TECH STARTER

1.6.1. Gruppenaufgabe Filmverwaltungsystem

Use Case

Du bist ein Filmfan und möchtest deine persönliche Filmsammlung digital verwalten. Mit diesem System kannst du Filme hinzufügen, aktualisieren, löschen und nach verschiedenen Kriterien suchen. So behältst du immer den Überblick über deine Sammlung.

Aufgabenbeschreibung

- Datenbank einrichten:
- Erstelle eine Tabelle `films` mit den Feldern:
  - `id` (Primärschlüssel)
  - `title` (VARCHAR)
  - `director` (VARCHAR)
  - `genre` (VARCHAR)
  - `release` (INTEGER)
  - `rating` (FLOAT)

- Funktionen implementieren:
- Film hinzufügen
- Film löschen
- Filminformationen aktualisieren
- Nach Filmen suchen

- Erweiterte Funktionen:
- Suche nach Genre, Regisseur oder Erscheinungsjahr.
- Sortiere Filme nach Bewertung oder Erscheinungsjahr.

Tipps

- Datenvalidierung:
  - Stelle sicher, dass das `release` eine gültige Jahreszahl ist.
  - Die `rating` sollte zwischen 0 und 10 liegen.

- Benutzerfreundlichkeit:
  - Gib klare Anweisungen und Rückmeldungen im Programm aus.
  - Verwende Try-Except-Blöcke, um Eingabefehler abzufangen.

Code Snippets

1. Film hinzufügen:

```
def add_movie(title, director, genre, releasedate, rating):
    connection = create_connection()
    cursor = connection.cursor()
    sql = "INSERT INTO filme (titel, director, genre, releaseyear, rate)
VALUES (%s, %s, %s, %s, %s)"
    values = (name, director, genre, release, rateing)
    cursor.execute(sql, values)
    connection.commit()
    cursor.close()
    connection.close()
```

Kleine Falle: Überprüfe, ob das `erscheinungsjahr` wirklich ein Integer ist und nicht versehentlich als String gespeichert wird.

2. Nach Filmen suchen:

```python
def search_movies(kriterium, wert):
    connection = create_connection()
    cursor = connection.cursor()
    sql = f"SELECT * FROM filme WHERE {kriterium} = %s"
    cursor.execute(sql, (wert,))
    result = cursor.fetchall()
    for row in result:
        print(f"Titel: {row[1]}, Regisseur: {row[2]}, Genre: {row[3]},
Jahr: {row[4]}, Bewertung: {row[5]}")
    cursor.close()
    connection.close()
```

Kleine Falle: Wenn `kriterium` direkt in die SQL-Abfrage eingefügt wird, kann das zu SQL-Injection führen. Stelle sicher, dass `kriterium` ein erlaubtes Feld ist und nicht aus Benutzereingaben stammt. Desweiteren prüfe mal ob die Art des Einfügens von Werten in ein SQL Query so sicher ist.

Allgemeine Hinweise für alle Projekte

- Netzwerkverbindung zwischen VMs:
- Stelle sicher, dass die Python-VM die Datenbank-VM über das Netzwerk erreichen kann.

- Datenbank-Berechtigungen:

- Erstelle einen Datenbankbenutzer mit den notwendigen Berechtigungen (SELECT, INSERT, UPDATE, DELETE).
- **Kleine Falle:** Wenn du die Standard-Benutzeranmeldeinformationen verwendest, könnte das Sicherheitsrisiken mit sich bringen.

- Dokumentation:
- Kommentiere deinen Code ausführlich.
- Erstelle eine README-Datei mit Installationsanweisungen und einer Beschreibung, wie das Programm verwendet wird.

- Versionskontrolle:
- Verwende Git, um Änderungen an deinem Code zu verfolgen.
- Kleine Falle: Vergiss nicht, sensible Informationen wie Passwörter oder geheime Schlüssel aus dem Repository auszuschließen.

Abschließender Hinweis:

Achte darauf, jeden Schritt gründlich zu lesen und zu verstehen, bevor du ihn ausführst oder kopierst. Manchmal können kleine Details einen großen Unterschied machen. Viel Spaß beim Programmieren!