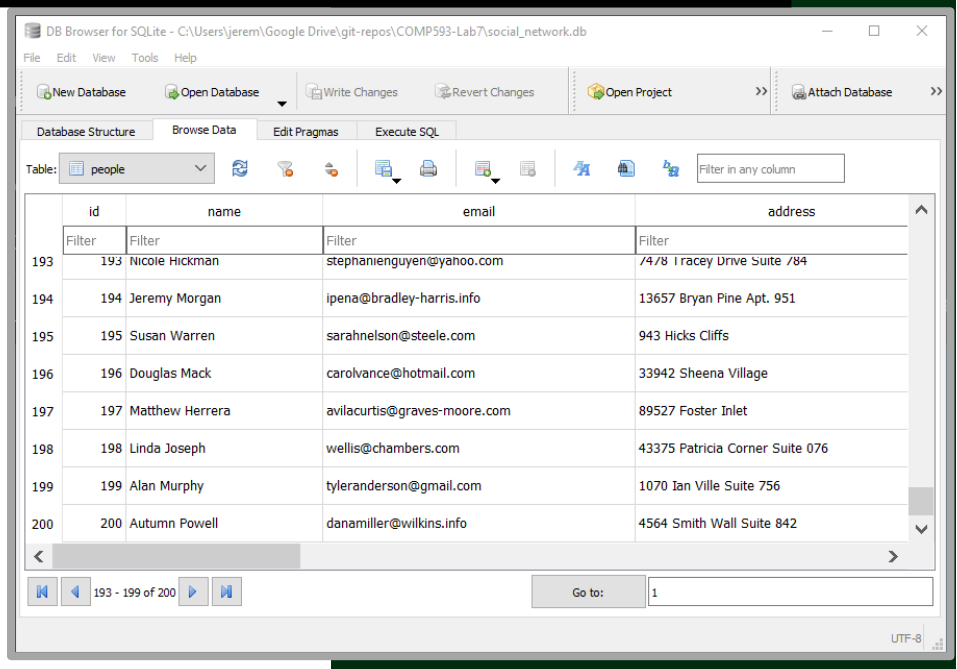
FLEMING

Lab 7: Social Network People



COMP 593 - Scripting Applications

TABLE OF CONTENTS

PURPOSE

The purpose of this lab assignment is to help set the stage for some more advanced data manipulation by creating and populating an SQLite database. We will be experimenting with the basics, such as how to **CREATE** a table and how to **INSERT** data into that table, and we will complete the task by reading data from that table using **SELECT** statements. This lab also introduces you to a library module called **Faker** which is useful for rapidly populating dummy data.

INTRODUCTION

During this lab, we will experiment with using a few different libraries and modules to connect to and interact with an SQLite database. After creating and testing our database, we will populate it with fake data and run some queries to see what kind of information we can retrieve.

The database will initially be designed to hold information about the users of a very simple Social Network and will be expanded in the next lab to also hold information about relationships between users.

WORKING WITH SQLITE DATABASES

We will be using the **sqlite3** library to work with our database, which is included in the Standard Python Library, so it should already be installed on your computer.

CREATING AN SQLITE DATABASE

After running the following code, an SQLite database file named **social_network.db** should be created in the current working directory (CWD). The **sqlite3.connect()** function, can be used to connect to an existing SQLite database file, or it will automatically create a new, empty database file if the specified file does not exist.

```
import sqlite3

# Opens a connection to an SQLite database.
# Returns a Connection object that represent the database connection.
# A new database file will be created if it doesn't already exist.
con = sqlite3.connect('social_network.db')
```

CREATING A TABLE

The following code will create a table named **people** within the database.

```
import sqlite3

# Open a connection to the database.
con = sqlite3.connect('social_network.db')

# Get a Cursor object that can be used to run SQL queries on the database.
cur = con.cursor()

# Define an SQL query that creates a table named 'people'.
# Each row in this table will hold information about a specific person.
create_ppl_tbl_query = """
    CREATE TABLE IF NOT EXISTS people
    (
        id         INTEGER PRIMARY KEY,
        name       TEXT NOT NULL,
        email      TEXT NOT NULL,
        address    TEXT NOT NULL,
        city       TEXT NOT NULL,
        province   TEXT NOT NULL,
        bio        TEXT,
        age        INTEGER,
        created_at DATETIME NOT NULL,
        updated_at DATETIME NOT NULL
    );
"""

# Execute the SQL query to create the 'people' table.
# Database operations like this are called transactions.
cur.execute(create_ppl_tbl_query)

# Commit (save) pending transactions to the database.
# Transactions must be committed to be persistent.
con.commit()

# Close the database connection.
# Pending transactions are not implicitly committed, so any
# pending transactions that have not been committed will be lost.
con.close()
```

After creating a table in the database, you can confirm that the table was created successfully by viewing the database using a database browser application, such as SQLite extension for VS Code or DB Browser for SQLite.

INSERTING DATA INTO A TABLE

The following code will insert one row into the **people** table.

```
import sqlite3
from datetime import datetime

con = sqlite3.connect('social_network.db')
cur = con.cursor()

# Define an SQL query that inserts a row of data in the people table.
# The ?'s are placeholders to be fill in when the query is executed.
# Specific values can be passed as a tuple into the execute() method.
add_person_query = """
    INSERT INTO people
    (
        name,
        email,
        address,
        city,
        province,
        bio,
        age,
        created_at,
        updated_at
    )
    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?);
"""

# Define a tuple of data for the new person to insert into people table
# Data values must be in the same order as specified in query
new_person = ('Bob Loblaw',
              'bob.loblaw@whatever.net',
              '123 Fake St.',
              'Fakesville',
              'Fake Edward Island',
              'Enjoys making funny sounds when talking.',
              46,
              datetime.now(),
              datetime.now())

# Execute query to add new person to people table
cur.execute(add_person_query, new_person)

con.commit()
con.close()
```

GETTING PEOPLE DATA FROM THE DATABASE

The following code queries the database for all rows in the **people** table, and then prints them out. It can be used to confirm that the new person was added to the **people** table as expected.

```
import sqlite3
from pprint import pprint

con = sqlite3.connect('social_network.db')
cur = con.cursor()

# Query the database for all information for all people.
cur.execute('SELECT * FROM people')

# Fetch all query results.
# The fetchall() method returns a list, where each list item
# is a tuple containing data from one row in the people table.
all_people = cur.fetchall()

# Pretty print (pprint) outputs data in an easier to read format.
pprint(all_people)

con.commit()
con.close()
```

WORKING WITH FAKER

Faker is a Python package that generates fake data. We are going to use it to populate the **people** table with fake people. Faker is not included in the Python Standard Library, so you must first install it using **pip install faker**.

The following code shows how Faker works by generating fake data for ten Canadian provinces.

```
from faker import Faker

# Create a faker object for English Canadian locale
fake = Faker("en_CA")

# Generate fake data for 10 provinces
for _ in range(10):
    province = fake.administrative_unit()
    population = fake.random_int(min=900000, max=100000000)
    print(f'The population of {province} is {population}.')
```

The Faker class has many "providers" that can be used to generate all sort of different fake data. You will need to read through the list of providers in the Faker documentation to find an appropriate provider to generate fake data for each column in the **people** table.

STUDENT INSTRUCTIONS

For this lab assignment, students will implement two scripts as described in the following subsections.

SCRIPT 1: CREATE DATABASE AND POPULATE PEOPLE TABLE

Implement a Python script that creates an SQLite database containing the **people** table shown in the above examples and inserts 200 fake people into the **people** table. The database must reside in the same directory as the script.

Use Faker to generate all data for the fake people, with the following exceptions and constraints:

- The **age** column must contain a random integer between 1 and 100. Use Faker or the random module in the Python Standard Library to generate the random integers.
- Use the **datetime** module to get the current date and time for the **created_at** and **updated_at** columns.

SCRIPT 2: QUERY THE DATABASE FOR OLD PEOPLE

Implement a Python script that:

- Uses an SQL queries to get the **name** and **age** of all people in the database who are at least 50 years old.
- Prints the **name** and **age** of each person in the query result within a sentence.
- Saves the names and ages of all old people to a CSV file (that includes a header row) in the same folder in which the script resides.

Hints:

- See example SQL queries in the lecture slides.
- The SQL query should include the keywords **SELECT**, **FROM**, **WHERE**, and **LIMIT**.
- Use the **fetchall()** method of the Cursor class to get a list of all query results.
- Use a **for** loop to iterate through the query results to print each sentence.
- Use pandas to convert the query results to a DataFrame and save it to a CSV file.

Example script output:

```
Windows PowerShell
PS C:\> python old_people.py
Jeffrey Farley is 60 years old.
Todd Johnson is 50 years old.
Joe West is 89 years old.
Sharon Bennett is 78 years old.
Aimee Myers is 95 years old.
Barry Young is 61 years old.
Robin Johnson is 96 years old.
Ryan Hobbs is 91 years old.
John Ellison is 100 years old.
Rodney Peterson is 78 years old.
Lisa Patterson is 94 years old.
Veronica Williams is 61 years old.
Denise Mcintyre is 81 years old.
Jason Evans is 88 years old.
Tanya Stone is 96 years old.
Charles Jones is 65 years old.
Cheryl Ho is 87 years old.
Edward Roberts is 70 years old.
Jessica Johnson PhD is 51 years old.
Cheryl Steele is 63 years old.
```

DROPBOX SUBMISSION

Submit a link to the GitHub repository containing your script, e.g., https://github.com/BobLoblaw/COMP593-Lab7.

ASSESSMENT

Criteria	Out Of	Assessment Criteria
GitHub	1	• GitHub repository that contains both script files • .db and .csv files excluded from repo using .gitignore
Script 1	4	• Correctly creates database and people table • Correctly populates people table with 200 fake people
Script 2	5	• Correctly queries database for old people • Correctly prints name and age of old people in sentences • Correctly saves old people names and ages to a CSV file
Total:	**10**	