Project 2

HEALTH CARE SYSTEM DATABASE

CSE 581

Mandar Mathure

299973901

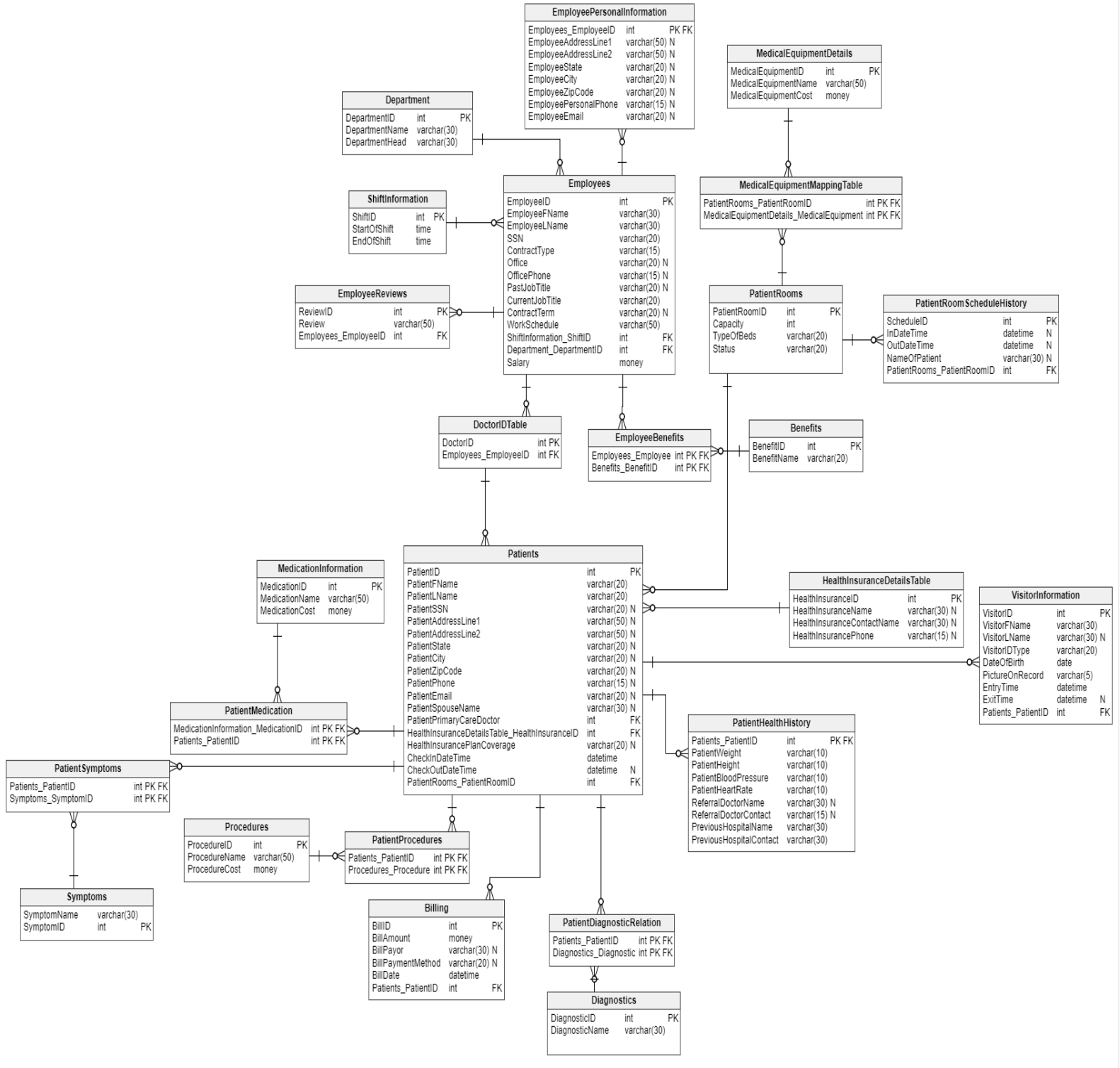
ABSTRACT

This project focuses on designing a database model for a Health Care System. The entities are modelled according to real world entities using the online database creation tool Vertabelo. The project uses all the core concepts of database design and development along with test scenarios with SQL code. The test scenarios are mainly used to depict the reliability of the database system that is designed. The design also addresses potential integrity and security issues of the data that would be inserted into the database. The database design is normalized upto 3rd normal form and a huge piece of data is split into small corresponding columns so that it would be easier to parse in future. Entity relationships like foreign keys are very well defined that correspond to the real world relationships between two entities that are in consideration. For example a relationship between doctors and patients , patients and medications and so on. Later on, several scenarios are presented that provide possible test cases and the reliability of the system designed is depicted by using several complicated SQL queries. The tools used for the design are:

Vertabelo: An online database design tool available for free at http://www.vertabelo.com

Microsoft SQL Management Server: for effective testing and simulation

ER DIAGRAM

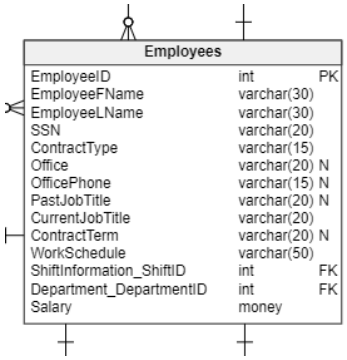

EmployeePersonalInformation
Employees_EmployeeID int PK FK
EmployeeAddressLine1 varchar(50) N
EmployeeAddressLine2 varchar(50) N
EmployeeState varchar(20) N
EmployeeCity varchar(20) N
EmployeeZipCode varchar(20) N
EmployeePersonalPhone varchar(15) N
EmployeeEmail varchar(20) N
MedicalEquipmentDetails
MedicalEquipmentID int PK
MedicalEquipmentName varchar(50)
MedicalEquipmentCost money
Department
DepartmentID int PK
DepartmentName varchar(30)
DepartmentHead varchar(30)
Employees
EmployeeID int PK
EmployeeFName varchar(30)
EmployeeLName varchar(30)
SSN varchar(20)
ContractType varchar(15)
Office varchar(20) N
OfficePhone varchar(15) N
PastJobTitle varchar(20) N
CurrentJobTitle varchar(20)
ContractTerm varchar(20) N
WorkSchedule varchar(50)
ShiftInformation_ShiftID int FK
Department_DepartmentID int FK
Salary money
MedicalEquipmentMappingTable
PatientRooms_PatientRoomID int PK FK
MedicalEquipmentDetails_MedicalEquipment int PK FK
ShiftInformation
ShiftID int PK
StartOfShift time
EndOfShift time
EmployeeReviews
ReviewID int PK
Review varchar(50)
Employees_EmployeeID int FK
PatientRooms
PatientRoomID int PK
Capacity int
TypeOfBeds varchar(20)
Status varchar(20)
PatientRoomScheduleHistory
ScheduleID int PK
InDateTime datetime N
OutDateTime datetime N
NameOfPatient varchar(30) N
PatientRooms_PatientRoomID int FK
DoctorIDTable
DoctorID int PK
Employees_EmployeeID int FK
EmployeeBenefits
Employees_Employee int PK FK
Benefits_BenefitID int PK FK
Benefits
BenefitID int PK
BenefitName varchar(20)
Patients
PatientID int PK
PatientFName varchar(20)
PatientLName varchar(20)
PatientSSN varchar(20) N
PatientAddressLine1 varchar(50) N
PatientAddressLine2 varchar(50) N
PatientState varchar(20) N
PatientCity varchar(20) N
PatientZipCode varchar(20) N
PatientPhone varchar(15) N
PatientEmail varchar(20) N
PatientSpouseName varchar(30) N
PatientPrimaryCareDoctor int FK
HealthInsuranceDetailsTable_HealthInsuranceID int FK
HealthInsurancePlanCoverage varchar(20) N
CheckInDateTime datetime
CheckOutDateTime datetime N
PatientRooms_PatientRoomID int FK
MedicationInformation
MedicationID int PK
MedicationName varchar(50)
MedicationCost money
HealthInsuranceDetailsTable
HealthInsuranceID int PK
HealthInsuranceName varchar(30) N
HealthInsuranceContactName varchar(30) N
HealthInsurancePhone varchar(15) N
VisitorInformation
VisitorID int PK
VisitorFName varchar(30)
VisitorLName varchar(30) N
VisitorIDType varchar(20)
DateOfBirth date
PictureOnRecord varchar(5)
EntryTime datetime
ExitTime datetime N
Patients_PatientID int FK
PatientMedication
MedicationInformation_MedicationID int PK FK
Patients_PatientID int PK FK
PatientHealthHistory
Patients_PatientID int PK FK
PatientWeight varchar(10)
PatientHeight varchar(10)
PatientBloodPressure varchar(10)
PatientHeartRate varchar(10)
ReferralDoctorName varchar(30) N
ReferralDoctorContact varchar(15) N
PreviousHospitalName varchar(30)
PreviousHospitalContact varchar(30)
PatientSymptoms
Patients_PatientID int PK FK
Symptoms_SymptomID int PK FK
Procedures
ProcedureID int PK
ProcedureName varchar(50)
ProcedureCost money
PatientProcedures
Patients_PatientID int PK FK
Procedures_Procedure int PK FK
Symptoms
SymptomName varchar(30)
SymptomID int PK
Billing
BillID int PK
BillAmount money
BillPayor varchar(30) N
BillPaymentMethod varchar(20) N
BillDate datetime
Patients_PatientID int FK
PatientDiagnosticRelation
Patients_PatientID int PK FK
Diagnostics_Diagnostic int PK FK
Diagnostics
DiagnosticID int PK
DiagnosticName varchar(30)

DESIGN CONSIDERATIONS

ENTITIES

- Employees

The employees table provides necessary information of all the employees in the heal care system. having a common employees table enables the user to have the records for all the employees in one place. This table gives the general info about the employee like EmployeeID, Name, Work schedule , Job title are that required in almost daily activities.



Primary key : EmloyeeID

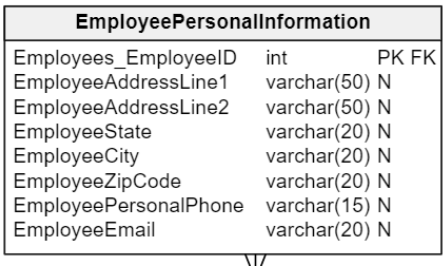
Every employee is provided a unique EmployeeID hence this column is well suited as a primary key.

Foreign keys: ShiftID, DepartmentID

The employees table references the shifts and the departments table to provide information about the time of work and the department of the employee like cardiology, neurology etc. This establishes a many to one relationship between employees and departments.

- EmployeePersonalInformation

This table is just an extension of the employees table as it provides additional information about the employee like address, personal phone etc. Since it's an extension, it has a one to one relationship between employees.

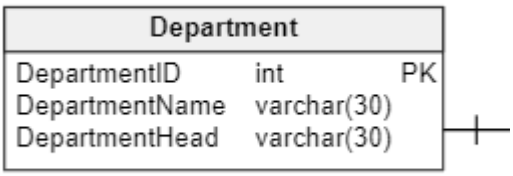


PK and FK : EmployeeID

This depicts a one to one relationship between Employees and EmployeePersonalInformation.

- Department

This table provides information about the department that an employee currently works. The employees and departments table have a one to many relationship between them.



PK : DepartmentID

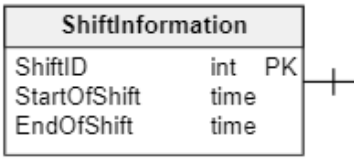
Each department is identified by a unique ID that can be referenced in Employees table as the foreign key.

- ShiftInformation

This table provides the time schedule and the shift details about a particular employee.

PK: ShiftID

Every shift is distinguished by a unique ID which is referenced in the Employees table as a foreign key depicting a one to many relationship between them. Every employee can have one shift at a time but every shift can have multiple employees.



- EmployeeReviews

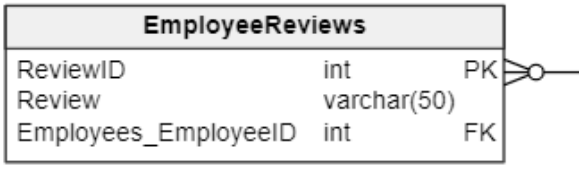
  I decided to keep a separate table for the reviews as one employee can have many reviews by different patients. The patient information is not referenced here by assuming that the reviews would be anonymous to the employees.

  PK: ReviewID

  Every Review has a unique identifier which is the primary key.

  FK: EmloyeeID

  The review table references the EmployeeID column of the Employees table depicting a one to many relationship between the Employee and reviews.

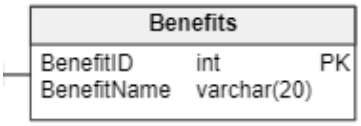


- Benefits

  This table categorizes all the benefits given to the employees in the health care system. One employee can have many benefits or no benefits at all.

  PK: BenefitID

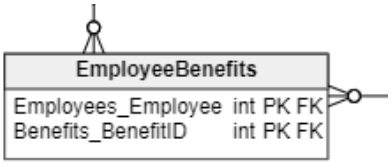
  Every Unique benefit is identified by a unique ID which is the primary key.



- EmployeeBenefits

  This is a linking tables that establishes a many to many relationship between the Employees and the Benefits table.

  PK and FK : EmployeeID and BenefitID



- DoctorID

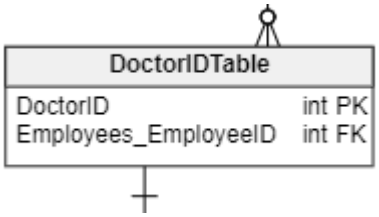
  This table has information about all the doctors their individual doctorId and their corresponding EmployeeID. Basically this table has information about an employee who is a doctor giving him/her a unique identifier as a DoctorID.

  PK: DoctorID

  Uniquely identifies each doctor in a group of doctors.

  FK:EmployeeID

  References the employees table to get the EmployeeID of the corresponding doctor.

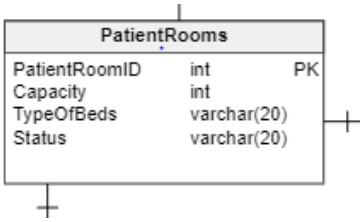


- PatientRooms

  This table provides information about the rooms in the health care center their location, status, types of beds etc.

  PK: PatientRoomID

  This uniquely identifies every room in the health care center.



- PatientRoomScheduleHistory

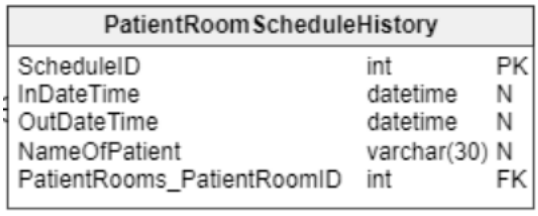
  This table provides the history of each room in the health care center. The column NameOfPatient contains the names of patients that once occupied the room and now have checked out.

  Basically once when a patient discharges or leaves a room his names should be entered into the schedule history table.

  PK: ScheduleID

  Every schedule history is identified by a unique ID

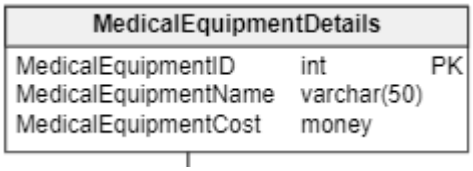
  FK: PatientRoomID .



This table references PatientRoomID from the PatientRooms table depicting a one to many relationship between the two tables.

- MedicalEquipmentDetails

This table provides information about all the medical equipments and their cost available in the health care center.

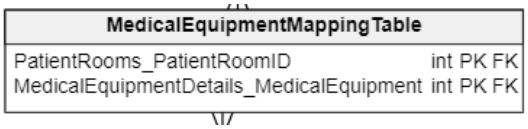


PK: MedicalEquipmentID

Each medical equipment is identified by a unique ID which is the primary key.

- MedicalEquipmentMappingTable

This table is basically a linking table that links the many to many relationship between the patient rooms and the medical equipments.



PK and FK: PatientRoomID and MedicalEquipmentID

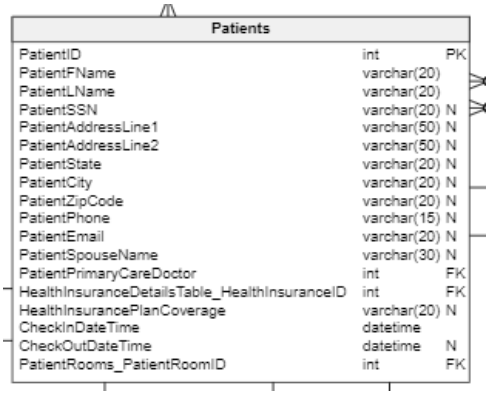
This table references two foreign keys from PatientRooms and MedicalEquipmentDetails table. Generally linking tables do not have primary keys but here the two foreign keys are also good candidates for primary keys as every combination of PatientRoomID and MedicalEquipmentID will be unique.

- Patients

The patients table provides information about the patients like name, address, primary care doctor etc.

PK: PatientID

Every patient is identified by his unique ID which is again referenced in other tables giving more information about the patients.



FK: PatientPrimaryCareDoctor, HealthInsuranceID, PatientRoomID

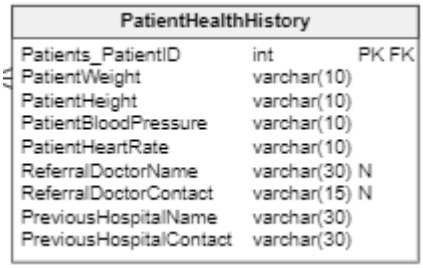
PatientPrimaryCareDoctor is referenced from the DoctorID table that maps the doctorsID to the corresponding employeeID of the doctor.

HealthInsuranceID references the HealthInsuranceDetails table giving information about the patients health insurance plan.

PatientRoomID is referenced from the PatientRooms table that gives information about which patient is a allotted what room.

- PatientHealthHistory

This table provides more information about the health history of the patient like vitals, referral doctor name etc. This table is basically an extension of the patients table.



PK and FK : PatientID

References the Patients table. Good candidate for PK and FK as it is a one to one relationship the two tables.

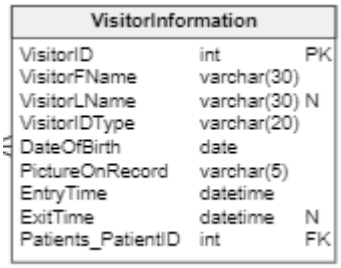
- HealthInsuranceDetailsTable

Provides details about various health insurances on record in the health care canter database.

PK: HealthInsuranceID

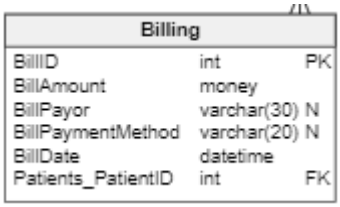
Uniquely identifies each health insurance record.

- VisitorInformation



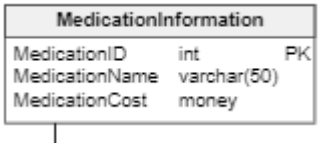
Provides information about visitors for patients.
PK: VisitorID
Each visitor is uniquely identified by his/her visitorID which is the primary key.
FK: PatientID
References PatientID from the patients table. This depicts a one to many relationship between patients and the Visitors.

- Billing



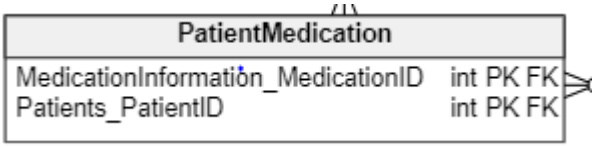
Gives information about billing details of each discharged patient. Whenever a bill is generated for a patient, an entry is made into this table that references the PatientID from the patients table. According to my design a bill will be generated for each patient that is discharged so in other words patients that are still in the health care center will have not have a corresponding bill.
PK: BillID
Uniquely identifies each bill in the database.
FK: PatientID
References PatientID from Patients table that establishes a one to many relationship between patients and bills as one patient can have many bills generated for his treatment.

- MedicationInformation



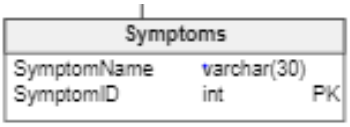
Gives information about each medication its name and cost.
PK: MedicationID
uniquel y identifies each medication by its ID

- PatientMedication



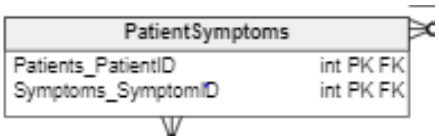
This is a linking table that establishes a many to many relationship between patients and medications.
PK and FK : MedicationID and PatientID

- Symptoms



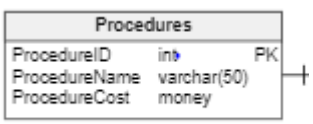
Provides information about the most common symptoms on record in the health care centre.
PK: SymptomID

- PatientSymptoms



A linking table that establishes a many to many relationship between patients and the symptoms.
PK and FK: PatientID and SymptomID

- Procedures

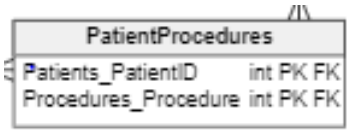


Provides information about different procedures that are done in the health care canter.
PK: ProcedureID
Indentifies each procedure by a unique ID

- PatientProcedures

  This is a lining table that provides a many to many relationship between Patients table and the procedures table.

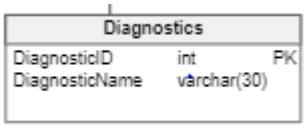
  PK and FK : PatientID and ProcedureID



- Diagnostics

  Gives details about all the most common diagnostics that are on record in the health care center.

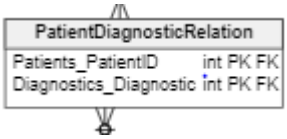
  PK: DiagnosticID



- PatientDiagnosticRelation

  This is a linking table that establishes a many to many relationship between patients and the Diagnostics table.

  PK and FK: PatientID and DiagnosticID



DATABASE CREATION SCRIPT

```
USE master
GO

/****** Object:  Database HealthCareCenter     ******/
IF DB_ID('HealthCareCenter') IS NOT NULL
       DROP DATABASE HealthCareCenter
GO

CREATE DATABASE HealthCareCenter
GO

USE HealthCareCenter
GO

--creating table Employees
CREATE TABLE Employees (
    EmployeeID int  NOT NULL IDENTITY,
    EmployeeFName varchar(30)  NOT NULL,
    EmployeeLName varchar(30)  NOT NULL,
    SSN varchar(20)  NOT NULL,
    ContractType varchar(15)  NOT NULL,
    Office varchar(20)  NULL,
    OfficePhone varchar(15)  NULL,
    PastJobTitle varchar(20)  NULL,
    CurrentJobTitle varchar(20)  NOT NULL,
    ContractTerm varchar(20)  NULL,
    WorkSchedule varchar(50)  NOT NULL,
    ShiftID int  NOT NULL, --FK
    DepartmentID int  NOT NULL,--FK
    Salary money  NOT NULL,
    CONSTRAINT PK_Employees PRIMARY KEY  (EmployeeID)
);

GO

--creating table EmployeePersonalInformation
CREATE TABLE EmployeePersonalInformation (
    EmployeeID int  NOT NULL IDENTITY, --FK
```

```
    EmployeeAddressLine1 varchar(50)  NULL,
    EmployeeAddressLine2 varchar(50)  NULL,
    EmployeeState varchar(20)  NULL,
    EmployeeCity varchar(20)  NULL,
    EmployeeZipCode varchar(20)  NULL,
    EmployeePersonalPhone varchar(15)  NULL,
    EmployeeEmail varchar(20)  NULL,
    CONSTRAINT PK_EmployeePersonalInformation PRIMARY KEY  (EmployeeID)

);

GO

--creating table Patients
CREATE TABLE Patients (
    PatientID int  NOT NULL IDENTITY,
    PatientFName varchar(20)  NOT NULL,
    PatientLName varchar(20)  NOT NULL,
    PatientSSN varchar(20)  NULL,
    PatientAddressLine1 varchar(50)  NULL,
    PatientAddressLine2 varchar(50)  NULL,
    PatientState varchar(20)  NULL,
    PatientCity varchar(20)  NULL,
    PatientZipCode varchar(20)  NULL,
    PatientPhone varchar(15)  NULL,
    PatientEmail varchar(20)  NULL,
    PatientSpouseName varchar(30)  NULL,
    PatientPrimaryCareDoctor int  NOT NULL, --FK
    HealthInsuranceID int  NOT NULL, --FK
    HealthInsurancePlanCoverage varchar(20)  NULL,
    CheckInDateTime datetime  NOT NULL,
    CheckOutDateTime datetime  NULL,
    PatientRoomID int  NOT NULL, --FK
    CONSTRAINT PK_Patients PRIMARY KEY  (PatientID)
);
GO

--creating table PatientHealthHistory
CREATE TABLE PatientHealthHistory (
    PatientID int  NOT NULL IDENTITY, --FK
    PatientWeight varchar(10)  NOT NULL,
    PatientHeight varchar(10)  NOT NULL,
    PatientBloodPressure varchar(10)  NOT NULL,
    PatientHeartRate varchar(10)  NOT NULL,
    ReferralDoctorName varchar(30)  NULL,
    ReferralDoctorContact varchar(15)  NULL,
    PreviousHospitalName varchar(30)  NOT NULL,
    PreviousHospitalContact varchar(30)  NOT NULL,
    CONSTRAINT PK_PatientHealthHistory PRIMARY KEY  (PatientID)
);

GO

--creating table Benefits
CREATE TABLE Benefits (
    BenefitID int  NOT NULL IDENTITY,
    BenefitName varchar(30)  NOT NULL,
    CONSTRAINT PK_Benefits PRIMARY KEY  (BenefitID)
);
GO

--creating table Department
CREATE TABLE Department (
    DepartmentID int  NOT NULL IDENTITY,
    DepartmentName varchar(30)  NOT NULL,
```

```
    DepartmentHead varchar(30)  NOT NULL,
    CONSTRAINT UQ_DepartmentName_Department UNIQUE (DepartmentName),
    CONSTRAINT PK_Department PRIMARY KEY  (DepartmentID)
);

GO

--creating table DoctorIDTable
CREATE TABLE DoctorIDTable (
    DoctorID int  NOT NULL IDENTITY,
    EmployeeID int  NOT NULL, --FK
    CONSTRAINT PK_DoctorIDTable PRIMARY KEY  (DoctorID)
);

GO

--creating table EmployeeBenefits
CREATE TABLE EmployeeBenefits (
    EmployeeID int  NOT NULL, --FK
    BenefitID int  NOT NULL, --FK
    CONSTRAINT PK_EmployeeBenefits PRIMARY KEY  (EmployeeID,BenefitID)
);
GO

--creating table EmployeeReviews
CREATE TABLE EmployeeReviews (
    ReviewID int  NOT NULL IDENTITY,
    Review varchar(50)  NOT NULL,
    EmployeeID int  NOT NULL, --FK
    CONSTRAINT PK_EmployeeReviews PRIMARY KEY  (ReviewID)
);
GO

--creating table ShiftInformation
CREATE TABLE ShiftInformation (
    ShiftID int  NOT NULL IDENTITY,
    StartOfShift time(0)  NOT NULL,
    EndOfShift time(0)  NOT NULL,
    CONSTRAINT PK_ShiftInformation PRIMARY KEY  (ShiftID)
);
GO

--creating table PatientRooms
CREATE TABLE PatientRooms (
    PatientRoomID int  NOT NULL IDENTITY,
    Capacity int  NOT NULL,
    TypeOfBeds varchar(20)  NOT NULL,
    [Status] varchar(20)  NOT NULL,
       Location varchar(20) NOT NULL,
    CONSTRAINT PK_PatientRooms PRIMARY KEY  (PatientRoomID)
);
GO

--creating table PatientScheduleHistory
CREATE TABLE PatientRoomScheduleHistory (
    ScheduleID int  NOT NULL IDENTITY,
    InDateTime datetime  NOT NULL,
    OutDateTime datetime NOT NULL,
       NameOfPatient varchar(30)  NULL,
    PatientRoomID int  NOT NULL, --FK
    CONSTRAINT PK_PatientRoomScheduleHistory PRIMARY KEY  (ScheduleID)
);
GO

--creating table MedicalEquipmentDetails
```

```
CREATE TABLE MedicalEquipmentDetails (
    MedicalEquipmentID int  NOT NULL IDENTITY,
    MedicalEquipmentName varchar(50)  NOT NULL,
    MedicalEquipmentCost money  NOT NULL,
    CONSTRAINT PK_MedicalEquipmentDetails PRIMARY KEY  (MedicalEquipmentID)
);
GO

--creating table MedicalEquipmentMappingTable
CREATE TABLE MedicalEquipmentMappingTable (
    PatientRoomID int  NOT NULL, --FK
    MedicalEquipmentID int  NOT NULL, --FK
    CONSTRAINT PK_MedicalEquipmentMappingTable PRIMARY KEY  (PatientRoomID,MedicalEquipmentID)
);
GO

--creating table Billing
CREATE TABLE Billing (
    BillID int  NOT NULL IDENTITY,
    BillAmount money  NOT NULL,
    BillPayor varchar(30)  NULL,
    BillPaymentMethod varchar(20)  NULL,
    BillDate datetime  NOT NULL,
    PatientID int  NOT NULL, --FK
    CONSTRAINT PK_Billing PRIMARY KEY  (BillID)
);
GO

--creating table HealthInsuranceDetailsTable
CREATE TABLE HealthInsuranceDetailsTable (
    HealthInsuranceID int  NOT NULL IDENTITY,
    HealthInsuranceName varchar(30)  NULL,
    HealthInsuranceContactName varchar(30)  NULL,
    HealthInsurancePhone varchar(15)  NULL,
    CONSTRAINT PK_HealthInsuranceDetailsTable PRIMARY KEY  (HealthInsuranceID)
);
GO

--creating table MedicationInformation
CREATE TABLE MedicationInformation (
    MedicationID int  NOT NULL IDENTITY,
    MedicationName varchar(50)  NOT NULL,
    MedicationCost money  NOT NULL,
    CONSTRAINT PK_MedicationInformation PRIMARY KEY  (MedicationID)
);
GO

--creating table PatientMedication
CREATE TABLE PatientMedication (
    MedicationID int  NOT NULL, --FK
    PatientID int  NOT NULL,--FK
    CONSTRAINT PK_PatientMedication PRIMARY KEY  (MedicationID,PatientID)
);
GO

--creating table Diagnostics
CREATE TABLE Diagnostics (
    DiagnosticID int  NOT NULL IDENTITY,
    DiagnosticName varchar(30)  NOT NULL,
    CONSTRAINT PK_Diagnostics PRIMARY KEY  (DiagnosticID)
);
GO

--creating table PatientDiagnosticRelation
CREATE TABLE PatientDiagnosticRelation (
```

```
       PatientID int  NOT NULL, --PK/FK
    DiagnosticID int  NOT NULL,--PK/FK
    CONSTRAINT PK_PatientDiagnosticRelation PRIMARY KEY  (PatientID,DiagnosticID)
);
GO

--creating table Procedures
CREATE TABLE [Procedures] (
    ProcedureID int  NOT NULL IDENTITY,
    ProcedureName varchar(50)  NOT NULL,
    ProcedureCost money  NOT NULL,
    CONSTRAINT PK_Procedures PRIMARY KEY  (ProcedureID)
);
GO

--creating table PatientProcedures
CREATE TABLE PatientProcedures (
    PatientID int  NOT NULL,--PK/FK
    ProcedureID int  NOT NULL,--PK/FK
    CONSTRAINT PK_PatientProcedures PRIMARY KEY  (PatientID,ProcedureID)
);
GO

--creating table Symptoms
CREATE TABLE Symptoms (
     SymptomID int  NOT NULL IDENTITY,
       SymptomName varchar(30)  NOT NULL,
    CONSTRAINT PK_Symptoms PRIMARY KEY  (SymptomID)
);
GO

--creating table PatientSymptoms
CREATE TABLE PatientSymptoms (
    PatientID int  NOT NULL, --PK/FK
    SymptomID int  NOT NULL,--PK/FK
    CONSTRAINT PK_PatientSymptoms PRIMARY KEY  (PatientID,SymptomID)
);
GO

--creating table VisitorInformation
CREATE TABLE VisitorInformation (
    VisitorID int  NOT NULL IDENTITY,
    VisitorFName varchar(30)  NOT NULL,
    VisitorLName varchar(30)  NULL,
    VisitorIDType varchar(20)  NOT NULL,
    DateOfBirth date  NOT NULL,
    PictureOnRecord varchar(5)  NOT NULL,
    EntryTime datetime  NOT NULL,
    ExitTime datetime  NULL,
    PatientID int  NOT NULL, --FK
    CONSTRAINT PK_VisitorInformation PRIMARY KEY  (VisitorID)
);
GO


SET IDENTITY_INSERT Department ON

INSERT INTO Department (DepartmentID,DepartmentName,DepartmentHead) VALUES
(1, 'Cardiology','John Doe'),
(2, 'ENT','Mary Johnson'),
(3, 'Neurology','Scott Taylor'),
(4, 'Psychiatry','Karen Cruise'),
(5, 'Radiology','Steve Walker')
```

```
SET IDENTITY_INSERT Department OFF
GO

SET IDENTITY_INSERT ShiftInformation ON

INSERT INTO ShiftInformation (ShiftID,StartOfShift,EndOfShift) VALUES
(1,CAST('08:00' AS time),CAST('16:00' AS time)),
(2,CAST('16:00' AS time),CAST('00:00' AS time)),
(3,CAST('00:00' AS time),CAST('08:00' AS time))



SET IDENTITY_INSERT ShiftInformation OFF
GO



SET IDENTITY_INSERT MedicalEquipmentDetails ON
INSERT INTO MedicalEquipmentDetails(MedicalEquipmentID,MedicalEquipmentName,MedicalEquipmentCost)
VALUES
(1,'Cardiac Monitor',10000),
(2,'Ventilator',15000),
(3,'Defibrillator',7000),
(4,'Anesthesia Machine',15000),
(5,'Centrifuge',3000),
(6,'Coagulation analyzer',500),
(7,'Centrifuge Machine',2000)



SET IDENTITY_INSERT MedicalEquipmentDetails OFF
GO


SET IDENTITY_INSERT PatientRooms ON

INSERT INTO PatientRooms (PatientRoomID,Capacity,TypeOfBeds,Status,Location) VALUES
(1,4,'Electric Beds','Available','1-101'),
(2,2,'Clinitron Beds','Occupied','1-102'),
(3,1,'Gatch Beds','Available','1-103'),
(4,4,'General Beds','Available','1-104'),
(5,4,'Electric Beds','Occupied','2-201'),
(6,1,'Electric Beds','Available','2-202'),
(7,3,'Gatch Beds','Under Maintenance','2-203')



SET IDENTITY_INSERT PatientRooms OFF
GO

SET IDENTITY_INSERT Employees ON

INSERT INTO Employees (EmployeeID,EmployeeFName,EmployeeLName,SSN,ContractType,ContractTerm,
Office,OfficePhone,PastJobTitle,CurrentJobTitle,WorkSchedule,ShiftID,DepartmentID,Salary)
VALUES
(1,'John','Doe','456-62-9856','Tenured','Permanent','3-301','687-234-6738','Jr.
Doctor','Doctor','Mon-Fri',1,1,120000),
(2,'Scott','Taylor','456-11-2933','Tenured','Permanent','3-302','687-234-
1134','Doctor','Doctor','Mon-Thu',1,3,120000),
(3,'Mary','Johnson','456-23-4858','Tenured','Permanent','3-303','687-234-9115','Jr.
Doctor','Doctor','Mon-Fri',2,2,120000),
(4,'Karen','Cruise','874-86-7756','Tenured','Permanent','3-304','687-234-
1199','Doctor','Doctor','Mon-Fri',1,4,120000),
(5,'Steve','Walker','669-11-2793','Tenured','Permanent','3-305','687-234-
6738','Doctor','Doctor','Mon-Wed',1,4,120000),
```

```
(6,'Alan','Shelton','765-44-1234','Full-Time','6 years',NULL,NULL,'Nurse','Nurse','Mon-
Fri',1,3,40000),
(7,'Blake','Clifton','674-11-8563','Full-Time','6 years',NULL,NULL,'Nurse','Nurse','Mon-
Sun',2,2,40000),
(8,'Alicia','James','776-55-5674','Part-Time','2 years',NULL,NULL,'Nurse','Nurse','Mon-
Sun',3,1,40000),
(9,'Stan','Damon','675-77-9111','Full-Time','6 years','3-306','687-234-4129','Jr.
Doctor','Doctor','Mon-Wed',1,1,90000),
(10,'Audrey','Jones','76-564-7246','Full-Time','6 years','3-307','687-234-
6790','Doctor','Doctor','Mon-Sat',3,2,90000)

SET IDENTITY_INSERT Employees OFF
GO

SET IDENTITY_INSERT EmployeePersonalInformation ON
INSERT INTO
EmployeePersonalInformation(EmployeeID,EmployeeAddressLine1,EmployeeAddressLine2,EmployeeState,
EmployeeCity,EmployeeZipCode,EmployeePersonalPhone,EmployeeEmail)
 VALUES
(1,'225 Genessee st',Null,'NY','Syracuse','13299','315-678-8811','john@gmail.com'),
(2,'222 Maryland Ave',Null,'NY','Syracuse','13889','315-443-6756','scott@yahoo.com'),
(3,'1060 Lancaster st',Null,'NY','Syracuse','13110','315-567-9876','mary@gmail.com'),
(4,'1111 Lennox Ave',Null,'NY','Syracuse','13299','315-678-7654','karen@gmail.com'),
(5,'456 Westminster Drive',Null,'NY','Syracuse','13610','315-167-7656','steve@gmail.com'),
(6,'676 Roosevelt ','Gifford st','NY','Syracuse','13876','315-611-1122','alan@outlook.com'),
(7,'634 Hawthorne st',Null,'NY','Syracuse','13456','315-678-2233','blake@yahoo.com'),
(8,'765 Woodbridge Drive',Null,'NY','Syracuse','13600','315-676-5543','alicia@gmail.com'),
(9,'356 Comstock Ave',Null,'NY','Syracuse','13299','315-678-7862','stan@rediff.com'),
(10,'1040 Lancaster',Null,'NY','Syracuse','13299','315-678-2211','audrey@gmail.com')



SET IDENTITY_INSERT EmployeePersonalInformation OFF
GO
SET IDENTITY_INSERT EmployeeReviews ON

INSERT INTO EmployeeReviews(ReviewID,Review,EmployeeID) VALUES
(1,'Excellent',4),
(2,'Good',6),
(3,'Average',7),
(4,'Excellent',1),
(5,'Excellent',2),
(6,'Excellent',3),
(7,'Needs Improvement',5),
(8,'Excellent',1)

SET IDENTITY_INSERT EmployeeReviews OFF

GO


INSERT INTO MedicalEquipmentMappingTable (PatientRoomID,MedicalEquipmentID ) VALUES
(1,1),(1,3),(1,4),(1,5),(1,6),(2,1),(2,5),(3,1),(3,4),(5,2),(4,7),(6,1),(7,1),(7,2);

GO

SET IDENTITY_INSERT DoctorIDTable ON

INSERT INTO DoctorIDTable (DoctorID,EmployeeID) VALUES
(1,1),(2,2),(3,3),(4,4),(5,5),(6,9),(7,10)
```

```
SET IDENTITY_INSERT DoctorIDTable OFF
GO

SET IDENTITY_INSERT PatientRoomScheduleHistory ON

INSERT INTO PatientRoomScheduleHistory
(ScheduleID,InDateTime,OutDateTime,NameOfPatient,PatientRoomID)
VALUES
(1,'12-1-2017 10:22:10','12-7-2017 18:30:00','Victor Murphy',2),
(2,'11-29-2017 9:11:10','12-1-2017 15:21:00','Alex Jones',4)


SET IDENTITY_INSERT PatientRoomScheduleHistory OFF
GO

SET IDENTITY_INSERT Benefits ON

INSERT INTO Benefits (BenefitID,BenefitName) VALUES
(1,'Free Medication'),(2,'Paid leaves'),(3,'Free Health Insurance'),(4,'Flexible Work Schedule')

SET IDENTITY_INSERT Benefits OFF
GO

INSERT INTO EmployeeBenefits (EmployeeID,BenefitID) VALUES
(1,1),(1,2),(1,3),(2,1),(2,2),(2,4),(3,1),(3,2),(4,4),(4,2),(4,3),(5,1),(5,3),(6,1),(7,1),(8,1),
(9,1),(9,2),(10,1),(10,4)
GO




SET IDENTITY_INSERT HealthInsuranceDetailsTable ON

INSERT INTO HealthInsuranceDetailsTable
(HealthInsuranceID,HealthInsuranceName,HealthInsuranceContactName,HealthInsurancePhone)
VALUES
(1,'Aetna','Mark Tyler','654-876-1678'),
(2,'NYSOH','Amanda Green','315-776-8745'),
(3,'Assurant','Robert Jones','765-567-8967'),
(4,'ISO','Christian Rodriguez','672-876-1199'),
(5,'AmeriHealth','Dave Jacob','315-556-8735')

SET IDENTITY_INSERT HealthInsuranceDetailstable OFF
GO

SET IDENTITY_INSERT Patients ON

INSERT INTO Patients(PatientID,PatientFName,PatientLName,PatientSSN,PatientAddressLine1,
PatientAddressLine2,PatientState,PatientCity,PatientZipCode,PatientPhone,
PatientEmail,PatientSpouseName,PatientPrimaryCareDoctor,HealthInsuranceID,
HealthInsurancePlanCoverage,CheckInDateTime,CheckOutDateTime,PatientRoomID) VALUES

(1,'Marcus','Alonso','567-77-1234','120 Cambridge st','Harbor Drive','NY','Albany','12459','456-886-
7657',NULL,NULL,2,1,'Gold','11-1-2017 9:30:00',NULL,1),
(2,'Thomas','Taylor','567-33-6753','1000 Roosevelt Ave',NULL,'NY','Syracuse','13210','315-155-
6789','thomas@gmail.com',NULL,1,4,'Silver','11-15-2017 9:11:10',null,1),

(3,'Alex','Jones','276-44-2756','1212 Harvard Pl',NULL,'NY','Syracuse','13210','315-155-
0077','alex@gmail.com',NULL,3,4,'Silver','11-29-2017 9:11:10','12-1-2017 15:21:00',4),
(4,'Victor','Murphy','556-65-7733','450 Lexington st',NULL,'NY','Rochester','12455','345-118-
9990',NULL,NULL,1,3,'Platinum','12-1-2017 10:22:10','12-7-2017 18:30:00',2),
(5,'Tom','Brown','556-61-1111','1080 Trinity pl',NULL,'NY','Syracuse','12322','315-556-
4480',NULL,NULL,1,3,'Platinum','12-4-2017 10:30:00',NULL,3)
```

```
SET IDENTITY_INSERT Patients OFF
GO

SET IDENTITY_INSERT PatientHealthHistory ON

INSERT INTO
PatientHealthHistory(PatientID,PatientWeight,PatientHeight,PatientBloodPressure,PatientHeartRate,Ref
erralDoctorName,
ReferralDoctorContact,PreviousHospitalName,PreviousHospitalContact) VALUES

(1,'120 lbs','165 cm','90/60','65 bpm','Cynthia Griffo','655-219-9985','Crouse Hospital','800-567-
8800'),

(2,'165 lbs','179 cm','120/80','72 bpm','Judy Jones','315-912-5566','Fortis Hospital','711-285-
1166'),
(3,'143 lbs','171 cm','90/60','78 bpm','Neil Randall','456-571-8783','Jupiter Hospital','811-555-
9999'),
(4,'180 lbs','160 cm','140/90','74 bpm','Neil Randall','456-571-8783','Jupiter Hospital','811-555-
9999'),
(5,'167 lbs','156 cm','120/80','63 bpm','Henry Thomas','444-786-6546','Health Care Hospital','855-
776-1186')



SET IDENTITY_INSERT PatientHealthHistory OFF
GO

SET IDENTITY_INSERT VisitorInformation ON

INSERT INTO VisitorInformation(VisitorID,VisitorFName,VisitorLName,VisitorIDType,DateOfBirth,
PictureOnRecord,EntryTime,ExitTime,PatientID) VALUES
(1,'Edward',NULL,'Driving License','07-21-1980','Yes','11-2-2017 11:00:00','11-2-2017 14:00:00',1),
(2,'Luis','Suarez','Driving License','05-09-1978','NO','12-4-2017 11:00:00','12-4-2017 13:00:00',1),
(3,'Mathew','Joseph','Passport','02-02-1989','Yes','11-17-2017 12:00:00','11-18-2017 09:00:00',2),

(4,'David','Moses','Driving License','05-08-1990','Yes','12-2-2017 11:00:00','12-2-2017 13:00:00',4)


SET IDENTITY_INSERT VisitorInformation OFF
GO

SET IDENTITY_INSERT Diagnostics ON

INSERT INTO Diagnostics(DiagnosticID,DiagnosticName) VALUES

(1,'Cardiovascular disease'),(2,'Heart Failure'),(3,'Vertigo'),(4,'Sinusitis'),
(5,'Epilepsy'),(6,'Dementia'),(7,'Mental Disorder'),(8,'Viral Fever')

SET IDENTITY_INSERT Diagnostics OFF
GO

INSERT INTO PatientDiagnosticRelation(PatientID,DiagnosticID) VALUES
(1,1),(1,2),(2,7),(3,1),(4,4),(4,3),(4,8),(5,1)
GO

SET IDENTITY_INSERT Billing ON
INSERT INTO Billing(BillID,BillAmount,BillPayor,BillPaymentMethod,BillDate,PatientID) VALUES
(1,22000,'Insurance','Wire Transfer','12-1-2017',3),
(2,33000,'Self','Check','12-7-2017',4)


SET IDENTITY_INSERT Billing OFF
GO

SET IDENTITY_INSERT Symptoms ON
```

```
INSERT INTO Symptoms(SymptomID,SymptomName) VALUES
(1,'Chest Pain'),(2,'Fatigue'),(3,'Abdominal Pain'),(4,'Depression'),(5,'Cough'),(6,'Headache')

SET IDENTITY_INSERT Symptoms OFF
GO

INSERT INTO PatientSymptoms(PatientID,SymptomID) VALUES
(1,1),(1,2),(2,4),(3,1),(4,3),(4,6),(5,1),(5,2)
GO

SET IDENTITY_INSERT MedicationInformation ON

INSERT INTO MedicationInformation(MedicationID,MedicationName,MedicationCost) VALUES
(1,'Atorvastatin Calcium',100),(2,'Lisinopril',75),(3,'Augmentin',75),(4,'Diamorphine',120),
(5,'Warfarin',200)

SET IDENTITY_INSERT MedicationInformation OFf
GO


INSERT INTO PatientMedication(MedicationID,PatientID) VALUES
(1,3),(1,2),(1,5),(2,3),(3,3),(3,4),(4,1),(5,1),(5,5)
GO

SET IDENTITY_INSERT [Procedures] ON

INSERT INTO [Procedures](ProcedureID,ProcedureName,ProcedureCost) VALUES
(1,'Angioplasty',33000),(2,'Heart Valve Surgery',30000),(3,'ENDOSCOPY',12000),(4,'X-Ray',400),
(5,'ECG',4000),(6,'MRI',3000)


SET IDENTITY_INSERT [Procedures] OFF
GO

INSERT INTO PatientProcedures(PatientID,ProcedureID) VALUES
(1,2),(1,6),(2,4),(3,1),(3,5),(4,3),(5,2),(5,5)
GO


-- adding foreign keys

ALTER TABLE Employees ADD CONSTRAINT FK_Employees_Department
    FOREIGN KEY (DepartmentID)
    REFERENCES Department (DepartmentID);

ALTER TABLE Employees ADD CONSTRAINT FK_Employees_ShiftInformation
    FOREIGN KEY (ShiftID)
    REFERENCES ShiftInformation (ShiftID);


ALTER TABLE Billing ADD CONSTRAINT FK_Billing_Patients
    FOREIGN KEY (PatientID)
    REFERENCES Patients (PatientID);

       ALTER TABLE Patients ADD CONSTRAINT FK_Patients_DoctorIDTable
    FOREIGN KEY (PatientPrimaryCareDoctor)
    REFERENCES DoctorIDTable (DoctorID);

ALTER TABLE Patients ADD CONSTRAINT FK_Patients_HealthInsuranceDetailsTable
    FOREIGN KEY (HealthInsuranceID)
    REFERENCES HealthInsuranceDetailsTable (HealthInsuranceID);
```

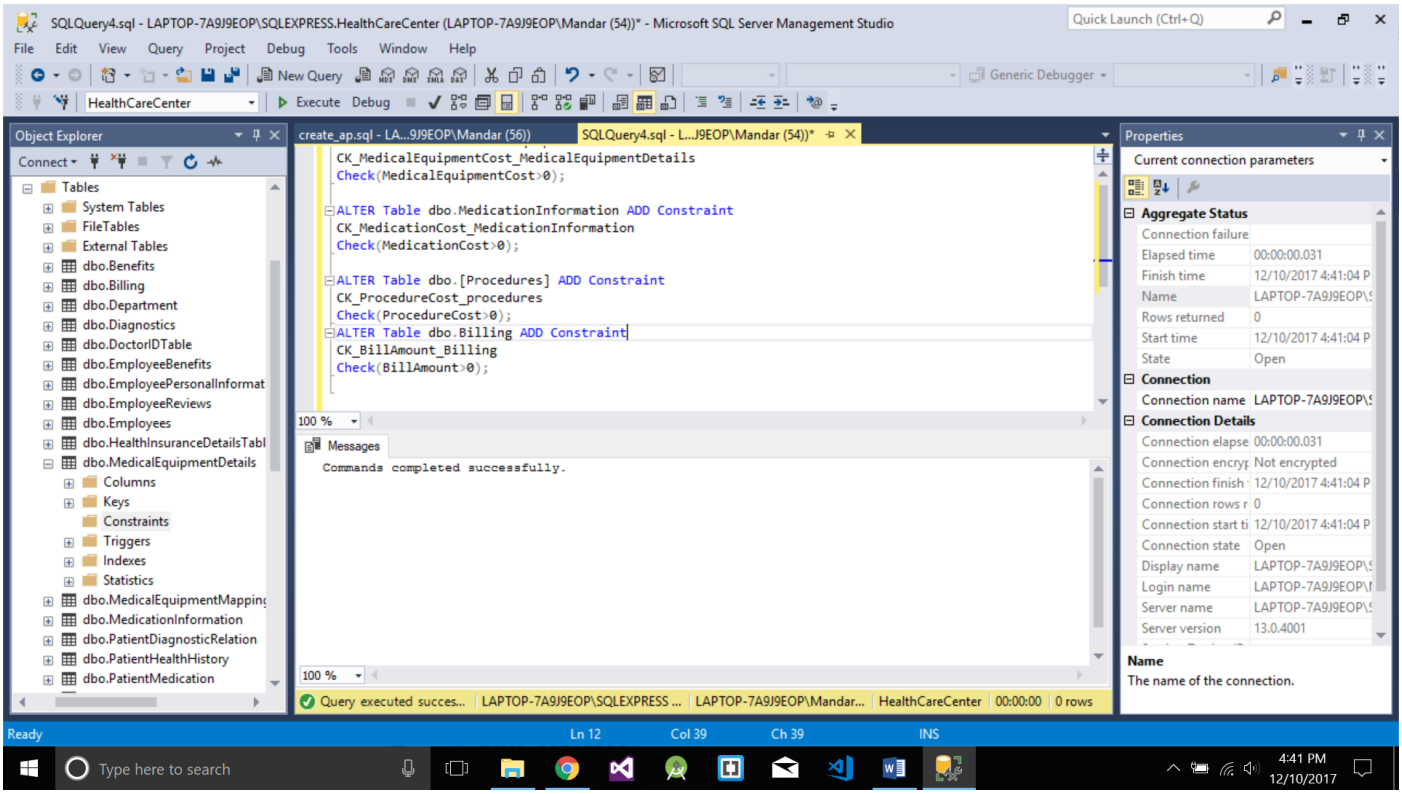
```
ALTER TABLE Patients ADD CONSTRAINT FK_Patients_PatientRooms
    FOREIGN KEY (PatientRoomID)
    REFERENCES PatientRooms (PatientRoomID);


ALTER TABLE DoctorIDTable ADD CONSTRAINT FK_DoctorIDTable_Employees
    FOREIGN KEY (EmployeeID)
    REFERENCES Employees (EmployeeID);

ALTER TABLE EmployeeBenefits ADD CONSTRAINT FK_EmployeeBenefits_Benefits
    FOREIGN KEY (BenefitID)
    REFERENCES Benefits (BenefitID);

ALTER TABLE EmployeeBenefits ADD CONSTRAINT FK_EmployeeBenefits_Employees
    FOREIGN KEY (EmployeeID)
    REFERENCES Employees (EmployeeID);

ALTER TABLE EmployeePersonalInformation ADD CONSTRAINT FK_EmployeePersonalInformation_Employees
    FOREIGN KEY (EmployeeID)
    REFERENCES Employees (EmployeeID);

ALTER TABLE EmployeeReviews ADD CONSTRAINT FK_EmployeeReviews_Employees
    FOREIGN KEY (EmployeeID)
    REFERENCES Employees (EmployeeID);



ALTER TABLE MedicalEquipmentMappingTable ADD CONSTRAINT
FK_MedicalEquipmentMappingTable_MedicalEquipmentDetails
    FOREIGN KEY (MedicalEquipmentID)
    REFERENCES MedicalEquipmentDetails (MedicalEquipmentID);

ALTER TABLE MedicalEquipmentMappingTable ADD CONSTRAINT FK_MedicalEquipmentMappingTable_PatientRooms
    FOREIGN KEY (PatientRoomID)
    REFERENCES PatientRooms (PatientRoomID);

ALTER TABLE PatientDiagnosticRelation ADD CONSTRAINT FK_PatientDiagnosticRelation_Diagnostics
    FOREIGN KEY (DiagnosticID)
    REFERENCES Diagnostics (DiagnosticID);

ALTER TABLE PatientDiagnosticRelation ADD CONSTRAINT FK_PatientDiagnosticRelation_Patients
    FOREIGN KEY (PatientID)
    REFERENCES Patients (PatientID);

ALTER TABLE PatientHealthHistory ADD CONSTRAINT FK_PatientHealthHistory_Patients
    FOREIGN KEY (PatientID)
    REFERENCES Patients (PatientID);

ALTER TABLE PatientMedication ADD CONSTRAINT FK_PatientMedicationRelation_MedicationInformation
    FOREIGN KEY (MedicationID)
    REFERENCES MedicationInformation (MedicationID);

ALTER TABLE PatientMedication ADD CONSTRAINT FK_PatientMedicationRelation_Patients
    FOREIGN KEY (PatientID)
    REFERENCES Patients (PatientID);

ALTER TABLE PatientProcedures ADD CONSTRAINT FK_PatientProcedures_Patients
    FOREIGN KEY (PatientID)
    REFERENCES Patients (PatientID);

ALTER TABLE PatientProcedures ADD CONSTRAINT FK_PatientProcedures_Procedures
    FOREIGN KEY (ProcedureID)
    REFERENCES [Procedures] (ProcedureID);
```

```
ALTER TABLE PatientRoomScheduleHistory ADD CONSTRAINT FK_PatientRoomScheduleHistory_PatientRooms
    FOREIGN KEY (PatientRoomID)
    REFERENCES PatientRooms (PatientRoomID);

ALTER TABLE PatientSymptoms ADD CONSTRAINT FK_PatientSymptoms_Patients
    FOREIGN KEY (PatientID)
    REFERENCES Patients (PatientID);

ALTER TABLE PatientSymptoms ADD CONSTRAINT FK_PatientSymptoms_Symptoms
    FOREIGN KEY (SymptomID)
    REFERENCES Symptoms (SymptomID);



ALTER TABLE VisitorInformation ADD CONSTRAINT FK_VisitorInformation_Patients
    FOREIGN KEY (PatientID)
    REFERENCES Patients (PatientID);

       -- checking whether each foreign key is created or not
       --select * from sys.foreign_keys

CREATE NONCLUSTERED INDEX IX_Billing_BillDate ON Billing(
       BillDate ASC
)
GO

CREATE NONCLUSTERED INDEX IX_Patients_CheckInDateTime ON Patients(
       CheckInDateTime DESC
)
GO

CREATE NONCLUSTERED INDEX IX_Employees_WorkSchedule ON Employees(
       WorkSchedule ASC
)
GO

CREATE NONCLUSTERED INDEX IX_Patients_PatientRoomID ON Patients(
       PatientRoomID ASC
)
GO
```

CHECK CONSTRAINTS:

The following are the check constraints that are put on the tables in the database.

```
ALTER Table dbo.MedicalEquipmentDetails ADD Constraint
CK_MedicalEquipmentCost_MedicalEquipmentDetails
Check(MedicalEquipmentCost>0);

ALTER Table dbo.MedicationInformation ADD Constraint
CK_MedicationCost_MedicationInformation
Check(MedicationCost>0);

ALTER Table dbo.[Procedures] ADD Constraint
CK_ProcedureCost_procedures
Check(ProcedureCost>0);
ALTER Table dbo.Billing ADD Constraint
CK_BillAmount_Billing
Check(BillAmount>0);
```



INDEXES

Added 4 indexes on columns that I believe would be use for frequently in joins and search conditions. Adding these indexes would greatly improve the performance of the SQL queries mostly in searches.

1. `IX_Billing_BillDate`
   This index is added on the BillDate column of the billing table. I think that we would frequently want to go through the bills in the health care system according to the date. So, an index on this column would improve the performance.

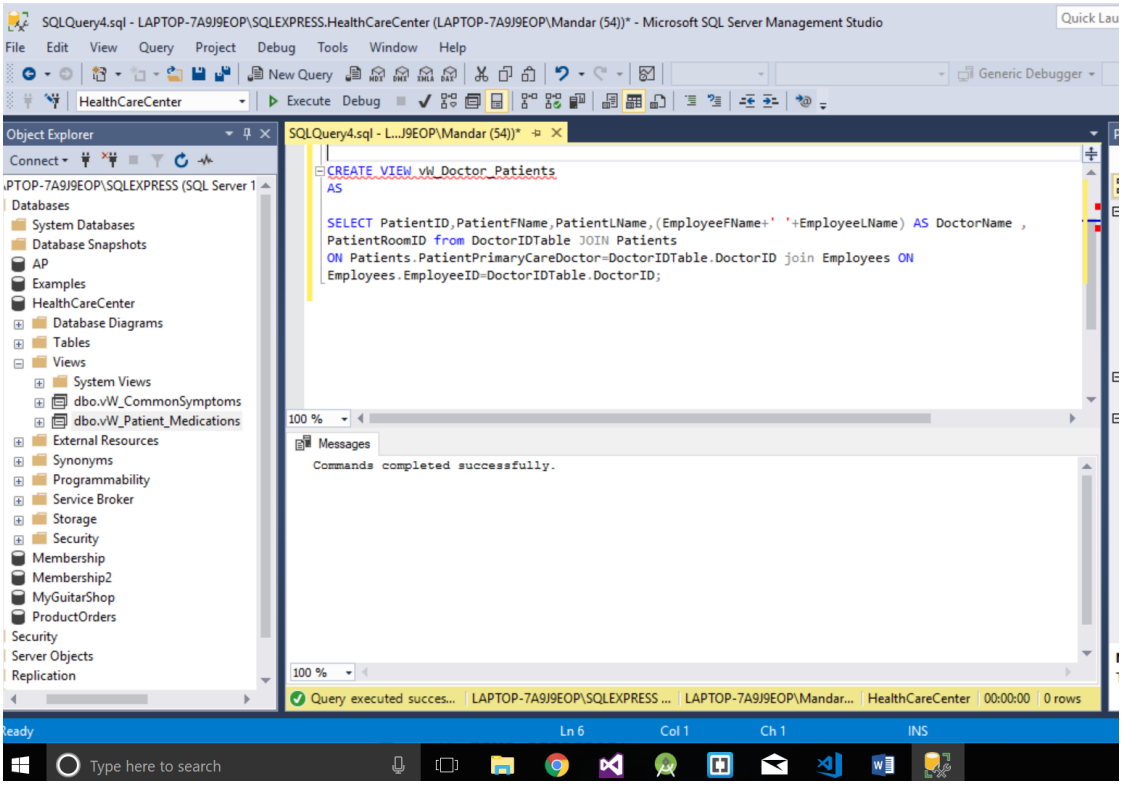
2. `IX_Patients_CheckInDateTime`
   This index is added on the CheckInDateTime column of the Patients table. I think that we would frequently want the information of a Patient in the health care system according to his/her check in time. So, an index on this column would be helpful.

3. `IX_Employees_WorkSchedule`
   This index is added on the WorkSchedule column of the Employees table. We would need information about who is currently working and who is working when on a daily basis in our health care system. Hence index on this column is a good way to improve performance.

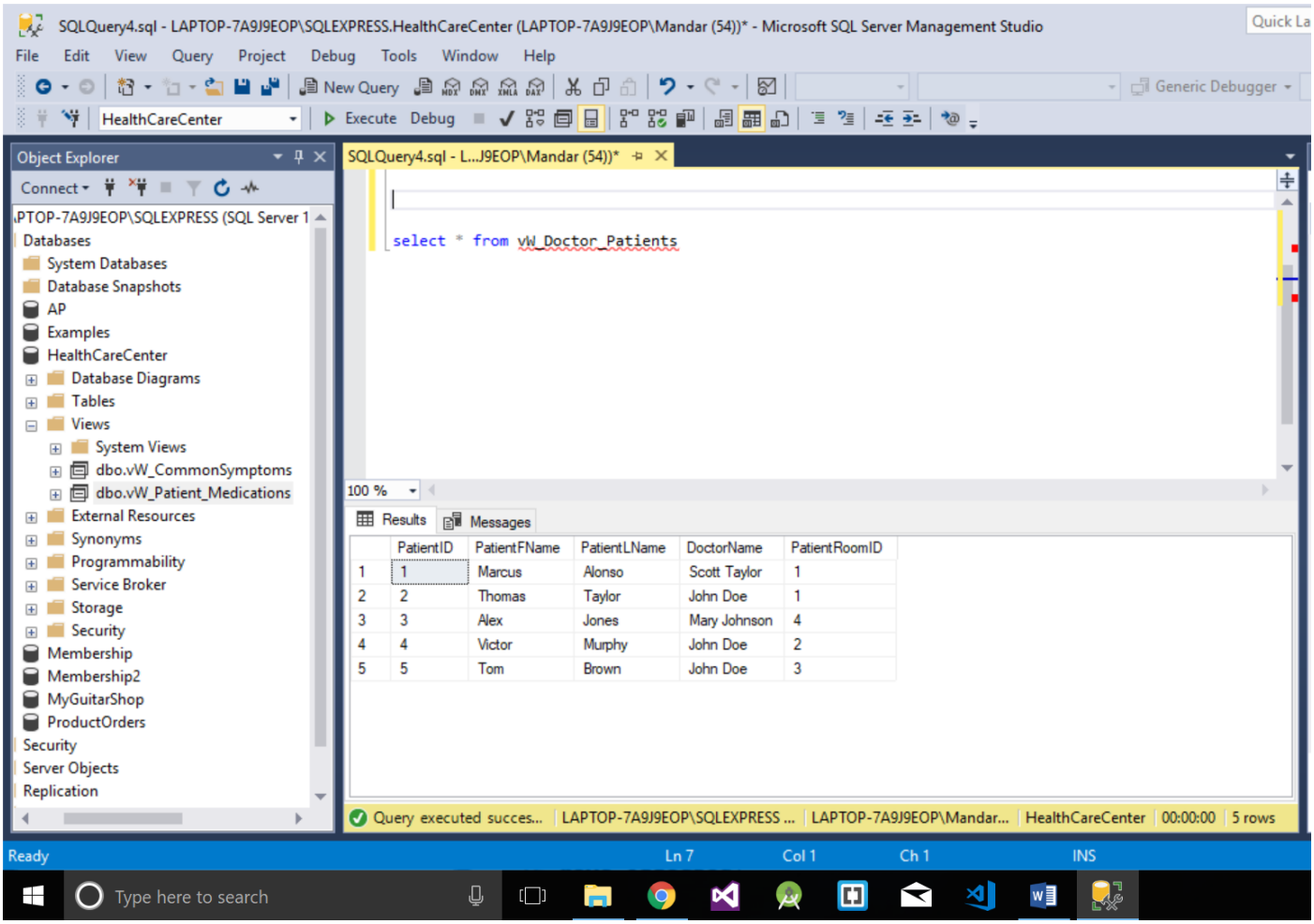
4. `IX_Patients_PatientRoomID`
   This index is added on the PatientRoomID foreign key column of the Patients table. This column would occur in almost all the searches when we want to find out where a particular patient is located hence the index.

VIEWS

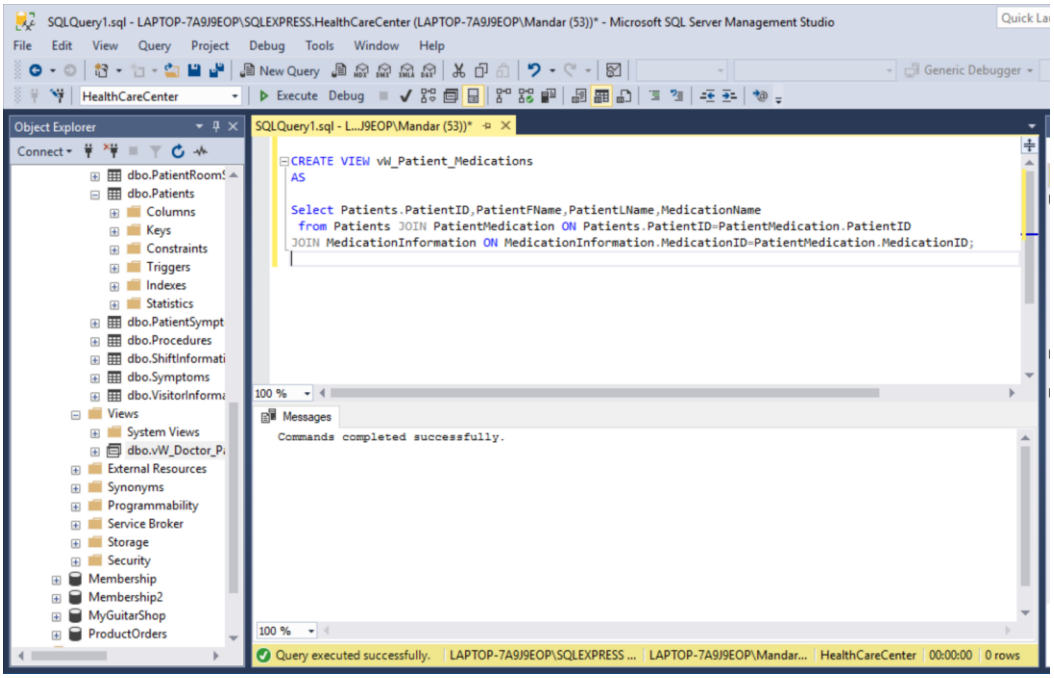
1. Created a view that gives information about all the patients and their associated doctors in the database. This information would be certainly needed on a daily basis hence I think creating a view with this information would be the best practice.



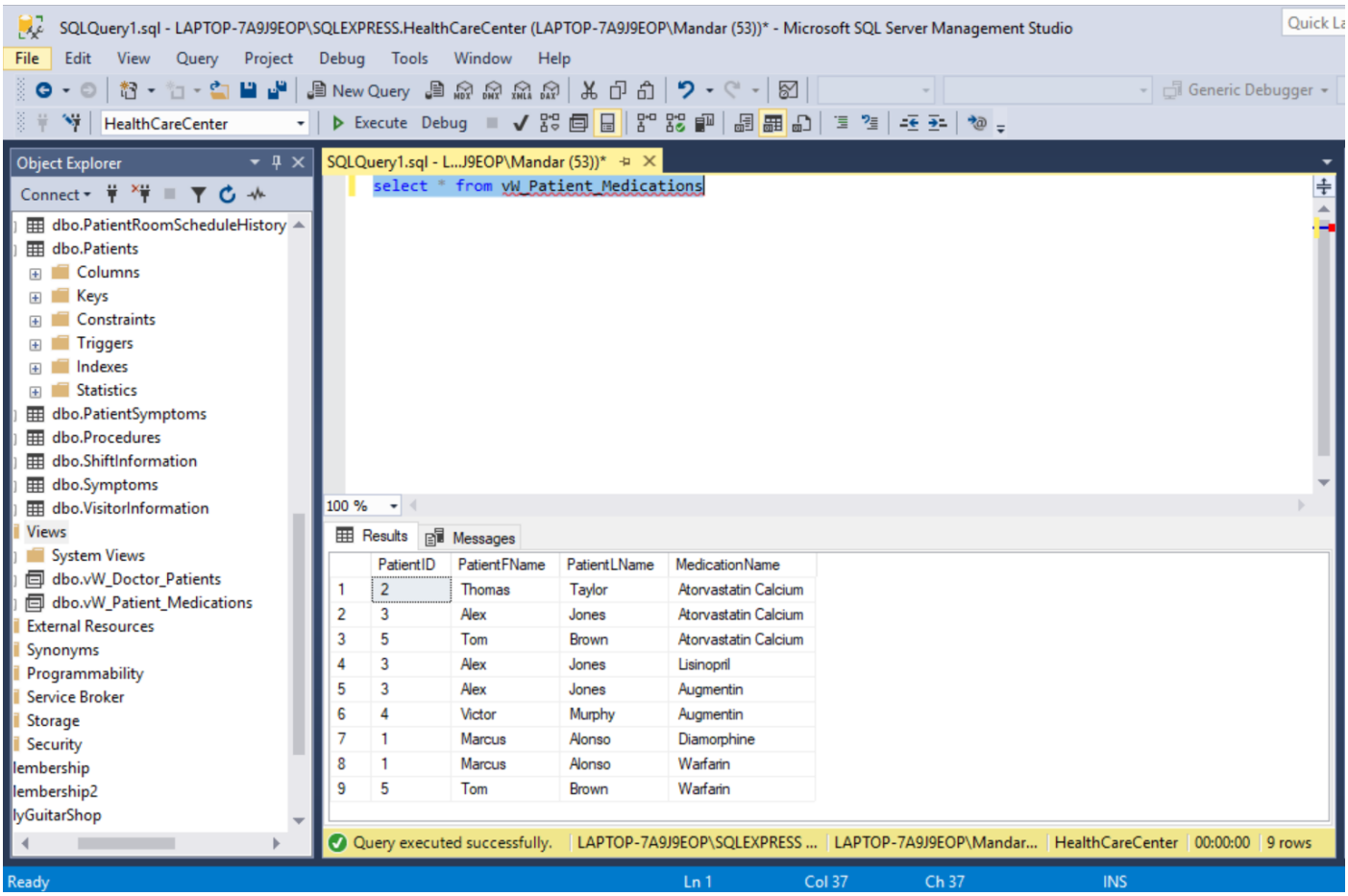

```
CREATE VIEW vW_Doctor_Patients
AS

SELECT PatientID,PatientFName,PatientLName,(EmployeeFName+' '+EmployeeLName) AS DoctorName ,
PatientRoomID from DoctorIDTable JOIN Patients
ON Patients.PatientPrimaryCareDoctor=DoctorIDTable.DoctorID join Employees ON
Employees.EmployeeID=DoctorIDTable.DoctorID;
```



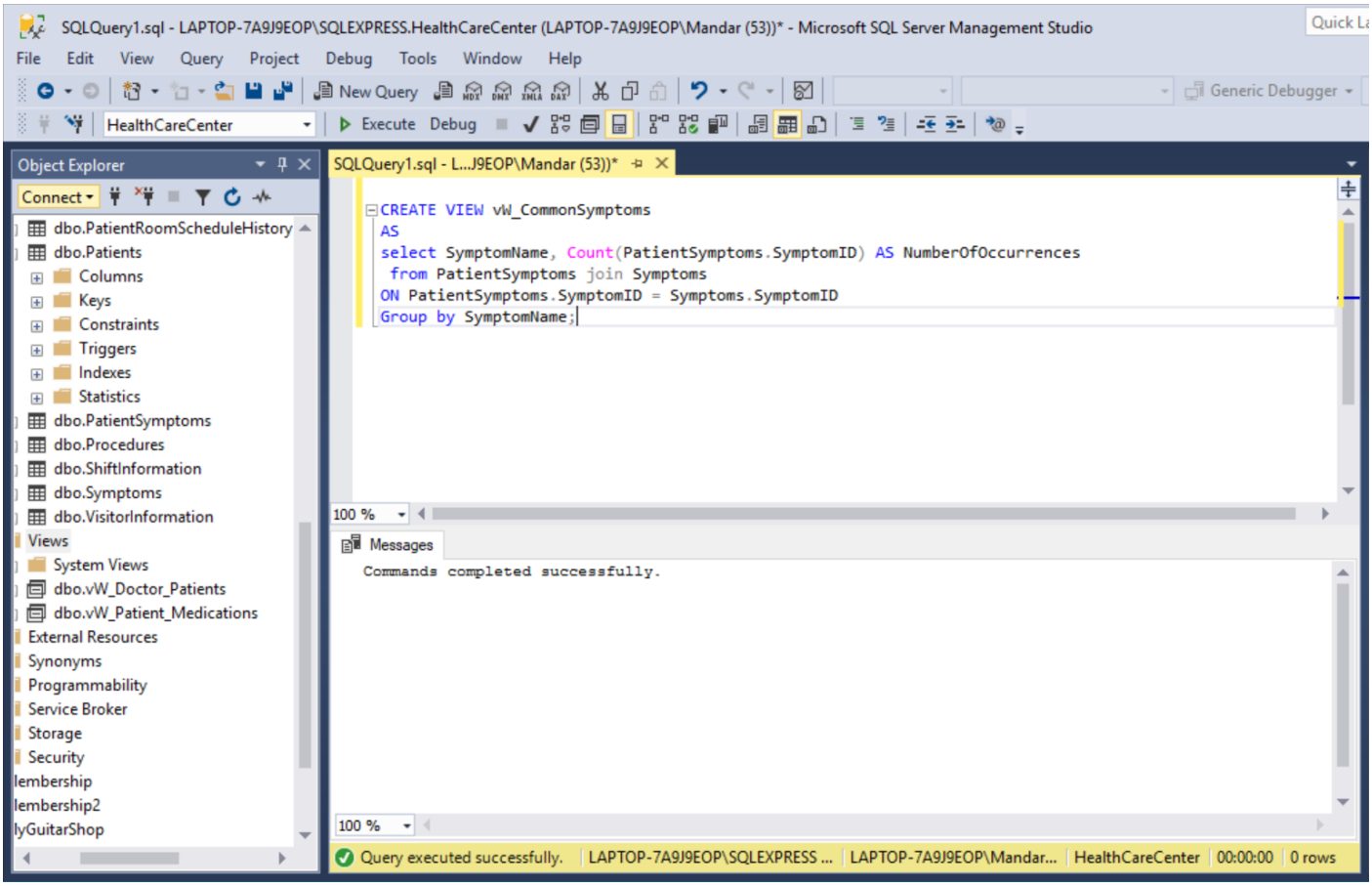
2. Created a view that gives a list of medications prescribed to the patient. This information also would be needed frequently so that the patient and his/her relatives would be up to date with the medications.



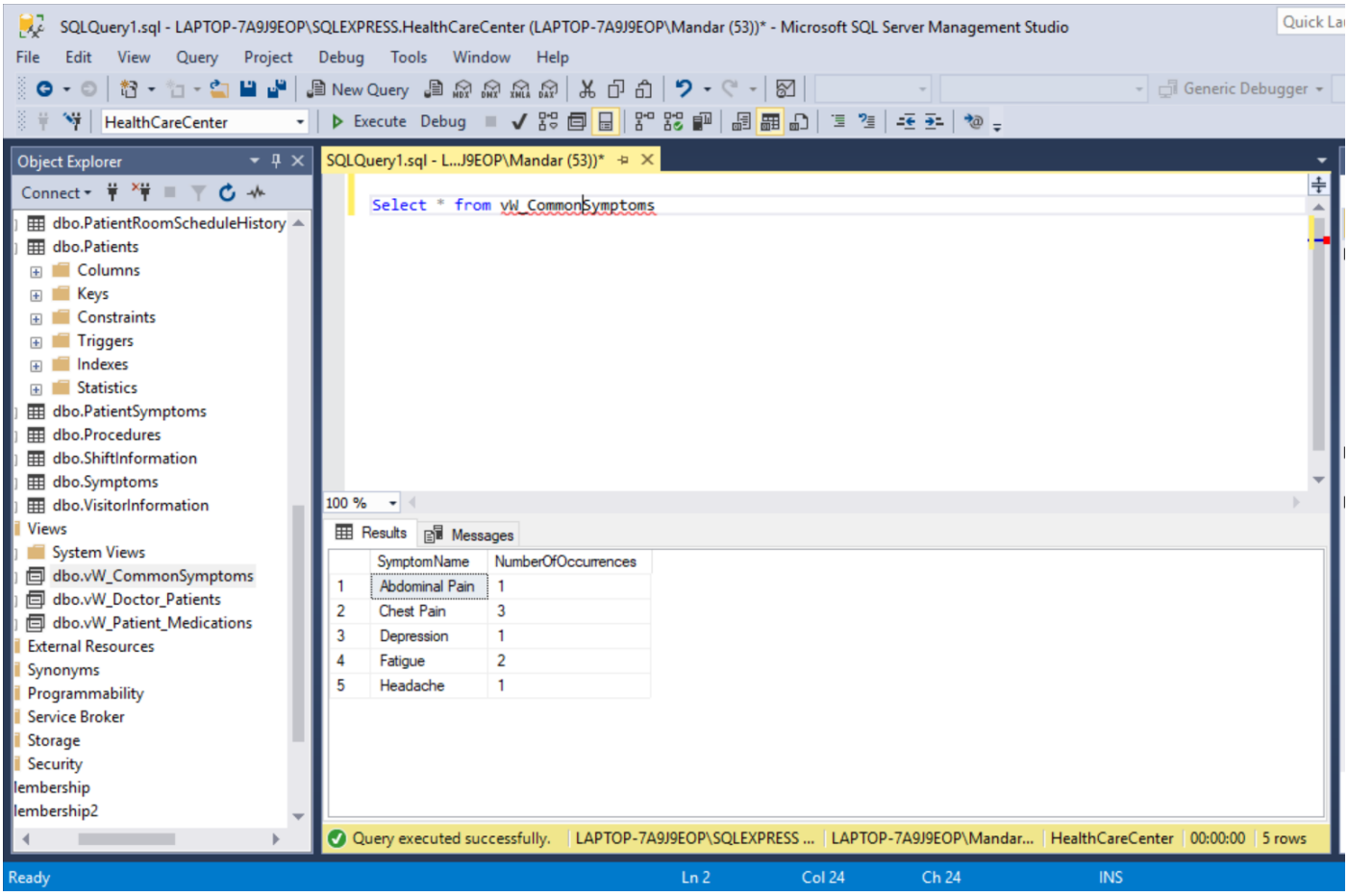
```
CREATE VIEW vW_Patient_Medications
AS

Select Patients.PatientID,PatientFName,PatientLName,MedicationName
 from Patients JOIN PatientMedication ON Patients.PatientID=PatientMedication.PatientID
JOIN MedicationInformation ON MedicationInformation.MedicationID=PatientMedication.MedicationID;
```



3. Created a view that provides information about the most commonly observed symptoms in the patients that were admitted to the health center database. This information is very crucial as it can help to analyze the occurrences of certain diseases and can help to establish preventive measures to avoid an epidemic.



```
CREATE VIEW vW_CommonSymptoms
AS
select SymptomName, Count(PatientSymptoms.SymptomID) AS NumberOfOccurrences
 from PatientSymptoms join Symptoms
ON PatientSymptoms.SymptomID = Symptoms.SymptomID
Group by SymptomName;
```

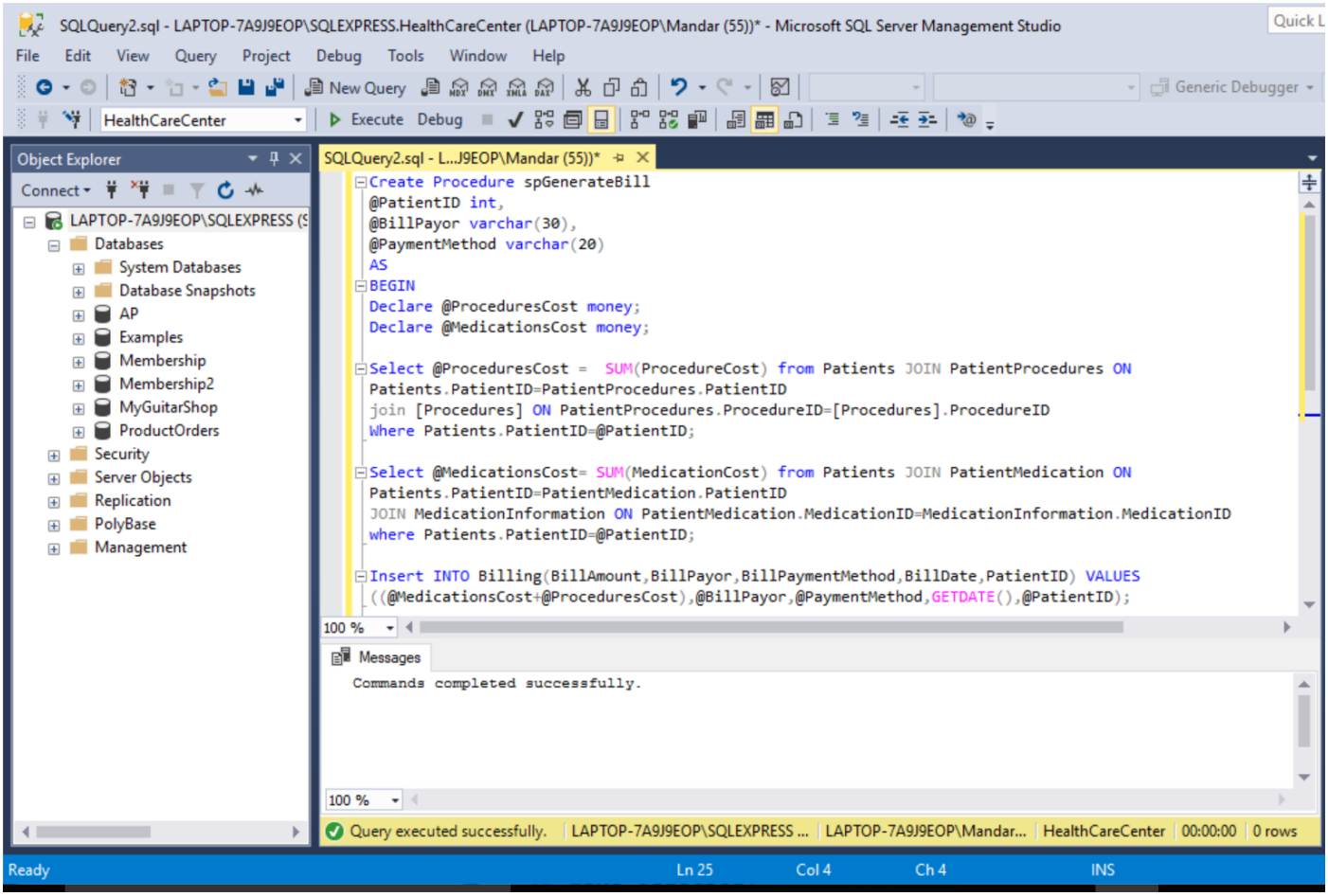


STORED PROCEDURES

1. According to my design a bill is generated with the corresponding PatientID whenever a Patient is discharged from the health centre. So, it would be very convenient if I have a stored procedure to just take the PatientID, BillPayor and PaymentMethod as the parameters and generate a bill for that patient. This stored procedure would also insert the bill generated into the BILLING table and update the CheckOutDateTime in the Patients Table from null to a date time value.

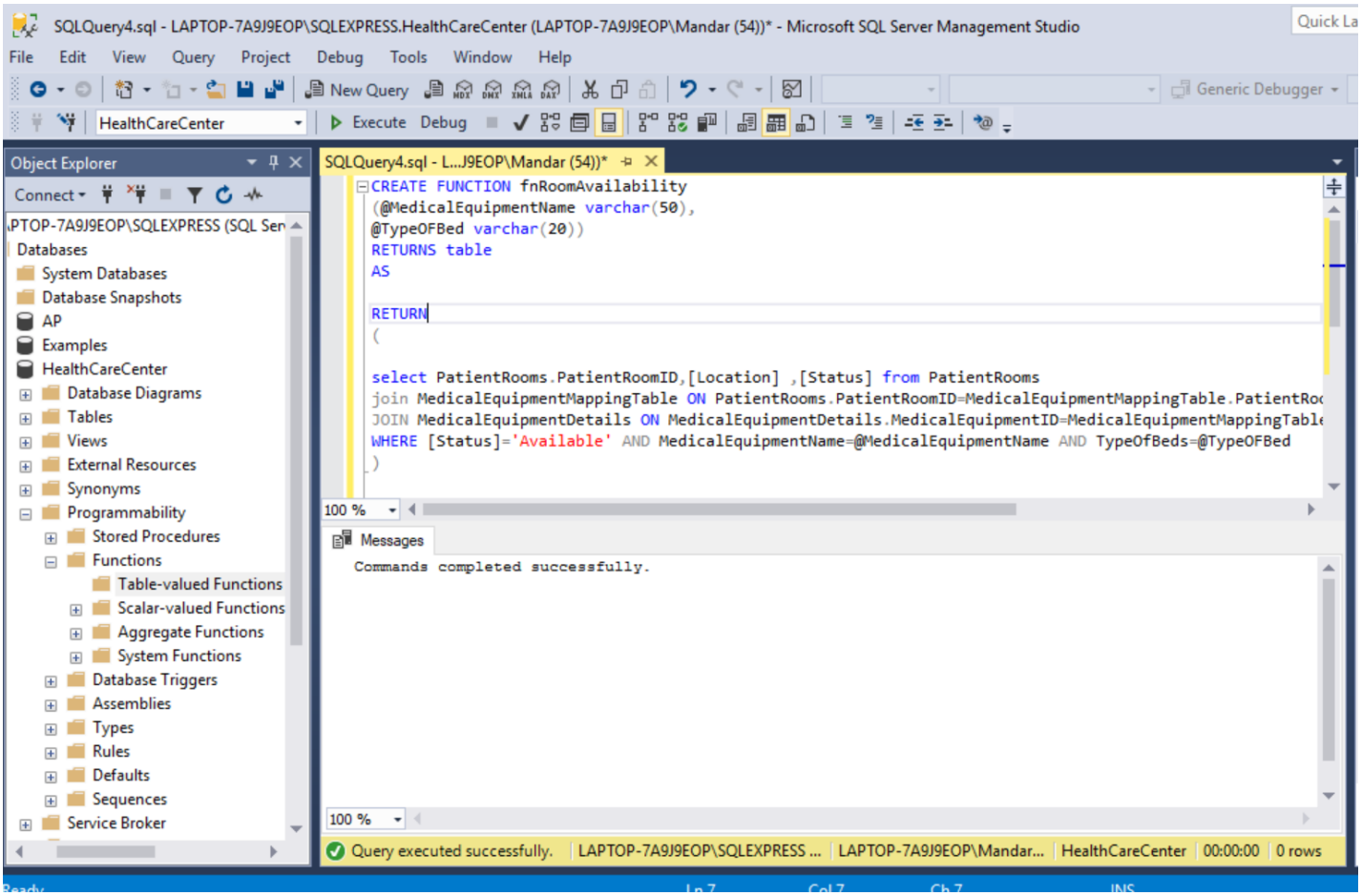
```
Create Procedure spGenerateBill
@PatientID int,
@BillPayor varchar(30),
@PaymentMethod varchar(20)
AS
BEGIN
Declare @ProceduresCost money;
Declare @MedicationsCost money;

Select @ProceduresCost =  SUM(ProcedureCost) from Patients JOIN PatientProcedures ON
Patients.PatientID=PatientProcedures.PatientID
join [Procedures] ON PatientProcedures.ProcedureID=[Procedures].ProcedureID
Where Patients.PatientID=@PatientID;
```

```
Select @MedicationsCost= SUM(MedicationCost) from Patients JOIN PatientMedication ON
Patients.PatientID=PatientMedication.PatientID
JOIN MedicationInformation ON PatientMedication.MedicationID=MedicationInformation.MedicationID
where Patients.PatientID=@PatientID;

Insert INTO Billing(BillAmount,BillPayor,BillPaymentMethod,BillDate,PatientID) VALUES
((@MedicationsCost+@ProceduresCost),@BillPayor,@PaymentMethod,GETDATE(),@PatientID);

Update Patients Set CheckOutDateTime=GETDATE() Where PatientID=@PatientID;

END
```



FUNCTIONS

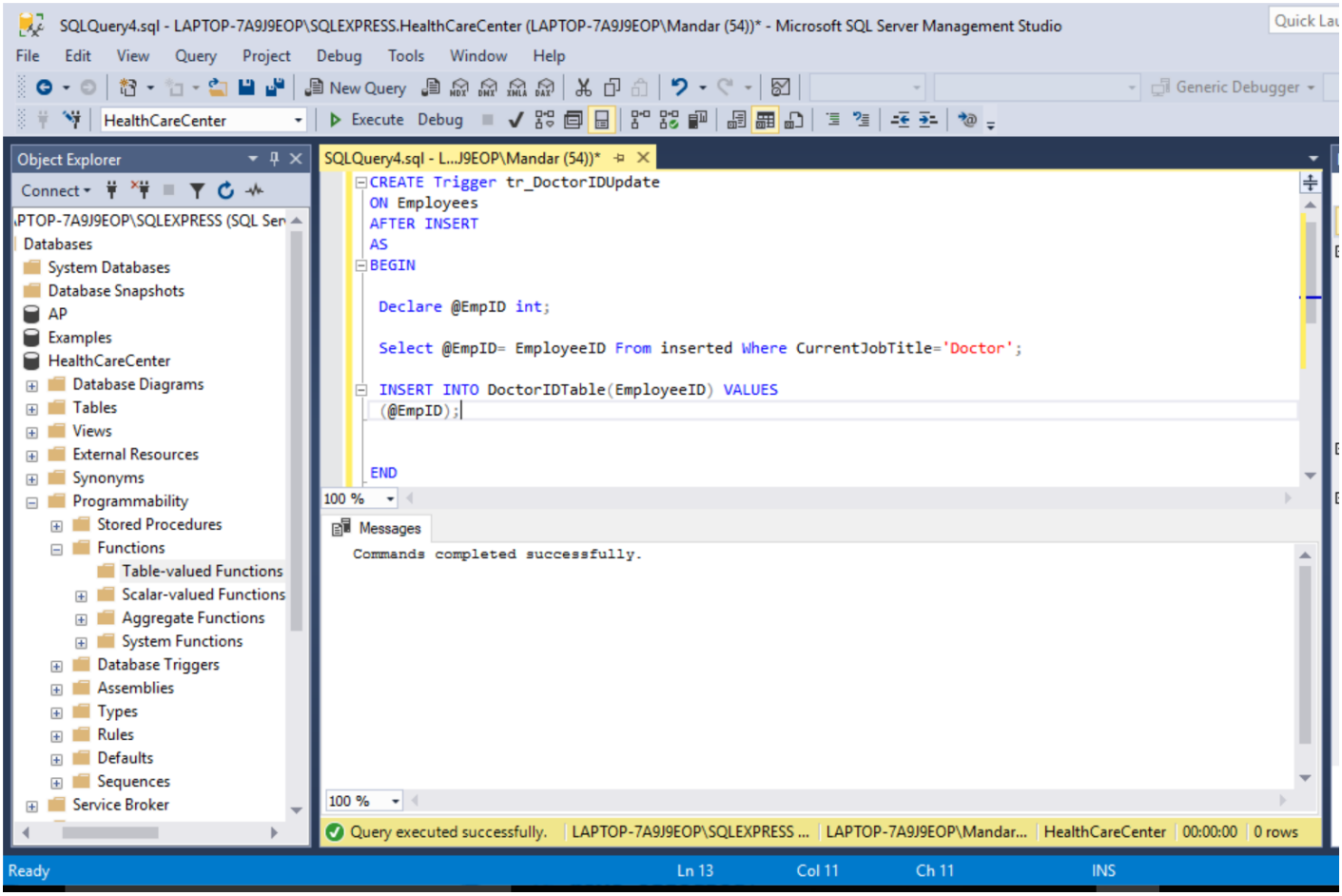
1. Created a function the gives a list of next available rooms that are not occupied and have the required medical equipment. This scenario would be helpful as it would provide the room availability if a new patient is going to be admitted in the health care centre.




```
CREATE FUNCTION fnRoomAvailability
(@MedicalEquipmentName varchar(50),
@TypeOFBed varchar(20))
RETURNS table
AS

RETURN
(

select PatientRooms.PatientRoomID,[Location] ,[Status] from PatientRooms
join MedicalEquipmentMappingTable ON
PatientRooms.PatientRoomID=MedicalEquipmentMappingTable.PatientRoomID
JOIN MedicalEquipmentDetails ON
MedicalEquipmentDetails.MedicalEquipmentID=MedicalEquipmentMappingTable.MedicalEquipmentID
WHERE [Status]='Available' AND MedicalEquipmentName=@MedicalEquipmentName AND TypeOfBeds=@TypeOFBed
)
```

TRIGGERS

1. According to my design I have a DoctorID table and and Employees table. The DoctorID table consists of all the Doctors in the health care Center. Therefore, each doctor has an unique EmployeeID as well as a DoctorID. Hence I have added a trigger that automatically inserts a row into DoctorID table if a new doctor is added into the employees table.



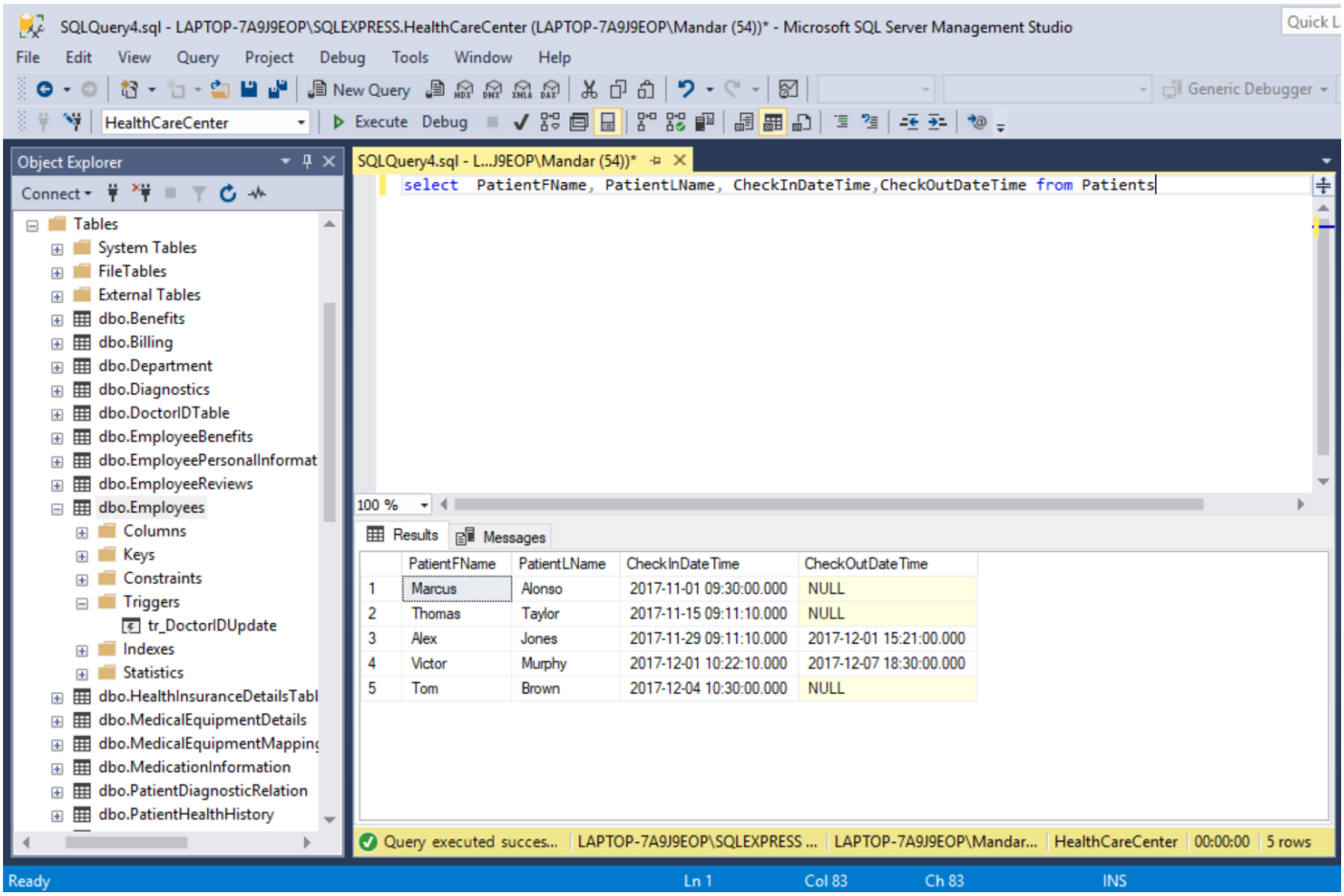

```
CREATE Trigger tr_DoctorIDUpdate
ON Employees
AFTER INSERT
AS
BEGIN

 Declare @EmpID int;

 Select @EmpID= EmployeeID From inserted Where CurrentJobTitle='Doctor';

 INSERT INTO DoctorIDTable(EmployeeID) VALUES
 (@EmpID);


END
```

TESTING

Scenario 1:

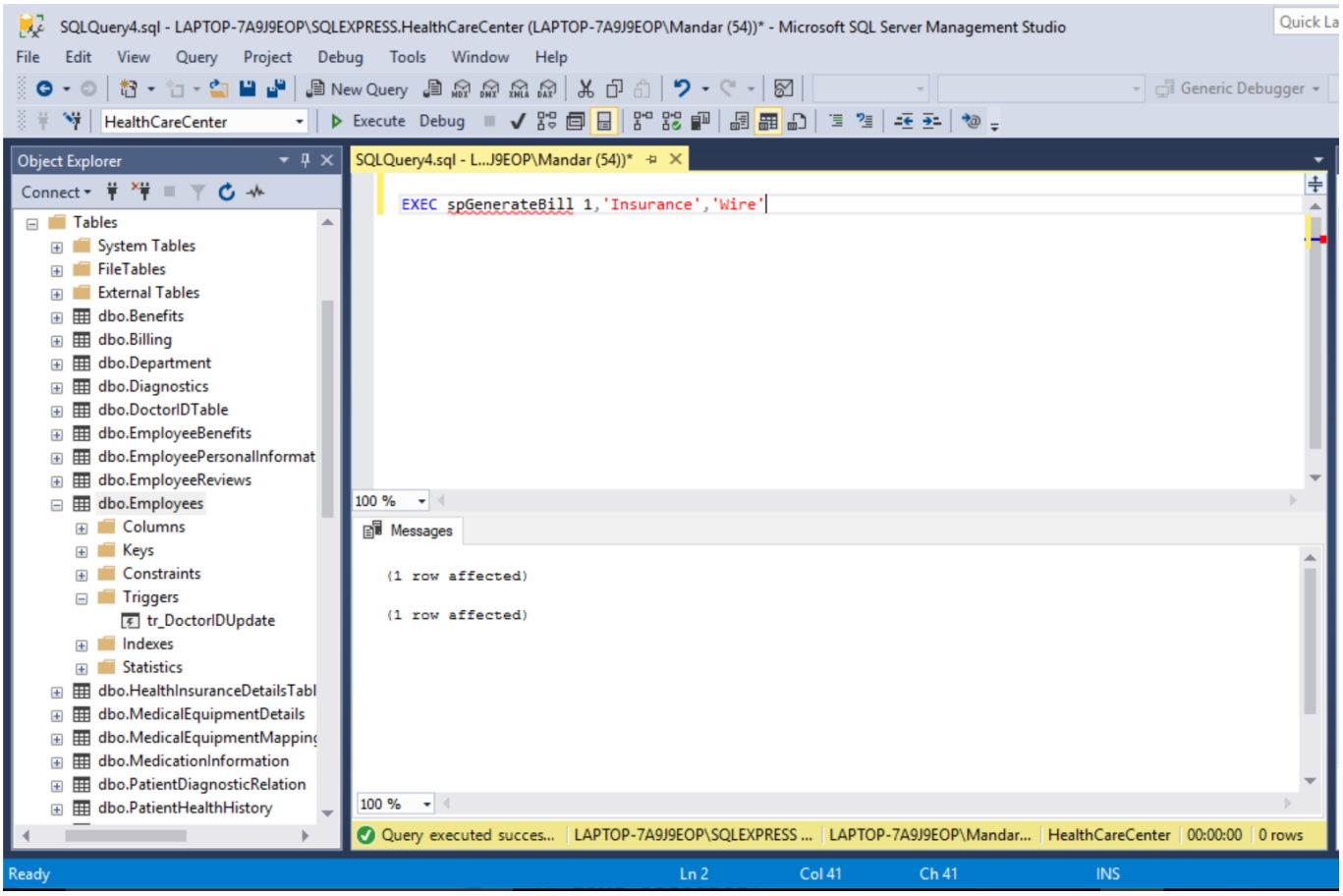
Generating a bill for a customer automatically when the customer is about to discharge. We can use a stored procedure that we have created – spGenerateBill to achieve this functionality.

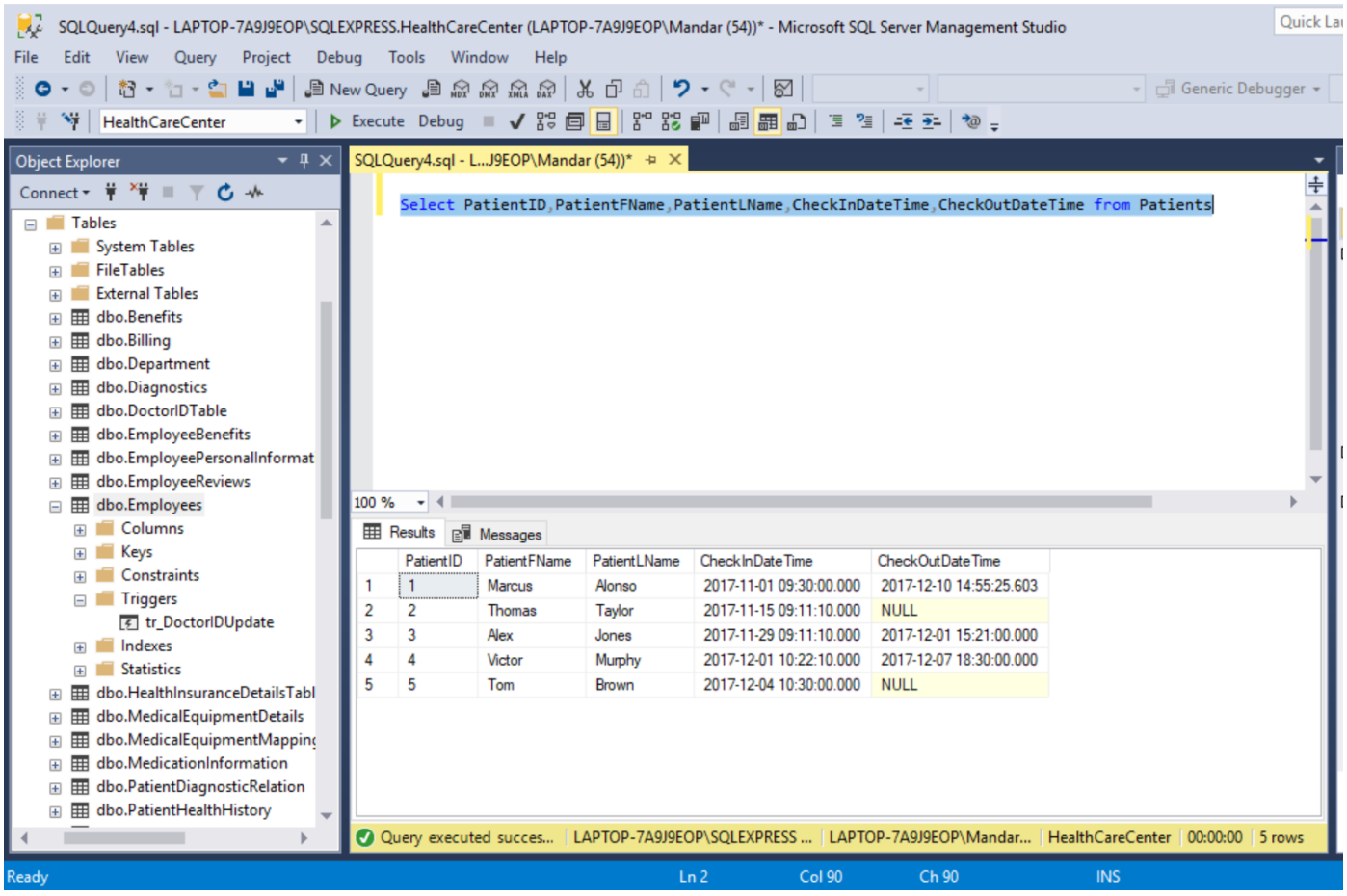
Before execution of the stored procedure :



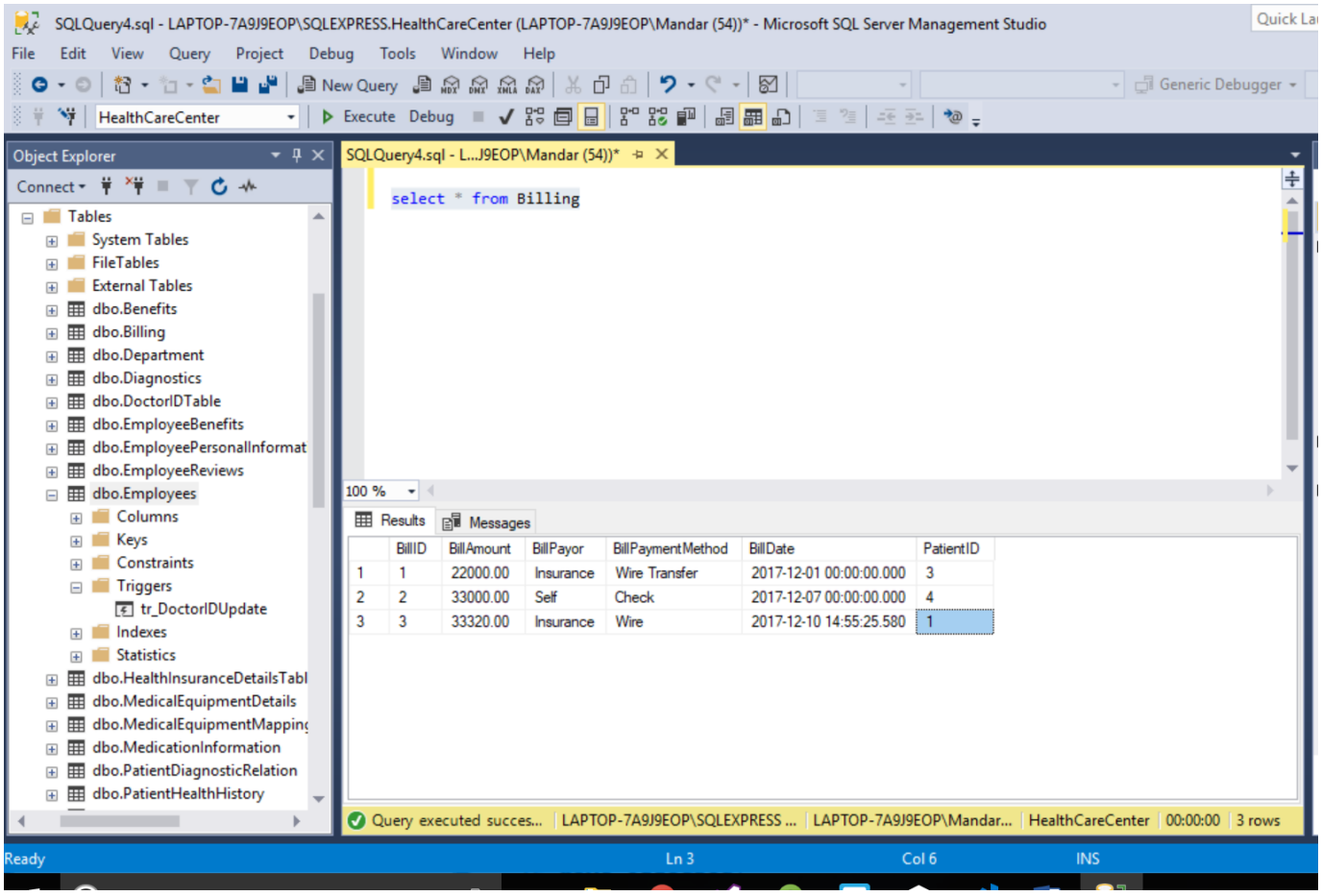
This is the patients table that we have currently on record. Here the CheckOutDateTime =null simply signifies that the patient is currently under treatment and not discharged yet.

Suppose we want to discharge the patient ‘Marcus Alonso’ so we need to generate the bill and put a value in the CheckOutDateTime column as todays date and time.


SQLQuery4.sql - LAPTOP-7A9J9EOP\SQLEXPRESS.HealthCareCenter (LAPTOP-7A9J9EOP\Mandar (54))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
Generic Debugger
HealthCareCenter
Execute Debug
Object Explorer
Connect
Tables
System Tables
FileTables
External Tables
dbo.Benefits
dbo.Billing
dbo.Department
dbo.Diagnostics
dbo.DoctorIDTable
dbo.EmployeeBenefits
dbo.EmployeePersonalInformat
dbo.EmployeeReviews
dbo.Employees
Columns
Keys
Constraints
Triggers
tr_DoctorIDUpdate
Indexes
Statistics
dbo.HealthInsuranceDetailsTabl
dbo.MedicalEquipmentDetails
dbo.MedicalEquipmentMapping
dbo.MedicationInformation
dbo.PatientDiagnosticRelation
dbo.PatientHealthHistory
SQLQuery4.sql - L...J9EOP\Mandar (54))*
EXEC spGenerateBill 1,'Insurance','Wire'
100 %
Messages
(1 row affected)
(1 row affected)
100 %
Query executed succes...
LAPTOP-7A9J9EOP\SQLEXPRESS ...
LAPTOP-7A9J9EOP\Mandar...
HealthCareCenter
00:00:00
0 rows
Ready
Ln 2
Col 41
Ch 41
INS


SQLQuery4.sql - LAPTOP-7A9J9EOP\SQLEXPRESS.HealthCareCenter (LAPTOP-7A9J9EOP\Mandar (54))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
Generic Debugger
HealthCareCenter
Execute Debug
Object Explorer
Connect
Tables
System Tables
FileTables
External Tables
dbo.Benefits
dbo.Billing
dbo.Department
dbo.Diagnostics
dbo.DoctorIDTable
dbo.EmployeeBenefits
dbo.EmployeePersonalInformat
dbo.EmployeeReviews
dbo.Employees
Columns
Keys
Constraints
Triggers
tr_DoctorIDUpdate
Indexes
Statistics
dbo.HealthInsuranceDetailsTabl
dbo.MedicalEquipmentDetails
dbo.MedicalEquipmentMapping
dbo.MedicationInformation
dbo.PatientDiagnosticRelation
dbo.PatientHealthHistory
SQLQuery4.sql - L...J9EOP\Mandar (54))*
Select PatientID,PatientFName,PatientLName,CheckInDateTime,CheckOutDateTime from Patients
100 %
Results
Messages
PatientID PatientFName PatientLName CheckInDateTime CheckOutDateTime
1 1 Marcus Alonso 2017-11-01 09:30:00.000 2017-12-10 14:55:25.603
2 2 Thomas Taylor 2017-11-15 09:11:10.000 NULL
3 3 Alex Jones 2017-11-29 09:11:10.000 2017-12-01 15:21:00.000
4 4 Victor Murphy 2017-12-01 10:22:10.000 2017-12-07 18:30:00.000
5 5 Tom Brown 2017-12-04 10:30:00.000 NULL
Query executed succes...
LAPTOP-7A9J9EOP\SQLEXPRESS ...
LAPTOP-7A9J9EOP\Mandar...
HealthCareCenter
00:00:00
5 rows
Ready
Ln 2
Col 90
Ch 90
INS

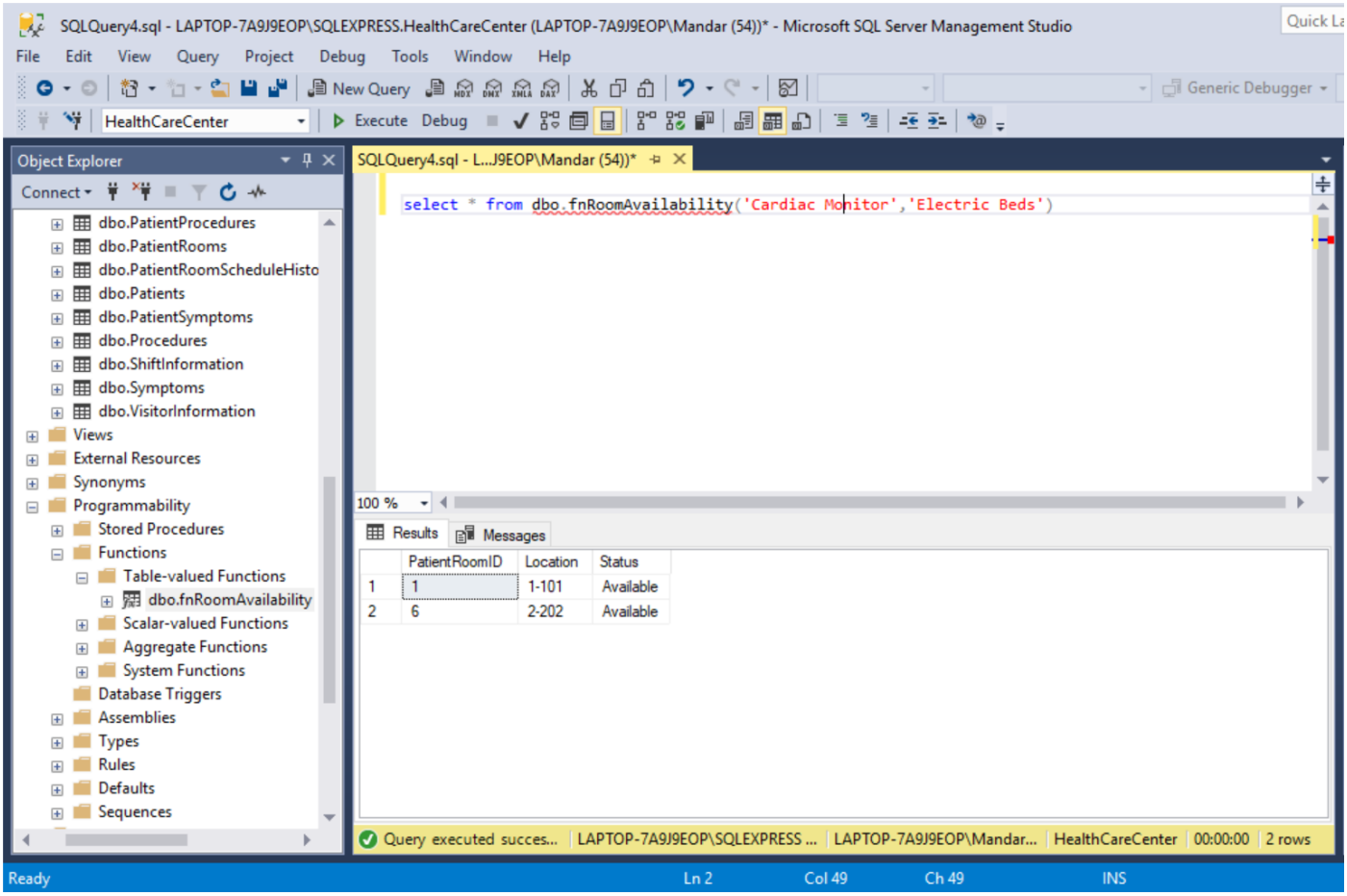
As we can see a corresponding CheckOutDateTime is entered in the respective column. And also a bill is generated for ‘Marcus Alonso’s’ patient ID which is 1.



Scenario 2:

To find out the next available room according to the type of bed and medical equipment available in the room specified.

We can use the function – fnRoomAvailability to achieve this purpose.

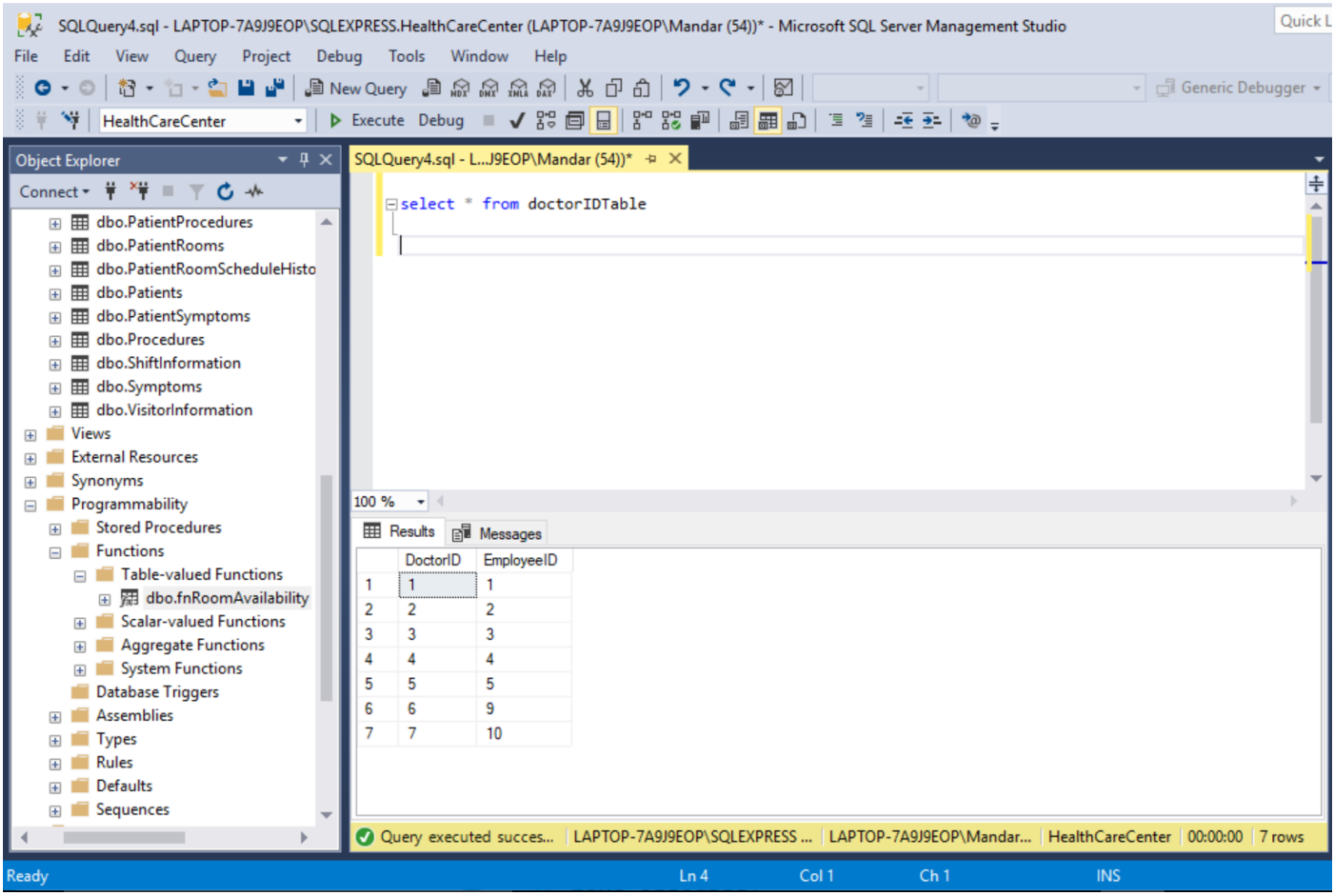


Here we want a list of all the rooms that have a Cardiac monitor and an electric bed as the bed type and the result set returned by the function is in the above image.

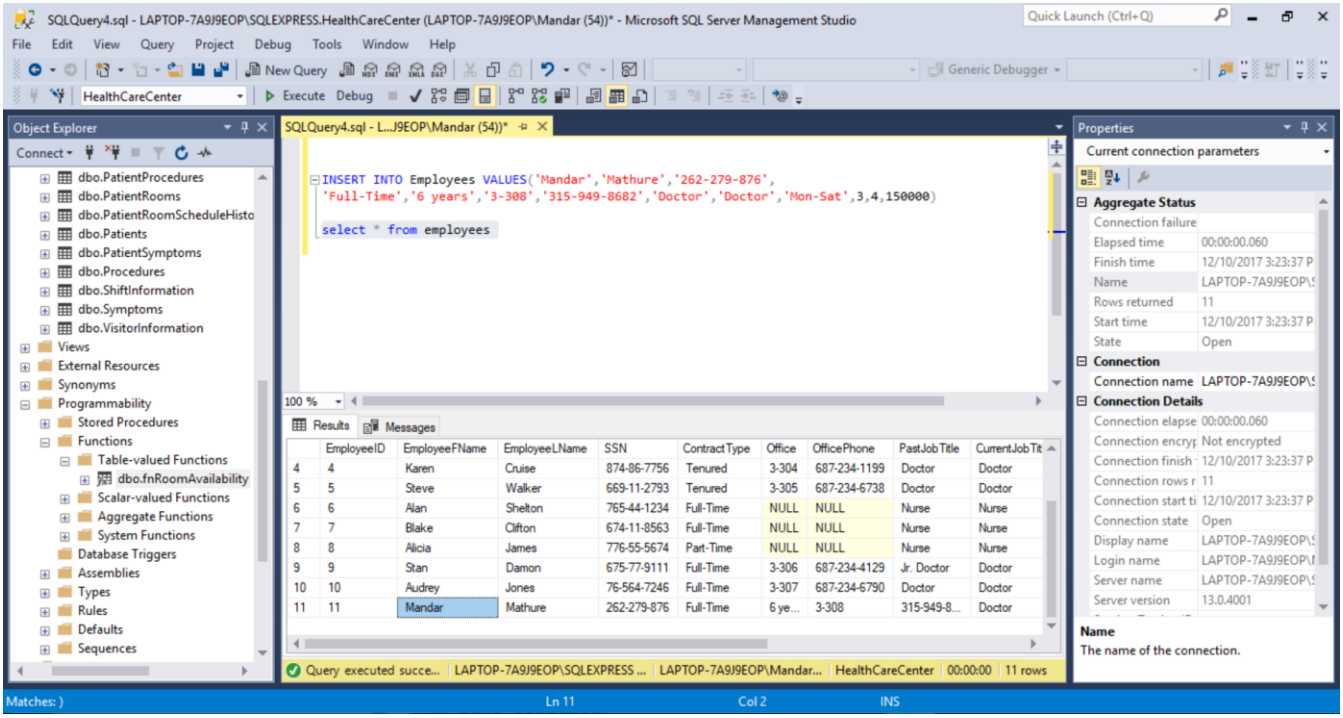
Scenario 3:

In this scenario we would test the trigger that we have created that automatically inserts a row into the doctors table when a doctor is inserted into the employees table.

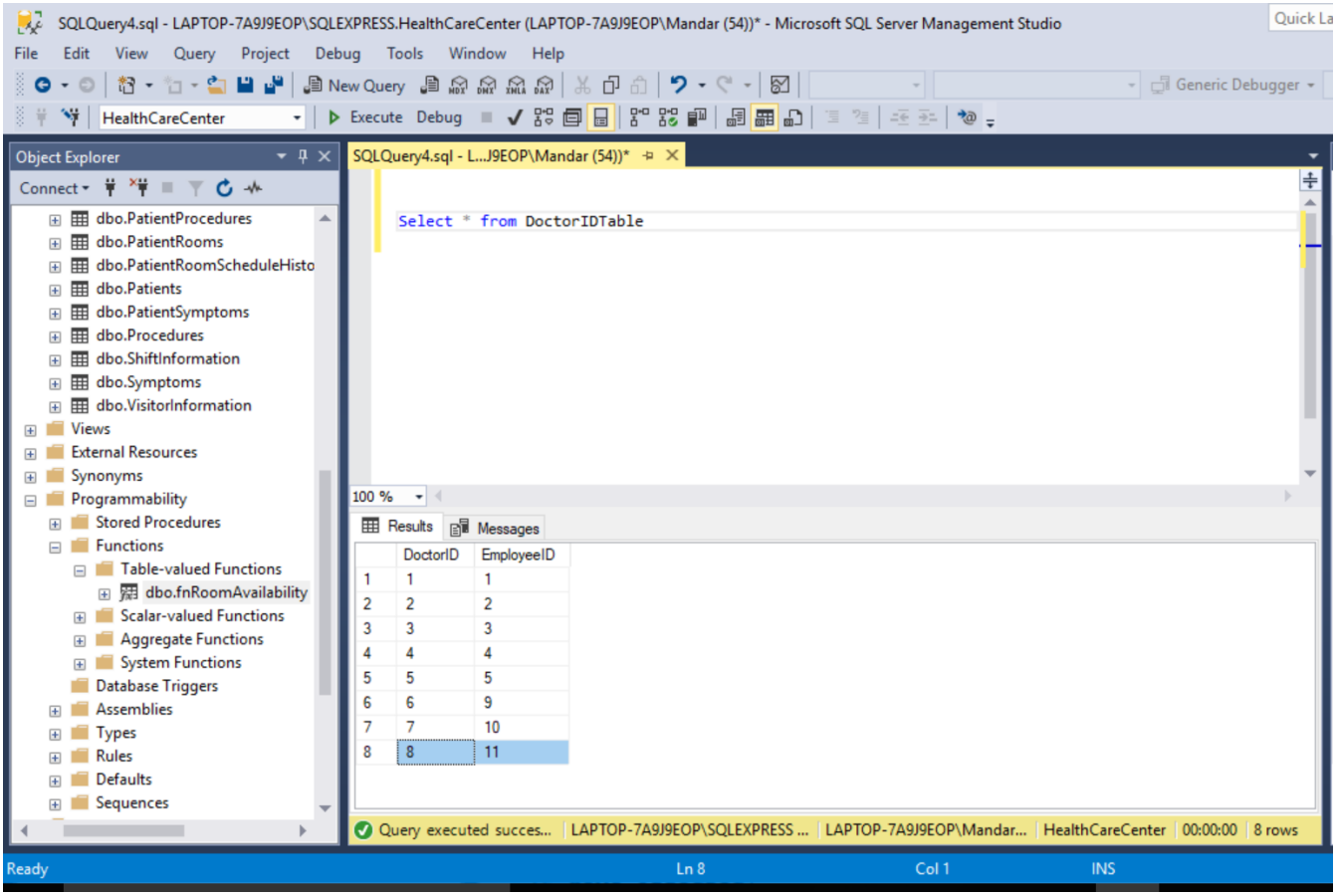
Before Insertion: we have 7 doctors.



After inserting a doctor in the employees table the DoctorID table should also be updated with the corresponding EmployeeID and DoctorID.



A new doctor is added in the employees table by the name ‘Mandar’. Now we have to check whether this doctor is also added into the DoctorID table.



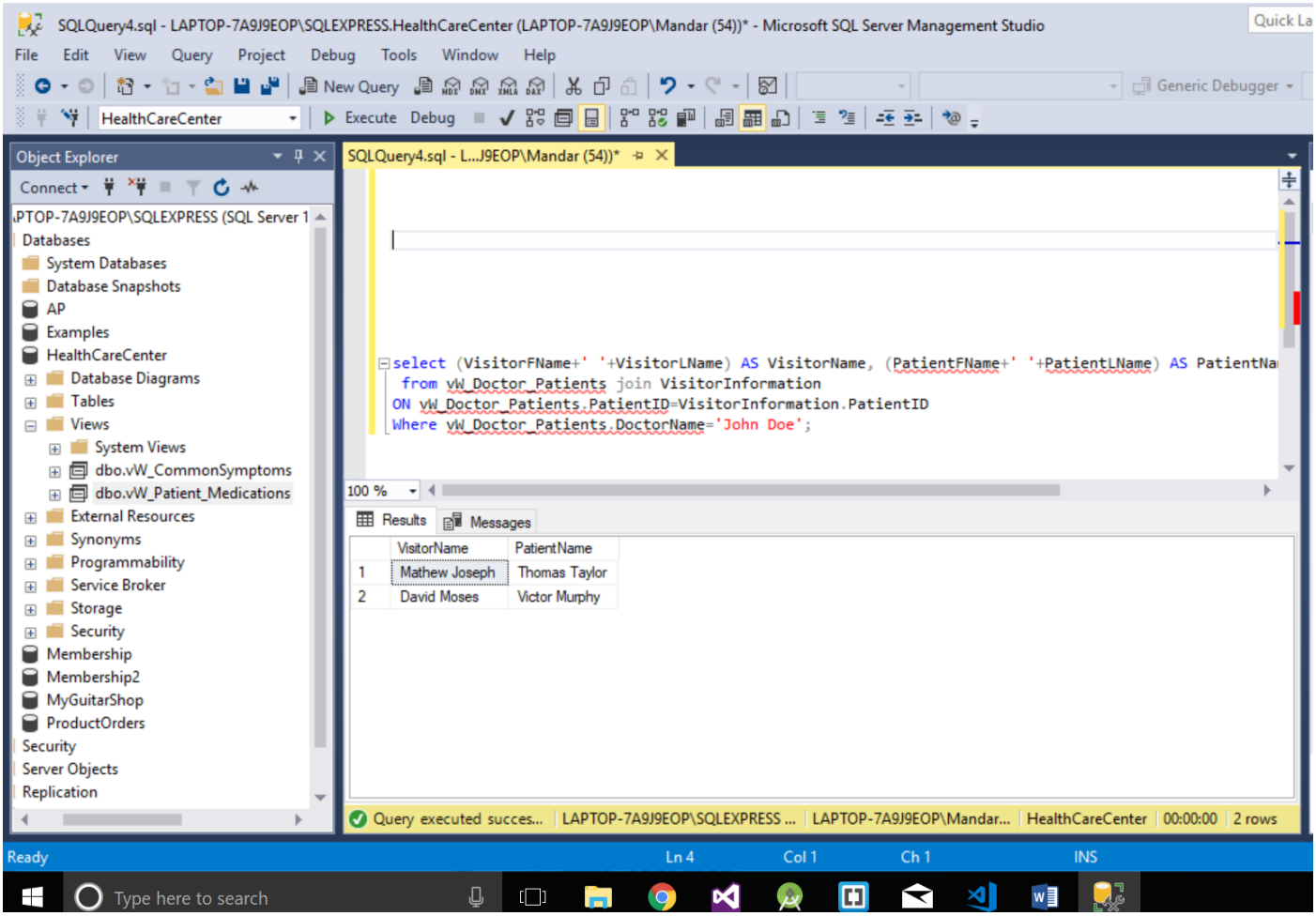
As we can see that the EmployeeID of ‘Mandar’ is added into the DoctorID table. Hence the trigger is working perfectly.

Scenario 4:

Consider a scenario where we want the names of the Visitors to the patients whose primary care doctor is say John Doe. A select query for the would be :

```
select (VisitorFName+' '+VisitorLName) AS VisitorName, (PatientFName+' '+PatientLName) AS
PatientName
 from vW_Doctor_Patients join VisitorInformation
ON vW_Doctor_Patients.PatientID=VisitorInformation.PatientID
Where vW_Doctor_Patients.DoctorName='John Doe';
```

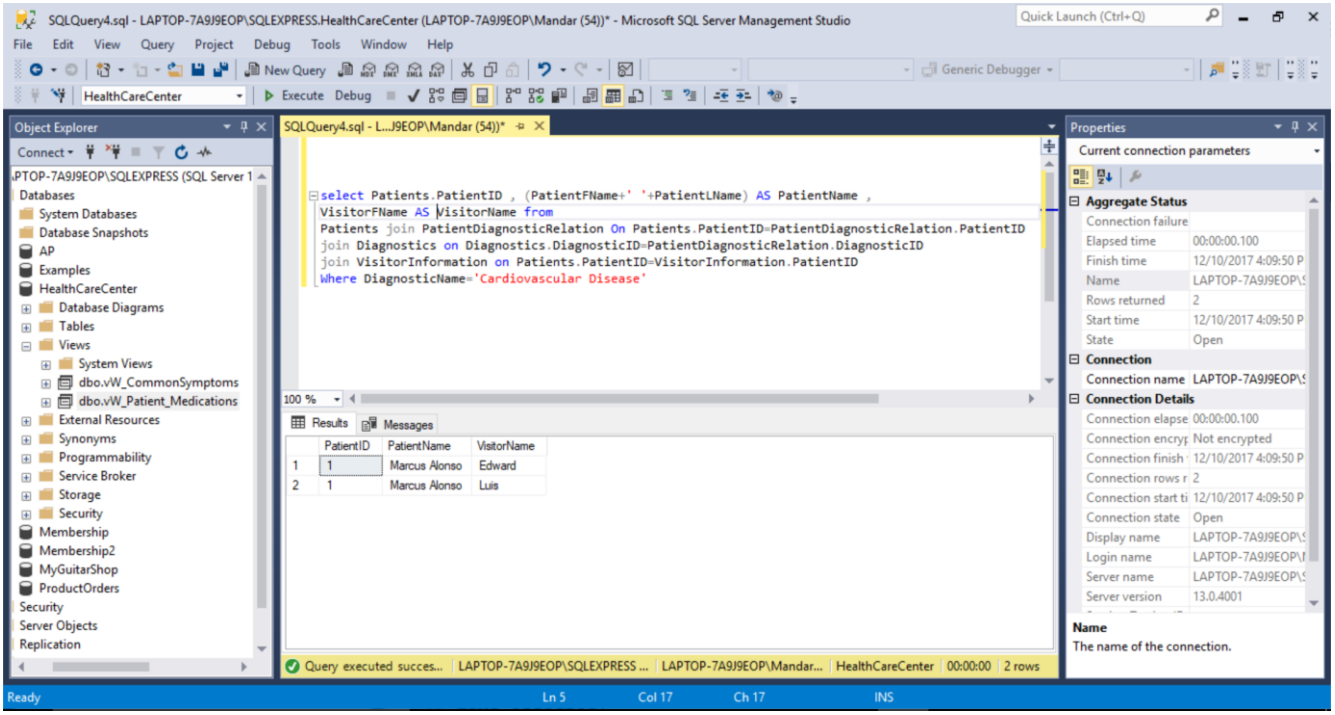
Here we are using the view that we have created earlier.



Scenario 5:

Consider a scenario where we want to find the visitor names of patients who are diagnosed with a particular disease. In this example we write a query to find out all the visitors who have recently visited a patient that has been diagnosed with a Cardiovascular disease.

```
select Patients.PatientID , (PatientFName+' '+PatientLName) AS PatientName ,
VisitorFName AS VisitorName from
Patients join PatientDiagnosticRelation On Patients.PatientID=PatientDiagnosticRelation.PatientID
join Diagnostics on Diagnostics.DiagnosticID=PatientDiagnosticRelation.DiagnosticID
join VisitorInformation on Patients.PatientID=VisitorInformation.PatientID
Where DiagnosticName='Cardiovascular Disease'
```



There are more patients who have been diagnosed with cardiovascular disease, but they do not have any visitors. Hence the only patient that has visitors and who has been diagnosed with cardiovascular disease is returned in the result set.

CONCLUSIONS AND REMARKS

This project was a comprehensive review of the whole course CSE 581 – DATABASE MANAGEMENT SYSTEMS. The project consists of designing and implementing a health care centre database according to some suggestions given in the Project requirements document. The project designs a database for a health care centre and models entity relationships according to real-world logic. The entity relationship diagram for this project is made in Vertabelo an online database design tool. Microsoft SQL management server is used as an implementation environment for the project. After designing and modelling of the whole project, the implementation step consists of writing a script that will automatically generate tables and insert 'test' data into our database. The database is designed according to a possible 'Business Model' that can be used for any similar system.

Indexes, Views, Stored Procedures, Functions, Triggers are also created keeping in mind various possible scenarios that could be encountered daily for inserting or retrieving data from the database system. In the testing section, 5 possible scenarios are presented, and the database integrity is thoroughly tested for the presented scenarios.

Overall the project consists of the following features:

- Database Design using ER diagrams(Vertabelo)
- Database Implementation using Microsoft SQL Server
- Database implementation Script including insertion of test data
- Implementations of Functions, Triggers, Stored Procedures
- 5 possible scenarios and their query results