PROJECT PART 3
Translate the logical data model for the Oracle Enterprise DBMS

https://github.com/nicolefang1/CSC423/

a. Develop SQL code to create the entire database schema, reflecting the constraints identified in previous steps

```
query = """
    -- Create Clinic
    CREATE TABLE IF NOT EXISTS Clinic (
        clinicNo integer NOT NULL primary key,
        clinicName text,
        address text NOT NULL,
        telephone int NOT NULL);
    """
cursor.execute(query)
query = """
    -- Create Staff
    CREATE TABLE IF NOT EXISTS Staff (
        staffNo integer NOT NULL primary key,
        staffName text,
        address text,
        telephone int,
        DOB date,
        staffPos text,
        salary integer,
        clinicNo integer,
        FOREIGN KEY (clinicNo) REFERENCES Clinic(clinicNo));
    """
cursor.execute(query)
query = """
    -- Create Owner
    CREATE TABLE IF NOT EXISTS Owner (
        ownerNo integer NOT NULL primary key,
        ownerName text,
        address text,
        telephone int);
    """
cursor.execute(query)
query = """
    -- Create Pet
    CREATE TABLE IF NOT EXISTS Pet (
        petNo integer NOT NULL primary key,
```

```
        petName text,
        DOB date,
        species text,
        breed text,
        color text,
        ownerNo integer,
        FOREIGN KEY (ownerNo) REFERENCES Owner(ownerNo));
    """
cursor.execute(query)
query = """
    -- Create Examination
    CREATE TABLE IF NOT EXISTS Examination (
        examNo integer NOT NULL primary key,
        chiefComplaint text,
        description text,
        dateSeen date,
        actions text,
        clinicNo integer,
        petNo integer,
        FOREIGN KEY (clinicNo) REFERENCES Clinic(clinicNo),
        FOREIGN KEY (petNo) REFERENCES Pet(petNo));
    """
cursor.execute(query)
```

The cursor could only execute one statement at a time, and for easier debugging, I split the create tables into their own executed queries. In python, foreign keys are constraints added at the end as “foreign key (fk_attribute) references refTable(refAttribute)” as opposed to the “fk_attribute references refTable(refAttribute)” in Oracle. Instead of varchar(255), I used text. Instead of int, I used integer. Date was left relatively untouched, and I removed some check constraints when they didn’t work with the create.

---

b. Create at least 5 tuples for each relation in your database

```
clinics = [(1000, 'Rural Vets', '123rd St', '8002343456'),
           (2000, 'Paws Clinic', '45th St', '8004567890'),
           (3000, 'ABCDogs', '786th Ave', '8001231234'),
           (4000, 'ABCats', '135th Blvd', '8001237890'),
           (5000, 'PetCare Clinic', '10th st', '8008909753')]
cursor.executemany('INSERT INTO Clinic VALUES(?,?,?,?);', clinics)

staff = [(1001, 'Alice Baker', 'address1', '1232343456', '1990-11-16', 'Manager', 10000, 1000),
         (1002, 'Bob Carp', 'address2', '1234567890', '1998-12-06', 'Groomer', 5000, 1000),
         (1003, 'Charlie Davis', 'address3', '1231231234', '1980-07-24', 'Vet', 11000, 1000),
```

```
        (1004, 'Danielle Espina', 'address4', '1230987654', '1983-01-30', 'Vet', 11000, 1000),
        (1005, 'Eve Fisher', 'address5', '1239876543', '2000-04-01', 'Groomer', 5000, 1000)]
cursor.executemany('INSERT INTO Staff VALUES(?,?,?,?,?,?,?,?)', staff)

owners = [(1101, 'Alice', 'address1', '1232343456'),
         (1102, 'Barbara', 'address6', '1231234567'),
         (1103, 'Clara', 'address7', '1230987654'),
         (1104, 'Damian', 'address8', '1234567456'),
         (1105, 'Edna', 'address9', '8901234567')]
cursor.executemany('INSERT INTO Owner VALUES(?,?,?,?);', owners)

pets = [(1011, 'Spots', '2010-01-01', 'Dog', 'Dalmatian', 'White', 1101),
       (1012, 'Boots', '2012-02-02', 'Cat', 'Tabby', 'Orange', 1102),
       (1013, 'Oreo', '2018-03-03', 'Cat', 'Shorthair', 'Gray', 1103),
       (1014, 'Sparky', '2012-02-29', 'Dog', 'Bicheom', 'White', 1104),
       (1015, 'Nemo', '2010-05-05', 'Fish', 'Goldfish', 'Gold', 1105)]
cursor.executemany('INSERT INTO Pet VALUES(?,?,?,?,?,?,?);', pets)

exams = [(1111, 'XYZ', 'ABC', '2022-12-01', 'ZZZ', 1000, 1011),
        (1112, 'WXY', 'ABC', '2022-12-02', 'YYY', 1000, 1012),
        (1113, 'VWX', 'ABC', '2022-12-03', 'XXX', 1000, 1013),
        (1114, 'UVW', 'ABC', '2022-12-04', 'VVV', 1000, 1014),
        (1115, 'TUV', 'ABC', '2022-12-05', 'UUU', 1000, 1015)]
cursor.executemany('INSERT INTO Examination VALUES(?,?,?,?,?,?,?);', exams)
```

I used executemany to insert multiple rows into the table. This part was the most infuriating due to missed commas and semicolons and the persistent “sqlite3.ProgrammingError: Incorrect number of bindings supplied” error. Other than that, the date was changed from to_date(‘mm/dd/yyyy’) to just the standard ‘yyyy-mm-dd’ to avoid messing with the datetime class.

---

c. Develop 5 SQL queries using embedded SQL (see Python tutorial)

1) List all staff that work at clinic address 123rd St:

```
Select s.*
From clinic c, staff s
Where c.clinicNo = s.clinicNo and c.address like '123rd St';
```

2) List all owners of pets that had an exam on Dec-01-22:

```
Select o.ownerName
From examination e, pet p, owner o
Where e.petNo = p.petNo and p.ownerNo = o.ownerNo and dateSeen =
'2022-12-01';
```

3) Show all information on the pets of owners with the first name 'Alice':

```
Select p.*
From Pet p, Owner o
Where p.ownerNo = o.ownerNo AND ownerName like 'Alice%';
```

4) Find the number of examinations of dogs with complaint 'XYZ':

```
Select count(*)
From Pet p, Examination e
Where p.petNo = e.petNo and species like 'Dog' and chiefComplaint like 'XYZ';
```

5) Show all information on owners who had a 'ZZZ' action during an exam:

```
Select o.*
From Examination e, Pet p, Owner o
Where p.ownerNo = o.ownerNo and e.petNo = p.petNo and e.actions like 'ZZZ';
```

```
/Users/nicolefang/PycharmProjects/csc423/venv/bin/python /Users/nicolefang/PycharmProjects/c
   staffNo        staffName
0     1001      Alice Baker
1     1002         Bob Carp
2     1003    Charlie Davis
3     1004  Danielle Espina
4     1005       Eve Fisher
Index(['staffNo', 'staffName'], dtype='object')
---------------------------
  ownerName
0     Alice
Index(['ownerName'], dtype='object')
---------------------------
   petNo petName         DOB species      breed  color  ownerNo
0   1011   Spots  2010-01-01     Dog  Dalmatian  White     1101
Index(['petNo', 'petName', 'DOB', 'species', 'breed', 'color', 'ownerNo'], dtype='object')
---------------------------
   count(*)
0         1
---------------------------
   ownerNo ownerName   address   telephone
0     1101     Alice  address1  1232343456
Index(['ownerNo', 'ownerName', 'address', 'telephone'], dtype='object')
---------------------------
```

---

d. Upload all the code and documentation to GitHub.

https://github.com/nicolefang1/CSC423/

Rather than Visual Studio Code, I used PyCharm CE due to my familiarity with it.