Prompts

Prompt #1:
Given the following Python function to create a database

```
def create_database():
    Base = declarative_base()

    friendships = Table('friendships', Base.metadata,
                        Column('person_id', Integer, ForeignKey('people.id'), primary_key=True),
                        Column('friend_id', Integer, ForeignKey('people.id'), primary_key=True))

    club_members = Table('club_members', Base.metadata,
                         Column('person_id', Integer, ForeignKey('people.id'), primary_key=True),
                         Column('club_id', Integer, ForeignKey('clubs.id'), primary_key=True))

    class Person(Base):
        __tablename__ = 'people'
        id = Column(Integer, primary_key=True)
        name = Column(String)
        age = Column(Integer)
        gender = Column(String)
        location = Column(String)
        friends = relationship("Person",
                               secondary=friendships,
                               primaryjoin=id == friendships.c.person_id,
                               secondaryjoin=id == friendships.c.friend_id)
        clubs = relationship("Club", secondary=club_members, back_populates="members")

    class Club(Base):
        __tablename__ = 'clubs'
        id = Column(Integer, primary_key=True)
        description = Column(String)
        members = relationship("Person", secondary=club_members, back_populates="clubs")

    if os.path.exists("social_network.db"):
        os.remove("social_network.db")
    engine = create_engine(f'sqlite:///{"social_network.db"}', echo=False)
    Base.metadata.create_all(engine)

    Session = sessionmaker(bind=engine)
    session = Session()

    return session, Club, Person, friendships, club_members
```

Explain what is going on

Prompt #2:

Given the following python code:

```
# Function to load data from CSV into the database
def load_data_from_csv(session, Club, Person, friendships, club_members,
csv_path="members.csv"):
   # Step 1: Clear existing data from all relevant tables
   session.query(Person).delete()
   session.query(Club).delete()
   session.query(friendships).delete()
   session.query(club_members).delete()

   session.commit()  # Commit the deletion of all existing records

   # Load the CSV data
   df = pd.read_csv("members.csv", converters = {'Friendships': eval, "Clubs": eval})
   ### START ADDING CODE HERE ###
```

Modify the code for load_data_from_csv function to populate the database. To do that, you'll use the members.csv file which is available. The data is stored in a single table with one row for each person. The Friendships column contains the IDs of everyone the person in that row considers a friend. The Clubs column contains the names of each club the person in that row is a part of.

Prompt #3:

The code produced the following error:

```
AttributeError                            Traceback (most recent call last)
Cell In[11], line 3
      1 # The code below creates the database and reads in the data
      2 session, Club, Person, friendships, club_members = create_database()
----> 3 load_data_from_csv(session, Club, Person, friendships, club_members, "members.csv")
      5 # If your load_data_from_csv function is working correctly, then you should have read in
data correctly into all four tables in the database.
      7 print_amount = 3

Cell In[10], line 49, in load_data_from_csv(session, Club, Person, friendships, club_members,
csv_path)
    47 # Step 3: Establish friendships
    48 for person in session.query(Person).all():
---> 49     for friend_id in person.friend_ids:
    50         friend = session.query(Person).get(friend_id)
    51         if friend:

AttributeError: 'Person' object has no attribute 'friend_ids'
```

Please help to rectify

Prompt #4:

Write a function called get_club_members_by_description. This function shall query the data in the database. This function should accept a description of a club and a session, and return a list of all its members. Ensure that this function returns a list containing the defined Person objects. It must input only a club description.

Prompt #5:

Write a function named get_friends_of_person. This function should accept the name of a person and a session, return a list of all the people they consider to be friends. Ensure that this function returns a list containing the defined Person objects. The input must be only the name of a person.

Prompt #6:

Write a function called get_persons_who_consider_them_friend. This function should take two parameters: the name of an indiviual and a session. It will return a list of people who count this individual as a friend. It's important to remember that in the database, friendship isn't necessarily mutual. For example, Alice might consider Bob a friend, but Bob might not feel the same way about Alice. The function must return a list of Person objects for everyone who considers the input name their friend. The input to this function should strictly be the name of the person you're inquiring about