Introduction to Database Systems Final Project

Daniel Mogilevsky

Caden Rice

Vincent Zhang

Yoojin Jeong

CSE 3241

The Ohio State University

Columbus, OH

3/14/2021

**Table of Contents**

1. **Section 1** …………………………………………………………………………..2
2. Introduction ………………………………………………………………………..2
3. ERD Model ………………………………………………………………………..2
4. Relational Schema …………………………………………………………….…..3
5. Relational Algebra ………………………………………………………………...9
6. Database Normalization ………………………………………………………….12
7. **Section 2 (User Manual)** ………………………………………………………..13
8. Databse Tables……………………………………………………………………13
9. Select Queries.……………………………………………………………………21
10. Insert and Delete Queries.……………..…………………………………………27
11. Indexes.……………..……………………………………………………………29
12. Views………………….……………..…………………………………………..30
13. Transactions…………….……………..…………………………………………31
14. **Section 3 (Team Reports and Checkpoint Documents)**......………………....31
15. Team Member Contributions……….....…………………………………………31
16. Reflection On Completion Progress…..…………………………………………32
17. Description of Feedback Recieved/…...…………………………………………32
18. Previous Project Checkpoints..………..…………………………………………32

**Introduction**

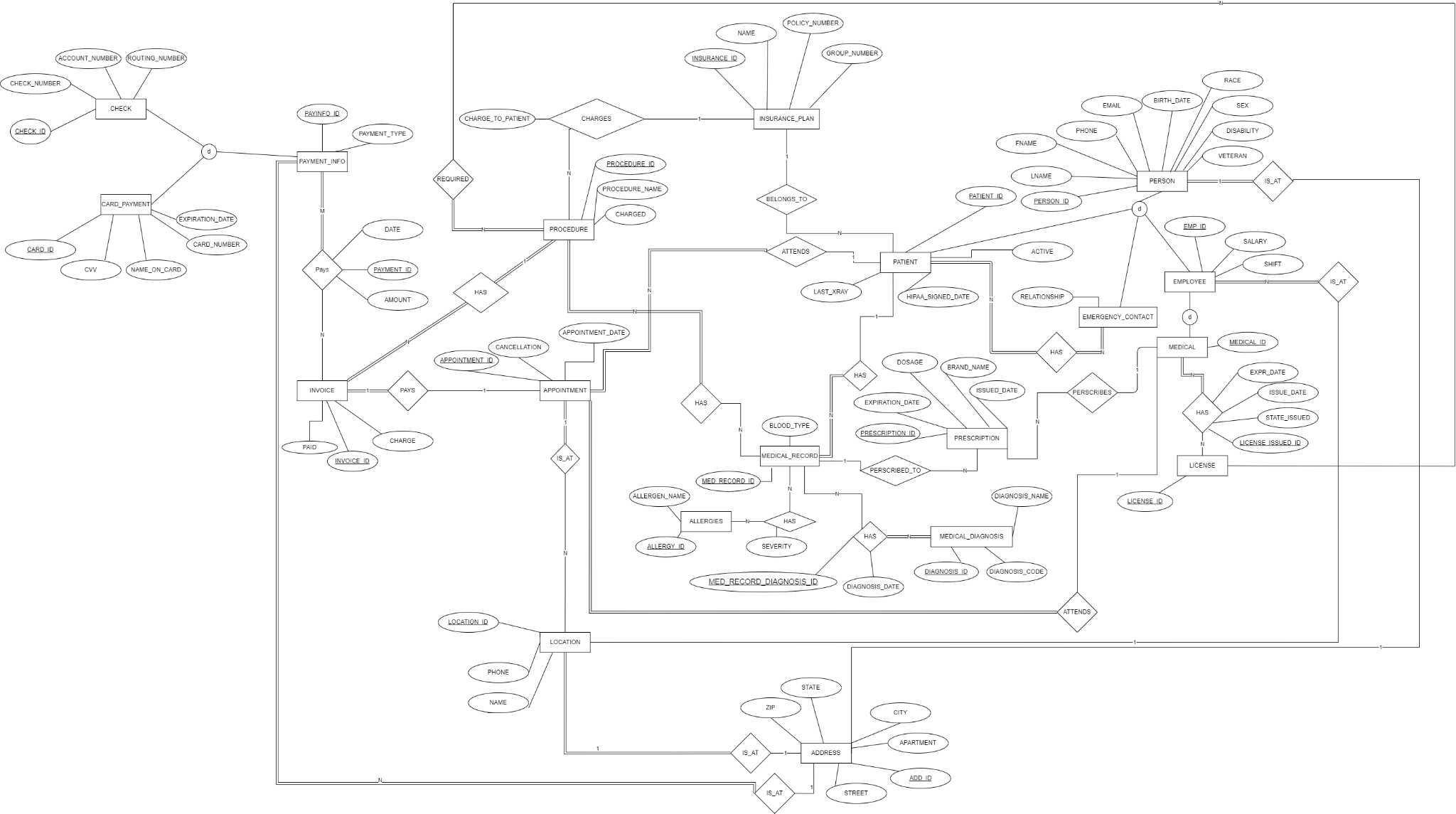
Our team, consisting of Daniel Mogilevsky, Caden Rice, Vincent Zhang, and Yoojin Jeong, has produced a database for use in a dental office. The database stores all the necessary information for the functioning of a dental office, such as patient, employee, and medical information and includes pre-built queries that allow retrieval of this information in a manner immediately useful for dental office operations. For example, our database stores patient payment information for later use and is capable of showing which patients produced the most revenue for the office.

Using an SQL database offers many benefits over traditional file storage of information. Unlike traditional file storage, information and records can be related to each other and queried in a more complex way, for example, showing all of the patients seen by a certain doctor. Storing such associations in a flat-file structure would take a significant amount of storage and be difficult to maintain. Thus the database approach provides far greater functionality than the traditional approach, as well as greater performance, scalability, and reliability.

Our database will support the financial, employee management, patient management, and appointment and procedure scheduling aspect of running a dental office. It will additionally include the option to add new dental office locations in case of business expansions and store prescriptions prescribed to patients by doctors.

**ERD Model**

The ERD diagram can be found on page 3. The diagram uses chen notation to represent entity relationships. Each entity in the database has a corresponding rectangle to represent itself on the ERD. Using diamonds along with chen notations, we are able to represent entity to entity relationships. Double lines represent mandatory participation, while single lines represent optional participation. Using a “d” inside a circle along with connecting lines, we are able to represent sub/super classes. Using an oval, we represent entity and relationship attributes. If the text inside the oval is underlined, it represents a key attribute. All entities are related to one other entity, leaving no stand-alone entities.



### **Relational Schema**

**Procedure entity with procedure\_ID as a primary key. procedure\_Name, charge, required\_License\_Type as attributes. Foriegn key medRecord\_ID from Medical\_Record entity, medical\_ID from Medical entity, license\_ID from License entity, and invoice\_ID from Invoice entity.**

* Procedures - procedure\_ID, procedure\_Name, medRecord\_ID, medical\_ID, charge (drawn from Procedure\_Charges), License\_ID, invoice\_ID

| procedure\_ID (PK) | procedure\_Name | medRecord\_ID (FK) | medical\_ID (FK) | charge | License\_ID(FK) | invoice\_ID (FK) |
| --- | --- | --- | --- | --- | --- | --- |

**Insurance\_Plan entity with insurance\_ID as a primary key. Attributes are Name, Policy\_Number, Group\_Number.**

* Insurance Plan - insurance\_ID, Name, Policy\_Number, Group\_Number

| *insurance\_ID (PK)* | *Name* | *Policy\_Number* | *Group\_Number* |
| --- | --- | --- | --- |

**Procedure\_Charges is a many-to-many relationship between Procedure entity and Insurance\_Plan entity. There are two primary keys, insurance\_ID and procedure\_Name from each entity. Insurance\_ID attribute is also a foriegn key coming from the Insurance\_Plan entity. charge\_To\_Patient is an attribute of a relationship.**

*Procedure\_Charges - Procedure\_ChargeID, insurance\_ID, procedure\_Name, chargeToPatient (Many-many relationship)*

| *insurance\_ID (FK) (PK)* | *procedure\_Name (PK)* | *chargeToPatient* |
| --- | --- | --- |

**Appointment\_Invoices is a one-to-one relationship between Appointment entity and Invoice entity. Thus, this entity gets two primary keys from each entity.**

*Appointment\_Invoices - appointment\_ID, Invoice\_ID (one to one)*

| *apointment\_ID (PK)* | *invoice\_ID (PK)* |
| --- | --- |

**Appointment entity with appointment\_ID as a primary key. Having date, cancellation as attributes. Type of cancellation is boolean. Foreign key patient\_ID from Patient entity, medical\_ID from Medical entity, Location\_ID from Location entity.**

* Appointment - appointment\_ID, date, patient\_ID, medical\_ID, cancellation:boolean, Location\_ID

| appointment\_ID (PK) | date | patient\_ID (FK) | medical\_ID(FK) | cancellation | Location\_ID(FK) |
| --- | --- | --- | --- | --- | --- |

**Location entity with Location\_ID as a primary key. Attributes are Phone and Name. addr\_ID is a foreign key from Address entity.**

* Location - Location\_ID, addr\_ID, phone, Name

| Location\_ID (PK) | addr\_ID (FK) | Phone | Name |
| --- | --- | --- | --- |

**Address entity with addr\_ID as a primary key. Attributes are Street, Apartment, City, State, ZIP, and Country.**

* Address - addr\_ID, Street, Apartment, City, State, ZIP, Country

| addr\_ID(PK) | Street | Apartment | City | State | ZIP | Country |
| --- | --- | --- | --- | --- | --- | --- |

**Invoice entity with Invoice\_ID as a primary key. Attributes are Charge and Paid. Type of Paid is boolean. patient\_ID is a foreign key from Patient entity.**

* Invoice - invoice\_ID, patient\_ID, charge, paid (boolean)

| invoice\_ID (PK) | charge | paid(boolean) | patient\_ID (FK) |
| --- | --- | --- | --- |

**Payment\_Info entity with payInfo\_ID as a primary key. Type is an attribute and there are three options: cash, card, check.**

* Payment\_Info - payInfo\_ID, type(Cash, Card, Check)

| payInfo\_ID (PK) | type |
| --- | --- |

**Card\_Payment is a subclass of the Payment\_Info entity. Having cardPaymentId as a primary key, payInfo\_ID as a foreign key which is from Payment\_Info entity. Card\_number, cvv, expiration\_date, name\_on\_card, addr\_ID are attributes.**

○ Card\_Payment - cardPaymentId, payInfo\_ID, card\_number, cvv, expiration\_date, name\_on\_card, addr\_ID

| cardPaymentId (PK) | payInfo\_ID(FK) | card\_number | cvv | expiration\_date | name\_on\_card | addr\_ID(FK) |
| --- | --- | --- | --- | --- | --- | --- |

**Check is another subclass of the Payment\_Info entity. CheckID is a primary key and checkNumber, accountNumber, routingNumber are attributes. payInfo\_ID is a foreign key from Payment\_Info entity.**

○ Check - checkNumber, accountNumber, routingNumber, payInfo\_ID

| CheckID (PK) | checkNumber | accountNumber | routingNumber | payInfo\_ID(FK) |
| --- | --- | --- | --- | --- |

**InvoicePayment is a many-to-many relationship between Payment\_Info entity and Invoice entity. Primary key is payment\_ID and attributes are Date and Amount. Foreign keys are from each entity, Payment\_Info and Invoice.**

* InvoicePayment - payment\_ID, date, amount, payInfo\_ID, invoice\_ID

| payment\_ID(PK) | date | amount | payInfo\_ID (FK) | invoice\_ID(FK) |
| --- | --- | --- | --- | --- |

**Prescriptions is an entity with Prescription\_ID as a primary key. Attributes are dosage, brand\_name, issued\_Date, and expiration\_Date. Foreign keys from Medical entity and Medical\_Record entity.**

* Prescriptions - Prescription\_ID, Medical\_ID, dosage, brand\_name, medRecord\_ID, issued\_Date, expiration\_Date

| prescription\_ID (PK) | Medicial\_ID(FK) | dosage | brand\_name | medRecord\_ID(FK) | issued\_Date | expiration\_Date |
| --- | --- | --- | --- | --- | --- | --- |

**Allergies entity has allergy\_ID as a primary key, allergen\_name as an attribute.**

* Allergies - allergy\_ID, allergen\_name

| allergy\_ID (PK) | allergen\_name |
| --- | --- |

**Medical\_Record entity has medRecord\_ID as a primary key and blood\_type as an attribute.**

* Medical\_Record - medRecord\_ID, blood type

| medRecord\_ID(PK) | Blood type |
| --- | --- |

**Medical\_Diagnosis has diagnosis\_ID as a primary key. Attributes are diagnosisCode and diagnosisName.**

* Medical\_Diagnosis - diagnosis\_ID, diagnosisCode, diagnosisName

| diagnosis\_ID(PK) | diagnosisCode | diagnosisName |
| --- | --- | --- |

**MedicalRecord\_Allergies is a many-to-many relationship between Medical\_Record entity and Allergies entity. There are two primary keys, allergy\_ID and medRecord\_ID from each entity. Attribute named Severity has three options: mild, severe, deadly.**

*medicalRecord\_Allergies - allergy\_ID, medRecord\_ID, severity (Mild, Severe, Deadly) (Many to many relationship)*

| allergy\_ID(PK) | medRecord\_ID(PK) | severity |
| --- | --- | --- |

**MedicalRecord\_Diagnosis is a many-to-many relationship between Medical\_Record entity and Medical\_Diagnosis entity. It has its own primary key, medRecord\_DiagnosisID. diagnosisDate is an attribute of a relationship. Foriegn keys are medRecord\_ID from medical\_Record entity and diagnosisID from Diagnosis entity.**

*medicalRecord\_Diagnosis - medRecord\_ID, diagnosisID, diagnosisDate (Many to many relationship)*

| *medRecord\_DiagnosisID (PK)* | *medRecord\_ID(FK)* | *diagnosisID(FK)* | *diagnosisDate* |
| --- | --- | --- | --- |

**License has License\_ID as a primary key. Attributes are Type and Licensure\_Name.**

* License - License\_ID, Type, Licensure\_Name

| License\_ID (PK) | type | Licensure\_Name |
| --- | --- | --- |

**Medical\_Licenses is a many-to-many relationship between Medical entity and License entity. A primary key is LicenseIssued\_ID. Attributes are Issue\_Date, Exp\_Date, stateIssued. Foreign keys are Medical\_ID from Medical entity and License\_ID from License entity.**

*medical\_Licenses - medical\_ID, License\_ID, Issue\_Date, Exp\_Date, stateIssued (Ohio, Virginia) (Many to many)*

| licenseIssued\_ID(PK) | medical\_ID(FK) | License\_ID(FK) | Issue\_Date | Exp\_Date | stateIssued |
| --- | --- | --- | --- | --- | --- |

**Patient\_medRecord is a one-to-one relationship between Patient entity and Medical\_Record entity. There are two primary keys from each entity.**

*patient\_medRecord - patientID, medRecord\_ID (one to one relationship)*

| *patientID (PK)* | *medRecord\_ID (PK)* |
| --- | --- |

**Person entity has person\_ID as a primary key. Attributes are SSN, fname, lname, phone, email, birth\_date, race, sex, disability, and veteran. Sex has two options: male and female. Type of disability and veteran is boolean. Addr\_ID is a foreign key from Address entity.**

* Person - personId, SSN, fname, lname, phone, email, addr\_ID, birth\_date, race, sex (M, F), disability (boolean), veteran (boolean)

| person\_ID(PK) | SSN | fname | lname | phone | race | sex | disability | veteran | addr\_ID(FK) | email | birth\_date |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Patient is a subclass of the Person entity. Patient\_ID is a primary key. Attributes are active, HIPPA\_signed\_date, and last\_xRay. Type of Active attribute is boolean. Person\_ID is a foriegn key from the superclass, Person. Foreign keys are payInfo\_ID from Payment\_Info, medical\_ID from Medical, insurance\_ID from Insurance\_Plan.**

○ Patient - patient\_ID, person\_ID, active:boolean, payInfo\_ID, medical\_ID, insurance\_ID, HIPAA\_signed\_date (a date field, but it empty serves as true/false), last\_xRay

| patient\_ID(PK) | person\_ID(FK) | active | payInfo\_ID(FK) | medical\_ID(FK) | insurance\_ID(FK) | HIPAA\_signed\_date | last\_xRay |
| --- | --- | --- | --- | --- | --- | --- | --- |

**Emergency\_Contacts is a subclass of the Person entity. Having person\_ID and patient\_ID as primary keys. Attribute relationship telling relationship between patient and the contact user. Patient and Emergency\_Contact have many to many relationship.**

○ Emergency\_Contacts - personID, patient\_ID, relationship (many to many)

| person\_ID (PK) | patient\_ID (PK) | relationship |
| --- | --- | --- |

**Employee is a subclass of the Person entity. Emp\_ID is a primary key. Attributes are Salary and Shift. Person\_ID is a foreign key from the superclass, Person. Another foreign key, Location\_ID is from the Location entity.**

○ Employee - Emp\_ID, personID, Salary, shift, Location\_ID

| Emp\_ID(PK) | person\_ID(FK) | Salary | Shift | Location\_ID(FK) |
| --- | --- | --- | --- | --- |

**Medical is a subclass of the Employee entity, which is a subclass of the Person entity. Medical entity has medical\_ID as a primary key. Foreign key, emp\_ID is from the Employee entity, which is the superclass.**

○ Medical - medical\_ID, emp\_ID

| medical\_ID (PK) | emp\_ID (FK) |
| --- | --- |

**Relational Algebra**

The following are the select queries provided as part of our database package, written in relational algebra.

**Create a list of patients and the medications they currently take**

PATIENTS\_WITH\_MED\_ID ← PATIENT ⨝ Patient.patientlID= patient\_medRecord.patientID (patient\_medRecord)

PATIENTS\_WITH\_MED\_RECORDS ← PATIENT\_WITH\_MED\_ID ⨝ PATIENT\_WITH\_MED\_ID.medRecordlID= medical\_Record.medRecordtID (medical\_Record)

PATIENTS\_WITH\_PRESCRIPTIONS ← PATIENTS\_WITH\_MED\_RECORDS ⨝ PATIENT\_WITH\_MED\_RECORD..medRecordlID= Prescriptions.medRecordtID AND Presciptions.Active = ‘true’ (Prescriptions)

PEOPLE\_WITH\_PRESCRIPTIONS ← PATIENTS\_WITH\_PRESCRIPTIONS ⨝ PATIENT\_WITH\_PRESCRIPTIONS.personID= Person.personID (Person)

patientID,fname,lname F brand\_name,dosage(PEOPLE\_WITH\_PRESCRIPTIONS )

**Display patient information for patients who currently have Delta Dental insurance**

**policy.**

PATIENT ⨝ Patient.insuranceID= INSURANCE\_PLAN.insuranceID σ(Name=’Delta Dental’)(INSURANCE\_PLAN)

**Generate a list of procedures and dates of service performed by doctor Smilow.**

SMILOW\_PROCEDURES ← PROCEDURE ⨝ Procedure.medicalID= Medical.medicalID σ(fname=’Smilow’)(Medical)

INVOICE\_APPOINTMENTS ← INVOICE ⨝ Invoice.invoicelID= Appointment\_Invoices.invoiceID (Appointment\_Invoices)

INVOICE\_WITH\_DATES ←INVOICE\_APPOINTMENTS ⨝ INVOICE\_APPOINTMENTS.appointmentlID= Appointment.appointmentID (Appointment)

SMILOW\_PROCEDURES\_WITH\_DATES ← SMILOW\_PROCEDURES ⨝ SMILOW\_PROCEDURES.invoicelID= INVOICE\_WITH\_DATES.invoiceID (INVOICE\_WITH\_DATES)

πprocedure\_Name, date (SMILOW\_PROCEDURES\_WITH\_DATES)

**Print out a list of past due invoices with patient contact information. Past due is**

**defined as over 30 days old with a balance over $10.**

PAST\_DUE ← σ paid = false (σ charge > 10 (σ TODAY() - invoice.date > 30 (INVOICE)))

PATIENT\_PAYMENT ← PATIENT ⨝ patient.payInfo\_ID = payment.payInfo\_ID (PAYMENT)

PATIENT\_PAST\_INVOICE ← PATIENT\_PAYMENT ⨝ patient\_payment.invoiceID = past\_due.invoiceID (PAST\_DUE)

PERSON\_WITH\_PAST\_INVOICE ← PATIENT\_PAST\_INVOICE ⨝ patient.personID = person.personID (PERSON)

π phone, email (PERSON\_WITH\_PAST\_INVOICE)

**Find the patients who brought the most revenue in the past year.**

INVOICE\_APPOINTMENT ← INVOICE ⨝ invoice.patient\_ID = appointment.patient\_ID (APPOINTMENT)

INVOICE\_APPOINTMENT\_PATIENT ← INVOICE\_APPOINTMENT ⨝ invoice\_appointment.patient\_ID = patient.patient\_ID (PATIENT)

INVOICE\_APPINTMENT\_PATIENT\_PAYMENT ← INVOICE\_APPOINTMENT\_PATIENT ⨝ invoice\_appointment\_patient.invoice\_ID = payment.invoice\_ID (PAYMENT)

PATIENT\_CHARGE ← π patient\_ID, charge (INVOICE\_APPINTMENT\_PATIENT\_PAYMENT) patient\_ID F SUM charge (PATIENT\_CHARGE)

**Create a list of doctors who performed less than 5 procedures this year.**

DOCTORS\_NUM\_PROCEDURES = p MedID, Procedure\_Count (Medical\_Id,FCount(Procedure\_id)(PROCEDURE))

DOCTORS\_LESS = σ(Procedure\_Count < 5)(MEDICAL)

RESULT = ΠFName, Lname, Medical\_ID (DOCTORS\_LESS ⨝ MedID = Medical\_ID (MEDICAL))

**Find the highest paying procedures, procedure price, and the total number of those**

**procedures performed.**

MAX\_CHARGE\_AMOUNT = FMAX(Charge)(PROCEDURE))

PROCEDURE\_INVOICES =σ(Charge = MAX\_CHARGE\_AMOUNT)(PROCEDURE)

PROCEDURE\_INVOICES\_COUNTS = p Procedure\_Name, Procedure\_Count (Procedure\_Name, Charge, FCount(Procedure\_id)(PROCEDURE\_INVOICES))

**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.**

TYPE ← π type (PAYMENT\_INFO)

PAYMENTS\_WITH\_TYPE ← PAYMENT ⨝ payment.paymentInfoID = PAYMENT\_INFO.payInfo\_ID (PAYMENTINFO) type F COUNT, SUM amount (PAYMENTS\_WITH\_TYPE)

**Find the name of the most popular insurance plan currently used by the patients.**

PATIENT\_INSURANCE ← PATIENT ⨝ patient.insurance\_ID = insurancePlan.insurance\_ID (INSURANCE)

INSURANCE\_COUNT ← name F COUNT (PATIENT\_INSURANCE)

F MAX COUNT (INSURANCE\_COUNT)

**List the names and addresses of all the dental offices**

LOCATIONS\_WITH\_ADDRESS ← LOCATION ⨝ Location.addr\_ID= Adress.addr\_ID (ADDRESS)

ΠName, Street, City, State, ZIP, Country(LOCATIONS\_WITH\_ADDRESS)

**Find the first and last name and phone number of the emergency contact for all patients.**

EMERGENCY\_CONTACTS\_WITH\_INFO← ( PERSON ⨝person\_ID = person\_ID EMERGENCY\_CONTACTS)

Πphone,fname,lname (EMERGENCY\_CONTACTS\_WITH\_INFO)

**Find all invoices that are over $10,000 for a patient.**

INFO ← ( PATIENT ⨝patient\_ID = appointment.patient\_ID APPOINTMENT)

CHARGE ← ( INVOICE ⨝ Invoice.invoicelD= Info\_Invoices.invoiceID INFO)

Πcharge > 10,000 (CHARGE)

### **Database Normalization**

According to the feedback given from CP03 we should have all of our tables in 3rd normal form since there were no points taken off, except for maybe a few that had a specific purpose like a composite key. I will be checking through the tables with the following method to make sure it is in 3rd normal form. Firstly, we make sure that it is in the 1st normal form by checking if every value is atomic. For the 2nd normal form we remove every value that is not dependent on the primary key and form a new table with the removed values to make sure every value is dependent on the primary key of the table. Finally, to make sure it is in 3rd normal form we remove the transitive dependencies from each non-key value.

Procedures, Insurance Plan, Appointment, Location, Address, Invoice, Payment\_Info, Card\_Payment, Check, InvoicePayment, Prescriptions, Allergies, License, medical\_Licenses, Medical\_Record, Medical\_Diagnosis, medicalRecord\_Allergies, medicalRecord\_Diagnoses, Person, Patient, Employee should all be in in its 3rd normal form.

Procedures: {Procedure\_ID} -> {ProcedureName, charge, required\_License\_Type}

Insurance Plan: {insurance\_ID} -> {Name, Policy\_Number, Group\_Number}

Appointment: {appointment\_ID} -> {date, cancellation}

Location: {Location\_ID} -> {Phone, Name}

Address: {addr\_ID} -> {Street, Apartment, City, State, ZIP, Country}

Invoice: {invoice\_ID} -> {charge, paid}

Payment\_Info: {payInfo\_ID} -> {type}

Card\_Payment: {cardPaymentId } -> {card\_number, cvv, expiration\_date, name\_on\_card}

Check: {CheckID} -> {checkNumber, accountNumber, routingNumber}

InvoicePayment: {payment\_ID} -> {date, amount}

Prescriptions: {prescription\_ID} -> {dosage, brand\_name, issued\_Date, expiration\_Date}

Allergies: {allergy\_ID} -> {allergy\_name}

License: {License\_ID} -> {type, Licensure\_Name}

medical\_Licenses: {licenseIssued\_ID} -> {Issue\_Date, Exp\_Date,stateIssued}

Medical\_Record: {medRecord\_ID} -> {blood\_type}

Medical\_Diagnosis: {diagnosis\_ID} -> {diagnosisCode, diagnosisName}

medicalRecord\_Allergies: {allergy\_ID, medRecord\_ID}-> {severity}

medicalRecord\_Diagnoses: {medRecord\_DiagnosisID} -> {diagnosisDate}

Person: {person\_ID} -> {SSN,fname, lname, phone, race, sex, disability, veteran, email, birth\_date}

Patient: {patient\_ID} -> {active, HIPAA\_signed\_date, last\_xRay}

Employee: {Emp\_ID} -> {Salary, Shift}

Procedure\_Charges, Appointment\_Invoices, patient\_medRecord, Emergency\_Contacts and Medical are all in 2nd normal form as composite keys to other tables.

## **Section 2 - User Manual (DB Implementation)**

**Database Tables**

**Procedures:** created to store procedures of dental treatments.

| **Field** | procedure\_ID | procedure\_Name | medRecord\_ID | medical\_ID | charge | License\_ID | invoice\_ID |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Key** | Primary key |  | Foreign key | Foreign key |  | Foreign key | Foreign key |
| **Constraint** | PRIMARY KEY |  | FOREIGN KEY  NOT NULL | FOREIGN KEY  NOT NULL |  | FOREIGN KEY  NOT NULL | FOREIGN KEY  NOT NULL |
| **Data type** | integer | char | integer | integer | integer | integer | integer |

Foreign key: MedRecord\_ID from Medical\_Record(MedRecord\_ID),

Medical\_ID from Medical(Medical\_ID),

Invoice\_ID from Invoice(Invoice\_ID),

License\_ID from License(License\_ID)

**Insurance\_Plan:** created to store information about insurance plans.

| **Field** | *insurance\_ID* | *Name* | *Policy\_Number* | *Group\_Number* |
| --- | --- | --- | --- | --- |
| **Key** | Primary key |  |  |  |
| **Constraint** | PRIMARY KEY | NOT NULL | NOT NULL | NOT NULL |
| **Data Type** | Integer | char | Integer | Integer |

**Procedure\_Charges:** created for the many-to-many relationship between Insurance\_Plan entity and Procedure entity.

| **Field** | *insurance\_ID* | *procedure\_Name* | *chargeToPatient* |
| --- | --- | --- | --- |
| **Key** | Primary key,  Foreign key | Primary key,  Foreign key |  |
| **Constraint** | PRIMARY KEY  FOREIGN KEY | PRIMARY KEY  FOREIGN KEY | NOT NULL |
| **Data Type** | Integer | Text | Integer |

Foreign key: Insurance\_ID from Insurance\_Plan(insurance\_ID),

procedure\_Name from Procedures(procedure\_Name)

**Appointment\_Invoices:** created for the one-to-one relationship between Appointment entity and Invoice entity.

| **Field** | *apointment\_ID* | *invoice\_ID* |
| --- | --- | --- |
| **Key** | Primary key | Primary key |
| **Constraint** | PRIMARY KEY  FOREIGN KEY | PRIMARY KEY  FOREIGN KEY |
| **Data Type** | Integer | Integer |

Foreign key: appointment\_ID from Appointment(appointment\_ID),

invoice\_ID from Invoice(Invoice\_ID)

**Appointment:** created to store appointment details for each patient.

| **Field** | appointment\_ID | date | patient\_ID | medical\_ID | cancellation | Location\_ID |
| --- | --- | --- | --- | --- | --- | --- |
| **Key** | Primary key |  | Foreign key | Foreign key |  | Foreign key |
| **Constraint** | PRIMARY KEY | NOT NULL | NOT NULL  FOREIGN KEY | NOT NULL  FOREIGN KEY |  | NOT NULL  FOREIGN KEY |
| **Data Type** | integer | int | integer | integer | boolean | char |

Foreign key: Location\_ID from Location(Location\_ID),

patient\_ID from Patient(patient\_ID),

medical\_ID from Medical(medical\_ID)

**Location:** created to store information about location.

| **Field** | Location\_ID | addr\_ID | Phone | Name |
| --- | --- | --- | --- | --- |
| **Key** | Primary key | Foreign key |  |  |
| **Constraint** | PRIMARY KEY | NOT NULL  FOREIGN KEY | NOT NULL | NOT NULL |
| **Data Type** | integer | integer | char | char |

Foreign key: addr\_ID from Address(addr\_ID)

**Address:** created to store details about address.

| **Field** | addr\_ID | Street | Apartment | City | State | ZIP | Country |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Key** | Primary key |  |  |  |  |  |  |
| **Constraint** | PRIMARY  KEY | NOT NULL |  | NOT NULL | NOT NULL | NOT NULL | NOT NULL |
| **Data Type** | integer | char | char | char | char | char | char |

**Invoice:** created to store invoice details for each patient.

| **Field** | invoice\_ID | charge | paid | patient\_ID |
| --- | --- | --- | --- | --- |
| **Key** | Primary key |  |  | Foreign key |
| **Constraint** | PRIMARY KEY | NOT NULL |  | NOT NULL  FOREIGN KEY |
| **Data Type** | integer | integer | boolean | integer |

Foreign key: patient\_ID from Patient(patient\_ID)

**Payment\_Info:** created to store information about payment; payment type

| **Field** | payInfo\_ID (PK) | type |
| --- | --- | --- |
| **Key** | Primary key |  |
| **Constraint** | PRIMARY KEY | NOT NULL |
| **Data Type** | integer | char |

**Card\_Payment:** created to store information from the card used for the payment.

| **Field** | cardPaymentId | payInfo\_ID | card\_number | cvv | expiration\_date | name\_on\_card | addr\_ID |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Key** | Primary key | Foreign key |  |  |  |  | Foreign key |
| **Constraint** | PRIMARY KEY | FOREIGN KEY  NOT NULL | NOT NULL | NOT NULL | NOT NULL | NOT NULL | FOREIGN KEY  NOT NULL |
| **Data Type** | integer | integer | char | integer | date | char | integer |

Foreign key: payInfo\_ID from Payment\_Info(payInfo\_ID),

addr\_ID from Address(addr\_ID)

**Check:** created to store information from the check used for the payment.

| **Field** | CheckID | checkNumber | accountNumber | routingNumber | payInfo\_ID |
| --- | --- | --- | --- | --- | --- |
| **Key** | Primary key |  |  |  | Foreign key |
| **Constraint** | PRIMARY KEY | NOT NULL | NOT NULL | NOT NULL | FOREIGN KEY  NOT NULL |
| **Data Type** | integer | integer | integer | integer | integer |

Foreign key: payInfo\_ID from Payment\_Info(payInfo\_ID)

**Invoice\_Payment:** created for many-to-many relationship between Invoice entity and Payment entity.

| **Field** | payment\_ID | date | amount | payInfo\_ID | invoice\_ID |
| --- | --- | --- | --- | --- | --- |
| **Key** | Primary key |  |  | Foreign key | Foreign key |
| **Constraint** | PRIMARY KEY | NOT NULL | NOT NULL | NOT NULL  FOREIGN KEY | NOT NULL  FOREIGN KEY |
| **Data Type** | integer | date | integer | integer | integer |

Foreign key: payInfo\_ID from Payment\_Info(payInfo\_ID),

invoice\_ID from Invoice(invoice\_ID)

**Prescriptions:** created to store details of prescriptions.

| **Field** | prescription\_ID | Medicial\_ID | dosage | brand\_name | medRecord\_ID | issued\_Date | expiration\_Date |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Key** | Primary key | Foreign key |  |  | Foreign key |  |  |
| **Constraint** | PRIMARY KEY | NOT NULL  FOREIGN KEY |  |  | NOT NULL  FOREIGN KEY | NOT NULL | NOT NULL |
| **Data Type** | integer | integer | char | char | integer | date | date |

Foreign key: medRecord\_ID from Medical\_Record(medRecord\_ID),

medical\_ID from Medical(medical\_ID)

**Allergies:** created to store information about allergies.

| **Field** | allergy\_ID | allergen\_name |
| --- | --- | --- |
| **Key** | Primary key |  |
| **Constraint** | PRIMARY KEY | NOT NULL |
| **Data Type** | integer | char |

**License:** created to store details about license.

| **Field** | License\_ID | type | Licensure\_Name |
| --- | --- | --- | --- |
| **Key** | Primary key |  |  |
| **Constraint** | PRIMARY KEY | NOT NULL | NOT NULL |
| **Data Type** | integer | char | char |

**Medical\_License:** created for many-to-many relationship between Medical entity and License entity.

| **Field** | licenseIssued\_ID | medical\_ID | License\_ID | Issue\_  Date | Exp\_Date | stateIssued |
| --- | --- | --- | --- | --- | --- | --- |
| **Key** | Primary key | Foreign key | Foreign key |  |  |  |
| **Constraint** | PRIMARY KEY | FOREIGN KEY  NOT NULL | FOREIGN KEY  NOT NULL | NOT NULL | NOT NULL | NOT NULL |
| **Data Type** | integer | integer | integer | date | date | char |

Foreign key: medical\_ID from Medical(medical\_ID),

license\_ID from License(license\_ID)

**Medical\_Record:** created to store medical records such as blood type.

| **Field** | medRecord\_ID | Blood type |
| --- | --- | --- |
| **Key** | Primary key |  |
| **Constraint** | PRIMARY KEY | NOT NULL |
| **Data Type** | integer | char |

**Medical\_Diagnosis:** created to store information about diagnosis.

| **Field** | diagnosis\_ID | diagnosisCode | diagnosisName |
| --- | --- | --- | --- |
| **Key** | Primary key |  |  |
| **Constraint** | PRIMARY KEY | NOT NULL | NOT NULL |
| **Data Type** | integer | char | char |

**MedicalRecord\_Allergies:** created for many-to-many relationship between Allergies entity and Medical\_Record entity.

| **Field** | allergy\_ID | medRecord\_ID | severity |
| --- | --- | --- | --- |
| **Key** | Primary key  Foreign key | Primary key  Foreign key |  |
| **Constraint** | PRIMARY KEY  FOREIGN KEY | PRIMARY KEY  FOREIGN KEY | NOT NULL |
| **Data Type** | integer | integer | char |

Foreign key: allergy\_ID from Allergies(allergy\_ID),

medRecord\_ID from Medical\_Record(medRecord\_ID)

**MedicalRecord\_Diagnosis:** created for many-to-many relationship between Medical\_Record entity and Diagnosis entity.

| **Field** | *medRecord\_DiagnosisID* | *medRecord\_ID* | *diagnosisID* | *diagnosisDate* |
| --- | --- | --- | --- | --- |
| **Key** | Primary key | Foreign key | Foreign key |  |
| **Constraint** | PRIMARY KEY | FOREIGN KEY  NOT NULL | FOREIGN KEY  NOT NULL | NOT NULL |
| **Data Type** | integer | integer | integer | date |

Foreign key: medRecord\_ID from Medical\_Record(medRecord\_ID),

diagnosis\_ID from Medical\_Diasnosis(diagnosis\_ID)

**Patient\_MedRecord:** created for one-to-one relationship between Patient entity and Medical\_Record entity.

| **Field** | *patientID* | *medRecord\_ID* |
| --- | --- | --- |
| **Key** | Primary key  Foreign key | Primary key  Foreign key |
| **Constraint** | PRIMARY KEY  FOREIGN KEY | PRIMARY KEY  FOREIGN KEY |
| **Data Type** | integer | integer |

Foreign key: patient\_ID from Patient(patient\_ID),

medRecord\_ID from Medical\_Record(medRecord\_ID)

**Person:** created to store information about each person in this database.

| **Field** | person\_ID | SSN | fname | lname | phone | race | sex | disability | veteran | addr\_ID | email | birth\_date |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Key** | Primary key |  |  |  |  |  |  |  |  | Foreign key |  |  |
| **Constraint** | PRIMARY KEY | NOT NULL | NOT NULL | NOT NULL | NOT NULL | NOT NULL | NOT NULL | NOT NULL | NOT NULL | FOREIGN KEY  NOT NULL | NOT NULL | NOT NULL |
| **Data Type** | int | int | char | char | int | char | char | char | char | int | char | date |

Foreign key: addr\_ID from Address(addr\_ID)

**Patient:** subclass of the Person entity. Created to store information about patients in the dental clinic.

| **Field** | Patient\_  ID | person\_ID | active | payInfo\_ID | medical\_ID | insurance\_ID | HIPAA\_signed\_date | last\_xRay |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Key** | Primary key | Foriegn key |  | Foreign key | Foreign key | Foreign key |  |  |
| **Constraint** | PRIMARY KEY | NOT NULL  FOREIGN KEY | NOT NULL | NOT NULL  FOREIGN KEY | NOT NULL  FOREIGN KEY | NOT NULL  FOREIGN KEY | NOT NULL | NOT NULL |
| **Data Type** | integer | integer | boolean | integer | integer | integer | date | date |

Foreign key: person\_ID from Person(person\_ID),

payInfo\_ID from Payment\_Info(payInfo\_ID),

medical\_ID from Medical(medical\_ID),

insurance\_ID from Insurance\_Plan(insurance\_ID)

**Emergency\_Contact:** subclass of the Person entity. Many-to-many relationship with Patient entity. Created to store emergency contacts of the patients.

| **Field** | person\_ID | patient\_ID | relationship |
| --- | --- | --- | --- |
| **Key** | Primary key  Foreign key | Primary key |  |
| **Constraint** | PRIMARY KEY  FOREIGN KEY | PRIMARY KEY  FOREIGN KEY | NOT NULL |
| **Data Type** | integer | integer | char |

Foreign key: person\_ID from Person(person\_ID),

patient\_ID from Patient(patient\_ID)

**Employee:** subclass of the Person entity. Created to store information about employees working in the dental clinic.

| **Field** | Emp\_ID | person\_ID | Salary | Shift | Location\_ID |
| --- | --- | --- | --- | --- | --- |
| **Key** | Primary key | Foreign key |  |  | Foreign key |
| **Constraint** | PRIMARY KEY | FOREIGN KEY  NOT NULL | NOT NULL | NOT NULL | FOREIGN KEY  NOT NULL |
| **Data Type** | integer | integer | integer | char | integer |

Foreign key: person\_ID from Person(person\_ID),

location\_ID from Location(location\_ID)

**Medical:** subclass of the Employee entity. Created to store medical employees working in the dental clinic.

| **Field** | medical\_ID (PK) | emp\_ID (FK) |
| --- | --- | --- |
| **Key** | Primary key | Foreign key |
| **Constraint** | PRIMARY KEY | FOREIGN KEY |
| **Date Type** | integer | integer |

Foreign key: emp\_ID from Employee(emp\_ID)

**SQL Select Queries**

Supplied below are the SQL select statements provided with the database.

**Create a list of patients and the medications they currently take, sorted by**

**patient’s last name and medication name in alphabetical order**

SELECT PERSON.LName, PERSON.FName, PRESCRIPTIONS.Brand\_name, PRESCRIPTIONS.Dosage, PRESCRIPTIONS.issued\_Date FROM PATIENT\_MEDRECORD

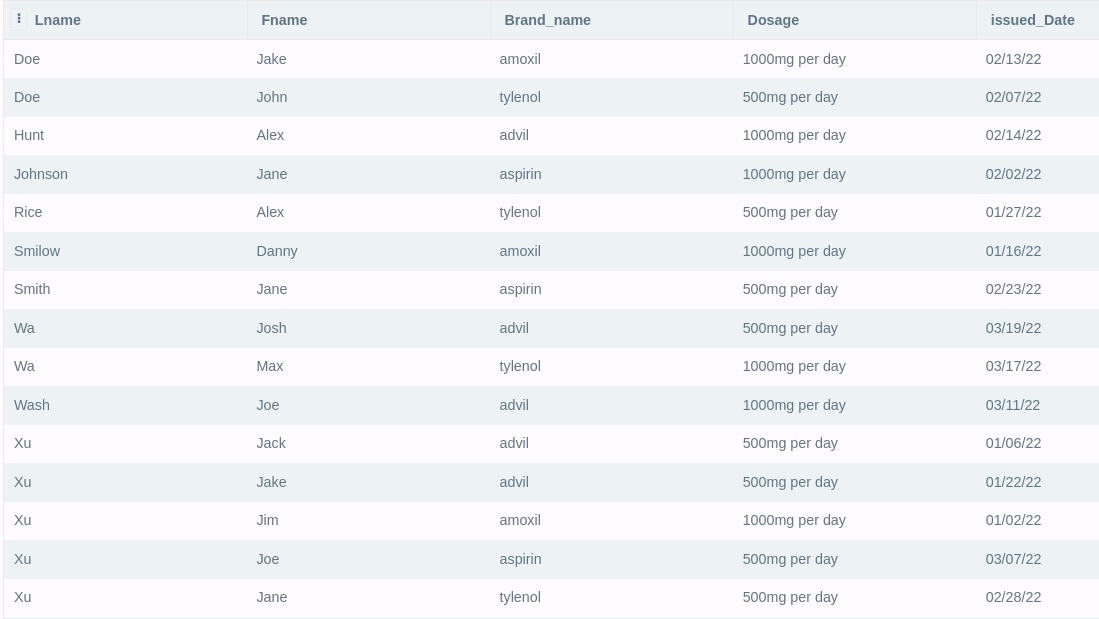
INNER JOIN PATIENT ON PATIENT.patient\_ID = PATIENT\_MEDRECORD.patient\_ID

INNER JOIN PRESCRIPTIONS ON PRESCRIPTIONS.medRecord\_ID = PATIENT\_MEDRECORD.medRecord\_ID

INNER JOIN PERSON ON PERSON.person\_ID = PATIENT.person\_ID

GROUP BY PERSON.LName, PERSON.FName

ORDER BY PERSON.Lname, PRESCRIPTIONS.Brand\_name;



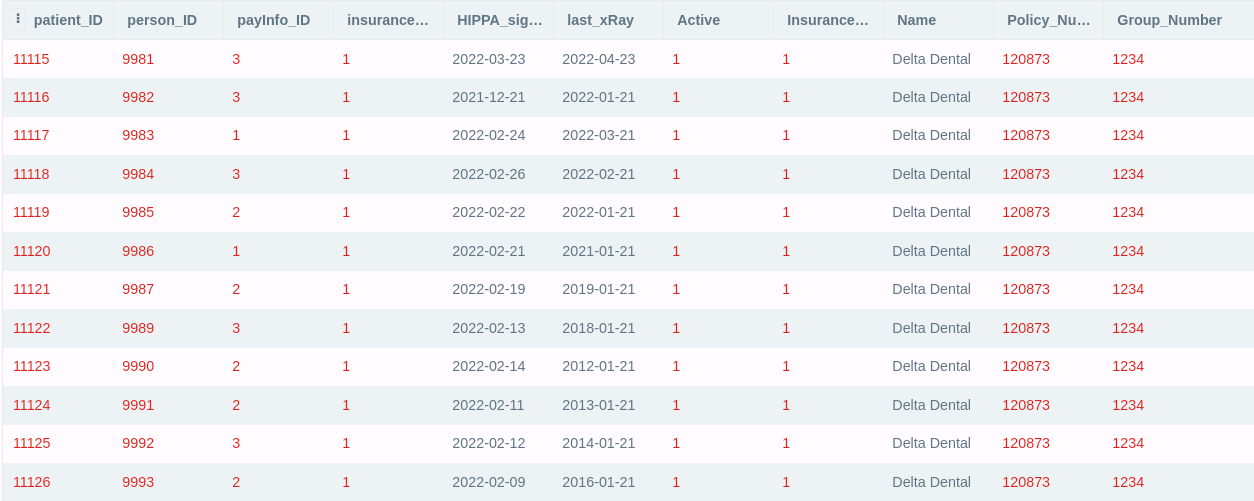
**Display patient information for patients who currently have Delta Dental**

**insurance policy.**

SELECT \* FROM PATIENT

INNER JOIN INSURANCE ON INSURANCE.insurance\_ID = PATIENT.insurance\_ID

WHERE INSURANCE.Name = 'Delta Dental';



**Generate a list of procedures and dates of service performed by doctor Smilow.**

SELECT PROCEDURES.procedurename,APPOINTMENT.appointment\_date FROM PROCEDURES

INNER JOIN INVOICE ON INVOICE.invoice\_ID = PROCEDURES.invoice\_ID

INNER JOIN APPOINTMENT\_INVOICES ON APPOINTMENT\_INVOICES.invoice\_id = INVOICE.invoice\_ID

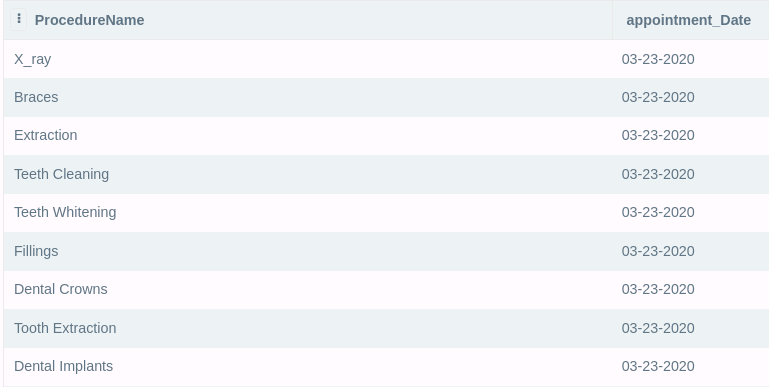
INNER JOIN APPOINTMENT ON APPOINTMENT.appointment\_id = APPOINTMENT\_INVOICES.appointment\_id

INNER JOIN MEDICAL ON MEDICAL.medical\_ID = PROCEDURES.medical\_ID

INNER JOIN EMPLOYEE ON EMPLOYEE.emp\_id = MEDICAL.emp\_id

INNER JOIN PERSON ON PERSON.person\_ID = EMPLOYEE.person\_ID

WHERE PERSON.Lname = 'Smilow';



**Print out a list of past due invoices with patient contact information. Past due is**

**defined as over 30 days old with a balance over $10.**

SELECT INVOICE.invoice\_ID, APPOINTMENT.appointment\_date, PERSON.Fname, PERSON.Lname FROM INVOICE

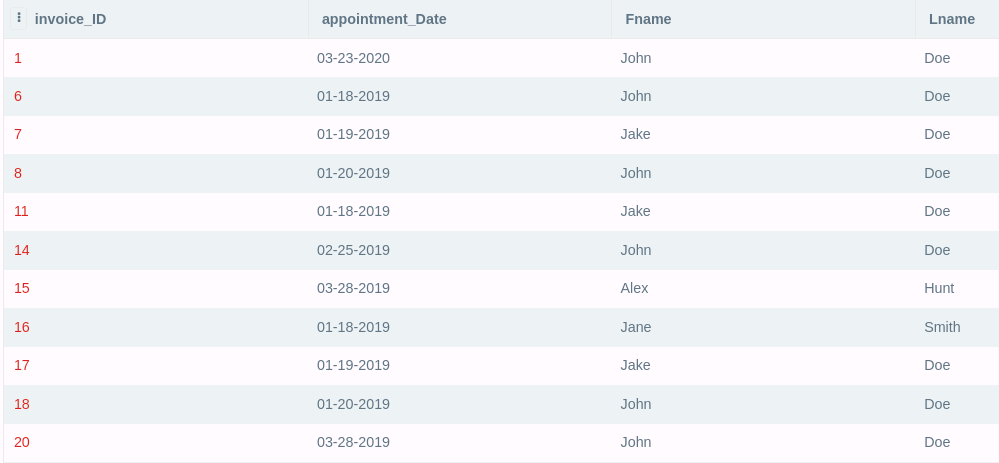
INNER JOIN PATIENT ON PATIENT.patient\_ID = INVOICE.patient\_ID

INNER JOIN APPOINTMENT\_INVOICES ON APPOINTMENT\_INVOICES.invoice\_id = INVOICE.invoice\_ID

INNER JOIN APPOINTMENT ON APPOINTMENT.appointment\_id = APPOINTMENT\_INVOICES.invoice\_id

INNER JOIN PERSON ON PERSON.person\_id = PATIENT.person\_ID

WHERE APPOINTMENT.appointment\_date < DateTime('Now', 'LocalTime', '-30 Day') AND (INVOICE.charge - INVOICE.paid > 10);



**Find the patients who brought the most revenue in the past year. You can define**

**how many records you want to display in the result of this query.**

SELECT PERSON.Fname, PERSON.Lname, INVOICE.paid FROM INVOICE

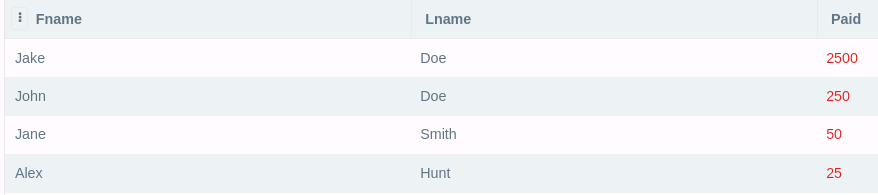
INNER JOIN PATIENT ON PATIENT.patient\_ID = INVOICE.patient\_ID

INNER JOIN PERSON ON PERSON.person\_id = PATIENT.person\_ID

GROUP BY PERSON.Fname, PERSON.Lname

ORDER BY INVOICE.paid DESC

LIMIT 10;



**Create a list of doctors who performed less than 5 procedures this year.**

SELECT M.medical\_ID, PERSON.fname, PERSON.lname FROM MEDICAL AS M

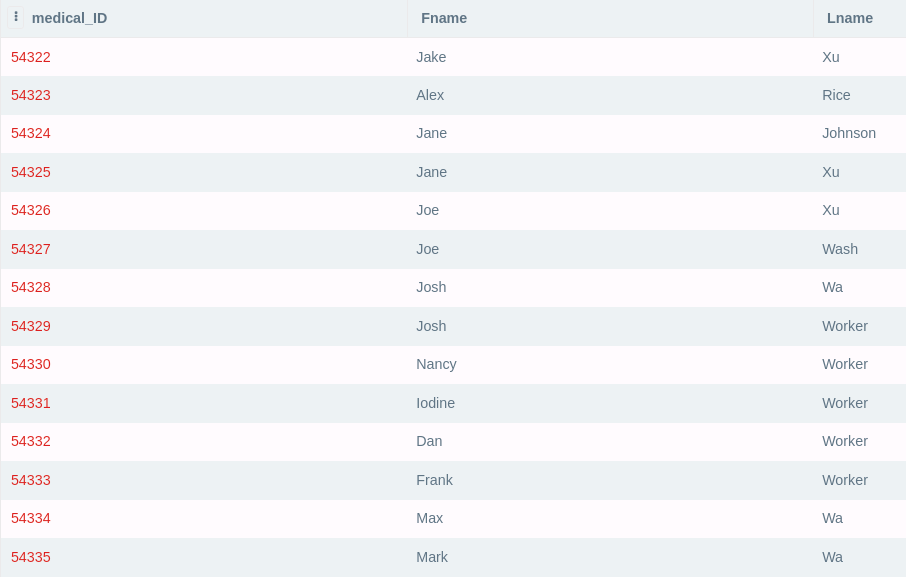
LEFT OUTER JOIN PROCEDURES AS P ON M.medical\_ID = P.medical\_ID

INNER JOIN EMPLOYEE ON EMPLOYEE.emp\_id = M.emp\_ID

INNER JOIN PERSON ON PERSON.person\_ID = EMPLOYEE.person\_id

GROUP BY M.medical\_ID

HAVING COUNT(P.procedure\_ID) < 5;



**Find the highest paying procedures, procedure price, and the total number of**

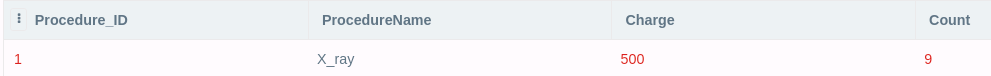
**those procedures performed. Sort your list with highest paying procedures**

**showing at the top of your list.**

SELECT P.procedure\_ID, P.procedureName,P.charge, COUNT(P.procedure\_ID) AS Count

FROM PROCEDURES AS P

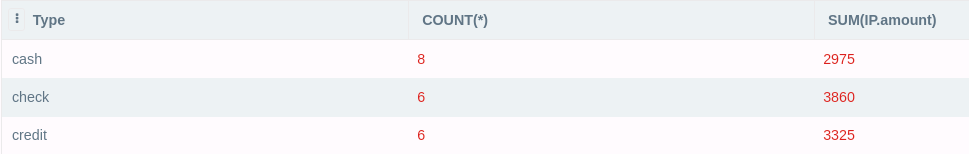
ORDER BY P.charge;



**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.**

SELECT PI.Type, COUNT(\*) , SUM(IP.amount) FROM INVOICE\_PAYMENT AS IP JOIN PAYMENT\_INFO AS PI ON IP.payinfo\_id = PI.payInfo\_ID GROUP BY PI.payinfo\_id;



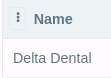
**Find the name of the most popular insurance plan currently used by the patients.**

SELECT I.Name FROM PATIENT AS P

INNER JOIN INSURANCE AS I ON I.insurance\_ID = P.insurance\_ID

GROUP BY (I.Name)

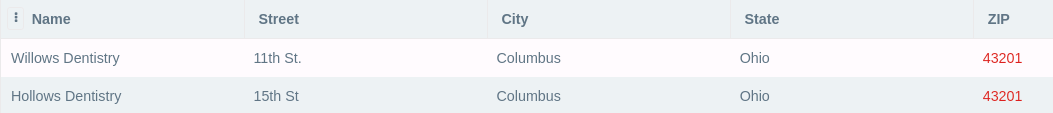
ORDER BY COUNT(I.Name) DESC LIMIT 1;



**List the names and addresses of all the dental offices**

SELECT LOCATION.Name, ADDRESS.Street, ADDRESS.City, ADDRESS.State, ADDRESS.Zip FROM LOCATION

INNER JOIN ADDRESS ON ADDRESS.addr\_ID = LOCATION.addr\_ID;



**Find the first and last name and phone number of the emergency contact for all patients.**

SELECT PERSON.Phone, PERSON.Fname, PERSON.Lname FROM PERSON

INNER JOIN EMERGENCY\_CONTACTS ON EMERGENCY\_CONTACTS.person\_ID = PERSON.person\_ID



**Find all invoices that are over $1,000 for a patient.**

SELECT \* FROM INVOICE

WHERE CHARGE > 1000



### **Insert and Delete SQL code samples**

In order to insert records into the database, INSERT statements must be run. Subsequently to delete records, DELETE statements must be run. Below are example of INSERT statements and the DELETE statements to remove the data entered by the INSERT statements. These queries will not provide an output, but will instead update the database which can then be seen by querying the respective tables.

**INSERT APPOINTMENTS AND THEIR RESPECTIVE INVOICES INTO THE DATABASE**

INSERT INTO APPOINTMENT

(appointment\_Date, patient\_ID, medical\_ID, Cancellation, Location\_ID)

VALUES

(DATE(), 11115, 2, 0, 1),

(DATE(), 11116, 2, 1, 1);

INSERT INTO INVOICE

(Charge, Paid, patient\_ID)

VALUES

(500, 250,11115),

(2500, 2500,11116);

INSERT INTO APPOINTMENT\_INVOICES

(Appointment\_ID, Invoice\_ID)

VALUES

(21, 21),

(22, 22);

**DELETE ALL APPOINTMENTS AND INVOICES FOR A PATIENT**

DELETE FROM APPOINTMENT

WHERE patient\_ID=11115;

DELETE FROM INVOICE

WHERE patient\_ID=11116;

**INSERT ADDRESSES AND OFFICE LOCATIONS INTO THE DATABASE**

INSERT INTO ADDRESS

(Street, Apartment, City, State, ZIP, Country)

VALUES

('11th St.', NULL,'Columbus', 'Ohio', 43201, 'United States'),

('15th St','255','Columbus','Ohio', 43201,'United States') ;

INSERT INTO LOCATION

(addr\_ID, Phone, Name)

VALUES

(3, '646-379-0000', 'Willow’s Dentistry')

(4, ‘123-456-7890’, ‘Willow’s Dentistry City Office’);

**DELETE ALL OFFICE LOCATIONS AND ADDRESSES WITH AN APARTMENT NUMBER**

DELETE FROM LOCATION

WHERE addr\_id IN (SELECT addr\_id FROM ADDRESS WHERE apartment IS NOT NULL);

DELETE FROM ADDRESS

WHERE apartment IS NOT NULL;

**INSERT PAYMENTS AND THEIR RESPECTIVE PAYMENT INFORMATION**

INSERT INTO PAYMENT\_INFO

(payInfo\_ID, Type)

VALUES

(4,'cash'),

(5,'check'),

(6,'credit'),

(7,'credit'),

(8,'check');

INSERT INTO CARD\_PAYMENT

(cardPaymentId,payInfo\_ID, card\_Number, cvv, expiration\_date, name\_on\_card, addr\_ID)

VALUES

(12,3, '378282246310005',123,DATE(),'VINCENT M ZHENG', 2),

(13,3, '371449635398431',321,DATE(),'DANIEL MOGLISKY', 2);

INSERT INTO CHECK\_PAYMENT (CheckID,payInfo\_ID,checkNumber,accountNumber,routingNumber)

VALUES

(11,2, 1111, 1111111111111, 111111111),

(12,5, 1112, 1111111111112, 111111112);

**DELETE ALL CHECK PAYMENTS FROM THE DB**

DELETE FROM PAYMENT\_INFO

WHERE Type=’check’;

DELETE FROM CHECK\_PAYMENT;

**Indexes**

Indexes can be used to improve the performance of our database. In our employee table, a hash index on the shift attribute would be highly beneficial. This would allow for quick access to all employees on a certain shift. Hash indexes are ideal for looking up values based on equality tests. In our medicalrecord\_diagnosis table, a B-tree index on diagnosisDate would be highly beneficial. This would allow for quick access to getting all diagnoses in a certain date range. B-tree index is best for looking up values based on range tests. The code for these indexes are below.

CREATE INDEX SHIFT\_HASH ON EMPLOYEE(Shift);

CREATE INDEX DIAGNOSIS\_DATE\_TREE ON MEDICALRECORD\_DIAGNOSIS(diagnosisDate);

**Views**

A view can be created to generate a revenue report for each person in our database. This view shows the name of each person with a related invoice. For each person, it will display the number of invoices they have, and how much they have paid in total on those invoices. A sample result from this view and the code for the view is shown below.

CREATE VIEW REVENUE\_REPORT AS SELECT PERSON.Fname AS 'First Name', PERSON.Lname AS 'Last Name', SUM(INVOICE.paid) AS Revenue, COUNT(\*) AS Invoices FROM INVOICE

INNER JOIN PATIENT ON PATIENT.patient\_ID = INVOICE.patient\_ID

INNER JOIN PERSON ON PERSON.person\_id = PATIENT.person\_ID

GROUP BY PERSON.Fname, PERSON.Lname

ORDER BY SUM(INVOICE.paid) DESC;

Result (First Name, Last Name, Revenue, Invoices):

(Jake, Doe, 2500, 1)

(John, Doe, 500, 2)

(Jane, Smith, 50, 1)

(Alex, Hunt, 25, 1)

A view can be created to get a report on how much each insurance is bringing in. This view shows the total amount paid, total amount charged, number of invoices, average paid, and average charged for each insurance. This gives a good idea of how much each insurance is making and how well their users are at paying their invoices. A sample result from this view and the code for the view is shown below.

CREATE VIEW INSURANCE\_REPORT AS SELECT INSURANCE.Name, COUNT(\*) AS Invoices, SUM(INVOICE.paid) AS 'Revenue Paid', AVG(INVOICE.Paid) AS 'Average Paid', SUM(INVOICE.charge) AS 'Revenue Charged', AVG(INVOICE.Charge) AS 'Average Charged' FROM INVOICE

INNER JOIN PATIENT ON PATIENT.patient\_ID = INVOICE.patient\_ID

INNER JOIN INSURANCE ON INSURANCE.insurance\_id = PATIENT.insurance\_id

GROUP BY INSURANCE.insurance\_id

ORDER BY SUM(INVOICE.paid) DESC;

Result (Name, Invoices, Revenue Paid, Average Paid, Revenue Charged, Average Charged):

(Delta Dental, 5, 3075, 615, 3325, 665)

(Sad Insurance, 1, 20, 20, 50, 50)

**TRANSACTIONS**

The two sample transactions I would establish is one for insertion of a new payment information using the check and paying for a bill payment and the other transaction is to add a new employee to our working roster with a schedule. It is crucial that we execute these transactions as one unit of processing to keep our database consistent so that information would not be lost or created by accident.

BEGIN TRANSACTION New\_Payment

INSERT OR ROLLBACK INTO CHECK\_PAYMENT (1,3,'378282246310005',123,DATE(),'VINCENT M ZHENG', 2);

UPDATE OR ROLL BACK INTO INVOICE\_PAYMENT

SET amount = 100;

SET date = 4/16/22;

WHERE payment\_ID = ‘1’;

COMMIT;

END TRANSACTION;

BEGIN TRANSACTION Add\_New\_Employee

INSERT OR ROLLBACK INTO PERSON

(9981, 284164870, 'John', 'Doe', 1112223333, 'White', 'M', 0, 0, 'a@b.com', '03-23-2000', 1);

INSERT OR ROLL BACK INTO EMPLOYEE  
(100, 9998, 43093, '7am-5pm', 1);

COMMIT;

END TRANSACTION;

**Section 3 - Team Reports and Graded Checkpoint Documents**

**Team Member Contributions**

Caden Rice - ERD Diagram, SQL Indices, SQL Views, SQL Queries, relational mapping, relational algebra, SQL insert statements

Vincent Zheng - Conducted research on dentistry databases, listed entity, attributes and relationships, mapped to EERD to a schema, relational algebra queries, applied normalization to check for 3NF and BCNF for all tables, SQL create and insert statements, and transactions

Daniel Mogilevsky - Project Manager, All SQL Files, Majority of SQL SELECT queries, all SQL DELETE queries, validation of all SQL create, select, insert, delete queries, majority of schema design decisions, introduction on final project, majority of relational algebra, assisted with ERD

Yoojin Jeong - Relationships between entities, possible queries, relational algebra, relational schema, mapped EERD to a schema, SQL create statements, SQL insert statement, database table description

**Reflection On Completion Progress**

Share your feedback on your teamwork dynamics, project timeframe, work schedules, project

development process, comments, and suggestions. What would you do differently to make this

process more efficient? Provide suggestions for future teams.

Overall completed projects early and worked largely asynchronously, dividing the work to different members. This made it harder to cooperate on the project. Considering the team was very busy this semester, we did well. Mid-checkpoint and final check meeting, greater communication during project checkpoints.

**Description of Feedback Received**

The first feedback we received was very positive with minor adjustments to the relational schema such as removing unnecessary specializations and attributes and adding some missed features.

We mostly got positive feedback for CP02. However, we had some minor issues from relational schema such as missing attributes, unnecessary specialization, wrong relationships. Also, foreign keys had to be specified.

We had a few issues for CP03. The ERD should not have lines crossing over each other. The SQL code was poorly formatted. It should have been in separate text files and not as a ZIP file. The code for the create and insert statements should be separated. Query A should be sorted by last name. Query d has no output. Query H was completely missing. Additional query a is missing 15th street.

**Previous Project Checkpoints**

**CP01:**

1. Team Members: Vincent Zheng, Yoojin Jeong, Daniel Mogilevsky, Caden Rice

We planned communication through discord. Setting up meetings when we are all available. There are some issues regarding available time to meet up but we worked through it. No issues so far.

1. We mostly went off the given requirements and prior knowledge, however, the following resources were used:

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3009547/#:~:text=The%20record%20may%20consist%20of,patient%20identification%20information%2C%20and%20a>

<https://vertabelo.com/blog/database-design-for-the-dental-office/>

<https://www.smilesofcary.com/general-dentistry-procedures>

Based on the research conducted the team gave the Patient entity attributes for storing demographic information. A subclass of employee was created, as it was discovered that there is a distinction between medical and non-medical employees and that medical employees may have more requirements and attributes than non-medical employees. Payments should be tied to invoices, which are related to specific appointments. This and the information given to the team in the requirements was enough to begin designing a database.

1. Two of the additional features included:

* Ability to specify a location for the dentist's office in case new locations show up. This involves a location entity, and employees and appointments are both tied to a location
* Ability to store prescriptions prescribed to patients by dentists. Using a prescription entity that is related to patients and doctors.

1. Additional Assumptions:
   * An appointment can only be at one location, and employees can work at only one location at a time
   * For each procedure and insurance plan combo, there is one unique charge to the patient
   * A patient may have many allergies and may receive many prescriptions
2. List of possible entities and their attributes

* Procedures - procedure\_ID, procedure\_Name, patient\_ID, ID,dentist\_ID, charge (drawn from Procedure\_Charges), required\_License (license\_ID)
* Insurance Plan - insurance\_ID, Name, Policy\_Number, Group\_Number

*Procedure\_Charges - insurance\_ID, procedure\_Name, chargeToPatient (Many-many relationship)*

* Appointment - appointment\_ID, date, patient\_ID, dentist\_ID, cancellation:boolean, Location\_ID, invoice\_ID
* Location - Location\_ID, addr\_ID, phone, Name
* Address - addr\_ID, Street, Apartment, City, State, ZIP, Country
* Invoice - invoice\_ID, payment\_ID, appointment\_ID, patient\_ID, charge
* Payment\_Info - payInfo\_ID, payment\_Type (cash, check, credit card), card\_number, cvv, expiration\_date, name\_on\_card, Billing\_Address (addr\_ID)
* Payment - payment\_ID, date, amount, payInfo\_ID
* Prescriptions - Prescription\_ID, Dentist\_ID, dosage, brand\_name, Patient\_ID
* Allergies - allergy\_ID, allergen\_name

*patient\_Allergies - allergy\_ID, patient\_ID, severity (Mild, Severe, Deadly) (Many to many relationship)*

* License - License\_ID, Name
* Person - SSN, name, phone, email, addr\_ID, birth\_date, race, sex (M, F), disability (boolean), veteran (boolean)
  + Patient - patient\_ID, SSN, active:boolean, payInfo\_ID, Dentist\_ID, insurance\_ID, HIPAA\_signed\_date (a date field, but it empty serves as true/false), last\_xRay, payInfo\_ID, eContact\_Name
  + Employee - Emp\_ID, SSN, Salary, shift, Location\_ID
    - Medical - medical\_ID, License\_ID, Licensure, emp\_ID
      * Dentist - Dentist\_ID, medical\_ID
      * Hygienists - Hygienist\_ID, medical\_ID
      * Dental assistants - Assistant\_ID, medical\_ID

1. The following are the relationships between our entities:

A patient makes payments.

A patient sees a dentist.

A patient has an insurance plan.

A patient has procedures done on them.

A patient makes an appointment.

During an appointment the patient sees a dentist.

The appointment is at a location

A location has an address

The appointment has an associate invoice

An invoice has payments and is associated with a patient and appointment

Dentist prescribe prescriptions to patients

A payment has payment info which has an address

A procedure is done by a dentist.

A procedure charge is based on the procedure and insurance plan.

A Person has an address.

A patient has allergies.

Appointment contains procedures.

Payment\_info describes payment.

Medical employees have a license.

Procedure includes procedure\_charges

Employees and patients are people

Non-medical and medical employees are employees

1. Here are four possible queries that can be done using our database:

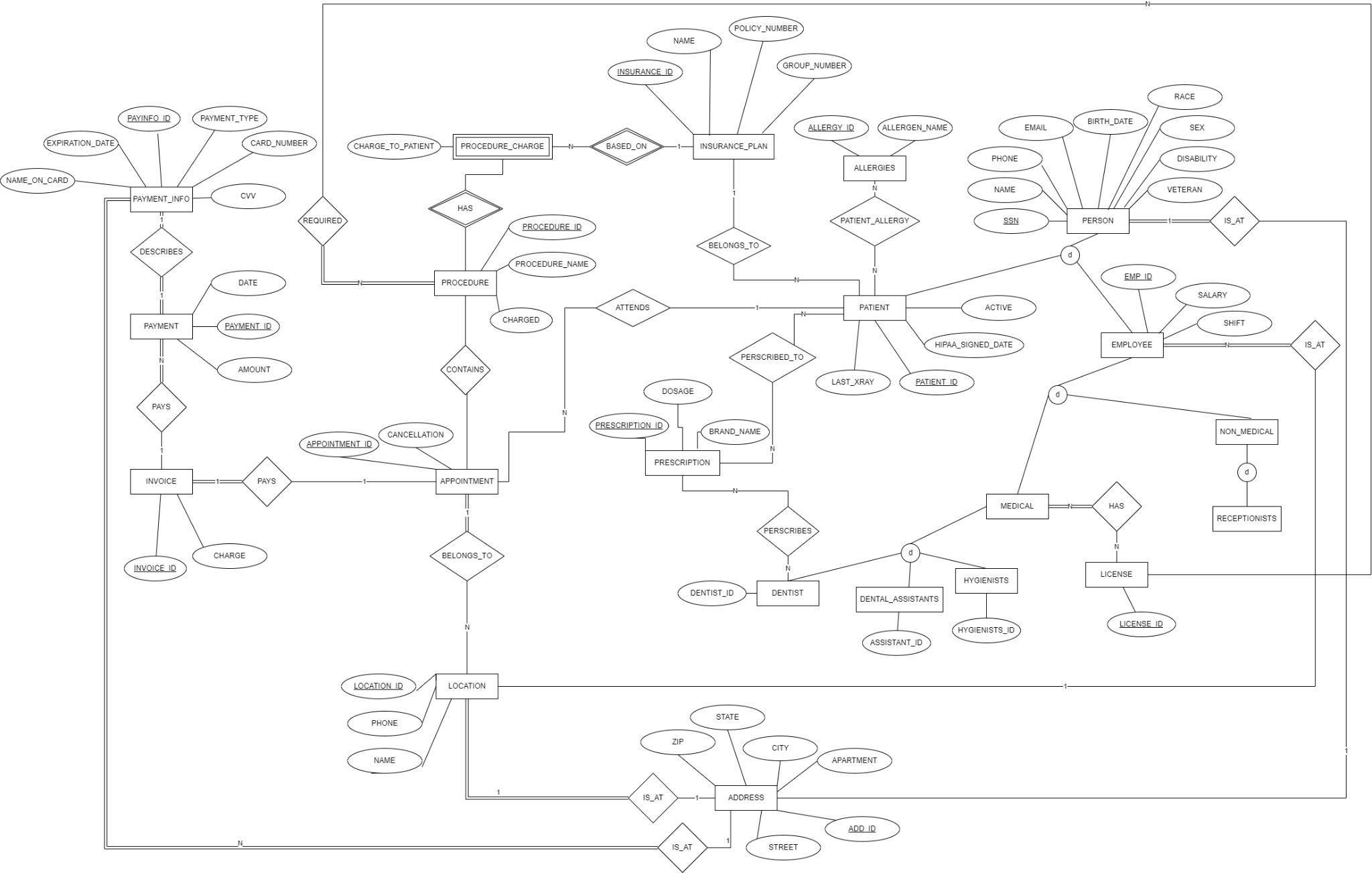
A dentist wants to see a patient's recent treatment.

A patient wants to know his/her next appointment date.

A dentist wants to know a patient’s insurance plan.

A dentist wants to check when a patient’s last visit was.

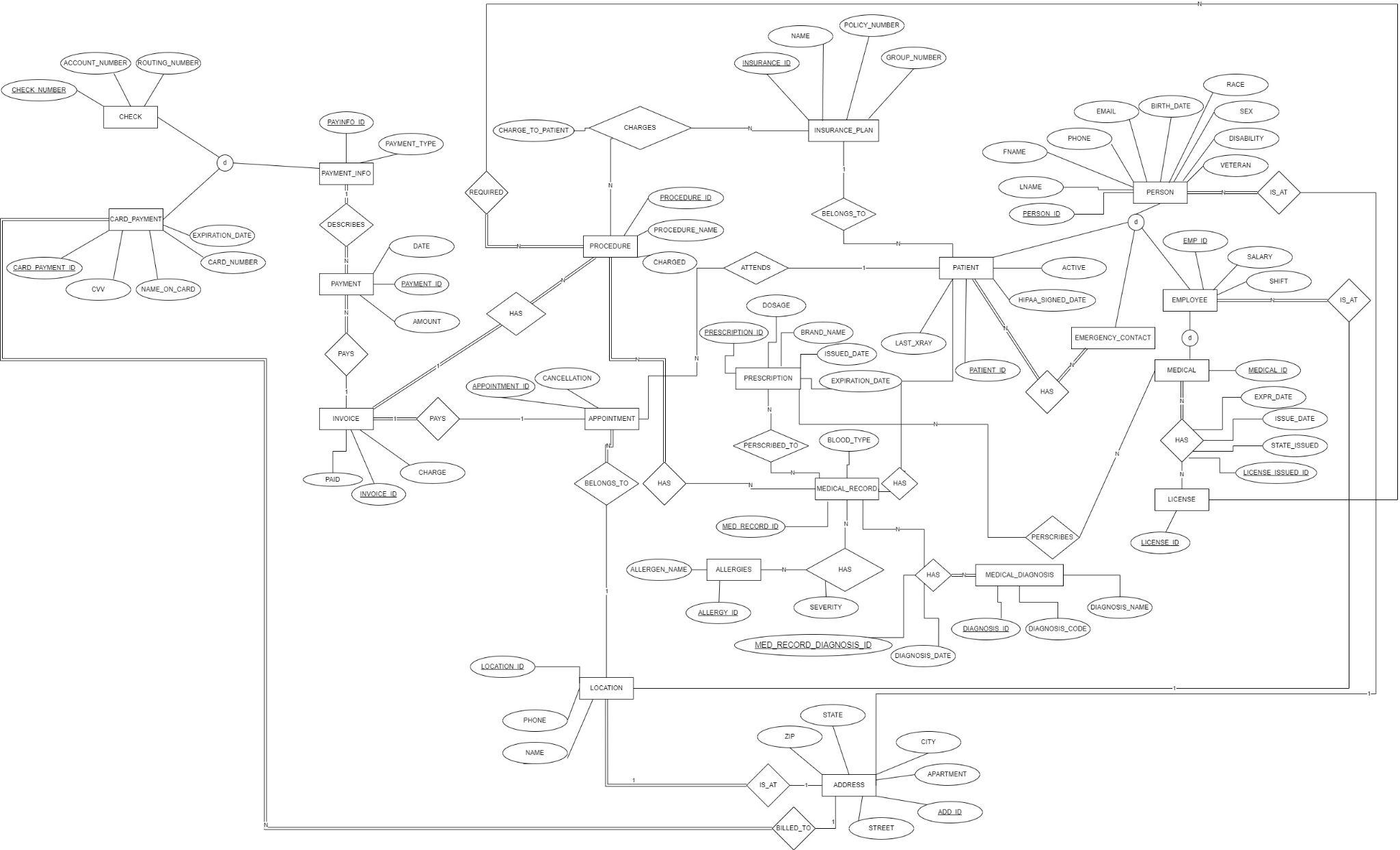
**CP02:**

Old ERD:

Feedback Received on CP01:

ERD looks great overall. With few small changes, you should be all set to work on your relational schema. Remove unnecessary specializations or add attributes to them as discussed in class today. Invoice needs to have a relationship with Procedure. Do not connect Procedure with Appointment. Some appointments may be cancelled. Payment methods not fully implemented. Consider specializations. They should be disjointed as each payment logically belongs to one category. You can apply more than one payment to the same invoice. Incorrect use of nested multipart attributes for payment types, consider Specializations instead. Not sure about Location entity. Is this in case this practice gets another office(location)? If so, connect with Address entity. You may need to store emergency contact(s) for a patient. Think about the best way to do this. Licensure info? Emergency Contacts, Medical history entity?

After receiving feedback we have made the following changes:



And from this improved ERD, we mapped the entities to tables.

Steps taken to ensure correctness:

We mapped each regular entity to a table, simple attributes mapped to an attribute relation, broken composite attributes into simple, and figured out a primary key

The weak entities were mapped using the Many-to-One relationships from our ERD and we used a foreign key for the primary relationship to the primary entity as well as creating a primary key for the weak entities.

We mapped 1:N relationships by adding a key from the 1 side as a foreign key to the entity on the N side.

Next we mapped the 1:1 relationships by creating a new cross reference as a new table of foreign keys to relate the two, merging the two tables together, or adding the partial key as a foreign key to the primary.

The mapping of M:N relationship we created a new relation and included the primary keys of the participating entities as foreign keys and to include any simple attributes

Mapping the multivalued attributes creates a new relation, if it is composite we include its simple components, and create a foreign key to its relationship or the entity it belongs to.

The mapping of more than 2 n-ary relations was not used in our ERD.

For specialization (only applied to Payment Info and Person, Patient, Employee, Medical Employee) , we relate the child entity to the parent by including foreign keys to the parent entity on the children. Finally, we did not map any union types as we did not use any. We Repeated these steps for all relationships listed.

* Procedures - procedure\_ID, procedure\_Name, medRecord\_ID, medical\_ID, charge (drawn from Procedure\_Charges), required\_License, invoice\_ID

| procedure\_ID (PK) | procedure\_Name | medRecord\_ID (FK) | medical\_ID (FK) | charge | required\_License\_Type | invoice\_ID (FK) |
| --- | --- | --- | --- | --- | --- | --- |

* Insurance Plan - insurance\_ID, Name, Policy\_Number, Group\_Number

| *insurance\_ID (PK)* | *Name* | *Policy\_Number* | *Group\_Number* |
| --- | --- | --- | --- |

*Procedure\_Charges - Procedure\_ChargeID, insurance\_ID, procedure\_Name, chargeToPatient (Many-many relationship)*

| *insurance\_ID (FK) (PK)* | *procedure\_Name (PK)* ***(FK)*** | *chargeToPatient* |
| --- | --- | --- |

*Appointment\_Invoices -* ***appointment\_ID****, I****nvoice\_ID (one to one)***

| ***apointment\_ID (PK) (FK)*** | ***invoice\_ID (PK) (FK)*** |
| --- | --- |

* Appointment - appointment\_ID, date, patient\_ID, medical\_ID, cancellation:boolean, Location\_ID

| appointment\_ID (PK) | date | patient\_ID (FK) | medical\_ID(FK) | cancellation | Location\_ID(FK) |
| --- | --- | --- | --- | --- | --- |

* Location - Location\_ID, addr\_ID, phone, Name

| Location\_ID (PK) | addr\_ID (FK) | Phone | Name |
| --- | --- | --- | --- |

* Address - addr\_ID, Street, Apartment, City, State, ZIP, Country

| addr\_ID(PK) | Street | Apartment | City | State | ZIP | Country |
| --- | --- | --- | --- | --- | --- | --- |

* Invoice - invoice\_ID, patient\_ID, charge, paid (boolean)

| invoice\_ID (PK) | charge | paid(boolean) | patient\_ID (FK) |
| --- | --- | --- | --- |

* Payment\_Info - payInfo\_ID, type(Cash, Card, Check)

| payInfo\_ID (PK) | type |
| --- | --- |

* + Card\_Payment - cardPaymentId, payInfo\_ID, card\_number, cvv, expiration\_date, name\_on\_card, addr\_ID

| cardPaymentId (PK) | payInfo\_ID(FK) | card\_number | cvv | expiration\_date | name\_on\_card | addr\_ID |
| --- | --- | --- | --- | --- | --- | --- |

* + Check - checkNumber, accountNumber, routingNumber, payInfo\_ID

| checkNumber (PK) | accountNumber | routingNumber | payInfo\_ID(FK) |
| --- | --- | --- | --- |

* Payment - payment\_ID, date, amount, payInfo\_ID, invoice\_ID

| payment\_ID(PK) | date | amount | payInfo\_ID (FK) | invoice\_ID(FK) |
| --- | --- | --- | --- | --- |

* Prescriptions - Prescription\_ID, Medical\_ID, dosage, brand\_name, medRecord\_ID, issued\_Date, expiration\_Date

| prescription\_ID (PK) | Medicial\_ID(FK) | dosage | brand\_name | medRecord\_ID(FK) | issued\_Date | expiration\_Date |
| --- | --- | --- | --- | --- | --- | --- |

* Allergies - allergy\_ID, allergen\_name

| allergy\_ID (PK) | allergen\_name |
| --- | --- |

* License - License\_ID, Type, Licensure\_Name

| License\_ID (PK) | type | Licensure\_Name |
| --- | --- | --- |

*medical\_Licenses - medical\_ID, License\_ID, Issue\_Date, Exp\_Date, stateIssued (Ohio, Virginia) (Many to many)*

| licenseIssued\_ID(PK) | medical\_ID(FK) | License\_ID(FK) | Issue\_Date | Exp\_Date | stateIssued |
| --- | --- | --- | --- | --- | --- |

* Medical\_Record - **medRecord\_ID**, blood type

| medRecord\_ID(PK) | Blood type |
| --- | --- |

* Medical\_Diagnosis - diagnosis\_ID, diagnosisCode, diagnosisName

| diagnosis\_ID(PK) | diagnosisCode | diagnosisName |
| --- | --- | --- |

*medicalRecord\_Allergies - allergy\_ID, medRecord\_ID, severity (Mild, Severe, Deadly) (Many to many relationship)*

| allergy\_ID(PK) (FK) | medRecord\_ID(PK) (FK) | severity |
| --- | --- | --- |

*medicalRecord\_Diagnoses - medRecord\_ID, diagnosisID (Many to many relationship)*

| *medRecord\_DiagnosisID (PK)* | *medRecord\_ID* (FK) | *diagnosisID* (FK) | *diagnosisDate* |
| --- | --- | --- | --- |

*patient\_medRecord - patientID, medRecord\_ID (one to one relationship)*

| *patientID (PK)* (FK) | *medRecord\_ID (PK)*(FK) |
| --- | --- |

* Person - personId, SSN, name, phone, email, addr\_ID, birth\_date, race, sex (M, F), disability (boolean), veteran (boolean)

| person\_ID(PK) | SSN | fname | lname | phone | race | sex | disability | veteran | addr\_ID(FK) | email | birth\_date |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

* + Patient - patient\_ID, person\_ID, active:boolean, payInfo\_ID, medical\_ID

insurance\_ID, HIPAA\_signed\_date (a date field, but it empty serves as true/false), last\_xRay

| patient\_ID(PK) | person\_ID(FK) | active | payInfo\_ID(FK) | medical\_ID(FK) | insurance\_ID(FK) | HIPAA\_signed\_date | last\_xRay |
| --- | --- | --- | --- | --- | --- | --- | --- |

* + Emergency\_Contacts - personID, patient\_ID (many to many)

| personID (PK) | patient\_ID (PK) |
| --- | --- |

* + Employee - Emp\_ID, personID, Salary, shift, Location\_ID

| Emp\_ID(PK) | person\_ID(FK) | Salary | Shift | Location\_ID(FK) |
| --- | --- | --- | --- | --- |

* + - Medical - medical\_ID, emp\_ID

| medical\_ID (PK) | emp\_ID (FK) |
| --- | --- |

With our database design revised and mapped to a relational schema, we can now map out the following queries in relational algebra:

a. Create a list of patients and the medications they currently take

1. PATIENTS\_WITH\_MED\_ID ← PATIENT ⨝ Patient.patientlID= patient\_medRecord.patientID (patient\_medRecord)
2. PATIENTS\_WITH\_MED\_RECORDS ← PATIENT\_WITH\_MED\_ID ⨝ PATIENT\_WITH\_MED\_ID.medRecordlID= medical\_Record.medRecordtID (medical\_Record)
3. PATIENTS\_WITH\_PRESCRIPTIONS ← PATIENTS\_WITH\_MED\_RECORDS ⨝ PATIENT\_WITH\_MED\_RECORD..medRecordlID= Prescriptions.medRecordtID AND Presciptions.Active = ‘true’ (Prescriptions)
4. PEOPLE\_WITH\_PRESCRIPTIONS ← PATIENTS\_WITH\_PRESCRIPTIONS ⨝ PATIENT\_WITH\_PRESCRIPTIONS.personID= Person.personID (Person)
5. **patientID,fname,lname****F** brand\_name,dosage(PEOPLE\_WITH\_PRESCRIPTIONS)

b. Display patient information for patients who currently have Delta Dental insurance

policy.

1. PATIENT ⨝ Patient.insuranceID= INSURANCE\_PLAN.insuranceID σ(Name=’Delta Dental’)(INSURANCE\_PLAN)

c. Generate a list of procedures and dates of service performed by doctor Smilow.

1. SMILOW\_PROCEDURES ← PROCEDURE ⨝ Procedure.medicalID= Medical.medicalID σ(fname=’Smilow’)(Medical)
2. INVOICE\_APPOINTMENTS ← INVOICE ⨝ Invoice.invoicelID= Appointment\_Invoices.invoiceID (Appointment\_Invoices)
3. INVOICE\_WITH\_DATES ←INVOICE\_APPOINTMENTS ⨝ INVOICE\_APPOINTMENTS.appointmentlID= Appointment.appointmentID (Appointment)
4. SMILOW\_PROCEDURES\_WITH\_DATES ← SMILOW\_PROCEDURES ⨝ SMILOW\_PROCEDURES.invoicelID= INVOICE\_WITH\_DATES.invoiceID (INVOICE\_WITH\_DATES)
5. **πprocedure\_Name, date** (SMILOW\_PROCEDURES\_WITH\_DATES)

d. Print out a list of past due invoices with patient contact information. Past due is

defined as over 30 days old with a balance over $10.

PAST\_DUE ← σ paid = false (σ charge > 10 (σ TODAY() - invoice.date > 30 (INVOICE)))

PATIENT\_PAYMENT ← PATIENT ⨝ patient.payInfo\_ID = payment.payInfo\_ID (PAYMENT)

PATIENT\_PAST\_INVOICE ← PATIENT\_PAYMENT ⨝ patient\_payment.invoiceID = past\_due.invoiceID (PAST\_DUE)

PERSON\_WITH\_PAST\_INVOICE ← PATIENT\_PAST\_INVOICE ⨝ patient.personID = person.personID (PERSON)

π phone, email (PERSON\_WITH\_PAST\_INVOICE)

e. Find the patients who brought the most revenue in the past year.

INVOICE\_APPOINTMENT ← INVOICE ⨝ invoice.patient\_ID = appointment.patient\_ID (APPOINTMENT)

INVOICE\_APPOINTMENT\_PATIENT ← INVOICE\_APPOINTMENT ⨝ invoice\_appointment.patient\_ID = patient.patient\_ID (PATIENT)

INVOICE\_APPINTMENT\_PATIENT\_PAYMENT ← INVOICE\_APPOINTMENT\_PATIENT ⨝ invoice\_appointment\_patient.invoice\_ID = payment.invoice\_ID (PAYMENT)

PATIENT\_CHARGE ← π patient\_ID, charge (INVOICE\_APPINTMENT\_PATIENT\_PAYMENT)

patient\_ID F SUM charge (PATIENT\_CHARGE)

f. Create a list of doctors who performed less than 5 procedures this year.   
DOCTORS\_NUM\_PROCEDURES = p MedID, Procedure\_Count (Medical\_Id,FCount(Procedure\_id)(PROCEDURE))  
DOCTORS\_LESS = σ(Procedure\_Count < 5)(MEDICAL)  
RESULT = ΠFName, Lname, Medical\_ID (DOCTORS\_LESS ⨝ MedID = Medical\_ID (MEDICAL))

g. Find the highest paying procedures, procedure price, and the total number of those

procedures performed.

MAX\_CHARGE\_AMOUNT = FMAX(Charge)(PROCEDURE))

PROCEDURE\_INVOICES =σ(Charge = MAX\_CHARGE\_AMOUNT)(PROCEDURE)   
PROCEDURE\_INVOICES\_COUNTS = p Procedure\_Name, Procedure\_Count (Procedure\_Name, Charge, FCount(Procedure\_id)(PROCEDURE\_INVOICES))

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.

TYPE ← π type (PAYMENT\_INFO)

PAYMENTS\_WITH\_TYPE ← PAYMENT ⨝ payment.paymentInfoID = PAYMENT\_INFO.payInfo\_ID (PAYMENTINFO)

type F COUNT, SUM amount (PAYMENTS\_WITH\_TYPE)

i. Find the name of the most popular insurance plan currently used by the patients.

PATIENT\_INSURANCE ← PATIENT ⨝ patient.insurance\_ID = insurancePlan.insurance\_ID (INSURANCE)

INSURANCE\_COUNT ← name F COUNT (PATIENT\_INSURANCE)

F MAX COUNT (INSURANCE\_COUNT)

Additional Queries

In addition to these queries, the following were drafted:

1. List the names and addresses of all the dental offices

LOCATIONS\_WITH\_ADDRESS ← LOCATION ⨝ Location.addr\_ID= Adress.addr\_ID (ADDRESS)

ΠName, Street, City, State, ZIP, Country(LOCATIONS\_WITH\_ADDRESS)

2. Find the first and last name and phone number of the emergency contact for all patients.

EMERGENCY\_CONTACTS\_WITH\_INFO← ( PERSON ⨝person\_ID = person\_ID EMERGENCY\_CONTACTS)

Πphone,fname,lname (EMERGENCY\_CONTACTS\_WITH\_INFO)

3. Find all invoices that are over $10,000 for a patient.

INFO ← ( PATIENT ⨝patient\_ID = appointment.patient\_ID APPOINTMENT)

CHARGE ← ( INVOICE ⨝ Invoice.invoicelD= Info\_Invoices.invoiceID INFO)

Πcharge > 10,000 (CHARGE)

CP03:

After once again receiving feedback the following relational schema was constructed:

* Procedures - procedure\_ID, procedure\_Name, medRecord\_ID, medical\_ID, charge (drawn from Procedure\_Charges), required\_License, invoice\_ID

| procedure\_ID (PK) | procedure\_Name | medRecord\_ID (FK) | medical\_ID (FK) | charge | required\_License\_Type | invoice\_ID (FK) |
| --- | --- | --- | --- | --- | --- | --- |

* Insurance Plan - insurance\_ID, Name, Policy\_Number, Group\_Number

| *insurance\_ID (PK)* | *Name* | *Policy\_Number* | *Group\_Number* |
| --- | --- | --- | --- |

*Procedure\_Charges - Procedure\_ChargeID, insurance\_ID, procedure\_Name, chargeToPatient (Many-many relationship)*

| *insurance\_ID (FK) (PK)* | *procedure\_Name (PK)* | *chargeToPatient* |
| --- | --- | --- |

*Appointment\_Invoices -* ***appointment\_ID****, I****nvoice\_ID (one to one)***

| ***apointment\_ID (PK)*** | ***invoice\_ID (PK)*** |
| --- | --- |

* Appointment - appointment\_ID, date, patient\_ID, medical\_ID, cancellation:boolean, Location\_ID

| appointment\_ID (PK) | date | patient\_ID (FK) | medical\_ID(FK) | cancellation | Location\_ID(FK) |
| --- | --- | --- | --- | --- | --- |

* Location - Location\_ID, addr\_ID, phone, Name

| Location\_ID (PK) | addr\_ID (FK) | Phone | Name |
| --- | --- | --- | --- |

* Address - addr\_ID, Street, Apartment, City, State, ZIP, Country

| addr\_ID(PK) | Street | Apartment | City | State | ZIP | Country |
| --- | --- | --- | --- | --- | --- | --- |

* Invoice - invoice\_ID, patient\_ID, charge, paid (boolean)

| invoice\_ID (PK) | charge | paid(boolean) | patient\_ID (FK) |
| --- | --- | --- | --- |

* Payment\_Info - payInfo\_ID, type(Cash, Card, Check)

| payInfo\_ID (PK) | type |
| --- | --- |

* + Card\_Payment - cardPaymentId, payInfo\_ID, card\_number, cvv, expiration\_date, name\_on\_card, addr\_ID

| cardPaymentId (PK) | payInfo\_ID(FK) | card\_number | cvv | expiration\_date | name\_on\_card | addr\_ID |
| --- | --- | --- | --- | --- | --- | --- |

* + Check - checkNumber, accountNumber, routingNumber, payInfo\_ID

| CheckID (PK) | checkNumber | accountNumber | routingNumber | payInfo\_ID(FK) |
| --- | --- | --- | --- | --- |

* InvoicePayment - payment\_ID, date, amount, payInfo\_ID, invoice\_ID

| payment\_ID(PK) | date | amount | payInfo\_ID (FK) | invoice\_ID(FK) |
| --- | --- | --- | --- | --- |

* Prescriptions - Prescription\_ID, Medical\_ID, dosage, brand\_name, medRecord\_ID, issued\_Date, expiration\_Date

| prescription\_ID (PK) | Medicial\_ID(FK) | dosage | brand\_name | medRecord\_ID(FK) | issued\_Date | expiration\_Date |
| --- | --- | --- | --- | --- | --- | --- |

* Allergies - allergy\_ID, allergen\_name

| allergy\_ID (PK) | allergen\_name |
| --- | --- |

* License - License\_ID, Type, Licensure\_Name

| License\_ID (PK) | type | Licensure\_Name |
| --- | --- | --- |

*medical\_Licenses - medical\_ID, License\_ID, Issue\_Date, Exp\_Date, stateIssued (Ohio, Virginia) (Many to many)*

| licenseIssued\_ID(PK) | medical\_ID(FK) | License\_ID(FK) | Issue\_Date | Exp\_Date | stateIssued |
| --- | --- | --- | --- | --- | --- |

* Medical\_Record - **medRecord\_ID**, blood type

| medRecord\_ID(PK) | Blood type |
| --- | --- |

* Medical\_Diagnosis - diagnosis\_ID, diagnosisCode, diagnosisName

| diagnosis\_ID(PK) | diagnosisCode | diagnosisName |
| --- | --- | --- |

*medicalRecord\_Allergies - allergy\_ID, medRecord\_ID, severity (Mild, Severe, Deadly) (Many to many relationship)*

| allergy\_ID(PK) | medRecord\_ID(PK) | severity |
| --- | --- | --- |

*medicalRecord\_Diagnoses - medRecord\_ID, diagnosisID (Many to many relationship)*

| *medRecord\_DiagnosisID (PK)* | *medRecord\_ID* | *diagnosisID* | *diagnosisDate* |
| --- | --- | --- | --- |

*patient\_medRecord - patientID, medRecord\_ID (one to one relationship)*

| *patientID (PK)* | *medRecord\_ID (PK)* |
| --- | --- |

* Person - personId, SSN, name, phone, email, addr\_ID, birth\_date, race, sex (M, F), disability (boolean), veteran (boolean)

| person\_ID(PK) | SSN | fname | lname | phone | race | sex | disability | veteran | addr\_ID(FK) | email | birth\_date |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

* + Patient - patient\_ID, person\_ID, active:boolean, payInfo\_ID, medical\_ID

insurance\_ID, HIPAA\_signed\_date (a date field, but it empty serves as true/false), last\_xRay

| patient\_ID(PK) | person\_ID(FK) | active | payInfo\_ID(FK) | medical\_ID(FK) | insurance\_ID(FK) | HIPAA\_signed\_date | last\_xRay |
| --- | --- | --- | --- | --- | --- | --- | --- |

* + Emergency\_Contacts - personID, patient\_ID (many to many)

| personID (PK) | patient\_ID (PK) |
| --- | --- |

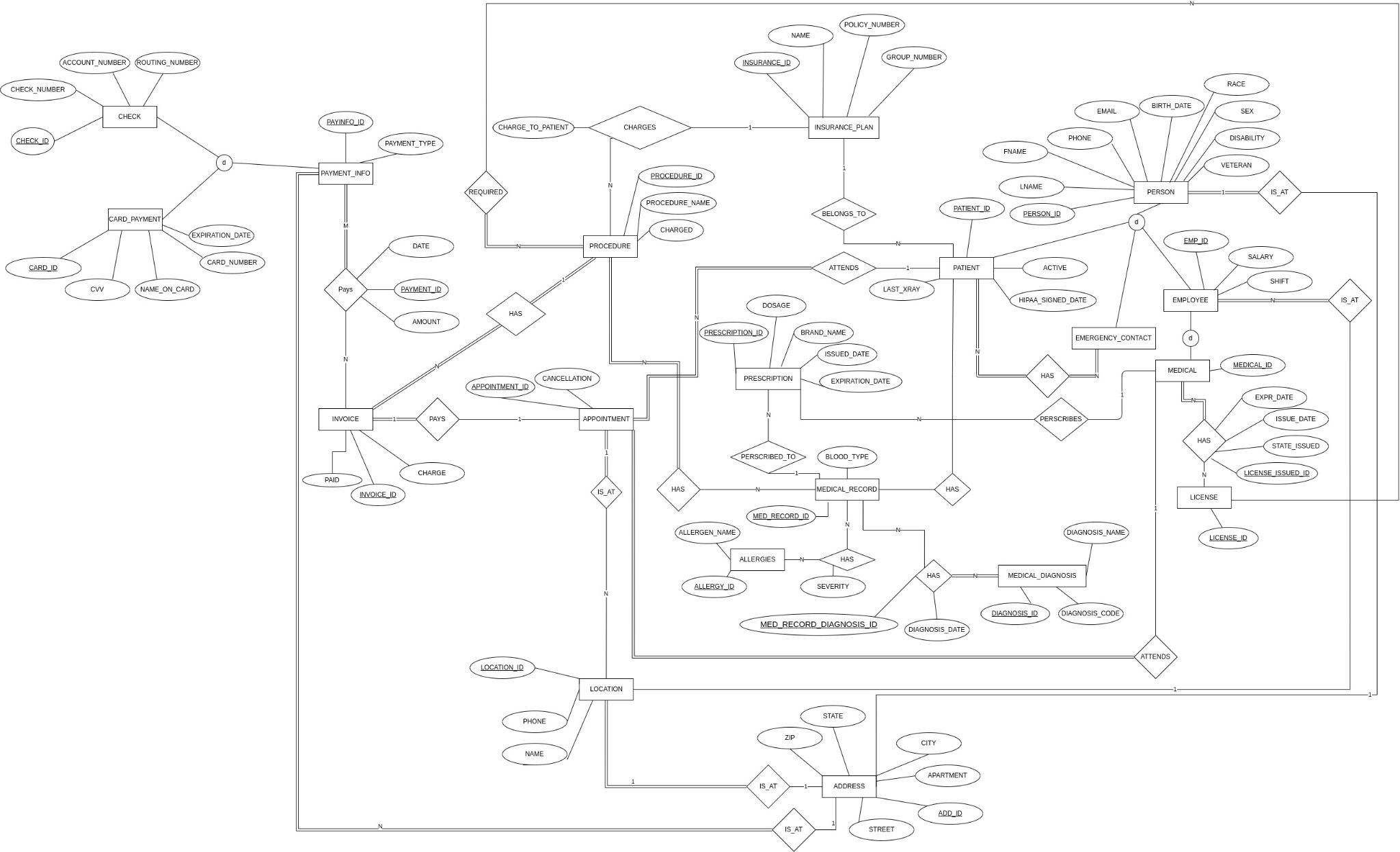
* + Employee - Emp\_ID, personID, Salary, shift, Location\_ID

| Emp\_ID(PK) | person\_ID(FK) | Salary | Shift | Location\_ID(FK) |
| --- | --- | --- | --- | --- |

* + - Medical - medical\_ID, emp\_ID

| medical\_ID (PK) | emp\_ID (FK) |
| --- | --- |

And the ERD was remade:



The following SQL script was written to generate the database:

CREATE TABLE Procedures (

Procedure\_ID INT NOT NULL PRIMARY KEY,

ProcedureName VARCHAR(50),

MedRecord\_ID INT NOT NULL,

Medical\_ID INT NOT NULL,

Charge INT,

Required\_License\_Type VARCHAR(50),

Invoice\_ID INT NOT NULL,

FOREIGN KEY(MedRecord\_ID) REFERENCES medRecord(MedRecord\_ID),

FOREIGN KEY(Medical\_ID) REFERENCES Medical(med\_ID),

FOREIGN KEY(Invoice\_ID) REFERENCES Invoice(Invoice\_ID)

);

INSERT INTO PROCEDURES

(Procedure\_ID, ProcedureName, MedRecord\_ID,Medical\_ID,Charge,Required\_License\_Type,Invoice\_ID)

VALUES

(1,'X\_ray',1,54321,500,'Doctor',1),

(2,'Braces',1,54321,2500,'Doctor',1),

(3,'Extraction',1,54321,250,'Doctor',1),

(4,'Teeth Cleaning',1,1,25,'Doctor',1),

(5,'Teeth Whitening',1,1,50,'Doctor',1)

;

CREATE TABLE INSURANCE (

Insurance\_ID INTEGER NOT NULL PRIMARY KEY,

Name VARCHAR(255) NOT NULL,

Policy\_Number INT NOT NULL,

Group\_Number INT NOT NULL

);

INSERT INTO INSURANCE

(Name, Policy\_Number, Group\_Number)

VALUES

('Delta Dental',120873,1234),

('Sad Insurance',987322,234)

;

CREATE TABLE PROCEDURE\_CHARGES(

Insurance\_ID INTEGER NOT NULL,

ProcedureName Text NOT NULL,

ChargeToPatient INT NOT NULL,

PRIMARY KEY(Insurance\_ID , ProcedureName),

FOREIGN KEY (Insurance\_ID) REFERENCES Insurance(Insurance\_ID),

FOREIGN KEY (ProcedureName) REFERENCES Procedure(ProcedureName)

);

INSERT INTO PROCEDURE\_CHARGES

(Insurance\_ID,ProcedureName, ChargeToPatient)

VALUES

(1, 'X\_ray',500),

(1, 'Braces',2500),

(2,'Extraction',250),

(2,'Teeth Cleaning',25),

(2,'Teeth Whitening', 50)

;

CREATE TABLE APPOINTMENT\_INVOICES (

Appointment\_ID INT NOT NULL,

Invoice\_ID INT NOT NULL,

PRIMARY KEY(appointment\_ID, Invoice\_ID),

FOREIGN KEY(appointment\_ID) REFERENCES Appointment(appointment\_ID),

FOREIGN KEY(Invoice\_ID ) REFERENCES Invoice(Invoice\_ID)

);

INSERT INTO APPOINTMENT\_INVOICES

(Appointment\_ID, Invoice\_ID)

VALUES

(1, 1),

(1, 2),

(1, 3),

(2, 4),

(2, 5)

;

CREATE TABLE APPOINTMENT (

appointment\_ID INTEGER PRIMARY KEY AUTOINCREMENT,

appointment\_Date DATE NOT NULL,

patient\_ID INT NOT NULL,

medical\_ID INT NOT NULL,

Cancellation BOOLEAN,

Location\_ID VARCHAR(255) NOT NULL,

FOREIGN KEY(Location\_ID) REFERENCES Location(Location\_ID),

FOREIGN KEY(patient\_ID) REFERENCES Patient(patient\_ID),

FOREIGN KEY(medical\_ID) REFERENCES Medical(medical\_ID)

);

INSERT INTO APPOINTMENT

(appointment\_Date, patient\_ID, medical\_ID, Cancellation, Location\_ID)

VALUES

(DATE(), 1, 2, 0, 1),

(DATE(), 1, 2, 1, 1)

;

CREATE TABLE LOCATION (

Location\_ID INTEGER PRIMARY KEY AUTOINCREMENT,

addr\_ID INT NOT NULL,

Phone VARCHAR(12) NOT NULL,

Name VARCHAR(255) NOT NULL,

FOREIGN KEY(addr\_ID) REFERENCES Address(addr\_ID)

);

INSERT INTO LOCATION

(addr\_ID, Phone, Name)

VALUES

(1, '646-379-0000', 'Willow’s Dentistry')

;

CREATE TABLE ADDRESS (

addr\_ID INTEGER PRIMARY KEY AUTOINCREMENT,

Street VARCHAR(255) NOT NULL,

Apartment VARCHAR(255),

City VARCHAR(255) NOT NULL,

State VARCHAR(255) NOT NULL,

ZIP VARCHAR(255) NOT NULL,

Country VARCHAR(255) NOT NULL

);

INSERT INTO ADDRESS

(Street, Apartment, City, State, ZIP, Country)

VALUES

('11th St.', NULL,'Columbus', 'Ohio', 43201, 'United States'),

('15th St','255','Columbus','Ohio', 43201,'United States')

;

CREATE TABLE INVOICE (

invoice\_ID INTEGER PRIMARY KEY AUTOINCREMENT,

Charge INT NOT NULL,

Paid int,

patient\_ID INT NOT NULL,

FOREIGN KEY(patient\_ID) REFERENCES Patient(patient\_ID)

);

INSERT INTO INVOICE

(Charge, Paid, patient\_ID)

VALUES

(500, 250,11115),

(2500, 2500,11116),

(250, 250,11115),

(25, 25,11117),

(50, 50,11118)

;

CREATE TABLE PAYMENT\_INFO (

payInfo\_ID INT NOT NULL PRIMARY KEY,

Type VARCHAR(10) NOT NULL

);

INSERT INTO PAYMENT\_INFO

(payInfo\_ID, Type)

VALUES

(1,'cash'),

(2,'check'),

(3,'credit'),

(4,'credit'),

(5,'check')

;

CREATE TABLE CARD\_PAYMENT (

cardPaymentId INT NOT NULL PRIMARY KEY,

payInfo\_ID INT NOT NULL,

card\_Number VARCHAR(16) NOT NULL,

cvv INT NOT NULL,

expiration\_date DATE NOT NULL,

name\_on\_card VARCHAR(30) NOT NULL,

addr\_ID INT NOT NULL,

FOREIGN KEY(payInfo\_ID) REFERENCES Payment\_Info(payInfo\_ID),

FOREIGN KEY(addr\_ID) REFERENCES Address(addr\_ID)

);

INSERT INTO CARD\_PAYMENT

(cardPaymentId,payInfo\_ID, card\_Number, cvv, expiration\_date, name\_on\_card, addr\_ID)

VALUES

(1,3, '378282246310005',123,DATE(),'VINCENT M ZHENG', 2),

(2,3, '371449635398431',321,DATE(),'DANIEL MOGLISKY', 2)

;

CREATE TABLE CHECK\_PAYMENT (

CheckID INT NOT NULL PRIMARY KEY,

payInfo\_ID INT NOT NULL,

checkNumber INT NOT NULL,

accountNumber INT NOT NULL,

routingNumber INT NOT NULL,

FOREIGN KEY(payInfo\_ID) REFERENCES Payment\_Info(payInfo\_ID)

);

INSERT INTO CHECK\_PAYMENT (CheckID,payInfo\_ID,checkNumber,accountNumber,routingNumber)

VALUES

(1,2, 1111, 1111111111111, 111111111),

(2,5, 1112, 1111111111112, 111111112)

;

CREATE TABLE INVOICE\_PAYMENT (

payment\_ID INT NOT NULL PRIMARY KEY,

payInfo\_ID INT NOT NULL,

invoice\_ID INT NOT NULL,

paymentDate DATE NOT NULL,

Amount INT NOT NULL,

FOREIGN KEY(payInfo\_ID) REFERENCES Payment\_Info(payInfo\_ID),

FOREIGN KEY(invoice\_ID) REFERENCES Invoice(invoice\_ID)

);

INSERT INTO INVOICE\_PAYMENT

VALUES

(1,3, 1, 03/03/22, 250),

(2,3, 1, 03/07/22, 2500),

(3,2, 1, 03/20/22, 250),

(4,2, 1, 04/01/22, 2500),

(5,1, 1, 04/02/22, 25),

(6,1, 1, 04/02/22, 50)

;

CREATE TABLE PRESCRIPTIONS (

prescription\_ID INT NOT NULL PRIMARY KEY,

medRecord\_ID INT NOT NULL,

medical\_ID INT NOT NULL,

Dosage VARCHAR(40) NOT NULL,

Brand\_name VARCHAR(40) NOT NULL,

issued\_Date DATE NOT NULL,

expiration\_Date DATE NOT NULL,

FOREIGN KEY(medRecord\_ID) REFERENCES Medical\_Record(medRecord\_ID),

FOREIGN KEY(medical\_ID) REFERENCES Medical(medical\_ID)

);

INSERT INTO PRESCRIPTIONS

(prescription\_ID, medRecord\_ID, medical\_ID, Dosage, Brand\_name, issued\_Date, expiration\_Date)

VALUES

(23456, 121212, 11119, '500mg per day', 'tylenol', '2022-2-7', '2022-2-21'),

(23457, 131313, 11118, '1000mg per day', 'amoxil', '2022-2-13', '2022-2-27'),

(23458, 141414, 11117, '1000mg per day', 'advil', '2022-2-14', '2022-2-28'),

(23459, 151515, 11116, '500mg per day', 'aspirin', '2022-2-23', '2022-3-9'),

(23460, 161616, 11115, '500mg per day', 'tylenol', '2022-2-28', '2022-3-14')

;

CREATE TABLE ALLERGIES (

allergy\_ID INT NOT NULL PRIMARY KEY,

Allergen\_name VARCHAR(30) NOT NULL

);

INSERT INTO ALLERGIES

(allergy\_ID, Allergen\_name)

VALUES

(1, 'peanut'),

(2, 'gluten'),

(3, 'milk'),

(4, 'egg')

;

CREATE TABLE LICENSE (

license\_ID INT NOT NULL PRIMARY KEY,

Type VARCHAR(30) NOT NULL,

Licensure\_name VARCHAR(30) NOT NULL

);

INSERT INTO LICENSE

(license\_ID, Type, Licensure\_name)

VALUES

(1, 'license', 'Dentist License'),

(2, 'license', 'Dental Hygiene License'),

(3, 'certificate', 'Dental Assistant Certifiaction')

;

CREATE TABLE MEDICAL\_LICENSE (

licenseIssued\_ID INT NOT NULL PRIMARY KEY,

issue\_Date DATE NOT NULL,

Exp\_Date DATE NOT NULL,

stateIssued VARCHAR(30) NOT NULL,

medical\_ID INT NOT NULL,

license\_ID INT NOT NULL,

FOREIGN KEY(medical\_ID) REFERENCES Medical(medical\_ID),

FOREIGN KEY(license\_ID) REFERENCES license(license\_ID)

);

INSERT INTO MEDICAL\_LICENSE

(licenseIssued\_ID, issue\_Date, Exp\_Date, stateIssued, medical\_ID, license\_ID)

VALUES

(33333, '2021-3-16', '2023-3-16', 'OH', 54321, 1),

(44444, '2021-7-23', '2023-7-23', 'OH', 54322, 2),

(55555, '2021-6-4', '2023-6-4', 'OH', 54323, 2),

(66666, '2021-5-12', '2023-5-12', 'OH', 54324, 3),

(77777, '2021-5-27', '2023-5-27', 'OH', 54325, 3)

;

CREATE TABLE MEDICAL\_RECORD (

medRecord\_ID INT NOT NULL PRIMARY KEY,

Blood\_type VARCHAR(5) NOT NULL

);

INSERT INTO MEDICAL\_RECORD

(medRecord\_ID, Blood\_type)

VALUES

(121212, 'A'),

(131313, 'B'),

(141414, 'A'),

(151515, 'O'),

(161616, 'AB')

;

CREATE TABLE MEDICAL\_DIAGNOSIS (

diagnosis\_ID INT NOT NULL PRIMARY KEY,

diagnosisCode VARCHAR(20) NOT NULL,

diagnosisName VARCHAR(30) NOT NULL

);

INSERT INTO MEDICAL\_DIAGNOSIS

(diagnosis\_ID, diagnosisCode, diagnosisName)

VALUES

(13579, 97531, 'cracked teeth'),

(13578, 87531, 'falling dental work'),

(13577, 77531, 'poor oral hygiene'),

(13576, 67531, 'braces check'),

(13575, 57531, 'sensitive teeth')

;

CREATE TABLE MEDICALRECORD\_ALLERGIES (

allergy\_ID INT NOT NULL,

medRecord\_ID INT NOT NULL,

Severity VARCHAR(30) NOT NULL,

FOREIGN KEY(allergy\_ID) REFERENCES allergies(allergy\_ID),

FOREIGN KEY(medRecord\_ID) REFERENCES medical\_record(medRecord\_ID),

PRIMARY KEY(allergy\_ID, medRecord\_ID)

);

INSERT INTO MEDICALRECORD\_ALLERGIES

(allergy\_ID, medRecord\_ID, Severity)

VALUES

(1, 121212, 'mild'),

(1, 131313, 'deadly'),

(2, 141414, 'mild'),

(3, 151515, 'severe'),

(4, 161616, 'mild')

;

CREATE TABLE MEDICALRECORD\_DIAGNOSIS (

medRecord\_diagnosisID INT NOT NULL,

medRecord\_ID INT NOT NULL,

diagnosis\_ID INT NOT NULL,

diagnosisDate DATE NOT NULL,

FOREIGN KEY(medRecord\_ID) REFERENCES medical\_record(medRecordID),

FOREIGN KEY(diagnosis\_ID) REFERENCES medical\_diagnosis(diagnosis\_ID),

PRIMARY KEY(medRecord\_diagnosisID)

);

INSERT INTO MEDICALRECORD\_DIAGNOSIS

(medRecord\_diagnosisID, medRecord\_ID, diagnosis\_ID, diagnosisDate)

VALUES

(35423, 121212, 13579, '2022-2-6'),

(23457, 131313, 13578, '2022-2-12'),

(15234, 141414, 13577, '2022-2-13'),

(32541, 151515, 13576, '2022-2-22'),

(23421, 161616, 13575, '2022-2-17')

;

CREATE TABLE PATIENT\_MEDRECORD (

patient\_ID INT NOT NULL,

medRecord\_ID INT NOT NULL,

FOREIGN KEY(patient\_ID) REFERENCES patient(patient\_ID),

FOREIGN KEY(medRecord\_ID) REFERENCES medical\_record(medRecordID),

PRIMARY KEY(patient\_ID, medRecord\_ID)

);

INSERT INTO PATIENT\_MEDRECORD

(patient\_ID, medRecord\_ID)

VALUES

(11115, 121212),

(11116, 131313),

(11117, 141414),

(11118, 151515),

(11119, 161616)

;

CREATE TABLE PERSON (

person\_ID INT NOT NULL PRIMARY KEY,

Ssn INT NOT NULL,

Fname VARCHAR(255) NOT NULL,

Lname VARCHAR(255) NOT NULL,

Phone INT NOT NULL,

Race VARCHAR(20) NOT NULL,

Sex VARCHAR(1) NOT NULL,

Disability VARCHAR(20) NOT NULL,

Veteran VARCHAR(1) NOT NULL,

Email VARCHAR(40) NOT NULL,

birth\_Date DATE NOT NULL,

addr\_ID INT NOT NULL,

FOREIGN KEY(addr\_ID) REFERENCES Address(addr\_ID)

);

INSERT INTO PERSON

(person\_ID, Ssn, Fname, Lname, Phone, Race, Sex, Disability, Veteran, Email, birth\_Date, addr\_ID)

VALUES

(9981, 284164870, 'John', 'Doe', 1112223333, 'White', 'M', 0, 0, 'a@b.com', '03-23-2000', 1),

(9982, 284864871, 'Jake', 'Doe', 1112423333, 'Asian', 'M', 0, 0, 'c@b.com', '04-23-2002', 1),

(9983, 284664872, 'Alex', 'Hunt', 1112923333, 'White', 'M', 1, 1, 'dd@bb.com', '06-03-2001', 1),

(9984, 284464873, 'Jane', 'Smith', 0112223333, 'White', 'F', 0, 1, 'aaa@bb.com', '07-23-1994', 1),

(9985, 284164844, 'Jane', 'Xu', 1112023333, 'Asian', 'F', 0, 0, 'abbb@bb.com', '03-23-1986', 1),

(9991, 384164870, 'Danny', 'Smilow', 1912223333, 'White', 'M', 0, 0, 'a@a.com', '02-23-2000', 1),

(9992, 384864871, 'Jake', 'Xu', 1912423333, 'Asian', 'M', 0, 0, 'c@a.com', '04-23-2000', 1),

(9993, 384664872, 'Alex', 'Rice', 1192923333, 'White', 'M', 1, 1, 'dd@aa.com', '06-03-2000', 1),

(9994, 384464873, 'Jane', 'Johnson', 9112223333, 'White', 'F', 0, 1, 'aaa@aa.com', '07-23-1984', 1),

(9995, 384164844, 'Jane', 'Xu', 1912023333, 'Asian', 'F', 0, 0, 'abbb@aa.com', '03-23-1976', 1)

;

CREATE TABLE PATIENT (

patient\_ID INT NOT NULL PRIMARY KEY,

person\_ID INT NOT NULL,

payInfo\_ID INT NOT NULL,

insurance\_ID INT NOT NULL,

HIPPA\_signed\_date DATE NOT NULL,

last\_xRay DATE NOT NULL,

Active boolean NOT NULL,

FOREIGN KEY(person\_ID) REFERENCES Person(person\_ID),

FOREIGN KEY(payInfo\_ID) REFERENCES payment\_info(payInfo\_ID),

FOREIGN KEY(insurance\_ID) REFERENCES insurance\_plan(insurance\_ID)

);

INSERT INTO PATIENT

(patient\_ID, person\_ID, payInfo\_ID, insurance\_ID, HIPPA\_signed\_date, last\_xRay, Active)

VALUES

(11115, 9981, 3, 1,'2022-03-23', '2022-04-23', 1),

(11116, 9982, 3, 1,'2021-12-21', '2022-01-21', 1),

(11117, 9983, 1, 1,'2022-02-24', '2022-03-21', 1),

(11118, 9984, 3, 1,'2022-02-26', '2022-02-21', 1),

(11119, 9985, 2, 1,'2022-02-01', '2022-01-21', 1)

;

CREATE TABLE EMERGENCY\_CONTACTS (

person\_ID INT NOT NULL,

patient\_ID INT NOT NULL,

FOREIGN KEY(person\_ID) REFERENCES person(personID),

FOREIGN KEY(patient\_ID) REFERENCES patient(patient\_ID),

PRIMARY KEY(person\_ID, patient\_ID)

);

INSERT INTO EMERGENCY\_CONTACTS

(person\_ID, patient\_ID)

VALUES

(9991, 11115),

(9992, 11116),

(9993, 11117),

(9994, 11118),

(9995, 11119)

;

CREATE TABLE EMPLOYEE (

emp\_ID INT NOT NULL PRIMARY KEY,

person\_ID INT NOT NULL,

Salary INT NOT NULL,

Shift VARCHAR(30) NOT NULL,

location\_ID INT NOT NULL,

FOREIGN KEY(location\_ID) REFERENCES location(location\_ID),

FOREIGN KEY(person\_ID) REFERENCES person(person\_ID)

);

INSERT INTO EMPLOYEE

(emp\_ID, person\_ID, Salary, Shift, location\_ID)

VALUES

(100, 9991, 43093, '7am-5pm', 1),

(101, 9992, 73095, '7am-5pm', 1),

(102, 9993, 143097, '8pm-6am', 1),

(103, 9994, 43391, '8pm-6am', 1),

(104, 9995, 82090, '7am-5pm', 1)

;

CREATE TABLE MEDICAL (

medical\_ID INT NOT NULL PRIMARY KEY,

emp\_ID INT NOT NULL,

FOREIGN KEY(emp\_ID) REFERENCES employee(emp\_ID)

);

INSERT INTO MEDICAL

(medical\_ID, emp\_ID)

VALUES

(54321 , 100),

(54322 , 101),

(54323 , 102),

(54324 , 103),

(54325 , 104)

;

Using the database, we are able to generate the following SQL queries:

1. Create a list of patients and the medications they currently take. Sort your list by

patient’s last name and medication name in alphabetical order. Include other

applicable details such as date prescribed and dosage.

SELECT PERSON.FName, PRESCRIPTIONS.Brand\_name, PRESCRIPTIONS.Dosage, PRESCRIPTIONS.issued\_Date FROM PATIENT\_MEDRECORD

INNER JOIN PATIENT ON PATIENT.patient\_ID = PATIENT\_MEDRECORD.patient\_ID

INNER JOIN PRESCRIPTIONS ON PRESCRIPTIONS.medRecord\_ID = PATIENT\_MEDRECORD.medRecord\_ID

INNER JOIN PERSON ON PERSON.person\_ID = PATIENT.person\_ID

GROUP BY PERSON.Fname

ORDER BY PERSON.Fname, PRESCRIPTIONS.Brand\_name;

b. Display patient information for patients who currently have Delta Dental

insurance policy.

SELECT \* FROM PATIENT

INNER JOIN INSURANCE ON INSURANCE.insurance\_ID = PATIENT.insurance\_ID

WHERE INSURANCE.Name = 'Delta Dental';

c. Generate a list of procedures and dates of service performed by doctor Smilow.

SELECT PROCEDURES.procedurename,APPOINTMENT.appointment\_date FROM PROCEDURES

INNER JOIN INVOICE ON INVOICE.invoice\_ID = PROCEDURES.invoice\_ID

INNER JOIN APPOINTMENT\_INVOICES ON APPOINTMENT\_INVOICES.invoice\_id = INVOICE.invoice\_ID

INNER JOIN APPOINTMENT ON APPOINTMENT.appointment\_id = APPOINTMENT\_INVOICES.appointment\_id

INNER JOIN MEDICAL ON MEDICAL.medical\_ID = PROCEDURES.medical\_ID

INNER JOIN EMPLOYEE ON EMPLOYEE.emp\_id = MEDICAL.emp\_id

INNER JOIN PERSON ON PERSON.person\_ID = EMPLOYEE.person\_ID

WHERE PERSON.Lname = 'Smilow'

d. Print out a list of past due invoices with patient contact information. Past due is

defined as over 30 days old with a balance over $10.

SELECT INVOICE.invoice\_ID, APPOINTMENT.appointment\_date, PERSON.Fname, PERSON.Lname FROM INVOICE

INNER JOIN PATIENT ON PATIENT.patient\_ID = INVOICE.patient\_ID

INNER JOIN APPOINTMENT\_INVOICES ON APPOINTMENT\_INVOICES.invoice\_id = INVOICE.invoice\_ID

INNER JOIN APPOINTMENT ON APPOINTMENT.appointment\_id = APPOINTMENT\_INVOICES.invoice\_id

INNER JOIN PERSON ON PERSON.person\_id = PATIENT.person\_ID

WHERE APPOINTMENT.appointment\_date < DateTime('Now', 'LocalTime', '-30 Day') AND (INVOICE.charge - INVOICE.paid > 10);

e. Find the patients who brought the most revenue in the past year. You can define

how many records you want to display in the result of this query.

SELECT PERSON.Fname, PERSON.Lname, INVOICE.paid FROM INVOICE

INNER JOIN PATIENT ON PATIENT.patient\_ID = INVOICE.patient\_ID

INNER JOIN PERSON ON PERSON.person\_id = PATIENT.person\_ID

GROUP BY PERSON.Fname, PERSON.Lname

ORDER BY INVOICE.paid DESC

LIMIT 10;

f. Create a list of doctors who performed less than 5 procedures this year.

SELECT M.medical\_ID, PERSON.fname, PERSON.lname FROM MEDICAL AS M

LEFT OUTER JOIN PROCEDURES AS P ON M.medical\_ID = P.medical\_ID

INNER JOIN EMPLOYEE ON EMPLOYEE.emp\_id = M.emp\_ID

INNER JOIN PERSON ON PERSON.person\_ID = EMPLOYEE.person\_id

GROUP BY M.medical\_ID

HAVING COUNT(P.procedure\_ID) < 5;

g. Find the highest paying procedures, procedure price, and the total number of

those procedures performed. Sort your list with highest paying procedures

showing at the top of your list.

SELECT P.procedure\_ID, P.procedureName,P.charge, COUNT(P.procedure\_ID) AS Count

FROM PROCEDURES AS P

ORDER BY P.charge;

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. Find the name of the most popular insurance plan currently used by the patients.

SELECT I.Name FROM PATIENT AS P

INNER JOIN INSURANCE AS I ON I.insurance\_ID = P.insurance\_ID

GROUP BY (I.Name)

ORDER BY COUNT(I.Name) DESC LIMIT 1;

## Additional Queries

In addition to these queries, the following were drafted:

1. List the names and addresses of all the dental offices

SELECT LOCATION.Name, ADDRESS.Street, ADDRESS.City, ADDRESS.State, ADDRESS.Zip FROM LOCATION

INNER JOIN ADDRESS ON ADDRESS.addr\_ID = LOCATION.addr\_ID;

2. Find the first and last name and phone number of the emergency contact for all patients.

SELECT PERSON.Phone, PERSON.Fname, PERSON.Lname FROM PERSON

INNER JOIN EMERGENCY\_CONTACTS ON EMERGENCY\_CONTACTS.person\_ID = PERSON.person\_ID

3. Find all invoices that are over $1,000 for a patient.

SELECT \* FROM INVOICE

WHERE CHARGE > 1000