Healthcare Database Design

Deliverables:

- Description of the organization
- ER diagram with min/max specifications
- Constraints not in ER diagram
- Relational Schema
- Queries
- DDM + DML + SQL
- ORM Implementation

---

1. Introduction

The Weight Loss and Personal Nutrition (WLPN) database is designed to support the operations of a company that specializes in weight management, nutrition counseling, and personal training services. It is an organization with the aim to provide personalized weight loss programs through the integration of medical, nutritional, and fitness information to support the client's health goals.

2. Description of the Organization

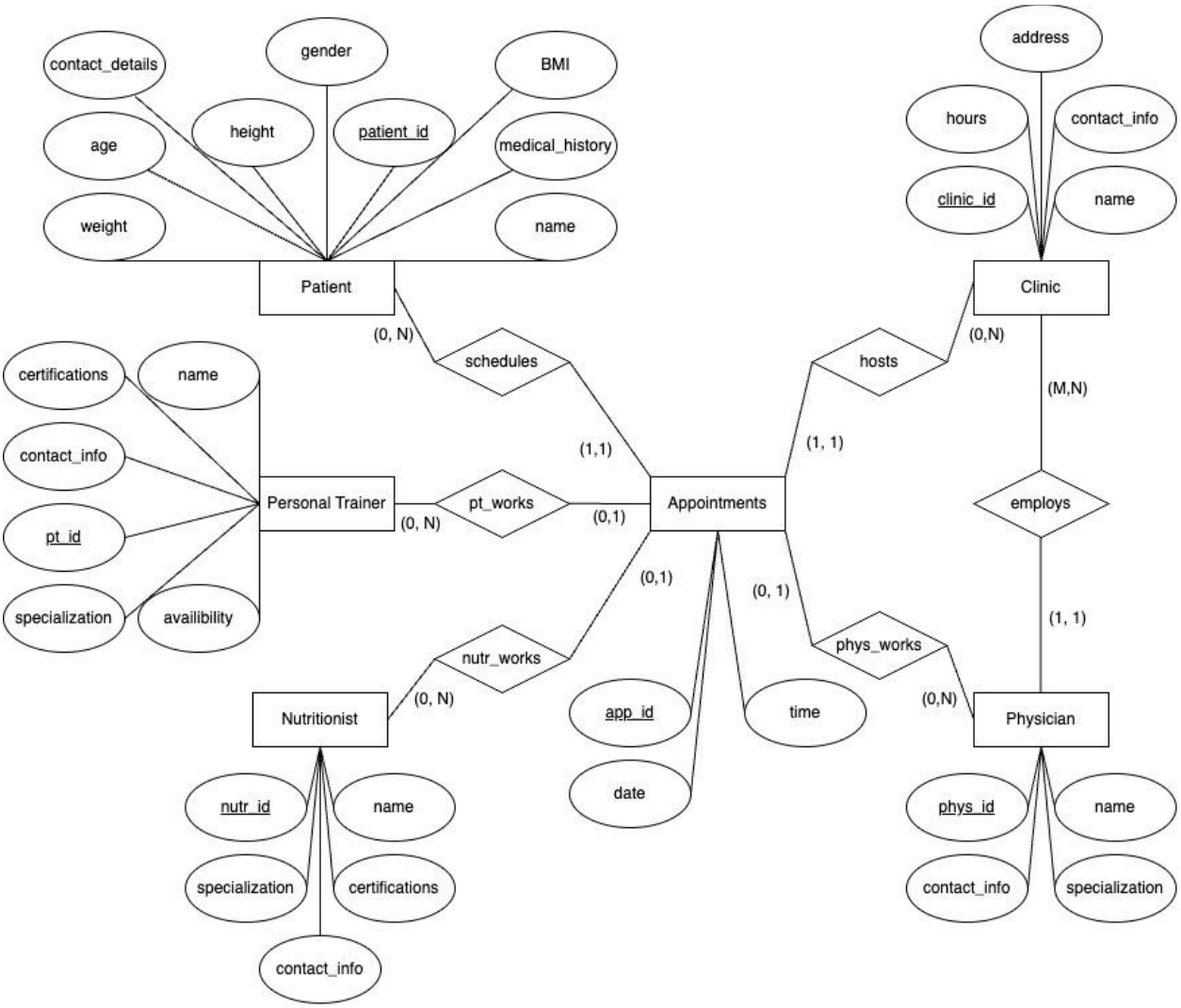
The WLPN Database stores important information regarding the organization's weight loss programs, medical professionals, fitness trainers, and patients. The database facilitates the coordination between the healthcare professionals, personal trainers, and their clients to deliver well-rounded and personalized weight loss solutions.

The database contains data of the following entities:

- **Patients**: table stores details of individuals enrolled in the weight loss program. The database stores each patient's unique ID, name, age, gender, contact details, weight, height, BMI, and medical history. Patients are linked to diet and workout plans as well as their appointments with healthcare professionals.
- **Clinics**: they represent the physical locations where consultations and medical evaluations are conducted. Each clinic has a unique ID, name, address, contact information, and operating hours.
- **Physicians**: they are responsible for overseeing the medical aspects of the weight loss programs. Each physician has a unique ID, name, specialization, contact information, and clinic association. Physicians are connected to appointments with patients and are responsible for approving or adjusting treatment plans.
- **Nutritionists**: these develop personalized diet plans for patients based on their health conditions and weight loss goals. The database includes each nutritionist's unique ID, name, certifications, specialties (clinical or sports nutrition), and contact information. Nutritionists are linked to the diet plans they create for individual patients.
- **Personal Trainers (PTs)**: they are responsible for creating and implementing workout plans tailored to each patient's fitness level and weight loss objectives. The database stores the trainer's unique ID, name, certification, specialization (e.g., strength training or cardio), and availability. Trainers are associated with workout plans scheduled with patients.
- **Appointments**: this tracks all interactions between patients and healthcare professionals (physicians, nutritionists, and trainers). Each appointment has a unique ID and includes patient ID, professional ID (either phys_id or nutr_id), clinic ID, date, time, and type of appointment (consultation, follow-up, or training session).
    - The ProfessionalID attribute in the appointments table can refer to either phys_id (Physician) or nutr_id (Nutritionist)
    - The Employs table supports a Many-to-Many relationship between Physicians and Clinics

3. ER Diagram

The figure below shows the ER diagram of the Weight Loss and Personal Nutrition (WLPN) database



4. ER Diagram Uncaptured Constraints

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

- Only one professional per appointment and not multiple per consultation so making sure others are null (Physician has appointment while trainer and nutritionist are set to null).
- The start time of appointments cannot be after the end time, and they must be within clinic operating hours
- Neither a professional nor a patient may be booked at the same time for an appointment. The only exception would be if parallel appointments for professionals were allowed.
- There should be valid certifications for each professional specialty so that appointments are booked correctly (Physician goes with FamilyMedicine, or Nutritionist goes with StrengthTraining).
- Professionals must be active status to allow bookings.
- Physicians must have a valid clinic_id (not null). The Employs table enforces this association if the many-to-many relationship is allowed.
- Age constraints should be added if there is a program that requires a certain age so a check on the patient's age might be required.
- Any calculations with medical history like BMI, should be generated with a calculator column to ensure accuracy.
- Any diet or workout plans should have start and end dates that should not overlap within each other but there can be one workout plan while there is a diet plan since they may be working together for the patient.
- Any plan can only be created or edited by the patient’s currently active nutritionist or trainer to ensure there is no one else modifying the patient's data.
- If there is an email or phone number, there should be a check to validate them (Phone number should match the normal pattern and an email should have a valid @ like gmail, etc...).
    - Phone Number: (XXX) XXX-XXXX
    - Email: example.name@gmail.com
- Any data like MedicalHistory, DietPlan, WorkoutPlan, should be restricted to certain users with correct permissions.
- There should be triggers in place to track any changes to certain medical records and appointments
- The system should also not allow the deletion of a patient if there are any existing appointments and or records that should be kept on file in case the patient needs them. Can be enforced with the use of FK constraints.

5. Relational Schema

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

5.1 Relational Schema with Referential Integrity

Patient(patient_id, name, medical_history _history, BMI, gender, height, pContactInfo, age, weight)

Personal trainer(pt_id, name, certifications, contact_info, specialization, availability

Nutritionist(nutr_id, name, specialization, certifications, nContactInfo)

Appointments(appt_id, patient_id, clinic_id, pt_id, nutr_id,phys_id, date, time)

Foreign key (patient_id) references patient (patient_id)

Foreign key (clinic_id) references clinic (clinic_id)

Foreign key (pt_id) references personal trainer (pt_id)

Foreign key (nutr_id) references nutritionist (nutr_id)

Foreign key (phys_id) references Physician (phys_id)

Clinic (clinic_id, address, hours, contact_info, name)

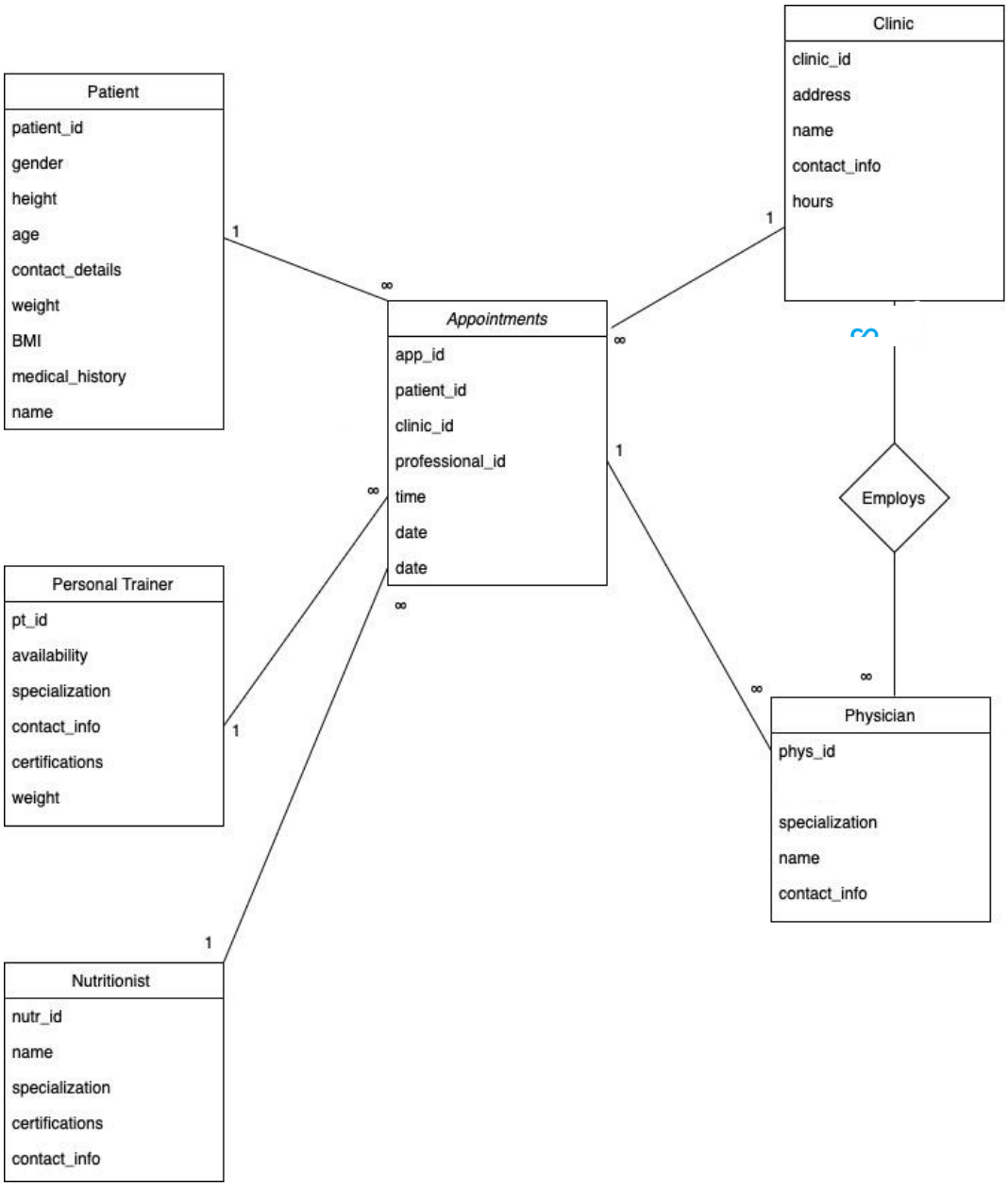
Physician (phys_id, clinic_id, name, specialization, contact_info)

Foreign key (clinic_id) references clinic (clinic_id)

Employs (phys_id, clinic_id)

Foreign key (phys_id) references physician (phys_id)

Foreign key (clinic_id) references clinic (phys_id)



5.2 Relational Table Details

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

Table Name	Attribute	Description
Patient	patient_id	Unique patient ID
	name	Full name of the patient
	medical_history	Medical history of the patient
	BMI	Body Mass Index of the patient
	gender	Gender of the patient
	height	Height of the patient
	pContactInfo	Contact information of the patient
	age	Age of the patient
	weight	Weight of the patient
Personal Trainer	pt_id	Unique personal trainer ID
	name	Full name of the personal trainer
	certifications	Certifications of the personal trainer
	contact_info	Contact information of the personal trainer
	specialization	Area of specialization
	availability	Availability schedule of the trainer
Nutritionist	nutr_id	Unique nutritionist ID
	name	Full name of the nutritionist
	specialization	Area of specialization

	certifications	Certifications of the nutritionist
	nContactInfo	Contact information of the nutritionist
Appointments	appt_id	Unique appointment ID
	patient_id	ID of the patient with the appointment
	clinic_id	ID of the clinic where the appointment takes place
	pt_id	ID of the personal trainer associated with the appointment
	nutr_id	ID of the nutritionist associated with the appointment
	phys_id	ID of the physician associated with the appointment
	date	Date of the appointment
	time	Time of the appointment
Clinic	clinic_id	Unique clinic ID
	address	Address of the clinic
	hours	Operating hours of the clinic
	contact_info	Contact information of the clinic
	name	Name of the clinic
Physician	phys_id	Unique physician ID
	name	Full name of the physician
	specialization	Specialization of the physician
	contact_info	Contact information of the physician
Employs	phys_id	ID of the physician employed at the clinic
	clinic_id	ID of the clinic employing the physician

6. Queries

The following table summarizes the queries in the EMPTRAINING Database.

Query Name	Description	Output	Relations Accessed
Number_of_appointments_per_physician	Find each physician's name along with the total number of appointments they have, ordered by appointment count.	• Physician_name • Appointment_count	• Physician • Appointments
Most_active_trainer	It will find the personal trainer with the most appointments	• Appointment_count • Trainer_name	• Appointments • Personal Trainer
BMI_Nutritionist_Analysis	It lists the pateints with an appointment with a nutritionist, listing them by BMI in descending order. Also lists their nutritionist and number of appointments	• Patient_nameame • BMI • Nutritionist_name • Appointment_count	• Patient • Appointments • Nutritionist
Avg_sessions_per_patient	Shows the average number of training sessions per patient that is completed, most active patients displayed first	• Avg_sessions • Patient_name	• Appointments • Patient • Personal Trainer

7. ORM Implementation

```
from typing import List, Optional
from sqlalchemy import String, Integer
from sqlalchemy.orm import DeclarativeBase, Mapped, mapped_column, relationship, Session
from sqlalchemy import create_engine, select, ForeignKey, String, Integer, Date, Time
import datetime

engine = create_engine(
"postgresql+psycopg2://postgres:132113@localhost:5433/pedroricardo"
)

# Classes created by Pedro Castro

class Base(DeclarativeBase):
pass

# Physician Table - created by Pedro (can have many appointments)
class Physician(Base):
__tablename__ = "physician"
phys_id: Mapped[int] = mapped_column(Integer, primary_key=True)
name: Mapped[str] = mapped_column(String(100))
specialization: Mapped[str] = mapped_column(String(100))
contact_info: Mapped[str] = mapped_column(String(100))
clinic_id: Mapped[int] = mapped_column(Integer)

appointments: Mapped[List["Appointment"]] = relationship(back_populates="physician")

def __repr__(self) -> str: #represents the object as a string
return f"Physician(phys_id={self.phys_id!r}, name={self.name!r},
specialization={self.specialization!r}, contact_info={self.contact_info!r},
clinic_id={self.clinic_id!r})"

# Appointment Table - created by Pedro (can have one physician)
class Appointment(Base):
```

```
    __tablename__ = "appointments"
    appt_id: Mapped[int] = mapped_column(Integer, primary_key=True)
    patient_id: Mapped[int] = mapped_column(Integer)
    clinic_id: Mapped[int] = mapped_column(Integer)
    pt_id: Mapped[int] = mapped_column(Integer, nullable=True)
    nutr_id: Mapped[int] = mapped_column(Integer, nullable=True)
    phys_id: Mapped[int] = mapped_column(ForeignKey("physician.phys_id"))
    date: Mapped[datetime.date] = mapped_column(Date)
    time: Mapped[datetime.time] = mapped_column(Time)



    physician: Mapped["Physician"] = relationship(back_populates="appointments")


    def __repr__(self) -> str: #represents the object as a string
    return f"Appointment(appt_id={self.appt_id!r}, patient_id={self.patient_id!r},
    clinic_id={self.clinic_id!r}, phys_id={self.phys_id!r}, date={self.date!r}, time={self.time!r})"



    Base.metadata.create_all(engine)
```

 # Classes created by Dylan Campbell

```
# Clinic Table - created by Dylan
class Clinic(Base):
   __tablename__ = "clinic"

   clinic_id: Mapped[int] = mapped_column(Integer, primary_key=True)
   name: Mapped[str] = mapped_column(String(100))
   address: Mapped[str] = mapped_column(String(200))
   hours: Mapped[str] = mapped_column(String(100))
   contact_info: Mapped[str] = mapped_column(String(100))

   # Relationship with physicians through employs table
   physicians: Mapped[List["Physician"]] = relationship(
      secondary="employs",
      back_populates="clinics"
   )

   def __repr__(self) -> str:
      return f"Clinic(clinic_id={self.clinic_id!r}, name={self.name!r}, address={self.address!r})"
```

```
# Employs Table (many-to-many relationship) - created by Dylan
class Employs(Base):
    __tablename__ = "employs"

    phys_id: Mapped[int] = mapped_column(ForeignKey("physician.phys_id"),
primary_key=True)
    clinic_id: Mapped[int] = mapped_column(ForeignKey("clinic.clinic_id"), primary_key=True)

    def __repr__(self) -> str:
        return f"Employs(phys_id={self.phys_id!r}, clinic_id={self.clinic_id!r})"
```

 # Classes created by Avery Walker

```
class Base(DeclarativeBase):
    pass


# Clinic Table - Made by Avery

class Clinic(Base):
    __tablename__ = "clinic"


    clinic_id: Mapped[int] = mapped_column(Integer, primary_key=True)
    name: Mapped[str] = mapped_column(String(100))
    address: Mapped[str] = mapped_column(String(200))
    hours: Mapped[str] = mapped_column(String(100))
    contact_info: Mapped[str] = mapped_column(String(100))


    # One clinic can have many appointments
    appointments: Mapped[list["Appointment"]] = relationship(back_populates="clinic")


    def __repr__(self):
        return f"Clinic(clinic_id={self.clinic_id}, name={self.name})"


# Appointments Table - Made by Avery
```

```
class Appointment(Base):
    __tablename__ = "appointments"

    appt_id: Mapped[int] = mapped_column(Integer, primary_key=True)
    patient_id: Mapped[int] = mapped_column(Integer)
    clinic_id: Mapped[int] = mapped_column(ForeignKey("clinic.clinic_id"))
    pt_id: Mapped[int] = mapped_column(Integer, nullable=True)
    nutr_id: Mapped[int] = mapped_column(Integer, nullable=True)
    phys_id: Mapped[int] = mapped_column(Integer, nullable=True)
    date: Mapped[datetime.date] = mapped_column(Date)
    time: Mapped[datetime.time] = mapped_column(Time)

    # Each appointment is linked to one clinic
    clinic: Mapped["Clinic"] = relationship(back_populates="appointments")

    def __repr__(self):
        return f"Appointment(appt_id={self.appt_id}, clinic_id={self.clinic_id}, date={self.date})"

Base.metadata.create_all(engine)
```

 # Classes created by Xander Estevez

```
from typing import List, Optional
from sqlalchemy import String, Integer, ForeignKey, Date, Time, create_engine, select, func
from sqlalchemy.orm import DeclarativeBase, Mapped, mapped_column, relationship, Session
import datetime


engine = create_engine("postgresql+psycopg2://postgres:Comp353$@localhost:5433/postgres")

class Base(DeclarativeBase):
    pass

# Personal Trainer Table – Made by Xander
```

```
class PersonalTrainer(Base):

    __tablename__ = "personal_trainer"

    pt_id: Mapped[int] = mapped_column(Integer, primary_key=True)
    name: Mapped[str] = mapped_column(String(100))
    certifications: Mapped[str] = mapped_column(String(100))
    specialization: Mapped[str] = mapped_column(String(100))
    availability: Mapped[str] = mapped_column(String(100))
    contact_info: Mapped[str] = mapped_column(String(100))

    appointments: Mapped[List["Appointment"]] = relationship(back_populates="trainer")

    def __repr__(self) -> str:
        return f"PersonalTrainer(pt_id={self.pt_id!r}, name={self.name!r})"


# Appointments Table – Made by Xander

class Appointment(Base):

    __tablename__ = "appointments"

    appt_id: Mapped[int] = mapped_column(Integer, primary_key=True)
    patient_id: Mapped[int] = mapped_column(Integer)
    clinic_id: Mapped[int] = mapped_column(Integer)
    pt_id: Mapped[Optional[int]] = mapped_column(ForeignKey("personal_trainer.pt_id"),
    nullable=True)
    nutr_id: Mapped[Optional[int]] = mapped_column(Integer, nullable=True)
    phys_id: Mapped[Optional[int]] = mapped_column(Integer, nullable=True)
    date: Mapped[datetime.date] = mapped_column(Date)
    time: Mapped[datetime.time] = mapped_column(Time)


    trainer: Mapped[Optional["PersonalTrainer"]] = relationship(back_populates="appointments")


    def __repr__(self) -> str:
        return f"Appointment(appt_id={self.appt_id!r}, pt_id={self.pt_id!r}, date={self.date!r})"


    Base.metadata.create_all(engine)
```

 # Classes created by Liam Rich

#Patient class defined by Liam Rich

class Patient(Base): **tablename** = "patient"

patient_id: Mapped[int] = mapped_column(Integer, primary_key=True)
name: Mapped[str] = mapped_column(String(100))
medical_history: Mapped[Optional[str]] = mapped_column(String, nullable=True)
BMI: Mapped[Optional[float]] = mapped_column(Float, nullable=True)
gender: Mapped[Optional[str]] = mapped_column(String(10), nullable=True)
height: Mapped[Optional[float]] = mapped_column(Float, nullable=True)
pContactInfo: Mapped[Optional[str]] = mapped_column(String(100), nullable=True)
age: Mapped[int] = mapped_column(Integer)
weight: Mapped[Optional[float]] = mapped_column(Float, nullable=True)

Relationship to appointments
appointments: Mapped[List["Appointment"]] = relationship(back_populates="patient")

def __repr__(self) -> str:
  return f"Patient(patient_id={self.patient_id!r}, name={self.name!r}, age={self.age!r},
BMI={self.BMI!r})"

#Nutritionist class defined by Liam Rich

class Nutritionist(Base): **tablename** = "nutritionist"

nutr_id: Mapped[int] = mapped_column(Integer, primary_key=True)
name: Mapped[str] = mapped_column(String(100))
specialization: Mapped[Optional[str]] = mapped_column(String(100), nullable=True)
certifications: Mapped[Optional[str]] = mapped_column(String, nullable=True)
nContactInfo: Mapped[Optional[str]] = mapped_column(String(100), nullable=True)

Relationships
appointments: Mapped[List["Appointment"]] = relationship(back_populates="nutritionist")

def __repr__(self) -> str:
  return f"Nutritionist(nutr_id={self.nutr_id!r}, name={self.name!r},

```
specialization={self.specialization!r})"
```

 # Object creation (data insertion) – Pedro Castro

```
with Session(engine) as session:
# Insert physicians
dr_smith = Physician(
name="Dr. Smith",
specialization="Family Medicine",
contact_info="drsmith@example.com",
clinic_id=1
)
dr_jones = Physician(
name="Dr. Jones",
specialization="Endocrinology",
contact_info="drjones@example.com",
clinic_id=2
)

# Insert appointments
appt1 = Appointment(
patient_id=2,
clinic_id=2,
pt_id=2,
nutr_id=2,
phys_id=2,
date="2025-03-21",
time="11:30:00"
)
appt2 = Appointment(
patient_id=3,
clinic_id=1,
pt_id=None,
nutr_id=1,
```

```
    phys_id=1,
    date="2025-03-22",
    time="09:00:00"
    )
    appt3 = Appointment(
    patient_id=5,
    clinic_id=2,
    pt_id=1,
    nutr_id=2,
    phys_id=2,
    date="2025-03-24",
    time="12:00:00"
    )
    appt4 = Appointment(
    patient_id=6,
    clinic_id=1,
    pt_id=None,
    nutr_id=1,
    phys_id=1,
    date="2025-03-25",
    time="15:00:00"
    )
    appt5 = Appointment(
    patient_id=8,
    clinic_id=1,
    pt_id=2,
    nutr_id=2,
    phys_id=1,
    date="2025-03-27",
    time="09:30:00"
    )

    session.add_all([dr_smith, dr_jones, appt1, appt2, appt3, appt4, appt5])
    session.commit()
```

 # Object creation (data insertion) – Dylan Campbell

```
with Session(engine) as session:
    # Insert clinics
    clinic1 = Clinic(
        name="Downtown Wellness Center",
        address="123 Main St, Chicago, IL",
        hours="9:00-17:00 Mon-Fri",
        contact_info="info@downtownwellness.com"
    )

    clinic2 = Clinic(
        name="Lakeview Health Clinic",
        address="456 Park Ave, Chicago, IL",
        hours="8:00-19:00 Mon-Sat",
        contact_info="contact@lakeviewhealth.com"
    )

    clinic3 = Clinic(
        name="North Side Medical Center",
        address="789 Wilson Blvd, Chicago, IL",
        hours="7:00-18:00 Mon-Sun",
        contact_info="info@northsidemedical.com"
    )

    # Add clinics to session
    session.add_all([clinic1, clinic2, clinic3])
    session.commit()

    # Create employs relationships
    employs_relationships = [
        Employs(phys_id=1, clinic_id=1),  # Dr. Smith at Downtown
        Employs(phys_id=1, clinic_id=2),  # Dr. Smith also at Lakeview
        Employs(phys_id=2, clinic_id=2),  # Dr. Jones at Lakeview
        Employs(phys_id=2, clinic_id=3),  # Dr. Jones also at North Side
        Employs(phys_id=3, clinic_id=1),  # Dr. Lee at Downtown
        Employs(phys_id=4, clinic_id=3)   # Dr. Rodriguez at North Side
    ]

    session.add_all(employs_relationships)
    session.commit()
```

 # Object creation (data insertion) – Avery Walker

```
with Session(engine) as session:
# Add Clinic Info
      clinic1 = Clinic(
       name="Chicago Running Institute",
       address="111 N Wabash Ave, Chicago, IL",
```

```
        hours="8:00-18:00 Mon-Fri",
        contact_info="appts@chicagorunning.com"
    )
    clinic2 = Clinic(
        name="Gold Coast Plastic Surgery",
        address="303 Institute Pl, Chicago, IL",
        hours="9:00-17:00 Mon-Sat",
        contact_info="appts@gcplastic.com"
    )

    session.add_all([clinic1, clinic2])
    session.commit()

    # Add appointments associated with each clinic
    appointments = [
        Appointment(patient_id=1, clinic_id=clinic1.clinic_id, date=datetime.date(2025, 4, 15),
time=datetime.time(9, 0)),
        Appointment(patient_id=2, clinic_id=clinic1.clinic_id, date=datetime.date(2025, 4, 16),
time=datetime.time(10, 30)),
        Appointment(patient_id=3, clinic_id=clinic2.clinic_id, date=datetime.date(2025, 4, 17),
time=datetime.time(11, 15)),
        Appointment(patient_id=4, clinic_id=clinic2.clinic_id, date=datetime.date(2025, 4, 18),
time=datetime.time(13, 45)),
        Appointment(patient_id=5, clinic_id=clinic2.clinic_id, date=datetime.date(2025, 4, 19),
time=datetime.time(15, 0)),
    ]

    session.add_all(appointments)
    session.commit()
```

 # Object creation (data insertion) – Xander Estevez

```
with Session(engine) as session:

    t1 = PersonalTrainer(name="John Doe", certifications="Certified PT", specialization="Strength",
availability="Weekdays", contact_info="john@example.com")

    t2 = PersonalTrainer(name="Emily Carter", certifications="Certified PT", specialization="Cardio",
availability="Weekends", contact_info="emily@example.com")

    session.add_all([t1, t2])
    session.flush()


    appointments = [
        Appointment(patient_id=1, clinic_id=1, pt_id=t1.pt_id, date=datetime.date(2025, 4, 1),
    time=datetime.time(10, 0)),
```

```
    Appointment(patient_id=2, clinic_id=1, pt_id=t1.pt_id, date=datetime.date(2025, 4, 2),
time=datetime.time(11, 0)),
    Appointment(patient_id=3, clinic_id=2, pt_id=t2.pt_id, date=datetime.date(2025, 4, 3),
time=datetime.time(12, 0)),
    Appointment(patient_id=4, clinic_id=1, pt_id=t1.pt_id, date=datetime.date(2025, 4, 4),
time=datetime.time(13, 0)),
    Appointment(patient_id=5, clinic_id=2, pt_id=t2.pt_id, date=datetime.date(2025, 4, 5),
time=datetime.time(14, 0)),
    ]

    session.add_all(appointments)
    session.commit()
```

 # Object creation (data insertion) – Liam Rich

```
# Insert patients - Liam Rich
with Session(engine) as session:
    # Insert patients
    patients = [
        Patient(
            name="Alice Johnson",
            medical_history="Hypertension",
            BMI=24.5,
            gender="Female",
            height=165,
            pContactInfo="alice@example.com",
            age=30,
            weight=68
        ),
        Patient(
```

```
        name="Bob Williams",
        medical_history="Diabetes",
        BMI=28.3,
        gender="Male",
        height=175,
        pContactInfo="bob@example.com",
        age=45,
        weight=85
    ),
    Patient(
        name="Charlie Brown",
        medical_history="Asthma",
        BMI=22.0,
        gender="Male",
        height=180,
        pContactInfo="charlie@example.com",
        age=28,
        weight=75
    ),
    Patient(
        name="Diana Prince",
        medical_history="Anemia",
        BMI=21.5,
        gender="Female",
        height=170,
        pContactInfo="diana@example.com",
        age=33,
        weight=60
    ),
    Patient(
        name="Ethan Hunt",
        medical_history="High Cholesterol",
        BMI=26.7,
        gender="Male",
        height=182,
```

```
        pContactInfo="ethan@example.com",
        age=40,
        weight=88
    ),
    Patient(
        name="Fiona Glenanne",
        medical_history="Thyroid issues",
        BMI=23.0,
        gender="Female",
        height=168,
        pContactInfo="fiona@example.com",
        age=35,
        weight=62
    ),
    Patient(
        name="George Bailey",
        medical_history="Migraine",
        BMI=24.0,
        gender="Male",
        height=174,
        pContactInfo="george@example.com",
        age=38,
        weight=80
    ),
    Patient(
        name="Helen Parr",
        medical_history="Back Pain",
        BMI=25.0,
        gender="Female",
        height=167,
        pContactInfo="helen@example.com",
        age=42,
        weight=70
    )
]
```

```
    session.add_all(patients)
    session.commit()

# Insert nutritionists - Liam Rich
with Session(engine) as session:
    nutritionists = [
        Nutritionist(
            name="Jane Smith",
            specialization="Sports Nutrition",
            certifications="Certified Nutritionist",
            nContactInfo="janesmith@example.com"
        ),
        Nutritionist(
            name="Tom Green",
            specialization="Clinical Nutrition",
            certifications="Certified Nutritionist",
            nContactInfo="tomgreen@example.com"
        ),
        Nutritionist(
            name="Sarah Johnson",
            specialization="Weight Management",
            certifications="Registered Dietitian",
            nContactInfo="sarahjohnson@example.com"
        ),
        Nutritionist(
            name="Michael Chen",
            specialization="Metabolic Health",
            certifications="Clinical Nutritionist",
            nContactInfo="michaelchen@example.com"
        )
    ]

    session.add_all(nutritionists)
    session.commit()
```

 # Query and Output: Join Appointment and Physician to list appointments - Pedro Castro

```
session = Session(engine)

print("## Pedro Castro Query##")

with Session(engine) as session:
stmt = (
select(Appointment, Physician)
.join(Appointment.physician)
.order_by(Appointment.date)
.limit(15)
)
results = session.execute(stmt).all()

for appt, phys in results:
print(f"{appt.date} at {appt.time} - {phys.name} ({phys.specialization})")
```

```
## Pedro Castro Query##
2025-03-21 at 11:30:00 - Dr. Jones (Endocrinology)
2025-03-21 at 11:30:00 - Dr. Jones (Endocrinology)
2025-03-21 at 11:30:00 - Dr. Jones (Endocrinology)
2025-03-22 at 09:00:00 - Dr. Smith (Family Medicine)
2025-03-22 at 09:00:00 - Dr. Smith (Family Medicine)
2025-03-22 at 09:00:00 - Dr. Smith (Family Medicine)
2025-03-23 at 14:00:00 - Dr. Lee (Cardiology)
2025-03-24 at 12:00:00 - Dr. Jones (Endocrinology)
2025-03-24 at 12:00:00 - Dr. Jones (Endocrinology)
2025-03-24 at 12:00:00 - Dr. Jones (Endocrinology)
2025-03-25 at 15:00:00 - Dr. Smith (Family Medicine)
2025-03-25 at 15:00:00 - Dr. Smith (Family Medicine)
2025-03-25 at 15:00:00 - Dr. Lee (Cardiology)
2025-03-26 at 13:00:00 - Dr. Patel (Orthopedics)
2025-03-27 at 09:30:00 - Dr. Smith (Family Medicine)
```