PHASE 3: SUBMISSION CHECKLIST/SIGNOFF SHEET

Group: Rihab Junagadhwala, Pragya Kansal, Kayla Salerno, Arub Sumayli

Deliverables:

- Description of the organization
- ER diagram with min/max specifications
- Constraints not in ER diagram
- [Revised] Relational Schema with Referential Integrity
- [Revised] Queries with brief description
- Completed ORM Implementation
- Group Assessment
- Group Status Report

We have each reviewed the contents of this deliverable.

Phase Leader	Kayla Salerno	_____
Phase Recorder	Arub Sumayli	_____
Phase Checker	Rihab Junagadhwala	_____
Technical Advisor	Pragya Kansal	_____

1. Introduction

This is a database that stores patient information for the pediatrics department of a hospital. The purpose of our database is to keep track of important information about patients such as their illness/condition, insurance provider, date of last appointment, date of birth, and other relevant information. We are also storing the entities that the patients have a relationship with, such as their parents/guardians, primary physician, and nurse, along with their corresponding attributes.

The contact person that provided insight into the structure of a typical pediatrics department was a pediatrician we know who works at NorthShore Evanston Hospital. They summarized the general organization of the department, important members that are usually found in the database, and the type of data that is typically stored. We adapted our database to reflect the key parts of a pediatrics database in relation to patient information.

2. Requirements Description

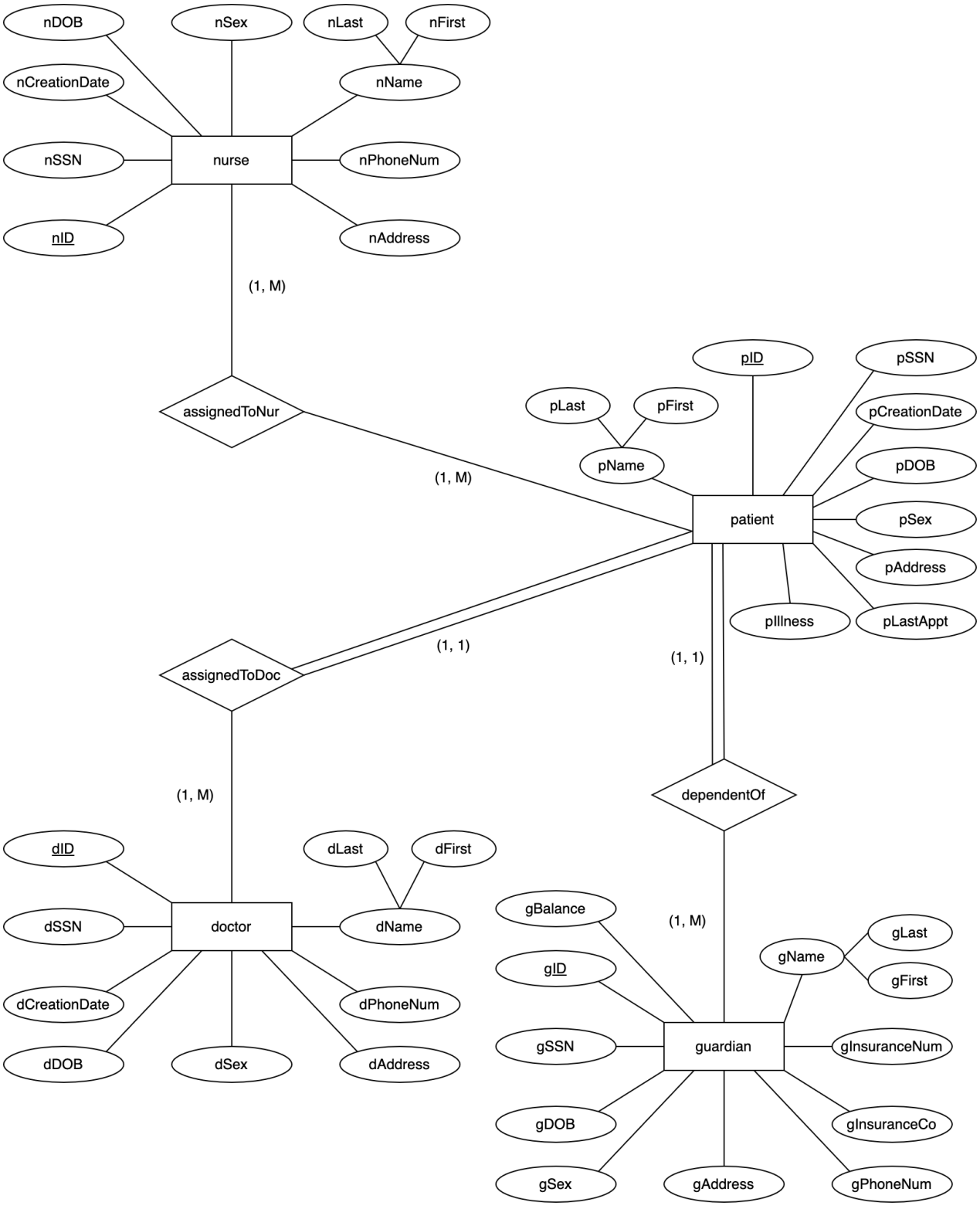
The Hospital Pediatrics Patient Database (HOSPPEDSPATIENTS) stores relevant information regarding a hospital's desire to track patient information in the Pediatrics department. The HOSPPEDSPATIENTS Database stores information regarding patients, parents/guardians of the patients, doctors, and nurses. Below are more specifics regarding the entities, their attributes, and the relationships between the entities.

- Patients are children who have been admitted because they have a condition to be treated. Patient information includes the patient's full name, social security number, patient ID, creation date, date of birth, sex, address, insurance, insurance company, doctor, nurse(s), last appointment date, and condition/illness. Patients can only have one doctor and one guardian, but they can have multiple nurses.
- The guardian is whom the patient legally and financially depends on. Guardian information includes their full name, social security number, date of birth, sex, address, phone number, insurance company, and insurance number. A patient can only have one parent/guardian on record. Guardians can have more than one dependent/patient.

- Doctors are the individuals who treat the patients. Their information includes doctor ID, full name, social security number, date of birth, creation date, sex, address, phone number and list of patients. Doctors can have multiple patients, but patients can only have one doctor.
- Nurses are the individuals who assist the doctor with overseeing the patient's health. Their information includes their full name, nurse ID, date of birth, creation date, social security number, sex, address, phone number and list of patients. Nurses can have multiple patients and patients can have multiple nurses.
- The database keeps a running record of all patients in the hospital's Pediatrics department, as well as the doctor and nurses assigned to each patient. Patients are connected to their guardians so they share certain information with them, such as the existence of insurance.

3. ER Diagram

The figure below shows the ER diagram of the Hospital Pediatrics Patient Database (HOSPEDSPATIENTS).



4. ER Diagram Uncaptured Constraints

The following is a list of constraints that are not captured by the ER diagram of HOSPEDSPATIENTS.

- Patients must be under the insurance of their parent/guardian(s).
- The phone number format must be XXXXXXXXXX, or Area Code Phone Number.
- For biological sex, the three options are male, female, and intersex.
- The social security number must be written as: XXXXXXXXX.
- For the amount the guardian needs to pay, or gBalance, the currency must be in US dollars.
- CreationDate and DOB Format: MM/DD/YYYY.
- The address format must be: Street, City, State Zip Code.
- Patients, guardians, doctors, and nurses must have residency in Illinois.
- Patients must be born no earlier than 2004.

5. Relational Schema

This section provides the relational schema with referential integrity and the relational table details.

*For this phase of the project, we removed the attributes gInsuranceCo and gInsuranceNum from the dependentOf relationship. We found that they were not necessary for our outputs. Additionally, we removed them to simplify the organization since we covered only the core aspects of SQLAlchemy.

5.1 Relational Schema with Referential Integrity

patient (pID, pLast, pFirst, pSSN, pCreationDate, pDOB, pSex, pAddress, pLastAppt, pIllness)

guardian (gID, gLast, gFirst, gSSN, gDOB, gSex, gAddress, gPhoneNum, gInsuranceCo, gInsuranceNum)

doctor (dID, dLast, dFirst, dSSN, dCreationDate, dDOB, dSex, dAddress, dPhoneNum)

nurse (nID, nLast, nFirst, nSSN, nCreationDate, nDOB, nSex, nAddress, nPhoneNum)

assignedToNur(pID, nID)

foreign key (pID) references patient (pID)

foreign key (nID) references nurse (nID)

assignedToDoc(pID, dID)

foreign key (pID) references patient (pID)

foreign key (dID) references doctor (dID)

dependentOf(pID, gID)

foreign key (pID) references patient (pID)

foreign key (gID) references guardian (gID)

5.2 Relational Table Details

The relational schema given in Section 5.1 was mapped into the following tables in the HOSPPEDSPATIENTS database. Primary keys have been underlined. Tables that have multiple attributes underlined represent composite keys.

Table Name	**Attribute**	**Description**
patient	pID	unique patient ID
	pLast	last name of patient
	pFirst	first name of patient
	pSSN	social security number of patient
	pCreationDate	date that patient was admitted into hospital and added to database
	pDOB	birthdate of patient
	pSex	biological sex of patient
	pAddress	home address of patient
	pLastAppt	date of patient's last appointment
	pIllness	description of patient condition
guardian	gID	unique guardian ID
	gLast	last name of guardian
	gFirst	first name of guardian

	gSSN	social security number of guardian
	gDOB	birthdate of guardian
	gSex	biological sex of guardian
	gAddress	home address of guardian
	gPhoneNum	phone number of guardian
	gInsuranceCo	name of insurance company
	gInsuranceNum	insurance card number
	gBalance	payment amount remaining
doctor	dID	unique doctor ID
	dLast	last name of doctor
	dFirst	first name of doctor
	dSSN	social security number of doctor
	dCreationDate	date that doctor started working at hospital and added to database
	dDOB	birthdate of doctor
	dSex	biological sex of doctor
	dAddress	address of doctor
	dPhoneNum	phone number of doctor
	dPatientList	list of patients for doctor
nurse	nID	unique nurse id
	nLast	last name of nurse
	nFirst	first name of doctor
	nSSN	social security number of nurse
	nCreationDate	date that nurse started working at hospital and added to database
	nDOB	date of birth of nurse

	nSex	biological sex of nurse
	nAddress	address of nurse
	nPhoneNum	phone number of doctor
	nPatientList	list of patients for nurse
assignedToNur	pID	unique patient id (foreign key), references pID from patient
	nID	unique nurse id (foreign key), references nID from nurse
assignedToDoc	pID	unique patient id (foreign key), references pID from patient
	dID	unique doctor id (foreign key), references dID
dependentOf	pID	unique patient id (foreign key), references pID from patient
	gID	unique guardian id (foreign key), references gID from guardian

6. Queries

The following table summarizes the queries in the HOSPEDSPATIENTS Database.

Query Name	Description	Output	Relations Accessed
Query1 (Pragya)	Find the patient ID, first name, last name, illness, and address of all patients who have an illness of "Pneumonia" and live with their guardian. Sort the outputs in ascending alphabetical order of the patient's first name.	• pID • pFirst • pLast • pIllness • pAddress	• guardian • patient • dependentOf
Query2 (Arub)	For each guardian, print gLast, gBalance, gPhoneNum, and pID. Sort the output by the amount of money in each gBalance in	• gFirst • gLast • gBalance • gPhoneNum • pID	• guardian • patient

	descending order. Only print gBalance exceeding $10000.		
Query3 (Rihab)	Find the doctors that have patients who are female. List the doctor's first and last names, and the number of female patients each of them have. Sort by the number of female patients in ascending order.	• dLast • dFirst • numFemale Patients	• doctor • patients
Query4 (Kayla)	Print the ID, last and first name, sex, and the number of patients a nurse has for female nurses with 3 or fewer patients. Group by nurse ID. Sort the output by last name in ascending order.	• nID • nLast • nFirst • nSex • numPatients	• nurse • assignedToNur

7. ORM Implementation

7.1 DDL ORM Implementation

#NOTE: Drop the address & user_account tables before running this script

```
from typing import List
from typing import Optional
from sqlalchemy import ForeignKey
from sqlalchemy import String, Integer
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:insert-password@localhost/postgres")
```

#Define Classes/Tables

```
class Base(DeclarativeBase):
        pass


class Base(DeclarativeBase):
        pass

class Patient(Base):
        __tablename__ = 'patient'
        pID: Mapped[str] = mapped_column(String(10), primary_key = True, nullable = False)
        pLast: Mapped[str] = mapped_column(String(30), nullable = False)
        pFirst: Mapped[str] = mapped_column(String(30), nullable = False)
        pSSN: Mapped[int] = mapped_column(Integer, nullable = False)
        pCreationDate: Mapped[int] = mapped_column(String(10), nullable = False)
        pDOB: Mapped[str] = mapped_column(String(10), nullable = False)
        pSex: Mapped[str] = mapped_column(String(10), nullable = False)
        pAddress: Mapped[str] = mapped_column(String(100), nullable = False)
        pLastAppt: Mapped[str] = mapped_column(String(10), nullable = False)
        pIllness: Mapped[str] = mapped_column(String(250), nullable = False)
```

```
    def __repr__(self) -> str:
        return f"Patient(pID={self.pID!r}, pLast={self.pLast!r}), pFirst={self.pLast!r},
    pSSN={self.pSSN!r}, pCreationDate={self.pCreationDate!r}, pDOB={self.pDOB!r},
    pSex={self.pSex!r}, pAddress={self.pAddress!r}, pLastAppt={self.pLastAppt!r},
    pIllness={self.pIllness!r})"
```

class Guardian(Base):

```
    __tablename__ = 'guardian'
    gID: Mapped[str] = mapped_column(String(10), primary_key = True, nullable = False)
    gLast: Mapped[str] = mapped_column(String(30), nullable = False)
    gFirst: Mapped[str] = mapped_column(String(30), nullable = False)
    gSSN: Mapped[int] = mapped_column(Integer, nullable = False)
    gCreationDate: Mapped[int] = mapped_column(String (10), nullable = False)
    gDOB: Mapped[int] = mapped_column(String (10), nullable = False)
    gSex: Mapped[str] = mapped_column(String(10), nullable = False)
    gAddress: Mapped[str] = mapped_column(String(100), nullable = False)
    gPhoneNum: Mapped[int] = mapped_column(Integer, nullable = False)
    gInsuranceCo: Mapped[str] = mapped_column(String(100), nullable = False)
    gInsuranceNum: Mapped[int] = mapped_column(Integer, nullable = False)
    gBalane: Mapped[int] = mapped_column(Integer, nullable = False)

    def __repr__(self) -> str:
  return f"Guardian()"
```

class Doctor(Base):

```
    __tablename__ = 'doctor'
    dID: Mapped[str] = mapped_column(String(10), primary_key = True, nullable = False)
    dLast: Mapped[str] = mapped_column(String(30), nullable = False)
    dFirst: Mapped[str] = mapped_column(String(30), nullable = False)
    dSSN: Mapped[int] = mapped_column(Integer, nullable = False)
    dCreationDate: Mapped[str] = mapped_column(String(10), nullable = False)
    dDOB: Mapped[str] = mapped_column(String(10), nullable = False)
    dSex: Mapped[str] = mapped_column(String(10), nullable = False)
    dAddress: Mapped[str] = mapped_column(String(100), nullable = False)
    dPhoneNum: Mapped[int] = mapped_column(Integer, nullable = False)

    def __repr__(self) -> str:
        return f"Address(id={self.id!r}, email_address={self.email_address!r})"
```

class Nurse(Base):

__tablename__ = 'nurse'
nID: Mapped[str] = mapped_column(String(10), primary_key = True, nullable = False)
nLast: Mapped[str] = mapped_column(String(30), nullable = False)
nFirst: Mapped[str] = mapped_column(String(30), nullable = False)
nSSN: Mapped[int] = mapped_column(Integer, nullable = False)
nCreationDate: Mapped[str] = mapped_column(String(10), nullable = False)
nDOB: Mapped[str] = mapped_column(String(10), nullable = False)
nSex: Mapped[str] = mapped_column(String(10), nullable = False)
nAddress: Mapped[str] = mapped_column(String(100), nullable = False)
nPhoneNum: Mapped[int] = mapped_column(Integer, nullable = False)

class AssignedToNur(Base):

__tablename__ = 'assignedToNur'
pID: Mapped[str] = mapped_column(String(10), ForeignKey('patient.pID'), primary_key = True, nullable = False)
nID: Mapped[str] = mapped_column(String(10), ForeignKey('nurse.nID'), nullable=False)
patient = relationship('Patient', back_populates = 'assigned_doctor')
doctor = relationship('Doctor', back_populates = 'assigned_patients')

Patient.assigned_nurses = relationship('AssignedToNur', back_populates='patient')
Nurse.assigned_patients = relationship('AssignedToNur', back_populates='nurse')

class AssignedToDoc(Base):

__tablename__ = 'assignedToDoc'
pID: Mapped[str] = mapped_column(String(10), ForeignKey('patient.pID'), primary_key = True, nullable = False)
dID: Mapped[str] = mapped_column(String(10), ForeignKey('doctor.dID'), nullable = False)
assigned_doctors: Mapped['Doctor'] = relationship(back_populates = 'assignedToDoc')
assigned_patients: Mapped['Patient'] = relationship(back_populates = 'assignedToDoc')

class DependentOf(Base):

__tablename__ = 'dependentOf'
pID: Mapped[str] = mapped_column(String(10), ForeignKey(patient.pID), primary_key = True, nullable = False)
gID: Mapped[str] = mapped_column(String(10), ForeignKey(guardian.gID), nullable = False)

#Create Tables
Base.metadata.create_all(engine)

#Insert Data
```
with Session(engine) as session:
  patient1 = Patient(
        pID="5478828149",
        pLast="Krystle",
        pFirst="Carree",
        pSSN=880668772,
        pCreationDate="05/19/2011",
        pDOB="03/01/2007",
        pSex="Female",
        pAddress="123 Elm Street, Chicago, IL 60601",
        pLastAppt="9/15/2012",
        pIllness="Pneumonia")

patient2 = Patient(
        pID="2353169554",
        pLast="Wilhelmina",
        pFirst="Muhammad",
        pSSN=289612554,
        pCreationDate="07/01/2022",
        pDOB="07/12/2018",
        pSex="Male",
        pAddress="456 Maple Avenue, Springfield, IL 62701",
        pLastAppt="7/28/2022",
        pIllness="Pneumonia")

patient3 = Patient(
        pID="2867034035",
        pLast="Vania",
        pFirst="Marielle",
        pSSN=810946847,
        pCreationDate="10/26/2019",
        pDOB="03/24/2009",
        pSex="Female",
        pAddress="04 Grasskamp Trail, Chicago, IL 60626",
        pLastAppt="6/7/2020",
        pIllness="Bronchitis")
```

```
patient4 = Patient(
   pID="1608930009",
   pLast="Elsie",
   pFirst="Sayres",
   pSSN=402543798,
   pCreationDate="10/07/2020",
   pDOB="03/31/2017",
   pSex="Male",
   pAddress="101 Pine Lane, Rockford, IL 61101",
   pLastAppt="9/22/2021",
   pIllness="Meningitis")

patient5 = Patient(
   pID="1135007012",
   pLast="Elsworth",
   pFirst="Emmeline",
   pSSN=779483273,
   pCreationDate="01/19/2017",
   pDOB="04/06/2012",
   pSex="Female",
   pAddress="234 Birch Road, Aurora, IL 60501",
   pLastAppt="7/31/2019",
   pIllness="Strep throat")
```

(Continued…)

```
session.add_all([patient1, patient2, patient3, patient4, patient5, patient6, patient7, patient8,
patient9, patient10, patient11, patient12, patient13, patient14, patient15, patient16, patient17,
patient18, patient19, patient20, patient21, patient22, patient23, patient24, patient25, patient26,
patient27, patient28, patient29, patient30, patient31, patient32, patient33, patient34, patient35,
patient36, patient37, patient38, patient39, patient40, patient41, patient42, patient43, patient44,
patient45, patient46, patient47, patient48, patient49, patient50])
session.commit()

with Session(engine) as session:
        guardian1 = Guardian(
   gID="8520489390",
   gLast="Krystle",
   gFirst="Newlyn",
```

```
    gSSN=750817686,
    gDOB="06/14/1957",
    gSex="Female",
    gAddress="123 Elm Street, Chicago, IL 60601",
    gPhoneNum="3034941575",
    gInsuranceCo="Hettinger Group",
    gInsuranceNum="6891339960",
    gBalance=51666.84)

guardian2 = Guardian(
    gID="7251298740",
    gLast="Wilhelmina",
    gFirst="Standering",
    gSSN=594051136,
    gDOB="01/03/1977",
    gSex="Female",
    gAddress="456 Maple Avenue, Springfield, IL 62701",
    gPhoneNum="4618171573",
    gInsuranceCo="Sanford and Sons",
    gInsuranceNum="6857558183",
    gBalance=84567.45)

guardian3 = Guardian(
    gID="6445290620",
    gLast="Vania",
    gFirst="Nicolson",
    gSSN=769412787,
    gDOB="10/30/1980",
    gSex="Female",
    gAddress="789 Oak Drive, Peoria, IL 61601",
    gPhoneNum="3419702825",
    gInsuranceCo="Macejkovic Inc",
    gInsuranceNum="8545278918",
    gBalance=88203.48)

guardian4 = Guardian(
    gID="7483012498",
    gLast="Elsie",
    gFirst="Blacklock",
    gSSN=319616619,
```

```
    gDOB="12/10/1979",
    gSex="Female",
    gAddress="101 Pine Lane, Rockford, IL 61101",
    gPhoneNum="6479535323",
    gInsuranceCo="King Group",
    gInsuranceNum="2921673665",
    gBalance=68576.94)

guardian5 = Guardian(
    gID="3285231780",
    gLast="Elsworth",
    gFirst="Pennrington",
    gSSN=300410599,
    gDOB="11/22/1956",
    gSex="Male",
    gAddress="234 Birch Road, Aurora, IL 60501",
    gPhoneNum="6513595804",
    gInsuranceCo="West-Rolfson",
    gInsuranceNum="1526907097",
    gBalance=28278.86)

(Continued…)

session.add_all([guardian1, guardian2, guardian3, guardian4, guardian5, guardian6, guardian7,
guardian8, guardian9,
guardian10, guardian11, guardian12, guardian13, guardian14, guardian15, guardian16,
guardian17, guardian18, guardian19,
guardian20, guardian21])
session.commit()

# Nurse
with Session(engine) as session:
    nurse1 = Nurse(
    nID="8141746251",
    nLast="Cawston",
    nFirst="Crichton",
    nSSN=133386325,
    nCreationDate="07/14/1993",
    nDOB="09/07/1941",
    nSex="Male",
    nAddress="789 Birch Lane, Elgin, IL 60120",
```

```
    nPhoneNum="8607907498")

nurse2 = Nurse(
    nID="0154784443",
    nLast="Van de Vlies",
    nFirst="Neddie",
    nSSN=117903132,
    nCreationDate="05/02/2016",
    nDOB="08/26/1952",
    nSex="Female",
    nAddress="456 Cedar Drive, Schaumburg, IL 60173",
    nPhoneNum="3883137131")

nurse3 = Nurse(
    nID="7312243304",
    nLast="Shillington",
    nFirst="Annice",
    nSSN=607028352,
    nCreationDate="07/21/2003",
    nDOB="05/10/1976",
    nSex="Female",
    nAddress="222 Rosewood Avenue, Champaign, IL 61820",
    nPhoneNum="3511379758")

nurse4 = Nurse(
    nID="8809364287",
    nLast="Fellis",
    nFirst="Ward",
    nSSN=402910166,
    nCreationDate="06/20/2002",
    nDOB="07/27/1968",
    nSex="Male",
    nAddress="777 Maple Road, Joliet, IL 60401",
    nPhoneNum="9866889824")

nurse5 = Nurse(
    nID="2152501608",
    nLast="Maidment",
    nFirst="Heddie",
    nSSN=184403209,
```

```
    nCreationDate="02/01/1993",
    nDOB="12/01/1963",
    nSex="Female",
    nAddress="888 Orchid Court, Springfield, IL 62701",
    nPhoneNum="8901433799")

(Continued…)

session.add_all([nurse1, nurse2, nurse3, nurse4, nurse5, nurse6, nurse7, nurse8, nurse9, nurse10,
nurse11, nurse12,
nurse13, nurse14, nurse15, nurse16, nurse17, nurse18, nurse19, nurse20, nurse21, nurse22,
nurse23, nurse24, nurse25,
nurse26, nurse27, nurse28, nurse29, nurse30])
session.commit()

# Doctor
with Session(engine) as session:

        doctor1 = Doctor(
        dID='2433573130',
        dLast='Brewster',
        dFirst='Hannie',
        dSSN=737633372,
        dCreationDate='01/31/1998',
        dDOB='04/29/1955',
        dSex='Female',
        dAddress='5085 Brickson Park Point, Chicago, IL 60601',
                dPhoneNum="3487007747"
        )

doctor2 = Doctor(
    dID='4772535918',
    dLast='Sherlock',
    dFirst='Malanie',
    dSSN=428365612,
    dCreationDate='04/21/2003',
    dDOB='08/13/1952',
    dSex='Female',
    dAddress='60 Portage Parkway, Springfield, IL 62701',
    dPhoneNum="1605413705"
```

```
)

doctor3 = Doctor(
    dID='1550505211',
    dLast='Woodhead',
    dFirst='Jeff',
    dSSN=375677629,
    dCreationDate='01/03/2022',
    dDOB='09/06/1965',
    dSex='Male',
    dAddress='3 Graedel Road, Peoria, IL 61601',
    dPhoneNum="9626948996"
)

doctor4 = Doctor(
    dID='1228121907',
    dLast='Wrassell',
    dFirst='Dante',
    dSSN=490999123,
    dCreationDate='04/05/2014',
    dDOB='08/23/1989',
    dSex='Intersex',
    dAddress='5 Mosinee Way, Champaign, IL 61801',
    dPhoneNum="5977596025"
)

doctor5 = Doctor(
    dID='3790986631',
    dLast='Farmery',
    dFirst='Guendolen',
    dSSN=327519163,
    dCreationDate='10/13/1999',
    dDOB='05/11/1954',
    dSex='Female',
    dAddress='9753 South Crossing, Rockford, IL 61101',
    dPhoneNum="2848412538"
)
```

(Continued…)

```
session.add_all([doctor1, doctor2, doctor3, doctor4, doctor5, doctor6, doctor7, doctor8, doctor9,
doctor10,
doctor11, doctor12, doctor13, doctor14, doctor15, doctor16, doctor17, doctor18, doctor19,
doctor20])
session.commit()

with Session(engine) as session:
        nur_assign1 = AssignedToNur(pID="5478828149", nID="8224044106")
        nur_assign2 = AssignedToNur(pID="5478828149", nID="9207937719")
        nur_assign3 = AssignedToNur(pID="2353169554", nID="7312243304")
        nur_assign4 = AssignedToNur(pID="2353169554", nID="7502613455")
        nur_assign5 = AssignedToNur(pID="2867034035", nID="8809364287")

(Continued…)

session.add_all([nur_assign1, nur_assign2, nur_assign3, nur_assign4, nur_assign5,
                nur_assign6, nur_assign7, nur_assign8, nur_assign9, nur_assign10,
                nur_assign11, nur_assign12, nur_assign13, nur_assign14, nur_assign15,
                nur_assign16, nur_assign17, nur_assign18, nur_assign19, nur_assign20,
                nur_assign21, nur_assign22, nur_assign23, nur_assign24, nur_assign25,
                nur_assign26, nur_assign27, nur_assign28, nur_assign29, nur_assign30,
                nur_assign31, nur_assign32, nur_assign33, nur_assign34, nur_assign35,
                nur_assign36, nur_assign37, nur_assign38, nur_assign39, nur_assign40,
                nur_assign41, nur_assign42, nur_assign43, nur_assign44, nur_assign45,
                nur_assign46, nur_assign47, nur_assign48, nur_assign49, nur_assign50,
                nur_assign51, nur_assign52, nur_assign53, nur_assign54, nur_assign55,
                nur_assign56, nur_assign57, nur_assign58, nur_assign59, nur_assign60,
                nur_assign61, nur_assign62, nur_assign63, nur_assign64, nur_assign65,
                nur_assign66, nur_assign67, nur_assign68, nur_assign69, nur_assign70,
                nur_assign71, nur_assign72, nur_assign73, nur_assign74, nur_assign75,
                nur_assign76, nur_assign77, nur_assign78, nur_assign79, nur_assign80,
                nur_assign81, nur_assign82, nur_assign83, nur_assign84, nur_assign85,
                nur_assign86, nur_assign87, nur_assign88, nur_assign89, nur_assign90,
                nur_assign91, nur_assign92, nur_assign93, nur_assign94, nur_assign95,
                nur_assign96, nur_assign97])
session.commit()

# AssignedToDoc
with Session(engine) as session:
        doc_assign1 = AssignedToDoc(pID="5478828149", dID="4775744720")
```

```
        doc_assign2 = AssignedToDoc(pID="2353169554", dID="2059515203")
        doc_assign3 = AssignedToDoc(pID="2867034035", dID="9513444856")
        doc_assign4 = AssignedToDoc(pID="1608930009", dID="5012485009")
        doc_assign5 = AssignedToDoc(pID="1135007012", dID="2059515203")

(Continued…)

session.add_all([doc_assign1, doc_assign2, doc_assign3, doc_assign4, doc_assign5,
                 doc_assign6, doc_assign7, doc_assign8, doc_assign9, doc_assign10,
                 doc_assign11, doc_assign12, doc_assign13, doc_assign14, doc_assign15,
                 doc_assign16, doc_assign17, doc_assign18, doc_assign19, doc_assign20,
                 doc_assign21, doc_assign22, doc_assign23, doc_assign24, doc_assign25,
                 doc_assign26, doc_assign27, doc_assign28, doc_assign29, doc_assign30,
                 doc_assign31, doc_assign32, doc_assign33, doc_assign34, doc_assign35,
                 doc_assign36, doc_assign37, doc_assign38, doc_assign39, doc_assign40,
                 doc_assign41, doc_assign42, doc_assign43, doc_assign44, doc_assign45,
                 doc_assign46, doc_assign47, doc_assign48, doc_assign49, doc_assign50])
session.commit()

# DependentOf
with Session(engine) as session:
        dependentOf1 = DependentOf(pID='5478828149', gID='8520489390')
        dependentOf2 = DependentOf(pID='2353169554', gID='7251298740')
        dependentOf3 = DependentOf(pID='2867034035', gID='6445290620')
        dependentOf4 = DependentOf(pID='1608930009', gID='7483012498')
        dependentOf5 = DependentOf(pID='1135007012', gID='3285231780')

(Continued…)

session.add_all([dependentOf1, dependentOf2, dependentOf3, dependentOf4, dependentOf5,
                 dependentOf6, dependentOf7, dependentOf8, dependentOf9, dependentOf10,
                 dependentOf11, dependentOf12, dependentOf13, dependentOf14, dependentOf15,
                 dependentOf16, dependentOf17, dependentOf18, dependentOf19, dependentOf20,
                 dependentOf21, dependentOf22, dependentOf23, dependentOf24, dependentOf25,
                 dependentOf26, dependentOf27, dependentOf28, dependentOf29, dependentOf30,
                 dependentOf31, dependentOf32, dependentOf33, dependentOf34, dependentOf35,
                 dependentOf36, dependentOf37, dependentOf38, dependentOf39, dependentOf40,
                 dependentOf41, dependentOf42, dependentOf43, dependentOf44, dependentOf45,
                 dependentOf46, dependentOf47, dependentOf48, dependentOf49, dependentOf50])
session.commit()
```

7.3 Simple Queries

Query 1 (Pragya):

Description: Find the patient ID, first name, last name, illness, and address of all patients who have an illness of “Pneumonia” and live with their guardian. Sort the outputs in ascending alphabetical order of the patient’s first name.

```
session = Session(engine)
print("\n--------------------")
print("\n## Query 1 (Pragya) ##\n")
result = (
    session.query(Patient)
    .join(DependentOf, Patient.pID == DependentOf.pID)
    .join(Guardian, DependentOf.gID == Guardian.gID)
    .filter(Patient.pIllness == 'Pneumonia')
    .order_by(Patient.pFirst, Patient.pLast)
    .all()
)
if not result:
    print("No patients with Pneumonia found.")
else:
    for patient in result:
        print(
            f"pID: {patient.pID}\nFirst Name: {patient.pFirst}\nLast Name: {patient.pLast}\n"
            f"Illness: {patient.pIllness}\nPatient Address: {patient.pAddress}\n\n"
        )
```

Query 1 Results:

```
## Query 1 (Pragya) ##

pID: 0219096953
First Name: Bruce
Last Name: Krystle
Illness: Pneumonia
Patient Address: 123 Elm Street, Chicago, IL 60601


pID: 5478828149
First Name: Carree
Last Name: Krystle
Illness: Pneumonia
Patient Address: 123 Elm Street, Chicago, IL 60601


pID: 3322578003
First Name: Corinna
Last Name: Elsworth
Illness: Pneumonia
Patient Address: 234 Birch Road, Aurora, IL 60501


pID: 0943273102
First Name: Erhart
Last Name: Enrique
Illness: Pneumonia
Patient Address: 87634 Londonderry Crossing, Naperville, IL 60540


pID: 6040373601
First Name: Fowler
Last Name: Dyson
Illness: Pneumonia
Patient Address: 555 Daisy Avenue, Elgin, IL 60120


pID: 0184960495
First Name: Ham
Last Name: Maulkin
Illness: Pneumonia
Patient Address: 565 Orchid Place, Skokie, IL 60076


pID: 4647884775
First Name: Janela
Last Name: Snare
Illness: Pneumonia
Patient Address: 33528 Tony Pass, Champaign, IL 61820


pID: 9517512813
First Name: Jessica
Last Name: Wheeler
Illness: Pneumonia
Patient Address: 777 Sunflower Drive, Oak Park, IL 60301


pID: 2353169554
First Name: Muhammad
Last Name: Wilhelmina
Illness: Pneumonia
Patient Address: 456 Maple Avenue, Springfield, IL 62701


pID: 1993446753
First Name: Reinhard
Last Name: Dockrill
Illness: Pneumonia
Patient Address: 404 Doe Crossing Way, Des Plaines, IL 60016


pID: 0500203431
First Name: Rriocard
Last Name: Leak
Illness: Pneumonia
Patient Address: 555 Daisy Avenue, Elgin, IL 60120


pID: 9244135949
First Name: Shaylyn
Last Name: Spear
Illness: Pneumonia
Patient Address: 111 Willow Lane, Champaign, IL 61820
```

Query 2 (Arub):

Description: For each guardian, print gLast, gBalance, gPhoneNum, and pID. Sort the output by the amount of money in each gBalance in descending order. Only print gBalance exceeding $10000.

```
session = Session(engine)
print("\n--------------------")
print("\n## Query2 (Arub) ##\n")
result = (
    session.query(
        Guardian.gLast,
        func.sum(Guardian.gBalance).label('TotalBalance'),
        Guardian.gPhoneNum,
        Dependent.pID
    )
    .join(Dependent, Guardian.gID == Dependent.gID)
    .group_by(Guardian.gID, Guardian.gLast, Guardian.gPhoneNum, Dependent.pID)
    .having(func.sum(Guardian.gBalance) > 10000)
    .order_by(func.sum(Guardian.gBalance).desc())
    .all()
)

for guardian in result:
    print(
        f"Last Name: {guardian.gLast}, "
        f"Total Balance: {guardian.TotalBalance}, "
        f"Phone Number: {guardian.gPhoneNum}, "
        f"Dependent ID: {guardian.pID}"
    )
```

Query 2 Results:

Query 3 (Rihab):

Description: Find the doctors that have patients who are female. List the doctor's first and last names, and the number of female patients each of them have. Sort by the number of female patients in ascending order.

```
print("\n--------------------")
print("\n## Query3 (Rihab) ##\n")
result = (
   session.query(Doctor.dFirst, Doctor.dLast,
func.count(AssignedToDoc.pID).label('numFemalePatients'))
   .join(AssignedToDoc, Doctor.dID == AssignedToDoc.dID)
   .join(Patient, AssignedToDoc.pID == Patient.pID)
   .filter(Patient.pSex == 'Female')
   .group_by(Doctor.dFirst, Doctor.dLast)
   .having(func.count(AssignedToDoc.pID) >= 1)
   .order_by('numFemalePatients')
   .all()
)
for doctor in result:
   print(f"Last Name: {doctor.dLast}\n"
   f"First Name: {doctor.dFirst}\n"
   f"Number of Female Patients: {doctor.numFemalePatients}\n"
   f"\n")
```

Query 3 Results:

```
## Query3 (Rihab) ##

Last Name: Roseanna
First Name: Kropach
Number of Female Patients: 1


Last Name: Rafe
First Name: Molan
Number of Female Patients: 1


Last Name: Nerti
First Name: Dorot
Number of Female Patients: 1


Last Name: Jeff
First Name: Woodhead
Number of Female Patients: 1


Last Name: Guendolen
First Name: Farmery
Number of Female Patients: 1


Last Name: Darryl
First Name: Meere
Number of Female Patients: 1


Last Name: Milena
First Name: Kornacki
Number of Female Patients: 1


Last Name: Fancy
First Name: Mepsted
Number of Female Patients: 2
```

```
Last Name: Ernst
First Name: Twigger
Number of Female Patients: 2


Last Name: Tamarah
First Name: Baybutt
Number of Female Patients: 2


Last Name: Dante
First Name: Wrassell
Number of Female Patients: 2


Last Name: Allissa
First Name: Gunson
Number of Female Patients: 2


Last Name: Malanie
First Name: Sherlock
Number of Female Patients: 2


Last Name: Geordie
First Name: Bushell
Number of Female Patients: 2


Last Name: Audi
First Name: Caldroni
Number of Female Patients: 2


Last Name: Prince
First Name: Coon
Number of Female Patients: 2
```

Query 4 (Kayla):
Description: Print the ID, last and first name, sex, and the number of patients a nurse has for female nurses with 3 or fewer patients. Group by nurse ID. Sort the output by last name in ascending order.

```
session = Session(engine)
print("\n--------------------")
print("\n## Query4 (Kayla) ##\n")
result = (
   session.query(Nurse.nID, Nurse.nLast, Nurse.nFirst, Nurse.nSex,
func.count(AssignedToNur.pID).label("numPatients"))
   .filter(Nurse.nSex == "Female")
   .join(AssignedToNur, Nurse.nID == AssignedToNur.nID)
   .group_by(Nurse.nID)
   .having(func.count(AssignedToNur.pID) <= 3)
   .order_by(Nurse.nLast)
   .all()
)
print("List of nurses and the number of patients they have:\n")
for nurse in result:
   print(
      f"nID: {nurse.nID}\n"
      f"Last Name: {nurse.nLast}\n"
      f"First Name: {nurse.nFirst}\n"
      f"Sex: {nurse.nSex}\n"
      f"Number of Patients: {nurse.numPatients}\n\n"
   )
```

Query 4 Results:

```
## Query 4 (Kayla) ##

List of nurses and the number of patients they have:

nID: 8699777841
Last Name: Bradburn
First Name: Babette
Sex: Female
Number of Patients: 2


nID: 7521039246
Last Name: Cass
First Name: Anica
Sex: Female
Number of Patients: 2


nID: 1637468687
Last Name: Laing
First Name: Jamie
Sex: Female
Number of Patients: 3


nID: 2152501608
Last Name: Maidment
First Name: Heddie
Sex: Female
Number of Patients: 2


nID: 1572910909
Last Name: Matejovsky
First Name: Michelle
Sex: Female
Number of Patients: 3


nID: 9405259113
Last Name: Rignoldes
First Name: Grayce
Sex: Female
Number of Patients: 3


nID: 6918254634
Last Name: Sabatini
First Name: Mareah
Sex: Female
Number of Patients: 1


nID: 5964724701
Last Name: Tuison
First Name: Darcy
Sex: Female
Number of Patients: 2


--------------------
```

GROUP STATUS REPORT

GROUP: Rihab, Pragya, Kayla, Arub **PHASE #:** 3

Dates & attendance at group meetings in this phase:

Saturday, Nov 25	6:30 PM - 11:00 PM	All group members present
Sunday, Nov 26	8:30 PM - 12:00 AM	All group members present
Monday, Nov 27	8:00 PM - 12:30 AM	Pragya, Kayla, Rihab

Overview of progress on project:

In this phase of the project, we wrote a DDL script and ORM implementation for our HOSPEDSPATIENTS database, which includes the classes ‘patient’, ‘guardian’, ‘doctor’, ‘nurse’, ‘assignedToNur’, ‘assignedToDoc’, and ‘dependentOf’. We then each wrote a query focusing on a class. Our deliverable includes the content from Phases 1 and 2 with minor updates, the DDL script and ORM implementation, the four queries, and the outputs of the queries.

CONTRIBUTIONS OF GROUP MEMBERS

Leader: Kayla Salerno

- Created ‘nurse’ and ‘assignedToDoc’ classes for the script
- Inserted data for the ‘nurse’ and ‘assignedToNur’ classes
- Debugged and reformatted script
- Wrote Query 4

Phase Recorder: Arub Sumayli

- Created ‘guardian’ class for the script
- Wrote Query 2

Phase Checker: Rihab Junagadhwala

- Created ‘doctor’ and ‘dependentOf’ classes for the script
- Inserted data for the ‘doctor’ and ‘dependentOf’ classes
- Debugged and reformatted script
- Wrote Query 3

Technical Advisor: Pragya Kansal

- Created ‘patient’ class for the script
- Inserted data for ‘assignedToDoc’ and ‘guardian’ classes
- Debugged and reformatted script
- Wrote Query 1