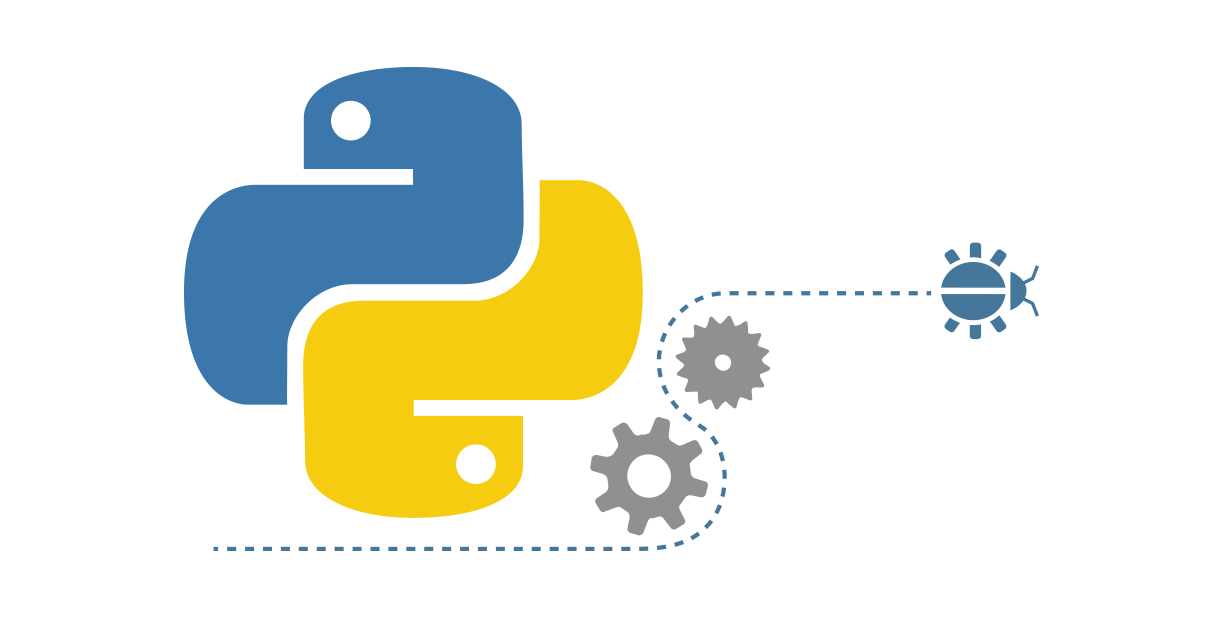
Application Design for

an app launch on Google Play Store



Submitted by: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date of Submission: 5th January 2019.

Submitted to: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Under the Guidance of : Junaid Khateeb (Director, Khateeb Insitute of Technical Education)

**Certificate of Completion**

This is to certify that,

Mr./Ms. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_has successfully implemented an application designed to study the data and generate insights for an app launch on Google Playstore.

The Application has been accepted as a completed project as it meets all the requirements specified.

5th January 2020

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Khateeb Institute of Technical Education)

Acknowledgement:

**I would like to express our sincere gratitude to my principal, director and HOD of computer engineering department from Rajiv Gandhi Institute of Technology, Dr. Sanjay U.Bokade and Dr.S.Y.Ket for providing their invaluable guidance, comments and suggestions throughout the course of the project. I would also like to thank prof.Dilip Dalgade and other professors who have been supportive throughout.**

Table of Contents

[Section 1 5](#_Toc29327233)

[System requirement specifications: 5](#_Toc29327234)

[Section 2 6](#_Toc29327235)

[Technology used: 6](#_Toc29327236)

[a) Python: 6](#_Toc29327237)

[b) Anaconda Navigator 6](#_Toc29327238)

[c)Microsoft excel 6](#_Toc29327239)

[d)WampServer 7](#_Toc29327240)

[e) Microsoft Word**:** 7](#_Toc29327241)

[Section 3: 8](#_Toc29327242)

[Data provided by the client: 8](#_Toc29327243)

[Section 4: 9](#_Toc29327244)

[Screenshots of the respective codes of the questions and their outputs 9](#_Toc29327245)

[Section 5: Testing 91](#_Toc29327246)

[Section 6: The Source Code 92](#_Toc29327247)

[Mainscreen.py 92](#_Toc29327248)

[Funcques.py 135](#_Toc29327249)

# Section 1

## System requirement specifications:

1)A PC or Laptop with Minimum Intel Dual Core Processor with at least speed of 2.0 Ghz

2)Minimum of 2gb Ram 1600Mhz

3)If the System is a 64-bit architecture than 64-bit application must be installed and If the System is a 32-bit architecture than 32-bit application must be installed.

4)Operating System: Windows 7 and Above, Linux or Mac OS X

5)Disk Space of minimum 5 GB

# Section 2

## Technology used:

a) Python: the first technology(software) we used to program the information as per requirements was python. Python is a programming language which is at a high level interpreted, interactive and object-oriented scripting language. python is designed to be highly readable.it uses English keywords frequently where as other languages uses punctuations, and it has fewer syntactical construction than other languages.

* Python is interpreted
* Python is interactive
* Python is object oriented

Python is a programming language that is easier to understand and enables our project to be completed at a faster rate due to features it offers.

b) Anaconda Navigator: anaconda is a free and open-source distribution of python programing languages and R programming languages for scientific computing (data science, large scale data processing).it aims to simplify package management and deployment.

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, macOS, and Linux.

* Spyder present in anaconda navigator is an integrated development environment (IDE) for python. Sypder was used throughout our program.

c)Microsoft excel**:** Microsoft Excel is a spreadsheet developed by Microsoft for Windows, macOS, Android and iOS.it features calculations graphic tools, pivot tables and a macro programing language called visual Basic for applications. Our data(information) in order to process by making use of python programming language was provide to us by Microsoft Excel. Through Microsoft excel we were enable to access the data making use of the pd.read\_csv([provide location of the file]) functions present in python .we were able to read data and make our logical programming statements

d)WampServer: Wamp server is a database system that we used in python in order to store data and retrieve it WampServer is a utility designed to allows you to create Web applications and manage your server and databases. WampServer is a Windows web development environment. It allows you to create web applications with Apache2, PHP and a MySQL database. It also comes with PhpMyAdmin and SQLite Manager to easily manage your databases. WampServer installs automatically (installer), and its usage is very intuitive. You will be able to tune your server without even touching the setting files. WampServer is the only packaged solution that will allow you to reproduce your production server. Once WampServer is installed, you have the possibility to add as many Apache, MySQL, and PHP releases as you want. WampServer also has a tray icon to manage your server and its settings.

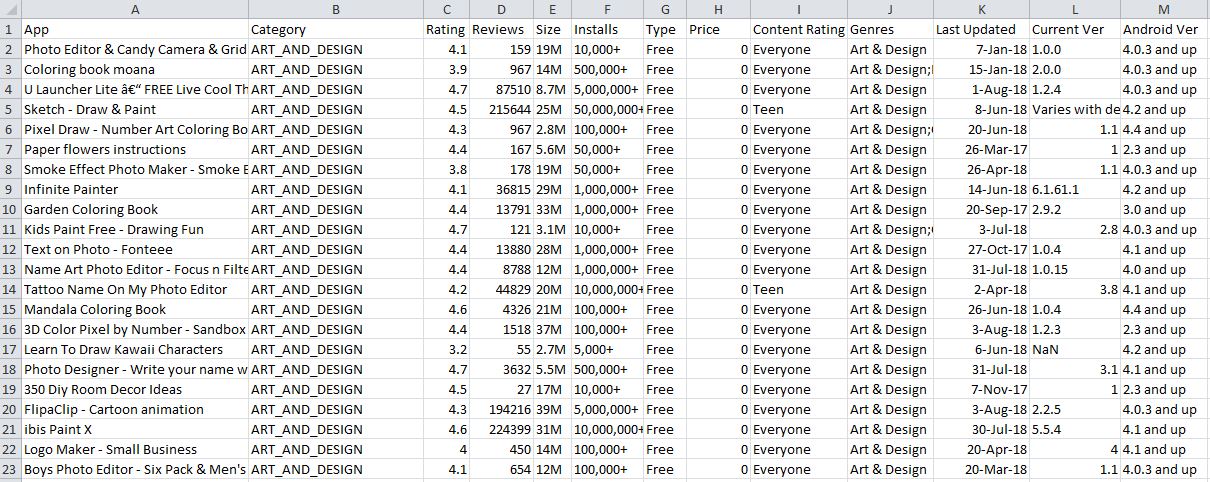
### e) Microsoft Word**:**

Microsoft Word is a word processing program that was first developed by Microsoft in 1983. Since that time, Microsoft has released an abundance of updated versions, each offering more features and incorporating better technology than the one before it. The most current web-based version of Microsoft Word is [Office 365](https://www.lifewire.com/how-to-install-office-365-4589808), but the software version of [Microsoft Office 2019](https://www.lifewire.com/microsoft-office-2019-4159375) includes Word 2019.

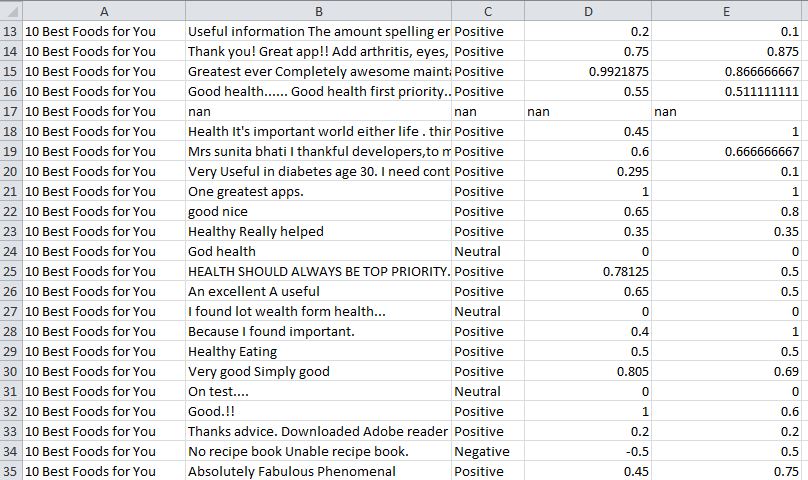
# Section 3:

## Data provided by the client:

Data set 1 provided by the client is a



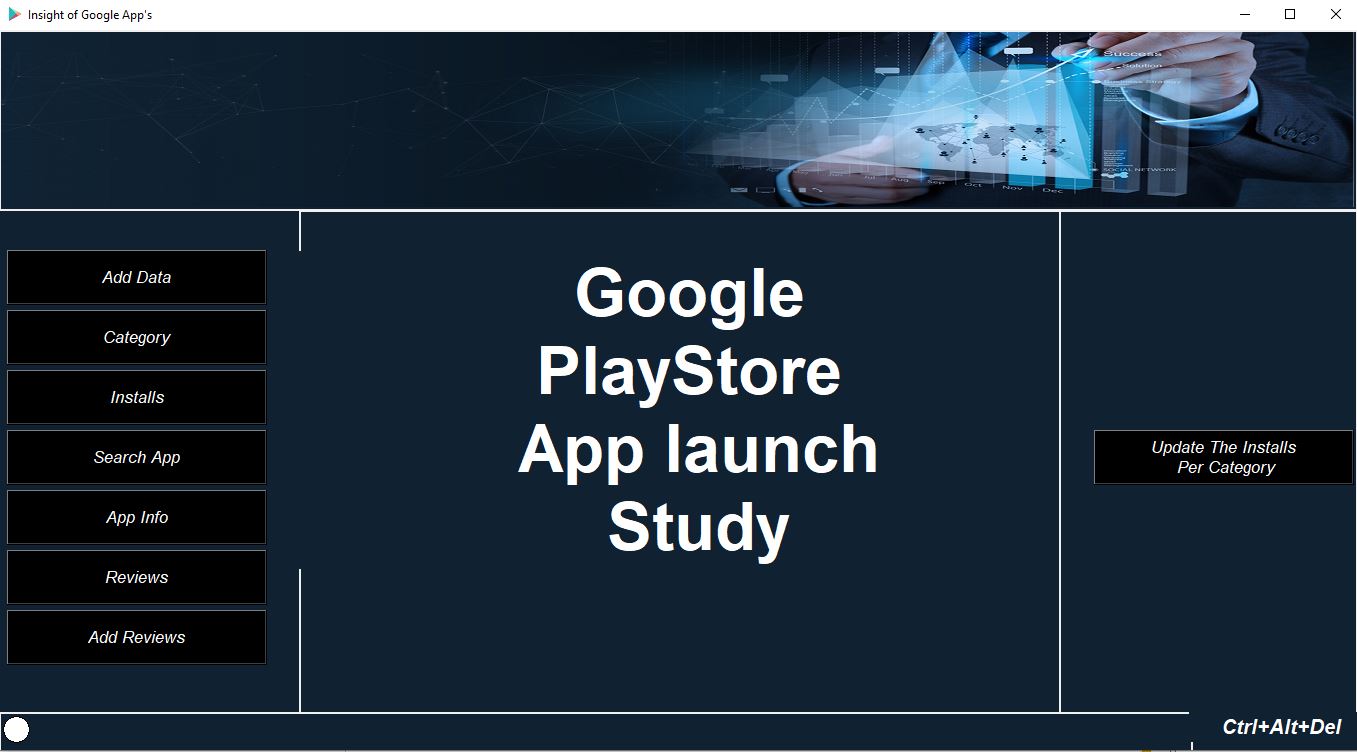
Data set 2 provided by the client is a User Review Dataset

****

# Section 4:

## Screenshots of the respective codes of the questions and their outputs

**The MainScreen**

****

#===============================================main screen======================================================

root=Tk()

root.title("Insight of Google App's")

width\_value=root.winfo\_screenwidth()

height\_value=root.winfo\_screenheight()

root.geometry("%dx%d+250+100"%(1360,720))

root.configure(background='#102131')

root.iconbitmap(r"C:\\InternshipFinal\\google.ico")

#=================================top canvas===================================================================

photocanvas=Canvas(root,width =1355,height=177,bg='#102131')

photocanvas.place(x=0,y=0)

myimg=PhotoImage(file="C:\\InternshipFinal\\predictive\_analytics\_banner.png")

photocanvas.create\_image(0,0,anchor=NW,image=myimg)

photocanvas.image =myimg

#================================================main canvas ==============================

mcanvas=Canvas(width = 760,height=500,bg='#102131',bd='0')

mcanvas.place(x=300,y=180)

head=Label(mcanvas,text="Google \nPlayStore \n App launch \nStudy",width=30,font=("Lucida",50,'bold'),fg='#ffffff',bg='#102131')

mcanvas.create\_window(400, 200, window=head)

#=====================================================options==================================================

lbl\_over = Button(root,text = "Add Data",width=25,height='2',font=("Lucida",13,'italic'),fg='#ffffff',bg='black',command=add\_app\_data)

#lbl\_over.bind("<Button-1>")

lbl\_over.place(x=8,y=220)

lbl\_category = Button(root,text = "Category",width=25,height='2',font=("Lucida",13,'italic'),fg='#ffffff',bg='black',command=category)

#lbl\_category.bind("<Button-1>")

lbl\_category.place(x=8,y=220+60)

lbl\_Installs = Button(root,text = "Installs",width=25,height='2',font=("Lucida",13,'italic'),fg='#ffffff',bg='black',command=install)

#lbl\_Installs.bind("<Button-1>")

lbl\_Installs.place(x=8,y=220+60+60)

lbl\_searchapp = Button(root,text = "Search App",width=25,height='2',font=("Lucida",13,'italic'),fg='#ffffff',bg='black',command=fn.searchapp)

#lbl\_searchapp.bind("<Button-1>")

lbl\_searchapp.place(x=8,y=220+60+120)

lbl\_machine = Button(root,text = "App Info",width=25,height='2',font=("Lucida",13,'italic'),fg='#ffffff',bg='black',command=app)

lbl\_machine.bind("<Button-1>")

lbl\_machine.place(x=8,y=220+60+120+60)

lbl\_review = Button(root,text = "Reviews",width=25,height='2',font=("Lucida",13,'italic'),fg='#ffffff',bg='black',command=rrev)

#lbl\_review.bind("<Button-1>")

lbl\_review.place(x=8,y=220+60+120+120)

lbl\_lastupdate = Button(root,text = "Add Reviews",width=25,height='2',font=("Lucida",13,'italic'),fg='#ffffff',bg='black',command=add\_rev)

lbl\_lastupdate.bind("<Button-1>")

lbl\_lastupdate.place(x=8,y=220+60+120+180)

#======================================right canvas============================

rcanvas=Canvas(width = 295,height=500,bg='#102131')

rcanvas.place(x=1060,y=180)

Button(rcanvas,text = "Update The Installs\n Per Category",width=25,height='2',font=("Lucida",13,'italic'),fg='#ffffff',bg='black',command=Update\_cat).place(x=35,y=220)

#======================================bottom canvas============================

bottom=Canvas(width = 1190,height=500,bg='#102131')

bottom.place(x=0,y=682)

ball=bottom.create\_oval(4,4,30,30,fill='#ffffff')

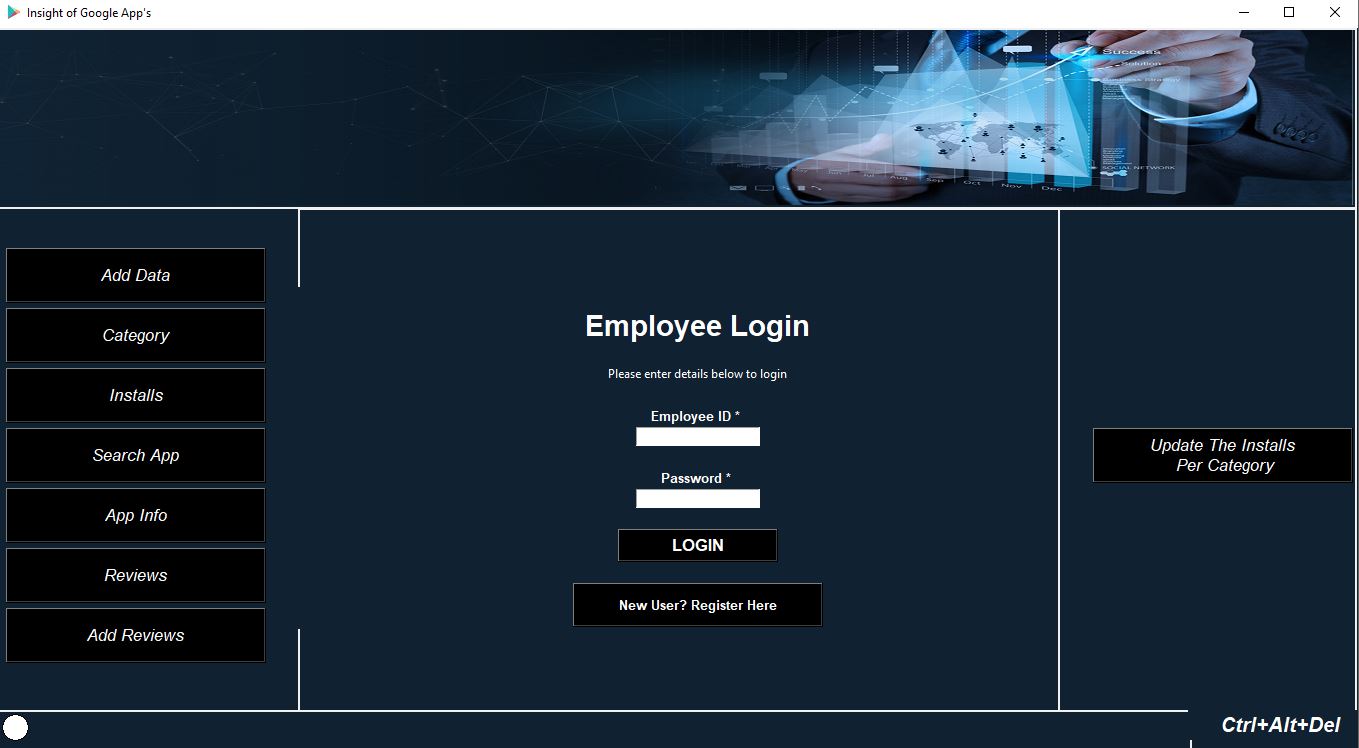
#==================================================group name==============================================

name=Label(root,text="Ctrl+Alt+Del",width=15,height=1,font=("Helvetica",15,'bold','italic'),fg='#ffffff',bg='#102131')

name.place(x=1190,y=682)

root.mainloop()

**Login Form**



def login():

global mcanvas

global username\_verify

global password\_verify

mcanvas.delete("all")

val=Label(mcanvas,width=400,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(400,250, window=val)

# df= pd.read\_csv("C:\\Users\\Harsh\\Desktop\\internship\\googleplaystore-App-data.csv")

username\_verify = StringVar()

password\_verify = StringVar()

Label(val, text="Employee Login", width="400", height="2", font=("Lucida", 22, 'bold'), fg='white', bg='#102131').pack()

Label(val, text="", bg='#102131',width='100', height='17').place(x=45, y=120) # blue background in middle of window

Label(val, text="Please enter details below to login", bg='#102131', fg='white').pack()

Label(val, text="", bg='#102131').pack() # for leaving a space in between

Label(val, text="Employee ID \* ", font=("Open Sans", 10, 'bold'), bg='#102131', fg='white').pack()

Entry(val, textvar=username\_verify).pack()

Label(val, text="", bg='#102131').pack() # for leaving a space in between

Label(val, text="Password \* ", font=("Open Sans", 10, 'bold'), bg='#102131', fg='white').pack()

Entry(val, textvar=password\_verify, show="\*").pack()

Label(val, text="", bg='#102131').pack() # for leaving a space in between

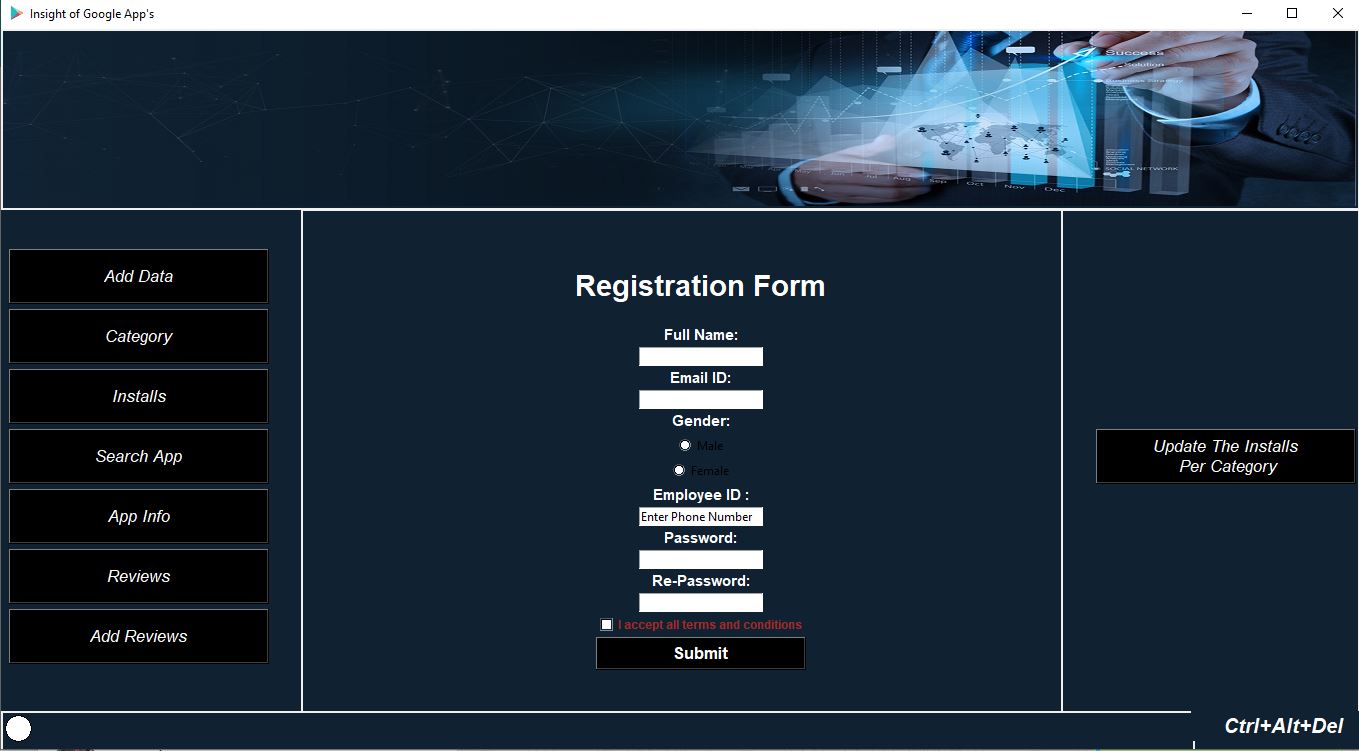
Button(val, text="LOGIN", bg="black", width=15, height=1, font=("Open Sans", 13, 'bold'), fg='white',command=login\_verify).pack()

Label(val, text="", bg='#102131').pack() # for leaving a space in between

Button(val, text="New User? Register Here", height="2", width="30", bg='black', font=("Open Sans", 10, 'bold'), fg='white',command=register).pack()

mcanvas.update()

**Register Form:**



def register():

global mcanvas

global fullname

global email

global password

global repassword

global phone

global gender

global tnc

global mcanvas

mcanvas.delete("all")

val=Label(mcanvas,width=400,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(400,250, window=val)

fullname = StringVar()

email = StringVar()

password = StringVar()

repassword = StringVar()

phone= StringVar()

gender = IntVar()

tnc = IntVar()

# configuring the window

Label(val, text="Registration Form", width='32', height="2", font=("Lucida", 22, 'bold'), fg='white', bg='#102131').pack()

Label(val, text="", bg='#102131', width='100', height='20').place(x=45, y=120)

Label(val, text="Full Name:", font=("Open Sans", 11, 'bold'), fg='white', bg='#102131', anchor=W).pack()

Entry(val, textvar=fullname).pack()

Label(val, text="Email ID:", font=("Open Sans", 11, 'bold'), fg='white', bg='#102131', anchor=W).pack()

Entry(val, textvar=email).pack()

Label(val, text="Gender:", font=("Open Sans", 11, 'bold'), fg='white', bg='#102131', anchor=W).pack()

Radiobutton(val, text="Male", variable=gender, value=1, bg='#102131').pack()

Radiobutton(val, text="Female", variable=gender, value=2, bg='#102131').pack()

Label(val, text="Employee ID :", font=("Open Sans", 11, 'bold'), fg='white', bg='#102131', anchor=W).pack()

Entry(val, textvar=phone).pack()

phone.set('Enter Phone Number')

# droplist = OptionMenu(val, university, \*list1)

# droplist.config(width=17)

# university.set('--select your university--')

# droplist.pack()

Label(val, text="Password:", font=("Open Sans", 11, 'bold'), fg='white', bg='#102131', anchor=W).pack()

Entry(val, textvar=password, show="\*").pack()

Label(val, text="Re-Password:", font=("Open Sans", 11, 'bold'), fg='white', bg='#102131', anchor=W).pack()

entry\_4 = Entry(val, textvar=repassword, show="\*")

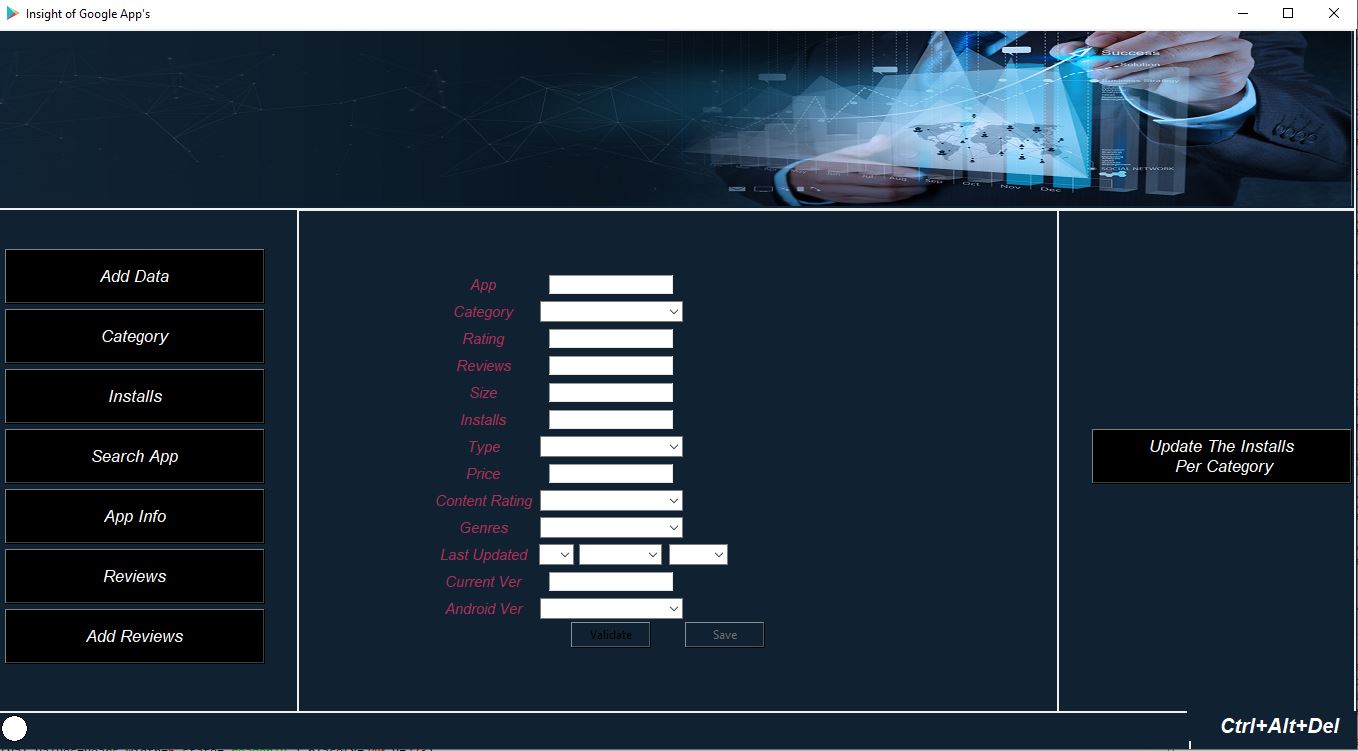
entry\_4.pack()

Checkbutton(val, text="I accept all terms and conditions", variable=tnc, bg='#102131', font=("Open Sans", 9, 'bold'), fg='brown').pack()

Button(val, text='Submit', width=20, font=("Open Sans", 13, 'bold'), bg='black', fg='white',command=register\_user).pack()

mcanvas.update()

**The Frame To Add the Data Which can be accessed only after login**



def add\_app\_data():

global mcanvas,screen,df,data

dates=[]

month=['January', 'February', 'March', 'April','May','June','July','August','September', 'October', 'November','December']

years=[]

for i in range(1,32):

dates.append(i)

for i in range(2010,2020):

years.append(i)

data=pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

mcanvas.delete("all")

val=Label(mcanvas,width=600,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(300,250, window=val)

header=data.columns.tolist()

category= list(OrderedDict.fromkeys(data['Category']))

content=list(OrderedDict.fromkeys(data['Content Rating']))

genre=list(OrderedDict.fromkeys(data['Genres']))

txt=[]

datecombo=[]

for i in range(1,14):

Label(val,text=header[i-1],width=11,font=("Lucida",11,'italic'),fg='#ab3059',bg='#102131').grid(row=i,column=0,padx=2,pady=2)

for i in range(1,14):

if i!=2 and i!=10 and i!=9 and i!=7 and i!=11 and i!=13:

txtfield=tk.Entry(val,bg="white")

txt.append(txtfield)

txtfield.grid(row=i,column=1,padx=2,pady=2)

elif i==2:

combo=ttk.Combobox(val,values=category)

txt.append(combo)

combo.grid(row=2,column=1,padx=2,pady=2)

elif i==9:

combo=ttk.Combobox(val,values=content,state="readonly")

txt.append(combo)

combo.grid(row=9,column=1,padx=2,pady=2)

elif i==10:

combo=ttk.Combobox(val,values=genre,state="readonly")

txt.append(combo)

combo.grid(row=10,column=1,padx=2,pady=2)

elif i==7:

combo=ttk.Combobox(val,values=['Free','Paid'],state="readonly")

txt.append(combo)

combo.grid(row=7,column=1,padx=2,pady=2)

elif i==11:

combo=ttk.Combobox(val,values=dates,width=2,state="readonly").place(x=110,y=273)

datecombo.append(combo)

combo=ttk.Combobox(val,values=month,width=10,state="readonly").place(x=150,y=273)

datecombo.append(combo)

combo=ttk.Combobox(val,values=years,width=6,state="readonly").place(x=240,y=273)

datecombo.append(combo)

elif i==13:

combo=ttk.Combobox(val,values=list(data['Android Ver'].unique()),state="readonly")

txt.append(combo)

combo.grid(row=13,column=1,padx=2,pady=2)

btn\_save=tk.Button(val,text='Save',state="disabled",width=10,bg="#102131",command=lambda:saveing(txt,btn\_save,'C:\\InternshipFinal\\App-data.csv',datecombo))

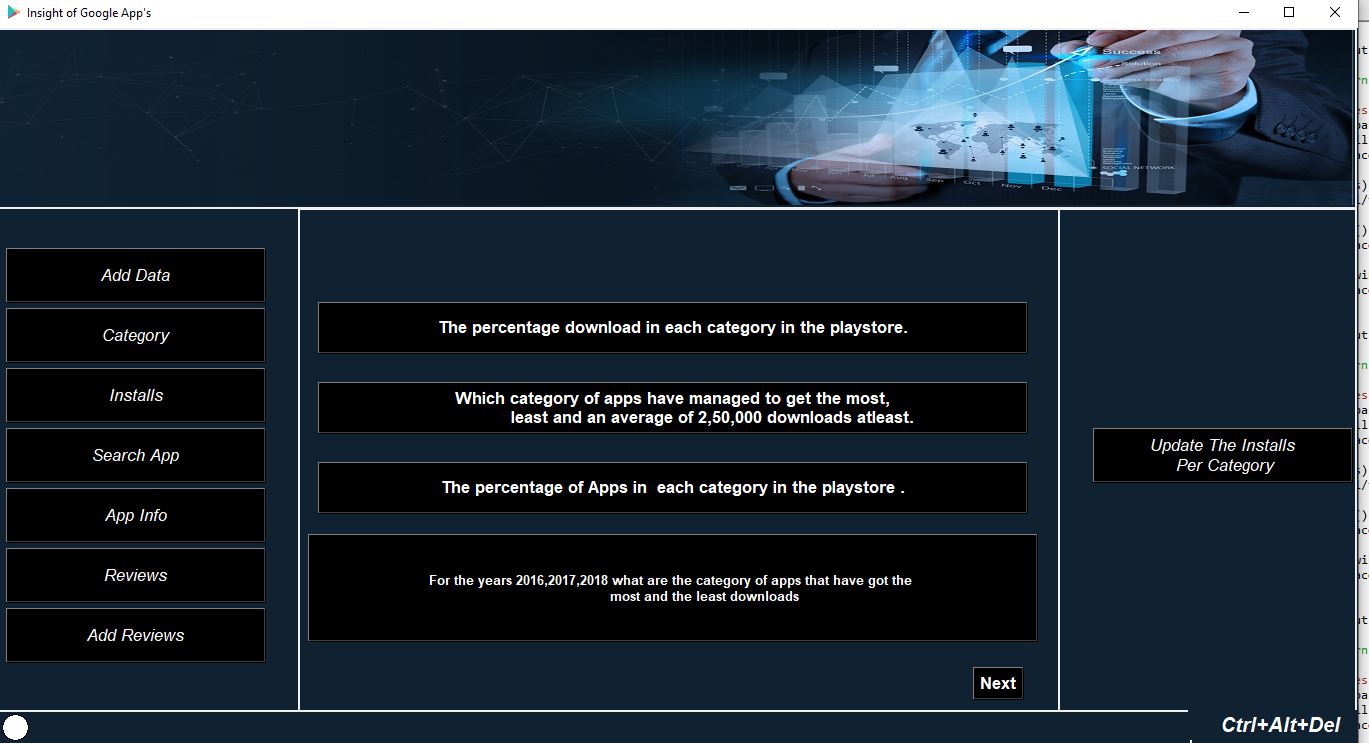
btn\_validate=tk.Button(val,text='Validate',width=10,bg="#102131",command=lambda:validate(txt,btn\_save,datecombo))

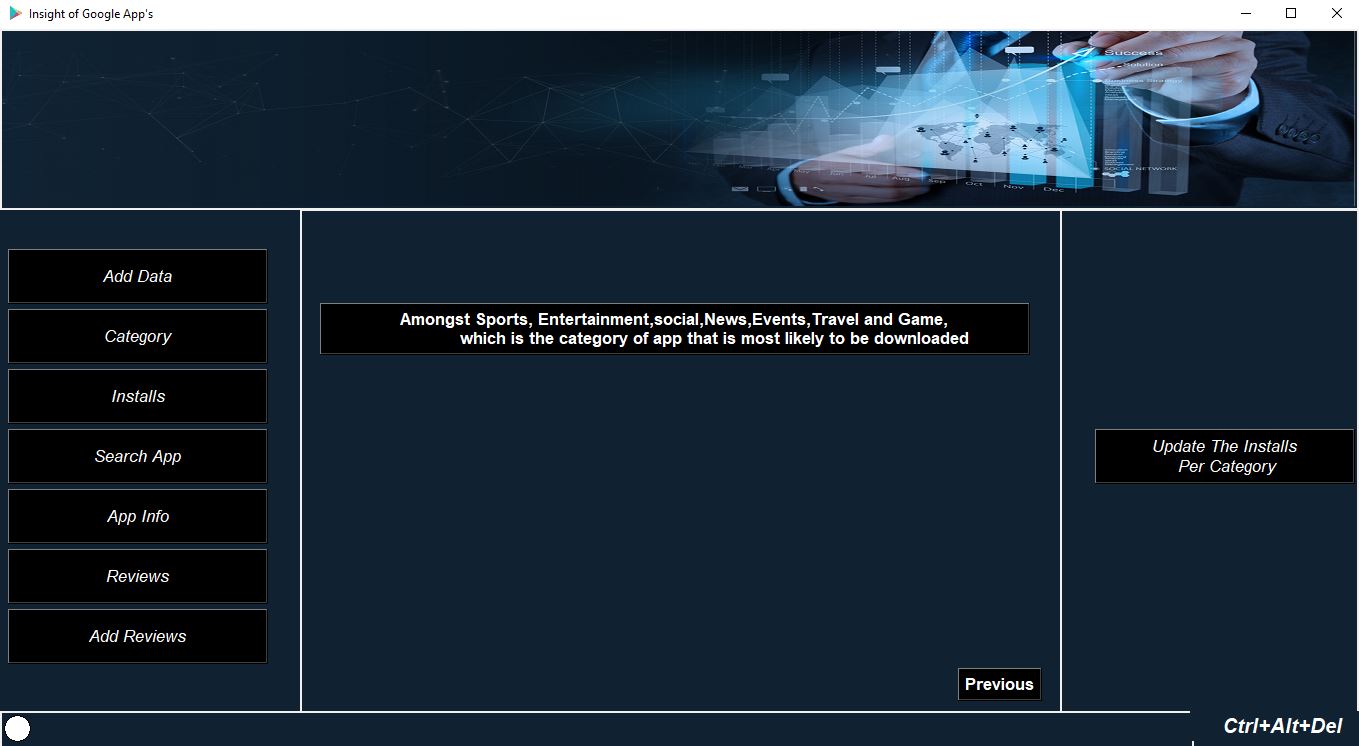
btn\_validate.grid(row=14,column=1)

btn\_save.grid(row=14,column=2)

mcanvas.create\_window()

mcanvas.update()

**Category Frame**

****

def category():

global mcanvas

mcanvas.delete("all")

# q=mcanvas.create\_rectangle(40,40,500,80,fill='black')

q1 = Button(mcanvas,text = "The percentage download in each category in the playstore.",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq1)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 120, window=q1)

q3 = Button(mcanvas,text = """Which category of apps have managed to get the most,

least and an average of 2,50,000 downloads atleast.""",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq3)

# q4.bind("<Button-1>", function\_q4)

# q4.place(x=40,y=120)

mcanvas.create\_window(375,200, window=q3)

q0 = Button(mcanvas,text = "The percentage of Apps in each category in the playstore .",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq0)

# q5.bind("<Button-1>", function\_q5)

mcanvas.create\_window(375,280, window=q0)

q6 = Button(mcanvas,text = """For the years 2016,2017,2018 what are the category of apps that have got the

most and the least downloads""",width=90,height=6,font=("Lucida",10,'bold'),fg='#ffffff',bg='black',command=fn.functq6)

# q5.bind("<Button-1>", function\_q5)

mcanvas.create\_window(375,380, window=q6)

b=Button(mcanvas, text="Next",font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=nextc1)

mcanvas.create\_window(700,475, window=b)

mcanvas.update()

def nextc1():

global mcanvas

mcanvas.delete("all")

# q=mcanvas.create\_rectangle(40,40,500,80,fill='black')

q8 = Button(mcanvas,text = """Amongst Sports, Entertainment,social,News,Events,Travel and Game,

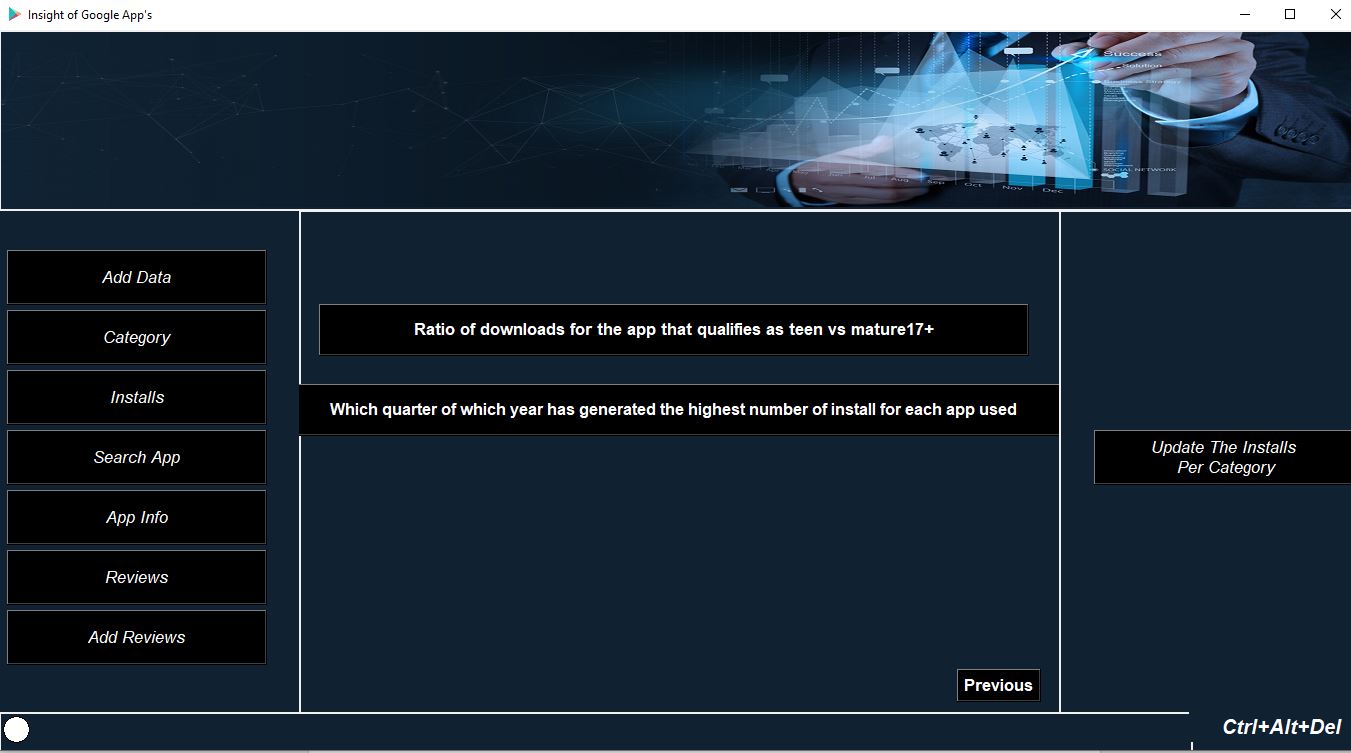
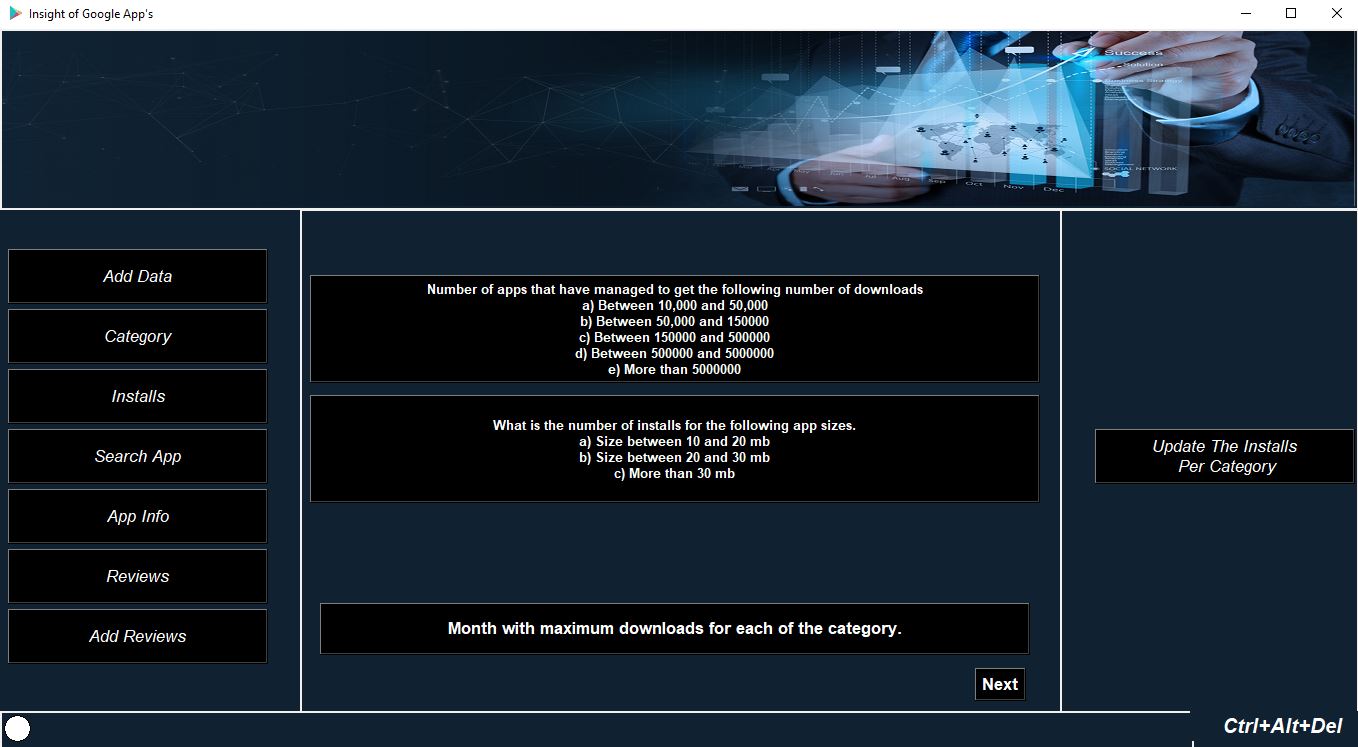
which is the category of app that is most likely to be downloaded""",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq8)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 120, window=q8)

b=Button(mcanvas, text="Previous",font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=category)

mcanvas.create\_window(700,475, window=b)

**Installs Screen **

def install():

global mcanvas

mcanvas.delete("all")

# q=mcanvas.create\_rectangle(40,40,500,80,fill='black')

q2 = Button(mcanvas,text = """Number of apps that have managed to get the following number of downloads

a) Between 10,000 and 50,000

b) Between 50,000 and 150000

c) Between 150000 and 500000

d) Between 500000 and 5000000

e) More than 5000000""",width=90,height=6,font=("Lucida",10,'bold'),fg='#ffffff',bg='black',command=fn.functq2)

# q3.bind("<Button-1>", function\_q3)

# q3.place(x=40,y=120)

mcanvas.create\_window(375,120, window=q2)

q5 = Button(mcanvas,text = """What is the number of installs for the following app sizes.

a) Size between 10 and 20 mb

b) Size between 20 and 30 mb

c) More than 30 mb""",width=90,height=6,font=("Lucida",10,'bold'),fg='#ffffff',bg='black',command=fn.functq5)

# q4.bind("<Button-1>", function\_q4)

# q4.place(x=40,y=200)

mcanvas.create\_window(375,240, window=q5)

q10\_1 = Button(mcanvas,text = "Month with maximum downloads for each of the category.",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=funct10)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 420, window=q10\_1)

b=Button(mcanvas, text="Next",font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=nexti1)

mcanvas.create\_window(700,475, window=b)

mcanvas.update()

def nexti1():

global mcanvas

mcanvas.delete("all")

# q=mcanvas.create\_rectangle(40,40,500,80,fill='black')

q10\_2 = Button(mcanvas,text = "Ratio of downloads for the app that qualifies as teen vs mature17+",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq10\_2)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 120, window=q10\_2)

q11 = Button(mcanvas,text = "Which quarter of which year has generated the highest number of install for each app used",width=78,height=2,font=("Lucida",12,'bold'),fg='#ffffff',bg='black',command=fn.functq1)

# q3.bind("<Button-1>", function\_q3)

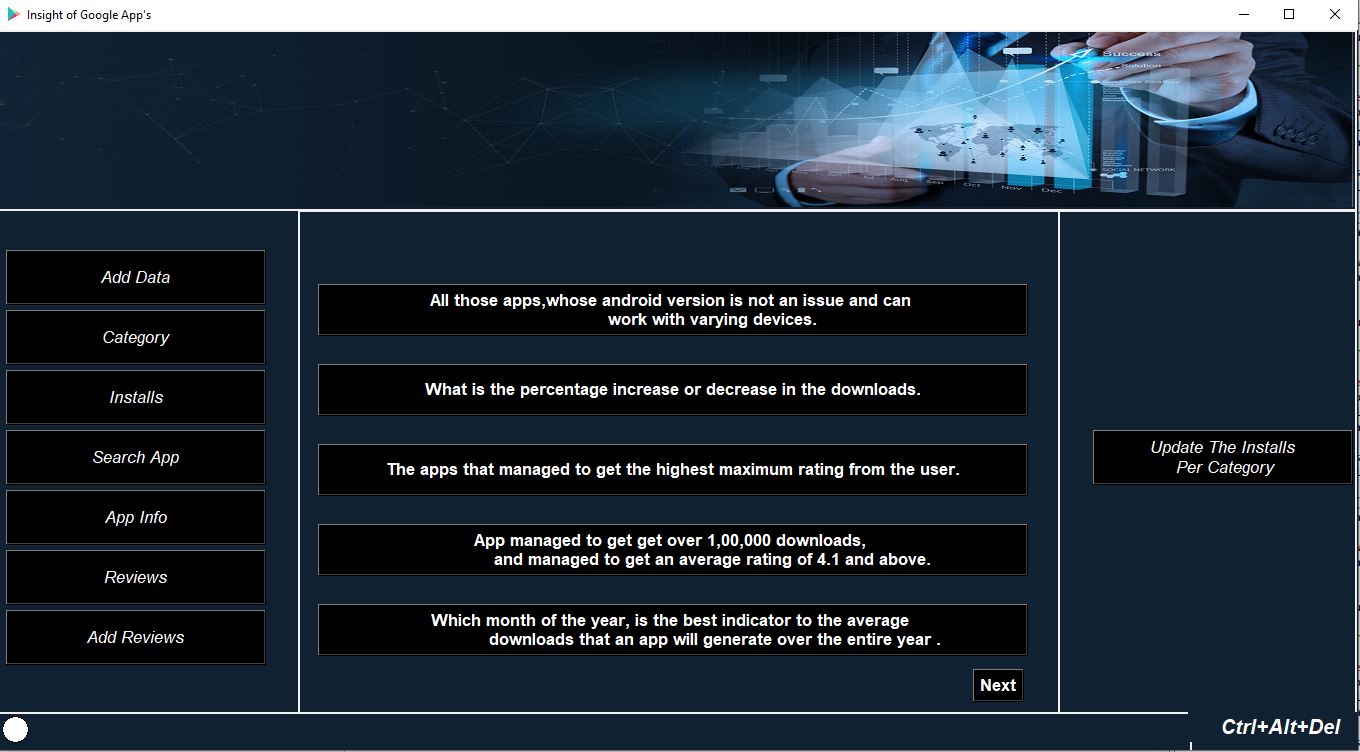
mcanvas.create\_window(375, 200, window=q11)

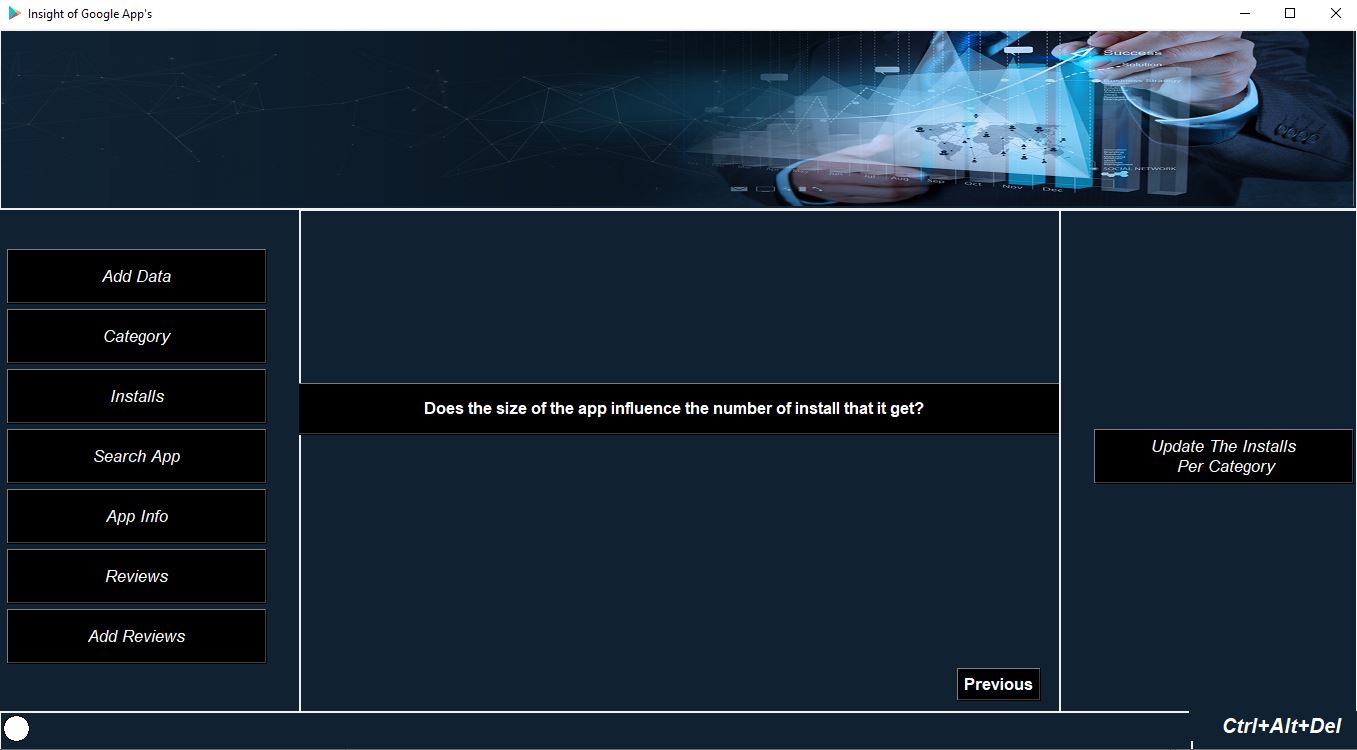
b=Button(mcanvas, text="Previous",font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=install)

mcanvas.create\_window(700,475, window=b)

mcanvas.update()

**The App Information Screen**

****

****

def app():

global mcanvas

mcanvas.delete("all")

# q=mcanvas.create\_rectangle(40,40,500,80,fill='black')

q7 = Button(mcanvas,text = """All those apps,whose android version is not an issue and can

work with varying devices.""",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq7)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 100, window=q7)

q7\_2 = Button(mcanvas,text = "What is the percentage increase or decrease in the downloads.",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq7\_2)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 180, window=q7\_2)

q4 = Button(mcanvas,text = "The apps that managed to get the highest maximum rating from the user.",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq4)

# q5.bind("<Button-1>", function\_q5)

mcanvas.create\_window(375,260, window=q4)

q9 = Button(mcanvas,text = """App managed to get get over 1,00,000 downloads,

and managed to get an average rating of 4.1 and above.""",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq9)

# q5.bind("<Button-1>", function\_q5)

mcanvas.create\_window(375,340, window=q9)

q16 = Button(mcanvas,text = """Which month of the year, is the best indicator to the average

downloads that an app will generate over the entire year .""",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=funct16)

# q5.bind("<Button-1>", function\_q5)

mcanvas.create\_window(375,420, window=q16)

b=Button(mcanvas, text="Next",font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=nexta1)

mcanvas.create\_window(700,475, window=b)

mcanvas.update()

def nexta1():

global mcanvas

mcanvas.delete("all")

# q=mcanvas.create\_rectangle(40,40,500,80,fill='black')

q17 = Button(mcanvas,text = "Does the size of the app influence the number of install that it get?",width=78,height=2,font=("Lucida",12,'bold'),fg='#ffffff',bg='black',command=fn.functq17)

# q3.bind("<Button-1>", function\_q3)

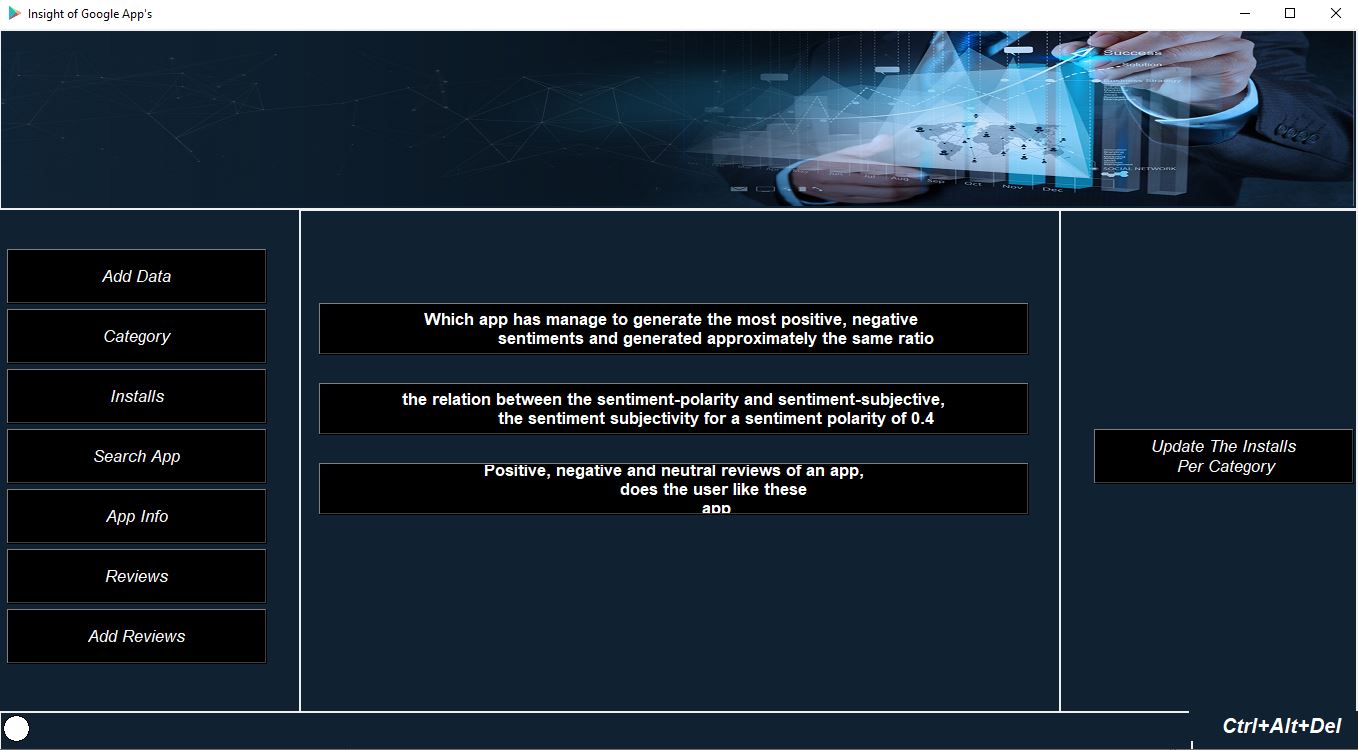
mcanvas.create\_window(375, 200, window=q17)

b=Button(mcanvas, text="Previous",font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=app)

mcanvas.create\_window(700,475, window=b)

mcanvas.update()

**The Review Screen**



def rrev():

global mcanvas

mcanvas.delete("all")

q12 = Button(mcanvas,text = """Which app has manage to generate the most positive, negative

sentiments and generated approximately the same ratio""",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=twelve)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 120, window=q12)

q13 = Button(mcanvas,text = """the relation between the sentiment-polarity and sentiment-subjective,

the sentiment subjectivity for a sentiment polarity of 0.4""",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.function\_q13)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 200, window=q13)

# q=mcanvas.create\_rectangle(40,40,500,80,fill='black')

q14\_15 = Button(mcanvas,text = """Positive, negative and neutral reviews of an app,

does the user like these

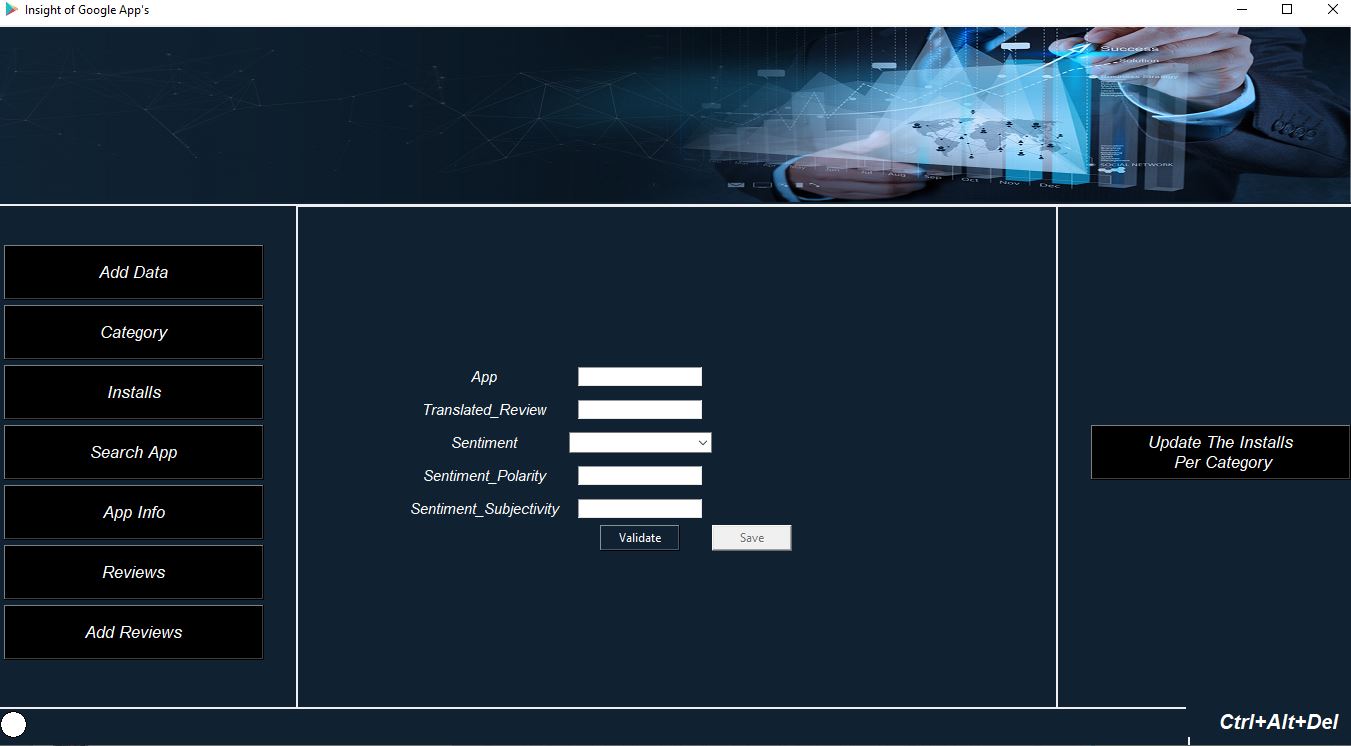
app""",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fourteen)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 280, window=q14\_15)

mcanvas.update()

**Q18) The Frame To add reviews**



def check1(x):

d=[]

for i in x:

if i.get()=='':

tk.messagebox.showwarning('Fields empty','Please provide all the fields')

return True

try:

if(isinstance(float(x[3].get()), float) and isinstance(float(x[4].get()), float)):

if x[2].get()=='Neutral':

if float(x[3].get())==0 and 1>=float(x[4].get())>=0:

d.append(False)

else:

tk.messagebox.showwarning('Neutral sentiment','Please provide a 0 in Sentiment polarity and Sentiment Subjectivity.')

return True

elif x[2].get()=='Positive':

if float(x[3].get())>0 and 1>=float(x[4].get())>=0:

d.append(False)

else:

tk.messagebox.showwarning('Positive sentiment','Please provide a positive value in Sentiment polarity and Sentiment Subjectivity.')

return True

elif x[2].get()=='Negative':

if float(x[3].get())<0 and 1>=float(x[4].get())>=0:

d.append(False)

else:

tk.messagebox.showwarning('Positive sentiment','Please provide a negative value in Sentiment polarity and non negative value in Sentiment Subjectivity.')

return True

except:

tk.messagebox.showwarning('Wrong Value','Please provide a float value in Sentiment polarity and Sentiment Subjectivity.')

return True

if set(d)==False:

return False

tk.messagebox.showinfo('Validate Succesfully','Now click on the Save Button')

def validate2(x,y):

global sample

App=x[0].get()

d=0

ap=sample['App'].unique()

for i in ap:

if i.strip()==App.strip():

msg='App named '+App+' is already present'

tk.messagebox.showerror("Error",msg)

d=1

if(check1(x)):

d=1

if d==0:

y.config(state='normal')

def add\_rev():

global screen,df,data,sample

dates=[]

sample=pd.read\_csv('C:\\InternshipFinal\\user.csv')

header2=sample.columns.tolist()

global mcanvas

val=Label(mcanvas,width=600,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(300,250, window=val)

txt=[]

datecombo=[]

month=['January', 'February', 'March', 'April','May','June','July','August','September', 'October', 'November','December']

years=[]

for i in range(1,32):

dates.append(i)

for i in range(2010,2020):

years.append(i)

mcanvas.delete("all")

val=Label(mcanvas,width=600,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(300,250, window=val)

txt2=[]

for i in range(1,6):

tk.Label(val,text=header2[i-1],width=17,font=("Lucida",11,'italic'),fg='#ffffff',bg='#102131').grid(row=i,column=0,padx=5,pady=5)

for i in range(1,6):

if i!=3:

txtfield=tk.Entry(val,bg="white")

txt2.append(txtfield)

txtfield.grid(row=i,column=2)

elif i==3:

combo=ttk.Combobox(val,values=['Positive','Negative','Neutral'],state="readonly")

txt2.append(combo)

combo.grid(row=3,column=2)

btn\_save1=tk.Button(val,text='Save',state="disabled",fg='#ffffff',width=10,command=lambda:saveing(txt2,btn\_save1,'C:\\InternshipFinal\\user.csv',''))

btn\_validate1=tk.Button(val,text='Validate',width=10,fg='#ffffff',bg="#102131",command=lambda:validate2(txt2,btn\_save1))

btn\_validate1.grid(row=7,column=2)

btn\_save1.grid(row=7,column=3)

root.mainloop()

1. **What is the percentage download in each category on the playstore?**

Code: def functq1():

global screen

screen = Tk()

big\_frame = Frame(screen,width='1010',height=750)

big\_frame.place(x=10,y=60)

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

w=1000

h=900

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

screen.geometry("%dx%d+%d+%d"%(w,h,x,y))

screen.configure(background='white')

df = pd.DataFrame()

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

list1={}

print(list1)

df['Installs']=df['Installs'].str.replace('+','')

df['Installs']=df['Installs'].str.replace(',','')

df['Installs']=df['Installs'].astype(int)

# print(sum(df['Installs']))

category ={}

sum1=[]

for i in df['Category']:

category.update({i:0})

for i in category.keys():

t2 = (df[i==(df.Category)].Installs).tolist()

sum1.append(sum(t2))

category.update({i:float(((sum(t2))/(sum(df['Installs'])))\*100)})

print(category)

list1=list(category.values())

# print(list1)

figure1 = plt.Figure(figsize=(14,9), dpi=70)

# color = cm.rainbow(np.linspace(0, 1, len(x\_label)))

#fig1, ax1 = plt.subplots()

axesObject = figure1.add\_subplot(111)

labels = ['{0} = {1:1.2f} % '.format(i,j) for i,j in zip(category.keys(),category.values())]

theme = plt.get\_cmap('hsv')

axesObject.set\_prop\_cycle("color", [theme(1. \* i / len(list1))for i in range(len(list1))])

axesObject.pie(list1,autopct='%1.2f ',startangle=90)

axesObject.set\_title("Percentage Apps in Each Category")

#ax3.xlim(0,3.0)

canvas = FigureCanvasTkAgg(figure1,big\_frame)

canvas.draw()

canvas.get\_tk\_widget().pack( fill=BOTH, expand=True)

toolbar = NavigationToolbar2Tk(canvas,big\_frame)

toolbar.update()

canvas.\_tkcanvas.pack( fill=BOTH, expand=True)

figure1.legend(labels,bbox\_to\_anchor=(0.3,1))

string="""From The Above Pie Chart,

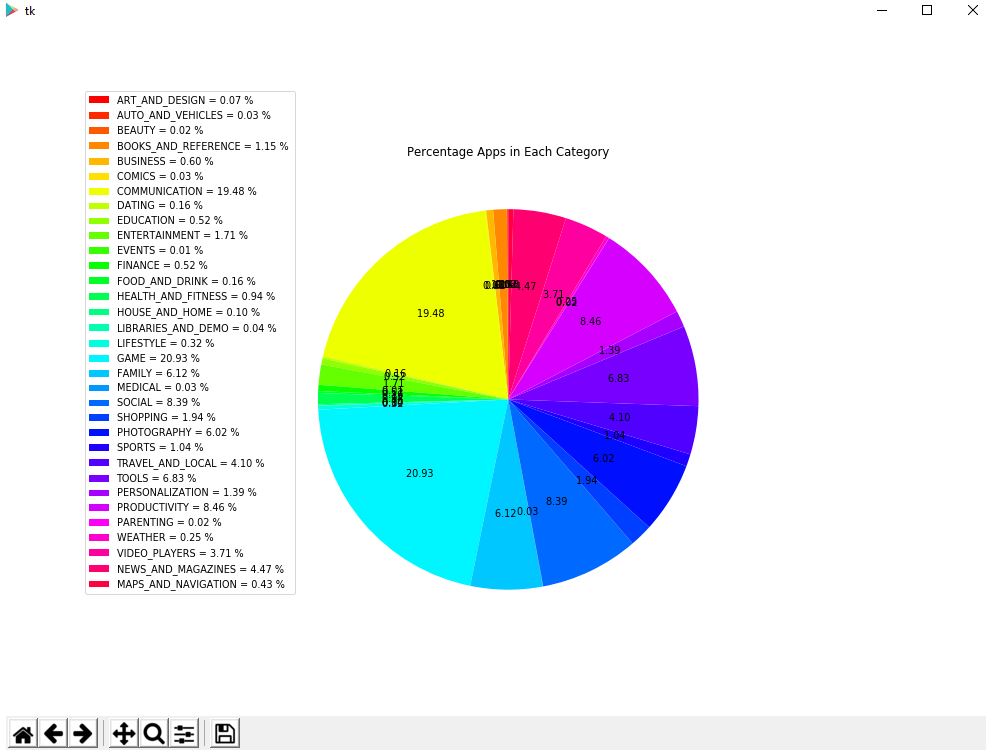
We get the percentage Apps in Each Category """

Label(screen,text=string,font=("Calibri",13,'italic'),fg='#102131',bg='white').place(x=500,y=560)

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()



1. **How many apps have managed to get the following number of downloads**

**a) Between 10,000 and 50,000**

**b) Between 50,000 and 150000**

**c) Between 150000 and 500000**

**d) Between 500000 and 5000000**

**e) More than 5000000**

Code: def functq2():

# initializing the tkinter window

global screen

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

screen.title("Apps vs Downloads") # mentioning title of the window

adjustWindow(screen) # configuring the window

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

df=df.replace(np.NaN,-1)

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

df['Installs'] = pd.to\_numeric(df['Installs'])

list2=["More than 5M","500k-5M","150k-500k","50k-150k","10k-50k"]

dict1,dict2,dict3,dict4,dict5={},{},{},{},{}

# dict6={}

dict1=(pd.value\_counts(df['Installs']>=5000000))

a1=len(df)-dict1.values[0]

dict2=(pd.value\_counts((df["Installs"]>=500000) & (df["Installs"]<5000000)))

a2=len(df)-dict2.values[0]

dict3=(pd.value\_counts((df["Installs"]>=150000) & (df["Installs"]<500000)))

a3=len(df)-dict3.values[0]

dict4=(pd.value\_counts((df["Installs"]>=50000) & (df["Installs"]<150000)))

a4=len(df)-dict4.values[0]

dict5=(pd.value\_counts((df["Installs"]>=10000) & (df["Installs"]<50000)))

a5=len(df)-dict5.values[0]

# dict6=pd.value\_counts(df["Installs"]<10000)

# a6=len(df)-dict6.values[0]

list1=[a1,a2,a3,a4,a5]

color = cm.rainbow(np.linspace(0, 2, 10))

fig=Figure(figsize=(5,4),dpi=100)

chart=fig.add\_subplot(111)

chart.bar(list2,list1,color=color)

chart.set\_ylabel("Frequency")

chart.set\_xlabel("Installs")

chart.grid()

fig.suptitle("Count-plot for Installs")

canvas = FigureCanvasTkAgg(fig, master=screen) # A tk.DrawingArea.

canvas.draw()

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

canvas.mpl\_connect("key\_press\_event", on\_key\_press)

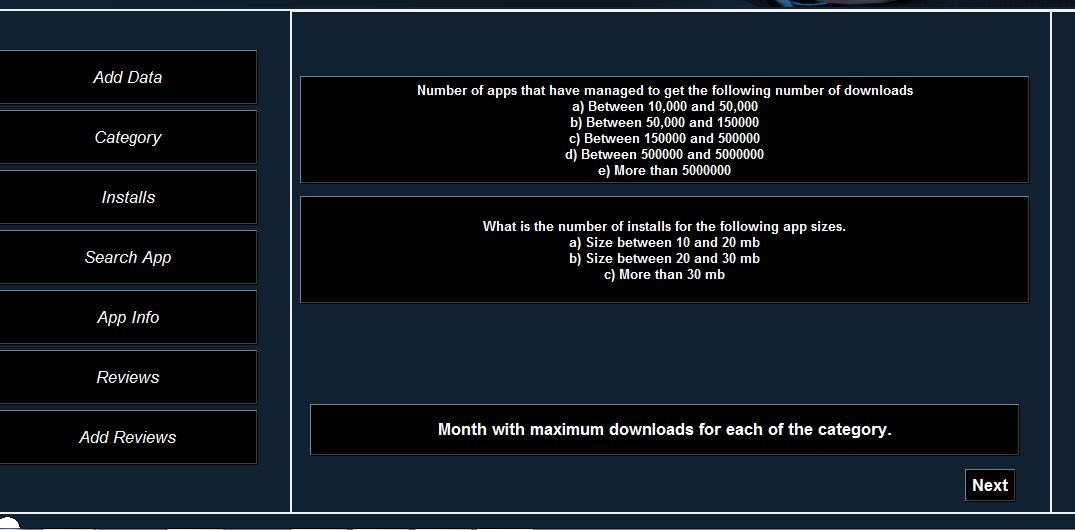
# this is necessary on Windows to prevent

# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()





1. **Which category of apps have managed to get the most,least and an average of 2,50,000 downloads atleast.**

Code:

def functq3():

global screen

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

screen.title("Apps vs Downloads")

w = 600 # width for the window size

h = 600 # height for the window size

ws = screen.winfo\_screenwidth() # width of the screen

hs = screen.winfo\_screenheight() # height of the screen

x = (ws/2) - (w/2) # calculate x and y coordinates for the Tk window

y = (hs/2) - (h/2)

screen.geometry('%dx%d+%d+%d' % (w, h, x, y)) # set the dimensions of the screen and where it is placed

screen.resizable(False, False) # disabling the resize option for the window

screen.configure(background='white') # configuring the window

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

df=df.replace(np.NaN,0)

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

df['Installs'] = pd.to\_numeric(df['Installs'])

category=df['Category'].unique()

list1=df['Installs']

ans=[]

count = []

for i in category:

total=0

c=0

for j in range(len(df['Category'])):

if df['Category'][j]==i:

total=total+list1[j]

c+=1

# print(total)

ans.append(total)

count.append(c)

#print(ans)

# print(count)

cat,avg = [],[]

for index in range(len(ans)):

cat.append(category[index])

avg.append(round(ans[index]/count[index]))

# print(avg)

# print(cat)

lowest = []

for index in range(len(avg)):

if avg[index]<250000:

lowest.append(category[index])

# print(lowest)

label = category

# print(label)

val = avg

color = cm.rainbow(np.linspace(0, 1, len(label)))

fig=Figure(figsize=(8,5),dpi=60)

chart=fig.add\_subplot(111)

chart.barh(label,val,color=color)

chart.set\_ylabel("Category")

chart.set\_xlabel("Average Installs")

chart.grid()

fig.suptitle("Category with Their Average Download")

canvas = FigureCanvasTkAgg(fig, master=screen) # A tk.DrawingArea.

canvas.draw()

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

canvas.mpl\_connect("key\_press\_event", on\_key\_press)

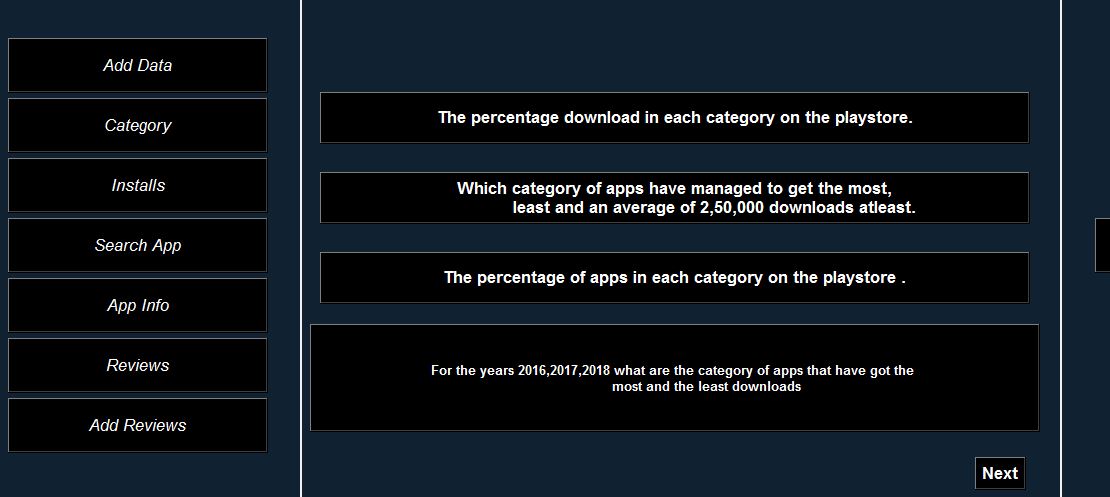
# this is necessary on Windows to prevent

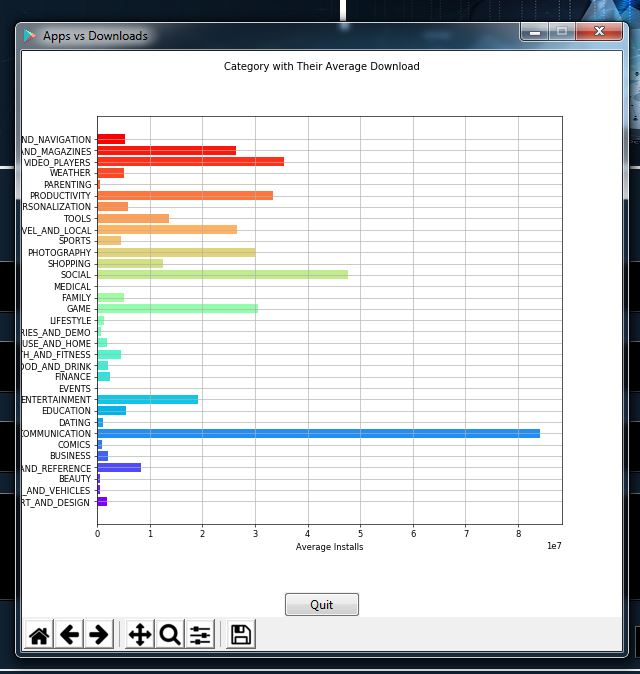
# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()





1. **Which category of apps have managed to get the highest maximum average ratings from the users.Display the result using suaitable visualization tool(s) and also update the data into the database.**

Code:

def functq4():

global screen

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

screen.title("Rating Vs Category ") # mentioning title of the window

adjustWindow(screen) # configuring the window

category ={}

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

df=df.replace(np.NaN,0)

catreview = {}

for index in range(len(df)):

if df['Category'][index] in catreview:

catreview[df['Category'][index]][0]+=df['Rating'][index]

catreview[df['Category'][index]][1]+=1

# rating+=df['Rating'][index]

else:

catreview[df['Category'][index]]=[df['Rating'][index],1]

# rating+=df['Rating'][index]

total=0

count=0

for i in df['Rating']:

total+=i

count+=1

avg= total/count

y=[]

x=[]

for i in catreview:

if catreview[i][0]/catreview[i][1]>=avg:

avgcat = (catreview[i][0]/catreview[i][1])

x.append(i)

y.append(float(avgcat))

# print(y)

# print(x)

color = cm.rainbow(np.linspace(0, 2, 15))

figure3 = plt.Figure(figsize=(5,4), dpi=80)

ax3 = figure3.add\_subplot(111)

ax3.scatter(y,x,color=color)

scatter3 = FigureCanvasTkAgg(figure3, screen)

scatter3.get\_tk\_widget().place(x=10,y=0)

ax3.grid()

ax3.set\_xlabel("RATING")

ax3.set\_ylabel("CATEGORY")

ax3.set\_title('CATEGORIES WITH HIGHEST MAXIMUM AVERAGE RATING')

canvas = FigureCanvasTkAgg(figure3, master=screen) # A tk.DrawingArea.

canvas.draw()

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

# canvas.mpl\_connect("key\_press\_event", on\_key\_press)

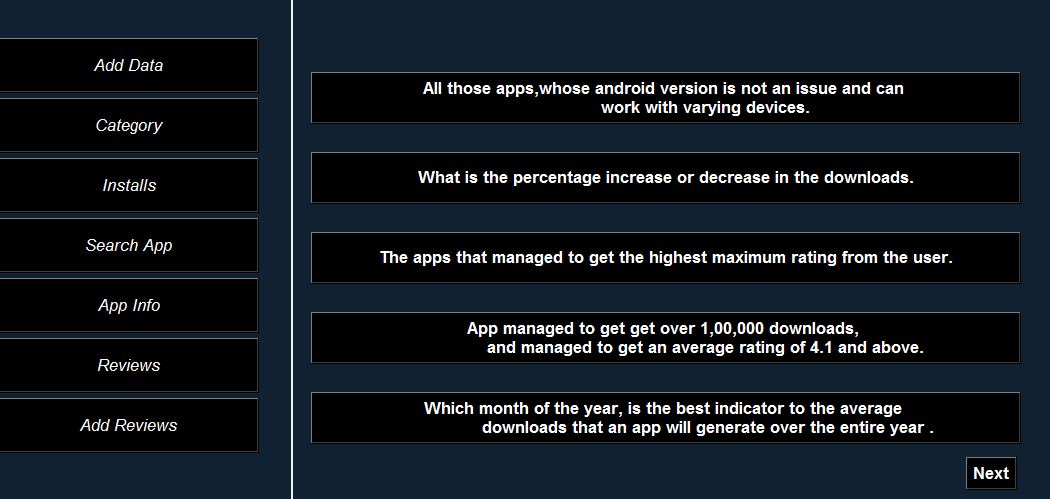
# this is necessary on Windows to prevent

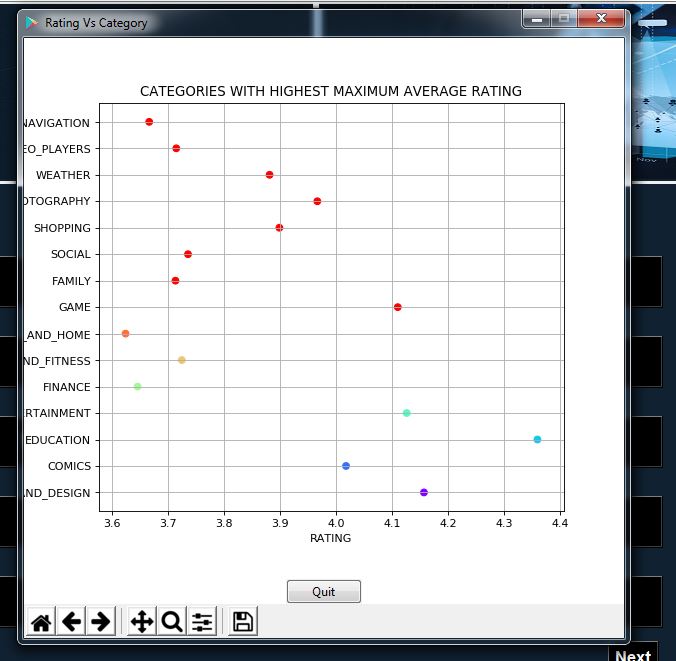
# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()





**5) What is the number of installs for the following app sizes.**

**a) Size between 10 and 20 mb**

**b) Size between 20 and 30 mb**

**c) More than 30 mb**

code:

def functq5():

global screen

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

w = 600 # width for the window size

h = 600 # height for the window size

ws = screen.winfo\_screenwidth() # width of the screen

hs = screen.winfo\_screenheight() # height of the screen

x = (ws/2) - (w/2) # calculate x and y coordinates for the Tk window

y = (hs/2) - (h/2)

screen.geometry('%dx%d+%d+%d' % (w, h, x, y)) # set the dimensions of the screen and where it is placed

screen.resizable(False, False) # disabling the resize option for the window

screen.configure(background='white') # configuring the window

df= pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

list2=['More than 30 mb','20-30 mb','10-20 mb']

df['Size'] = df['Size'].map(lambda x: x.rstrip('M'))

df['Size'] = df['Size'].map(lambda x: str(round((float(x.rstrip('k'))/1024), 1)) if x[-1]=='k' else x)

df['Size'] = df['Size'].map(lambda x: np.nan if x.startswith('Varies') else x)

df['Size']=df['Size'].replace(np.NaN,-999)

df['Size']=df['Size'].astype(float)

#print(df['Category'].unique())

#print(df['Size'])

df['Installs']=df['Installs'].str.replace('+','')

df['Installs']=df['Installs'].str.replace(',','')

df['Installs']=df['Installs'].astype(int)

dict1,dict2,dict3,dict4,dict5,dict6={},{},{},{},{},{}

a,b,c=[],[],[]

for i in range(len(df)):

if df["Size"][i]>=30:

a.append(df['Installs'][i])

elif 20<=df["Size"][i]<30:

b.append(df['Installs'][i])

elif 10<=df["Size"][i]<20:

c.append(df['Installs'][i])

a2=(sum(b))

a3=(sum(c))

a1=(sum(a))

# dict1=(pd.value\_counts(df["Size"]>=30))

# a1=len(df)-dict1.values[0]

# print(a1)

# dict2=(pd.value\_counts((df["Size"]>=20) & (df["Size"]<30)))

# a2=len(df)-dict2.values[0]

# print(a2)

# dict3=(pd.value\_counts((df["Size"]>=10) & (df["Size"]<20)))

# a3=len(df)-dict3.values[0]

# print(a3)

#dict4=(pd.value\_counts((df["Size"]<10)))

#a4=len(df)-dict4.values[0]

#print(a4)

list1=[a1,a2,a3]

print(list1)

# plt.bar(list2,list1 , color='green')

# plt.title("mb vs app")

# plt.xlabel("Downloads")

# plt.ylabel("App")

# plt.xticks(rotation=90)

color = cm.rainbow(np.linspace(0, 2, 10))

fig=Figure(figsize=(5,4),dpi=100)

chart=fig.add\_subplot(111)

chart.bar(list2,list1,color=color)

chart.set\_ylabel("No of Installs")

chart.set\_xlabel("Sizes")

chart.grid()

fig.suptitle("No. of Installs Vs Size")

canvas = FigureCanvasTkAgg(fig, master=screen) # A tk.DrawingArea.

canvas.draw()

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

canvas.mpl\_connect("key\_press\_event", on\_key\_press)

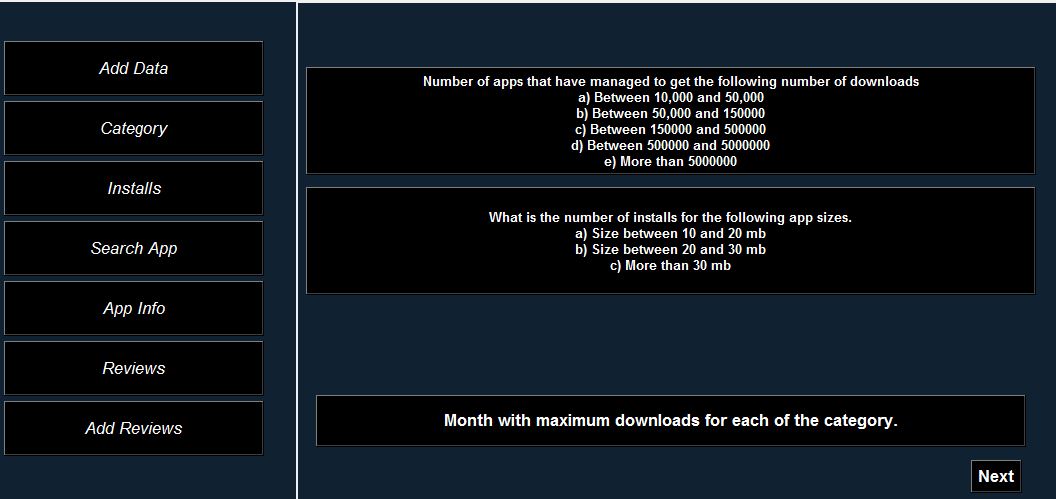
# this is necessary on Windows to prevent

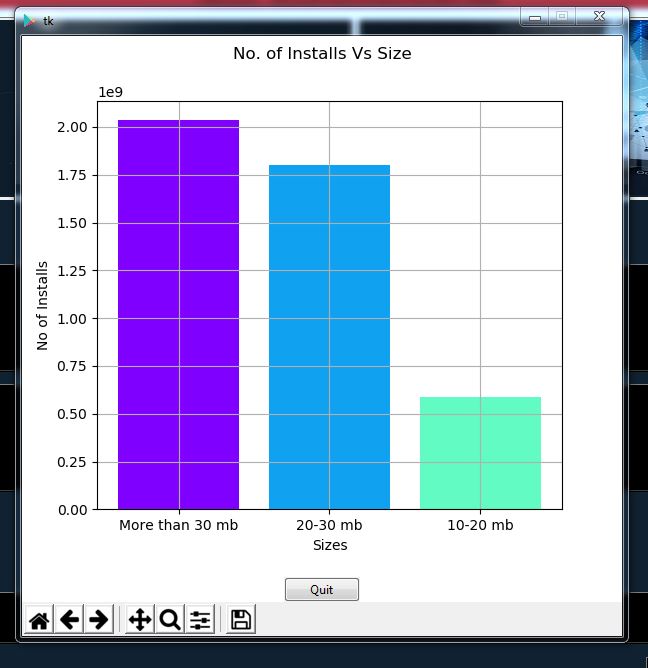
# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()





1. **For the years 2016,2017,2018 what are the category of apps that have got the most and the least downloads**

Code:

def functq6():

global screen

screen=Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

w=700

h=600

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

screen.geometry("%dx%d+%d+%d"%(w,h,x,y))

screen.configure(background='white')

# big\_frame = tk.Frame(root,bg='white',width='700',height=550,bd=4,relief=RIDGE)

# big\_frame.place(x=10,y=60)

# adjustWindow(root) # configuring the window

# Label(screen,text="").pack()

#

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

#print(df.head(5))

#df.drop(9148,axis=0, inplace=True)

#df.drop(10472,axis=0,inplace=True)

# Data cleaning for "Installs" column

#print(df['Installs'].head(5))

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

#print(df['Installs'].head(5))

df['Installs'] = pd.to\_numeric(df['Installs'])

d = pd.DatetimeIndex(df['Last Updated'])

df['year'] = d.year

df['month'] = d.month

#print((df['year'][5]))

#6) For the years 2016,2017,2018 what are the category of apps that have got the most and the least downloads. What is the percentage increase or decrease that the

dict\_2016 = {}

dict\_2017 = {}

dict\_2018 = {}

Category = []

for cat in df['Category'].unique():

Category.append(cat)

dict\_2016[cat]=0

dict\_2017[cat]=0

dict\_2018[cat]=0

#print(Category)

for index in range(len(df)):

if df['year'][index]==2016:

dict\_2016[df['Category'][index]] += df['Installs'][index]

if df['year'][index]==2017:

dict\_2017[df['Category'][index]] += df['Installs'][index]

if df['year'][index]==2018:

dict\_2018[df['Category'][index]] += df['Installs'][index]

#print(len(dict\_2016))

#print(len(dict\_2017))

#print(len(dict\_2018))

#print(dict\_2016)

#print(dict\_2017)

#print(dict\_2018)

max\_2016\_install = ["",0]

max\_2017\_install = ["",0]

max\_2018\_install = ["",0]

min\_2016\_install = ["",99999999999]

min\_2017\_install = ["",99999999999]

min\_2018\_install = ["",99999999999]

for cat in dict\_2016:

if max\_2016\_install[1] < dict\_2016[cat]:

max\_2016\_install[1] = dict\_2016[cat]

max\_2016\_install[0] = cat

if max\_2017\_install[1] < dict\_2017[cat]:

max\_2017\_install[1] = dict\_2017[cat]

max\_2017\_install[0] = cat

if max\_2018\_install[1] < dict\_2018[cat]:

max\_2018\_install[1] = dict\_2018[cat]

max\_2018\_install[0] = cat

if min\_2016\_install[1] > dict\_2016[cat]:

min\_2016\_install[1] = dict\_2016[cat]

min\_2016\_install[0] = cat

if min\_2017\_install[1] > dict\_2017[cat]:

min\_2017\_install[1] = dict\_2017[cat]

min\_2017\_install[0] = cat

if min\_2018\_install[1] > dict\_2018[cat]:

min\_2018\_install[1] = dict\_2018[cat]

min\_2018\_install[0] = cat

#print(max\_2016\_install)

#print(max\_2017\_install)

#print(max\_2018\_install)

#print(min\_2016\_install)

#print(min\_2017\_install)

#print(min\_2018\_install)

max\_install = [max\_2016\_install[1],max\_2017\_install[1],max\_2018\_install[1]]

min\_install = [min\_2016\_install[1],min\_2017\_install[1],min\_2018\_install[1]]

Years = ['2016','2017','2018']

pos = np.arange(len(Years))

bar\_width = 0.3

figure2 = plt.Figure(figsize=(8,4), dpi=85)

chart = figure2.add\_subplot(111)

Max\_bar = chart.bar(Years,max\_install,bar\_width,color='blue',edgecolor='blue')

Min\_bar = chart.bar(pos+bar\_width,min\_install,bar\_width,color='red',edgecolor='red')

chart.grid()

chart.set\_ylabel("Download")

chart.set\_xlabel('Years')

figure2.suptitle('Max and Min download across 2016-17-18 years for a category',fontsize=18)

plt.legend(['max','min'],loc=10)

max\_month = [max\_2016\_install[0],max\_2017\_install[0],max\_2018\_install[0]]

min\_month = [min\_2016\_install[0],min\_2017\_install[0],min\_2018\_install[0]]

for idx,rect in enumerate(Max\_bar):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,max\_month[idx],ha='center', va='bottom', rotation=0)

for idx,rect in enumerate(Min\_bar):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,min\_month[idx],ha='center', va='bottom', rotation=0)

canvas = FigureCanvasTkAgg(figure2, master=screen)

canvas.get\_tk\_widget().pack()

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

canvas.mpl\_connect("key\_press\_event", on\_key\_press)

# this is necessary on Windows to prevent

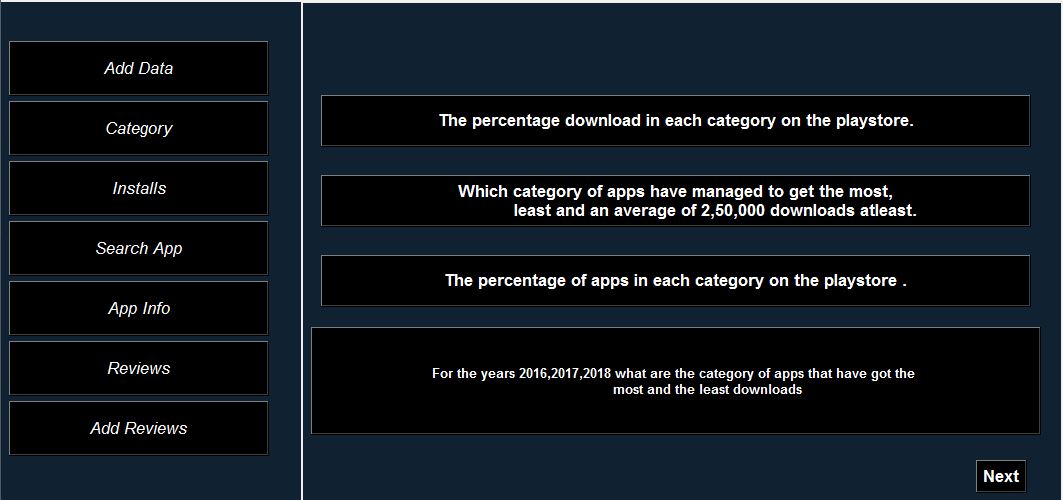
# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

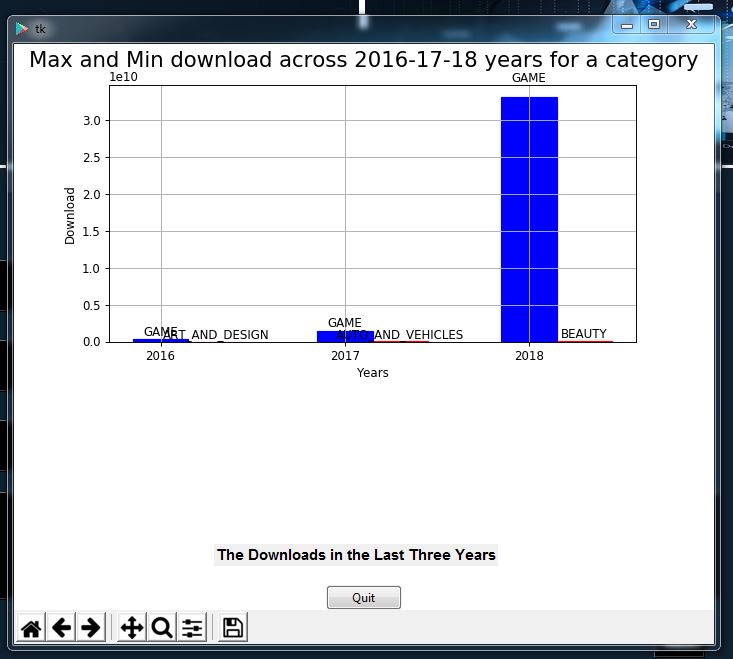
Label(screen,text="The Downloads in the Last Three Years",font=("Helvetica",11,'bold') ,borderwidth=2).place(x=200,y=500)

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()





**7)All those apps , whose android version is not an issue and can work with varying devices ,what is the percentage increase or decrease in the downloads**

Code:

def functq7():

global screen

screen=Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

w=720

h=600

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

screen.geometry("%dx%d+%d+%d"%(w,h,x,y))

screen.configure(background='white')

df= pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

df=df.replace(np.NaN,0)

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

df['Installs'] = pd.to\_numeric(df['Installs'])

varwith=[]

novar=[]

varcategory={}

nocat={}

for i in range(len(df['App'])):

if df['Android Ver'][i]=='Varies with device':

varwith.append(df['Installs'][i])

if df['Category'][i] in varcategory:

varcategory[df['Category'][i]]+=df['Installs'][i]

else:

varcategory[df['Category'][i]]=df['Installs'][i]

else:

novar.append(df['Installs'][i])

if df['Category'][i] in nocat:

nocat[df['Category'][i]]+=df['Installs'][i]

else:

nocat[df['Category'][i]]=df['Installs'][i]

# print(varwith)

# print(novar)

# print(varcategory)

# print(nocat)

sumvarcategory=sum(varwith)

sumnocat=sum(novar)

# print(sumvarcategory)

# print(sumnocat)

x=(len(varwith),len(novar))

# print(x)

androidver = ['Varying', 'Not varying']

figure1 = plt.Figure(figsize=(10,7), dpi=70)

color = cm.rainbow(np.linspace(0, 1, len(x)))

#fig1, ax1 = plt.subplots()

axesObject = figure1.add\_subplot(111)

# labels = ['{0} '.format(i,j) for i,j in zip(catcount.keys(),catcount.values())]

theme = plt.get\_cmap('hsv')

# axesObject.set\_prop\_cycle("color", [theme(1. \* i / len(catcount))for i in range(len(catcount))])

axesObject.pie(x,labels=androidver,autopct='%1.2f',startangle=90,colors=color,shadow=True,explode=[0.1,0])

axesObject.set\_title("Frequency of Varying Apps in Android version vs Apps in Non-varying Android Version in dataset")

#ax3.xlim(0,3.0)

# figure1.legend(labels,bbox\_to\_anchor=(0.3,1))

canvas = FigureCanvasTkAgg(figure1, master=screen) # A tk.DrawingArea.

canvas.draw()

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

canvas.mpl\_connect("key\_press\_event", on\_key\_press)

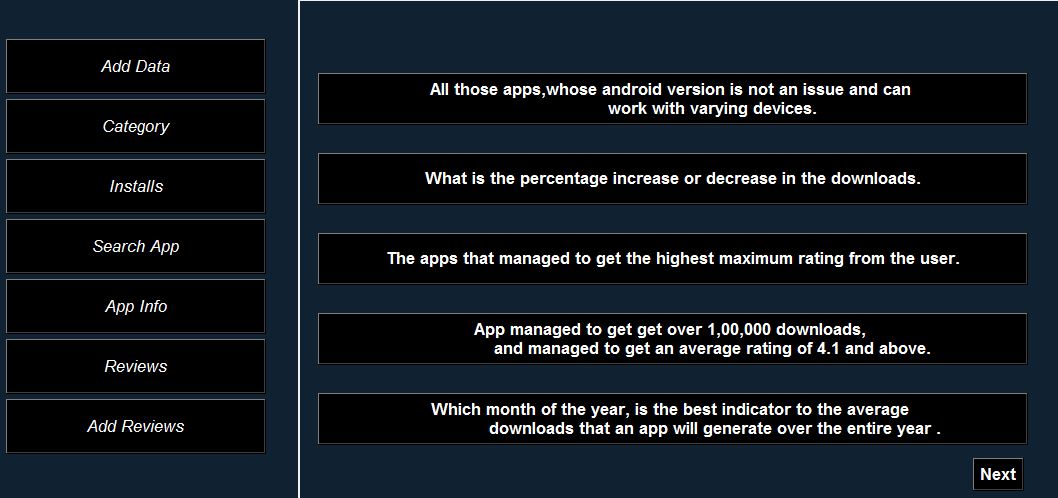
# this is necessary on Windows to prevent

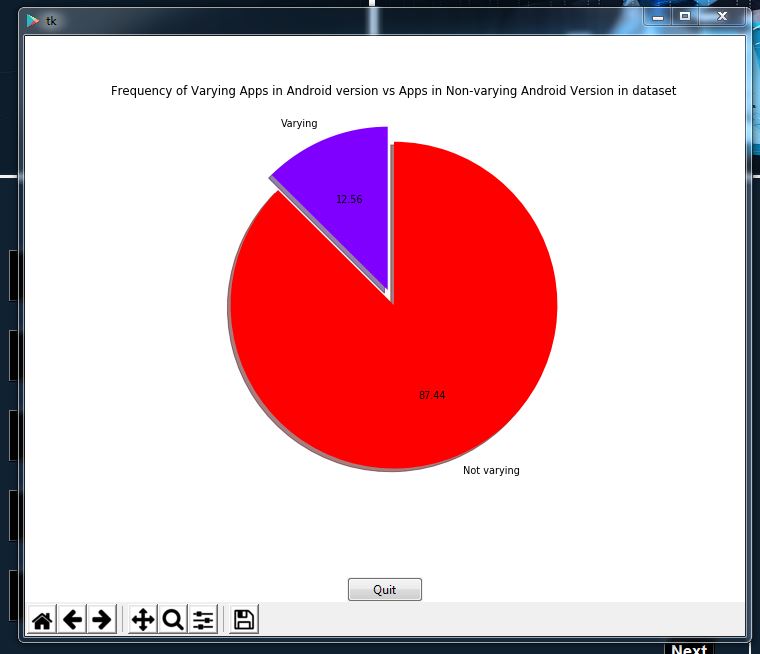
# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()





def functq7\_2():

global screen

screen = tk.Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

big\_frame = tk.Frame(screen,bg='white',width='700',height=450,bd=4,relief=RIDGE)

big\_frame.place(x=10,y=60)

w=720

h=550

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

screen.geometry("%dx%d+%d+%d"%(w,h,x,y))

screen.configure(background='white')

tk.Label(screen,text="",bg='white').pack()

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

#print(df.head(5))

#df.drop(9148,axis=0, inplace=True)

#df.drop(10472,axis=0,inplace=True)

# Data cleaning for "Installs" column

#print(df['Installs'].head(5))

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

#print(df['Installs'].head(5))

df['Installs'] = pd.to\_numeric(df['Installs'])

d = pd.DatetimeIndex(df['Last Updated'])

df['year'] = d.year

df['month'] = d.month

#print((df['year'][5]))

**#6) For the years 2016,2017,2018 what are the category of apps that have got the most and the least downloads. What is the percentage increase or decrease that the**

dict\_years = {}

for year in df['year'].unique():

dict\_years[year]=0

for index in range(len(df)):

dict\_years[df['year'][index]] += df['Installs'][index]

Years = []

list\_install = []

# for year in dict\_years:

# if year==2016 or year==2017 or year==2018:

# Years.append(str(year))

# list\_install.append(dict\_years[year])

for year in dict\_years:

Years.append((year))

list\_install.append(dict\_years[year])

# print(Years)

# print(list\_install)

new\_dict={}

for i in range(0,9):

new\_dict.update({Years[i]:list\_install[i]})

new\_dict1=dict(sorted(new\_dict.items(), key=operator.itemgetter(0),reverse=True))

keys=list(new\_dict1.keys())

values=list(new\_dict1.values())

print(keys)

print(values)

# for i in

# print(dict\_years)

x = dict\_years[2016]

y = dict\_years[2017]

z=dict\_years[2018]

per2016=1

per2017=((y-x)/(x+y))\*100

per2018=((z-y)/(y+z))\*100

# print(per2016,per2017,per2018)

Years.reverse()

list\_install.reverse()

figure2 = plt.Figure(figsize=(8,4), dpi=85)

chart = figure2.add\_subplot(111)

chart.plot(keys,values,color='blue')

#Min\_bar = chart.bar(pos+bar\_width,min\_install,bar\_width,color='pink',edgecolor='black')

chart.set\_ylabel("Years")

chart.set\_xlabel('Installs')

figure2.suptitle('Barchart on Installs on each Year ',fontsize=18)

chart.grid()

canvas = FigureCanvasTkAgg(figure2, master=big\_frame)

canvas.get\_tk\_widget().place(x=5,y=10)

String = """

% increase in 2016-17 is {:.1f}% and % increase in 2017-18 is {:.1f}%

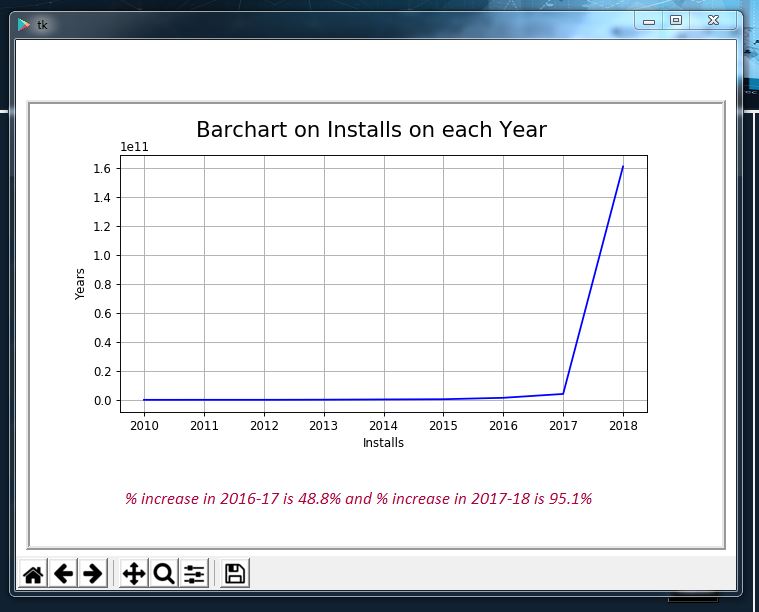
""".format(per2017,per2018)

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=40,y=360)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

screen.mainloop()



**8)Amongst sports, entertainment, social media,news,events,travel and games, which is the category of app that is most likely to be downloaded in the coming years, kindly make a prediction and back it with suitable findings. Also update the number of downloads that these categories have received into a database.**

**Code:**

def functq8():

global screen

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

screen.title("Apps to be most likely downloaded in the Upcoming Years") # mentioning title of the window

w = 600 # width for the window size

h = 500 # height for the window size

ws = screen.winfo\_screenwidth() # width of the screen

hs = screen.winfo\_screenheight() # height of the screen

x = (ws/2) - (w/2) # calculate x and y coordinates for the Tk window

y = (hs/2) - (h/2)

screen.geometry('%dx%d+%d+%d' % (w, h, x, y)) # set the dimensions of the screen and where it is placed

screen.resizable(False, False) # configuring the window

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

df=df.replace(np.NaN,0)

cat={'SPORTS':0,'ENTERTAINMENT':0,'SOCIAL':0,'NEWS\_AND\_MAGAZINES':0,'EVENTS':0,'TRAVEL\_AND\_LOCAL':0,'GAME':0}

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

#print(df['Installs'].head(5))

df['Installs'] = pd.to\_numeric(df['Installs'])

d = pd.DatetimeIndex(df['Last Updated'])

df['year'] = d.year

df['month'] = d.month

# dict\_2018={}

for i in range(len(df)):

if (df['year'][i]==2018):

if df['Category'][i] in cat:

if cat[df['Category'][i]]==0:

cat[df['Category'][i]]=1

else:

cat[df['Category'][i]]+=1

# print(cat)

color = cm.rainbow(np.linspace(0, 2, 15))

fig=Figure(figsize=(5,4),dpi=100)

chart=fig.add\_subplot(111)

k=list(cat.keys())

v=list(cat.values())

l=v.index(max(v))

print(k[l])

chart.barh(k,v,color=color)

chart.set\_ylabel("No of Installs")

chart.set\_xlabel("Categories")

chart.grid()

fig.suptitle("Count-plot for Installs")

canvas = FigureCanvasTkAgg(fig, master=screen) # A tk.DrawingArea.

canvas.draw()

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

canvas.mpl\_connect("key\_press\_event", on\_key\_press)

# this is necessary on Windows to prevent

# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

String = (f""" The Most Likely App to be downloaded in the

upcoming Years is {k[l]}""")

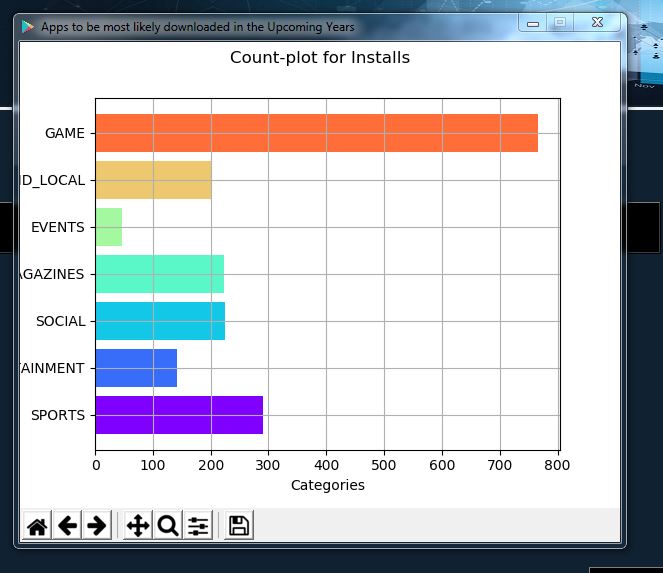
Label(screen,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=400,y=690)

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()





**9) All those apps who have managed to get over 1,00,000 downloads, have they managed to get an average rating of 4.1 and above? An we conclude something in co-relation to the number of downloads and the ratings received.**

Code:

def functq9():

global screen

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

screen = tk.Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

big\_frame = tk.Frame(screen,bg='white',width='600',height='630',bd=4)

big\_frame.place(x=50,y=60)

w=700

h=700

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

screen.geometry("%dx%d+%d+%d"%(w,h,x,y))

screen.configure(background='white')

rating = 4.1

installs = 100000

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

print(df['Rating'])

temp = []

for index in range(len(df['Rating'])):

if df['Rating'][index] >= rating:

temp.append(1)

else:

temp.append(0)

cat\_rating= pd.DataFrame(zip(temp,temp),columns=["cat\_Ratings","ignore"])

df = pd.concat([df,cat\_rating],axis=1)

df.drop("ignore",axis=1,inplace=True)

df.drop(df.index[9148], inplace=True)

# Data cleaning for "Installs" column

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

df['Installs'] = pd.to\_numeric(df['Installs'])

rating\_sum = 0

rate=[]

#1169

""" """

counter=0

for index in range(len(df)):

try:

if df['Installs'][index]>=installs:

#if df['Rating'][index]>=rating:""" """

rate.append(1)

rating\_sum+=df['Rating'][index]

counter+=1

""" """

else:

rate.append(0)

except:

#print(index)

continue

#print(len(rate))

avg\_rating = (rating\_sum/counter)

""" """

#print(df['Installs'].corr(df['Rating']))

""" """

val = "Yes" if (rating\_sum/counter)>=rating else "No"

rel = "Greater than" if val == "Yes" else "Lesser than"

fig, ax = plt.subplots(figsize=(10, 10))

l1 ='{}>='.format(installs)

l2 ='<{}'.format(installs)

size=[rate.count(1),rate.count(0)]

label = [l1,l2]

title = 'Count of {}'.format(rating)

figure1 = plt.Figure(figsize=(8,8), dpi=70)

labels1 = ['{0} = {1:1.2f} % '.format(i,j) for i,j in zip(label,size)]

#color = cm.rainbow(np.linspace(0, 1, 10))

#fig1, ax1 = plt.subplots()

ax3 = figure1.add\_subplot(111)

ax3.pie(size, labels=label,colors = ['green','cyan'], autopct='%1.1f%%', startangle=200)

ax3.set\_title(title)

ax3.legend(labels1,bbox\_to\_anchor=(1,1))

#ax3.xlim(0,3.0)

pie\_plot = FigureCanvasTkAgg(figure1, big\_frame)

pie\_plot.get\_tk\_widget().place(x=-50,y=-70)

Label(big\_frame,text="--Results--",font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=220,y=470)

String = "Average rating of all the apps who managed to get over {} download is {:.1f}".format(installs,avg\_rating)

Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=0,y=500)

String ="""{}! All those apps who have managed to get over {} downloads ,

they have to get an average rating of {:.1f} which is {} than {} """.format(val,installs,avg\_rating,rel,rating)

Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=0,y=530)

#ax3.legend(loc=0)

toolbar = NavigationToolbar2Tk(pie\_plot, screen)

toolbar.update()

pie\_plot.mpl\_connect("key\_press\_event", on\_key\_press)

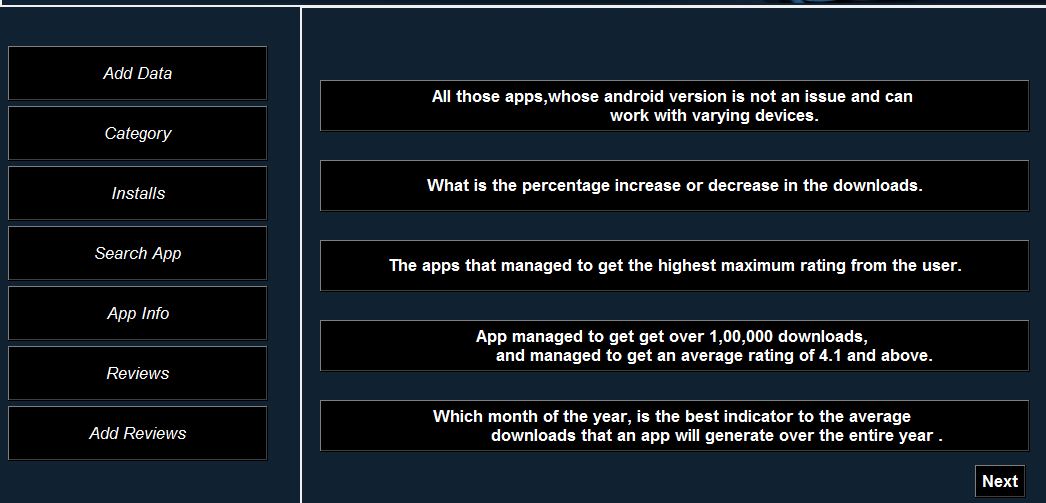
# this is necessary on Windows to prevent

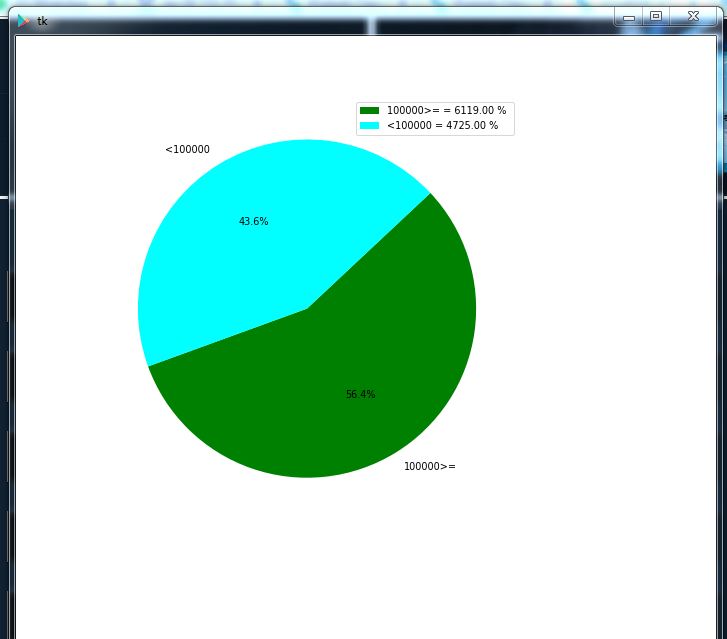
# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()





**10) Across all the years ,which month has seen the maximum downloads fr each of the category. What is the ratio of downloads for the app that qualifies as teen versus mature17+Across all the years ,which month has seen the maximum downloads fr each of the category. What is the ratio of downloads for the app that qualifies as teen versus mature17+**

**Code:**

def mont():

global root

global cat

global can

root = Tk()

root.title("Insight of Google App's")

width\_value=root.winfo\_screenwidth()

root.configure(background='Cyan') # configuring the window

height\_value=root.winfo\_screenheight()

root.geometry("%dx%d+0+0"%(width\_value, height\_value))

mcan=Canvas(root,width=800,height=700,bg='white')

mcan.place(x=300,y=70)

data=pd.read\_csv('C:\\InternshipFinal\\App-data.csv')

data=data.replace(np.nan,'Not Available')

data['Installs'] = data['Installs'].map(lambda x: x.rstrip('+'))

data['Installs'] = data['Installs'].map(lambda x: ''.join(x.split(',')))

data['Installs'] = pd.to\_numeric(data['Installs'])

d = pd.DatetimeIndex(data['Last Updated'])

data['year'] = d.year

data['month'] = d.month

mon={1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0}

for i in range(len(data)):

if data['Category'][i]== cat.get():

if data['month'][i] in mon:

if mon[data['month'][i]]==0:

mon[data['month'][i]]=data['Installs'][i]

else:

mon[data['month'][i]]+=data['Installs'][i]

x=list(mon.keys())

y=list(mon.values())

figure1 = plt.Figure(figsize=(10,8), dpi=70)

axesObject = figure1.add\_subplot(111)

axesObject.bar(x,y)

axesObject.set\_title(f"Maximum Downloads in a month for a {cat.get()}")

can= FigureCanvasTkAgg(figure1,mcan)

can.get\_tk\_widget().pack( fill=BOTH, expand=True)

toolbar = NavigationToolbar2Tk(can,mcan)

toolbar.update()

def funct10():

global cat

global mcanvas

mcanvas.delete("all")

val=Label(mcanvas,width=600,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(300,250, window=val)

data=pd.read\_csv('C:\\InternshipFinal\\App-data.csv')

data=data.replace(np.nan,'Not Available')

cat=StringVar()

choices = list(data['Category'].unique())

Label(val, text='Select Category', anchor='w').grid(row=0, column=0 ,padx=5,pady=5, sticky="w")

app=ttk.Combobox(val, width=40,state="readonly",text=cat,values=choices)

app.grid(row=0, column=1 ,padx=5,pady=5, sticky="w")

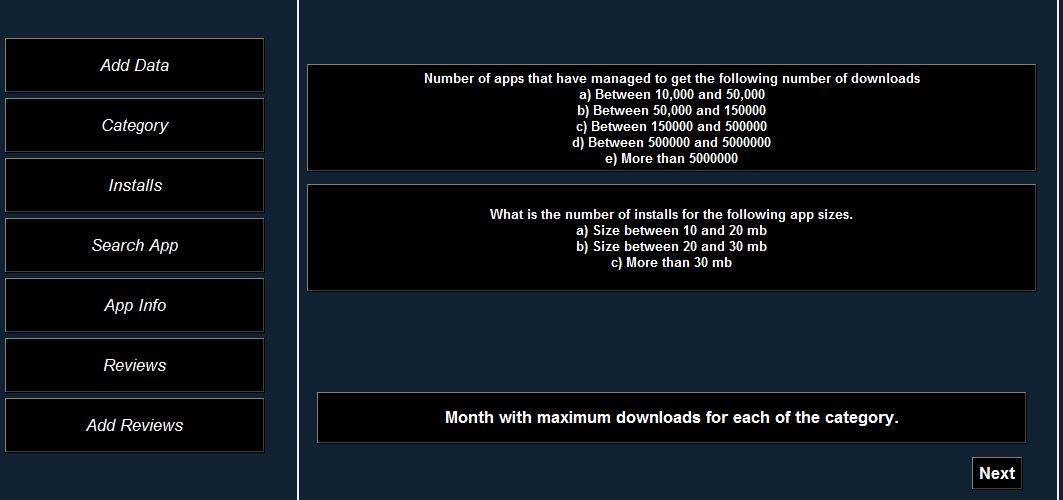
app.set("--select--")

r=Button(val,text='Search',width=12,command=mont)

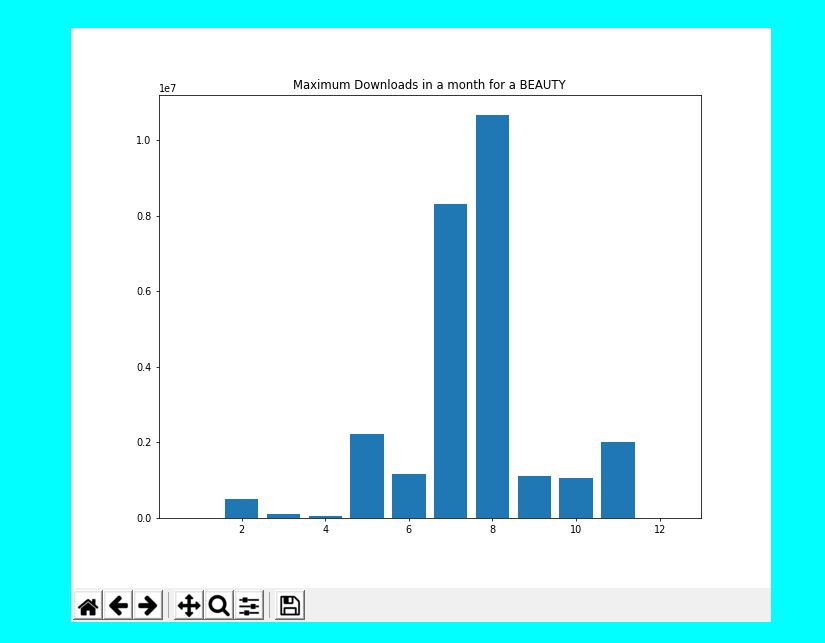
r.grid(row=0, column=3 ,padx=5,pady=5)

mcanvas.create\_window()

mcanvas.update()







def functq10\_2():

global screen

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

screen.title("Apps vs Downloads") # mentioning title of the window

adjustWindow(screen) # configuring the window

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

df=df.replace(np.NaN,0)

ratio={'Teen':0,'Mature 17+':0}

for i in range(len(df)):

if df['Content Rating'][i] in ratio:

if ratio[df['Content Rating'][i]]==0:

ratio[df['Content Rating'][i]]=1

else:

ratio[df['Content Rating'][i]]+=1

color = cm.rainbow(np.linspace(0, 2, 10))

fig=Figure(figsize=(5,4),dpi=100)

chart=fig.add\_subplot(111)

chart.bar(ratio.keys(),ratio.values(),color=color)

chart.set\_ylabel("ratio")

chart.set\_xlabel("content Rating")

chart.grid()

fig.suptitle("Ratio")

chart.legend()

canvas = FigureCanvasTkAgg(fig, master=screen) # A tk.DrawingArea.

canvas.draw()

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

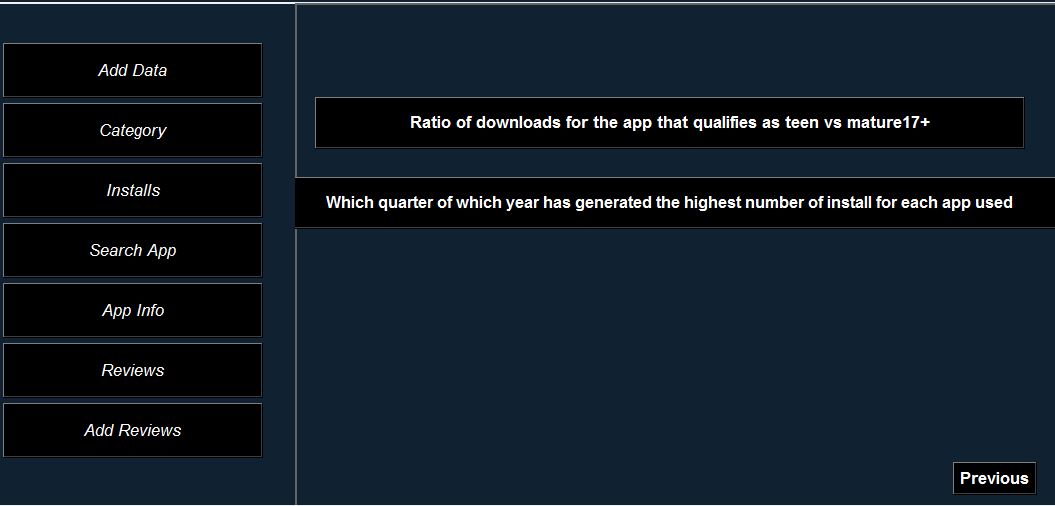
canvas.mpl\_connect("key\_press\_event", on\_key\_press)

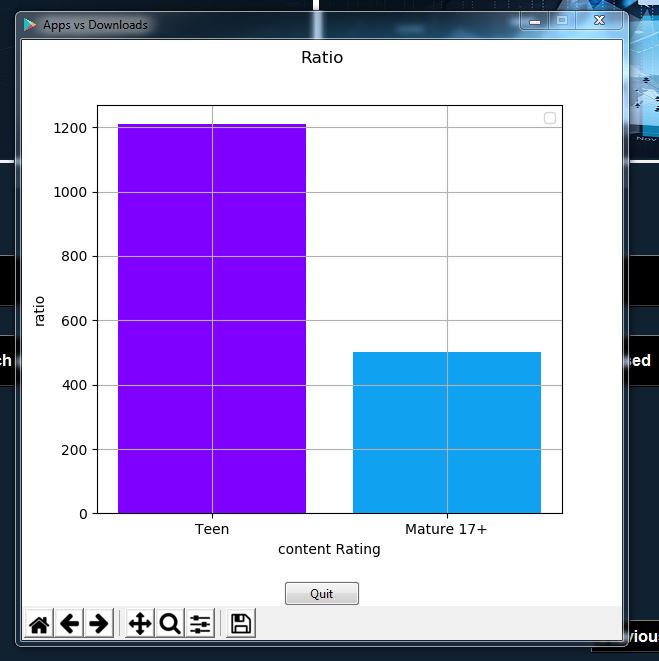
# this is necessary on Windows to prevent

# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()



**11) Which quarter of which year has generated the highest number of install for each app used in the study?**

**Code:**

def question11():

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

screen.title("Apps vs Downloads")

w = 1000 # width for the window size

h = 600 # height for the window size

ws = screen.winfo\_screenwidth() # width of the screen

hs = screen.winfo\_screenheight() # height of the screen

x = (ws/2) - (w/2) # calculate x and y coordinates for the Tk window

y = (hs/2) - (h/2)

screen.geometry('%dx%d+%d+%d' % (w, h, x, y)) # set the dimensions of the screen and where it is placed

screen.configure(background='white') # configuring the window

Years=[2010,2011,2012,2013,2014,2015,2016,2017,2018]

data = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

data['Installs'] = data['Installs'].map(lambda x: x.rstrip('+'))

data['Installs'] = data['Installs'].map(lambda x: ''.join(x.split(',')))

#print(data['Installs'].head(5))

data['Installs'] = pd.to\_numeric(data['Installs'])

d = pd.DatetimeIndex(data['Last Updated'])

data['year'] = d.year

data['month'] = d.month

list\_year=[]

for j in Years:

quar1={1:0,2:0,3:0}

quar2={4:0,5:0,6:0}

quar3={7:0,8:0,9:0}

quar4={10:0,11:0,12:0}

for i in range(len(data)):

if data['year'][i]== j:

if data['month'][i] in quar1:

quar1[data['month'][i]]+=data['Installs'][i]

elif data['month'][i] in quar2:

quar2[data['month'][i]]+=data['Installs'][i]

elif data['month'][i] in quar3:

quar3[data['month'][i]]+=data['Installs'][i]

elif data['month'][i] in quar4:

quar4[data['month'][i]]+=data['Installs'][i]

if sum(quar1.values())>sum(quar2.values()) and sum(quar1.values())>sum(quar3.values()) and sum(quar1.values())>sum(quar4.values()):

list\_year.append(quar1)

elif sum(quar2.values())>sum(quar3.values()) and sum(quar2.values())>sum(quar4.values()):

list\_year.append(quar2)

elif sum(quar3.values())>sum(quar4.values()):

list\_year.append(quar3)

else:

list\_year.append(quar4)

print(list\_year)

#dict1={}

#for i in range(len(list\_year)):

# dict1.update({Years[i]:list\_year[i]})

#print(dict1)

list10=[]

Month1,Month2,Month3 = [],[],[]

for i in range(len(list\_year)):

list2=[]

for j in (list\_year[i].keys()):

print(j)

list2.append(j)

list10.append(list2)

#print(list10)

for j in range(1):

for i in range(len(list10)):

Month1.append(list10[i][j])

for j in range(1,2):

for i in range(len(list10)):

Month2.append(list10[i][j])

for j in range(2,3):

for i in range(len(list10)):

Month3.append(list10[i][j])

print(Month1)

print("---------")

print(Month2)

print("---------")

print(Month3)

print("---------")

list1=[]

for i in range(len(list\_year)):

list2=[]

for j in (list\_year[i].values()):

print(j)

list2.append(j)

list1.append(list2)

Years = []

for i in range(2010,2019):

Years.append(str(i))

Quatmonth\_list=[]

for j in range(0,3):

list2=[]

for i in range(len(list1)):

list2.append(list1[i][j])

Quatmonth\_list.append(list2)

pos = np.arange(len(Years))

bar\_width = 0.3

figure2 = plt.Figure(figsize=(10,4), dpi=100)

chart = figure2.add\_subplot(111)

bar1 = chart.bar(Years,Quatmonth\_list[0],bar\_width,color='green',edgecolor='black')

bar2 = chart.bar(pos+bar\_width,Quatmonth\_list[1],bar\_width,color='yellow',edgecolor='black')

bar3 = chart.bar(pos+bar\_width\*2,Quatmonth\_list[2],bar\_width,color='red',edgecolor='black')

chart.set\_ylabel("Installs")

chart.set\_xlabel('Years')

figure2.suptitle('Group Barchart - Quater Month across the year',fontsize=18)

for idx,rect in enumerate(bar1):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,Month1[idx],ha='center', va='bottom', rotation=0)

for idx,rect in enumerate(bar2):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,Month2[idx],ha='center', va='bottom', rotation=0)

for idx,rect in enumerate(bar3):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,Month3[idx],ha='center', va='bottom', rotation=0)

canvas = FigureCanvasTkAgg(figure2, master=screen)

canvas.get\_tk\_widget().place(x=0,y=100)

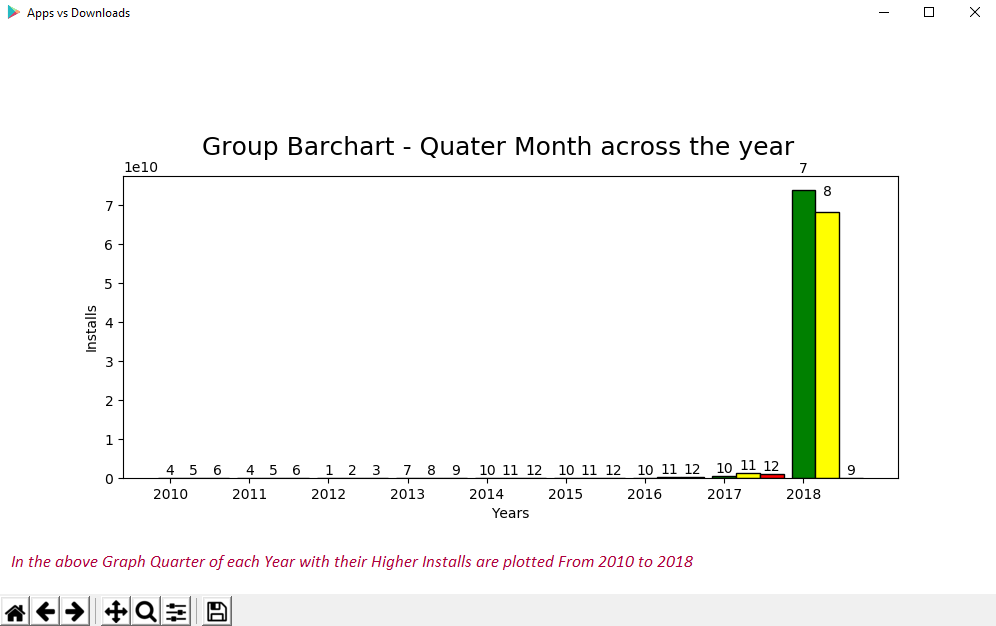
toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

String="In the above Graph Quarter of each Year with their Higher Installs are plotted From 2010 to 2018"

tk.Label(screen,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=10,y=520)

screen.mainloop()



**12) Which of all the apps given have managed to generate the most positive and negative sentiments.Also figure out the app which has generated approximately the same ratio for positive and negative sentiments.**

Code:

def sentim():

global senti

root = Tk()

root.title("Insight of Google App's")

width\_value=root.winfo\_screenwidth()

height\_value=root.winfo\_screenheight()

root.geometry("%dx%d+0+0"%(width\_value, height\_value))

root.configure(background='Cyan')

big\_frame = Frame(root)

big\_frame.pack()

canvas=[]

for i in range(1):

can=Canvas(big\_frame,width=320,height=600,bg='white')

canvas.append(can)

can.grid(row=1,column=i)

scroll1=Scrollbar(canvas[0])

positive=Listbox(canvas[0],yscrollcommand = scroll1.set,height=35,width=45,bg='light green')

scroll1.pack(side = 'right', fill = 'both')

positive.pack(side = 'left', fill = 'both')

updated\_app={}

data=pd.read\_csv('C:\\InternshipFinal\\user.csv')

data=data.replace(np.nan,'Not Available')

app={}

if senti.get()=='--select--' :

root.destroy()

for i in data['App']:

app.update({i:0})

for i in range(len(data)):

if (data['App'][i] in app) and data['Sentiment'][i]==senti.get():

if app[data['App'][i]]==0:

app[data['App'][i]]=1

else:

app[data['App'][i]]+=1

for key, value in sorted(app.items(), key=lambda item: item[1],reverse=True):

updated\_app.update({key:value})

if senti.get()!='Same Ratio':

for i in updated\_app:

positive.insert(END,i,updated\_app[i])

if senti.get()=='Same Ratio':

app={}

for i in data['App']:

app.update({i:[0,0]})

# print(app)

for i in range(len(data)):

if (data['App'][i] in app) and data['Sentiment'][i]=='Positive':

if (app[data['App'][i]][0]) == 0:

app[data['App'][i]][0]=1

else:

app[data['App'][i]][0]+=1

for i in range(len(data)):

if (data['App'][i] in app) and data['Sentiment'][i]=='Negative':

if (app[data['App'][i]][1])==0:

app[data['App'][i]][1]=1

else:

app[data['App'][i]][1]+=1

same={}

for i in app:

if app[i][0]==0 or app[i][1]==0:

continue

elif 0.75<float((app[i][0]/app[i][1]))<1.25:

if (1-app[i][0]/app[i][1])<0:

a=(1-app[i][0]/app[i][1])\*(-1)

else:

a=(1-app[i][0]/app[i][1])

same.update({i:a})

for i in same:

positive.insert(END,i)

def twelve():

global senti

mcanvas.delete("all")

val=Label(mcanvas,width=600,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(300,250, window=val)

data=pd.read\_csv('C:\\InternshipFinal\\user.csv')

data=data.replace(np.nan,'Not Available')

senti=StringVar()

choices=['Positive','Negative','Same Ratio']

Label(val, text='Select Sentiment', anchor='w').grid(row=0, column=0 ,padx=5,pady=5, sticky="w")

app=Combobox(val , width=40,state="readonly",text=senti,values=choices)

app.grid(row=0, column=1 ,padx=5,pady=5, sticky="w")

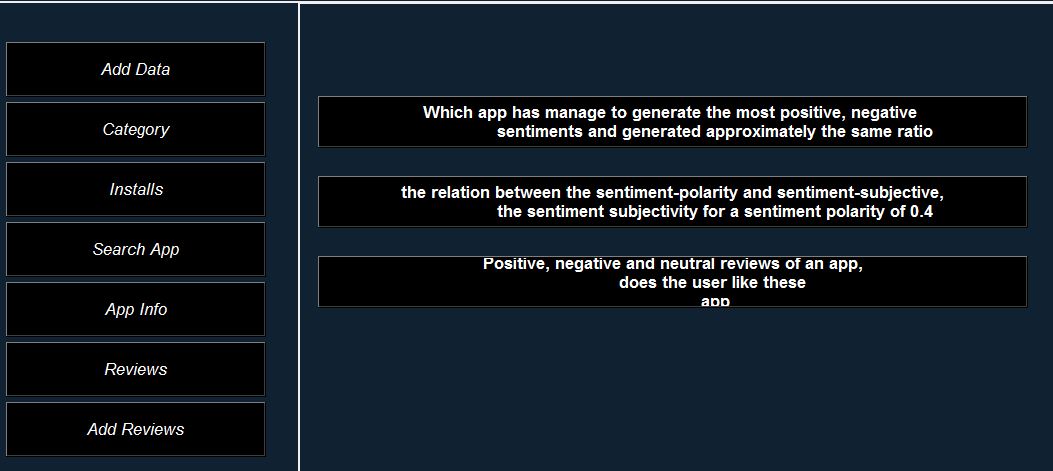
app.set("--select--")

r=Button(val,text='Search',width=12,command=sentim)

r.grid(row=0, column=3 ,padx=5,pady=5)

mcanvas.create\_window()

mcanvas.update()







13) Study and find out the relation between the Sentiment-polarity and sentimentsubjectivity of all the apps. What is the sentiment subjectivity for a sentiment polarity of 0.4.

Code:

def function\_q13():

global screen,df,dict\_app\_relation

dict\_app\_relation={}

root = Tk()

root.iconbitmap(r"C:\\InternshipFinal\\google.ico")

big\_frame = tk.Frame(root,bg='white',width='700',height='630',bd=4,relief=RIDGE)

big\_frame.place(x=50,y=60)

w=700

h=600

ws=root.winfo\_screenwidth()

hs=root.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

df = pd.read\_csv("C:\\InternshipFinal\\user.csv")

df=df.replace(np.NaN,-999)

dict\_app\_index\_count={}

for index in range(len(df['App'])):

app = df['App'][index]

if app in dict\_app\_index\_count:

dict\_app\_index\_count[app][1]+=1

else:

dict\_app\_index\_count[app]=[index,1]

# after this for loop dict\_app\_index\_count will hold the app name as key and it's first index in data set and total count in data set as item

for app in dict\_app\_index\_count:

index = dict\_app\_index\_count[app][0]

count = dict\_app\_index\_count[app][1]

sub,pol=[],[]

for i in range(count):

c = index+i

sub.append(df['Sentiment\_Subjectivity'][c])

pol.append(df['Sentiment\_Polarity'][c])

newRelation1(app,sub,pol)

app\_no = np.arange(len(dict\_app\_relation.keys()))

relation = []

for i in dict\_app\_relation:

relation.append(dict\_app\_relation[i])

figure3 = plt.Figure(figsize=(6,4), dpi=100)

ax3 = figure3.add\_subplot(111)

ax3.scatter(app\_no,relation, color = '#102131')

scatter3 = FigureCanvasTkAgg(figure3, root)

scatter3.get\_tk\_widget().place(x=50,y=45)

ax3.grid()

ax3.set\_xlabel("Applications in sequence")

ax3.set\_ylabel("Correlation")

ax3.set\_title("The Co-rrelation for Polarity V/s Subjectivity for all apps")

toolbar = NavigationToolbar2Tk(scatter3,root)

toolbar.update()

String = """

In this Scatter plot each point represent the correlation

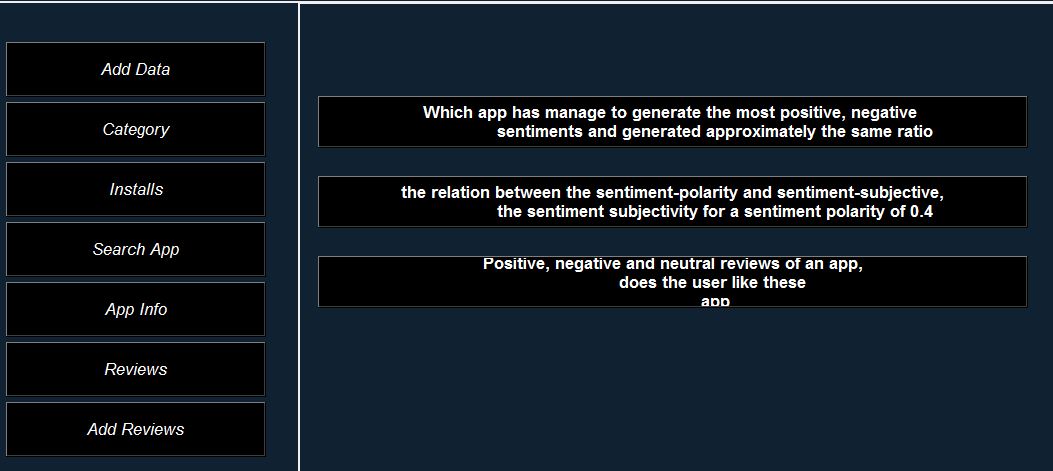
between sentiment polarity and sentiment subjectivity And

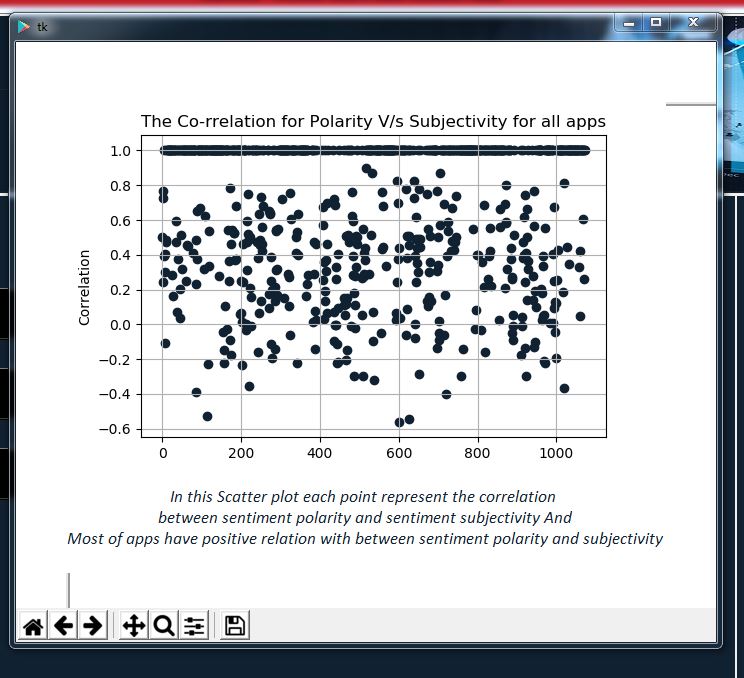
Most of apps have positive relation with between sentiment polarity and subjectivity

"""

tk.Label(root,text=String,font=("Calibri",13,'italic'),fg='#102131',bg='white').place(x=0,y=420)

root.mainloop()





**14) Generate an interface where the client can see the reviews categorized as positive.negative and neutral ,once they have selected the app from a list of apps available for the study.**

**15) Is it advisable to launch an app like ’10 Best foods for you’? Do the users like these apps?**

(code and output for 14 and 15 is done together)

Code:

ef revv():

global root

global search

global big\_frame

global filtered

global appli

global list\_of\_apps\_most\_positive\_sentiments

global list\_of\_apps\_most\_negative\_sentiments

global list\_of\_apps\_most\_average\_sentiments

root = Tk()

root.title("Insight of Google App's")

width\_value=root.winfo\_screenwidth()

height\_value=root.winfo\_screenheight()

root.geometry("%dx%d+0+0"%(1300,700))

root.configure(background='Cyan')

big\_frame = Frame(root)

big\_frame.pack()

l=Label(big\_frame,text='Positive',width=15,anchor=CENTER)

l.config(font=("Lucida", 16,'bold'))

l.grid(row=0, column=1 ,padx=5,pady=5)

l=Label(big\_frame,text='Neutral',width=15,anchor=CENTER)

l.config(font=("Lucida", 16,'bold'))

l.grid(row=0, column=2 ,padx=5,pady=5)

l=Label(big\_frame,text='Negative',width=15,anchor=CENTER)

l.config(font=("Lucida", 16,'bold'))

l.grid(row=0, column=3 ,padx=5,pady=5)

data=pd.read\_csv('C:\\InternshipFinal\\user.csv')

# print(data)

data=data.replace(np.nan,'Not Available')

x = search.get()

print(appli[filtered.index(search.get())])

list\_of\_apps\_most\_positive\_sentiments = []

list\_of\_apps\_most\_negative\_sentiments = []

list\_of\_apps\_most\_average\_sentiments = []

list\_of\_apps\_most\_zero\_sentiments = []

list\_of\_apps\_most\_positive\_sentiments = (data[(data.App == appli[filtered.index(search.get())]) & (data.Sentiment == 'Positive')].Translated\_Review).tolist()

# print(list\_of\_apps\_most\_positive\_sentiments)

list\_of\_apps\_most\_negative\_sentiments = (data[(data.App == appli[filtered.index(search.get())]) & (data.Sentiment == 'Negative')].Translated\_Review).tolist()

# print(list\_of\_apps\_most\_negative\_sentiments)

list\_of\_apps\_most\_average\_sentiments = (data[(data.App == appli[filtered.index(search.get())]) & (data.Sentiment == 'Neutral')].Translated\_Review).tolist()

# print(list\_of\_apps\_most\_average\_sentiments )

canvas=[]

for i in range(4):

can=Canvas(big\_frame,width=320,height=600,bg='#003b6b')

canvas.append(can)

can.grid(row=1,column=i)

scroll1=Scrollbar(canvas[1])

scroll2=Scrollbar(canvas[3])

scroll3=Scrollbar(canvas[2])

positive=Listbox(canvas[1],yscrollcommand = scroll1.set,height=35,width=45,bg='light green')

negative=Listbox(canvas[3],yscrollcommand = scroll2.set,height=35,width=43,bg='white')

neutral=Listbox(canvas[2],yscrollcommand = scroll3.set,height=35,width=45,bg='#ffcccb')

scroll1.pack(side = 'right', fill = 'both')

scroll2.pack(side = 'right', fill = 'both')

scroll3.pack(side = 'right', fill = 'both')

positive.pack(side = 'left', fill = 'both')

negative.pack( side = 'left', fill = 'both' )

neutral.pack( side = 'left', fill = 'both' )

for i in list\_of\_apps\_most\_positive\_sentiments:

positive.insert(END,i)

for i in list\_of\_apps\_most\_average\_sentiments:

neutral.insert(END,i)

for i in list\_of\_apps\_most\_negative\_sentiments:

negative.insert(END,i)

if (len(list\_of\_apps\_most\_positive\_sentiments)>len(list\_of\_apps\_most\_negative\_sentiments)) and (len(list\_of\_apps\_most\_positive\_sentiments)>len(list\_of\_apps\_most\_average\_sentiments)):

Label(canvas[0],text='User liked this app',width=25,anchor=CENTER,font=("Helvetica",15,'bold','italic')).pack()

elif (len(list\_of\_apps\_most\_negative\_sentiments)>len(list\_of\_apps\_most\_average\_sentiments)):

Label(canvas[0],text='User disliked this app',width=25,anchor=CENTER,font=("Helvetica",15,'bold','italic')).pack()

else:

Label(canvas[0],text='User neither liked nor disliked this app',width=25,anchor=CENTER,font=("Helvetica",15,'bold','italic')).pack()

def fourteen():

global search

global mcanvas

global filtered

global appli

mcanvas.delete("all")

val=Label(mcanvas,width=600,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(400,250, window=val)

data=pd.read\_csv('C:\\InternshipFinal\\user.csv')

data=data.replace(np.nan,'Not Available')

appli=list(OrderedDict.fromkeys(data['App']))

filtered=[]

for i in appli:

filtered.append(i[0:10])

search=StringVar()

Label(val, text='Search apps', anchor='w').grid(row=0, column=0 ,padx=5,pady=5, sticky="w")

app=Combobox(val , width=40,state="readonly",text=search,values=filtered)

app.grid(row=0, column=1 ,padx=5,pady=5, sticky="w")

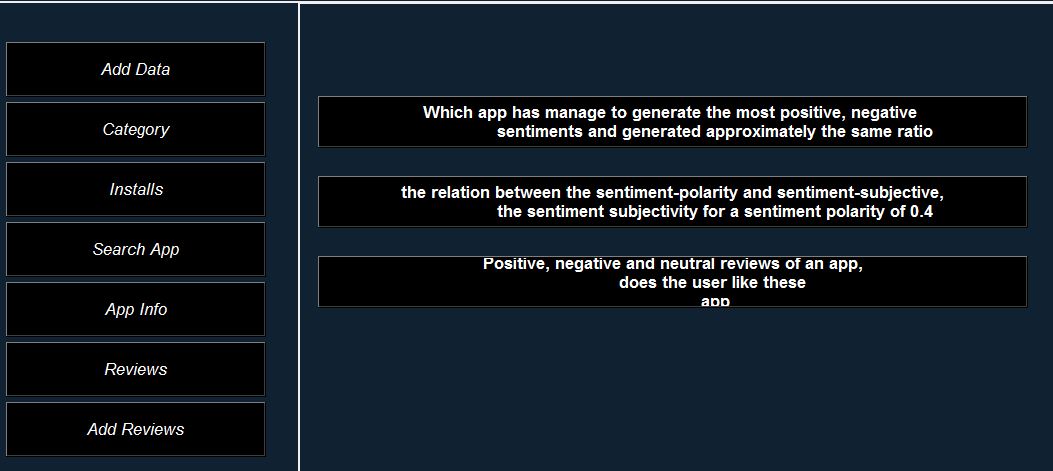
app.set("--select--")

r=Button(val,text='Review',width=12,command=revv)

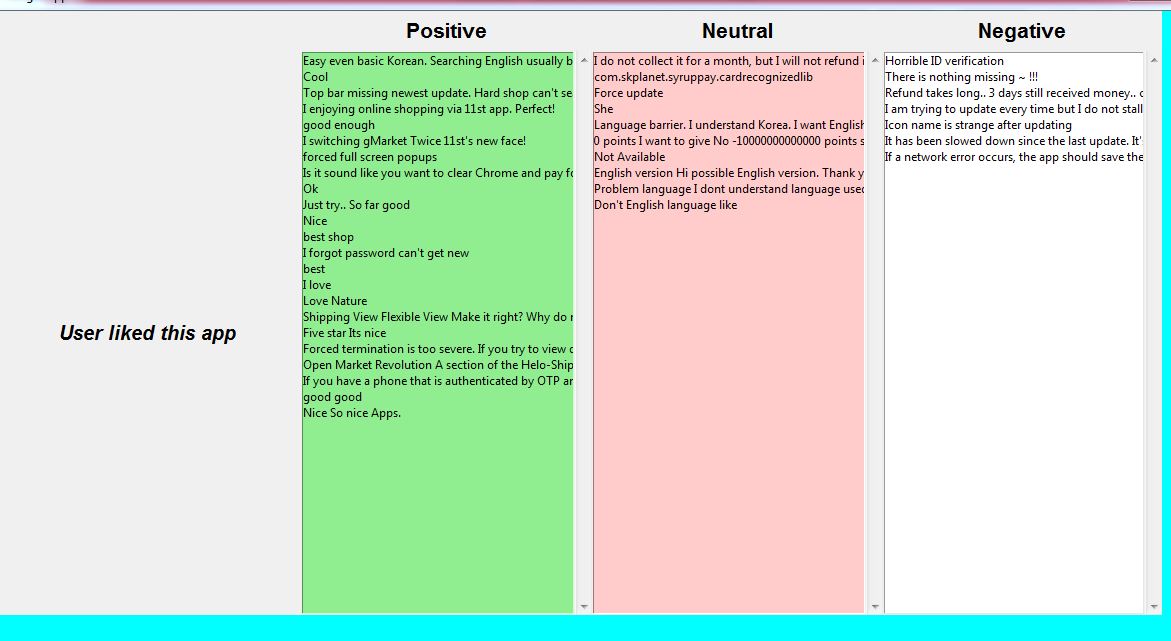
r.grid(row=0, column=3 ,padx=5,pady=5)

mcanvas.create\_window()

mcanvas.update()







**16) Which month(s) of the year , is the best indicator to the avarage downloads that an app will generate over the entire year?**

Code for Question 16

def year():

global root

global cat

global can

root = Tk()

root.title("Insight of Google App's")

width\_value=root.winfo\_screenwidth()

root.configure(background='Cyan') # configuring the window

height\_value=root.winfo\_screenheight()

root.geometry("%dx%d+0+0"%(width\_value, height\_value))

mcan=Canvas(root,width=800,height=700,bg='white')

mcan.place(x=300,y=70)

data=pd.read\_csv('C:\\InternshipFinal\\App-data.csv')

data=data.replace(np.nan,'Not Available')

data['Installs'] = data['Installs'].map(lambda x: x.rstrip('+'))

data['Installs'] = data['Installs'].map(lambda x: ''.join(x.split(',')))

data['Installs'] = pd.to\_numeric(data['Installs'])

d = pd.DatetimeIndex(data['Last Updated'])

data['year'] = d.year

data['month'] = d.month

# print(data['month'])

# print(data['year'])

mon={1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0}

for i in range(len(data)):

a=int(cat.get())

# print(a)

# print(data['year'][i])

if int(data['year'][i]) == a:

# print(data['month'][i])

if data['month'][i] in mon:

# print(data['month'][i])

if mon[data['month'][i]]==0:

mon[data['month'][i]]=data['Installs'][i]

# print(data['Installs'][i])

else:

mon[data['month'][i]]+=data['Installs'][i]

x=list(mon.keys())

y=list(mon.values())

figure1 = plt.Figure(figsize=(10,8), dpi=70)

axesObject = figure1.add\_subplot(111)

axesObject.bar(x,y)

axesObject.set\_title(f"Maximum Downloads in a month for a {cat.get()}")

can= FigureCanvasTkAgg(figure1,mcan)

can.get\_tk\_widget().pack( fill=BOTH, expand=True)

toolbar = NavigationToolbar2Tk(can,mcan)

toolbar.update()

def funct16():

global cat

global mcanvas

mcanvas.delete("all")

val=Label(mcanvas,width=600,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(300,250, window=val)

data=pd.read\_csv('C:\\InternshipFinal\\App-data.csv')

data=data.replace(np.nan,'Not Available')

d = pd.DatetimeIndex(data['Last Updated'])

data['year'] = d.year

data['month'] = d.month

cat=StringVar()

choices = list(data['year'].unique())

Label(val, text='Select Year', anchor='w').grid(row=0, column=0 ,padx=5,pady=5, sticky="w")

app=ttk.Combobox(val, width=40,state="readonly",text=cat,values=choices)

app.grid(row=0, column=1 ,padx=5,pady=5, sticky="w")

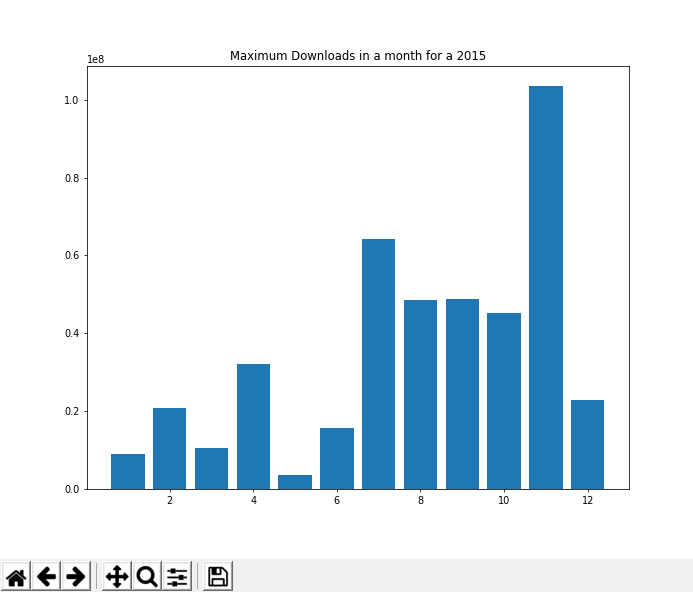
app.set("--select--")

r=Button(val,text='Search',width=12,command=year)

r.grid(row=0, column=3 ,padx=5,pady=5)

mcanvas.create\_window()

mcanvas.update()



**Q17) Does the size of the App influence the number of installs that it gets ? if,yes the trend is positive or negative with the increase in the app size.**

Code for Question 17

def functq17():

global screen

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

w = 600 # width for the window size

h = 700 # height for the window size

ws = screen.winfo\_screenwidth() # width of the screen

hs = screen.winfo\_screenheight() # height of the screen

x = (ws/2) - (w/2) # calculate x and y coordinates for the Tk window

y = (hs/2) - (h/2)

screen.geometry('%dx%d+%d+%d' % (w, h, x, y)) # set the dimensions of the screen and where it is placed

screen.resizable(False, False) # disabling the resize option for the window

screen.configure(background='white') # configuring the window

df= pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

list2=['More than 30 mb','20-30 mb','10-20 mb','Less Than 10 mb']

df['Size'] = df['Size'].map(lambda x: x.rstrip('M'))

df['Size'] = df['Size'].map(lambda x: str(round((float(x.rstrip('k'))/1024), 1)) if x[-1]=='k' else x)

df['Size'] = df['Size'].map(lambda x: np.nan if x.startswith('Varies') else x)

df['Size']=df['Size'].replace(np.NaN,-999)

df['Size']=df['Size'].astype(float)

#print(df['Category'].unique())

#print(df['Size'])

df['Installs']=df['Installs'].str.replace('+','')

df['Installs']=df['Installs'].str.replace(',','')

df['Installs']=df['Installs'].astype(int)

dict1,dict2,dict3,dict4,dict5,dict6={},{},{},{},{},{}

a,b,c,d=[],[],[],[]

for i in range(len(df)):

if df["Size"][i]>=30:

a.append(df['Installs'][i])

elif 20<=df["Size"][i]<30:

b.append(df['Installs'][i])

elif 10<=df["Size"][i]<20:

c.append(df['Installs'][i])

elif (df['Size'][i]<10):

d.append(df['Installs'][i])

a2=(sum(b))

a3=(sum(c))

a1=(sum(a))

a4=(sum(d))

list1=[a1,a2,a3,a4]

print(list1)

color = cm.rainbow(np.linspace(0, 2, 10))

fig=Figure(figsize=(3,2),dpi=100)

chart=fig.add\_subplot(111)

chart.bar(list2,list1,color=color)

chart.set\_ylabel("No of Installs")

chart.set\_xlabel("Sizes")

chart.grid()

fig.suptitle("No. of Installs Vs Size")

canvas = FigureCanvasTkAgg(fig, screen) # A tk.DrawingArea.

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

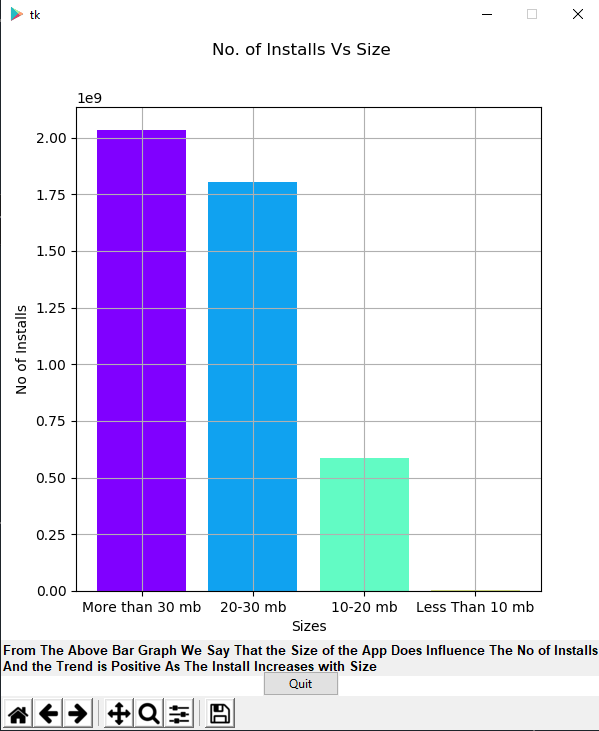
canvas.mpl\_connect("key\_press\_event", on\_key\_press)

Label(screen,text="From The Above Bar Graph We Say That the Size of the App Does Influence The No of Installs \nAnd the Trend is Positive As The Install Increases with Size",font=("Lucida",10,'bold')).place(x=0,y=610)

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()



# Section 5: Testing

We have carried out the successful testing of our application with the main frame executing properly and all the buttons working correctly.

The graphs are displaying correctly in the canvas, we have also added a toolbar just in case a graph doesn’t fit the screen.

For the Registration Form, Add Data Form and Add User Review Form are the only forms where validation has been used, where the user cannot input wrong values and the form will give an error if the user enters wrong values.

We have added validate button and save button in the Add Data Form and Add User Review Form, If the user attempted to click on the save button without clicking on the validate button there would no effect as the save button is disabled and will be enable only when the validation is correct.

# Section 6: The Source Code

## Mainscreen.py

# -\*- coding: utf-8 -\*-

"""

Created on Fri Jan 3 14:31:23 2020

@author: reube

"""

from functques import \*

from tkinter import \*

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk

from tkinter import PhotoImage

import functques as fn

from collections import OrderedDict

import tkinter as tk

#from adddata import \*

import time

import os

global mcanvas

import pandas as pd

import numpy as np

import tkinter.messagebox as tm

from PIL import Image

import pymysql

from matplotlib.backend\_bases import key\_press\_handler

from matplotlib.figure import Figure

import matplotlib.cm as cm

from collections import OrderedDict

#myimg=PhotoImage(file='C:\\Users\\GANDHI\\Desktop\\code\\python\\time series\\banner\_playsotre\_algorithm.png')

#photocanvas.create\_image(0,0,anchor=NW,image=myimg)

#w=Label(root,text="CATEGORY",width=1000,height=1,font=("Helvetica",15,'bold'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").pack()

#lcanvas=Canvas(width = 300,height=500,bg='#102131')

#lcanvas.place(x=0,y=180)

def on\_key\_press(event):

print("you pressed {}".format(event.key))

key\_press\_handler(event, canvas, toolbar)

def cancel():

mcanvas.delete("all")

head=Label(mcanvas,text="""Google Play Store App launch Study""",width=30,height=4,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(370, 150, window=head)

mcanvas.update()

def \_quit():

global screen

screen.quit() # stops mainloop

screen.destroy()

def saveing(x,y,z,p):

global data

connection = pymysql.connect(host="localhost", user="root", password="", database="googleplaystore") # database connection

cursor = connection.cursor()

value=[]

if z=='C:\\InternshipFinal\\App-data.csv':

date1=p[0].get()

month=p[1].get()

year=p[2].get()

date=month+' '+date1+','+' '+year

print(date)

dd=data.columns.tolist()

elif z=='C:\\InternshipFinal\\user.csv':

dd=sample.columns.tolist()

for i in x:

value.append(i.get())

print(value)

if z=='C:\\InternshipFinal\\App-data.csv':

insert\_query = "INSERT INTO appdata (appname,category,rating,review,size,install,type,price,cont\_rat,genres,last\_updated,current\_version,android\_version) VALUES('"+ value[0]+ "', '"+ value[1]+ "', '"+ value[2] + "', '"+ value[3] + "', '"+ value[4] + "', '"+ value[5]+ "', '"+ value[6]+ "', '"+ value[7]+ "', '"+ value[8]+ "', '"+ value[9]+ "', '"+ date+ "', '"+ value[10] + "', '"+ value[11] + "' );" # queries for inserting values

cursor.execute(insert\_query) # executing the

connection.commit() # commiting the connection then closing it.

connection.close() # closing the connection of the database

else:

insert\_query = "INSERT INTO addreview (app,trans\_rev,sentiment,sent\_polar,sent\_subj) VALUES('"+ value[0]+ "', '"+ value[1]+ "', '"+ value[2] + "', '"+ value[3] + "', '"+ value[4] + "' );"

cursor.execute(insert\_query)

print(insert\_query)

connection.commit() # commiting the connection then closing it.

connection.close() # closing the connection of the database

if z=='C:\\InternshipFinal\\App-data.csv':

value.insert(10,date)

#print(value)

value[5]=str(value[5])+'+'

value[7]='$'+str(value[7])

#print(value)

#print(dd)

dp=pd.DataFrame([value],columns=dd)

dat=data.append(dp)

elif z=='C:\\InternshipFinal\\user.csv':

dp=pd.DataFrame([value],columns=dd)

dat=sample.append(dp)

tk.messagebox.showinfo('Success','Data Successfully Written')

dat.to\_csv(z,index=False)

y.config(state='disabled')

def check(x,z):

d=[]

for i in x:

if i.get()=='':

tk.messagebox.showwarning('Fields empty','Please provide all the fields')

return True

for i in z:

if i.get()=='':

tk.messagebox.showwarning('Fields empty','Please provide all the fields')

return True

try:

if(isinstance(float(x[2].get()), float)):# code for checking the user entered a valid rating in the entry field

if(float(x[2].get())<=5 and float(x[2].get())>=0):

d.append(False)

else:

tk.messagebox.showerror('Out of range','Rating should be between 0 to 5 only')

return True

except:

tk.messagebox.showwarning('Wrong Value','Please provide a float value in rating column')

return True

try:

if(isinstance(int(x[3].get()), int)):

d.append(False)

except:

tk.messagebox.showwarning('Wrong Value','Please provide a integer value in Reviews')

return True

try:

if(isinstance(float(x[4].get()[:-1]), float)):

if(x[4].get()[-1]=='k' or x[4].get()[-1]=='M'):

d.append(False)

else:

tk.messagebox.showerror('Size',"Size should end with 'k' or 'M'")

return True

except:

tk.messagebox.showwarning('Wrong Value','Please provide a integer value followed in size column')

return True

try:

if(isinstance(float(x[5].get()), float)):

d.append(False)

except:

tk.messagebox.showwarning('Wrong Value','Please provide a integer value in Installs')

return True

try:

if x[6].get()=='Free':

if x[7].get()=='0':

d.append(False)

else:

tk.messagebox.showwarning('Free app','Please enter 0 in price column')

return True

except:

print('hi')

try:

if(isinstance(float(x[7].get()), float)):

d.append(False)

except:

tk.messagebox.showwarning('Wrong Value','Please provide a float value in Price')

return True

if set(d)==False:

return False

tk.messagebox.showinfo('Validate Succesfully','Now click on the Save Button')

def check1(x):

d=[]

for i in x:

if i.get()=='':

tk.messagebox.showwarning('Fields empty','Please provide all the fields')

return True

try:

if(isinstance(float(x[3].get()), float) and isinstance(float(x[4].get()), float)):

if x[2].get()=='Neutral':

if float(x[3].get())==0 and 1>=float(x[4].get())>=0:

d.append(False)

else:

tk.messagebox.showwarning('Neutral sentiment','Please provide a 0 in Sentiment polarity and Sentiment Subjectivity.')

return True

elif x[2].get()=='Positive':

if float(x[3].get())>0 and 1>=float(x[4].get())>=0:

d.append(False)

else:

tk.messagebox.showwarning('Positive sentiment','Please provide a positive value in Sentiment polarity and Sentiment Subjectivity.')

return True

elif x[2].get()=='Negative':

if float(x[3].get())<0 and 1>=float(x[4].get())>=0:

d.append(False)

else:

tk.messagebox.showwarning('Positive sentiment','Please provide a negative value in Sentiment polarity and non negative value in Sentiment Subjectivity.')

return True

except:

tk.messagebox.showwarning('Wrong Value','Please provide a float value in Sentiment polarity and Sentiment Subjectivity.')

return True

if set(d)==False:

return False

tk.messagebox.showinfo('Validate Succesfully','Now click on the Save Button')

def validate2(x,y):

global sample

App=x[0].get()

d=0

ap=sample['App'].unique()

for i in ap:

if i.strip()==App.strip():

msg='App named '+App+' is already present'

tk.messagebox.showerror("Error",msg)

d=1

if(check1(x)):

d=1

if d==0:

y.config(state='normal')

def add\_rev():

global screen,df,data,sample

dates=[]

sample=pd.read\_csv('C:\\InternshipFinal\\user.csv')

header2=sample.columns.tolist()

global mcanvas

val=Label(mcanvas,width=600,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(300,250, window=val)

txt=[]

datecombo=[]

month=['January', 'February', 'March', 'April','May','June','July','August','September', 'October', 'November','December']

years=[]

for i in range(1,32):

dates.append(i)

for i in range(2010,2020):

years.append(i)

mcanvas.delete("all")

val=Label(mcanvas,width=600,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(300,250, window=val)

#

txt2=[]

for i in range(1,6):

tk.Label(val,text=header2[i-1],width=17,font=("Lucida",11,'italic'),fg='#ffffff',bg='#102131').grid(row=i,column=0,padx=5,pady=5)

for i in range(1,6):

if i!=3:

txtfield=tk.Entry(val,bg="white")

txt2.append(txtfield)

txtfield.grid(row=i,column=2)

elif i==3:

combo=ttk.Combobox(val,values=['Positive','Negative','Neutral'],state="readonly")

txt2.append(combo)

combo.grid(row=3,column=2)

btn\_save1=tk.Button(val,text='Save',state="disabled",fg='#ffffff',width=10,command=lambda:saveing(txt2,btn\_save1,'C:\\InternshipFinal\\user.csv',''))

btn\_validate1=tk.Button(val,text='Validate',width=10,fg='#ffffff',bg="#102131",command=lambda:validate2(txt2,btn\_save1))

btn\_validate1.grid(row=7,column=2)

btn\_save1.grid(row=7,column=3)

root.mainloop()

def validate(x,y,z):

App=x[0].get()

d=0

ap=data['App']

for i in ap:

if i.strip()==App.strip():

msg='App named '+App+' is already present'

tk.messagebox.showerror("Error",msg)

d=1

break

if check(x,z):

d=1

if d==0:

y.config(state='normal')

def add\_app\_data():

global mcanvas,screen,df,data

dates=[]

month=['January', 'February', 'March', 'April','May','June','July','August','September', 'October', 'November','December']

years=[]

for i in range(1,32):

dates.append(i)

for i in range(2010,2020):

years.append(i)

data=pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

mcanvas.delete("all")

val=Label(mcanvas,width=600,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(300,250, window=val)

header=data.columns.tolist()

category= list(OrderedDict.fromkeys(data['Category']))

content=list(OrderedDict.fromkeys(data['Content Rating']))

genre=list(OrderedDict.fromkeys(data['Genres']))

txt=[]

datecombo=[]

for i in range(1,14):

Label(val,text=header[i-1],width=11,font=("Lucida",11,'italic'),fg='#ab3059',bg='#102131').grid(row=i,column=0,padx=2,pady=2)

for i in range(1,14):

if i!=2 and i!=10 and i!=9 and i!=7 and i!=11 and i!=13:

txtfield=tk.Entry(val,bg="white")

txt.append(txtfield)

txtfield.grid(row=i,column=1,padx=2,pady=2)

elif i==2:

combo=ttk.Combobox(val,values=category)

txt.append(combo)

combo.grid(row=2,column=1,padx=2,pady=2)

elif i==9:

combo=ttk.Combobox(val,values=content,state="readonly")

txt.append(combo)

combo.grid(row=9,column=1,padx=2,pady=2)

elif i==10:

combo=ttk.Combobox(val,values=genre,state="readonly")

txt.append(combo)

combo.grid(row=10,column=1,padx=2,pady=2)

elif i==7:

combo=ttk.Combobox(val,values=['Free','Paid'],state="readonly")

txt.append(combo)

combo.grid(row=7,column=1,padx=2,pady=2)

elif i==11:

combo=ttk.Combobox(val,values=dates,width=2,state="readonly").place(x=110,y=273)

datecombo.append(combo)

combo=ttk.Combobox(val,values=month,width=10,state="readonly").place(x=150,y=273)

datecombo.append(combo)

combo=ttk.Combobox(val,values=years,width=6,state="readonly").place(x=240,y=273)

datecombo.append(combo)

elif i==13:

combo=ttk.Combobox(val,values=list(data['Android Ver'].unique()),state="readonly")

txt.append(combo)

combo.grid(row=13,column=1,padx=2,pady=2)

btn\_save=tk.Button(val,text='Save',state="disabled",width=10,bg="#102131",command=lambda:saveing(txt,btn\_save,'C:\\InternshipFinal\\App-data.csv',datecombo))

btn\_validate=tk.Button(val,text='Validate',width=10,bg="#102131",command=lambda:validate(txt,btn\_save,datecombo))

btn\_validate.grid(row=14,column=1)

btn\_save.grid(row=14,column=2)

mcanvas.create\_window()

mcanvas.update()

def login\_verify():

global username\_verify

global password\_verify

connection = pymysql.connect(host="localhost", user="root", password="", database="googleplaystore") # database connection

cursor = connection.cursor()

select\_query = "SELECT \* FROM details where empid = '" + username\_verify.get() + "' AND password = '" + password\_verify.get() + "';" # queries for retrieving values

print(select\_query)

cursor.execute(select\_query) # executing the queries

student\_info = cursor.fetchall()

print(student\_info)

connection.commit() # commiting the connection then closing it.

connection.close() # closing the connection of the database

if student\_info:

messagebox.showinfo("Congratulation", "Login Succesfull") # displaying message for successful login

add\_app\_data()# opening welcome window

else:

messagebox.showerror("Error", "Invalid Username or Password") #

def login():

global mcanvas

global username\_verify

global password\_verify

mcanvas.delete("all")

val=Label(mcanvas,width=400,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(400,250, window=val)

# df= pd.read\_csv("C:\\Users\\Harsh\\Desktop\\internship\\googleplaystore-App-data.csv")

username\_verify = StringVar()

password\_verify = StringVar()

Label(val, text="Employee Login", width="400", height="2", font=("Lucida", 22, 'bold'), fg='white', bg='#102131').pack()

Label(val, text="", bg='#102131',width='100', height='17').place(x=45, y=120) # blue background in middle of window

Label(val, text="Please enter details below to login", bg='#102131', fg='white').pack()

Label(val, text="", bg='#102131').pack() # for leaving a space in between

Label(val, text="Employee ID \* ", font=("Open Sans", 10, 'bold'), bg='#102131', fg='white').pack()

Entry(val, textvar=username\_verify).pack()

Label(val, text="", bg='#102131').pack() # for leaving a space in between

Label(val, text="Password \* ", font=("Open Sans", 10, 'bold'), bg='#102131', fg='white').pack()

Entry(val, textvar=password\_verify, show="\*").pack()

Label(val, text="", bg='#102131').pack() # for leaving a space in between

Button(val, text="LOGIN", bg="black", width=15, height=1, font=("Open Sans", 13, 'bold'), fg='white',command=login\_verify).pack()

Label(val, text="", bg='#102131').pack() # for leaving a space in between

Button(val, text="New User? Register Here", height="2", width="30", bg='black', font=("Open Sans", 10, 'bold'), fg='white',command=register).pack()

mcanvas.update()

#displaying message for invalid details

def register\_user():

global mcanvas

global fullname

global email

global password

global repassword

global phone

global gender

global tnc

if fullname.get() and email.get() and password.get() and repassword.get() and gender.get(): # checking for all empty values in entry field

if (len(phone.get())!=10) and int(phone.get()): # checking for selection of university

ph\_no=Label(mcanvas, text="Enter the Valid Phone Number", fg="red",font=("Lucida", 11), width='30', anchor=W, bg='white')

mcanvas.create\_window(200,480,window=ph\_no)

return

else:

if tnc.get(): # checking for acceptance of agreement

if re.match("^.+@(\[?)[a-zA-Z0-9-.]+.([a-zA-Z]{2,3}|[0-9]{1,3})(]?)$", email.get()): # validating the email

if password.get() == repassword.get(): # checking both password match or not

# if u enter in this block everything is fine just enter the values in database

gender\_value = 'male'

if gender.get() == 2:

gender\_value = 'female'

connection = pymysql.connect(host="localhost", user="root", password="", database="googleplaystore") # database connection

cursor = connection.cursor()

insert\_query = "INSERT INTO details (empid,fullname,email, password, gender) VALUES('"+ phone.get() + "', '"+ fullname.get() + "', '"+ email.get() + "', '"+password.get() + "', '"+gender\_value + "' );" # queries for inserting values

cursor.execute(insert\_query) # executing the queries

connection.commit() # commiting the connection then closing it.

connection.close() # closing the connection of the database

rs=Label(mcanvas, text="Registration Sucess", fg="green", font=("Lucida", 11), width='30', anchor=W, bg='white')

mcanvas.create\_window(200,480,window=rs)# printing successful registration message

pl=Button(mcanvas, text='Proceed to Login ->', width=20, font=("Open Sans", 9, 'bold'), bg='brown', fg='white',command=login)

mcanvas.create\_window(500,480,window=pl) # button to navigate back to login page

else:

ps=Label(mcanvas, text="Password does not match", fg="red", font=("Lucida", 11), width='30', anchor=W, bg='white')

mcanvas.create\_window(200,480,window=ps)

return

else:

pvi=Label(mcanvas, text="Please enter valid email id", fg="red", font=("Lucida", 11), width='30', anchor=W, bg='white')

mcanvas.create\_window(200,480,window=pvi)

return

else:

pat=Label(mcanvas, text="Please accept the agreement", fg="red", font=("Lucida", 11), width='30', anchor=W, bg='white')

mcanvas.create\_window(200,480,window=pat)

return

else:

pfi=Label(mcanvas, text="Please fill all the details", fg="red",font=("Lucida", 11), width='30', anchor=W, bg='white')

mcanvas.create\_window(200,480,window=pfi)

return

mcanvas.update()

def register():

global mcanvas

global fullname

global email

global password

global repassword

global phone

global gender

global tnc

global mcanvas

mcanvas.delete("all")

val=Label(mcanvas,width=400,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(400,250, window=val)

fullname = StringVar()

email = StringVar()

password = StringVar()

repassword = StringVar()

phone= StringVar()

gender = IntVar()

tnc = IntVar()

# configuring the window

Label(val, text="Registration Form", width='32', height="2", font=("Lucida", 22, 'bold'), fg='white', bg='#102131').pack()

Label(val, text="", bg='#102131', width='100', height='20').place(x=45, y=120)

Label(val, text="Full Name:", font=("Open Sans", 11, 'bold'), fg='white', bg='#102131', anchor=W).pack()

Entry(val, textvar=fullname).pack()

Label(val, text="Email ID:", font=("Open Sans", 11, 'bold'), fg='white', bg='#102131', anchor=W).pack()

Entry(val, textvar=email).pack()

Label(val, text="Gender:", font=("Open Sans", 11, 'bold'), fg='white', bg='#102131', anchor=W).pack()

Radiobutton(val, text="Male", variable=gender, value=1, bg='#102131').pack()

Radiobutton(val, text="Female", variable=gender, value=2, bg='#102131').pack()

Label(val, text="Employee ID :", font=("Open Sans", 11, 'bold'), fg='white', bg='#102131', anchor=W).pack()

Entry(val, textvar=phone).pack()

phone.set('Enter Phone Number')

# droplist = OptionMenu(val, university, \*list1)

# droplist.config(width=17)

# university.set('--select your university--')

# droplist.pack()

Label(val, text="Password:", font=("Open Sans", 11, 'bold'), fg='white', bg='#102131', anchor=W).pack()

Entry(val, textvar=password, show="\*").pack()

Label(val, text="Re-Password:", font=("Open Sans", 11, 'bold'), fg='white', bg='#102131', anchor=W).pack()

entry\_4 = Entry(val, textvar=repassword, show="\*")

entry\_4.pack()

Checkbutton(val, text="I accept all terms and conditions", variable=tnc, bg='#102131', font=("Open Sans", 9, 'bold'), fg='brown').pack()

Button(val, text='Submit', width=20, font=("Open Sans", 13, 'bold'), bg='black', fg='white',command=register\_user).pack()

mcanvas.update()

"""THE END OF ADDING DATA FORMS AND LOGIN AND REGISTRATION FORM """

""" QUESTION 16 CODE"""

def year():

global root

global cat

global can

root = Tk()

root.title("Insight of Google App's")

width\_value=root.winfo\_screenwidth()

root.configure(background='Cyan') # configuring the window

height\_value=root.winfo\_screenheight()

root.geometry("%dx%d+0+0"%(width\_value, height\_value))

mcan=Canvas(root,width=800,height=700,bg='white')

mcan.place(x=300,y=70)

data=pd.read\_csv('C:\\InternshipFinal\\App-data.csv')

data=data.replace(np.nan,'Not Available')

data['Installs'] = data['Installs'].map(lambda x: x.rstrip('+'))

data['Installs'] = data['Installs'].map(lambda x: ''.join(x.split(',')))

data['Installs'] = pd.to\_numeric(data['Installs'])

d = pd.DatetimeIndex(data['Last Updated'])

data['year'] = d.year

data['month'] = d.month

# print(data['month'])

# print(data['year'])

mon={1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0}

for i in range(len(data)):

a=int(cat.get())

# print(a)

# print(data['year'][i])

if int(data['year'][i]) == a:

# print(data['month'][i])

if data['month'][i] in mon:

# print(data['month'][i])

if mon[data['month'][i]]==0:

mon[data['month'][i]]=data['Installs'][i]

# print(data['Installs'][i])

else:

mon[data['month'][i]]+=data['Installs'][i]

x=list(mon.keys())

y=list(mon.values())

figure1 = plt.Figure(figsize=(10,8), dpi=70)

axesObject = figure1.add\_subplot(111)

axesObject.bar(x,y)

axesObject.set\_title(f"Maximum Downloads in a month for a {cat.get()}")

can= FigureCanvasTkAgg(figure1,mcan)

can.get\_tk\_widget().pack( fill=BOTH, expand=True)

toolbar = NavigationToolbar2Tk(can,mcan)

toolbar.update()

def funct16():

global cat

global mcanvas

mcanvas.delete("all")

val=Label(mcanvas,width=600,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(300,250, window=val)

data=pd.read\_csv('C:\\InternshipFinal\\App-data.csv')

data=data.replace(np.nan,'Not Available')

d = pd.DatetimeIndex(data['Last Updated'])

data['year'] = d.year

data['month'] = d.month

cat=StringVar()

choices = list(data['year'].unique())

Label(val, text='Select Year', anchor='w').grid(row=0, column=0 ,padx=5,pady=5, sticky="w")

app=ttk.Combobox(val, width=40,state="readonly",text=cat,values=choices)

app.grid(row=0, column=1 ,padx=5,pady=5, sticky="w")

app.set("--select--")

r=Button(val,text='Search',width=12,command=year)

r.grid(row=0, column=3 ,padx=5,pady=5)

mcanvas.create\_window()

mcanvas.update()

""" QUESTION 10 CODE"""

def mont():

global root

global cat

global can

root = Tk()

root.title("Insight of Google App's")

width\_value=root.winfo\_screenwidth()

root.configure(background='Cyan') # configuring the window

height\_value=root.winfo\_screenheight()

root.geometry("%dx%d+0+0"%(width\_value, height\_value))

mcan=Canvas(root,width=800,height=700,bg='white')

mcan.place(x=300,y=70)

data=pd.read\_csv('C:\\InternshipFinal\\App-data.csv')

data=data.replace(np.nan,'Not Available')

data['Installs'] = data['Installs'].map(lambda x: x.rstrip('+'))

data['Installs'] = data['Installs'].map(lambda x: ''.join(x.split(',')))

data['Installs'] = pd.to\_numeric(data['Installs'])

d = pd.DatetimeIndex(data['Last Updated'])

data['year'] = d.year

data['month'] = d.month

mon={1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0}

for i in range(len(data)):

if data['Category'][i]== cat.get():

if data['month'][i] in mon:

if mon[data['month'][i]]==0:

mon[data['month'][i]]=data['Installs'][i]

else:

mon[data['month'][i]]+=data['Installs'][i]

x=list(mon.keys())

y=list(mon.values())

figure1 = plt.Figure(figsize=(10,8), dpi=70)

axesObject = figure1.add\_subplot(111)

axesObject.bar(x,y)

axesObject.set\_title(f"Maximum Downloads in a month for a {cat.get()}")

can= FigureCanvasTkAgg(figure1,mcan)

can.get\_tk\_widget().pack( fill=BOTH, expand=True)

toolbar = NavigationToolbar2Tk(can,mcan)

toolbar.update()

def funct10():

global cat

global mcanvas

mcanvas.delete("all")

val=Label(mcanvas,width=600,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(300,250, window=val)

data=pd.read\_csv('C:\\InternshipFinal\\App-data.csv')

data=data.replace(np.nan,'Not Available')

cat=StringVar()

choices = list(data['Category'].unique())

Label(val, text='Select Category', anchor='w').grid(row=0, column=0 ,padx=5,pady=5, sticky="w")

app=ttk.Combobox(val, width=40,state="readonly",text=cat,values=choices)

app.grid(row=0, column=1 ,padx=5,pady=5, sticky="w")

app.set("--select--")

r=Button(val,text='Search',width=12,command=mont)

r.grid(row=0, column=3 ,padx=5,pady=5)

mcanvas.create\_window()

mcanvas.update()

"""Question 12"""

def sentim():

global senti

root = Tk()

root.title("Insight of Google App's")

width\_value=root.winfo\_screenwidth()

height\_value=root.winfo\_screenheight()

root.geometry("%dx%d+0+0"%(width\_value, height\_value))

root.configure(background='Cyan')

big\_frame = Frame(root)

big\_frame.pack()

canvas=[]

for i in range(1):

can=Canvas(big\_frame,width=320,height=600,bg='white')

canvas.append(can)

can.grid(row=1,column=i)

scroll1=Scrollbar(canvas[0])

positive=Listbox(canvas[0],yscrollcommand = scroll1.set,height=35,width=45,bg='light green')

scroll1.pack(side = 'right', fill = 'both')

positive.pack(side = 'left', fill = 'both')

updated\_app={}

data=pd.read\_csv('C:\\InternshipFinal\\user.csv')

data=data.replace(np.nan,'Not Available')

app={}

if senti.get()=='--select--' :

root.destroy()

for i in data['App']:

app.update({i:0})

for i in range(len(data)):

if (data['App'][i] in app) and data['Sentiment'][i]==senti.get():

if app[data['App'][i]]==0:

app[data['App'][i]]=1

else:

app[data['App'][i]]+=1

for key, value in sorted(app.items(), key=lambda item: item[1],reverse=True):

updated\_app.update({key:value})

if senti.get()!='Same Ratio':

for i in updated\_app:

positive.insert(END,i,updated\_app[i])

if senti.get()=='Same Ratio':

app={}

for i in data['App']:

app.update({i:[0,0]})

# print(app)

for i in range(len(data)):

if (data['App'][i] in app) and data['Sentiment'][i]=='Positive':

if (app[data['App'][i]][0]) == 0:

app[data['App'][i]][0]=1

else:

app[data['App'][i]][0]+=1

for i in range(len(data)):

if (data['App'][i] in app) and data['Sentiment'][i]=='Negative':

if (app[data['App'][i]][1])==0:

app[data['App'][i]][1]=1

else:

app[data['App'][i]][1]+=1

same={}

for i in app:

if app[i][0]==0 or app[i][1]==0:

continue

elif 0.75<float((app[i][0]/app[i][1]))<1.25:

if (1-app[i][0]/app[i][1])<0:

a=(1-app[i][0]/app[i][1])\*(-1)

else:

a=(1-app[i][0]/app[i][1])

same.update({i:a})

for i in same:

positive.insert(END,i)

def twelve():

global senti

mcanvas.delete("all")

val=Label(mcanvas,width=600,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(300,250, window=val)

data=pd.read\_csv('C:\\InternshipFinal\\user.csv')

data=data.replace(np.nan,'Not Available')

senti=StringVar()

choices=['Positive','Negative','Same Ratio']

Label(val, text='Select Sentiment', anchor='w').grid(row=0, column=0 ,padx=5,pady=5, sticky="w")

app=Combobox(val , width=40,state="readonly",text=senti,values=choices)

app.grid(row=0, column=1 ,padx=5,pady=5, sticky="w")

app.set("--select--")

r=Button(val,text='Search',width=12,command=sentim)

r.grid(row=0, column=3 ,padx=5,pady=5)

mcanvas.create\_window()

mcanvas.update()

"""Question 14 And Question 15"""

def revv():

global root

global search

global big\_frame

global filtered

global appli

global list\_of\_apps\_most\_positive\_sentiments

global list\_of\_apps\_most\_negative\_sentiments

global list\_of\_apps\_most\_average\_sentiments

root = Tk()

root.title("Insight of Google App's")

width\_value=root.winfo\_screenwidth()

height\_value=root.winfo\_screenheight()

root.geometry("%dx%d+0+0"%(1300,700))

root.configure(background='Cyan')

big\_frame = Frame(root)

big\_frame.pack()

l=Label(big\_frame,text='Positive',width=15,anchor=CENTER)

l.config(font=("Lucida", 16,'bold'))

l.grid(row=0, column=1 ,padx=5,pady=5)

l=Label(big\_frame,text='Neutral',width=15,anchor=CENTER)

l.config(font=("Lucida", 16,'bold'))

l.grid(row=0, column=2 ,padx=5,pady=5)

l=Label(big\_frame,text='Negative',width=15,anchor=CENTER)

l.config(font=("Lucida", 16,'bold'))

l.grid(row=0, column=3 ,padx=5,pady=5)

data=pd.read\_csv('C:\\InternshipFinal\\user.csv')

# print(data)

data=data.replace(np.nan,'Not Available')

x = search.get()

print(appli[filtered.index(search.get())])

list\_of\_apps\_most\_positive\_sentiments = []

list\_of\_apps\_most\_negative\_sentiments = []

list\_of\_apps\_most\_average\_sentiments = []

list\_of\_apps\_most\_zero\_sentiments = []

list\_of\_apps\_most\_positive\_sentiments = (data[(data.App == appli[filtered.index(search.get())]) & (data.Sentiment == 'Positive')].Translated\_Review).tolist()

# print(list\_of\_apps\_most\_positive\_sentiments)

list\_of\_apps\_most\_negative\_sentiments = (data[(data.App == appli[filtered.index(search.get())]) & (data.Sentiment == 'Negative')].Translated\_Review).tolist()

# print(list\_of\_apps\_most\_negative\_sentiments)

list\_of\_apps\_most\_average\_sentiments = (data[(data.App == appli[filtered.index(search.get())]) & (data.Sentiment == 'Neutral')].Translated\_Review).tolist()

# print(list\_of\_apps\_most\_average\_sentiments )

canvas=[]

for i in range(4):

can=Canvas(big\_frame,width=320,height=600,bg='#003b6b')

canvas.append(can)

can.grid(row=1,column=i)

scroll1=Scrollbar(canvas[1])

scroll2=Scrollbar(canvas[3])

scroll3=Scrollbar(canvas[2])

positive=Listbox(canvas[1],yscrollcommand = scroll1.set,height=35,width=45,bg='light green')

negative=Listbox(canvas[3],yscrollcommand = scroll2.set,height=35,width=43,bg='white')

neutral=Listbox(canvas[2],yscrollcommand = scroll3.set,height=35,width=45,bg='#ffcccb')

scroll1.pack(side = 'right', fill = 'both')

scroll2.pack(side = 'right', fill = 'both')

scroll3.pack(side = 'right', fill = 'both')

positive.pack(side = 'left', fill = 'both')

negative.pack( side = 'left', fill = 'both' )

neutral.pack( side = 'left', fill = 'both' )

for i in list\_of\_apps\_most\_positive\_sentiments:

positive.insert(END,i)

for i in list\_of\_apps\_most\_average\_sentiments:

neutral.insert(END,i)

for i in list\_of\_apps\_most\_negative\_sentiments:

negative.insert(END,i)

if (len(list\_of\_apps\_most\_positive\_sentiments)>len(list\_of\_apps\_most\_negative\_sentiments)) and (len(list\_of\_apps\_most\_positive\_sentiments)>len(list\_of\_apps\_most\_average\_sentiments)):

Label(canvas[0],text='User liked this app',width=25,anchor=CENTER,font=("Helvetica",15,'bold','italic')).pack()

elif (len(list\_of\_apps\_most\_negative\_sentiments)>len(list\_of\_apps\_most\_average\_sentiments)):

Label(canvas[0],text='User disliked this app',width=25,anchor=CENTER,font=("Helvetica",15,'bold','italic')).pack()

else:

Label(canvas[0],text='User neither liked nor disliked this app',width=25,anchor=CENTER,font=("Helvetica",15,'bold','italic')).pack()

def fourteen():

global search

global mcanvas

global filtered

global appli

mcanvas.delete("all")

val=Label(mcanvas,width=600,height=8,font=("Lucida",30,'bold'),fg='black',bg='#102131')

mcanvas.create\_window(400,250, window=val)

data=pd.read\_csv('C:\\InternshipFinal\\user.csv')

data=data.replace(np.nan,'Not Available')

appli=list(OrderedDict.fromkeys(data['App']))

filtered=[]

for i in appli:

filtered.append(i[0:10])

# print(canvas)

search=StringVar()

Label(val, text='Search apps', anchor='w').grid(row=0, column=0 ,padx=5,pady=5, sticky="w")

app=Combobox(val , width=40,state="readonly",text=search,values=filtered)

app.grid(row=0, column=1 ,padx=5,pady=5, sticky="w")

app.set("--select--")

r=Button(val,text='Review',width=12,command=revv)

r.grid(row=0, column=3 ,padx=5,pady=5)

mcanvas.create\_window()

mcanvas.update()

"""TO UPDATE CATEGORIES INSTALL Q8 part 2"""

def Update\_cat():

df = pd.DataFrame()

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

# dict1={}

# dict1=pd.value\_counts(df['Category'])

list1={}

print(list1)

df['Installs']=df['Installs'].str.replace('+','')

df['Installs']=df['Installs'].str.replace(',','')

df['Installs']=df['Installs'].astype(int)

# print(sum(df['Installs']))

category ={}

sum1=[]

for i in df['Category']:

category.update({i:0})

for i in category.keys():

t2 = (df[i==(df.Category)].Installs).tolist()

sum1.append(sum(t2))

category.update({i:sum(t2)})

# print(category)

list1=list(category.values())

print(list1)

list2=list(category.keys())

print(list2)

connection = pymysql.connect(host="localhost", user="root", password="", database="googleplaystore") # database connection

cursor = connection.cursor()

cursor.execute("TRUNCATE TABLE catupdate")

for i in range(len(list2)):

insert\_query = "INSERT INTO catupdate (Categories,Downloads) VALUES('"+ list2[i] + "', '"+ str(list1[i]) + "' );" # queries for inserting values

cursor.execute(insert\_query) # executing the queries

connection.commit() # commiting the connection then closing it.

connection.close()

tk.messagebox.showinfo('Updated',"The Number Of Installs Have Been Updated")

"""Question 12th"""

def sent():

global root

global can

global senti

global updated\_app

global scroll1

global positive

canvas=[]

for i in range(1):

can=tk.Canvas(big\_frame,width=320,height=600,bg='white')

canvas.append(can)

can.grid(row=1,column=i)

scroll1=Scrollbar(canvas[0])

positive=Listbox(canvas[0],yscrollcommand = scroll1.set,height=35,width=45,bg='light green')

scroll1.pack(side = 'right', fill = 'both')

positive.pack(side = 'left', fill = 'both')

updated\_app={}

data=pd.read\_csv('C:\\InternshipFinal\\user.csv')

data=data.replace(np.nan,'Not Available')

app={}

if senti.get()=='--select--' :

root.destroy()

for i in data['App']:

app.update({i:0})

for i in range(len(data)):

if (data['App'][i] in app) and data['Sentiment'][i]==senti.get():

if app[data['App'][i]]==0:

app[data['App'][i]]=1

else:

app[data['App'][i]]+=1

for key, value in sorted(app.items(), key=lambda item: item[1],reverse=True):

updated\_app.update({key:value})

if senti.get()!='Same Ratio':

for i in updated\_app:

positive.insert(END,i,updated\_app[i])

if senti.get()=='Same Ratio':

app={}

for i in data['App']:

app.update({i:[0,0]})

# print(app)

for i in range(len(data)):

if (data['App'][i] in app) and data['Sentiment'][i]=='Positive':

if (app[data['App'][i]][0]) == 0:

app[data['App'][i]][0]=1

else:

app[data['App'][i]][0]+=1

for i in range(len(data)):

if (data['App'][i] in app) and data['Sentiment'][i]=='Negative':

if (app[data['App'][i]][1])==0:

app[data['App'][i]][1]=1

else:

app[data['App'][i]][1]+=1

same={}

for i in app:

if app[i][0]==0 or app[i][1]==0:

continue

elif 0.75<float((app[i][0]/app[i][1]))<1.25:

if (1-app[i][0]/app[i][1])<0:

a=(1-app[i][0]/app[i][1])\*(-1)

else:

a=(1-app[i][0]/app[i][1])

same.update({i:a})

for i in same:

positive.insert(END,i)

def Question12():

global root

global senti

global big\_frame

global updated\_app

global can

global scroll1

global positive

root = Tk()

root.title("Insight of Google App's")

width\_value=root.winfo\_screenwidth()

root.configure(background='Cyan') # configuring the window

height\_value=root.winfo\_screenheight()

root.geometry("%dx%d+0+0"%(900,900))

mainframe = Frame(root)

mainframe.pack()

big\_frame = Frame(root)

big\_frame.pack()

data=pd.read\_csv('C:\\InternshipFinal\\user.csv')

data=data.replace(np.nan,'Not Available')

senti=StringVar()

choices=['Positive','Negative','Same Ratio']

Label(mainframe, text='Select Sentiment', anchor='w').grid(row=0, column=0 ,padx=5,pady=5, sticky="w")

app=Combobox(mainframe , width=40,state="readonly",text=senti,values=choices)

app.grid(row=0, column=1 ,padx=5,pady=5, sticky="w")

app.set("--select--")

r=Button(mainframe,text='Search',width=12,command=sent)

r.grid(row=0, column=3 ,padx=5,pady=5)

# negative=Listbox(canvas[0],yscrollcommand = scroll1.set,height=35,width=43,bg='white')

# neutral=Listbox(canvas[0],yscrollcommand = scroll1.set,height=35,width=45,bg='#ffcccb')

root.mainloop()

"""THE MAIN SCREEN GUI """

def category():

global mcanvas

mcanvas.delete("all")

# q=mcanvas.create\_rectangle(40,40,500,80,fill='black')

q1 = Button(mcanvas,text = "The percentage download in each category in the playstore.",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq1)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 120, window=q1)

q3 = Button(mcanvas,text = """Which category of apps have managed to get the most,

least and an average of 2,50,000 downloads atleast.""",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq3)

# q4.bind("<Button-1>", function\_q4)

# q4.place(x=40,y=120)

mcanvas.create\_window(375,200, window=q3)

q0 = Button(mcanvas,text = "The percentage of Apps in each category in the playstore .",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq0)

# q5.bind("<Button-1>", function\_q5)

mcanvas.create\_window(375,280, window=q0)

q6 = Button(mcanvas,text = """For the years 2016,2017,2018 what are the category of apps that have got the

most and the least downloads""",width=90,height=6,font=("Lucida",10,'bold'),fg='#ffffff',bg='black',command=fn.functq6)

# q5.bind("<Button-1>", function\_q5)

mcanvas.create\_window(375,380, window=q6)

b=Button(mcanvas, text="Next",font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=nextc1)

mcanvas.create\_window(700,475, window=b)

mcanvas.update()

def nextc1():

global mcanvas

mcanvas.delete("all")

# q=mcanvas.create\_rectangle(40,40,500,80,fill='black')

q8 = Button(mcanvas,text = """Amongst Sports, Entertainment,social,News,Events,Travel and Game,

which is the category of app that is most likely to be downloaded""",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq8)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 120, window=q8)

b=Button(mcanvas, text="Previous",font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=category)

mcanvas.create\_window(700,475, window=b)

mcanvas.update()

def install():

global mcanvas

mcanvas.delete("all")

# q=mcanvas.create\_rectangle(40,40,500,80,fill='black')

q2 = Button(mcanvas,text = """Number of apps that have managed to get the following number of downloads

a) Between 10,000 and 50,000

b) Between 50,000 and 150000

c) Between 150000 and 500000

d) Between 500000 and 5000000

e) More than 5000000""",width=90,height=6,font=("Lucida",10,'bold'),fg='#ffffff',bg='black',command=fn.functq2)

# q3.bind("<Button-1>", function\_q3)

# q3.place(x=40,y=120)

mcanvas.create\_window(375,120, window=q2)

q5 = Button(mcanvas,text = """What is the number of installs for the following app sizes.

a) Size between 10 and 20 mb

b) Size between 20 and 30 mb

c) More than 30 mb""",width=90,height=6,font=("Lucida",10,'bold'),fg='#ffffff',bg='black',command=fn.functq5)

# q4.bind("<Button-1>", function\_q4)

# q4.place(x=40,y=200)

mcanvas.create\_window(375,240, window=q5)

q10\_1 = Button(mcanvas,text = "Month with maximum downloads for each of the category.",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=funct10)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 420, window=q10\_1)

b=Button(mcanvas, text="Next",font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=nexti1)

mcanvas.create\_window(700,475, window=b)

mcanvas.update()

def nexti1():

global mcanvas

mcanvas.delete("all")

# q=mcanvas.create\_rectangle(40,40,500,80,fill='black')

q10\_2 = Button(mcanvas,text = "Ratio of downloads for the app that qualifies as teen vs mature17+",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq10\_2)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 120, window=q10\_2)

q11 = Button(mcanvas,text = "Which quarter of which year has generated the highest number of install for each app used",width=78,height=2,font=("Lucida",12,'bold'),fg='#ffffff',bg='black',command=fn.question11)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 200, window=q11)

b=Button(mcanvas, text="Previous",font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=install)

mcanvas.create\_window(700,475, window=b)

mcanvas.update()

def rrev():

global mcanvas

mcanvas.delete("all")

q12 = Button(mcanvas,text = """Which app has manage to generate the most positive, negative

sentiments and generated approximately the same ratio""",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=twelve)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 120, window=q12)

q13 = Button(mcanvas,text = """the relation between the sentiment-polarity and sentiment-subjective,

the sentiment subjectivity for a sentiment polarity of 0.4""",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.function\_q13)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 200, window=q13)

# q=mcanvas.create\_rectangle(40,40,500,80,fill='black')

q14\_15 = Button(mcanvas,text = """Positive, negative and neutral reviews of an app,

does the user like these

app""",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fourteen)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 280, window=q14\_15)

mcanvas.update()

def app():

global mcanvas

mcanvas.delete("all")

# q=mcanvas.create\_rectangle(40,40,500,80,fill='black')

q7 = Button(mcanvas,text = """All those apps,whose android version is not an issue and can

work with varying devices.""",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq7)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 100, window=q7)

q7\_2 = Button(mcanvas,text = "What is the percentage increase or decrease in the downloads.",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq7\_2)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 180, window=q7\_2)

q4 = Button(mcanvas,text = "The apps that managed to get the highest maximum rating from the user.",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq4)

# q5.bind("<Button-1>", function\_q5)

mcanvas.create\_window(375,260, window=q4)

q9 = Button(mcanvas,text = """App managed to get get over 1,00,000 downloads,

and managed to get an average rating of 4.1 and above.""",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=fn.functq9)

# q5.bind("<Button-1>", function\_q5)

mcanvas.create\_window(375,340, window=q9)

q16 = Button(mcanvas,text = """Which month of the year, is the best indicator to the average

downloads that an app will generate over the entire year .""",width=70,height=2,font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=funct16)

# q5.bind("<Button-1>", function\_q5)

mcanvas.create\_window(375,420, window=q16)

b=Button(mcanvas, text="Next",font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=nexta1)

mcanvas.create\_window(700,475, window=b)

mcanvas.update()

def nexta1():

global mcanvas

mcanvas.delete("all")

# q=mcanvas.create\_rectangle(40,40,500,80,fill='black')

q17 = Button(mcanvas,text = "Does the size of the app influence the number of install that it get?",width=78,height=2,font=("Lucida",12,'bold'),fg='#ffffff',bg='black',command=fn.functq17)

# q3.bind("<Button-1>", function\_q3)

mcanvas.create\_window(375, 200, window=q17)

b=Button(mcanvas, text="Previous",font=("Lucida",13,'bold'),fg='#ffffff',bg='black',command=app)

mcanvas.create\_window(700,475, window=b)

mcanvas.update()

#===============================================main screen======================================================

root=Tk()

root.title("Insight of Google App's")

width\_value=root.winfo\_screenwidth()

height\_value=root.winfo\_screenheight()

root.geometry("%dx%d+250+100"%(1360,720))

root.configure(background='#102131')

root.iconbitmap(r"C:\\InternshipFinal\\google.ico")

#=================================top canvas===================================================================

photocanvas=Canvas(root,width =1355,height=177,bg='#102131')

photocanvas.place(x=0,y=0)

myimg=PhotoImage(file="C:\\InternshipFinal\\predictive\_analytics\_banner.png")

photocanvas.create\_image(0,0,anchor=NW,image=myimg)

photocanvas.image =myimg

#================================================main canvas ==============================

mcanvas=Canvas(width = 760,height=500,bg='#102131',bd='0')

mcanvas.place(x=300,y=180)

head=Label(mcanvas,text="Google \nPlayStore \n App launch \nStudy",width=30,font=("Lucida",50,'bold'),fg='#ffffff',bg='#102131')

mcanvas.create\_window(400, 200, window=head)

#=====================================================options==================================================

lbl\_over = Button(root,text = "Add Data",width=25,height='2',font=("Lucida",13,'italic'),fg='#ffffff',bg='black',command=add\_app\_data)

#lbl\_over.bind("<Button-1>")

lbl\_over.place(x=8,y=220)

lbl\_category = Button(root,text = "Category",width=25,height='2',font=("Lucida",13,'italic'),fg='#ffffff',bg='black',command=category)

#lbl\_category.bind("<Button-1>")

lbl\_category.place(x=8,y=220+60)

lbl\_Installs = Button(root,text = "Installs",width=25,height='2',font=("Lucida",13,'italic'),fg='#ffffff',bg='black',command=install)

#lbl\_Installs.bind("<Button-1>")

lbl\_Installs.place(x=8,y=220+60+60)

lbl\_searchapp = Button(root,text = "Search App",width=25,height='2',font=("Lucida",13,'italic'),fg='#ffffff',bg='black',command=fn.searchapp)

#lbl\_searchapp.bind("<Button-1>")

lbl\_searchapp.place(x=8,y=220+60+120)

lbl\_machine = Button(root,text = "App Info",width=25,height='2',font=("Lucida",13,'italic'),fg='#ffffff',bg='black',command=app)

lbl\_machine.bind("<Button-1>")

lbl\_machine.place(x=8,y=220+60+120+60)

lbl\_review = Button(root,text = "Reviews",width=25,height='2',font=("Lucida",13,'italic'),fg='#ffffff',bg='black',command=rrev)

#lbl\_review.bind("<Button-1>")

lbl\_review.place(x=8,y=220+60+120+120)

lbl\_lastupdate = Button(root,text = "Add Reviews",width=25,height='2',font=("Lucida",13,'italic'),fg='#ffffff',bg='black',command=add\_rev)

lbl\_lastupdate.bind("<Button-1>")

lbl\_lastupdate.place(x=8,y=220+60+120+180)

#======================================right canvas============================

rcanvas=Canvas(width = 295,height=500,bg='#102131')

rcanvas.place(x=1060,y=180)

Button(rcanvas,text = "Update The Installs\n Per Category",width=25,height='2',font=("Lucida",13,'italic'),fg='#ffffff',bg='black',command=Update\_cat).place(x=35,y=220)

#======================================bottom canvas============================

bottom=Canvas(width = 1190,height=500,bg='#102131')

bottom.place(x=0,y=682)

ball=bottom.create\_oval(4,4,30,30,fill='#ffffff')

#==================================================group name==============================================

name=Label(root,text="Ctrl+Alt+Del",width=15,height=1,font=("Helvetica",15,'bold','italic'),fg='#ffffff',bg='#102131')

name.place(x=1190,y=682)

root.mainloop()

## Funcques.py

# -\*- coding: utf-8 -\*-

"""

Created on Fri Jan 3 14:34:15 2020

@author: reube

"""

# -\*- coding: utf-8 -\*-

"""

Created on Mon Dec 23 19:39:12 2019

@author: GANDHI

"""

import tkinter as tk

from tkinter import \*

from tkinter.ttk import \*

import pandas as pd

import operator

import matplotlib.pyplot as plt

import numpy as np

from matplotlib.backends.backend\_tkagg import (FigureCanvasTkAgg, NavigationToolbar2Tk)

from matplotlib.backend\_bases import key\_press\_handler

from matplotlib.figure import Figure

import matplotlib.cm as cm

from collections import OrderedDict

global screen

def adjustWindow(window):

w = 600 # width for the window size

h = 600 # height for the window size

ws = window.winfo\_screenwidth() # width of the screen

hs = window.winfo\_screenheight() # height of the screen

x = (ws/2) - (w/2) # calculate x and y coordinates for the Tk window

y = (hs/2) - (h/2)

window.geometry('%dx%d+%d+%d' % (w, h, x, y)) # set the dimensions of the screen and where it is placed

window.resizable(False, False) # disabling the resize option for the window

window.configure(background='white') # making the background white of the window

def \_quit():

global screen

screen.quit() # stops mainloop

screen.destroy()

def on\_key\_press(event):

print("you pressed {}".format(event.key))

key\_press\_handler(event, canvas, toolbar)

def functq0():

global screen

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

# big\_frame = Frame(screen,bg='white',width='1010',height=450,bd=4)

# big\_frame.place(x=10,y=60)

screen.title("percentage of category") # mentioning title of the window

w = 1000 # width for the window size

h = 700 # height for the window size

ws = screen.winfo\_screenwidth() # width of the screen

hs = screen.winfo\_screenheight() # height of the screen

x = (ws/2) - (w/2) # calculate x and y coordinates for the Tk window

y = (hs/2) - (h/2)

screen.geometry('%dx%d+%d+%d' % (w, h, x, y)) # set the dimensions of the screen and where it is placed

screen.resizable(False, False) # disabling the resize option for the window

screen.configure(background='white') # configuring the window

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

df=df.replace(np.NaN,-1)

catcount={}

for index in range(len(df)):

if df['Category'][index]==-1:

continue

if df['Category'][index] in catcount:

catcount[df['Category'][index]]+=1

else:

catcount[df['Category'][index]]=1

figure1 = plt.Figure(figsize=(14,9), dpi=70)

# color = cm.rainbow(np.linspace(0, 1, len(x\_label)))

#fig1, ax1 = plt.subplots()

axesObject = figure1.add\_subplot(111)

labels = ['{0} '.format(i,j) for i,j in zip(catcount.keys(),catcount.values())]

theme = plt.get\_cmap('hsv')

axesObject.set\_prop\_cycle("color", [theme(1. \* i / len(catcount))for i in range(len(catcount))])

axesObject.pie(catcount.values(),autopct='%1.2f ',startangle=90)

axesObject.set\_title("Percentage Download in Each Category")

#ax3.xlim(0,3.0)

figure1.legend(labels,bbox\_to\_anchor=(0.3,1))

canvas = FigureCanvasTkAgg(figure1, master=screen) # A tk.DrawingArea.

canvas.draw()

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

canvas.mpl\_connect("key\_press\_event", on\_key\_press)

# this is necessary on Windows to prevent

# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()

def functq1():

global screen

screen = Tk()

big\_frame = Frame(screen,width='1010',height=750)

big\_frame.place(x=10,y=60)

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

w=1000

h=900

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

screen.geometry("%dx%d+%d+%d"%(w,h,x,y))

screen.configure(background='white')

df = pd.DataFrame()

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

# dict1={}

# dict1=pd.value\_counts(df['Category'])

list1={}

print(list1)

df['Installs']=df['Installs'].str.replace('+','')

df['Installs']=df['Installs'].str.replace(',','')

df['Installs']=df['Installs'].astype(int)

# print(sum(df['Installs']))

category ={}

sum1=[]

for i in df['Category']:

category.update({i:0})

for i in category.keys():

t2 = (df[i==(df.Category)].Installs).tolist()

sum1.append(sum(t2))

category.update({i:float(((sum(t2))/(sum(df['Installs'])))\*100)})

print(category)

list1=list(category.values())

# print(list1)

figure1 = plt.Figure(figsize=(14,9), dpi=70)

# color = cm.rainbow(np.linspace(0, 1, len(x\_label)))

#fig1, ax1 = plt.subplots()

axesObject = figure1.add\_subplot(111)

labels = ['{0} = {1:1.2f} % '.format(i,j) for i,j in zip(category.keys(),category.values())]

theme = plt.get\_cmap('hsv')

axesObject.set\_prop\_cycle("color", [theme(1. \* i / len(list1))for i in range(len(list1))])

axesObject.pie(list1,autopct='%1.2f ',startangle=90)

axesObject.set\_title("Percentage Apps in Each Category")

#ax3.xlim(0,3.0)

canvas = FigureCanvasTkAgg(figure1,big\_frame)

canvas.draw()

canvas.get\_tk\_widget().pack( fill=BOTH, expand=True)

toolbar = NavigationToolbar2Tk(canvas,big\_frame)

toolbar.update()

canvas.\_tkcanvas.pack( fill=BOTH, expand=True)

figure1.legend(labels,bbox\_to\_anchor=(0.3,1))

string="""From The Above Pie Chart,

We get the percentage Apps in Each Category """

Label(screen,text=string,font=("Calibri",13,'italic'),fg='#102131',bg='white').place(x=500,y=560)

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

# figureObject, axesObject = plt.subplots(figsize=(10,10))

# this is necessary on Windows to prevent

# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

screen.mainloop()

def functq2():

# initializing the tkinter window

global screen

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

screen.title("Apps vs Downloads") # mentioning title of the window

adjustWindow(screen) # configuring the window

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

df=df.replace(np.NaN,-1)

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

df['Installs'] = pd.to\_numeric(df['Installs'])

list2=["More than 5M","500k-5M","150k-500k","50k-150k","10k-50k"]

dict1,dict2,dict3,dict4,dict5={},{},{},{},{}

# dict6={}

dict1=(pd.value\_counts(df['Installs']>=5000000))

a1=len(df)-dict1.values[0]

dict2=(pd.value\_counts((df["Installs"]>=500000) & (df["Installs"]<5000000)))

a2=len(df)-dict2.values[0]

dict3=(pd.value\_counts((df["Installs"]>=150000) & (df["Installs"]<500000)))

a3=len(df)-dict3.values[0]

dict4=(pd.value\_counts((df["Installs"]>=50000) & (df["Installs"]<150000)))

a4=len(df)-dict4.values[0]

dict5=(pd.value\_counts((df["Installs"]>=10000) & (df["Installs"]<50000)))

a5=len(df)-dict5.values[0]

# dict6=pd.value\_counts(df["Installs"]<10000)

# a6=len(df)-dict6.values[0]

list1=[a1,a2,a3,a4,a5]

color = cm.rainbow(np.linspace(0, 2, 10))

fig=Figure(figsize=(5,4),dpi=100)

chart=fig.add\_subplot(111)

chart.bar(list2,list1,color=color)

chart.set\_ylabel("Frequency")

chart.set\_xlabel("Installs")

chart.grid()

fig.suptitle("Count-plot for Installs")

canvas = FigureCanvasTkAgg(fig, master=screen) # A tk.DrawingArea.

canvas.draw()

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

canvas.mpl\_connect("key\_press\_event", on\_key\_press)

# this is necessary on Windows to prevent

# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()

#def on\_key\_press(event):

# print("you pressed {}".format(event.key))

# key\_press\_handler(event, canvas, toolbar)

#

#canvas.mpl\_connect("key\_press\_event", on\_key\_press)

def functq3():

global screen

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

screen.title("Apps vs Downloads")

w = 600 # width for the window size

h = 600 # height for the window size

ws = screen.winfo\_screenwidth() # width of the screen

hs = screen.winfo\_screenheight() # height of the screen

x = (ws/2) - (w/2) # calculate x and y coordinates for the Tk window

y = (hs/2) - (h/2)

screen.geometry('%dx%d+%d+%d' % (w, h, x, y)) # set the dimensions of the screen and where it is placed

screen.resizable(False, False) # disabling the resize option for the window

screen.configure(background='white') # configuring the window

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

df=df.replace(np.NaN,0)

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

df['Installs'] = pd.to\_numeric(df['Installs'])

category=df['Category'].unique()

list1=df['Installs']

ans=[]

count = []

for i in category:

total=0

c=0

for j in range(len(df['Category'])):

if df['Category'][j]==i:

total=total+list1[j]

c+=1

# print(total)

ans.append(total)

count.append(c)

#print(ans)

# print(count)

cat,avg = [],[]

for index in range(len(ans)):

cat.append(category[index])

avg.append(round(ans[index]/count[index]))

# print(avg)

# print(cat)

lowest = []

for index in range(len(avg)):

if avg[index]<250000:

lowest.append(category[index])

# print(lowest)

label = category

# print(label)

val = avg

color = cm.rainbow(np.linspace(0, 1, len(label)))

fig=Figure(figsize=(8,5),dpi=60)

chart=fig.add\_subplot(111)

chart.barh(label,val,color=color)

chart.set\_ylabel("Category")

chart.set\_xlabel("Average Installs")

chart.grid()

fig.suptitle("Category with Their Average Download")

canvas = FigureCanvasTkAgg(fig, master=screen) # A tk.DrawingArea.

canvas.draw()

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

canvas.mpl\_connect("key\_press\_event", on\_key\_press)

# this is necessary on Windows to prevent

# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()

def functq4():

global screen

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

screen.title("Rating Vs Category ") # mentioning title of the window

adjustWindow(screen) # configuring the window

category ={}

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

df=df.replace(np.NaN,0)

catreview = {}

for index in range(len(df)):

if df['Category'][index] in catreview:

catreview[df['Category'][index]][0]+=df['Rating'][index]

catreview[df['Category'][index]][1]+=1

# rating+=df['Rating'][index]

else:

catreview[df['Category'][index]]=[df['Rating'][index],1]

# rating+=df['Rating'][index]

total=0

count=0

for i in df['Rating']:

total+=i

count+=1

avg= total/count

y=[]

x=[]

for i in catreview:

if catreview[i][0]/catreview[i][1]>=avg:

avgcat = (catreview[i][0]/catreview[i][1])

x.append(i)

y.append(float(avgcat))

# print(y)

# print(x)

color = cm.rainbow(np.linspace(0, 2, 15))

figure3 = plt.Figure(figsize=(5,4), dpi=80)

ax3 = figure3.add\_subplot(111)

ax3.scatter(y,x,color=color)

scatter3 = FigureCanvasTkAgg(figure3, screen)

scatter3.get\_tk\_widget().place(x=10,y=0)

ax3.grid()

ax3.set\_xlabel("RATING")

ax3.set\_ylabel("CATEGORY")

ax3.set\_title('CATEGORIES WITH HIGHEST MAXIMUM AVERAGE RATING')

canvas = FigureCanvasTkAgg(figure3, master=screen) # A tk.DrawingArea.

canvas.draw()

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

# canvas.mpl\_connect("key\_press\_event", on\_key\_press)

# this is necessary on Windows to prevent

# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()

def functq5():

global screen

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

w = 600 # width for the window size

h = 600 # height for the window size

ws = screen.winfo\_screenwidth() # width of the screen

hs = screen.winfo\_screenheight() # height of the screen

x = (ws/2) - (w/2) # calculate x and y coordinates for the Tk window

y = (hs/2) - (h/2)

screen.geometry('%dx%d+%d+%d' % (w, h, x, y)) # set the dimensions of the screen and where it is placed

screen.resizable(False, False) # disabling the resize option for the window

screen.configure(background='white') # configuring the window

df= pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

list2=['More than 30 mb','20-30 mb','10-20 mb']

df['Size'] = df['Size'].map(lambda x: x.rstrip('M'))

df['Size'] = df['Size'].map(lambda x: str(round((float(x.rstrip('k'))/1024), 1)) if x[-1]=='k' else x)

df['Size'] = df['Size'].map(lambda x: np.nan if x.startswith('Varies') else x)

df['Size']=df['Size'].replace(np.NaN,-999)

df['Size']=df['Size'].astype(float)

#print(df['Category'].unique())

#print(df['Size'])

df['Installs']=df['Installs'].str.replace('+','')

df['Installs']=df['Installs'].str.replace(',','')

df['Installs']=df['Installs'].astype(int)

dict1,dict2,dict3,dict4,dict5,dict6={},{},{},{},{},{}

a,b,c=[],[],[]

for i in range(len(df)):

if df["Size"][i]>=30:

a.append(df['Installs'][i])

elif 20<=df["Size"][i]<30:

b.append(df['Installs'][i])

elif 10<=df["Size"][i]<20:

c.append(df['Installs'][i])

a2=(sum(b))

a3=(sum(c))

a1=(sum(a))

# dict1=(pd.value\_counts(df["Size"]>=30))

# a1=len(df)-dict1.values[0]

# print(a1)

# dict2=(pd.value\_counts((df["Size"]>=20) & (df["Size"]<30)))

# a2=len(df)-dict2.values[0]

# print(a2)

# dict3=(pd.value\_counts((df["Size"]>=10) & (df["Size"]<20)))

# a3=len(df)-dict3.values[0]

# print(a3)

#dict4=(pd.value\_counts((df["Size"]<10)))

#a4=len(df)-dict4.values[0]

#print(a4)

list1=[a1,a2,a3]

print(list1)

# plt.bar(list2,list1 , color='green')

# plt.title("mb vs app")

# plt.xlabel("Downloads")

# plt.ylabel("App")

# plt.xticks(rotation=90)

color = cm.rainbow(np.linspace(0, 2, 10))

fig=Figure(figsize=(5,4),dpi=100)

chart=fig.add\_subplot(111)

chart.bar(list2,list1,color=color)

chart.set\_ylabel("No of Installs")

chart.set\_xlabel("Sizes")

chart.grid()

fig.suptitle("No. of Installs Vs Size")

canvas = FigureCanvasTkAgg(fig, master=screen) # A tk.DrawingArea.

canvas.draw()

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

canvas.mpl\_connect("key\_press\_event", on\_key\_press)

# this is necessary on Windows to prevent

# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()

def functq6():

global screen

screen=Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

w=700

h=600

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

screen.geometry("%dx%d+%d+%d"%(w,h,x,y))

screen.configure(background='white')

# big\_frame = tk.Frame(root,bg='white',width='700',height=550,bd=4,relief=RIDGE)

# big\_frame.place(x=10,y=60)

# adjustWindow(root) # configuring the window

# Label(screen,text="").pack()

#

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

#print(df.head(5))

#df.drop(9148,axis=0, inplace=True)

#df.drop(10472,axis=0,inplace=True)

# Data cleaning for "Installs" column

#print(df['Installs'].head(5))

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

#print(df['Installs'].head(5))

df['Installs'] = pd.to\_numeric(df['Installs'])

d = pd.DatetimeIndex(df['Last Updated'])

df['year'] = d.year

df['month'] = d.month

#print((df['year'][5]))

#6) For the years 2016,2017,2018 what are the category of apps that have got the most and the least downloads. What is the percentage increase or decrease that the

dict\_2016 = {}

dict\_2017 = {}

dict\_2018 = {}

Category = []

for cat in df['Category'].unique():

Category.append(cat)

dict\_2016[cat]=0

dict\_2017[cat]=0

dict\_2018[cat]=0

#print(Category)

for index in range(len(df)):

if df['year'][index]==2016:

dict\_2016[df['Category'][index]] += df['Installs'][index]

if df['year'][index]==2017:

dict\_2017[df['Category'][index]] += df['Installs'][index]

if df['year'][index]==2018:

dict\_2018[df['Category'][index]] += df['Installs'][index]

#print(len(dict\_2016))

#print(len(dict\_2017))

#print(len(dict\_2018))

#print(dict\_2016)

#print(dict\_2017)

#print(dict\_2018)

max\_2016\_install = ["",0]

max\_2017\_install = ["",0]

max\_2018\_install = ["",0]

min\_2016\_install = ["",99999999999]

min\_2017\_install = ["",99999999999]

min\_2018\_install = ["",99999999999]

for cat in dict\_2016:

if max\_2016\_install[1] < dict\_2016[cat]:

max\_2016\_install[1] = dict\_2016[cat]

max\_2016\_install[0] = cat

if max\_2017\_install[1] < dict\_2017[cat]:

max\_2017\_install[1] = dict\_2017[cat]

max\_2017\_install[0] = cat

if max\_2018\_install[1] < dict\_2018[cat]:

max\_2018\_install[1] = dict\_2018[cat]

max\_2018\_install[0] = cat

if min\_2016\_install[1] > dict\_2016[cat]:

min\_2016\_install[1] = dict\_2016[cat]

min\_2016\_install[0] = cat

if min\_2017\_install[1] > dict\_2017[cat]:

min\_2017\_install[1] = dict\_2017[cat]

min\_2017\_install[0] = cat

if min\_2018\_install[1] > dict\_2018[cat]:

min\_2018\_install[1] = dict\_2018[cat]

min\_2018\_install[0] = cat

#print(max\_2016\_install)

#print(max\_2017\_install)

#print(max\_2018\_install)

#print(min\_2016\_install)

#print(min\_2017\_install)

#print(min\_2018\_install)

max\_install = [max\_2016\_install[1],max\_2017\_install[1],max\_2018\_install[1]]

min\_install = [min\_2016\_install[1],min\_2017\_install[1],min\_2018\_install[1]]

Years = ['2016','2017','2018']

pos = np.arange(len(Years))

bar\_width = 0.3

figure2 = plt.Figure(figsize=(8,4), dpi=85)

chart = figure2.add\_subplot(111)

Max\_bar = chart.bar(Years,max\_install,bar\_width,color='blue',edgecolor='blue')

Min\_bar = chart.bar(pos+bar\_width,min\_install,bar\_width,color='red',edgecolor='red')

chart.grid()

chart.set\_ylabel("Download")

chart.set\_xlabel('Years')

figure2.suptitle('Max and Min download across 2016-17-18 years for a category',fontsize=18)

plt.legend(['max','min'],loc=10)

max\_month = [max\_2016\_install[0],max\_2017\_install[0],max\_2018\_install[0]]

min\_month = [min\_2016\_install[0],min\_2017\_install[0],min\_2018\_install[0]]

for idx,rect in enumerate(Max\_bar):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,max\_month[idx],ha='center', va='bottom', rotation=0)

for idx,rect in enumerate(Min\_bar):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,min\_month[idx],ha='center', va='bottom', rotation=0)

canvas = FigureCanvasTkAgg(figure2, master=screen)

canvas.get\_tk\_widget().pack()

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

canvas.mpl\_connect("key\_press\_event", on\_key\_press)

# this is necessary on Windows to prevent

# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

Label(screen,text="The Downloads in the Last Three Years",font=("Helvetica",11,'bold') ,borderwidth=2).place(x=200,y=500)

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()

def functq7():

global screen

screen=Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

w=720

h=600

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

screen.geometry("%dx%d+%d+%d"%(w,h,x,y))

screen.configure(background='white')

df= pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

df=df.replace(np.NaN,0)

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

df['Installs'] = pd.to\_numeric(df['Installs'])

varwith=[]

novar=[]

varcategory={}

nocat={}

for i in range(len(df['App'])):

if df['Android Ver'][i]=='Varies with device':

varwith.append(df['Installs'][i])

if df['Category'][i] in varcategory:

varcategory[df['Category'][i]]+=df['Installs'][i]

else:

varcategory[df['Category'][i]]=df['Installs'][i]

else:

novar.append(df['Installs'][i])

if df['Category'][i] in nocat:

nocat[df['Category'][i]]+=df['Installs'][i]

else:

nocat[df['Category'][i]]=df['Installs'][i]

# print(varwith)

# print(novar)

# print(varcategory)

# print(nocat)

sumvarcategory=sum(varwith)

sumnocat=sum(novar)

# print(sumvarcategory)

# print(sumnocat)

x=(len(varwith),len(novar))

# print(x)

androidver = ['Varying', 'Not varying']

figure1 = plt.Figure(figsize=(10,7), dpi=70)

color = cm.rainbow(np.linspace(0, 1, len(x)))

#fig1, ax1 = plt.subplots()

axesObject = figure1.add\_subplot(111)

# labels = ['{0} '.format(i,j) for i,j in zip(catcount.keys(),catcount.values())]

theme = plt.get\_cmap('hsv')

# axesObject.set\_prop\_cycle("color", [theme(1. \* i / len(catcount))for i in range(len(catcount))])

axesObject.pie(x,labels=androidver,autopct='%1.2f',startangle=90,colors=color,shadow=True,explode=[0.1,0])

axesObject.set\_title("Frequency of Varying Apps in Android version vs Apps in Non-varying Android Version in dataset")

#ax3.xlim(0,3.0)

# figure1.legend(labels,bbox\_to\_anchor=(0.3,1))

canvas = FigureCanvasTkAgg(figure1, master=screen) # A tk.DrawingArea.

canvas.draw()

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

canvas.mpl\_connect("key\_press\_event", on\_key\_press)

# this is necessary on Windows to prevent

# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()

def functq7\_2():

global screen

screen = tk.Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

big\_frame = tk.Frame(screen,bg='white',width='700',height=450,bd=4,relief=RIDGE)

big\_frame.place(x=10,y=60)

w=720

h=550

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

screen.geometry("%dx%d+%d+%d"%(w,h,x,y))

screen.configure(background='white')

tk.Label(screen,text="",bg='white').pack()

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

#print(df.head(5))

#df.drop(9148,axis=0, inplace=True)

#df.drop(10472,axis=0,inplace=True)

# Data cleaning for "Installs" column

#print(df['Installs'].head(5))

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

#print(df['Installs'].head(5))

df['Installs'] = pd.to\_numeric(df['Installs'])

d = pd.DatetimeIndex(df['Last Updated'])

df['year'] = d.year

df['month'] = d.month

#print((df['year'][5]))

#6) For the years 2016,2017,2018 what are the category of apps that have got the most and the least downloads. What is the percentage increase or decrease that the

dict\_years = {}

for year in df['year'].unique():

dict\_years[year]=0

for index in range(len(df)):

dict\_years[df['year'][index]] += df['Installs'][index]

Years = []

list\_install = []

# for year in dict\_years:

# if year==2016 or year==2017 or year==2018:

# Years.append(str(year))

# list\_install.append(dict\_years[year])

for year in dict\_years:

Years.append((year))

list\_install.append(dict\_years[year])

# print(Years)

# print(list\_install)

new\_dict={}

for i in range(0,9):

new\_dict.update({Years[i]:list\_install[i]})

new\_dict1=dict(sorted(new\_dict.items(), key=operator.itemgetter(0),reverse=True))

keys=list(new\_dict1.keys())

values=list(new\_dict1.values())

print(keys)

print(values)

# for i in

# print(dict\_years)

x = dict\_years[2016]

y = dict\_years[2017]

z=dict\_years[2018]

per2016=1

per2017=((y-x)/(x+y))\*100

per2018=((z-y)/(y+z))\*100

# print(per2016,per2017,per2018)

Years.reverse()

list\_install.reverse()

figure2 = plt.Figure(figsize=(8,4), dpi=85)

chart = figure2.add\_subplot(111)

chart.plot(keys,values,color='blue')

#Min\_bar = chart.bar(pos+bar\_width,min\_install,bar\_width,color='pink',edgecolor='black')

chart.set\_ylabel("Years")

chart.set\_xlabel('Installs')

figure2.suptitle('Barchart on Installs on each Year ',fontsize=18)

chart.grid()

canvas = FigureCanvasTkAgg(figure2, master=big\_frame)

canvas.get\_tk\_widget().place(x=5,y=10)

String = """

% increase in 2016-17 is {:.1f}% and % increase in 2017-18 is {:.1f}%

""".format(per2017,per2018)

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=40,y=360)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

screen.mainloop()

#x axis in order 2014

def functq8():

global screen

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

screen.title("Apps to be most likely downloaded in the Upcoming Years") # mentioning title of the window

w = 600 # width for the window size

h = 500 # height for the window size

ws = screen.winfo\_screenwidth() # width of the screen

hs = screen.winfo\_screenheight() # height of the screen

x = (ws/2) - (w/2) # calculate x and y coordinates for the Tk window

y = (hs/2) - (h/2)

screen.geometry('%dx%d+%d+%d' % (w, h, x, y)) # set the dimensions of the screen and where it is placed

screen.resizable(False, False) # configuring the window

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

df=df.replace(np.NaN,0)

cat={'SPORTS':0,'ENTERTAINMENT':0,'SOCIAL':0,'NEWS\_AND\_MAGAZINES':0,'EVENTS':0,'TRAVEL\_AND\_LOCAL':0,'GAME':0}

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

#print(df['Installs'].head(5))

df['Installs'] = pd.to\_numeric(df['Installs'])

d = pd.DatetimeIndex(df['Last Updated'])

df['year'] = d.year

df['month'] = d.month

# dict\_2018={}

for i in range(len(df)):

if (df['year'][i]==2018):

if df['Category'][i] in cat:

if cat[df['Category'][i]]==0:

cat[df['Category'][i]]=1

else:

cat[df['Category'][i]]+=1

# print(cat)

color = cm.rainbow(np.linspace(0, 2, 15))

fig=Figure(figsize=(5,4),dpi=100)

chart=fig.add\_subplot(111)

k=list(cat.keys())

v=list(cat.values())

l=v.index(max(v))

print(k[l])

chart.barh(k,v,color=color)

chart.set\_ylabel("No of Installs")

chart.set\_xlabel("Categories")

chart.grid()

fig.suptitle("Count-plot for Installs")

canvas = FigureCanvasTkAgg(fig, master=screen) # A tk.DrawingArea.

canvas.draw()

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

canvas.mpl\_connect("key\_press\_event", on\_key\_press)

# this is necessary on Windows to prevent

# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

String = (f""" The Most Likely App to be downloaded in the

upcoming Years is {k[l]}""")

Label(screen,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=400,y=690)

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()

def functq9():

global screen

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

screen = tk.Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

big\_frame = tk.Frame(screen,bg='white',width='600',height='630',bd=4)

big\_frame.place(x=50,y=60)

w=700

h=700

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

screen.geometry("%dx%d+%d+%d"%(w,h,x,y))

screen.configure(background='white')

rating = 4.1

installs = 100000

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

print(df['Rating'])

temp = []

for index in range(len(df['Rating'])):

if df['Rating'][index] >= rating:

temp.append(1)

else:

temp.append(0)

cat\_rating= pd.DataFrame(zip(temp,temp),columns=["cat\_Ratings","ignore"])

df = pd.concat([df,cat\_rating],axis=1)

df.drop("ignore",axis=1,inplace=True)

df.drop(df.index[9148], inplace=True)

# Data cleaning for "Installs" column

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

df['Installs'] = pd.to\_numeric(df['Installs'])

rating\_sum = 0

rate=[]

#1169

""" """

counter=0

for index in range(len(df)):

try:

if df['Installs'][index]>=installs:

#if df['Rating'][index]>=rating:""" """

rate.append(1)

rating\_sum+=df['Rating'][index]

counter+=1

""" """

else:

rate.append(0)

except:

#print(index)

continue

#print(len(rate))

avg\_rating = (rating\_sum/counter)

""" """

#print(df['Installs'].corr(df['Rating']))

""" """

val = "Yes" if (rating\_sum/counter)>=rating else "No"

rel = "Greater than" if val == "Yes" else "Lesser than"

fig, ax = plt.subplots(figsize=(10, 10))

l1 ='{}>='.format(installs)

l2 ='<{}'.format(installs)

size=[rate.count(1),rate.count(0)]

label = [l1,l2]

title = 'Count of {}'.format(rating)

figure1 = plt.Figure(figsize=(8,8), dpi=70)

labels1 = ['{0} = {1:1.2f} % '.format(i,j) for i,j in zip(label,size)]

#color = cm.rainbow(np.linspace(0, 1, 10))

#fig1, ax1 = plt.subplots()

ax3 = figure1.add\_subplot(111)

ax3.pie(size, labels=label,colors = ['green','cyan'], autopct='%1.1f%%', startangle=200)

ax3.set\_title(title)

ax3.legend(labels1,bbox\_to\_anchor=(1,1))

#ax3.xlim(0,3.0)

pie\_plot = FigureCanvasTkAgg(figure1, big\_frame)

pie\_plot.get\_tk\_widget().place(x=-50,y=-70)

Label(big\_frame,text="--Results--",font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=220,y=470)

String = "Average rating of all the apps who managed to get over {} download is {:.1f}".format(installs,avg\_rating)

Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=0,y=500)

String ="""{}! All those apps who have managed to get over {} downloads ,

they have to get an average rating of {:.1f} which is {} than {} """.format(val,installs,avg\_rating,rel,rating)

Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=0,y=530)

#ax3.legend(loc=0)

toolbar = NavigationToolbar2Tk(pie\_plot, screen)

toolbar.update()

pie\_plot.mpl\_connect("key\_press\_event", on\_key\_press)

# this is necessary on Windows to prevent

# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()

def functq10\_2():

global screen

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

screen.title("Apps vs Downloads") # mentioning title of the window

adjustWindow(screen) # configuring the window

df = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

df=df.replace(np.NaN,0)

ratio={'Teen':0,'Mature 17+':0}

for i in range(len(df)):

if df['Content Rating'][i] in ratio:

if ratio[df['Content Rating'][i]]==0:

ratio[df['Content Rating'][i]]=1

else:

ratio[df['Content Rating'][i]]+=1

color = cm.rainbow(np.linspace(0, 2, 10))

fig=Figure(figsize=(5,4),dpi=100)

chart=fig.add\_subplot(111)

chart.bar(ratio.keys(),ratio.values(),color=color)

chart.set\_ylabel("ratio")

chart.set\_xlabel("content Rating")

chart.grid()

fig.suptitle("Ratio")

chart.legend()

canvas = FigureCanvasTkAgg(fig, master=screen) # A tk.DrawingArea.

canvas.draw()

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

canvas.mpl\_connect("key\_press\_event", on\_key\_press)

# this is necessary on Windows to prevent

# Fatal Python Error: PyEval\_RestoreThread: NULL tstate

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()

#def \_2018():

# Years=[2010,2011,2012,2013,2014,2015,2016,2017,2018]

# listyear=[]

#

# for j in Years:

# quar1={1:0,2:0,3:0}

# quar2={4:0,5:0,6:0}

# quar3={7:0,8:0,9:0}

# quar4={10:0,11:0,12:0}

#

#

# for i in range(len(data)):

# if data['year'][i]== j:

# if data['month'][i] in quar1:

# quar1[data['month'][i]]+=data['Installs'][i]

# elif data['month'][i] in quar2:

# quar2[data['month'][i]]+=data['Installs'][i]

# elif data['month'][i] in quar3:

# quar3[data['month'][i]]+=data['Installs'][i]

# elif data['month'][i] in quar4:

# quar4[data['month'][i]]+=data['Installs'][i]

# if sum(quar1.values())>sum(quar2.values()) and sum(quar1.values())>sum(quar3.values()) and sum(quar1.values())>sum(quar4.values()):

# listyear.append(quar1)

# elif sum(quar2.values())>sum(quar3.values()) and sum(quar2.values())>sum(quar4.values()):

# listyear.append(quar2)

# elif sum(quar3.values())>sum(quar4.values()):

# listyear.append(quar3)

# else:

# listyear.append(quar4)

# return listyear

#

def question11():

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

screen.title("Apps vs Downloads")

w = 1000 # width for the window size

h = 600 # height for the window size

ws = screen.winfo\_screenwidth() # width of the screen

hs = screen.winfo\_screenheight() # height of the screen

x = (ws/2) - (w/2) # calculate x and y coordinates for the Tk window

y = (hs/2) - (h/2)

screen.geometry('%dx%d+%d+%d' % (w, h, x, y)) # set the dimensions of the screen and where it is placed

screen.configure(background='white') # configuring the window

Years=[2010,2011,2012,2013,2014,2015,2016,2017,2018]

data = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

data['Installs'] = data['Installs'].map(lambda x: x.rstrip('+'))

data['Installs'] = data['Installs'].map(lambda x: ''.join(x.split(',')))

#print(data['Installs'].head(5))

data['Installs'] = pd.to\_numeric(data['Installs'])

d = pd.DatetimeIndex(data['Last Updated'])

data['year'] = d.year

data['month'] = d.month

list\_year=[]

for j in Years:

quar1={1:0,2:0,3:0}

quar2={4:0,5:0,6:0}

quar3={7:0,8:0,9:0}

quar4={10:0,11:0,12:0}

for i in range(len(data)):

if data['year'][i]== j:

if data['month'][i] in quar1:

quar1[data['month'][i]]+=data['Installs'][i]

elif data['month'][i] in quar2:

quar2[data['month'][i]]+=data['Installs'][i]

elif data['month'][i] in quar3:

quar3[data['month'][i]]+=data['Installs'][i]

elif data['month'][i] in quar4:

quar4[data['month'][i]]+=data['Installs'][i]

if sum(quar1.values())>sum(quar2.values()) and sum(quar1.values())>sum(quar3.values()) and sum(quar1.values())>sum(quar4.values()):

list\_year.append(quar1)

elif sum(quar2.values())>sum(quar3.values()) and sum(quar2.values())>sum(quar4.values()):

list\_year.append(quar2)

elif sum(quar3.values())>sum(quar4.values()):

list\_year.append(quar3)

else:

list\_year.append(quar4)

print(list\_year)

#dict1={}

#for i in range(len(list\_year)):

# dict1.update({Years[i]:list\_year[i]})

#print(dict1)

list10=[]

Month1,Month2,Month3 = [],[],[]

for i in range(len(list\_year)):

list2=[]

for j in (list\_year[i].keys()):

print(j)

list2.append(j)

list10.append(list2)

#print(list10)

for j in range(1):

for i in range(len(list10)):

Month1.append(list10[i][j])

for j in range(1,2):

for i in range(len(list10)):

Month2.append(list10[i][j])

for j in range(2,3):

for i in range(len(list10)):

Month3.append(list10[i][j])

print(Month1)

print("---------")

print(Month2)

print("---------")

print(Month3)

print("---------")

list1=[]

for i in range(len(list\_year)):

list2=[]

for j in (list\_year[i].values()):

print(j)

list2.append(j)

list1.append(list2)

Years = []

for i in range(2010,2019):

Years.append(str(i))

Quatmonth\_list=[]

for j in range(0,3):

list2=[]

for i in range(len(list1)):

list2.append(list1[i][j])

Quatmonth\_list.append(list2)

pos = np.arange(len(Years))

bar\_width = 0.3

figure2 = plt.Figure(figsize=(10,4), dpi=100)

chart = figure2.add\_subplot(111)

bar1 = chart.bar(Years,Quatmonth\_list[0],bar\_width,color='green',edgecolor='black')

bar2 = chart.bar(pos+bar\_width,Quatmonth\_list[1],bar\_width,color='yellow',edgecolor='black')

bar3 = chart.bar(pos+bar\_width\*2,Quatmonth\_list[2],bar\_width,color='red',edgecolor='black')

chart.set\_ylabel("Installs")

chart.set\_xlabel('Years')

figure2.suptitle('Group Barchart - Quater Month across the year',fontsize=18)

for idx,rect in enumerate(bar1):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,Month1[idx],ha='center', va='bottom', rotation=0)

for idx,rect in enumerate(bar2):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,Month2[idx],ha='center', va='bottom', rotation=0)

for idx,rect in enumerate(bar3):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,Month3[idx],ha='center', va='bottom', rotation=0)

canvas = FigureCanvasTkAgg(figure2, master=screen)

canvas.get\_tk\_widget().place(x=0,y=100)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

String="In the above Graph Quarter of each Year with their Higher Installs are plotted From 2010 to 2018"

tk.Label(screen,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=10,y=520)

screen.mainloop()

def newRelation1(app,x,y):

global dict\_app\_relation

for i in x:

if i==-999:

x.remove(i)

y.remove(i)

if x==[] or y==[]:

return

data = pd.DataFrame({'Sentiment\_pol':y , 'Sentiment\_sub': x})

val = data['Sentiment\_pol'].corr(data['Sentiment\_sub'])

dict\_app\_relation[app] = val

def function\_q13():

global screen,df,dict\_app\_relation

dict\_app\_relation={}

root = Tk()

root.iconbitmap(r"C:\\InternshipFinal\\google.ico")

big\_frame = tk.Frame(root,bg='white',width='700',height='630',bd=4,relief=RIDGE)

big\_frame.place(x=50,y=60)

w=700

h=600

ws=root.winfo\_screenwidth()

hs=root.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

df = pd.read\_csv("C:\\InternshipFinal\\user.csv")

df=df.replace(np.NaN,-999)

dict\_app\_index\_count={}

for index in range(len(df['App'])):

app = df['App'][index]

if app in dict\_app\_index\_count:

dict\_app\_index\_count[app][1]+=1

else:

dict\_app\_index\_count[app]=[index,1]

# after this for loop dict\_app\_index\_count will hold the app name as key and it's first index in data set and total count in data set as item

for app in dict\_app\_index\_count:

index = dict\_app\_index\_count[app][0]

count = dict\_app\_index\_count[app][1]

sub,pol=[],[]

for i in range(count):

c = index+i

sub.append(df['Sentiment\_Subjectivity'][c])

pol.append(df['Sentiment\_Polarity'][c])

newRelation1(app,sub,pol)

app\_no = np.arange(len(dict\_app\_relation.keys()))

relation = []

for i in dict\_app\_relation:

relation.append(dict\_app\_relation[i])

figure3 = plt.Figure(figsize=(6,4), dpi=100)

ax3 = figure3.add\_subplot(111)

ax3.scatter(app\_no,relation, color = '#102131')

scatter3 = FigureCanvasTkAgg(figure3, root)

scatter3.get\_tk\_widget().place(x=50,y=45)

ax3.grid()

ax3.set\_xlabel("Applications in sequence")

ax3.set\_ylabel("Correlation")

ax3.set\_title("The Co-rrelation for Polarity V/s Subjectivity for all apps")

toolbar = NavigationToolbar2Tk(scatter3,root)

toolbar.update()

String = """

In this Scatter plot each point represent the correlation

between sentiment polarity and sentiment subjectivity And

Most of apps have positive relation with between sentiment polarity and subjectivity

"""

tk.Label(root,text=String,font=("Calibri",13,'italic'),fg='#102131',bg='white').place(x=0,y=420)

root.mainloop()

def functq17():

global screen

screen = Tk()

screen.iconbitmap(r"C:\\InternshipFinal\\google.ico")

w = 600 # width for the window size

h = 700 # height for the window size

ws = screen.winfo\_screenwidth() # width of the screen

hs = screen.winfo\_screenheight() # height of the screen

x = (ws/2) - (w/2) # calculate x and y coordinates for the Tk window

y = (hs/2) - (h/2)

screen.geometry('%dx%d+%d+%d' % (w, h, x, y)) # set the dimensions of the screen and where it is placed

screen.resizable(False, False) # disabling the resize option for the window

screen.configure(background='white') # configuring the window

df= pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

list2=['More than 30 mb','20-30 mb','10-20 mb','Less Than 10 mb']

df['Size'] = df['Size'].map(lambda x: x.rstrip('M'))

df['Size'] = df['Size'].map(lambda x: str(round((float(x.rstrip('k'))/1024), 1)) if x[-1]=='k' else x)

df['Size'] = df['Size'].map(lambda x: np.nan if x.startswith('Varies') else x)

df['Size']=df['Size'].replace(np.NaN,-999)

df['Size']=df['Size'].astype(float)

#print(df['Category'].unique())

#print(df['Size'])

df['Installs']=df['Installs'].str.replace('+','')

df['Installs']=df['Installs'].str.replace(',','')

df['Installs']=df['Installs'].astype(int)

dict1,dict2,dict3,dict4,dict5,dict6={},{},{},{},{},{}

a,b,c,d=[],[],[],[]

for i in range(len(df)):

if df["Size"][i]>=30:

a.append(df['Installs'][i])

elif 20<=df["Size"][i]<30:

b.append(df['Installs'][i])

elif 10<=df["Size"][i]<20:

c.append(df['Installs'][i])

elif (df['Size'][i]<10):

d.append(df['Installs'][i])

a2=(sum(b))

a3=(sum(c))

a1=(sum(a))

a4=(sum(d))

list1=[a1,a2,a3,a4]

print(list1)

color = cm.rainbow(np.linspace(0, 2, 10))

fig=Figure(figsize=(3,2),dpi=100)

chart=fig.add\_subplot(111)

chart.bar(list2,list1,color=color)

chart.set\_ylabel("No of Installs")

chart.set\_xlabel("Sizes")

chart.grid()

fig.suptitle("No. of Installs Vs Size")

canvas = FigureCanvasTkAgg(fig, screen) # A tk.DrawingArea.

canvas.get\_tk\_widget().pack(side=TOP, fill=BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, screen)

toolbar.update()

canvas.mpl\_connect("key\_press\_event", on\_key\_press)

Label(screen,text="From The Above Bar Graph We Say That the Size of the App Does Influence The No of Installs \nAnd the Trend is Positive As The Install Increases with Size",font=("Lucida",10,'bold')).place(x=0,y=610)

button = Button(master=screen, text="Quit", command=\_quit)

button.pack(side=BOTTOM)

screen.mainloop()

def month(x):

if x[0:3]=='Jan':

return 1

elif x[0:3]=='Feb':

return 2

elif x[0:3]=='Mar':

return 3

elif x[0:3]=='Apr':

return 4

elif x[0:3]=='Ma' or x[0:3]=='May':

return 5

elif x[0:3]=='Jun':

return 6

elif x[0:3]=='Jul':

return 7

elif x[0:3]=='Aug':

return 8

elif x[0:3]=='Sep':

return 9

elif x[0:3]=='Oct':

return 10

elif x[0:3]=='Nov':

return 11

elif x[0:3]=='Dec':

return 12

def install():

global sample

Installs=[]

for i in sample['Installs']: #converting string based installs into integer based

if i=='Free':

Installs.append(0)

else:

Installs.append(int(i.replace('+','').replace(',','')))

return Installs

def dates\_str\_to\_int():

global sample

dates=sample['Last Updated']

year=[]

counter=0

for i in dates:

year.append([int(i[-8:-6]),month(i[:-9]),int(i[-4:])])

counter=counter+1

return year

def display(x,y,z):

for i in x:

for j in set(i):

y.insert('end',j)

def filtering(value,canvas\_listbox):

global sample

installs=install()

year=dates\_str\_to\_int()

rating=sample['Rating']

category=sample['Category'].unique()

ans=[]

for i in category:

ans.append([])

for i in range(len(installs)):

if i!=10472 and installs[i]==value[0]:

if rating[i]>=value[1]:

if year[i][2]==value[2]:

for j in range(len(category)):

if category[j]==sample['Category'][i] :

ans[j].append(sample['App'][i])

canvas\_listbox.delete(0,'end')

display(ans,canvas\_listbox,category)

def getting(install,rating,year,category,canvas\_listbox):

if install.get().strip()!='' and rating.get().strip()!='' and year.get().strip()!='' and category.get().strip()!='':

value=[int(install.get().replace(',','').replace('+','')),float(rating.get()),int(year.get()),str(category.get())]

filtering(value,canvas\_listbox)

else:

tk.messagebox.showerror('Error','Please select values')

def searchapp():

global screen,sample

sample = pd.read\_csv("C:\\InternshipFinal\\App-data.csv")

screen = tk.Tk()

w=1300

h=730

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

screen.geometry("%dx%d+%d+%d"%(w,h,x,y))

category=list(sample['Category'].unique())

big\_frame = tk.Frame(screen,bg='#102131',width='1300',height='730')

big\_frame.place(x=0,y=0)

sample.drop(index=[10472],inplace=True)

sample=sample.replace(np.NaN,0)

year=[2010,2011,2012,2013,2014,2015,2016,2017,2018]

rating=[]

for i in range(5):

for j in range(10):

rating.append(i+(j/10))

rating.append(5.0)

tk.Label(big\_frame,text='Installs',width=10,height=1,font=("Helvetica",15,'bold'),fg='#ffffff',bg='#000000', borderwidth=2, relief="groove").place(x=550,y=60)

tk.Label(big\_frame,text='Rating',width=10,height=1,font=("Helvetica",15,'bold'),fg='#ffffff',bg='#000000', borderwidth=2, relief="groove").place(x=350,y=60)

tk.Label(big\_frame,text='Year',width=10,height=1,font=("Helvetica",15,'bold'),fg='#ffffff',bg='#000000', borderwidth=2, relief="groove").place(x=150,y=60)

tk.Label(big\_frame,text='Category',width=10,height=1,font=("Helvetica",15,'bold'),fg='#ffffff',bg='#000000', borderwidth=2, relief="groove").place(x=750,y=60)

combo\_category=ttk.Combobox(big\_frame,width=17,values=category,state="readonly")

combo\_category.place(x=750,y=110)

combo\_install=ttk.Combobox(big\_frame,width=17,values=['0','10+','100+','1,000+','10,000+','1,00,000+','10,00,000+','1,00,00,000+'],state="readonly")

combo\_install.place(x=550,y=110)

combo\_rating=ttk.Combobox(big\_frame,width=17,values=rating,state="readonly")

combo\_rating.place(x=350,y=110)

combo\_year=ttk.Combobox(big\_frame,width=17,values=year,state="readonly")

combo\_year.place(x=150,y=110)

canvas=tk.Canvas(big\_frame,width=970,height=450,bg='pink')

canvas.place(x=150,y=150)

scroll1=tk.Scrollbar(canvas)

canvas\_listbox=tk.Listbox(canvas,yscrollcommand = scroll1.set,height=20,width=96,bg='#A9D0F5',font=('Calibri',14,'bold'))

canvas\_listbox.pack( side = 'left', fill = 'both' )

scroll1.pack(side='right', fill='y' )

scroll1.config( command = canvas\_listbox.yview )

btn\_search=tk.Button(big\_frame,text='Search',height=1,font=("Helvetica",15,'bold'),fg="white",width=15,bg="black",command=lambda:getting(combo\_install,combo\_rating,combo\_year,combo\_category,canvas\_listbox))

btn\_search.place(x=1020,y=85)

screen.mainloop()