PROJECT – 2

University Medical Centre Database

ABSTRACT

The objective of this project is to design, construct, and test a database management system that will be used to keep University Medical Center's data. This database will also look into the typical activities that take place in a hospital setting and will, as a result, take into account a number of base tables, such as those for medical staff, as well as linking tables, such as those for patient appointments, doctor appointments, nurse appointments, schedules for operating rooms, schedules for inpatient rooms, and schedules for equipment reservations.

INTRODUCTION

Design, Implementation, and Testing make up the three phases of this project. The various Entity Relationships as well as the pertinent tables are included in the design. Along with discussing potential business logics and transactions to take into account, we'll also go over how to include them into the design. Codes that build tables, views, triggers, procedures, etc. are executed during implementation. Executing multiple scenarios to assess the database's complexity is the final step in the testing phase.

DESIGN CONSIDERATION:

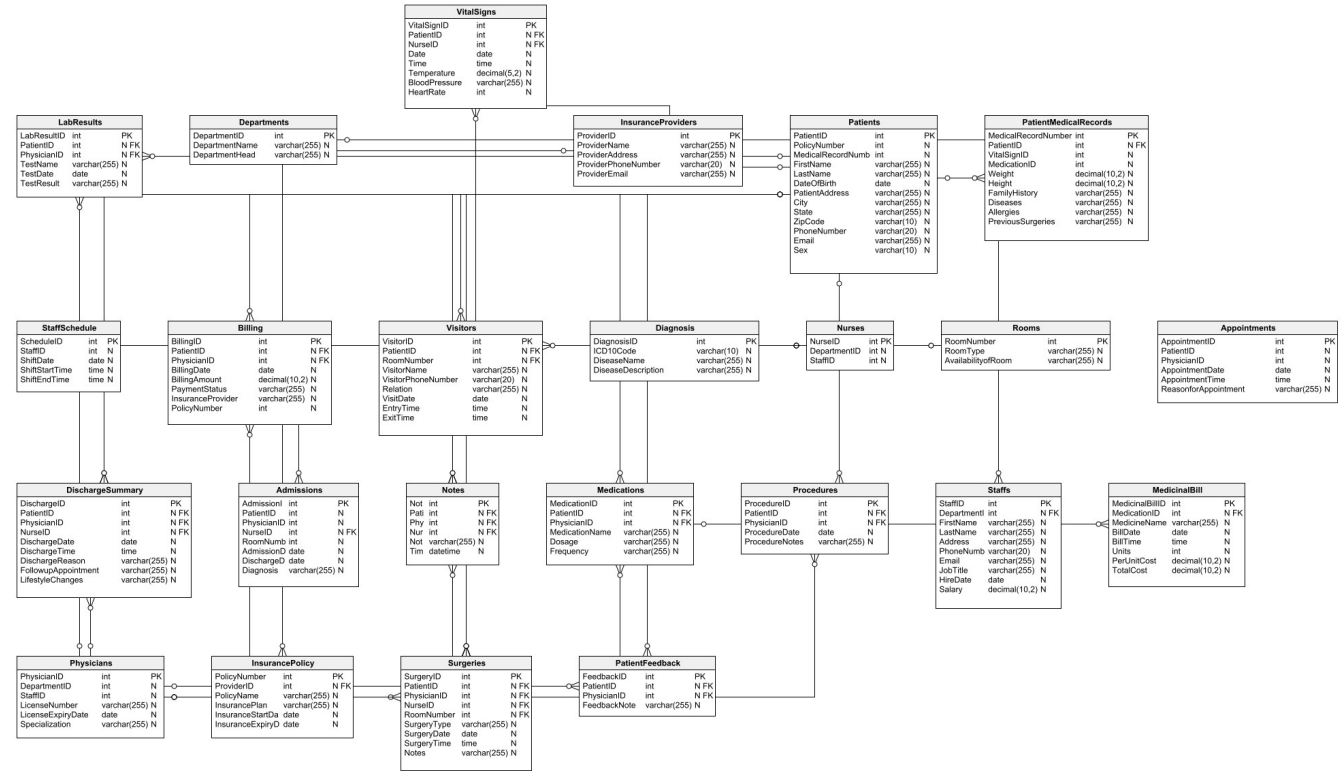
The following entities and attributes are listed by the analysis based on the given project requirements and specifications:

S.NO.	ENTITY	ATTRIBUTES
1	Departments	**DepartmentID [PK]**, DepartmentName,DepartmentHead
2	Staffs	**StaffID [PK], DepartmentID [FK],** FirstName, LastName, Address, PhoneNumber, Email, JobTitle,HireDate, Salary
3	StaffSchedule	**ScheduleID [PK], StaffID [FK]**, ShiftDate,ShiftStartTime. ShiftEndTime
4	Physicians	**PhysicianID [PK]**, **DepartmentID [FK], StaffID[FK]**, LicenseNumber, LicenseExpiryDate, Specialization
5	Nurses	**NurseID [PK]**, **DepartmentID [FK], StaffID [FK]**,DepartmentID,StuffID
6	InsuranceProviders	**ProviderID [PK]**, ProviderName, ProviderAddress,ProviderPhoneNumber, ProviderEmail
7	InsurancePolicy	**PolicyNumber [PK]**, **ProviderID [FK]**, PolicyName,InsurancePlan, InsuranceStartDate, InsuranceExpiryDate
8	Patients	**PatientID [PK]**, **PolicyNumber [FK]**, FirstName, LastName, DateOfBirth, PatientAddress, City, State,ZipCode, PhoneNumber, Email, Sex
9	Billing	**BillingID [PK]**, **PatientID[FK],** PolicyNumber [FK],ProviderID [FK], BillingDate, BillingAmount, PaymentStatus
10	Medications	**MedicationID [PK], PatientID [FK], PhysicianID[FK]**, MedicationName, Dosage, Frequency
11	VitalSigns	**VitalSignID [PK], PatientID [FK], NurseID [FK]**,Date, Time, Temperature, BloodPressure, HeartRate
12	MedicinalBill	MedicinalBill**[PK]**, MedicationID [FK], MedicineName, BillDate, BillTime, Units,PerUnitCost, TotalCost

13	PatientMedicalRecords	**MedicalRecordNumber [PK], PatientID [FK]**,**VitalSignID [FK], MedicationID [FK],** Weight,Height, FamilyHistory, Diseases, Allergies, PreviousSurgeries
14	Appointments	**AppointmentID [PK], PatientID [FK], PhysicianID[FK]**, **NurseId [FK],** AppointmentDate, AppointmentTime, ReasonforAppointment
15	Notes	**NoteID [PK], AppointmentID [FK]**,**Note, Date**
16	Rooms	**RoomNumber [PK]**, RoomType, AvailabilityofRoom
17	Diagnosis	**DiagnosisID [PK]**, ICD10Code, DiseaseName,DiseaseDescription
18	Admissions	**AdmissionID [PK], PatientID [FK], PhysicianID [FK], NurseID [FK], RoomNumber [FK], DiagnosisID [FK], AdmissionDate, DischargeDate**
19	Admissions	**AdmissionID [PK], PatientID [FK], PhysicianID[FK], NurseID [FK], RoomNumber [FK], DiagnosisID [FK],** AdmissionDate, DischargeDate
20	LabResults	LabResultID [PK], PatientID [FK], PhysicianID [FK],TestName, TestDate, TestResult
21	Surgery	**SurgeryID[PK], PatientID[FK], PhysicianID[FK],NurseID[FK], RoomNumber, Surgerytype,SurgeryDate,SurgeryTime**
22	Procedures	**ProcedureID [PK], PatientID [FK], PhysicianID [FK]**, **NurseID [FK], SurgeryID [FK]** ProcedureDate,ProcedureNotes, ProcedureType
23	Visitors	**VisitorID [PK], PatientID [FK], RoomNumber [FK],** VisitorName, VisitorPhoneNumber, Relation, VisitDateEntryTime, ExitTime
24	DischargeSummary	**DischargeID [PK], PatientID [FK], PhysicianID[FK], NurseID [FK], DischargeDate, DischargeTime, DischargeReason, FollowupAppointment, LifestyleChanges**
25	PatientFeedback	**FeedbackID [PK], PatientID [FK], PhysicianID[FK],** FeedbackNote

ENTITY RELATIONSHIP

This relationship as shown in the figure below is incorporated in the 3NF. There are 20+ tables shown here.



B. IMPLEMENTATION :

In this phase, the following is achieved:

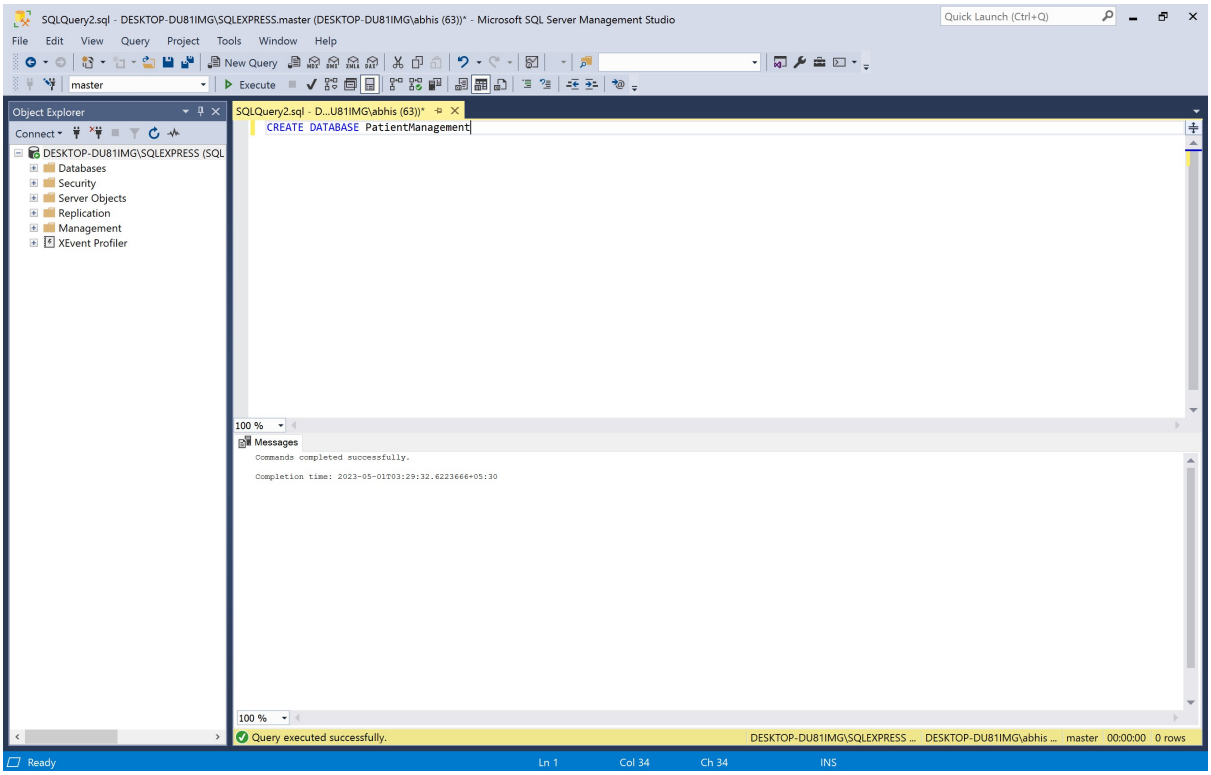
- The tables, columns, primary keys, data types, nullabilities, and relationships are determined ● The design is normalized into its 3rd Normal Form. ● And, potential integrityand security issues are also addressed

//Creating a

Database//Source

Code:

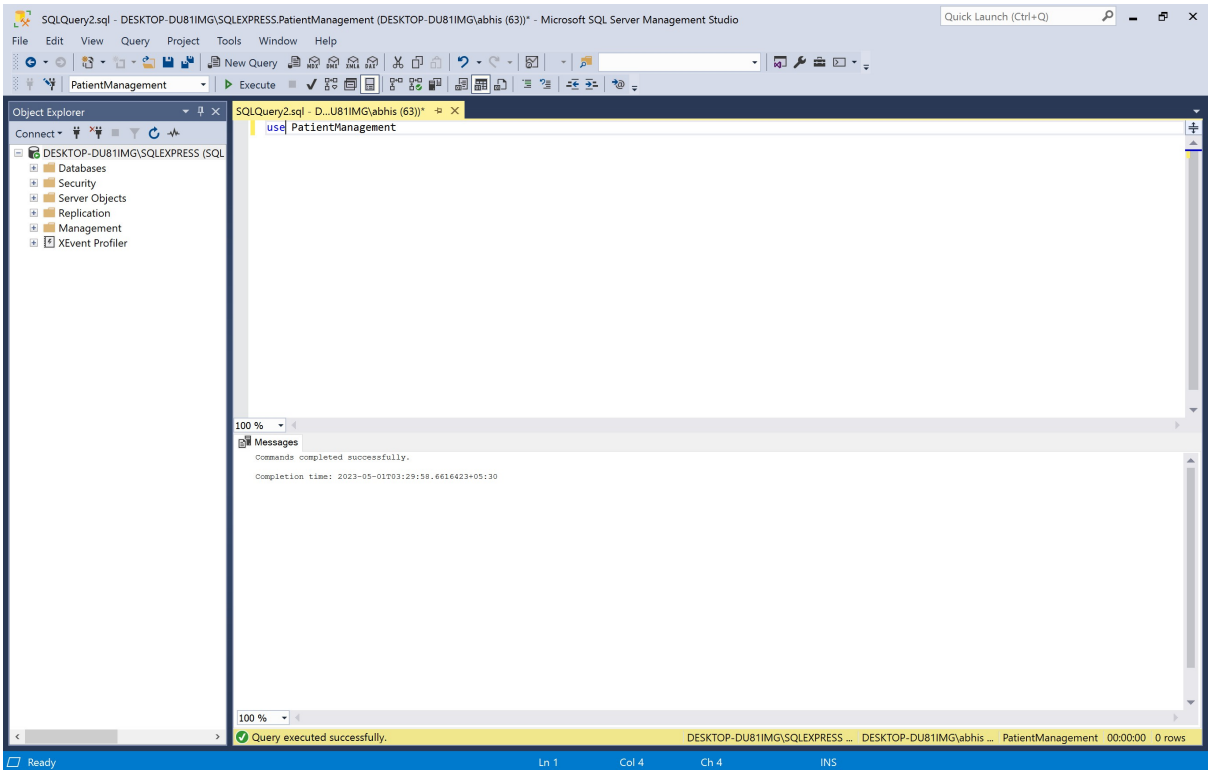
```
CREATE DATABASE
PatientManagement;Results:
```



Comments:

The screenshot above depicts the output when the database is created. The database is named ‘PatientManagement’. It can be seen that the new database was created successfully in the server.

```
************************************************************
This script creates table for database named PatientManagement
************************************************************

*USE PatientManagement;
```




```
-- Creates tables for database PatientManagement

-- Departments table
CREATE TABLE Departments (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(255) NOT
    NULL,DepartmentHead VARCHAR(255) NOT
    NULL
);

-- Staffs table
CREATE TABLE Staffs (
    StaffID INT PRIMARY KEY,
    DepartmentID INT NOT NULL,
    FirstName VARCHAR(50) NOT
    NULL,LastName VARCHAR(50) NOT
    NULL, Address VARCHAR(255),
    PhoneNumber VARCHAR(25),
    Email VARCHAR(255),
    JobTitle
    VARCHAR(50),HireDate
    DATE,
    Salary DECIMAL(10,2),
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
);
```

```sql
--StaffSchedule table
CREATE TABLE StaffSchedule
    ( ScheduleID INT PRIMARY
    KEY,StaffID INT NOT NULL,
    ShiftDate DATE,
    ShiftStartTime
    TIME,ShiftEndTime
    TIME,
    FOREIGN KEY (StaffID) REFERENCES Staffs(StaffID)
);


-- physicians table
CREATE TABLE Physicians (
    PhysicianID INT NOT NULL PRIMARY KEY,
    DepartmentID INT NOT
    NULL,StaffID INT NOT NULL,
    LicenseNumber VARCHAR(50) NOT NULL,
    LicenseExpiryDate DATE NOT NULL,
    Specialization VARCHAR(50) NOT NULL,
    FOREIGN KEY (DepartmentID) REFERENCES
    Departments(DepartmentID),FOREIGN KEY (StaffID) REFERENCES
    Staffs(StaffID)
);


-- nurses table
CREATE TABLE Nurses (
    NurseID INT NOT NULL PRIMARY KEY,
    DepartmentID INT NOT
    NULL,StaffID INT NOT NULL,
    FOREIGN KEY (DepartmentID) REFERENCES
    Departments(DepartmentID),FOREIGN KEY (StaffID) REFERENCES
    Staffs(StaffID)
);

--InsuranceProviders table
CREATE TABLE InsuranceProviders
    (ProviderID INT PRIMARY KEY,
    ProviderName VARCHAR(255) NOT NULL,
    ProviderAddress VARCHAR(255) NOT NULL,
    ProviderPhoneNumber VARCHAR(20) NOT
    NULL,ProviderEmail VARCHAR(255) NOT NULL
);


—InsurancePolicy table
CREATE TABLE InsurancePolicy
    ( PolicyNumber INT PRIMARY KEY,
    ProviderID INT NOT NULL,
    PolicyName VARCHAR(255) NOT NULL,
    InsurancePlan VARCHAR(255) NOT
    NULL,InsuranceStartDate DATE NOT
    NULL, InsuranceExpiryDate DATE NOT
    NULL,
    FOREIGN KEY (ProviderID) REFERENCES InsuranceProviders(ProviderID)
);



-- patients table
CREATE TABLE Patients (
    PatientID INT NOT NULL PRIMARY KEY,
    PolicyNumber INT NOT NULL,
    FirstName VARCHAR(50) NOT
    NULL,LastName VARCHAR(50) NOT
    NULL,
```

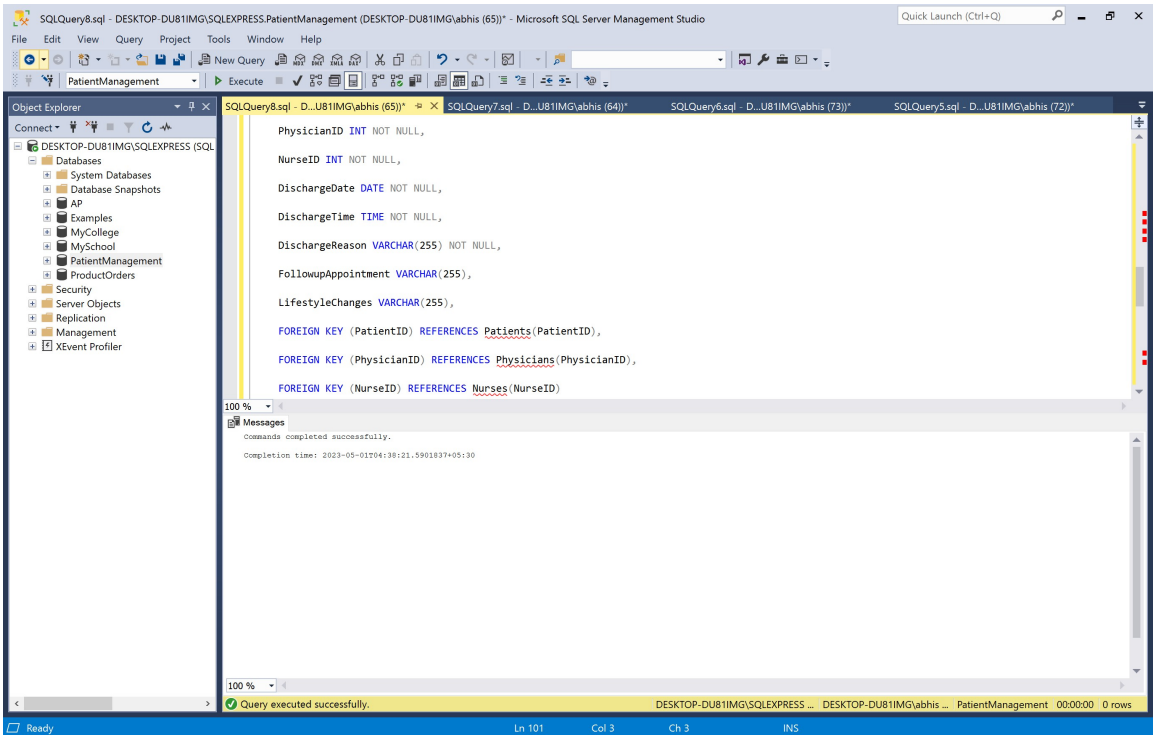
```
    DateOfBirth DATE NOT NULL,
    PatientAddress
    VARCHAR(100),City
    VARCHAR(50),
    State VARCHAR(50),
    ZipCode VARCHAR(10),
    PhoneNumber VARCHAR(20),
    Email VARCHAR(100) NOT
    NULL,Sex VARCHAR(10) NOT
    NULL,
    FOREIGN KEY (PolicyNumber) REFERENCES InsurancePolicy(PolicyNumber),

);


-- billing table
CREATE TABLE Billing (
    BillingID INT PRIMARY
    KEY,PatientID INT NOT
    NULL, PolicyNumber INT Not
    NUll,ProviderID INT NOT
    NULL, BillingDate DATE NOT
    NULL,
    BillingAmount DECIMAL(10,2) NOT
    NULL,PaymentStatus VARCHAR(255) NOT
    NULL,
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (PolicyNumber) REFERENCES
    InsurancePolicy(PolicyNumber),FOREIGN KEY (ProviderID) REFERENCES
    InsuranceProviders(ProviderID)
);

-- medications table
CREATE TABLE Medications (
    MedicationID INT PRIMARY
    KEY,PatientID INT NOT NULL,
    PhysicianID INT NOT NULL,
    MedicationName VARCHAR(255) NOT
    NULL,Dosage VARCHAR(255) NOT NULL,
    Frequency VARCHAR(255) NOT NULL,
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID)
);

-- VitalSigns table
CREATE TABLE VitalSigns (
    VitalSignID INT PRIMARY
    KEY,PatientID INT NOT NULL,
    NurseID INT NOT NULL,
    Date
    DATE,Time
    TIME,
    Temperature DECIMAL(5,2),
    BloodPressure
    VARCHAR(255),HeartRate INT,
    FOREIGN KEY (PatientID) REFERENCES
    Patients(PatientID),FOREIGN KEY (NurseID) REFERENCES
    Nurses(NurseID)
);



—MedicinalBill table
CREATE TABLE MedicinalBill
    ( MedicinalBillID INT PRIMARY KEY,
    MedicationID INT NOT NULL,
    MedicineName VARCHAR(255) NOT
    NULL,BillDate DATE NOT NULL,
```

```
BillTime TIME NOT NULL,
```

```
    Units INT NOT NULL,
    PerUnitCost DECIMAL(10,2) NOT
    NULL,TotalCost DECIMAL(10,2) NOT
    NULL,
    FOREIGN KEY (MedicationID) REFERENCES Medications(MedicationID)
);

-- PatientMedicalRecords table
CREATE TABLE PatientMedicalRecords (
    MedicalRecordNumber INT PRIMARY
    KEY,PatientID INT NOT NULL,
    VitalSignID INT NOT NULL,
    MedicationID INT NOT
    NULL,Weight DECIMAL(10,2),
    Height DECIMAL(10,2),
    FamilyHistory VARCHAR(255),
    Diseases VARCHAR(255),
    Allergies VARCHAR(255),
    PreviousSurgeries VARCHAR(255),
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (VitalSignID) REFERENCES VitalSigns(VitalSignID),
    FOREIGN KEY (MedicationID) REFERENCES Medications(MedicationID)
);

-- Appointments table
CREATE TABLE Appointments (
    AppointmentID INT NOT NULL PRIMARY
    KEY,PatientID INT NOT NULL,
    PhysicianID INT NOT
    NULL,NurseID INT NOT
    NULL,
    AppointmentDate DATE NOT NULL,
    AppointmentTime TIME NOT NULL,
    ReasonforAppointment VARCHAR(255) NOT NULL,
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES
    Physicians(PhysicianID),FOREIGN KEY (NurseID) REFERENCES
    Nurses (NurseID)
);


-- Notes table
CREATE TABLE Notes (
    NoteID INT PRIMARY KEY,
    AppointmentID INT NOT
    NULL,Note VARCHAR(500),
    Date DATETIME,
    FOREIGN KEY (AppointmentID) REFERENCES Appointments (AppointmentID)
);

-- Rooms table
CREATE TABLE Rooms (
    RoomNumber INT PRIMARY
    KEY,RoomType VARCHAR(50),
    AvailabilityofRoom BIT NOT NULL DEFAULT 1
);



-- Diagnosis table
CREATE TABLE Diagnosis (
    DiagnosisID INT PRIMARY KEY,
    ICD10Code VARCHAR(255) NOT NULL,
    DiseaseName VARCHAR(255) NOT NULL,
    DiseaseDescription TEXT NOT NULL
);
```

```sql
-- Admissions table
CREATE TABLE Admissions (
    AdmissionID INT NOT NULL PRIMARY
    KEY,PatientID INT NOT NULL,
    PhysicianID INT NOT
    NULL,NurseID INT NOT
    NULL,
    RoomNumber INT NOT NULL,
    AdmissionDate DATE NOT
    NULL,DischargeDate DATE NOT
    NULL,DiagnosisID INT NOT
    NULL
    FOREIGN KEY (PatientID) REFERENCES Patients (PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES Physicians
    (PhysicianID),FOREIGN KEY (NurseID) REFERENCES Nurses
    (NurseID),
    FOREIGN KEY (RoomNumber) REFERENCES Rooms (RoomNumber),
    FOREIGN KEY (DiagnosisID) REFERENCES Diagnosis (DiagnosisID)
);

-- LabResults table
CREATE TABLE LabResults (
    LabResultID INT PRIMARY KEY,
    PatientID INT NOT NULL,
    PhysicianID INT NOT NULL,
    TestName VARCHAR(255) NOT NULL,
    TestDate DATE NOT NULL,
    TestResult VARCHAR(255) NOT NULL,
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID)
);

-- Surgeries table
CREATE TABLE Surgeries (
    SurgeryID INT PRIMARY
    KEY,PatientID INT NOT
    NULL, PhysicianID INT NOT
    NULL, NurseID INT NOT
    NULL,
    RoomNumber INT NOT
    NULL,SurgeryType
    VARCHAR(50),SurgeryDate
    DATE, SurgeryTime TIME,
    Notes VARCHAR(1000),
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES
    Physicians(PhysicianID),FOREIGN KEY (NurseID) REFERENCES
    Nurses(NurseID),
    FOREIGN KEY (RoomNumber) REFERENCES Rooms(RoomNumber)
);


-- Procedures table
CREATE TABLE Procedures (
    ProcedureID INT NOT NULL PRIMARY
    KEY,PatientID INT NOT NULL,
    PhysicianID INT NOT NULL,
    NurseID INT NOT NULL,
    SurgeryID INT NOT NULL,
    ProcedureDate DATE,
    ProcedureNotes VARCHAR(255),
    FOREIGN KEY (PatientID) REFERENCES Patients (PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES Physicians
    (PhysicianID),FOREIGN KEY (NurseID) REFERENCES Nurses
    (NurseID),
    FOREIGN KEY (SurgeryID) REFERENCES Surgeries (SurgeryID)
);
```

```sql
-- visitors table
CREATE TABLE Visitors (
    VisitorID INT PRIMARY KEY,
    PatientID INT NOT NULL,
    RoomNumber INT NOT NULL,
    VisitorName VARCHAR(255),
    VisitorPhoneNumber
    VARCHAR(20),Relation
    VARCHAR(255), VisitDate DATE,
    EntryTime
    TIME,ExitTime
    TIME,
    FOREIGN KEY (PatientID) REFERENCES
    Patients(PatientID),FOREIGN KEY (RoomNumber) REFERENCES
    Rooms(RoomNumber)
);

-- DiscahargeSummary table
CREATE TABLE DischargeSummary (
    DischargeID INT PRIMARY
    KEY,PatientID INT NOT NULL,
    PhysicianID INT NOT NULL,
    NurseID INT NOT NULL,
    DischargeDate DATE NOT NULL,
    DischargeTime TIME NOT NULL,
    DischargeReason VARCHAR(255) NOT
    NULL,FollowupAppointment VARCHAR(255),
    LifestyleChanges VARCHAR(255),
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES
    Physicians(PhysicianID),FOREIGN KEY (NurseID) REFERENCES
    Nurses(NurseID)
);

-- PatientFeedback table
CREATE TABLE PatientFeedback (
    FeedbackID INT PRIMARY
    KEY,PatientID INT NOT NULL,
    PhysicianID INT NOT NULL,
    FeedbackNote VARCHAR(1000),
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID)
);
```



Comments : Created all tables together and attached the screenshot for the same.

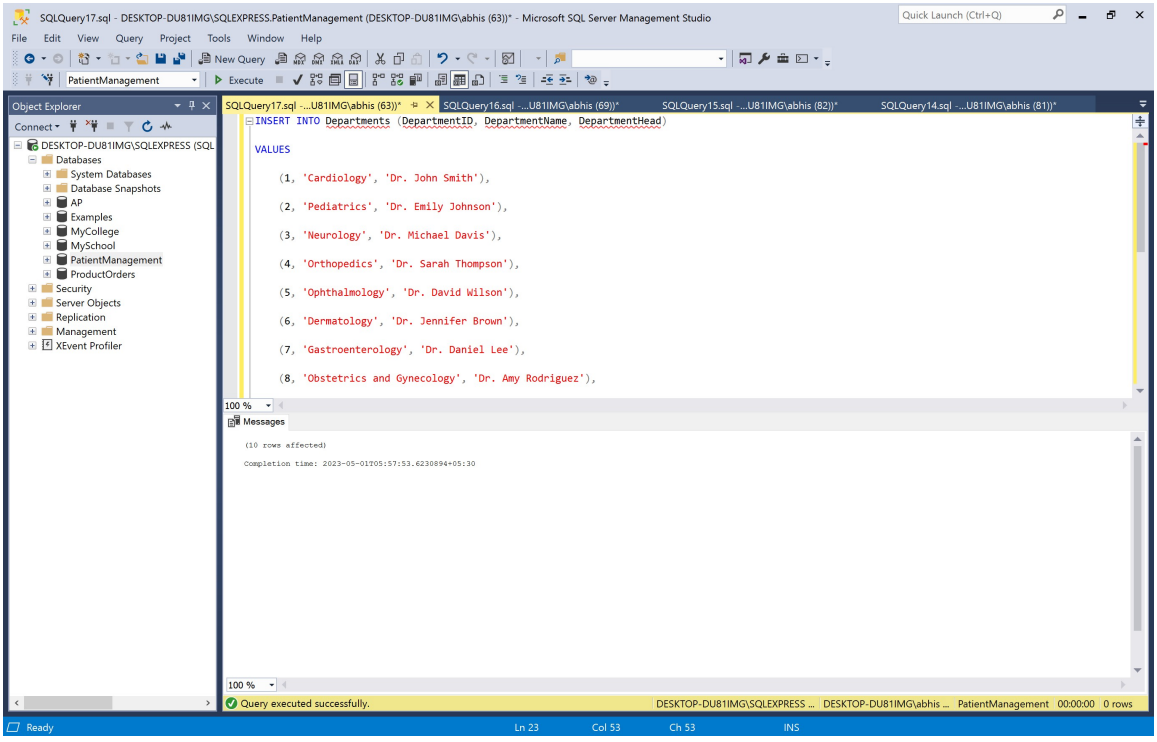
//The following script shows the source code for the insertion of values in the tables//

Insert data into the tables:

Departments:

Query:

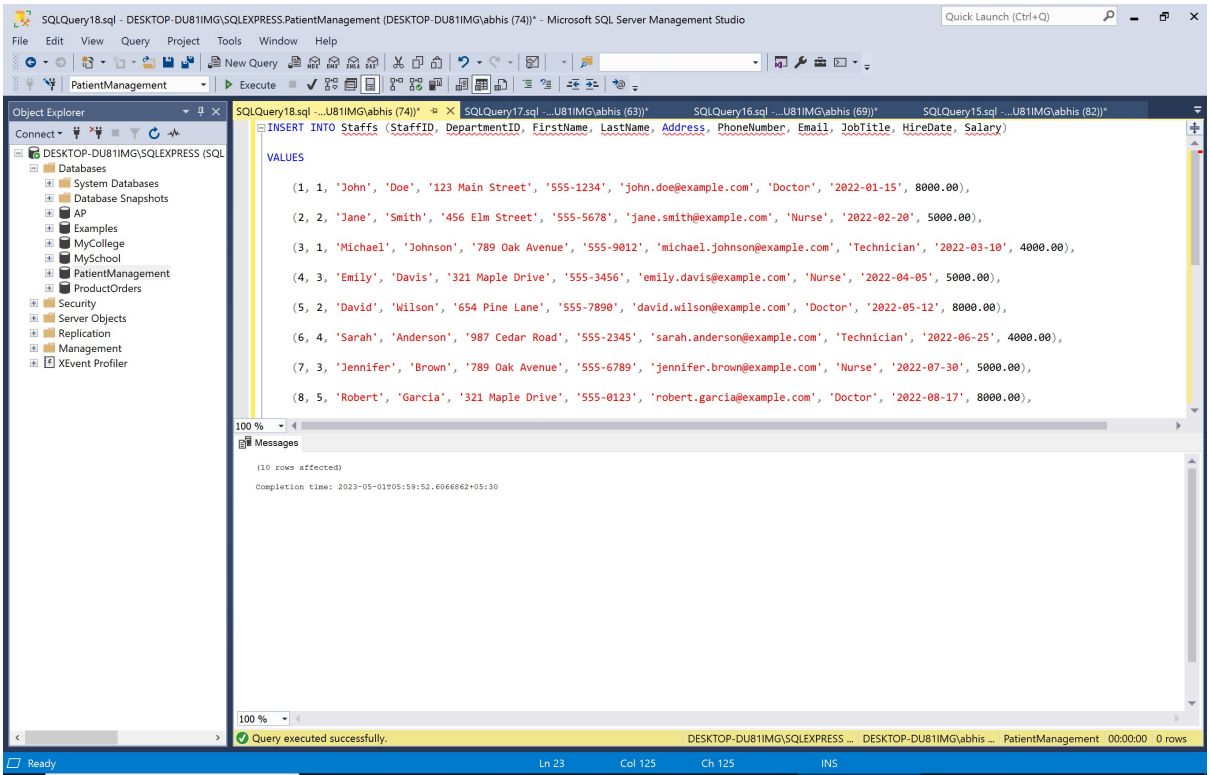
```
INSERT INTO Departments (DepartmentID, DepartmentName,
DepartmentHead)VALUES
    (1, 'Cardiology', 'Dr. John Smith'),
    (2, 'Pediatrics', 'Dr. Emily
    Johnson'),(3, 'Neurology', 'Dr. Michael
    Davis'),
    (4, 'Orthopedics', 'Dr. Sarah Thompson'),
    (5, 'Ophthalmology', 'Dr. David Wilson'),
    (6, 'Dermatology', 'Dr. Jennifer Brown'),
    (7, 'Gastroenterology', 'Dr. Daniel Lee'),
    (8, 'Obstetrics and Gynecology', 'Dr. Amy
    Rodriguez'),(9, 'Psychiatry', 'Dr. Robert Johnson'),
    (10, 'Internal Medicine', 'Dr. Karen Thompson');
```



Staffs

Query:

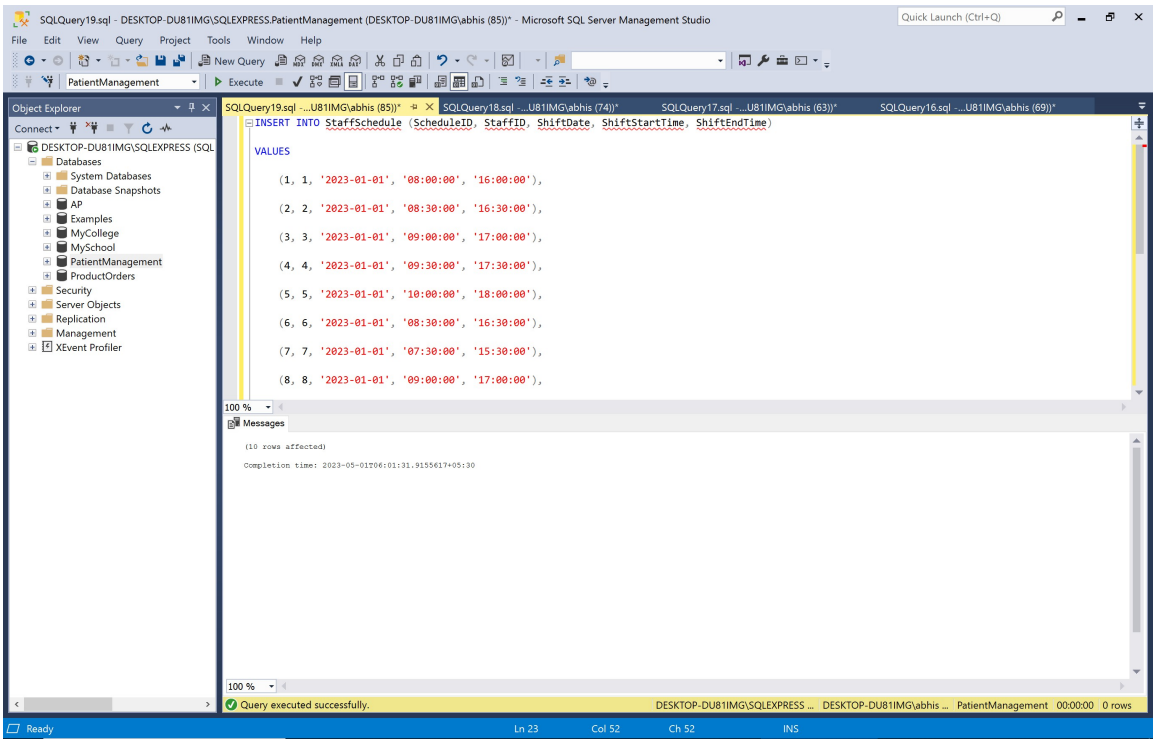
```
INSERT INTO Staffs (StaffID, DepartmentID, FirstName, LastName,
Address,PhoneNumber, Email, JobTitle, HireDate, Salary)
VALUES
    (1, 1, 'John', 'Doe', '123 Main Street', '555-1234',
'john.doe@example.com', 'Doctor', '2022-01-15', 8000.00),
    (2, 2, 'Jane', 'Smith', '456 Elm Street', '555-5678',
'jane.smith@example.com', 'Nurse', '2022-02-20', 5000.00),
    (3, 1, 'Michael', 'Johnson', '789 Oak Avenue', '555-9012',
'michael.johnson@example.com', 'Technician', '2022-03-10', 4000.00),
    (4, 3, 'Emily', 'Davis', '321 Maple Drive', '555-3456',
'emily.davis@example.com', 'Nurse', '2022-04-05', 5000.00),
    (5, 2, 'David', 'Wilson', '654 Pine Lane', '555-7890',
'david.wilson@example.com', 'Doctor', '2022-05-12', 8000.00),
    (6, 4, 'Sarah', 'Anderson', '987 Cedar Road', '555-2345',
'sarah.anderson@example.com', 'Technician', '2022-06-25', 4000.00),
    (7, 3, 'Jennifer', 'Brown', '789 Oak Avenue', '555-6789',
'jennifer.brown@example.com', 'Nurse', '2022-07-30', 5000.00),
    (8, 5, 'Robert', 'Garcia', '321 Maple Drive', '555-0123',
'robert.garcia@example.com', 'Doctor', '2022-08-17', 8000.00),
    (9, 4, 'Amy', 'Lee', '654 Pine Lane', '555-4567', 'amy.lee@example.com',
'Technician', '2022-09-22', 4000.00),
    (10, 1, 'Daniel', 'Taylor', '987 Cedar Road', '555-8901',
'daniel.taylor@example.com', 'Doctor', '2022-10-11', 8000.00);
```



StaffSchedule

Query:

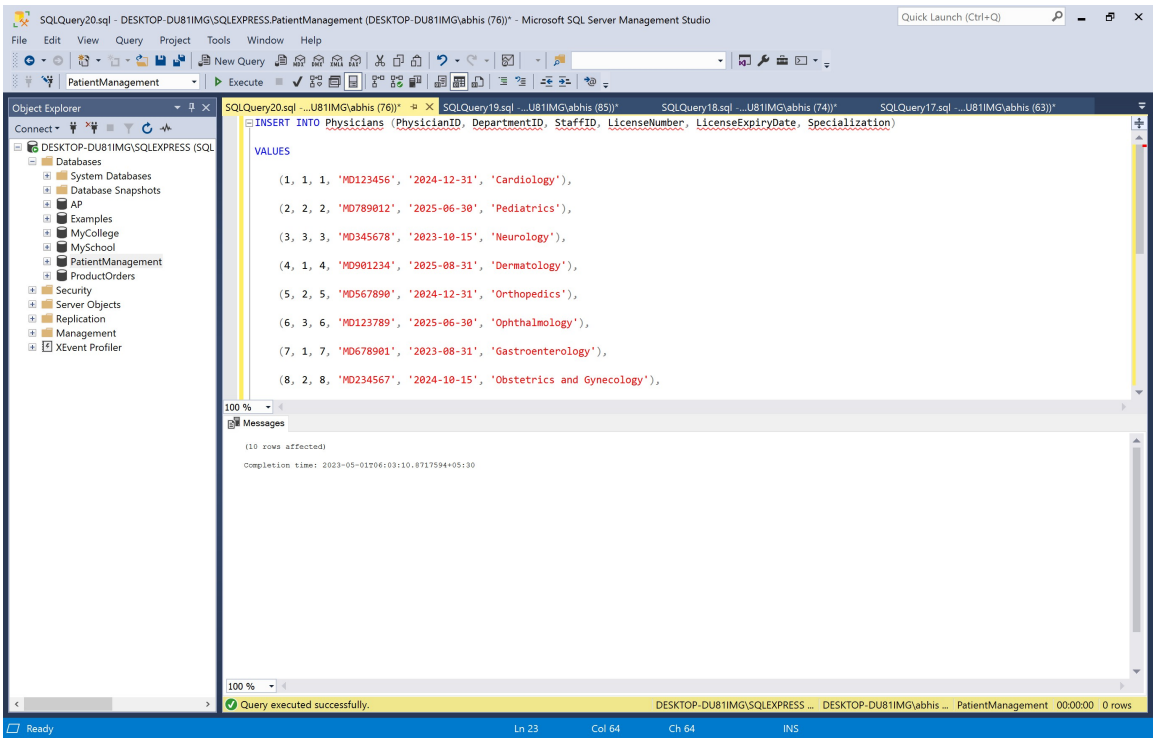
```
INSERT INTO StaffSchedule (ScheduleID, StaffID, ShiftDate,
ShiftStartTime,ShiftEndTime)
VALUES
     (1  1   '2023-01-01   '08:00:00   '16:00:00'
     ,  ,       ',           ',             ),
     (2  2   '2023-01-01   '08:30:00   '16:30:00'
     ,  ,       ',           ',             ),
     (3  3   '2023-01-01   '09:00:00   '17:00:00'
     ,  ,       ',           ',             ),
     (4  4   '2023-01-01   '09:30:00   '17:30:00'
     ,  ,       ',           ',             ),
     (5  5   '2023-01-01   '10:00:00   '18:00:00'
     ,  ,       ',           ',             ),
     (6  6   '2023-01-01   '08:30:00   '16:30:00'
     ,  ,       ',           ',             ),
     (7  7   '2023-01-01   '07:30:00   '15:30:00'
     ,  ,       ',           ',             ),
     (8  8   '2023-01-01   '09:00:00   '17:00:00'
     ,  ,       ',           ',             ),
     (9  9   '2023-01-01   '08:00:00   '16:00:00'
     ,  ,       ',           ',             ),
    (10, 10, '2023-01-01', '08:30:00', '16:30:00');
```

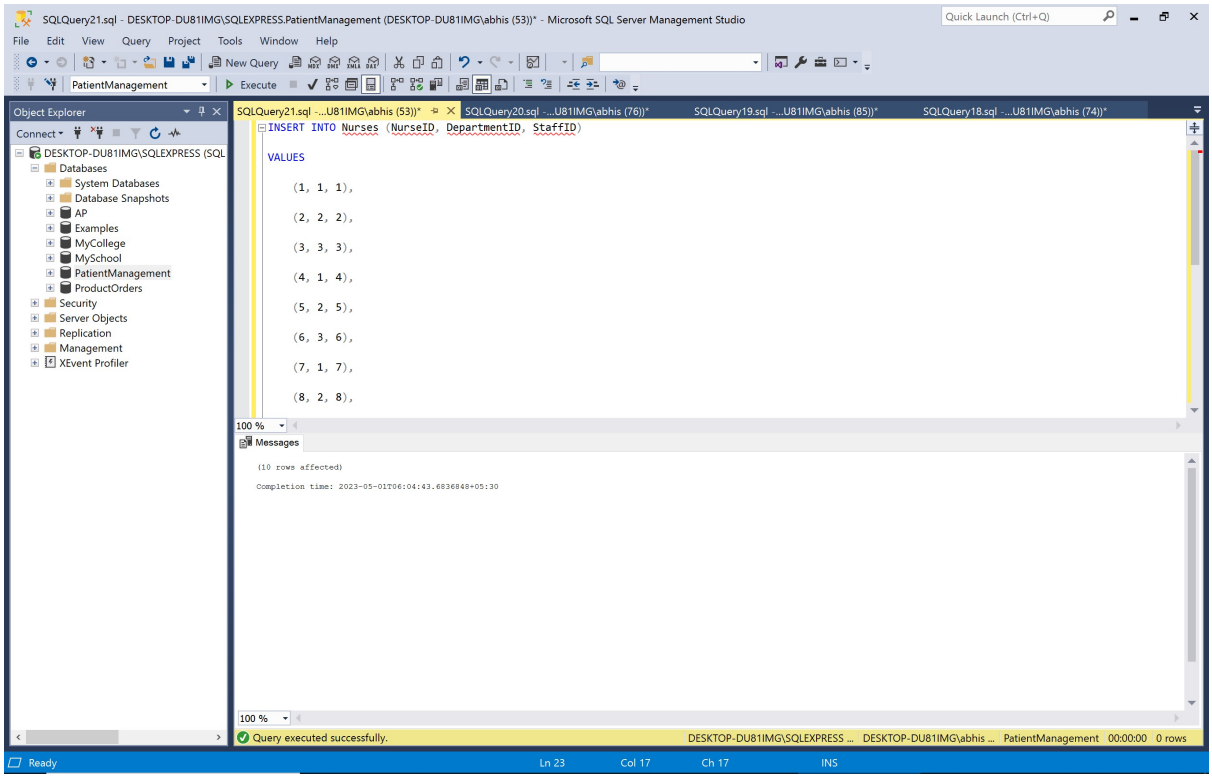


Physician

sQuery:

```
INSERT INTO Physicians (PhysicianID, DepartmentID, StaffID,
LicenseNumber,LicenseExpiryDate, Specialization)
VALUES
    (1, 1, 1, 'MD123456', '2024-12-31', 'Cardiology'),
    (2, 2, 2, 'MD789012', '2025-06-30', 'Pediatrics'),
    (3, 3, 3, 'MD345678', '2023-10-15', 'Neurology'),
    (4, 1, 4, 'MD901234', '2025-08-31', 'Dermatology'),
    (5, 2, 5, 'MD567890', '2024-12-31', 'Orthopedics'),
    (6, 3, 6, 'MD123789', '2025-06-30', 'Ophthalmology'),
    (7, 1, 7, 'MD678901', '2023-08-31', 'Gastroenterology'),
    (8, 2, 8, 'MD234567', '2024-10-15', 'Obstetrics and
    Gynecology'),
    (9, 3, 9, 'MD890123', '2025-12-31', 'Psychiatry'),
    (10, 1, 10, 'MD456789', '2024-06-30', 'Internal Medicine');
```


SQLQuery20.sql - DESKTOP-DU81IMG\SQLEXPRESS.PatientManagement (DESKTOP-DU81IMG\abhis (76))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
PatientManagement
Execute
Object Explorer
Connect
DESKTOP-DU81IMG\SQLEXPRESS (SQL
Databases
System Databases
Database Snapshots
AP
Examples
MyCollege
MySchool
PatientManagement
ProductOrders
Security
Server Objects
Replication
Management
XEvent Profiler
SQLQuery20.sql -...U81IMG\abhis (76))*
SQLQuery19.sql -...U81IMG\abhis (85))*
SQLQuery18.sql -...U81IMG\abhis (74))*
SQLQuery17.sql -...U81IMG\abhis (63))*
INSERT INTO Physicians (PhysicianID, DepartmentID, StaffID, LicenseNumber, LicenseExpiryDate, Specialization)
VALUES
(1, 1, 1, 'MD123456', '2024-12-31', 'Cardiology'),
(2, 2, 2, 'MD789012', '2025-06-30', 'Pediatrics'),
(3, 3, 3, 'MD345678', '2023-10-15', 'Neurology'),
(4, 1, 4, 'MD901234', '2025-08-31', 'Dermatology'),
(5, 2, 5, 'MD567890', '2024-12-31', 'Orthopedics'),
(6, 3, 6, 'MD123789', '2025-06-30', 'Ophthalmology'),
(7, 1, 7, 'MD678901', '2023-08-31', 'Gastroenterology'),
(8, 2, 8, 'MD234567', '2024-10-15', 'Obstetrics and Gynecology'),
100 %
Messages
(10 rows affected)
Completion time: 2023-05-01T06:03:10.8717594+05:30
100 %
Query executed successfully.
DESKTOP-DU81IMG\SQLEXPRESS ...
DESKTOP-DU81IMG\abhis ...
PatientManagement
00:00:00
0 rows
Ready
Ln 23
Col 64
Ch 64
INS



Nurse

s

Query:

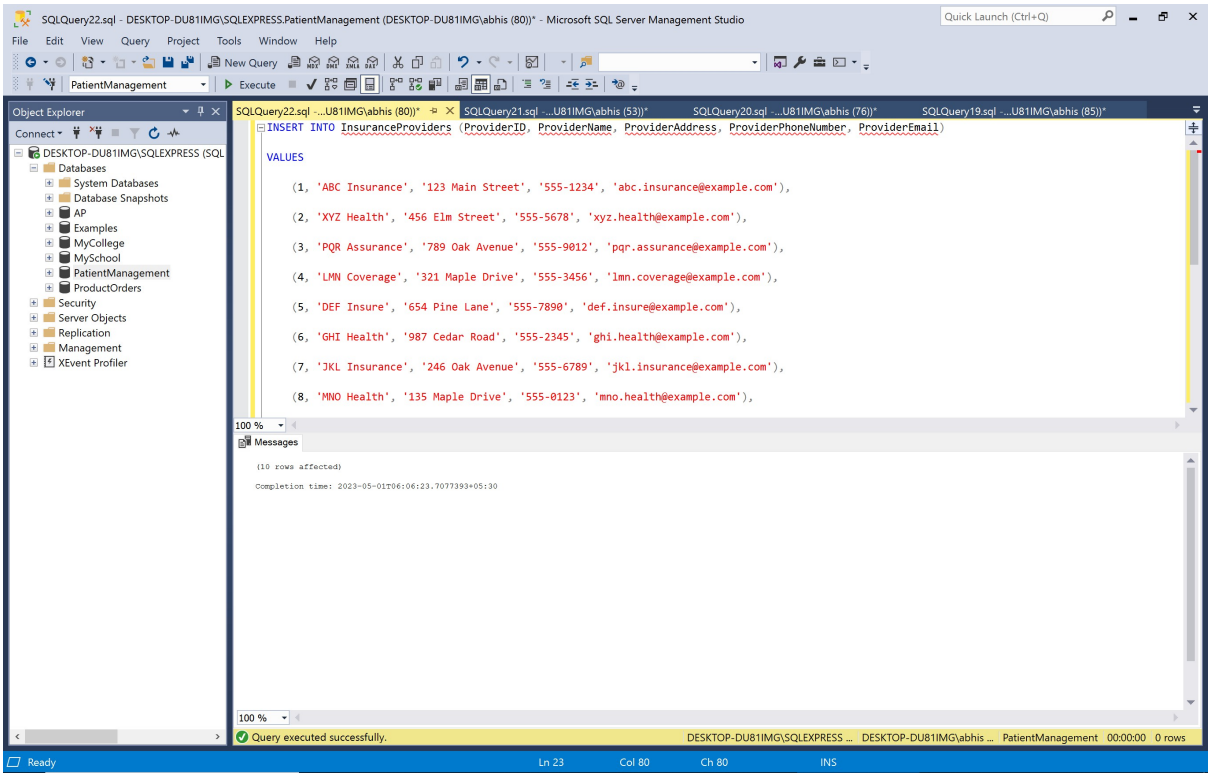
```
INSERT INTO Nurses (NurseID, DepartmentID,
StaffID)VALUES
     (1  1   1)
     ,  ,    ,
     (2  2   2)
     ,  ,    ,
     (3  3   3)
     ,  ,    ,
     (4  1   4)
     ,  ,    ,
     (5  2   5)
     ,  ,    ,
     (6  3   6)
     ,  ,    ,
     (7  1   7)
     ,  ,    ,
     (8  2   8)
     ,  ,    ,
     (9  3   9)
     ,  ,    ,
    (10, 1, 10);
```

InsuranceProvider

sQuery:

```
INSERT INTO InsuranceProviders (ProviderID, ProviderName,
ProviderAddress,ProviderPhoneNumber, ProviderEmail)
VALUES
    (1, 'ABC Insurance', '123 Main Street',
'555-1234','abc.insurance@example.com'),
    (2, 'XYZ Health', '456 Elm Street', '555-5678', 'xyz.health@example.com'),
    (3, 'PQR Assurance', '789 Oak Avenue',
'555-9012','pqr.assurance@example.com'),
    (4, 'LMN Coverage', '321 Maple Drive',
'555-3456','lmn.coverage@example.com'),
```

```
    (5, 'DEF Insure', '654 Pine Lane', '555-7890', 'def.insure@example.com'),
    (6, 'GHI Health', '987 Cedar Road', '555-2345', 'ghi.health@example.com'),
    (7, 'JKL Insurance', '246 Oak Avenue',
'555-6789','jkl.insurance@example.com'),
    (8, 'MNO Health', '135 Maple Drive', '555-0123', 'mno.health@example.com'),
    (9, 'RST Assurance', '864 Pine Lane',
'555-4567','rst.assurance@example.com'),
    (10, 'UVW Insure', '573 Cedar Road', '555-8901', 'uvw.insure@example.com');
```



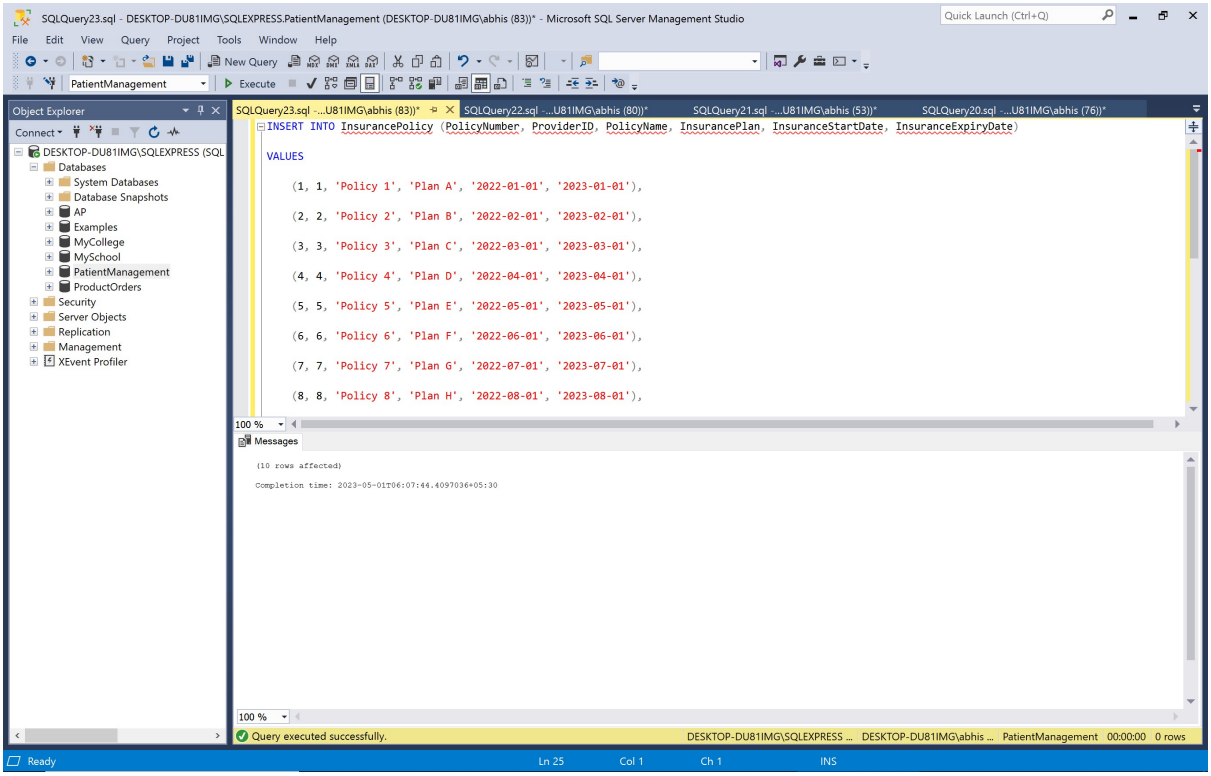
InsurancePolic

yQuery:

```
INSERT INTO InsurancePolicy (PolicyNumber, ProviderID,
PolicyName,InsurancePlan, InsuranceStartDate, InsuranceExpiryDate)
VALUES
    (1  1   'Poli  1'   'Pl  A'   '2022-01-01  '2023-01-01'
    ,   ,     cy   ,    an   ,           ',        ),
    (2  2   'Poli  2'   'Pl  B'   '2022-02-01  '2023-02-01'
    ,   ,     cy   ,    an   ,           ',        ),
    (3  3   'Poli  3'   'Pl  C'   '2022-03-01  '2023-03-01'
    ,   ,     cy   ,    an   ,           ',        ),
    (4  4   'Poli  4'   'Pl  D'   '2022-04-01  '2023-04-01'
    ,   ,     cy   ,    an   ,           ',        ),
```

```
(5, 5, 'Policy 5', 'Plan E', '2022-05-01', '2023-05-01'),
(6, 6, 'Policy 6', 'Plan F', '2022-06-01', '2023-06-01'),
```

```
    (7, 7, 'Policy 7', 'Plan G', '2022-07-01', '2023-07-01'),
    (8, 8, 'Policy 8', 'Plan H', '2022-08-01', '2023-08-01'),
    (9, 9, 'Policy 9', 'Plan I', '2022-09-01', '2023-09-01'),
    (10, 10, 'Policy 10', 'Plan J', '2022-10-01', '2023-10-01');
```

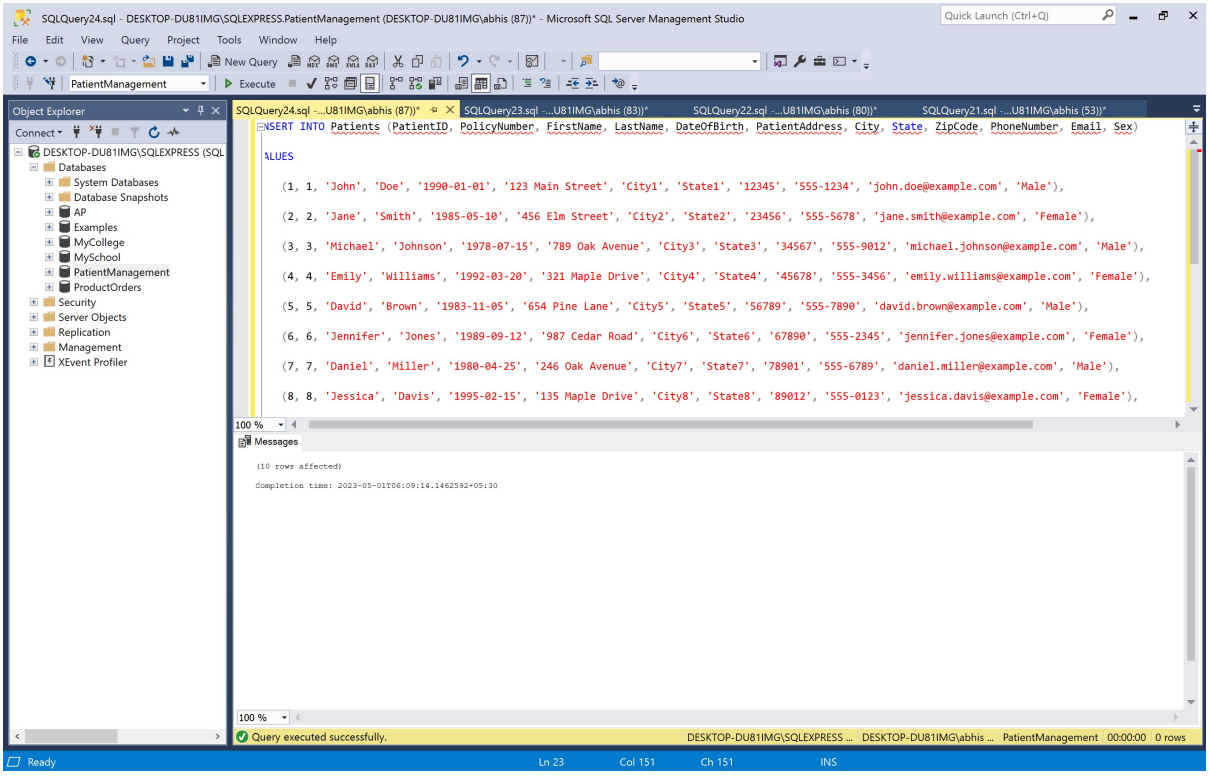


Patient

sQuery:

```
INSERT INTO Patients (PatientID, PolicyNumber, FirstName, LastName,
DateOfBirth, PatientAddress, City, State, ZipCode, PhoneNumber, Email,
Sex)VALUES
    (1, 1, 'John', 'Doe', '1990-01-01', '123 Main Street', 'City1', 'State1',
'12345', '555-1234', 'john.doe@example.com', 'Male'),
    (2, 2, 'Jane', 'Smith', '1985-05-10', '456 Elm Street', 'City2', 'State2',
'23456', '555-5678', 'jane.smith@example.com', 'Female'),
    (3, 3, 'Michael', 'Johnson', '1978-07-15', '789 Oak Avenue', 'City3',
'State3', '34567', '555-9012', 'michael.johnson@example.com', 'Male'),
    (4, 4, 'Emily', 'Williams', '1992-03-20', '321 Maple Drive', 'City4',
'State4', '45678', '555-3456', 'emily.williams@example.com', 'Female'),
    (5, 5, 'David', 'Brown', '1983-11-05', '654 Pine Lane', 'City5', 'State5',
'56789', '555-7890', 'david.brown@example.com', 'Male'),
    (6, 6, 'Jennifer', 'Jones', '1989-09-12', '987 Cedar Road', 'City6',
'State6', '67890', '555-2345', 'jennifer.jones@example.com', 'Female'),
    (7, 7, 'Daniel', 'Miller', '1980-04-25', '246 Oak Avenue', 'City7',
'State7', '78901', '555-6789', 'daniel.miller@example.com', 'Male'),
```

```
    (8, 8, 'Jessica', 'Davis', '1995-02-15', '135 Maple Drive', 'City8',
'State8', '89012', '555-0123', 'jessica.davis@example.com', 'Female'),
    (9, 9, 'Christopher', 'Wilson', '1975-08-30', '864 Pine Lane', 'City9',
'State9', '90123', '555-4567', 'christopher.wilson@example.com', 'Male'),
    (10, 10, 'Samantha', 'Taylor', '1987-06-08', '573 Cedar Road', 'City10',
'State10', '01234', '555-8901', 'samantha.taylor@example.com', 'Female');
```

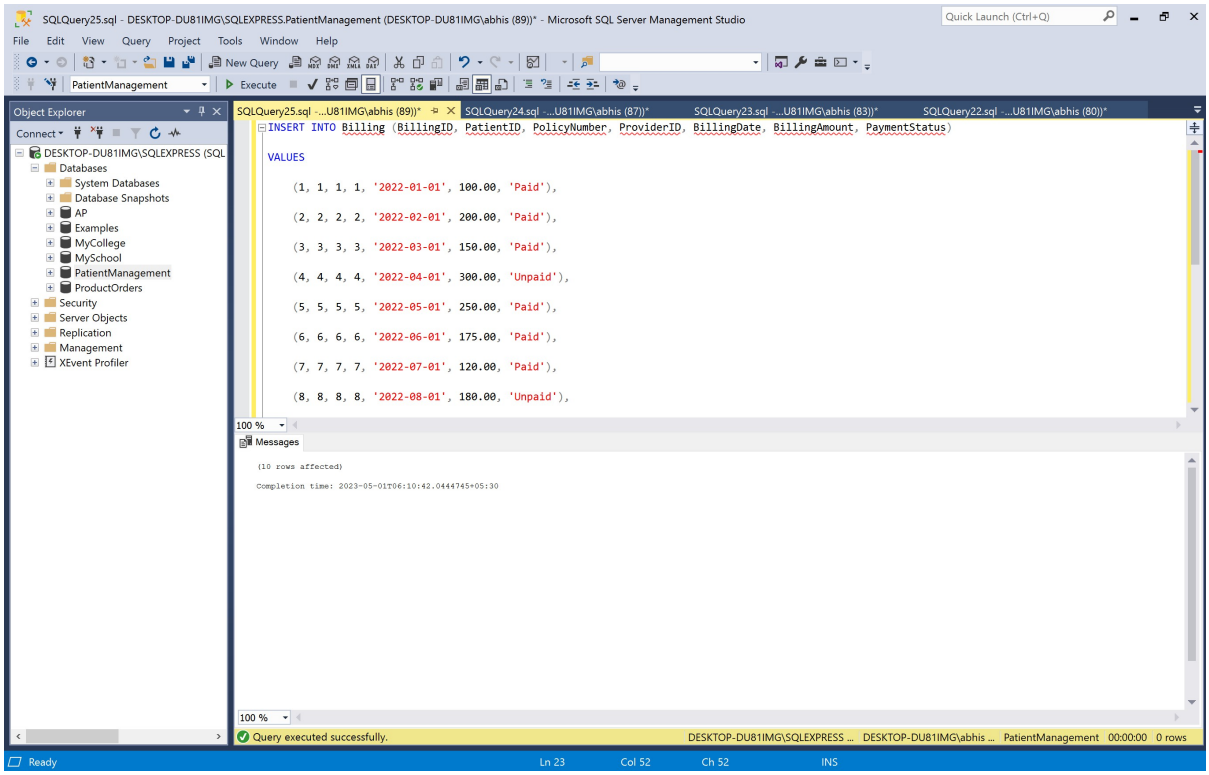


Billing

Query:

```
INSERT INTO Billing (BillingID, PatientID, PolicyNumber,
ProviderID,BillingDate, BillingAmount, PaymentStatus)
VALUES
    (1  1   1  1   '2022-01-01   100.0   'Paid'),
    ,   ,   ,  ,       ',         0,
    (2  2   2  2   '2022-02-01   200.0   'Paid'),
    ,   ,   ,  ,       ',         0,
    (3  3   3  3   '2022-03-01   150.0   'Paid'),
    ,   ,   ,  ,       ',         0,
    (4  4   4  4   '2022-04-01   300.0   'Unpaid'
    ,   ,   ,  ,       ',         0,    ),
    (5  5   5  5   '2022-05-01   250.0   'Paid'),
    ,   ,   ,  ,       ',         0,
    (6  6   6  6   '2022-06-01   175.0   'Paid'),
    ,   ,   ,  ,       ',         0,
    (7  7   7  7   '2022-07-01   120.0   'Paid'),
    ,   ,   ,  ,       ',         0,
```

```
(8  8   8  8   '2022-08-01   180.0   'Unpaid'
,   ,   ,  ,       ',          0,    ),

(9  9   9  9   '2022-09-01   220.0   'Paid'),
,   ,   ,  ,       ',          0,

(10, 10, 10, 10, '2022-10-01', 240.00, 'Paid');
```

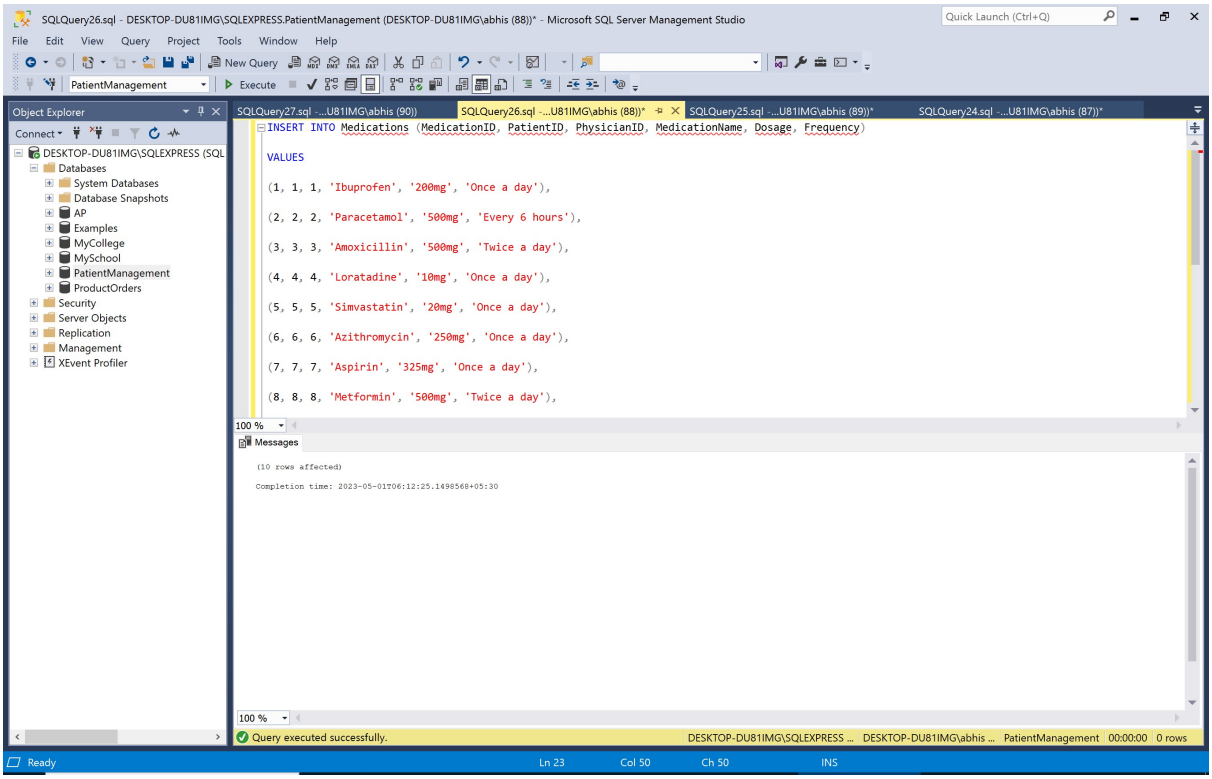


Medication

sQuery:

```
INSERT INTO Medications (MedicationID, PatientID, PhysicianID,
MedicationName,Dosage, Frequency)
VALUES
 (1   1  1   'Ibuprofen', '200mg', 'Once a day'),
 ,    ,  ,
 (2   2  2   'Paracetamol', '500mg', 'Every 6
 ,    ,  ,   hours'),
 (3   3  3   'Amoxicillin', '500mg', 'Twice a day'),
 ,    ,  ,
 (4   4  4   'Loratadine', '10mg', 'Once a day'),
 ,    ,  ,
 (5   5  5   'Simvastatin', '20mg', 'Once a day'),
 ,    ,  ,
 (6   6  6   'Azithromycin', '250mg', 'Once a day'),
 ,    ,  ,
 (7   7  7   'Aspirin', '325mg', 'Once a day'),
 ,    ,  ,
```

```
(8   8  8   'Metformin', '500mg', 'Twice a day'),
,    ,  ,
(9   9  9   'Naproxen', '500mg', 'Twice a day'),
,    ,  ,
(10, 10, 10, 'Metoprolol', '25mg', 'Once a day');
```

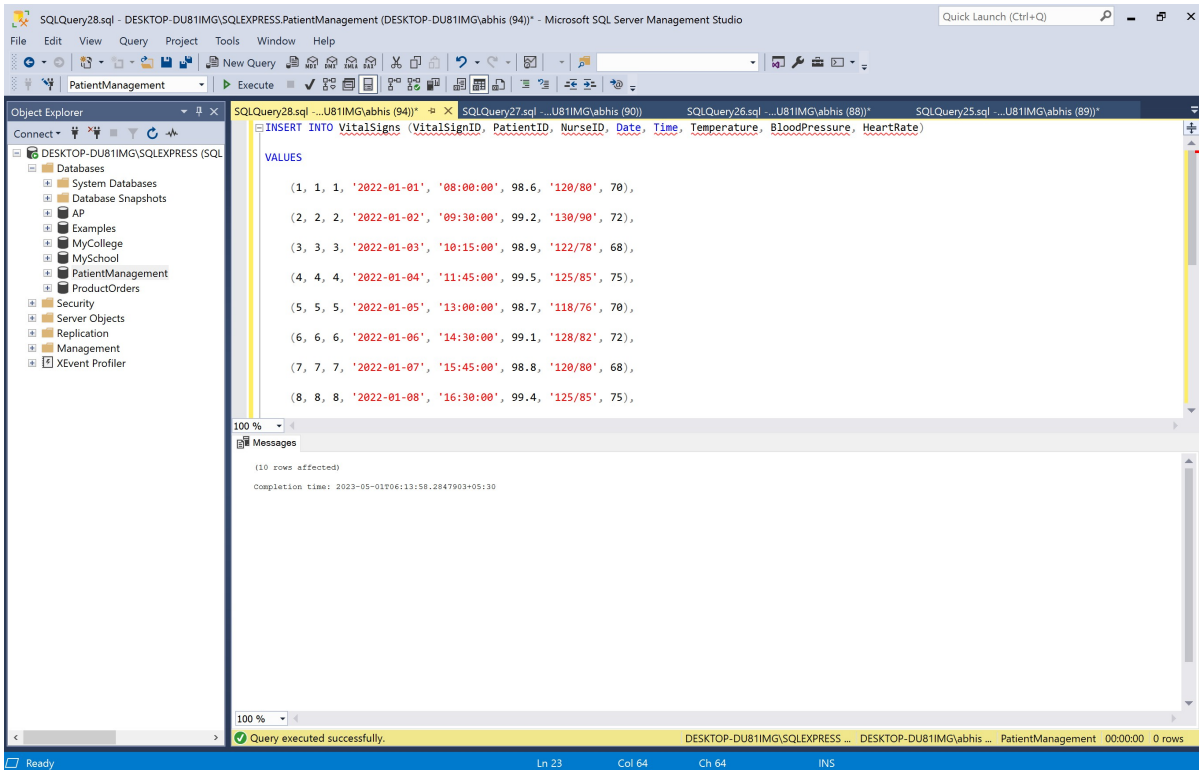


VitalSign

sQuery:

```
INSERT INTO VitalSigns (VitalSignID, PatientID, NurseID, Date,
Time,Temperature, BloodPressure, HeartRate)
VALUES
     (1  1   1   '2022-01-01    '08:00:00   98.   '120/80   70)
     ,   ,   ,       ',              ',    6,      ',       ,

     (2  2   2   '2022-01-02    '09:30:00   99.   '130/90   72)
     ,   ,   ,       ',              ',    2,      ',       ,

     (3  3   3   '2022-01-03    '10:15:00   98.   '122/78   68)
     ,   ,   ,       ',              ',    9,      ',       ,

     (4  4   4   '2022-01-04    '11:45:00   99.   '125/85   75)
     ,   ,   ,       ',              ',    5,      ',       ,

     (5  5   5   '2022-01-05    '13:00:00   98.   '118/76   70)
     ,   ,   ,       ',              ',    7,      ',       ,

     (6  6   6   '2022-01-06    '14:30:00   99.   '128/82   72)
     ,   ,   ,       ',              ',    1,      ',       ,

     (7  7   7   '2022-01-07    '15:45:00   98.   '120/80   68)
     ,   ,   ,       ',              ',    8,      ',       ,

     (8  8   8   '2022-01-08    '16:30:00   99.   '125/85   75)
     ,   ,   ,       ',              ',    4,      ',       ,
```

```
(9  9   9  '2022-01-09   '17:15:00   98.   '122/78   70)
,   ,   ,       ',              ',   5,      ',       ,
(10, 10, 10, '2022-01-10', '18:00:00', 99.0, '130/90', 72);
```

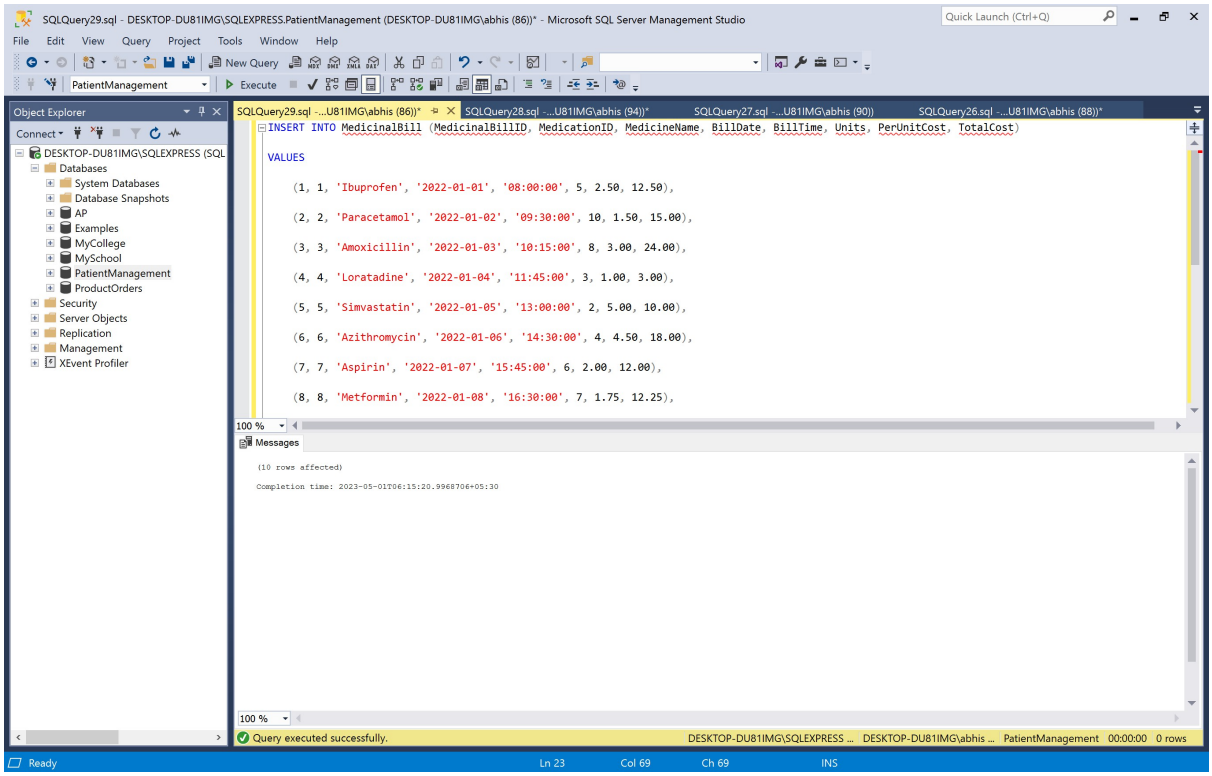


MedicinalBil

lQuery:

```
INSERT INTO MedicinalBill (MedicinalBillID, MedicationID,
MedicineName,BillDate, BillTime, Units, PerUnitCost, TotalCost)
VALUES
    (1  1   'Ibuprofen', '2022-01-01', '08:00:00', 5, 2.50, 12.50),
    ,   ,

    (2  2   'Paracetamol', '2022-01-02', '09:30:00', 10, 1.50,
    ,   ,   15.00),

    (3  3   'Amoxicillin', '2022-01-03', '10:15:00', 8, 3.00,
    ,   ,   24.00),

    (4  4   'Loratadine', '2022-01-04', '11:45:00', 3, 1.00, 3.00),
    ,   ,

    (5  5   'Simvastatin', '2022-01-05', '13:00:00', 2, 5.00,
    ,   ,   10.00),

    (6  6   'Azithromycin', '2022-01-06', '14:30:00', 4, 4.50,
    ,   ,   18.00),
```

```
(7, 7, 'Aspirin', '2022-01-07', '15:45:00', 6, 2.00, 12.00),
(8, 8, 'Metformin', '2022-01-08', '16:30:00', 7, 1.75, 12.25),
(9, 9, 'Naproxen', '2022-01-09', '17:15:00', 9, 3.50, 31.50),
(10, 10, 'Metoprolol', '2022-01-10', '18:00:00', 1, 6.00, 6.00);
```

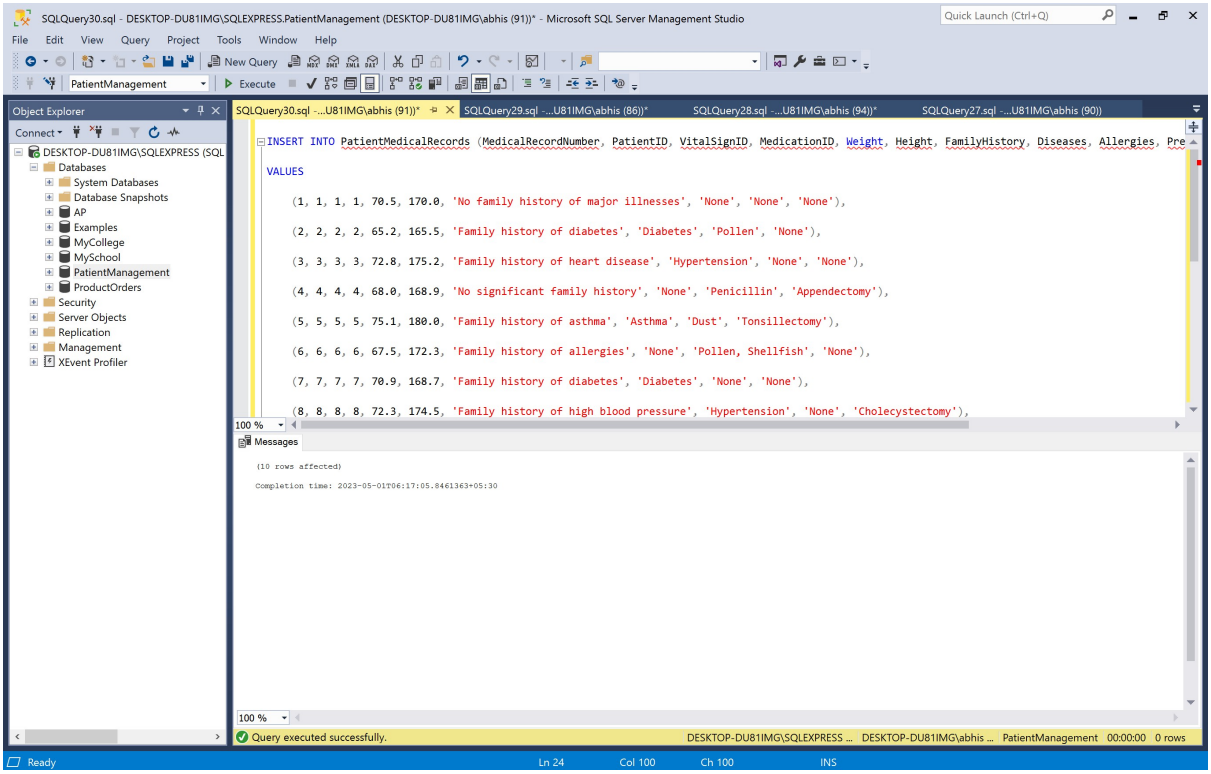


PatientMedicalRecords

Query:

```
INSERT INTO PatientMedicalRecords (MedicalRecordNumber, PatientID,
VitalSignID,MedicationID, Weight, Height, FamilyHistory, Diseases, Allergies,
PreviousSurgeries)
VALUES
    (1, 1, 1, 1, 70.5, 170.0, 'No family history of major illnesses',
'None','None', 'None'),
    (2, 2, 2, 2, 65.2, 165.5, 'Family history of diabetes',
'Diabetes','Pollen', 'None'),
    (3, 3, 3, 3, 72.8, 175.2, 'Family history of heart
disease','Hypertension', 'None', 'None'),
    (4, 4, 4, 4, 68.0, 168.9, 'No significant family history',
'None','Penicillin', 'Appendectomy'),
    (5, 5, 5, 5, 75.1, 180.0, 'Family history of asthma', 'Asthma',
'Dust','Tonsillectomy'),
    (6, 6, 6, 6, 67.5, 172.3, 'Family history of allergies', 'None',
'Pollen,Shellfish', 'None'),
    (7, 7, 7, 7, 70.9, 168.7, 'Family history of diabetes', 'Diabetes',
'None','None'),
    (8, 8, 8, 8, 72.3, 174.5, 'Family history of high blood
pressure','Hypertension', 'None', 'Cholecystectomy'),
    (9, 9, 9, 9, 69.6, 169.8, 'Family history of arthritis',
'Arthritis','None', 'None'),
```

```
    (10, 10, 10, 10, 73.0, 176.0, 'No significant family history',
'None','None', 'Appendectomy');
```

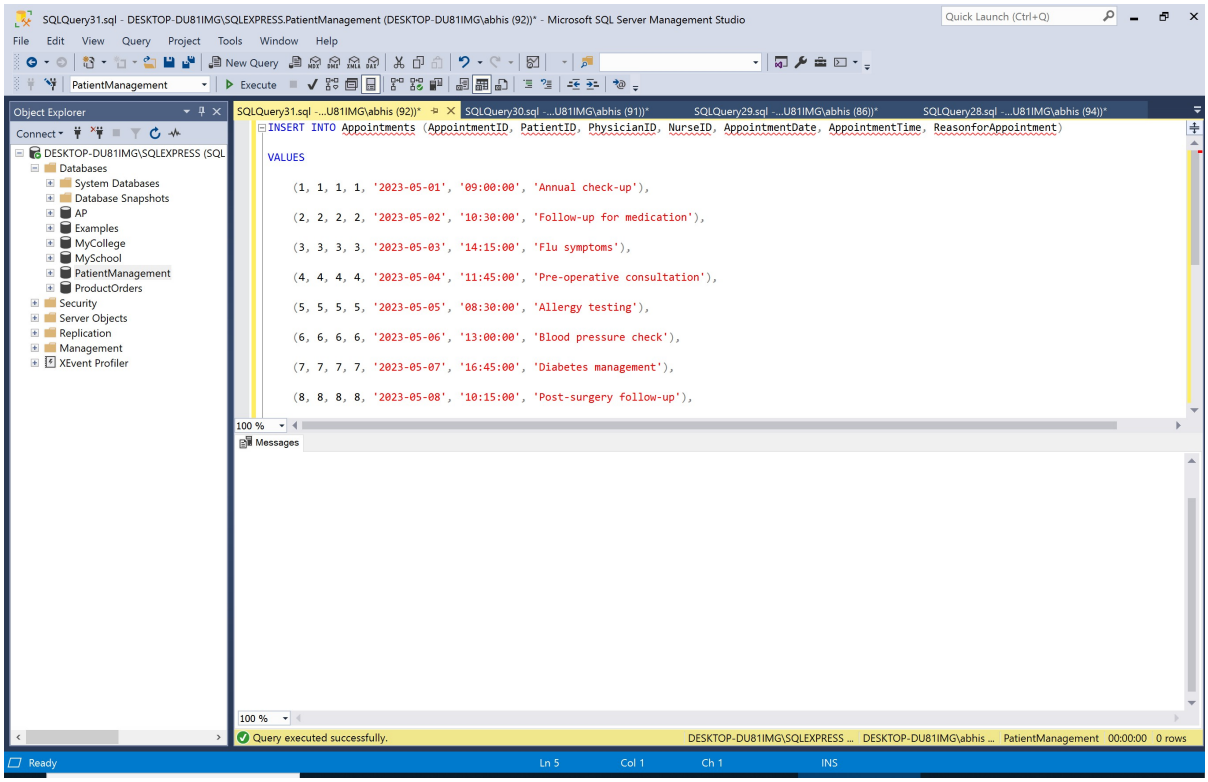


Appointment

sQuery:

```
INSERT INTO Appointments (AppointmentID, PatientID, PhysicianID,
NurseID,AppointmentDate, AppointmentTime, ReasonforAppointment)
VALUES
    (1  1   1  1   '2023-05-01   '09:00:00   'Annual check-up'),
    ,   ,   ,  ,       ',          ',
    (2  2   2  2   '2023-05-02   '10:30:00   'Follow-up for medication'),
    ,   ,   ,  ,       ',          ',
    (3  3   3  3   '2023-05-03   '14:15:00   'Flu symptoms'),
    ,   ,   ,  ,       ',          ',
    (4  4   4  4   '2023-05-04   '11:45:00   'Pre-operative
    ,   ,   ,  ,       ',          ',       consultation'),
    (5  5   5  5   '2023-05-05   '08:30:00   'Allergy testing'),
    ,   ,   ,  ,       ',          ',
    (6  6   6  6   '2023-05-06   '13:00:00   'Blood pressure check'),
    ,   ,   ,  ,       ',          ',
```

```
(7  7   7  7   '2023-05-07   '16:45:00   'Diabetes management'),
,   ,   ,  ,       ',           ',

(8  8   8  8   '2023-05-08   '10:15:00   'Post-surgery follow-up'),
,   ,   ,  ,       ',           ',

(9  9   9  9   '2023-05-09   '12:30:00   'Arthritis pain'),
,   ,   ,  ,       ',           ',

(10, 10, 10, 10, '2023-05-10', '15:30:00', 'Routine vaccination');
```

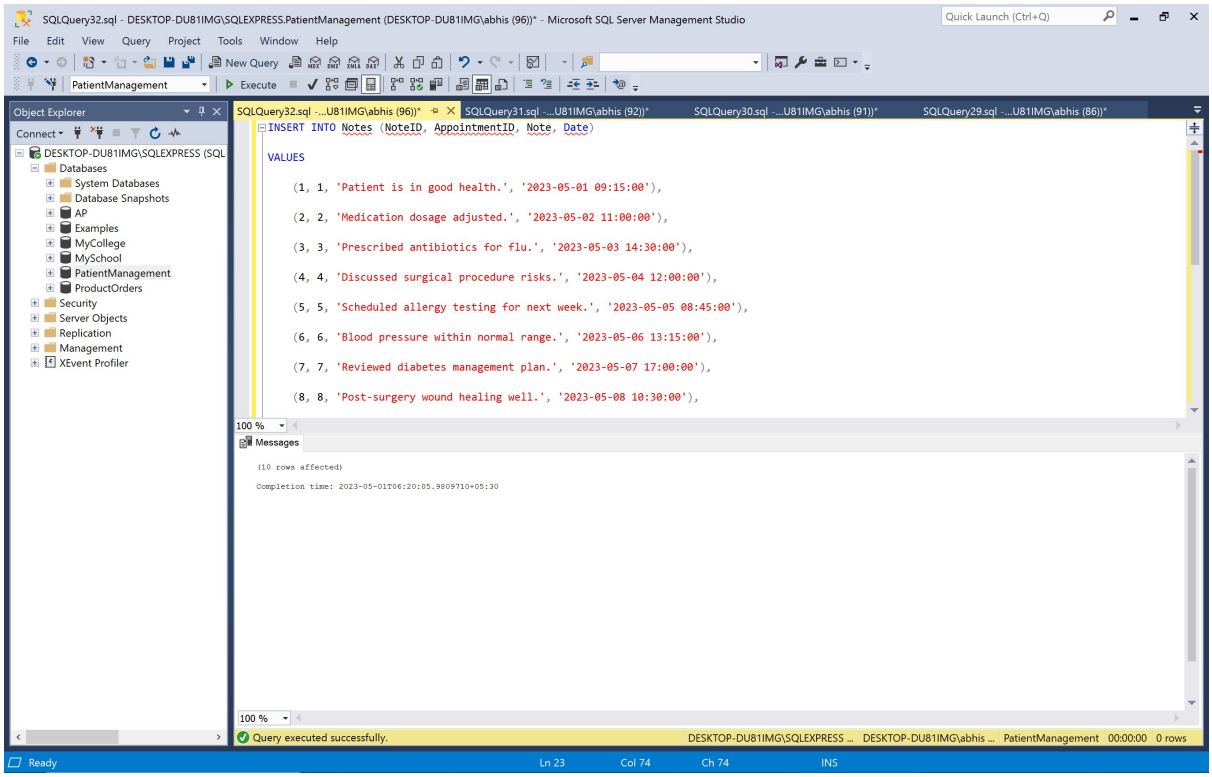


Notes

Query:

```
INSERT INTO Notes (NoteID, AppointmentID, Note,
Date)VALUES
     (1, 1, 'Patient is in good health.', '2023-05-01 09:15:00'),
     (2, 2, 'Medication dosage adjusted.', '2023-05-02 11:00:00'),
     (3, 3, 'Prescribed antibiotics for flu.', '2023-05-03 14:30:00'),
     (4, 4, 'Discussed surgical procedure risks.', '2023-05-04 12:00:00'),
     (5, 5, 'Scheduled allergy testing for next week.', '2023-05-05 08:45:00'),
     (6, 6, 'Blood pressure within normal range.', '2023-05-06 13:15:00'),
     (7, 7, 'Reviewed diabetes management plan.', '2023-05-07 17:00:00'),
     (8, 8, 'Post-surgery wound healing well.', '2023-05-08 10:30:00'),
```

```
(9  9   'Prescribed pain medication for arthritis.', '2023-05-09
,   ,   13:00:00'),

(10, 10, 'Administered routine vaccination.', '2023-05-10 15:45:00');
```



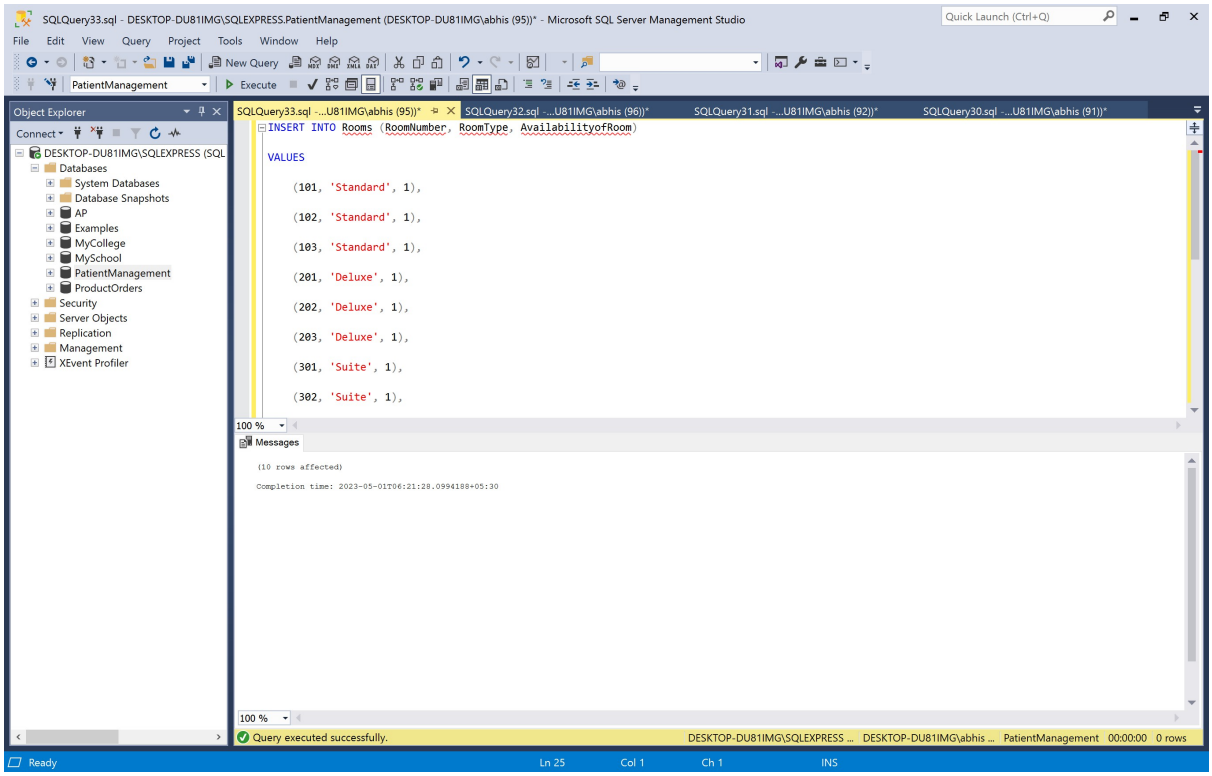
Room

s

Query:

```
INSERT INTO Rooms (RoomNumber, RoomType,
AvailabilityofRoom)VALUES

      (10   'Standard',
      1,    1),

      (10   'Standard',
      2,    1),

      (10   'Standard',
      3,    1),

      (20   'Deluxe', 1),
      1,

      (20   'Deluxe', 1),
      2,

      (20   'Deluxe', 1),
      3,
```

```
(30   'Suite', 1),
1,

(30   'Suite', 1),
2,

(30   'Suite', 1),
3,

(40   'VIP Suite',
1,    1);
```

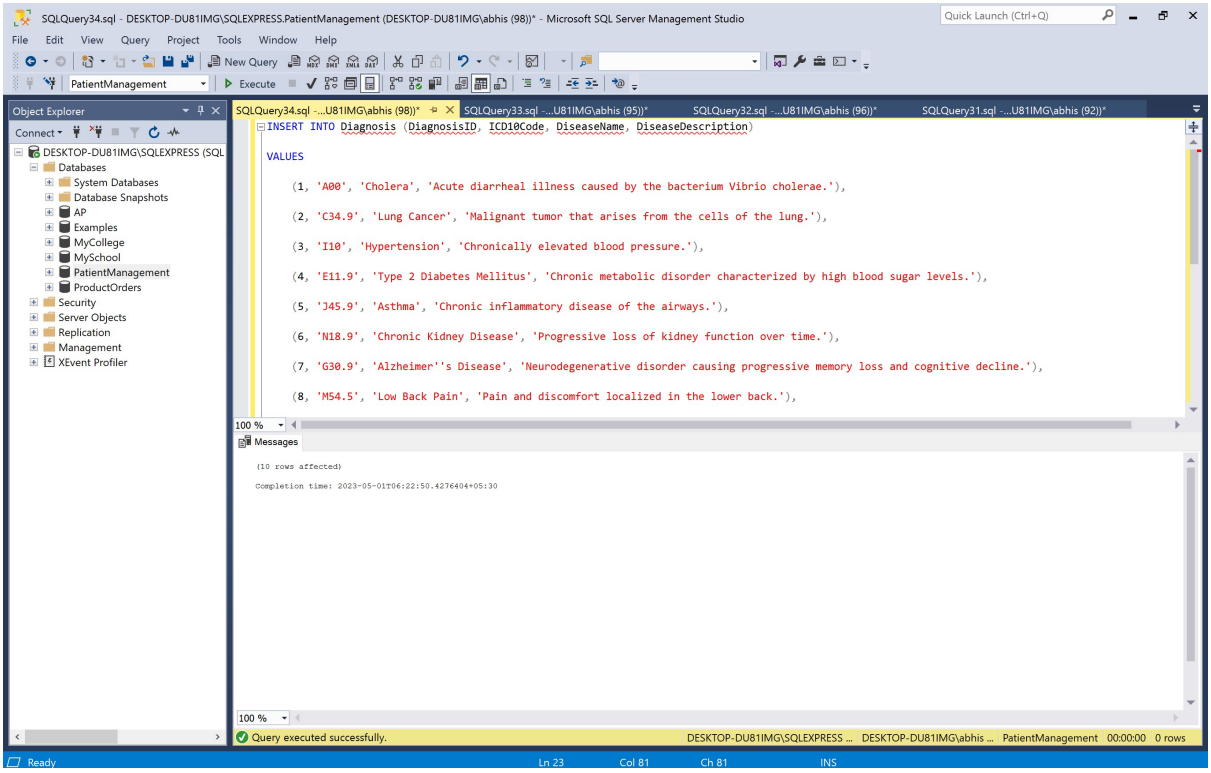


Diagnosi

sQuery:

```
INSERT INTO Diagnosis (DiagnosisID, ICD10Code, DiseaseName,
DiseaseDescription)VALUES
    (1, 'A00', 'Cholera', 'Acute diarrheal illness caused by the bacterium
Vibrio cholerae.'),
    (2, 'C34.9', 'Lung Cancer', 'Malignant tumor that arises from the cells of
the lung.'),
    (3, 'I10', 'Hypertension', 'Chronically elevated blood pressure.'),
    (4, 'E11.9', 'Type 2 Diabetes Mellitus', 'Chronic metabolic disorder
characterized by high blood sugar levels.'),
    (5, 'J45.9', 'Asthma', 'Chronic inflammatory disease of the airways.'),
    (6, 'N18.9', 'Chronic Kidney Disease', 'Progressive loss of kidney function
over time.'),
    (7, 'G30.9', 'Alzheimer''s Disease', 'Neurodegenerative disorder causing
progressive memory loss and cognitive decline.'),
    (8, 'M54.5', 'Low Back Pain', 'Pain and discomfort localized in the lower
back.'),
    (9, 'G47.0', 'Insomnia', 'Sleep disorder characterized by difficulty
falling asleep or staying asleep.'),
```

```
    (10, 'R07.9', 'Chest Pain', 'Pain or discomfort felt in the chest
region.');
```

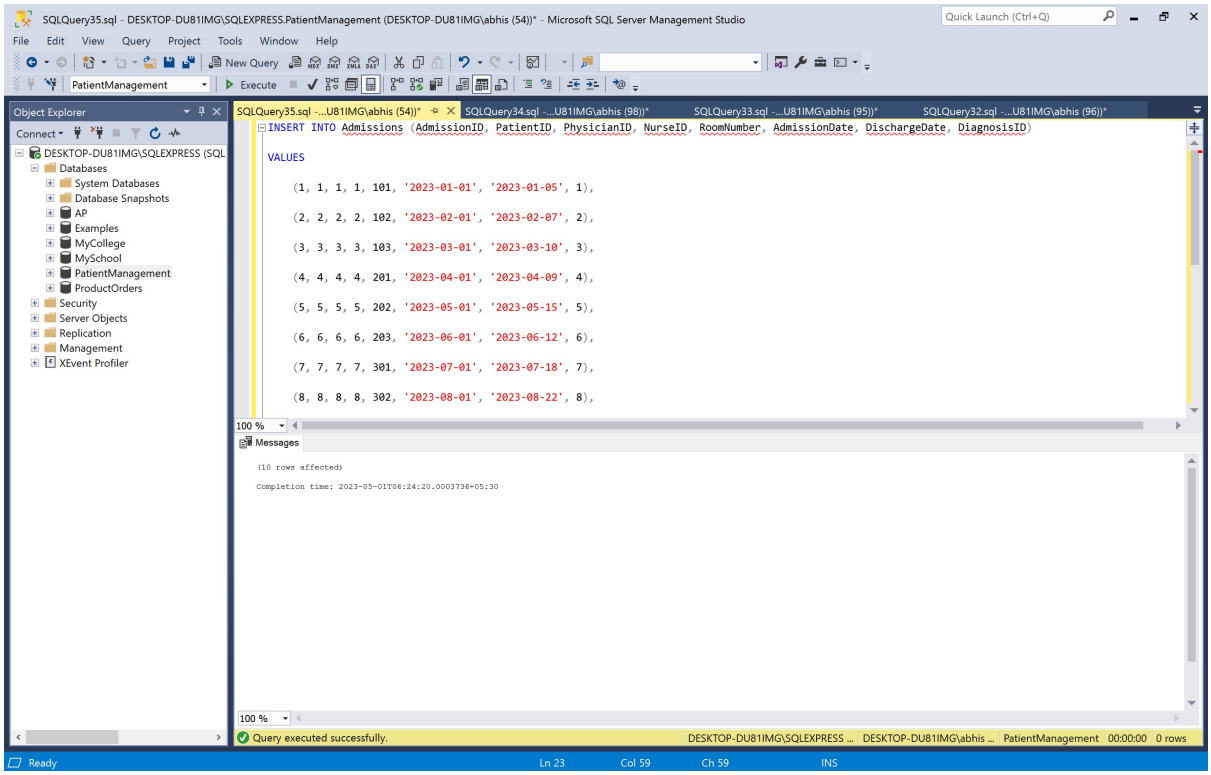


Admission

sQuery:

```
INSERT INTO Admissions (AdmissionID, PatientID, PhysicianID,
NurseID,RoomNumber, AdmissionDate, DischargeDate, DiagnosisID)
VALUES
```

```
(1, 1, 1, 1, 101, '2023-01-01', '2023-01-05', 1),
(2, 2, 2, 2, 102, '2023-02-01', '2023-02-07', 2),
(3, 3, 3, 3, 103, '2023-03-01', '2023-03-10', 3),
(4, 4, 4, 4, 201, '2023-04-01', '2023-04-09', 4),
(5, 5, 5, 5, 202, '2023-05-01', '2023-05-15', 5),
(6, 6, 6, 6, 203, '2023-06-01', '2023-06-12', 6),
(7, 7, 7, 7, 301, '2023-07-01', '2023-07-18', 7),
(8, 8, 8, 8, 302, '2023-08-01', '2023-08-22', 8),
(9, 9, 9, 9, 303, '2023-09-01', '2023-09-25', 9),
(10, 10, 10, 10, 401, '2023-10-01', '2023-10-30', 10);
```

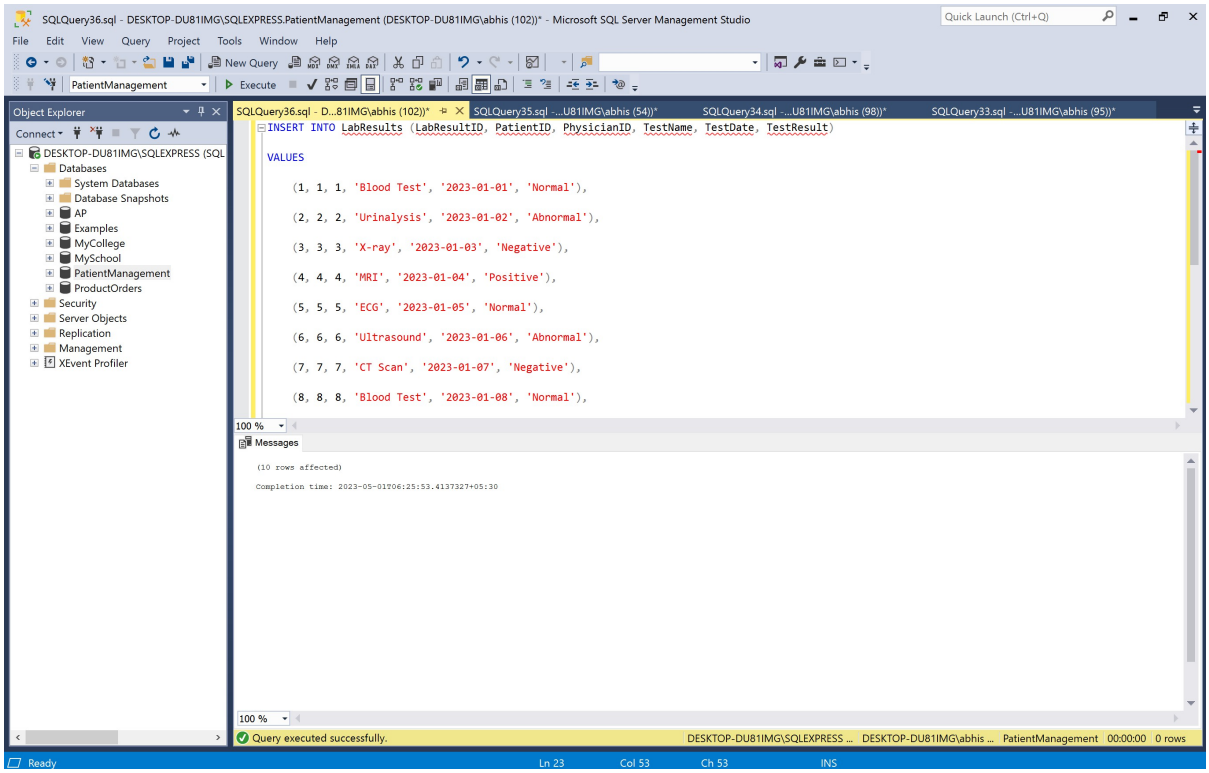


LabResult

sQuery:

```
INSERT INTO LabResults (LabResultID, PatientID, PhysicianID,
TestName,TestDate, TestResult)
VALUES
     (1  1   1  'Blood Test', '2023-01-01', 'Normal'),
     ,   ,   ,

     (2  2   2  'Urinalysis', '2023-01-02',
     ,   ,   ,  'Abnormal'),

     (3  3   3  'X-ray', '2023-01-03', 'Negative'),
     ,   ,   ,

     (4  4   4  'MRI', '2023-01-04', 'Positive'),
     ,   ,   ,

     (5  5   5  'ECG', '2023-01-05', 'Normal'),
     ,   ,   ,

     (6  6   6  'Ultrasound', '2023-01-06',
     ,   ,   ,  'Abnormal'),

     (7  7   7  'CT Scan', '2023-01-07', 'Negative'),
     ,   ,   ,
```

```
(8  8   8  'Blood Test', '2023-01-08', 'Normal'),
,   ,   ,

(9  9   9  'Urinalysis', '2023-01-09',
,   ,   ,  'Abnormal'),

(10, 10, 10, 'X-ray', '2023-01-10', 'Negative');
```

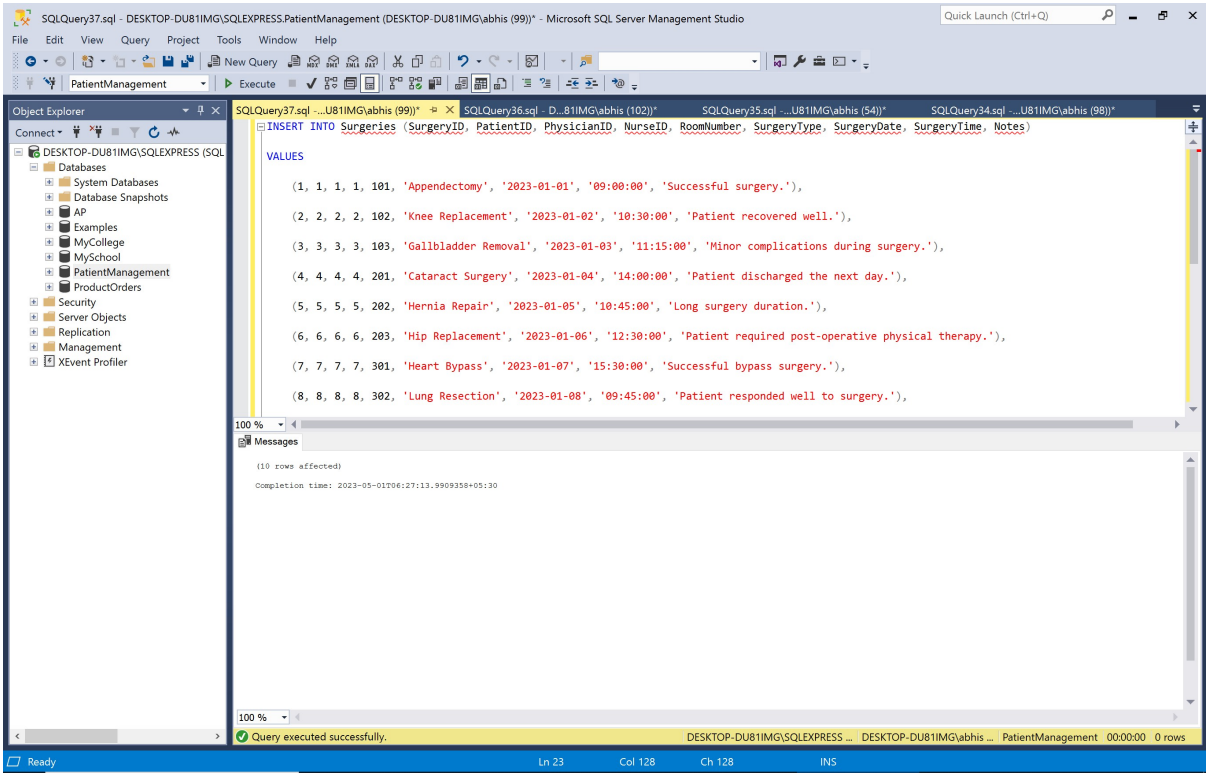


Surgerie

sQuery:

```
INSERT INTO Surgeries (SurgeryID, PatientID, PhysicianID, NurseID,
RoomNumber,SurgeryType, SurgeryDate, SurgeryTime, Notes)
VALUES
    (1, 1, 1, 1, 101, 'Appendectomy', '2023-01-01', '09:00:00',
'Successfulsurgery.'),
    (2, 2, 2, 2, 102, 'Knee Replacement', '2023-01-02', '10:30:00',
'Patientrecovered well.'),
    (3, 3, 3, 3, 103, 'Gallbladder Removal', '2023-01-03', '11:15:00',
'Minorcomplications during surgery.'),
    (4, 4, 4, 4, 201, 'Cataract Surgery', '2023-01-04', '14:00:00',
'Patientdischarged the next day.'),
    (5, 5, 5, 5, 202, 'Hernia Repair', '2023-01-05', '10:45:00', 'Long
surgeryduration.'),
    (6, 6, 6, 6, 203, 'Hip Replacement', '2023-01-06', '12:30:00', 'Patient
required post-operative physical therapy.'),
    (7, 7, 7, 7, 301, 'Heart Bypass', '2023-01-07', '15:30:00', 'Successful
bypass surgery.'),
```

```
    (8, 8, 8, 8, 302, 'Lung Resection', '2023-01-08', '09:45:00', 'Patient
responded well to surgery.'),
    (9, 9, 9, 9, 303, 'Tonsillectomy', '2023-01-09', '11:30:00', 'Routine
surgery.'),
    (10, 10, 10, 10, 401, 'Spinal Fusion', '2023-01-10', '13:15:00', 'Complex
procedure requiring extensive recovery period.');
```

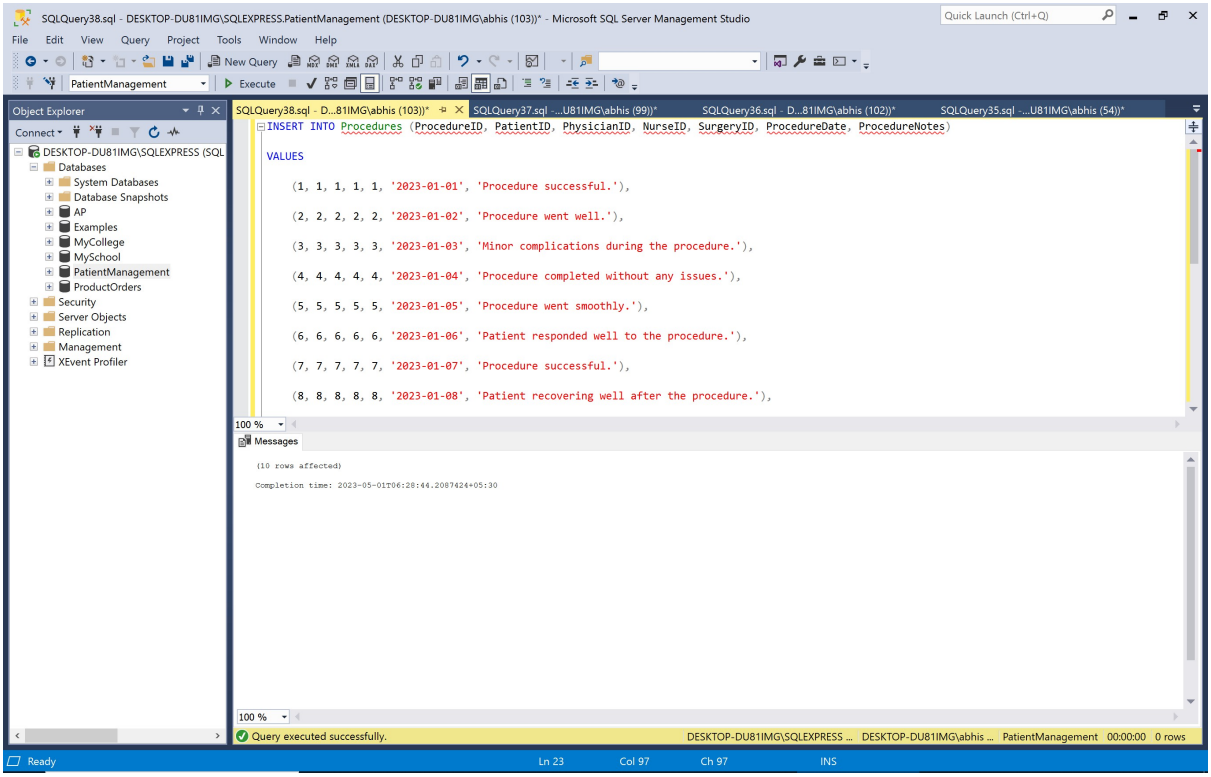


Procedure

sQuery:

```
INSERT INTO Procedures (ProcedureID, PatientID, PhysicianID,
NurseID,SurgeryID, ProcedureDate, ProcedureNotes)
VALUES
    (1,  1   1  1  1   '2023-01-01   'Procedure successful.'),
         ,   ,  ,  ,        ',
    (2,  2   2  2  2   '2023-01-02   'Procedure went well.'),
         ,   ,  ,  ,        ',
    (3,  3   3  3  3   '2023-01-03   'Minor complications during the
         ,   ,  ,  ,        ',       procedure.'),
    (4,  4   4  4  4   '2023-01-04   'Procedure completed without any issues.'),
         ,   ,  ,  ,        ',
    (5,  5   5  5  5   '2023-01-05   'Procedure went smoothly.'),
         ,   ,  ,  ,        ',
    (6,  6   6  6  6   '2023-01-06   'Patient responded well to the procedure.'),
         ,   ,  ,  ,        ',
    (7,  7   7  7  7   '2023-01-07   'Procedure successful.'),
         ,   ,  ,  ,        ',
```

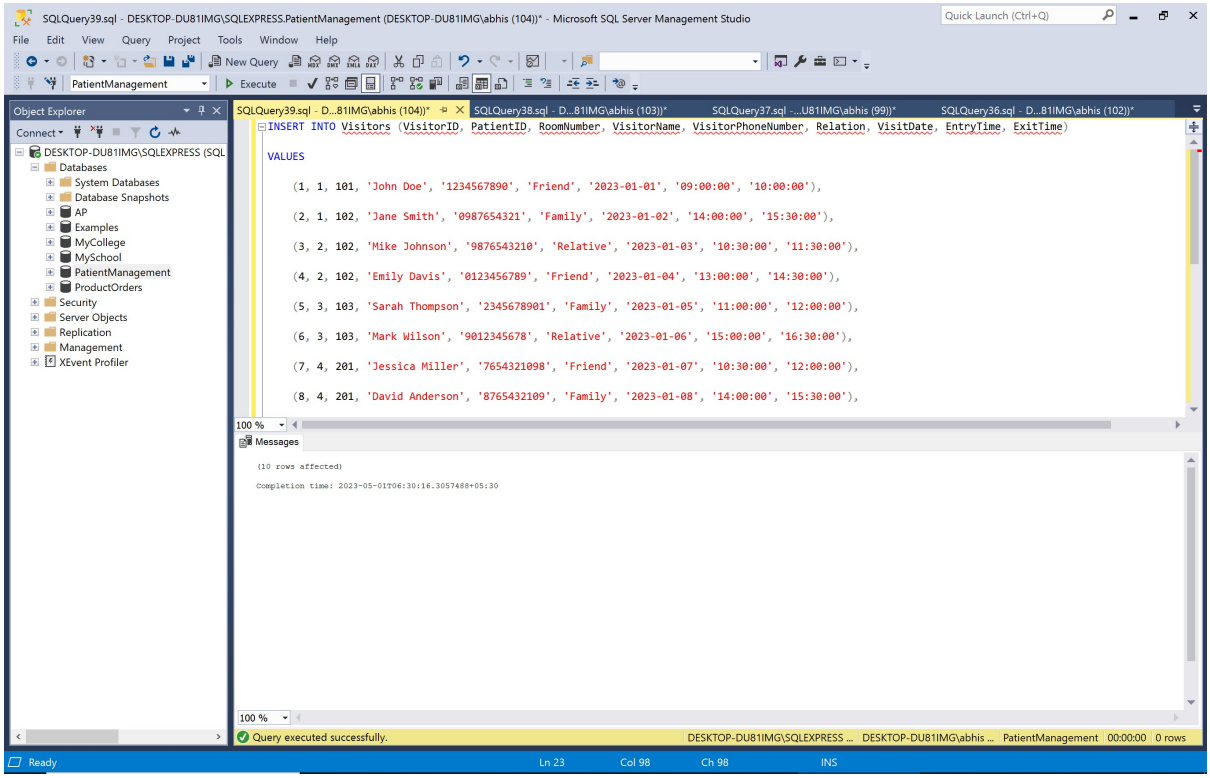
```
    (8,  8   8  8  8   '2023-01-08   'Patient recovering well after the
         ,   ,  ,  ,        ',

 procedure.'
 ),

     (9, 9,    9  9   '2023-01-09   'Routine procedure.'),
     9,        ,  ,        ',

    (10, 10, 10, 10, 10, '2023-01-10', 'Complex procedure with extensive post-
operative care.');
```



Visitors

Query:

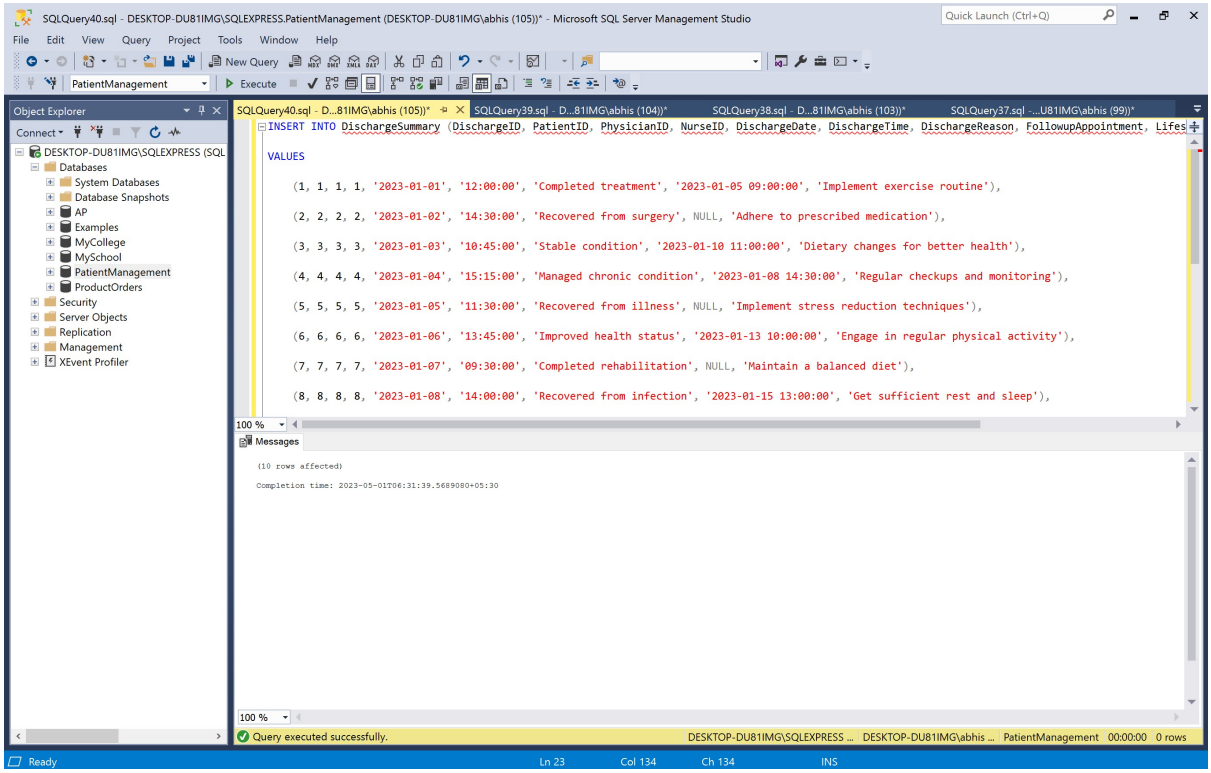
```
INSERT INTO Visitors (VisitorID, PatientID, RoomNumber,
VisitorName,VisitorPhoneNumber, Relation, VisitDate, EntryTime,
ExitTime)
VALUES
    (1, 1, 101, 'John Doe', '1234567890', 'Friend', '2023-01-01', '09:00:00',
'10:00:00'),
    (2, 1, 102, 'Jane Smith', '0987654321', 'Family', '2023-01-02', '14:00:00',
'15:30:00'),
    (3, 2, 102, 'Mike Johnson', '9876543210', 'Relative', '2023-01-03',
'10:30:00', '11:30:00'),
    (4, 2, 102, 'Emily Davis', '0123456789', 'Friend', '2023-01-04',
'13:00:00', '14:30:00'),
    (5, 3, 103, 'Sarah Thompson', '2345678901', 'Family', '2023-01-05',
'11:00:00', '12:00:00'),
    (6, 3, 103, 'Mark Wilson', '9012345678', 'Relative', '2023-01-06',
'15:00:00', '16:30:00'),
    (7, 4, 201, 'Jessica Miller', '7654321098', 'Friend', '2023-01-07',
'10:30:00', '12:00:00'),
    (8, 4, 201, 'David Anderson', '8765432109', 'Family', '2023-01-08',
'14:00:00', '15:30:00'),
    (9, 5, 202, 'Amy Brown', '4567890123', 'Relative', '2023-01-09',
'11:30:00', '12:30:00'),
    (10, 5, 202, 'Michael Wilson', '5678901234', 'Friend', '2023-01-10',
'13:00:00', '14:00:00');
```



DischargeSummar

yQuery:

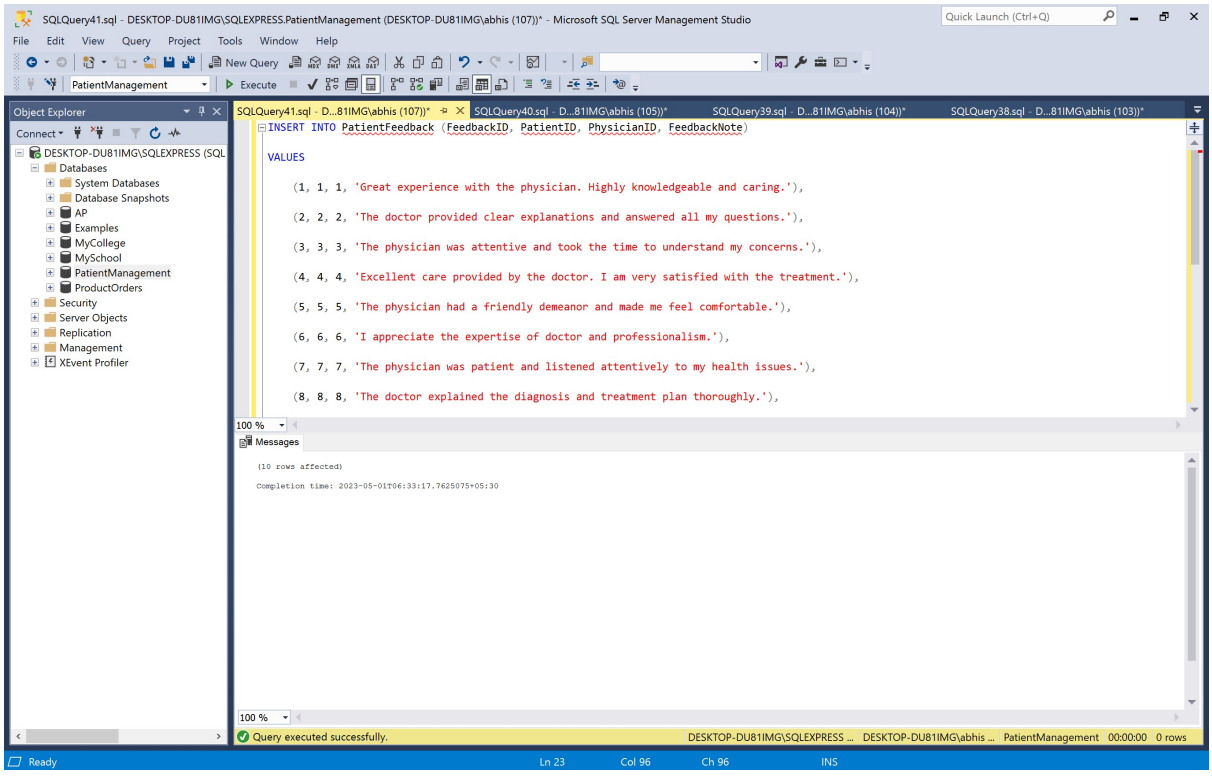
```
INSERT INTO DischargeSummary (DischargeID, PatientID, PhysicianID,
NurseID,DischargeDate, DischargeTime, DischargeReason, FollowupAppointment,
LifestyleChanges)
VALUES
    (1, 1, 1, 1, '2023-01-01', '12:00:00', 'Completed treatment', '2023-01-05
09:00:00', 'Implement exercise routine'),
    (2, 2, 2, 2, '2023-01-02', '14:30:00', 'Recovered from surgery',
NULL,'Adhere to prescribed medication'),
    (3, 3, 3, 3, '2023-01-03', '10:45:00', 'Stable condition', '2023-01-10
11:00:00', 'Dietary changes for better health'),
    (4, 4, 4, 4, '2023-01-04', '15:15:00', 'Managed chronic condition',
'2023-01-08 14:30:00', 'Regular checkups and monitoring'),
    (5, 5, 5, 5, '2023-01-05', '11:30:00', 'Recovered from illness',
NULL,'Implement stress reduction techniques'),
    (6, 6, 6, 6, '2023-01-06', '13:45:00', 'Improved health
status','2023-01-13 10:00:00', 'Engage in regular physical
activity'),
    (7, 7, 7, 7, '2023-01-07', '09:30:00', 'Completed rehabilitation',
NULL,'Maintain a balanced diet'),
    (8, 8, 8, 8, '2023-01-08', '14:00:00', 'Recovered from
infection','2023-01-15 13:00:00', 'Get sufficient rest and sleep'),
    (9, 9, 9, 9, '2023-01-09', '11:15:00', 'Managed acute symptoms',
NULL,'Avoid smoking and alcohol consumption'),
    (10, 10, 10, 10, '2023-01-10', '12:30:00', 'Stable vital signs',
'2023-01-17 15:00:00', 'Practice stress management techniques');
```



PatientFeedbac

kQuery:

```
INSERT INTO PatientFeedback (FeedbackID, PatientID, PhysicianID,
FeedbackNote)VALUES
    (1, 1, 1, 'Great experience with the physician. Highly knowledgeable and
caring.'),
    (2, 2, 2, 'The doctor provided clear explanations and answered all my
questions.'),
    (3, 3, 3, 'The physician was attentive and took the time to understand my
concerns.'),
    (4, 4, 4, 'Excellent care provided by the doctor. I am very satisfied with
the treatment.'),
    (5, 5, 5, 'The physician had a friendly demeanor and made me feel
comfortable.'),
    (6, 6, 6, 'I appreciate the expertise of doctor and professionalism.'),
    (7, 7, 7, 'The physician was patient and listened attentively to my health
issues.'),
    (8, 8, 8, 'The doctor explained the diagnosis and treatment plan
thoroughly.'),
    (9, 9, 9, 'I am grateful for the support of physician and guidance during
my recovery.'),
    (10, 10, 10, 'The expertise of doctor and care have made a positive impact
on my health.');
```



C. TESTING

This phase includes the following:

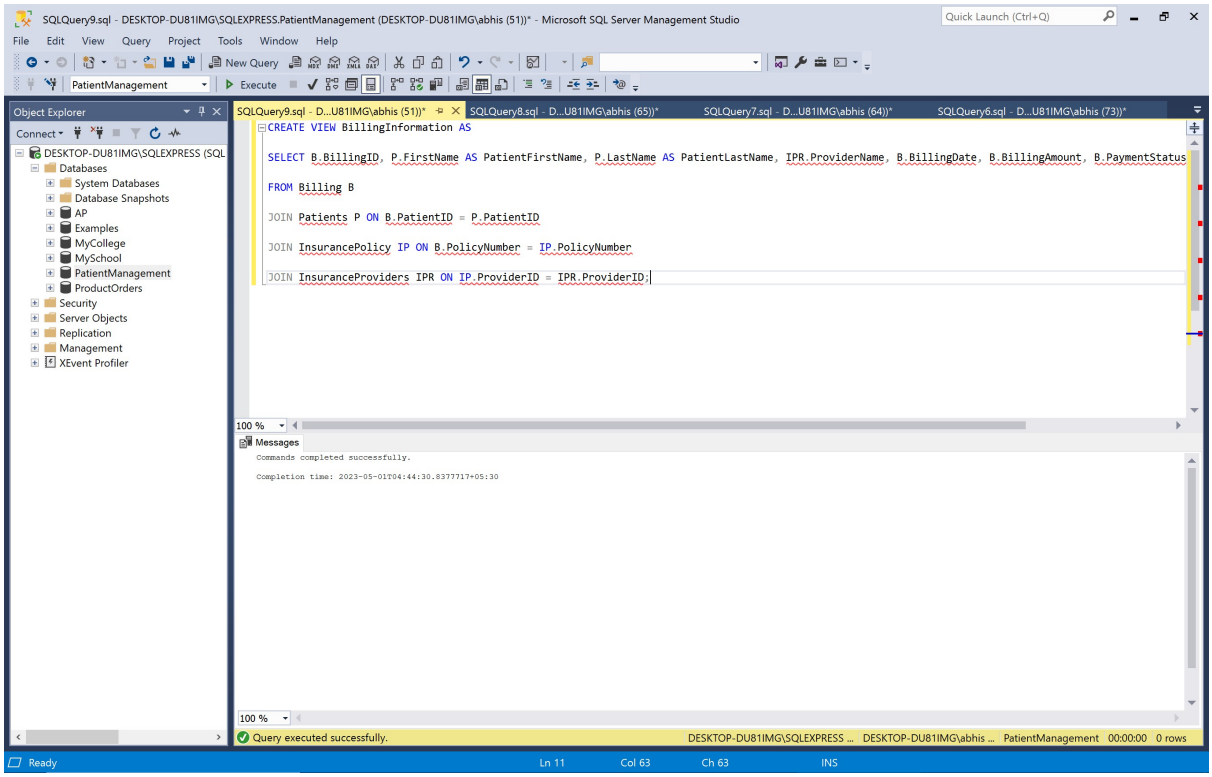
1. Populating the database with test data
2. Producing various reports
3. Demonstrating the reliability through several scenarios
4. Performing several transactions.

Create 4 views, 4 stored procedures, 4 user defined functions, 4 triggers, and 4 transactions

Creation of views:

1. BillingInformation

```
CREATE VIEW BillingInformation AS
SELECT B.BillingID, P.FirstName AS PatientFirstName, P.LastName
AS PatientLastName, IPR.ProviderName, B.BillingDate,
B.BillingAmount,B.PaymentStatus
FROM Billing B
JOIN Patients P ON B.PatientID = P.PatientID
JOIN InsurancePolicy IP ON B.PolicyNumber = IP.PolicyNumber
JOIN InsuranceProviders IPR ON IP.ProviderID = IPR.ProviderID;
```



Comments:

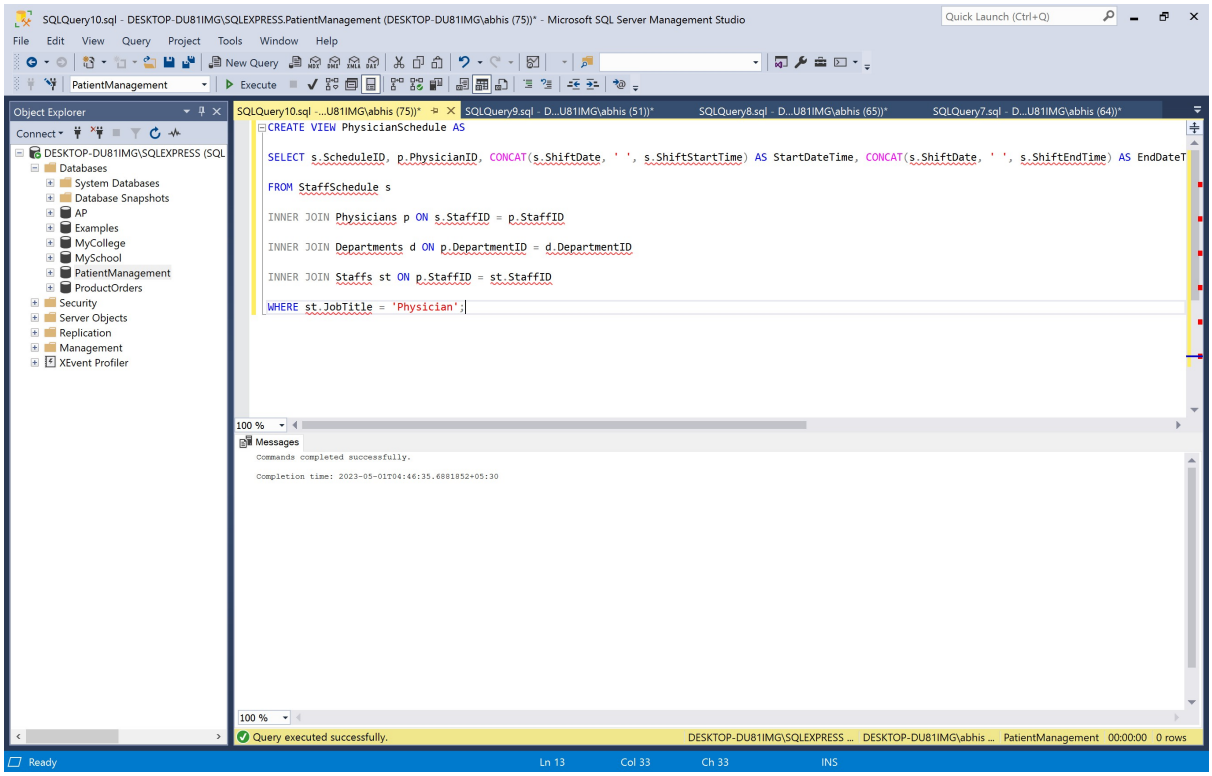
The above screenshot depicts the output for the view. This view retrieves information about the bills of patients which includes patient information and insurance provider details.

2. PhysiciansSchedule:

```
CREATE VIEW PhysicianSchedule AS
SELECT s.ScheduleID, p.PhysicianID, CONCAT(s.ShiftDate, ' ',
s.ShiftStartTime)AS StartDateTime, CONCAT(s.ShiftDate, ' ', s.ShiftEndTime) AS
EndDateTime, d.DepartmentName, CONCAT(st.FirstName, ' ', st.LastName) AS
PhysicianName
FROM StaffSchedule s
INNER JOIN Physicians p ON s.StaffID = p.StaffID
INNER JOIN Departments d ON p.DepartmentID =
d.DepartmentIDINNER JOIN Staffs st ON p.StaffID =
st.StaffID
WHERE st.JobTitle = 'Physician';
```

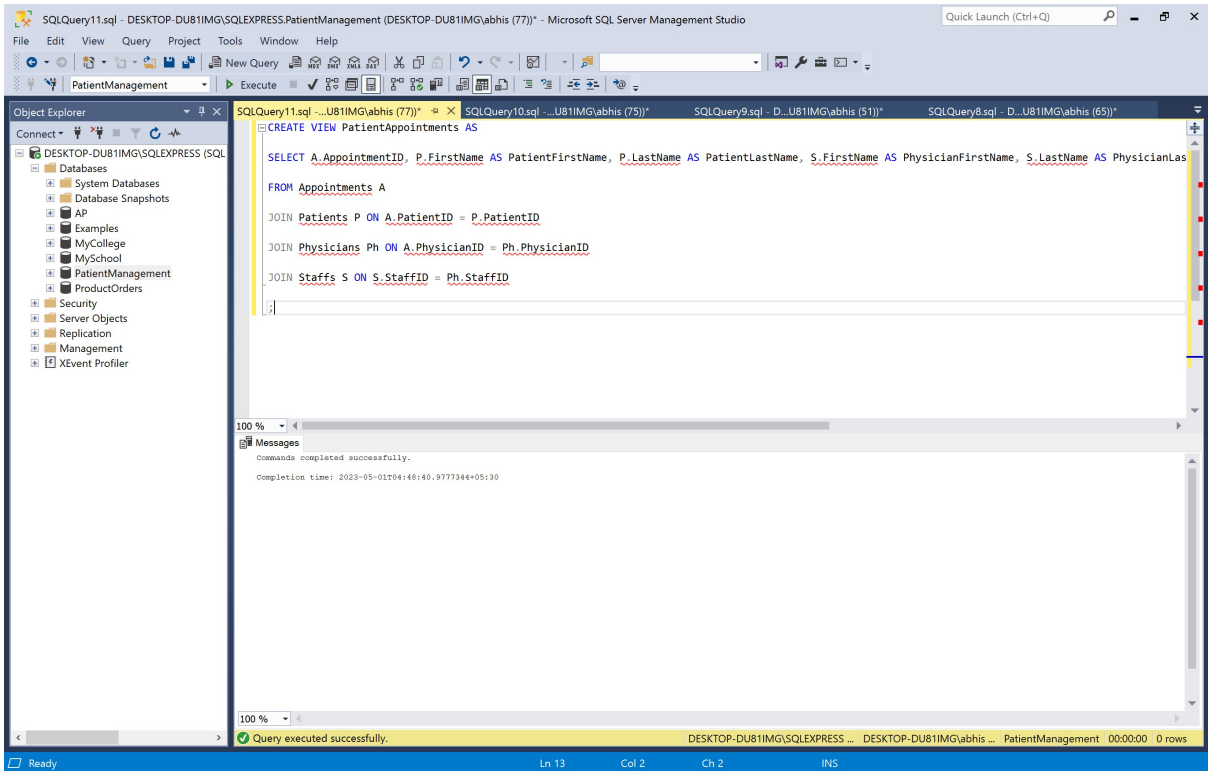
Comments:

The above screenshot depicts the output when the view is created. The view is named as PhysiciansSchedule. This view can be used to display the schedules of physicians using the physicians table.



3. PatientAppointments

```
CREATE VIEW PatientAppointments AS
SELECT A.AppointmentID, P.FirstName AS PatientFirstName, P.LastName AS
PatientLastName, S.FirstName AS PhysicianFirstName, S.LastName AS
PhysicianLastName, A.AppointmentDate, A.AppointmentTime,
A.ReasonforAppointmentFROM Appointments A
JOIN Patients P ON A.PatientID = P.PatientID
JOIN Physicians Ph ON A.PhysicianID =
Ph.PhysicianIDJOIN Staffs S ON S.StaffID =
Ph.StaffID
;
```

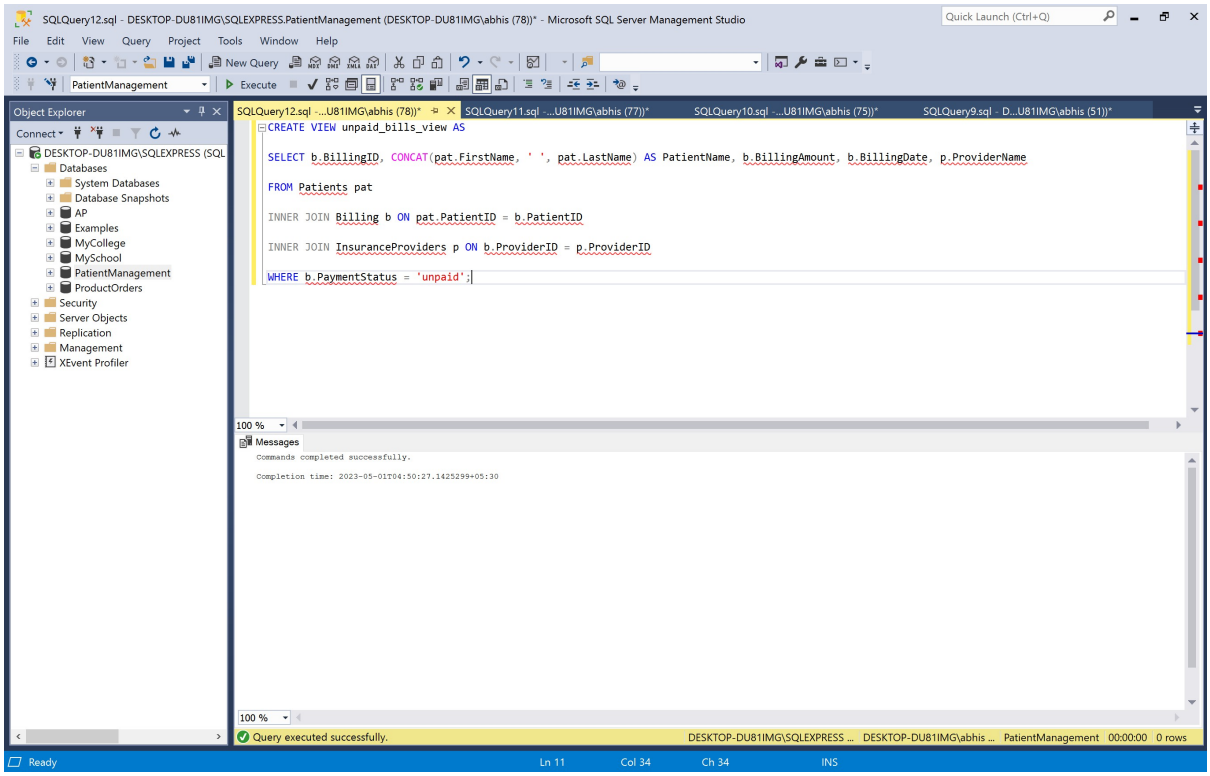


Comments:

The above screenshot depicts the output for the view named PatientAppointment. Overall, this code creates a view that combines patient medical records from the "PatientMedicalRecords" table with patient information from the "Patients" table and vital sign details from the "VitalSigns" table. It provides a consolidated view of relevant data for medical records analysis or reporting purposes.

4.UnpaidBills

```
CREATE VIEW unpaid_bills_view AS
SELECT b.BillingID, CONCAT(pat.FirstName, ' ', pat.LastName) AS
PatientName,b.BillingAmount, b.BillingDate, p.ProviderName
FROM Patients pat
INNER JOIN Billing b ON pat.PatientID = b.PatientID
INNER JOIN InsuranceProviders p ON b.ProviderID =
p.ProviderIDWHERE b.PaymentStatus = 'unpaid';
```



Comments:

The above screenshot depicts the output for the above query. This view will display the unpaid bills of patients, including their names and billing details. The view unpaid_bills_view has been successfully created. This view selects the BillingID, concatenated PatientName, BillingAmount, BillingDate, and ProviderName from the Patients, Billing, and InsuranceProviders tables, respectively. It only includes records where the PaymentStatus is set to 'unpaid'.

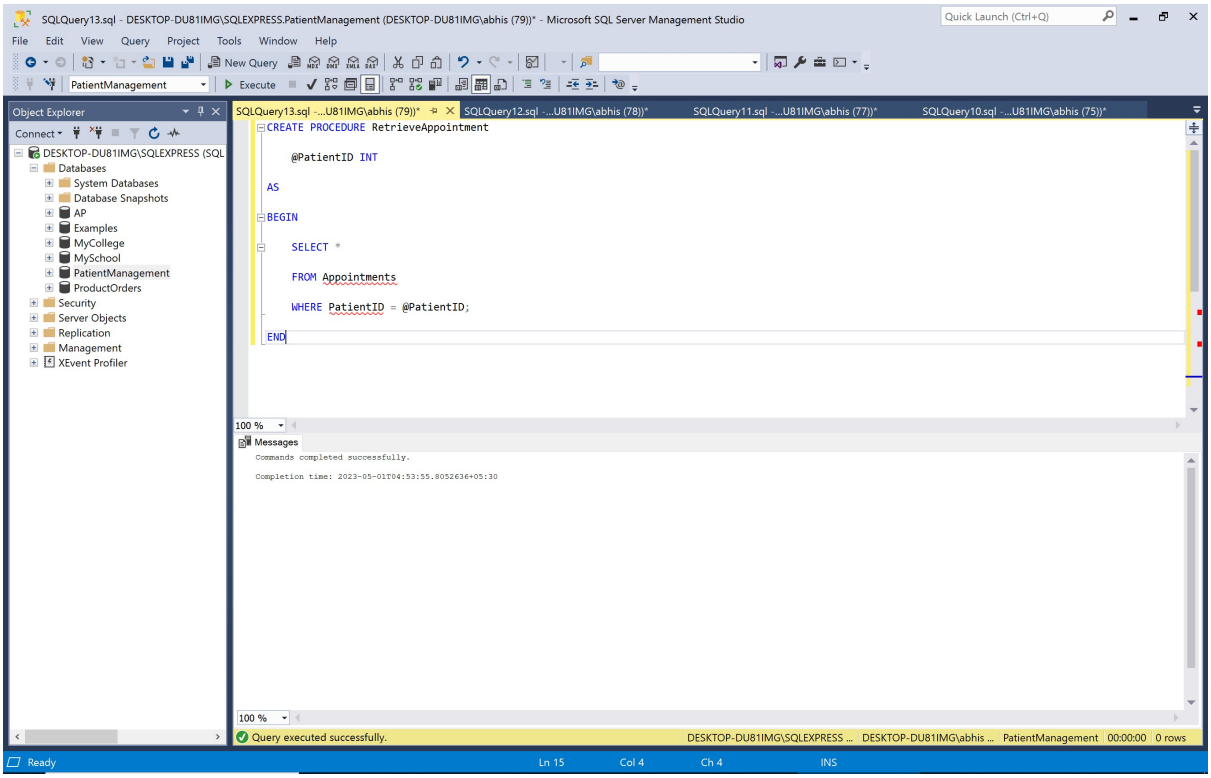
This view can be used to view the unpaid_bills_view in your queries to retrieve the unpaid billing information.

Stored Procedures

1. RetrieveAppointment

```
CREATE PROCEDURE
    RetrieveAppointment@PatientID
    INT
AS
BEGIN
    SELECT *
    FROM Appointments
    WHERE PatientID = @PatientID;
```

```
END
```



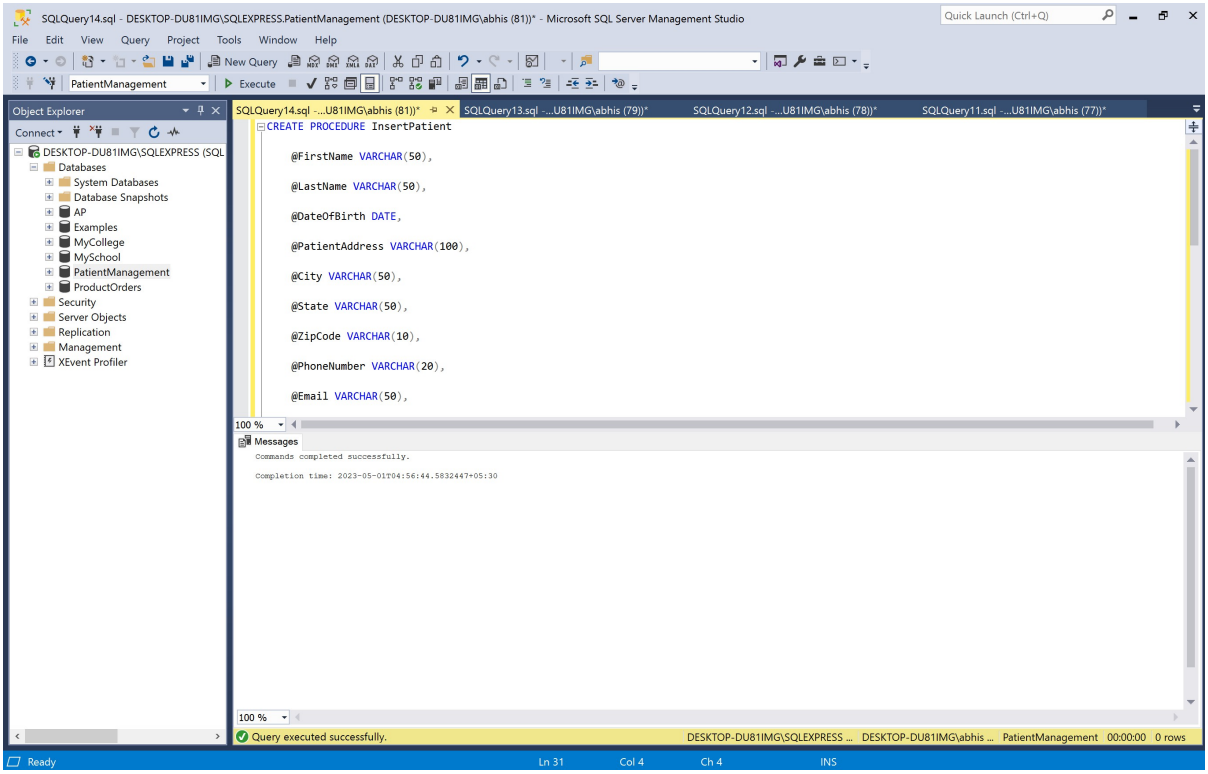
Comments:

The above screenshot displays the output for the given stored procedure query. This stored procedure retrieves all appointments for a specific patient. This procedure accepts an input parameter @PatientID of type INT and retrieves all appointments for the specified patient from the Appointments table. This stored procedure can be executed by passing the @PatientID parameter to retrieve the relevant appointment information.

2. InsertPatient

```
CREATE PROCEDURE InsertPatient
    @FirstName VARCHAR(50),
    @LastName VARCHAR(50),
    @DateOfBirth DATE,
    @PatientAddress
    VARCHAR(100),@City
    VARCHAR(50),
    @State VARCHAR(50),
    @ZipCode VARCHAR(10),
    @PhoneNumber
    VARCHAR(20),@Email
    VARCHAR(50),
    @Sex CHAR(1)
```

```
AS
BEGIN
    INSERT INTO Patients (FirstName, LastName, DateOfBirth,
PatientAddress,City, State, ZipCode, PhoneNumber, Email, Sex)
    VALUES (@FirstName, @LastName, @DateOfBirth, @PatientAddress,
@City,@State, @ZipCode, @PhoneNumber, @Email, @Sex);
END
```



Comments:

The stored procedure InsertPatient has been successfully created. This procedure allows you to insert a new patient record into the Patients table. It accepts the following input parameters:

@FirstName: VARCHAR(50) - First name of the patient.

@LastName: VARCHAR(50) - Last name of the patient.

@DateOfBirth: DATE - Date of birth of the patient.

@PatientAddress: VARCHAR(100) - Address of the patient.

@City: VARCHAR(50) - City of the patient's residence.

@State: VARCHAR(50) - State of the patient's residence.

@ZipCode: VARCHAR(10) - Zip code of the patient's residence.

@PhoneNumber: VARCHAR(20) - Phone number of the

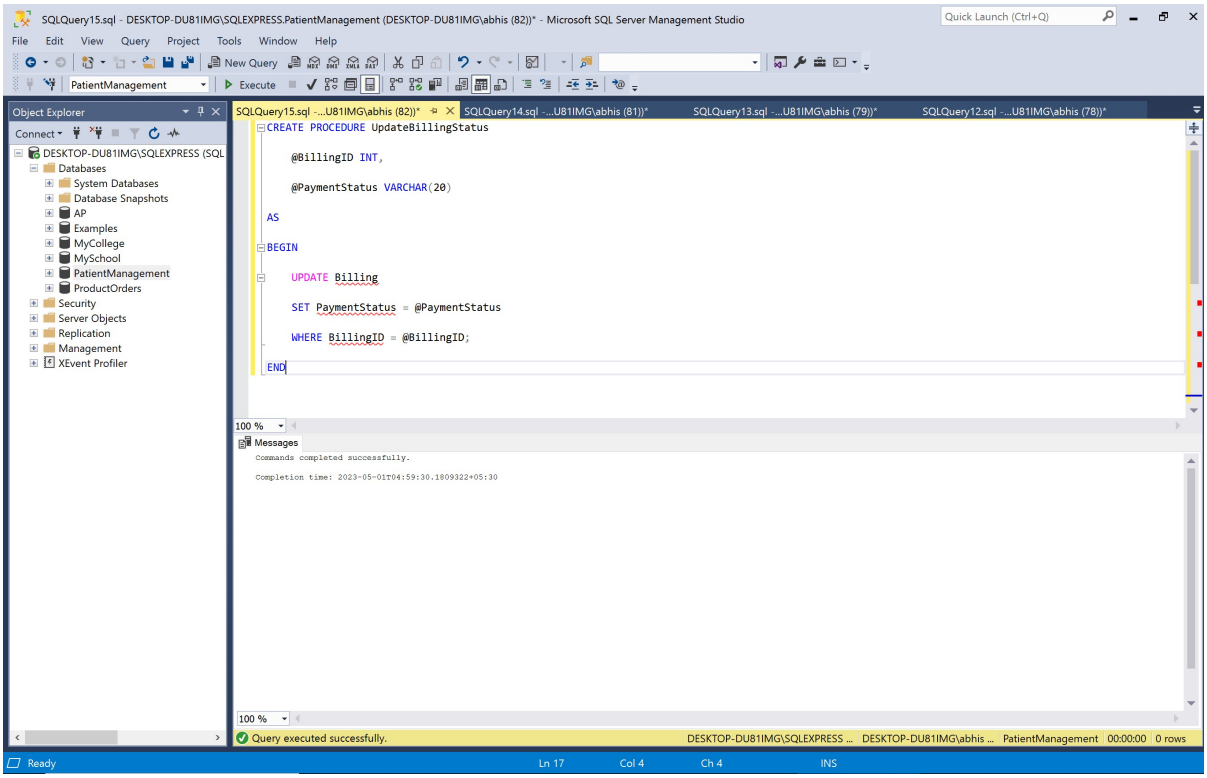
patient.@Email: VARCHAR(50) - Email address of the patient.

@Sex: CHAR(1) - Sex of the patient.

The stored procedure performs an INSERT operation, adding a new record to the Patients table with the provided input values.

3. UpdateBillingStatus

```
CREATE PROCEDURE
    UpdateBillingStatus@BillingID
    INT,
    @PaymentStatus VARCHAR(20)
AS
BEGIN
    UPDATE Billing
    SET PaymentStatus =
    @PaymentStatusWHERE BillingID =
    @BillingID;
END
```



Comments:

The stored procedure UpdateBillingStatus has been successfully created. This procedureallows to update the payment status of a billing record in the Billing table. It accepts thefollowing input parameters:

@BillingID: INT - The ID of the billing record to be updated.

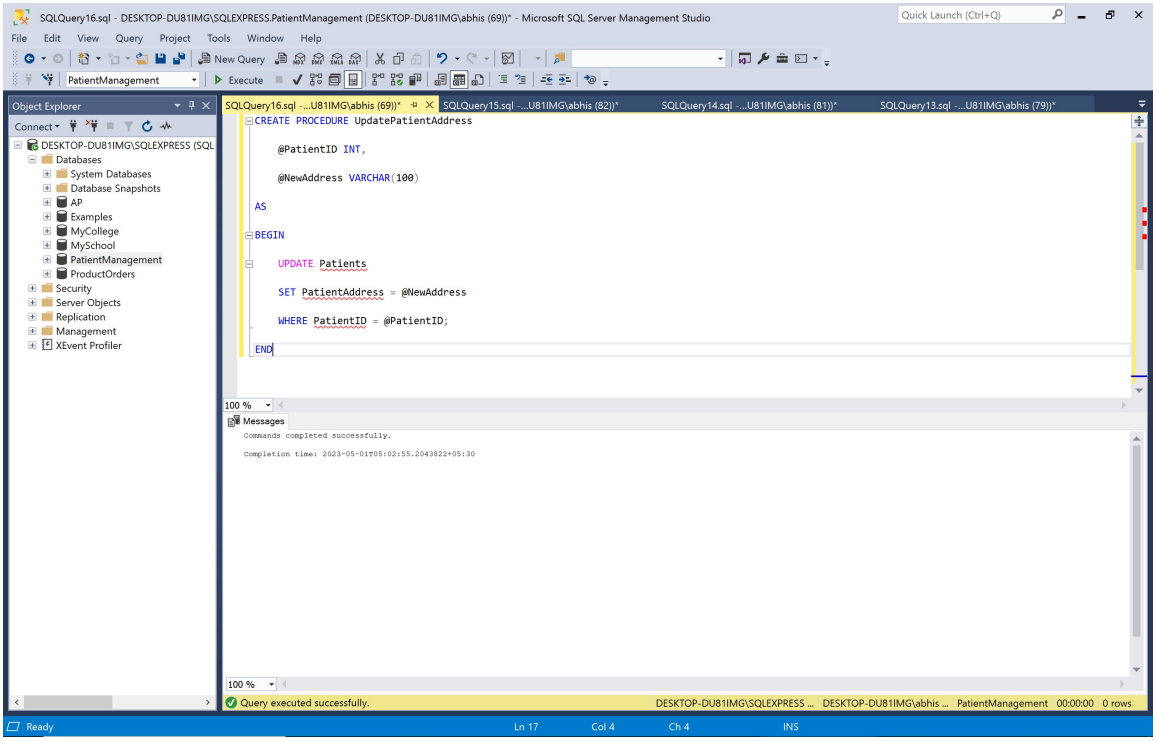
@PaymentStatus: VARCHAR(20) - The new payment status value.

The stored procedure performs an UPDATE operation, setting the PaymentStatus column to the provided @PaymentStatus value for the billing record with the matching BillingID in the Billing table.

This stored procedure can be used to execute, by passing the @BillingID and@PaymentStatus parameters to update the payment status of a billing record.

4.UpdatePatientAddress

```
CREATE PROCEDURE
    UpdatePatientAddress@PatientID
    INT,
    @NewAddress VARCHAR(100)
AS
BEGIN
    UPDATE Patients
    SET PatientAddress =
    @NewAddressWHERE PatientID =
    @PatientID;
END
```



Comments:

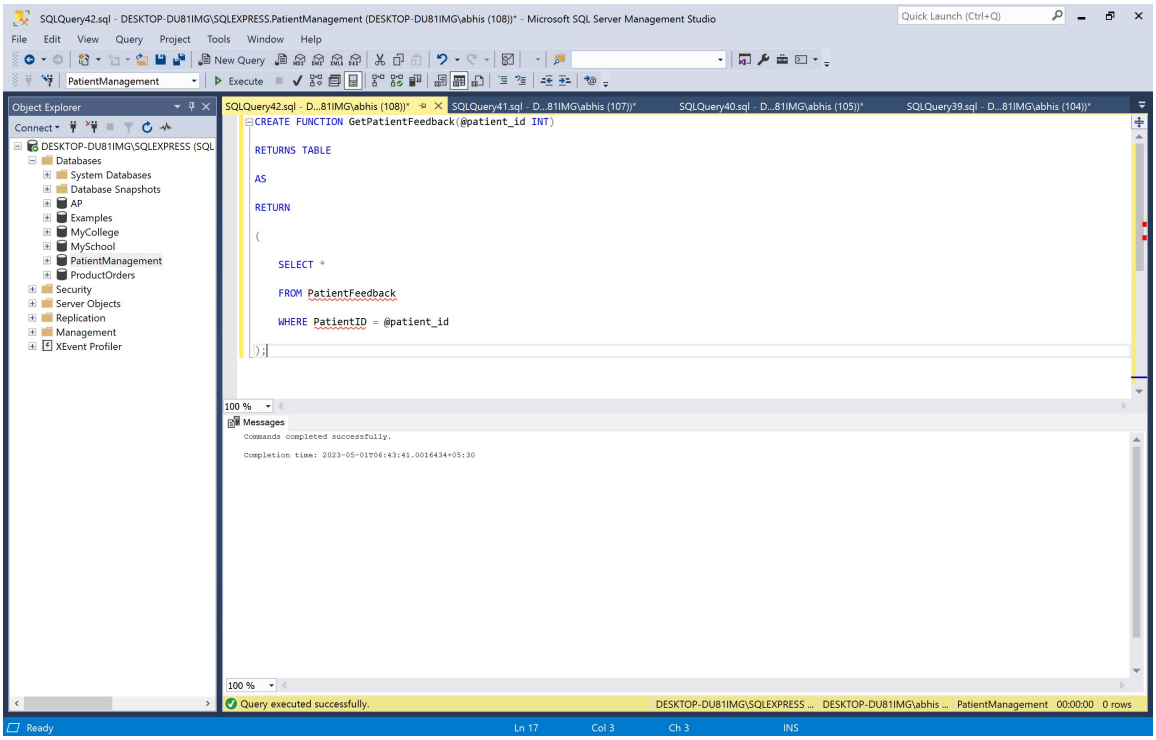
The above screenshot depicts the output for the above stored procedure query. This stored procedure UpdatePatientAddress takes two parameters: @PatientID (the ID of the patient whose address needs to be updated) and @NewAddress (the new address value). It updates the PatientAddress column in the Patients table for the specified patient with the new address value provided.

User Defined Functions

1. GetPatientFeedbac

kQuery:

```
CREATE FUNCTION GetPatientFeedback(@patient_id
INT)RETURNS TABLE
AS
RETUR
N(
    SELECT *
    FROM PatientFeedback
    WHERE PatientID = @patient_id
);
```



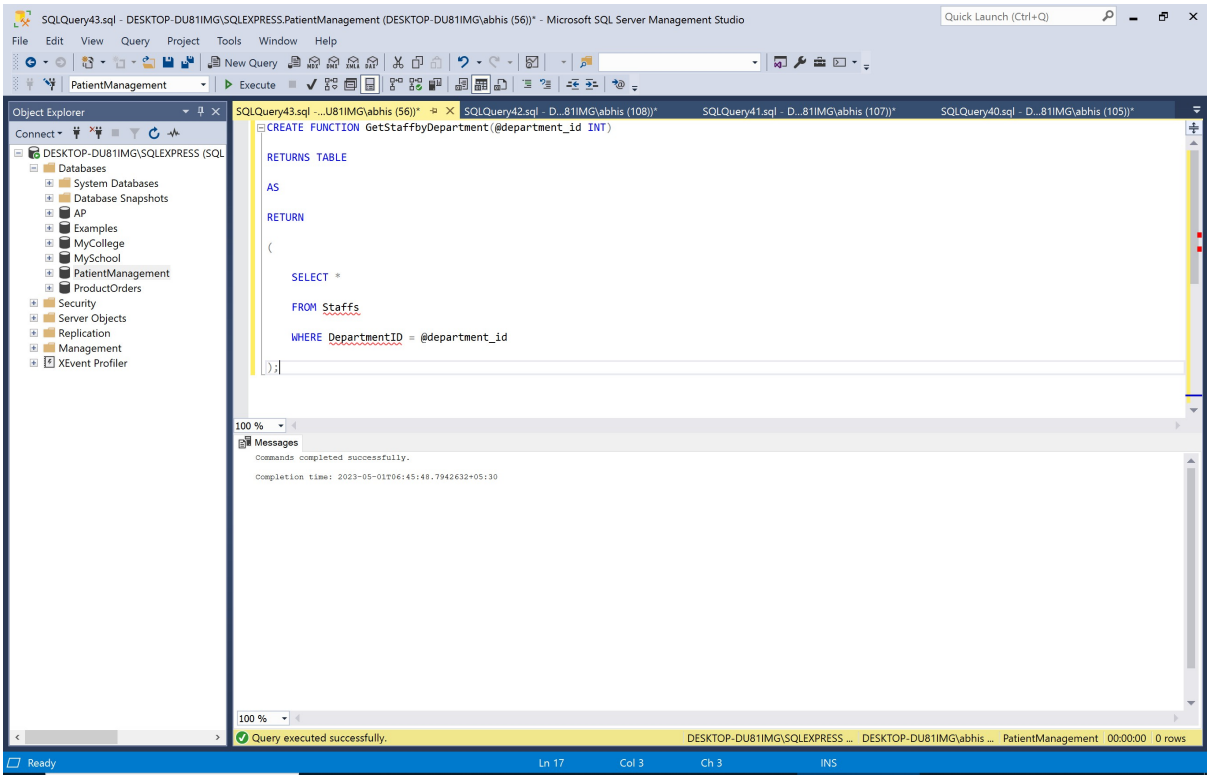
Comments:

The query is for creating the get_patient_feedback function. It is a SQL Server function that takes a patient ID as input and returns a table with all the feedback records associated with that patient ID from the PatientFeedback table**.** This query will return all the feedback recordsassociated with the specified patient ID from the PatientFeedback table, during testing process.

2. GetStaffbyDepartment

Query:

```
CREATE FUNCTION GetStaffbyDepartment(@department_id
INT)RETURNS TABLE
AS
RETUR
N(
    SELECT *
    FROM Staffs
    WHERE DepartmentID = @department_id
);
```



Comments:

The code is creating a user-defined function that retrieves staff members based on the provided @department_id.

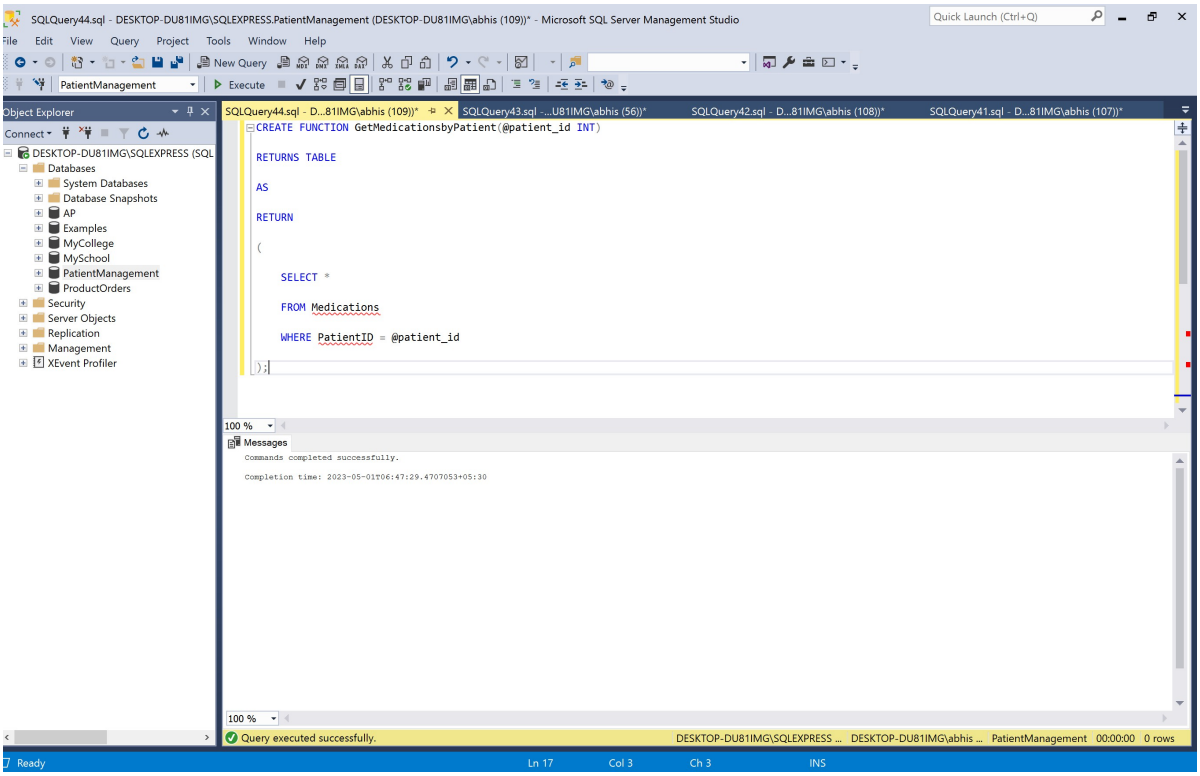
The function returns a table with all the columns from the Staffs table.

The WHERE clause filters the result to include only the rows where the DepartmentID matches the input @department_id. The function can be used in a SQL query to retrieve staff members based on the department ID.

3. GetMedicationsbyPatient

Query:

```
CREATE FUNCTION GetMedicationsbyPatient(@patient_id
INT)RETURNS TABLE
AS
RETUR
N(
    SELECT *
    FROM Medications
    WHERE PatientID = @patient_id
);
```



Comments:

The code is creating a user-defined function named GetMedicationsbyPatient.

The function takes an input parameter @patient_id of type INT.

The function returns a table that represents the result of the

query.The query selects all columns from the Medications table.

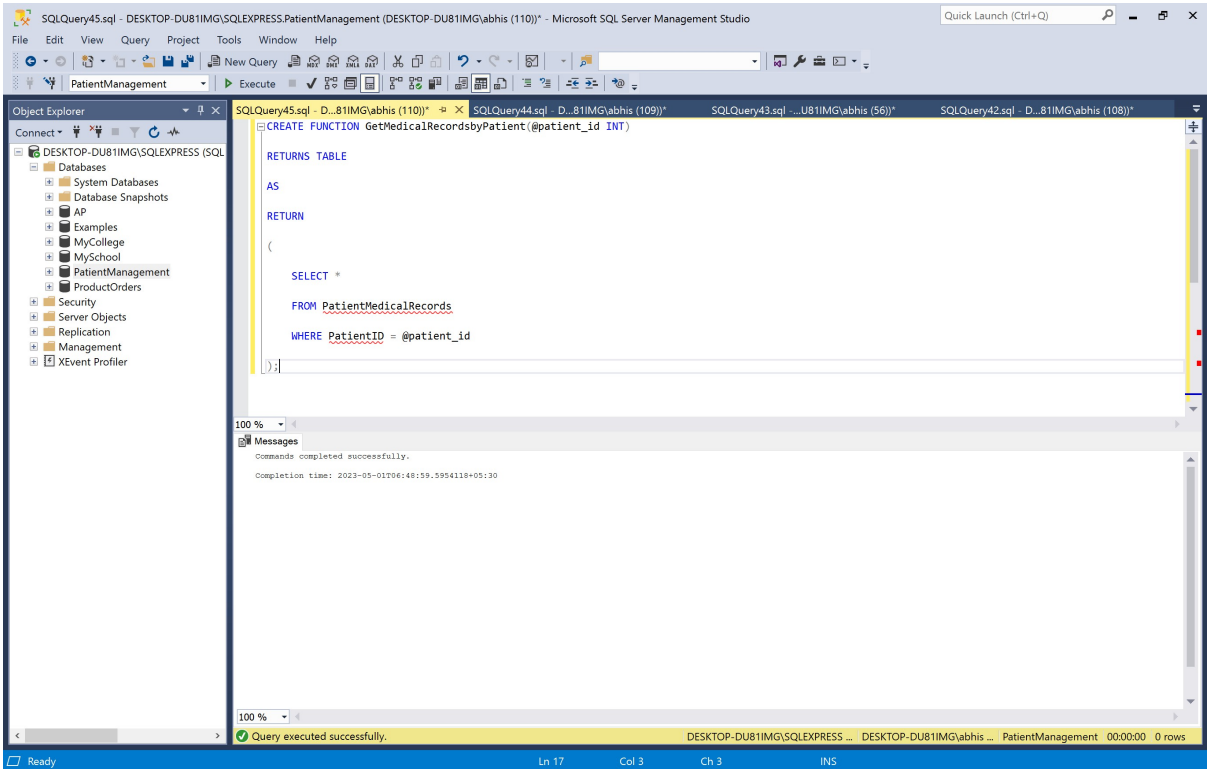
The WHERE clause filters the result to include only the rows where the PatientID matches the input @patient_id.

This function can be used in a SQL query to retrieve medications for a specific patient.

4. GetMedicalRecordsbyPatien

tQuery:

```
CREATE FUNCTION GetMedicalRecordsbyPatient(@patient_id
INT)RETURNS TABLE
AS
RETUR
N(
    SELECT *
    FROM PatientMedicalRecords
    WHERE PatientID = @patient_id
);
```



Comments:

The code is creating a user-defined function named GetMedicalRecordsbyPatient.

The function takes an input parameter @patient_id of type INT.

The function returns a table that represents the result of the query.

The query selects all columns from the PatientMedicalRecords table.

The WHERE clause filters the result to include only the rows where the PatientID matches the input @patient_id.

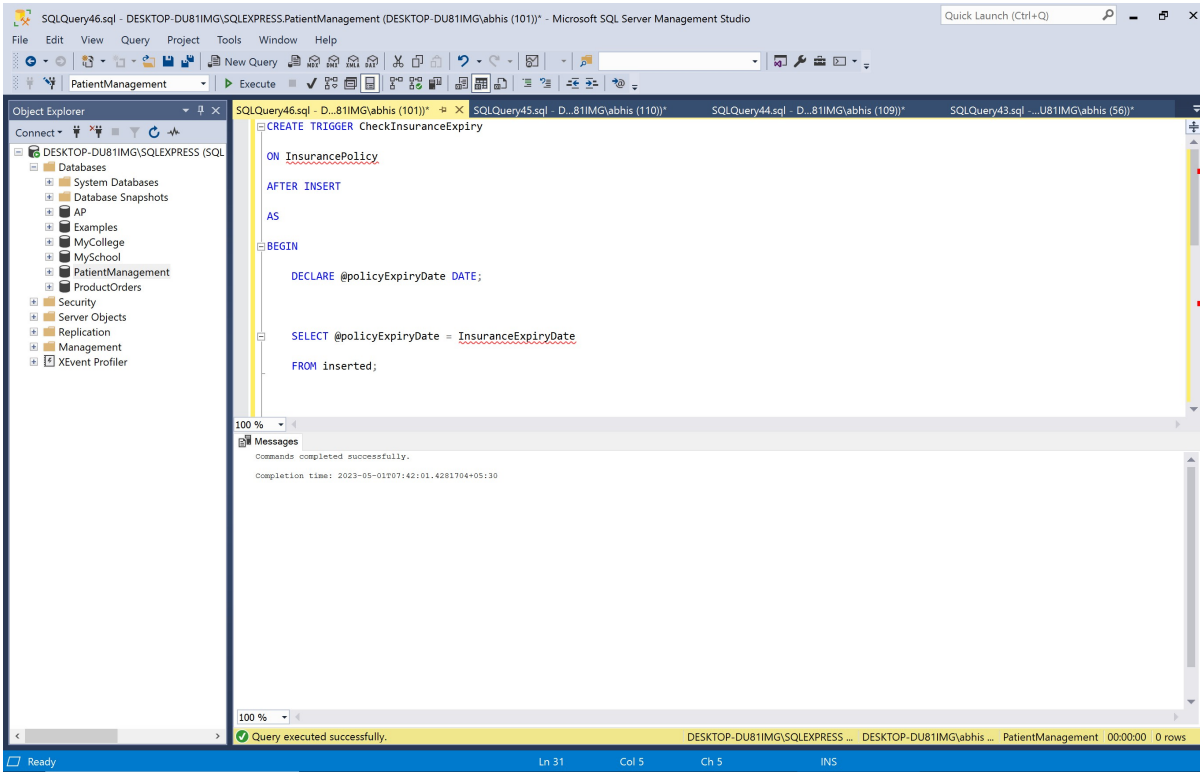
This function can be used in a SQL query to retrieve medical records for a specific patient**.**

Triggers

1. CheckInsuranceExpir

yQuery:

```
CREATE TRIGGER
CheckInsuranceExpiryON
InsurancePolicy
AFTER
INSERTAS
BEGIN
    DECLARE @policyExpiryDate DATE;

    SELECT @policyExpiryDate =
    InsuranceExpiryDateFROM inserted;

    IF @policyExpiryDate <
    GETDATE()BEGIN
        RAISERROR ('Insurance policy has expired.', 16,
        1);ROLLBACK TRANSACTION;
    END;
END;
```



Comments:

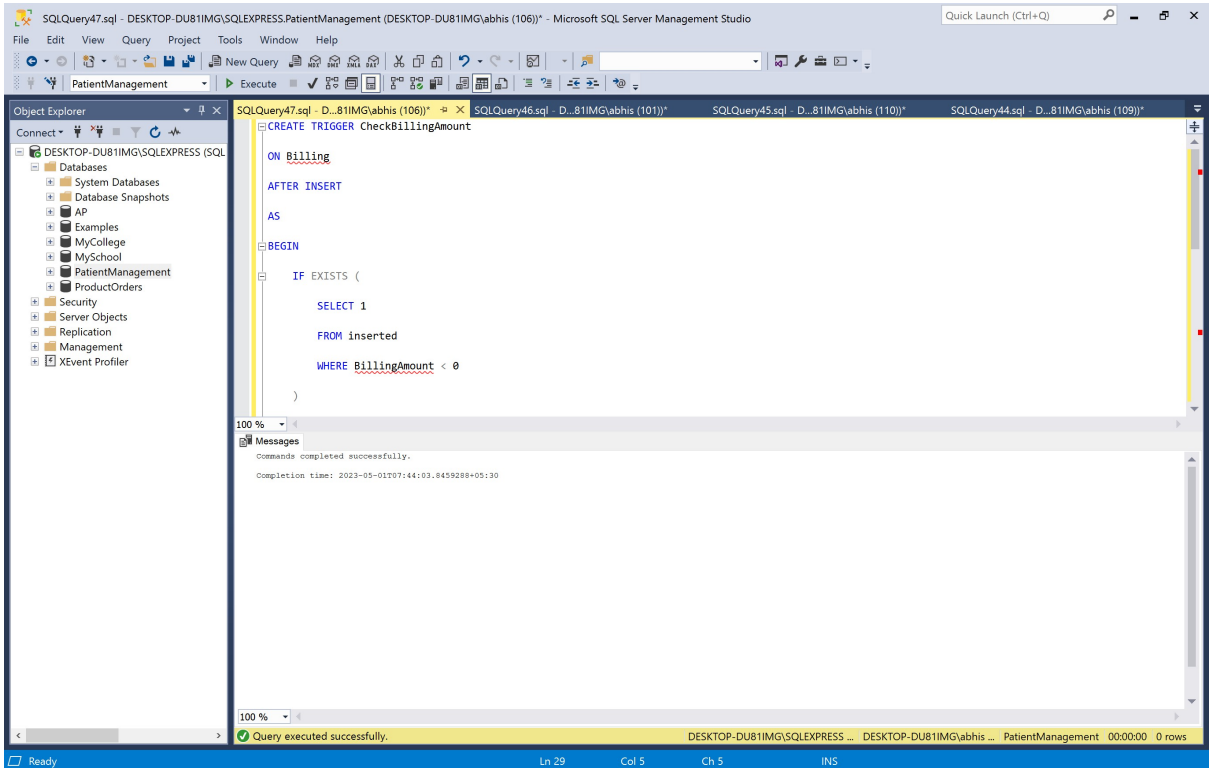
- The code is creating a trigger named **CheckInsuranceExpiry** on the **InsurancePolicy** table.
- The trigger is set to execute after an **INSERT** operation on the **InsurancePolicy** table.
- Inside the trigger, a variable **@policyExpiryDate** of type **DATE** is declared.
- The **SELECT** statement retrieves the insurance expiry date from the **inserted** table,which contains the rows affected by the **INSERT** operation.
- The **IF** statement checks if the **@policyExpiryDate** is less than the current date(**GETDATE()**).
- If the condition is true, the **RAISERROR** statement raises an error messageindicating that the insurance policy has expired.
- The **ROLLBACK TRANSACTION** statement is used to undo the **INSERT** operation and any other changes made within the same transaction.

This trigger ensures that only valid insurance policies are inserted into the **InsurancePolicy**table by checking the expiry date. If an expired policy is inserted, an error is raised, and thetransaction is rolled back to maintain data integrity.

2. CheckBillingAmoun

tQuery:

```
CREATE TRIGGER
CheckBillingAmountON Billing
AFTER
INSERTAS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted
        WHERE BillingAmount < 0
    )
    BEGIN
        RAISERROR ('Billing amount cannot be negative.', 16,
        1);ROLLBACK TRANSACTION;
    END;
END;
```



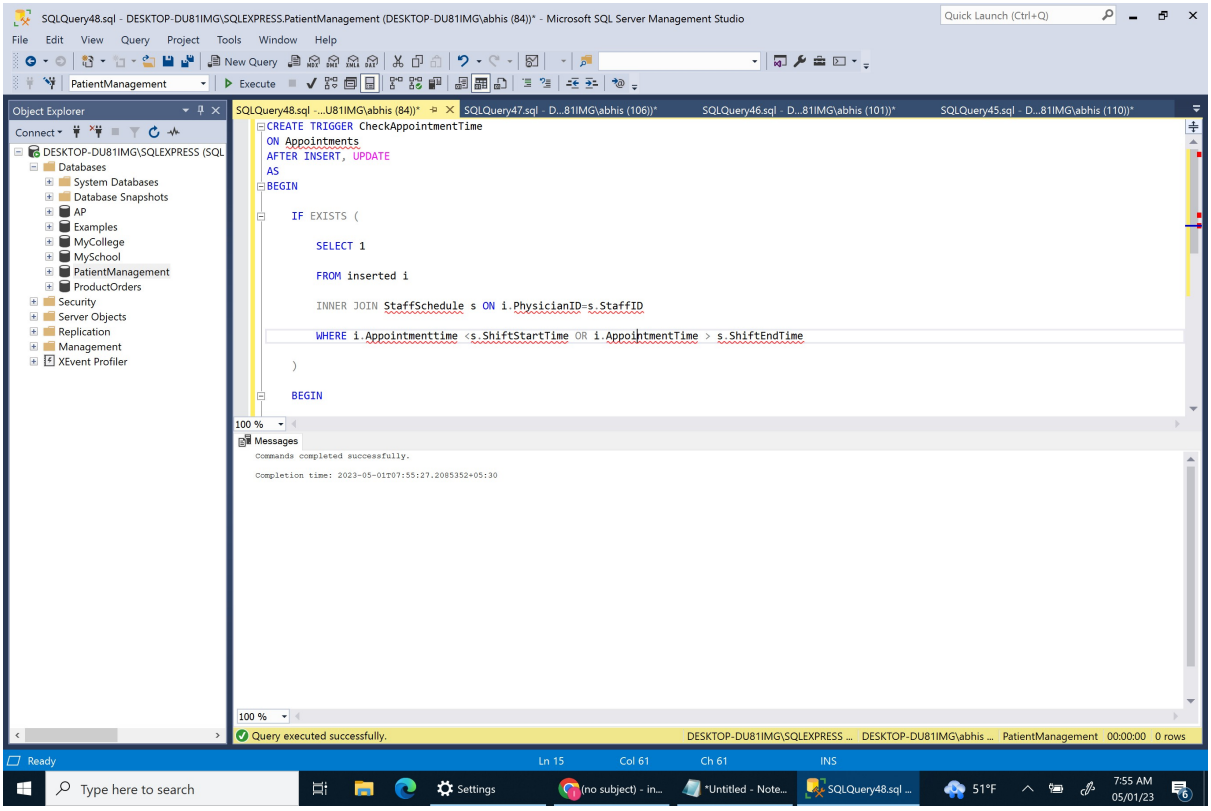
Comments:

- The code is creating a trigger named CheckBillingAmount on the Billing table.
- The trigger is set to execute after an INSERT operation on the Billing table.
- Inside the trigger, an IF statement is used to check if there are any rows in the insertedtable (representing the newly inserted rows) where the BillingAmount is less than 0 (negative).
- If such rows exist, the RAISERROR statement is executed to raise an error with the message "Billing amount cannot be negative." The error has a severity level of 16 anda state of 1.
- Additionally, the ROLLBACK TRANSACTION statement is used to undo theINSERT operation and any other changes made within the same transaction.

3. CheckAppointmentSchedul

eQuery:

```
CREATE TRIGGER
CheckAppointmentTimeON Appointments
AFTER INSERT,
UPDATEAS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted i
        INNER JOIN StaffSchedule s ON i.PhysicianID=s.StaffID
        WHERE I.Appointmenttime <s.ShiftStartTime OR i.AppointmentTime >
s.ShiftEndTime
    )
    BEGIN
        RAISERROR ('Appointmnent time is outside of working hours', 16,
        1);ROLLBACK TRANSACTION;
    END;
END;
```



Comments:

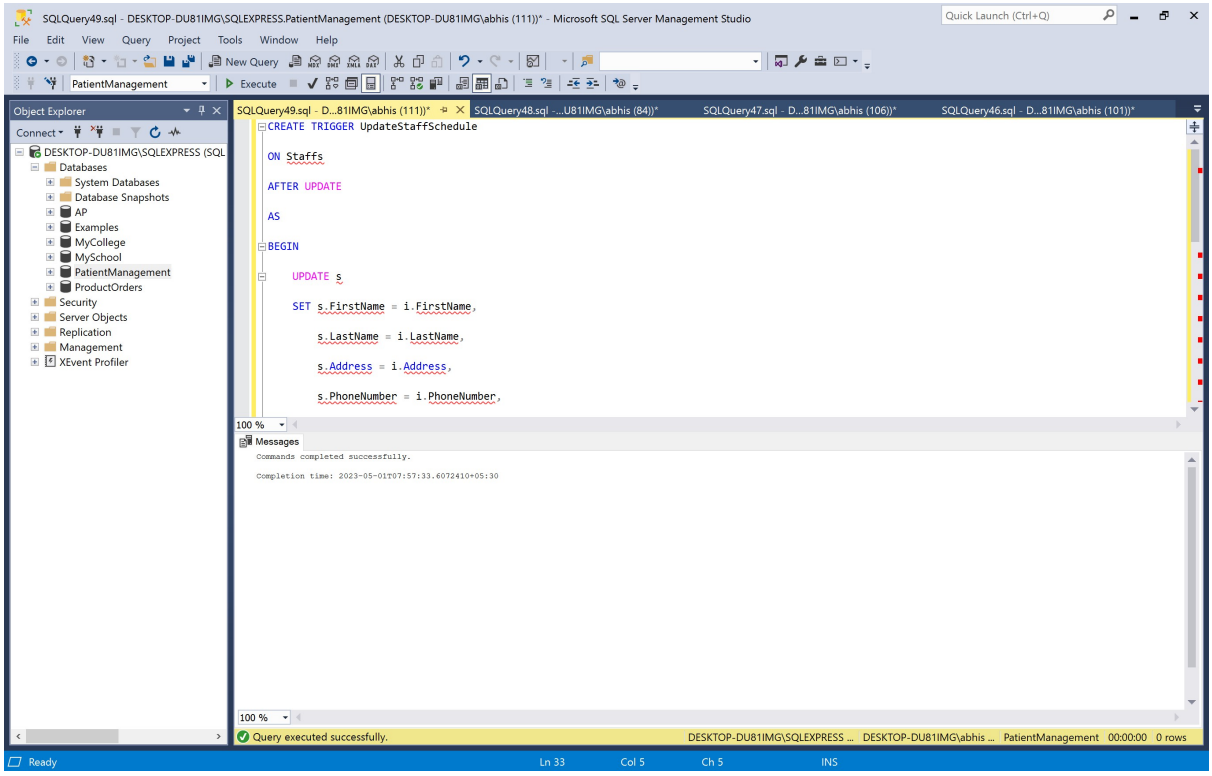
- The code is creating a trigger named CheckAppointmentTime on the Appointmentstable.
- The trigger is set to execute after an INSERT or UPDATE operation on the Appointments table.

- Inside the trigger, an IF statement is used to check if there are any rows in the inserted table (representing the newly inserted or updated rows) where the appointment time isoutside of the working hours for the corresponding physician.
- The check is performed by joining the inserted table with the StaffSchedule table onthe PhysicianID and StaffID columns, respectively. This allows retrieving the shift start time (ShiftStartTime) and shift end time (ShiftEndTime) for each physician.
- If any appointment time falls outside of the physician's working hours, the RAISERROR statement is executed to raise an error with the message "Appointment time is outside of working hours." The error has a severity level of 16 and a state of 1.
- Additionally, the ROLLBACK TRANSACTION statement is used to undo theINSERT or UPDATE operation and any other changes made within the same transaction.

4. UpdateStaffSchedul

eQuery:

```
CREATE TRIGGER
UpdateStaffScheduleON Staffs
AFTER
UPDATEAS
BEGIN
    UPDATE s
    SET s.FirstName = i.FirstName,
        s.LastName = i.LastName,
        s.Address = i.Address,
        s.PhoneNumber =
        i.PhoneNumber,s.Email =
        i.Email,
        s.JobTitle       =
        i.JobTitle,s.HireDate =
        i.HireDate, s.Salary  =
        i.Salary
    FROM Staffs s
    INNER JOIN inserted i ON s.StaffID = i.StaffID;
END;
```



Comments:

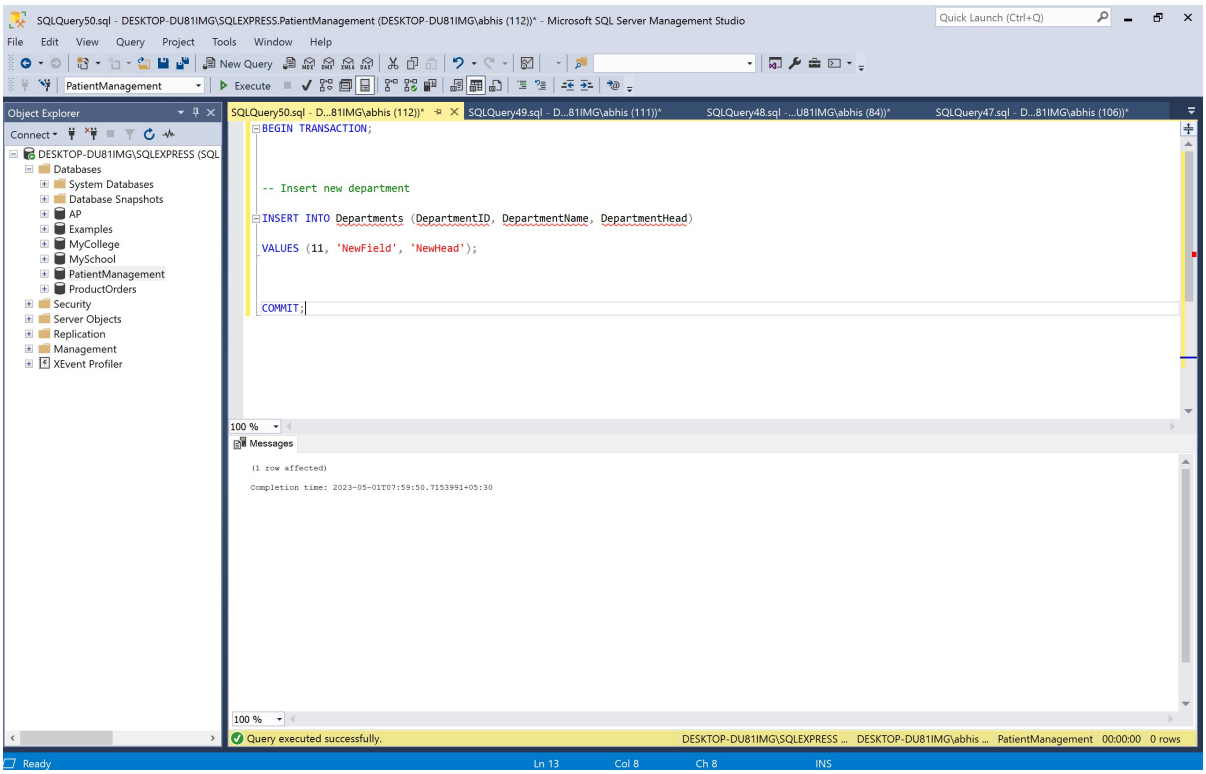
- The code is creating a trigger named UpdateStaffSchedule on the Staffs table.
- The trigger is set to execute after an UPDATE operation on the Staffs table.
- Inside the trigger, an UPDATE statement is used to modify the corresponding rows inthe Staffs table with the updated values from the inserted table.
- The UPDATE statement uses table aliases s for the Staffs table and i for the insertedtable.
- The INNER JOIN clause is used to join the Staffs table with the inserted table basedon the StaffID column.
- The SET clause specifies the column updates to be performed, assigning thecorresponding column values from the inserted table to the Staffs table.
- By performing the join and update, the trigger ensures that the corresponding rows inthe Staffs table are updated with the latest values after an update operation.

Transactions

1. **CreateDepartment**

Query:

```
BEGIN TRANSACTION;

-- Insert new department
INSERT INTO Departments (DepartmentID, DepartmentName,
DepartmentHead)VALUES (11, 'NewField', 'NewHead');

COMMIT;
```



Comments : The above screenshot gives the output for the transaction. A new department is inserted.

2. ScheduleAppointmen

tQuery:

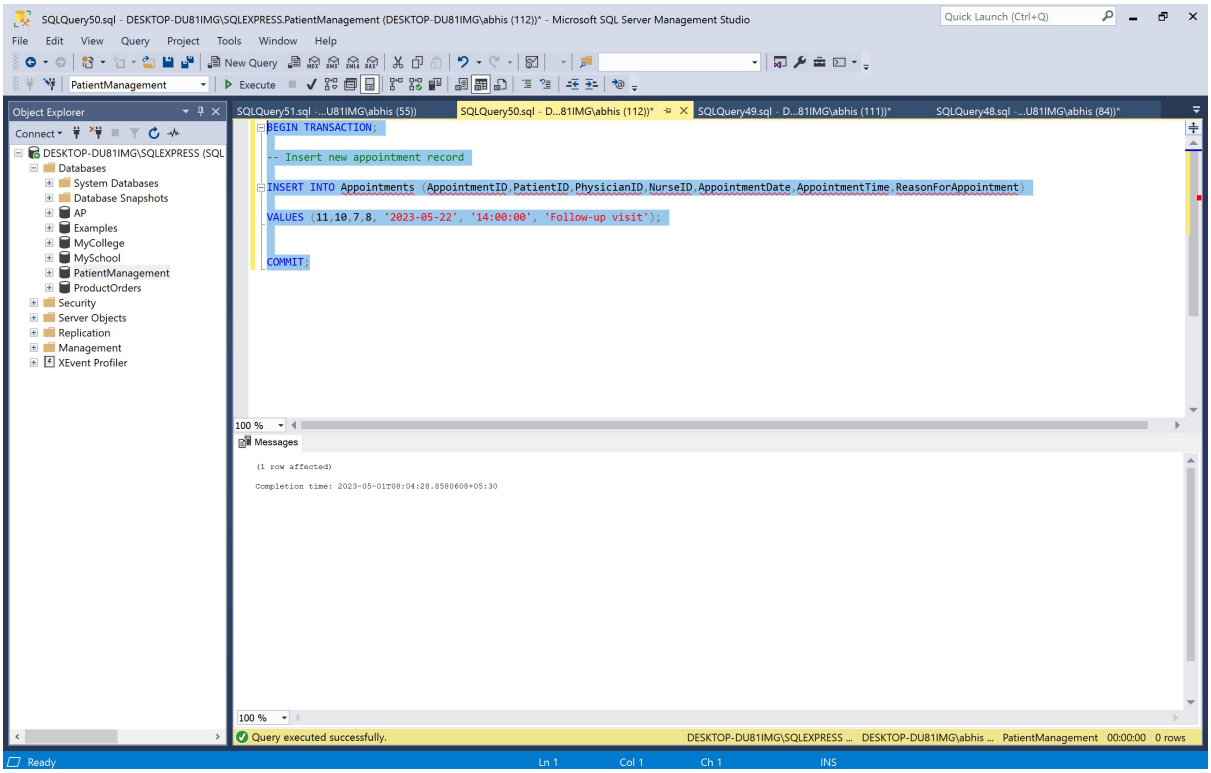
BEGIN TRANSACTION;
-- Insert new appointment

recordINSERT INTO

Appointments

(AppointmentID,PatientID,PhysicianID,NurseID,AppointmentDate,AppointmentTime,ReasonForAppointment)

```
VALUES (11,10,7,8, '2023-05-22', '14:00:00', 'Follow-up visit');

COMMIT;
```



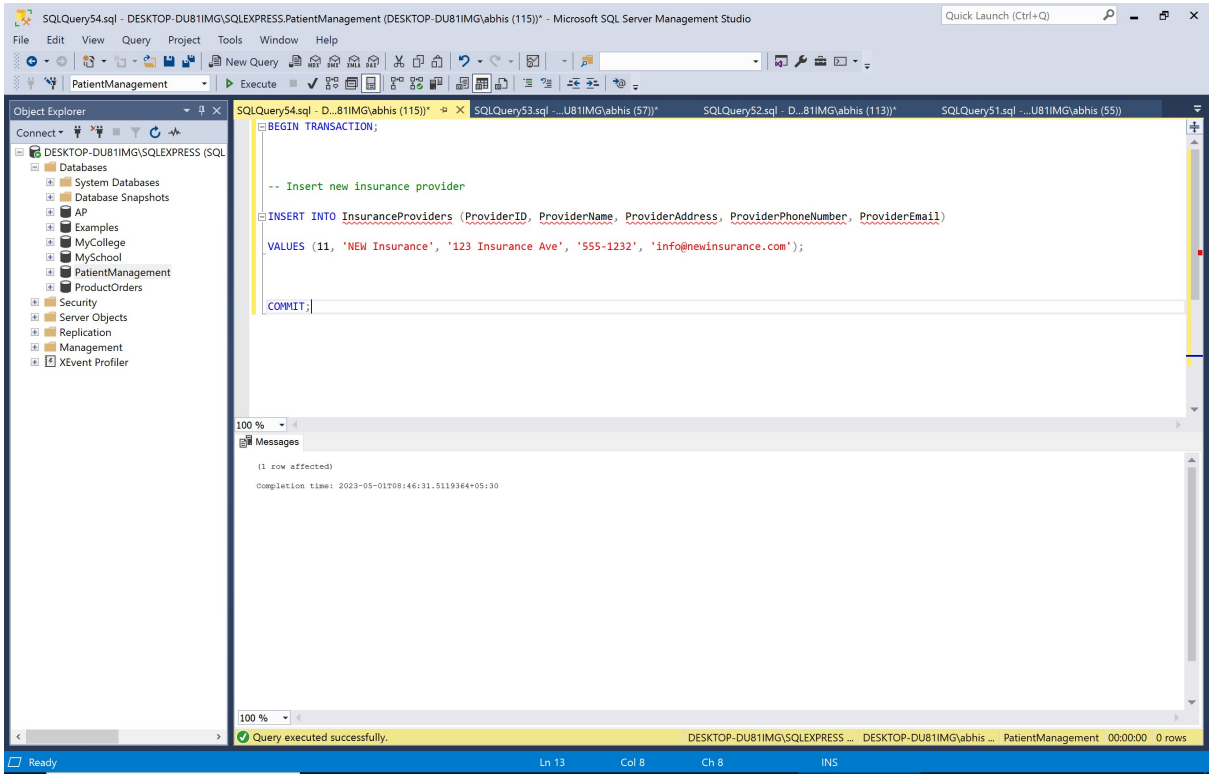
Comments:

The above screenshot gives the output for the transaction. A appointment is added to the schedule and is added to the appointments table.

3. AddInsuranceProvide

rQuery:

```
BEGIN TRANSACTION;

-- Insert new insurance provider
INSERT INTO InsuranceProviders (ProviderID, ProviderName,
ProviderAddress,ProviderPhoneNumber, ProviderEmail)
VALUES (11, 'NEW Insurance', '123 Insurance Ave',
'555-1232','info@newinsurance.com');

COMMIT;
```



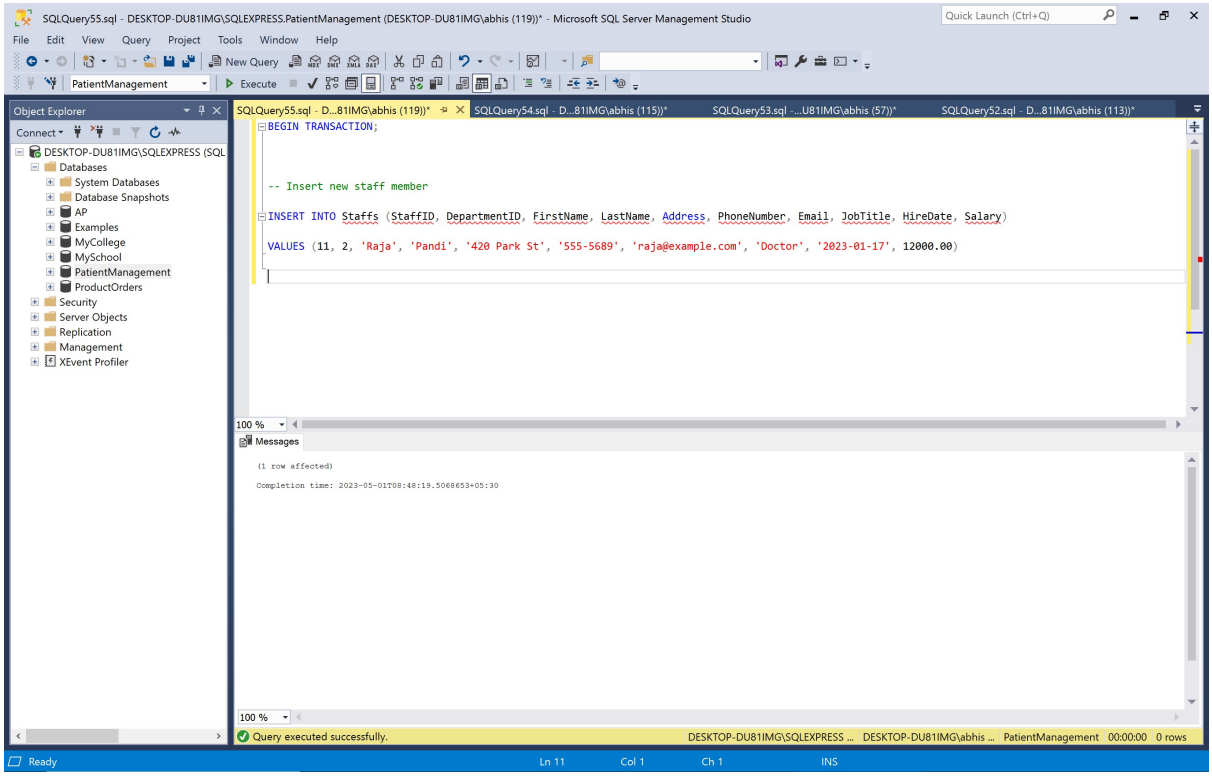
Comments:

The above screenshot depicts the output for the above transaction query. A new insurance provider is added to the InsuranceProvider table.

4. AddStaf

fQuery:

```
BEGIN TRANSACTION;

-- Insert new staff member
INSERT INTO Staffs (StaffID, DepartmentID, FirstName, LastName,
Address,PhoneNumber, Email, JobTitle, HireDate, Salary)
VALUES (11, 2, 'Raja', 'Pandi', '420 Park St', '555-5689', 'raja@example.com',
'Doctor', '2023-01-17', 12000.00)
```



Comments:

The above screenshot depicts the output for the above transaction query. A new staff is added to the Staffs table.

CONCLUSIONS/ REMARKS

- In this Project, I learned how to design, implement and test a database for aUniversity Medical Center.
- The project was mainly divided into 3 phases which included designing of thedatabase using E/R diagram and drawing conclusions from the same
- In the implementation phase, determination of tables, columns, primary keys,datatypes, nullabilities and relationships was achieved.
- In the testing phase, several business logics and performance and efficiencyimprovements were applied.