index

February 25, 2022

1 Database Admin 101 - Lab

1.1 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.

1.2 Objectives

You will be able to:

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

1.3 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?

1.4 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**.

[12]:
```
# Import necessary packages
import sqlite3
import pandas as pd
```

[13]:
```
# Create the database school.sqlite
conn = sqlite3.connect('school.sqlite')
```

1.5 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.

[14]:
```
# Your code here
cur = conn.cursor()
cur.execute("""CREATE TABLE cont_info (
                                        id INTEGER PRIMARY KEY,
                                        fname TEXT,
                                        lname TEXT,
                                        role TEXT,
                                        tel INTEGER,
                                        street TEXT,
                                        city TEXT,
                                        state TEXT,
                                        zpcode TEXT
                                        );
            """)
```

```
---------------------------------------------------------------------------
OperationalError                          Traceback (most recent call last)
/tmp/ipykernel_218/320789521.py in <module>
      1 # Your code here
      2 cur = conn.cursor()
----> 3 cur.execute("""CREATE TABLE cont_info (
      4                                         id INTEGER PRIMARY KEY,
      5                                         fname TEXT,

OperationalError: table cont_info already exists
```

1.6 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

with open('contact_list.pickle', 'rb') as f:
    contacts = pickle.load(f)
```

[ ]:
```
# Iterate over the contact list and populate the contactInfo table here
for c in contacts:
    fname = c["firstName"]
    lname = c["lastName"]
    role = c["role"]
    tel = c["telephone "]
    street = c["street"]
```

```
    city = c["city"]
    state = c["state"]
    zpcode = c["zipcode "]
    cur.execute("""INSERT INTO cont_info (fname, lname, role, tel, street,␣
 ↪city, state, zpcode)
    VALUES ("{}", "{}", "{}", "{}", "{}", "{}", "{}", "{}")""".format(fname,␣
 ↪lname, role, tel, street, city, state, zpcode))
```

Query the Table to Ensure it is populated

[ ]:
```
# Your code here
cur.execute("""SELECT * FROM cont_info;""")
df = pd.DataFrame(cur.fetchall())
df
```

1.7 Commit Your Changes to the Database

Persist your changes by committing them to the database.

[ ]:
```
# Your code here
conn.commit()
```

1.8 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
q ="""
CREATE TABLE grades (
             userId INTEGER NOT NULL,
             courseId INTEGER NOT NULL,
             grade INTEGER,
             PRIMARY (userId, courseId)
             );

"""
cur.execute(q)
```

1.9 Remove Duplicate Entries

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

[15]:
```
# Find the duplicate entry
cur.execute("""SELECT id, fname, lname, tel, COUNT(*) FROM cont_info
GROUP BY fname HAVING COUNT(*) != 1 and COUNT(*) != 0""").fetchall()
```

[15]:
```
[(4, 'Andrew', 'Stepp', 7866419252, 2),
 (1, 'Christine', 'Holden', 2035687697, 2),
 (2, 'Christopher', 'Warren', 2175150957, 2),
 (8, 'Ed', 'Lyman', 5179695576, 2),
 (6, 'Jane', 'Evans', 3259909290, 3),
 (3, 'Linda', 'Jacobson', 4049446441, 2),
 (7, 'Mary', 'Raines', 9075772295, 2)]
```

[ ]:
```
# cur.execute("""
# SELECT fname, lname, tel, COUNT(*)
# FROM cont_info
# GROUP BY fname
# HAVING COUNT(*) > 1;
# """).fetchall()
```

[10]:
```
# Delete the duplicate entry
cur.execute("""DELETE FROM cont_info WHERE id = 5""")
```

[10]:
```
<sqlite3.Cursor at 0x7fb3ec9a7650>
```

[11]:
```
# Check that the duplicate entry was removed
cur.execute("""SELECT id, fname, lname, tel, COUNT(*) FROM cont_info
GROUP BY fname HAVING COUNT(*) != 1 and COUNT(*) != 0""").fetchall()
```

[11]:
```
[(4, 'Andrew', 'Stepp', 7866419252, 2),
 (1, 'Christine', 'Holden', 2035687697, 2),
 (2, 'Christopher', 'Warren', 2175150957, 2),
 (8, 'Ed', 'Lyman', 5179695576, 2),
 (6, 'Jane', 'Evans', 3259909290, 3),
 (3, 'Linda', 'Jacobson', 4049446441, 2),
 (7, 'Mary', 'Raines', 9075772295, 2)]
```

[34]:
```
cur.execute("""SELECT * FROM cont_info WHERE id = 8""").fetchall()
```

[34]:
```
[(8,
  'Ed',
  'Lyman',
  'student',
  5179695576,
```

```
 '3478 Be Sreet',
 'Lansing',
 'MI',
 '48933')]
```

1.10 Updating an Address

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

[45]:
```
# Update Ed's address
cur.execute("""
UPDATE cont_info
SET street = "2910 Simpson Avenue",
    city = "York",
    state = "PA",
    zpcode = 17403
WHERE fname = "Ed" AND lname = "Lyman";
""")
```

[45]:
```
<sqlite3.Cursor at 0x7fb255c6b810>
```

[50]:
```
# Query the database to ensure the change was made
cur.execute("""SELECT id, fname, street, city, state, zpcode FROM cont_info
GROUP BY fname ORDER BY id ASC""").fetchall()
```

[50]:
```
[(1, 'Christine', '1672 Whitman Court', 'Stamford', 'CT', '06995'),
 (2, 'Christopher', '1935 University Hill Road', 'Champaign', 'IL', '61938'),
 (3, 'Linda', '479 Musgrave Street', 'Atlanta', 'GA', '30303'),
 (4, 'Andrew', '2981 Lamberts Branch Road', 'Hialeah', 'Fl', '33012'),
 (6, 'Jane', '1461 Briarhill Lane', 'Abilene', 'TX', '79602'),
 (7, 'Mary', '3975 Jerry Toth Drive', 'Ninilchik', 'AK', '99639'),
 (8, 'Ed', '2910 Simpson Avenue', 'York', 'PA', '17403')]
```

1.11 Commit Your Changes to the Database

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

[51]:
```
# Your code here
conn.commit()
```

1.12 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.

[ ]:
```

```