Assignment 2
Sarah Cummings
CSC455

Part 1: Decompose the Following Schema into 3NF(License Number, Renewed, Status, Status Date, Driver Type, License Type, Original Issue Date, Name, Sex, Chauffer City, Chauffer State, Record Number)

License(License Number, Renewed, Status, Status Date, Driver Type, License Type, Original Issue Date, Name, Sex, Chauffer City, Record Number)

Chauffeur(Chauffeur City, Chauffer State)

SQL DDL to create the Tables:

```
--create license table
CREATE TABLE License(
        LicenseNumber NUMBER(10),
        Renewed VARCHAR(10),
        Status VARCHAR(10),
        StatusDate VARCHAR(10),
        DriverType VARCHAR(20),
        LicenseType VARCHAR(10),
        OriginalIssueDate VARCHAR(12),
        Name VARCHAR(20),
        Sex VARCHAR(6),
        ChaufferCity VARCHAR(20),
        RecordNumber VARCHAR(15),
CONSTRAINT LicenseFK
        Foreign Key(ChauffeurCity)
        REFERENCES Chauffeur(ChauffeurCity),
CONSTRAINT LicensePK
        Primary Key(RecordNumber)
);
--create second table
CREATE TABLE Chauffeur(
        ChauffeurCity VARCHAR(20),
        ChauffeurState VARCHAR(2),
CONSTRAINT ChauffeurPK
        Primary Key(ChauffeurCity)
);
```

Part 2: also see submitted python file, which contains code that is copied below

```
import sqlite3

#Connect to database
conn=sqlite3.connect('csc455.db')

#Request a cursor from the database
c=conn.cursor()

LicenseTable= '''CREATE TABLE License
(
   LicenseNumber NUMBER(10),
   Renewed VARCHAR(10),
   Status VARCHAR(10),
   StatusDate VARCHAR(10),
   DriverType VARCHAR(20),
   LicenseType VARCHAR(10),
   OriginalIssueDate VARCHAR(12),
   Name VARCHAR(20),
   Sex VARCHAR(6),
   ChauffeurCity VARCHAR(20),
   ChauffeurState VARCHAR(2),
   RecordNumber VARCHAR(15),
   CONSTRAINT LicenseFK
     Foreign Key(ChauffeurCity)
   REFERENCES Chauffeur(ChauffeurCity),
   CONSTRAINT LicensePK
     Primary Key(RecordNumber)
   );'''



ChauffeurTable= '''CREATE TABLE Chauffeur
(
   ChauffeurCity VARCHAR(20),
   ChauffeurState VARCHAR(2),
   CONSTRAINT ChauffeurPK
      Primary Key(ChauffeurCity)
);'''

#Drop tables if they exists
c.execute("DROP TABLE IF EXISTS License")
c.execute("DROP TABLE IF EXISTS Chauffeur")
```

```
#Create the tables
c.execute(LicenseTable)
c.execute(ChauffeurTable)

#Open and read file
fd= open('Public_Chauffeurs_Short.csv','r')
#Read all lines from the file into variable
allLines= fd.readlines()
fd.close() #close file

#Create new variable that eliminates First Row of file
dataLines= allLines[1:]

#Replace Null strings with None
for line in dataLines:
   valueList = line.strip().split(',')
   for n,i in enumerate(valueList):
      if i == 'NULL':
         valueList[n] = None
   #Add all data into SQL table
   c.execute("INSERT INTO License VALUES (?,?,?,?,?,?,?,?,?,?,?,?);", valueList)



#Take Distinct ChauffeurCity and ChauffeurState data from License table and add it to
Chauffeur Table
c.execute("INSERT INTO Chauffeur(ChauffeurCity, ChauffeurState) SELECT DISTINCT
ChauffeurCity, ChauffeurState FROM License;")



#Create new license table without Chauffeur State Data
License1NF = '''CREATE TABLE License2
(
   LicenseNumber NUMBER(10),
   Renewed VARCHAR(10),
   Status VARCHAR(10),
   StatusDate VARCHAR(10),
   DriverType VARCHAR(20),
   LicenseType VARCHAR(10),
   OriginalIssueDate VARCHAR(12),
   Name VARCHAR(20),
```

```
   Sex VARCHAR(6),
   ChauffeurCity VARCHAR(20),
   RecordNumber VARCHAR(15),
   CONSTRAINT LicenseFK
     Foreign Key(ChauffeurCity)
   REFERENCES Chauffeur(ChauffeurCity),
   CONSTRAINT LicensePK
     Primary Key(RecordNumber)
   );'''

#Drop new license table if it exists
c.execute("DROP TABLE IF EXISTS License2")
#Create the table
c.execute(License1NF)
#Insert all columns from original License table except ChauffeurState
c.execute('''INSERT INTO License2(LicenseNumber,
   Renewed, Status, StatusDate, DriverType, LicenseType,
   OriginalIssueDate, Name, Sex, ChauffeurCity, RecordNumber)
   SELECT LicenseNumber, Renewed, Status, StatusDate, DriverType,
   LicenseType, OriginalIssueDate, Name, Sex, ChauffeurCity, RecordNumber FROM
License;''')

#Get rid of previous license table
c.execute("DROP TABLE License")


conn.commit()
conn.close()
```

Part 3: Queries to answer the following questions:

1. Find all the animals (their names) that take less than 1.5 hours to feed.
   ```
   SELECT AName FROM Animal
       WHERE TimeToFeed < 1.5;
   ```

2. Find all the rare animals and sort the query output by feeding time (any direction)
   ```
   SELECT AName, TimeToFeed FROM Animal
       WHERE ACategory = 'rare'
       ORDER BY TimeToFeed;
   ```

3. Find the animal names and categories for animals related to a bear (hint: use the LIKE operator)
   ```
   SELECT AName, ACategory FROM Animal
       WHERE AName LIKE '%bear';
   ```

4. Return the listings for all animals whose rarity is not specified in the database
   ```
   SELECT * FROM Animal
       WHERE ACategory is NULL;
   ```

5. Find the rarity rating of all animals that require between 1 and 2.5 hours to be fed
   ```
   SELECT AName, ACategory FROM Animal
       WHERE TimeToFeed >=1 AND TimeToFeed<=2.5;
   ```

6. Find the names of the animals that are related to the tiger and are not common
   ```
   SELECT AName FROM Animal
       WHERE AName Like '%tiger' AND ACategory != 'common';
   ```

7. Find the minimum and maximum feeding time amongst all the animals in the zoo (single query)
   ```
   SELECT MIN(TimeToFeed), MAX(TimeToFeed) FROM Animal;
   ```

8. Find the average feeding time for all the rare animals
   ```
   SELECT AVG(TimeToFeed) FROM Animal
       WHERE ACategory = 'rare';
   ```

9. Find listings for animals with ID less than 10 and also require more than 2 hours to feed.
   ```
   SELECT * FROM Animal
       WHERE AID < 10 AND TimeToFeed > 2;
   ```