Machine Learning & GUI Based Project

Application Design for an app launch on Google PlayStore

Submitted by team: COLORADO BISON

Team Members: 1) ATHARV SANDEEP DESAI

2)KAUSTUBH NITIN PRABHU

Date of Submission: 10th July 2019.

Submitted to : Fox Domotics Pvt. Ltd

Under the Guidance of : Junaid Khateeb

(Director, Khateeb Insitute of Technical Education)

Acknowledgements :

We would like to express my sincere gratitude to our professor Junaid Khateeb for teaching us essential concepts in machine learning and GUI using python programming language. We are highly obliged by their invaluable guidance and suggestions throughout the course of the project.

Table Of Contents

Contents **Page No.**

A. System Requirement Specifications

The aim of this project aims to analyse the two datasets containing the detailed data about various applications available on Google Play Store. Based on data like application category, size, price, number of installs, content rating, review count , reviews, a graphical representation has been made to predict the parameters that can be useful for launching a successful android application on the Google Play Store.

For designing a successful android application, it is mandatory to identify the parameters and trends that makes an application successful on the playstore. This project helps one know how well their application will work on Google Play Store based on features of the application and what improvements can be made. It will also help developers in improving existing applications to achieve higher customer satisfaction levels and better reviews and ratings on Play Store.

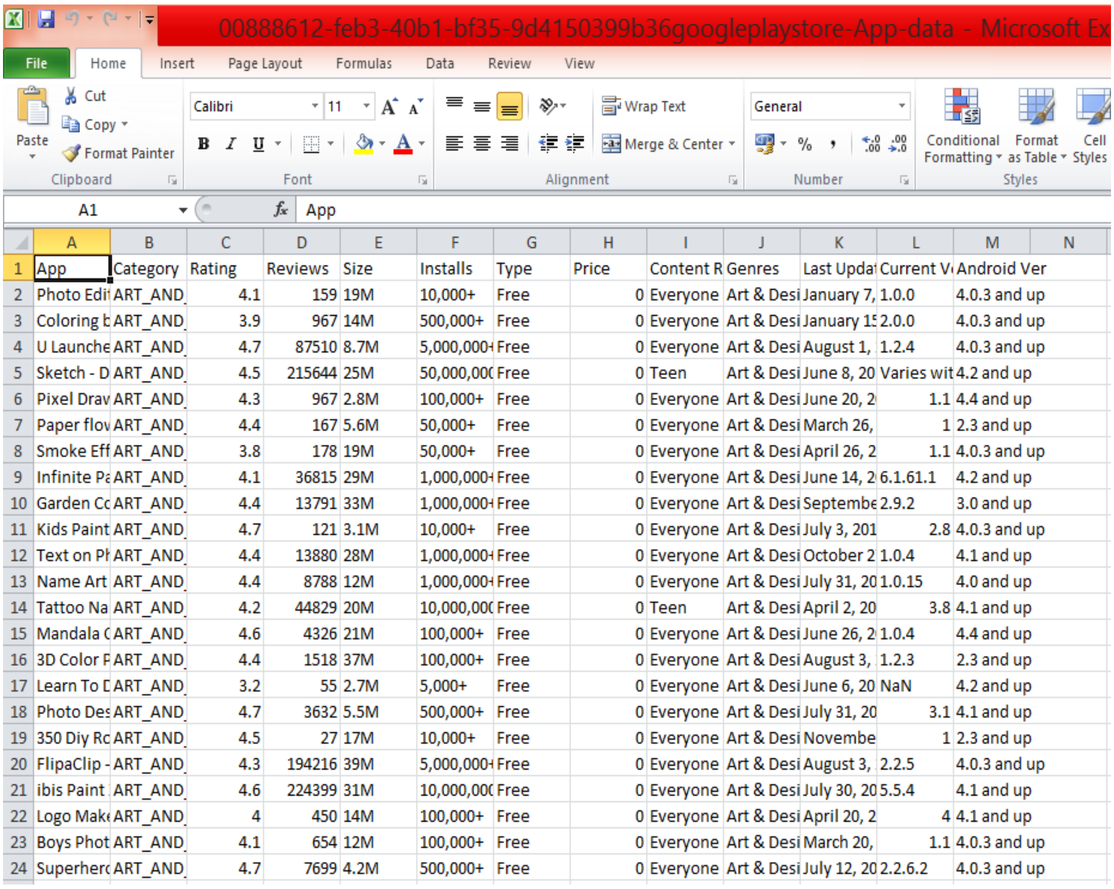
B. Technology used

- ➔ For implementing this project, we used Anaconda Navigator IDE and the programming language used was python.
- ➔ The data analysis and graphical representation was done using python libraries such as Pandas, NumPy, Matplotlib and Seaborn.
- ➔ The Graphical User Interface was implemented using Tkinter library in Python

C. Data Provided By Client

The client has provided two datasets for analysing the App data.

1. Playstore Apps Category wise segregated dataset



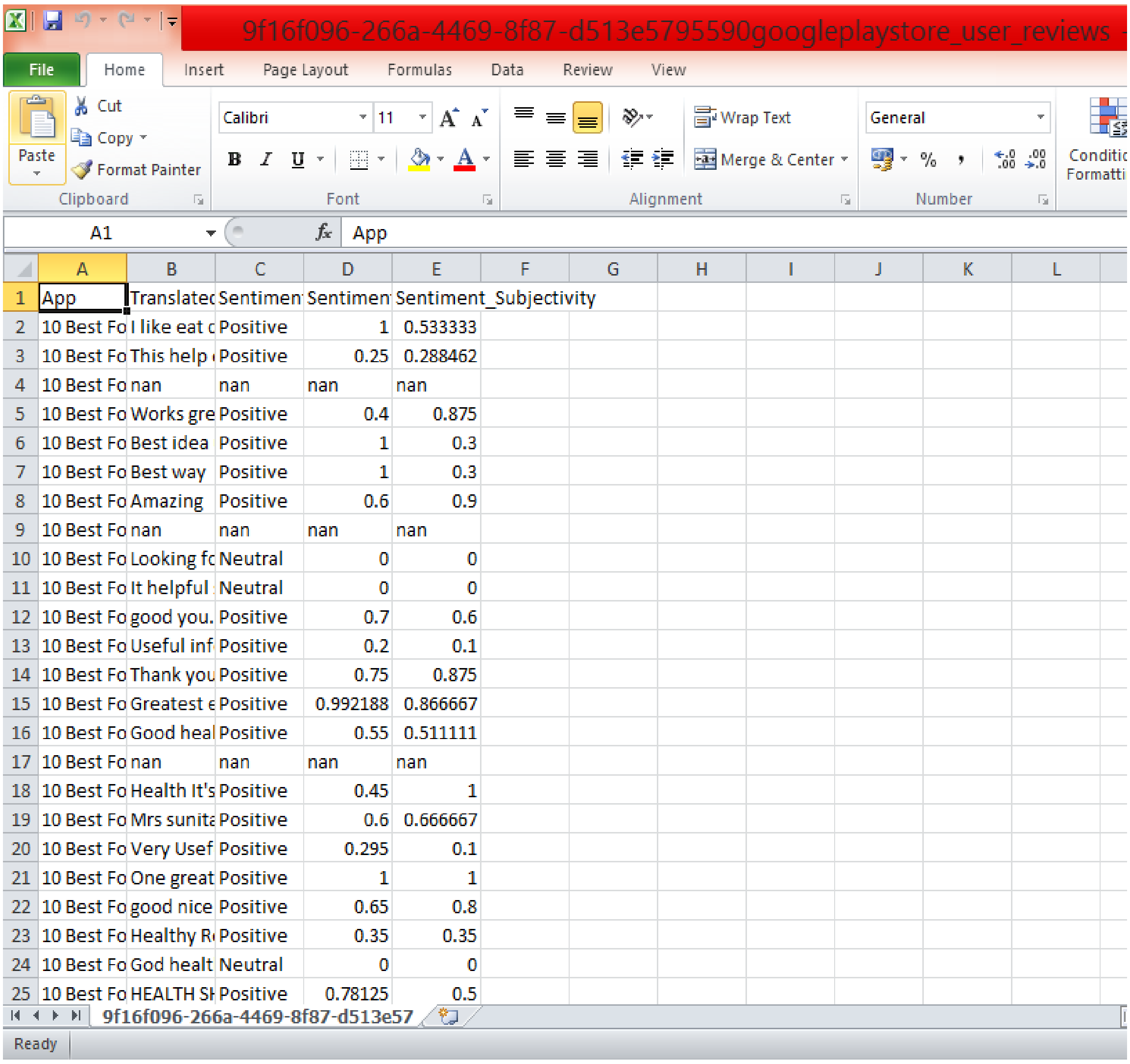
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content R	Genres	Last Updat	Current V	Android Ver	
2	Photo Edi	ART_AND_	4.1	159	19M	10,000+	Free	0	Everyone	Art & Desi	January 7,	1.0.0	4.0.3 and up	
3	Coloring b	ART_AND_	3.9	967	14M	500,000+	Free	0	Everyone	Art & Desi	January 15	2.0.0	4.0.3 and up	
4	U Launche	ART_AND_	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Desi	August 1,	1.2.4	4.0.3 and up	
5	Sketch - D	ART_AND_	4.5	215644	25M	50,000,000	Free	0	Teen	Art & Desi	June 8, 20	Varies wit	4.2 and up	
6	Pixel Drav	ART_AND_	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Desi	June 20, 2	1.1	4.4 and up	
7	Paper flov	ART_AND_	4.4	167	5.6M	50,000+	Free	0	Everyone	Art & Desi	March 26,	1	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	1,000,000+	Free	0	Everyone	Art & Desi	June 14, 2	6.1.61.1	4.2 and up	
10	Garden Cc	ART_AND_	4.4	13791	33M	1,000,000+	Free	0	Everyone	Art & Desi	Septembe	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 Pł	ART_AND_	4.4	13880	28M	1,000,000+	Free	0	Everyone	Art & Desi	October 2	1.0.4	4.1 and up	
13	Name Art	ART_AND_	4.4	8788	12M	1,000,000+	Free	0	Everyone	Art & Desi	July 31, 20	1.0.15	4.0 and up	
14	Tattoo Na	ART_AND_	4.2	44829	20M	10,000,000	Free	0	Teen	Art & Desi	April 2, 20	3.8	4.1 and up	
15	Mandala (	ART_AND_	4.6	4326	21M	100,000+	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	100,000+	Free	0	Everyone	Art & Desi	August 3,	1.2.3	2.3 and up	
17	Learn To D	ART_AND_	3.2	55	2.7M	5,000+	Free	0	Everyone	Art & Desi	June 6, 20	NaN	4.2 and up	
18	Photo Des	ART_AND_	4.7	3632	5.5M	500,000+	Free	0	Everyone	Art & Desi	July 31, 20	3.1	4.1 and up	
19	350 Diy Rc	ART_AND_	4.5	27	17M	10,000+	Free	0	Everyone	Art & Desi	Novembe	1	2.3 and up	
20	FlipaClip -	ART_AND_	4.3	194216	39M	5,000,000+	Free	0	Everyone	Art & Desi	August 3,	2.2.5	4.0.3 and up	
21	ibis Paint	ART_AND_	4.6	224399	31M	10,000,000	Free	0	Everyone	Art & Desi	July 30, 20	5.5.4	4.1 and up	
22	Logo Make	ART_AND_	4	450	14M	100,000+	Free	0	Everyone	Art & Desi	April 20, 2	4	4.1 and up	
23	Boys Phot	ART_AND_	4.1	654	12M	100,000+	Free	0	Everyone	Art & Desi	March 20,	1.1	4.0.3 and up	
24	Superherc	ART_AND_	4.7	7699	4.2M	500,000+	Free	0	Everyone	Art & Desi	July 12, 20	2.2.6.2	4.0.3 and up	

Columns in this first dataset are explained below-

- App ------------------------Application name
- Category------------------Category the app belongs to
- Rating---------------------Overall user rating of the app
- Reviews-------------------Number of user reviews for the app
- Size-------------------------Size of the app
- Installs--------------------Number of user downloads/installs for the app
- Type-----------------------Paid or Free
- Price-----------------------Price of the app
- Content Rating-------- Age group app targeted at - Children / Mature 17+ / Adult

- Genres-------------------An app can belong to multiple genres
- Last Updated-----------Date when the app was last updated on Play Store
- Current Ver-------------Current version of the app available on Play Store
- Android Ver------------Min required Android version

2. Google Playstore User Reviews dataset



	A	B	C	D	E	F	G	H	I	J	K	L
1	App	Translatec	Sentimen	Sentimen	Sentiment_Subjectivity							
2	10 Best Fo	I like eat c	Positive	1	0.533333							
3	10 Best Fo	This help	Positive	0.25	0.288462							
4	10 Best Fo	nan	nan	nan	nan							
5	10 Best Fo	Works gre	Positive	0.4	0.875							
6	10 Best Fo	Best idea	Positive	1	0.3							
7	10 Best Fo	Best way	Positive	1	0.3							
8	10 Best Fo	Amazing	Positive	0.6	0.9							
9	10 Best Fo	nan	nan	nan	nan							
10	10 Best Fo	Looking fc	Neutral	0	0							
11	10 Best Fo	It helpful	Neutral	0	0							
12	10 Best Fo	good you.	Positive	0.7	0.6							
13	10 Best Fo	Useful inf	Positive	0.2	0.1							
14	10 Best Fo	Thank you	Positive	0.75	0.875							
15	10 Best Fo	Greatest e	Positive	0.992188	0.866667							
16	10 Best Fo	Good heal	Positive	0.55	0.511111							
17	10 Best Fo	nan	nan	nan	nan							
18	10 Best Fo	Health It's	Positive	0.45	1							
19	10 Best Fo	Mrs sunita	Positive	0.6	0.666667							
20	10 Best Fo	Very Usef	Positive	0.295	0.1							
21	10 Best Fo	One great	Positive	1	1							
22	10 Best Fo	good nice	Positive	0.65	0.8							
23	10 Best Fo	Healthy R	Positive	0.35	0.35							
24	10 Best Fo	God healt	Neutral	0	0							
25	10 Best Fo	HEALTH SI	Positive	0.78125	0.5							

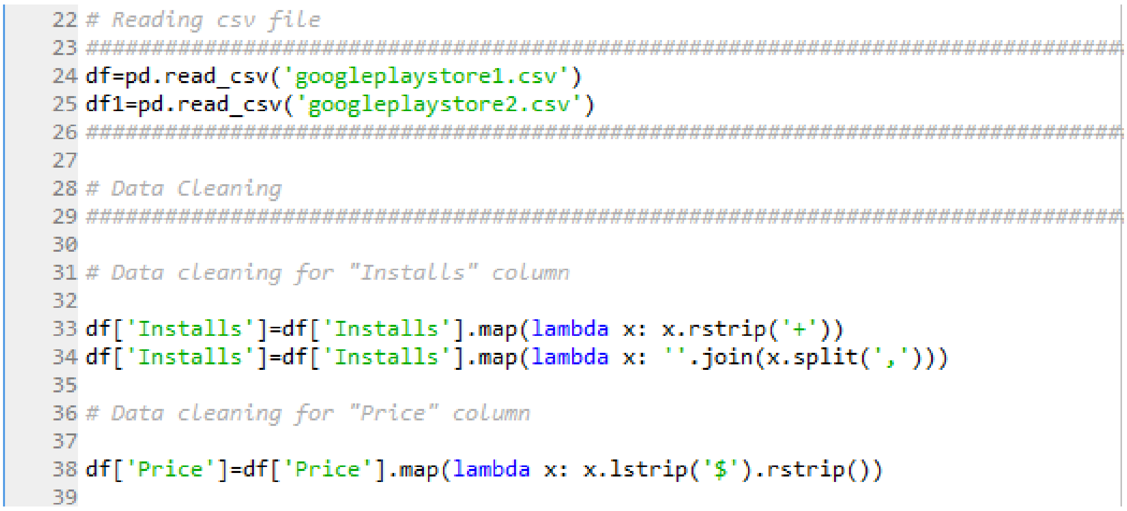
Columns in this first dataset are explained below-

- App ------------------------Application name
- Translated_Review-----User reviews for the particular app
- Sentiment----------------Opinion that is held or expressed by user for the app
- Sentiment polarity-------- lies in the range of [-1,1] where 1 means positive statement and -1 means a negative statement.
- Sentiment Subjectivity--- refers to the meaning or tone of a given reviews

D. Data Cleaning and Preparation

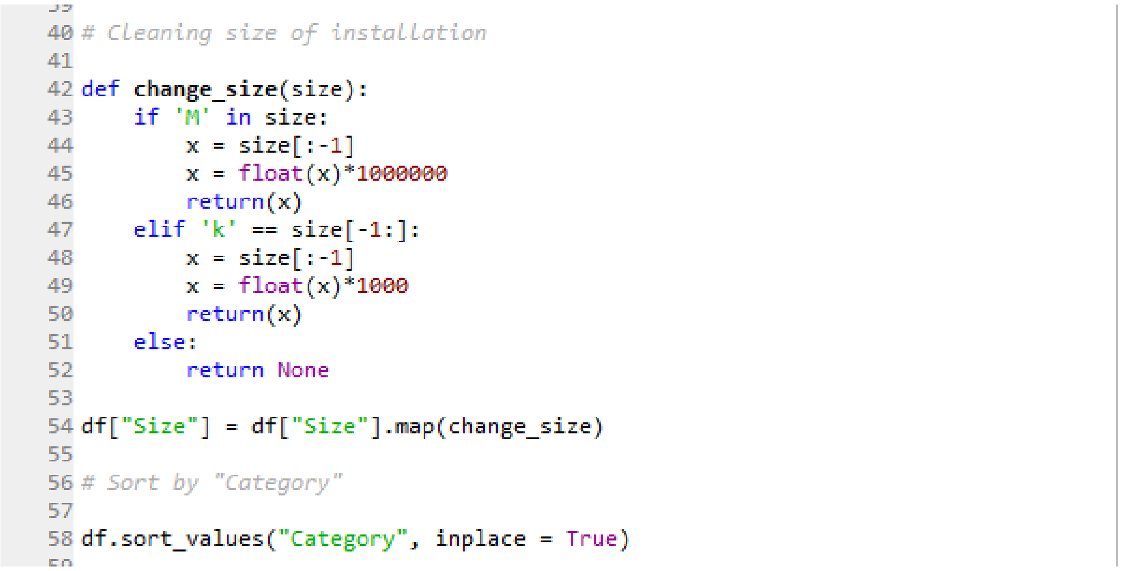
Both the Datasets contained many Nan values, null values or texts in the columns where numerical values are expected. So our first step was to remove those rows with null values in it. Also the dataset contained many information that are irrelevant in predicting the rating of app. Thus, second step is to trim the unrelated and unnecessary data in the column and convert the data in the required numerical format for data analysis.

1. Data cleaning for installs and price column



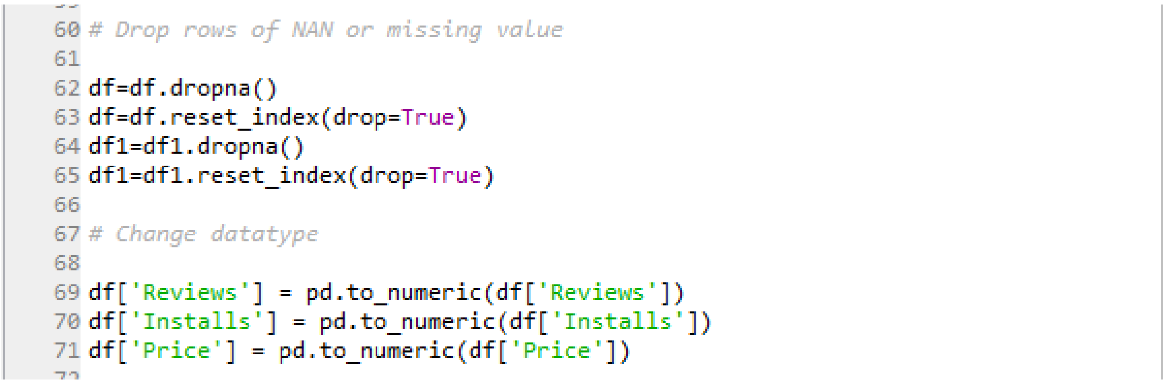
```
# Reading csv file
################################################################################
df=pd.read_csv('googleplaystore1.csv')
df1=pd.read_csv('googleplaystore2.csv')
################################################################################

# Data Cleaning
################################################################################

# Data cleaning for "Installs" column

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

# Data cleaning for "Price" column

df['Price']=df['Price'].map(lambda x: x.lstrip('$').rstrip())

```

2. Data Preparation by replacing the Million and Thousand signs in Installs



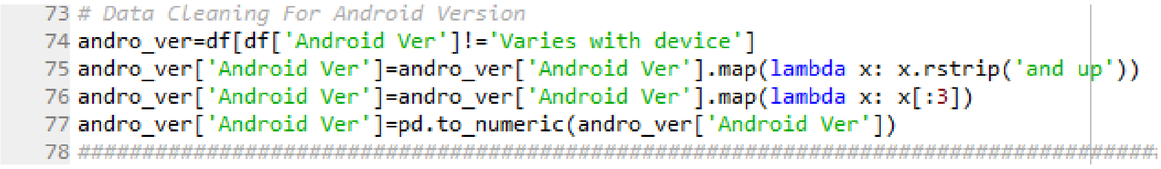
```
# Cleaning size of installation

def change_size(size):
    if 'M' in size:
        x = size[:-1]
        x = float(x)*1000000
        return(x)
    elif 'k' == size[-1:]:
        x = size[:-1]
        x = float(x)*1000
        return(x)
    else:
        return None

df["Size"] = df["Size"].map(change_size)

# Sort by "Category"

df.sort_values("Category", inplace = True)
```

3. Removing NAN values in both Datasets and Changing the datatype of required columns ‘Reviews’ ,’Installs’ and ‘Price’ to Numeric



```
# Drop rows of NAN or missing value

df=df.dropna()
df=df.reset_index(drop=True)
df1=df1.dropna()
df1=df1.reset_index(drop=True)

# Change datatype

df['Reviews'] = pd.to_numeric(df['Reviews'])
df['Installs'] = pd.to_numeric(df['Installs'])
df['Price'] = pd.to_numeric(df['Price'])
```

4. Removing the redundant strings in Android Version Column of dataset



```
# Data Cleaning For Android Version
andro_ver=df[df['Android Ver']!='Varies with device']
andro_ver['Android Ver']=andro_ver['Android Ver'].map(lambda x: x.rstrip('and up'))
andro_ver['Android Ver']=andro_ver['Android Ver'].map(lambda x: x[:3])
andro_ver['Android Ver']=pd.to_numeric(andro_ver['Android Ver'])
#####################################################################################
```

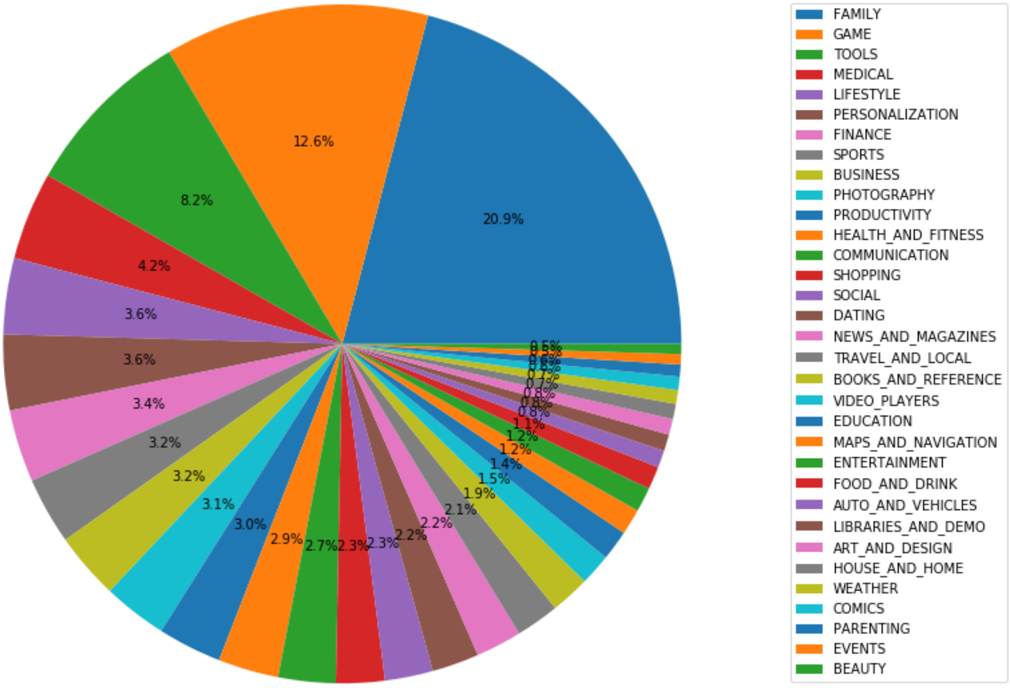
Finally , after removing absent and unrelated data in columns and making its datatype compatible for performing data analysis operations, the co-relation between different parameters of the dataset has been obtained for successful launching of an application in future.

E. Screenshots of Outputs with their codes

Followed below are the outputs of different questions as expected with the screenshots of their codes.

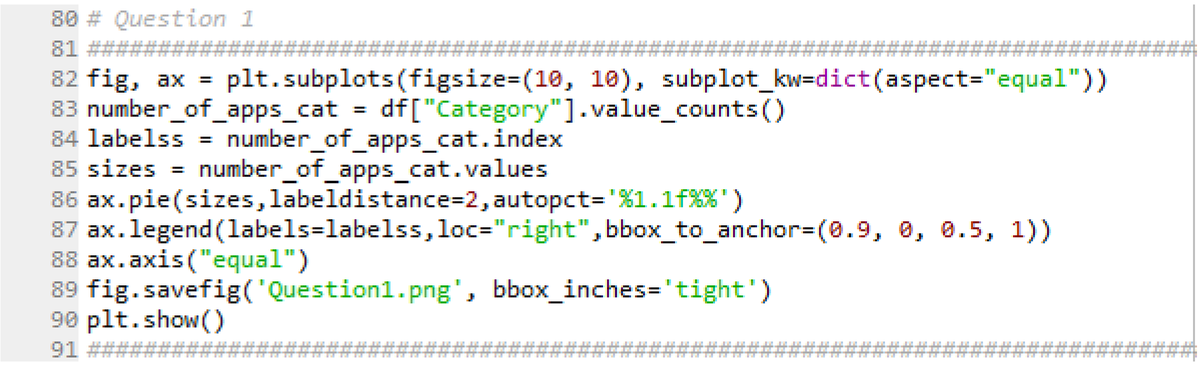
Question 1: Percentage download in each category on the playstore.

Output:



Pie Chart: Percentage Download in Each Category On Playstore

Code:



```
# Question 1
################################################################################
fig, ax = plt.subplots(figsize=(10, 10), subplot_kw=dict(aspect="equal"))
number_of_apps_cat = df["Category"].value_counts()
labelss = number_of_apps_cat.index
sizes = number_of_apps_cat.values
ax.pie(sizes,labeldistance=2,autopct='%1.1f%%')
ax.legend(labels=labelss,loc="right",bbox_to_anchor=(0.9, 0, 0.5, 1))
ax.axis("equal")
fig.savefig('Question1.png', bbox_inches='tight')
plt.show()
################################################################################
```

Question 2: Apps that have managed to get the following no. of downloads

a) Between 10,000 and 50,000

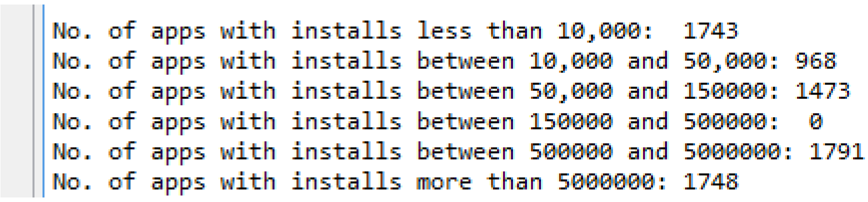
b) Between 50,000 and 150000

c) Between 150000 and 500000

d) Between 500000 and 5000000

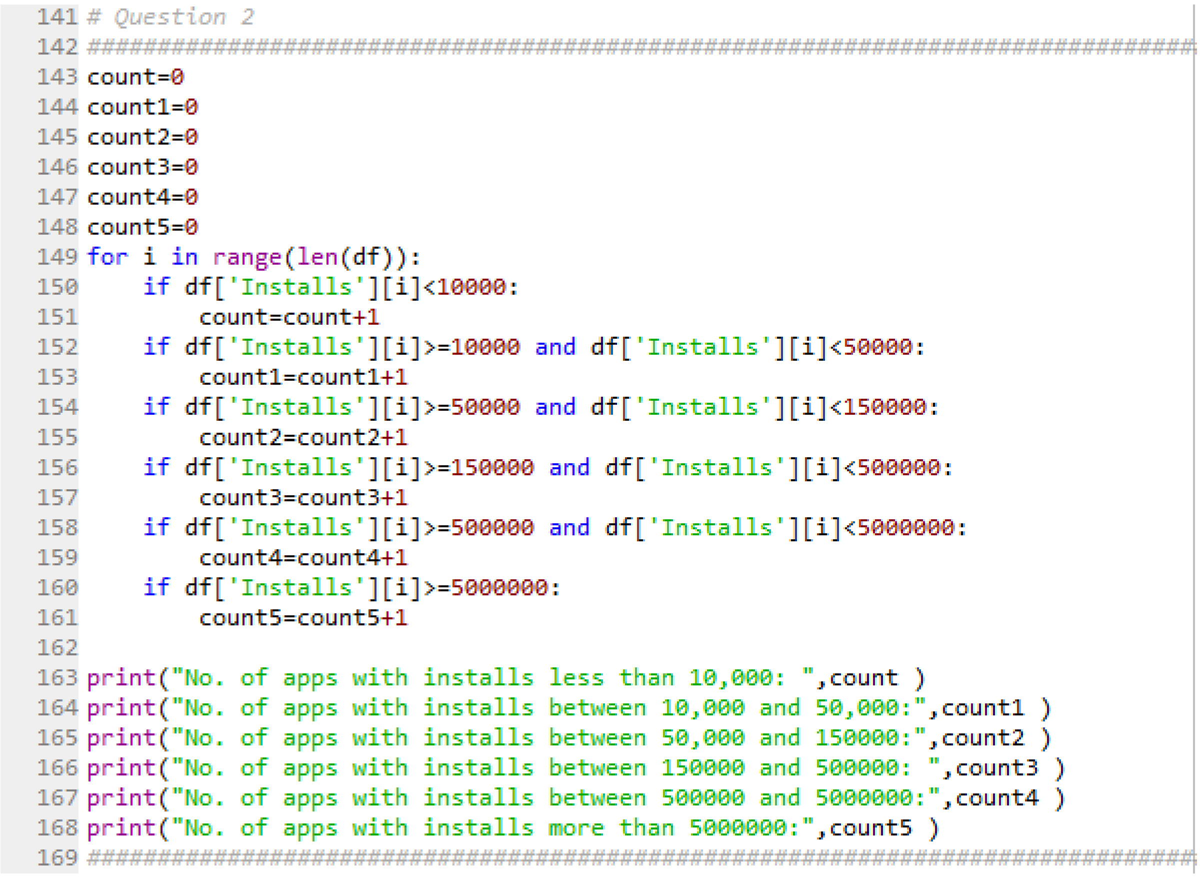
e) More than 5000000

Output:



```
No. of apps with installs less than 10,000:  1743
No. of apps with installs between 10,000 and 50,000: 968
No. of apps with installs between 50,000 and 150000: 1473
No. of apps with installs between 150000 and 500000:  0
No. of apps with installs between 500000 and 5000000: 1791
No. of apps with installs more than 5000000: 1748
```

Code:

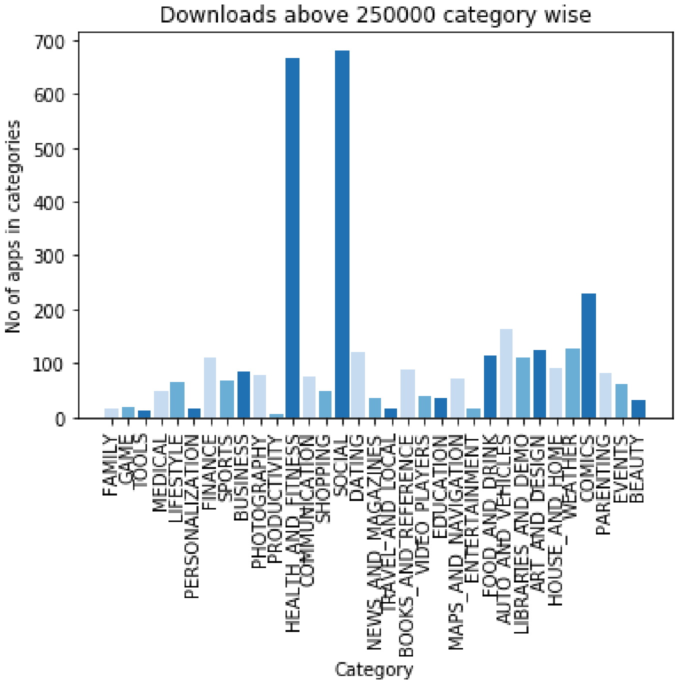


```
# Question 2
################################################################################
count=0
count1=0
count2=0
count3=0
count4=0
count5=0
for i in range(len(df)):
    if df['Installs'][i]<10000:
        count=count+1
    if df['Installs'][i]>=10000 and df['Installs'][i]<50000:
        count1=count1+1
    if df['Installs'][i]>=50000 and df['Installs'][i]<150000:
        count2=count2+1
    if df['Installs'][i]>=150000 and df['Installs'][i]<500000:
        count3=count3+1
    if df['Installs'][i]>=500000 and df['Installs'][i]<5000000:
        count4=count4+1
    if df['Installs'][i]>=5000000:
        count5=count5+1

print("No. of apps with installs less than 10,000: ",count )
print("No. of apps with installs between 10,000 and 50,000:",count1 )
print("No. of apps with installs between 50,000 and 150000:",count2 )
print("No. of apps with installs between 150000 and 500000: ",count3 )
print("No. of apps with installs between 500000 and 5000000:",count4 )
print("No. of apps with installs more than 5000000:",count5 )
################################################################################
```

Question 3: Category of apps have managed to get the most,least and an average of 2,50,000 downloads atleast.

Output:

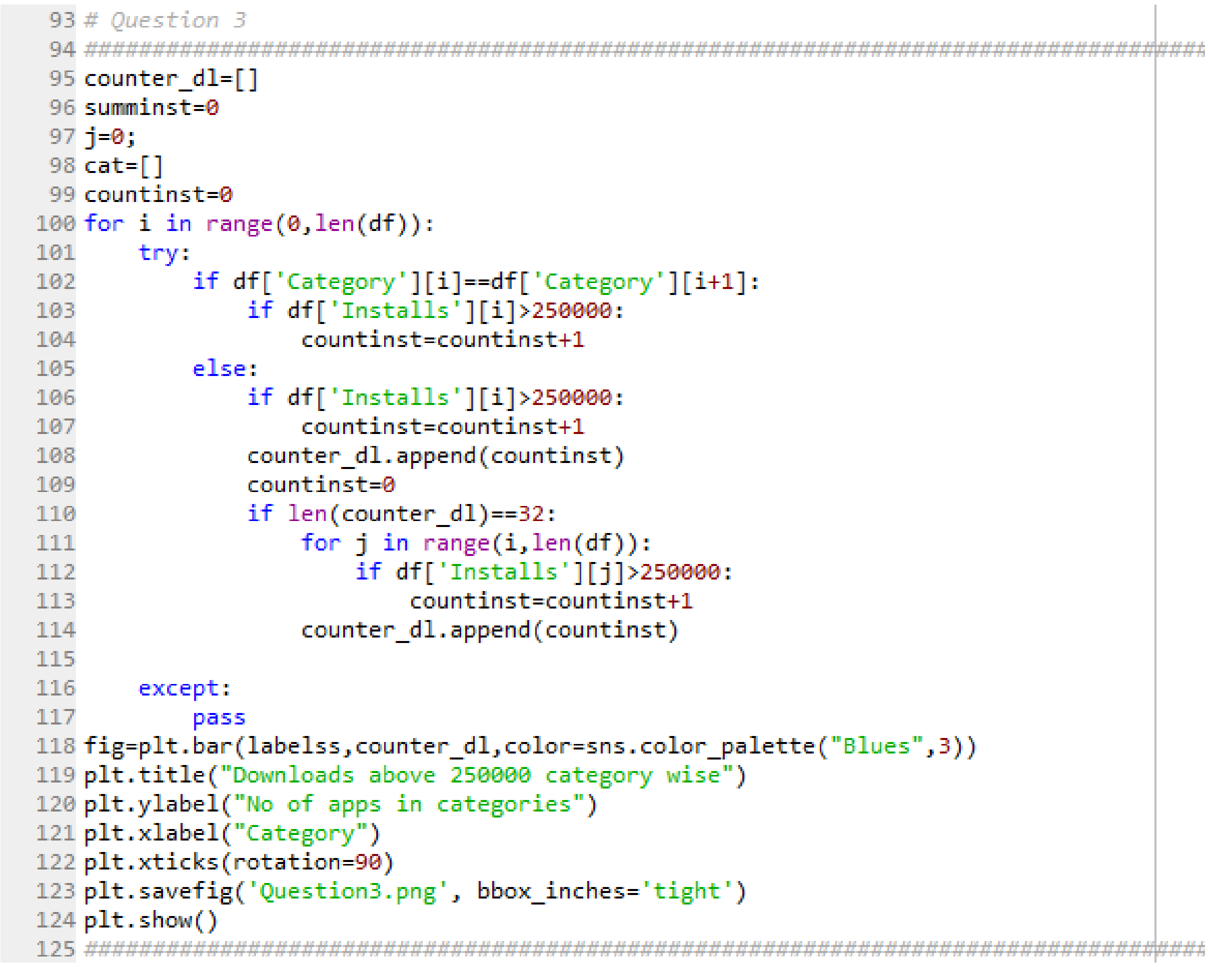
Bar Graph: No. of Apps in Categories with 250,000+ Downloads



--> Social Category has the Most No. Apps with 250,000 downloads
-->Productivity Category has the Least No. Apps with 250,000 downloads

---

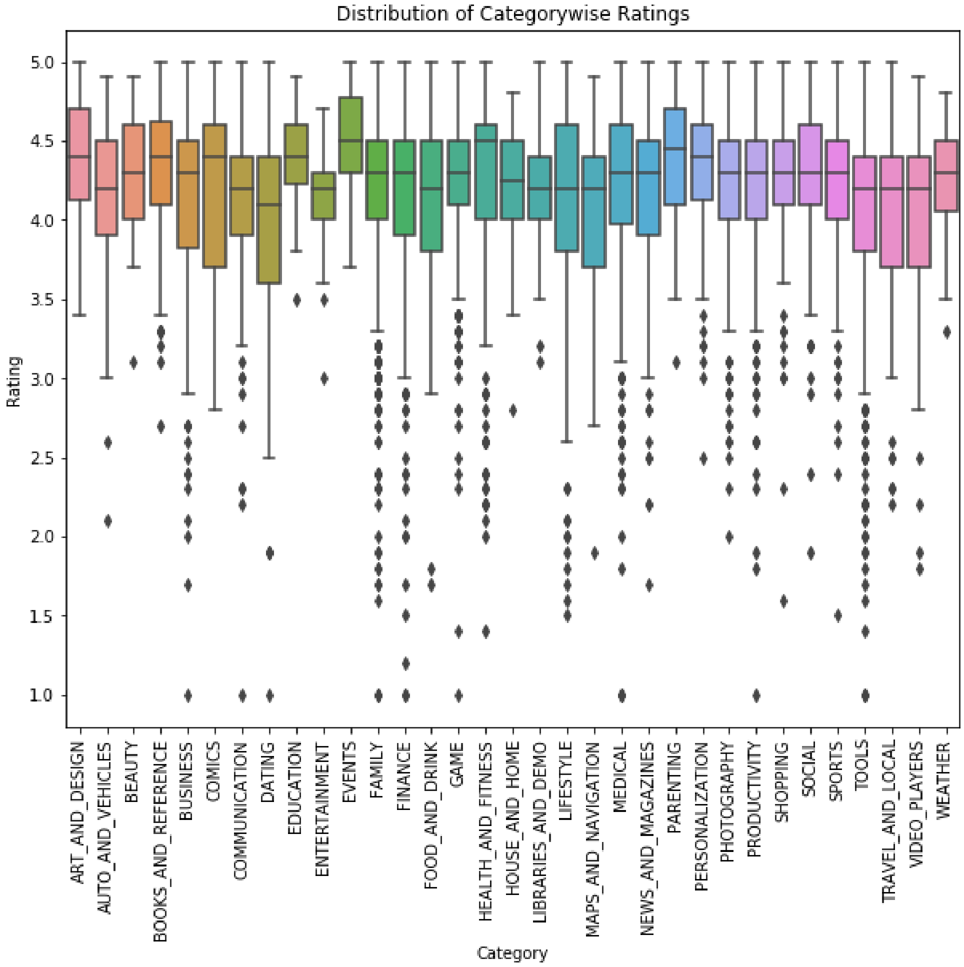
Code:



```
# Question 3
####################################################################################
counter_dl=[]
summinst=0
j=0;
cat=[]
countinst=0
for i in range(0,len(df)):
    try:
        if df['Category'][i]==df['Category'][i+1]:
            if df['Installs'][i]>250000:
                countinst=countinst+1
        else:
            if df['Installs'][i]>250000:
                countinst=countinst+1
            counter_dl.append(countinst)
            countinst=0
            if len(counter_dl)==32:
                for j in range(i,len(df)):
                    if df['Installs'][j]>250000:
                        countinst=countinst+1
                counter_dl.append(countinst)

    except:
        pass
fig=plt.bar(labelss,counter_dl,color=sns.color_palette("Blues",3))
plt.title("Downloads above 250000 category wise")
plt.ylabel("No of apps in categories")
plt.xlabel("Category")
plt.xticks(rotation=90)
plt.savefig('Question3.png', bbox_inches='tight')
plt.show()
####################################################################################
```

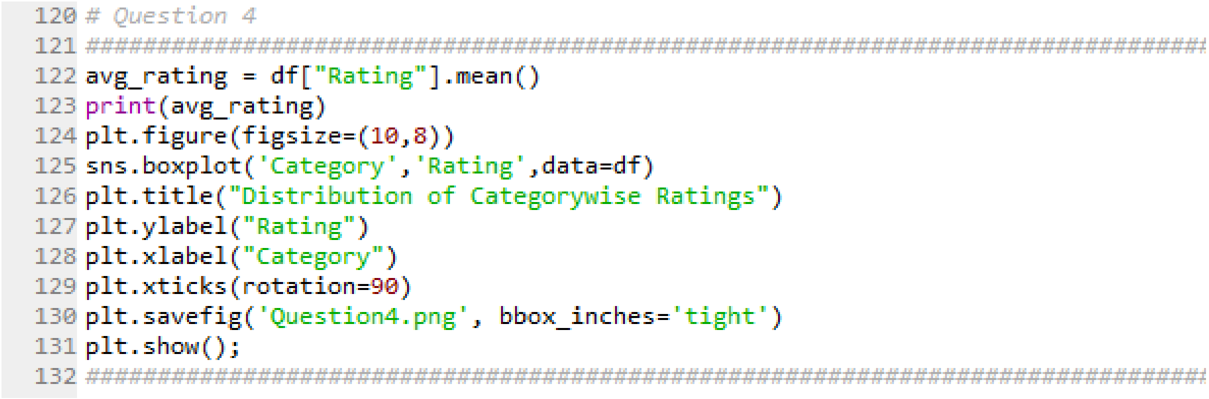
Question 4: Which category of apps have managed to get the highest maximum average ratings from the users.

Output:



--> Events Category Apps has highest maximum average ratings from users

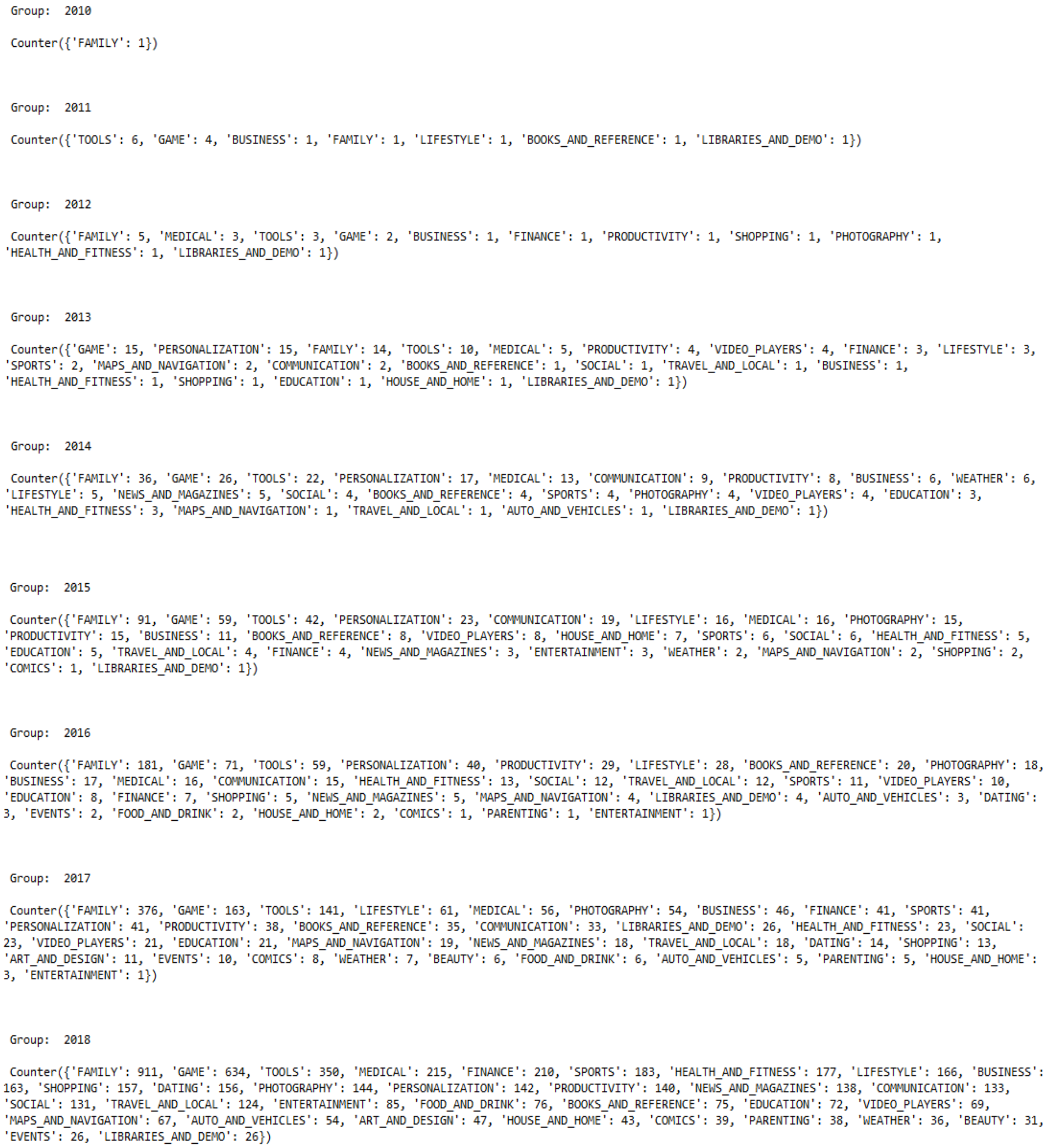
Code:



```
# Question 4
################################################################################
avg_rating = df["Rating"].mean()
print(avg_rating)
plt.figure(figsize=(10,8))
sns.boxplot('Category','Rating',data=df)
plt.title("Distribution of Categorywise Ratings")
plt.ylabel("Rating")
plt.xlabel("Category")
plt.xticks(rotation=90)
plt.savefig('Question4.png', bbox_inches='tight')
plt.show();
################################################################################
```

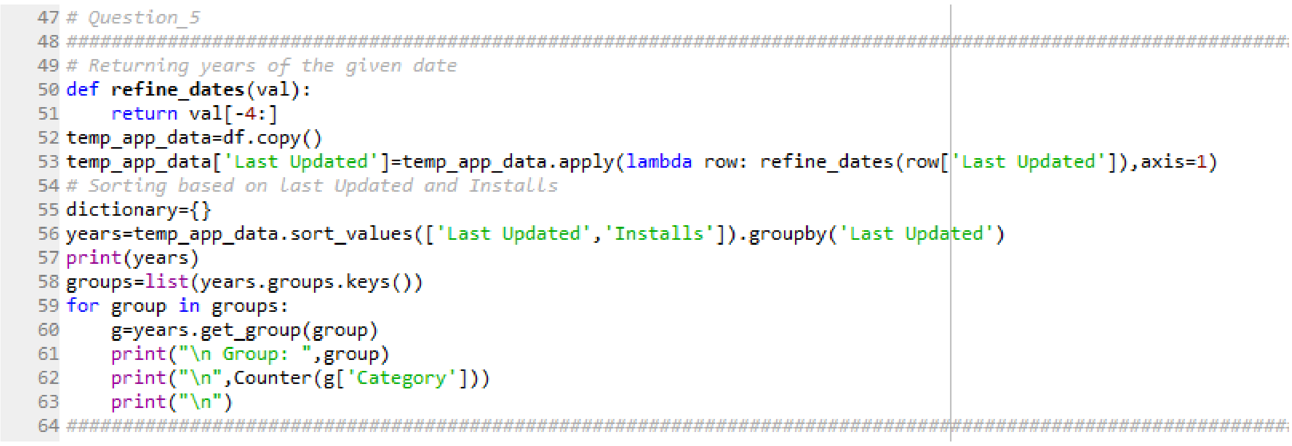
Question 5: Download trend category wise over the period for which the data is being made available.

Output:



```
 Group:  2010

 Counter({'FAMILY': 1})


 Group:  2011

 Counter({'TOOLS': 6, 'GAME': 4, 'BUSINESS': 1, 'FAMILY': 1, 'LIFESTYLE': 1, 'BOOKS_AND_REFERENCE': 1, 'LIBRARIES_AND_DEMO': 1})


 Group:  2012

 Counter({'FAMILY': 5, 'MEDICAL': 3, 'TOOLS': 3, 'GAME': 2, 'BUSINESS': 1, 'FINANCE': 1, 'PRODUCTIVITY': 1, 'SHOPPING': 1, 'PHOTOGRAPHY': 1,
'HEALTH_AND_FITNESS': 1, 'LIBRARIES_AND_DEMO': 1})


 Group:  2013

 Counter({'GAME': 15, 'PERSONALIZATION': 15, 'FAMILY': 14, 'TOOLS': 10, 'MEDICAL': 5, 'PRODUCTIVITY': 4, 'VIDEO_PLAYERS': 4, 'FINANCE': 3, 'LIFESTYLE': 3,
'SPORTS': 2, 'MAPS_AND_NAVIGATION': 2, 'COMMUNICATION': 2, 'BOOKS_AND_REFERENCE': 1, 'SOCIAL': 1, 'TRAVEL_AND_LOCAL': 1, 'BUSINESS': 1,
'HEALTH_AND_FITNESS': 1, 'SHOPPING': 1, 'EDUCATION': 1, 'HOUSE_AND_HOME': 1, 'LIBRARIES_AND_DEMO': 1})


 Group:  2014

 Counter({'FAMILY': 36, 'GAME': 26, 'TOOLS': 22, 'PERSONALIZATION': 17, 'MEDICAL': 13, 'COMMUNICATION': 9, 'PRODUCTIVITY': 8, 'BUSINESS': 6, 'WEATHER': 6,
'LIFESTYLE': 5, 'NEWS_AND_MAGAZINES': 5, 'SOCIAL': 4, 'BOOKS_AND_REFERENCE': 4, 'SPORTS': 4, 'PHOTOGRAPHY': 4, 'VIDEO_PLAYERS': 4, 'EDUCATION': 3,
'HEALTH_AND_FITNESS': 3, 'MAPS_AND_NAVIGATION': 1, 'TRAVEL_AND_LOCAL': 1, 'AUTO_AND_VEHICLES': 1, 'LIBRARIES_AND_DEMO': 1})


 Group:  2015

 Counter({'FAMILY': 91, 'GAME': 59, 'TOOLS': 42, 'PERSONALIZATION': 23, 'COMMUNICATION': 19, 'LIFESTYLE': 16, 'MEDICAL': 16, 'PHOTOGRAPHY': 15,
'PRODUCTIVITY': 15, 'BUSINESS': 11, 'BOOKS_AND_REFERENCE': 8, 'VIDEO_PLAYERS': 8, 'HOUSE_AND_HOME': 7, 'SPORTS': 6, 'SOCIAL': 6, 'HEALTH_AND_FITNESS': 5,
'EDUCATION': 5, 'TRAVEL_AND_LOCAL': 4, 'FINANCE': 4, 'NEWS_AND_MAGAZINES': 3, 'ENTERTAINMENT': 3, 'WEATHER': 2, 'MAPS_AND_NAVIGATION': 2, 'SHOPPING': 2,
'COMICS': 1, 'LIBRARIES_AND_DEMO': 1})


 Group:  2016

 Counter({'FAMILY': 181, 'GAME': 71, 'TOOLS': 59, 'PERSONALIZATION': 40, 'PRODUCTIVITY': 29, 'LIFESTYLE': 28, 'BOOKS_AND_REFERENCE': 20, 'PHOTOGRAPHY': 18,
'BUSINESS': 17, 'MEDICAL': 16, 'COMMUNICATION': 15, 'HEALTH_AND_FITNESS': 13, 'SOCIAL': 12, 'TRAVEL_AND_LOCAL': 12, 'SPORTS': 11, 'VIDEO_PLAYERS': 10,
'EDUCATION': 8, 'FINANCE': 7, 'SHOPPING': 5, 'NEWS_AND_MAGAZINES': 5, 'MAPS_AND_NAVIGATION': 4, 'LIBRARIES_AND_DEMO': 4, 'AUTO_AND_VEHICLES': 3, 'DATING':
3, 'EVENTS': 2, 'FOOD_AND_DRINK': 2, 'HOUSE_AND_HOME': 2, 'COMICS': 1, 'PARENTING': 1, 'ENTERTAINMENT': 1})


 Group:  2017

 Counter({'FAMILY': 376, 'GAME': 163, 'TOOLS': 141, 'LIFESTYLE': 61, 'MEDICAL': 56, 'PHOTOGRAPHY': 54, 'BUSINESS': 46, 'FINANCE': 41, 'SPORTS': 41,
'PERSONALIZATION': 41, 'PRODUCTIVITY': 38, 'BOOKS_AND_REFERENCE': 35, 'COMMUNICATION': 33, 'LIBRARIES_AND_DEMO': 26, 'HEALTH_AND_FITNESS': 23, 'SOCIAL':
23, 'VIDEO_PLAYERS': 21, 'EDUCATION': 21, 'MAPS_AND_NAVIGATION': 19, 'NEWS_AND_MAGAZINES': 18, 'TRAVEL_AND_LOCAL': 18, 'DATING': 14, 'SHOPPING': 13,
'ART_AND_DESIGN': 11, 'EVENTS': 10, 'COMICS': 8, 'WEATHER': 7, 'BEAUTY': 6, 'FOOD_AND_DRINK': 6, 'AUTO_AND_VEHICLES': 5, 'PARENTING': 5, 'HOUSE_AND_HOME':
3, 'ENTERTAINMENT': 1})


 Group:  2018

 Counter({'FAMILY': 911, 'GAME': 634, 'TOOLS': 350, 'MEDICAL': 215, 'FINANCE': 210, 'SPORTS': 183, 'HEALTH_AND_FITNESS': 177, 'LIFESTYLE': 166, 'BUSINESS':
163, 'SHOPPING': 157, 'DATING': 156, 'PHOTOGRAPHY': 144, 'PERSONALIZATION': 142, 'PRODUCTIVITY': 140, 'NEWS_AND_MAGAZINES': 138, 'COMMUNICATION': 133,
'SOCIAL': 131, 'TRAVEL_AND_LOCAL': 124, 'ENTERTAINMENT': 85, 'FOOD_AND_DRINK': 76, 'BOOKS_AND_REFERENCE': 75, 'EDUCATION': 72, 'VIDEO_PLAYERS': 69,
'MAPS_AND_NAVIGATION': 67, 'AUTO_AND_VEHICLES': 54, 'ART_AND_DESIGN': 47, 'HOUSE_AND_HOME': 43, 'COMICS': 39, 'PARENTING': 38, 'WEATHER': 36, 'BEAUTY': 31,
'EVENTS': 26, 'LIBRARIES_AND_DEMO': 26})
```

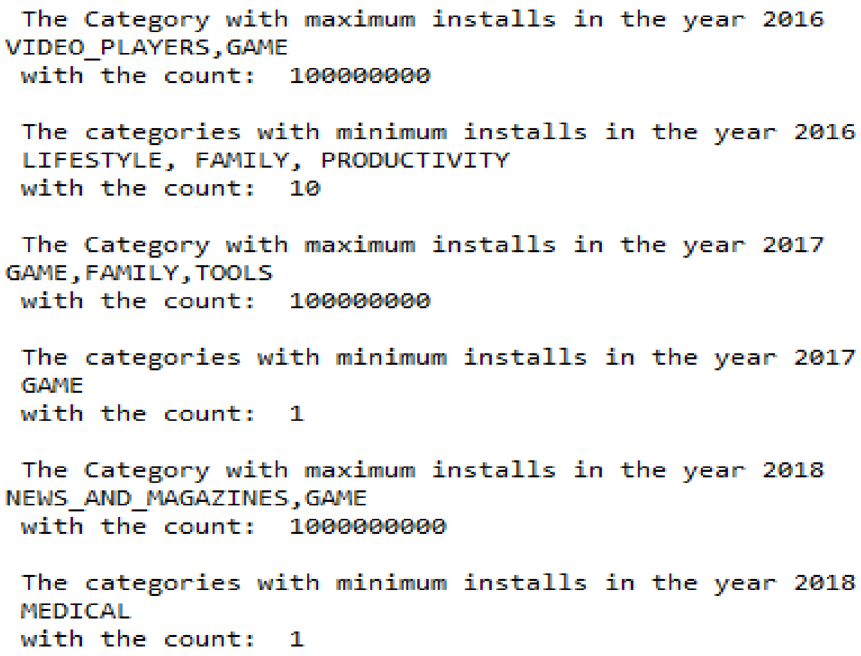
Code:



```
# Question_5
#############################################################################################################
# Returning years of the given date
def refine_dates(val):
    return val[-4:]
temp_app_data=df.copy()
temp_app_data['Last Updated']=temp_app_data.apply(lambda row: refine_dates(row['Last Updated']),axis=1)
# Sorting based on last Updated and Installs
dictionary={}
years=temp_app_data.sort_values(['Last Updated','Installs']).groupby('Last Updated')
print(years)
groups=list(years.groups.keys())
for group in groups:
    g=years.get_group(group)
    print("\n Group: ",group)
    print("\n",Counter(g['Category']))
    print("\n")
#############################################################################################################
```

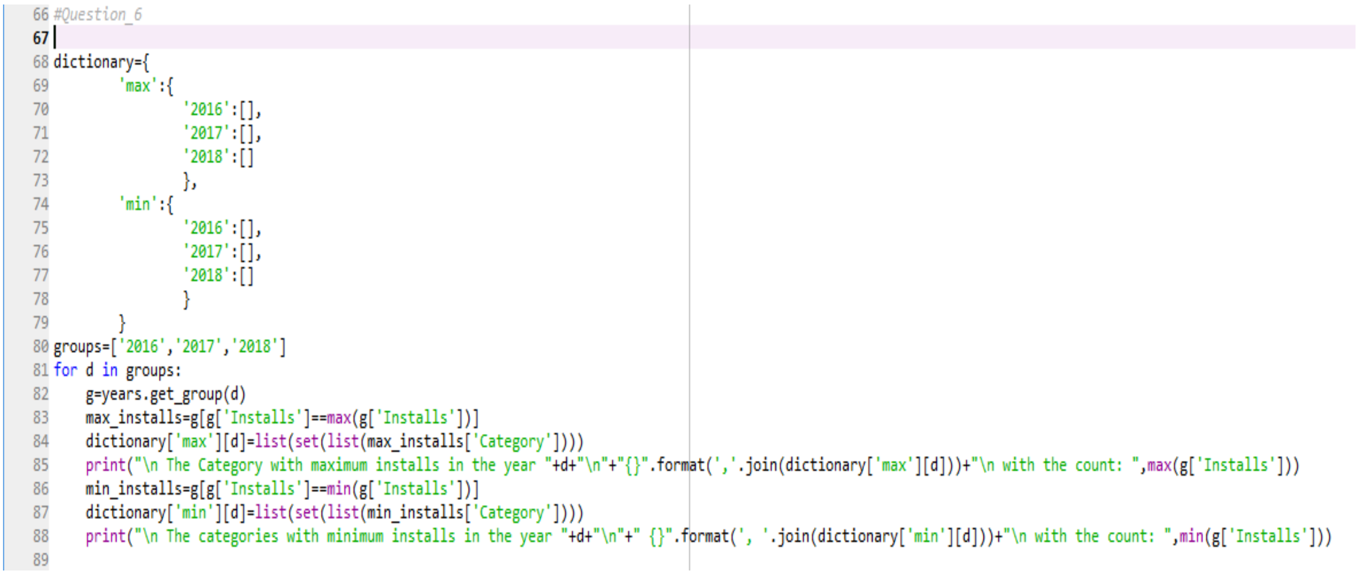
<u>Question 6:</u> For the years 2016,2017, 2018 what are the category of apps that have got the most and the least downloads. What is the percentage increase or decrease that the apps have got over the period of three years.

Output:



```
 The Category with maximum installs in the year 2016
VIDEO_PLAYERS,GAME
 with the count:  100000000

 The categories with minimum installs in the year 2016
 LIFESTYLE, FAMILY, PRODUCTIVITY
 with the count:  10

 The Category with maximum installs in the year 2017
GAME,FAMILY,TOOLS
 with the count:  100000000

 The categories with minimum installs in the year 2017
 GAME
 with the count:  1

 The Category with maximum installs in the year 2018
NEWS_AND_MAGAZINES,GAME
 with the count:  1000000000

 The categories with minimum installs in the year 2018
 MEDICAL
 with the count:  1
```

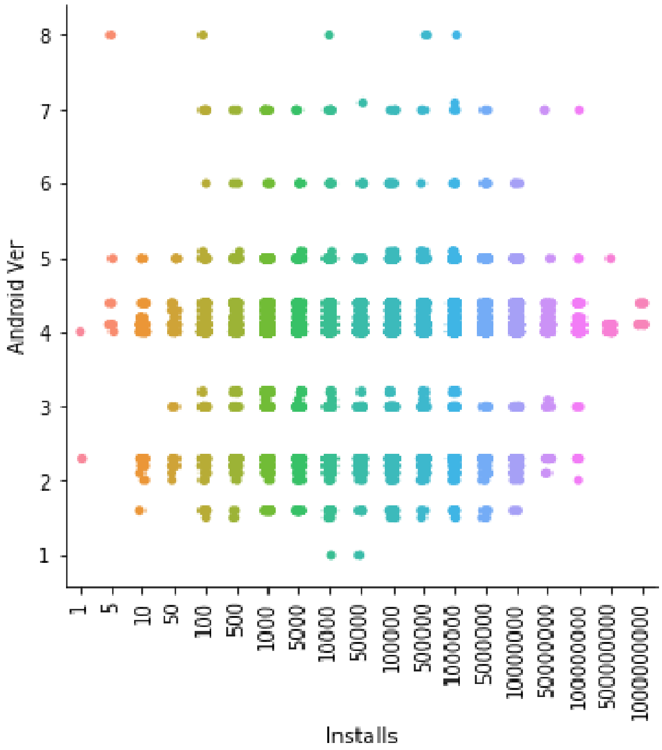
Code:



```
#Question_6

dictionary={
        'max':{
                '2016':[],
                '2017':[],
                '2018':[]
                },
        'min':{
                '2016':[],
                '2017':[],
                '2018':[]
                }
        }
groups=['2016','2017','2018']
for d in groups:
    g=years.get_group(d)
    max_installs=g[g['Installs']==max(g['Installs'])]
    dictionary['max'][d]=list(set(list(max_installs['Category'])))
    print("\n The Category with maximum installs in the year "+d+"\n"+"{}".format(','.join(dictionary['max'][d]))+"\n with the count: ",max(g['Installs']))
    min_installs=g[g['Installs']==min(g['Installs'])]
    dictionary['min'][d]=list(set(list(min_installs['Category'])))
    print("\n The categories with minimum installs in the year "+d+"\n"+" {}".format(', '.join(dictionary['min'][d]))+"\n with the count: ",min(g['Installs']))

```

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

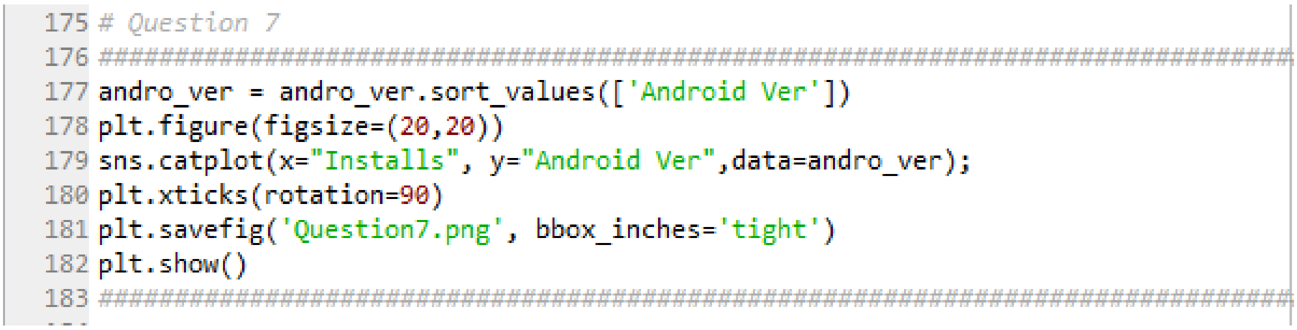
Output:



--> Initial increase and later, gradual decrease is seen in the no. of versions for Apps

--> Apps with highest & lowest no. of installs belong to android versions between 4 to 5

Code:

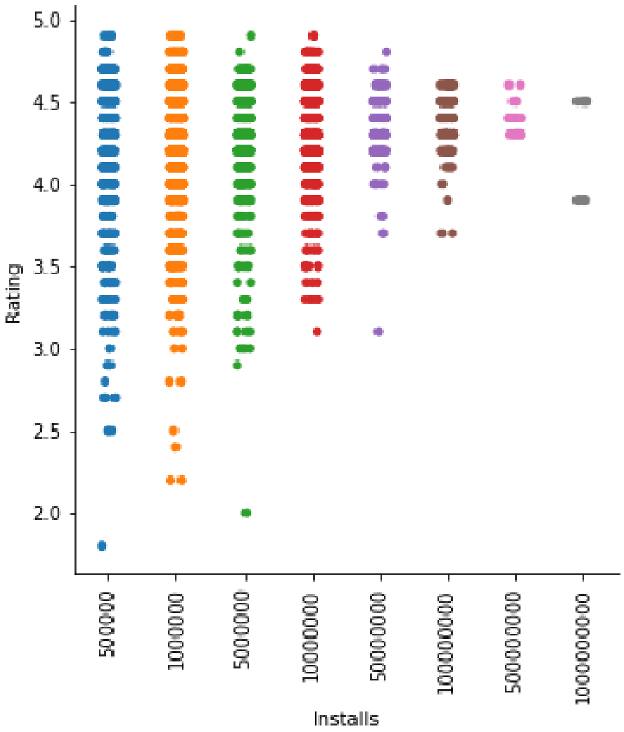


```
# Question 7
################################################################################
andro_ver = andro_ver.sort_values(['Android Ver'])
plt.figure(figsize=(20,20))
sns.catplot(x="Installs", y="Android Ver",data=andro_ver);
plt.xticks(rotation=90)
plt.savefig('Question7.png', bbox_inches='tight')
plt.show()
################################################################################
```

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

Output:

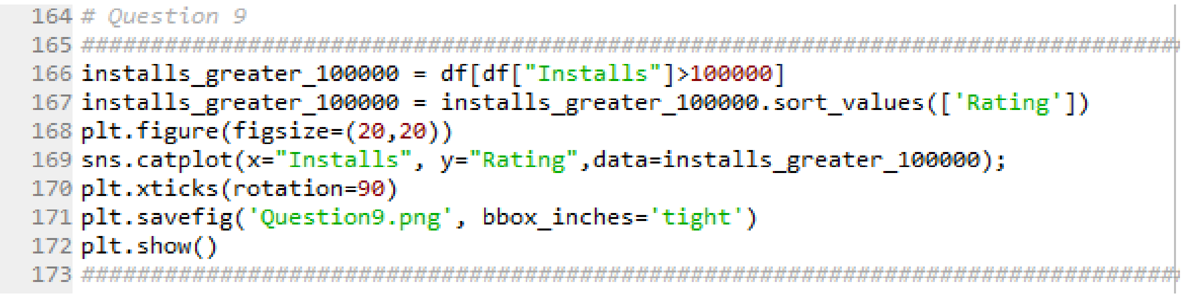
Co-relation between Ratings and No. of Downloads



--> As the No. of Downloads increases, Average ratings increases too

--> Also, from the graph, apps who have managed to get over 1,00,000 downloads, haven't managed to get average rating of 4.1

Code:

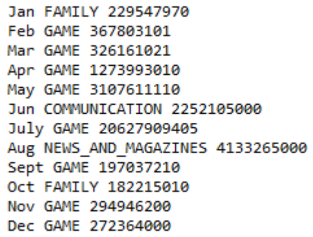


```
# Question 9
################################################################################
installs_greater_100000 = df[df["Installs"]>100000]
installs_greater_100000 = installs_greater_100000.sort_values(['Rating'])
plt.figure(figsize=(20,20))
sns.catplot(x="Installs", y="Rating",data=installs_greater_100000);
plt.xticks(rotation=90)
plt.savefig('Question9.png', bbox_inches='tight')
plt.show()
################################################################################
```

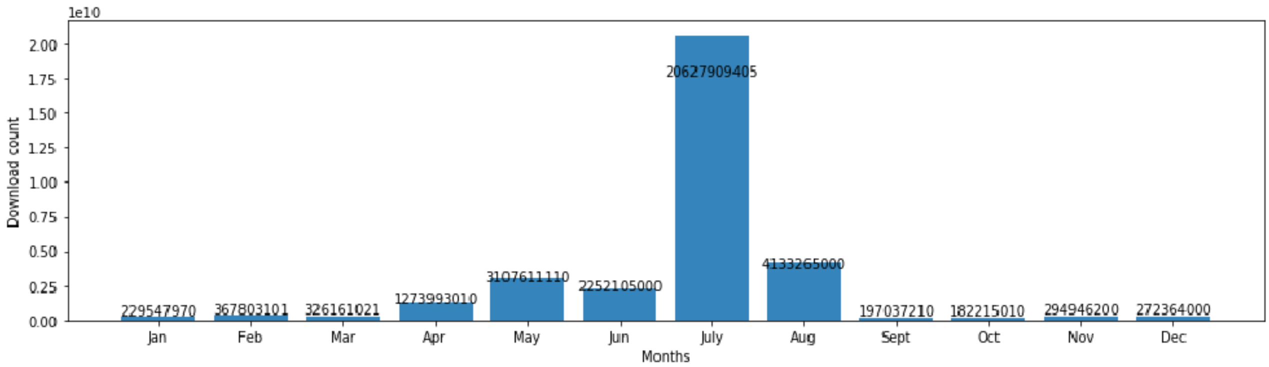
Question 10: Across all the years ,which month has seen the maximum downloads for each of the category.

Output:

Relation between Month and App Category with maximum downloads in that month

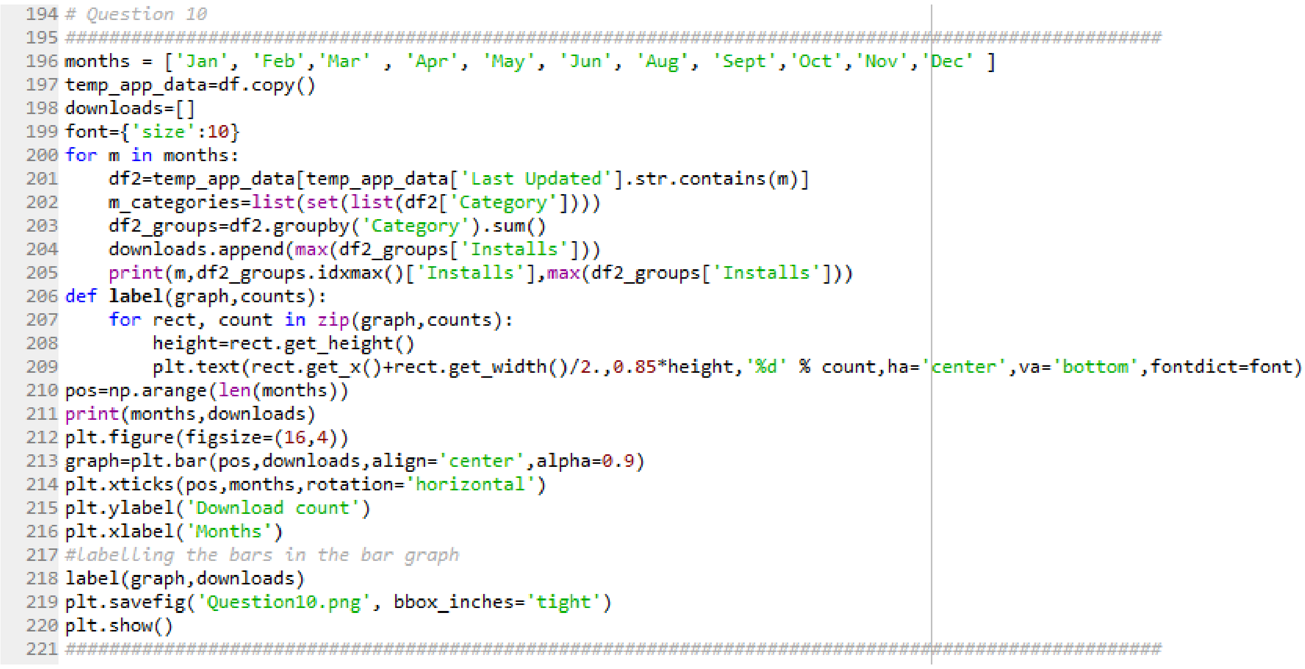


```
Jan FAMILY 229547970
Feb GAME 367803101
Mar GAME 326161021
Apr GAME 1273993010
May GAME 3107611110
Jun COMMUNICATION 2252105000
July GAME 20627909405
Aug NEWS_AND_MAGAZINES 4133265000
Sept GAME 197037210
Oct FAMILY 182215010
Nov GAME 294946200
Dec GAME 272364000
```



--> Game Category has highest number of downloads for 8 months in an year

Code:



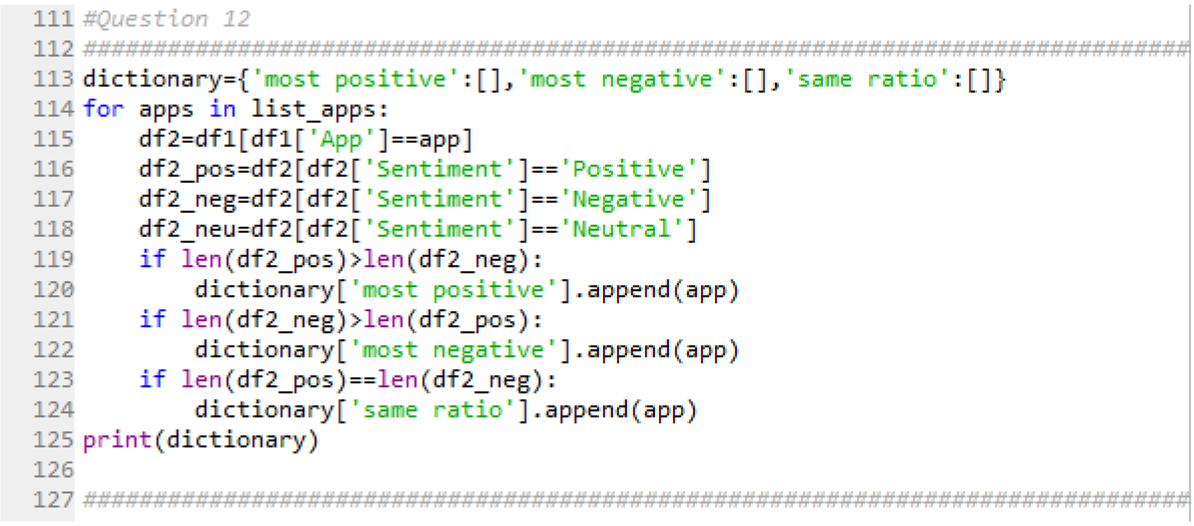
```
# Question 10
##################################################################################################
months = ['Jan', 'Feb','Mar' , 'Apr', 'May', 'Jun', 'Aug', 'Sept','Oct','Nov','Dec' ]
temp_app_data=df.copy()
downloads=[]
font={'size':10}
for m in months:
    df2=temp_app_data[temp_app_data['Last Updated'].str.contains(m)]
    m_categories=list(set(list(df2['Category'])))
    df2_groups=df2.groupby('Category').sum()
    downloads.append(max(df2_groups['Installs']))
    print(m,df2_groups.idxmax()['Installs'],max(df2_groups['Installs']))
def label(graph,counts):
    for rect, count in zip(graph,counts):
        height=rect.get_height()
        plt.text(rect.get_x()+rect.get_width()/2.,0.85*height,'%d' % count,ha='center',va='bottom',fontdict=font)
pos=np.arange(len(months))
print(months,downloads)
plt.figure(figsize=(16,4))
graph=plt.bar(pos,downloads,align='center',alpha=0.9)
plt.xticks(pos,months,rotation='horizontal')
plt.ylabel('Download count')
plt.xlabel('Months')
#labelling the bars in the bar graph
label(graph,downloads)
plt.savefig('Question10.png', bbox_inches='tight')
plt.show()
##################################################################################################
```

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

Output: → Most positive(not entire list since too long for report),negative and neutral apps lists output are displayed below

{'most positive': 'A+ Gallery - Photos & Videos', 'Anatomy Learning - 3D Atlas', 'HESI A2 Pocket Prep', 'Cymera Camera- Photo Editor, Filter,Collage,Layout', 'Fandango Movies - Times + Tickets', 'CWT To Go', 'Dictionary - Merriam-Webster', 'Educational Games for Kids', 'Goibibo - Flight Hotel Bus Car IRCTC Booking App', 'Blood Donor', 'Farm Fruit Pop: Party Time', 'Garmin Connect™', 'Dairy Queen', 'Aviary Effects: Classic', 'CallApp: Caller ID, Blocker & Phone Call Recorder', 'Easy Healthy Recipes', 'Healthy Recipes Free', 'Apex Launcher', 'ASUS Cover for ZenFone 2', 'BIG Launcher', 'Daily Yoga - Yoga Fitness Plans', 'Fresh EBT - Food Stamp Balance', 'E*TRADE Mobile', 'Barbie Life™', 'Comedy Central', 'BeWild Free Dating & Chat App', 'Google Handwriting Input', 'Asana: organize team projects', 'Files Go by Google: Free up space on your phone', 'Google Keep' 'most negative': [], 'same ratio': ['File Manager', 'Calculator - free calculator, multi calculator app', 'Common Core', 'Dashlane Free Password Manager', 'Best Car Wallpapers', 'Disney Heroes: Battle Mode', 'Barclays US for Android', 'Dog Sim Online: Raise a Family', 'Extreme Car Driving Simulator', 'Cut the Rope FULL FREE', "Davis's Drug Guide", 'Bad Piggies', 'Bualuang mBanking', 'Baseball Boy!', 'Hotstar', 'HD Camera', 'Easy Installer - Apps On SD', 'CBS News', 'Free Foreclosure Real Estate Search by USHUD.com', 'CM Browser - Ad Blocker , Fast Download , Privacy']}

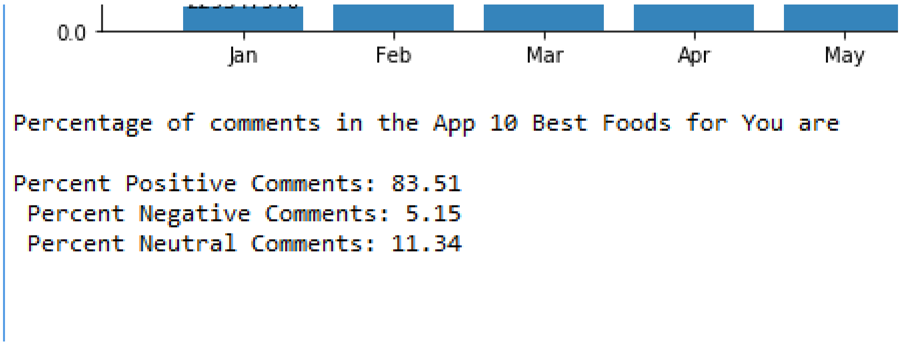
Code:



```
#Question 12
################################################################################
dictionary={'most positive':[],'most negative':[],'same ratio':[]}
for apps in list_apps:
    df2=df1[df1['App']==app]
    df2_pos=df2[df2['Sentiment']=='Positive']
    df2_neg=df2[df2['Sentiment']=='Negative']
    df2_neu=df2[df2['Sentiment']=='Neutral']
    if len(df2_pos)>len(df2_neg):
        dictionary['most positive'].append(app)
    if len(df2_neg)>len(df2_pos):
        dictionary['most negative'].append(app)
    if len(df2_pos)==len(df2_neg):
        dictionary['same ratio'].append(app)
print(dictionary)

################################################################################
```

Question 15: Is it advisable to launch an app like ’10 Best foods for you’? Do the users like these apps?

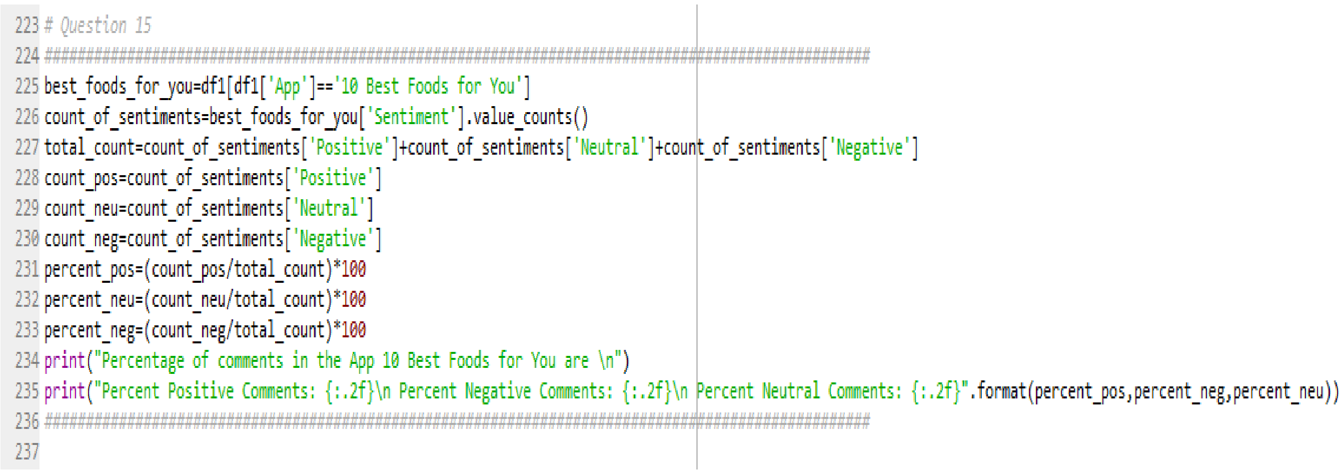
Output:



```
Percentage of comments in the App 10 Best Foods for You are

Percent Positive Comments: 83.51
 Percent Negative Comments: 5.15
 Percent Neutral Comments: 11.34
```

➔ Since majority of comments for the app ’10 Best foods for you’ are positive, it can be concluded that the users like this app

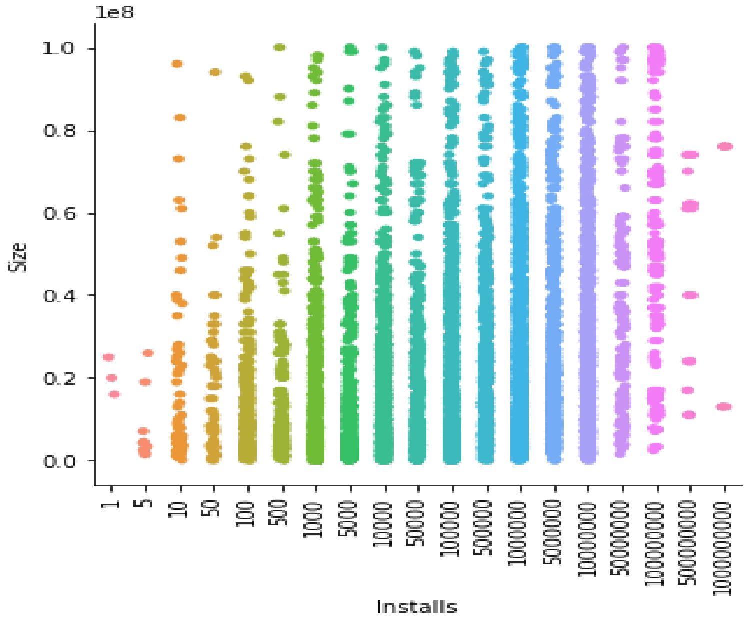
Code:



```
# Question 15
####################################################################################################
best_foods_for_you=df1[df1['App']=='10 Best Foods for You']
count_of_sentiments=best_foods_for_you['Sentiment'].value_counts()
total_count=count_of_sentiments['Positive']+count_of_sentiments['Neutral']+count_of_sentiments['Negative']
count_pos=count_of_sentiments['Positive']
count_neu=count_of_sentiments['Neutral']
count_neg=count_of_sentiments['Negative']
percent_pos=(count_pos/total_count)*100
percent_neu=(count_neu/total_count)*100
percent_neg=(count_neg/total_count)*100
print("Percentage of comments in the App 10 Best Foods for You are \n")
print("Percent Positive Comments: {:.2f}\n Percent Negative Comments: {:.2f}\n Percent Neutral Comments: {:.2f}".format(percent_pos,percent_neg,percent_neu))
####################################################################################################

```

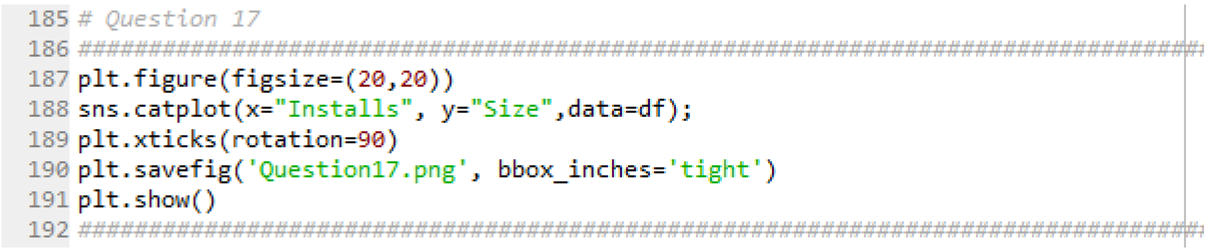
Question 17: Does the size of the App influence the number of installs that it gets ? if,yes the trend is positive or negative with the increase in the app size.

Output:



--> From the graph shown, there is no specific relation between size & installs

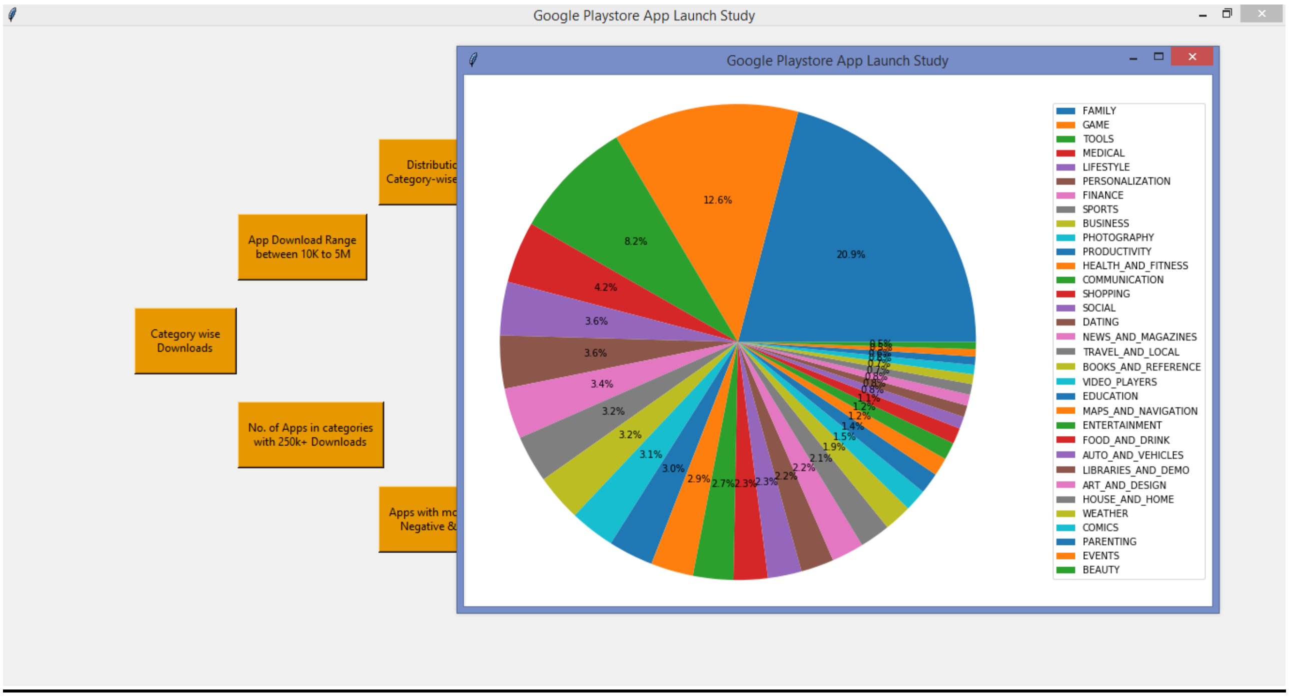
Code:



```
# Question 17
##################################################################################
plt.figure(figsize=(20,20))
sns.catplot(x="Installs", y="Size",data=df);
plt.xticks(rotation=90)
plt.savefig('Question17.png', bbox_inches='tight')
plt.show()
##################################################################################
```

Graphical User Interface for the project:

Shown below is the GUI with buttons for each question of the project implemented above



➔ After pressing the buttons, the output will be displayed as shown above. The code for the GUI is present in the Final code given below

Final Code:

```
# Importing Python Libraries
###########################################################################################
##########
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
import os
import csv
from tkinter import *
import tkinter.ttk
from PIL import ImageTk, Image
from tkinter import messagebox
###########################################################################################
##########

# Reading csv file
###########################################################################################
##########
df=pd.read_csv('googleplaystore1.csv')
df1=pd.read_csv('googleplaystore2.csv')
```

```
##########################################################################################
##########

# Data Cleaning
##########################################################################################
##########

# Data cleaning for "Installs" column

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

# Data cleaning for "Price" column

df['Price']=df['Price'].map(lambda x: x.lstrip('$').rstrip())

# Cleaning size of installation

def change_size(size):
    if 'M' in size:
        x = size[:-1]
        x = float(x)*1000000
        return(x)
    elif 'k' == size[-1:]:
        x = size[:-1]
        x = float(x)*1000
        return(x)
    else:
        return None

df["Size"] = df["Size"].map(change_size)

# Sort by "Category"

df.sort_values("Category", inplace = True)

# Drop rows of NAN or missing value

df=df.dropna()
df=df.reset_index(drop=True)
df1=df1.dropna()
df1=df1.reset_index(drop=True)

# Change datatype

df['Reviews'] = pd.to_numeric(df['Reviews'])
df['Installs'] = pd.to_numeric(df['Installs'])
df['Price'] = pd.to_numeric(df['Price'])

# Data Cleaning For Android Version
andro_ver=df[df['Android Ver']!='Varies with device']
andro_ver['Android Ver']=andro_ver['Android Ver'].map(lambda x: x.rstrip('and up'))
andro_ver['Android Ver']=andro_ver['Android Ver'].map(lambda x: x[:3])
andro_ver['Android Ver']=pd.to_numeric(andro_ver['Android Ver'])
##########################################################################################
##########
```

```
# Question 1
##########################################################################################
##########
fig, ax = plt.subplots(figsize=(10, 10), subplot_kw=dict(aspect="equal"))
number_of_apps_cat = df["Category"].value_counts()
labelss = number_of_apps_cat.index
sizes = number_of_apps_cat.values
ax.pie(sizes,labeldistance=2,autopct='%1.1f%%')
ax.legend(labels=labelss,loc="right",bbox_to_anchor=(0.9, 0, 0.5, 1))
ax.axis("equal")
fig.savefig('Question1.png', bbox_inches='tight')
plt.show()
##########################################################################################
##########

# Question 3
##########################################################################################
##########
counter_dl=[]
summinst=0
j=0;
cat=[]
countinst=0
for i in range(0,len(df)):
    try:
        if df['Category'][i]==df['Category'][i+1]:
            if df['Installs'][i]>250000:
                countinst=countinst+1
        else:
            if df['Installs'][i]>250000:
                countinst=countinst+1
            counter_dl.append(countinst)
            countinst=0
            if len(counter_dl)==32:
                for j in range(i,len(df)):
                    if df['Installs'][j]>250000:
                        countinst=countinst+1
                counter_dl.append(countinst)

    except:
        pass
fig=plt.bar(labelss,counter_dl,color=sns.color_palette("Blues",3))
plt.title("Downloads above 250000 category wise")
plt.ylabel("No of apps in categories")
plt.xlabel("Category")
plt.xticks(rotation=90)
plt.savefig('Question3.png', bbox_inches='tight')
plt.show()
##########################################################################################
##########

# Question 4
##########################################################################################
##########
avg_rating = df["Rating"].mean()
print(avg_rating)
plt.figure(figsize=(10,8))
```

```
sns.boxplot('Category','Rating',data=df)
plt.title("Distribution of Categorywise Ratings")
plt.ylabel("Rating")
plt.xlabel("Category")
plt.xticks(rotation=90)
plt.savefig('Question4.png', bbox_inches='tight')
plt.show();
##########################################################################################
##########

# Question 2
##########################################################################################
##########
c=[0,0,0,0,0,0]
for i in range(len(df)):
    if df['Installs'][i]<10000:
        c[0]=c[0]+1
    if df['Installs'][i]>=10000 and df['Installs'][i]<50000:
        c[1]=c[1]+1
    if df['Installs'][i]>=50000 and df['Installs'][i]<150000:
        c[2]=c[2]+1
    if df['Installs'][i]>=150000 and df['Installs'][i]<500000:
        c[3]=c[3]+1
    if df['Installs'][i]>=500000 and df['Installs'][i]<5000000:
        c[4]=c[4]+1
    if df['Installs'][i]>=5000000:
        c[5]=c[5]+1
##########################################################################################
#########

# Quetion 9
##########################################################################################
##########
installs_greater_100000 = df[df["Installs"]>100000]
installs_greater_100000 = installs_greater_100000.sort_values(['Rating'])
plt.figure(figsize=(20,20))
sns.catplot(x="Installs", y="Rating",data=installs_greater_100000);
plt.xticks(rotation=90)
plt.savefig('Question9.png', bbox_inches='tight')
plt.show()
##########################################################################################
##########

# Question 7
##########################################################################################
##########
andro_ver = andro_ver.sort_values(['Android Ver'])
plt.figure(figsize=(20,20))
sns.catplot(x="Installs", y="Android Ver",data=andro_ver);
plt.xticks(rotation=90)
plt.savefig('Question7.png', bbox_inches='tight')
plt.show()
##########################################################################################
##########

# Question 17
```

```
##########################################################################################
##########
plt.figure(figsize=(20,20))
sns.catplot(x="Installs", y="Size",data=df);
plt.xticks(rotation=90)
plt.savefig('Question17.png', bbox_inches='tight')
plt.show()
##########################################################################################
##########

# Question 10
##########################################################################################
##########
months = ['Jan', 'Feb','Mar' , 'Apr', 'May', 'Jun', 'Aug', 'Sept','Oct','Nov','Dec' ]
temp_app_data=df.copy()
downloads=[]
font={'size':10}
for m in months:
    df2=temp_app_data[temp_app_data['Last Updated'].str.contains(m)]
    m_categories=list(set(list(df2['Category'])))
    df2_groups=df2.groupby('Category').sum()
    downloads.append(max(df2_groups['Installs']))
    print(m,df2_groups.idxmax()['Installs'],max(df2_groups['Installs']))
def label(graph,counts):
    for rect, count in zip(graph,counts):
        height=rect.get_height()
        plt.text(rect.get_x()+rect.get_width()/2.,0.85*height,'%d' % count,ha='center',va='bottom',fontdict=font)
pos=np.arange(len(months))
print(months,downloads)
plt.figure(figsize=(16,4))
graph=plt.bar(pos,downloads,align='center',alpha=0.9)
plt.xticks(pos,months,rotation='horizontal')
plt.ylabel('Download count')
plt.xlabel('Months')
#labelling the bars in the bar graph
label(graph,downloads)
plt.savefig('Question10.png', bbox_inches='tight')
plt.show()
##########################################################################################
##########

# Question 15
##########################################################################################
##########
best_foods_for_you=df1[df1['App']=='10 Best Foods for You']
count_of_sentiments=best_foods_for_you['Sentiment'].value_counts()
total_count=count_of_sentiments['Positive']+count_of_sentiments['Neutral']+count_of_sentiments['Negative'
]
count_pos=count_of_sentiments['Positive']
count_neu=count_of_sentiments['Neutral']
count_neg=count_of_sentiments['Negative']
percent_pos=(count_pos/total_count)*100
percent_neu=(count_neu/total_count)*100
percent_neg=(count_neg/total_count)*100
print("Percent Positive Comments: {:.2f}\n Percent Negative Comments: {:.2f}\n Percent Neutral Comments:
{:.2f}".format(percent_pos,percent_neg,percent_neu))
```

```
###########################################################################################
##########

# Question 5
###########################################################################################
##########
# Returning years of the given date
def refine_dates(val):
   return val[-4:]
temp_app_data=df.copy()
temp_app_data['Last Updated']=temp_app_data.apply(lambda row: refine_dates(row['Last Updated']),axis=1)
# Sorting based on last Updated and Installs
dictionary={}
years=temp_app_data.sort_values(['Last Updated','Installs']).groupby('Last Updated')
print(years)
groups=list(years.groups.keys())
for group in groups:
   g=years.get_group(group)
   print("\n Group: ",group)
   print("\n",Counter(g['Category']))
   print("\n")
###########################################################################################
##########

# Qustion_6
###########################################################################################
##########
print("\n\n##########################################################################\n#######
##################################################################\n\n")
dictionary={
      'max':{
            '2016':[],
            '2017':[],
            '2018':[]
            },
      'min':{
            '2016':[],
            '2017':[],
            '2018':[]
            }
      }
groups=['2016','2017','2018']
for d in groups:
   g=years.get_group(d)
   max_installs=g[g['Installs']==max(g['Installs'])]
   dictionary['max'][d]=list(set(list(max_installs['Category'])))
   print("\n The Category with maximum installs in the year
"+d+"\n"+"{}".format(','.join(dictionary['max'][d]))+"\n with the count: ",max(g['Installs']))
   min_installs=g[g['Installs']==min(g['Installs'])]
   dictionary['min'][d]=list(set(list(min_installs['Category'])))
   print("\n The categories with minimum installs in the year "+d+"\n"+" {}".format(',
'.join(dictionary['min'][d]))+"\n with the count: ",min(g['Installs']))
###########################################################################################
##########

# Question 14
```

```
##################################################################################################
##########
dictionary={}
list_apps=list(set(list(df1['App'])))
for app in list_apps:
    af = df1[df1['App']== app]
    dictionary[app] = {
            'positive': [],
            'negative':[],
            'neutral':[]
            }
    af_pos = af[af['Sentiment'] == 'Positive']
    af_pos = af[af['Sentiment'] == 'Negative']
    af_pos = af[af['Sentiment'] == 'Neutral']

dictionary[app]['positive'] = list(af_pos['Translated_Review'])
dictionary[app]['negative'] = list(af_pos['Translated_Review'])
dictionary[app]['neutral'] = list(af_pos['Translated_Review'])
##################################################################################################
##########

# Question 12
##################################################################################################
##########
dictionary={'most positive':[],'most negative':[],'same ratio':[]}
for i in range(len(list_apps)):
    df2=df1[df1['App']==list_apps[i]]
    df2_pos=df2[df2['Sentiment']=='Positive']
    df2_neg=df2[df2['Sentiment']=='Negative']
    df2_neu=df2[df2['Sentiment']=='Neutral']
    if len(df2_pos)>len(df2_neg)+len(df2_neg):
        dictionary['most positive'].append(list_apps[i])
    if len(df2_neg)>len(df2_pos)+len(df2_neg):
        dictionary['most negative'].append(list_apps[i])
    if len(df2_pos)==len(df2_neg):
        dictionary['same ratio'].append(list_apps[i])
print(dictionary)
##################################################################################################
##########

# Graphical User Interface Section
##################################################################################################
##########

# Making Functions For Each Buttons
def Q1():
    screen1=Toplevel(screen)
    image=Image.open("Question1.png")
    photo=ImageTk.PhotoImage(image)
    label=Label(screen1,image=photo)
    label.image = photo # keep a reference
    label.pack()

def Q2():
    screen1=Toplevel(screen)
    image=Image.open("Question2.png")
    photo=ImageTk.PhotoImage(image)
    label=Label(screen1,image=photo)
    label.image = photo # keep a reference
    label.pack()


def Q3():
    screen1=Toplevel(screen)
    image=Image.open("Question3.png")
    photo=ImageTk.PhotoImage(image)
    label=Label(screen1,image=photo)
    label.image=photo # keep a reference
    label.pack()

def Q4():
    screen1=Toplevel(screen)
    image=Image.open("Question4.png")
    photo=ImageTk.PhotoImage(image)
    label=Label(screen1,image=photo)
    label.image = photo # keep a reference
    label.pack()

def Q5a():
    screen2=Toplevel(screen)
    image=Image.open("Q5a.png")
    photo=ImageTk.PhotoImage(image)
    label=Label(screen2,image=photo)
    label.image = photo # keep a reference
    label.pack()

def Q5b():
    screen2=Toplevel(screen)
    image=Image.open("Q5b.png")
    photo=ImageTk.PhotoImage(image)
    label=Label(screen2,image=photo)
    label.image = photo # keep a reference
    label.pack()

def Q5():
    screen1=Toplevel(screen)
    Button(screen1,text='\n  2010-2014 \n Trends  \n',bg="#e79700",command=Q5a).place(x=20,y=20)
    Button(screen1,text='\n  2015-2018 \n Trends  \n',bg="#e79700",command=Q5b).place(x=20,y=90)
    screen.mainloop()

def Q6():
    screen1=Toplevel(screen)
    image=Image.open("Question6.png")
    photo=ImageTk.PhotoImage(image)
    label=Label(screen1,image=photo)
    label.image = photo # keep a reference
    label.pack()

def Q7():
    screen1=Toplevel(screen)
    image=Image.open("Question7.png")
    photo=ImageTk.PhotoImage(image)
    label=Label(screen1,image=photo)
    label.image = photo # keep a reference
    label.pack()

def Q9():
    screen1=Toplevel(screen)
    image=Image.open("Question9.png")
    photo=ImageTk.PhotoImage(image)
    label=Label(screen1,image=photo)
    label.image = photo # keep a reference
    label.pack()

def Q10():
    screen1=Toplevel(screen)
    image=Image.open("Question10.png")
    photo=ImageTk.PhotoImage(image)
    label=Label(screen1,image=photo)
    label.image = photo # keep a reference
    label.pack()

def Q12():
    screen1=Toplevel(screen)
    image=Image.open("Question12.png")
    photo=ImageTk.PhotoImage(image)
    label=Label(screen1,image=photo)
    label.image = photo # keep a reference
    label.pack()

def Q15():
    screen1=Toplevel(screen)
    image=Image.open("Question10.png")
    photo=ImageTk.PhotoImage(image)
    label=Label(screen1,image=photo)
    label.image = photo # keep a reference
    label.pack()

def Q17():
    screen1=Toplevel(screen)
    image=Image.open("Question17.png")
    photo=ImageTk.PhotoImage(image)
    label=Label(screen1,image=photo)
    label.image = photo # keep a reference
    label.pack()

#Making Interface

screen=Tk()
screen.title("Google Playstore App Launch Study")
screen.geometry("1280x720+20+40")
img = ImageTk.PhotoImage(Image.open("GUICenterimage.png"))
panel = Label(screen, image = img)
panel.image=img
panel.pack(side = "bottom", fill = "both", expand = "yes")
Button(screen,text='\n    Category wise    \n  Downloads  \n',bg="#e79700",command=Q1).place(x=140,y=300)
Button(screen,text='\n  App Download Range  \n   between 10K to 5M
\n',bg="#e79700",command=Q2).place(x=250,y=200)
Button(screen,text='\n  No. of Apps in categories  \n  with 250k+ Downloads
\n',bg="#e79700",command=Q3).place(x=250,y=400)
```

```
Button(screen,text='\n  Distribution of \n Category-wise ratings
\n',bg="#e79700",command=Q4).place(x=400,y=120)
Button(screen,text='\n  Year-wise App Category  \n  Download Trend
\n',bg="#e79700",command=Q5).place(x=1050,y=290)
Button(screen,text='\n  Correlation between  \n  No. of Installs & Android Version
\n',bg="#e79700",command=Q7).place(x=550,y=50)
Button(screen,text='\n  Ratings V/S Downloads   \n \n',bg="#e79700",command=Q9).place(x=762,y=120)
Button(screen,text='\n  Month-wise App Category  \n  Download Count
\n',bg="#e79700",command=Q10).place(x=900,y=200)
Button(screen,text='\n  10 Best Foods For You  \n  User Reviews
\n',bg="#e79700",command=Q15).place(x=940,y=390)
Button(screen,text='\n    Size Effect    \n  on Installs  \n',bg="#e79700",command=Q17).place(x=620,y=550)
Button(screen,text='\n  Max & Min Installs \n for years 2016,2017,2018
\n',bg="#e79700",command=Q6).place(x=762,y=490)
Button(screen,text='\n  Apps with most Positive  \n  Negative & Neutral
\n',bg="#e79700").place(x=400,y=490)
screen.mainloop()
```

Conclusion: Thus, we have Implemented the analysis of both the playstore datasets and displayed the data analysis in the form of different graphs like Pie chart, boxplots, catplots etc. Visualizing the output data and the graphs will assist the company in predicting the parameters and the category of app to be launched in future.