

PROJECT 2

HealthCare Center Database

Ashwini Shekhar Phadke
SUID : 575445713

Contents

Abstract	3
Design Considerations	4
Relationship between tables.....	4
PatientDetails Table	4
PatientMedication Table	4
MedicineInfo Table.....	5
ProcedureInfo Table.....	5
PatientHealthRecord Table	5
PatientProcedureHistory Table	5
PatientHospitalization Details Table.....	5
Billing Table	5
Payor Table	5
PaymentMethod Table	5
InsurancePlan	6
PlanCoverage Table	6
InsuranceCompany Table	6
Referred by Table	6
Symptoms Table	6
Patient Room Schedule Table.....	6
EmployeeDetails table	6
Contact Details Table.....	6
EmployeeWorkSchedule Table	7
MedicalDevices Table	7
BedDetails Table.....	7
Location Table	7
Entity Relationship Diagram.....	8
Implementation	9
Security Considerations	22
Testing.....	23
Conclusion	31
Remarks	31

Abstract

A database in simple terms is collection of data for storage, accessibility and retrieval . As the technology advances it has become archaic to store information manually. The digitization of healthcare databases is happening at a fast speed. For a complex data such as in case of a large hospital it becomes even more important to organize it properly. Hospital employees and patients form the main users of the hospital database. Tasks related to them are incorporated in the design. The employees are concerned with the administration of the hospital while the patient's records refers to the procedures, tasks, and health information.

It is necessary to maintain this information in a organized way in form of relational database to support ease of use.

The design of the database should be maintainable and clean owing to the growing nature. Also the end users are going to be people with no technical knowledge to maintain it so the structure should be very readable.

In the following pages, the proposed design for a hospital database is given. The SQL queries related to it are also included. Various stored procedures, views and functions are implemented on the database to test its fullest potential.

Design Considerations

While designing a healthcare database, care should be taken that the organization of data is able to map the real world entities. For this design, I have tried to create tables mapping the basic functionalities often seen in any hospital scenario.

The hospital employees have their personal information stored in databases along with their past employment history and current schedules.

As far as patients are concerned, the hospital needs to maintain their past health records, procedure history, medicines prescribed to them and also their insurance records.

In the following pages the design for the database is explained

Relationship between tables.

PatientDetails Table

- 1) This table contains all the information related to the patients in the hospital. The patient names, SSN details and information about marital status is included in this table
- 2) This table serves as the main table to get all the information related to the individual patient.
- 3) Patientid is the primary key which uniquely defines all the rows in the table.
This key is used as a reference in many other tables to get information about the patient.
- 4) In order to keep the data organized, the address and contact information of the patient is stored in another table and the addressid and contactid are included as the foreign keys to this table so as to access the information
- 5) This table shares one to many relationship with many other tables. The patient id acts as a foreign key in other tables.
- 6) This table shares one to many relationship with billing table, visitors table, patient medication, patient procedure history etc.

PatientMedication Table

- 1) This table contains all the details of the dosage quantity of the medicines prescribed to the patient.
- 2) The details of the medicine can be obtained from the medicine info table which has a foreign key reference.
- 3) Also, the pthealthid reference is used to access the health record of the patient which will be useful for the doctor to change the treatment course.

MedicineInfo Table

- 1) This table is identified uniquely by its primary key MedicineInfoId
- 2) It contains cost of medicine and medicine description.

ProcedureInfo Table

- 1) This table contains the list of different medical procedures and its cost.
- 2) This table basically serves as a directory of various procedures that can be performed in the hospital.
- 3) This information is used in the patientprocedure table to refer the various procedures performed on an individual patient.

PatientHealthRecord Table

- 1) This table contains all the information related to patients vitals. It also stores the date on which the readings were taken for future reference
- 2) The primary treating doctors name is also saved in this table to keep a reference in case of emergency.
- 3) The patientid which is a foreign key in this table uniquely identifies each record to a individual patient.
- 4) It shares a many to one relationship with patient details table.

PatientProcedureHistory Table

- 1) The information related to the medical procedures performed on the patient are stored in this table.
- 2) The name of the operating doctor is also stored in it.
- 3) The ProcedureHistoryID serves as the primary key and the data is associated with a individual patient using the patientid which is the foreign key
- 4) Also, the patient records can be accessed by this table using the PtHealthId key.

PatientHospitalization Details Table

- 1) For the record of the hospital, it is important to note the check in and check out times of the patients.
- 2) This is noted for individuals this table

Billing Table

- 1) The final billing amount of the stay in the hospital for patients is stored in this table
- 2) The foreign keys for payment method and payor reveal the details of the bill paid

Payor Table

- 1) This table contains information as to who will be settling the bill among options like insurance company, self or family.

PaymentMethod Table

- 1) This table contains information about the way the payment would be made that is either by case or card or cheque or other.

InsurancePlan

- 1) The information related to insurance provider company and the name of the plan is stored here.
- 2) Since there is a many to many relationship between the patient and the insurance plan, a linking table is used between them
 - a. The Insurance plan ID is the primary key for this table

PlanCoverage Table

- 1) This table contains information about the coverage offered to the plan holders.
- 2) Information about the procedures performed and medicines used can be retrieved with the foreign keys present so that the cost can be calculated.

InsuranceCompany Table

- 1) The insurance company table contains list of all the companies providing valid insurances to the patients.
- 2) The contact information of the various insurance company plans is saved
- 3) Also, this table shares a one to many relationship with the insuranceplans table.

Referred by Table

- 1) This table contains information about the referral doctor and details of the patient history can be accessed from this table to check the past medical records.

Symptoms Table

- 1) This table contains information about the various symptoms and its description
- 2) This table serves as a directory reference for diagnosis and is included in the patient history of the records.

Patient Room Schedule Table

- 1) Patient Room Schedule contains information about the check in and check out time of the patients and availability of the room for next individual

EmployeeDetails table

- 1) This table primarily contains information about the name, SSN etc of the employees working in the hospital and also their designation in the institution that is if they work as a doctor or nurse or Manager etc.
- 2) The table is uniquely identified with the EmployeeID primary key.
- 3) This table shares one to many relationship with other tables which are concerned with the

Contact Details Table

- 1) The details to contact the employee or patient including the phone and email id.

EmployeeWorkSchedule Table

- 1) The employee schedule contains details about the shift start and end time of the employees and also information about the location where they will be working by accessing the foreign key.

MedicalDevices Table

- 1) The table gives information about various medical devices and the location where they are present in the patient room

BedDetails Table

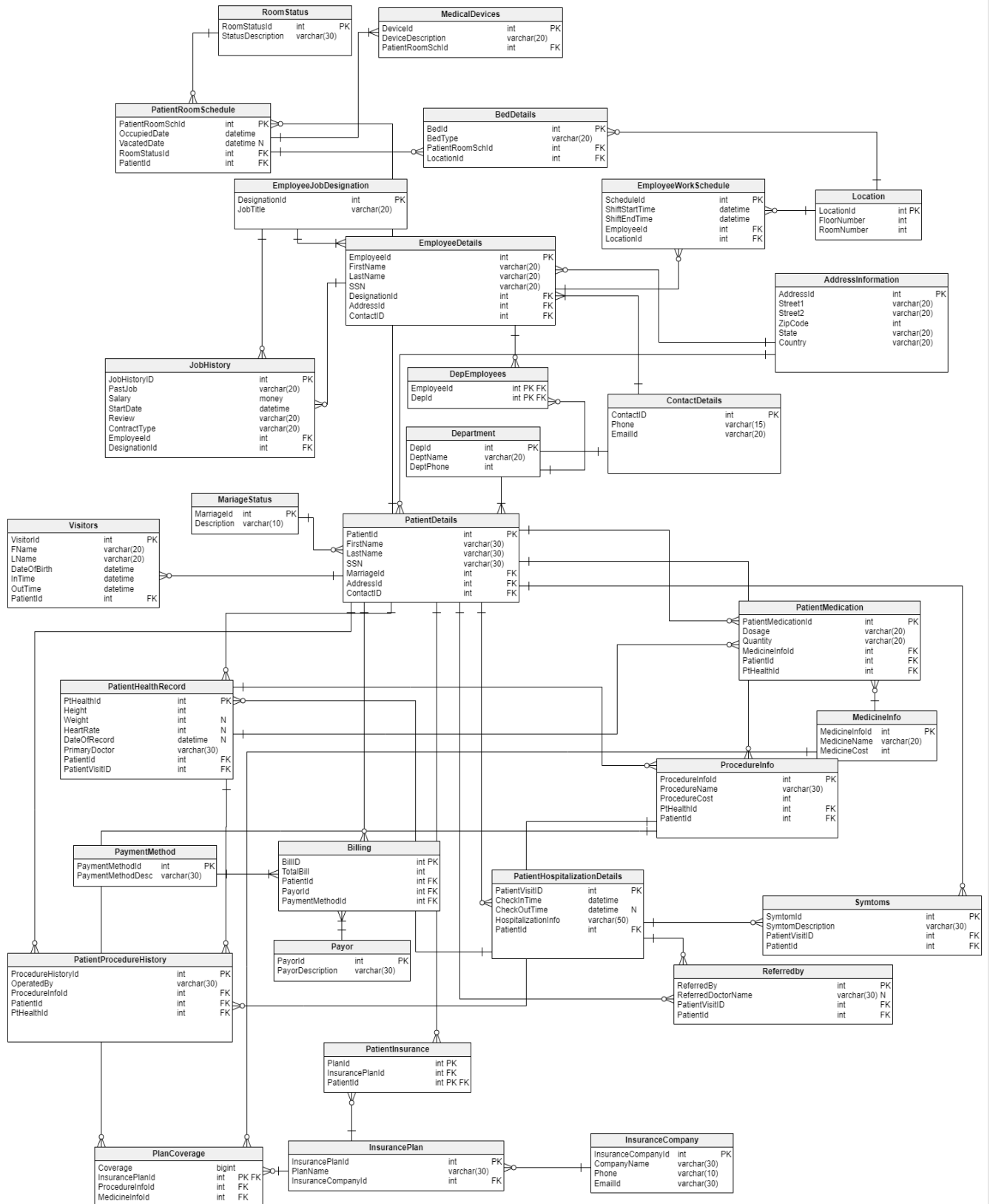
- 1) This table provides details about the types of beds present in the rooms and their locations

Location Table

- 1) This table holds information about the room and floor number in the hospital.
- 2) It is used in the employee schedule table to determine the location of working place for the employee

The entity relationship diagram representing the relationship between all these tables mentioned above is as follows.

Entity Relationship Diagram



Implementation

Following is the sql script to create the tables mentioned above

```
-- Table: AddressInformation
CREATE TABLE AddressInformation (
    AddressId int NOT NULL,
    Street1 varchar(20) NOT NULL,
    Street2 varchar(20) NOT NULL,
    ZipCode int NOT NULL,
    State varchar(20) NOT NULL,
    Country varchar(20) NOT NULL,
    CONSTRAINT AddressInformation_pk PRIMARY KEY (AddressId)
);

-- Table: BedDetails
CREATE TABLE BedDetails (
    BedId int NOT NULL,
    BedType int NOT NULL,
    PatientRoomSchId int NOT NULL,
    LocationId int NOT NULL,
    CONSTRAINT BedDetails_pk PRIMARY KEY (BedId)
);

-- Table: Billing
CREATE TABLE Billing (
    BillID int NOT NULL,
    TotalBill int NOT NULL,
    PatientId int NOT NULL,
    PayorId int NOT NULL,
    PaymentMethodId int NOT NULL,
    CONSTRAINT Billing_pk PRIMARY KEY (BillID)
);

-- Table: ContactDetails
CREATE TABLE ContactDetails (
    ContactID int NOT NULL,
    Phone varchar(15) NOT NULL,
    EmailId varchar(20) NOT NULL,
    CONSTRAINT ContactDetails_pk PRIMARY KEY (ContactID)
);

-- Table: DepEmployees
CREATE TABLE DepEmployees (
    EmployeeId int NOT NULL,
    DepId int NOT NULL,
    CONSTRAINT DepEmployees_pk PRIMARY KEY (EmployeeId,DepId)
);

-- Table: Department
CREATE TABLE Department (
    DepId int NOT NULL,
    DeptName varchar(20) NOT NULL,
    DeptPhone int NOT NULL,
    CONSTRAINT Department_pk PRIMARY KEY (DepId)
);
```

```
-- Table: EmployeeDetails
CREATE TABLE EmployeeDetails (
    EmployeeId int NOT NULL,
    FirstName varchar(20) NOT NULL,
    LastName varchar(20) NOT NULL,
    SSN varchar(20) NOT NULL,
    DesignationId int NOT NULL,
    AddressId int NOT NULL,
    ContactID int NOT NULL,
    CONSTRAINT EmployeeDetails_pk PRIMARY KEY (EmployeeId)
);

-- Table: EmployeeJobDesignation
CREATE TABLE EmployeeJobDesignation (
    DesignationId int NOT NULL,
    JobTitle varchar(20) NOT NULL,
    CONSTRAINT EmployeeJobDesignation_pk PRIMARY KEY (DesignationId)
);

-- Table: EmployeeWorkSchedule
CREATE TABLE EmployeeWorkSchedule (
    ScheduleId int NOT NULL,
    ShiftStartTime datetime NOT NULL,
    ShiftEndTime datetime NOT NULL,
    EmployeeId int NOT NULL,
    LocationId int NOT NULL,
    CONSTRAINT EmployeeWorkSchedule_pk PRIMARY KEY (ScheduleId)
);

-- Table: InsuranceCompany
CREATE TABLE InsuranceCompany (
    InsuranceCompanyId int NOT NULL,
    CompanyName varchar(30) NOT NULL,
    Phone int NOT NULL,
    EmailId varchar(30) NOT NULL,
    CONSTRAINT InsuranceCompany_pk PRIMARY KEY (InsuranceCompanyId)
);

-- Table: InsurancePlan
CREATE TABLE InsurancePlan (
    InsurancePlanId int NOT NULL,
    PlanName varchar(30) NOT NULL,
    InsuranceCompanyId int NOT NULL,
    CONSTRAINT InsurancePlan_pk PRIMARY KEY (InsurancePlanId)
);

-- Table: JobHistory
CREATE TABLE JobHistory (
    JobHistoryID int NOT NULL,
    PastJob varchar(20) NOT NULL,
    Salary money NOT NULL,
    StartDate datetime NOT NULL,
    Review varchar(20) NOT NULL,
    ContractType varchar(20) NOT NULL,
    EmployeeId int NOT NULL,
    DesignationId int NOT NULL,
    CONSTRAINT JobHistory_pk PRIMARY KEY (JobHistoryID)
);
```

```
-- Table: Location
CREATE TABLE Location (
    LocationId int NOT NULL,
    FloorNumber int NOT NULL,
    RoomNumber int NOT NULL,
    CONSTRAINT Location_pk PRIMARY KEY (LocationId)
);

-- Table: MariageStatus
CREATE TABLE MariageStatus (
    MarriageId int NOT NULL,
    Description varchar(10) NOT NULL,
    CONSTRAINT MariageStatus_pk PRIMARY KEY (MarriageId)
);

-- Table: MedicalDevices
CREATE TABLE MedicalDevices (
    DeviceId int NOT NULL,
    DeviceDescription int NOT NULL,
    PatientRoomSchId int NOT NULL,
    CONSTRAINT MedicalDevices_pk PRIMARY KEY (DeviceId)
);

-- Table: MedicineInfo
CREATE TABLE MedicineInfo (
    MedicineInfoId int NOT NULL,
    MedicineName varchar(20) NOT NULL,
    MedicineCost int NOT NULL,
    CONSTRAINT MedicineInfo_pk PRIMARY KEY (MedicineInfoId)
);

-- Table: PatientDetails
CREATE TABLE PatientDetails (
    PatientId int NOT NULL,
    FirstName varchar(30) NOT NULL,
    LastName varchar(30) NOT NULL,
    SSN varchar(30) NOT NULL,
    MarriageId int NOT NULL,
    AddressId int NOT NULL,
    ContactID int NOT NULL,
    CONSTRAINT PatientDetails_pk PRIMARY KEY (PatientId)
);

-- Table: PatientHealthRecord
CREATE TABLE PatientHealthRecord (
    PtHealthId int NOT NULL,
    Height int NOT NULL,
    Weight int NULL,
    HeartRate int NULL,
    DateOfRecord datetime NULL,
    PrimaryDoctor varchar(30) NOT NULL,
    PatientId int NOT NULL,
    PatientVisitID int NOT NULL,
    CONSTRAINT PatientHealthRecord_pk PRIMARY KEY (PtHealthId)
);

-- Table: PatientHospitalizationDetails
CREATE TABLE PatientHospitalizationDetails (
```

```
PatientVisitID int NOT NULL,
CheckInTime datetime NOT NULL,
CheckOutTime datetime NULL,
HospitalizationInfo varchar(50) NOT NULL,
PatientId int NOT NULL,
CONSTRAINT PatientHospitalizationDetails_pk PRIMARY KEY (PatientVisitID)
);

-- Table: PatientInsurance
CREATE TABLE PatientInsurance (
    PlanId int NOT NULL,
    InsurancePlanId int NOT NULL,
    PatientId int NOT NULL,
    CONSTRAINT PatientInsurance_pk PRIMARY KEY (PlanId,PatientId)
);

-- Table: PatientMedication
CREATE TABLE PatientMedication (
    PatientMedicationId int NOT NULL,
    Dosage varchar(20) NOT NULL,
    Quantity int NOT NULL,
    MedicineInfoId int NOT NULL,
    PatientId int NOT NULL,
    PtHealthId int NOT NULL,
    CONSTRAINT PatientMedication_pk PRIMARY KEY (PatientMedicationId)
);

-- Table: PatientProcedureHistory
CREATE TABLE PatientProcedureHistory (
    ProcedureHistoryId int NOT NULL,
    ProcedureName datetime NOT NULL,
    OperatedBy varchar(30) NOT NULL,
    ProcedureInfoId int NOT NULL,
    PatientId int NOT NULL,
    PtHealthId int NOT NULL,
    CONSTRAINT PatientProcedureHistory_pk PRIMARY KEY (ProcedureHistoryId)
);

-- Table: PatientRoomSchedule
CREATE TABLE PatientRoomSchedule (
    PatientRoomSchId int NOT NULL,
    OccupiedDate datetime NOT NULL,
    VacatedDate datetime NULL,
    RoomStatusId int NOT NULL,
    PatientId int NOT NULL,
    CONSTRAINT PatientRoomSchedule_pk PRIMARY KEY (PatientRoomSchId)
);

-- Table: PaymentMethod
CREATE TABLE PaymentMethod (
    PaymentMethodId int NOT NULL,
    PaymentMethodDesc varchar(30) NOT NULL,
    CONSTRAINT PaymentMethod_pk PRIMARY KEY (PaymentMethodId)
);

-- Table: Payor
CREATE TABLE Payor (
    PayorId int NOT NULL,
```

```
PayorDescription varchar(30) NOT NULL,  
CONSTRAINT Payor_pk PRIMARY KEY (PayorId)  
);  
  
-- Table: PlanCoverage  
CREATE TABLE PlanCoverage (  
    Coverage bigint NOT NULL,  
    InsurancePlanId int NOT NULL,  
    ProcedureInfoId int NOT NULL,  
    MedicineInfoId int NOT NULL,  
    CONSTRAINT PlanCoverage_pk PRIMARY KEY (InsurancePlanId)  
);  
  
-- Table: ProcedureInfo  
CREATE TABLE ProcedureInfo (  
    ProcedureInfoId int NOT NULL,  
    ProcedureName varchar(30) NOT NULL,  
    ProcedureCost int NOT NULL,  
    PtHealthId int NOT NULL,  
    PatientId int NOT NULL,  
    CONSTRAINT ProcedureInfo_pk PRIMARY KEY (ProcedureInfoId)  
);  
  
-- Table: Referredby  
CREATE TABLE Referredby (  
    ReferredBy int NOT NULL,  
    ReferredDoctorName varchar(30) NULL,  
    PatientVisitID int NOT NULL,  
    PatientId int NOT NULL,  
    CONSTRAINT Referredby_pk PRIMARY KEY (ReferredBy)  
);  
  
-- Table: RoomStatus  
CREATE TABLE RoomStatus (  
    RoomStatusId int NOT NULL,  
    StatusDescription int NOT NULL,  
    CONSTRAINT RoomStatus_pk PRIMARY KEY (RoomStatusId)  
);  
  
-- Table: Symtoms  
CREATE TABLE Symtoms (  
    SymtomId int NOT NULL,  
    SymtomDescription varchar(30) NOT NULL,  
    PatientVisitID int NOT NULL,  
    PatientId int NOT NULL,  
    CONSTRAINT Symtoms_pk PRIMARY KEY (SymtomId)  
);  
  
-- Table: Visitors  
CREATE TABLE Visitors (  
    VisitorId int NOT NULL,  
    FName varchar(20) NOT NULL,  
    LName varchar(20) NOT NULL,  
    DateOfBirth datetime NOT NULL,  
    InTime datetime NOT NULL,  
    OutTime datetime NOT NULL,  
    PatientId int NOT NULL,  
    CONSTRAINT Visitors_pk PRIMARY KEY (VisitorId)  
);
```

```
-- foreign keys
-- Reference: BedDetails_Location (table: BedDetails)
ALTER TABLE BedDetails ADD CONSTRAINT BedDetails_Location
    FOREIGN KEY (LocationId)
    REFERENCES Location (LocationId);

-- Reference: BedDetails_PatientRoomSchedule (table: BedDetails)
ALTER TABLE BedDetails ADD CONSTRAINT BedDetails_PatientRoomSchedule
    FOREIGN KEY (PatientRoomSchId)
    REFERENCES PatientRoomSchedule (PatientRoomSchId);

-- Reference: Billing_PatientDetails (table: Billing)
ALTER TABLE Billing ADD CONSTRAINT Billing_PatientDetails
    FOREIGN KEY (PatientId)
    REFERENCES PatientDetails (PatientId);

-- Reference: Billing_PaymentMethod (table: Billing)
ALTER TABLE Billing ADD CONSTRAINT Billing_PaymentMethod
    FOREIGN KEY (PaymentMethodId)
    REFERENCES PaymentMethod (PaymentMethodId);

-- Reference: Billing_Payor (table: Billing)
ALTER TABLE Billing ADD CONSTRAINT Billing_Payor
    FOREIGN KEY (PayorId)
    REFERENCES Payor (PayorId);

-- Reference: DepEmployees_Department (table: DepEmployees)
ALTER TABLE DepEmployees ADD CONSTRAINT DepEmployees_Department
    FOREIGN KEY (DepId)
    REFERENCES Department (DepId);

-- Reference: DepEmployees_EmployeeDetails (table: DepEmployees)
ALTER TABLE DepEmployees ADD CONSTRAINT DepEmployees_EmployeeDetails
    FOREIGN KEY (EmployeeId)
    REFERENCES EmployeeDetails (EmployeeId);

-- Reference: EmployeeDetails_AddressInformation (table: EmployeeDetails)
ALTER TABLE EmployeeDetails ADD CONSTRAINT EmployeeDetails_AddressInformation
    FOREIGN KEY (AddressId)
    REFERENCES AddressInformation (AddressId);

-- Reference: EmployeeDetails_ContactDetails (table: EmployeeDetails)
ALTER TABLE EmployeeDetails ADD CONSTRAINT EmployeeDetails_ContactDetails
    FOREIGN KEY (ContactID)
    REFERENCES ContactDetails (ContactID);

-- Reference: EmployeeDetails_EmployeeJobDesignation (table: EmployeeDetails)
ALTER TABLE EmployeeDetails ADD CONSTRAINT EmployeeDetails_EmployeeJobDesignation
    FOREIGN KEY (DesignationId)
    REFERENCES EmployeeJobDesignation (DesignationId);

-- Reference: EmployeeWorkSchedule_EmployeeDetails (table: EmployeeWorkSchedule)
ALTER TABLE EmployeeWorkSchedule ADD CONSTRAINT EmployeeWorkSchedule_EmployeeDetails
    FOREIGN KEY (EmployeeId)
    REFERENCES EmployeeDetails (EmployeeId);

-- Reference: EmployeeWorkSchedule_Location (table: EmployeeWorkSchedule)
```

```
ALTER TABLE EmployeeWorkSchedule ADD CONSTRAINT EmployeeWorkSchedule_Location
FOREIGN KEY (LocationId)
REFERENCES Location (LocationId);

-- Reference: InsurancePlan_InsuranceCompany (table: InsurancePlan)
ALTER TABLE InsurancePlan ADD CONSTRAINT InsurancePlan_InsuranceCompany
FOREIGN KEY (InsuranceCompanyId)
REFERENCES InsuranceCompany (InsuranceCompanyId);

-- Reference: JobHistory_EmployeeDetails (table: JobHistory)
ALTER TABLE JobHistory ADD CONSTRAINT JobHistory_EmployeeDetails
FOREIGN KEY (EmployeeId)
REFERENCES EmployeeDetails (EmployeeId);

-- Reference: JobHistory_EmployeeJobDesignation (table: JobHistory)
ALTER TABLE JobHistory ADD CONSTRAINT JobHistory_EmployeeJobDesignation
FOREIGN KEY (DesignationId)
REFERENCES EmployeeJobDesignation (DesignationId);

-- Reference: MedicalDevices_PatientRoomSchedule (table: MedicalDevices)
ALTER TABLE MedicalDevices ADD CONSTRAINT MedicalDevices_PatientRoomSchedule
FOREIGN KEY (PatientRoomSchId)
REFERENCES PatientRoomSchedule (PatientRoomSchId);
-- Reference: PatientDetails_AddressInformation (table: PatientDetails)
ALTER TABLE PatientDetails ADD CONSTRAINT PatientDetails_AddressInformation
FOREIGN KEY (AddressId)
REFERENCES AddressInformation (AddressId);

-- Reference: PatientDetails_ContactDetails (table: PatientDetails)
ALTER TABLE PatientDetails ADD CONSTRAINT PatientDetails_ContactDetails
FOREIGN KEY (ContactID)
REFERENCES ContactDetails (ContactID);

-- Reference: PatientDetails_MariageStatus (table: PatientDetails)
ALTER TABLE PatientDetails ADD CONSTRAINT PatientDetails_MariageStatus
FOREIGN KEY (MarriageId)
REFERENCES MariageStatus (MarriageId);

-- Reference: PatientHealthRecord_PatientDetails (table: PatientHealthRecord)
ALTER TABLE PatientHealthRecord ADD CONSTRAINT PatientHealthRecord_PatientDetails
FOREIGN KEY (PatientId)
REFERENCES PatientDetails (PatientId);

-- Reference: PatientHealthRecord_PatientVisitDetails (table: PatientHealthRecord)
ALTER TABLE PatientHealthRecord ADD CONSTRAINT PatientHealthRecord_PatientVisitDetails
FOREIGN KEY (PatientVisitID)
REFERENCES PatientHospitalizationDetails (PatientVisitID);

-- Reference: PatientInsurance_InsurancePlan (table: PatientInsurance)
ALTER TABLE PatientInsurance ADD CONSTRAINT PatientInsurance_InsurancePlan
FOREIGN KEY (InsurancePlanId)
REFERENCES InsurancePlan (InsurancePlanId);

-- Reference: PatientInsurance_PatientDetails (table: PatientInsurance)
ALTER TABLE PatientInsurance ADD CONSTRAINT PatientInsurance_PatientDetails
FOREIGN KEY (PatientId)
REFERENCES PatientDetails (PatientId);
```

```
-- Reference: PatientMedication_MedicineInfo (table: PatientMedication)
ALTER TABLE PatientMedication ADD CONSTRAINT PatientMedication_MedicineInfo
FOREIGN KEY (MedicineInfoId)
REFERENCES MedicineInfo (MedicineInfoId);

-- Reference: PatientMedication_PatientDetails (table: PatientMedication)
ALTER TABLE PatientMedication ADD CONSTRAINT PatientMedication_PatientDetails
FOREIGN KEY (PatientId)
REFERENCES PatientDetails (PatientId);

-- Reference: PatientMedication_PatientHealthRecord (table: PatientMedication)
ALTER TABLE PatientMedication ADD CONSTRAINT PatientMedication_PatientHealthRecord
FOREIGN KEY (PtHealthId)
REFERENCES PatientHealthRecord (PtHealthId);

-- Reference: PatientProcedureHistory_PatientDetails (table: PatientProcedureHistory)
ALTER TABLE PatientProcedureHistory ADD CONSTRAINT PatientProcedureHistory_PatientDetails
FOREIGN KEY (PatientId)
REFERENCES PatientDetails (PatientId);

-- Reference: PatientProcedureHistory_PatientHealthRecord (table:
PatientProcedureHistory)
ALTER TABLE PatientProcedureHistory ADD CONSTRAINT
PatientProcedureHistory_PatientHealthRecord
FOREIGN KEY (PtHealthId)
REFERENCES PatientHealthRecord (PtHealthId);

-- Reference: PatientProcedureHistory_ProcedureInfo (table: PatientProcedureHistory)
ALTER TABLE PatientProcedureHistory ADD CONSTRAINT PatientProcedureHistory_ProcedureInfo
FOREIGN KEY (ProcedureInfoId)
REFERENCES ProcedureInfo (ProcedureInfoId);

-- Reference: PatientRoomSchedule_PatientDetails (table: PatientRoomSchedule)
ALTER TABLE PatientRoomSchedule ADD CONSTRAINT PatientRoomSchedule_PatientDetails
FOREIGN KEY (PatientId)
REFERENCES PatientDetails (PatientId);

-- Reference: PatientRoomSchedule_RoomStatus (table: PatientRoomSchedule)
ALTER TABLE PatientRoomSchedule ADD CONSTRAINT PatientRoomSchedule_RoomStatus
FOREIGN KEY (RoomStatusId)
REFERENCES RoomStatus (RoomStatusId);

-- Reference: PatientVisitDetails_PatientDetails (table: PatientHospitalizationDetails)
ALTER TABLE PatientHospitalizationDetails ADD CONSTRAINT
PatientVisitDetails_PatientDetails
FOREIGN KEY (PatientId)
REFERENCES PatientDetails (PatientId);

-- Reference: PlanCoverage_InsurancePlan (table: PlanCoverage)
ALTER TABLE PlanCoverage ADD CONSTRAINT PlanCoverage_InsurancePlan
FOREIGN KEY (InsurancePlanId)
REFERENCES InsurancePlan (InsurancePlanId);

-- Reference: PlanCoverage_MedicineInfo (table: PlanCoverage)
ALTER TABLE PlanCoverage ADD CONSTRAINT PlanCoverage_MedicineInfo
FOREIGN KEY (MedicineInfoId)
REFERENCES MedicineInfo (MedicineInfoId);
```



```
-- Reference: PlanCoverage_ProcedureInfo (table: PlanCoverage)
ALTER TABLE PlanCoverage ADD CONSTRAINT PlanCoverage_ProcedureInfo
    FOREIGN KEY (ProcedureInfoId)
    REFERENCES ProcedureInfo (ProcedureInfoId);

-- Reference: ProcedureInfo_PatientDetails (table: ProcedureInfo)
ALTER TABLE ProcedureInfo ADD CONSTRAINT ProcedureInfo_PatientDetails
    FOREIGN KEY (PatientId)
    REFERENCES PatientDetails (PatientId);

-- Reference: ProcedureInfo_PatientHealthRecord (table: ProcedureInfo)
ALTER TABLE ProcedureInfo ADD CONSTRAINT ProcedureInfo_PatientHealthRecord
    FOREIGN KEY (PtHealthId)
    REFERENCES PatientHealthRecord (PtHealthId);

-- Reference: Referredby_PatientDetails (table: Referredby)
ALTER TABLE Referredby ADD CONSTRAINT Referredby_PatientDetails
    FOREIGN KEY (PatientId)
    REFERENCES PatientDetails (PatientId);

-- Reference: Referredby_PatientHospitalizationDetails (table: Referredby)
ALTER TABLE Referredby ADD CONSTRAINT Referredby_PatientHospitalizationDetails
    FOREIGN KEY (PatientVisitID)
    REFERENCES PatientHospitalizationDetails (PatientVisitID);

-- Reference: Symtoms_PatientDetails (table: Symtoms)
ALTER TABLE Symtoms ADD CONSTRAINT Symtoms_PatientDetails
    FOREIGN KEY (PatientId)
    REFERENCES PatientDetails (PatientId);

-- Reference: Symtoms_PatientHospitalizationDetails (table: Symtoms)
ALTER TABLE Symtoms ADD CONSTRAINT Symtoms_PatientHospitalizationDetails
    FOREIGN KEY (PatientVisitID)
    REFERENCES PatientHospitalizationDetails (PatientVisitID);

-- Reference: Visitors_PatientDetails (table: Visitors)
ALTER TABLE Visitors ADD CONSTRAINT Visitors_PatientDetails
    FOREIGN KEY (PatientId)
    REFERENCES PatientDetails (PatientId);
-----
```

Following is the script to insert dummy data into tables.

```
--Inserting data to EmployeeDetails
INSERT INTO EmployeeDetails VALUES (1, 'Eric', 'Dawson', 'BB345', 1, 10, 200)
INSERT INTO EmployeeDetails VALUES (2, 'Nikita', 'Deshpande', 'AA123', 1, 11, 300)
INSERT INTO EmployeeDetails VALUES (3, 'Nishtha', 'Kalra', 'CC678', 2, 12, 400)
INSERT INTO EmployeeDetails VALUES (4, 'Sid', 'Singh', 'DD789', 2, 13, 500)

--Inserting data to EmployeeJobDesignation
INSERT INTO EmployeeJobDesignation VALUES (1, 'Doctor')
INSERT INTO EmployeeJobDesignation VALUES (2, 'Nurse')
INSERT INTO EmployeeJobDesignation VALUES (3, 'Manager')
INSERT INTO EmployeeJobDesignation VALUES (4, 'Chief Surgeon')

--Inserting data into Department
INSERT INTO Department VALUES (11, 'Cardiology', 225)
INSERT INTO Department VALUES (22, 'Urology', 333)
INSERT INTO Department VALUES (33, 'Neurology', 443)
INSERT INTO Department VALUES (44, 'PathLab', 199)

--Inserting data into DepEmployees
INSERT INTO DepEmployees VALUES (1, 11)
INSERT INTO DepEmployees VALUES (2, 11)
INSERT INTO DepEmployees VALUES (3, 22)
INSERT INTO DepEmployees VALUES (4, 33)

--Inserting data into job history
INSERT INTO JobHistory VALUES (100, 'doctor', 2000, 01-04-2010, 'Good
Employee', 'temporary', 1, 1)
INSERT INTO JobHistory VALUES (200, 'doctor', 2000, 01-04-
2010, 'Excellent', 'permanent', 2, 1)
INSERT INTO JobHistory VALUES (300, 'doctor', 2000, 01-04-2010, 'Hardworking
Employee', 'contract based', 3, 2)
INSERT INTO JobHistory VALUES (400, 'doctor', 2000, 01-04-2010, 'Average rating', 'under
training', 4, 2)
UPDATE JobHistory SET StartDate='2010-01-04' , PastJob='doctor' where JobHistoryID=100
UPDATE JobHistory SET StartDate='2011-04-04' , PastJob='NA' where JobHistoryID=200
UPDATE JobHistory SET StartDate='2012-04-04' , PastJob='Student' where JobHistoryID=300
UPDATE JobHistory SET StartDate='2017-01-01' , PastJob='Student' where JobHistoryID=400

--Inserting data to Location
INSERT INTO Location VALUES (1, 1, 101)
INSERT INTO Location VALUES (2, 2, 201)
INSERT INTO Location VALUES (3, 3, 303)
INSERT INTO Location VALUES (4, 4, 404)

--Inserting data into AddressInformation
INSERT INTO AddressInformation VALUES (10, 'Apt1', 'Westcott St', 42311, 'NY', 'USA')
INSERT INTO AddressInformation VALUES (11, 'Apt3', 'ErieBlvd', 11111, 'NY', 'USA')
INSERT INTO AddressInformation VALUES (12, 'Apt5', 'DellSt', 32456, 'NY', 'USA')
INSERT INTO AddressInformation VALUES (13, 'Apt6', 'Lancaster', 54321, 'NY', 'USA')

INSERT INTO AddressInformation VALUES (14, 'Apt10', 'Westcott St', 42311, 'NY', 'USA')
INSERT INTO AddressInformation VALUES (15, 'Apt45', 'Newton Avenue', 324567, 'NY', 'USA')
INSERT INTO AddressInformation VALUES (16, 'Apt3', 'Franklin St', 324567, 'NY', 'USA')
INSERT INTO AddressInformation VALUES (17, 'Apt9', 'Ostrom Avenue', 54341, 'NY', 'USA')

--Inserting into contactdetails
INSERT INTO ContactDetails VALUES (200, '3159497745', 'abc@gmail.com')
INSERT INTO ContactDetails VALUES (300, '3156789934', 'abc@yahoo.com')
```

```

INSERT INTO ContactDetails VALUES (400, '3153335678', 'xyz@hotmail.com')
INSERT INTO ContactDetails VALUES (500, '3152348765', 'pqr@gmail.com')

INSERT INTO ContactDetails VALUES (600, '3159490000', 'patient1@gmail.com')
INSERT INTO ContactDetails VALUES (700, '8901234432', 'blahblah@yahoo.com')
INSERT INTO ContactDetails VALUES (800, '5555555555', 'test@hotmail.com')
INSERT INTO ContactDetails VALUES (900, '3150059900', 'ashwini@gmail.com')
--Inserting into EmployeeWorkSchedule
INSERT INTO EmployeeWorkSchedule VALUES (101, '6:00', '10:00', 1, 1)
INSERT INTO EmployeeWorkSchedule VALUES (102, '6:00', '18:00', 2, 1)
INSERT INTO EmployeeWorkSchedule VALUES (103, '7:00', '15:00', 3, 2)
INSERT INTO EmployeeWorkSchedule VALUES (104, '8:00', '20:00', 4, 3)

UPDATE EmployeeWorkSchedule SET ShiftStartTime='6:00', ShiftEndTime='10:00' where
EmployeeId=1

--Inserting data into roomstatus
INSERT INTO RoomStatus VALUES (1, 'Occupied')
INSERT INTO RoomStatus VALUES (2, 'Available')
ALTER TABLE RoomStatus ALTER COLUMN StatusDescription varchar(20)

--Inserting data into marriagestatus
INSERT INTO MariageStatus VALUES (1, 'Married')
INSERT INTO MariageStatus VALUES (2, 'Unmarried')
INSERT INTO MariageStatus VALUES (3, 'Divorced')
INSERT INTO MariageStatus VALUES (4, 'Widowed')

--Inserting into PatientDetails
INSERT INTO PatientDetails VALUES (1, 'Derek', 'Shepard', 'xx909', 1, 14, 600)
INSERT INTO PatientDetails VALUES (2, 'Amanda', 'Oberoi', 'BB865', 2, 15, 700)
INSERT INTO PatientDetails VALUES (3, 'Meredeth', 'Grey', 'AA276', 1, 16, 800)
INSERT INTO PatientDetails VALUES (4, 'Izzie', 'Stevens', 'NN584', 4, 17, 900)

--Inserting into PatientHospitalizationDetails
INSERT INTO PatientHospitalizationDetails VALUES (201, 01-02-2017, 01-04-2017, 'Admitted
for head injury', 1)
INSERT INTO PatientHospitalizationDetails VALUES (202, 05-07-2017, 05-13-2017, 'Operated for
appendicitis.Due for followup', 2)
INSERT INTO PatientHospitalizationDetails VALUES (203, 11-02-2017, 11-02-2017, 'Regular
annual checkup.Patient is healthy', 3)
INSERT INTO PatientHospitalizationDetails VALUES (204, 08-02-2017, 08-02-2017, 'Conducted
bloodtests.Awaited reports', 4)
UPDATE PatientHospitalizationDetails SET CheckInTime='2017-01-02', CheckOutTime='2017-01-
04' where PatientId=1
UPDATE PatientHospitalizationDetails SET CheckInTime='2017-05-07', CheckOutTime='2017-05-
13' where PatientId=2
UPDATE PatientHospitalizationDetails SET CheckInTime='2017-11-02', CheckOutTime='2017-11-
02' where PatientId=3
UPDATE PatientHospitalizationDetails SET CheckInTime='2017-08-02', CheckOutTime='2017-08-
02' where PatientId=4
--Inserting into PatientHealthRecord
INSERT INTO PatientHealthRecord VALUES (100, 165, 70, 84, 01-02-2017, 'Dr.Eric', 1, 201)
INSERT INTO PatientHealthRecord VALUES (200, 195, 90, 75, 05-07-2017, 'Dr.Nikita', 2, 202)
INSERT INTO PatientHealthRecord VALUES (300, 155, 100, 79, 11-02-2017, 'Dr.Eric', 3, 203)
INSERT INTO PatientHealthRecord VALUES (400, 185, 55, 80, 08-02-2017, 'Dr.Eric', 4, 204)
UPDATE PatientHealthRecord SET DateOfRecord='2017-01-02' where PtHealthId=100

```

```

UPDATE PatientHealthRecord SET DateOfRecord='2017-05-07' where PtHealthId=200
UPDATE PatientHealthRecord SET DateOfRecord='2017-11-02' where PtHealthId=300
UPDATE PatientHealthRecord SET DateOfRecord='2017-08-02' where PtHealthId=400
--Inserting into PatientRoomSchedule
INSERT INTO PatientRoomSchedule VALUES (111,01-02-2017,01-04-2017,2,1)
INSERT INTO PatientRoomSchedule VALUES (222,05-07-2017,05-13-2017,2,2)
INSERT INTO PatientRoomSchedule VALUES (333,11-02-2017,11-02-2017,2,3)
INSERT INTO PatientRoomSchedule VALUES (444,08-02-2017,08-02-2017,2,4)
UPDATE PatientRoomSchedule SET OccupiedDate='2017-01-02', VacatedDate='2017-01-04' where
PatientId=1
UPDATE PatientRoomSchedule SET OccupiedDate='2017-05-07',VacatedDate='2017-05-13' where
PatientId=2
UPDATE PatientRoomSchedule SET OccupiedDate='2017-11-02',VacatedDate='2017-11-02' where
PatientId=3
UPDATE PatientRoomSchedule SET OccupiedDate='2017-08-02',VacatedDate='2017-08-02'where
PatientId=4
SELECT * FROM PatientRoomSchedule
--Inserting into ProcedureInfo
INSERT INTO ProcedureInfo VALUES (1,'Angioplasty',1000,100,1)
INSERT INTO ProcedureInfo VALUES (2,'Liver Transplant',1500,200,2)
INSERT INTO ProcedureInfo VALUES (3,'Liposuction',1900,300,3)
INSERT INTO ProcedureInfo VALUES (4,'CT Scan',500,400,4)
--Inserting into PatientProcedureHistory
ALTER TABLE PatientProcedureHistory DROP COLUMN ProcedureName
INSERT INTO PatientProcedureHistory VALUES (101,'Dr. Eric',1,1,100)
INSERT INTO PatientProcedureHistory VALUES (102,'Dr. Nikita',2,2,200)
INSERT INTO PatientProcedureHistory VALUES (103,'Dr.Amanda',3,3,300)
INSERT INTO PatientProcedureHistory VALUES (104,'Dr.Steve',4,4,400)
SELECT * FROM PatientProcedureHistory
--Inserting into MedicineInfo
INSERT INTO MedicineInfo VALUES (1001,'Paracetamol',200)
INSERT INTO MedicineInfo VALUES (1002,'Azithromicin',40)
INSERT INTO MedicineInfo VALUES (1003,'Dexona',100)
INSERT INTO MedicineInfo VALUES (1004,'Doxycylin',40)
--Inserting into PatientMedication
INSERT INTO PatientMedication VALUES (1,'three times a day','20ml',1001,1,100)
INSERT INTO PatientMedication VALUES (2,'two times a day','10ml',1002,2,200)
INSERT INTO PatientMedication VALUES (3,'once a day','1 tablet',1003,3,300)
INSERT INTO PatientMedication VALUES (4,'three times a day','2 capsules',1004,4,400)
--Inserting into InsuranceCompany
ALTER TABLE InsuranceCompany ALTER COLUMN Phone varchar(20)
INSERT INTO InsuranceCompany VALUES (11,'Aetna','3159457712','aetna@gmail.com')
INSERT INTO InsuranceCompany VALUES (12,'United
Healthcare','3159457712','info@unitedhealth.com')
INSERT INTO InsuranceCompany VALUES (13,'Obamacare','3224569900','info@obamacare.com')
INSERT INTO InsuranceCompany VALUES (14,'LIC','3159434455','info@lic.com')

--Inserting into InsurancePlan
INSERT INTO InsurancePlan VALUES (111,'Aetna Life Cover',11)
INSERT INTO InsurancePlan VALUES (122,'United Advantage Plan',12)
INSERT INTO InsurancePlan VALUES (133,'Obamacare Protection',13)
INSERT INTO InsurancePlan VALUES (144,'LIC HealthInsurance',14)
--Inserting into PlanCoverage
INSERT INTO PlanCoverage VALUES (10000,111,1,1001)
INSERT INTO PlanCoverage VALUES (20000,122,2,1002)
INSERT INTO PlanCoverage VALUES (35000,133,3,1003)
INSERT INTO PlanCoverage VALUES (100000,144,4,1004)
--Inserting into PatientInsurance

```

```

INSERT INTO PatientInsurance VALUES (1000,111,1)
INSERT INTO PatientInsurance VALUES (1001,122,2)
INSERT INTO PatientInsurance VALUES (1002,133,3)
INSERT INTO PatientInsurance VALUES (1003,144,4)

--Inserting into PaymentMethod
INSERT INTO PaymentMethod VALUES (1,'Cash')
INSERT INTO PaymentMethod VALUES (2,'Card')
INSERT INTO PaymentMethod VALUES (3,'Cheque')

--Inserting into Payor
INSERT INTO Payor VALUES (1,'Self')
INSERT INTO Payor VALUES (2,'Insurance Company')
INSERT INTO Payor VALUES (3,'Relative/Family')

--Inserting into Billing
INSERT INTO Billing VALUES (1,2000,1,1,1)
INSERT INTO Billing VALUES (2,5000,2,2,3)
INSERT INTO Billing VALUES (3,7000,3,2,3)
INSERT INTO Billing VALUES (4,1000,4,3,2)

--Inserting into Visitors
INSERT INTO Visitors VALUES (1,'John','Parker',01-03-1999,'4:00','6:00',1)
INSERT INTO Visitors VALUES (2,'Jane','Dawson',03-05-1994,'16:00','19:00',1)
INSERT INTO Visitors VALUES (3,'Tina','Baker',06-07-1990,'15:00','16:00',2)
INSERT INTO Visitors VALUES (4,'Rani','Desai',02-04-1993,'8:00','9:00',3)
UPDATE Visitors SET DateOfBirth=1999-01-03 where VisitorId=1
UPDATE Visitors SET DateOfBirth=1994-03-05 where VisitorId=2
UPDATE Visitors SET DateOfBirth=1990-06-07 where VisitorId=3
UPDATE Visitors SET DateOfBirth=1993-02-04 where VisitorId=4

--Inserting into Symtoms
INSERT INTO Symtoms VALUES (1,'High Fever.Heart burn',201,1)
INSERT INTO Symtoms VALUES (2,'Head ache.Pain in abdomen',202,2)
INSERT INTO Symtoms VALUES (3,'High Blood pressure',203,3)
INSERT INTO Symtoms VALUES (4,'Stomach pain.Weakenss',204,4)

--Inserting into Referred by
INSERT INTO Referredby VALUES (1,'Dr.Emily',201,1)
INSERT INTO Referredby VALUES (2,'Dr.Baker',201,2)
INSERT INTO Referredby VALUES (3,'Dr.Jaret',201,3)
INSERT INTO Referredby VALUES (4,'Dr.Tiffany',201,4)

--Inserting into MedicalDevices
ALTER TABLE MedicalDevices ALTER COLUMN DeviceDescription varchar(20)
INSERT INTO MedicalDevices VALUES (1,'Oxygen Cylinder',111)
INSERT INTO MedicalDevices VALUES (2,'Heart Monitor',222)
INSERT INTO MedicalDevices VALUES (3,'Syringes',333)
INSERT INTO MedicalDevices VALUES (4,'O2 Monitor',444)

--Insert Into BedDetails
ALTER TABLE BedDetails ALTER COLUMN BedType varchar(20)
INSERT INTO BedDetails VALUES (1,'Electric bed',111,1)
INSERT INTO BedDetails VALUES (2,'Strecher',222,2)
INSERT INTO BedDetails VALUES (3,'Clinitron bed',333,3)
INSERT INTO BedDetails VALUES (4,'Low Height bed',444,4)

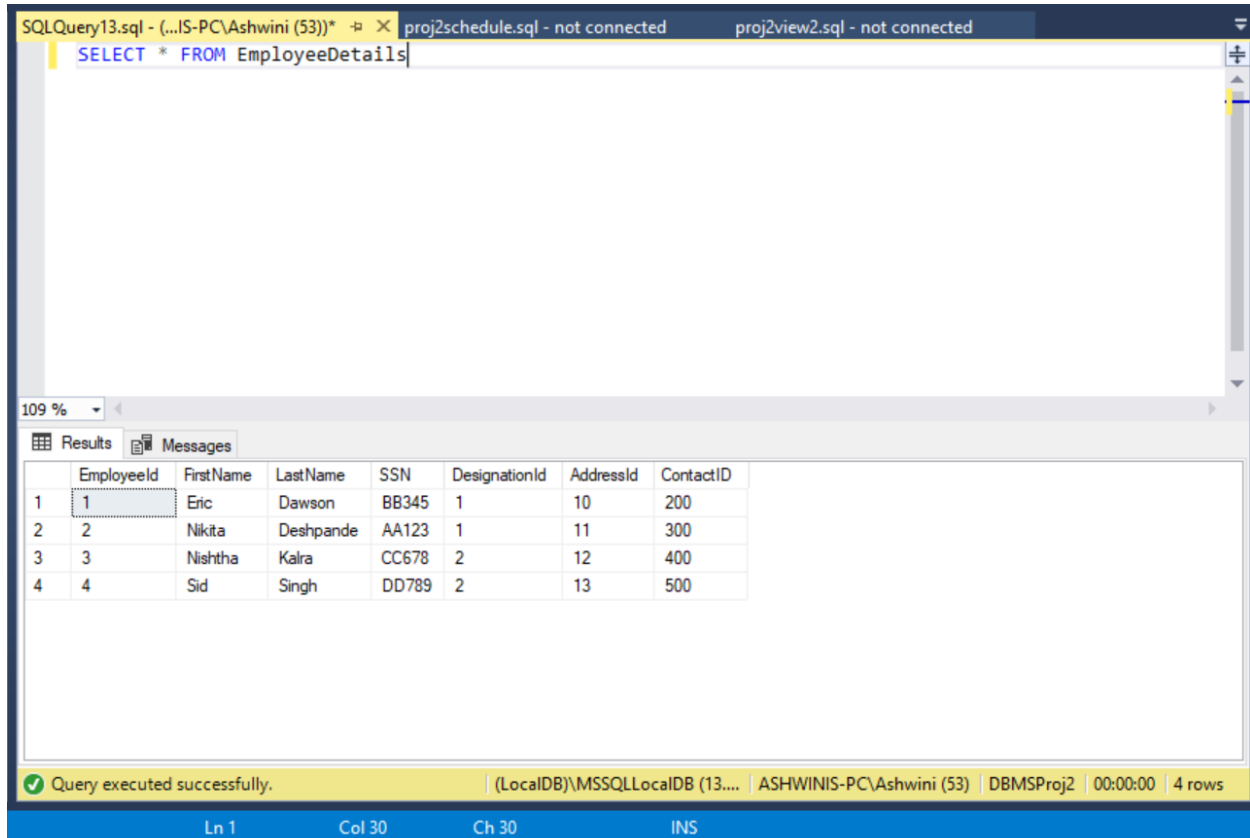
```

Security Considerations

- 1) For the system to be secure, there should be secured access to database either by Windows authentication or Sever Authentication.
- 2) Different roles should be declared with different permissions to create protection for the data
- 3) For example, only the billing department should get access to the patient insurance details and no one else.
- 4) The patient medical records are confidential information and the permission to access it should be given only to the doctor and no one else
- 5) The employee schedule should be visible only to the concerned employee,
- 6) The database administrator should define these roles and grant specific permissions to each role.

Testing

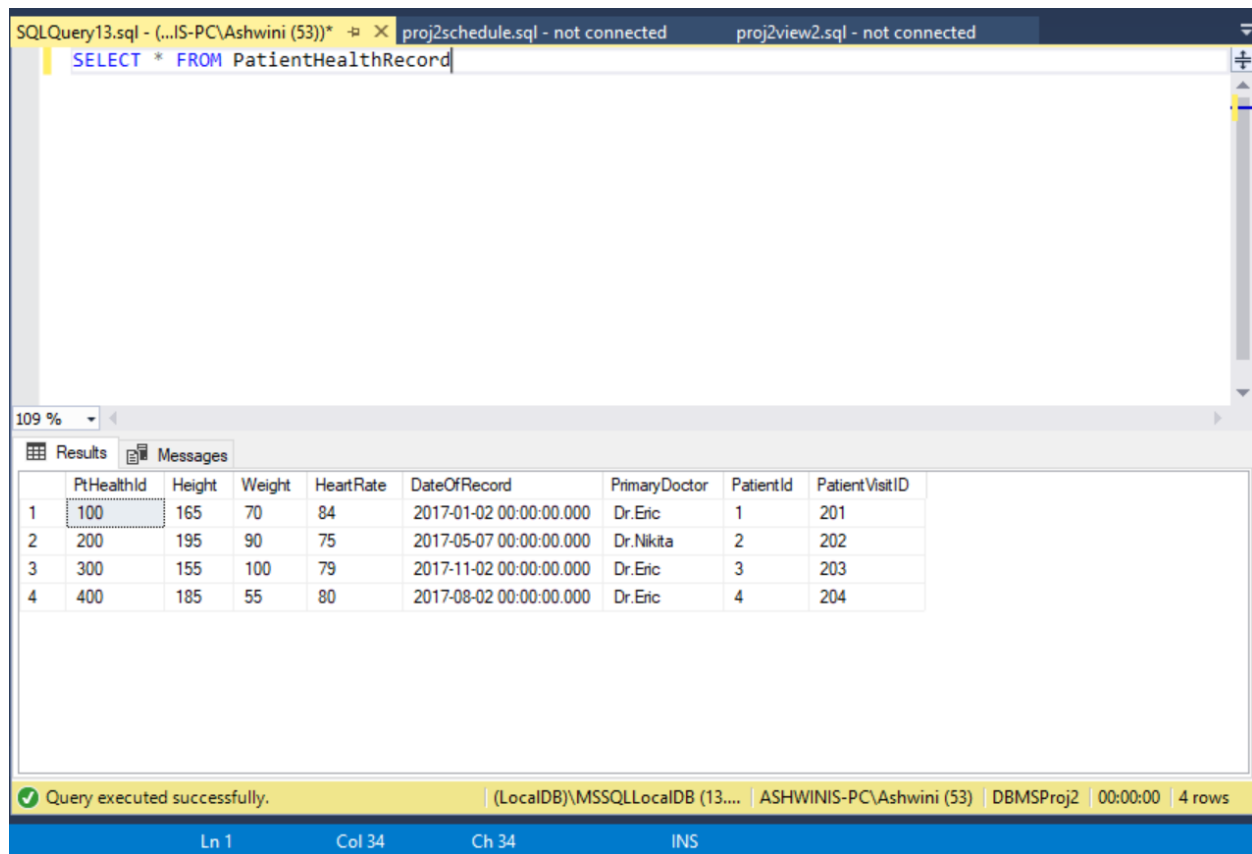
Following are few screen shot just to check of the insertion of dummy data into the database was properly done.



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a query window with the text `SELECT * FROM EmployeeDetails`. The bottom pane shows the results of the query, which is a table with 8 columns: EmployeeId, FirstName, LastName, SSN, DesignationId, AddressId, and ContactID. The table contains 4 rows of data. The status bar at the bottom indicates that the query was executed successfully and returned 4 rows.

	EmployeeId	FirstName	LastName	SSN	DesignationId	AddressId	ContactID
1	1	Eric	Dawson	BB345	1	10	200
2	2	Nikita	Deshpande	AA123	1	11	300
3	3	Nishtha	Kalra	CC678	2	12	400
4	4	Sid	Singh	DD789	2	13	500

The above screenshot shows data in the employee details table.



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the query: `SELECT * FROM PatientHealthRecord`. The bottom pane shows the results of the query, which is a table with 9 columns: PtHealthId, Height, Weight, HeartRate, DateOfRecord, PrimaryDoctor, PatientId, and PatientVisitId. The table contains 4 rows of data. The status bar at the bottom indicates that the query was executed successfully, returning 4 rows.

	PtHealthId	Height	Weight	HeartRate	DateOfRecord	PrimaryDoctor	PatientId	PatientVisitId
1	100	165	70	84	2017-01-02 00:00:00.000	Dr.Eric	1	201
2	200	195	90	75	2017-05-07 00:00:00.000	Dr.Nikita	2	202
3	300	155	100	79	2017-11-02 00:00:00.000	Dr.Eric	3	203
4	400	185	55	80	2017-08-02 00:00:00.000	Dr.Eric	4	204

This screenshot shows patient details.

In the following section certain views, functions and stored procedures are developed to ease the data retrieval from the database

- 1) **Stored Procedure** to find medicine prescribed and dosage for a patient. The doctor prescribing the medicine is also retrieved. This procedure can be used by the nurses to schedule the medicine dosage to the patient. They also have the doctor information if they need any help regarding the prescription.

```
CREATE PROCEDURE MedicineInformation @patid int
AS
BEGIN

    SELECT m.MedicineName, pm.Dosage, pm.Quantity, ph.PrimaryDoctor
    FROM
    MedicineInfo m
    JOIN
    PatientMedication pm on m.MedicineInfoId=pm.MedicineInfoId
    JOIN
    PatientHealthRecord ph on ph.PatientId=pm.PatientId
    WHERE ph.PatientId= @patid

END

EXEC MedicineInformation 2
```

The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows the SQL script for the 'MedicineInformation' stored procedure, which is designed to retrieve medicine details for a specific patient. The script includes a CREATE PROCEDURE statement, followed by a SELECT query that joins the MedicineInfo, PatientMedication, and PatientHealthRecord tables. The query filters results by the patient ID (@patid). Below the script, the 'EXEC MedicineInformation 2' command is highlighted. The bottom pane shows the results of the query execution, displaying a single row of data for patient 2. The status bar at the bottom indicates that the query was executed successfully, returning 1 row.

	MedicineName	Dosage	Quantity	PrimaryDoctor
1	Azithromicin	two times a day	10ml	Dr.Nikita

Query executed successfully. (LocalDB)\MSSQLLocalDB (13.... ASHWINI-PC\Ashwini (52) DBMSProj2 00:00:00 1 rows

- 2) The following **stored procedure** gives information about the employee schedule and the shiftstart and end timings along with the location to work.

It accepts the first name and last name of the employee as input parameters.

It can be used by the staff to find their working schedule

```
CREATE PROC sp_Schedule @FirstName varchar(30), @LastName varchar(30)
AS
SELECT ed.FirstName,ed.LastName,ej.JobTitle,
es.ShiftStartTime,es.ShiftEndTime,loc.FloorNumber,loc.RoomNumber
FROM
EmployeeDetails ed join EmployeeJobDesignation ej
on ed.EmployeeId=ej.DesignationId
join EmployeeWorkSchedule es
on es.EmployeeId=ed.EmployeeId
join Location loc on loc.LocationId=es.LocationId
WHERE
ed.FirstName=@FirstName AND ed.LastName=@LastName
```

The screenshot shows a SQL query editor with the following content:

```
SQLQuery13.sql - (...IS-PC\Ashwini (53))*   proj2sp.sql - not connected   proj2schedule.sql - not connected
CREATE PROC sp_Schedule @FirstName varchar(30), @LastName varchar(30)
AS
SELECT ed.FirstName,ed.LastName,ej.JobTitle,
es.ShiftStartTime,es.ShiftEndTime,loc.FloorNumber,loc.RoomNumber
FROM
EmployeeDetails ed join EmployeeJobDesignation ej
on ed.EmployeeId=ej.DesignationId
join EmployeeWorkSchedule es
on es.EmployeeId=ed.EmployeeId
join Location loc on loc.LocationId=es.LocationId
WHERE
ed.FirstName=@FirstName AND ed.LastName=@LastName

EXEC sp_Schedule 'Eric' , 'Dawson'
```

The results pane shows a single row of data:

	FirstName	LastName	Job Title	Shift Start Time	Shift End Time	Floor Number	Room Number
1	Eric	Dawson	Doctor	06:00:00.0000000	10:00:00.0000000	1	101

The status bar at the bottom indicates "Disconnected." and shows the current position as Ln 14, Col 34, Ch 34. The bottom right corner shows "Project 2(3).pdf - Adobe Acrobat Reader DC".

- 3) The following **view** can be used by the billing department to find the insurance information of a particular patient and the coverage amount .
 They can also get the contact information of the insurance company to process the bill.
 Only this information is visible to the billing department without having to look up through the patient records as they are confidential.

```
CREATE VIEW PatientInsuranceInformation
AS
Select pd.PatientId,pd.FirstName,pd.LastName,
ip.PlanName,ic.CompanyName,ic.EmailId,ic.Phone,pc.Coverage
FROM
PatientDetails pd join PatientInsurance pi on pd.PatientId=pi.PatientId
join InsurancePlan ip on ip.InsurancePlanId=pi.InsurancePlanId
join InsuranceCompany ic on ic.InsuranceCompanyId=ip.InsuranceCompanyId
join PlanCoverage pc on pc.InsurancePlanId=pi.InsurancePlanId
```

The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows the SQL script for creating the view `PatientInsuranceInformation`. The script uses `SELECT` to retrieve patient details, insurance plan information, and coverage amounts, joined together. The bottom pane shows the results of the query, which is a table with 9 columns: `PatientId`, `FirstName`, `LastName`, `PlanName`, `CompanyName`, `EmailId`, `Phone`, and `Coverage`. The results table contains 4 rows of data.

	PatientId	FirstName	LastName	PlanName	CompanyName	EmailId	Phone	Coverage
1	1	Derek	Shepard	Aetna Life Cover	Aetna	aetna@gmail.com	3159457712	10000
2	2	Amanda	Oberoi	United Advantage Plan	United Healthcare	info@unitedhealth.com	3159457712	20000
3	3	Meredeth	Grey	Obamacare Protection	Obamacare	info@obamacare.com	3224569900	35000
4	4	Izzie	Stevens	LIC HealthInsurance	LIC	info@lic.com	3159434455	100000

Query executed successfully. (LocalDB)\MSSQLLocalDB (13.... ASHWINI-PC\Ashwini (53) DBMSPProj2 00:00:00 4 rows

- 4) The following **view** can be used by the scheduling department for checking if the room is occupied or vacant. They can schedule another patient if the room is available. No information about the patient is visible to the staff and patient privacy is maintained.

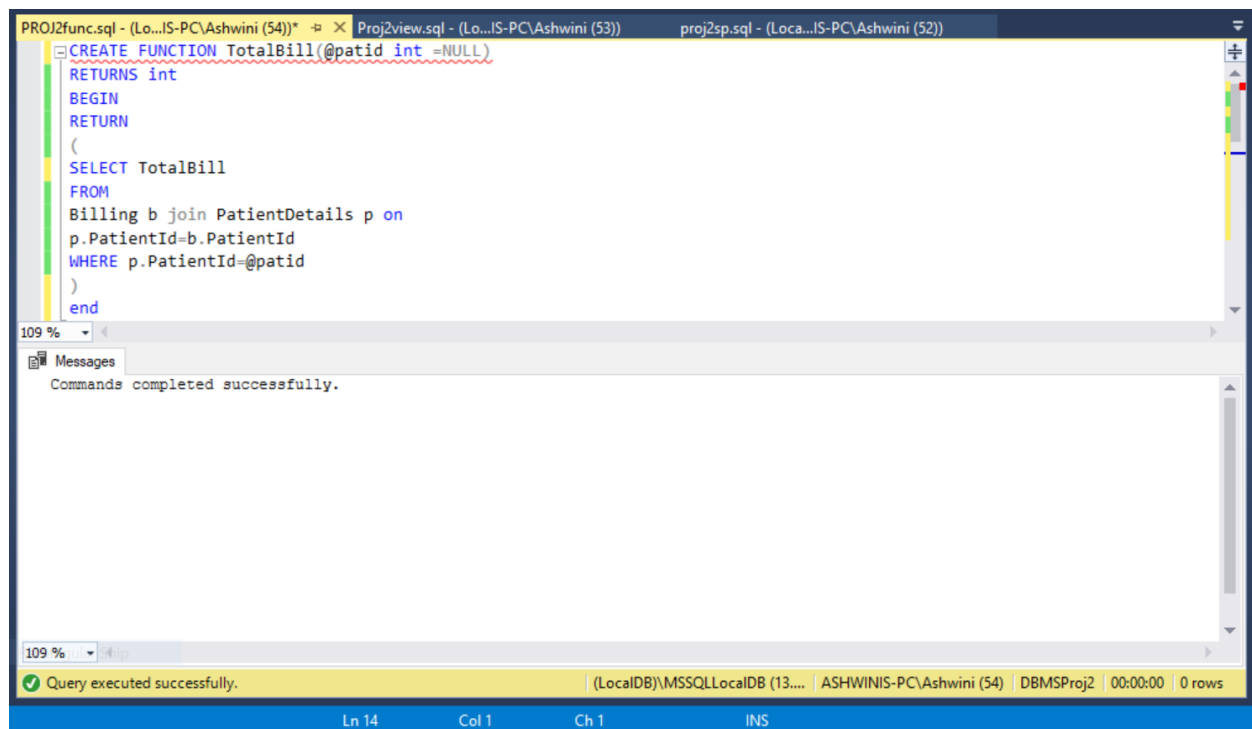
```
Create view Roomavailability
AS
SELECT ps.PatientRoomSchId AS 'Room',
rs.StatusDescription,
ps.VacatedDate
FROM
PatientRoomSchedule ps join RoomStatus rs
on
ps.RoomStatusId=rs.RoomStatusId
```

The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the SQL script for creating a view named 'Roomavailability'. The script selects 'PatientRoomSchId' (aliased as 'Room'), 'StatusDescription', and 'VacatedDate' from the 'PatientRoomSchedule' and 'RoomStatus' tables, joined on 'RoomStatusId'. The bottom pane shows the results of the query, which is a table with 4 rows and 4 columns: 'Room', 'StatusDescription', 'VacatedDate', and an implicit column for the join key. The status bar at the bottom indicates 'Query executed successfully.' and shows 4 rows.

	Room	StatusDescription	VacatedDate
1	111	Available	2017-01-04 00:00:00.000
2	222	Available	2017-05-13 00:00:00.000
3	333	Available	2017-11-02 00:00:00.000
4	444	Available	2017-08-02 00:00:00.000

- 5) The **following function** creates a utility to retrieve the bill generated for a patient. This function accepts patientID as the input parameter.

```
CREATE FUNCTION TotalBill(@patid int =NULL)
RETURNS int
BEGIN
RETURN
(
SELECT TotalBill
FROM
Billing b join PatientDetails p on
p.PatientId=b.PatientId
WHERE p.PatientId=@patid
)
end
```



This function is used in the **script** below to generate information about the procedure the patient went through and the cost that he owes to the hospital. This will create an easy bill generation process

```
USE DBMSProj2
GO
DECLARE @bill int
EXEC @bill= TotalBill @patid=1
SELECT
p.FirstName,p.LastName,p.ProcedureName,@bill AS TotalBill
FROM
PatientDetails p JOIN PatientProcedureHistory ph
on
p.PatientId=ph.PatientId
```

```
join ProcedureInfo pi on ph.ProcedureInfoId=pi.ProcedureInfoId
join Billing b on p.PatientId=b.PatientId
WHERE b.TotalBill=@bill
```

The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows the execution of a query in the 'proj2func.sql' file. The query uses the stored procedure 'DBMSProj2' with parameters '@bill' and '@patid'. The bottom pane shows the results of the query, which is a single row with the following data:

First Name	Last Name	Procedure Name	Total Bill
Derek	Shepard	Angioplasty	2000

The status bar at the bottom indicates that the query was executed successfully, returning 1 row in 00:00:00 seconds.

Conclusion

This database contains and covers all the scenarios that can be generated in any healthcare organization. The testcases generated above show that information can be retrieved from this database in a systematic manner.

Digitization of such potentially huge information will make it more secure and organized. We can further improvise this design by improving its security aspects. A strong backend model boosts the front end and makes the system fast and efficient.

Remarks

This project gave me solid understanding of all the concepts covered so far in the course and gave a chance to implement the design by modelling real life entities. It helped me in understanding the process of designing and implementing a entire database based on user requirements.