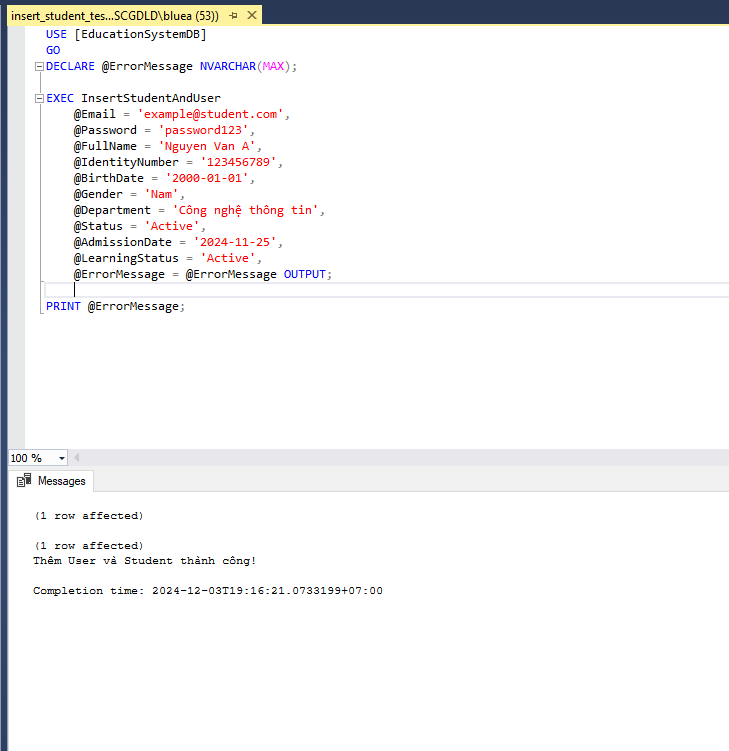
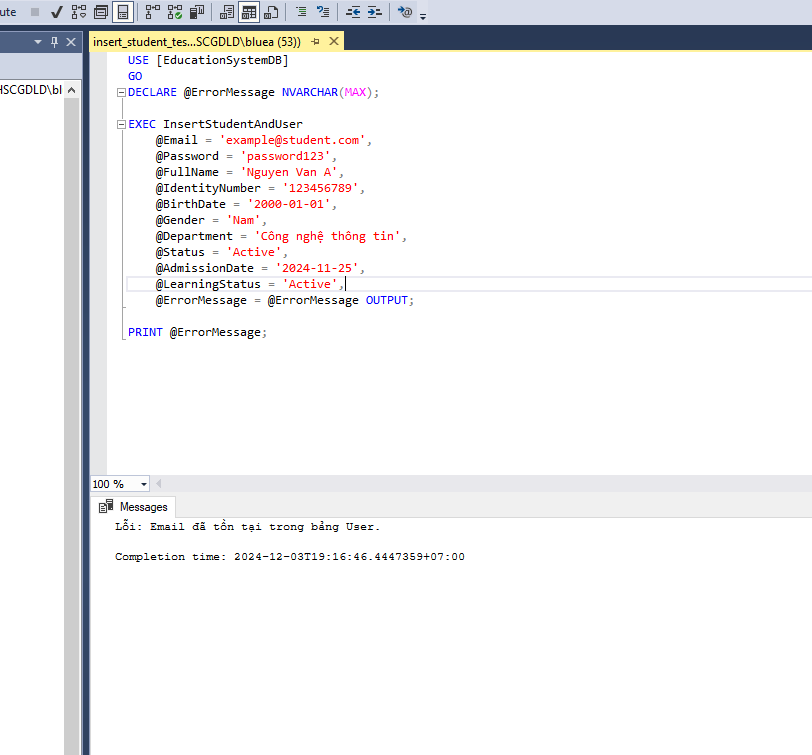
@ErrorMessage = @ErrorMessage OUTPUT;

PRINT @ErrorMessage;

SELECT TOP (200) Id, Email, Password, FullName, IdentityNumber, BirthDate, Gender, Department, Status

FROM [User]

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

Thêm thành công:  
  
Thêm trùng báo lỗi:  


1. Thủ tục UPDATE:

Mô tả thủ tục: Cập nhật thông tin học sinh và tài khoản người dùng, kiểm tra check validate giá trị nhập

– Kiểm tra validation các giá trị nhập

-- Kiểm tra Email có tồn tại trong bảng User chưa

-- Cập nhật User

-- Cập nhật Student

-- Thông báo kết quả

Input:

@UserId INT,

@Email VARCHAR(50),

@Password VARCHAR(50),

@FullName NVARCHAR(50),

@IdentityNumber VARCHAR(50) = NULL,

@BirthDate DATE = NULL,

@Gender NVARCHAR(50),

@Department NVARCHAR(50),

@Status NVARCHAR(50),

@AdmissionDate DATE,

@LearningStatus NVARCHAR(50),

@ErrorMessage NVARCHAR(MAX) OUTPUT

Output:   
Thông báo lỗi validate:  
Lỗi: Email không hợp lệ. Độ dài phải từ 1 đến 50 ký tự

Lỗi: Email không hợp lệ. Phải có định dạng hợp lệ (vd: example@domain.com).  
Lỗi: Password không hợp lệ. Độ dài phải từ 1 đến 50 ký tự.

Lỗi: Identity Number không hợp lệ. Độ dài không được quá 50 ký tự.

Lỗi: Admission Date không được để trống.

Lỗi: Email đã tồn tại trong bảng User.

Lỗi: Không tìm thấy User với Id đã cho.

Lỗi: Không tìm thấy Student với UserId đã cho.  
và các validate dữ liệu tương tự khác.

Thông báo khi thành công:  
Cập nhật thành công.

Câu lệnh tạo thủ tục:

USE [EducationSystemDB]

GO

CREATE PROCEDURE UpdateStudentAndUser

@UserId INT,

@Email VARCHAR(50),

@Password VARCHAR(50),

@FullName NVARCHAR(50),

@IdentityNumber VARCHAR(50) = NULL,

@BirthDate DATE = NULL,

@Gender NVARCHAR(50),

@Department NVARCHAR(50),

@Status NVARCHAR(50),

@AdmissionDate DATE,

@LearningStatus NVARCHAR(50),

@ErrorMessage NVARCHAR(MAX) OUTPUT

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

-- Kiểm tra độ dài Email

IF @Email IS NULL OR LEN(@Email) = 0 OR LEN(@Email) > 50

BEGIN

SET @ErrorMessage = N'Lỗi: Email không hợp lệ. Độ dài phải từ 1 đến 50 ký tự.';

RETURN;

END

-- Kiểm tra tính hợp lệ của Email

IF NOT (@Email LIKE '%@%.%')

BEGIN

SET @ErrorMessage = N'Lỗi: Email không hợp lệ. Phải có định dạng hợp lệ (vd: example@domain.com).';

RETURN;

END

-- Kiểm tra độ dài Password

IF @Password IS NULL OR LEN(@Password) = 0 OR LEN(@Password) > 50

BEGIN

SET @ErrorMessage = N'Lỗi: Password không hợp lệ. Độ dài phải từ 1 đến 50 ký tự.';

RETURN;

END

-- Kiểm tra độ dài Fullname

IF @FullName IS NULL OR LEN(@FullName) = 0 OR LEN(@FullName) > 50

BEGIN

SET @ErrorMessage = N'Lỗi: Fullname không hợp lệ. Độ dài phải từ 1 đến 50 ký tự.';

RETURN;

END

-- Kiểm tra độ dài Gender

IF @Gender IS NULL OR LEN(@Gender) = 0 OR LEN(@Gender) > 50

BEGIN

SET @ErrorMessage = N'Lỗi: Gender không hợp lệ. Độ dài phải từ 1 đến 50 ký tự.';

RETURN;

END

-- Kiểm tra độ dài Department

IF @Department IS NULL OR LEN(@Department) = 0 OR LEN(@Department) > 50

BEGIN

SET @ErrorMessage = N'Lỗi: Department không hợp lệ. Độ dài phải từ 1 đến 50 ký tự.';

RETURN;

END

-- Kiểm tra độ dài Identity Number

IF @IdentityNumber IS NOT NULL AND LEN(@IdentityNumber) > 50

BEGIN

SET @ErrorMessage = N'Lỗi: Identity Number không hợp lệ. Độ dài không được quá 50 ký tự.';

RETURN;

END

-- Check không empty Admission Date

IF @AdmissionDate IS NULL

BEGIN

SET @ErrorMessage = N'Lỗi: Admission Date không được để trống.';

RETURN;

END

-- Kiểm tra độ dài LearningStatus

IF @LearningStatus IS NULL OR LEN(@LearningStatus) = 0 OR LEN(@LearningStatus) > 50

BEGIN

SET @ErrorMessage = N'Lỗi: Learning Status không hợp lệ. Độ dài phải từ 1 đến 50 ký tự.';

RETURN;

END

-- Kiểm tra độ dài LearningStatus

IF LEN(@Status) = 0 OR LEN(@Status) > 50

BEGIN

SET @ErrorMessage = N'Lỗi: Status không hợp lệ. Độ dài phải từ 1 đến 50 ký tự.';

RETURN;

END

-- Kiểm tra Email có tồn tại trong bảng User chưa

IF EXISTS (SELECT 1 FROM [User] WHERE Id != @UserId AND Email = @Email)

BEGIN

SET @ErrorMessage = N'Lỗi: Email đã tồn tại trong bảng User.';

RETURN;

END

-- Cập nhật User

UPDATE [dbo].[User]

SET

Email = @Email,

Password = @Password,

FullName = @FullName,

IdentityNumber = @IdentityNumber,

BirthDate = @BirthDate,

Gender = @Gender,

Department = @Department,

Status = @Status

WHERE Id = @UserId;

IF @@ROWCOUNT = 0

BEGIN

SET @ErrorMessage = N'Lỗi: Không tìm thấy User với Id đã cho.';

ROLLBACK TRANSACTION;

RETURN;

END

-- Cập nhật Student

UPDATE [dbo].[Student]

SET

AdmissionDate = @AdmissionDate,

LearningStatus = @LearningStatus

WHERE UserId = @UserId;

IF @@ROWCOUNT = 0

BEGIN

SET @ErrorMessage = N'Lỗi: Không tìm thấy Student với UserId đã cho.';

ROLLBACK TRANSACTION;

RETURN;

END

-- Hoàn tất giao dịch

COMMIT TRANSACTION;

SET @ErrorMessage = N'Cập nhật thành công.';

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

SET @ErrorMessage = ERROR\_MESSAGE();

END CATCH

END

Nhóm câu lệnh thực thi thủ tục mẫu:

USE [EducationSystemDB]

GO

DECLARE @ErrorMessage NVARCHAR(MAX);

EXEC UpdateStudentAndUser

@UserId = 9,

@Email = 'example123@email.com',

@Password = 'securepassword',

@FullName = 'Nguyen Van A',

@IdentityNumber = '123456789',

@BirthDate = '2000-01-01',

@Gender = 'Nam',

@Department = 'IT',

@Status = 'Active',

@AdmissionDate = '2024-09-01',

@LearningStatus = 'Active',

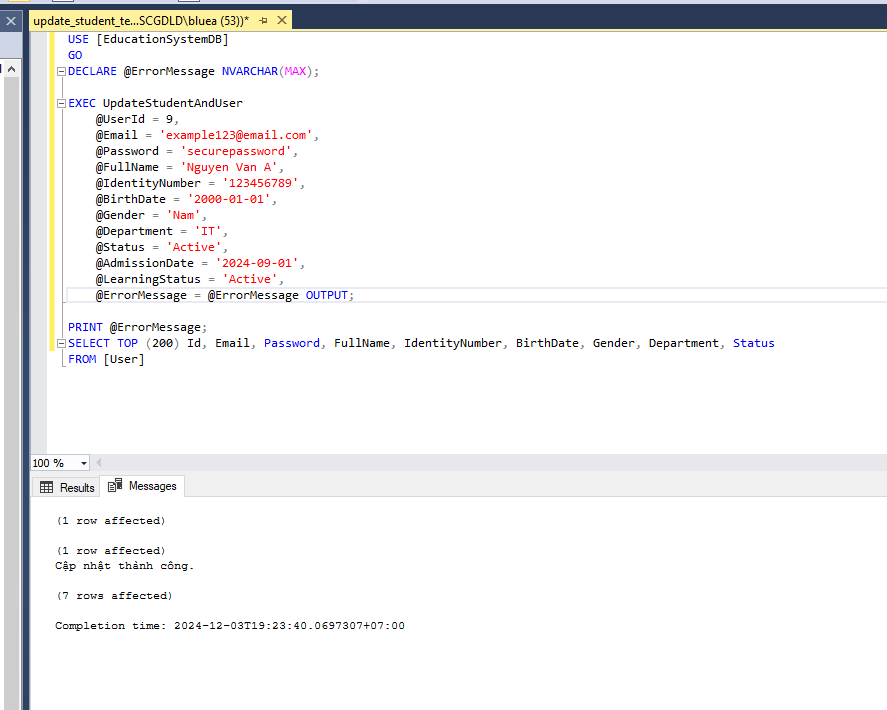
@ErrorMessage = @ErrorMessage OUTPUT;

PRINT @ErrorMessage;

SELECT TOP (200) Id, Email, Password, FullName, IdentityNumber, BirthDate, Gender, Department, Status

FROM [User]

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

Cập nhật thành công:  


1. Thủ tục DELETE:

Mô tả thủ tục: Xoá thông tin học sinh và tài khoản người dùng, check validate tồn tại

-- Kiểm tra UserId có tồn tại trong bảng User hay không

-- Xóa Student

-- Xóa User

-- Thông báo kết quả

Input:

@UserId INT,

@ErrorMessage NVARCHAR(MAX) OUTPUT

Output:   
Thông báo lỗi validate:  
Lỗi: Không tìm thấy User với Id đã cho.

Lỗi: Không thể xóa User hoặc UserId không hợp lệ.

Thông báo khi thành công:  
Xóa thành công.

Câu lệnh tạo thủ tục:

USE [EducationSystemDB]

GO

CREATE PROCEDURE DeleteStudentAndUser

@UserId INT,

@ErrorMessage NVARCHAR(MAX) OUTPUT

AS

BEGIN

BEGIN TRANSACTION;

-- Kiểm tra UserId có tồn tại trong bảng User hay không

IF NOT EXISTS (SELECT 1 FROM [dbo].[User] WHERE Id = @UserId)

BEGIN

SET @ErrorMessage = N'Lỗi: Không tìm thấy User với Id đã cho.';

ROLLBACK TRANSACTION;

RETURN;

END

-- Kiểm tra trạng thái User

IF EXISTS (SELECT 1 FROM [dbo].[User] WHERE Id = @UserId AND Status = 'Active')

BEGIN

SET @ErrorMessage = N'Lỗi: Không thể xóa User đang ở trạng thái hoạt động.';

ROLLBACK TRANSACTION;

RETURN;

END

BEGIN TRY

-- Xóa Student

DELETE FROM [dbo].[Student]

WHERE UserId = @UserId;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

SET @ErrorMessage = ERROR\_MESSAGE();

RETURN;

END CATCH

BEGIN TRY

-- Xóa User

DELETE FROM [dbo].[User]

WHERE Id = @UserId;

-- Kiểm tra xem việc xóa có thành công hay không

IF @@ROWCOUNT = 0

BEGIN

SET @ErrorMessage = N'Lỗi: Không thể xóa User hoặc UserId không hợp lệ.';

ROLLBACK TRANSACTION;

RETURN;

END

-- Hoàn tất giao dịch

COMMIT TRANSACTION;

SET @ErrorMessage = N'Xóa thành công.';

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

SET @ErrorMessage = ERROR\_MESSAGE();

END CATCH

END

Nhóm câu lệnh thực thi thủ tục mẫu:

USE [EducationSystemDB]

GO

DECLARE @ErrorMessage NVARCHAR(MAX);

EXEC DeleteStudentAndUser

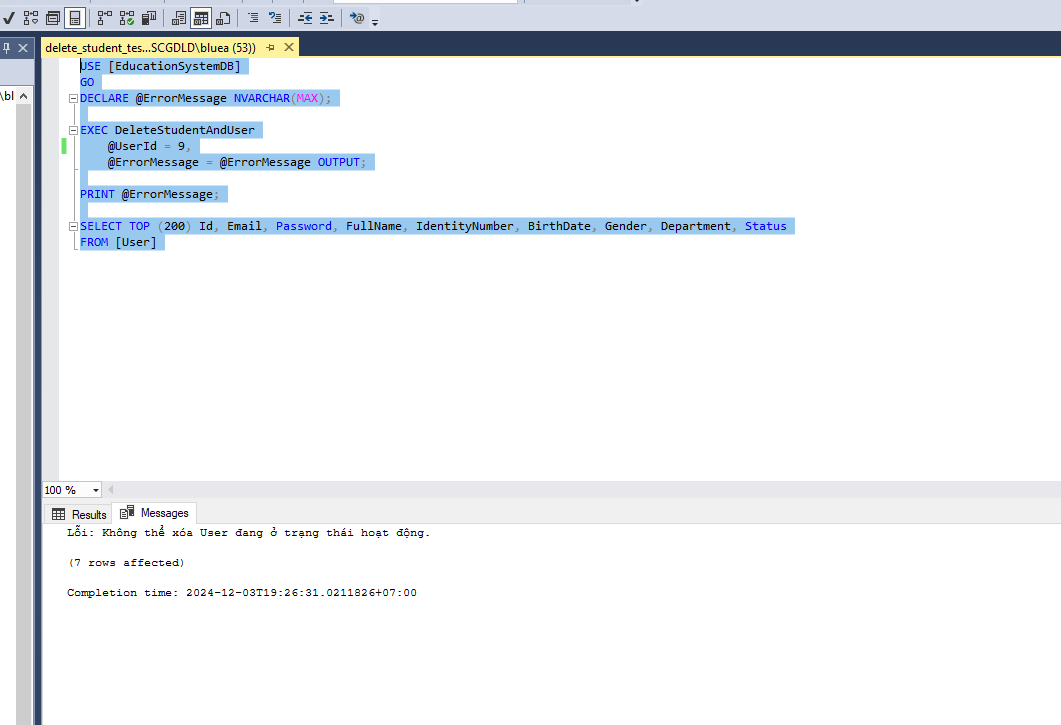
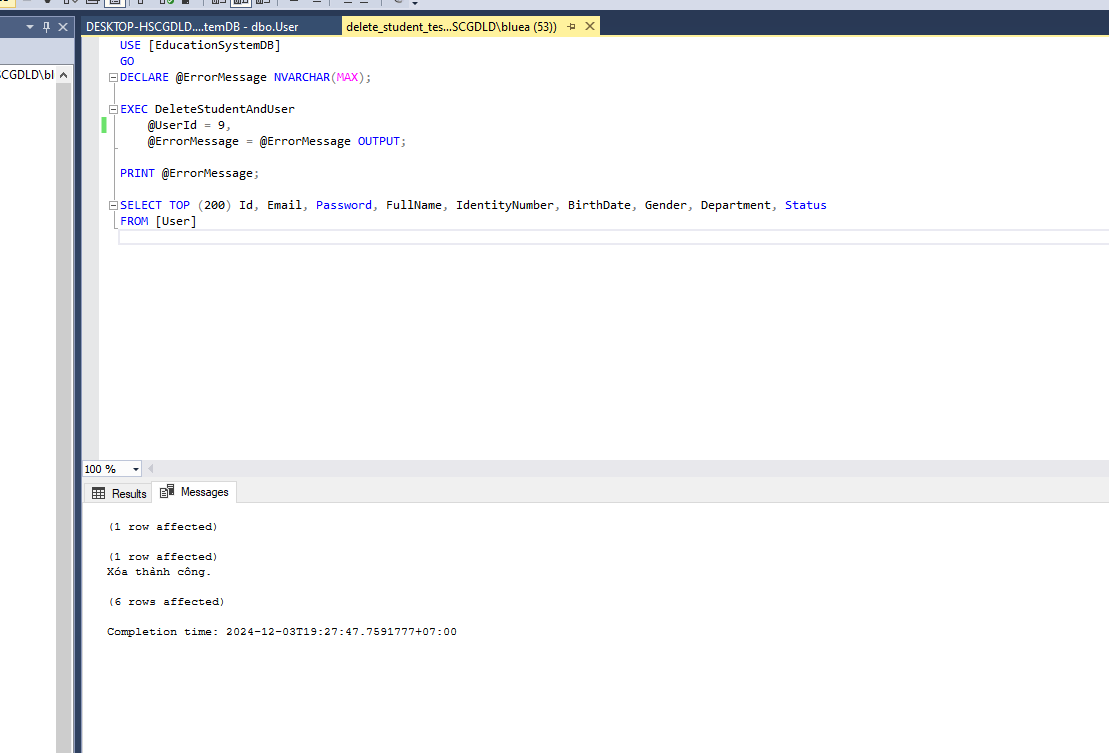
@UserId = 9,

@ErrorMessage = @ErrorMessage OUTPUT;

PRINT @ErrorMessage;

SELECT TOP (200) Id, Email, Password, FullName, IdentityNumber, BirthDate, Gender, Department, Status

FROM [User]

Kết quả các màn hình hiển thị từ DBMS minh họa cho việc thực hiện thủ tục  
  
Không thể xóa trong người dùng trong trạng thái hoạt động:  
  
  
Xóa thành công:  
  


**2.2 TRIGGER**

1. Trigger 1 (hoặc Nhóm các Trigger 1)

Mô tả trigger: Sinh viên không được đăng kí môn trùng trong Thời khóa biểu.  
-- Kiểm tra trùng lịch  
-- Ghi nhận lỗi nếu phát hiện lịch trùng  
-- Thêm bản ghi nếu không trùng

– Thông báo kết quả

Thông báo lỗi:  
Sinh viên đã đăng ký môn học trùng thời gian trong thời khóa biểu!  
Thông báo thành công:   
Thêm lịch học thành công.

Câu lệnh tạo (các) trigger:

USE [EducationSystemDB]

GO

CREATE TRIGGER trg\_PreventDuplicateSchedule

ON Timetable

INSTEAD OF INSERT

AS

BEGIN

-- Kiểm tra trùng lịch

IF EXISTS (

SELECT 1

FROM Timetable AS t

INNER JOIN Schedule AS s1 ON t.ScheduleId = s1.Id

INNER JOIN Inserted AS i ON t.StudentId = i.StudentId

INNER JOIN Schedule AS s2 ON i.ScheduleId = s2.Id

WHERE

(s1.StartTime < s2.EndTime AND s1.EndTime > s2.StartTime) -- Thời gian bị trùng

)

BEGIN

-- Ghi nhận lỗi nếu phát hiện lịch trùng

PRINT N'Sinh viên đã đăng ký môn học trùng thời gian trong thời khóa biểu!';

ROLLBACK TRANSACTION;

END

ELSE

BEGIN

-- Thêm bản ghi nếu không trùng

PRINT N'Thêm lịch học thành công.';

INSERT INTO Timetable (StudentId, ScheduleId)

SELECT StudentId, ScheduleId

FROM Inserted;

END

END;

Nhóm câu lệnh kiểm tra trigger: (*minh họa đủ các trường hợp nêu ra trong trigger*)

USE [EducationSystemDB]

GO

-- Insert overlapping timetable record

BEGIN TRY

INSERT INTO Timetable (StudentId, ScheduleId)

VALUES (2, 3); -- Student 1 with Schedule 2 (overlaps with Schedule 1)

END TRY

BEGIN CATCH

END CATCH

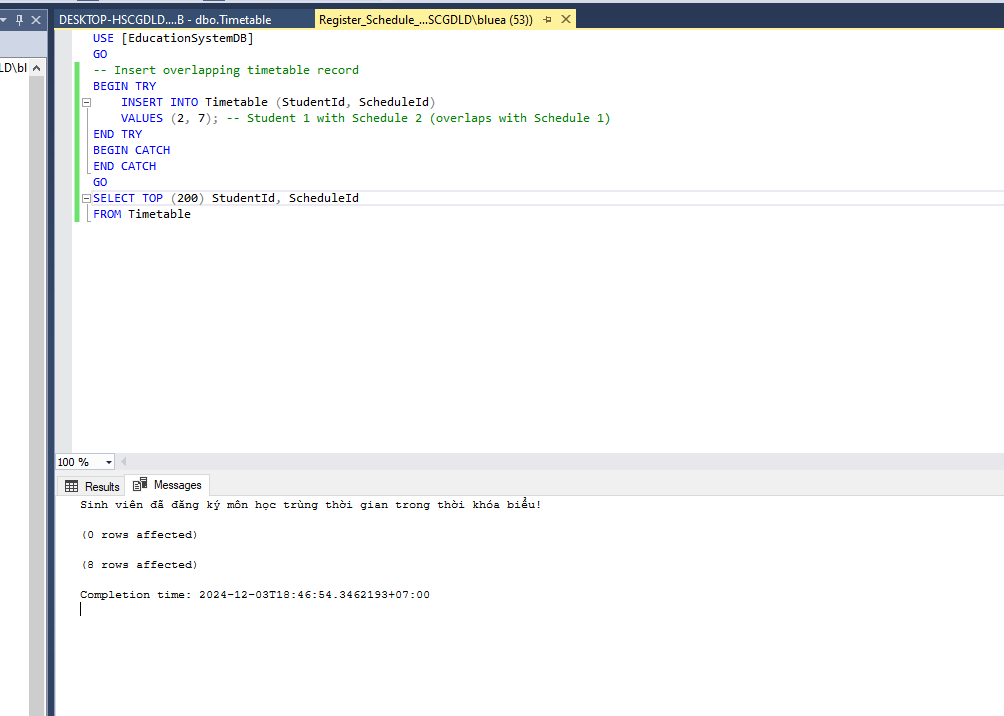
GO

SELECT TOP (200) StudentId, ScheduleId

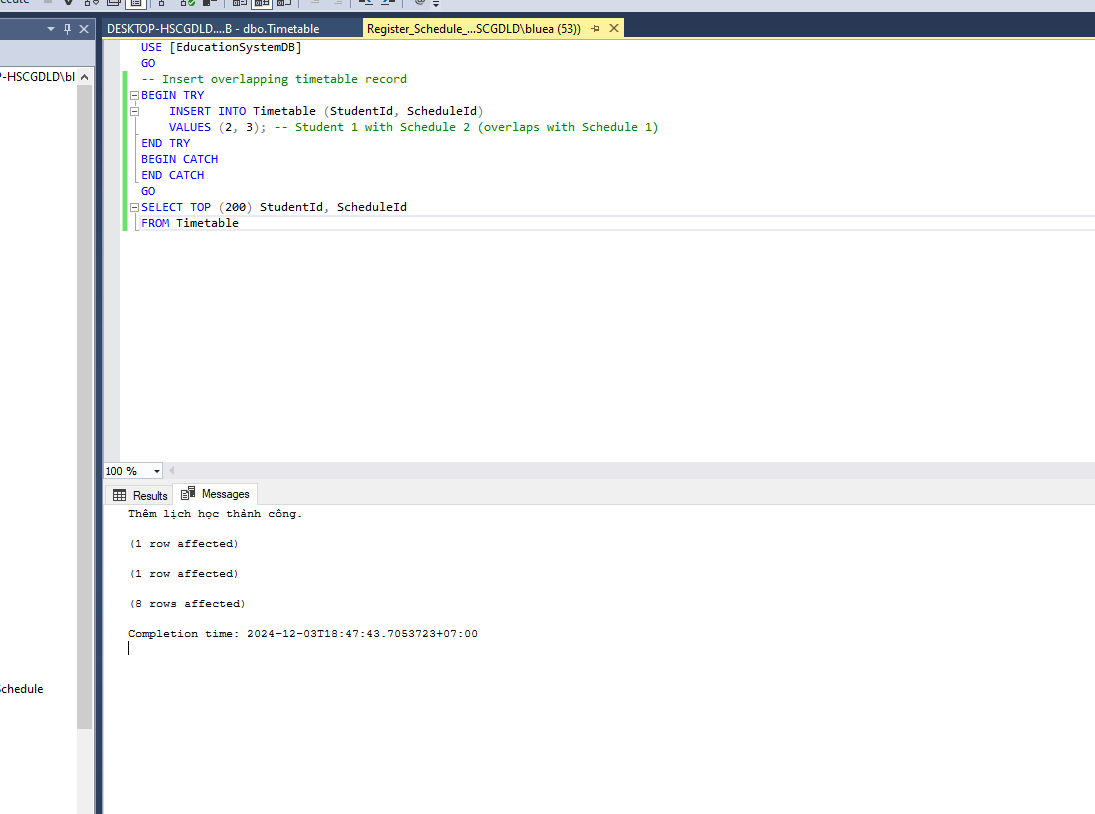
FROM Timetable

Kết quả các màn hình hiển thị từ DBMS minh họa cho trigger.

Thêm trùng lịch



Thêm thành công:



1. Trigger 2

Mô tả trigger: ta có thể cập nhật số học sinh mỗi khoa đếm theo tình trạng học tập (thuộc tính status vừa thêm vào User ở mục 1 (active or deactive))Chỉ tính active.  
-- Tạo bảng tạm để lưu các Khoa bị ảnh hưởng  
 -- Lấy tất cả khoa có xảy ra thêm hoặc xoá khoa không trùng thêm vào bảng tạm lưu để xử lý  
-- Cập nhật thống kê cho các khoa bị ảnh hưởng  
-- Thêm mới các Department chưa có trong DepartmentStats

Câu lệnh tạo (các) trigger:

USE [EducationSystemDB]

GO

CREATE TRIGGER TR\_UpdateDepartmentStats\_Student

ON [Student]

AFTER INSERT, UPDATE, DELETE

AS

BEGIN

SET NOCOUNT ON;

DECLARE @AffectedDepts TABLE (Department NVARCHAR(50));

-- Lấy tất cả department có xảy ra thêm hoặc xoá department không trùng thêm vào bảng tạm lưu để xử lý

INSERT INTO @AffectedDepts

SELECT DISTINCT u.Department

FROM [User] u

INNER JOIN inserted i ON u.Id = i.UserId

WHERE u.Department IS NOT NULL

UNION

SELECT DISTINCT u.Department

FROM [User] u

INNER JOIN deleted d ON u.Id = d.UserId

WHERE u.Department IS NOT NULL;

-- Cập nhật thống kê cho các Department bị ảnh hưởng

UPDATE ds

SET ds.TotalUsers = (

SELECT COUNT(\*)

FROM [User] u

INNER JOIN Student s ON u.Id = s.UserId

WHERE u.Department = ds.Department AND u.Status = 'Active' AND s.LearningStatus = 'Active'

),

ds.LastUpdated = GETDATE()

FROM DepartmentStats ds

WHERE ds.Department IN (SELECT Department FROM @AffectedDepts);

-- Thêm mới các Department chưa có trong DepartmentStats

INSERT INTO DepartmentStats (Department, TotalUsers, LastUpdated)

SELECT

dept.Department,

(

SELECT COUNT(\*)

FROM [User] u

INNER JOIN Student s ON u.Id = s.UserId

WHERE u.Department = dept.Department AND u.Status = 'Active' AND s.LearningStatus = 'Active'

),

GETDATE()

FROM @AffectedDepts dept

WHERE NOT EXISTS (

SELECT 1

FROM DepartmentStats ds

WHERE ds.Department = dept.Department

);

END

Nhóm câu lệnh kiểm tra trigger: (*minh họa đủ các trường hợp nêu ra trong trigger*)

USE [EducationSystemDB]

GO

DECLARE @ErrorMessage NVARCHAR(MAX);

EXEC InsertStudentAndUser

@Email = 'student4@gmail.com',

@Password = N'123',

@FullName = N'Student 5',

@IdentityNumber = N'023984821392',

@BirthDate = N'2000-01-01',

@Gender = N'Nam',

@Department = N'Kinh tế',

@Status = N'Active',

@AdmissionDate = N'2003-06-15',

@LearningStatus = N'Active',

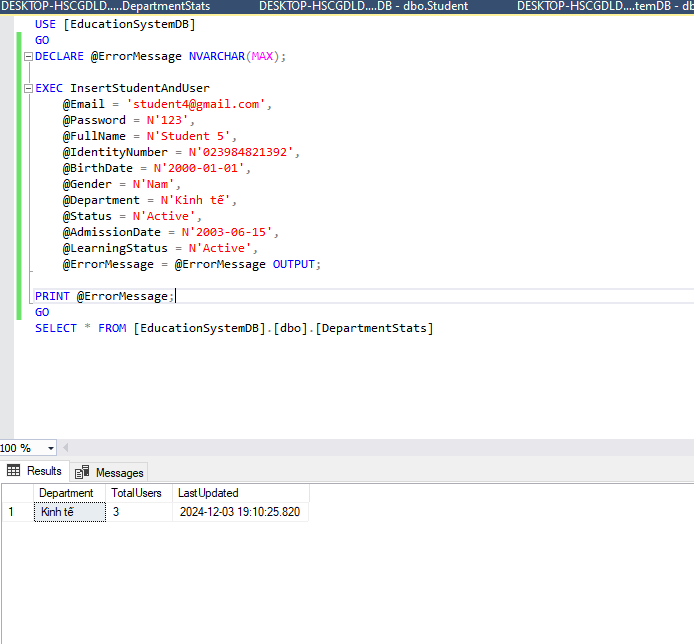
@ErrorMessage = @ErrorMessage OUTPUT;

PRINT @ErrorMessage;

GO

SELECT \* FROM [EducationSystemDB].[dbo].[DepartmentStats]

Kết quả các màn hình hiển thị từ DBMS minh họa cho trigger.

Thêm một sinh viên mới và cập nhật đếm số lượng người trong khoa:  


**2.3 THỦ TỤC**

1. Thủ tục 1

Mô tả thủ tục: Truy vấn thời khóa biểu của sinh viên

Input:  
@StudentId INT,

@SemesterCode NVARCHAR(50) = NULL,

@Date DATE = NULL,

@LessonNum INT = NULL,

@CourseName NVARCHAR(50) = NULL,

@ClassroomName NVARCHAR(50) = NULL

Output: danh sách lịch trình của sinh viên

Câu lệnh tạo thủ tục:

USE [EducationSystemDB]

GO

CREATE OR ALTER PROCEDURE GetStudentSchedule

@StudentId INT,

@SemesterCode NVARCHAR(50) = NULL,

@Date DATE = NULL,

@LessonNum INT = NULL,

@CourseName NVARCHAR(50) = NULL,

@ClassroomName NVARCHAR(50) = NULL

AS

BEGIN

SELECT

s.StartTime AS "StartTime",

s.EndTime AS "EndTime",

s.LessonNum AS "LessonNum",

s.RoomName AS "RoomName",

c.Name AS "SubjectName",

cl.Name AS "ClassName",

cl.SemesterCode AS "SemesterCode",

t.FullName AS "TeacherName",

te.Degree AS "TeacherDegree"

FROM Timetable tt

JOIN Schedule s ON tt.ScheduleId = s.Id

JOIN Classroom cl ON s.ClassroomId = cl.Id

JOIN Course c ON cl.CourseId = c.Id

JOIN Teacher te ON cl.TeacherId = te.UserId

JOIN [User] t ON te.UserId = t.Id

WHERE tt.StudentId = @StudentId

AND (

(@SemesterCode IS NULL OR cl.SemesterCode = @SemesterCode)

AND (@Date IS NULL OR CAST(s.StartTime AS DATE) = @Date)

AND (@LessonNum IS NULL OR s.LessonNum = @LessonNum)

AND (@CourseName IS NULL OR c.Name LIKE '%' + @CourseName + '%')

AND (@ClassroomName IS NULL OR cl.Name LIKE '%' + @ClassroomName + '%')

)

ORDER BY s.StartTime, s.LessonNum;

END

Nhóm câu lệnh thực thi thủ tục mẫu:

USE [EducationSystemDB]

GO

-- Truy vấn tất cả thời khóa biểu của sinh viên trong học kỳ

EXEC GetStudentSchedule

@StudentId = 2,

@SemesterCode = 'Spring\_HK1'

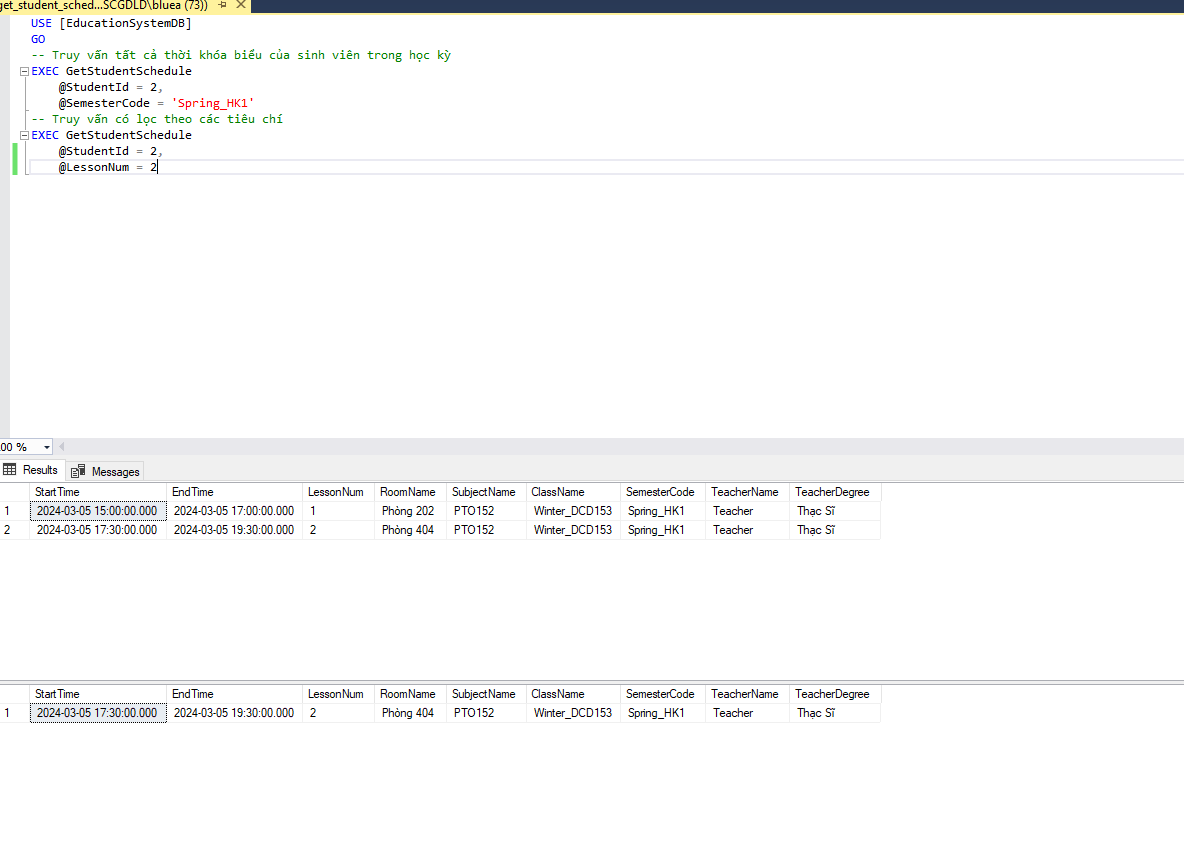
-- Truy vấn có lọc theo các tiêu chí

EXEC GetStudentSchedule

@StudentId = 2,

@LessonNum = 2

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

Truy vấn danh sách lịch học lọc theo thuộc tính:  


1. Thủ tục 2

Mô tả thủ tục: In ra bảng điểm một môn học của một sinh viên tương ứng

Input:  
@StudentId INT,

@SemesterCode NVARCHAR(50)

Output: Bảng điểm chứa các cột ứng với các giá trị tương ứng như Môn, lớp, Điểm bài tập số 1, Hệ số BT1, Điểm bài tập số 2, Hệ số BT2, Điểm bài kiểm tra, hệ số bài KT, Điểm tổng kết môn.

Câu lệnh tạo thủ tục:

USE [EducationSystemDB]

GO

CREATE OR ALTER PROCEDURE GetStudentCourseGrades

@StudentId INT,

@SemesterCode NVARCHAR(50)

AS

BEGIN

WITH ExerciseGrades AS (

-- Lấy điểm các bài tập và đánh số thứ tự cho mỗi bài tập trong lớp

SELECT

cl.Id AS ClassroomId,

se.StudentId,

se.Point AS ExercisePoint,

e.Coefficient AS ExerciseCoef,

ROW\_NUMBER() OVER (PARTITION BY cl.Id ORDER BY e.EndTime) AS ExerciseNum

FROM StudentExercise se

JOIN Exercise e ON se.ExerciseId = e.Id

JOIN Classroom cl ON e.ClassroomId = cl.Id

WHERE se.StudentId = @StudentId

AND cl.SemesterCode = @SemesterCode

),

ExamGrades AS (

-- Tính điểm kiểm tra dựa trên số câu đúng

SELECT

e.ClassroomId,

se.StudentId,

e.Coefficient AS ExamCoef,

CAST(COUNT(CASE WHEN c.Answer = q.Answer THEN 1 END) AS FLOAT) / COUNT(\*) \* 10 AS ExamPoint

FROM StudentExam se

JOIN Exam e ON se.ExamId = e.Id

JOIN Choice c ON se.ChoiceId = c.Id

JOIN Question q ON c.QuestionId = q.Id

WHERE se.StudentId = @StudentId

GROUP BY e.ClassroomId, se.StudentId, e.Coefficient

)

SELECT

c.Name AS [Môn học],

cl.Name AS [Lớp học],

cl.SemesterCode AS [Học kỳ],

MAX(CASE WHEN eg.ExerciseNum = 1 THEN eg.ExercisePoint END) AS [Điểm BT1],

MAX(CASE WHEN eg.ExerciseNum = 1 THEN eg.ExerciseCoef END) AS [Hệ số BT1],

MAX(CASE WHEN eg.ExerciseNum = 2 THEN eg.ExercisePoint END) AS [Điểm BT2],

MAX(CASE WHEN eg.ExerciseNum = 2 THEN eg.ExerciseCoef END) AS [Hệ số BT2],

exg.ExamPoint AS [Điểm kiểm tra],

exg.ExamCoef AS [Hệ số kiểm tra],

sc.Point AS [Điểm tổng kết]

FROM StudentClassroom sc

JOIN Classroom cl ON sc.ClassroomId = cl.Id

JOIN Course c ON cl.CourseId = c.Id

LEFT JOIN ExerciseGrades eg ON sc.ClassroomId = eg.ClassroomId AND sc.StudentId = eg.StudentId

LEFT JOIN ExamGrades exg ON sc.ClassroomId = exg.ClassroomId AND sc.StudentId = exg.StudentId

WHERE sc.StudentId = @StudentId

AND cl.SemesterCode = @SemesterCode

GROUP BY

c.Name,

cl.Name,

cl.SemesterCode,

exg.ExamPoint,

exg.ExamCoef,

sc.Point

ORDER BY c.Name

END

Nhóm câu lệnh thực thi thủ tục mẫu:

USE [EducationSystemDB]

GO

-- In bảng điểm của sinh viên trong học kỳ

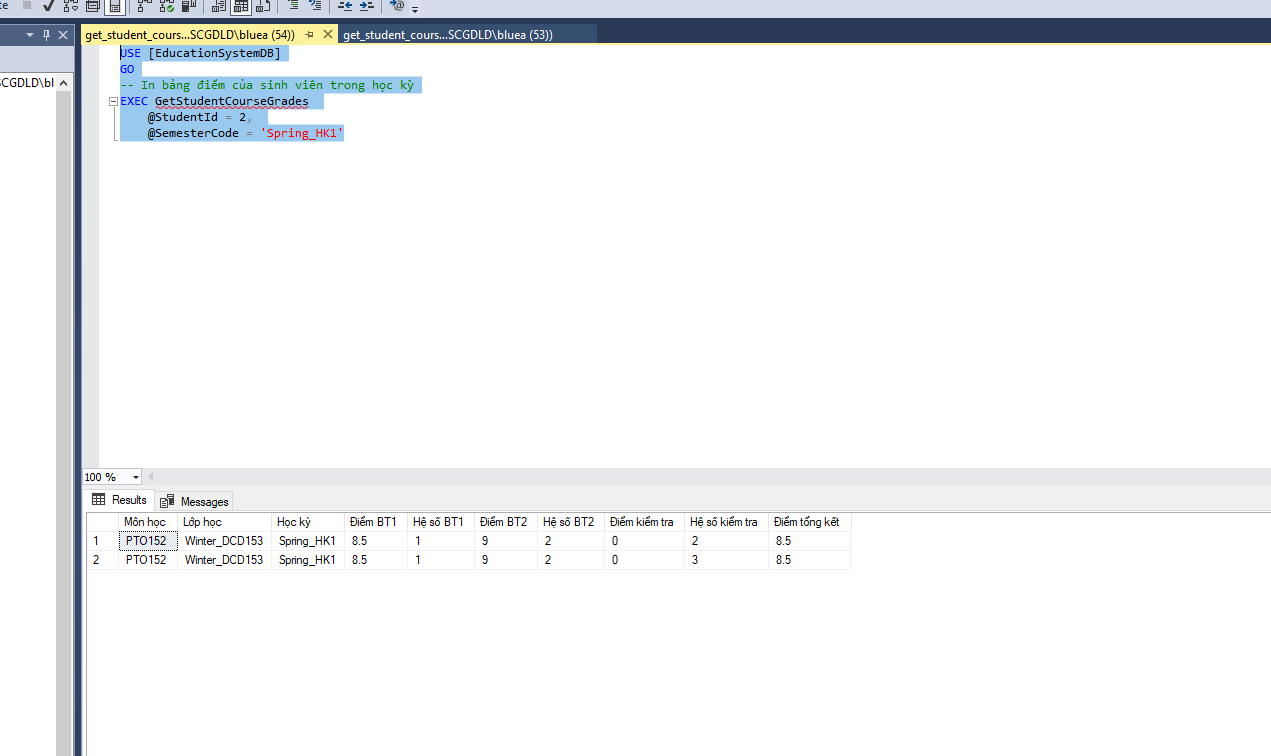
EXEC GetStudentCourseGrades

@StudentId = 2,

@SemesterCode = 'Spring\_HK1'

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

Truy vấn bảng điểm :



**2.4 HÀM**

**Hàm 1**

**Mô tả hàm:** Tính điểm của sinh viên trong 1 lớp học

-- Tính điểm thi (tổng điểm của các câu trả lời đúng có trọng số)

-- Tính điểm bài tập (tổng điểm bài tập có trọng số)

Tổng điểm cuối = (Tổng điểm có trọng số) / (Tổng hệ số)

**Input:**

@StudentId INT,

@ClassroomId INT

**Output:** Điểm final

**Câu lệnh hàm:**

USE [EducationSystemDB]

GO

CREATE OR ALTER FUNCTION CalculateStudentClassroomFinalGrade

(

@StudentId INT,

@ClassroomId INT

)

RETURNS FLOAT

AS

BEGIN

DECLARE @FinalGrade FLOAT = 0

DECLARE @ExamGrade FLOAT = 0

DECLARE @ExerciseGrade FLOAT = 0

DECLARE @TotalCoefficient INT = 0

-- Calculate exam grade (weighted sum of correct answers' points)

SELECT

@ExamGrade = SUM(CASE WHEN c.Answer = q.Answer THEN q.Point ELSE 0 END \* e.Coefficient),

@TotalCoefficient = @TotalCoefficient + SUM(e.Coefficient)

FROM StudentExam se

JOIN Exam e ON se.ExamId = e.Id

JOIN Choice c ON se.ChoiceId = c.Id

JOIN Question q ON c.QuestionId = q.Id

WHERE se.StudentId = @StudentId

AND e.ClassroomId = @ClassroomId

-- Calculate exercise grade (weighted sum of exercise points)

SELECT

@ExerciseGrade = SUM(se.Point \* e.Coefficient),

@TotalCoefficient = @TotalCoefficient + SUM(e.Coefficient)

FROM StudentExercise se

JOIN Exercise e ON se.ExerciseId = e.Id

WHERE se.StudentId = @StudentId

AND e.ClassroomId = @ClassroomId

AND se.Point IS NOT NULL

-- Calculate final grade

SET @FinalGrade = (@ExamGrade + @ExerciseGrade) / NULLIF(@TotalCoefficient, 0)

RETURN @FinalGrade

END

GO

Nhóm câu lệnh thực thi thủ tục mẫu:

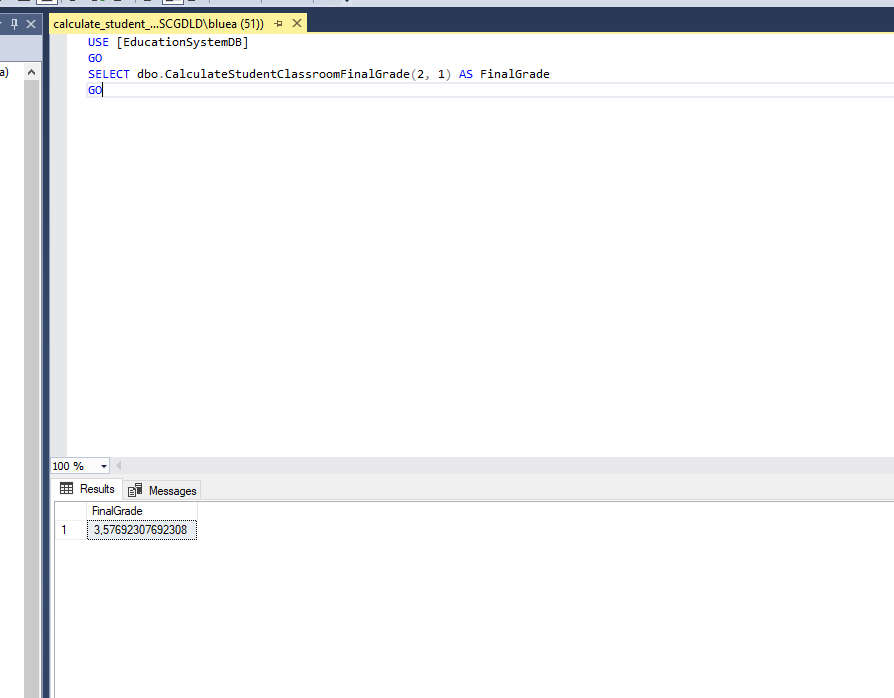
USE [EducationSystemDB]

GO

SELECT dbo.CalculateStudentClassroomFinalGrade(2, 1) AS FinalGrade

GO

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

Kết quả điểm cuối môn:  


**Hàm 2**

**Mô tả hàm:** Tính điểm trung bình tích lũy, tín chỉ tích lũy của 1 SV

-- Tính điểm cuối kỳ cho từng lớp học

-- Nhận điểm cao nhất cho mỗi khóa học, cho nỗ lực nào là tốt nhất

**Input:**

@StudentId INT

**Output:** Thông tin Tổng tín chỉ, điểm GPA, chi tiết môn

**Câu lệnh hàm:**

USE [EducationSystemDB]

GO

CREATE OR ALTER FUNCTION CalculateStudentGPA

(

@StudentId INT

)

RETURNS TABLE

AS

RETURN

(

WITH ClassroomGrades AS (

-- Tính điểm cuối kỳ cho từng lớp học

SELECT DISTINCT

cl.CourseId,

c.Name AS CourseName,

c.NumberOfCredits,

cl.Id AS ClassroomId,

cl.SemesterCode,

dbo.CalculateStudentClassroomFinalGrade(@StudentId, cl.Id) AS FinalGrade

FROM StudentExercise se

JOIN Exercise e ON se.ExerciseId = e.Id

JOIN Classroom cl ON e.ClassroomId = cl.Id

JOIN Course c ON cl.CourseId = c.Id

WHERE se.StudentId = @StudentId

UNION

SELECT DISTINCT

cl.CourseId,

c.Name AS CourseName,

c.NumberOfCredits,

cl.Id AS ClassroomId,

cl.SemesterCode,

dbo.CalculateStudentClassroomFinalGrade(@StudentId, cl.Id) AS FinalGrade

FROM StudentExam se

JOIN Exam e ON se.ExamId = e.Id

JOIN Classroom cl ON e.ClassroomId = cl.Id

JOIN Course c ON cl.CourseId = c.Id

WHERE se.StudentId = @StudentId

),

BestGrades AS (

-- Nhận điểm cao nhất cho mỗi khóa học, cho nỗ lực nào là tốt nhất

SELECT

CourseId,

CourseName,

NumberOfCredits,

MAX(FinalGrade) AS BestGrade,

-- Chi tiết bổ sung về nỗ lực tốt nhất

STRING\_AGG(

CONCAT(

N'Học kì: ', SemesterCode,

N', Điểm số: ', CAST(FinalGrade AS DECIMAL(4,2))

),

' | '

) WITHIN GROUP (ORDER BY FinalGrade DESC) AS AllAttempts

FROM ClassroomGrades

GROUP BY CourseId, CourseName, NumberOfCredits

)

SELECT

SUM(NumberOfCredits) AS TotalCredits,

CAST(SUM(NumberOfCredits \* BestGrade) / NULLIF(SUM(NumberOfCredits), 0) AS DECIMAL(4,2)) AS GPA,

STRING\_AGG(

CONCAT(

CourseName,

' (', NumberOfCredits, N' tín chỉ)',

N': Điểm cao nhất = ', CAST(BestGrade AS DECIMAL(4,2)),

CHAR(13),

N'Toàn bộ: ', AllAttempts

),

CHAR(13) + CHAR(13)

) AS CourseDetails

FROM BestGrades

WHERE BestGrade >= 0 -- chỉ lấy điểm nếu lớn hơn 0

)

GO

Nhóm câu lệnh thực thi thủ tục mẫu:

USE [EducationSystemDB]

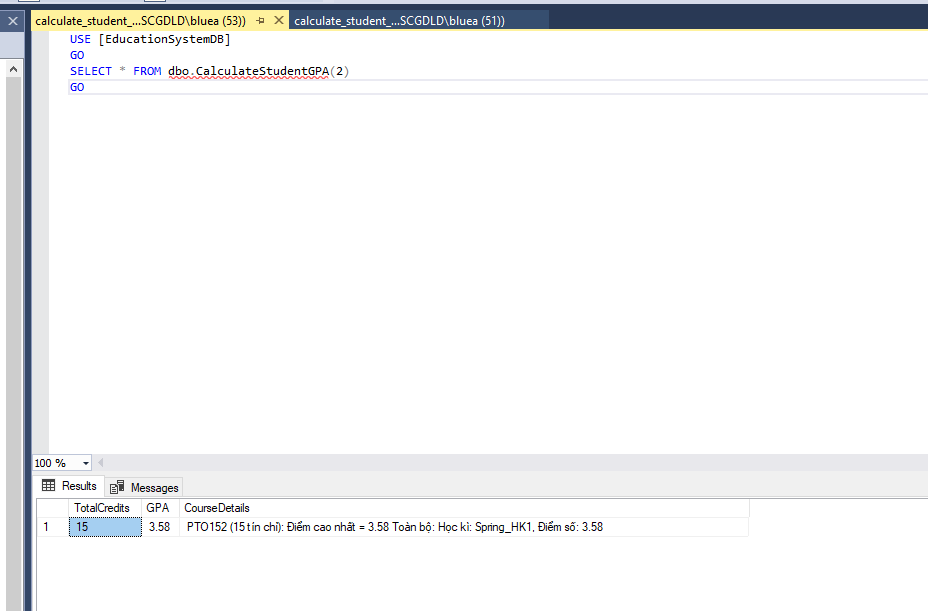
GO

SELECT \* FROM dbo.CalculateStudentGPA(2)

GO

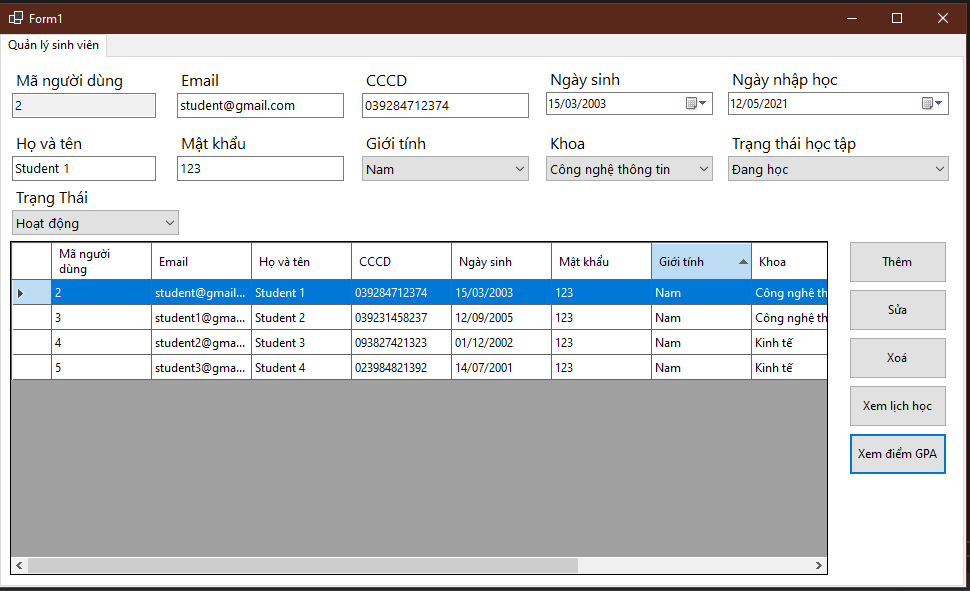
Kết quả các màn hình hiển thị từ DBMS minh họa cho việc thực hiện thủ tục

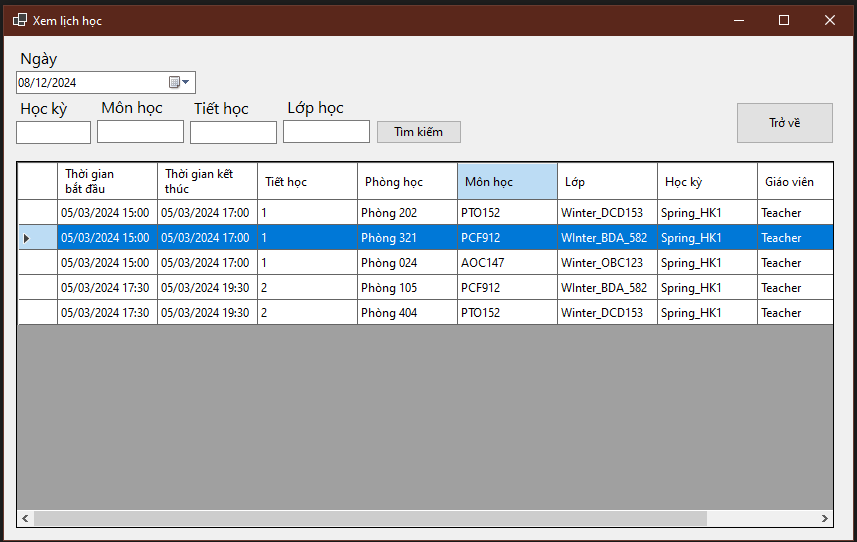
Tổng số tín chỉ, điểm GPA, chi tiết môn:



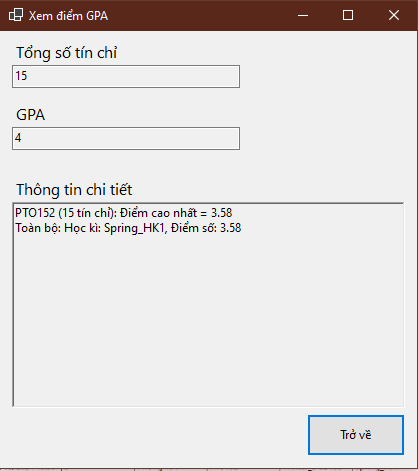
**3. ỨNG DỤNG KẾT NỐI VỚI CSDL**

Màn hình 1: Màn hình quản lý sinh viên thêm, sửa, xóa



Màn hình 2: Màn hình xem lịch học, lọc theo thuộc tính  


Màn hình 3: Màn hình xem điểm GPA của sinh viên



**4. Thiết kế ERD cập nhật mới nhất**

