Play Store App Review Analysis

The Play Store apps data has enormous potential to drive app-making businesses to success. Actionable insights can be drawn for developers to work on and capture the Android market.

Each app (row) has values for catergory, rating, size, and more. Another dataset contains customer reviews of the android apps.

Explore and analyze the data to discover key factors responsible for app engagement and success.

Importing Libraries

```
#Import all library that will be used in entire project
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
```

Mount Drive And Import Data

```
#Mount google drive for access of the play store dataset
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Importing Dataset
File_path='/content/drive/MyDrive/Capstone Projects/Ted talk view prediction/'
play_store_data= pd.read_csv(File_path + 'Play Store Data.csv')
user_reviews_data=pd.read_csv(File_path + 'User Reviews.csv')
```

```
# First Look
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.max_colwidth', -1)
play_store_data.head()
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Cont Rat
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Every
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Every
2	U Launcher Lite – FREE Live Cool	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Every

```
#tail of data
play_store_data.tail()
```

	App	Category	Rating	Reviews	Size	Installs	Type
10836	Sya9a Maroc - FR	FAMILY	4.5	38	53M	5,000+	Free
10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3.6M	100+	Free
10838	Parkinson Exercices FR	MEDICAL	NaN	3	9.5M	1,000+	Free
10839	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5	114	Varies with device	1,000+	Free
	iHoroscope						

```
play_store_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10841 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             10841 non-null  object 
 1   Category        10841 non-null  object 
 2   Rating          9367 non-null   float64
 3   Reviews         10841 non-null  object 
 4   Size            10841 non-null  object 
 5   Installs        10841 non-null  object 
 6   Type            10840 non-null  object 
 7   Price           10841 non-null  object 
 8   Content Rating  10840 non-null  object 
 9   Genres          10841 non-null  object 
 10  Last Updated    10841 non-null  object 
 11  Current Ver     10833 non-null  object 
 12  Android Ver     10838 non-null  object 
dtypes: float64(1), object(12)
memory usage: 1.1+ MB
```

```
user_reviews_data.head()
```

	App	Translated_Review	Sentiment	Sentiment_Polarity	Sentiment_Subjectivity
0	10 Best Foods for You	I like eat delicious food. That's I'm cooking food myself, case "10 Best Foods" helps lot, also "Best Before (Shelf Life)"	Positive	1.00	0.533333
1	10 Best Foods for You	This help eating healthy exercise regular basis	Positive	0.25	0.288462
	10 Best				

```
user_reviews_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64295 entries, 0 to 64294
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   App                     64295 non-null  object 
 1   Translated_Review       37427 non-null  object 
 2   Sentiment               37432 non-null  object 
 3   Sentiment_Polarity      37432 non-null  float64
 4   Sentiment_Subjectivity  37432 non-null  float64
dtypes: float64(2), object(3)
memory usage: 2.5+ MB
```

Let's Drive Into The Play Store Data

```
#Discription of Data
play_store_data.describe(include='all')
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Con Ra
count	10841	10841	9367.000000	10841	10841	10841	10840	10841	1
unique	9660	34	NaN	6002	462	22	3	93	
top	ROBLOX	FAMILY	NaN	0	Varies with device	1,000,000+	Free	0	Ever
freq	9	1972	NaN	596	1695	1579	10039	10040	
mean	NaN	NaN	4.193338	NaN	NaN	NaN	NaN	NaN	
std	NaN	NaN	0.537431	NaN	NaN	NaN	NaN	NaN	
min	NaN	NaN	1.000000	NaN	NaN	NaN	NaN	NaN	
25%	NaN	NaN	4.000000	NaN	NaN	NaN	NaN	NaN	
50%	NaN	NaN	4.300000	NaN	NaN	NaN	NaN	NaN	
75%	NaN	NaN	4.500000	NaN	NaN	NaN	NaN	NaN	

Handling Missing Vaules

```
# Missing Value Count Function
def show_missing():
    missing = play_store_data.columns[play_store_data.isnull().any()].tolist()
    return missing

# Missing data counts and percentage
print('Missing Data Count')
print(play_store_data[show_missing()].isnull().sum().sort_values(ascending = False))
print('--'*50)
print('Missing Data Percentage')
print(round(play_store_data[show_missing()].isnull().sum().sort_values(ascending = False)
```

```
Missing Data Count
Rating            1474
Current Ver       8
Android Ver       3
Type              1
Content Rating    1
dtype: int64
----------------------------------------------------------------------------------------
Missing Data Percentage
Rating            13.60
Current Ver       0.07
Android Ver       0.03
Type              0.01
Content Rating    0.01
dtype: float64
```

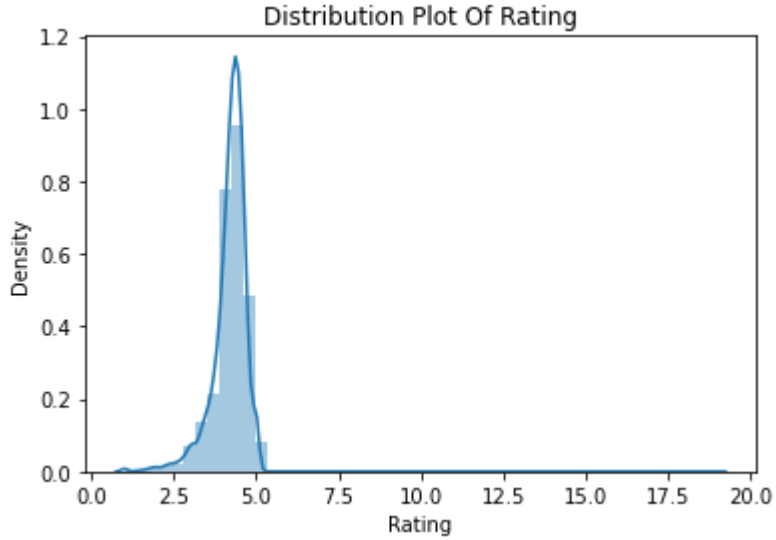
Rating

```
#data of null rating values
play_store_data[play_store_data['Rating'].isnull()].head()
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Con Ra
23	Mcqueen Coloring pages	ART_AND_DESIGN	NaN	61	7.0M	100,000+	Free	0	Ever
113	Wrinkles and rejuvenation	BEAUTY	NaN	182	5.7M	100,000+	Free	0	Ever
123	Manicure - nail design	BEAUTY	NaN	119	3.7M	50,000+	Free	0	Ever
126	Skin Care and Natural Beauty	BEAUTY	NaN	654	7.4M	100,000+	Free	0	

```
#Distribution Plot Of Rating
sns.distplot(play_store_data.Rating.dropna())
plt.title('Distribution Plot Of Rating')
```

```
Text(0.5, 1.0, 'Distribution Plot Of Rating')
```




```
#median of rating
play_store_data.Rating.dropna().median()
```

```
4.3
```

```
#Fill NaN Values in Rating Column with Median
play_store_data['Rating'] = play_store_data['Rating'].fillna(play_store_data['Rating'].me


# Missing Value Count Function
def show_missing():
    missing = play_store_data.columns[play_store_data.isnull().any()].tolist()
    return missing

# Missing data counts and percentage
print('Missing Data Count')
print(play_store_data[show_missing()].isnull().sum().sort_values(ascending = False))
print('--'*50)
print('Missing Data Percentage')
print(round(play_store_data[show_missing()].isnull().sum().sort_values(ascending = False)
```

```
Missing Data Count
Current Ver       8
Android Ver       3
Type              1
Content Rating    1
dtype: int64
----------------------------------------------------------------------------------------
Missing Data Percentage
Current Ver       0.07
Android Ver       0.03
Type              0.01
Content Rating    0.01
dtype: float64
```

```
#Dropping null as they are now negligible
play_store_data.dropna(inplace=True)


play_store_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10829 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             10829 non-null  object 
 1   Category        10829 non-null  object 
 2   Rating          10829 non-null  float64
 3   Reviews         10829 non-null  object 
 4   Size            10829 non-null  object 
 5   Installs        10829 non-null  object 
 6   Type            10829 non-null  object 
 7   Price           10829 non-null  object 
 8   Content Rating  10829 non-null  object 
 9   Genres          10829 non-null  object 
 10  Last Updated    10829 non-null  object 
 11  Current Ver     10829 non-null  object 
 12  Android Ver     10829 non-null  object 
dtypes: float64(1), object(12)
memory usage: 1.2+ MB
```

Cleaning Data And Making It In Proper Format

App

```
#looking for duplicate apps if any
play_store_data['App'].value_counts().head().reset_index()
```

	index	App
0	ROBLOX	9
1	CBS Sports App - Scores, News, Stats & Watch Live	8
2	Candy Crush Saga	7
3	8 Ball Pool	7
4	ESPN	7

```
#data of one of the duplicate app
play_store_data[play_store_data['App']=="ROBLOX"]
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	
1653	ROBLOX	GAME	4.5	4447388	67M	100,000,000+	Free	0	Everyone 10+	A
1701	ROBLOX	GAME	4.5	4447346	67M	100,000,000+	Free	0	Everyone 10+	A
1748	ROBLOX	GAME	4.5	4448791	67M	100,000,000+	Free	0	Everyone 10+	A
1841	ROBLOX	GAME	4.5	4449882	67M	100,000,000+	Free	0	Everyone 10+	A
1870	ROBLOX	GAME	4.5	4449910	67M	100,000,000+	Free	0	Everyone 10+	A
2016	ROBLOX	FAMILY	4.5	4449910	67M	100,000,000+	Free	0	Everyone 10+	A

```
#As all rows have same data, only a slight difference in Reviews, we will delete duplicat
play_store_data.drop_duplicates(subset=['App'],inplace=True)
```

									Everyone	A

Reviews

4527	ROBLOX	FAMILY	4.5	4443407	67M	100,000,000+	Free	0	Everyone	A

```
#type of reviews
type(play_store_data['Reviews'].iloc[0])
```

```
str
```

```
#converting Reviews type into integer
play_store_data['Reviews'] = pd.to_numeric(play_store_data['Reviews'])
```

Size

```
#size value counts
play_store_data['Size'].value_counts().head()
```

```
Varies with device    1226
12M                   181
11M                   181
13M                   177
14M                   176
Name: Size, dtype: int64
```

```
play_store_data['Size'].value_counts().tail()
```

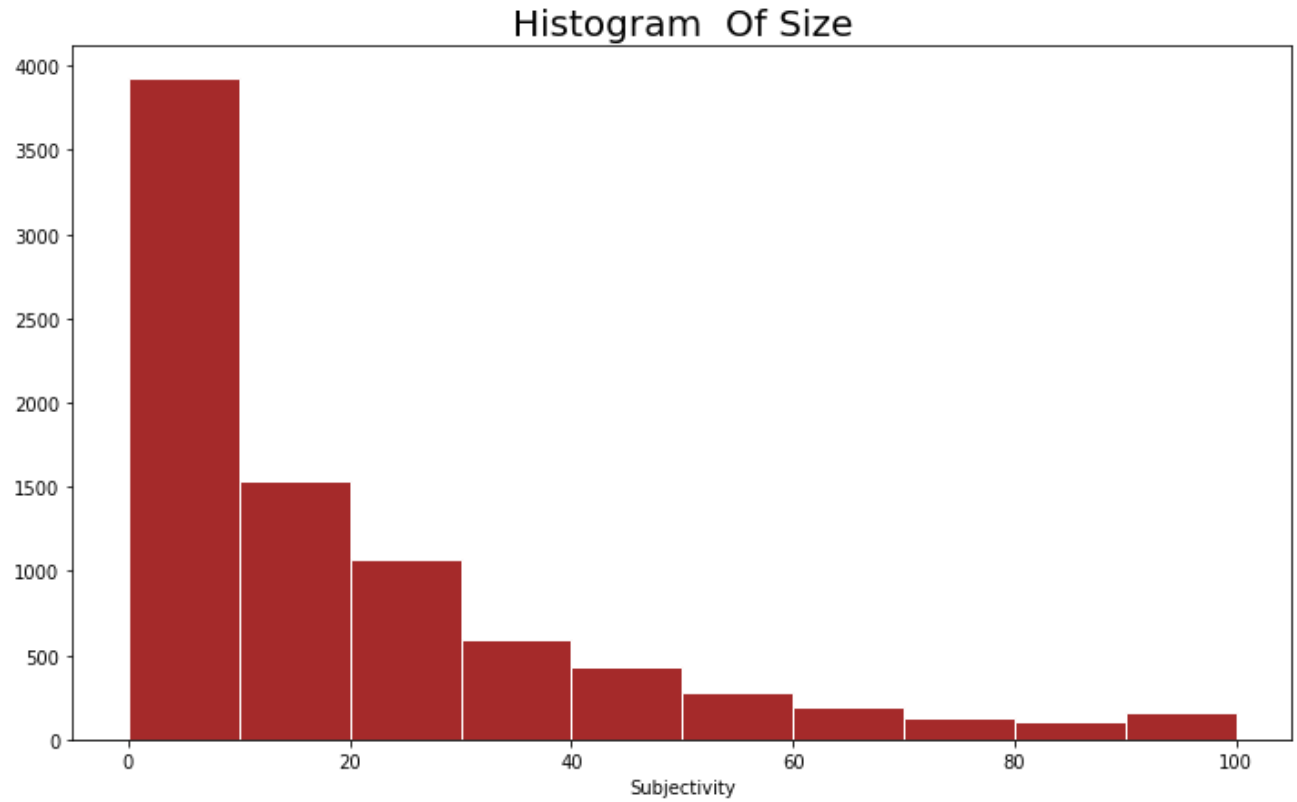
```
721k    1
430k    1
429k    1
200k    1
619k    1
Name: Size, dtype: int64
```

```
#Impute nulls inplace of Varies with device
play_store_data['Size'] = play_store_data['Size'].apply(lambda x: x.replace('Varies with
```

```
#removing M and k from values, also coverting KB into MB
play_store_data['Size'] =play_store_data['Size'].apply(lambda x: x.replace('M', '') if 'M
play_store_data['Size'] = play_store_data['Size'].apply(lambda x: float(x.replace('k', ''
```

```
#converting Size type into float
play_store_data['Size'] = play_store_data['Size'].astype(float)
```

```
#Histogram Of Size
plt.figure(figsize=(12,7))
plt.xlabel("Subjectivity")
plt.title("Histogram  Of Size",fontsize=20)
plt.hist(play_store_data.Size.dropna(),color="brown",edgecolor="white")
plt.show()
```




```
#size median
play_store_data.Size.dropna().median()
```

```
12.0
```

```
#Fill NaN Values in Size Column with Median
play_store_data['Size'] = play_store_data['Size'].fillna(play_store_data['Size'].median()
```

Installs

```
#Installs value counts
play_store_data['Installs'].value_counts()
```

```
1,000,000+        1416
100,000+          1112
10,000+           1029
10,000,000+        937
1,000+             886
100+               709
5,000,000+         607
500,000+           504
50,000+            468
5,000+             467
10+                384
500+               328
50+                204
50,000,000+        202
100,000,000+       188
```

```
5+                82
1+                67
500,000,000+      24
1,000,000,000+    20
0+                14
Name: Installs, dtype: int64
```

```
#Removing + and ,
play_store_data['Installs']=play_store_data['Installs'].apply(lambda x:x.replace('+','')
play_store_data['Installs']=play_store_data['Installs'].apply(lambda x:x.replace(',','')
```

```
#type of Installs
type(play_store_data.Installs.iloc[0])
```

```
str
```

```
#converting Installs type into integer
play_store_data['Installs'] = pd.to_numeric(play_store_data['Installs'])
```

Price

```
#Price value counts
play_store_data['Price'].value_counts().head()
```

```
0        8895
$0.99    143
$2.99    124
$1.99    73
$4.99    70
Name: Price, dtype: int64
```

```
#removing $
play_store_data['Price']=play_store_data['Price'].apply(lambda x:x.replace('$','') if '$'
```

```
#type of Price
type(play_store_data.Price.iloc[0])
```

```
str
```

```
#converting Price type into float
play_store_data['Price']=play_store_data['Price'].astype(float)
```

```
#checking for null values in data
play_store_data.isna().any().any()
```

```
False
```

```
play_store_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9648 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             9648 non-null   object 
 1   Category        9648 non-null   object 
 2   Rating          9648 non-null   float64
 3   Reviews         9648 non-null   int64  
 4   Size            9648 non-null   float64
 5   Installs        9648 non-null   int64  
 6   Type            9648 non-null   object 
 7   Price           9648 non-null   float64
 8   Content Rating  9648 non-null   object 
 9   Genres          9648 non-null   object 
 10  Last Updated    9648 non-null   object 
 11  Current Ver     9648 non-null   object 
 12  Android Ver     9648 non-null   object 
dtypes: float64(3), int64(2), object(8)
memory usage: 1.0+ MB
```

Now play store data is cleaned and ready to use

Analyzing Each Feature Separately

Category

```
#unique categories
len(play_store_data['Category'].unique())
```

```
33
```

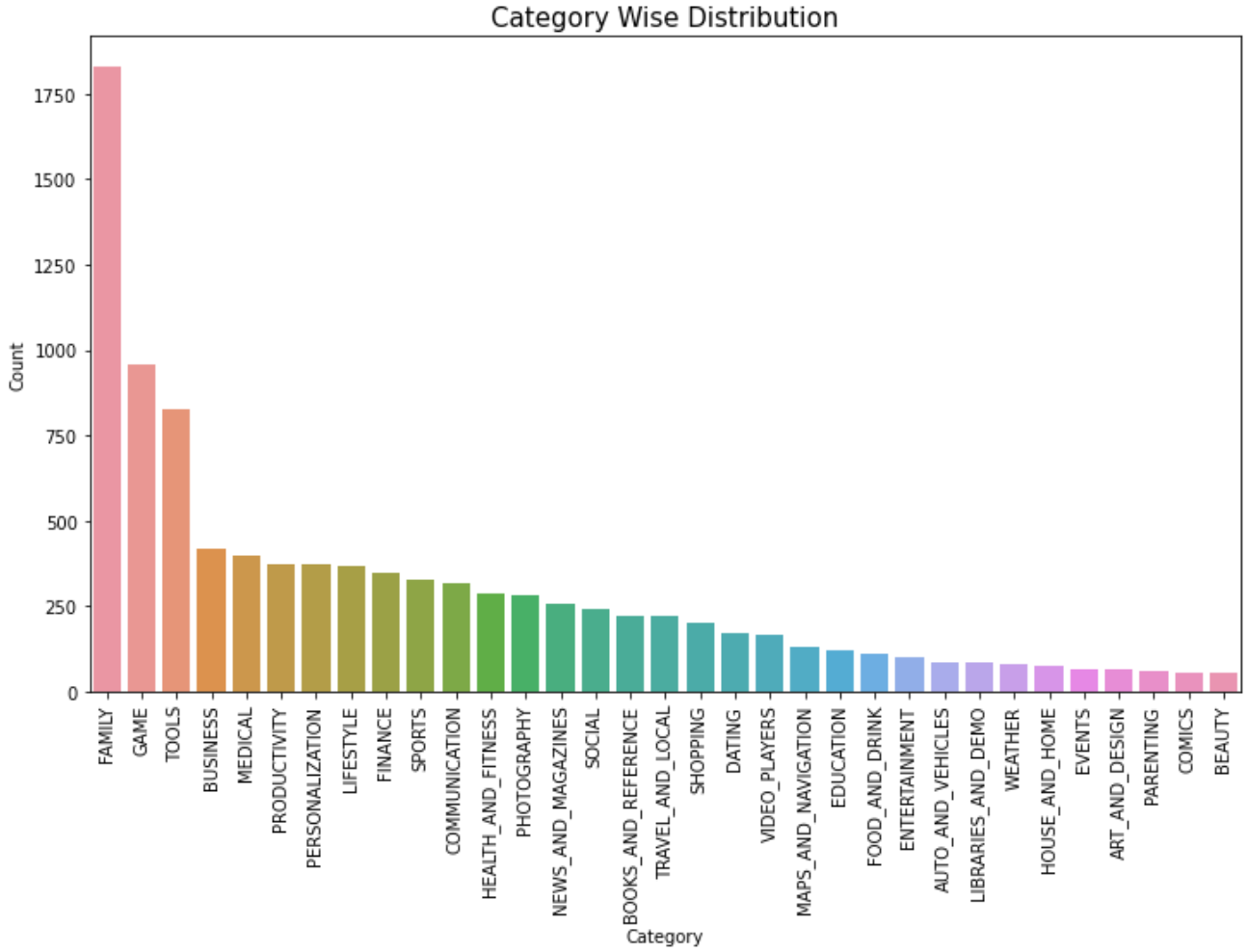
```
#category-wise counts
category_count=play_store_data['Category'].value_counts().reset_index().rename(columns={'
category_count
```

	Category	Count
0	FAMILY	1828
1	GAME	959
2	TOOLS	825
3	BUSINESS	420
4	MEDICAL	395
5	PRODUCTIVITY	374
6	PERSONALIZATION	374
7	LIFESTYLE	369
8	FINANCE	345
9	SPORTS	325
10	COMMUNICATION	315
11	HEALTH_AND_FITNESS	288
12	PHOTOGRAPHY	281
13	NEWS_AND_MAGAZINES	254
14	SOCIAL	239
15	BOOKS_AND_REFERENCE	221
16	TRAVEL_AND_LOCAL	219
17	SHOPPING	202
18	DATING	171
19	VIDEO_PLAYERS	163
20	MAPS_AND_NAVIGATION	131
21	EDUCATION	119
22	FOOD_AND_DRINK	112
23	ENTERTAINMENT	102
24	AUTO_AND_VEHICLES	85
25	LIBRARIES_AND_DEMO	83
26	WEATHER	79
27	HOUSE_AND_HOME	74
28	EVENTS	64
29	ART_AND_DESIGN	63
30	PARENTING	60

Most Of The Apps Belongs To The Family category

32 BEAUTY 53

```
plt.figure(figsize=(12,7))
sns.barplot(x='Category',y='Count',data=category_count)
plt.title('Category Wise Distribution',fontsize=15)
plt.xticks(rotation=90, horizontalalignment="center")
plt.show()
```



19% Apps Belongs To The Family Category

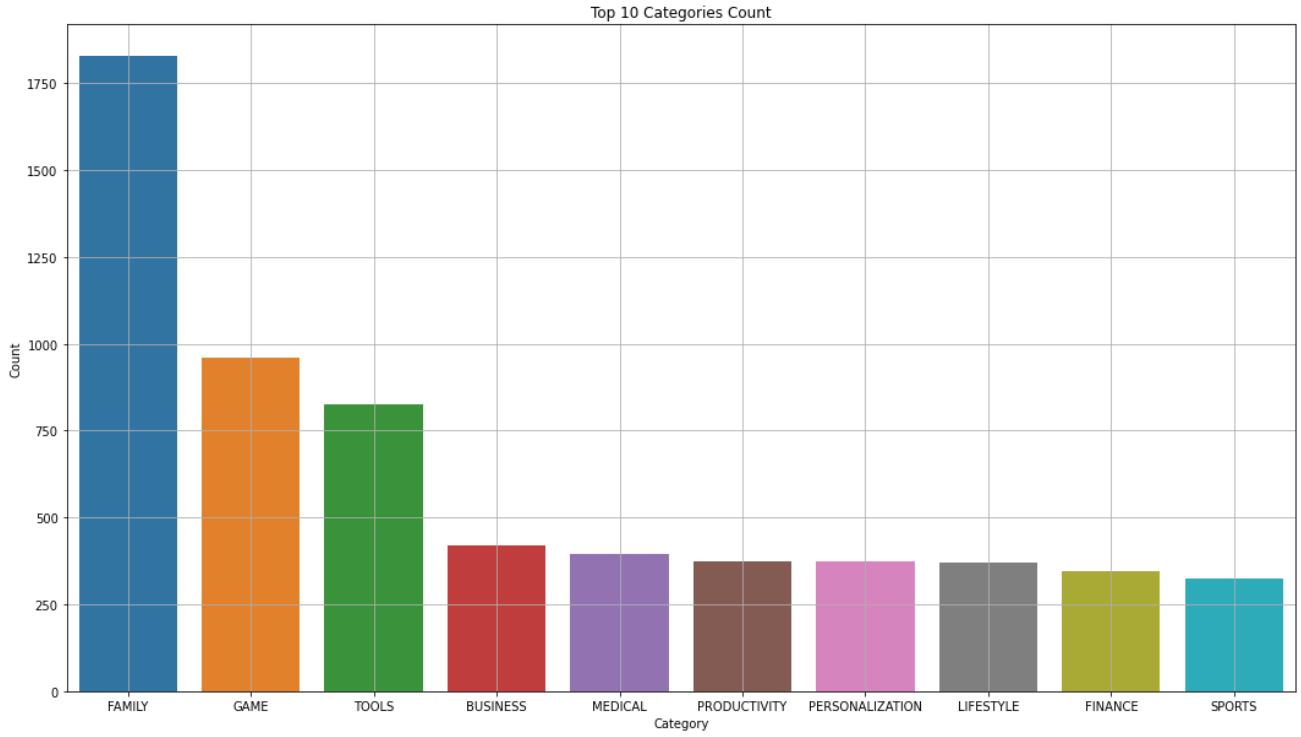
Top 10 categories

```
Top10_categories=play_store_data['Category'].value_counts().reset_index().head(10)
Top10_categories.rename(columns={'index':'Category','Category':'Count'},inplace=True)
Top10_categories
```

	Category	Count
0	FAMILY	1828
1	GAME	959
2	TOOLS	825
3	BUSINESS	420
4	MEDICAL	395
5	PRODUCTIVITY	374
6	PERSONALIZATION	374
7	LIFESTYLE	369
8	FINANCE	345
9	SPORTS	325

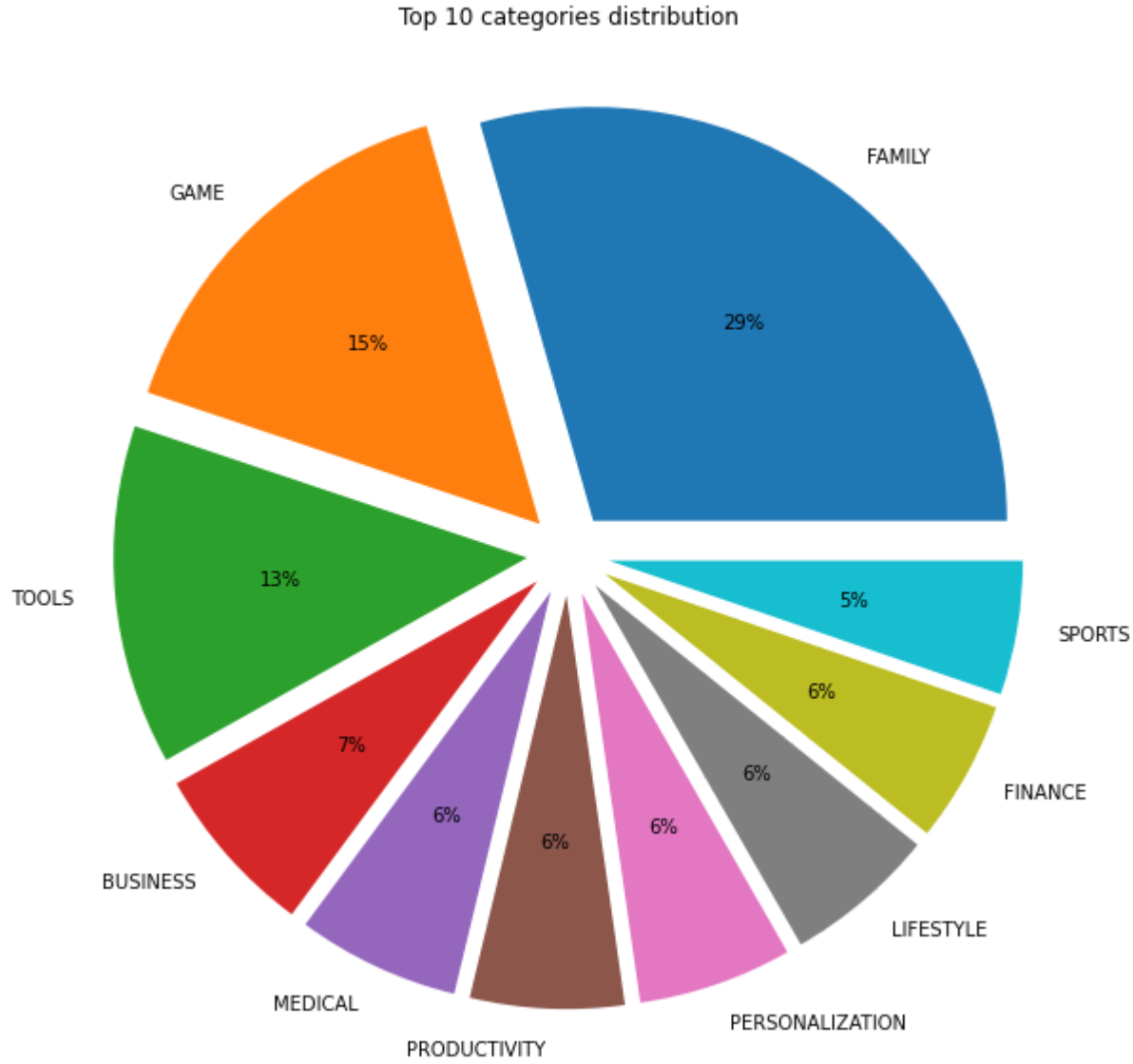
```
#Top 10 Categories Count
plt.rcParams['figure.figsize'] = (18, 10)
sns.barplot(Top10_categories['Category'],Top10_categories['Count'])
plt.grid()
plt.title('Top 10 Categories Count')
```

```
Text(0.5, 1.0, 'Top 10 Categories Count')
```



Top 10 categories distribution

```
plt.pie(Top10_categories['Count'],labels=Top10_categories['Category'],autopct='%.0f%%',ex
plt.title('Top 10 categories distribution')
plt.show()
```



Among Top 10 Categories FAmily,Games And Tools Contribute The Most

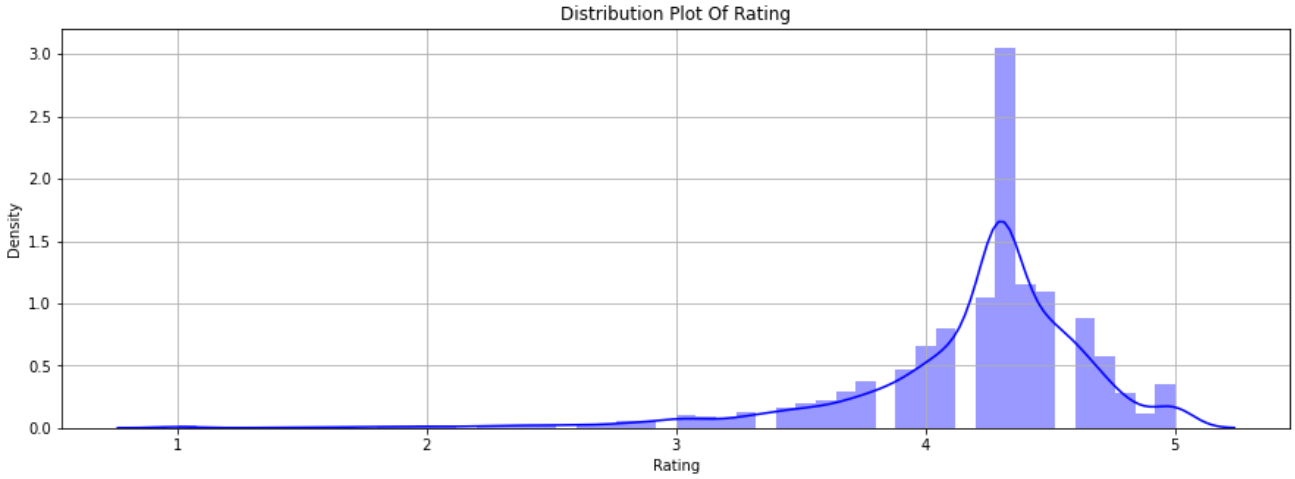
Rating

```
#Description of Rating
play_store_data['Rating'].describe()
```

```
count    9648.000000
mean        4.192465
std         0.496552
min         1.000000
25%         4.000000
50%         4.300000
75%         4.500000
max         5.000000
Name: Rating, dtype: float64
```

```
#Distribution Plot Of Rating
plt.rcParams['figure.figsize'] = (15, 5)
sns.distplot(play_store_data['Rating'],color="blue")
plt.grid()
plt.title('Distribution Plot Of Rating')
```

```
Text(0.5, 1.0, 'Distribution Plot Of Rating')
```



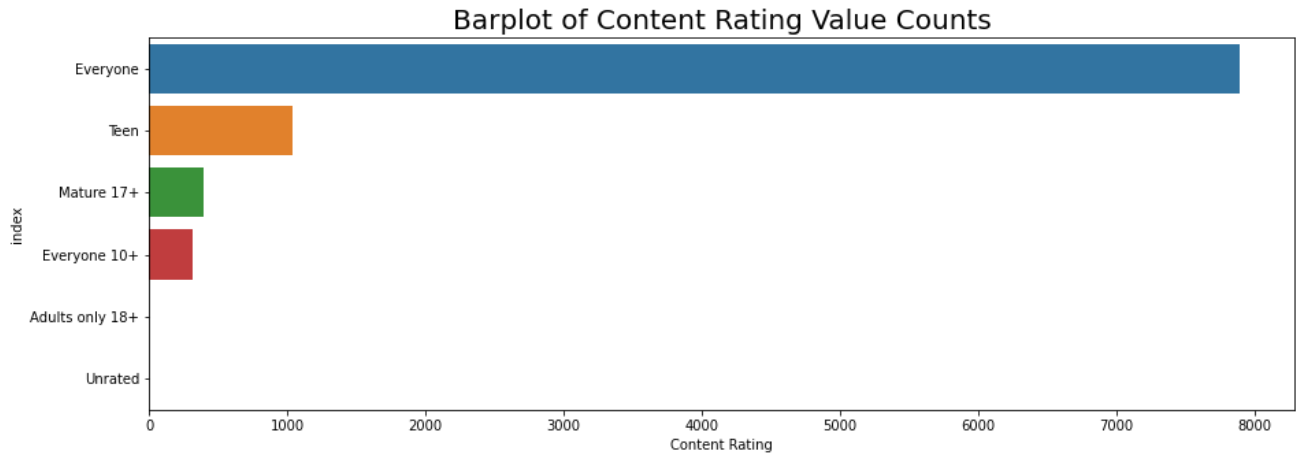
Data is negatively skewed with mean rating of 4.19

Content Rating

```
#Content rating value counts
value_c=play_store_data["Content Rating"].value_counts().reset_index()
```

```
#barplot of content rating value counts
sns.barplot(x="Content Rating",y="index",data=value_c)
plt.title("Barplot of Content Rating Value Counts",fontsize=20)
```

```
Text(0.5, 1.0, 'Barplot of Content Rating Value Counts')
```



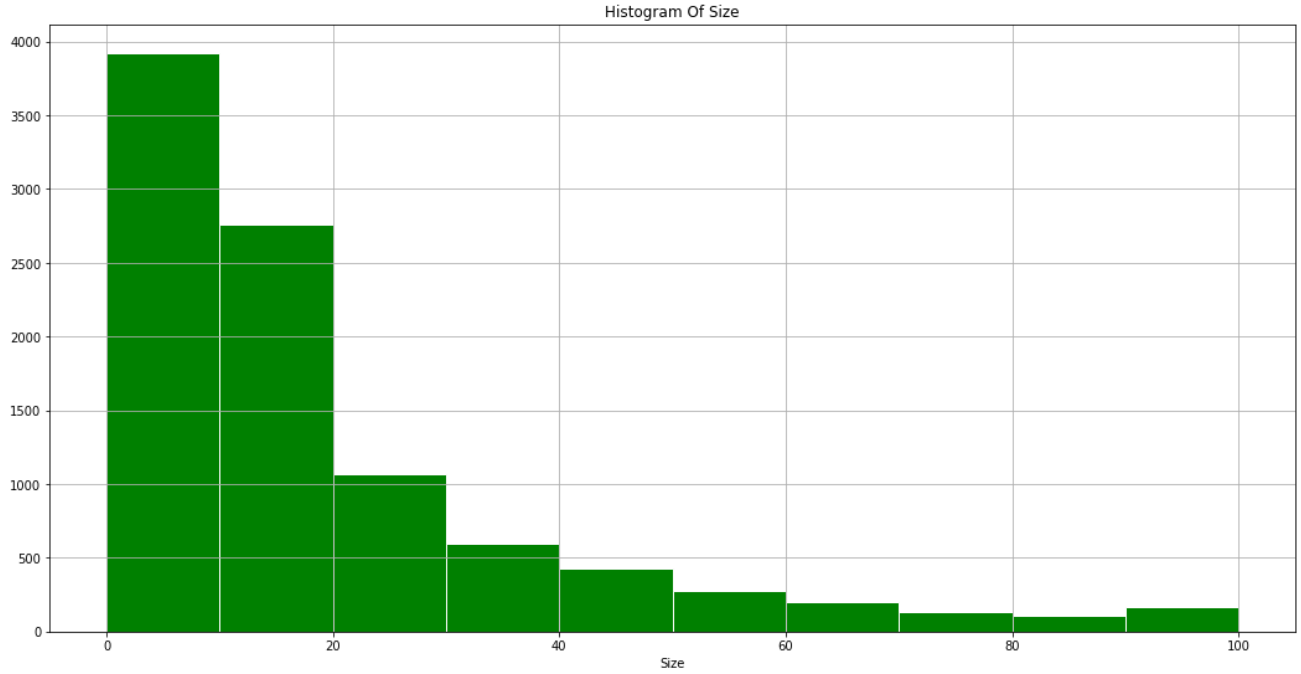
Most of The Applications Are Created For Everyone

Size

```
#Discription Of Size
play_store_data['Size'].describe()
```

```
count    9648.000000
mean       19.344761
std        20.590271
min         0.008500
25%         5.300000
50%        12.000000
75%        25.000000
max       100.000000
Name: Size, dtype: float64
```

```
#Histogram Of Size
plt.figure(figsize=(18,9))
plt.xlabel("Size")
plt.title("Histogram Of Size")
plt.hist(play_store_data.Size.dropna(),color="green",edgecolor="white")
plt.grid()
plt.show()
```



Data is positively skewed with median size of 12 MB and max size of 100 MB

Price

```
#Discription of feature Price
play_store_data['Price'].describe()
```

```
count    9648.000000
mean     1.100193
std      16.861727
min      0.000000
25%      0.000000
50%      0.000000
75%      0.000000
max      400.000000
Name: Price, dtype: float64
```

```
#Price value counts
price_count= play_store_data['Price'].value_counts().reset_index()
price_count.rename(columns={'index':'Price','Price':'Count'},inplace=True)
price_count.head()
```

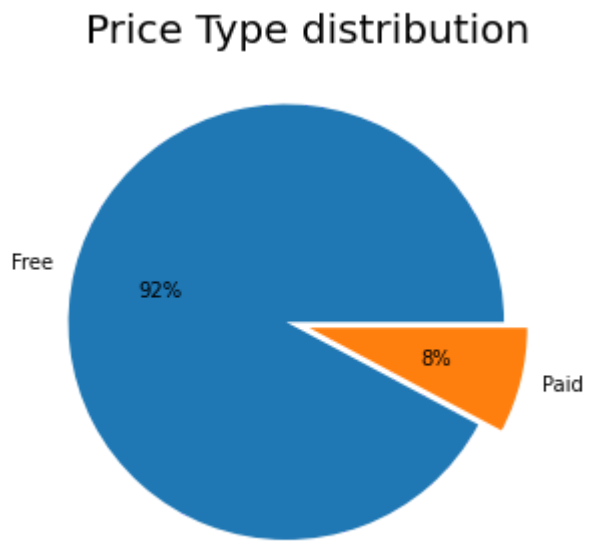
	Price	Count
0	0.00	8895
1	0.99	143
2	2.99	124
3	1.99	73
4	4.99	70

```
#Distribution Plot Of Price
plt.rcParams['figure.figsize'] = (15, 5)
sns.distplot(play_store_data['Price'])
plt.grid()
plt.title("Distribution Plot Of Price")
```

```
Text(0.5, 1.0, 'Distribution Plot Of Price')
```


```
#price type dataframe
dft=play_store_data['Type'].value_counts().reset_index()
```

```
#Price Type distribution
plt.rcParams['figure.figsize'] = (10,5)
plt.pie(dft['Type'],labels=dft['index'],autopct='%.0f%%',explode=[0.1,0.01])
plt.title('Price Type distribution',fontsize=20)
plt.show()
```



- Data is positively skewed with mean price of 1, max price of 400.
- Approximately 92% apps are free.

Genres

```
#Genres Value Counts
genres_count=play_store_data['Genres'].value_counts().reset_index()
genres_count.rename(columns={'index':'Genres','Genres':'count'},inplace=True)
genres_count.head(10)
```

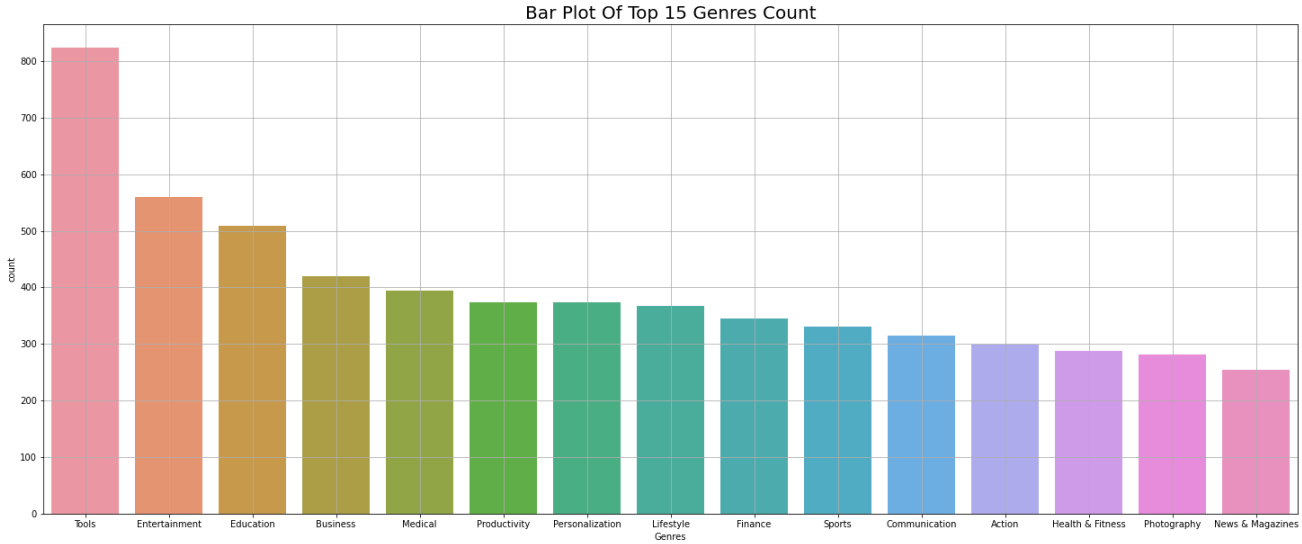
	Genres	count
0	Tools	824
1	Entertainment	560
2	Education	509
3	Business	420
4	Medical	395
5	Productivity	374
6	Personalization	374
7	Lifestyle	368
8	Finance	345
9	Sports	331

```
#Top 15 Genres
Top_15_genres=genres_count.head(15)
Top_15_genres
```

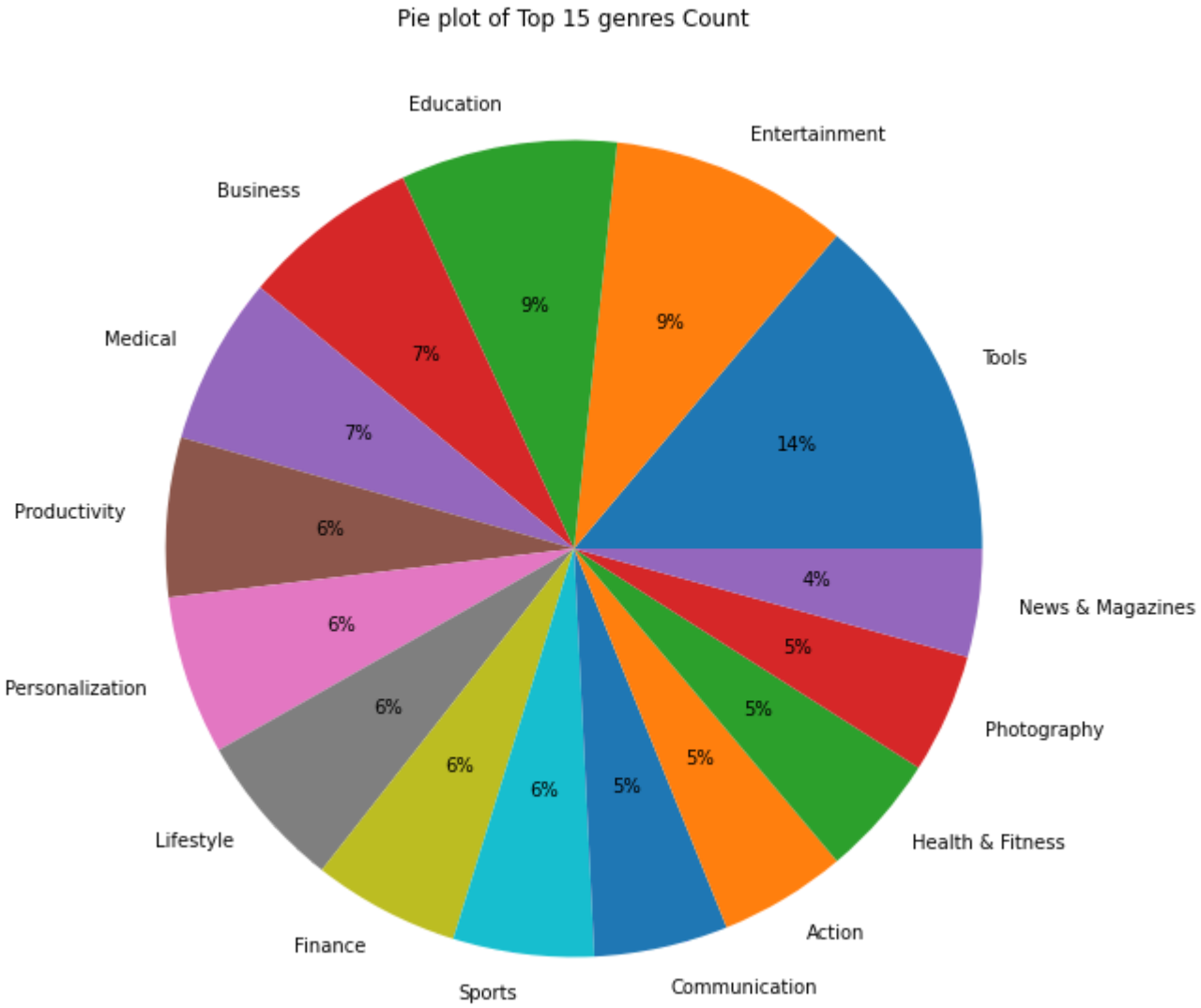
	Genres	count
0	Tools	824
1	Entertainment	560
2	Education	509
3	Business	420
4	Medical	395
5	Productivity	374
6	Personalization	374
7	Lifestyle	368
8	Finance	345
9	Sports	331
10	Communication	315
11	Action	299
12	Health & Fitness	288
13	Photography	281
14	News & Magazines	254

```
#Bar Plot Of Top 15 Genres Count
plt.rcParams['figure.figsize'] = (25, 10)
sns.barplot(Top_15_genres['Genres'],Top_15_genres['count'])
plt.grid()
plt.title('Bar Plot Of Top 15 Genres Count',fontsize=20)
```

```
Text(0.5, 1.0, 'Bar Plot Of Top 15 Genres Count')
```




```
#Pie plot of Top 15 genres Count
plt.rcParams['figure.figsize'] = (20, 10)
plt.pie(Top_15_genres['count'],labels=Top_15_genres['Genres'],autopct='%.0f%%')
plt.title('Pie plot of Top 15 genres Count')
plt.grid()
plt.show()
```



Tools is the most used genre in apps.

Analysis On How Diiferent Features Impacted On Apps

```
play_store_data.head()
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Conten Ratin
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19.0	10000	Free	0.0	Everyon
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14.0	500000	Free	0.0	Everyon
2	U Launcher Lite – FREE Live Cool	ART_AND_DESIGN	4.7	87510	8.7	5000000	Free	0.0	Everyon

Top 50 Install Apps

```
#Installs value counts
play_store_data['Installs'].value_counts().tail()
```

```
5             82
1             67
500000000     24
1000000000    20
0             14
Name: Installs, dtype: int64
```

```
#Top 50 installs apps
Top50_installs_apps=play_store_data.sort_values(by=['Installs'], ascending =False).head(5
Top50_installs_apps
```

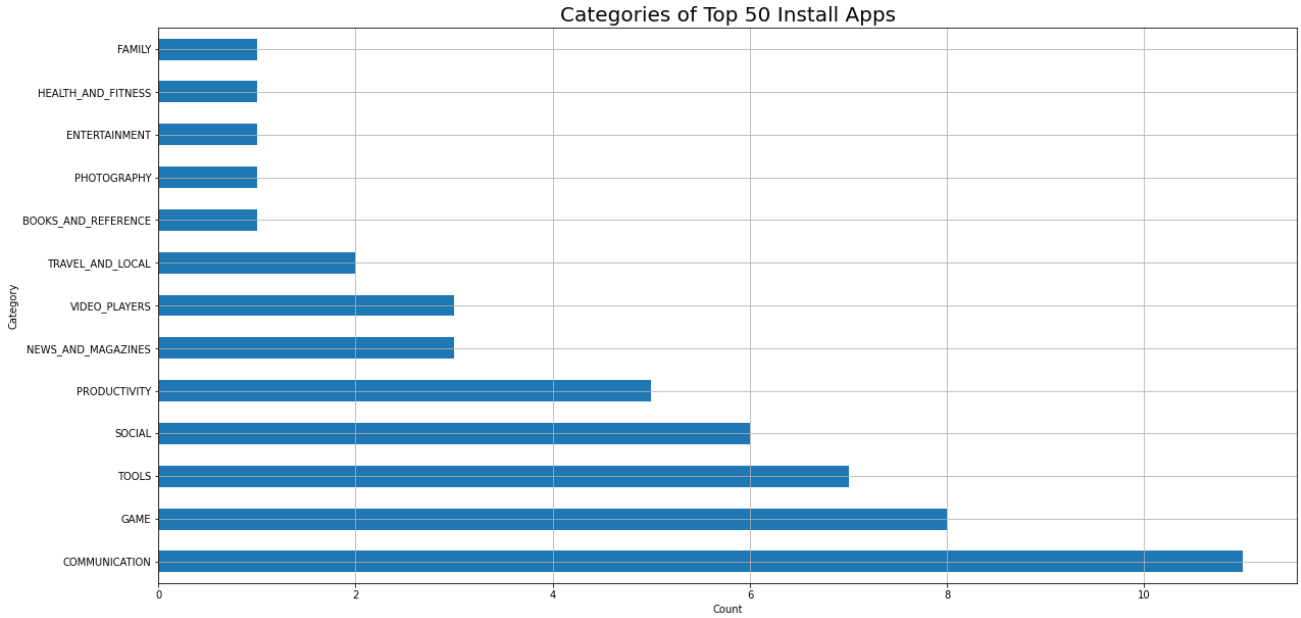
	App	Category	Rating	Reviews	Size	Installs	Typ
3736	Google News	NEWS_AND_MAGAZINES	3.9	877635	13.0	1000000000	Fre
3117	Maps - Navigate & Explore	TRAVEL_AND_LOCAL	4.3	9235155	12.0	1000000000	Fre
3127	Google Street View	TRAVEL_AND_LOCAL	4.2	2129689	12.0	1000000000	Fre
3665	YouTube	VIDEO_PLAYERS	4.3	25655305	12.0	1000000000	Fre
3687	Google Play Movies & TV	VIDEO_PLAYERS	3.7	906384	12.0	1000000000	Fre
2554	Google+	SOCIAL	4.2	4831125	12.0	1000000000	Fre
2545	Instagram	SOCIAL	4.5	66577313	12.0	1000000000	Fre
2544	Facebook	SOCIAL	4.1	78158306	12.0	1000000000	Fre
3234	Google	TOOLS	4.4	8033493	12.0	1000000000	Fre
152	Google Play Books	BOOKS_AND_REFERENCE	3.9	1433233	12.0	1000000000	Fre
341	Hangouts	COMMUNICATION	4.0	3419249	12.0	1000000000	Fre
3454	Google Drive	PRODUCTIVITY	4.4	2731171	12.0	1000000000	Fre
391	Skype - free IM & video calls	COMMUNICATION	4.1	10484169	12.0	1000000000	Fre
2808	Google Photos	PHOTOGRAPHY	4.5	10858556	12.0	1000000000	Fre
	Messenger						

Categories of Top 50 Install App's

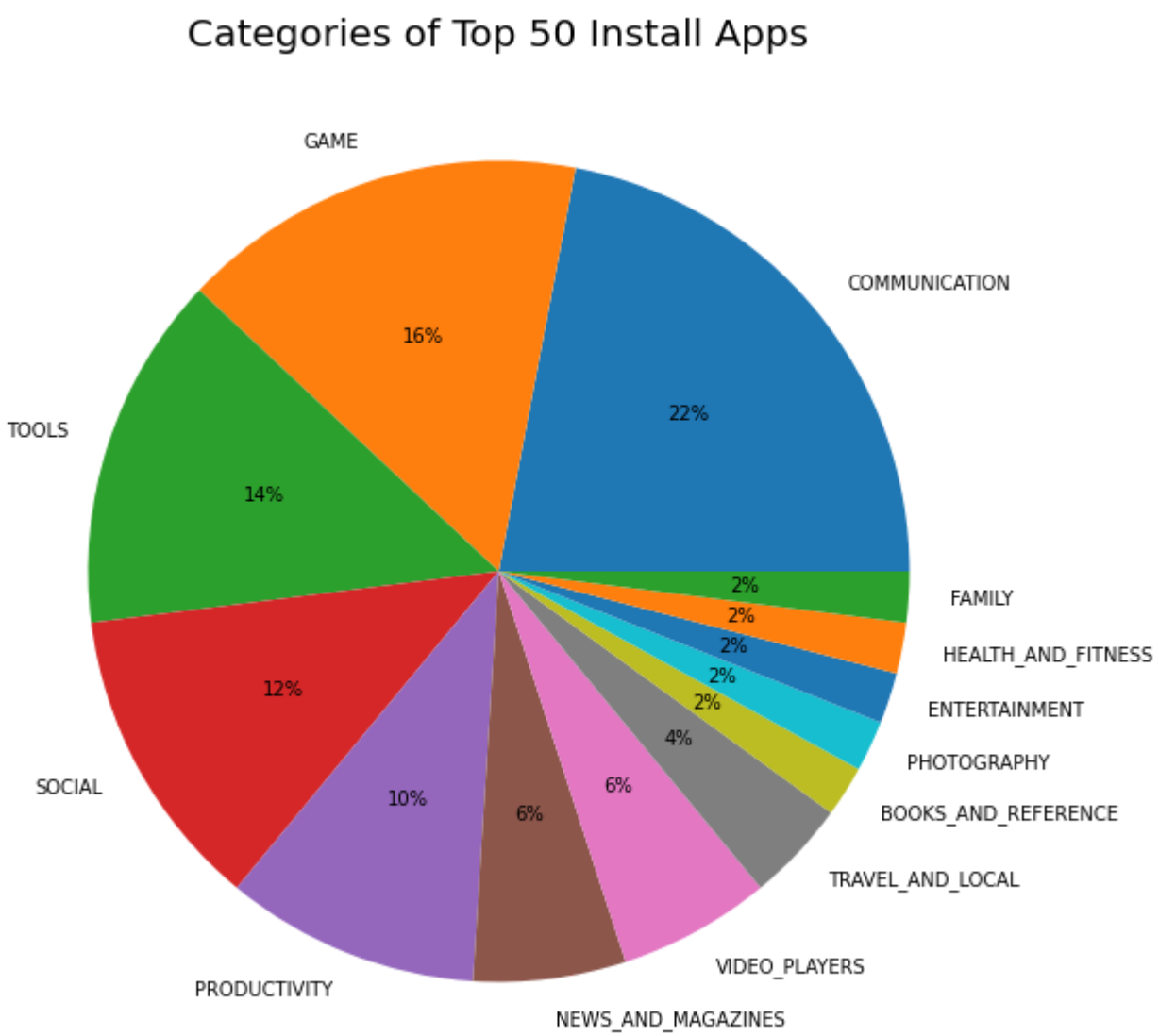
```
#Categories of Top 50 Install App's
category_of_Top50_installs_apps=Top50_installs_apps['Category'].value_counts().reset_inde
category_of_Top50_installs_apps
```

	Category	Count
0	COMMUNICATION	11
1	GAME	8
2	TOOLS	7
3	SOCIAL	6
4	PRODUCTIVITY	5
5	NEWS_AND_MAGAZINES	3
6	VIDEO_PLAYERS	3
7	TRAVEL_AND_LOCAL	2
8	BOOKS_AND_REFERENCE	1
9	PHOTOGRAPHY	1
10	ENTERTAINMENT	1
11	HEALTH_AND_FITNESS	1
12	FAMILY	1

```
#Categories of Top 50 Install Apps
Top50_installs_apps['Category'].value_counts().plot(kind="barh")
plt.title("Categories of Top 50 Install Apps",fontsize=20)
plt.ylabel("Category")
plt.xlabel("Count")
plt.grid()
```




```
#Categories of Top 50 Install Apps
plt.rcParams['figure.figsize'] = (10,10)
plt.pie(category_of_Top50_installs_apps['Count'],labels=category_of_Top50_installs_apps['
plt.title('Categories of Top 50 Install Apps',fontsize=20)
plt.show()
```



Among Top 50 install apps 22% belongs to the communication category and 16% belongs to the Game category.

Category And Rating

```
#Top 5 Categories by mean rating
Category_by_mean_rating=play_store_data.groupby('Category')['Rating'].mean().reset_index(
Category_by_mean_rating.head(5)
```

	Category	Mean Rating
10	EVENTS	4.395313
0	ART_AND_DESIGN	4.373016
8	EDUCATION	4.363866
3	BOOKS_AND_REFERENCE	4.334389
23	PERSONALIZATION	4.324866

Events Is Top Category By Mean Rating

Category And Installs

```
Top5_Category_by_no_of_install=play_store_data.groupby('Category')['Installs'].sum().rese
Top5_Category_by_no_of_install
```

	Category	Total Installs
14	GAME	13878924415
6	COMMUNICATION	11038276251
29	TOOLS	8001271905
25	PRODUCTIVITY	5793091369
27	SOCIAL	5487867902

Game Is The Category Having Maximum Installs

Genres And Installs

```
Top5_genres_by_no_of_install=play_store_data.groupby('Genres')['Installs'].sum().reset_in
Top5_genres_by_no_of_install
```

	Genres	Total Installs
35	Communication	11038276251
107	Tools	7991271905
81	Productivity	5793091369
100	Social	5487867902
80	Photography	4649147655

Communication Is The Genres Having Maximum Installs

Mean Rating of Top 5 Categories By No. Of Installs

```
#Mean Rating of Top 5 Categories By No. Of Installs
mean_rating_of_top5_categories_by_no_of_installs=pd.merge(Top5_Category_by_no_of_install,
mean_rating_of_top5_categories_by_no_of_installs
```

	Category	Total Installs	Mean Rating
0	GAME	13878924415	4.249948
1	COMMUNICATION	11038276251	4.154921
2	TOOLS	8001271905	4.073455
3	PRODUCTIVITY	5793091369	4.206150
4	SOCIAL	5487867902	4.255230

Category And Price

```
#Category by mean price
Category_by_mean_price=play_store_data.groupby('Category')['Price'].mean().reset_index(na
Category_by_mean_price.head(5)
```

	Category	Mean Price
12	FINANCE	8.408203
18	LIFESTYLE	6.398022
20	MEDICAL	2.520759
10	EVENTS	1.718594
11	FAMILY	1.312292

Finance Is The Category Having Highest Mean Price

Mean Price And Mean Rating of Top 5 Categories By No. Of Installs

```
#Mean Price And Mean Rating of Top 5 Categories By No. Of Installs
mean_price_of_top5_categories_by_no_of_installs=pd.merge(mean_rating_of_top5_categories_b
mean_price_of_top5_categories_by_no_of_installs
```

	Category	Total Installs	Mean Rating	Mean Price
0	GAME	13878924415	4.249948	0.296465
1	COMMUNICATION	11038276251	4.154921	0.263937
2	TOOLS	8001271905	4.073455	0.322739
3	PRODUCTIVITY	5793091369	4.206150	0.670936
4	SOCIAL	5487867902	4.255230	0.066820

Top 50 App's by Price

```
#Top50_apps_by_price
Top50_apps_by_price= play_store_data.sort_values(by=['Price'],ascending=False).head(50)
Top50_apps_by_price
```

	App	Category	Rating	Reviews	Size	Installs	Type	Pr
4367	I'm Rich - Trump Edition	LIFESTYLE	3.6	275	7.300	10000	Paid	400
5373	I AM RICH PRO PLUS	FINANCE	4.0	36	41.000	1000	Paid	399
5359	I am rich(premium)	FINANCE	3.5	472	0.965	5000	Paid	399
5351	I am rich	LIFESTYLE	3.8	3547	1.800	100000	Paid	399
5369	I am Rich	FINANCE	4.3	180	3.800	5000	Paid	399
4362	💎 I'm rich	LIFESTYLE	3.8	718	26.000	10000	Paid	399
5364	I am rich (Most expensive app)	FINANCE	4.1	129	2.700	1000	Paid	399
9934	I'm Rich/Eu sou Rico/أنا غني/我很有錢	LIFESTYLE	4.3	0	40.000	0	Paid	399
5362	I Am Rich Pro	FAMILY	4.4	201	2.700	5000	Paid	399
5358	I am Rich!	FINANCE	3.8	93	22.000	1000	Paid	399
5356	I Am Rich Premium	FINANCE	4.1	1867	4.700	50000	Paid	399
5354	I am Rich Plus	FAMILY	4.0	856	8.700	10000	Paid	399
4197	most expensive app (H)	FAMILY	4.3	6	1.500	100	Paid	399
9917	Eu Sou Rico	FINANCE	4.3	0	1.400	0	Paid	394
5366	I Am Rich	FAMILY	3.6	217	4.900	10000	Paid	389
5357	I am extremely Rich	LIFESTYLE	2.9	41	2.900	1000	Paid	379
5355	I am rich VIP	LIFESTYLE	3.8	411	2.600	10000	Paid	299
9719	EP Cook Book	MEDICAL	4.3	0	3.200	0	Paid	200
6692	cronometra-br	PRODUCTIVITY	4.3	0	5.400	0	Paid	154

Category Distribution Of Top 50 App's By Price

```
#Category Distribution Of Top 50 App's By Price
Top50_apps_by_price['Category'].value_counts().reset_index().rename(columns={'index':'Cat
```

	Category	Count
0	MEDICAL	12
1	FINANCE	10
2	FAMILY	10
3	LIFESTYLE	7
4	BUSINESS	3
5	PHOTOGRAPHY	2
6	PRODUCTIVITY	1
7	EVENTS	1
8	SPORTS	1
9	TOOLS	1
10	COMMUNICATION	1
11	GAME	1

Among Top 50 Apps By Price, Category Of Medical,Finance And Family Contribute The Most

Count Of Apps Of Free And Paid Categories

```
#Count_of_free_and_paid_categories Apps
no_of_apps_of_free_and_paid_categories= play_store_data.groupby(['Category','Type'])['App
no_of_apps_of_free_and_paid_categories.head(10)
```

	Category	Type	Count
0	ART_AND_DESIGN	Free	60
1	ART_AND_DESIGN	Paid	3
2	AUTO_AND_VEHICLES	Free	82
3	AUTO_AND_VEHICLES	Paid	3
4	BEAUTY	Free	53
5	BOOKS_AND_REFERENCE	Free	193
6	BOOKS_AND_REFERENCE	Paid	28
7	BUSINESS	Free	408
8	BUSINESS	Paid	12
9	COMICS	Free	56

```
#Count Of Maximum Paid Category Apps
no_of_apps_of_free_and_paid_categories[no_of_apps_of_free_and_paid_categories['Type']=='P
```

	Category	Type	Count
21	FAMILY	Paid	182
38	MEDICAL	Paid	83
27	GAME	Paid	82
44	PERSONALIZATION	Paid	80
56	TOOLS	Paid	77
6	BOOKS_AND_REFERENCE	Paid	28
48	PRODUCTIVITY	Paid	28
11	COMMUNICATION	Paid	27
54	SPORTS	Paid	24
34	LIFESTYLE	Paid	19
46	PHOTOGRAPHY	Paid	19
23	FINANCE	Paid	17
29	HEALTH_AND_FITNESS	Paid	15
8	BUSINESS	Paid	12
58	TRAVEL_AND_LOCAL	Paid	12
62	WEATHER	Paid	8
13	DATING	Paid	6
36	MAPS_AND_NAVIGATION	Paid	5
15	EDUCATION	Paid	4
60	VIDEO_PLAYERS	Paid	4
3	AUTO_AND_VEHICLES	Paid	3
52	SOCIAL	Paid	3
1	ART_AND_DESIGN	Paid	3
40	NEWS_AND_MAGAZINES	Paid	2
42	PARENTING	Paid	2
25	FOOD_AND_DRINK	Paid	2
50	SHOPPING	Paid	2
17	ENTERTAINMENT	Paid	2
32	LIBRARIES_AND_DEMO	Paid	1
19	EVENTS	Paid	1

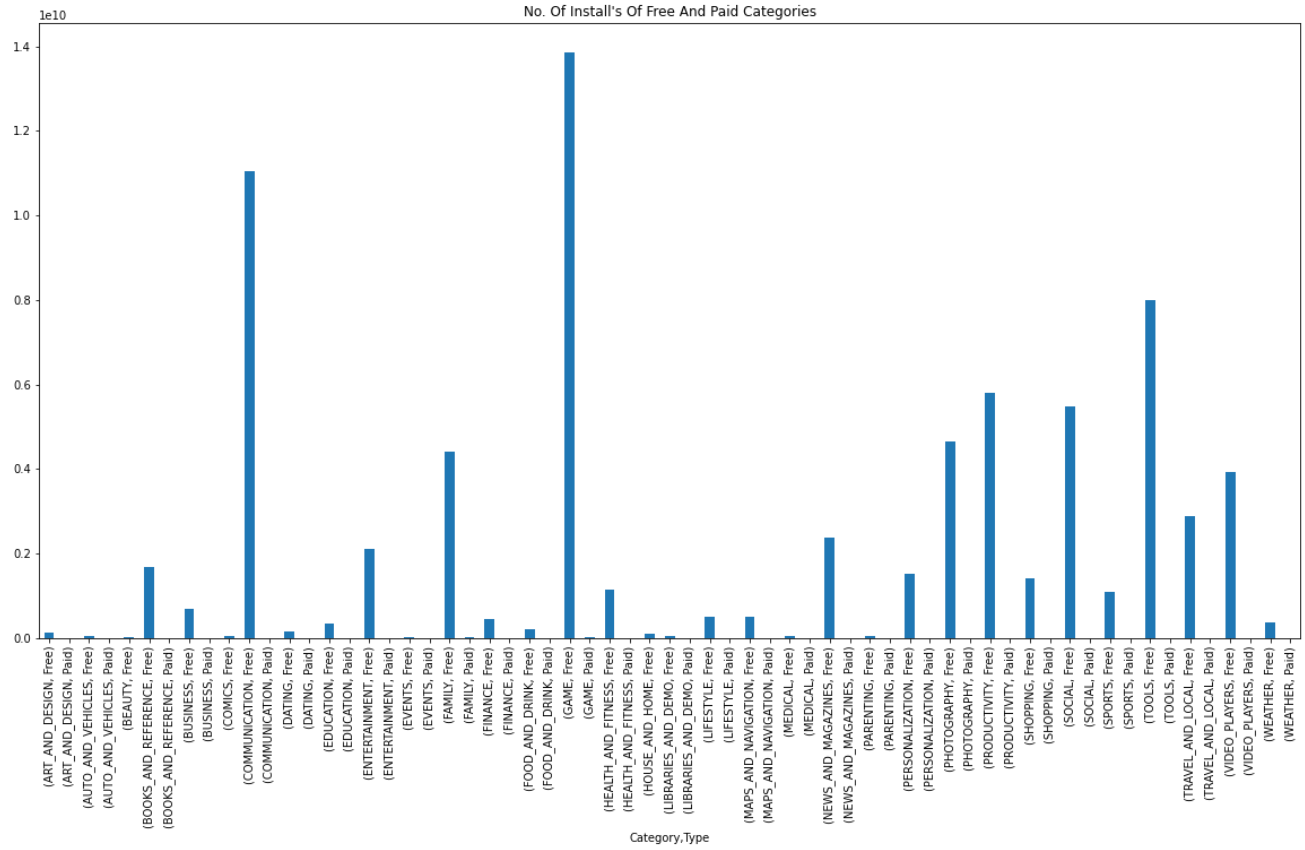
Family Category Have The Highest No. Of Paid Apps

No. Of Install's Of Free And Paid Categories

```
#No. Of Install's Of Free And Paid Categories
no_of_installs_of_free_and_paid_categories= play_store_data.groupby(['Category','Type'])[
no_of_installs_of_free_and_paid_categories['Installs1']=np.log2(no_of_installs_of_free_an
```

```
#No. Of Install's Of Free And Paid Categories
plt.figure(figsize = (20,10))
play_store_data.groupby(['Category','Type'])['Installs'].sum().plot(kind="bar")
plt.title("No. Of Install's Of Free And Paid Categories")
```

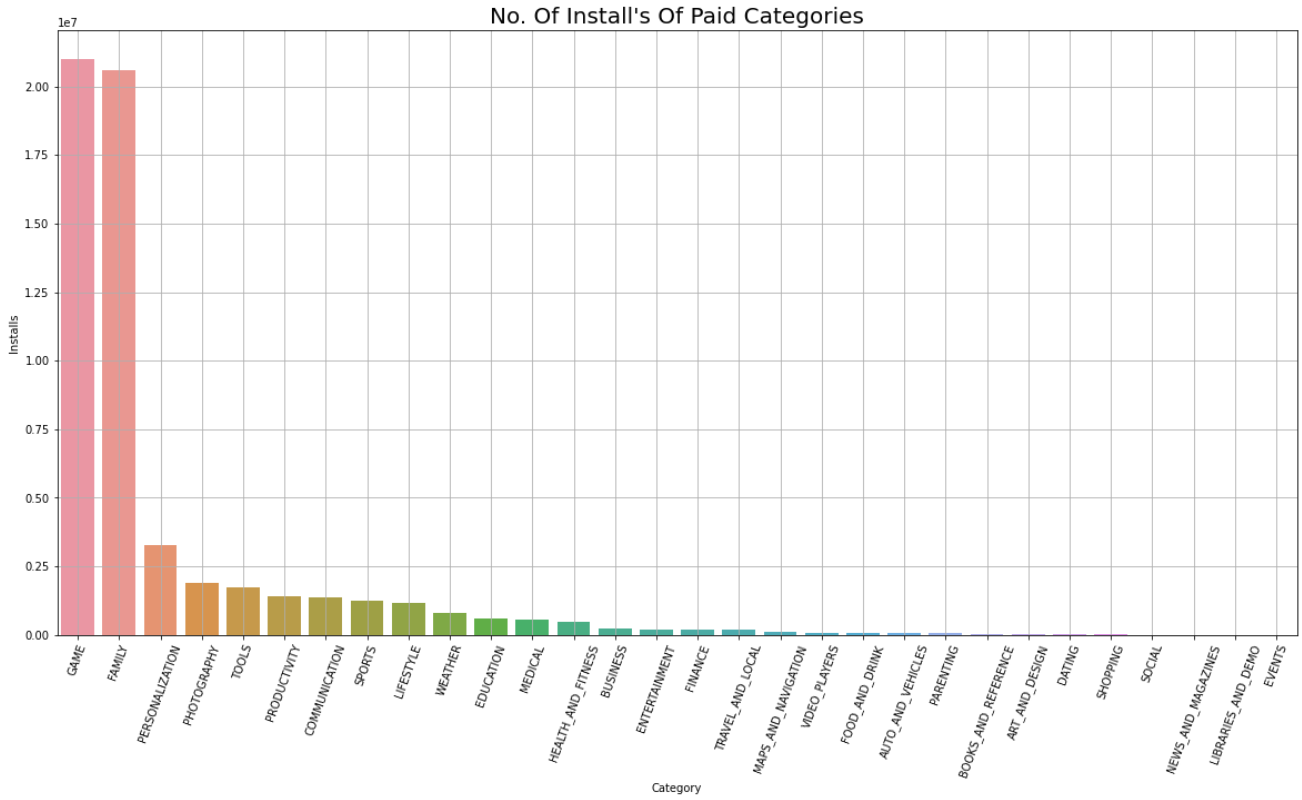
```
Text(0.5, 1.0, "No. Of Install's Of Free And Paid Categories")
```




```
#Category And No. Of Paid Install Apps
sorteddf=no_of_installs_of_free_and_paid_categories[no_of_installs_of_free_and_paid_categ
sorteddf
```

	Category	Type	Installs	Installs1
27	GAME	Paid	20999965	24.323884
21	FAMILY	Paid	20599814	24.296128
44	PERSONALIZATION	Paid	3257794	21.635464
46	PHOTOGRAPHY	Paid	1878740	20.841334
56	TOOLS	Paid	1727431	20.720197
48	PRODUCTIVITY	Paid	1412055	20.429365
11	COMMUNICATION	Paid	1360050	20.375228
54	SPORTS	Paid	1243815	20.246340
34	LIFESTYLE	Paid	1179110	20.169267
62	WEATHER	Paid	812000	19.631120
15	EDUCATION	Paid	602000	19.199404
38	MEDICAL	Paid	560833	19.097212
29	HEALTH_AND_FITNESS	Paid	474110	18.854862
8	BUSINESS	Paid	212775	17.698969
17	ENTERTAINMENT	Paid	200000	17.609640
23	FINANCE	Paid	185602	17.501853
58	TRAVEL_AND_LOCAL	Paid	183060	17.481957
36	MAPS_AND_NAVIGATION	Paid	121100	16.885839
60	VIDEO_PLAYERS	Paid	71000	16.115531
25	FOOD_AND_DRINK	Paid	60000	15.872675
3	AUTO_AND_VEHICLES	Paid	50150	15.613962
42	PARENTING	Paid	50100	15.612523
6	BOOKS_AND_REFERENCE	Paid	23316	14.509033
1	ART_AND_DESIGN	Paid	16000	13.965784
13	DATING	Paid	11350	13.470405
50	SHOPPING	Paid	10100	13.302068
52	SOCIAL	Paid	6000	12.550747
40	NEWS_AND_MAGAZINES	Paid	5500	12.425216
32	LIBRARIES_AND_DEMO	Paid	100	6.643856
19	EVENTS	Paid	1	0.000000

```
#No. Of Install's  Paid Categories
plt.figure(figsize = (20,10))
sns.barplot(x="Category", y="Installs", data=sorteddf)
plt.title("No. Of Install's Of Paid Categories",fontsize=20)
plt.xticks(rotation=70, horizontalalignment="center")
plt.grid()
```



Game And Family Category Have The Highest No. Of Installs Of Paid Apps

App's With 5 Star Rating

```
#App's With 5 Star Rating
five_star_rating_apps=play_store_data[play_store_data['Rating']==5]
five_star_rating_apps['App'].nunique()
```

```
271
```

Top 10 Categories Of 5 Star Rating App's

```
#Top 10 Categories Of 5 Star Rating App's
five_star_rating_apps['Category'].value_counts().reset_index().rename(columns={'index':'C
```

	Category	Count
0	FAMILY	67
1	LIFESTYLE	29
2	MEDICAL	25
3	BUSINESS	18
4	TOOLS	17
5	GAME	12
6	HEALTH_AND_FITNESS	12
7	PERSONALIZATION	10
8	SOCIAL	8
9	PRODUCTIVITY	8

There are 271 apps which have 5 star rating. In which 67 apps belongs to the Family category.

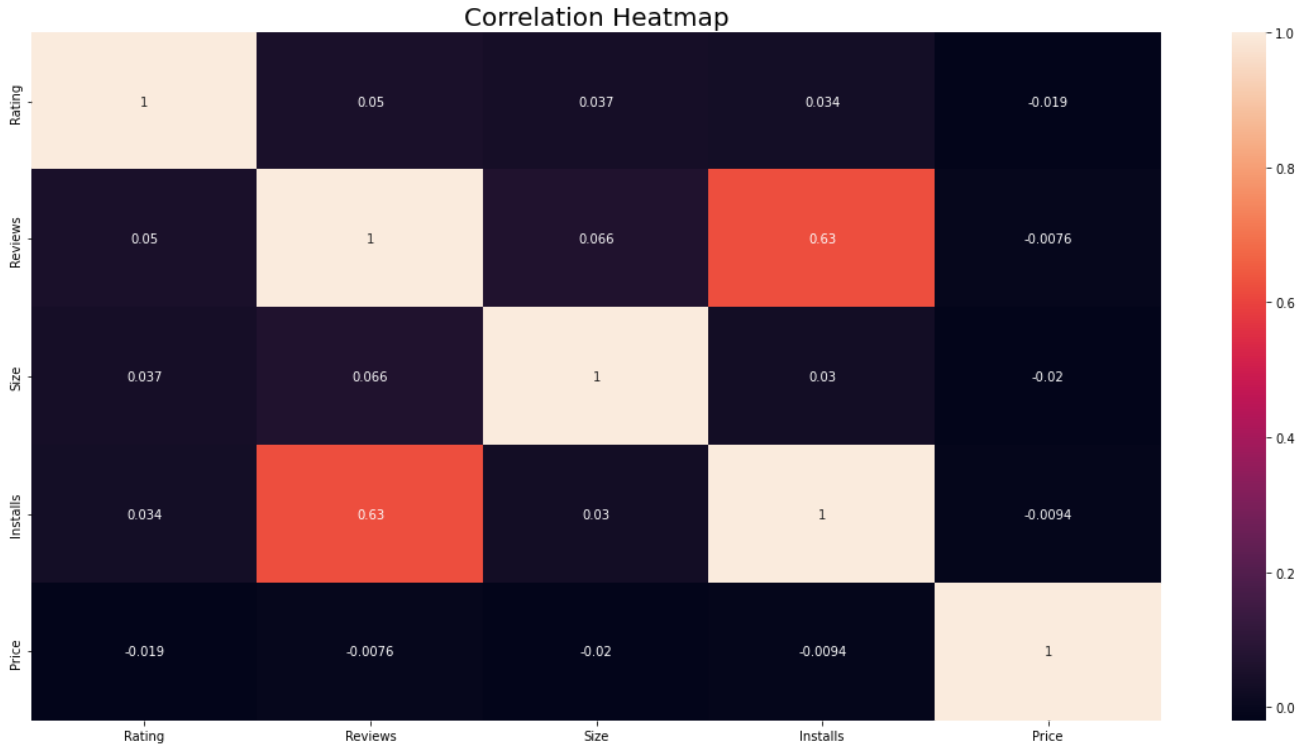
Correlation Heatmap

```
#Correlation
play_store_data.corr()
```

	Rating	Reviews	Size	Installs	Price
Rating	1.000000	0.050215	0.037383	0.034310	-0.018673
Reviews	0.050215	1.000000	0.066147	0.625158	-0.007604
Size	0.037383	0.066147	1.000000	0.030467	-0.019590
Installs	0.034310	0.625158	0.030467	1.000000	-0.009413
Price	-0.018673	-0.007604	-0.019590	-0.009413	1.000000

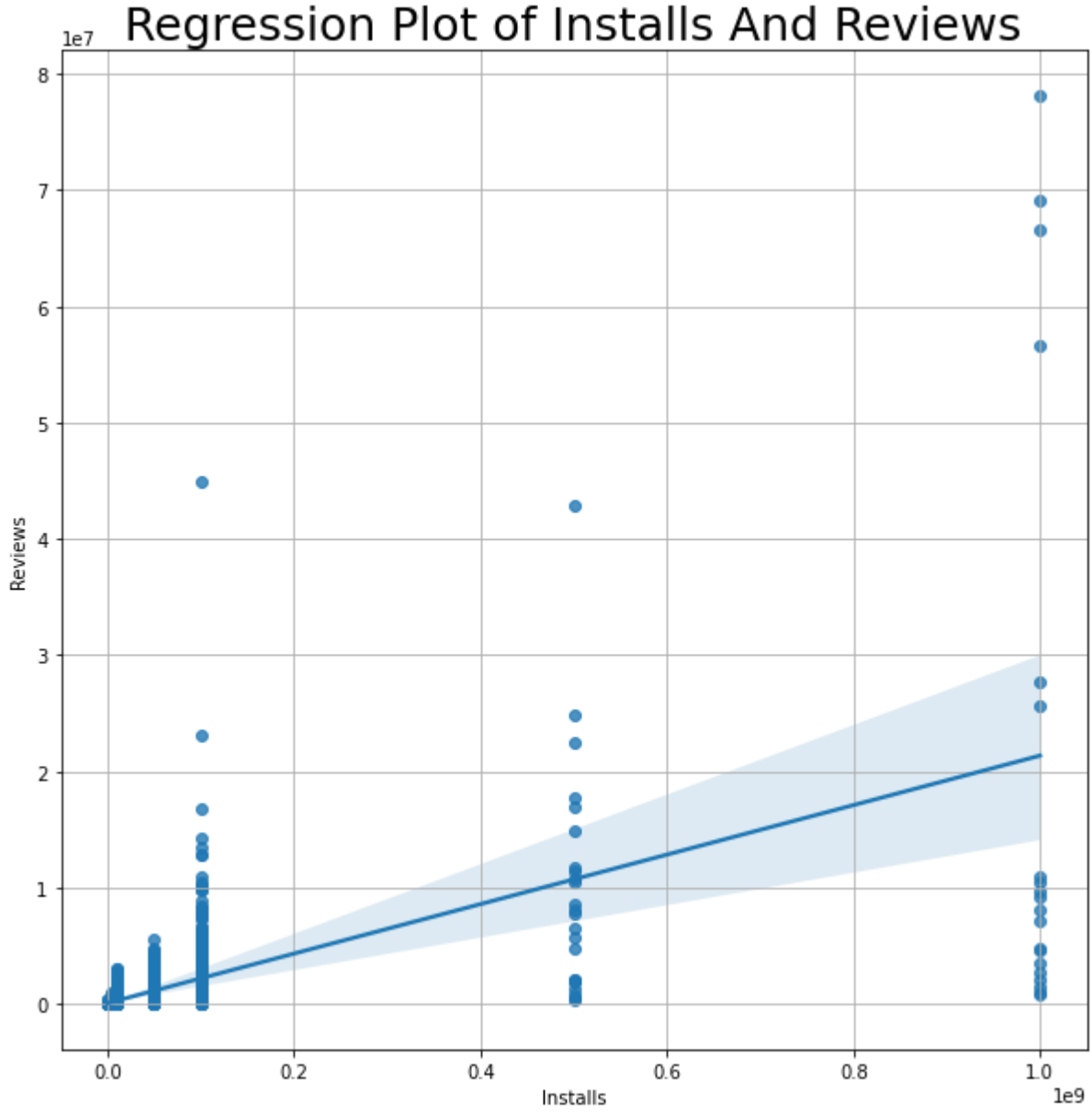
```
#Correlation Heatmap
plt.figure(figsize = (20,10))
sns.heatmap(play_store_data.corr(), annot= True)
plt.title("Correlation Heatmap",fontsize=20)
```

```
Text(0.5, 1.0, 'Correlation Heatmap')
```



Regression Plot Of Installs And Reviews

```
#Regression Plot Of Installs And Reviews
sns.regplot(x="Installs",y="Reviews",data=play_store_data)
plt.title("Regression Plot of Installs And Reviews",fontsize=25)
plt.grid()
```



There is some significant amount of positive correlation between Installs and Reviews. This is expected as no. of Installs increases more interactions will happen which leads to increasing no. of Reviews.

Let's Drive Into The User Reviews Data

```
#First Look Of User Reviews Data
user_reviews_data.head()
```

	App	Translated_Review	Sentiment	Sentiment_Polarity	Sentiment_Subjectivity
0	10 Best Foods for You	I like eat delicious food. That's I'm cooking food myself, case "10 Best Foods" helps lot, also "Best Before (Shelf Life)"	Positive	1.00	0.533333
1	10 Best Foods for You	This help eating healthy exercise regular basis	Positive	0.25	0.288462
	10 Best				

```
user_reviews_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64295 entries, 0 to 64294
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   App                     64295 non-null  object 
 1   Translated_Review       37427 non-null  object 
 2   Sentiment               37432 non-null  object 
 3   Sentiment_Polarity      37432 non-null  float64
 4   Sentiment_Subjectivity  37432 non-null  float64
dtypes: float64(2), object(3)
memory usage: 2.5+ MB
```

Handling missing values

```
#data of Translated null values
user_reviews_data[user_reviews_data['Translated_Review'].isnull()].head()
```

	App	Translated_Review	Sentiment	Sentiment_Polarity	Sentiment_Subjectivity
2	10 Best Foods for You	NaN	NaN	NaN	NaN
	10 Best				

```
#Dropping Nulls of Translated Review as all other feature values are also null
user_reviews_data.dropna(subset=['Translated_Review'],inplace=True)
```

```
#Checking data after removing nulls
user_reviews_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 37427 entries, 0 to 64230
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   App                     37427 non-null  object 
 1   Translated_Review       37427 non-null  object 
 2   Sentiment               37427 non-null  object 
 3   Sentiment_Polarity      37427 non-null  float64
 4   Sentiment_Subjectivity  37427 non-null  float64
dtypes: float64(2), object(3)
memory usage: 1.7+ MB
```

Combining Both Dataset

```
#unique apps is user_reviews_data
user_reviews_data['App'].nunique()
```

```
865
```

```
#unique apps is play_store_data
play_store_data['App'].nunique()
```

```
9648
```

```
#Merging both data
combined_data=pd.merge(play_store_data,user_reviews_data, on='App')
```

```
#Unique Apps in combined data
combined_data['App'].nunique()
```

```
816
```

```
#About combined data
combined_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 35929 entries, 0 to 35928
Data columns (total 17 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   App                     35929 non-null  object 
 1   Category                35929 non-null  object 
 2   Rating                  35929 non-null  float64
 3   Reviews                 35929 non-null  int64  
 4   Size                    35929 non-null  float64
 5   Installs                35929 non-null  int64  
 6   Type                    35929 non-null  object 
```

```
 7   Price                   35929 non-null  float64
 8   Content Rating          35929 non-null  object
 9   Genres                  35929 non-null  object
 10  Last Updated            35929 non-null  object
 11  Current Ver             35929 non-null  object
 12  Android Ver             35929 non-null  object
 13  Translated_Review       35929 non-null  object
 14  Sentiment               35929 non-null  object
 15  Sentiment_Polarity      35929 non-null  float64
 16  Sentiment_Subjectivity  35929 non-null  float64
dtypes: float64(5), int64(2), object(10)
memory usage: 4.9+ MB
```

```
#Checking for null in combined data
combined_data.isnull().any().sum()
```

```
0
```

Analysis Of Combined Data

Translated Reviews

Most Frequent Words In Translated Reviews

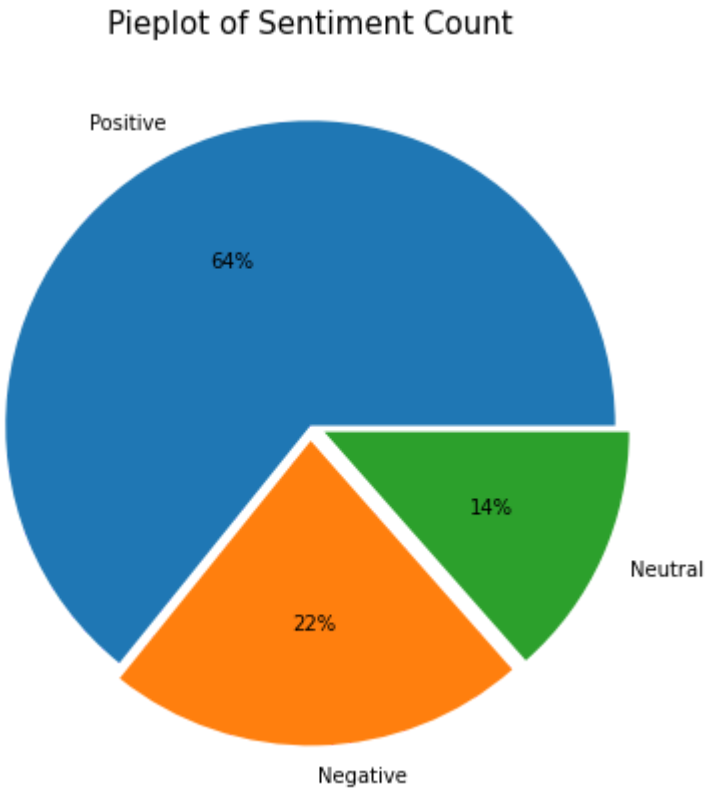
```
#Importing worcloud
from wordcloud import WordCloud, STOPWORDS

comment_words = ''
stopwords = set(STOPWORDS)
for val in combined_data.Translated_Review:
    # typecaste each val to string
    val = str(val)
    # split the value
    tokens = val.split()
    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    comment_words += " ".join(tokens)+" "

wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.title("Most Frequent Words In Translated Reviews",fontsize=15)
plt.tight_layout(pad = 0)

plt.show()
```

Sentiment

```
#Sentiment count
sentiment_count= combined_data.Sentiment.value_counts().reset_index().rename(columns={'in
sentiment_count
```

	Sentiment	count
0	Positive	23073
1	Negative	8005
2	Neutral	4851

```
#Pieplot of Sentiment Count
plt.rcParams['figure.figsize'] = (15, 7)
plt.pie(sentiment_count['count'],labels=sentiment_count['Sentiment'],autopct='%.0f%%',exp
plt.title('Pieplot of Sentiment Count',size=15)
plt.show()
```




```
#Appwise sentiment count
Appwise_sentiment_count=combined_data.groupby(['App','Sentiment'])['Sentiment'].count().r
Appwise_sentiment_count.head()
```

	App	Sentiment	count
0	10 Best Foods for You	Negative	10
1	10 Best Foods for You	Neutral	22
2	10 Best Foods for You	Positive	162
3	11st	Negative	7
4	11st	Neutral	9

```
#Top20 apps with most positive sentiment
Top_apps_with_max_sentiment=Appwise_sentiment_count.sort_values(by="count",ascending=Fals
Top20_apps_with_most_positive_sentiment=Top_apps_with_max_sentiment[Top_apps_with_max_sen
Top20_apps_with_most_positive_sentiment
```

	App	Sentiment	count
2115	Helix Jump	Positive	209
1279	Duolingo: Learn Languages Free	Positive	200
824	Calorie Counter - Macros	Positive	174
664	Bowmasters	Positive	169
827	Calorie Counter - MyFitnessPal	Positive	169
2	10 Best Foods for You	Positive	162
1951	Google Photos	Positive	143
54	8fit Workouts & Meal Planner	Positive	137
1845	Garena Free Fire	Positive	136
1103	DRAGON BALL LEGENDS	Positive	127
298	Angry Birds Classic	Positive	124
749	CBS Sports App - Scores, News, Stats & Watch Live	Positive	123
966	ColorNote Notepad Notes	Positive	121
830	Calorie Counter - MyNetDiary	Positive	120

Helix Jump App Have The Most Positive Sentiment Count

818	Calorie Counter & Diet Tracker	Positive	109

Categories of Top20 Apps With Most Positive Sentiment

1021	Crunchyroll - Everything Anime	Positive	105

```
#merging 2 dataset Top20_apps_with_most_positive_sentiment  and combined_data
d1=pd.merge(Top20_apps_with_most_positive_sentiment,combined_data, on='App')
categories_of_Top20_apps_with_most_positive_sentiment=d1[['App','Sentiment_x','count','Ca
d2=categories_of_Top20_apps_with_most_positive_sentiment.groupby(['App','Sentiment_x','Ca
d2['Category'].value_counts().reset_index().rename(columns={'index':'Category','Category'
```

	Category	Count