Chapter 6 – Team 10

Sagar Devkate, Carlos Flores, Christina Hyman, Daulet Kapezov, Zallocco Filippo

CS 623: Database Management Systems

Dr. William Buttgieg

October 24, 2022

**Step 6.1 - Begin with the list of the tables that the entities and relationships from the E-R diagram mapped to naturally, from the sample project section at the end of chapter 4.**

policyType (company, policyType, medicalCoPay, labCoPay, pharmacyCoPay)

InsurancePolicy ( policyNo, *company*, *policyType,* startDate, endingDate)

1. Values are single-valued; records are unique; attributes are atomic => The table is in 1NF.
2. Table in 2NF
3. 1NF;  policyNo -> company, policytype, startDate, endingDate. And policytype -> medicalCoPay, labCoPay, pharmacyCoPay,  Therefore, the table is in 2NF if we separate it into two tables.
4. Table in 3NF
5. Table in BCNF

Patient (patientNo, name, address, phone, dateOfBirth, sex, *insuranceCo, policyNo*, relationshipToInsured)

1. The values are single-valued and records are unique. However, attributes are not atomic => Therefore we will transform the ‘address’ attribute to a composite key consisting of street, zipCode, city, and state. We are also breaking up the ‘name’ into ‘firstName’ and ‘lastName’.
2. 1NF, patientNo → firstName, lastName, street, zipCode, city, state,1 phone, dateOfBirth, sex, insuranceCo, policyNo, relationshipToInsured

Table is in 2NF since all of non-prime attributes are fully functionally dependent on the primary key patientNo

1. 2NF, The insuranceCo attribute is transitively dependent on patientNo since patientNo → policyNo and policyNo → insuranceCo based on the insurance policy table. Therefore we will remove the insuranceCo attribute to avoid the redundancy.
2. The table is in BCNF.

The new tables are below.

Patient1 (patientNo, firstName, lastName, street, *zipCode*, phone, dateOfBirth, sex*, policyNo*, relationshipToInsured);

Zip (zipCode, city, state);

Staff (staffNo, name, title, specialty, address, phone)

1. 1NF: Same as Patient, we will transform the ‘address’ attribute into Staff (staffNo, name, title, specialty, phone, address, city, state, zip
2. 2NF: The table is in 2NF as it is in 1NF and all of it’s non-primary key attributes are fully functionally dependent on its primary key

3. 3NF: New table zip created for staff member attributes are city and state. The table is in 3NF as it is in 2NF and there are no transitive dependencies.

4. The table is in BCNF

Staff (staffNo, firstName, lastName, title, specialty, street, zipCode, phone)

Availability (*staffNo*, availDate, startTime, endTime) 

1. Values are single-valued; records are unique; attributes are atomic => The table is in 1NF.
2. 1NF, staffNo, availDate → startTime, endTime
3. 2NF, There are no transitive dependencies, therefore the table is in 3NF.
4. The table is not in BCNF since the ‘availDate’ under AvailabilityDate is not a superkey.

Room (roomNo, *roomType*, condition)

1. Values are single-valued; records are unique; attributes are atomic. Therefore the table is in 1NF.
2. 1NF, roomNo → roomType, condition

Table is in 2NF since all of non-prime attributes are fully functionally dependent on the primary key roomNo

1. RoomNo -> roomType and roomType -> condition. We eliminate this transitive dependency separating the table.
2. 2NF, There are no transitive dependencies, therefore the table is in 3NF.
3. The table is in BCNF.

roomType(roomType, condition)

1. Visit (visitNo, *patientNo*, visitdate*,* visittime, duration, reason, visitType, visitCost, *staffNo, roomNo)*
2. 1NF: Visit (visitNo, patientNo, visitdate, visittime, duration, reason, visitType, visitCost, staffNo, roomNo)
3. 2NF:  The table is in 2NF as it’s already in 1NF and it’s non-primary key attributes are fully functionally dependent on the primary key.
4. Visit (visitNo, visitdate, visittime, duration, reason, visitType, visitCost) staffNo(patientNo, roomNo) visitType -> viditCost, so we separate it to Eliminate transitive dependency.
5. 3NF: The table is in 3NF as it is in 2NF and there are no transitive dependencies.
6. The table is in BCNF

Visit (visitNo, *patientNo*, visitdate*,* visittime, duration, reason,visitType, *staffNo, roomNo)*

visitType (visitType, visitCost)

Appointment (*patientNo*, apptdate, appttime, reason, *staffNo*, *visitNo*)

1. 1NF: Appointment (patientNo, apptdate, appttime, reason, staffno, visitNo)
2. 2NF: The table is in 2NF as it’s already in 1NF and it’s non-primary key attributes are fully functionally dependent on the primary key.
3. 3NF: The table is in 3NF as it is in 2NF and there are no transitive dependencies.
4. The table is in BCNF

Appointment *(patientNo,* apptdate, appttime, reason*, staffNo, visitNo)*

Referral (refNo, *visitNo*, refTo, reason)

1. Tuple cells hold single-valued inputs; records are unique; attributes are atomic. Hence, the table is in first normal form (1FN).
2. 1NF, refNo → refTo, reason

The Referral table is in Second Normal Form since all 1NF requirements are satisfied and the table’s non-prime attributes are dependent on the primary key, refNo.

1. 2NF, there are no transitive dependencies in Referral, so the table is in 3NF as well.
2. Since all the previous normalization requirements are satisfied and that Referral has only one unique superkey, refNo, the table is also in BCNF.

Bill (invoiceNo, billDate, totalAmount, dueDate, *patientNo,* amountPaid)

1. Tuple cells hold single-valued inputs; records are unique; attributes are atomic. Thus, the table is in first normal form (1FN).
2. 1NF, invoiceNo → billDate, totalAmount, dueDate, patientNo, amountPaid

The Bill table is in Second normal Form since all 1NF requirements are met and the table’s non-prime attributes are dependent on the primary key, invoiceNo.

1. I am going to delete the attribute amountPaid from the Bill table as it critical to the Payment table and not to the Bill table. Therefore, Bill is now 3NF having met all 2NF requirements.
2. Since all the previous normalization requirements are met and that Referral has only one unique superkey, refNo, the table is also in BCNF.

Bill (invoiceNo, billDate, totalAmount, dueDate, *patientNo*)

Charge (*invoiceNo*, serviceType, serviceDate, amountCharged)

1. Tuple cells hold single-valued inputs; records are unique; attributes are atomic. Thus, the table is in first normal form (1FN).

       2.   1NF, and there are three primary keys and one non-prime key in this relation.

invoiceNo, serviceType, serviceDate → amountCharge: we are going to define how

these dependencies of each primary key follows:

invoiceNo→serviceType

serviceType→serviceDate

serviceType→amountCharged

Thereby, we have the following tables:

Charge1 (invoiceNo, *serviceType*)

serviceType (serviceType, serviceDate, amountCharged)

3. 2NF, there are no transitive dependencies; consequently, the table is also in 3FN.

4.   The second functional dependency in Charge, serviceType → amountCharged is difficult to implement, because the serviceType can represent a missed appointment, a visit, a lab test, a prescription, or a procedure performed. Since a union is needed to represent these possibilities, we choose to ignore this dependency in the relational model, and allow amountCharged to remain in the Charge table. We assume the procedure for calculating bills will check the value of service type and be able to assign the correct charge. So, the table is also BCNF

Payment (*invoiceNo*, date, amountPaid, *insuranceCoPayer, patientPayer, insurancePolNoPayer)*

1. 1NF, values are single-valued, records are unique, and attributes are atomic
2. 1NF, and invoiceNo, date → amountPaid, insuranceCoPayer, patientPayer, insurancePolNoPayer
3. 2NF, insuranceCoPayer is unnecessary in Payment table because it is already a feature identifying patient in its table. patientPayer is also redundant because this attribute it is used anywherelese. Thus, it serves no purpose in this table as well. As a consequence, the table is now in 3NF.
4. The table is in 3NF, and has one superkey. Thereby, the Payment table is also in BCNF.

Payment (*invoiceNo*, date, amountPaid*, insurnacePolNoPayer)*

DiagnosisMenu (diagCode, diagName)

1. 1NF: DiagnosisMenu (daigCode, diagName)
2. 2NF: The table is in 2NF as it is in 1NF and its non-primary key attributes are fully functionally dependent on its primary key.
3. 3NF: The table is in 3NF as it is in 2NF and there are no transitive dependencies.
4. The table is in BCNF

ProcedureMenu (procCode, procName, cost)

1. 1NF: Procedure Menu (procCode, procName, cost)
2. 2NF: The table is in 2NF since it is already in 1NF and t’s non-primary key attributes are fully functionally dependant on the primary key.
3. 3NF: The table is in 3NF as it is in 2NF and there are no transitive dependencies.
4. The table is in BCNF

PrescriptionScript (scriptNo, *visitNo*, dateWritten, itemPrescribed, quantityPrescribed, directions, numberRefills) DAULET

1. Values are single-valued; records are unique; attributes are atomic => The table is in 1NF.
2. 1NF; itemPrescribed, quantityPrescribed and numberRefills are not functionally dependent on scriptNo. These attributes are dependent on specific Medication, so we add them to the PrescriptionMedication table.
3. 2NF, There is no transitive dependencies  => The table is in 3NF
4. The table is in BCNF.

PrescriptionScript (scriptNo, *visitNo*, dateWritten, directions)

PrescriptionMedication (RXNumber, *scriptNo*, drugDispensed, quantityPrescribed, quantityDispensed , dateDispensed, refillsRemaining, cost)

1. Values are single-valued; records are unique; attributes are atomic => The table is in 1NF.
2. All of the attributes are functional dependent on the primary key => The table is in 2NF
3. There is transitive dependency drugDispensed -> cost.
4. The table is BCNF.

PrescriptionMedication (RXNumber, *scriptNo*, *drugDispensed*, quantityPrescribed, quantityDispensed , dateDispensed, refillsRemaining)

Drug(drugDispensed*, cost*)

LabTest (testNo, *prescriptionNo*, testType, testDate, testTime, cost, result)

1. Values are single-values; records are unique; attributes are atomic=> The table is in 1NF.
2. All of the attributes are functional dependent on the primary key => The table is in 2NF.
3. There is a transitive dependency testType -> cost.
4. Table in BCNF

LabTest(testNo, testDate, testTime, result)

TestType(testType, cost)

ProcedurePerformed (*visitNo*, *procCode*, result)

1. Values are single-values; records are unique; attributes are atomic => Table is in 1NF
2. All attributes are single-valued; records are unique; attributes are atomic=> The table is in 1NF
3. All non-primary attributes are functionally dependent on the primary key, in 2NF.
4. No transitive dependencies; table in 3NF
5. Table in BCNF

Diagnosis (*visitNo, diagCode*, dateOnset, symptoms, severity, prognosis)

1. Values are single values; records are unique; attributes are atomic => The table is in 1NF.
2. All non-primary attributes are functionally dependent on the primary key.
3. No transitive dependencies: table is in 3NF
4. Table is in BCNF

**Step 6.2 – Review and update the data dictionary and list of assumptions (as needed).**

**6.3 For each table, write the table name and write out the name, data types, and size of all the data items.**

**6.4 – Design SQL statements to create all tables needed to implement the design. Then create the tables in the database.**

**6.5 – Design SQL statements to create indexes for foreign keys and for any other columns that will be used most often for queries. Then execute the SQL statements in the database.**

CREATE TABLE PolicyType (

company VARCHAR2(50) NOT NULL,

policytype                  CHAR(1) NOT NULL,

  medicalCoPay           NUMBER(9, 2),

labCoPay                   NUMBER(9, 2),

pharmacyCoPay        NUMBER(9, 2),

CONSTRAINT PolicyType\_company\_policyType\_pk PRIMARY KEY (company, policyType)

);

Text

Description automatically generated

CREATE TABLE InsurancePolicy (

policyNo NUMBER(9) NOT NULL PRIMARY KEY,

company         VARCHAR2(30),

policyType                CHAR(1),

  startDate                   DATE,

endDate                  DATE,

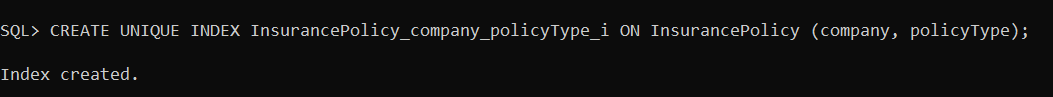
CONSTRAINT InsurancePolicy\_company\_policyType\_fk FOREIGN KEY (company, policyType) REFERENCES PolicyType (company, policyType) ON DELETE CASCADE

);

A computer screen capture

Description automatically generated with low confidence

CREATE UNIQUE INDEX InsurancePolicy\_company\_policyType\_i ON InsurancePolicy (company, policyType);



InsurancePolicy ( policyNo, *company*, insuredName, *policytype*, startDate, endingDate)

policyType(company, policytype, medicalCoPay, labCoPay, pharmacyCoPay)

CREATE TABLE Zip (

zipCode CHAR(5) NOT NULL PRIMARY KEY,

city VARCHAR2(20),

state CHAR(2)

);

Text

Description automatically generated

CREATE TABLE Patient (

patientNo   NUMBER(9) NOT NULL PRIMARY KEY,

firstName   VARCHAR2(30) NOT NULL,

lastName   VARCHAR2(30) NOT NULL,

street   VARCHAR2(20),

zipCode   CHAR(5),

phone   CHAR(10),

dateOfBirth   DATE,

sex   CHAR(1),

policyNo   NUMBER(9),

CONSTRAINT Patient\_policyNo\_fk FOREIGN KEY (policyNo) REFERENCES InsurancePolicy (policyNo) ON DELETE CASCADE,

CONSTRAINT Patient\_zipCode\_fk FOREIGN KEY (zipCode) REFERENCES Zip (zipCode) ON DELETE CASCADE

);

CREATE UNIQUE INDEX Patient\_policyNo\_i ON Patient(policyNo);

CREATE  UNIQUE INDEX Patient\_zipCode\_i ON Patient(zipCode);

Text

Description automatically generated

Text

Description automatically generated with medium confidence

CREATE TABLE Staff(

staffNo   NUMBER(9) NOT NULL PRIMARY KEY,

firstName VARCHAR2(30) NOT NULL,

lastName VARCHAR2(30) NOT NULL,

title   VARCHAR2(4) NOT NULL,

specialty   VARCHAR2(30),

Street VARCHAR2(25),

zipCode CHAR(5),

phone   CHAR(10),

CONSTRAINT Staff\_zipCode\_fk FOREIGN KEY (zipCode) REFERENCES Zip (zipCode) ON DELETE CASCADE

);

CREATE  UNIQUE INDEX Staff\_zipCode\_i ON Staff(zipCode);

Text

Description automatically generated

CREATE TABLE Availability(

staffNo               NUMBER(9) NOT NULL,

availDate   DATE NOT NULL,

startTime        VARCHAR2(8),

endTime       VARCHAR2(8),

CONSTRAINT Availability\_staffNo\_availDate\_pk PRIMARY KEY (staffNo, availDate),

CONSTRAINT Availability\_staffNo\_fk FOREIGN KEY (staffNo) REFERENCES Staff (staffNo)

);

Text

Description automatically generated

CREATE  UNIQUE INDEX Availability\_staffNo\_i ON Availability(staffNo);

Text

Description automatically generated

CREATE TABLE Room (

roomNo  NUMBER(3) NOT NULL PRIMARY KEY,

roomType CHAR(1),

condition VARCHAR2(50)

);

Text

Description automatically generated

CREATE TABLE visitType(

visitType VARCHAR(30) NOT NULL PRIMARY KEY,

visitCost NUMBER(10,2)

);

Text

Description automatically generated

CREATE TABLE Visit(

visitNo             NUMBER(9) NOT NULL,

patientNo                                   NUMBER(9) NOT NULL,

visitdate             DATE,

visittime                                      VARCHAR2(30),

duration                           NUMBER(2),

reason             VARCHAR2(100),

visitType                                     VARCHAR(30) NOT NULL,

staffNo             NUMBER(9) NOT NULL,

roomNo             NUMBER(3) NOT NULL,

CONSTRAINT Visit\_visitNo\_pk PRIMARY KEY (visitNo),

CONSTRAINT Visit\_patientNo\_fk FOREIGN KEY (patientNo) REFERENCES Patient(patientNo) ON DELETE CASCADE,

CONSTRAINT Visit\_roomNo\_fk FOREIGN KEY (roomNo) REFERENCES Room(roomNo) ON DELETE CASCADE,

CONSTRAINT Visit\_visitType\_fk FOREIGN KEY (visitType) REFERENCES visitType(visitType) ON DELETE CASCADE

);

CREATE UNIQUE INDEX Visit\_patientNo\_i ON Visit(patientNo);

CREATE INDEX Visit\_roomNo\_i ON Visit(roomNo);

CREATE INDEX Visit\_visitType\_i ON Visit(visitType);

Text

Description automatically generated

CREATE TABLE Appointment(

patientNo NUMBER(9) NOT NULL,

apptDate DATE NOT NULL,

apptTime VARCHAR(30) NOT NULL,

reason VARCHAR2(50),

staffNo NUMBER(9) NOT NULL,

visitNo NUMBER(9),

CONSTRAINT Appointment\_patientNo\_apptdate\_pk PRIMARY KEY (patientNo, apptdate),

CONSTRAINT Appointment\_patientNo\_fk FOREIGN KEY (patientNo) REFERENCES Patient(patientNo) ON DELETE CASCADE,

CONSTRAINT Appointment\_staffNo\_fk FOREIGN KEY (staffNo) REFERENCES Staff(staffNo)  ON DELETE CASCADE,

CONSTRAINT Appointment\_visitNo\_fk FOREIGN KEY (visitNo) REFERENCES Visit(visitNo) ON DELETE CASCADE

);

CREATE INDEX Appointment\_patientNo\_i ON Appointment(patientNo);

CREATE INDEX Appointment\_staffNo\_i ON Appointment(staffNo);

CREATE UNIQUE INDEX Appointment\_visitNo\_i ON Appointment(visitNo);

Text

Description automatically generated

CREATE TABLE Referral (

refNo NUMBER(6) NOT NULL PRIMARY KEY,

visitNo NUMBER(6) NOT NULL,

refTo VARCHAR2(30),

Reason VARCHAR2(50),

CONSTRAINT Referral\_visitNo\_fk FOREIGN KEY (visitNo) REFERENCES Visit(visitNo) ON DELETE CASCADE

);

CREATE UNIQUE INDEX Referral\_visitNo\_i ON Referral(visitNo);

Text

Description automatically generated

CREATE TABLE Bill (

invoiceNo NUMBER(9) NOT NULL PRIMARY KEY,

billDate DATE ,

totalAmount NUMBER(9, 2),

dueDate DATE,

patientNo NUMBER(9) NOT NULL,

CONSTRAINT Bill\_patientNo\_fk FOREIGN KEY (patientNo) REFERENCES Patient(patientNo) ON DELETE CASCADE

);

CREATE INDEX Bill\_patientNo\_i ON Bill(patientNo);

Text

Description automatically generated

CREATE TABLE Charge(

invoiceNo NUMBER(9) NOT NULL,

serviceType VARCHAR2(75) NOT NULL,

serviceDate DATE NOT NULL,

amountCharged NUMBER(3,2),

CONSTRAINT Charge\_invoice\_serviceType\_serviceDate\_pk PRIMARY KEY (invoiceNo, serviceType, serviceDate)

);

Text

Description automatically generated  
  
  
CREATE TABLE Payment (

invoiceNo                                  NUMBER(9) NOT NULL,  
datePaid                                    DATE NOT NULL,   
amountPaid                              NUMBER(9,2),  
policyNo                                    NUMBER(9),

CONSTRAINT Payment\_invoiceNo\_date\_pk PRIMARY KEY(invoiceNo, datePaid),

CONSTRAINT Payment\_invoiceNo\_fk FOREIGN KEY(invoiceNo) REFERENCES Bill(invoiceNo),

CONSTRAINT Payment\_insCo\_polNo\_fk FOREIGN KEY(policyNo) REFERENCES InsurancePolicy(policyNo)

);

Text

Description automatically generated

CREATE UNIQUE INDEX Payment\_invoiceNo\_i ON Payment(invoiceNo);

CREATE UNIQUE INDEX Payment\_policyNo\_i ON Payment(policyNo);

Text

Description automatically generated

CREATE TABLE Drug(

drugDispensed VARCHAR2(30) Primary Key NOT NULL,

cost NUMBER(9,2)

);

Text

Description automatically generated with medium confidence

CREATE TABLE TestType(

testType CHAR(1) NOT NULL PRIMARY KEY,

Cost NUMBER(10,2)

);

Text

Description automatically generated

CREATE TABLE DiagnosisMenu (

diagCode CHAR(6),

diagName VARCHAR2(50),

CONSTRAINT DiagnosisMenu\_diagCode\_pk PRIMARY KEY (diagCode));

Text

Description automatically generated

CREATE TABLE ProcedureMenu (

procCode CHAR(6) NOT NULL,

procName VARCHAR2(50),

cost NUMBER(10,2),

CONSTRAINT ProcedureMenu\_procCode\_pk PRIMARY KEY (procCode));

Text

Description automatically generated

CREATE TABLE PrescriptionScript (

scriptNo NUMBER(9),

visitNo NUMBER(9),

dateWritten DATE,

Directions VARCHAR(50),

CONSTRAINT PrescriptionScript\_scriptNo\_pk PRIMARY KEY (scriptNo),

CONSTRAINT PrescriptionScript\_visittNo\_fk FOREIGN KEY (visitNo) REFERENCES Visit(visitNo)

);

CREATE INDEX PrescriptionScript\_visitNo\_i ON PrescriptionScript(visitNo);

Text

Description automatically generated

Text

Description automatically generated with medium confidence

CREATE TABLE PrescriptionMedication (

RXNumber   NUMBER(9) NOT NULL PRIMARY KEY,

scriptNo   NUMBER(9),

drugDispensed   VARCHAR2(30),

dateDnispensed   DATE,

           quantityPrescribed          NUMBER(3),

quantityDispensed   NUMBER(3),

refillsRemaining   NUMBER(2),

CONSTRAINT PrescriptionMedication\_scriptNo\_fk FOREIGN KEY (scriptNo) REFERENCES PrescriptionScript(scriptNo)

);

CREATE UNIQUE INDEX PrescriptionMedication\_scriptNo\_i ON PrescriptionMedication(scriptNo);

Text

Description automatically generated

CREATE TABLE LabTest (

testNo   NUMBER(9) NOT NULL,

RXNumber NUMBER(9) NOT NULL,

testType CHAR(1),

testDate DATE,

testTime VARCHAR2(30),

cost NUMBER(9,2),

result VARCHAR2(30),

CONSTRAINT LabTest\_testNo\_pk PRIMARY KEY (testNo),

CONSTRAINT LabTest\_RXNumber\_fk FOREIGN KEY (RXNumber) REFERENCES PrescriptionMedication(RXNumber) ON DELETE CASCADE,

CONSTRAINT LabTest\_TestType\_fk FOREIGN KEY (TestType) REFERENCES TestType(TestType) ON DELETE CASCADE

);

CREATE UNIQUE INDEX LabTest\_RXNumber\_i ON LabTest (RXNumber);

CREATE UNIQUE INDEX LabTest\_testType\_i ON LabTest (testType);

Text

Description automatically generated with medium confidence

CREATE TABLE ProcedurePerformed (

visitNo   NUMBER(6)   NOT NULL,

procCode   CHAR(6)   NOT NULL,

result   VARCHAR2(50),

CONSTRAINT ProcedurePerformed\_visitNo\_procCode\_pk PRIMARY KEY (visitNo, procCode),

CONSTRAINT ProcedurePerformed\_visitNo\_fk FOREIGN KEY(visitNo) REFERENCES Visit(visitNo),

CONSTRAINT ProcedurePerformed\_procCode\_fk FOREIGN KEY(procCode) REFERENCES ProcedureMenu(procCode));

CREATE INDEX ProcedurePerformed\_procCode\_i ON ProcedurePerformed(procCode);

Text

Description automatically generated

CREATE TABLE Diagnosis (

visitNo NUMBER(6) NOT NULL,

diagCode CHAR(6) NOT NULL,

dateOnset DATE,

symptoms VARCHAR2(50),

Severity NUMBER(2),

Prognosis VARCHAR(50),

CONSTRAINT Diagnosis\_visitNo\_diagCode\_pk PRIMARY KEY(visitNo, diagCode),

CONSTRAINT Diagnosis\_visitNo\_fk FOREIGN KEY (visitNo) REFERENCES Visit (visitNo),

CONSTRAINT Diagnosis\_diagCode\_fk FOREIGN KEY (diagCode) REFERENCES DiagnosisMenu (diagCode));

CREATE UNIQUE INDEX Diagnosis\_visitNo\_i ON Diagnosis (visitNo);

CREATE UNIQUE INDEX Diagnosis\_diagCode\_i ON Diagnosis (diagCode);

Text

Description automatically generated

**Step 6.6 - Design SQL statements to insert at least five records in each table, preserving all constraints.** **Then insert the records into the tables.**

INSERT INTO PolicyType VALUES(‘Allstate’, ‘P’, 2340.90, 970.00, 500.00);

INSERT INTO PolicyType VALUES(‘Geico’, ‘P’, 3400.90, 790.00, 90.00);

INSERT INTO PolicyType VALUES(‘AIG’, ‘P’, 8760.00,  670.00, 100.00);

INSERT INTO PolicyType VALUES(‘NewYorkState’, ‘S’, 4012.09, 507.00, 225.00);

INSERT INTO PolicyType VALUES(‘StateOfMassachusetts’, ‘S’, 13402.91, 969.00, 130.00);

Text

Description automatically generated

INSERT INTO InsurancePolicy VALUES(500000123, ‘Allstate’, ‘P’, ‘10-OCT-2022’, ‘09-OCT-2023’);

INSERT INTO InsurancePolicy VALUES(600000124, ‘Geico’, ‘P’, ‘11-NOV-2022’, ‘10-NOV-2023’);

INSERT INTO InsurancePolicy VALUES(700000125, ‘AIG’, ‘P’, ‘12-JAN-2022’, ‘11-JAN-2023’);

INSERT INTO InsurancePolicy VALUES(800000126, ‘NewYorkState’, ‘S’, ‘07-JUN-2022’, ‘06-JUN-2023’);

INSERT INTO InsurancePolicy VALUES(900000127, ‘StateOfMassachusetts’, ‘S’, ‘05-MAY-2022’, ‘04-MAY-2023’);

Text

Description automatically generated

INSERT INTO Zip VALUES(10010, ‘NewYorkCity’, ‘NY’);

INSERT INTO Zip VALUES(10312, ‘NewYorkCity’, ‘NY’);

INSERT INTO Zip VALUES(10401, ‘NewYorkCity’, ‘NY’);

INSERT INTO Zip VALUES(10532, ‘NewYorkCity’, ‘NY’);

INSERT INTO Zip VALUES(10623, ‘NewYorkCity’, ‘NY’);

Text

Description automatically generated

INSERT INTO Patient VALUES(789001, ‘Mitch’, ‘Chin’, ‘Bradford Road’, 10312, 678456098, ‘13-MAR-1993’, ‘M’, 600000124);

INSERT INTO Patient VALUES(789002, ‘Rossella’, ‘Ahmad’, ‘Ross Stress’, 10010, 3749754273, ‘12-FEB-1992’, ‘F’, 500000123);

INSERT INTO Patient VALUES(789003, ‘Karla’, ‘Rapezvkov’, ‘JFK Avenue’, 10401, 234546586, ‘14-MAR-1994’, ‘F’, 700000125);

INSERT INTO Patient VALUES(789004, ‘Joseph’, ‘Shoellerman’, ‘Vincelly Street’, 10532, 321345231, ‘15-APR-1995’, ‘M’, 800000126);

INSERT INTO Patient VALUES(789005, ‘Jesus’, ‘Muñoz’, ‘Wall Street’, 10623, 687904321, ‘16-MAY-1997’, ‘M’, 900000127);

Text

Description automatically generated

INSERT INTO Staff Values(100679, ‘Marla’, ‘Smith’, ‘CNM’, ‘Midwife’, ‘01 Broadway’, 10010, 890765342);

INSERT INTO Staff Values(100680, ‘Michael’, ‘Ross’, ‘Mr.’, ‘Receptionist’, ‘215 8th Ave’, 10312, 654987123);

INSERT INTO Staff Values(100681, ‘Abdhul’, ‘Sagar’, ‘MD’, ‘Surgeon’, ‘12 Allen St’, 10401, 687321456);

INSERT INTO Staff Values(100682, ‘Homer’, ‘Hardwood’, ‘MD’, ‘Physician’,’17 William Street’ ,10532, 098567432);

INSERT INTO Staff Values(100683, ‘Donna’, ‘Paulsen’, ‘RN’, ‘Registered Nurse’,  ‘1 Pace Plaza’ ,10623, 798567345);

Text

Description automatically generated with medium confidence

INSERT INTO Availability VALUES(100679, ‘16-NOV-2022’, ‘10:00’, ‘15:00’);

INSERT INTO Availability VALUES(100680, ‘15-NOV-2022’, ‘09:00’, ‘17:00’);

INSERT INTO Availability VALUES(100681, ‘20-NOV-2022’, ‘12:00’, ‘18:00’);

INSERT INTO Availability VALUES(100682, ‘18-NOV-2022’, ‘15:00’, ‘18:00’);

INSERT INTO Availability VALUES(100683, ‘17-NOV-2022’, ‘09:00’, ‘16:00’);

Text

Description automatically generated

INSERT INTO Room VALUES(101, ‘R’ ,  ‘Acceptable’);  
INSERT INTO Room VALUES(102, ‘R’, ‘Not enough beds’);

INSERT INTO Room VALUES(103, ‘O’, ‘Acceptable’);  
INSERT INTO Room VALUES(104,  ‘D’, ‘Not enough beds’);  
INSERT INTO Room VALUES(105,  ‘R’, ‘Acceptable’);

Text

Description automatically generated

INSERT INTO VisitType VALUES(‘New Patient’, 300.00);

INSERT INTO VisitType VALUES(‘Follow-Up’, 250.00);

INSERT INTO VisitType VALUES(‘Walk-in/Same day’, 300.00);

INSERT INTO VisitType VALUES(‘Urgent’, 350.00);

INSERT INTO VisitType VALUES(‘Bloodwork/Vaccination’, 175.00);

Text

Description automatically generated

INSERT INTO Visit VALUES(1, 789001, ‘07-OCT-2022’, ‘13:15’, 15, ‘Flu symptoms’, ‘Walk-in/Same day’, 100682, 101);

INSERT INTO Visit VALUES(2, 789002, ‘21-SEP-2022’, ‘15:00’, 60, ‘Childbirth’, ‘Urgent’, 100679, 103);

INSERT INTO Visit VALUES(3, 789003, ‘28-SEP-2022’, ‘12:30’, 30, ‘Low energy levels’, ‘Bloodwork/Vaccination’, 100683, 103);

INSERT INTO Visit VALUES(4, 789004, ‘03-OCT-2022’, ‘16:45’, 15, ‘Injury to ankle’, ‘Walk-in/Same day’, 100682, 105);

INSERT INTO Visit VALUES(5, 789005, ‘07-OCT-2022’, ‘09:45’, 30, ‘Chest pain’, ‘New Patient’, 100682, 105);

A picture containing background pattern

Description automatically generated

INSERT INTO Appointment VALUES(789001, ‘07-OCT-2022’, ‘13:15’, ‘Flu Shot’, 100682, 1);

INSERT INTO Appointment VALUES(789002, ‘21-SEP-2022’, ‘15:00’, ‘Childbirth’, 100679, 2);

INSERT INTO Appointment VALUES(789003, ‘28-SEP-2022’, ‘12:30’, ‘Anemia’, 100683, 3);

INSERT INTO Appointment VALUES(789004, ‘03-OCT-2022’, ‘16:45’, ‘Ankle Pain’, 100682, 4);

INSERT INTO Appointment VALUES(789005, ‘07-OCT-2022’, ‘09:45’, ‘Chest Discomfort’, 100682, 5);

Text

Description automatically generated

INSERT INTO Referral VALUES(135791, 1 , ‘Marla Smith’, ‘Flu symptoms’);

INSERT INTO Referral VALUES(135792, 2, ‘Micheal Ross’, ‘Childbirth’);

INSERT INTO Referral VALUES(135793, 3, ‘Sagar Abdhul’, ‘Low energy levels’);

INSERT INTO Referral VALUES(135794, 4, ‘Homer Hardwood’, ‘Injury to ankle’);

INSERT INTO Referral VALUES(135795, 5, ‘Paulsen Donna’, ‘Chest Pain’);

Text

Description automatically generated

INSERT INTO Bill VALUES(111222333, ‘07-OCT-2022’, 250.00, ‘07-NOV-2022’, 789001);

INSERT INTO Bill VALUES(234567433, ‘21-SEP-2022’, 300.00, ‘21-OCT-2022’, 789002 );

INSERT INTO Bill VALUES(322546677, ‘28-SEP-2022’, 350.00, ‘28-OCT-2022’, 789003);

INSERT INTO Bill VALUES(789532345, ‘03-OCT-2022’, 175.00, ‘03-NOV-2022’, 789004);

INSERT INTO Bill VALUES(323412387, ‘07-OCT-2022’, 300.00, ‘07-NOV-2022’, 789005);

Text

Description automatically generated

INSERT INTO Charge VALUES(111222333, ‘Routine’, ‘07-OCT-2022’, 69.99 );

INSERT INTO Charge VALUES(234567433, ‘Operation’, ‘21-SEP-2022’, 59.99)

INSERT INTO Charge VALUES(322546677, ‘Routine’, ‘28-SEP-2022’, 89.99);

INSERT INTO Charge VALUES(789532345, ‘Operation’, ‘03-OCT-2022’, 99.99);

INSERT INTO Charge VALUES(323412387, ‘Testing’, ‘07-OCT-2022’, 69.99);

Text

Description automatically generated

INSERT INTO Payment VALUES(111222333, ‘07-OCT-2022’, 69.99, 500000123);

INSERT INTO Payment VALUES(234567433, ‘15-OCT-2022’, 59.99, 600000124);

INSERT INTO Payment VALUES(322546677, ‘31-OCT-2022’, 89.99, 700000125);

INSERT INTO Payment VALUES(789532345, ‘21-OCT-2022’, 99.99, 800000126);

INSERT INTO Payment VALUES(323412387, ‘05-OCT-2022’, 69.99, 900000127);

Text

Description automatically generated

INSERT INTO Drug VALUES(‘Stanozolol’, 99.99);

INSERT INTO Drug VALUES(‘Methandienone’, 119.99);

INSERT INTO Drug VALUES(‘Trenbolone Acetate’, 149.99);

INSERT INTO Drug VALUES(‘Oxandrolone’, 99.99);

INSERT INTO Drug VALUES(‘Fluoxymesterone’, 129.99);

Text

Description automatically generated

INSERT INTO TestType VALUES(‘A’, 29.99);

INSERT INTO TestType VALUES(‘B’, 27.99);

INSERT INTO TestType VALUES(‘C’, 25.99);

INSERT INTO TestType VALUES(‘D’, 99.99);

INSERT INTO TestType VALUES(‘E’, 19/99);

Text

Description automatically generated

INSERT INTO DiagnosisMenu VALUES(‘A0112B’, ‘Septicemia’);

INSERT INTO DiagnosisMenu VALUES(‘A152B1’, ‘Depressive disorders’);

INSERT INTO DiagnosisMenu VALUES(’A15253’, ‘Schizophrenia spectrum’);

INSERT INTO DiagnosisMenu VALUES(‘A5601F’, ‘Diabetes mellitus with complication’);

INSERT INTO DiagnosisMenu VALUES(‘A253G1’, ‘Alcohol-related disorders’);

Text

Description automatically generated

INSERT INTO ProcedureMenu VALUES(‘29540’, ‘Antibiotics’, 129.99);

INSERT INTO ProcedureMenu VALUES(‘59400’, ‘Depression Therapy’, 130.24);

INSERT INTO ProcedureMenu VALUES(‘99212’, ‘Individual therapy. Psychotherapy.’, 249.99);

INSERT INTO ProcedureMenu VALUES(‘90688’, ‘Medication for diabet’, 69.99);

INSERT INTO ProcedureMenu VALUES(‘99203’, ‘Behavioral treatment’, 349.99);

Text

Description automatically generated

INSERT INTO PrescriptionScript VALUES(1, 1, ‘07-OCT-2022’, ‘Take two before bed’);  
INSERT INTO PrescriptionScript VALUES(2, 2, ‘27-NOV-2022’, ‘Take once after food’ );

INSERT INTO PrescriptionScript VALUES(3, 3, ‘25-SEP-2022’, ‘Twice a day’);  
INSERT INTO PrescriptionScript VALUES(4, 4, ‘05-OCT-2022’, ‘Once a week’);  
INSERT INTO PrescriptionScript VALUES(5, 5, ‘09-OCT-2022’ , ‘Take four every hour’);

Text

Description automatically generated

INSERT INTO PrescriptionMedication VALUES(1, 1, ‘Stanozolol’, ‘08-OCT-2022’, 2, 4, 2);

INSERT INTO PrescriptionMedication VALUES(2, 2, ‘Methandienone’, ‘28-NOV-2022’, 1, 3,2);

INSERT INTO PrescriptionMedication VALUES(3, 3, ‘Trenbolone Acetate’, ‘26-SEP-2022’, 1, 4, 3);

INSERT INTO PrescriptionMedication VALUES(4, 4, ‘Oxandrolone’, ‘06-OCT-2022’, 3, 4, 1);

INSERT INTO PrescriptionMedication VALUES(5, 5, ‘Fluoxymesterone’, ‘10-OCT-2022’, 2, 6, 4);

Text

Description automatically generated

INSERT INTO LabTest VALUES(5, 1, ‘A’, ‘11-SEP-2022’, ‘15:00’, 18.99, ‘Positive’);

INSERT INTO LabTest VALUES(4, 2, ‘B’, ‘25-OCT-2022’, ‘14:30’, 28.99, ‘Positive’);

INSERT INTO LabTest VALUES(3, 3, ‘C’, ‘2-OCT-2022’, ‘12:00’, 18.99, ‘Positive’);

INSERT INTO LabTest VALUES(2, 4, ‘D’, ‘13-SEP-2022’, ‘13:00’, 25.99, ‘Negative’);

INSERT INTO LabTest VALUES(1, 5, ‘E’, ‘17-SEP-2022’, ‘11:30’, 30.99, ‘Positive’);

Text

Description automatically generated

INSERT INTO ProcedurePerformed VALUES(1,‘29540’,’Positive’);

INSERT INTO ProcedurePerformed VALUES(2,‘59400’, ‘Positive’);

INSERT INTO ProcedurePerformed VALUES(3,‘99212’, ‘Positive’);

INSERT INTO ProcedurePerformed VALUES(4,‘90688’, ‘Negative’);

INSERT INTO ProcedurePerformed VALUES(5,‘99203’, ‘Positive’);

Text

Description automatically generated

INSERT INTO Diagnosis VALUES(1, ‘A0112B’, ‘07-OCT-2022’, ‘Nauseated’, 3, ‘Dead’);  
 INSERT INTO Diagnosis VALUES(2, ‘A152B1’, ‘21-SEP-2022’, ‘Dizzy’, 4, ‘Under observation’);  
 INSERT INTO Diagnosis VALUES(3, ’A15253’, ‘28-SEP-2022’, ‘chronic leg pain’, 5, ‘Under Dead’);   
INSERT INTO Diagnosis VALUES(4, ‘A5601F’, ‘03-OCT-2022’, ‘losing hearing ’, 4, ‘Recovery’);

 INSERT INTO Diagnosis VALUES(5, ‘A253G1’, ‘07-OCT-2022’, ‘paresthesia’, 2, ‘Under observation’);

Text

Description automatically generated

Step **6.7 - Design SQL statements that will process five non-routine requests for information from the database. Then execute the SQL statements in the database.**

This statement shows the number of drugs required to every patient.

SELECT  p.firstName, p.lastName, COUNT(RXNumber) as Drugs

FROM PrescriptionMedication m, PrescriptionScript s, Visit v, Patient p

WHERE v.patientNo = p.patientNo and s.visitNo = v.visitNo and s.scriptNo = m.scriptNo

GROUP BY  p.firstName, p.lastName;

Text

Description automatically generated

This statement will be used to find the quantity of drugs that are being dispensed by type.

SELECT m.drugDispensed, SUM(m.quantityDispensed)

FROM PrescriptionMedication m

GROUP BY drugDispensed

ORDER BY drugDispensed desc;

Text

Description automatically generated

This statement will be used to show the different policy type payouts per insurance policy

SELECT  i.company ,p.medicalCoPay, p.labCoPay, p.pharmacyCoPay

FROM PolicyType p, InsurancePolicy i

WHERE p.policytype=i.policytype ORDER BY p.company;

Text

Description automatically generated

This statement will select patients who had visits on October 7th 2022 and the reason for the visit.

SELECT  p.firstName, p.lastName, v.visitdate, v.reason

FROM Patient p,

Visit v

WHERE  p.patientNo = v.patientNo and v.visitDate = ‘07-OCT-2022’

ORDER BY p.lastName;

Text

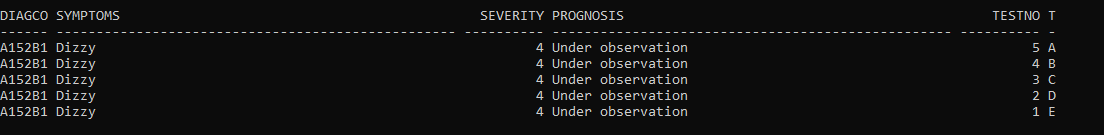
Description automatically generated

This statement will be used to show the diagnosis per Labtest

SELECT d.diagCode, d.symptoms, d.severity, d.prognosis, l.testNo, l.testType

FROM Diagnosis d, LabTest l

WHERE symptoms = ‘Dizzy’



**Step 6.8 - Design one trigger for your project. Then create the trigger in the database.**

Create a trigger that will update an attribute on one table whenever a row is updated on another table. Show your work by providing screenshots of creating the trigger in the database.

Note: This trigger should be different from the one you created in Chapter 5.

**Step 6.9 - Design and execute SQL statements to demonstrate that the trigger is working as expected.** To demonstrate that the trigger is working as expected, provide a screenshot of the data before and after the trigger is executed.

CREATE OR REPLACE TRIGGER refilRemainingCount

BEFORE INSERT OR UPDATE ON PrescriptionMedication

REFERENCING NEW AS NEW OLD AS OLD

FOR EACH ROW

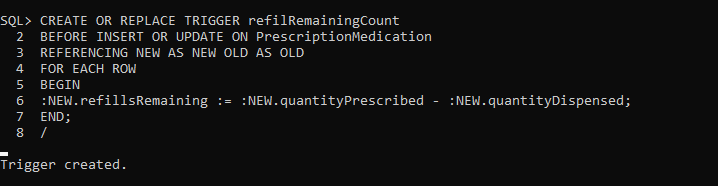
BEGIN

:NEW.refillsRemaining := :NEW.quantityPrescribed - :NEW.quantityDispensed;

END;

/

Trigger created.



We insert a new value to check whether the trigger works or not. As we can see here, the value of refillsRemaining changed using quantityPrescribed and quantityDispensed to calculate it.

