Sri Lanka Institute of Information Technology

Group Members Details

- **Course Name:** Database Management System for Security
- **Course code:** IE2042
- **Batch:** Y2.S1.WD.CS

IT Number	**Name**
IT23822412	H.M.S.T. Karunarathna (Group Leader)
IT23834088	Raksha Mariyanesan
IT23833920	Rachana Mariyanesan
IT20625566	D.S.W. Weerakoon

Properly Documented Assumptions

1. User Roles and Inheritance

Assumption 1:
All users in the system (patients, doctors, nurses, admin staff) share common attributes (Name, Email, Phone) and are represented by a general User entity.
Justification:
The scenario specifies shared attributes, and the ERD uses an "IS A" relationship for inheritance. This supports normalization and avoids data redundancy.

2. Patient Visits and Admissions

Assumption 2:
A patient can have multiple outpatient visits and multiple inpatient admissions, each identified by unique IDs (VisitID for outpatient, AdmissionID for inpatient).
Justification:
The scenario mentions multiple visits/admissions per patient. This supports one-to-many relationships and accurate tracking of patient history.

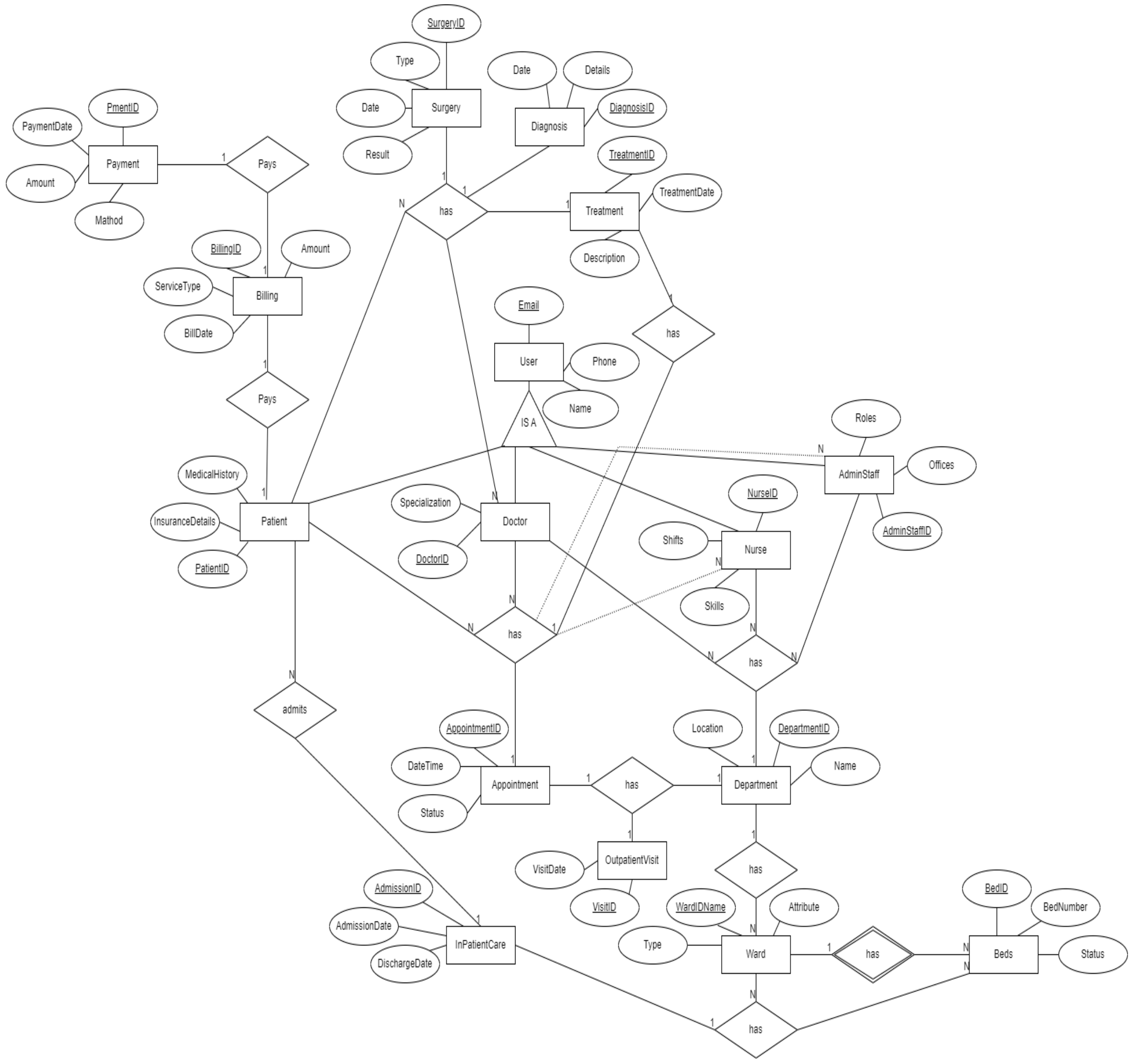
3. Appointment Scheduling

Assumption 3:
Appointments are scheduled between patients and doctors, optionally involving a nurse and an admin staff member who facilitates the appointment.
Justification:
The scenario and ERD show that appointments can involve multiple staff members, reflecting real-world clinic operations.

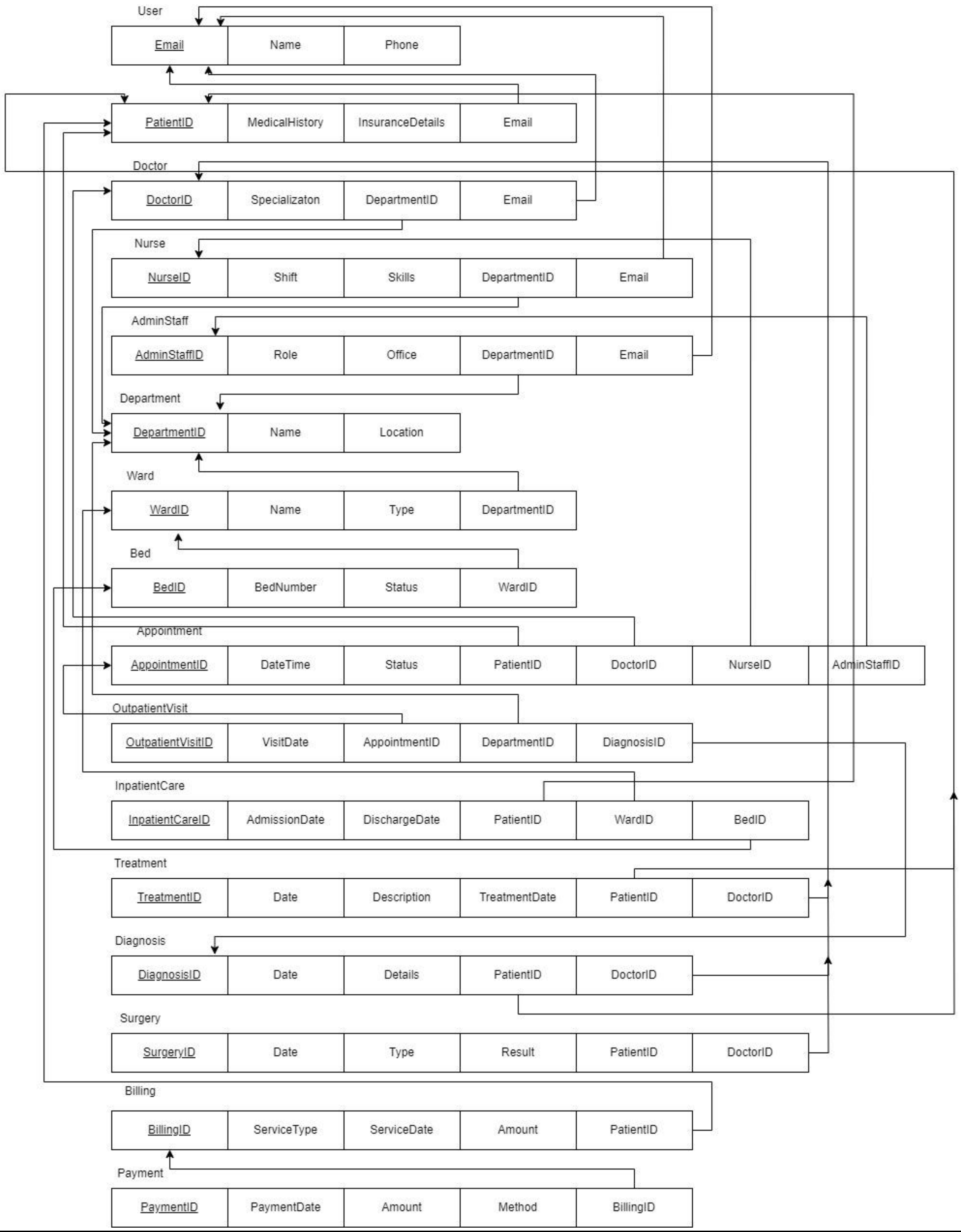
4. Billing and Payments

Assumption 4:
Each billing record is linked to a patient and may cover one or more services (treatments, diagnostics, etc.). Payments can be made in multiple installments per bill.
Justification:
The scenario states that patients can pay in installments, and the ERD shows a one-to-many relationship between Billing and Payment.

ER Diagram


SurgeryID
Type
Date
Surgery
Result
Date
Details
Diagnosis
DiagnosisID
PmentID
PaymentDate
Payment
Amount
Mathod
Pays
has
Treatment
TreatmentID
TreatmentDate
Description
BillingID
Amount
ServiceType
Billing
BillDate
Pays
has
Email
User
Phone
Name
IS A
Roles
AdminStaff
Offices
AdminStaffID
MedicalHistory
InsuranceDetails
Patient
PatientID
Specialization
Doctor
DoctorID
NurseID
Shifts
Nurse
Skills
has
has
admits
AppointmentID
DateTime
Appointment
Status
has
Location
DepartmentID
Department
Name
OutpatientVisit
VisitDate
VisitID
has
AdmissionID
AdmissionDate
InPatientCare
DischargeDate
WardIDName
Attribute
Type
Ward
has
BedID
BedNumber
Beds
Status
has
1
N

Logical Model



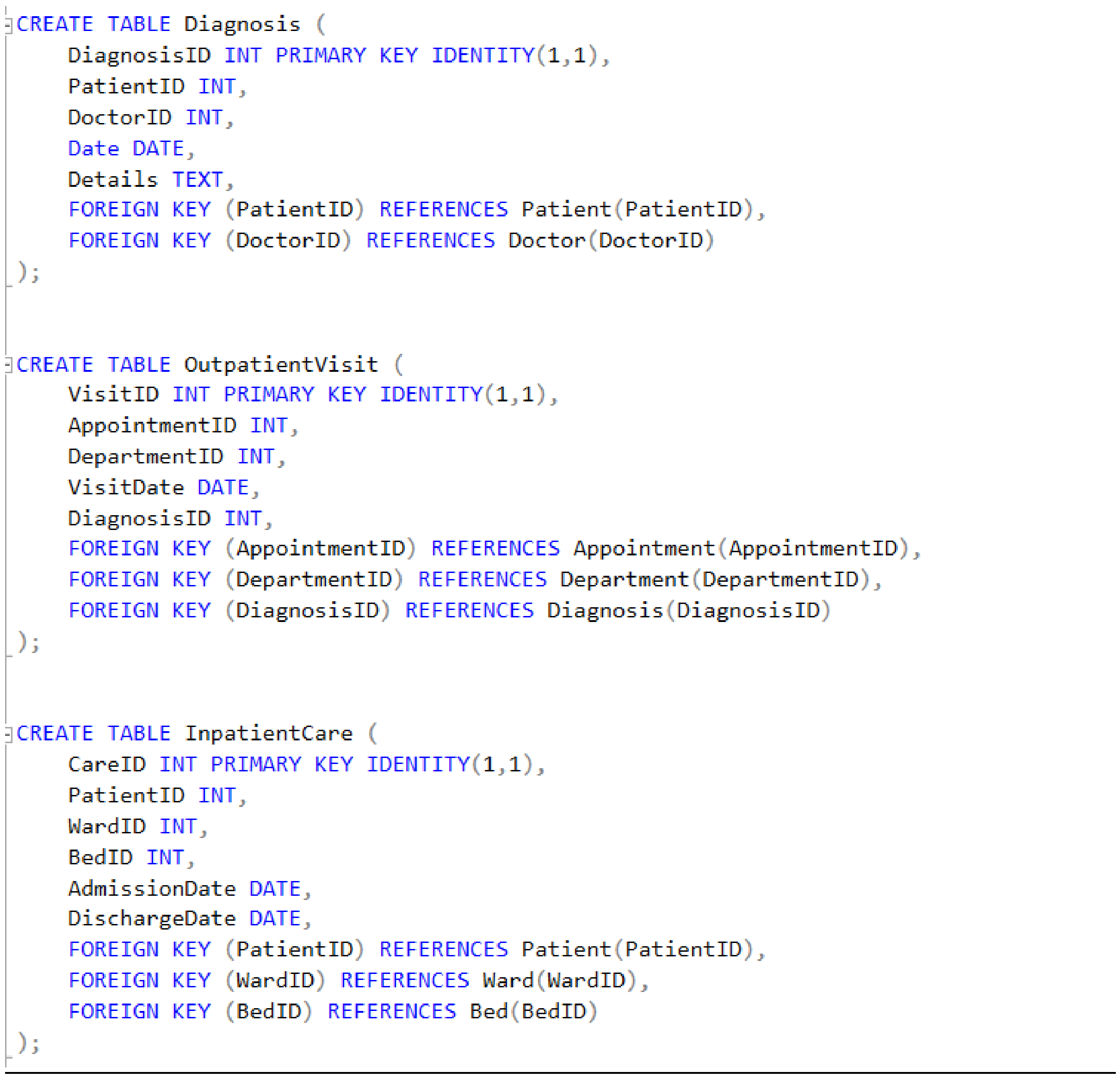
3 NF Normalization

This logical schema is already in 3 NF Form.

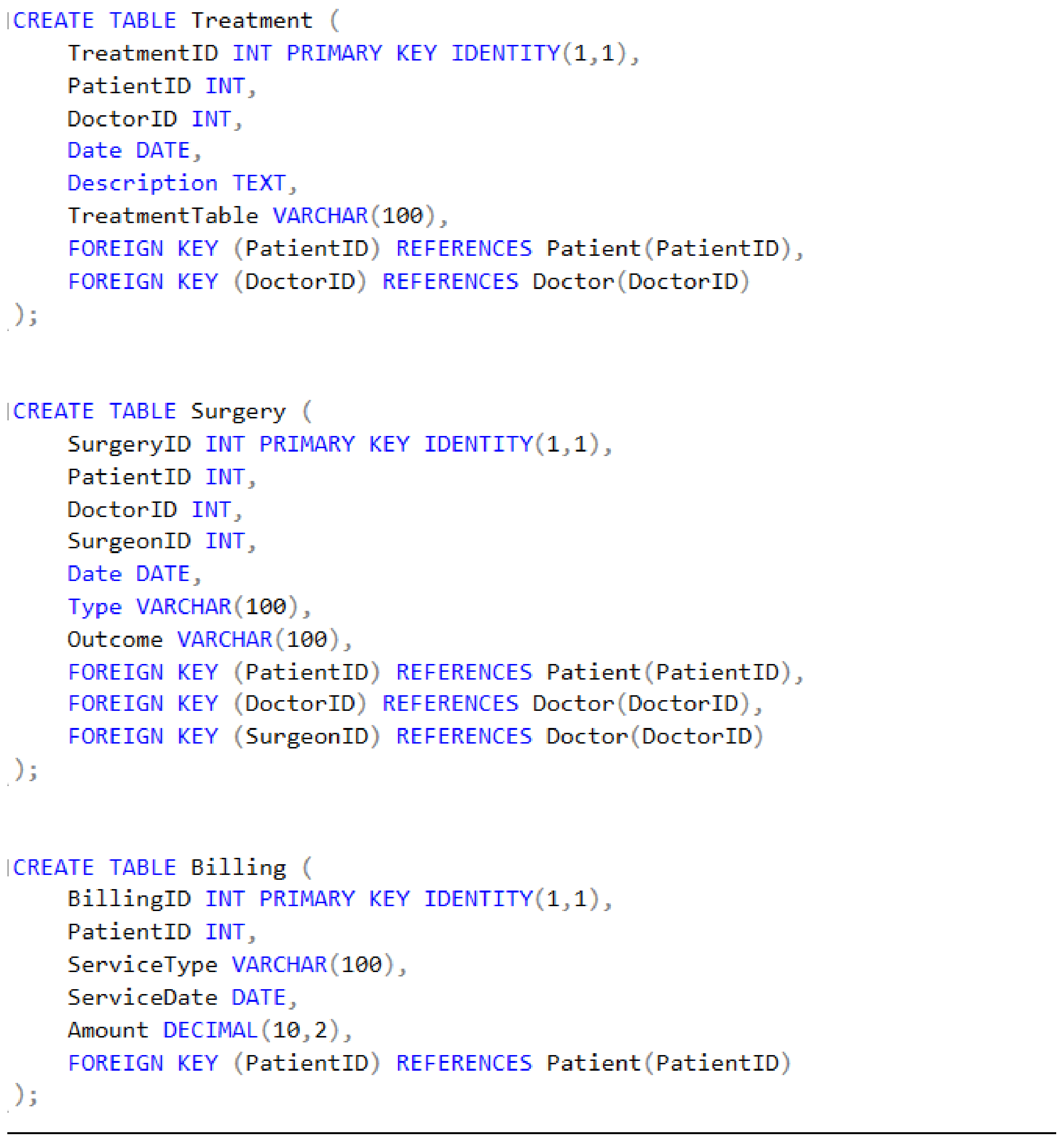
Because, all non-key attributes are fully functionally dependent on the primary key & there are no transitive dependencies.

All attributes are atomic.

Table Implementations

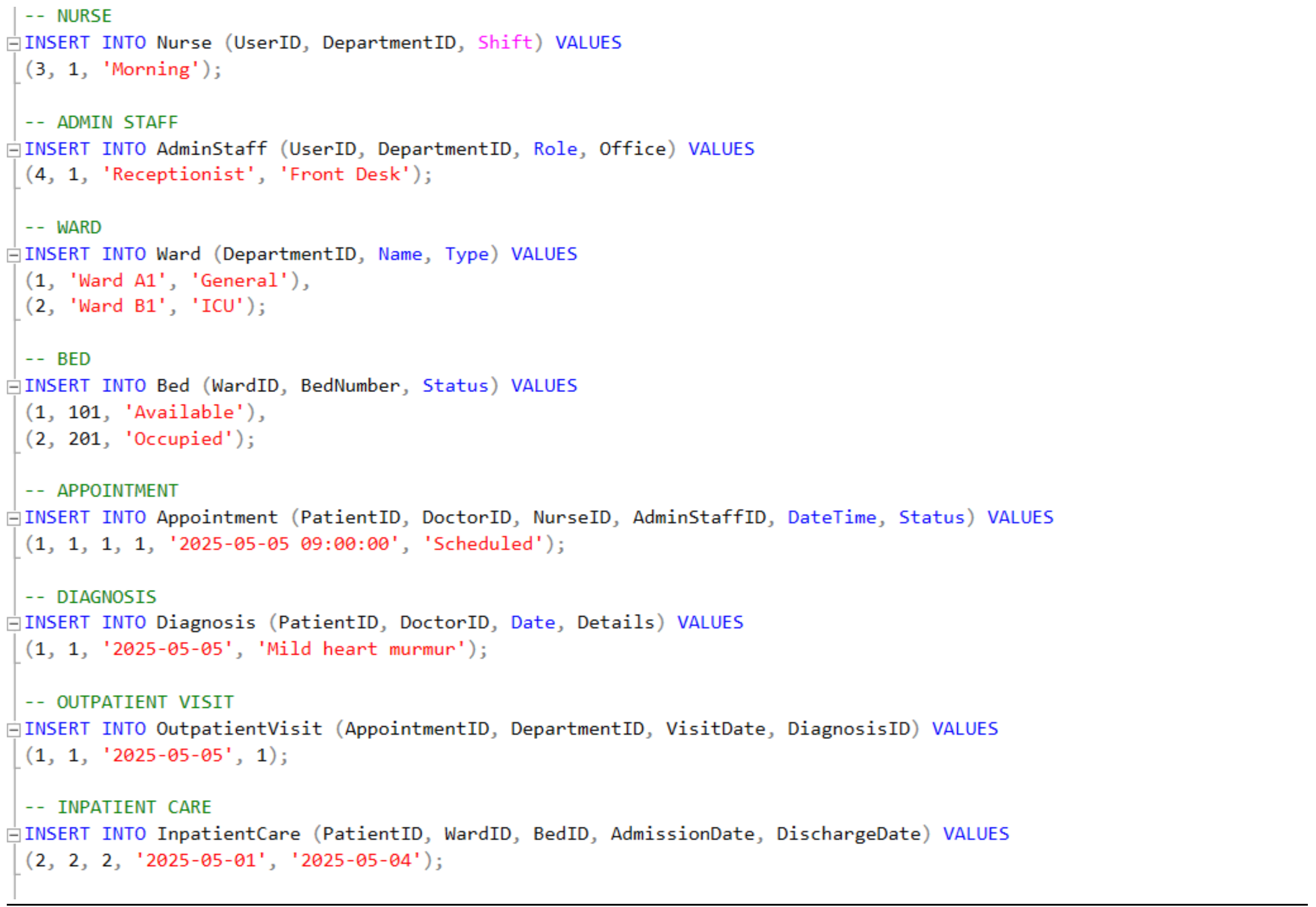


```sql
CREATE TABLE Diagnosis (
    DiagnosisID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT,
    DoctorID INT,
    Date DATE,
    Details TEXT,
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID)
);


CREATE TABLE OutpatientVisit (
    VisitID INT PRIMARY KEY IDENTITY(1,1),
    AppointmentID INT,
    DepartmentID INT,
    VisitDate DATE,
    DiagnosisID INT,
    FOREIGN KEY (AppointmentID) REFERENCES Appointment(AppointmentID),
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID),
    FOREIGN KEY (DiagnosisID) REFERENCES Diagnosis(DiagnosisID)
);


CREATE TABLE InpatientCare (
    CareID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT,
    WardID INT,
    BedID INT,
    AdmissionDate DATE,
    DischargeDate DATE,
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY (WardID) REFERENCES Ward(WardID),
    FOREIGN KEY (BedID) REFERENCES Bed(BedID)
);
```

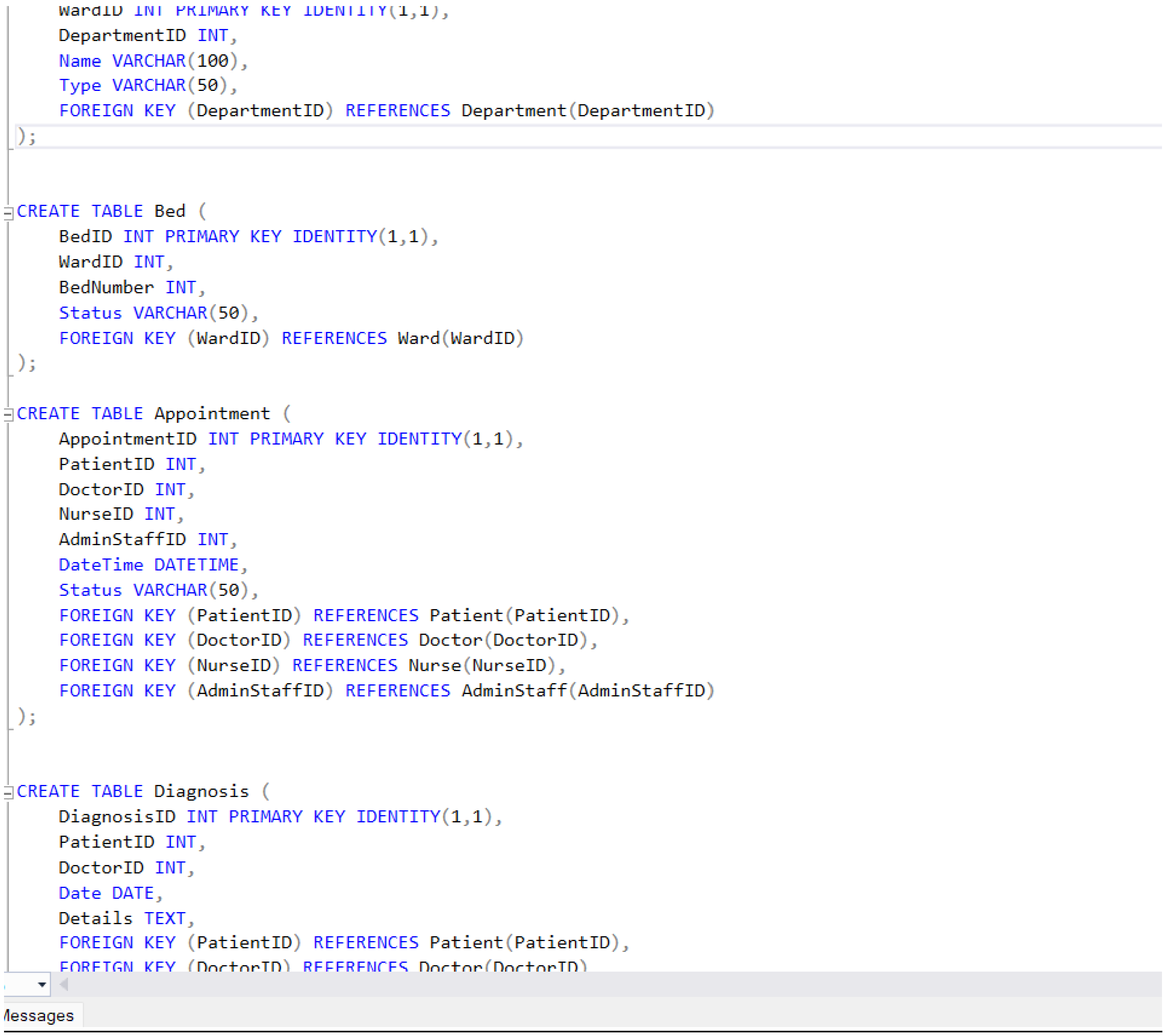


```sql
CREATE TABLE Treatment (
    TreatmentID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT,
    DoctorID INT,
    Date DATE,
    Description TEXT,
    TreatmentTable VARCHAR(100),
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID)
);


CREATE TABLE Surgery (
    SurgeryID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT,
    DoctorID INT,
    SurgeonID INT,
    Date DATE,
    Type VARCHAR(100),
    Outcome VARCHAR(100),
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID),
    FOREIGN KEY (SurgeonID) REFERENCES Doctor(DoctorID)
);


CREATE TABLE Billing (
    BillingID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT,
    ServiceType VARCHAR(100),
    ServiceDate DATE,
    Amount DECIMAL(10,2),
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID)
);
```

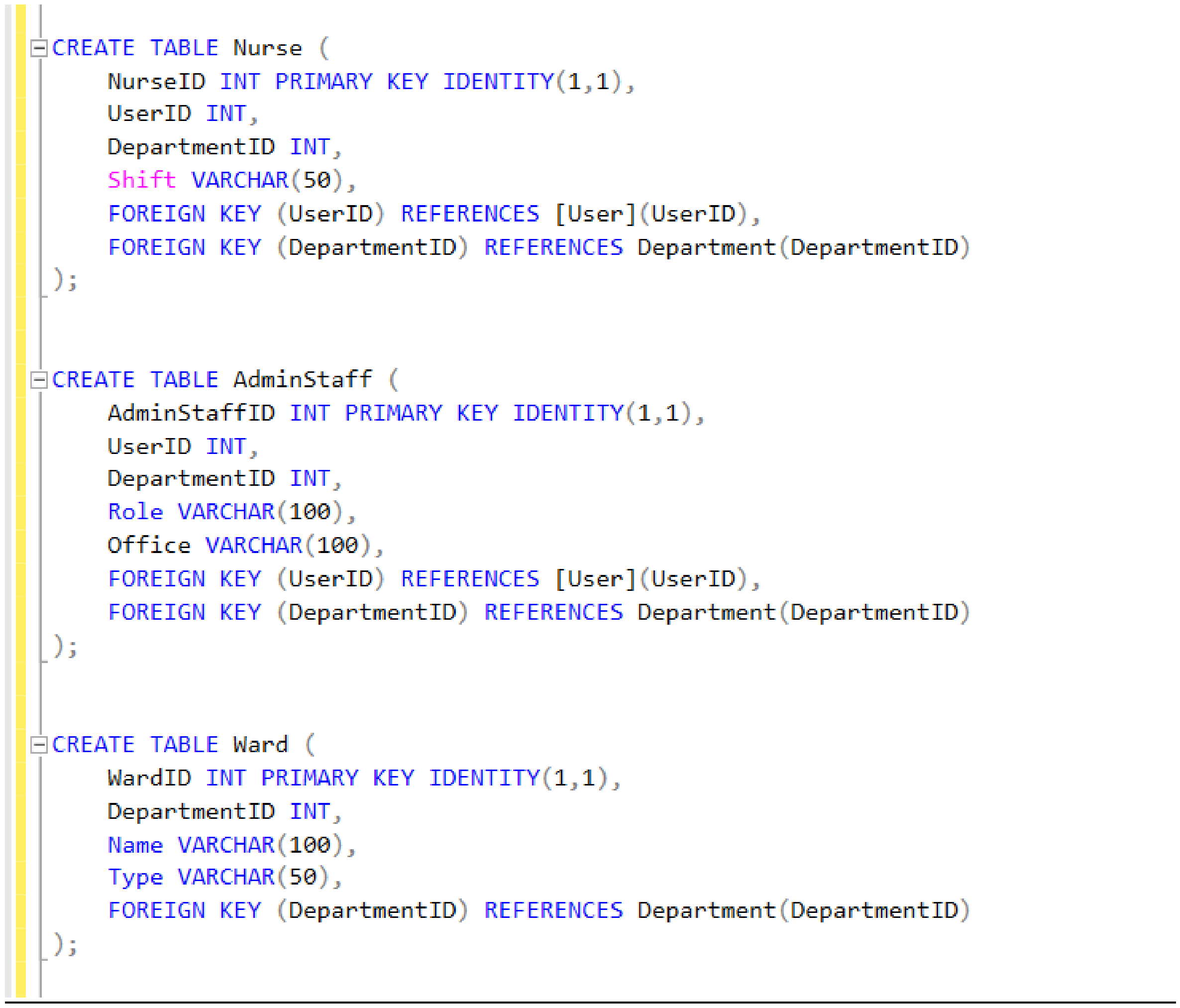
```sql
CREATE TABLE Payment (
    PaymentID INT PRIMARY KEY IDENTITY(1,1),
    BillingID INT,
    PaymentDate DATE,
    Amount DECIMAL(10,2),
    Method VARCHAR(50),
    FOREIGN KEY (BillingID) REFERENCES Billing(BillingID)
);

--USER
INSERT INTO [User] (Name, Email, Phone) VALUES
('John Doe', 'john@example.com', '1111111111'),
('Alice Smith', 'alice@example.com', '2222222222'),
('Nurse Clara', 'clara@example.com', '3333333333'),
('Joe Admin', 'joe@example.com', '4444444444'),
('Jane Roe', 'jane@example.com', '5555555555');

-- DEPARTMENT
INSERT INTO Department (Name, Location) VALUES
('Cardiology', 'Block A'),
('Neurology', 'Block B');

-- PATIENT
INSERT INTO Patient (UserID, MedicalHistory, Insurance) VALUES
(1, 'Hypertension', 'HealthSecure'),
(5, 'Asthma', 'MediCare');

-- DOCTOR
INSERT INTO Doctor (UserID, DepartmentID, Specialization) VALUES
(2, 1, 'Cardiologist');

-- NURSE
INSERT INTO Nurse (UserID, DepartmentID, Shift) VALUES
(3, 1, 'Morning');
```



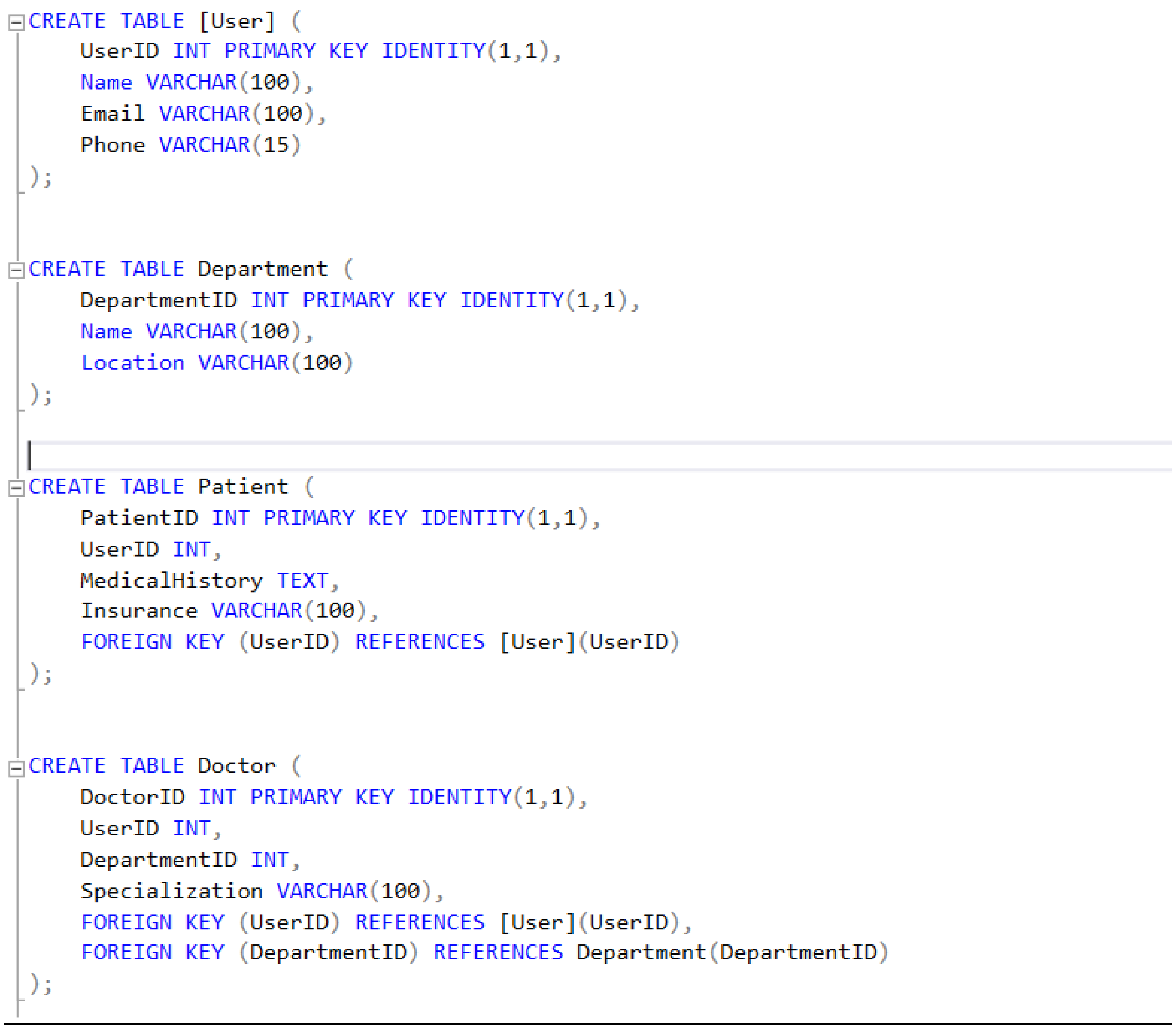
```sql
-- NURSE
INSERT INTO Nurse (UserID, DepartmentID, Shift) VALUES
(3, 1, 'Morning');

-- ADMIN STAFF
INSERT INTO AdminStaff (UserID, DepartmentID, Role, Office) VALUES
(4, 1, 'Receptionist', 'Front Desk');

-- WARD
INSERT INTO Ward (DepartmentID, Name, Type) VALUES
(1, 'Ward A1', 'General'),
(2, 'Ward B1', 'ICU');

-- BED
INSERT INTO Bed (WardID, BedNumber, Status) VALUES
(1, 101, 'Available'),
(2, 201, 'Occupied');

-- APPOINTMENT
INSERT INTO Appointment (PatientID, DoctorID, NurseID, AdminStaffID, DateTime, Status) VALUES
(1, 1, 1, 1, '2025-05-05 09:00:00', 'Scheduled');

-- DIAGNOSIS
INSERT INTO Diagnosis (PatientID, DoctorID, Date, Details) VALUES
(1, 1, '2025-05-05', 'Mild heart murmur');

-- OUTPATIENT VISIT
INSERT INTO OutpatientVisit (AppointmentID, DepartmentID, VisitDate, DiagnosisID) VALUES
(1, 1, '2025-05-05', 1);

-- INPATIENT CARE
INSERT INTO InpatientCare (PatientID, WardID, BedID, AdmissionDate, DischargeDate) VALUES
(2, 2, 2, '2025-05-01', '2025-05-04');
```



```
    WardID INT PRIMARY KEY IDENTITY(1,1),
    DepartmentID INT,
    Name VARCHAR(100),
    Type VARCHAR(50),
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);


CREATE TABLE Bed (
    BedID INT PRIMARY KEY IDENTITY(1,1),
    WardID INT,
    BedNumber INT,
    Status VARCHAR(50),
    FOREIGN KEY (WardID) REFERENCES Ward(WardID)
);

CREATE TABLE Appointment (
    AppointmentID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT,
    DoctorID INT,
    NurseID INT,
    AdminStaffID INT,
    DateTime DATETIME,
    Status VARCHAR(50),
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID),
    FOREIGN KEY (NurseID) REFERENCES Nurse(NurseID),
    FOREIGN KEY (AdminStaffID) REFERENCES AdminStaff(AdminStaffID)
);


CREATE TABLE Diagnosis (
    DiagnosisID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT,
    DoctorID INT,
    Date DATE,
    Details TEXT,
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID)
```

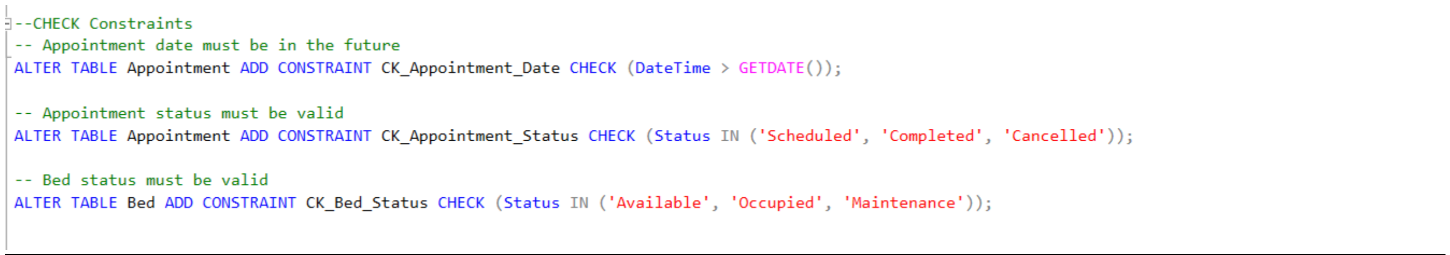


```sql
CREATE TABLE Nurse (
    NurseID INT PRIMARY KEY IDENTITY(1,1),
    UserID INT,
    DepartmentID INT,
    Shift VARCHAR(50),
    FOREIGN KEY (UserID) REFERENCES [User](UserID),
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);


CREATE TABLE AdminStaff (
    AdminStaffID INT PRIMARY KEY IDENTITY(1,1),
    UserID INT,
    DepartmentID INT,
    Role VARCHAR(100),
    Office VARCHAR(100),
    FOREIGN KEY (UserID) REFERENCES [User](UserID),
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);


CREATE TABLE Ward (
    WardID INT PRIMARY KEY IDENTITY(1,1),
    DepartmentID INT,
    Name VARCHAR(100),
    Type VARCHAR(50),
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);
```

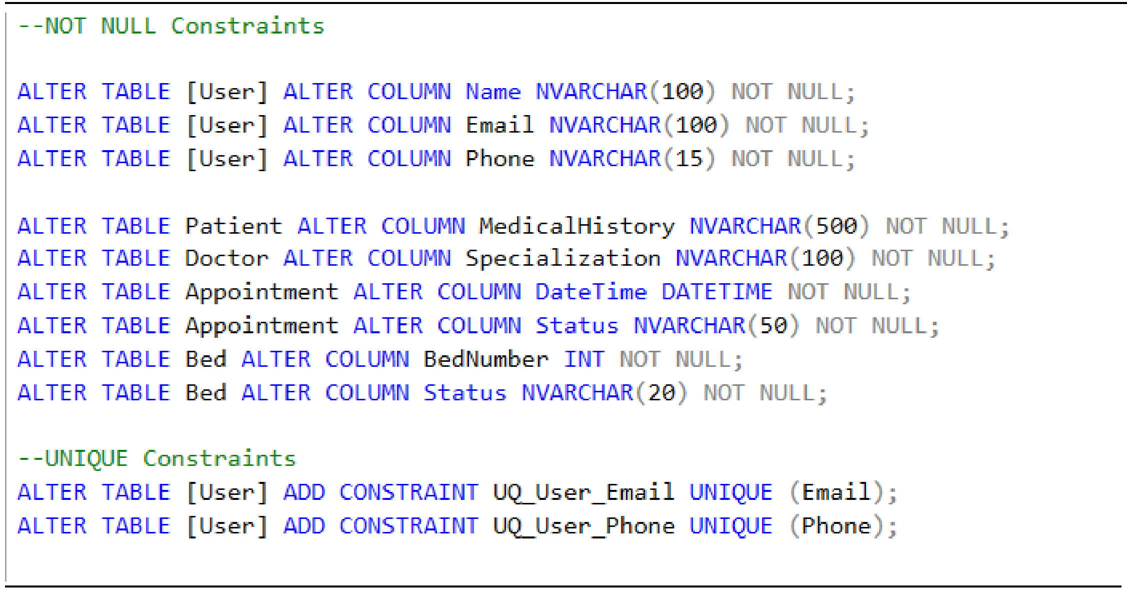


```sql
CREATE TABLE [User] (
    UserID INT PRIMARY KEY IDENTITY(1,1),
    Name VARCHAR(100),
    Email VARCHAR(100),
    Phone VARCHAR(15)
);


CREATE TABLE Department (
    DepartmentID INT PRIMARY KEY IDENTITY(1,1),
    Name VARCHAR(100),
    Location VARCHAR(100)
);


CREATE TABLE Patient (
    PatientID INT PRIMARY KEY IDENTITY(1,1),
    UserID INT,
    MedicalHistory TEXT,
    Insurance VARCHAR(100),
    FOREIGN KEY (UserID) REFERENCES [User](UserID)
);


CREATE TABLE Doctor (
    DoctorID INT PRIMARY KEY IDENTITY(1,1),
    UserID INT,
    DepartmentID INT,
    Specialization VARCHAR(100),
    FOREIGN KEY (UserID) REFERENCES [User](UserID),
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);
```

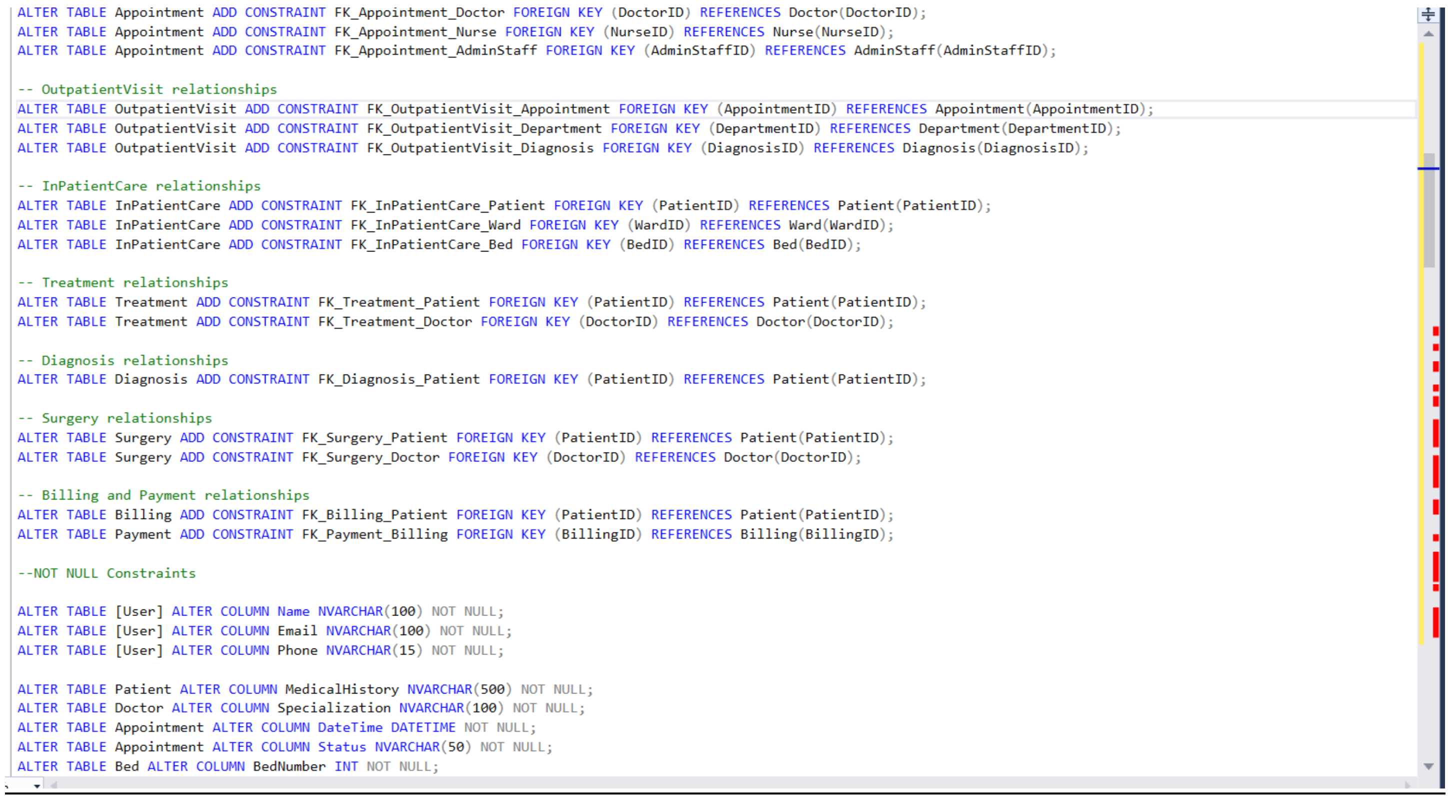
Constraints Implementation



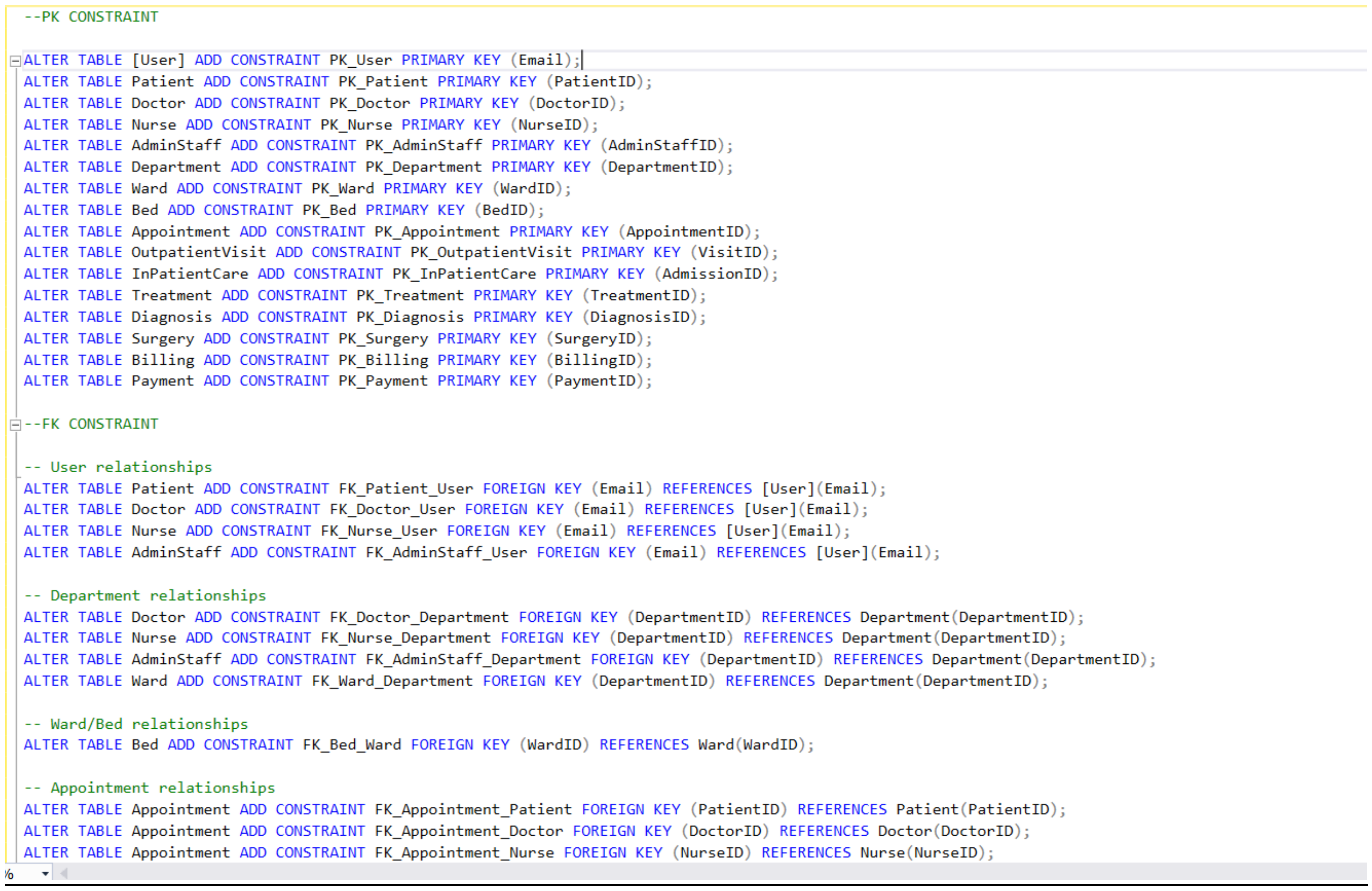
```sql
--CHECK Constraints
-- Appointment date must be in the future
ALTER TABLE Appointment ADD CONSTRAINT CK_Appointment_Date CHECK (DateTime > GETDATE());

-- Appointment status must be valid
ALTER TABLE Appointment ADD CONSTRAINT CK_Appointment_Status CHECK (Status IN ('Scheduled', 'Completed', 'Cancelled'));

-- Bed status must be valid
ALTER TABLE Bed ADD CONSTRAINT CK_Bed_Status CHECK (Status IN ('Available', 'Occupied', 'Maintenance'));
```



```sql
--NOT NULL Constraints

ALTER TABLE [User] ALTER COLUMN Name NVARCHAR(100) NOT NULL;
ALTER TABLE [User] ALTER COLUMN Email NVARCHAR(100) NOT NULL;
ALTER TABLE [User] ALTER COLUMN Phone NVARCHAR(15) NOT NULL;

ALTER TABLE Patient ALTER COLUMN MedicalHistory NVARCHAR(500) NOT NULL;
ALTER TABLE Doctor ALTER COLUMN Specialization NVARCHAR(100) NOT NULL;
ALTER TABLE Appointment ALTER COLUMN DateTime DATETIME NOT NULL;
ALTER TABLE Appointment ALTER COLUMN Status NVARCHAR(50) NOT NULL;
ALTER TABLE Bed ALTER COLUMN BedNumber INT NOT NULL;
ALTER TABLE Bed ALTER COLUMN Status NVARCHAR(20) NOT NULL;

--UNIQUE Constraints
ALTER TABLE [User] ADD CONSTRAINT UQ_User_Email UNIQUE (Email);
ALTER TABLE [User] ADD CONSTRAINT UQ_User_Phone UNIQUE (Phone);
```

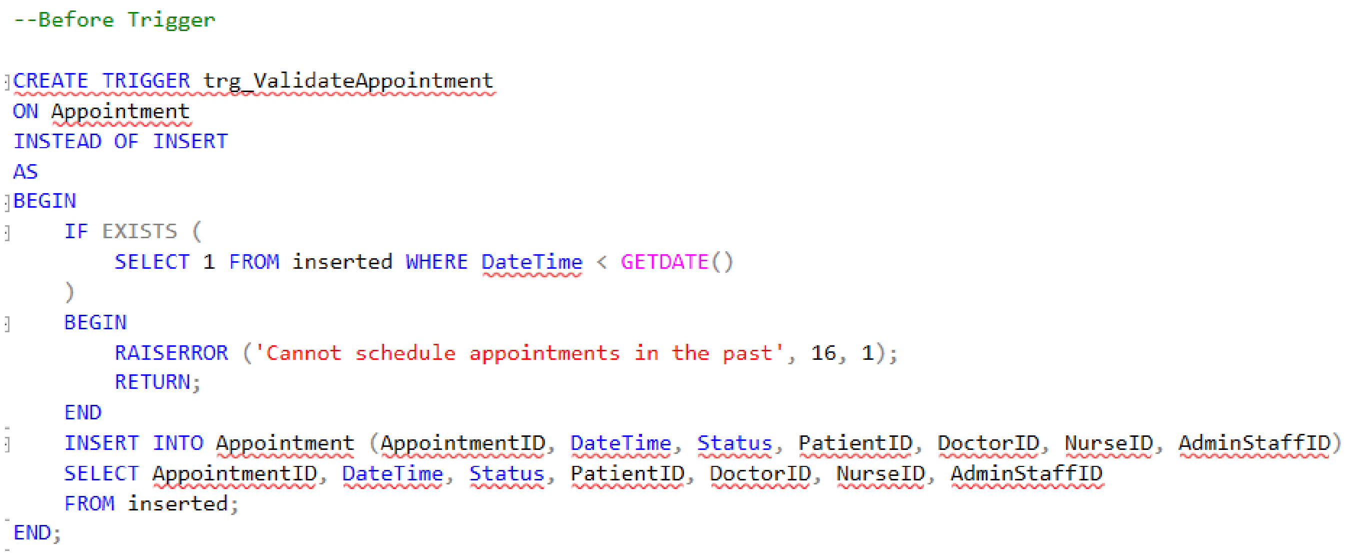


```sql
ALTER TABLE Appointment ADD CONSTRAINT FK_Appointment_Doctor FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID);
ALTER TABLE Appointment ADD CONSTRAINT FK_Appointment_Nurse FOREIGN KEY (NurseID) REFERENCES Nurse(NurseID);
ALTER TABLE Appointment ADD CONSTRAINT FK_Appointment_AdminStaff FOREIGN KEY (AdminStaffID) REFERENCES AdminStaff(AdminStaffID);

-- OutpatientVisit relationships
ALTER TABLE OutpatientVisit ADD CONSTRAINT FK_OutpatientVisit_Appointment FOREIGN KEY (AppointmentID) REFERENCES Appointment(AppointmentID);
ALTER TABLE OutpatientVisit ADD CONSTRAINT FK_OutpatientVisit_Department FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID);
ALTER TABLE OutpatientVisit ADD CONSTRAINT FK_OutpatientVisit_Diagnosis FOREIGN KEY (DiagnosisID) REFERENCES Diagnosis(DiagnosisID);

-- InPatientCare relationships
ALTER TABLE InPatientCare ADD CONSTRAINT FK_InPatientCare_Patient FOREIGN KEY (PatientID) REFERENCES Patient(PatientID);
ALTER TABLE InPatientCare ADD CONSTRAINT FK_InPatientCare_Ward FOREIGN KEY (WardID) REFERENCES Ward(WardID);
ALTER TABLE InPatientCare ADD CONSTRAINT FK_InPatientCare_Bed FOREIGN KEY (BedID) REFERENCES Bed(BedID);

-- Treatment relationships
ALTER TABLE Treatment ADD CONSTRAINT FK_Treatment_Patient FOREIGN KEY (PatientID) REFERENCES Patient(PatientID);
ALTER TABLE Treatment ADD CONSTRAINT FK_Treatment_Doctor FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID);

-- Diagnosis relationships
ALTER TABLE Diagnosis ADD CONSTRAINT FK_Diagnosis_Patient FOREIGN KEY (PatientID) REFERENCES Patient(PatientID);

-- Surgery relationships
ALTER TABLE Surgery ADD CONSTRAINT FK_Surgery_Patient FOREIGN KEY (PatientID) REFERENCES Patient(PatientID);
ALTER TABLE Surgery ADD CONSTRAINT FK_Surgery_Doctor FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID);

-- Billing and Payment relationships
ALTER TABLE Billing ADD CONSTRAINT FK_Billing_Patient FOREIGN KEY (PatientID) REFERENCES Patient(PatientID);
ALTER TABLE Payment ADD CONSTRAINT FK_Payment_Billing FOREIGN KEY (BillingID) REFERENCES Billing(BillingID);

--NOT NULL Constraints

ALTER TABLE [User] ALTER COLUMN Name NVARCHAR(100) NOT NULL;
ALTER TABLE [User] ALTER COLUMN Email NVARCHAR(100) NOT NULL;
ALTER TABLE [User] ALTER COLUMN Phone NVARCHAR(15) NOT NULL;

ALTER TABLE Patient ALTER COLUMN MedicalHistory NVARCHAR(500) NOT NULL;
ALTER TABLE Doctor ALTER COLUMN Specialization NVARCHAR(100) NOT NULL;
ALTER TABLE Appointment ALTER COLUMN DateTime DATETIME NOT NULL;
ALTER TABLE Appointment ALTER COLUMN Status NVARCHAR(50) NOT NULL;
ALTER TABLE Bed ALTER COLUMN BedNumber INT NOT NULL;
```

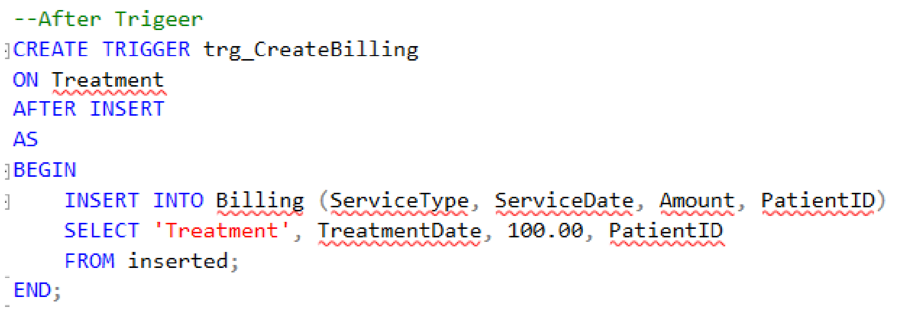


```sql
--PK CONSTRAINT

ALTER TABLE [User] ADD CONSTRAINT PK_User PRIMARY KEY (Email);
ALTER TABLE Patient ADD CONSTRAINT PK_Patient PRIMARY KEY (PatientID);
ALTER TABLE Doctor ADD CONSTRAINT PK_Doctor PRIMARY KEY (DoctorID);
ALTER TABLE Nurse ADD CONSTRAINT PK_Nurse PRIMARY KEY (NurseID);
ALTER TABLE AdminStaff ADD CONSTRAINT PK_AdminStaff PRIMARY KEY (AdminStaffID);
ALTER TABLE Department ADD CONSTRAINT PK_Department PRIMARY KEY (DepartmentID);
ALTER TABLE Ward ADD CONSTRAINT PK_Ward PRIMARY KEY (WardID);
ALTER TABLE Bed ADD CONSTRAINT PK_Bed PRIMARY KEY (BedID);
ALTER TABLE Appointment ADD CONSTRAINT PK_Appointment PRIMARY KEY (AppointmentID);
ALTER TABLE OutpatientVisit ADD CONSTRAINT PK_OutpatientVisit PRIMARY KEY (VisitID);
ALTER TABLE InPatientCare ADD CONSTRAINT PK_InPatientCare PRIMARY KEY (AdmissionID);
ALTER TABLE Treatment ADD CONSTRAINT PK_Treatment PRIMARY KEY (TreatmentID);
ALTER TABLE Diagnosis ADD CONSTRAINT PK_Diagnosis PRIMARY KEY (DiagnosisID);
ALTER TABLE Surgery ADD CONSTRAINT PK_Surgery PRIMARY KEY (SurgeryID);
ALTER TABLE Billing ADD CONSTRAINT PK_Billing PRIMARY KEY (BillingID);
ALTER TABLE Payment ADD CONSTRAINT PK_Payment PRIMARY KEY (PaymentID);

--FK CONSTRAINT

-- User relationships
ALTER TABLE Patient ADD CONSTRAINT FK_Patient_User FOREIGN KEY (Email) REFERENCES [User](Email);
ALTER TABLE Doctor ADD CONSTRAINT FK_Doctor_User FOREIGN KEY (Email) REFERENCES [User](Email);
ALTER TABLE Nurse ADD CONSTRAINT FK_Nurse_User FOREIGN KEY (Email) REFERENCES [User](Email);
ALTER TABLE AdminStaff ADD CONSTRAINT FK_AdminStaff_User FOREIGN KEY (Email) REFERENCES [User](Email);

-- Department relationships
ALTER TABLE Doctor ADD CONSTRAINT FK_Doctor_Department FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID);
ALTER TABLE Nurse ADD CONSTRAINT FK_Nurse_Department FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID);
ALTER TABLE AdminStaff ADD CONSTRAINT FK_AdminStaff_Department FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID);
ALTER TABLE Ward ADD CONSTRAINT FK_Ward_Department FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID);

-- Ward/Bed relationships
ALTER TABLE Bed ADD CONSTRAINT FK_Bed_Ward FOREIGN KEY (WardID) REFERENCES Ward(WardID);

-- Appointment relationships
ALTER TABLE Appointment ADD CONSTRAINT FK_Appointment_Patient FOREIGN KEY (PatientID) REFERENCES Patient(PatientID);
ALTER TABLE Appointment ADD CONSTRAINT FK_Appointment_Doctor FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID);
ALTER TABLE Appointment ADD CONSTRAINT FK_Appointment_Nurse FOREIGN KEY (NurseID) REFERENCES Nurse(NurseID);
```

Triggers

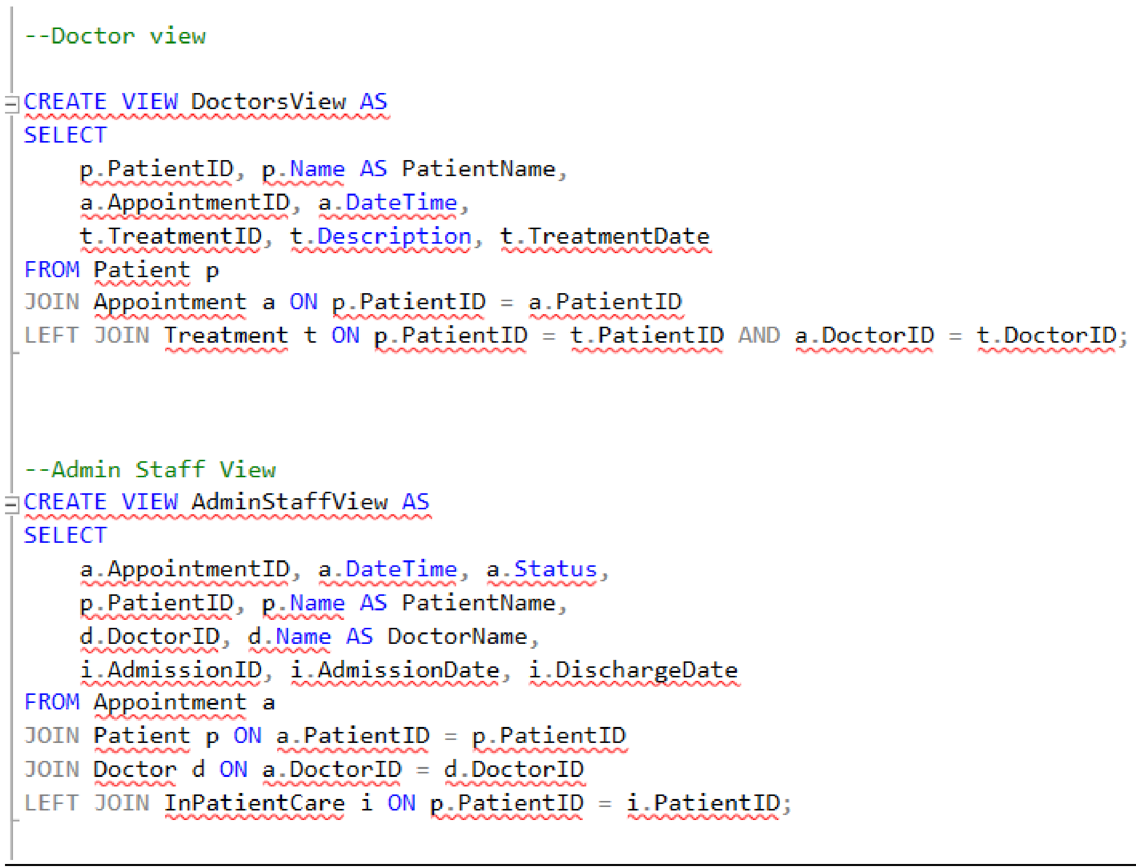


```sql
--Before Trigger

CREATE TRIGGER trg_ValidateAppointment
ON Appointment
INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1 FROM inserted WHERE DateTime < GETDATE()
    )
    BEGIN
        RAISERROR ('Cannot schedule appointments in the past', 16, 1);
        RETURN;
    END
    INSERT INTO Appointment (AppointmentID, DateTime, Status, PatientID, DoctorID, NurseID, AdminStaffID)
    SELECT AppointmentID, DateTime, Status, PatientID, DoctorID, NurseID, AdminStaffID
    FROM inserted;
END;
```



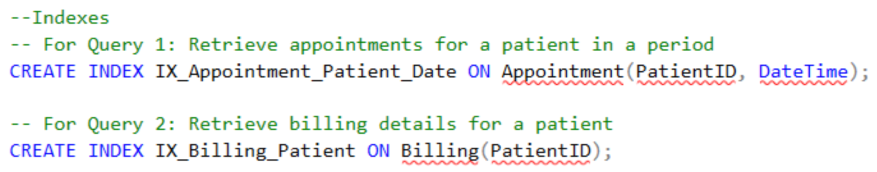
```sql
--After Trigeer
CREATE TRIGGER trg_CreateBilling
ON Treatment
AFTER INSERT
AS
BEGIN
    INSERT INTO Billing (ServiceType, ServiceDate, Amount, PatientID)
    SELECT 'Treatment', TreatmentDate, 100.00, PatientID
    FROM inserted;
END;
```

Views



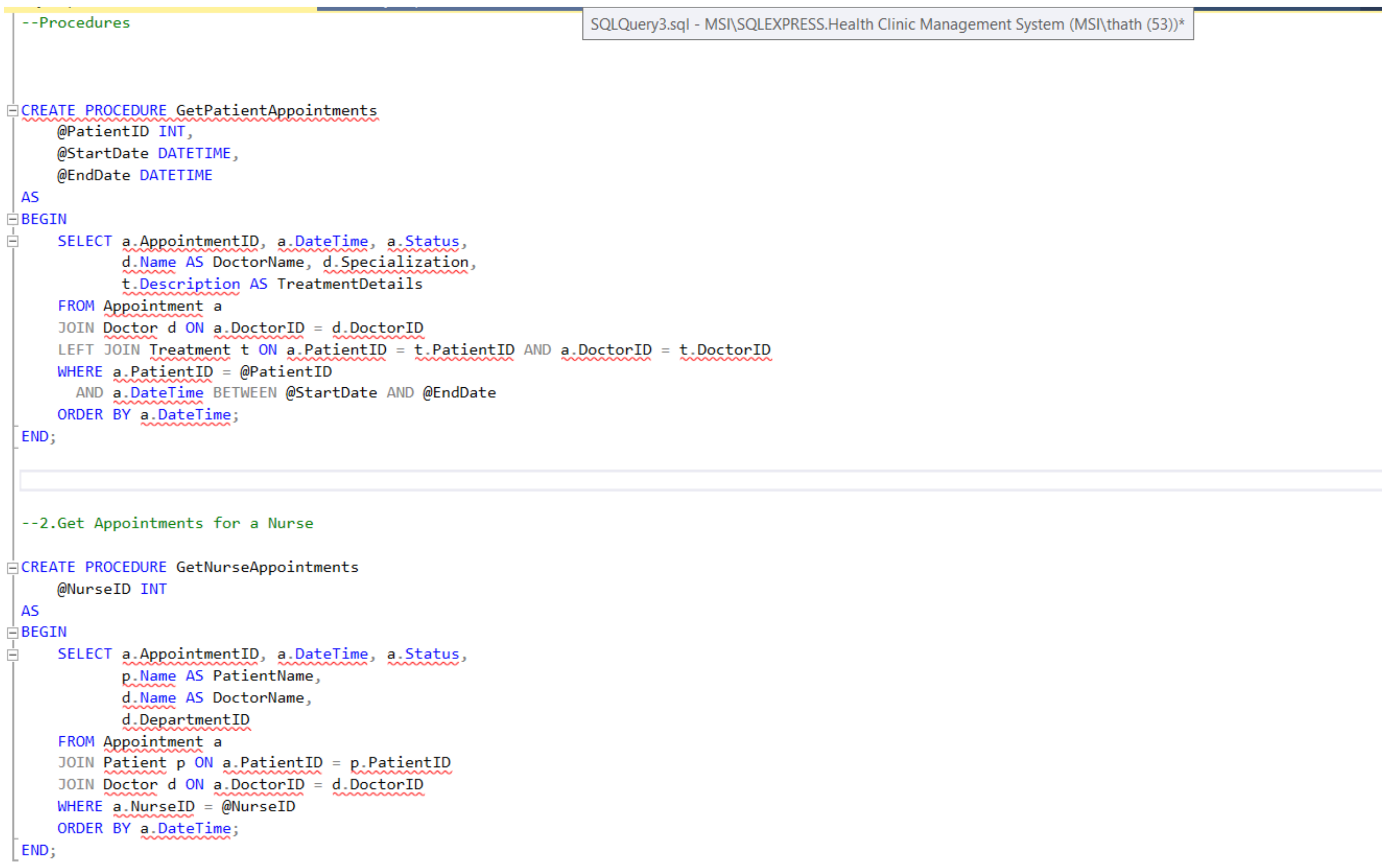
```sql
--Doctor view

CREATE VIEW DoctorsView AS
SELECT
    p.PatientID, p.Name AS PatientName,
    a.AppointmentID, a.DateTime,
    t.TreatmentID, t.Description, t.TreatmentDate
FROM Patient p
JOIN Appointment a ON p.PatientID = a.PatientID
LEFT JOIN Treatment t ON p.PatientID = t.PatientID AND a.DoctorID = t.DoctorID;



--Admin Staff View
CREATE VIEW AdminStaffView AS
SELECT
    a.AppointmentID, a.DateTime, a.Status,
    p.PatientID, p.Name AS PatientName,
    d.DoctorID, d.Name AS DoctorName,
    i.AdmissionID, i.AdmissionDate, i.DischargeDate
FROM Appointment a
JOIN Patient p ON a.PatientID = p.PatientID
JOIN Doctor d ON a.DoctorID = d.DoctorID
LEFT JOIN InPatientCare i ON p.PatientID = i.PatientID;
```

Indexes



```sql
--Indexes
-- For Query 1: Retrieve appointments for a patient in a period
CREATE INDEX IX_Appointment_Patient_Date ON Appointment(PatientID, DateTime);

-- For Query 2: Retrieve billing details for a patient
CREATE INDEX IX_Billing_Patient ON Billing(PatientID);
```

Stored Procedures



SQLQuery3.sql - MSI\SQLEXPRESS.Health Clinic Management System (MSI\thath (53))*

```sql
--Procedures


CREATE PROCEDURE GetPatientAppointments
    @PatientID INT,
    @StartDate DATETIME,
    @EndDate DATETIME
AS
BEGIN
    SELECT a.AppointmentID, a.DateTime, a.Status,
           d.Name AS DoctorName, d.Specialization,
           t.Description AS TreatmentDetails
    FROM Appointment a
    JOIN Doctor d ON a.DoctorID = d.DoctorID
    LEFT JOIN Treatment t ON a.PatientID = t.PatientID AND a.DoctorID = t.DoctorID
    WHERE a.PatientID = @PatientID
      AND a.DateTime BETWEEN @StartDate AND @EndDate
    ORDER BY a.DateTime;
END;


--2.Get Appointments for a Nurse

CREATE PROCEDURE GetNurseAppointments
    @NurseID INT
AS
BEGIN
    SELECT a.AppointmentID, a.DateTime, a.Status,
           p.Name AS PatientName,
           d.Name AS DoctorName,
           d.DepartmentID
    FROM Appointment a
    JOIN Patient p ON a.PatientID = p.PatientID
    JOIN Doctor d ON a.DoctorID = d.DoctorID
    WHERE a.NurseID = @NurseID
    ORDER BY a.DateTime;
END;
```

Vulnerability 1:SQL Injection

SQL injection is one of the most common and dangerous vulnerabilities in web applications that use databases. It occurs when an attacker injects or manipulates malicious SQL code through user input fields, resulting in the execution of unauthorized commands by the database.

Techniques Used:

- Attackers misuse poor input validation by injecting SQL commands into input fields (search fields or login forms).
- The application interprets malicious input and executes it on behalf of the database, allowing attackers to change queries

Impact

- **Data Breach:**

  Attacker access to sensitive data like personal, financial, or medical records.

- **Unauthorized Access:**

  Evade authentication with admin privileges or access to limited data.

- **Data Manipulation:**

  Alter, insert, or delete records, compromising data integrity and reliability.

- **Business Disruption:**

  Interfere with business processes, create downtime, and affect reputation.

- **Lateral Movement:**

  Leverage compromised access to attack other systems

Mitigation and Countermeasures

- **Input Validation and Sanitization:**

  Validate and sanitize all user input to prevent running malicious SQL code.

- **Parameterized Queries/Prepared Statements:**

  Use parameterized statements to keep SQL code separated from data, so that attacks injecting malicious queries are not possible.

- **Least Privilege Principle:**

  Restrict database user privileges to the least amount possible.

- **Web Application Firewalls (WAF):**

  Use WAFs to detect and prevent SQL injection attacks in real-time.

- **Continuous Security Testing:**

  Regularly scan and test software for vulnerabilities, including penetration testing and code reviews

Vulnerability 2: Weak Authentication and Access Controls

Weak authentication controls and poor access controls make databases susceptible to unauthorized access by attackers or malicious insiders.

Techniques Used:

- Use of default, blank, or weak passwords that are easily guessable or widely known.
- Poor implementation of multi-factor authentication (MFA) or role-based access controls (RBAC), allowing unauthorized users to escalate privileges or access confidential data.
- Failing to monitor and secure user sessions, thus rendering session hijacking or abuse feasible

Impact

- **Unauthorized Data Access:**

  Attackers or intruders gain access to confidential patient, employee, or financial data.

- **Data Leakage:**

  Accidental or intentional release of sensitive data, leading to privacy violations and penalties.

- **Data Manipulation:**

  Improper users change or delete records, impacting data integrity and business credibility.

- **System Compromise:**

  Infiltrators can use compromised accounts to embed malware or continue system exploitation

Mitigation and Countermeasures

- **Strong Password Policies:**

Require strong, regularly rotated passwords and lock out default or inactive accounts.

- **Multi-Factor Authentication (MFA):**

Require a second authentication factor beyond passwords for all users, especially administrators.

- **Role-Based Access Control (RBAC):**

Assign permissions based on job roles and implement the principle of least privilege.

- **Session Management:**

Implement secure session controls like timeouts and re-authentication for privileged activities.

- **Access Auditing and Monitoring:**

Audit and monitor access logs on a regular basis to detect and respond to suspicious activity promptly