Medical Center

CSE581 – Project 2

Abstract - This project aims to design and implement a robust and efficient database management system (DBMS) for a medical institute using SQL. The database will cater to the various needs of the medical setting, encompassing patient management, physician and nurse schedules, medical procedures, medication prescriptions, lab results, billing, and more. The database design will involve creating several base tables such as Patients, Physicians, Nurses, Notes, Appointments, Admissions, Procedures, Medications, Lab Results, and Billing. These tables will capture essential information including patient details, medical history, insurance information, physician profiles, nurse information, notes made by healthcare professionals, appointment scheduling, patient admissions and discharges, medical procedures performed, prescribed medications, laboratory test results, and billing information. The completed DBMS project aims to provide a comprehensive platform for managing medical data efficiently and accurately.

By:
Tanmay Sameer Unhale
SUID: 899172356
tunhale@syr.edu

Contents:

Introduction:

This Healthcare Management System titled “Medical Center” is a comprehensive project that aims to streamline and optimize various processes within a healthcare facility, leveraging a well-designed and efficient database. This project encompasses a range of functionalities, including patient management, physician coordination, nurse assignments, appointment scheduling, medical procedures, medication tracking, and billing management. By utilizing this integrated system, healthcare providers can enhance the overall patient experience, improve operational efficiency, and ensure accurate and timely healthcare delivery. The database forms the backbone of the Healthcare Management System, containing structured and organized data pertaining to patients, physicians, nurses, rooms, procedures, medications, specialities, and more. It facilitates seamless data retrieval, storage, and manipulation, enabling healthcare professionals to access critical information quickly and make informed decisions. Patients play a central role in the system, with their details, such as names, dates of birth, genders, and contact information, being recorded. Patient addresses and insurance information are also captured, allowing for efficient communication and insurance coverage management. Appointments can be scheduled between patients and physicians, with details such as appointment dates, times, and reasons being recorded. Physicians and nurses are an integral part of the system, with their respective details, including names, contact information, and specialties, being stored. Physicians can be associated with specific procedures and patients, and their licenses are recorded. Nurses are assigned to various departments, and their contact information is maintained for effective communication. The system facilitates the management of medical procedures, including surgeries, blood work, stitching, chemotherapy, and imaging scans. It also tracks medication details, such as names, manufacturers, dosages, and frequencies, ensuring accurate administration and monitoring. Moreover, patient notes, lab results, and billing information are stored, providing a holistic view of the patient's medical journey. Billing management is another crucial aspect of the project, with insurance details, total bills, insurance coverage, and patient payments being recorded. This enables efficient billing processing and ensures accurate financial management.

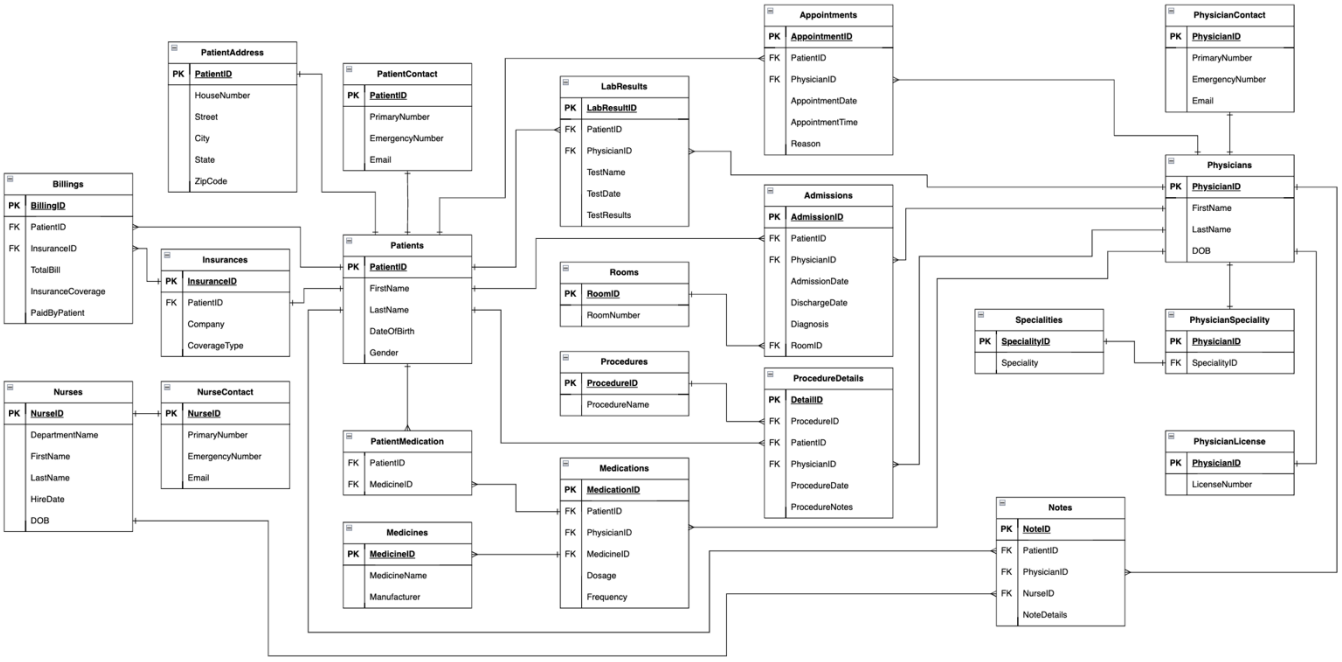
Project Phases:

This project consists of three phases:
1.Design Phase
2.Implementation Phase
3.Testing Phase

1. Design Phase

1.1 Database Design

The ER-Diagram for the database is this:



There are 22 tables in this following database with one-to-one and one-to-many relationship. There is also a many-to-many relationship which is resolved using a linking table.

1.2 Tables and Columns

1. Patients Table
    - PatientID (Primary Key)
    - FirstName
    - LastName
    - DateOfBirth
    - Gender

2. PatientContact Table:
    - PatientID (Primary Key)
    - PrimaryNumber
    - EmergencyNumber
    - Email

3. PatientAddress Table:

- PatientID (Primary Key)
- HouseNumber
- Street
- City
- State
- ZipCode

4. Insurances Table:
   - InsuranceID (Primary Key)
   - PatientID
   - Company
   - CoverageType

5. Billings Table:
   - BillingID (Primary Key)
   - PatientID
   - InsuranceID
   - TotalBill
   - InsuranceCoverage
   - PaidByPatient

6. Physicians Table:
   - PhysicianID (Primary Key)
   - FirstName
   - LastName
   - DOB
7. PhysicianContact Table:
   - PhysicianID (Primary Key)
   - PrimaryNumber
   - EmergencyNumber
   - Email

8. PhysicianLicense Table:
   - PhysicianID (Primary Key)
   - LicenseNumber

9. Specialities Table:
   - SpecialityID (Primary Key)
   - Speciality

10. PhysicianSpeciality Table:
    - PhysicianID (Primary Key)
    - SpecialityID

11. Nurses Table:
    - NurseID (Primary Key)

- DepartmentName
- FirstName
- LastName
- HireDate
- DOB

12. NurseContact Table:
    - NurseID (Primary Key)
    - PrimaryNumber
    - EmergencyNumber
    - Email

13. Appointments Table:
    - AppointmentID (Primary Key)
    - PatientID
    - PhysicianID
    - AppointmentDate
    - AppointmentTime
    - Reason

14. Rooms Table:
    - RoomID (Primary Key)
    - RoomNumber

15. Admissions Table:
    - AdmissionID (Primary Key)
    - PatientID
    - PhysicianID
    - RoomID
    - AdmissionDate
    - DischargeDate
    - Diagnosis

16. Procedures Table:
    - ProcedureID (Primary Key)
    - ProcedureName

17. ProcedureDetails Table:
    - DetailID (Primary Key)
    - ProcedureID
    - PatientID
    - PhysicianID
    - ProcedureDate
    - ProcedureNotes

18. Medicines Table:

- MedicineID (Primary Key)
- MedicineName
- Manufacturer

19. Medications Table:
    - MedicationID (Primary Key)
    - PatientID
    - PhysicianID
    - MedicineID
    - Dosage
    - Frequency

20. Notes Table:
    - NoteID (Primary Key)
    - PatientID
    - PhysicianID
    - NurseID
    - NoteDetails

21. LabResults Table:
    - LabResultID (Primary Key)
    - PatientID
    - PhysicianID
    - TestName
    - TestDate
    - TestResults

22. PatientMedication Table:
    - PatientID (Foreign Key)
    - MedicineID (Foreign Key)

1.3 Potential Integrity and Security Threats:

Possible integrity and security issues may be reduced by normalizing the tables. Referential integrity refers to the accurate upkeep of table relationships. This is employed to address integrity-related issues in database design. Every field in a base table designated as a foreign key must be capable of holding null values, values from a parent table's primary key, or values from a candidate key in order for referential integrity to be preserved in a database. This implies that every foreign key in a foreign key table must match a primary key in the connected database. Removing a record that contains a value that is referred to by a foreign key in another table would violate referential integrity if it were not preserved. Referential integrity is also included, changing the database design to minimize integrity and security problems while minimizing duplication. I've reached third normal form on my E/R diagram. I've divided the tables into several tables so that they can morph into the third normal form.

Here, the patient information is divided into Patients table, PatientAddress table and PatientContact table. The physician details are divided into Physicians table, PhysicianContact table, PhysicianLicense table and PysicianSpeciality table. The nurse details are stored in Nurses table and NurseContact table.

1.4 Relationships between table:

These following are **one-to-one** relationships:
Patients and PatientContact – As there is one contact detail for same patient.
Patients and PatientAddress – As there is one address detail for same patient.
Patients and Insurances – As there is one insurance detail for same patient.
Physicians and PhysicianContact – As there is one contact detail for same physician.
Physicians and PhysicianSpeciality – As there is one speciality for same physician (assumption).
Physicians and PhysicianLicense – As there is one license details for the same physician.
PhysicianSpeciality and Specialities – As there is one speciality for every physician.
Nurses and NurseContact – As there is one contact detail for same nurse.

These following are **one-to-many** relationships:
Patients and Billings – As one patient can have multiple bills.
Insurances and Billings – As one insurance can be used in multiple bills.
Patients and Appointments – As one patient can have multiple appointments.
Patients and LabResults – As one patient can have multiple lab tests and lab results.
Patients and Admissions – As one patient can be admitted multiple times (assumption).
Patients and ProcedureDetails – As one patient can have multiple procedure.
Patients and Notes – As one patient can have multiple notes.
Rooms and Admission – As one room can be assigned to multiple admission.
Procedures and ProcedureDetails – As same one procedure can be used multiple time.
Physician and Appointments – As one physician can have multiple appointments.
Physician and ProcedureDetails – As one physician might have to perform multiple procedures.
Physician and Notes – As one physician can have multiple notes.
Physician and Medications – As one doctor can give multiple medicines to multiple patients.
Medications and Medicines – As one medicine can be prescribed multiple.

Many-to-many relationship:
Patients and Medications tables has a many to many relationship and PatientMedication table is used as a linking table here.

2. Implementation Phase

2.1 Creating all tables

Code to create all tables:

```
CREATE TABLE Patients
(
   PatientID INT NOT NULL IDENTITY(1,1),
   FirstName VARCHAR(20) NOT NULL,
   LastName VARCHAR(20) NOT NULL,
   DateOfBirth DATE NOT NULL,
   Gender VARCHAR(1) NOT NULL,
   PRIMARY KEY (PatientID)
);

CREATE TABLE PatientContact
(
   PatientID INT NOT NULL IDENTITY(1,1),
   PrimaryNumber VARCHAR(15) NOT NULL,
   EmergencyNumber VARCHAR(15) NOT NULL,
   Email VARCHAR(30) NOT NULL,
   PRIMARY KEY (PatientID)
);

CREATE TABLE PatientAddress
(
   PatientID INT NOT NULL IDENTITY(1,1),
   HouseNumber VARCHAR(4) NOT NULL,
   Street VARCHAR(20) NOT NULL,
   City VARCHAR(20) NOT NULL,
   State VARCHAR(20) NOT NULL,
   ZipCode  VARCHAR(6) NOT NULL,
   PRIMARY KEY (PatientID)
);

CREATE TABLE Insurances
```

```
(
   InsuranceID INT NOT NULL IDENTITY(1,1),
   PatientID INT NOT NULL,
   Company VARCHAR(30) NOT NULL,
   CoverageType VARCHAR(20) NOT NULL,
   PRIMARY KEY (InsuranceID),
   FOREIGN KEY (PatientID) REFERENCES Patients(PatientID)
);

CREATE TABLE Billings
(
   BillingID INT NOT NULL IDENTITY(1,1),
   PatientID INT NOT NULL,
   InsuranceID INT NOT NULL,
   TotalBill MONEY NOT NULL,
   InsuranceCoverage MONEY NOT NULL,
   PaidByPatient MONEY NOT NULL,
   PRIMARY KEY (BillingID),
   FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
   FOREIGN KEY (InsuranceID) REFERENCES Insurances(InsuranceID)
);

CREATE TABLE Physicians
(
   PhysicianID INT NOT NULL IDENTITY(1,1),
   FirstName VARCHAR(20) NOT NULL,
   LastName VARCHAR(20) NOT NULL,
   DOB DATE NOT NULL,
   PRIMARY KEY (PhysicianID)
);

CREATE TABLE PhysicianContact
(
   PhysicianID INT NOT NULL IDENTITY(1,1),
   PrimaryNumber VARCHAR(15) NOT NULL,
   EmergencyNumber VARCHAR(15) NOT NULL,
   Email VARCHAR(30) NOT NULL,
```

```
    PRIMARY KEY (PhysicianID)
);

CREATE TABLE PhysicianLicense
(
    PhysicianID INT NOT NULL IDENTITY(1,1),
    LicenseNumber VARCHAR(7) NOT NULL,
    PRIMARY KEY (PhysicianID)
);

CREATE TABLE Specialities
(
    SpecialityID INT NOT NULL IDENTITY(1,1),
    Speciality VARCHAR(20) NOT NULL,
    PRIMARY KEY (SpecialityID)
);

CREATE TABLE PhysicianSpeciality
(
    PhysicianID INT NOT NULL IDENTITY(1,1),
    SpecialityID INT NOT NULL,
    PRIMARY KEY (PhysicianID),
    FOREIGN KEY (SpecialityID) REFERENCES Specialities(SpecialityID)
);

CREATE TABLE Nurses
(
    NurseID INT NOT NULL IDENTITY(1,1),
    DepartmentName VARCHAR(20) NOT NULL,
    FirstName VARCHAR(20) NOT NULL,
    LastName VARCHAR(20) NOT NULL,
    HireDate DATE NOT NULL,
    DOB DATE NOT NULL,
    PRIMARY KEY (NurseID)
);

CREATE TABLE NurseContact
```

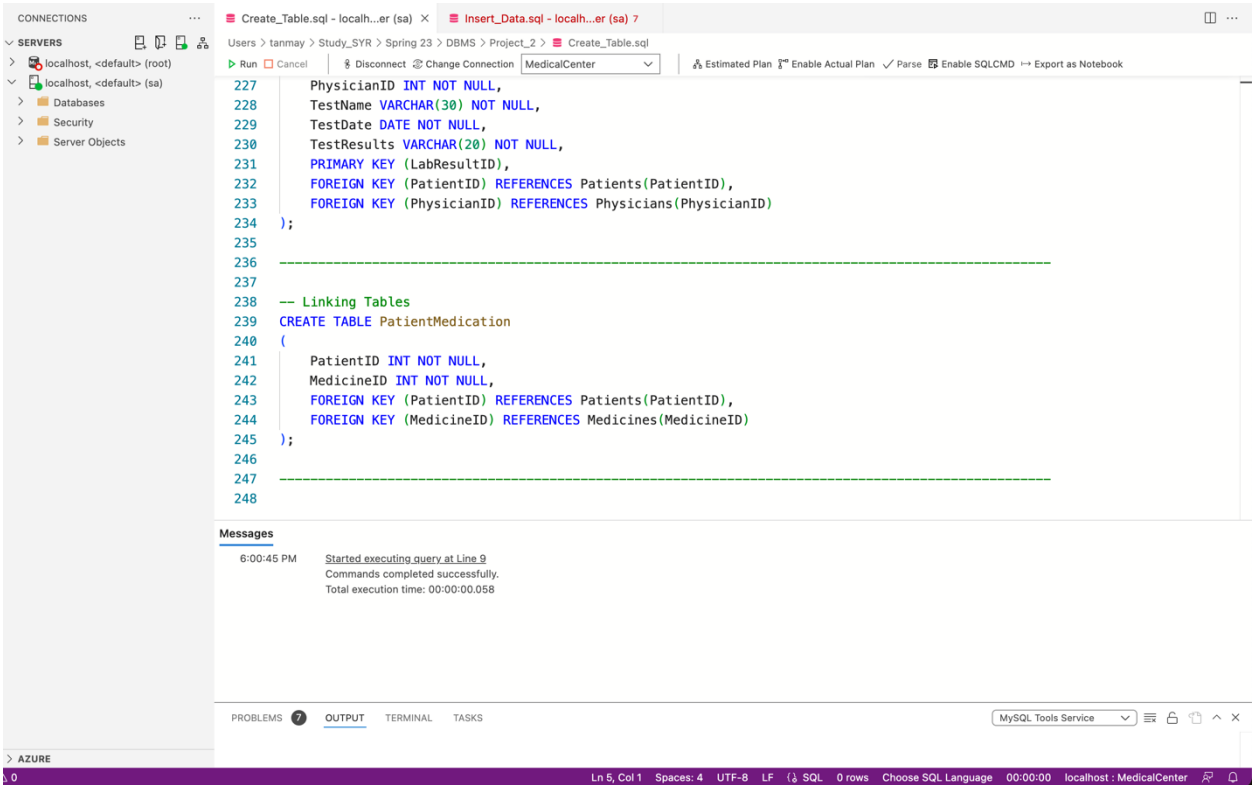
```
(
   NurseID INT NOT NULL IDENTITY(1,1),
   PrimaryNumber VARCHAR(15) NOT NULL,
   EmergencyNumber VARCHAR(15) NOT NULL,
   Email VARCHAR(30) NOT NULL,
   PRIMARY KEY (NurseID)
);

CREATE TABLE Appointments
(
   AppointmentID INT NOT NULL IDENTITY(1,1),
   PatientID INT NOT NULL,
   PhysicianID INT NOT NULL,
   AppointmentDate DATE NOT NULL,
   AppointmentTime TIME NOT NULL,
   Reason VARCHAR(50) NOT NULL,
   PRIMARY KEY (AppointmentID),
   FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
   FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID)
);

CREATE TABLE Rooms
(
   RoomID INT NOT NULL IDENTITY(1,1),
   RoomNumber INT NOT NULL,
   PRIMARY KEY (RoomID)
);

CREATE TABLE Admissions
(
   AdmissionID INT NOT NULL IDENTITY(1,1),
   PatientID INT NOT NULL,
   PhysicianID INT NOT NULL,
   RoomID INT NOT NULL,
   AdmissionDate DATE NOT NULL,
   DischargeDate DATE ,
   Diagnosis VARCHAR(50) NOT NULL,
```

```
    PRIMARY KEY (AdmissionID),
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID),
    FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID)
);

CREATE TABLE Procedures
(
    ProcedureID INT NOT NULL IDENTITY(1,1),
    ProcedureName VARCHAR(20) NOT NULL,
    PRIMARY KEY (ProcedureID)
);

CREATE TABLE ProcedureDetails
(
    DetailID INT NOT NULL IDENTITY(1,1),
    ProcedureID INT NOT NULL,
    PatientID INT NOT NULL,
    PhysicianID INT NOT NULL,
    ProcedureDate DATE NOT NULL,
    ProcedureNotes VARCHAR(50) NOT NULL,
    PRIMARY KEY (DetailID),
    FOREIGN KEY (ProcedureID) REFERENCES Procedures(ProcedureID),
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID),
);

CREATE TABLE Medicines
(
    MedicineID INT NOT NULL IDENTITY(1,1),
    MedicineName VARCHAR(30) NOT NULL,
    Manufacturer VARCHAR(30) NOT NULL,
    PRIMARY KEY (MedicineID)
);

CREATE TABLE Medications
(
```

```
    MedicationID INT NOT NULL IDENTITY(1,1),
    PatientID INT NOT NULL,
    PhysicianID INT NOT NULL,
    MedicineID INT NOT NULL,
    Dosage VARCHAR(15) NOT NULL,
    Frequency VARCHAR(15) NOT NULL,
    PRIMARY KEY (MedicationID),
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID),
    FOREIGN KEY (MedicineID) REFERENCES Medicines(MedicineID)
);

CREATE TABLE Notes
(
    NoteID INT NOT NULL IDENTITY(1,1),
    PatientID INT NOT NULL,
    PhysicianID INT NOT NULL,
    NurseID INT NOT NULL,
    NoteDetails VARCHAR(50) NOT NULL,
    PRIMARY KEY (NoteID),
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID),
    FOREIGN KEY (NurseID) REFERENCES Nurses(NurseID)
);

CREATE TABLE LabResults
(
    LabResultID INT NOT NULL IDENTITY(1,1),
    PatientID INT NOT NULL,
    PhysicianID INT NOT NULL,
    TestName VARCHAR(30) NOT NULL,
    TestDate DATE NOT NULL,
    TestResults VARCHAR(20) NOT NULL,
    PRIMARY KEY (LabResultID),
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (PhysicianID) REFERENCES Physicians(PhysicianID)
);
```

```
CREATE TABLE PatientMedication
(
   PatientID INT NOT NULL,
   MedicineID INT NOT NULL,
   FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
   FOREIGN KEY (MedicineID) REFERENCES Medicines(MedicineID)
);
```

Screenshot of execution:



Comment:
Now we have created all the tables in our database.

2.2 Inserting data into tables

Code to insert data:

```
INSERT INTO Rooms --
(RoomNumber)
VALUES
(101),(102),(103),(104),(105),
(201),(202),(203),(204),(205);

INSERT INTO Procedures --
(ProcedureName)
VALUES
('Surgery'),('Blood-Work'),('Stitching'),('Chemotherapy'),
('X-Ray'),('CT-Scan'),('MRI'),('C-Section');

INSERT INTO Medicines --
(MedicineName,Manufacturer)
VALUES
('Paracetamol',  'Johnson & Johnson'),
('Ibuprofen',    'Pfizer'),
('Aspirin',      'Johnson & Johnson'),
('Amoxicillin',  'Novartis'),
('Atorvastatin', 'Pfizer'),
('Metformin',    'AstraZeneca'),
('Omeprazole',   'AstraZeneca'),
('Simvastatin',  'Merck'),
('Levothyroxine','Pfizer'),
('Losartan',     'Merck');

INSERT INTO Specialities --
(Speciality)
VALUES
('Cardiology'),('Dermatology'),('Orthopedics'),
('Gastroenterology'),('Pediatrics'),('Neurology'),
('Ophthalmology'),('Oncology'),('Nephrology');

INSERT INTO Patients --
```

```
(FirstName, LastName, DateOfBirth, Gender)
VALUES
('Emma', 'Johnson', '1995-06-15', 'F'),
('Olivia', 'Smith', '1992-11-27', 'F'),
('Noah', 'Williams', '1990-09-08', 'M'),
('Liam', 'Brown', '1991-04-03', 'M'),
('Ava', 'Jones', '1976-02-21', 'F'),
('Sophia', 'Davis', '1974-08-12', 'F'),
('Mason', 'Miller', '1978-12-05', 'M'),
('Ethan', 'Taylor', '1973-07-30', 'M'),
('Isabella', 'Anderson', '1958-10-17', 'F'),
('Charlotte', 'Thomas', '1962-03-25', 'F');

INSERT INTO PatientContact --
(PrimaryNumber, EmergencyNumber, Email)
VALUES
('123-457-8900', '987-154-3210', 'emma.johnson@example.com'),
('231-567-1901', '826-543-2102', 'olivia.smith@example.com'),
('325-678-5012', '765-432-1598', 'noah.williams@example.com'),
('456-729-0143', '614-311-0987', 'liam.brown@example.com'),
('557-890-1034', '543-210-9806', 'ava.jones@example.com'),
('678-901-2355', '432-109-8565', 'sophia.davis@example.com'),
('779-012-3456', '371-098-7654', 'mason.miller@example.com'),
('890-163-4667', '260-987-6533', 'ethan.taylor@example.com'),
('901-234-1178', '229-876-5332', 'isabella.anderson@example.com'),
('212-345-6189', '898-765-4221', 'charlotte.thomas@example.com');

INSERT INTO PatientAddress --
(HouseNumber, Street, City, State, ZipCode)
VALUES
('123', 'Main Street', 'New York City', 'NY', '10001'),
('456', 'Maple Avenue', 'Buffalo', 'NY', '14201'),
('321', 'Elm Street', 'Rochester', 'NY', '14601'),
('987', 'Birch Avenue', 'Syracuse', 'NY', '13201'),
('543', 'Willow Lane', 'Albany', 'NY', '12201'),
('246', 'Oak Street', 'New York City', 'NY', '10002'),
('789', 'Cedar Avenue', 'Buffalo', 'NY', '14202'),
```

```
('654', 'Elm Court', 'Rochester', 'NY', '14602'),
('321', 'Birch Road', 'Syracuse', 'NY', '13202'),
('876', 'Willow Lane', 'Albany', 'NY', '12202');

INSERT INTO Nurses --
(DepartmentName, FirstName, LastName, HireDate, DOB)
VALUES
('Surgical-Assistant', 'Emily', 'Johnson', '2022-01-15', '1990-05-12'),
('Emergency', 'Michael', 'Smith', '2021-08-02', '1988-11-23'),
('Labor-Delivery', 'Sophia', 'Anderson', '2022-03-10', '1992-07-08'),
('Clinical-Nurse', 'James', 'Brown', '2021-05-29', '1991-02-17'),
('Labor-Delivery', 'Olivia', 'Taylor', '2022-06-18', '1993-09-06'),
('Emergency', 'Jacob', 'Martinez', '2023-02-08', '1994-04-29'),
('Surgical-Assistant', 'Isabella', 'Davis', '2022-11-14', '1990-12-03');

INSERT INTO NurseContact
(PrimaryNumber, EmergencyNumber, Email)
VALUES
('987-654-3770', '113-456-3890', 'emily.johnson@example.com'),
('836-543-2109', '234-597-8801', 'michael.smith@example.com'),
('735-432-1098', '305-678-9002', 'sophia.anderson@example.com'),
('654-351-0587', '456-739-0103', 'james.brown@example.com'),
('543-210-9676', '527-890-1214', 'olivia.taylor@example.com'),
('462-169-8765', '618-901-2245', 'jacob.martinez@example.com'),
('361-098-6654', '719-012-3156', 'isabella.davis@example.com');

INSERT INTO Insurances
(PatientID, Company, CoverageType)
VALUES
(1,'Blue Cross Blue Shield',   'Full'),
(2,'UnitedHealth Group',      'Full'),
(3,'Blue Cross Blue Shield',   'Full'),
(4,'Cigna',                'Partial'),
(5,'UnitedHealth Group',      'Partial'),
(6,'Aetna',                'Full'),
(7,'Aetna',                'Full'),
(8,'Aetna',                'Full'),
```

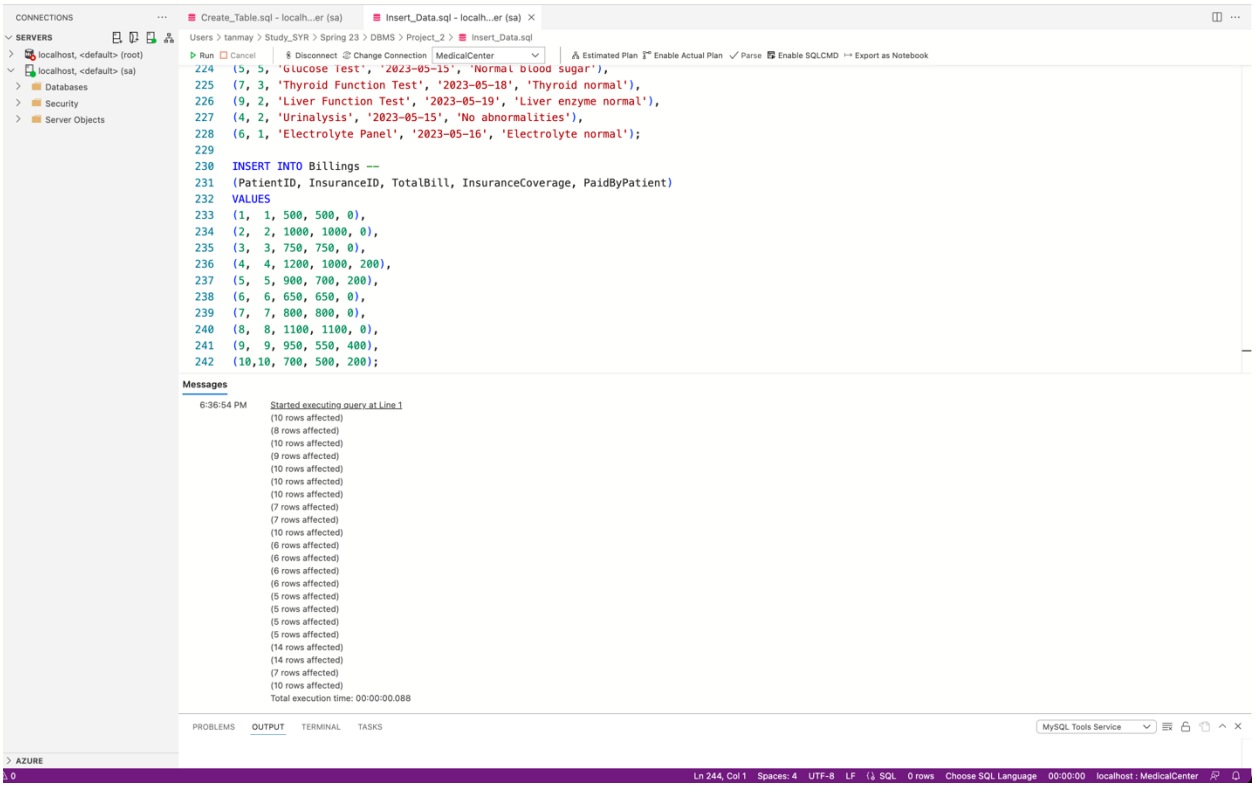
```
(9,'Cigna',                 'Partial'),
(10,'Cigna',                'Partial');

INSERT INTO Physicians --
(FirstName, LastName, DOB)
VALUES
('John', 'Smith', '1980-06-15'),
('Emily', 'Johnson', '1985-09-23'),
('Michael', 'Brown', '1976-11-30'),
('Sophia', 'Anderson', '1990-04-05'),
('David', 'Williams', '1982-02-18'),
('Olivia', 'Taylor', '1988-07-12');

INSERT INTO PhysicianContact --
(PrimaryNumber, EmergencyNumber, Email)
VALUES
('315-287-6543', '912-456-7890', 'john.smith@example.com'),
('233-187-0123', '345-654-3210', 'emily.johnson@example.com'),
('564-937-6541', '247-234-5678', 'michael.brown@example.com'),
('918-087-0123', '936-456-7890', 'sophia.anderson@example.com'),
('123-644-3210', '432-886-6541', 'david.williams@example.com'),
('907-664-5688', '180-487-6543', 'olivia.taylor@example.com');

INSERT INTO PhysicianLicense --
(LicenseNumber)
VALUES
(9876543), (4567890), (9870123),
(6543210), (9876541), (2345678);

INSERT INTO PhysicianSpeciality --
(SpecialityID)
VALUES
(2),(1),(6),(5),(3),(7);

INSERT INTO Appointments --
(PatientID, PhysicianID, AppointmentDate, AppointmentTime, Reason)
VALUES
```

```
(2,  5, '2023-05-15', '10:00:00', 'Routine check-up'),
(4,  2, '2023-05-16', '14:30:00', 'Follow-up for medication'),
(6,  1, '2023-05-17', '09:15:00', 'Initial consultation'),
(8,  5, '2023-05-19', '15:00:00', 'Annual physical'),
(10, 5, '2023-05-21', '12:00:00', 'Orthopedic evaluation');

INSERT INTO Admissions --
(PatientID, PhysicianID, RoomID, AdmissionDate, DischargeDate, Diagnosis)
VALUES
(1, 6, 1, '2023-05-10', '2023-05-15', 'Cataract'),
(3, 2, 10, '2023-05-12', NULL, 'Heart Blockages'),
(5, 5, 2, '2023-05-15', '2023-05-17', 'Fractured Arm'),
(7, 3, 4, '2023-05-18', NULL, 'Epilepsy'),
(9, 2, 6, '2023-05-20', '2023-05-23', 'Arrhythmias');

INSERT INTO ProcedureDetails --
(ProcedureID, PatientID, PhysicianID, ProcedureDate, ProcedureNotes)
VALUES
(1, 1, 6, '2023-05-12', 'Remove cataract using laser'),
(6, 3, 2, '2023-05-13', 'CT-Scan to check details'),
(5, 5, 5, '2023-05-16', 'X-Ray to see the fracture'),
(7, 7, 3, '2023-05-19', 'MRI scan of the brain'),
(7, 9, 2, '2023-05-22', 'MRI scan of the hearth');

INSERT INTO Notes --
(PatientID, PhysicianID, NurseID, NoteDetails)
VALUES
(1, 6, 1, 'Eye recovery is good'),
(3, 2, 2, 'Heart condition getting better'),
(5, 5, 2, 'Vital signs are normal'),
(7, 3, 4, 'Medications are not showing improvement'),
(9, 2, 6, 'Performed diagnostic tests. Results pending');

INSERT INTO Medications --
(PatientID, PhysicianID, MedicineID, Dosage, Frequency)
VALUES
(1, 6, 3, '3 Tabs'  , '3x - 3 Days'),
```

```
(1, 6, 5, '2 Tabs'  , '1x - 4 Days'),
(2, 5, 2, '20 ml'   , '2x - 2 Days'),
(3, 2, 8, '1 Tab'   , '1x - 6 Days'),
(3, 2, 6, '30 ml'   , '2x - 3 Days'),
(4, 2, 1, '4 Tabs'  , '2x - 3 Days'),
(5, 5, 2, '20 ml'   , '3x - 3 Days'),
(6, 1, 7, '10 ml'   , '1x - 5 Days'),
(7, 3, 4, '2 Tabs'  , '3x - 5 Days'),
(7, 3, 9, '5 ml'    , '2x - 7 Days'),
(7, 3, 10,'3 Tabs'  , '1x - 7 Days'),
(8, 5, 2, '15 ml'   , '3x - 3 Days'),
(9, 2, 1, '2 Tabs'  , '2x - 4 Days'),
(10,5, 5, '4 Tabs'  , '1x - 5 Days');

INSERT INTO PatientMedication --
(PatientID, MedicineID)
VALUES
(1,3), (1,5), (2,2), (3,8), (3,6), (4,1), (5,2),
(6,7), (7,4), (7,9), (7,10), (8,2), (9,1), (10,5);

INSERT INTO LabResults --
(PatientID, PhysicianID, TestName, TestDate, TestResults)
VALUES
(1, 6, 'Complete Blood Count', '2023-05-11', 'Normal'),
(3, 2, 'Lipid Profile', '2023-05-12', 'Cholesterol normal'),
(5, 5, 'Glucose Test', '2023-05-15', 'Normal blood sugar'),
(7, 3, 'Thyroid Function Test', '2023-05-18', 'Thyroid normal'),
(9, 2, 'Liver Function Test', '2023-05-19', 'Liver enzyme normal'),
(4, 2, 'Urinalysis', '2023-05-15', 'No abnormalities'),
(6, 1, 'Electrolyte Panel', '2023-05-16', 'Electrolyte normal');

INSERT INTO Billings --
(PatientID, InsuranceID, TotalBill, InsuranceCoverage, PaidByPatient)
VALUES
(1,  1, 500, 500, 0),
(2,  2, 1000, 1000, 0),
(3,  3, 750, 750, 0),
```

```
(4,  4, 1200, 1000, 200),
(5,  5, 900, 700, 200),
(6,  6, 650, 650, 0),
(7,  7, 800, 800, 0),
(8,  8, 1100, 1100, 0),
(9,  9, 950, 550, 400),
(10,10, 700, 500, 200);
```

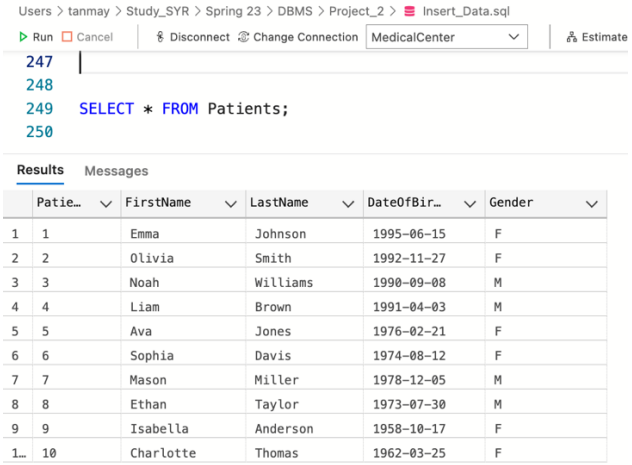
Screenshot of execution:



Comment:
Now the data is inserted in the database.

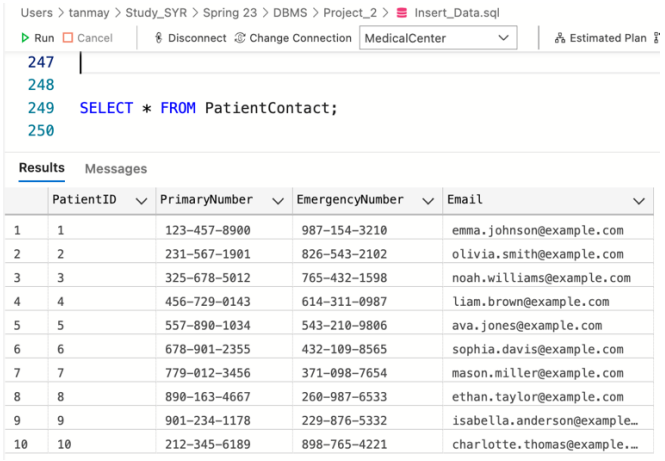
2.3 Observing tables

1. Patients Table



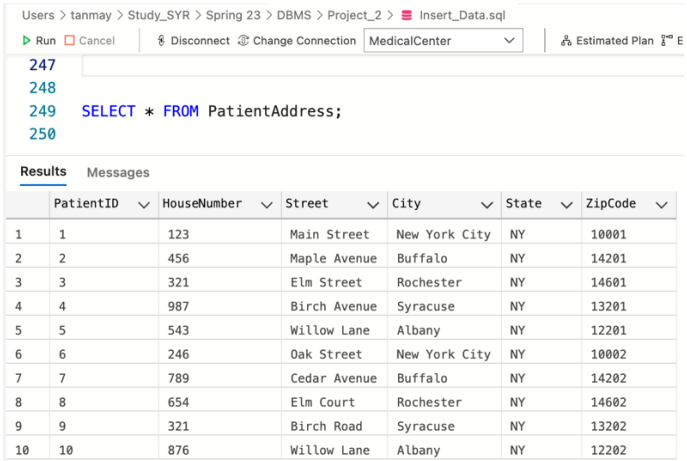
	Patie…	FirstName	LastName	DateOfBir…	Gender
1	1	Emma	Johnson	1995-06-15	F
2	2	Olivia	Smith	1992-11-27	F
3	3	Noah	Williams	1990-09-08	M
4	4	Liam	Brown	1991-04-03	M
5	5	Ava	Jones	1976-02-21	F
6	6	Sophia	Davis	1974-08-12	F
7	7	Mason	Miller	1978-12-05	M
8	8	Ethan	Taylor	1973-07-30	M
9	9	Isabella	Anderson	1958-10-17	F
1…	10	Charlotte	Thomas	1962-03-25	F

2. PatientContact Table:



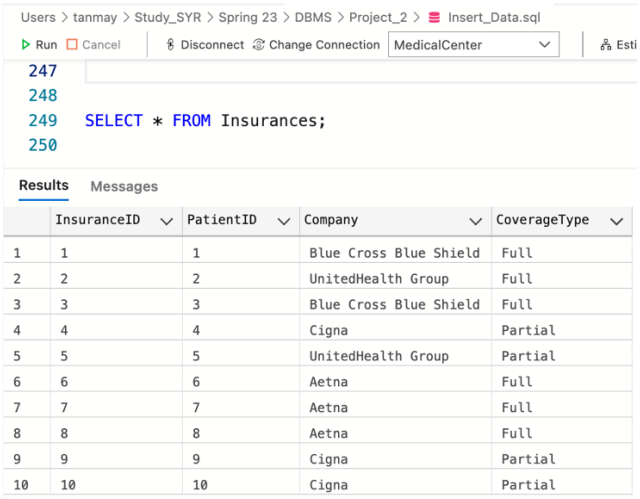
	PatientID	PrimaryNumber	EmergencyNumber	Email
1	1	123-457-8900	987-154-3210	emma.johnson@example.com
2	2	231-567-1901	826-543-2102	olivia.smith@example.com
3	3	325-678-5012	765-432-1598	noah.williams@example.com
4	4	456-729-0143	614-311-0987	liam.brown@example.com
5	5	557-890-1034	543-210-9806	ava.jones@example.com
6	6	678-901-2355	432-109-8565	sophia.davis@example.com
7	7	779-012-3456	371-098-7654	mason.miller@example.com
8	8	890-163-4667	260-987-6533	ethan.taylor@example.com
9	9	901-234-1178	229-876-5332	isabella.anderson@example…
10	10	212-345-6189	898-765-4221	charlotte.thomas@example.…

3. PatientAddress Table:



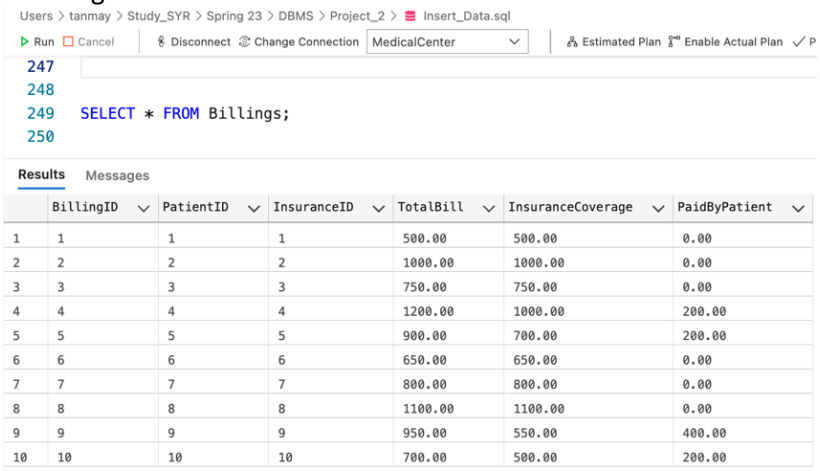
	PatientID	HouseNumber	Street	City	State	ZipCode
1	1	123	Main Street	New York City	NY	10001
2	2	456	Maple Avenue	Buffalo	NY	14201
3	3	321	Elm Street	Rochester	NY	14601
4	4	987	Birch Avenue	Syracuse	NY	13201
5	5	543	Willow Lane	Albany	NY	12201
6	6	246	Oak Street	New York City	NY	10002
7	7	789	Cedar Avenue	Buffalo	NY	14202
8	8	654	Elm Court	Rochester	NY	14602
9	9	321	Birch Road	Syracuse	NY	13202
10	10	876	Willow Lane	Albany	NY	12202

4. Insurances Table:



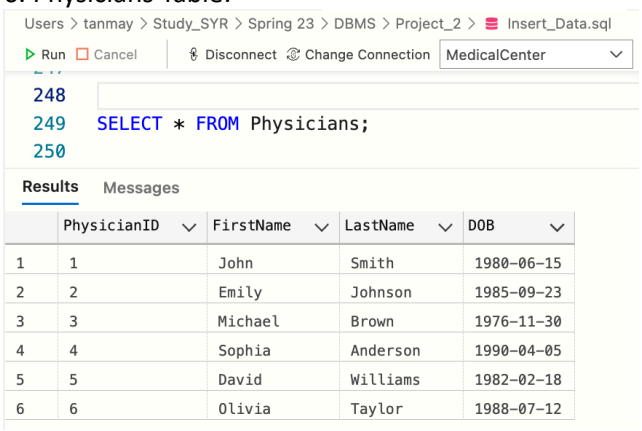
	InsuranceID	PatientID	Company	CoverageType
1	1	1	Blue Cross Blue Shield	Full
2	2	2	UnitedHealth Group	Full
3	3	3	Blue Cross Blue Shield	Full
4	4	4	Cigna	Partial
5	5	5	UnitedHealth Group	Partial
6	6	6	Aetna	Full
7	7	7	Aetna	Full
8	8	8	Aetna	Full
9	9	9	Cigna	Partial
10	10	10	Cigna	Partial

5. Billings Table:



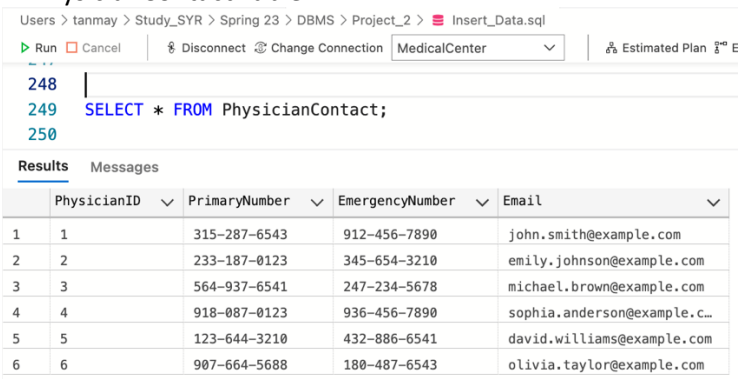
	BillingID	PatientID	InsuranceID	TotalBill	InsuranceCoverage	PaidByPatient
1	1	1	1	500.00	500.00	0.00
2	2	2	2	1000.00	1000.00	0.00
3	3	3	3	750.00	750.00	0.00
4	4	4	4	1200.00	1000.00	200.00
5	5	5	5	900.00	700.00	200.00
6	6	6	6	650.00	650.00	0.00
7	7	7	7	800.00	800.00	0.00
8	8	8	8	1100.00	1100.00	0.00
9	9	9	9	950.00	550.00	400.00
10	10	10	10	700.00	500.00	200.00

6. Physicians Table:



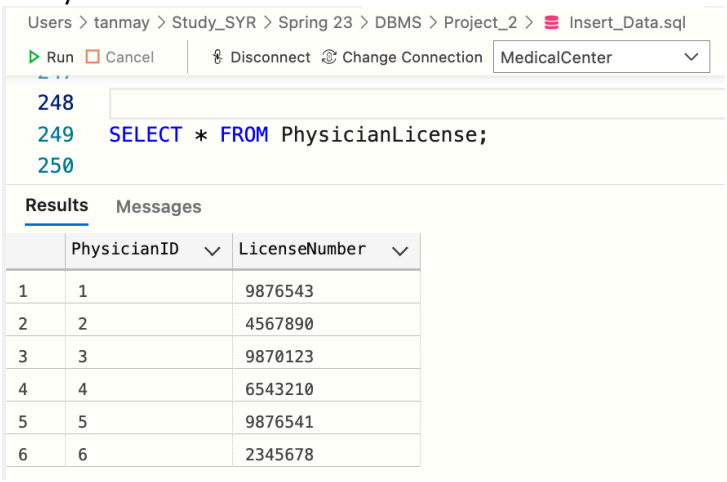
	PhysicianID	FirstName	LastName	DOB
1	1	John	Smith	1980-06-15
2	2	Emily	Johnson	1985-09-23
3	3	Michael	Brown	1976-11-30
4	4	Sophia	Anderson	1990-04-05
5	5	David	Williams	1982-02-18
6	6	Olivia	Taylor	1988-07-12

7. PhysicianContact Table:



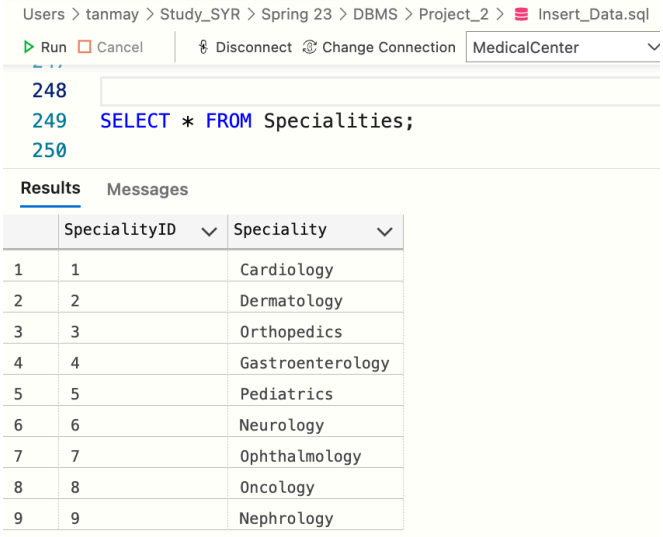
	PhysicianID	PrimaryNumber	EmergencyNumber	Email
1	1	315-287-6543	912-456-7890	john.smith@example.com
2	2	233-187-0123	345-654-3210	emily.johnson@example.com
3	3	564-937-6541	247-234-5678	michael.brown@example.com
4	4	918-087-0123	936-456-7890	sophia.anderson@example.c…
5	5	123-644-3210	432-886-6541	david.williams@example.com
6	6	907-664-5688	180-487-6543	olivia.taylor@example.com

8. PhysicianLicense Table:



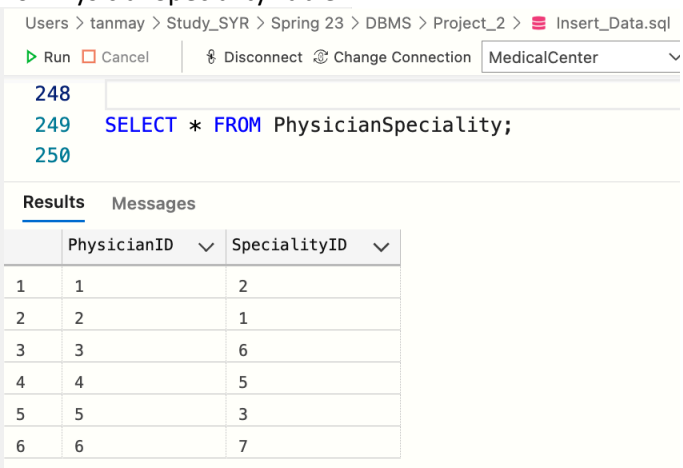
	PhysicianID	LicenseNumber
1	1	9876543
2	2	4567890
3	3	9870123
4	4	6543210
5	5	9876541
6	6	2345678

9. Specialities Table:



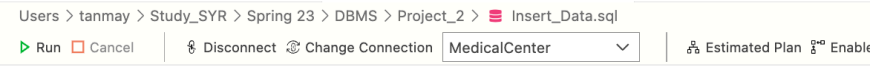
	SpecialityID	Speciality
1	1	Cardiology
2	2	Dermatology
3	3	Orthopedics
4	4	Gastroenterology
5	5	Pediatrics
6	6	Neurology
7	7	Ophthalmology
8	8	Oncology
9	9	Nephrology

10. PhysicianSpeciality Table:



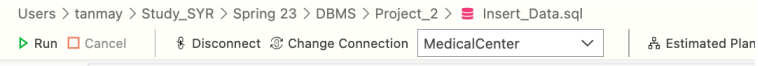
	PhysicianID	SpecialityID
1	1	2
2	2	1
3	3	6
4	4	5
5	5	3
6	6	7

11. Nurses Table:



	NurseID	DepartmentName	FirstName	LastName	HireDate	DOB
1	1	Surgical-Assistant	Emily	Johnson	2022-01-15	1990-05-12
2	2	Emergency	Michael	Smith	2021-08-02	1988-11-23
3	3	Labor-Delivery	Sophia	Anderson	2022-03-10	1992-07-08
4	4	Clinical-Nurse	James	Brown	2021-05-29	1991-02-17
5	5	Labor-Delivery	Olivia	Taylor	2022-06-18	1993-09-06
6	6	Emergency	Jacob	Martinez	2023-02-08	1994-04-29
7	7	Surgical-Assistant	Isabella	Davis	2022-11-14	1990-12-03

12. NurseContact Table:



	NurseID	PrimaryNumber	EmergencyNumber	Email
1	1	987-654-3770	113-456-3890	emily.johnson@example.com
2	2	836-543-2109	234-597-8801	michael.smith@example.com
3	3	735-432-1098	305-678-9002	sophia.anderson@example.c…
4	4	654-351-0587	456-739-0103	james.brown@example.com
5	5	543-210-9676	527-890-1214	olivia.taylor@example.com
6	6	462-169-8765	618-901-2245	jacob.martinez@example.com
7	7	361-098-6654	719-012-3156	isabella.davis@example.com

13. Appointments Table:

Users > tanmay > Study_SYR > Spring 23 > DBMS > Project_2 > Insert_Data.sql

Run | Cancel | Disconnect | Change Connection | MedicalCenter | Estimated Plan | Enable Actual Plan | Parse | Enable SQLCM

```
248
249  SELECT * FROM Appointments;
250
```

Results | Messages

	AppointmentID	PatientID	PhysicianID	AppointmentDate	AppointmentTime	Reason
1	1	2	5	2023-05-15	10:00:00	Routine check-up
2	2	4	2	2023-05-16	14:30:00	Follow-up for medication
3	3	6	1	2023-05-17	09:15:00	Initial consultation
4	4	8	5	2023-05-19	15:00:00	Annual physical
5	5	10	5	2023-05-21	12:00:00	Orthopedic evaluation

14. Rooms Table:

Users > tanmay > Study_SYR > Spring 23 > DBMS > Project_2 > Insert_Data.sql

Run | Cancel | Disconnect | Change Connection | MedicalCenter

```
248
249  SELECT * FROM Rooms;
250
```

Results | Messages

	RoomID	RoomNumber
1	1	101
2	2	102
3	3	103
4	4	104
5	5	105
6	6	201
7	7	202
8	8	203
9	9	204
10	10	205

15. Admissions Table:

Users > tanmay > Study_SYR > Spring 23 > DBMS > Project_2 > Insert_Data.sql

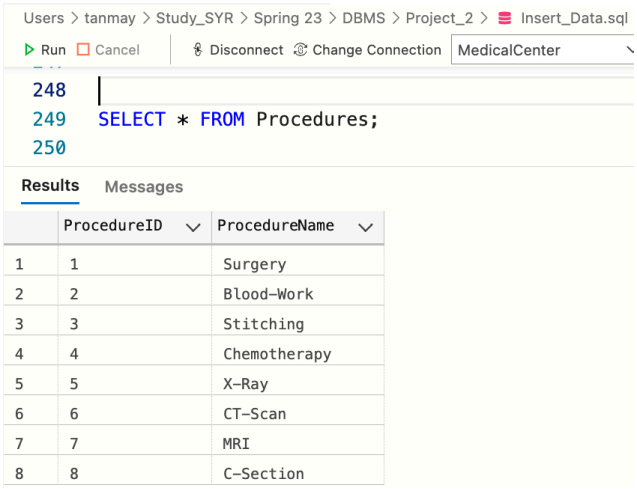
Run | Cancel | Disconnect | Change Connection | MedicalCenter | Estimated Plan | Enable Actual Plan | Parse | Enable SC

```
248
249  SELECT * FROM Admissions;
250
```

Results | Messages

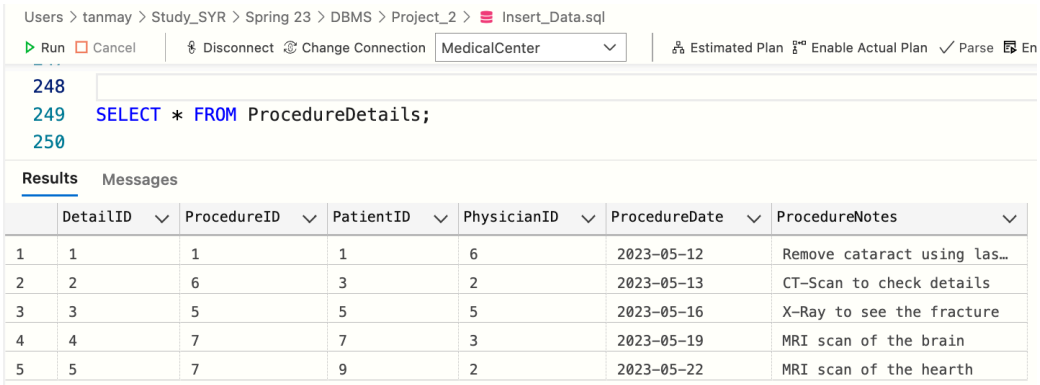
	AdmissionID	PatientID	PhysicianID	RoomID	AdmissionDate	DischargeDate	Diagnosis
1	1	1	6	1	2023-05-10	2023-05-15	Cataract
2	2	3	2	10	2023-05-12	*NULL*	Heart Blockages
3	3	5	5	2	2023-05-15	2023-05-17	Fractured Arm
4	4	7	3	4	2023-05-18	*NULL*	Epilepsy
5	5	9	2	6	2023-05-20	2023-05-23	Arrhythmias

16. Procedures Table:



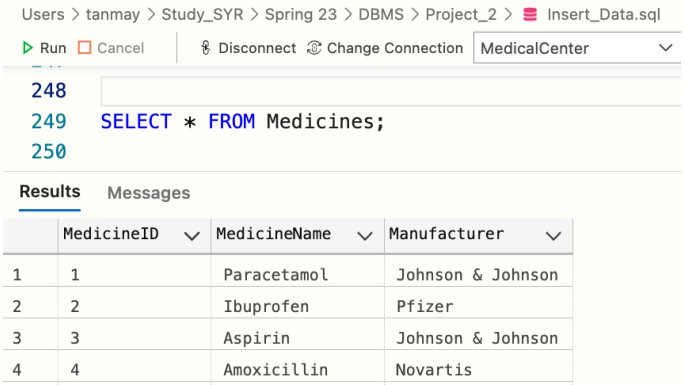
	ProcedureID	ProcedureName
1	1	Surgery
2	2	Blood-Work
3	3	Stitching
4	4	Chemotherapy
5	5	X-Ray
6	6	CT-Scan
7	7	MRI
8	8	C-Section

17. ProcedureDetails Table:



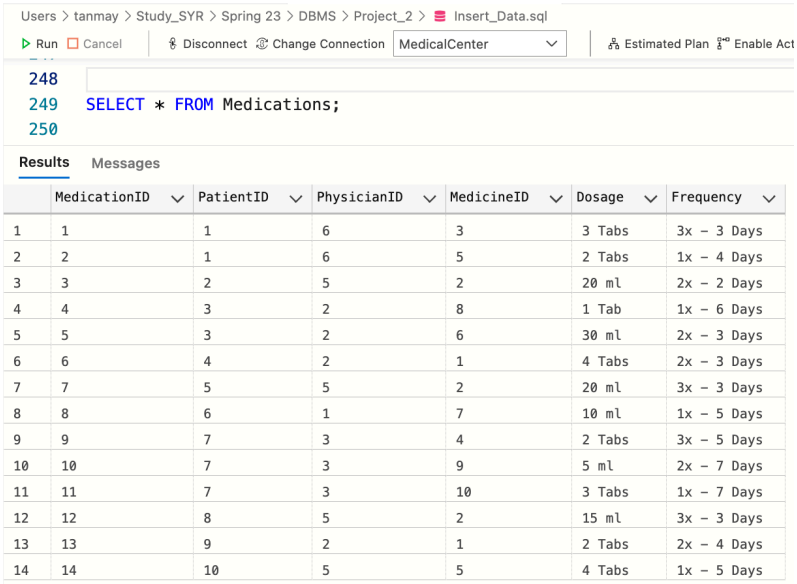
	DetailID	ProcedureID	PatientID	PhysicianID	ProcedureDate	ProcedureNotes
1	1	1	1	6	2023-05-12	Remove cataract using las…
2	2	6	3	2	2023-05-13	CT-Scan to check details
3	3	5	5	5	2023-05-16	X-Ray to see the fracture
4	4	7	7	3	2023-05-19	MRI scan of the brain
5	5	7	9	2	2023-05-22	MRI scan of the hearth

18. Medicines Table:



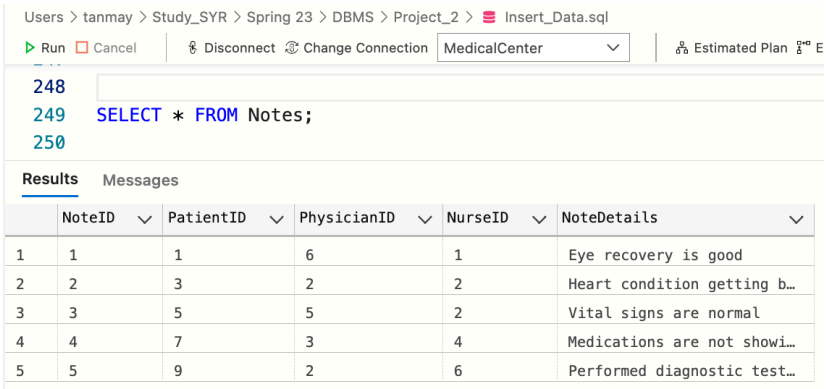
	MedicineID	MedicineName	Manufacturer
1	1	Paracetamol	Johnson & Johnson
2	2	Ibuprofen	Pfizer
3	3	Aspirin	Johnson & Johnson
4	4	Amoxicillin	Novartis
5	5	Atorvastatin	Pfizer
6	6	Metformin	AstraZeneca
7	7	Omeprazole	AstraZeneca
8	8	Simvastatin	Merck
9	9	Levothyroxine	Pfizer
10	10	Losartan	Merck

19. Medications Table:



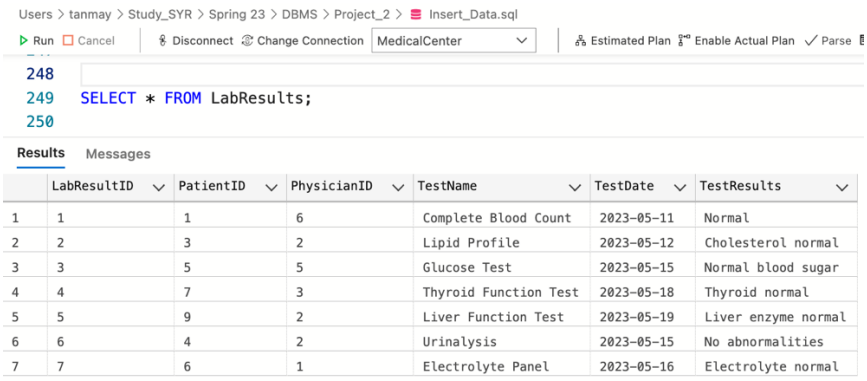
	MedicationID	PatientID	PhysicianID	MedicineID	Dosage	Frequency
1	1	1	6	3	3 Tabs	3x – 3 Days
2	2	1	6	5	2 Tabs	1x – 4 Days
3	3	2	5	2	20 ml	2x – 2 Days
4	4	3	2	8	1 Tab	1x – 6 Days
5	5	3	2	6	30 ml	2x – 3 Days
6	6	4	2	1	4 Tabs	2x – 3 Days
7	7	5	5	2	20 ml	3x – 3 Days
8	8	6	1	7	10 ml	1x – 5 Days
9	9	7	3	4	2 Tabs	3x – 5 Days
10	10	7	3	9	5 ml	2x – 7 Days
11	11	7	3	10	3 Tabs	1x – 7 Days
12	12	8	5	2	15 ml	3x – 3 Days
13	13	9	2	1	2 Tabs	2x – 4 Days
14	14	10	5	5	4 Tabs	1x – 5 Days

20. Notes Table:



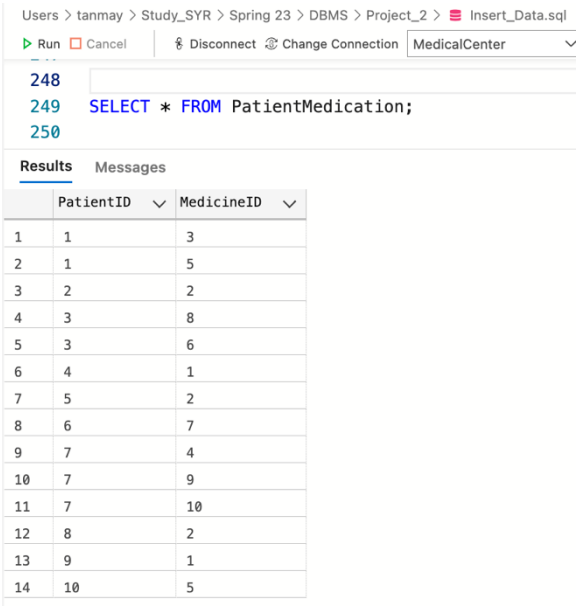
	NoteID	PatientID	PhysicianID	NurseID	NoteDetails
1	1	1	6	1	Eye recovery is good
2	2	3	2	2	Heart condition getting b…
3	3	5	5	2	Vital signs are normal
4	4	7	3	4	Medications are not showi…
5	5	9	2	6	Performed diagnostic test…

21. LabResults Table:



	LabResultID	PatientID	PhysicianID	TestName	TestDate	TestResults
1	1	1	6	Complete Blood Count	2023-05-11	Normal
2	2	3	2	Lipid Profile	2023-05-12	Cholesterol normal
3	3	5	5	Glucose Test	2023-05-15	Normal blood sugar
4	4	7	3	Thyroid Function Test	2023-05-18	Thyroid normal
5	5	9	2	Liver Function Test	2023-05-19	Liver enzyme normal
6	6	4	2	Urinalysis	2023-05-15	No abnormalities
7	7	6	1	Electrolyte Panel	2023-05-16	Electrolyte normal

22. PatientMedication Table:



	PatientID	MedicineID
1	1	3
2	1	5
3	2	2
4	3	8
5	3	6
6	4	1
7	5	2
8	6	7
9	7	4
10	7	9
11	7	10
12	8	2
13	9	1
14	10	5

Here we can see all the values in table that were inserted.

3. Testing Phase

Here in this section we will be creating Views, Stored Procedures, Functions, Triggers, Transaction and Scripts to test how everything is working with this database model.

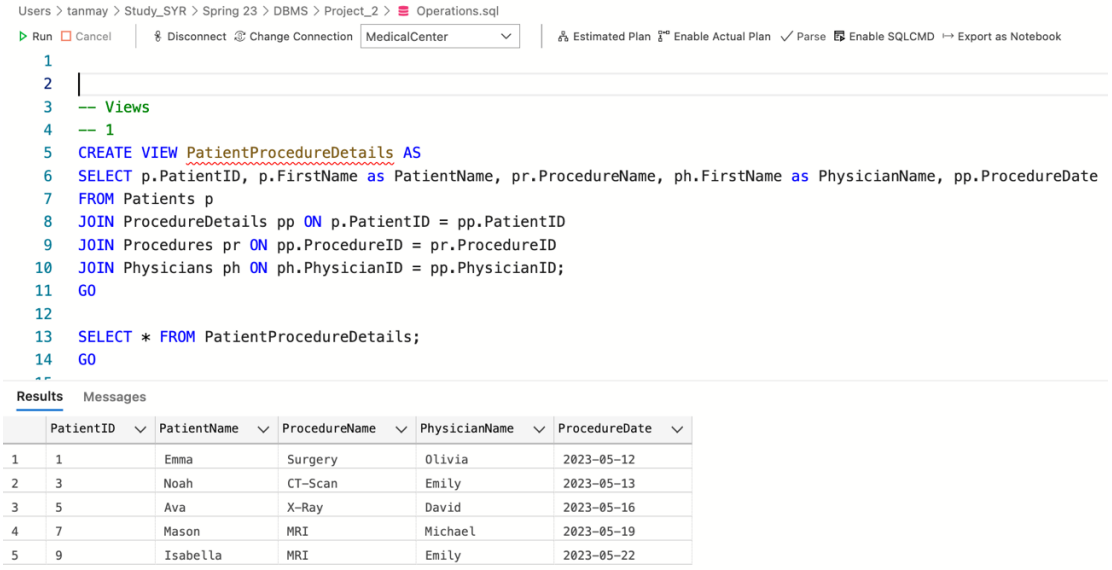
3.1 Views

1. PatientProcedureDetails

Code:

```
CREATE VIEW PatientProcedureDetails AS
SELECT p.PatientID, p.FirstName as PatientName, pr.ProcedureName, ph.FirstName as PhysicianName,
pp.ProcedureDate
FROM Patients p
JOIN ProcedureDetails pp ON p.PatientID = pp.PatientID
JOIN Procedures pr ON pp.ProcedureID = pr.ProcedureID
JOIN Physicians ph ON ph.PhysicianID = pp.PhysicianID;
GO
SELECT * FROM PatientProcedureDetails;
GO
```

Execution:



```
Users > tanmay > Study_SYR > Spring 23 > DBMS > Project_2 > Operations.sql
Run  Cancel | Disconnect  Change Connection  MedicalCenter | Estimated Plan  Enable Actual Plan  Parse  Enable SQLCMD  Export as Notebook

1
2
3   -- Views
4   -- 1
5   CREATE VIEW PatientProcedureDetails AS
6   SELECT p.PatientID, p.FirstName as PatientName, pr.ProcedureName, ph.FirstName as PhysicianName, pp.ProcedureDate
7   FROM Patients p
8   JOIN ProcedureDetails pp ON p.PatientID = pp.PatientID
9   JOIN Procedures pr ON pp.ProcedureID = pr.ProcedureID
10  JOIN Physicians ph ON ph.PhysicianID = pp.PhysicianID;
11  GO
12
13  SELECT * FROM PatientProcedureDetails;
14  GO
```

Results | Messages

	PatientID	PatientName	ProcedureName	PhysicianName	ProcedureDate
1	1	Emma	Surgery	Olivia	2023-05-12
2	3	Noah	CT-Scan	Emily	2023-05-13
3	5	Ava	X-Ray	David	2023-05-16
4	7	Mason	MRI	Michael	2023-05-19
5	9	Isabella	MRI	Emily	2023-05-22

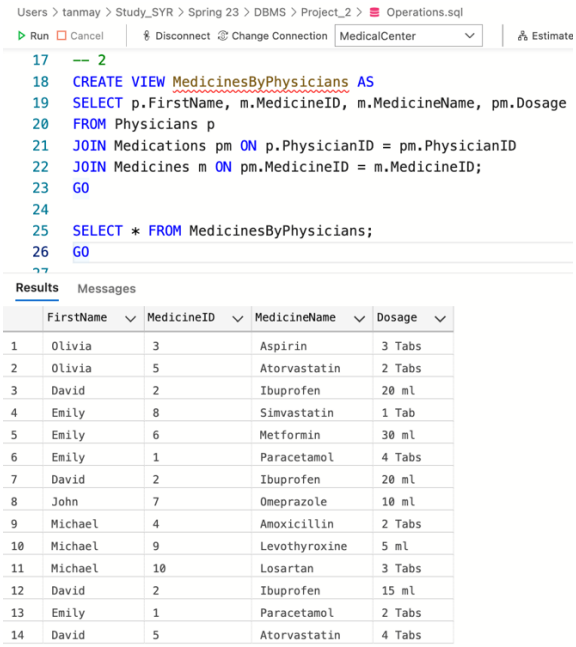
Comment:
This view gives the details about the patients who are having a procedure along with the physician's name along with the procedure date.

2. MedicinesByPhysicians

Code:

```
CREATE VIEW MedicinesByPhysicians AS
SELECT p.FirstName, m.MedicineID, m.MedicineName, pm.Dosage
FROM Physicians p
JOIN Medications pm ON p.PhysicianID = pm.PhysicianID
JOIN Medicines m ON pm.MedicineID = m.MedicineID;
GO

SELECT * FROM MedicinesByPhysicians;
GO
```

Execution:



Users > tanmay > Study_SYR > Spring 23 > DBMS > Project_2 > Operations.sql

Run | Cancel | Disconnect | Change Connection | MedicalCenter | Estimate

```
17  -- 2
18  CREATE VIEW MedicinesByPhysicians AS
19  SELECT p.FirstName, m.MedicineID, m.MedicineName, pm.Dosage
20  FROM Physicians p
21  JOIN Medications pm ON p.PhysicianID = pm.PhysicianID
22  JOIN Medicines m ON pm.MedicineID = m.MedicineID;
23  GO
24
25  SELECT * FROM MedicinesByPhysicians;
26  GO
```

Results | Messages

	FirstName	MedicineID	MedicineName	Dosage
1	Olivia	3	Aspirin	3 Tabs
2	Olivia	5	Atorvastatin	2 Tabs
3	David	2	Ibuprofen	20 ml
4	Emily	8	Simvastatin	1 Tab
5	Emily	6	Metformin	30 ml
6	Emily	1	Paracetamol	4 Tabs
7	David	2	Ibuprofen	20 ml
8	John	7	Omeprazole	10 ml
9	Michael	4	Amoxicillin	2 Tabs
10	Michael	9	Levothyroxine	5 ml
11	Michael	10	Losartan	3 Tabs
12	David	2	Ibuprofen	15 ml
13	Emily	1	Paracetamol	2 Tabs
14	David	5	Atorvastatin	4 Tabs

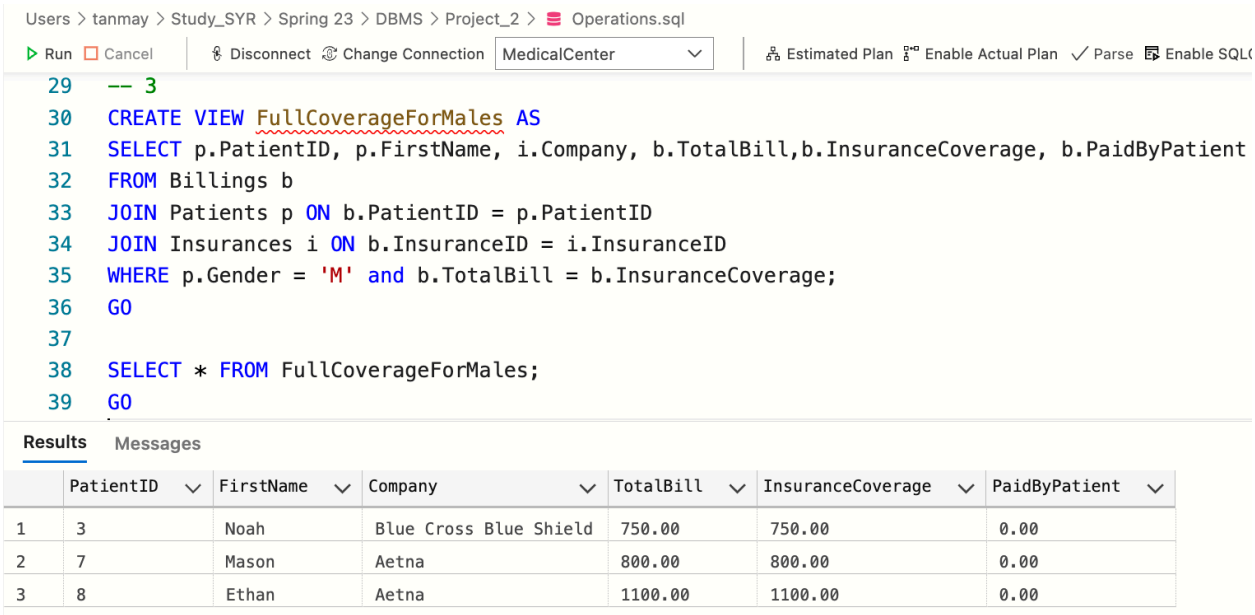
Comment:
This view gives the details about the medicines given by physicians along with the dosage that was prescribed to the patients.

3. FullCoverageForMales

Code:

```
CREATE VIEW FullCoverageForMales AS
SELECT p.PatientID, p.FirstName, i.Company, b.TotalBill,b.InsuranceCoverage, b.PaidByPatient
FROM Billings b
JOIN Patients p ON b.PatientID = p.PatientID
JOIN Insurances i ON b.InsuranceID = i.InsuranceID
WHERE p.Gender = 'M' and b.TotalBill = b.InsuranceCoverage;
GO

SELECT * FROM FullCoverageForMales;
GO
```

Execution:



	PatientID	FirstName	Company	TotalBill	InsuranceCoverage	PaidByPatient
1	3	Noah	Blue Cross Blue Shield	750.00	750.00	0.00
2	7	Mason	Aetna	800.00	800.00	0.00
3	8	Ethan	Aetna	1100.00	1100.00	0.00

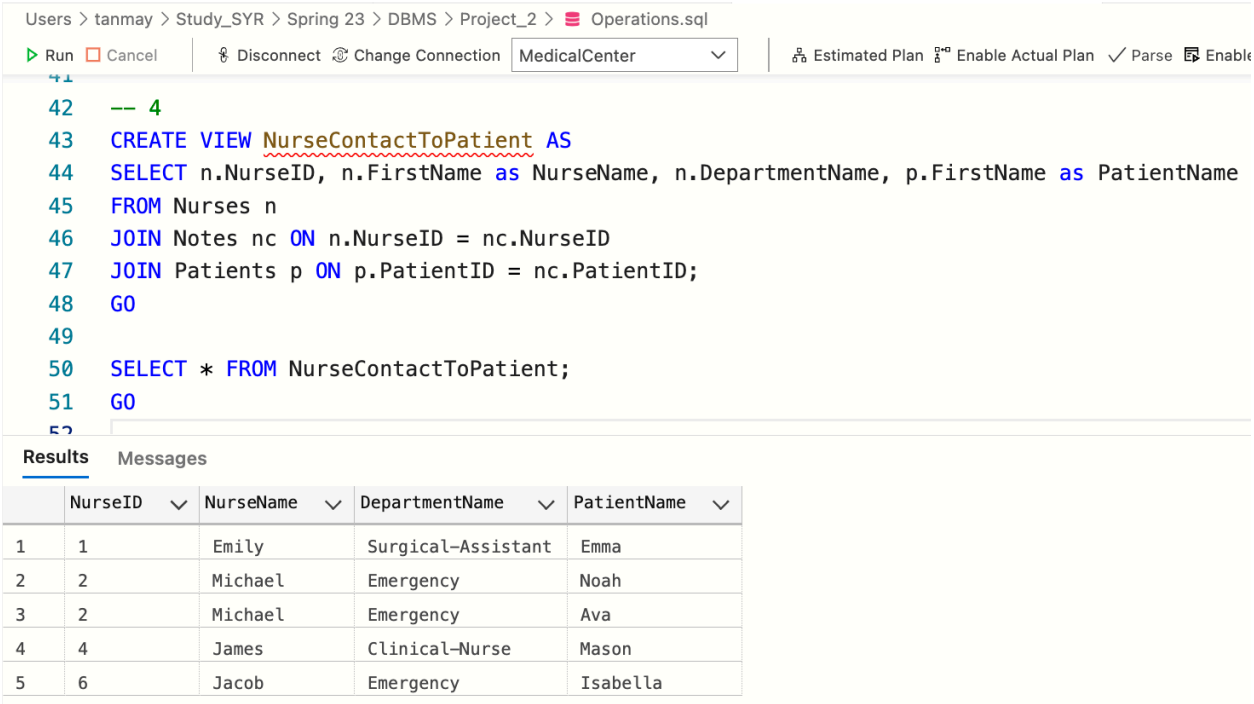
Comments:
This view gives the billing details for 'Male' patients for whom the insurance company covered the entire medical expense.

4. NurseContactToPatient

Code:

```
CREATE VIEW NurseContactToPatient AS
SELECT n.NurseID, n.FirstName as NurseName, n.DepartmentName, p.FirstName as PatientName
FROM Nurses n
JOIN Notes nc ON n.NurseID = nc.NurseID
JOIN Patients p ON p.PatientID = nc.PatientID;
GO

SELECT * FROM NurseContactToPatient;
GO
```

Execution:



	NurseID	NurseName	DepartmentName	PatientName
1	1	Emily	Surgical-Assistant	Emma
2	2	Michael	Emergency	Noah
3	2	Michael	Emergency	Ava
4	4	James	Clinical-Nurse	Mason
5	6	Jacob	Emergency	Isabella

Comment:
This view gives the nurse details like their name and department for the patients they were assigned to.

3.2 Stored Procedures

1. InsertPatientDetails

Code:

```
CREATE PROCEDURE InsertPatientDetails
(
   @FirstName VARCHAR(20),@LastName VARCHAR(20),@DateOfBirth DATE,@Gender VARCHAR(1),
   @PNumber VARCHAR(15),@ENumber VARCHAR(15),@Email VARCHAR(30),
   @HouseNum VARCHAR(4),@Street VARCHAR(20),@City VARCHAR(20),@State
VARCHAR(20),@ZipCode VARCHAR(6)
)
AS
BEGIN
   INSERT INTO Patients (FirstName, LastName, DateOfBirth, Gender)
   VALUES (@FirstName, @LastName, @DateOfBirth, @Gender)
   INSERT INTO PatientContact(PrimaryNumber, EmergencyNumber, Email)
   VALUES (@PNumber, @ENumber, @Email)
   INSERT INTO PatientAddress (HouseNumber, Street, City, State, ZipCode)
   VALUES (@HouseNum, @Street, @City, @State, @ZipCode)
END
GO

EXEC InsertPatientDetails
   @FirstName = 'John', @LastName = 'Jill', @DateOfBirth = '1990-12-12', @Gender = 'M',
   @PNumber = '773-456-7898', @ENumber = '985-654-3217', @Email = 'john.jill@example.com',
   @HouseNum = '993', @Street = 'Main Street', @City = 'New York', @State = 'NY', @ZipCode = '15645';
GO

SELECT TOP 1 * FROM Patients ORDER BY PatientID DESC
SELECT TOP 1 * FROM PatientContact ORDER BY PatientID DESC;
SELECT TOP 1 * FROM PatientAddress ORDER BY PatientID DESC;
GO
```

Execution:

Users > tanmay > Study_SYR > Spring 23 > DBMS > Project_2 > Operations.sql

Run | Cancel | Disconnect | Change Connection | MedicalCenter | Estimated Plan | Enable Actual Plan | Parse | Enable SQLCMD | Export as

```
56  -- Procedures
57  -- 1
58  CREATE PROCEDURE InsertPatientDetails
59  (
60      @FirstName VARCHAR(20),@LastName VARCHAR(20),@DateOfBirth DATE,@Gender VARCHAR(1),
61      @PNumber VARCHAR(15),@ENumber VARCHAR(15),@Email VARCHAR(30),
62      @HouseNum VARCHAR(4),@Street VARCHAR(20),@City VARCHAR(20),@State VARCHAR(20),@ZipCode VARCHAR(6)
63  )
64  AS
65  BEGIN
66      INSERT INTO Patients (FirstName, LastName, DateOfBirth, Gender)
67      VALUES (@FirstName, @LastName, @DateOfBirth, @Gender)
68      INSERT INTO PatientContact(PrimaryNumber, EmergencyNumber, Email)
69      VALUES (@PNumber, @ENumber, @Email)
70      INSERT INTO PatientAddress (HouseNumber, Street, City, State, ZipCode)
71      VALUES (@HouseNum, @Street, @City, @State, @ZipCode)
72  END
73  GO
74
75  EXEC InsertPatientDetails
76      @FirstName = 'John', @LastName = 'Jill', @DateOfBirth = '1990-12-12', @Gender = 'M',
77      @PNumber = '773-456-7898', @ENumber = '985-654-3217', @Email = 'john.jill@example.com',
78      @HouseNum = '993', @Street = 'Main Street', @City = 'New York', @State = 'NY', @ZipCode = '15645';
79  GO
80
81  SELECT TOP 1 * FROM Patients ORDER BY PatientID DESC
82  SELECT TOP 1 * FROM PatientContact ORDER BY PatientID DESC;
83  SELECT TOP 1 * FROM PatientAddress ORDER BY PatientID DESC;
84  GO
```

Results | Messages

	PatientID	FirstName	LastName	DateOfBirth	Gender
1	10	Charlotte	Thomas	1962-03-25	F

	PatientID	PrimaryNumber	EmergencyNumber	Email
1	11	773-456-7898	985-654-3217	john.jill@example.com

	PatientID	HouseNumber	Street	City	State	ZipCode
1	11	993	Main Street	New York	NY	15645

Comment:

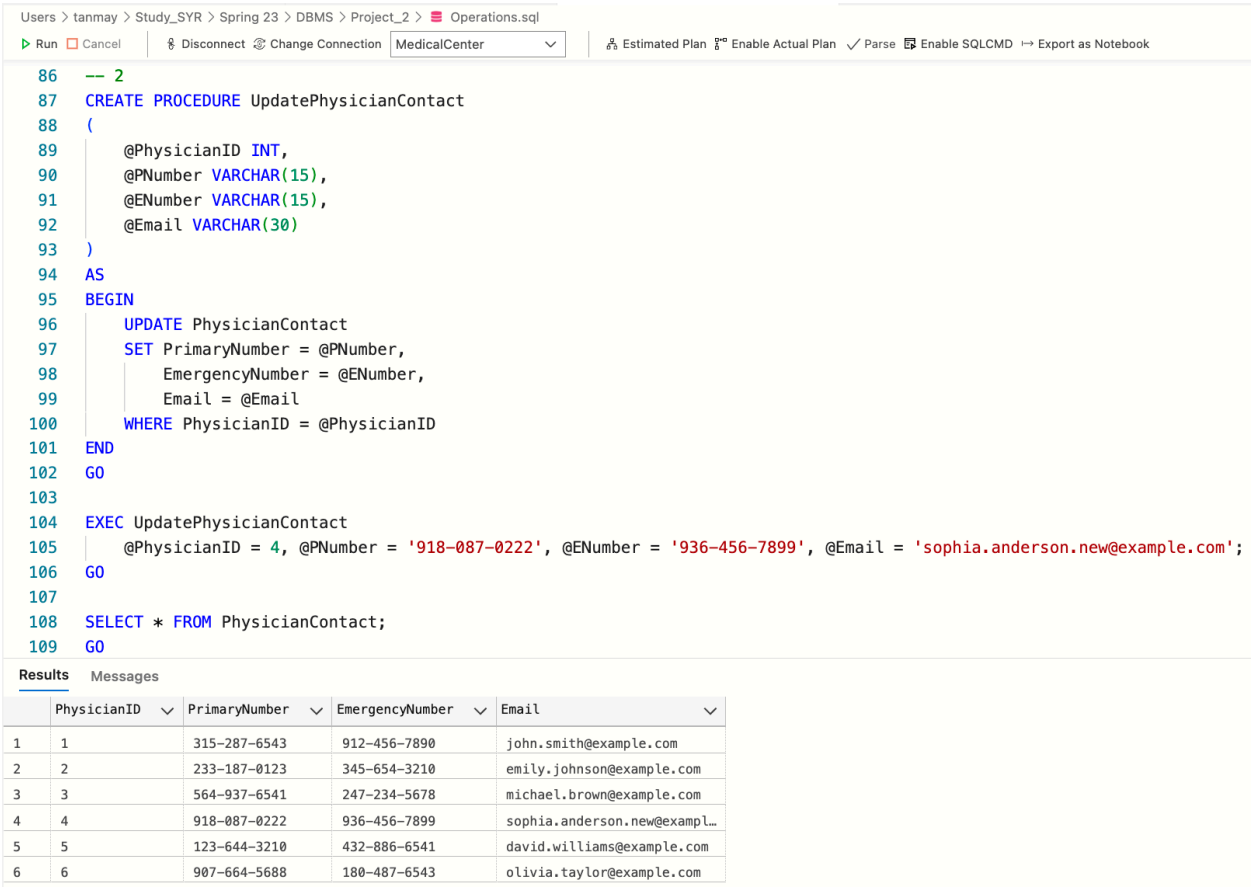
This stored procedure is used to insert patient details into Patients, PatientAddress and PatientContact table. The SELECT query which is used displays the latest values that was inserted in the table. Here we can see that PatientID is now 11 after inserting new values (initially there were only 10 rows in this table).

2. UpdatePhysicianContact

Code:

```
CREATE PROCEDURE UpdatePhysicianContact
(
   @PhysicianID INT,
   @PNumber VARCHAR(15),
   @ENumber VARCHAR(15),
   @Email VARCHAR(30)
)
AS
BEGIN
   UPDATE PhysicianContact
   SET PrimaryNumber = @PNumber,
      EmergencyNumber = @ENumber,
      Email = @Email
   WHERE PhysicianID = @PhysicianID
END
GO

EXEC UpdatePhysicianContact
   @PhysicianID = 4, @PNumber = '918-087-0222', @ENumber = '936-456-7899', @Email =
'sophia.anderson.new@example.com';
GO

SELECT * FROM PhysicianContact;
GO
```

Execution:



Comment:

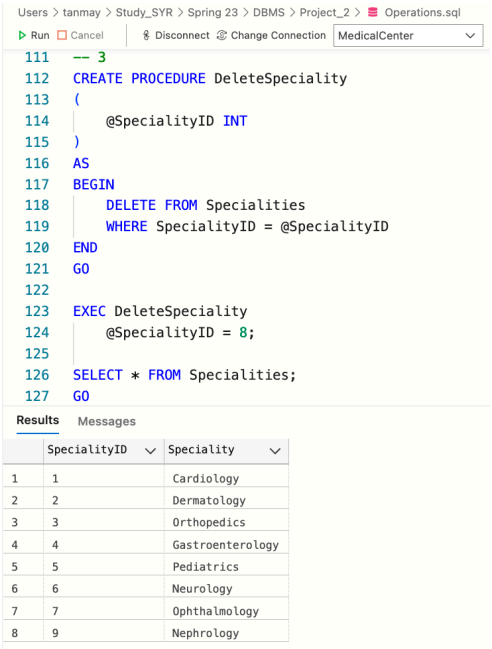
In this procedure we update the physician's contact details. We can see that the details for physician with PhysicianID = 4 have been updated.

3. DeleteSpeciality

Code:

```
CREATE PROCEDURE DeleteSpeciality
(
   @SpecialityID INT
)
AS
BEGIN
   DELETE FROM Specialities
   WHERE SpecialityID = @SpecialityID
END
GO

EXEC DeleteSpeciality
   @SpecialityID = 8;

SELECT * FROM Specialities;
GO
```

Execution:



Comment:

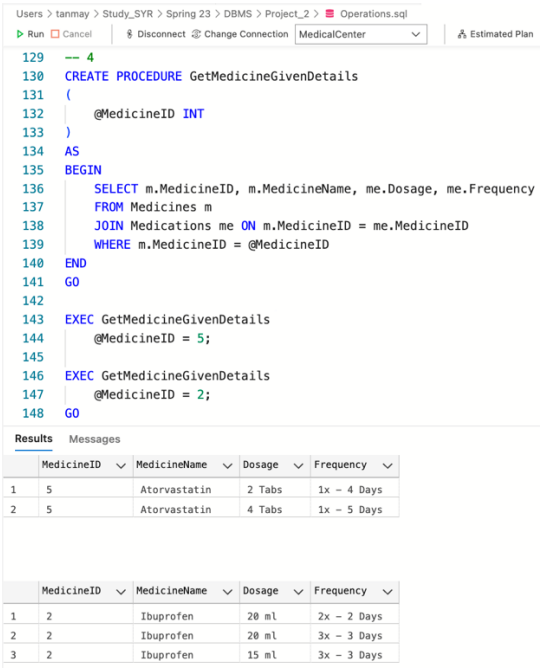
This procedure deletes the speciality from the Specialities table.

4. GetMedicineGivenDetails

Code:

```
CREATE PROCEDURE GetMedicineGivenDetails
(
   @MedicineID INT
)
AS
BEGIN
   SELECT m.MedicineID, m.MedicineName, me.Dosage, me.Frequency
   FROM Medicines m
   JOIN Medications me ON m.MedicineID = me.MedicineID
   WHERE m.MedicineID = @MedicineID
END
GO
EXEC GetMedicineGivenDetails
   @MedicineID = 5;
EXEC GetMedicineGivenDetails
   @MedicineID = 2;
```

Execution:



Users > tanmay > Study_SYR > Spring 23 > DBMS > Project_2 > Operations.sql

Run | Cancel | Disconnect | Change Connection | MedicalCenter | Estimated Plan

```
129  -- 4
130  CREATE PROCEDURE GetMedicineGivenDetails
131  (
132      @MedicineID INT
133  )
134  AS
135  BEGIN
136      SELECT m.MedicineID, m.MedicineName, me.Dosage, me.Frequency
137      FROM Medicines m
138      JOIN Medications me ON m.MedicineID = me.MedicineID
139      WHERE m.MedicineID = @MedicineID
140  END
141  GO
142
143  EXEC GetMedicineGivenDetails
144      @MedicineID = 5;
145
146  EXEC GetMedicineGivenDetails
147      @MedicineID = 2;
148  GO
```

Results | Messages

	MedicineID	MedicineName	Dosage	Frequency
1	5	Atorvastatin	2 Tabs	1x – 4 Days
2	5	Atorvastatin	4 Tabs	1x – 5 Days

	MedicineID	MedicineName	Dosage	Frequency
1	2	Ibuprofen	20 ml	2x – 2 Days
2	2	Ibuprofen	20 ml	3x – 3 Days
3	2	Ibuprofen	15 ml	3x – 3 Days

Comment:

This procedure displays the medicines that was prescribed to patients along with the dosage and frequency.

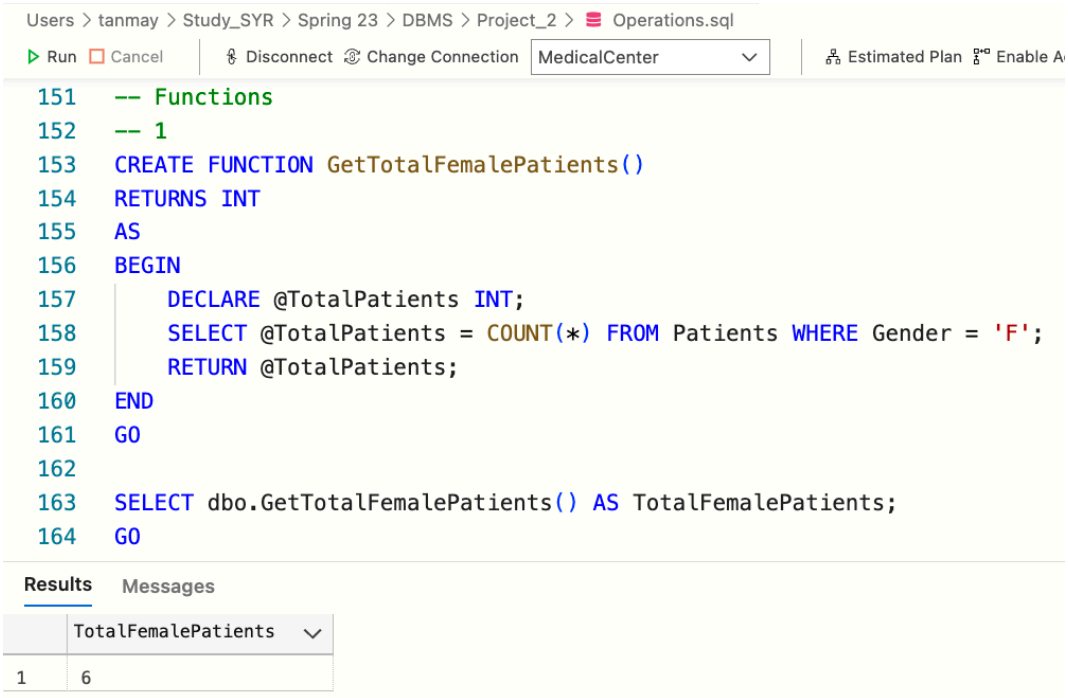
3.3 User-defined Functions

1. GetTotalFemalePatients

Code:

```
CREATE FUNCTION GetTotalFemalePatients()
RETURNS INT
AS
BEGIN
   DECLARE @TotalPatients INT;
   SELECT @TotalPatients = COUNT(*) FROM Patients WHERE Gender = 'F';
   RETURN @TotalPatients;
END
GO
SELECT dbo.GetTotalFemalePatients() AS TotalFemalePatients;
GO
```

Execution:



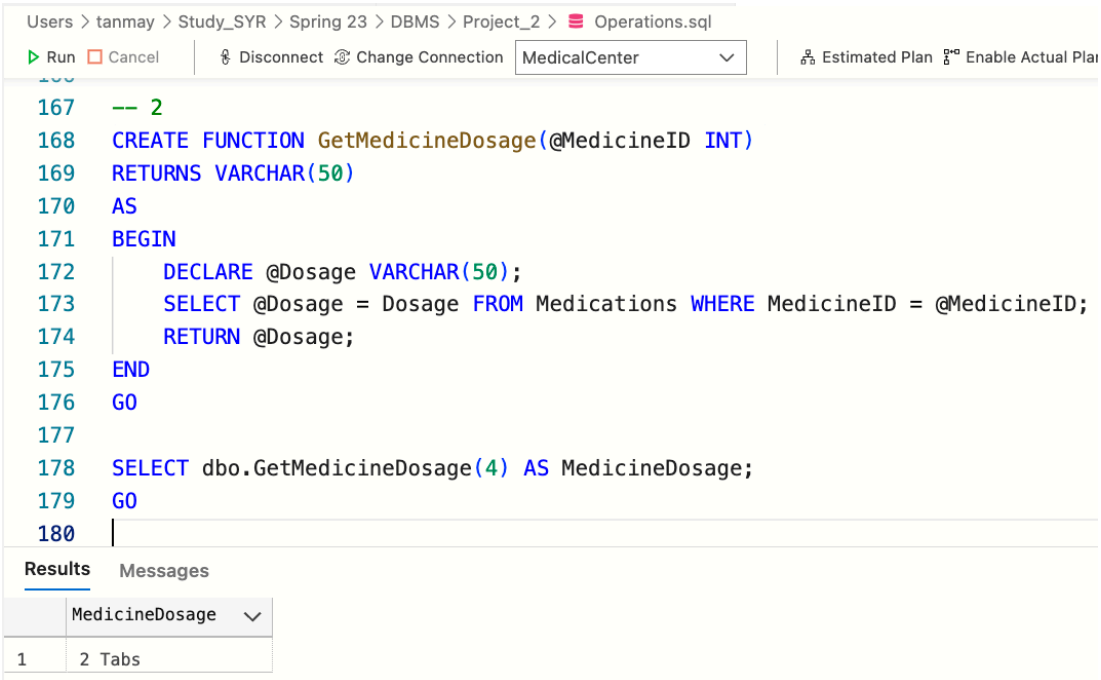
Comment:
This function displays the total female patients in the database.

2. GetMedicineDosage

Code:

```
CREATE FUNCTION GetMedicineDosage(@MedicineID INT)
RETURNS VARCHAR(50)
AS
BEGIN
   DECLARE @Dosage VARCHAR(50);
   SELECT @Dosage = Dosage FROM Medications WHERE MedicineID = @MedicineID;
   RETURN @Dosage;
END
GO
SELECT dbo.GetMedicineDosage(4) AS MedicineDosage;
GO
```

Execution:



Comment:
This function gives the medicine dosage prescribed for the selected medicine.

3. GetPatientAge

Code:

```
CREATE FUNCTION GetPatientAge(@PatientID INT)
RETURNS INT
AS
BEGIN
   DECLARE @Age INT;
   SELECT @Age = DATEDIFF(YEAR, DateOfBirth, GETDATE()) FROM Patients WHERE PatientID =
@PatientID;
   RETURN @Age;
END
GO
SELECT dbo.GetPatientAge(6) AS PatientAge;
GO
```

Execution:

Comment:

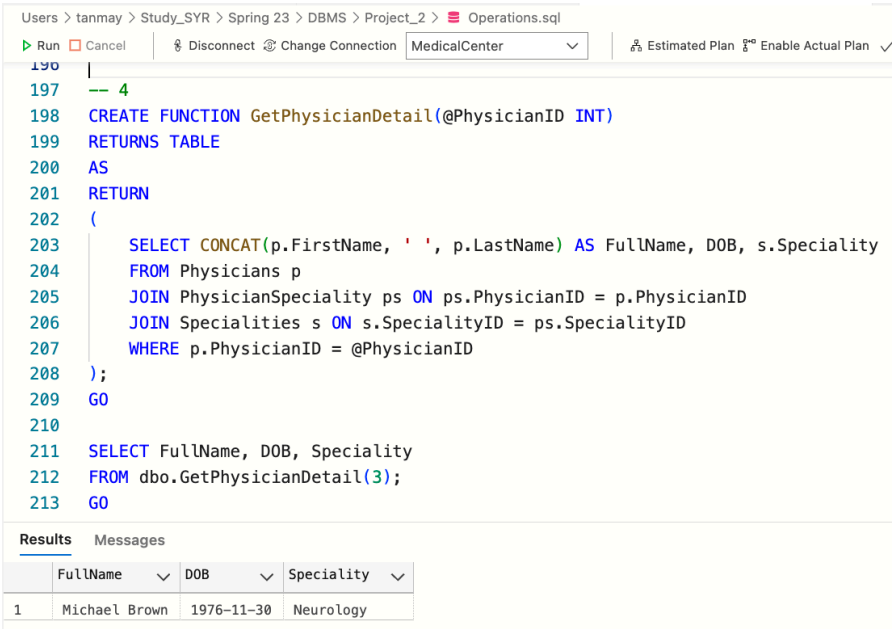
This function gives the current age for the patient based on their date of birth for the given PatientID.

4. GetPhysicianDetails

Code:

```
CREATE FUNCTION GetPhysicianDetail(@PhysicianID INT)
RETURNS TABLE
AS
RETURN
(
   SELECT CONCAT(p.FirstName, ' ', p.LastName) AS FullName, DOB, s.Speciality
   FROM Physicians p
   JOIN PhysicianSpeciality ps ON ps.PhysicianID = p.PhysicianID
   JOIN Specialities s ON s.SpecialityID = ps.SpecialityID
   WHERE p.PhysicianID = @PhysicianID
);
GO
SELECT FullName, DOB, Speciality
FROM dbo.GetPhysicianDetail(3);
GO
```

Execution:



Comment:

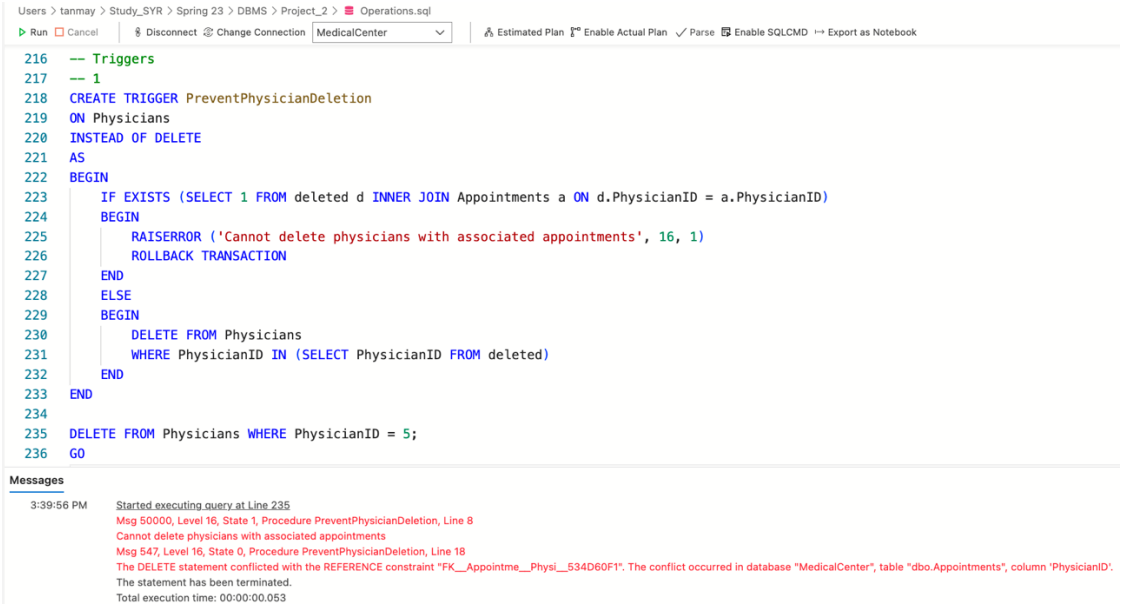
This function returns a table returning the full name, date of birth and speciality of a physician for a given PhysicianID.

3.4 Triggers

1. PreventPhysicianDeletion
Code:

```
CREATE TRIGGER PreventPhysicianDeletion
ON Physicians
INSTEAD OF DELETE
AS
BEGIN
  IF EXISTS (SELECT 1 FROM deleted d INNER JOIN Appointments a ON d.PhysicianID = a.PhysicianID)
  BEGIN
    RAISERROR ('Cannot delete physicians with associated appointments', 16, 1)
    ROLLBACK TRANSACTION
  END
  ELSE
  BEGIN
    DELETE FROM Physicians
    WHERE PhysicianID IN (SELECT PhysicianID FROM deleted)
  END
END
DELETE FROM Physicians WHERE PhysicianID = 5;
```

Execution:



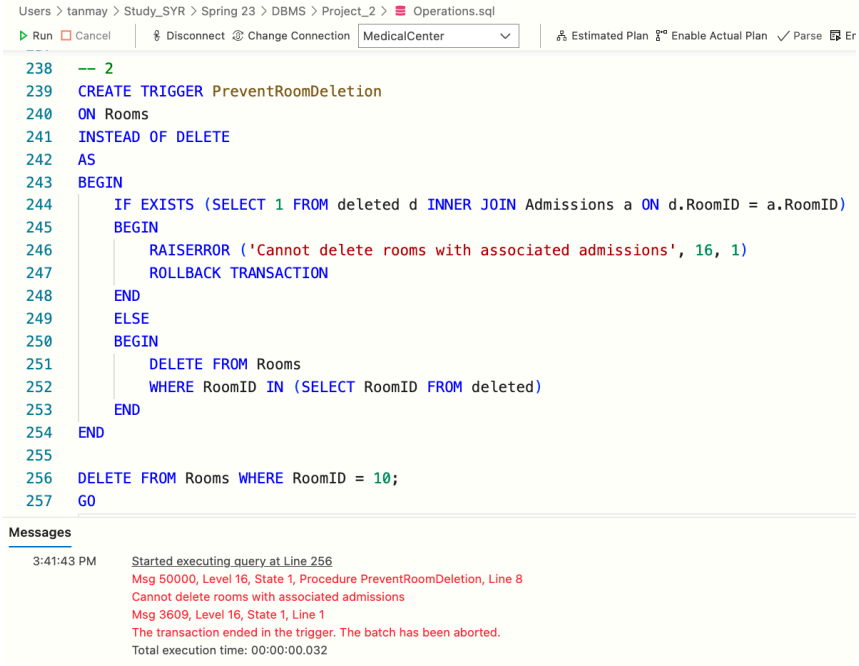
Comment:
This trigger is used to prevent deletion of records from the Physicians table if they have associated records in the Appointments table.

2. PreventRoomDeletion

Code:

```
CREATE TRIGGER PreventRoomDeletion
ON Rooms
INSTEAD OF DELETE
AS
BEGIN
  IF EXISTS (SELECT 1 FROM deleted d INNER JOIN Admissions a ON d.RoomID = a.RoomID)
  BEGIN
    RAISERROR ('Cannot delete rooms with associated admissions', 16, 1)
    ROLLBACK TRANSACTION
  END
  ELSE
  BEGIN
    DELETE FROM Rooms
    WHERE RoomID IN (SELECT RoomID FROM deleted)
  END
END
DELETE FROM Rooms WHERE RoomID = 10;
```

Execution:



Comment:

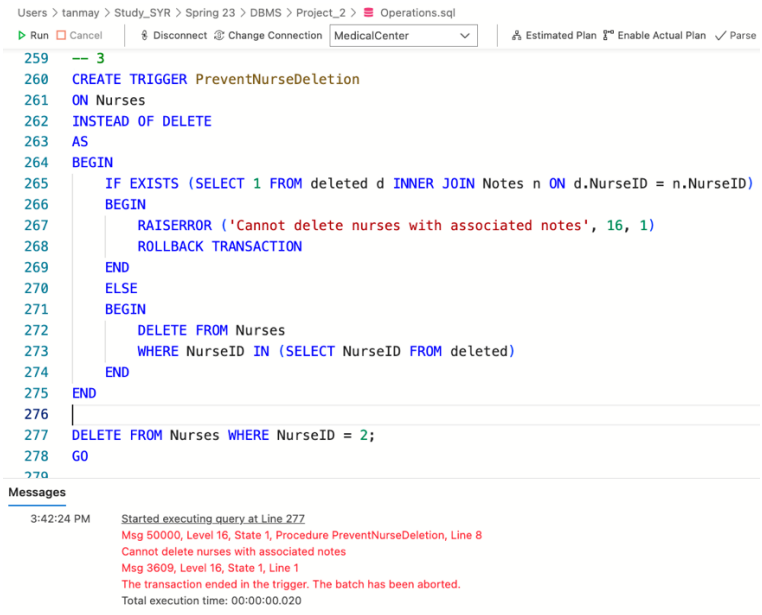
This trigger is used to prevent deletion of records from the Rooms table if they have associated records in the Admissions table.

3. PreventNurseDeletion

Code:

```
CREATE TRIGGER PreventNurseDeletion
ON Nurses
INSTEAD OF DELETE
AS
BEGIN
   IF EXISTS (SELECT 1 FROM deleted d INNER JOIN Notes n ON d.NurseID = n.NurseID)
   BEGIN
      RAISERROR ('Cannot delete nurses with associated notes', 16, 1)
      ROLLBACK TRANSACTION
   END
   ELSE
   BEGIN
      DELETE FROM Nurses
      WHERE NurseID IN (SELECT NurseID FROM deleted)
   END
END
DELETE FROM Nurses WHERE NurseID = 2;
```

Execution:



Comment:

This trigger is used to prevent deletion of records from the Nurses table if they have associated records in the Notes table.

4. PreventMedicineDeletion

Code:

```
CREATE TRIGGER PreventMedicineDeletion
ON Medicines
INSTEAD OF DELETE
AS
BEGIN
  IF EXISTS (SELECT 1 FROM deleted d INNER JOIN Medications m ON d.MedicineID = m.MedicineID)
  BEGIN
    RAISERROR ('Cannot delete medicines with associated medications', 16, 1)
    ROLLBACK TRANSACTION
  END
  ELSE
  BEGIN
    DELETE FROM Medicines
    WHERE MedicineID IN (SELECT MedicineID FROM deleted)
  END
END
DELETE FROM Medicines WHERE MedicineID = 10;
```

Execution:

Comment:
This trigger is used to prevent deletion of records from the Medicines table if they have associated records in the Medications table.

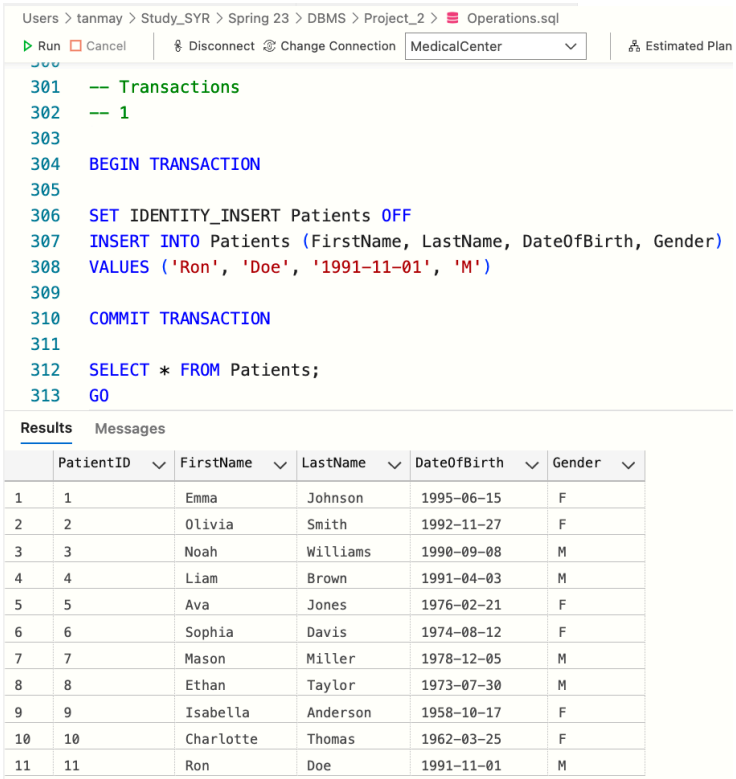
3.5 Transactions

Transaction 1:

Code:

```
BEGIN TRANSACTION

SET IDENTITY_INSERT Patients OFF
INSERT INTO Patients (FirstName, LastName, DateOfBirth, Gender)
VALUES ('Ron', 'Doe', '1991-11-01', 'M')

COMMIT TRANSACTION

SELECT * FROM Patients;
GO
```

Execution:



	PatientID	FirstName	LastName	DateOfBirth	Gender
1	1	Emma	Johnson	1995-06-15	F
2	2	Olivia	Smith	1992-11-27	F
3	3	Noah	Williams	1990-09-08	M
4	4	Liam	Brown	1991-04-03	M
5	5	Ava	Jones	1976-02-21	F
6	6	Sophia	Davis	1974-08-12	F
7	7	Mason	Miller	1978-12-05	M
8	8	Ethan	Taylor	1973-07-30	M
9	9	Isabella	Anderson	1958-10-17	F
10	10	Charlotte	Thomas	1962-03-25	F
11	11	Ron	Doe	1991-11-01	M

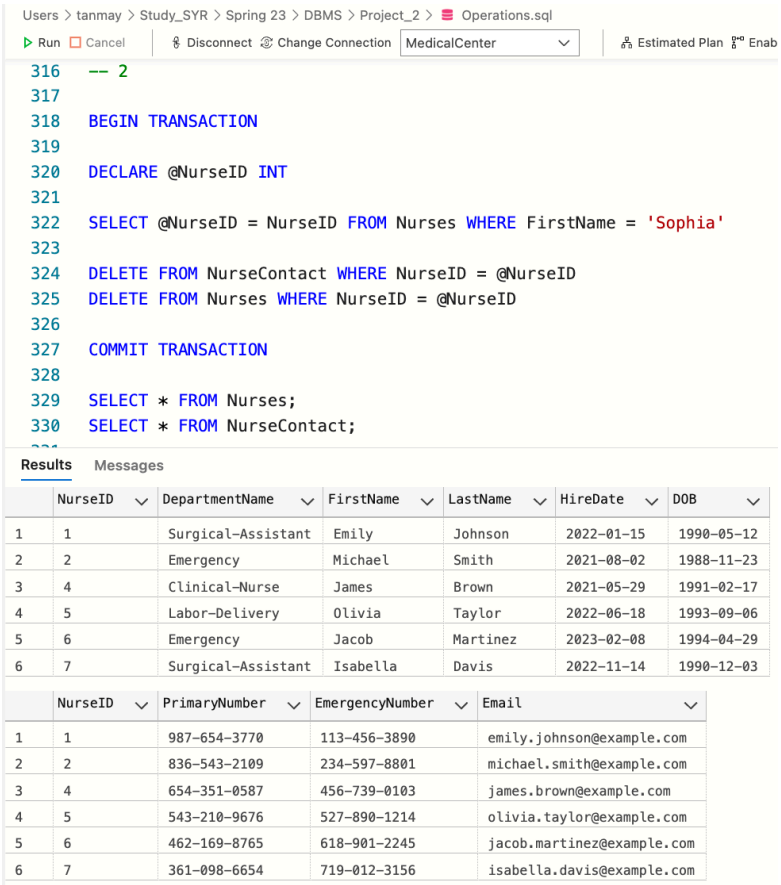
Comment:
Transaction to insert value in Patients table.

Transaction 2:

Code:

```
BEGIN TRANSACTION
DECLARE @NurseID INT
SELECT @NurseID = NurseID FROM Nurses WHERE FirstName = 'Sophia'
DELETE FROM NurseContact WHERE NurseID = @NurseID
DELETE FROM Nurses WHERE NurseID = @NurseID
COMMIT TRANSACTION

SELECT * FROM Nurses;
SELECT * FROM NurseContact;
```

Execution:



	NurseID	DepartmentName	FirstName	LastName	HireDate	DOB
1	1	Surgical-Assistant	Emily	Johnson	2022-01-15	1990-05-12
2	2	Emergency	Michael	Smith	2021-08-02	1988-11-23
3	4	Clinical-Nurse	James	Brown	2021-05-29	1991-02-17
4	5	Labor-Delivery	Olivia	Taylor	2022-06-18	1993-09-06
5	6	Emergency	Jacob	Martinez	2023-02-08	1994-04-29
6	7	Surgical-Assistant	Isabella	Davis	2022-11-14	1990-12-03

	NurseID	PrimaryNumber	EmergencyNumber	Email
1	1	987-654-3770	113-456-3890	emily.johnson@example.com
2	2	836-543-2109	234-597-8801	michael.smith@example.com
3	4	654-351-0587	456-739-0103	james.brown@example.com
4	5	543-210-9676	527-890-1214	olivia.taylor@example.com
5	6	462-169-8765	618-901-2245	jacob.martinez@example.com
6	7	361-098-6654	719-012-3156	isabella.davis@example.com

Comment:

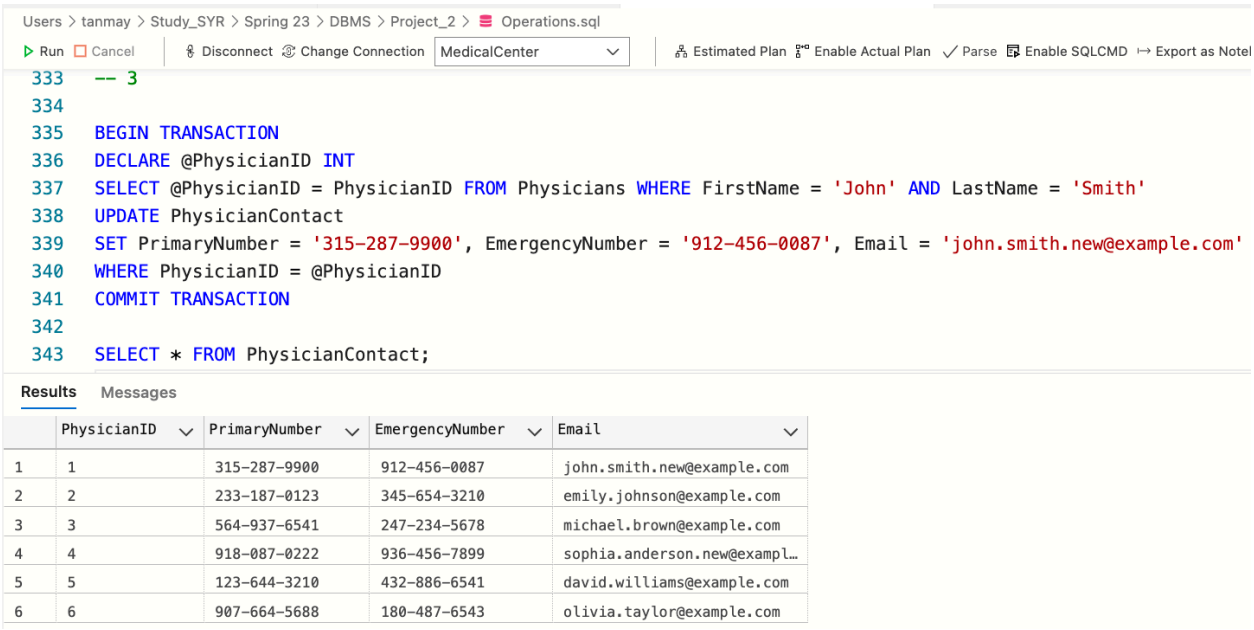
Here the nurse details of ‘Sophia’ are deleted from Nurses and NurseContact table.

Transaction 3:

Code:

```
BEGIN TRANSACTION
DECLARE @PhysicianID INT
SELECT @PhysicianID = PhysicianID FROM Physicians WHERE FirstName = 'John' AND LastName = 'Smith'
UPDATE PhysicianContact
SET PrimaryNumber = '315-287-9900', EmergencyNumber = '912-456-0087', Email =
'john.smith.new@example.com'
WHERE PhysicianID = @PhysicianID
COMMIT TRANSACTION

SELECT * FROM PhysicianContact;
```

Execution:



	PhysicianID	PrimaryNumber	EmergencyNumber	Email
1	1	315-287-9900	912-456-0087	john.smith.new@example.com
2	2	233-187-0123	345-654-3210	emily.johnson@example.com
3	3	564-937-6541	247-234-5678	michael.brown@example.com
4	4	918-087-0222	936-456-7899	sophia.anderson.new@exampl…
5	5	123-644-3210	432-886-6541	david.williams@example.com
6	6	907-664-5688	180-487-6543	olivia.taylor@example.com

Comment:

Here we can see that the values in PhysicianContact tables have been updated.

Transaction 4:

Code:

```
BEGIN TRANSACTION
DECLARE @NewMedicineName VARCHAR(30)
SET @NewMedicineName = 'NewLosartan'
UPDATE Medicines
SET MedicineName = @NewMedicineName
WHERE MedicineName = 'Losartan';
COMMIT TRANSACTION

SELECT * FROM Medicines;
```

Execution:

	MedicineID	MedicineName	Manufacturer
1	1	Paracetamol	Johnson & Johnson
2	2	Ibuprofen	Pfizer
3	3	Aspirin	Johnson & Johnson
4	4	Amoxicillin	Novartis
5	5	Atorvastatin	Pfizer
6	6	Metformin	AstraZeneca
7	7	Omeprazole	AstraZeneca
8	8	Simvastatin	Merck
9	9	Levothyroxine	Pfizer
10	10	NewLosartan	Merck

Comment:

Here we can see that the medicine name has been updated.

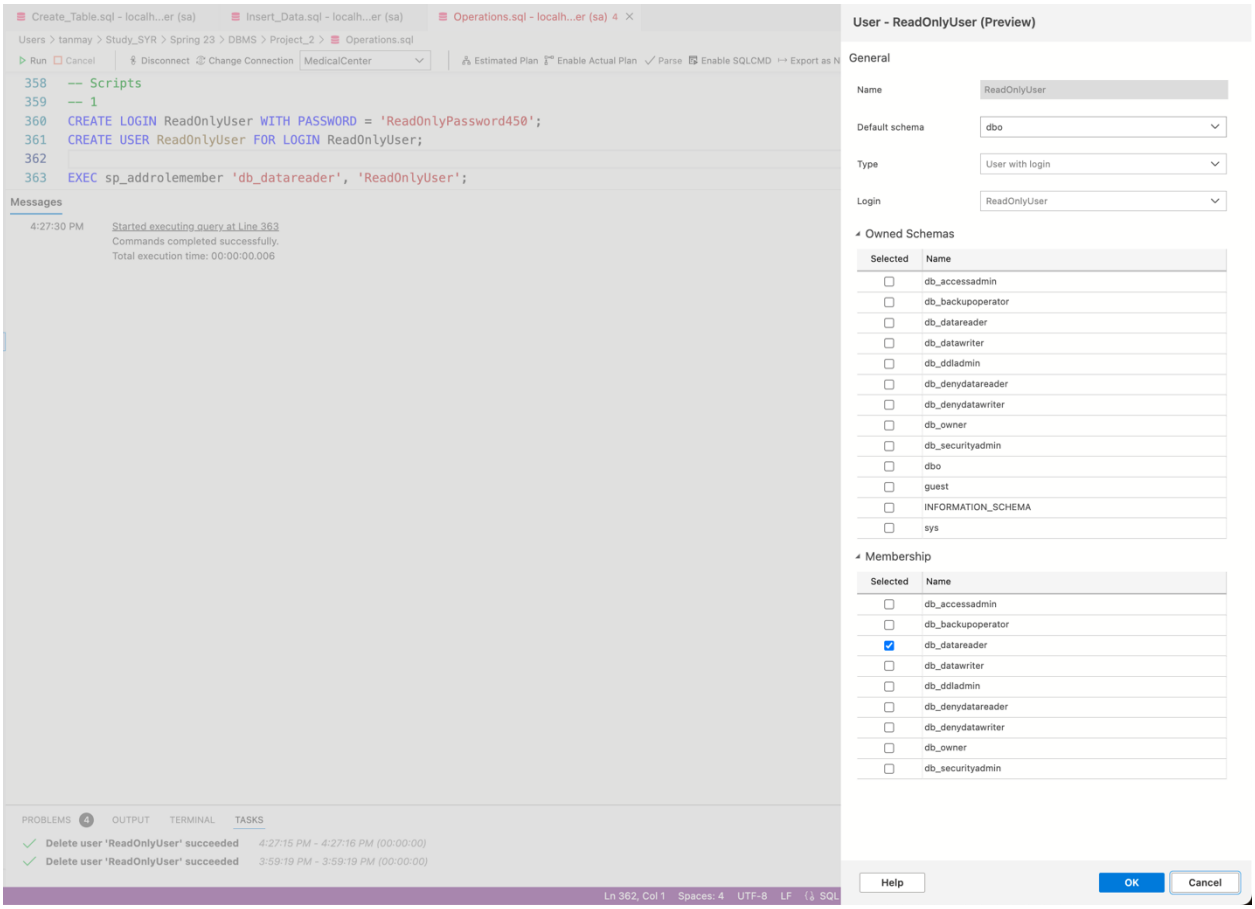
3.6 Scripts

Script 1:

Code:

```
CREATE LOGIN ReadOnlyUser WITH PASSWORD = 'ReadOnlyPassword450';
CREATE USER ReadOnlyUser FOR LOGIN ReadOnlyUser;

EXEC sp_addrolemember 'db_datareader', 'ReadOnlyUser';
```

Execution:



Comment:

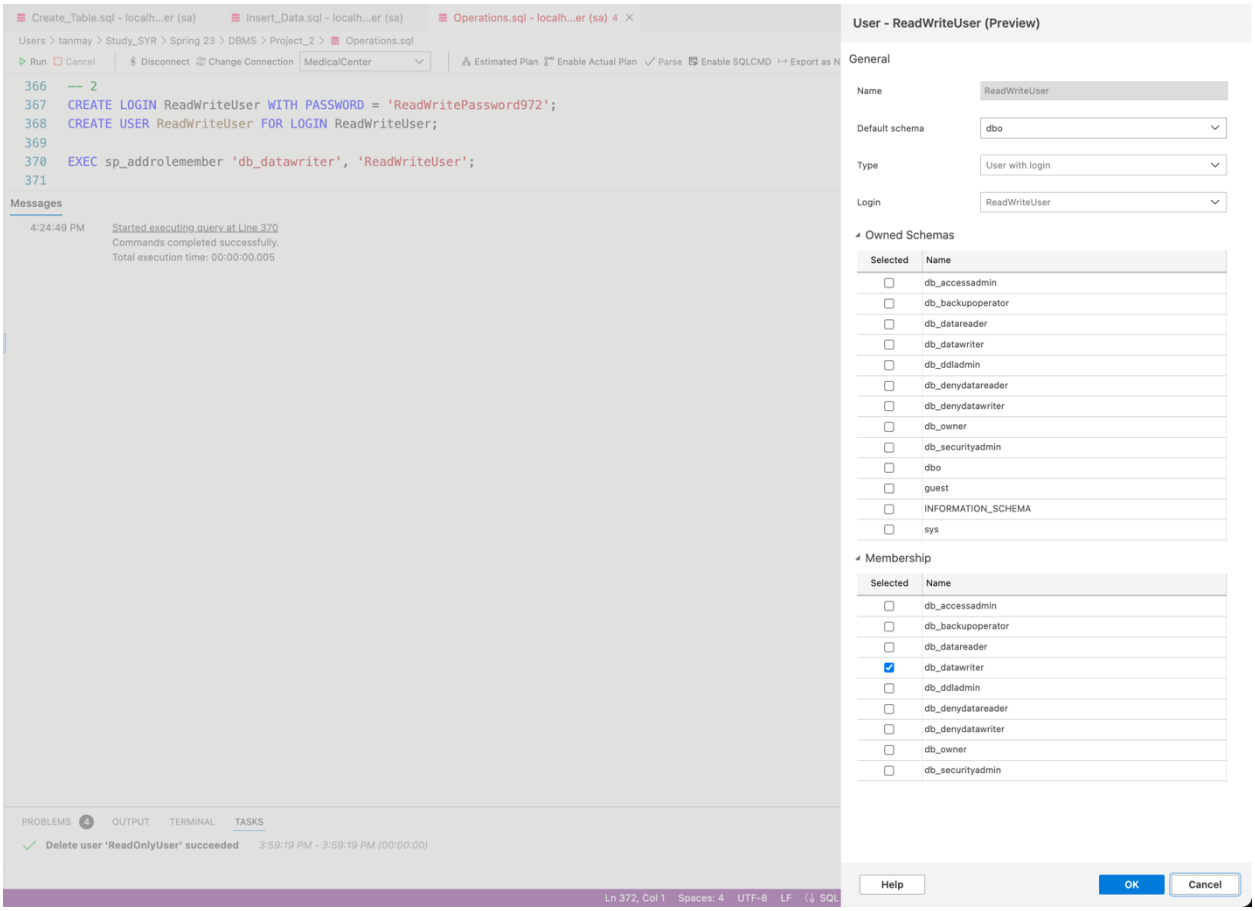
Here we can see that the new user is created who has data-reader membership.

Script 2:

Code:

```
CREATE LOGIN ReadWriteUser WITH PASSWORD = 'ReadWritePassword972';
CREATE USER ReadWriteUser FOR LOGIN ReadWriteUser;

EXEC sp_addrolemember 'db_datawriter', 'ReadWriteUser';
```

Execution:



Comment:

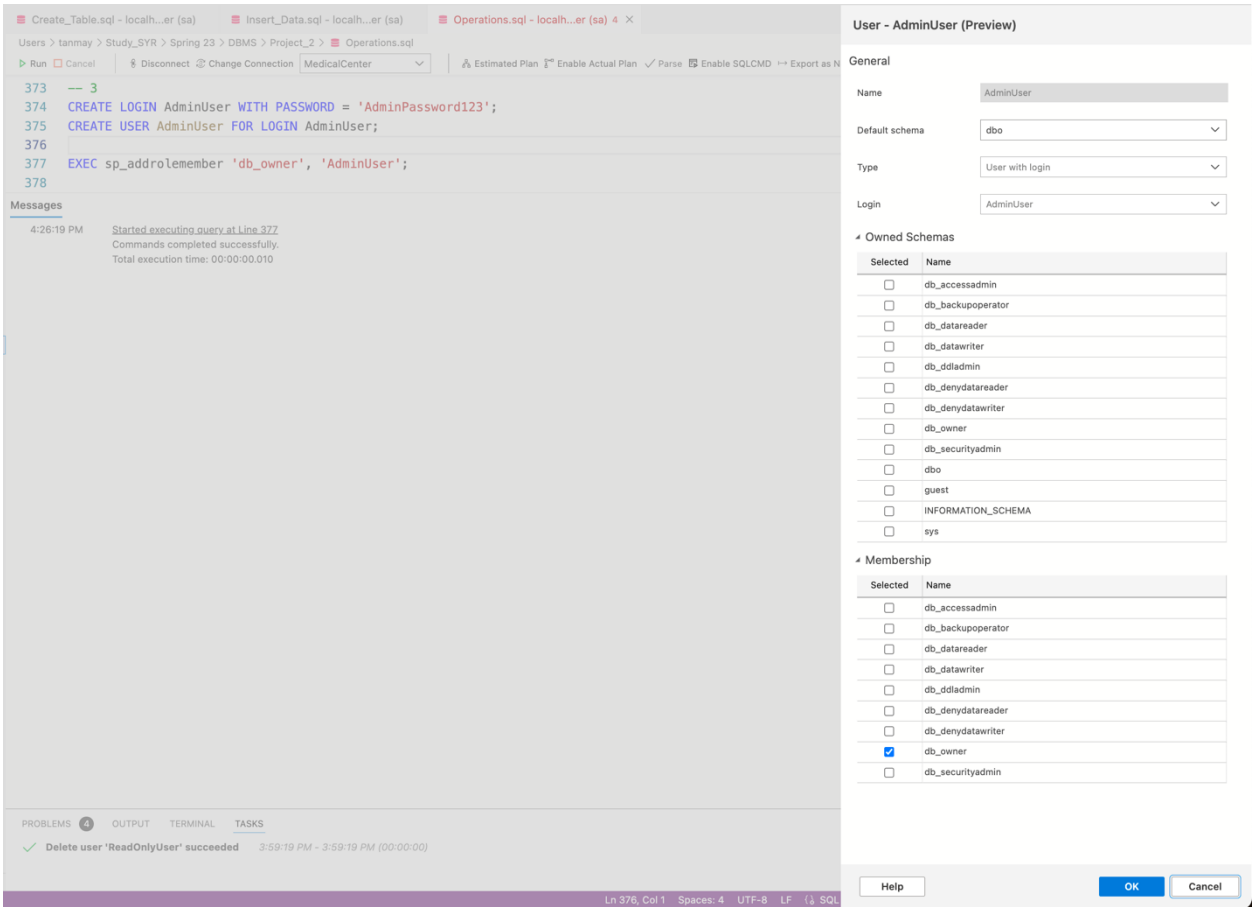
Here we can see that the new user is created who has data-writer membership.

Script 3:

Code:

```
CREATE LOGIN AdminUser WITH PASSWORD = 'AdminPassword123';
CREATE USER AdminUser FOR LOGIN AdminUser;

EXEC sp_addrolemember 'db_owner', 'AdminUser';
```

Execution:



Comment:

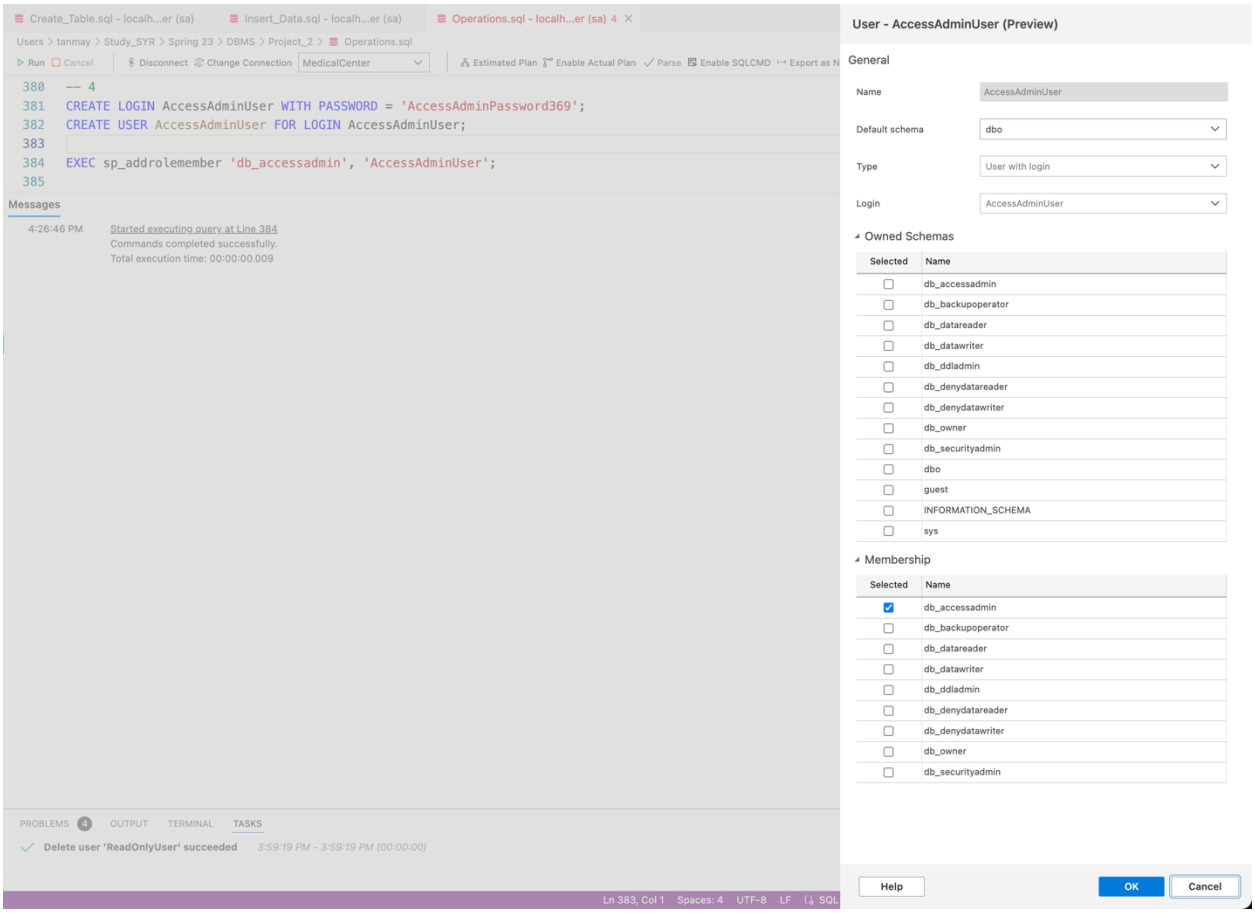
Here we can see that the new user is created who has owner membership.

Script 4:

Code:

```
CREATE LOGIN AccessAdminUser WITH PASSWORD = 'AccessAdminPassword369';
CREATE USER AccessAdminUser FOR LOGIN AccessAdminUser;

EXEC sp_addrolemember 'db_accessadmin', 'AccessAdminUser';
```

Execution:



Comment:

Here we can see that the new user is created who has access-admin membership.

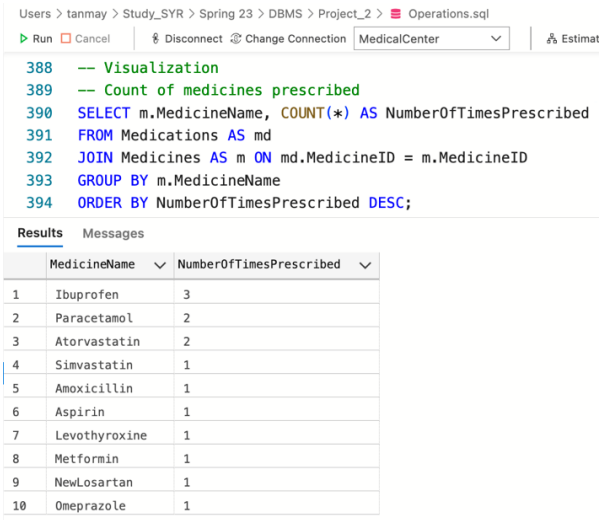
3.7 Business Report

1. Number of times a medicine was prescribed.

Code:

```
SELECT m.MedicineName, COUNT(*) AS NumberOfTimesPrescribed
FROM Medications AS md
JOIN Medicines AS m ON md.MedicineID = m.MedicineID
GROUP BY m.MedicineName
ORDER BY NumberOfTimesPrescribed DESC;
```

Data:

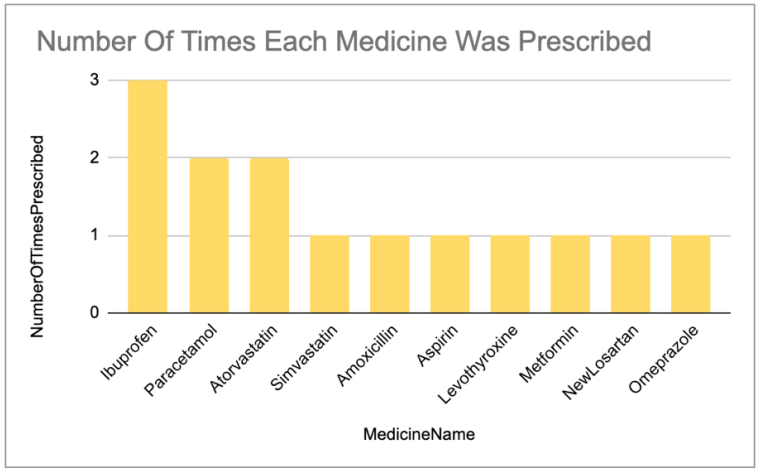


```
Users > tanmay > Study_SYR > Spring 23 > DBMS > Project_2 > Operations.sql
Run  Cancel  Disconnect  Change Connection  MedicalCenter  Estimat
388  -- Visualization
389  -- Count of medicines prescribed
390  SELECT m.MedicineName, COUNT(*) AS NumberOfTimesPrescribed
391  FROM Medications AS md
392  JOIN Medicines AS m ON md.MedicineID = m.MedicineID
393  GROUP BY m.MedicineName
394  ORDER BY NumberOfTimesPrescribed DESC;
```

Results | Messages

	MedicineName	NumberOfTimesPrescribed
1	Ibuprofen	3
2	Paracetamol	2
3	Atorvastatin	2
4	Simvastatin	1
5	Amoxicillin	1
6	Aspirin	1
7	Levothyroxine	1
8	Metformin	1
9	NewLosartan	1
10	Omeprazole	1

Graph:



Comment:

This bar graph shows the number of times each medicine in the database was prescribed to the patient. This shows us which medicine was prescribed more and helps us analyse what type of diseases do frequently occur.

2. Contribution of medicine manufacturers

Code:

```
SELECT m.Manufacturer, COUNT(*) AS Contributon
FROM Medications AS md
JOIN Medicines AS m ON md.MedicineID = m.MedicineID
GROUP BY m.Manufacturer
ORDER BY Contributon DESC;
```

Data:

Users > tanmay > Study_SYR > Spring 23 > DBMS > Project_2 > Operations.sql

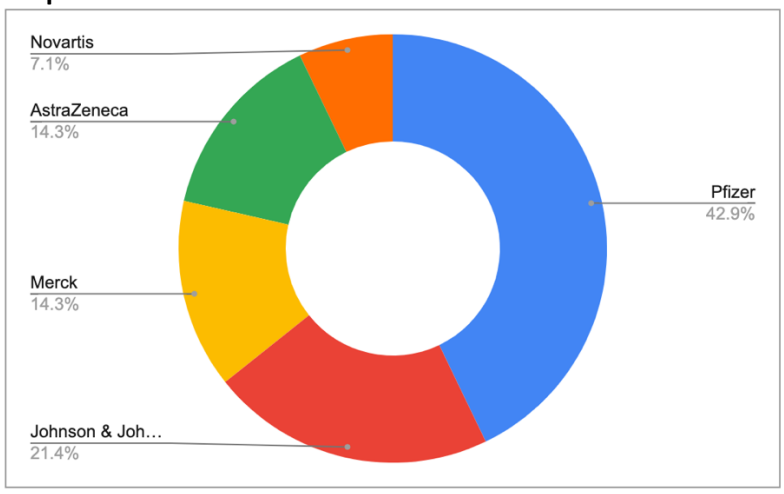
Run | Cancel | Disconnect | Change Connection | MedicalCenter

```
397  -- Contribution of manufacturers in medicine
398  SELECT m.Manufacturer, COUNT(*) AS Contributon
399  FROM Medications AS md
400  JOIN Medicines AS m ON md.MedicineID = m.MedicineID
401  GROUP BY m.Manufacturer
402  ORDER BY Contributon DESC;
403
```

Results | Messages

	Manufacturer	Contributon
1	Pfizer	6
2	Johnson & Johnson	3
3	Merck	2
4	AstraZeneca	2
5	Novartis	1

Graph:



Comment:

This donut graph shows the percentage of medicine manufacturers who have their medicines in the database.

3. Money covered by insurance companies

Code:

```
SELECT i.Company, SUM(b.InsuranceCoverage) AS TotalCoverage
FROM Billings AS b
JOIN Insurances as i ON i.InsuranceID = b.InsuranceID
GROUP BY i.Company
ORDER BY SUM(b.InsuranceCoverage) DESC;
```

Data:

Users > tanmay > Study_SYR > Spring 23 > DBMS > Project_2 > Operations.sql

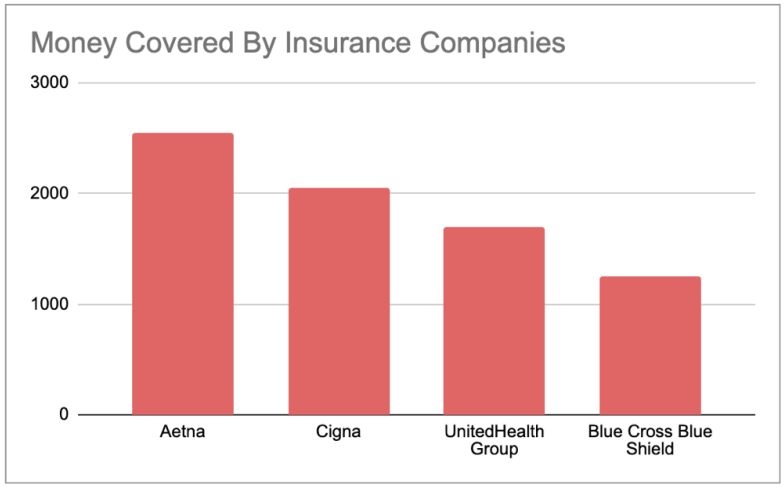
Run | Cancel | Disconnect | Change Connection | MedicalCenter | Estimate

```
405  -- Money convered by insurance companies
406  SELECT i.Company, SUM(b.InsuranceCoverage) AS TotalCoverage
407  FROM Billings AS b
408  JOIN Insurances as i ON i.InsuranceID = b.InsuranceID
409  GROUP BY i.Company
410  ORDER BY SUM(b.InsuranceCoverage) DESC;
411
```

Results | Messages

	Company	TotalCoverage
1	Aetna	2550.00
2	Cigna	2050.00
3	UnitedHealth Group	1700.00
4	Blue Cross Blue Shield	1250.00

Graph:



Comment:

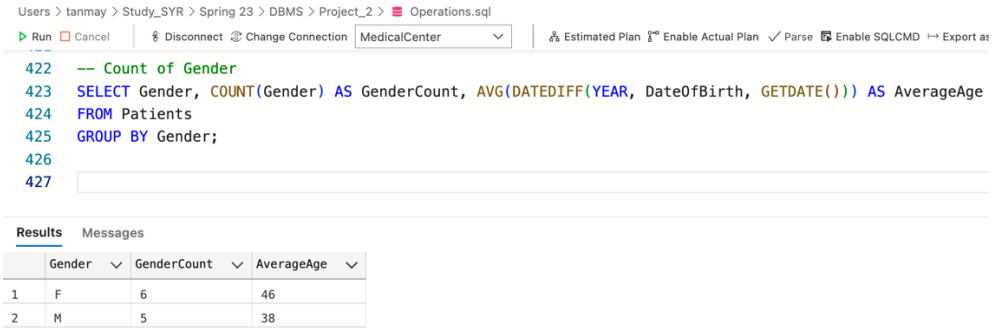
This bar graph shows the amount of money paid by each insurance company in the database.

4. Patient details

Code:

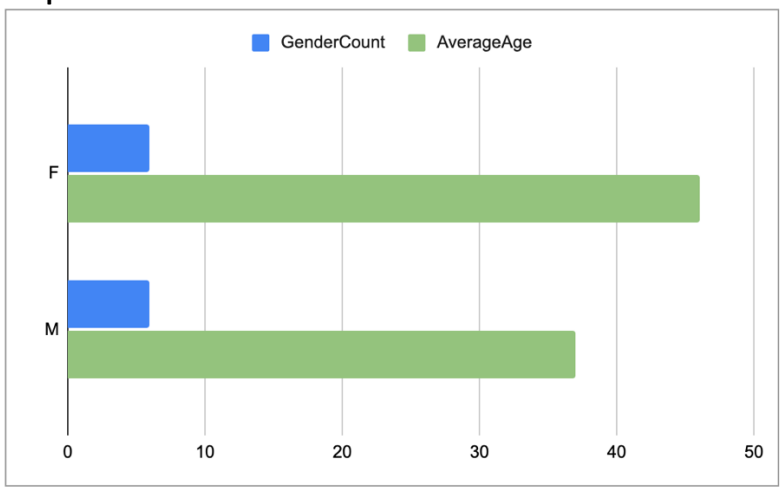
```
SELECT Gender, COUNT(Gender) AS GenderCount, AVG(DATEDIFF(YEAR, DateOfBirth,
GETDATE())) AS AverageAge
FROM Patients
GROUP BY Gender;
```

Data:



	Gender	GenderCount	AverageAge
1	F	6	46
2	M	5	38

Graph:



Comment:

This bar graph shows the count of male and female patients along with the average age of each gender.

4. Conclusion

The implemented database design provides a comprehensive and organized system for managing medical center records. It allows for the efficient storage, retrieval, and management of patient information, addresses, medical records, doctor details, and appointment records. By leveraging the relationships between the tables, medical center staff can easily access relevant information, streamline administrative tasks, and provide enhanced patient care. The database design promotes accuracy, reliability, and accessibility of data, facilitating effective decision-making and improving the overall operational efficiency of the medical center.

4.1 Remarks

From this project I learnt many things such as designing a database and learnt to write complex views, functions, triggers, stored procedures, and scripts. I also learnt to make business reports.