SMILOW DENTISTRY

Smilow Dentistry Database

Aaron Post, Noah Perkins, Keyang Zhang, and Saeed Alneyadi

CSE 3241

Professor Zina Pichkar

July 29, 2022

Table of Contents

Team Description, Introduction………………………………………………………………...3

Project Summary…………………………………………………………………………………4

E(ERD)..5

Relational Schema Documentation……………………………………………………………..6-7

Complete Relational Schema……………………………………………………………….8-10

Relational Schema Diagram………………………………………………………………….10-11

Relational Algebra……………………………………………………………………………12-14

Normalization………………………………………………………………………………………14

User Manual: Table Descriptions…………………………………………………………….15-26

User Manual: Catalog of SQL Queries……………………………………………………….27-33

User Manual: Insert and Delete SQL Examples…………………………………………………34

User Manual: Two Indexes……………………………………………………………………...34-35

User Manual: Two Views…………………………………………………………………….…..35

User Manual: Two Transactions…………………………………………………………………36

Team Member Contributions…………………………………………………………………….37

Project Reflection………………………………………………………………………………..37

Feedback and Revision Process………………………………………………………………37-38

Project Checkpoints……………………………………………………………………………...39

Testing Queries and SQL………………………………………………………………………...40

Section 1 - Database Description

Team Description

Our team consists of Aaron Post, Keyang Zhang, Noah Perkins, and Saeed Alneyadi. Over the course of this project, we communicated via group text on Discord. There were some timezone differences between members of our group (as well as different work schedules), but we accounted for these concerns by regularly messaging each other to make sure we were on track to complete the necessary work.

Introduction

Almost every business could benefit from a database; this includes the obvious “data giants” like Google and Facebook, but also extends to many small businesses. Small businesses can benefit from a database because it provides efficient means to store, access, and manipulate customer records. According to business.com, popular database software used for business includes Informatica, Azure Data Catalog, and more. In the case of a business such as Smilow Dentistry, however, a medical-oriented DBMS would likely be used instead such as Denticon, Curve Dental, etc. Ideally, though, a custom database solution would be created for every business to accommodate their specific needs best through multiple interviews. To emulate this process, we interviewed a family member with decades of experience in the administration side of the medical industry. He explained how the registration, scheduling, and billing processes work. Firstly, a customer calls the dentist's office to schedule an appointment. If it is their first appointment, then they must give extensive information about their demographics. This includes (but is not limited to) their insurance, name, date of birth, social security number, address, phone number, place of work, medical history, type of insurance, and whether or not their insurance is within the network. Many offices have deals with specific insurance companies which can offer additional benefits to patients within the network. In other words, their financial support may be limited if a customer’s insurance is out of the network. Once all of this information is communicated, their information is stored in the database. They can now schedule an appointment over the phone, specifying a time and date that works for the client and dentist. The first appointment will be the longest, and any follow-ups should be shorter unless there are special conditions such as procedures or x-rays. The client will then attend their appointment, and the Dentist will record what services they performed (procedures etc) and complete the billing form for the insurance company. The bill is then sent to the client’s insurance company for review and to pay the dentist according to the insurance policies. Finally, the patient receives an EOB (explanation of insurance benefits and what needs to be paid by them) and their invoice. As for employees, each employee at the dentist's office has several attributes that will need to be noted. This includes their name, date of birth, address, social security number, schooling (level and where), certifications and dates of expiration, years of experience, previous places of work, special skills/talents/training, accommodations if they are disabled, and which location/department they will work in.

Project Summary

Through this project, we will create a relational database for our client Smilow Dentistry. It will be fully functional and have everything the client should expect including registration scheduling and billing. We will complete all necessary, industry-standard steps, such as creating an E(ERD), relational schema, queries using relational algebra and SQL, populating data records, and documenting everything in the form of a user manual.

Database Additional Features

a.) Save a preferred day of the week and time for appointments

- Two more attributes on the patient: Preferred Day, Preferred Time

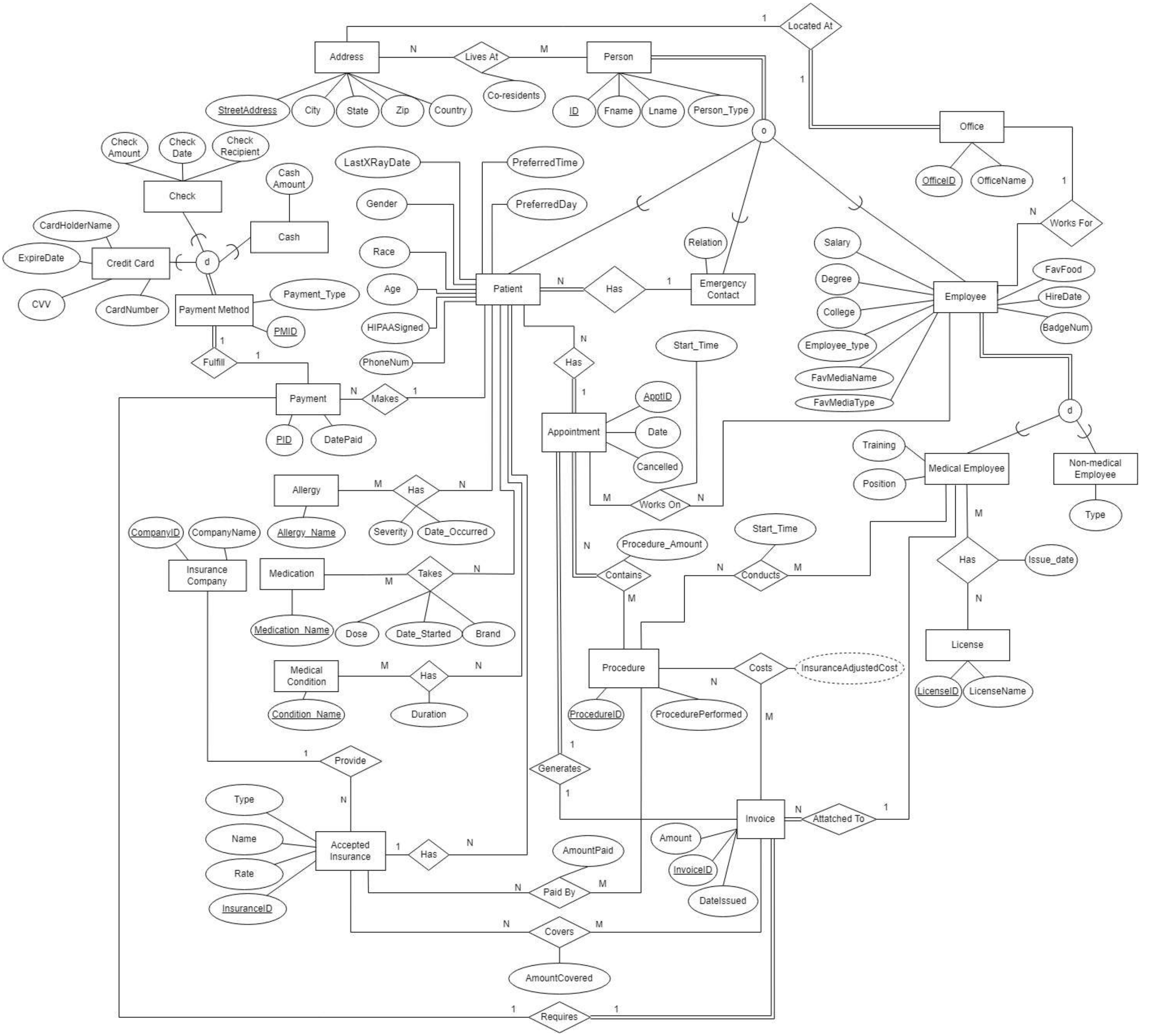
b.) Employee Hire Date Anniversary Parties

- Four more attributes on an employee: Hire Date, Favorite Food, Favorite Media Name(Interstellar), Favorite Media Type(Movie)

Requirements & Assumptions

- Assuming that patient insurance types are different than the accepted type, so a patient could have insurance that is not accepted.
- Offices can take multiple appointments, and each appointment belongs to this exact office and each appointment must be taken by the office.
- One appointment generates one billing record, and each billing record must be generated only by one appointment.
- One appointment must have and can contain multiple dental procedures, and each dental procedure is contained in one appointment.
- An insurance plan can pay for multiple dental procedures, and each dental procedure is paid for by one insurance plan.
- Offices can have multiple employees, and each employee belongs to this exact office.
- One patient can have one insurance plan, and each insurance plan can be selected by multiple patients.
- One insurance plan can have multiple accepted insurance policy types, and each accepted insurance policy type is included in one insurance plan.
- One patient can have multiple appointments, and each appointment belongs to the patient that makes it, and each appointment must have its owner.
- Each patient can make multiple payments, and each payment belongs to one patient.
- Each patient can have multiple kinds of allergies, and one allergy can be possessed by multiple patients.
- Each patient can take multiple medications, and each medication can be taken by multiple patients.
- Each patient can have multiple medical conditions, and each medical condition can belong to multiple patients.

(E)ERD (File also included for better visibility, "EERD.jpg")

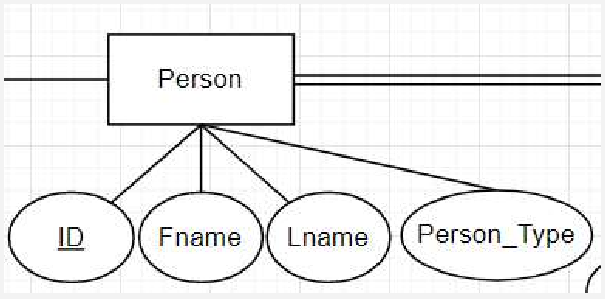


Relational Schema Documentation

We used the following six-step algorithm to create our relational schema:

I. Handle Regular Entity Types

We first mapped every regular entity into a relation, adding all simple attributes as attributes of the relation. The primary key of each entity also becomes the primary key of the relation.



Example Regular Entity: Person

Person(ID, Fname, Name, Person_Type)

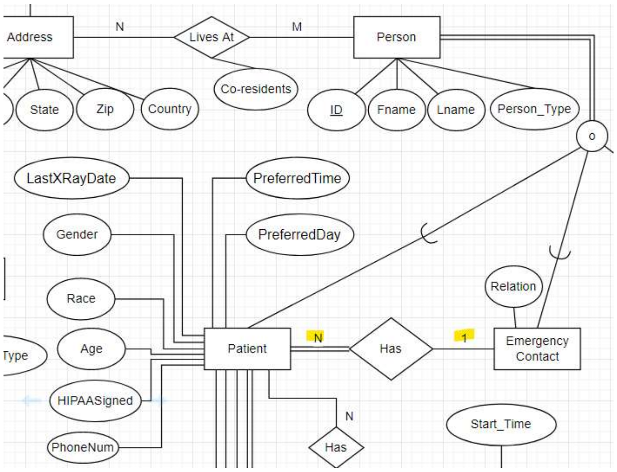
- ID is the Primary Key on the ERD, so it becomes the Primary Key of the relation.
- Fname, Lname, and Person_Type are all simple attributes, so they are added as attributes of the relation.

II. Handle Weak Entity Types

There were no weak entities in our ERD that we had to map into relations. If we did have one, we would have created a new relation and added all of its simple attributes just like a regular entity, and then add the primary key of its owner as a foreign key to the relation.

III. Handle Binary N:1 Relationships

We then mapped all N:1 Relationships by using the Foreign Key approach, adding the key attribute from the 1 side as a foreign key to the relation on the N side. There is an example below.



M:1 Relationship: Patient Has Emergency Contact

EmergencyContact(PersonID, Relation)

- Patient is a subclass of Person, so it includes Person's Primary Key "ID" as a Foreign Key called "PersonID"

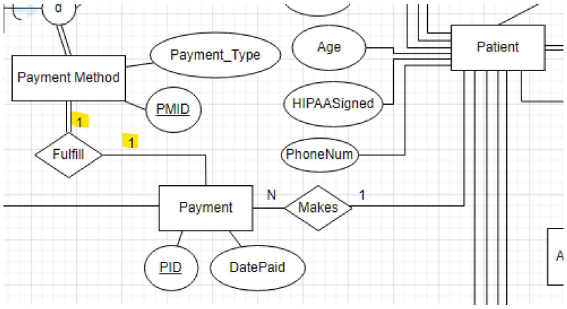
Patient(PersonID, EID, IPID, LastXRayDate, Gender, Race, Age, HIPAASigned, PreferredTime, PreferredDay)

- Also includes Foreign Key "PersonID" because Patient is also a subclass of Person
- Includes Key of Emergency Contact "PersonID" as a foreign key because of N:1 Relation
- Includes IPID because of another relationship.

IV. Binary 1:1 Relationship

We then mapped all 1:1 relationships, using the foreign key approach. We accomplished this by adding the key attribute from the partial participation side as a foreign key to the fully participating side. We chose to use the foreign key approach for every 1:1 relationship because it was easier to read and communicate effectively as a team. There is an example below.

1:1 Relationship: PaymentMethod Fulfills Payment



Payment(PID, PayeeID, DatePaid)

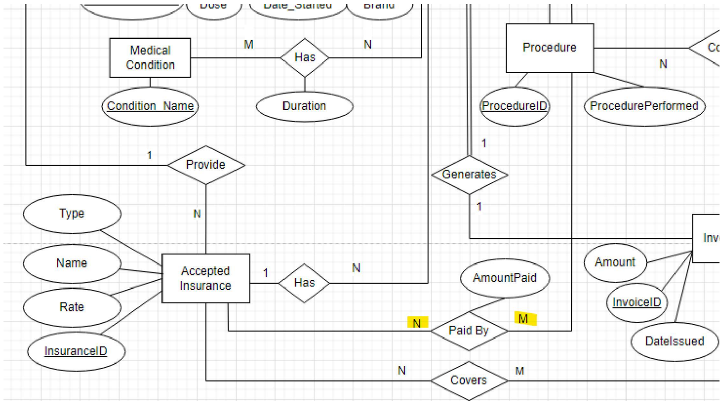
- Includes Foreign Key “PayeeID” to Key “PersonID” in Patient because of N:1 Relationship
- Primary Key is “PID”

PaymentMethod(PMID, PID, Payment_Type)

- Includes Foreign Key “PID” to Key “PID” in Payment because of 1:1 Relationship
- Primary Key is “PMID”

V. Binary M:N Relationship

We then mapped all M:N relationships which were the most complex to deal with. After mapping both entities as regular entity types, we also created “Join” relations including Primary Keys from both entities as Foreign Keys in the new relation, as well as any attributes that were attached to that relationship. There is an example below.



Regular Entities:

AcceptedInsurance(IPID, Name, Type, Rate)

- Primary Key is “IPID”

Procedure(ProcedureID, ProcedurePerformed)

- Primary Key is “ProcedureID”

Join:

Procedure-Accepted-Insurance-Join(ProcedureID, IPID, AmountPaid, PerUnitCharge)

- Includes Foreign Key “ProcedureID” to Primary Key “ProcedureID” in Procedure
- Includes Foreign Key “IPID” to Primary Key “IPID” in AcceptedInsurance

VI. Multivalued Attributes

We have no multivalued attributes, so we do not need to map them. If we had multivalued attributes, we would create a new relation and employ the foreign key approach again.

Complete Relational Schema

Person(ID, Fname, Name, Person_Type)

Primary Key: "ID"

Employee(PersonID, OID, Employee_Type, BadgeNum, College, Degree, Salary, FavMediaName, FavMediaType, FavFood, HireDate)

Foreign Key: "PersonID" to Primary Key "ID" in Person

Foreign Key: "OID" to Primary Key "OID" in Office

MedicalEmployee(PersonID, Training, Position)

Foreign Key: "PersonID" to Key "PersonID" in Employee

NonMedicalEmployee(PersonID, Type)

Foreign Key: "PersonID" to Key "PersonID" in Employee

Patient(PersonID, EID, IPID, LastXRayDate, Gender, Race, Age, HIPAASigned, PreferredTime, PreferredDay)

Foreign Key: "PersonID" to Primary Key "ID" in Person

Foreign Key: "EID" to Key "PersonID" in EmergencyContact

Foreign Key: "IPID" to Primary Key "IPID" in AcceptedInsurance

EmergencyContact(PersonID,Relation)

Foreign Key: "PersonID" to Primary Key "ID" in Person

Address(StreetAddress, City, State, Zip, Country)

Primary Key: "StreetAddress"

Office(OID, StreetAddress,OfficeName)

Primary Key: "OID"

Foreign Key: "StreetAddress" to Primary Key "StreetAddress" in Address

License(LicenseID, LicenseName)

Primary Key: "LicenseID"

Payment(PID, PayeeID, DatePaid)

Primary Key: "PID"

Foreign Key: "PayeeID" to Key "PersonID" in Patient

PaymentMethod(PMID, PID, Payment_Type)

Primary Key: "PMID"

Foreign Key: "PID" to Primary Key "PID" in Payment

Cash(PMID, CashAmount)

Foreign Key: "PMID" to Primary Key "PMID" in PaymentMethod

Check(PMID, CheckAmount, CheckDate, CheckRecipient)

Foreign Key: "PMID" to Primary Key "PMID" in PaymentMethod

CreditCard(PMID, CVV, ExpireDate, CardNumber, CardHolderName)

Foreign Key: "PMID" to Primary Key "PMID" in PaymentMethod

Allergy(Allergy_Name)

Primary Key: "Allergy_Name"

Medication(Medication_Name)

Primary Key: “Medication_Name

MedicalCondition(Condition_Name)

Primary Key: “Condition_Name”

AcceptedInsurance(IPID, Name, Type, Rate)

Primary Key: “IPID”

Invoice(IID, PID, ProfessionalsID, DateIssued, Amount)

Primary Key: “IID”

Foreign Key: “PID” to Primary Key “PID” in Payment

Foreign Key: “ProfessionalsID” to “PersonID” in MedicalProfessional

Appointment(AppointmentID, PatientID, IID, Date, Cancelled)

Primary Key: “AppointmentID”

Foreign Key: “PatientID” to Key “PersonID” in Patient

Foreign Key: “IID” to Primary Key “IID” in Invoice

Procedure(ProcedureID, ProcedurePerformed)

Primary Key: “ProcedureID”

MedicalEmployee-License-Join(PersonID, LicenseID, Issue_date)

Foreign Key: “PersonID” to Key “PersonID” in MedicalEmployee

Foreign Key: “LicenseID” to Primary Key “LicenseID” in License

Patient-Allergy-Join(PatientID, AID)

Foreign Key: “PatientID” to Key “PersonID” in Patient

Foreign Key: “AID” to Primary Key “Allergy_Name” in Allergy

Patient-Medications-Join(PatientID, Medication_Name, Brand)

Foreign Key: “PatientID” to Key “PersonID” in Patient

Foreign Key: “Medication_Name” to Primary Key “Medication_Name” in Medication

Patient-MedicalConditions-Join(PatientID, Condition_Name)

Foreign Key: “PatientID” to Key “PersonID” in Patient

Foreign Key: “Condition_Name” to Primary Key “Condition_Name” in MedicalCondition

Person-Address-Join(PersonID, StreetAddress, Address_Type)

Foreign Key: “PersonID” to Primary Key “ID” in Person

Foreign Key: “StreetAddress” to Primary Key “StreetAddress” in Address

Procedure-MedicalProfessional-Join(ProcedureID, ProfessionalsID,Start_Time)

Foreign Key: “ProcedureID” to Primary Key “ProcedureID” in Procedure

Foreign Key: “ProfessionalsID” to Key “PersonID” in MedicalEmployee

Procedure-Accepted-Insurance-Join(ProcedureID, IPID, AmountPaid, PerUnitCharge)

Foreign Key: “ProcedureID” to Primary Key “ProcedureID” in Procedure

Foreign Key: “IPID” to Primary Key “IPID” in AcceptedInsurance

Procedure-Appointment-Join(ProcedureID, AppointmentID, Procedure_Amount)

Foreign Key: “ProcedureID” to Primary Key “ProcedureID” in Procedure

Foreign Key: “AppointmentID” to Primary Key “AppointmentID” in Appointment

Employee-Appointment-Join(EmployeeID, AppointmentID, Start_Time)

Foreign Key: "EmployeeID" to Key "PersonID" in Employee

Foreign Key: "AppointmentID" to Primary Key "AppointmentID" in Appointment

Invoice-AcceptedInsurance-Join(IID, IPID, AmountCovered)

Foreign Key: "IID" to Primary Key "IID" in Invoice

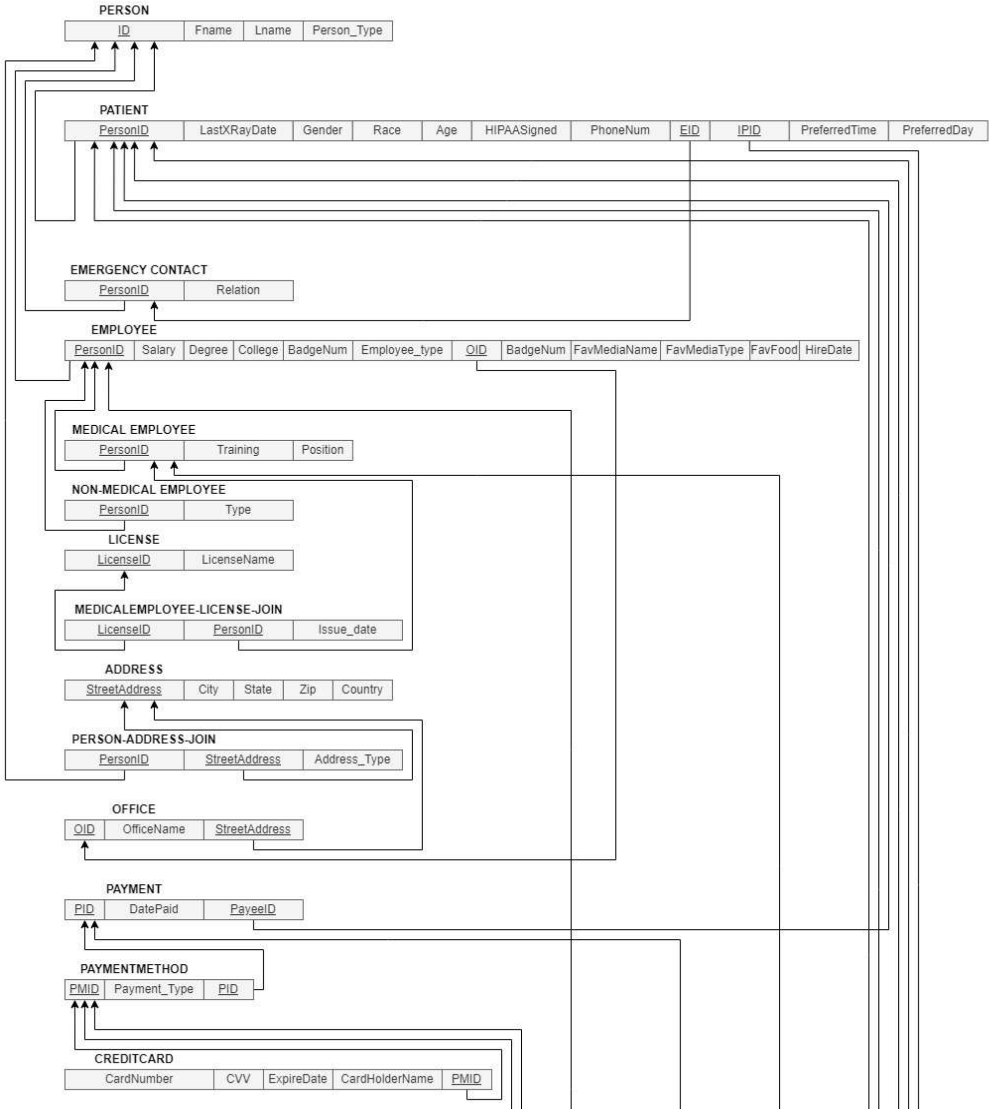
Foreign Key: "IPID" to Primary Key "IPID" in AcceptedInsurance

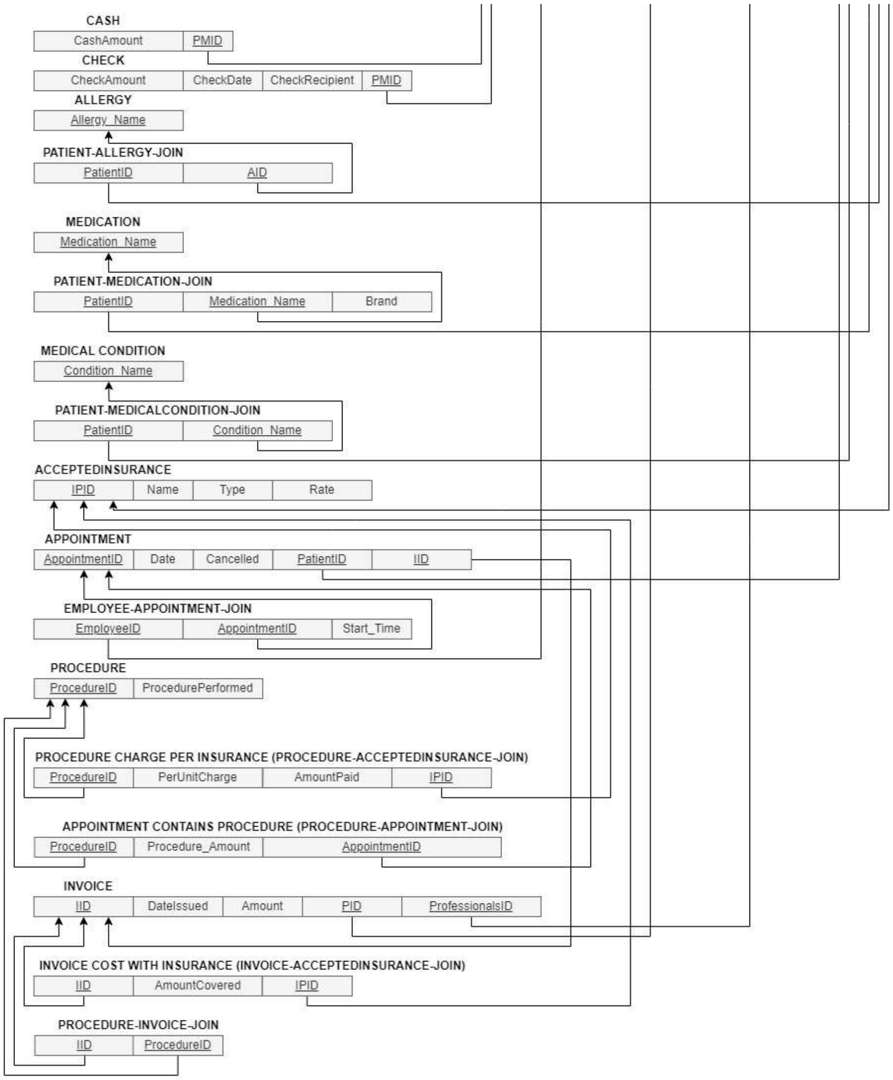
Procedure-Invoice-Join(ProcedureID, IID)

Foreign Key: "ProcedureID" to Primary Key "ProcedureID" in Procedure

Foreign Key: "IID" to Primary Key "IID" in Invoice

Relational Diagram (File also included for better visibility, "RelationalDiagram.jpg")


PERSON
ID | Fname | Lname | Person_Type
PATIENT
PersonID | LastXRayDate | Gender | Race | Age | HIPAASigned | PhoneNum | EID | IPID | PreferredTime | PreferredDay
EMERGENCY CONTACT
PersonID | Relation
EMPLOYEE
PersonID | Salary | Degree | College | BadgeNum | Employee_type | OID | BadgeNum | FavMediaName | FavMediaType | FavFood | HireDate
MEDICAL EMPLOYEE
PersonID | Training | Position
NON-MEDICAL EMPLOYEE
PersonID | Type
LICENSE
LicenseID | LicenseName
MEDICALEMPLOYEE-LICENSE-JOIN
LicenseID | PersonID | Issue_date
ADDRESS
StreetAddress | City | State | Zip | Country
PERSON-ADDRESS-JOIN
PersonID | StreetAddress | Address_Type
OFFICE
OID | OfficeName | StreetAddress
PAYMENT
PID | DatePaid | PayeeID
PAYMENTMETHOD
PMID | Payment_Type | PID
CREDITCARD
CardNumber | CVV | ExpireDate | CardHolderName | PMID



CASH

CashAmount	PMID

CHECK

CheckAmount	CheckDate	CheckRecipient	PMID

ALLERGY

Allergy_Name

PATIENT-ALLERGY-JOIN

PatientID	AID

MEDICATION

Medication_Name

PATIENT-MEDICATION-JOIN

PatientID	Medication_Name	Brand

MEDICAL CONDITION

Condition_Name

PATIENT-MEDICALCONDITION-JOIN

PatientID	Condition_Name

ACCEPTEDINSURANCE

IPID	Name	Type	Rate

APPOINTMENT

AppointmentID	Date	Cancelled	PatientID	IID

EMPLOYEE-APPOINTMENT-JOIN

EmployeeID	AppointmentID	Start_Time

PROCEDURE

ProcedureID	ProcedurePerformed

PROCEDURE CHARGE PER INSURANCE (PROCEDURE-ACCEPTEDINSURANCE-JOIN)

ProcedureID	PerUnitCharge	AmountPaid	IPID

APPOINTMENT CONTAINS PROCEDURE (PROCEDURE-APPOINTMENT-JOIN)

ProcedureID	Procedure_Amount	AppointmentID

INVOICE

IID	DateIssued	Amount	PID	ProfessionalsID

INVOICE COST WITH INSURANCE (INVOICE-ACCEPTEDINSURANCE-JOIN)

IID	AmountCovered	IPID

PROCEDURE-INVOICE-JOIN

IID	ProcedureID

Relational Algebra

Simple Queries

$SQ1P1 \leftarrow Person \bowtie_{Person.ID = Patient.PersonID} Patient$

$SQ1P2 \leftarrow SQ1P1 \bowtie_{SQ1P1.PersonID = Patient.PersonID} Patient\_Medication\_Join$

$SQ1P3 \leftarrow SQ1P2 \bowtie_{SQ1P2.Medication_{Name} = Medication.Medication\_Name} Medication$

$SQ1P3 \leftarrow \sigma_{Fname, Lname, SQ1P2.Medication\_Name, Brand}(SQ1P3)$

This relational algebra shows each patient's info is listed with their medications. This consists of three JOINS between four relations and SELECT operation.

$SQ2P1 \leftarrow \sigma_{IPID, Name}(AcceptedInsurance)$

$SQ2P2 \leftarrow Patient \bowtie_{Patient.IPID = SQ2P1.IPID \ AND \ SQ2P1.Name = "Omega"} SQ2P1$

This relational algebra represents patients with insurance from Delta Dental. It consists of one SELECT and one JOIN between two relations.

$SQ3P1 \leftarrow Appointment \bowtie_{Appointment.AppointmentID = Procedure\_Appointment\_Join.AppointmentID} Procedure\_Appointment\_Join$

$SQ3P2 \leftarrow SQ1P1 \bowtie_{SQ3P1.ProcedureId = Procedure.ProcedureID} Patient\_Medication\_Join$

$SQ3P3 \leftarrow SQ3P2 \bowtie_{SQ3P2.ProcedureID = Procedure\_MedicalEmployee\_Join.ProcedureID} Procedure\_MedicalEmployee\_Join$

$SQ3P4 \leftarrow SQ3P3 \bowtie_{SQ3P3.ProfessionalsID = MedicalEmployee.PersonID} MedicalEmployee$

$SQ3P5 \leftarrow SQ3P4 \bowtie_{SQ3P4.PersonID = Person.ID} Person$

$SQ3P6 \leftarrow \sigma_{ProcedurePerformed, Date}(\pi_{Lname = "Smilow"}(SQ3P5))$

This relational algebra gives doctor Smilow performed procedures list each with their dates. This contains one SELECT and five JOINS between six relations.

$SQ5P1 \leftarrow Person \bowtie_{Patient.PersonID = Person.ID} Patient$

$SQ5P2 \leftarrow SQ5P1 \bowtie_{SQ5P1.PersonID = Appointment.PatientID} Appointment$

$SQ5P3 \leftarrow SQ5P2 \bowtie_{SQ5P2.IID = Invoice.IID} Invoice$

$SQ5P4 \leftarrow \sigma_{ID, Fname, Lname, DateIssued, Amount}(\pi_{DateIssued \ BETWEEN \ '2021/01/01' \ AND \ '2021/12/31'}(SQ5P3))$

This relational algebra shows a list of patient contact information with past due invoices. Past due invoices are the ones that are defined as over 30 days old with a balance over $10. This contains one SELECT, one PROJECT, and three JOINS between four relations.

$SQ6P1 \leftarrow Person \bowtie_{MedicalEmployee.PersonID = Person.ID\ AND\ MedicalEmployee.position = 'Dentist'} MedicalEmployee$

$SQ6P2 \leftarrow SQ6P1 \bowtie_{SQ6P1.PersonID = Procedure\_MedicalEmployee\_Join.professionalsid} Procedure\_MedicalEmployee\_Join$

$SQ6P3 \leftarrow SQ6P2 \bowtie_{SQ6P2.ProcedureID = Procedure.ProcedureID} Procedure$

$SQ6P3 \leftarrow \Gamma_{COUNT\ Procedure.ProcedureID}(SQ6P3)$

$SQ6P4 \leftarrow \sigma_{Fname,\ \ Lname}(\pi_{Number < 5}(SQ6P3))$

This relational algebra presents the patients who lead the most revenue in the past year. This contains one SELECT, one PROJECT three JOINS between four relations.

$SQ7P1 \leftarrow Appointment \bowtie_{Appointment.AppointmentID = Procedure\_Appointment\_Join.AppointmentID} Procedure\_Appointment\_Join$

$SQ7P2 \leftarrow Procedure \bowtie_{SQ7P1.ProcedureID = Procedure.ProcedureID} SQ7P1$

$SQ7P3 \leftarrow SQ7P2 \bowtie_{SQ7P2.IID = Invoice.IID} Invoice$

$SQ7P4 \leftarrow \sigma_{ProcedureID,\ \ ProcedurePerformed,\ \ MAX(Amount)}(SQ7P3)$

This relational algebra shows doctors list who performed less than five procedures this year. This contains one SELECT and three JOINS between four relations.

$SQ8P1 \leftarrow Payment \bowtie_{Payment.PID = PaymentMethod.PID} PaymentMethod$

$SQ8P2 \leftarrow SQ8P1 \bowtie_{SQ8P1.PID = Invoice.PID} Invoice$

$SQ8P3 \leftarrow {}_{Payment\_Type}\Gamma_{\ \ COUNT\ distinct\ PID,\ \ SUM\ Amount}(SQ8P2)$

This relational algebra represents procedures with the highest pay, their prices, and the total number of them performed. It contains one SELECT and two JOINS between three relations.

$SQ9P1 \leftarrow Patient \bowtie_{Patient.IPID = AcceptedInsurance.IPID} AcceptedInsurance$

$SQ9P2 \leftarrow {}_{SQ9P1.Name}\Gamma_{COUNT\ SQ9P1.IPID}(SQ9P1)$

$SQ9P3 \leftarrow {}_{SQ9P1.Name}\Gamma_{COUNT\ Number}(SQ9P2)$

This relational algebra finds the patients' most popular insurance plan name. It contains two functions and one JOINS between two relations.

Extra Queries

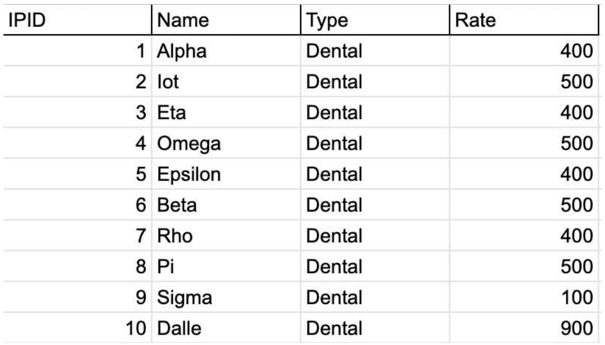
$EXQ1 \leftarrow Patient \times Appointment$
$EXQ1 \leftarrow EXQ1 \bowtie_{Cancelled=False} Invoice$
$EXQ1 \leftarrow \Gamma_{AVERAGE\ Amount}(EXQ1)$
This relational algebra represents the patient info with their uncanceled appointments, and the average amount paid for each. This consists of one cross product, one JOIN, and one function.

$EXQ2 \leftarrow Patient \bowtie_{Patient.PersonID\ =\ Patient\_Allergy\_Join.PersonID} Pateint\_Allergy\_Join$
$EXQ2 \leftarrow EXQ2 \bowtie_{Allergy.allergy\_name\ =\ Allergy.allergy\_name} Allergy$
$EXQ2 \leftarrow {}_{Allergy.allergy\_name}\Gamma_{COUNT\ Distinct\ PersonID}(EXQ2)$
This relational algebra shows the count of allergies type each patient has. This consists of two JOINs and one function.

$EXQ3 \leftarrow \pi_{DateIssued\ >\ 2022/01/01\ AND\ DateIssued\ <\ 2022/12/31}(Invoice)$
$EXQ3 \leftarrow \Gamma_{SUM\ Amount}(EXQ3)$
This relational algebra gives the total payments in the past year (2021). It consists of one PROJECT and one function.

Normalization
Our relational schema is already normalized to BCNF. We know this because it is 1NF since every domain value in our schema is atomic. It is 2NF because the schema is in 1NF, and every attribute that isn't the key is fully dependent on the key. It is also in 3NF because it is in 3NF, and all non-key attributes are non-transitively dependent on the key. Finally, it is in BCNF because it is in 3NF, and all determinants are candidate keys. We intentionally built the relational schema in this fashion to reduce normalization work later on.

Table Description



IPID	Name	Type	Rate
1	Alpha	Dental	400
2	Iot	Dental	500
3	Eta	Dental	400
4	Omega	Dental	500
5	Epsilon	Dental	400
6	Beta	Dental	500
7	Rho	Dental	400
8	Pi	Dental	500
9	Sigma	Dental	100
10	Dalle	Dental	900

AcceptedInsurance

Purpose: This table holds all accepted dental insurances with different rates.

Fields: IPID, Name, Type, Rate

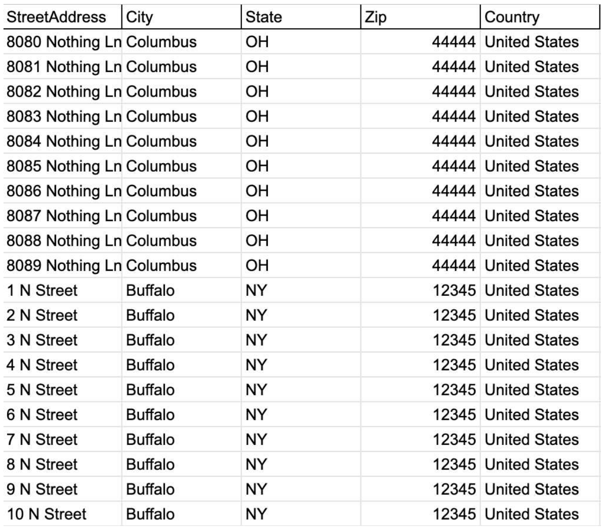
Constraints:

IPID: PRIMARY KEY

Primary Key: IPID

Foreign Key: N/A

SQL Approved Data Types: INT, VARCHAR, CHAR



StreetAddress	City	State	Zip	Country
8080 Nothing Ln	Columbus	OH	44444	United States
8081 Nothing Ln	Columbus	OH	44444	United States
8082 Nothing Ln	Columbus	OH	44444	United States
8083 Nothing Ln	Columbus	OH	44444	United States
8084 Nothing Ln	Columbus	OH	44444	United States
8085 Nothing Ln	Columbus	OH	44444	United States
8086 Nothing Ln	Columbus	OH	44444	United States
8087 Nothing Ln	Columbus	OH	44444	United States
8088 Nothing Ln	Columbus	OH	44444	United States
8089 Nothing Ln	Columbus	OH	44444	United States
1 N Street	Buffalo	NY	12345	United States
2 N Street	Buffalo	NY	12345	United States
3 N Street	Buffalo	NY	12345	United States
4 N Street	Buffalo	NY	12345	United States
5 N Street	Buffalo	NY	12345	United States
6 N Street	Buffalo	NY	12345	United States
7 N Street	Buffalo	NY	12345	United States
8 N Street	Buffalo	NY	12345	United States
9 N Street	Buffalo	NY	12345	United States
10 N Street	Buffalo	NY	12345	United States

Address

Purpose: This table holds all addresses.

Fields: StreetAddress, City, State, Zip, Country.

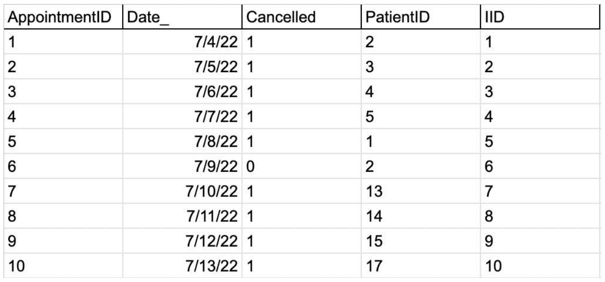
Constraints:

StreetAddress: PRIMARY KEY

Primary Key: StreetAddress

Foreign Key: N/A

SQL Approved Data Types: VARCHAR, CHAR, INT



AppointmentID	Date_	Cancelled	PatientID	IID
1	7/4/22	1	2	1
2	7/5/22	1	3	2
3	7/6/22	1	4	3
4	7/7/22	1	5	4
5	7/8/22	1	1	5
6	7/9/22	0	2	6
7	7/10/22	1	13	7
8	7/11/22	1	14	8
9	7/12/22	1	15	9
10	7/13/22	1	17	10

Appointment

Purpose: This table holds all the appointments made.

Fields: AppointmentID, Date, Cancelled, PatientID, IID

Constraints:

AppointmentID: PRIMARY KEY

PatientID: FOREIGN KEY

IID: FOREIGN KEY

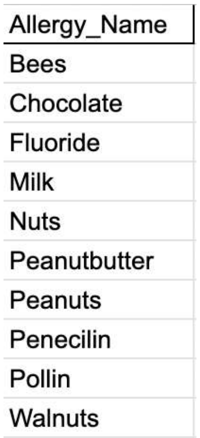
Primary Key: AppointmentID

Foreign Key:

“PatientID” to Key “PersonID” in Patient;

“IID” to Primary Key “IID” in Invoice.

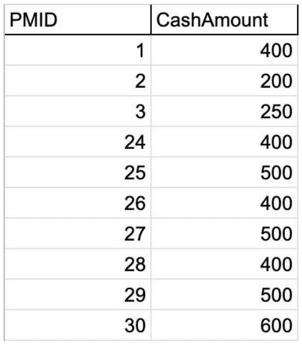
SQL Approved Data Types: INT, DATE



Allergy_Name
Bees
Chocolate
Fluoride
Milk
Nuts
Peanutbutter
Peanuts
Penecilin
Pollin
Walnuts

Allergy

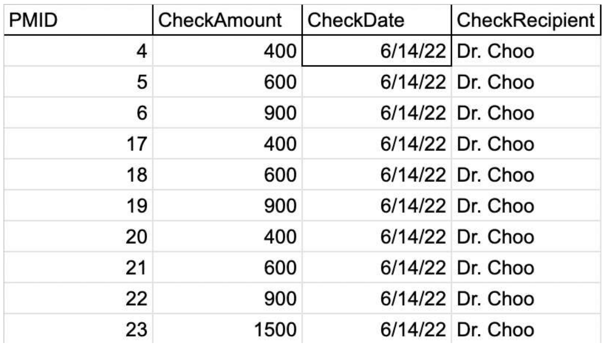
Purpose: This table holds all possible allergies.
Fields: Allergy_Name
Constraints:
Allergy_Name: PRIMARY KEY
Primary Key: Allergy_Name
Foreign Key: N/A
SQL Approved Data Types: VARCHAR



PMID	CashAmount
1	400
2	200
3	250
24	400
25	500
26	400
27	500
28	400
29	500
30	600

Cash

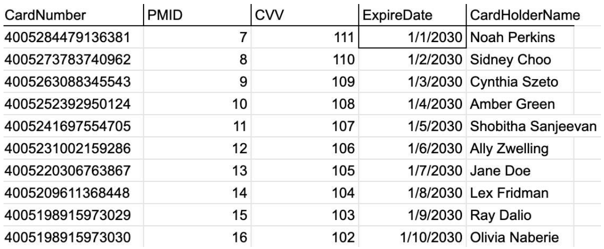
Purpose: This table holds all transactions made with payment method “cash.”
Fields: PMID, CashAmount
Constraints:
PMID: FOREIGN KEY;
CashAmount: NOT NULL.
Primary Key: N/A
Foreign Key:
“PMID” to Primary Key “PMID” in PaymentMethod.
SQL Approved Data Types: INT



PMID	CheckAmount	CheckDate	CheckRecipient
4	400	6/14/22	Dr. Choo
5	600	6/14/22	Dr. Choo
6	900	6/14/22	Dr. Choo
17	400	6/14/22	Dr. Choo
18	600	6/14/22	Dr. Choo
19	900	6/14/22	Dr. Choo
20	400	6/14/22	Dr. Choo
21	600	6/14/22	Dr. Choo
22	900	6/14/22	Dr. Choo
23	1500	6/14/22	Dr. Choo

Check

Purpose: This table holds all transactions made with payment method “check.”
Fields: PMID, CheckAmount, CheckDate, CheckRecipient
Constraints:
PMID: FOREIGN KEY
Primary Key: N/A
Foreign Key:
“PMID” to Primary Key “PMID” in PaymentMethod
SQL Approved Data Types: INT, DATE, VARCHAR



CardNumber	PMID	CVV	ExpireDate	CardHolderName
4005284479136381	7	111	1/1/2030	Noah Perkins
4005273783740962	8	110	1/2/2030	Sidney Choo
4005263088345543	9	109	1/3/2030	Cynthia Szeto
4005252392950124	10	108	1/4/2030	Amber Green
4005241697554705	11	107	1/5/2030	Shobitha Sanjeevan
4005231002159286	12	106	1/6/2030	Ally Zwelling
4005220306763867	13	105	1/7/2030	Jane Doe
4005209611368448	14	104	1/8/2030	Lex Fridman
4005198915973029	15	103	1/9/2030	Ray Dalio
4005198915973030	16	102	1/10/2030	Olivia Naberie

CreditCard

Purpose: This table holds all credit card records from patients.

Fields: CardNumber, PMID, CVV, ExpireDate, CardHolderName

Constraints:

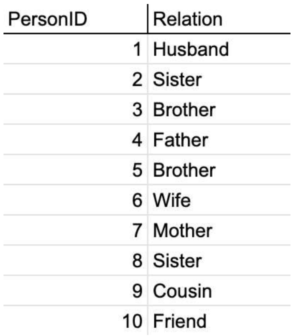
PMID: FOREIGN KEY

Primary Key: N/A

Foreign Key:

"PMID" to Primary Key "PMID" in PaymentMethod

SQL Approved Data Types: INT, DATE, VARCHAR



PersonID	Relation
1	Husband
2	Sister
3	Brother
4	Father
5	Brother
6	Wife
7	Mother
8	Sister
9	Cousin
10	Friend

EmergencyContact

Purpose: This table holds all emergency contacts of corresponding patients.

Fields: PersonID, Relation

Constraints:

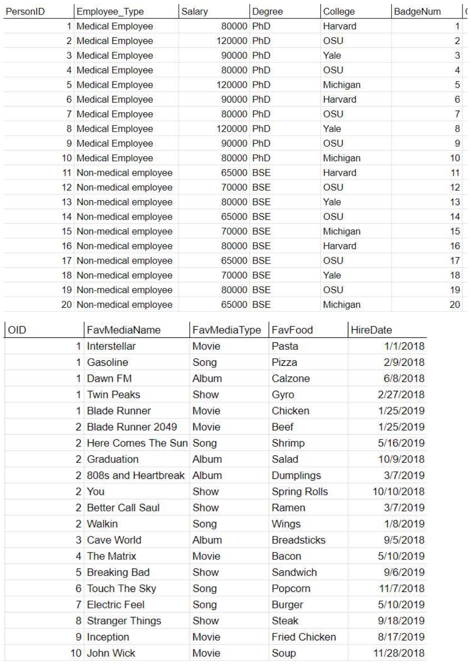
PersonID: FOREIGN KEY

Primary Key: N/A

Foreign Key:

"PersonID" to Primary Key "ID" in Person

SQL Approved Data Types: INT, VARCHAR



PersonID	Employee_Type	Salary	Degree	College	BadgeNum	OID	FavMediaName	FavMediaType	FavFood	HireDate
1	Medical Employee	80000	PhD	Harvard	1	1	Interstellar	Movie	Pasta	1/1/2018
2	Medical Employee	120000	PhD	OSU	2	1	Gasoline	Song	Pizza	2/9/2018
3	Medical Employee	90000	PhD	Yale	3	1	Dawn FM	Album	Calzone	6/8/2018
4	Medical Employee	80000	PhD	OSU	4	1	Twin Peaks	Show	Gyro	2/27/2018
5	Medical Employee	120000	PhD	Michigan	5	1	Blade Runner	Movie	Chicken	1/25/2019
6	Medical Employee	90000	PhD	Harvard	6	2	Blade Runner 2049	Movie	Beef	1/25/2019
7	Medical Employee	80000	PhD	OSU	7	2	Here Comes The Sun	Song	Shrimp	5/16/2019
8	Medical Employee	120000	PhD	Yale	8	2	Graduation	Album	Salad	10/9/2018
9	Medical Employee	90000	PhD	OSU	9	2	808s and Heartbreak	Album	Dumplings	3/7/2019
10	Medical Employee	80000	PhD	Michigan	10	2	You	Show	Spring Rolls	10/10/2018
11	Non-medical employee	65000	BSE	Harvard	11	2	Better Call Saul	Show	Ramen	3/7/2019
12	Non-medical employee	70000	BSE	OSU	12	2	Walkin	Song	Wings	1/8/2019
13	Non-medical employee	80000	BSE	Yale	13	3	Cave World	Album	Breadsticks	9/5/2018
14	Non-medical employee	65000	BSE	OSU	14	4	The Matrix	Movie	Bacon	5/10/2019
15	Non-medical employee	70000	BSE	Michigan	15	5	Breaking Bad	Show	Sandwich	9/6/2019
16	Non-medical employee	80000	BSE	Harvard	16	6	Touch The Sky	Song	Popcorn	11/7/2018
17	Non-medical employee	65000	BSE	OSU	17	7	Electric Feel	Song	Burger	5/10/2019
18	Non-medical employee	70000	BSE	Yale	18	8	Stranger Things	Show	Steak	9/18/2019
19	Non-medical employee	80000	BSE	OSU	19	9	Inception	Movie	Fried Chicken	8/17/2019
20	Non-medical employee	65000	BSE	Michigan	20	10	John Wick	Movie	Soup	11/28/2018

Employee

Purpose: This table holds all employees.

Fields: PersonID, Employee_Type, Salary, Degree, College, BadgeNum, OID

Constraints:

PersonID: FOREIGN KEY

OID: FOREIGN KEY

Primary Key: N/A

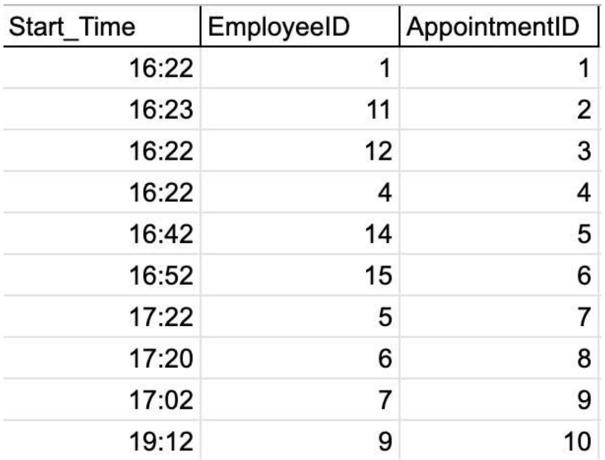
Foreign Key:

"PersonID" to Primary Key "ID" in Person;

"OID" to Primary Key "OID" in Office.

SQL Approved Data Types: INT, VARCHAR

(Table split into two halves for readability.)



Start_Time	EmployeeID	AppointmentID
16:22	1	1
16:23	11	2
16:22	12	3
16:22	4	4
16:42	14	5
16:52	15	6
17:22	5	7
17:20	6	8
17:02	7	9
19:12	9	10

Employee-Appointment-Join

Purpose: This table joins Employee and Appointment.

Fields: Start_Time, EmployeeID, AppointmentID

Constraints:

EmployeeID: FOREIGN KEY

AppointmentID: FOREIGN KEY

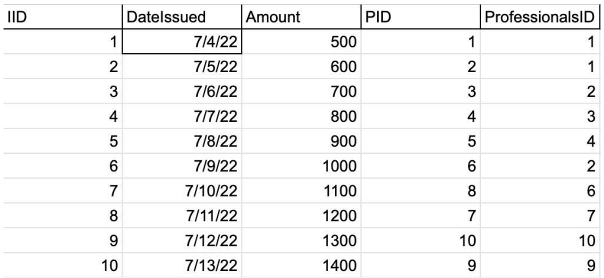
Primary Key: N/A

Foreign Key:

“EmployeeID” to Key “PersonID” in Employee;

“AppointmentID” to Primary Key “AppointmentID” in Appointment.

SQL Approved Data Types: TIME, INT



IID	DateIssued	Amount	PID	ProfessionalsID
1	7/4/22	500	1	1
2	7/5/22	600	2	1
3	7/6/22	700	3	2
4	7/7/22	800	4	3
5	7/8/22	900	5	4
6	7/9/22	1000	6	2
7	7/10/22	1100	8	6
8	7/11/22	1200	7	7
9	7/12/22	1300	10	10
10	7/13/22	1400	9	9

Invoice

Purpose: This table holds all invoices.

Fields: IID, DateIssued, Amount, PID, ProfessionalsID

Constraints:

IID: PRIMARY KEY

PID: FOREIGN KEY

ProfessionalsID: FOREIGN KEY

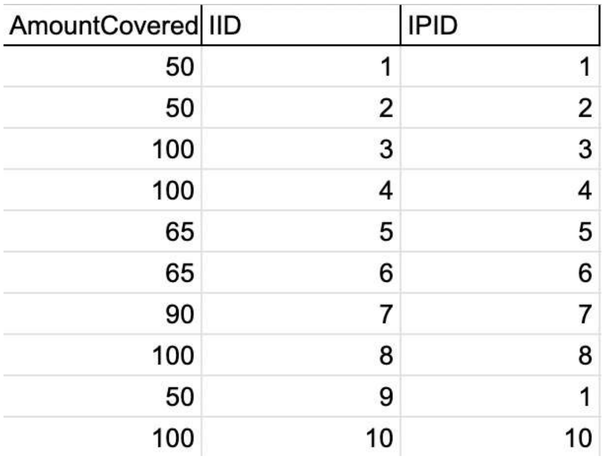
Primary Key: IID

Foreign Key:

“PID” to Primary Key “PID” in Payment;

“ProfessionalsID” to “PersonID” in MedicalProfessional.

SQL Approved Data Types: INT, DATE



AmountCovered	IID	IPID
50	1	1
50	2	2
100	3	3
100	4	4
65	5	5
65	6	6
90	7	7
100	8	8
50	9	1
100	10	10

Invoice-AcceptedInsurance-Join

Purpose: This table joins invoice and accepted insurance.

Fields: AmountCovered, IID, IPID

Constraints:

IID: FOREIGN KEY

IPID: FOREIGN KEY

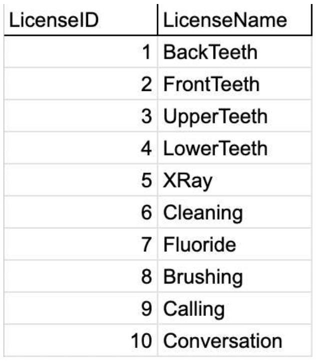
Primary Key: N/A

Foreign Key:

“IID” to Primary Key “IID” in Invoice;

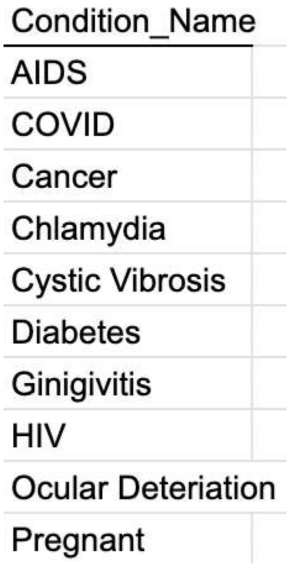
“IPID” to Primary Key “IPID” in AcceptedInsurance.

SQL Approved Data Types: INT



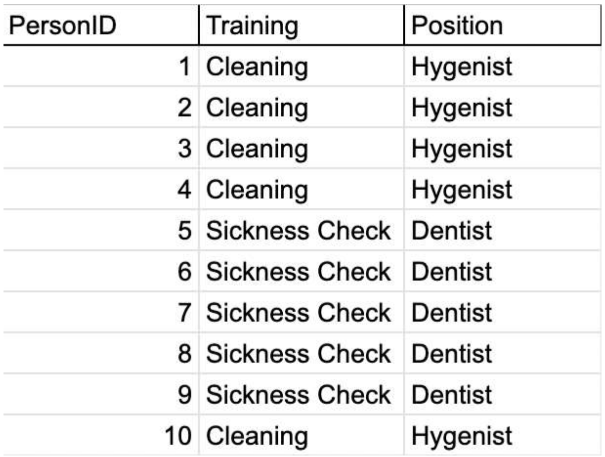
LicenseID	LicenseName
1	BackTeeth
2	FrontTeeth
3	UpperTeeth
4	LowerTeeth
5	XRay
6	Cleaning
7	Fluoride
8	Brushing
9	Calling
10	Conversation

License
Purpose: This table holds all dental licenses.
Fields: LicenseID, LicenseName
Constraints:
License: PRIMARY KEY
Primary Key: LicenseID
Foreign Key: N/A
SQL Approved Data Types: INT, VARCHAR



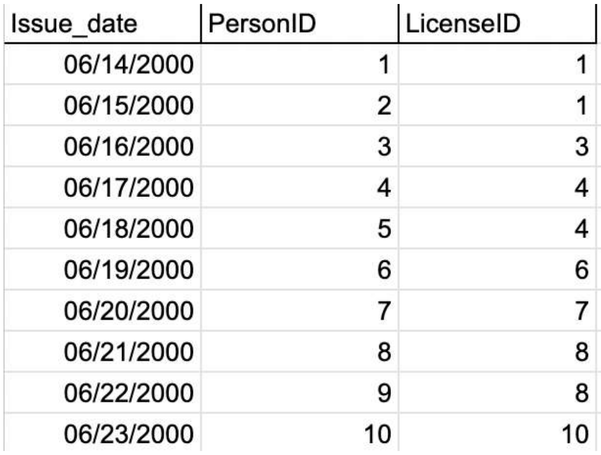
Condition_Name
AIDS
COVID
Cancer
Chlamydia
Cystic Vibrosis
Diabetes
Ginigivitis
HIV
Ocular Deteriation
Pregnant

MedicalCondition
Purpose: This table holds all medical conditions.
Fields: Condition_Name
Constraints:
Condition_Name: PRIMARY KEY
Primary Key: Condition_Name
Foreign Key: N/A
SQL Approved Data Types: VARCHAR



PersonID	Training	Position
1	Cleaning	Hygenist
2	Cleaning	Hygenist
3	Cleaning	Hygenist
4	Cleaning	Hygenist
5	Sickness Check	Dentist
6	Sickness Check	Dentist
7	Sickness Check	Dentist
8	Sickness Check	Dentist
9	Sickness Check	Dentist
10	Cleaning	Hygenist

MedicalEmployee
Purpose: This table holds all medical employees.
Fields: PersonID, Training, Position
Constraints:
PersonID: FOREIGN KEY
Primary Key: N/A
Foreign Key:
"PersonID" to Key "PersonID" in Employee
SQL Approved Data Types: INT, VARCHAR



Issue_date	PersonID	LicenseID
06/14/2000	1	1
06/15/2000	2	1
06/16/2000	3	3
06/17/2000	4	4
06/18/2000	5	4
06/19/2000	6	6
06/20/2000	7	7
06/21/2000	8	8
06/22/2000	9	8
06/23/2000	10	10

MedicalEmployee-License-Join

Purpose: This table joins medical employee and license.

Fields: Issue_date, PersonID, LicenseID

Constraints:

PersonID: FOREIGN KEY

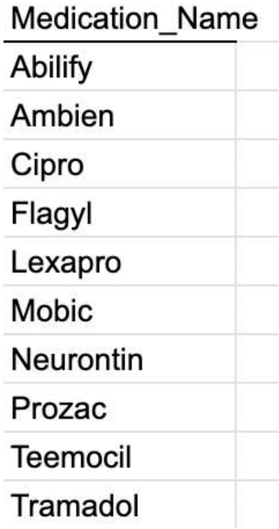
LicenseID: FOREIGN KEY

Primary Key: N/A

Foreign Key: “PersonID” to Key “PersonID” in MedicalEmployee;

“LicenseID” to Primary Key “LicenseID” in License.

SQL Approved Data Types: DATE, INT



Medication_Name
Abilify
Ambien
Cipro
Flagyl
Lexapro
Mobic
Neurontin
Prozac
Teemocil
Tramadol

Medication

Purpose: This table holds all medications provided.

Fields: Medication_Name

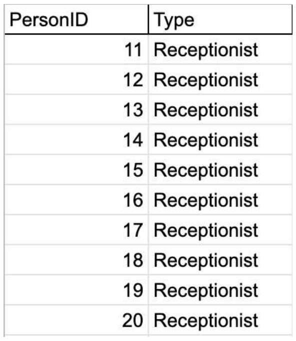
Constraints:

Medication_Name: PRIMARY KEY

Primary Key: Medication_Name

Foreign Key: N/A

SQL Approved Data Types: VARCHAR



PersonID	Type
11	Receptionist
12	Receptionist
13	Receptionist
14	Receptionist
15	Receptionist
16	Receptionist
17	Receptionist
18	Receptionist
19	Receptionist
20	Receptionist

NonMedicalEmployee

Purpose: This table holds all non-medical employees.

Fields: PersonID, Type

Constraints:

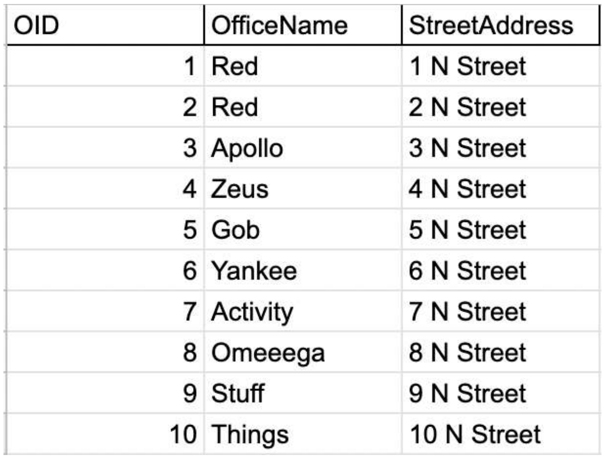
PersonID: FOREIGN KEY

Primary Key: N/A

Foreign Key:

“PersonID” to Key “PersonID” in Employee

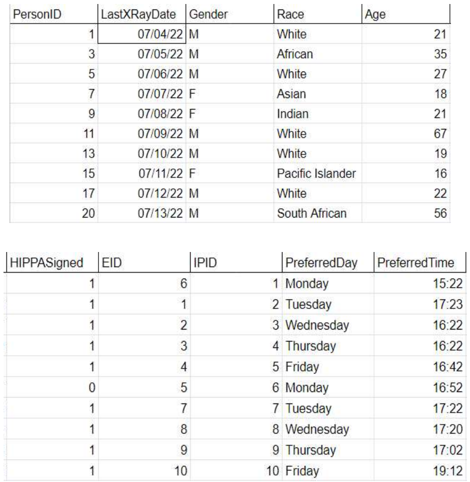
SQL Approved Data Types: INT, VARCHAR



OID	OfficeName	StreetAddress
1	Red	1 N Street
2	Red	2 N Street
3	Apollo	3 N Street
4	Zeus	4 N Street
5	Gob	5 N Street
6	Yankee	6 N Street
7	Activity	7 N Street
8	Omeeega	8 N Street
9	Stuff	9 N Street
10	Things	10 N Street

Office

Purpose: This table holds all dental offices.
Fields: OID, OfficeName, StreetAddress
Constraints:
OID: PRIMARY KEY
StreetAddress: FOREIGN KEY
Primary Key: OID
Foreign Key:
“StreetAddress” to Primary Key “StreetAddress” in Address
SQL Approved Data Types: INT, VARCHAR



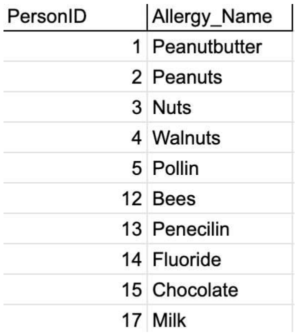
PersonID	LastXRayDate	Gender	Race	Age
1	07/04/22	M	White	21
3	07/05/22	M	African	35
5	07/06/22	M	White	27
7	07/07/22	F	Asian	18
9	07/08/22	F	Indian	21
11	07/09/22	M	White	67
13	07/10/22	M	White	19
15	07/11/22	F	Pacific Islander	16
17	07/12/22	M	White	22
20	07/13/22	M	South African	56

HIPPASigned	EID	IPID	PreferredDay	PreferredTime
1	6	1	Monday	15:22
1	1	2	Tuesday	17:23
1	2	3	Wednesday	16:22
1	3	4	Thursday	16:22
1	4	5	Friday	16:42
0	5	6	Monday	16:52
1	7	7	Tuesday	17:22
1	8	8	Wednesday	17:20
1	9	9	Thursday	17:02
1	10	10	Friday	19:12

(Table split into two halves for readability.)

Patient

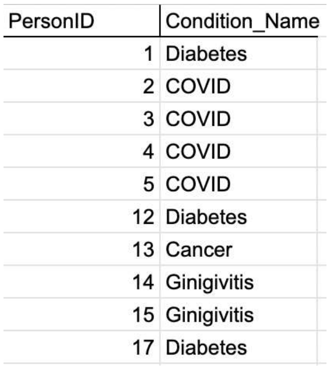
Purpose: This table holds all patients.
Fields: PersonID, LastXRayDate, Gender, Race, Age, HIPPASigned, EID, IPID, PreferredDay, PreferredTime
Constraints:
PersonID: FOREIGN KEY
EID: FOREIGN KEY
IPID: FOREIGN KEY
Primary Key: N/A
Foreign Key:
“PersonID” to Primary Key “ID” in Person;
“EID” to Key “PersonID” in EmergencyContact;
“IPID” to Primary Key “IPID” in AcceptedInsurance.
SQL Approved Data Types: INT, DATE, CHAR, VARCHAR



PersonID	Allergy_Name
1	Peanutbutter
2	Peanuts
3	Nuts
4	Walnuts
5	Pollin
12	Bees
13	Penecilin
14	Fluoride
15	Chocolate
17	Milk

Patient-Allergy-Join

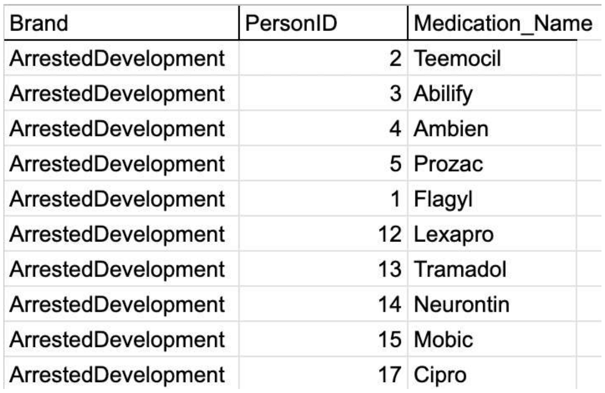
Purpose: This table joins patient and corresponding possible allergies.
Fields: PersonID, Allergy_Name
Constraints:
PatientID: FOREIGN KEY
AID: FOREIGN KEY
Primary Key: N/A
Foreign Key:
“PatientID” to Key “PersonID” in Patient;
“AID” to Primary Key “Allergy_Name” in Allergy.
SQL Approved Data Types: INT, VARCHAR



PersonID	Condition_Name
1	Diabetes
2	COVID
3	COVID
4	COVID
5	COVID
12	Diabetes
13	Cancer
14	Ginigivitis
15	Ginigivitis
17	Diabetes

Patient-MedicalCondition-Join

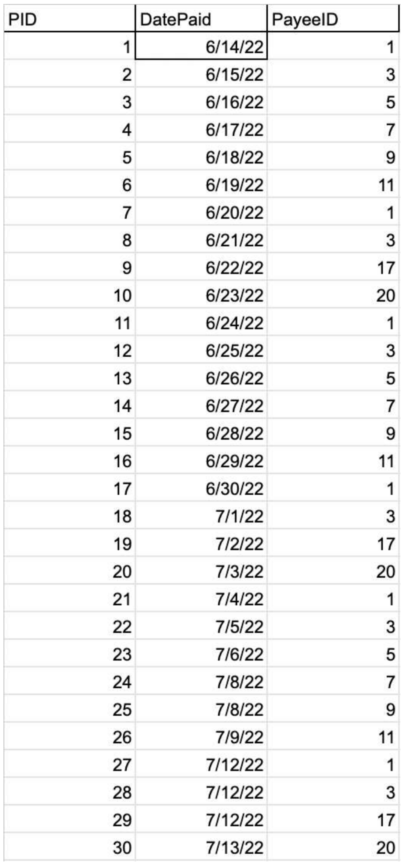
Purpose: This table joins patient and corresponding possible medical conditions.
Fields: PersonID, Condition_Name
Constraints:
PatientID: FOREIGN KEY
Condition_Name: FOREIGN KEY
Primary Key: N/A
Foreign Key:
"PatientID" to Key "PersonID" in Patient;
"Condition_Name" to Primary Key "Condition_Name" in MedicalCondition.
SQL Approved Data Types: INT, VARCHAR



Brand	PersonID	Medication_Name
ArrestedDevelopment	2	Teemocil
ArrestedDevelopment	3	Abilify
ArrestedDevelopment	4	Ambien
ArrestedDevelopment	5	Prozac
ArrestedDevelopment	1	Flagyl
ArrestedDevelopment	12	Lexapro
ArrestedDevelopment	13	Tramadol
ArrestedDevelopment	14	Neurontin
ArrestedDevelopment	15	Mobic
ArrestedDevelopment	17	Cipro

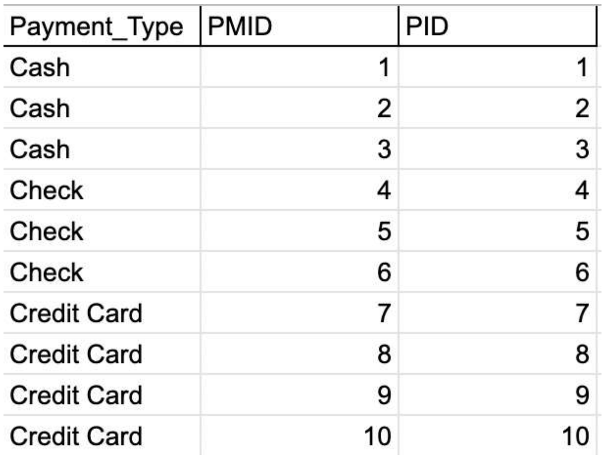
Patient-Medication-Join

Purpose: This table joins Patient and corresponding medications taken.
Fields: Brand, PersonID, Medication_Name
Constraints:
PatientID: FOREIGN KEY
Medication_Name: FOREIGN KEY
Primary Key: N/A
Foreign Key:
"PatientID" to Key "PersonID" in Patient;
"Medication_Name" to Primary Key "Medication_Name" in Medication.
SQL Approved Data Types: VARCHAR, INT



PID	DatePaid	PayeeID
1	6/14/22	1
2	6/15/22	3
3	6/16/22	5
4	6/17/22	7
5	6/18/22	9
6	6/19/22	11
7	6/20/22	1
8	6/21/22	3
9	6/22/22	17
10	6/23/22	20
11	6/24/22	1
12	6/25/22	3
13	6/26/22	5
14	6/27/22	7
15	6/28/22	9
16	6/29/22	11
17	6/30/22	1
18	7/1/22	3
19	7/2/22	17
20	7/3/22	20
21	7/4/22	1
22	7/5/22	3
23	7/6/22	5
24	7/8/22	7
25	7/8/22	9
26	7/9/22	11
27	7/12/22	1
28	7/12/22	3
29	7/12/22	17
30	7/13/22	20

Payment
Purpose: This table holds all payments made.
Fields: PID, DatePaid, PayeeID
Constraints:
PID: PRIMARY KEY
PayeeID: FOREIGN KEY
Primary Key: PID
Foreign Key:
"PayeeID" to Key "PersonID" in Patient
SQL Approved Data Types: INT, DATE

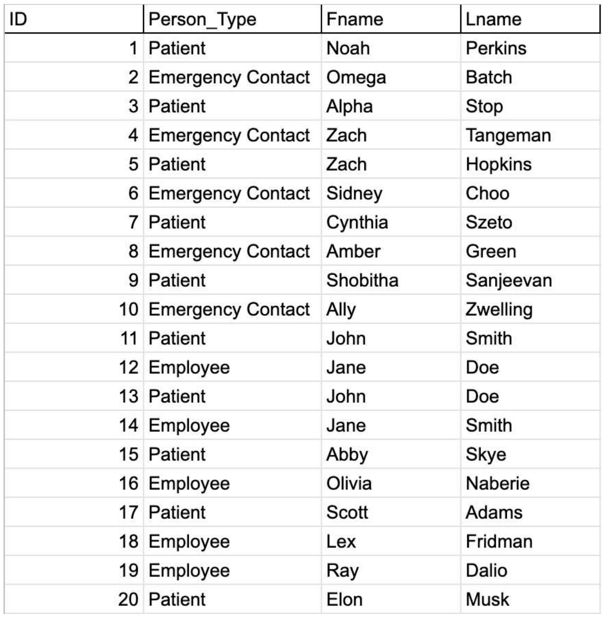


Payment_Type	PMID	PID
Cash	1	1
Cash	2	2
Cash	3	3
Check	4	4
Check	5	5
Check	6	6
Credit Card	7	7
Credit Card	8	8
Credit Card	9	9
Credit Card	10	10

PaymentMethod
Purpose: This table holds all possible payment methods.
Fields: Payment_Type, PMID, PID
Constraints:
PMID: PRIMARY KEY
PID: FOREIGN KEY
Primary Key: PMID
Foreign Key:
"PID" to Primary Key "PID" in Payment
SQL Approved Data Types: VARCHAR, INT

Person

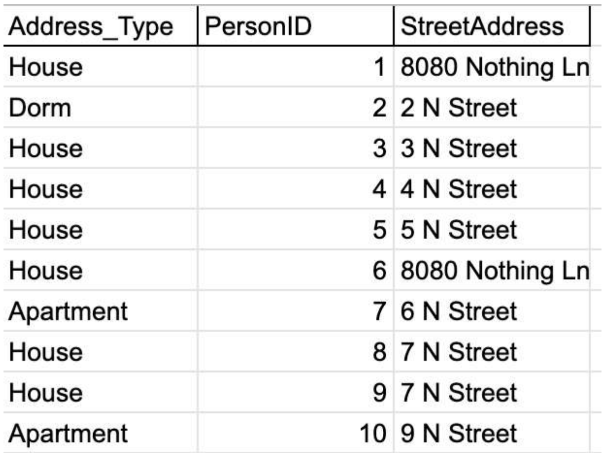
Purpose: This table holds all people that have relationship with the dental office.
Fields: ID, Person_Type, Fname, Lname
Constraints:
ID: PRIMARY KEY
Primary Key: ID
Foreign Key: N/A
SQL Approved Data Types: INT, VARCHAR



ID	Person_Type	Fname	Lname
1	Patient	Noah	Perkins
2	Emergency Contact	Omega	Batch
3	Patient	Alpha	Stop
4	Emergency Contact	Zach	Tangeman
5	Patient	Zach	Hopkins
6	Emergency Contact	Sidney	Choo
7	Patient	Cynthia	Szeto
8	Emergency Contact	Amber	Green
9	Patient	Shobitha	Sanjeevan
10	Emergency Contact	Ally	Zwelling
11	Patient	John	Smith
12	Employee	Jane	Doe
13	Patient	John	Doe
14	Employee	Jane	Smith
15	Patient	Abby	Skye
16	Employee	Olivia	Naberie
17	Patient	Scott	Adams
18	Employee	Lex	Fridman
19	Employee	Ray	Dalio
20	Patient	Elon	Musk

Person-Address-Join

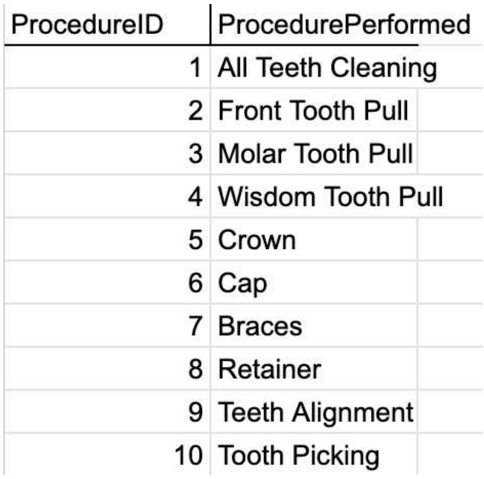
Purpose: This table joins person and address.
Fields: Address_Type, PersonID, StreetAddress
Constraints:
PersonID: FOREIGN KEY
StreetAddress: FOREIGN KEY
Primary Key: N/A
Foreign Key:
"PersonID" to Primary Key "ID" in Person;
"StreetAddress" to Primary Key "StreetAddress" in Address.
SQL Approved Data Types: VARCHAR, INT



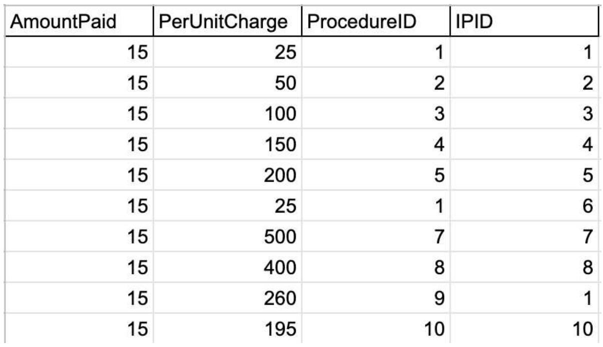
Address_Type	PersonID	StreetAddress
House	1	8080 Nothing Ln
Dorm	2	2 N Street
House	3	3 N Street
House	4	4 N Street
House	5	5 N Street
House	6	8080 Nothing Ln
Apartment	7	6 N Street
House	8	7 N Street
House	9	7 N Street
Apartment	10	9 N Street

Procedure

Purpose: This table holds all provided procedures in the dental office.
Fields: ProcedureID, ProcedurePerformed
Constraints:
ProcedureID: PRIMARY KEY
Primary Key: ProcedureID
Foreign Key: N/A
SQL Approved Data Types: INT, VARCHAR



ProcedureID	ProcedurePerformed
1	All Teeth Cleaning
2	Front Tooth Pull
3	Molar Tooth Pull
4	Wisdom Tooth Pull
5	Crown
6	Cap
7	Braces
8	Retainer
9	Teeth Alignment
10	Tooth Picking



AmountPaid	PerUnitCharge	ProcedureID	IPID
15	25	1	1
15	50	2	2
15	100	3	3
15	150	4	4
15	200	5	5
15	25	1	6
15	500	7	7
15	400	8	8
15	260	9	1
15	195	10	10

Procedure-Accepted-Insurance-Join

Purpose: This table joins procedure and corresponding accepted insurances.

Fields: AmountPaid, PerUnitCharge, ProcedureID, IPID

Constraints:

ProcedureID: FOREIGN KEY

IPID: FOREIGN KEY

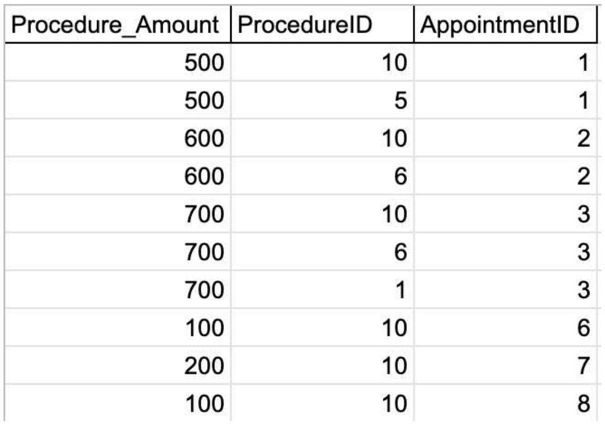
Primary Key: N/A

Foreign Key:

“ProcedureID” to Primary Key “ProcedureID” in Procedure;

“IPID” to Primary Key “IPID” in AcceptedInsurance.

SQL Approved Data Types: INT



Procedure_Amount	ProcedureID	AppointmentID
500	10	1
500	5	1
600	10	2
600	6	2
700	10	3
700	6	3
700	1	3
100	10	6
200	10	7
100	10	8

Procedure-Appointment-Join

Purpose: This table joins Procedure and corresponding appointment.

Fields: Procedure_Amount, ProcedureID, AppointmentID

Constraints:

ProcedureID: FOREIGN KEY

AppointmentID: FOREIGN KEY

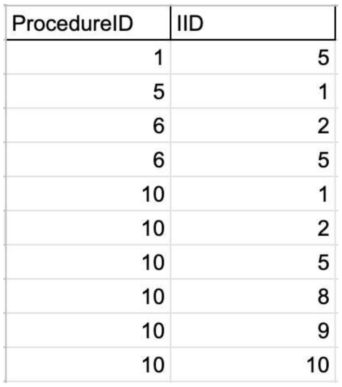
Primary Key: N/A

Foreign Key:

“ProcedureID” to Primary Key “ProcedureID” in Procedure;

“AppointmentID” to Primary Key “AppointmentID” in Appointment.

SQL Approved Data Types: INT



ProcedureID	IID
1	5
5	1
6	2
6	5
10	1
10	2
10	5
10	8
10	9
10	10

Procedure-Invoice-Join

Purpose: This table joins procedure and corresponding invoice.

Fields: ProcedureID, IID

Constraints:

ProcedureID: FOREIGN KEY

IID: FOREIGN KEY

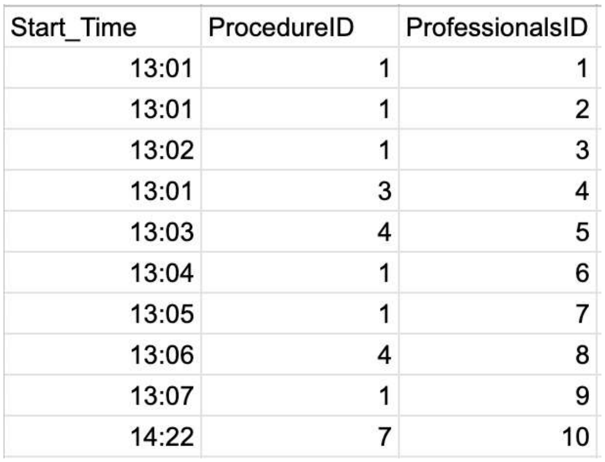
Primary Key: N/A

Foreign Key:

“ProcedureID” to Primary Key “ProcedureID” in Procedure;

“IID” to Primary Key “IID” in Invoice.

SQL Approved Data Types: INT



Start_Time	ProcedureID	ProfessionalsID
13:01	1	1
13:01	1	2
13:02	1	3
13:01	3	4
13:03	4	5
13:04	1	6
13:05	1	7
13:06	4	8
13:07	1	9
14:22	7	10

Procedure-MedicalProfessional-Join

Purpose: This table joins procedure and related medical professionals.

Fields: Start_Time, ProcedureID, ProfessionalsID

Constraints:

ProcedureID: FOREIGN KEY

ProfessionalsID: FOREIGN KEY

Primary Key: N/A

Foreign Key:

"ProcedureID" to Primary Key "ProcedureID" in Procedure;

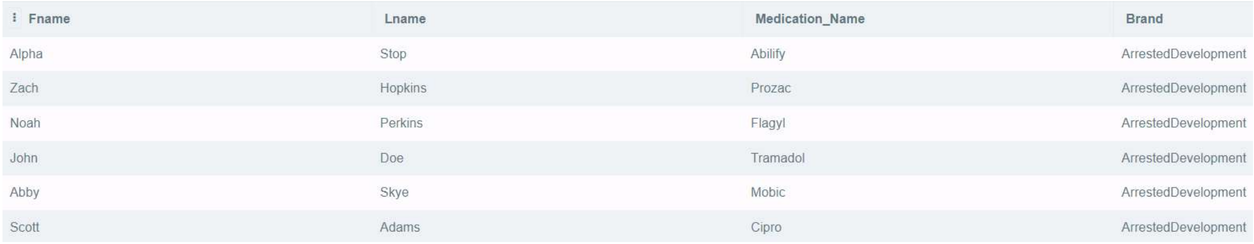
"ProfessionalsID" to Key "PersonID" in MedicalEmployee.

SQL Approved Data Types: TIME, INT

Catalog of SQL Queries

Simple Query 1: Create a list of patients and the medications they currently take.

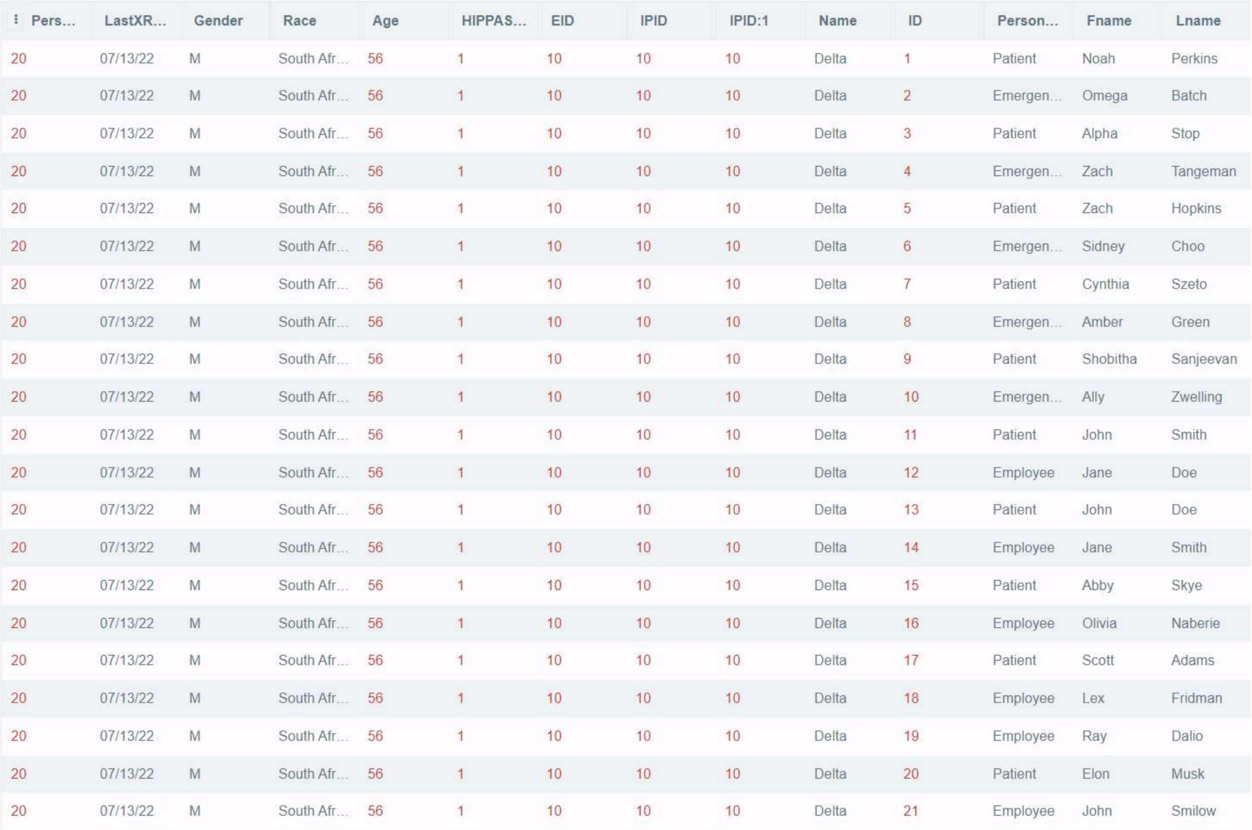
```
CREATE VIEW SQ1P1
AS  SELECT  *
    FROM    Person, Patient
    WHERE   Person.ID = Patient.PersonID;
CREATE VIEW SQ1P2
AS  SELECT  *
    FROM    SQ1P1, Patient_Medication_Join
    WHERE   SQ1P1.personid = Patient_Medication_Join.PersonID;
CREATE VIEW SQ1P3
AS  SELECT  Fname, Lname, SQ1P2.Medication_Name, Brand
    FROM    SQ1P2 , Medication
    WHERE   SQ1P2.Medication_Name = Medication.Medication_Name;
```



Fname	Lname	Medication_Name	Brand
Alpha	Stop	Abilify	ArrestedDevelopment
Zach	Hopkins	Prozac	ArrestedDevelopment
Noah	Perkins	Flagyl	ArrestedDevelopment
John	Doe	Tramadol	ArrestedDevelopment
Abby	Skye	Mobic	ArrestedDevelopment
Scott	Adams	Cipro	ArrestedDevelopment

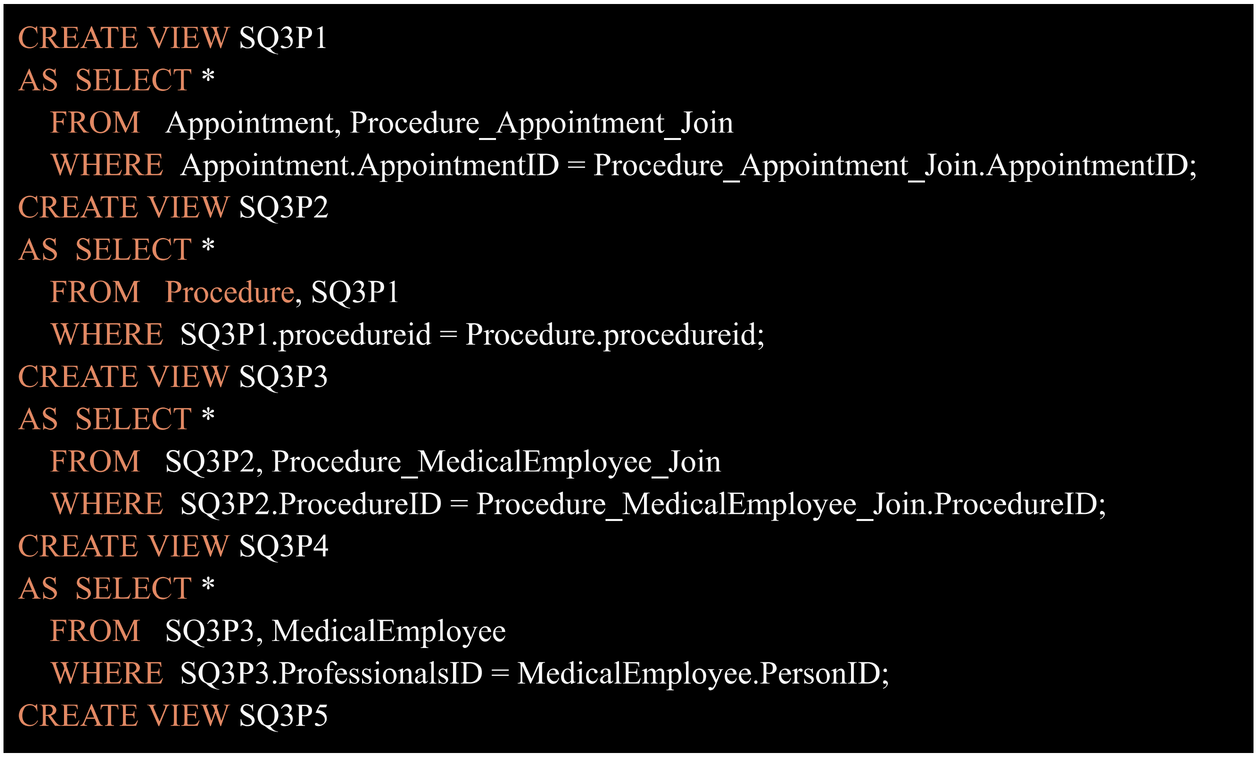
Simple Query 2: Display patient information for patients who currently have Delta Dental insurance policy.

```
CREATE VIEW SQ2P1
AS  SELECT IPID, Name
    FROM AcceptedInsurance;
CREATE VIEW SQ2P2
AS  SELECT *
    FROM   Patient, SQ2P1, Person
    WHERE  Patient.IPID = SQ2P1.IPID AND SQ2P1.Name = "Delta";
```



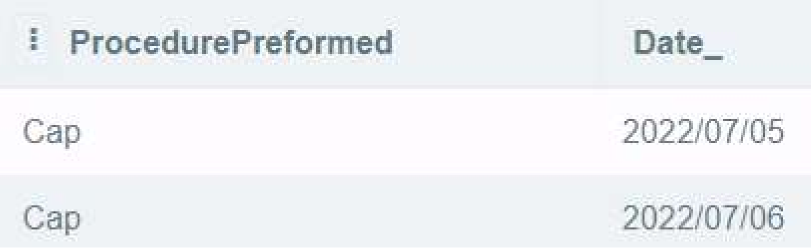
⁝ Pers...	LastXR...	Gender	Race	Age	HIPPAS...	EID	IPID	IPID:1	Name	ID	Person...	Fname	Lname
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	1	Patient	Noah	Perkins
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	2	Emergen...	Omega	Batch
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	3	Patient	Alpha	Stop
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	4	Emergen...	Zach	Tangeman
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	5	Patient	Zach	Hopkins
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	6	Emergen...	Sidney	Choo
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	7	Patient	Cynthia	Szeto
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	8	Emergen...	Amber	Green
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	9	Patient	Shobitha	Sanjeevan
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	10	Emergen...	Ally	Zwelling
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	11	Patient	John	Smith
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	12	Employee	Jane	Doe
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	13	Patient	John	Doe
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	14	Employee	Jane	Smith
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	15	Patient	Abby	Skye
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	16	Employee	Olivia	Naberie
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	17	Patient	Scott	Adams
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	18	Employee	Lex	Fridman
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	19	Employee	Ray	Dalio
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	20	Patient	Elon	Musk
20	07/13/22	M	South Afr...	56	1	10	10	10	Delta	21	Employee	John	Smilow

Simple Query 3: Generate a list of procedures and dates of service performed by doctor Smilow.



```
CREATE VIEW SQ3P1
AS  SELECT *
   FROM   Appointment, Procedure_Appointment_Join
   WHERE  Appointment.AppointmentID = Procedure_Appointment_Join.AppointmentID;
CREATE VIEW SQ3P2
AS  SELECT *
   FROM   Procedure, SQ3P1
   WHERE  SQ3P1.procedureid = Procedure.procedureid;
CREATE VIEW SQ3P3
AS  SELECT *
   FROM   SQ3P2, Procedure_MedicalEmployee_Join
   WHERE  SQ3P2.ProcedureID = Procedure_MedicalEmployee_Join.ProcedureID;
CREATE VIEW SQ3P4
AS  SELECT *
   FROM   SQ3P3, MedicalEmployee
   WHERE  SQ3P3.ProfessionalsID = MedicalEmployee.PersonID;
CREATE VIEW SQ3P5
```

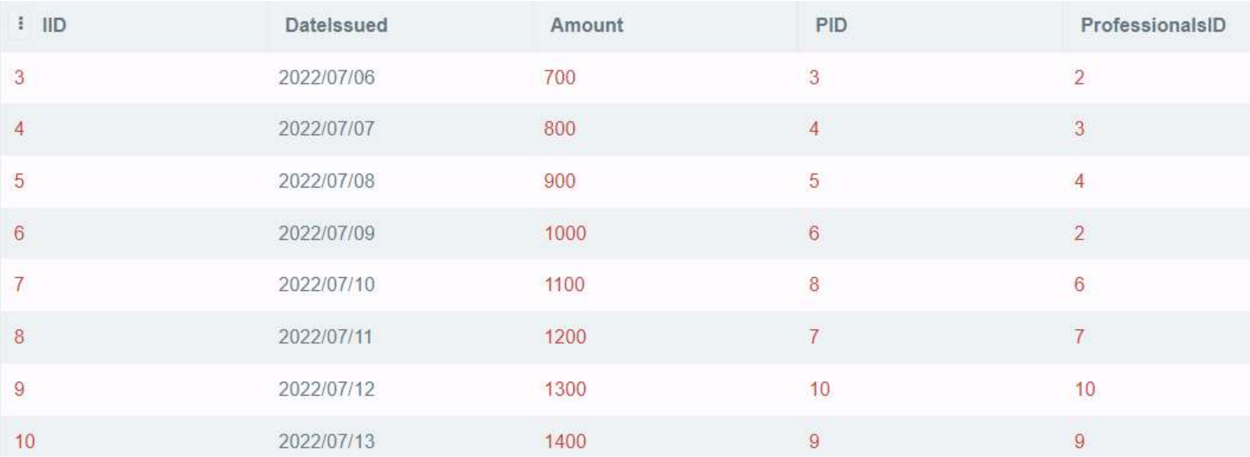
```
AS  SELECT *
   FROM   SQ3P4, Person
   WHERE  SQ3P4.personid = Person.ID;
CREATE VIEW SQ3P6
AS  SELECT ProcedurePreformed, Date_
         FROM   SQ3P5
   WHERE  Lname = "Smilow";
```



ProcedurePreformed	Date_
Cap	2022/07/05
Cap	2022/07/06

Simple Query 4: 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.

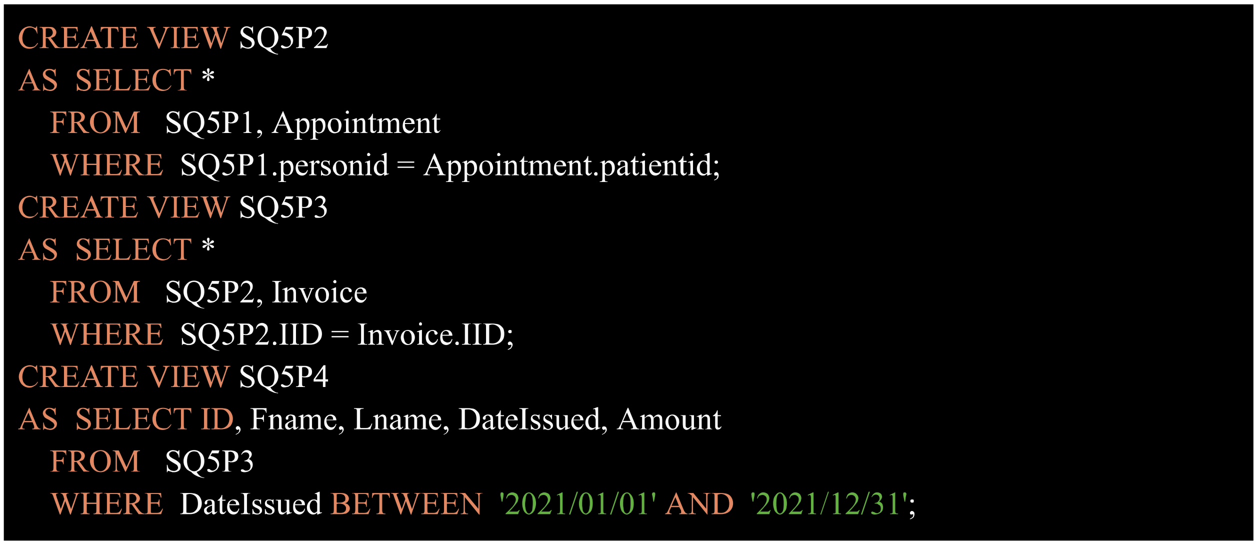
```
CREATE VIEW SQ4P1
AS SELECT *
   FROM   Invoice
   WHERE  DateIssued > '2022/06/30' AND Amount > 600;
```



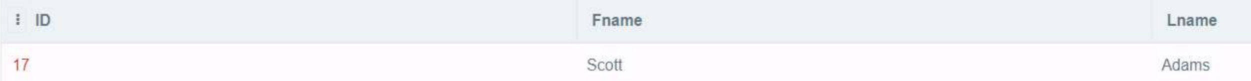
IID	DateIssued	Amount	PID	ProfessionalsID
3	2022/07/06	700	3	2
4	2022/07/07	800	4	3
5	2022/07/08	900	5	4
6	2022/07/09	1000	6	2
7	2022/07/10	1100	8	6
8	2022/07/11	1200	7	7
9	2022/07/12	1300	10	10
10	2022/07/13	1400	9	9

Simple Query 5: Find the patients who brought the most revenue in the past year.

```
CREATE VIEW SQ5P1
AS  SELECT *
   FROM   Person, Patient
   WHERE  Patient.PersonID = Person.ID;
```

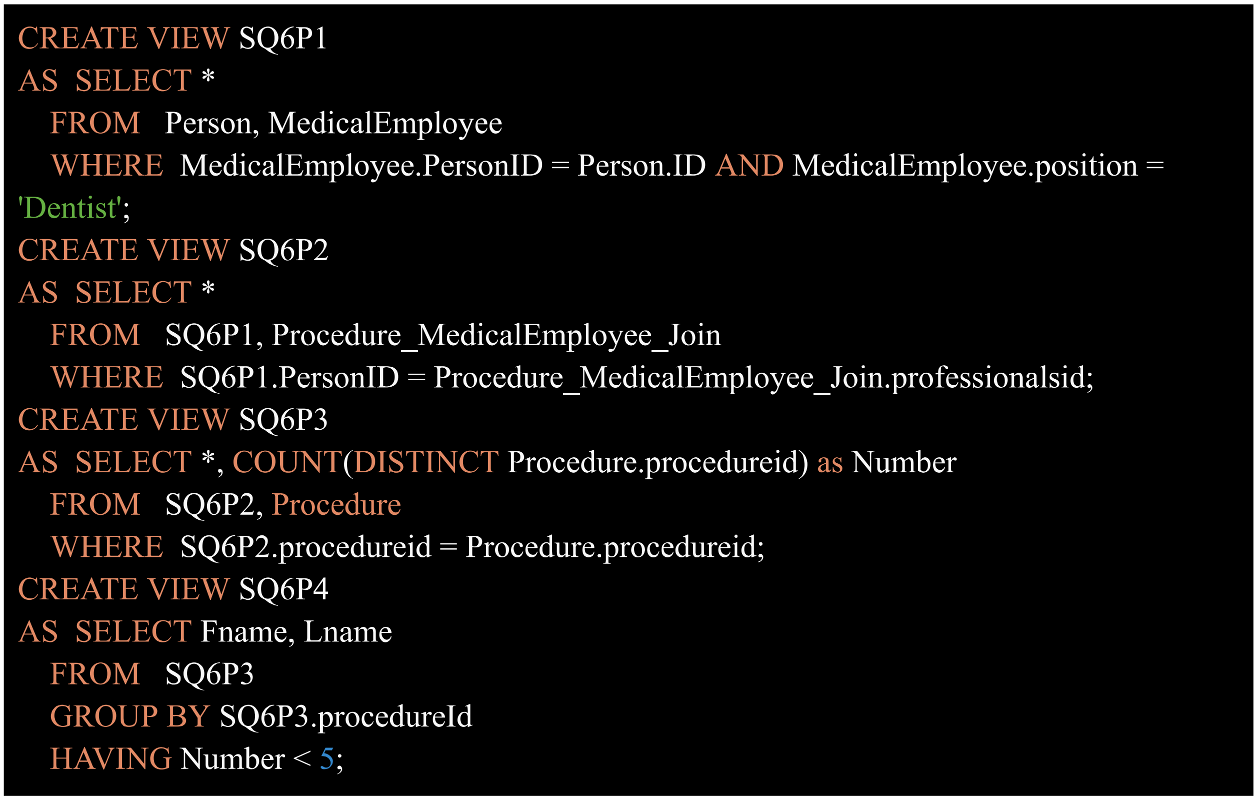


```
CREATE VIEW SQ5P2
AS  SELECT *
   FROM   SQ5P1, Appointment
   WHERE  SQ5P1.personid = Appointment.patientid;
CREATE VIEW SQ5P3
AS  SELECT *
   FROM   SQ5P2, Invoice
   WHERE  SQ5P2.IID = Invoice.IID;
CREATE VIEW SQ5P4
AS  SELECT ID, Fname, Lname, DateIssued, Amount
   FROM   SQ5P3
   WHERE  DateIssued BETWEEN  '2021/01/01' AND  '2021/12/31';
```

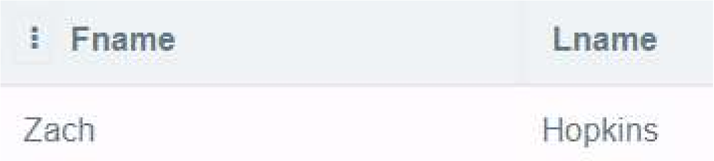


⁝ ID	Fname	Lname
17	Scott	Adams

Simple Query 6: Create a list of doctors who performed less than 5 procedures this year.

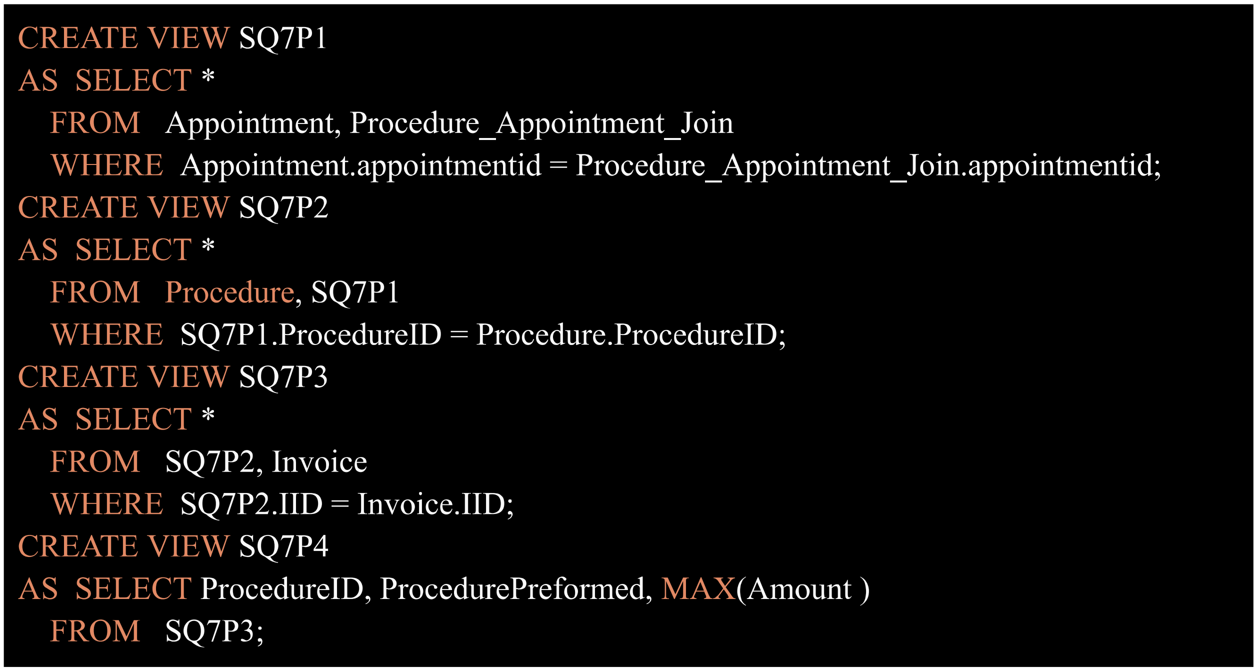


```
CREATE VIEW SQ6P1
AS  SELECT *
   FROM   Person, MedicalEmployee
   WHERE  MedicalEmployee.PersonID = Person.ID AND MedicalEmployee.position =
'Dentist';
CREATE VIEW SQ6P2
AS  SELECT *
   FROM   SQ6P1, Procedure_MedicalEmployee_Join
   WHERE  SQ6P1.PersonID = Procedure_MedicalEmployee_Join.professionalsid;
CREATE VIEW SQ6P3
AS  SELECT *, COUNT(DISTINCT Procedure.procedureid) as Number
   FROM   SQ6P2, Procedure
   WHERE  SQ6P2.procedureid = Procedure.procedureid;
CREATE VIEW SQ6P4
AS  SELECT Fname, Lname
   FROM   SQ6P3
   GROUP BY SQ6P3.procedureId
   HAVING Number < 5;
```

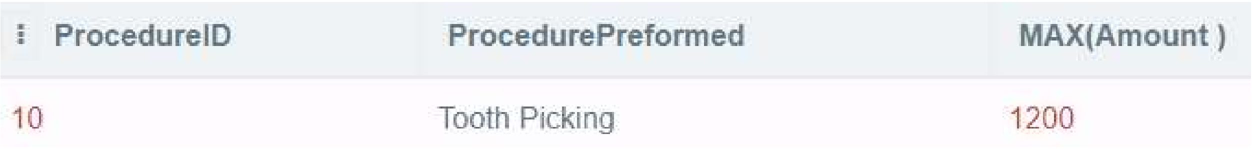


Fname	Lname
Zach	Hopkins

Simple Query 7: Find the highest paying procedures, procedure price, and the total number of those procedures performed.

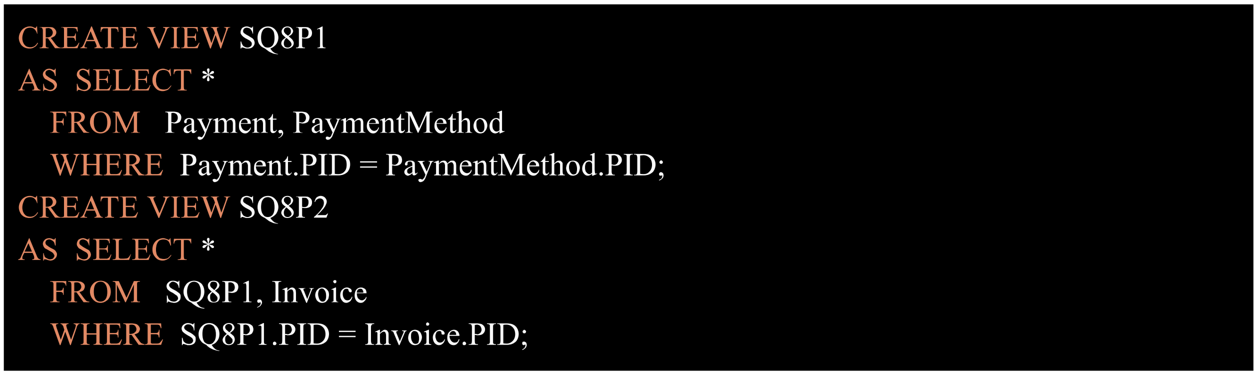


```
CREATE VIEW SQ7P1
AS  SELECT *
   FROM   Appointment, Procedure_Appointment_Join
   WHERE  Appointment.appointmentid = Procedure_Appointment_Join.appointmentid;
CREATE VIEW SQ7P2
AS  SELECT *
   FROM   Procedure, SQ7P1
   WHERE  SQ7P1.ProcedureID = Procedure.ProcedureID;
CREATE VIEW SQ7P3
AS  SELECT *
   FROM   SQ7P2, Invoice
   WHERE  SQ7P2.IID = Invoice.IID;
CREATE VIEW SQ7P4
AS  SELECT ProcedureID, ProcedurePreformed, MAX(Amount )
   FROM   SQ7P3;
```



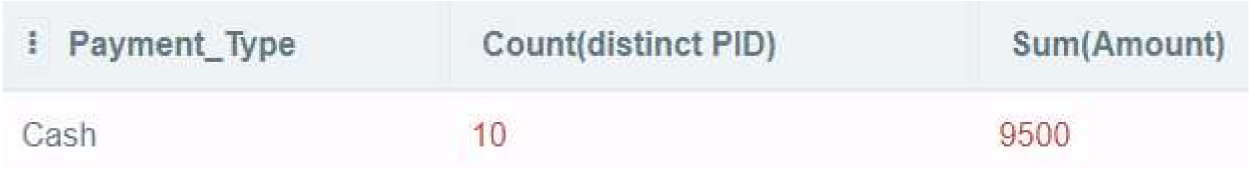
ProcedureID	ProcedurePreformed	MAX(Amount)
10	Tooth Picking	1200

Simple Query 8: Create a list of all payment types accepted, the number of times each of them was used, and the total amount charged to that type of payment.



```
CREATE VIEW SQ8P1
AS  SELECT *
   FROM   Payment, PaymentMethod
   WHERE  Payment.PID = PaymentMethod.PID;
CREATE VIEW SQ8P2
AS  SELECT *
   FROM   SQ8P1, Invoice
   WHERE  SQ8P1.PID = Invoice.PID;
```

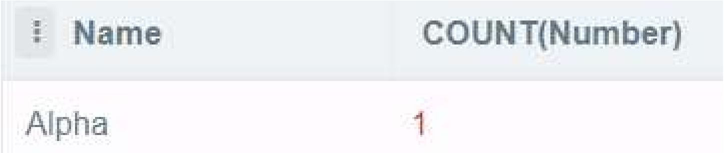
```
CREATE VIEW SQ8P3
AS  SELECT Payment_type, Count(distinct PID), Sum(Amount)
    FROM   SQ8P2;
```



Payment_Type	Count(distinct PID)	Sum(Amount)
Cash	10	9500

Simple Query 9: Find the name of the most popular insurance plan currently used by the patients.

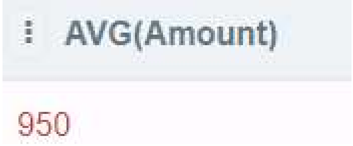
```
CREATE VIEW SQ9P1
AS  SELECT *
    FROM   Patient, AcceptedInsurance
    WHERE  Patient.IPID = AcceptedInsurance.IPID;
CREATE VIEW SQ9P2
AS  SELECT SQ9P1.Name, COUNT(distinct SQ9P1.IPID) as Number
    FROM   SQ9P1;
CREATE VIEW SQ9P3
AS  SELECT SQ9P2.Name, COUNT(Number)
    FROM   SQ9P2;
```



Name	COUNT(Number)
Alpha	1

Extra Query 1: Average payment of uncanceled appointments.

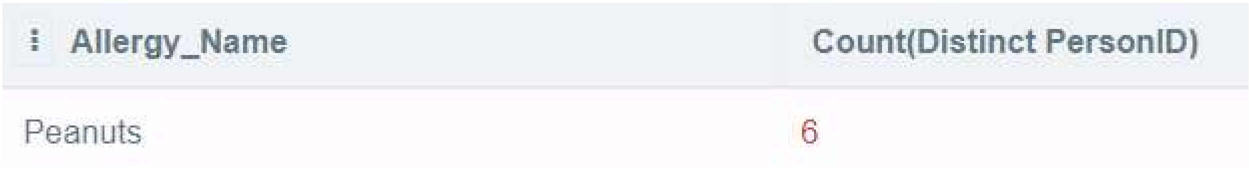
```
CREATE VIEW EXQ1
AS SELECT AVG(Amount) FROM (SELECT * FROM (SELECT * FROM Patient,
Appointment), Invoice)
WHERE Cancelled = FALSE;
```



AVG(Amount)
950

Extra Query 2: Patient Count of Type of Allergies.

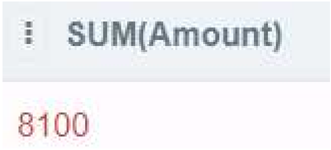
```
CREATE VIEW EXQ2
AS SELECT Allergy.allergy_name, Count(Distinct PersonID)
FROM (SELECT * FROM Patient, Patient_Allergy_Join
WHERE Patient.PersonID = Patient_Allergy_Join.PersonID), Allergy
WHERE Allergy.allergy_name = Allergy.allergy_name;
```



Allergy_Name	Count(Distinct PersonID)
Peanuts	6

Extra Query 3: Total paid in the year 2021.

```
CREATE VIEW EXQ3
AS SELECT SUM(Amount) FROM Invoice WHERE DateIssued BETWEEN '2022/01/01'
AND '2022/12/31';
```

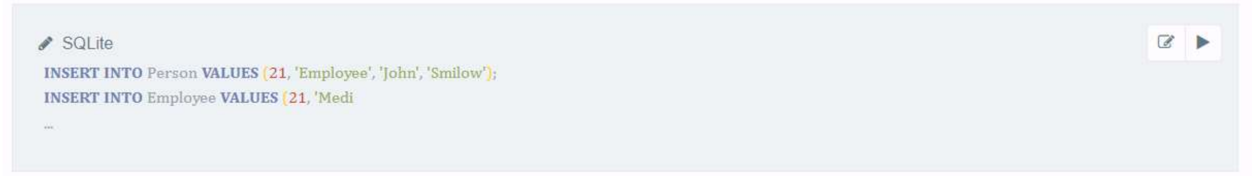


SUM(Amount)
8100

Insert and Delete SQL Examples

Inserting new doctor Smilow.

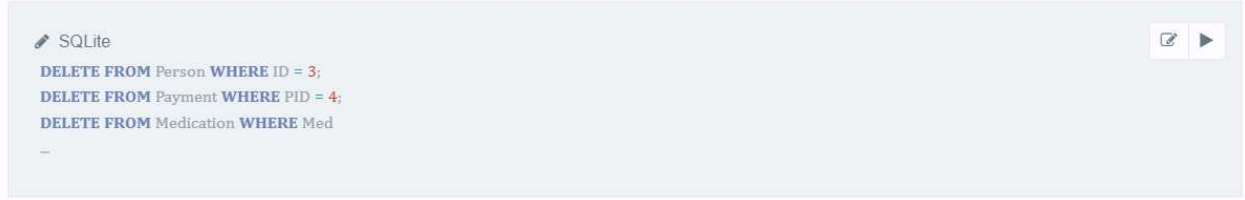
```
INSERT into Person VALUES (21, 'Employee', 'John', 'Smilow');
INSERT into Employee VALUES (21, 'Medical Employee', 80000, 'PhD', 'Harvard', 1, 1);
INSERT into MedicalEmployee VALUES (21, 'Everything', 'Doctor');
INSERT into Procedure_MedicalEmployee_Join VALUES ('16:22', 6, 21);
```



```
INSERT INTO Person VALUES (21, 'Employee', 'John', 'Smilow');
INSERT INTO Employee VALUES (21, 'Medi
...
```

Deleting the person with ID = 3, Payment where PID = 4, and the medication where Medication_Name = ‘Ambien’ while cascade deleting all subclasses and join tables.

```
DELETE FROM Person WHERE ID = 3;
DELETE FROM Payment WHERE PID = 4;
DELETE FROM Medication WHERE Medication_Name = 'Ambien';
```

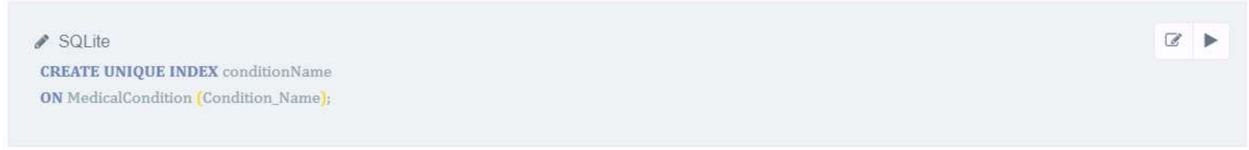


```
DELETE FROM Person WHERE ID = 3;
DELETE FROM Payment WHERE PID = 4;
DELETE FROM Medication WHERE Med
...
```

Two Indexes

Make a cluster index for the condition name on MedicalCondition.

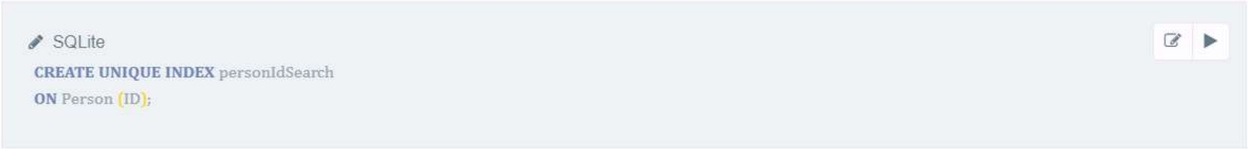
```
CREATE UNIQUE INDEX conditionName
ON MedicalCondition (Condition_Name);
```



```
CREATE UNIQUE INDEX conditionName
ON MedicalCondition (Condition_Name);
```

Make a cluster index for the id on all persons

```
CREATE UNIQUE INDEX personIdSearch
ON Person (ID);
```

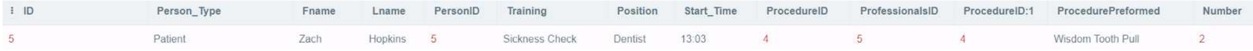


```
CREATE UNIQUE INDEX personIdSearch
ON Person (ID);
```

Two Views

Gets the combined columns of Person, MedicalEmployee, Procedure_MedicalEmployee_Join, and then counts the distinct procedure ids to find the number of procedures that have been performed.

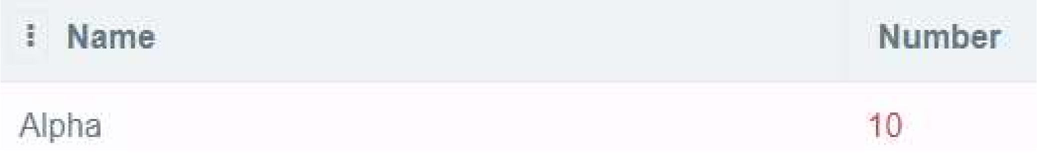
```
CREATE VIEW SQ6P1
AS  SELECT *
   FROM   Person, MedicalEmployee
   WHERE  MedicalEmployee.PersonID = Person.ID AND MedicalEmployee.position =
'Dentist';
CREATE VIEW SQ6P2
AS  SELECT *
   FROM   SQ6P1, Procedure_MedicalEmployee_Join
   WHERE  SQ6P1.PersonID = Procedure_MedicalEmployee_Join.professionalsid;
CREATE VIEW SQ6P3
AS  SELECT *, COUNT(DISTINCT Procedure.procedureid) as Number
   FROM   SQ6P2, Procedure
   WHERE  SQ6P2.procedureid = Procedure.procedureid;
```



ID	Person_Type	Fname	Lname	PersonID	Training	Position	Start_Time	ProcedureID	ProfessionalsID	ProcedureID:1	ProcedurePreformed	Number
5	Patient	Zach	Hopkins	5	Sickness Check	Dentist	13:03	4	5	4	Wisdom Tooth Pull	2

Gets the number of people who are using insurance plans

```
CREATE VIEW SQ9P1
AS  SELECT *
   FROM   Patient, AcceptedInsurance
   WHERE  Patient.IPID = AcceptedInsurance.IPID;
CREATE VIEW SQ9P2
AS  SELECT SQ9P1.Name, COUNT(distinct SQ9P1.IPID) as Number
   FROM   SQ9P1;
```



Name	Number
Alpha	10

Two Transactions

Transaction I

Purpose: Adding a new appointment for a new patient.
It is important to execute these in a single unit because this transaction involves multiple tables. In this case, an appointment must relate to a patient. We have to add all corresponding records to make sure every related table has an up-to-date data stored.

```
BEGIN TRANSACTION;
    INSERT OR ROLLBACK INTO Person values (22, 'Patient', 'Zina',
'Pichkar');
    INSERT OR ROLLBACK INTO Patient VALUES (22, 'Tuesday', '10:00',
'07/13/22', 'M', 'White', 23, 1, 9, 1);
    INSERT OR ROLLBACK INTO Appointment VALUES (11, '7/14/22', 0, 18, 11);

COMMIT;
```

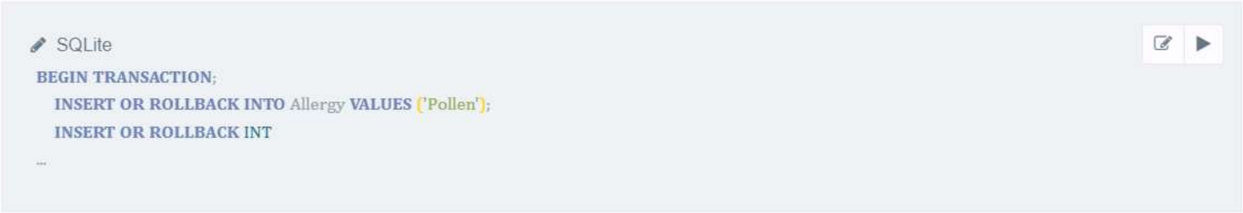
```
BEGIN TRANSACTION;
  INSERT OR ROLLBACK INTO Person VALUES (22, 'Patient', 'Zina', 'Pichkar');
  IN
...
```

Transaction II

Purpose: Adding a new allergy to an existing patient.
It is important to execute these in a single unit because this transaction involves multiple tables. In this case, an allergy must relate to an existing patient. We have to add all corresponding records to make sure every related table has an up-to-date data stored.

```
BEGIN TRANSACTION;
    INSERT OR ROLLBACK INTO Allergy VALUES ('Pollen');
    INSERT OR ROLLBACK INTO Patient_Allergy_Join VALUES (22, 'Pollen');

COMMIT;
```



```
BEGIN TRANSACTION;
  INSERT OR ROLLBACK INTO Allergy VALUES ('Pollen');
  INSERT OR ROLLBACK INT
...
```

Section 3 - Team Reports and Graded Checkpoint Documents

Team Member Contributions

Aaron Post worked on the (E)ERD, relational schema and diagram, introduction, documentation, and formatted the final report document. Noah Perkins did the majority of our SQL code, worked on the relational schema, populated the database, made the create, insert, and delete queries, and helped work on other various aspects earlier on in the project. Keyang Zhang worked on the (E)ERD, table descriptions, and transactions. Overall, we believe the three of us contributed our respective fair shares of work. While Saeed Alneyadi did complete the relational algebra for the second checkpoint, he did not maintain those queries to work with the updated schema and ERD between checkpoints. He was also very inactive for almost all of the semester and very hard to reach.

Project Reflection

This process was very challenging. For many of us, this was our first big semester-long group project. The most important advice we would have for future groups is to really focus on getting each checkpoint right the first time; the more mistakes you make, the more challenging each further checkpoint will be. The vertical development of a relational database means that mistakes will have to be corrected in many different areas. Even for our final report, we were still having to go back and make changes to the ERD, then the schema, then the schema diagram, then the relational algebra, and so on. This process can easily become a headache, so work really hard, in the beginning, to make things easier later on. Also, for those who are taking this course over the summer, don’t underestimate the importance of creating a schedule for working together. Many of us still had obligations such as jobs or other classes which we had to plan around.

Feedback and Revision Process

CP01 Feedback:

- ERD needs a bit more work.
- Consider using appropriate generalizations/specializations.
- Remove duplication of atteributes. Identify all derived attributes.
- Remove job type-based Union. Entities without attributes should not be present (empty sets).
- Make sure that the invoice (billing) is connected with payments, insurance, procedures.

Revisions Made:

- Remade entire ERD
    - Removed Job-Based Union, but added other generalizations.
    - Removed duplicate attributes.
    - “Fixed Payments” though we completely changed the payment system later on.

CP02 Feedback:

- Payment is missing relationships and does not handle payment methods.
- Insurance companies are not considered.
- Missing cardinality, and some relationships have incorrect cardinality.
- Relational schema was missing many Foreign Keys.
- Queries had many issues, most of which we should have caught by checking back over our work.

Revisions Made:

- Overhauled the payment system, adding necessary relationships and a new entity to handle the various payment methods.
- Overhauled the Insurance system, adding necessary relationships and the new Insurance Company entity.
- Added missing cardinality and fixed incorrect cardinality.
- “Fixed queries,” but these still had to be fixed at a later date because they were still not entirely correct.

CP03 Feedback:

- Patient, Employee and Emergency contact can be the same person.
- It is better to assign Emergency Contacts via relationship of Person with itself (self-join). Use 'type' attributes for Generaliztion/specialization cases.
- Use meaningful names for PK instead of simple 'ID'.
- Need attributes to describe M:N relationships such as Patient:Allergy such as severety, date_occured, etc.
- Relation between patient and an attribute of Credit Card?
- Refer back to CP02 feedback on PAYMENT and INVOICE entities and relationship between them.
- Show your final schema using sentence notation as learned in class.
- Many queries still have basic syntax errors and don’t work properly. Cascading needs to added. Generally, these queries still need many changes to function correctly.

Revisions Made:

- Changed specialization of Patient, Employee, and Emergency Contact to overlap.
- Added necessary “type” attributes for specialization.
- Added self-joins.
- Changed many Primary Key names.
- Added many fitting attributes for M:N relationships.
- Removed unnecessary relationship between patient and credit card
- Fixed payment and invoice relationship
- Overhauled relational schema to be in sentence notation, as well as adding a relational diagram.
- Queries were overhauled to work correctly, and necessary cascading was implemented.

Project Checkpoints

All previous CP documents are included–see "CP01.PDF" "CP02.PDF" "CP03.PDF"

Resources:

Daniel Post (Aaron's father who is an admin at a hospital)

"Best Dental Software with Customer Database." *GetApp*,

www.getapp.com/healthcare-pharmaceuticals-software/dental/f/customer-database.

Accessed 31 May 2022.

Uzialko, Adam. "What Is Data Management?" *Business.Com*, 16 Feb. 2022,

www.business.com/articles/what-is-data-management.

Part II - The SQL Database

Testing Queries and SQL

To test our queries and database design, we used SQLLite at https://sqliteonline.com/. All necessary files to use our database are included in this submission.