Database Admin 101 - Lab

Introduction

In this lab, you'll go through the process of `designing` and `creating` a database. From there, you'll begin to `populate` this table with mock data provided to you.

Objectives

You will be able to:

- Use knowledge of the `structure` of databases to `create` a database and `populate` it

The Scenario

You are looking to design a database for a school that will house various information from student grades to contact information, class roster lists and attendance. First, think of how you would design such a database. What tables would you include? What columns would each table have? What would be the primary means to join said tables?

Creating the Database

Now that you've put a little thought into how you might design your database, it's time to go ahead and create it! Start by import the necessary packages. Then, create a database called **school.sqlite**.

```
# Import necessary packages
import pandas as pd
import sqlite3

# Create the database school.sqlite (a connection)
conn = sqlite3.Connection('school.sqlite')
```

Create a Table for Contact Information

Create a table called contactInfo to house contact information for both students and staff. Be sure to include columns for first name, last name, role (student/staff), telephone number, street, city, state, and zipcode. Be sure to also create a primary key for the table.

```
# brian-added
cur = conn.cursor()
cur.execute("""
CREATE TABLE contactInfo (
        userId INTEGER PRIMARY KEY,
        firstName TEXT,
```

```
        lastName TEXT,
        role TEXT,
        telephone INTEGER,
        street TEXT,
        city TEXT,
        state TEXT,
        zipcode TEXT
        );
""")
```

```
<sqlite3.Cursor at 0x217ae318f80>
```

Populate the Table

Below, code is provided for you in order to load a list of dictionaries. Briefly examine the list. Each dictionary in the list will serve as an entry for your contact info table. Once you've briefly investigated the structure of this data, write a for loop to iterate through the list and create an entry in your table for each person's contact info.

```
# Load the list of dictionaries; just run this cell
import pickle
# opens the file 'contact_list.pickle' in read-binary mode ('rb')
with open('contact_list.pickle', 'rb') as f:
    contacts = pickle.load(f)

# Iterate over the contact list and populate the contactInfo table
here
for contact in contacts:
    firstName = contact['firstName']
    lastName = contact['lastName']
    role = contact['role']
    telephone  = contact['telephone ']
    street = contact['street']
    city = contact['city']
    state = contact['state']
    zipcode  = contact['zipcode ']
    cur.execute("""INSERT INTO contactInfo (firstName, lastName, role,
telephone, street, city, state, zipcode)
                   VALUES ('{}', '{}', '{}', '{}', '{}', '{}', '{}',
'{}');
                """.format(firstName, lastName, role, telephone,
street, city, state, zipcode) )
```

Query the Table to Ensure it is populated

```
# brian-answer
cur.execute("""SELECT * FROM contactInfo;""")
# fetches all the rows of a query result
# and returns them as a list of tuples
```

```
df = pd.DataFrame(cur.fetchall())
# an attribute of the cursor that
# provides information about the columns
df.columns = [x[0] for x in cur.description]
# # the DataFrame created from the query result
df
```

```
   userId    firstName  lastName     role   telephone  \
0       1    Christine    Holden    staff  2035687697
1       2  Christopher    Warren  student  2175150957
2       3        Linda  Jacobson    staff  4049446441
3       4       Andrew     Stepp  student  7866419252
4       5         Jane     Evans  student  3259909290
5       6         Jane     Evans  student  3259909290
6       7         Mary    Raines  student  9075772295
7       8           Ed     Lyman  student  5179695576

                     street       city state zipcode
0         1672 Whitman Court   Stamford    CT   06995
1  1935 University Hill Road  Champaign    IL   61938
2       479 Musgrave Street    Atlanta    GA   30303
3  2981 Lamberts Branch Road    Hialeah    Fl   33012
4       1461 Briarhill Lane    Abilene    TX   79602
5       1461 Briarhill Lane    Abilene    TX   79602
6     3975 Jerry Toth Drive  Ninilchik    AK   99639
7             3478 Be Sreet    Lansing    MI   48933
```

Commit Your Changes to the Database

Persist your changes by committing them to the database.

```
# brian-answer
conn.commit()
```

Create a Table for Student Grades

Create a new table in the database called "grades". In the table, include the following fields: userId, courseId, grade.

** This problem is a bit more tricky and will require a dual key. (A nuance you have yet to see.) Here's how to do that:

```
CREATE TABLE table_name(
   column_1 INTEGER NOT NULL,
   column_2 INTEGER NOT NULL,
   ...
   PRIMARY KEY(column_1,column_2,...)
);
```

```
# Create the grades table
cur.execute("""
    CREATE TABLE grades(
        userId INTEGER NOT NULL,
        courseId INTEGER NOT NULL,
        grade INTEGER,
        PRIMARY KEY(userId, courseId))
""")
```

```
<sqlite3.Cursor at 0x217ae318f80>
```

Remove Duplicate Entries

An analyst just realized that there is a duplicate entry in the contactInfo table! Find and remove it.

```
# Find the duplicate entry
cur.execute("""
    SELECT
            firstName, lastName, telephone, COUNT(*)
    FROM contactInfo
    GROUP BY firstName, lastName, telephone
    HAVING COUNT(*) > 1;
""").fetchall()
```

```
[('Jane', 'Evans', 3259909290, 2)]
```

```
# Delete the duplicate entry
# Delete the duplicate entry
cur.execute("""
    DELETE FROM contactInfo
    WHERE telephone = 3259909290;
""")
```

```
<sqlite3.Cursor at 0x217ae318f80>
```

```
# Check that the duplicate entry was removed
cur.execute("""
    SELECT firstName, lastName, telephone, COUNT(*)
    FROM contactInfo
    GROUP BY firstName, lastName, telephone
    HAVING COUNT(*) > 1;
""").fetchall()
```

```
[]
```

Updating an Address

Ed Lyman just moved to `2910 Simpson Avenue York, PA 17403`. Update his address accordingly.

```
# Update Ed's address
# brian-answer
u = '''
    UPDATE contactInfo
        SET street = "2910 Simpson Avenue",
            city = "York",
            state = "PA",
            zipcode = "17403"
    WHERE firstName = "Ed";
    '''
# executing
cur.execute(u)

# Query the database to ensure the change was made
# search-query
q = '''SELECT * FROM contactInfo where firstName = "Ed";'''
# fetching answer
pd.read_sql(q, conn)
```

```
   userId firstName lastName     role   telephone                street
city  \
0       8        Ed    Lyman  student  5179695576  2910 Simpson Avenue
York

  state zipcode
0    PA   17403
```

```
# Alternatively
cur.execute("""SELECT *
            FROM contactInfo
            WHERE firstName = "Ed"
;""")
df = pd.DataFrame(cur.fetchall())
df.columns = [x[0] for x in cur.description]
df
```

```
   userId firstName lastName     role   telephone                street
city  \
0       8        Ed    Lyman  student  5179695576  2910 Simpson Avenue
York

  state zipcode
0    PA   17403
```

Commit Your Changes to the Database

Once again, persist your changes by committing them to the database.

```
# Your code here
conn.commit()
```

Summary

While there's certainly more to do with setting up and managing this database, you got a taste for creating, populating, and maintaining databases! Feel free to continue fleshing out this exercise for more practice.