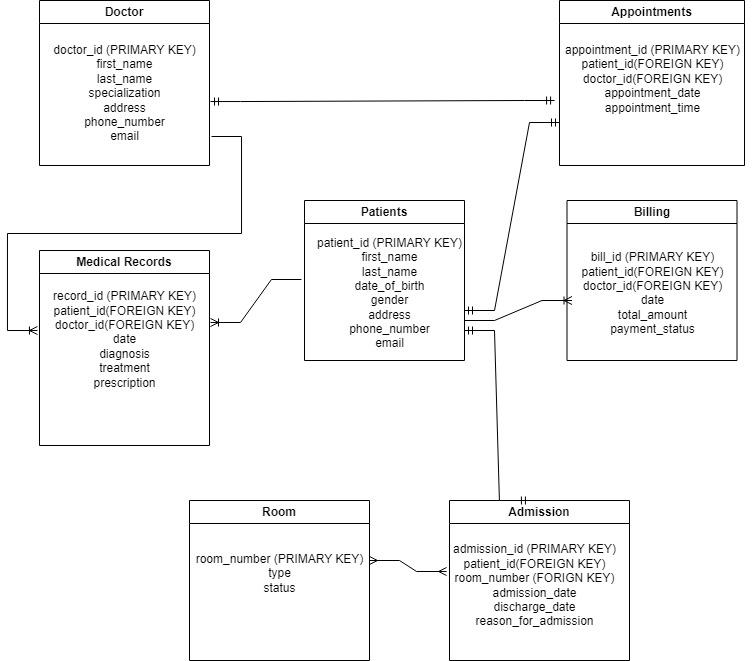
**Create a DB Schema for Hospital Management System.**

**Define the schema along with the constraints indicating the relationships between the entities.**

****

**Patients(**patient\_id (Primary Key),first\_name,last\_name,date\_of\_birth,gender

,address,phone\_number,email)

Patients patient\_id refers Appointments patient\_id,

Patients patient\_id refers Medical Records patient\_id,

Patients patient\_id refers Billing patient\_id,

Patients patient\_id refers Admissions patient\_id

**Doctors (**doctor\_id (Primary Key),first\_name,last\_name,date\_of\_birth,gender, address,

phone\_number,email)

Doctors doctor\_id refers Appointments doctor\_id,

Doctors doctor\_id refers Medical Records doctor\_id,

**Appointments(**appointment\_id,patient\_id,doctor\_id ,appointment\_date,appointment\_time)

Appointments patient\_id(FK) refers Patient patient\_id,

Appointments patient\_id(FK) refers Doctor doctor\_id,

**Medical Records(**record\_id,patient\_id,doctor\_id,date,diagnosis,treatment,prescription,

prescription)

Medical Recordspatient\_id(FK) refers Patient patient\_id,

Medical Records patient\_id(FK) refers Doctor doctor\_id,

**Billing table(**bill\_id ,patient\_id ,doctor\_id,date,total\_amount,payment\_status)

Billing patient\_id(FK) refers Patient patient\_id,

Billing patient\_id(FK) refers Doctor doctor\_id,

**Rooms(**room\_number(PK), type, status)

Rooms room\_number(PK) refers admission room\_number

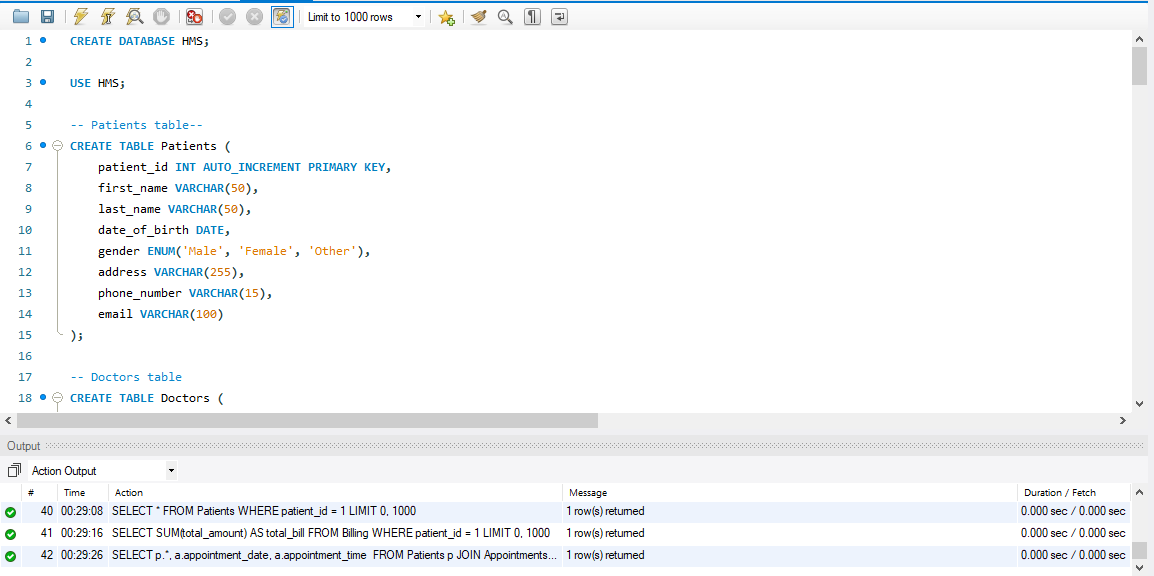
**Admissions (**admission\_id,patient\_id,room\_number,

admission\_date,discharge\_date,reason\_for\_admission)

Admissions patient\_id(FK) refers Patient patient\_id,

Admissions room\_number refer Rooms room\_number,

// CREATION OF DATABASE



**CREATE DATABASE HMS;**

**USE HMS;**

**-- Patients table--**

CREATE TABLE Patients (

patient\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

date\_of\_birth DATE,

gender ENUM('Male', 'Female', 'Other'),

address VARCHAR(255),

phone\_number VARCHAR(15),

email VARCHAR(100)

);

**-- Doctors table**

CREATE TABLE Doctors (

doctor\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

specialization VARCHAR(100),

address VARCHAR(255),

phone\_number VARCHAR(15),

email VARCHAR(100)

);

**-- Appointments table**

CREATE TABLE Appointments (

appointment\_id INT AUTO\_INCREMENT PRIMARY KEY,

patient\_id INT,

doctor\_id INT,

appointment\_date DATE,

appointment\_time TIME,

FOREIGN KEY (patient\_id) REFERENCES Patients(patient\_id),

FOREIGN KEY (doctor\_id) REFERENCES Doctors(doctor\_id)

);

**-- Medical Records table**

CREATE TABLE Medical\_Records (

record\_id INT AUTO\_INCREMENT PRIMARY KEY,

patient\_id INT,

doctor\_id INT,

date DATE,

diagnosis VARCHAR(255),

treatment VARCHAR(255),

prescription VARCHAR(255),

FOREIGN KEY (patient\_id) REFERENCES Patients(patient\_id),

FOREIGN KEY (doctor\_id) REFERENCES Doctors(doctor\_id)

);

**-- Billing table**

CREATE TABLE Billing (

bill\_id INT AUTO\_INCREMENT PRIMARY KEY,

patient\_id INT,

doctor\_id INT,

date DATE,

total\_amount DECIMAL(10, 2),

payment\_status ENUM('Pending', 'Paid'),

FOREIGN KEY (patient\_id) REFERENCES Patients(patient\_id),

FOREIGN KEY (doctor\_id) REFERENCES Doctors(doctor\_id)

);

**-- Rooms table**

CREATE TABLE Rooms (

room\_number INT AUTO\_INCREMENT PRIMARY KEY,

type VARCHAR(50),

status ENUM('Occupied', 'Vacant')

);

**-- Admissions table**

CREATE TABLE Admissions (

admission\_id INT AUTO\_INCREMENT PRIMARY KEY,

patient\_id INT,

room\_number INT,

admission\_date DATE,

discharge\_date DATE,

reason\_for\_admission VARCHAR(255),

FOREIGN KEY (patient\_id) REFERENCES Patients(patient\_id),

FOREIGN KEY (room\_number) REFERENCES Rooms(room\_number)

);

**SHOW TABLES;**

**-- Insert data into Appointments table**

INSERT INTO Appointments (patient\_id, doctor\_id, appointment\_date, appointment\_time)

VALUES (1, 1, '2024-02-21', '10:00:00');

**-- Insert data into Billing table**

INSERT INTO Billing (patient\_id, doctor\_id, date, total\_amount, payment\_status)

VALUES (1, 1, '2024-02-21', 100.00, 'Pending');

**-- Insert data into Rooms table**

INSERT INTO Rooms (type, status)

VALUES ('Private', 'Vacant');

**-- Insert data into Admissions table**

INSERT INTO Admissions (patient\_id, room\_number, admission\_date, discharge\_date, reason\_for\_admission)

VALUES (1, 1, '2024-02-21', '2024-02-25', 'High fever');

**-- - 1 Write necessary queries to register new user roles and personas**

**-- Registering a new patient**

INSERT INTO Patients (first\_name, last\_name, date\_of\_birth, gender, address, phone\_number, email)

VALUES ('Deepak', 'Mishra', '2003-09-02', 'Male', 'Mharal, Kalyan', '125-678-9012', 'deepakmishra@gmail.com');



**-- Registering a new doctor**

INSERT INTO Doctors (first\_name, last\_name, specialization, address, phone\_number, email)

VALUES ('Dr. Satyam', '', 'Mishra', 'Varap, Kalyan', '987-654-3210', 'satyammishra@gmail.com');



**-- - 2 Write necessary queries to add to the list of diagnosis of the patient tagged by date.**

INSERT INTO Medical\_Records (patient\_id, doctor\_id, date, diagnosis, treatment, prescription)

VALUES (1, 1, '2024-02-21', 'Fever', 'Rest and fluids', 'Paracetamol');



**-- - 3 Write necessary queries to fetch required details of a particular patient.**

SELECT \* FROM Patients WHERE patient\_id = 1;



**-- - 4 Write necessary queries to prepare bill for the patient at the end of checkout.**

SELECT patient\_id, SUM(total\_amount) AS total\_bill FROM Billing WHERE patient\_id = 1;



**-- - 5 Write necessary queries to fetch and show data from various related tables (Joins)**

SELECT p.\*, a.appointment\_date, a.appointment\_time

FROM Patients p

JOIN Appointments a ON p.patient\_id = a.patient\_id

WHERE p.patient\_id = 1;



**-- - 6 Optimize repeated read operations using views/materialized views.**

CREATE VIEW Patient\_Appointment\_Details AS

SELECT p.\*, a.appointment\_date, a.appointment\_time

FROM Patients p

JOIN Appointments a ON p.patient\_id = a.patient\_id;



**-- - 7 Optimize read operations using indexing wherever required. (Create index on at least 1 table)**

CREATE INDEX idx\_patient\_id ON Patients(patient\_id);

**-- - 8 Try optimizing bill generation using stored procedures.**

DELIMITER //

CREATE PROCEDURE Generate\_Bill(IN patient\_id INT)

BEGIN

DECLARE total DECIMAL(10, 2);

SELECT SUM(total\_amount) INTO total FROM Billing WHERE patient\_id = patient\_id;

SELECT total;

END //

DELIMITER ;

CALL Generate\_Bill(1);



**-- - 9 Add necessary triggers to indicate when patients medical insurance limit has expired.**

DELIMITER //

CREATE TRIGGER Insurance\_Expiration

BEFORE INSERT ON Admissions

FOR EACH ROW

BEGIN

IF NEW.admission\_date < DATE\_SUB(NOW(), INTERVAL 1 YEAR) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Patient\'s medical insurance limit has expired';

END IF;

END //

DELIMITER ;

Q2) Write a report on your understanding of Rendering and Design Patterns. Mention and elaborate where a particular Rendering pattern is applicable and is well suited for which use case.

**Ans :-**

**Introduction:**

Rendering is a crucial aspect of software development, particularly in fields like computer graphics, game development, and user interface design. It involves the process of generating an image from a model by means of computer software.

**Rendering Techniques:**

**•Immediate Mode Rendering:**

Rendering commands are issued directly to the graphics hardware for immediate execution. It is suitable for applications requiring real-time interaction and responsiveness, such as video games and simulations.

**•Retained Mode Rendering:**

This technique involves constructing a scene graph or retaining a representation of the scene in memory, which is then rendered to the screen. Retained mode rendering is commonly used in graphical user interfaces (GUIs) and applications where scene complexity is high and pre-processing can improve rendering performance.

**Design Patterns in Rendering:**

**Factory Method Pattern:**

This pattern is used to create objects without specifying the exact class of the object to be created. In rendering, the factory method pattern can be applied to create different types of rendering objects (e.g., renderers, shaders) based on specific requirements or configurations.

**Composite Pattern:**

The composite pattern is used to represent hierarchical structures as a tree of objects. In rendering, the composite pattern can be employed to represent complex scenes composed of multiple objects and sub-objects, enabling efficient rendering and manipulation of the scene graph.

**Observer Pattern:**

This pattern is used to establish a one-to-many dependency between objects, where changes in one object trigger updates in dependent objects. In rendering, the observer pattern can be utilized to implement event-driven rendering systems, where changes in the scene trigger updates in the rendering pipeline.

**Conclusion :-**

rendering patterns play a vital role in shaping the design and implementation of rendering systems, enabling developers to tackle challenges effectively and deliver compelling visual experiences across a wide range of applications.