Hospital DataBase Management System

Problem Statement :

Design a database management system for a hospital that keeps track of the following information:

- Patients: patient information including personal details
- Appointments: appointment information, including the date, time, and doctor assigned.
- Doctors: doctor information, including personal details, specializations, and schedule.
- Nurses: nurse information, including personal details and schedule.
- Rooms: room information, including room number, room type, and availability.
- Prescriptions: prescription information, including the medication prescribed and dosage.

The system should also be able to handle the following operations:

- Schedule appointments and assign doctors to patients.
- Assign patients to rooms and update room availability.
- Prescribe medications and calculate the total cost of prescriptions.

The relationships between the entities include:

- Patients can have multiple Appointments, and an Appointment can have one Patient.
- Appointments can have one Doctor, and a Doctor can have multiple Appointments.
- Patients can be assigned to one Room, and a Room can have one Patient.
- Patients can have multiple Prescriptions, and a Prescription can be given to multiple Patients.
- Prescriptions can have one Doctor, and a Doctor can have multiple Prescriptions.
- Nurses can be assigned to multiple Rooms, and a Room can have multiple Nurses..
- Appointment can be assigned to one Prescriptions, and a Prescription can have one Appointment.

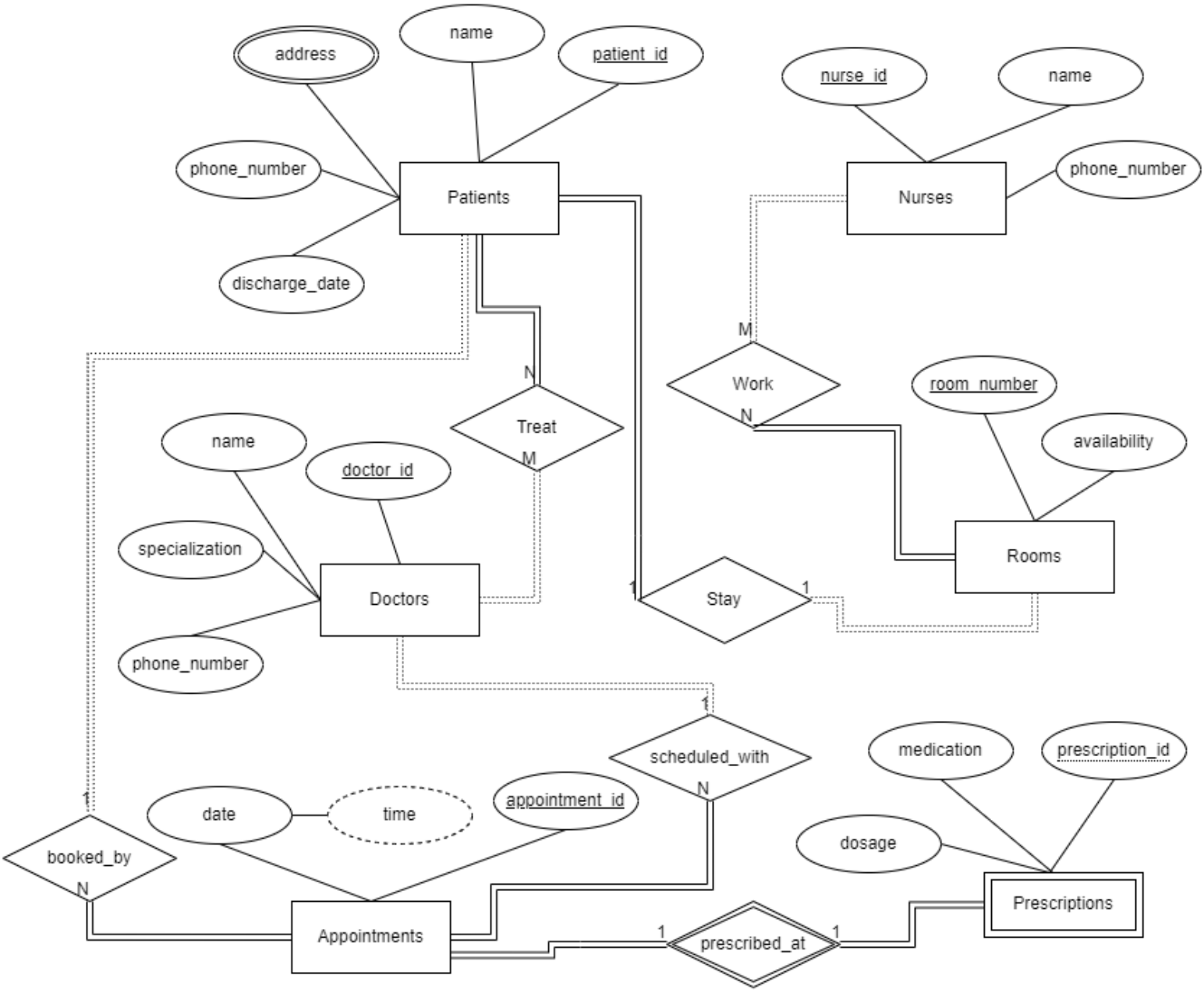
Solution :

Explanation:

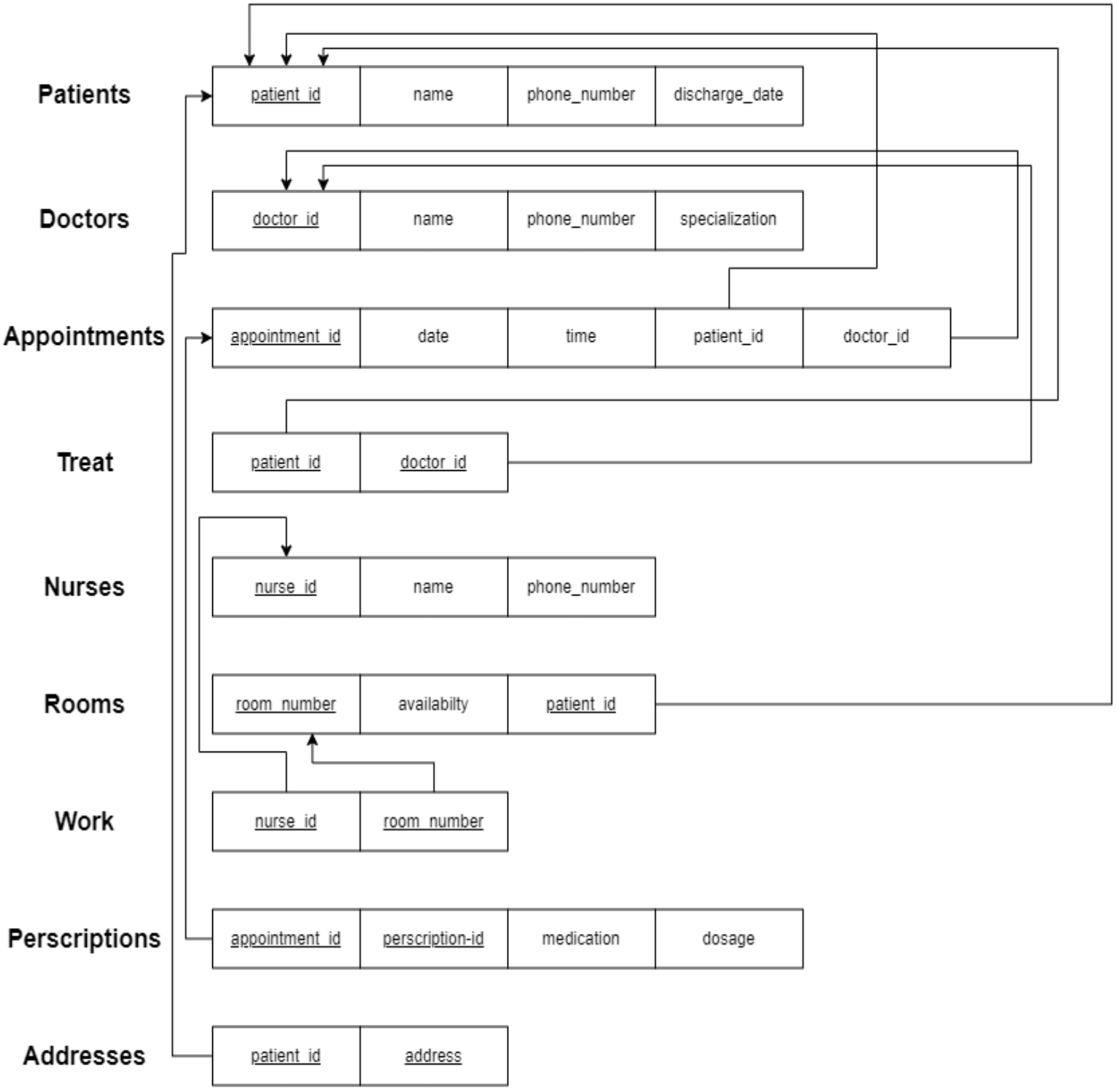
- The ER diagram includes entities for Patients, Appointments, Doctors, Nurses, Rooms and Prescriptions
- The Patients entity has attributes such as Patient ID, Name, Address, Phone Number and Discharge Date

- The Appointment entity has attributes such as Appointment ID, Date, Time, Doctor ID, and Patient ID.
- The Doctors entity has attributes such as Doctor ID, Name, Specialization, Phone Number and Schedule.
- The Nurses entity has attributes such as Nurse ID, Name, Phone Number and Schedule.
- The Rooms entity has attributes such as Room Number and Availability.
- The Prescriptions entity has attributes such as Prescription ID, Medication, Dosage and Appointment ID.

ER Diagram:


address
name
patient_id
phone_number
Patients
discharge_date
nurse_id
name
phone_number
Nurses
M
Work
N
room_number
availability
Rooms
N
Treat
M
name
doctor_id
specialization
Doctors
phone_number
1
Stay
1
1
scheduled_with
N
1
booked_by
N
date
time
appointment_id
Appointments
medication
prescription_id
dosage
Prescriptions
1
prescribed_at
1

Mapping:



Creating the Tables:

```sql
CREATE TABLE Patients (
    patient_id INT PRIMARY KEY,
    name VARCHAR(25) NOT NULL,
    phone_number VARCHAR(10) NOT NULL,
    discharge_date DATE,
);


CREATE TABLE Doctors (
    doctor_id INT PRIMARY KEY,
    name VARCHAR(25) NOT NULL,
    specialization VARCHAR(25) NOT NULL,
    phone_number VARCHAR(10) NOT NULL,
);

CREATE TABLE Appointments (
    appointment_id INT PRIMARY KEY,
    date DATE NOT NULL,
    time TIME AS (TIME(date)) PERSISTED,
    doctor_id INT NOT NULL,
    patient_id INT NOT NULL,
    FOREIGN KEY (doctor_id) REFERENCES Doctors(doctor_id),
    FOREIGN KEY (patient_id) REFERENCES Patients(patient_id)
);

CREATE TABLE Treat (
  doctor_id INT NOT NULL,
  patient_id INT NOT NULL,
  FOREIGN KEY (doctor_id) REFERENCES Doctors(doctor_id),
  FOREIGN KEY (patient_id) REFERENCES Patients(patient_id),
  PRIMARY KEY (doctor_id, patient_id)
);

CREATE TABLE Rooms (
  room_number INT PRIMARY KEY,
  availability ENUM('Yes', 'No') NOT NULL,
  patient_id INT,
  FOREIGN KEY (patient_id) REFERENCES Patients(patient_id)
);


CREATE TABLE Nurses (
  nurse_id INT PRIMARY KEY,
```

```
  name VARCHAR(25) NOT NULL,
  phone_number VARCHAR(10) NOT NULL,
);


CREATE TABLE Work (
  nurse_id INT NOT NULL,
  patient_id INT NOT NULL,
  FOREIGN KEY (nurse_id) REFERENCES Nurses(nurse_id),
  FOREIGN KEY (room_number) REFERENCES Rooms(room_number),
  PRIMARY KEY (nurse_id, room_number)
);

CREATE TABLE Prescriptions (
  appointment_id INT,
  prescription_id INT,
  medication VARCHAR(25) NOT NULL,
  dosage VARCHAR(25) NOT NULL,
  PRIMARY KEY ( prescription_id, appointment_id),
  FOREIGN KEY (appointment_id) REFERENCES Appointments(appointment_id)
);

CREATE TABLE Addresses (
  patient_id INT NOT NULL,
  address VARCHAR(25),
  PRIMARY KEY (patient_id, address),
  FOREIGN KEY (patient_id) REFERENCES Patients(patient_id)
)
```

Inserting into Tables:

```
INSERT INTO Patients VALUES(1, 'John Doe', '555-555-5555', '2022-01-01');
INSERT INTO Patients VALUES(2, 'Jane Smith', '555-555-5556', '2022-02-01');
INSERT INTO Patients VALUES(3, 'Bob Johnson', '555-555-5557', '2022-03-
01');

INSERT INTO Doctors VALUES(1, 'Dr. Smith', 'Surgeon', '555-555-5558');
INSERT INTO Doctors VALUES(2, 'Dr. Johnson', 'Pediatrician', '555-555-
5559');
INSERT INTO Doctors VALUES(3, 'Dr. Williams', 'Cardiologist', '555-555-
5560');
```

```
INSERT INTO Appointments VALUES(1, '2022-01-01', '09:00:00', 1, 1);
INSERT INTO Appointments VALUES(2, '2022-01-02', '10:00:00', 2, 2);
INSERT INTO Appointments VALUES(3, '2022-01-03', '11:00:00', 3, 3);

INSERT INTO Treat VALUES(1, 1);
INSERT INTO Treat VALUES(2, 2);
INSERT INTO Treat VALUES(3, 3);

INSERT INTO Nurses VALUES(1, 'Nurse Jane', '555-555-5561');
INSERT INTO Nurses VALUES(2, 'Nurse Bob', '555-555-5562');
INSERT INTO Nurses VALUES(3, 'Nurse Mary', '555-555-5563');

INSERT INTO Rooms VALUES(101, 'Yes', 1);
INSERT INTO Rooms VALUES(102, 'No', NULL);
INSERT INTO Rooms VALUES(103, 'Yes', 3);

INSERT INTO Work VALUES(1, 101);
INSERT INTO Work VALUES(2, 102);
INSERT INTO Work VALUES(3, 103);

INSERT INTO Prescriptions VALUES(1, 1, 'Ibuprofen', '2 tablets 3 times a
day');
INSERT INTO Prescriptions VALUES(2, 2, 'Amoxicillin', '1 tablet 2 times a
day');
INSERT INTO Prescriptions VALUES(3, 3, 'Metformin', '1 tablet 3 times a
day');

INSERT INTO Addresses VALUES(1, '123 Main St.');
INSERT INTO Addresses VALUES(2, '456 Park Ave.');
INSERT INTO Addresses VALUES(3, '789 Elm St.');
INSERT INTO Addresses VALUES(1, '435 Main St');
INSERT INTO Addresses VALUES(2, '526 Park Ave');
INSERT INTO Addresses VALUES(3, '459 Elm St');
```

Queries:

1. Retrieve the names of all patients who have an appointment scheduled with a doctor who specializes in "Cardiology" on the date "2022-05-15":

```
SELECT Patients.name
```

```
FROM Patients
JOIN Appointments ON Patients.patient_id = Appointments.patient_id
JOIN Doctors ON Appointments.doctor_id = Doctors.doctor_id
WHERE Appointments.date = '2022-05-15' AND Doctors.specialization =
  'Cardiology';
```

2. Retrieve the phone numbers of all patients who are currently hospitalized:

```
SELECT Patients.phone_number
FROM Patients
JOIN Rooms ON Patients.patient_id = Rooms.patient_id
WHERE Rooms.availability = 'No';
```

3. Retrieve the names of all patients who are currently hospitalized and are being treated by a nurse with ID = 3:

```
SELECT Patients.name
FROM Patients
JOIN Rooms ON Patients.patient_id = Rooms.patient_id
JOIN Work ON Rooms.room_number = Work.room_number
WHERE Rooms.availability = 'No' AND Work.nurse_id = 3;
```

4. What is the medication prescribed for patients with ID 123 in their last appointment?

```
SELECT Prescriptions.medication
FROM Prescriptions
JOIN Appointments ON Prescriptions.appointment_id =
  Appointments.appointment_id
WHERE Appointments.patient_id = 123
ORDER BY Appointments.date DESC
LIMIT 1;
```

5. Retrieve the average salary of all nurses who work in room number 101:

```
SELECT AVG(salary)
FROM Nurses
JOIN Work ON Nurses.nurse_id = Work.nurse_id
JOIN Rooms ON Work.room_number = Rooms.room_number
WHERE Rooms.room_number = 101;
```

6. Retrieve the name, phone number, and specialization of all doctors who treated more than 2 patients in the month of January:

```
SELECT name, phone_number, specialization
FROM Doctors
JOIN Treat ON Doctors.doctor_id = Treat.doctor_id
JOIN Appointments ON Treat.patient_id = Appointments.patient_id
WHERE MONTH(date) = 1
GROUP BY Doctors.doctor_id
HAVING COUNT(DISTINCT Treat.patient_id) > 2;
```

7. Write a query that returns the name and specialization of all doctors who have treated patients who have been discharged and have a phone number that contains the string "555".

```
SELECT Rooms.room_type, COUNT(DISTINCT Patients.patient_id) total_patients,
  AVG(DATEDIFF(discharge_date, admit_date)) avg_length_of_stay
FROM Patients
INNER JOIN Rooms ON Patients.patient_id = Rooms.patient_id
WHERE discharge_date IS NOT NULL
GROUP BY Rooms.room_type;
```

8. What are the unique addresses of patients who have had a prescription for a specific medication?

```
SELECT DISTINCT address FROM Addresses
WHERE patient_id IN (SELECT patient_id FROM Prescriptions
WHERE medication = 'DOLO');
```

9. Write a query that gets the earliest and latest appointment dates, as well as the total patient count for appointments where the patient is currently not discharged and for appointments where the patient is discharged.

```
SELECT MIN(date) as earliest_appointment, MAX(date) as latest_appointment,
SUM(patient_count) as total_patient_count
FROM
  (SELECT date, COUNT(patient_id) as patient_count
  FROM Appointments
  WHERE EXISTS (SELECT 1 FROM Patients WHERE Patients.patient_id =
Appointments.patient_id AND discharge_date IS NULL)
  GROUP BY date
```

```
  UNION
  SELECT date, COUNT(patient_id) as patient_count FROM Appointments
  WHERE NOT EXISTS (SELECT 1 FROM Patients WHERE Patients.patient_id =
Appointments.patient_id AND discharge_date IS NULL)
GROUP BY date) as appts;
```

Stored Procedures:

1. A stored procedure to retrieve all the appointments for a specific patient:

```
CREATE PROCEDURE get_patient_appointments (IN patient_id INT) IS
X Appointments%rowtype;
CURSOR C IS
    SELECT a.appointment_id, a.date, a.time
    FROM Appointments a
    JOIN Treat t ON a.patient_id = t.patient_id
    JOIN Doctors d ON t.doctor_id = d.doctor_id
    WHERE a.patient_id = patient_id;
BEGIN
    FOR X IN C LOOP
        SYS.DBMS_OUTPUT.PUT_LINE(X.appointment_id || ' ' || X.date || ' '
|| X.time);
    END LOOP;
END;
/
```

2. Procedure to update the phone number of a patient and all related doctors and nurses:

```
CREATE PROCEDURE update_patient_phone_number (IN patient_id INT, IN
phone_number VARCHAR(10)) IS
X Doctors%rowtype;
Y Nurses%rowtype;
DECLARE doctor_id INT;
DECLARE nurse_id INT;
CURSOR cur1 IS
    SELECT doctor_id
    FROM Treat
    WHERE patient_id = patient_id;
CURSOR cur2 IS
    SELECT nurse_id
```

```
    FROM Work WHERE
    patient_id = patient_id;

UPDATE Patients SET phone_number = phone_number WHERE patient_id =
patient_id;
BEGIN
    FOR X IN cur1 LOOP
        UPDATE Doctors SET phone_number = phone_number WHERE X.doctor_id =
doctor_id;
    END LOOP;
    FOR Y IN cur2 LOOP
        UPDATE Nurses SET phone_number = phone_number WHERE Y.nurse_id =
nurse_id;
    END LOOP;
END;
/
```

3. Procedure to check if a patient has a room assigned, and if not, assign them one:

```
CREATE PROCEDURE assign_room(IN patient_id INT) IS
X Rooms%rowtype;
DECLARE room_number INT;
DECLARE cur_patient_id INT;
DECLARE done INT DEFAULT FALSE;
CURSOR cur IS
    SELECT patient_id FROM Rooms WHERE availability = 'Yes' ORDER BY
room_number LIMIT 1;
BEGIN
    OPEN cur;
    FETCH cur INTO room_number;
    IF done THEN
        SYS.DBMS_OUTPUT.PUT_LINE'Sorry, all rooms are currently
occupied.');
    ELSE
        UPDATE Rooms SET patient_id = patient_id, availability = 'No' WHERE
room_number = room_number;
        SYS.DBMS_OUTPUT.PUT_LINE('Room number ', room_number, ' has been
assigned to patient ID ', patient_id);
    END IF;
    CLOSE cur;
END;
/
```

Triggers:

1. Trigger to update the availability of a room when a patient is discharged:

```
CREATE TRIGGER update_room_availability
AFTER UPDATE ON Patients
FOR EACH ROW
BEGIN
DECLARE room_number INT;
DECLARE availability VARCHAR(25);

IF OLD.discharge_date IS NOT NULL AND NEW.discharge_date IS NULL THEN
    SET room_number = (SELECT room_number FROM Rooms WHERE patient_id =
OLD.patient_id);
    SET availability = 'No';
    UPDATE Rooms SET availability = availability WHERE room_number =
room_number;
END IF;
END;
/
```

2. Trigger to prevent patients from being discharged if they have an upcoming appointment:

```
CREATE TRIGGER prevent_discharge
BEFORE UPDATE ON Patients
FOR EACH ROW
BEGIN
DECLARE patient_id INT;
DECLARE discharge_date DATE;

SET patient_id = NEW.patient_id;
SET discharge_date = NEW.discharge_date;

IF discharge_date IS NOT NULL THEN
    IF EXISTS (SELECT appointment_id FROM Appointments WHERE patient_id =
patient_id AND date > discharge_date) THEN
        RAISE_APPLICATION_ERROR(-20009,'Patient has an upcoming appointment
and cannot be discharged.');
    END IF;
END IF;
END;
```

```
/
```

3. A trigger to prevent a doctor from being deleted if they have ongoing appointments:

```
CREATE TRIGGER prevent_doctor_deletion
BEFORE DELETE ON Doctors
FOR EACH ROW
BEGIN
    DECLARE appointment_count INT;
    SELECT COUNT(*) INTO appointment_count
    FROM Appointments
    WHERE doctor_id = OLD.doctor_id;
    IF appointment_count > 0 THEN
         RAISE_APPLICATION_ERROR(-20009,'Cannot delete a doctor with
ongoing appointments');
    END IF;
END;
/
```