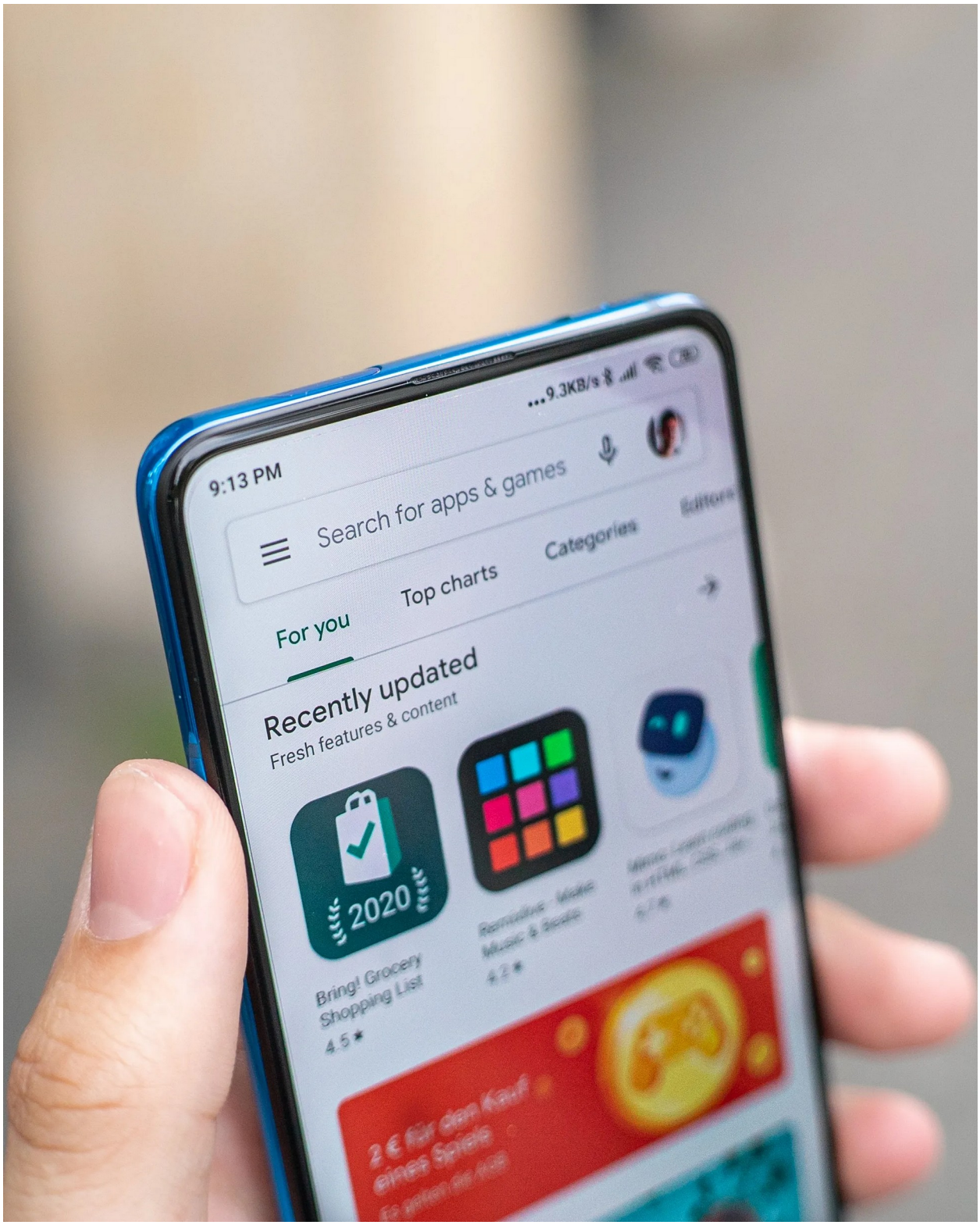
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('ggplot')
```



1.pandas: Pandas is a powerful library for data manipulation and analysis that provides data structures like DataFrames, which allow you to efficiently work with structured data. Pandas offers a wide range of functions and methods to handle tasks such as data cleaning,

transformation, aggregation, and exploration. It is particularly useful for working with tabular data, making it an essential library for data analysis tasks.

2.numpy: NumPy is a fundamental library for numerical computing in Python. It provides a high-performance multidimensional array object, along with tools for working with arrays. NumPy is extensively used for mathematical operations, array manipulation, linear algebra, random number generation, and more. It is a core library in the Python scientific computing ecosystem and often used in conjunction with pandas for efficient data manipulation and analysis.

3.matplotlib.pyplot: Matplotlib is a widely used plotting library in Python. The pyplot module provides a simple interface to create various types of plots, including line plots, scatter plots, bar plots, histograms, and more. It allows you to customize every aspect of a plot, such as labels, titles, colors, and styles. Matplotlib is versatile and can be used for basic exploratory data visualization as well as creating publication-quality figures.

4.seaborn: Seaborn is a statistical data visualization library built on top of Matplotlib. It provides a higher-level interface for creating visually appealing and informative statistical graphics. Seaborn simplifies the process of creating complex plots such as heatmaps, violin plots, box plots, and regression plots. It also offers various built-in color palettes and themes to enhance the aesthetics of your visualizations.

5.plt.style.use: This line of code sets the plot style to 'ggplot', which mimics the aesthetics of the popular ggplot2 library in R. The 'ggplot' style provides a clean and professional look to your plots with a gray background, gridlines, and well-defined color schemes. It helps to create visually appealing and easily interpretable plots.

These libraries together form a powerful toolkit for data analysis and visualization in Python. They are extensively used in the data science community and provide a wide range of functionality to handle diverse data analysis tasks.

```
df =
pd.read_csv("https://raw.githubusercontent.com/jasonchang0/kaggle-google-apps/master/google-play-store-apps/googleplaystore.csv")
```

The code snippet pd.read_csv("playstore.csv") uses the pandas library to read a CSV (Comma-Separated Values) file named "playstore.csv" and assigns the resulting data to a DataFrame called df. The DataFrame is a two-dimensional tabular data structure in pandas, similar to a table in a relational database. The read_csv() function is a convenient way to read CSV files and convert them into a DataFrame, allowing for easy manipulation, exploration, and analysis of the data contained in the file. Once the CSV file is read and stored in the DataFrame df, you can perform various operations and analyses on the data using pandas and other libraries.

Understanding the dataset

Then let's see how the data looks using the head() function

```
df.head()
```

```
                                                 App        Category
Rating  \
0     Photo Editor & Candy Camera & Grid & ScrapBook  ART_AND_DESIGN
4.1
1                                 Coloring book moana  ART_AND_DESIGN
3.9
2  U Launcher Lite – FREE Live Cool Themes, Hide ...  ART_AND_DESIGN
4.7
3                             Sketch - Draw & Paint  ART_AND_DESIGN
4.5
4              Pixel Draw - Number Art Coloring Book  ART_AND_DESIGN
4.3

  Reviews  Size     Installs  Type Price Content Rating  \
0     159   19M      10,000+  Free     0       Everyone
1     967   14M     500,000+  Free     0       Everyone
2   87510  8.7M   5,000,000+  Free     0       Everyone
3  215644   25M  50,000,000+  Free     0           Teen
4     967  2.8M     100,000+  Free     0       Everyone

                      Genres      Last Updated         Current Ver  \
0               Art & Design   January 7, 2018               1.0.0
1  Art & Design;Pretend Play  January 15, 2018               2.0.0
2               Art & Design    August 1, 2018               1.2.4
3               Art & Design      June 8, 2018  Varies with device
4    Art & Design;Creativity     June 20, 2018                 1.1

    Android Ver
0  4.0.3 and up
1  4.0.3 and up
2  4.0.3 and up
3    4.2 and up
4    4.4 and up
```

First, let's start by understanding the shape of our database.

```
df.shape #tells us the shape of the data
```

```
(10841, 13)
```

```
df.columns
```

```
Index(['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs',
'Type',
       'Price', 'Content Rating', 'Genres', 'Last Updated', 'Current
Ver',
       'Android Ver'],
      dtype='object')
```

```
df.isnull().sum()
```

```
App                    0
Category               0
Rating              1474
Reviews                0
Size                   0
Installs               0
Type                   1
Price                  0
Content Rating         1
Genres                 0
Last Updated           0
Current Ver            8
Android Ver            3
dtype: int64
```

```
df.dropna(subset=['Rating', 'Type', 'Content Rating', 'Current Ver',
'Android Ver'], inplace=True)
```

```
df.isnull().sum()
```

```
App                0
Category           0
Rating             0
Reviews            0
Size               0
Installs           0
Type               0
Price              0
Content Rating     0
Genres             0
Last Updated       0
Current Ver        0
Android Ver        0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1325514 entries, 0 to 2312942
Data columns (total 24 columns):
 #   Column              Non-Null Count    Dtype
---  ------              --------------    -----
 0   App Name            1325513 non-null  object
 1   App Id              1325514 non-null  object
 2   Category            1325514 non-null  object
 3   Rating              1325514 non-null  float64
 4   Rating Count        1325514 non-null  float64
 5   Installs            1325514 non-null  object
 6   Minimum Installs    1325514 non-null  float64
 7   Maximum Installs    1325514 non-null  int64
```

```
 8   Free                1325514 non-null  bool
 9   Price               1325514 non-null  float64
 10  Currency            1325501 non-null  object
 11  Size                1325514 non-null  object
 12  Minimum Android     1321880 non-null  object
 13  Developer Id        1325491 non-null  object
 14  Developer Website   1325514 non-null  object
 15  Developer Email     1325511 non-null  object
 16  Released            1290745 non-null  object
 17  Last Updated        1325514 non-null  object
 18  Content Rating      1325514 non-null  object
 19  Privacy Policy      1325514 non-null  object
 20  Ad Supported        1325514 non-null  bool
 21  In App Purchases    1325514 non-null  bool
 22  Editors Choice      1325514 non-null  bool
 23  Scraped Time        1325514 non-null  object
dtypes: bool(4), float64(4), int64(1), object(15)
memory usage: 217.4+ MB
```

```
df["Reviews"].describe()
```

```
count     9360
unique    5990
top          2
freq        83
Name: Reviews, dtype: object
```

```
df["Reviews"] = df["Reviews"].astype("int64")
```

```
df["Reviews"].describe().round()
```

```
count        9360.0
mean       514377.0
std       3145023.0
min             1.0
25%           187.0
50%          5955.0
75%         81628.0
max      78158306.0
Name: Reviews, dtype: float64
```

```
print(len(df["Size"].unique()))
df["Size"].unique()
```

```
413
```

```
array(['19M', '14M', '8.7M', '25M', '2.8M', '5.6M', '29M', '33M',
'3.1M',
       '28M', '12M', '20M', '21M', '37M', '5.5M', '17M', '39M', '31M',
       '4.2M', '23M', '6.0M', '6.1M', '4.6M', '9.2M', '5.2M', '11M',
       '24M', 'Varies with device', '9.4M', '15M', '10M', '1.2M',
```

```
'26M',
       '8.0M', '7.9M', '56M', '57M', '35M', '54M', '201k', '3.6M',
'5.7M',
       '8.6M', '2.4M', '27M', '2.7M', '2.5M', '7.0M', '16M', '3.4M',
       '8.9M', '3.9M', '2.9M', '38M', '32M', '5.4M', '18M', '1.1M',
       '2.2M', '4.5M', '9.8M', '52M', '9.0M', '6.7M', '30M', '2.6M',
       '7.1M', '22M', '6.4M', '3.2M', '8.2M', '4.9M', '9.5M', '5.0M',
       '5.9M', '13M', '73M', '6.8M', '3.5M', '4.0M', '2.3M', '2.1M',
       '42M', '9.1M', '55M', '23k', '7.3M', '6.5M', '1.5M', '7.5M',
'51M',
       '41M', '48M', '8.5M', '46M', '8.3M', '4.3M', '4.7M', '3.3M',
'40M',
       '7.8M', '8.8M', '6.6M', '5.1M', '61M', '66M', '79k', '8.4M',
       '3.7M', '118k', '44M', '695k', '1.6M', '6.2M', '53M', '1.4M',
       '3.0M', '7.2M', '5.8M', '3.8M', '9.6M', '45M', '63M', '49M',
'77M',
       '4.4M', '70M', '9.3M', '8.1M', '36M', '6.9M', '7.4M', '84M',
'97M',
       '2.0M', '1.9M', '1.8M', '5.3M', '47M', '556k', '526k', '76M',
       '7.6M', '59M', '9.7M', '78M', '72M', '43M', '7.7M', '6.3M',
'334k',
       '93M', '65M', '79M', '100M', '58M', '50M', '68M', '64M', '34M',
       '67M', '60M', '94M', '9.9M', '232k', '99M', '624k', '95M',
'8.5k',
       '41k', '292k', '80M', '1.7M', '10.0M', '74M', '62M', '69M',
'75M',
       '98M', '85M', '82M', '96M', '87M', '71M', '86M', '91M', '81M',
       '92M', '83M', '88M', '704k', '862k', '899k', '378k', '4.8M',
       '266k', '375k', '1.3M', '975k', '980k', '4.1M', '89M', '696k',
       '544k', '525k', '920k', '779k', '853k', '720k', '713k', '772k',
       '318k', '58k', '241k', '196k', '857k', '51k', '953k', '865k',
       '251k', '930k', '540k', '313k', '746k', '203k', '26k', '314k',
       '239k', '371k', '220k', '730k', '756k', '91k', '293k', '17k',
       '74k', '14k', '317k', '78k', '924k', '818k', '81k', '939k',
'169k',
       '45k', '965k', '90M', '545k', '61k', '283k', '655k', '714k',
'93k',
       '872k', '121k', '322k', '976k', '206k', '954k', '444k', '717k',
       '210k', '609k', '308k', '306k', '175k', '350k', '383k', '454k',
       '1.0M', '70k', '812k', '442k', '842k', '417k', '412k', '459k',
       '478k', '335k', '782k', '721k', '430k', '429k', '192k', '460k',
       '728k', '496k', '816k', '414k', '506k', '887k', '613k', '778k',
       '683k', '592k', '186k', '840k', '647k', '373k', '437k', '598k',
       '716k', '585k', '982k', '219k', '55k', '323k', '691k', '511k',
       '951k', '963k', '25k', '554k', '351k', '27k', '82k', '208k',
       '551k', '29k', '103k', '116k', '153k', '209k', '499k', '173k',
       '597k', '809k', '122k', '411k', '400k', '801k', '787k', '50k',
       '643k', '986k', '516k', '837k', '780k', '20k', '498k', '600k',
       '656k', '221k', '228k', '176k', '34k', '259k', '164k', '458k',
```

```
       '629k', '28k', '288k', '775k', '785k', '636k', '916k', '994k',
       '309k', '485k', '914k', '903k', '608k', '500k', '54k', '562k',
       '847k', '948k', '811k', '270k', '48k', '523k', '784k', '280k',
       '24k', '892k', '154k', '18k', '33k', '860k', '364k', '387k',
       '626k', '161k', '879k', '39k', '170k', '141k', '160k', '144k',
       '143k', '190k', '376k', '193k', '473k', '246k', '73k', '253k',
       '957k', '420k', '72k', '404k', '470k', '226k', '240k', '89k',
       '234k', '257k', '861k', '467k', '676k', '552k', '582k',
'619k'],
      dtype=object)
```

```
df["Size"].replace("M","", regex=True, inplace = True)
df["Size"].replace("k","", regex=True, inplace = True)
```

```
df["Size"].unique()
```

```
array(['19', '14', '8.7', '25', '2.8', '5.6', '29', '33', '3.1', '28',
       '12', '20', '21', '37', '5.5', '17', '39', '31', '4.2', '23',
       '6.0', '6.1', '4.6', '9.2', '5.2', '11', '24',
       'Varies with device', '9.4', '15', '10', '1.2', '26', '8.0',
'7.9',
       '56', '57', '35', '54', '201', '3.6', '5.7', '8.6', '2.4',
'27',
       '2.7', '2.5', '7.0', '16', '3.4', '8.9', '3.9', '2.9', '38',
'32',
       '5.4', '18', '1.1', '2.2', '4.5', '9.8', '52', '9.0', '6.7',
'30',
       '2.6', '7.1', '22', '6.4', '3.2', '8.2', '4.9', '9.5', '5.0',
       '5.9', '13', '73', '6.8', '3.5', '4.0', '2.3', '2.1', '42',
'9.1',
       '55', '7.3', '6.5', '1.5', '7.5', '51', '41', '48', '8.5',
'46',
       '8.3', '4.3', '4.7', '3.3', '40', '7.8', '8.8', '6.6', '5.1',
'61',
       '66', '79', '8.4', '3.7', '118', '44', '695', '1.6', '6.2',
'53',
       '1.4', '3.0', '7.2', '5.8', '3.8', '9.6', '45', '63', '49',
'77',
       '4.4', '70', '9.3', '8.1', '36', '6.9', '7.4', '84', '97',
'2.0',
       '1.9', '1.8', '5.3', '47', '556', '526', '76', '7.6', '59',
'9.7',
       '78', '72', '43', '7.7', '6.3', '334', '93', '65', '100', '58',
       '50', '68', '64', '34', '67', '60', '94', '9.9', '232', '99',
       '624', '95', '292', '80', '1.7', '10.0', '74', '62', '69',
'75',
       '98', '85', '82', '96', '87', '71', '86', '91', '81', '92',
'83',
       '88', '704', '862', '899', '378', '4.8', '266', '375', '1.3',
       '975', '980', '4.1', '89', '696', '544', '525', '920', '779',
```

```
       '853', '720', '713', '772', '318', '241', '196', '857', '953',
       '865', '251', '930', '540', '313', '746', '203', '314', '239',
       '371', '220', '730', '756', '293', '317', '924', '818', '939',
       '169', '965', '90', '545', '283', '655', '714', '872', '121',
       '322', '976', '206', '954', '444', '717', '210', '609', '308',
       '306', '175', '350', '383', '454', '1.0', '812', '442', '842',
       '417', '412', '459', '478', '335', '782', '721', '430', '429',
       '192', '460', '728', '496', '816', '414', '506', '887', '613',
       '778', '683', '592', '186', '840', '647', '373', '437', '598',
       '716', '585', '982', '219', '323', '691', '511', '951', '963',
       '554', '351', '208', '551', '103', '116', '153', '209', '499',
       '173', '597', '809', '122', '411', '400', '801', '787', '643',
       '986', '516', '837', '780', '498', '600', '656', '221', '228',
       '176', '259', '164', '458', '629', '288', '775', '785', '636',
       '916', '994', '309', '485', '914', '903', '608', '500', '562',
       '847', '948', '811', '270', '523', '784', '280', '892', '154',
       '860', '364', '387', '626', '161', '879', '170', '141', '160',
       '144', '143', '190', '376', '193', '473', '246', '253', '957',
       '420', '404', '470', '226', '240', '234', '257', '861', '467',
       '676', '552', '582', '619'], dtype=object)
```

```
size_median = df[df["Size"]!="Varies with device"]
["Size"].astype(float).median()
df["Size"].replace("Varies with device", size_median, inplace=True)
df.Size = pd.to_numeric(df.Size)
df.Size.head()
```

```
0    19.0
1    14.0
2     8.7
3    25.0
4     2.8
Name: Size, dtype: float64
```

```
df.Size.describe().round()
```

```
count    9360.0
mean       34.0
std        85.0
min         1.0
25%         8.0
50%        16.0
75%        30.0
max       994.0
Name: Size, dtype: float64
```

```
#Installs
df["Installs"].unique()
```

```
array(['10,000+', '500,000+', '5,000,000+', '50,000,000+', '100,000+',
       '50,000+', '1,000,000+', '10,000,000+', '5,000+',
```

```
'100,000,000+',
       '1,000,000,000+', '1,000+', '500,000,000+', '100+', '500+',
'10+',
       '5+', '50+', '1+'], dtype=object)
```

```
df.Installs = df.Installs.apply(lambda x:x.replace("+",""))
df.Installs = df.Installs.apply(lambda x:x.replace(",",""))
df.Installs = df.Installs.apply(lambda x:int(x))
```

```
df["Installs"].unique()
```

```
array([     10000,     500000,    5000000,   50000000,     100000,
           50000,    1000000,   10000000,       5000,  100000000,
      1000000000,       1000,  500000000,        100,        500,
              10,          5,         50,          1], dtype=int64)
```

```
#Price
df["Price"].unique()
```

```
array(['0', '$4.99', '$3.99', '$6.99', '$7.99', '$5.99', '$2.99',
'$3.49',
       '$1.99', '$9.99', '$7.49', '$0.99', '$9.00', '$5.49', '$10.00',
       '$24.99', '$11.99', '$79.99', '$16.99', '$14.99', '$29.99',
       '$12.99', '$2.49', '$10.99', '$1.50', '$19.99', '$15.99',
'$33.99',
       '$39.99', '$3.95', '$4.49', '$1.70', '$8.99', '$1.49', '$3.88',
       '$399.99', '$17.99', '$400.00', '$3.02', '$1.76', '$4.84',
'$4.77',
       '$1.61', '$2.50', '$1.59', '$6.49', '$1.29', '$299.99',
'$379.99',
       '$37.99', '$18.99', '$389.99', '$8.49', '$1.75', '$14.00',
'$2.00',
       '$3.08', '$2.59', '$19.40', '$3.90', '$4.59', '$15.46',
'$3.04',
       '$13.99', '$4.29', '$3.28', '$4.60', '$1.00', '$2.95', '$2.90',
       '$1.97', '$2.56', '$1.20'], dtype=object)
```

```
df.Price = df.Price.apply(lambda x:x.replace("$",""))
df.Price = df.Price.apply(lambda x:float(x))
```

```
df["Price"].unique()
```

```
array([  0.  ,   4.99,   3.99,   6.99,   7.99,   5.99,   2.99,   3.49,
         1.99,   9.99,   7.49,   0.99,   9.  ,   5.49,  10.  ,  24.99,
        11.99,  79.99,  16.99,  14.99,  29.99,  12.99,   2.49,  10.99,
         1.5 ,  19.99,  15.99,  33.99,  39.99,   3.95,   4.49,   1.7 ,
         8.99,   1.49,   3.88, 399.99,  17.99, 400.  ,   3.02,   1.76,
         4.84,   4.77,   1.61,   2.5 ,   1.59,   6.49,   1.29, 299.99,
       379.99,  37.99,  18.99, 389.99,   8.49,   1.75,  14.  ,   2.  ,
         3.08,   2.59,  19.4 ,   3.9 ,   4.59,  15.46,   3.04,  13.99,
```

```
        4.29,   3.28,   4.6 ,   1.  ,   2.95,   2.9 ,   1.97,   2.56,
        1.2 ])
```

```
df["Genres"] = df["Genres"].str.split(";").str[0]
```

```
df["Genres"].replace("Music & Audio", "Music", inplace =True)
```

```
df["Last Updated"] = pd.to_datetime(df["Last Updated"])
```

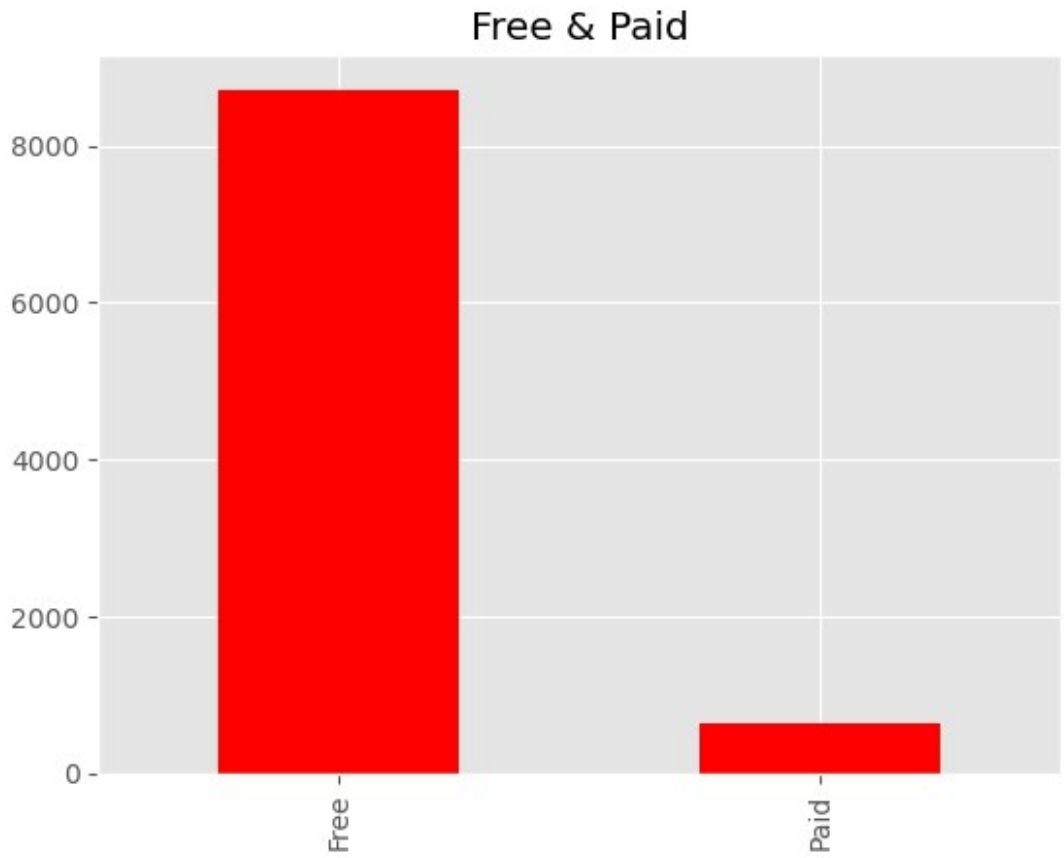
```
df.head()
```

```
                                                 App        Category
Rating  \
0     Photo Editor & Candy Camera & Grid & ScrapBook  ART_AND_DESIGN
4.1
1                                Coloring book moana  ART_AND_DESIGN
3.9
2  U Launcher Lite – FREE Live Cool Themes, Hide ...  ART_AND_DESIGN
4.7
3                             Sketch - Draw & Paint  ART_AND_DESIGN
4.5
4              Pixel Draw - Number Art Coloring Book  ART_AND_DESIGN
4.3

   Reviews  Size  Installs  Type  Price Content Rating
Genres  \
0      159  19.0     10000  Free    0.0       Everyone  Art & Design

1      967  14.0    500000  Free    0.0       Everyone  Art & Design

2    87510   8.7   5000000  Free    0.0       Everyone  Art & Design

3   215644  25.0  50000000  Free    0.0           Teen  Art & Design

4      967   2.8    100000  Free    0.0       Everyone  Art & Design


  Last Updated         Current Ver   Android Ver
0   2018-01-07               1.0.0  4.0.3 and up
1   2018-01-15               2.0.0  4.0.3 and up
2   2018-08-01               1.2.4  4.0.3 and up
3   2018-06-08  Varies with device    4.2 and up
4   2018-06-20                 1.1    4.4 and up
```

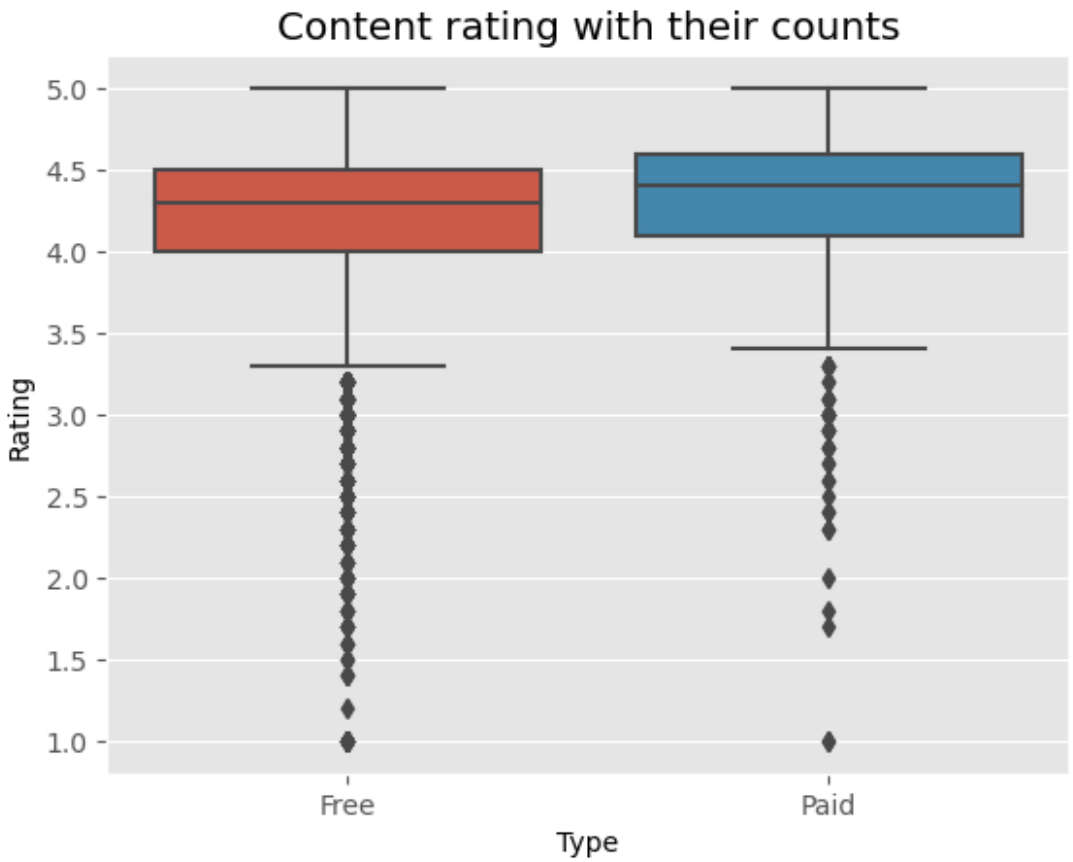
```
#Plot Bar Graph for the different types of apps
```

```
df["Type"].value_counts().plot(kind="bar", color ="red")
plt.title("Free & Paid")
```

```
Text(0.5, 1.0, 'Free & Paid')
```

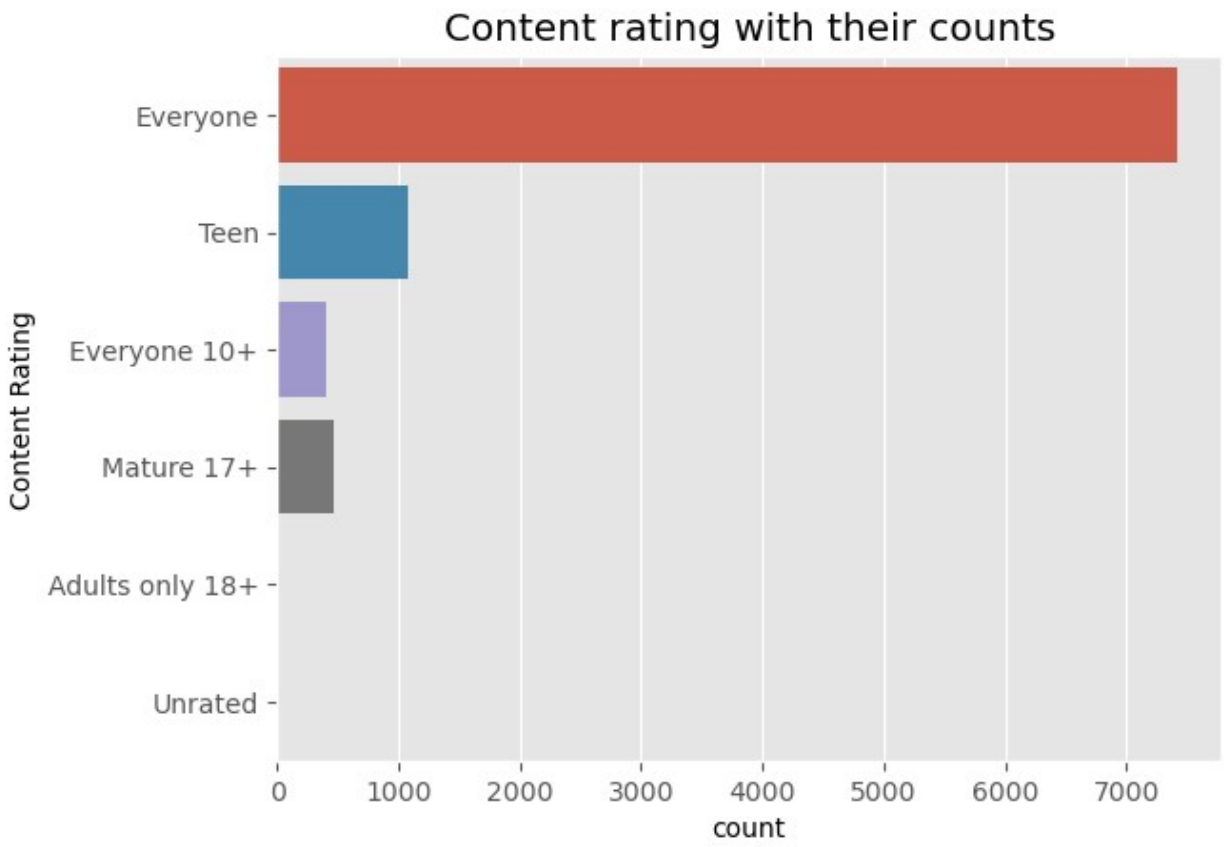



```
sns.boxplot(x = "Type", y = "Rating", data = df)
plt.title("Content rating with their counts")

Text(0.5, 1.0, 'Content rating with their counts')
```



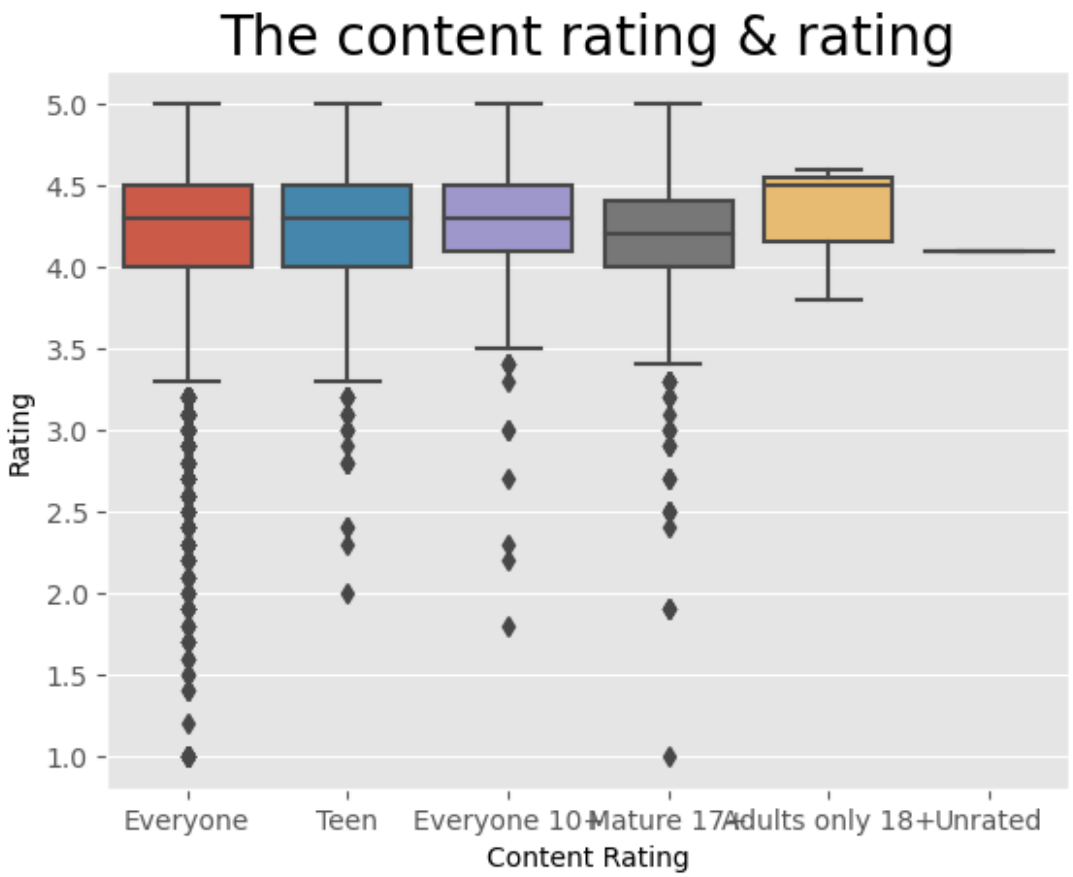

```
#Bar plot for the rating column with the number of the categories
sns.countplot(y = "Content Rating", data = df)
plt.title("Content rating with their counts")
```

```
Text(0.5, 1.0, 'Content rating with their counts')
```



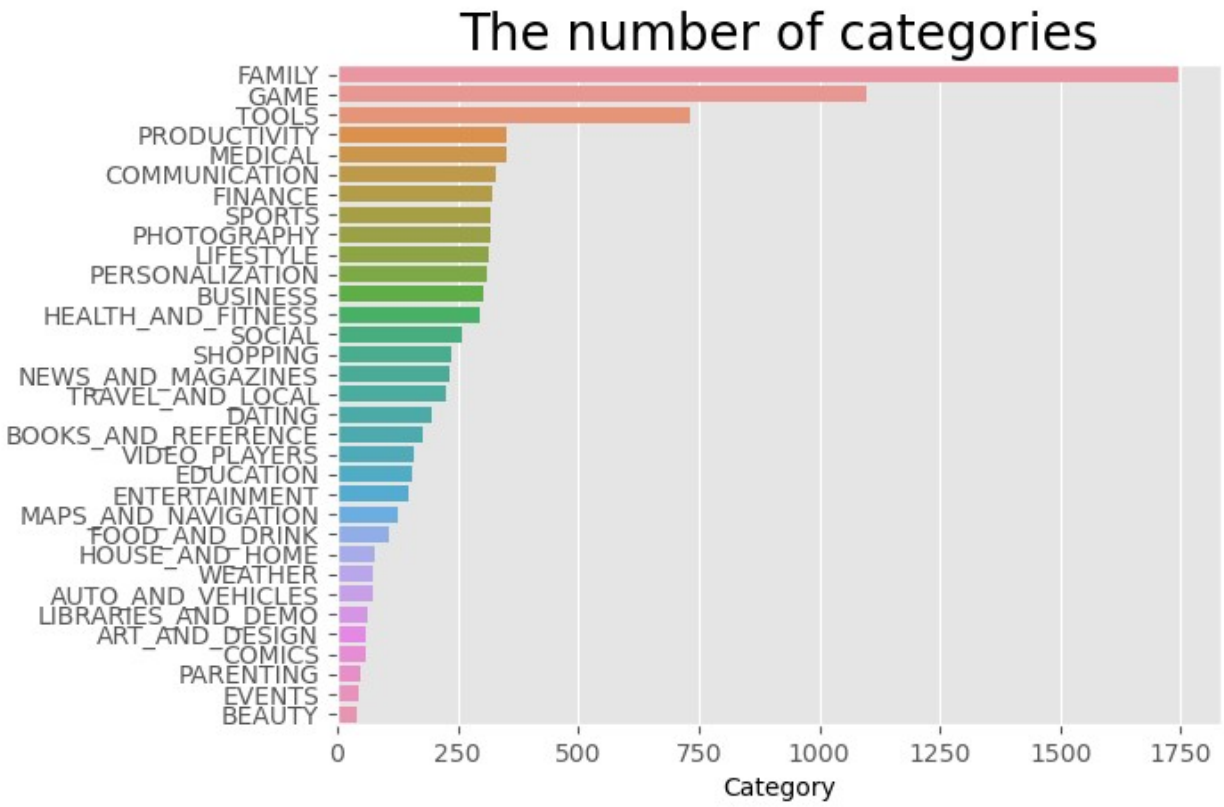

```
sns.boxplot(x = "Content Rating", y = "Rating", data = df)
plt.title("The content rating & rating", size=20)
```

```
Text(0.5, 1.0, 'The content rating & rating')
```

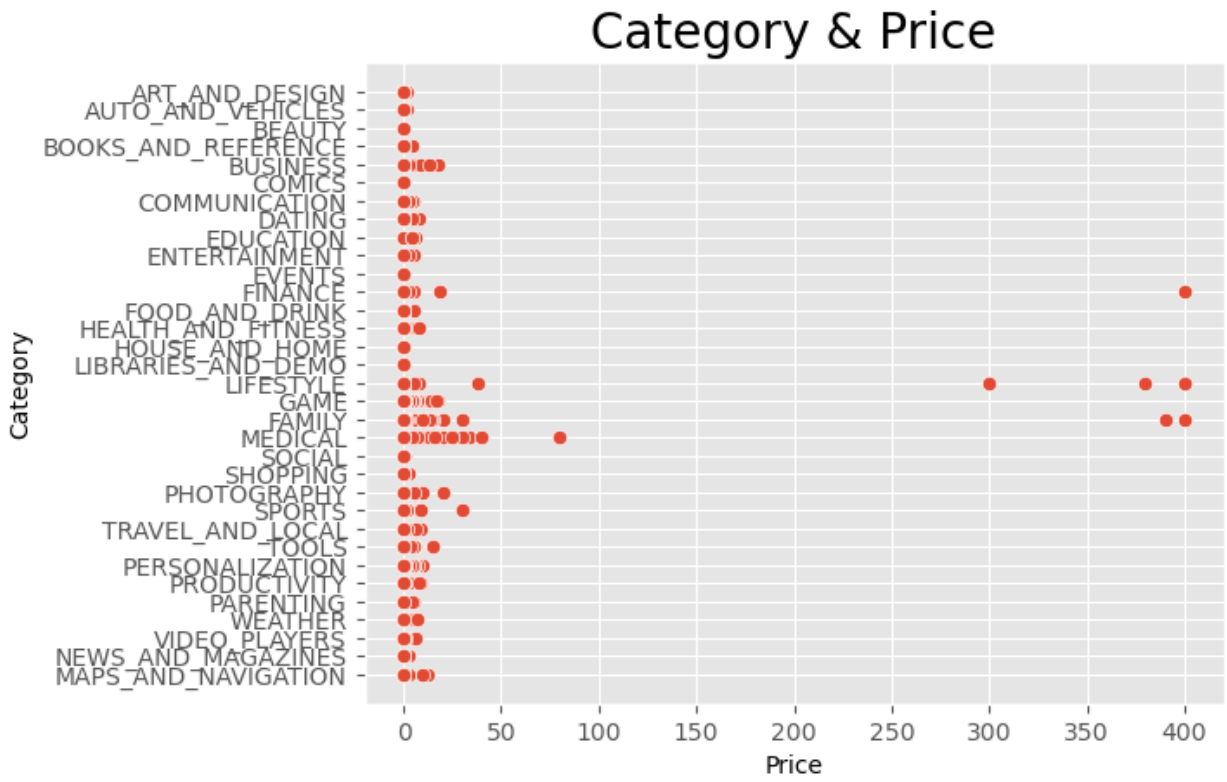



```
cat_num = df["Category"].value_counts()
sns.barplot(x = cat_num, y = cat_num.index, data = df)
plt.title("The number of categories", size=20)
```

```
Text(0.5, 1.0, 'The number of categories')
```

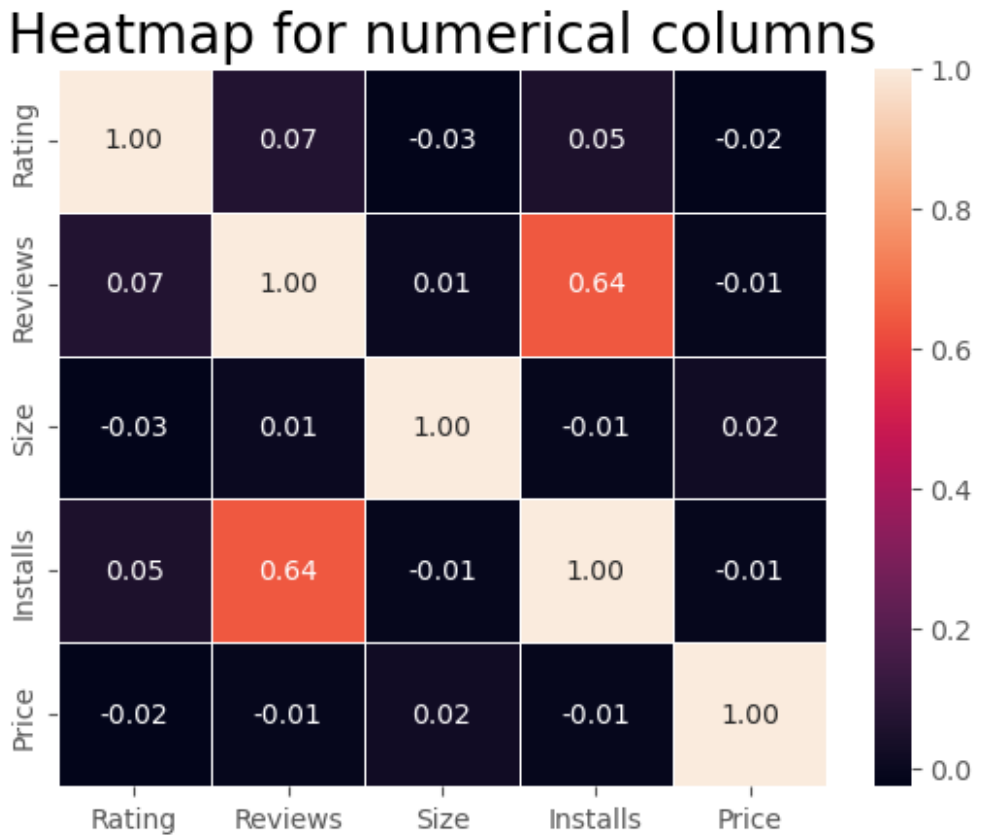



```
sns.scatterplot(data = df, y = "Category", x = "Price")
plt.title("Category & Price", size=20)

Text(0.5, 1.0, 'Category & Price')
```



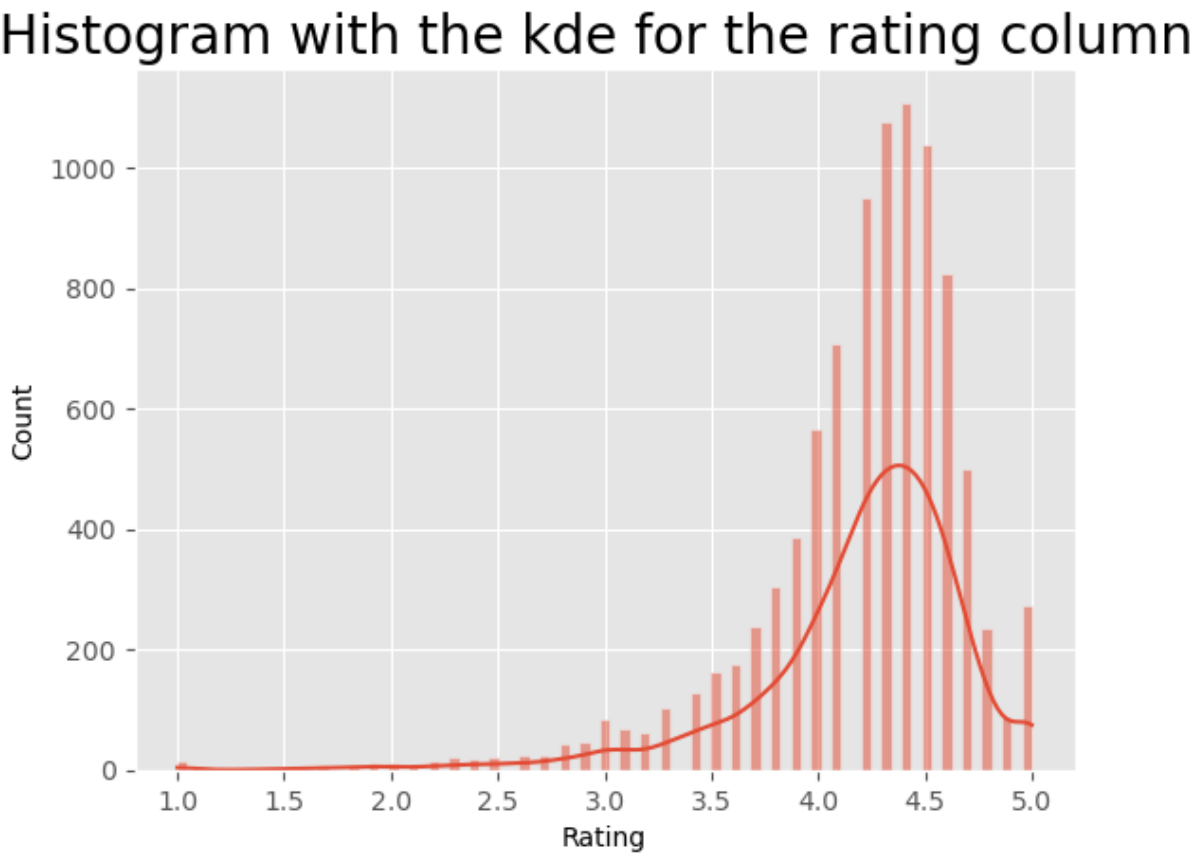

```
#Heat map for numerical columns
sns.heatmap(df.corr(), annot = True, linewidths=.5, fmt=".2f")
plt.title("Heatmap for numerical columns", size=20)
```

```
C:\Users\syedt\AppData\Local\Temp\ipykernel_15588\622182597.py:2:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only
valid columns or specify the value of numeric_only to silence this
warning.
  sns.heatmap(df.corr(), annot = True, linewidths=.5, fmt=".2f")

Text(0.5, 1.0, 'Heatmap for numerical columns')
```



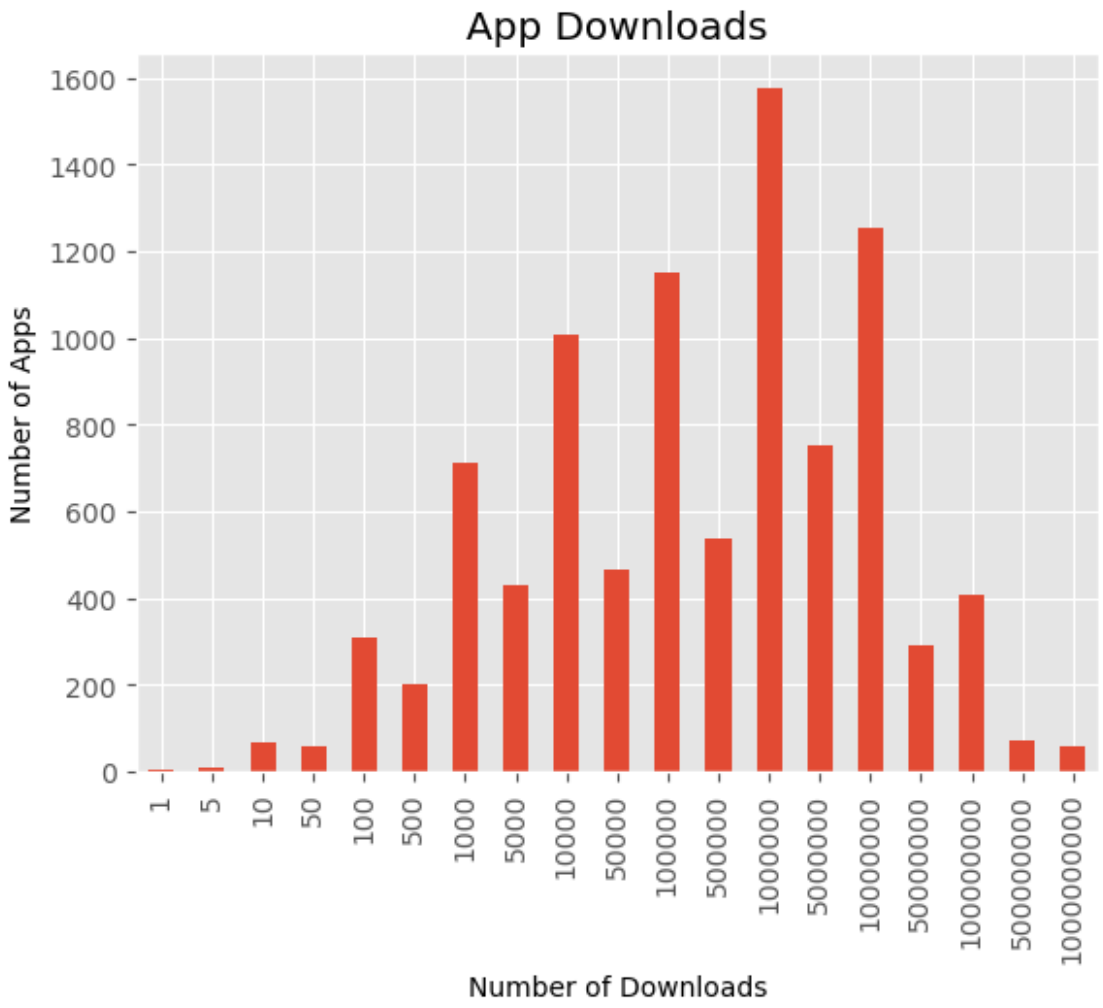

```
#histogram with kde for the ratings
sns.histplot(df["Rating"], kde = True)
plt.title("Histogram with the kde for the rating column ", size=20,)
```

```
Text(0.5, 1.0, 'Histogram with the kde for the rating column ')
```




```
ax = df['Installs'].value_counts().sort_index(ascending=True) \
    .plot(kind='bar', title='App Downloads')

ax.set_xlabel('Number of Downloads')
ax.set_ylabel('Number of Apps')
```

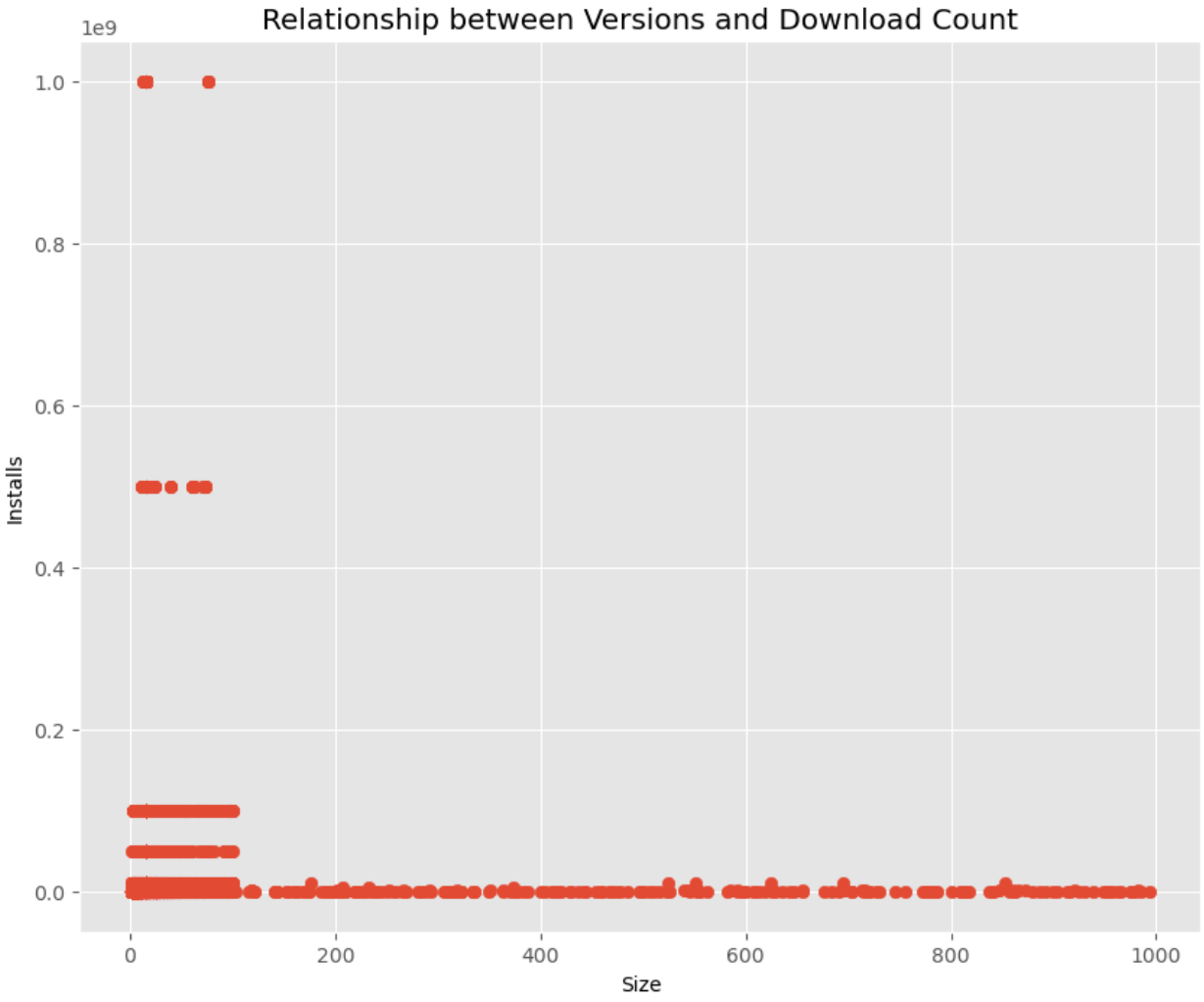
```
Text(0, 0.5, 'Number of Apps')
```




```
from wordcloud import WordCloud
category_counts = df['Category'].value_counts()

# Create a text with category frequencies
text = ' '.join([f'{category}: {count}' for category, count in
category_counts.items()])

# Create a WordCloud object
wordcloud = WordCloud(background_color='white', width=800,
height=400).generate_from_text(text)

# Set the figure size
plt.figure(figsize=(10, 5))

# Create the word bubble using scatter plot
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
```

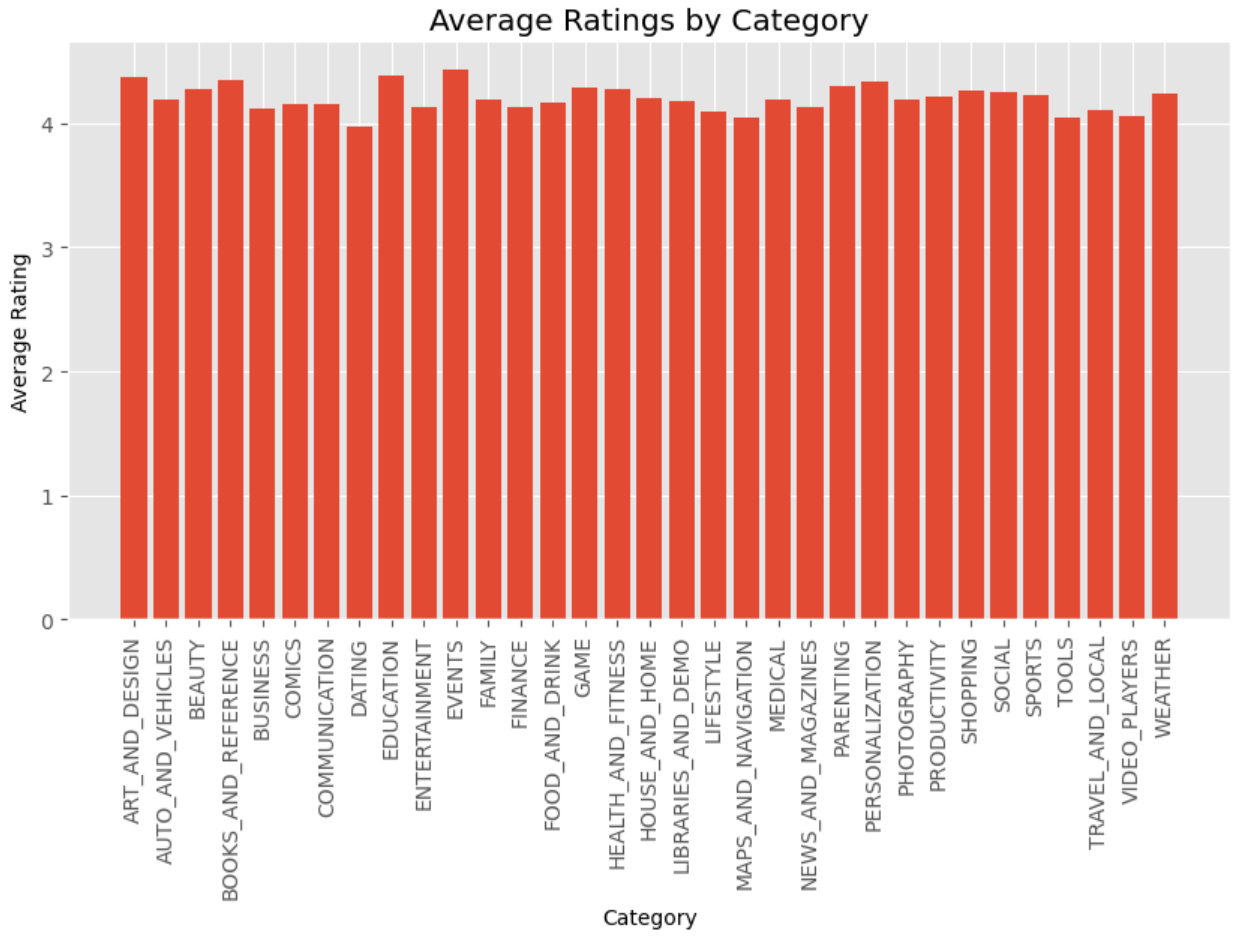
```
# Show the word bubble
plt.show()
```

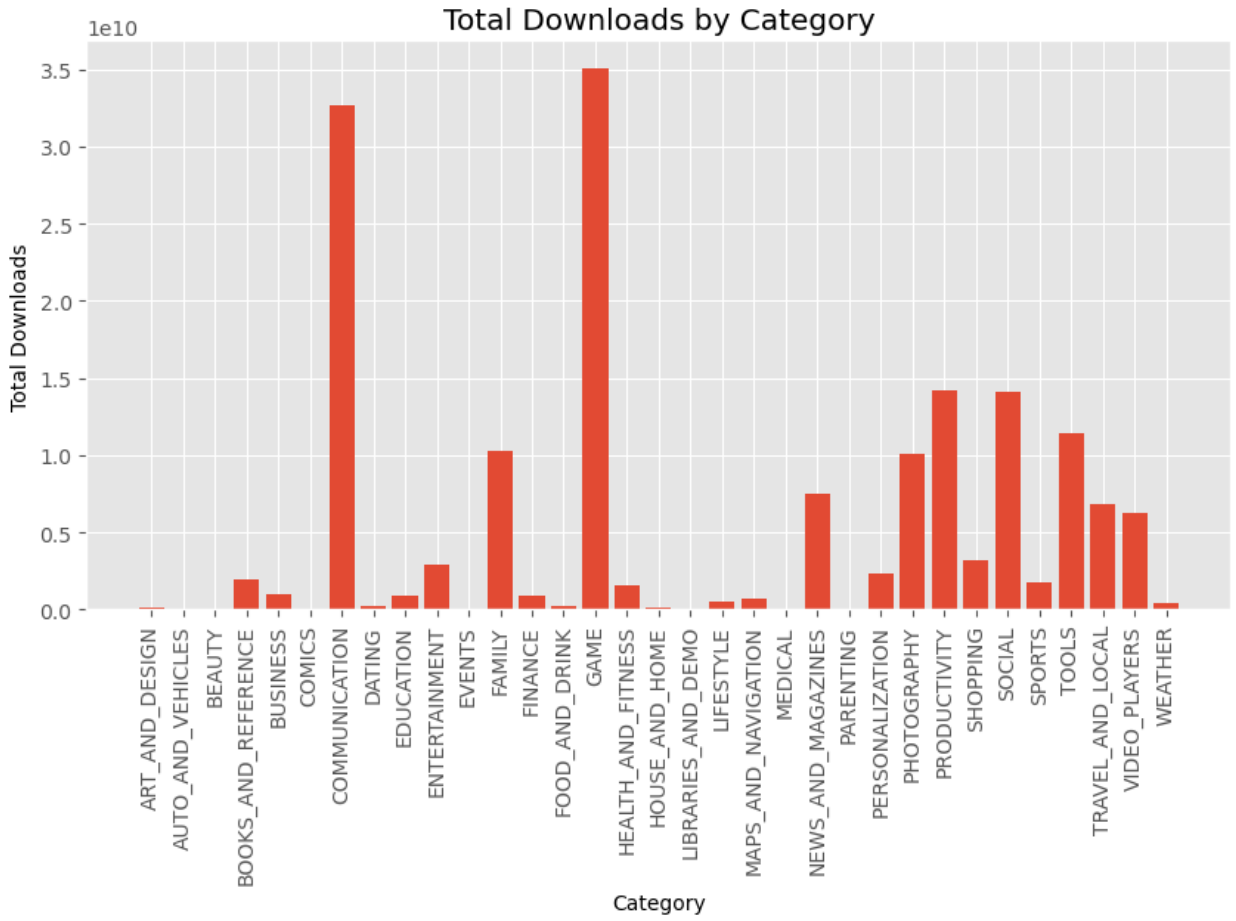

```
# Assuming 'df' is your DataFrame and 'Release Date' and 'Download
Count' are the relevant columns

# Convert the 'Release Date' column to a numerical format
#df['Release Date'] = pd.to_numeric(df['Release Date'],
errors='coerce')

# Create a scatter plot to analyze the relationship
plt.figure(figsize=(10, 8))
plt.scatter(df['Size'], df['Installs'])
plt.xlabel('Size')
plt.ylabel('Installs')
plt.title('Relationship between Versions and Download Count')
plt.show()
```



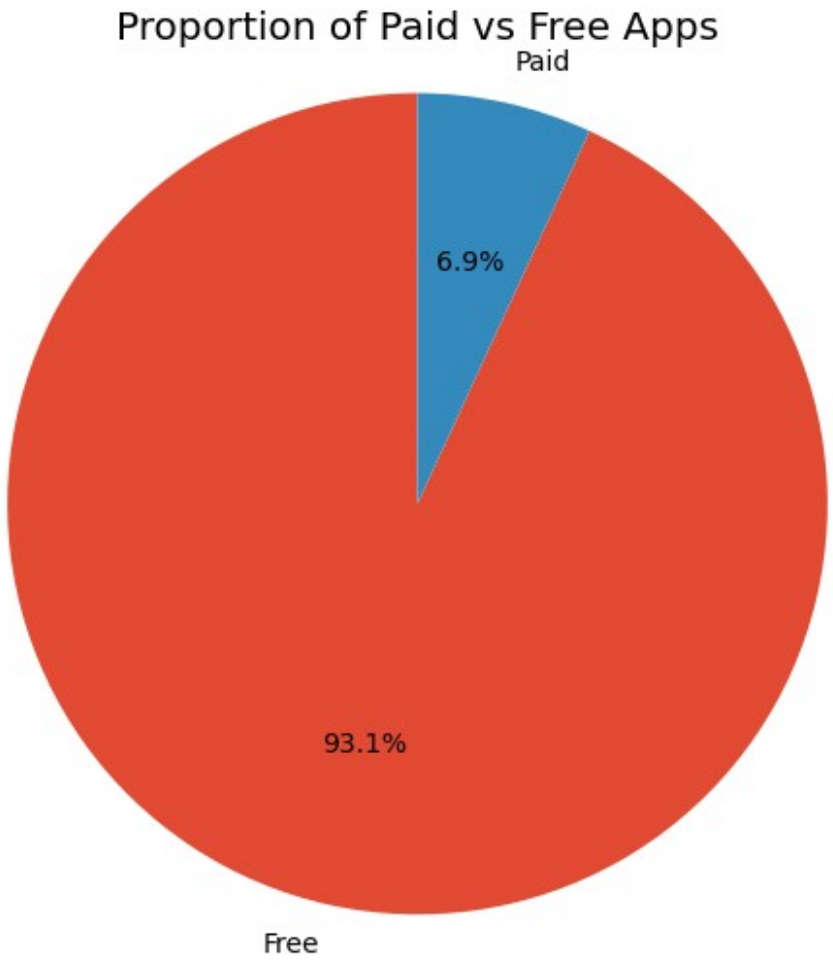

```
# Assuming 'df' is your DataFrame and 'Category', 'Rating', and
'Installs' are the relevant columns

# Calculate the average ratings for each category
average_ratings = df.groupby('Category')['Rating'].mean()

# Calculate the total download counts for each category
total_downloads = df.groupby('Category')['Installs'].sum()

# Create a bar plot for average ratings
plt.figure(figsize=(10, 5))
plt.bar(average_ratings.index, average_ratings.values)
plt.xlabel('Category')
plt.ylabel('Average Rating')
plt.title('Average Ratings by Category')
plt.xticks(rotation=90)
plt.show()
```

```
# Create a bar plot for total download counts
plt.figure(figsize=(10, 5))
plt.bar(total_downloads.index, total_downloads.values)
plt.xlabel('Category')
plt.ylabel('Total Downloads')
plt.title('Total Downloads by Category')
plt.xticks(rotation=90)
plt.show()
```

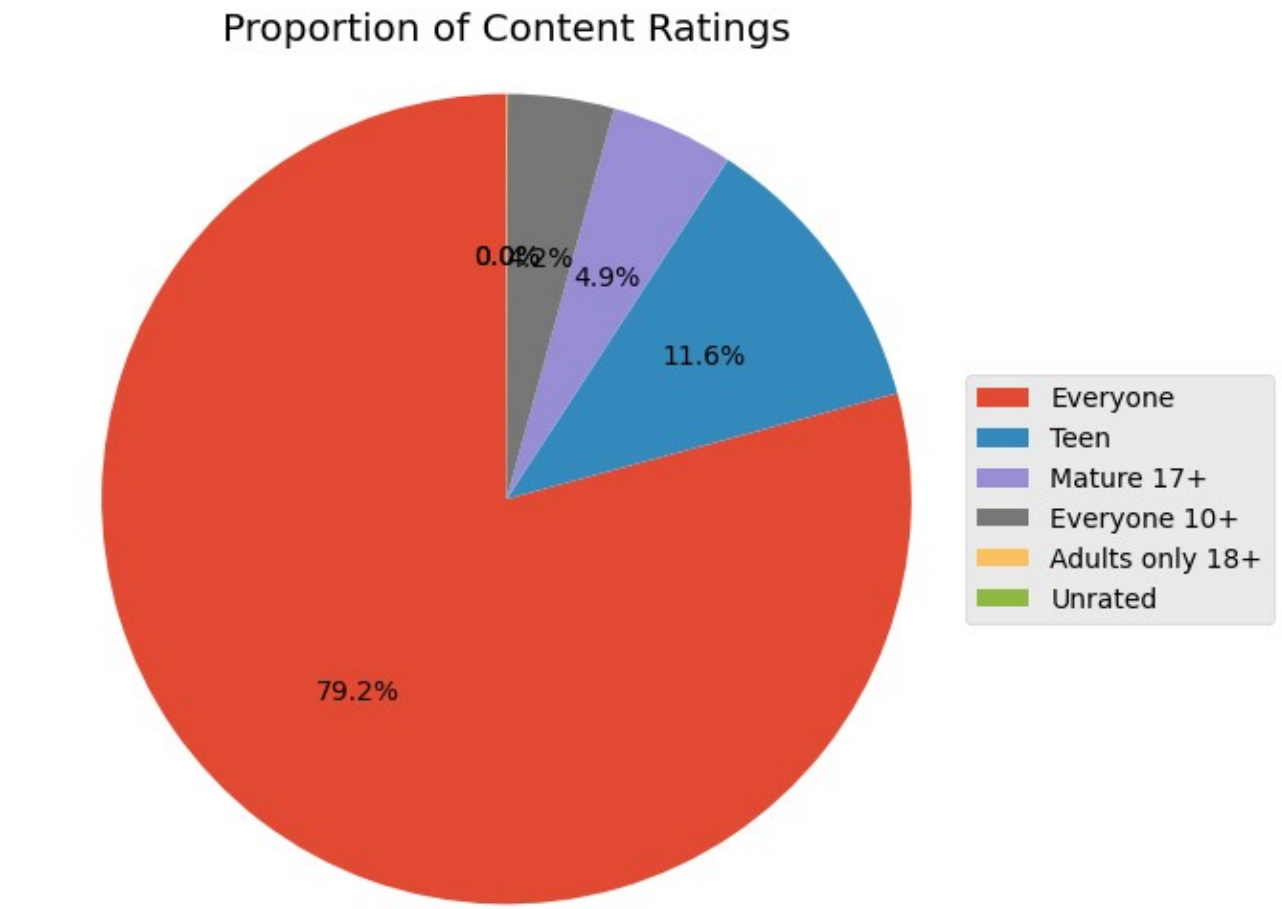





```
# Assuming 'df' is your DataFrame and 'Type' is the column indicating
app type (paid or free)

# Calculate the count of paid and free apps
app_counts = df['Type'].value_counts()

# Create a pie chart
plt.figure(figsize=(6, 6))
plt.pie(app_counts, labels=app_counts.index, autopct='%1.1f%%',
startangle=90)
plt.title('Proportion of Paid vs Free Apps')
plt.axis('equal')

# Display the pie chart
plt.show()
```




```
# Assuming 'df' is your DataFrame and 'Content Rating' is the column
indicating app content rating

# Calculate the count of each content rating
app_counts = df['Content Rating'].value_counts()

# Create a pie chart
plt.figure(figsize=(6, 6))
plt.pie(app_counts, labels=None, autopct='%1.1f%%', startangle=90)
plt.title('Proportion of Content Ratings')
plt.axis('equal')

# Create a separate key using legend
plt.legend(labels=app_counts.index, loc='center left',
bbox_to_anchor=(1, 0.5))

# Display the pie chart
plt.show()
```


Proportion of Content Ratings
4.9%
11.6%
79.2%
Everyone
Teen
Mature 17+
Everyone 10+
Adults only 18+
Unrated