

CSE 581 Intro to DBMS

PROJECT2

Rahul Ravikanti

981682170

Table Of Contents

A.	Abstract.....	3
B.	Tables Creation.....	4-10
C.	Data Insertion.....	11-20
D.	ER Diagram.....	21
E.	Views.....	22-28
F.	Stored Procedures.....	29-37
G.	Functions.....	38-45
H.	Triggers.....	46-53
I.	Transactions.....	54-61
J.	Scripts.....	62-65
K.	Business Reports.....	66-69
L.	Conclusion.....	70
	Remarks	71

Abstract

The development of a database for the Patient Access and Registration department at the University Medical Center is a significant project aimed at improving patient care and promoting coordination among healthcare providers. The database will be responsible for capturing patient data from admission to discharge, with a focus on optimizing hospital operations, reducing errors, and promoting resource utilization. Since the University Medical Center is a complex organization, the database will need to interface with several administrative departments, including Human Resources, Finance, Information Technology, Marketing, Health Information Management, Facilities Management, Compliance and Legal, and Quality and Risk Management.

The primary goal of this project is to streamline patient registration, including pre-registration, check-in, personal and insurance information, consent forms, payment collection, and medical history. Once registered, patients will undergo several processes, such as admission, diagnosis, treatment, monitoring, tests and procedures, consultations, and discharge planning before leaving the hospital. By organizing patient data in this manner, the database will ensure smooth patient transitions from hospital to home, improve patient care, and optimize resource utilization. Moreover, by providing accurate and up-to-date patient information to healthcare providers, the database will help reduce errors and facilitate coordination among different departments.

Overall, the development of a database for the Patient Access and Registration department at the University Medical Center is a vital initiative towards enhancing patient care and improving hospital operations. By promoting coordination among healthcare providers, optimizing resource utilization, and reducing errors, the database will help the University Medical Center provide better patient care and outcomes.

Tables Creation

The code for creation of tables is as follows:

```
CREATE TABLE Insurances(
    InsuranceID INT PRIMARY KEY IDENTITY(1,1),
    InsuranceName NVARCHAR(100) NOT NULL,
    InsuranceNumber NVARCHAR(100) NOT NULL,
    InsurancePlanType NVARCHAR(50) NOT NULL,
    InsuranceEffectiveDate DATE NOT NULL,
    InsuranceExpirationDate DATE NOT NULL,
);

CREATE TABLE LegalLicenses (
    LicenseID INT PRIMARY KEY IDENTITY(1,1),
    LicenseName NVARCHAR(100) NOT NULL,
    LicenseDescription NVARCHAR(500) NULL
);

CREATE TABLE HealthDepartments (
    DepartmentID INT PRIMARY KEY IDENTITY(1,1),
    DepartmentName NVARCHAR(100) NOT NULL,
    DepartmentDescription NVARCHAR(500) NULL
);

CREATE TABLE Patients (
    PatientID INT PRIMARY KEY IDENTITY(1,1),
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    DateOfBirth DATE NOT NULL,
    Gender CHAR(1) NOT NULL,
    PhoneNumber VARCHAR(20) NOT NULL,
    Address VARCHAR(100) NOT NULL,
    City VARCHAR(50) NOT NULL,
    State VARCHAR(50) NOT NULL,
    ZipCode VARCHAR(10) NOT NULL,
    InsuranceID INT NOT NULL,
    FOREIGN KEY (InsuranceID) REFERENCES Insurances (InsuranceID),
);
```

```
CREATE TABLE Physicians (
    PhysicianID INT PRIMARY KEY IDENTITY(1,1),
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL,
    DateOfBirth DATE NOT NULL,
    Gender NVARCHAR(10) NOT NULL,
    PhoneNumber NVARCHAR(20) NOT NULL,
    LicenseID INT NOT NULL,
    DepartmentID INT NOT NULL,
    CONSTRAINT FK_Physicians_License FOREIGN KEY (LicenseID) REFERENCES LegalLicenses(LicenseID),
    CONSTRAINT FK_Physicians_Department FOREIGN KEY (DepartmentID) REFERENCES HealthDepartments(DepartmentID)
);
```

```
CREATE TABLE Nurses (
    NurseID INT PRIMARY KEY IDENTITY(1,1),
    FirstName VARCHAR(255) NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    Gender CHAR(1) NOT NULL,
    DateOfBirth DATE NOT NULL,
    PhoneNumber VARCHAR(20) NOT NULL,
    Address VARCHAR(255) NOT NULL,
    City VARCHAR(255) NOT NULL,
    State VARCHAR(255) NOT NULL,
    ZipCode VARCHAR(10) NOT NULL,
    DutiesID INT NOT NULL,
    FOREIGN KEY (DutiesID) REFERENCES MedicalDuties(DutiesID)
);
```

```
CREATE TABLE Notes (
    NoteID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT NOT NULL,
    PhysicianID INT NOT NULL,
    NurseID INT NOT NULL,
    NoteDate DATETIME NOT NULL,
    NoteText NVARCHAR(MAX) NOT NULL,
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID),
    FOREIGN KEY (NurseID) REFERENCES Nurses(NurseID)
);
```

```
CREATE TABLE Appointments (
    AppointmentID INT PRIMARY KEY IDENTITY(1,1),
    PatientPhysicianID INT NOT NULL,
    AppointmentDateTime DATETIME NOT NULL,
    Reason VARCHAR(255) NOT NULL,
    CONSTRAINT FK_Appointments_PatientPhysicianID FOREIGN KEY (PatientPhysicianID)
    REFERENCES PatientPhysician (PatientPhysicianID),
    CONSTRAINT UQ_Appointments_AppointmentDateTime UNIQUE (AppointmentDateTime)
);
```

```
CREATE TABLE Admissions (
    AdmissionID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT NOT NULL,
    PhysicianID INT NOT NULL,
    RoomID INT NOT NULL,
    DiagnosisID INT NOT NULL,
    AdmissionDate DATETIME NOT NULL,
    DischargeDate DATETIME NULL DEFAULT NULL,
    FOREIGN KEY (DiagnosisID) REFERENCES MedicalDiagnosis(DiagnosisID),
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (RoomID) REFERENCES HospitalRooms(RoomID),
    FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID),
    CHECK (AdmissionDate <= DischargeDate)
);
```

```
CREATE TABLE Procedures (
    ProcedureID INT PRIMARY KEY IDENTITY(1,1),
    ICD10CodeID INT NOT NULL,
    PatientPhysicianID INT NOT NULL,
    ProcedureDate DATE NOT NULL,
    FOREIGN KEY (ICD10CodeID) REFERENCES ICD10Codes (ICD10CodeID),
    FOREIGN KEY (PatientPhysicianID) REFERENCES PatientPhysician (PatientPhysicianID)
);
```

```
CREATE TABLE Billing (
    BillingID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT NOT NULL,
    PaymentID INT NOT NULL,
    BillingDate DATE NOT NULL,
    Amount DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (PatientID) REFERENCES Patients (PatientID),
```

```
    FOREIGN KEY (PaymentID) REFERENCES Payments (PaymentID),  
);
```

```
CREATE TABLE LabResults(  
    LabResultsID INT PRIMARY KEY IDENTITY(1,1),  
    TestName VARCHAR(100),  
    TestDate DATE,  
    TestResults VARCHAR(100),  
    PatientID INT,  
    PhysicianID INT,  
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),  
    FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID)  
);
```

```
CREATE TABLE Medications (  
    MedicationID INT PRIMARY KEY IDENTITY(1,1),  
    MedicationName VARCHAR(100),  
    Dosage VARCHAR(50),  
    Frequency VARCHAR(50),  
    PrescribingPhysicianID INT,  
    PatientID INT,  
    FOREIGN KEY (PrescribingPhysicianID) REFERENCES Physicians(PhysicianID),  
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID)  
);
```

```
CREATE TABLE ICD10Codes (  
    ICD10CodeID INT PRIMARY KEY IDENTITY(1,1),  
    Code VARCHAR(10) NOT NULL,  
    Description VARCHAR(255) NOT NULL  
);
```

```
CREATE TABLE Payments (  
    PaymentID INT PRIMARY KEY IDENTITY(1,1),  
    PaymentDate DATE NOT NULL,  
    Amount DECIMAL(10,2) NOT NULL  
);
```

```
CREATE TABLE PatientPhysician (
    PatientPhysicianID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT NOT NULL,
    PhysicianID INT NOT NULL,
    CONSTRAINT FK_PatientPhysician_PatientID FOREIGN KEY (PatientID) REFERENCES Patients
    (PatientID),
    CONSTRAINT FK_PatientPhysician_PhysicianID FOREIGN KEY (PhysicianID) REFERENCES
    Physicians (PhysicianID)
);
```

```
CREATE TABLE MedicalDuties (
    DutiesID INT PRIMARY KEY IDENTITY(1,1),
    DutiesName NVARCHAR(100) NOT NULL,
    DutiesDescription NVARCHAR(500) NULL
);
```

```
CREATE TABLE HospitalRooms (
    RoomID INT PRIMARY KEY IDENTITY(1,1),
    RoomNumber NVARCHAR(50) NOT NULL,
    RoomType NVARCHAR(50) NOT NULL,
    RoomDescription NVARCHAR(500) NULL
);
```

```
CREATE TABLE MedicalDiagnosis (
    DiagnosisID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT NOT NULL,
    PhysicianID INT NOT NULL,
    ICD10CodeID INT NOT NULL,
    DiagnosisDate DATE NOT NULL,
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID),
    FOREIGN KEY (ICD10CodeID) REFERENCES ICD10Codes (ICD10CodeID),
);
```

```
CREATE TABLE MedicationHistory(
    MedicationHistoryID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT NOT NULL,
    MedicationID INT NOT NULL,
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (MedicationID) REFERENCES Medications(MedicationID)
);
```

```
CREATE TABLE MedicationCode (
    MedicationCodeID INT PRIMARY KEY IDENTITY(1,1),
    MedicationID INT,
    CodeValue VARCHAR(50),
    CodeDescription VARCHAR(255),
    FOREIGN KEY (MedicationID) REFERENCES Medications(MedicationID)
);
```

```
CREATE TABLE Schedule (
    ScheduleID INT PRIMARY KEY IDENTITY(1,1),
    PhysicianID INT NOT NULL,
    AppointmentID INT NOT NULL,
    ScheduleDate DATE NOT NULL,
    ScheduleTime TIME NOT NULL,
    FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID),
    FOREIGN KEY (AppointmentID) REFERENCES Appointments(AppointmentID),
    UNIQUE (PhysicianID, ScheduleDate, ScheduleTime),
    CHECK (ScheduleDate >= GETDATE()),
    CHECK (ScheduleTime >= '00:00:00' AND ScheduleTime <= '23:59:59')
);
```

```
CREATE TABLE DischargeInstructions(
    DischargeInstructionID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT NOT NULL,
    PhysicianID INT NOT NULL,
    DischargeInstructions VARCHAR(250),
    DischargeDate DATE NOT NULL,
    FOREIGN KEY (PatientID) REFERENCES Patients (PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID)
);
```

The tables created in UniversityMedicalCenter

Tables	
>	dbo.Admissions
>	dbo.Appointments
>	dbo.Billing
>	dbo.DischargeInstructions
>	dbo.HealthDepartments
>	dbo.HospitalRooms
>	dbo.ICD10Codes
>	dbo.Insurances
>	dbo.LabResults
>	dbo.LegalLicenses
>	dbo.MedicalDiagnosis
>	dbo.MedicalDuties
>	dbo.MedicationCode
>	dbo.MedicationHistory
>	dbo.Medications
>	dbo.Notes
>	dbo.Nurses
>	dbo.PatientPhysician
>	dbo.Patients
>	dbo.Payments
>	dbo.Physicians
>	dbo.Procedures
>	dbo.Schedule

The insert statements for inserting data into above tables

```
INSERT INTO LegalLicenses (LicenseName, LicenseDescription) VALUES
('Medical License', 'This license allows a person to practice medicine legally.'),
('Nursing License', 'This license allows a person to practice nursing legally.'),
('Pharmacy License', 'This license allows a person to operate a pharmacy legally.'),
('Dental License', 'This license allows a person to practice dentistry legally.'),
('Physical Therapy License', 'This license allows a person to practice physical therapy legally.'),
('Occupational Therapy License', 'This license allows a person to practice occupational therapy legally.'),
('Chiropractic License', 'This license allows a person to practice chiropractic care legally.'),
('Psychologist License', 'This license allows a person to practice psychology legally.'),
('Veterinary License', 'This license allows a person to practice veterinary medicine legally.'),
('EMT License', 'This license allows a person to practice emergency medical services legally.');
```

```
INSERT INTO Insurances (InsuranceName, InsuranceNumber, InsurancePlanType,
InsuranceEffectiveDate, InsuranceExpirationDate) VALUES
('Blue Cross Blue Shield', '12345', 'PPO', '2022-01-01', '2022-12-31'),
('Aetna', '98765', 'HMO', '2022-02-01', '2022-11-30'),
('Cigna', '45678', 'POS', '2022-03-01', '2022-12-31'),
('Humana', '87654', 'EPO', '2022-04-01', '2022-10-31'),
('United Healthcare', '23456', 'HSA', '2022-05-01', '2022-12-31'),
('Kaiser Permanente', '56789', 'HMO', '2022-06-01', '2022-11-30'),
('Molina Healthcare', '54321', 'PPO', '2022-07-01', '2022-12-31'),
('Oscar Health', '67890', 'POS', '2022-08-01', '2022-10-31'),
('Medicare', '78901', 'Original Medicare', '2022-09-01', '2022-12-31'),
('Medicaid', '90123', 'Managed Medicaid', '2022-10-01', '2022-12-31');
```

```
INSERT INTO HealthDepartments (DepartmentName, DepartmentDescription) VALUES
('Pediatrics', 'Department of medicine focused on the care of infants, children, and
adolescents.'),
('Ophthalmology', 'Department of medicine focused on the diagnosis and treatment of eye
disorders.'),
('Orthopedics', 'Department of medicine focused on the diagnosis and treatment of
musculoskeletal disorders.'),
('Neurology', 'Department of medicine focused on the diagnosis and treatment of disorders of
the nervous system.'),
('Cardiology', 'Department of medicine focused on the diagnosis and treatment of heart and
cardiovascular disorders.'),
('Gastroenterology', 'Department of medicine focused on the diagnosis and treatment of
digestive system disorders.'),
('Obstetrics and Gynecology', 'Department of medicine focused on the care of women during
pregnancy, childbirth, and the postpartum period.'),
('Dermatology', 'Department of medicine focused on the diagnosis and treatment of skin
disorders.'),
('Otolaryngology', 'Department of medicine focused on the diagnosis and treatment of ear,
nose, and throat disorders.'),
('Psychiatry', 'Department of medicine focused on the diagnosis and treatment of mental health
disorders.');
```

```
INSERT INTO Patients (FirstName, LastName, DateOfBirth, Gender, PhoneNumber, Address,
City, State, ZipCode, InsuranceID) VALUES
('John', 'Doe', '1985-06-12', 'M', '555-555-1212', '123 Main St', 'Anytown', 'CA', '12345', 1),
('Jane', 'Smith', '1990-09-22', 'F', '555-555-1313', '456 Maple Ave', 'Othertown', 'NY', '54321', 2),
('David', 'Lee', '1972-03-04', 'M', '555-555-1414', '789 Oak Blvd', 'Somewhere', 'TX', '67890', 3),
('Lisa', 'Nguyen', '1988-12-31', 'F', '555-555-1515', '321 Elm St', 'Anytown', 'CA', '12345', 1),
('Michael', 'Garcia', '1975-07-20', 'M', '555-555-1616', '654 Cedar Ave', 'Othertown', 'NY',
'54321', 2),
('Emily', 'Kim', '1995-02-15', 'F', '555-555-1717', '987 Pine St', 'Somewhere', 'TX', '67890', 3),
('William', 'Johnson', '1980-11-02', 'M', '555-555-1818', '555 Fifth St', 'Anytown', 'CA', '12345',
1),
('Sarah', 'Martinez', '1989-04-27', 'F', '555-555-1919', '777 Sixth Ave', 'Othertown', 'NY', '54321',
2),
('Daniel', 'Rodriguez', '1978-08-14', 'M', '555-555-2020', '111 Seventh St', 'Somewhere', 'TX',
'67890', 3),
('Megan', 'Chen', '1993-01-07', 'F', '555-555-2121', '222 Eighth Ave', 'Anytown', 'CA', '12345', 1);
```

```
INSERT INTO Physicians (FirstName, LastName, DateOfBirth, Gender, PhoneNumber, LicenseID, DepartmentID) VALUES
```

```
('Sophia', 'Lee', '1986-09-12', 'F', '555-555-1212', 11, 7),
('Ethan', 'Nguyen', '1978-11-23', 'M', '555-555-1313', 12, 8),
('Ava', 'Garcia', '1991-02-01', 'F', '555-555-1414', 13, 9),
('Jacob', 'Kim', '1983-05-31', 'M', '555-555-1515', 14, 10),
('Isabella', 'Johnson', '1988-08-10', 'F', '555-555-1616', 15, 11),
('Noah', 'Martinez', '1985-03-17', 'M', '555-555-1717', 16, 12),
('Olivia', 'Rodriguez', '1993-07-28', 'F', '555-555-1818', 17, 13),
('Liam', 'Chen', '1980-12-02', 'M', '555-555-1919', 18, 14),
('Emma', 'Gonzalez', '1982-01-15', 'F', '555-555-2020', 19, 15),
('Mason', 'Perez', '1974-09-30', 'M', '555-555-2121', 20, 16);
```

```
INSERT INTO PatientPhysician (PatientID, PhysicianID)
```

```
VALUES (1, 2),
(2, 3),
(3, 4),
(4, 5),
(5, 6),
(6, 7),
(7, 8),
(8, 9),
(9, 10),
(10, 1);
```

```
INSERT INTO Medications (MedicationName, Dosage, Frequency, PrescribingPhysicianID, PatientID)
```

```
VALUES ('Lisinopril', '20mg', 'Daily', 3, 1),
('Metformin', '1000mg', 'Twice daily', 5, 2),
('Levothyroxine', '75mcg', 'Daily', 2, 3),
('Atorvastatin', '40mg', 'Nightly', 1, 4),
('Albuterol', '90mcg', 'As needed', 4, 5),
('Ibuprofen', '800mg', 'Every 8 hours', 2, 6),
('Hydrocodone', '10mg', 'As needed', 6, 7),
('Trazodone', '50mg', 'Nightly', 1, 8),
('Omeprazole', '20mg', 'Daily', 4, 9),
('Aspirin', '81mg', 'Daily', 5, 10);
```

```
INSERT INTO MedicationHistory (PatientID, MedicationID)
VALUES (1, 1),
(2, 3),
(3, 5),
(4, 7),
(5, 9),
(6, 2),
(7, 4),
(8, 6),
(9, 8),
(10, 10);
```

```
INSERT INTO MedicalDuties (DutiesName, DutiesDescription)
VALUES
('Diagnosing and treating illnesses', 'Physicians diagnose and treat illnesses and injuries'),
('Prescribing and administering medication', 'Physicians prescribe and administer medications to patients'),
('Performing medical procedures', 'Physicians perform medical procedures such as surgeries'),
('Ordering and interpreting diagnostic tests', 'Physicians order and interpret diagnostic tests such as blood tests, X-rays, and CT scans'),
('Providing preventative care and health education', 'Physicians provide preventative care and health education to patients'),
('Managing patient care and medical records', 'Physicians manage patient care and medical records'),
('Collaborating with other healthcare professionals', 'Physicians collaborate with other healthcare professionals such as nurses and therapists'),
('Managing and supervising medical staff', 'Physicians manage and supervise medical staff such as nurses and medical assistants'),
('Researching and developing new treatments and technologies', 'Physicians participate in research and development of new treatments and technologies'),
('Advocating for patient rights and healthcare policy', 'Physicians advocate for patient rights and healthcare policy');
```

```
INSERT INTO Nurses (FirstName, LastName, Gender, DateOfBirth, PhoneNumber, Address, City, State, ZipCode, DutiesID)
```

```
VALUES
```

```
('Emily', 'Smith', 'F', '1990-05-12', '555-1234', '123 Main St', 'New York', 'NY', '10001', 1),
('Megan', 'Johnson', 'F', '1993-09-15', '555-2345', '456 Maple Ave', 'Los Angeles', 'CA', '90001', 2),
('Liam', 'Brown', 'M', '1991-02-20', '555-3456', '789 Oak St', 'Chicago', 'IL', '60007', 1),
('Avery', 'Davis', 'F', '1989-11-25', '555-4567', '321 Pine Rd', 'Miami', 'FL', '33010', 3),
('Noah', 'Garcia', 'M', '1992-07-04', '555-5678', '654 Cedar Blvd', 'Dallas', 'TX', '75001', 2),
('Abigail', 'Martinez', 'F', '1988-12-30', '555-6789', '987 Elm St', 'San Francisco', 'CA', '94101', 3),
('Ethan', 'Wilson', 'M', '1994-03-10', '555-7890', '246 Birch Dr', 'Boston', 'MA', '02101', 1),
('Olivia', 'Anderson', 'F', '1990-08-22', '555-8901', '753 Oak Ave', 'Seattle', 'WA', '98101', 2),
('Mason', 'Taylor', 'M', '1993-01-17', '555-9012', '159 Maple Ln', 'Houston', 'TX', '77001', 3),
('Sophia', 'Thomas', 'F', '1989-06-03', '555-0123', '852 Elm Blvd', 'Atlanta', 'GA', '30301', 1);
```

```
INSERT INTO Notes (PatientID, PhysicianID, NurseID, NoteDate, NoteText)
```

```
VALUES
```

```
(1, 6, 1, '2023-04-29 13:00:00', 'Patient presented with mild fever and cough. Prescribed antibiotics.'),
```

```
(2, 7, 2, '2023-04-29 14:30:00', 'Patient complaining of pain in left knee. Advised to rest and prescribed pain medication.'),
```

```
(3, 8, 3, '2023-04-29 16:00:00', 'Patient recovering well from surgery. Incision site looks clean and dry.'),
```

```
(4, 9, 4, '2023-04-30 09:00:00', 'Patient reports feeling dizzy and lightheaded. Vital signs within normal range.'),
```

```
(5, 10, 5, '2023-04-30 10:30:00', 'Patient presented with rash on arms and legs. Prescribed antihistamines.'),
```

```
(6, 6, 6, '2023-04-30 12:00:00', 'Patient feeling better after taking antibiotics. Advised to complete full course of medication.'),
```

```
(7, 7, 7, '2023-04-30 13:30:00', 'Patient reports feeling more pain in left knee. Prescribed stronger pain medication.'),
```

```
(8, 8, 8, '2023-04-30 15:00:00', 'Patient discharged from hospital. Instructed to follow up with physician in two weeks.'),
```

```
(9, 9, 9, '2023-04-30 16:30:00', 'Patient feeling better after resting for an hour. Discharged from emergency department.'),
```

```
(10, 10, 10, '2023-05-01 09:00:00', 'Patient reports no improvement in rash. Referred to dermatologist for further evaluation.');
```

```
INSERT INTO Appointments (PatientPhysicianID, AppointmentDateTime, Reason)  
VALUES
```

```
(1, '2023-05-10 10:00:00', 'Routine check-up'),  
(2, '2023-05-11 14:30:00', 'Follow-up after surgery'),  
(3, '2023-05-12 09:00:00', 'Annual physical exam'),  
(4, '2023-05-13 11:00:00', 'Consultation for back pain'),  
(5, '2023-05-14 13:00:00', 'Vaccination appointment'),  
(6, '2023-05-15 15:30:00', 'Consultation for allergies'),  
(7, '2023-05-16 16:00:00', 'Follow-up for diabetes management'),  
(8, '2023-05-17 10:30:00', 'Routine check-up'),  
(9, '2023-05-18 11:30:00', 'Follow-up after hospitalization'),  
(10, '2023-05-19 14:00:00', 'Consultation for weight management');
```

```
INSERT INTO ICD10Codes (Code, Description)
```

```
VALUES
```

```
('A00', 'Cholera'),  
(‘A01’, ‘Typhoid and paratyphoid fevers’),  
(‘A02’, ‘Other salmonella infections’),  
(‘A03’, ‘Shigellosis’),  
(‘A04’, ‘Other bacterial intestinal infections’),  
(‘A05’, ‘Other bacterial foodborne intoxications, not elsewhere classified’),  
(‘A06’, ‘Amoebiasis’),  
(‘A07’, ‘Other protozoal intestinal diseases’),  
(‘A08’, ‘Viral and other specified intestinal infections’),  
(‘A09’, ‘Diarrhoea and gastro-enteritis of presumed infectious origin’);
```

```
INSERT INTO MedicalDiagnosis (PatientID, PhysicianID, ICD10CodeID, DiagnosisDate)
```

```
VALUES
```

```
(1, 6, 1, '2023-04-29'),  
(2, 7, 2, '2023-04-29'),  
(3, 8, 3, '2023-04-29'),  
(4, 9, 4, '2023-04-30'),  
(5, 10, 5, '2023-04-30'),  
(6, 6, 6, '2023-04-30'),  
(7, 7, 7, '2023-04-30'),  
(8, 8, 8, '2023-04-30'),  
(9, 9, 9, '2023-04-30'),  
(10, 10, 10, '2023-05-01');
```

```
INSERT INTO HospitalRooms (RoomNumber, RoomType, RoomDescription)
VALUES
('101', 'Standard', 'A standard room with a single bed'),
('102', 'Standard', 'A standard room with a single bed'),
('103', 'Standard', 'A standard room with a single bed'),
('201', 'Standard', 'A standard room with a single bed'),
('202', 'Standard', 'A standard room with a single bed'),
('203', 'Standard', 'A standard room with a single bed'),
('301', 'Deluxe', 'A deluxe room with two beds and a view of the garden'),
('302', 'Deluxe', 'A deluxe room with two beds and a view of the garden'),
('303', 'Deluxe', 'A deluxe room with two beds and a view of the garden'),
('401', 'Suite', 'A luxurious suite with a king-sized bed and a separate living area');
```

```
INSERT INTO Admissions (PatientID, PhysicianID, RoomID, DiagnosisID, AdmissionDate,
DischargeDate)
VALUES
(1, 3, 2, 1, '2022-04-01 12:00:00', '2022-04-05 10:00:00'),
(2, 2, 1, 3, '2022-05-10 08:00:00', '2022-05-15 14:00:00'),
(3, 1, 3, 2, '2022-06-20 16:00:00', '2022-06-22 18:00:00'),
(4, 4, 4, 4, '2022-07-03 10:00:00', '2022-07-06 12:00:00'),
(5, 2, 5, 5, '2022-08-01 07:00:00', '2022-08-02 16:00:00'),
(6, 3, 6, 6, '2022-09-05 14:00:00', NULL),
(7, 1, 7, 7, '2022-10-11 11:00:00', NULL),
(8, 4, 8, 8, '2022-11-20 08:00:00', NULL),
(9, 2, 9, 9, '2022-12-01 15:00:00', NULL),
(10, 3, 10, 10, '2023-01-02 10:00:00', NULL);
```

```
INSERT INTO Procedures (ICD10CodeID, PatientPhysicianID, ProcedureDate)
VALUES
(1, 1, '2022-03-15'),
(2, 2, '2022-04-20'),
(3, 3, '2022-05-10'),
(4, 4, '2022-06-01'),
(5, 5, '2022-07-04'),
(6, 6, '2022-08-15'),
(7, 7, '2022-09-21'),
(8, 8, '2022-10-10'),
(9, 9, '2022-11-02'),
(10, 10, '2022-12-25');
```

```
INSERT INTO MedicationCode (MedicationID, CodeValue, CodeDescription)
VALUES
```

```
(1, '123', 'Medication A code 123'),
(2, '456', 'Medication B code 456'),
(3, '789', 'Medication C code 789'),
(4, '321', 'Medication D code 321'),
(5, '654', 'Medication E code 654'),
(6, '987', 'Medication F code 987'),
(7, '741', 'Medication G code 741'),
(8, '852', 'Medication H code 852'),
(9, '963', 'Medication I code 963'),
(10, '159', 'Medication J code 159');
```

```
INSERT INTO LabResults (TestName, TestDate, TestResults, PatientID, PhysicianID)
VALUES
```

```
('Blood sugar test', '2023-04-29', '120 mg/dL', 1, 5),
('Urinalysis', '2023-04-28', 'Normal', 2, 6),
('Complete blood count', '2023-04-27', 'Normal', 3, 7),
('Electrocardiogram', '2023-04-26', 'Sinus rhythm', 4, 8),
('Stool culture', '2023-04-25', 'Positive for Salmonella', 5, 9),
('Thyroid function test', '2023-04-24', 'Normal', 6, 10),
('Liver function test', '2023-04-23', 'Elevated ALT levels', 7, 1),
('Kidney function test', '2023-04-22', 'Normal', 8, 2),
('Lipid panel', '2023-04-21', 'Elevated LDL cholesterol', 9, 3),
('Pregnancy test', '2023-04-20', 'Positive', 10, 4);
```

```
INSERT INTO Schedule (PhysicianID, AppointmentID, ScheduleDate, ScheduleTime)
VALUES
```

```
(1, 1, '2023-05-01', '09:00:00'),
(1, 2, '2023-05-02', '10:00:00'),
(1, 3, '2023-05-03', '11:00:00'),
(2, 4, '2023-05-04', '14:00:00'),
(2, 5, '2023-05-05', '15:00:00'),
(2, 6, '2023-05-06', '16:00:00'),
(3, 7, '2023-05-07', '09:00:00'),
(3, 8, '2023-05-08', '10:00:00'),
(3, 9, '2023-05-09', '11:00:00'),
(4, 10, '2023-05-10', '14:00:00');
```

```
INSERT INTO Payments (PaymentDate, Amount)
```

```
VALUES
```

```
('2023-04-30', 500.00),  
('2023-04-29', 1000.00),  
('2023-04-28', 750.00),  
('2023-04-27', 1250.00),  
('2023-04-26', 900.00),  
('2023-04-25', 600.00),  
('2023-04-24', 800.00),  
('2023-04-23', 1500.00),  
('2023-04-22', 1100.00),  
('2023-04-21', 950.00);
```

```
INSERT INTO Billing (PatientID, PaymentID, BillingDate, Amount)
```

```
VALUES
```

```
(1, 1, '2023-04-01', 500.00),  
(2, 2, '2023-04-02', 750.00),  
(3, 3, '2023-04-03', 1000.00),  
(4, 4, '2023-04-04', 1500.00),  
(5, 5, '2023-04-05', 2000.00),  
(6, 6, '2023-04-06', 2500.00),  
(7, 7, '2023-04-07', 3000.00),  
(8, 8, '2023-04-08', 3500.00),  
(9, 9, '2023-04-09', 4000.00),  
(10, 10, '2023-04-10', 4500.00);
```

```
INSERT INTO DischargeInstructions (PatientID, PhysicianID, DischargeInstructions,  
DischargeDate)
```

```
VALUES
```

```
(1, 2, 'Take medication as prescribed', '2022-04-01'),  
(1, 2, 'Follow up with primary care physician in 1 week', '2022-04-01'),  
(3, 4, 'Increase physical activity gradually', '2022-04-03'),  
(3, 4, 'Return to work in 2 weeks', '2022-04-03'),  
(5, 6, 'Maintain a healthy diet and exercise regimen', '2022-04-05'),  
(5, 6, 'Take medication as directed by physician', '2022-04-05'),  
(7, 8, 'Avoid strenuous activity for 1 week', '2022-04-07'),  
(7, 8, 'Follow up with specialist in 2 weeks', '2022-04-07'),  
(9, 10, 'Get plenty of rest and drink fluids', '2022-04-09'),  
(9, 10, 'Take medication as prescribed', '2022-04-09');
```

Successful execution of insert statements

The screenshot shows the SSMS interface with three query panes open:

- SQLQuery_1 - local...re (sa)**: Contains the following INSERT INTO Billing statement:

```

265 ('2023-04-25', 600.00),
266 ('2023-04-24', 800.00),
267 ('2023-04-23', 1500.00),
268 ('2023-04-22', 1100.00),
269 ('2023-04-21', 950.00);
    
```

- SQLQuery_2 - local...er (sa)**: Contains the following INSERT INTO Billing statement:

```

271 INSERT INTO Billing (PatientID, PaymentID, BillingDate, Amount)
272 VALUES
273 (1, 1, '2023-04-01', 500.00),
274 (1, 2, '2023-04-02', 750.00),
275 (3, 3, '2023-04-03', 1000.00),
276 (4, 4, '2023-04-04', 1500.00),
277 (5, 5, '2023-04-05', 2000.00),
278 (6, 6, '2023-04-06', 2500.00),
279 (7, 7, '2023-04-07', 3000.00),
280 (8, 8, '2023-04-08', 3500.00),
281 (9, 9, '2023-04-09', 4000.00),
282 (10, 10, '2023-04-10', 4500.00);
    
```

- SQLQuery_3 - local...er (sa)**: Contains the following INSERT INTO DischargeInstructions statement:

```

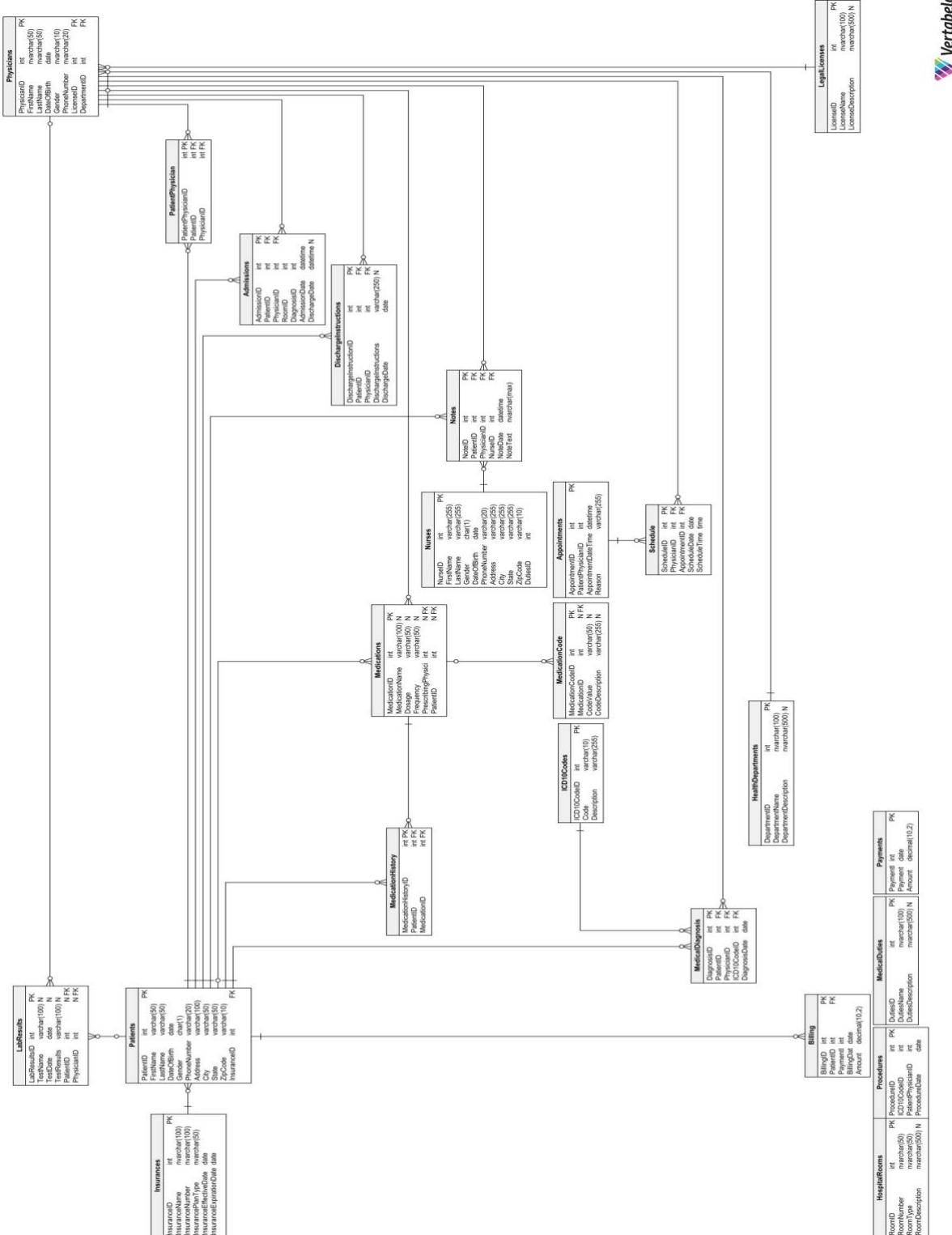
283 INSERT INTO DischargeInstructions (PatientID, PhysicianID, DischargeInstructions, DischargeDate)
284 VALUES
285
286 (1, 1, 'Take medication as prescribed', '2022-04-01'),
287 (1, 2, 'Follow up with primary care physician in 1 week', '2022-04-01'),
288 (3, 4, 'Increase physical activity gradually', '2022-04-03'),
289 (3, 4, 'Return to work in 2 weeks', '2022-04-03'),
290 (5, 6, 'Maintain a healthy diet and exercise regimen', '2022-04-05'),
291 (5, 6, 'Take medication as directed by physician', '2022-04-05'),
292 (7, 8, 'Avoid strenuous activity for 1 week', '2022-04-07'),
293 (7, 8, 'Follow up with specialist in 2 weeks', '2022-04-07'),
294 (9, 10, 'Get plenty of rest and drink fluids', '2022-04-09'),
295 (9, 10, 'Take medication as prescribed', '2022-04-09');
    
```

Messages pane:

Started executing query at Line 285
 (10 rows affected)
 Total execution time: 00:00:00.053

Ln 296, Col 56 Spaces: 4 UTF-8 LF 0 rows MSSQL 00:00:00 localhost:UniversityMedicalCenter

The ER Diagram for the database UniversityMedicalCenter is as follows (Rotated).



Code for creating views

View that shows all patients and their corresponding medications:

```
CREATE VIEW PatientMedications AS
SELECT p.FirstName, p.LastName, m.MedicationName, m.Dosage, m.Frequency
FROM Patients p
INNER JOIN Medications m ON p.PatientID = m.PatientID;
```

```
select * from PatientBilling;
```

	FirstName	LastName	Amount	BillingDate
1	John	Doe	500.00	2023-04-01
2	Jane	Smith	750.00	2023-04-02
3	David	Lee	1000.00	2023-04-03
4	Lisa	Nguyen	1500.00	2023-04-04
5	Michael	Garcia	2000.00	2023-04-05
6	Emily	Kim	2500.00	2023-04-06
7	William	Johnson	3000.00	2023-04-07
8	Sarah	Martinez	3500.00	2023-04-08
9	Daniel	Rodriguez	4000.00	2023-04-09
10	Megan	Chavez	4500.00	2023-04-10

Retrieved 10 rows.

View that shows all patients and their corresponding billing information:

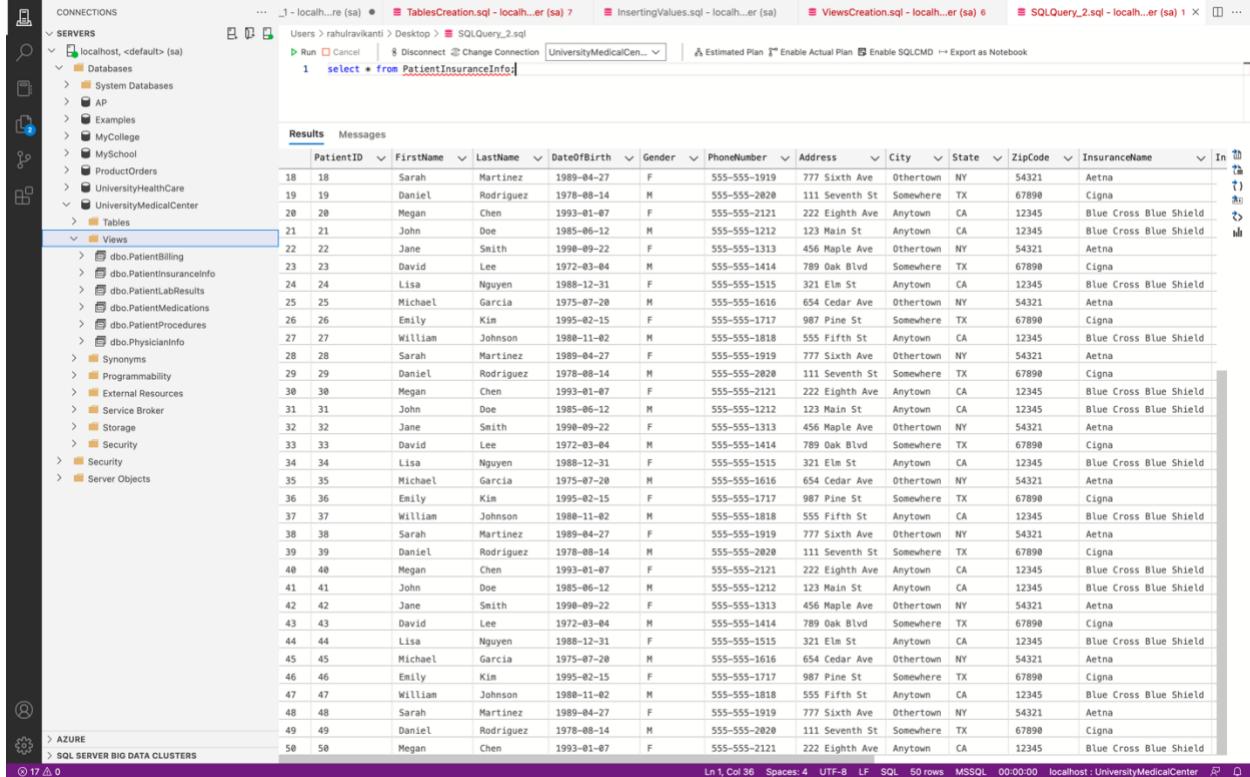
`CREATE VIEW PatientBilling AS`

`SELECT p.FirstName, p.LastName, b.Amount, b.BillingDate`

`FROM Patients p`

`INNER JOIN Billing b ON p.PatientID = b.PatientID;`

`select * from PatientInsuranceInfo;`



The screenshot shows the SSMS interface with the following details:

- Connections:** _1 - localhost...re (sa)
- Servers:** localhost, <default> (sa)
- Databases:** System Databases, AP, Examples, MyCollege, MySchool, ProductOrders, UniversityHealthCare
- Tables:** UniversityMedicalCenter
- Views:** PatientBilling, PatientInsuranceInfo (selected), PatientLabResults, PatientMedications, PatientProcedures, PhysicianInfo, Synonyms, Programmability, External Resources, Service Broker, Storage, Security, Server Objects
- Estimated Plan:** Enabled Actual Plan, Enable SQLCMD, Export as Notebook
- Query:** `select * from PatientInsuranceInfo;`
- Results Grid:** Displays 50 rows of data with columns: PatientID, FirstName, LastName, DateOfBirth, Gender, PhoneNumber, Address, City, State, ZipCode, InsuranceName.
- Messages:** No messages displayed.
- Status Bar:** Ln1, Col 36, Spaces: 4, UTF-8, LF, SQL, 50 rows, MSSQL, 00:00:00, localhost:UniversityMedicalCenter

Retrieved 50 rows.

View that shows all patients and their corresponding lab results:

CREATE VIEW PatientLabResults AS

```
SELECT p.FirstName, p.LastName, l.TestName, l.TestDate, l.TestResults
FROM Patients p
INNER JOIN LabResults l ON p.PatientID = l.PatientID;
```

select * from PatientLabResults;

The screenshot shows the SSMS interface with the following details:

- Object Explorer (Left):** Shows the database structure under "UniversityMedicalCenter". The "Views" node is selected.
- SQL Query Editor (Top):** Contains the query: `select * from PatientLabResults;`
- Results Grid (Bottom):** Displays 10 rows of data from the PatientLabResults view.
- Status Bar (Bottom):** Shows "Ln 1, Col 33" and "10 rows".

	FirstName	LastName	TestName	TestDate	TestResults
1	John	Doe	Blood sugar test	2023-04-29	120 mg/dL
2	Jane	Smith	Urinalysis	2023-04-28	Normal
3	David	Lee	Complete blood count	2023-04-27	Normal
4	Lisa	Nguyen	Electrocardiogram	2023-04-26	Sinus rhythm
5	Michael	Garcia	Stool culture	2023-04-25	Positive for Salmonella
6	Emily	Kim	Thyroid function test	2023-04-24	Normal
7	William	Johnson	Liver function test	2023-04-23	Elevated ALT levels
8	Sarah	Martinez	Kidney function test	2023-04-22	Normal
9	Daniel	Rodriguez	Lipid panel	2023-04-21	Elevated LDL cholesterol
10	Megan	Chen	Pregnancy test	2023-04-20	Positive

Retrieved 10 rows.

View that shows all patients and their corresponding procedures:

```
CREATE VIEW PatientProcedures AS
```

```
SELECT p.FirstName, p.LastName, pr.ICD10CodeID, pr.ProcedureDate
```

```
FROM Patients p
```

```
INNER JOIN PatientPhysician pp ON p.PatientID = pp.PatientID
```

```
INNER JOIN Procedures pr ON pp.PatientPhysicianID = pr.PatientPhysicianID;
```

```
select * from PatientMedications;
```

The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the database structure, including servers, databases, tables, and views.
- Query Editor:** Contains the SQL query: `select * from PatientMedications;`
- Results Grid:** Displays the output of the query with 30 rows of data.
- Status Bar:** Shows the following information: Line 1, Col 34, Spaces: 4, UTF-8, LF, SQL, 30 rows, MSSQL, 00:00:00, localhost : UniversityMedicalCenter.

	FirstName	LastName	MedicationName	Dosage	Frequency
1	John	Doe	Lisinopril	20mg	Daily
2	Jane	Smith	Metformin	500mg	Twice daily
3	David	Lee	Lевотироксин	75mcg	Daily
4	Lisa	Nguyen	Atorvastatin	40mg	Nightly
5	Michael	Garcia	Albuterol	90mcg	As needed
6	Emily	Kim	Ibuprofen	800mg	Every 8 hours
7	William	Johnson	Hydrocodone	10mg	As needed
8	Sarah	Martinez	Trazadone	50mg	Nightly
9	Daniel	Rodriguez	Omeprazole	20mg	Daily
10	Megan	Chen	Aspirin	81mg	Daily
11	John	Doe	Lisinopril	20mg	Daily
12	Jane	Smith	Metformin	1000mg	Twice daily
13	David	Lee	Левотироксин	75mcg	Daily
14	Lisa	Nguyen	Atorvastatin	40mg	Nightly
15	Michael	Garcia	Albuterol	90mcg	As needed
16	Emily	Kim	Ibuprofen	800mg	Every 8 hours
17	William	Johnson	Hydrocodone	10mg	As needed
18	Sarah	Martinez	Trazadone	50mg	Nightly
19	Daniel	Rodriguez	Omeprazole	20mg	Daily
20	Megan	Chen	Aspirin	81mg	Daily
21	John	Doe	Lisinopril	20mg	Daily
22	Jane	Smith	Metformin	1000mg	Twice daily
23	David	Lee	Левотироксин	75mcg	Daily
24	Lisa	Nguyen	Atorvastatin	40mg	Nightly
25	Michael	Garcia	Albuterol	90mcg	As needed
26	Emily	Kim	Ibuprofen	800mg	Every 8 hours
27	William	Johnson	Hydrocodone	10mg	As needed
28	Sarah	Martinez	Trazadone	50mg	Nightly
29	Daniel	Rodriguez	Omeprazole	20mg	Daily
30	Megan	Chen	Aspirin	81mg	Daily

Retrieved 30 rows.

View that displays patient information along with their insurance information:

CREATE VIEW PatientInsuranceInfo AS

```
SELECT p.PatientID, p.FirstName, p.LastName, p.DateOfBirth, p.Gender, p.PhoneNumber,
p.Address, p.City, p.State, p.ZipCode, i.InsuranceName, i.InsuranceNumber,
i.InsurancePlanType, i.InsuranceEffectiveDate, i.InsuranceExpirationDate
FROM Patients p
JOIN Insurances i ON p.InsuranceID = i.InsuranceID;
```

select * from PatientProcedures;

The screenshot shows the SSMS interface with the following details:

- Object Explorer (Left):** Shows the database structure under "UniversityMedicalCenter" database, including "Tables" and "Views".
- Query Editor (Top):** Displays the query: `select * from PatientProcedures;`
- Results Grid (Center):** Shows the output of the query with 10 rows of data.
- Status Bar (Bottom):** Provides information about the session: Line 1, Col 33, Spaces: 4, UTF-8, LF, SQL, 10 rows, MSSQL, 00:00:00, localhost:UniversityMedicalCenter.

	FirstName	LastName	ICD10CodeID	ProcedureDate
1	John	Doe	1	2022-03-15
2	Jane	Smith	2	2022-04-28
3	David	Lee	3	2022-05-18
4	Lisa	Nguyen	4	2022-06-01
5	Michael	Garcia	5	2022-07-04
6	Emily	Kim	6	2022-08-15
7	William	Johnson	7	2022-09-21
8	Sarah	Martinez	8	2022-10-10
9	Daniel	Rodriguez	9	2022-11-02
10	Megan	Chen	10	2022-12-25

Retrieved 10 rows.

View that displays physician information along with their department and license information:

CREATE VIEW PhysicianInfo AS

```
SELECT phy.PhysicianID, phy.FirstName, phy.LastName, phy.DateOfBirth, phy.Gender,
phy.PhoneNumber, l.LicenseName, l.Description, d.DepartmentName,
d.DepartmentDescription
FROM Physicians phy
JOIN LegalLicenses l ON phy.LicenseID = l.LicenseID
JOIN HealthDepartments d ON phy.DepartmentID = d.DepartmentID;
```

select * from PhysicianInfo;

The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the database structure with servers, databases, tables, and views.
- Query Editor:** Contains the T-SQL command: `select * from PhysicianInfo;`
- Status Bar:** Shows the number of rows (40), file size (40 rows), and connection details (MSSQL 00:00:00 localhost\UniversityMedicalCenter).

PhysicianID	FirstName	LastName	DateOfBirth	Gender	PhoneNumber	LicenseName	LicenseDescription	DepartmentName
8	Liam	Chen	1980-12-02	M	555-555-1919	Psychologist License	This license allows a per..	Neurology
9	Emma	Gonzalez	1982-01-15	F	555-555-2020	Veterinary License	This license allows a per..	Cardiology
10	Mason	Perez	1974-09-30	M	555-555-2121	EMT License	This license allows a per..	Gastroenterology
11	Sophia	Lee	1986-09-12	F	555-555-1212	Medical License	This license allows a per..	Obstetrics and Gy..
12	Ethan	Nguyen	1978-11-23	M	555-555-1313	Nursing License	This license allows a per..	Dermatology
13	Ava	Garcia	1991-02-01	F	555-555-1414	Pharmacy License	This license allows a per..	Otolaryngology
14	Jacob	Kim	1983-05-31	M	555-555-1515	Dental License	This license allows a per..	Psychiatry
15	Isabella	Johnson	1988-08-10	F	555-555-1616	Physical Therapy License	This license allows a per..	Pediatrics
16	Noah	Martinez	1985-03-17	M	555-555-1717	Occupational Therapy Lice..	This license allows a per..	Ophthalmology
17	Olivia	Rodriguez	1993-07-28	F	555-555-1818	Chiropractic License	This license allows a per..	Orthopedics
18	Liam	Chen	1980-12-02	M	555-555-1919	Psychologist License	This license allows a per..	Neurology
19	Emma	Gonzalez	1982-01-15	F	555-555-2020	Veterinary License	This license allows a per..	Cardiology
20	Mason	Perez	1974-09-30	M	555-555-2121	EMT License	This license allows a per..	Gastroenterology
21	Sophia	Lee	1986-09-12	F	555-555-1212	Medical License	This license allows a per..	Obstetrics and Gy..
22	Ethan	Nguyen	1978-11-23	M	555-555-1313	Nursing License	This license allows a per..	Dermatology
23	Ava	Gar Nguyen	1991-02-01	F	555-555-1414	Pharmacy License	This license allows a per..	Otolaryngology
24	Jacob	Kim	1983-05-31	M	555-555-1515	Dental License	This license allows a per..	Psychiatry
25	Isabella	Johnson	1988-08-10	F	555-555-1616	Physical Therapy License	This license allows a per..	Pediatrics
26	Noah	Martinez	1985-03-17	M	555-555-1717	Occupational Therapy Lice..	This license allows a per..	Ophthalmology
27	Olivia	Rodriguez	1993-07-28	F	555-555-1818	Chiropractic License	This license allows a per..	Orthopedics
28	Liam	Chen	1980-12-02	M	555-555-1919	Psychologist License	This license allows a per..	Neurology
29	Emma	Gonzalez	1982-01-15	F	555-555-2020	Veterinary License	This license allows a per..	Cardiology
30	Mason	Perez	1974-09-30	M	555-555-2121	EMT License	This license allows a per..	Gastroenterology
31	Sophia	Lee	1986-09-12	F	555-555-1212	Medical License	This license allows a per..	Obstetrics and Gy..
32	Ethan	Nguyen	1978-11-23	M	555-555-1313	Nursing License	This license allows a per..	Dermatology
33	Ava	Garcia	1991-02-01	F	555-555-1414	Pharmacy License	This license allows a per..	Otolaryngology
34	Jacob	Kim	1983-05-31	M	555-555-1515	Dental License	This license allows a per..	Psychiatry
35	Isabella	Johnson	1988-08-10	F	555-555-1616	Physical Therapy License	This license allows a per..	Pediatrics
36	Noah	Martinez	1985-03-17	M	555-555-1717	Occupational Therapy Lice..	This license allows a per..	Ophthalmology
37	Olivia	Rodriguez	1993-07-28	F	555-555-1818	Chiropractic License	This license allows a per..	Orthopedics
38	Liam	Chen	1980-12-02	M	555-555-1919	Psychologist License	This license allows a per..	Neurology
39	Emma	Gonzalez	1982-01-15	F	555-555-2020	Veterinary License	This license allows a per..	Cardiology
40	Mason	Perez	1974-09-30	M	555-555-2121	EMT License	This license allows a per..	Gastroenterology

Retrieved 40 rows.

Views created after successful execution of above code.

```

    <--> Views
        > dbo.PatientBilling
        > dbo.PatientInsuranceInfo
        > dbo.PatientLabResults
        > dbo.PatientMedications
        > dbo.PatientProcedures
        > dbo.PhysicianInfo
    
```

Execution of Views

```

CREATE VIEW PatientMedications AS
SELECT p.FirstName, p.LastName, m.MedicationName, m.Dosage, m.Frequency
FROM Patients p
INNER JOIN Medications m ON p.PatientID = m.PatientID;

CREATE VIEW PatientBilling AS
SELECT p.FirstName, p.LastName, b.Amount, b.BillingDate
FROM Patients p
INNER JOIN Billing b ON p.PatientID = b.PatientID;

CREATE VIEW PatientLabResults AS
SELECT p.FirstName, p.LastName, l.TestName, l.TestDate, l.TestResults
FROM Patients p
INNER JOIN LabResults l ON p.PatientID = l.PatientID;

CREATE VIEW PatientProcedures AS
SELECT p.FirstName, p.LastName, pr.ICD10CodeID, pr.ProcedureDate
FROM Patients p
INNER JOIN PatientPhysician pp ON p.PatientID = pp.PatientID
INNER JOIN Procedures pr ON pp.PatientPhysicianID = pr.PatientPhysicianID;

CREATE VIEW PatientInsuranceInfo AS
SELECT p.PatientID, p.FirstName, p.LastName, p.DateOfBirth, p.Gender, p.PhoneNumber, p.Address, p.City, p.State, p.ZipCode, i.InsuranceName, i.InsuranceNumber, i.InsurancePolicy
FROM Patients p
JOIN Insurances i ON p.InsuranceID = i.InsuranceID;
    
```

Messages

6:01:43 PM Started executing query at Line 28
Commands completed successfully.
Total execution time: 00:00:00.011

Stored Procedures:

1. Stored Procedure for adding a new patient

```
CREATE PROCEDURE add_patient
    @FirstName VARCHAR(50),
    @LastName VARCHAR(50),
    @DateOfBirth DATE,
    @Gender CHAR(1),
    @PhoneNumber VARCHAR(20),
    @Address VARCHAR(100),
    @City VARCHAR(50),
    @State VARCHAR(50),
    @ZipCode VARCHAR(10),
    @InsuranceID INT
AS
BEGIN
    INSERT INTO Patients (FirstName, LastName, DateOfBirth, Gender, PhoneNumber, Address,
    City, State, ZipCode, InsuranceID)
    VALUES (@FirstName, @LastName, @DateOfBirth, @Gender, @PhoneNumber, @Address,
    @City, @State, @ZipCode, @InsuranceID)
END
```

CSE 581 INTRO TO DATABASE MANAGEMENT SYSTEM

```

CREATE PROCEDURE add_patient
    @FirstName VARCHAR(50),
    @LastName VARCHAR(50),
    @DateOfBirth DATE,
    @Gender CHAR(1),
    @PhoneNumber VARCHAR(20),
    @Address VARCHAR(100),
    @City VARCHAR(50),
    @State VARCHAR(50),
    @ZipCode VARCHAR(10),
    @InsuranceID INT
AS
BEGIN
    INSERT INTO Patients (FirstName, LastName, DateOfBirth, Gender, PhoneNumber, Address, City, State, ZipCode, InsuranceID)
    VALUES (@FirstName, @LastName, @DateOfBirth, @Gender, @PhoneNumber, @Address, @City, @State, @ZipCode, @InsuranceID)
END

```

Messages

6:29:17 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.046

`EXEC add_patient 'John', 'Doe', '1990-05-15', 'M', '555-555-5555', '123 Main St', 'Anytown', 'CA', '12345', 1`

```

INSERT INTO Patients (FirstName, LastName, DateOfBirth, Gender, PhoneNumber, Address, City, State, ZipCode, InsuranceID)
VALUES (@FirstName, @LastName, @DateOfBirth, @Gender, @PhoneNumber, @Address, @City, @State, @ZipCode, @InsuranceID)

```

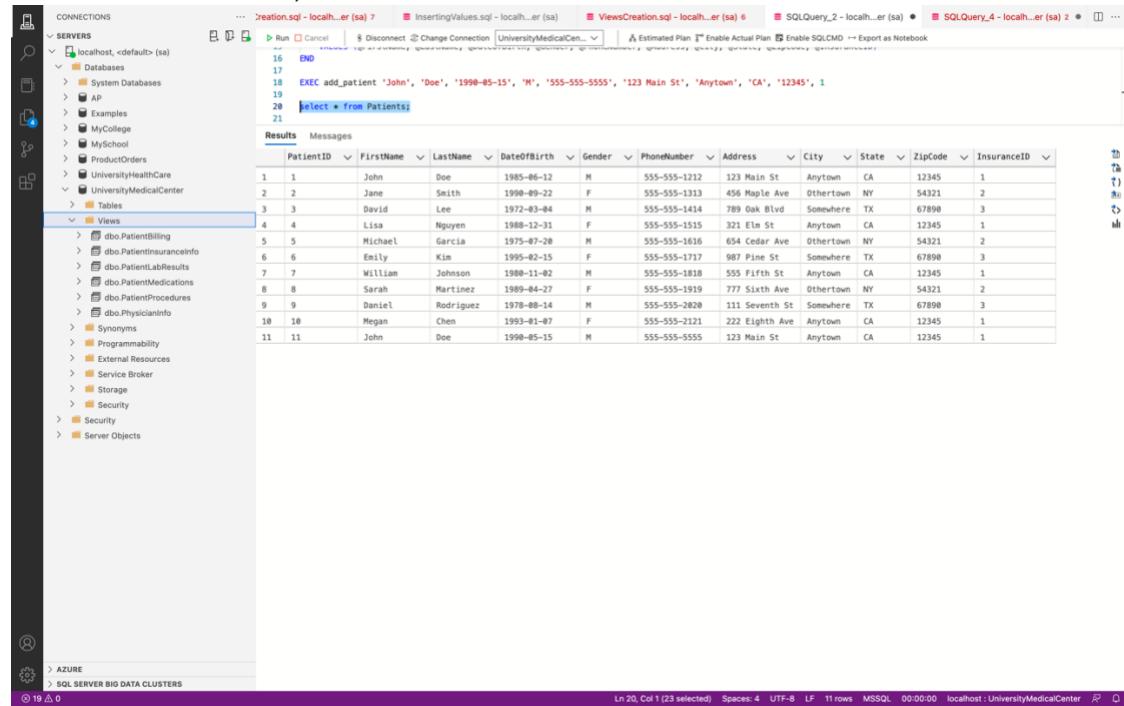
Messages

6:31:20 PM Started executing query at Line 18
(1 row affected)
Total execution time: 00:00:00.032

Inserted a patient record using the stored procedure

Added 1 row (new patient using the stored procedure)

`select * from Patients;`



The screenshot shows the SSMS interface with the following details:

- Connections:** localhost, <default> (sa)
- Databases:** UniversityMedicalCenter
- Tables:** Patients
- Views:** (empty)
- Results:** A grid showing 11 rows of patient data.

The results grid has the following columns:

PatientID	FirstName	LastName	DateOfBirth	Gender	PhoneNumber	Address	City	State	ZipCode	InsuranceID
1	John	Doe	1985-06-12	M	555-555-1212	123 Main St	Anytown	CA	12345	1
2	Jane	Smith	1990-09-22	F	555-555-1313	456 Maple Ave	Othertown	NY	54321	2
3	David	Lee	1972-03-04	M	555-555-1414	789 Oak Blvd	Somewhere	TX	67890	3
4	Lisa	Nguyen	1988-12-31	F	555-555-1515	321 Elm St	Anytown	CA	12345	1
5	Michael	Garcia	1975-07-28	M	555-555-1616	654 Cedar Ave	Othertown	NY	54321	2
6	Emily	Kim	1995-02-15	F	555-555-1717	987 Pine St	Somewhere	TX	67890	3
7	William	Johnson	1988-11-02	M	555-555-1818	555 Fifth St	Anytown	CA	12345	1
8	Sarah	Martinez	1989-04-27	F	555-555-1919	777 Sixth Ave	Othertown	NY	54321	2
9	Daniel	Rodriguez	1978-08-14	M	555-555-2020	111 Seventh St	Somewhere	TX	67890	3
10	Megan	Chen	1993-01-07	F	555-555-2121	222 Eighth Ave	Anytown	CA	12345	1
11	John	Doe	1990-05-15	M	555-555-5555	123 Main St	Anytown	CA	12345	1

2. Stored Procedure for updating billing amount of a patient

```
CREATE PROCEDURE UpdateBillingAmount
    @PatientId INT,
    @Amount DECIMAL(10,2)
AS
BEGIN
    UPDATE Billing
    SET Amount = @Amount
    WHERE PatientID = @PatientId
END
```

CSE 581 INTRO TO DATABASE MANAGEMENT SYSTEM

The screenshot shows the SQL Server Management Studio (SSMS) interface. On the left, the Object Explorer displays the database structure for 'UniversityMedicalCenter'. In the center, the 'creation.sql' query editor contains the following T-SQL code:

```

CREATE PROCEDURE UpdateBillingAmount
    @PatientId INT,
    @Amount DECIMAL(10,2)
AS
BEGIN
    UPDATE Billing
    SET Amount = @Amount
    WHERE PatientID = @PatientId
END

```

Below the code, the 'Messages' pane shows the execution log:

```

6:46:32 PM Started executing query at Line 24.
Commands completed successfully.
Total execution time: 00:00:00.021

```

At the bottom, the status bar indicates: Ln 24, Col 1 (172 selected) Spaces: 4 UTF-8 LF 0 rows MSSQL 00:00:00 localhost:UniversityMedicalCenter

EXEC UpdateBillingAmount @patientID = 1, @amount = 100.50;

CSE 581 INTRO TO DATABASE MANAGEMENT SYSTEM

```

19
20
21
22
23
24 CREATE PROCEDURE UpdateBillingAmount
25     @PatientId INT,
26     @Amount DECIMAL(10,2)
27 AS
28 BEGIN
29     UPDATE Billing
30     SET Amount = @Amount
31     WHERE PatientID = @PatientId
32 END
33
34 EXEC UpdateBillingAmount @PatientID = 1, @amount = 100.50;
35
36
37
38
39
40

```

Messages

6:50:32 PM Started executing query at Line 34
(1 row affected)
Total execution time: 00:00:00.028

AZURE
SQL SERVER BIO DATA CLUSTERS

1 Row affected (updated billing amount of patient with PatientID 1).

3. Stored procedure for adding a note

```
CREATE PROCEDURE AddNote
```

```
    @PatientId INT,
    @PhysicianId INT,
    @NurseId INT,
    @NoteDate DATETIME,
    @NoteText NVARCHAR(MAX)
```

```
AS
```

```
BEGIN
```

```
    INSERT INTO Notes (PatientID, PhysicianID, NurseID, NoteDate, NoteText)
```

```
        VALUES (@PatientId, @PhysicianId, @NurseId, @NoteDate, @NoteText)
```

```
END
```

```
CREATE PROCEDURE AddNote
    @PatientId INT,
    @PhysicianId INT,
    @NurseId INT,
    @NoteDate DATETIME,
    @NoteText NVARCHAR(MAX)
AS
BEGIN
    INSERT INTO Notes (PatientID, PhysicianID, NurseID, NoteDate, NoteText)
        VALUES (@PatientId, @PhysicianId, @NurseId, @NoteDate, @NoteText)
END
```

Messages

6:52:54 PM Started executing query at Line 37
Commands completed successfully.
Total execution time: 00:00:00.036

```
EXEC AddNote
```

```
@PatientId = 3,
```

CSE 581 INTRO TO DATABASE MANAGEMENT SYSTEM

```

@PhysicianId = 2,
@NurseId = 3,
@NoteDate = '2023-04-30 10:30:00',
@NoteText = 'Patient is responding well to treatment.'

```

The screenshot shows the SSMS interface with the following details:

- Connections:** Localhost, <default> (sa)
- Servers:** localhost, <default> (sa) -> Databases, Tables, Views.
- Query Window:**

```

40      @NurseId INT,
41      @NoteDate DATETIME,
42      @NoteText NVARCHAR(MAX)
43  AS
44  BEGIN
45      INSERT INTO Notes (PatientID, PhysicianID, NurseID, NoteDate, NoteText)
46      VALUES (@PatientId, @PhysicianId, @NurseId, @NoteDate, @NoteText)
47  END
48
49
50
51
52  EXEC AddNote,
53      @PatientId = 3,
54      @PhysicianId = 2,
55      @NurseId = 3,
56      @NoteDate = '2023-04-30 10:30:00',
57      @NoteText = 'Patient is responding well to treatment.'
58
59
60
61

```
- Messages:**
 - Started executing query at Line 52
 - [1 row affected]
 - Total execution time: 00:00:00.008
- Bottom Status Bar:** Ln 52, Col 1 [T7] selected | Spaces: 4 | UTF-8 | LF | 0 rows | MSSQL | 00:00:00 | localhost : UniversityMedicalCenter

1 Row Affected (Added a note).

4. Stored procedure for getting all admissions for a specific patient.

```
CREATE PROCEDURE GetAdmissionsByPatientId
```

```
    @PatientId INT
```

```
AS
```

```
BEGIN
```

```
    SELECT * FROM Admissions WHERE PatientID = @PatientId
```

```
END
```

```

CREATE PROCEDURE GetAdmissionsByPatientId
    @PatientId INT
AS
BEGIN
    INSERT INTO Notes (PatientID, PhysicianID, NurseID, NoteDate, NoteText)
    VALUES (@PatientId, @PhysicianId, @NurseId, @NoteDate, @NoteText)
END
EXEC AddNote
@PatientId = 3,
@PhysicianId = 2,
@NurseId = 3,
@NoteDate = '2023-04-30 10:38:00',
@NoteText = 'Patient is responding well to treatment.'

```

Messages

6:57:44 PM Started executing query at Line 52
(1 row affected)
Total execution time: 00:00:00.008

[EXEC GetAdmissionsByPatientId @PatientId = 1;](#)

The screenshot shows the SSMS interface. On the left is the Object Explorer tree view, which includes sections for SERVERS, Databases, Tables, Views, and AZURE. The central area contains a query window with the following code:

```

65 SELECT * FROM Admissions WHERE PatientID = @PatientId
66 END
68 EXEC GetAdmissionsByPatientId @PatientId = 1;
69
70
71
72

```

Below the code, the Results tab displays a single row of data from the Admissions table:

AdmissionID	PatientID	PhysicianID	RoomID	DiagnosisID	AdmissionDate	DischargeDate
1	1	3	2	1	2022-04-01 12:00:00.000	2022-04-05 10:00:00.000

At the bottom of the screen, the status bar shows: Ln 68, Col 1 (45 selected) Spaces: 4 UTF-8 LF 1 rows MSSQL 00:00:00 localhost:UniversityMedicalCenter

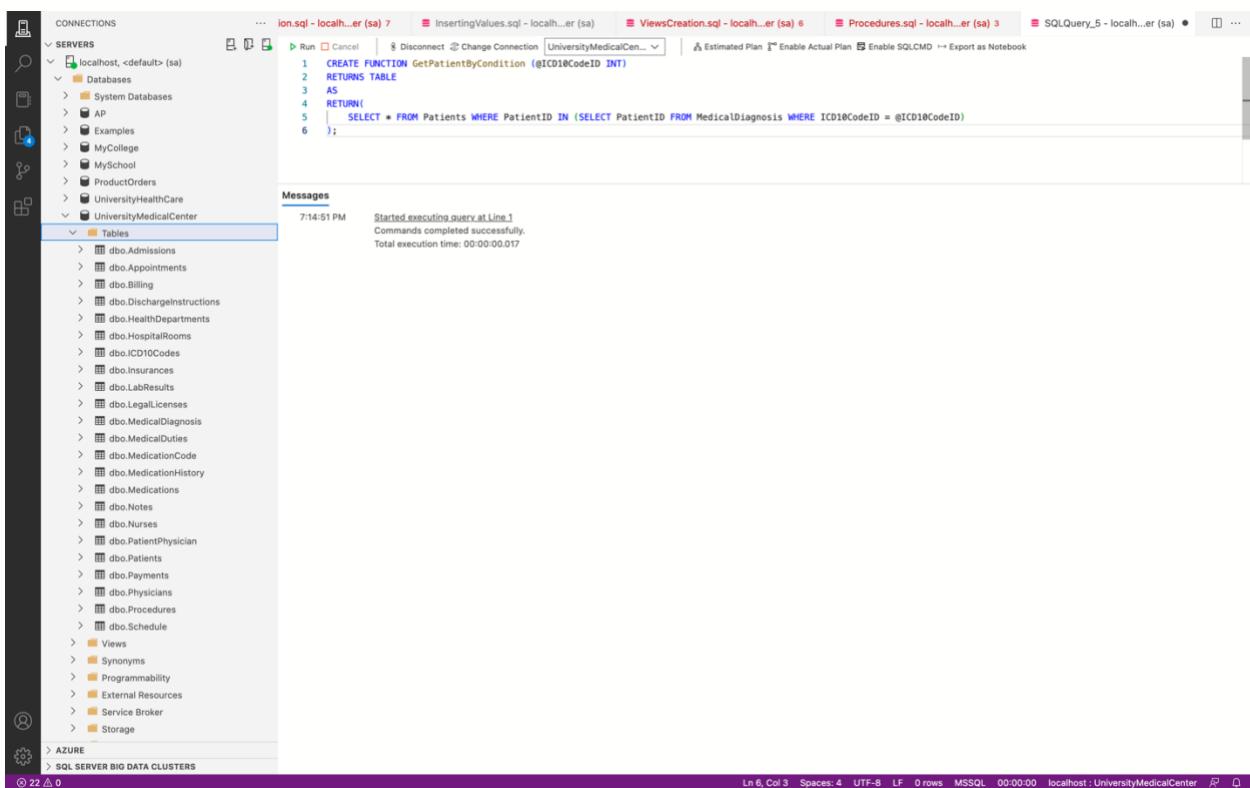
Created Stored Procedures

- ▽ **Stored Procedures**
 - dbo.add_patient
 - dbo.AddNote
 - dbo.GetAdmissionsByPatientId
 - dbo.UpdateBillingAmount

Functions

1. Function for getting patient details by condition.

```
CREATE FUNCTION GetPatientByCondition (@ICD10CodeID INT)
RETURNS TABLE
AS
RETURN(
    SELECT * FROM Patients WHERE PatientID IN (SELECT PatientID FROM MedicalDiagnosis
    WHERE ICD10CodeID = @ICD10CodeID)
);
```



```
SELECT * FROM dbo.GetPatientByCondition (5);
```

The screenshot shows the SSMS interface with the following details:

- Object Explorer (Left):** Shows the database structure under "UniversityMedicalCenter" including SERVERS, Databases, Tables, Views, Synonyms, Programmability, External Resources, Service Broker, and Storage.
- Query Editor (Top):** Displays the T-SQL code for creating a function and executing it with parameter value 5.


```

CREATE FUNCTION GetPatientByCondition (@ICD10CodeID INT)
RETURNS TABLE
AS
BEGIN
    SELECT * FROM Patients WHERE PatientID IN (SELECT PatientID FROM MedicalDiagnosis WHERE ICD10CodeID = @ICD10CodeID)
END;
SELECT * FROM dbo.GetPatientByCondition (5);
      
```
- Results Grid (Bottom):** Shows the output of the query, which is a single row of patient information.

	PatientID	FirstName	LastName	DateOfBirth	Gender	PhoneNumber	Address	City	State	ZipCode	InsuranceID
1	5	Michael	Garcia	1975-07-20	M	555-555-1616	654 Cedar Ave	OtherTown	NY	54321	2

Retrieved 1 row.

2. Get patient full name by ID.

```

CREATE FUNCTION GetPatientNameByID (@PatientID INT)
RETURNS VARCHAR(100)
AS
BEGIN
    DECLARE @FullName VARCHAR(100)

    SELECT @FullName = CONCAT(FirstName, ' ', LastName)
    FROM Patients
    WHERE PatientID = @PatientID

    RETURN @FullName
END

```

The screenshot shows the SSMS interface with the following details:

- Connections:** Procedures.sql - localhost\sa (sa)
- Servers:** localhost, <default> (sa)
- Databases:** System Databases, AP, Examples, MyCollege, MySchool, ProductOrders, UniversityHealthCare, UniversityMedicalCenter
- Tables:** dbo.Admissions, dbo.Appointments, dbo.Billing, dbo.DischargeInstructions, dbo.HealthDepartments, dbo.HospitalRooms, dbo.ICD10Codes, dbo.Insurances, dbo.LabResults, dbo.LegalLicenses, dbo.MedicalDiagnosis, dbo.MedicalDuties, dbo.MedicationCode, dbo.MedicationHistory, dbo.Medications, dbo.Notes, dbo.Nurses, dbo.PatientPhysician, dbo.Patients, dbo.Payments, dbo.Physicians, dbo.Procedures, dbo.Schedule, Views, Synonyms, Programmability, External Resources, Service Broker, Storage
- AZURE:** SQL SERVER BIO DATA CLUSTERS
- Messages:** Started executing query at Line 1. Commands completed successfully. Total execution time: 00:00:00.036
- Status Bar:** Ln 12, Col 4 Spaces: 4 UTF-8 LF 0 rows MSSQL 00:00:00 localhost : UniversityMedicalCenter

CSE 581 INTRO TO DATABASE MANAGEMENT SYSTEM

```
SELECT dbo.GetPatientNameByID(1);
```

The screenshot shows the SSMS interface with the following details:

- Connections:** Shows multiple connections including "Procedures.sql - local...er (sa) 3", "SQLQuery_5 - local...er (sa) 2", "SQLQuery_4 - local...er (sa) 2", "SQLQuery_2.sql - local...er (sa) 1", and "SQLQuery_3 - local...er (sa) 3".
- Servers:** Shows the "localhost, <default> (sa)" server with its databases: System Databases, AP, Examples, MyCollege, MySchool, ProductOrders, UniversityHealthCare, and UniversityMedicalCenter.
- Tables:** A list of tables under the UniversityMedicalCenter database, including Admissions, Appointments, Billing, DischargeInstructions, HealthDepartments, HospitalRooms, ICD10Codes, Insurances, LabResults, LegalLicenses, MedicalDiagnosis, MedicalDuties, MedicationCode, MedicationHistory, Medications, Notes, Nurses, PatientPhysician, Patients, Payments, Physicians, Procedures, Schedule, Views, Synonyms, Programmability, External Resources, Service Broker, and Storage.
- Query Editor:** Contains the following T-SQL code:

```

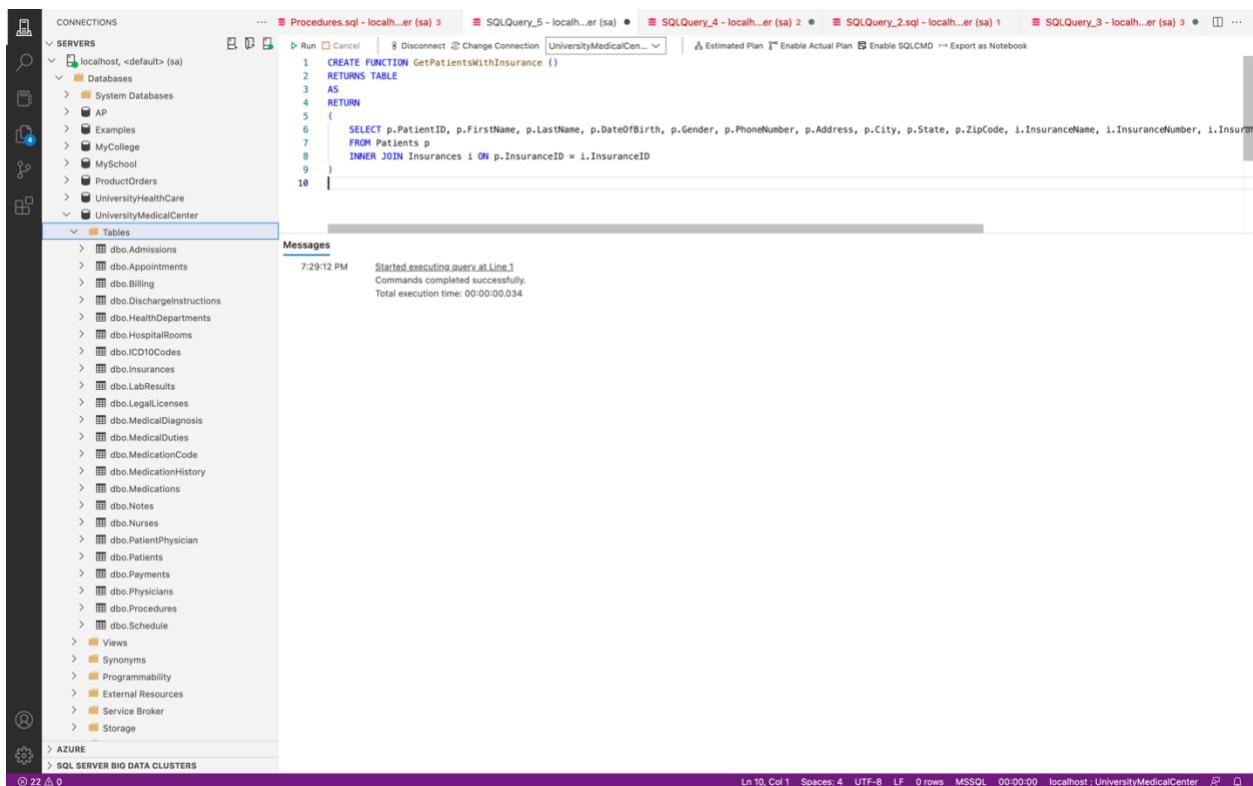
6
7   SELECT @fullName = CONCAT(FirstName, ' ', LastName)
8   FROM Patients
9   WHERE PatientID = @patientID
10
11  RETURN @fullName
12
13
14  SELECT dbo.GetPatientNameByID(1);

```
- Results:** The results pane displays the output of the query, showing a single row with the value "John Doe".
- Status Bar:** Shows "Ln 14, Col 1 (33 selected)", "Spaces: 4", "UTF-8", "LF", "1 rows", "MSSQL", "00:00:00", "localhost : UniversityMedicalCenter".

Retrieved 1 Row.

3. Function to get a list of all patients and their insurance information:

```
CREATE FUNCTION GetPatientsWithInsurance ()
RETURNS TABLE
AS
RETURN
(
    SELECT p.PatientID, p.FirstName, p.LastName, p.DateOfBirth, p.Gender, p.PhoneNumber,
    p.Address, p.City, p.State, p.ZipCode, i.InsuranceName, i.InsuranceNumber,
    i.InsurancePlanType, i.InsuranceEffectiveDate, i.InsuranceExpirationDate
    FROM Patients p
    INNER JOIN Insurances i ON p.InsuranceID = i.InsuranceID
)
```



CSE 581 INTRO TO DATABASE MANAGEMENT SYSTEM

```
SELECT * FROM GetPatientsWithInsurance();
```

PatientID	FirstName	LastName	DateOfBirth	Gender	PhoneNumber	Address	City	State	ZipCode	InsuranceName
1	John	Doe	1985-06-12	M	555-555-1212	123 Main St	Anytown	CA	12345	Blue Cross Blue Shield
2	Jane	Smith	1990-09-22	F	555-555-1313	456 Maple Ave	Othertown	NY	54321	Aetna
3	David	Lee	1972-03-04	M	555-555-1414	789 Oak Blvd	Somewhere	TX	67890	Cigna
4	Lisa	Nguyen	1988-12-31	F	555-555-1515	321 Elm St	Anytown	CA	12345	Blue Cross Blue Shield
5	Michael	Garcia	1975-07-20	M	555-555-1616	654 Cedar Ave	Othertown	NY	54321	Aetna
6	Emily	Kim	1995-02-15	F	555-555-1717	987 Pine St	Somewhere	TX	67890	Cigna
7	William	Johnson	1988-11-02	M	555-555-1818	555 Fifth St	Anytown	CA	12345	Blue Cross Blue Shield
8	Sarah	Martinez	1989-04-27	F	555-555-1919	777 Sixth Ave	Othertown	NY	54321	Aetna
9	Daniel	Rodriguez	1978-08-14	M	555-555-2020	111 Seventh St	Somewhere	TX	67890	Cigna
10	Megan	Chen	1993-01-07	F	555-555-2121	222 Eighth Ave	Anytown	CA	12345	Blue Cross Blue Shield
11	John	Doe	1990-05-15	M	555-555-5555	123 Main St	Anytown	CA	12345	Blue Cross Blue Shield

4. Function to get billing amount.

```
CREATE FUNCTION GetBillingAmount(@date DATE)
RETURNS DECIMAL(10,2)
AS
BEGIN
DECLARE @total DECIMAL(10,2);
SELECT @total = SUM(Amount)
FROM Billing
WHERE BillingDate = @date;
RETURN @total;
END;
```

The screenshot shows the SSMS interface with the following details:

- Left pane (Object Explorer):** Shows the server tree under 'localhost, <default> (sa)' with databases like System Databases, AP, Examples, MyCollege, MySchool, ProductOrders, UniversityHealthCare, and UniversityMedicalCenter. Under UniversityMedicalCenter, there is a 'Tables' node expanded, listing various tables such as Admissions, Appointments, Billing, DischargeInstructions, HealthDepartments, HospitalRooms, ICD10Codes, Insurances, LabResults, LegalLicenses, MedicalDiagnosis, MedicalDuties, MedicationCode, MedicationHistory, Medications, Notes, Nurses, PatientPhysician, Patients, Payments, Physicians, Procedures, Schedule, Views, Synonyms, Programmability, External Resources, Service Broker, and Storage.
- Center pane (Query Editor):** Displays the T-SQL code for creating the function:

```

1 CREATE FUNCTION GetBillingAmount(@date DATE)
2 RETURNS DECIMAL(10,2)
3 AS
4 BEGIN
5     DECLARE @total DECIMAL(10,2);
6     SELECT @total = SUM(Amount)
7     FROM Billing
8     WHERE BillingDate = @date;
9     RETURN @total;
10 END;
11

```

- Bottom right pane (Messages):** Shows the execution log:
 - Started executing query at Line 1
 - Commands completed successfully.
 - Total execution time: 00:00:00.034
- Bottom status bar:** Shows 'Ln 11, Col 1 (204 selected)', 'Spaces: 4', 'UTF-8', 'LF', '0 rows', 'MSSQL', '00:00:00', 'localhost : UniversityMedicalCenter', and icons for copy, paste, and refresh.

```
SELECT dbo.GetBillingAmount('2023-04-01') AS TotalBillingAmount;
```

The screenshot shows the SSMS interface with the following details:

- Connections:** localhost, <default> (sa)
- Databases:** MyCollege, MySchool, ProductOrders, UniversityHealthCare, UniversityMedicalCenter
- Tables:** Admissions, Appointments, Billing (selected), DischargeInstructions, HealthDepartments, HospitalRooms, ICD10Codes, Insurances, LabResults, LegalLicenses, MedicalDiagnosis, MedicalDuties, MedicationCode, MedicationHistory, Medications, Notes, Nurses, PatientPhysician
- Billing Table Columns:** BillingID (PK, int, not null), PatientID (FK, int, not null), PaymentID (FK, int, not null), BillingDate (date, not null), Amount (decimal(10,2), not null)
- Buttons:** Run, Cancel, Disconnect, Change Connection, UniversityMedicalCen..., Estimated Plan, Enable Actual Plan, Enable SQLCMD, Export as Notebook.
- Results:** Shows a single row with TotalBillingAmount = 100.50.
- Status Bar:** Ln 1, Col 65, Spaces: 4, UTF-8, LF, 1 rows, MSSQL, 00:00:00, localhost : UniversityMedicalCenter, 0.

The created Functions

The Object Explorer tree view shows the following structure:

- Functions** (selected)
 - Table-valued Functions**
 - > `dbo.GetPatientByCondition`
 - > `dbo.GetPatientsWithInsurance`
 - Scalar-valued Functions**
 - > `dbo.GetBillingAmount`
 - > `dbo.GetPatientNameByID`

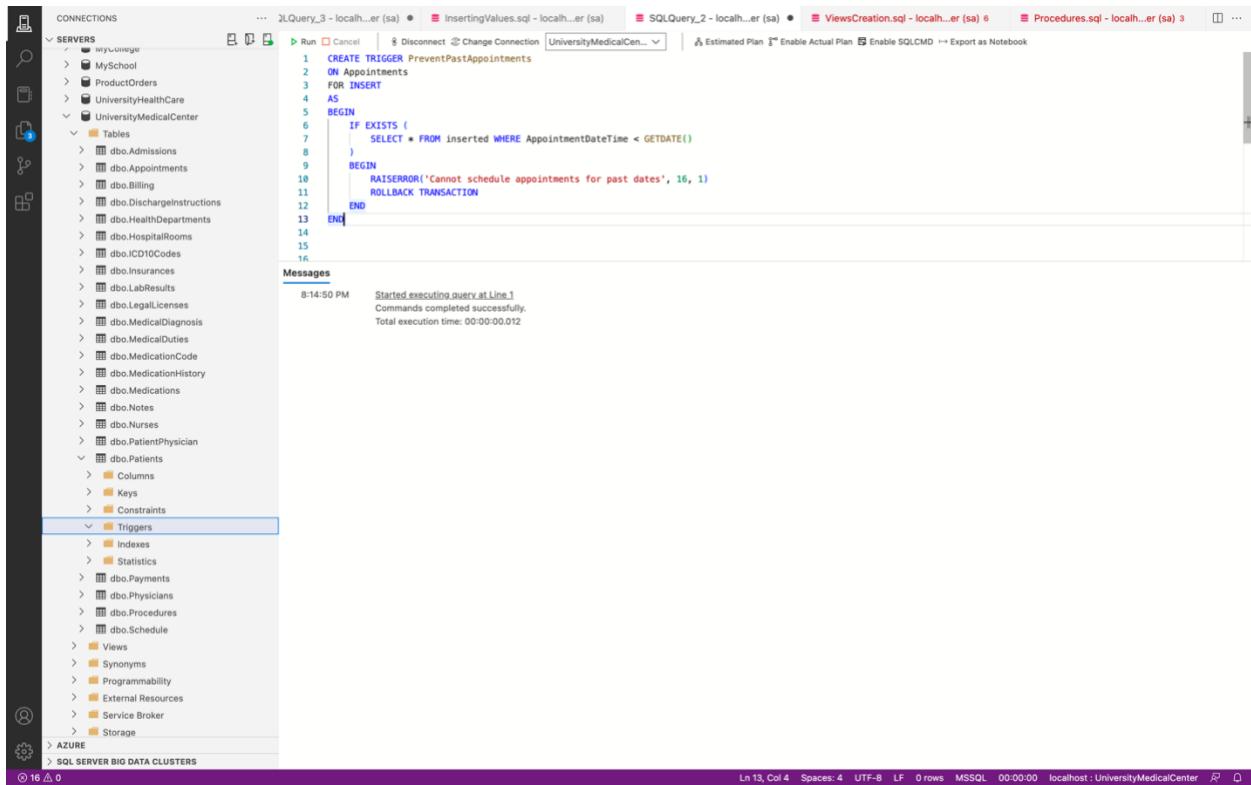
Triggers

1. Triggers for preventing past appointments.

```

CREATE TRIGGER PreventPastAppointments
ON Appointments
FOR INSERT
AS
BEGIN
    IF EXISTS (
        SELECT * FROM inserted WHERE AppointmentDateTime < GETDATE()
    )
    BEGIN
        RAISERROR('Cannot schedule appointments for past dates', 16, 1)
        ROLLBACK TRANSACTION
    END
END

```



```
INSERT INTO Appointments(PatientPhysicianID, AppointmentDateTime, Reason)
VALUES(1,GETDATE(), 'Covid-19 Symptoms');
```

The screenshot shows the SSMS interface. On the left is the Object Explorer tree view, which includes a 'Servers' node under 'MySCHOOL' containing 'UniversityMedicalCenter' and its tables (Admissions, Appointments, Billing, DischargeInstructions, HealthDepartments, HospitalRooms, ICD10Codes, Insurances, LabResults, LegalLicenses, MedicalDiagnosis, MedicalDuties, MedicationCode, MedicationHistory, Medications, Notes, Nurses, PatientPhysician, Patients), constraints, triggers, and other objects like Payments, Physicians, Procedures, Schedule, Views, Synonyms, Programmability, External Resources, Service Broker, and Storage. There is also an 'AZURE' node and a 'SQL SERVER BIO DATA CLUSTERS' node.

The main window displays a T-SQL script:

```

5 BEGIN
6 IF EXISTS (
7     SELECT * FROM inserted WHERE AppointmentDateTime < GETDATE()
8 )
9 BEGIN
10     RAISERROR('Cannot schedule appointments for past dates', 16, 1)
11     ROLLBACK TRANSACTION
12 END
13 END
14
15 INSERT INTO Appointments(PatientPhysicianID, AppointmentDateTime, Reason)
16 VALUES(1,GETDATE(), 'Covid-19 Symptoms');
17
18
19
~n
```

Below the script, the 'Messages' pane shows the execution results:

```

8:19:20 PM Started executing query at Line 15
Msg 50000, Level 16, State 1, Procedure PreventPastAppointments, Line 10
Cannot schedule appointments for past dates
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
Total execution time: 00:00:00.049
```

At the bottom, the status bar indicates: Ln 15, Col 1 (115 selected) Spaces: 4 UTF-8 LF 0 rows MSSQL 00:00:00 localhost:UniversityMedicalCenter

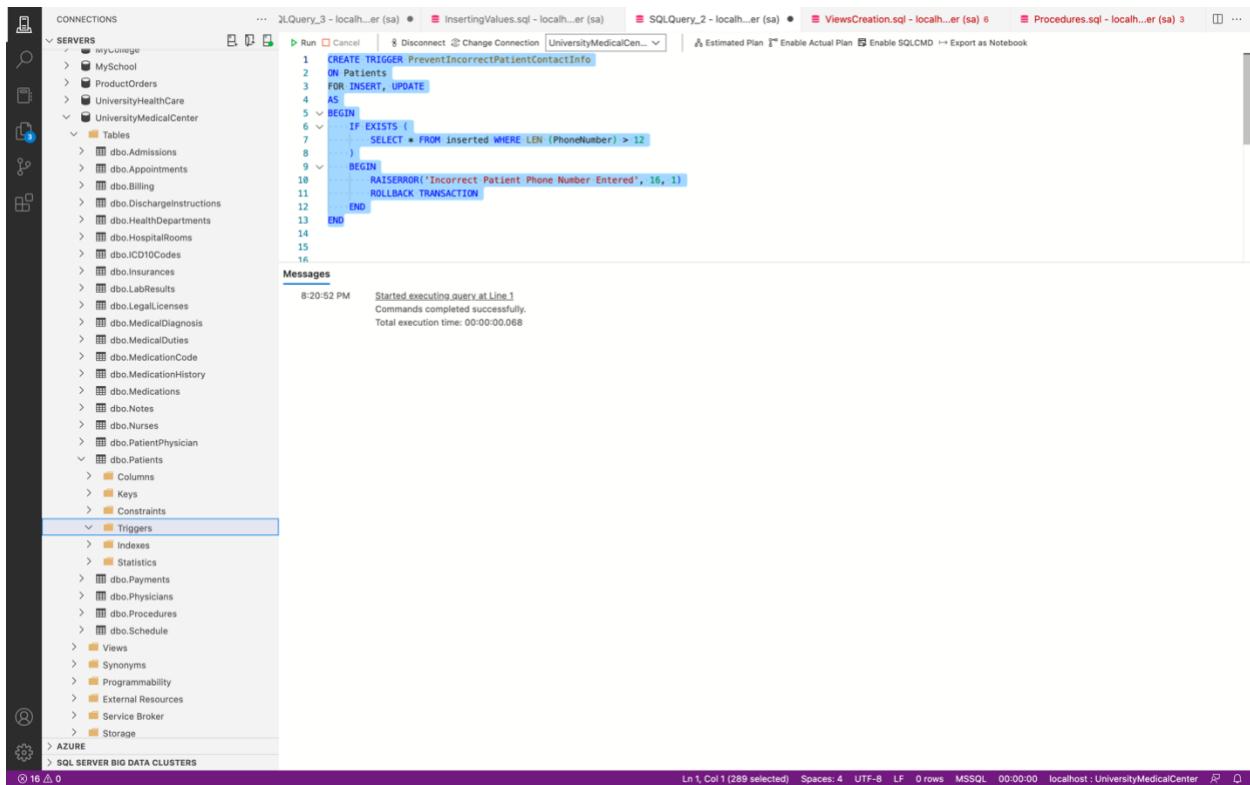
We got the error as expected.

2. Trigger for entering incorrect phone number

```

CREATE TRIGGER PreventIncorrectPatientContactInfo
ON Patients
FOR INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT * FROM inserted WHERE LEN (PhoneNumber) > 12
    )
    BEGIN
        RAISERROR('Incorrect Patient Phone Number Entered', 16, 1)
        ROLLBACK TRANSACTION
    END
END

```



```
INSERT INTO Patients (FirstName, LastName, DateOfBirth, Gender, PhoneNumber, Address, City, State, ZipCode, InsuranceID)
VALUES ('Ram', 'kanna', '1985-05-12', 'M', '3155-555-1234', '123 Main St', 'Syr', 'CA', '12345', 1)
```

The screenshot shows the SSMS interface with the following details:

- Connections:** Shows a connection to "UniversityMedicalCenter" (sa).
- Servers:** Shows "MySchool" and its objects.
- Tables:** Shows various tables under "dbo" such as Admissions, Appointments, Billing, DischargeInstructions, HealthDepartments, HospitalRooms, ICD10Codes, Insurances, LabResults, LegalLicenses, MedicalDiagnosis, MedicalDuties, MedicationCode, MedicationHistory, Medications, Notes, Nurses, PatientPhysician, and Patients.
- Triggers:** Shows triggers like Payments, Physicians, Procedures, Schedule, Views, Synonyms, Programmability, External Resources, Service Broker, and Storage.
- Azure:** Shows "SQL SERVER BIO DATA CLUSTERS".
- Messages:** Displays the following error message:


```
Started executing query at Line 1
Msg 50000, Level 16, State 1, Procedure PreventIncorrectPatientContactInfo, Line 10
Incorrect Patient Phone Number Entered
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.

Total execution time: 00:00:00.036
```
- Status Bar:** Shows "Ln 2, Col 100 Spaces: 4 UTF-8 LF 0 rows MSSQL 00:00:00 localhost : UniversityMedicalCenter".

Error occurred as expected.

3. Trigger for getting notes with no text

```

CREATE TRIGGER NoTextNotes
ON Notes
AFTER INSERT
AS
BEGIN
    IF EXISTS (SELECT * FROM inserted WHERE NoteText IS NULL OR NoteText = '')
    BEGIN
        RAISERROR('NoteText cannot be empty', 16, 1);
        ROLLBACK TRANSACTION;
        RETURN;
    END
END

```

The screenshot shows the SSMS interface with the following details:

- CONNECTIONS:** Localhost - local...er (sa)
- SERVERS:** myUnive...
- Tables:** dbo.Admissions, dbo.Appointments, dbo.Billing, dbo.DischargeInstructions, dbo.HealthDepartments, dbo.HospitalRooms, dbo.ICD10Codes, dbo.Insurances, dbo.LabResults, dbo.LegalLicenses, dbo.MedicalDiagnosis, dbo.MedicalDuties, dbo.MedicationCode, dbo.MedicationHistory, dbo.Medications, dbo.Notes, dbo.Nurses, dbo.PatientPhysician, dbo.Patients, dbo.Payments, dbo.Physicians, dbo.Procedures, dbo.Schedule, dbo.Views, dbo.Synonyms, dbo.Programmability, dbo.ExternalResources, dbo.ServiceBroker, dbo.Storage.
- Triggers:** NoTextNotes (selected)
- Code:**

```

1 CREATE TRIGGER NoTextNotes
2 ON Notes
3 AFTER INSERT
4 AS
5 BEGIN
6     IF EXISTS (SELECT * FROM inserted WHERE NoteText IS NULL OR NoteText = '')
7     BEGIN
8         RAISERROR('NoteText cannot be empty', 16, 1);
9         ROLLBACK TRANSACTION;
10    RETURN;
11 END
12 END
13

```
- Messages:**
 - Started executing query at Line 1
 - Commands completed successfully.
 - Total execution time: 00:00:00.014
- Status Bar:** Ln 12, Col 4 Spaces: 4 UTF-8 LF 0 rows MSSQL 00:00:00 localhost : UniversityMedicalCenter

**INSERT INTO Notes (PatientID, PhysicianID, NurseID, NoteDate, NoteText) VALUES
(3, 2, 3, '2023-04-29 09:30:00', "");**

The screenshot shows the SSMS interface. On the left is the Object Explorer tree view, which includes connections, servers (MySchool, ProductOrders, UniversityHealthCare, UniversityMedicalCenter), tables (Admissions, Appointments, Billing, DischargeInstructions, HealthDepartments, HospitalRooms, ICD10Codes, Insurances, LabResults, LegalLicenses, MedicalDiagnosis, MedicalDuties, MedicationCode, MedicationHistory, Medications, Notes, Nurses, PatientPhysician, Patients, Triggers), and various system objects like Columns, Keys, Constraints, and Storage. The right side contains a query editor window with the following code:

```

6 IF EXISTS (SELECT * FROM inserted WHERE NoteText IS NULL OR NoteText = '')
7 BEGIN
8     RAISERROR('NoteText cannot be empty', 16, 1);
9     ROLLBACK TRANSACTION;
10    RETURN;
11 END
12
13
14
15 INSERT INTO Notes (PatientID, PhysicianID, NurseID, NoteDate, NoteText) VALUES
16 (3, 2, 3, '2023-04-29 09:30:00', '');
17
18

```

Below the query editor is the 'Messages' pane, which displays the execution log:

```

Started executing query at Line 15
Msg 50000, Level 16, State 1, Procedure NoteTextNotes, Line 8
NoteText cannot be empty
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
Total execution time: 00:00:00.042

```

At the bottom of the interface, there are status indicators: Line 15, Col 1 (117 selected), Spaces: 4, UTF-8, LF, 0 rows, MSSQL, 00:00:00, localhost:UniversityMedicalCenter.

Error occurred As expected.

4. Trigger to prevent a diagnosis with future date

```

CREATE TRIGGER PreventFutureDateDiagnosis
ON MedicalDiagnosis
AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT * FROM inserted
        WHERE DiagnosisDate > GETDATE()
    )
    BEGIN
        RAISERROR('Cannot insert or update a diagnosis with a future date.', 16, 1)
        ROLLBACK TRANSACTION
    END
END

```

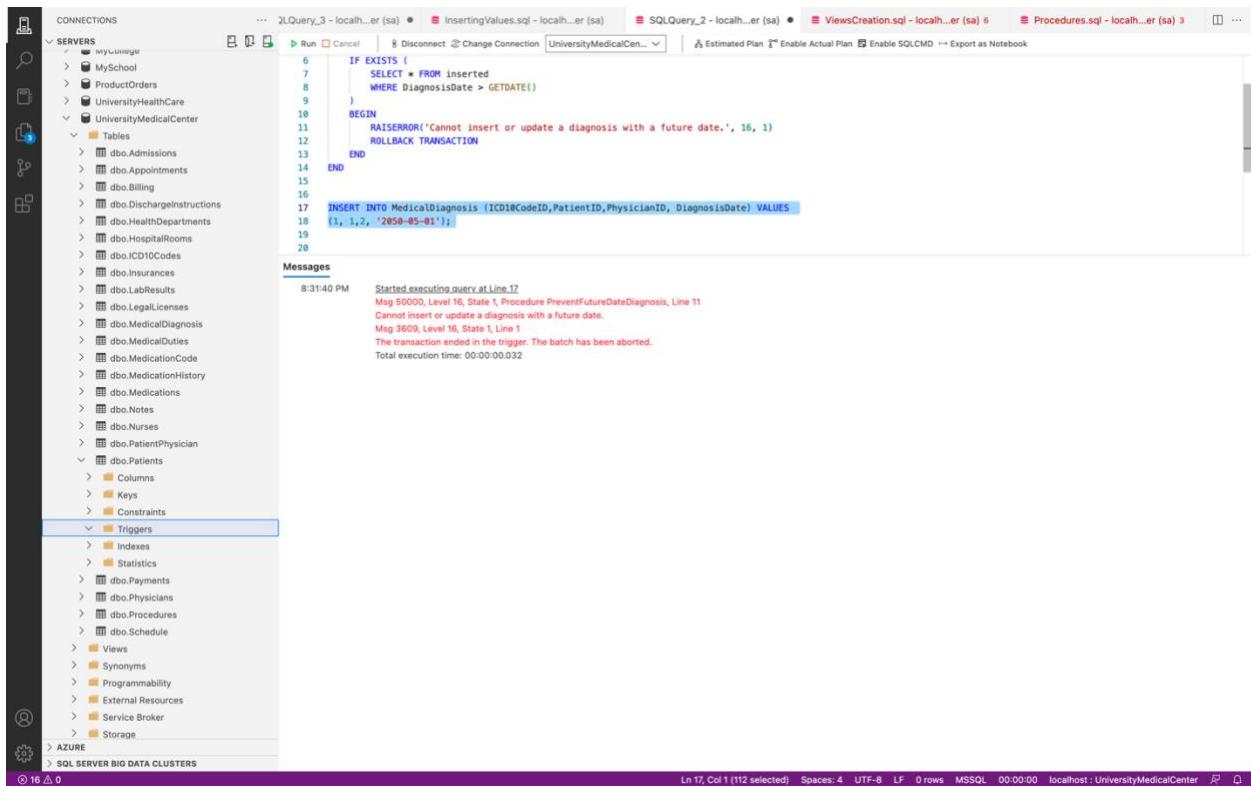
The screenshot shows the SQL Server Management Studio (SSMS) interface. In the Object Explorer on the left, under the 'UniversityMedicalCenter' database, the 'Triggers' node is selected. In the main results pane, the T-SQL code for creating the trigger is displayed. The code defines a trigger on the 'MedicalDiagnosis' table that fires after an insert or update. It checks if there are any rows in the inserted table where the 'DiagnosisDate' is greater than the current date. If such rows exist, it raises an error and rolls back the transaction. The 'Messages' pane at the bottom shows that the command was started at 8:29:44 PM, completed successfully, and took 0:00:00.037.

```

CREATE TRIGGER PreventFutureDateDiagnosis
ON MedicalDiagnosis
AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT * FROM inserted
        WHERE DiagnosisDate > GETDATE()
    )
    BEGIN
        RAISERROR('Cannot insert or update a diagnosis with a future date.', 16, 1)
        ROLLBACK TRANSACTION
    END
END

```

INSERT INTO MedicalDiagnosis (ICD10CodeID, PatientID, PhysicianID, DiagnosisDate) **VALUES**
(1, 1, 2, '2050-05-01');



The screenshot shows the SSMS interface. In the center, a query window displays a T-SQL script. The script includes a trigger-like logic to prevent inserting or updating a diagnosis with a future date. It features an IF EXISTS block that selects from the inserted table where the diagnosis date is greater than the current date. If found, it raises an error and rolls back the transaction. The script then proceeds to insert the values (1, 1, 2, '2050-05-01') into the MedicalDiagnosis table.

```

6   IF EXISTS (
7     SELECT * FROM inserted
8     WHERE DiagnosisDate > GETDATE()
9   )
10  BEGIN
11    RAISERROR('Cannot insert or update a diagnosis with a future date.', 16, 1)
12    ROLLBACK TRANSACTION
13  END
14
15
16
17  INSERT INTO MedicalDiagnosis (ICD10CodeID, PatientID, PhysicianID, DiagnosisDate) VALUES
18  (1, 1, 2, '2050-05-01');
19
20

```

In the bottom right corner of the query window, there is a status bar showing: Ln 17, Col 1 (112 selected), Spaces: 4, UTF-8, LF, 0 rows, MSSQL, 00:00:00, localhost : UniversityMedicalCenter.

The left side of the screen shows the Object Explorer (Object Browser). It lists various databases, tables, and objects within the 'UniversityMedicalCenter' database. The 'Triggers' node under the 'Tables' node is currently selected.

The 'Messages' pane at the bottom shows the execution log:

- Started executing query at Line 17
- Msg 50000, Level 16, State 1, Procedure PreventFutureDateDiagnosis, Line 11
Cannot insert or update a diagnosis with a future date.
- Msg 3609, Level 16, State 1, Line 1
The transaction ended in 1. The batch has been aborted.
- Total execution time: 00:00:00.032

Error occurred as expected.

Transactions

1.

```
BEGIN TRAN;
UPDATE Patients
SET FirstName = 'RR' WHERE PatientID = 1;
COMMIT TRAN;
```

The screenshot shows the SSMS interface with a query window containing the following script:

```

CONNECTIONS
  SERVERS
    myUnivege
      MySchool
      ProductOrders
      UniversityHealthCare
      UniversityMedicalCenter
        Tables
          dbo.Admissions
          dbo.Appointments
          dbo.Billing
          dbo.DischargeInstructions
          dbo.HealthDepartments
          dbo.HospitalRooms
          dbo.ICD10Codes
          dbo.Insurances
          dbo.LabResults
          dbo.LegalLicenses
          dbo.MedicalDiagnosis
          dbo.MedicalDuties
          dbo.MedicationCode
          dbo.MedicationHistory
          dbo.Medications
          dbo.Notes
          dbo.Nurses
          dbo.PatientPhysician
          dbo.Patients
            Columns
              PatientID (PK, int, not null)
              FirstName (varchar(50), not null)
              LastName (varchar(50), not null)
              DateOfBirth (date, not null)
              Gender (char(1), not null)
              PhoneNumber (varchar(20), not null)
              Address (varchar(100), not null)
              City (varchar(50), not null)
              State (varchar(50), not null)
              ZipCode (varchar(10), not null)
              InsuranceID (FK, int, not null)
            Keys
            Constraints
            Triggers
            Indexes
        AZURE
        SQL SERVER BIO DATA CLUSTERS
  @16 0

```

The query window has the following tabs: Run, Cancel, Disconnect, Change Connection, UniversityMedicalCen..., Estimated Plan, Enable Actual Plan, Enable SQLCMD, Export as Notebook.

Messages

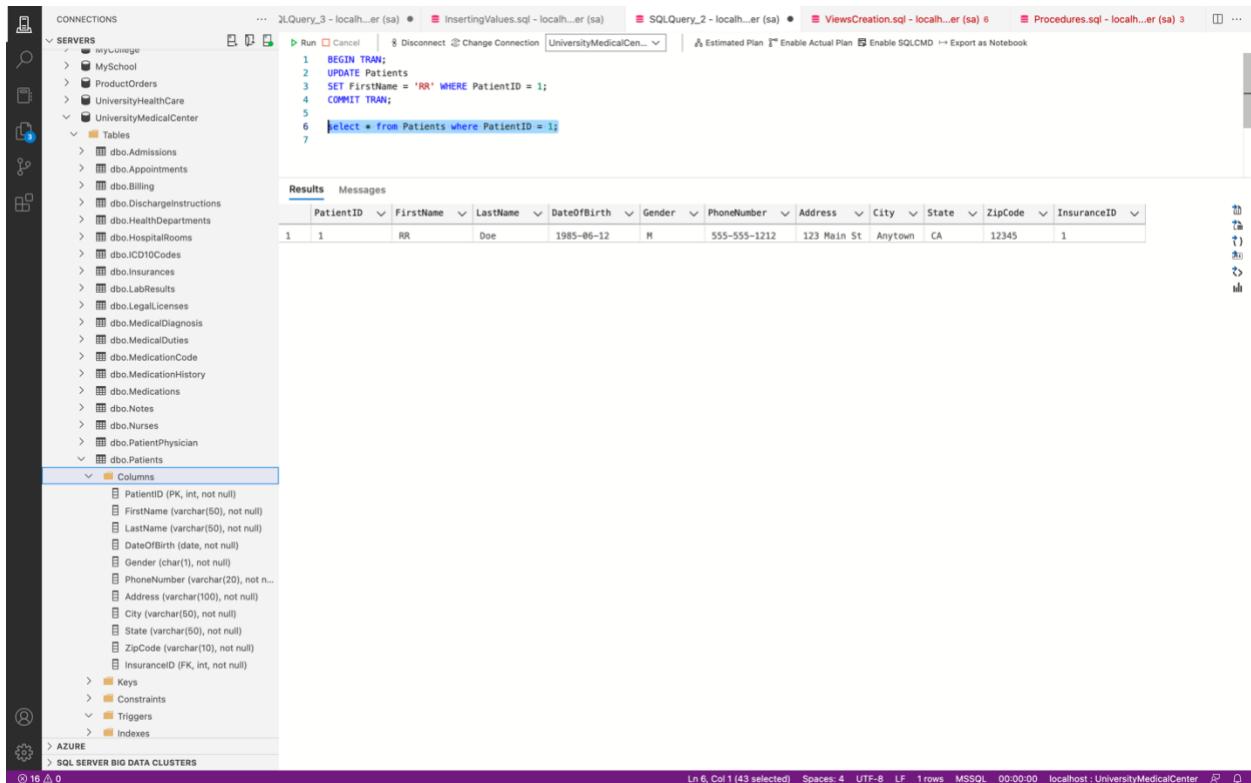
9:00:52 PM Started executing query at Line 1
 (1 row affected)
 Total execution time: 00:00:00.047

Ln 4, Col 13 Spaces: 4 UTF-8 LF 0 rows MSSQL 00:00:00 localhost : UniversityMedicalCenter

CSE 581 INTRO TO DATABASE MANAGEMENT SYSTEM

Updated the FirstName of PatientId 1

`select * from Patients where PatientID = 1;`



The screenshot shows the SSMS interface with the following details:

- Connections:** 2LQuery_3 - local...er (sa), InsertingValues.sql - local...er (sa), SQLQuery_2 - local...er (sa), ViewsCreation.sql - local...er (sa), Procedures.sql - local...er (sa).
- Servers:** wylonege, MySchool, ProductOrders, UniversityHealthCare, UniversityMedicalCenter.
- Tables:** dbo.Admissions, dbo.Appointments, dbo.Billing, dbo.DischargeInstructions, dbo.HealthDepartments, dbo.HospitalRooms, dbo.ICD10Codes, dbo.Insurances, dbo.LabResults, dbo.LegalLicenses, dbo.MedicalDiagnosis, dbo.MedicalDuties, dbo.MedicationCode, dbo.MedicationHistory, dbo.Medications, dbo.Notes, dbo.Nurses, dbo.PatientPhysician, dbo.Patients.
- Columns:** PatientID (PK, int, not null), FirstName (varchar(50), not null), LastName (varchar(50), not null), DateOfBirth (date, not null), Gender (char(1), not null), PhoneNumber (varchar(20), not null), Address (varchar(100), not null), City (varchar(50), not null), State (varchar(50), not null), ZipCode (varchar(10), not null), InsuranceID (FK, int, not null).
- Keys:** Primary Key (PatientID).
- Constraints:** Foreign Key (InsuranceID).
- Triggers:**
- Indexes:**

Query Window:

```

1 BEGIN TRANSACTION;
2 UPDATE Patients
3 SET FirstName = 'RR' WHERE PatientID = 1;
4 COMMIT TRANSACTION;
5
6 select * from Patients where PatientID = 1;
7

```

Results Grid:

PatientID	FirstName	LastName	DateOfBirth	Gender	PhoneNumber	Address	City	State	ZipCode	InsuranceID
1	RR	Doe	1985-06-12	M	555-555-1212	123 Main St	Anytown	CA	12345	1

2.

```
BEGIN TRAN;
UPDATE HospitalRooms
SET RoomType = 'Suite' WHERE RoomID = 7;
COMMIT TRAN;
```

The screenshot shows the SSMS interface with the following details:

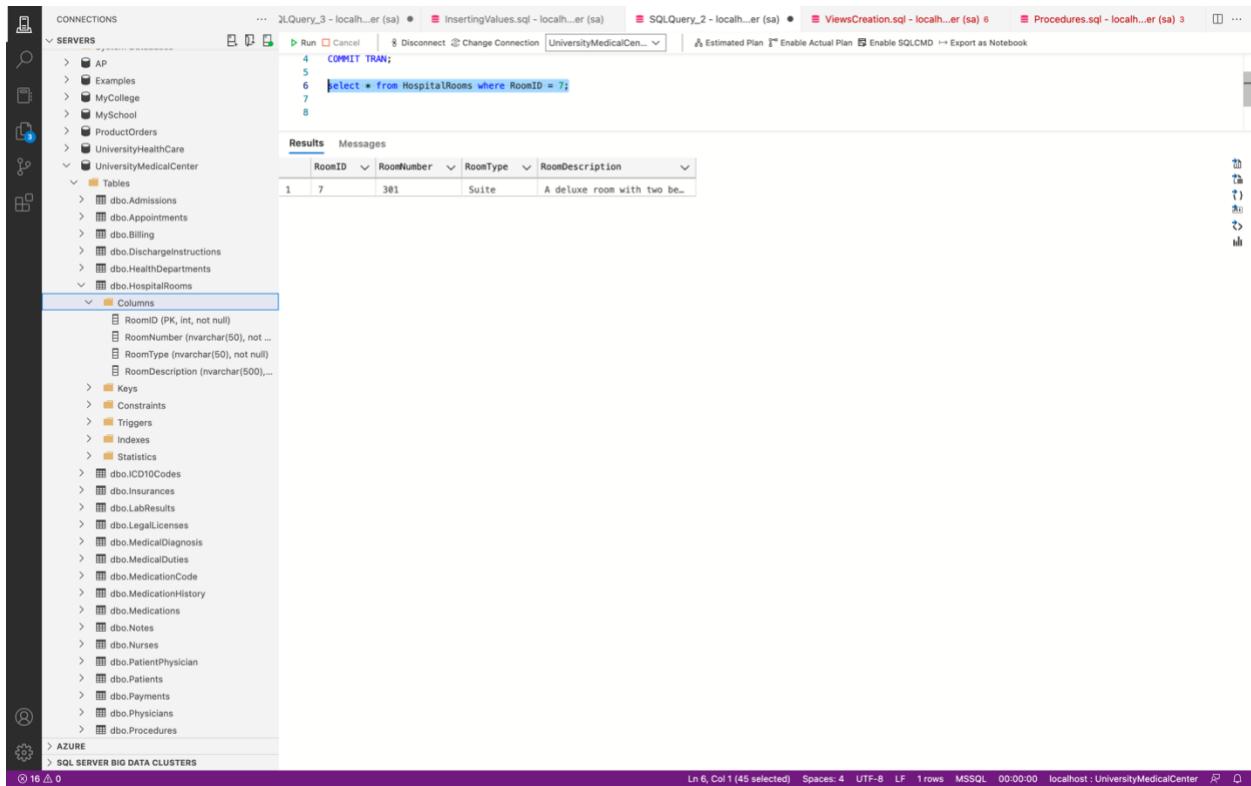
- Connections:** A tree view of servers including AP, Examples, MyCollege, MySchool, ProductOrders, UniversityHealthCare, and UniversityMedicalCenter.
- Query Window:** Contains the following T-SQL code:


```
1 BEGIN TRAN;
2 UPDATE HospitalRooms
3 SET RoomType = 'Suite' WHERE RoomID = 7;
4 COMMIT TRAN;
5
6
```
- Messages:** Displays the execution log:


```
9:05:38 PM Started executing query at Line 1
(1 row affected)
Total execution time: 00:00:00.031
```
- Object Explorer:** Shows the database schema for `dbo.HospitalRooms`, including columns (RoomID, RoomNumber, RoomType, RoomDescription), keys, constraints, triggers, indexes, and statistics.
- Bottom Status Bar:** Shows the following information: Ln 3, Col 23, Spaces: 4, UTF-8, LF, 0 rows, MSSQL, 00:00:00, localhost : UniversityMedicalCenter.

Updated the room type to Suite for roomID 7

`select * from HospitalRooms where RoomID = 7;`



The screenshot shows the SSMS interface with a query window open. The query is:

```

4 COMMIT TRAN;
5
6 select * from HospitalRooms where RoomID = 7;
7
8

```

The results grid displays one row of data:

RoomID	RoomNumber	RoomType	RoomDescription
7	301	Suite	A deluxe room with two be...

The left pane shows the database structure, including tables like dbo.Admissions, dbo.Appointments, etc., under the UniversityMedicalCenter database.

3.

```
BEGIN TRAN;
UPDATE Billing
SET Amount = 2000.00 WHERE BillingID = 3;
COMMIT TRAN;
```

The screenshot shows the SSMS interface with the following details:

- CONNECTIONS:** UniversityMedicalCenter (selected)
- SERVERS:** AP, Examples, MyCollege, MySchool, ProductOrders, UniversityHealthCare, UniversityMedicalCenter (selected)
- Tables:** dbo.Admissions, dbo.Appointments, dbo.Billing (selected), dbo.DischargeInstructions, dbo.HealthDepartments, dbo.HospitalRooms, dbo.ICD10Codes, dbo.Insurances, dbo.LabResults, dbo.LegalLicenses, dbo.MedicalDiagnosis
- Columns:** Under the Billing table, columns listed include BillingID, PatientID, PaymentID, BillingDate, Amount, RoomID, RoomNumber, RoomType, RoomDescription, and RoomCapacity.
- Messages:**
 - Started executing_query_at Line 1
 - [1 row affected]
 - Total execution time: 00:00:00.025
- Toolbar:** Run, Cancel, Disconnect, Change Connection, Estimated Plan, Enable Actual Plan, Enable SQLCMD, Export as Notebook.
- Status Bar:** Ln 4, Col 13, Spaces: 4, UTF-8, LF, 0 rows, MSSQL, 00:00:00, localhost:UniversityMedicalCenter, Refresh, Stop.

CSE 581 INTRO TO DATABASE MANAGEMENT SYSTEM

```
select * from Billing where BillingID = 3;
```

```

1 BEGIN TRAN;
2 UPDATE BILLING
3 SET Amount = 2000.00 WHERE BillingID = 3;
4 COMMIT TRAN;
5
6 select * from Billing where BillingID = 3;
7
8

```

	BillingID	PatientID	PaymentID	BillingDate	Amount
1	3	3	3	2023-04-03	2000.00

Results Messages

BillingID PatientID PaymentID BillingDate Amount

1 3 3 3 2023-04-03 2000.00

Columns Keys Constraints Triggers Indexes Statistics

dbo.Admissions dbo.Appointments dbo.Billing

Columns Keys Constraints Triggers Indexes Statistics

dbo.DischargeInstructions dbo.HealthDepartments

Columns Keys Constraints Triggers Indexes Statistics

dbo.HospitalRooms

Columns Keys Constraints Triggers Indexes Statistics

dbo.ICD10Codes dbo.Insurances dbo.LabResults

Columns Keys Constraints Triggers Indexes Statistics

dbo.LegalLicenses dbo.MedicalDiagnosis

AZURE

SQL SERVER BIDS DATA CLUSTERS

Ln 6, Col 1 (42 selected) Spaces: 4 UTF-8 LF 1 rows MSSQL 00:00:00 localhost :UniversityMedicalCenter

4.

```
BEGIN TRAN;
UPDATE Medications
SET Dosage = '40mg' WHERE PatientID = 1;
COMMIT TRAN;
```

The screenshot shows the SSMS interface with the following details:

- Connections:** Multiple connections listed, including 'InsertingValues.sql - localh...er (sa)', 'SQLQuery_2 - localh...er (sa)', 'ViewsCreation.sql - localh...er (sa)', and 'Procedures.sql - localh...er (sa)'.
- Script Editor:** The script being run is:


```
1 BEGIN TRAN;
2 UPDATE Medications
3 SET Dosage = '40mg' WHERE PatientID = 1;
4 COMMIT TRAN;
```
- Messages:** The output shows:
 - Started executing query at Line 1
 - (3 rows affected)
 - Total execution time: 0:00:00.0026
- Servers:** A tree view of database objects under 'UniversityMedicalCenter' (e.g., Billing, DischargeInstructions, HospitalDepartments, HospitalRooms, ICD10Codes, Insurances, LabResults, LegalLicenses, MedicalDiagnosis, MedicalDuties, MedicationCode, MedicationHistory, Medications, Notes, Nurses, PatientPhysician, Patients, Payments).
- Azure and BI:** Options for Azure and SQL Server BI Data Clusters.
- Status Bar:** Shows 'Ln 1, Col 1 (87 selected)', 'Spaces: 4', 'UTF-8', 'LF', '0 rows', 'MSSQL', '00:00:00', 'localhost : UniversityMedicalCenter', and icons for refresh, save, and help.

CSE 581 INTRO TO DATABASE MANAGEMENT SYSTEM

Updated dosage of patientID to 40mg.

```
select * from Medications where PatientID = 1;
```

	MedicationID	MedicationName	Dosage	Frequency	PrescribingPhysicianID	PatientID
1	1	Lisinopril	40mg	Daily	3	1

Scripts

1.

```
CREATE LOGIN RRLogin WITH PASSWORD = 'rravikan@123'
CREATE USER RRUser FOR LOGIN RRLogin;
```

The screenshot shows the SSMS interface. The Object Explorer on the left lists several database objects under the 'Servers' node, including 'dbo.Billing', 'dbo.DischargeInstructions', 'dbo.HealthDepartments', 'dbo.HospitalRooms', 'dbo.ICD10Codes', 'dbo.Insurances', 'dbo.LabResults', 'dbo.LegalLicenses', 'dbo.MedicalDiagnosis', 'dbo.MedicalDuties', 'dbo.MedicationCode', 'dbo.MedicationHistory', 'dbo.Medications', 'dbo.Notes', 'dbo.Nurses', 'dbo.PatientPhysician', 'dbo.Patients', and 'dbo.Payments'. The 'dbo.Insurances' object is currently selected. The 'Messages' pane at the bottom shows the execution results of the two T-SQL statements:

```
Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.021
```

At the bottom of the screen, the status bar displays: Ln 2, Col 38 Spaces: 4 UTF-8 LF 0 rows MSSQL 00:00:00 localhost:UniversityMedicalCenter

CSE 581 INTRO TO DATABASE MANAGEMENT SYSTEM

2.

```
CREATE LOGIN RavisLogin WITH PASSWORD = 'rravikan@123'
```

```
GO
```

```
CREATE USER RavisUser FOR LOGIN RavisLogin
```

```
GO
```

```
DENY UPDATE, DELETE ON Patients TO RavisUser
```

```
GO
```

The screenshot shows the SSMS interface. The Object Explorer on the left lists database objects like dbo.Billing, dbo.DischargeInstructions, dbo.HealthDepartments, dbo.HospitalRooms, dbo.ICD10Codes, dbo.Insurances, dbo.LabResults, dbo.LegalLicenses, dbo.MedicalDiagnosis, dbo.MedicalDuties, dbo.MedicationCode, dbo.MedicationHistory, dbo.Medications, dbo.Notes, dbo.Nurses, dbo.PatientPhysician, dbo.Patients, and dbo.Payments. The query window on the right contains the provided T-SQL script. The Messages pane at the bottom shows the following log:

- 9:31:29 PM Started executing query at Line 1
Commands completed successfully.
- 9:31:29 PM Started executing query at Line 3
Commands completed successfully.
- 9:31:29 PM Started executing query at Line 5
Commands completed successfully.
- 9:31:29 PM Total execution time: 00:00:00.021

At the bottom of the screen, the status bar displays: Ln 6, Col 3 Spaces: 4 UTF-8 LF 0 rows MSSQL 00:00:00 localhost : UniversityMedicalCenter

3.

```

CREATE ROLE SuperAdmin;
GRANT UPDATE, DELETE ON Patients TO SuperAdmin;
GRANT UPDATE, DELETE ON Billing TO SuperAdmin;
GRANT INSERT ON LabResults TO SuperAdmin;
ALTER ROLE db_datareader ADD MEMBER SuperAdmin

```

The screenshot shows the SSMS interface with the following details:

- Connections:** XQuery_3 - local...er (sa)
- Servers:** UniversityMedicalCen...
- Script:**

```

1 CREATE ROLE SuperAdmin;
2 GRANT UPDATE, DELETE ON Patients TO SuperAdmin;
3 GRANT UPDATE, DELETE ON Billing TO SuperAdmin;
4 GRANT INSERT ON LabResults TO SuperAdmin;
5 ALTER ROLE db_datareader ADD MEMBER SuperAdmin

```
- Messages:**
 - Started executing query at Line 1
 - Commands completed successfully.
 - Total execution time: 00:00:00.070
- Object Explorer:** Shows the database structure including tables like dbo.Billing, dbo.DischargeInstructions, dbo.HealthDepartments, dbo.HospitalRooms, and dbo.Insurances.
- Status Bar:** Lines 1, Col 1 (207 selected), Spaces: 4, UTF-8, LF, 0 rows, MSSQL, 00:00:00, localhost:UniversityMedicalCenter

4.

```
EXEC sp_helpsrvrolemember
'serveradmin';
EXEC sp_helpsrvrolemember
'sysadmin';
```

ServerRole	MemberName	MemberSID
sysadmin	sa	0x01
sysadmin	BUILTIN\Administrators	0x010200000000000520000000...
sysadmin	NT AUTHORITY\NETWORK SERVER	0x010100000000000514000000...

Business Reports

1.

```
SELECT p1.FirstName + ' ' + p1.LastName AS NameOfPatient,
       pp.PatientID,
       phy1.FirstName + ' ' + phy1.LastName AS NameOfPhysician,
       phy1.DepartmentID,
       COUNT(*) AS NumOfVisits
  FROM Patients p1
 JOIN PatientPhysician pp ON p1.PatientID = pp.PatientID
 JOIN Physicians phy1 ON pp.PhysicianID = phy1.PhysicianID
 JOIN Appointments Ap ON pp.PatientPhysicianID = Ap.PatientPhysicianID
 GROUP BY p1.FirstName, p1.LastName, pp.PatientID, phy1.FirstName, phy1.LastName,
          phy1.DepartmentID
 ORDER BY NumOfVisits DESC;
```

NameOfPatient	PatientID	NameOfPhysician	DepartmentID	NumOfVisits
RR Doe	1	Ethan Nguyen	8	1
Jane Smith	2	Ava Garcia	9	1
David Lee	3	Jacob Kim	10	1
Lisa Nguyen	4	Isabella Johnson	11	1
Michael Garcia	5	Noah Martinez	12	1
Emily Kim	6	Olivia Rodriguez	13	1
William Johnson	7	Liam Chen	14	1
Sarah Martinez	8	Emma Gonzalez	15	1
Daniel Rodriguez	9	Mason Perez	16	1
Megan Chen	10	Sophia Lee	7	1

2.

```
SELECT Patients.FirstName, Patients.LastName, SUM(Billing.Amount) AS BilledTotal
FROM Billing
INNER JOIN Patients ON Billing.PatientID = Patients.PatientID
GROUP BY Patients.FirstName, Patients.LastName
ORDER BY BilledTotal DESC;
```

The screenshot shows the SSMS interface with the following details:

- Left pane (Object Explorer):** Shows the database structure under "localhost, <default> (sa)". It includes SERVERS, Databases (System Databases, AP, MyCollege, MySchool, ProductOrders, UniversityHealthCare), Examples, and Programmability (Stored Procedures, Functions, Database Triggers, Assemblies, Types, Sequences, External Resources, Service Broker, Storage, Security, Server Objects).
- Top bar:** Includes buttons for Run, Disconnect, Change Connection, Estimated Plan, Actual Plan, Enable SQLCMD, Export as Notebook, and a toolbar with various icons.
- Query window:** Displays the T-SQL query:

```
1 SELECT Patients.FirstName, Patients.LastName, SUM(Billing.Amount) AS BilledTotal
2 FROM Billing
3 INNER JOIN Patients ON Billing.PatientID = Patients.PatientID
4 GROUP BY Patients.FirstName, Patients.LastName
5 ORDER BY BilledTotal DESC;
```
- Results pane:** Shows the output of the query in a grid format:

	FirstName	LastName	BilledTotal
1	Megan	Chen	4500.00
2	Daniel	Rodriguez	4000.00
3	Sarah	Martinez	3500.00
4	William	Johnson	3000.00
5	Emily	Kim	2500.00
6	David	Lee	2000.00
7	Michael	Garcia	2000.00
8	Lisa	Nguyen	1500.00
9	Jane	Smith	750.00
10	RR	Doe	100.50
- Bottom status bar:** Shows "Ln 1, Col 1 (229 selected)", "Spaces: 4", "UTF-8", "LF", "10 rows", "MSSQL", "00:00:00", "localhost : UniversityMedicalCenter", and icons for refresh, search, and help.

3.

```
SELECT Patients.FirstName, Patients.LastName, Physicians.FirstName, Physicians.LastName,
DischargeInstructions.DischargeInstructions, DischargeInstructions.DischargeDate
FROM DischargeInstructions
INNER JOIN Patients ON DischargeInstructions.PatientID = Patients.PatientID
INNER JOIN Physicians ON DischargeInstructions.PhysicianID = Physicians.PhysicianID
ORDER BY DischargeInstructions.DischargeDate DESC;
```

The screenshot shows the SSMS interface with the following details:

- Object Explorer (Left):** Shows the database structure under the 'UniversityMedicalCenter' database, including tables like 'dbo.Admissions', 'dbo.Appointments', 'dbo.Billing', 'dbo.DischargeInstructions', etc.
- Query Editor (Top):** Displays the T-SQL query provided in the question.
- Results Grid (Bottom):** Shows the output of the query, which lists 10 rows of patient and physician information along with their discharge instructions and dates.

	FirstName	LastName	FirstName	LastName	DischargeInstructions	DischargeDate
1	Daniel	Rodriguez	Mason	Perez	Get plenty of rest and dr...	2022-04-09
2	David	Rodriguez	Mason	Perez	Take medication as prescr...	2022-04-09
3	William	Johnson	Liam	Chen	Avoid strenuous activity ...	2022-04-07
4	William	Johnson	Liam	Chen	Follow up with specialist...	2022-04-07
5	Michael	Garcia	Noah	Martinez	Maintain a healthy diet a...	2022-04-05
6	Michael	Garcia	Noah	Martinez	Take medication as direct...	2022-04-05
7	David	Lee	Jacob	Kim	Increase physical activit...	2022-04-03
8	David	Lee	Jacob	Kim	Return to work in 2 weeks	2022-04-03
9	RR	Doe	Ethan	Nguyen	Take medication as prescr...	2022-04-01
10	RR	Doe	Ethan	Nguyen	Follow up with primary ca...	2022-04-01

4.

```
SELECT HospitalRooms.RoomType, COUNT(*) AS CountOfAdmissions, COUNT(DISTINCT
Admissions.RoomID) AS CountOfOccupiedRooms
FROM Admissions
INNER JOIN HospitalRooms ON Admissions.RoomID = HospitalRooms.RoomID
GROUP BY HospitalRooms.RoomType;
```

```
SELECT Admissions.AdmissionID, CONCAT(Patients.FirstName, ' ', Patients.LastName) AS
NameOfPatient, HospitalRooms.RoomNumber, HospitalRooms.RoomType,
Admissions.AdmissionDate, Admissions.DischargeDate
FROM Admissions
INNER JOIN HospitalRooms ON Admissions.RoomID = HospitalRooms.RoomID
INNER JOIN Patients ON Admissions.PatientID = Patients.PatientID
ORDER BY HospitalRooms.RoomType, Admissions.AdmissionDate;
```

FirstName	LastName	FirstName	LastName	DischargeInstructions	DischargeDate
Daniel	Rodriguez	Mason	Perez	Get plenty of rest and dr...	2022-04-09
Daniel	Rodriguez	Mason	Perez	Take medication as prescr...	2022-04-09
William	Johnson	Liam	Chen	Avoid strenuous activity ...	2022-04-07
William	Johnson	Liam	Chen	Follow up with specialist...	2022-04-07
Michael	Garcia	Noah	Martinez	Maintain a healthy diet a...	2022-04-05
Michael	Garcia	Noah	Martinez	Take medication as direct...	2022-04-05
David	Lee	Jacob	Kim	Increase physical activit...	2022-04-03
David	Lee	Jacob	Kim	Return to work in 2 weeks	2022-04-03
RR	Doe	Ethan	Nguyen	Take medication as prescr...	2022-04-01
RR	Doe	Ethan	Nguyen	Follow up with primary ca...	2022-04-01

Conclusion

To conclude, the database design and implementation project was a comprehensive process that started with understanding the project statement and specifications, followed by determining the information to store, identifying tables, and addressing potential security and integrity issues. The next step involved designing an E/R diagram and normalizing the design into its 3rd Normal Form. Then, during the implementation phase, the schema was created along with tables, necessary relationships, and constraints, and potential security and integrity issues were resolved. This phase also involved creating views, stored procedures, user-defined functions, triggers, transactions, and scripts to create users with various security levels, passwords, and roles. During the testing phase, the database was populated with meaningful test data and the views, stored procedures, functions, triggers, transactions, and scripts were tested with complex scenarios. Finally, at least four business reports were generated. The project highlighted the significance of careful planning, attention to detail, and a comprehensive testing process in ensuring database integrity and security. The resulting database provided a solid foundation for further development and growth, capable of supporting complex business reports and insights that could assist in decision-making and drive success.

Remarks

- The project has taught me the importance of careful planning when designing a database system for a complex organization like a hospital.
- I learned how to determine the necessary information to store and how to design tables and logic for testing and implementation.
- The project showed me how to address potential integrity and security issues in a database system.
- I gained experience in creating views, stored procedures, functions, triggers, and transactions for a database system.
- The project allowed me to practice creating business reports and extracting meaningful insights from data using complex queries with aggregate and analytic functions, CTEs, and OLAP transactions.
- I learned how to normalize a database design into its 3rd Normal Form and to use the CamelCase naming standard to make the design easy to read.
- The project taught me how to code a database system from scratch, without relying on auto-generated SQL.
- I gained experience in testing a database system's logic and integrity by running complex scenarios and transactions and explaining each scenario's objective in the comments area.
- I learned how to generate business reports to provide valuable insights to decision-makers, including creating visualizations using BI tools like Microsoft Power BI.
- The project allowed me to practice populating a database with test data to ensure its functionality.
- I gained a deeper understanding of the importance of database systems in healthcare organizations and how they can facilitate care coordination among healthcare providers and optimize hospital operations.
- The project showed me how to create a descriptive E/R diagram using a modeling tool like Untitled Diagram, Visual Paradigm, Vertabelo, or dbdiagram.io.
- I learned how to create scripts to create users with various security levels, passwords, and roles in a database system.
- The project taught me the importance of documenting my work and providing descriptive comments in my code to ensure its readability and maintainability.
- I gained experience in analyzing data and presenting my findings in a clear and concise manner in a report.