**PART – 1**

**DDL Commands to create tables**

1. **Patient**Schema: Patient (PatientID: integer, BloodTypeID: integer, Name: string, Age: integer, Gender: string)  
   CREATE TABLE Patient (  
    PatientID SERIAL PRIMARY KEY,  
    BloodTypeID INTEGER REFERENCES BloodGroup(BloodTypeID),  
    Name VARCHAR(50),  
    Age INTEGER,  
    Gender VARCHAR(10)  
   );

A screen shot of a computer

Description automatically generated

1. **BloodGroup**Schema: BloodGroup (BloodTypeID: integer, BloodType: string)  
     
   CREATE TABLE BloodGroup (  
    BloodTypeID SERIAL PRIMARY KEY,  
    BloodType VARCHAR(5) UNIQUE  
   );

A black screen with white text

Description automatically generated

1. **InsuranceProvider**  
   Schema: InsuranceProvider (InsuranceProviderID: integer, PatientID: integer, InsuranceProviderName: string)  
     
   CREATE TABLE InsuranceProvider (  
    InsuranceProviderID SERIAL PRIMARY KEY,  
    PatientID INTEGER REFERENCES Patient(PatientID),  
    InsuranceProviderName VARCHAR(50)  
   );

A black screen with white text

Description automatically generated

1. **Admission**Schema: Admission (AdmissionID: integer, PatientID: integer, DoctorID: integer, HospitalID: integer, MedicationID: integer, AdmissionDate: date, DischargeDate: date)  
     
   CREATE TABLE Admission (  
    AdmissionID SERIAL PRIMARY KEY,  
    PatientID INTEGER REFERENCES Patient(PatientID),  
    DoctorID INTEGER REFERENCES Doctor(DoctorID),  
    HospitalID INTEGER REFERENCES Hospital(HospitalID),  
    MedicationID INTEGER REFERENCES Medication(MedicationID),  
    AdmissionDate DATE,  
    DischargeDate DATE  
   );

A computer screen with white text

Description automatically generated

1. **AdmissionType**Schema: AdmissionType (AdmissionTypeID: integer, AdmissionID: integer, AdmissionType: string)  
     
   CREATE TABLE AdmissionType (

AdmissionTypeID SERIAL PRIMARY KEY,

AdmissionID INTEGER REFERENCES Admission(AdmissionID),

AdmissionType VARCHAR(20)

);

**A black screen with white text

Description automatically generated**

1. **Billing**Schema: Billing (BillingID: integer, AdmissionID: integer, BillingAmount: real)  
     
   CREATE TABLE Billing (

BillingID SERIAL PRIMARY KEY,

AdmissionID INTEGER REFERENCES Admission(AdmissionID),

BillingAmount DECIMAL(10,2)

);

**A black screen with white text

Description automatically generated**

1. **TestResult**

Schema: TestResult (AdmissionID: integer, TestResults: string)  
  
CREATE TABLE TestResult (

AdmissionID INTEGER REFERENCES Admission(AdmissionID),

TestResults TEXT,

PRIMARY KEY (AdmissionID)

);

**A black screen with white text

Description automatically generated**

1. **Hospital**

Schema: Hospital (HospitalID: integer, HospitalName: string)  
  
CREATE TABLE Hospital (

HospitalID SERIAL PRIMARY KEY,

HospitalName VARCHAR(50) NOT NULL

);

A black screen with white text

Description automatically generated

1. **Room**

Schema: Room (RoomID: integer, HospitalID: integer, RoomNumber: integer)  
  
CREATE TABLE Room (

RoomID SERIAL PRIMARY KEY,

HospitalID INTEGER REFERENCES Hospital(HospitalID),

RoomNumber INTEGER

);

A black screen with white text

Description automatically generated

1. **Doctor**

Schema: Doctor (DoctorID: integer, HospitalID: integer, DoctorName: string)  
  
CREATE TABLE Doctor (

DoctorID SERIAL PRIMARY KEY,

HospitalID INTEGER REFERENCES Hospital(HospitalID),

DoctorName VARCHAR(50) NOT NULL

);

A black screen with white text

Description automatically generated

1. **Medication**

Schema: Medication (MedicationID: integer, DiagnosisID: integer, MedicineName: string)  
CREATE TABLE Medication (

MedicationID SERIAL PRIMARY KEY,

DiagnosisID INTEGER REFERENCES Diagnosis(DiagnosisID),

MedicineName VARCHAR(50) NOT NULL

);

A black screen with white text

Description automatically generated

1. **Diagnosis**

Schema: Diagnosis (DiagnosisID: integer, MedicalCondition: string)  
  
CREATE TABLE Diagnosis (

DiagnosisID SERIAL PRIMARY KEY,

MedicalCondition VARCHAR(100) NOT NULL

);

A black screen with white text

Description automatically generated

1. **DiagAdm (Associate Entity)**

Schema: DiagAdm (AdmissionID: integer, AdmissionID: integer)  
  
CREATE TABLE DiagAdm (

AdmissionID INTEGER REFERENCES Admission(AdmissionID),

AdmissionID INTEGER REFERENCES Diagnosis(DiagnosisID),

PRIMARY KEY (AdmissionID, DiagnosisID)

);

A black screen with white text

Description automatically generated

**DML Commands to display all the rows inserted into the tables**

**A black screen with white text

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screen shot of a computer

Description automatically generated

**PART – 2**

The tables are already in the BCNF form.

Justification:

1. Patient:
   1. The primary key (PatientID) uniquely identifies each patient.
   2. All attributes (Name, Age, Gender) depend on the entire primary key, not just a part of it.
2. BloodGroup:
   1. The primary key (BloodTypeID) uniquely identifies each blood group.
   2. BloodType depends on the entire primary key.
3. InsuranceProvider:
   1. The primary key (InsuranceProviderID) uniquely identifies each provider for each patient.
   2. PatientID (foreign key) depends on the whole primary key (ensures a provider is associated with a specific patient).
   3. InsuranceProviderName depends on the whole primary key (indirectly through PatientID).
4. Admission:
   1. AdmissionID uniquely identifies each admission record.
   2. All foreign keys (PatientID, DoctorID, HospitalID, MedicationID) depend on the entire AdmissionID, not just PatientID.
5. AdmissionType:
   1. AdmissionType depends on the Admission (via AdmissionID as foreign key). The dependency chain goes through the entire AdmissionID for AdmissionType to be valid.
6. Billing:
   1. The primary key (BillingID) uniquely identifies each billing record.
   2. AdmissionID (foreign key) depends on the whole primary key (ensures a billing is associated with a specific admission).
7. TestResult:
   1. The foreign key (AdmissionID) depends on the whole primary key of Admission (assuming Admission is in BCNF).
8. Hospital:
   1. The primary key (HospitalID) uniquely identifies each hospital.
   2. HospitalName depends on the whole primary key.
9. Room:
   1. The primary key (RoomID) uniquely identifies each room.
   2. HospitalID (foreign key) depends on the whole primary key (ensures a room belongs to a specific hospital).
10. Doctor:
    1. DoctorID uniquely identifies each doctor.
    2. Since a doctor can only work in one hospital (assumption according to the dataset), HospitalID directly depends on the entire DoctorID (no partial dependency on PatientID from Admission).
11. Medication:
    1. MedicationID uniquely identifies each medication.
    2. DiagnosisID (foreign key) as a foreign key referencing the Diagnosis table. Since a medication is specific to a diagnosis (assumption according to the dataset), DiagnosisID directly depends on the entire MedicationID (no partial dependency on PatientID from Admission).
12. Diagnosis:
    1. The primary key (DiagnosisID) uniquely identifies each diagnosis.
    2. MedicalCondition depends on the whole primary key.
13. DiagAdm (Associative Entity):
    1. By definition, an associative entity has a composite primary key that uniquely identifies each relationship between the participating entities (Admission and Diagnosis in this case).

Here are all the relations (in BCNF) form:

1. Patient:
   1. PatientID (Primary Key)
   2. BloodTypeID (Foreign Key references BloodGroup)
   3. Name
   4. Age
   5. Gender
2. BloodGroup:
   1. BloodTypeID (Primary Key)
   2. BloodType
3. InsuranceProvider:
   1. InsuranceProviderID (Primary Key)
   2. PatientID (Foreign Key references Patient)
   3. InsuranceProviderName
4. Admission:
   1. AdmissionID (Primary Key)
   2. PatientID (Foreign Key references Patient)
   3. DoctorID (Foreign Key references Doctor)
   4. HospitalID (Foreign Key references Hospital)
   5. MedicationID (Foreign Key references Medication)
   6. AdmissionDate
   7. DischargeDate
5. AdmissionType:
   1. AdmissionTypeID (Primary Key)
   2. AdmissionID (Foreign Key references Admission)
   3. AdmissionType (Emergency, Elective, Urgent)
6. Billing:
   1. BillingID (Primary Key)
   2. AdmissionID (Foreign Key references Admission)
   3. BillingAmount
7. TestResult:
   1. AdmissionID (Foreign Key references Admission)
   2. TestResults
8. Hospital:
   1. HospitalID (Primary Key)
   2. HospitalName
9. Room:
   1. RoomID (Primary Key)
   2. HospitalID (Foreign Key references Hospital)
   3. RoomNumber
10. Doctor:
    1. DoctorID (Primary Key)
    2. HospitalID (Foreign Key references Hospital)
    3. DoctorName
11. Medication:
    1. MedicationID (Primary Key)
    2. DiagnosisID (Foreign Key references Diagnosis)
    3. MedicineName
12. Diagnosis:
    1. DiagnosisID (Primary Key)
    2. MedicalCondition
13. DiagAdm (Associative Entity):
    1. AdmissionID (Foreign Key references Admission) – Primary Key part 1
    2. DiagnosisID (Foreign Key references Diagnosis) – Primary Key part 2

**PART – 3**

**Functionalities and their corresponding screenshots are as follows:**

1. **Insert a new patient:**

INSERT INTO BloodGroup (BloodType) VALUES ({blood\_type}) RETURNING BloodTypeID;

INSERT INTO Patient (Name, Age, Gender, BloodTypeID) VALUES ({name}, {age}, {gender}, {bloodtype\_id}) RETURNING PatientID;

INSERT INTO InsuranceProvider (PatientID, InsuranceProviderName) VALUES ({patient\_id}, {insurance\_provider});

INSERT INTO Hospital (HospitalName) VALUES ({hospital}) RETURNING HospitalID;

INSERT INTO Room (HospitalID, RoomNumber) VALUES ({hospital\_id}, {room\_number});

INSERT INTO Doctor (HospitalID, DoctorName) VALUES ({hospital\_id}, {doctor}) RETURNING DoctorID;

INSERT INTO Diagnosis (MedicalCondition) VALUES ({medical\_condition}) RETURNING DiagnosisID;

INSERT INTO Medication (DiagnosisID, MedicineName) VALUES ({diagnosis\_id}, {medication}) RETURNING MedicationID;

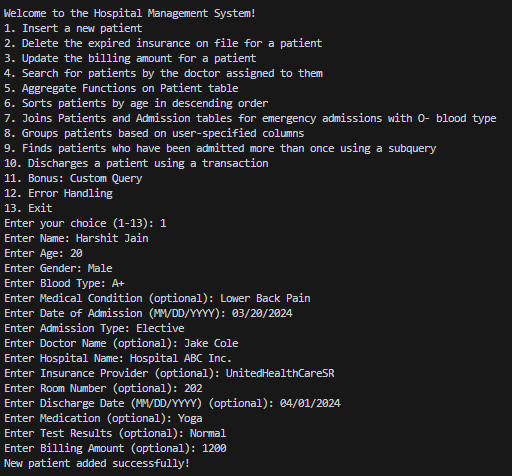
INSERT INTO Admission (PatientID, DoctorID, HospitalID, MedicationID, AdmissionDate, DischargeDate) VALUES ({patient\_id}, {doctor\_id}, {hospital\_id}, {medication\_id}, {admission\_date}, {discharge\_date}) RETURNING AdmissionID;

INSERT INTO AdmissionType (AdmissionID, AdmissionType) VALUES ({admission\_id}, {admission\_type});

INSERT INTO Billing (AdmissionID, BillingAmount) VALUES ({admission\_id}, {billing\_amount});

INSERT INTO TestResult (AdmissionID, TestResults) VALUES ({admission\_id}, {test\_results});

INSERT INTO DiagAdm (AdmissionID, DiagnosisID) VALUES ({admission\_id}, {diagnosis\_id});  
  
Screenshot below:



A black background with white text

Description automatically generated

1. **Delete the expired insurance on file for a patient**  
   DELETE FROM InsuranceProvider WHERE PatientID = {patient\_id};  
   A screen shot of a computer screen

   Description automatically generated  
   A screenshot of a computer program

   Description automatically generated  
   A black screen with white text

   Description automatically generated
2. **Update the billing amount for a patient**

UPDATE Billing SET BillingAmount = {new\_billing\_amount}

WHERE AdmissionID = {admission\_id}  
A screenshot of a computer program

Description automatically generated  
A screen shot of a computer

Description automatically generated

1. **Search for patients by the doctor assigned to them**  
   SELECT Admission.PatientID, Name, AdmissionID, Admission.DoctorID, DoctorName, AdmissionDate, DischargeDate

FROM Admission

JOIN Patient ON Admission.PatientID = Patient.PatientID

JOIN Doctor ON Admission.DoctorID = Doctor.DoctorID

WHERE Doctor.DoctorName LIKE '%{doctor\_name}%'  
A screenshot of a computer

Description automatically generated

1. **Aggregate Functions on Patient table**

SELECT COUNT(\*) FROM Patient;  
A screenshot of a computer

Description automatically generated

SELECT AVG(Age) FROM Patient;  
A screenshot of a computer

Description automatically generated

1. **Sorts patients by age in descending order**

SELECT \* FROM Patient ORDER BY Age DESC;  
A screenshot of a computer screen

Description automatically generated

1. **Joins Patients and Admission tables for emergency admissions with O- blood type**

SELECT P.Name, BG.BloodType, A.AdmissionDate, AT.AdmissionType

FROM Patient P

INNER JOIN Admission A ON P.PatientID = A.PatientID

INNER JOIN AdmissionType AT ON A.AdmissionID = AT.AdmissionID

INNER JOIN BloodGroup BG ON P.BloodTypeID = BG.BloodTypeID

WHERE AT.AdmissionType = 'Emergency' AND BG.BloodType = 'O-';

A screenshot of a computer

Description automatically generated

1. **Groups patients based on user-specified columns**

SELECT AVG(P.Age), P.{column} FROM Patient P GROUP BY {column};

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. **Finds patients with a length of stay exceeding a threshold using a subquery**subquery = SELECT PatientID, AdmissionDate, DischargeDate, (DischargeDate - AdmissionDate) AS LengthOfStay FROM Admission;

query = SELECT P.PatientID, P.Name, Stay.AdmissionDate, Stay.DischargeDate, Stay.LengthOfStay

FROM Patient P

INNER JOIN ({subquery}) AS Stay ON P.PatientID = Stay.PatientID

WHERE Stay.LengthOfStay > {threshold\_days};

A screenshot of a computer

Description automatically generated

1. **Discharges a patient using a transaction**

BEGIN TRANSACTION;

SELECT \* FROM Admission WHERE PatientID = {patient\_id} AND DischargeDate IS NULL;

UPDATE Admission SET DischargeDate = {discharge\_date} WHERE AdmissionID = {active\_admission[0]};  
COMMIT/ROLLBACK;  
  
a. DischargeDate is NULL. That means, it’s an active admission.  
A black screen with white text

Description automatically generated

b. Patient is discharged and DischargeDate has been updated to the current date.  
A screenshot of a computer program

Description automatically generated

c. DischargeDate updated (COMMIT) in the database.

A black screen with white text

Description automatically generated

d. Since the patient is successfully discharged and that patient is no longer having an active admission, the transaction will ROLLBACK if ran again.

A screenshot of a computer

Description automatically generated

1. **Bonus: Custom Query**

Here, the user can enter any additional query they would like.  
A screenshot of a computer

Description automatically generated

**PART – 4**

**CLI Interface**

A screenshot of a computer

Description automatically generated