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

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

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

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

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

InsurancePolicy (company, policyNo, insuredName, policytype, medicalCoPay, labCoPay, pharmacyCoPay, startDate, endingDate)

DiagnosisMenu (diagCode, diagName)

ProcedureMenu (procCode, procName, cost)

Room (roomNo, roomType, condition)

PrescriptionScript (scriptNo, *visitNo*, dateWritten, itemPrescribed, quanityPrescribed, directions, numberRefills)

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

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

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

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

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

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

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

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

**The Explanation.**

The relationship sets are: Have, Covers, Consists, Lists, Makes, Pays, Participates, Creates, Schedules, Executed, Requires, Involves, Provides, Produces, Results, Determines, Recommends, States, Specifies, Follows, Gives.

**Have**: The Have relationship is a one-to-many relationship between Insurance Policy and Patient, and can be represented by placing the key of Insurance Policy in Patient. We place *insuranceCo* and *policyNo* in the Patient table as the foreign keys.

**Covers**: The Covers relationship is a one-to-many relationship between Insurance Policy and Payment. So we add *insurancePolicycompany* and *insurancePolicyPolicyNo* to the Payment table as the foreign key.

**Consists**: The Consists relationship is a one-to-many relationship between Bill and Payment. The Consists relationship has already been represented by placing the primary key of Bill in Payment. So we add *invoiceNo* to the Payment table as the foreign key, although it is part of the primary key.

**Lists**: The one-to-many Lists relationship can be represented by placing the primary key of Bill in the Charge table, so we add *invoiceNo* to the Charge table as a foreign key, although it is part of the primary key.

**Makes**: The Makes relationship is a one-to-many relationship between the Patient and Appointment. So we add *patientPayer* to the Payment table as a foreign key.

**Pay**: Pay is a one-to-many relationship between the Patient and Bill. So we can place the primary key of Patient as the foreign key to the Bill table, which is *patientNo.*

**Participates**: This is a one-to-many relationship between Patient and Visit. Thereby, *patientNo* is a foreign key in the Visit table.

**Makes**: The Creates relationship is a one-to-many relationship between Patient and Appointment. The Creates relationship has already been represented by placing the primary key of Patient in Appointment. So we add *patientNo* to the Appointment table as the foreign key, although it is part of the primary key.

**Schedules**: Schedules is a one-to-one relationship between the Appointment and Visit. Therefore we will designate the primary key of Visits as the foreign key for the Appointment table, which is *visitNo*.

**Execute**: Execute(d) is a one-to-many relationship between Room and Visit. As a consequence, *roomNo* will be the foreign key in the Visit table.

**Requires**: Requires is a one-to-many relationship between Appointment and Staff. Hence, *staffNo* will be a foreign key in the Appointment table.

**Involves**: Involves is a one-to-many relationship between Staff and Visit. Thus, *staffNo* will be the foreign key in the Visit table.

**Provides**: Provides is a one-to-many relationship between Staff and Availability. Furthermore, availability is a dependent entity, which stems from the entity staff, the entity-owner of availability. Therefore, staffNo is included in the Availability table as its candidate key.

**Produces**: The Produces relationship is a one-to-many relationship between Visit and PrescriptionScript. Therefore we need to place *visitNo* under the PrescriptionScript table as the foreign key.

**Results**: The Results relationship is a one-to-many relationship between Visit and ProcedurePerformed. This relationship has already been represented by placing the primary key of Visit in ProcedurePerformed. So we add *visitNo* to the ProcedurePerformed table as the foreign key, although it is part of the primary key.

**Determines**: The Determines relationship is a one-to-many relationship between Visit and Diagnosis. This relationship has already been represented by placing the primary key of Visit in Diagnosis. This is shown by placing *visitNo* under the Diagnosis table as the foreign key, although it is part of the primary key.

**Recommends**: The Recommends relationship is a one-to-many relationship between Visit and Referral. So we add *visitNo* to the Referral table as the foreign key, although it is part of the primary key.

**States**: The states relationship is one-to-many. We represent this relationship by placing a foregin entity in the table of Prescription Script representing many Lab Test. The foreign key will be *testNo*

**Specifies**: The Specifies relationship is one-to-many. This is represented using a foreign key of *RXNumber* within the prescription script table.

**Follows**: The follows relationship is many-to-one. We represent this relationship by placing the “one” entity Procedures Menu in the table for “many” side, this will make *procCode* the forgein key.

**Gives**: The Gives relationship is many-to-one. We represent it by placing a “one,” Diagnosis Menu in the table for the “many” side, *diagCode* will be the foregin key.