

Arcilla, Andrew Sean D.

BSCS C204

FINALS LAB TASK 5: CLI using MySQL and Python

Finals Lab Task 5. CLI using Mysql and Python

1. Make sure you have installed the following pre-requisites before proceeding:
 - a. Mysql-connector
 - b. Mysql-connector-python
 - c. Xampp is running along with Apache and Mysql in the background
2. Create the following database in Mysql;
 - a. Database name: **moviesDB** with the ff: fields:

movie_id	int(10) Primary Key
title	varchar(50) NOT NULL
main_actor	varchar(50) NOT NULL
director	varchar(50) NOT NULL
genre	varchar(25) NOT NULL
gross_sales	float
ratings	(G, PG, R13, R16,X) varchar(5)
 - b. Insert at least 5 records
 - c. Create a user named **test_user** and assign a **password** and give it an admin access by checking necessary
3. Guided by the Demo code attached in this task. test_DemoDB.py
4. Kindly continue working on the code that will allow the user to navigate through the Database and perform simple CRUD operations. Follow the following **CLI Menu Options**:

```
----- MOVIE DATABASE CLI -----
1. Add Employee
2. View Employees
3. Update Employee
4. Delete Employee
5. Search Employee
6. Display Total Records
7. Exit
Select an option (1-6): |
```

5. The user should be able perform the ff: in your program.

MOVIE DATABASE CRUD APP

- 1- Add New Record
- 2- View all records,
- 3- Update a Record and show the updates,
- 4- Delete a record
- 5- Search A Record**
- 6- Display **Total Numbers** of Movies stored in the database
- 7- Exit
6. For additional challenge, Task – You are to add a **SEARCH option** in the MENU that will allow the user to search by Name or emp_id, then display the information about the record being search. You may use Like statement and fetchOne method in my SQL to do this,
7. You are also going to add a method that will display the the **total number of records** in your database – You may use SQL count statement for this.
8. What to submit:
 - a. UI Menu
 - b. Sample Output
 - c. Source Code
 - d. Exported sql file

Source Code:

```
import mysql.connector

conn = mysql.connector.connect(
    host="localhost",
    user="test_user",
    password="passwordis101010",
    database="moviesDB"
)

cursor = conn.cursor()

def add_movie():
    print("\n=== ADD NEW MOVIE ===")
    movie_id = input("Movie ID: ")
    title = input("Title: ")
    actor = input("Main Actor: ")
    director = input("Director: ")
    genre = input("Genre: ")
    sales = float(input("Gross Sales: "))
    rating = input("Rating (G, PG, R13, R16, X): ")

    sql = "INSERT INTO movies (movie_id, title, main_actor, director, genre, gross_sales, ratings) VALUES (%s,%s,%s,%s,%s,%s,%s)"
    cursor.execute(sql, (movie_id, title, actor, director, genre, sales, rating))
    conn.commit()
    print("\nMovie added successfully!\n")

def view_movies():
    print("\n=== MOVIE LIST ===")
    cursor.execute("SELECT * FROM movies")
    rows = cursor.fetchall()
    for row in rows:
        print(row)
    print("\nTotal Records:", len(rows))

def update_movie():
    print("\n=== UPDATE MOVIE ===")
    movie_id = input("Enter Movie ID to update: ")

    title = input("New Title: ")
    actor = input("New Main Actor: ")
    director = input("New Director: ")
    genre = input("New Genre: ")
    sales = float(input("New Gross Sales: "))
    rating = input("New Rating: ")

    sql = """UPDATE movies SET
        title=%s, main_actor=%s, director=%s, genre=%s,
        gross_sales=%s, ratings=%s WHERE movie_id=%s"""
    cursor.execute(sql, (title, actor, director, genre, sales, rating, movie_id))
    conn.commit()

    print("\nMovie updated successfully!\n")
```

```

def delete_movie():
    print("\n=== DELETE MOVIE ===")
    movie_id = input("Enter Movie ID to delete: ")

    cursor.execute("DELETE FROM movies WHERE movie_id=%s", (movie_id,))
    conn.commit()

    print("\nMovie deleted successfully!\n")

def search_movie():
    print("\n=== SEARCH MOVIE ===")
    keyword = input("Enter movie title or actor to search: ")

    sql = "SELECT * FROM movies WHERE title LIKE %s OR main_actor LIKE %s"
    search_key = "%" + keyword + "%"

    cursor.execute(sql, (search_key, search_key))
    result = cursor.fetchone()

    if result:
        print("\nRecord Found:")
        print(result)
    else:
        print("\nNo record found.")

def count_movies():
    cursor.execute("SELECT COUNT(*) FROM movies")
    count = cursor.fetchone()[0]
    print(f"\nTotal Movies in Database: {count}\n")

def menu():
    while True:
        print("""
=====
        MOVIE DATABASE CRUD APP
=====
1 - Add New Record
2 - View All Records
3 - Update a Record
4 - Delete a Record
5 - Search a Record
6 - Display Total Number of Movies
7 - Exit
=====
""")

        choice = input("Enter choice: ")

        if choice == "1":
            add_movie()
        elif choice == "2":
            view_movies()
        elif choice == "3":
            update_movie()
        elif choice == "4":
            delete_movie()
        elif choice == "5":

```

```

        search_movie()
    elif choice == "6":
        count_movies()
    elif choice == "7":
        print("Exiting program...")
        break
    else:
        print("Invalid input! Try again.\n")

menu()
conn.close()

```

Sample Output:

`SELECT * FROM `movies``

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

☐ Show all | Number of rows: | Filter rows: | Sort by key:

Extra options

	movie_id	title	main_actor	director	genre	gross_sales	ratings
<input type="checkbox"/> Edit Copy Delete	1	Justice League	Ben Affleck	Zack Snyder	Action	657.9	PG
<input type="checkbox"/> Edit Copy Delete	2	Zack Snyder's Justice League	Henry Cavill	Zack Snyder	Action	0	R13
<input type="checkbox"/> Edit Copy Delete	3	Batman v Superman	Ben Affleck	Zack Snyder	Action	873.6	PG
<input type="checkbox"/> Edit Copy Delete	4	Man of Steel	Henry Cavill	Zack Snyder	Action	668	PG
<input type="checkbox"/> Edit Copy Delete	5	Wonder Woman	Gal Gadot	Patty Jenkins	Action	821.8	PG

☐ Check all | With selected: [Edit](#) [Copy](#) [Delete](#) [Export](#)

☐ Show all | Number of rows: | Filter rows: | Sort by key:

Query results operations

```

=====
      MOVIE DATABASE CRUD APP
=====
1 - Add New Record
2 - View All Records
3 - Update a Record
4 - Delete a Record
5 - Search a Record
6 - Display Total Number of Movies
7 - Exit
=====

Enter choice: 2

=== MOVIE LIST ===
(1, 'Justice League', 'Ben Affleck', 'Zack Snyder', 'Action', 657.9, 'PG')
(2, "Zack Snyder's Justice League", 'Henry Cavill', 'Zack Snyder', 'Action', 0.0, 'R13')
(3, 'Batman v Superman', 'Ben Affleck', 'Zack Snyder', 'Action', 873.6, 'PG')
(4, 'Man of Steel', 'Henry Cavill', 'Zack Snyder', 'Action', 668.0, 'PG')
(5, 'Wonder Woman', 'Gal Gadot', 'Patty Jenkins', 'Action', 821.8, 'PG')

Total Records: 5

```

```

=====
      MOVIE DATABASE CRUD APP
=====
1 - Add New Record
2 - View All Records
3 - Update a Record
4 - Delete a Record
5 - Search a Record
6 - Display Total Number of Movies
7 - Exit
=====

Enter choice: 1

=== ADD NEW MOVIE ===
Movie ID: 090
Title: Spider-Man 2
Main Actor: Tobey Maguire
Director: Sam Raimi
Genre: Action
Gross Sales: 719.9
Rating (G, PG, R13, R16, X): PG

Movie added successfully!

```

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

☐ Show all | Number of rows: Filter rows: Sort by key:

Extra options

				movie_id	title	main_actor	director	genre	gross_sales	ratings			
<input type="checkbox"/>		Edit		Copy		Delete	1	Justice League	Ben Affleck	Zack Snyder	Action	657.9	PG
<input type="checkbox"/>		Edit		Copy		Delete	2	Zack Snyder's Justice League	Henry Cavill	Zack Snyder	Action	0	R13
<input type="checkbox"/>		Edit		Copy		Delete	3	Batman v Superman	Ben Affleck	Zack Snyder	Action	873.6	PG
<input type="checkbox"/>		Edit		Copy		Delete	4	Man of Steel	Henry Cavill	Zack Snyder	Action	668	PG
<input type="checkbox"/>		Edit		Copy		Delete	5	Wonder Woman	Gal Gadot	Patty Jenkins	Action	821.8	PG
<input type="checkbox"/>		Edit		Copy		Delete	90	Spider-Man 2	Tobey Maguire	Sam Raimi	Action	719.9	PG

☐ Check all With selected: Edit Copy Delete Export

☐ Show all | Number of rows: Filter rows: Sort by key: