

1) Create a DB Schema for Hospital Management System.

CREATE DATABASE IF NOT EXISTS hospital\_management\_system;

USE hospital\_management\_system;

2) Schema of all entity

**Patient Entity**

CREATE TABLE IF NOT EXISTS Patient (

Patient\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Name VARCHAR(100),

Date\_of\_Birth DATE,

Gender VARCHAR(10),

Contact\_Number VARCHAR(20),

Address VARCHAR(255),

Insurance\_Limit DECIMAL(10, 2),

Insurance\_Expired INT DEFAULT 0

);

Attributes: Patient\_ID (Primary Key), Name, Date\_of\_Birth, Gender, Contact\_Number, Address, Insurance\_Limit, Insurance\_Expired

Key Points:

Patient\_ID is the primary key.

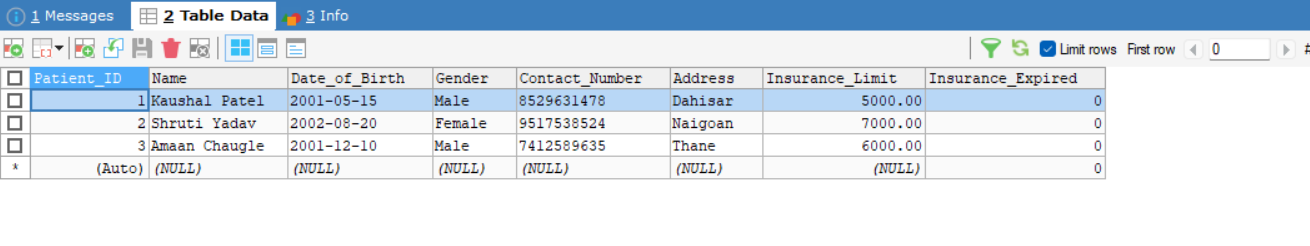
Insurance\_Limit should be a positive value.

>INSERT INTO Patient (Name, Date\_of\_Birth, Gender, Contact\_Number, Address, Insurance\_Limit) VALUES

('Kaushal Patel', '2001-05-15', 'Male', '8529631475', 'Dahisar', 5000.00),

('Shruti Yadav', '2002-08-20', 'Female', '9517538524', 'Naigoan', 7000.00),

('Amaan Chaugle', '2001-12-10', 'Male', '7412589635', 'Thane', 6000.00);



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Doctor Entity**

CREATE TABLE IF NOT EXISTS Doctor (

Doctor\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Name VARCHAR(100),

Specialization VARCHAR(100),

Contact\_Number VARCHAR(20),

Email VARCHAR(100),

Department\_ID INT,

Role\_ID INT,

FOREIGN KEY (Department\_ID) REFERENCES Department(Department\_ID),

FOREIGN KEY (Role\_ID) REFERENCES UserRoles(Role\_ID)

);

Attributes: Doctor\_ID (Primary Key), Name, Specialization, Contact\_Number, Email, Department\_ID (Foreign Key)

Key Points:

Doctor\_ID is the primary key.

Department\_ID is a foreign key referencing the Department table.

Role\_ID is a foreign key referencing the UderRoles table.

Relationships: Many-to-One relationship with Departments.

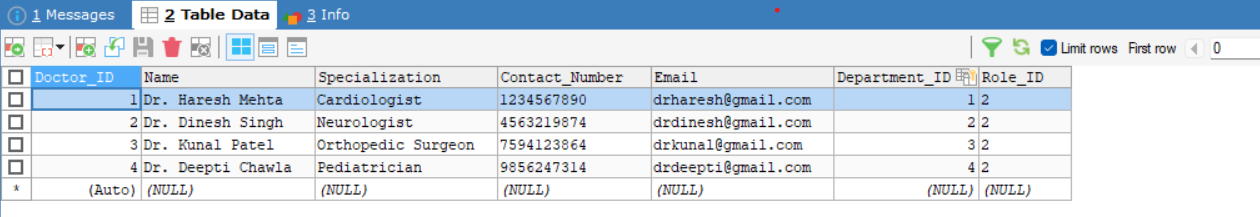
>INSERT INTO Doctor (NAME, Specialization, Contact\_Number, Email, Department\_ID, Role\_ID) VALUES

('Dr. Haresh Mehta', 'Cardiologist', '1234567890', 'drharesh@gmail.com', 1, 2),

('Dr. Dinesh Singh', 'Neurologist', '4563219874', 'drdinesh@gmail.com', 2, 2),

('Dr. Kunal Patel', 'Orthopedic Surgeon', '7594123864', 'drkunal@gmail.com', 3, 2),

('Dr. Deepti Chawla', 'Pediatrician', '9856247314', 'drdeepti@gmail.com', 4, 2);



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Department Entity**

CREATE TABLE IF NOT EXISTS Department (

Department\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Department\_Name VARCHAR(100)

);

Attributes: Department\_ID (Primary Key), Department\_Name

Key Points:

Department\_ID is the primary key.

Relationships: One-to-Many relationship with Doctors.

>INSERT INTO Department (Department\_Name) VALUES

('Cardiology'),

('Neurology'),

('Orthopedics'),

(‘Pediatrics’);



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Disease Entity**

CREATE TABLE IF NOT EXISTS Disease (

Disease\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Disease\_Name VARCHAR(100),

Description TEXT,

Treatment TEXT

);

Attributes: Disease\_ID (Primary Key), Disease\_Name, Description, Treatment

Key Points:

Disease\_ID is the primary key.

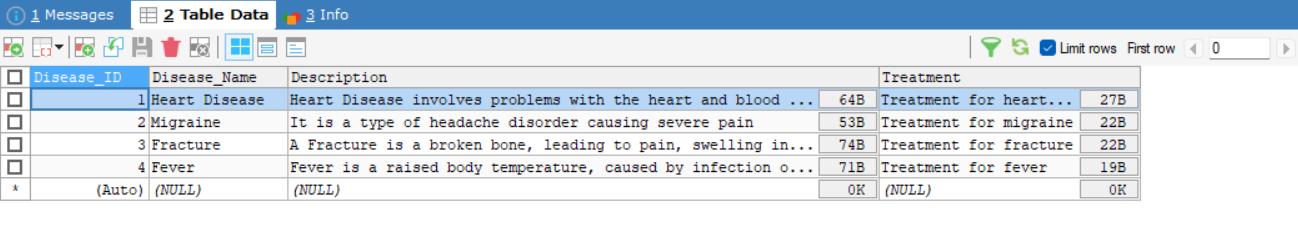
>INSERT INTO Diseases (Disease\_Name, Description, Treatment) VALUES

('Heart Disease', Heart Disease involves problems with the heart and blood vessels ', 'Treatment for heart disease'),

('Migraine', It is a type of headache disorder causing severe pain ', 'Treatment for migraine'),

('Fracture', A Fracture is a broken bone, leading to pain, swelling in the injured area ', 'Treatment for fracture'),

('Fever', ' Fever is a raised body temperature, caused by infection or inflammation, 'Treatment for fever');

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Appointment Entity**

CREATE TABLE IF NOT EXISTS Appointment (

Appointment\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Patient\_ID INT,

Doctor\_ID INT,

Appointment\_Date DATE,

Appointment\_Time TIME,

Appointment\_Status VARCHAR(20),

Disease\_ID INT,

FOREIGN KEY (Patient\_ID) REFERENCES Patient(Patient\_ID),

FOREIGN KEY (Doctor\_ID) REFERENCES Doctor(Doctor\_ID),

FOREIGN KEY (Disease\_ID) REFERENCES Disease(Disease\_ID)

);

Attributes: Appointment\_ID (Primary Key), Patient\_ID (Foreign Key), Doctor\_ID (Foreign Key), Appointment\_Date, Appointment\_Time, Appointment\_Status, Disease\_ID (Foreign Key)

KeyPoints:

Appointment\_ID is the primary key.

Patient\_ID and Doctor\_ID are foreign keys referencing the Patients and Doctors tables respectively.

Relationships:

Many-to-One relationship with Patients and Doctors.

Many-to-One relationship with Diseases.

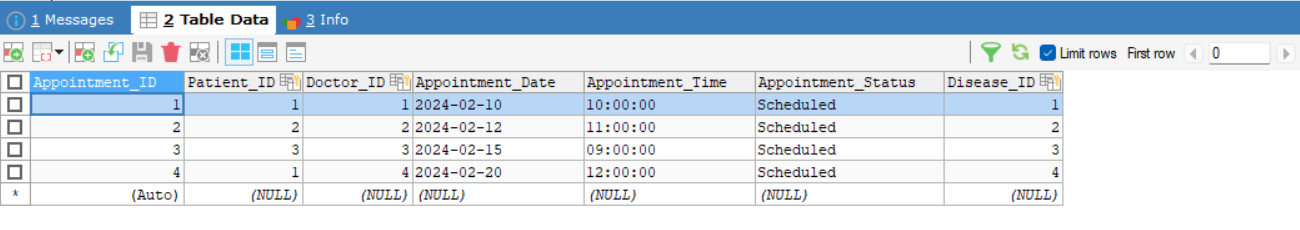
>INSERT INTO Appointments (Patient\_ID, Doctor\_ID, Appointment\_Date, Appointment\_Time, Appointment\_Status, Disease\_ID) VALUES

(1, 1, '2024-02-10', '10:00:00', 'Scheduled', 1),

(2, 2, '2024-02-12', '11:00:00', 'Scheduled', 2),

(3, 3, '2024-02-15', '09:00:00', 'Scheduled', 3),

(1, 4, '2024-02-20', '12:00:00', 'Scheduled', 4);



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Billing Entity**

CREATE TABLE IF NOT EXISTS Billing (

Bill\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Patient\_ID INT,

Appointment\_ID INT,

Disease\_ID INT,

Total\_Amount DECIMAL(10, 2),

Payment\_Status VARCHAR(20),

Billing\_Date DATE,

FOREIGN KEY (Patient\_ID) REFERENCES Patient(Patient\_ID),

FOREIGN KEY (Appointment\_ID) REFERENCES Appointment(Appointment\_ID),

FOREIGN KEY (Disease\_ID) REFERENCES Disease(Disease\_ID)

);

Attributes: Bill\_ID (Primary Key), Patient\_ID (Foreign Key), Appointment\_ID (Foreign Key), Disease\_ID (Foreign Key), Total\_Amount, Payment\_Status, Billing\_Date

Key Points:

Bill\_ID is the primary key.

Patient\_ID, Appointment\_ID, and Disease\_ID are foreign keys referencing Patients, Appointments, and Diseases tables respectively.

Relationships:

Many-to-One relationship with Patients, Appointments, and Diseases.

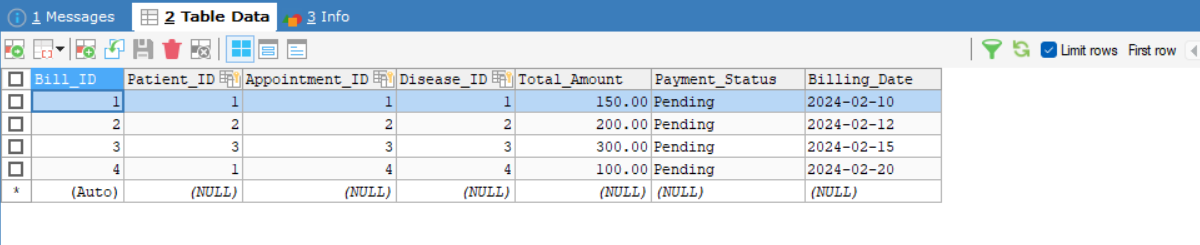
> INSERT INTO Billing (Patient\_ID, Appointment\_ID, Disease\_ID, Total\_Amount, Payment\_Status, Billing\_Date) VALUES

(1, 1, 1, 150.00, 'Pending', '2024-02-10'),

(2, 2, 2, 200.00, 'Pending', '2024-02-12'),

(3, 3, 3, 300.00, 'Pending', '2024-02-15'),

(1, 4, 4, 100.00, 'Pending', '2024-02-20');



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**UserRoles Entity**

CREATE TABLE IF NOT EXISTS UserRoles (

Role\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Role\_Name VARCHAR(100) UNIQUE

);

The User Role table defines different roles that users can have within the system.Examples of user roles in an HMS may include "Admin", "Doctor", "Nurse", "Receptionist", etc.

Each user role represents a set of permissions and access levels within the system.

> INSERT INTO UserRoles (Role\_Name) VALUES

('Administrator'),

('Doctor'),

('Nurse'),

('Receptionist');



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**UserPersonas Entity**

CREATE TABLE IF NOT EXISTS UserPersonas (

User\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Role\_ID INT,

Name VARCHAR(100),

Contact\_Number VARCHAR(20),

Email VARCHAR(100),

FOREIGN KEY (Role\_ID) REFERENCES UserRoles(Role\_ID)

);

The User Persona table defines specific personas or profiles that users can have within a role.

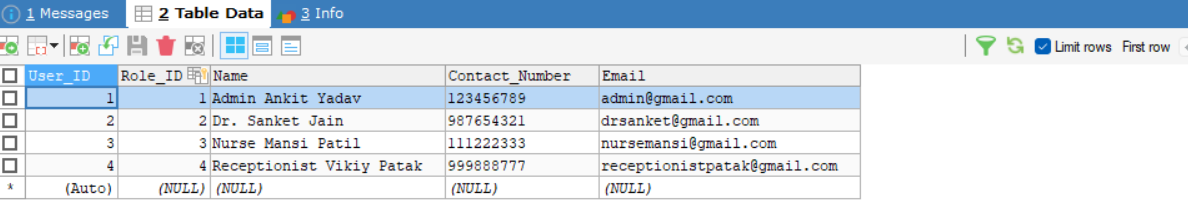
> INSERT INTO UserPersonas (Role\_ID, Name, Contact\_Number, Email) VALUES

(1, 'Admin Ankit Yadav ', '123456789', 'admin@gmail.com'),

(2, 'Dr. Sanket Jain', '987654321', 'drsanket@gmail.com'),

(3, 'Nurse Mansi Patil', '111222333', 'nursemansi@gmail.com'),

(4, 'Receptionist Vikiy Patak', '999888777', 'receptionistpatak@gmail.com');



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3) Indicate the Normalization form being used in the schema defined and why you chose to keep it that particular normal form?

->The HMS schema follows the Third Normal Form (3NF) by ensuring:

No repeating groups exist in tables.

Each table appears to represent a distinct entity (e.g., Patient, Doctor, Appointment, etc.), and there are no repeating groups within these tables. Each column holds atomic values.

Non-key attributes are fully functionally dependent on the primary key.

In the Patient table, attributes like Name, Date\_of\_Birth, Gender, etc., are functionally dependent on the Patient\_ID. Similarly, in other tables like Doctor, Appointment, and Billing, attributes seem to depend on their respective primary keys.

There are no transitive dependencies between non-key attributes.

There don't appear to be any transitive dependencies in the schema. Each non-key attribute seems to depend only on the primary key of its respective table, and not on other non-key attributes.

4) Write necessary queries to register new user roles and personas

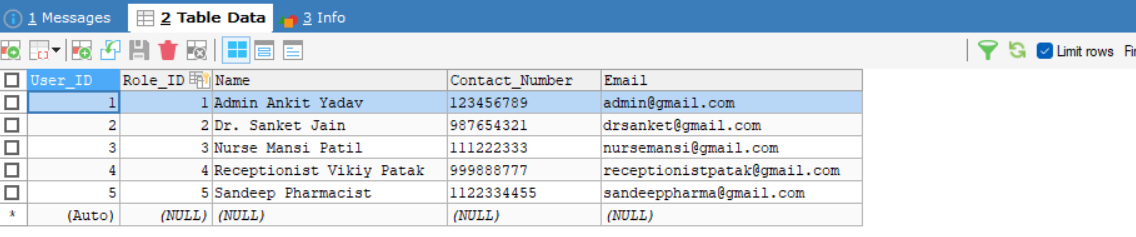
Query to register a new user role:

>INSERT INTO UserRole(Role\_Name) VALUES (‘**Pharmacist**');



Query to register a new user persona associated with the newly created user role’

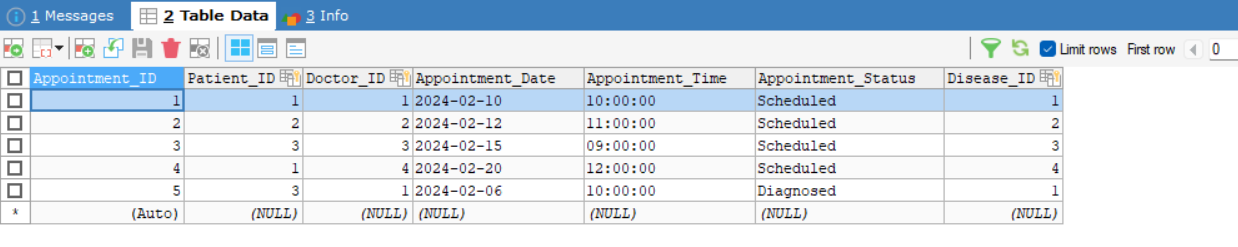
> INSERT INTO UserPersonas (Role\_ID, Name, Contact\_Number, Email) VALUES (5, 'Sandeep Pharmacist', '1122334455', sandeeppharma@gmail.com);



5) Write necessary queries to add to the list of diagnosis of the patient tagged by date.

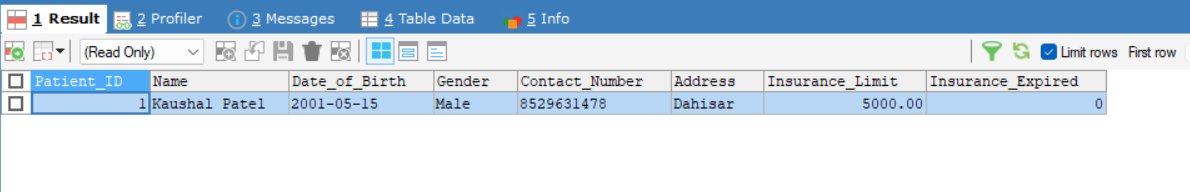
> INSERT INTO Appointment (Patient\_ID, Doctor\_ID, Appointment\_Date, Appointment\_Time, Appointment\_Status, Disease\_ID)

VALUES (3, 1, '2024-02-06', '10:00', 'Diagnosed', 1);



6) Write necessary queries to fetch required details of a particular patient.

> SELECT \* FROM Patient WHERE Patient\_ID = 1;



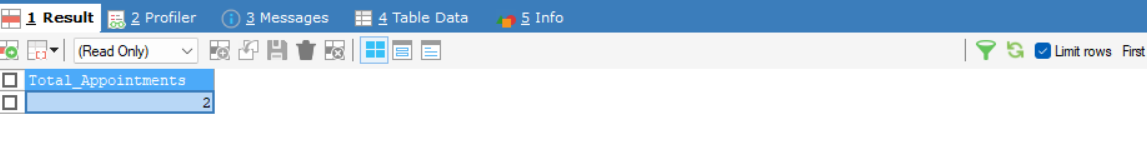
7) Write necessary queries to prepare bill for the patient at the end of checkout.

-> First, I have checked how many appointments there are for a particular patient.

>SELECT COUNT(\*) AS Total\_Appointments

FROM Appointment

WHERE Patient\_ID = 1;



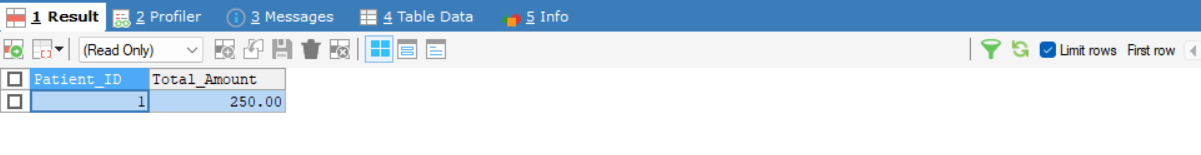
-> Then, I have calculated the total amount of those appointments.

> SELECT Patient\_ID, SUM(Total\_Amount) AS Total\_Amount

FROM Billing

WHERE Patient\_ID = 1

AND Payment\_Status = 'Pending';



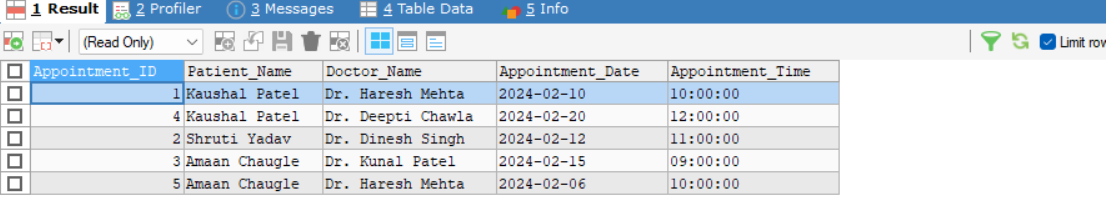
8) Write necessary queries to fetch and show data from various related tables (Joins)

> SELECT A.Appointment\_ID, P.Name AS Patient\_Name, D.Name AS Doctor\_Name, A.Appointment\_Date, A.Appointment\_Time

FROM Appointments A

JOIN Patient P ON A.Patient\_ID = P.Patient\_ID

JOIN Doctor D ON A.Doctor\_ID = D.Doctor\_ID;



> SELECT B.Bill\_ID, P.Name AS Patient\_Name, D.Name AS Doctor\_Name, DS.Disease\_Name, B.Total\_Amount, B.Payment\_Status, B.Billing\_Date

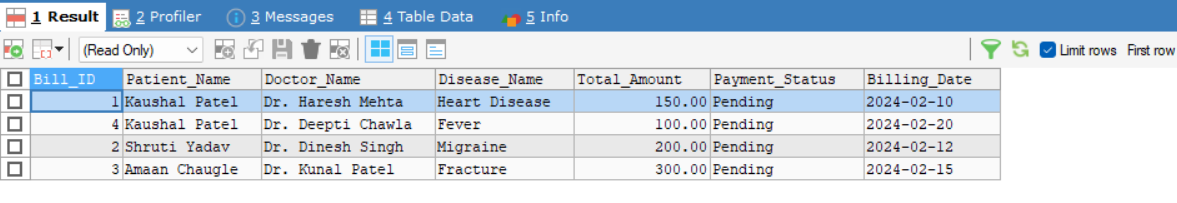
FROM Billing B

JOIN Appointments A ON B.Appointment\_ID = A.Appointment\_ID

JOIN Patient P ON A.Patient\_ID = P.Patient\_ID

JOIN Doctor D ON A.Doctor\_ID = D.Doctor\_ID

JOIN Diseases DS ON B.Disease\_ID = DS.Disease\_ID;



> SELECT A.Appointment\_ID, P.Name AS Patient\_Name, DS.Disease\_Name, DPT.Department\_Name, A.Appointment\_Date, A.Appointment\_Time

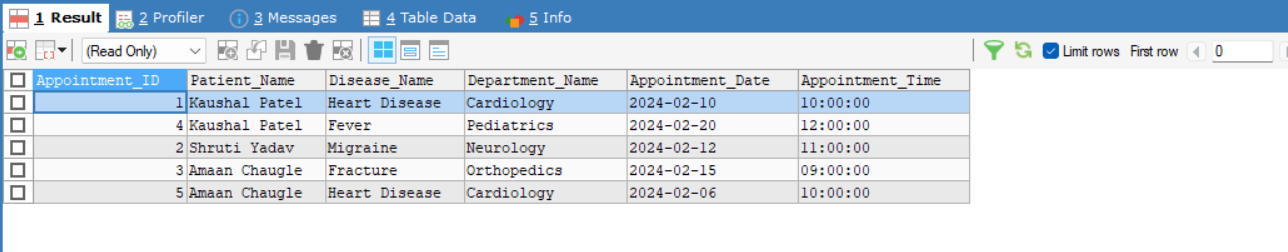
FROM Appointments A

JOIN Patient P ON A.Patient\_ID = P.Patient\_ID

JOIN Diseases DS ON A.Disease\_ID = DS.Disease\_ID

JOIN Doctor D ON A.Doctor\_ID = D.Doctor\_ID

JOIN Department DPT ON D.Department\_ID = DPT.Department\_ID;



9) Optimize repeated read operations using views/materialized views.

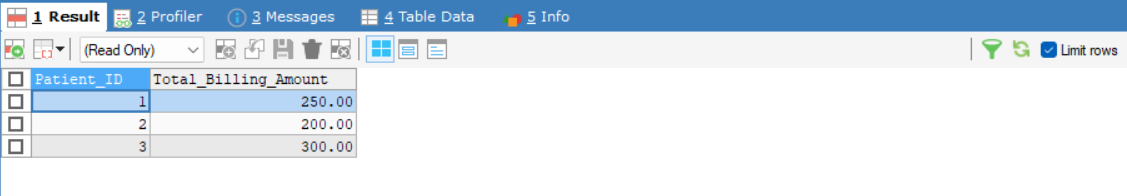
> CREATE VIEW TotalBillingPerPatient AS

SELECT Patient\_ID, SUM(Total\_Amount) AS Total\_Billing\_Amount

FROM Billing

GROUP BY Patient\_ID;

> SELECT \* FROM TotalBillingPerPatient;



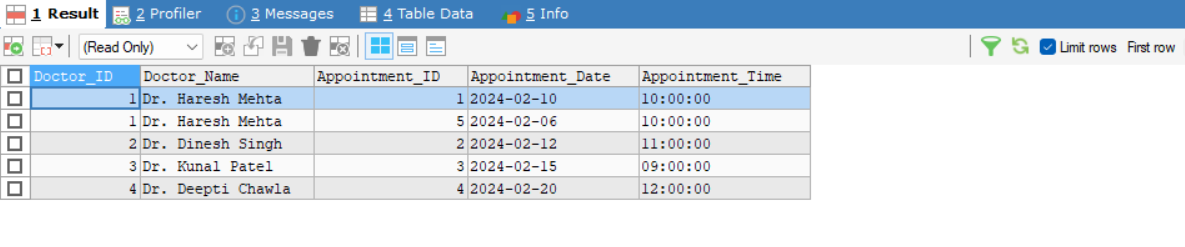
> CREATE VIEW AppointmentsPerDoctor AS

SELECT d.Doctor\_ID, d.Name AS Doctor\_Name, a.Appointment\_ID, a.Appointment\_Date, a.Appointment\_Time

FROM Doctor d

LEFT JOIN Appointment a ON d.Doctor\_ID = a.Doctor\_ID;

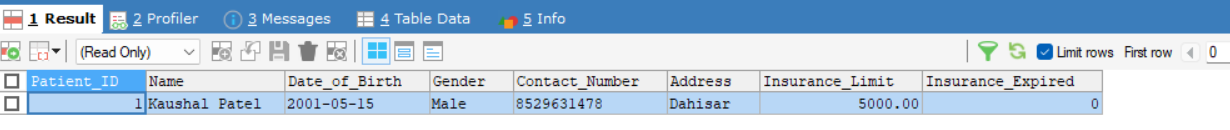
> SELECT \* FROM AppointmentsPerDoctor;



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

> CREATE INDEX idx\_patient\_id ON Patient(Patient\_ID);

> SELECT \* FROM Patient WHERE Patient\_ID = 1;



11) Try optimizing bill generation using stored procedures.

> DELIMITER //

CREATE PROCEDURE GenerateBill(IN patientID INT)

BEGIN

DECLARE totalAmount DECIMAL(10, 2);

SELECT SUM(Total\_Amount)

INTO totalAmount

FROM Billing

WHERE Patient\_ID = patientID;

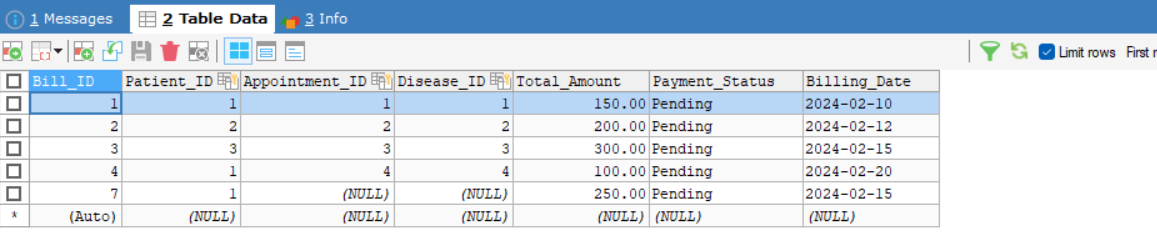
INSERT INTO Billing (Patient\_ID, Total\_Amount, Billing\_Date, Payment\_Status)

VALUES (patientID, totalAmount, CURDATE(), 'Pending');

END//

DELIMITER ;

> CALL GenerateBill(1);



12) Add necessary triggers to indicate when patients medical insurance limit has expired.

> DELIMITER //

CREATE TRIGGER CheckInsuranceLimit

AFTER INSERT ON Appointment

FOR EACH ROW

BEGIN

DECLARE totalAmount DECIMAL(10, 2);

DECLARE insuranceLimit DECIMAL(10, 2);

SELECT SUM(D.Price)

INTO totalAmount

FROM Appointment A

LEFT JOIN Disease D ON A.Disease\_ID = D.Disease\_ID

WHERE A.Patient\_ID = NEW.Patient\_ID;

SELECT Insurance\_Limit

INTO insuranceLimit

FROM Patient

WHERE Patient\_ID = NEW.Patient\_ID;

IF totalAmount > insuranceLimit THEN

UPDATE Patient

SET Insurance\_Expired = 1

WHERE Patient\_ID = NEW.Patient\_ID;

END IF;

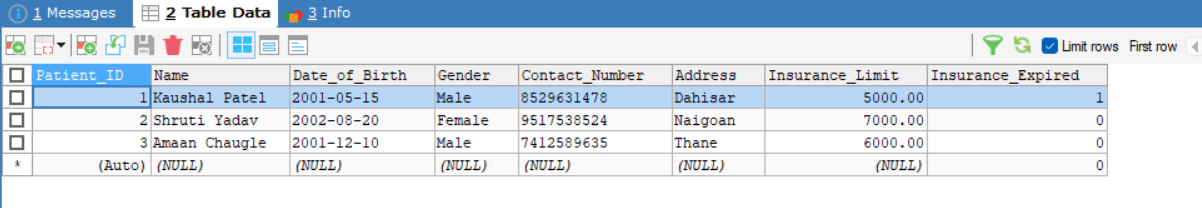
END;

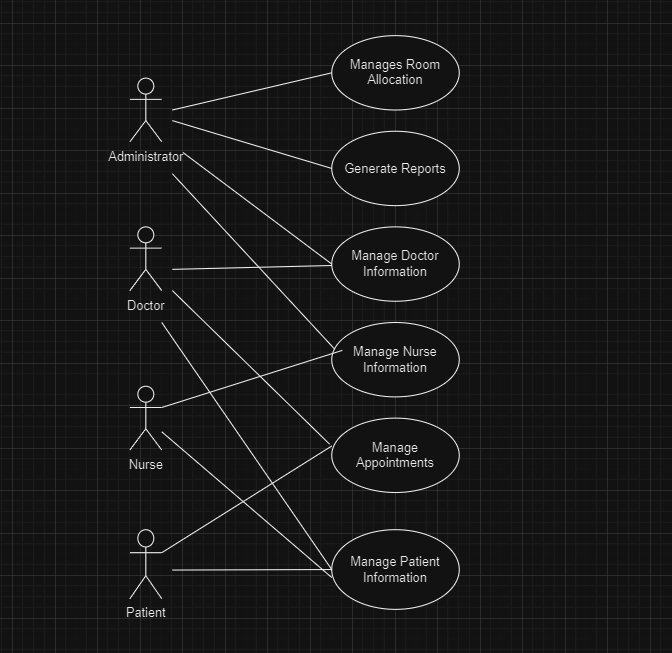
//

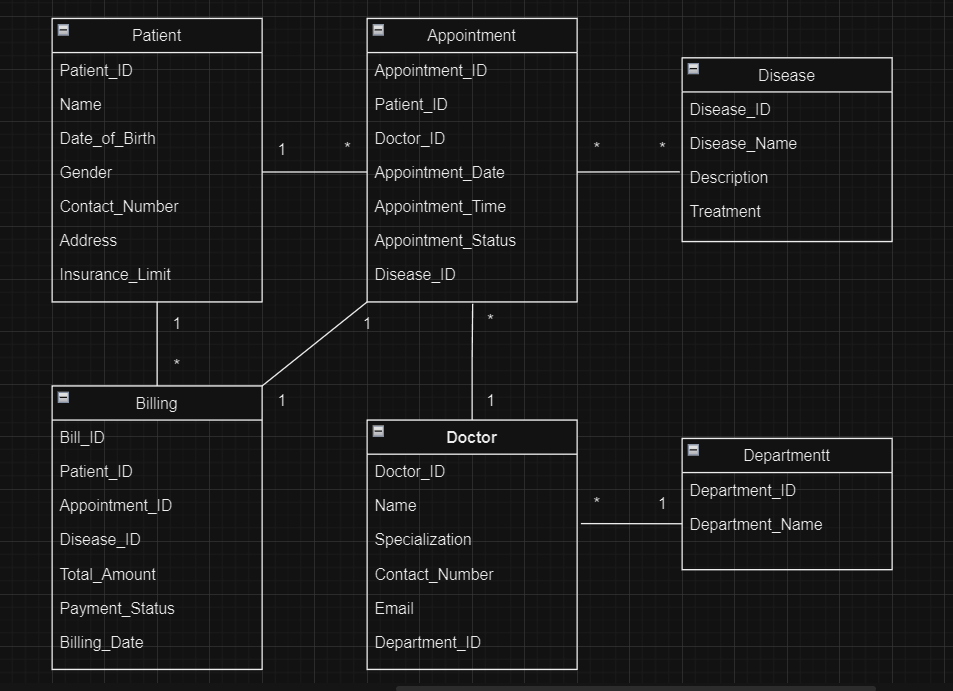
DELIMITER ;

> INSERT INTO Billing (Patient\_ID, Appointment\_ID, Disease\_ID, Total\_Amount, Payment\_Status, Billing\_Date) VALUES

(1, 1, 1, 5100.00, 'Pending', '2024-02-19');



****

****