DATABASE MANAGEMENT SYSTEM – CSA0593

ASSIGNMENT 2

R. YASHWANTH VARMA

192311392

QUESTION:

Design a database to manage patients, doctors, appointments, and medical records.

- Model tables for patients, doctors, appointments, and medical records.

- Write stored procedures for booking, rescheduling, and cancelling appointments.

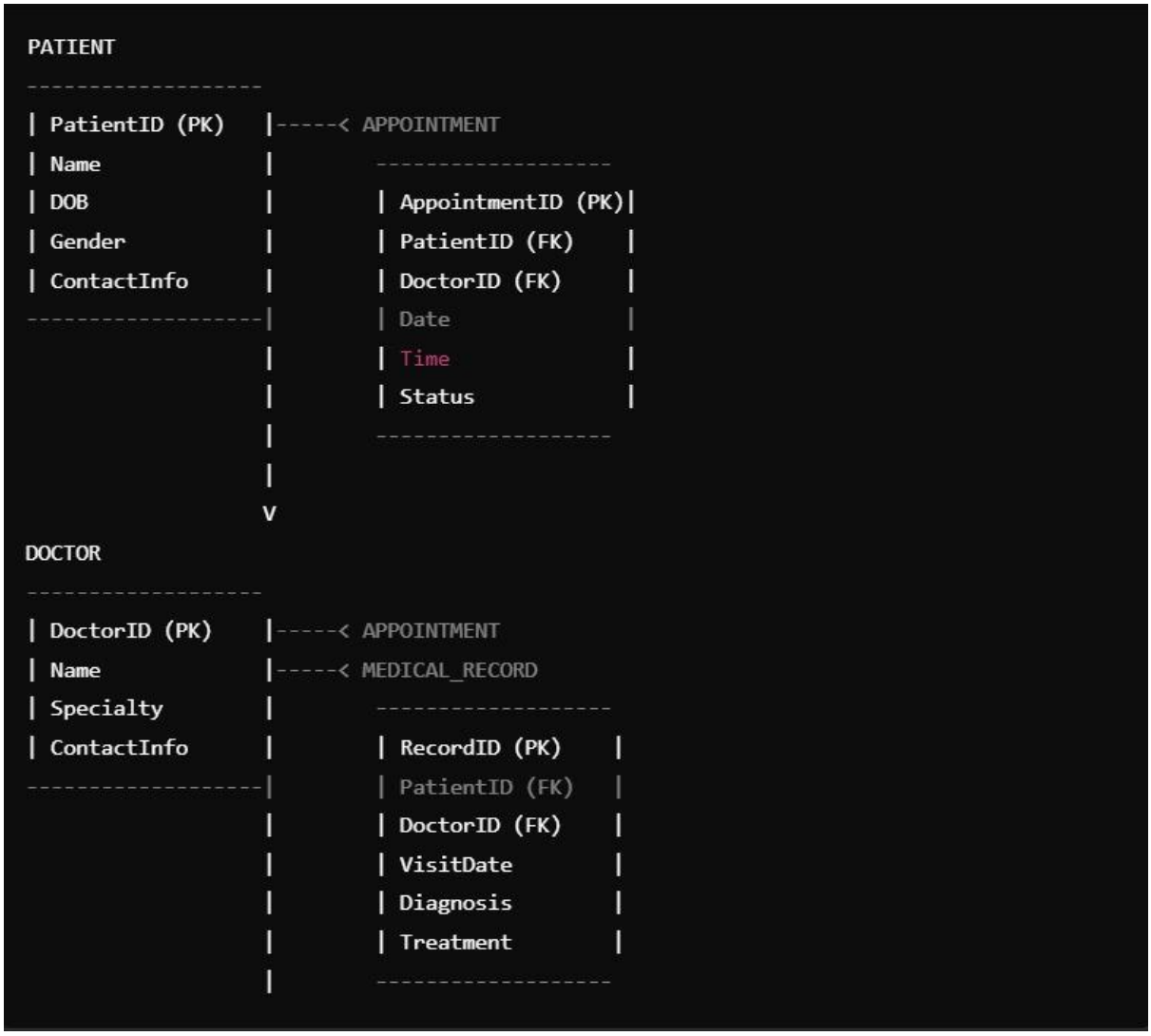
- Implement triggers to update appointment availability and patient records.

- Write SQL queries to analyse patient visit statistics and doctor workload.

ANSWER:

CONCEPTUAL ER DIAGRAM:

LOGICAL ER DIAGRAM:



PHYSICAL ER DIAGRAM:

SQL:

Database Schema:

Mysql

```
CREATE DATABASE MedicalManagement;



USE MedicalManagement;



CREATE TABLE Patients (
  PatientID INT AUTO_INCREMENT PRIMARY KEY,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  DateOfBirth DATE,
  ContactNumber VARCHAR(20)
);



CREATE TABLE Doctors (
  DoctorID INT AUTO_INCREMENT PRIMARY KEY,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  Specialty VARCHAR(100)
);
```

```
CREATE TABLE Appointments (
  AppointmentID INT AUTO_INCREMENT PRIMARY KEY,
  PatientID INT,
  DoctorID INT,
  AppointmentDate DATE,
  AppointmentTime TIME,
  Status VARCHAR(50),
  FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
  FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID)
);


CREATE TABLE MedicalRecords (
  RecordID INT AUTO_INCREMENT PRIMARY KEY,
  PatientID INT,
  DoctorID INT,
  VisitDate DATE,
  Diagnosis VARCHAR(255),
  Treatment VARCHAR(255),
  FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
  FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID)
);
```

Stored Procedures:

mysql

```
DELIMITER //

CREATE PROCEDURE sp_BookAppointment(
  IN patientID INT,
  IN doctorID INT,
  IN appointmentDate DATE,
  IN appointmentTime TIME
)

BEGIN
  INSERT INTO Appointments (PatientID, DoctorID, AppointmentDate, AppointmentTime, Status)
  VALUES (patientID, doctorID, appointmentDate, appointmentTime, 'Booked');
END //


CREATE PROCEDURE sp_RescheduleAppointment(
  IN appointmentID INT,
  IN newAppointmentDate DATE,
  IN newAppointmentTime TIME
)
```

```
BEGIN
  UPDATE Appointments
  SET AppointmentDate = newAppointmentDate,
      AppointmentTime = newAppointmentTime
  WHERE AppointmentID = appointmentID;
END //

CREATE PROCEDURE sp_CancelAppointment(
  IN appointmentID INT
)

BEGIN
  UPDATE Appointments
  SET Status = 'Cancelled'
  WHERE AppointmentID = appointmentID;
END //

DELIMITER;
```

Triggers:

```
mysql
DELIMITER //

CREATE TRIGGER tr_UpdateAppointmentAvailability
AFTER INSERT ON Appointments
FOR EACH ROW
```

```
BEGIN
  UPDATE Doctors
  SET Availability = 'Unavailable'
  WHERE DoctorID = NEW.DoctorID;
END //

CREATE TRIGGER tr_UpdatePatientRecords
AFTER INSERT ON MedicalRecords
FOR EACH ROW
BEGIN
  UPDATE Patients
  SET LastVisitDate = NEW.VisitDate
  WHERE PatientID = NEW.PatientID;
END //

DELIMITER;
```

SQL Queries:

```
mysql
-- Patient Visit Statistics
SELECT
  Patients.PatientID,
  Patients.FirstName,
  Patients.LastName,
  COUNT(*) AS TotalVisits
```

```
FROM
  Patients
  JOIN MedicalRecords ON Patients.PatientID = MedicalRecords.PatientID
GROUP BY
  Patients.PatientID, Patients.FirstName, Patients.LastName;

-- Doctor Workload
SELECT
  Doctors.DoctorID,
  Doctors.FirstName,
  Doctors.LastName,
  COUNT(*) AS TotalAppointments
FROM
  Doctors
  JOIN Appointments ON Doctors.DoctorID = Appointments.DoctorID
GROUP BY
  Doctors.DoctorID, Doctors.FirstName, Doctors.LastName;
```

Conclusion:

This database design provides a comprehensive foundation for managing patients, doctors, appointments, and medical records. The stored procedures simplify appointment booking, rescheduling, and cancellation, while the triggers ensure data consistency and accuracy. The SQL queries enable analysis of patient visit statistics and doctor workload.