****

**Introduction**

A Hospital Management System (HMS) is a comprehensive solution designed to streamline the administrative, financial, and clinical operations of a healthcare facility. It serves as a centralized platform for managing various aspects of hospital operations, including patient information, appointment scheduling, billing, inventory management, and more.

**Challenges**

The hospital is facing challenges in managing its growing patient base, medical records, appointments, and billing. The current manual system is time-consuming, prone to errors, and lacks the ability to quickly retrieve patient information. The hospital management has decided to implement a Hospital Management System (HMS) to streamline its operations and improve patient care.

**Significance**

The HMS will make easy the process of managing patient information, appointments, medical records, billing, and medicine. The system will provide a centralized database for storing and retrieving patient information, including medical history, allergies, and pre-existing conditions. The system will also manage appointments.

The HMS will also automate the billing process, generating accurate bills based on the services provided to the patient. The system will track payments, manage accounts receivable, and generate financial reports. The inventory management module will track the hospital's inventory of medicines, medical supplies, and equipment.

The HMS will also provide a platform for communication between hospital staff, doctors, and patients. The system will enable hospital staff to collaborate on patient care, share medical records, and schedule appointments. The system will also provide patients with access to their medical records, appointment schedules, and billing information.

**Objectives**

The previous system is facing a lot of challenges to maintain the information of patient, doctors, billing, so this system will provide more flexibility. The main objectives of this project include:

* Maintain the information of Patients like ID, Name, Blood Type, Contact
* Maintain the information of doctors like ID, name, qualification, department
* Maintain the information of Appointments like Appointment date, time, doctor
* Maintain the information of billing like bill id, total cost, remaining balance
* Store the information of all staff members like nurse information.
* Keep the track of medical History of patient
* Keep the track of room which has been assigned to a patient
* Keep the information of departments

**Problem Statement**

In light of the challenges faced by the hospital in managing its growing patient base, medical records, appointments, and billing, there is a pressing need for the implementation of a comprehensive Hospital Management System (HMS). The current manual system proves to be time-consuming, error-prone, and lacks efficient retrieval of patient information. Therefore, the hospital management has initiated a project to develop and deploy an HMS to address these challenges and enhance patient care.

**Conceptual Design**

**Entity: Patient**

**Attributes: (Patient\_ID, Patient\_Name, Gender, Blood\_Type, Phone, Emergency\_Contact, Admission\_date, Discharge\_date)**

**Entity: Doctor**

**Attributes: (Doctor\_ID, Doctor\_name, Qualification, Specialization)**

**Entity: Nurse**

**Attributes: (Nurse\_ID, Name, contact, Gender, Qualification)**

**Entity: Room**

**Attributes: (Room\_ID, Room\_Cost, Room\_Type)**

**Entity: Department**

**Attributes: (Department\_id, Department\_name, Emp\_Num)**

**Entity: StaffRecord**

**Attributes: (Emp\_id, Name, Date\_Join, Date\_Separation, Email, Address, Contact\_Num)**

**Entity: Medicine**

**Attributes: (Medicine\_id, Medicine\_name, Cost)**

**Entity: Appointment**

**Attributes: (Appointment\_id, Date, Time, Doctor\_id)**

**Entity: Medical\_History**

**Attributes: (Record\_id, Patient\_name, Allergies)\**

**Entity: Prescription**

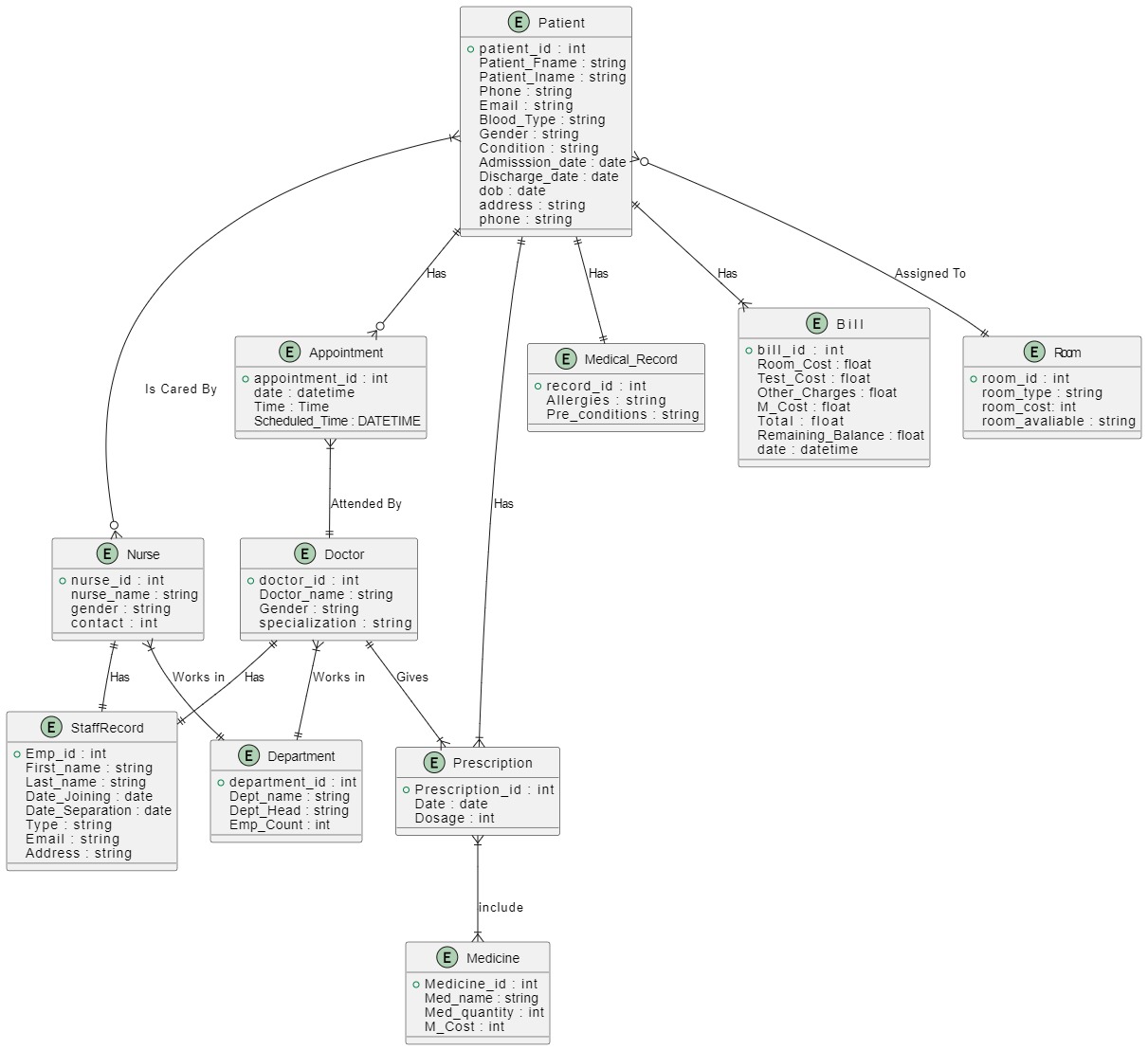
**Attributes: (Prescription\_id, Patient\_id, Medicine, Date)**

**Entity: Bill**

**Attributes: (Bill\_id, Total, M\_Cost, Remaining\_Balance,**

**Other\_Charges, Room\_Cost, Date)**

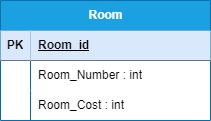
**ER Diagram**

****

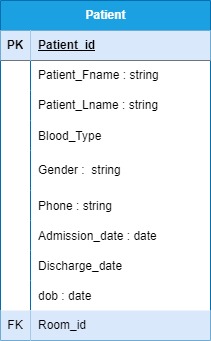
|  |  |
| --- | --- |
| A diagram of a medical procedure  Description automatically generated  A patient can have multiple bills associated with them .A bill must be charged to one and only one patient | A patient requires a medical history record associated with a hospital. Each medical history record is linked to a patient.  Every patient must be allocated to precisely one room. A room may accommodate zero or multiple patients.  A patient may be take cared by one or more nurse. A nurse may be assigned to one or many patient  A medicine has one or many prescriptions. A prescription has a one or many medicine  A staff record is exclusively linked to one and only one doctor. A doctor must have one and only one record under staff.  A nurse must have one and only one record under staff .A staff record has a one and only one nurse  A staff member must be connected with one and only one department. A department may have one to many staff member.  A doctor may have one to many Appointment.An Appointment must be scheduled with one and only one doctor  An appointment must be linked to one and only one patient. A Patient may have zero to many Appointment  A prescription linked with one and only one patient .A patient may have one or many prescription. |

**Relational Schema**

**Room**

****

**Patient**

****

**Relationship: A patient has many relationships to one room**

**Because of that we use room\_id(primary key) as foreign key in patient table**

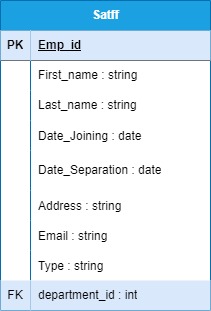
**Appointment**

****

**Relationship: There is of one patient, doctor has relationship to many appointments**

**Because of that we use primary key of patient and doctor in appointment table**

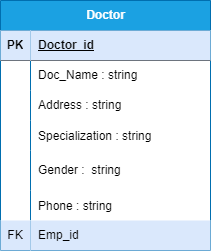
**Staff**

****

**Relationship: There is one department has relationship to multiple staff**

**Because of that we use department\_id (primary key) of departments as a foreign key in staff.**

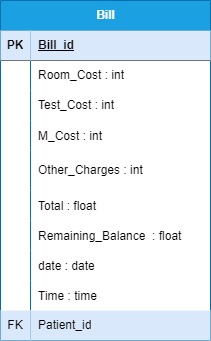
**Doctor**

****

**Relationship: There is one to one relationship between staff and doctor**

**Because of that we use emp\_id of staff as foreign key in doctor table we can use visa versa**

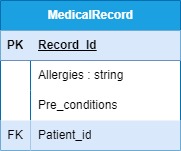
**Bill**

****

**Relationship: There is one patient has relationship to many bills**

**Because of that we use patient\_id as foreign key in bill table**

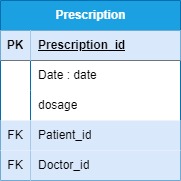
**Medical Record**

****

**Relationship: There is one to one relationship between Patient and medical Record**

**Because of that we use patient\_id as a foreign key in medical\_record Table**

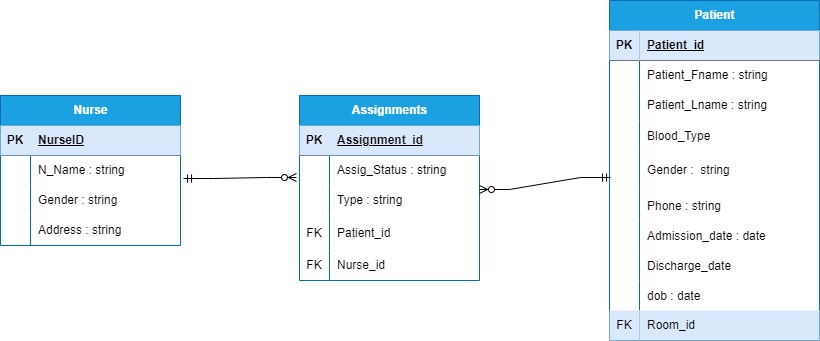
**Prescription**

****

**There is one patient has relationship to many prescription and a doctor also can give many prescriptions**

**Because of that we use patient\_id, doctor\_id as foreign key in prescription table**

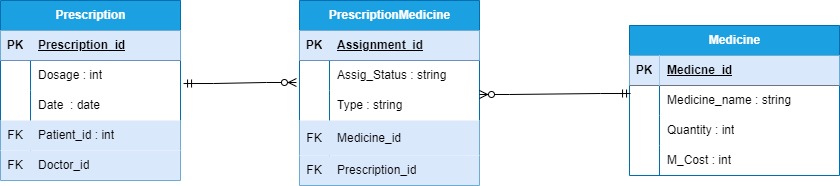
**Nurse**

****

**Relationship: There is many to many relationship between nurse and patient**

**Because of that we create combined table assignments where we use patient\_id and nurse\_id as a foreign key in assignments.**

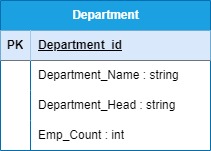
**Medicine**

****

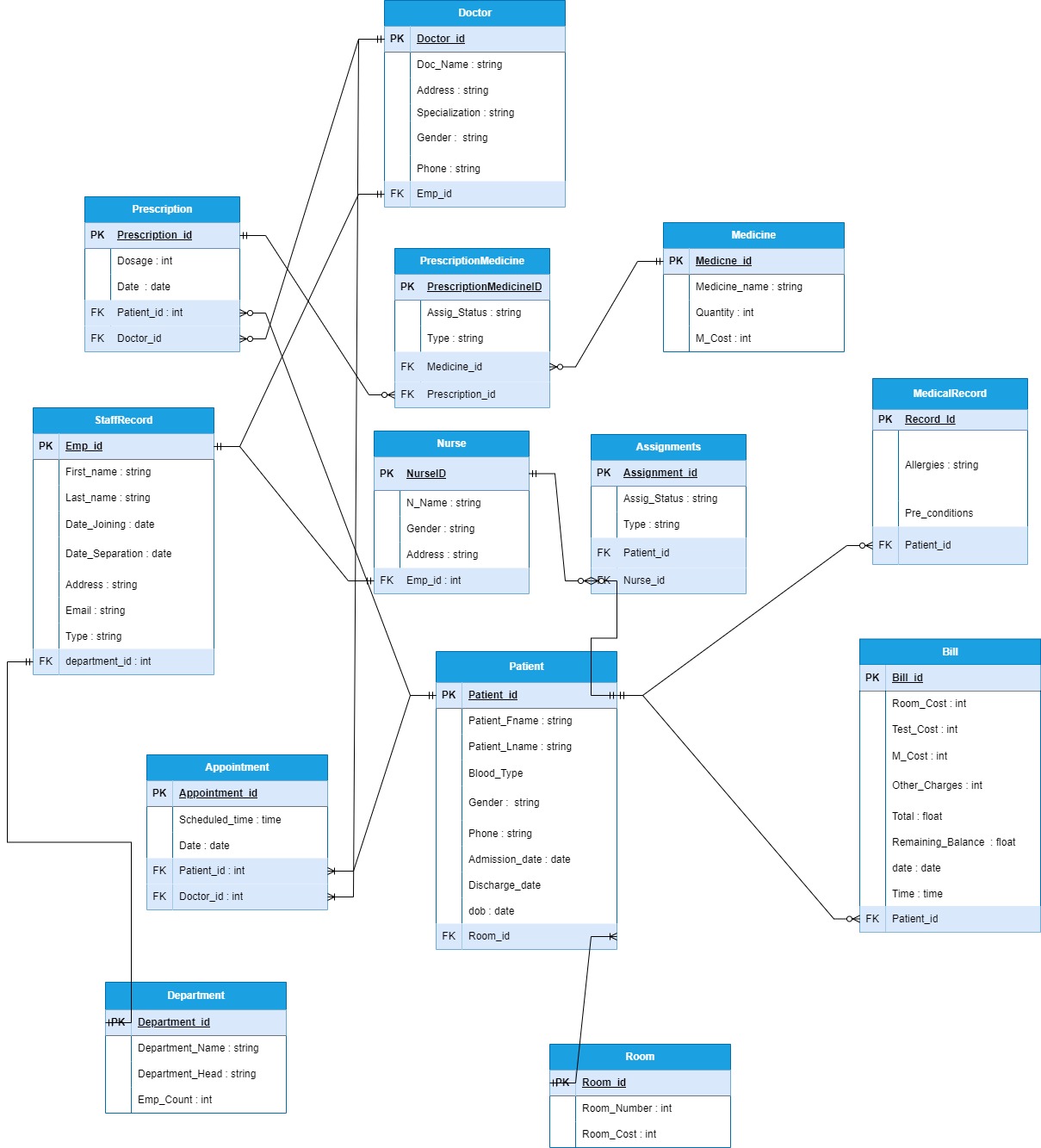
**There is many to many relationships between medicine and prescription.**

**Because of that we make joined table prescriptionMedicine in which we use medicine\_id and prescription\_id as a foreign key**

**Department**

****

**Relational Model**

****

**Schema**

**Patient (Patient\_ID, Patient\_Name, Gender, Blood\_Type, Phone, Emergency\_Contact, Admission\_date, Discharge\_date, Room\_ID)**

**Doctor (Doctor\_ID, Doctor\_name, Address, Gender, Phone Specialization)**

**Nurse (Nurse\_ID, Name, contact, Gender, Qualification, Emp\_id)**

**Room (Room\_ID, Room\_Cost, Room\_Type)**

**Department (Department\_id, Department\_name, Department\_Head, Emp\_Num)**

**StaffRecord (Emp\_id, Name, Date\_Join, Date\_Separation, Email, Address, Contact\_Num, Department\_id)**

**Medicine (Medicine\_id, Medicine\_name, Cost)**

**Appointment (Appointment\_id, Date, Time, Doctor\_id, Patient\_id)**

**Medical\_History (Record\_id, Patient\_name, Allergies, Patient\_id)**

**Prescription (Prescription\_id, Patient\_id, Medicine, Date,**

**Patient\_id, Doctor\_id)**

**Bill (Bill\_id, Total, M\_Cost, Remaining\_Balance,**

**Other\_Charges, Room\_Cost, Date, Patient\_id)**

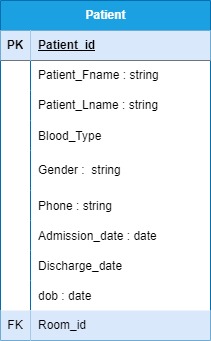
**Assignment (Assignment\_id, Date, Nurse\_ID, Patient\_id)**

**MedicinePrescription (MedicinePrescription\_id, Assign\_Status,**

**Type, Medicine\_id, Prescription\_id)**

**NORMALIZATION**

**PATIENT**

****

**1NF**

The table complies with the First Normal Form (1NF) as it contains only atomic values in each column without any repeating groups.

**Functional Dependencies**

(patient\_id → patient\_fname, patient\_lname, Phone, blood\_Type, email, gender, admission\_date, discharge\_date,dob,Room\_id )

**2NF**

Regarding Second Normal Form (2NF), it meets the criteria by having a primary key, "patient\_id," which uniquely identifies each row, and all non-key attributes are fully functionally dependent on this primary key.

**3NF**

Moving on to Third Normal Form (3NF), the table satisfies this requirement as well since it contains no transitive dependencies. All non-key attributes, such as patient name, phone number, blood group, email, gender, room\_id, admission date, and discharge date, are directly determined by the primary key "patient\_id."

**APPOINTMENT**



**1NF**

The table is in the First Normal Form (1NF) because it contains atomic values in each column, ` and there are no repeating groups.

**Functional dependenciy**

( Appointment\_id → Scheduled\_ Time,Date, Doctor\_ID, Patient\_id **)**

**2NF**

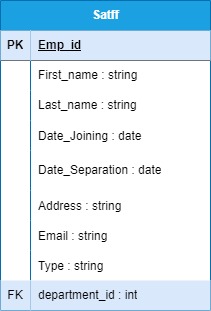
 It is also in the Second Normal Form (2NF) since it has a primary key, “Appt\_id,” which uniquely identifies each row, and all non-key attributes are fully functionally dependent on the primary key.

**3NF**

The table satisfies the Third Normal Form (3NF) because it does not contain any transitive dependencies. All non-key attributes are directly dependent on the primary keys, “Apptointment\_id,”.

This set of functional dependencies indicates that for each appointment, identified by “Appointment\_id,” the attributes “Scheduled\_time,” “Date,” “Doctor\_ID,” and “Patient\_id” are directly determined. There are no transitive dependencies, ensuring that the “Appointment” table is well-structured and adheres to 3NF principles.

**STAFF RECORD**



**1NF Compliance:** The table is in the First Normal Form (1NF) because it does not contain repeating group.

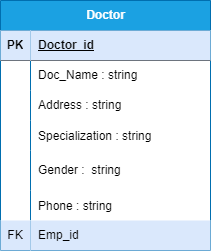
**Functional Dependencies:**

emp\_id → First\_name,Last\_name, DoB, Date\_joining, Type, email, address, data\_separation, department\_id,

**2NF Compliance:** It is in the Second Normal Form (2NF) since it has a primary key, “emp\_id,” which uniquely identifies each row, and all non-key attributes are fully functionally dependent on the primary key.

**3NF Compliance:** The StaffRecord table is in 3rd NF because there is no any transitive property exist in the table.

**DOCTOR**



**1NF**

 Each column in the table contains atomic (indivisible) values, meeting the requirements of 1NF.

**Functional Dependencies:**

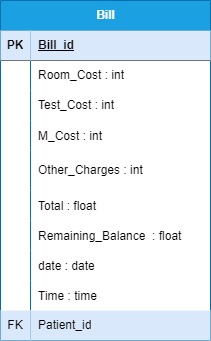
(doctor\_id → doc\_name,phone,gender,address,specialization, Emp\_id)

**2NF**

 The table goes beyond 1NF by having a primary key, “doctor\_id,” which uniquely identifies each row. All non-key attributes are fully functionally dependent on the primary key, adhering to 2NF principles.

**3NF Compliance:**The table satisfies the Third Normal Form (3NF) because it does not contain any transitive dependencies. All non-key attributes are directly dependent on the primary key, “Doctor\_id.”

**BILL**

****

**1NF**

 The table is in the First Normal Form (1NF) because it contains atomic values in each column, and there are no repeating groups.

**Functional Dependencies:**

bill\_id → date, room\_cost, test\_cost, other\_charges, m\_cost, total,patient\_id , time,date,remaning\_balace,float )

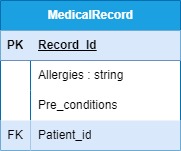
**2NF**

It is also in the Second Normal Form (2NF) since it has a primary key, “bill\_id,” which uniquely identifies each row, and all non-key attributes are fully functionally dependent on the primary key.

**3NF**

The table satisfies the Third Normal Form (3NF) because it does not contain any transitive dependencies. All non-key attributes are directly dependent on the primary key, bill\_id

**Medical history**



**1NF**

The table is in the First Normal Form (1NF) because it contains atomic values in each column, and there are no repeating groups.

**Functional Dependencies:**

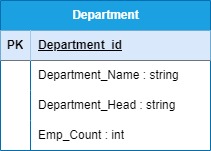
(patient\_id → allergies, pre\_conditions)

**2NF Compliance:** It is also in the Second Normal Form (2NF) since it has a primary key, “Record\_id,” which uniquely identifies each row, and all non-key attributes are fully functionally dependent on the primary key.

**3NF**

 The table satisfies the Third Normal Form (3NF) because it does not contain any transitive dependencies. All non-key attributes are directly dependent on the primary key, “Record\_id.”

**Department**

****

**1NF**

The table is in the First Normal Form (1NF) because it contains atomic values in each column, and there are no repeating groups.

**Functional Dependencies:**

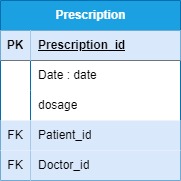
(Department\_id → Department\_name, Department\_Head,Emp\_count)

**2NF Compliance:** It is also in the Second Normal Form (2NF) since it has a primary key, “Department\_id,” which uniquely identifies each row, and all non-key attributes are fully functionally dependent on the primary key.

**3NF**

 The table satisfies the Third Normal Form (3NF) because it does not contain any transitive dependencies. All non-key attributes are directly dependent on the primary key, “Deparatment\_id.”

**Prescription**

****

**1NF**

The table is in the First Normal Form (1NF) because it contains atomic values in each column, and there are no repeating groups.

**Functional Dependencies:**

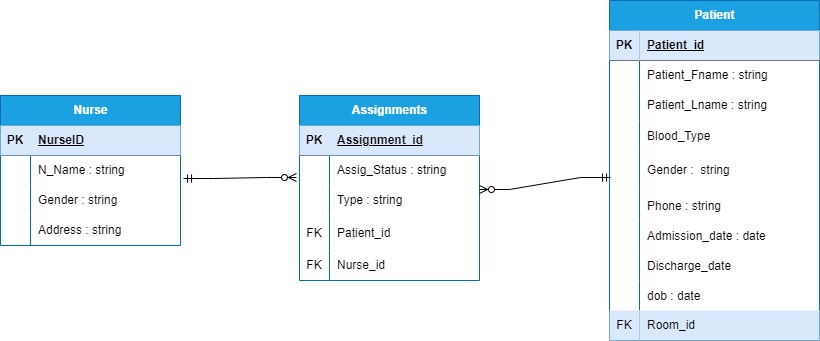
(Prescription\_id → date, dosage,Patient\_id,Doctor\_id)

**2NF Compliance:** It is also in the Second Normal Form (2NF) since it has a primary key, “Prescription\_id,” which uniquely identifies each row, and all non-key attributes are fully functionally dependent on the primary key.

**3NF**

 The table satisfies the Third Normal Form (3NF) because it does not contain any transitive dependencies. All non-key attributes are directly dependent on the primary key, “Prescription\_id.”

**Nurse**

****

**1NF**

The table is in the First Normal Form (1NF) because it contains atomic values in each column, and there are no repeating groups.

**Functional Dependencies:**

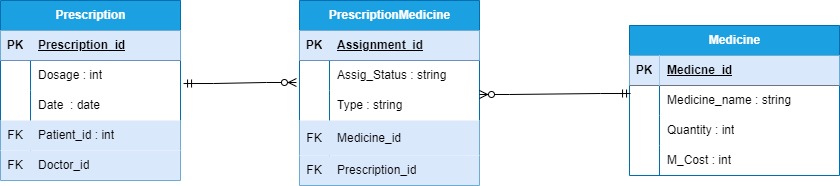
(Nurse\_id → N\_name, Gender,Address)

**2NF Compliance:** It is also in the Second Normal Form (2NF) since it has a primary key, “Nurse\_id,” which uniquely identifies each row, and all non-key attributes are fully functionally dependent on the primary key.

**3NF**

 The table satisfies the Third Normal Form (3NF) because it does not contain any transitive dependencies. All non-key attributes are directly dependent on the primary key, “Nurse\_id.”

**Medicine**

****

**1NF**

The table is in the First Normal Form (1NF) because it contains atomic values in each column, and there are no repeating groups.

**Functional Dependencies:**

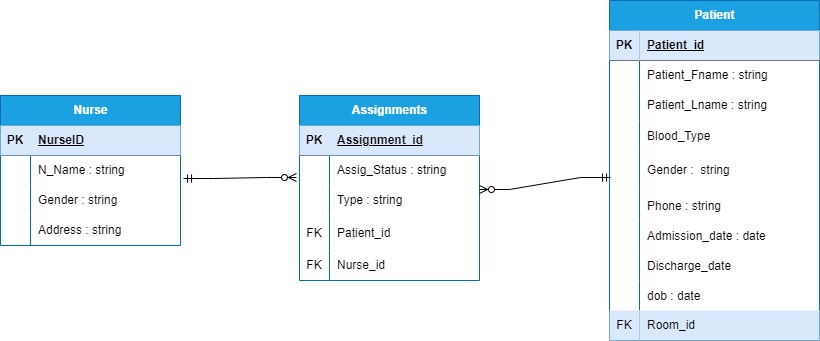
(Medicine\_id → Medicine\_name, Quantity,M\_cost)

**2NF Compliance:** It is also in the Second Normal Form (2NF) since it has a primary key, “Medicine\_id,” which uniquely identifies each row, and all non-key attributes are fully functionally dependent on the primary key.

**3NF**

 The table satisfies the Third Normal Form (3NF) because it does not contain any transitive dependencies. All non-key attributes are directly dependent on the primary key, “Medicne\_id.”

**Assignemet**

****

**1NF**

The table is in the First Normal Form (1NF) because it contains atomic values in each column, and there are no repeating groups.

**Functional Dependencies:**

(Assignment\_id → Assig\_status, Type,Patient\_id,Nurse\_id)

**2NF Compliance:** It is also in the Second Normal Form (2NF) since it has a primary key, “Assignemnt\_id,” which uniquely identifies each row, and all non-key attributes are fully functionally dependent on the primary key.

**3NF**

 The table satisfies the Third Normal Form (3NF) because it does not contain any transitive dependencies. All non-key attributes are directly dependent on the primary key, “Assignment\_id.”

**Implementation**

create database if not exists HMS;

create table Room (

Room\_id int primary key

, Room\_Number int

, Room\_Cost int

);

create table Department (

Department\_id int primary key,

Department\_name varchar(100),

Department\_Head varchar(100),

Emp\_Count int

);

create table Patient (

Patient\_id int primary key,

Patient\_Fname varchar(100) not null,

Patient\_Lname varchar(100) not null,

Blood\_Type varchar(20),

Gender varchar(50),

Phone varchar(100) not null,

Admission\_Date date,

Discharge\_Date date,

DOB date,

Room\_id int,

constraint FK\_ROOMID FOREIGN key(Room\_id) references Room(Room\_id)

on delete cascade on update cascade

);

Phone varchar(100) not null,

Admission\_Date date,

Discharge\_Date date,

DOB date,

Room\_id int,

constraint FK\_ROOMID FOREIGN key(Room\_id) references Room(Room\_id)

on delete cascade on update cascade

);

create table Bill (

Bill\_id int primary key,

Room\_Cost int,

Test\_Cost int,

M\_Cost int,

Other\_Charges int,

Total int,

Remaining\_Balance int,

Bill\_Date date,

Bill\_Time time,

patient\_id int,

constraint FK\_PatientID FOREIGN key(Patient\_id) references Patient(Patient\_id)

on delete cascade on update cascade

);

create Table MedicalRecord (

Record\_id int primary key,

PreCondition varchar(100),

Patient\_id int,

constraint FK\_RPatientID FOREIGN key(Patient\_id) references Patient(Patient\_id)

on delete cascade on update cascade);

create table StaffRecord (

Emp\_id int primary key,

First\_name varchar(100) not null,

Last\_name varchar(100) not null,

Date\_Joining date,

Date\_Separation date,

Address varchar(100),

Email varchar(100),

Type varchar(100),

Department\_id int,

constraint FK\_DepartmentID FOREIGN key(Department\_id) references Department(Department\_id)

on delete cascade on update cascade);

create table Doctor (

Doctor\_id int primary key,

Address varchar(100),

Specialization varchar(100),

Gender varchar(100),

Phone varchar(100),

Emp\_id int,

constraint FK\_EmpID FOREIGN key(Emp\_id) references StaffRecord(Emp\_id)

on delete cascade on update cascade

);

create table Appointment (

Appointment\_id int primary key,

Scheduled\_Time time,

A\_Date date,

Patient\_id int,

Doctor\_id int,

constraint FK\_APatientID FOREIGN key(Patient\_id) references Patient(Patient\_id)

on delete cascade on update cascade,

constraint FK\_Doctor\_ID FOREIGN key(Doctor\_id) references Doctor(Doctor\_id)

on delete cascade on update cascade

);

create table Prescription (

Prescription\_id int primary key,

Dosage int,

Date date,

Patient\_id int,

Doctor\_id int,

constraint FK\_PPatientID FOREIGN key(Patient\_id) references Patient(Patient\_id)

on delete cascade on update cascade,

constraint FK\_PDoctor\_ID FOREIGN key(Doctor\_id) references Doctor(Doctor\_id)

on delete cascade on update cascade);

create table Nurse (

Nurse\_id int primary key,

N\_Name varchar(100),

Gender varchar(100),

Address varchar(100),

Emp\_id int,

constraint FK\_NEmpID FOREIGN key(Emp\_id) references StaffRecord(Emp\_id)

on delete cascade on update cascade

);

create table Medicine (

Medicine\_id int primary key,

Medicine\_name varchar(100),

Quantity int,

M\_Cost int);

create table Assignements (

Assignment\_id int primary key,

A\_Status varchar(100),

A\_Type varchar(100),

Patient\_id int,

Nurse\_id int,

constraint FK\_ASPatientID FOREIGN key(Patient\_id) references Patient(Patient\_id)

on delete cascade on update cascade,

constraint FK\_NurseID FOREIGN key(Nurse\_id) references Nurse(Nurse\_id)

on delete cascade on update cascade);

create table Prescription\_Medicines (

PM\_id int primary key,

M\_Quantity int,

Medicine\_id int,

Prescription\_id int,

constraint FK\_MedicineID FOREIGN key(Medicine\_id) references Medicine(Medicine\_id)

on delete cascade on update cascade,

constraint FK\_PrescriptionID FOREIGN key(Prescription\_id) references Prescription(Prescription\_id)

on delete cascade on update cascade

);

**Testing**

**#query 1 Find the total bills of all patients**

**SELECT p.Patient\_id, p.Patient\_Fname, p.Patient\_Lname, SUM(b.Total) AS Total\_Bill\_Amount**

**FROM Patient p**

**JOIN Bill b ON p.Patient\_id = b.patient\_id**

**GROUP BY p.Patient\_id, p.Patient\_Fname, p.Patient\_Lname;**

**#query2 Retrieve all patients along with their assigned doctors**

**SELECT p.Patient\_id, p.Patient\_Fname, p.Patient\_Lname, d.Doctor\_id, d.Specialization**

**FROM Patient p**

**JOIN Appointment a ON p.Patient\_id = a.Patient\_id**

**JOIN Doctor d ON a.Doctor\_id = d.Doctor\_id;**

**#query 3**

**SELECT \***

**FROM Patient**

**WHERE Admission\_Date > '2024-04-15';**