CMPT 354 Fall 2024
Database Systems
Martin Ester
TAs: Jackson de Faria, Fitzpatrick Laddaran
Assignment 7

Total marks: 100 (10% of the course)
Due date: November 29, 2024

In this assignment, you will write a simple application program for the Yelp database, created in Assignment 3, and implement several functions for querying and modifying the database.

Preparation

1. You can use the Python programming language to implement the application. Helpdesk has provided the following instructions and examples for using the CSIL databases from various programming languages including Python: http://www.sfu.ca/computing/about/support/csil/windows/how-to-use-sql.html#sample-codes.
2. You can find your password for the database connection in your database's dbo.helpdesk table in SQL Server -- right click on it and select "Select top 1000 rows".
3. We assume that you are familiar with Python. Here, we suggest a tutorial about creating applications based on databases in Python with SQL Server Database:

   https://www.c-sharpcorner.com/UploadFile/75a48f/micro/

Application Requirements

Your application should either have a graphical user interface or a command line interface with a hierarchical menu to support the following functions. Your submitted application should be run directly on the workstations in CSIL. Since we are using Python, you can simply submit a .py file that we can directly run. Note that once the program is running, it should allow the tester to test all its functions and not terminate unless the tester manually closes the window or console.

The functions to be implemented are as follows:

Login

1. This function allows the user to log in the interface to have access to all other functionalities. The user must be remembered by the system for further operations in the same session.
2. The user must enter a valid user ID.
3. If the user ID is invalid, an appropriate message should be shown to the user.

Search Business

1. This function allows the user to search for businesses that satisfy certain criteria.
2. A user should be able to set the following filters as their search criteria: minimum number of stars, city, and name (or part of the name). The search is not case-sensitive. It means that the search for upper or lower case must return the same results.
3. The user can choose one of the following three orderings of the results: by name, by city, or by number of stars.
4. After the search is complete, a list of search results must be shown to the user. The list must include the following information for each business: id, name, address, city and number of stars. The results must be ordered according to the chosen attribute. The results can be shown on the terminal or in a GUI.
5. If the search result is empty, an appropriate message should be shown to the user.

Search Users

6. This function allows the user to search for users that satisfy certain criteria.
7. A user should be able to set the following filters as their search criteria: name (or a part of the name), minimum review count, minimum average stars. The search is not case-sensitive.
8. After the search is complete, a list of search results must be shown to the user. The list must include the following information for each user: id, name, review count, useful, funny, cool, average stars, and the date when the user was registered at Yelp. The results must be ordered by name. The results can be shown on the terminal or in a GUI.
9. If the search result is empty, an appropriate message should be shown to the user.

Make Friend

1. A user must be able to select another user from the results of the function Search Users and create a friendship. This can be done by entering the user's ID in a terminal or by clicking on a user in a GUI. The selected user will be a friend of the user that is logged in to the database.
2. The friendship should be recorded in the Friendship table.

Review Business

1. A user should be able to review a business.
2. To make a review, a user must enter the business's ID in a terminal or click on a business returned by Search Business in a GUI.
3. The user must provide the number of stars (integer between 1 and 5).
4. The review should be recorded in the Review table. Create a review ID consisting of the ID of the logged user and the current date.

5. The program should update the number of stars and the count of reviews for the reviewed business. You need to make sure that the triggers you implemented in assignment 4 are working properly with your application program.

Submission

Test your application to make sure it can successfully connect to the database and run. Import/restore the data to the original status where necessary. **At submission time your database must contain an identical copy of the provided data without modifications** (insertions, deletions, updates) for the purpose of running test cases by the TA. Also, make sure that the triggers from assignment 4 are created in your database.

Submit a zip-file Assignment7.zip to CourSys.

The zip file should include the source code, with an executable .py application file for your project.

Guide on how-to SQL in SFU

For Assignment 7, you are going to create an application that connects with the database we created in Assignment 4. Please take the following steps to create a starter Python code that runs in the CSIL machines and connect to the database.

Step 1: obtain your database passphrase

Connecting to the database requires a passphrase specifically for this connection. **It is not your standard SFU password.** It's a password randomly created by the CSIL teams when they were setting up the database for our CMPT354 class.

To get your passphrase, first log-in one of the CSIL computers, in e.g. room ASB 9840. Then open **SQL Server Management Studio** in Windows, and connect to the SQL Server. If you are not able to complete this step, please reach to one of the TAs as soon as possible. If you get problems with the certificate, for example, please check this post on Coursys.

After successfully connecting to the SQL server, open the database list on the `Object Explorer` on the left and look for your database. It is called, for example `abc8354`, where `abc8` is your SFU user and the `354` in the end means the number of the class (CMPT-354). This part is identical to what we did in Assignment 4 to view the databases.

Your passphrase is stored in the relation `dbo.helpdesk`, so by viewing the content of this database (e.g. right-click and `Select top 1000 rows`) you can get your passphrase.

Step 2: Create a Python code that connects to the database

Open the File Explorer, navigate the to `U:` disk, and create a python script inside this folder (or, you can also create a folder in this directory, and create the python script inside it). You can edit this file using e.g. VS Code in CSIL.

We have two libraries available to connect to the database already installed in the CSIL python environment, `pymssql` and `pyodbc`:

Option 1: connect using `pymssql`

The base code for using `pymssql` in CSIL is like the following:

```
import pymssql
```

```
conn = pymssql.connect(host='cypress.csil.sfu.ca', user='s_username',
password='provided passphrase', database='<username><coursenumber>')

## Creates a cursor, that is a class from pymssql that is used to make
## queries against the database and obtaining results
cursor = conn.cursor(as_dict=True)

## Execute a query.
cursor.execute('SELECT TOP(10) * FROM dbo.User_yelp')

## Iterate through the returned rows
for row in cursor:
    print('row:', row)
```

In the sample code above, you should change the `pymssql.connect()` function call parameters with your username, passphrase obtained in step 1, and your database name. For your username, you should keep the `s_` prefix, e.g. if your user is `abc8`, then the parameter is `'s_abc8'`, and the database parameter `abc8354`.

Note that in `cursor.execute()` you can also have parameters to use e.g. with `WHERE` clauses:

```
## Execute a query.
cursor.execute(
    'SELECT * \
    FROM dbo.Business \
    WHERE stars >= %s \
        AND review_count >= %s',
    params = (4.5, 100)
)

row = cursor.fetchone()
if row is not None:
    print('Successfully found one or more business(es):')
    while row is not None:
        print(row)
        row = cursor.fetchone()
else:
    print('Business not found!')
```

In real world applications, using `params` is safer than constructing a string with all the parameters concatenated, because of SQL injection. Finally, in the end of the application you can close the connection:

```
conn.close()
```

Option 2: pyodbc

To connect with `pyodbc` you can use:

```
import pyodbc

conn = pyodbc.connect('driver={ODBC Driver 18 for SQL
Server};server=cypress.csil.sfu.ca;uid=s_username;pwd=providedpassphrase;Encry
pt=yes;TrustServerCertificate=yes')
```

The other operations are similar to `pymssql`.

You can also check the documentations of pymssql and pyodbc for more information and more examples.

For the interface development, if you choose to create a GUI, you should use the `tkinter` library, that comes with Python by default on Windows and is already installed in CSIL.