Part 3: Smilow’s Database SQL Implementation

Team 2: Anushka Nath, Sam Ziessler, Karl Chavez, Christina Duong

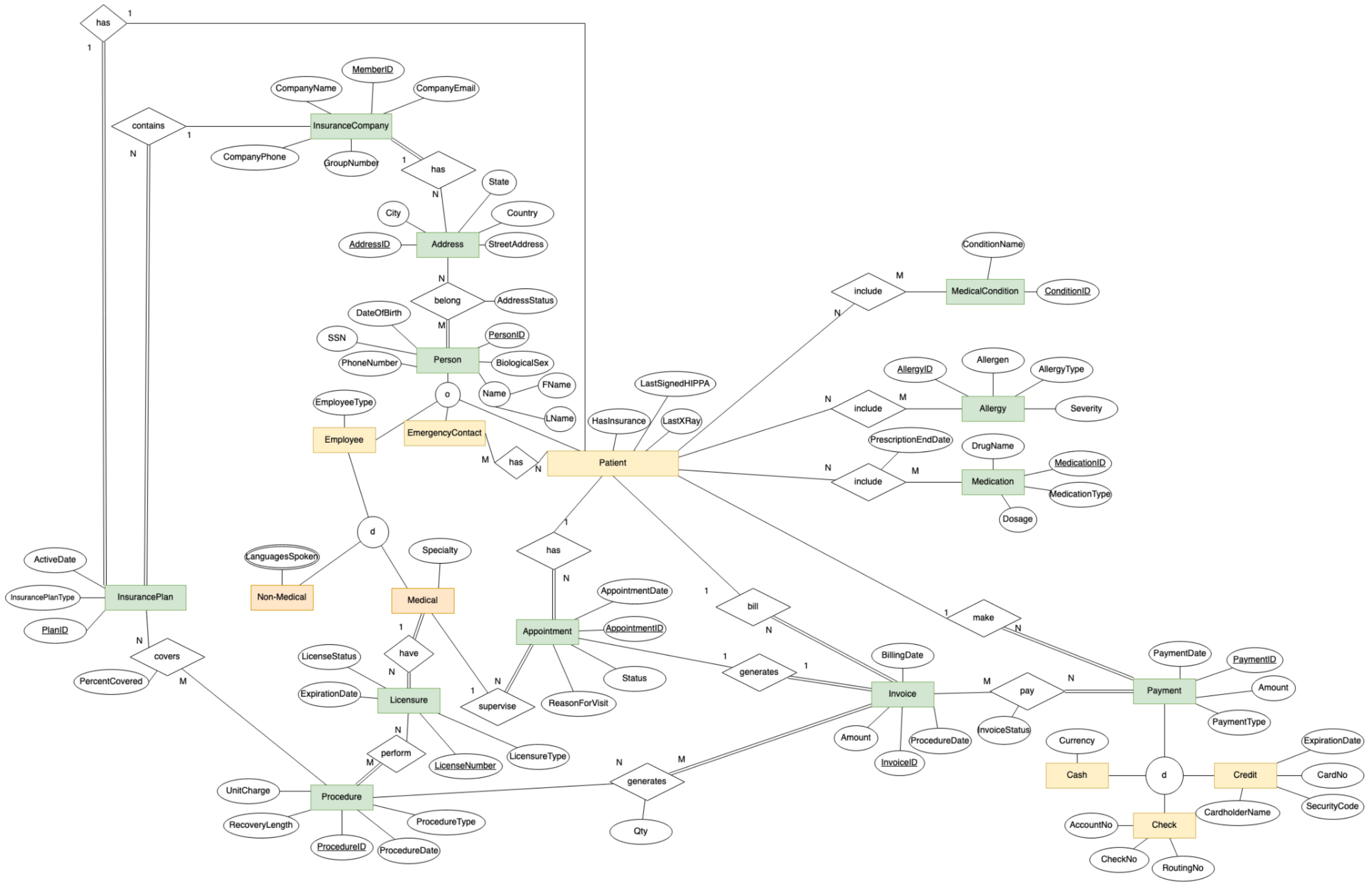
Intr Database Sys (7194)

March 2023

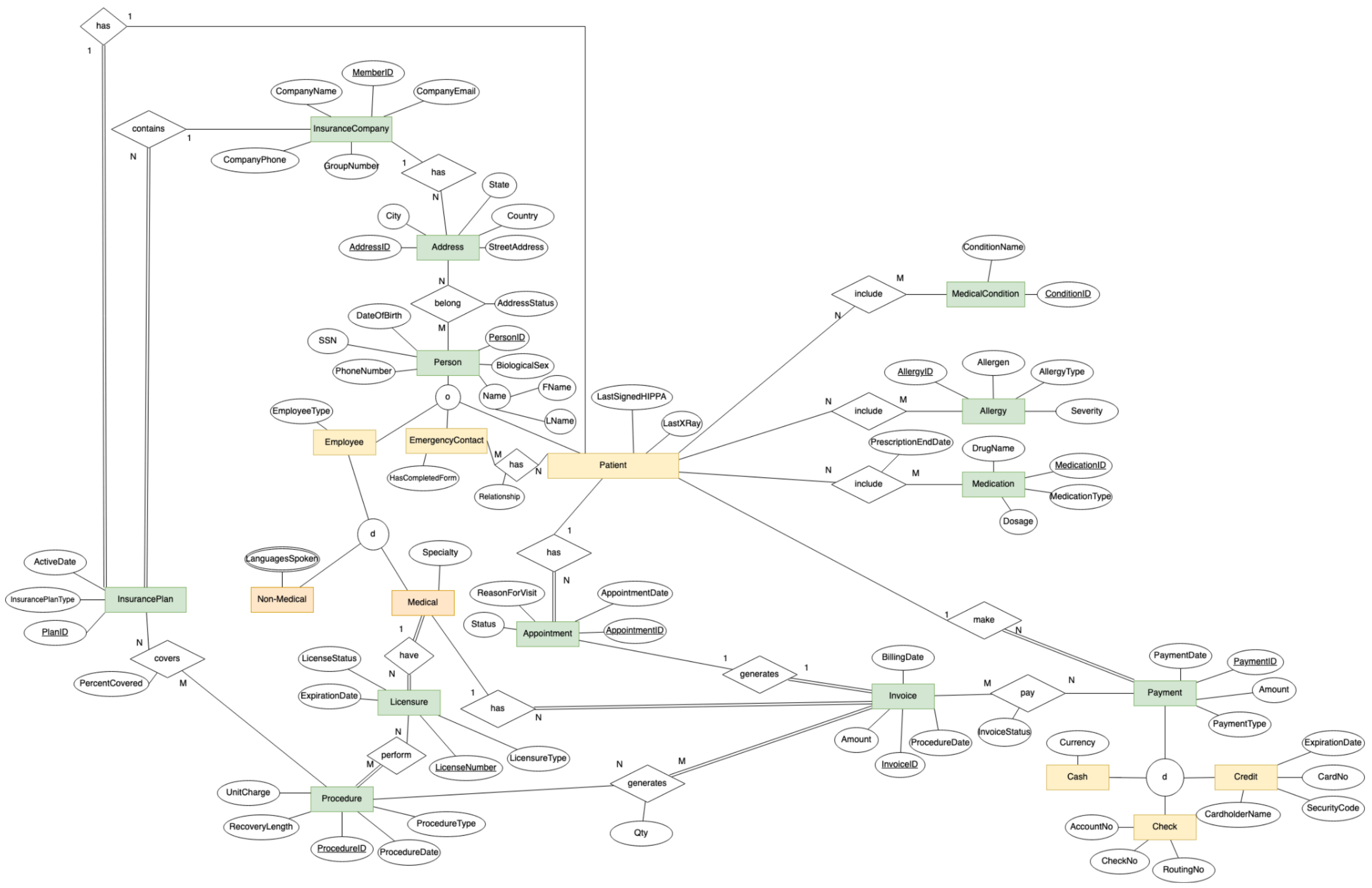
Part 1: ERD Updates

1. We removed the “HasInsurance” attribute from “Patient”.
2. We removed the relation between “Patient” and “Invoice”.
3. We removed the relation between “Medical” and “Appointment”.
4. We added the relation between “Medical” and “Invoice”.
5. We removed the total cardinality between “InsuranceCompany” and “Address”.
6. We removed the total cardinality between “Address” and “Person”.
7. We removed the total cardinality between “Invoice” and “Payment”.

Original ERD:



New ERD:


has
MemberID
CompanyName
CompanyEmail
contains
InsuranceCompany
CompanyPhone
GroupNumber
has
State
City
Country
AddressID
Address
StreetAddress
ConditionName
include
MedicalCondition
ConditionID
belong
AddressStatus
DateOfBirth
PersonID
SSN
Person
BiologicalSex
PhoneNumber
FName
Allergen
AllergyID
AllergyType
o
Name
LastSignedHIPPA
include
Allergy
Severity
EmployeeType
LName
LastXRay
PrescriptionEndDate
Employee
EmergencyContact
DrugName
has
MedicationID
Patient
HasCompletedForm
include
Medication
Relationship
MedicationType
Dosage
d
ActiveDate
LanguagesSpoken
Specialty
has
InsurancePlanType
InsurancePlan
Non-Medical
Medical
ReasonForVisit
AppointmentDate
PlanID
Status
Appointment
AppointmentID
make
covers
have
PaymentDate
LicenseStatus
BillingDate
PaymentID
PercentCovered
generates
Amount
has
Invoice
pay
Payment
ExpirationDate
Licensure
PaymentType
InvoiceStatus
Amount
ProcedureDate
LicensureType
perform
ExpirationDate
LicenseNumber
InvoiceID
Currency
generates
CardNo
UnitCharge
Procedure
Cash
d
Credit
SecurityCode
CardholderName
RecoveryLength
ProcedureType
AccountNo
Check
ProcedureID
ProcedureDate
Qty
CheckNo
RoutingNo
1
N
M

Part 2: Normalization

The final schema is in bold.

Patient(PatientID (FK), LastSignedHIPPA, LastXRay, PlanID)

- Dependency: {PatientID} -> {LastSignedHIPPA, LastXRay,PlanID}
- No partial or transitive dependency, determinants are candidate keys, BCNF

Employee (EmployeeID (FK), MedicalFlag, Specialty)

- Dependency: {EmployeeID} -> {MedicalFlag, Specialty}
- No partial or transitive dependency, determinants are candidate keys, BCNF

Payment (PaymentID, PaymentDate, Amount, PaymentType, Currency, PatientID (FK))

- Dependency: {PaymentID} -> {PaymentDate, Amount, PaymentType, Currency, PatientID}
- No partial or transitive dependency, determinants are candidate keys, BCNF

Licensure (LicenseNumber, LicenseType, ExpirationDate, LicenseStatus, MedicalID (FK))

- Dependency: {LicenseNumber} -> {LicenseType, ExpirationDate, LicenseStatus, MedicalID}
- No partial or transitive dependency, determinants are candidate keys, BCNF

Invoice (InvoiceID, BillingDate, AppointmentID (FK), ProcedureDate, MedicalID (FK), Amount)

- Partial dependency: {AppointmentID} -> {ProcedureDate, MedicalID}
    - These attributes (ProcedureDate, MedicalID) are fully dependent on AppointmentID, so they are removed from the Invoice table
- Partial dependency: {InvoiceID} -> {BillingDate, Amount}
- Partial dependency, 1NF

Invoice(InvoiceID (PK), BillingDate, Amount)

- Dependency: {InvoiceID} -> {BillingDate, Amount}
- No partial or transitive dependency, determinants are candidate keys, BCNF

Appointment (ApptID, ApptDate, Status, ReasonForVisit, PatientID (FK), MedicalID (FK), InvoiceID (FK))

- Dependency: {ApptID} -> {ApptDate, Status, ReasonForVisit, PatientID, MedicalID, InvoiceID}
- No partial or transitive dependency, determinants are candidate keys, BCNF

Check (PaymentID (FK), AccountNo, CheckNo, RoutingNo)

- Dependency: {PaymentID} -> {AccountNo}
- Transitive Dependency: {AccountNo} -> {CheckNo, RoutingNo}
- Transitive but no partial dependency, 2NF, split into two tables

CheckingAccount(PaymentID (FK), AccountNo)

- Dependency: {PaymentID} -> {AccountNo}
- No partial or transitive dependency, determinants are candidate keys, BCNF

CheckInfo(AccountNo (FK), CheckNo, RoutingNo)

- Dependency: {AccountNo} -> {CheckNo, RoutingNo}
- No partial or transitive dependency, determinants are candidate keys, BCNF

Credit (PaymentID (FK), CardNo, CardholderName, SecurityCode, ExpirationDate)

- Dependency: {PaymentID} -> {CardNo}
- Transitive Dependency: {CardNo} -> {CardholderName, SecurityCode, ExpirationDate}
- Transitive but no partial dependency, 2NF, split into two tables

CreditAccount(PaymentID (FK), CardNo)

- Dependency: {PaymentID} -> {CardNo}
- No partial or transitive dependency, determinants are candidate keys, BCNF

CreditCardInfo(CardNo (FK), CardholderName, SecurityCode, ExpirationDate)

- Dependency: {CardNo} -> {CardholderName, SecurityCode, ExpirationDate}
- No partial or transitive dependency, determinants are candidate keys, BCNF

NonMedical_Languages (EmployeeID (FK), LanguagesSpoken)

- Dependency: {LanguagesSpoken} -> {EmployeeID}
- No partial or transitive dependency, determinants are candidate keys, BCNF

InsurancePlan_Procedure (PlanID (FK), ProcedureID (FK), PercentCovered)

- Dependency: {PlanID, ProcedureID} -> {PercentCovered}
- No partial or transitive dependency, determinants are candidate keys, BCNF

Licensure_Procedure (LicenseNumber (FK), ProcedureID (FK))

- No dependencies, BCNF

Procedure_Invoice (ProcedureID (FK), InvoiceID (FK), Qty)

- Dependency: {ProcedureID, InvoiceID} -> {Qty}
- No partial or transitive dependency, determinants are candidate keys, BCNF

Invoice_Payment (InvoiceID (FK), PaymentID (FK), InvoiceStatus)

- Dependency: {InvoiceID, PaymentID} -> {InvoiceStatus}
- No partial or transitive dependency, determinants are candidate keys, BCNF

Patient_Medication (PatientID (FK), MedicationID (FK), PrescripEnd, PrescripStart)

- Dependency: {PatientID, MedicationID} -> {PrescripEnd, PrescripStart}
- No partial or transitive dependency, determinants are candidate keys, BCNF

Patient_Allergy (PatientID (FK), AllergyID (FK))

- No dependency, BCNF

Patient_MedicalCondition (PatientID (FK), ConditionID (FK), DiagnosisDate)

- Dependency: {PatientID, ConditionID} -> {DiagnosisDate}
- No partial or transitive dependency, determinants are candidate keys, BCNF

Person_Address (AddressID (FK), PersonID (FK), AddressStatus)

- Dependency: {AddressID, PersonID} -> {AddressStatus}
- No partial or transitive dependency, determinants are candidate keys, BCNF

EmergencyContact_Patient (EmergencyPersonID (FK), Relationship, PatientID (FK))

- Dependency: {PatientID, EmergencyPersonID} -> {Relationship}
- No partial or transitive dependency, determinants are candidate keys, BCNF

Address (AddressID, StreetAddress, Zipcode, State, City, Country, MemberID (FK))

- *Dependency: {AddressID} -> {StreetAddress, Zipcode, State, City, Country}*
- No partial or transitive dependency, determinants are candidate keys, BCNF

InsurancePlan (PlanID, InsurancePlanType, ActiveDate, MemberID (FK), PatientID (FK))

- *Dependency: {PlanID} -> {InsurancePlanType, ActiveDate, MemberID}*
- No partial or transitive dependency, determinants are candidate keys, BCNF

Person (PersonID, FName, LName, SSN, DateOfBirth, BiologicalSex, PhoneNumber)

- *Dependency: {PersonID} -> {Fname, Lname, SSN, DateOfBirth, BiologicalSex, PhoneNumber}*

InsuranceCompany (MemberID, CompanyName, CompanyEmail, GroupNumber, CompanyPhone, AddressID)

- *Dependency: {MemberID} -> {CompanyNmae, CompanyEmail, GroupNumber, CompanyPhone, AddressID}*
- No partial or transitive dependency, determinants are candidate keys, BCNF

Procedure (ProcedureID, UnitCharge, RecoveryLength, ProcedureType, ProcedureDate)

- *Dependency: {ProcedureID} -> {UnitCharge, RecoveryLength, ProcedureType, ProcedureDate}*
- No partial or transitive dependency, determinants are candidate keys, BCNF

Allergy (AllergyID, Allergen, AllergyType, Severity)

- *Dependency: {AllergyID} -> {Allergen, AllergyType, Severity}*
- No partial or transitive dependency, determinants are candidate keys, BCNF

Medication (MedicationID, DrugName MedicationType, Dosage)

- *Dependency: {MedicationID} -> {DrugName, MedicationType, Dosage}*
- No partial or transitive dependency, determinants are candidate keys, BCNF

MedicalCondition (ConditionID, ConditionName)

- *Dependency: {ConditionID} -> {ConditionName}*
- No partial or transitive dependency, determinants are candidate keys, BCNF

Part 3: Defining Database with SQL

Creating the database is done through CreateQueries.txt. Filling the database is done through InsertQueries.txt. Both files are attached. ****SQLite was used**.**

Part 4: SQL Queries

This was done through SimpleQueries.txt, and it is attached.

Part 5: Additional Queries

This was done through ExtraQueries.txt, and it is attached.

Part 6: Cross Check

We have checked that all of our SQL code is properly formatted, easy to read, and labeled each query with SQL supported comments.

Part 7: Team Member Contributions

All members contributed to this project. Team work has been good and we regularly meet to work on the project.

Part 8: All of the Work

Relational Queries

a. Create a list of patients and the medications they currently take
   i. Patient_Person ← $\pi_{(\text{PatientIID, FName, LName})}$(Person $\bowtie_{\text{PersonID = PatientID}}$ Patient)
   ii. Current_Medication ← $\sigma_{\text{PrescripStart<TODAY (PrescipEnd > TODAY OR PrescripEnd = NULL)}}$(Patient_Medication)
   iii. Medication_Info ← $\pi_{(\text{PatientID, DrugName})}$(Current_Medication * Medication)
   iv. **Patient_Medication ← $\pi_{(\text{FName, LName, DrugName})}$ (Patient_Person * Medication_Info)**
b. Display Patient information for patients who currently have Delta Dental insurance policy
   i. Patient_Person ←(Person $\bowtie_{\text{PersonID = PatientID}}$ Patient)
   ii. Plan_Company ← $\pi_{(\text{PatientID, CompanyName})}$ (InsurancePlan * InsuranceCompany)
   iii. **DeltaDental_Patients ← $\sigma_{\text{CompanyName="Delta Dental"}}$ (Patient_Person * Plan_Company)**
c. Generate a list of procedures and service dates performed by Dr. Smillow

   i. Medical_Employee $\leftarrow \sigma_{\text{MedicalFlag = “True”}}$(Employee)
   ii. Medical_Person $\leftarrow$ Medical_Employee $\bowtie_{\text{PersonID = EmployeeID}}$ Person
   iii. Dr_Smillow $\leftarrow \sigma_{\text{Lname = “Smillow”}}$ Medical_Person
   iv. Smillow_LicenseNo $\leftarrow$ Dr_Smillow * Licensure_Procedure
   v. **Smillow_Procedures $\leftarrow$ Smillow_LicenseNo * Procedure**
   vi. Name_Date $\leftarrow \pi_{\text{ProcedureID, ProcedureType, ProcedureDate}}$(Procedure)
   vii. **Result $\leftarrow$ Smillow_Procedures * Name_Date**

d. Print out a list of due invoices with patient contact info.
   i. Past_Due $\leftarrow \sigma_{\text{(Amount > 10) AND (TODAY >= BillingDate + 30)}}$ Invoice
   ii. PastDue_Invoices $\leftarrow$ Invoice * Past_Due
   iii. Patient_Info $\leftarrow \pi_{\text{PhoneNumber}}$(Patient $\bowtie_{\text{PatientID = PersonID}}$ Person)
   iv. **Result $\leftarrow$ PastDue_Invoices $\bowtie_{\text{PersonID = PatientID}}$ Patient_Info**

e. Find the patients who brought the most revenue in the past year
   i. Patient_Person $\leftarrow$ Patient $\bowtie_{\text{PatientID = PersonID}}$Person
   ii. Invoices $\leftarrow$ Patient_Person * Invoice
   iii. Current_Invoices $\leftarrow \sigma_{\text{BillingDate >= ’2022-01-01’ AND BillingDate < ’2023-01-01’}}$(Invoices)
   iv. **Current_Invoice_Sums $\leftarrow {}_{\text{(Fname, Lname)}}\mathfrak{F}_{\text{SUM Amount}}$(Current_Invoices)**
   v. *SQL features can now be used to sort the top X number of patients in Current_Invoice_Sums.*

f. Create a list of doctors who performed less than 5 procedures this year.
   i. Medical_Employee $\leftarrow \sigma_{\text{MedicalFlag = “True”}}$(Employee)
   ii. Medical_Person $\leftarrow$ Medical_Employee $\bowtie_{\text{EmployeeID = PersonID}}$Person
   iii. LicenseNos $\leftarrow$ Medical_Person * Licensure_Procedure
   iv. Procedures $\leftarrow$ LicenseNos * Procedure
   v. Current_Procedures $\leftarrow \sigma_{\text{ProcedureDate > ’2023-01-01’}}$(Procedures)
   vi. Current_Procedures_Count $\leftarrow {}_{\text{(Fname, Lname)}}\mathfrak{F}_{\text{COUNT ProcedureID}}$(Current_Procedures)
   vii. **Result $\leftarrow \pi_{\text{Fname, Lname}}(\sigma_{\text{Count\_procedureid < 5}}$(Current_Procedures_Count))**

g. Find the highest paying procedures, procedure price, and the total number of those procedures performed.
   i. Procedure_Info $\leftarrow \pi_{\text{ProcedureID, ProcedureType, UnitCharge}}$ (Procedure)
   ii. Procedure_Count $\leftarrow {}_{\text{ProcedureType}}\mathfrak{F}_{\text{COUNT ProcedureID}}$ (Procedure_Info)
   iii. Procedure_Price $\leftarrow \pi_{\text{ProcedureType, UnitCharge, COUNT\_ProcedureID}}$(Procedure_Count * Procedure_Info)
   iv. *SQL features can now be used to sort the top X highest paying procedures in Procedure_Price.*

h. Create a list of all payment types accepted, number of times each of them was used, and total amount charged to that type of payment.
   i. Payment_Type_Amount $\leftarrow \pi_{\text{PaymentType, Amount}}$(Payment)
   ii. Sum_Count $\leftarrow {}_{\text{PaymentType}}\mathcal{F}_{\text{COUNT PaymentType, SUM Amount}}$ (Payment_Type_Amount)
   iii. **Result $\leftarrow \pi_{\text{PaymentType, COUNT Amount, SUM Amount}}$(Sum_Count)**

i. List ids and names of insurance plans ever used by patients and how many patients have that plan.
   i. Plan_Description $\leftarrow \rho_{\text{PlanID, PlanName, ActiveDate, MemberID, PatientID}}$(InsurancePlan)
   ii. Id_Name $\leftarrow \pi_{\text{PlanID, PlanName}}$(Plan_Description)
   iii. Id_PlanName_Count $\leftarrow {}_{\text{PlanID, PlanName}}\mathcal{F}_{\text{COUNT PlanName}}$ (Id_Name)
   iv. **Result $\leftarrow \pi_{\text{PlanID, PlanName, COUNT PlanName}}$(Id_PlanName_Count)**

Additional Interesting Queries

a. Outerjoins: List all the employee names, and if they have a license, display their license number and expiration date. List all employee names regardless if they have a license.
   i. Employee_Person $\leftarrow \pi_{\text{(EmployeeID, FName, LName)}}$(Person $\bowtie_{\text{PersonID = EmployeeID}}$ Employee)
   ii. Employee_License $\leftarrow$ (Employee_Person $⟕_{\text{(EmployeeID=MedicalID)}}$ Licensure)
   iii. **EmployeeInfo_License $\leftarrow \pi_{\text{(FName, LName, LicenseNumber, ExpirationDate)}}$(Employee_License)**

b. Aggregate Functions: Find the total number of allergies grouped by patient.
   i. Patient_Person $\leftarrow$ (Person $\bowtie_{\text{PersonID = PatientID}}$ Patient)
   ii. **Result $\leftarrow {}_{\text{PatientID}}\mathcal{F}_{\text{SUM(COUNT AllergyID)}}$ Patient_Person $\bowtie_{\text{PersonID = PatientID}}$ Patient_Allergy**

c. Extra Entities from PART 1: List the names of each person and the state of their addresses where the address is valid.
   i. P_A $\leftarrow \pi_{\text{(FName, LName, AddressID)}}$(Person * Person_Address)
   ii. PersonState $\leftarrow \pi_{\text{(FName, LName, State, AddressStatus)}}$(P_A * Address)
   iii. **Result $\leftarrow \pi_{\text{(FName, LName, State)}}$ ($\sigma_{\text{AddressStatus = 'valid'}}$(PersonState))**

Part 2 Feedback

ERD: 1. Emergency Contact needs attributes. Otherwise, nice work updating things!
Schema: 1:1 Relationships not handled correctly - the FK only goes on the side with mandatory participation. It looks like you did not finish Step 4 of your step-by-step

mapping. Algebra: See annotated pdf, overall everything looks pretty good, just a couple small errors. Table Specs: looks good