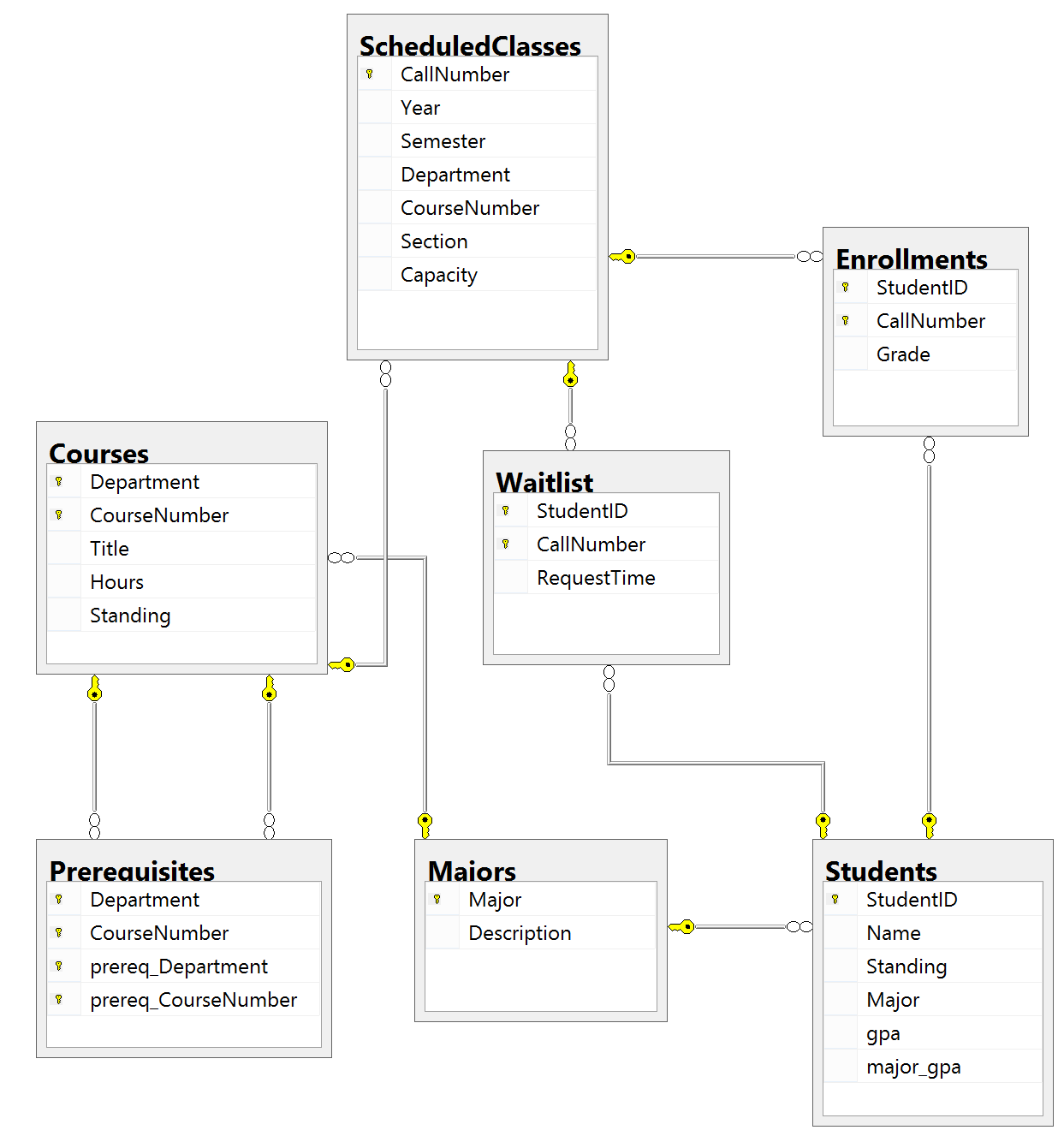
DESIGN A PROJECT

* Member of the Project: **Phuc Le**
* Description of the Project: this project is to develop an enrollment system for a school. The database designed is to record student information, the school’s available majors, all offered courses, and all scheduled courses. In addition, the database also helps users to manage and keep track of enrolling process.
* Software and Hardware Requirement for this project:
  + Microsoft SQL Server 2012 or newer version
  + PC running Windows operating system, at least Windows 7 SP1 or Windows server 2008 R2 SP1 is recommended.
  + NET 3.5 SP1 is also a requirement
  + At least 2GB of RAM, Processor Speed 2.0 GHz or faster, and at least 6 GB hard drive space are required.
* Project Interest: Although the project scope is small in comparison to a real enrollment system of a university, I feel very interested in developing it as my class project. This project may be a good demonstration of relational database mastery through the application of database design, usage of store procedures, indexes, user-defined function, and triggers, and the incorporation of audit tables.

DATABASE DIAGRAM



SQL SERVER DATABASE SCHEMA

/\* **Table Majors** \*/

CREATE TABLE Majors(  
 Major Varchar(3) Primary key,  
 Description Varchar(30) DEFAULT 'To be described');

INSERT INTO Majors VALUES ('ACC','Accounting');  
INSERT INTO Majors VALUES ('FIN','Finance');  
INSERT INTO Majors VALUES ('IS','Information Systems');  
INSERT INTO Majors VALUES ('MKT','Marketing');

/\* **Table Students** \*/  
CREATE TABLE Students(  
 StudentID Varchar(3) PRIMARY KEY,  
 Name Varchar(10) NOT NULL,  
 Standing Tinyint,  
 Major Varchar(3) CONSTRAINT fk\_students\_major REFERENCES Majors(Major),  
 gpa Decimal(2,1) DEFAULT 0.0,  
 major\_gpa Decimal(2,1));

INSERT INTO Students VALUES ('101','Andy',4,'IS',2.8,3.2);  
INSERT INTO Students VALUES ('102','Betty',2,null,3.2,null);  
INSERT INTO Students VALUES ('103','Cindy',3,'IS',2.5,3.5);  
INSERT INTO Students VALUES ('104','David',2,'FIN',3.3,3.0);  
INSERT INTO Students VALUES ('105','Ellen',1,null,2.8,null);  
INSERT INTO Students VALUES ('106','Frank',3,'MKT',3.1,2.9);  
INSERT INTO Students VALUES ('107','Jim',2,'MKT',3.7,null);  
INSERT INTO Students VALUES ('108','Susan',3,'MKT',3.5,3.4);  
INSERT INTO Students VALUES ('109','Anna',3,'FIN',3.4,3.2);  
INSERT INTO Students VALUES ('110','Jose',4,'MKT',3.2,3.5);  
INSERT INTO Students VALUES ('111','Mary',4,'ACC',3.8,null);  
INSERT INTO Students VALUES ('112','Lisa',2,'IS',3.5,3.6);  
INSERT INTO Students VALUES ('113','Alex',2,null,3.2,null);  
INSERT INTO Students VALUES ('114','Fillipe',3,'IS',3.3,3.3);  
INSERT INTO Students VALUES ('115','Joven',3,'FIN',3.1,3.4);  
INSERT INTO Students VALUES ('116','Camila',2,null,3.8,3.7);  
INSERT INTO Students VALUES ('117','Mario',2,null,3.4,3.6);  
INSERT INTO Students VALUES ('118','Ian',3,'FIN',2.4,3.2);  
INSERT INTO Students VALUES ('119','Monica',4,'FIN',2.7,3.0);  
INSERT INTO Students VALUES ('120','Claudia',3,'IS',3.9,3.7);  
INSERT INTO Students VALUES ('121','Kelly',2,'ACC',3.4,3.5);  
INSERT INTO Students VALUES ('122','Rosa',2,'MKT',3.1,null);  
INSERT INTO Students VALUES ('123','Sam',2,null,3.7,3.5);  
INSERT INTO Students VALUES ('124','Jason',3,'ACC',3.2,null);

/\* **Table Courses** \*/

CREATE TABLE Courses(  
 Department Varchar(3) NOT NULL

CONSTRAINT fk\_courses\_dept REFERENCES Majors(Major),  
 CourseNumber Varchar(3) NOT NULL,  
 Title Varchar(30) NOT NULL,  
 Hours Tinyint DEFAULT 1,  
 Standing Tinyint DEFAULT 1,  
 PRIMARY KEY (Department ,CourseNumber));

INSERT INTO Courses VALUES ('IS','300','Intro to MIS',3,2);  
INSERT INTO Courses VALUES ('IS','301','Business Communicatons',3,2);  
INSERT INTO Courses VALUES ('IS','310','Statistics',3,2);  
INSERT INTO Courses VALUES ('IS','340','Business Application',3,3);  
INSERT INTO Courses VALUES ('IS','355','Networks',3,3);  
INSERT INTO Courses VALUES ('IS','380','Database',3,3);  
INSERT INTO Courses VALUES ('IS','385','Systems',3,3);  
INSERT INTO Courses VALUES ('IS','480','Adv Database',3,4);  
INSERT INTO Courses VALUES ('FIN','300','Business Finance',3,3);

INSERT INTO Courses VALUES ('FIN','350','Investment Principles',3,3);  
INSERT INTO Courses VALUES ('ACC','320','Cost Accounting',3,3);  
INSERT INTO Courses VALUES ('ACC','470','Auditing',4,4);  
INSERT INTO Courses VALUES ('MKT','300','Basic Marketing',3,2);

/\* **Table ScheduledClasses** \*/  
CREATE TABLE ScheduledClasses(  
 CallNumber Numeric (5,0) PRIMARY KEY,  
 Year Numeric (4, 0) NOT NULL,  
 Semester Varchar(3) NOT NULL,  
 Department Varchar(3) NOT NULL,  
 CourseNumber Varchar(3) NOT NULL,  
 Section tinyint DEFAULT 1,  
 Capacity smallint DEFAULT 15);

ALTER TABLE ScheduledClasses  
ADD CONSTRAINT fk\_schclasses\_dept\_cnum FOREIGN KEY  
(Department, CourseNumber) REFERENCES Courses (Department, CourseNumber);

INSERT INTO ScheduledClasses VALUES (10110,2013,'Fa','IS','300',1,4);  
INSERT INTO ScheduledClasses VALUES (10115,2013,'Fa','IS','300',2,4);  
INSERT INTO ScheduledClasses VALUES (10120,2013,'Fa','IS','380',1,3);  
INSERT INTO ScheduledClasses VALUES (10121,2013,'Fa','IS','380',2,3);  
INSERT INTO ScheduledClasses VALUES (10125,2013,'Fa','IS','300',1,5);  
INSERT INTO ScheduledClasses VALUES (10130,2013,'Fa','IS','301',1,6);  
INSERT INTO ScheduledClasses VALUES (10131,2013,'Fa','IS','301',2,6);  
INSERT INTO ScheduledClasses VALUES (10135,2013,'Fa','IS','340',1,5);  
INSERT INTO ScheduledClasses VALUES (10141,2013,'Fa','FIN','300',1,4);  
INSERT INTO ScheduledClasses VALUES (10142,2013,'Fa','FIN','300',2,4);  
INSERT INTO ScheduledClasses VALUES (10145,2013,'Fa','FIN','350',2,5);  
INSERT INTO ScheduledClasses VALUES (10150,2013,'Fa','ACC','320',1,3);  
INSERT INTO ScheduledClasses VALUES (10160,2013,'Fa','MKT','300',1,3);

/\* **Table Prerequisites** \*/

CREATE TABLE Prerequisites(  
 Department Varchar(3),  
 CourseNumber Varchar(3),  
 prereq\_Department Varchar(3),  
 prereq\_CourseNumber Varchar(3),  
 PRIMARY KEY (Department, CourseNumber, prereq\_Department , prereq\_CourseNumber));

ALTER TABLE Prerequisites  
ADD CONSTRAINT fk\_prereq\_dept\_cnum FOREIGN KEY  
(Department, CourseNumber) REFERENCES Courses (Department,CourseNumber),  
CONSTRAINT fk\_prereq\_pdept\_pcnum FOREIGN KEY  
(prereq\_Department, prereq\_CourseNumber) REFERENCES courses (Department,CourseNumber);

INSERT INTO Prerequisites VALUES ('IS','380','IS','300');  
INSERT INTO Prerequisites VALUES ('IS','380','IS','301');  
INSERT INTO Prerequisites VALUES ('IS','380','IS','310');  
INSERT INTO Prerequisites VALUES ('IS','385','IS','310');  
INSERT INTO Prerequisites VALUES ('IS','355','IS','300');  
INSERT INTO Prerequisites VALUES ('IS','480','IS','380');  
INSERT INTO Prerequisites VALUES ('FIN','350','FIN','300');  
INSERT INTO Prerequisites VALUES ('ACC','470','ACC','320');

/\* **Table Enrollments** \*/  
CREATE TABLE Enrollments(

StudentID Varchar(3)

CONSTRAINT fk\_enrollments\_snum REFERENCES Students(StudentID),  
 CallNumber Numeric (5,0)

CONSTRAINT fk\_enrollments\_callnum REFERENCES  
 ScheduledClasses(CallNumber),  
 Grade Varchar(2),  
 PRIMARY KEY (StudentID, CallNumber));

/\* **Table Waitlist** \*/  
CREATE TABLE Waitlist (  
 StudentID Varchar(3)

CONSTRAINT fk\_waitlist\_snum REFERENCES Students(StudentID),  
 CallNumber Numeric (5,0)

CONSTRAINT fk\_waitlist\_callnum REFERENCES  
 ScheduledClasses(CallNumber),  
 RequestTime Datetime,

PRIMARY KEY (StudentID, CallNumber));

ALTER TABLE Waitlist ADD CONSTRAINT DF\_Waitlist DEFAULT GETDATE() FOR RequestTime;

/\* **Table Indexes** \*/

CREATE INDEX majors\_idx ON Majors (Major);

CREATE INDEX students\_idx ON Students (StudentID, Name);

CREATE INDEX Courses\_idx ON Courses (Department, CourseNumber);

CREATE INDEX ScheduledClasses\_idx ON ScheduledClasses (CallNumber);

AUDIT TABLES

CREATE TABLE Majors\_Audit(

AuditID INT IDENTITY PRIMARY KEY NOT NULL,

Major Varchar(3),

Description Varchar(30),

[User] Varchar (25),

Operator Varchar(6),

[DateTime] DateTime

);

CREATE TABLE Students\_Audit(

AuditID INT IDENTITY PRIMARY KEY NOT NULL,

StudentID Varchar(3),

Name Varchar(10),

Standing Tinyint,

Major Varchar(3),

gpa Decimal(2,1),

major\_gpa Decimal(2,1),

[User] Varchar (25),

Operator Varchar(6),

[DateTime] DateTime

);

CREATE TABLE Courses\_Audit(

AuditID INT IDENTITY PRIMARY KEY NOT NULL,

Department Varchar(3),

CourseNumber Varchar(3),

Title Varchar(30),

Hours Tinyint,

Standing Tinyint,

[User] Varchar(25),

Operator Varchar(6),

[DateTime] DateTime

);

CREATE TABLE ScheduledClasses\_Audit(

AuditID INT IDENTITY PRIMARY KEY NOT NULL,

CallNumber Numeric (5,0),

Year Numeric (4,0),

Semester Varchar(3),

Department Varchar(3),

CourseNumber Varchar(3),

Section tinyint,

Capacity smallint,

[User] Varchar(25),

Operator Varchar(6),

[DateTime] DateTime

);

CREATE TABLE Prerequisites\_Audit(

AuditID INT IDENTITY PRIMARY KEY NOT NULL,

Department Varchar(3),

CourseNumber Varchar(3),

prereq\_Department Varchar(3),

prereq\_CourseNumber Varchar(3),

[User] Varchar(25),

Operator Varchar(6),

[DateTime] DateTime

);

CREATE TABLE Enrollments\_Audit(

AuditID INT IDENTITY PRIMARY KEY NOT NULL,

StudentID Varchar(3),

CallNumber Numeric (5,0),

Grade Varchar(2),

[User] Varchar(25),

Operator Varchar(6),

[DateTime] DateTime

);

CREATE TABLE Waitlist\_Audit (

AuditID INT IDENTITY PRIMARY KEY NOT NULL,

StudentID Varchar(3),

CallNumber Numeric (5,0),

RequestTime DateTime,

[User] Varchar(25),

Operator Varchar(6),

[DateTime] DateTime

);

STORED PROCEDURES AND TRIGGERS FOR EACH TABLE

/\* **Table Majors** \*/

CREATE PROC ***sp\_InsertMajor***

@Major Varchar(3),

@Description Varchar(30)

AS

BEGIN

IF NOT EXISTS (SELECT \* FROM Majors WHERE Major = @Major)

BEGIN

IF @Major IS NULL

THROW 50001, 'Invalid Major', 1;

ELSE

BEGIN

IF @Description IS NOT NULL

INSERT INTO Majors (Major, Description) VALUES ( @Major, @Description);

ELSE -- Description picks up default value

INSERT INTO Majors (Major) VALUES ( @Major);

END

END

ELSE

PRINT CONVERT(varchar, @Major) + ' already existed.';

END

CREATE PROC ***sp\_UpdateMajor***

@Major Varchar(3),

@Description Varchar(30)

AS

BEGIN

IF EXISTS (SELECT \* FROM Majors WHERE Major = @Major)

BEGIN

IF @Description IS NOT NULL

UPDATE Majors SET Description = @Description WHERE Major = @Major;

ELSE -- Description picks up default value

UPDATE Majors SET Description = 'To be described' WHERE Major = @Major;

END

ELSE

PRINT CONVERT(varchar, @Major) + ' does not exist.';

END

/\*A major can be deleted if no one has that major and Department does not offer course in that major.

\*/

CREATE PROC ***sp\_DeleteMajor***

@Major Varchar(3)

AS

BEGIN

IF NOT EXISTS (SELECT \* FROM Majors WHERE Major = @Major)

PRINT CONVERT(varchar, @Major) + ' is not found.';

ELSE

BEGIN

IF (EXISTS (Select \* FROM Students WHERE Major = @Major) OR

EXISTS (Select \* FROM Courses WHERE Department = @Major))

PRINT CONVERT(varchar, @Major) + ' can not be deleted due to referential integrity';

ELSE

DELETE FROM Majors WHERE Major = @Major;

END

END

CREATE TRIGGER ***Majors\_INSERT\_UPDATE***

ON Majors

FOR INSERT, UPDATE

AS

DECLARE @Operator Varchar (6);

DECLARE @Major Varchar(3);

DECLARE @Description Varchar(30);

DECLARE @Description\_deleted Varchar(30);

DECLARE @User Varchar(50);

DECLARE @DateTime Datetime;

IF COLUMNS\_UPDATED() <> 0

BEGIN

SELECT @Major = Major FROM INSERTED;

SELECT @Description = Description FROM INSERTED;

SELECT @Description\_deleted = Description FROM DELETED;

SELECT @User = SUSER\_NAME();

SELECT @DateTime = SYSDATETIME();

BEGIN

IF EXISTS (SELECT \* FROM deleted) -- updated cols + old rows means action=update

Begin

If (@Description != @Description\_deleted)

begin

SET @Operator = 'UPDATE';

INSERT INTO Majors\_Audit (Major, Description, [User], Operator, [DateTime]) VALUES (@Major, @Description\_deleted, CURRENT\_USER, @Operator, SYSDATETIME()) ;

end

Else

PRINT 'Update nothing on Majors';

End

ELSE

Begin

SET @Operator = 'INSERT'; -- updated columns and nothing deleted means action=insert

INSERT INTO Majors\_Audit (Major, Description, [User], Operator, [DateTime]) VALUES (@Major, @Description, CURRENT\_USER, @Operator, SYSDATETIME()) ;

End

PRINT @User + ' ' + @Operator + 's Major on ' + Convert (Varchar, @DateTime, 120);

END

END

/\* **Table Students** \*/

CREATE PROC ***sp\_InsertStudent***

@StudentID Varchar(3),

@Name Varchar(10),

@Standing Tinyint,

@Major Varchar(3),

@gpa Decimal (2,1),

@major\_gpa Decimal(2,1)

AS

BEGIN

IF NOT EXISTS (SELECT \* FROM Students WHERE StudentID = @StudentID)

BEGIN

IF @Name IS NULL

THROW 50001, 'Invalid Name', 1;

ELSE

BEGIN

IF EXISTS (SELECT \* FROM Majors WHERE Major = @Major) OR @Major IS NULL

INSERT INTO Students (StudentID, Name, Standing, Major, gpa, major\_gpa)

VALUES (@StudentID, @Name, @Standing, @Major, @gpa, @major\_gpa);

ELSE

THROW 50001, 'Invalid Major', 1;

END

END

ELSE

PRINT 'Student ID: ' + CONVERT(varchar, @StudentID) + ' already existed.';

END

CREATE PROC ***sp\_UpdateStudent***

@StudentID Varchar(3),

@Name Varchar(10),

@Standing Tinyint,

@Major Varchar(3),

@gpa Decimal (2,1),

@major\_gpa Decimal(2,1)

AS

BEGIN

IF EXISTS (SELECT \* FROM Students WHERE StudentID = @StudentID)

BEGIN

IF @Name IS NOT NULL

IF EXISTS (SELECT \* FROM Majors WHERE Major = @Major) OR @Major IS NULL

UPDATE Students SET

Name = @Name, Standing = @Standing, Major = @Major, gpa = @gpa, major\_gpa = @major\_gpa WHERE StudentID = @StudentID;

ELSE

THROW 50001, 'Invalid Major', 1;

ELSE

PRINT 'Student Name ' + CONVERT(varchar, @StudentID) + ' can not be null';

END

ELSE

PRINT 'Student ID ' + CONVERT(varchar, @StudentID) + ' does not exist.';

END

/\* A Student can be deleted if he or she has not enroll and has not been in waitlist \*/

CREATE PROC ***sp\_DeleteStudent***

@StudentID Varchar(3)

AS

BEGIN

IF NOT EXISTS (SELECT \* FROM Students WHERE StudentID = @StudentID)

PRINT CONVERT(Varchar, @StudentID) + ' is not found.';

ELSE

BEGIN

IF (EXISTS (Select \* FROM Enrollments WHERE StudentID = @StudentID) OR

EXISTS (Select \* FROM Waitlist WHERE StudentID = @StudentID))

PRINT 'StudentID ' + CONVERT(varchar, @StudentID) + ' can not be deleted due to referential integrity (this student has enrolled or has been in waitlist';

ELSE

DELETE FROM Students WHERE StudentID = @StudentID;

END

END

CREATE TRIGGER ***Students\_INSERT\_UPDATE***

ON Students

FOR INSERT, UPDATE

AS

DECLARE @StudentID Varchar(3);

DECLARE @Name Varchar(10);

DECLARE @Standing Tinyint;

DECLARE @Major Varchar(3);

DECLARE @gpa Decimal(2,1);

DECLARE @major\_gpa Decimal (2,1);

DECLARE @Operator Varchar (6);

DECLARE @User Varchar(50);

DECLARE @DateTime Datetime;

DECLARE @Name\_Deleted Varchar(10);

DECLARE @Standing\_Deleted Tinyint;

DECLARE @Major\_Deleted Varchar(3);

DECLARE @gpa\_Deleted Decimal(2,1);

DECLARE @major\_gpa\_Deleted Decimal (2,1);

IF COLUMNS\_UPDATED() <> 0

BEGIN

SELECT @StudentID = StudentID FROM INSERTED;

SELECT @Name = Name FROM INSERTED;

SELECT @Standing = Standing FROM INSERTED;

SELECT @Major = Major FROM INSERTED;

SELECT @gpa = gpa FROM INSERTED;

SELECT @major\_gpa = @major\_gpa FROM INSERTED;

SELECT @User = SUSER\_NAME();

SELECT @DateTime = SYSDATETIME();

SELECT @Name\_Deleted = Name FROM DELETED;

SELECT @Standing\_Deleted = Standing FROM DELETED;

SELECT @Major\_Deleted = Major FROM DELETED;

SELECT @gpa\_Deleted = gpa FROM DELETED;

SELECT @major\_gpa\_Deleted = @major\_gpa FROM DELETED;

IF EXISTS (SELECT \* FROM deleted)

Begin

SET @Operator = 'UPDATE';

If (@Name !=@Name\_Deleted OR @Standing !=@Standing\_Deleted OR @Major != @Major\_Deleted OR @gpa != @gpa\_Deleted OR @major\_gpa !=@major\_gpa\_Deleted)

INSERT INTO Students\_Audit (StudentID, Name, Standing, Major, gpa, major\_gpa, [User], Operator, [DateTime])

VALUES (@StudentID, @Name\_Deleted, @Standing\_Deleted, @Major\_Deleted, @gpa\_Deleted, @major\_gpa\_Deleted, @User, @Operator, @DateTime);

Else

PRINT 'Update nothing on Students';

End

ELSE

Begin

SET @Operator = 'INSERT';

INSERT INTO Students\_Audit (StudentID, Name, Standing, Major, gpa, major\_gpa, [User], Operator, [DateTime])

VALUES (@StudentID, @Name, @Standing, @Major, @gpa, @major\_gpa, @User, @Operator, @DateTime);

End

PRINT @User + ' ' + @Operator + 'S Student on ' + Convert (Varchar, @DateTime, 120);

END

/\* **Table Courses** \*/

CREATE PROC **sp\_InsertCourse**

@Department Varchar(3),

@CourseNumber Varchar(3),

@Title Varchar(30),

@Hours Tinyint,

@Standing Tinyint

AS

BEGIN

IF NOT EXISTS (SELECT \* FROM Courses WHERE Department = @Department AND CourseNumber= @CourseNumber)

BEGIN

IF @Department IS NULL OR @CourseNumber IS NULL OR @Title IS NULL OR @Hours IS NULL OR @Standing IS NULL

THROW 50001, 'All field value must not be null', 1;

ELSE

BEGIN

IF EXISTS (SELECT \* FROM Majors WHERE Major = @Department)

INSERT INTO Courses (Department, CourseNumber, Title, Hours, Standing) VALUES (@Department, @CourseNumber, @Title, @Hours, @Standing);

ELSE

THROW 50001, 'Invalid Major', 1;

END

END

ELSE

PRINT 'Course: ' + CONVERT(varchar, @Department) + CONVERT(varchar, @CourseNumber)+ ' already existed.';

END

CREATE PROC ***sp\_UpdateCourse***

@Department Varchar(3),

@CourseNumber Varchar(3),

@Title Varchar(30),

@Hours Tinyint,

@Standing Tinyint

AS

BEGIN

IF EXISTS (SELECT \* FROM Courses WHERE Department = @Department AND CourseNumber= @CourseNumber)

BEGIN

IF @Department IS NULL OR @CourseNumber IS NULL OR @Title IS NULL OR @Hours IS NULL OR @Standing IS NULL

THROW 50001, 'All field value must not be null', 1;

ELSE

BEGIN

IF EXISTS (SELECT \* FROM Majors WHERE Major = @Department)

UPDATE Courses SET Title = @Title, Hours= @Hours, Standing=@Standing WHERE Department = @Department AND CourseNumber = @CourseNumber;

ELSE

THROW 50001, 'Invalid Major', 1;

END

END

ELSE

PRINT 'Course: ' + CONVERT(varchar, @Department) + CONVERT(varchar, @CourseNumber) + ' does not exist.';

END

/\*A course can be deleted if it has not been enrolled, scheduled, participated in waitlist\*/

CREATE PROC ***sp\_DeleteCourse***

@Department Varchar(3),

@CourseNumber Varchar(3)

AS

BEGIN

IF NOT EXISTS (SELECT \* FROM Courses WHERE Department = @Department AND CourseNumber= @CourseNumber)

PRINT 'Course: ' + CONVERT(varchar, @Department) + CONVERT(varchar, @CourseNumber) + ' does not exist.';

ELSE

BEGIN

IF (EXISTS (Select \* FROM ScheduledClasses WHERE Department = @Department AND CourseNumber = @CourseNumber) OR

EXISTS (SELECT \* from ScheduledClasses s, Enrollments e WHERE s.CallNumber = e.CallNumber and Department = @Department and CourseNumber=@CourseNumber) OR

EXISTS (SELECT \* from ScheduledClasses s, Waitlist w WHERE s.CallNumber = w.CallNumber and Department = @Department and CourseNumber=@CourseNumber))

PRINT 'Course: ' + CONVERT(varchar, @Department) + CONVERT(varchar, @CourseNumber) + ' can not be deleted due to referential integrity (this student has enrolled or has been in waitlist';

ELSE

DELETE FROM Courses WHERE Department = @Department AND CourseNumber= @CourseNumber;

END

END

CREATE TRIGGER ***Courses\_INSERT\_UPDATE***

ON Courses

FOR INSERT, UPDATE

AS

DECLARE @Department Varchar(3);

DECLARE @CourseNumber Varchar(3);

DECLARE @Title Varchar(30);

DECLARE @Hours Tinyint;

DECLARE @Standing Tinyint;

DECLARE @Title\_Deleted Varchar(30);

DECLARE @Hours\_Deleted Tinyint;

DECLARE @Standing\_Deleted Tinyint;

DECLARE @Operator Varchar (6);

DECLARE @User Varchar(50);

DECLARE @DateTime Datetime;

IF COLUMNS\_UPDATED() <> 0

BEGIN

SELECT @Department = Department FROM INSERTED;

SELECT @CourseNumber = CourseNumber FROM INSERTED;

SELECT @Title = Title FROM INSERTED;

SELECT @Hours = Hours FROM INSERTED;

SELECT @Standing = Standing FROM INSERTED;

SELECT @Title\_Deleted = Title FROM DELETED;

SELECT @Hours\_Deleted = Hours FROM DELETED;

SELECT @Standing\_Deleted = Standing FROM DELETED;

SELECT @User = SUSER\_NAME();

SELECT @DateTime = SYSDATETIME();

IF EXISTS (SELECT \* FROM deleted)

Begin

SET @Operator = 'UPDATE';

If (@Title != @Title\_Deleted OR @Hours != @Hours\_Deleted OR @Standing != @Standing\_Deleted)

INSERT INTO Courses\_Audit(Department, CourseNumber, Title, Hours, Standing, [User], Operator, [DateTime]) VALUES

(@Department, @CourseNumber, @Title\_Deleted, @Hours\_Deleted, @Standing\_Deleted, @User, @Operator, @DateTime);

Else

PRINT 'Update nothing on Courses';

End

ELSE

Begin

SET @Operator = 'INSERT';

INSERT INTO Courses\_Audit(Department, CourseNumber, Title, Hours, Standing, [User], Operator, [DateTime]) VALUES

(@Department, @CourseNumber, @Title, @Hours, @Standing, @User, @Operator, @DateTime);

End

PRINT @User + ' ' + @Operator + 'S a course on ' + Convert (Varchar, @DateTime, 120);

END

/\* **Table ScheduledClasses** \*/

CREATE PROC ***sp\_InsertScheduledClass***

@CallNumber Numeric (5,0),

@Year Numeric (4,0),

@Semester Varchar(3),

@Department Varchar(3),

@CourseNumber Varchar(3),

@Section tinyint,

@Capacity smallint

AS

BEGIN

IF NOT EXISTS (SELECT \* FROM ScheduledClasses WHERE CallNumber = @CallNumber)

BEGIN

IF @CallNumber IS NULL OR @Year IS NULL OR @Semester IS NULL OR @Department IS NULL OR @CourseNumber IS NULL OR @Section IS NULL OR @Capacity IS NULL

THROW 50001, 'All field value must not be null', 1;

ELSE

BEGIN

IF EXISTS (SELECT \* FROM Courses WHERE Department = @Department AND CourseNumber= @CourseNumber)

INSERT INTO ScheduledClasses( CallNumber, Year, Semester, Department, CourseNumber, Section, Capacity) VALUES (@CallNumber, @Year, @Semester, @Department, @CourseNumber, @Section, @Capacity) ;

ELSE

THROW 50001, 'Invalid Course', 1;

END

END

ELSE

PRINT 'Call Number ' + CONVERT(varchar, @CallNumber) + ' was already scheduled';

END

CREATE PROC ***sp\_UpdateScheduledClass***

@CallNumber Numeric (5,0),

@Year Numeric (4,0),

@Semester Varchar(3),

@Department Varchar(3),

@CourseNumber Varchar(3),

@Section tinyint,

@Capacity smallint

AS

BEGIN

IF EXISTS (SELECT \* FROM ScheduledClasses WHERE CallNumber = @CallNumber)

BEGIN

IF @CallNumber IS NULL OR @Year IS NULL OR @Semester IS NULL OR @Department IS NULL OR @CourseNumber IS NULL OR @Section IS NULL OR @Capacity IS NULL

THROW 50001, 'All field value must not be null', 1;

ELSE

BEGIN

IF EXISTS (SELECT \* FROM Courses WHERE Department = @Department AND CourseNumber= @CourseNumber)

UPDATE ScheduledClasses SET Year = @Year, Semester = @Semester, Department = @Department, CourseNumber = @CourseNumber, Section = @Section, Capacity = @Capacity WHERE CallNumber = @CallNumber

ELSE

THROW 50001, 'Invalid Course', 1;

END

END

ELSE

PRINT 'Call Number ' + CONVERT(varchar, @CallNumber) + ' does not exist';

END

/\*A Scheduled Class can be deleted if it has been enrolled, scheduled, or participated in waitlist

\*/

CREATE PROC ***sp\_DeleteScheduledClass***

@CallNumber Numeric

AS

BEGIN

IF NOT EXISTS (SELECT \* FROM ScheduledClasses WHERE CallNumber = @CallNumber)

PRINT 'Call Number: ' + CONVERT(varchar, @CallNumber) + ' does not exist.';

ELSE

BEGIN

IF (

EXISTS (SELECT \* from Enrollments WHERE CallNumber = @CallNumber) OR

EXISTS (SELECT \* from Waitlist WHERE CallNumber = @CallNumber))

PRINT 'Call Number: ' + CONVERT(varchar, @CallNumber) + ' can not be deleted due to referential integrity (this student has enrolled or has been in waitlist';

ELSE

DELETE FROM ScheduledClasses WHERE CallNumber = @CallNumber;

END

END

CREATE TRIGGER ***ScheduledClasses\_INSERT\_UPDATE***

ON ScheduledClasses

FOR INSERT, UPDATE

AS

DECLARE @CallNumber Numeric (5,0);

DECLARE @Year Numeric (4,0);

DECLARE @Semester Varchar(3);

DECLARE @Department Varchar(3);

DECLARE @CourseNumber Varchar(3);

DECLARE @Section tinyint;

DECLARE @Capacity smallint;

DECLARE @Year\_Deleted Numeric (4,0);

DECLARE @Semester\_Deleted Varchar(3);

DECLARE @Department\_Deleted Varchar(3);

DECLARE @CourseNumber\_Deleted Varchar(3);

DECLARE @Section\_Deleted tinyint;

DECLARE @Capacity\_Deleted smallint;

DECLARE @Operator Varchar (6);

DECLARE @User Varchar(50);

DECLARE @DateTime Datetime;

IF COLUMNS\_UPDATED() <> 0

BEGIN

SELECT @CallNumber = CallNumber FROM INSERTED;

SELECT @Year = Year FROM INSERTED;

SELECT @Semester = Semester FROM INSERTED;

SELECT @Department = Department FROM INSERTED;

SELECT @CourseNumber = CourseNumber FROM INSERTED;

SELECT @Section = Section FROM INSERTED;

SELECT @Capacity = Capacity FROM INSERTED;

SELECT @Year\_Deleted = Year FROM DELETED;

SELECT @Semester\_Deleted = Semester FROM DELETED;

SELECT @Department\_Deleted = Department FROM DELETED;

SELECT @CourseNumber\_Deleted = CourseNumber FROM DELETED;

SELECT @Section\_Deleted = Section FROM DELETED;

SELECT @Capacity\_Deleted = Capacity FROM DELETED;

SELECT @User = SUSER\_NAME();

SELECT @DateTime = SYSDATETIME();

IF EXISTS (SELECT \* FROM deleted)

Begin

SET @Operator = 'UPDATE';

If (@Year != @Year\_Deleted OR @Semester != @Semester\_Deleted OR @Department != @Department\_Deleted OR @CourseNumber != @CourseNumber\_Deleted OR @Section != @Section\_Deleted OR @Capacity != @Capacity\_Deleted)

INSERT INTO ScheduledClasses\_Audit(CallNumber, Year, Semester, Department, CourseNumber, Section, Capacity, [User], Operator, [DateTime])

VALUES (@CallNumber, @Year\_Deleted, @Semester\_Deleted, @Department\_Deleted, @CourseNumber\_Deleted, @Section\_Deleted, @Capacity\_Deleted, @User, @Operator, @DateTime);

Else

PRINT 'Update nothing on ScheduledClasses';

End

ELSE

Begin

SET @Operator = 'INSERT';

INSERT INTO ScheduledClasses\_Audit(CallNumber, Year, Semester, Department, CourseNumber, Section, Capacity, [User], Operator, [DateTime])

VALUES (@CallNumber, @Year, @Semester, @Department, @CourseNumber, @Section, @Capacity, @User, @Operator, @DateTime);

End

PRINT @User + ' ' + @Operator + 'S a ScheduledClass on ' + Convert (Varchar, @DateTime, 120);

END

/\* **Table Prerequisites** \*/

CREATE PROC ***sp\_InsertPrerequisite***

@Department Varchar(3),

@CourseNumber Varchar(3),

@prereq\_Department Varchar(3),

@prereq\_CourseNumber Varchar(3)

AS

BEGIN

IF NOT EXISTS (SELECT \* FROM Prerequisites WHERE Department = @Department AND CourseNumber= @CourseNumber AND prereq\_Department = @prereq\_Department AND prereq\_CourseNumber = @prereq\_CourseNumber)

BEGIN

IF EXISTS (SELECT \* FROM Courses WHERE Department = @Department AND CourseNumber= @CourseNumber)

BEGIN

IF EXISTS (SELECT \* FROM Courses WHERE Department = @prereq\_Department AND CourseNumber= @prereq\_CourseNumber)

BEGIN

IF (CONVERT(INT, @CourseNumber) > CONVERT(INT, @prereq\_CourseNumber))

INSERT INTO Prerequisites(Department, CourseNumber, prereq\_Department, prereq\_CourseNumber) VALUES (@Department, @CourseNumber, @prereq\_Department, @prereq\_CourseNumber);

ELSE

PRINT '@CourseNumber must greater than @prereq\_CourseNumber';

END

ELSE

THROW 50001, 'Invalid @prereq\_Department and @prereq\_CourseNumber', 1;

END

ELSE

THROW 50001, 'Invalid @Department and @CourseNumber', 1;

END

ELSE

PRINT 'This Prerequisite already existed.';

END

CREATE PROC ***sp\_DeletePrerequisite***

@Department Varchar(3),

@CourseNumber Varchar(3),

@prereq\_Department Varchar(3),

@prereq\_CourseNumber Varchar(3)

AS

BEGIN

IF NOT EXISTS (SELECT \* FROM Prerequisites WHERE Department = @Department AND CourseNumber= @CourseNumber AND prereq\_Department = @prereq\_Department AND prereq\_CourseNumber = @prereq\_CourseNumber)

PRINT 'This Prerequisite is not found.';

ELSE

DELETE FROM Prerequisites WHERE Department = @Department AND CourseNumber= @CourseNumber AND prereq\_Department = @prereq\_Department AND prereq\_CourseNumber = @prereq\_CourseNumber;

END

CREATE TRIGGER ***Prerequisites\_INSERT\_UPDATE***

ON Prerequisites

FOR INSERT, UPDATE

AS

DECLARE @Department Varchar(3);

DECLARE @CourseNumber Varchar(3);

DECLARE @prereq\_Department Varchar(3);

DECLARE @prereq\_CourseNumber Varchar(3);

DECLARE @Operator Varchar (6);

DECLARE @User Varchar(50);

DECLARE @DateTime Datetime;

IF COLUMNS\_UPDATED() <> 0

BEGIN

SELECT @Department = Department FROM INSERTED;

SELECT @CourseNumber = CourseNumber FROM INSERTED;

SELECT @prereq\_Department = prereq\_Department FROM INSERTED;

SELECT @prereq\_CourseNumber = prereq\_CourseNumber FROM INSERTED;

SELECT @User = SUSER\_NAME();

SELECT @DateTime = SYSDATETIME();

IF EXISTS (SELECT \* FROM deleted)

Begin

-- SET @Operator = 'UPDATE';

PRINT @User + ' tries to update a prerequisite on ' + Convert (Varchar, @DateTime, 120) + '; however, a prerequisite cannot be updated.';

End

ELSE

Begin

SET @Operator = 'INSERT';

INSERT INTO Prerequisites\_Audit(Department, CourseNumber, prereq\_Department, prereq\_CourseNumber, [User], Operator, [DateTime]) VALUES

(@Department, @CourseNumber, @prereq\_Department, @prereq\_CourseNumber, @User, @Operator, @DateTime);

PRINT @User + ' ' + @Operator + 'S a prerequisite on ' + Convert (Varchar, @DateTime, 120);

End

END

/\* **Table Enrollments** \*/

CREATE PROC ***sp\_InsertEnrollment***

@StudentID Varchar(3),

@CallNumber Numeric (5,0)

AS

BEGIN

IF NOT EXISTS (SELECT \* FROM Enrollments WHERE StudentID = @StudentID AND CallNumber = @CallNumber)

BEGIN

IF @CallNumber IS NULL OR @StudentID IS NULL

THROW 50001, 'All field value must not be null', 1;

ELSE

BEGIN

IF EXISTS (SELECT \* FROM Students WHERE StudentID = @StudentID)

IF EXISTS (SELECT \* FROM ScheduledClasses WHERE CallNumber = @CallNumber)

INSERT INTO Enrollments(StudentID, CallNumber) VALUES (@StudentID, @CallNumber);

ELSE

THROW 50001, 'Invalid Scheduled Course call number', 1;

ELSE

THROW 50001, 'Invalid Student', 1;

END

END

ELSE

PRINT 'Student ' + @StudentID + 'has enrolled ' + CONVERT(varchar, @CallNumber) + ' before';

END

CREATE PROC ***sp\_UpdateEnrollment***

@StudentID Varchar(3),

@CallNumber Numeric (5,0),

@Grade Varchar(2)

AS

BEGIN

IF @CallNumber IS NULL OR @StudentID IS NULL

THROW 50001, 'All field value must not be null', 1;

ELSE

BEGIN

IF EXISTS (SELECT \* FROM Students WHERE StudentID = @StudentID)

IF EXISTS (SELECT \* FROM ScheduledClasses WHERE CallNumber = @CallNumber)

IF EXISTS (SELECT \* FROM Enrollments WHERE StudentID = @StudentID AND CallNumber = @CallNumber)

UPDATE Enrollments SET Grade = @Grade WHERE StudentID = @StudentID AND CallNumber = @CallNumber;

ELSE

PRINT 'Student ' + @StudentID + ' has not enrolled ' + CONVERT(varchar, @CallNumber) + ' before';

ELSE

THROW 50001, 'Invalid Scheduled Course call number', 1;

ELSE

THROW 50001, 'Invalid Student', 1;

END

END

CREATE PROC ***sp\_DeleteEnrollment***

@StudentID Varchar(3),

@CallNumber Numeric (5,0)

AS

BEGIN

IF EXISTS (SELECT \* FROM Students WHERE StudentID = @StudentID)

IF EXISTS (SELECT \* FROM ScheduledClasses WHERE CallNumber = @CallNumber)

IF EXISTS (SELECT \* FROM Enrollments WHERE StudentID = @StudentID AND CallNumber = @CallNumber)

IF EXISTS (SELECT \* FROM Enrollments WHERE StudentID = @StudentID AND CallNumber = @CallNumber AND Grade IS NULL)

DELETE FROM Enrollments WHERE StudentID = @StudentID AND CallNumber = @CallNumber;

ELSE

PRINT 'Can not delete this enrollment because grade was assigned';

ELSE

PRINT 'Student ' + @StudentID + ' has not enrolled ' + CONVERT(varchar, @CallNumber) + ' before';

ELSE

THROW 50001, 'Invalid Scheduled Course call number', 1;

ELSE

THROW 50001, 'Invalid Student', 1;

END

CREATE TRIGGER ***Enrollments\_INSERT\_UPDATE***

ON Enrollments

FOR INSERT, UPDATE

AS

DECLARE @StudentID Varchar(3);

DECLARE @CallNumber Numeric (5,0);

DECLARE @Grade Varchar(2);

DECLARE @Grade\_Deleted Varchar(2);

DECLARE @Operator Varchar (6);

DECLARE @User Varchar(50);

DECLARE @DateTime Datetime;

IF COLUMNS\_UPDATED() <> 0

BEGIN

SELECT @StudentID = StudentID FROM INSERTED;

SELECT @CallNumber = CallNumber FROM INSERTED;

SELECT @Grade = Grade FROM INSERTED;

SELECT @Grade\_Deleted = Grade FROM DELETED;

SELECT @User = SUSER\_NAME();

SELECT @DateTime = SYSDATETIME();

IF EXISTS (SELECT \* FROM deleted)

Begin

SET @Operator = 'UPDATE';

If (@Grade != @Grade\_Deleted OR (@Grade IS NOT NULL AND @Grade\_Deleted IS NULL))

INSERT INTO Enrollments\_Audit(StudentID, CallNumber, Grade, [User], Operator, [DateTime]) VALUES (@StudentID, @CallNumber, @Grade\_Deleted, @User, @Operator, @DateTime);

Else

PRINT 'Update nothing on Enrollments';

End

ELSE

Begin

SET @Operator = 'INSERT';

INSERT INTO Enrollments\_Audit(StudentID, CallNumber, Grade, [User], Operator, [DateTime]) VALUES (@StudentID, @CallNumber, @Grade, @User, @Operator, @DateTime);

End

PRINT @User + ' ' + @Operator + 'S an enrollment record on ' + Convert (Varchar, @DateTime, 120);

END

/\* **Table Waitlist** \*/

CREATE PROC ***sp\_InsertWaitlistRecord***

@StudentID Varchar(3),

@CallNumber Numeric (5,0)

AS

BEGIN

IF NOT EXISTS (SELECT \* FROM Waitlist WHERE StudentID = @StudentID AND CallNumber = @CallNumber)

BEGIN

IF @CallNumber IS NULL OR @StudentID IS NULL

THROW 50001, 'All field value must not be null', 1;

ELSE

BEGIN

IF EXISTS (SELECT \* FROM Students WHERE StudentID = @StudentID)

IF EXISTS (SELECT \* FROM ScheduledClasses WHERE CallNumber = @CallNumber)

INSERT INTO Waitlist(StudentID, CallNumber) VALUES (@StudentID, @CallNumber);

ELSE

THROW 50001, 'Invalid Scheduled Course call number', 1;

ELSE

THROW 50001, 'Invalid Student', 1;

END

END

ELSE

PRINT 'Student ' + @StudentID + 'has been in Waitlist for course ' + CONVERT(varchar, @CallNumber) + ' before';

END

CREATE PROC ***sp\_UpdateWaitlistRecord***

@StudentID Varchar(3),

@CallNumber Numeric (5,0)

AS

BEGIN

IF @CallNumber IS NULL OR @StudentID IS NULL

THROW 50001, 'All field value must not be null', 1;

ELSE

BEGIN

IF EXISTS (SELECT \* FROM Students WHERE StudentID = @StudentID)

IF EXISTS (SELECT \* FROM ScheduledClasses WHERE CallNumber = @CallNumber)

IF EXISTS (SELECT \* FROM Waitlist WHERE StudentID = @StudentID AND CallNumber = @CallNumber)

UPDATE Waitlist SET RequestTime = GETDATE() WHERE StudentID = @StudentID AND CallNumber = @CallNumber;

ELSE

PRINT 'Student ' + @StudentID + ' has not been in waitlist for course ' + CONVERT(varchar, @CallNumber) + ' before';

ELSE

THROW 50001, 'Invalid Scheduled Course call number', 1;

ELSE

THROW 50001, 'Invalid Student', 1;

END

END

CREATE PROC ***sp\_DeleteWaitlistRecord***

@StudentID Varchar(3),

@CallNumber Numeric (5,0)

AS

BEGIN

IF EXISTS (SELECT \* FROM Students WHERE StudentID = @StudentID)

IF EXISTS (SELECT \* FROM ScheduledClasses WHERE CallNumber = @CallNumber)

IF EXISTS (SELECT \* FROM Waitlist WHERE StudentID = @StudentID AND CallNumber = @CallNumber)

DELETE FROM Waitlist WHERE StudentID = @StudentID AND CallNumber = @CallNumber;

ELSE

PRINT 'Student ' + @StudentID + ' has not been in waitlist for course ' + CONVERT(varchar, @CallNumber) + ' before';

ELSE

THROW 50001, 'Invalid Scheduled Course call number', 1;

ELSE

THROW 50001, 'Invalid Student', 1;

END

CREATE TRIGGER ***Waitlist\_INSERT\_UPDATE***

ON Waitlist

FOR INSERT, UPDATE

AS

DECLARE @StudentID Varchar(3)

DECLARE @CallNumber Numeric (5,0);

DECLARE @RequestTime DateTime;

DECLARE @RequestTime\_Deleted DateTime;

DECLARE @Operator Varchar (6);

DECLARE @User Varchar(50);

DECLARE @DateTime Datetime;

IF COLUMNS\_UPDATED() <> 0

BEGIN

SELECT @StudentID = StudentID FROM INSERTED;

SELECT @CallNumber = CallNumber FROM INSERTED;

SELECT @RequestTime = RequestTime FROM INSERTED;

SELECT @RequestTime\_Deleted = RequestTime FROM DELETED;

SELECT @User = SUSER\_NAME();

SELECT @DateTime = SYSDATETIME();

IF EXISTS (SELECT \* FROM deleted)

Begin

SET @Operator = 'UPDATE';

If (@RequestTime != @RequestTime\_Deleted)

INSERT INTO Waitlist\_Audit (StudentID, CallNumber, RequestTime, [User], Operator, [DateTime]) VALUES (@StudentID, @CallNumber, @RequestTime\_Deleted, @User, @Operator, @DateTime);

Else

PRINT 'Update nothing on Waitlist';

End

ELSE

Begin

SET @Operator = 'INSERT';

INSERT INTO Waitlist\_Audit (StudentID, CallNumber, RequestTime, [User], Operator, [DateTime]) VALUES (@StudentID, @CallNumber, @RequestTime, @User, @Operator, @DateTime);

End

PRINT @User + ' ' + @Operator + 'S a waitlist record on ' + Convert (Varchar, @DateTime, 120);

END

USER-DEFINED FUNCTION

CREATE FUNCTION fnCountStudentHasAtLeastGPAGroupByMajors

(@gpa Decimal(2,1)) RETURNS TABLE

RETURN

(SELECT Majors.Description AS MAJOR, COUNT (StudentID) as [NUMBER OF STUDENTS] FROM Students INNER JOIN Majors

ON Students.Major = Majors.Major

WHERE gpa >= @gpa

GROUP BY (Majors.Description) HAVING (Majors.Description IS NOT NULL AND Majors.Description != 'To be described')); -- declared major students only

-- TESTING

SELECT \* FROM fnCountStudentHasAtLeastGPAGroupByMajors(2.5);

-- SCREEN CAPTURED

