CBSE

Computer Science Project



Blockbuster:   
Database Management system for movies

Made By: Atharava Srivastava

Class: 12th Aryabhatta

Roll No.: 4

Acknowledgement

I would like to express mt gratitude towards my Computer Science Teacher, Mrs. Shruti Mehta for her valuable guidance, I would also like to thank our Principal Ms. Neera Pandey and the School Management for providing me the opportunity to work on this project.

I am grateful to my parents and my brother for their consistent support which made this project successful. Lastly, I would like to extend thanks to my classmates who helped me during the making of this project.

Contents

1. Story behind the name ‘Blockbuster’
2. Purpose of the project
3. Requirements  
    Hardware used in the project  
    Software used in the project
4. Implementation  
    Attributes used  
    Features available  
    Source Code  
    User Interface  
    MySQL Table
5. Recommendations
6. Conclusion
7. Bibliography

# Story behind the name

Blockbuster was a business founded by David Cook in 1985 as a single home video rental shop. As the company saw growth it became a public store chain featuring video game rentals, DVD-by-mail, streaming, video on demand and cinema theater.

Poor leadership and the residual effects of the Great Recession were major factors leading to Blockbuster’s decline. The competition from Netflix’s mail order service and Redbox automated kiosks ultimately led to the company filing for bankruptcy in 2010.

In 2011, its remaining 1700 stores were bought by Dish Network, a satellite television provider. By 2014, the last 300 company-owned stores were closed.

Dish retained a small number of franchise agreements, enabling some privately owned franchises to remain open irrespective of the termination of the corporate support for the brand.

Following a series of further closures in 2019, only one franchise store remains open in Bend, Oregon, United States.



**The Last Blockbuster, Bend, Oregon, USA**

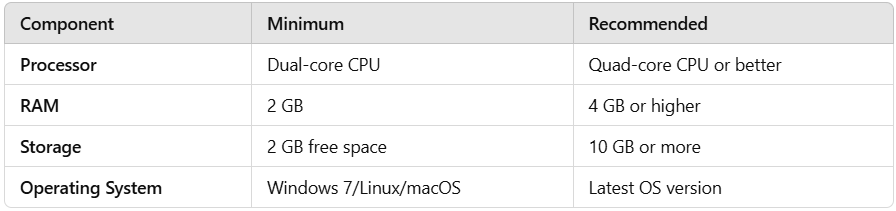
Purpose of the project

I decided to create this project so that I would be able to maintain a database for movies which allowed me to store information pertaining to the movie and let me access it easily.

Although such services are already available, creating this project also helped learn about working with user interfaces while using Tkinter to create the UI for my project.

Creating this project also helped me become more familiar with Python and its connectivity with MySQL.

Requirements

1. Hardware requirements:
2. Software requirements:  
   Python Version 3.13  
   Tkinter Version 8.6  
   MySQL Command line client Version 8.0

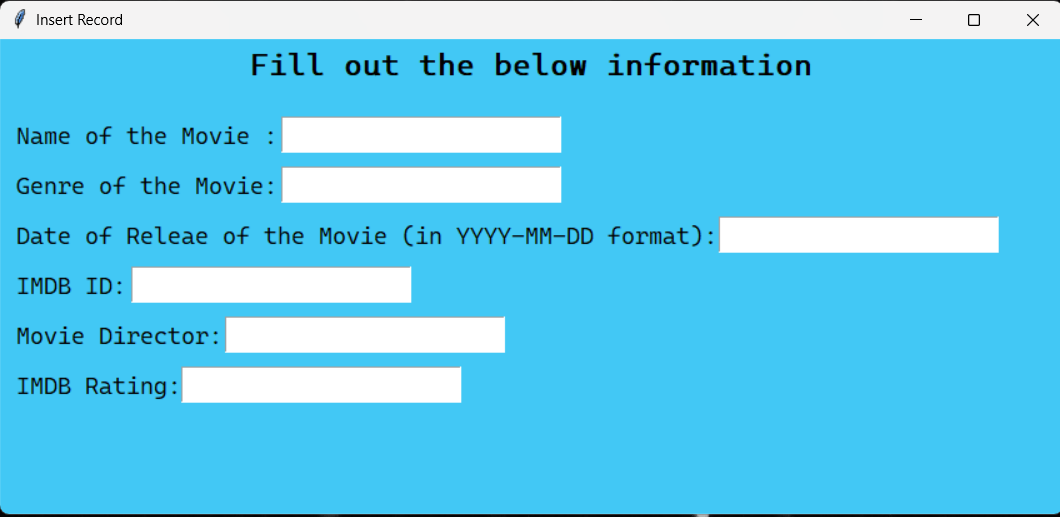
Implementation

1. Attributes used:

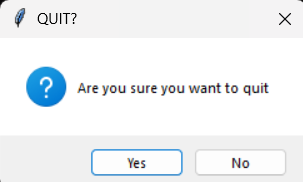
Movie Name  
Genre  
Date of Release  
IMDB ID

Director   
Rating

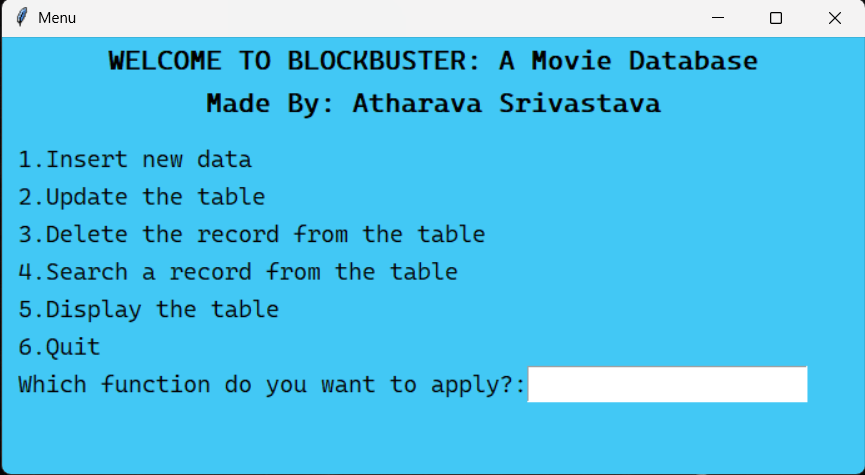
1. Features available:  
   Insert records  
   Update a record using name or IMDB ID of the movie  
   Delete a record using name or IMDB ID of the movie  
   Check availability of a record using name or IMDB ID of the movie  
   Display all the records available  
   Exit the applet
2. Source code:
3. *#Importing required libraries*
4. import tkinter as tk
5. from tkinter import StringVar
6. from tkinter import messagebox
7. import mysql.connector as sq
9. *#Creating the database Movie\_database and a table inside it called Movies*
10. **def** **create**():
11. mydb = sq.connect(host="localhost",user="root",password="root")
12. mycursor = mydb.cursor()
13. sql = "Create database Movie\_database"
14. mycursor.execute(sql)
15. mycursor.execute("Use Movie\_database")
16. mydb.commit()
17. mycursor.execute("Create table Movies (Movie\_Name VARCHAR(500), Genre VARCHAR(100), date\_of\_release DATE, IMDB\_id INTEGER,Director VARCHAR(500), Rating VARCHAR(500))")
18. mydb.commit()
19. *#Checking the existence of the database and creating it if the database does not exist*
20. **def** **check\_database\_existence**():
21. try:
22. mydb = sq.connect(host="localhost",user="root",password="root",database="Movie\_database")
23. except sq.Error as e:
24. if e.errno == 1049:  *#1049 is the MySQL error code for "Unknown database"*
25. create()
26. else:
27. **print**(**f**"Error: {e}")
28. check\_database\_existence()
29. *#Re-opening the menu window whenever the close button is clicked.*
30. **def** **recreate\_root**():
31. global root
32. root=tk.Tk()
33. root.geometry('690x350')
34. root.configure(bg='#42c8f5')
35. root.title('Menu')
36. lab3=tk.Label(root, text="WELCOME TO BLOCKBUSTER: A Movie Database",font=('Cascadia Mono SemiLight',16,'bold'),bg='#42c8f5')
37. lab6=tk.Label(root, text="Made By: Atharava Srivastava",font=('Cascadia Mono SemiLight',16,'bold'),bg='#42c8f5')
38. lab3.pack()
39. lab6.pack()
40. lin1=tk.Label(root,text="1.Insert new records",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
41. lin2=tk.Label(root,text="2.Update a record",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
42. lin3=tk.Label(root,text="3.Delete a record",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
43. lin4=tk.Label(root,text="4.Search a record",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
44. lin5=tk.Label(root,text="5.Display the data",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
45. lin6=tk.Label(root,text="6.Quit",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
46. lin1.place(x=10,y=80)
47. lin2.place(x=10,y=110)
48. lin3.place(x=10,y=140)
49. lin4.place(x=10,y=170)
50. lin5.place(x=10,y=200)
51. lin6.place(x=10,y=230)
52. ch=StringVar()
54. lab1=tk.Label(root,text="Which function do you want to apply?:",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
55. lab1.place(x=10,y=260)
56. en1=tk.Entry(root, textvariable=ch, font=('Cascadia Mono SemiLight',14))
57. en1.place(x=420,y=263)
58. *#Function to ask for confirmation from user if quiting*
59. **def** **on\_closing**():
60. if messagebox.askyesno(title='QUIT?',message='Are you sure you want to quit'):
61. root.destroy()
62. else:
63. pass
65. *#Function to accept the choice from user of menu items*
66. **def** **choicefunc**(event=None):
67. choice=ch.get()
68. *#To insert new records*
69. if choice=='1':
70. insert()
71. *#To update a record*
72. elif choice=='2':
73. update()
74. *#To delete a record*
75. elif choice=='3':
76. delete()
77. *#To search a record*
78. elif choice=='4':
79. search()
80. *#To display the data*
81. elif choice=='5':
82. display()
83. *#To exit the program*
84. elif choice=='6':
85. exit\_=tk.Tk()
86. exit\_.geometry('500x100')
87. exit\_.config(bg='#42c8f5')
88. exit\_.title('Exit')
89. label\_0=tk.Label(exit\_, text="Thank You!",font=('Cascadia Mono SemiLight',16,'bold'),bg='#42c8f5')
90. label\_1=tk.Label(exit\_, text="Hope you have a nice day!",font=('Cascadia Mono SemiLight',16,'bold'),bg='#42c8f5')
91. label\_0.pack()
92. label\_1.pack()
93. root.destroy()
94. *#Invaild input*
95. else:
96. lab2=tk.Label(root,text="Please Enter Valid Input!",font=('Cascadia Mono SemiLight',15),bg='#42c8f5')
97. lab2.place(x=170,y=300)
98. ch.set('')
99. en1.bind ('<Return>',choicefunc)
101. *#Recreating the option window*
102. root.mainloop()
104. *#Function for inserting data*
105. **def** **insert**():
107. *#Closing menu window*
108. root.destroy()
110. *#Creating window to insert data*
111. insertk=tk.Tk()
112. insertk.geometry('850x380')
113. insertk.configure(bg='#42c8f5')
114. insertk.title('Insert Record')
116. *#Creating labels*
117. lab0=tk.Label(insertk,text='Fill out the below information',bg='#42c8f5',font=('Cascadia Mono SemiLight',18,'bold'))
118. lab1=tk.Label(insertk,text="Name of the Movie :",bg='#42c8f5',font=('Cascadia Mono SemiLight',14))
119. lab2=tk.Label(insertk,text="Genre of the Movie:",bg='#42c8f5',font=('Cascadia Mono SemiLight',14))
120. lab3=tk.Label(insertk,text="Date of Releae of the Movie (in YYYY-MM-DD format):",bg='#42c8f5',font=('Cascadia Mono SemiLight',14))
121. lab4=tk.Label(insertk,text="IMDB ID:",bg='#42c8f5',font=('Cascadia Mono SemiLight',14))
122. lab5=tk.Label(insertk,text="Movie Director:",bg='#42c8f5',font=('Cascadia Mono SemiLight',14))
123. lab6=tk.Label(insertk,text="IMDB Rating:",bg='#42c8f5',font=('Cascadia Mono SemiLight',14))
125. *#Placing labels*
126. lab0.pack()
127. lab1.place(x=10,y=60)
128. lab2.place(x=10,y=100)
129. lab3.place(x=10,y=140)
130. lab4.place(x=10,y=180)
131. lab5.place(x=10,y=220)
132. lab6.place(x=10,y=260)
133. *#Initializing variables to read the entry box data*
134. nm=StringVar()
135. genre=StringVar()
136. dor=StringVar()
137. code=StringVar()
138. dr=StringVar()
139. rt=StringVar()
140. *#Creating entry boxes*
141. en1=tk.Entry(insertk,textvariable=nm,font=('Cascadia Mono SemiLight',14))
142. en2=tk.Entry(insertk,textvariable=genre,font=('Cascadia Mono SemiLight',14))
143. en3=tk.Entry(insertk,textvariable=dor,font=('Cascadia Mono SemiLight',14))
144. en4=tk.Entry(insertk,textvariable=code,font=('Cascadia Mono SemiLight',14))
145. en5=tk.Entry(insertk,textvariable=dr,font=('Cascadia Mono SemiLight',14))
146. en6=tk.Entry(insertk,textvariable=rt,font=('Cascadia Mono SemiLight',14))
147. en7=tk.Entry(insertk)
149. *#Placing entry boxes*
150. en1.place(x=225,y=62)
151. en2.place(x=225,y=102)
152. en3.place(x=575,y=142)
153. en4.place(x=105,y=182)
154. en5.place(x=180,y=222)
155. en6.place(x=145,y=262)
157. *#Function to execute query for entering data in MySQL table*
158. **def** **insertin**(event=None):
160. *#Getting information from entry boxes*
161. Name=nm.get()
162. Genre=genre.get()
163. DOR=dor.get()
164. Movie\_code=code.get()
165. Director=dr.get()
166. Rating=rt.get()
167. *#Connecting to MySQL and executing the query*
168. mydb = sq.connect(host="localhost",user="root",password="root",database="movie\_database")
169. mycursor = mydb.cursor()
170. sql = "INSERT INTO Movies (Movie\_Name, Genre, Date\_of\_release, IMDB\_id, Director, Rating) VALUES (%s,%s,%s,%s,%s,%s)"
171. val = (Name,Genre,DOR,Movie\_code,Director,Rating)
172. mycursor.execute(sql, val)
173. mydb.commit()
175. *#Displaying message for successful insert*
176. added=tk.Label(insertk,text='Record Inserted',font=('Cascadia Mono SemiLight',20),bg='#42c8f5')
177. added.place(x=300,y=295)
178. *#Setting all entry boxes to blank so that new data can be entered*
179. nm.set('')
180. genre.set('')
181. dor.set('')
182. code.set('')
183. dr.set('')
184. rt.set('')
186. insertk.bind\_all('<Return>', insertin)
188. *#Function to ask confirmation from user for quiting*
189. **def** **on\_closing**():
190. if messagebox.askyesno(title='QUIT?',message='Are you sure you want to quit'):
191. insertk.destroy() *#Closing this window*
192. recreate\_root() *#Re-opening root window*
193. else:
194. pass
196. insertk.protocol('WM\_DELETE\_WINDOW',on\_closing)
197. insertk.mainloop()
198. *#Function for updating a record*
199. **def** **update**(event=None):
200. *#Closing menu window*
201. root.destroy()
202. *#Creating window to update a record*
203. updatetk=tk.Tk()
204. updatetk.geometry('1000x100')
205. updatetk.configure(bg='#42c8f5')
206. updatetk.title('Update Record')
207. *#Asking user if they want to use the movie name or IMDB ID to update the record*
208. *#Creating labels*
209. lab0=tk.Label(updatetk,text='Update Record',font=('Cascadia Mono SemiLight',18,'bold'),bg='#42c8f5')
210. lab1=tk.Label(updatetk,text='Enter 1 to use movie name or 2 to use IMDB ID to update the data:',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
211. *#Placing labels*
212. lab0.pack()
213. lab1.place(x=10,y=50)
214. *#Initializing variable to read the entry box data*
215. val=StringVar()
216. *#Creating and placing entry box*
217. en1=tk.Entry(updatetk,textvariable=val,font=('Cascadia Mono SemiLight',14))
218. en1.place(x=740,y=50)
219. *#Function to ask for confirmation and close the window*
220. **def** **on\_closing**():
221. if messagebox.askyesno(title='QUIT?',message='Are you sure you want to quit'):
222. nonlocal updatetk
223. updatetk.destroy() *#Closing this window*
224. recreate\_root() *#Re-opening root window*
225. else:
226. pass
227. updatetk.protocol('WM\_DELETE\_WINDOW',on\_closing)
228. *#Checking whether user wants to update using movie name or IMDB ID*
229. **def** **updateit**(event=None):
230. char=val.get()
231. while char!='':
232. if char=='1':
233. updatewithname()
234. break
235. elif char=='2':
236. updatewithid()
237. break
238. *#Function to update using movie name*
239. **def** **updatewithname**():
240. *#Closing choice window*
241. nonlocal updatetk
242. updatetk.destroy()
243. *#Creating window to update using movie name*
244. nametk=tk.Tk()
245. nametk.geometry('750x380')
246. nametk.configure(bg='#42c8f5')
247. nametk.title('Updating Record using name')
249. *#Creating labels*
250. lab0=tk.Label(nametk,text='Update Record',font=('Cascadia Mono SemiLight',18,'bold'),bg='#42c8f5')
251. lab1=tk.Label(nametk,text='--> 1. Movie name ',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
252. lab2=tk.Label(nametk,text='--> 2. Genre',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
253. lab3=tk.Label(nametk,text='--> 3. Date of Release',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
254. lab4=tk.Label(nametk,text='--> 4. IMDB Id',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
255. lab5=tk.Label(nametk,text='--> 5. Director',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
256. lab9=tk.Label(nametk,text='--> 6. IMDB Rating',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
257. lab6=tk.Label(nametk,text="Movie name whose record you want to update:",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
258. lab7=tk.Label(nametk,text="Record you want to update:",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
259. lab8=tk.Label(nametk,text="Enter the change:",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
260. *#Placing labels*
261. lab0.pack()
262. lab6.place(x=10,y=50)
263. lab1.place(x=10,y=80)
264. lab2.place(x=10,y=110)
265. lab3.place(x=10,y=140)
266. lab4.place(x=10,y=170)
267. lab5.place(x=10,y=200)
268. lab9.place(x=10,y=230)
269. lab7.place(x=10,y=260)
270. lab8.place(x=10,y=290)
271. *#Initialising variables to read entry box*
272. upe=StringVar()
273. fi=StringVar()
274. fich=StringVar()
275. *#Creating entry boxes*
276. en1=tk.Entry(nametk,textvariable=upe,font=('Cascadia Mono SemiLight',14))
277. en2=tk.Entry(nametk,textvariable=fi,font=('Cascadia Mono SemiLight',14))
278. en3=tk.Entry(nametk,textvariable=fich,font=('Cascadia Mono SemiLight',14))
279. *#Placing entry boxes*
280. en1.place(x=490,y=52)
281. en2.place(x=300,y=262)
282. en3.place(x=200,y=294)
284. *#Function to execute query for updating record using movie name*
285. **def** **finallyupdating**(event=None):
287. *#Getting information from entry boxes*
288. up=upe.get()
289. fields=**int**(fi.get())
290. fieldch=fich.get()
291. *#Checking for which data is to be updated*
292. if fields==1:
293. field='Movie\_Name'
294. elif fields==2:
295. field='Genre'
296. elif fields==3:
297. field='Date\_of\_release'
298. elif fields==4:
299. field='IMDB\_id'
300. elif fields==5:
301. field='Director'
302. elif fields==6:
303. field='Rating'
304. *#Connecting to MySQL and executing the query*
305. mydb= sq.connect(host="localhost",user="root",passwd="root",database="Movie\_database")
306. cursor=mydb.cursor()
307. update="UPDATE Movies set {} = '{}' WHERE Movie\_Name like '{}'".format(field,fieldch,up)
308. cursor.execute(update)
309. mydb.commit()
310. *#Displaying message for successful update*
311. lab0=tk.Label(nametk,text="Record Updated!",font=('Cascadia Mono SemiLight',24),bg='#42c8f5')
312. lab0.place(x=240,y=325)
313. *#Setting all entry boxes to blank so that more updates can be done*
314. upe.set('')
315. fi.set('')
316. fich.set('')
317. *#Function to ask confirmation from user for quiting*
318. **def** **on\_closingname**():
319. if messagebox.askyesno(title='QUIT?',message='Are you sure you want to quit'):
320. nonlocal nametk
321. nametk.destroy() *#Closing this window*
322. recreate\_root() *# Re-opening root window*
323. else:
324. pass
325. nametk.bind\_all('<Return>', finallyupdating)
326. nametk.protocol('WM\_DELETE\_WINDOW',on\_closingname)
327. *#Function to update using IMDB ID*
328. **def** **updatewithid**():
329. *#Closing choice window*
330. nonlocal updatetk
331. updatetk.destroy()
332. *#Creating window to update using IMDB ID*
333. idtk=tk.Tk()
334. idtk.geometry('850x380')
335. idtk.configure(bg='#42c8f5')
336. idtk.title('Updating Record using IMDB ID')
338. *#Creating labels*
339. lab0=tk.Label(idtk,text='Update Record',font=('Cascadia Mono SemiLight',18,'bold'),bg='#42c8f5')
340. lab1=tk.Label(idtk,text='--> 1. Movie name ',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
341. lab2=tk.Label(idtk,text='--> 2. Genre',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
342. lab3=tk.Label(idtk,text='--> 3. Date of Release',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
343. lab4=tk.Label(idtk,text='--> 4. IMDB Id',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
344. lab5=tk.Label(idtk,text='--> 5. Director',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
345. lab9=tk.Label(idtk,text='--> 6. IMDB Rating',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
346. lab6=tk.Label(idtk,text="IMDB ID of the movie whose record you want to update:",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
347. lab7=tk.Label(idtk,text="Record you want to update:",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
348. lab8=tk.Label(idtk,text="Enter the change:",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
349. *#Placing labels*
350. lab0.pack()
351. lab6.place(x=10,y=50)
352. lab1.place(x=10,y=80)
353. lab2.place(x=10,y=110)
354. lab3.place(x=10,y=140)
355. lab4.place(x=10,y=170)
356. lab5.place(x=10,y=200)
357. lab9.place(x=10,y=230)
358. lab7.place(x=10,y=260)
359. lab8.place(x=10,y=290)
360. *#Initialising variables to read entry box*
361. upe=StringVar()
362. fi=StringVar()
363. fich=StringVar()
364. *#Creating entry boxes*
365. en1=tk.Entry(idtk,textvariable=upe,font=('Cascadia Mono SemiLight',14))
366. en2=tk.Entry(idtk,textvariable=fi,font=('Cascadia Mono SemiLight',14))
367. en3=tk.Entry(idtk,textvariable=fich,font=('Cascadia Mono SemiLight',14))
368. *#Placing entry boxes*
369. en1.place(x=600,y=52)
370. en2.place(x=300,y=262)
371. en3.place(x=200,y=294)
373. *#Function to execute query for updating record using IMDB ID*
374. **def** **finallyupdating**(event=None):
376. *#Getting information from entry boxes*
377. up=upe.get()
378. fields=**int**(fi.get())
379. fieldch=fich.get()
380. *#Checking for which data is to be updated*
381. if fields==1:
382. field='Movie\_Name'
383. elif fields==2:
384. field='Genre'
385. elif fields==3:
386. field='Date\_of\_release'
387. elif fields==4:
388. field='IMDB\_id'
389. elif fields==5:
390. field='Director'
391. elif fields==6:
392. field='Rating'
393. *#Connecting to MySQL and executing the query*
394. mydb= sq.connect(host="localhost",user="root",passwd="root",database="Movie\_database")
395. cursor=mydb.cursor()
396. update="UPDATE Movies set {} = '{}' WHERE IMDB\_ID= '{}'".format(field,fieldch,up)
397. cursor.execute(update)
398. mydb.commit()
399. *#Displaying message for successful update*
400. lab0=tk.Label(idtk,text="Record Updated!",font=('Cascadia Mono SemiLight',24),bg='#42c8f5')
401. lab0.place(x=240,y=325)
402. *#Setting all entry boxes to blank so that more updates can be made*
403. upe.set('')
404. fi.set('')
405. fich.set('')
406. *#Function to ask confirmation from user for quiting*
407. **def** **on\_closingid**():
408. if messagebox.askyesno(title='QUIT?',message='Are you sure you want to quit'):
409. nonlocal idtk
410. idtk.destroy() *#Closing this window*
411. recreate\_root() *#Re-opening root window*
412. else:
413. pass
414. idtk.bind\_all('<Return>', finallyupdating)
415. idtk.protocol('WM\_DELETE\_WINDOW',on\_closingid)
417. updatetk.bind\_all('<Return>', updateit)
418. updateit()
419. updatetk.protocol('WM\_DELETE\_WINDOW',on\_closing)
420. updatetk.mainloop()
421. *#Function for deleting a record*
422. **def** **delete**():
423. *#Closing menu window*
424. root.destroy()
425. *#Creating window to delete a record*
426. deletetk=tk.Tk()
427. deletetk.geometry('1000x100')
428. deletetk.configure(bg='#42c8f5')
429. deletetk.title('Delete Record')
430. *#Asking user if they want to use the movie name or IMDB ID to delete the record*
431. *#Creating labels*
432. lab0=tk.Label(deletetk,text='Delete Record',font=('Cascadia Mono SemiLight',18,'bold'),bg='#42c8f5')
433. lab1=tk.Label(deletetk,text='Enter 1 to use movie name or 2 to use IMDB ID to delete the data:',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
434. *#Placing labels*
435. lab0.pack()
436. lab1.place(x=10,y=50)
437. *#Initializing variable to read the entry box data*
438. val=StringVar()
439. *#Creating and placing entry box*
440. en1=tk.Entry(deletetk,textvariable=val,font=('Cascadia Mono SemiLight',14))
441. en1.place(x=730,y=53)
442. *#Function to ask for confirmation and close the window*
443. **def** **on\_closing**():
444. if messagebox.askyesno(title='QUIT?',message='Are you sure you want to quit'):
445. nonlocal deletetk
446. deletetk.destroy() *#Closing this window*
447. recreate\_root() *#Re-opening root window*
448. else:
449. pass
450. deletetk.protocol('WM\_DELETE\_WINDOW',on\_closing)
451. *#Checking whether user wants to delete using movie name or IMDB ID*
452. **def** **deleteit**(event=None):
453. char=val.get()
454. while char!='':
455. if char=='1':
456. deletewithname()
457. break
458. elif char=='2':
459. deletewithid()
460. break
461. *#Function to delete using movie name*
462. **def** **deletewithname**():
463. *#Closing choice window*
464. nonlocal deletetk
465. deletetk.destroy()
466. *#Creating window to delete using movie name*
467. nametk=tk.Tk()
468. nametk.geometry('1060x130')
469. nametk.configure(bg='#42c8f5')
470. nametk.title('Deleting Record using name')
472. *#Creating labels*
473. lab0=tk.Label(nametk,text='Delete Record',font=('Cascadia Mono SemiLight',18,'bold'),bg='#42c8f5')
474. lab1=tk.Label(nametk,text='Enter the name of the movie whose data you want to delete: ',font=('Cascadia Mono SemiLight',18),bg='#42c8f5')
475. *#Placing labels*
476. lab0.pack()
477. lab1.place(x=10,y=50)
479. *#Initialising variable to read entry box*
480. de=StringVar()
481. *#Creating and placing entry box*
482. en1=tk.Entry(nametk,textvariable=de,font=('Cascadia Mono SemiLight',14))
483. en1.place(x=825,y=59)
485. *#Function to execute query for deleteing record using movie name*
486. **def** **finallydeleting**(event=None):
487. *#Getting information from entry box*
488. dele=de.get()
489. *#Connecting to MySQL and executing the query*
490. c=sq.connect(host="localhost",user="root",passwd="root",database="movie\_database")
491. cursor=c.cursor()
492. sql="DELETE FROM Movies WHERE Movie\_Name like '%{}%'".format(dele)
493. cursor.execute(sql)
494. c.commit()
495. *#Displaying message for successful delete*
496. lab2=tk.Label(nametk,text="Record Deleted!",font=('Cascadia Mono SemiLight',15,'bold'),bg='#42c8f5')
497. lab2.place(x=400,y=90)
499. *#Setting all entry boxes to blank so that more updates can be done*
500. de.set('')
501. *#Function to ask confirmation from user for quiting*
502. **def** **on\_closingname**():
503. if messagebox.askyesno(title='QUIT?',message='Are you sure you want to quit'):
504. nonlocal nametk
505. nametk.destroy() *#Closing this window*
506. recreate\_root() *#Re-opening root window*
507. else:
508. pass
509. nametk.bind\_all('<Return>', finallydeleting)
510. nametk.protocol('WM\_DELETE\_WINDOW',on\_closingname)
511. *#Function to delete using IMDB ID*
512. **def** **deletewithid**():
513. *#Closing choice window*
514. nonlocal deletetk
515. deletetk.destroy()
516. *#Creating window to delete using movie name*
517. idtk=tk.Tk()
518. idtk.geometry('1150x110')
519. idtk.configure(bg='#42c8f5')
520. idtk.title('Deleting Record using IMDB ID')
522. *#Creating labels*
523. lab0=tk.Label(idtk,text='Delete Record',font=('Cascadia Mono SemiLight',18,'bold'),bg='#42c8f5')
524. lab1=tk.Label(idtk,text='Enter the IMDB ID of the movie whose data you want to delete: ',font=('Cascadia Mono SemiLight',18),bg='#42c8f5')
525. *#Placing labels*
526. lab0.pack()
527. lab1.place(x=10,y=50)
529. *#Initialising variable to read entry box*
530. de=StringVar()
531. *#Creating and placing entry boxes*
532. en1=tk.Entry(idtk,textvariable=de,font=('Cascadia Mono SemiLight',14))
533. en1.place(x=870,y=59)
535. *#Function to execute query for deleteing record using IMDB ID*
536. **def** **finallydeleting**(event=None):
537. *#Getting information from entry box*
538. dele=de.get()
540. *#Connecting to MySQL and executing the query*
541. c=sq.connect(host="localhost",user="root",passwd="root",database="movie\_database")
542. cursor=c.cursor()
543. sql="DELETE FROM Movies WHERE IMDB\_id like '%{}%'".format(dele)
544. cursor.execute(sql)
545. c.commit()
546. *#Displaying message for successful delete*
547. lab2=tk.Label(idtk,text="Record Deleted!",font=('Cascadia Mono SemiLight',15,'bold'),bg='#42c8f5')
548. lab2.place(x=320,y=80)
550. *#Setting all entry boxes to blank so that more updates can be done*
551. de.set('')
552. *#Function to ask confirmation from user for quiting*
553. **def** **on\_closingid**():
554. if messagebox.askyesno(title='QUIT?',message='Are you sure you want to quit'):
555. nonlocal idtk
556. idtk.destroy() *#Closing this window*
557. recreate\_root() *#Re-opening root window*
558. else:
559. pass
560. idtk.bind\_all('<Return>', finallydeleting)
561. idtk.protocol('WM\_DELETE\_WINDOW',on\_closingid)
563. deletetk.bind\_all('<Return>',deleteit)
564. deleteit()
565. deletetk.mainloop()
567. *#Function for searching a record*
568. **def** **search**():
569. *#Closing menu window*
570. root.destroy()
571. *#Creating window to search a record*
572. searchtk=tk.Tk()
573. searchtk.geometry('1000x100')
574. searchtk.configure(bg='#42c8f5')
575. searchtk.title('Search Record')
577. *#Asking user if they want to use the movie name or IMDB ID to search the record*
578. *#Creating labels*
579. lab0=tk.Label(searchtk,text='Search Record',font=('Cascadia Mono SemiLight',18,'bold'),bg='#42c8f5')
580. lab1=tk.Label(searchtk,text='Enter 1 to use movie name or 2 to use IMDB ID to search the data:',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
581. *#Placing labels*
582. lab0.pack()
583. lab1.place(x=10,y=50)
584. *#Initializing variable to read the entry box data*
585. val=StringVar()
586. *#Creating and placing entry box*
587. en1=tk.Entry(searchtk,textvariable=val,font=('Cascadia Mono SemiLight',14))
588. en1.place(x=730,y=53)
589. *#Function to ask for confirmation and close the window*
590. **def** **on\_closing**():
591. if messagebox.askyesno(title='QUIT?',message='Are you sure you want to quit'):
592. nonlocal searchtk
593. searchtk.destroy() *#Closing this window*
594. recreate\_root() *#Re-opening root window*
595. else:
596. pass
597. searchtk.protocol('WM\_DELETE\_WINDOW',on\_closing)
598. *#Checking whether user wants to search using movie name or IMDB ID*
599. **def** **searchit**(event=None):
600. char=val.get()
601. while char!='':
602. if char=='1':
603. searchwithname()
604. break
605. elif char=='2':
606. searchwithid()
607. break
608. *#Function to search using movie name*
609. **def** **searchwithname**():
610. *#Closing choice window*
611. nonlocal searchtk
612. searchtk.destroy()
613. *#Creating window to search using movie name*
614. nametk = tk.Tk()
615. nametk.geometry('950x100')
616. nametk.configure(bg='#42c8f5')
617. nametk.title('Searching Record using name')
618. *#Creating labels*
619. lab0 = tk.Label(nametk, text='Search Record', font=('Cascadia Mono SemiLight', 18, 'bold'), bg='#42c8f5')
620. lab1 = tk.Label(nametk, text='Enter the name of the movie whose data you want to search: ', font=('Cascadia Mono SemiLight', 14), bg='#42c8f5')
621. *#Placing labels*
622. lab0.pack()
623. lab1.place(x=10,y=50)
624. *#Initializing variable to read entry box*
625. search = StringVar()
626. *#Creating and placing entry box*
627. en1 = tk.Entry(nametk, textvariable=search, font=('Cascadia Mono SemiLight', 14))
628. en1.place(x=650,y=53)
629. *#Function to execute query for searching record using movie name and displaying it*
630. **def** **finallysearching**(event=None):
631. try:
632. *#Getting information from entry box*
633. name = search.get()
634. *#Connecting to MySQL and executing the query*
635. mydb = sq.connect(host='localhost', user='root', password='root', database='movie\_database')
636. cursor = mydb.cursor()
637. cursor.execute("SELECT \* FROM Movies WHERE Movie\_Name LIKE '%{}%'".format(name))
638. *#Reading the output provied by MySQL*
639. result = cursor.fetchall()
640. *#Displaying output according to the result obtained from the query*
641. *#If record exists*
642. if result:
644. *#New window to display data*
645. display\_window = tk.Tk()
646. display\_window.geometry('600x100')
647. display\_window.configure(bg='#42c8f5')
648. display\_window.title('Search Results for Movie Name')
649. *#Creating header labels for the table*
650. header\_labels = ['Movie Name', 'IMDB ID', 'Director', 'Year', 'Genre']
651. for i, header in **enumerate**(header\_labels):
652. tk.Label(display\_window, text=header, font=('Cascadia Mono SemiLight', 16), bg='#42c8f5').grid(row=0, column=i, padx=10, pady=10)
653. *#Adding movie records in rows*
654. for i, record in **enumerate**(result):
655. for j, value in **enumerate**(record[1:]):
656. tk.Label(display\_window, text=value, font=('Cascadia Mono SemiLight', 14), bg='#42c8f5').grid(row=i + 1, column=j, padx=10, pady=5)
657. display\_window.mainloop()
658. *#If record does not exist*
659. else:
660. messagebox.showinfo("No Results", "No records found for the given movie name.")
661. except **Exception** as e:
662. **print**(**f**"Error: {e}")
663. *#Function to ask confirmation from user for quiting*
664. **def** **on\_closingname**():
665. if messagebox.askyesno(title='QUIT?',message='Are you sure you want to quit'):
666. nonlocal nametk
667. nametk.destroy() *#Closing this window*
668. recreate\_root() *#Re-opening root window*
669. else:
670. pass
671. nametk.bind\_all('<Return>', finallysearching)
672. nametk.protocol('WM\_DELETE\_WINDOW', on\_closingname)
673. *#Function to search using movie name*
674. **def** **searchwithid**():
675. *#Closing choice window*
676. nonlocal searchtk
677. searchtk.destroy()
678. *#Creating window to search using IMDB ID*
679. idtk = tk.Tk()
680. idtk.geometry('950x100')
681. idtk.configure(bg='#42c8f5')
682. idtk.title('Searching Record using IMDB ID')
683. *#Creating labels*
684. lab0 = tk.Label(idtk, text='Search Record', font=('Cascadia Mono SemiLight', 18, 'bold'), bg='#42c8f5')
685. lab1 = tk.Label(idtk, text='Enter the IMDB ID of the movie whose data you want to search: ', font=('Cascadia Mono SemiLight', 14), bg='#42c8f5')
686. *#Placing labels*
687. lab0.pack()
688. lab1.place(x=10,y=50)
689. *#Initializing variable to read entry box*
690. search = StringVar()
691. *#Creating and placing entry box*
692. en1 = tk.Entry(idtk, textvariable=search, font=('Cascadia Mono SemiLight', 14))
693. en1.place(x=690,y=53)
694. *#Function to execute query for searching record using movie name and displaying it*
695. **def** **finallysearching**(event=None):
696. try:
697. *#Getting information from entry box*
698. idd = search.get()
699. *#Connecting to MySQL and executing the query*
700. mydb = sq.connect(host='localhost', user='root', password='root', database='movie\_database')
701. cursor = mydb.cursor()
702. cursor.execute("SELECT \* FROM Movies WHERE IMDB\_id LIKE '%{}%'".format(idd))
703. *#Reading the output provied by MySQL*
704. result = cursor.fetchall()
705. *#Displaying output according to the result obtained from the query*
706. *#If record exists*
707. if result:
709. *#New window to display data*
710. display\_window = tk.Tk()
711. display\_window.geometry('600x100')
712. display\_window.configure(bg='#42c8f5')
713. display\_window.title('Search Results for Movie Name')
714. *#Creating header labels for the table*
715. header\_labels = ['Movie Name', 'IMDB ID', 'Director', 'Year', 'Genre']
716. for i, header in **enumerate**(header\_labels):
717. tk.Label(display\_window, text=header, font=('Cascadia Mono SemiLight', 16), bg='#42c8f5').grid(row=0, column=i, padx=10, pady=10)
718. *#Adding movie records in rows*
719. for i, record in **enumerate**(result):
720. for j, value in **enumerate**(record[1:]):
721. tk.Label(display\_window, text=value, font=('Cascadia Mono SemiLight', 14), bg='#42c8f5').grid(row=i + 1, column=j, padx=10, pady=5)
722. display\_window.mainloop()
723. else:
724. messagebox.showinfo("No Results", "No records found for the given IMDB ID.")
725. except **Exception** as e:
726. **print**(**f**"Error: {e}")
727. *#Function to ask confirmation from user for quiting*
728. **def** **on\_closingid**():
729. if messagebox.askyesno(title='QUIT?',message='Are you sure you want to quit'):
730. nonlocal idtk
731. idtk.destroy() *#Closing this window*
732. recreate\_root() *#Re-opening root window*
733. else:
734. pass
735. idtk.bind\_all('<Return>', finallysearching)
736. idtk.protocol('WM\_DELETE\_WINDOW', on\_closingid)
737. searchtk.bind\_all('<Return>',searchit)
738. *#Function for displaying the data*
739. **def** **display**():
740. *#Closing menu window*
741. root.destroy()
742. *#Creating window to search a record*
743. displaytk=tk.Tk()
744. displaytk.configure(bg='#42c8f5')
745. displaytk.geometry('1000x550')
746. displaytk.title('Display Record')
747. *#Connecting to MySQL and executing the query*
748. mydb=sq.connect(host="localhost", user="root", password="root", database="movie\_database")
749. cursor=mydb.cursor()
750. sql="SELECT \* FROM Movies ORDER BY Movie\_Name"
751. cursor.execute(sql)
752. myresult=cursor.fetchall()
753. *#Creating a frame*
754. container=tk.Frame(displaytk, bg='#42c8f5')
755. container.pack(fill='both', expand=True)
756. *#Creating the canvas and the scrollbar*
757. canvas = tk.Canvas(container, bg='#42c8f5')
758. scrollbar = tk.Scrollbar(container, orient='vertical', command=canvas.yview)
759. scrollable\_frame = tk.Frame(canvas, bg='#42c8f5')
760. *#Configuring the scrollable frame*
761. scrollable\_frame.bind("<Configure>",**lambda** e: canvas.configure(scrollregion=canvas.bbox("all")))
762. canvas.create\_window((0, 0), window=scrollable\_frame, anchor="nw")
763. canvas.configure(yscrollcommand=scrollbar.set)
764. *#Adding the header labels*
765. header\_labels = ['Movie Name', 'Genre', 'Date of Release', 'IMDB Id', 'Director', 'Rating']
766. for i, header in **enumerate**(header\_labels):
767. tk.Label(scrollable\_frame, text=header, font=('Cascadia Mono SemiLight', 16), bg='#42c8f5').grid(row=0, column=i, padx=10, pady=10)
768. *#Adding movie records in rows*
769. for i, record in **enumerate**(myresult):
770. for j, value in **enumerate**(record):
771. tk.Label(scrollable\_frame, text=value, font=('Cascadia Mono SemiLight', 14), bg='#42c8f5').grid(row=i+1, column=j, padx=10, pady=5)
772. *#Placing the canvas and the scrollbar*
773. canvas.pack(side="left", fill="both", expand=True)
774. scrollbar.pack(side="right", fill="y")
775. *#Function to ask confirmation from user for quitting*
776. **def** **on\_closing**():
777. if messagebox.askyesno(title='QUIT?', message='Are you sure you want to quit'):
778. displaytk.destroy()  *#Closing this window*
779. recreate\_root()  *#Re-opening root window*
780. else:
781. pass
782. displaytk.protocol('WM\_DELETE\_WINDOW', on\_closing)
783. displaytk.mainloop()
784. *#Root window*
785. root=tk.Tk()
786. root.geometry('690x350')
787. root.configure(bg='#42c8f5')
788. root.title('Menu')
789. lab3=tk.Label(root, text="WELCOME TO BLOCKBUSTER: A Movie Database",font=('Cascadia Mono SemiLight',16,'bold'),bg='#42c8f5')
790. lab6=tk.Label(root, text="Made By: Atharava Srivastava",font=('Cascadia Mono SemiLight',16,'bold'),bg='#42c8f5')
791. lab3.pack()
792. lab6.pack()
793. lin1=tk.Label(root,text="1.Insert new data ",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
794. lin2=tk.Label(root,text="2.Update the table",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
795. lin3=tk.Label(root,text="3.Delete the record from the table",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
796. lin4=tk.Label(root,text="4.Search a record from the table",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
797. lin5=tk.Label(root,text="5.Display the table",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
798. lin6=tk.Label(root,text="6.Quit",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
799. lin1.place(x=10,y=80)
800. lin2.place(x=10,y=110)
801. lin3.place(x=10,y=140)
802. lin4.place(x=10,y=170)
803. lin5.place(x=10,y=200)
804. lin6.place(x=10,y=230)
805. ch=StringVar()
807. lab1=tk.Label(root,text="Which function do you want to apply?:",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
808. lab1.place(x=10,y=260)
809. en1=tk.Entry(root, textvariable=ch, font=('Cascadia Mono SemiLight',14))
810. en1.place(x=420,y=263)
811. *#Function to ask confirmation from user for quiting*
812. **def** **on\_closing**():
813. if messagebox.askyesno(title='QUIT?',message='Are you sure you want to quit'):
814. root.destroy() *#Closing root window*
815. else:
816. pass
817. *#Function to accept the choice user from menu items*
818. **def** **choicefunc**(event=None):
819. choice=ch.get()
820. *#To insert new data*
821. if choice=='1':
822. insert()
823. *#To update a record*
824. elif choice=='2':
825. update()
826. *#To delete a record*
827. elif choice=='3':
828. delete()
829. *#To search a record*
830. elif choice=='4':
831. search()
832. *#To display the data*
833. elif choice=='5':
834. display()
835. *#To exit the program*
836. elif choice=='6':
837. exit\_=tk.Tk()
838. exit\_.geometry('500x100')
839. exit\_.config(bg='#42c8f5')
840. exit\_.title('Exit')
841. label\_0=tk.Label(exit\_, text="Thank You!",font=('Cascadia Mono SemiLight',16,'bold'),bg='#42c8f5')
842. label\_1=tk.Label(exit\_, text="Hope you have a nice day!",font=('Cascadia Mono SemiLight',16,'bold'),bg='#42c8f5')
843. label\_0.pack()
844. label\_1.pack()
845. root.destroy()
846. *#Invaild input*
847. else:
848. lab2=tk.Label(root,text="Please Enter Valid Input!",font=('Cascadia Mono SemiLight',15),bg='#42c8f5')
849. lab2.place(x=170,y=300)
850. ch.set('')
852. en1.bind('<Return>',choicefunc)
853. User Interface:



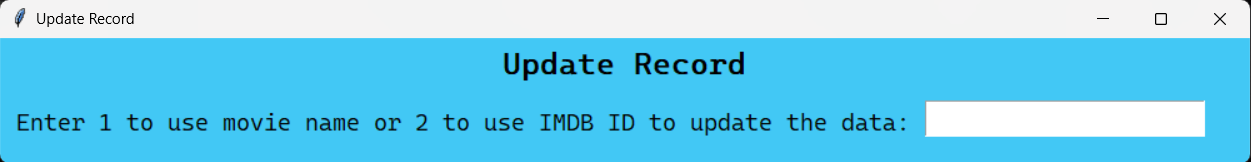
Main Menu



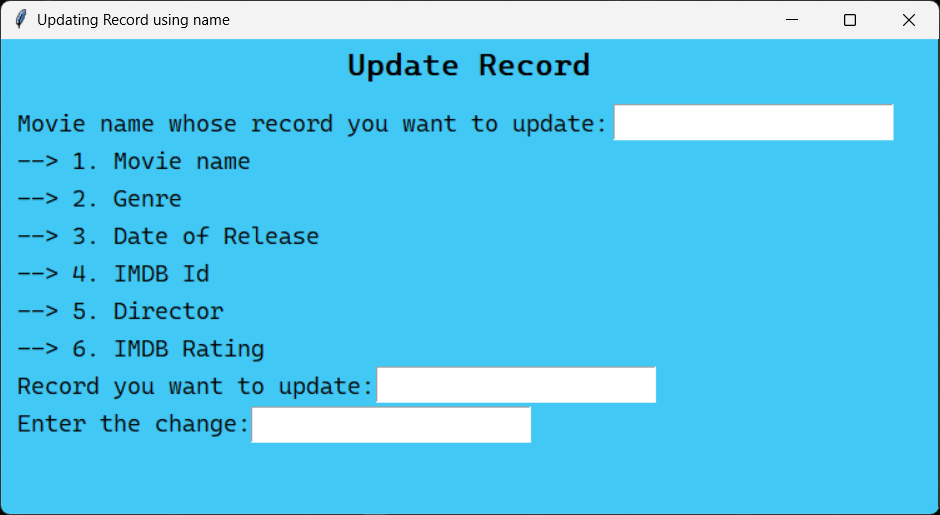
Exit Confirmation



Inserting record



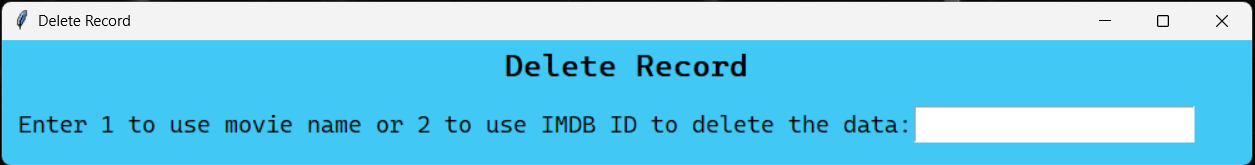
Update choice



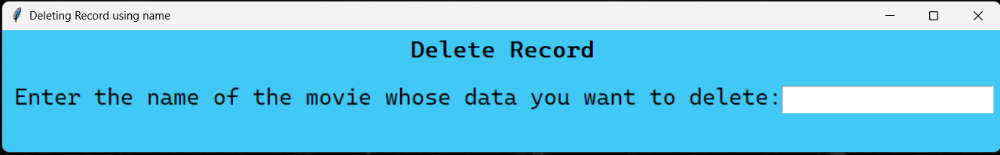
Updating with name



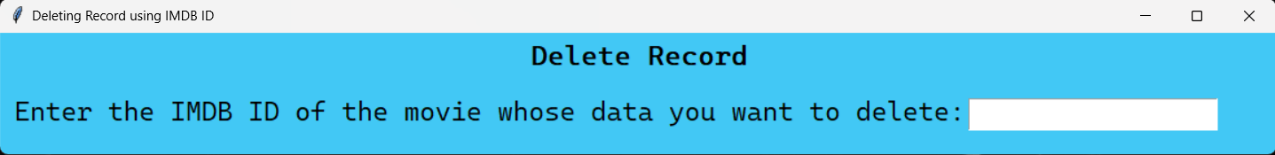
Updating with IMDB ID



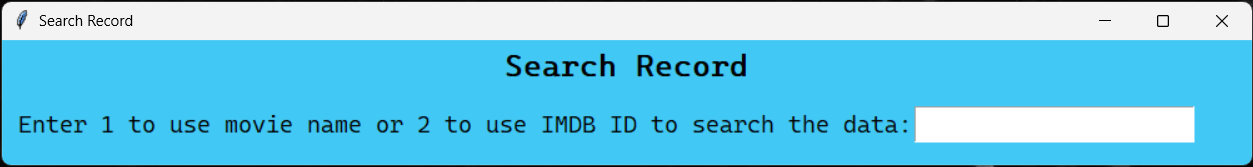
Delete choice



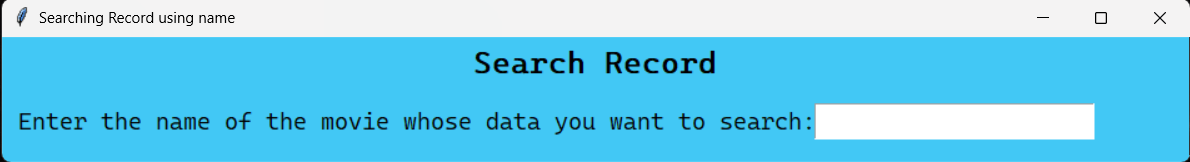
Deleting with name



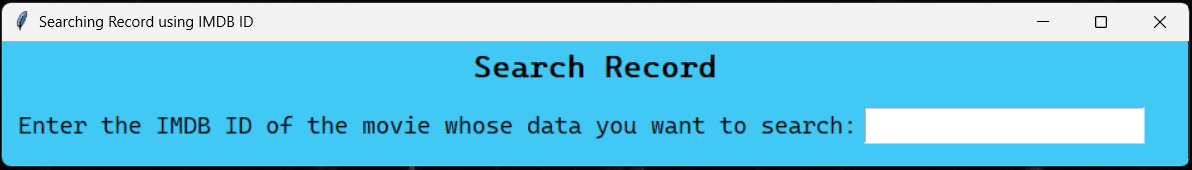
Deleting with IMDB ID



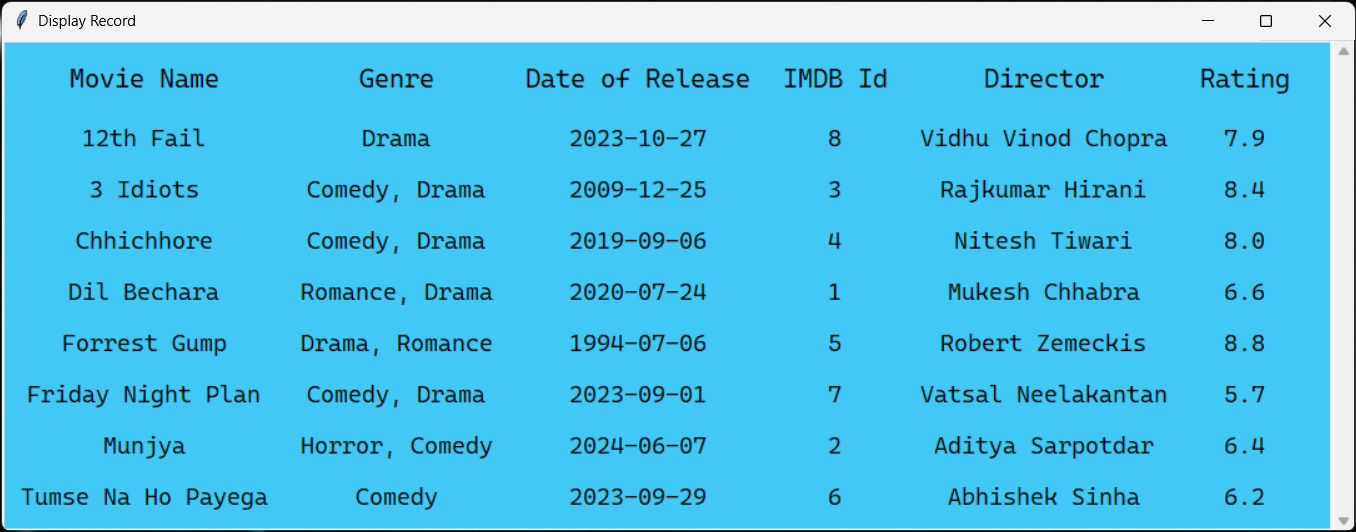
Search choice

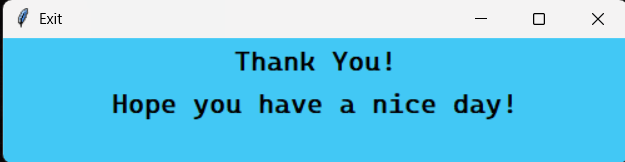


Searching with name

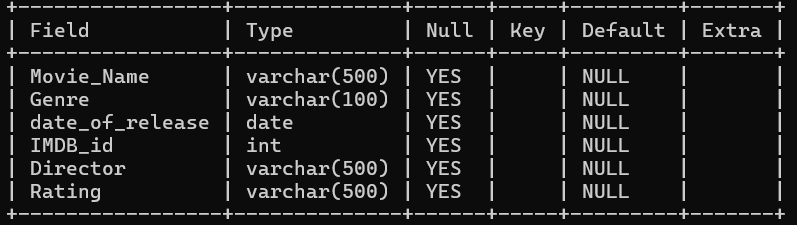


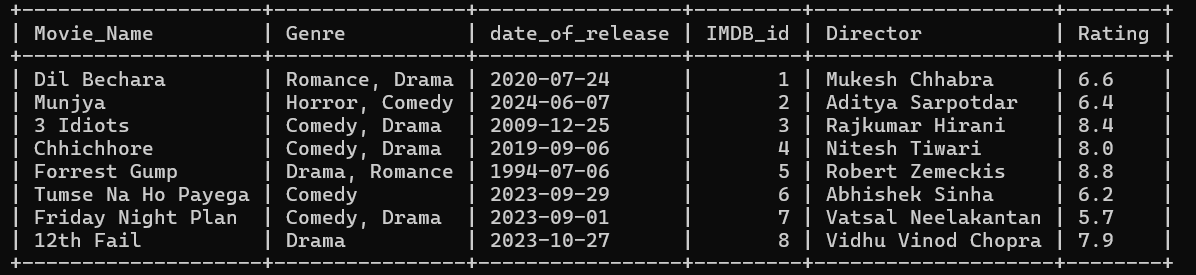
Searching with IMDB ID



Display

Exit

1. MySQL Table:

Table Properties

Dummy Database

Recommendations

1. More features can be added to this project like:
   * 1. Ability to add whether the movie has been watched yet or not
     2. Ability to add personal reviews for the movie
2. The UI can be improved by using advanced functions of Tkinter.
3. The code can be optimized further to reduce its time complexity.

Conclusion

This applet can be used for personal use to store information about movies locally on computer systems.

Creating this project helped me explore the world of UI and Python connectivity with MySQL.

Bibliography

1. Information about Blockbuster and logo: <https://en.wikipedia.org/wiki/Blockbuster_(retailer)>
2. The Last Blockbuster image 1:  
   <https://news.airbnb.com/store-manager-lists-worlds-last-blockbuster-on-airbnb-for-local-residents/>
3. The Last Blockbuster image 2:  
   <https://www.businessinsider.com/inside-last-blockbuster-in-the-world-photo-tour-bend-oregon#the-last-open-blockbuster-store-is-located-in-a-plaza-in-bend-oregon-1>