

Syracuse University, CSE- 581

Project 2: Designing and Implementing the Patient Access and Registration Database for University Medical Center

Jannah Hassan-Badwan



Abstract:

This project report includes the full design and implementation of the Patient Access and Registration database for the University Medical Center. In this project a very thorough ER diagram was created to determine the tables, contents of the tables, and table relationships that would be included in the final database implementation. The ER diagram was used to guide the implementation and SQL code that was created during implementation. Included in the implementation of the database was a final count of 25 tables with constraints and other specifications, four tested views, four tested stored procedures, four tested user defined functions, four triggers, and four tested transactions. Along with this, four scripts were also developed to create users in the database that have varying levels of security permissions. The source code, test cases, and screenshots are all included in the full report below.

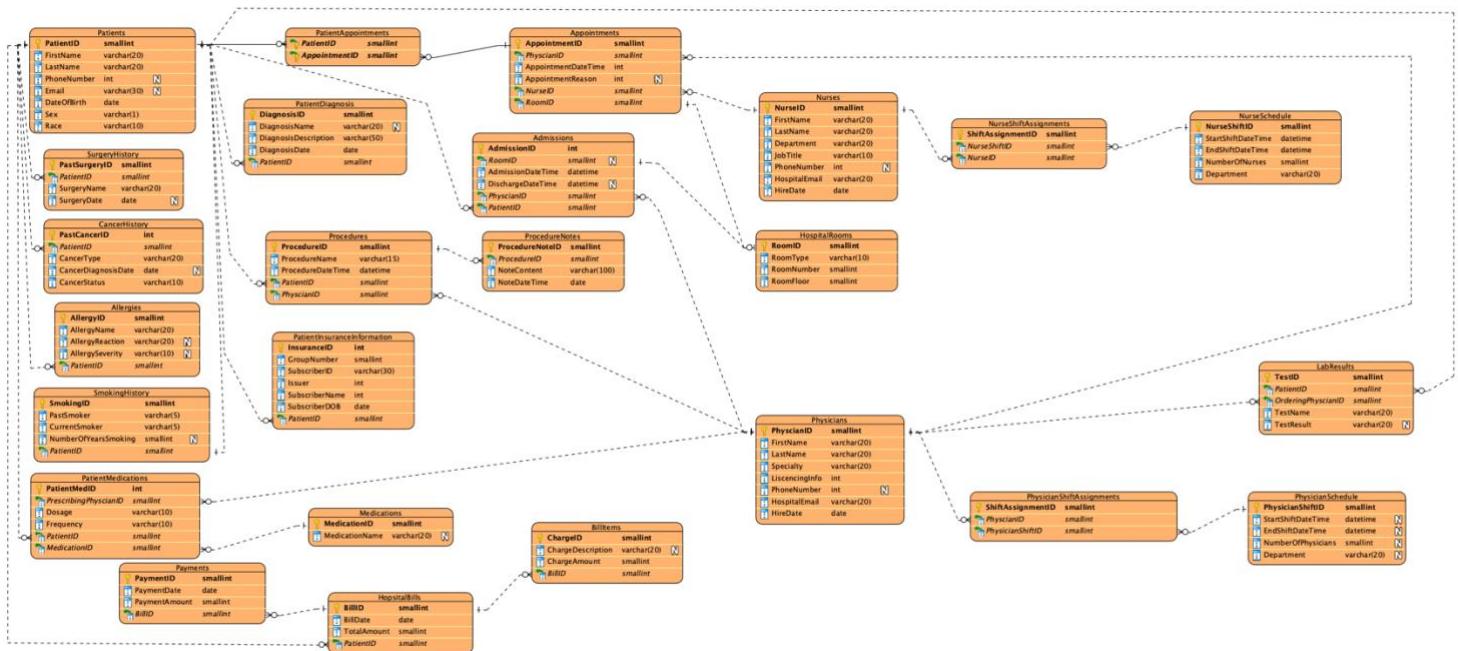
Table of Contents

Abstract	2
Design	4
Implementation	6
Testing	31
Conclusions	38
Appendix A: Implementation Screenshots.....	39
Appendix B: Testing Screenshots.....	51

Design of Database

Introduction:

Designing of this database for Patient Access and Registration began with brainstorming the information I felt was necessary to include in the database and what would accurately reflect the steps of the patient experience in the hospital. Along with this decision making about the relationships between all of the tables in the database had to be carefully thought of in order to maintain data and referential integrity. All of these decisions culminated into the final ER diagram design shown below.



Design Considerations:

The main design considerations that I feel are noteworthy based on my ER diagram and database implementation are choosing to create five separate tables (SurgeryHistory, CancerHistory, Allergies, SmokingHistory, and PatientMedications) that consist of all the information that would be considered a patient's medical history. At first I thought it would be possible to just include the medical history of a patient in the patients table but later felt that the table would be too large and unmanageable if I were to have designed it that way. Another more prominent design decision that I made was choosing to split the hospital billing into three different tables. The first table being HospitalBills that includes every bill that was sent to every patient and includes only the total amount of the bill. The second table being the BillItems table that included every charge that was in each bill. I made this a separate table because I felt that there was no other way to make sure that an itemized bill would be able to be generated from just the HospitalBills table. This way you can generate a view to show all the itemized charged of a bill and be able to figure out how the total amount arose. The third table was used to track payments and that is the Payments table. I felt that it was better to keep the payments in a separate table from the table were the bills

are kept because for hospital bills they are often a very large amount and tend to be paid over time rather than in one large chunk. This way we can use the Payments table to track all of the payments made to a specific bill based on its BillID. In terms of design decisions based around relationships of the table I followed logic and knowledge from previous experience I have had in the medical field to make those decisions. There were a multiple many to many relationships that were connected via linking tables and there were also one to one relationships for more specific relationships like patients and smoking history, each patient should only have one record in smoking history based on what that table holds.

Database Objects:

Along with the 25 tables in the ER diagram there are many other database objects included in the database implementation that are listed below.

Views:

- FullPatientSummary
- PatientBilling
- LabResultsAndDiagnosisView
- PatientProcedures

Stored Procedures:

- spUpdatePatientPersonalInfo
- spAddDiagnosis
- spAddNewPatient
- spAddNewPatientAppointment

User-defined functions:

- fnRemainingBillBalance
- fnGetPatientAdmissionHistory
- fnGetAllPhysicianAppointments
- fnGetPatientLabHistory

Triggers

- trPreventDoubleRoomOccupancy
- trPreventDoubleBooking
- trPreventMedicationAllergy
- trPreventSurgeryWithAllergy

Users Created with Different Permissions/Security Levels

- DrGoodwin
- NurseNelly
- SchedulerSam
- NurseManagerMike

Implementation of Database

Source Code for Creating Database and Schema:

```
CREATE DATABASE OurHospital; --creating the database for our schema and tables  
GO
```

```
USE OurHospital; --using that database to house the schema  
GO
```

```
CREATE SCHEMA my_hospital; --creating the schema  
GO
```

Source Code for Creating All Tables:

--Table 1

```
CREATE TABLE OurHospital.my_hospital.Patients(  
    PatientID smallint PRIMARY KEY NOT NULL,  
    FirstName varchar(20) NOT NULL,  
    LastName varchar(20) NOT NULL,  
    Email varchar(30) NULL,  
    PhoneNumber int NULL,  
    DateOfBirth date NOT NULL,  
    Sex varchar(1) NOT NULL,  
    Race varchar(10) NULL  
);  
GO
```

--Table 2

```
CREATE TABLE OurHospital.my_hospital.SurgeryHistory(  
    PastSurgeryID smallint PRIMARY KEY NOT NULL,  
    SurgeryName varchar(20) NOT NULL,  
    SurgeryDatev date NULL,  
    PatientID smallint NOT NULL,  
    CONSTRAINT SurgeryHistoryFK_PatientID  
        FOREIGN KEY (PatientID) REFERENCES  
        OurHospital.my_hospital.Patients(PatientID)  
);  
GO
```

--Table 3

```
CREATE TABLE OurHospital.my_hospital.CancerHistory(  
    PastCancerID smallint PRIMARY KEY NOT NULL,  
    PatientID smallint NOT NULL,
```

```
CancerType varchar(20) NOT NULL,  
CancerDiagnosisDate date NULL,  
CancerStatus varchar(10) NOT NULL,  
CONSTRAINT CancerHistoryFK_PatientID  
    FOREIGN KEY (PatientID) REFERENCES  
        OurHospital.my_hospital.Patients(PatientID)  
);  
GO
```

--Table 4

```
CREATE TABLE OurHospital.my_hospital.Allergies(  
    AllergyID smallint PRIMARY KEY NOT NULL,  
    AllergyName varchar(20) NOT NULL,  
    AllergyReaction varchar(20) NULL,  
    AllergySeverity varchar(10) NULL,  
    PatientID smallint NOT NULL,  
    CONSTRAINT AllergiesFK_PatientID  
        FOREIGN KEY (PatientID) REFERENCES  
            OurHospital.my_hospital.Patients(PatientID)  
);  
GO
```

--Table 5

```
CREATE TABLE OurHospital.my_hospital.SmokingHistory(  
    SmokingID smallint PRIMARY KEY NOT NULL,  
    PastSmoker varchar(5) NOT NULL,  
    CurrentSmoker varchar(5) NOT NULL,  
    NumberOfYearsSmoking smallint NULL,  
    PatientID smallint NOT NULL,  
    CONSTRAINT SmokingHistoryFK_PatientID  
        FOREIGN KEY (PatientID) REFERENCES  
            OurHospital.my_hospital.Patients(PatientID)  
);  
GO
```

--Table 6

```
CREATE TABLE OurHospital.my_hospital.PatientDiagnosis(  
    DiagnosisID smallint PRIMARY KEY NOT NULL,  
    DiagnosisName varchar(20) NULL,  
    DiagnosisDescription varchar(50) NOT NULL,  
    DiagnosisDate date NOT NULL,
```

```
PatientID smallint NOT NULL,
CONSTRAINT PatientDiagnosisFK_PatientID
    FOREIGN KEY (PatientID) REFERENCES
        OurHospital.my_hospital.Patients(PatientID)
);
GO
```

--Table 7

```
CREATE TABLE OurHospital.my_hospital.PatientInsuranceInformation(
    InsuranceID smallint PRIMARY KEY NOT NULL,
    GroupNumber smallint NOT NULL,
    SubscriberID varchar(30) NOT NULL,
    Issuer int NOT NULL,
    SubscriberFirstName varchar(10) NOT NULL,
    SubscriberLastName varchar(10) NOT NULL,
    SubscriberDOB date NOT NULL,
    PatientID smallint NOT NULL,
    CONSTRAINT PatientInsuranceInformationFK_PatientID
        FOREIGN KEY (PatientID) REFERENCES
            OurHospital.my_hospital.Patients(PatientID)
);
GO
```

--Table 8

```
CREATE TABLE OurHospital.my_hospital.HospitalBills(
    BillID smallint PRIMARY KEY NOT NULL,
    BillDate date NOT NULL,
    TotalAmount smallint NOT NULL,
    PatientID smallint NOT NULL,
    CONSTRAINT HospitalBillsFK_PatientID
        FOREIGN KEY (PatientID) REFERENCES
            OurHospital.my_hospital.Patients(PatientID)
);
GO
```

--Table 9

```
CREATE TABLE OurHospital.my_hospital.Payments(
    PaymentID smallint PRIMARY KEY NOT NULL,
    PaymentDate date NOT NULL,
    PaymentAmount smallint NOT NULL,
    BillID smallint NOT NULL,
```

```
CONSTRAINT PaymentsFK_BillID
    FOREIGN KEY (BillID) REFERENCES
        OurHospital.my_hospital.HospitalBills(BillID)
);
GO
```

--Table 10

```
CREATE TABLE OurHospital.my_hospital.BillItems(
    ChargeID smallint PRIMARY KEY NOT NULL,
    ChargeDescription varchar(30) NOT NULL,
    ChargeAmount smallint NOT NULL,
    BillID smallint NOT NULL,
    CONSTRAINT BillItemsFK_BillID
        FOREIGN KEY (BillID) REFERENCES
            OurHospital.my_hospital.HospitalBills(BillID)
);
GO
```

--Table 11

```
CREATE TABLE OurHospital.my_hospital.Medications(
    MedicationID smallint PRIMARY KEY NOT NULL,
    MedicationName varchar(20) NOT NULL
);
GO
```

--Table 12

```
CREATE TABLE OurHospital.my_hospital.Physicians(
    PhysicianID smallint PRIMARY KEY NOT NULL,
    FirstName varchar(20) NOT NULL,
    LastName varchar(20) NOT NULL,
    Specialty varchar(20) NOT NULL,
    LiscenceNumber int NOT NULL,
    PhoneNumber int NOT NULL,
    HospitalEmail varchar(30) NOT NULL,
    HireDate date NOT NULL
);
GO
```

--Table 13

```
CREATE TABLE OurHospital.my_hospital.PatientMedications(
    PatientMedID smallint PRIMARY KEY NOT NULL,
```

```

Dosage varchar(20) NOT NULL,
Frequency varchar(20) NOT NULL,
PatientID smallint NOT NULL,
PhysicianID smallint NOT NULL,
MedicationID smallint NOT NULL,
CONSTRAINT PatientMedicationsFK_PatientID
    FOREIGN KEY (PatientID) REFERENCES
        OurHospital.my_hospital.Patients(PatientID),
CONSTRAINT PatientMedicationsFK_PhysicianID
    FOREIGN KEY (PhysicianID) REFERENCES
        OurHospital.my_hospital.Physicians(PhysicianID),
CONSTRAINT PatientMedicationsFK_MedicationID
    FOREIGN KEY (MedicationID) REFERENCES
        OurHospital.my_hospital.Medications(MedicationID)
);
GO

```

--Table 14

```

CREATE TABLE OurHospital.my_hospital.HospitalProcedures(
    ProcedureID smallint PRIMARY KEY NOT NULL,
    ProcedureName varchar(15) NOT NULL,
    ProcedureDateTime smalldatetime NOT NULL,
    PatientID smallint NOT NULL,
    PhysicianID smallint NOT NULL,
    CONSTRAINT HospitalProceduresFK_PatientID
        FOREIGN KEY (PatientID) REFERENCES
            OurHospital.my_hospital.Patients(PatientID),
    CONSTRAINT HospitalProceduresFK_PhysicianID
        FOREIGN KEY (PhysicianID) REFERENCES
            OurHospital.my_hospital.Physicians(PhysicianID)
);
GO

```

--Table 15

```

CREATE TABLE OurHospital.my_hospital.ProcedureNotes(
    ProcedureNoteID smallint PRIMARY KEY NOT NULL,
    ProcedureID smallint NOT NULL,
    NoteContent varchar(100) NOT NULL,
    NoteDateTime datetime NOT NULL,
    NoteAuthorID smallint NOT NULL,
    CONSTRAINT ProcedureNotesFK_ProcedureID
        FOREIGN KEY (ProcedureID) REFERENCES
            OurHospital.my_hospital.HospitalProcedures(ProcedureID)
)

```

```
);  
GO
```

--Table 16

```
CREATE TABLE OurHospital.my_hospital.PhysicianSchedule(  
    PhysicianShiftID smallint PRIMARY KEY NOT NULL,  
    StartShiftDateTime datetime NOT NULL,  
    EndShiftDateTime datetime NOT NULL,  
    NumberOfPhysicians smallint NOT NULL,  
    Department varchar(20) NOT NULL,  
);  
GO
```

--Table 17

```
CREATE TABLE OurHospital.my_hospital.PhysicianShiftAssignments(  
    ShiftAssignmentID smallint PRIMARY KEY NOT NULL,  
    PhysicianID smallint NOT NULL,  
    PhysicianShiftID smallint NOT NULL  
    CONSTRAINT PhysicianShiftAssignmentsFK_PhysicianID  
        FOREIGN KEY (PhysicianID) REFERENCES  
            OurHospital.my_hospital.Physicians(PhysicianID),  
    CONSTRAINT PhysicianShiftAssignmentsFK_PhysicianShiftID  
        FOREIGN KEY (PhysicianShiftID) REFERENCES  
            OurHospital.my_hospital.PhysicianSchedule(PhysicianShiftID)  
);  
GO
```

--Table 18

```
CREATE TABLE OurHospital.my_hospital.LabResults(  
    TestID smallint PRIMARY KEY NOT NULL,  
    PatientID smallint NOT NULL,  
    PhysicianID smallint NOT NULL,  
    TestName varchar(20) NOT NULL,  
    TestResult varchar(20) NULL,  
    CONSTRAINT LabResultsFK_PhysicianID  
        FOREIGN KEY (PhysicianID) REFERENCES  
            OurHospital.my_hospital.Physicians(PhysicianID),  
    CONSTRAINT LabResultsFK_PatientID  
        FOREIGN KEY (PatientID) REFERENCES  
            OurHospital.my_hospital.Patients(PatientID)  
);  
GO
```

--Table 19

```
CREATE TABLE OurHospital.my_hospital.HospitalRooms(
    RoomID smallint PRIMARY KEY NOT NULL,
    RoomType varchar(10) NOT NULL,
    RoomNumber smallint NOT NULL,
    RoomFloor smallint NOT NULL
);
GO
```

--Table 20

```
CREATE TABLE OurHospital.my_hospital.Nurses(
    NurselD smallint PRIMARY KEY NOT NULL,
    FirstName varchar(20) NOT NULL,
    LastName varchar(20) NOT NULL,
    HospitalEmail varchar(30) NOT NULL,
    PhoneNumber int NULL,
    Department varchar(20) NOT NULL,
    JobTitle varchar(20) NOT NULL,
    HireDate date NOT NULL
);
GO
```

--Table 21

```
CREATE TABLE OurHospital.my_hospital.NurseSchedule(
    NurseShiftID smallint PRIMARY KEY NOT NULL,
    StartShiftDateTime datetime NOT NULL,
    EndShiftDateTime datetime NOT NULL,
    NumberOfNurses smallint NOT NULL,
    Department varchar(20) NOT NULL
);
```

--Table 22

```
CREATE TABLE OurHospital.my_hospital.NurseShiftAssignments(
    ShiftAssignmentID smallint PRIMARY KEY NOT NULL,
    NurseShiftID smallint NOT NULL,
    NurselD smallint NOT NULL,
    CONSTRAINT NurseShiftAssignmentsFK_NurseShiftID
        FOREIGN KEY (NurseShiftID) REFERENCES
            OurHospital.my_hospital.NurseSchedule(NurseShiftID),
    CONSTRAINT NursesFK_NurselD
        FOREIGN KEY (NurselD) REFERENCES
            OurHospital.my_hospital.Nurses(NurselD)
);
```

GO

--Table 23

```
CREATE TABLE OurHospital.my_hospital.Appointments(
    AppointmentID smallint PRIMARY KEY NOT NULL,
    PhysicianID smallint NOT NULL,
    AppointmentDateTime datetime NOT NULL,
    AppointmentReason varchar(50) NOT NULL,
    NurseID smallint NOT NULL,
    RoomID smallint NOT NULL,
    CONSTRAINT AppointmentsFK_NurseID
        FOREIGN KEY (NurseID) REFERENCES
            OurHospital.my_hospital.Nurses(NurseID),
    CONSTRAINT AppointmentsFK_RoomID
        FOREIGN KEY (RoomID) REFERENCES
            OurHospital.my_hospital.HospitalRooms(RoomID),
    CONSTRAINT AppointmentsFK_PhysicianID
        FOREIGN KEY (PhysicianID) REFERENCES
            OurHospital.my_hospital.Physicians(PhysicianID)
);
```

GO

--Table 24

```
CREATE TABLE OurHospital.my_hospital.PatientAppointments(
    PatientID smallint NOT NULL,
    AppointmentID smallint NOT NULL,
    CONSTRAINT PatientAppointmentsPK
        PRIMARY KEY (PatientID, AppointmentID),
    CONSTRAINT PatientAppointmentsFK_PatientID
        FOREIGN KEY (PatientID) REFERENCES
            OurHospital.my_hospital.Patients(PatientID),
    CONSTRAINT PatientAppointmentsFK_AppointmentID
        FOREIGN KEY (AppointmentID) REFERENCES
            OurHospital.my_hospital.Appointments(AppointmentID)
);
```

GO

--Table 25

```
CREATE TABLE OurHospital.my_hospital.Admissions(
    AdmissionID smallint PRIMARY KEY NOT NULL,
    RoomID smallint NOT NULL,
    AdmissionDateTime datetime NOT NULL,
    DischargeDateTime datetime NOT NULL,
```

```

    PhysicianID smallint NOT NULL,
    PatientID smallint NOT NULL,
    CONSTRAINT AdmissionsFK_RoomID
        FOREIGN KEY (RoomID) REFERENCES
            OurHospital.my_hospital.HospitalRooms(RoomID),
    CONSTRAINT AdmissionsFK_PhysicianID
        FOREIGN KEY (PhysicianID) REFERENCES
            OurHospital.my_hospital.Physicians(PhysicianID),
    CONSTRAINT AdmissionsFK_PatientID
        FOREIGN KEY (PatientID) REFERENCES
            OurHospital.my_hospital.Patients(PatientID)
);
GO

```

Source Code for Creating All Views:

```

USE OurHospital;
GO

```

```

/*Creation of a view that shows a full patient's summary including patient's personal
information, information about medications,
information about past cancer, surgeries, diagnosis, and admission/discharge
information*/
CREATE VIEW my_hospital.FullPatientSummary AS
    SELECT my_hospital.Patients.PatientID, FirstName, LastName, DateOfBirth, Sex,
    Race, SurgeryName, CancerType, CancerDiagnosisDate, CancerStatus, AllergyName,
    AllergyReaction, AllergySeverity, AdmissionDateTime, DischargeDateTime,
    DiagnosisName, DiagnosisDescription, DiagnosisDate,
    my_hospital.PatientMedications.MedicationID, Dosage, Frequency
    FROM my_hospital.Patients
    LEFT JOIN my_hospital.SurgeryHistory ON my_hospital.Patients.PatientID =
    my_hospital.SurgeryHistory.PatientID
    LEFT JOIN my_hospital.PatientMedications ON my_hospital.Patients.PatientID =
    my_hospital.PatientMedications.PatientID
    LEFT JOIN my_hospital.CancerHistory ON my_hospital.Patients.PatientID =
    my_hospital.CancerHistory.PatientID
    LEFT JOIN my_hospital.Allergies ON my_hospital.Patients.PatientID =
    my_hospital.Allergies.PatientID
    LEFT JOIN my_hospital.PatientDiagnosis ON my_hospital.Patients.PatientID =
    my_hospital.PatientDiagnosis.PatientID
    LEFT JOIN my_hospital.Admissions ON my_hospital.Patients.PatientID =
    my_hospital.Admissions.PatientID
GO

```

```
/*Creates a view that puts together all of the bills and charges in each bill for each patient*/
CREATE VIEW my_hospital.PatientBilling AS
    SELECT my_hospital.Patients.PatientID, FirstName, LastName,
    my_hospital.HospitalBills.BillID, BillDate, TotalAmount, ChargeAmount,
    ChargeDescription
    FROM my_hospital.Patients
    LEFT JOIN my_hospital.HospitalBills ON Patients.PatientID = HospitalBills.PatientID
    LEFT JOIN my_hospital.BillItems ON HospitalBills.BillID = BillItems.BillID

/*Creating a view that includes all of the lab results and diagnosis for every patient*/
CREATE VIEW my_hospital.LabResultsAndDiagnosisView AS
    SELECT Patients.PatientID, FirstName, LastName, TestName, TestResult,
    DiagnosisName, DiagnosisDescription, DiagnosisDate
    FROM my_hospital.Patients
    LEFT JOIN my_hospital.LabResults ON Patients.PatientID = LabResults.PatientID
    LEFT JOIN my_hospital.PatientDiagnosis ON Patients.PatientID =
    PatientDiagnosis.PatientID;
```

GO

```
/*Creates a view to show all patient procedures */
CREATE VIEW my_hospital.PatientProcedures AS
    SELECT Patients.PatientID, FirstName, LastName, ProcedureName,
    ProcedureDateTime, PhysicianID
    FROM my_hospital.Patients JOIN my_hospital.HospitalProcedures
        ON Patients.PatientID = HospitalProcedures.PatientID
```

GO

Source Code for Creating All Stored Procedures:

```
USE OurHospital;
GO
```

```
CREATE PROCEDURE my_hospital.spAddDiagnosis
    @PatientID smallint,
    @DiagnosisName varchar(20) = NULL,
    @DiagnosisDescription varchar(50),
    @DiagnosisDate date
AS
BEGIN
    --check to see if patientID is valid
    IF NOT EXISTS (SELECT 1 FROM my_hospital.Patients WHERE PatientID = @PatientID)
        THROW 5200010, 'Invalid PatientID. The patient does not exist in the Patients
            table.', 1;
    --check to see if diagnosis date is valid
```

```
IF (@DiagnosisDate > GETDATE())
    THROW 520011, 'Invalid DiagnosisDate. The date cannot be in the future.', 1;
--check to see if diagnosis name length is valid if its not null
IF (@DiagnosisName IS NOT NULL AND LEN(@DiagnosisName) > 20)
    THROW 520012, 'Invalid DiagnosisName. The name cannot be greater than
20 characters.', 1;
--check to see if diagnosis description is the correct length
IF (LEN(@DiagnosisDescription) > 50)
    THROW 520013, 'Invalid DiagnosisDescription. The description cannot be
greater than 50 characters.', 1;

INSERT INTO my_hospital.PatientDiagnosis (DiagnosisName, DiagnosisDescription,
DiagnosisDate, PatientID)
VALUES (@DiagnosisName, @DiagnosisDescription, @DiagnosisDate, @PatientID)
END
GO
```

```
USE OurHospital;
GO
```

```
CREATE PROCEDURE my_hospital.spAddNewPatient
@PatientID smallint,
@FirstName varchar(20),
@LastName varchar(20),
@email varchar(30) = NULL,
@PhoneNumber int = NULL,
@DateOfBirth date,
@Sex varchar(1),
@Race varchar(10) = NULL
AS
BEGIN
--check to make sure fields that are not nullable are filled
IF (@PatientID IS NULL OR @FirstName IS NULL OR @LastName IS NULL OR
@DateOfBirth IS NULL OR @Sex IS NULL)
    THROW 520004, 'Incomplete input data. PatientID, FirstName, LastName,
DateOfBirth and Sex are required.', 1;
--check to make sure length of first name and last name is correct
IF (LEN(@FirstName) > 20 OR LEN(@LastName) > 20)
    THROW 520005, 'Invalid input data. FirstName and LastName must be less
than or equal to 20 characters', 1;
--check to make sure length of email is correct
IF (LEN(@Email) > 30)
```

```
THROW 520006, 'Invalid input data. Email must be less than or equal to 30
characters', 1;
--check to make sure length of sex is correct
IF (LEN(@Sex) > 1)
    THROW 520007, 'Invalid input data. Sex must only be one character, M for
    Male and F for female', 1;
--check to make sure race is the correct length if it wasn't null
IF(@RACE IS NOT NULL AND LEN(@Race) > 10)
    THROW 520008, 'Invalid input data. Race must be less than or equal to 10
    characters', 1;
--check to make sure date of birth is valid
IF (@DateOfBirth > GETDATE())
    THROW 520009, 'Invalid input data. Date of birth cannot be in the future.', 1;

INSERT INTO my_hospital.Patients (PatientID, FirstName, LastName, Email,
PhoneNumber, DateOfBirth, Sex, Race)
VALUES (@PatientID, @FirstName, @LastName, @Email, @PhoneNumber,
@DateOfBirth, @Sex, @Race)
END
GO
```

```
USE OurHospital;
GO
```

```
CREATE PROCEDURE my_hospital.spUpdatePatientPersonalInfo
    @PatientID smallint = NULL,
    @FirstName varchar(20) = NULL,
    @LastName varchar(20) = NULL,
    @Email varchar(30) = NULL,
    @PhoneNumber int = NULL,
    @DateOfBirth date = NULL,
    @Sex varchar(1) = NULL,
    @Race varchar(10) = NULL
AS
BEGIN
    --check to make sure fields that are not nullable are filled
    IF (@PatientID IS NULL OR @FirstName IS NULL OR @LastName IS NULL OR
        @DateOfBirth IS NULL OR @Sex IS NULL)
        THROW 520014, 'Incomplete input data. PatientID, FirstName, LastName,
        DateOfBirth and Sex are required.', 1;
    --check to make sure length of first name and last name is correct
    IF (LEN(@FirstName) > 20 OR LEN(@LastName) > 20)
```

```
THROW 520015, 'Invalid input data. FirstName and LastName must be less
than or equal to 20 characters', 1;
--check to make sure length of email is correct
IF (LEN(@Email) > 30)
    THROW 520016, 'Invalid input data. Email must be less than or equal to 30
    characters', 1;
--check to make sure length of sex is correct
IF (LEN(@Sex) > 1)
    THROW 520017, 'Invalid input data. Sex must only be one character, M for
    Male and F for female', 1;
--check to make sure race is the correct length if it wasn't null
IF(@RACE IS NOT NULL AND LEN(@Race) > 10)
    THROW 520018, 'Invalid input data. Race must be less than or equal to 10
    characters', 1;
--check to make sure date of birth is valid
IF (@DateOfBirth > GETDATE())
    THROW 520019, 'Invalid input data. Date of birth cannot be in the future.', 1;
-- check if patient ID exists in the Patients table
IF NOT EXISTS(SELECT * FROM my_hospital.Patients WHERE PatientID = @PatientID)
    THROW 520020, 'Invalid input data. PatientID does not exist in the Patients
    table', 1;

-- update the patient information
UPDATE my_hospital.Patients
SET FirstName = @FirstName,
    LastName = @LastName,
    Email = @Email,
    PhoneNumber = @PhoneNumber,
    DateOfBirth = @DateOfBirth,
    Sex = @Sex,
    Race = @Race
WHERE PatientID = @PatientID;
END
GO

USE OurHospital;
GO

CREATE PROCEDURE my_hospital.AddNewPatientAppointment
    @PatientID smallint,
    @AppointmentDateTime datetime,
    @AppointmentReason varchar(50),
    @NurseID smallint,
```

```
@RoomID smallint,
@PhysicianID smallint,
@AppointmentID smallint

AS
BEGIN
    --check that NurseID is valid
    IF NOT EXISTS (SELECT 1 FROM my_hospital.Nurses WHERE NurseID = @NurseID)
        THROW 520021, 'Invalid NurseID, NurseID does not exist in the Nurses table',
        1;

    --check that RoomID is valid
    IF NOT EXISTS (SELECT 1 FROM my_hospital.HospitalRooms WHERE @RoomID =
    @RoomID)
        THROW 520022, 'Invalid RoomID, RoomID does not exist in the
        HospitalRooms table', 1;

    --check that PatientID is valid
    IF NOT EXISTS (SELECT 1 FROM my_hospital.Patients WHERE PatientID = @PatientID)
        THROW 520023, 'Invalid PatientID, PatientID does not exist in the Patients
        table', 1;

    --check that PhysicianID is valid
    IF NOT EXISTS (SELECT 1 FROM my_hospital.Physicians WHERE PhysicianID =
    @PhysicianID)
        THROW 520024, 'Invalid PhysicianID, PhysicianID does not exist in the
        Physicians table', 1;

    --check that AppointmentID does not already exist
    IF EXISTS (SELECT 1 FROM my_hospital.Appointments WHERE AppointmentID =
    @AppointmentID)
        THROW 520025, 'AppointmentID already exists', 1;

    --check if appointment reason is long enough
    IF (LEN(@AppointmentReason) > 50)
        THROW 520026, 'Invalid data, appointment reason has to be no greater than
        50 characters', 1;

    IF (@AppointmentDateTime < GETDATE())
        THROW 520027, 'Invalid appointment date, appointment cannot be made in
        the past', 1;

    --insert new appointment data into appointments table
```

```
INSERT INTO my_hospital.Appointments (AppointmentDateTime, AppointmentReason,
NurseID, RoomID, PhysicianID, AppointmentID)
VALUES (@AppointmentDateTime, @AppointmentReason, @NurseID, @RoomID,
@PhysicianID, @AppointmentID);

--insert new record into patient appointments table
INSERT INTO my_hospital.PatientAppointments (PatientID, AppointmentID)
VALUES (@PatientID, @AppointmentID);
END
GO
```

Source Code for Creating All User-Defined Functions:

```
USE OurHospital;
GO
```

```
/*Creates a function that will subtract all of the payments made towards a patient's bill
giving them the final
remaining balance*/
CREATE FUNCTION my_hospital.fnRemainingBillBalance
    (@BillID smallint,
     @PatientID smallint)
RETURNS smallint
BEGIN
    RETURN (SELECT (TotalAmount - SUM(PaymentAmount)) AS RemainingBalance
            FROM my_hospital.HospitalBills JOIN my_hospital.Payments
              ON HospitalBills.BillID = Payments.BillID
            WHERE PatientID = @PatientID AND HospitalBills.BillID = @BillID
            GROUP BY TotalAmount)
END;
GO
```

```
/*Creates a function that gets all of a particular patient's admission and discharge history*/
CREATE FUNCTION my_hospital.fnGetPatientAdmissionHistory
    (@PatientID smallint)
RETURNS TABLE
RETURN (SELECT Patients.PatientID, AdmissionID, AdmissionDateTime, DischargeDateTime
      FROM my_hospital.Patients JOIN my_hospital.Admissions
        ON Patients.PatientID = Admissions.PatientID
      WHERE Patients.PatientID = @PatientID);
GO
```

```
/*Creates a function that gets all of the appointments a physician has scheduled*/
CREATE FUNCTION my_hospital.fnGetAllPhysicianAppointments
```

```
(@PhysicianID smallint)
RETURNS TABLE
RETURN (SELECT Physicians.PhysicianID, AppointmentID, AppointmentDateTime
      FROM my_hospital.Physicians JOIN my_hospital.Appointments
        ON Physicians.PhysicianID = Appointments.PhysicianID
     WHERE Physicians.PhysicianID = @PhysicianID);
GO
```

```
/*Creates a function that gets all of a patients lab results/lab history*/
CREATE FUNCTION my_hospital.fnGetPatientLabHistory
    (@PatientID smallint)
RETURNS TABLE
RETURN (SELECT Patients.PatientID, TestID, TestName, TestResult
      FROM my_hospital.Patients JOIN my_hospital.LabResults
        ON Patients.PatientID = LabResults.PatientID
     WHERE Patients.PatientID = @PatientID);
GO
```

Source Code for Creating All Triggers:

```
USE OurHospital;
GO
```

```
/*Creating a trigger to rollback the insertion of a procedure into the hospital procedures
table if a patient has a severe allergy to anesthesia*/
CREATE TRIGGER trPreventSurgeryWithAllergy
    ON my_hospital.HospitalProcedures
INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted JOIN my_hospital.Allergies ON Allergies.PatientID =
        inserted.PatientID
        WHERE Allergies.AllergyName = 'Anesthesia' AND Allergies.AllergySeverity =
        'Severe'
    )
    BEGIN
        ROLLBACK TRANSACTION;
        THROW 520030, 'This patient has a severe allergy to anesthesia and cannot
        have surgery scheduled', 1;
    END
    ELSE
        INSERT INTO my_hospital.HospitalProcedures
```

```
SELECT * FROM inserted;
COMMIT TRANSACTION;
END
GO

/*Creates a trigger that prevents hospital staff from making an appointment with a
physician that already has
an appointment scheduled for that time*/
CREATE TRIGGER trPreventDoubleBooking
    ON my_hospital.Appointments
INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted JOIN my_hospital.Appointments ON Appointments.PhysicianID =
        inserted.PhysicianID
        WHERE Appointments.AppointmentDateTime = inserted.AppointmentDateTime
    )
    BEGIN
        ROLLBACK TRANSACTION;
        THROW 520031, 'The physician already has an appointment scheduled for that time.
        Please choose a different time.', 1;
    END
    ELSE
        INSERT INTO my_hospital.Appointments
        SELECT * FROM inserted;
    END
GO
```

```
/*creates a trigger that prevents a new patient from being admitted into a room that is
already occupied by someone else*/
CREATE TRIGGER trPreventDoubleRoomOccupancy
    ON my_hospital.Admissions
INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted JOIN my_hospital.Admissions ON Admissions.RoomID =
        inserted.RoomID
        WHERE Admissions.DischargeDateTime IS NULL
    )
```

```

)
BEGIN
    ROLLBACK TRANSACTION;
    THROW 520032, 'Another patient is already admitted in this room. Please choose a
different room.', 1;
END
ELSE
    INSERT INTO my_hospital.Admissions
        SELECT * FROM inserted;
END
GO

/*Creates a trigger to prevent prescribing or giving a patient a medication that they are
allergic to*/
CREATE TRIGGER trPreventMedicationAllergy
    ON my_hospital.PatientMedications
INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted JOIN my_hospital.Allergies ON inserted.PatientID = Allergies.PatientID
        JOIN my_hospital.Medications ON inserted.MedicationID = Medications.MedicationID
        WHERE Allergies.AllergyName = Medications.MedicationName
    )
    BEGIN
        ROLLBACK TRANSACTION;
        THROW 520033, 'This patient is allergic to this medication and cannot receive it', 1;
    END
    ELSE
        INSERT INTO my_hospital.PatientMedications (PatientMedID, Dosage, Frequency,
        PatientID, PhysicianID, MedicationID)
        SELECT PatientMedId, Dosage, Frequency, PatientID, PhysicianID, MedicationID
        FROM inserted;
END
GO

```

Source Code for Creating Users with Different Security/Permissions:

```
--start script 1
USE OurHospital;
GO
/*creating a user named Dr.Goodwin who has access to look at all patient history and read
and write to patient medications,
patient diagnosis, and patient admissions tables*/
CREATE LOGIN DrGoodwin WITH PASSWORD = 'goodwin$19';
CREATE USER DrGoodwin FOR LOGIN DrGoodwin;
GRANT SELECT ON OBJECT::my_hospital.Patients TO DrGoodwin;
GRANT SELECT, INSERT, UPDATE ON OBJECT::my_hospital.PatientMedications TO
DrGoodwin;
GRANT SELECT, INSERT, UPDATE ON OBJECT::my_hospital.PatientDiagnosis TO DrGoodwin;
GRANT SELECT, INSERT, UPDATE ON OBJECT::my_hospital.Admissions TO DrGoodwin;
GRANT SELECT ON OBJECT::my_hospital.SurgeryHistory TO DrGoodwin;
GRANT SELECT ON OBJECT::my_hospital.SmokingHistory TO DrGoodwin;
GRANT SELECT ON OBJECT::my_hospital.CancerHistory TO DrGoodwin;
GRANT SELECT ON OBJECT::my_hospital.Allergies TO DrGoodwin;
GO
```

```
--start script 2
USE OurHospital;
GO
/*Creates a user that is a nurse named nelly, she only has read access to patient
medications, diagnosis, and admissions
But she has read and write access to all tables regarding patient history because nurses
normally take the patient's medical
history down during intake*/
CREATE LOGIN NurseNelly WITH PASSWORD = 'nelly$11';
CREATE USER NurseNelly FOR LOGIN NurseNelly;
GRANT SELECT, INSERT, UPDATE ON OBJECT::my_hospital.Patients TO NurseNelly;
GRANT SELECT ON OBJECT::my_hospital.PatientMedications TO NurseNelly;
GRANT SELECT ON OBJECT::my_hospital.PatientDiagnosis TO NurseNelly;
GRANT SELECT ON OBJECT::my_hospital.Admissions TO NurseNelly;
GRANT SELECT, INSERT, UPDATE ON OBJECT::my_hospital.SurgeryHistory TO NurseNelly;
GRANT SELECT, INSERT, UPDATE ON OBJECT::my_hospital.SmokingHistory TO NurseNelly;
GRANT SELECT, INSERT, UPDATE ON OBJECT::my_hospital.CancerHistory TO NurseNelly;
GRANT SELECT, INSERT, UPDATE ON OBJECT::my_hospital.Allergies TO NurseNelly;
GO
```

```
--start script 3
USE OurHospital;
GO
```

```
/*Creates a user named Sam that is just a scheduler of patient appointments, so he only
has access to read and write the
appointments and patient appointments tables*/
CREATE LOGIN SchedulerSam WITH PASSWORD = 'samiam$13';
CREATE USER SchedulerSam FOR LOGIN SchedulerSam;
GRANT SELECT, INSERT, UPDATE ON OBJECT::my_hospital.PatientAppointments TO
SchedulerSam;
GRANT SELECT, INSERT, UPDATE ON OBJECT::my_hospital.Appointments TO SchedulerSam;
GRANT SELECT ON OBJECT::my_hospital.Patients TO SchedulerSam;
GO

--start script 4
USE OurHospital;
GO
/*Creates a user who is the Nurse Manager and creates the schedules of all the nurses in
all departments of the hospital*/
CREATE LOGIN NurseManagerMike WITH PASSWORD = 'managingisthegame$12';
CREATE USER NurseManagerMike FOR LOGIN NurseManagerMike;
GRANT SELECT, INSERT, UPDATE ON OBJECT::my_hospital.NurseSchedule TO
NurseManagerMike;
GRANT SELECT, INSERT, UPDATE ON OBJECT::my_hospital.NurseShiftAssignments TO
NurseManagerMike;
GRANT SELECT, INSERT, UPDATE ON OBJECT::my_hospital.Nurses TO NurseManagerMike;
GO
```

Source Code for Populating the Database (with at least 5 rows per table):

```
--inserting 5 rows of data into the patients table, essentially creating 5 patients
INSERT INTO OurHospital.my_hospital.Patients (PatientID, FirstName, LastName, Email,
PhoneNumber, DateOfBirth, Sex, Race)
VALUES
(1, 'Mandy', 'McMillan', 'mmcmillan@gmail.com', NULL, '1990-01-01', 'F', 'White'),
(2, 'Malek', 'Hussain', 'mhussain@hotmail.com', NULL, '1991-02-14', 'M', 'White'),
(3, 'Jannah', 'Hassan', 'jhassan@gmail.com', NULL, '1995-04-29', 'F', 'MidEast'),
(4, 'Josh', 'Johnson', 'jjohnson@yahoo.com', NULL, '1960-09-17', 'M', NULL),
(5, 'Savannah', 'Lee', 'ssimpson@hotmail.com', NULL, '2001-08-05', 'F', 'Asian');
GO
```

```
--inserting 5 rows of data into the Surgery History table
INSERT INTO OurHospital.my_hospital.SurgeryHistory (PastSurgeryID, SurgeryName,
SurgeryDatev, PatientID)
VALUES
(1, 'Appendectomy', '1999-01-05', 1),
(2, 'Mastectomy', '2020-05-15', 1),
```

```
(3, 'Meniscus Repair', '2006-09-18', 4),
(4, 'Hernia Repair', '2019-06-11', 5),
(5, 'Nephrectomy', '2010-07-01', 4);
```

--inserting 5 rows of data into the Cancer History table

```
INSERT INTO OurHospital.my_hospital.CancerHistory (PastCancerID, PatientID, CancerType,
CancerDiagnosisDate, CancerStatus)
```

```
VALUES
```

```
(1, 1, 'Breast Cancer', '2019-02-16', 'Remission'),
(2, 1, 'Uterine Cancer', '2012-06-29', 'Remission'),
(3, 4, 'Kidney Cancer', '2005-05-15', 'Remission'),
(4, 4, 'Lung Cancer', '2007-08-20', 'Active'),
(5, 4, 'Liver Cancer', '2009-07-14', 'Active');
```

--inserting 5 rows of data into allergies table

```
INSERT INTO OurHospital.my_hospital.Allergies (AllergyID, AllergyName, AllergyReaction,
AllergySeverity, PatientID)
```

```
VALUES
```

```
(1, 'Anesthesia', 'Airways close', 'Severe', 3),
(2, 'Penicillin', 'Skin Rash', 'Mild', 1),
(3, 'Dairy', 'Upset Stomach', 'Severe', 2),
(4, 'Peanuts', 'Anaphylaxis', 'Severe', 5),
(5, 'Sulfonamide', 'Hives', 'Mild', 4);
```

--inserting 5 rows of data into the SmokingHistory table

```
INSERT INTO OurHospital.my_hospital.SmokingHistory (SmokingID, PastSmoker,
CurrentSmoker, NumberOfYearsSmoking, PatientID)
```

```
VALUES
```

```
(1, 'No', 'No', 0, 1),
(2, 'Yes', 'No', 5, 2),
(3, 'No', 'No', 0, 3),
(4, 'Yes', 'Yes', 25, 4),
(5, 'No', 'No', 0, 5);
```

--inserting 5 rows into the PatientDiagnosis table

```
INSERT INTO OurHospital.my_hospital.PatientDiagnosis (DiagnosisID, DiagnosisName,
DiagnosisDescription, DiagnosisDate, PatientID)
```

```
VALUES
```

```
(1, 'Flu', 'Patient has flu symptoms', '2023-04-15', 1),
(2, 'Broken Arm', 'Broken arm from fall while playing basketball', '2023-04-20', 2),
(3, 'Migraine', 'Patient has a migraine', '2023-04-22', 3),
(4, 'Pneumonia', 'Symptoms include cough, poor breathing', '2023-04-25', 4),
(5, 'COVID-19', 'Tested positive for COVID-19', '2023-03-20', 5);
```

```
--inserting 5 rows into the PatientInsuranceInformation table
INSERT INTO OurHospital.my_hospital.PatientInsuranceInformation (InsuranceID,
GroupNumber, SubscriberID, Issuer, SubscriberFirstName, SubscriberLastName,
SubscriberDOB, PatientID)
```

```
VALUES
```

```
(1, 1234, 'ABC123', 100, 'Mandy', 'McMillan', '1990-01-01', 1),
(2, 5678, 'DEF456', 200, 'Malek', 'Hussain', '1991-02-14', 2),
(3, 9012, 'GHI789', 300, 'Jannah', 'Hassan', '1995-04-29', 3),
(4, 3456, 'JKL012', 400, 'Josh', 'Johnson', '1960-09-17', 4),
(5, 7890, 'MNO345', 500, 'Savannah', 'Lee', '2001-08-05', 5);
```

```
--inserting 5 rows into the HospitalBills table
```

```
INSERT INTO OurHospital.my_hospital.HospitalBills (BillID, BillDate, TotalAmount, PatientID)
```

```
VALUES
```

```
(1, '2023-04-16', 9000, 1),
(2, '2023-04-21', 2000, 2),
(3, '2023-04-23', 500, 3),
(4, '2023-04-25', 30000, 4),
(5, '2023-04-21', 1000, 5);
```

```
--inserting 5 rows into the Payments table
```

```
INSERT INTO OurHospital.my_hospital.Payments (PaymentID, PaymentDate,
PaymentAmount, BillID)
```

```
VALUES
```

```
(1, '2023-04-17', 3000, 1),
(2, '2023-04-23', 500, 2),
(3, '2023-04-26', 500, 2),
(4, '2023-04-29', 1000, 4),
(5, '2023-04-24', 500, 3);
```

```
--inserting 5 rows into the Bill items table
```

```
INSERT INTO OurHospital.my_hospital.BillItems (ChargeID, ChargeDescription,
ChargeAmount, BillID)
```

```
VALUES
```

```
(1, 'Lab Test', 250, 1),
(2, 'X-ray', 1250, 1),
(3, 'MRI', 1500, 1),
(4, 'Consultation', 1000, 1),
(5, 'Surgery', 5000, 1);
```

```
--inserting 5 rows into the Medications table
```

```
INSERT INTO OurHospital.my_hospital.Medications (MedicationID, MedicationName)
```

VALUES
(1, 'Ibuprofen'),
(2, 'Acetaminophen'),
(3, 'Lisinopril'),
(4, 'Metformin'),
(5, 'Levothyroxine');

--inserting 5 rows into the Physicians table

```
INSERT INTO OurHospital.my_hospital.Physicians (PhysicianID, FirstName, LastName, Specialty, LiscenceNumber, PhoneNumber, HospitalEmail, HireDate)
```

VALUES
(1, 'John', 'Johnson', 'Cardiology', 123456789, NULL, 'jjohnson@ourhospital.com', '2020-01-01'),
(2, 'Jane', 'Smith', 'Urgent', 987654321, NULL, 'jsmith@ourhospital.com', '2019-05-01'),
(3, 'Michael', 'Johnson', 'Oncology', 456123789, NULL, 'mjohnson@ourhospital.com', '2018-07-15'),
(4, 'Elizabeth', 'Brown', 'Neurology', 789654321, NULL, 'ebrown@ourhospital.com', '2021-02-01'),
(5, 'David', 'Lee', 'Surgery', 234567890, NULL, 'dlee@ourhospital.com', '2017-11-01');

--inserting 5 rows into the Patient medications table

```
INSERT INTO OurHospital.my_hospital.PatientMedications (PatientMedID, Dosage, Frequency, PatientID, PhysicianID, MedicationID)
```

VALUES
(1, '600mg', '2x a day', 1, 1, 1),
(2, '300mg', '3x a day', 2, 3, 2),
(3, '20mg', '2x day', 2, 3, 3),
(4, '200mg', '2x a day', 4, 1, 4),
(5, '60mg', '1x a day', 1, 5, 5);

--inserting 5 rows into the hospital procedures table

```
INSERT INTO OurHospital.my_hospital.HospitalProcedures(ProcedureID, ProcedureName, ProcedureDateTime, PatientID, PhysicianID)
```

VALUES
(1, 'Surgery', '2023-04-01 08:00:00', 1, 5),
(2, 'CT scan', '2023-04-02 10:30:00', 2, 3),
(3, 'X-ray', '2023-04-03 11:45:00', 1, 4),
(4, 'MRI', '2023-04-03 13:15:00', 1, 2),
(5, 'Ultrasound', '2023-04-05 15:30:00', 5, 1);

--inserting 5 rows into the procedure notes table

```
INSERT INTO OurHospital.my_hospital.ProcedureNotes (ProcedureNoteID, ProcedureID, NoteContent, NoteDateTime)
```

VALUES

(1, 1, 'Patient responded well to the surgery', '2023-04-01 10:00:00'),
(2, 2, 'CT went well, results pending', '2023-04-02 11:00:00'),
(3, 3, 'X-Ray went well, results pending', '2023-04-03 12:30:00'),
(4, 4, 'MRI went well', '2023-04-03 10:00:00'),
(5, 5, 'Ultrasound went well', '2023-04-05 16:00:00');

--inserting 5 rows into the physician schedule table

INSERT INTO OurHospital.my_hospital.PhysicianSchedule (PhysicianShiftID, StartShiftDateTime, EndShiftDateTime, NumberOfPhysicians, Department)

VALUES

(1, '2023-05-01 08:00:00', '2023-05-01 16:00:00', 2, 'Neurology'),
(2, '2023-05-01 16:00:00', '2023-05-02 00:00:00', 2, 'Neurology'),
(3, '2023-05-02 08:00:00', '2023-05-02 16:00:00', 2, 'Cardiology'),
(4, '2023-05-02 16:00:00', '2023-05-03 00:00:00', 2, 'Cardiology'),
(5, '2023-05-01 08:00:00', '2023-05-01 16:00:00', 3, 'Oncology');

--inserting 5 rows into the PhysicianShiftAssignments

INSERT INTO OurHospital.my_hospital.PhysicianShiftAssignments (ShiftAssignmentID, PhysicianID, PhysicianShiftID)

VALUES

(1, 4, 1),
(2, 4, 2),
(3, 1, 3),
(4, 1, 4),
(5, 3, 5);

--inserting 5 rows into the Lab results table

INSERT INTO OurHospital.my_hospital.LabResults (TestID, PatientID, PhysicianID, TestName, TestResult)

VALUES

(1, 3, 5, 'Blood Sugar', 'Normal'),
(2, 1, 2, 'Hemoglobin A1c', 'High'),
(3, 2, 4, 'Lipid Panel', 'Normal'),
(4, 4, 1, 'WBC', 'Low'),
(5, 5, 3, 'TSH', 'Low');

--inserting 5 rows into the hospital rooms table

INSERT INTO OurHospital.my_hospital.HospitalRooms (RoomID, RoomType, RoomNumber, RoomFloor)

VALUES

(1, 'ER Room', 101, 1),
(2, 'ER Room', 102, 1),

```
(3, 'ER Room', 103, 1),
(4, 'Urgent', 201, 2),
(5, 'Urgent', 202, 2),
(6, 'Urgent', 203, 2),
(7, 'Urgent', 204, 2),
(8, 'Urgent', 205, 2),
(9, 'Urgent', 206, 2),
(10, 'OR', 401, 4);
```

--inserting 5 rows into the Nurses table

```
INSERT INTO OurHospital.my_hospital.Nurses (NurseID, FirstName, LastName,
HospitalEmail, PhoneNumber, Department, JobTitle, HireDate)
VALUES
(1, 'Samuel', 'Smith', 'ssmith@ourhospital.com', NULL, 'Urgent', 'LPN', '2020-01-15'),
(2, 'Mike', 'Samson', 'msamson@ourhospital.com', NULL, 'Oncology', 'LPN', '2019-05-07'),
(3, 'Emily', 'Manchester', 'emanchester@ourhospital.com', NULL, 'Surgery', 'RN', '2021-02-28'),
(4, 'Declan', 'Brown', 'dbrown@ourhospital.com', NULL, 'Neurology', 'RN', '2018-11-01'),
(5, 'Karen', 'Davis', 'kdavis@ourhospital.com', NULL, 'Cardiology', 'RN', '2022-03-20');
```

--insert 5 rows into the Nurse Schedule table

```
INSERT INTO OurHospital.my_hospital.NurseSchedule (NurseShiftID, StartShiftDateTime,
EndShiftDateTime, NumberOfNurses, Department)
VALUES
(1, '2023-05-01 08:00:00', '2023-05-01 16:00:00', 2, 'Urgent'),
(2, '2023-05-01 16:00:00', '2023-05-02 00:00:00', 2, 'Oncology'),
(3, '2023-05-02 08:00:00', '2023-05-02 16:00:00', 2, 'Surgery'),
(4, '2023-05-02 16:00:00', '2023-05-03 00:00:00', 2, 'Neurology'),
(5, '2023-05-01 08:00:00', '2023-05-01 16:00:00', 3, 'Cardiology');
```

--insert 5 rows into the Nurse shift assignments

```
INSERT INTO OurHospital.my_hospital.NurseShiftAssignments (ShiftAssignmentID,
NurseShiftID, NurseID)
VALUES
(1, 1, 1),
(2, 2, 2),
(3, 3, 3),
(4, 4, 4),
(5, 5, 5);
```

--inserts 5 rows into the Appointments

```
INSERT INTO OurHospital.my_hospital.Appointments (AppointmentID, PhysicianID,
AppointmentDateTime, AppointmentReason, NurseID, RoomID)
```

VALUES

(1, 1, '2023-05-10 10:00:00', 'Annual Checkup', 1, 4),
(2, 2, '2023-05-15 14:30:00', 'Sore Throat', 2, 5),
(3, 1, '2023-05-18 09:15:00', 'High Blood Pressure', 3, 6),
(4, 5, '2023-05-22 11:45:00', 'Abdominal Pain', 4, 7),
(5, 2, '2023-05-29 16:00:00', 'Flu Symptoms', 5, 8);

--inserts 5 rows into the Patient Appointments table

INSERT INTO OurHospital.my_hospital.PatientAppointments (PatientID, AppointmentID)

VALUES

(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5);

--inserts 5 rows into the Admissions table

INSERT INTO OurHospital.my_hospital.Admissions (AdmissionID, RoomID,
AdmissionDateTime, DischargeDateTime, PhysicianID, PatientID)

VALUES

(1, 4, '2023-04-15 10:00:00', '2023-04-15 11:00:00', 2, 1),
(2, 10, '2023-04-20 14:00:00', '2023-02-16 18:00:00', 5, 2),
(3, 6, '2023-04-22 10:00:00', '2022-04-22 10:30:00', 4, 3),
(4, 7, '2023-04-25 09:00:00', '2023-04-25 11:00:00', 1, 4),
(5, 8, '2022-03-20 14:00:00', '2022-03-20 15:00:00', 2, 5);

Testing of All Created Objects

View Testing:

- 1) This test case was very simple and just wanted to see if the view returned the correct information based on how we populated the database previously. Below is the source code for this simple test used to test the FullPatientSummaryView:

```
USE OurHospital;
GO
```

```
/*Testing to see if the FullPatientSummary view works correctly which it does it brings all the patients prior medical history*/
SELECT *
FROM my_hospital.FullPatientSummary
```

- 2) This test case was for the PatientBilling view and wanted to see if the view output the correct resulting table for one patient's billing information. The source code for the test case is shown below:

```
USE OurHospital;
GO
```

```
/*Testing the view that brings all a patient's bills, it is successful as we specified one patient and it brought all their bills with itemized charges*/
SELECT
FROM my_hospital.PatientBilling
WHERE PatientID = 1;
```

- 3) This test case was for the LabResultsAndDiagnosisView. The test case just wanted to make sure that the view pulled the correct information for the patient specified in the test case. This test case revealed that the view should have been coded as a union rather than join such that the resulting table would not be confusing like it was in this test case:

```
USE OurHospital;
GO
```

```
/*Testing the view that brings all a patient's lab results and diagnosis, in hindsight this should have been a union rather than a join because then the data would be formatted correctly in the resulting table */
SELECT *
FROM my_hospital.LabResultsAndDiagnosisView;
```

-
- 4) The last test case for the views includes 2 parts and was for the PatientProceduresView. This test case was intended to make sure that the view pulls only the specific patient's procedures:

USE OurHospital;

GO

/*Testing the view that brings all the procedures associated with a patient, first test is without a patient specified*/

SELECT *

FROM my hospital.PatientProcedures;

/*Second test is with patientID = 1 specified*/

SELECT *

FROM my hospital.PatientProcedures

WHERE PatientID = 1;

Stored Procedures Testing:

- 1) Test case to test the stored procedure spAddNewPatient by trying to add a new patient that has a date of birth that is in the future, the stored procedure was successful as it made sure to throw an error. Below is the source code for the test case:

USE OurHospital;

GO

/*Testing the stored procedure by trying to add a new patient that has a date of birth that is in the future*/

@EXEC my_hospital.spAddNewPatient

@PatientID = 6,

@FirstName = 'Johnny',

@LastName = 'Templton',

@Email = 'jrocket@hotmail.com',

@PhoneNumber = NULL,

@DateOfBirth = '2023-06-20',

@Sex = 'M',

@Race = 'White'

-
- 2) Testing the stored procedure spAddDiagnosis by trying to add a new patient diagnosis for a PatientID that is not in the Patients table and is thus invalid. Test was successful as the stored procedure threw an error stating that the patient does not exist in the Patients table. Below is the source code for the test case:

USE OurHospital;

GO

/*Testing the stored procedure by trying to add a new patient diagnosis for an invalid patient id*/

EXEC my_hospital.spAddDiagnosis

```
@PatientID = 6,
@DiagnosisName = 'Anemia',
@DiagnosisDescription = 'Low iron from labs',
@DiagnosisDate = '2023-04-30'
```

- 3) Testing the stored procedure spUpdatePatientPersonalInfo by trying to update the information of a patient that doesn't exist. Test case was successful as the stored procedure threw an error making sure that didn't happen. Below is the source code for the test case:

USE OurHospital;

GO

/*Testing the stored procedure by trying to add update the information of a patient that doesn't exist*/

EXEC my_hospital.spUpdatePatientPersonalInfo

```
@PatientID = 10,
@FirstName = 'John',
@LastName = 'Doe',
@Email = NULL,
@PhoneNumber = NULL,
@DateOfBirth = '1999-11-30',
@Sex = 'M',
@Race = 'White'
```

- 4) Testing the stored procedure spAddNewPatientAppointment by trying to add an appointment with a physician that already has an appointment at that time. The test case was successful because the stored procedure threw the correct error. Below is the source code for the test case:

```
USE OurHospital;
GO

/*Testing the stored procedure by trying to add an appointment with a physician that already has an appointent at that time*/
EXEC my_hospital.AddNewPatientAppointment
    @PatientID = 2,
    @AppointmentDateTime = '2023-05-10 10:00:00',
    @AppointmentReason = 'Sick',
    @NurseID = 1,
    @RoomID = 4,
    @PhysicianID = 1,
    @AppointmentID = 6
```

User-Defined Functions Testing:

- 1) Straightforward test case for fnRemainingBillBalance, test case was successful. Below is the source code:

```
/*Testing user defined function and correct output for this patient should be $3000*/
PRINT 'Balance due for Mandy McMillan is $' + CONVERT(varchar, my_hospital.fnRemainingBillBalance(1, 1));
```

- 2) Simple test case to make sure fnGetPatientAdmissionHistory returns the correct rows for the patient id specified. Test case was successful. Below is the source code:

```
/*Testing user defined function and correct output for this patient should be their one admission in a table*/
SELECT *
FROM my_hospital.fnGetPatientAdmissionHistory(1);
```

- 3) Similar test case to the previous test case was used for fnGetAllPhysicianAppointments for the specified physician id. Test case was successful and source code is shown below:

```
/*Testing user defined function and correct output for this physician should be the two appointments that she has scheduled right now */
SELECT *
FROM my_hospital.fnGetAllPhysicianAppointments(2);
```

- 4) Last test case is similar to testcases 2 and 3 to test fnGetPatientLabHistory. Test case was successful and the source code is shown below:

```
/*Testing user defined function and correct output for this should be the one lab that
patientID = 5 had which is the TSH lab*/

SELECT *
FROM my_hospital.fnGetPatientLabHistory(5);
```

Trigger Testing:

- 1) Test case for the trigger trPreventSurgeryWithAllergy. Test case threw error so it was successful and the source code is shown below:

*/*Testing to make sure the trigger trPreventSurgeryWithAllergy does not let us schedule a patient for a surgery if they have a severe allergy to anesthesia*/*

```
INSERT INTO OurHospital.my_hospital.HospitalProcedures(ProcedureID,
ProcedureName, ProcedureDateTime, PatientID, PhysicianID)

VALUES(6, 'Surgery', '2023-06-21', 3, 5);
```

- 2) Test case for the trigger trPreventDoubleBooking. Test case threw error so it was successful and the source code is shown below:

*/*Testing to make sure the trigger trPreventDoubleBooking does not let us double book a physician if the already have a patient appointment at the same*/*

```
INSERT INTO OurHospital.my_hospital.Appointments (AppointmentID, PhysicianID,
AppointmentDateTime, AppointmentReason, NurseID, RoomID)
```

```
VALUES(6, 1, '2023-04-16 09:15:00', 'Sick', 1, 1);
```

- 3) Test case for the trigger trPreventDoubleRoomOccupancy. Test case allowed us to insert the row since it did not lead to a double booking of a room. It was successful and the source code is shown below:

*/*Testing to make sure the trigger trPreventDoubleRoomOccupancy works by letting us insert a new admission if it does not lead to a double book*/*

```
INSERT INTO OurHospital.my_hospital.Admissions (AdmissionID, RoomID,
AdmissionDateTime, DischargeDateTime, PhysicianID, PatientID)
```

```
VALUES(6, 9, '2023-04-16 10:00:00', '2023-04-16 11:00:00', 2, 1);
```

-
- 4) Test case for the trigger trPreventMedicationAllergy. Test case inserted a new allergy into the allergy table for the patient with patient id of 3. Then the test case tried to add the medication that the patient has an allergy of, to the patient's medication list. The test case was successful because the trigger preventing this insertion from happening. The source code is shown below:

/*Testing to make sure the trigger trPreventMedicationAllergy by making sure it does not let us add a medication to a patient who has an allergy to the medication */

```
/*First adding new allergy for PatientID 3 */
INSERT INTO OurHospital.my_hospital.Allergies (AllergyID, AllergyName,
AllergyReaction, AllergySeverity, PatientID)
VALUES (6, 'Metformin', 'Fatal', 'Severe', 3);
GO

/*now trying to insert what the patient is allergic to, should throw error via the trigger*/
INSERT INTO OurHospital.my_hospital.PatientMedications(PatientMedID, Dosage,
Frequency, PatientID, PhysicianID, MedicationID)
VALUES (6, '300mg', '1x per day', 3, 1, 4);
GO
```

Conclusions

Overall, I felt that this project was detail oriented and heavy. I felt that the project itself was a great way to end the semester because it truly put all of what we have learned in class into a real project that we had to implement and oversee every part of. The project was harder for me than I expected I think that doing what Professor Ercanli had said during the semester and creating a cheat sheet of all the mistakes you ever made so you don't make them again is actually really important and I feel if I had done that throughout the semester I would have been better prepared for this project. Other than that I feel that I performed well on this project but with more time and energy I can make it a much more seamless process. I think it would have also been cool to populate the tables with more data to see what other kinds of meaningful data we could extract from a database design like this. I'm really glad I was able to have the opportunity to do a project like this because I feel like it also gave me more insight into what is actually done in the field of data science and what a future job might look like for me.

Appendix A: Implementation Screenshots

Creating database and schema:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the 'OurHospital' database structure. The central pane displays T-SQL code for creating a database and a schema:

```
CREATE DATABASE OurHospital; --creating the database for our schema and tables
GO

USE OurHospital; --using that database to house the schema
GO

CREATE SCHEMA my_hospital; --creating the schema
GO
```

The 'Messages' pane at the bottom indicates that the commands completed successfully.

Creating all tables:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the 'master' database structure. The central pane displays T-SQL code for creating four tables:

```
--Table 1
CREATE TABLE OurHospital.my_hospital.Patients(
    PatientID smallint PRIMARY KEY NOT NULL,
    FirstName varchar(20) NOT NULL,
    LastName varchar(20) NOT NULL,
    Email varchar(30) NULL,
    PhoneNumber int NULL,
    DateOfBirth date NOT NULL,
    Sex varchar(1) NOT NULL,
    Race varchar(10) NULL
);
GO

--Table 2
CREATE TABLE OurHospital.my_hospital.SurgeryHistory(
    PastSurgeryID smallint PRIMARY KEY NOT NULL,
    SurgeryName varchar(20) NOT NULL,
    SurgeryDate date NULL,
    PatientID smallint NOT NULL,
    CONSTRAINT SurgeryHistoryFK_PatientID
        FOREIGN KEY (PatientID) REFERENCES OurHospital.my_hospital.Patients(PatientID)
);
GO

--Table 3
CREATE TABLE OurHospital.my_hospital.CancerHistory(
    PastCancerID smallint PRIMARY KEY NOT NULL,
    PatientID smallint NOT NULL,
    CancerType varchar(20) NOT NULL,
    CancerDiagnosisDate date NULL,
    CancerTreatmentStatus tinyint NULL,
    CONSTRAINT CancerHistoryFK_PatientID
        FOREIGN KEY (PatientID) REFERENCES OurHospital.my_hospital.Patients(PatientID)
);
GO

--Table 4
```

The 'Messages' pane at the bottom indicates that the commands completed successfully.

Project 2 Creating Tables.sql - MALEK\SQLEXPRESS.master (MALEK\malek (83)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

master Execute

Object Explorer

```
--Table 4
CREATE TABLE OurHospital.my_hospital.Allergies(
    AllergyID smallint PRIMARY KEY NOT NULL,
    AllergyName varchar(20) NOT NULL,
    AllergyReaction varchar(20) NULL,
    AllergySeverity varchar(10) NULL,
    PatientID smallint NOT NULL,
    CONSTRAINT AllergiesFK_PatientID
        FOREIGN KEY (PatientID) REFERENCES OurHospital.my_hospital.Patients(PatientID)
);
GO

--Table 5
CREATE TABLE OurHospital.my_hospital.SmokingHistory(
    SmokingID smallint PRIMARY KEY NOT NULL,
    PastSmoker varchar(5) NOT NULL,
    CurrentSmoker varchar(5) NOT NULL,
    NumberOfYearsSmoking smallint NULL,
    PatientID smallint NOT NULL,
    CONSTRAINT SmokingHistoryFK_PatientID
        FOREIGN KEY (PatientID) REFERENCES OurHospital.my_hospital.Patients(PatientID)
);
GO

--Table 6
CREATE TABLE OurHospital.my_hospital.PatientDiagnosis(
    DiagnosisID smallint PRIMARY KEY NOT NULL,
    DiagnosisName varchar(20) NULL,
    DiagnosisDescription varchar(50) NOT NULL,
    DiagnosisDate date NOT NULL,
    PatientID smallint NOT NULL,
    CONSTRAINT PatientDiagnosisFK_PatientID
        FOREIGN KEY (PatientID) REFERENCES OurHospital.my_hospital.Patients(PatientID)
);
GO
```

70 %

Messages

Commands completed successfully.

Completion time: 2023-04-29T12:23:03.8109418-04:00

Query execute

Project 2 Creating...(MALEK\malek (83)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

master Execute

Object Explorer

```
--Table 7
CREATE TABLE OurHospital.my_hospital.PatientInsuranceInformation(
    InsuranceID smallint PRIMARY KEY NOT NULL,
    GroupNumber smallint NOT NULL,
    SubscriberID varchar(30) NOT NULL,
    Issue int NOT NULL,
    SubscriberFirstName varchar(10) NOT NULL,
    SubscriberLastName varchar(10) NOT NULL,
    SubscriberDOD date NOT NULL,
    PatientID smallint NOT NULL,
    CONSTRAINT PatientInsuranceInformationFK_PatientID
        FOREIGN KEY (PatientID) REFERENCES OurHospital.my_hospital.Patients(PatientID)
);
GO

--Table 8
CREATE TABLE OurHospital.my_hospital.HospitalBills(
    BillID smallint PRIMARY KEY NOT NULL,
    BillDate date NOT NULL,
    TotalAmount smallint NOT NULL,
    PatientID smallint NOT NULL,
    CONSTRAINT HospitalBillsFK_PatientID
        FOREIGN KEY (PatientID) REFERENCES OurHospital.my_hospital.Patients(PatientID)
);
GO

--Table 9
CREATE TABLE OurHospital.my_hospital.Payments(
    PaymentID smallint PRIMARY KEY NOT NULL,
    PaymentDate date NOT NULL,
    PaymentAmount smallint NOT NULL,
    BillID smallint NOT NULL,
    CONSTRAINT PaymentsFK_BillID
        FOREIGN KEY (BillID) REFERENCES OurHospital.my_hospital.HospitalBills(BillID)
);
GO
```

70 %

Messages

Commands completed successfully.

Completion time: 2023-04-29T12:23:03.8109418-04:00

Query executed successfully.

Project 2 Creating Tables.sql - MALEK\SQLEXPRESS.master (MALEK\malek (83)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

New Query MDX DMX XML DAX Execute

master

Object Explorer

```
--Table 10
CREATE TABLE OurHospital.my_hospital.BillItems(
    ChargeID smallint PRIMARY KEY NOT NULL,
    ChargeDescription varchar(30) NOT NULL,
    ChargeAmount smallint NOT NULL,
    BillID smallint NOT NULL,
    CONSTRAINT BillItemsFK_BillID
        FOREIGN KEY (BillID) REFERENCES OurHospital.my_hospital.HospitalBills(BillID)
);
GO

--Table 11
CREATE TABLE OurHospital.my_hospital.Medications(
    MedicationID smallint PRIMARY KEY NOT NULL,
    MedicationName varchar(20) NOT NULL
);
GO

--Table 12
CREATE TABLE OurHospital.my_hospital.Physicians(
    PhysicianID smallint PRIMARY KEY NOT NULL,
    FirstName varchar(20) NOT NULL,
    LastName varchar(20) NOT NULL,
    Specialty varchar(20) NOT NULL,
    LiscenseNumber int NOT NULL,
    PhoneNumber int NULL,
    HospitalEmail varchar(30) NOT NULL,
    HireDate date NOT NULL
);
GO
```

70% Messages

Commands completed successfully.

Completion time: 2023-04-28T12:23:03.8109418-04:00

Project 2 Creating...(MALEK\malek (83))

File Edit View Query Project Tools Window Help

New Query MDX DMX XML DAX Execute

master

Object Explorer

```
--Table 13
CREATE TABLE OurHospital.my_hospital.PatientMedications(
    PatientMedID smallint PRIMARY KEY NOT NULL,
    Dosage varchar(20) NOT NULL,
    Frequency varchar(20) NOT NULL,
    PatientID smallint NOT NULL,
    PhysicianID smallint NOT NULL,
    MedicationID smallint NOT NULL,
    CONSTRAINT PatientMedicationsFK_PatientID
        FOREIGN KEY (PatientID) REFERENCES OurHospital.my_hospital.Patients(PatientID),
    CONSTRAINT PatientMedicationsFK_PhysicianID
        FOREIGN KEY (PhysicianID) REFERENCES OurHospital.my_hospital.Physicians(PhysicianID),
    CONSTRAINT PatientMedicationsFK_MedicationID
        FOREIGN KEY (MedicationID) REFERENCES OurHospital.my_hospital.Medications(MedicationID)
);
GO

--Table 14
CREATE TABLE OurHospital.my_hospital.HospitalProcedures(
    ProcedureID smallint PRIMARY KEY NOT NULL,
    ProcedureName varchar(15) NOT NULL,
    ProcedureDateTime smalldatetime NOT NULL,
    PatientID smallint NOT NULL,
    PhysicianID smallint NOT NULL,
    CONSTRAINT HospitalProceduresFK_PatientID
        FOREIGN KEY (PatientID) REFERENCES OurHospital.my_hospital.Patients(PatientID),
    CONSTRAINT HospitalProceduresFK_PhysicianID
        FOREIGN KEY (PhysicianID) REFERENCES OurHospital.my_hospital.Physicians(PhysicianID)
);
GO

--Table 15
CREATE TABLE OurHospital.my_hospital.ProcedureNotes(
    ProcedureNoteID smallint PRIMARY KEY NOT NULL,
    ProcedureID smallint NOT NULL,
    NoteContent varchar(100) NOT NULL,
    NoteDateTime datetime NOT NULL,
    NoteAuthorID smallint NOT NULL,
    CONSTRAINT ProcedureNotesFK_ProcedureID
        FOREIGN KEY (ProcedureID) REFERENCES OurHospital.my_hospital.HospitalProcedures(ProcedureID)
);
GO
```

70% Messages

Commands completed successfully.

70% Messages

Query executed successfully.

Project 2 Creating Tables.sql - MALEK\SQLEXPRESS.master (MALEK\malek (83)) - Microsoft SQL Server Management Studio

```

--Table 16
CREATE TABLE OurHospital.my_hospital.PhysicianSchedule(
    PhysicianShiftID smallint PRIMARY KEY NOT NULL,
    StartShiftDateTime datetime NOT NULL,
    EndShiftDateTime datetime NOT NULL,
    NumberOfPhysicians smallint NOT NULL,
    Department varchar(20) NOT NULL,
);
GO

--Table 17
CREATE TABLE OurHospital.my_hospital.PhysicianShiftAssignments(
    ShiftAssignmentID smallint PRIMARY KEY NOT NULL,
    PhysicianID smallint NOT NULL,
    PhysicianShiftID smallint NOT NULL,
    CONSTRAINT PhysicianShiftAssignmentsFK_PhysicianID
        FOREIGN KEY (PhysicianID) REFERENCES OurHospital.my_hospital.Physicians(PhysicianID),
    CONSTRAINT PhysicianShiftAssignmentsFK_PhysicianShiftID
        FOREIGN KEY (PhysicianShiftID) REFERENCES OurHospital.my_hospital.PhysicianSchedule(PhysicianShiftID)
);
GO

--Table 18
CREATE TABLE OurHospital.my_hospital.LabResults(
    TestID smallint PRIMARY KEY NOT NULL,
    PatientID smallint NOT NULL,
    PhysicianID smallint NOT NULL,
    TestName varchar(20) NOT NULL,
    TestResult varchar(20) NULL,
    CONSTRAINT LabResultsFK_PhysicianID
        FOREIGN KEY (PhysicianID) REFERENCES OurHospital.my_hospital.Physicians(PhysicianID),
    CONSTRAINT LabResultsFK_PatientID
        FOREIGN KEY (PatientID) REFERENCES OurHospital.my_hospital.Patients(PatientID)
);
GO

```

70 %

Messages

Commands completed successfully.

Completion time: 2023-04-29T12:23:03.8109418-04:00

Project 2 Creating...(MALEK\malek (83))

```

--Table 19
CREATE TABLE OurHospital.my_hospital.HospitalRooms(
    RoomID smallint PRIMARY KEY NOT NULL,
    RoomType varchar(10) NOT NULL,
    RoomNumber smallint NOT NULL,
    RoomFloor smallint NOT NULL
);
GO

--Table 20
CREATE TABLE OurHospital.my_hospital.Nurses(
    NurseID smallint PRIMARY KEY NOT NULL,
    FirstName varchar(20) NOT NULL,
    LastName varchar(20) NOT NULL,
    HospitalEmail varchar(30) NOT NULL,
    PhoneNumber int NULL,
    Department varchar(20) NOT NULL,
    JobTitle varchar(20) NOT NULL,
    HireDate date NOT NULL
);
GO

--Table 21
CREATE TABLE OurHospital.my_hospital.NurseSchedule(
    NurseShiftID smallint PRIMARY KEY NOT NULL,
    StartShiftDateTime datetime NOT NULL,
    EndShiftDateTime datetime NOT NULL,
    NumberOfNurses smallint NOT NULL,
    Department varchar(20) NOT NULL
);
GO

--Table 22
CREATE TABLE OurHospital.my_hospital.NurseShiftAssignments(
    ShiftAssignmentID smallint PRIMARY KEY NOT NULL,
    NurseShiftID smallint NOT NULL,
    CONSTRAINT NurseShiftAssignmentsFK_NurseShiftID
        FOREIGN KEY (NurseShiftID) REFERENCES OurHospital.my_hospital.NurseSchedule(NurseShiftID)
);
GO

```

70 %

Messages

Commands completed successfully.

Completion time: 2023-04-29T12:23:03.8109418-04:00

70 %

Query executed successfully.

Project 2 Creating Tables.sql - MALEK\SQLEXPRESS.master (MALEK\malek (83)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

master Execute

Object Explorer

```
--Table 24
CREATE TABLE OurHospital.my_hospital.PatientAppointments(
    PatientID smallint NOT NULL,
    AppointmentID smallint NOT NULL,
    CONSTRAINT PatientAppointmentsPK PRIMARY KEY (PatientID, AppointmentID),
    CONSTRAINT PatientAppointmentsFK_PatientID FOREIGN KEY (PatientID) REFERENCES OurHospital.my_hospital.Patients(PatientID),
    CONSTRAINT PatientAppointmentsFK_AppointmentID FOREIGN KEY (AppointmentID) REFERENCES OurHospital.my_hospital.Appointments(AppointmentID)
);
GO

--Table 25
CREATE TABLE OurHospital.my_hospital.Admissions(
    AdmissionID smallint PRIMARY KEY NOT NULL,
    RoomID smallint NOT NULL,
    AdmissionDateTime datetime NOT NULL,
    DischargeDateTime datetime NOT NULL,
    PhysicianID smallint NOT NULL,
    PatientID smallint NOT NULL,
    CONSTRAINT AdmissionsPK_RoomID FOREIGN KEY (RoomID) REFERENCES OurHospital.my_hospital.HospitalRooms(RoomID),
    CONSTRAINT AdmissionsPK_PhysicianID FOREIGN KEY (PhysicianID) REFERENCES OurHospital.my_hospital.Physicians(PhysicianID),
    CONSTRAINT AdmissionsPK_PatientID FOREIGN KEY (PatientID) REFERENCES OurHospital.my_hospital.Patients(PatientID)
);
GO
```

70 %

Messages

Commands completed successfully.

Completion time: 2023-04-29T12:23:03.8109418-04:00

70 %

Query executed successfully.

Object Explorer

- + Database Diagrams
- + Tables
 - + System Tables
 - + FileTables
 - + External Tables
 - + Graph Tables
 - + my_hospital.Admissions
 - + my_hospital.Allergies
 - + my_hospital.Appointments
 - + my_hospital.BillItems
 - + my_hospital.CancerHis
 - + my_hospital.HospitalBills
 - + my_hospital.HospitalProcedures
 - + my_hospital.LabResult
 - + my_hospital.Medicatio
 - + my_hospital.Nurses
 - + my_hospital.NurseSch
 - + my_hospital.NurseShift
 - + my_hospital.PatientAp
 - + my_hospital.PatientDiag
 - + my_hospital.PatientIns
 - + my_hospital.PatientMed
 - + my_hospital.Patients
 - + my_hospital.Payments
 - + my_hospital.Physicians
 - + my_hospital.PhysicianS
 - + my_hospital.Procedure
 - + my_hospital.SmokingH
 - + my_hospital.SurgeryHi
- + Views

Creating All Stored Procedures:

The screenshot shows the SQL Server Management Studio interface with two tabs: 'Project 2 Creating...I (MALEK\malek (87))' and 'Project 2 Creating...(MALEK\malek (85))'. The code in the editor is as follows:

```

USE OurHospital;
GO

CREATE PROCEDURE my_hospital.AddNewPatientAppointment
    @PatientID smallint,
    @AppointmentDateTime datetime,
    @AppointmentReason varchar(50),
    @NurseID smallint,
    @RoomID smallint,
    @PhysicianID smallint,
    @AppointmentID smallint
AS
BEGIN
    --check that NurseID is valid
    IF NOT EXISTS (SELECT 1 FROM my_hospital.Nurses WHERE NurseID = @NurseID)
        THROW 520021, 'Invalid NurseID, NurseID does not exist in the Nurses table', 1;

    --check that RoomID is valid
    IF NOT EXISTS (SELECT 1 FROM my_hospital.HospitalRooms WHERE @RoomID = RoomID)
        THROW 520022, 'Invalid RoomID, RoomID does not exist in the HospitalRooms table', 1;

    --check that PatientID is valid
    IF NOT EXISTS (SELECT 1 FROM my_hospital.Patients WHERE PatientID = @PatientID)
        THROW 520023, 'Invalid PatientID, PatientID does not exist in the Patients table', 1;

    --check that PhysicianID is valid
    IF NOT EXISTS (SELECT 1 FROM my_hospital.Physicians WHERE PhysicianID = @PhysicianID)
        THROW 520024, 'Invalid PhysicianID, PhysicianID does not exist in the Physicians table', 1;

    --check that AppointmentID does not already exist
    IF EXISTS (SELECT 1 FROM my_hospital.Appointments WHERE AppointmentID = @AppointmentID)
        THROW 520025, 'AppointmentID already exists', 1;

    --check if appointment reason is long enough
    IF (LEN(@AppointmentReason) > 50)
        THROW 520026, 'Invalid data, appointment reason has to be no greater than 50 characters', 1;

    IF (@AppointmentDateTime < GETDATE())
        THROW 520027, 'Invalid appointment date, appointment cannot be made in the past', 1;

    --insert new appointment data into appointments table
    INSERT INTO my_hospital.Appointments (AppointmentID, AppointmentReason, NurseID, RoomID, PhysicianID, AppointmentTime)
    VALUES (@AppointmentID, @AppointmentReason, @NurseID, @RoomID, @PhysicianID, @AppointmentDateTime);

    --insert new record into patient appointments table
    INSERT INTO my_hospital.PatientAppointments (PatientID, AppointmentID)
    VALUES (@PatientID, @AppointmentID);
END

```

The status bar at the bottom indicates 'Query executed successfully.' and '0 rows'.

The screenshot shows the SQL Server Management Studio interface with two tabs: 'Project 2 Creating...I (MALEK\malek (87))' and 'Project 2 Creating...(MALEK\malek (85))'. The code in the editor is as follows:

```

USE OurHospital;
GO

CREATE PROCEDURE my_hospital.spAddNewPatient
    @PatientID smallint,
    @firstName varchar(20),
    @lastName varchar(20),
    @Email varchar(30) = NULL,
    @PhoneNumber int = NULL,
    @DateOfBirth date,
    @Sex varchar(1),
    @Race varchar(10) = NULL
AS
BEGIN
    --check to make sure fields that are not nullable are filled
    IF (@PatientID IS NULL OR @firstName IS NULL OR @lastName IS NULL OR @DateOfBirth IS NULL OR @Sex IS NULL)
        THROW 520004, 'Incomplete input data. PatientID, FirstName, LastName, DateOfBirth and Sex are required.', 1;
    --check to make sure length of first name and last name is correct
    IF (LEN(@FirstName) > 20 OR LEN(@LastName) > 20)
        THROW 520005, 'Invalid input data. Firstname and LastName must be less than or equal to 20 characters', 1;
    --check to make sure length of email is correct
    IF (LEN(@Email) > 30)
        THROW 520006, 'Invalid input data. Email must be less than or equal to 30 characters', 1;
    --check to make sure length of sex is correct
    IF (LEN(@Sex) > 1)
        THROW 520007, 'Invalid input data. Sex must only be one character, M for Male and F for female', 1;
    --check to make sure race is the correct length if it wasn't null
    IF (@RACE IS NOT NULL AND LEN(@Race) > 10)
        THROW 520008, 'Invalid input data. Race must be less than or equal to 10 characters', 1;
    --check to make sure date of birth is valid
    IF (@DateOfBirth > GETDATE())
        THROW 520009, 'Invalid input data. Date of birth cannot be in the future.', 1;

    INSERT INTO my_hospital.Patients (PatientID, FirstName, LastName, Email, PhoneNumber, DateOfBirth, Sex, Race)
    VALUES (@PatientID, @FirstName, @LastName, @Email, @PhoneNumber, @DateOfBirth, @Sex, @Race)
END
GO

```

The status bar at the bottom indicates 'Query executed successfully.' and '0 rows'.

Project 2 Creating...I (MALEK\malek (87)) * Project 2 Creating...(MALEK\malek (85))

```

USE OurHospital;
GO

CREATE PROCEDURE my_hospital.spAddDiagnosis
    @PatientID smallint,
    @DiagnosisName varchar(20) = NULL,
    @DiagnosisDescription varchar(50),
    @DiagnosisDate date
AS
BEGIN
    --check to see if patientID is valid
    IF NOT EXISTS (SELECT 1 FROM my_hospital.Patients WHERE PatientID = @PatientID)
        THROW 5200010, 'Invalid PatientID. The patient does not exist in the Patients table.', 1;
    --check to see if diagnosis date is valid
    IF (@DiagnosisDate > GETDATE())
        THROW 520011, 'Invalid DiagnosisDate. The date cannot be in the future.', 1;
    --check to see if diagnosis name length is valid if its not null
    IF (@DiagnosisName IS NOT NULL AND LEN(@DiagnosisName) > 20)
        THROW 520012, 'Invalid DiagnosisName. The name cannot be greater than 20 characters.', 1;
    --check to see if diagnosis description is the correct length
    IF (LEN(@DiagnosisDescription) > 50)
        THROW 520013, 'Invalid DiagnosisDescription. The description cannot be greater than 50 characters.', 1;

    INSERT INTO my_hospital.PatientDiagnosis (DiagnosisName, DiagnosisDescription, DiagnosisDate, PatientID)
    VALUES (@DiagnosisName, @DiagnosisDescription, @DiagnosisDate, @PatientID)
END
GO

```

80 %

Messages

Commands completed successfully.

Completion time: 2023-04-30T00:27:54.4668894-04:00

80 %

Query executed successfully. MALEK\SQLEXPRESS (16.0 RTM) MALEK\malek (87) OurHospital 00:00:00 0 rows

Project 2 Creating...I (MALEK\malek (87)) * Project 2 Creating...(MALEK\malek (85))

```

USE OurHospital;
GO

CREATE PROCEDURE my_hospital.spUpdatePatientPersonalInfo
    @PatientID smallint = NULL,
    @FirstName varchar(20) = NULL,
    @LastName varchar(20) = NULL,
    @Email varchar(30) = NULL,
    @PhoneNumber int = NULL,
    @DateOfBirth date = NULL,
    @Sex varchar(1) = NULL,
    @Race varchar(10) = NULL
AS
BEGIN
    --check to make sure fields that are not nullable are filled
    IF (@PatientID IS NULL OR @FirstName IS NULL OR @LastName IS NULL OR @DateOfBirth IS NULL OR @Sex IS NULL)
        THROW 520014, 'Incomplete input data. PatientID, FirstName, LastName, DateOfBirth and Sex are required.', 1;
    --check to make sure length of first name and last name is correct
    IF (LEN(@FirstName) > 20 OR LEN(@LastName) > 20)
        THROW 520015, 'Invalid input data. FirstName and LastName must be less than or equal to 20 characters.', 1;
    --check to make sure length of email is correct
    IF (LEN(@Email) > 30)
        THROW 520016, 'Invalid input data. Email must be less than or equal to 30 characters.', 1;
    --check to make sure length of sex is correct
    IF (LEN(@Sex) > 1)
        THROW 520017, 'Invalid input data. Sex must only be one character, M for Male and F for female.', 1;
    --check to make sure race is the correct length if it wasn't null
    IF (@Race IS NOT NULL AND LEN(@Race) > 10)
        THROW 520018, 'Invalid input data. Race must be less than or equal to 10 characters.', 1;
    --check to make sure date of birth is valid
    IF (@DateOfBirth > GETDATE())
        THROW 520019, 'Invalid input data. Date of birth cannot be in the future.', 1;
    -- check if patient ID exists in the Patients table
    IF NOT EXISTS(SELECT * FROM my_hospital.Patients WHERE PatientID = @PatientID)
        THROW 520020, 'Invalid input data. PatientID does not exist in the Patients table.', 1;

    -- update the patient information
    UPDATE my_hospital.Patients
    SET FirstName = @FirstName,
        LastName = @LastName,
        Email = @Email,
        PhoneNumber = @PhoneNumber,
        DateOfBirth = @DateOfBirth,
        Sex = @Sex,
        Race = @Race
    WHERE PatientID = @PatientID;
END
GO

```

60 %

Messages

Commands completed successfully.

Completion time: 2023-04-30T00:40:04.0433639-04:00

60 %

Query executed successfully. MALEK\SQLEXPRESS (16.0 RTM) MALEK\malek (87) OurHospital 00:00:00 0 rows

Creating All the Triggers:

Project 2 Creating Triggers.sql - MALEK\SQLEXPRESS.OurHospital (MALEK\malek (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

OurHospital Execute New Query

Object Explorer

Project 2 Creating... (MALEK\malek (52)) Project 2 Creating... (MALEK\malek (71)) Project 2 Creating... (MALEK\malek (87)) Project 2 Creating... (MALEK\malek (85))

```
USE OurHospital;
GO

/*Creating a trigger to rollback the insertion of a procedure into the hospital procedures table if a patient has a severe allergy to anesthesia*/
CREATE TRIGGER trPreventSurgeryWithAllergy
    ON my_hospital.HospitalProcedures
    INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM Inserted JOIN my_hospital.Allergies ON Allergies.PatientID = inserted.PatientID
        WHERE Allergies.AllergyName = 'Anesthesia' AND Allergies.AllergySeverity = 'Severe'
    )
    BEGIN
        ROLLBACK TRANSACTION;
        THROW 500030, 'This patient has a severe allergy to anesthesia and cannot have surgery scheduled', 1;
    END
    ELSE
        INSERT INTO my_hospital.HospitalProcedures
        SELECT * FROM inserted;
        COMMIT TRANSACTION;
    END
```

100 %

Messages

Commands completed successfully.

Completion time: 2023-04-30T12:37:56.0547750-04:00

Project 2 Creating Triggers.sql - MALEK\SQLEXPRESS.OurHospital (MALEK\malek (52)) - Microsoft SQL Server Management Studio Quick Launch (Ctrl+Q)

File Edit View Query Project Tools Window Help

OurHospital Execute New Query

Object Explorer

Project 2 Creating... (MALEK\malek (52)) Project 2 Creating... (MALEK\malek (71)) Project 2 Creating... (MALEK\malek (87))

```
USE OurHospital;
GO

/*Creates a trigger that prevents hospital staff from making an appointment with a physician that already has an appointment scheduled for that time*/
CREATE TRIGGER trPreventDoubleBooking
    ON my_hospital.Appointments
    INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted JOIN my_hospital.Appointments ON Appointments.PhysicianID = inserted.PhysicianID
        WHERE Appointments.AppointmentDateTime = inserted.AppointmentDateTime
    )
    BEGIN
        ROLLBACK TRANSACTION;
        THROW 500031, 'The physician already has an appointment scheduled for that time. Please choose a different time.', 1;
    END
    ELSE
        INSERT INTO my_hospital.Appointments
        SELECT * FROM inserted;
    END
```

100 %

Messages

Commands completed successfully.

Completion time: 2023-04-30T12:50:02.9331640-04:00

Project 2 Creating Triggers.sql - MALEK\SQLEXPRESS (16.0 RTM) - MALEK\malek (52) OurHospital 00:00:00 0 rows

Project 2 Creating... (MALEK\malek (52)) Project 2 Creating... (MALEK\malek (71)) Project 2 Creating... (MALEK\malek (87))

```
USE OurHospital;
GO

/*creates a trigger that prevents a new patient from being admitted into a room that is already occupied by someone else*/
CREATE TRIGGER trPreventDoubleRoomOccupancy
    ON my_hospital.Admissions
    INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted JOIN my_hospital.Admissions ON Admissions.RoomID = inserted.RoomID
        WHERE Admissions.DischargeDateTime IS NULL
    )
    BEGIN
        ROLLBACK TRANSACTION;
        THROW 500032, 'Another patient is already admitted in this room. Please choose a different room.', 1;
    END
    ELSE
        INSERT INTO my_hospital.Admissions
        SELECT * FROM inserted;
    END
```

100 %

Messages

Commands completed successfully.

Completion time: 2023-04-30T12:57:22.4270987-04:00

Project 2 Creating Triggers.sql - MALEK\SQLEXPRESS (16.0 RTM) - MALEK\malek (52) OurHospital 00:00:00 0 rows

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database schema for 'OurHospital'. The main pane shows a T-SQL script for creating a trigger:

```

USE OurHospital;
GO

/*Creates a trigger to prevent prescribing or giving a patient a medication that they are allergic to*/
CREATE TRIGGER trPreventMedicationAllergy
ON my_hospital.PatientMedications
INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted JOIN my_hospital.Allergies ON inserted.PatientID = Allergies.PatientID
        JOIN my_hospital.Medications ON inserted.MedicationID = Medications.MedicationID
        WHERE Allergies.AllergyName = Medications.MedicationName
    )
    BEGIN
        ROLLBACK TRANSACTION;
        THROW 520033, 'This patient is allergic to this medication and cannot receive it', 1;
    END
    ELSE
        INSERT INTO my_hospital.PatientMedications (PatientMedID, Dosage, Frequency, PatientID, PhysicianID, MedicationID)
        SELECT PatientMedID, Dosage, Frequency, PatientID, PhysicianID, MedicationID
        FROM inserted;
END

```

The status bar at the bottom indicates: MALEK\SQLEXPRESS (16.0 RTM) MALEK\malek (52) OurHospital 00:00:00 0 rows.

Creating All User-Defined Functions:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database schema for 'OurHospital'. The main pane shows a T-SQL script for creating several user-defined functions:

```

USE OurHospital;
GO

/*Creates a function that will subtract all of the payments made towards a patient's bill giving them the final remaining balance*/
CREATE FUNCTION my_hospital.fnRemainingBillBalance
    (@BillID smallint,
     @PatientID smallint)
RETURNS smallint
BEGIN
    RETURN (SELECT (TotalAmount - SUM(PaymentAmount)) AS RemainingBalance
            FROM my_hospital.HospitalBills JOIN my_hospital.Payments
            ON HospitalBills.BillID = Payments.BillID
            WHERE PatientID = @PatientID AND HospitalBills.BillID = @BillID
            GROUP BY TotalAmount);
END;
GO

/*Creates a function that gets all of a particular patient's admission and discharge history*/
CREATE FUNCTION my_hospital.fnGetPatientAdmissionHistory
    (@PatientID smallint)
RETURNS TABLE
RETURN (SELECT Patients.PatientID, AdmissionID, AdmissionDateTime, DischargeDateTime
        FROM my_hospital.Patients JOIN my_hospital.Admissions
        ON Patients.PatientID = Admissions.PatientID
        WHERE Patients.PatientID = @PatientID);
GO

/*Creates a function that gets all of the appointments a physician has scheduled*/
CREATE FUNCTION my_hospital.fnGetAllPhysicianAppointments
    (@PhysicianID smallint)
RETURNS TABLE
RETURN (SELECT Physicians.PhysicianID, AppointmentID, AppointmentDateTime
        FROM my_hospital.Physicians JOIN my_hospital.Appointments
        ON Physicians.PhysicianID = Appointments.PhysicianID
        WHERE Physicians.PhysicianID = @PhysicianID);
GO

/*Creates a function that gets all of a patients lab results/lab history*/
CREATE FUNCTION my_hospital.fnGetPatientLabHistory
    (@PatientID smallint)
RETURNS TABLE
RETURN (SELECT Patients.PatientID, TestID, TestName, TestResult
        FROM my_hospital.Patients JOIN my_hospital.LabResults
        ON Patients.PatientID = LabResults.PatientID
        WHERE Patients.PatientID = @PatientID);
GO

```

Creating All Views:

The screenshot shows the Object Explorer on the left and a query editor window on the right. The query editor contains several CREATE VIEW statements for different views:

- `CREATE VIEW my_hospital.FullPatientSummary AS`
- `SELECT my_hospital.Patients.PatientID, FirstName, LastName, DateOfBirth, Sex, Race, SurgeryName, my_hospital.HospitalAdmissions.AdmissionID, my_hospital.HospitalAdmissions.DischargeDate, AdmisionDate, DischargeDate, Diagnosis, DiagnosisDescription, DiagnosisSeverity, my_hospital.PatientMedications.MedicationID, Dosage, Frequency`
- `FROM my_hospital.Patients`
- `LEFT JOIN my_hospital.SurgeryHistory ON my_hospital.Patients.PatientID = my_hospital.SurgeryHistory.PatientID`
- `LEFT JOIN my_hospital.PatientMedications ON my_hospital.Patients.PatientID = my_hospital.PatientMedications.PatientID`
- `LEFT JOIN my_hospital.CancerHistory ON my_hospital.Patients.PatientID = my_hospital.CancerHistory.PatientID`
- `LEFT JOIN my_hospital.SmokingHistory ON my_hospital.Patients.PatientID = my_hospital.SmokingHistory.PatientID`
- `LEFT JOIN my_hospital.PatientDiagnosis ON my_hospital.Patients.PatientID = my_hospital.PatientDiagnosis.PatientID`
- `LEFT JOIN my_hospital.Admissions ON my_hospital.Patients.PatientID = my_hospital.Admissions.PatientID`

Other view definitions include:

- `CREATE VIEW my_hospital.PatientBills AS`
- `SELECT Patients.PatientID, FirstName, LastName, my_hospital.HospitalBills.BillID, BillDate, TotalAmount, ChargeAmount, ChargeDescription`
- `FROM my_hospital.Patients`
- `LEFT JOIN my_hospital.HospitalBills ON Patients.PatientID = HospitalBills.PatientID`
- `LEFT JOIN my_hospital.HospitalBills.BillItems ON HospitalBills.BillID = BillItems.BillID`

And more:

- `CREATE VIEW my_hospital.LabResultsAndDiagnosis AS`
- `SELECT Patients.PatientID, FirstName, LastName, TestResult, DiagnosisName, DiagnosisDescription, DiagnosisDate`
- `FROM my_hospital.Patients`
- `LEFT JOIN my_hospital.LabResults ON Patients.PatientID = LabResults.PatientID`
- `LEFT JOIN my_hospital.PatientDiagnosis ON Patients.PatientID = PatientDiagnosis.PatientID`

Final view definitions:

- `CREATE VIEW my_hospital.PatientProcedures AS`
- `SELECT Patients.PatientID, FirstName, LastName, ProcedureName, ProcedureDateTime, PhysicianID`
- `FROM my_hospital.Patients`
- `LEFT JOIN my_hospital.HospitalProcedures ON Patients.PatientID = HospitalProcedures.PatientID`

Completion message:

```
Commands completed successfully.
Completion time: 2023-04-27T20:55:05.9958029+04:00
```

Creating Different Users:

The screenshot shows the Object Explorer on the left and a query editor window on the right. The query editor contains several CREATE LOGIN and CREATE USER statements for different users:

- `CREATE LOGIN DrGoodwin WITH PASSWORD = 'DrGoodwin$19';`
- `CREATE USER DrGoodwin FOR LOGIN DrGoodwin;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.PatientMedications TO DrGoodwin;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.PatientDiagnosis TO DrGoodwin;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.Admissions TO DrGoodwin;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.SurgeryHistory TO DrGoodwin;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.SmokingHistory TO DrGoodwin;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.CancerHistory TO DrGoodwin;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.Allergies TO DrGoodwin;`

For the `NurseWelly` user:

- `CREATE LOGIN NurseWelly WITH PASSWORD = 'Nelly$11';`
- `CREATE USER NurseWelly FOR LOGIN NurseWelly;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.Patients TO NurseWelly;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.PatientMedications TO NurseWelly;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.PatientDiagnosis TO NurseWelly;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.Admissions TO NurseWelly;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.SurgeryHistory TO NurseWelly;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.SmokingHistory TO NurseWelly;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.CancerHistory TO NurseWelly;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.Allergies TO NurseWelly;`

For the `SchedulerSam` user:

- `CREATE LOGIN SchedulerSam WITH PASSWORD = 'Samiam$13';`
- `CREATE USER SchedulerSam FOR LOGIN SchedulerSam;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.PatientAppointments TO SchedulerSam;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.Appointments TO SchedulerSam;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.Patients TO SchedulerSam;`

For the `NurseManagerMike` user:

- `CREATE LOGIN NurseManagerMike WITH PASSWORD = 'ManagingisTheGame$12';`
- `CREATE USER NurseManagerMike FOR LOGIN NurseManagerMike;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.NurseSchedule TO NurseManagerMike;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.NurseShiftAssignments TO NurseManagerMike;`
- `GRANT SELECT, INSERT, UPDATE ON OBJECT: my_hospital.Nurses TO NurseManagerMike;`

Completion messages:

```
Commands completed successfully.
Completion time: 2023-04-30T14:34:19.1981194+04:00
```

Populating All Tables:

```
-- Inserting 5 rows of data into the patients table, essentially creating 5 patients
INSERT INTO OurHospital_my_hospital.Patients (PatientID, FirstName, LastName, Email, PhoneNumber, DateOfBirth, Sex, Race)
VALUES
(1, 'Mandy', 'McMiller', 'mcwilliams@gmail.com', NULL, '1990-01-01', 'F', 'White'),
(2, 'Malek', 'Hassan', 'mohamedhassan123@gmail.com', NULL, '1990-02-01', 'M', 'White'),
(3, 'Sarah', 'Johnson', 'sarahjohnson123@gmail.com', NULL, '1990-03-01', 'F', 'White'),
(4, 'Josh', 'Johnson', 'johnsonj@yahoo.com', NULL, '1988-09-17', 'M', 'White'),
(5, 'Savannah', 'Lee', 'simeonleeh@mail.com', NULL, '2001-08-05', 'F', 'Asian');

-- Inserting 5 rows of data into the SurgeryHistory table
INSERT INTO OurHospital_my_hospital.SurgeryHistory (PatientID, SurgeryID, SurgeryName, SurgeryDate, PatientID)
VALUES
(1, 'Appendectomy', '1999-01-05', 1),
(2, 'Cataract Surgery', '2012-02-10', 1),
(3, 'Meniscus Repair', '2006-09-18', 4),
(4, 'Hernia Repair', '2010-06-11', 5),
(5, 'Nephrectomy', '2018-08-01', 4);

-- Inserting 5 rows of data into the CancerHistory table
INSERT INTO OurHospital_my_hospital.CancerHistory (PatientID, CancerType, CancerDiagnosisDate, CancerStatus)
VALUES
(1, 'Breast Cancer', '2010-02-15', 'Remission'),
(2, 'Uterine Cancer', '2012-04-20', 'Remission'),
(3, 'Lung Cancer', '2014-06-25', 'Active'),
(4, 'Lung Cancer', '2007-08-20', 'Active'),
(5, 'Liver Cancer', '2009-07-14', 'Active');

-- Inserting 5 rows of data into the Allergies table
INSERT INTO OurHospital_my_hospital.Allergies (AllergyID, AllergyName, AllergyReaction, AllergySeverity, PatientID)
VALUES
(1, 'Anesthesia', 'Always close', 'Severe', 3),
(2, 'Penicillin', 'Skin rash', 'Mild', 1),
(3, 'Dairy', 'Upset stomach', 'Severe', 2),
(4, 'Aspirin', 'Headache', 'Mild', 5),
(5, 'Sulfonamide', 'Nausea', 'Mild', 4);

-- Inserting 5 rows of data into the SmokingHistory table
INSERT INTO OurHospital_my_hospital.SmokingHistory (SmokingID, PastSmoker, CurrentSmoker, NumberOfYearsSmoking, PatientID)
VALUES
(1, 'Yes', 'No', 1),
(2, 'Yes', 'No', 5, 2),
(3, 'No', 'No', 0, 3),
(4, 'Yes', 'Yes', 4),
(5, 'No', 'Yes', 0, 5);

-- Inserting 5 rows into the PatientDiagnosis table
INSERT INTO OurHospital_my_hospital.PatientDiagnosis (DiagnosisID, DiagnosisName, DiagnosisDescription, DiagnosisDate, PatientID)
VALUES
(1, 'Jilt', 'Patient has flu symptoms', '2023-01-10', 1),
(2, 'Broken Arm', 'Broke arm from fall while playing basketball', '2023-04-20', 2),
(3, 'Migraine', 'Patient has a migraine', '2023-04-22', 3),
(4, 'Pneumonia', 'Symptoms include cough, poor breathing', '2023-04-25', 4),
(5, 'COVID-19', 'Tested positive for COVID-19', '2023-03-20', 5);

-- Inserting 5 rows into the PatientInsuranceInformation table
INSERT INTO OurHospital_my_hospital.PatientInsurance (InsuranceID, GroupNumber, SubscriberID, Issuer, SubscriberFirstName, SubscriberLastName, SubscriberDOB, PatientID)
VALUES
(1, 1234, 'ABC123', 1000, 'Mandy', 'McMiller', '1990-01-01', 1),
(2, 1235, 'XYZ456', 2000, 'Malek', 'Hassan', '1990-02-01', 2),
(3, 9876, 'GHI789', 3000, 'Sarah', 'Johnson', '1990-03-01', 3),
(4, 3456, 'JKL123', 4000, 'Josh', 'Johnson', '1988-09-17', 4),
(5, 7890, 'PQR345', 5000, 'Savannah', 'Lee', '2001-08-05', 5);

-- Messages
(0 rows affected)
(0 rows affected)
Completion time: 2023-04-30T15:34:09.3130495+04:00
100 % -
```

```
-- Inserting 5 rows into the HospitalBills table
INSERT INTO OurHospital_my_hospital.HospitalBills (BillID, BillDate, TotalAmount, PatientID)
VALUES
(1, '2023-04-16', 9000, 1),
(2, '2023-04-21', 2000, 2),
(3, '2023-04-23', 500, 3),
(4, '2023-04-23', 1000, 4),
(5, '2023-04-21', 1000, 5);

-- Inserting 5 rows into the Payments table
INSERT INTO OurHospital_my_hospital.Payments (PaymentID, PaymentDate, PaymentAmount, BillID)
VALUES
(1, '2023-04-17', 3000, 1),
(2, '2023-04-23', 500, 2),
(3, '2023-04-26', 500, 2),
(4, '2023-04-29', 1000, 4),
(5, '2023-04-24', 500, 3);

-- Inserting 5 rows into the BillItems table
INSERT INTO OurHospital_my_hospital.BillItems (ChargeID, ChargeDescription, ChargeAmount, BillID)
VALUES
(1, 'Lab Test', 250, 1),
(2, 'X-ray', 1250, 1),
(3, 'MRI', 1500, 1),
(4, 'Consultation', 1000, 1),
(5, 'Surgery', 5000, 1);

-- Inserting 5 rows into the Medications table
INSERT INTO OurHospital_my_hospital.Medications (MedicationID, MedicationName)
VALUES
(1, 'Ibuprofen'),
(2, 'Acetaminophen'),
(3, 'Lisinopril'),
(4, 'Metformin'),
(5, 'Levothyroxine');

-- Inserting 5 rows into the Physicians table
INSERT INTO OurHospital_my_hospital.Physicians (PhysicianID, FirstName, LastName, Specialty, LicensceNumber, PhoneNumber, HospitalEmail, HireDate)
VALUES
(1, 'John', 'Johnson', 'Cardiology', 123456789, NULL, 'johnson@ourhospital.com', '2020-01-01'),
(2, 'Jane', 'Smith', 'Emergency', 987654321, NULL, 'jane@ourhospital.com', '2019-05-01'),
(3, 'Michael', 'Brown', 'Neurology', 543210987, NULL, 'michael@ourhospital.com', '2018-07-15'),
(4, 'Elizabeth', 'Brown', 'Neurology', 789654321, NULL, 'elizabeth@ourhospital.com', '2021-02-01'),
(5, 'David', 'Lee', 'Surgery', 234567890, NULL, 'david@ourhospital.com', '2017-11-01');

-- Inserting 5 rows into the PatientMedications table
INSERT INTO OurHospital_my_hospital.PatientMedications (PatientMedID, Dosage, Frequency, PatientID, PhysicianID, MedicationID)
VALUES
(1, '600mg', '2x a day', 1, 1, 1),
(2, '300mg', '3x a day', 2, 3, 2),
(3, '200mg', '2x day', 2, 3, 3),
(4, '200mg', '2x a day', 4, 1, 4),
(5, '60mg', '1x a day', 1, 5, 5);

-- Inserting 5 rows into the hospital_procedures table
INSERT INTO OurHospital_my_hospital.HospitalProcedures (ProcedureID, ProcedureName, ProcedureDateTime, PatientID, PhysicianID)
VALUES
(1, 'Surgery', '2023-04-01 08:00:00', 1, 5),
(2, 'CT scan', '2023-04-02 10:30:00', 2, 3),
(3, 'X-ray', '2023-04-03 11:45:00', 1, 4),
(4, 'MRI', '2023-04-03 13:15:00', 1, 2),
(5, 'Ultrasound', '2023-04-05 15:30:00', 5, 1);

-- Messages
(0 rows affected)
(0 rows affected)
Completion time: 2023-04-30T15:34:09.3130495+04:00
100 % -
```

Project 2 Populating Database.sql - MALEK\SQLEXPRESS.OurHospital (MALEK\malek) - Microsoft SQL Server Management Studio

File Edit View Project Tools Window Help

Object Explorer

OurHospital

Project 2 Population... (MALEK\malek (56)) Project 2 Creating... (MALEK\malek (52)) Project 2 Creating... (MALEK\malek (55))

```
--Inserting 5 rows into the procedure notes table
INSERT INTO OurHospital.my_hospital_ProcedureNotes (ProcedureNoteID, ProcedureID, NoteContent, NoteDateTime)
VALUES
(1, 'Patient responded well to the surgery', '2023-04-01 10:00:00'),
(2, '1. CT went well, results pending', '2023-04-01 11:00:00'),
(3, 'X-Ray went well, results pending', '2023-04-01 12:00:00'),
(4, 'MRI went well', '2023-04-01 10:00:00'),
(5, 'Ultrasound went well', '2023-04-01 16:00:00');

--Inserting 5 rows into the physician schedule table
INSERT INTO OurHospital.my_hospital_PhysicianSchedule (PhysicianShiftID, StartShiftDateTime, EndShiftDateTime, NumberOfPhysicians, Department)
VALUES
(1, '2023-05-01 08:00:00', '2023-05-01 16:00:00', 2, 'Neurology'),
(2, '2023-05-01 08:00:00', '2023-05-02 16:00:00', 2, 'Cardiology'),
(3, '2023-05-02 08:00:00', '2023-05-02 16:00:00', 2, 'Cardiology'),
(4, '2023-05-02 16:00:00', '2023-05-03 08:00:00', 2, 'Cardiology'),
(5, '2023-05-01 08:00:00', '2023-05-01 16:00:00', 3, 'Oncology');

--Inserting 5 rows into the PhysicianShiftAssignments
INSERT INTO OurHospital.my_hospital_PhysicianShiftAssignments (ShiftAssignmentID, PhysicianID, PhysicianShiftID)
VALUES
(1, 1),
(2, 4, 2),
(3, 1, 3),
(4, 1, 4),
(5, 3, 5);

--Inserting 5 rows into the Lab results table
INSERT INTO OurHospital.my_hospital_LabResults (TestID, PatientID, PhysicianID, TestName, TestResult)
VALUES
(1, 1, 5, 'Blood Sugar', 'Normal'),
(2, 1, 2, 'Hemoglobin A1C', 'High'),
(3, 2, 4, 'Lipid Panel', 'Normal'),
(4, 4, 1, 'WBC', 'Low'),
(5, 5, 3, 'TSH', 'Low');

--Inserting 5 rows into the hospital rooms table
INSERT INTO OurHospital.my_hospital_HospitalRooms (RoomID, RoomType, RoomNumber, RoomFloor)
VALUES
(1, 'ER Room', 101, 1),
(2, 'ER Room', 102, 1),
(3, 'ER Room', 103, 1),
(4, 'Urgent', 201, 2),
(5, 'Urgent', 202, 2),
(6, 'Urgent', 203, 2),
(7, 'Urgent', 204, 2),
(8, 'Urgent', 205, 2),
(9, 'Urgent', 206, 2),
(10, 'OK', 401, 4);

--Inserting 5 rows into the Nurses table
INSERT INTO OurHospital.my_hospital_Nurses (NurseID, FirstName, LastName, HospitalEmail, PhoneNumber, Department, JobTitle, HireDate)
VALUES
(1, 'Samuel', 'Smith', 'smith@ourhospital.com', NULL, 'Urgent', 'URN', '2020-01-15'),
(2, 'Mike', 'Sanson', 'msanson@ourhospital.com', NULL, 'Oncology', 'UPN', '2019-05-07'),
(3, 'Emily', 'Manchester', 'emanchester@ourhospital.com', NULL, 'Surgey', 'RN', '2021-02-28'),
(4, 'Declan', 'Brown', 'dbrown@ourhospital.com', NULL, 'Neurology', 'RN', '2018-11-01'),
(5, 'Karen', 'Davis', 'kdavis@ourhospital.com', NULL, 'Cardiology', 'RN', '2012-03-27');

--Insert 5 rows into the Nurse Schedule table
<#NURSE_HR#>
<#NURSE_HR#>
<#NURSE_HR#>
<#NURSE_HR#>
<#NURSE_HR#>
```

File Edit View Project Tools Window Help

Object Explorer

OurHospital

Project 2 Population... (MALEK\malek (56)) Project 2 Creating... (MALEK\malek (52)) Project 2 Creating... (MALEK\malek (55))

Messages

it row affected
it row affected
Completion time: 2023-04-30T18:34:09.3100000+00:00

100 %

Project 2 Populating Database.sql - MALES\SQLEXPRESS\OurHospital (MALEK\malek (56)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

Connect to Server

OurHospital

Project 2 Populating Database.sql (MALEK\malek (52)) Project 2 Creating... (MALEK\malek (85))

```
--Insert 5 rows into the Nurse Schedule table
INSERT INTO OurHospital.my_hospital.NurseSchedule (NurseShiftID, StartShiftDateTime, EndShiftDateTime, NumberOfNurses, Department)
VALUES
(1, '2023-05-01 08:00:00', '2023-05-01 16:00:00', 2, 'Urgent'),
(2, '2023-05-01 16:00:00', '2023-05-02 00:00:00', 2, 'Oncology'),
(3, '2023-05-02 08:00:00', '2023-05-02 16:00:00', 2, 'Surgery'),
(4, '2023-05-02 16:00:00', '2023-05-03 00:00:00', 2, 'Neurology'),
(5, '2023-05-01 08:00:00', '2023-05-01 16:00:00', 3, 'Cardiology');

--insert 5 rows into the Nurse shift assignments
INSERT INTO OurHospital.my_hospital.NurseShiftAssignments (ShiftAssignmentID, NurseShiftID, NurseID)
VALUES
(1, 1, 1),
(2, 2, 2),
(3, 3, 3),
(4, 4, 4),
(5, 5, 5);

--inserts 5 rows into the Appointments
INSERT INTO OurHospital.my_hospital.Appointments (AppointmentID, PhysicianID, AppointmentDateTime, AppointmentReason, NurseID, RoomID)
VALUES
(1, 1, '2023-05-10 10:00:00', 'Annual Checkup', 1, 4),
(2, 2, '2023-05-14 14:30:00', 'Sore Throat', 2, 5),
(3, 1, '2023-05-18 09:15:00', 'High Blood Pressure', 3, 6),
(4, 5, '2023-05-22 11:45:00', 'Abdominal Pain', 4, 7),
(5, 2, '2023-05-29 16:00:00', 'Flu Symptoms', 5, 8);

--inserts 5 rows into the Patient Appointments table
INSERT INTO OurHospital.my_hospital.PatientAppointments (PatientID, AppointmentID)
VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5);

--inserts 5 rows into the Admissions table
INSERT INTO OurHospital.my_hospital.Admissions (AdmissionID, RoomID, AdmissionDateTime, DischargeDateTime, PhysicianID, PatientID)
VALUES
(1, 4, '2023-04-15 10:00:00', '2023-04-15 11:00:00', 2, 1),
(2, 2, '2023-04-20 14:00:00', '2023-04-20 18:00:00', 5, 2),
(3, 6, '2023-04-22 10:00:00', '2023-04-22 10:00:00', 4, 3),
(4, 7, '2023-04-25 09:00:00', '2023-04-25 11:00:00', 1, 4),
(5, 8, '2022-03-20 14:00:00', '2022-03-20 15:00:00', 2, 5);
```

100 %

Messages

(5 rows affected)

(5 rows affected)

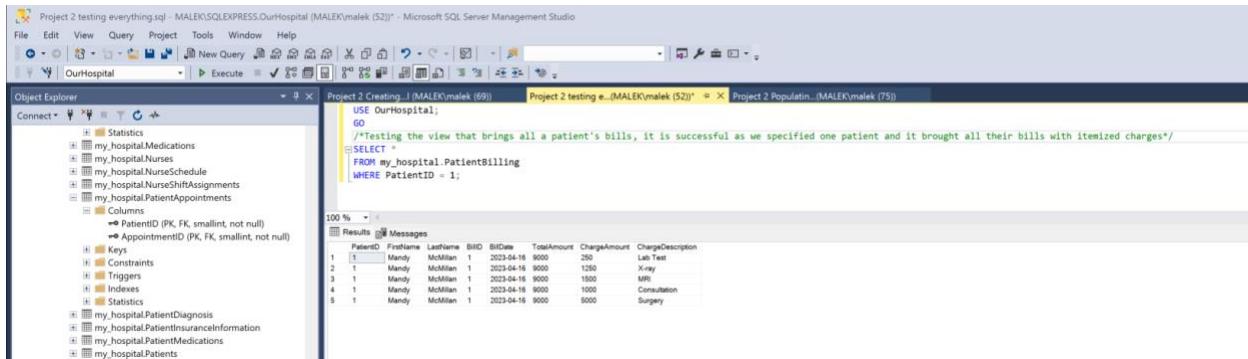
Completion time: 2023-04-30 07:19:34:09.3130435-04:00

100 %

Query executed successfully.

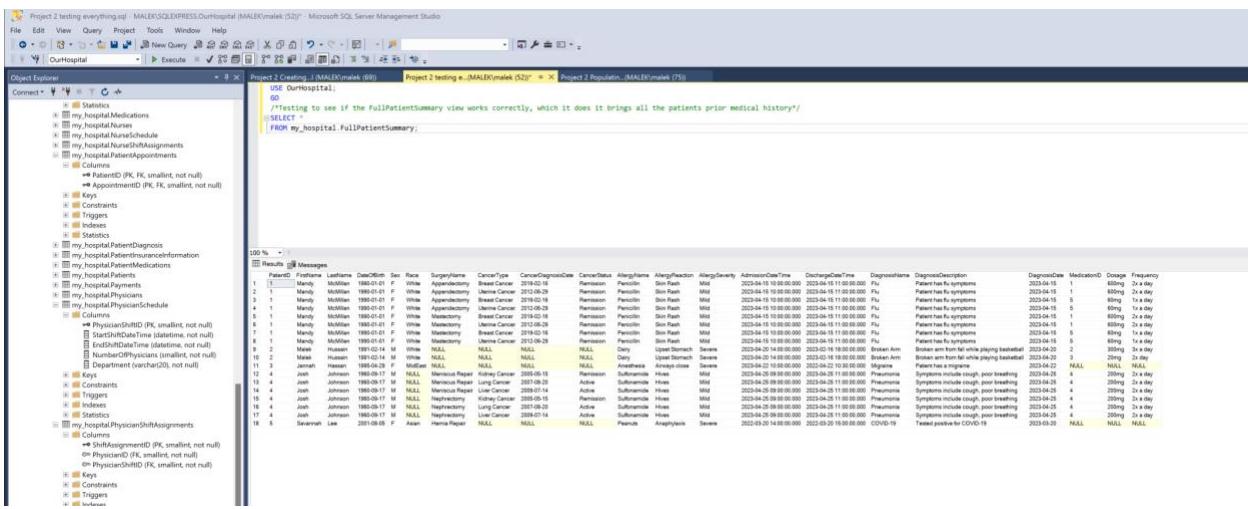
Appendix B: Testing Screenshots

Testing All Views:



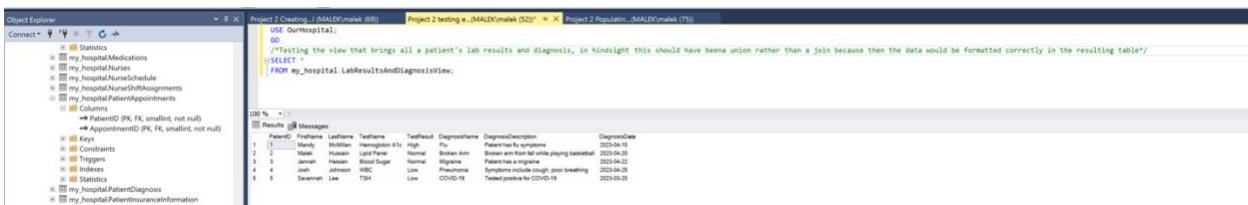
```
USE OurHospital;
GO
/*Testing the view that brings all a patient's bills, it is successful as we specified one patient and it brought all their bills with itemized charges*/
SELECT *
FROM my_hospital.PatientBilling
WHERE PatientID = 1;
```

PatientID	Fname	Lname	BID	BillNo	BillAmount	ChargeAmount	ChargeDescription
1	Mandy	McMillan	1	2020-04-16	8000	260	Lab Test
2	Mandy	McMillan	1	2020-04-16	9000	1250	X-ray
3	Mandy	McMillan	1	2020-04-16	9000	1500	MRI
4	Mandy	McMillan	1	2020-04-16	9000	1000	Consultation
5	Mandy	McMillan	1	2020-04-16	9000	5000	Surgery



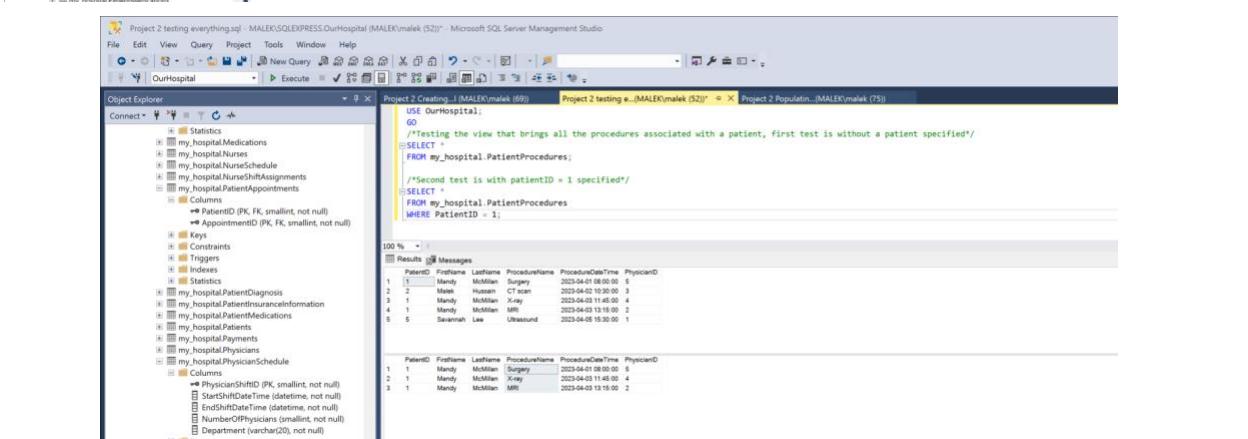
```
USE OurHospital;
GO
/*Testing to see if the FullPatientSummary view works correctly, which it does it brings all the patients prior medical history*/
SELECT *
FROM my_hospital.FullPatientSummary;
```

PatientID	Fname	Lname	DOB	Gender	SurgeryType	SurgeonName	ProcedureDate	Hospital	HospitalRoom	HospitalFloor	AdmissionDate	DischargeDate	DischargeTime	DischargeReason	DischargeDoc	MedicationID	Drugs	Frequency
1	Mandy	McMillan	1980-01-01	F	White Appendectomy	Breast Cancer	2019-02-10	Remington	Pencil	Skin Rash	Mid	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
2	Mandy	McMillan	1980-01-01	F	White Appendectomy	Uterine Cancer	2012-06-10	Remington	Pencil	Skin Rash	Mid	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
3	Mandy	McMillan	1980-01-01	F	White Appendectomy	Uterine Cancer	2012-06-10	Remington	Pencil	Skin Rash	Mid	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
4	Mandy	McMillan	1980-01-01	F	White Appendectomy	Uterine Cancer	2012-06-10	Remington	Pencil	Skin Rash	Mid	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
5	Mandy	McMillan	1980-01-01	F	White Appendectomy	Uterine Cancer	2012-06-10	Remington	Pencil	Skin Rash	Mid	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
6	Mandy	McMillan	1980-01-01	F	White Appendectomy	Uterine Cancer	2012-06-10	Remington	Pencil	Skin Rash	Mid	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
7	Mandy	McMillan	1980-01-01	F	White Appendectomy	Uterine Cancer	2012-06-10	Remington	Pencil	Skin Rash	Mid	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
8	Mandy	McMillan	1980-01-01	F	White Appendectomy	Uterine Cancer	2012-06-10	Remington	Pencil	Skin Rash	Mid	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
9	Jennah	Hassen	1981-02-14	M	White Appendectomy	N/A	N/A	N/A	N/A	N/A	N/A	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
10	Jennah	Hassen	1981-02-14	M	White Appendectomy	N/A	N/A	N/A	N/A	N/A	N/A	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
11	Jennah	Hassen	1981-04-29	M	White Appendectomy	N/A	N/A	N/A	N/A	N/A	N/A	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
12	Jess	Johnson	1980-09-17	M	White Appendectomy	N/A	N/A	N/A	N/A	N/A	N/A	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
13	Jess	Johnson	1980-09-17	M	White Appendectomy	N/A	N/A	N/A	N/A	N/A	N/A	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
14	Jess	Johnson	1980-09-17	M	White Appendectomy	N/A	N/A	N/A	N/A	N/A	N/A	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
15	Jess	Johnson	1980-09-17	M	White Appendectomy	N/A	N/A	N/A	N/A	N/A	N/A	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
16	Jess	Johnson	1980-09-17	M	White Appendectomy	N/A	N/A	N/A	N/A	N/A	N/A	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
17	Jess	Johnson	1980-09-17	M	White Appendectomy	N/A	N/A	N/A	N/A	N/A	N/A	2020-04-16 00:00:00	2020-04-16 00:00:00	Fu				
18	Savannah	Lee	2001-08-05	F	Asian Appendectomy	N/A	N/A	N/A	N/A	N/A	N/A	2020-04-16 00:00:00	2020-04-16 00:00:00	Covid-19				



```
USE OurHospital;
GO
/*Testing the view that brings all a patient's lab results and diagnosis, in hindsight this should have beena union rather than a join because then the data would be formatted correctly in the resulting table*/
SELECT *
FROM my_hospital.LabResultsAndDiagnosisView;
```

PatientID	Fname	Lname	TestName	TestResult	DiagnosisName	DiagnosisDescription	DiagnosisDate
1	Mandy	McMillan	Hematology A/B	High	Fu	Patient has symptoms	2020-04-18
2	Mandy	McMillan	Low Mammogram	Normal	Flu	Patient has symptoms	2020-04-18
3	Jennah	Hassen	Blood Type	Normal	Migraine	Patient has a migraine	2020-04-22
4	Jess	Johnson	WBC	Low	Pneumonia	Symptoms include cough, poor breathing	2020-04-28
5	Savannah	Lee	TSH	Normal	Pneumonia	Symptoms include cough, poor breathing	2020-04-28



```
USE OurHospital;
GO
/*Testing the view that brings all the procedures associated with a patient, first test is without a patient specified*/
SELECT *
FROM my_hospital.PatientProcedures;

/*Second test is with PatientID = 1 specified*/
SELECT *
FROM my_hospital.PatientProcedures
WHERE PatientID = 1;
```

PatientID	Fname	Lname	ProcedureName	ProcedureDate	PhysicianID
1	Mandy	McMillan	Surgery	2020-04-01 08:00:00	\$
2	Mandy	McMillan	CT scan	2020-04-02 10:30:00	3
3	Mandy	McMillan	X-ray	2020-04-02 11:45:00	4
4	Mandy	McMillan	Ultrasound	2020-04-03 08:00:00	2
5	Savannah	Lee	Ultrasound	2020-04-05 15:30:00	1

Testing All Stored Procedures:

Screenshot 1: Testing spAddDiagnosis

```

USE OurHospital;
GO
/*Testing the stored procedure by trying to add a new patient diagnosis for an invalid patient id*/
EXEC my_hospital.spAddDiagnosis
@PatientID = 6,
@DiagnosisName = 'Anemia',
@DiagnosisDescription = 'Low iron from labs',
@DiagnosisDate = '2023-04-30'

```

Screenshot 2: Testing spUpdatePatientPersonalInfo

```

USE OurHospital;
GO
/*Testing the stored procedure by trying to add update the information of a patient that doesn't exist*/
EXEC my_hospital.spUpdatePatientPersonalInfo
@PatientID = 10,
@FirstName = 'John',
@LastName = 'Doe',
@Email = NULL,
@PhoneNumber = NULL,
@DateOfBirth = '1999-11-30',
@Sex = 'M',
@Race = 'White'

```

Screenshot 3: Testing spAddNewPatientAppointment

```

USE OurHospital;
GO
/*Testing the stored procedure by trying to add an appointment with a physician that already has an appointent at that time*/
EXEC my_hospital.AddNewPatientAppointment
@PatientID = 2,
@AppointmentDateTime = '2023-05-10 10:00:00',
@AppointmentReason = 'Sick',
@NurseID = 1,
@RoomID = 4,
@PhysicianID = 1,
@AppointmentID = 6

```

Screenshot 4: Testing spAddNewPatient

```

USE OurHospital;
GO
/*Testing the stored procedure by trying to add a new patient that has a date of birth that is in the future*/
EXEC my_hospital.spAddNewPatient
@PatientID = 6,
@FirstName = 'Johnny',
@LastName = 'Tempton',
@email = 'jrocket@hotmail.com',
@PhoneNumber = NULL,
@DateOfBirth = '2023-06-20',
@Sex = 'M',
@Race = 'White'

```

Testing All User-Defined Functions:

Project 2 testing everything.sql - MALEK\SQLEXPRESS.OurHospital (MALEK\malek (52)) - Microsoft SQL Server Management Studio

```

File Edit View Query Project Tools Window Help
Connect Object Explorer Project 2 Creating... (MALEK\malek (53)) Project 2 testing e... (MALEK\malek (52)) X Project 2 Populatin... (MALEK\malek (75))
New Query Execute
Object Explorer
Connect
  ↪ AppointmentID (PK, FK, smallint, not null)
    Keys
    Constraints
    Triggers
    Indexes
    Statistics
  my_hospital.PatientDiagnosis
  my_hospital.PatientInsuranceInformation
  my_hospital.PatientMedications
  my_hospital.Patients
  ↪ Messages
  Balance due for Mandy McMillan is $6000
Completion time: 2023-04-30T19:38:54.9226907-04:00
  
```

Project 2 testing everything.sql - MALEK\SQLEXPRESS.OurHospital (MALEK\malek (52)) - Microsoft SQL Server Management Studio

```

File Edit View Query Project Tools Window Help
Connect Object Explorer Project 2 Creating... (MALEK\malek (53)) Project 2 testing e... (MALEK\malek (52)) X Project 2 Populatin... (MALEK\malek (75))
New Query Execute
Object Explorer
Connect
  ↪ AppointmentID (PK, FK, smallint, not null)
    Keys
    Constraints
    Triggers
    Indexes
    Statistics
  my_hospital.PatientDiagnosis
  my_hospital.PatientInsuranceInformation
  my_hospital.PatientMedications
  my_hospital.Patients
  ↪ Results Messages
  SELECT *
  FROM my_hospital.fnGetPatientAdmissionHistory(1);
  
```

PatentID	AdmissionID	AdmissionDateTime	DischargeDateTime
1	1	2023-04-15 10:00:00.000	2023-04-15 11:00:00.000

Project 2 testing everything.sql - MALEK\SQLEXPRESS.OurHospital (MALEK\malek (52)) - Microsoft SQL Server Management Studio

```

File Edit View Query Project Tools Window Help
Connect Object Explorer Project 2 Creating... (MALEK\malek (53)) Project 2 testing e... (MALEK\malek (52)) X Project 2 Populatin... (MALEK\malek (75))
New Query Execute
Object Explorer
Connect
  ↪ AppointmentID (PK, FK, smallint, not null)
    Keys
    Constraints
    Triggers
    Indexes
    Statistics
  my_hospital.PatientDiagnosis
  my_hospital.PatientInsuranceInformation
  my_hospital.PatientMedications
  my_hospital.Patients
  ↪ Results Messages
  SELECT *
  FROM my_hospital.fnGetAllPhysicianAppointments(2);
  
```

PhysicianID	AppointmentID	AppointmentDateTime
1	2	2023-05-15 14:30:00.000
2	5	2023-05-29 16:00:00.000

Project 2 testing everything.sql - MALEK\SQLEXPRESS.OurHospital (MALEK\malek (52)) - Microsoft SQL Server Management Studio

```

File Edit View Query Project Tools Window Help
Connect Object Explorer Project 2 Creating... (MALEK\malek (53)) Project 2 testing e... (MALEK\malek (52)) X Project 2 Populatin... (MALEK\malek (75))
New Query Execute
Object Explorer
Connect
  ↪ AppointmentID (PK, FK, smallint, not null)
    Keys
    Constraints
    Triggers
    Indexes
    Statistics
  my_hospital.PatientDiagnosis
  my_hospital.PatientInsuranceInformation
  my_hospital.PatientMedications
  my_hospital.Patients
  ↪ Results Messages
  SELECT *
  FROM fnGetPatientLabHistory(5);
  
```

PatientID	TestID	TestName	TestResult
5	5	TSH	Low

Testing All Triggers:

Project 2 testing everything.adl - MALEK\SQLEXPRESS.OurHospital (MALEK\malek (52)) - Microsoft SQL Server Management Studio

```

File Edit View Query Project Tools Window Help
New Query Execute
Object Explorer Connect Project 2 testing e...(MALEK\malek (53)) Project 2 Population...(MALEK\malek (75))
Connect
Triggers
Indexes
Statistics
Tables
my_Hospital_CancerHistory
my_hospital_HospitalBills
my_hospital_HospitalProcedures
Columns
ProcedureID (PK, smallint, not null)
ProcedureName (varchar(15), not null)
ProcedureDateTime (smalldatetime, not null)
PatientID (FK, smallint, not null)
PhysicianID (FK, smallint, not null)

100 %
Messages
Key 52001, Level 16, State 1, Procedure trPreventsSurgeryWithAllergy, Line 14 (Batch Start Line 1)
This patient has a severe allergy to anesthesia and cannot have surgery scheduled
Completion time: 2023-04-30T19:49:16.1255759-04:00

```

Project 2 testing everything.adl - MALEK\SQLEXPRESS.OurHospital (MALEK\malek (52)) - Microsoft SQL Server Management Studio

```

File Edit View Query Project Tools Window Help
New Query Execute
Object Explorer Connect Project 2 testing e...(MALEK\malek (53)) Project 2 Population...(MALEK\malek (75))
Connect
MySchool
OurHospital
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
my_hospital_Admissions
my_hospital_Allergies
my_hospital_Appointments
Columns
AppointmentID (PK, smallint, not null)
PhysicianID (FK, smallint, not null)

100 %
Messages
Key 52001, Level 16, State 1, Procedure trPreventsDoubleBooking, Line 14 (Batch Start Line 1)
The physician already has an appointment scheduled for that time. Please choose a different time.
Completion time: 2023-04-30T19:54:24.9904240-04:00

```

Project 2 testing everything.adl - MALEK\SQLEXPRESS.OurHospital (MALEK\malek (52)) - Microsoft SQL Server Management Studio

```

File Edit View Query Project Tools Window Help
New Query Execute
Object Explorer Connect Project 2 testing e...(MALEK\malek (53)) Project 2 Population...(MALEK\malek (75))
Connect
MALEK\SQLEXPRESS (SQL Server 16.0.1000 - Malek\malek)
Databases
System Databases
Database Snapshots
AP
Examples
MyCollege
MySchool
OurHospital
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
my_hospital_Admissions
my_hospital_Appointments
Columns
AdmissionID (PK, smallint, not null)
RoomID (FK, smallint, not null)

100 %
Messages
(1 row affected)
(1 row affected)
Completion time: 2023-04-30T20:09:34.5749066-04:00

```

Project 2 testing everything.adl - MALEK\SQLEXPRESS.OurHospital (MALEK\malek (52)) - Microsoft SQL Server Management Studio

```

File Edit View Query Project Tools Window Help
New Query Execute
Object Explorer Connect Project 2 testing e...(MALEK\malek (53)) Project 2 Population...(MALEK\malek (75))
Connect
my_hospital_Medications
my_hospital_Nurses
my_hospital_NurseSchedule
my_hospital_NurseShiftAssignments
my_hospital_PatientAppointments
Columns
PatientID (PK, FK, smallint, not null)
AppointmentID (PK, FK, smallint, not null)
Keys
Constraints
Triggers
Indexes
Rooms
my_hospital_PatientDiagnosis
my_hospital_PatientInsuranceInformation
my_hospital_PatientMedications
Columns

100 %
Messages
(1 row affected)
(1 row affected)
Completion time: 2023-04-30T20:14:39.3277923-04:00

```