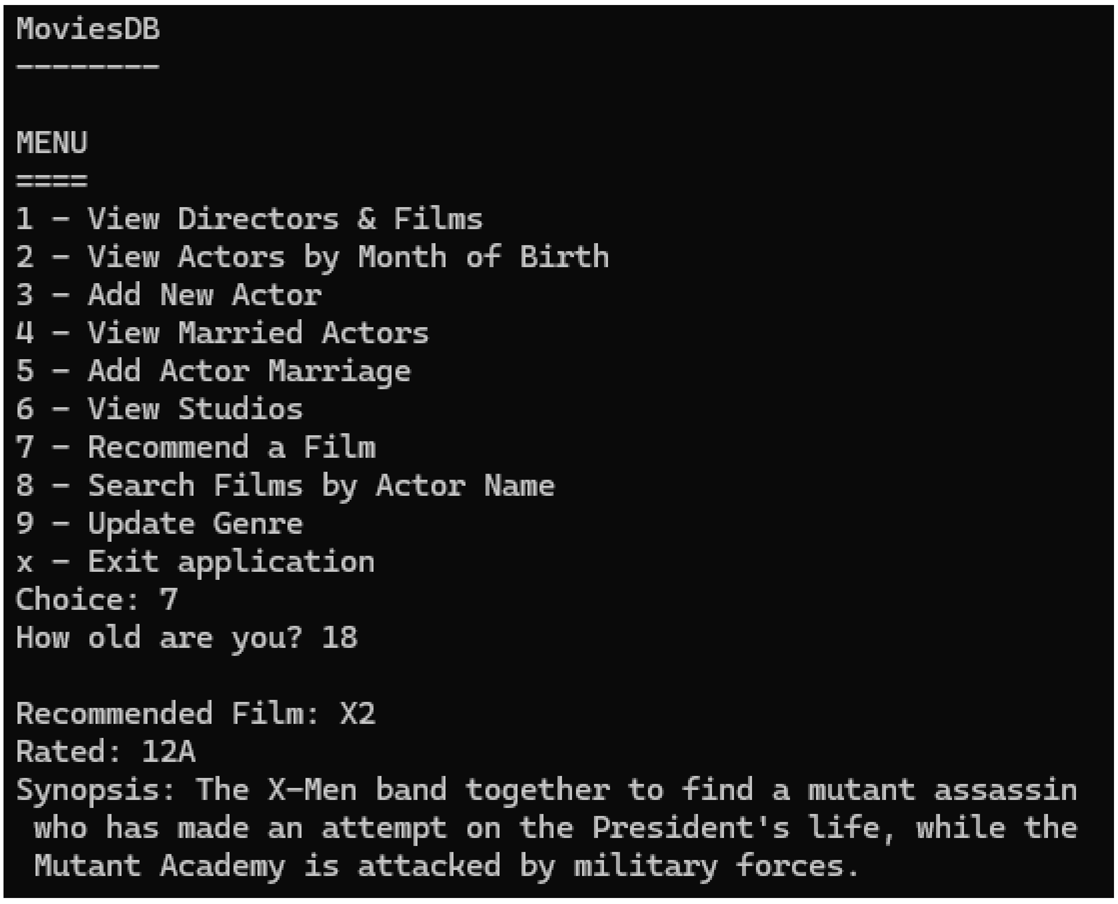
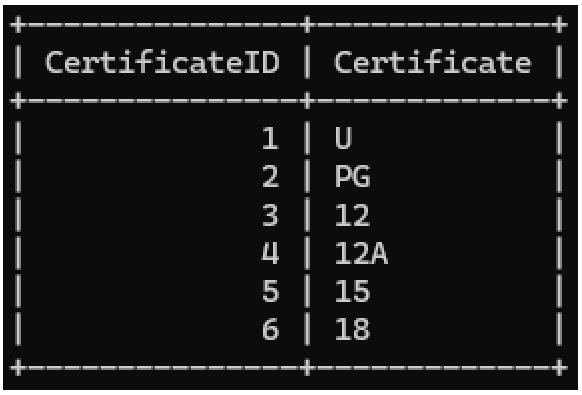
Innovation:

Option 7:



```
MoviesDB
--------

MENU
====
1 - View Directors & Films
2 - View Actors by Month of Birth
3 - Add New Actor
4 - View Married Actors
5 - Add Actor Marriage
6 - View Studios
7 - Recommend a Film
8 - Search Films by Actor Name
9 - Update Genre
x - Exit application
Choice: 7
How old are you? 18

Recommended Film: X2
Rated: 12A
Synopsis: The X-Men band together to find a mutant assassin
 who has made an attempt on the President's life, while the
 Mutant Academy is attacked by military forces.
```

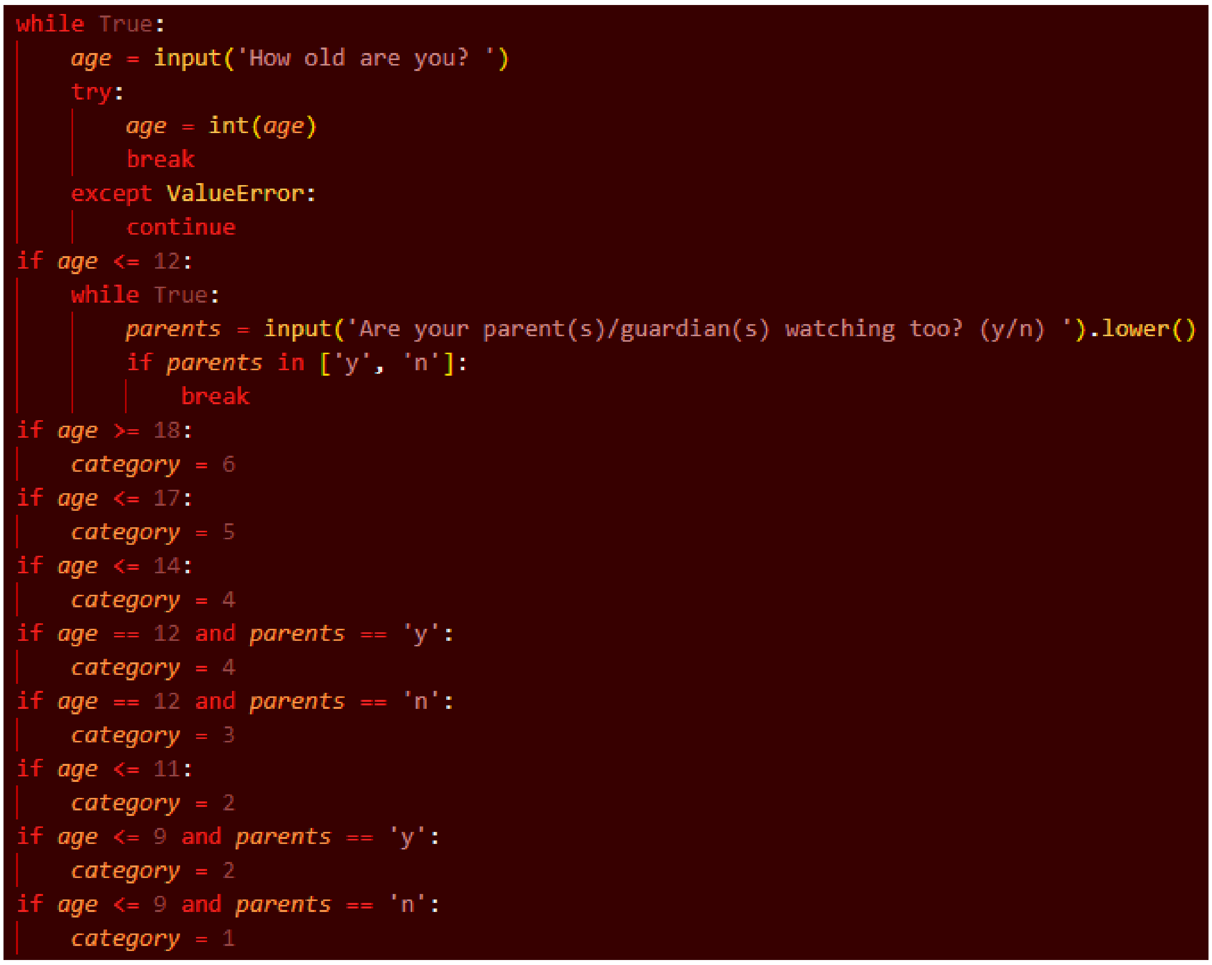
I created a seventh option for the app that generates a film recommendation based on age. If appropriate it will also ask if a parent or guardian is present. Then, based on the user input it will select a film at random from a list of age appropriate films and return the film name, rating and synopsis.



```
+---------------+-------------+
| CertificateID | Certificate |
+---------------+-------------+
|             1 | U           |
|             2 | PG          |
|             3 | 12          |
|             4 | 12A         |
|             5 | 15          |
|             6 | 18          |
+---------------+-------------+
```

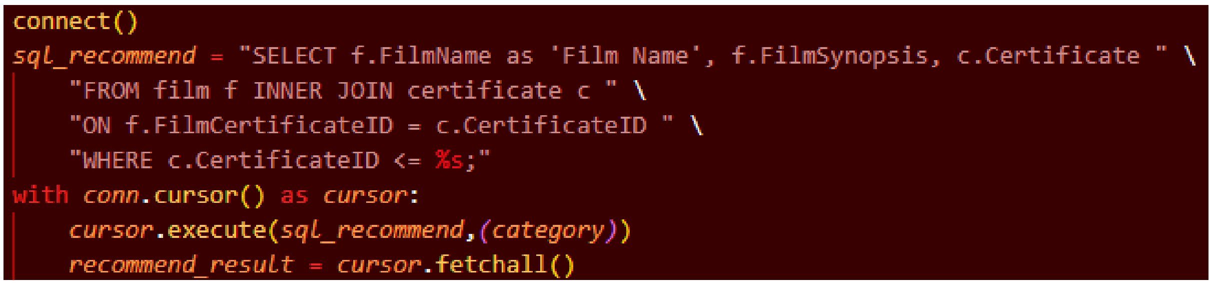
The certificates are as above.

I decided that children 9 and below that have their parents or guardians present would be recommended films PG and lower, otherwise they would be recommended films rated U only. Children between 10 and 11 would be recommended PG films by default. Films rated 12A would only be recommended to children aged 12 with a parent or guardian present while 13 and 14 year olds would be recommended these by default. Children aged 15, 16 and 17 would be recommended films 15 and below. And finally anyone aged 18 and above would receive recommendations from all classifications. I used this reference to base my decisions.



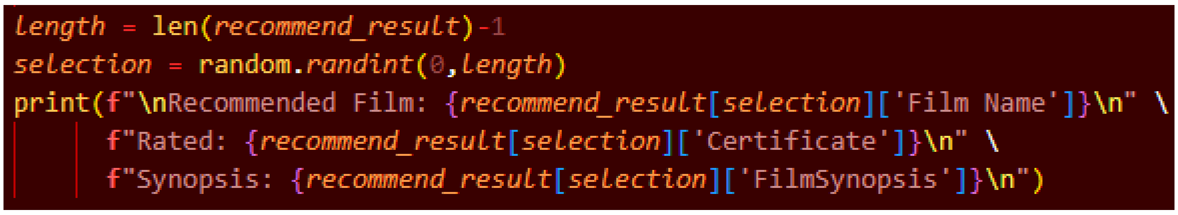
```python
while True:
    age = input('How old are you? ')
    try:
        age = int(age)
        break
    except ValueError:
        continue
if age <= 12:
    while True:
        parents = input('Are your parent(s)/guardian(s) watching too? (y/n) ').lower()
        if parents in ['y', 'n']:
            break
if age >= 18:
    category = 6
if age <= 17:
    category = 5
if age <= 14:
    category = 4
if age == 12 and parents == 'y':
    category = 4
if age == 12 and parents == 'n':
    category = 3
if age <= 11:
    category = 2
if age <= 9 and parents == 'y':
    category = 2
if age <= 9 and parents == 'n':
    category = 1
```

The appropriate film rating is selected using a series of if statements..



```python
connect()
sql_recommend = "SELECT f.FilmName as 'Film Name', f.FilmSynopsis, c.Certificate " \
    "FROM film f INNER JOIN certificate c " \
    "ON f.FilmCertificateID = c.CertificateID " \
    "WHERE c.CertificateID <= %s;"
with conn.cursor() as cursor:
    cursor.execute(sql_recommend,(category))
    recommend_result = cursor.fetchall()
```

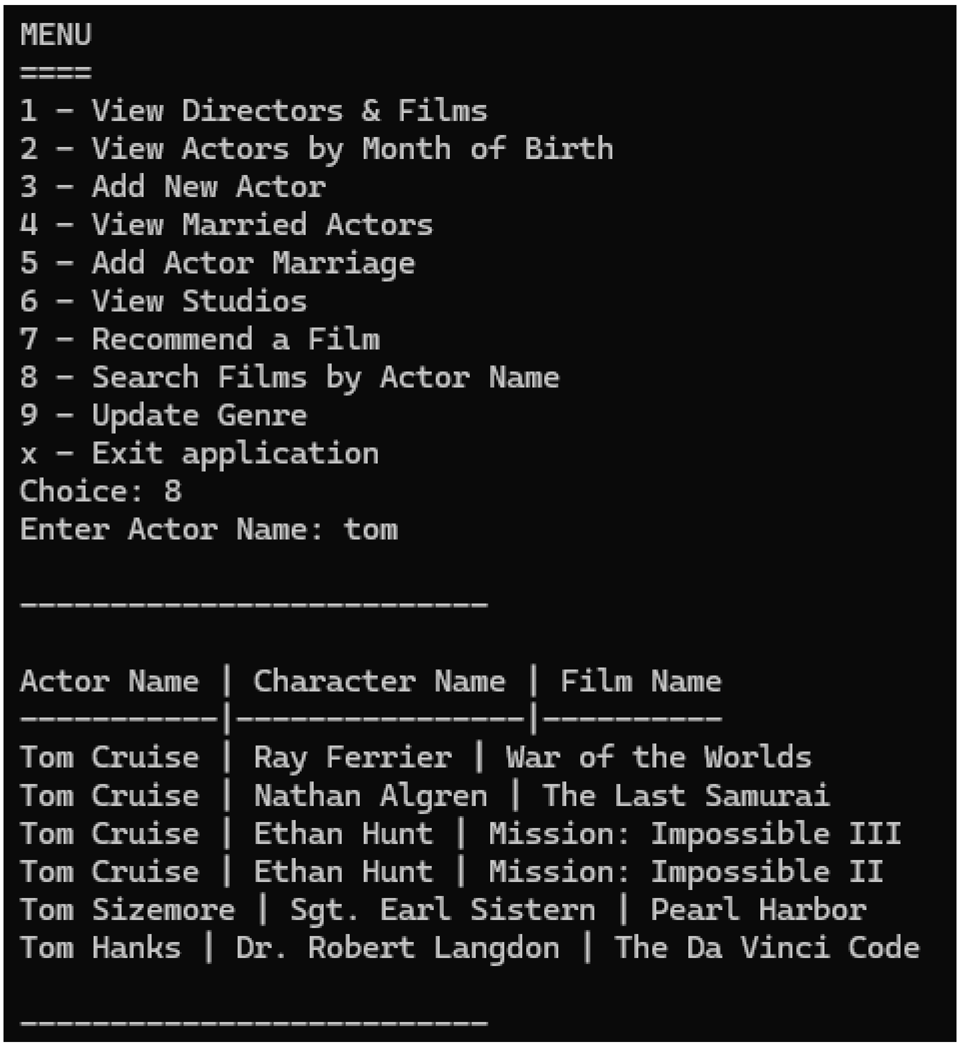
It then fetches all appropriate films of that age rating and below.



```python
length = len(recommend_result)-1
selection = random.randint(0,length)
print(f"\nRecommended Film: {recommend_result[selection]['Film Name']}\n" \
      f"Rated: {recommend_result[selection]['Certificate']}\n" \
      f"Synopsis: {recommend_result[selection]['FilmSynopsis']}\n")
```

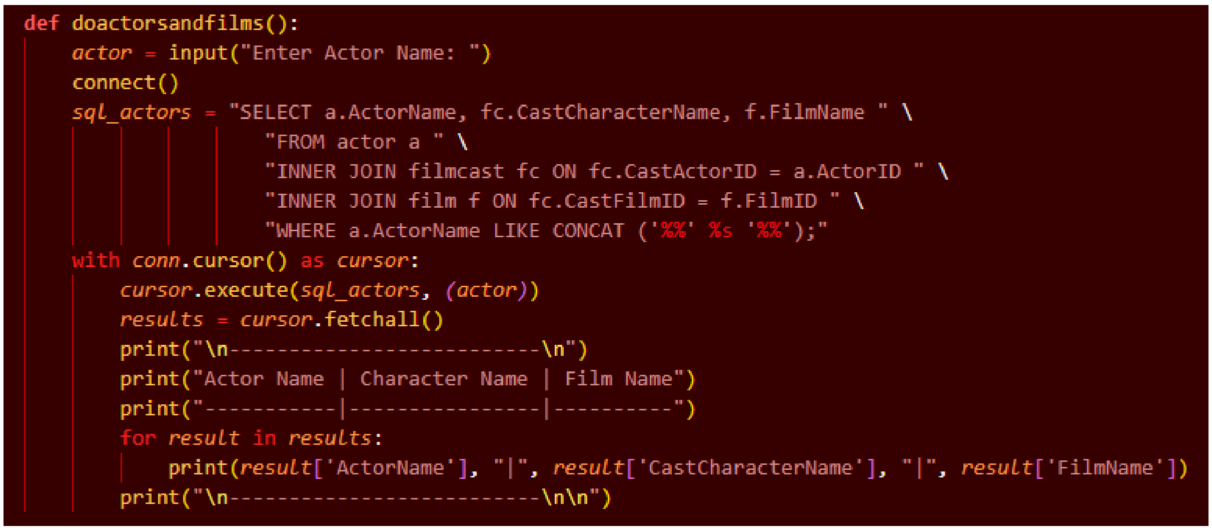
Then using the random package, it generates a random number and selects the film at that index and displays the title, rating and synopsis to the user.

Option 8:



```
MENU
====
1 - View Directors & Films
2 - View Actors by Month of Birth
3 - Add New Actor
4 - View Married Actors
5 - Add Actor Marriage
6 - View Studios
7 - Recommend a Film
8 - Search Films by Actor Name
9 - Update Genre
x - Exit application
Choice: 8
Enter Actor Name: tom

--------------------------


Actor Name | Character Name | Film Name
-----------|----------------|----------
Tom Cruise | Ray Ferrier | War of the Worlds
Tom Cruise | Nathan Algren | The Last Samurai
Tom Cruise | Ethan Hunt | Mission: Impossible III
Tom Cruise | Ethan Hunt | Mission: Impossible II
Tom Sizemore | Sgt. Earl Sistern | Pearl Harbor
Tom Hanks | Dr. Robert Langdon | The Da Vinci Code

--------------------------
```

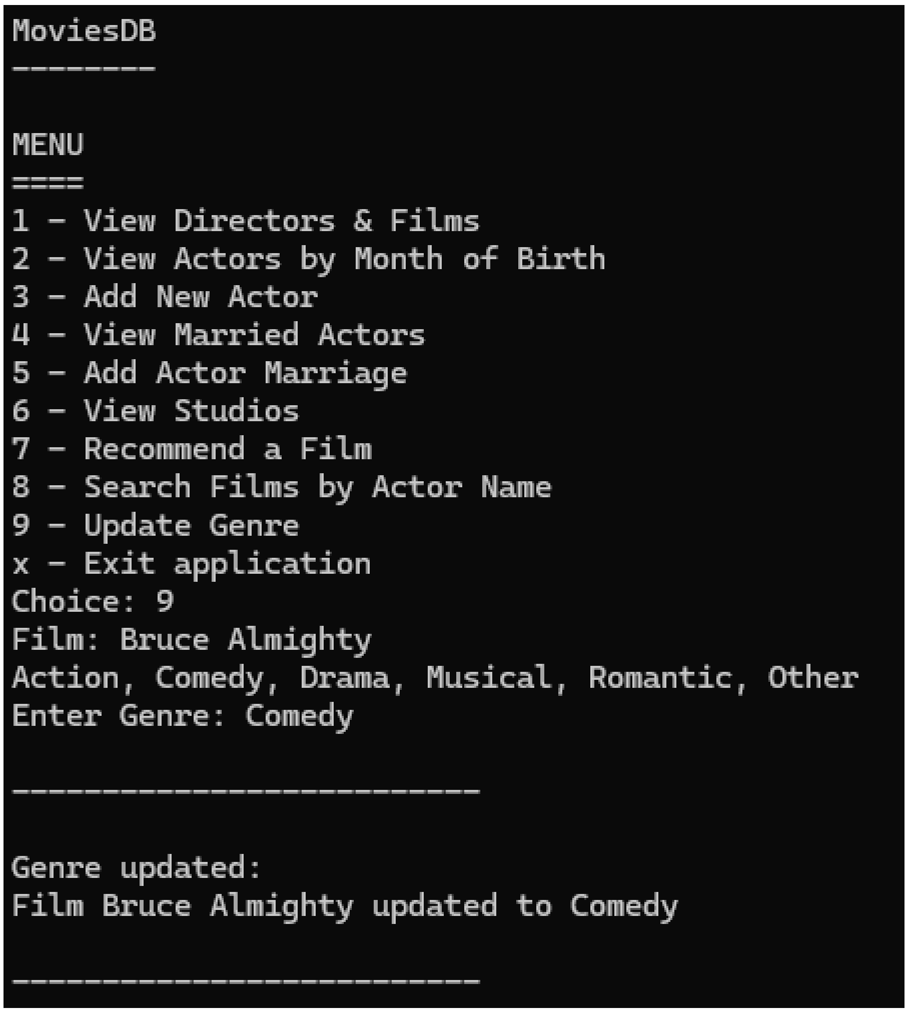
I created an eighth option that searches the database by actor and returns the actor name, character name and film name.



```python
def doactorsandfilms():
    actor = input("Enter Actor Name: ")
    connect()
    sql_actors = "SELECT a.ActorName, fc.CastCharacterName, f.FilmName " \
                 "FROM actor a " \
                 "INNER JOIN filmcast fc ON fc.CastActorID = a.ActorID " \
                 "INNER JOIN film f ON fc.CastFilmID = f.FilmID " \
                 "WHERE a.ActorName LIKE CONCAT ('%%' %s '%%');"
    with conn.cursor() as cursor:
        cursor.execute(sql_actors, (actor))
        results = cursor.fetchall()
        print("\n-------------------------\n")
        print("Actor Name | Character Name | Film Name")
        print("-----------|----------------|----------")
        for result in results:
            print(result['ActorName'], "|", result['CastCharacterName'], "|", result['FilmName'])
        print("\n-------------------------\n\n")
```

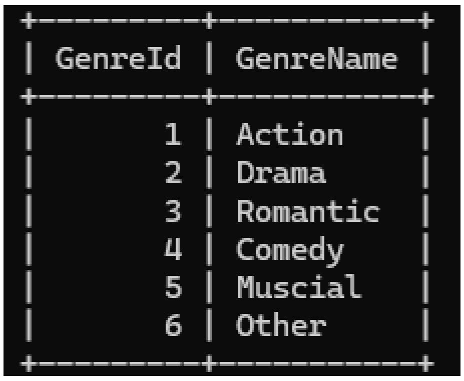
It does this by creating a join on ActorID and FilmID and returning the appropriate results.

Option 9:

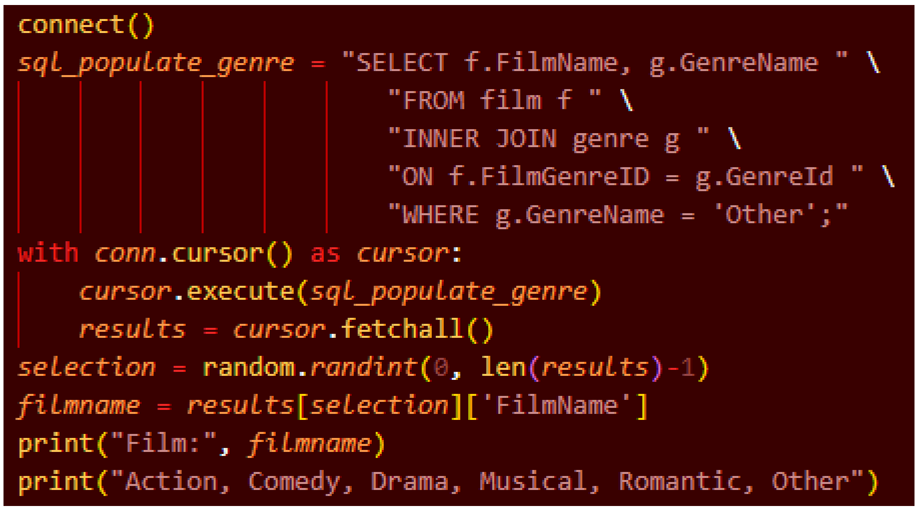


```
MoviesDB
--------

MENU
====
1 - View Directors & Films
2 - View Actors by Month of Birth
3 - Add New Actor
4 - View Married Actors
5 - Add Actor Marriage
6 - View Studios
7 - Recommend a Film
8 - Search Films by Actor Name
9 - Update Genre
x - Exit application
Choice: 9
Film: Bruce Almighty
Action, Comedy, Drama, Musical, Romantic, Other
Enter Genre: Comedy

--------------------------

Genre updated:
Film Bruce Almighty updated to Comedy

--------------------------
```

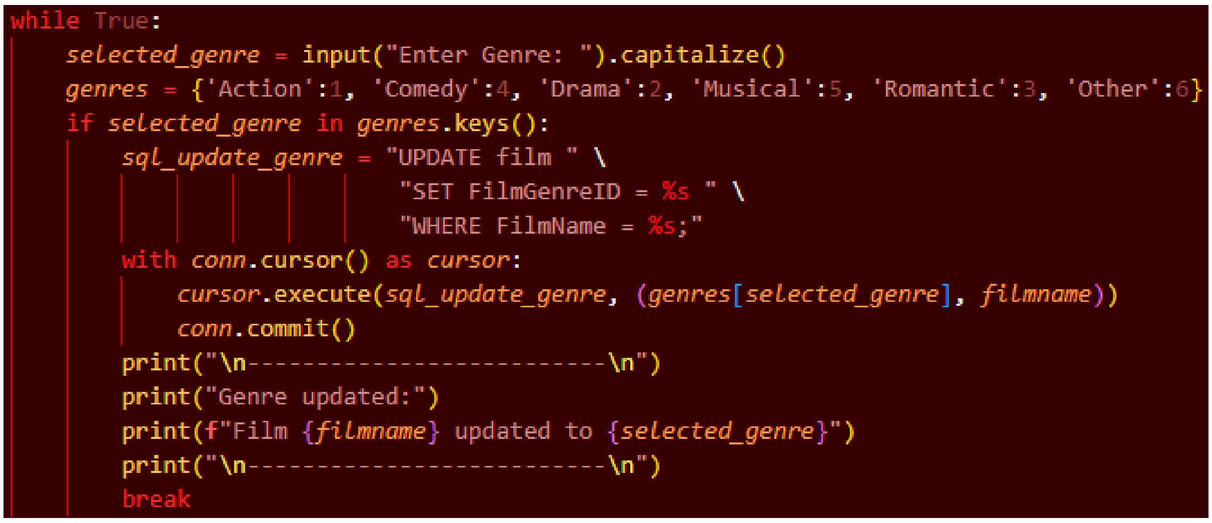
I created a ninth option to update the genre of a random film.


```
+---------+-----------+
| GenreId | GenreName |
+---------+-----------+
|       1 | Action    |
|       2 | Drama     |
|       3 | Romantic  |
|       4 | Comedy    |
|       5 | Muscial   |
|       6 | Other     |
+---------+-----------+
```

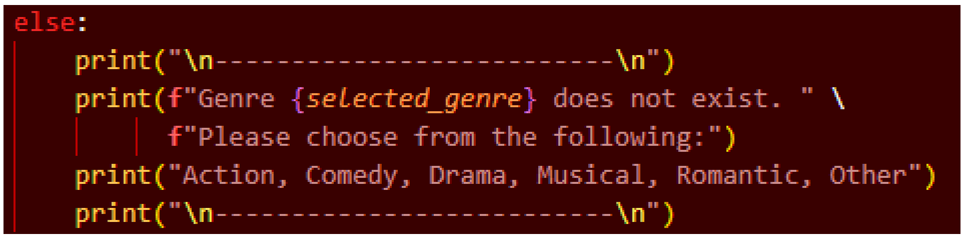
Films have been assigned the above genres, so they are the only ones the user can choose from when updating the genre of a film.


```python
connect()
sql_populate_genre = "SELECT f.FilmName, g.GenreName " \
                     "FROM film f " \
                     "INNER JOIN genre g " \
                     "ON f.FilmGenreID = g.GenreId " \
                     "WHERE g.GenreName = 'Other';"
with conn.cursor() as cursor:
    cursor.execute(sql_populate_genre)
    results = cursor.fetchall()
selection = random.randint(0, len(results)-1)
filmname = results[selection]['FilmName']
print("Film:", filmname)
print("Action, Comedy, Drama, Musical, Romantic, Other")
```

When the ninth option is selected, a film is chosen at random from a list of films that have been assigned a genre of ‘Other’ and the user is presented with the list of available genres to choose from.


```python
while True:
    selected_genre = input("Enter Genre: ").capitalize()
    genres = {'Action':1, 'Comedy':4, 'Drama':2, 'Musical':5, 'Romantic':3, 'Other':6}
    if selected_genre in genres.keys():
        sql_update_genre = "UPDATE film " \
                           "SET FilmGenreID = %s " \
                           "WHERE FilmName = %s;"
        with conn.cursor() as cursor:
            cursor.execute(sql_update_genre, (genres[selected_genre], filmname))
            conn.commit()
        print("\n-------------------------\n")
        print("Genre updated:")
        print(f"Film {filmname} updated to {selected_genre}")
        print("\n-------------------------\n")
        break
```

A user input is then taken. This user input is processed by the script and updates the film genre as appropriate.



```
else:
    print("\n--------------------------\n")
    print(f"Genre {selected_genre} does not exist. " \
            f"Please choose from the following:")
    print("Action, Comedy, Drama, Musical, Romantic, Other")
    print("\n--------------------------\n")
```

If the user inputs anything other than the options available then they are reminded of the options and then prompted to enter again using a while loop.