Sri Lanka Institute of Information Technology

Health Clinic Management System

Group 6:

IT23859838	D M T S Rathnamalala
IT23611788	H M B D Herath
IT23621138	D T Gunasekara
IT23839274	P B U R Wickramasinghe

Database Management Systems for Security - IE2042

May, 2025

Table of Contents

Part 01: Health Clinic Management System Database Creation

01.Assumptions

- The ISA relationships, ‘User’ and ‘Service’ are specialization relationships as the sub-classes are derived from the super-class.

- Both the ISA relationships are in total specialization as all the entities in the super-class are included in at least one sub-class and the ISA relationships follow the disjoint-ness constraint, as an entity only belongs to one subclass; entities cannot exist in multiple sub-classes.

- A patient can have multiple mobile numbers and email addresses, hence multi valued attributes are implemented.

- A patient can have multiple treatments, appointments and bills.

- A doctor can belong to only one department at a time.

- A surgery team consisting of Doctors and Nurses is present for each Surgery.

- A ward belongs to exactly one department.

- Appointments are scheduled by the administration staff.

- Appointments are linked to the Patient as well as the Medical staff.

- Treatments are linked to Services and may have prescriptions.

- The Patients are billed for services they take.

- Payments can be completed outright or via installments for the Bills.

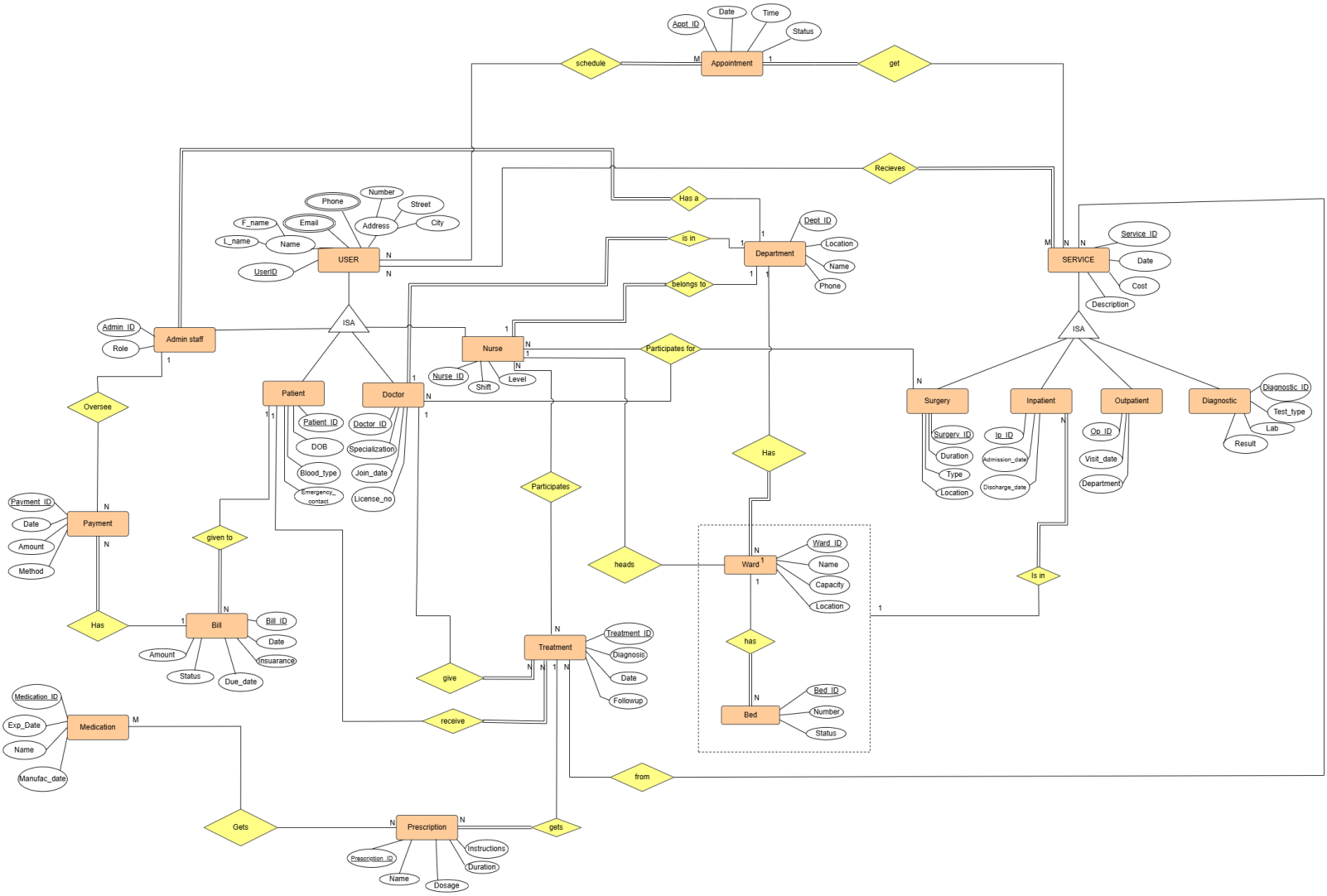
- Payments for the Bills are handled by the administrative staff.

- A head Nurse is present for each ward.

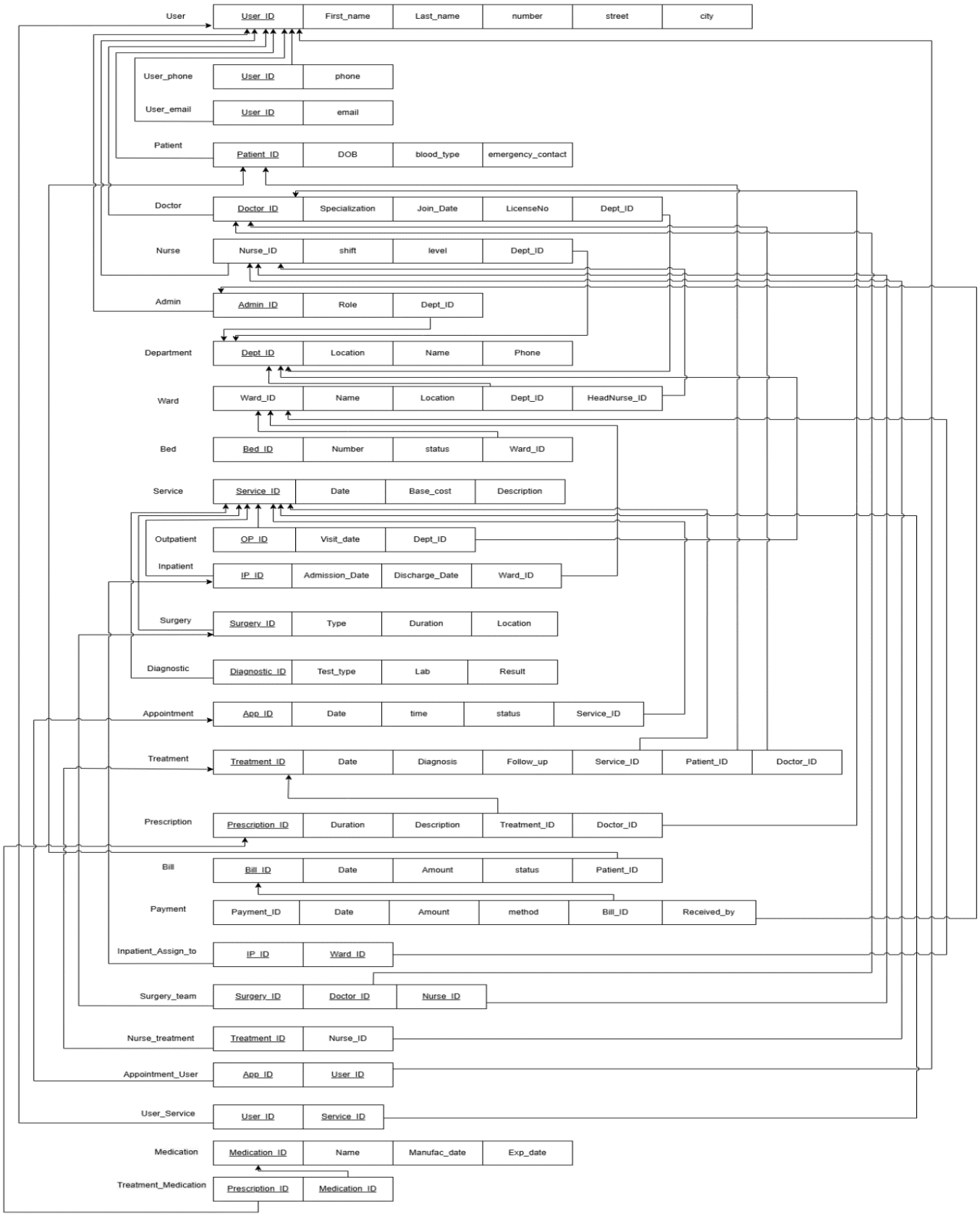
- A prescription can include multiple medications and a medication can be part of multiple prescriptions.

- Ward and Bed entities are connected through an aggregation relationship when they are associating the Inpatient entity.

02.EER diagram:



03.Relational Schema:



04.Normalization:

The above relational schema is already in the **3rd Normal Form (3NF)** as it fulfills the requirements stated below:

- **1st Normal Form (1NF)** is met by:
  - All the attributes contain atomic values.
  - There are no multivalued or composite attributes present.
  - There is a single identifiable attribute present for each table (Primary Key).

- **2nd Normal Form (2NF)** is met by:
  - The logical model satisfies 1NF.
  - All non-key attributes are fully functionally dependent on the Primary Key; there are no partial dependencies present.

- **3rd Normal Form (3NF)** is met by:
  - The logical model satisfies 2NF.
  - There are not transitive dependencies present in the relational schema.

05.Database Queries:

5.1 Table creation (DDL):

SQLQuery2.sql - E...XCALIBUR\SAM (64))* SQLQuery1.sql - E...XCALIBUR\SAM (60))*

```
CREATE DATABASE HealthClinicManagement;
GO

USE HealthClinicManagement;
GO

----- USER Table
CREATE TABLE Users (
    User_ID VARCHAR(50) NOT NULL,
    First_name VARCHAR(50) NOT NULL,
    Last_name VARCHAR(50) NOT NULL,
    number VARCHAR(20),
    street VARCHAR(50),
    city VARCHAR(50),

    CONSTRAINT User_PK PRIMARY KEY (User_ID),

);

----- User_Phone Table
CREATE TABLE User_Phone (
    User_ID VARCHAR(50) NOT NULL,
    phone VARCHAR(20) NOT NULL,

    CONSTRAINT User_phone_PK PRIMARY KEY (User_ID, phone),
    CONSTRAINT User_phone_FK FOREIGN KEY (User_ID) REFERENCES Users (User_ID),
    CONSTRAINT phone_numeric_check CHECK (phone NOT LIKE '%[0-9]*')
);
```

```
----- User_email Table
CREATE TABLE User_email (
    User_ID VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL,

    CONSTRAINT User_email_PK PRIMARY KEY (User_ID, email),
    CONSTRAINT User_email_FK FOREIGN KEY (User_ID) REFERENCES Users (User_ID),
    CONSTRAINT email_unique UNIQUE (email)
);

----- Department Table
CREATE TABLE Department (
    Dept_ID VARCHAR(50) NOT NULL,
    Location VARCHAR(100) NOT NULL,
    Name VARCHAR(100) NOT NULL,
    Phone VARCHAR(20),

    CONSTRAINT Department_PK PRIMARY KEY (Dept_ID),
    CONSTRAINT Department_phone_unique UNIQUE (Phone)
);

----- Patient Table
CREATE TABLE Patient (
    Patient_ID VARCHAR(50) NOT NULL,
    DOB DATE NOT NULL,
    blood_type VARCHAR(3),
    emergency_contact VARCHAR(100),

    CONSTRAINT Patient_PK PRIMARY KEY (Patient_ID),
    CONSTRAINT Patient_User_FK FOREIGN KEY (Patient_ID) REFERENCES Users (User_ID),
    CONSTRAINT Blood_type_check CHECK (blood_type IN ('A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-'))
);

----- Doctor Table
CREATE TABLE Doctor (
    Doctor_ID VARCHAR(50) NOT NULL,
    Specialization VARCHAR(100) NOT NULL,
    Join_Date DATE NOT NULL,
    LicenseNo VARCHAR(50) NOT NULL,
    Dept_ID VARCHAR(50) NOT NULL,

    CONSTRAINT Doctor_PK PRIMARY KEY (Doctor_ID),
    CONSTRAINT Doctor_User_FK FOREIGN KEY (Doctor_ID) REFERENCES Users (User_ID),
    CONSTRAINT Doctor_department_FK FOREIGN KEY (Dept_ID) REFERENCES Department (Dept_ID),
    CONSTRAINT LicenseNo_unique UNIQUE (LicenseNo)
);
```

```sql
----- Nurse Table
CREATE TABLE Nurse (
    Nurse_ID VARCHAR(50) NOT NULL,
    shift VARCHAR(20) NOT NULL,
    level VARCHAR(50) NOT NULL,
    Dept_ID VARCHAR(50) NOT NULL,

    CONSTRAINT Nurse_PK PRIMARY KEY (Nurse_ID),
    CONSTRAINT Nurse_User_FK FOREIGN KEY (Nurse_ID) REFERENCES Users (User_ID),
    CONSTRAINT Nurse_department_FK FOREIGN KEY (Dept_ID) REFERENCES Department (Dept_ID),
    CONSTRAINT Shift_check CHECK (shift IN ('Morning', 'Evening', 'Night'))
);

----- Admin Table
CREATE TABLE Admin (
    Admin_ID VARCHAR(50) NOT NULL,
    Role VARCHAR(50) NOT NULL,
    Dept_ID VARCHAR(50) NOT NULL,

    CONSTRAINT Admin_PK PRIMARY KEY (Admin_ID),
    CONSTRAINT Admin_User_FK FOREIGN KEY (Admin_ID) REFERENCES Users (User_ID),
    CONSTRAINT Admin_department_FK FOREIGN KEY (Dept_ID) REFERENCES Department (Dept_ID)
);

----- Ward Table
CREATE TABLE Ward (
    Ward_ID VARCHAR(50) NOT NULL,
    Name VARCHAR(100) NOT NULL,
    Location VARCHAR(100) NOT NULL,
    Dept_ID VARCHAR(50) NOT NULL,
    HeadNurse_ID VARCHAR(50),

    CONSTRAINT Ward_PK PRIMARY KEY (Ward_ID),
    CONSTRAINT Ward_department_FK FOREIGN KEY (Dept_ID) REFERENCES Department (Dept_ID),
    CONSTRAINT Ward_Nurse_FK FOREIGN KEY (HeadNurse_ID) REFERENCES Nurse (Nurse_ID)
);

----- Bed Table
CREATE TABLE Bed (
    Bed_ID VARCHAR(50),
    Number INT NOT NULL,
    Status VARCHAR(20) NOT NULL  DEFAULT 'Available',
    Ward_ID VARCHAR(50),

    CONSTRAINT Bed_PK PRIMARY KEY (Bed_ID),
    CONSTRAINT Ward_Bed_FK FOREIGN KEY (Ward_ID) REFERENCES Ward (Ward_ID),
    CONSTRAINT Bed_Status_CHECK CHECK (Status IN ('Available', 'Occupied', 'Maintenance'))
);
```

```sql
----- Service Table
CREATE TABLE Service (
    Service_ID VARCHAR(50),
    Date DATE NOT NULL,
    Base_cost DECIMAL(10,2) NOT NULL,
    Description VARCHAR(255),
    S_type VARCHAR(50) NOT NULL,

    CONSTRAINT Service_PK PRIMARY KEY (Service_ID),
    CONSTRAINT Service_Cost_CHECK CHECK (Base_cost >= 0)
);

----- Outpatient Table
CREATE TABLE Outpatient (
    OP_ID VARCHAR(50),
    Visit_date DATE NOT NULL,
    Dept_ID VARCHAR(50),
    Service_ID VARCHAR(50),

    CONSTRAINT Outpatient_PK PRIMARY KEY (OP_ID),
    CONSTRAINT Outpatient_Service_FK FOREIGN KEY (Service_ID) REFERENCES Service(Service_ID),
    CONSTRAINT Outpatient_Dept_FK FOREIGN KEY (Dept_ID) REFERENCES Department(Dept_ID)
);

----- Inpatient Table
CREATE TABLE Inpatient (
    IP_ID VARCHAR(50),
    Admission_Date DATE NOT NULL,
    Discharge_Date DATE,
    Ward_ID VARCHAR(50),
    Service_ID VARCHAR(50),

    CONSTRAINT Inpatient_PK PRIMARY KEY (IP_ID),
    CONSTRAINT Inpatient_Service_FK FOREIGN KEY (Service_ID) REFERENCES Service(Service_ID),
    CONSTRAINT Inpatient_Ward_FK FOREIGN KEY (Ward_ID) REFERENCES Ward(Ward_ID)
);

----- Surgery Table
CREATE TABLE Surgery (
    Surgery_ID VARCHAR(50),
    Type VARCHAR(100) NOT NULL,
    Duration TIME NOT NULL,
    Location VARCHAR(100),
    Service_ID VARCHAR(50),

    CONSTRAINT Surgery_PK PRIMARY KEY (Surgery_ID),
    CONSTRAINT Surgery_Service_FK FOREIGN KEY (Service_ID) REFERENCES Service(Service_ID)
);
```

```sql
----- Diagnostic Table
CREATE TABLE Diagnostic (
    Diagnostic_ID VARCHAR(50),
    Test_type VARCHAR(100) NOT NULL,
    Lab VARCHAR(100),
    Result VARCHAR(255),
    Service_ID VARCHAR(50),

    CONSTRAINT Diagnostic_PK PRIMARY KEY (Diagnostic_ID),
    CONSTRAINT Diagnostic_Service_FK FOREIGN KEY (Service_ID) REFERENCES Service(Service_ID)
);

----- Appointment Table
CREATE TABLE Appointment (
    App_ID VARCHAR(50),
    Date DATE NOT NULL,
    Time TIME NOT NULL,
    Status VARCHAR(50) NOT NULL DEFAULT 'Scheduled',
    Service_ID VARCHAR(50),

    CONSTRAINT Appointment_PK PRIMARY KEY (App_ID),
    CONSTRAINT Appointment_Status_CHECK CHECK (Status IN ('Scheduled', 'Completed', 'Cancelled')),
    CONSTRAINT Appointment_Service_FK FOREIGN KEY (Service_ID) REFERENCES Service(Service_ID)
);

----- Treatment Table
CREATE TABLE Treatment (
    Treatment_ID VARCHAR(50),
    Date DATE NOT NULL,
    Diagnosis VARCHAR(255),
    Follow_up VARCHAR(255),
    Service_ID VARCHAR(50),
    Patient_ID VARCHAR(50),
    Doctor_ID VARCHAR(50),

    CONSTRAINT Treatment_PK PRIMARY KEY (Treatment_ID),
    CONSTRAINT Treatment_Service_FK FOREIGN KEY (Service_ID) REFERENCES Service(Service_ID),
    CONSTRAINT Treatment_Patient_FK FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID),
    CONSTRAINT Treatment_Doctor_FK FOREIGN KEY (Doctor_ID) REFERENCES Doctor(Doctor_ID)
);

----- Prescription Table
CREATE TABLE Prescription (
    Prescription_ID VARCHAR(50),
    Duration VARCHAR(100),
    Description VARCHAR(255),
    Treatment_ID VARCHAR(50),
    Doctor_ID VARCHAR(50),
```

```sql
----- Bill Table
CREATE TABLE Bill (
    Bill_ID VARCHAR(50),
    Date DATE NOT NULL,
    Amount DECIMAL(10,2) NOT NULL,
    Status VARCHAR(50) NOT NULL  DEFAULT 'Pending',
    Patient_ID VARCHAR(50),

    CONSTRAINT Bill_PK PRIMARY KEY (Bill_ID),
    CONSTRAINT Bill_Status_CHECK CHECK (Status IN ('Pending', 'Paid', 'Cancelled')),
    CONSTRAINT Bill_Patient_FK FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID)
);

----- Payment Table
CREATE TABLE Payment (
    Payment_ID VARCHAR(50),
    Date DATE NOT NULL,
    Amount DECIMAL(10,2) NOT NULL,
    Method VARCHAR(50) NOT NULL,
    Bill_ID VARCHAR(50),
    Received_by VARCHAR(50),

    CONSTRAINT Payment_PK PRIMARY KEY (Payment_ID),
    CONSTRAINT Payment_Method_CHECK CHECK (Method IN ('Cash', 'Card', 'Transfer')),
    CONSTRAINT Payment_Bill_FK FOREIGN KEY (Bill_ID) REFERENCES Bill(Bill_ID),
    CONSTRAINT Payment_Receiver_FK FOREIGN KEY (Received_by) REFERENCES Admin(Admin_ID)
);

----- Inpatient_Assign_to Table
CREATE TABLE Inpatient_Assign_to (
    IP_ID VARCHAR(50),
    Ward_ID VARCHAR(50),

    CONSTRAINT Inpatient_Assign_PK PRIMARY KEY (IP_ID, Ward_ID),
    CONSTRAINT Inpatient_Assign_IP_FK FOREIGN KEY (IP_ID) REFERENCES Inpatient(IP_ID),
    CONSTRAINT Inpatient_Assign_Ward_FK FOREIGN KEY (Ward_ID) REFERENCES Ward(Ward_ID)
);

----- Surgery_Team Table
CREATE TABLE Surgery_Team (
    Surgery_ID VARCHAR(50),
    Doctor_ID VARCHAR(50),
    Nurse_ID VARCHAR(50),

    CONSTRAINT Surgery_Team_PK PRIMARY KEY (Surgery_ID, Doctor_ID, Nurse_ID),
    CONSTRAINT Surgery_Team_Surgery_FK FOREIGN KEY (Surgery_ID) REFERENCES Surgery(Surgery_ID),
    CONSTRAINT Surgery_Team_Doctor_FK FOREIGN KEY (Doctor_ID) REFERENCES Doctor(Doctor_ID),
    CONSTRAINT Surgery_Team_Nurse_FK FOREIGN KEY (Nurse_ID) REFERENCES Nurse(Nurse_ID)
);
```

```sql
----- Nurse_Treatment Table
CREATE TABLE Nurse_Treatment (
    Treatment_ID VARCHAR(50),
    Nurse_ID VARCHAR(50),

    CONSTRAINT Nurse_Treatment_PK PRIMARY KEY (Treatment_ID, Nurse_ID),
    CONSTRAINT Nurse_Treatment_Treatment_FK FOREIGN KEY (Treatment_ID) REFERENCES Treatment(Treatment_ID),
    CONSTRAINT Nurse_Treatment_Nurse_FK FOREIGN KEY (Nurse_ID) REFERENCES Nurse(Nurse_ID)
);

----- Appointment_User Table
CREATE TABLE Appointment_User (
    App_ID VARCHAR(50),
    User_ID VARCHAR(50),

    CONSTRAINT Appointment_User_PK PRIMARY KEY (App_ID, User_ID),
    CONSTRAINT Appointment_User_App_FK FOREIGN KEY (App_ID) REFERENCES Appointment(App_ID),
    CONSTRAINT Appointment_User_User_FK FOREIGN KEY (User_ID) REFERENCES Users(User_ID)
);

----- User_Service Table
CREATE TABLE User_Service (
    User_ID VARCHAR(50),
    Service_ID VARCHAR(50),

    CONSTRAINT User_Service_PK PRIMARY KEY (User_ID, Service_ID),
    CONSTRAINT User_Service_User_FK FOREIGN KEY (User_ID) REFERENCES Users(User_ID),
    CONSTRAINT User_Service_Service_FK FOREIGN KEY (Service_ID) REFERENCES Service(Service_ID)
);

----- Medication Table
CREATE TABLE Medication (
    Medication_ID VARCHAR(50),
    Name VARCHAR(100) NOT NULL,
    Manufacture_date DATE NOT NULL,
    Exp_date DATE NOT NULL,

    CONSTRAINT Medication_PK PRIMARY KEY (Medication_ID),
    CONSTRAINT Medication_Expiry_CHECK CHECK (Exp_date > Manufacture_date)
);

----- Treatment_Medication Table
CREATE TABLE Treatment_Medication (
    Prescription_ID VARCHAR(50),
    Medication_ID VARCHAR(50),

    CONSTRAINT Treatment_Medication_PK PRIMARY KEY (Prescription_ID, Medication_ID),
    CONSTRAINT Treatment_Medication_Prescription_FK FOREIGN KEY (Prescription_ID) REFERENCES Prescription(Prescription_ID),
    CONSTRAINT Treatment_Medication_Medication_FK FOREIGN KEY (Medication_ID) REFERENCES Medication(Medication_ID)
);
```

5.2 Data Insertion (DML):

```sql
-- User Table
INSERT INTO Users(User_ID, First_name, Last_name, number, street, city) VALUES
('U001', 'John', 'Doe', '123', 'Main St', 'Colombo'),
('U002', 'Emily', 'Clark', '456', 'Lake Rd', 'Kandy'),
('U003', 'Michael', 'Smith', '789', 'Hill St', 'Galle'),
('U004', 'Sophia', 'Perera', '321', 'Park Ave', 'Colombo'),
('U005', 'Daniel', 'Fernando', '654', 'Ocean Rd', 'Negombo'),
('U006', 'Olivia', 'Silva', '987', 'River St', 'Jaffna'),
('U007', 'Liam', 'Jayasinghe', '111', 'Garden Ln', 'Colombo'),
('U008', 'Isla', 'Kumar', '222', 'City Blvd', 'Matara'),
('U009', 'Ethan', 'Dias', '9', 'Liberty St', 'Colombo'),
('U010', 'Amaya', 'Wickramasinghe', '10', 'Hill Crest', 'Kandy'),
('U011', 'Jason', 'De Silva', '11', 'Garden Way', 'Galle'),
('U012', 'Nadeesha', 'Fernando', '12', 'Ocean Ave', 'Negombo'),
('U013', 'Roshan', 'Perera', '13', 'River Rd', 'Colombo'),
('U014', 'Aanya', 'Senanayake', '14', 'Lake View', 'Jaffna'),
('U015', 'Dulina', 'Piyasiri', '1' ,'Cinnamon hill', 'Colombo'),
('U016', 'Ashen', 'Adikari', '19', 'Crest Blvd', 'Gampaha');

-- User_Phone Table
INSERT INTO User_Phone (User_ID, phone) VALUES
('U001', '0711234567'),
('U002', '0722345676'),
('U003', '0773456784'),
('U004', '0759876546'),
('U005', '0765432123'),
('U006', '0788765437'),
('U007', '0707654329'),
('U008', '0796543217'),
('U009', '0719876549'),
('U010', '0727654320'),
('U011', '0778765435'),
('U012', '0789988773'),
('U013', '0751122332'),
('U014', '0795566778'),
('U015', '0734576345'),
('U016', '0796854640');
```

```sql
-- User_email Table
INSERT INTO User_email (User_ID, email) VALUES
('U001', 'john.doe@gmail.com'),
('U002', 'emily.clark@gmail.com'),
('U003', 'michael.smith@gmail.com'),
('U004', 'sophia.perera@gmail.com'),
('U005', 'daniel.fernando@gmail.com'),
('U006', 'olivia.silva@gmail.com'),
('U007', 'liam.jayasinghe@gmail.com'),
('U008', 'isla.kumar@gmail.com'),
('U009', 'ethan.dias@gmail.com'),
('U010', 'amaya.w@gmail.com'),
('U011', 'jason.desilva@gmail.com'),
('U012', 'nadeesha.f@gmail.com'),
('U013', 'roshan.p@gmail.com'),
('U014', 'aanya.s@gmail.com'),
('U015', 'dulina.p@gmail.com'),
('U016', 'ashen.a@gmail.com');

-- Department Table
INSERT INTO Department (Dept_ID, Location, Name, Phone) VALUES
('D001', 'Building A', 'Cardiology', '0112233445'),
('D002', 'Building B', 'Neurology', '0113344556'),
('D003', 'Building C', 'Pediatrics', '0114455667'),
('D004', 'Building D', 'Oncology', '0214455667');

-- Patient Table
INSERT INTO Patient (Patient_ID, DOB, blood_type, emergency_contact) VALUES
('U001', '1990-06-15', 'A+', '0751112233'),
('U002', '1985-12-20', 'B-', '0723334455'),
('U009', '1992-08-10', 'O+', '0712223344'),
('U010', '1999-05-20', 'AB-', '0775556677');

-- Doctor Table
INSERT INTO Doctor (Doctor_ID, Specialization, Join_Date, LicenseNo, Dept_ID) VALUES
('U003', 'Cardiologist', '2020-03-01', 'LIC1001', 'D001'),
('U004', 'Neurologist', '2019-07-12', 'LIC1002', 'D002'),
('U011', 'Pediatrician', '2018-01-05', 'LIC1003', 'D003'),
('U012', 'Oncologist', '2021-09-15', 'LIC1004', 'D004');

-- Nurse Table
INSERT INTO Nurse (Nurse_ID, shift, level, Dept_ID) VALUES
('U005', 'Morning', 'Senior', 'D001'),
('U006', 'Night', 'Junior', 'D002'),
('U013', 'Evening', 'Senior', 'D002'),
('U014', 'Morning', 'Junior', 'D003');
```

```sql
-- Admin Table
INSERT INTO Admin (Admin_ID, Role, Dept_ID) VALUES
('U007', 'System Administrator', 'D003'),
('U008', 'HR Manager', 'D001'),
('U015', 'System Administrator', 'D002'),
('U016', 'HR Manager', 'D002');

-- Ward Table
INSERT INTO Ward (Ward_ID, Name, Location, Dept_ID, HeadNurse_ID) VALUES
('W001', 'General Ward', 'First Floor', 'D001', 'U005'),
('W002', 'ICU', 'Ground Floor', 'D002', 'U006'),
('W003', 'Cardio ICU', '3rd Floor', 'D001', 'U013'),
('W004', 'General Ward', '2nd Floor', 'D003', 'U014');

-- Bed Table
INSERT INTO Bed (Bed_ID, Number, Status, Ward_ID) VALUES
('B001', 101, 'Available', 'W001'),
('B002', 102, 'Occupied', 'W001'),
('B003', 103, 'Maintenance', 'W001'),
('B004', 104, 'Available', 'W002'),
('B005', 105, 'Occupied', 'W002'),
('B006', 106, 'Available', 'W003'),
('B007', 107, 'Available', 'W003'),
('B008', 108, 'Occupied', 'W003');


--Service Table
INSERT INTO Service (Service_ID, Date, Base_cost, Description) VALUES
('S001', '2023-05-01', 1500.00, 'Routine checkup'),
('S002', '2023-05-02', 5000.00, 'MRI scan'),
('S003', '2023-05-03', 1200.00, 'Appendectomy'),
('S004', '2023-05-04', 3000.00, 'Blood test panel'),
('S005', '2023-05-05', 2000.00, 'Physical therapy session'),
('S006', '2023-05-06', 2500.00, 'Follow-up consultation'),
('S007', '2023-05-07', 8000.00, 'Knee replacement surgery'),
('S008', '2023-05-08', 3500.00, 'Echocardiogram'),
('S009', '2023-05-09', 1800.00, 'Vaccination'),
('S010', '2023-05-10', 4500.00, 'Colonoscopy'),
('S011', '2023-05-11', 5500.00, 'CAT Scan'),
('S012', '2023-05-21', 6500.00, 'Hip Surgery'),
('S013', '2023-05-19', 4300.00, 'LASIK Surgery'),
('S014', '2023-05-23', 3500.00, 'Echocardiogram'),
('S015', '2023-05-26', 4500.00, 'Colonoscopy');
```

```sql
-- Outpatient Table
INSERT INTO Outpatient (OP_ID, Visit_date, Dept_ID, Service_ID) VALUES
('OP001', '2023-05-01', 'D001', 'S001'),
('OP002', '2023-05-04', 'D002', 'S004'),
('OP003', '2023-05-06', 'D003', 'S006'),
('OP004', '2023-05-08', 'D001', 'S008'),
('OP005', '2023-05-03', 'D001', 'S005');

--Inpatient Table
INSERT INTO Inpatient (IP_ID, Admission_Date, Discharge_Date, Ward_ID, Service_ID) VALUES
('IP001', '2023-05-03', '2023-05-07', 'W001', 'S003'),
('IP002', '2023-05-07', NULL, 'W003', 'S007'),
('IP003', '2023-05-10', NULL, 'W002', 'S010'),
('IP004', '2023-05-11', '2023-05-11', 'W002', 'S011');

--Surgery Table
INSERT INTO Surgery (Surgery_ID, Type, Duration, Location, Service_ID) VALUES
('SU001', 'Hip replacement', '02:30:00', 'OR 1', 'S012'),
('SU002', 'LASIK surgey', '03:45:00', 'OR 2', 'S013');

--Diagnostic Table
INSERT INTO Diagnostic (Diagnostic_ID, Test_type, Lab, Result, Service_ID) VALUES
('DI001', 'MRI', 'Main Radiology Lab', 'No abnormalities detected', 'S002'),
('DI002', 'Blood Test', 'Central Lab', 'Elevated white blood cells', 'S009'),
('DI003', 'Echocardiogram', 'Cardiology Lab', 'Normal heart function', 'S014'),
('DI004', 'Colonoscopy', 'Gastro Lab', 'Small polyp detected', 'S015');

--Appointment Table
INSERT INTO Appointment (App_ID, Date, Time, Status, Service_ID) VALUES
('AP001', '2023-05-01', '09:00:00', 'Completed', 'S001'),
('AP002', '2023-05-02', '10:30:00', 'Completed', 'S002'),
('AP003', '2023-05-06', '14:00:00', 'Completed', 'S006'),
('AP004', '2023-05-11', '11:00:00', 'Scheduled', NULL),
('AP005', '2023-05-12', '15:30:00', 'Scheduled', NULL),
('AP006', '2023-05-13', '15:00:00', 'Scheduled', NULL);

--Treatment Table
INSERT INTO Treatment (Treatment_ID, Date, Diagnosis, Follow_up, Service_ID, Patient_ID, Doctor_ID) VALUES
('T001', '2023-05-01', 'Hypertension', 'Follow up in 3 months', 'S001', 'U001', 'U003'),
('T002', '2023-05-02', 'Migraine', 'Follow up if symptoms persist', 'S002', 'U002', 'U012'),
('T003', '2023-05-03', 'Appendicitis', 'Post-surgery check in 2 weeks', 'S003', 'U010', 'U011'),
('T004', '2023-05-04', 'Viral infection', 'Rest and fluids', 'S004', 'U009', 'U004'),
('T005', '2023-05-07', 'Osteoarthritis', 'Physical therapy sessions', 'S007', 'U001', 'U003');
```

```sql
--Prescription Table
INSERT INTO Prescription (Prescription_ID, Duration, Description, Treatment_ID, Doctor_ID) VALUES
('P001', '30 days', 'Lisinopril 10mg daily', 'T001', 'U004'),
('P002', '7 days', 'Sumatriptan 50mg as needed', 'T002', 'U011'),
('P003', '10 days', 'Amoxicillin 500mg every 8 hours', 'T003', 'U012'),
('P004', '5 days', 'Ibuprofen 400mg every 6 hours', 'T004', 'U003'),
('P005', '14 days', 'Naproxen 500mg twice daily', 'T005', 'U004');

--Bill Table
INSERT INTO Bill (Bill_ID, Date, Amount, Status, Patient_ID) VALUES
('BL001', '2023-05-02', 1500.00, 'Paid', 'U001'),
('BL002', '2023-05-03', 5000.00, 'Paid', 'U002'),
('BL003', '2023-05-04', 12000.00, 'Pending', 'U001'),
('BL004', '2023-05-05', 3000.00, 'Paid', 'U009'),
('BL005', '2023-05-08', 8000.00, 'Pending', 'U010');

--Payment Table
INSERT INTO Payment (Payment_ID, Date, Amount, Method, Bill_ID, Received_by) VALUES
('PY001', '2023-05-02', 1000.00, 'Card', 'BL001', 'U007'),
('PY002', '2023-05-03', 2500.00, 'Transfer', 'BL002', 'U007'),
('PY003', '2023-05-05', 1500.00, 'Cash', 'BL004', 'U015');

--Inpatient_Assigned_to Table
INSERT INTO Inpatient_Assign_to (IP_ID, Ward_ID) VALUES
('IP001', 'W001'),
('IP002', 'W003'),
('IP003', 'W002');

--Surgey Team Table
INSERT INTO Surgery_Team (Surgery_ID, Doctor_ID, Nurse_ID) VALUES
('SU001', 'U003', 'U014'),
('SU001', 'U004', 'U013'),
('SU002', 'U011', 'U006'),
('SU002', 'U012', 'U005');

--Nurse Treatment Table
INSERT INTO Nurse_Treatment (Treatment_ID, Nurse_ID) VALUES
('T001', 'U013'),
('T002', 'U014'),
('T003', 'U006'),
('T004', 'U005');
```

```sql
--Appointment User Table
INSERT INTO Appointment_User (App_ID, User_ID) VALUES
('AP001', 'U001'),
('AP001', 'U002'),
('AP003', 'U009'),
('AP004', 'U010'),
('AP005', 'U002');

--User Service table
INSERT INTO User_Service (User_ID, Service_ID) VALUES
('U001', 'S001'),
('U002', 'S005'),
('U003', 'S003'),
('U004', 'S008'),
('U005', 'S007'),
('U006', 'S001'),
('U007', 'S006'),
('U008', 'S004'),
('U009', 'S010'),
('U010', 'S011');

--Medication Table
INSERT INTO Medication (Medication_ID, Name, Manufacture_date, Exp_date) VALUES
('M001', 'Lisinopril', '2022-01-15', '2024-01-15'),
('M002', 'Sumatriptan', '2022-03-20', '2023-09-20'),
('M003', 'Amoxicillin', '2023-01-10', '2024-01-10'),
('M004', 'Ibuprofen', '2022-11-05', '2024-05-05'),
('M005', 'Naproxen', '2022-09-12', '2024-03-12');

--Medication Treatment Table
INSERT INTO Treatment_Medication (Prescription_ID, Medication_ID) VALUES
('P001', 'M001'),
('P002', 'M004'),
('P003', 'M002'),
('P004', 'M003'),
('P001', 'M005'),
('P005', 'M002'),
('P005', 'M003'),
('P002', 'M001');
```

06. Triggers:

6.1 BEFORE Trigger:

The BEFORE trigger validates the patient age before patient data is entered to the database. The trigger checks if the patient is older than 18 years, and if not an error message is shown.

```
CREATE TRIGGER tr_ValidatePatientAge
ON Patient
INSTEAD OF INSERT
AS
BEGIN
    -- Checks if patient is at least 18 years old
    IF EXISTS (
        SELECT 1 FROM inserted
        WHERE DATEDIFF(YEAR, DOB, GETDATE()) < 18
    )
    BEGIN
        RAISERROR('Patient must be at least 18 years old', 16, 1)
        ROLLBACK TRANSACTION
    END
    ELSE
    BEGIN
        -- Proceed with the insert if validation is correct
        INSERT INTO Patient (Patient_ID, DOB, blood_type, emergency_contact)
        SELECT Patient_ID, DOB, blood_type, emergency_contact FROM inserted
    END
END;
```

6.2 AFTER Trigger:

The AFTER trigger tracks appointment changes. If there are any changes to be implemented, the changes are logged into a new table ‘Appointment_Audit’. This table can be used to observe who changed what and when.

```
CREATE TABLE Appointment_Audit (
    Audit_ID INT IDENTITY(1,1) PRIMARY KEY,
    App_ID VARCHAR(50),
    Old_Status VARCHAR(50),
    New_Status VARCHAR(50),
    Changed_By VARCHAR(50),
    Change_Date DATETIME DEFAULT GETDATE()
);

CREATE TRIGGER tr_LogAppointmentStatusChange
ON Appointment
AFTER UPDATE
AS
BEGIN
    -- Only log if status changed
    IF UPDATE(Status)
    BEGIN
        INSERT INTO Appointment_Audit (App_ID, Old_Status, New_Status, Changed_By)
        SELECT
            i.App_ID,
            d.Status,
            i.Status,
            SYSTEM_USER
        FROM inserted i
        JOIN deleted d ON i.App_ID = d.App_ID
        WHERE i.Status <> d.Status
    END
END;
```

07.Views:

7.1 View 01 (Doctors view):

The Doctor view shows a doctors their patients' appointments, treatments, prescriptions and other related data.

```
CREATE VIEW vw_DoctorPatientDetails AS
SELECT
    d.Doctor_ID,
    u.First_name + ' ' + u.Last_name AS Doctor_Name,
    p.Patient_ID,
    pu.First_name + ' ' + pu.Last_name AS Patient_Name,
    a.App_ID,
    a.Date AS Appointment_Date,
    a.Time AS Appointment_Time,
    a.Status AS Appointment_Status,
    t.Treatment_ID,
    t.Diagnosis,
    t.Follow_up,
    pr.Prescription_ID,
    pr.Description AS Prescription_Details
FROM Doctor d
JOIN Users u ON d.Doctor_ID = u.User_ID
JOIN Treatment t ON d.Doctor_ID = t.Doctor_ID
JOIN Patient p ON t.Patient_ID = p.Patient_ID
JOIN Users pu ON p.Patient_ID = pu.User_ID
LEFT JOIN Appointment a ON t.Service_ID = a.Service_ID
LEFT JOIN Prescription pr ON t.Treatment_ID = pr.Treatment_ID;
```

7.2 View 02 (Admin Staff view):

The Admin staff view provides the employees information on appointment schedules and other admission details along with patient, doctor, department and ward information.

```
CREATE VIEW vw_AdminAppointmentDetails AS
SELECT
    a.App_ID,
    u.First_name + ' ' + u.Last_name AS Patient_Name,
    a.Date AS Appointment_Date,
    a.Time AS Appointment_Time,
    a.Status AS Appointment_Status,
    d.Name AS Department_Name,
    doc.Doctor_ID AS Doctor_ID,
    doc.LicenseNo AS Doctor_License,
    ip.Admission_Date,
    ip.Discharge_Date,
    w.Name AS Ward_Name
FROM Appointment a
JOIN Appointment_User au ON a.App_ID = au.App_ID
JOIN Users u ON au.User_ID = u.User_ID
LEFT JOIN Service s ON a.Service_ID = s.Service_ID
LEFT JOIN Treatment t ON s.Service_ID = t.Service_ID
LEFT JOIN Doctor doc ON t.Doctor_ID = doc.Doctor_ID
LEFT JOIN Department d ON doc.Dept_ID = d.Dept_ID
LEFT JOIN Inpatient ip ON s.Service_ID = ip.Service_ID
LEFT JOIN Ward w ON ip.Ward_ID = w.Ward_ID
WHERE au.User_ID IN (SELECT Patient_ID FROM Patient);
```

08.Indexes:

8.1 Index for Query 01:

This index optimizes searches for a patients appointments in a given period of time.

```
CREATE INDEX idx_Appointment_Patient_Date_Status
ON Appointment_User(User_ID)
INCLUDE (App_ID);
```

8.2 Index for Query 02:

This index optimizes billing information retrieval of patients and is equipped with frequently accessed columns.

```
CREATE INDEX idx_Bill_Patient
ON Bill(Patient_ID)
INCLUDE (Date, Amount, Status);
```

09.Procedures:

9.1 Procedure for Patient Appointments:

This stored procedure retrieves all appointments of a patient within a given period of time.

```
CREATE PROCEDURE sp_GetPatientAppointments
    @PatientID VARCHAR(50),
    @StartDate DATE,
    @EndDate DATE
AS
BEGIN
    SELECT
        a.App_ID,
        a.Date,
        a.Time,
        a.Status,
        d.Name AS Department,
        doc.Dept_ID AS Doctor_ID,
        s.Description AS Service_Description
    FROM Appointment a
    JOIN Appointment_User au ON a.App_ID = au.App_ID
    LEFT JOIN Service s ON a.Service_ID = s.Service_ID
    LEFT JOIN Treatment t ON s.Service_ID = t.Service_ID
    LEFT JOIN Doctor doc ON t.Doctor_ID = doc.Doctor_ID
    LEFT JOIN Department d ON doc.Dept_ID = d.Dept_ID
    WHERE au.User_ID = @PatientID
    AND a.Date BETWEEN @StartDate AND @EndDate
    ORDER BY a.Date, a.Time;
END;


EXEC sp_GetPatientAppointments
    @PatientID = 'U001',
    @StartDate = '2020-05-01',
    @EndDate = '2025-05-31';
```

9.2 Procedure for Nurse Appointments:

This stored procedure shows all appointments assigned to a specific Nurse that has patient and treatment details.

```
CREATE PROCEDURE sp_GetNurseAppointments
@NurseID VARCHAR(50)
AS
BEGIN
    SELECT
        a.App_ID,
        a.Date,
        a.Time,
        a.Status,
        u.First_name + ' ' + u.Last_name AS Patient_Name,
        t.Diagnosis,
        nt.Treatment_ID
    FROM Appointment a
    JOIN Appointment_User au ON a.App_ID = au.App_ID
    JOIN Users u ON au.User_ID = u.User_ID
    JOIN Treatment t ON a.Service_ID = t.Service_ID
    JOIN Nurse_Treatment nt ON t.Treatment_ID = nt.Treatment_ID
    WHERE nt.Nurse_ID = @NurseID
    ORDER BY a.Date, a.Time;
END;


EXEC sp_GetNurseAppointments @NurseID = 'U014';
```

Part 02: Database Vulnerabilities: Analysis and Mitigation

1. Privilege Escalation via Database Roles (Broken Access Control):

This Vulnerability occurs when a database system fails to properly apply restrictions to authenticated user actions. This flaw grants access users to perform actions beyond their privilege limit such as accessing sensitive data and altering database configurations.

Attack Methods:

- **Identifying Misconfigured Roles within the system:** Attackers go through the system to identify user roles with excessive privileges.
- **Exploiting indirect access through stored procedures and views**: attackers attempt to find what procedures or views present in the system could allow them to conduct restricted actions.
- **Exploiting Application Logic flaws:** unauthorized access can be gained by allowing unauthorized actions via poorly controlled queries or APIs.

Impact of Attacks:

- **Data Leaks:** Exposure of confidential data, in this case confidential patient records, medical staff records and financial data.
- **Unauthorized Data manipulation:** Unauthorized users may modify and delete sensitive data as well as may insert false data.
- **Compromised System functions:** users with malicious intentions could disable essential security features or corrupt fundamental system operations.

Countermeasures and Mitigation strategies:

- **Implementing strict RBAC rules:** Assigning database roles with strict permission levels via Role based access control (RBAC) so the users can only perform the intended tasks for that specific role.
- **Enforcing the principle of least privilege:** by granting the least amount of privileges so users can carry out their actions effectively, unnecessary administrative privileges can be minimized.
- **Periodic Access Auditing:** periodically reviewing database roles and user permissions to see if there has been any misconfigurations and correcting them.
- **Avoiding hardcoding admin credentials:** instead of embedding access in code, managing them dynamically ensures more security.

2. Insecure Database Configuration:

This Vulnerability occurs when a database system is deployed with default settings, weak configurations or unchanged credentials. Database setups like these are exposed to unauthorized access without needing complex exploits.

Attack Methods:

- **Scanning for open Database ports:** attackers identify exposed database services using tools like Nmap.
- **Brute-force attacks on credentials:** attacking using commonly used username and password pairs are used to gain unauthorized access (e.g. root:root, admin:admin).
- **Exploiting open permissions:** Poorly configured database systems or publicly accessible services allow threat agents to directly interact with sensitive database.

Impact of Attacks:

- **Unauthorized data access:** Attackers are able to view and extract sensitive data without worrying about permissions.
- **Database modification:** Attackers can alter records or enter false data into records.
- **Business disruption and data ransom:** Attackers may encrypt data holding it for ransom, leading to financial loss and system downtime affecting operations and users.

Countermeasures and Mitigation strategies:

- **Immediate change of default credentials:** replace all default usernames and passwords during database setup.
- **Disable unused services and network ports:** this can aid in reducing the systems attack surface.
- **Enabling Multi-Factor Authentication (MFA):** MFA helps to protect administrative accounts with multiple authentications.
- **Regular Auditing:** Consistently reviewing database settings helps in mitigating attacks due to insecure database configuration.
- **Implement security updates and patches:** Regular application of security patches can aid in addressing known vulnerabilities and configuration issues in a database system.

For further reference the EER Diagram and the Relational schema can be accessed through the following links:

- EER diagram: https://tinyurl.com/2jn9w2et
- Relational Schema: https://tinyurl.com/54zfa25u