

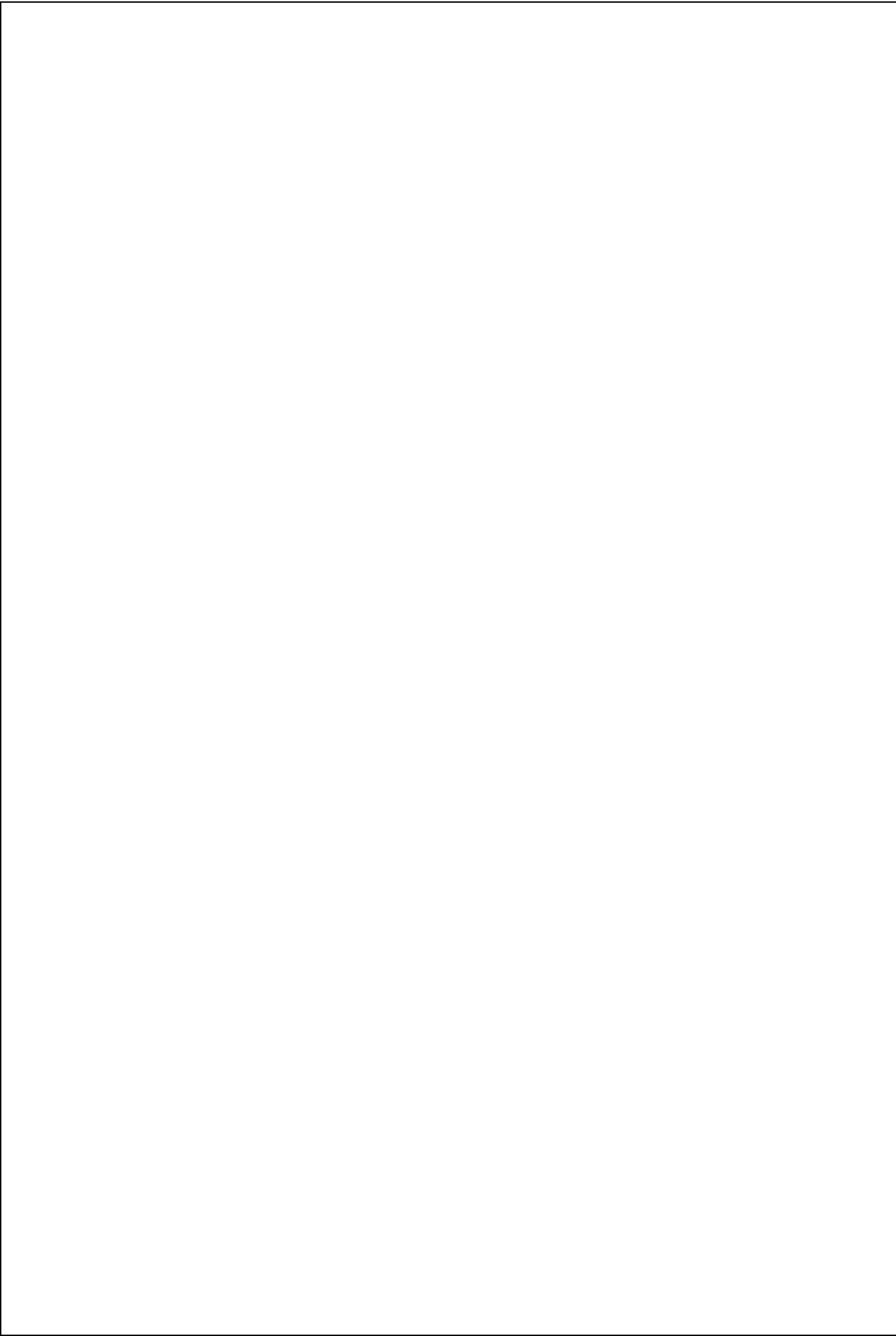
# CBSE

## Computer Science Project



Blockbuster:  
Database Management system for movies

Made By: Atharava Srivastava  
Class: 12<sup>th</sup> Aryabhatta  
Roll No.: 11603528



# 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

Watched it

## 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), Watched_it
        VARCHAR(5))")
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",databas
            e="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("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=('Cascadia Mono SemiLight',14),bg='#42c8f5')
43.     lin2=tk.Label(root,text="2.Update a
    record",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
44.     lin3=tk.Label(root,text="3.Delete a
    record",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
45.     lin4=tk.Label(root,text="4.Search a
    record",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
46.     lin5=tk.Label(root,text="5.Display the
    data",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
47.     lin6=tk.Label(root,text="6.Quit",font=('Cascadia 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=('Cascadia Mono SemiLight',14),bg='#42c8f5')
59.     lab1.place(x=10,y=260)
60.
61.     en1=tk.Entry(root, textvariable=ch, font=('Cascadia 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.         insertk=tk.Tk()
120.         insertk.geometry('850x400')
121.         insertk.configure(bg='#42c8f5')
122.         insertk.title('Insert Record')
123.
124.         #Creating Labels
125.         lab0=tk.Label(insertk,text='Fill out the below
            information',bg='#42c8f5',font=('Cascaadia Mono
            SemiLight',18,'bold'))
126.         lab1=tk.Label(insertk,text="Name of the Movie
            :",bg='#42c8f5',font=('Cascaadia Mono SemiLight',14))
127.         lab2=tk.Label(insertk,text="Genre of the
            Movie:",bg='#42c8f5',font=('Cascaadia Mono SemiLight',14))
128.         lab3=tk.Label(insertk,text="Date of Releae of the
            Movie (in YYYY-MM-DD format):",bg='#42c8f5',font=('Cascaadia
            Mono SemiLight',14))
129.         lab4=tk.Label(insertk,text="IMDB
            ID:",bg='#42c8f5',font=('Cascaadia Mono SemiLight',14))
130.         lab5=tk.Label(insertk,text="Movie
            Director:",bg='#42c8f5',font=('Cascaadia Mono SemiLight',14))
131.         lab6=tk.Label(insertk,text="IMDB
            Rating:",bg='#42c8f5',font=('Cascaadia Mono SemiLight',14))
132.         lab7=tk.Label(insertk,text="Have you watched it
            yet?:",bg='#42c8f5',font=('Cascaadia Mono SemiLight',14))
133.
134.         #Placing Labels
135.         lab0.pack()
136.         lab1.place(x=10,y=60)
137.         lab2.place(x=10,y=100)
138.         lab3.place(x=10,y=140)
139.         lab4.place(x=10,y=180)
140.         lab5.place(x=10,y=220)
141.         lab6.place(x=10,y=260)
142.         lab7.place(x=10,y=300)
143.
144.         #Initializing variables to read the entry box data
145.         nm=StringVar()
146.         genre=StringVar()
147.         dor=StringVar()
148.         code=StringVar()
149.         dr=StringVar()

```

```

150.         rt=StringVar()
151.         seen=StringVar()
152.
153.         #Creating entry boxes
154.         en0=tk.Entry(insertk)
155.         en1=tk.Entry(insertk,textvariable=nm,font=('Casadia
Mono SemiLight',14))
156.         en2=tk.Entry(insertk,textvariable=genre,font=('Casca
dia Mono SemiLight',14))
157.         en3=tk.Entry(insertk,textvariable=dor,font=('Casadia
Mono SemiLight',14))
158.         en4=tk.Entry(insertk,textvariable=code,font=('Casca
dia Mono SemiLight',14))
159.         en5=tk.Entry(insertk,textvariable=dr,font=('Casadia
Mono SemiLight',14))
160.         en6=tk.Entry(insertk,textvariable=rt,font=('Casadia
Mono SemiLight',14))
161.         en7=tk.Entry(insertk,textvariable=seen,font=('Casca
dia Mono SemiLight',14))
162.
163.         #Placing entry boxes
164.         en1.place(x=225,y=62)
165.         en2.place(x=225,y=102)
166.         en3.place(x=575,y=142)
167.         en4.place(x=105,y=182)
168.         en5.place(x=180,y=222)
169.         en6.place(x=145,y=262)
170.         en7.place(x=290,y=302)
171.
172.         #Function to execute query for entering data in MySQL
table
173.         def insertin(event=None):
174.
175.             #Getting information from entry boxes
176.             Name=nm.get()
177.             Genre=genre.get()
178.             DOR=dor.get()
179.             Movie_code=code.get()
180.             Director=dr.get()
181.             Rating=rt.get()
182.             Seen=seen.get()
183.             #Connecting to MySQL and executing the query
184.             mydb =
sq.connect(host="localhost",user="root",password="root",databas
e="movie_database")
185.             mycursor = mydb.cursor()

```

```

186.         sql = "INSERT INTO Movies (Movie_Name, Genre,
           Date_of_release, IMDB_id, Director, Rating, Watched_it) VALUES
           (%s,%s,%s,%s,%s,%s,%s)"
187.         val =
           (Name,Genre,DOR,Movie_code,Director,Rating,Seen)
188.         '''sql = "INSERT INTO Movies (Movie_Name, Genre,
           Date_of_release, IMDB_id, Director, Rating, Watched_it) VALUES
           (%s,%s,%s,%s,%s,%s,%s)"
189.         val =
           (Name,Genre,DOR,Movie_code,Director,Rating,Seen)'''
190.         mycursor.execute(sql, val)
191.         mydb.commit()
192.
193.         #Displaying message for successful insert
194.         added=tk.Label(inserttk,text='Record
           Inserted',font=('Cascadia Mono SemiLight',20),bg='#42c8f5')
195.         added.place(x=300,y=340)
196.
197.         #Setting all entry boxes to blank so that new
           data can be entered
198.         nm.set('')
199.         genre.set('')
200.         dor.set('')
201.         code.set('')
202.         dr.set('')
203.         rt.set('')
204.         seen.set('')
205.
206.         inserttk.bind_all('<Return>', insertin)
207.
208.         #Function to ask confirmation from user for quitting
209.         def on_closing():
210.             if messagebox.askyesno(title='QUIT?',message='Are
           you sure you want to quit'):
211.                 inserttk.destroy() #Closing this window
212.                 recreate_root() #Re-opening root window
213.             else:
214.                 pass
215.
216.         inserttk.protocol('WM_DELETE_WINDOW',on_closing)
217.         inserttk.mainloop()
218.
219.         #Function for updating a record
220.         def update(event=None):
221.
222.             #Closing menu window
223.             root.destroy()

```

```

224.
225.     #Creating window to update a record
226.     updatetk=tk.Tk()
227.     updatetk.geometry('1000x100')
228.     updatetk.configure(bg='#42c8f5')
229.     updatetk.title('Update Record')
230.
231.     #Asking user if they want to use the movie name or
IMDB ID to update the record
232.     #Creating Labels
233.     lab0=tk.Label(updatetk,text='Update
Record',font=('Casadia Mono
SemiLight',18,'bold'),bg='#42c8f5')
234.     lab1=tk.Label(updatetk,text='Enter 1 to use movie
name or 2 to use IMDB ID to update the data:',font=('Casadia
Mono SemiLight',14),bg='#42c8f5')
235.
236.     #Placing Labels
237.     lab0.pack()
238.     lab1.place(x=10,y=50)
239.
240.     #Initializing variable to read the entry box data
241.     val=StringVar()
242.
243.     #Creating and placing entry box
244.     en1=tk.Entry(updatetk,textvariable=val,font=('Casadi
a Mono SemiLight',14))
245.     en1.place(x=740,y=50)
246.
247.     #Function to ask for confirmation and close the
window
248.     def on_closing():
249.         if
messagebox.askyesno(title='QUIT?',message='Are you sure you
want to quit'):
250.             nonlocal updatetk
251.             updatetk.destroy() #Closing this window
252.             recreate_root() #Re-opening root window
253.         else:
254.             pass
255.     updatetk.protocol('WM_DELETE_WINDOW',on_closing)
256.
257.     #Checking whether user wants to update using movie
name or IMDB ID
258.     def updateit(event=None):
259.         char=val.get()
260.         while char!='':

```



```

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

```

```

296.         lab1.place(x=10,y=80)
297.         lab2.place(x=10,y=110)
298.         lab3.place(x=10,y=140)
299.         lab4.place(x=10,y=170)
300.         lab5.place(x=10,y=200)
301.         lab9.place(x=10,y=230)
302.         lab7.place(x=10,y=260)
303.         lab8.place(x=10,y=290)
304.
305.         #Initialising variables to read entry box
306.         upe=StringVar()
307.         fi=StringVar()
308.         fich=StringVar()
309.
310.         #Creating entry boxes
311.         en1=tk.Entry(nametk,textvariable=upe,font=('Casca
dia Mono SemiLight',14))
312.         en2=tk.Entry(nametk,textvariable=fi,font=('Cascad
ia Mono SemiLight',14))
313.         en3=tk.Entry(nametk,textvariable=fich,font=('Casc
adia Mono SemiLight',14))
314.
315.         #Placing entry boxes
316.         en1.place(x=490,y=52)
317.         en2.place(x=300,y=262)
318.         en3.place(x=200,y=294)
319.
320.         #Function to execute query for updating record
using movie name
321.         def finallyupdating(event=None):
322.
323.             #Getting information from entry boxes
324.             up=upe.get()
325.             fields=int(fi.get())
326.             fieldch=fich.get()
327.
328.             #Checking for which data is to be updated
329.             if fields==1:
330.                 field='Movie_Name'
331.             elif fields==2:
332.                 field='Genre'
333.             elif fields==3:
334.                 field='Date_of_release'
335.             elif fields==4:
336.                 field='IMDB_id'
337.             elif fields==5:
338.                 field='Director'

```

```

339.         elif fields==6:
340.             field='Rating'
341.
342.             #Connecting to MySQL and executing the query
343.             mydb=
sq.connect(host="localhost",user="root",passwd="root",database=
"movie_database")
344.             cursor=mydb.cursor()
345.             update="UPDATE Movies set {} = '{}' WHERE
Movie_Name = '{}'.format(field,fieldch,up)
346.             cursor.execute(update)
347.             mydb.commit()
348.
349.             #Displaying message for successful update
350.             lab0=tk.Label(nametk,text="Record
Updated!",font=('Cascadia Mono SemiLight',24),bg='#42c8f5')
351.             lab0.place(x=240,y=325)
352.
353.             #Setting all entry boxes to blank so that
more updates can be done
354.             upe.set('')
355.             fi.set('')
356.             fich.set('')
357.
358.             #Function to ask confirmation from user for
quiting
359.             def on_closingname():
360.                 if
messagebox.askyesno(title='QUIT?',message='Are you sure you
want to quit'):
361.                     nonlocal nametk
362.                     nametk.destroy() #Closing this window
363.                     recreate_root() # Re-opening root window
364.                 else:
365.                     pass
366.             nametk.bind_all('<Return>', finallyupdating)
367.             nametk.protocol('WM_DELETE_WINDOW',on_closingname
)
368.
369.             #Function to update using IMDB ID
370.             def updatewithid():
371.
372.                 #Closing choice window
373.                 nonlocal updatetk
374.                 updatetk.destroy()
375.
376.                 #Creating window to update using IMDB ID

```

```

377.         idtk=tk.Tk()
378.         idtk.geometry('850x380')
379.         idtk.configure(bg='#42c8f5')
380.         idtk.title('Updating Record using IMDB ID')
381.
382.         #Creating Labels
383.         lab0=tk.Label(idtk,text='Update
Record',font=('Cascadia Mono
SemiLight',18,'bold'),bg='#42c8f5')
384.         lab1=tk.Label(idtk,text='--> 1. Movie name
',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
385.         lab2=tk.Label(idtk,text='--> 2.
Genre',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
386.         lab3=tk.Label(idtk,text='--> 3. Date of
Release',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
387.         lab4=tk.Label(idtk,text='--> 4. IMDB
Id',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
388.         lab5=tk.Label(idtk,text='--> 5.
Director',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
389.         lab9=tk.Label(idtk,text='--> 6. IMDB
Rating',font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
390.         lab6=tk.Label(idtk,text="IMDB ID of the movie
whose record you want to update:",font=('Cascadia Mono
SemiLight',14),bg='#42c8f5')
391.         lab7=tk.Label(idtk,text="Record you want to
update:",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
392.         lab8=tk.Label(idtk,text="Enter the
change:",font=('Cascadia Mono SemiLight',14),bg='#42c8f5')
393.
394.         #Placing labels
395.         lab0.pack()
396.         lab6.place(x=10,y=50)
397.         lab1.place(x=10,y=80)
398.         lab2.place(x=10,y=110)
399.         lab3.place(x=10,y=140)
400.         lab4.place(x=10,y=170)
401.         lab5.place(x=10,y=200)
402.         lab9.place(x=10,y=230)
403.         lab7.place(x=10,y=260)
404.         lab8.place(x=10,y=290)
405.
406.         #Initialising variables to read entry box
407.         upe=StringVar()
408.         fi=StringVar()
409.         fich=StringVar()
410.
411.         #Creating entry boxes

```

```

412.         en1=tk.Entry(idtk,textvariable=upe,font=('Cascadia Mono SemiLight',14))
413.         en2=tk.Entry(idtk,textvariable=fi,font=('Cascadia Mono SemiLight',14))
414.         en3=tk.Entry(idtk,textvariable=fich,font=('Cascadia Mono SemiLight',14))
415.
416.         #Placing entry boxes
417.         en1.place(x=600,y=52)
418.         en2.place(x=300,y=262)
419.         en3.place(x=200,y=294)
420.
421.         #Function to execute query for updating record
         using IMDB ID
422.         def finallyupdating(event=None):
423.
424.             #Getting information from entry boxes
425.             up=upe.get()
426.             fields=int(fi.get())
427.             fieldch=fich.get()
428.
429.             #Checking for which data is to be updated
430.             if fields==1:
431.                 field='Movie_Name'
432.             elif fields==2:
433.                 field='Genre'
434.             elif fields==3:
435.                 field='Date_of_release'
436.             elif fields==4:
437.                 field='IMDB_id'
438.             elif fields==5:
439.                 field='Director'
440.             elif fields==6:
441.                 field='Rating'
442.
443.             #Connecting to MySQL and executing the query
444.             mydb=
sq.connect(host="localhost",user="root",passwd="root",database=
"movie_database")
445.             cursor=mydb.cursor()
446.             update="UPDATE Movies set {} = '{}' WHERE
IMDB_ID= '{}'".format(field,fieldch,up)
447.             cursor.execute(update)
448.             mydb.commit()
449.
450.             #Displaying message for successful update

```

```

451.             lab0=tk.Label(idtk,text="Record
Updated!",font=('Cascadia Mono SemiLight',24),bg='#42c8f5')
452.             lab0.place(x=240,y=325)
453.
454.             #Setting all entry boxes to blank so that
more updates can be made
455.             upe.set('')
456.             fi.set('')
457.             fich.set('')
458.
459.             #Function to ask confirmation from user for
quiting
460.             def on_closingid():
461.                 if
messagebox.askyesno(title='QUIT?',message='Are you sure you
want to quit'):
462.                     nonlocal idtk
463.                     idtk.destroy() #Closing this window
464.                     recreate_root() #Re-opening root window
465.                 else:
466.                     pass
467.                 idtk.bind_all('<Return>', finallyupdating)
468.                 idtk.protocol('WM_DELETE_WINDOW',on_closingid)
469.
470.             updatetk.bind_all('<Return>', updateit)
471.             updateit()
472.             updatetk.protocol('WM_DELETE_WINDOW',on_closing)
473.             updatetk.mainloop()
474.
475.             #Function for deleting a record
476.             def delete():
477.
478.                 #Closing menu window
479.                 root.destroy()
480.
481.                 #Creating window to delete a record
482.                 deletetk=tk.Tk()
483.                 deletetk.geometry('1000x100')
484.                 deletetk.configure(bg='#42c8f5')
485.                 deletetk.title('Delete Record')
486.
487.                 #Asking user if they want to use the movie name or
IMDB ID to delete the record
488.                 #Creating labels
489.                 lab0=tk.Label(deletetk,text='Delete
Record',font=('Cascadia Mono
SemiLight',18,'bold'),bg='#42c8f5')

```

```

490.         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')
491.
492.         #Placing Labels
493.         lab0.pack()
494.         lab1.place(x=10,y=50)
495.
496.         #Initializing variable to read the entry box data
497.         val=StringVar()
498.
499.         #Creating and placing entry box
500.         en1=tk.Entry(deletetk,textvariable=val,font=('Cascadi
        a Mono SemiLight',14))
501.         en1.place(x=730,y=53)
502.
503.         #Function to ask for confirmation and close the
        window
504.         def on_closing():
505.             if
        messagebox.askyesno(title='QUIT?',message='Are you sure you
        want to quit'):
506.                 nonlocal deletetk
507.                 deletetk.destroy() #Closing this window
508.                 recreate_root() #Re-opening root window
509.             else:
510.                 pass
511.         deletetk.protocol('WM_DELETE_WINDOW',on_closing)
512.
513.         #Checking whether user wants to delete using movie
        name or IMDB ID
514.         def deleteit(event=None):
515.             char=val.get()
516.             while char!='':
517.                 if char=='1':
518.                     deletewithname()
519.                     break
520.                 elif char=='2':
521.                     deletewithid()
522.                     break
523.
524.         #Function to delete using movie name
525.         def deletewithname():
526.
527.             #Closing choice window
528.             nonlocal deletetk
529.             deletetk.destroy()

```



```

530.
531.         #Creating window to delete using movie name
532.         nametk=tk.Tk()
533.         nametk.geometry('1060x130')
534.         nametk.configure(bg='#42c8f5')
535.         nametk.title('Deleting Record using name')
536.
537.         #Creating Labels
538.         lab0=tk.Label(nametk,text='Delete
Record',font=('Cascadia Mono
SemiLight',18,'bold'),bg='#42c8f5')
539.         lab1=tk.Label(nametk,text='Enter the name of the
movie whose data you want to delete: ',font=('Cascadia Mono
SemiLight',18),bg='#42c8f5')
540.
541.         #Placing Labels
542.         lab0.pack()
543.         lab1.place(x=10,y=50)
544.
545.         #Initialising variable to read entry box
546.         de=StringVar()
547.
548.         #Creating and placing entry box
549.         en1=tk.Entry(nametk,textvariable=de,font=('Cascad
ia Mono SemiLight',14))
550.         en1.place(x=825,y=59)
551.
552.         #Function to execute query for deleteing record
using movie name
553.         def finallydeleting(event=None):
554.             #Getting information from entry box
555.             dele=de.get()
556.
557.             #Connecting to MySQL and executing the query
558.             c=sq.connect(host="localhost",user="root",pas
swd="root",database="movie_database")
559.             cursor=c.cursor()
560.             sql="DELETE FROM Movies WHERE
Movie_Name='{}'.format(dele)
561.             cursor.execute(sql)
562.             c.commit()
563.
564.             #Displaying message for successful delete
565.             lab2=tk.Label(nametk,text="Record
Deleted!",font=('Cascadia Mono
SemiLight',15,'bold'),bg='#42c8f5')
566.             lab2.place(x=400,y=90)

```



```

567.
568.             #Setting all entry boxes to blank so that
             more updates can be done
569.             de.set('')
570.
571.             #Function to ask confirmation from user for
             quitting
572.             def on_closingname():
573.                 if
             messagebox.askyesno(title='QUIT?',message='Are you sure you
             want to quit'):
574.                 nonlocal nametk
575.                 nametk.destroy() #Closing this window
576.                 recreate_root() #Re-opening root window
577.             else:
578.                 pass
579.             nametk.bind_all('<Return>', finallydeleting)
580.             nametk.protocol('WM_DELETE_WINDOW',on_closingname
             )
581.
582.             #Function to delete using IMDB ID
583.             def deletewithid():
584.
585.                 #Closing choice window
586.                 nonlocal deletetk
587.                 deletetk.destroy()
588.
589.                 #Creating window to delete using movie name
590.                 idtk=tk.Tk()
591.                 idtk.geometry('1150x110')
592.                 idtk.configure(bg='#42c8f5')
593.                 idtk.title('Deleting Record using IMDB ID')
594.
595.                 #Creating Labels
596.                 lab0=tk.Label(idtk,text='Delete
             Record',font=('Cascadia Mono
             SemiLight',18,'bold'),bg='#42c8f5')
597.                 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')
598.
599.                 #Placing Labels
600.                 lab0.pack()
601.                 lab1.place(x=10,y=50)
602.
603.                 #Initialising variable to read entry box
604.                 de=StringVar()

```

```

605.
606.         #Creating and placing entry boxes
607.         en1=tk.Entry(idtk,textvariable=de,font=('Cascadia
        Mono SemiLight',14))
608.         en1.place(x=870,y=59)
609.
610.         #Function to execute query for deleteing record
        using IMDB ID
611.         def finallydeleting(event=None):
612.             #Getting information from entry box
613.             dele=de.get()
614.
615.             #Connecting to MySQL and executing the query
616.             c=sq.connect(host="localhost",user="root",pas
        swd="root",database="movie_database")
617.             cursor=c.cursor()
618.             sql="DELETE FROM Movies WHERE IMDB_id =
        '{}'.format(dele)
619.             cursor.execute(sql)
620.             c.commit()
621.
622.             #Displaying message for successful delete
623.             lab2=tk.Label(idtk,text="Record
        Deleted!",font=('Cascadia Mono
        SemiLight',15,'bold'),bg='#42c8f5')
624.             lab2.place(x=320,y=80)
625.
626.             #Setting all entry boxes to blank so that
        more updates can be done
627.             de.set('')
628.
629.             #Function to ask confirmation from user for
        quitting
630.             def on_closingid():
631.                 if
        messagebox.askyesno(title='QUIT?',message='Are you sure you
        want to quit'):
632.                     nonlocal idtk
633.                     idtk.destroy() #Closing this window
634.                     recreate_root() #Re-opening root window
635.                 else:
636.                     pass
637.             idtk.bind_all('<Return>', finallydeleting)
638.             idtk.protocol('WM_DELETE_WINDOW',on_closingid)
639.
640.         deletetk.bind_all('<Return>',deleteit)
641.         deleteit()

```

```

642.         deletetk.mainloop()
643.
644.
645.     #Function for searching a record
646.     def search():
647.
648.         #Closing menu window
649.         root.destroy()
650.
651.         #Creating window to search a record
652.         searchtk=tk.Tk()
653.         searchtk.geometry('1000x100')
654.         searchtk.configure(bg='#42c8f5')
655.         searchtk.title('Search Record')
656.
657.         #Asking user if they want to use the movie name or
        IMDB ID to search the record
658.         #Creating Labels
659.         lab0=tk.Label(searchtk,text='Search
        Record',font=('Cascadia Mono
        SemiLight',18,'bold'),bg='#42c8f5')
660.         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')
661.
662.         #Placing Labels
663.         lab0.pack()
664.         lab1.place(x=10,y=50)
665.
666.         #Initializing variable to read the entry box data
667.         val=StringVar()
668.
669.         #Creating and placing entry box
670.         en1=tk.Entry(searchtk,textvariable=val,font=('Cascadia
        a Mono SemiLight',14))
671.         en1.place(x=730,y=53)
672.
673.         #Function to ask for confirmation and close the
        window
674.         def on_closing():
675.             if
        messagebox.askyesno(title='QUIT?',message='Are you sure you
        want to quit'):
676.                 nonlocal searchtk
677.                 searchtk.destroy() #Closing this window
678.                 recreate_root() #Re-opening root window
679.             else:

```

```

680.                 pass
681.         searchtk.protocol('WM_DELETE_WINDOW',on_closing)
682.
683.         #Checking whether user wants to search using movie
name or IMDB ID
684.         def searchit(event=None):
685.             char=val.get()
686.             while char!='':
687.                 if char=='1':
688.                     searchwithname()
689.                     break
690.                 elif char=='2':
691.                     searchwithid()
692.                     break
693.
694.         #Function to search using movie name
695.         def searchwithname():
696.
697.             #Closing choice window
698.             nonlocal searchtk
699.             searchtk.destroy()
700.
701.             #Creating window to search using movie name
702.             nametk = tk.Tk()
703.             nametk.geometry('950x100')
704.             nametk.configure(bg='#42c8f5')
705.             nametk.title('Searching Record using name')
706.
707.             #Creating Labels
708.             lab0 = tk.Label(nametk, text='Search Record',
709.                             font=('Cascadia Mono SemiLight', 18, 'bold'), bg='#42c8f5')
710.             lab1 = tk.Label(nametk, text='Enter the name of
711. the movie whose data you want to search: ', font=('Cascadia
712. Mono SemiLight', 14), bg='#42c8f5')
713.
714.             #Placing Labels
715.             lab0.pack()
716.             lab1.place(x=10,y=50)
717.
718.             #Initializing variable to read entry box
719.             search = StringVar()
720.
721.             #Creating and placing entry box
722.             en1 = tk.Entry(nametk, textvariable=search,
723.                             font=('Cascadia Mono SemiLight', 14))
724.             en1.place(x=650,y=53)

```

```

722.         #Function to execute query for searching record
        using movie name and displaying it
723.         def finallysearching(event=None):
724.             try:
725.                 #Getting information from entry box
726.                 idd = search.get()
727.
728.                 #Connecting to MySQL and executing the
        query
729.                 mydb = sq.connect(host='localhost',
        user='root', password='root', database='movie_database')
730.                 cursor = mydb.cursor()
731.                 cursor.execute("SELECT * FROM Movies
        WHERE Movie_name = '{}'.format(idd))
732.
733.                 #Reading the output provied by MySQL
734.                 result = cursor.fetchall()
735.                 #Displaying output according to the
        result obtained from the query
736.                 #If record exists
737.                 if result:
738.
739.                     #New window to display data
740.                     display_window = tk.Tk()
741.                     display_window.geometry('1040x110')
742.                     display_window.configure(bg='#42c8f5'
        )
743.                     display_window.title('Search Results
        for Movie Name')
744.
745.                     #Creating header labels for the table
746.                     header_labels = ['Movie Name',
        'Genre', 'Date of Release', 'IMDB ID', 'Director',
        'Rating', 'Watched it?']
747.                     for i, header in
        enumerate(header_labels):
748.                         tk.Label(display_window,
        text=header, font=('Cascadia Mono SemiLight', 16),
        bg='#42c8f5').grid(row=0, column=i, padx=10, pady=10)
749.
750.                     #Adding movie records in rows
751.                     for i, record in enumerate(result):
752.                         for j, value in
        enumerate(record):
753.                             tk.Label(display_window,
        text=value, font=('Cascadia Mono SemiLight', 14),
        bg='#42c8f5').grid(row=i + 1, column=j, padx=10, pady=5)

```

```

754.
755.             display_window.mainloop()
756.         else:
757.             messagebox.showinfo("No Results", "No
records found for the given IMDB ID.")
758.         except Exception as e:
759.             print("Error: {e}")
760.
761.         #Function to ask confirmation from user for
quitting
762.         def on_closingname():
763.             if
messagebox.askyesno(title='QUIT?',message='Are you sure you
want to quit'):
764.                 nonlocal nametk
765.                 nametk.destroy() #Closing this window
766.                 recreate_root() #Re-opening root window
767.             else:
768.                 pass
769.                 nametk.bind_all('<Return>', finallysearching)
770.                 nametk.protocol('WM_DELETE_WINDOW',
on_closingname)
771.
772.         #Function to search using movie name
773.         def searchwithid():
774.
775.             #Closing choice window
776.             nonlocal searchtk
777.             searchtk.destroy()
778.
779.             #Creating window to search using IMDB ID
780.             idtk = tk.Tk()
781.             idtk.geometry('950x100')
782.             idtk.configure(bg='#42c8f5')
783.             idtk.title('Searching Record using IMDB ID')
784.
785.             #Creating Labels
786.             lab0 = tk.Label(idtk, text='Search Record',
font=('Cascadia Mono SemiLight', 18, 'bold'), bg='#42c8f5')
787.             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')
788.
789.             #Placing labels
790.             lab0.pack()
791.             lab1.place(x=10,y=50)
792.

```

```

793.         #Initializing variable to read entry box
794.         search = StringVar()
795.
796.         #Creating and placing entry box
797.         en1 = tk.Entry(idtk, textvariable=search,
798.             font=('Cascadia Mono SemiLight', 14))
799.         en1.place(x=690,y=53)
800.         #Function to execute query for searching record
            using movie name and displaying it
801.         def finallysearching(event=None):
802.             try:
803.                 #Getting information from entry box
804.                 idd = search.get()
805.
806.                 #Connecting to MySQL and executing the
                    query
807.                 mydb = sq.connect(host='localhost',
808.                     user='root', password='root', database='movie_database')
809.                 cursor = mydb.cursor()
810.                 cursor.execute("SELECT * FROM Movies
811.                     WHERE IMDB_id = '{}'.format(idd))
812.
813.                 #Reading the output provied by MySQL
814.                 result = cursor.fetchall()
815.                 #Displaying output according to the
                    result obtained from the query
816.                 #If record exists
817.                 if result:
818.
819.                     #New window to display data
820.                     display_window = tk.Tk()
821.                     display_window.geometry('1040x110')
822.                     display_window.configure(bg='#42c8f5'
823.                         )
824.                     display_window.title('Search Results
825.                         for IMDB ID')
826.
827.                     #Creating header labels for the table
828.                     header_labels = ['Movie Name',
829.                         'Genre','Date of Release', 'IMDB ID', 'Director',
830.                         'Rating','Watched it?']
831.
832.                     for i, header in
833.                         enumerate(header_labels):
834.                             tk.Label(display_window,
835.                                 text=header, font=('Cascadia Mono SemiLight', 16),
836.                                 bg='#42c8f5').grid(row=0, column=i, padx=10, pady=10)

```

```

827.
828.             #Adding movie records in rows
829.             for i, record in enumerate(result):
830.                 for j, value in
                     enumerate(record):
831.                     tk.Label(display_window,
                        text=value, font=('Cascadia Mono SemiLight', 14),
                        bg='#42c8f5').grid(row=i + 1, column=j, padx=10, pady=5)
832.
833.                     display_window.mainloop()
834.                 else:
835.                     messagebox.showinfo("No Results", "No
                        records found for the given IMDB ID.")
836.                     except Exception as e:
837.                         print("Error: {e}")
838.
839.             #Function to ask confirmation from user for
                quitting
840.             def on_closingid():
841.                 if
                     messagebox.askyesno(title='QUIT?',message='Are you sure you
                        want to quit'):
842.                     nonlocal idtk
843.                     idtk.destroy() #Closing this window
844.                     recreate_root() #Re-opening root window
845.                 else:
846.                     pass
847.                 idtk.bind_all('<Return>', finallysearching)
848.                 idtk.protocol('WM_DELETE_WINDOW', on_closingid)
849.
850.                 searchtk.bind_all('<Return>',searchit)
851.
852.             #Function for displaying the data
853.             def display():
854.
855.                 #Closing menu window
856.                 root.destroy()
857.
858.                 #Creating window to search a record
859.                 displaytk=tk.Tk()
860.                 displaytk.configure(bg='#42c8f5')
861.                 displaytk.geometry('1210x400')
862.                 displaytk.title('Display Record')
863.
864.                 #Connecting to MySQL and executing the query
865.                 mydb=sq.connect(host="localhost", user="root",
                        password="root", database="movie_database")

```



```

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

```

```

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

```

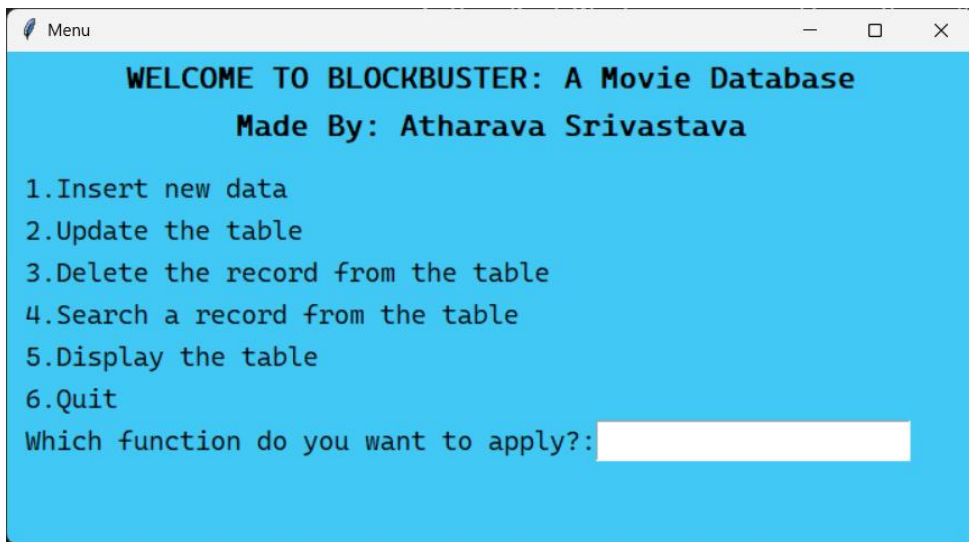
```

939.     en1.place(x=420,y=263)
940.
941.     #Function to ask confirmation from user for quitting
942.     def on_closing():
943.         if messagebox.askyesno(title='QUIT?',message='Are
you sure you want to quit'):
944.             root.destroy() #Closing root window
945.
946.             #Creating exit window
947.             exit_=tk.Tk()
948.             exit_.geometry('500x100')
949.             exit_.config(bg='#42c8f5')
950.             exit_.title('Exit')
951.             label_0=tk.Label(exit_, text="Thank
You!",font=('Cascadia Mono SemiLight',16,'bold'),bg='#42c8f5')
952.             label_1=tk.Label(exit_, text="Hope you have a
nice day!",font=('Cascadia Mono
SemiLight',16,'bold'),bg='#42c8f5')
953.             label_0.pack()
954.             label_1.pack()
955.         else:
956.             pass
957.
958.     root.protocol('WM_DELETE_WINDOW',on_closing)
959.
960.     #Function to accept the choice user from menu items
961.     def choicefunc(event=None):
962.         choice=ch.get()
963.         #To insert new data
964.         if choice=='1':
965.             insert()
966.         #To update a record
967.         elif choice=='2':
968.             update()
969.         #To delete a record
970.         elif choice=='3':
971.             delete()
972.         #To search a record
973.         elif choice=='4':
974.             search()
975.         #To display the data
976.         elif choice=='5':
977.             display()
978.         #To exit the program
979.         elif choice=='6':
980.             #Creating exit window
981.             exit_=tk.Tk()

```

```
982.         exit_.geometry('500x100')
983.         exit_.config(bg='#42c8f5')
984.         exit_.title('Exit')
985.         label_0=tk.Label(exit_, text="Thank
    You!",font=('Cascadia Mono SemiLight',16,'bold'),bg='#42c8f5')
986.         label_1=tk.Label(exit_, text="Hope you have a
    nice day!",font=('Cascadia Mono
    SemiLight',16,'bold'),bg='#42c8f5')
987.         label_0.pack()
988.         label_1.pack()
989.         root.destroy()
990.
991.         #Invaild input
992.         else:
993.             lab2=tk.Label(root,text="Please Enter Valid
    Input!",font=('Cascadia Mono SemiLight',15),bg='#42c8f5')
994.             lab2.place(x=170,y=300)
995.             ch.set('')
996.
997.         #Binding Return to key accept final inputs.
998.         en1.bind('<Return>',choicefunc)
```

## 4. User Interface:



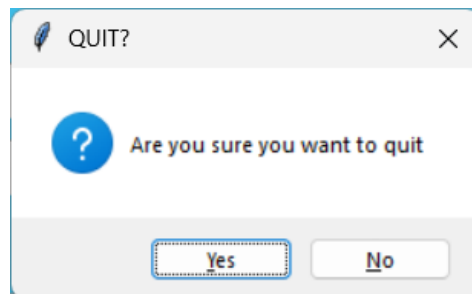
Menu

**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?:

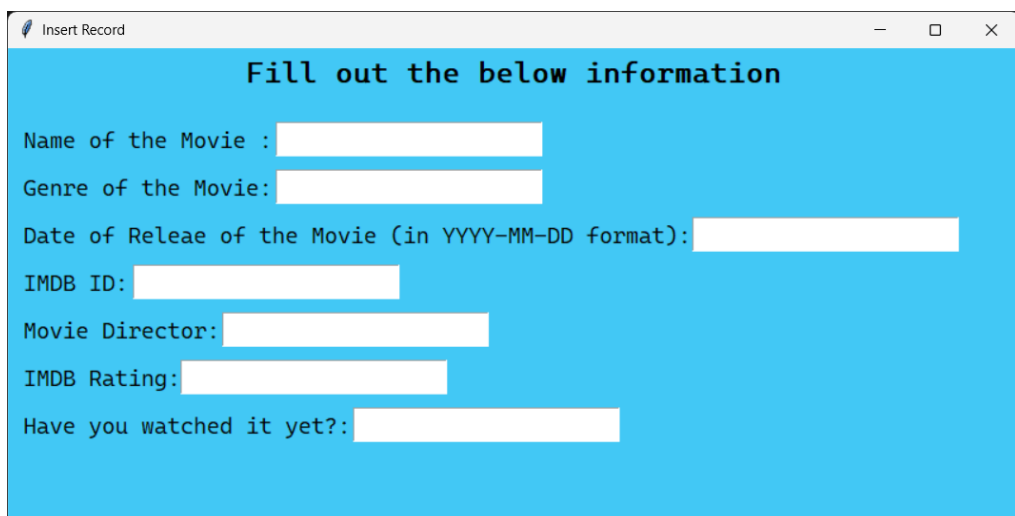
Main Menu



QUIT?

Are you sure you want to quit

Exit Confirmation



Insert Record

**Fill out the below information**

Name of the Movie :

Genre of the Movie:

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

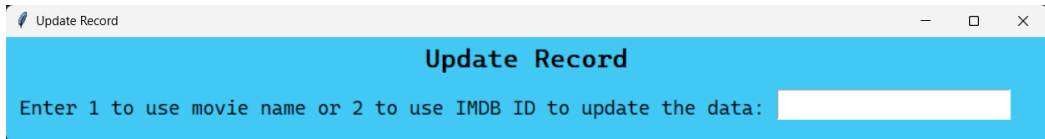
IMDB ID:

Movie Director:

IMDB Rating:

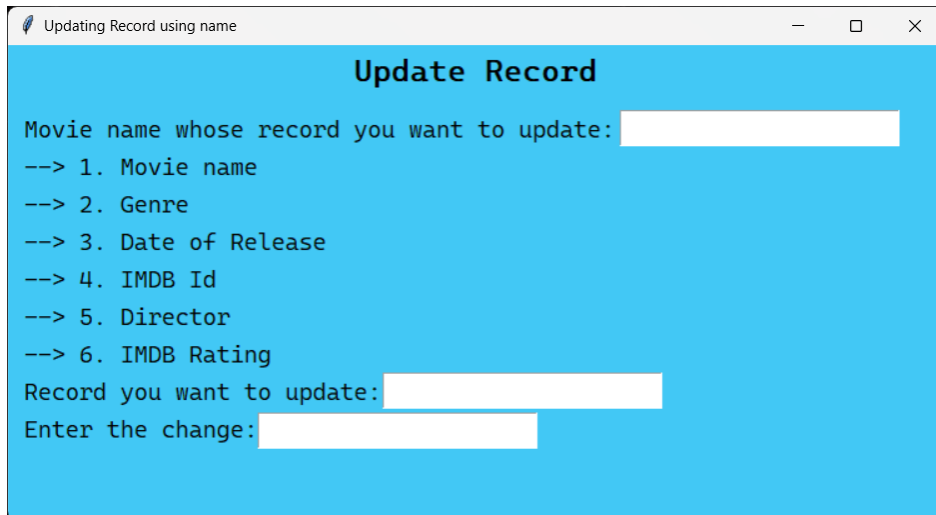
Have you watched it yet?:

Inserting record




A terminal window titled "Update Record" with a blue background. It contains the text "Enter 1 to use movie name or 2 to use IMDB ID to update the data:" followed by a white input field.

## Update choice



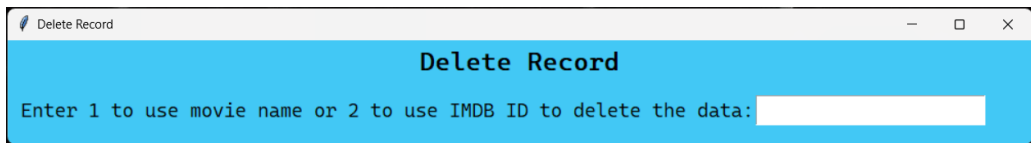
A terminal window titled "Updating Record using name" with a blue background. It contains the text "Movie name whose record you want to update:" followed by a white input field. Below this is a list of six options: "1. Movie name", "2. Genre", "3. Date of Release", "4. IMDB Id", "5. Director", and "6. IMDB Rating". Each option is preceded by "-->". Below the list are two more white input fields: "Record you want to update:" and "Enter the change:".

## Updating with name



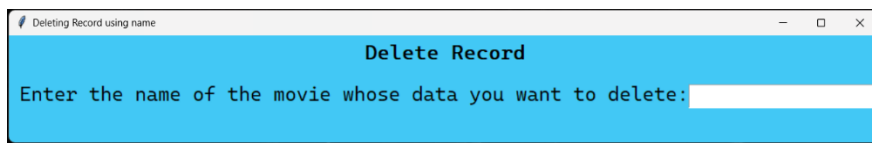
A terminal window titled "Updating Record using IMDB ID" with a blue background. It contains the text "IMDB ID of the movie whose record you want to update:" followed by a white input field. Below this is a list of six options: "1. Movie name", "2. Genre", "3. Date of Release", "4. IMDB Id", "5. Director", and "6. IMDB Rating". Each option is preceded by "-->". Below the list are two more white input fields: "Record you want to update:" and "Enter the change:".

## Updating with IMDB ID



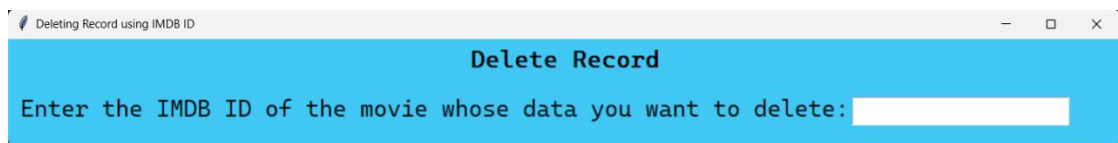
The screenshot shows a web browser window with the title 'Delete Record'. The page has a blue header with the text 'Delete Record'. Below the header, there is a text input field with the placeholder text 'Enter 1 to use movie name or 2 to use IMDB ID to delete the data:'. The input field is empty.

Delete choice



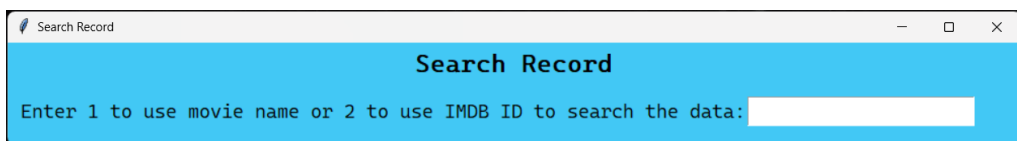
The screenshot shows a web browser window with the title 'Deleting Record using name'. The page has a blue header with the text 'Delete Record'. Below the header, there is a text input field with the placeholder text 'Enter the name of the movie whose data you want to delete:'. The input field is empty.

Deleting with name



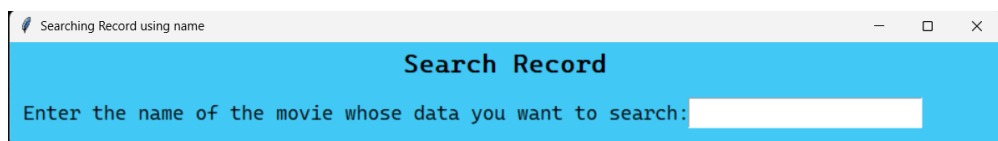
The screenshot shows a web browser window with the title 'Deleting Record using IMDB ID'. The page has a blue header with the text 'Delete Record'. Below the header, there is a text input field with the placeholder text 'Enter the IMDB ID of the movie whose data you want to delete:'. The input field is empty.

Deleting with IMDB ID



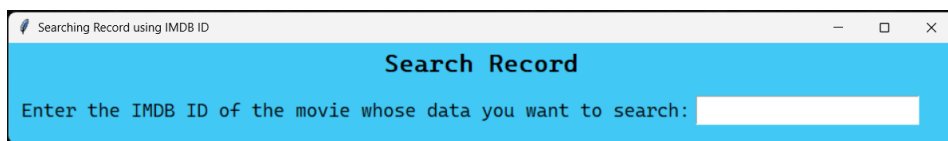
The screenshot shows a web browser window with the title 'Search Record'. The page has a blue header with the text 'Search Record'. Below the header, there is a text input field with the placeholder text 'Enter 1 to use movie name or 2 to use IMDB ID to search the data:'. The input field is empty.

Search choice



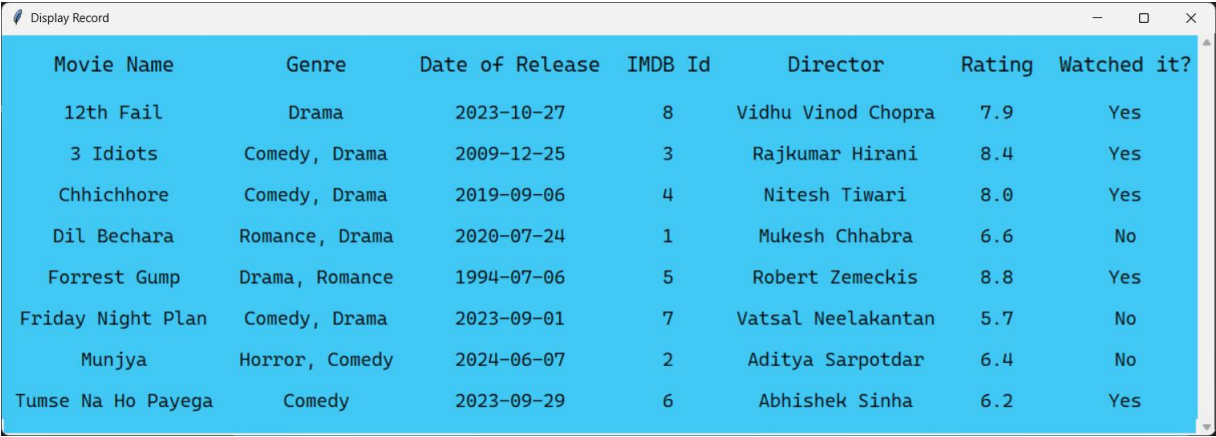
The screenshot shows a web browser window with the title 'Searching Record using name'. The page has a blue header with the text 'Search Record'. Below the header, there is a text input field with the placeholder text 'Enter the name of the movie whose data you want to search:'. The input field is empty.

Searching with name



The screenshot shows a web browser window with the title 'Searching Record using IMDB ID'. The page has a blue header with the text 'Search Record'. Below the header, there is a text input field with the placeholder text 'Enter the IMDB ID of the movie whose data you want to search:'. The input field is empty.

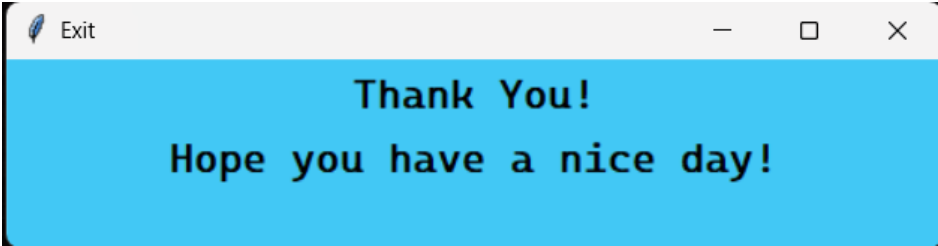
Searching with IMDB ID



A screenshot of a window titled "Display Record". The window contains a table with 7 columns: "Movie Name", "Genre", "Date of Release", "IMDB Id", "Director", "Rating", and "Watched it?". The table lists 9 movies. The background of the table is light blue.

Movie Name	Genre	Date of Release	IMDB Id	Director	Rating	Watched it?
12th Fail	Drama	2023-10-27	8	Vidhu Vinod Chopra	7.9	Yes
3 Idiots	Comedy, Drama	2009-12-25	3	Rajkumar Hirani	8.4	Yes
Chhichhore	Comedy, Drama	2019-09-06	4	Nitesh Tiwari	8.0	Yes
Dil Bechara	Romance, Drama	2020-07-24	1	Mukesh Chhabra	6.6	No
Forrest Gump	Drama, Romance	1994-07-06	5	Robert Zemeckis	8.8	Yes
Friday Night Plan	Comedy, Drama	2023-09-01	7	Vatsal Neelakantan	5.7	No
Munjya	Horror, Comedy	2024-06-07	2	Aditya Sarpotdar	6.4	No
Tumse Na Ho Payega	Comedy	2023-09-29	6	Abhishek Sinha	6.2	Yes

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	
Watched_it	varchar(5)	YES		NULL	

Table Properties

Movie_Name	Genre	date_of_release	IMDB_id	Director	Rating	Watched_it
Dil Bechara	Romance, Drama	2020-07-24	1	Mukesh Chhabra	6.6	No
Munjya	Horror, Comedy	2024-06-07	2	Aditya Sarpotdar	6.4	No
3 Idiots	Comedy, Drama	2009-12-25	3	Rajkumar Hirani	8.4	Yes
Chhichhore	Comedy, Drama	2019-09-06	4	Nitesh Tiwari	8.0	Yes
Forrest Gump	Drama, Romance	1994-07-06	5	Robert Zemeckis	8.8	Yes
Tumse Na Ho Payega	Comedy	2023-09-29	6	Abhishek Sinha	6.2	Yes
Friday Night Plan	Comedy, Drama	2023-09-01	7	Vatsal Neelakantan	5.7	No
12th Fail	Drama	2023-10-27	8	Vidhu Vinod Chopra	7.9	Yes

Dummy Database

# Recommendations

1. More features can be added to this project like:
  - i. Ability to add personal reviews for the movie
  - ii. Allow users to make a fuzzy search, i.e., ability to search even with wrong spellings
  - iii. Display data with some kind of sort
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 12<sup>th</sup> Computer Science textbook