Table of Contents

Introduction

The purpose of this assignment is to perform database operations using Python and SQLite.

The Movies Database

You have accumulated many movies over the years and have decided to create a database to track them. You have created 3 files: movieUI.py, db.py and objects.py. The objects.py file defines your class for the Movies and the db.py file connects to your SQLite database.

Your task for this assignment is to create some SQL queries and make some modifications to the moviesUI.py and db.py programs.

Student Files

For this program, you will need to download and unzip the Student Files - Wk3.zip file.
You will need to rename the Student Files folder to **ab_Assignment_3B,** where a is the first letter of your first name and b is the first letter of your last name (all in lower case). You will be modifying the files in the ab_Assignment_3B folder.

Database Creation

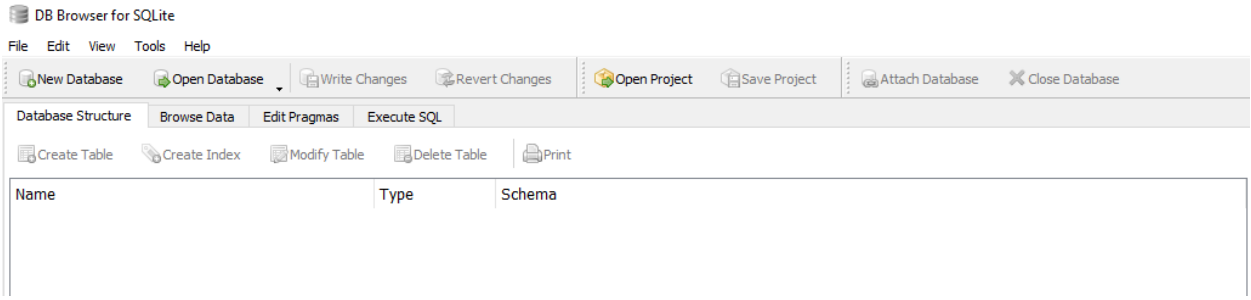
Python has the SQLite library which users can use. Users can create and execute SQL statements with Python. This can be cumbersome when creating databases as there is GUI interface.

There is a database browser available for SQLite which makes the process simpler. You will need to visit: **https://sqlitebrowser.org/**

After visiting the site, select **Download** on the top menu. Next, select the download of your choice. I selected: **DB Browser for SQLite - Standard installer for 64-bit Windows**

After downloading the software, you will need to click on the software to install.
I clicked on: **DB.Browser.for.SQLite-3.12.1-win64-v2.msi**
Use the defaults when installing (I added the desktop icon).

I clicked on the DB Browser and an empty listing of databases appeared.



Now, you can create your database by typing in your SQL or running from a file. To execute the SQL statements, you need to do the following:

On the top menu, select: File > Database from SQL file …

A dialog appears: "**Choose a file to Import**".
Navigate to your db folder which contains the file: **create_movie_db.sql**

Click on the file and click Open.
Next a dialog box appears: "**Choose a filename to save under**".
In the File name: box near the bottom of the screen enter: **movies**

A dialog box should next appear "Import completed" if you were successful. Click **OK.**
You will now visually see your tables and the tables content.

You can browse the data or click **Close Database** near the top of the screen and next close the program.

Now that the database has been created, you can run your moviesUI.py program.

SQL Creation

You will need to use the DB Browser tool to create 6 SQL queries. These queries should be stored in the db folder of your ab_Assignment_3B folder:

1. Click on Open Database to open your movies.db database.

2. Use the Execute SQL tab to enter a query that that selects all columns from the Movie table where the category ID is 2 and click Run SQL button (>) to execute the statement. This should display the results. Click on the Save SQL file button  and save the file as **3A_SQL_1.sql**

3. Modify the value for the category ID in the query so it only selects movies that have a category ID of 3. Then, run this query and view the results.
   Using the drop-down menu hold the mouse button down by the arrow on the Save button, click on the Save SQL file as: **3A_SQL_2.sql**

4. Modify the query so it only returns the name and the year columns. Then, run this query and view the results.
   Using the drop-down menu hold the mouse button down by the arrow on the Save button, click on the Save SQL file as: **3A_SQL_3.sql**

5. Modify the query so it sorts the result set by year in descending order.
   Using the drop-down menu hold the mouse button down by the arrow on the Save button, click on the Save SQL file as: **3A_SQL_4.sql**

6. Enter an INSERT statement that inserts a new row into the Movie table. Use the following values: name = 'Juno", year = '2007', minutes = 96 and categoryID = 2. Then, run this SQL statement. This should not display a result, but is should add a new row to the Movie table.
   Using the drop-down menu hold the mouse button down by the arrow on the Save button, click on the Save SQL file as: **3A_SQL_5.sql**

7. Use the Browse data tab to browse the Movie table and view the new row.

8. Use the Execute SQL tab to run a DELETE statement that deletes the new row.
   Using the drop-down menu hold the mouse button down by the arrow on the Save button, click on the Save SQL file as: **3A_SQL_6.sql**

9. Continue to experiment until you are sure that you understand the SQL statements that are used by the Movie List program**.**

Program Enhancements

Improve the delete command -

1. In the **db.py** program:
   - Add a get_movie() function that gets a Movie object for the specified movie id.

   Details

   - Copy the get_movies_by_category(category_id) function and rename the function: get_movies(movie_id)
   - Within the get_movies function,

     Replace:

```
    results = c.fetchall()

  movies = []
  for row in results:
    movies.append(make_movie(row))
  return movies
```

     With:

```
    row = c.fetchone()
  movie = make_movie(row)

  return movie
```

2. In the **moviesUI.py** program:
   - Modify the delete_movie() function so it gets a Movie object for the specified ID and asks whether you are sure you want to delete the movie as shown below:

   This code should only delete the movie the user enters "y" to confirm the operation.

   Details

   - Within the delete_movie() function**, REMOVE** all the function code after this statement: movie_id = int(input("Movie ID: "))

     Now **ADD** the following code:

```
 Movie = db.get_movie(movie_id)

 prompt = f"Are you sure you want to delete '{Movie.name}' (y/n):"
 response = input(prompt)

 if response.lower() =="y":
    db.delete_movie(movie_id)
    print(f"'{Movie.name} was deleted from the database.\n")
else:
    print(f"'{Movie.name} was NOT deleted from the database.\n")
```

Add the min command -

3. In the **db.py** program:
   - Add a get_movies_by minutes() function that gets a list of Movie objects that have a running time that's less than the number of minutes passed to it as a parameter.

Details

- In the db.py program, copy get_movies_by_category(category_id) function and rename it: get_movies_by_minutes(max_minutes)

  Change this line: WHERE Movie.categoryID = ?'''
  To:
  WHERE minutes < ?
  ORDER BY minutes ASC'''

  Change this line: c.execute(query, (category_id,))
  To: c.execute(query, (max_minutes,))

4. In the **movieUI.py** program:
   - Add a display_movies_by_minutes() function that prompts the user to enter the maximum number of minutes and displays all selected movies. This should sort the movies by minutes in ascending order.

Details

- Add the following code:

```
def display_movies_by_minutes():
    max_min = int(input("Maximum number of minutes: "))
    print()
    movies = db.get_movies_by_minutes(max_min)
    display_movies(movies, f"LESS THAN {max_min}")
```

5. In the **movieUI.py** program, modify the **main()** function so that it provides for the min command. The min command should follow the year command and have a prompt of:

   "View movies with a maximum value of minutes".

   Details

   - Renumber your global variables as follows:
     CAT = 1
     YEAR = 2
     MIN = 3
     ADD = 4
     DEL = 5
     EXIT = 6

   - **Insert** the Menu prompt:

     COM_MENU += f"{MIN} - View movies with a maximum value of minutes\n"

     After the View movies by year Menu prompt.

   - **Add** an elif to handle the display movies by minutes after the display_movies_by_year:

     ```
     elif command == MIN:
         display_movies_by_minutes()
     ```

Program Output

Below is some sample output from the Komodo IDE.

```
Please select your COMMAND choice: (1-6): 5
Movie ID: 14
Are you sure you want to delete 'Juno' (y/n):y
'Juno was deleted from the database.

COMMAND MENU
1 - View movies by category
2 - View movies by year
3 - View movies with a maximum value of minutes
4 - Add a movie
5 - Delete a movie
6 - Exit the program

Please select your COMMAND choice: (1-6): 3
Maximum number of minutes: 100

MOVIES - LESS THAN 100
  ID Name                                  Year   Mins  Category
-----------------------------------------------------------------
   4 Ice Age                               2002   81    Animation
   5 Toy Story                             1995   81    Animation
   1 Spirit: Stallion of the Cimarron      2002   83    Animation
   3 Aladdin                               1992   90    Animation
   6 Monty Python and the Holy Grail       1975   91    Comedy
   7 Monty Python's Life of Brian          1979   94    Comedy

COMMAND MENU
1 - View movies by category
2 - View movies by year
3 - View movies with a maximum value of minutes
4 - Add a movie
5 - Delete a movie
6 - Exit the program

Please select your COMMAND choice: (1-6):
```

Program Notes

Students will be modifying the files in the following folder:

ab_Assignment_3B

Where a is the students first letter of their first name and b is the first letter of their last name.

The Python file is required to have a programmer header containing the following:

```
#-----------------------------------------------------------------------------
# Assignment:    Assignment 3B
#
# Program Name:  xy_Assignment_3B.py
#                xy is the first letter of your first and last name
#
# Purpose:       The purpose of this program is
#                Fill in the purpose of this program here.
#
# Author:        Your Name
# Course:        231CIS109.950
#
# Created:       Today's Date in the Month Day, Year (4 digit) format
#
#----------------------------------------------------------------------------
```

You will need to zip your ab_Assignment_3B folder and submit the zip file.

Submit Instructions

- Before submitting, Refer to the Python Programming Coding Standards document on Canvas.
- Correct any spelling or grammar errors.
- Submit file via Canvas's submission tool.