

FINALS LAB TASK 5

I. PROBLEM

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. Xamp 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 SQL functions
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 Movie
2. View Movies
3. Update Movies
4. Delete a Movie
5. Search a Movie
6. Display Total Records
7. Exit

Select an option (1-6): |
```

5. The user should be able to 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 searched. 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 **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

II. CODE

The image shows a code editor with two tabs open. The left tab is named 'movieCli.py' and the right tab is named 'moviesdb.sql'. The code in 'movieCli.py' is a Python script using the mysql.connector library to interact with a MySQL database. It defines a configuration dictionary 'DB_CONFIG' with default values for host ('localhost'), user ('test_user'), password ('StrongPassword123'), database ('moviesDB'), and port (3306). It includes a function 'ask_db_credentials()' to prompt the user for database connection details, fall-backing to the defaults if no input is provided. The code then connects to the database and runs a query to select all rows from the 'movies' table, displaying the results. The 'moviesdb.sql' file contains the schema for the 'moviesDB' database, including a table 'movies' with columns: movie_id (Primary Key, int(10)), title (varchar(50)), main_actor (varchar(50)), director (varchar(50)), genre (varchar(25)), gross_sales (float), and ratings (enum(G, PG, R13, R16,X)).

```
movieCli.py
import mysql.connector
from mysql.connector import Error
import getpass
import sys

# ----- CONFIG DEFAULTS -----
DB_CONFIG = {
    'host': 'localhost',
    'user': 'test_user',
    'password': 'StrongPassword123',
    'database': 'moviesDB',
    'port': 3306
}

# -----
def ask_db_credentials():
    """Ask user for DB credentials (allow ENTER to keep defaults)."""
    print("Enter database connection details (press Enter to keep default shown in brackets.)")

    # Host
    host = input(f"Host [{DB_CONFIG['host']}]: ").strip() or DB_CONFIG['host']

    # User
    user = input(f"User [{DB_CONFIG['user']}]: ").strip() or DB_CONFIG['user']

    # Password (plain input for PyCharm compatibility)
    pwd_default = DB_CONFIG.get('password') or ''
    print("(Leave Password blank to use the default password.)")
    password = input("Password: ").strip()
    if password == "" and pwd_default != "":
        password = pwd_default

    # Database
    database = input(f"Database [{DB_CONFIG['database']}]: ").strip() or DB_CONFIG['database']

    return {
        'host': host,
        'user': user,
        'password': password,
        'database': database,
        'port': port
    }

# Port
port_input = input(f"Port [{DB_CONFIG['port']}]: ").strip()
port = int(port_input) if port_input else DB_CONFIG['port']

# Connect to database
def connect_db(cfg):
    try:
        conn = mysql.connector.connect(**cfg)
        return conn
    except Error as e:
        print(f"ERROR: Could not connect to database: {e}")
        return None

# Run query
def run_query(conn, query, params=None, fetch=False):
    try:
        cur = conn.cursor(dictionary=True)
        cur.execute(query, params or ())
        if fetch:
            rows = cur.fetchall()
            cur.close()
            return rows
        else:
            conn.commit()
            affected = cur.rowcount
            cur.close()
            return affected
    except Error as e:
        print(f"ERROR: {e}")

# -----
if __name__ == "__main__":
    cfg = ask_db_credentials()
    conn = connect_db(cfg)
    if conn:
        print("Connected to database successfully!")
        rows = run_query(conn, "SELECT * FROM movies")
        if rows:
            for row in rows:
                print(row)
        else:
            print("No rows found in movies table.")


moviesdb.sql
-- MySQL dump 10.13 Distrib 8.0.22, for Win64 (x86_64)
--
-- Host: localhost    Database: moviesDB
-- ------------------------------------------------------
-- Server version: 8.0.22 MySQL Community Server - GPL

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS */;
/*!40014 SET FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE */;
/*!40101 SET SQL_MODE='STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION' */;
/*!40101 SET @OLD_SQL_NOTES=@@SQL_NOTES */;
/*!40101 SET SQL_NOTES=0 */;

-- 
-- Table structure for table `movies`
--

CREATE TABLE `movies` (
  `movie_id` int(10) NOT NULL 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` enum('G', 'PG', 'R13', 'R16,X')
);
```

```
71     except Error as e:
72         print("Database error:", e)
73         return None
74
75     def menu(): 1 usage
76         print("\n----- MOVIE DATABASE CLI -----")
77         print("*1. Add Movie")
78         print("*2. View All Movies")
79         print("*3. Update Movie")
80         print("*4. Delete Movie")
81         print("*5. Search Movie")
82         print("*6. Display Total Records")
83         print("*7. Exit")
84         choice = input("Select an option (1-7): ").strip()
85         return choice
86
87     def input_movie_fields(existing=None): 2 usages
88         # existing: dict used for default values on update
89         def getfield_name, default=""):
90             prompt = f'{field_name}'
91             if default not in ("", None):
92                 prompt += f' [{default}]'
93             prompt += ": "
94             val = input(prompt).strip()
95             return val if val != "" else default
96
97             title = get(field_name) "Title", existing.get('title') if existing else ""
98             main_actor = get(field_name) "Main Actor", existing.get('main_actor') if existing else ""
99             director = get(field_name) "Director", existing.get('director') if existing else ""
100            genre = get(field_name) "Genre", existing.get('genre') if existing else ""
101            gross_sales_raw = get(field_name) "Gross Sales (numeric)", existing.get('gross_sales') if existing else ""
102            ratings = get(field_name) "Ratings (G, PG, R, PG-13)", existing.get('ratings') if existing else ""
103
104            gross_sales = None
```

```
105        if gross_sales_raw != "" and gross_sales_raw is not None:
106            try:
107                gross_sales = float(gross_sales_raw)
108            except ValueError:
109                print("Warning: Gross Sales invalid, stored as NULL.")
110                gross_sales = None
111
112        return {
113            'title': title,
114            'main_actor': main_actor,
115            'director': director,
116            'genre': genre,
117            'gross_sales': gross_sales,
118            'ratings': ratings
119        }
120
121    def add_movie(conn): 1 usage
122        print("\n-- Add Movie --")
123        data = input_movie_fields()
124        if not data['title'] or not data['main_actor'] or not data['director'] or not data['genre']:
125            print("Title, Main Actor, Director and Genre are required.")
126            return
127        q = """INSERT INTO movies (title, main_actor, director, genre, gross_sales, ratings)
128              VALUES (%s, %s, %s, %s, %s)"""
129        res = run_query(conn, q, params=(data['title'], data['main_actor'], data['director'],
130                                         data['genre'], data['gross_sales'], data['ratings']))
131        if res is not None:
132            print("Movie added successfully (rows affected):", res)
133
134    def view_movies(conn): 1 usage
135        print("\n-- All Movies --")
136        rows = run_query(conn, query="SELECT * FROM movies ORDER BY movie_id", fetch=True)
137        if rows is None:
138            return
```

```
139     if not rows:
140         print("No records.")
141         return
142     # print header
143     print("{:<6} {:<30} {:<20} {:<12} {:>10} {:<8}".format(
144         *args: "I0", "Title", "Main Actor", "Director", "Genre", "GrossSales", "Rate"))
145     print("-*110")
146     for r in rows:
147         print("{:<6} {:<30} {:<20} {:<12} {:>10} {:<8}".format(
148             *args: r['movie_id'],
149             (r['title'] or "")[:30],
150             (r['main_actor'] or "")[:20],
151             (r['director'] or "")[:20],
152             (r['genre'] or "")[:12],
153             f"{r['gross_sales']:.2f}" if r['gross_sales'] is not None else "",
154             r['ratings'] or ""))
155     )
156
157 def update_movie(conn): 1 usage
158     print("\n-- Update Movie --")
159     try:
160         mid = int(input("Enter movie_id to update: ").strip())
161     except ValueError:
162         print("Invalid id.")
163         return
164     rows = run_query(conn, query: "SELECT * FROM movies WHERE movie_id=%s", params: (mid,), fetch=True)
165     if rows is None:
166         return
167     if len(rows) == 0:
168         print("No movie with that id.")
169         return
170     existing = rows[0]
171     print("Current values (press Enter to keep current):")
172     new_data = input_movie_fields(existing)
```

```
173     if not new_data['title'] or not new_data['main_actor'] or not new_data['director'] or not new_data['genre']:
174         print("Title, Main Actor, Director and Genre are required.")
175         return
176     q = """UPDATE movies SET title=%s, main_actor=%s, director=%s, genre=%s,
177         gross_sales=%s, ratings=%s WHERE movie_id=%s"""
178     res = run_query(conn, q, params: (new_data['title'], new_data['main_actor'], new_data['director'],
179                                         new_data['genre'], new_data['gross_sales'], new_data['ratings'], mid))
180     if res is not None:
181         print("Movie updated (rows affected):", res)
182
183 def delete_movie(conn): 1 usage
184     print("\n-- Delete Movie --")
185     try:
186         mid = int(input("Enter movie_id to delete: ").strip())
187     except ValueError:
188         print("Invalid id.")
189         return
190     rows = run_query(conn, query: "SELECT title FROM movies WHERE movie_id=%s", params: (mid,), fetch=True)
191     if rows is None:
192         return
193     if not rows:
194         print("No movie with that id.")
195         return
196     confirm = input(f"Are you sure you want to delete '{rows[0]['title']}'? (y/N): ").strip().lower()
197     if confirm != 'y':
198         print("Deletion cancelled.")
199         return
200     res = run_query(conn, query: "DELETE FROM movies WHERE movie_id=%s", params: (mid,))
201     if res is not None:
202         print("Deleted (rows affected):", res)
203
204 def search_movie(conn): 1 usage
205     print("\n-- Search Movie --")
206     choice = input('Search by \'id\' or enter text to search title/actor/director: ').strip()
```

```
207     if choice == "":
208         print("No input.")
209         return
210
211     # If the user entered only digits, treat it as an ID search (convenience).
212     if choice.isdigit():
213         movie_id = int(choice)
214         rows = run_query(conn, query: "SELECT * FROM movies WHERE movie_id=%s", params: (movie_id,), fetch=True)
215
216     # If user literally typed 'id', ask for the id next.
217     elif choice.lower() == 'id':
218         try:
219             movie_id = int(input("Enter movie_id: ").strip())
220         except ValueError:
221             print("Invalid id.")
222             return
223         rows = run_query(conn, query: "SELECT * FROM movies WHERE movie_id=%s", params: (movie_id,), fetch=True)
224
225     # Otherwise treat the input as a text search (partial match)
226 else:
227     term = "%" + choice + "%"
228     rows = run_query(conn,
229                     query: "SELECT * FROM movies WHERE title LIKE %s OR main_actor LIKE %s OR director LIKE %s",
230                     params: (term, term, term), fetch=True)
231
232 if rows is None:
233     return
234 if not rows:
235     print("No records found.")
236     return
237
238 # Print results
239 print("{:<6} {:<30} {:<20} {:<20} {:<12} {:>10} {:<8}.format(
240     *args: "ID", "Title", "Main Actor", "Director", "Genre", "GrossSales", "Rate"))
```

```
241     print("-" * 110)
242     for r in rows:
243         print("{:<6} {:<30} {:<20} {:<20} {:<12} {:>10} {:<8}.format(
244             *args: r['movie_id'],
245             (r['title'] or "")[:30],
246             (r['main_actor'] or "")[:20],
247             (r['director'] or "")[:20],
248             (r['genre'] or "")[:12],
249             f"{r['gross_sales']:.2f}" if r['gross_sales'] is not None else **,
250             r['ratings'] or **
251         ))
252
253 def display_count(conn): 1 usage
254     rows = run_query(conn, query: "SELECT COUNT(*) AS total FROM movies", fetch=True)
255     if rows is None:
256         return
257     print("Total number of movies:", rows[0]['total'])
258
259 def main(): 1 usage
260     cfg = ask_db_credentials()
261     conn = connect_db(cfg)
262     if conn is None:
263         print("Exiting.")
264         sys.exit(1)
265
266     print("Connected to database.")
267     try:
268         while True:
269             choice = menu()
270             if choice == '1':
271                 add_movie(conn)
272             elif choice == '2':
273                 view_movies(conn)
274             elif choice == '3':
```

```
275             update_movie(conn)
276     elif choice == '4':
277         delete_movie(conn)
278     elif choice == '5':
279         search_movie(conn)
280     elif choice == '6':
281         display_count(conn)
282     elif choice == '7':
283         print("Goodbye.")
284         break
285     else:
286         print("Invalid selection. Choose 1-7.")
287     finally:
288         conn.close()
289
290 if __name__ == "__main__":
291     main()
```

III. SAMPLE OUTPUT

```
"D:\School\School Works\Pycharm\.venv\Scripts\python.exe" "D:\School\School Works\Pycharm\Finals\CLI Act\movieCLI.py"
Enter database connection details (press Enter to keep default shown in brackets).
Host [localhost]:
User [test_user]:
(Leave Password blank to use the default password.)
Password:
Database [moviesDB]:
Port [3306]:
Connected to database.

----- MOVIE DATABASE CLI -----
1. Add Movie
2. View All Movies
3. Update Movie
4. Delete Movie
5. Search Movie
6. Display Total Records
7. Exit
Select an option (1-7): 2

-- All Movies --
ID      Title           Main Actor       Director        Genre       GrossSales Rate
-----
1      Blade Runner     Harrison Ford    Ridley Scott   Sci-Fi      327.50 R
2      The Matrix       Keanu Reeves     Wachowski     Sci-Fi      463.50 R
3      Spirited Away   Rumi Hiragi     Hayao Miyazaki Animation  347.70 G
4      The Godfather    Marlon Brando   Francis Ford Coppola Crime    246.00 R
5      Interstellar     Matthew McConaughey Christopher Nolan Sci-Fi      677.50 PG-13

----- MOVIE DATABASE CLI -----
1. Add Movie
2. View All Movies
3. Update Movie
4. Delete Movie
5. Search Movie
6. Display Total Records
```

```
7. Exit
Select an option (1-7): 1

-- Add Movie --
Title: The Lost Kingdom
Main Actor: Daniel Reyes
Director: Maria Santos
Genre: Adventure
Gross Sales (numeric): 152.7
Ratings (G, PG, R, PG-13): PG
Movie added successfully (rows affected): 1

----- MOVIE DATABASE CLI -----
1. Add Movie
2. View All Movies
3. Update Movie
4. Delete Movie
5. Search Movie
6. Display Total Records
7. Exit
Select an option (1-7): 2

-- All Movies --
ID      Title           Main Actor       Director        Genre      GrossSales Rate
-----+
1       Blade Runner     Harrison Ford   Ridley Scott   Sci-Fi    327.50 R
2       The Matrix        Keanu Reeves    Wachowski     Sci-Fi    463.50 R
3       Spirited Away    Rumi Hiragi    Hayao Miyazaki Animation 347.70 G
4       The Godfather     Marlon Brando  Francis Ford Coppola Crime 246.00 R
5       Interstellar      Matthew McConaughey Christopher Nolan Sci-Fi    677.50 PG-13
7       The Lost Kingdom Daniel Reyes    Maria Santos   Adventure 152.70 PG

----- MOVIE DATABASE CLI -----
1. Add Movie
2. View All Movies
3. Update Movie
```

```
4. Delete Movie
5. Search Movie
6. Display Total Records
7. Exit
Select an option (1-7): 3

-- Update Movie --
Enter movie_id to update: 7
Current values (press Enter to keep current):
Title [The Lost Kingdom]:
Main Actor [Daniel Reyes]:
Director [Maria Santos]:
Genre [Adventure]: Drama
Gross Sales (numeric) [152.7]: 160.9
Ratings (G, PG, R, PG-13) [PG]: PG-13
Movie updated (rows affected): 1

----- MOVIE DATABASE CLI -----
1. Add Movie
2. View All Movies
3. Update Movie
4. Delete Movie
5. Search Movie
6. Display Total Records
7. Exit
Select an option (1-7): 2

-- All Movies --
ID      Title           Main Actor       Director        Genre      GrossSales Rate
-----+
1       Blade Runner     Harrison Ford   Ridley Scott   Sci-Fi    327.50 R
2       The Matrix        Keanu Reeves    Wachowski     Sci-Fi    463.50 R
3       Spirited Away    Rumi Hiragi    Hayao Miyazaki Animation 347.70 G
4       The Godfather     Marlon Brando  Francis Ford Coppola Crime 246.00 R
5       Interstellar      Matthew McConaughey Christopher Nolan Sci-Fi    677.50 PG-13
7       The Lost Kingdom Daniel Reyes    Maria Santos   Drama    160.90 PG-13
```

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

-- Delete Movie --
Enter movie_id to delete: 7
Are you sure you want to delete 'The Lost Kingdom'? (y/N): Y
\Deleted (rows affected): 1

----- MOVIE DATABASE CLI -----
1. Add Movie
2. View All Movies
3. Update Movie
4. Delete Movie
5. Search Movie
6. Display Total Records
7. Exit
Select an option (1-7): 2
Invalid selection. Choose 1-7.

----- MOVIE DATABASE CLI -----
1. Add Movie
2. View All Movies
3. Update Movie
4. Delete Movie
5. Search Movie
6. Display Total Records
7. Exit
Select an option (1-7): 2

----- MOVIE DATABASE CLI -----
ID      Title           Main Actor       Director        Genre     GrossSales Rate
---- -----
1      Blade Runner    Harrison Ford   Ridley Scott   Sci-Fi    327.50 R
2      The Matrix      Keanu Reeves    Wachowski     Sci-Fi    463.50 R
3      Spirited Away  Rumi Hiiragi   Hayao Miyazaki Animation 347.70 G
4      The Godfather   Marlon Brando  Francis Ford Coppola Crime 246.00 R
5      Interstellar    Matthew McConaughey Christopher Nolan Sci-Fi   677.50 PG-13

----- MOVIE DATABASE CLI -----
1. Add Movie
2. View All Movies
3. Update Movie
4. Delete Movie
5. Search Movie
6. Display Total Records
7. Exit
Select an option (1-7): 5

----- MOVIE DATABASE CLI -----
Search by 'id' or enter text to search title/actor/director: 4
ID      Title           Main Actor       Director        Genre     GrossSales Rate
---- -----
4      The Godfather   Marlon Brando  Francis Ford Coppola Crime 246.00 R
```

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

Process finished with exit code 0
```