Exploratory Data Analysis on Google Playstore Apps Data

Introduction

img

Google Play Store or formerly Android Market, is a digital distribution service developed and operated by Google. It is an official apps store that provides variety content such as apps, books, magazines, music, movies and television programs. It serves an as platform to allow users with 'Google certified' Android operating system devices to donwload applications developed and published on the platform either with a charge or free of cost. With the rapidly growth of Android devices and apps, it would be interesting to perform data analysis on the data to obtain valuable insights.

The dataset that is going to be used is '**Google Play Store Apps**' from **Kaggle**. It contains thousands of web scraped Play Store apps data for analysing the Android market. The tools that are going to be used for this EDA would be numpy, pandas, matplotlib and seaborn etc. which I have learnt from Jovian-course.

```
!pip install jovian --upgrade --quiet
```

```
import jovian
```

```
# Execute this to save new versions of the notebook
jovian.commit(project="google-playstore-apps")
```

```
[jovian] Detected Colab notebook...
[jovian] Uploading colab notebook to Jovian...
Committed successfully! https://jovian.ai/abhineets17/google-playstore-apps
```

```
'https://jovian.ai/abhineets17/google-playstore-apps'
```

Google Playstore Apps Exploratory Data Analysis

Exploratory Data Analysis

Exploratory data analysis (EDA) is used to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

Here we are using the **Google Playstore dataset**, which contains details about the Apps in playstore, there are more than 10,0000+ Apps in the playstore.

Objective:

This project focuses on the analysis of the Play Store data set in Kaggle. The dataset we are using is taken from the Kaggle, the link of the dataset is given below:

Data set link:- Kaggle link

The aim of this project is:

1. Using the data to analyze consumer trends and determine which type of apps are the most popular and profitable.
2. Classifying applications based on their categories.
3. Presenting the growth of applications.
4. Comparing different categories of applications.

Discussion of Google play store dataset will involve various steps such as:

1. Loading the data into data frame
2. Cleaning the data
3. Extracting statistics from the dataset
4. Exploratory analysis and visualizations
5. Questions that can be asked from the dataset
6. Summary & Conclusion
7. Reference
8. Future Work

Data Preparation and Cleaning

In this section, we will be loading the Google Store Apps data stored in csv using pandas which is a fast and powerful python library for data analysis and easy data manipulation in pandas DataFrame object. It is usually used for working with tabular data (e.g data in spreadsheet) in various formats such as CSV, Excel spreadsheets, HTML tables, JSON etc. We will then perform some data preparation and also cleaning on it.

```
!pip install opendatasets --upgrade --quiet
```

Downloading the dataset.

```
import opendatasets as od


download_url = "https://www.kaggle.com/datasets/gauthamp10/google-playstore-apps"
od.download(download_url)
```

```
Skipping, found downloaded files in "./google-playstore-apps" (use force=True to force
download)
```

```
data_filename ='./google-playstore-apps/Google-Playstore.csv'
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import scipy.stats as stats
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

Load the apps and reviews data into pandas dataframe

```
df = pd.read_csv(data_filename)
```

After loading the dataset, we can start the exploration but before that, we need to check and see that the dataset is ready for performing several exploration operations or not, so let's first have a look at the structure and the manner in which the data is organized.

```
df
```

	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs	Ma
0	Gakondo	com.ishakwe.gakondo	Adventure	0.0	0.0	10+	10.0	
1	Ampere Battery Info	com.webserveis.batteryinfo	Tools	4.4	64.0	5,000+	5000.0	
2	Vibook	com.doantiepvien.crm	Productivity	0.0	0.0	50+	50.0	
3	Smart City Trichy Public Service Vehicles 17UC...	cst.stJoseph.ug17ucs548	Communication	5.0	5.0	10+	10.0	
4	GROW.me	com.horodyski.grower	Tools	0.0	0.0	100+	100.0	
...	...	...	...	...	...	...	...	
2312939	□□□—□ □□□	com.rxsj.ssjj	Role Playing	4.3	16775.0	100,000+	100000.0	3
2312940	ORU Online	com.threedream.oruonline	Education	0.0	0.0	100+	100.0	
2312941	Data Structure	datastructure.appoworld.datastucture	Education	0.0	0.0	100+	100.0	
2312942	Devi Suktam	ishan.devi.suktam	Music & Audio	3.5	8.0	1,000+	1000.0	
2312943	Biliyor Musun - Sonsuz Yarış	com.yyazilim.biliyormusun	Trivia	5.0	12.0	100+	100.0	

2312944 rows × 24 columns

Getting the number of rows and columns of the dataframe.

```
df.shape
```

```
(2312944, 24)
```

Look at the first 10 records in the apps dataframe

```
df.head(10)
```

	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs	Max Ir
0	Gakondo	com.ishakwe.gakondo	Adventure	0.0	0.0	10+	10.0	
1	Ampere Battery Info	com.webserveis.batteryinfo	Tools	4.4	64.0	5,000+	5000.0	
2	Vibook	com.doantiepvien.crm	Productivity	0.0	0.0	50+	50.0	
3	Smart City Trichy Public Service Vehicles 17UC...	cst.stJoseph.ug17ucs548	Communication	5.0	5.0	10+	10.0	
4	GROW.me	com.horodyski.grower	Tools	0.0	0.0	100+	100.0	
5	IMOCCI	com.imocci	Social	0.0	0.0	50+	50.0	
6	unlimited 4G data prank free app	getfreedata.superfatiza.unlimitedjiodataprank	Libraries & Demo	4.5	12.0	1,000+	1000.0	
7	The Everyday Calendar	com.mozaix.simoneboard	Lifestyle	2.0	39.0	500+	500.0	
8	WhatsOpen	com.whatsopen.app	Communication	0.0	0.0	10+	10.0	
9	Neon 3d Iron Tech Keyboard Theme	com.ikeyboard.theme.neon_3d.iron.tech	Personalization	4.7	820.0	50,000+	50000.0	6

10 rows × 24 columns

Look at the random 10 records in the apps dataframe

```
df.sample(10)
```

	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs
693832	Photo Recovery 2019	com.nabsal.recoverphotos	Tools	3.6	152.0	50,000+	50000.0
2143187	NVVS Law Injury Help App	nvvs.pi.law	Lifestyle	0.0	0.0	10+	10.0
170122	Verben - Trainer	com.composapps.verbentrainer	Education	4.4	553.0	50,000+	50000.0
2155917	WineAdvisor	com.wineadvisor	Food & Drink	3.3	1188.0	100,000+	100000.0
1512718	Mechanical Production Process	com.infoland.mechanical_production_process	Education	3.7	20.0	5,000+	5000.0
1188655	WBRN	com.wbrn.player	Music & Audio	0.0	0.0	50+	50.0
78356	Up Dance Sport Team	com.appeasybuild.androidupdance	Health & Fitness	0.0	0.0	100+	100.0
731291	कक्षा 11 गणित NCERT	com.rdseducation.maths11.hindi.ncert	Education	4.2	37.0	10,000+	10000.0
387959	Salón Backstage	com.bewe.app1793	Lifestyle	0.0	0.0	1+	1.0
497996	7th Class Social Science NCERT Solution in Hindi	com.social.science.hindi	Education	0.0	0.0	500+	500.0

10 rows × 24 columns

Description of App Dataset columns

This files contains Application data of more than 600K applications with the following 24 attributes:-

```
1.App Name
2.App Id
3.Category
4.Rating
5.Rating Count
6.Installs
7.Minimum Installs
8.Maximum Installs
9.Free
10.Price
11.Currency
12.Size
13.Minimum Android
14.Developer Id
```

```
    15.Developer Website
    16.Developer Email
    17.Released
    18.Privacy Policy
    19.Last Updated
    20.Content Rating
    21.Ad Supported
    22.In app purchases
    23.Editor Choice
    24.Scraped Time
```

Total column in dataset.

```
df.columns
```

```
Index(['App Name', 'App Id', 'Category', 'Rating', 'Rating Count', 'Installs',
       'Minimum Installs', 'Maximum Installs', 'Free', 'Price', 'Currency',
       'Size', 'Minimum Android', 'Developer Id', 'Developer Website',
       'Developer Email', 'Released', 'Last Updated', 'Content Rating',
       'Privacy Policy', 'Ad Supported', 'In App Purchases', 'Editors Choice',
       'Scraped Time'],
      dtype='object')
```

Total length of the column.

```
len(df.columns)
```

```
24
```

Look that the info of the dataframe

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2312944 entries, 0 to 2312943
Data columns (total 24 columns):
 #   Column             Dtype
---  ------             -----
 0   App Name           object
 1   App Id             object
 2   Category           object
 3   Rating             float64
 4   Rating Count       float64
 5   Installs           object
 6   Minimum Installs   float64
 7   Maximum Installs   int64
 8   Free               bool
 9   Price              float64
 10  Currency           object
 11  Size               object
 12  Minimum Android    object
 13  Developer Id       object
 14  Developer Website  object
 15  Developer Email    object
 16  Released           object
 17  Last Updated       object
 18  Content Rating     object
 19  Privacy Policy     object
 20  Ad Supported       bool
 21  In App Purchases   bool
 22  Editors Choice     bool
 23  Scraped Time       object
dtypes: bool(4), float64(4), int64(1), object(15)
memory usage: 361.8+ MB
```

By diagnosing the data frame, we know that:

1.There are 23 columns of properties with 10k+ rows of data.

2.Column 'Reviews', 'Size', 'Installs' and 'Price' are in the type of 'object'.

3.Values of column 'Size' are strings representing size in 'M' as Megabytes, 'k' as kilobytes and also 'Varies with devices'.

4.Values of column 'Installs' are strings representing install amount with symbols such as ',' and '+'.

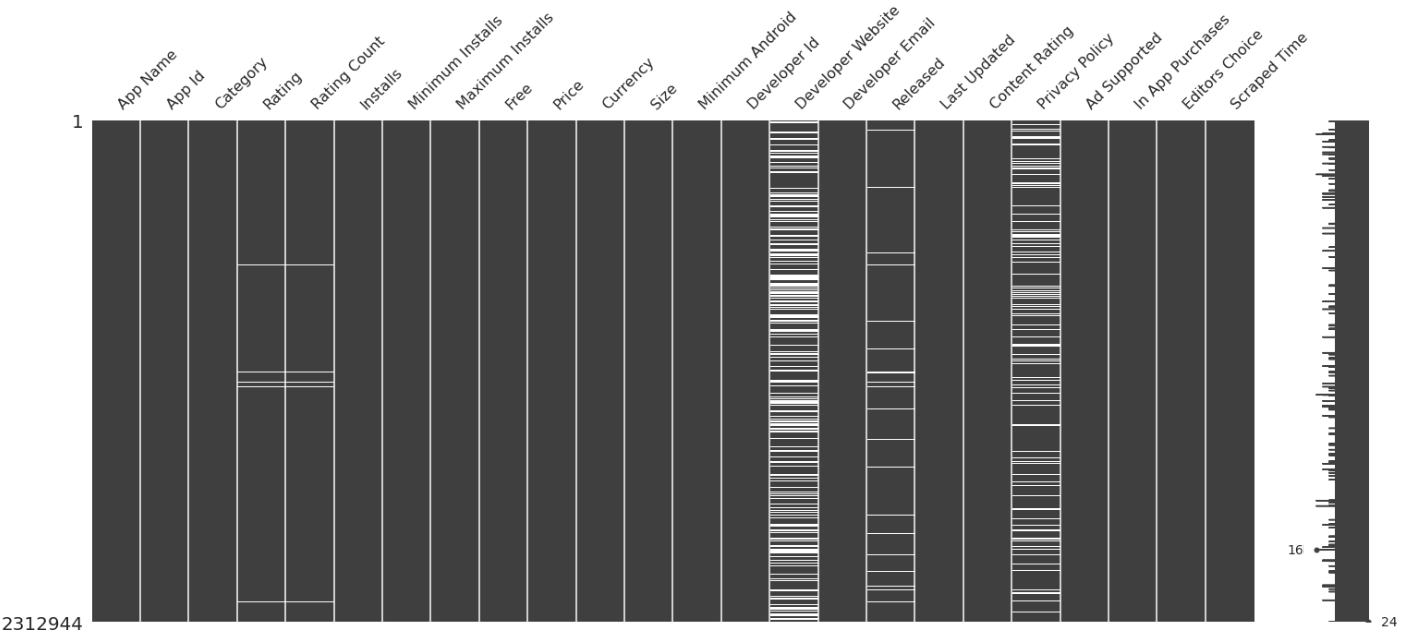
5.Values of column 'Price' are strings representing price with symbol '$'.

```
# USED FOR CHECKING MISSING VALUES VISUALLY!
import missingno as msno
```

Missingno library offers a very nice way to visualize the distribution of NaN values. Missingno is a Python library and compatible with Pandas.

```
# USING MISSING NO LIBRARY!  --> WHITE LINES SHOWS MISSING ROWS IN THE DATA!

msno.matrix(df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c54d2fd0>
```



White lines shows missing rows in the data.

```
#  Used to detect missing values.
df.isna()
```

	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	...	Developer Website	Develop Em
0	False	False	False	False	False	False	False	False	False	False	...	False	Fal
1	False	False	False	False	False	False	False	False	False	False	...	False	Fal
2	False	False	False	False	False	False	False	False	False	False	...	True	Fal
3	False	False	False	False	False	False	False	False	False	False	...	False	Fal
4	False	False	False	False	False	False	False	False	False	False	...	False	Fal
...	...	...	...	...	...	...	...	...	...	...	...	...	
2312939	False	False	False	False	False	False	False	False	False	False	...	False	Fal
2312940	False	False	False	False	False	False	False	False	False	False	...	False	Fal
2312941	False	False	False	False	False	False	False	False	False	False	...	True	Fal
2312942	False	False	False	False	False	False	False	False	False	False	...	False	Fal
2312943	False	False	False	False	False	False	False	False	False	False	...	True	Fal

2312944 rows × 24 columns

```
# Used to know the sum null values in a specific row in pandas dataframe.
df.isna().sum()
```

```
App Name                2
App Id                  0
Category                0
Rating              22883
Rating Count        22883
Installs              107
Minimum Installs      107
```

```
Maximum Installs          0
Free                      0
Price                     0
Currency                135
Size                    196
Minimum Android        6530
Developer Id             33
Developer Website    760835
Developer Email          31
Released              71053
Last Updated              0
Content Rating            0
Privacy Policy       420953
Ad Supported              0
In App Purchases          0
Editors Choice            0
Scraped Time              0
dtype: int64
```

Percentage of missing value per column

```
#df.isna().sum().sort_values()

missing_percentages = df.isna().sum().sort_values(ascending = False) / len(df)
missing_percentages
```

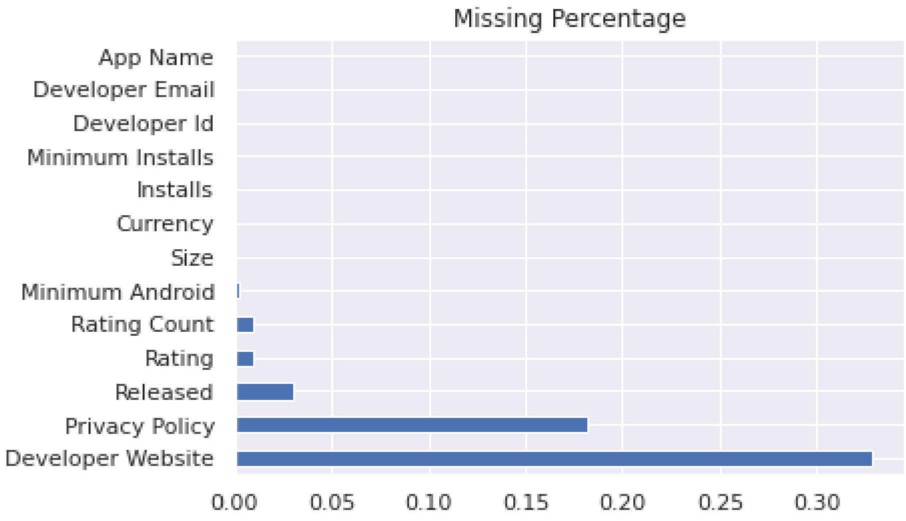
```
Developer Website    3.289466e-01
Privacy Policy       1.819988e-01
Released             3.071972e-02
Rating               9.893452e-03
Rating Count         9.893452e-03
Minimum Android      2.823242e-03
Size                 8.474049e-05
Currency             5.836717e-05
Installs             4.626139e-05
Minimum Installs     4.626139e-05
Developer Id         1.426753e-05
Developer Email      1.340283e-05
App Name             8.646988e-07
App Id               0.000000e+00
Price                0.000000e+00
Free                 0.000000e+00
Maximum Installs     0.000000e+00
Last Updated         0.000000e+00
Content Rating       0.000000e+00
Category             0.000000e+00
Ad Supported         0.000000e+00
In App Purchases     0.000000e+00
Editors Choice       0.000000e+00
```

```
Scraped Time          0.000000e+00
dtype: float64
```

```
missing_percentages[missing_percentages !=0]
```

```
Developer Website    3.289466e-01
Privacy Policy       1.819988e-01
Released             3.071972e-02
Rating               9.893452e-03
Rating Count         9.893452e-03
Minimum Android      2.823242e-03
Size                 8.474049e-05
Currency             5.836717e-05
Installs             4.626139e-05
Minimum Installs     4.626139e-05
Developer Id         1.426753e-05
Developer Email      1.340283e-05
App Name             8.646988e-07
dtype: float64
```

```
missing_percentages[missing_percentages !=0].plot(kind = 'barh',title='Missing Percenta
```



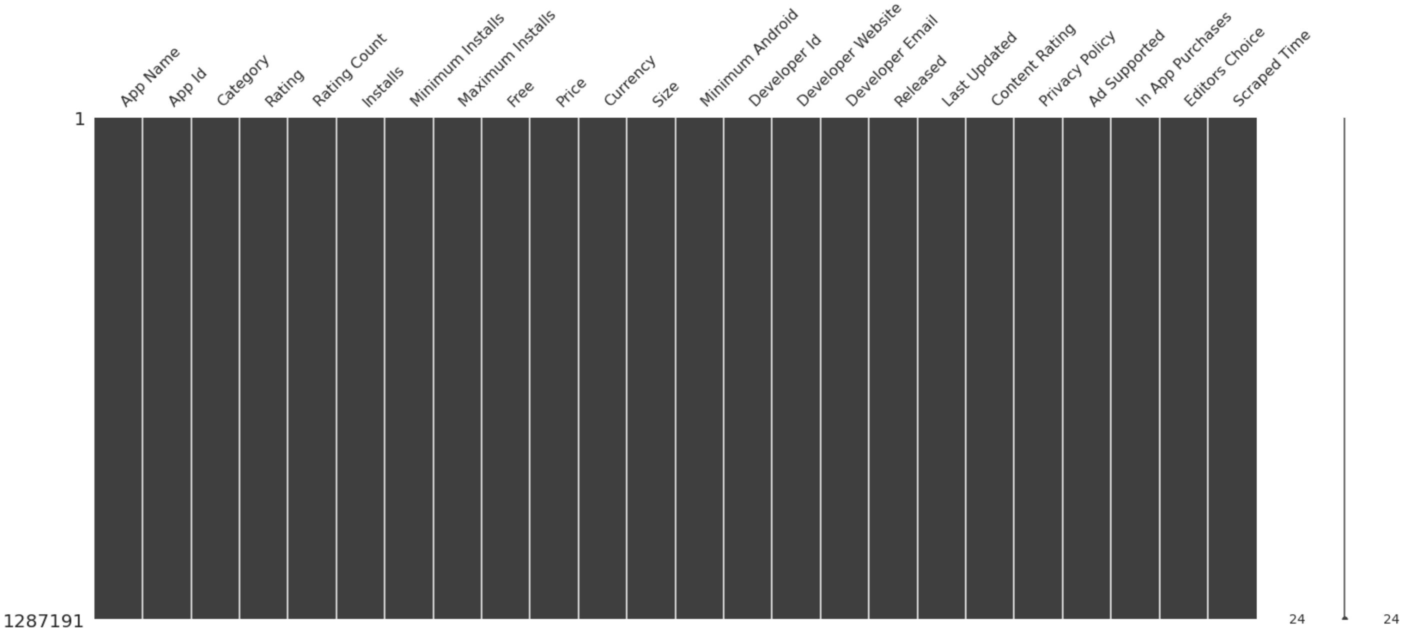
```
# Number of rows having null values in the dataset (In percentage)

print("Number of rows having null values in the dataset:")
missing_info = (len(df[df.isnull().any(axis=1)]) / len(df) )*100
print(len(df[df.isnull().any(axis=1)]),' which is ' ,round(missing_info,2) , '%')
```

```
Number of rows having null values in the dataset:
1025753  which is  44.35 %
```

```
#DROPPING NANS, NULL ENTRIES IN THE DATA!
df= df.dropna()
```

```
#NO WHITE LINES --> DATA HAS NO MISSING DATA!
msno.matrix(df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4b3cbced0>
```



There is no white lines in row it means there is no missing data.

Converting Installs column to float

```
df.Installs = df.Installs.str.replace('+','')
df.Installs = df.Installs.str.replace(',','')
df.Installs.fillna(0,inplace=True)
df.Installs = df.Installs.astype('float')
```

```
# # create a copy of the original
# data = df.copy()

# #check the value counts of each column and look for any weirdness
# for col in data:
#     print(col)
#     print(data[col].value_counts(normalize=True))
#     print('-------------------------------')
```

Exploratory Analysis & Visulaization

Columns we will analyze:

1. Rating
2. Category
3. Free/Paid apps
4. Content Rating
5. Ad Supported and In App Purchases

1.Rating

```
df.Rating
```

```
0          0.0
1          4.4
4          0.0
5          0.0
9          4.7
          ... 
2312933    4.0
2312934    0.0
2312938    3.4
2312940    0.0
2312942    3.5
Name: Rating, Length: 1287191, dtype: float64
```

```
rating = df.Rating.unique()
len(rating)
```

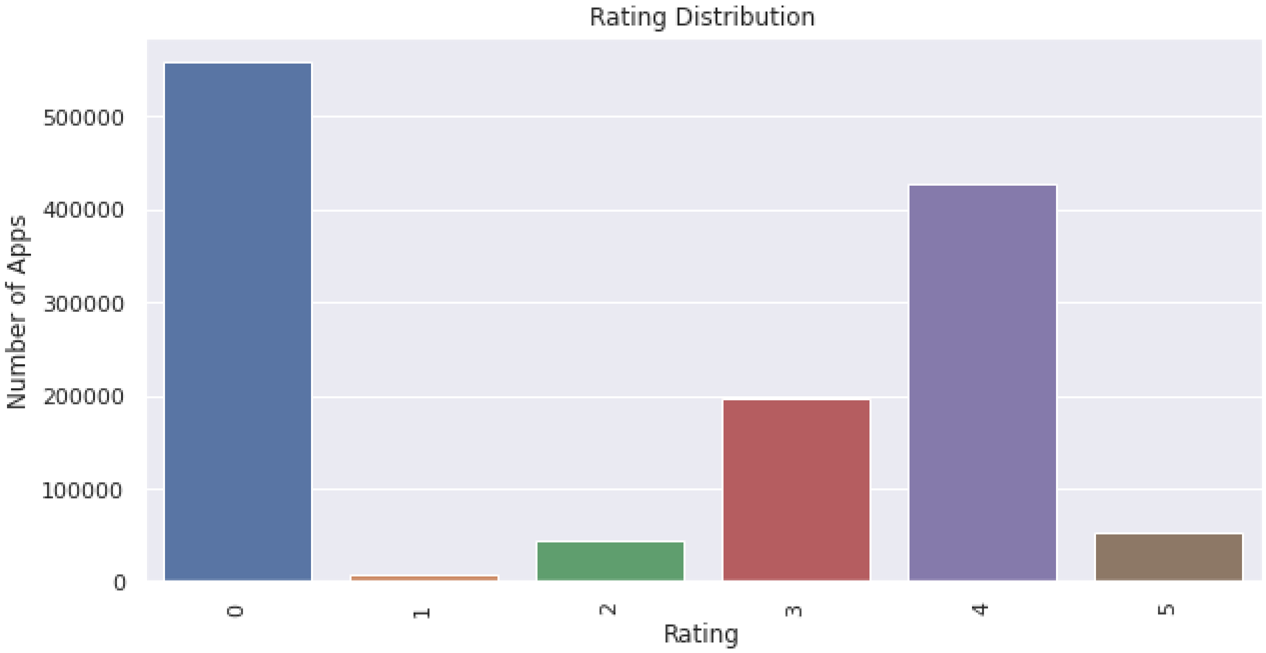
```
42
```

```
# Rating data type
df.Rating.dtype
```

```
dtype('float64')
```

```
#replace all NaN values with zeros
df['Rating'] = df['Rating'].fillna(0)

#convert 'Rating' column from float to integer
df['Rating'] = df['Rating'].astype(int)
```

```
# show the distribution of rating
plt.figure(figsize=(10, 5))
sns.countplot(x='Rating', data=df)
plt.title('Rating Distribution')
plt.xticks(rotation=90)
plt.ylabel('Number of Apps')
plt.show()
```

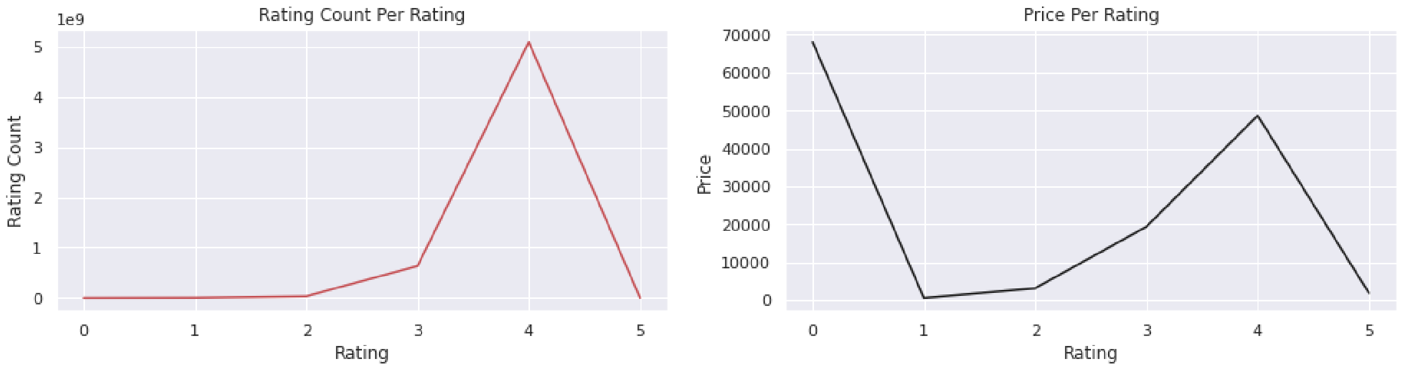


Rating distribution shows that most of the apps are get zero(0) rating,its not good but four(4) rating get 2nd highest rating of number of apps.

```
df.columns
```

```
Index(['App Name', 'App Id', 'Category', 'Rating', 'Rating Count', 'Installs',
       'Minimum Installs', 'Maximum Installs', 'Free', 'Price', 'Currency',
       'Size', 'Minimum Android', 'Developer Id', 'Developer Website',
       'Developer Email', 'Released', 'Last Updated', 'Content Rating',
       'Privacy Policy', 'Ad Supported', 'In App Purchases', 'Editors Choice',
       'Scraped Time'],
      dtype='object')
```

```
# plot the graphs of reviews, installs and price per rating
rating_df = df.groupby('Rating').sum().reset_index()

fig, axes = plt.subplots(1, 2, figsize=(14, 4))

axes[0].plot(rating_df['Rating'], rating_df['Rating Count'], 'r')
axes[0].set_xlabel('Rating')
axes[0].set_ylabel('Rating Count')
axes[0].set_title('Rating Count Per Rating')

axes[1].plot(rating_df['Rating'], rating_df['Price'], 'k')
axes[1].set_xlabel('Rating')
axes[1].set_ylabel('Price')
axes[1].set_title('Price Per Rating')

plt.tight_layout(pad=2)
plt.show()
```



1st graph shows four(4) rating have highest number of rating count. 2nd graph shows most of higher price apps gets zero(0) rating.

2.Category

```
df.Category
```

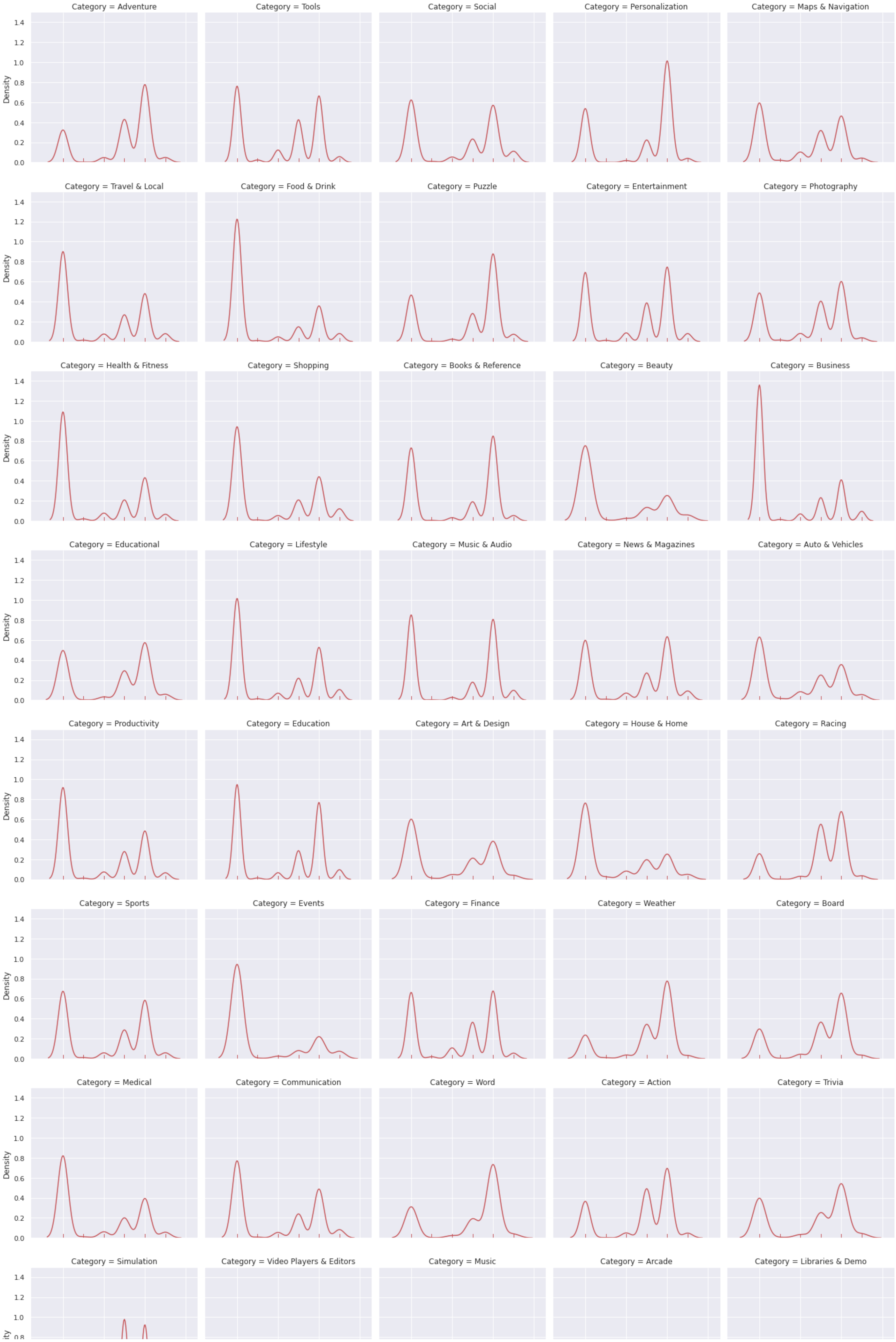
```
0                  Adventure
1                      Tools
4                      Tools
5                     Social
9            Personalization
                 ...
2312933        Music & Audio
2312934            Education
2312938            Education
2312940            Education
2312942        Music & Audio
Name: Category, Length: 1287191, dtype: object
```

```
# Count the number of apps in each 'Category'.
num_apps_in_category = df.Category.value_counts()
num_apps_in_category
```

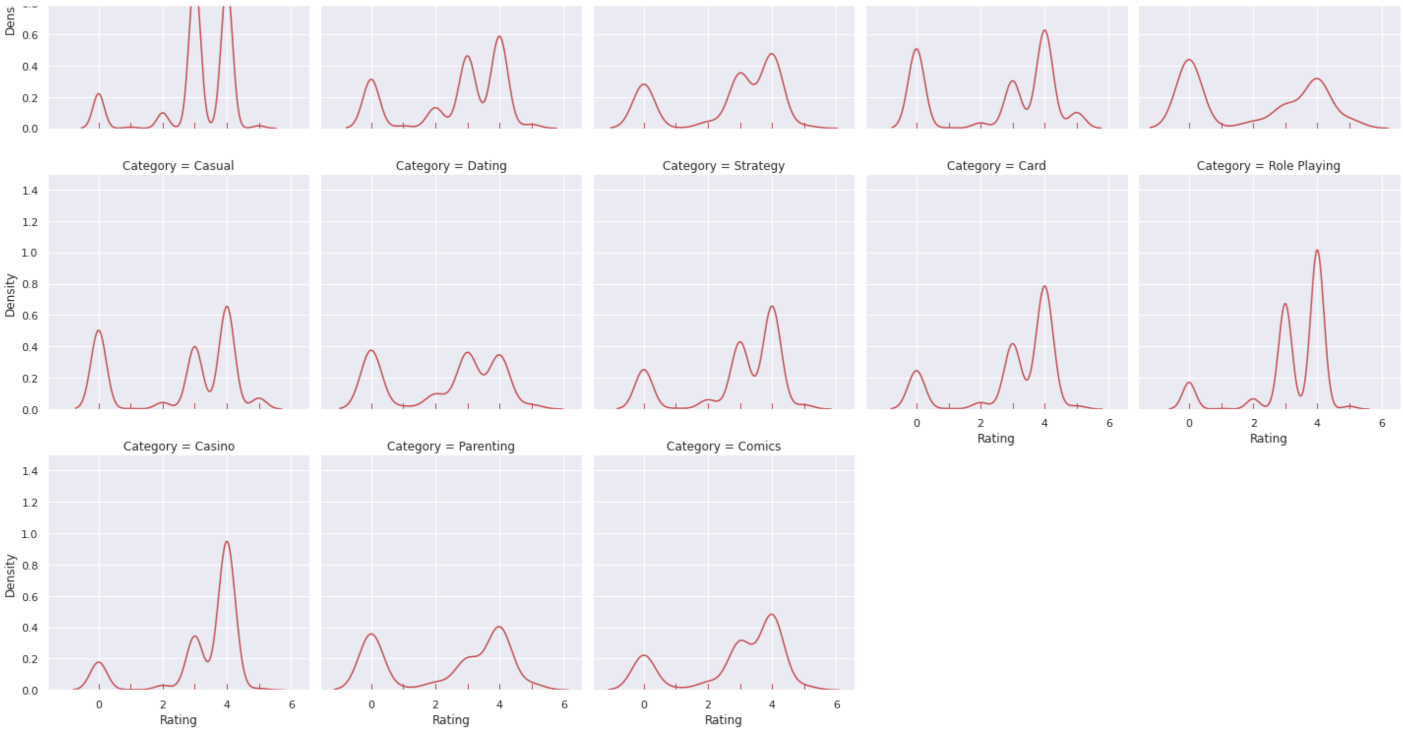
```
Education                  127219
Business                   100311
Music & Audio               87008
Lifestyle                   75017
Tools                       67215
Entertainment               63581
Books & Reference           56515
Health & Fitness            51770
Shopping                    48981
Productivity                46960
Travel & Local              46613
Food & Drink                45773
Finance                     44111
Personalization             37819
Communication               30703
News & Magazines            29320
```

```
Sports                      29180
Social                      27164
Puzzle                      25069
Casual                      21958
Medical                     20260
Arcade                      19925
Photography                 16827
Maps & Navigation           15821
Educational                 13048
Simulation                  12569
Action                      12385
Auto & Vehicles             10744
Adventure                   10295
House & Home                 8707
Events                       8497
Art & Design                 8007
Video Players & Editors      6883
Beauty                       6179
Trivia                       5885
Role Playing                 5711
Racing                       5383
Board                        5175
Word                         4989
Card                         4858
Strategy                     4117
Weather                      4103
Dating                       3429
Casino                       2810
Parenting                    2381
Libraries & Demo             2371
Music                        2200
Comics                       1345
Name: Category, dtype: int64
```

```
correlation = sns.FacetGrid(df, col='Category', palette="Set1",  col_wrap=5, height=4)
correlation = (correlation.map(sns.distplot, "Rating", hist=False, rug=True, color="r"))
```

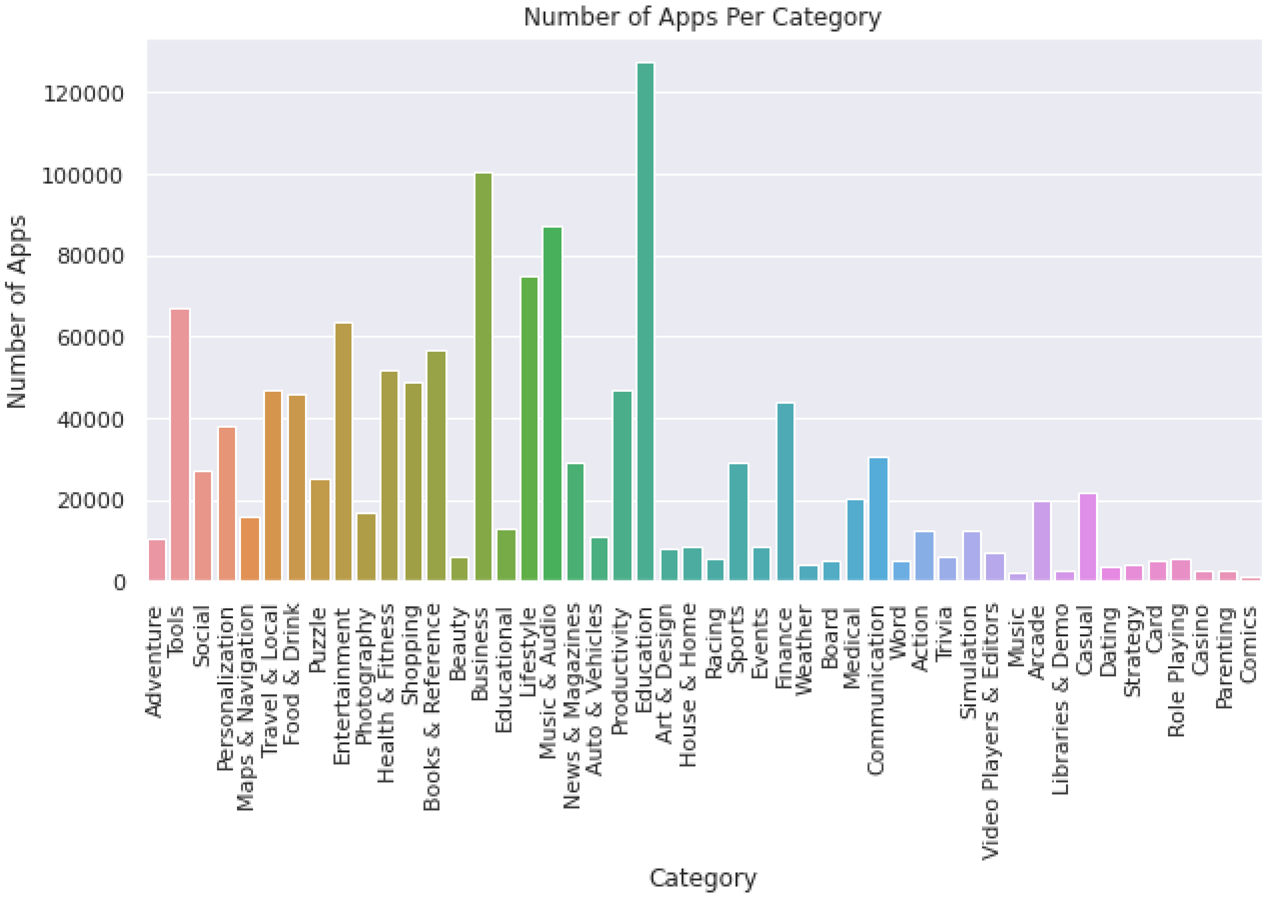


Category = Adventure
Category = Tools
Category = Social
Category = Personalization
Category = Maps & Navigation
Category = Travel & Local
Category = Food & Drink
Category = Puzzle
Category = Entertainment
Category = Photography
Category = Health & Fitness
Category = Shopping
Category = Books & Reference
Category = Beauty
Category = Business
Category = Educational
Category = Lifestyle
Category = Music & Audio
Category = News & Magazines
Category = Auto & Vehicles
Category = Productivity
Category = Education
Category = Art & Design
Category = House & Home
Category = Racing
Category = Sports
Category = Events
Category = Finance
Category = Weather
Category = Board
Category = Medical
Category = Communication
Category = Word
Category = Action
Category = Trivia
Category = Simulation
Category = Video Players & Editors
Category = Music
Category = Arcade
Category = Libraries & Demo
Density
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4



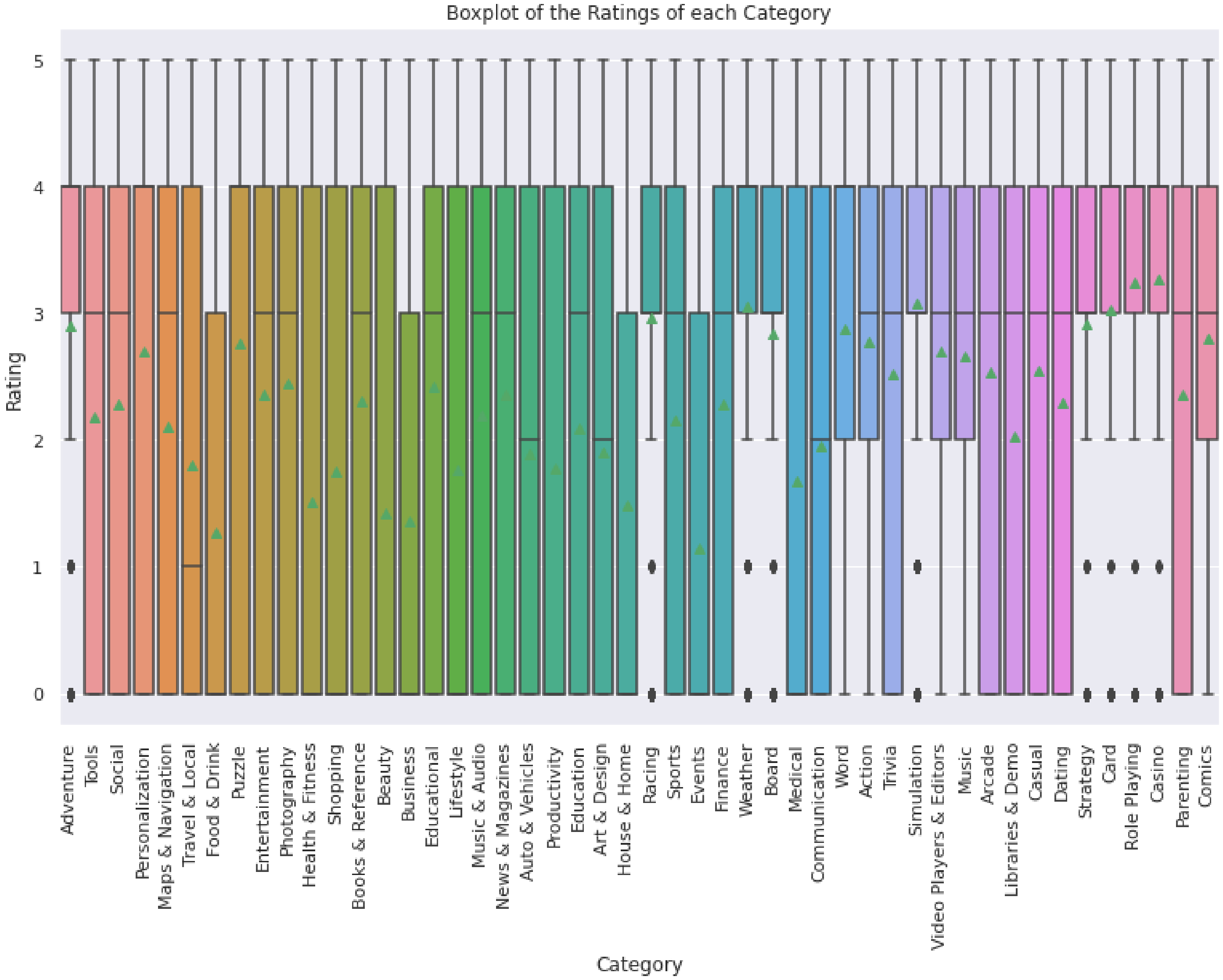
By the horizontal is the rating value, and vertically is quantity of the rating. This graph is the correlation between category and rating.

```
# get the number of apps for each category
sns.set_style('darkgrid')
plt.figure(figsize=(10, 5))
sns.countplot(x='Category', data=df)
plt.title('Number of Apps Per Category')
plt.xticks(rotation=90)
plt.ylabel('Number of Apps')
plt.show()
```



This graph shows, Education category have highest number of apps. And Comics category have lowest number of apps.

```
# Boxplot of the ratings for each category
plt.figure(figsize=(13,8));
plt.title('Boxplot of the Ratings of each Category');
sns.boxplot(x=df['Category'],y=df['Rating'],showmeans=True)
plt.xticks(rotation=90);
```



Casino category have highest rating.And Events category have lowest rating.

```
group_category = df.groupby('Category')
sorted_rating_by_category = group_category['Rating'].mean().sort_values(ascending=False
sorted_rating_by_category
```

```
Category
Casino                     3.264413
Role Playing               3.240413
Simulation                 3.079720
Weather                    3.051426
Card                       3.028613
Racing                     2.963589
Strategy                   2.920087
Adventure                  2.908014
Word                       2.882742
Board                      2.843285
Comics                     2.805204
Action                     2.780379
Puzzle                     2.765647
Personalization            2.705228
Video Players & Editors    2.698242
```

```
Music                     2.668182
Casual                    2.542946
Arcade                    2.532196
Trivia                    2.518946
Photography               2.450051
Educational               2.425889
News & Magazines          2.363029
Entertainment             2.360343
Parenting                 2.356993
Books & Reference         2.313598
Dating                    2.295713
Finance                   2.286006
Social                    2.281291
Music & Audio             2.194028
Tools                     2.184259
Sports                    2.162543
Maps & Navigation         2.106378
Education                 2.094624
Libraries & Demo          2.035006
Communication             1.948474
Art & Design              1.902335
Auto & Vehicles           1.890637
Travel & Local            1.807886
Productivity              1.779238
Lifestyle                 1.767399
Shopping                  1.753598
Medical                   1.680503
Health & Fitness          1.510663
House & Home              1.488458
Beauty                    1.427901
Business                  1.363250
Food & Drink              1.274725
Events                    1.144992
Name: Rating, dtype: float64
```

```python
# Taking the top 5 categories and the bottom 5 categories
top_5_category = sorted_rating_by_category.index[0:5]
bottom_5_category = sorted_rating_by_category.index[-5:]
print('The top 5 rated categories are {},{},{},{},and {}'.format(*list(top_5_category))
print('The bottom 5 rated categories are {},{},{},{}, and {}'.format(*list(bottom_5_cat
```

```
The top 5 rated categories are Casino,Role Playing,Simulation,Weather,and Card
The bottom 5 rated categories are House & Home,Beauty,Business,Food & Drink, and Events
```

```python
# Making a dataset consisting of only apps from the top 5 and bottom 5 categories
top_5_category_index = [i for i,x in enumerate(df['Category'].isin(top_5_category)) if
bottom_5_category_index = [i for i, x in enumerate(df['Category'].isin(bottom_5_categor

top_5_category_data = df.iloc[top_5_category_index].reset_index().drop('index',axis=1)
```

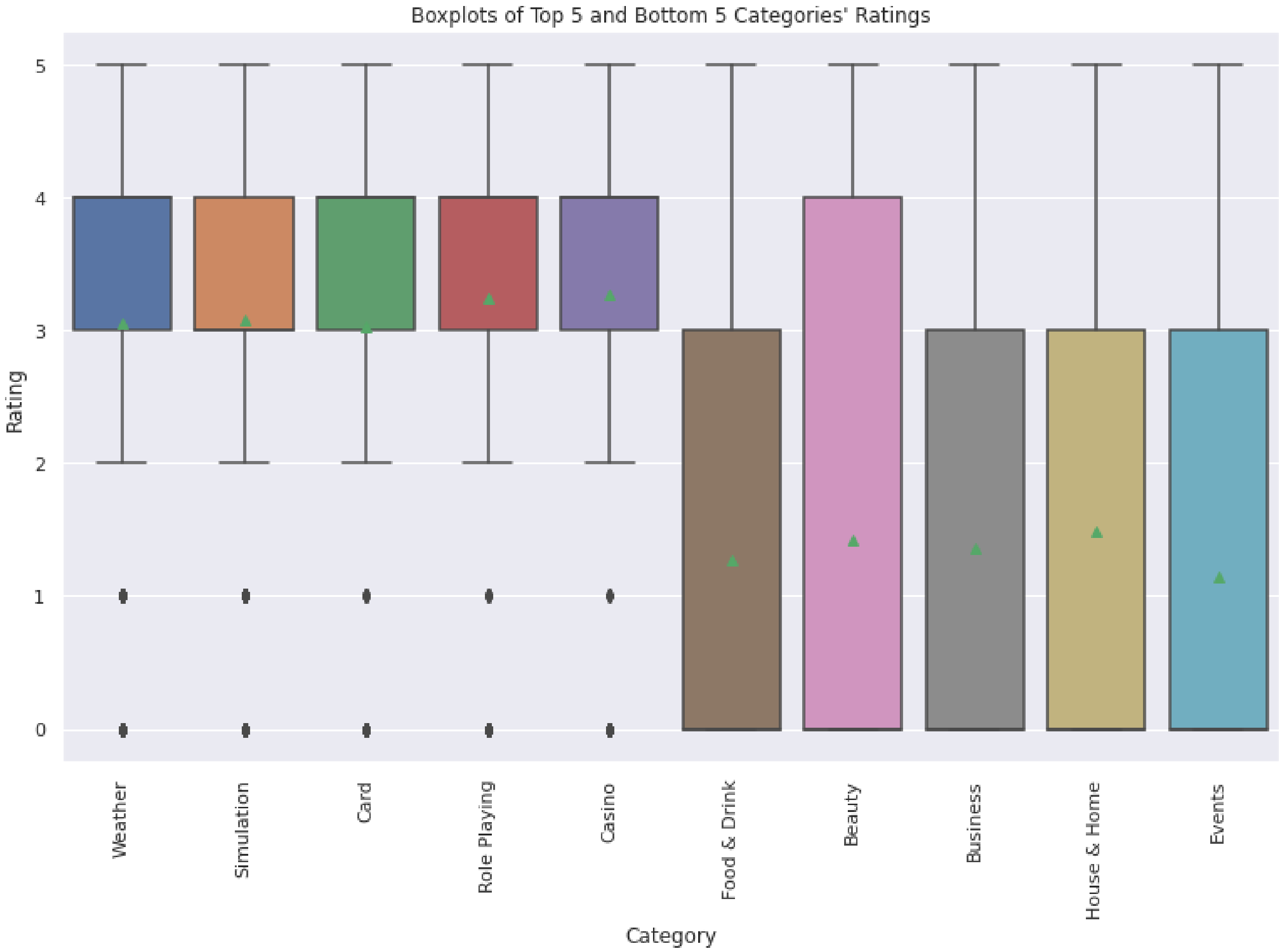
```
bottom_5_category_data = df.iloc[bottom_5_category_index].reset_index().drop('index',ax

top_bottom_category_data = pd.concat([top_5_category_data,bottom_5_category_data],axis=
top_bottom_category_data
```

	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs
0	Mega Ramps Car Stunts: Ultimate Races Car Games	com.vg.speed.car.racing.furious.stunts.games	Weather	4	149.0	50000.0	50000.0
1	Extreme Flight Sim	com.mtsgames.extremeflightsim	Simulation	3	36.0	1000.0	1000.0
2	Call to Sniper Duty: 3D Assassin FPS Battle 2020	com.aps.sniper.fury	Weather	4	1232.0	500000.0	500000.0
3	Virtual Baby Simulator - Mother Simulator 2020	com.vi.kidsimulator.virtualdad	Simulation	4	118.0	10000.0	10000.0
4	Euro Truck Simulator 2020 - Cargo Truck Driver	com.offroad.cargo.truck.simulator.euro	Simulation	3	9443.0	1000000.0	1000000.0
...	...	...	...	...	...	...	...
169462	Client Mobile Access	com.tempositions.cwa	Business	0	0.0	100.0	100.0
169463	Jax Auto Accident 411	com.wearesas.jaxautoaccident	Business	0	0.0	1.0	1.0
169464	Go Rentals Agent	com.gorentals.driver	Business	0	0.0	10.0	10.0
169465	DkVentas	ar.com.dekagb.dekaventas	Business	0	0.0	50.0	50.0
169466	Asset Data Collection	com.linq.assetdatacollection	Business	0	0.0	500.0	500.0

199518 rows × 24 columns

```
# Boxplot of these categories and ratings
plt.figure(figsize=(13,8))
plt.title('Boxplots of Top 5 and Bottom 5 Categories\' Ratings')
```

```
sns.boxplot(x='Category',y='Rating',data=top_bottom_category_data,showmeans=True)
plt.xticks(rotation=90);
```



Even though it looks as if there are some differences between the ratings of these top 5 and bottom 5 categories' ratings', they don't seem to differ by much.

3.Free/Paid apps

```
df['Price'].dtype
```

```
dtype('float64')
```

```
df['Free'].dtype
```

```
dtype('bool')
```

```
df['Price'].value_counts(normalize=True)
```

```
0.000000     9.816461e-01
0.990000     3.779548e-03
1.990000     2.430875e-03
2.990000     1.916576e-03
4.990000     1.359550e-03
                 ...
5.630000     7.768855e-07
```

```
5.040000    7.768855e-07
5.010000    7.768855e-07
4.840000    7.768855e-07
3.041816    7.768855e-07
Name: Price, Length: 631, dtype: float64
```

```
# Create a new feature (Free/Paid) to the dataset

if df['Price'].dtype == 'object' :
    df['Price'] = df['Price'].apply(lambda x : x.strip('$')).astype(float)
free_paid = ['Free' if i == 0 else 'Paid' for i in df['Price']]
free_paid_series = pd.Series(free_paid,name = 'Free/Paid')
df['Free/Paid'] = free_paid_series
df
```

	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs
0	Gakondo	com.ishakwe.gakondo	Adventure	0	0.0	10.0	10.0
1	Ampere Battery Info	com.webserveis.batteryinfo	Tools	4	64.0	5000.0	5000.0
4	GROW.me	com.horodyski.grower	Tools	0	0.0	100.0	100.0
5	IMOCCI	com.imocci	Social	0	0.0	50.0	50.0
9	Neon 3d Iron Tech Keyboard Theme	com.ikeyboard.theme.neon_3d.iron.tech	Personalization	4	820.0	50000.0	50000.0
...	...	...	...	...	...	...	...
2312933	Caustic Editor for VolcaSample	com.singlecellsoftware.kvsampler	Music & Audio	4	344.0	500000.0	500000.0
2312934	Vietnamese - English Translator	com.eliminatesapps.vietnamesetranslator	Education	0	0.0	5.0	5.0
2312938	Lero TOEFL Recorder + Timer	com.toefltimer	Education	3	17.0	1000.0	1000.0
2312940	ORU Online	com.threedream.oruonline	Education	0	0.0	100.0	100.0
2312942	Devi Suktam	ishan.devi.suktam	Music & Audio	3	8.0	1000.0	1000.0

1287191 rows × 25 columns

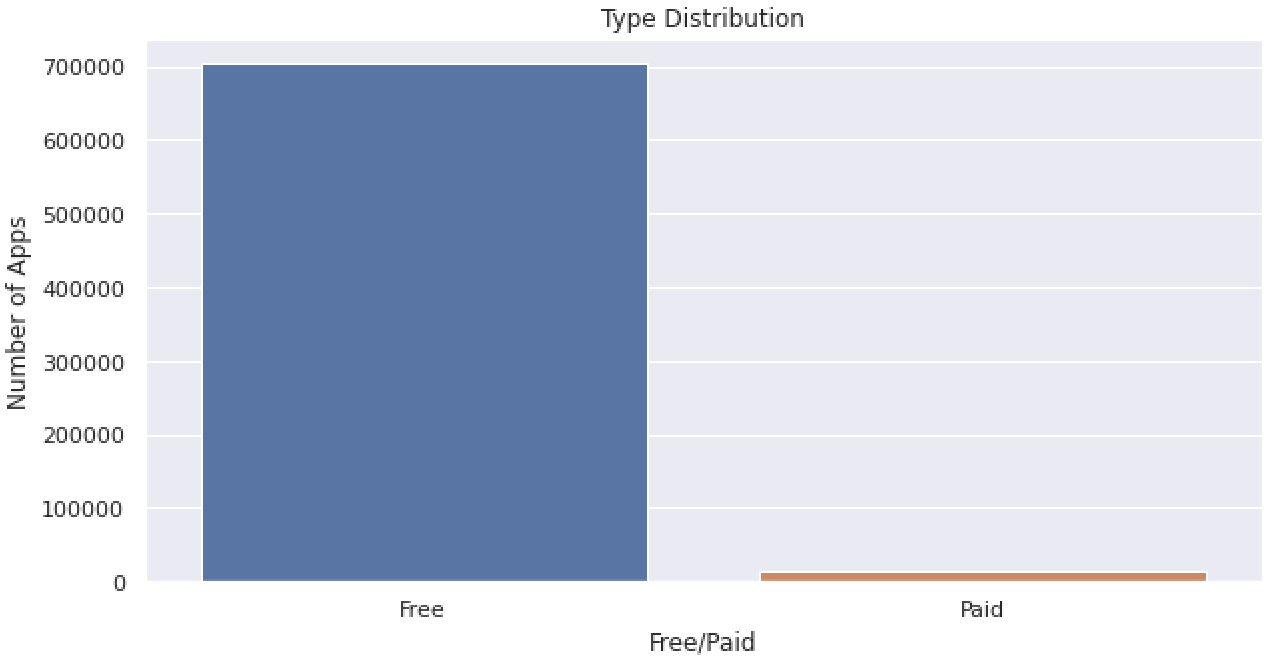
Previously there are only 24 columns now we create one more columns Free/Paid now this dataset have 25 columns.

```
# Group the data according to Free/Paid Apps
group_price = df.groupby('Free/Paid')
print('The average rating for the Free apps are {}.'.format(group_price['Rating'].mean(
print('The average rating for the Paid For apps are {}'.format(group_price['Rating'].me
```

```
The average rating for the Free apps are 2.0687411002729843.
The average rating for the Paid For apps are 2.076330639011743
```

It seems that the Paid apps have a higher average rating than the Free apps.

```
# application type distribution
plt.figure(figsize=(10, 5))
sns.countplot(df['Free/Paid'])
plt.title('Type Distribution')
plt.ylabel('Number of Apps')
plt.show()
```



This could be attributed to the larger amount of people who rated the Free apps as oppose to the Paid ones. The larger amount of people might account for the bigger spread of the rating, as more people could have rated the apps with a low rating.

4.Content Rating

```
df['Content Rating']
```

```
0          Everyone
1          Everyone
4          Everyone
5              Teen
9          Everyone
             ...
2312933    Everyone
2312934    Everyone
```

```
2312938    Everyone
2312940    Everyone
2312942    Everyone
Name: Content Rating, Length: 1287191, dtype: object
```

```
df["Content Rating"].value_counts()
```

```
Everyone           1122381
Teen                110037
Mature 17+           34244
Everyone 10+         20425
Adults only 18+         85
Unrated                 19
Name: Content Rating, dtype: int64
```

Everyone content rating have highest number of rating. Unrated content rating has lowest.

```
print('Top of choices of Adults only 18+')

top_choices_adults = df.loc[(df['Content Rating'] == 'Adults only 18+')]
top_choices_adults.sort_values(by='Maximum Installs', ascending = False).head(5)
```

```
Top of choices of Adults only 18+
```

	App Name	App Id	Category	Rating	Rating Count	Installs	Mir Ir
1772842	Yahoo Fantasy Sports: Football, Baseball & More	com.yahoo.mobile.client.android.fantasyfootball	Sports	4	326264.0	10000000.0	10000
73178	DraftKings - Daily Fantasy Sports for Cash	com.draftkings.dknativermgGP	Sports	4	81954.0	1000000.0	1000
571340	FanDuel Fantasy Sports	com.fanduel.android.self	Sports	4	43071.0	1000000.0	1000
1758248	LuckyCash - Win real money and coupons !	com.moneyapp.makemoney	Lifestyle	4	204936.0	1000000.0	1000
273744	Panda Cube Smash - Big Win with Lucky Puzzle G...	com.panda.unity.blastsaga	Puzzle	3	115226.0	1000000.0	1000

5 rows × 25 columns

```
print('Top of choices of Teens')
```

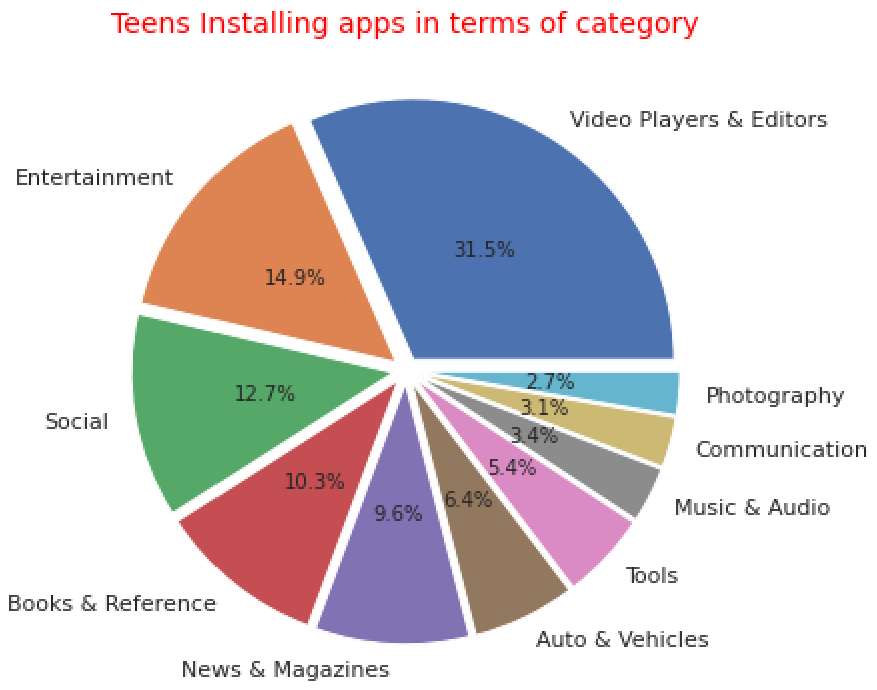
```
top_choices_teen = df.loc[(df['Content Rating'] == 'Teen')]
top_choices_teen.sort_values(by='Maximum Installs', ascending = False).head(5)
```

```
Top of choices of Teens
```

	App Name	App Id	Category	Rating	Rating Count	Installs	Minim Inst
881403	YouTube	com.google.android.youtube	Video Players & Editors	4	112440547.0	5.000000e+09	5.000000e-
167781	Google TV (previously Play Movies & TV)	com.google.android.videos	Video Players & Editors	4	1825673.0	5.000000e+09	5.000000e-
1643722	Google Play Games	com.google.android.play.games	Entertainment	4	12016421.0	1.000000e+09	1.000000e-
1108596	Currents	com.google.android.apps.plus	Social	4	6359366.0	1.000000e+09	1.000000e-
304824	Instagram	com.instagram.android	Social	3	120206190.0	1.000000e+09	1.000000e-

5 rows × 25 columns

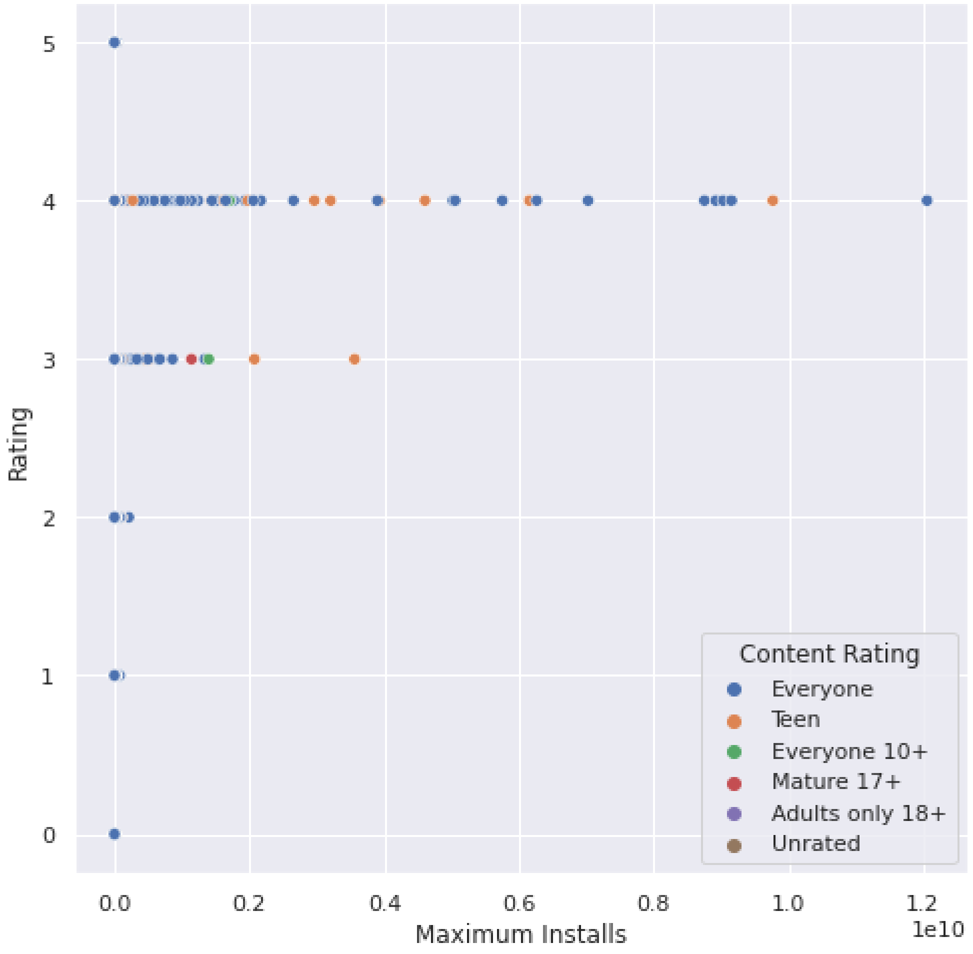
```
# Teens Installing apps in terms of category

teen_category = df[(df['Content Rating'] == 'Teen')]
teen_category = teen_category.groupby(['Category'])['Maximum Installs'].max().sort_valu
teen_category = teen_category.head(10)
pie,ax =plt.subplots(figsize = [10,6])
labels = teen_category.keys()

plt.pie(x=teen_category, autopct = '%.1f%%', explode=[0.05]*10, labels=labels, pctdista
plt.title("Teens Installing apps in terms of category", fontsize=14, color='red');

# Rest of the category have a low significance so they are not included.
```



This chart shows, In teen content rating, Video Players & Editors category have highest number of install.

```
plt.figure(figsize=(8,8))
sns.scatterplot(x='Maximum Installs',y='Rating',data=df,hue='Content Rating')
```

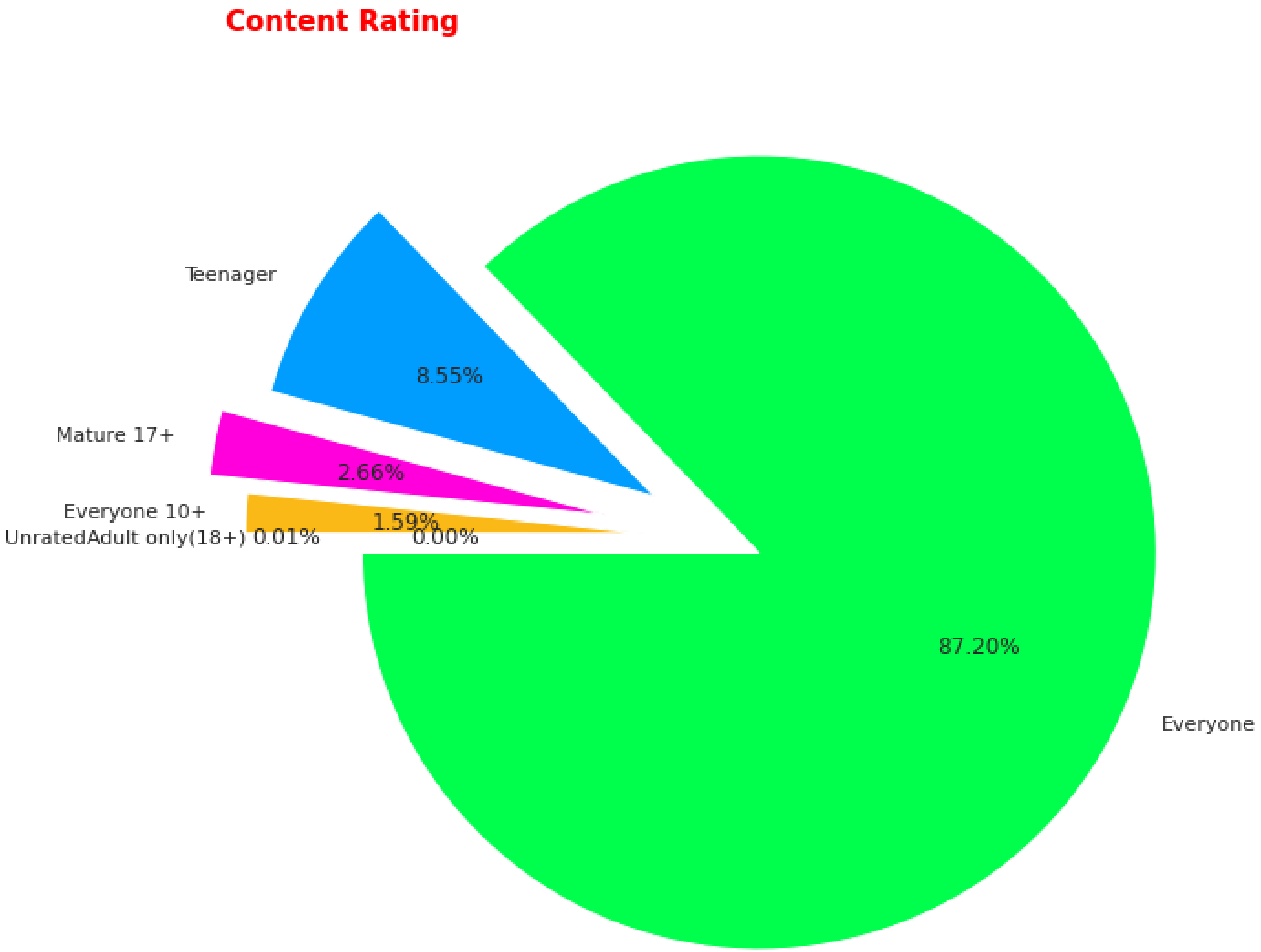
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc5037e3b10>
```



Everyone content rating have maximun installs as well as got highest rating but it also get lowest rating.

```
#  content rating of each category
plt.pie(df['Content Rating'].value_counts(),explode=[0.1,0.2,0.3,0.2,0.5,0.1],autopct='
        labels=['Everyone','Teenager','Mature 17+','Everyone 10+','Unrated','Adult only
        colors=['#00ff4c','#009dff','#ff00dd','#fbb917','#ff00ff','#ff002b'])
plt.title('Content Rating',fontdict={'size':15,'weight':'bold'},loc='left',color='red')
plt.plot()
```

```
[]
```



Content ratings on Google Play are provided by the International Age Rating Coalition (IARC) and are designed to help developers communicate locally relevant content ratings to users.

Regional IARC authorities maintain guidelines which are used to determine the maturity level of the content in an app.

So,Everyone Content ratings category have share approx 87% which is highest in all of the category.

5.Ad Supported and In App Purchases

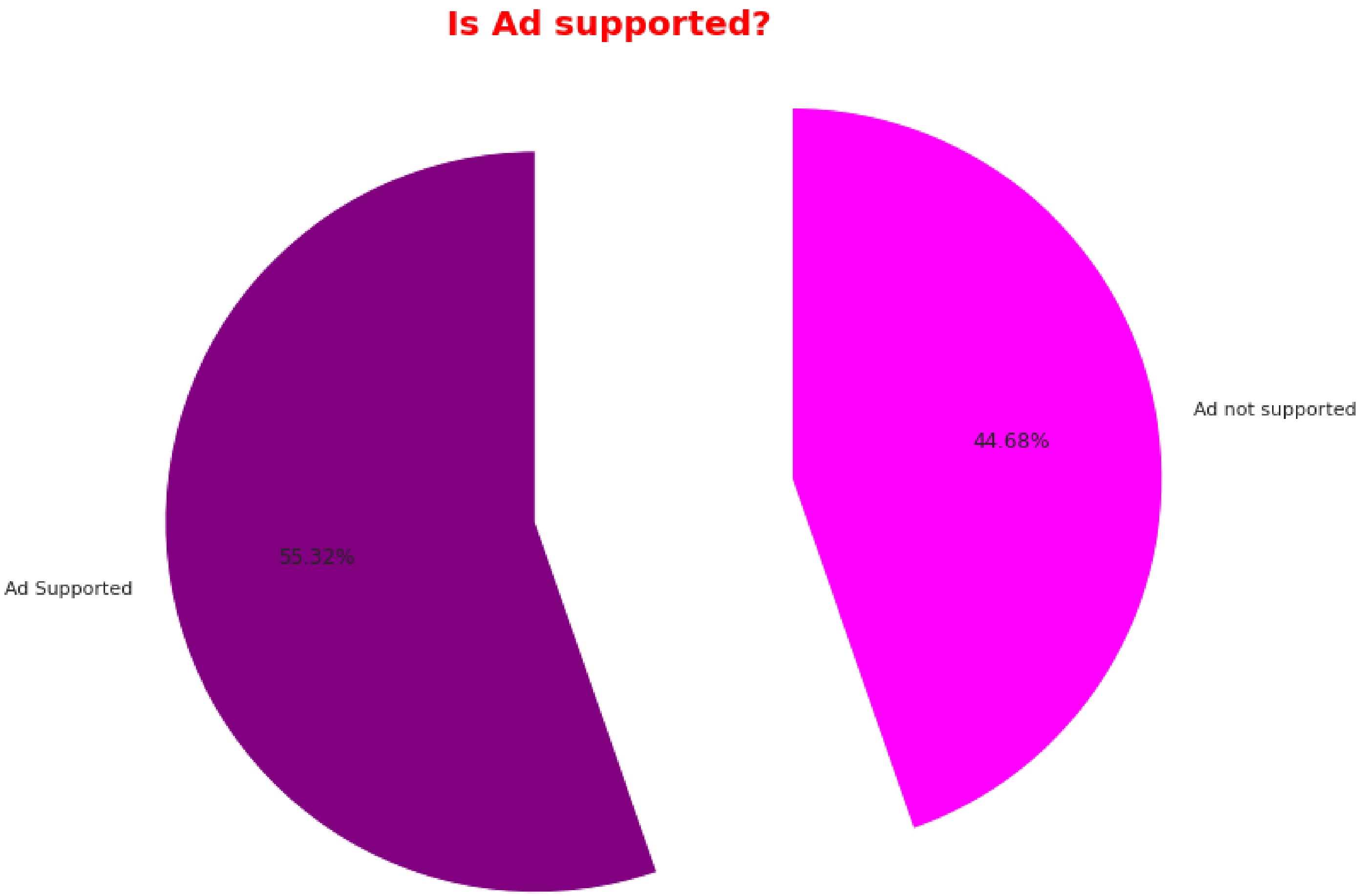
```
df['Ad Supported'].value_counts()
```

```
False     712073
True      575118
```

```
Name: Ad Supported, dtype: int64
```

```
plt.pie(df['Ad Supported'].value_counts(),radius=1,autopct='%0.2f%%',
        explode=[0.2,0.5],colors=['#800080','#ff00ff'],
        labels=['Ad Supported','Ad not supported'],startangle=90)
plt.title('Is Ad supported?',fontdict={'size':20,'weight':'bold'},color='red')
plt.plot()
```

```
[]
```



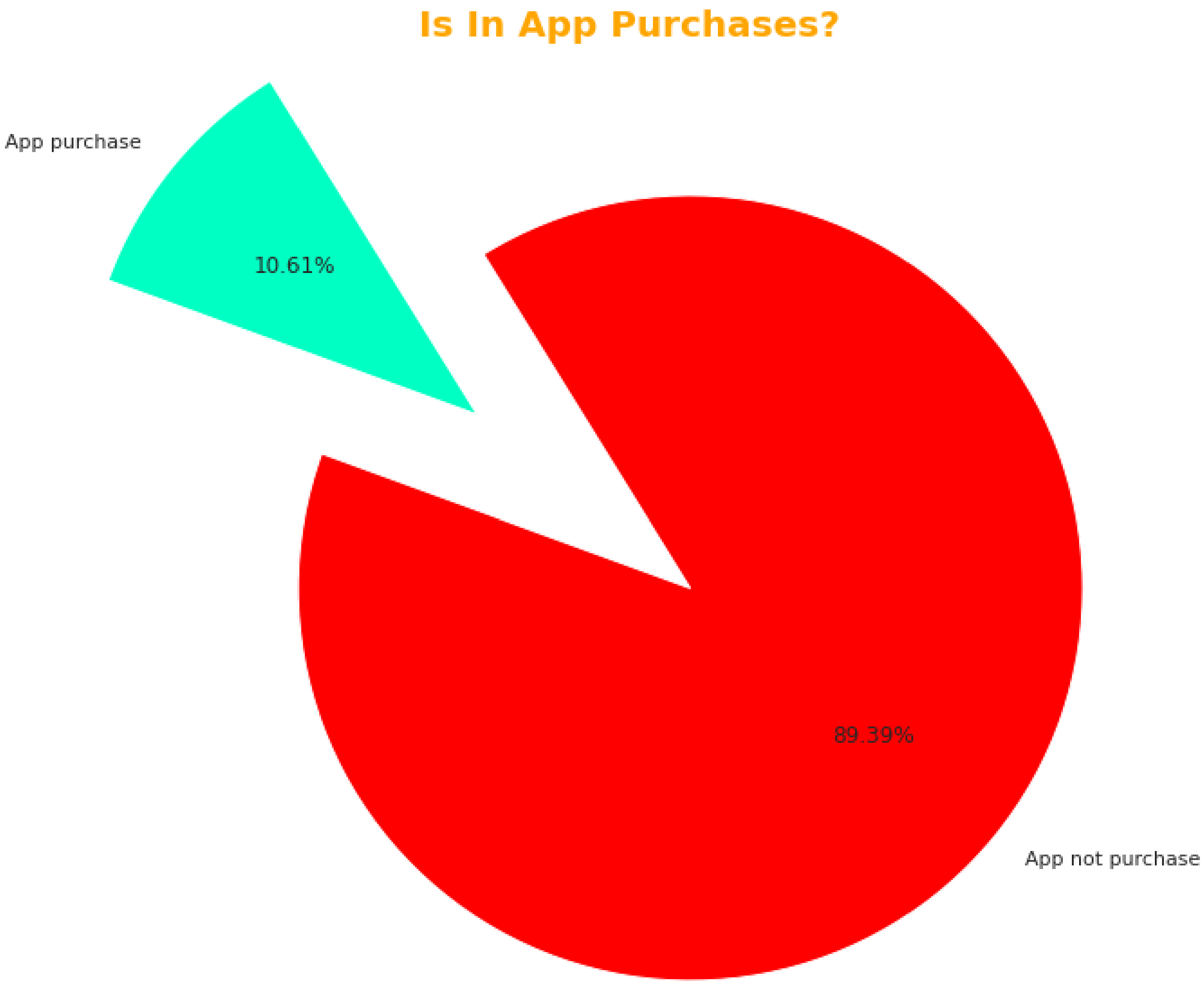
Most of apps are ad supported.

```
df['In App Purchases'].value_counts()
```

```
False    1150584
True      136607
Name: In App Purchases, dtype: int64
```

```
plt.pie(df['In App Purchases'].value_counts(),radius=1,autopct='%0.2f%%',
        explode=[0.2,0.5],colors=['#ff0000','#00ffc3'],
        labels=['App not purchase','App purchase'],startangle=160)
plt.title('Is In App Purchases?',fontdict={'size':20,'weight':'bold'},color='orange')
plt.plot()
```

```
[]
```



In total number of apps only approx 11% apps are purchase.

Ask & Answer Questions

1. What is highest rated category?
2. What year has the most number of apps released?
3. Does the average apps size change over the years and which app have maximum size?
4. What is the percentage proportion between free and paid apps?
5. What are the top 10 categories with most rating how many apps were free and paid in this category?
6. Which category are maximum and minimum average price?
7. What are the top 5 apps on the maximum number of installs?
8. Which app have maximum number of installs in productivity category and also list the top 5 apps name.
9. What is the most month, day and weekday apps get updated?
10. How many apps can work on android version 4?
11. Which are the apps that have made the highest earning in dataset?

1.What is highest rated category?

```
rated_category = df.groupby('Category').mean().sort_values('Rating', ascending = False)
rated_category
```

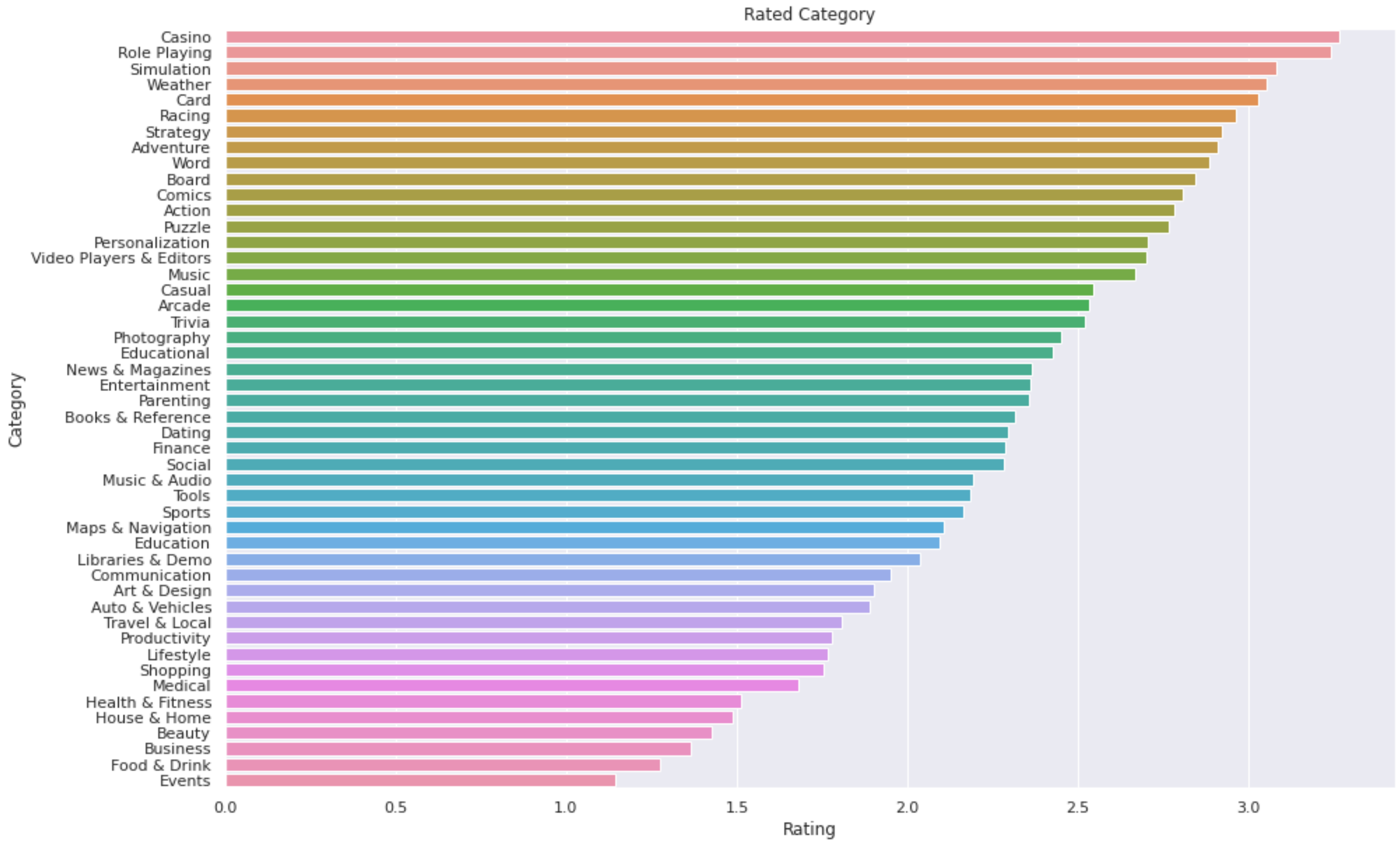
	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Suppo
Category								
Casino	3.264413	15446.556940	3.824626e+05	3.824626e+05	6.836868e+05	0.986833	0.049690	0.816
Role Playing	3.240413	21107.953773	7.495091e+05	7.495091e+05	1.364023e+06	0.912275	0.416638	0.732
Simulation	3.079720	16476.221895	8.065992e+05	8.065992e+05	1.366358e+06	0.963243	0.155310	0.890
Weather	3.051426	6714.326834	4.387615e+05	4.387615e+05	7.689832e+05	0.970753	0.172961	0.718
Card	3.028613	9388.739605	3.546714e+05	3.546714e+05	5.968861e+05	0.960066	0.160812	0.800
Racing	2.963589	31413.471484	1.597923e+06	1.597923e+06	2.946558e+06	0.981423	0.047912	0.913
Strategy	2.920087	54069.378431	9.203962e+05	9.203962e+05	1.696129e+06	0.915230	0.285655	0.691
Adventure	2.908014	11865.362797	4.637283e+05	4.637283e+05	8.510392e+05	0.931423	0.298970	0.782
Word	2.882742	10279.170575	3.760803e+05	3.760803e+05	6.424314e+05	0.974945	0.065460	0.907
Board	2.843285	10656.794783	5.561943e+05	5.561943e+05	8.804166e+05	0.952077	0.274375	0.785
Comics	2.805204	4387.811896	2.213437e+05	2.213437e+05	4.044997e+05	0.975465	0.076840	0.740
Action	2.780379	39908.844651	1.255225e+06	1.255225e+06	2.253728e+06	0.968187	0.095863	0.855
Puzzle	2.765647	7846.629582	3.961671e+05	3.961671e+05	7.005177e+05	0.967530	0.087405	0.878
Personalization	2.705228	2421.643856	1.627295e+05	1.627295e+05	2.968963e+05	0.964595	0.080508	0.895
Video Players & Editors	2.698242	36255.042133	2.470273e+06	2.470273e+06	4.149891e+06	0.974430	0.116901	0.633
Music	2.668182	10290.352727	6.665484e+05	6.665484e+05	1.347570e+06	0.958636	0.102995	0.837
Casual	2.542946	15117.147509	7.025761e+05	7.025761e+05	1.282254e+06	0.978823	0.055781	0.854
Arcade	2.532196	11305.388507	6.711048e+05	6.711048e+05	1.192086e+06	0.978369	0.054867	0.855
Trivia	2.518946	5793.960748	1.719811e+05	1.719811e+05	3.243612e+05	0.989975	0.027813	0.887
Photography	2.450051	11393.510430	9.543937e+05	9.543937e+05	1.530229e+06	0.979378	0.088771	0.791
Educational	2.425889	2145.637646	3.458873e+05	3.458873e+05	6.051161e+05	0.930411	0.227666	0.531
News & Magazines	2.363029	1009.197544	1.455383e+05	1.455383e+05	2.829182e+05	0.998124	0.005086	0.611
Entertainment	2.360343	2995.274485	2.106492e+05	2.106492e+05	4.108678e+05	0.989934	0.041082	0.658
Parenting	2.356993	2569.350693	8.857886e+04	8.857886e+04	1.767226e+05	0.972701	0.086077	0.442
Books & Reference	2.313598	966.267770	6.321703e+04	6.321703e+04	1.437385e+05	0.969849	0.215175	0.761
Dating	2.295713	3812.272383	1.644528e+05	1.644528e+05	3.375663e+05	0.996792	0.143742	0.597
Finance	2.286006	3737.195960	1.153053e+05	1.153053e+05	2.161939e+05	0.994287	0.045564	0.204
Social	2.281291	12374.409476	4.087869e+05	4.087869e+05	8.604942e+05	0.994699	0.045669	0.355
Music & Audio	2.194028	1750.185684	1.337153e+05	1.337153e+05	2.133478e+05	0.990070	0.050428	0.784
Tools	2.184259	5813.115807	7.530543e+05	7.530543e+05	1.241538e+06	0.974782	0.139118	0.437
Sports	2.162543	7191.036052	2.537919e+05	2.537919e+05	4.532203e+05	0.974674	0.197818	0.474
Maps & Navigation	2.106378	3267.100563	1.450437e+05	1.450437e+05	2.473730e+05	0.972505	0.177829	0.236
Education	2.094624	724.014746	3.800498e+04	3.800498e+04	6.912506e+04	0.970468	0.196650	0.377

	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Suppo
Category								
Libraries & Demo	2.035006	1253.100801	9.797967e+04	9.797967e+04	2.110882e+05	0.994939	0.028001	0.331
Communication	1.948474	14498.868058	1.223719e+06	1.223719e+06	1.882691e+06	0.991271	0.043122	0.273
Art & Design	1.902335	1756.463345	1.122746e+05	1.122746e+05	1.777412e+05	0.989259	0.087897	0.667
Auto & Vehicles	1.890637	1027.820923	1.383338e+05	1.383338e+05	2.662689e+05	0.985387	0.112822	0.203
Travel & Local	1.807886	1141.933216	9.135874e+04	9.135874e+04	1.852283e+05	0.978654	0.081739	0.224
Productivity	1.779238	2814.253940	4.997318e+05	4.997318e+05	8.182463e+05	0.980175	0.150094	0.217
Lifestyle	1.767399	1073.937308	6.608853e+04	6.608853e+04	1.125675e+05	0.988976	0.075540	0.302
Shopping	1.753598	3957.900492	1.283540e+05	1.283540e+05	2.420028e+05	0.998898	0.002980	0.139
Medical	1.680503	549.311352	2.726977e+04	2.726977e+04	4.518396e+04	0.962192	0.949973	0.151
Health & Fitness	1.510663	1354.468515	7.817202e+04	7.817202e+04	1.374228e+05	0.985281	0.077985	0.196
House & Home	1.488458	740.079706	4.734438e+04	4.734438e+04	9.213666e+04	0.995521	0.019768	0.286
Beauty	1.427901	612.305066	4.774960e+04	4.774960e+04	8.411697e+04	0.998867	0.004035	0.373
Business	1.363250	554.849109	3.785982e+04	3.785982e+04	6.433359e+04	0.996381	0.047647	0.096
Food & Drink	1.274725	1007.517729	3.049261e+04	3.049261e+04	5.495763e+04	0.997946	0.007865	0.151
Events	1.144992	86.567141	6.930926e+03	6.930926e+03	1.346970e+04	0.999529	0.002408	0.197

Answer

Casino category is the highest rated category.

```
sns.set(rc= {'figure.figsize':(15,10)})
sns.barplot(x="Rating",y= rated_category.index, data = rated_category)
plt.title('Rated Category');
```



Casino category is the highest rated category and Events category is the less rated category.

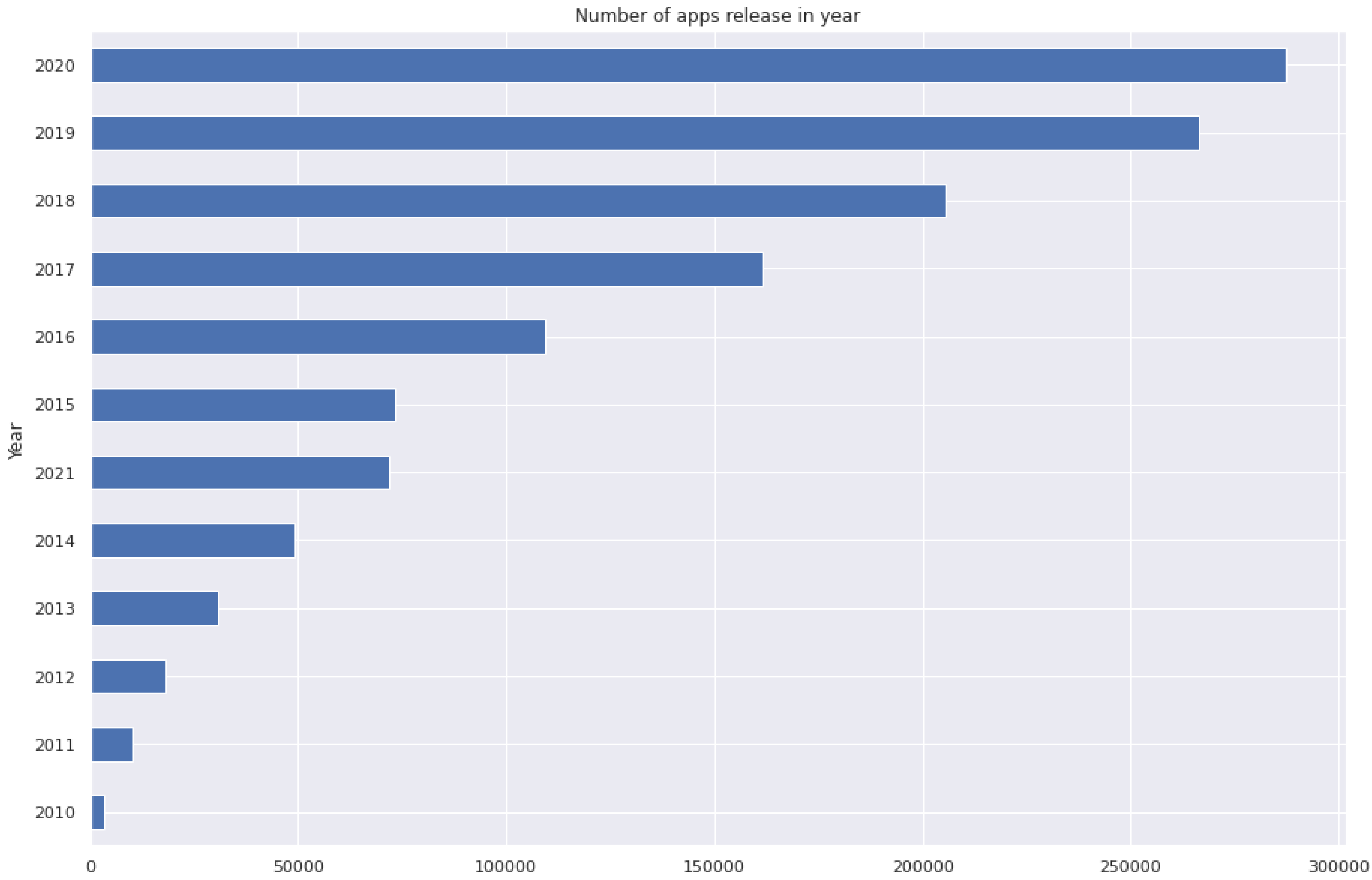
2.What year has the most number of apps released?

```
df['year'] = pd.DatetimeIndex(df['Released']).year
```

```
df.year
```

```
0          2020
1          2020
4          2020
5          2018
9          2019
           ...
2312933    2014
2312934    2020
2312938    2018
2312940    2018
2312942    2016
Name: year, Length: 1287191, dtype: int64
```

```
df['year'].value_counts(ascending = True).plot(kind ='barh',title = 'Number of apps rel
```



Answer

Year 2020 released most number of apps.

We can find total app numbers increases on yearly basis, as data shows, the number of apps in the last couples of years is approximately equal to all the apps combined in all previous years !

3.Does the average apps size change over the years and which app have maximum size?

```
df.Size.fillna('00M',inplace = True)
```

```
import numpy as np
```

```
#remove **nan** and from Size and creating new column **size_in_KB**
#that contains all sizes converted to **K** and converted to float datatype


df['size_u'] = np.nan
for i in df.index:
  if 'M' in df.Size[i]:
    df['size_u'][i] = df.Size[i].replace('M','').replace(',','')
    df['size_u'][i] = float(df['size_u'][i])
    df['size_u'][i] = df['size_u'][i]*1000

  elif 'k' in df.Size[i]:
      df['size_u'][i]= df.Size[i].replace('k','').replace(',','')
```

```
        df['size_u'][i]= float(df['size_u'][i])

    elif 'G' in df.Size[i]:
        df['size_u'][i]= df.Size[i].replace('G','').replace(',','')
        df['size_u'][i]= float(df['size_u'][i])
        df['size_u'][i]= df['size_u'][i]*1000000

    else:
        df['size_u'][i]=np.nan
```

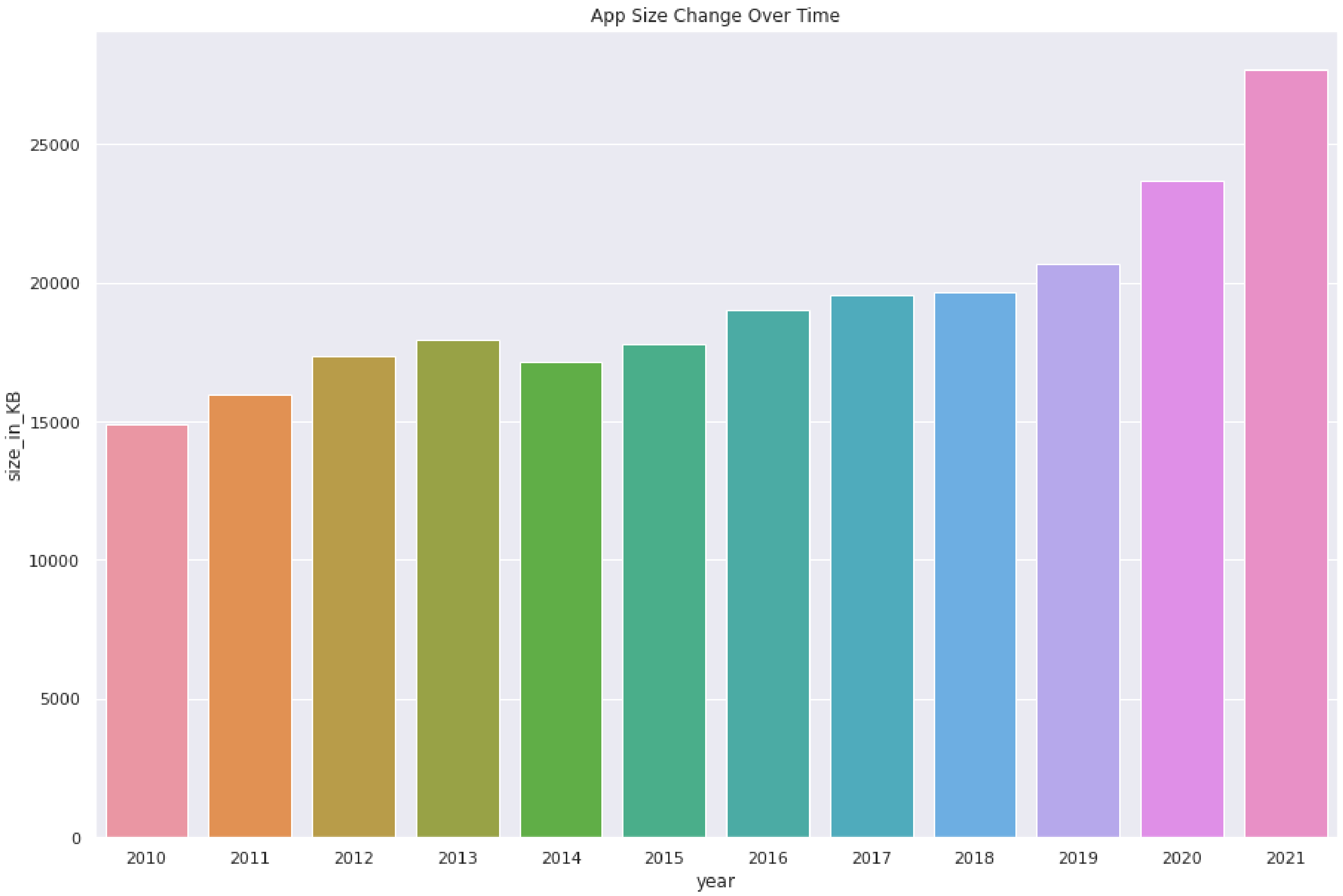
```
df.rename(columns={"size_u": "size_in_KB"},inplace=True)
```

```
df['size_in_KB'].fillna(0, inplace=True)
```

```
data_size=df.groupby('year').mean()
data_size
```

	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Ad Supported	Pur
year									
2010	3.346760	154574.901343	1.041326e+07	1.041326e+07	1.855876e+07	0.845301	0.867685	0.511967	0.2
2011	3.255160	34684.957166	2.585970e+06	2.585970e+06	4.431800e+06	0.874697	0.719111	0.479891	0.1
2012	3.167197	38824.198224	2.337961e+06	2.337961e+06	3.961945e+06	0.904506	0.581267	0.468753	0.1
2013	3.037188	23854.138558	1.242169e+06	1.242169e+06	2.288286e+06	0.931669	0.331843	0.472043	0.1
2014	2.941643	14879.161748	7.169854e+05	7.169854e+05	1.167376e+06	0.949250	0.274149	0.462177	0.1
2015	2.720087	8284.074792	5.675256e+05	5.675256e+05	9.187220e+05	0.970900	0.145563	0.449338	0.1
2016	2.455360	5275.133188	2.502832e+05	2.502832e+05	4.564403e+05	0.977036	0.130615	0.421842	0.1
2017	2.240850	3616.847431	1.909556e+05	1.909556e+05	3.344875e+05	0.982655	0.104708	0.419949	0.1
2018	2.036604	2001.599439	1.377538e+05	1.377538e+05	2.493870e+05	0.986573	0.092671	0.403417	0.0
2019	1.790340	1183.750062	9.007078e+04	9.007078e+04	1.551671e+05	0.989907	0.068127	0.429567	0.0
2020	1.748831	628.695398	5.041881e+04	5.041881e+04	9.058681e+04	0.991651	0.058686	0.478307	0.0
2021	1.290164	297.118023	2.130417e+04	2.130417e+04	4.121285e+04	0.997522	0.013682	0.570080	0.0

```
sns.barplot(x=data_size.index,y='size_in_KB',data=data_size);
plt.title('App Size Change Over Time');
```



Answer

Yes, As expected, generally average apps size increases with Time. Year 2021 has maximum apps size.

```
df[df.size_in_KB == df.size_in_KB.max()]
```

	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Pri
93175	Titan Quest: Legendary Edition	com.hg.titanquestedition	Role Playing	4	1387.0	5000.0	5000.0	8329	False	19.9
529736	Titan Quest	com.dotemu.titanquest	Action	4	24339.0	100000.0	100000.0	223859	False	7.9

2 rows × 27 columns

Answer

Maximum size for the given data is **Titan Quest** by **HandyGames**.

4.What is the percentage proportion between free and paid apps?

```
data_free=df.groupby('Free').count()
data_free
```

	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Price	Currency	...	U

Free	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Price	Currency	...	U
Free												
False	23625	23625	23625	23625	23625	23625	23625	23625	23625	23625	...	
True	1263566	1263566	1263566	1263566	1263566	1263566	1263566	1263566	1263566	1263566	...	12

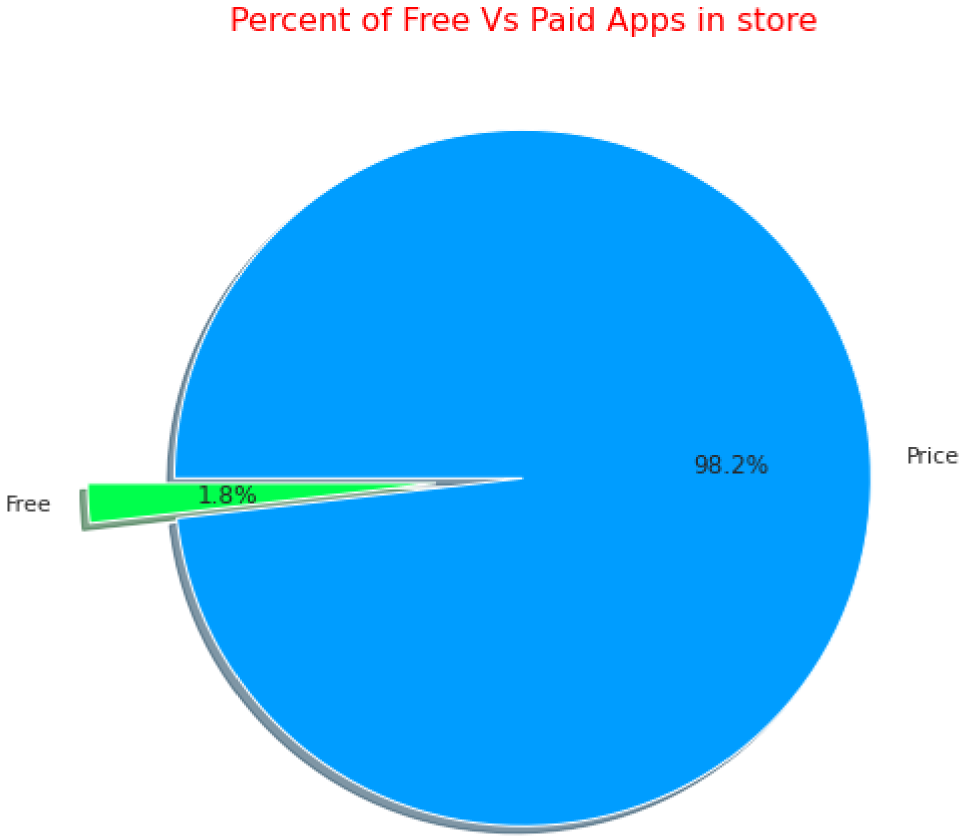
2 rows × 26 columns

```
data_free['Category'][1]/(data_free['Category'][1]+data_free['Category'][0])*100
```

```
98.16460804962122
```

Around 98 % of the apps are free.

```
plt.figure(figsize=(8,8))
count = data_free['Category']
plt.pie(count, explode=(0.25,0), labels=['Free', 'Price'], autopct='%1.1f%%', shadow=Tr
        colors=['#00ff4c','#009dff'])
plt.title('Percent of Free Vs Paid Apps in store', size = 16,color='red')
plt.show()
```



Answer

Approx 98% of apps are free and nearly 2% apps are paid.

5.What are the top 10 categories with most rating how many apps were free and paid in this category?

```
Rating_Cat = df.groupby(['Category'])['Rating'].sum().sort_values(ascending=False).rese
Rating_Cat
```

	Category	Rating
0	Education	266476
1	Music & Audio	190898
2	Entertainment	150073
3	Tools	146815
4	Business	136749
5	Lifestyle	132585
6	Books & Reference	130753
7	Personalization	102309
8	Finance	100838
9	Shopping	85893

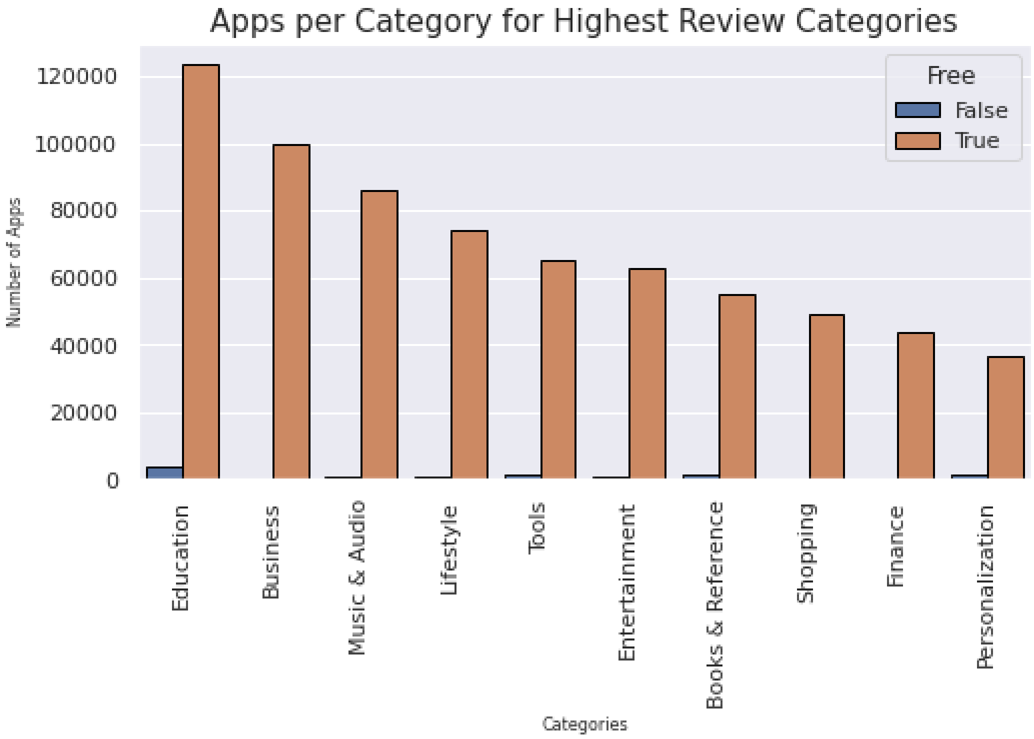
```
Rating_Cat2 = df[df['Category'].isin(Rating_Cat.Category)]
Rating_Cat2.head()
```

	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs	Maximur Install
1	Ampere Battery Info	com.webserveis.batteryinfo	Tools	4	64.0	5000.0	5000.0	766
4	GROW.me	com.horodyski.grower	Tools	0	0.0	100.0	100.0	47
9	Neon 3d Iron Tech Keyboard Theme	com.ikeyboard.theme.neon_3d.iron.tech	Personalization	4	820.0	50000.0	50000.0	6243
23	Coloring Book Barbaie	com.bisgumah.barbie	Entertainment	3	736.0	500000.0	500000.0	64645
39	Sudan Flag Wallpaper: Flags, Country HD Images	com.techzit.sudanflagwallpaper	Personalization	0	0.0	100.0	100.0	46

5 rows × 27 columns

```
plt.subplots(figsize=(8, 4))

sns.countplot(x = Rating_Cat2['Category'],order = Rating_Cat2['Category'].value_counts(
              hue = Rating_Cat2['Free'], edgecolor = (0,0,0))
```

```
plt.xlabel('Categories', fontsize = 8)
plt.xticks(rotation = 90)
plt.ylabel('Number of Apps', fontsize = 8)
plt.title('Apps per Category for Highest Review Categories', fontsize = 15)

plt.show()
```



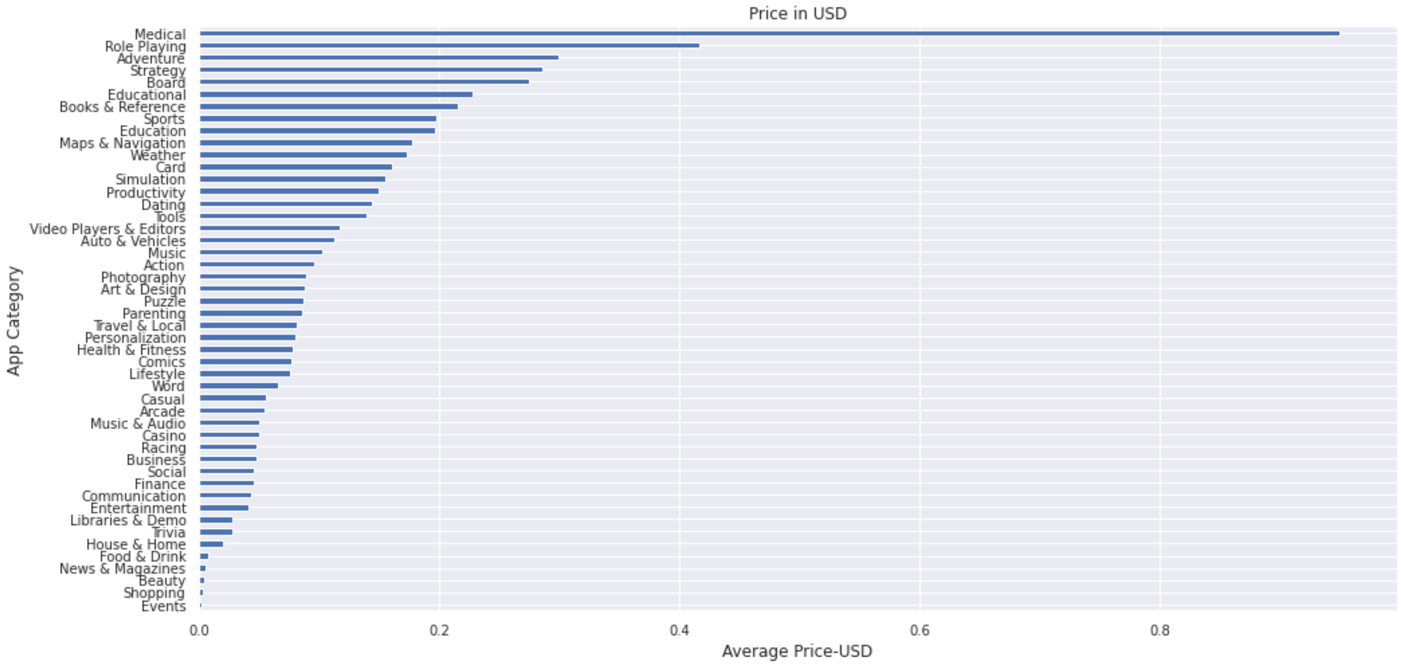
Answer

Education category have maximum number of free apps and Education have maximum number of paid apps.

6.Which category are maximum and minimum average price?

```
fig = plt.figure(figsize=(16,8))
df.groupby('Category').mean().sort_values(by='Price',ascending = True)['Price'].plot(ki
plt.ylabel('App Category')
plt.xlabel('Average Price-USD')
```

```
Text(0.5, 0, 'Average Price-USD')
```



Answer

Medical App category has maximum average price and **Events app** category has minimum average price.

7.What are the top 5 apps on the maximum number of installs?

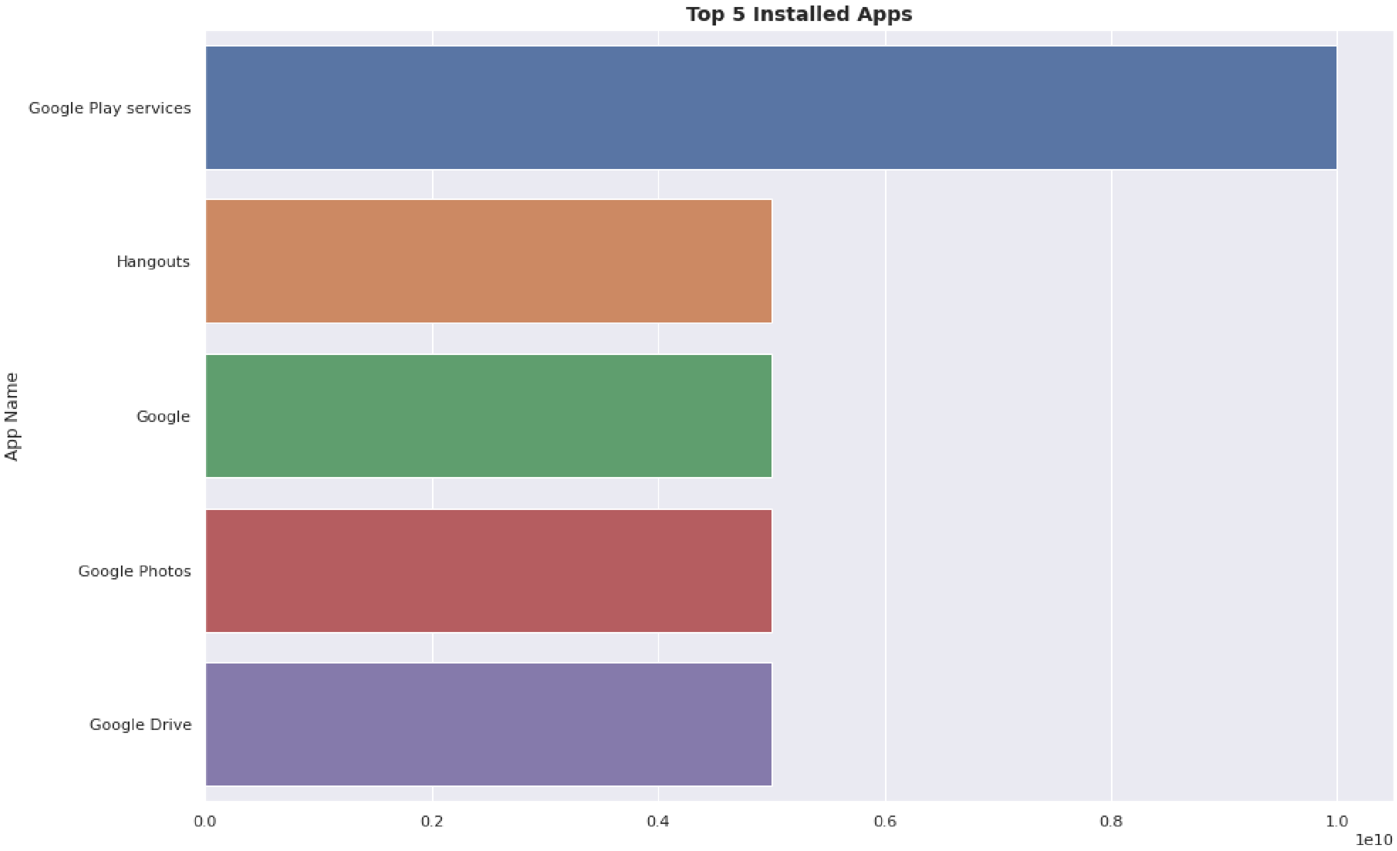
```
df1 = df.sort_values(by=['Installs'], ascending=False)
df1.head(5)
```

	App Name	App Id	Category	Rating	Rating Count	Installs
2155096	Google Play services	com.google.android.gms	Tools	4	35128398.0	1.000000e+10
1773294	Google	com.google.android.googlequicksearchbox	Tools	4	19798962.0	5.000000e+09
167781	Google TV (previously Play Movies & TV)	com.google.android.videos	Video Players & Editors	4	1825673.0	5.000000e+09
944254	Google Chrome: Fast & Secure	com.android.chrome	Communication	4	31481796.0	5.000000e+09
893676	Google Drive	com.google.android.apps.docs	Productivity	4	639307.0	5.000000e+09

5 rows × 27 columns

```
sns.set_style("darkgrid")
x=df1.groupby("App Name").Installs.sum().sort_values(ascending=False).head(5)
sns.barplot(x.values,x.index)
```

```
plt.title("Top 5 Installed Apps", fontdict= { 'fontsize': 14,'fontweight':'bold'})
plt.show()
```



Answer

The 5 apps that have the most number of installs are: Google Play services, Google, Google TV (previously Play Movies & TV), Google Chrome: Fast & Secure, Google Drive

8.Which app have maximum number of installs in productivity category and also list the top 5 apps name.

```
productivity = df[df['Category']==('Productivity')]
print('The dataset contains '+ (str((productivity['App Name'].nunique())))+ ' productiv
```

```
The dataset contains 46486 productivity apps.
```

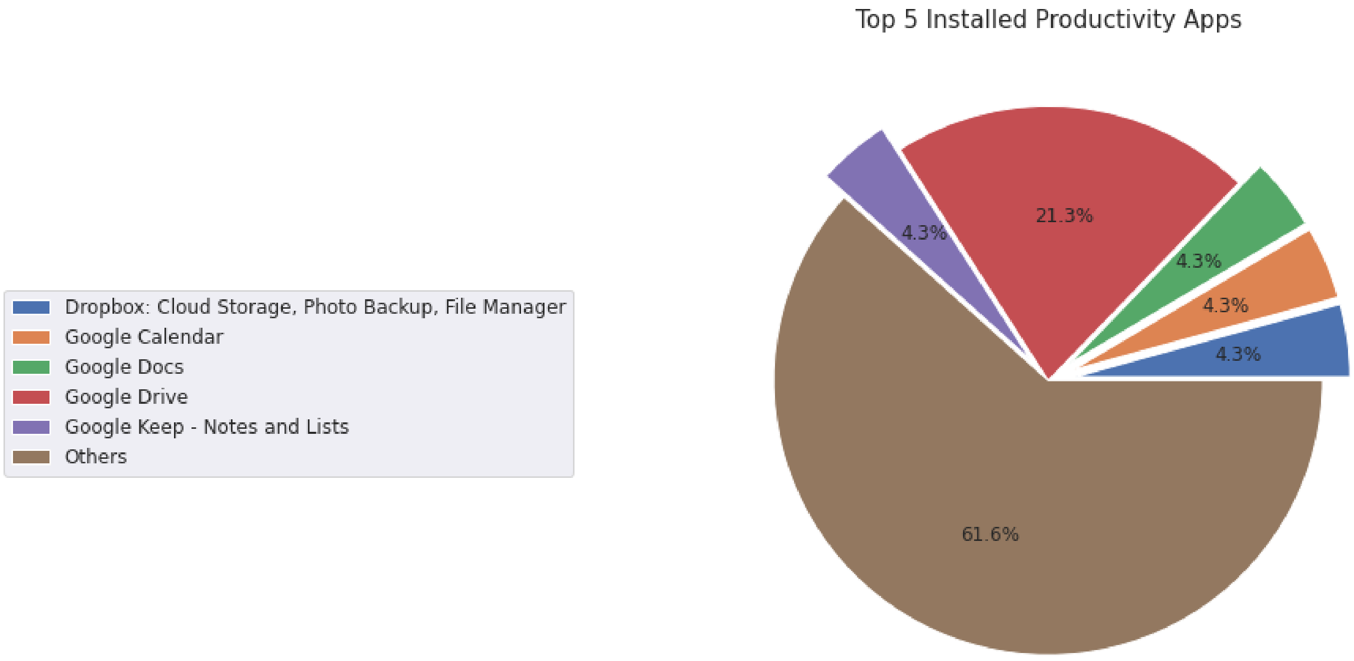
```
top_install = productivity.groupby("App Name")["Installs"].sum().nlargest(5).index.toli
top_install
```

```
['Google Drive',
 'Dropbox: Cloud Storage, Photo Backup, File Manager',
 'Google Calendar',
 'Google Docs',
 'Google Keep - Notes and Lists']
```

```
def fuc(row):
    if row["App Name"] not in top_install:
        return("Others")
    else:
        return row['App Name']
```

```
productivity['Top_Installed_Product'] = productivity.apply(fuc,axis=1)
#productivity.head()
```

```
top_install = productivity.groupby("Top_Installed_Product")["Installs"].sum()

plt.subplots(figsize = (8, 8))
explode = (0.1,0.1,0.1,0,0.1,0)

top_install_index = top_install.index
labels = top_install.index

ax.set(facecolor = 'grey')
pie = plt.pie(top_install, explode = explode, startangle = 0, autopct = '%1.1f%%')
plt.legend(pie[0], labels, bbox_to_anchor = (0, 0.5), loc = "center right",
           fontsize = 12, bbox_transform = plt.gcf().transFigure)
plt.title('Top 5 Installed Productivity Apps', fontsize = 15)

plt.show()
```



Answer

Google Drive is the maximum number of installs approx 21% install share in total. The list of 5 apps name are:-

1.Google Drive

2.Dropbox: Cloud Storage, Photo Backup, File Manager

3.Google Calendar

4.Google Docs

5.Google Keep - Notes and Lists

9.What is the most month, day and weekday apps get updated?

```
df['Last Updated'] = pd.to_datetime(df['Last Updated'])
df['Month'] = df['Last Updated'].dt.month
```

```
df['MonthDay'] = df['Last Updated'].dt.day
df['WeekDay'] = df['Last Updated'].dt.weekday
```

```
#import plotly.express as px
```

```


def most_frequent(col, data = df):

  # grouping the dataset by a giving feature.
    dataset = data[col].value_counts().to_frame().reset_index().head(20).rename(columns

  # ploting a Plotly bar chart that shows the number of releases.
    fig = px.bar(dataset ,x=col, y='Count', barmode='group', color='Count',
                 color_continuous_scale='plotly3_r')

     # formating.
    fig.update_layout(autosize=False, width=750, height=500, xaxis_title=col,
                      yaxis_title="Count")
    fig.show()
```

```
most_frequent('Month')
```

Answer

In May(5) month most of apps get updated.

```
most_frequent('MonthDay')
```

Answer

Most of apps get updated 14th day of month.

```
most_frequent('WeekDay')
```

Answer

Most of the apps get updated on Monday(1).

10.How many apps can work on android version 4?

```
def working_on_v4(version) :
    try :
        if version.startswith('4') :
            return 'Yes'
        else :
            return 'No'

    except :
        return np.nan
```

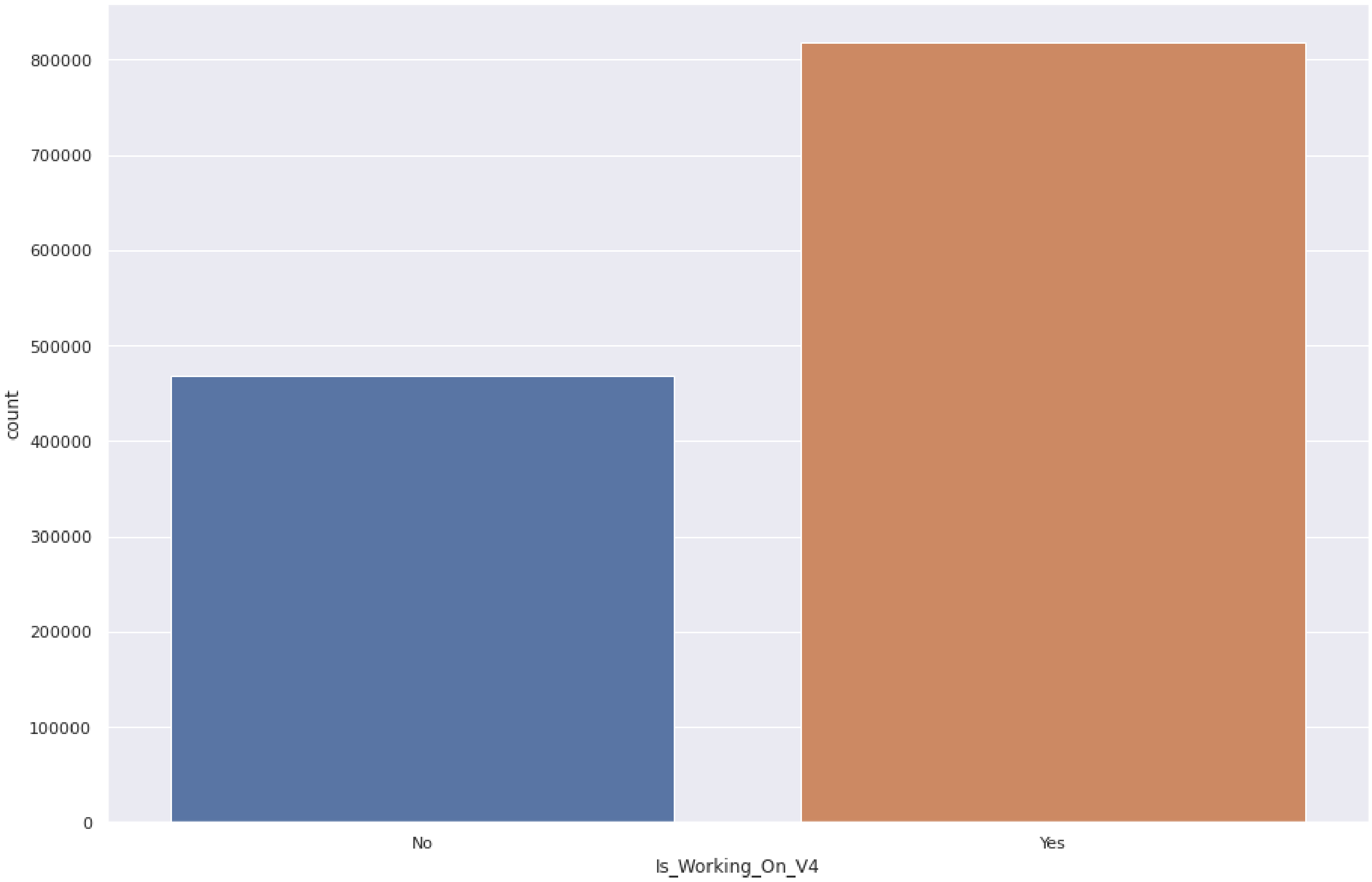
```
df['Is_Working_On_V4'] = df['Minimum Android'].apply(working_on_v4)
```

```
df['Is_Working_On_V4'].value_counts()
```

```
Yes    818607
No     468584
Name: Is_Working_On_V4, dtype: int64
```

```
sns.countplot(x='Is_Working_On_V4',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc501fe8d50>
```



Answer

Around 800K+ apps are working on version 4 and approx 450K+ are not working on version 4.

11.Which are the apps that have made the highest earning in dataset?

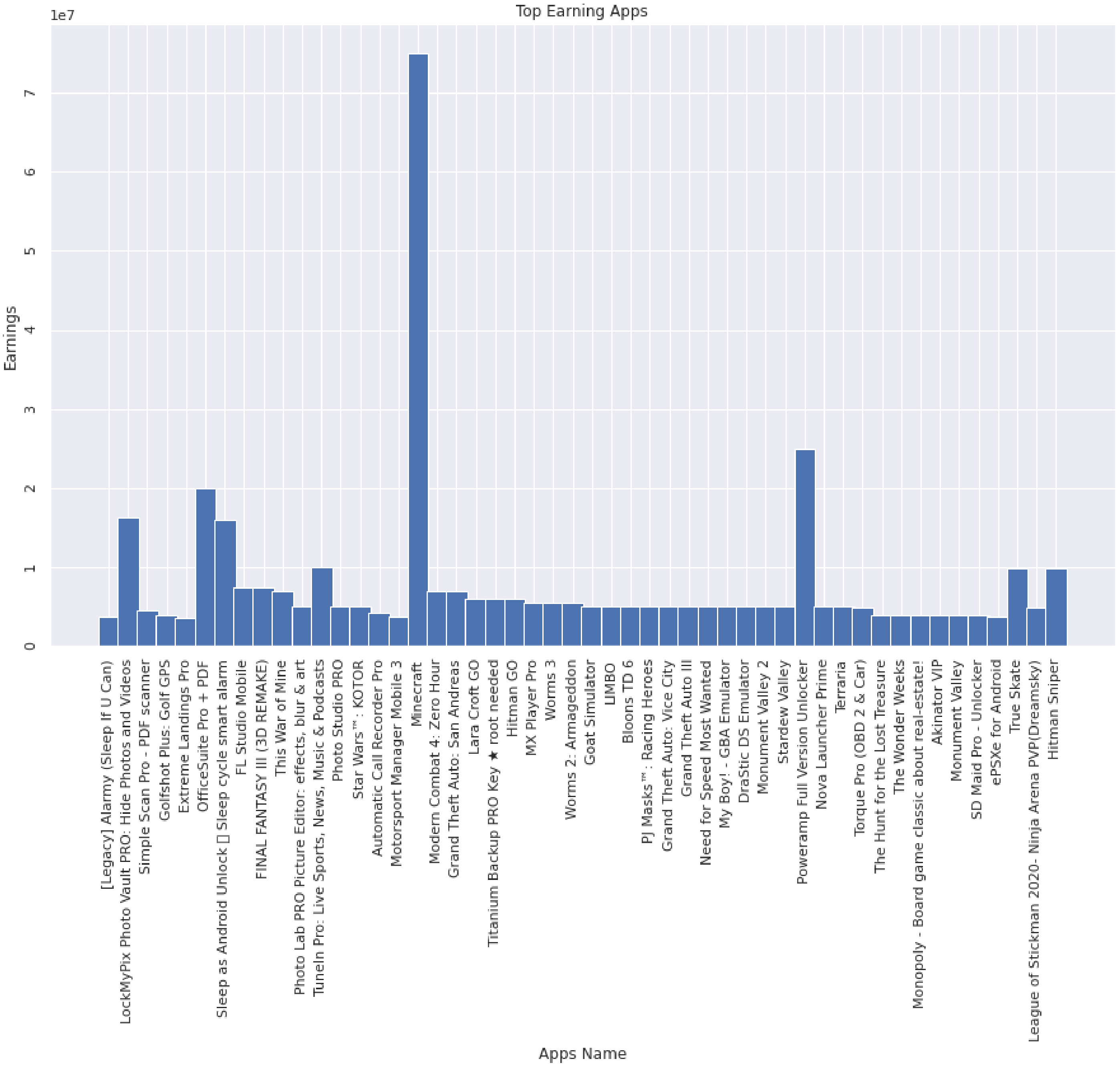
```
earning_df =df[['App Name', 'Installs', 'Price']]
```

```
earning_df['Earnings'] = earning_df['Installs'] * earning_df['Price'];
```

```
earning_df_sorted_by_Earnings = earning_df.sort_values(by='Earnings', ascending=False).
```

```
earning_df_sorted_by_Price = earning_df_sorted_by_Earnings.sort_values(by='Price', asce
```

```
# PLot a bar chart of earning at y and app names at x
plt.figure(figsize=(15,9))
plt.bar(earning_df_sorted_by_Price['App Name'], earning_df_sorted_by_Price.Earnings, wi
        label=earning_df_sorted_by_Price.Earnings)
plt.xlabel("Apps Name")
plt.ylabel("Earnings")
plt.tick_params(rotation=90)
plt.title("Top Earning Apps");
```



Answer

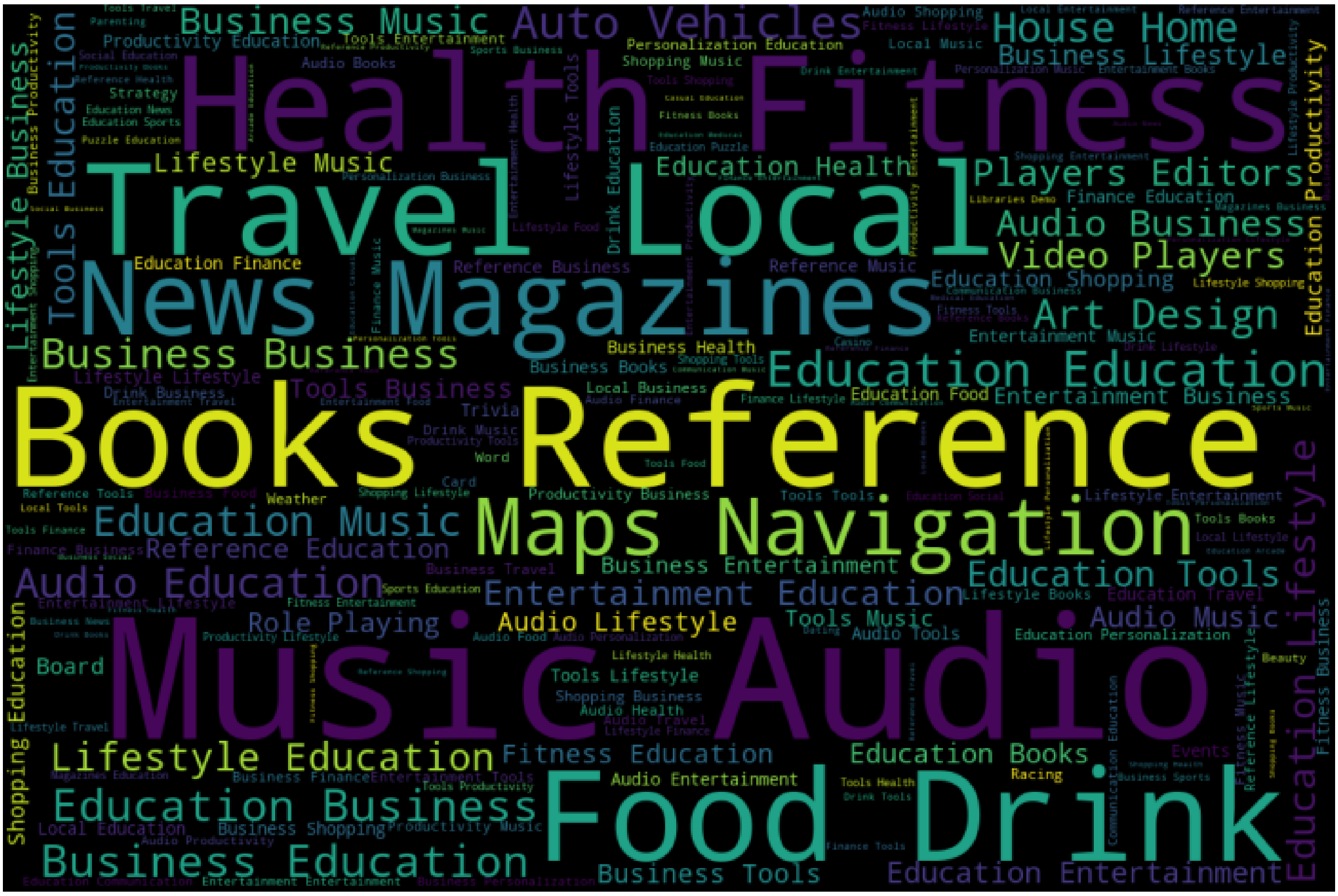
The top three apps with highest earnings found on google playstore app are:-

1. Minecraft
2. Poweramp Full Version Unlocker
3. Officesuite Pro+ Pdf

WordCloud

```
from wordcloud import WordCloud
```

```
plt.subplots(figsize=(20,9))
wordcloud = WordCloud(
                          background_color='black',
                              width=1080,
                              height=720
                             ).generate(" ".join(df.Category))
plt.imshow(wordcloud)
plt.axis('off')
plt.savefig('graph.png')
plt.show()
```



The above word cloud has been generated using Google-Playstore.csv file in the dataset.

Advantage of wordcloud is identifying new SEO keywords to target and also analyzing customer and employee feedback.

Summary & Conclusion

After Analyzing the dataset I have got answers to some of the serious & interesting question which any of the android users would love to know:-

1. What is highest rated category?
2. What year has the most number of apps released?
3. Does the average apps size change over the years and which app have maximum size?
4. What is the percentage proportion between free and paid apps?
5. What are the top 10 categories with most rating how many apps were free and paid in this category?
6. Which category are maximum and minimum average price?
7. What are the top 5 apps on the maximum number of installs?

8. Which app have maximum number of installs in productivity category and also list the top 5 apps name.
9. What is the most month, day and weekday apps get updated?
10. How many apps can work on android version 4?
11. Which are the apps that have made the highest earning in dataset?

We went through a lot of ways to visualize the data collected from this enormous dataset. This analysis was mostly focused on the monetization aspect of the apps, but a lot of other aspects were also covered such as the rating, content rating, the evolution of the number of new apps release etc. In the end, we showed how there is an exponentionnal increase of the number of apps as the time goes on. We also saw that the apps which seem to generate the most revenue. the pricing for paid apps depending on the content rating, the evoltion of categories over time and so on. The amount of inferences we can get from this dataset is humongous, due to how rich it is.

Reference

[1] Python offical documentation. https://docs.python.org/3/

[2] Requests library. https://pypi.org/project/requests/

[3] Aakash N S, Exploratory Data Analysis. https://jovian.ai/learn/zero-to-data-analyst-bootcamp/assignment/exploratory-data-analysis-project

[4] Pandas library documentation. https://pandas.pydata.org/docs/

[5] Working with Google colab. https://machinelearningmastery.com/google-colab-for-machine-learning-projects/

[6] Kaggle. https://www.kaggle.com/datasets/gauthamp10/google-playstore-apps

[7] Visualization. https://www.analyticsvidhya.com/blog/2021/02/an-intuitive-guide-to-visualization-in-python/

Future Work

Exploring the correlation between the size of the app and the version of Android on the number of installs and also exploring reviews and sentiment of the users as per the the category of the application.

After this analysis I want to take it one step up by deploying a Web App for answering to different questions of many users all around the world, which may help different app developers for making certain decisions before starting their work.