

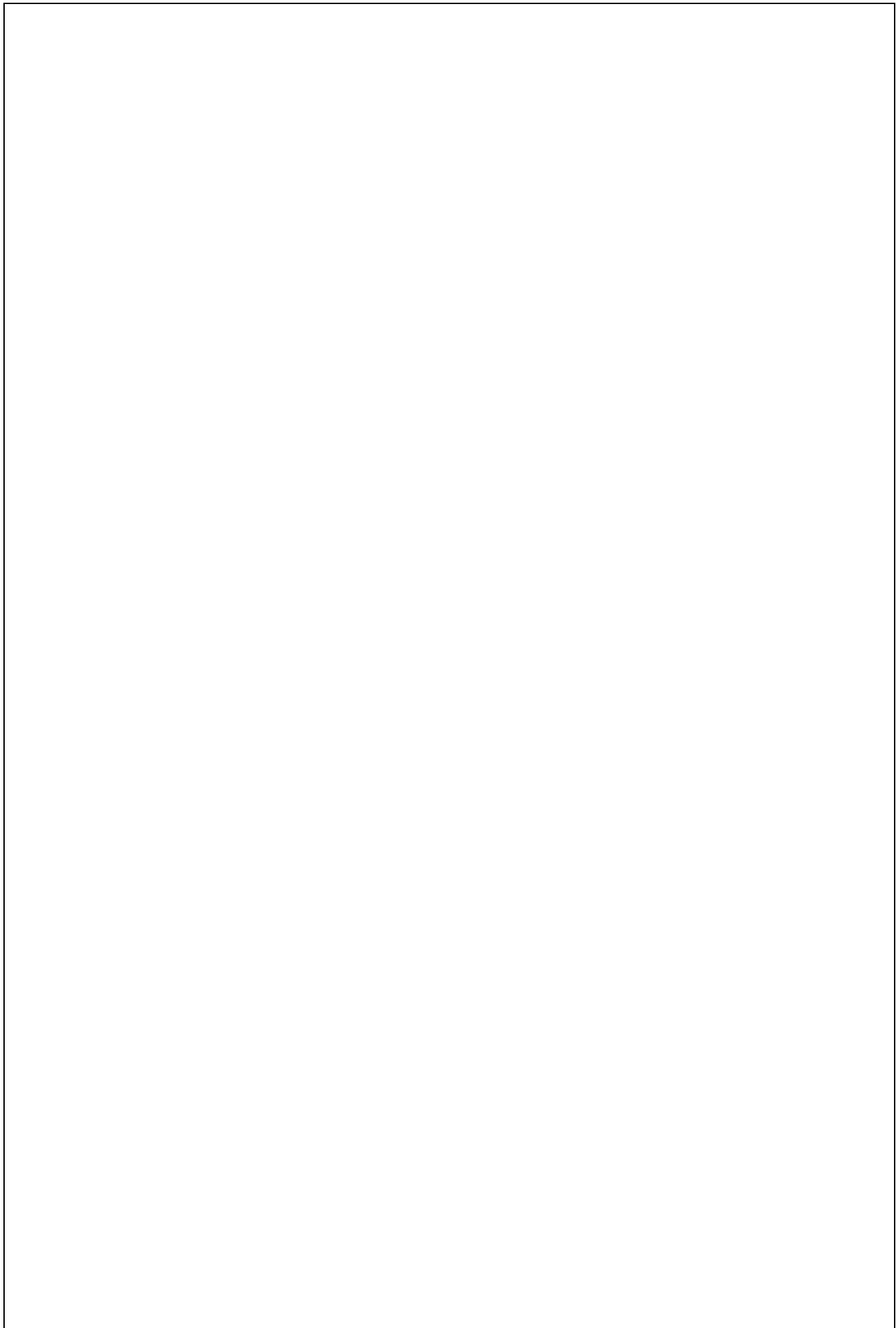
CBSE

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 my 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:

Component	Minimum	Recommended
Processor	Dual-core CPU	Quad-core CPU or better
RAM	2 GB	4 GB or higher
Storage	2 GB free space	10 GB or more
Operating System	Windows 7/Linux/macOS	Latest OS version

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

2. 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

3. Source code:

```
1. #Importing required libraries
2. import tkinter as tk
3. from tkinter import StringVar
4. from tkinter import messagebox
5. import mysql.connector as sq
6.
7.
8. #Creating the database Movie_database and a table inside it called Movies
9. def create():
10.     mydb = sq.connect(host="localhost",user="root",password="root")
11.     mycursor = mydb.cursor()
12.     sql = "Create database Movie_database"
13.     mycursor.execute(sql)
14.     mycursor.execute("Use Movie_database")
15.     mydb.commit()
16.     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))")
17.     mydb.commit()
18.
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_databas
            e")
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.
30. #Re-opening the menu window whenever the close button is clicked.
31. def recreate_root():
32.     global root
33.     root=tk.Tk()
34.     root.geometry('690x350')
35.     root.configure(bg='#42c8f5')
36.     root.title('Menu')
37.     lab3=tk.Label(root, text="WELCOME TO BLOCKBUSTER: A Movie
        Database",font=('Cascadia Mono SemiLight',16,'bold'),bg='#42c8f5')
38.     lab6=tk.Label(root, text="Made By: Atharava Srivastava",font=('Cascadia
        Mono SemiLight',16,'bold'),bg='#42c8f5')
39.     lab3.pack()
```

```

40.     lab6.pack()
41.
42.     lin1=tk.Label(root,text="1.Insert new records",font=('Casadia Mono
    SemiLight',14),bg='#42c8f5')
43.     lin2=tk.Label(root,text="2.Update a record",font=('Casadia Mono
    SemiLight',14),bg='#42c8f5')
44.     lin3=tk.Label(root,text="3.Delete a record",font=('Casadia Mono
    SemiLight',14),bg='#42c8f5')
45.     lin4=tk.Label(root,text="4.Search a record",font=('Casadia Mono
    SemiLight',14),bg='#42c8f5')
46.     lin5=tk.Label(root,text="5.Display the data",font=('Casadia Mono
    SemiLight',14),bg='#42c8f5')
47.     lin6=tk.Label(root,text="6.Quit",font=('Casadia Mono
    SemiLight',14),bg='#42c8f5')
48.
49.     lin1.place(x=10,y=80)
50.     lin2.place(x=10,y=110)
51.     lin3.place(x=10,y=140)
52.     lin4.place(x=10,y=170)
53.     lin5.place(x=10,y=200)
54.     lin6.place(x=10,y=230)
55.
56.     ch=StringVar()
57.
58.     lab1=tk.Label(root,text="Which function do you want to
    apply?:",font=('Casadia Mono SemiLight',14),bg='#42c8f5')
59.     lab1.place(x=10,y=260)
60.
61.     en1=tk.Entry(root, textvariable=ch, font=('Casadia Mono SemiLight',14))
62.     en1.place(x=420,y=263)
63.
64.     #Function to ask for confirmation from user if quitting
65.     def on_closing():
66.         if messagebox.askyesno(title='QUIT?',message='Are you sure you want
    to quit'):
67.             root.destroy()
68.         else:
69.             pass
70.
71.     #Function to accept the choice from user of menu items
72.     def choicefunc(event=None):
73.         choice=ch.get()
74.         #To insert new records
75.         if choice=='1':
76.             insert()
77.         #To update a record
78.         elif choice=='2':
79.             update()
80.         #To delete a record
81.         elif choice=='3':

```

```

82.         delete()
83.         #To search a record
84.         elif choice=='4':
85.             search()
86.         #To display the data
87.         elif choice=='5':
88.             display()
89.         #To exit the program
90.         elif choice=='6':
91.             exit_=tk.Tk()
92.             exit_.geometry('500x100')
93.             exit_.config(bg='#42c8f5')
94.             exit_.title('Exit')
95.             label_0=tk.Label(exit_, text="Thank You!",font=('Cascadia Mono
SemiLight',16,'bold'),bg='#42c8f5')
96.             label_1=tk.Label(exit_, text="Hope you have a nice
day!",font=('Cascadia Mono SemiLight',16,'bold'),bg='#42c8f5')
97.             label_0.pack()
98.             label_1.pack()
99.             root.destroy()
100.            #Invaild input
101.            else:
102.                lab2=tk.Label(root,text="Please Enter Valid
Input!",font=('Cascadia Mono SemiLight',15),bg='#42c8f5')
103.                lab2.place(x=170,y=300)
104.                ch.set('')
105.
106.            en1.bind ('<Return>',choicefunc)
107.
108.            #Recreating the option window
109.            root.mainloop()
110.
111.
112.            #Function for inserting data
113.            def insert():
114.
115.                #Closing menu window
116.                root.destroy()
117.
118.                #Creating window to insert data
119.                inserttk=tk.Tk()
120.                inserttk.geometry('850x380')
121.                inserttk.configure(bg='#42c8f5')
122.                inserttk.title('Insert Record')
123.
124.                #Creating Labels
125.                lab0=tk.Label(inserttk,text='Fill out the below
information',bg='#42c8f5',font=('Cascadia Mono SemiLight',18,'bold'))
126.                lab1=tk.Label(inserttk,text="Name of the Movie
:",bg='#42c8f5',font=('Cascadia Mono SemiLight',14))

```

```

127.         lab2=tk.Label(insertk,text="Genre of the
        Movie:",bg='#42c8f5',font=('Cascadia Mono SemiLight',14))
128.         lab3=tk.Label(insertk,text="Date of Release of the Movie (in YYYY-MM-
        DD format):",bg='#42c8f5',font=('Cascadia Mono SemiLight',14))
129.         lab4=tk.Label(insertk,text="IMDB ID:",bg='#42c8f5',font=('Cascadia
        Mono SemiLight',14))
130.         lab5=tk.Label(insertk,text="Movie
        Director:",bg='#42c8f5',font=('Cascadia Mono SemiLight',14))
131.         lab6=tk.Label(insertk,text="IMDB
        Rating:",bg='#42c8f5',font=('Cascadia Mono SemiLight',14))
132.
133.         #Placing Labels
134.         lab0.pack()
135.         lab1.place(x=10,y=60)
136.         lab2.place(x=10,y=100)
137.         lab3.place(x=10,y=140)
138.         lab4.place(x=10,y=180)
139.         lab5.place(x=10,y=220)
140.         lab6.place(x=10,y=260)
141.
142.         #Initializing variables to read the entry box data
143.         nm=StringVar()
144.         genre=StringVar()
145.         dor=StringVar()
146.         code=StringVar()
147.         dr=StringVar()
148.         rt=StringVar()
149.
150.         #Creating entry boxes
151.         en1=tk.Entry(insertk,textvariable=nm,font=('Cascadia Mono
        SemiLight',14))
152.         en2=tk.Entry(insertk,textvariable=genre,font=('Cascadia Mono
        SemiLight',14))
153.         en3=tk.Entry(insertk,textvariable=dor,font=('Cascadia Mono
        SemiLight',14))
154.         en4=tk.Entry(insertk,textvariable=code,font=('Cascadia Mono
        SemiLight',14))
155.         en5=tk.Entry(insertk,textvariable=dr,font=('Cascadia Mono
        SemiLight',14))
156.         en6=tk.Entry(insertk,textvariable=rt,font=('Cascadia Mono
        SemiLight',14))
157.         en7=tk.Entry(insertk)
158.
159.         #Placing entry boxes
160.         en1.place(x=225,y=62)
161.         en2.place(x=225,y=102)
162.         en3.place(x=575,y=142)
163.         en4.place(x=105,y=182)
164.         en5.place(x=180,y=222)
165.         en6.place(x=145,y=262)

```

```

166.
167.         #Function to execute query for entering data in MySQL table
168.         def insertin(event=None):
169.
170.             #Getting information from entry boxes
171.             Name=nm.get()
172.             Genre=genre.get()
173.             DOR=dor.get()
174.             Movie_code=code.get()
175.             Director=dr.get()
176.             Rating=rt.get()
177.
178.             #Connecting to MySQL and executing the query
179.             mydb =
sq.connect(host="localhost",user="root",password="root",database="movie_databas
e")
180.             mycursor = mydb.cursor()
181.             sql = "INSERT INTO Movies (Movie_Name, Genre, Date_of_release,
IMDB_id, Director, Rating) VALUES (%s,%s,%s,%s,%s,%s)"
182.             val = (Name,Genre,DOR,Movie_code,Director,Rating)
183.             mycursor.execute(sql, val)
184.             mydb.commit()
185.
186.             #Displaying message for successful insert
187.             added=tk.Label(inserttk,text='Record Inserted',font=('Cascadia
Mono SemiLight',20),bg='#42c8f5')
188.             added.place(x=300,y=295)
189.
190.             #Setting all entry boxes to blank so that new data can be
entered
191.             nm.set('')
192.             genre.set('')
193.             dor.set('')
194.             code.set('')
195.             dr.set('')
196.             rt.set('')
197.
198.             inserttk.bind_all('<Return>', insertin)
199.
200.             #Function to ask confirmation from user for quitting
201.             def on_closing():
202.                 if messagebox.askyesno(title='QUIT?',message='Are you sure you
want to quit'):
203.                     inserttk.destroy() #Closing this window
204.                     recreate_root() #Re-opening root window
205.                 else:
206.                     pass
207.
208.             inserttk.protocol('WM_DELETE_WINDOW',on_closing)
209.             inserttk.mainloop()

```

```

210.
211.     #Function for updating a record
212.     def update(event=None):
213.
214.         #Closing menu window
215.         root.destroy()
216.
217.         #Creating window to update a record
218.         updatetk=tk.Tk()
219.         updatetk.geometry('1000x100')
220.         updatetk.configure(bg='#42c8f5')
221.         updatetk.title('Update Record')
222.
223.         #Asking user if they want to use the movie name or IMDB ID to update
the record
224.         #Creating Labels
225.         lab0=tk.Label(updatetk,text='Update Record',font=('Cascadia Mono
SemiLight',18,'bold'),bg='#42c8f5')
226.         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')
227.
228.         #Placing Labels
229.         lab0.pack()
230.         lab1.place(x=10,y=50)
231.
232.         #Initializing variable to read the entry box data
233.         val=StringVar()
234.
235.         #Creating and placing entry box
236.         en1=tk.Entry(updatetk,textvariable=val,font=('Cascadia Mono
SemiLight',14))
237.         en1.place(x=740,y=50)
238.
239.         #Function to ask for confirmation and close the window
240.         def on_closing():
241.             if messagebox.askyesno(title='QUIT?',message='Are you sure
you want to quit'):
242.                 nonlocal updatetk
243.                 updatetk.destroy() #Closing this window
244.                 recreate_root() #Re-opening root window
245.             else:
246.                 pass
247.         updatetk.protocol('WM_DELETE_WINDOW',on_closing)
248.
249.         #Checking whether user wants to update using movie name or IMDB ID
250.         def updateit(event=None):
251.             char=val.get()
252.             while char!='':
253.                 if char=='1':
254.                     updatewithname()

```

```

255.             break
256.         elif char=='2':
257.             updatewithid()
258.             break
259.
260.     #Function to update using movie name
261.     def updatewithname():
262.
263.         #Closing choice window
264.         nonlocal updatetk
265.         updatetk.destroy()
266.
267.         #Creating window to update using movie name
268.         nametk=tk.Tk()
269.         nametk.geometry('750x380')
270.         nametk.configure(bg='#42c8f5')
271.         nametk.title('Updating Record using name')
272.
273.         #Creating Labels
274.         lab0=tk.Label(nametk,text='Update Record',font=('Cascaadia Mono
275.             SemiLight',18,'bold'),bg='#42c8f5')
276.         lab1=tk.Label(nametk,text='--> 1. Movie name ',font=('Cascaadia
277.             Mono SemiLight',14),bg='#42c8f5')
278.         lab2=tk.Label(nametk,text='--> 2. Genre',font=('Cascaadia Mono
279.             SemiLight',14),bg='#42c8f5')
280.         lab3=tk.Label(nametk,text='--> 3. Date of
281.             Release',font=('Cascaadia Mono SemiLight',14),bg='#42c8f5')
282.         lab4=tk.Label(nametk,text='--> 4. IMDB Id',font=('Cascaadia Mono
283.             SemiLight',14),bg='#42c8f5')
284.         lab5=tk.Label(nametk,text='--> 5. Director',font=('Cascaadia Mono
285.             SemiLight',14),bg='#42c8f5')
286.         lab9=tk.Label(nametk,text='--> 6. IMDB Rating',font=('Cascaadia
287.             Mono SemiLight',14),bg='#42c8f5')
288.         lab6=tk.Label(nametk,text="Movie name whose record you want to
289.             update:",font=('Cascaadia Mono SemiLight',14),bg='#42c8f5')
290.         lab7=tk.Label(nametk,text="Record you want to
291.             update:",font=('Cascaadia Mono SemiLight',14),bg='#42c8f5')
292.         lab8=tk.Label(nametk,text="Enter the change:",font=('Cascaadia
293.             Mono SemiLight',14),bg='#42c8f5')
294.
295.         #Placing Labels
296.         lab0.pack()
297.         lab6.place(x=10,y=50)
298.         lab1.place(x=10,y=80)
299.         lab2.place(x=10,y=110)
300.         lab3.place(x=10,y=140)
301.         lab4.place(x=10,y=170)
302.         lab5.place(x=10,y=200)
303.         lab9.place(x=10,y=230)
304.         lab7.place(x=10,y=260)

```



```

295.         lab8.place(x=10,y=290)
296.
297.         #Initialising variables to read entry box
298.         upe=StringVar()
299.         fi=StringVar()
300.         fich=StringVar()
301.
302.         #Creating entry boxes
303.         en1=tk.Entry(nametk,textvariable=upe,font=('Cascadia Mono
Semilight',14))
304.         en2=tk.Entry(nametk,textvariable=fi,font=('Cascadia Mono
Semilight',14))
305.         en3=tk.Entry(nametk,textvariable=fich,font=('Cascadia Mono
Semilight',14))
306.
307.         #Placing entry boxes
308.         en1.place(x=490,y=52)
309.         en2.place(x=300,y=262)
310.         en3.place(x=200,y=294)
311.
312.         #Function to execute query for updating record using movie name
313.         def finallyupdating(event=None):
314.
315.             #Getting information from entry boxes
316.             up=upe.get()
317.             fields=int(fi.get())
318.             fieldch=fich.get()
319.
320.             #Checking for which data is to be updated
321.             if fields==1:
322.                 field='Movie_Name'
323.             elif fields==2:
324.                 field='Genre'
325.             elif fields==3:
326.                 field='Date_of_release'
327.             elif fields==4:
328.                 field='IMDB_id'
329.             elif fields==5:
330.                 field='Director'
331.             elif fields==6:
332.                 field='Rating'
333.
334.             #Connecting to MySQL and executing the query
335.             mydb=
sq.connect(host="localhost",user="root",passwd="root",database="Movie_database"
)
336.             cursor=mydb.cursor()
337.             update="UPDATE Movies set {} = '{}' WHERE Movie_Name like
'{}'".format(field,fieldch,up)
338.             cursor.execute(update)

```

```

339.         mydb.commit()
340.
341.         #Displaying message for successful update
342.         lab0=tk.Label(nametk,text="Record Updated!",font=('Cascadia
Mono SemiLight',24),bg='#42c8f5')
343.         lab0.place(x=240,y=325)
344.
345.         #Setting all entry boxes to blank so that more updates can
be done
346.         upe.set('')
347.         fi.set('')
348.         fich.set('')
349.
350.         #Function to ask confirmation from user for quitting
351.         def on_closingname():
352.             if messagebox.askyesno(title='QUIT?',message='Are you sure
you want to quit'):
353.                 nonlocal nametk
354.                 nametk.destroy() #Closing this window
355.                 recreate_root() # Re-opening root window
356.             else:
357.                 pass
358.         nametk.bind_all('<Return>', finallyupdating)
359.         nametk.protocol('WM_DELETE_WINDOW',on_closingname)
360.
361.         #Function to update using IMDB ID
362.         def updatewithid():
363.
364.             #Closing choice window
365.             nonlocal updatetk
366.             updatetk.destroy()
367.
368.             #Creating window to update using IMDB ID
369.             idtk=tk.Tk()
370.             idtk.geometry('850x380')
371.             idtk.configure(bg='#42c8f5')
372.             idtk.title('Updating Record using IMDB ID')
373.
374.             #Creating Labels
375.             lab0=tk.Label(idtk,text='Update Record',font=('Cascadia Mono
SemiLight',18,'bold'),bg='#42c8f5')
376.             lab1=tk.Label(idtk,text='--> 1. Movie name ',font=('Cascadia
Mono SemiLight',14),bg='#42c8f5')
377.             lab2=tk.Label(idtk,text='--> 2. Genre',font=('Cascadia Mono
SemiLight',14),bg='#42c8f5')
378.             lab3=tk.Label(idtk,text='--> 3. Date of Release',font=('Cascadia
Mono SemiLight',14),bg='#42c8f5')
379.             lab4=tk.Label(idtk,text='--> 4. IMDB Id',font=('Cascadia Mono
SemiLight',14),bg='#42c8f5')

```

```

380.         lab5=tk.Label(idtk,text='--> 5. Director',font=('Cascadia Mono
      SemiLight',14),bg='#42c8f5')
381.         lab9=tk.Label(idtk,text='--> 6. IMDB Rating',font=('Cascadia
      Mono SemiLight',14),bg='#42c8f5')
382.         lab6=tk.Label(idtk,text="IMDB ID of the movie whose record you
      want to update:",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
383.         lab7=tk.Label(idtk,text="Record you want to
      update:",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
384.         lab8=tk.Label(idtk,text="Enter the change:",font=('Cascadia Mono
      SemiLight',14),bg='#42c8f5')
385.
386.         #Placing Labels
387.         lab0.pack()
388.         lab6.place(x=10,y=50)
389.         lab1.place(x=10,y=80)
390.         lab2.place(x=10,y=110)
391.         lab3.place(x=10,y=140)
392.         lab4.place(x=10,y=170)
393.         lab5.place(x=10,y=200)
394.         lab9.place(x=10,y=230)
395.         lab7.place(x=10,y=260)
396.         lab8.place(x=10,y=290)
397.
398.         #Initialising variables to read entry box
399.         upe=StringVar()
400.         fi=StringVar()
401.         fich=StringVar()
402.
403.         #Creating entry boxes
404.         en1=tk.Entry(idtk,textvariable=upe,font=('Cascadia Mono
      SemiLight',14))
405.         en2=tk.Entry(idtk,textvariable=fi,font=('Cascadia Mono
      SemiLight',14))
406.         en3=tk.Entry(idtk,textvariable=fich,font=('Cascadia Mono
      SemiLight',14))
407.
408.         #Placing entry boxes
409.         en1.place(x=600,y=52)
410.         en2.place(x=300,y=262)
411.         en3.place(x=200,y=294)
412.
413.         #Function to execute query for updating record using IMDB ID
414.         def finallyupdating(event=None):
415.
416.             #Getting information from entry boxes
417.             up=upe.get()
418.             fields=int(fi.get())
419.             fieldch=fich.get()
420.
421.             #Checking for which data is to be updated

```

```

422.         if fields==1:
423.             field='Movie_Name'
424.         elif fields==2:
425.             field='Genre'
426.         elif fields==3:
427.             field='Date_of_release'
428.         elif fields==4:
429.             field='IMDB_id'
430.         elif fields==5:
431.             field='Director'
432.         elif fields==6:
433.             field='Rating'
434.
435.         #Connecting to MySQL and executing the query
436.         mydb=
sq.connect(host="localhost",user="root",passwd="root",database="Movie_database"
)
437.         cursor=mydb.cursor()
438.         update="UPDATE Movies set {} = '{}' WHERE IMDB_ID=
'{}'.format(field,fieldch,up)
439.         cursor.execute(update)
440.         mydb.commit()
441.
442.         #Displaying message for successful update
443.         lab0=tk.Label(idtk,text="Record Updated!",font=('Cascadia
Mono SemiLight',24),bg='#42c8f5')
444.         lab0.place(x=240,y=325)
445.
446.         #Setting all entry boxes to blank so that more updates can
be made
447.         upe.set('')
448.         fi.set('')
449.         fich.set('')
450.
451.         #Function to ask confirmation from user for quitting
452.         def on_closingid():
453.             if messagebox.askyesno(title='QUIT?',message='Are you sure
you want to quit'):
454.                 nonlocal idtk
455.                 idtk.destroy() #Closing this window
456.                 recreate_root() #Re-opening root window
457.             else:
458.                 pass
459.             idtk.bind_all('<Return>', finallyupdating)
460.             idtk.protocol('WM_DELETE_WINDOW',on_closingid)
461.
462.         updatetk.bind_all('<Return>', updateit)
463.         updateit()
464.         updatetk.protocol('WM_DELETE_WINDOW',on_closing)
465.         updatetk.mainloop()

```

```

466.
467.     #Function for deleting a record
468.     def delete():
469.
470.         #Closing menu window
471.         root.destroy()
472.
473.         #Creating window to delete a record
474.         deletetk=tk.Tk()
475.         deletetk.geometry('1000x100')
476.         deletetk.configure(bg='#42c8f5')
477.         deletetk.title('Delete Record')
478.
479.         #Asking user if they want to use the movie name or IMDB ID to delete
         the record
480.         #Creating Labels
481.         lab0=tk.Label(deletetk,text='Delete Record',font=('Cascadia Mono
         SemiLight',18,'bold'),bg='#42c8f5')
482.         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')
483.
484.         #Placing Labels
485.         lab0.pack()
486.         lab1.place(x=10,y=50)
487.
488.         #Initializing variable to read the entry box data
489.         val=StringVar()
490.
491.         #Creating and placing entry box
492.         en1=tk.Entry(deletetk,textvariable=val,font=('Cascadia Mono
         SemiLight',14))
493.         en1.place(x=730,y=53)
494.
495.         #Function to ask for confirmation and close the window
496.         def on_closing():
497.             if messagebox.askyesno(title='QUIT?',message='Are you sure
         you want to quit'):
498.                 nonlocal deletetk
499.                 deletetk.destroy() #Closing this window
500.                 recreate_root() #Re-opening root window
501.             else:
502.                 pass
503.         deletetk.protocol('WM_DELETE_WINDOW',on_closing)
504.
505.         #Checking whether user wants to delete using movie name or IMDB ID
506.         def deleteit(event=None):
507.             char=val.get()
508.             while char!='':
509.                 if char=='1':
510.                     deletewithname()

```

```

511.             break
512.         elif char=='2':
513.             deletewithid()
514.             break
515.
516.     #Function to delete using movie name
517.     def deletewithname():
518.
519.         #Closing choice window
520.         nonlocal deletetk
521.         deletetk.destroy()
522.
523.         #Creating window to delete using movie name
524.         nametk=tk.Tk()
525.         nametk.geometry('1060x130')
526.         nametk.configure(bg='#42c8f5')
527.         nametk.title('Deleting Record using name')
528.
529.         #Creating Labels
530.         lab0=tk.Label(nametk,text='Delete Record',font=('Cascadia Mono
531.             SemiLight',18,'bold'),bg='#42c8f5')
532.         lab1=tk.Label(nametk,text='Enter the name of the movie whose
533.             data you want to delete: ',font=('Cascadia Mono SemiLight',18),bg='#42c8f5')
534.
535.         #Placing Labels
536.         lab0.pack()
537.         lab1.place(x=10,y=50)
538.
539.         #Initialising variable to read entry box
540.         de=StringVar()
541.
542.         #Creating and placing entry box
543.         en1=tk.Entry(nametk,textvariable=de,font=('Cascadia Mono
544.             SemiLight',14))
545.         en1.place(x=825,y=59)
546.
547.         #Function to execute query for deleting record using movie name
548.         def finallydeleting(event=None):
549.             #Getting information from entry box
550.             dele=de.get()
551.
552.             #Connecting to MySQL and executing the query
553.             c=sq.connect(host="localhost",user="root",passwd="root",data
554.                 base="movie_database")
555.             cursor=c.cursor()
556.             sql="DELETE FROM Movies WHERE Movie_Name like
557.                 '%{}%'".format(dele)
558.             cursor.execute(sql)
559.             c.commit()

```

```

556.             #Displaying message for successful delete
557.             lab2=tk.Label(nametk,text="Record Deleted!",font=('Cascadia
Mono SemiLight',15,'bold'),bg='#42c8f5')
558.             lab2.place(x=400,y=90)
559.
560.             #Setting all entry boxes to blank so that more updates can
be done
561.             de.set('')
562.
563.             #Function to ask confirmation from user for quitting
564.             def on_closingname():
565.                 if messagebox.askyesno(title='QUIT?',message='Are you sure
you want to quit'):
566.                     nonlocal nametk
567.                     nametk.destroy() #Closing this window
568.                     recreate_root() #Re-opening root window
569.                 else:
570.                     pass
571.             nametk.bind_all('<Return>', finallydeleting)
572.             nametk.protocol('WM_DELETE_WINDOW',on_closingname)
573.
574.             #Function to delete using IMDB ID
575.             def deletewithid():
576.
577.                 #Closing choice window
578.                 nonlocal deletetk
579.                 deletetk.destroy()
580.
581.                 #Creating window to delete using movie name
582.                 idtk=tk.Tk()
583.                 idtk.geometry('1150x110')
584.                 idtk.configure(bg='#42c8f5')
585.                 idtk.title('Deleting Record using IMDB ID')
586.
587.                 #Creating Labels
588.                 lab0=tk.Label(idtk,text='Delete Record',font=('Cascadia Mono
SemiLight',18,'bold'),bg='#42c8f5')
589.                 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')
590.
591.                 #Placing Labels
592.                 lab0.pack()
593.                 lab1.place(x=10,y=50)
594.
595.                 #Initialising variable to read entry box
596.                 de=StringVar()
597.
598.                 #Creating and placing entry boxes
599.                 en1=tk.Entry(idtk,textvariable=de,font=('Cascadia Mono
SemiLight',14))

```

```

600.         en1.place(x=870,y=59)
601.
602.         #Function to execute query for deleting record using IMDB ID
603.         def finallydeleting(event=None):
604.             #Getting information from entry box
605.             dele=de.get()
606.
607.             #Connecting to MySQL and executing the query
608.             c=sq.connect(host="localhost",user="root",passwd="root",data
base="movie_database")
609.             cursor=c.cursor()
610.             sql="DELETE FROM Movies WHERE IMDB_id like
        '%{}%'".format(dele)
611.             cursor.execute(sql)
612.             c.commit()
613.
614.             #Displaying message for successful delete
615.             lab2=tk.Label(idtk,text="Record Deleted!",font=('Cascadia
        Mono SemiLight',15,'bold'),bg='#42c8f5')
616.             lab2.place(x=320,y=80)
617.
618.             #Setting all entry boxes to blank so that more updates can
        be done
619.             de.set('')
620.
621.             #Function to ask confirmation from user for quitting
622.             def on_closingid():
623.                 if messagebox.askyesno(title='QUIT?',message='Are you sure
        you want to quit'):
624.                     nonlocal idtk
625.                     idtk.destroy() #Closing this window
626.                     recreate_root() #Re-opening root window
627.                 else:
628.                     pass
629.                 idtk.bind_all('<Return>', finallydeleting)
630.                 idtk.protocol('WM_DELETE_WINDOW',on_closingid)
631.
632.             deletetk.bind_all('<Return>',deleteit)
633.             deleteit()
634.             deletetk.mainloop()
635.
636.
637.         #Function for searching a record
638.         def search():
639.
640.             #Closing menu window
641.             root.destroy()
642.
643.             #Creating window to search a record
644.             searchtk=tk.Tk()

```



```

645.         searchtk.geometry('1000x100')
646.         searchtk.configure(bg='#42c8f5')
647.         searchtk.title('Search Record')
648.
649.         #Asking user if they want to use the movie name or IMDB ID to search
        the record
650.         #Creating Labels
651.         lab0=tk.Label(searchtk,text='Search Record',font=('Casadia Mono
        SemiLight',18,'bold'),bg='#42c8f5')
652.         lab1=tk.Label(searchtk,text='Enter 1 to use movie name or 2 to use
        IMDB ID to search the data:',font=('Casadia Mono SemiLight',14),bg='#42c8f5')
653.
654.         #Placing Labels
655.         lab0.pack()
656.         lab1.place(x=10,y=50)
657.
658.         #Initializing variable to read the entry box data
659.         val=StringVar()
660.
661.         #Creating and placing entry box
662.         en1=tk.Entry(searchtk,textvariable=val,font=('Casadia Mono
        SemiLight',14))
663.         en1.place(x=730,y=53)
664.
665.         #Function to ask for confirmation and close the window
666.         def on_closing():
667.             if messagebox.askyesno(title='QUIT?',message='Are you sure
        you want to quit'):
668.                 nonlocal searchtk
669.                 searchtk.destroy() #Closing this window
670.                 recreate_root() #Re-opening root window
671.             else:
672.                 pass
673.         searchtk.protocol('WM_DELETE_WINDOW',on_closing)
674.
675.         #Checking whether user wants to search using movie name or IMDB ID
676.         def searchit(event=None):
677.             char=val.get()
678.             while char!='':
679.                 if char=='1':
680.                     searchwithname()
681.                     break
682.                 elif char=='2':
683.                     searchwithid()
684.                     break
685.
686.         #Function to search using movie name
687.         def searchwithname():
688.
689.             #Closing choice window

```



```

733.                 display_window = tk.Tk()
734.                 display_window.geometry('600x100')
735.                 display_window.configure(bg= '#42c8f5')
736.                 display_window.title('Search Results for Movie
Name')
737.
738.                 #Creating header labels for the table
739.                 header_labels = ['Movie Name', 'IMDB ID',
'Director', 'Year', 'Genre']
740.                 for i, header in enumerate(header_labels):
741.                     tk.Label(display_window, text=header,
font=('Cascadia Mono SemiLight', 16), bg= '#42c8f5').grid(row=0, column=i,
padx=10, pady=10)
742.
743.                 #Adding movie records in rows
744.                 for i, record in enumerate(result):
745.                     for j, value in enumerate(record[1:]):
746.                         tk.Label(display_window, text=value,
font=('Cascadia Mono SemiLight', 14), bg= '#42c8f5').grid(row=i + 1, column=j,
padx=10, pady=5)
747.
748.                 display_window.mainloop()
749.
750.                 #If record does not exist
751.                 else:
752.                     messagebox.showinfo("No Results", "No records found
for the given movie name.")
753.
754.                 except Exception as e:
755.                     print(f"Error: {e}")
756.
757.                 #Function to ask confirmation from user for quitting
758.                 def on_closingname():
759.                     if messagebox.askyesno(title='QUIT?',message='Are you sure
you want to quit'):
760.                         nonlocal nametk
761.                         nametk.destroy() #Closing this window
762.                         recreate_root() #Re-opening root window
763.                     else:
764.                         pass
765.                     nametk.bind_all('<Return>', finallysearching)
766.                     nametk.protocol('WM_DELETE_WINDOW', on_closingname)
767.
768.                 #Function to search using movie name
769.                 def searchwithid():
770.
771.                     #Closing choice window
772.                     nonlocal searchtk
773.                     searchtk.destroy()
774.

```

```

775.         #Creating window to search using IMDB ID
776.         idtk = tk.Tk()
777.         idtk.geometry('950x100')
778.         idtk.configure(bg='#42c8f5')
779.         idtk.title('Searching Record using IMDB ID')
780.
781.         #Creating Labels
782.         lab0 = tk.Label(idtk, text='Search Record', font=('Cascadia Mono
           SemiLight', 18, 'bold'), bg='#42c8f5')
783.         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')
784.
785.         #Placing Labels
786.         lab0.pack()
787.         lab1.place(x=10,y=50)
788.
789.         #Initializing variable to read entry box
790.         search = StringVar()
791.
792.         #Creating and placing entry box
793.         en1 = tk.Entry(idtk, textvariable=search, font=('Cascadia Mono
           SemiLight', 14))
794.         en1.place(x=690,y=53)
795.
796.         #Function to execute query for searching record using movie name
           and displaying it
797.         def finallysearching(event=None):
798.             try:
799.                 #Getting information from entry box
800.                 idd = search.get()
801.
802.                 #Connecting to MySQL and executing the query
803.                 mydb = sq.connect(host='localhost', user='root',
           password='root', database='movie_database')
804.                 cursor = mydb.cursor()
805.                 cursor.execute("SELECT * FROM Movies WHERE IMDB_id LIKE
           '%{}%'".format(idd))
806.
807.                 #Reading the output provied by MySQL
808.                 result = cursor.fetchall()
809.
810.                 #Displaying output according to the result obtained from
           the query
811.                 #If record exists
812.                 if result:
813.
814.                     #New window to display data
815.                     display_window = tk.Tk()
816.                     display_window.geometry('600x100')
817.                     display_window.configure(bg='#42c8f5')

```

```

818.                 display_window.title('Search Results for Movie
      Name')
819.
820.                 #Creating header labels for the table
821.                 header_labels = ['Movie Name', 'IMDB ID',
      'Director', 'Year', 'Genre']
822.                 for i, header in enumerate(header_labels):
823.                     tk.Label(display_window, text=header,
      font=('Cascadia Mono SemiLight', 16), bg='#42c8f5').grid(row=0, column=i,
      padx=10, pady=10)
824.
825.                 #Adding movie records in rows
826.                 for i, record in enumerate(result):
827.                     for j, value in enumerate(record[1:]):
828.                         tk.Label(display_window, text=value,
      font=('Cascadia Mono SemiLight', 14), bg='#42c8f5').grid(row=i + 1, column=j,
      padx=10, pady=5)
829.
830.                 display_window.mainloop()
831.             else:
832.                 messagebox.showinfo("No Results", "No records found
      for the given IMDB ID.")
833.         except Exception as e:
834.             print(f"Error: {e}")
835.
836.         #Function to ask confirmation from user for quitting
837.         def on_closingid():
838.             if messagebox.askyesno(title='QUIT?',message='Are you sure
      you want to quit'):
839.                 nonlocal idtk
840.                 idtk.destroy() #Closing this window
841.                 recreate_root() #Re-opening root window
842.             else:
843.                 pass
844.             idtk.bind_all('<Return>', finallysearching)
845.             idtk.protocol('WM_DELETE_WINDOW', on_closingid)
846.
847.         searchtk.bind_all('<Return>',searchit)
848.
849.         #Function for displaying the data
850.         def display():
851.
852.             #Closing menu window
853.             root.destroy()
854.
855.             #Creating window to search a record
856.             displaytk=tk.Tk()
857.             displaytk.configure(bg='#42c8f5')
858.             displaytk.geometry('1000x550')
859.             displaytk.title('Display Record')

```

```

860.
861.     #Connecting to MySQL and executing the query
862.     mydb=sql.connect(host="localhost", user="root", password="root",
        database="movie_database")
863.     cursor=mydb.cursor()
864.     sql="SELECT * FROM Movies ORDER BY Movie_Name"
865.     cursor.execute(sql)
866.     myresult=cursor.fetchall()
867.
868.     #Creating a frame
869.     container=tk.Frame(displaytk, bg='#42c8f5')
870.     container.pack(fill='both', expand=True)
871.
872.     #Creating the canvas and the scrollbar
873.     canvas = tk.Canvas(container, bg='#42c8f5')
874.     scrollbar = tk.Scrollbar(container, orient='vertical',
        command=canvas.yview)
875.     scrollable_frame = tk.Frame(canvas, bg='#42c8f5')
876.
877.     #Configuring the scrollable frame
878.     scrollable_frame.bind("<Configure>",lambda e:
        canvas.configure(scrollregion=canvas.bbox("all")))
879.     canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")
880.     canvas.configure(yscrollcommand=scrollbar.set)
881.
882.     #Adding the header labels
883.     header_labels = ['Movie Name', 'Genre', 'Date of Release', 'IMDB
        Id', 'Director', 'Rating']
884.     for i, header in enumerate(header_labels):
885.         tk.Label(scrollable_frame, text=header, font=('Casadia Mono
            SemiLight', 16), bg='#42c8f5').grid(row=0, column=i, padx=10, pady=10)
886.
887.     #Adding movie records in rows
888.     for i, record in enumerate(myresult):
889.         for j, value in enumerate(record):
890.             tk.Label(scrollable_frame, text=value, font=('Casadia Mono
                SemiLight', 14), bg='#42c8f5').grid(row=i+1, column=j, padx=10, pady=5)
891.
892.     #Placing the canvas and the scrollbar
893.     canvas.pack(side="left", fill="both", expand=True)
894.     scrollbar.pack(side="right", fill="y")
895.
896.     #Function to ask confirmation from user for quitting
897.     def on_closing():
898.         if messagebox.askyesno(title='QUIT?', message='Are you sure you
            want to quit'):
899.             displaytk.destroy() #Closing this window
900.             recreate_root() #Re-opening root window
901.         else:
902.             pass

```

```

903.
904.         displaytk.protocol('WM_DELETE_WINDOW', on_closing)
905.         displaytk.mainloop()
906.
907.         #Root window
908.         root=tk.Tk()
909.         root.geometry('690x350')
910.         root.configure(bg='#42c8f5')
911.         root.title('Menu')
912.         lab3=tk.Label(root, text="WELCOME TO BLOCKBUSTER: A Movie
Database",font=('Cascadia Mono SemiLight',16,'bold'),bg='#42c8f5')
913.         lab6=tk.Label(root, text="Made By: Atharava Srivastava",font=('Cascadia
Mono SemiLight',16,'bold'),bg='#42c8f5')
914.         lab3.pack()
915.         lab6.pack()
916.
917.         lin1=tk.Label(root,text="1.Insert new data ",font=('Cascadia Mono
SemiLight',14),bg='#42c8f5')
918.         lin2=tk.Label(root,text="2.Update the table",font=('Cascadia Mono
SemiLight',14),bg='#42c8f5')
919.         lin3=tk.Label(root,text="3.Delete the record from the
table",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
920.         lin4=tk.Label(root,text="4.Search a record from the
table",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
921.         lin5=tk.Label(root,text="5.Display the table",font=('Cascadia Mono
SemiLight',14),bg='#42c8f5')
922.         lin6=tk.Label(root,text="6.Quit",font=('Cascadia Mono
SemiLight',14),bg='#42c8f5')
923.
924.         lin1.place(x=10,y=80)
925.         lin2.place(x=10,y=110)
926.         lin3.place(x=10,y=140)
927.         lin4.place(x=10,y=170)
928.         lin5.place(x=10,y=200)
929.         lin6.place(x=10,y=230)
930.
931.         ch=StringVar()
932.
933.         lab1=tk.Label(root,text="Which function do you want to
apply?:",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
934.         lab1.place(x=10,y=260)
935.         en1=tk.Entry(root, textvariable=ch, font=('Cascadia Mono SemiLight',14))
936.         en1.place(x=420,y=263)
937.
938.         #Function to ask confirmation from user for quitting
939.         def on_closing():
940.             if messagebox.askyesno(title='QUIT?',message='Are you sure you
want to quit'):
941.                 root.destroy() #Closing root window
942.             else:

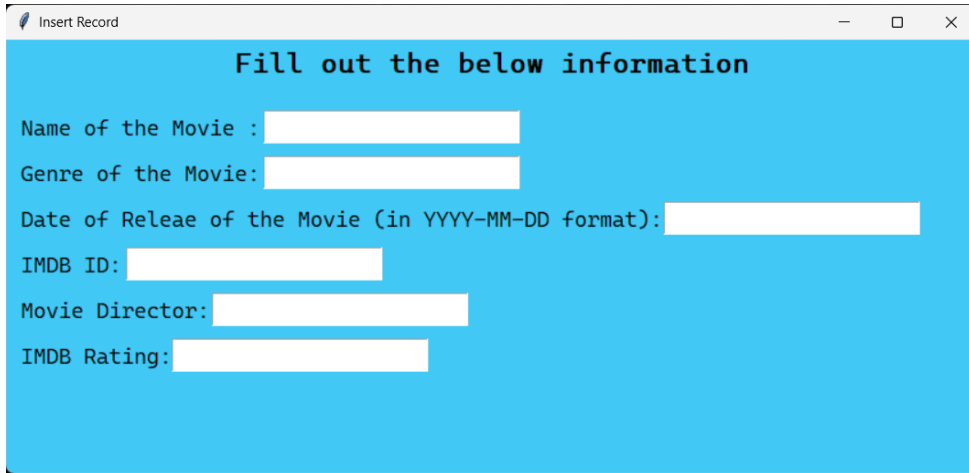
```

```

943.             pass
944.
945.         #Function to accept the choice user from menu items
946.         def choicefunc(event=None):
947.             choice=ch.get()
948.             #To insert new data
949.             if choice=='1':
950.                 insert()
951.             #To update a record
952.             elif choice=='2':
953.                 update()
954.             #To delete a record
955.             elif choice=='3':
956.                 delete()
957.             #To search a record
958.             elif choice=='4':
959.                 search()
960.             #To display the data
961.             elif choice=='5':
962.                 display()
963.             #To exit the program
964.             elif choice=='6':
965.                 exit_=tk.Tk()
966.                 exit_.geometry('500x100')
967.                 exit_.config(bg='#42c8f5')
968.                 exit_.title('Exit')
969.                 label_0=tk.Label(exit_, text="Thank You!",font=('Cascadia Mono
SemiLight',16,'bold'),bg='#42c8f5')
970.                 label_1=tk.Label(exit_, text="Hope you have a nice
day!",font=('Cascadia Mono SemiLight',16,'bold'),bg='#42c8f5')
971.                 label_0.pack()
972.                 label_1.pack()
973.                 root.destroy()
974.             #Invaild input
975.             else:
976.                 lab2=tk.Label(root,text="Please Enter Valid
Input!",font=('Cascadia Mono SemiLight',15),bg='#42c8f5')
977.                 lab2.place(x=170,y=300)
978.                 ch.set('')
979.
980.         en1.bind('<Return>',choicefunc)

```


4. User Interface:



The 'Insert Record' window has a light blue background and a title bar with a feather icon and the text 'Insert Record'. It contains the following text and input fields:

Fill out the below information

Name of the Movie :

Genre of the Movie:

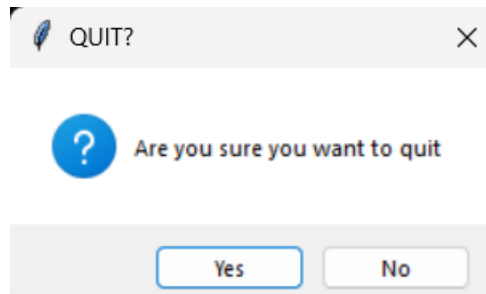
Date of Release of the Movie (in YYYY-MM-DD format):

IMDB ID:

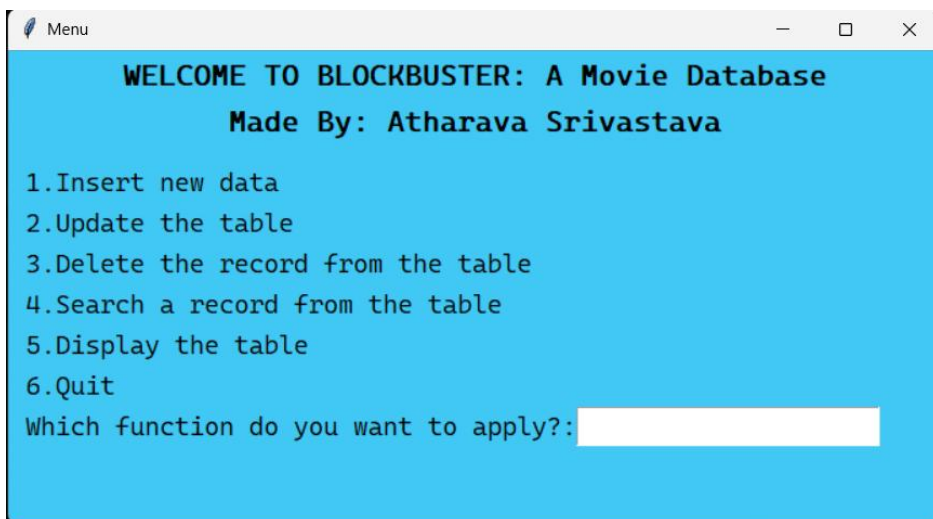
Movie Director:

IMDB Rating:

Main Menu



Exit Confirmation



The 'Menu' window has a light blue background and a title bar with a feather icon and the text 'Menu'. It contains the following text and input field:

WELCOME TO BLOCKBUSTER: A Movie Database

Made By: Atharava Srivastava

1.Insert new data

2.Update the table

3.Delete the record from the table

4.Search a record from the table

5.Display the table

6.Quit

Which function do you want to apply?:

Inserting record

```
Update Record
Enter 1 to use movie name or 2 to use IMDB ID to update the data: 
```

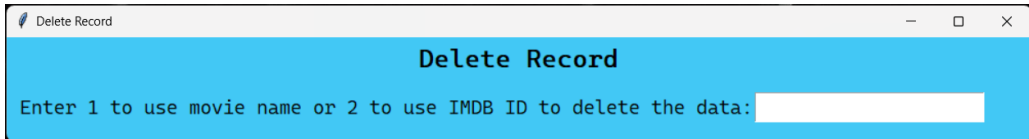
Update choice

```
Updating Record using name
Update Record
Movie name whose record you want to update: 
--> 1. Movie name
--> 2. Genre
--> 3. Date of Release
--> 4. IMDB Id
--> 5. Director
--> 6. IMDB Rating
Record you want to update: 
Enter the change: 
```

Updating with name

```
Updating Record using IMDB ID
Update Record
IMDB ID of the movie whose record you want to update: 
--> 1. Movie name
--> 2. Genre
--> 3. Date of Release
--> 4. IMDB Id
--> 5. Director
--> 6. IMDB Rating
Record you want to update: 
Enter the change: 
```

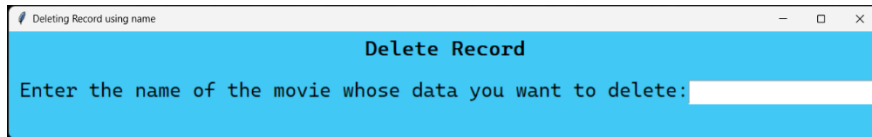
Updating with IMDB ID



Delete Record

Enter 1 to use movie name or 2 to use IMDB ID to delete the data:

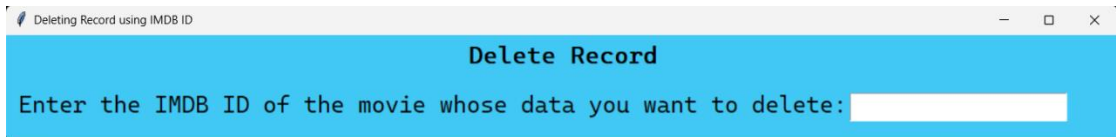
Delete choice



Delete Record

Enter the name of the movie whose data you want to delete:

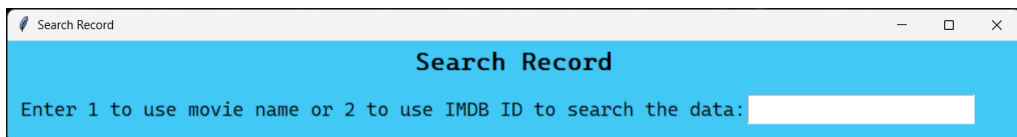
Deleting with name



Delete Record

Enter the IMDB ID of the movie whose data you want to delete:

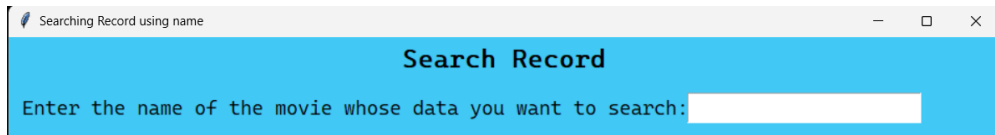
Deleting with IMDB ID



Search Record

Enter 1 to use movie name or 2 to use IMDB ID to search the data:

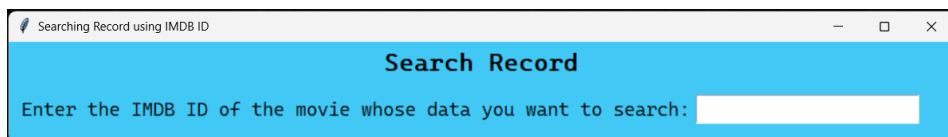
Search choice



Search Record

Enter the name of the movie whose data you want to search:

Searching with name



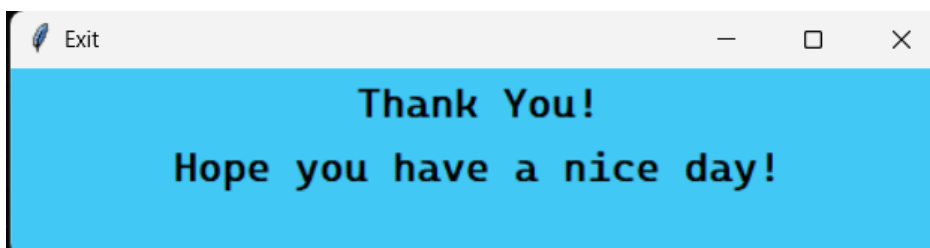
Search Record

Enter the IMDB ID of the movie whose data you want to search:

Searching with IMDB ID

Display Record					
Movie Name	Genre	Date of Release	IMDB Id	Director	Rating
12th Fail	Drama	2023-10-27	8	Vidhu Vinod Chopra	7.9
3 Idiots	Comedy, Drama	2009-12-25	3	Rajkumar Hirani	8.4
Chhichhore	Comedy, Drama	2019-09-06	4	Nitesh Tiwari	8.0
Dil Bechara	Romance, Drama	2020-07-24	1	Mukesh Chhabra	6.6
Forrest Gump	Drama, Romance	1994-07-06	5	Robert Zemeckis	8.8
Friday Night Plan	Comedy, Drama	2023-09-01	7	Vatsal Neelakantan	5.7
Munjya	Horror, Comedy	2024-06-07	2	Aditya Sarpotdar	6.4
Tumse Na Ho Payega	Comedy	2023-09-29	6	Abhishek Sinha	6.2

Display



Exit

5. MySQL Table:

Field	Type	Null	Key	Default	Extra
Movie_Name	varchar(500)	YES		NULL	
Genre	varchar(100)	YES		NULL	
date_of_release	date	YES		NULL	
IMDB_id	int	YES		NULL	
Director	varchar(500)	YES		NULL	
Rating	varchar(500)	YES		NULL	

Table Properties

Movie_Name	Genre	date_of_release	IMDB_id	Director	Rating
Dil Bechara	Romance, Drama	2020-07-24	1	Mukesh Chhabra	6.6
Munjya	Horror, Comedy	2024-06-07	2	Aditya Sarpotdar	6.4
3 Idiots	Comedy, Drama	2009-12-25	3	Rajkumar Hirani	8.4
Chhichhore	Comedy, Drama	2019-09-06	4	Nitesh Tiwari	8.0
Forrest Gump	Drama, Romance	1994-07-06	5	Robert Zemeckis	8.8
Tumse Na Ho Payega	Comedy	2023-09-29	6	Abhishek Sinha	6.2
Friday Night Plan	Comedy, Drama	2023-09-01	7	Vatsal Neelakantan	5.7
12th Fail	Drama	2023-10-27	8	Vidhu Vinod Chopra	7.9

Dummy Database

Recommendations

1. More features can be added to this project like:
 - i. Ability to add whether the movie has been watched yet or not
 - ii. 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>
4. <https://docs.python.org/3/library/tk.html>
5. <https://stackoverflow.com/>
6. <https://docs.python.org/3/>
7. NCERT Class 12th Computer Science textbook