REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
CHEMICAL AND METALLURGICAL FACULTY
DEPARTMENT OF MATHEMATICAL ENGINEERING

SQL APPLICATIONS

DATABASE PROJECT LIKE USIS

INSTRUCTOR
Prof. Dr. Aydın SECER

15058009
Sümeyye SEREN

DEADLINE:25.05.2020

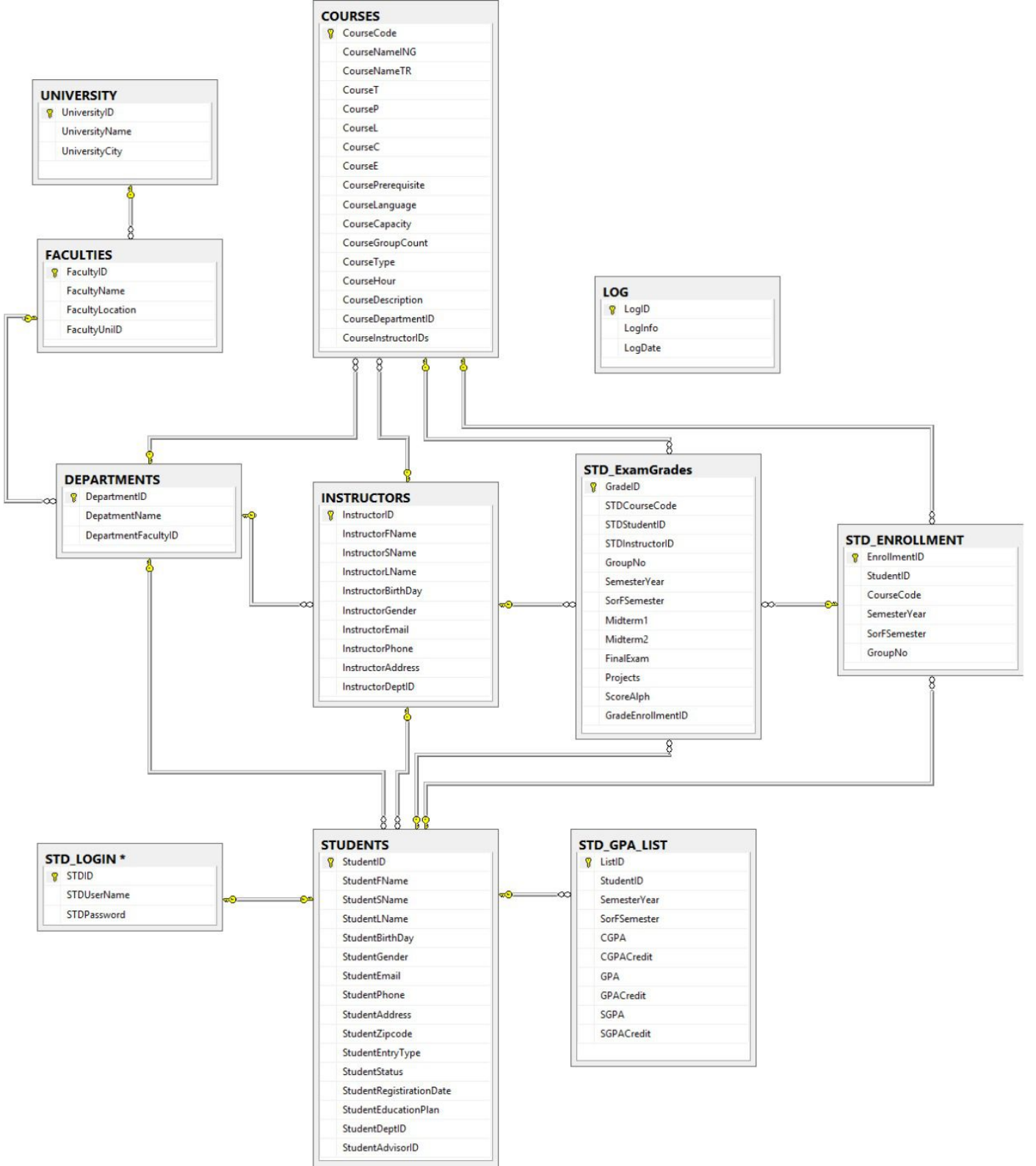
SCHOOL Database

Project

08.05.2020

Assignment: Prepare a simple School database project looks like “USIS”. Please use all topics in SQL that we learned during the semester. The due date is given on onlinekampus2.yildiz.edu.tr.

Prof. Dr. Aydın SEÇER


COURSES
CourseCode
CourseNameING
CourseNameTR
CourseT
CourseP
CourseL
CourseC
CourseE
CoursePrerequisite
CourseLanguage
CourseCapacity
CourseGroupCount
CourseType
CourseHour
CourseDescription
CourseDepartmentID
CourseInstructorIDs
UNIVERSITY
UniversityID
UniversityName
UniversityCity
FACULTIES
FacultyID
FacultyName
FacultyLocation
FacultyUnilD
LOG
LogID
LogInfo
LogDate
DEPARTMENTS
DepartmentID
DepatmentName
DepartmentFacultyID
INSTRUCTORS
InstructorID
InstructorFName
InstructorSName
InstructorLName
InstructorBirthDay
InstructorGender
InstructorEmail
InstructorPhone
InstructorAddress
InstructorDeptID
STD_ExamGrades
GradeID
STDCourseCode
STDStudentID
STDInstructorID
GroupNo
SemesterYear
SorFSemester
Midterm1
Midterm2
FinalExam
Projects
ScoreAlph
GradeEnrollmentID
STD_ENROLLMENT
EnrollmentID
StudentID
CourseCode
SemesterYear
SorFSemester
GroupNo
STD_LOGIN *
STDID
STDUserName
STDPassword
STUDENTS
StudentID
StudentFName
StudentSName
StudentLName
StudentBirthDay
StudentGender
StudentEmail
StudentPhone
StudentAddress
StudentZipcode
StudentEntryType
StudentStatus
StudentRegistirationDate
StudentEducationPlan
StudentDeptID
StudentAdvisorID
STD_GPA_LIST
ListID
StudentID
SemesterYear
SorFSemester
CGPA
CGPACredit
GPA
GPACredit
SGPA
SGPACredit

USIS.sql

home > sumeyye.seren > Documents > sumeyye.seren > SQL_USISPROJECT > USIS.sql

```sql
--DATABASE CREATION EXAMPLES
CREATE DATABASE UsisCopy ON PRIMARY(NAME = 'UsisCopy_data', FILENAME = 'C:\PATH\OF\UsisCopy_data.mdf')
LOG ON(NAME = 'UsisCopy_log', FILENAME = 'C:\PATH\OF\LOGS\UsisCopy_log.ldf')

CREATE DATABASE UsisCopy
BACKUP DATABASE UsisCopy TO DISK = 'C:\WHERE\UsisCopy.bak'

DROP DATABASE UsisCopy

RESTORE DATABASE UsisCopy FROM DISK = 'C:\PATHs\UsisCopy.bak'

--DATABASE CREATING
CREATE DATABASE USIS;
USE USIS;

--TABLE CREATING
CREATE TABLE UNIVERSITY
(
      UniversityID nvarchar(50) not null primary key,
      UniversityName varchar(50) not null,
      UniversityCity varchar(50) not null

)

CREATE TABLE FACULTIES
(
      FacultyID nvarchar(50) not null primary key ,
      FacultyName varchar(50) not null,
      FacultyLocation nvarchar(100),
      FacultyUniID nvarchar(50) not null
)

CREATE TABLE DEPARTMENTS
(
      DepartmentID nvarchar(50) not null primary key,
      DepatmentName varchar(50) not null,
      DepartmentFacultyID nvarchar(50) not null
)

CREATE TABLE INSTRUCTORS
(
      InstructorID int not null primary key identity(1,1),
      InstructorFName varchar(50) not null,
      InstructorSName varchar(50),
      InstructorLName varchar(50) not null,
      InstructorBirthDay date,
      InstructorGender char(1),
      InstructorEmail nvarchar(50) not null,
      InstructorPhone nvarchar(35),
      InstructorAddress nvarchar(100),
      InstructorDeptID nvarchar(50) foreign key references DEPARTMENTS(DepartmentID)
)
```

USIS.sql

home > sumeyye.seren > Documents > sumeyye.seren > SQL_USISPROJECT > USIS.sql

```
52  )
53
54  CREATE TABLE COURSES
55  (
56      CourseCode nvarchar(50) not null primary key,
57      CourseNameING varchar(50),
58      CourseNameTR varchar(50),
59      CourseT int,
60      CourseP int,
61      CourseL int,
62      CourseC int,
63      CourseE int,
64      CoursePrerequisite nvarchar(50),
65      CourseLanguage varchar(20),
66      CourseCapacity int,
67      CourseGroupCount int,
68      CourseType char(15),
69      CourseHour int,
70      CourseDescription text,
71      CourseDepartmentID nvarchar(50) not null,
72      CourseInstructorIDs int foreign key references INSTRUCTORS(InstructorID)
73
74  )
75
76
77  CREATE TABLE STUDENTS
78  (
79      StudentID nvarchar(9) not null primary key,
80      StudentFName varchar(50) not null,
81      StudentSName varchar(50),
82      StudentLName varchar(50) not null,
83      StudentBirthDay date,
84      StudentGender char(1),
85      StudentEmail nvarchar(50) not null,
86      StudentPhone nvarchar(35),
87      StudentAddress nvarchar(100),
88      StudentZipcode int,
89      StudentEntryType varchar(15) not null,
90      StudentStatus varchar(10) not null,
91      StudentRegistirationDate int not null,
92      StudentEducationPlan int not null,
93      StudentDeptID nvarchar(50) foreign key references DEPARTMENTS(DepartmentID),
94      StudentAdvisorID int foreign key references INSTRUCTORS(InstructorID)
95  )
96
97  CREATE TABLE STD_ENROLLMENT
98  (
99      EnrollmentID int not null primary key identity(1,1),
100     StudentID nvarchar(9) not null foreign key references STUDENTS(StudentID),
101     CourseCode nvarchar(50) foreign key references COURSES(CourseCode),
102     SemesterYear int not null,
103     SorFSemester char(6) not null,
```

USIS.sql

home > sumeyye.seren > Documents > sumeyye.seren > SQL_USISPROJECT > USIS.sql

```
        StudentID nvarchar(9) not null foreign key references STUDENTS(StudentID),
        CourseCode nvarchar(50) foreign key references COURSES(CourseCode),
        SemesterYear int not null,
        SorFSemester char(6) not null,
        GroupNo int
)

CREATE TABLE STD_GPA_LIST
(
        ListID int not null primary key identity(1,1),
        StudentID nvarchar(9) foreign key references STUDENTS(StudentID),
        SemesterYear int not null,
        SorFSemester char(6) not null,
        CGPA float,
        CGPACredit int,
        GPA float,
        GPACredit int,
        SGPA float,
        SGPACredit int
)

CREATE TABLE STD_ExamGrades
(
        GradeID int not null primary key identity(1,1),
        STDCourseCode nvarchar(50) foreign key references COURSES(CourseCode),
        STDStudentID nvarchar(9) foreign key references STUDENTS(StudentID),
        STDInstructorID int foreign key references INSTRUCTORS(InstructorID),
        GroupNo int,
        SemesterYear int not null,
        SorFSemester char(6) not null,
        Midterm1 float,
        Midterm2 float,
        FinalExam float,
        Projects float,
        ScoreAlph varchar(2),
        GradeEnrollmentID int foreign key references STD_ENROLLMENT(EnrollmentID)
)

CREATE TABLE STD_LOGIN
(
        STDID nvarchar(9) primary key foreign key references STUDENTS(StudentID),
        STDUserName nvarchar(9) not null,
        STDPassword nvarchar(10) not null
)

CREATE TABLE LOG
(
        LogID int not null primary key identity(1,1),
        LogInfo varchar(max),
        LogDate date
)

--PLAYING WITH TABLES
```

USIS.sql

home > sumeyye.seren > Documents > sumeyye.seren > SQL_USISPROJECT > USIS.sql

```sql
--PLAYING WITH TABLES
CREATE TABLE FOO
(
   FirstC int primary key not null,
   SecondC  date unique,
);

SELECT * INTO FOO1 FROM FOO;

CREATE TABLE FOO2
(
   One float primary key not null,
   Two float,
);

ALTER TABLE FOO
ADD ThirdC nvarchar(3);

ALTER TABLE FOO2
ALTER COLUMN Two int;

ALTER TABLE FOO1
DROP COLUMN FirstC;

--CONSTRAINT ADDING and DELETING
ALTER TABLE FOO
ADD CONSTRAINT UN_CLMN UNIQUE (FirstC,SecondC);

ALTER TABLE FOO
DROP CONSTRAINT UN_CLMN;

--PLAYGROUND CLEANING
TRUNCATE TABLE FOO;

DROP TABLE FOO1;
DROP TABLE FOO2;
DROP TABLE FOO;

 -- FOREIGN KEY ASSIGNING with CONSTRAINTS
ALTER TABLE FACULTIES ADD CONSTRAINT FK_FacultyuniID
FOREIGN KEY (FacultyUniID) REFERENCES UNIVERSITY(UniversityID)

ALTER TABLE DEPARTMENTS ADD CONSTRAINT FK_DepartmentFacultyID
FOREIGN KEY (DepartmentFacultyID) REFERENCES FACULTIES(FacultyID)

ALTER TABLE COURSES ADD CONSTRAINT FK_CourseDepartmentID
FOREIGN KEY (CourseDepartmentID) REFERENCES DEPARTMENTS(DepartmentID)


--MANUEL INSERTING VALUES
INSERT INTO UNIVERSITY(UniversityID, UniversityName, UniversityCity)
VALUES('TRISTANBU07', 'YILDIZ TECHNICAL UNIVERSITY','ISTANBUL')
```

USIS.sql

home > sumeyye.seren > Documents > sumeyye.seren > SQL_USISPROJECT > USIS.sql

```sql
--MANUEL INSERTING VALUES
INSERT INTO UNIVERSITY(UniversityID, UniversityName, UniversityCity)
VALUES('TRISTANBU07', 'YILDIZ TECHNICAL UNIVERSITY','ISTANBUL')

INSERT INTO FACULTIES(FacultyID, FacultyName, FacultyLocation, FacultyUniID)
VALUES('KMB','Chemical and Metallurgical Engineering','DAVUTPASA','TRISTANBU07')

INSERT INTO DEPARTMENTS(DepartmentID, DepatmentName,DepartmentFacultyID)
VALUES('MTM','MATHEMATICAL ENGINEERING', 'KMB')

INSERT INTO INSTRUCTORS(InstructorFName, InstructorLName, InstructorEmail, InstructorGender, InstructorPhone, InstructorDeptID)
VALUES('AYDIN','SECER','asecer@yildiz.edu.tr','M','+9005555555555', 'MTM')

INSERT INTO COURSES(CourseCode,CourseNameTR,CourseNameING,CourseLanguage,CourseGroupCount,CourseHour,CourseCapacity,CourseDepartmentID,CourseT,CourseP,CourseL,CourseC,CourseE,CourseType)
VALUES('MTM4662','SQL UYGULAMALARI', 'MSSQL DATABASE', 'ENGLISH',1,3,100,'MTM',3,0,0,3,6,'DEPARTMENT')

INSERT INTO STUDENTS(StudentID,StudentFName,StudentLName,StudentEmail,StudentGender,StudentPhone,StudentBirthDay,StudentEducationPlan,StudentStatus,StudentDeptID,StudentEntryType,StudentRegistira
VALUES('15058009','SUMEYYE','SEREN','l5815009@std.yildiz.edu.tr','F','+9005555555555','1997-01-24',2018,'ACTIVE','MTM','OSS',2015)

INSERT INTO STD_ENROLLMENT(StudentID,CourseCode,SemesterYear,SorFSemester,GroupNo)
VALUES(15058009,'MTM4662',2020,'SPRING',2)

INSERT INTO STD_ExamGrades(STDCourseCode,STDStudentID,GroupNo,SemesterYear,SorFSemester,Midterm1,Midterm2,FinalExam,Projects,ScoreAlph)
VALUES('MTM4662',15058009,2,2020,'SPRING',100,100,100,100,'AA')

INSERT INTO STD_GPA_LIST(StudentID,SemesterYear,SorFSemester,GPA,GPACredit,SGPA,SGPACredit)
VALUES(15058009,2020,'SPRING',4.0,165,4.0,32)

INSERT INTO STD_LOGIN(STDID,STDUserName,STDPassword)
VALUES(15058009,'l5815009','aJskeux*')

GO
--CHECK VALUES OF TABLES
SELECT * FROM UNIVERSITY

SELECT * FROM FACULTIES

SELECT * FROM DEPARTMENTS

SELECT * FROM INSTRUCTORS

SELECT * FROM STUDENTS

SELECT * FROM COURSES

SELECT * FROM STD_ENROLLMENT

SELECT * FROM STD_ExamGrades

SELECT * FROM STD_GPA_LIST

SELECT * FROM STD_LOGIN
```

```sql
--UPDATE TABLE
ALTER TABLE STUDENTS
ADD StudentPassword nvarchar(8);

--CHECK
SELECT * FROM STUDENTS
--UPDATE ROWS

UPDATE STUDENTS SET StudentPassword = 'JKDO4*JS' WHERE StudentID = 12345
UPDATE STUDENTS SET StudentPassword = 'QWERTY2*' WHERE StudentID = 15058000
UPDATE STUDENTS SET StudentPassword = 'QWERTY1*' WHERE StudentID = 15058009

ALTER TABLE STUDENTS
DROP COLUMN StudentPassword

SELECT * FROM STUDENTS

-- STORED PROCEDURES TO INSERT VALUES
GO
CREATE PROCEDURE InsertSTUDENTS
    (
      @studentID nvarchar(50)
      , @firstname varchar(50)
      , @lastname varchar(50)
      , @gender char(1)
      , @Adress nvarchar(100)
      , @zipcode int
      , @birthdate date
      , @email nvarchar(50)
      , @phone nvarchar(35)
      , @entryType varchar(15)
      , @status varchar(10)
      , @regdate int
      , @eduplan int
      , @deptID nvarchar(50)
      , @advisorid int
    )
AS
BEGIN
INSERT INTO STUDENTS(StudentID,StudentFName,StudentLName,StudentGender,StudentAddress,StudentZipcode,StudentBirthDay,StudentEmail,StudentPhone,StudentEntryType,StudentStatus,StudentRegistirationD
VALUES(@studentID,@firstname,@lastname,@gender,@Adress,@zipcode,@birthdate,@email,@phone,@entryType,@status,@regdate,@eduplan,@deptID,@advisorid)
END
--EXEC
EXECUTE InsertSTUDENTS 15058000,TEST,TEST,N,'UNKNOWN',000,'2020-05-17','test.test@yildiz.edu.tr',5555555555,'OSS','PASSIVE',2020,2018,'MTM',1;
EXECUTE InsertSTUDENTS 15058025,TEST2,TEST2,N,'UNKNOWN',000,'2020-05-18','test2.test2@yildiz.edu.tr',5555555555,'OSS','PASSIVE',2020,2018,'MTM',1;

GO
CREATE PROCEDURE InsertFACULTIES
(
    @facultyID nvarchar(50)
    ,@facultyName varchar(50)
    ,@facultyLocation nvarchar(100)
```

```sql
CREATE PROCEDURE InsertFACULTIES
(
    @facultyID nvarchar(50)
   ,@facultyName varchar(50)
   ,@facultyLocation nvarchar(100)
   ,@facultyUniID nvarchar(50)
)
AS
BEGIN
INSERT INTO FACULTIES(FacultyID,FacultyName,FacultyLocation,FacultyUniID)
VALUES (@facultyID,@facultyName,@facultyLocation,@facultyUniID)
END
EXECUTE InsertFACULTIES 'BLM','ELECTRICAL AND ELECTRONICS','DAVUTPASA','TRISTANBU07';
GO
CREATE PROCEDURE InsertCOURSES
(
      @courseCode nvarchar(50)
     ,@courseNameING varchar(50)
     ,@courseNameTR varchar(50)
     ,@courseT int
     ,@courseP int
     ,@courseL int
     ,@courseC int
     ,@courseE int
     ,@coursePrerequisite nvarchar(50)
     ,@courseLanguage varchar(20)
     ,@courseCapacity int
     ,@courseGroupCount int
     ,@courseType char(15)
     ,@courseHour int
     ,@courseDescription text
     ,@courseDepartmentID nvarchar(50)
     ,@courseInstructorIDs int
)
AS
BEGIN
INSERT INTO COURSES(CourseCode,CourseNameING,CourseNameTR,CourseT,CourseP,CourseL,CourseC,CourseE,CoursePrerequisite,CourseLanguage,CourseCapacity,CourseGroupCount,CourseType,CourseHour,CourseDes
VALUES (@courseCode,@courseNameING,@courseNameTR,@courseT,@courseP,@courseL,@courseC,@courseE,@coursePrerequisite,@courseLanguage,@courseCapacity,@courseGroupCount,@courseType,@courseHour,@course
END
--EXEC
EXECUTE InsertCOURSES 'MTM3522','ALGEBRA','CEBIR',3,0,0,3,3,'NONE','ENGLISH',100,2,'DEPARTMENT',3,'ITS ABOUT MATH SCIENCE','MTM',1;
GO

--STORED PROCEDURE TO VIEW GRADES PER COURSE OR STUDENT
GO
CREATE PROCEDURE ViewGrades
(
   @studentid nvarchar(50)=NULL
  ,@fname varchar(50)=null
  ,@lname varchar(50)=null
  ,@stdcourseid nvarchar(50)=null
)
```

USIS.sql

home > sumeyye.seren > Documents > sumeyye.seren > SQL_USISPROJECT > USIS.sql

```sql

--STORED PROCEDURE TO VIEW GRADES PER COURSE OR STUDENT
GO
CREATE PROCEDURE ViewGrades
(
   @studentid nvarchar(50)=NULL
  ,@fname varchar(50)=null
  ,@lname varchar(50)=null
  ,@stdcourseid nvarchar(50)=null
)
AS
BEGIN

    SELECT
       s.StudentID AS [STUDENT ID],
       s.StudentFName AS [STUDENT FIRST],
       s.StudentLName AS [STUDENT LAST],
       e.SemesterYear AS [ENROLLMENT YEAR],
       g.ScoreAlph AS GRADE,
       c.CourseCode AS [COURSE NAME]

   FROM
   STD_ExamGrades g JOIN STD_ENROLLMENT e ON g.GradeEnrollmentID = e.EnrollmentID
   JOIN STUDENTS s ON s.StudentID=e.StudentID
   JOIN COURSES c ON c.CourseCode = e.CourseCode
   WHERE
   (s.StudentID = @studentid OR @studentid IS NULL)
   AND
   (s.StudentFName= @fname OR @fname IS NULL )
   AND
   (s.StudentLName= @lname OR @lname IS NULL )
   AND
   (c.CourseCode = @stdcourseid OR @stdcourseid IS NULL )
END
--EXEC
EXECUTE ViewGrades @studentid = 15058009
GO

--STORE PROCEDURE FOR NEW ENROLLMENTS

CREATE PROCEDURE NewEnrollment
(
     @courseid nvarchar(50)
    ,@studentid nvarchar(50)
    ,@instructerid int
)
AS
BEGIN
INSERT INTO STD_ENROLLMENT(StudentID, SemesterYear)
VALUES (@studentid, YEAR(GETDATE()))

      DECLARE @EnrollmentID int
      SELECT @EnrollmentID = SCOPE_IDENTITY()
```

Ln 1, Col 1 Tab Size: 4 UTF-8 CRLF SQL MSSQL Disconnected

USIS.sql

home > sumeyye.seren > Documents > sumeyye.seren > SQL_USISPROJECT > USIS.sql

```sql
--STORE PROCEDURE FOR NEW ENROLLMENTS

CREATE PROCEDURE NewEnrollment
(
     @courseid nvarchar(50)
    ,@studentid nvarchar(50)
    ,@instructerid int
)
AS
BEGIN
INSERT INTO STD_ENROLLMENT(StudentID, SemesterYear)
VALUES (@studentid, YEAR(GETDATE()))

      DECLARE @EnrollmentID int
      SELECT @EnrollmentID = SCOPE_IDENTITY()

INSERT INTO COURSES(CourseCode,CourseInstructorIDs)
VALUES (@courseid, @instructerid)
END
GO
--TRIGGER TO GET LOG WHEN INSERT, DELETE, UPDATE ON STUDENTS TABLE
Create TRIGGER  USISS
ON STUDENTS FOR INSERT, UPDATE, DELETE NOT FOR REPLICATION
AS
SET NOCOUNT ON;

    DECLARE @operation as Varchar(10)
    DECLARE @Count as int
    SET @operation = 'Inserted'
            SELECT @Count = COUNT(*) FROM DELETED
    if @Count > 0
        BEGIN
            SET @operation = 'Deleted'
            SELECT @Count = COUNT(*) FROM INSERTED
            IF @Count > 0
            SET @operation = 'Updated'
        END

    if @operation = 'Deleted'

            BEGIN
                Insert into LOG(LogDate,LogInfo)
                SELECT GETDATE(),'Deleted'  from deleted
            END


    if @operation = 'Inserted'

            BEGIN
                Insert into LOG(LogDate,LogInfo)
                SELECT GETDATE(),'inserted' from inserted
            END
```

0 0 | Ln 1, Col 1 | Tab Size: 4 | UTF-8 | CRLF | SQL | MSSQL | Disconnected

USIS.sql

home > sumeyye.seren > Documents > sumeyye.seren > SQL_USISPROJECT > USIS.sql

```sql
--TRIGGER TO GET LOG WHEN INSERT, DELETE, UPDATE ON STUDENTS TABLE
Create TRIGGER  USISS
ON STUDENTS FOR INSERT, UPDATE, DELETE NOT FOR REPLICATION
AS
SET NOCOUNT ON;

    DECLARE @operation as Varchar(10)
    DECLARE @Count as int
    SET @operation = 'Inserted'
            SELECT @Count = COUNT(*) FROM DELETED
    if @Count > 0
        BEGIN
            SET @operation = 'Deleted'
            SELECT @Count = COUNT(*) FROM INSERTED
            IF @Count > 0
            SET @operation = 'Updated'
        END

    if @operation = 'Deleted'

            BEGIN
                Insert into LOG(LogDate,LogInfo)
                SELECT GETDATE(),'Deleted'  from deleted
            END


    if @operation = 'Inserted'

            BEGIN
                Insert into LOG(LogDate,LogInfo)
                SELECT GETDATE(),'inserted' from inserted
            END

    ELSE
            BEGIN
                INSERT INTO LOG(LogDate,LogInfo)
                SELECT GETDATE(),'Updated' from inserted

            END
GO
--CHECK
SELECT * FROM LOG

INSERT INTO STUDENTS(StudentID,StudentFName,StudentLName,StudentEmail,StudentGender,StudentPhone,StudentBirthDay,StudentEducationPlan,StudentStatus,StudentDeptID,StudentEntryType,StudentRegistira
VALUES('12345','ABCD','EFGH','12345@std.yildiz.edu.tr','N','+9005555555555','2020-05-18',2018,'PASSIVE','MTM','OSS',2015)
SELECT * FROM LOG


GO
--TRANSACTION EXAMPLES

SELECT * FROM STUDENTS
```

USIS.sql

home > sumeyye.seren > Documents > sumeyye.seren > SQL_USISPROJECT > USIS.sql

```sql
--TRANSACTION EXAMPLES

SELECT * FROM STUDENTS

BEGIN TRY
BEGIN TRANSACTION
DELETE FROM STUDENTS WHERE StudentID='15058001';
UPDATE STUDENTS SET StudentEntryType='UNKNOWN' WHERE StudentLName='TEST';
RAISERROR('THIS STUDENT IS NOT EXIST',16,1)
COMMIT
END TRY
BEGIN CATCH
ROLLBACK
END CATCH

--CHECK-- NOTHING CHANGE , BECAUSE OCCUR AN ERROR IN TRANSACTION
SELECT * FROM STUDENTS

SELECT * FROM LOG
```