Section 1 :

System Requirement Specification//

Design app which shows different analysis of the given dataset.

The different requirements are:

1) What is the percentage download in each star on the playstore ?

2) How many apps have managed to 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

3) Which category of apps have managed to get the most, least and an average of 250,000 downloads at least ?

4) Which category of apps have managed to get the highest maximum average ratings from the users ?

5) What is the download trend category wise over the period for which the data is being made vailabile ?

6) For the years 201620172018 What are the category of apps that have got the most and the least downloads. What is the percentage increase or decrease that the apps have got over the period of three years ?

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

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 prediction and back it with suitable findings ?

9) All those apps who have managed to it over 100.000 downloads, have they managed to get an average rating of 4.1 and above? Can we conclude something in correlation to the number of downloads and the ratings received ?

10) Across all the years which month has seen the maximum downloads for each of the category. What is the ratio of downloads for the app that qualifies asteen versus mature 17?

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

12) Which of all the apps gwen 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.

13) Study and find out the relation between the Sentiment polarity and sentiment subjectivity af all the apps.

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 able to launch an app like 10 Best foods for you? Do users like these apps ?

16) Which month(of the year, is the best indicator to the average downloads that an app will generate over the entire year ?

17) 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.

18) Provide an interface to add new data to both the datasets provided.

EXTRAS:

1. Cleaning of Data
2. Login and Register with Validation
3. Top Free,Paid,Trending Apps
4. Comparison of 2 Apps of the same Category

Section 2 :

Technology used

Anaconda version 1.9.7
Spyder version 3.3.2
Processor : Core i3
Ram : 4GB
OS: Windows 10
Wamp Server: Phpmyadmin

ANACONDA
Anaconda is the installation program used by Fedora, Red Hat Enterprise Linux and some other distributions. During installation, a target computer's hardware is identified and configured and the appropriate file systems for the system's architecture are created. Finally, anaconda allows the user to install the operating system software on the target computer. Anaconda can also upgrade existing installations of earlier versions of the same distribution. After the installation is complete, you can reboot into your installed system and continue doing customization using the initial setup program. Anaconda is a fairly sophisticated installer. It supports installation from local and remote sources such as CDs and DVDs, Images stored on a hard drive, NFS, HTTP, and FTP. Installation can be scripted with kickstart to provide a fully unattended installation that can be duplicated on scores of machines. It can also be run over VNC on headless machines. A variety of advanced storage devices including LVM, RAID, ISCSI, and multipath are supported from the partitioning program. Anaconda provides advanced debugging features such as remote logging, access to the python interactive debugger, and remote saving of exception dumps.

SPYDER

Spyder is the Scientific Python Development Environment:

- A powerful interactive development environment for the Python language with advanced editing, interactive testing, debugging and introspection features.
- and a numerical computing environment thanks to the support of IPython and popular Python libraries such as NumPy,matplotlib,canvas,etc.
- Spyder may also be used as a library providing powerful console-related widgets for you based applications for example, it may be used to integrate a debugging console directly the layout of your graphical user interface**.**

Python Libraries:

1. pandas
2. matplotlib
3. canvas
4. seaborn

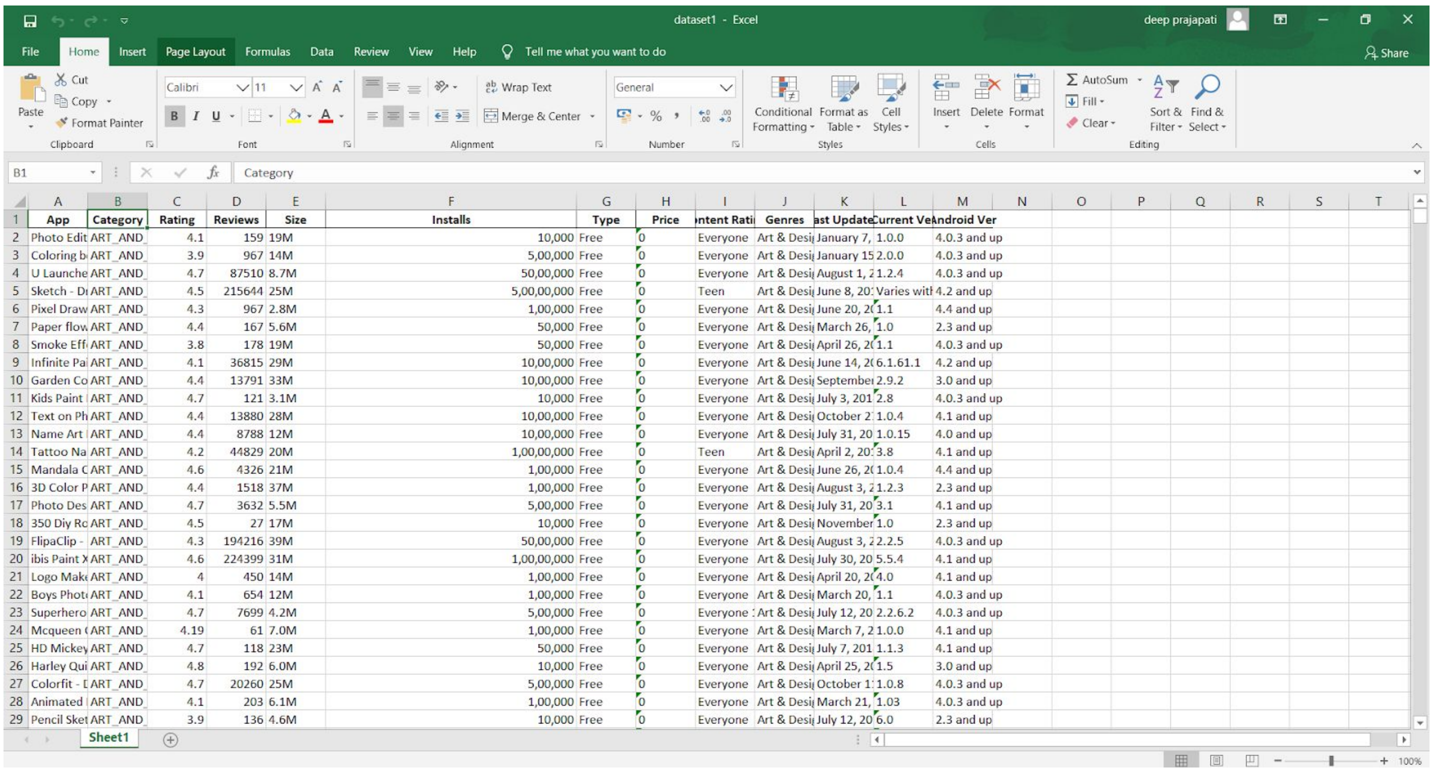
Ram : 4GB
It is used for fast processing of large amount of dataset.

Wamp Server:

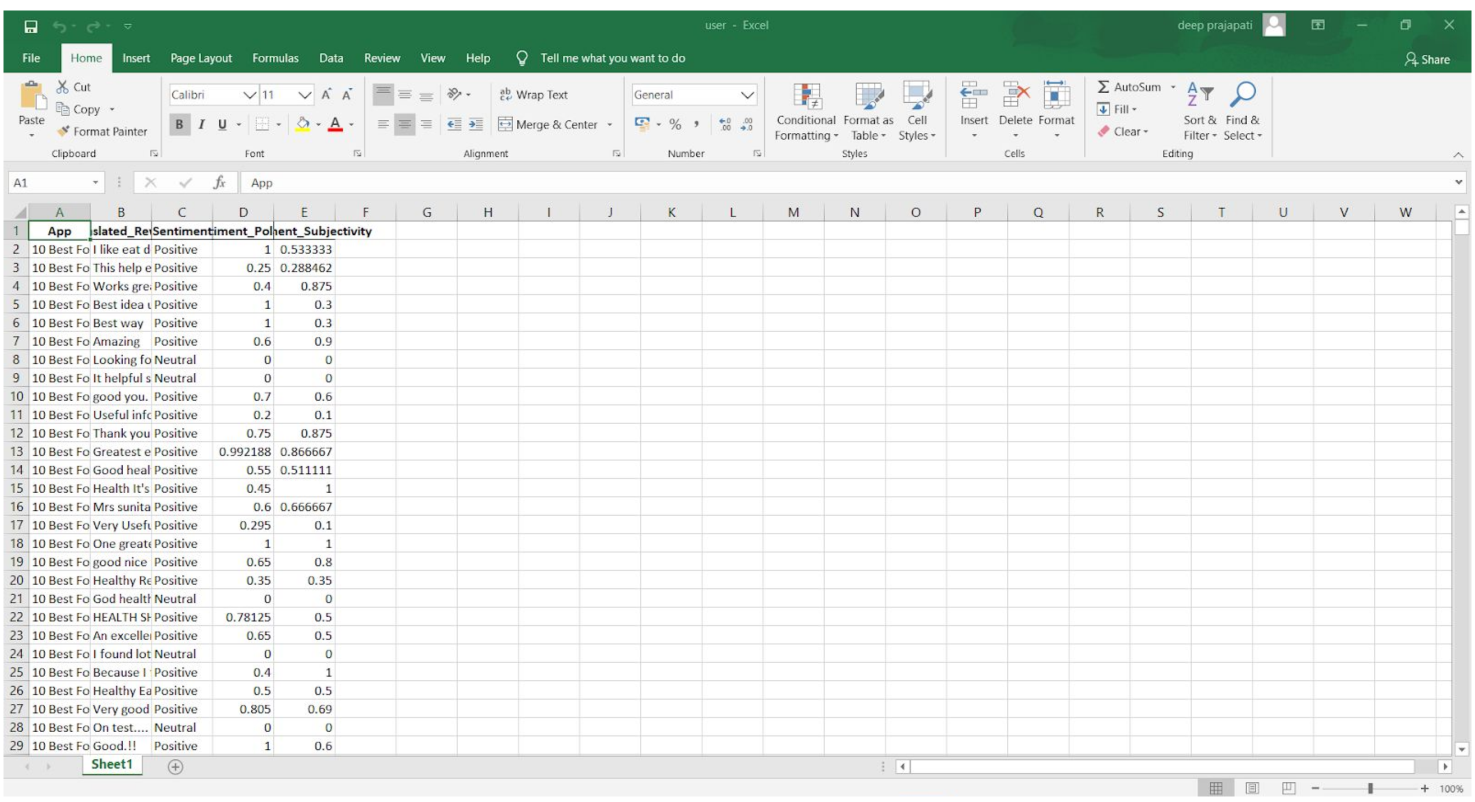
1. PhpMyadmin
2. It provides a separate database maintaining environment. Here we create different database according to our need and work accordingly

Section 3 :

Data Provided by the client:



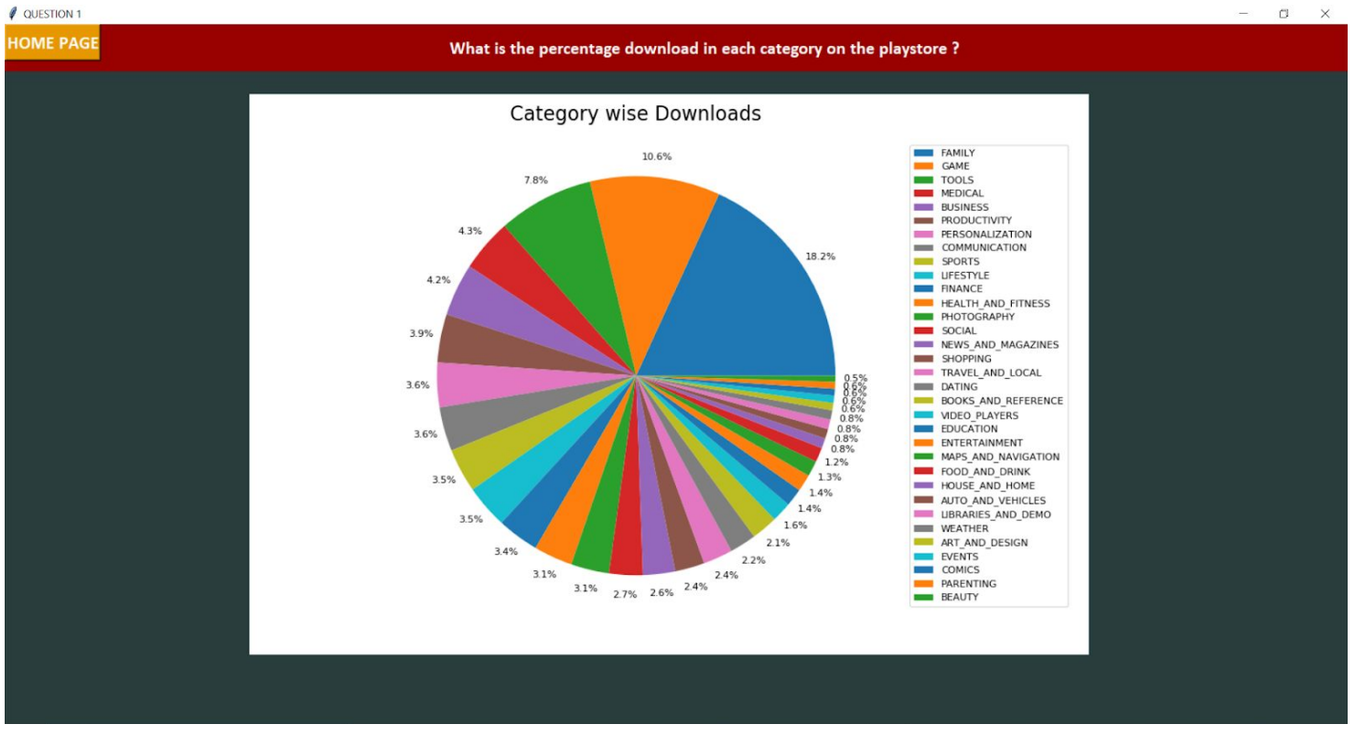
	App	Category	Rating	Reviews	Size	Installs	Type	Price	›ntent Rati	Genres	ast Update	Current Ve	Android Ver
2	Photo Edit	ART_AND_	4.1	159	19M	10,000	Free	0	Everyone	Art & Desi	January 7,	1.0.0	4.0.3 and up
3	Coloring b	ART_AND_	3.9	967	14M	5,00,000	Free	0	Everyone	Art & Desi	January 15	2.0.0	4.0.3 and up
4	U Launche	ART_AND_	4.7	87510	8.7M	50,00,000	Free	0	Everyone	Art & Desi	August 1, 2	1.2.4	4.0.3 and up
5	Sketch - D	ART_AND_	4.5	215644	25M	5,00,00,000	Free	0	Teen	Art & Desi	June 8, 20	Varies wit	4.2 and up
6	Pixel Draw	ART_AND_	4.3	967	2.8M	1,00,000	Free	0	Everyone	Art & Desi	June 20, 2(	1.1	4.4 and up
7	Paper flow	ART_AND_	4.4	167	5.6M	50,000	Free	0	Everyone	Art & Desi	March 26,	1.0	2.3 and up
8	Smoke Eff	ART_AND_	3.8	178	19M	50,000	Free	0	Everyone	Art & Desi	April 26, 2(	1.1	4.0.3 and up
9	Infinite Pa	ART_AND_	4.1	36815	29M	10,00,000	Free	0	Everyone	Art & Desi	June 14, 2(	6.1.61.1	4.2 and up
10	Garden Co	ART_AND_	4.4	13791	33M	10,00,000	Free	0	Everyone	Art & Desi	September	2.9.2	3.0 and up
11	Kids Paint	ART_AND_	4.7	121	3.1M	10,000	Free	0	Everyone	Art & Desi	July 3, 201	2.8	4.0.3 and up
12	Text on Ph	ART_AND_	4.4	13880	28M	10,00,000	Free	0	Everyone	Art & Desi	October 2	1.0.4	4.1 and up
13	Name Art	ART_AND_	4.4	8788	12M	10,00,000	Free	0	Everyone	Art & Desi	July 31, 20	1.0.15	4.0 and up
14	Tattoo Na	ART_AND_	4.2	44829	20M	1,00,00,000	Free	0	Teen	Art & Desi	April 2, 20	3.8	4.1 and up
15	Mandala C	ART_AND_	4.6	4326	21M	1,00,000	Free	0	Everyone	Art & Desi	June 26, 2(	1.0.4	4.4 and up
16	3D Color P	ART_AND_	4.4	1518	37M	1,00,000	Free	0	Everyone	Art & Desi	August 3, 2	1.2.3	2.3 and up
17	Photo Des	ART_AND_	4.7	3632	5.5M	5,00,000	Free	0	Everyone	Art & Desi	July 31, 20	3.1	4.1 and up
18	350 Diy Ro	ART_AND_	4.5	27	17M	10,000	Free	0	Everyone	Art & Desi	November	1.0	2.3 and up
19	FlipaClip -	ART_AND_	4.3	194216	39M	50,00,000	Free	0	Everyone	Art & Desi	August 3, 2	2.2.5	4.0.3 and up
20	ibis Paint X	ART_AND_	4.6	224399	31M	1,00,00,000	Free	0	Everyone	Art & Desi	July 30, 20	5.5.4	4.1 and up
21	Logo Make	ART_AND_	4	450	14M	1,00,000	Free	0	Everyone	Art & Desi	April 20, 2(	4.0	4.1 and up
22	Boys Phot	ART_AND_	4.1	654	12M	1,00,000	Free	0	Everyone	Art & Desi	March 20,	1.1	4.0.3 and up
23	Superhero	ART_AND_	4.7	7699	4.2M	5,00,000	Free	0	Everyone	Art & Desi	July 12, 20	2.2.6.2	4.0.3 and up
24	Mcqueen (	ART_AND_	4.19	61	7.0M	1,00,000	Free	0	Everyone	Art & Desi	March 7, 2	1.0.0	4.1 and up
25	HD Mickey	ART_AND_	4.7	118	23M	50,000	Free	0	Everyone	Art & Desi	July 7, 201	1.1.3	4.1 and up
26	Harley Qui	ART_AND_	4.8	192	6.0M	10,000	Free	0	Everyone	Art & Desi	April 25, 2(	1.5	3.0 and up
27	Colorfit - [	ART_AND_	4.7	20260	25M	5,00,000	Free	0	Everyone	Art & Desi	October 1	1.0.8	4.0.3 and up
28	Animated	ART_AND_	4.1	203	6.1M	1,00,000	Free	0	Everyone	Art & Desi	March 21,	1.03	4.0.3 and up
29	Pencil Sket	ART_AND_	3.9	136	4.6M	10,000	Free	0	Everyone	Art & Desi	July 12, 20	6.0	2.3 and up



	App	slated_Re	Sentiment	timent_Pol	nent_Subjectivity
2	10 Best Fo	I like eat d	Positive	1	0.533333
3	10 Best Fo	This help e	Positive	0.25	0.288462
4	10 Best Fo	Works gre	Positive	0.4	0.875
5	10 Best Fo	Best idea u	Positive	1	0.3
6	10 Best Fo	Best way	Positive	1	0.3
7	10 Best Fo	Amazing	Positive	0.6	0.9
8	10 Best Fo	Looking fo	Neutral	0	0
9	10 Best Fo	It helpful s	Neutral	0	0
10	10 Best Fo	good you.	Positive	0.7	0.6
11	10 Best Fo	Useful info	Positive	0.2	0.1
12	10 Best Fo	Thank you	Positive	0.75	0.875
13	10 Best Fo	Greatest e	Positive	0.992188	0.866667
14	10 Best Fo	Good heal	Positive	0.55	0.511111
15	10 Best Fo	Health It's	Positive	0.45	1
16	10 Best Fo	Mrs sunita	Positive	0.6	0.666667
17	10 Best Fo	Very Usefu	Positive	0.295	0.1
18	10 Best Fo	One great	Positive	1	1
19	10 Best Fo	good nice	Positive	0.65	0.8
20	10 Best Fo	Healthy Re	Positive	0.35	0.35
21	10 Best Fo	God health	Neutral	0	0
22	10 Best Fo	HEALTH SH	Positive	0.78125	0.5
23	10 Best Fo	An excelle	Positive	0.65	0.5
24	10 Best Fo	I found lot	Neutral	0	0
25	10 Best Fo	Because I	Positive	0.4	1
26	10 Best Fo	Healthy Ea	Positive	0.5	0.5
27	10 Best Fo	Very good	Positive	0.805	0.69
28	10 Best Fo	On test....	Neutral	0	0
29	10 Best Fo	Good.!!	Positive	1	0.6

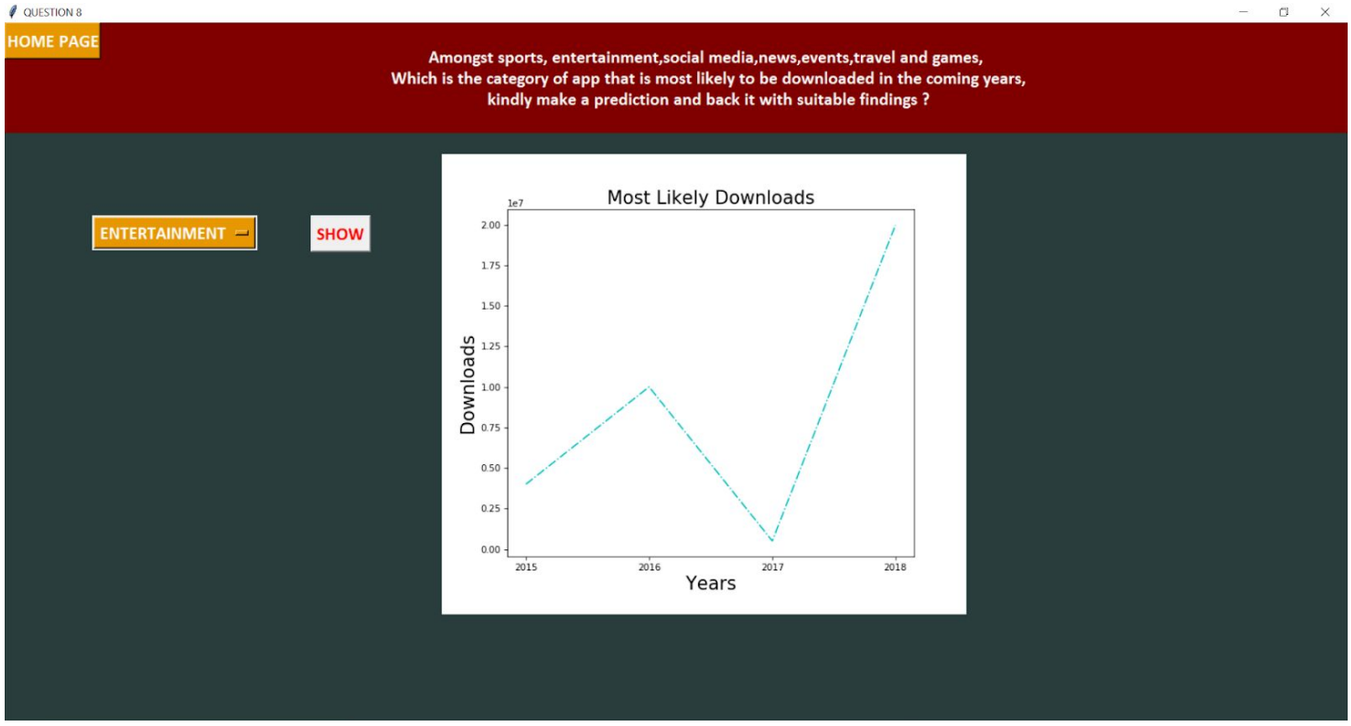
Section 4 :

ScreenShots



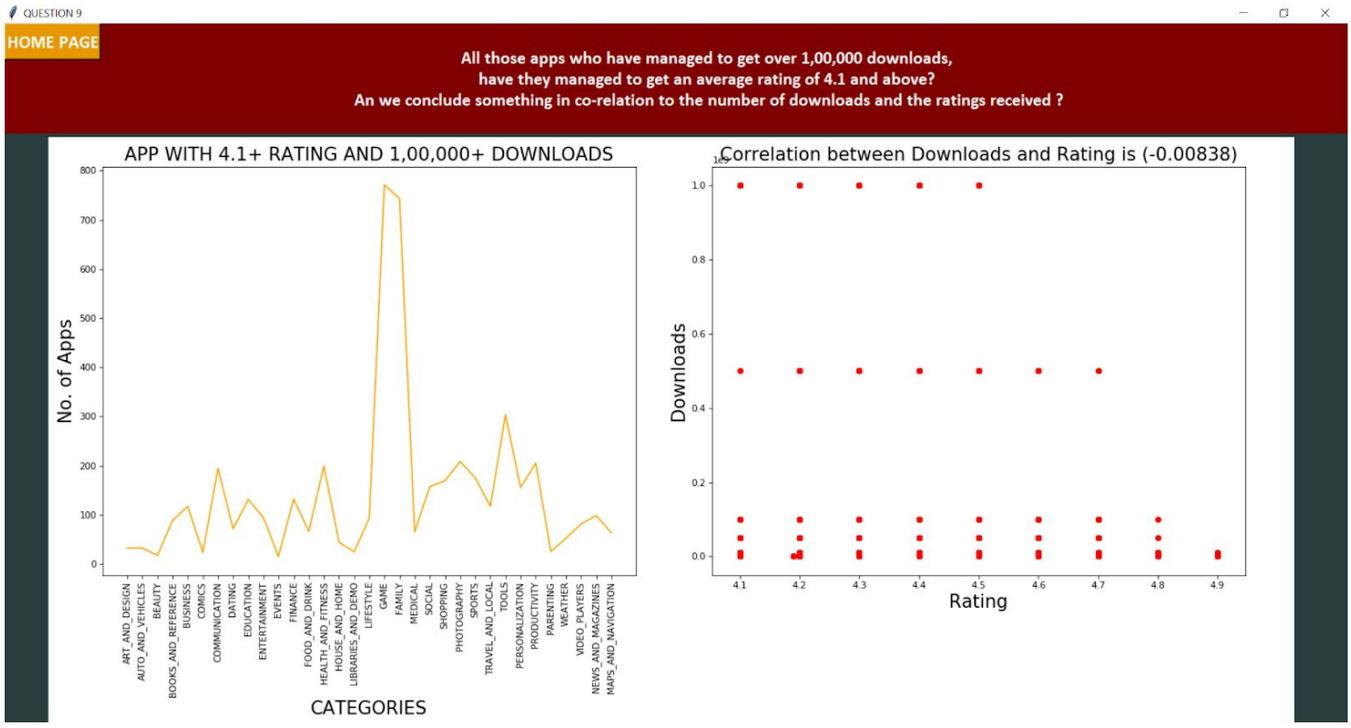
Code for Category-wise Downloads of Apps:

```
def fig1():

    fig1 = Toplevel(home)

    fig1.title("QUESTION 1")

    createWindow(fig1)

    Label(fig1, text="What is the percentage download in each category on the playstore ? ", width="160", height="2",font=Label1_font, fg='white', bg='#990000').place(x=0, y=0)

    b1 = Button(fig1, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font, fg='white',command=backtohome).place(x=0, y=0)

    f = Figure(figsize=(12, 8), dpi=80)

    a = f.add_subplot(111)

    a.pie(df['Category'].value_counts().values, autopct='%1.1f%%',pctdistance = 1.1 , labeldistance = 1.2)

    a.set_title("Category wise Downloads",fontsize=20)

    a.legend(df['Category'].value_counts().index, loc='center left', bbox_to_anchor=(1.04, 0.5), ncol=1)

    canvas = FigureCanvasTkAgg(f, fig1)

    canvas.get_tk_widget().place(x=280, y=80)

    canvas.draw()

    f.tight_layout()
```



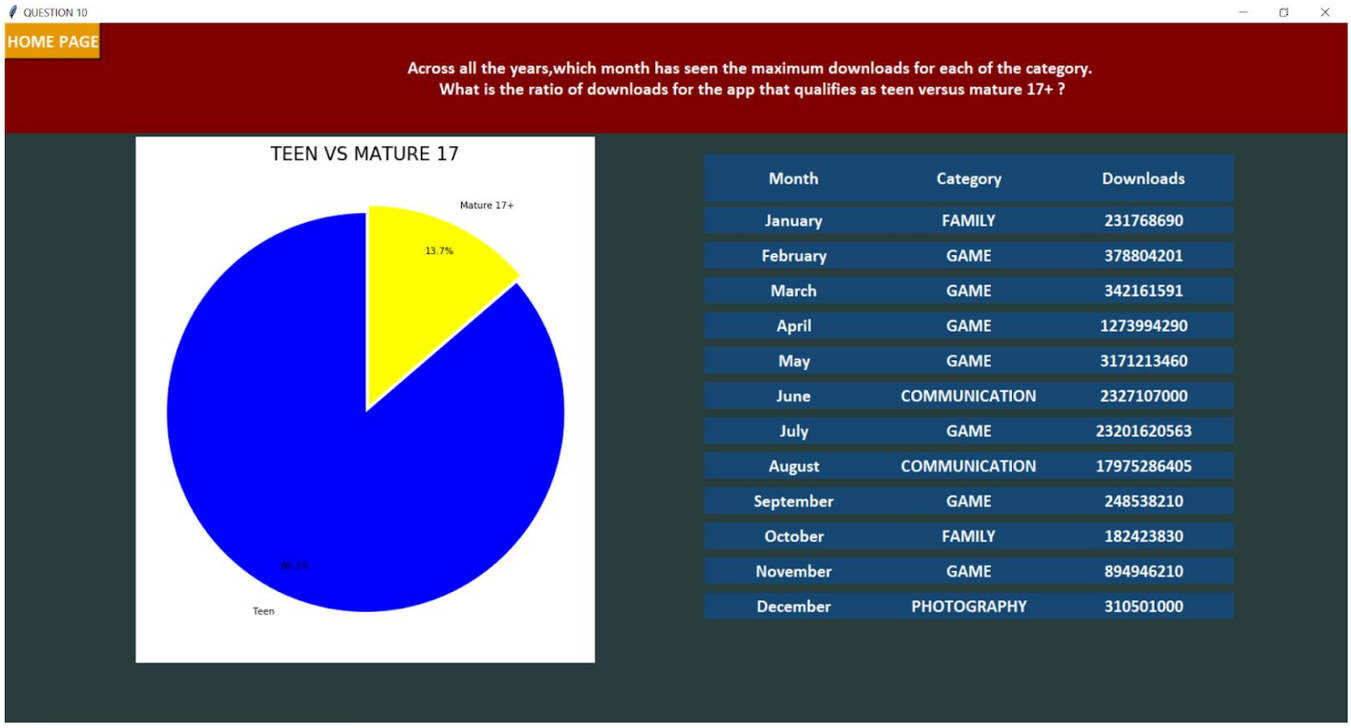
Code for Year wise trend of downloads of categories :

```
def fig8():
    fig8 = Toplevel(home)
    fig8.title("QUESTION 8")
    createWindow(fig8)
    Label(fig8,text="Amongst sports, entertainment,social media,news,events,travel and games,\n
 Which is the category of app that is most likely to be downloaded in the coming years,\n kindly make
 a prediction and back it with suitable findings ?",width="160", height="5", font=Label1_font,
 fg='white', bg='#800000').place(x=0, y=0)
    b1 = Button(fig8, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command=backtohome).place(x=0, y=0)
    # Now starting with the drop down list
    OPTIONS =
['SPORTS','ENTERTAINMENT','SOCIAL','NEWS_AND_MAGAZINES','EVENTS','TRAVEL_AND_LOCA
L','GAME']
    variable = StringVar(fig8)
    variable.set('CATEGORY')
    w = OptionMenu(fig8, variable, *OPTIONS)
    w.place(x=100, y=220)
    w.configure(bg="#e79700", fg='white', height="1", font=Button1_font)
    b = Button(fig8, text='SHOW', fg='red', height="1", font=Button1_font, command=lambda:
 new_plot(TrendDict, variable.get(),fig8))
    b.place(x=350, y=220)
```



Code for Apps with rating above 4.1 and more than 100000 downloads :

```
def fig9():
    fig9 = Toplevel(home)
    fig9.title("QUESTION 9")
    createWindow(fig9)
    Label(fig9,text="All those apps who have managed to get over 1,00,000 downloads,\n have they
  managed to get an average rating of 4.1 and above? \n An we conclude something in co-relation to
  the number of downloads and the ratings received ?",width="160", height="5", font=Label1_font,
  fg='white', bg='#800000').place(x=0, y=0)
    b1 = Button(fig9, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
  fg='white',command=backtohome).place(x=0, y=0)
    x=dataframe[['Category','Rating','Installs']]
    z=x.to_dict(orient='split')
    cat =dataframe['Category'].unique()
    cat=cat.tolist()
    d1=[]
    for i in cat:
        count=0
        for j in range(len(z['data'])):
            if(i==z['data'][j][0]):
                if(z['data'][j][1]>=4.1 and z['data'][j][2]>=100000):
                    count+=1
        d1.append(count)
    a = x[['Rating','Installs']]
    rat = a[(a['Rating']>=4.1) & (a['Installs']>=100000)]
    corr = rat['Installs'].corr(rat['Rating'])
    figure9 = py.Figure(figsize = (19,9) , dpi = 75)
    ax9 = figure9.add_subplot(1,2,1)
    ax9.set_title('APP WITH 4.1+ RATING AND 1,00,000+ DOWNLOADS',fontsize=20)
    ax9.plot(cat,d1,color='orange')
    ax9.set_xticklabels(cat,rotation=90,ha='center')
    ax9.set_ylabel('DOWNLOADS',fontsize=20)
    ax9.set_xlabel('CATEGORIES',fontsize=20)
    ax9b = figure9.add_subplot(1,2,2)
    ax9b.scatter(rat['Rating'],rat['Installs'],marker="8",c='r')
    ax9b.set_ylabel("Downloads",fontsize=20)
    ax9b.set_xlabel("Rating",fontsize=20)
    ax9b.set_title("Correlation between Downloads and Rating is (-0.00838)",fontsize=20)
    canvas = FigureCanvasTkAgg(figure9, fig9)
    canvas.get_tk_widget().place(x=50, y=130)
    canvas.draw()
    figure9.tight_layout()
```

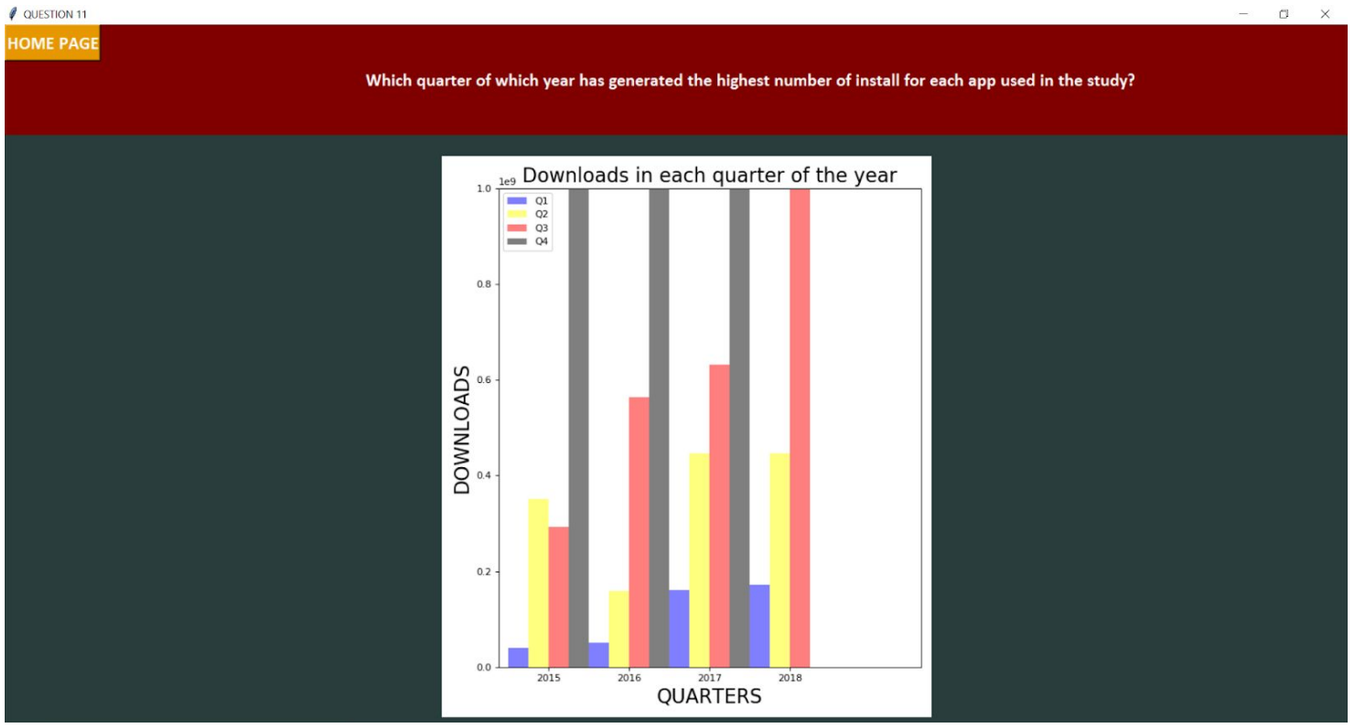


Month	Category	Downloads
January	FAMILY	231768690
February	GAME	378804201
March	GAME	342161591
April	GAME	1273994290
May	GAME	3171213460
June	COMMUNICATION	2327107000
July	GAME	23201620563
August	COMMUNICATION	17975286405
September	GAME	248538210
October	FAMILY	182423830
November	GAME	894946210
December	PHOTOGRAPHY	310501000

Code for distribution of Teen vs Mature 17 and month wise category downloads :

```
  fig10 = Toplevel(home)
  fig10.title("QUESTION 10")
  createWindow(fig10)
  Label(fig10,text="Across all the years,which month has seen the maximum downloads for each of
the category.\n What is the ratio of downloads for the app that qualifies as teen versus mature 17+
?",width="170", height="5", font=Label1_font, fg='white', bg='#800000').grid(row=0,column=0)
  cat = dataframe['Category'].unique()
  b1 = Button(fig10, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
fg='white',command=backtohome).place(x=0, y=0)
  cat=cat.tolist()
  x = dataframe[['Category','Installs','Month']]
  df = x.to_dict(orient='split')
  downloads = {}
  for i in cat:
    downloads[i] = [0]*13
  for i in cat:
    for j in range(len(df['data'])):
      if (i==df['data'][j][0]):
        month = df['data'][j][2]
        downloads[i][month] += df['data'][j][1]
  max_month = [0]*13
  max_category = [0]*13
  for j in range(1,13):
    ins = 0
    for i in cat:
      c = ''
      if(downloads[i][j] > ins):
        ins = downloads[i][j]
        c = i
        max_month[j] = ins
        max_category[j] = c
  max_month.pop(0)
  max_category.pop(0)
  l4=['Month','Category','Downloads']

  c=0
  for i in range(len(l4)):
    Label(fig10, text=l4[i],width="20", height="2", font=Label1_font, fg='white',
bg='#174873').place(x=800+c, y=150)
    c+=200
  c=10
  for i in range(len(max_month)):
```

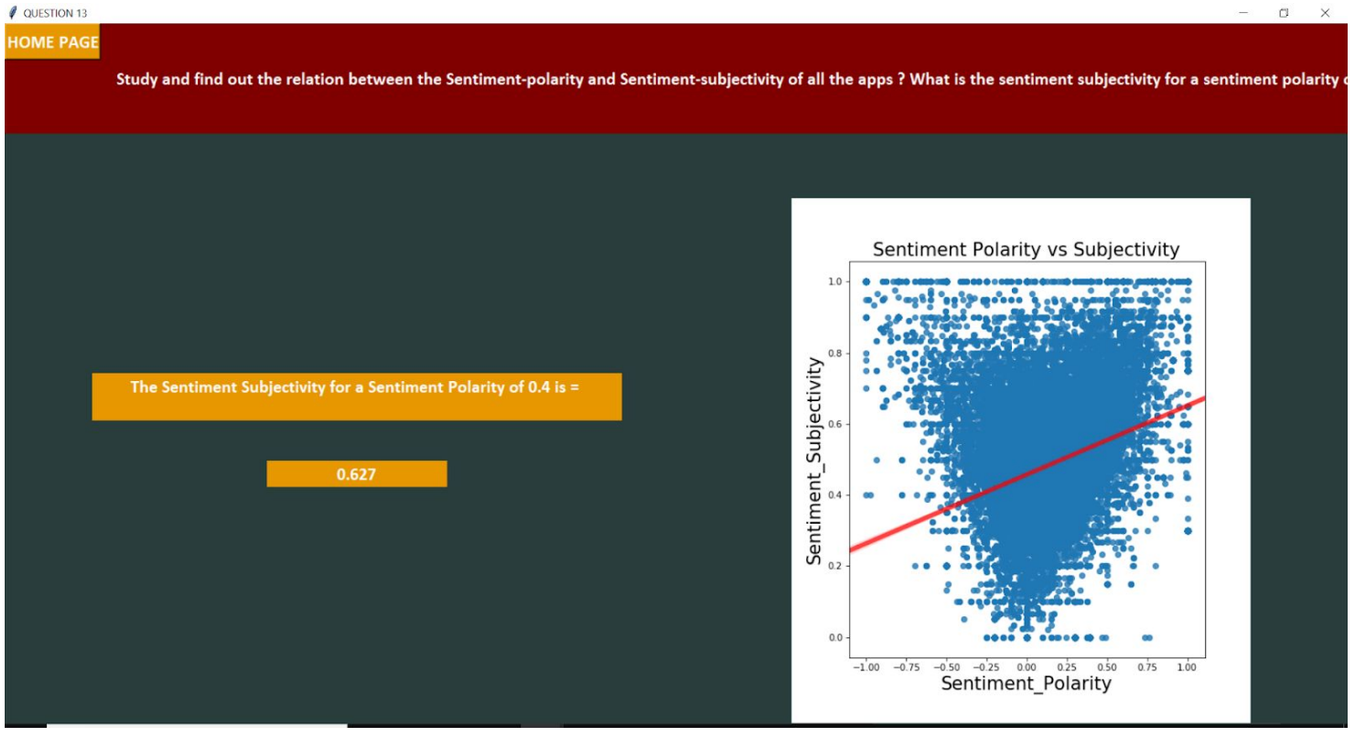
```
    Label(fig10, text=calendar.month_name[i+1],width="20", height="1", font=Label1_font, fg='white',
bg='#174873').place(x=800, y=200+c)
    c+=40
  c=10
  for i in range(len(max_month)):
    Label(fig10, text=max_category[i],width="20", height="1", font=Label1_font, fg='white',
bg='#174873').place(x=1000, y=200+c)
    c+=40
  c=10
  for i in range(len(max_month)):
    Label(fig10, text=max_month[i],width="20", height="1", font=Label1_font, fg='white',
bg='#174873').place(x=1200, y=200+c)
    c+=40
  df = dataframe[['Content Rating','Installs']]
  teen  = []
  mature = []
  for i in range(len(df)):
    if(df['Content Rating'].iloc[i]=="Teen"):
      teen.append(df['Installs'].iloc[i])
    elif(df['Content Rating'].iloc[i]=="Mature 17"):
      mature.append(df['Installs'].iloc[i])
  teen_t = sum(teen)
  mature_t = sum(mature)
  total = teen_t  + mature_t
  teen_t = round((teen_t/total)*100,3)
  mature_t = round((mature_t/total)*100,3)
  percentage = [teen_t,mature_t]
  explode = (0.01,0.03)
  figure10 = py.Figure(figsize = (7,8) , dpi = 75)
  ax9 = figure10.add_subplot(111)
  ax9.pie(percentage,labels=('Teen','Mature 17+'),colors =
('Blue','Yellow'),autopct="%1.1f%%",startangle=90, pctdistance=0.85, explode = explode)
  ax9.axis('equal')
  ax9.set_title("TEEN VS MATURE 17",fontsize=20)
  canvas = FigureCanvasTkAgg(figure10, fig10)
  canvas.get_tk_widget().place(x=150, y=130)
  canvas.draw()
  figure10.tight_layout()
```



Code for quarter wise distribution of downloads of apps across 4 years :

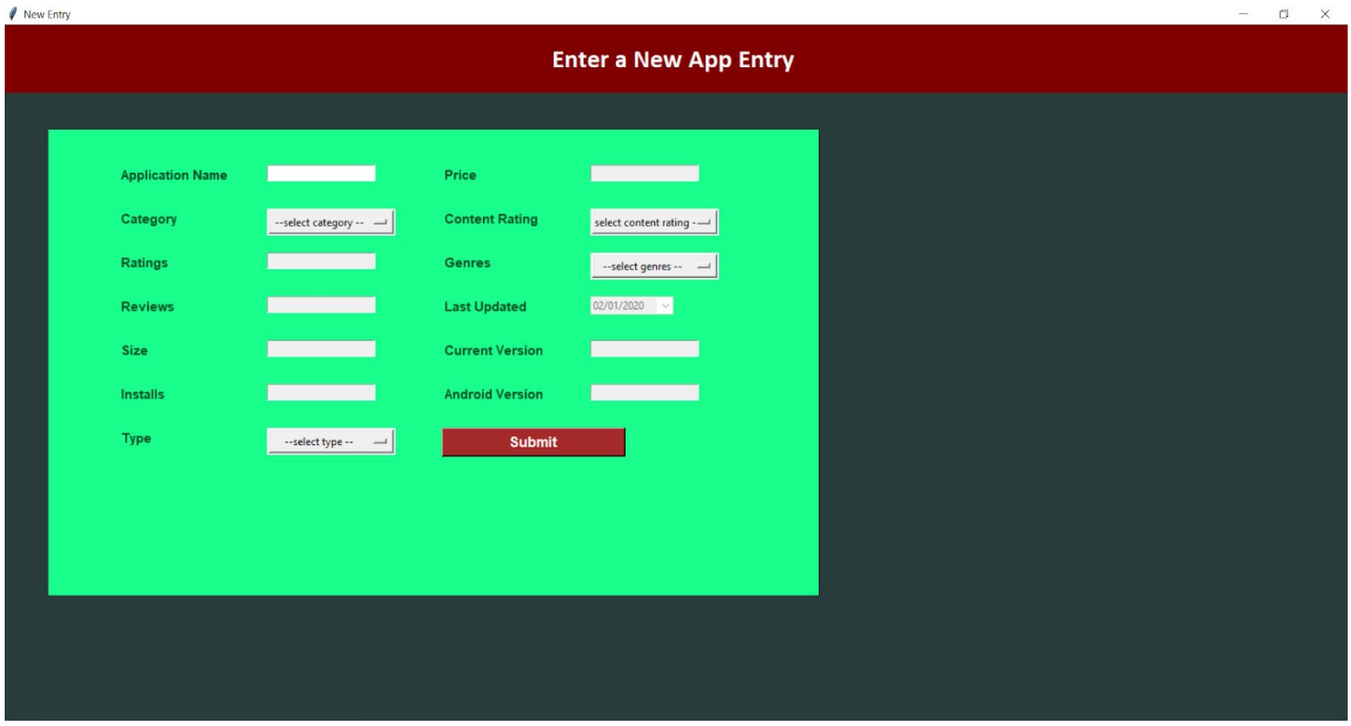
```
def fig11():
    fig11 = Toplevel(home)
    fig11.title("QUESTION 11")
    createWindow(fig11)
    Label(fig11,text="Which quarter of which year has generated the highest number of install for each
app used in the study?",width="170", height="5", font=Label1_font, fg='white',
bg='#800000').place(x=0, y=0)
    b1 = Button(fig11, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
fg='white',command=backtohome).place(x=0, y=0)
    cat =dataframe['Category'].unique()
    cat=cat.tolist()
    x = dataframe[['App','Category','Installs','Year','Month']]
    df = x.to_dict(orient='split')
    q1=[1,2,3]
    q2=[4,5,6]
    q3=[7,8,9]
    q4=[10,11,12]
    year = [2015,2016,2017,2018]
    quarter = {}
    for i in year:
      quarter[i] = [0]*4
    for i in range(len(x)):
      for j in year:
        if(df['data'][i][3] == j):
          if(df['data'][i][4] in q1):
            quarter[j][0] += df['data'][i][2]
          if(df['data'][i][4] in q2):
            quarter[j][1] += df['data'][i][2]
          if(df['data'][i][4] in q3):
            quarter[j][2] += df['data'][i][2]
          if(df['data'][i][4] in q4):
            quarter[j][3] += df['data'][i][2]
    q = ['Q1','Q2','Q3','Q4']
    figure11 = py.Figure(figsize = (7,8) , dpi = 80)
    ax11 = figure11.add_subplot(111)
    width = 0.25
    pos = list(range(len(quarter)))
    ax11.bar(pos,quarter[2015],width,alpha=0.5,color='blue',label='2015')
    ax11.bar([p + width for p in pos],quarter[2016],width,alpha=0.5,color='yellow',label='2016')
    ax11.bar([p + width*2 for p in pos],quarter[2017],width,alpha=0.5,color='red',label='2017')
    ax11.bar([p + width*3 for p in pos],quarter[2018],width,alpha=0.5,color='black',label='2018')
    ax11.set_xticks([p + 1.5*width for p in pos])
    ax11.set_xticklabels(year)
```

```
ax11.set_xlim(min(pos)-width, max(pos)+width*8)
ax11.set_ylim([0,1000000000])
ax11.legend(q, loc='upper left')
ax11.set_xlabel("QUARTERS",fontsize=20)
ax11.set_ylabel("DOWNLOADS",fontsize=20)
ax11.set_title("Downloads in each quarter of the year",fontsize=20)
canvas = FigureCanvasTkAgg(figure11, fig11)
canvas.get_tk_widget().place(x=500, y=150)
canvas.draw()
figure11.tight_layout()
```



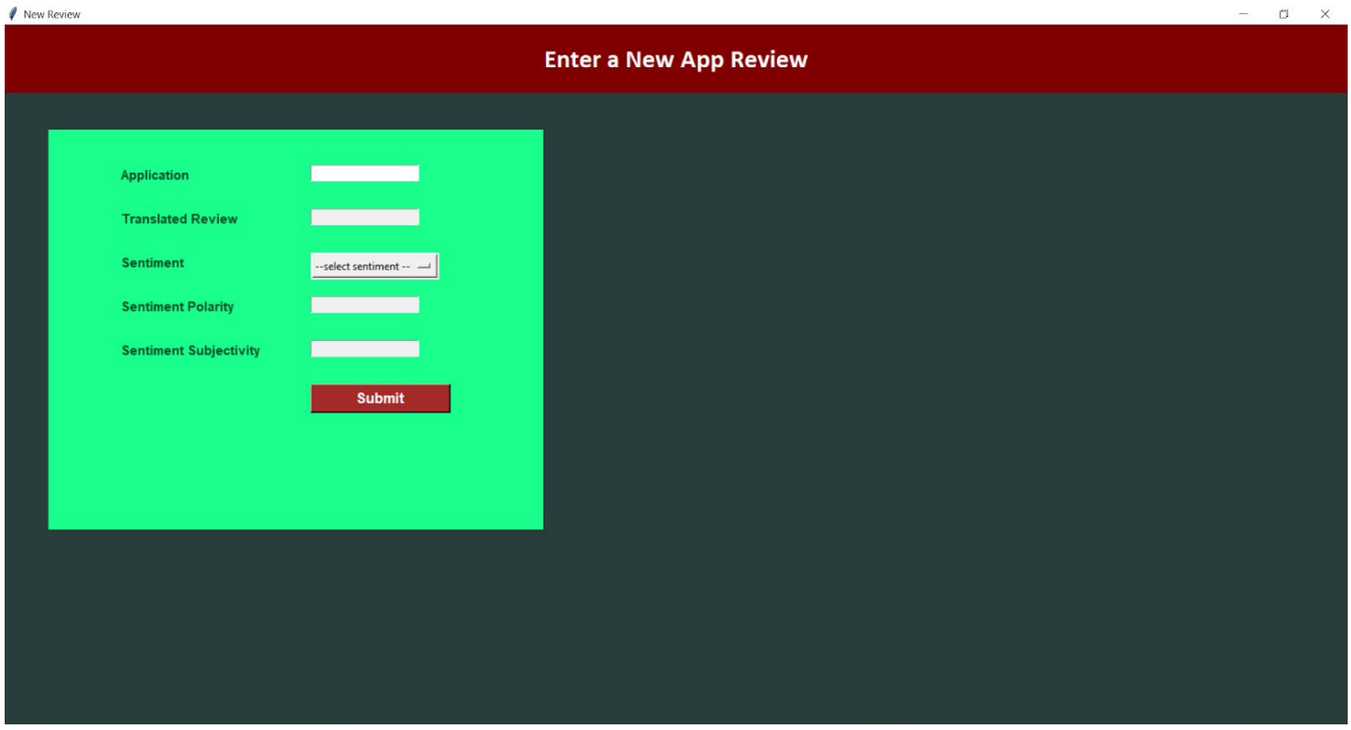
Code for comparison of Sentiment Polarity & Sentiment Subjectivity :

```
def fig13():
    fig13 = Toplevel(home)
    fig13.title("QUESTION 13")
    createWindow(fig13)
    Label(fig13,text="Study and find out the relation between the Sentiment-polarity and
 sentiment-subjectivity of all the apps ?",width="170", height="5", font=Label1_font, fg='white',
 bg='#800000').place(x=0, y=0)
    b1 = Button(fig13, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command=backtohome).place(x=0, y=0)
    sp = list(df2['Sentiment_Polarity'])
    ss = list(df2['Sentiment_Subjectivity'])
    figure13 = py.Figure(figsize = (7,8) , dpi = 75)
    ax13 = figure13.add_subplot(111)
    ax13.set_title('Relation between Sentiment_Polarity and Sentiment_Subjectivity',fontsize=20)
    ax13.set_xlabel('Sentiment_Polarity',fontsize=20)
    ax13.set_ylabel('Sentiment_Subjectivity',fontsize=20)
    s = review_df[review_df['Sentiment_Polarity']==0.4]
    m = round(s['Sentiment_Subjectivity'].mean(),3)
    Label(fig13,text="The Sentiment Subjectivity for a Sentiment Polarity of 0.4 is = \n",bg="#e79700",
 width="60", height="2", font=Button1_font, fg='white').place(x=100, y=400)
    Label(fig13,text=m,bg="#e79700", width="20", height="1", font=Button1_font,
 fg='white').place(x=300, y=500)
#    ax13.scatter(sp,ss)
    sns.regplot(sp,ss,scatter=True,ax=ax13,line_kws={"color":"r","alpha":0.7,"lw":5})
    canvas = FigureCanvasTkAgg(figure13, fig13)
    canvas.get_tk_widget().place(x=900, y=200)
    canvas.draw()
```



Code for providing an interface to add data to App_data sheet :

```
def check_entry(new_entry):
    global df
    Label(new_entry,text="Updated successfully", width="30", height="1", font=Label_font, bg='brown',
 fg='white').place(x=50,y=500)
    d = datetime.strptime(str(lastupdated.get()),'%d/%m/%Y')
    # d = d.strftime('%B %d %Y')
    print(d)
    print(d.year)
    print(d.month)
    appdetails = {'App': app.get(), 'Category': category.get(), 'Rating': ratings.get(), 'Reviews':
 reviews.get(),
                  'Size': size.get(), 'Installs': installs.get(), 'Type': apptype.get(), 'Price': price.get(),
                  'Content Rating': contentrating.get(), 'Genres': genres.get(), 'Last Updated': d,
                  'Current Ver': currentver.get(), 'Android Ver':
 androidver.get(),'Year':d.year,'Month':d.month}
    column = list(appdetails.keys())
    new_data_frame = pd.DataFrame([appdetails],columns = column)
    df = df.append(new_data_frame,ignore_index=True,sort = False)
    df.to_csv("APP_DETAILS.csv",index=False)
```



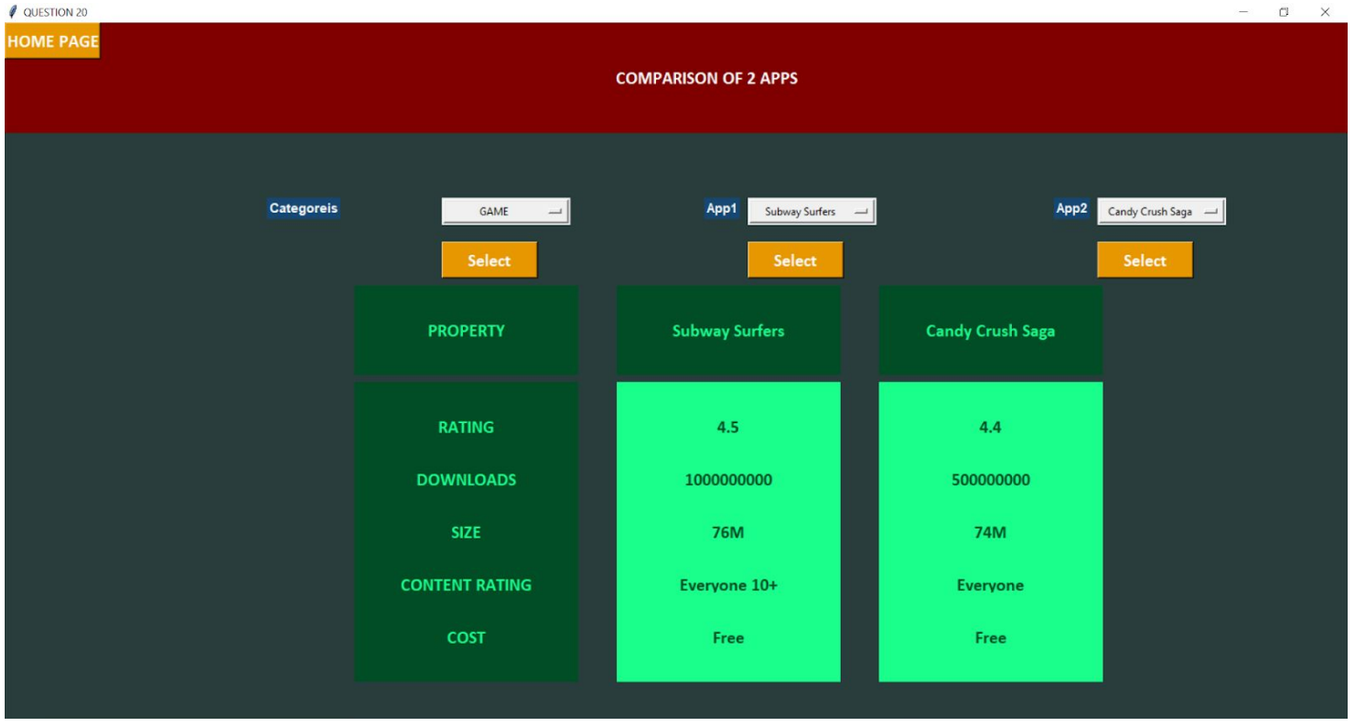
Code for providing an interface to add data to user_review sheet :

```
def check_review(new_review):
    global review_df
    Label(new_review,text="Updated successfully", width="30", height="1", font=Label_font,
 bg='brown', fg='white',anchor=W).place(x=50,y=500)
    review_details =
 {'App':application.get(),'Translated_Review':trans_review.get(),'Sentiment':sentiment.get(),'Sentiment_
 Polarity': senti_polarity.get(),'Sentiment_Subjectivity':senti_subject.get()}
    column = list(review_details.keys())
    new_data_frame = pd.DataFrame([review_details],columns = column)
    review_df = review_df.append(new_data_frame,ignore_index = True , sort = False)
    review_df.to_csv("REVIEW.csv",index=False)

def fig18():
    global fig18
    fig18 = Toplevel(home)
    fig18.title("QUESTION 18")
    createWindow(fig18)
    Label(fig18, text="Provide an interface to add new data to both the datasets provided.",
  width="170", height="5",font=Label1_font, fg='white', bg='#800000').place(x=0, y=0)
    b1 = Button(fig18, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command=backtohome).place(x=0, y=0)
    b2 = Button(fig18, text="New App Record", bg="#e79700", width="25", height="1",
 font=Button1_font, fg='white',command=new_entry).place(x=250, y=220)
    b3 = Button(fig18, text="Review App", bg="#e79700", width="25", height="1", font=Button1_font,
 fg='white',command=new_review).place(x=650, y=220)
```

EXTRA FEATURE

Code for Top Free ,Paid and Trending Apps in the playstore :

```
def fig19():
    fig19 = Toplevel(home)
    fig19.title("QUESTION 19")
    createWindow(fig19)
    Label(fig19, text="Top Apps in the PlayStore ",width="170", height="5", font=Label1_font,
 fg='white', bg='#800000').place(x=0, y=0)
    b1 = Button(fig19, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command=backtohome).place(x=0, y=0)
    Maxfree = Question_19.Maxfree
    Maxpaid = Question_19.Maxpaid
    Trending = Question_19.Trending
    Label(fig19, text="Top Free Apps ",width="45", height="5", font=Label1_font, bg='#004d26',
 fg='#1aff8c').place(x=120, y=200)
    Label(fig19, text="Top Paid Apps ",width="45", height="5", font=Label1_font, bg='#004d26',
 fg='#1aff8c').place(x=520, y=200)
    Label(fig19, text="Trending Apps ",width="45", height="5", font=Label1_font, bg='#004d26',
 fg='#1aff8c').place(x=920, y=200)
    c=10
    for i in range(len(Maxfree)):
      Label(fig19, text="{0}) {1}".format(i+1,Maxfree[i]),width="45", height="3", font=Label1_font,
 fg='#004d26', bg='#1aff8c').place(x=120, y=300+c)
      c+=60
    c=10
    for i in range(len(Maxfree)):
      Label(fig19, text="{0}) {1}".format(i+1,Maxpaid[i]),width="45", height="3", font=Label1_font,
 fg='#004d26', bg='#1aff8c').place(x=520, y=300+c)
      c+=60
    c=10
    for i in range(len(Maxfree)):
      Label(fig19, text="{0}) {1}".format(i+1,Trending[i]),width="45", height="3", font=Label1_font,
 fg='#004d26', bg='#1aff8c').place(x=920, y=300+c)
      c+=60
```



Code for comparison of 2 apps belonging to the same category in the playstore

```
def fig20():
    global fig20
    global cate
    fig20 = Toplevel(home)
    fig20.title("QUESTION 20")
    createWindow(fig20)
    cate=StringVar(fig20)
    Label(fig20, text="COMPARISON OF 2 APPS", width="160", height="5",font=Label1_font,
 fg='white', bg='#800000').place(x=0, y=0)
    b1 = Button(fig20, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command=backtohome).place(x=0, y=0)
    cat=Question_20.cat
    Label(fig20, text="Categoreis", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873',
 anchor=W).place(x=300,y=200)
    droplist = OptionMenu(fig20,cate, *cat)
    droplist.config(width=17)
    cate.set('--select type --')
    droplist.place(x=500, y=200)
    b2 = Button(fig20, text="Select", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command=lambda : callApp1(str(cate.get()),fig20)).place(x=500, y=250)
```

Section 5 :

Testing

Types of Software Testing

Software testing is generally classified into two main broad categories: functional testing and non-functional testing. There is also another general type of testing called maintenance testing.

1. Functional Testing

Functional testing involves the testing of the functional aspects of a software application. When you're performing functional tests, you have to test each and every functionality. You need to see whether you're getting the desired results or not.

There are several types of functional testing, such as:

- Unit testing
- Integration testing
- End-to-end testing
- Regression testing
- Acceptance testing
- White box testing
- Black box testing
- System Testing

Functional tests are performed both manually and using automation tools. For this kind of testing, manual testing is easy, but you should use tools when necessary.

2. Non-functional Testing

Non-functional testing is the testing of non-functional aspects of an application, such as performance, reliability, usability, security, and so on. Non-functional tests are performed after the functional tests.

With non-functional testing, you can improve your software's quality to a great extent. Functional tests also improve the quality, but with non-functional tests, you have the opportunity to make your software even better. Non-functional testing allows you to polish the software. This kind of testing is not about whether the software works or not. Rather, it's about how well the software runs, and many other things.

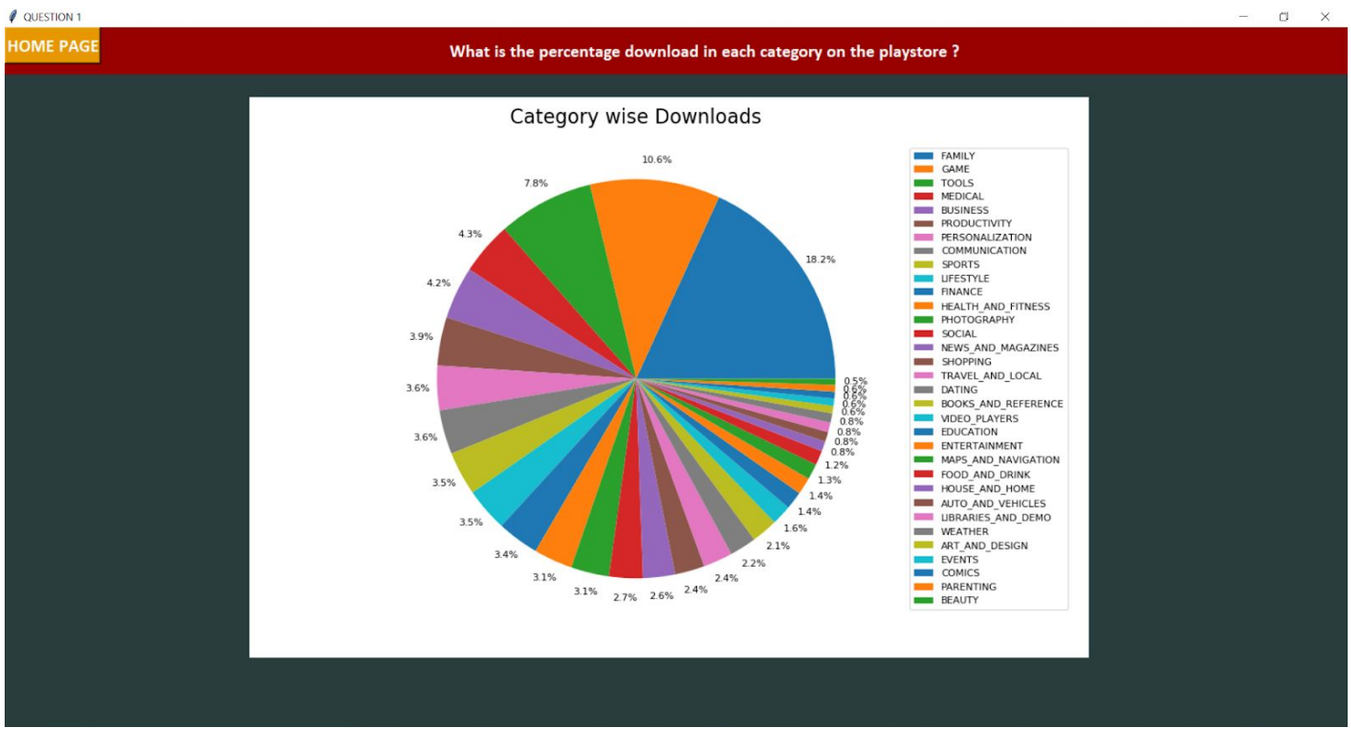
Non-functional tests are not generally run manually. In fact, it's difficult to perform this kind of tests manually. So these tests are usually executed using tools.

There are several types of non-functional testing, such as:

- Performance testing
- Security testing
- Scalability testing
- Volume testing
- Stress testing

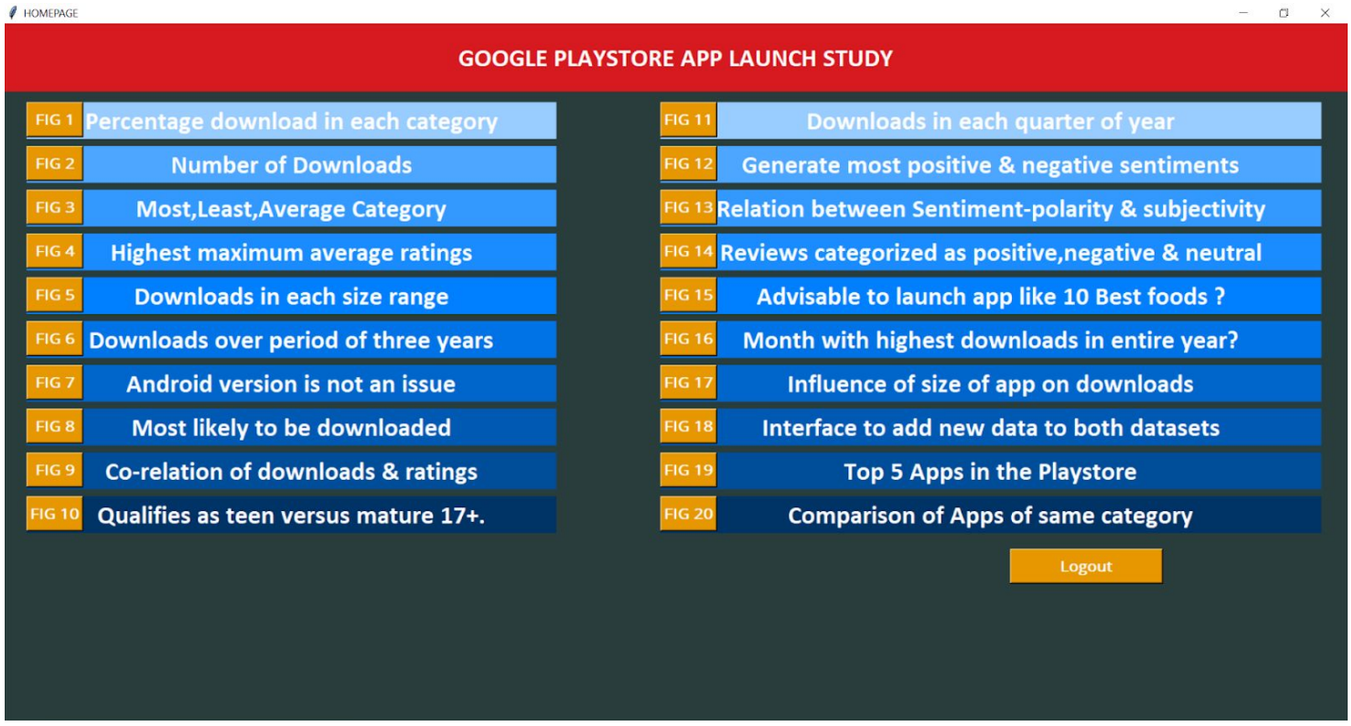
1. Unit Testing

Testing each component or module of your software project is known as unit testing. To perform this kind of testing, knowledge of programming is necessary. So only programmers do this kind of tests, not testers. Unit Testing includes testing of individual features independently. All features specified in the requirements sections have been successfully unit tested. A screenshot of one unit test case is as below .



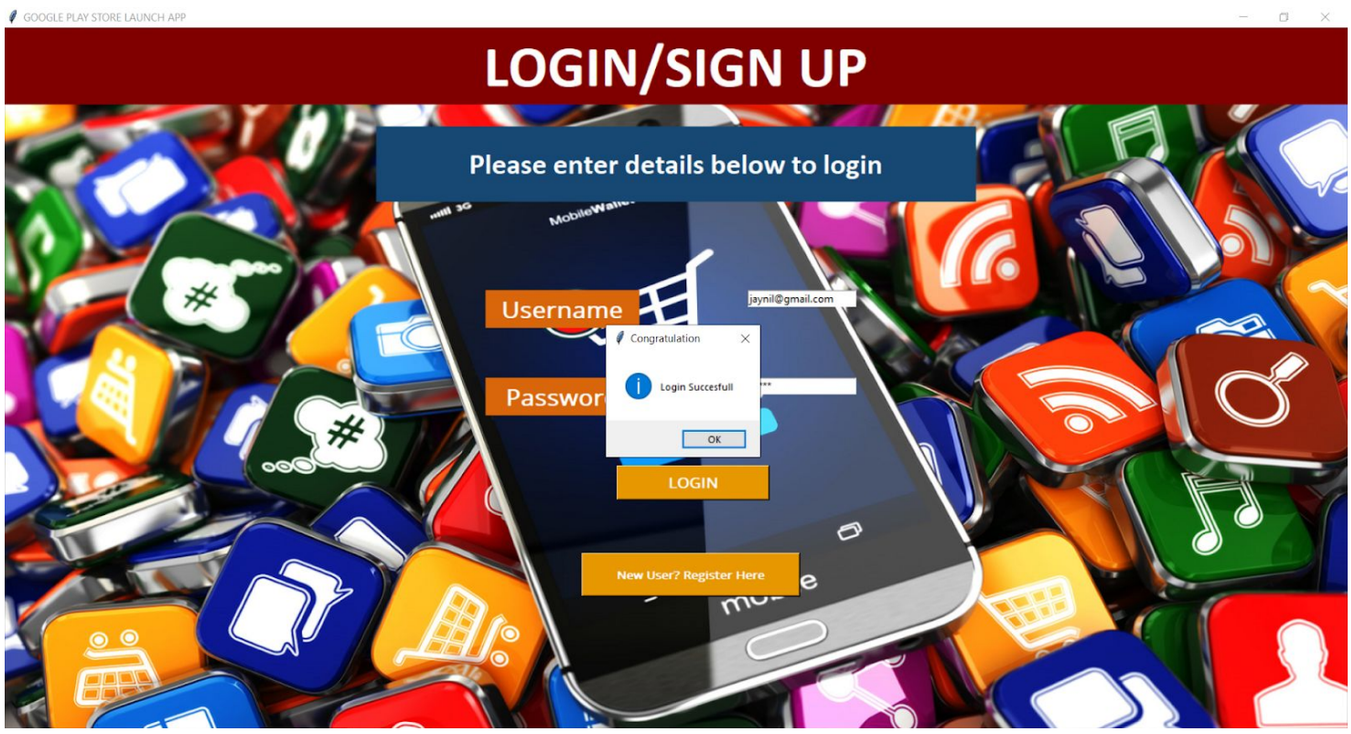
2. Integration testing

After integrating the modules, you need to see if the combined modules work together or not. This type of testing is known as integration testing. You need to perform fewer integration tests than unit tests. All the individual features are combined into one application with a home screen to navigate to each feature independently.



3. System Testing

In this software is tested so that it works fine for different operating system.It is covered under the black box testing technique. In this we just focus on required input and output without focusing on internal working. In this we have security testing, recovery testing , stress testing and performance testing. The whole system is integrated with database and tested against the test cases of login and register. The test cases have been successfully passed by the system.



Validations

1. Login and Register Screen:

In this screen, we have done the validations for

a. email pattern
b. password cross-checking and strength
c. name and mobile
d. not allowing duplicate emails in database
e. candidate name cannot be empty

2. Interface to add data to App_data sheet :

In this screen, we have done the validations for

a. Application name
b. Rating
c. Reviews
d. Type & Content Rating
e. Category & Genre
f. Last Updated

3. Interface to add data to User_review sheet:

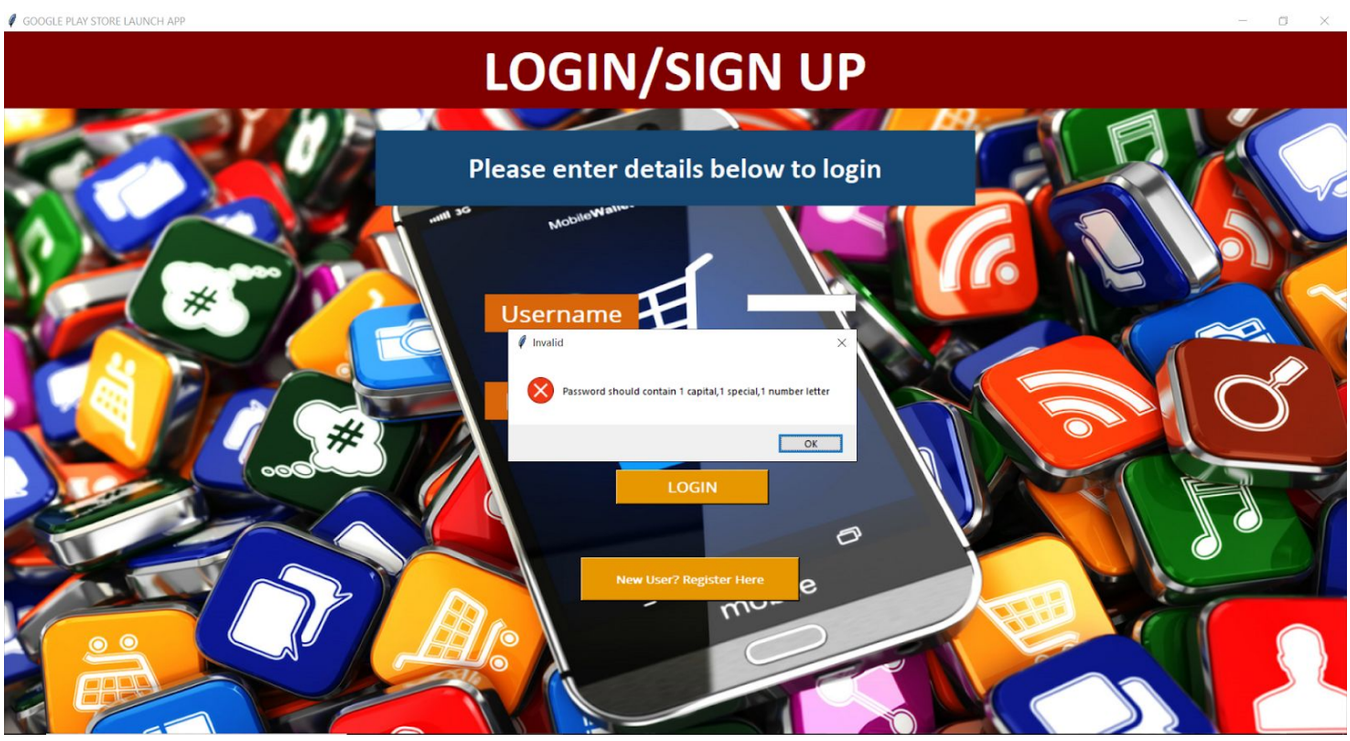
In this screen, we have done the validations for

a. Application Name
b. Sentiment to be chosen among three types
c. Sentiment Polarity & Subjectivity

Also , general non empty validations for each Entry field have been done.

Error messages for every validation is shown to the user with correct format and proper notations for each field.

These validations will be useful to maintain database consistency.



Section 6 :

Final code:

```
import seaborn as sns

import matplotlib

import matplotlib.pyplot as py

import matplotlib.cm as cm

matplotlib.use("TkAgg")

from matplotlib.colors import Normalize

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

from matplotlib.figure import Figure

from tkinter import *

import pandas as pd

import numpy as np

from tkcalendar import Calendar,DateEntry

import pymysql

import tkinter.messagebox as tm

import re

import calendar

import random

from datetime import datetime

import warnings

import Question_19

import Question_20

warnings.filterwarnings('ignore')

# Creating the data frame to read the csv file

df = pd.read_excel('C:\\Users\\Jaynil
 Gaglani\\Machine_Learning_Coding_Track_Module_5\\Datasets_Python_Project\\app_data_cleaned.
 xlsx',header=0)

#df.drop(index = 10472 , inplace = True)

#df.drop(index = 9148,inplace=True)

##Creating the  second data frame to read the review csv file for adding new data

review_df = pd.read_csv('C:\\Users\\Jaynil
 Gaglani\\Machine_Learning_Coding_Track_Module_5\\Datasets_Python_Project\\user_reviews.csv',h
 eader=0)

##Cleaning the data of the Installs column

#df['Installs'] = df['Installs'].str.strip('+')

df['Last Updated'] = df['Last Updated'].astype('datetime64[ns]')

df['Year'] = pd.DatetimeIndex(df['Last Updated']).year
```

```
CatYear = df.groupby(by=['Category', 'Year'])

TrendDict = CatYear.Installs.mean().to_dict()

dataframe = df

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

df2 = pd.read_csv('C:\\Users\\Jaynil
 Gaglani\\Machine_Learning_Coding_Track_Module_5\\Datasets_Python_Project\\user_reviews.csv',h
 eader=0)

cat = list(df['Category'].unique())

Label_font = ("Calibri", 22, "bold")

Label1_font = ("Calibri", 15, "bold")

Button_font = ("Calibri", 13, 'bold')

Button1_font = ("Calibri", 15, 'bold')


def adjustWindow(window):

    #    window.geometry('1366x768')

   w = 1366

   h = 768

   ws = window.winfo_screenwidth()     # width for the window size of current device

   hs = window.winfo_screenheight()     # height for the window size of current device

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

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

   window.geometry('%dx%d+%d+%d' %(w, h, x, y))

   window.resizable(True, True)

   window.configure(background="#adad85")

#

def createWindow(window):

#    window.geometry('1366x768')

   w = 1366

   h = 768

   ws = home.winfo_screenwidth()     # width for the window size of current device

   hs = home.winfo_screenheight()     # height for the window size of current device

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

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

   window.geometry('%dx%d+%d+%d' %(w, h, x, y))

   window.resizable(True, True)

   window.configure(background="#293d3d")


def saveStudent():

   try:
```

```
        conn = pymysql.connect(user="root", password="", host="localhost",database="playstore")
        cur = conn.cursor()
        query = cur.execute("""insert into
    student_details(fullname,email,mobile,password,gender)values(%s,%s,%s,%s,%s)"""
    ,(cand_name.get(),email.get(),mobile.get(),password.get(),genderValue.get()))
        # print(query)
        conn.commit()
        conn.close()
        # Show message for successing
        tm.showinfo("Save" , "Success!")
        # Initializing for each input.
        candNameEntry.config(state='disabled')
        emailEntry.config(state='disabled')
        mobileEntry.config(state='disabled')
      except Exception as e:
        print(e)
      # If error on saving , shows error message.
        tm.showerror("Save" ,"Failed to save!")


def cancel():
    cand_name.set("")
    email.set("")
    mobile.set("")
    password.set("")
    repassword.set("")
    root.quit()


def backtohome():
    backtohome = Toplevel(home)
    backtohome.title("HOMEPAGE")
    createWindow(backtohome)
    l = Label(backtohome, text="GOOGLE PLAYSTORE APP LAUNCH STUDY", width="500",
    height="2", font=Label_font, fg='white',bg='#d61a1f').pack()
    b = Button(home,text="Logout",width="15",height="1",font=("Open Sans", 13,
    'bold'),fg="white",bg="#e79700",command=destroy).place(x=1150,y=600)
    l1 = Label(backtohome, text="Percentage download in each category", width="40", height="1",
    font=Label_font, fg='white',bg='#99ccff').place(x=25, y=90)
    b1 = Button(backtohome, text="FIG 1", bg="#e79700", width="5", height="1", font=("Open Sans",
    13, 'bold'), fg='white',command=fig1).place(x=25, y=90)
    l2 = Label(backtohome, text="Number of Downloads", width="40", height="1", font=Label_font,
    fg='white',bg='#4da6ff').place(x=25, y=140)
```

```
  b2 = Button(backtohome, text="FIG 2", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig2).place(x=25, y=140)

  l3 = Label(backtohome, text="Most,Least,Average Category", width="40", height="1", font=Label_font, fg='white',bg='#3399ff').place(x=25, y=190)

  b3 = Button(backtohome, text="FIG 3", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig3).place(x=25, y=190)

  l4 = Label(backtohome, text="Highest maximum average ratings", width="40", height="1", font=Label_font, fg='white',bg='#1a8cff').place(x=25, y=240)

  b4 = Button(backtohome, text="FIG 4", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig4).place(x=25, y=240)

  l5 = Label(backtohome, text="Download category wise over period", width="40", height="1", font=Label_font, fg='white',bg='#0080ff').place(x=25, y=290)

  b5 = Button(backtohome, text="FIG 5", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig5).place(x=25, y=290)

  l6 = Label(backtohome, text="Downloads over period of three years", width="40", height="1", font=Label_font, fg='white',bg='#0073e6').place(x=25, y=340)

  b6 = Button(backtohome, text="FIG 6", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig6).place(x=25, y=340)

  l7 = Label(backtohome, text="Android version is not an issue", width="40", height="1", font=Label_font, fg='white',bg='#0066cc').place(x=25, y=390)

  b7 = Button(backtohome, text="FIG 7", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig7).place(x=25, y=390)

  l8 = Label(backtohome, text="Most likely to be downloaded", width="40", height="1", font=Label_font, fg='white',bg='#0059b3').place(x=25, y=440)

  b8 = Button(backtohome, text="FIG 8", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig8).place(x=25, y=440)

  l9 = Label(backtohome, text="Co-relation of downloads & ratings", width="40", height="1", font=Label_font, fg='white',bg='#004d99').place(x=25, y=490)

  b9 = Button(backtohome, text="FIG 9", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig9).place(x=25, y=490)

  l10 = Label(backtohome, text="Qualifies as teen versus mature 17+.", width="40", height="1", font=Label_font, fg='white',bg='#003366').place(x=25, y=540)

  b10 = Button(backtohome, text="FIG 10", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig10).place(x=25, y=540)

  l11 = Label(backtohome, text="Downloads in each quarter of year", width="50", height="1",font=Label_font, fg='white', bg='#99ccff').place(x=750, y=90)

  b11 = Button(backtohome, text="FIG 11", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig11).place(x=750, y=90)

  l12 = Label(backtohome, text="Generate most positive & negative sentiments", width="50", height="1", font=Label_font,fg='white', bg='#4da6ff').place(x=750, y=140)

  b12 = Button(backtohome, text="FIG 12", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig12).place(x=750, y=140)

  l13 = Label(backtohome, text="Relation between Sentiment-polarity & subjectivity", width="50", height="1",font=Label_font, fg='white', bg='#3399ff').place(x=750, y=190)

  b13 = Button(backtohome, text="FIG 13", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig13).place(x=750, y=190)

  l14 = Label(backtohome, text="Reviews categorized as positive,negative & neutral", width="50", height="1",font=Label_font, fg='white', bg='#1a8cff').place(x=750, y=240)
```

```
    b14 = Button(backtohome, text="FIG 14", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig14).place(x=750, y=240)

    l15 = Label(backtohome, text="Advisable to launch app like 10 Best foods ?", width="50", height="1",font=Label_font, fg='white', bg='#0080ff').place(x=750, y=290)

    b15 = Button(backtohome, text="FIG 15", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig15).place(x=750, y=290)

    l16 = Label(backtohome, text="Month with highest downloads in entire year?", width="50", height="1",font=Label_font, fg='white', bg='#0073e6').place(x=750, y=340)

    b16 = Button(backtohome, text="FIG 16", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig16).place(x=750, y=340)

    l17 = Label(backtohome, text="Influence of size of app on downloads", width="50", height="1", font=Label_font,fg='white', bg='#0066cc').place(x=750, y=390)

    b17 = Button(backtohome, text="FIG 17", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig17).place(x=750, y=390)

    l18 = Label(backtohome, text="Interface to add new data to both datasets", width="50", height="1", font=Label_font,fg='white', bg='#0059b3').place(x=750, y=440)

    b18 = Button(backtohome, text="FIG 18", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig18).place(x=750, y=440)

    l19 = Label(backtohome, text="Top 5 Apps in the Playstore", width="50", height="1", font=Label_font,fg='white', bg='#004d99').place(x=750, y=490)

    b19 = Button(backtohome, text="FIG 19", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig19).place(x=750, y=490)

    l20 = Label(backtohome, text="Comparison of Apps of same category", width="50", height="1", font=Label_font,fg='white', bg='#003366').place(x=750, y=540)

    b20 = Button(backtohome, text="FIG 20", bg="#e79700", width="5", height="1", font=("Open Sans", 13, 'bold'), fg='white',command=fig20).place(x=750, y=540)


def fig1():

# no of apps download in each category distribution

    fig1 = Toplevel(home)

    fig1.title("QUESTION 1")

    createWindow(fig1)

    Label(fig1, text="What is the percentage download in each category on the playstore ? ", width="160", height="2",font=Label1_font, fg='white', bg='#990000').place(x=0, y=0)

    b1 = Button(fig1, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font, fg='white',command=backtohome).place(x=0, y=0)

    f = Figure(figsize=(12, 8), dpi=80)

    a = f.add_subplot(111)

    a.pie(df['Category'].value_counts().values, autopct='%1.1f%%',pctdistance = 1.1 , labeldistance = 1.2)

    a.set_title("Category wise Downloads",fontsize=20)

    a.legend(df['Category'].value_counts().index, loc='center left', bbox_to_anchor=(1.04, 0.5), ncol=1)

    canvas = FigureCanvasTkAgg(f, fig1)

    canvas.get_tk_widget().place(x=280, y=80)

    canvas.draw()
```

```
    f.tight_layout()


def fig2():

    # pd.cut()

    fig2 = Toplevel(home)

    fig2.title("QUESTION 2")

    createWindow(fig2)

    Label(fig2,text="How many apps have managed to get the following number of downloads \n a) Between 10,000 and 50,000 \t b) Between 50,000 and 150000 \n c) Between 150000 and 500000 \t d) Between 500000 and 5000000 \n e) More than 5000000",width="160", height="4", font=Label1_font, fg='white', bg='#800000').place(x=0, y=0)

    b1 = Button(fig2, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font, fg='white',command=backtohome).place(x=0, y=0)

    cut_bins = df.Installs.unique()

    req_cut_bins = [10000, 50000, 150000, 500000, 5000000, 50000000, 10000000000]

    InstallCount = pd.cut(df['Installs'], req_cut_bins,labels=['10k-50k', '50k-150k', '150k-500k', '500k-5000k', '5000k-50000k', '50000k +'],include_lowest=True,right = False)

    figure1 = py.Figure(figsize=(7,8), dpi=80)

    ax1 = figure1.add_subplot(111)

    ax1.set_xlabel("Download Ranges",fontsize=20)

    ax1.set_ylabel("Number of Downloads",fontsize=20)

    bar1 = FigureCanvasTkAgg(figure1, fig2)

    bar1.get_tk_widget().place(x =500,y=130)

    InstallCount.value_counts(sort = False).plot(kind='bar', legend=False, ax=ax1,color=['red','green','black','orange','blue','yellow'])

    ax1.set_title('No of Apps Vs. No of Downloads',fontsize=20)

    figure1.tight_layout()


def fig3():

    fig3 = Toplevel(home)

    fig3.title("QUESTION 3")

    createWindow(fig3)

    Label(fig3,text="Which category of apps have managed to get most, least and \n an average of 2,50,000 downloads atleast ?",width="160", height="3", font=Label1_font, fg='white', bg='#800000').place(x=0, y=0)

    b1 = Button(fig3, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font, fg='white',command=backtohome).place(x=0, y=0)

    x=df[['Category','Installs']]

    cat = list(x['Category'].unique())

    d1={}

    d2={}

    d3={}
```

```
    for i in cat:
        d1[i]=0
        d2[i]=0
    z=x.to_dict(orient='split')
    for i in cat:
        count=0
        for j in range(len(z['data'])):
            if i==z['data'][j][0]:
                d2[i]+=1
                count=count+z['data'][j][1]
                d1[i]=count
    for i in d1.keys():
        d3[i]=(d1[i]/d2[i])
        if(d3[i]>=250000 and d3[i]<=520000):
            avg=i
    Max = max(d1.items(), key=lambda x : x[1])
    Min = min(d1.items(), key=lambda x : x[1])
    Label(fig3,text="Category of App with Maximum downloads : ",width="50", height="3",
 font=Label1_font, fg='white', bg='#ff9900').place(x=530,y=100)
    Label(fig3,text=Max[0],width="20", height="1", font=Label1_font, fg='white',
 bg='#003366').place(x=530,y=200)
    Label(fig3,text=Max[1],width="20", height="1", font=Label1_font, fg='white',
 bg='#003366').place(x=830,y=200)
    Label(fig3,text="Category of App with Minimum downloads : ",width="50", height="3",
 font=Label1_font, fg='white', bg='#ff9900').place(x=530,y=300)
    Label(fig3,text=Min[0],width="20", height="1", font=Label1_font, fg='white',
 bg='#003366').place(x=530,y=400)
    Label(fig3,text=Min[1],width="20", height="1", font=Label1_font, fg='white',
 bg='#003366').place(x=830,y=400)
    Label(fig3,text="Category of App with Average downloads of 2,50,000",width="50", height="3",
 font=Label1_font, fg='white', bg='#ff9900').place(x=530,y=500)
    Label(fig3,text=avg,width="20", height="1", font=Label1_font, fg='white',
 bg='#003366').place(x=530,y=600)
#
def fig4():
    fig4 = Toplevel(home)
    fig4.title("QUESTION 4")
    createWindow(fig4)
    Label(fig4, text="Which category of apps have managed to get the highest maximum average
 ratings from the users ?",width="170", height="3", font=Label1_font, fg='white',
 bg='#800000').place(x=0, y=0)
    b1 = Button(fig4, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command=backtohome).place(x=0, y=0)
```

```
    x=dataframe[['Category','Rating']]
    cat = list(x['Category'].unique())
    d1={}
    d2={}
    d3={}
    d4 = []
    for i in cat:
        d1[i]=0
        d2[i]=0
    z=x.to_dict(orient='split')
    for i in cat:
        count=0
        for j in range(len(z['data'])):
            if i==z['data'][j][0]:
                d2[i]+=1
                count=count+z['data'][j][1]
                d1[i]=count
    for i in d1.keys():
        d3[i]=round(d1[i]/d2[i],3)
        d4.append(d3[i])
    Max = max(d3.items(), key=lambda x : x[1])
    col1 = list(d3.keys())
    col2 = list(d3.values())
    dk = pd.DataFrame(list(zip(col1,col2)))
    dk[1] = dk[1].astype(float)
#    conn = pymysql.connect(host="localhost",user="root",passwd="",database="playstore")
#    cur = conn.cursor()
#    for i in range(len(dk)):
#        insert_query = """INSERT INTO app_rating (Category,Average_Rating) VALUES (%s,%s)"""
#        params = (dk[0][i],float(dk[1][i]))
#        cur.execute(insert_query,params)
#    conn.commit()
#    conn.close()
    f = Figure(figsize=(15, 10), dpi=65)
    ax = f.add_subplot(111)
    ax.plot(cat,d4, 'o', color='red')
    ax.set_xlabel("Categories",fontsize=20)
    ax.set_ylabel("Ratings",fontsize=20)
```

```
    ax.set_title("Category wise Average Ratings",fontsize=20)
    ax.set_xticklabels(cat, rotation=90, ha='center')
    canvas = FigureCanvasTkAgg(f, fig4)
    canvas.get_tk_widget().place(x=280, y=100)
    canvas.draw()
    f.tight_layout()
#
def fig5():
    global fig5y,fig5z
    fig5 = Toplevel(home)
    fig5.title("QUESTION 5")
    createWindow(fig5)
    Label(fig5, text="What is the number of installs for the following app sizes.\n a) Size between 10 and
 20 mb \n b) Size between 20 and 30 mb \n c) More than 30 mb",width="170", height="4",
 font=Label1_font, fg='white', bg='#800000').place(x=0, y=0)
    b1 = Button(fig5, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command=backtohome).place(x=0, y=0)
    x= list(dataframe['Size'])
    d = list(dataframe['Installs'])
    count=[0,0,0,0,0,0,0,0]
    for i in range (0,len(x)-1):
        if x[i]!='Varies with device':
            if(x[i]>='10M' and x[i]<'20M'):
                count[0]+=d[i]
            elif(x[i]>='20M' and x[i]<'30M'):
                count[1]+=d[i]
            elif(x[i]>='30M' and x[i]<'40M'):
                count[2]+=d[i]
            elif(x[i]>='40M' and x[i]<'50M'):
                count[3]+=d[i]
            elif(x[i]>='50M' and x[i]<'60M'):
                count[4]+=d[i]
            elif(x[i]>='60M' and x[i]<'70M'):
                count[5]+=d[i]
            elif(x[i]>='70M' and x[i]<'80M'):
                count[6]+=d[i]
            elif(x[i]>='80M'):
                count[7]+=d[i]
    fig5y=count
    y = list([count[0],count[1],count[2]])
```

```
    # print(dataframe[dataframe.Rating>=4.1])
    z = ['10-20','20-30','30+']
    fig5z=['10-20','20-30','30-40','40-50','50-60','60-70','70-80','80+']
    f = Figure(figsize=(10, 9), dpi=70)
    ax5 = f.add_subplot(111)
    ax5.bar(z,y,color=['orange','b','g'],width = 0.5)
    ax5.set_xlabel("Size Ranges",fontsize=20)
    ax5.set_ylabel("Number of Installs",fontsize=20)
    ax5.set_title("Installs in each Size Range",fontsize=20)
    ax5.set_xticklabels(z, ha='center')
    canvas = FigureCanvasTkAgg(f, fig5)
    canvas.get_tk_widget().place(x=400, y=150)
    canvas.draw()
    f.tight_layout()


def fig6():
    fig6 = Toplevel(home)
    fig6.title("QUESTION 6")
    createWindow(fig6)
    Label(fig6,text="For the years 2016,2017,2018 what are the category of apps that have got the most
 and the least downloads ?",width="170", height="5", font=Label1_font, fg='white',
 bg='#800000').place(x=0, y=0)
    b1 = Button(fig6, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command=backtohome).place(x=0, y=0)
    cat = dataframe['Category'].unique()
    cat=cat.tolist()
    x = dataframe[['Category','Installs','Year']]
    year = [2016,2017,2018]
    downloads = {}
    df = x.to_dict(orient='split')
    for i in cat:
      count = 0
      for j in range(len(df['data'])):
        if (i==df['data'][j][0] and df['data'][j][2] in year):
          count = count + df['data'][j][1]
          downloads[i] = count
    Max = max(downloads.items(), key=lambda x : x[1])
    Min = min(downloads.items(), key=lambda x : x[1])
    Label(fig6,text="Category of App with Maximum downloads : ",width="50", height="4",
 font=Label1_font, fg='white', bg='#ff9900').place(x=550,y=200)
```

```
    Label(fig6,text=Max[0],width="20", height="2", font=Label1_font, fg='white',
bg='#003366').place(x=550,y=350)

    Label(fig6,text=Max[1],width="20", height="2", font=Label1_font, fg='white',
bg='#003366').place(x=850,y=350)

    Label(fig6,text="Category of App with Least downloads : ",width="50", height="4",
font=Label1_font, fg='white', bg='#ff9900').place(x=550,y=500)

    Label(fig6,text=Min[0],width="20", height="2", font=Label1_font, fg='white',
bg='#003366').place(x=550,y=650)

    Label(fig6,text=Min[1],width="20", height="2", font=Label1_font, fg='white',
bg='#003366').place(x=850,y=650)


def fig7():

    fig7 = Toplevel()

    fig7.title("QUESTION 7")

    createWindow(fig7)

    Label(fig7,text="All those apps,whose android version is not an issue and can work with varying
devices.\n What is the percentage increase or decrease in the downloads ?",width="170",
height="5", font=Label1_font, fg='white', bg='#800000').place(x=0, y=0)

    b1 = Button(fig7, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
fg='white',command=backtohome).place(x=0, y=0)

    android_df = df.copy()

    android_df = android_df[android_df['Android Ver'] == 'Varies with device']

    android_df.sort_values('Year',inplace = True)

    year = ['2013','2014','2015','2016','2017','2018']

    k = []

    d=[]

    for i in range(2018 - 2012 + 1):

        k.append(android_df[android_df.Year == (2012 + i)]['Installs'].sum())

    for i in range(2018 - 2012):

        m=((k[i + 1] - k[i]) / (k[i] ))* 100

        d.append(m)

    # print(d)

    Label(fig7,text="2013 to 2014 Download percent change : 403.1137332022288\n2014 to 2015
Download percent change : 81.86387622149837\n2015 to 2016 Download percent change :
1806.729112102136\n2016 to 2017 Download percent change : 41.47751974465249\n2017 to 2018
Download percent change : 12104.35411777218",width="65", height="8", font=Label1_font,
fg='white', bg='#174873').place(x=800, y=350)

    f = Figure(figsize=(12, 9), dpi=60)

    ax7 = f.add_subplot(111)

    ax7.plot(year,d,linestyle='--',color='r')

    ax7.set_title("Year wise App Andoid Version",fontsize=20)

    ax7.set_xticklabels(year,ha='center')

    ax7.set_xlabel('Years',fontsize=20)
```

```
    ax7.set_ylabel('Varied Downloads',fontsize=20)
    canvas = FigureCanvasTkAgg(f, fig7)
    canvas.get_tk_widget().place(x=50, y=150)
    canvas.draw()
    f.tight_layout()
def getTrendDict(TrendDict,Category):
    years = []
    install = []
    for category , installs in TrendDict.items():
        if list(category)[0] == Category:
            years.append(list(category)[1])
            install.append(installs)
    return years,install


def new_plot(TrendDict,Category,fig):
    years,install = getTrendDict(TrendDict,Category)
    conn = pymysql.connect(host="localhost",user="root",passwd="",database="playstore")
    cur = conn.cursor()
    for i in range(len(years)):
      query = """INSERT INTO app_category (Category,Year,Downloads) VALUES (%s,%s,%s) """
      params = (Category,years[i],install[i])
      cur.execute(query,params)
    conn.commit()
    conn.close()
    figure = py.Figure(figsize = (8,7) , dpi = 75)
    ax = figure.add_subplot(111)
    ax.set_xticks(ticks = years)
    ax.set_title("Most Likely Downloads",fontsize=20)
    ax.set_xlabel("Years",fontsize=20)
    ax.set_ylabel("Downloads",fontsize=20)
    ax.set_xticklabels(labels = years)
    colors = ['b','g','r','c','m','y','k','w']
    r = random.randint(0,7)
    ax.plot(years,install,color=colors[r],linestyle='-.')
    canvas = FigureCanvasTkAgg(figure, fig)
    canvas.get_tk_widget().place(x=500, y=150)
    canvas.draw()
    figure.tight_layout()
```

```
def fig8():

    fig8 = Toplevel(home)

    fig8.title("QUESTION 8")

    createWindow(fig8)

    Label(fig8,text="Amongst sports, entertainment,social media,news,events,travel and games,\n
  Which is the category of app that is most likely to be downloaded in the coming years,\n kindly make
  a prediction and back it with suitable findings ?",width="160", height="5", font=Label1_font,
  fg='white', bg='#800000').place(x=0, y=0)

    b1 = Button(fig8, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
  fg='white',command=backtohome).place(x=0, y=0)

    # Now starting with the drop down list

    OPTIONS =
['SPORTS','ENTERTAINMENT','SOCIAL','NEWS_AND_MAGAZINES','EVENTS','TRAVEL_AND_LOCA
L','GAME']

    variable = StringVar(fig8)

    variable.set('CATEGORY')

    w = OptionMenu(fig8, variable, *OPTIONS)

    w.place(x=100, y=220)

    w.configure(bg="#e79700", fg='white', height="1", font=Button1_font)

    b = Button(fig8, text='SHOW', fg='red', height="1", font=Button1_font, command=lambda:
  new_plot(TrendDict, variable.get(),fig8))

    b.place(x=350, y=220)


def fig9():

    fig9 = Toplevel(home)

    fig9.title("QUESTION 9")

    createWindow(fig9)

    Label(fig9,text="All those apps who have managed to get over 1,00,000 downloads,\n have they
  managed to get an average rating of 4.1 and above? \n An we conclude something in co-relation to
  the number of downloads and the ratings received ?",width="160", height="5", font=Label1_font,
  fg='white', bg='#800000').place(x=0, y=0)

    b1 = Button(fig9, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
  fg='white',command=backtohome).place(x=0, y=0)

    x=dataframe[['Category','Rating','Installs']]

    z=x.to_dict(orient='split')

    cat =dataframe['Category'].unique()

    cat=cat.tolist()

    d1=[]

    for i in cat:

        count=0

        for j in range(len(z['data'])):

            if(i==z['data'][j][0]):
```

```
            if(z['data'][j][1]>=4.1 and z['data'][j][2]>=100000):
                count+=1
        d1.append(count)
    a = x[['Rating','Installs']]
    rat = a[(a['Rating']>=4.1) & (a['Installs']>=100000)]
    corr = rat['Installs'].corr(rat['Rating'])
#    print(corr)
#    print(np.corrcoef(rat['Installs'],rat['Rating']))
    figure9 = py.Figure(figsize = (19,9) , dpi = 75)
    ax9 = figure9.add_subplot(1,2,1)
    ax9.set_title('APP WITH 4.1+ RATING AND 1,00,000+ DOWNLOADS',fontsize=20)
    ax9.plot(cat,d1,color='orange')
    ax9.set_xticklabels(cat,rotation=90,ha='center')
    ax9.set_ylabel('No. of Apps',fontsize=20)
    ax9.set_xlabel('CATEGORIES',fontsize=20)
    ax9b = figure9.add_subplot(1,2,2)
    ax9b.scatter(rat['Rating'],rat['Installs'],marker="8",c='r')
    ax9b.set_ylabel("Downloads",fontsize=20)
    ax9b.set_xlabel("Rating",fontsize=20)
    ax9b.set_title("Correlation between Downloads and Rating is (-0.00838)",fontsize=20)
    canvas = FigureCanvasTkAgg(figure9, fig9)
    canvas.get_tk_widget().place(x=50, y=130)
    canvas.draw()
    figure9.tight_layout()


def fig10():
    fig10 = Toplevel(home)
    fig10.title("QUESTION 10")
    createWindow(fig10)
    Label(fig10,text="Across all the years,which month has seen the maximum downloads for each of
 the category.\n What is the ratio of downloads for the app that qualifies as teen versus mature 17+
 ?",width="170", height="5", font=Label1_font, fg='white', bg='#800000').grid(row=0,column=0)
    cat = dataframe['Category'].unique()
    b1 = Button(fig10, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command=backtohome).place(x=0, y=0)
    cat=cat.tolist()
    x = dataframe[['Category','Installs','Month']]
    df = x.to_dict(orient='split')
    downloads = {}
    for i in cat:
```

```
    downloads[i] = [0]*13
  for i in cat:
    for j in range(len(df['data'])):
      if (i==df['data'][j][0]):
        month = df['data'][j][2]
        downloads[i][month] += df['data'][j][1]
  max_month = [0]*13
  max_category = [0]*13
  for j in range(1,13):
    ins = 0
    for i in cat:
      c = ''
      if(downloads[i][j] > ins):
        ins = downloads[i][j]
        c = i
        max_month[j] = ins
        max_category[j] = c
  max_month.pop(0)
  max_category.pop(0)
  l4=['Month','Category','Downloads']


  c=0
  for i in range(len(l4)):
    Label(fig10, text=l4[i],width="20", height="2", font=Label1_font, fg='white',
bg='#174873').place(x=800+c, y=150)
    c+=200
  c=10
  for i in range(len(max_month)):
    Label(fig10, text=calendar.month_name[i+1],width="20", height="1", font=Label1_font, fg='white',
bg='#174873').place(x=800, y=200+c)
    c+=40
  c=10
  for i in range(len(max_month)):
    Label(fig10, text=max_category[i],width="20", height="1", font=Label1_font, fg='white',
bg='#174873').place(x=1000, y=200+c)
    c+=40
  c=10
  for i in range(len(max_month)):
    Label(fig10, text=max_month[i],width="20", height="1", font=Label1_font, fg='white',
bg='#174873').place(x=1200, y=200+c)
```

```
    c+=40
  df = dataframe[['Content Rating','Installs']]
  teen  = []
  mature = []
  for i in range(len(df)):
    if(df['Content Rating'].iloc[i]=="Teen"):
      teen.append(df['Installs'].iloc[i])
    elif(df['Content Rating'].iloc[i]=="Mature 17"):
      mature.append(df['Installs'].iloc[i])
  teen_t = sum(teen)
  mature_t = sum(mature)
  total = teen_t  + mature_t
  teen_t = round((teen_t/total)*100,3)
  mature_t = round((mature_t/total)*100,3)
  percentage = [teen_t,mature_t]
  explode = (0.01,0.03)
  figure10 = py.Figure(figsize = (7,8) , dpi = 75)
  ax9 = figure10.add_subplot(111)
  ax9.pie(percentage,labels=('Teen','Mature 17+'),colors =
('Blue','Yellow'),autopct="%1.1f%%",startangle=90, pctdistance=0.85, explode = explode)
  ax9.axis('equal')
  ax9.set_title("TEEN VS MATURE 17",fontsize=20)
  canvas = FigureCanvasTkAgg(figure10, fig10)
  canvas.get_tk_widget().place(x=150, y=130)
  canvas.draw()
  figure10.tight_layout()


def fig11():
  fig11 = Toplevel(home)
  fig11.title("QUESTION 11")
  createWindow(fig11)
  Label(fig11,text="Which quarter of which year has generated the highest number of install for each
app used in the study?",width="170", height="5", font=Label1_font, fg='white',
bg='#800000').place(x=0, y=0)
  b1 = Button(fig11, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
fg='white',command=backtohome).place(x=0, y=0)
  cat =dataframe['Category'].unique()
  cat=cat.tolist()
  x = dataframe[['App','Category','Installs','Year','Month']]
  df = x.to_dict(orient='split')
```

```
q1=[1,2,3]
q2=[4,5,6]
q3=[7,8,9]
q4=[10,11,12]
year = [2015,2016,2017,2018]
quarter = {}
for i in year:
  quarter[i] = [0]*4
for i in range(len(x)):
  for j in year:
    if(df['data'][i][3] == j):
      if(df['data'][i][4] in q1):
        quarter[j][0] += df['data'][i][2]
      if(df['data'][i][4] in q2):
        quarter[j][1] += df['data'][i][2]
      if(df['data'][i][4] in q3):
        quarter[j][2] += df['data'][i][2]
      if(df['data'][i][4] in q4):
        quarter[j][3] += df['data'][i][2]
q = ['Q1','Q2','Q3','Q4']
figure11 = py.Figure(figsize = (7,8) , dpi = 80)
ax11 = figure11.add_subplot(111)
width = 0.25
pos = list(range(len(quarter)))
ax11.bar(pos,quarter[2015],width,alpha=0.5,color='blue',label='2015')
ax11.bar([p + width for p in pos],quarter[2016],width,alpha=0.5,color='yellow',label='2016')
ax11.bar([p + width*2 for p in pos],quarter[2017],width,alpha=0.5,color='red',label='2017')
ax11.bar([p + width*3 for p in pos],quarter[2018],width,alpha=0.5,color='black',label='2018')
ax11.set_xticks([p + 1.5*width for p in pos])
ax11.set_xticklabels(year)
ax11.set_xlim(min(pos)-width, max(pos)+width*8)
ax11.set_ylim([0,1000000000])
ax11.legend(q, loc='upper left')
ax11.set_xlabel("QUARTERS",fontsize=20)
ax11.set_ylabel("DOWNLOADS",fontsize=20)
ax11.set_title("Downloads in each quarter of the year",fontsize=20)
canvas = FigureCanvasTkAgg(figure11, fig11)
canvas.get_tk_widget().place(x=500, y=150)
```

```
    canvas.draw()
    figure11.tight_layout()


def fig12():
    fig12 = Toplevel(home)
    fig12.title("QUESTION 12")
    createWindow(fig12)
    Label(fig12,text="Which of all the apps given have managed to generate the most positive and
 negative sentiments. \n Also figure out the app which has generated approximately the same ratio for
 positive and negative sentiments ?",width="160", height="5", font=Label1_font, fg='white',
 bg='#800000').place(x=0, y=0)
    b1 = Button(fig12, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command=backtohome).place(x=0, y=0)
    x=df2[['App','Sentiment']]
    z=x.to_dict(orient='split')
    app=df2['App'].unique()
    app=app.tolist()
    p={}
    n={}
    d=[]
    for i in app:
      p[i]=0
      n[i]=0
    for i in app:
      c1=0
      c2=0
      for j in range(len(z['data'])):
          if i==z['data'][j][0]:
              if z['data'][j][1]=='Positive':
                  c1+=1
                  p[i]=c1
              else:
                  c2+=1
                  n[i]=c2
    for j in app:
      if n[j]!=0:
          if (p[j]/n[j])==1:
              d.append(j)
    Max = max(p.items(), key=lambda x : x[1])
    Min = max(n.items(), key=lambda x : x[1])
```

```
    Label(fig12,text="App with Maximum Positive Sentiments ",width="50", height="3",
font=Label1_font, fg='white', bg='#ff9900').place(x=230,y=200)

    Label(fig12,text=Max[0],width="20", height="1", font=Label1_font, fg='white',
bg='#003366').place(x=230,y=300)

    Label(fig12,text=Max[1],width="20", height="1", font=Label1_font, fg='white',
bg='#003366').place(x=530,y=300)

    Label(fig12,text="App with Maximum Negative Sentiments : ",width="50", height="3",
font=Label1_font, fg='white', bg='#ff9900').place(x=230,y=400)

    Label(fig12,text=Min[0],width="20", height="1", font=Label1_font, fg='white',
bg='#003366').place(x=230,y=500)

    Label(fig12,text=Min[1],width="20", height="1", font=Label1_font, fg='white',
bg='#003366').place(x=530,y=500)

    c = 10

    Label(fig12,text="Apps with Same Positive to Negative Ratio",width="50", height="3",
font=Label1_font, fg='white', bg='#ff9900').place(x=900,y=200)

    for i in range(5):

        Label(fig12,text=d[i],width="50", height="1", font=Label1_font, fg='white',
bg='#003366').place(x=900,y=300+c)

        c+=50


def fig13():

    fig13 = Toplevel(home)

    fig13.title("QUESTION 13")

    createWindow(fig13)

    Label(fig13,text="Study and find out the relation between the Sentiment-polarity and
Sentiment-subjectivity of all the apps ? What is the sentiment subjectivity for a sentiment polarity of
0.4",width="170", height="5", font=Label1_font, fg='white', bg='#800000').place(x=0, y=0)

    b1 = Button(fig13, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
fg='white',command=backtohome).place(x=0, y=0)

    sp = list(df2['Sentiment_Polarity'])

    ss = list(df2['Sentiment_Subjectivity'])

    figure13 = py.Figure(figsize = (7,8) , dpi = 75)

    ax13 = figure13.add_subplot(111)

    ax13.set_title('Sentiment Polarity vs Subjectivity',fontsize=20)

    ax13.set_xlabel('Sentiment_Polarity',fontsize=20)

    ax13.set_ylabel('Sentiment_Subjectivity',fontsize=20)

    s = review_df[review_df['Sentiment_Polarity']==0.4]

    m = round(s['Sentiment_Subjectivity'].mean(),3)

    Label(fig13,text="The Sentiment Subjectivity for a Sentiment Polarity of 0.4 is = \n",bg="#e79700",
width="60", height="2", font=Button1_font, fg='white').place(x=100, y=400)

    Label(fig13,text=m,bg="#e79700", width="20", height="1", font=Button1_font,
fg='white').place(x=300, y=500)

    sns.regplot(sp,ss,scatter=True,ax=ax13,line_kws={"color":"r","alpha":0.7,"lw":5})

    canvas = FigureCanvasTkAgg(figure13, fig13)
```

```
    canvas.get_tk_widget().place(x=900, y=200)
    canvas.draw()
    # figure13.tight_layout()


def search_review(fig,search):
    if(len(search)==0):
        tm.showerror("Invalid!","App Name cannot be empty")
        searchEntry.focus_set()
    else:
        search = str(search)
        conn = pymysql.connect(host="localhost",user="root",passwd="",database="playstore")
        cur = conn.cursor()
        query = """ SELECT App,Positive,Negative,Neutral FROM app_review WHERE App = %s"""
        params = (search)
        cur.execute(query,params)
        search_query = cur.fetchall()
        conn.commit()
        conn.close()
        c=100
        texts = ['App Name :','No. of Positive Sentiments :','No. of Negative Sentiments :','No. of Neutral
 Sentiments :']
        for i in range(4):
          Label(fig, text=texts[i],width="60", height="3", font=Label1_font, fg='white',
 bg='#174873',anchor=W).place(x=50,y=300+c)
          Label(fig, text=search_query[0][i],width="20", height="3", font=Label1_font, fg='white',
 bg='#174873').place(x=300,y=300+c)
          c+=60
        figure14 = py.Figure(figsize = (7,8) , dpi = 75)
        ax14 = figure14.add_subplot(111)
        s = [int(x) for x in search_query[0][1:]]
        l = ['Positive','Negative','Neutral']
        ax14.bar(l,s,color=['r','g','b'])
        ax14.set_xticklabels(l, ha='center')
        ax14.set_ylabel("No. of Sentiments",fontsize=20)
        ax14.set_xlabel("Sentiments",fontsize=20)
        ax14.set_title("Classification of Sentiments",fontsize=20)
        canvas = FigureCanvasTkAgg(figure14, fig)
        canvas.get_tk_widget().place(x=700, y=150)
        canvas.draw()
        figure14.tight_layout()
```

```
def fig14():
    fig14 = Toplevel(home)
    fig14.title("QUESTION 14")
    createWindow(fig14)
    Label(fig14, text="Generate an interface where the client can see the reviews categorized as positive.negative and neutral ,once they \n have selected the app from a list of apps available for the study.",width="170", height="5", font=Label1_font, fg='white', bg='#800000').place(x=0, y=0)
    b1 = Button(fig14, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font, fg='white',command=backtohome).place(x=0, y=0)
#    df1 = df2[['App','Translated_Review','Sentiment_Polarity']]
#    apps = list(df1['App'].unique())
#    df1 = df1.to_dict(orient='split')
#    df = pd.DataFrame(columns=['App','Positive','Negative','Neutral'],index=range(len(apps)))
#    c =  0
#    for a in apps:
#      counts = [0,0,0]
#      df['App'].iloc[c] = a
#      for i in range(len(df1['data'])):
#        if(df1['data'][i][0]==a):
#          if(df1['data'][i][2]>0):
#            counts[0] += 1
#          elif(df1['data'][i][2]<0):
#            counts[1] += 1
#          elif(df1['data'][i][2]==0):
#            counts[2] += 1
#      df['Positive'].iloc[c] = counts[0]
#      df['Negative'].iloc[c] = counts[1]
#      df['Neutral'].iloc[c] = counts[2]
#      c+=1
#    final_df = df
#    final_df.to_excel('sentiments.xlsx')
    search = StringVar()
    global searchEntry
    Label(fig14, text="Search an App : ",width="50", height="5", font=Label1_font, fg='white', bg='#174873',anchor=N).place(x=100, y=200)
    searchEntry = Entry(fig14,textvar=search)
    searchEntry.place(x=300,y=260)
    b2 = Button(fig14, text="Search", bg="#e79700", width="10", height="1", font=Button1_font, fg='white',command=lambda:search_review(fig14,search.get())).place(x=150,y=250)
```

```
def fig15():
    fig15 = Toplevel(home)
    fig15.title("QUESTION 15")
    createWindow(fig15)
    Label(fig15, text="Is it advisable to launch an app like ’10 Best foods for you’? Do the users like these apps ?",width="170", height="5", font=Label1_font, fg='white', bg='#800000').place(x=0, y=0)
    b1 = Button(fig15, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font, fg='white',command=backtohome).place(x=0, y=0)
    kk = df2[(df2.App == '10 Best Foods for You') & (df2.Sentiment == 'Positive')]
    c = kk['Sentiment'].count()
    c1 = kk['Sentiment_Polarity'].sum()
    # print(c,c1)
    Label(fig15,text="The Total number of positive Sentiments recieved by the App are 162 \n\n and Total Sentiment Polarity is 95.37216720779222 ",width="100", height="8", font=Label1_font, fg='black', bg='white').place(x=300, y=200)
    Label(fig15,text="So it is Advisable to Launch the app like '10 BEST FOODS FOR YOU'",width="100", height="8", font=Label1_font, fg='black', bg='white').place(x=300, y=500)


def fig16():
    fig16 = Toplevel(home)
    fig16.title("QUESTION 16")
    createWindow(fig16)
    Label(fig16, text="Which month(s) of the year , is the best indicator to the average downloads that an app will generate over the entire year?",width="170", height="5", font=Label1_font, fg='white', bg='#800000').place(x=0, y=0)
    b1 = Button(fig16, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font, fg='white',command=backtohome).place(x=0, y=0)
    label = [0]*13
    for i in range(1,13):
      label[i] = calendar.month_name[i]
    x = dataframe[['Month','Installs']]
    df = x.groupby('Month')['Installs'].mean()
    p = label.pop(0)
    my_cmap = cm.get_cmap('jet')
    my_norm = Normalize(vmin=-3,vmax=3)
    t = np.array(list(range(5)))
    figure16 = py.Figure(figsize = (11,8) , dpi = 75)
    ax16 = figure16.add_subplot(111)
    ax16.bar(label,df,color=my_cmap(my_norm(t)))
    ax16.set_ylabel('DOWNLOADS',fontsize=20)
    ax16.set_xlabel('MONTHS',fontsize=20)
```

```
    ax16.set_title("Comparison of average downloads monthwise",fontsize=20)
    ax16.set_xticklabels(label,rotation=90)
    canvas = FigureCanvasTkAgg(figure16, fig16)
    canvas.get_tk_widget().place(x=370, y=150)
    canvas.draw()
    figure16.tight_layout()


def fig17():
    fig17 = Toplevel(home)
    fig17.title("QUESTION 17")
    createWindow(fig17)
    Label(fig17, text="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.",width="170", height="5",
 font=Label1_font, fg='white', bg='#800000').place(x=0, y=0)
    b1 = Button(fig17, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command=backtohome).place(x=0, y=0)
    figure17 = py.Figure(figsize = (9,8) , dpi = 75)
    ax17 = figure17.add_subplot(111)
    ax17.plot(fig5z,fig5y,linestyle='dashed',color='b')
    ax17.set_ylabel('Number of Downloads',fontsize=20)
    ax17.set_xlabel('Size of App in MB',fontsize=20)
    ax17.set_title('Trend of Size vs Downloads',fontsize=20)
    canvas = FigureCanvasTkAgg(figure17, fig17)
    canvas.get_tk_widget().place(x=400, y=150)
    canvas.draw()
    figure17.tight_layout()


def fig19():
    fig19 = Toplevel(home)
    fig19.title("QUESTION 19")
    createWindow(fig19)
    Label(fig19, text="Top 5 Apps in the Playstore ",width="170", height="5", font=Label1_font,
 fg='white', bg='#800000').place(x=0, y=0)
    b1 = Button(fig19, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command=backtohome).place(x=0, y=0)
    Maxfree = Question_19.Maxfree
    Maxpaid = Question_19.Maxpaid
    Trending = Question_19.Trending
    Label(fig19, text="Top Free Apps ",width="45", height="5", font=Label1_font, bg='#004d26',
 fg='#1aff8c').place(x=120, y=200)
    Label(fig19, text="Top Paid Apps ",width="45", height="5", font=Label1_font, bg='#004d26',
```

```
 fg='#1aff8c').place(x=520, y=200)

   Label(fig19, text="Trending Apps ",width="45", height="5", font=Label1_font, bg='#004d26',
 fg='#1aff8c').place(x=920, y=200)

   c=10

   for i in range(len(Maxfree)):

     Label(fig19, text="{0}) {1}".format(i+1,Maxfree[i]),width="45", height="3", font=Label1_font,
 fg='#004d26', bg='#1aff8c').place(x=120, y=300+c)

     c+=60

   c=10

   for i in range(len(Maxfree)):

     Label(fig19, text="{0}) {1}".format(i+1,Maxpaid[i]),width="45", height="3", font=Label1_font,
 fg='#004d26', bg='#1aff8c').place(x=520, y=300+c)

     c+=60

   c=10

   for i in range(len(Maxfree)):

     Label(fig19, text="{0}) {1}".format(i+1,Trending[i]),width="45", height="3", font=Label1_font,
 fg='#004d26', bg='#1aff8c').place(x=920, y=300+c)

     c+=60

#validation

def is_valid_email(email):

   # Connecetion for mysql database

   conn = pymysql.connect(user="root", password="", host="localhost",database="playstore")

   cur = conn.cursor()

      # Excuting insert query

   select_query = """select 1 from student_details where email=%s"""

   mail=(email)

   cur.execute(select_query,mail)

   result = cur.fetchall()

   if len(result)>0:

      tm.showerror("Invalid","User name already exists!!....Try something new")

      return False

   conn.commit()

   conn.close()

   if len(email) > 7:

      return bool(re.match("^.+@(\[?)[a-zA-Z0-9-.]+.([a-zA-Z]{2,3}|[0-9]{1,3})(]?)$",email))

   return False


# Validating for each input

def validate1(event , input):

  if( input == "Candidate name"):
```

```
#    print(cand_name.get()
    if all(x.isalpha() or x.isspace() for x in cand_name.get()) and (len(cand_name.get())> 0):
      emailEntry.focus_set()
      emailEntry.config(state='normal')
    else:
      tm.showerror("Invalid!" ,"Candidate name cannot contain digits or special characters . Spaces are
 allowed. ")
      candNameEntry.focus_set()
  elif( input == "email"):
    if is_valid_email(email.get()):
      mobileEntry.focus_set()
      mobileEntry.config(state='normal')
    else:
      tm.showerror("Invalid!" ,"Incorrect email formats. ")
      emailEntry.focus_set()
  elif( input == "mobile"):
    if len(mobile.get()) == 10 and mobile.get().isdigit():
      passEntry.focus_set()
      passEntry.config(state='normal')
    else:
      tm.showerror("Invalid!" ,"Mobile number: digits only,strictly 10 digits. ")
      mobileEntry.focus_set()
  elif(input=="password"):
    word = password.get()
    if(re.match(r"^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#\$%\^&\*])(?=.{6,})", word)):
      repassEntry.focus_set()
      repassEntry.config(state='normal')
    else:
      tm.showerror("Invalid","Password should contain 1 capital,1 special,1 number letter")
      passEntry.focus_set()
  elif(input=="repassword"):
    if(str(password.get()) != str(repassword.get())):
      tm.showerror("Invalid","Password does not match ")
      repassEntry.focus_set()


def validate(event,input):
  if(input=='Application Name'):
    if all(x.isalnum() or x.isspace() for x in app.get()) and (len(app.get())> 0):
      catEntry.focus_set()
```

```
    catEntry.config(state='normal')
  else:
    tm.showerror("Invalid!" ,"Application name not valid . Spaces are allowed.")
    appEntry.focus_set()
elif(input=='Category'):
  if(category.get()!='--select category --'):
    ratEntry.focus_set()
    ratEntry.config(state='normal')
  else:
    tm.showerror("Invalid!" ,"Category is not selected.")
    catEntry.focus_set()
elif(input=='Ratings'):
  rating = float(ratings.get())
  if(isinstance(rating,(int,float)) and (0<=rating<=5)):
    reEntry.focus_set()
    reEntry.config(state='normal')
  else:
    tm.showerror("Invalid!","Rating should be a floating number between 0 & 5!")
    ratEntry.focus_set()
elif(input=='Reviews'):
  if(str(reviews.get()).isnumeric()):
    sizeEntry.focus_set()
    sizeEntry.config(state='normal')
  else:
    tm.showerror("Invalid!","Reviews should be a whole number!")
    reEntry.focus_set()
elif(input=='Size'):
  if(len(size.get())>0):
    insEntry.focus_set()
    insEntry.config(state='normal')
  else:
    tm.showerror("Invalid!","Size cannot be empty!")
    sizeEntry.focus_set()
elif(input=='Installs'):
  if(str(installs.get()).isnumeric()):
    typeEntry.focus_set()
    typeEntry.config(state='normal')
  else:
```

```
    tm.showerror("Invalid!","Installs should be a whole number!")
    insEntry.focus_set()
elif(input=='Type'):
  if(apptype.get()!='--select type --'):
    priceEntry.focus_set()
    priceEntry.config(state='normal')
  else:
    tm.showerror("Invalid!","Type cannot be empty!")
    typeEntry.focus_set()
elif(input=='Price'):
  if(len(price.get())>0):
    conEntry.focus_set()
    conEntry.config(state='normal')
  else:
    tm.showerror("Invalid!","Price cannot be empty!")
    priceEntry.focus_set()
elif(input=='Content Rating'):
  if(contentrating.get()!='--select content rating --'):
    gEntry.focus_set()
    gEntry.config(state='normal')
  else:
    tm.showerror("Invalid!","Content Rating cannot be empty!")
    conEntry.focus_set()
elif(input=='Genres'):
  if(genres.get()==category.get()):
    luEntry.focus_set()
    luEntry.config(state='normal')
  else:
    tm.showerror("Invalid!","Genre should be same as category!")
    gEntry.focus_set()
elif(input=='Last Updated'):
  if(len(lastupdated.get())>0):
    cvEntry.focus_set()
    cvEntry.config(state='normal')
  else:
    tm.showerror("Invalid!","Last Updated cannot be empty!")
    luEntry.focus_set()
elif(input=='Current Version'):
```

```
  if(len(currentver.get())>0):
    avEntry.focus_set()
    avEntry.config(state='normal')
  else:
    tm.showerror("Invalid!","Current Version cannot be empty!")
    cvEntry.focus_set()
elif(input=='Android Version'):
  if(len(androidver.get())>0):
    print("Success")
  else:
    tm.showerror("Invalid!","Android Version cannot be empty!")
    avEntry.focus_set()
elif(input=='Application'):
  if all(x.isalnum() or x.isspace() for x in application.get()) and (len(application.get())> 0):
    reviewEntry.focus_set()
    reviewEntry.config(state='normal')
  else:
    tm.showerror("Invalid!","Application Name not valid .Spaces are allowed. ")
    appEntry2.focus_set()
elif(input=='Translated Review'):
  if(len(trans_review.get())>0):
    sentEntry.focus_set()
    sentEntry.config(state='normal')
  else:
    tm.showerror("Invalid!","Translated Review cannot be empty")
    reviewEntry.focus_set()
elif(input=='Sentiment'):
  if(str(sentiment.get())!='--select sentiment --'):
    polEntry.focus_set()
    polEntry.config(state='normal')
  else:
    tm.showerror("Invalid!","Sentiment cannot be empty")
    sentEntry.focus_set()
elif(input=='Sentiment Polarity'):
  polarity = float(senti_polarity.get())
  if(isinstance(polarity,(int,float))):
    subEntry.focus_set()
    subEntry.config(state='normal')
```

```
    else:
      tm.showerror("Invalid!","Sentiment Polarity should be a floating number")
      polEntry.focus_set()
  elif(input=='Sentiment Subjectivity'):
    if(isinstance(float(senti_subject.get()),(int,float))):
      print("Success")
    else:
      tm.showerror("Invalid!","Sentiment Subjectivity should be a floating number")
      subEntry.focus_set()
#--
def check_entry(new_entry):
    global df
    Label(new_entry,text="Updated succesfully", width="25", height="1", font=Label_font, bg='brown',
 fg='white').place(x=50,y=600)
    d = datetime.strptime(str(lastupdated.get()),'%d/%m/%Y')
    appdetails = {'App': app.get(), 'Category': category.get(), 'Rating': ratings.get(), 'Reviews':
 reviews.get(),
                  'Size': size.get(), 'Installs': installs.get(), 'Type': apptype.get(), 'Price': price.get(),
                  'Content Rating': contentrating.get(), 'Genres': genres.get(), 'Last Updated': d,
                  'Current Ver': currentver.get(), 'Android Ver':
 androidver.get(),'Year':d.year,'Month':d.month}
    column = list(appdetails.keys())
    new_data_frame = pd.DataFrame([appdetails],columns = column)
    df = df.append(new_data_frame,ignore_index=True,sort = False)
    df.to_csv("APP_DETAILS.csv",index=False)

def check_review(new_review):
    global review_df
    Label(new_review,text="Updated succesfully", width="25", height="1", font=Label_font,
 bg='brown', fg='white',anchor=W).place(x=50,y=520)
    review_details =
{'App':application.get(),'Translated_Review':trans_review.get(),'Sentiment':sentiment.get(),'Sentiment_
 Polarity': senti_polarity.get(),'Sentiment_Subjectivity':senti_subject.get()}
    column = list(review_details.keys())
    new_data_frame = pd.DataFrame([review_details],columns = column)
    review_df = review_df.append(new_data_frame,ignore_index = True , sort = False)
    review_df.to_csv("REVIEW.csv",index=False)

def new_entry():
    global app, category, ratings, reviews, size, installs, apptype, price, contentrating, genres,
 lastupdated, currentver, androidver
```

```
  global
appEntry,catEntry,ratEntry,reEntry,sizeEntry,insEntry,typeEntry,priceEntry,conEntry,gEntry,luEntry,cv
Entry,avEntry

  new_entry = Toplevel(fig18)

  new_entry.title("New Entry")

  createWindow(new_entry)


  app = StringVar(new_entry)

  category = StringVar(new_entry)

  ratings = StringVar(new_entry)

  reviews = StringVar(new_entry)

  size = StringVar(new_entry)

  installs = StringVar(new_entry)

  apptype = StringVar(new_entry)

  price = StringVar(new_entry)

  contentrating = StringVar(new_entry)

  genres = StringVar(new_entry)

  lastupdated = StringVar(new_entry)

  currentver = StringVar(new_entry)

  androidver = StringVar(new_entry)


  Label(new_entry, text="Enter a New App Entry ", width="500", height="2", font=Label_font,
fg='white',bg='#800000').pack()


  Label(new_entry, text="", bg='#1aff8c', width='125', height='35').place(x=50, y=120)

  Label(new_entry, text="Application Name", font=("Open Sans", 11, 'bold'), fg='#004d26',
bg='#1aff8c', anchor=W).place(x=130, y=160)

  appEntry=Entry(new_entry, textvar=app)

  appEntry.place(x=300, y=160)

  appEntry.bind("<Return>", lambda event : validate(event, "Application Name"))

  appEntry.bind("<Tab>", lambda event : validate(event, "Application Name"))


  Label(new_entry, text="Category", font=("Open Sans", 11, 'bold'), fg='#004d26', bg='#1aff8c',
anchor=W).place(x=130,y=210)

  catEntry=Entry(new_entry, textvar=category)

  catEntry.place(x=300, y=210)

  list1 = cat

  droplist = OptionMenu(new_entry, category, *list1)

  droplist.config(width=17)

  category.set('--select category --')
```

```
  droplist.place(x=300, y=210)
  catEntry.bind("<Return>", lambda event : validate(event, "Category"))
  catEntry.bind("<Tab>", lambda event : validate(event, "Category"))

  Label(new_entry, text="Ratings", font=("Open Sans", 11, 'bold'), fg='#004d26', bg='#1aff8c',
anchor=W).place(x=130,y=260)
  ratEntry=Entry(new_entry, textvar=ratings)
  ratEntry.place(x=300, y=260)
  ratEntry.bind("<Return>", lambda event : validate(event, "Ratings"))
  ratEntry.bind("<Tab>", lambda event : validate(event, "Ratings"))

  Label(new_entry, text="Reviews", font=("Open Sans", 11, 'bold'), fg='#004d26', bg='#1aff8c',
anchor=W).place(x=130,y=310)
  reEntry=Entry(new_entry, textvar=reviews)
  reEntry.place(x=300, y=310)
  reEntry.bind("<Return>", lambda event : validate(event, "Reviews"))
  reEntry.bind("<Tab>", lambda event : validate(event, "Reviews"))

  Label(new_entry, text="Size", font=("Open Sans", 11, 'bold'), fg='#004d26', bg='#1aff8c',
anchor=W).place(x=130,y=360)
  sizeEntry=Entry(new_entry, textvar=size)
  sizeEntry.place(x=300, y=360)
  sizeEntry.bind("<Return>", lambda event : validate(event, "Size"))
  sizeEntry.bind("<Tab>", lambda event : validate(event, "Size"))

  Label(new_entry, text="Installs", font=("Open Sans", 11, 'bold'), fg='#004d26', bg='#1aff8c',
anchor=W).place(x=130,y=410)
  insEntry=Entry(new_entry, textvar=installs)
  insEntry.place(x=300, y=410)
  insEntry.bind("<Return>", lambda event : validate(event, "Installs"))
  insEntry.bind("<Tab>", lambda event : validate(event, "Installs"))

  Label(new_entry, text="Type", font=("Open Sans", 11, 'bold'), fg='#004d26', bg='#1aff8c',
anchor=W).place(x=130,y=460)
  typeEntry=Entry(new_entry, textvar=apptype)
  typeEntry.place(x=300, y=460)
  list2 = ['Free','Paid']
  droplist = OptionMenu(new_entry, apptype, *list2)
  droplist.config(width=17)
  apptype.set('--select type --')
```

```
  droplist.place(x=300, y=460)
  typeEntry.bind("<Return>", lambda event : validate(event, "Type"))
  typeEntry.bind("<Tab>", lambda event : validate(event, "Type"))

  Label(new_entry, text="Price", font=("Open Sans", 11, 'bold'), fg='#004d26', bg='#1aff8c',
anchor=W).place(x=500,y=160)
  priceEntry=Entry(new_entry, textvar=price)
  priceEntry.place(x=670, y=160)
  priceEntry.bind("<Return>", lambda event : validate(event, "Price"))
  priceEntry.bind("<Tab>", lambda event : validate(event, "Price"))

  Label(new_entry, text="Content Rating", font=("Open Sans", 11, 'bold'), fg='#004d26', bg='#1aff8c',
anchor=W).place(x=500, y=210)
  conEntry=Entry(new_entry, textvar=contentrating)
  conEntry.place(x=670, y=210)
  list3 = ['Everyone', 'Teen', 'Everyone 10', 'Mature 17', 'Adults only 18','Unrated']
  droplist = OptionMenu(new_entry, contentrating, *list3)
  droplist.config(width=17)
  contentrating.set('--select content rating --')
  droplist.place(x=670, y=210)
  conEntry.bind("<Return>", lambda event : validate(event, "Content Rating"))
  conEntry.bind("<Tab>", lambda event : validate(event, "Content Rating"))

  Label(new_entry, text="Genres", font=("Open Sans", 11, 'bold'), fg='#004d26', bg='#1aff8c',
anchor=W).place(x=500,y=260)
  gEntry=Entry(new_entry, textvar=genres)
  gEntry.place(x=670, y=260)
  list1 = cat
  droplist = OptionMenu(new_entry, genres, *list1)
  droplist.config(width=17)
  genres.set('--select genres --')
  droplist.place(x=670, y=260)
  gEntry.bind("<Return>", lambda event : validate(event, "Genres"))
  gEntry.bind("<Tab>", lambda event : validate(event, "Genres"))

  Label(new_entry, text="Last Updated", font=("Open Sans", 11, 'bold'), fg='#004d26', bg='#1aff8c',
anchor=W).place(x=500, y=310)
  luEntry=DateEntry(new_entry , textvariable = lastupdated , date_pattern='dd/mm/y')
  luEntry.place(x=670, y=310)
  luEntry.bind("<Return>", lambda event : validate(event, "Last Updated"))
```

```
    luEntry.bind("<Tab>", lambda event : validate(event, "Last Updated"))


    Label(new_entry, text="Current Version", font=("Open Sans", 11, 'bold'), fg='#004d26',
 bg='#1aff8c', anchor=W).place(x=500, y=360)
    cvEntry=Entry(new_entry, textvar=currentver)
    cvEntry.place(x=670, y=360)
    cvEntry.bind("<Return>", lambda event : validate(event, "Current Version"))
    cvEntry.bind("<Tab>", lambda event : validate(event, "Current Version"))


    Label(new_entry, text="Android Version", font=("Open Sans", 11, 'bold'), fg='#004d26',
 bg='#1aff8c', anchor=W).place(x=500, y=410)
    avEntry=Entry(new_entry, textvar=androidver)
    avEntry.place(x=670, y=410)
    avEntry.bind("<Return>", lambda event : validate(event, "Android Version"))
    avEntry.bind("<Tab>", lambda event : validate(event, "Android Version"))


    ratEntry.config(state='disabled')
    reEntry.config(state='disabled')
    sizeEntry.config(state='disabled')
    insEntry.config(state='disabled')
    typeEntry.config(state='disabled')
    priceEntry.config(state='disabled')
    conEntry.config(state='disabled')
    # luEntry.config(state='disabled')
    cvEntry.config(state='disabled')
    avEntry.config(state='disabled')
    Button(new_entry, text='Submit', width=20, font=("Open Sans", 13, 'bold'), bg='brown',
 fg='white',command=lambda : check_entry(new_entry)).place(x=500, y=460)


def new_review():
    global application, trans_review, sentiment, senti_polarity, senti_subject
    global appEntry2, reviewEntry, sentEntry, polEntry, subEntry
    new_review = Toplevel(fig18)
    new_review.title("New Review")
    createWindow(new_review)
    application = StringVar(new_review)
    trans_review = StringVar(new_review)
    sentiment = StringVar(new_review)
    senti_polarity = StringVar(new_review)
    senti_subject = StringVar(new_review)
```

```
  Label(new_review, text="Enter a New App Review", width="500", height="2", font=Label_font, fg='white',bg='#800000').pack()

  Label(new_review, text="", bg='#1aff8c', width='80', height='30').place(x=50, y=120)


  Label(new_review, text="Application", font=("Open Sans", 11, 'bold'),fg='#004d26', bg='#1aff8c', anchor=W).place(x=130, y=160)

  appEntry2 = Entry(new_review, textvar=application)

  appEntry2.place(x=350, y=160)

  appEntry2.bind("<Return>", lambda event : validate(event, "Application"))

  appEntry2.bind("<Tab>", lambda event : validate(event, "Application"))


  Label(new_review, text="Translated Review", font=("Open Sans", 11, 'bold'),fg='#004d26', bg='#1aff8c',anchor=W).place(x=130, y=210)

  reviewEntry = Entry(new_review, textvar=trans_review)

  reviewEntry.place(x=350, y=210)

  reviewEntry.bind("<Return>", lambda event : validate(event, "Translated Review"))

  reviewEntry.bind("<Tab>", lambda event : validate(event, "Translated Review"))


  Label(new_review, text="Sentiment", font=("Open Sans", 11, 'bold'),fg='#004d26', bg='#1aff8c', anchor=W).place(x=130,y=260)

  sentEntry = Entry(new_review, textvar=sentiment)

  sentEntry.place(x=350, y=260)

  list1 = ['Positive','Negative','Neutral']

  droplist = OptionMenu(new_review, sentiment, *list1)

  droplist.config(width=17)

  sentiment.set('--select sentiment --')

  droplist.place(x=350, y=260)

  sentEntry.bind("<Return>", lambda event : validate(event, "Sentiment"))

  sentEntry.bind("<Tab>", lambda event : validate(event, "Sentiment"))


  Label(new_review, text="Sentiment Polarity", font=("Open Sans", 11, 'bold'), fg='#004d26', bg='#1aff8c',anchor=W).place(x=130, y=310)

  polEntry = Entry(new_review, textvar=senti_polarity)

  polEntry.place(x=350, y=310)

  polEntry.bind("<Return>", lambda event : validate(event, "Sentiment Polarity"))

  polEntry.bind("<Tab>", lambda event : validate(event, "Sentiment Polarity"))


  Label(new_review, text="Sentiment Subjectivity", font=("Open Sans", 11, 'bold'), fg='#004d26', bg='#1aff8c',anchor=W).place(x=130, y=360)

  subEntry = Entry(new_review, textvar=senti_subject)
```

```
    subEntry.place(x=350, y=360)
    subEntry.bind("<Return>", lambda event : validate(event, "Sentiment Subjectivity"))
    subEntry.bind("<Tab>", lambda event : validate(event, "Sentiment Subjectivity"))
    reviewEntry.config(state='disabled')
    sentEntry.config(state='disabled')
    polEntry.config(state='disabled')
    subEntry.config(state='disabled')
    Button(new_review, text='Submit', width=15, font=("Open Sans", 13, 'bold'), bg='brown',
 fg='white',command=lambda : check_review(new_review)).place(x=350, y=410)


def fig18():
    global fig18
    fig18 = Toplevel(home)
    fig18.title("QUESTION 18")
    createWindow(fig18)
    Label(fig18, text="Provide an interface to add new data to both the datasets provided.",
 width="170", height="5",font=Label1_font, fg='white', bg='#800000').place(x=0, y=0)
    b1 = Button(fig18, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command=backtohome).place(x=0, y=0)
    b2 = Button(fig18, text="New App Record", bg="#e79700", width="25", height="1",
 font=Button1_font, fg='white',command=new_entry).place(x=250, y=220)
    b3 = Button(fig18, text="Review App", bg="#e79700", width="25", height="1", font=Button1_font,
 fg='white',command=new_review).place(x=650, y=220)


def callApp1(category,fig):
    global app1,app2
    app1 = StringVar(fig)
    Label(fig, text="App1", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873',
 anchor=W).place(x=800,y=200)
    apps = list(Question_20.d2[category].keys())
    droplist = OptionMenu(fig,app1, *apps)
    droplist.config(width=17)
    app1.set("-- select app1 --")
    droplist.place(x=850, y=200)
    b3 = Button(fig, text="Select", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command = lambda : App2(apps,str(app1.get()),fig,category)).place(x=850, y=250)


def compare(fig,app1,app2,category):
    l1=[]
    l2=[]
    l3=['RATING','DOWNLOADS','SIZE','CONTENT RATING','COST']
```

```
    l1=Question_20.d2[category][app1]
    l2=Question_20.d2[category][app2]
    l4=['PROPERTY',app1,app2]
    c=0
    for i in range(len(l4)):
      Label(fig, text=l4[i],width="25", height="4", font=Label1_font,  bg='#004d26',
 fg='#1aff8c').place(x=400+c, y=300)
      c+=300
    c=10
    for i in range(len(l1)):
      Label(fig, text=l3[i],width="25", height="4", font=Label1_font, bg='#004d26',
 fg='#1aff8c').place(x=400, y=400+c)
      c+=60
    c=10
    for i in range(len(l1)):
      Label(fig, text=l1[i],width="25", height="4", font=Label1_font,  fg='#004d26',
 bg='#1aff8c').place(x=700, y=400+c)
      c+=60
    c=10
    for i in range(len(l2)):
      Label(fig, text=l2[i],width="25", height="4", font=Label1_font,  fg='#004d26',
 bg='#1aff8c').place(x=1000, y=400+c)
      c+=60


def App2(apps,app1,fig,category):
    app2 = StringVar(fig)
    Label(fig, text="App2", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873',
 anchor=W).place(x=1200,y=200)
    apps.remove(app1)
    droplist = OptionMenu(fig,app2, *apps)
    droplist.config(width=17)
    app2.set("-- select app2 --")
    droplist.place(x=1250, y=200)
    b3 = Button(fig, text="Select", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command = lambda : compare(fig,app1,str(app2.get()),category)).place(x=1250, y=250)


def fig20():
    global fig20
    global cate
    fig20 = Toplevel(home)
    fig20.title("QUESTION 20")
```

```
    createWindow(fig20)

    cate=StringVar(fig20)

    Label(fig20, text="COMPARISON OF 2 APPS", width="160", height="5",font=Label1_font,
 fg='white', bg='#800000').place(x=0, y=0)

    b1 = Button(fig20, text="HOME PAGE", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command=backtohome).place(x=0, y=0)

    cat=Question_20.cat

    Label(fig20, text="Categoreis", font=("Open Sans", 11, 'bold'), fg='white', bg='#174873',
 anchor=W).place(x=300,y=200)

    droplist = OptionMenu(fig20,cate, *cat)

    droplist.config(width=17)

    cate.set('--select type --')

    droplist.place(x=500, y=200)

    b2 = Button(fig20, text="Select", bg="#e79700", width="10", height="1", font=Button1_font,
 fg='white',command=lambda : callApp1(str(cate.get()),fig20)).place(x=500, y=250)


def destroy():

    screen.destroy()

def home():

    global home

    home = Toplevel(screen)

    home.title("HOMEPAGE")

    createWindow(home)

    b = Button(home,text="Logout",width="15",height="1",font=("Open Sans", 13,
 'bold'),fg="white",bg="#e79700",command=destroy).place(x=1150,y=600)

    l = Label(home, text="GOOGLE PLAYSTORE APP LAUNCH STUDY", width="500", height="2",
 font=Label_font, fg='white',bg='#d61a1f').pack()

    l1 = Label(home, text="Percentage download in each category", width="40", height="1",
 font=Label_font, fg='white',bg='#99ccff').place(x=25, y=90)

    b1 = Button(home, text="FIG 1", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
 'bold'), fg='white',command=fig1).place(x=25, y=90)

    l2 = Label(home, text="Number of Downloads", width="40", height="1", font=Label_font,
 fg='white',bg='#4da6ff').place(x=25, y=140)

    b2 = Button(home, text="FIG 2", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
 'bold'), fg='white',command=fig2).place(x=25, y=140)

    l3 = Label(home, text="Most,Least,Average Category", width="40", height="1", font=Label_font,
 fg='white',bg='#3399ff').place(x=25, y=190)

    b3 = Button(home, text="FIG 3", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
 'bold'), fg='white',command=fig3).place(x=25, y=190)

    l4 = Label(home, text="Highest maximum average ratings", width="40", height="1",
 font=Label_font, fg='white',bg='#1a8cff').place(x=25, y=240)

    b4 = Button(home, text="FIG 4", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
 'bold'), fg='white',command=fig4).place(x=25, y=240)

    l5 = Label(home, text="Downloads in each size range", width="40", height="1", font=Label_font,
 fg='white',bg='#0080ff').place(x=25, y=290)
```

```
  b5 = Button(home, text="FIG 5", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
'bold'), fg='white',command=fig5).place(x=25, y=290)

  l6 = Label(home, text="Downloads over period of three years", width="40", height="1",
font=Label_font, fg='white',bg='#0073e6').place(x=25, y=340)

  b6 = Button(home, text="FIG 6", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
'bold'), fg='white',command=fig6).place(x=25, y=340)

  l7 = Label(home, text="Android version is not an issue", width="40", height="1", font=Label_font,
fg='white',bg='#0066cc').place(x=25, y=390)

  b7 = Button(home, text="FIG 7", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
'bold'), fg='white',command=fig7).place(x=25, y=390)

  l8 = Label(home, text="Most likely to be downloaded", width="40", height="1", font=Label_font,
fg='white',bg='#0059b3').place(x=25, y=440)

  b8 = Button(home, text="FIG 8", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
'bold'), fg='white',command=fig8).place(x=25, y=440)

  l9 = Label(home, text="Co-relation of downloads & ratings", width="40", height="1",
font=Label_font, fg='white',bg='#004d99').place(x=25, y=490)

  b9 = Button(home, text="FIG 9", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
'bold'), fg='white',command=fig9).place(x=25, y=490)

  l10 = Label(home, text="Qualifies as teen versus mature 17+.", width="40", height="1",
font=Label_font, fg='white',bg='#003366').place(x=25, y=540)

  b10 = Button(home, text="FIG 10", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
'bold'), fg='white',command=fig10).place(x=25, y=540)

  l11 = Label(home, text="Downloads in each quarter of year", width="50",
height="1",font=Label_font, fg='white', bg='#99ccff').place(x=750, y=90)

  b11 = Button(home, text="FIG 11", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
'bold'), fg='white',command=fig11).place(x=750, y=90)

  l12 = Label(home, text="Generate most positive & negative sentiments", width="50", height="1",
font=Label_font,fg='white', bg='#4da6ff').place(x=750, y=140)

  b12 = Button(home, text="FIG 12", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
'bold'), fg='white',command=fig12).place(x=750, y=140)

  l13 = Label(home, text="Relation between Sentiment-polarity & subjectivity", width="50",
height="1",font=Label_font, fg='white', bg='#3399ff').place(x=750, y=190)

  b13 = Button(home, text="FIG 13", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
'bold'), fg='white',command=fig13).place(x=750, y=190)

  l14 = Label(home, text="Reviews categorized as positive,negative & neutral", width="50",
height="1",font=Label_font, fg='white', bg='#1a8cff').place(x=750, y=240)

  b14 = Button(home, text="FIG 14", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
'bold'), fg='white',command=fig14).place(x=750, y=240)

  l15 = Label(home, text="Advisable to launch app like 10 Best foods ?", width="50",
height="1",font=Label_font, fg='white', bg='#0080ff').place(x=750, y=290)

  b15 = Button(home, text="FIG 15", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
'bold'), fg='white',command=fig15).place(x=750, y=290)

  l16 = Label(home, text="Month with highest downloads in entire year?", width="50",
height="1",font=Label_font, fg='white', bg='#0073e6').place(x=750, y=340)

  b16 = Button(home, text="FIG 16", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
'bold'), fg='white',command=fig16).place(x=750, y=340)

  l17 = Label(home, text="Influence of size of app on downloads", width="50", height="1",
font=Label_font,fg='white', bg='#0066cc').place(x=750, y=390)
```

```
    b17 = Button(home, text="FIG 17", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
 'bold'), fg='white',command=fig17).place(x=750, y=390)


    l18 = Label(home, text="Interface to add new data to both datasets", width="50", height="1",
 font=Label_font,fg='white', bg='#0059b3').place(x=750, y=440)

    b18 = Button(home, text="FIG 18", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
 'bold'), fg='white',command=fig18).place(x=750, y=440)


    l19 = Label(home, text="Top 5 Apps in the Playstore", width="50", height="1",
 font=Label_font,fg='white', bg='#004d99').place(x=750, y=490)

    b19 = Button(home, text="FIG 19", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
 'bold'), fg='white',command=fig19).place(x=750, y=490)


    l20 = Label(home, text="Comparison of Apps of same category", width="50", height="1",
 font=Label_font,fg='white', bg='#003366').place(x=750, y=540)

    b20 = Button(home, text="FIG 20", bg="#e79700", width="5", height="1", font=("Open Sans", 13,
 'bold'), fg='white',command=fig20).place(x=750, y=540)


    # home.mainloop()

# home()

def register():

    root = Toplevel(screen)

    adjustWindow(root)

    Label(root, text="REGISTER HERE", width="500", height="2",font=("Calibri", 22, 'bold'), fg='white',
 bg='#d9660a').pack()

    photo=PhotoImage(file="mobile1.png")

    label=Label(root,borderwidth=0,image=photo)

    label.place(x=0,y=80)

    root.title("Student Data")

    mainframe = Frame(root)

    mainframe.place(x=500,y=80)

    global
 cand_name,email,mobile,tnc,candNameEntry,genderValue,emailEntry,mobileEntry,passEntry,repassE
 ntry,password,repassword                # order the elements of mainframe

    #Setting string variable for 6 input

    cand_name =  StringVar()

    password = StringVar()

    repassword = StringVar()

    email = StringVar()

    mobile = StringVar()

    candNameEntry = Entry(mainframe, width=30, textvar=cand_name)

    candNameEntry.grid(row=11, column=1 ,padx=25, pady=25)

    candNameEntry.bind("<Return>", lambda event: validate1(event, "Candidate name"))
```

```
  candNameEntry.bind("<Tab>", lambda event: validate1(event, "Candidate name"))

  emailEntry = Entry(mainframe, width=30, textvar=email)

  emailEntry.grid(row=14, column=1 ,padx=25, pady=25)

  emailEntry.bind("<Return>", lambda event: validate1(event, "email"))

  emailEntry.bind("<Tab>", lambda event: validate1(event, "email"))

  mobileEntry = Entry(mainframe, width=30, textvar=mobile)

  mobileEntry.grid(row=17, column=1 ,padx=25, pady=25)

  mobileEntry.bind("<Return>", lambda event: validate1(event, "mobile"))

  mobileEntry.bind("<Tab>", lambda event: validate1(event, "mobile"))

  passEntry = Entry(mainframe, textvar=password,width=30, show="*")

  passEntry.grid(row=20,column=1,padx=25,pady=25)

  passEntry.bind("<Return>", lambda event: validate1(event, "password"))

  passEntry.bind("<Tab>", lambda event: validate1(event, "password"))

  repassEntry=Entry(mainframe, textvar=repassword,width=30, show="*")

  repassEntry.grid(row=23,column=1,padx=25,pady=25)

  repassEntry.bind("<Return>", lambda event: validate1(event, "repassword"))

  repassEntry.bind("<Tab>", lambda event: validate1(event, "repassword"))

  passEntry.config(state='disabled')

  repassEntry.config(state='disabled')

  emailEntry.config(state='disabled')

  mobileEntry.config(state='disabled')

  genderValue = StringVar()

  genderFrame = Frame(mainframe)     # genderFrame is subpart of mainframe

  Radiobutton(genderFrame, text="Male" , variable=genderValue, value="Male").grid(row=0,
column=1, padx=5, pady=5)

  Radiobutton(genderFrame, text="Female" , variable=genderValue,value="Female").grid(row=0,
column=2, padx=5, pady=5)

  genderFrame.grid(row=26, column=1, padx=25, pady=25)

  genderValue.set("Male")

  btnFrame = Frame(mainframe)

  Button(btnFrame, text="Submit", command=saveStudent).grid(row=0, column=1, padx=25,
pady=25)

  Button(btnFrame, text="Cancel", command=cancel).grid(row=0, column=2, padx=25, pady=25)

  btnFrame.grid(row=30, column=1, padx=25, pady=25)

  Label(mainframe, text='Full Name:*', anchor='w').grid(row=11, column=0, padx=25, pady=25,
sticky="w")

  Label(mainframe, text='Email:*', anchor='w').grid(row=14, column=0 ,padx=25, pady=25,
sticky="w")

  Label(mainframe, text='Mobile Number:', anchor='w').grid(row=17, column=0 , padx=25, pady=25,
sticky="w")

  Label(mainframe, text="Password:",
```

```
anchor='w').grid(row=20,column=0,padx=25,pady=25,sticky="w")

  Label(mainframe, text="Re-Password:", anchor='w').grid(row=23, column=0 , padx=25, pady=25,
sticky="w")

  Label(mainframe, text='Gender:*', anchor='w').grid(row=26, column=0 , padx=25, pady=25,
sticky="w")


def login_verify():

  global studentID

  connection = pymysql.connect(host="localhost", user="root", passwd="",database="playstore") #
database connection

  cursor = connection.cursor()

  select_query = "SELECT * FROM student_details where email = '" +username_verify.get() + "' AND
password = '" + password_verify.get() + "';" # queries for retrieving values

  cursor.execute(select_query) # executing the queries

  student_info = cursor.fetchall()

  connection.commit() # commiting the connection then closing it.

  connection.close() # closing the connection of the database

  if student_info:

    messagebox.showinfo("Congratulation", "Login Successful") # displaying message for successful
login

    home()

  else:

    messagebox.showerror("Error", "Invalid Username or Password") # displaying message for invalid
details


def main_screen():

  global screen, username_verify, password_verify

  screen = Tk() # initializing the tkinter window

  username_verify = StringVar()

  password_verify = StringVar()

  screen.title("GOOGLE PLAY STORE LAUNCH APP") # mentioning title of the window

  adjustWindow(screen) # configuring the window

  photo=PhotoImage(file="mobile1.png")

  label=Label(screen,borderwidth=0,image=photo)

  label.place(x=0,y=80)

  Label(screen, text="LOGIN/SIGN UP", width="150", height="1",font=("Calibri",50, 'bold'),
fg='white', bg='#800000').pack()

  Label(screen, text="Please enter details below to
login",width="40",height="2",fg='white',bg='#174873',font=("Calibri",25,'bold')).pack(padx=25,pady=
25)

  Label(screen, text="Username", font=("Open Sans", 20,
'bold'),width="10",height="1",fg='white',bg='#d9660a').place(x=550,y=300)

  Entry(screen, textvar=username_verify).place(x=850,y=300)
```

```
    Label(screen, text="Password", font=("Open Sans", 20,
 'bold'),width="10",height="1",fg='white',bg='#d9660a').place(x=550,y=400)

    Entry(screen, textvar=password_verify, show="*").place(x=850,y=400)

    Button(screen, text="LOGIN", bg="#e79700", width="15", height="1", font=("Open Sans",13,
 'bold'), fg='white', command=login_verify).place(x=700,y=500)

    Button(screen, text="New User? Register Here", height="2", width="30",
 bg='#e79700',font=("Open Sans", 10, 'bold'), fg='white', command=register).place(x=660,y=600)

    screen.mainloop()

main_screen()
```