Pranit Kashiram Harekar SUID: 546527913 Syracuse University, NY

PROJECT 2 (CSE 581)

1 DESIGN MODIFICATIONS

There were few changes in the design of project 1. They are as follows,

1. Removed *'EmployeeStatus'* table as it can be generated in the employee table as a single record.
   E.g. `EmployeeStatus BIT NOT NULL        -- 1=Active OR 0=Inactive`
2. Removed *'JobType'* table as it can be replaced by IsUnion record in the JobInformation table.
   E.g. `IsUnion BIT NOT NULL               -- 1=Union OR 0=Not Union`
3. Merged '*CourseType'* Table in CourseInformation
4. Created separate table for '*Pre-requisites'.*
5. Removed extra table '*LocalAddress*', as it caused data duplication.
6. Merged '*Benefit and Coverage'* table,as it caused data duplication.

2 Complete SQL Code For University Database

```
/*-----------------------------------------------------------------------------
 CSE581 - Introduction to database management systems
-----------------------------------------------------------------------------
 Application:       Project #2. Implementation of "University Database",
                    data manipulation and creation of other database objects.
 Language:          SQL, Microsoft SQL Server 2014
 Platform:          HP Spectre x360-13-4103dx , Intel(R) Core(TM) i7-6500U
                    CPU @ 2.5GHz, Windows 10 Home
 Author:            Pranit Kashiram Harekar, SUID: 546527913, Syracuse University
                    (315) 450-3405, pharekar@syr.edu
 Source:            Prof. Dusan Palider, Syracuse University
                    Created using Vertabelo (http://vertabelo.com)
------------------------------------------------------------------------------*/



-- Tables in "University Database"

-- Table: Addresses
-- Description: This table stores addresses of the users.
CREATE TABLE pr2.Addresses (
    AddressID       INT          NOT NULL      IDENTITY(1, 1),
    Street1         VARCHAR(20)  NOT NULL,
    Street2         VARCHAR(20)  NULL,
    City            VARCHAR(20)  NOT NULL,
    States          VARCHAR(20)  NOT NULL,
    ZIP             INT          NOT NULL,
    CONSTRAINT AddressesPk PRIMARY KEY  (AddressID)
);




-- Table: AreaOfStudy
-- Description: This table combines colleges with different fields of studies.
CREATE TABLE pr2.AreaOfStudy (
    AreaOfStudyID   INT          NOT NULL      IDENTITY(1, 1),
    CollegeID       INT          NOT NULL,
    StudyTitle      VARCHAR(100) NOT NULL,
    CONSTRAINT AreaOfStudyPk PRIMARY KEY  (AreaOfStudyID)
);




-- Table: BenefitSelection
-- Description: This table provides benefit selection single/family/op-out.
CREATE TABLE pr2.BenefitSelection (
    BenefitSelectionID     INT          NOT NULL,
    BenefitSelection       TEXT         NOT NULL,
    CONSTRAINT BenefitSelectionPk PRIMARY KEY  (BenefitSelectionID)
);
```

```
-- Table: Benefits
-- Description: This table defines benefits in detail.
CREATE TABLE pr2.Benefits (
    BenefitID               INT                         NOT NULL      IDENTITY(1, 1),
    BenefitCost             MONEY                       NOT NULL,
    BenefitSelectionID      INT                         NOT NULL      DEFAULT 2,
    BenefitDescription      VARCHAR(100)                NOT NULL,
    CONSTRAINT BenefitsPk PRIMARY KEY  (BenefitID)
);




-- Table: Buildings
-- Description: This table stores buildings of University.
CREATE TABLE pr2.Buildings (
    Id                INT                 NOT NULL ,
    BuildingName      VARCHAR(20)         NOT NULL,
    CONSTRAINT BuildingsPk PRIMARY KEY  (Id)
);




-- Table: Classroom
-- Description: This table stores classroom information.
CREATE TABLE pr2.Classroom (
    ClassroomNumber         INT                  NOT NULL,
    BuildingsId             INT                  NOT NULL,
    MaximumSeating          INT                  NOT NULL,
    ProjectorInfo           INT                  NOT NULL,
    NumberOfWhiteBoards     INT                  NOT NULL      DEFAULT 2,
    AVEquipments            VARCHAR(20)          NOT NULL,
    CONSTRAINT ClassroomPk PRIMARY KEY  (ClassroomNumber)
);




-- Table: College
-- Description: This table stores college names.
CREATE TABLE pr2.College (
    CollegeID       INT           NOT NULL      IDENTITY(1, 1),
    CollegeName     TEXT          NOT NULL,
    CONSTRAINT CollegePk PRIMARY KEY  (CollegeID)
);




-- Table: CourseDuration
-- Description: This table stores course duration information.
CREATE TABLE pr2.CourseDuration (
    Id                            INT    NOT NULL      IDENTITY(1, 1),
    CourseScheduleID                INT    NOT NULL,
    WeekDaysId                      INT    NOT NULL,
    StartTime                       TIME   NOT NULL,
    EndTime                         TIME   NOT NULL,
    CONSTRAINT CourseDurationPk PRIMARY KEY  (Id)
);




-- Table: CourseEnrollment
-- Description: This table stores course enrollment information.
CREATE TABLE pr2.CourseEnrollment (
    EnrollmentID                    INT  NOT NULL IDENTITY(1, 1),
    CourseID                        INT  NOT NULL,
    StudentID                       INT  NOT NULL,
    StudentGradingStatus            INT  NOT NULL,
    LetterGradeId                   INT  NOT NULL,
       GradeValue                   INT  NOT NULL,
    CONSTRAINT CourseEnrollmentPk PRIMARY KEY  (EnrollmentID)
);




-- Table: CourseInformation
-- Description: This table stores courses information.
CREATE TABLE pr2.CourseInformation (
    CourseCode                      VARCHAR(6)          NOT NULL,
    CourseNumber                    INT                 NOT NULL ,
    CourseTitle                     VARCHAR(50)         NOT NULL,
    CourseDescription               VARCHAR(1000)       NOT NULL,
    CONSTRAINT CourseInformationPk PRIMARY KEY  (CourseCode,CourseNumber)
);




-- Table: CourseSchedule
-- Description: This table stores schedule of courses.
CREATE TABLE pr2.CourseSchedule (
    CourseScheduleID       INT                  NOT NULL      IDENTITY(1, 1),
    CourseCode             VARCHAR(6)           NOT NULL,
    CourseNumber           INT                  NOT NULL,
    FacultyID              INT                  NOT NULL,
    NumberOfSeats          INT                  NOT NULL,
    Location               INT                  NOT NULL,
    Semester               INT                  NOT NULL,
    CONSTRAINT CourseSchedulePk PRIMARY KEY  (CourseScheduleID)
);




-- Table: Employees
-- Description: This table stores employee information.
```

```
CREATE TABLE pr2.Employees (
    EmployeeID              INT            NOT NULL      IDENTITY(1, 1),
    PersonID                INT            NOT NULL,
    YearlyPay               MONEY          NOT NULL,
    JobInformationID        INT            NOT NULL,
    HealthBenefits          INT            NOT NULL,
    DentalBenefits          INT            NOT NULL,
    VisionBenefits          INT            NOT NULL,
    EmployeeStatus          BIT            NOT NULL,
    CONSTRAINT EmployeesPk PRIMARY KEY  (EmployeeID)
);




-- Table: Grades
-- Description: This table maps grade values to lettergrades.
CREATE TABLE pr2.LetterGrade (
       LetterGradeId        INT            NOT NULL      IDENTITY(1,1),
    LetterGrades            VARCHAR(1)     NOT NULL,
    Description             VARCHAR(20)    NOT NULL,
    CONSTRAINT GradesPk PRIMARY KEY  (LetterGradeId)
);




-- Table: JobInformation
-- Description: This table stores job details.
CREATE TABLE pr2.JobInformation (
    JobInformationID               INT                 NOT NULL,
    JobDescription                 VARCHAR(100)        NOT NULL,
    JobRequirements                VARCHAR(100)        NULL,
    MinPay                         MONEY               NOT NULL,
    MaxPay                         MONEY               NOT NULL,
    IsUnionJob                     BIT                 NOT NULL,
    CONSTRAINT JobInformationPk PRIMARY KEY  (JobInformationID)
);




-- Table: Person
-- Description: This table stores personal information.
CREATE TABLE pr2.Person (
    PersonID                INT                  NOT NULL      IDENTITY(1, 1),
    NTID                    VARCHAR(10)          NOT NULL,
    FirstName               VARCHAR(20)          NOT NULL,
    LastName                VARCHAR(20)          NOT NULL,
    UserPassword            VARCHAR(20)          NULL,
    SSN                     BIGINT               NOT NULL,
    DateOfBirth             DATE                 NOT NULL,
    HomeAddressID           INT                  NOT NULL,
    LocalAddressID          INT                  NOT NULL,
       EmailAddress         VARCHAR(50)          NOT NULL
       CHECK (EmailAddress Like '_%@_%._%'),
       PhoneNumber          VARCHAR(12) NULL
       CHECK (PhoneNumber Like'[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]'),
    CONSTRAINT PersonPk PRIMARY KEY  (PersonID)
);




-- Table: Prerequisites
-- Description: This table gives prerequisite courses.
CREATE TABLE pr2.Prerequisites (
    PrereqID                INT                 NOT NULL      IDENTITY(1, 1),
    ParentCode              VARCHAR(6)          NOT NULL,
    ParentNumber            INT                 NOT NULL,
    ChildCode               VARCHAR(6)          NOT NULL,
    ChildNumber             INT                 NOT NULL,
    CONSTRAINT PrerequisitesPk PRIMARY KEY  (PrereqID)
);




-- Table: Projector
-- Description: This table stores projector information.
CREATE TABLE pr2.Projector (
    ProjectorID             INT          NOT NULL      IDENTITY(1, 1),
    ProjectorText           TEXT         NOT NULL,
    CONSTRAINT ProjectorPk PRIMARY KEY  (ProjectorID)
);




-- Table: Semester
-- Description: This table stores semesters fall/spring/summerI/summerII/combined summer
CREATE TABLE pr2.Semester (
    Id               INT                 NOT NULL,
    Semester         VARCHAR(20)         NOT NULL,
    CONSTRAINT SemesterPk PRIMARY KEY  (Id)
);




-- Table: SemesterInformation
-- Description: This table stores semester information in detail.
CREATE TABLE pr2.SemesterInformation (
    SemesterID      INT           NOT NULL      IDENTITY(1, 1),
    Semester        INT           NOT NULL,
    ForYear         INT           NOT NULL,
    StartDate       DATE          NOT NULL,
    EndDate         DATE          NOT NULL,
    CONSTRAINT SemesterInformationPk PRIMARY KEY  (SemesterID)
);
```

```
-- Table: StudentAreaOfStudy
-- Description: This table provides student's field of study.
CREATE TABLE pr2.StudentAreaOfStudy (
    StudentStudyID   INT  NOT NULL IDENTITY(1, 1),
    StudentID        INT  NOT NULL,
    AreaOfStudyID    INT  NOT NULL ,
    IsMajor          BIT  NOT NULL,
    CONSTRAINT StudentAreaOfStudyPk PRIMARY KEY  (StudentStudyID)
);




-- Table: StudentGradingStatus
-- Description: This table provides student's grading status (regular/pass/fail/audit).
CREATE TABLE pr2.StudentGradingStatus (
    StudentGradingStatusID        INT          NOT NULL,
    GradingStatus                 TEXT         NOT NULL,
    CONSTRAINT StudentGradingStatusPk PRIMARY KEY  (StudentGradingStatusID)
);




-- Table: StudentInfo
-- Description: This table stores student's information.
CREATE TABLE pr2.StudentInfo (
    StudentID        INT  NOT NULL IDENTITY(1, 1),
    PersonID         INT  NOT NULL,
    StudentStatusID INT  NOT NULL,
    CONSTRAINT StudentInfoPk PRIMARY KEY  (StudentID)
);




-- Table: StudentStatus
-- Description: This table provides student's status (Undergraduate/Graduate/Non-
Matriculate/Graduated).
CREATE TABLE pr2.StudentStatus (
    StudentStatusID INT          NOT NULL,
    StudentStatus    TEXT         NOT NULL,
    CONSTRAINT StudentStatusPk PRIMARY KEY  (StudentStatusID)
);




-- Table: WeekDays
-- Description: This table gives weekdays(Mon/Tue/Wen/Thu/Fri/Sat/Sun).
CREATE TABLE pr2.WeekDays (
    Id        INT           NOT NULL ,
    DayText  TEXT           NOT NULL,
    CONSTRAINT WeekDaysPk PRIMARY KEY  (Id)
);


-- foreign keys
-- Reference:  AreaOfStudy (table: StudentAreaOfStudy)


ALTER TABLE pr2.StudentAreaOfStudy ADD CONSTRAINT AreaOfStudyForStudents
    FOREIGN KEY (AreaOfStudyID)
    REFERENCES pr2.AreaOfStudy (AreaOfStudyID)
;

-- Reference:  BenefitSelection (table: Benefits)


ALTER TABLE pr2.Benefits ADD CONSTRAINT SelectionOfBenefit
    FOREIGN KEY (BenefitSelectionID)
    REFERENCES pr2.BenefitSelection (BenefitSelectionID)
;

-- Reference:  ChildCourseInfo (table: Prerequisites)


ALTER TABLE pr2.Prerequisites ADD CONSTRAINT ChildCourseInfo
    FOREIGN KEY (ChildCode,ChildNumber)
    REFERENCES pr2.CourseInformation (CourseCode,CourseNumber)
;

-- Reference:  ClassroomBuildings (table: Classroom)


ALTER TABLE pr2.Classroom ADD CONSTRAINT ClassroomBuildings
    FOREIGN KEY (BuildingsId)
    REFERENCES pr2.Buildings (Id)
;

-- Reference:  ClassroomProjectorInfo (table: Classroom)


ALTER TABLE pr2.Classroom ADD CONSTRAINT ClassroomProjectorInfo
    FOREIGN KEY (ProjectorInfo)
    REFERENCES pr2.Projector (ProjectorID)
;

-- Reference:  CollegeForParticularAreaOfStudy (table: AreaOfStudy)


ALTER TABLE pr2.AreaOfStudy ADD CONSTRAINT CollegeForParticularAreaOfStudy
    FOREIGN KEY (CollegeID)
    REFERENCES pr2.College (CollegeID)
;

-- Reference:  CourseEnrollmentGrades (table: CourseEnrollment)


ALTER TABLE pr2.CourseEnrollment ADD CONSTRAINT CourseEnrollmentGrades
    FOREIGN KEY (LetterGradeId)
    REFERENCES pr2.LetterGrade (LetterGradeId)
;
```

```
-- Reference:  CourseIdforCourseEnrollment (table: CourseEnrollment)


ALTER TABLE pr2.CourseEnrollment ADD CONSTRAINT CourseIdforCourseEnrollment
    FOREIGN KEY (CourseID)
    REFERENCES pr2.CourseSchedule (CourseScheduleID)
;

-- Reference:  CourseSchedule (table: CourseSchedule)


ALTER TABLE pr2.CourseSchedule ADD CONSTRAINT Schedule
    FOREIGN KEY (CourseCode,CourseNumber)
    REFERENCES pr2.CourseInformation (CourseCode,CourseNumber)
;

-- Reference:  DailyCourseSchedule (table: CourseDuration)


ALTER TABLE pr2.CourseDuration ADD CONSTRAINT DailyCourseSchedule
    FOREIGN KEY (CourseScheduleID)
    REFERENCES pr2.CourseSchedule (CourseScheduleID)
;

-- Reference:  DayOfWeekForDailySchedule (table: CourseDuration)


ALTER TABLE pr2.CourseDuration ADD CONSTRAINT DayOfWeekForDailySchedule
    FOREIGN KEY (WeekDaysId)
    REFERENCES pr2.WeekDays (Id)
;

-- Reference:  DentalBenefits (table: Employees)


ALTER TABLE pr2.Employees ADD CONSTRAINT DentalBenefits
    FOREIGN KEY (DentalBenefits)
    REFERENCES pr2.Benefits (BenefitID)
;

-- Reference:  EmployeeInfoPeople (table: Employees)


ALTER TABLE pr2.Employees ADD CONSTRAINT EmployeeInfoPeople
    FOREIGN KEY (PersonID)
    REFERENCES pr2.Person (PersonID)
;

-- Reference:  EmployeeJobInfo (table: Employees)


ALTER TABLE pr2.Employees ADD CONSTRAINT EmployeeJobInfo
    FOREIGN KEY (JobInformationID)
    REFERENCES pr2.JobInformation (JobInformationID)
;

-- Reference:  Faculty (table: CourseSchedule)


ALTER TABLE pr2.CourseSchedule ADD CONSTRAINT Faculty
```

```
    FOREIGN KEY (FacultyID)
    REFERENCES pr2.Employees (EmployeeID)
;

-- Reference:  HealthBenefits (table: Employees)


ALTER TABLE pr2.Employees ADD CONSTRAINT HealthBenefits
    FOREIGN KEY (HealthBenefits)
    REFERENCES pr2.Benefits (BenefitID)
;

-- Reference:  HomeAddress (table: Person)


ALTER TABLE pr2.Person ADD CONSTRAINT HomeAddress
    FOREIGN KEY (HomeAddressID)
    REFERENCES pr2.Addresses (AddressID)
;

-- Reference:  LocalAddress (table: Person)


ALTER TABLE pr2.Person ADD CONSTRAINT LocalAddress
    FOREIGN KEY (LocalAddressID)
    REFERENCES pr2.Addresses (AddressID)
;

-- Reference:  Location (table: CourseSchedule)


ALTER TABLE pr2.CourseSchedule ADD CONSTRAINT Location
    FOREIGN KEY (Location)
    REFERENCES pr2.Classroom (ClassroomNumber)
;

-- Reference:  ParentCourseInfo (table: Prerequisites)


ALTER TABLE pr2.Prerequisites ADD CONSTRAINT ParentCourseInfo
    FOREIGN KEY (ParentCode,ParentNumber)
    REFERENCES pr2.CourseInformation (CourseCode,CourseNumber)
;

-- Reference:  SemesterInfoForCourseSchedule (table: CourseSchedule)


ALTER TABLE pr2.CourseSchedule ADD CONSTRAINT SemesterInfoForCourseSchedule
    FOREIGN KEY (Semester)
    REFERENCES pr2.SemesterInformation (SemesterID)
;

-- Reference:  SemesterName (table: SemesterInformation)


ALTER TABLE pr2.SemesterInformation ADD CONSTRAINT SemesterName
    FOREIGN KEY (Semester)
    REFERENCES pr2.Semester (Id)
;
```

```
-- Reference:  StudentAreaOfStudy (table: StudentAreaOfStudy)


ALTER TABLE pr2.StudentAreaOfStudy ADD CONSTRAINT StudentFieldOfStudy
    FOREIGN KEY (StudentID)
    REFERENCES pr2.StudentInfo (StudentID)
;

-- Reference:  StudentInfo (table: StudentInfo)


ALTER TABLE pr2.StudentInfo ADD CONSTRAINT StudentInformation
    FOREIGN KEY (PersonID)
    REFERENCES pr2.Person (PersonID)
;

-- Reference:  StudentInfoForCourseEnrollment (table: CourseEnrollment)


ALTER TABLE pr2.CourseEnrollment ADD CONSTRAINT StudentInfoForCourseEnrollment
    FOREIGN KEY (StudentID)
    REFERENCES pr2.StudentInfo (StudentID)
;

-- Reference:  StudentStatus (table: StudentInfo)


ALTER TABLE pr2.StudentInfo ADD CONSTRAINT StudentsStatus
    FOREIGN KEY (StudentStatusID)
    REFERENCES pr2.StudentStatus (StudentStatusID)
;

-- Reference:  StudentStatusCourseEnrollment (table: CourseEnrollment)


ALTER TABLE pr2.CourseEnrollment ADD CONSTRAINT StudentStatusCourseEnrollment
    FOREIGN KEY (StudentGradingStatus)
    REFERENCES pr2.StudentGradingStatus (StudentGradingStatusID)
;

-- Reference:  VisionBenefits (table: Employees)


ALTER TABLE pr2.Employees ADD CONSTRAINT VisionBenefits
    FOREIGN KEY (VisionBenefits)
    REFERENCES pr2.Benefits (BenefitID)
;


-- End of file.
```

3 Complete SQL Code for Data Loading

```
/*-----------------------------------------------------------------------------
 CSE581 - Introduction to database management systems
-------------------------------------------------------------------------------
 Application:       Project #2. Implementation of "University Database",
                    data manipulation and creation of other database objects.
 Language:          SQL, Microsoft SQL Server 2014
 Platform:          HP Spectre x360-13-4103dx , Intel(R) Core(TM) i7-6500U
                    CPU @ 2.5GHz, Windows 10 Home
 Author:            Pranit Kashiram Harekar, SUID: 546527913, Syracuse University
                    (315) 450-3405, pharekar@syr.edu
 Source:            Prof. Dusan Palider, Syracuse University
                    Created using Vertabelo (http://vertabelo.com)
-----------------------------------------------------------------------------*/


-- Data loading--
-- The objective of this SQL Code is to insert data into the tables of University Database.

INSERT INTO pr2.Addresses(Street1,Street2,City,States,ZIP)
       VALUES
              ('101 Ostrom Ave','Apartment 1','Syracuse','NY',13210),
              ('222 Madison street','Apt 4','Rochester','NY',11001),
              ('323 Roosevelt Ave','Apartment 6','Midland','TX',71901),
              ('172 Meadowbrook Dr','Apt 2','Springfield','IL',30781),
              ('500 Clarendon','S Crouse mansion','Fresno','CA',51005),
              ('301 Maple Street','Apartment 3','Phoenix','AZ',28807),
              ('605 Crawford Avenue','Apt 4','St Petersburg','FL',99010),
              ('230 Kensington Pl','WY Homes','Green Bay','WI',41829),
              ('124 Trinity Pl','House 2','Medford','OR',98237);


INSERT INTO pr2.BenefitSelection(BenefitSelectionID,BenefitSelection)
       VALUES
              (1,'Single'),
              (2,'Family'),
              (3,'Op-out');


INSERT INTO pr2.Benefits(BenefitCost,BenefitSelectionID,BenefitDescription)
       VALUES
              (7500,1,'Health benefit for single'),
              (15000,2,'Health benefit for family'),
              (1000,3,'Op-out, Health benefit by Max-life Insurance'),
              (2500,1,'Dental benefit for single'),
              (10000,2,'Dental benefit for family'),
              (750,3,'Op-out, Dental benefit by Apollo Dental Care'),
              (10000,1,'Vision benefits for single'),
              (20000,2,'Vision benefits for family'),
              (9550,3,'Op-out, Vision benefits by Angela Vision Care');


INSERT INTO pr2.Buildings(Id,BuildingName)
       VALUES
              (1,'Life-Sci Building'),
              (2,'Brockway Complex '),
              (3,'Crouse-Hinds Hall'),
              (4,'Flint Hall'),
              (5,'Haven Hall'),
              (6,'Hoople Ed Building'),
              (7,'Lyman Hall'),
              (8,'Carriage House'),
              (9,'Hawkins Building'),
              (10,'Physics Building');


INSERT INTO pr2.Projector(ProjectorText)
       VALUES
              ('PRJ101'),
              ('PRJ702'),
              ('PRJ106'),
              ('PRJ180'),
              ('PRJ104'),
              ('PRJ155'),
              ('PRJ343'),
              ('PRJ901');


INSERT INTO
pr2.Classroom(ClassroomNumber,BuildingsId,MaximumSeating,ProjectorInfo,NumberOfWhiteBoards,AVEquipments
)
       VALUES
              (201,4,55,6,2,'Computers'),
              (240,3,70,3,2,'Dual speakers'),
              (321,1,90,7,3,'Amplifiers'),
              (125,10,65,4,1,'Smart consoles'),
              (110,9,80,1,2,'OLED screen'),
              (301,7,50,5,1,'Microphones');


INSERT INTO pr2.College(CollegeName)
       VALUES
              ('Babson Medical College'),
              ('Smithdale law school'),
              ('Harrison Science College'),
              ('Hill Valley College'),
              ('Grand Lakes Management School'),
              ('Medlock School of Information'),
              ('Medfield dental college'),
              ('School of Education, Pendleton'),
              ('Worthington Art & science college');


INSERT INTO pr2.WeekDays(Id,DayText)
       VALUES
              (1,'Mon'),
              (2,'Tue'),
              (3,'Wed'),
              (4,'Thurs'),
              (5,'Fri'),
              (6,'Sat'),
              (7,'Sun');
```

```
INSERT INTO
pr2.Person(NTID,FirstName,LastName,UserPassword,SSN,DateOfBirth,HomeAddressID,LocalAddressID,EmailAddre
ss,PhoneNumber)
       VALUES
              ('JR73983031','Jessica','Reid','pqr001',2127997857,'1994-01-
04',6,1,'jreid14@uti.com','315-444-5556'),
              ('AT88108120','Alexandra','Terry','78gh45',3860731674,'1975-04-
11',4,2,'aterry@ymail.com','888-234-5102'),
              ('CP93205321','Cally','Potter','129fbn',5177870315,'1990-07-
20',5,7,'pottercally@sify.com','782-036-7190'),
              ('BH98679839','Bree','Hardy','atDemon290',2585954012,'1980-01-
24',8,3,'breehardy24@twc.com','508-590-3411'),
              ('JC63PNOD21','John','Cornor','3jk200',4793789451,'2000-04-
20',3,8,'johnx123@msn.com','438-923-9090'),
              ('JC74987935','Joe','Clemons','6opanm0',1274428191,'1996-09-
29',2,5,'clemonsjoe@gmail.com','966-327-8990'),
              ('KS74987935','Kendall','Stein','qwerty43',7914451113,'1984-02-
05',6,3,'kstein@hotmail.com','609-202-0786'),
              ('MB07154060','Marrisa','Bell','10nomp123',5954012070,'2003-06-
19',1,2,'marrisa12@mymail.com','315-450-4498'),
              ('RJ9998JK35','Rebecca','Jones','6angel23',4451791113,'1999-03-
08',8,5,'rhones@live.com','214-785-2911'),
              ('MD0715KLM0','Maria','Dsouza','batman44',9070954012,'2000-01-
30',1,3,'mariad@rediffmail.com','508-315-0899');


INSERT INTO pr2.Semester(Id,Semester)
       VALUES
              (1,'Fall'),
              (2,'Spring'),
              (3,'Summer I'),
              (4,'Summer II'),
              (5,'Combined Summer');


INSERT INTO pr2.CourseInformation(CourseCode,CourseNumber,CourseTitle,CourseDescription)
       VALUES
              ('CSE581',1,'Database Management','Database design and management'),
              ('ELE111',3,'Network Theory','Networks laws and theories'),
              ('GEO340',8,'Grography of Oil','Geological studies related to oil resources'),
              ('MAE486',2,'Fuel cell Science','Fuel cell thermodynamics'),
              ('MED503',6,'Hematopathology','Training in Hematology'),
              ('GEO125',4,'Evolution of Earth','Studies of origins and development of Earth'),
              ('TEL604',7,'Telecommunication','Studies of Telecommunication systems'),
              ('BCH621',5,'Adv Biochemistry','Advanced Biochemistry and Enzymology'),
              ('CIS385',3,'Operating Systems','Principles of Operating systems'),
              ('CHM202',5,'Basic Electrodes','Study of Electrodes');


INSERT INTO pr2.Prerequisites (ParentCode,ParentNumber,ChildCode,ChildNumber)
       VALUES
              ('CSE581',1,'CIS385',3),
              ('GEO340',8,'GEO125',4),
              ('MED503',6,'BCH621',5),
              ('MAE486',2,'CHM202',5),
              ('ELE111',3,'TEL604',7);


INSERT INTO
pr2.JobInformation(JobInformationID,JobDescription,JobRequirements,MinPay,MaxPay,IsUnionJob)
```

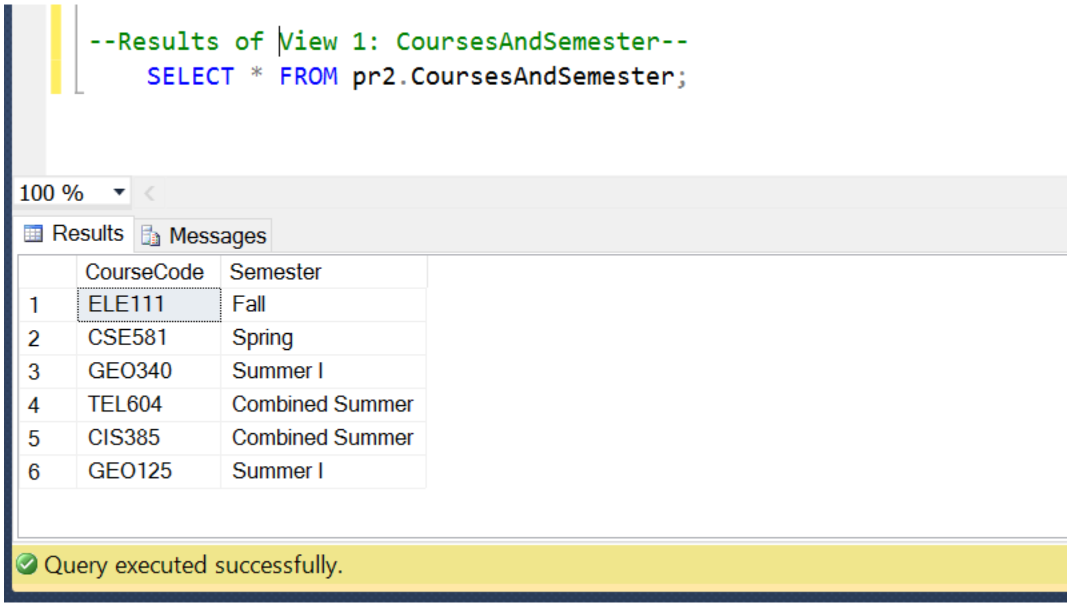
```
        VALUES
                (1,'Librarian','Master degree, 1 year experience',40000,60000,0),
                (2,'Administrative Assistant','Bachelors degree, 3 years experience',60000,80000,1),
                (3,'Associate Professor','PhD with 2 years experience',90000,120000,1),
                (4,'Job Recruiter','MBA with 5 years ecxperience',80000,100000,0),
                (5,'Database Manager','MS in computer science or equivalent degree',70000,100000,0),
                (6,'Desk attendant','Bachelors degree',40000,60000,0),
                (7,'Security Manager','5 years of experience',70000,80000,1),
                (8,'Food supervisor','NULL',30000,40000,0);


INSERT INTO
pr2.Employees(PersonID,YearlyPay,JobInformationID,HealthBenefits,DentalBenefits,VisionBenefits,Employee
Status)
        VALUES
                (1,80000,2,6,7,8,1),
                (4,100000,5,3,2,9,0),
                (5,60000,1,3,8,9,0),
                (2,120000,3,3,4,7,1),
                (3,120000,3,3,4,7,1),
                (8,100000,3,7,1,9,1);


INSERT INTO pr2.SemesterInformation(Semester,ForYear,StartDate,EndDate)
        VALUES
                (1,2015,'2015-01-09','2015-12-14'),
                (3,2015,'2015-01-05','2015-08-30'),
                (2,2015,'2015-01-20','2015-05-30'),
                (4,2015,'2015-05-05','2015-09-30'),
                (5,2015,'2015-01-05','2015-09-30'),
                (1,2016,'2016-01-09','2016-12-18'),
                (2,2016,'2016-01-19','2016-05-30');


INSERT INTO pr2.CourseSchedule(CourseCode,CourseNumber,FacultyID,NumberOfSeats,Location,Semester)
        VALUES
                ('ELE111',3,4,40,110,1),
                ('CSE581',1,6,70,321,3),
                ('GEO340',8,5,55,301,2),
                ('TEL604',7,5,25,240,5),
                ('CIS385',3,4,100,201,5),
                ('GEO125',4,6,60,125,2);


INSERT INTO pr2.CourseDuration(CourseScheduleID,WeekDaysId,StartTime,EndTime)
        VALUES
                (1,2,'7:00:00 AM','8:15:00 AM'),
                (2,1,'9:00:00 AM','10:30:00 AM'),
                (3,5,'8:30:00 AM','11:00:00 AM'),
                (5,3,'12:00:00 PM','1:45:00 PM'),
                (1,4,'2:00:00 PM','4:00:00 PM'),
                (2,5,'5:00:00 PM','6:30:00 PM'),
                (4,1,'6:00:00 PM','7:40:00 PM'),
                (4,6,'8:00:00 PM','9:15:00 PM');


INSERT INTO pr2.AreaOfStudy(CollegeID,StudyTitle)
        VALUES
                (1,'Medical Science'),
                (2,'Laws and management'),
                (3,'Biological science'),
                (4,'Geological Studies'),
                (5,'Business Administration'),
                (6,'Management of Information Science '),
                (7,'Dental science'),
                (8,'School of education'),
                (9,'Art & science');


INSERT INTO pr2.StudentStatus(StudentStatusID,StudentStatus)
        VALUES
                (1,'Undergraduate'),
                (2,'Graduate'),
                (3,'Non-matriculated'),
                (4,'Graduated');


INSERT INTO pr2.StudentInfo(PersonID,StudentStatusID)
        VALUES
                (10,3),
                (6,2),
                (9,4),
                (7,1);


INSERT INTO pr2.StudentAreaOfStudy(StudentID,AreaOfStudyID,IsMajor)
        VALUES
                (1,6,1),
                (4,2,0),
                (3,1,1),
                (2,7,1),
                (3,3,0),
                (4,4,1),
                (1,5,0),
                (2,8,0);


INSERT INTO pr2.LetterGrade(LetterGrades,Description)
        VALUES
                ('A', 'Outstanding'),
                ('B', 'Very good'),
                ('C', 'Good'),
                ('D', 'Medium'),
                ('E', 'Poor'),
                ('F', 'Failed');


INSERT INTO pr2.StudentGradingStatus(StudentGradingStatusID,GradingStatus)
        VALUES
                (1,'Regular'),
                (2,'Pass'),
                (3,'Fail'),
                (4,'Audit');


INSERT INTO pr2.CourseEnrollment(CourseID,StudentID,StudentGradingStatus,LetterGradeId,GradeValue)
        VALUES
                (2,4,1,3,80),
                (3,2,4,1,100),
                (6,1,2,2,95),
                (1,4,3,6,30),
                (4,3,1,4,60),
                (5,2,2,5,50);

-- End of file.
```

4 VIEWS

4.1 VIEW 1

SQL CODE:

```
--View 1: CoursesAndSemester--
--View description: This view displays courses and the corresponding semesters in which they are
offered.

CREATE VIEW CoursesAndSemester
AS
        SELECT CourseCode AS CourseCode, Semester.Semester AS Semester
        FROM CourseSchedule
        INNER JOIN SemesterInformation
        ON CourseSchedule.Semester=SemesterInformation.SemesterID
        Inner JOIN Semester
        On Semester.Id=SemesterInformation.Semester;
```



```
--Results of View 1: CoursesAndSemester--
    SELECT * FROM pr2.CoursesAndSemester;
```

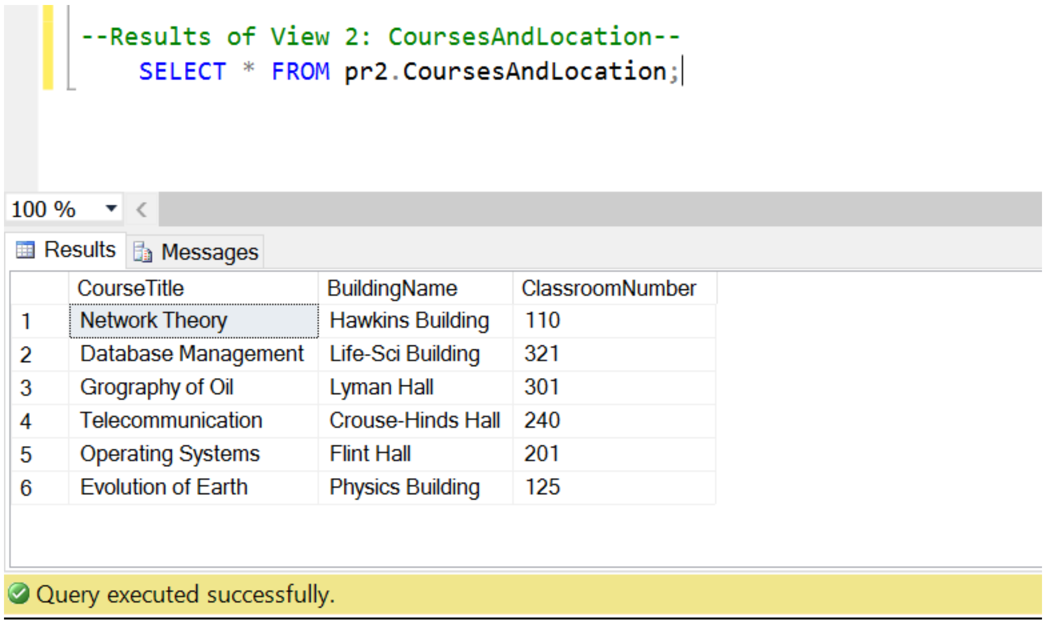
	CourseCode	Semester
1	ELE111	Fall
2	CSE581	Spring
3	GEO340	Summer I
4	TEL604	Combined Summer
5	CIS385	Combined Summer
6	GEO125	Summer I

Query executed successfully.

4.2 VIEW 2

SQL CODE:

```
--View 2: CoursesAndLocation--
--View description: This view displays courses and their location (i.e Building Name and Classroom
number)

CREATE VIEW CoursesAndLocation AS
SELECT CourseInformation.CourseTitle,Buildings.BuildingName,Classroom.ClassroomNumber
       FROM CourseInformation
       INNER JOIN CourseSchedule
       ON CourseSchedule.Coursecode=CourseInformation.CourseCode
       INNER JOIN Classroom
       ON Classroom.ClassroomNumber=CourseSchedule.Location
       INNER JOIN Buildings
       ON Classroom.BuildingsId=Buildings.Id;
```



```
--Results of View 2: CoursesAndLocation--
   SELECT * FROM pr2.CoursesAndLocation;
```

100 %

Results | Messages

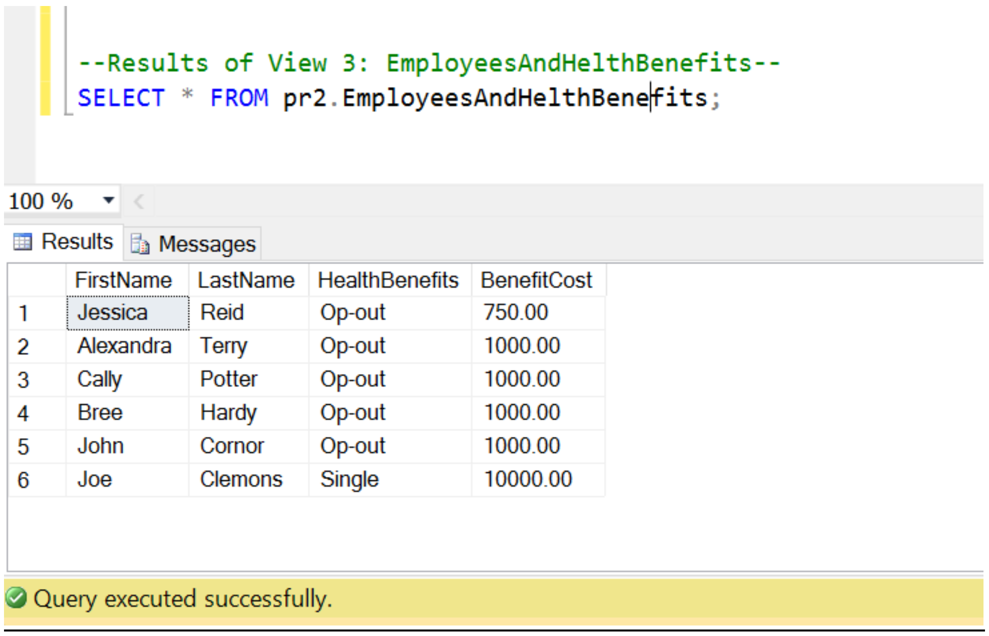
	CourseTitle	BuildingName	ClassroomNumber
1	Network Theory	Hawkins Building	110
2	Database Management	Life-Sci Building	321
3	Grography of Oil	Lyman Hall	301
4	Telecommunication	Crouse-Hinds Hall	240
5	Operating Systems	Flint Hall	201
6	Evolution of Earth	Physics Building	125

Query executed successfully.

4.3 VIEW 3

SQL CODE:

```
--View 3: EmployeesAndHelthBenefits--
--View description: This view provides employees and their health benefit information.

CREATE VIEW EmployeesAndHelthBenefits AS
SELECT FirstName,
              LastName,
              BenefitSelection AS HealthBenefits,
              BenefitCost
FROM Person
INNER JOIN Employees
ON Person.PersonID=Employees.EmployeeID
INNER JOIN Benefits
ON Benefits.BenefitID= Employees.HealthBenefits
INNER JOIN BenefitSelection
ON BenefitSelection.BenefitSelectionID=Benefits.BenefitSelectionID;
```



```
--Results of View 3: EmployeesAndHelthBenefits--
SELECT * FROM pr2.EmployeesAndHelthBenefits;
```

100 %

Results | Messages

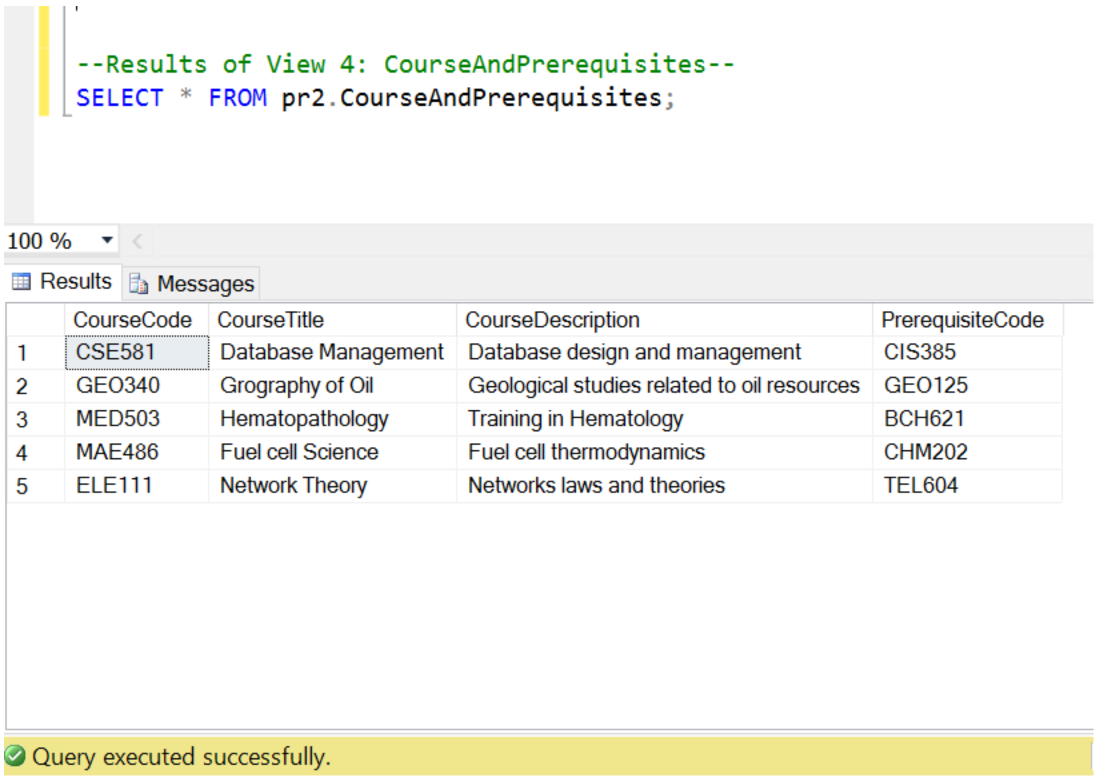
	FirstName	LastName	HealthBenefits	BenefitCost
1	Jessica	Reid	Op-out	750.00
2	Alexandra	Terry	Op-out	1000.00
3	Cally	Potter	Op-out	1000.00
4	Bree	Hardy	Op-out	1000.00
5	John	Cornor	Op-out	1000.00
6	Joe	Clemons	Single	10000.00

Query executed successfully.

4.4 VIEW 4

SQL CODE:

```
--View 4: CourseAndPrerequisites--
--View description: This view provides courses and their corresponding prerequisites.

CREATE VIEW CourseAndPrerequisites AS
SELECT CourseCode,
             CourseTitle,
             CourseDescription,
             Prerequisites.ChildCode AS PrerequisiteCode

FROM  CourseInformation
INNER JOIN  Prerequisites
ON Prerequisites.ParentCode=CourseInformation.CourseCode;
```



```
--Results of View 4: CourseAndPrerequisites--
SELECT * FROM pr2.CourseAndPrerequisites;
```

100 %

Results | Messages

	CourseCode	CourseTitle	CourseDescription	PrerequisiteCode
1	CSE581	Database Management	Database design and management	CIS385
2	GEO340	Grography of Oil	Geological studies related to oil resources	GEO125
3	MED503	Hematopathology	Training in Hematology	BCH621
4	MAE486	Fuel cell Science	Fuel cell thermodynamics	CHM202
5	ELE111	Network Theory	Networks laws and theories	TEL604

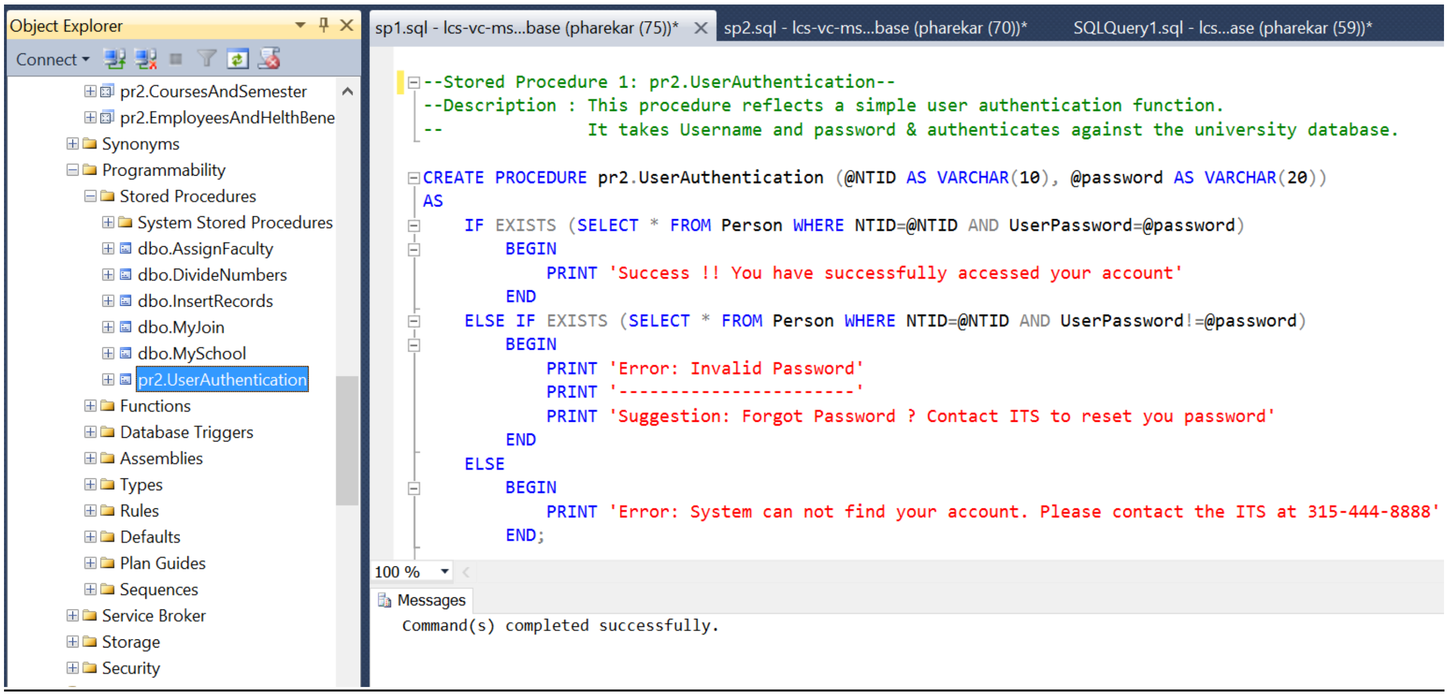
Query executed successfully.

5 Procedures And Functions

5.1 Stored Procedure 1

SQL CODE:

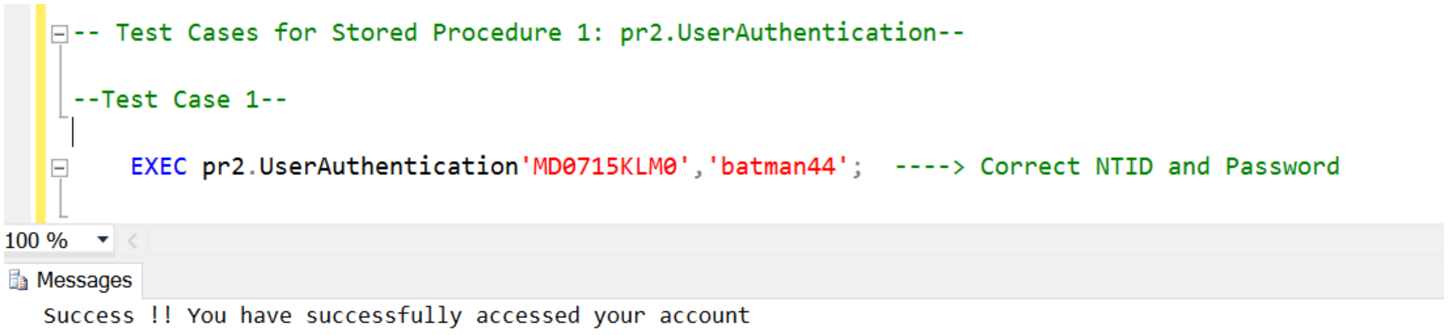
```
--Stored Procedure 1: pr2.UserAuthentication--
--Description: This procedure reflects a simple user authentication function.
--     It takes Username and password & authenticates against the university database.

CREATE PROCEDURE pr2.UserAuthentication (@NTID AS VARCHAR(10), @password AS VARCHAR(20))
AS
       IF EXISTS (SELECT * FROM Person WHERE NTID=@NTID AND UserPassword=@password)
              BEGIN
                     PRINT 'Success !! You have successfully accessed your account'
              END
       ELSE IF EXISTS (SELECT * FROM Person WHERE NTID=@NTID AND UserPassword!=@password)
              BEGIN
                     PRINT 'Error: Invalid Password'
                     PRINT '-----------------------'
                     PRINT 'Suggestion: Forgot Password ? Contact ITS to reset you password'
              END
       ELSE
              BEGIN
                   PRINT 'Error: System can not find your account. Please contact the ITS at 315-444-8888'
              END;
```



```
--Stored Procedure 1: pr2.UserAuthentication--
--Description : This procedure reflects a simple user authentication function.
--               It takes Username and password & authenticates against the university database.

CREATE PROCEDURE pr2.UserAuthentication (@NTID AS VARCHAR(10), @password AS VARCHAR(20))
AS
    IF EXISTS (SELECT * FROM Person WHERE NTID=@NTID AND UserPassword=@password)
        BEGIN
            PRINT 'Success !! You have successfully accessed your account'
        END
    ELSE IF EXISTS (SELECT * FROM Person WHERE NTID=@NTID AND UserPassword!=@password)
        BEGIN
            PRINT 'Error: Invalid Password'
            PRINT '----------------------'
            PRINT 'Suggestion: Forgot Password ? Contact ITS to reset you password'
        END
    ELSE
        BEGIN
            PRINT 'Error: System can not find your account. Please contact the ITS at 315-444-8888'
        END;
```

```
Command(s) completed successfully.
```

5.1.1 Test Cases

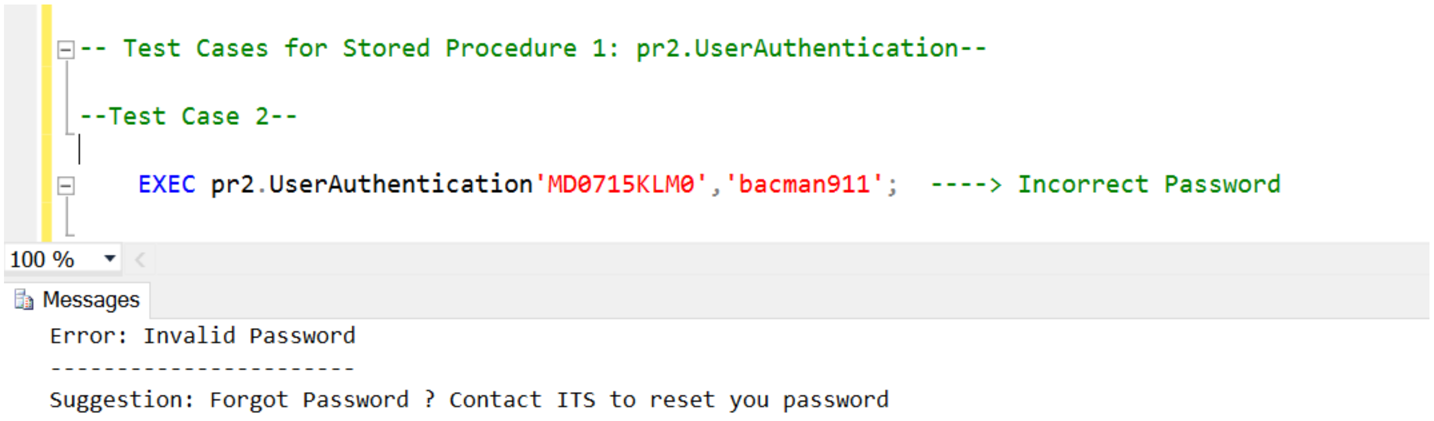


```
-- Test Cases for Stored Procedure 1: pr2.UserAuthentication--

--Test Case 1--

    EXEC pr2.UserAuthentication'MD0715KLM0','batman44';  ----> Correct NTID and Password
```

100 %

Messages

```
Success !! You have successfully accessed your account
```

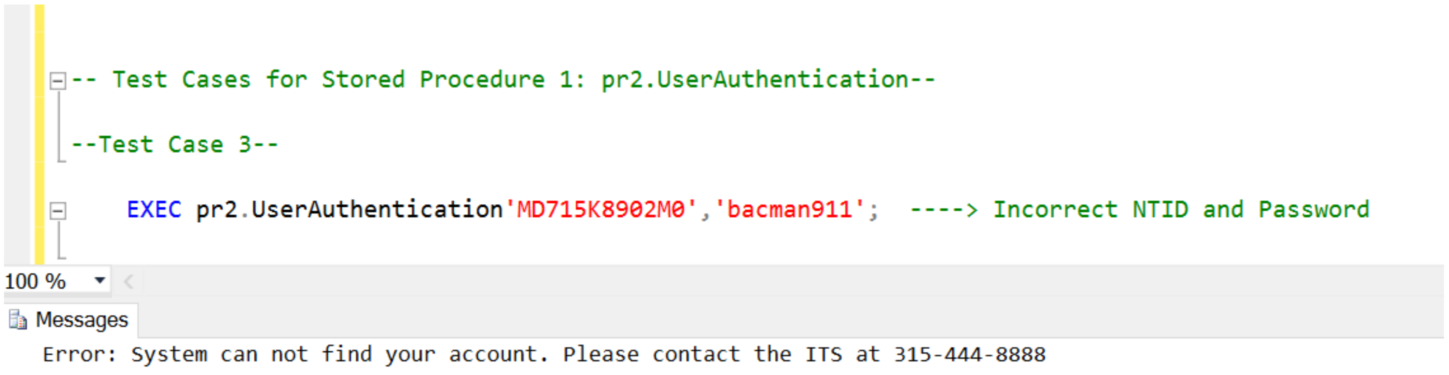


```
-- Test Cases for Stored Procedure 1: pr2.UserAuthentication--

--Test Case 2--

    EXEC pr2.UserAuthentication'MD0715KLM0','bacman911';  ----> Incorrect Password
```

100 %

Messages

```
Error: Invalid Password
-----------------------
Suggestion: Forgot Password ? Contact ITS to reset you password
```



```
-- Test Cases for Stored Procedure 1: pr2.UserAuthentication--

--Test Case 3--

    EXEC pr2.UserAuthentication'MD715K8902M0','bacman911';  ----> Incorrect NTID and Password
```

100 %

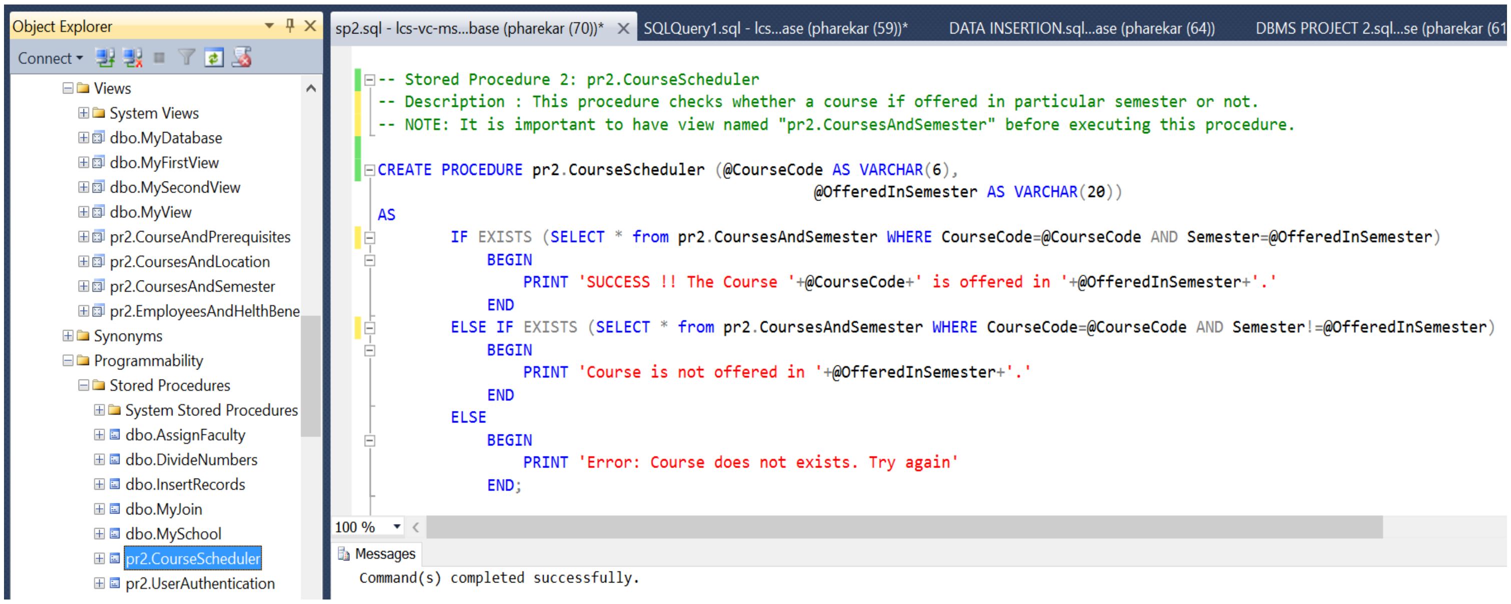
Messages

```
Error: System can not find your account. Please contact the ITS at 315-444-8888
```

5.2 Stored Procedure 2

SQL CODE:

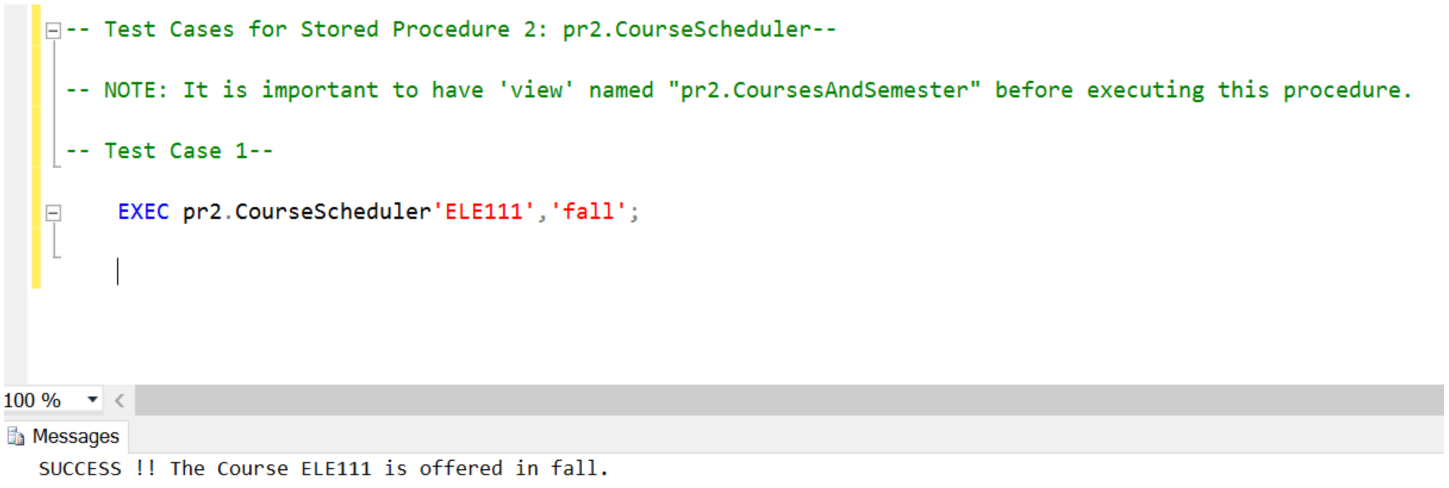
```
-- Stored Procedure 2: pr2.CourseScheduler
-- Description : This procedure checks whether a course if offered in particular semester or not.
-- NOTE: It is important to have view named "pr2.CoursesAndSemester" before executing this procedure. (i.e. View 1)

CREATE PROCEDURE pr2.CourseScheduler (@CourseCode AS VARCHAR(6),
                                                  @OfferedInSemester AS VARCHAR(20))
AS
       IF EXISTS (SELECT * from pr2.CoursesAndSemester
                     WHERE CourseCode=@CourseCode AND Semester=@OfferedInSemester)
              BEGIN
                     PRINT 'SUCCESS !! The Course '+@CourseCode+' is offered in '+@OfferedInSemester+'.'
              END
       ELSE IF EXISTS (SELECT * from pr2.CoursesAndSemester
                            WHERE CourseCode=@CourseCode AND Semester!=@OfferedInSemester)
              BEGIN
                     PRINT 'Course is not offered in '+@OfferedInSemester+'.'
              END
       ELSE
              BEGIN
                     PRINT 'Error: Course does not exists. Try again'
              END;
```



```
-- Stored Procedure 2: pr2.CourseScheduler
-- Description : This procedure checks whether a course if offered in particular semester or not.
-- NOTE: It is important to have view named "pr2.CoursesAndSemester" before executing this procedure.

CREATE PROCEDURE pr2.CourseScheduler (@CourseCode AS VARCHAR(6),
                                      @OfferedInSemester AS VARCHAR(20))
AS
        IF EXISTS (SELECT * from pr2.CoursesAndSemester WHERE CourseCode=@CourseCode AND Semester=@OfferedInSemester)
            BEGIN
                PRINT 'SUCCESS !! The Course '+@CourseCode+' is offered in '+@OfferedInSemester+'.'
            END
        ELSE IF EXISTS (SELECT * from pr2.CoursesAndSemester WHERE CourseCode=@CourseCode AND Semester!=@OfferedInSemester)
            BEGIN
                PRINT 'Course is not offered in '+@OfferedInSemester+'.'
            END
        ELSE
            BEGIN
                PRINT 'Error: Course does not exists. Try again'
            END;
```

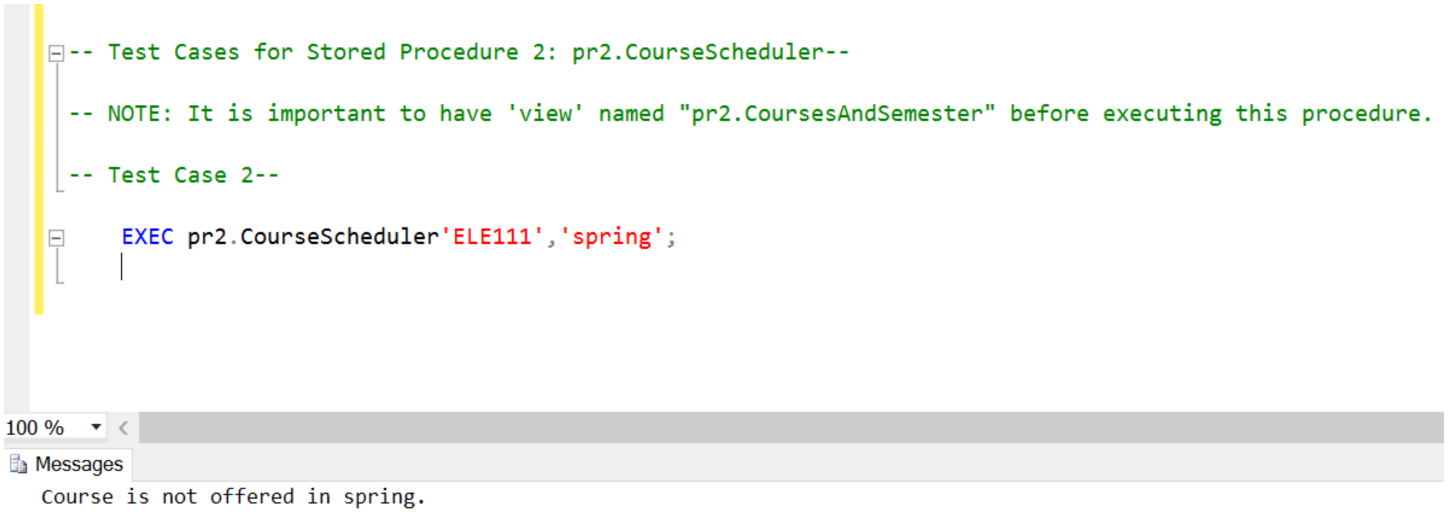
```
Command(s) completed successfully.
```

5.2.1 Test Cases



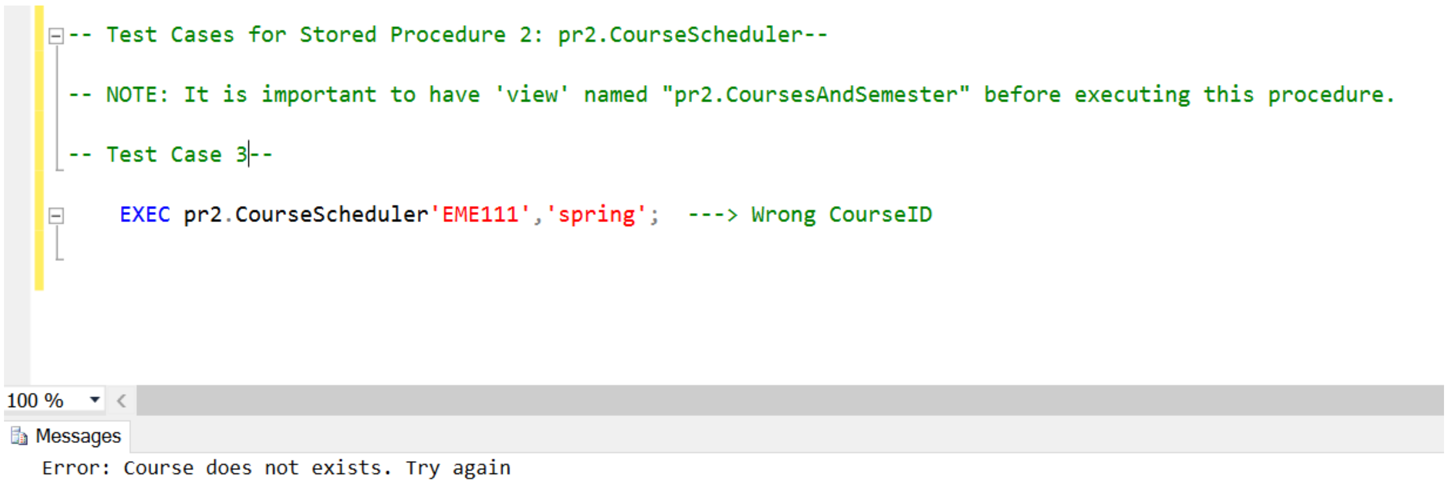
```
-- Test Cases for Stored Procedure 2: pr2.CourseScheduler--

-- NOTE: It is important to have 'view' named "pr2.CoursesAndSemester" before executing this procedure.

-- Test Case 1--

    EXEC pr2.CourseScheduler'ELE111','fall';
```

```
SUCCESS !! The Course ELE111 is offered in fall.
```



```
-- Test Cases for Stored Procedure 2: pr2.CourseScheduler--

-- NOTE: It is important to have 'view' named "pr2.CoursesAndSemester" before executing this procedure.

-- Test Case 2--

    EXEC pr2.CourseScheduler'ELE111','spring';
```

```
Course is not offered in spring.
```



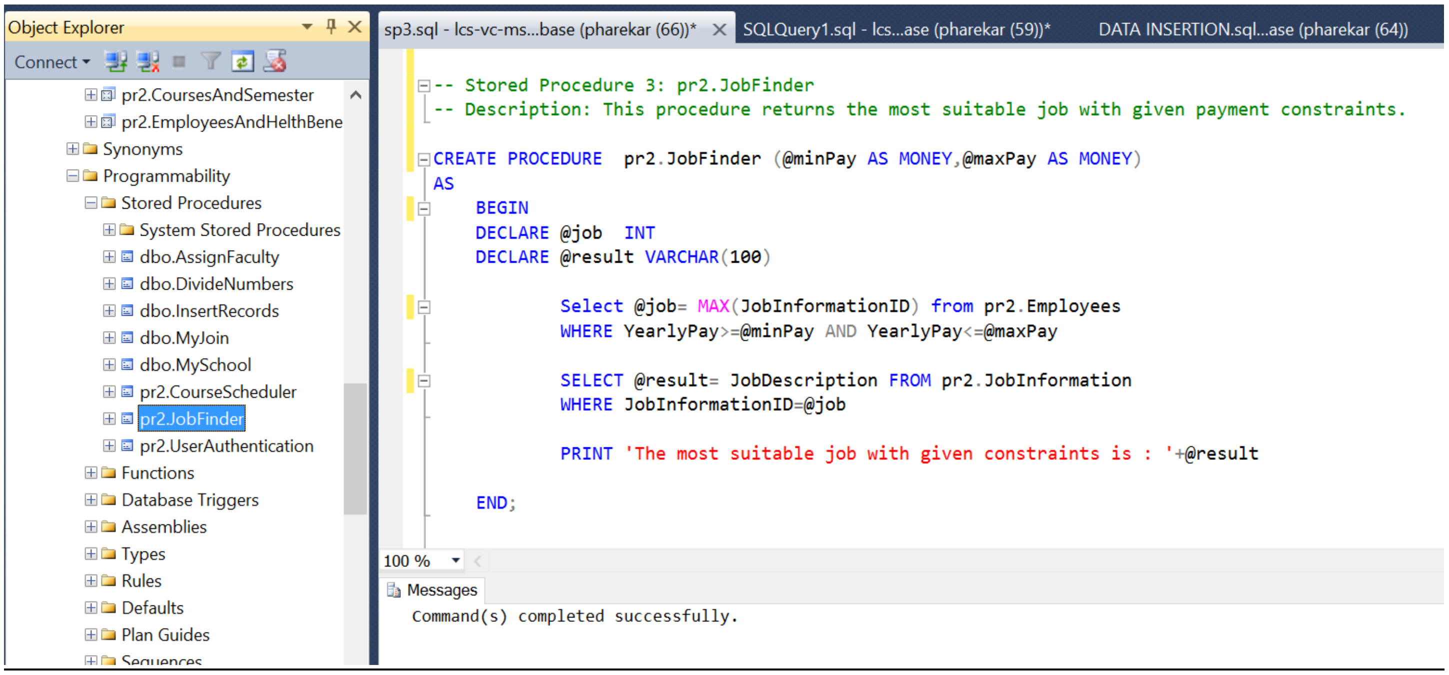
```
-- Test Cases for Stored Procedure 2: pr2.CourseScheduler--

-- NOTE: It is important to have 'view' named "pr2.CoursesAndSemester" before executing this procedure.

-- Test Case 3--

    EXEC pr2.CourseScheduler'EME111','spring';   ---> Wrong CourseID
```

```
Error: Course does not exists. Try again
```

5.3 Stored Procedure 3

SQL CODE:

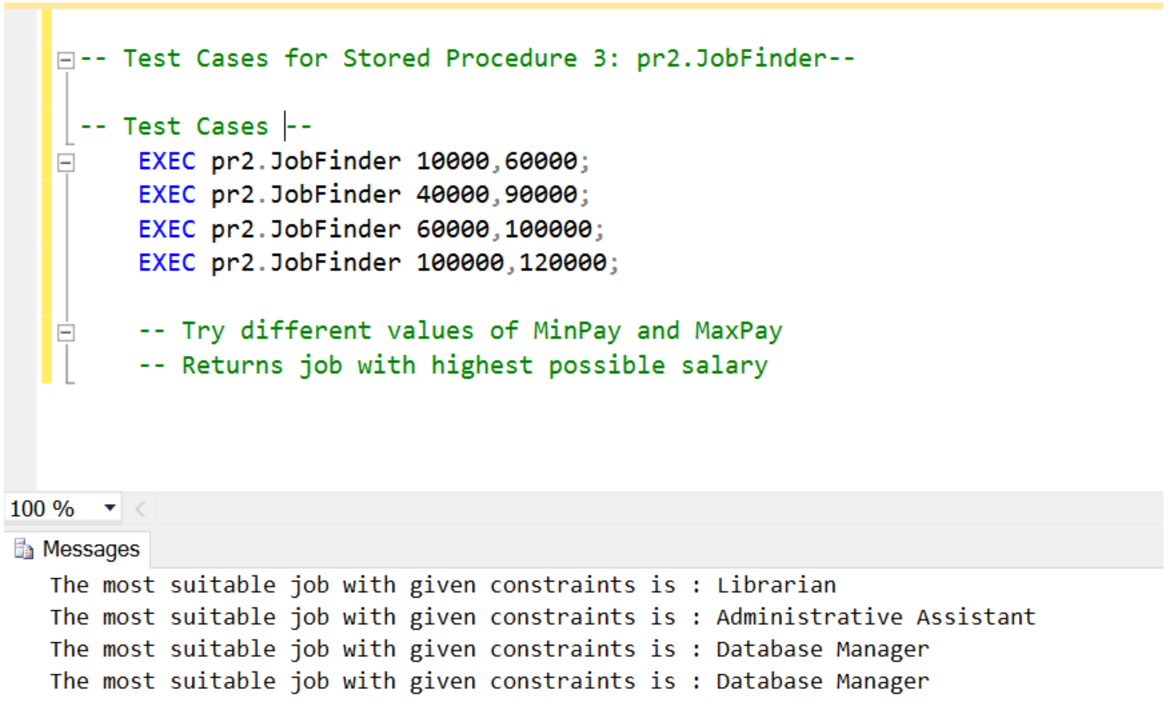
```
-- Stored Procedure 3: pr2.JobFinder
-- Description: This procedure returns the most suitable job with given payment constraints.

CREATE PROCEDURE  pr2.JobFinder (@minPay AS MONEY,@maxPay AS MONEY)
AS
       BEGIN
       DECLARE @job  INT
       DECLARE @result VARCHAR(100)

                     Select @job= MAX(JobInformationID) from pr2.Employees
                     WHERE YearlyPay>=@minPay AND YearlyPay<=@maxPay

                     SELECT @result= JobDescription FROM pr2.JobInformation
                     WHERE JobInformationID=@job

                     PRINT 'The most suitable job with given constraints is : '+@result

       END;
```



```
-- Stored Procedure 3: pr2.JobFinder
-- Description: This procedure returns the most suitable job with given payment constraints.

CREATE PROCEDURE  pr2.JobFinder (@minPay AS MONEY,@maxPay AS MONEY)
AS
       BEGIN
       DECLARE @job  INT
       DECLARE @result VARCHAR(100)

                     Select @job= MAX(JobInformationID) from pr2.Employees
                     WHERE YearlyPay>=@minPay AND YearlyPay<=@maxPay

                     SELECT @result= JobDescription FROM pr2.JobInformation
                     WHERE JobInformationID=@job

                     PRINT 'The most suitable job with given constraints is : '+@result

       END;
```

```
Command(s) completed successfully.
```

5.3.1 Test Cases



```
-- Test Cases for Stored Procedure 3: pr2.JobFinder--

-- Test Cases --
    EXEC pr2.JobFinder 10000,60000;
    EXEC pr2.JobFinder 40000,90000;
    EXEC pr2.JobFinder 60000,100000;
    EXEC pr2.JobFinder 100000,120000;

    -- Try different values of MinPay and MaxPay
    -- Returns job with highest possible salary
```

100 %

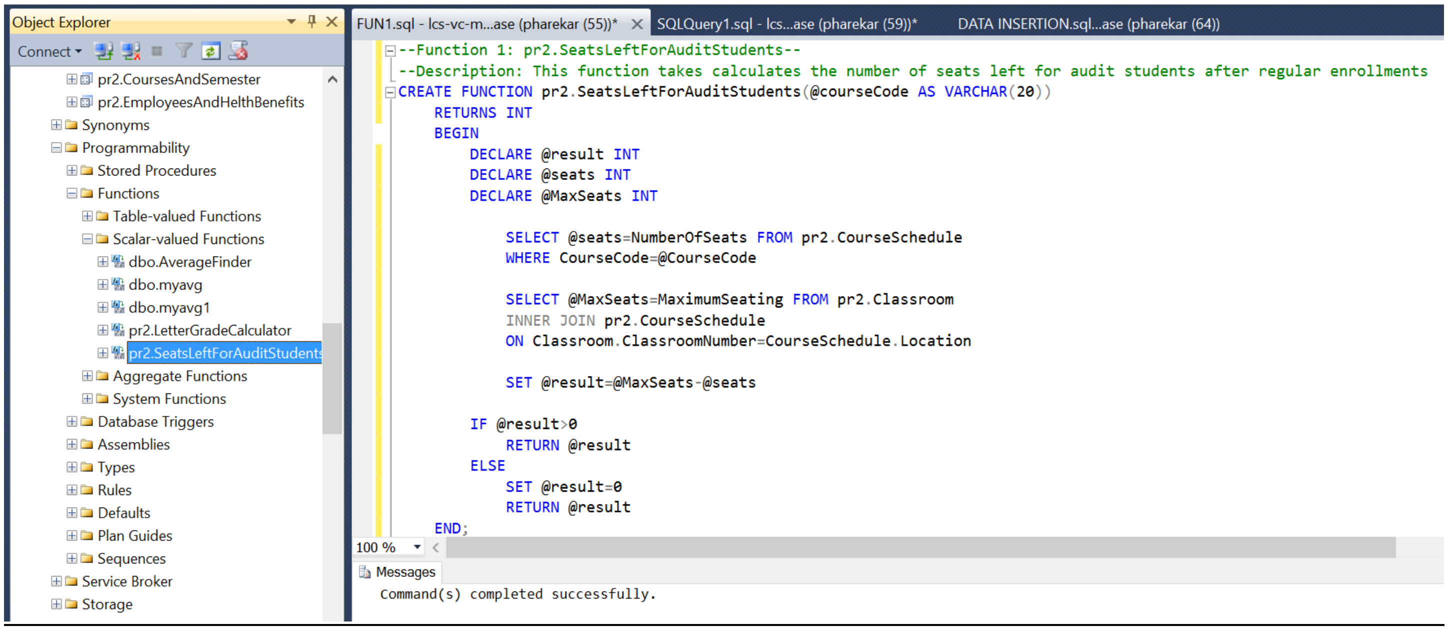
Messages

```
The most suitable job with given constraints is : Librarian
The most suitable job with given constraints is : Administrative Assistant
The most suitable job with given constraints is : Database Manager
The most suitable job with given constraints is : Database Manager
```

5.4 FUNCTION 1

<u>SQL CODE</u>

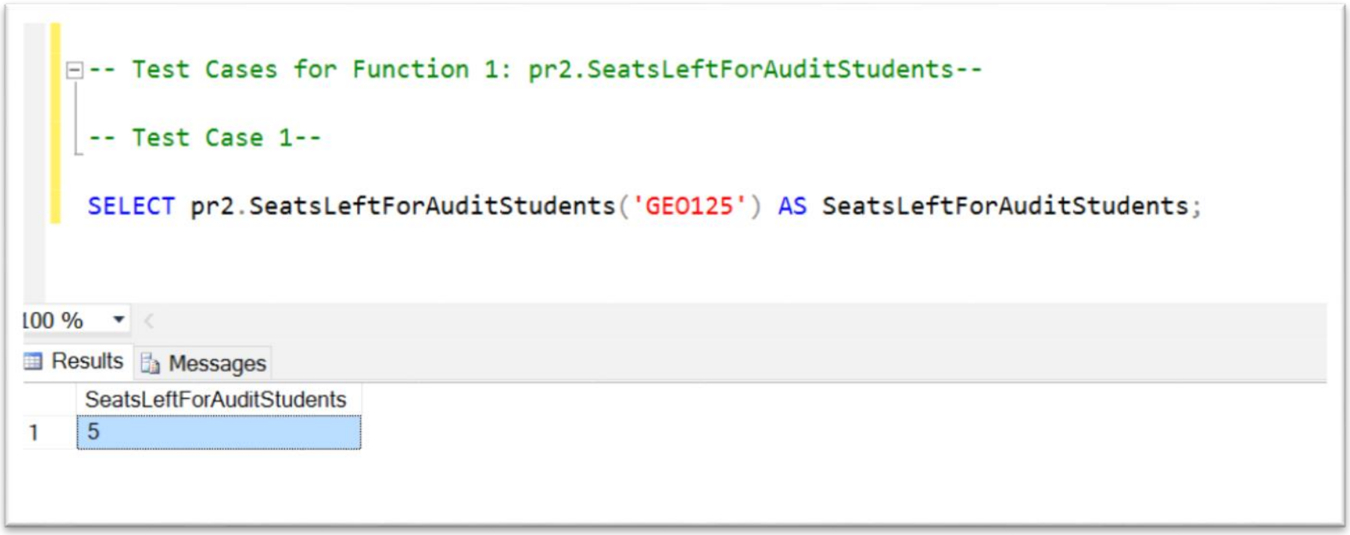
```
--Function 1: pr2.SeatsLeftForAuditStudents--
--Description: This function takes calculates the number of seats left for audit students after regular
enrollments
CREATE FUNCTION pr2.SeatsLeftForAuditStudents(@courseCode AS VARCHAR(20))
	RETURNS INT
	BEGIN
		DECLARE @result INT
		DECLARE @seats INT
		DECLARE @MaxSeats INT

			SELECT @seats=NumberOfSeats FROM pr2.CourseSchedule
			WHERE CourseCode=@CourseCode

			SELECT @MaxSeats=MaximumSeating FROM pr2.Classroom
			INNER JOIN pr2.CourseSchedule
			ON Classroom.ClassroomNumber=CourseSchedule.Location

			SET @result=@MaxSeats-@seats

		IF @result>0
			RETURN @result
		ELSE
			SET @result=0
			RETURN @result
	END;
```



```
--Function 1: pr2.SeatsLeftForAuditStudents--
--Description: This function takes calculates the number of seats left for audit students after regular enrollments
CREATE FUNCTION pr2.SeatsLeftForAuditStudents(@courseCode AS VARCHAR(20))
	RETURNS INT
	BEGIN
		DECLARE @result INT
		DECLARE @seats INT
		DECLARE @MaxSeats INT

			SELECT @seats=NumberOfSeats FROM pr2.CourseSchedule
			WHERE CourseCode=@CourseCode

			SELECT @MaxSeats=MaximumSeating FROM pr2.Classroom
			INNER JOIN pr2.CourseSchedule
			ON Classroom.ClassroomNumber=CourseSchedule.Location

			SET @result=@MaxSeats-@seats

		IF @result>0
			RETURN @result
		ELSE
			SET @result=0
			RETURN @result
	END;
```

```
Command(s) completed successfully.
```

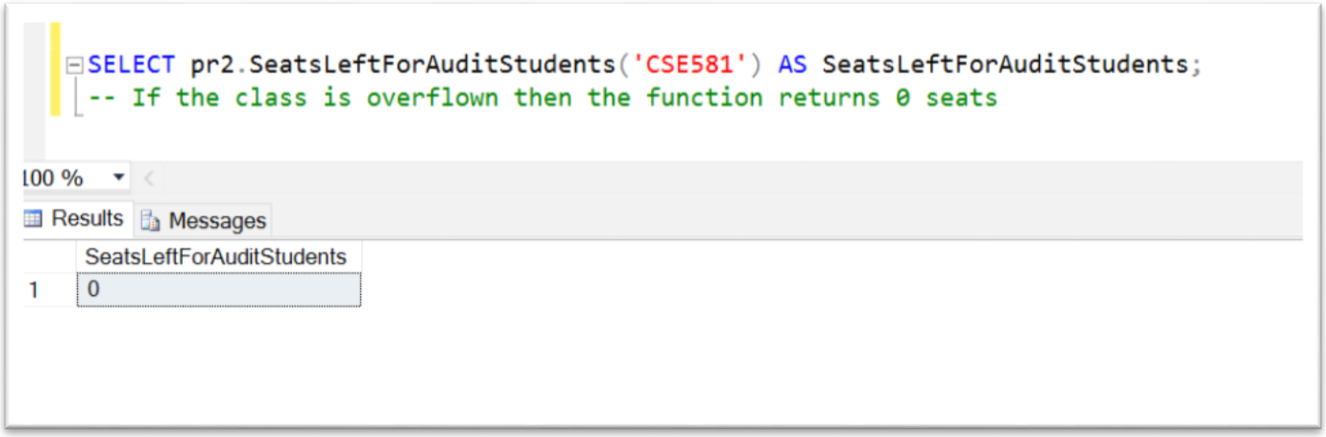
5.4.1 Test Cases



```
-- Test Cases for Function 1: pr2.SeatsLeftForAuditStudents--

-- Test Case 1--

SELECT pr2.SeatsLeftForAuditStudents('GEO125') AS SeatsLeftForAuditStudents;
```

```
   SeatsLeftForAuditStudents
1  5
```

```
Test Case 2
```



```
SELECT pr2.SeatsLeftForAuditStudents('CSE581') AS SeatsLeftForAuditStudents;
-- If the class is overflown then the function returns 0 seats
```

```
   SeatsLeftForAuditStudents
1  0
```

Thank You