University Medical Center Database Design

By Nicholas Landry
Syracuse University
nglandry@syr.edu

Abstract

In this report, we will be covering the full design and implementation of a database intended to be used by healthcare facilities. The database was created through Microsoft SQL Server Management Studio and includes 20 descriptive tables. The database also includes specific views, user functions, triggers, transactions, and procedures that are relevant for a healthcare environment. There are also specific users that have varying levels of permissions within the database. There are several test cases that provide insight into the functionality of the database, including screenshots.

Design

In our current world, many places have a need for an efficient data management system for hospitals and other health care environments due to the everlasting presence of COVID-19. With that in mind, creating a database that is able to handle information effectively seems the best way to handle all of the patients and information that a medical center would have to deal with.

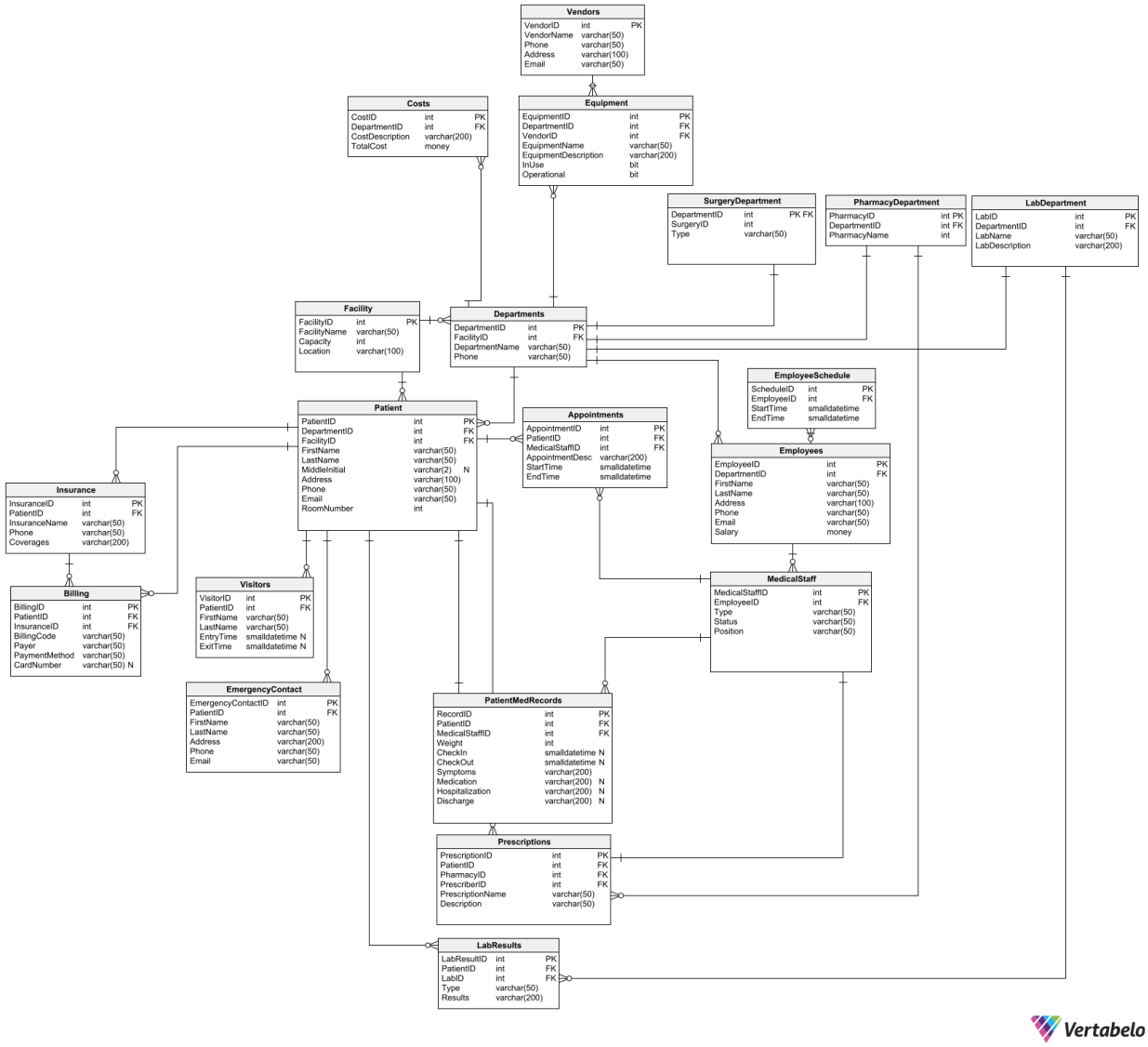

Figure 1. Database Diagram

To start off the design, the table Facility is used as the top most level, where information about facility names and locations are stored. Connected to that table is the Departments and Patient table with a many to one relationship. This is because many departments can exist within a facility and many patients can be in the facility at any given time. It also creates a many to one relationship between Patient and Departments since patients will generally be within certain departments of the facility that fulfill their specific needs, such as the emergency room.

There are 8 other tables that have a relationship with the Patient table. The first two are the Insurance and Billing tables, which also have a many to one relationship, which can be seen on the far left side of *Figure 1.* Insurance stores the patient's insurance coverages along with the name of the insurer and contact information. The Billing table stores payment information for the patients, which can be covered by insurance if it is within the patient's coverages. If insurance won't be the payer, someone else will have to be the payer and insert a credit card number to be stored on record.

From there, there is a Visitors table that will have all the information regarding a patient's visitors stored into it. The table keeps track of the visitors name, entry time and exit time in order to manage who is inside of the facility at any given time. This table has a many to one relationship with Patient, as a single patient can have multiple visitors at any given time. Another table is the EmergencyContact table, which has a many to one relationship with Patients as well. This table stores a patient's emergency contact information in cases where a patient is in an emergency situation. This included the name of the contact, their address, phone number and email address.

At the bottom of *Figure 1,* we have PatientMedRecords, Prescriptions, and LabResults. PatientMedRecords is a table that has a one to one relationship with Patient which stores medical information about the patient, including weight, check in/out times, symptoms, medication, hospitalization, and discharge. The Prescription table has a many to one relationship with Patient since a patient can have multiple prescriptions. This table stores information regarding any prescriptions a patient may have, including the name of the drug, the prescribor, a description of the drug, and the pharmacy it came from. At the very bottom, we have LabResults, which has a many to one relationship with Patient. This table stores the information about any lab tests a patient may have, which includes the type of test the patient received and the results.

The last table that has a relationship to Patient is the Appointments table, which has a one to many relationship since patients can have many appointments. This table stores which patient has an appointment with a specific medical employee, a description of the appointment, and a start/end time of the appointment.

Connected to the Appointments table is the MedicalStaff table, which stores the information regarding any medical staff that are employees of a facility. This table has a relationship to the Employees table, where information regarding any employee within a facility is stored. From there, the Employees table has a one to many relationship with EmployeeSchedule, which has the start and end times of all employee shifts. To bring us full circle, Employees has a one to many relationship with Departments, since many employees work within the departments in a facility.

That leaves us with the remaining tables that resided in the top right of *Figure 1:* Costs, Vendors, Equipment, SurgeryDepartment, PharmacyDepartment, and LabDepartment. The

Costs table stores any costs that a department may have, which includes a cost description and a total for that cost. The Equipment table stores data concerning equipment that a department has available to them and has information regarding the name of equipment, description, if the device is in use, and if the device is operational. Connected to the Equipment table is the Vendors table, which stores the information about the vendors who sold the equipment to the department. The tables SurgeryDepartment, PharmacyDepartment, and LabDepartment store the information about any subcategories that these departments may have. The PharmacyDepartment has a one to many relationship to Prescriptions and LabDepartment has a one to many relationship with LabResults since these tables contain the information about where either prescriptions or lab results came from within the facility.

In regards to views for the database, there are 5 views that will be implemented: WorksToday, PatientInsurance, DepartmentRoomAndAttendingStaff, PatientTestResults, and FacilityVacancy. The WorksToday view will display any employee and their work schedule if they have a shift during the current day. PatientInsurance displays the patient's name along with their insurance information for more seamless coverage lookup. DepartmentRoomAndAttendingStaff concatenates the patient's name, the department name they are in, their room number, and any staff that may be attending them. PatientTestResults displays the patient's name along with the type of test they may have received and the result regarding that test. Finally, FacilityVacancy takes the total number of patients in the Patient table and subtracts that sum from the current capacity of the facility and returns the current vacancy of the facilities within the Facility table.

There will be 2 stored procedures for this database. The first will be AppointmentsAtTime, which will take a smalldatetime as the parameter and check if there are any appointments up to 30 minutes in the future or are currently occurring. If an appointment meets these criteria, it will be returned in a table. The second is PatientsInDepartments, which will take a department name string as the parameter and return the total number of patients within that department.

There will be 2 user functions that can be executed. The first is OverAmount, where the user inputs an amount of money as a parameter. The function will then look into the Costs table and return any costs that are over the user's inputted amount of money. The second function is PatientLookUpByLastName, which takes a last name string as a parameter. This function will look up any patients that match the last name that was given and return the patient's full name, department name, and current room number.

There will be 2 transactions in the database. The first is DeletePastAppointments, which will delete any appointments where the end time has already passed. The second transaction will be DeleteVisitors, which will delete visitors that have left the facility.

There will be 1 trigger called NewPatientInsert, which will trigger upon insert into Patient. The trigger will fill any null values for FacilityID and DepartmentID, and default them with a value of 1.

There will be 2 user roles for the database. The first will be Administrator, which has access to all functions regarding the database. The second will be DOCTOR, which has the ability to SELECT, INSERT, UPDATE, and DELETE in the tables Patient, Appointments, PatientMedRecords, and Prescriptions. The DOCTOR role will also be able to SELECT from LabResults and EmployeeSchedule.

Implementation

The database creation script will be available in the appendix. The creation script starts out by creating the most important tables first.

```
CREATE VIEW WorksToday
AS
    SELECT (FirstName + ' ' + LastName) AS EmployeeName, StartTime, Endtime
    From Employees JOIN EmployeeSchedule ON Employees.EmployeeID = EmployeeSchedule.EmployeeID
    WHERE CAST(StartTime as date) = GETDATE();

CREATE VIEW PatientInsurance
AS
    SELECT (LastName + ', ' + FirstName) AS PatientName, InsuranceName, Coverages
    FROM Patient JOIN Insurance ON Patient.PatientID = Insurance.PatientID;

CREATE VIEW DepartmentRoomAndAttendingStaff
AS
    SELECT (v1.LastName + ', ' + v1.FirstName) AS PatientName, DepartmentName, RoomNumber, (v2.LastName + ', ' + v2.FirstName) AS AttendingStaffName
    FROM Patient v1 JOIN Departments ON v1.DepartmentID = Departments.DepartmentID
    JOIN Appointments ON v1.PatientID = Appointments.PatientID
    JOIN MedicalStaff ON Appointments.MedicalStaffID = MedicalStaff.MedicalStaffID
    JOIN Employees v2 ON MedicalStaff.EmployeeID = v2.EmployeeID;

CREATE VIEW PatientTestResults
AS
    SELECT (LastName + ', ' + FirstName) AS PatientName, Type, Results, LabName
    FROM Patient JOIN LabResults ON Patient.PatientID = LabResults.PatientID
    JOIN LabDepartment ON LabResults.LabID = LabDepartment.LabID;
GO

CREATE VIEW FacilityVacancy
AS
    SELECT FacilityName, (MAX(Capacity)-COUNT(PatientID)) AS CurrentVacancy
    FROM Facility JOIN Patient ON Facility.FacilityID = Patient.FacilityID
    GROUP BY FacilityName;
GO
```

Figure 2. View Creation Script

Figure 2 displays the code used to implement the views for the database. In WorksToday, it selects the employee name, start time and end time. It joins the Employees and EmployeeSchedule inorder to retrieve the relevant information to display who works on a given day. The WHERE clause casts the start time of the shift from smalldatetime to date and checks if the date matches the current date. If it does, it gets displayed in the view.

PatientInsurance selects the patient's name and insurance information by joining Patient and Insurance tables on PatientID.

DepartmentRoomAndAttendingStaff joins Patient, Departments, Appointments, MedicalStaff, and Employees in order to return the relevant columns.

PatientTestResults joins Patient, LabResults, and LabDepartment tables and selects the patient's name, the type of test they got, the results, and which lab it came from.

FacilityVacancy selects the facility name and counts the current amount of patient IDs in the facility, which then gets subtracted from the facility's capacity to calculate the current vacancy. The view joins Facility with Patient in order to get the patient IDs. The view is grouped by FacilityName.

```
CREATE PROCEDURE AppointmentsAtTime @time smalldatetime
AS
SELECT (v1.LastName + ', ' + v1.FirstName) AS PatientName, StartTime, EndTime, AppointmentDesc, (v2.LastName + ', ' + v2.FirstName) AS MedicalStaffName
FROM Patient v1 JOIN Appointments ON v1.PatientID = Appointments.PatientID
JOIN MedicalStaff ON Appointments.MedicalStaffID = MedicalStaff.MedicalStaffID
JOIN Employees v2 ON MedicalStaff.EmployeeID = v2.EmployeeID
WHERE DATEADD(minute, 30, @time) >= StartTime AND @time < EndTime;
GO

CREATE PROCEDURE PatientsInDepartment @depName varchar(50)
AS
BEGIN
SELECT DepartmentName, COUNT(PatientID) AS PatientCount
FROM Departments JOIN Patient ON Departments.DepartmentID = Patient.DepartmentID
WHERE @depName = DepartmentName
GROUP BY DepartmentName;
END
BEGIN
IF @@ROWCOUNT = 0
BEGIN
    THROW 50000, 'Invalid department name or department has no patients.', 1;
END
END
GO
```

Figure 3. Stored Procedures

The first procedure selects the patient name, start/end time of the appointment, appointment description, and medical staff name. It takes @time as a parameter and joins Patient, Appointments, MedicalStaff, and Employees. In the WHERE clause, it adds 30 minutes to the inputted time and checks if there is an appointment time that is greater than or equal to it, and if the end time is greater.

The second procedure, PatientsInDepartment, takes an input @depName then selects the total number of patients within that department using COUNT(). Then as a check, if the amount of rows returned is 0, then the department doesn't exist within the database. In this case, the procedure throws an error.

```
CREATE FUNCTION OverAmount(
    @total MONEY
)
RETURNS TABLE AS
RETURN
    SELECT DepartmentName, TotalCost
    FROM Costs JOIN Departments ON Costs.DepartmentID = Departments.DepartmentID
    WHERE TotalCost > @total;
GO

CREATE FUNCTION PatientLookUpByLastName(
    @PatientLast varchar(50)
)
RETURNS TABLE AS
RETURN
    SELECT (LastName + ', ' + FirstName) as PatientName, DepartmentName, RoomNumber
    FROM Patient JOIN Departments ON Patient.DepartmentID = Departments.DepartmentID
    WHERE LastName = @PatientLast;
```

Figure 4. User Functions

The OverAmount function takes @total as a parameter, which is of the datatype MONEY. It selects columns DepartmentName and TotalCost and returns any rows that have total cost that is greater than @total.

PatientLookUpByLastName takes @PatientLast varchar(50) as the parameter. The function selects the patient's name, department name, and room number. It then returns rows if @PatientLast matches any patient last names that exist in the database.

```
BEGIN TRANSACTION DeletePastAppointments
DELETE FROM Appointments WHERE EndTime < GETDATE()
COMMIT TRAN;
GO

BEGIN TRANSACTION DeleteVisitors
DELETE FROM Visitors WHERE ExitTime < GETDATE()
COMMIT TRAN;
```

Figure 5. Transactions

Figure 5 displays the code used to create the 2 transactions in the database. Both transactions get the current date and check if the end time or the exit time is passed the current date. If the date has passed, that entry gets deleted from the tables.

```
DROP TRIGGER IF EXISTS NewPatientInsert;
GO
CREATE TRIGGER NewPatientInsert ON Patient INSTEAD OF INSERT
AS BEGIN
    DECLARE @facID int;
    DECLARE @depID int;
    DECLARE @firstname varchar(50);
    DECLARE @lastname varchar(50);
    DECLARE @middleinit varchar(2);
    DECLARE @address varchar(100);
    DECLARE @phone varchar(50);
    DECLARE @email varchar(50);
    IF @facID IS NULL SET @facID = 1;
    IF @depID IS NULL SET @depID = 1;
    INSERT INTO Patient (DepartmentID, FacilityID, FirstName, LastName, MiddleInitial, Address, Phone, Email) VALUES
        (@depID, @facID, @firstname, @lastname, @middleinit, @address, @phone, @email);
END;
```

Figure 6. Trigger

The code for the trigger is shown in Figure 6. First, it checks if a trigger already exists. If one does exist by the same name, it gets dropped and a new one is created. It then declares the necessary variables in order to create a new entry in Patient, and replaces null values in @facID and @depID with a value of 1.

```
CREATE ROLE Administrator
GRANT INSERT, UPDATE, DELETE ON SCHEMA::UMCDatabase TO Administrator
EXEC sp_addrolemember db_datareader, Administrator;
GO

CREATE ROLE DOCTOR
GRANT INSERT, UPDATE, DELETE ON Patient TO DOCTOR
GRANT SELECT, INSERT, UPDATE, DELETE ON Appointments TO DOCTOR
GRANT SELECT, INSERT, UPDATE, DELETE ON PatientMedRecords TO DOCTOR
GRANT SELECT, INSERT, UPDATE, DELETE ON Prescriptions TO DOCTOR
GRANT SELECT ON LabResults TO DOCTOR
GRANT SELECT ON EmployeeSchedule TO DOCTOR;
GO
```

Figure 7. Roles

Figure 7 implements 2 roles to the database, Administrator and DOCTOR. Administrator is able to SELECT from any table since it's a part of db_datareader. It can also INSERT, UPDATE, or DELETE from any table in the database. The DOCTOR role is more restricted than the Administrator role and can only SELECT, INSERT, UPDATE, and DELETE from specified tables that are relevant to the job.

TESTING

All data within the testing can be referenced in the data insertion script in the appendix.

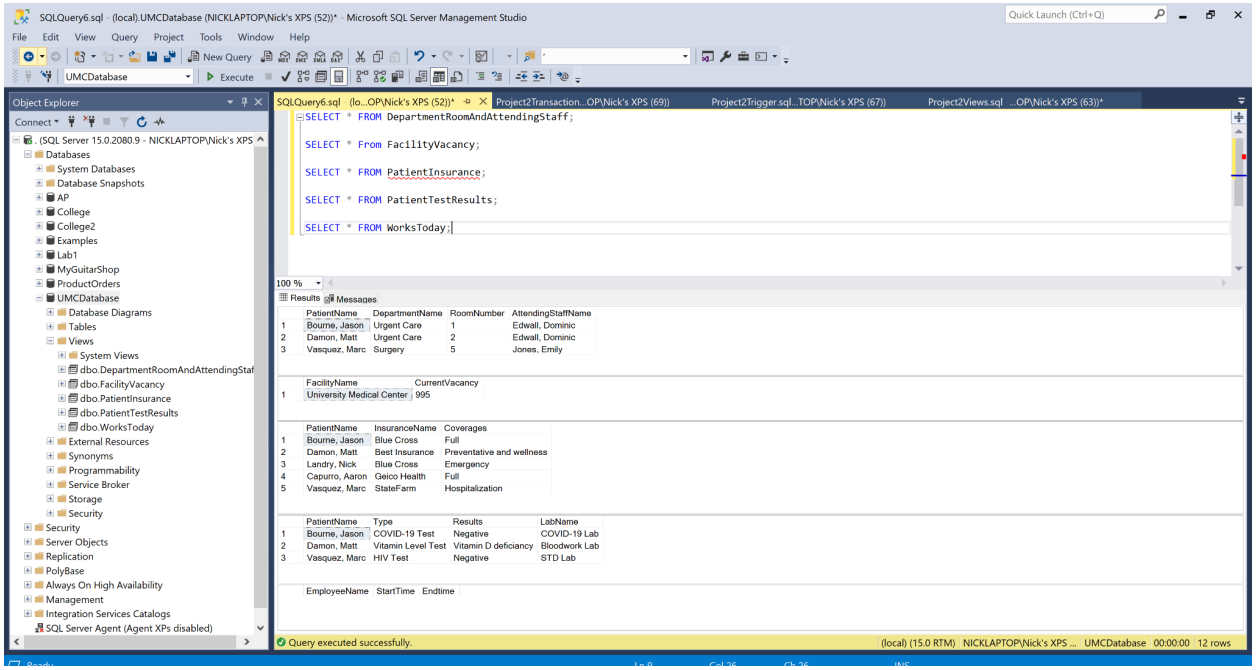

Figure 8. View Test Results

Figure 8 displays the testing for the views. The reason why the result for WorksToday is empty is because the shifts that are currently in the database are for 05/18/2021 and 05/19/2021. The testing was performed on 05/20/2021.

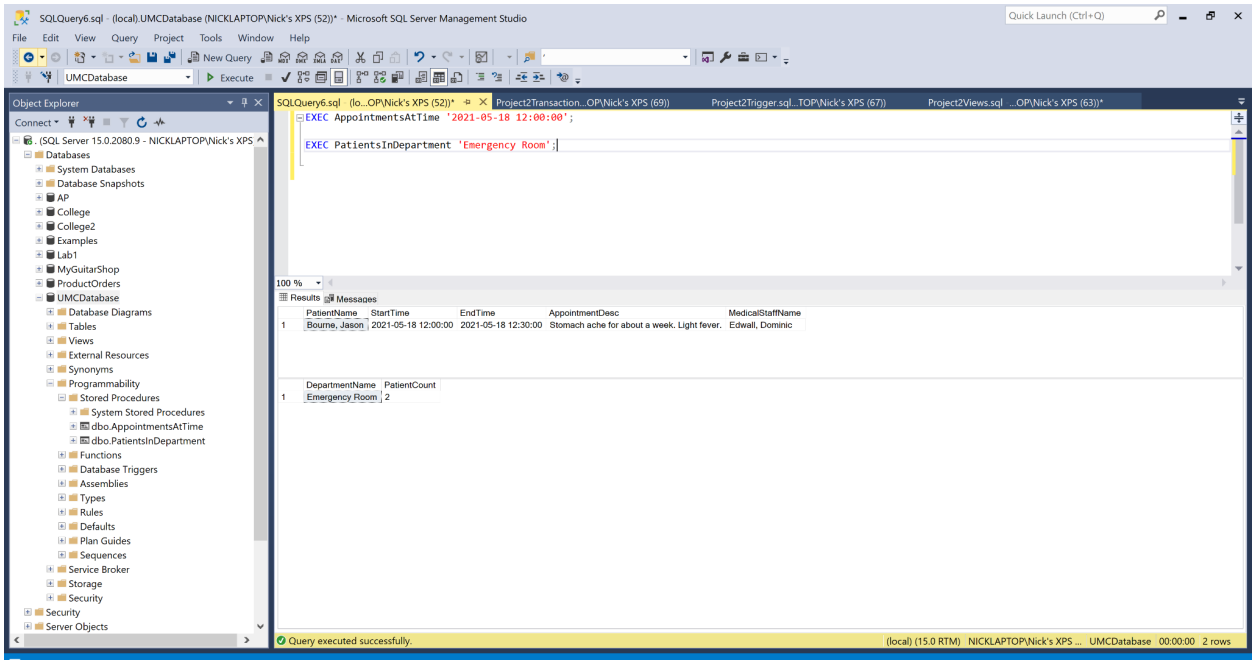

Figure 9. Stored Procedures Results

Both of the stored procedures function correctly and return the correct results. Figure 10 displays the case where there is not a department name that matches the inputted name.

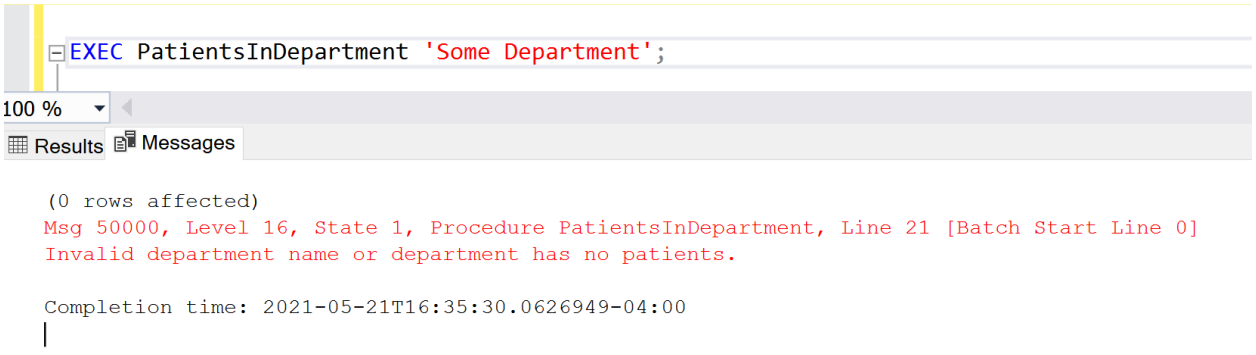

Figure 10. Stored Procedure Error Test

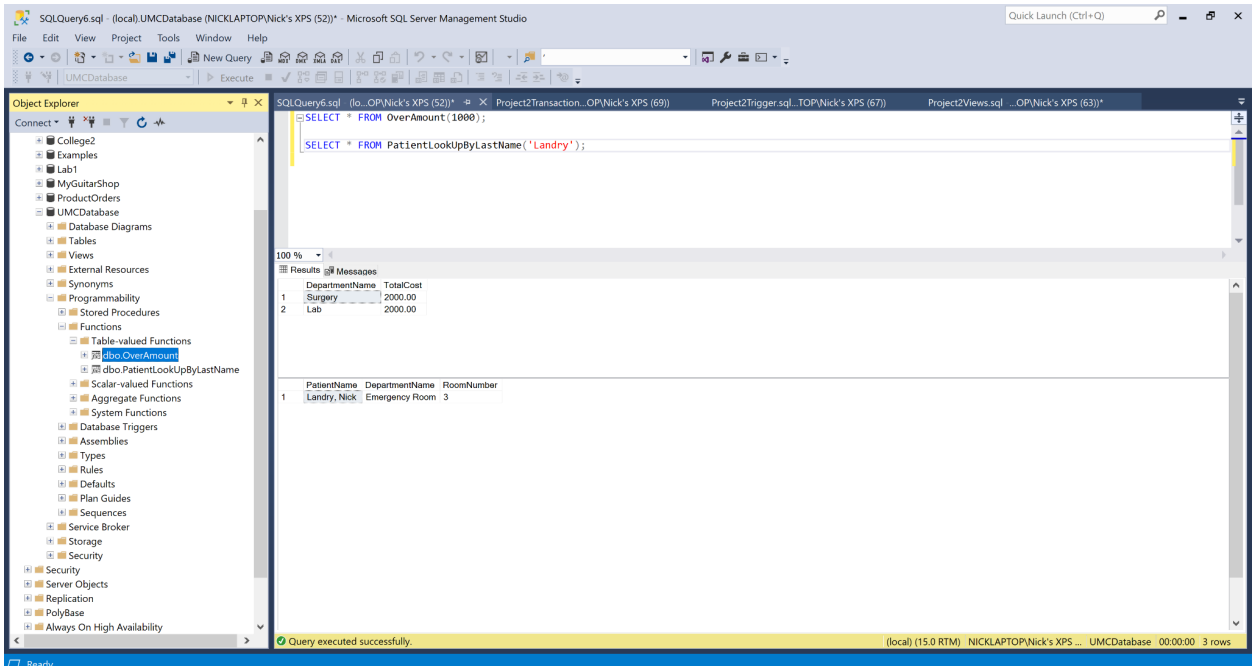

Figure 11. User Function Test

The above figure shows the test results for the user defined functions. The OverAmount function correctly displays the costs that are over $1000 and the PatientLookUpByLastName correctly returns patients that match the last name in the parameter.

```
SELECT * FROM Appointments;
SELECT * FROM Visitors;
```

100 %

Results | Messages

	AppointmentID	PatientID	MedicalStaffID	AppointmentDesc	StartTime	EndTime
1	1	1	1	Stomach ache for about a week. Light fever.	2021-05-18 12:00:00	2021-05-18 12:30:00
2	2	2	1	Chills and body aches	2021-05-18 13:00:00	2021-05-18 13:30:00
3	3	5	2	Brain surgery	2021-05-19 09:00:00	2021-05-19 10:00:00

	VisitorID	PatientID	FirstName	LastName	EntryTime	ExitTime
1	1	5	James	Vasquez	2021-05-18 12:00:00	2021-05-18 12:00:00

Figure 12. Before Transactions

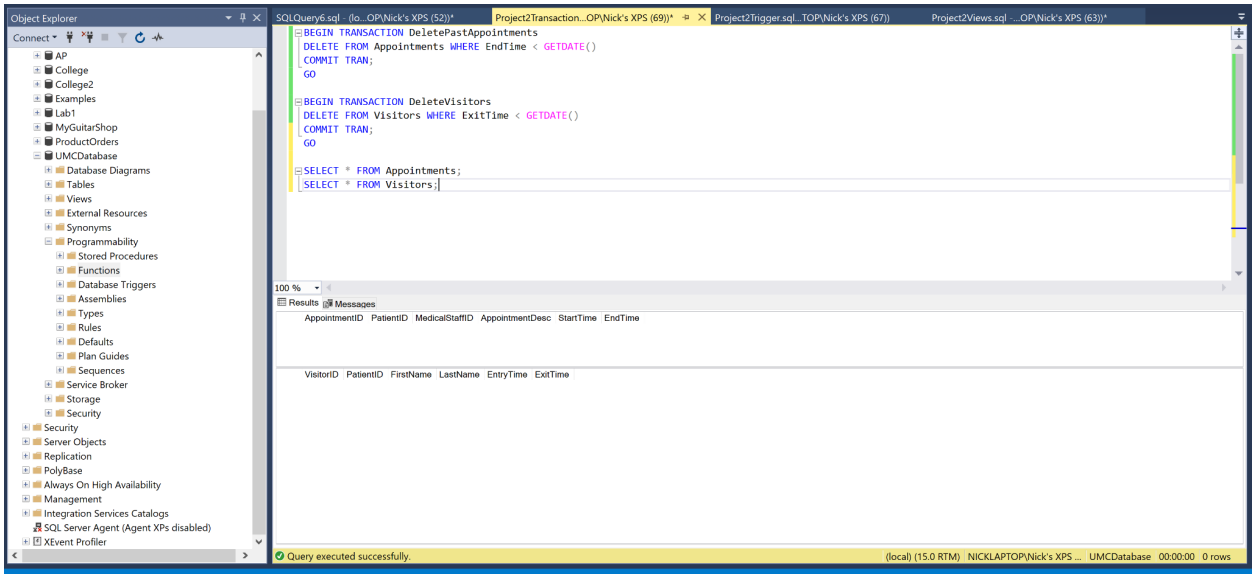

Figure 13. After Transactions

Figures 12 and 13 display the before and after of the 2 transactions for the database. Before, you can see that there was information about visitors and appointments still stored in the DB. After the transactions, these entries were deleted from their tables.

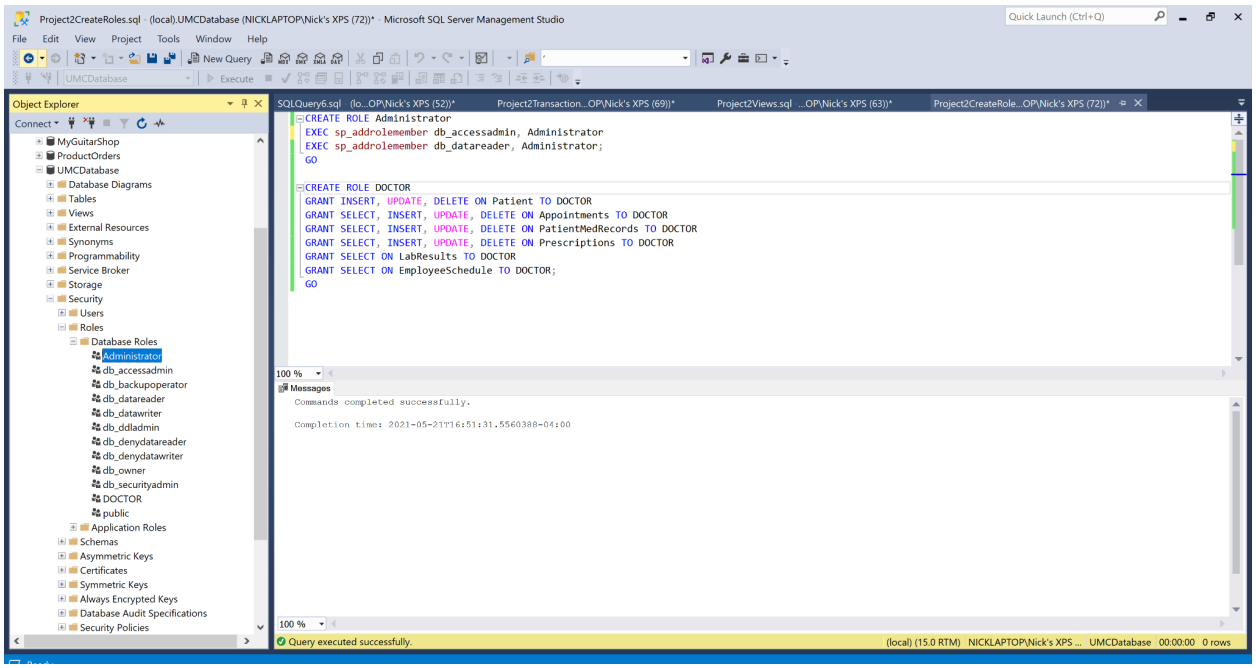

Figure 14. Role Creation Results

Figure 14 shows that creation of the 2 new roles were successful.

Conclusion

This report covers the database's functionality with sample data that may not be able to exhibit its full potential. In the future, it could handle thousands of entries. This project was very comprehensive and required knowledge from the whole semester in CSE 581. It served as a good way to demonstrate knowledge about SQL server database management and database design.

APPENDIX

1. **Project2DBCreation.sql**

```
USE master;
GO

IF DB_ID('UMCDatabase') IS NOT NULL
        DROP DATABASE UMCDatabase;
GO

CREATE DATABASE UMCDatabase;
GO

USE UMCDatabase;

-- Table: Facility
CREATE TABLE Facility (
   FacilityID int  PRIMARY KEY IDENTITY,
   FacilityName varchar(50)  NOT NULL,
   Capacity int  NOT NULL,
   [Location] varchar(100)  NOT NULL
);


CREATE TABLE Departments (
   DepartmentID int  PRIMARY KEY IDENTITY,
   FacilityID int  REFERENCES Facility (FacilityID),
   DepartmentName varchar(50)  NOT NULL,
   Phone varchar(50)  NOT NULL
  );

-- Table: Patient
CREATE TABLE Patient (
   PatientID int  PRIMARY KEY IDENTITY,
   DepartmentID int REFERENCES Departments (DepartmentID),
   FacilityID int  REFERENCES Facility (FacilityID),
   FirstName varchar(50)  NOT NULL,
   LastName varchar(50)  NOT NULL,
   MiddleInitial varchar(2)  NULL,
   [Address] varchar(100)  NOT NULL,
   Phone varchar(50)  NOT NULL,
   Email varchar(50)  NOT NULL UNIQUE,
   RoomNumber int NULL
```

```
);

CREATE TABLE Appointments (
    AppointmentID int  PRIMARY KEY IDENTITY,
    PatientID int  REFERENCES Patient (PatientID),
    MedicalStaffID int  NOT NULL,
    AppointmentDesc varchar(200)  NOT NULL,
    StartTime smalldatetime  NOT NULL,
    EndTime smalldatetime  NOT NULL
);

-- Table: Insurance
CREATE TABLE Insurance (
    InsuranceID int  PRIMARY KEY IDENTITY,
    PatientID int  REFERENCES Patient (PatientID),
    InsuranceName varchar(50)  NOT NULL,
    Phone varchar(50)  NOT NULL,
    Coverages varchar(200)  NOT NULL
);

-- Table: Billing
CREATE TABLE Billing (
    BillingID int  PRIMARY KEY IDENTITY,
    PatientID int  REFERENCES Patient (PatientID),
    InsuranceID int  REFERENCES Insurance (InsuranceID),
    BillingCode varchar(50)  NOT NULL,
    Payer varchar(50)  NOT NULL,
    PaymentMethod varchar(50)  NOT NULL,
    CardNumber varchar(50)  NULL

);

-- Table: Costs
CREATE TABLE Costs (
    CostID int  PRIMARY KEY IDENTITY,
    DepartmentID int  REFERENCES Departments (DepartmentID),
    CostDescription varchar(200)  NOT NULL,
    TotalCost money  NOT NULL
);


-- Table: EmergencyContact
CREATE TABLE EmergencyContact (
    EmergencyContactID int  PRIMARY KEY IDENTITY,
```

```
    PatientID int  REFERENCES Patient (PatientID),
    FirstName varchar(50)  NOT NULL,
    LastName varchar(50)  NOT NULL,
    Address varchar(200)  NOT NULL,
    Phone varchar(50)  NOT NULL,
    Email varchar(50)  NOT NULL UNIQUE
);

-- Table: Employees
CREATE TABLE Employees (
    EmployeeID int  PRIMARY KEY IDENTITY,
    DepartmentID int  REFERENCES Departments (DepartmentID),
    FirstName varchar(50)  NOT NULL,
    LastName varchar(50)  NOT NULL,
    Address varchar(100)  NOT NULL,
    Phone varchar(50)  NOT NULL,
    Email varchar(50)  NOT NULL UNIQUE,
    Salary money  NOT NULL
);

-- Table: EmployeeSchedule
CREATE TABLE EmployeeSchedule (
    ScheduleID int  PRIMARY KEY IDENTITY,
    EmployeeID int  REFERENCES Employees (EmployeeID),
    StartTime smalldatetime  NOT NULL,
    EndTime smalldatetime  NOT NULL
);


-- Table: Equipment
CREATE TABLE Equipment (
    EquipmentID int PRIMARY KEY IDENTITY,
    DepartmentID int  REFERENCES Departments (DepartmentID),
    VendorID int  NOT NULL,
    EquipmentName varchar(50)  NOT NULL,
    EquipmentDescription varchar(200)  NOT NULL,
    InUse bit  NOT NULL,
    Operational bit  NOT NULL
);


-- Table: LabDepartment
CREATE TABLE LabDepartment (
    DepartmentID int REFERENCES Departments (DepartmentID),
```

```
        LabID int PRIMARY KEY IDENTITY,
    LabName varchar(50)  NOT NULL,
    LabDescription varchar(200)  NOT NULL
);

-- Table: LabResults
CREATE TABLE LabResults (
    LabResultID int  PRIMARY KEY IDENTITY,
    PatientID int  REFERENCES Patient (PatientID),
    LabID int  REFERENCES LabDepartment (LabID),
    [Type] varchar(50)  NOT NULL,
    Results varchar(200)  NOT NULL
        );

-- Table: MedicalStaff
CREATE TABLE MedicalStaff (
    MedicalStaffID int  PRIMARY KEY IDENTITY,
    EmployeeID int  REFERENCES Employees (EmployeeID),
    [Type] varchar(50)  NOT NULL,
    [Status] varchar(50)  NOT NULL,
    Position varchar(50)  NOT NULL
);



-- Table: PatientMedRecords
CREATE TABLE PatientMedRecords (
    RecordID int  PRIMARY KEY IDENTITY,
    PatientID int  REFERENCES Patient (PatientID),
    MedicalStaffID int  REFERENCES MedicalStaff (MedicalStaffID),
    [Weight] int  NOT NULL,
    CheckIn smalldatetime  NULL,
    CheckOut smalldatetime  NULL,
    Symptoms varchar(200)  NOT NULL,
    Medication varchar(200)  NULL,
    Hospitalization varchar(200)  NULL,
    Discharge varchar(200)  NULL
);

-- Table: PharmacyDepartment
CREATE TABLE PharmacyDepartment (
    DepartmentID int REFERENCES Departments (DepartmentID),
        PharmacyID int PRIMARY KEY IDENTITY,
    PharmacyName varchar(50)  NOT NULL
```

```
);

-- Table: Prescriptions
CREATE TABLE Prescriptions (
    PrescriptionID int  PRIMARY KEY IDENTITY,
    PatientID int REFERENCES Patient (PatientID),
    PharmacyID int REFERENCES PharmacyDepartment (PharmacyID),
    PrescriptionName varchar(50)  NOT NULL,
    Description varchar(50)  NOT NULL,
    PrescriberID int  NOT NULL
);

-- Table: SurgeryDepartment
CREATE TABLE SurgeryDepartment (
    DepartmentID int REFERENCES Departments (DepartmentID),
    SurgeryID int  PRIMARY KEY IDENTITY,
    Type varchar(50)  NOT NULL
);

-- Table: Vendors
CREATE TABLE Vendors (
    VendorID int  PRIMARY KEY IDENTITY,
    VendorName varchar(50)  NOT NULL,
    Phone varchar(50)  NOT NULL,
    Address varchar(100)  NOT NULL,
    Email varchar(50)  NOT NULL UNIQUE
);

-- Table: Visitors
CREATE TABLE Visitors (
    VisitorID int  PRIMARY KEY IDENTITY,
    PatientID int  REFERENCES Patient (PatientID),
    FirstName varchar(50)  NOT NULL,
    LastName varchar(50)  NOT NULL,
    EntryTime smalldatetime  NULL,
    ExitTime smalldatetime  NULL
);
```

2. **Project2INSERTScript.sql**

```
USE UMCDatabase;
GO
```

```
SET IDENTITY_INSERT Facility ON;

INSERT INTO Facility (FacilityID, FacilityName, Capacity, Location) VALUES
	(1, 'University Medical Center', 1000, 'Syracuse, NY');

SET IDENTITY_INSERT Facility OFF;

SET IDENTITY_INSERT Departments ON;

INSERT INTO Departments (DepartmentID, FacilityID, DepartmentName, Phone) VALUES
	(1, 1, 'Urgent Care', '(111) 111-1111'),
	(2, 1, 'Emergency Room', '(111) 111-1122'),
	(3, 1, 'Pharmacy', '(111) 111-1133'),
	(4, 1, 'Surgery', '(111) 111-1144'),
	(5, 1, 'Lab', '(111) 111-1155');

SET IDENTITY_INSERT Departments OFF;

SET IDENTITY_INSERT Patient ON;

INSERT INTO Patient (PatientID, DepartmentID, FacilityID, FirstName, LastName, MiddleInitial,
Address, Phone, Email, RoomNumber) VALUES
	(1, 1, 1, 'Jason', 'Bourne', NULL, '1 Comstock Ave, Syracuse, NY', '(222) 222-2222',
'jasonbourne@gmail.com', 1),
	(2, 1, 1, 'Matt', 'Damon', NULL, '2 Comstock Ave, Syrcause, NY', '(222) 222-2211',
'mattdamon@gmail.com', 2),
	(3, 2, 1, 'Nick', 'Landry', 'G.', '1 Waverly Ave, Syracuse, NY', '(222) 222-2233',
'nicklandry@hotmail.com', 3),
	(4, 2, 1, 'Aaron', 'Capurro', NULL, '2 Marshall Street, Syracuse, NY', '(222) 222-2244',
'aaroncapurro@yahoo.com', 4),
	(5, 4, 1, 'Marc', 'Vasquez', NULL, '3 Euclid Ave, Syracuse, NY', '(222) 222-2255',
'marcvasquez@gmail.com', 5);

SET IDENTITY_INSERT Patient OFF;

SET IDENTITY_INSERT Appointments ON;

INSERT INTO Appointments (AppointmentID, PatientID, MedicalStaffID,AppointmentDesc,
StartTime, EndTime) VALUES
	(1, 1, 1, 'Stomach ache for about a week. Light fever.', '2021-5-18 12:00:00', '2021-5-18
12:30:00'),
	(2, 2, 1, 'Chills and body aches', '2021-5-18 13:00:00', '2021-5-18 13:30:00'),
	(3, 5, 2, 'Brain surgery', '2021-5-19 09:00:00', '2021-5-19 10:00:00');
```

```
SET IDENTITY_INSERT Appointments OFF;

SET IDENTITY_INSERT Insurance ON;

INSERT INTO Insurance (InsuranceID, PatientID, InsuranceName, Phone, Coverages) VALUES
	(1, 1, 'Blue Cross' , '(333) 333-1111', 'Full'),
	(2, 2, 'Best Insurance', '(132) 123-1234', 'Preventative and wellness'),
	(3, 3, 'Blue Cross', '(333) 333-1111', 'Emergency'),
	(4, 4, 'Geico Health', '(901) 901-0901', 'Full'),
	(5, 5, 'StateFarm', '(333) 333-1133', 'Hospitalization');

SET IDENTITY_INSERT Insurance OFF;

SET IDENTITY_INSERT Billing ON;

INSERT INTO Billing (BillingID, PatientID, InsuranceID, BillingCode, Payer, PaymentMethod,
CardNumber) VALUES
	(1, 1, 1, 10002, 'Insurance', 'Insurance', NULL),
	(2, 2, 2, 10026, 'Self', 'credit card', '1111-2222-3333-4444'),
	(3, 3, 3, 20109, 'Insurance', 'Insurance', NULL),
	(4, 4, 4, 50212, 'Insurance', 'Insurance', NULL),
	(5, 5, 5, 36281, 'Relative', 'credit card', '1234-1234-1234-1234');

SET IDENTITY_INSERT BILLING OFF;

SET IDENTITY_INSERT COSTS ON;

INSERT INTO Costs (CostID, DepartmentID, CostDescription, TotalCost) VALUES
	(1, 4, 'New surgery utensils' , 2000),
	(2, 3, 'prescription containers', 1000),
	(3, 5, 'new beakers', 2000);

SET IDENTITY_INSERT COSTS OFF;

SET IDENTITY_INSERT EmergencyContact ON;

INSERT INTO EmergencyContact (EmergencyContactID, PatientID, FirstName, LastName,
Address, Phone, Email) VALUES
	(1, 4, 'Joseph', 'Capurro', '6 Waterway Place, Chicago, IL', '(444) 444-4411',
'josephcapurro@gmail.com'),
	(2, 2, 'Lawrence', 'Lauren', '426 Deerfield Ave, Berkeley, CA', '(444) 441-4422',
'lawrencelauren@yahoo.com'),
	(3, 1, 'George', 'Jungle', '909 Jungle Street, Syracuse, NY', '(444) 451-5554',
'georgeofthejungle@jungle.com');
```

```
SET IDENTITY_INSERT EmergencyContact OFF;

SET IDENTITY_INSERT Employees ON;

INSERT INTO Employees (EmployeeID, DepartmentID, FirstName, LastName, Address,
Phone, Email, Salary) VALUES
        (1, 1, 'Nathan', 'Correll', '52 Cunningham Drive, Syracuse, NY', '(555) 555-5115',
'nathancorrell@hotmail.com', 60000),
        (2, 4, 'Dominic', 'Edwall', '6 Euclid Ave, Syracuse, NY', '(555) 555-5522',
'dominicedwall@yahoo.com', 200000),
        (3, 2, 'Emily', 'Jones', '42 Hinge Place, Syracuse, NY', '(555) 555-5533',
'emilyjones@gmail.com', 73000),
        (4, 3, 'Keith', 'Samsa', '51 Curry Street, Syracuse, NY', '(555) 555-5544',
'keithsamsa@yahoo.com', 50000),
        (5, 5, 'Barbara', 'Jean', '89 Mondstat Ave, Syracuse, NY', '(555) 555-5555',
'barbarajean@gmail.com', 120000);

SET IDENTITY_INSERT Employees OFF;

SET IDENTITY_INSERT EmployeeSchedule ON;

INSERT INTO EmployeeSchedule (ScheduleID, EmployeeID, StartTime, EndTime) VALUES
        (1, 1, '2021-05-19 09:00:00', '2021-05-19 17:00:00'),
        (2, 2, '2021-05-19 05:00:00', '2021-05-19 16:00:00'),
        (3, 3, '2021-05-20 10:00:00', '2021-05-20 19:00:00'),
        (4, 4, '2021-05-19 12:00:00', '2021-05-19 23:00:00'),
        (5, 5, '2021-05-20 06:00:00', '2021-05-20 14:00:00');

SET IDENTITY_INSERT EmployeeSchedule OFF;

SET IDENTITY_INSERT Equipment ON;

INSERT INTO Equipment (EquipmentID, DepartmentID, VendorID, EquipmentName,
EquipmentDescription, InUse, Operational) VALUES
        (1, 2, 1, 'Ventilator', 'Assists breathing by pumping oxygen into airways', 0, 1),
        (2, 1, 2, 'Heartbeat Monitor', 'Displays heartbeat of connected patient', 1, 1),
        (3, 4, 3, 'Da Vinci Surgery Robot', 'Surgery robot capable of precise operations', 0, 0);

SET IDENTITY_INSERT Equipment OFF;

SET IDENTITY_INSERT LabDepartment ON;

INSERT INTO LabDepartment (DepartmentID, LabID, LabName, LabDescription) VALUES
```

```
        (5, 1, 'Bloodwork Lab', 'Tests for various things concerning blood'),
        (5, 2, 'COVID-19 Lab', 'Tests for COVID-19'),
        (5, 3, 'STD Lab', 'Tests for STDs');

SET IDENTITY_INSERT LabDepartment OFF;

SET IDENTITY_INSERT LabResults ON;

INSERT INTO LabResults (LabResultID, PatientID, LabID, Type, Results) VALUES
        (1, 1, 2, 'COVID-19 Test', 'Negative'),
        (2, 2, 1, 'Vitamin Level Test', 'Vitamin D deficiancy'),
        (3, 5, 3, 'HIV Test', 'Negative');

SET IDENTITY_INSERT LabResults OFF;

SET IDENTITY_INSERT MedicalStaff ON;

INSERT INTO MedicalStaff (MedicalStaffID, EmployeeID, Type, Status, Position) VALUES
        (1, 2, 'Surgeon', 'Attending', 'Head brain surgeon'),
        (2, 3, 'Nurse/Assistant', 'Visiting', 'General nurse'),
        (3, 1, 'Doctor', 'Attending', 'On-Call doctor');

SET IDENTITY_INSERT MedicalStaff OFF;

SET IDENTITY_INSERT PatientMedRecords ON;

INSERT INTO PatientMedRecords (RecordID, PatientID, MedicalStaffID, Weight, CheckIn,
CheckOut, Symptoms, Medication, Hospitalization, Discharge) VALUES
        (1, 1, 3, 180, '2021-05-19 11:00:00', NULL, 'Stomach ache and light fever for about a
week', NULL, NULL, NULL),
        (2, 2, 3, 180, '2021-05-19 12:00:00', NULL, 'Chill and body aches', NULL, NULL, NULL),
        (3, 5, 1, 200, '2021-05-19 10:00:00', NULL, 'Major blunt head trauma', NULL,
'Hospitalized', NULL);

SET IDENTITY_INSERT PatientMedRecords OFF;

SET IDENTITY_INSERT PharmacyDepartment ON;

INSERT INTO PharmacyDepartment ( DepartmentID, PharmacyID, PharmacyName) VALUES
        (3, 1, 'Pain Medication'),
        (3, 2, 'Anti-depressants'),
        (3, 3, 'Antihistamines');

SET IDENTITY_INSERT PharmacyDepartment OFF;
```

```
SET IDENTITY_INSERT Prescriptions ON;

INSERT INTO Prescriptions (PrescriptionID, PatientID, PharmacyID, PrescriberID,
PrescriptionName, Description) VALUES
        (1, 1, 1, 3, 'Pain Meds', 'Codeine'),
        (2, 5, 1, 1, 'Pain Meds', 'Opioids'),
        (3, 2, 3, 3, 'Anti-anxiety', 'Hydroxyzine');

SET IDENTITY_INSERT Prescriptions OFF;

SET IDENTITY_INSERT SurgeryDepartment ON;

INSERT INTO SurgeryDepartment (DepartmentID, SurgeryID, Type) VALUES
        (4, 1, 'Brain'),
        (4, 2, 'Heart'),
        (4, 3, 'Limb');

SET IDENTITY_INSERT SurgeryDepartment OFF;

SET IDENTITY_INSERT Vendors ON;

INSERT INTO Vendors ( VendorID, VendorName, Phone, Address, Email) VALUES
        (1, 'Venting INC', '(666) 666-6666' , '6 Waverly Place, New York, NY',
'ventinginc@gmail.com'),
        (2, 'Healthy Hearts', '(666) 666-6611', '736 Euclid Ave, Syracuse, NY',
'hhearts@gmail.com'),
        (3, 'Da Vinci Tech', '(666) 666-6622', '102 Generation Street, San Francisco, CA',
'davincitech@yahoo.com');

SET IDENTITY_INSERT Vendors OFF;

SET IDENTITY_INSERT Visitors ON;

INSERT INTO Visitors (VisitorID, PatientID, FirstName, LastName, EntryTime, ExitTime)
VALUES
        (1, 5, 'James', 'Vasquez', '2021-05-18 12:00:00', '2021-05-18 12:00:00');

SET IDENTITY_INSERT Visitors OFF;
```