**COMP 353/453 Project Phase 3**

Deliverables:

* [Revised] ER diagram with min/max specifications
* [Revised] Relational Schema
* ORM Implementation

Assessment:

* Group status report

Phase Leader Sathvik Maridasana Nagaraj \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Phase Recorder Aaron Myrold \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Phase Checker Crisi Patelis \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Technical Advisor Michael Saban \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Introduction**

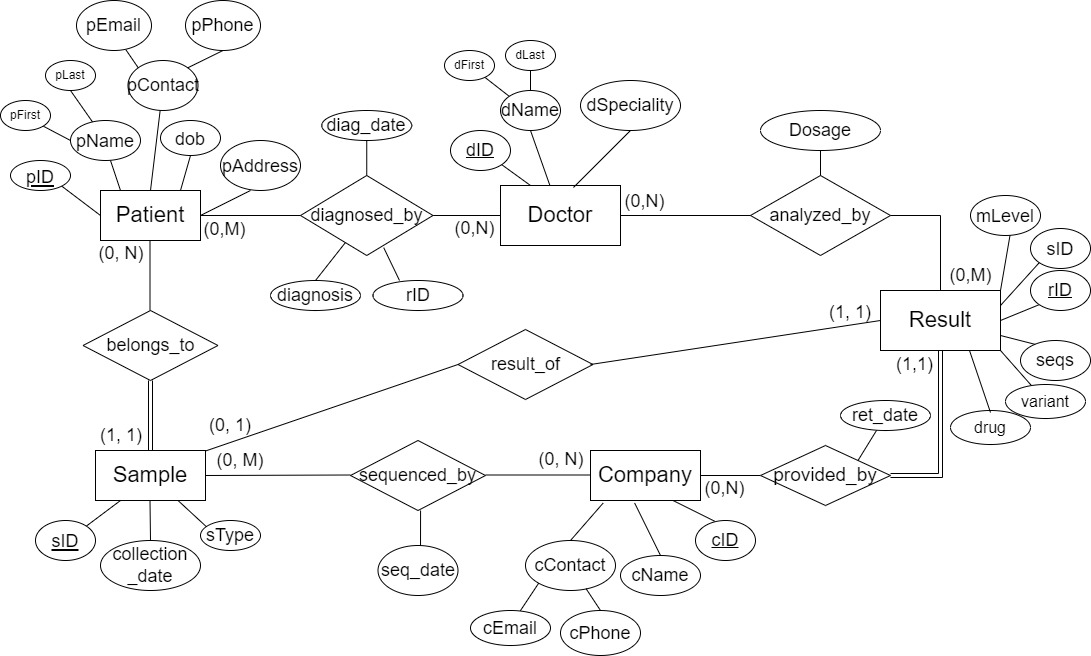
The database is based on a tool that Nick Miller has worked on called Flype. It consists of a relational database, in-house developed bioinformatics analysis pipelines, a connection framework to connect to external tools, and a web interface to provide easy and efficient access to the information stored in the database. The relational database stores things like patient information, sample and result information, result interpretation, and other information needed to provide quick and personalized healthcare.

1. **Description of the Organization**

The Healthcare database stores patient information along with their data related to test results such as Next Generation Sequencing (NGS) and Pharmacogenomic (PGX) testing. Maintaining a relational database that not only holds all patient information and results, but also interpretation of said results provides a quick and efficient way for healthcare providers to incorporate computational biology and genetic testing into a more personalized treatment for patients.

* Patients require genetic testing for a variety of illnesses. This involves collecting the correct samples and sending them off for sequencing. To efficiently track this, patients are given a unique patient ID along with storing their names, birth dates, and addresses. Patients may have multiple doctors working with them at one time or none at all.
* Over time, patients may require multiple different tests, so each sample includes a unique sample ID, a collection date, and information on the type of sample collected. Patients can have multiple samples taken, but each sample can only belong to one patient.
* A single collected sample may be sequenced by multiple companies, so contact information including the company name, email, and phone number are included along with a unique company ID. This allows tracking of communications with each company and when and where samples are sent. Every individual company may be sequencing multiple samples and the sequencing dates are stored in the relationship between the company and the sample.
* The results returned by the company are associated with the sample and patient IDs and include not only the sequencing results, but also identified and annotated gene variants of the sequenced samples with corresponding metabolism levels of a given drug based on those variants. A result can only belong to one company, but a company may send multiple results back. The sequencing return dates are stored in the relationship between the company and the result.
* While sequencing results contain annotation, a healthcare professional needs to analyze/interpret and relay that information to the patient’s electronic health record (EHR). Doctors’ names and specialties are stored under a unique ID and a diagnosis date is stored in the relationship between the doctor and the patient. Doctors can have multiple patients or none at all.

1. **ER Diagram**



1. **ER Diagram Uncaptured Constraints**

* Sequence strings in Results must be uppercase
* Address format: Street Number, followed by Direction (if applicable), then by Street Name(s)
  + No abbreviations such as St, Ave, N, S, etc.
  + Number, direction, and each name separated by a space
* Name format:
  + Patient and Doctor names: first name beginning with capital letter (pFirst/dFirst), last name beginning with a capital letter (pLast, dLast)
  + Company names:
* Date data types use the following format: YYYY-MM-DD
* Phone numbers use following format: XXX-XXX-XXXX
* ID format:
  + Patient IDs (pID): begin with “P” followed by three-digit number
  + Sample IDs (sID): begin with “S” followed by three-digit number
  + Result IDs (rID): begin with “R” followed by three-digit number
  + Doctor IDs (dID): begin with “D” followed by three-digit number
* Doctor Analysis
  + Dosage numerical values in mg
  + NULL represents incompatibility of drug with patient

1. **Relational Schema**

**5.1 Relational Schema with Referential Integrity**

Patient(pID,pFirst,pLast,pDOB,pAddress,pEmail,pPhone)

Sample(sID,collection\_date,sType,pID)

Foreign key (pID) references Patient (pID)

Sequenced\_By(cID,sID,seq\_date)

Foreign key (cID) references Company (cID)

Foreign key (sID) references Sample (sID)

Company(cID,cName,cEmail,cPhone)

provided\_by(cID,rID, ret\_date)

Foreign key (cID) references Company (cID)

Foreign key (rID) references Result (rID)

Result(rID,seqs,variant,drug,mlevel,sID)

Foreign key (sID) references Sample (sID)

analyzed\_by(rID,dID,dosage)

Foreign key (rID) references Result (rID)

Foreign key (dID) references Doctor (rID)

Doctor(dID,dFirst,dLast,dSpecialty)

diagnosed\_by(dID,pID,diag\_date,rID,diagnosis)

Foreign key (dID) references Doctor (dID)

Foreign key (pID) references Patient (pID)

**5.2 Relational Table Details**

| **Table Name** | **Attribute** | **Description** |
| --- | --- | --- |
| Patient | pID | unique patient ID |
| pFirst | patient’s first name |
| pLast | patient’s last name |
| pDOB | patient’s date of birth |
| pAddress | patient’s address |
| pEmail | patient’s email address |
| pPhone | patient’s phone number |
| Sample | sID | unique sample ID |
| collection\_date | date of sample’s collection |
| sType | type of sample |
| pID | patient ID |
| Sequenced\_by | cID | unique company ID |
| sID | unique sample ID |
| sequence\_date | Date sample was sequenced |
| Company | cID | unique company ID |
| cName | company name |
| cEmail | company email address |
| cPhone | company phone number |
| Result | rID | result ID |
| seqs | sequences |
| variant | Gene variant determined from sequencing |
| drug | Drug needed by patient |
| mlevel | Metabolism level of drug based on gene variant |
| sID | Sample ID |
| provided\_by | cID | Company ID |
| rID | Result ID |
| ret\_date | Date sequenced sample was returned |
| Doctor | dID | unique doctor ID |
| dFirst | doctor’s first name |
| dLast | doctor’s last name |
| dSpecialty | doctor’s specialty |
| analyzed\_by | rID | Result ID |
| dID | Doctor ID |
| dosage | Recommended dosage of drug based on patient genes |
| Diagnosed\_by | dID | unique doctor ID |
| pID | unique patient ID |
| diag\_date | Date of diagnosis |
| rID | Result ID |
| diagnosis | Doctor diagnosis of patient |

1. **Queries**

**6.1 Query Descriptions**

| **Query Name** | **Descriptions** | **Output** | **Relations Accessed** |
| --- | --- | --- | --- |
| High Metabolizing Patients | Find information on patients who have high drug metabolism and the number of drugs they are prescribed. Group by the patient ID and order by the number of drugs taken. | * pID * pFirst * pLast * pPhone * Drug count | * Patient * Sample * Result |
| maxDrugDosage | For each drug, find the highest dosage given to patient(s). Return patient information including first and last name, as well as patient ID. Group by drug in subquery and by patient ID/drug in main query, then sort the results in patient ID ascending order. | * pID * pFirst * pLast * drug * maxDosage | * Patient * Sample * Result * Analyzed\_by |
| doctorSampleCount | For each doctor, determine the number of each sample type that they have requested | * dID * dFirst * dLast * sType * sCount | * Diagnosed\_by * Patient * sample |
| doctorUniqueVariants | For each doctor, determine the number of variants analyzed and display those having doctor’s data having analyzed count >= 2. Order by variant count. | * dID * dFirst * dLast * dspecialty * variant\_count | * Doctor * Analyzed\_by * result |

**6.2 DQL**

**DQL:**

**High Metabolizing Patients**

SELECT Patient.pID,pFirst,pLast,pPhone,count(drug) as drugCount

FROM Patient,Sample,Result

WHERE Patient.pID = Sample.pID AND

Sample.sID = Result.sID AND

(mlevel = 'rapid' OR mlevel = 'ultrarapid')

GROUP BY Patient.pID

ORDER BY count(drug) DESC

;

**Output1:**

"pid","pfirst","plast","pPhone","drugCount"

"P001","Bradley","Ostberg","720-123-4567",2

"P003","Shania","Graves","206-555-6789",2

"P007","Kae","Andrews","404-567-8901",2

"P004","Wisteria","Poole","504-234-5678",1

"P005","Sasha","Law","916-345-6789",1

"P008","Dora","Peyton","503-678-9012",1

**maxDrugDosages**

SELECT Patient.pID, Patient.pFirst, Patient.pLast, Result.drug, max(Analyzed\_by.dosage) as maxDosage

FROM Patient, Sample, Result, Analyzed\_by

WHERE Patient.pID = Sample.pID AND

Sample.sID = Result.sID AND

Result.rID = Analyzed\_by.rID AND

(drug,dosage) IN (SELECT drug, max(dosage)

FROM Analyzed\_by,Result

WHERE Result.rID = Analyzed\_by.rID

GROUP BY drug)

GROUP BY Patient.pID, Result.drug

ORDER BY Patient.pID ASC

;

**Output2:**

"*pID*","*pFirst*","*pLast*","*drug*","*maxDosage*"

"P001","Bradley","Ostberg","Clopidogrel",300

"P003","Shania","Graves","Efavirenz",600

"P004","Wisteria","Poole","Celecoxib",65

"P007","Kae","Andrews","Voriconazole",350

"P010","Dorinda","Law","Voriconazole",350

**doctorSampleCount**

SELECT doctor.did, dfirst, dlast, sample.stype, count(stype) as sampleNum

FROM doctor, diagnosed\_by, patient, sample

WHERE doctor.did = diagnosed\_by.did AND

diagnosed\_by.pid = patient.pid AND

patient.pid = sample.pid

GROUP BY doctor.did, sample.stype

ORDER BY doctor.did ASC;

**Output3:**

"did" "dfirst" "dlast" "stype" "samplenum"

"D001" "Vincent" "Bird" "oral" 3

"D001" "Vincent" "Bird" "urine" 1

"D002" "Andrea" "Staford" "blood" 1

"D002" "Andrea" "Staford" "oral" 1

"D003" "Dennis" "Franklin" "blood" 2

"D003" "Dennis" "Franklin" "oral" 4

"D003" "Dennis" "Franklin" "urine" 1

"D004" "Marc" "Ruiz" "blood" 1

"D004" "Marc" "Ruiz" "oral" 2

"D005" "Halima" "Kim" "blood" 2

"D005" "Halima" "Kim" "oral" 1

"D006" "Jackson" "Cannon" "blood" 1

"D006" "Jackson" "Cannon" "oral" 2

"D007" "Isaiah" "Huffman" "blood" 1

"D007" "Isaiah" "Huffman" "oral" 1

"D007" "Isaiah" "Huffman" "urine" 1

**doctorUniqueVariants**

SELECT d.dID, d.dFirst, d.dLast, d.dSpecialty, COUNT(DISTINCT r.variant) AS variant\_count

FROM doctor d, Analyzed\_by A, Result r

WHERE d.dID = A.dID

AND r.rID = A.rID

GROUP BY d.dID

HAVING COUNT(DISTINCT r.variant) >= 2

ORDER BY variant\_count DESC;

**OUTPUT4 :**

"did" "dfirst" "dlast" "dspecialty" "variant\_count"

"D003" "Dennis" "Franklin" "Pathology" 4

"D004" "Marc" "Ruiz" "Cardiology" 2

"D007" "Isaiah" "Huffman" "Internal Medicine" 2

"D001" "Vincent" "Bird" "Internal Medicine" 2

"D005" "Halima" "Kim" "Medical Genetics" 2

"D006" "Jackson" "Cannon" "Family Medicine" 2

1. **ORM Implementation**

**7.1 Query Descriptions**

| **Query name** | **Description** | **Output** | **Relations Accessed** |
| --- | --- | --- | --- |
| BloodSamples | Finds patient data for patients that have had blood samples taken | * pID * pFirst * pLast * pDOB * pAddress * pEmail * pPhone | * Patient * Sample |
| PatientClopidogrelPrescription | Returns patient information for patients taking the drug ‘Clopidogrel’ | * pID * pFirst * pLast * pDOB * pAddress * pEmail * pPhone | * Patient * Sample * Result |
| PatientWithResultFromMaxHealth | Returns patients with sample results from Novartis | * pID * pFirst * pLast * pDOB * pAddress * pEmail * pPhone | * Patient * Sample * Result * Company |
| PatientsWwithVariant\_CYP2C19\*1/\*17 | Returns Patients with the Variant - CYP2C19\*1/\*17 | * pID * pFirst * pLast * pDOB * pAddress * pEmail * pPhone | * Patient * Sample * Result |

**7.2 Python Script**

#%%

from typing import List

from typing import Optional

from sqlalchemy import ForeignKey

from sqlalchemy import String, Integer, Date

from sqlalchemy.orm import DeclarativeBase

from sqlalchemy.orm import Mapped

from sqlalchemy.orm import mapped\_column

from sqlalchemy.orm import relationship

from sqlalchemy import create\_engine

from sqlalchemy.orm import Session

from sqlalchemy import select

#%%

#DB Connection: create\_engine(DBMS\_name+driver://<username>:<password>@<hostname>/<database\_name>)

engine = create\_engine("postgresql+psycopg2://postgres:<pw>@localhost/<db>")

#%%

#Define Classes/Tables

class Base(DeclarativeBase):

pass

# Base.metadata.drop\_all(bind=engine, tables=[Patient.\_\_table\_\_, Sample.\_\_table\_\_,Result.\_\_table\_\_,Company.\_\_table\_\_])

class Patient(Base):

\_\_tablename\_\_ = "Patient"

pid: Mapped[str] = mapped\_column(String(50), primary\_key=True)

pFirst: Mapped[str] = mapped\_column(String(50))

pLast: Mapped[str] = mapped\_column(String(50))

pDOB: Mapped[str] = mapped\_column(Date)

pAddress: Mapped[str] = mapped\_column(String(50))

pEmail: Mapped[str] = mapped\_column(String(50))

pPhone: Mapped[str] = mapped\_column(String(50))

Sample: Mapped[List["Sample"]] = relationship(

back\_populates="Patient", cascade="all, delete-orphan"

)

def \_\_repr\_\_(self) -> str: #represents the object as a string

return f"Patient(pid={self.pid!r}, pFirst={self.pFirst!r}, pLast={self.pLast!r}, pDOB={self.pDOB!r}, pAddress={self.pAddress!r}, pEmail={self.pEmail!r}, pPhone={self.pPhone!r})"

class Sample(Base):

\_\_tablename\_\_ = "Sample"

sid: Mapped[str] = mapped\_column(String(50), primary\_key=True)

collection\_date: Mapped[str] = mapped\_column(Date)

sType: Mapped[str] = mapped\_column(String(50))

pid: Mapped[str] = mapped\_column(String(50), ForeignKey("Patient.pid"))

Patient: Mapped["Patient"] = relationship(back\_populates="Sample")

Result: Mapped["Result"] = relationship(

back\_populates="Sample",cascade="all, delete-orphan"

)

def \_\_repr\_\_(self) -> str:

return f"Sample(sid={self.sid!r}, collection\_date={self.collection\_date!r}, sType={self.Stype!r})"

class Result(Base):

\_\_tablename\_\_ = "Result"

rid: Mapped[str] = mapped\_column(String(50), primary\_key=True)

seqs: Mapped[str] = mapped\_column(String(100))

variant: Mapped[str] = mapped\_column(String(50))

drug: Mapped[str] = mapped\_column(String(50))

mlevel: Mapped[str] = mapped\_column(String(50))

sid: Mapped[str] = mapped\_column(String(50), ForeignKey("Sample.sid"))

Sample: Mapped["Sample"] = relationship(back\_populates="Result")

cid: Mapped[Optional[List[str]]] = mapped\_column(String(50), ForeignKey("Company.cid"))

Company: Mapped[Optional['Company']] = relationship(

back\_populates='Result')

def \_\_repr\_\_(self) -> str:

return f"Result(rid={self.rid!r}, seqs={self.seqs!r}, variant={self.variant!r}, drug={self.drug!r}, mlevel={self.mlevel!r})"

class Company(Base):

\_\_tablename\_\_ = "Company"

cid: Mapped[str] = mapped\_column(String(50), primary\_key=True)

cName: Mapped[str] = mapped\_column(String(50))

cEmail: Mapped[str] = mapped\_column(String(50))

cPhone: Mapped[str] = mapped\_column(String(50))

Result: Mapped[List["Result"]] = relationship(

back\_populates='Company', cascade='all, delete-orphan')

def \_\_repr\_\_(self) -> str:

return f"Company(cid={self.cid!r}, cName={self.cName!r}, cEmail={self.cEmail!r}, cPhone={self.cPhone!r})"

#Create Tables

Base.metadata.create\_all(engine)

#%%

# Test Data

with Session(engine) as session:

P001 = Patient(

pid ='P001',

pFirst ='Bradley',

pLast ='Ostberg',

pDOB ='1978-05-23',

pAddress ='1433 Cherry Street, Denver, Colorado',

pEmail ='bostberg@gmail.com',

pPhone ='720-123-4567',

Sample = [

Sample(sid='S001',collection\_date='2016-05-02',sType='oral',

Result=Result(rid='R001',seqs='ATGGTCTTACTTGGTCTTGCAGAAGCAGGGTATGGAACAGTCCCTTTGTCTTCC',variant='CYP2C19\*1/\*17',drug='Clopidogrel',mlevel='rapid',cid='C001')),

Sample(sid='S002',collection\_date='2016-05-02',sType='blood',

Result=Result(rid='R002',seqs='GTCCTGCTCGCGCGCTCGCGCGCGCGCGCGCGCTGCGCGCTGCGCGCGCGCGC',variant='CYP2C19\*1/\*17',drug='Voriconazole',mlevel='rapid',cid='C001'))

])

P002 = Patient(

pid ='P002',

pFirst ='Shirlee',

pLast ='Mould',

pDOB ='1986-01-14',

pAddress ='789 Elmwood Avenue, Austin, Texas',

pEmail ='smould@yahoo.com',

pPhone ='512-789-0123',

Sample = [

Sample(sid='S003',collection\_date='2016-05-18',sType='blood',

Result=Result(rid='R003',seqs='CGGAGTGACACGTCTTGAACTGTGATGTTGTGTCTTCAGTTTCCGAGAAGGGC',variant='CYP2C19\*1/\*1',drug='Voriconazole',mlevel='normal',cid='C003')

)

])

P003 = Patient(

pid ='P003',

pFirst ='Shania',

pLast ='Graves',

pDOB ='1992-08-07',

pAddress ='5279 Maple Drive, Seattle, Washington',

pEmail ='sgraves@gmail.com',

pPhone ='206-555-6789',

Sample = [

Sample(sid='S004',collection\_date='2019-08-21',sType='urine',

Result=Result(rid='R004',seqs='TGCTGCCAACTTGGAGGCGCAGCGCGAGCGCGCGCGCGCGCGCGCGCGCGCGC',variant='CYP2B6\*1/\*4',drug='Efavirenz',mlevel='rapid',cid='C002')

),

Sample(sid='S013',collection\_date='2020-10-12',sType='oral',

Result=Result(rid='R013',seqs='GGAGCTTTGGGAGGAAGCCAGGAAGAGTGCTCAGAGCTGGGAGGTGTTGTGC',variant='CYP2C19\*1/\*17',drug='Clopidogrel',mlevel='rapid',cid='C003'))

])

P004 = Patient(

pid ='P004',

pFirst ='Wisteria',

pLast ='Poole',

pDOB ='1972-10-11',

pAddress ='2218 Oak Street, New Orleans, Louisiana',

pEmail ='wpoole@hotmail.com',

pPhone ='504-234-5678',

Sample = [

Sample(sid='S005',collection\_date='2017-11-20',sType='oral',

Result=Result(rid='R005',seqs='GAGGGGGATGTTGGAGCTGCGGCGTTGCCTCTGGGGTTCTAGGTGTTTTGCTG',variant='CYP2B6\*6/\*6',drug='Efavirenz',mlevel='poor',cid='C002')),

Sample(sid='S006',collection\_date='2017-12-13',sType='blood',

Result=Result(rid='R006',seqs='GGAGCGTGCGCTTGCGCGCGAGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGC',variant='CYP2C19\*17/\*17',drug='Celecoxib',mlevel='ultrarapid',cid='C001'

))

])

P005 = Patient(

pid ='P005',

pFirst ='Sasha',

pLast ='Law',

pDOB ='1999-03-29',

pAddress ='4002 Pine Avenue, Sacramento, California',

pEmail ='slaw@gmail.com',

pPhone ='916-345-6789',

Sample = [

Sample(sid='S007',collection\_date='2020-06-04',sType='oral',

Result=Result(rid='R007',seqs='ACGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCG',variant='CYP2C19\*17/\*17',drug='Voriconazole',mlevel='ultrarapid',cid='C003'))

])

P006 = Patient(

pid ='P006',

pFirst ='Tahnee',

pLast ='Harlan',

pDOB ='1983-06-09',

pAddress ='9026 Cedar Lane, Indianapolis, Indiana',

pEmail ='tharlan@yahoo.com',

pPhone ='317-456-7890',

Sample = [

Sample(sid='S008',collection\_date='2020-03-02',sType='oral',

Result=Result(rid='R008',seqs='GATTTGGTTGGGGAGTTGCTGAGGCAGAAGGCTGGCCAGTGTTCTCTGATTTA',variant='CYP2C19\*2/\*2',drug='Clopidogrel',mlevel='poor',cid='C003'))

])

P007 = Patient(

pid ='P007',

pFirst ='Kae',

pLast ='Andrews',

pDOB ='1979-12-01',

pAddress ='6890 Birchwood Drive, Atlanta, Georgia',

pEmail ='kandrews@yahoo.com',

pPhone ='404-567-8901',

Sample = [

Sample(sid='S009',collection\_date='2018-07-23',sType='urine',

Result=Result(rid='R009',seqs='GCTCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCG',variant='CYP2C19\*1/\*17',drug='Voriconazole',mlevel='rapid',cid='C001')),

Sample(sid='S010',collection\_date='2018-07-23',sType='blood',

Result=Result(rid='R010',seqs='TGGTACGTGTGAGTCCAGGGTCCAGGATAGGCGTCTCCATCCCTGTGATGGG',variant='CYP2C19\*17/\*17',drug='Clopidogrel',mlevel='ultrarapid',cid='C002'

))

])

P008 = Patient(

pid ='P008',

pFirst ='Dora',

pLast ='Peyton',

pDOB ='1996-04-12',

pAddress ='1753 Spruce Street, Portland, Oregon',

pEmail ='dpeyton@gmail.com',

pPhone ='503-678-9012',

Sample = [

Sample(sid='S011',collection\_date='2020-06-08',sType='oral',

Result=Result(rid='R011',seqs='ATCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGC',variant='CYP2B6\*1/\*4',drug='Efavirenz',mlevel='rapid',cid='C001'

))

])

P009 = Patient(

pid ='P009',

pFirst ='Orrell',

pLast ='Scrivener',

pDOB ='1988-09-03',

pAddress ='3387 Aspen Court, Baltimore, Maryland',

pEmail ='oscrivener@gmail.com',

pPhone ='410-789-0123',

Sample = [

Sample(sid='S012',collection\_date='2020-07-19',sType='oral',

Result=Result(rid='R012',seqs='CGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGC',variant='CYP2B6\*1/\*1',drug='Efavirenz',mlevel='normal',cid='C002')) ])

P010 = Patient(

pid ='P010',

pFirst ='Dorinda',

pLast ='Law',

pDOB ='1976-11-25',

pAddress ='620 Poplar Road, Kansas City, Missouri',

pEmail ='dlaw@hotmail.com',

pPhone ='816-234-5678',

Sample = [

Sample(sid='S014',collection\_date='2021-01-06',sType='oral',

Result=Result(rid='R014',seqs='CTCTCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGC',variant='CYP2C19\*1/\*1',drug='Clopidogrel',mlevel='normal',cid='C003')),

Sample(sid='S015',collection\_date='2020-11-12',sType='oral',

Result=Result(rid='R015',seqs='GTGAGTGTGAGTCTGGAGAGGATGAGGACAGGGAAGAGGGACGGGAGGGAGC',variant='CYP2C19\*1/\*1',drug='Voriconazole',mlevel='normal',cid='C002'

))

])

C001 = Company(

cid = 'C001',

cName = 'Medtronic',

cEmail = 'info@medtronic.com',

cPhone = '692-254-0958',

Result = [])

C002 = Company(

cid="C002",

cName='Novartis',

cEmail="contact.center@novartis.com",

cPhone="789-574-8531",

Result = [])

C003 =Company(

cid="C003",

cName="Max Health",

cEmail="info.mh@max.com",

cPhone="727-382-0333",

Result = [])

session.add\_all([P001,P002,P003,P004,P005,P006,P007,P008,P009,P010,C001,C002,C003])

session.commit()

**# Queries**

**# Query 1**

session = Session(engine)

print('## BloodSamples')

stmt = (

select(Patient)

.join(Patient.Sample)

.where(Sample.sType == "blood")

)

for p in session.scalars(stmt):

print(p)

**#Query 1 Output:**

## BloodSamples

Patient(pid='P001', pFirst='Bradley', pLast='Ostberg', pDOB=datetime.date(1978, 5, 23), pAddress='1433 Cherry Street, Denver, Colorado', pEmail='bostberg@gmail.com', pPhone='720-123-4567')

Patient(pid='P002', pFirst='Shirlee', pLast='Mould', pDOB=datetime.date(1986, 1, 14), pAddress='789 Elmwood Avenue, Austin, Texas', pEmail='smould@yahoo.com', pPhone='512-789-0123')

Patient(pid='P004', pFirst='Wisteria', pLast='Poole', pDOB=datetime.date(1972, 10, 11), pAddress='2218 Oak Street, New Orleans, Louisiana', pEmail='wpoole@hotmail.com', pPhone='504-234-5678')

Patient(pid='P007', pFirst='Kae', pLast='Andrews', pDOB=datetime.date(1979, 12, 1), pAddress='6890 Birchwood Drive, Atlanta, Georgia', pEmail='kandrews@yahoo.com', pPhone='404-567-8901')

**#Query 2**

print('## PatientClopidogrelPrescription ')

stmt1 = (

select(Patient)

.join(Patient.Sample)

.join(Sample.Result)

.where(Result.drug == "Clopidogrel")

)

for p in session.scalars(stmt1):

print(p)

**#Query 2 Output:**

## PatientClopidogrelPrescription

Patient(pid='P001', pFirst='Bradley', pLast='Ostberg', pDOB=datetime.date(1978, 5, 23), pAddress='1433 Cherry Street, Denver, Colorado', pEmail='bostberg@gmail.com', pPhone='720-123-4567')

Patient(pid='P003', pFirst='Shania', pLast='Graves', pDOB=datetime.date(1992, 8, 7), pAddress='5279 Maple Drive, Seattle, Washington', pEmail='sgraves@gmail.com', pPhone='206-555-6789')

Patient(pid='P006', pFirst='Tahnee', pLast='Harlan', pDOB=datetime.date(1983, 6, 9), pAddress='9026 Cedar Lane, Indianapolis, Indiana', pEmail='tharlan@yahoo.com', pPhone='317-456-7890')

Patient(pid='P007', pFirst='Kae', pLast='Andrews', pDOB=datetime.date(1979, 12, 1), pAddress='6890 Birchwood Drive, Atlanta, Georgia', pEmail='kandrews@yahoo.com', pPhone='404-567-8901')

Patient(pid='P010', pFirst='Dorinda', pLast='Law', pDOB=datetime.date(1976, 11, 25), pAddress='620 Poplar Road, Kansas City, Missouri', pEmail='dlaw@hotmail.com', pPhone='816-234-5678')

**# Query 3**

print("##PatientWithResultFromNovartis")

stmt2 = (

select(Patient,Company)

.join(Patient.Sample)

.join(Sample.Result)

.join(Result.Company)

.where(Company.cName == 'Novartis'))

for p in session.scalars(stmt2):

print(p)

**# Query 3 Output:**

##PatientWithResultFromNovartis

Patient(pid='P003', pFirst='Shania', pLast='Graves', pDOB=datetime.date(1992, 8, 7), pAddress='5279 Maple Drive, Seattle, Washington', pEmail='sgraves@gmail.com', pPhone='206-555-6789')

Patient(pid='P004', pFirst='Wisteria', pLast='Poole', pDOB=datetime.date(1972, 10, 11), pAddress='2218 Oak Street, New Orleans, Louisiana', pEmail='wpoole@hotmail.com', pPhone='504-234-5678')

Patient(pid='P007', pFirst='Kae', pLast='Andrews', pDOB=datetime.date(1979, 12, 1), pAddress='6890 Birchwood Drive, Atlanta, Georgia', pEmail='kandrews@yahoo.com', pPhone='404-567-8901')

Patient(pid='P009', pFirst='Orrell', pLast='Scrivener', pDOB=datetime.date(1988, 9, 3), pAddress='3387 Aspen Court, Baltimore, Maryland', pEmail='oscrivener@gmail.com', pPhone='410-789-0123')

Patient(pid='P010', pFirst='Dorinda', pLast='Law', pDOB=datetime.date(1976, 11, 25), pAddress='620 Poplar Road, Kansas City, Missouri', pEmail='dlaw@hotmail.com', pPhone='816-234-5678')

**Query 4:**

print('#Patients\_with\_Variant\_CYP2C19\*1/\*17#')

stmt4 = (

select(Patient)

.join(Patient.Sample)

.join(Sample.Result)

.where(Result.variant == "CYP2C19\*1/\*17")

)

for p in session.scalars(stmt4):

print(p)

**Query 4 Output:**

#Patients\_with\_Variant\_CYP2C19\*1/\*17#

Patient(pid='P001', pFirst='Bradley', pLast='Ostberg', pDOB=datetime.date(1978, 5, 23), pAddress='1433 Cherry Street, Denver, Colorado', pEmail='bostberg@gmail.com', pPhone='720-123-4567')

Patient(pid='P003', pFirst='Shania', pLast='Graves', pDOB=datetime.date(1992, 8, 7), pAddress='5279 Maple Drive, Seattle, Washington', pEmail='sgraves@gmail.com', pPhone='206-555-6789')

Patient(pid='P007', pFirst='Kae', pLast='Andrews', pDOB=datetime.date(1979, 12, 1), pAddress='6890 Birchwood Drive, Atlanta, Georgia', pEmail='kandrews@yahoo.com', pPhone='404-567-8901')

**Group Status Report**

**Phase 3**

Dates and attendance of team meetings:

Tuesday, April 18th 11:30am - 12:45pm All group members

Progress overview as of April 18th:

The Phase 3 deliverables have been completed. Our Pharmacogenomic Healthcare Database has been updated with classes and data specific to the ORM Implementation. Below are the responsibilities of each team member for creation of classes, queries in Python, and updates to our previous model to include our Python script.

Contributions of group members:

Phase Leader: Sathvik Maridasana Nagaraj

* Class: Company
* Query in Python : Patients\_with\_Variant\_CYP2C19\*1/\*17

Phase Recorder: Aaron Myrold

* Class: Company
* Class: Result
* Query in Python: PatientWithResultFromNovartis

Phase Checker: Crisi Patelis

* Query in Python: PatientClopidogrelPrescription
* Group status report

Technical Advisor: Michael Saban

* Revised ER diagram based on Phase 2 Feedback
* Class: Patient
* Class: Sample
* Class: Result
* Query in Python: BloodSamples