```
2. -- Created by Vertabelo (http://vertabelo.com)



-- tables
-- Table: Address
CREATE TABLE vamseepr2.Address (
    AddressId   INTEGER         NOT NULL    IDENTITY(1, 1),
    Address1    NVARCHAR(30)    NOT NULL,
    Address2    NVARCHAR(30)    NULL,
    City        NVARCHAR(30)    NOT NULL,
    State       NVARCHAR(50)    NOT NULL,
    Zipcode     NVARCHAR(5)     NOT NULL    CHECK (Zipcode LIKE '[1-9][0-9][0-9][0-9][0-9]'),
    CONSTRAINT  AddressPk       PRIMARY KEY (AddressId)
)
;

-- Table: Benefits
CREATE TABLE vamseepr2.Benefits (
    BenefitId       INTEGER         NOT NULL    IDENTITY(1, 1),
    BenefitCost     DECIMAL(5,2)    NOT NULL    CHECK (BenefitCost >= 0.0),
    SelectionType   INTEGER         NOT NULL,
    Description     NVARCHAR(100)   NULL,
    CONSTRAINT      BenefitsPk      PRIMARY KEY (BenefitId)
)
;

-- Table: Buildings
CREATE TABLE vamseepr2.Buildings (
    BuildingId      INTEGER         NOT NULL    IDENTITY(1, 1),
    BuildingName    NVARCHAR(50)    NOT NULL,
    CONSTRAINT      BuildingsPk     PRIMARY KEY (BuildingId)
)
;

-- Table: Classroom
CREATE TABLE vamseepr2.Classroom (
    ClassroomId         INTEGER         NOT NULL    IDENTITY(1, 1),
    BuildingId          INTEGER         NOT NULL,
    RoomNumber          NUMERIC(3,0)    NOT NULL    CHECK (RoomNumber > 0),
    MaxSeating          NUMERIC(3,0)    NOT NULL    CHECK (MaxSeating > 0),
    WhiteBoardCount     NUMERIC(2,0)    NULL        CHECK (WhiteBoardCount >= 0),
    AVEquip             NVARCHAR(1000)  NULL,
    ProjectorId         INTEGER         NOT NULL,
    CONSTRAINT          ClassroomPk     PRIMARY KEY (ClassroomId)
)
;

-- Table: College
CREATE TABLE vamseepr2.College (
    CollegeId       INTEGER         NOT NULL    IDENTITY(1, 1),
    CollegeName     NVARCHAR(150)   NOT NULL,
    CONSTRAINT      CollegePk       PRIMARY KEY (CollegeId)
)
;

-- Table: CourseDailySchedule
CREATE TABLE vamseepr2.CourseDailySchedule (
    DailyID             INTEGER     NOT NULL    IDENTITY(1, 1),
    StartTime           TIME(6)     NOT NULL,
    EndTime             TIME(6)     NOT NULL,
    CourseScheduleId    INTEGER     NOT NULL,
    DayOfWeek           INTEGER     NOT NULL,
    CONSTRAINT          CourseDailySchedulePk   PRIMARY KEY  (DailyID)
)
;
```

```
-- Table: CourseGrade
CREATE TABLE vamseepr2.CourseGrade (
    GradeId         INTEGER         NOT NULL    IDENTITY(1, 1),
    Grade           CHAR(1)         NOT NULL,
    Description     NVARCHAR(30)    NOT NULL,
    CONSTRAINT      CourseGradePk   PRIMARY KEY (GradeId)
)
;

-- Table: CourseSchedule
CREATE TABLE vamseepr2.CourseSchedule (
    CourseScheduleId        INTEGER             NOT NULL    IDENTITY(1, 1),
    ClassroomId             INTEGER             NOT NULL,
    SemesterId              INTEGER             NOT NULL,
    Faculty                 INTEGER             NOT NULL,
    CourseSeats             INTEGER             NOT NULL,
    CourseCode              NVARCHAR(3)         NOT NULL,
    CourseNumber            INTEGER             NOT NULL,
    CONSTRAINT              CourseSchedulePk    PRIMARY KEY (CourseScheduleId)
)
;

-- Table: Courses
CREATE TABLE vamseepr2.Courses (
    CourseCode      NVARCHAR(3)     NOT NULL,
    CourseNumber    INTEGER         NOT NULL CHECK  (CourseNumber > 0),
    CourseTitle     NVARCHAR(50)    NOT NULL,
    Description     NVARCHAR(500)   NULL,
    Credits         INTEGER         NOT NULL,
    CONSTRAINT      CoursesPk       PRIMARY KEY     (CourseCode,CourseNumber)
)
;

-- Table: DayOfWeek
CREATE TABLE vamseepr2.DayOfWeek (
    Id          INTEGER         NOT NULL        IDENTITY(1, 1),
    Text        NVARCHAR(30)    NOT NULL,
    CONSTRAINT  DayOfWeekPk     PRIMARY KEY     (Id)
)
;

-- Table: Employees
CREATE TABLE vamseepr2.Employees (
    EmployeeId      INTEGER         NOT NULL    IDENTITY(3670000, 1),
    FirstName       NVARCHAR(50)    NOT NULL,
    LastName        NVARCHAR(50)    NOT NULL,
    NTID            NVARCHAR(25)    NOT NULL,
    SSN             NVARCHAR(max)   NOT NULL    CHECK (SSN LIKE '[1-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9][0-9][0-9]'),
    AddressId       INTEGER         NOT NULL,
    YearlyPay       DECIMAL(8,2)    NOT NULL,
    JobId           INTEGER         NOT NULL,
    HealthBenefits  INTEGER         NOT NULL,
    VisionBenefits  INTEGER         NOT NULL,
    DentalBenefits  INTEGER         NOT NULL,
    IsActive        NVARCHAR(1)     NULL        DEFAULT 'Y' CHECK (IsActive IN ('Y','N')),
    CONSTRAINT      EmployeesPk     PRIMARY KEY (EmployeeId)
)
;

-- Table: Enrollment
CREATE TABLE vamseepr2.Enrollment (
    EnrollmentId        INTEGER         NOT NULL    IDENTITY(1, 1),
    StudentId           INTEGER         NOT NULL,
    GradeId             INTEGER         NOT NULL,
    CourseScheduleId    INTEGER         NOT NULL,
    GradeStatusId       INTEGER         NOT NULL,
    CONSTRAINT          EnrollmentPk    PRIMARY KEY (EnrollmentId)
)
;
```

```
-- Table: GradeStatus
CREATE TABLE vamseepr2.GradeStatus (
    GradeStatusId       INTEGER         NOT NULL    IDENTITY(1, 1),
    Text                NVARCHAR(15)    NOT NULL,
    CONSTRAINT          GradeStatusPk   PRIMARY KEY (GradeStatusId)
)
;

-- Table: JobInformation
CREATE TABLE vamseepr2.JobInformation (
    JobId               INTEGER             NOT NULL    IDENTITY(1, 1),
    JobTitle            NVARCHAR(50)        NOT NULL,
    Description         NVARCHAR(500)       NOT NULL,
    Requirements        NVARCHAR(1000)      NOT NULL,
    MinPay              DECIMAL(5,2)        NOT NULL,
    MaxPay              DECIMAL(7,2)        NOT NULL,
    JobPositionTypeId   INTEGER             NOT NULL,
    IsUnionJob          CHAR(1)             NOT NULL    DEFAULT 'Y' CHECK (IsUnionJob IN ('Y','N')),
    CONSTRAINT          JobInformationPk    PRIMARY KEY (JobId)
)
;

-- Table: JobPositionType
CREATE TABLE vamseepr2.JobPositionType (
    JobPositionTypeId   INTEGER             NOT NULL    IDENTITY(1, 1),
    PostionType         NVARCHAR(30)        NOT NULL,
    CONSTRAINT          JobPositionTypePk   PRIMARY KEY (JobPositionTypeId)
)
;

-- Table: Prerequisites
CREATE TABLE vamseepr2.Prerequisites (
    PrerequisiteId      INTEGER         NOT NULL    IDENTITY(1, 1),
    CourseCode          NVARCHAR(3)     NOT NULL,
    CourseNumber        INTEGER         NOT NULL,
    PrerequisiteCode    NVARCHAR(3)     NOT NULL,
    PrerequisiteNumber  INTEGER         NOT NULL,
    CONSTRAINT          PrerequisitesPk PRIMARY KEY (PrerequisiteId)
)
;

-- Table: ProgramSpecialization
CREATE TABLE vamseepr2.ProgramSpecialization (
    ProgramSpecializationId     INTEGER                 NOT NULL    IDENTITY(1, 1),
    ProgramId                   INTEGER                 NOT NULL,
    StudentId                   INTEGER                 NOT NULL,
    IsMajor                     CHAR(1)                 NOT NULL    DEFAULT 'Y' CHECK (IsMajor IN ('Y','N')),
    CONSTRAINT                  ProgramSpecializationPk PRIMARY KEY (ProgramSpecializationId)
)
;

-- Table: Programs
CREATE TABLE vamseepr2.Programs (
    ProgramId       INTEGER         NOT NULL    IDENTITY(1, 1),
    ProgramName     NVARCHAR(50)    NOT NULL,
    CollegeId       INTEGER         NOT NULL,
    CONSTRAINT      ProgramsPk      PRIMARY KEY (ProgramId)
)
;

-- Table: Projector
CREATE TABLE vamseepr2.Projector (
    ProjectorId     INTEGER         NOT NULL    IDENTITY(1, 1),
    Text            NVARCHAR(30)    NOT NULL,
    CONSTRAINT      ProjectorPk     PRIMARY KEY (ProjectorId)
)
;
```

```
-- Table: SelectionType
CREATE TABLE vamseepr2.SelectionType (
    SelectionTypeId     INTEGER         NOT NULL    IDENTITY(1, 1),
    Selection           NVARCHAR(30)    NOT NULL,
    CONSTRAINT          SelectionTypePk PRIMARY KEY (SelectionTypeId)
)
;

-- Table: Semester
CREATE TABLE vamseepr2.Semester (
    SemesterId      INTEGER     NOT NULL    IDENTITY(1, 1),
    Year            INTEGER     NOT NULL,
    FirstDay        DATE        NOT NULL,
    LastDay         DATE        NOT NULL,
    SemesterTextId  INTEGER     NOT NULL,
    CONSTRAINT      SemesterPk  PRIMARY KEY (SemesterId)
)
;

-- Table: SemesterText
CREATE TABLE vamseepr2.SemesterText (
    SemesterTextId  INTEGER         NOT NULL    IDENTITY(1, 1),
    Text            NVARCHAR(30)    NOT NULL,
    CONSTRAINT      SemesterTextPk  PRIMARY KEY (SemesterTextId)
)
;

-- Table: StudentStatus
CREATE TABLE vamseepr2.StudentStatus (
    StudentStatusId     INTEGER         NOT NULL    IDENTITY(1, 1),
    StudentStatus       NVARCHAR(30)    NOT NULL,
    CONSTRAINT          StudentStatusPk PRIMARY KEY (StudentStatusId)
)
;

-- Table: Students
CREATE TABLE vamseepr2.Students (
    StudentId       INTEGER         NOT NULL    IDENTITY(10000, 1),
    FirstName       NVARCHAR(50)    NOT NULL,
    LastName        NVARCHAR(50)    NOT NULL,
    NTID            NVARCHAR(25)    NOT NULL,
    Password        NVARCHAR(8)     NULL,
    DateOfBirth     DATE            NOT NULL,
    SSN             NVARCHAR(max)   NULL        CHECK (SSN LIKE '[1-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9][0-9][0-9]'),
    StudentStatusId INTEGER         NOT NULL,
    HomeAddress     INTEGER         NOT NULL,
    LocalAddress    INTEGER         NOT NULL,
    BillAmount      DECIMAL(8,2)    NULL        DEFAULT 0.0,
    CONSTRAINT      StudentsPk      PRIMARY KEY (StudentId)
)
;




-- foreign keys
-- Reference:  FkBuidlings (table: vamseepr2.Classroom)


ALTER TABLE vamseepr2.Classroom ADD CONSTRAINT FkBuidlings
    FOREIGN KEY     (BuildingId)
    REFERENCES      vamseepr2.Buildings (BuildingId)
;



-- Reference:  FkClassroom (table: vamseepr2.CourseSchedule)
```

```
ALTER TABLE vamseepr2.CourseSchedule ADD CONSTRAINT FkClassroom
    FOREIGN KEY     (ClassroomId)
    REFERENCES      vamseepr2.Classroom (ClassroomId)
;

-- Reference:  FkCollege (table: vamseepr2.Programs)


ALTER TABLE vamseepr2.Programs ADD CONSTRAINT FkCollege
    FOREIGN KEY     (CollegeId)
    REFERENCES      vamseepr2.College (CollegeId)
;

-- Reference:  FkCourseDailySchedule (table: vamseepr2.CourseDailySchedule)


ALTER TABLE vamseepr2.CourseDailySchedule ADD CONSTRAINT FkCourseDailySchedule
    FOREIGN KEY     (CourseScheduleId)
    REFERENCES      vamseepr2.CourseSchedule (CourseScheduleId)
;

-- Reference:  FkCourseScheduleCourses (table: vamseepr2.CourseSchedule)


ALTER TABLE vamseepr2.CourseSchedule ADD CONSTRAINT FkCourseScheduleCourses
    FOREIGN KEY     (CourseCode,CourseNumber)
    REFERENCES      vamseepr2.Courses (CourseCode,CourseNumber)
;

-- Reference:  FkDayOfWeek (table: vamseepr2.CourseDailySchedule)


ALTER TABLE vamseepr2.CourseDailySchedule ADD CONSTRAINT FkDayOfWeek
    FOREIGN KEY     (DayOfWeek)
    REFERENCES      vamseepr2.DayOfWeek (Id)
;

-- Reference:  FkDentalBenefits (table: vamseepr2.Employees)


ALTER TABLE vamseepr2.Employees ADD CONSTRAINT FkDentalBenefits
    FOREIGN KEY     (VisionBenefits)
    REFERENCES      vamseepr2.Benefits (BenefitId)
;

-- Reference:  FkEmployeesAddress (table: vamseepr2.Employees)


ALTER TABLE vamseepr2.Employees ADD CONSTRAINT FkEmployeesAddress
    FOREIGN KEY     (AddressId)
    REFERENCES      vamseepr2.Address (AddressId)
;

-- Reference:  FkEnrollment (table: vamseepr2.Enrollment)


ALTER TABLE vamseepr2.Enrollment ADD CONSTRAINT FkEnrollment
    FOREIGN KEY     (StudentId)
    REFERENCES      vamseepr2.Students (StudentId)
;

-- Reference:  FkEnrollmentCourseGrade (table: vamseepr2.Enrollment)

ALTER TABLE vamseepr2.Enrollment ADD CONSTRAINT FkEnrollmentCourseGrade
    FOREIGN KEY     (GradeId)
    REFERENCES      vamseepr2.CourseGrade (GradeId)
;
```

```
-- Reference:  FkEnrollmentCourseSchedule (table: vamseepr2.Enrollment)


ALTER TABLE vamseepr2.Enrollment ADD CONSTRAINT FkEnrollmentCourseSchedule
    FOREIGN KEY     (CourseScheduleId)
    REFERENCES      vamseepr2.CourseSchedule (CourseScheduleId)
;

-- Reference:  FkEnrollmentGradeStatus (table: vamseepr2.Enrollment)


ALTER TABLE vamseepr2.Enrollment ADD CONSTRAINT FkEnrollmentGradeStatus
    FOREIGN KEY     (GradeStatusId)
    REFERENCES      vamseepr2.GradeStatus (GradeStatusId)
;

-- Reference:  FkFaculty (table: vamseepr2.CourseSchedule)


ALTER TABLE vamseepr2.CourseSchedule ADD CONSTRAINT FkFaculty
    FOREIGN KEY     (Faculty)
    REFERENCES      vamseepr2.Employees (EmployeeId)
;

-- Reference:  FkHealthBenefits (table: vamseepr2.Employees)


ALTER TABLE vamseepr2.Employees ADD CONSTRAINT FkHealthBenefits
    FOREIGN KEY     (HealthBenefits)
    REFERENCES      vamseepr2.Benefits (BenefitId)
;

-- Reference:  FkJobInformation (table: vamseepr2.Employees)


ALTER TABLE vamseepr2.Employees ADD CONSTRAINT FkJobInformation
    FOREIGN KEY     (JobId)
    REFERENCES      vamseepr2.JobInformation (JobId)
;

-- Reference:  FkJobPositionType (table: vamseepr2.JobInformation)


ALTER TABLE vamseepr2.JobInformation ADD CONSTRAINT FkJobPositionType
    FOREIGN KEY     (JobPositionTypeId)
    REFERENCES      vamseepr2.JobPositionType (JobPositionTypeId)
;

-- Reference:  FkParentCourses (table: vamseepr2.Prerequisites)


ALTER TABLE vamseepr2.Prerequisites ADD CONSTRAINT FkParentCourses
    FOREIGN KEY     (CourseCode,CourseNumber)
    REFERENCES      vamseepr2.Courses (CourseCode,CourseNumber)
;

-- Reference:  FkPrerequisitesCourses (table: vamseepr2.Prerequisites)


ALTER TABLE vamseepr2.Prerequisites ADD CONSTRAINT FkPrerequisitesCourses
    FOREIGN KEY     (PrerequisiteCode,PrerequisiteNumber)
    REFERENCES      vamseepr2.Courses (CourseCode,CourseNumber)
;

-- Reference:  FkProgramSpecilization (table: vamseepr2.ProgramSpecialization)

ALTER TABLE vamseepr2.ProgramSpecialization ADD CONSTRAINT FkProgramSpecilization
    FOREIGN KEY     (StudentId)
    REFERENCES      vamseepr2.Students (StudentId)
;
```

```
-- Reference:  FkPrograms (table: vamseepr2.ProgramSpecialization)


ALTER TABLE vamseepr2.ProgramSpecialization ADD CONSTRAINT FkPrograms
    FOREIGN KEY     (ProgramId)
    REFERENCES      vamseepr2.Programs (ProgramId)
;

-- Reference:  FkProjector (table: vamseepr2.Classroom)


ALTER TABLE vamseepr2.Classroom ADD CONSTRAINT FkProjector
    FOREIGN KEY     (ProjectorId)
    REFERENCES      vamseepr2.Projector (ProjectorId)
;

-- Reference:  FkSectionSemester (table: vamseepr2.CourseSchedule)


ALTER TABLE vamseepr2.CourseSchedule ADD CONSTRAINT FkSectionSemester
    FOREIGN KEY     (SemesterId)
    REFERENCES      vamseepr2.Semester (SemesterId)
;

-- Reference:  FkSelectionType (table: vamseepr2.Benefits)


ALTER TABLE vamseepr2.Benefits ADD CONSTRAINT FkSelectionType
    FOREIGN KEY     (SelectionType)
    REFERENCES      vamseepr2.SelectionType (SelectionTypeId)
;

-- Reference:  FkSemesterText (table: vamseepr2.Semester)


ALTER TABLE vamseepr2.Semester ADD CONSTRAINT FkSemesterText
    FOREIGN KEY     (SemesterTextId)
    REFERENCES      vamseepr2.SemesterText (SemesterTextId)
;

-- Reference:  FkStudentStatus (table: vamseepr2.Students)


ALTER TABLE vamseepr2.Students ADD CONSTRAINT FkStudentStatus
    FOREIGN KEY     (StudentStatusId)
    REFERENCES      vamseepr2.StudentStatus (StudentStatusId)
;

-- Reference:  FkStudentsHomeAddress (table: vamseepr2.Students)


ALTER TABLE vamseepr2.Students ADD CONSTRAINT FkStudentsHomeAddress
    FOREIGN KEY     (HomeAddress)
    REFERENCES      vamseepr2.Address (AddressId)
;

-- Reference:  FkStudentsLocalAddress (table: vamseepr2.Students)


ALTER TABLE vamseepr2.Students ADD CONSTRAINT FkStudentsLocalAddress
    FOREIGN KEY     (LocalAddress)
    REFERENCES      vamseepr2.Address (AddressId)
;

-- Reference:  FkVisionBenefits (table: vamseepr2.Employees)


ALTER TABLE vamseepr2.Employees ADD CONSTRAINT FkVisionBenefits
    FOREIGN KEY     (DentalBenefits)
```

```
    REFERENCES      vamseepr2.Benefits (BenefitId)
;

-- End of table creation (Vertabelo)


/*************************************************************************************************/


3. --Insert statements for all tables (Vamsee)

INSERT INTO vamseepr2.Address
    VALUES('DOOR NO. 1','710 WESTCOTT','SYRACUSE','NY','13210');

INSERT INTO vamseepr2.Address
    VALUES('1113 EAST FAYETTE STREET','','SYRACUSE','NY','19910');

INSERT INTO vamseepr2.Address
    VALUES('100/2A C2 AKASIA BUIDLING','','ATLANTA','GR','13410');

INSERT INTO vamseepr2.Address
    VALUES('DOOR NO. 78','227 COMSTOCK','VIRGINIA','VG','13210');

INSERT INTO vamseepr2.Address
    VALUES('HOUSE NO. 69','HOLLYWOOD','SAN DIEGO','CL','11110');

INSERT INTO vamseepr2.Address
    VALUES('DUBAI MAIN BUSTAND','DUBAI MAIN ROAD','DUBAI','DB','67210');

INSERT INTO vamseepr2.Address
    VALUES('WHITE HOUSE','','WASHINGTON','DC','25981');

INSERT INTO vamseepr2.Address
    VALUES('NO. 9 LIVERPOOL STREET','','BOSTON','MA','67210');

INSERT INTO vamseepr2.Address
    VALUES('23, MERLYN STREET','SOUTH','NEW YORK CITY','NY','89210');

INSERT INTO vamseepr2.Address
    VALUES('223 E.CONCORD STREET','','ORLANDO','FL','67111');

SELECT * FROM vamseepr2.Address;

-----------------------------------------------------

INSERT INTO vamseepr2.Benefits
    VALUES(0.0,3,'NO BENEFIT TAKEN');

INSERT INTO vamseepr2.Benefits
    VALUES(300.2,1,'BENEFIT FOR A SINGLE PERSON');

INSERT INTO vamseepr2.Benefits
    VALUES(850.5,2,'BENEFIT FOR EMPLOYEE & FAMILY');

SELECT * FROM vamseepr2.Benefits;

-----------------------------------------------------

INSERT INTO vamseepr2.Buildings
    VALUES('HAWKINS BUILDING');

INSERT INTO vamseepr2.Buildings
    VALUES('PHYSICS BUILDING');

INSERT INTO vamseepr2.Buildings
    VALUES('SHAFFER BUILDING');

INSERT INTO vamseepr2.Buildings
    VALUES('TOLLEY BUILDING');
```

```
INSERT INTO vamseepr2.Buildings
    VALUES('WHITMAN BUILDING');

INSERT INTO vamseepr2.Buildings
    VALUES('ECS BUILDING');

SELECT * FROM vamseepr2.Buildings;

------------------------------------------------------

INSERT INTO vamseepr2.Classroom
    VALUES(1,300,50,0,'Screen',1);

INSERT INTO vamseepr2.Classroom
    VALUES(1,500,100,2,'Audio,Video,Mic',2);

INSERT INTO vamseepr2.Classroom
    VALUES(6,120,10,1,'Mic',2);

INSERT INTO vamseepr2.Classroom(BuildingId,RoomNumber,MaxSeating,ProjectorId)
    VALUES(3,365,20,3);

INSERT INTO vamseepr2.Classroom(BuildingId,RoomNumber,MaxSeating,WhiteBoardCount,ProjectorId)
    VALUES(4,500,80,2,2);

INSERT INTO vamseepr2.Classroom(BuildingId,RoomNumber,MaxSeating,AVEquip,ProjectorId)
    VALUES(2,010,100,'Mic,Speaker',1);

SELECT * FROM vamseepr2.Classroom;

------------------------------------------------------

INSERT INTO vamseepr2.College
    VALUES('SCHOOL OF ARCHITECTURE');

INSERT INTO vamseepr2.College
    VALUES('ENGINEERING & COMPUTER SCIENCE');

INSERT INTO vamseepr2.College
    VALUES('COLLEGE OF LAW');

INSERT INTO vamseepr2.College
    VALUES('SI NEW HOUSE');

INSERT INTO vamseepr2.College
    VALUES('MARTIN SCHOOL OF CITIZENSHIP');

INSERT INTO vamseepr2.College
    VALUES('COLLEGE OF MUSIC');

SELECT * FROM vamseepr2.College;

------------------------------------------------------

INSERT INTO vamseepr2.CourseDailySchedule
    VALUES('08:00','10:50',1,1);

INSERT INTO vamseepr2.CourseDailySchedule
    VALUES('11:00','12:30',8,1);

INSERT INTO vamseepr2.CourseDailySchedule
    VALUES('14:00','15:10',5,1);

INSERT INTO vamseepr2.CourseDailySchedule
    VALUES('08:00','9:50',4,3);

INSERT INTO vamseepr2.CourseDailySchedule
    VALUES('15:15','18:15',3,5);

INSERT INTO vamseepr2.CourseDailySchedule
```

```
    VALUES('07:50','10:20',6,3);

INSERT INTO vamseepr2.CourseDailySchedule
    VALUES('08:00','11:00',4,4);

INSERT INTO vamseepr2.CourseDailySchedule
    VALUES('08:00','10:50',1,4);

SELECT * FROM vamseepr2.CourseDailySchedule;

-----------------------------------------------------

INSERT INTO vamseepr2.CourseGrade
    VALUES('O', 'Outstanding');

INSERT INTO vamseepr2.CourseGrade
    VALUES('E', 'Exceeds Expectation');

INSERT INTO vamseepr2.CourseGrade
    VALUES('A', 'Acceptable');

INSERT INTO vamseepr2.CourseGrade
    VALUES('P', 'Poor');

INSERT INTO vamseepr2.CourseGrade
    VALUES('D', 'Dreadful');

INSERT INTO vamseepr2.CourseGrade
    VALUES('T', 'Troll');

SELECT * FROM vamseepr2.CourseGrade;

-----------------------------------------------------

INSERT INTO vamseepr2.Courses
    VALUES('CIS',300,'C PROGRAMMING',NULL,3);

INSERT INTO vamseepr2.Courses
    VALUES('CIS',400,'JAVA PROGRAMMING',NULL,3);

INSERT INTO vamseepr2.Courses
    VALUES('CIS',655,'COMPUTER ARCHITECTURE','Introduction to digital design. Interfacing of devices for I/O,      ↙
    memory and memory management. Input/output programming, via wait loops, hardware interrupts and calls to      ↙
    operating system services. ',4);

INSERT INTO vamseepr2.Courses
    VALUES('MSC',800,'STRING & CHORDS',NULL,3);

INSERT INTO vamseepr2.Courses
    VALUES('MSC',810,'MUSIC COMPOSITION & TUNING',NULL,5);

INSERT INTO vamseepr2.Courses
    VALUES('LAW',601,'CIVIL PROCEDURE','Procedural processes that guide the adjudication of civil actions in      ↙
    American courts. Allocation of judicial power between federal and state courts, focusing on the Federal Rules  ↙
    of Civil Procedure. Fundamental policies underlying particular procedural rules.',4);

INSERT INTO vamseepr2.Courses
    VALUES('LAW',603,'CONTRACTS',NULL,5);

INSERT INTO vamseepr2.Courses
    VALUES('LAW',604,'CRIMINAL LAW','Elements of various crimes and problems of statutory construction and        ↙
    interpretation. Substantive defenses, emphasizing the defense of insanity, as well as attempts and the specific↙
     crimes of conspiracy, theft, and homicide.',3);

INSERT INTO vamseepr2.Courses
    VALUES('SIH',208,'BIG IDEA IN ADVERTISING',NULL,3);

INSERT INTO vamseepr2.Courses
    VALUES('SIH',400,'DIVERSITY IN FASHION MEDIA','The issues that arise in the fashion industry and the messages  ↙
    that it communicates to the public through its associated media outlets.',3);
```

```
INSERT INTO vamseepr2.Courses
    VALUES('ARC',634,'THE ARCHITECTURE REVOLUTIONS',NULL,3);

INSERT INTO vamseepr2.Courses
    VALUES('ARC',500,'PLANS','This lecture/seminar course is primarily concerned with developing one's ability to
    read architectural ideas through the convention of plan',3);

INSERT INTO vamseepr2.Courses
    VALUES('ARC',611,'STRUCTURES','Structure introduces basic concepts of structural systems behavior including
    gravity and lateral loads, analysis of major structural forms, and the structural performance of materials. The
     final evaluation includes a research project.',3);

INSERT INTO vamseepr2.Courses
    VALUES('PAI',734,'PUBLIC BUDGETING',NULL,4);

INSERT INTO vamseepr2.Courses
    VALUES('PAI',698,'MEDIA LAW',NULL,3);


SELECT * FROM vamseepr2.Courses;

-----------------------------------------------------

INSERT INTO vamseepr2.CourseSchedule
    VALUES(1,1,3670001,50,'CIS',300);

INSERT INTO vamseepr2.CourseSchedule
    VALUES(1,1,3670001,50,'CIS',300);

INSERT INTO vamseepr2.CourseSchedule
    VALUES(1,2,3670003,50,'CIS',400);

INSERT INTO vamseepr2.CourseSchedule
    VALUES(6,2,3670002,150,'MSC',810);

INSERT INTO vamseepr2.CourseSchedule
    VALUES(3,2,3670003,150,'SIH',208);

INSERT INTO vamseepr2.CourseSchedule
    VALUES(4,2,3670010,20,'LAW',601);

INSERT INTO vamseepr2.CourseSchedule
    VALUES(5,3,3670010,45,'LAW',604);

INSERT INTO vamseepr2.CourseSchedule
    VALUES(4,1,3670003,35,'SIH',400);

SELECT * FROM vamseepr2.CourseSchedule;

-----------------------------------------------------

INSERT INTO vamseepr2.DayOfWeek
    VALUES('MONDAY');

INSERT INTO vamseepr2.DayOfWeek
    VALUES('TUESDAY');

INSERT INTO vamseepr2.DayOfWeek
    VALUES('WEDNESDAY');

INSERT INTO vamseepr2.DayOfWeek
    VALUES('THURSDAY');

INSERT INTO vamseepr2.DayOfWeek
    VALUES('FRIDAY');

INSERT INTO vamseepr2.DayOfWeek
    VALUES('SATURDAY');
```

```
INSERT INTO vamseepr2.DayOfWeek
    VALUES('SUNDAY');

SELECT * FROM vamseepr2.DayOfWeek;

-----------------------------------------------------

INSERT INTO vamseepr2.Employees
    VALUES('RICKY','MARTIN','RCKYMRT','789-45-6123',1,120000.0,3,2,2,2,'Y');

INSERT INTO vamseepr2.Employees
    VALUES('SAGE','BROSE','SGBR','147-25-3698',6,120000.0,3,3,2,2,'Y');

INSERT INTO vamseepr2.Employees
    VALUES('RON','PAPPAN','RNPPA','555-55-5555',5,24444.5,2,1,1,1,'Y');

INSERT INTO vamseepr2.Employees
    VALUES('JULES','ALCOCER','JULCER','333-22-4444',4,65000.02,4,3,3,3,'Y');

INSERT INTO vamseepr2.Employees
    VALUES('GRAYCE','BASHAW','GAYBA','654-12-8739',4,25500.50,4,2,1,1,'Y');

INSERT INTO vamseepr2.Employees
    VALUES('ANNETTA','TUCH','ANNCH','569-78-3658',2,50000.0,4,1,2,2,'Y');

INSERT INTO vamseepr2.Employees
    VALUES('CHARLEY','LEFFEW','FEWCHA','258-96-4123',10,100000.0,9,3,3,1,'Y');

INSERT INTO vamseepr2.Employees
    VALUES('ELANE','MILLSAPS','MILLNE','658-78-9999',4,11660.0,9,1,1,1,'N');

INSERT INTO vamseepr2.Employees
    VALUES('JENINE','HOUZE','HOUNI','856-45-6587',4,14560.80,5,3,3,3,'Y');

INSERT INTO vamseepr2.Employees
    VALUES('SEPTEMBER','WARE','SEPARE','236-12-5973',2,8000.0,6,3,1,1,'Y');

INSERT INTO vamseepr2.Employees
    VALUES('JESSIE','WARE','JESARE','836-92-4673',2,28000.0,2,3,1,1,'Y');

SELECT * FROM vamseepr2.Employees;

-----------------------------------------------------

INSERT INTO vamseepr2.Enrollment
    VALUES(10000,1,1,2);

INSERT INTO vamseepr2.Enrollment
    VALUES(10001,1,3,2);

INSERT INTO vamseepr2.Enrollment
    VALUES(10005,1,4,2);

INSERT INTO vamseepr2.Enrollment
    VALUES(10003,6,8,2);

INSERT INTO vamseepr2.Enrollment
    VALUES(10004,4,8,1);

INSERT INTO vamseepr2.Enrollment
    VALUES(10004,2,6,1);

INSERT INTO vamseepr2.Enrollment
    VALUES(10002,2,6,3);

INSERT INTO vamseepr2.Enrollment
    VALUES(10001,2,2,3);

SELECT * FROM vamseepr2.Enrollment;
```

```
-----------------------------------------------------

INSERT INTO vamseepr2.GradeStatus
    VALUES('REGULAR');

INSERT INTO vamseepr2.GradeStatus
    VALUES('PASS/FAIL');

INSERT INTO vamseepr2.GradeStatus
    VALUES('AUDIT');

SELECT * FROM vamseepr2.GradeStatus;

-----------------------------------------------------

INSERT INTO vamseepr2.JobInformation
    VALUES('ASST PROF', 'PROF WITH 1 YEAR TEACHING EXP', 'MS/Phd',500.5,8000.5,1,'Y');

INSERT INTO vamseepr2.JobInformation
    VALUES('PROF', 'PROF WITH 3 YEAR TEACHING EXP', 'MS/Phd',900.5,10000.5,1,'Y');

INSERT INTO vamseepr2.JobInformation
    VALUES('SENIOR PROF', 'PROF WITH 5 YEAR TEACHING EXP', 'MS/Phd, Research',900.5,21000.5,1,'Y');

INSERT INTO vamseepr2.JobInformation
    VALUES('BUSINESS OPS', 'EXP OF 1 YEAR MARKETING', 'MANAGER',800.5,8000.5,2,'N');

INSERT INTO vamseepr2.JobInformation
    VALUES('ADMISSION OFFICER', 'N/A', 'N/A',700.5,7999.99,2,'N');

INSERT INTO vamseepr2.JobInformation
    VALUES('ADMIN SPECIALIST 1', 'Clerical and administrative support to professionals', '',500.5,8000.5,3,'N');

INSERT INTO vamseepr2.JobInformation
    VALUES('ADMIN SPECIALIST 2', 'Coordination and implementation of office procedures and frequently have        ↙
    responsibility for specific projects and tasks', '',800.5,15000.5,3,'N');

INSERT INTO vamseepr2.JobInformation
    VALUES('CASIER', 'Collects payments by accepting cash, check, or charge payments from customers; making change ↙
    for cash customers', 'General Math Skills, Informing Others, Basic Safety, Job Knowledge',850.0,1200.99,4,'Y');

INSERT INTO vamseepr2.JobInformation
    VALUES('ACCOUNTANT', 'Substantiates financial transactions by auditing documents', 'Confidentiality, Time      ↙
    Management, Data Entry Management, General Math Skills',999.5,9999.99,4,'Y');

SELECT * FROM vamseepr2.JobInformation;

-----------------------------------------------------

INSERT INTO vamseepr2.JobPositionType
    VALUES('FACULTY');

INSERT INTO vamseepr2.JobPositionType
    VALUES('MANAGEMENT');

INSERT INTO vamseepr2.JobPositionType
    VALUES('FINANCIER');

INSERT INTO vamseepr2.JobPositionType
    VALUES('ADMIN');

SELECT * FROM vamseepr2.JobPositionType;

-----------------------------------------------------

INSERT INTO vamseepr2.Prerequisites
    VALUES('CIS',655,'CIS',300);

INSERT INTO vamseepr2.Prerequisites
    VALUES('CIS',655,'CIS',400);
```

```
INSERT INTO vamseepr2.Prerequisites
    VALUES('ARC',611,'ARC',500);

INSERT INTO vamseepr2.Prerequisites
    VALUES('MSC',810,'MSC',800);

INSERT INTO vamseepr2.Prerequisites
    VALUES('LAW',603,'LAW',601);

SELECT * FROM vamseepr2.Prerequisites;

-----------------------------------------------------

INSERT INTO vamseepr2.Programs
    VALUES('B.ARCH',1);

INSERT INTO vamseepr2.Programs
    VALUES('DESGIN AND BUILD TECHINIQUES',1);

INSERT INTO vamseepr2.Programs
    VALUES('COMPUTER SCIENCE',2);

INSERT INTO vamseepr2.Programs
    VALUES('COMPUTER ENGINEERING',2);

INSERT INTO vamseepr2.Programs
    VALUES('ELECTRICAL ENGINEERING',2);

INSERT INTO vamseepr2.Programs
    VALUES('BIO.TECH ENGINEERING',2);

INSERT INTO vamseepr2.Programs
    VALUES('STUDY OF CRIMINAL AND CIVIL CASES',3);

INSERT INTO vamseepr2.Programs
    VALUES('ADVOCACY',3);

INSERT INTO vamseepr2.Programs
    VALUES('ADVERTISING',4);

INSERT INTO vamseepr2.Programs
    VALUES('PUBLIC RELATIONS',4);

INSERT INTO vamseepr2.Programs
    VALUES('TELEVISION,RADIO & FILM',4);

INSERT INTO vamseepr2.Programs
    VALUES('ADMINISTARTIO & INTERNATIONLA RELATIONS',5);

INSERT INTO vamseepr2.Programs
    VALUES('MUSIC COMPOSITION',6);

INSERT INTO vamseepr2.Programs
    VALUES('APPLIED MUSIC & PERFORMANCE',6);

SELECT * FROM vamseepr2.Programs;
-----------------------------------------------------

INSERT INTO vamseepr2.ProgramSpecialization
    VALUES(4,10000,'Y');

INSERT INTO vamseepr2.ProgramSpecialization
    VALUES(6,10001,'N');

INSERT INTO vamseepr2.ProgramSpecialization
    VALUES(6,10000,'N');

INSERT INTO vamseepr2.ProgramSpecialization
    VALUES(11,10003,'Y');
```

```
INSERT INTO vamseepr2.ProgramSpecialization
    VALUES(8,10005,'Y');

INSERT INTO vamseepr2.ProgramSpecialization(ProgramId,StudentId)
    VALUES(6,10002);

SELECT * FROM vamseepr2.ProgramSpecialization;

-----------------------------------------------------

INSERT INTO vamseepr2.Projector
    VALUES('YES');

INSERT INTO vamseepr2.Projector
    VALUES('SMART BOARD');

INSERT INTO vamseepr2.Projector
    VALUES('NO');

SELECT * FROM vamseepr2.Projector;

-----------------------------------------------------

INSERT INTO vamseepr2.SelectionType
    VALUES('SINGLE');

INSERT INTO vamseepr2.SelectionType
    VALUES('FAMILY');

INSERT INTO vamseepr2.SelectionType
    VALUES('OP-OUT');

SELECT * FROM vamseepr2.SelectionType;

-----------------------------------------------------

INSERT INTO vamseepr2.Semester
    VALUES(2015,'05-AUG-2015','18-DEC-2015',1);

INSERT INTO vamseepr2.Semester
    VALUES(2016,'19-JAN-2016','19-MAY-2016',2);

INSERT INTO vamseepr2.Semester
    VALUES(2014,'05-AUG-2014','17-DEC-2014',1);

INSERT INTO vamseepr2.Semester
    VALUES(2015,'05-MAY-2015','18-JUN-2015',2);

INSERT INTO vamseepr2.Semester
    VALUES(2015,'25-JUN-2015','18-JUL-2015',3);

INSERT INTO vamseepr2.Semester
    VALUES(2014,'12-JAN-2014','05-MAY-2014',2);

SELECT * FROM vamseepr2.Semester;

-----------------------------------------------------

INSERT INTO vamseepr2.SemesterText
    VALUES('FALL');

INSERT INTO vamseepr2.SemesterText
    VALUES('SPRING');

INSERT INTO vamseepr2.SemesterText
    VALUES('SUMMER I');

INSERT INTO vamseepr2.SemesterText
    VALUES('SUMMER II');
```

```
INSERT INTO vamseepr2.SemesterText
    VALUES('COMBINED SUMMER');

SELECT * FROM vamseepr2.SemesterText;

-----------------------------------------------------

INSERT INTO vamseepr2.Students
    VALUES('MARK','ANTONY','MRKANTY','hellowor','05-AUG-91',NULL,1,1,1);

INSERT INTO vamseepr2.Students
    VALUES('TONY','STARK','STRKTN','jarvis78','1-JAN-72','999-99-9999',1,5,1);

INSERT INTO vamseepr2.Students
    VALUES('INCREDIBLE','HULK','INCHLK',NULL,'11-MAR-87',NULL,4,2,6);

INSERT INTO vamseepr2.Students
    VALUES('JIMMY','FALLON','JMFLLN',NULL,'19-SEP-71','567-78-8901',4,3,3);

INSERT INTO vamseepr2.Students
    VALUES('CONAN','O BREIN','CNNBRN','myshow1','20-JUL-96',NULL,1,5,1);

INSERT INTO vamseepr2.Students
    VALUES('HARRY','POTTER','HRPTR','magic','11-JAN-91',NULL,3,10,8);

SELECT * FROM vamseepr2.Students;

-----------------------------------------------------

INSERT INTO vamseepr2.StudentStatus
    VALUES('UNDERGRADUATE');

INSERT INTO vamseepr2.StudentStatus
    VALUES('GRADUATE');

INSERT INTO vamseepr2.StudentStatus
    VALUES('NON-MATRICULATED');

INSERT INTO vamseepr2.StudentStatus
    VALUES('GRADUATED');

SELECT * FROM vamseepr2.StudentStatus;


--End Of insert (Vamsee)


/*************************************************************************************************/


4. --Views (Vamsee)


/*This view displays the class schedule for all students*/

CREATE VIEW vamseepr2.viewStudentClassSchedule AS
    SELECT S.FirstName+' '+S.LastName                       AS StudentName,
        CS.CourseCode+''+CAST(CS.CourseNumber AS VARCHAR)   AS CourseName,
        D.Text                                              AS Day
        FROM vamseepr2.Students S
            LEFT OUTER JOIN vamseepr2.Enrollment E
            ON S.StudentId = E.StudentId
                INNER JOIN vamseepr2.CourseSchedule CS
                ON E.CourseScheduleId = CS.CourseScheduleId
                    INNER JOIN vamseepr2.CourseDailySchedule CDS
                    ON CS.CourseScheduleId = CDS.CourseScheduleId
                        INNER JOIN vamseepr2.DayOfWeek D
                        ON CDS.DayOfWeek = D.Id;
```

```
-----------------------------------------------------------------------------------------------------------------


/*This view displays the premium for each type of benefit all active employees have chosen*/

CREATE VIEW vamseepr2.viewEmployeeBenefits AS
    SELECT E.FirstName+' '+E.LastName   AS EmployeeName,
            B1.BenefitCost * 12         AS HealthPremium,
            B2.BenefitCost * 12         AS VisionPremium,
            B3.BenefitCost * 12         AS DentalPremium
            FROM vamseepr2.Employees E
                LEFT OUTER JOIN vamseepr2.Benefits B1
                ON E.HealthBenefits = B1.BenefitId
                    LEFT OUTER JOIN vamseepr2.Benefits B2
                    ON E.VisionBenefits = B2.BenefitId
                        LEFT OUTER JOIN vamseepr2.Benefits B3
                        ON E.DentalBenefits = B3.BenefitId
                        WHERE E.IsActive = 'Y'

-----------------------------------------------------------------------------------------------------------------

/*This view displays the course, it's faulty and the number of enrollments*/

CREATE VIEW vamseepr2.viewCourseEnrollments AS
        SELECT CS.CourseCode+''+CAST(CS.CourseNumber AS VARCHAR)    AS CourseName,
                E.FirstName+' '+E.LastName                          AS FacultyName,
                COUNT(EN.EnrollmentId)                              AS Strength
                FROM ((vamseepr2.CourseSchedule CS
                    LEFT OUTER JOIN vamseepr2.Employees E
                    ON CS.Faculty = E.EmployeeId)
                        LEFT OUTER JOIN vamseepr2.Enrollment EN
                        ON CS.CourseScheduleId = EN.CourseScheduleId)
                        GROUP BY CS.CourseCode+''+CAST(CS.CourseNumber AS VARCHAR),E.FirstName+' '+E.LastName

-----------------------------------------------------------------------------------------------------------------


/*This view displays the students with a major, major and the college*/

CREATE VIEW vamseepr2.viewStudentMajor AS
        SELECT S.FirstName+' '+S.LastName   AS StudentName,
        P.ProgramName                       AS Major,
        C.CollegeName                       AS College
        FROM vamseepr2.Students S
            LEFT OUTER JOIN vamseepr2.ProgramSpecialization PS
            ON S.StudentId = PS.StudentId
                LEFT OUTER JOIN vamseepr2.Programs P
                ON PS.ProgramId = P.ProgramId
                    LEFT OUTER JOIN vamseepr2.College C
                    ON P.CollegeId = C.CollegeId
                    WHERE PS.IsMajor = 'Y'

-- End of views (Vamsee)


/*************************************************************************************************/

5. -- Stored Procedures (Vamsee)

/*
This procedure generates the bill for a student based on the number of credits enrolled for.Only a financier can ↙
    generate the bill.
*/

CREATE PROCEDURE vamseepr2.USPBill
    --Parameters
    @financierId INT = NULL,
    @studentId INT = NULL,
    @creditFee DECIMAL(6,2) = NULL
AS
```

```
-- Variable
DECLARE @isFinancier INT, @isStudent INT, @billNew DECIMAL(8,2), @credits INT, @billOld DECIMAL(8,2)

-- @isFaculty is 0 if no record is found
SELECT @isFinancier = ( SELECT COUNT(E.EmployeeId)
                            FROM vamseepr2.Employees E
                                INNER JOIN vamseepr2.JobInformation JI
                                ON E.JobId = JI.JobId
                                    INNER JOIN vamseepr2.JobPositionType JP
                                    ON JI.JobPositionTypeId = JP.JobPositionTypeId
                                        WHERE JP.PostionType = 'FINANCE' AND E.EmployeeId = @financierId)

-- @isStudent is 0 if no record is found
SELECT @isStudent = ( SELECT COUNT(S.StudentId)
                            FROM vamseepr2.Students S
                                WHERE S.StudentId = @studentId)

IF @isFinancier = 0
BEGIN
        PRINT 'Error: Cannot generate bill, not a valid financier id'
        RETURN 0
END

ELSE
BEGIN
    IF @isStudent = 0
    BEGIN
        PRINT 'Error: Cannot generate grade report, not a valid student id'
        RETURN 0
    END

    ELSE
    BEGIN
        --Generate the bill depedning the number of credits and credit fee
        SELECT @billOld = (SELECT S.BillAmount
                            FROM vamseepr2.Students S
                                WHERE S.StudentId = @studentId)

        SELECT @credits = ( SELECT SUM(C.Credits)
                            FROM vamseepr2.Students S
                                INNER JOIN vamseepr2.Enrollment E
                                ON S.StudentId = E.StudentId
                                    INNER JOIN vamseepr2.CourseSchedule CS
                                    ON E.CourseScheduleId = CS.CourseScheduleId
                                        INNER JOIN vamseepr2.Courses C
                                        ON CS.CourseCode = C.CourseCode AND CS.CourseNumber = C.CourseNumber
                                            WHERE S.StudentId = @studentId
                                            GROUP BY S.StudentId)

        SET @billNew = @credits * @creditFee;

        UPDATE vamseepr2.Students
            SET BillAmount = @billNew
                WHERE StudentId = @studentId;

        IF @billOld = 0.0
        BEGIN
            PRINT 'Success'
            RETURN 1
        END
        ELSE
        BEGIN
            PRINT 'Success! Updated the bill to ' + @billNew
            RETURN 1
        END
    END
END
-----------------------------------------------------------------------------------------------------------------

--End of stored procedures (Vamsee)
```

```
/*****************************************************************************************************/

5. --User-defined Functions (Vamsee)

/*
This function generates the grade report for a student.Only a faculty can generate the report.
*/

CREATE FUNCTION vamseepr2.UFGradeReport
(   --Parameters
    @facultyId INTEGER = NULL,
    @studentId INTEGER = NULL
)
RETURNS @gradeTable TABLE(StudentName NVARCHAR(100), Course NVARCHAR(100), Grade CHAR(1))
AS
BEGIN
-- Variable
DECLARE @isFaculty INTEGER, @isStudent INTEGER

-- @isFaculty is 0 if no record is found
SELECT @isFaculty = ( SELECT COUNT(E.EmployeeId)
                            FROM vamseepr2.Employees E
                                INNER JOIN vamseepr2.JobInformation JI
                                ON E.JobId = JI.JobId
                                    INNER JOIN vamseepr2.JobPositionType JP
                                    ON JI.JobPositionTypeId = JP.JobPositionTypeId
                                        WHERE JP.PostionType = 'FACULTY' AND E.EmployeeId = @facultyId)

-- @isStudent is 0 if no record is found
SELECT @isStudent = ( SELECT COUNT(S.StudentId)
                            FROM vamseepr2.Students S
                                WHERE S.StudentId = @studentId)
IF @isFaculty = 0
BEGIN
        INSERT INTO @gradeTable(StudentName) VALUES('Error! Cannot generate grade report, not a valid faculty id')
END

ELSE
BEGIN
    IF @isStudent = 0
    BEGIN
        INSERT INTO @gradeTable(StudentName) VALUES('Error: Cannot generate grade report, not a valid student id')
    END

    ELSE
    BEGIN
        --Generate the grade report of the student
        INSERT INTO @gradeTable
            SELECT S.FirstName+' '+S.LastName, C.CourseCode+''+CAST(C.CourseNumber AS NVARCHAR), CG.Grade
            FROM vamseepr2.Students S
                INNER JOIN vamseepr2.Enrollment E
                ON S.StudentId = E.StudentId
                    INNER JOIN vamseepr2.CourseSchedule CS
                    ON E.CourseScheduleId = CS.CourseScheduleId
                        INNER JOIN vamseepr2.Courses C
                        ON CS.CourseCode = C.CourseCode AND CS.CourseNumber = C.CourseNumber
                            INNER JOIN vamseepr2.CourseGrade CG
                            ON E.GradeId = CG.GradeId
                                WHERE S.StudentId = @studentId;
    END
END
RETURN
END

-----------------------------------------------------------------------------------------------------------


/*This function returns a list of suitable classrooms and the building for a course*/
```

```
CREATE FUNCTION vamseepr2.UFClassroomReport
(   --Parameters
    @courseCode     NVARCHAR(3) = NULL,
    @courseNumber   INTEGER     = NULL
)
RETURNS @classroomTable TABLE(RoomNumber NUMERIC(3,0), BuildingName NVARCHAR(50))
AS
BEGIN
-- Variable
DECLARE @courseSeats INTEGER

    SELECT @courseSeats = CS.CourseSeats
            FROM vamseepr2.CourseSchedule CS
            WHERE CS.CourseCode = @courseCode AND CS.CourseNumber = @courseNumber;

    INSERT INTO @classroomTable
        SELECT C.RoomNumber, B.BuildingName FROM vamseepr2.Classroom C
                LEFT OUTER JOIN vamseepr2.Buildings B
                ON  C.BuildingId = B.BuildingId
                WHERE C.MaxSeating > @courseSeats;
    RETURN
END

-----------------------------------------------------------------------------------------------------------------------

/*
This function tells if a employee's pay is lower or higher than the average pay.
*/
CREATE FUNCTION vamseepr2.AveragePay
(--Parameters
    @employeeId INTEGER = NULL
)
RETURNS NVARCHAR(400)
BEGIN
DECLARE @pay DECIMAL(8,2), @sum DECIMAL(8,2), @averagePay DECIMAL(8,2), @totalEmps AS INTEGER, @empPay AS DECIMAL(8↙
    ,2), @result NVARCHAR(400)
    SET @sum = 0
    SET @averagePay = 0
    SET @TotalEmps = 0

    SET @empPay = (SELECT E.YearlyPay FROM vamseepr2.Employees E
                            WHERE E.EmployeeId = @employeeId)

    DECLARE AveragePay CURSOR
        FOR
        SELECT E.YearlyPay
            FROM vamseepr2.Employees E
                OPEN AveragePay
                    FETCH NEXT FROM AveragePay
                    INTO @pay
                    IF @pay IS NOT NULL
                    WHILE @@FETCH_STATUS = 0
                    BEGIN
                        SET @sum = @sum + @pay
                        SET @totalEmps = @totalEmps + 1
                        FETCH NEXT FROM AveragePay INTO @pay
                    END
                CLOSE AveragePay
    DEALLOCATE AveragePay

    SET @averagePay = @sum / @TotalEmps

    IF  @averagePay > @empPay
    BEGIN
        SET @result = 'Your pay is lower that the average pay of all employees'
    END
    ELSE
    BEGIN
        SET @result = 'Your pay is higher that the average pay of all employees'
    END
```

```
    RETURN @result
END

--End of function (Vamsee)


/*************************************************************************************************************/


/*
This trigger subtracts or adds the number of seats in a course
*/


CREATE TRIGGER vamseepr2.trCourseSeats
    ON vamseepr2.Enrollment AFTER INSERT AS
        BEGIN TRAN SeatUpdate
        DECLARE @courseS1 AS INTEGER
        DECLARE @courseS2 AS INTEGER
        DECLARE @seatCount AS INTEGER

            SELECT @courseS1 = (SELECT CourseScheduleId FROM INSERTED)
            SELECT @courseS2 = (SELECT CourseScheduleId FROM DELETED)

            IF @courseS1 IS NOT NULL
            BEGIN
                BEGIN TRY
                    SET @seatCount = (SELECT CS.CourseSeats
                                        FROM vamseepr2.CourseSchedule CS
                                        WHERE CS.CourseScheduleId = @courseS1)
                    IF @seatCount < 0
                    BEGIN
                         RAISERROR('Rolling back transaction, not enough seats', 16, 1);
                    END
                END TRY
                BEGIN CATCH
                    ROLLBACK TRAN SeatUpdate
                END CATCH;
                PRINT 'Insertion performed, trCourseSeats trigger executed'
                UPDATE vamseepr2.CourseSchedule
                    SET CourseSeats = CourseSeats - 1
                    WHERE CourseScheduleId = @courseS1
            END

            IF @courseS2 IS NOT NULL
            BEGIN
                PRINT 'Deletion performed, trCourseSeats trigger executed'
                UPDATE vamseepr2.CourseSchedule
                    SET CourseSeats = CourseSeats + 1
                    WHERE CourseScheduleId = @courseS2
            END
```