Saiprateek1 / Google-Play-Store-EDA-Project Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main

Google-Play-Store-EDA-Project / Sai_Prateek___EDA_Play_Store_App_Review_Analysis_Capstone_Project (1).ipynb

Saiprateek1 Add files via upload

1 contributor

1.6 MB

Play Store App Review Analysis

Project Type - EDA

Contribution - Individual

Name - Baki Venkata Sai Prateek

Project Summary

The Google Play Store is a digital distribution service operated and developed by Google. It serves as the official app store for the Android operating system, allowing users to browse and download applications. The Play Store offers a wide range of apps, including games, music, movies, and television shows, as well as books and magazines.

One of the primary revenue models for the Play Store is through the sale of app downloads and in-app purchases. When a user purchases an app or makes an in-app purchase, the developer of the app receives a percentage of the sale. The exact percentage varies depending on the specific terms of the sale, but it is typically around 70% for the developer and 30% for Google.

In addition to the sale of app downloads and in-app purchases, the Play Store also generates revenue through the sale of digital content such as music, movies, and television shows. These sales are typically handled through partnerships with media companies, and the revenue is split between Google and the media company.

Another way that the Play Store generates revenue is through advertising. Google AdMob is a mobile advertising platform that allows developers to monetize their apps through in-app ads. Advertisers bid on ad placements in apps, and the highest bidder gets their ad shown to users. The revenue from these ads is split between Google and the developer of the app.

One of the big challenges facing the Play Store is the issue of fake or malicious apps. There have been numerous instances of apps being uploaded to the Play Store that contain malware or are designed to trick users into making purchases or providing personal information. To combat this issue, Google has implemented various measures to try to detect and remove fake or malicious apps, but it remains a persistent problem.

Another challenge for the Play Store is the intense competition from other app stores. The App Store, operated by Apple, is a major competitor to the Play Store, and there are also a number of other app stores that offer alternative sources for Android apps. This competition can make it difficult for developers to get their apps noticed and can also drive down the prices that developers are able to charge for their apps.

Finally, the Play Store faces challenges related to the overall growth of the mobile app market. As more and more people use smartphones and tablets, the demand for apps is increasing, but the supply of app developers is not necessarily keeping pace. This can lead to a saturation of the market, which can make it difficult for developers to stand out and generate significant revenue from their apps.

Overall, the Google Play Store is a major player in the mobile app market, but it faces a number of challenges as it continues to grow and evolve. From the issue of fake or malicious apps to intense competition from other app stores, the Play Store has to work hard to maintain its position as a leading source for mobile apps.

GitHub Link -

https://github.com/Saiprateek1/Google-Play-Store-EDA-Project.git

Problem Statement

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.

Dataset Exploration

- Basic Understanding of play store dataset and potential columns for data cleaning

In [3]:
```
#Imported python libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:
```
# Drive mount for access the files
from google.colab import drive
drive.mount('/content/drive')
```

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

In [5]:
```
#File path ,play store data as df_ps and User data as df_ur
file_path = '/content/drive/MyDrive/AlmaBetter/Capstone Projects/'

df_ps = pd.read_csv(file_path + 'Play Store Data.csv')
df_ur = pd.read_csv(file_path + 'User Reviews.csv')
```

In [6]:
```
# Read first 5 rows of play store data
df_ps.head()
```

Out[6]:

App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updatec

0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	January 7, 2018
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	June 8 2018
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20 2018

In [7]:

```
# Read last 5 rows of play store data
df_ps.tail()
```

Out[7]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres
10836	Sya9a Maroc - FR	FAMILY	4.5	38	53M	5,000+	Free	0	Everyone	Education
10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3.6M	100+	Free	0	Everyone	Education
10838	Parkinson Exercices FR	MEDICAL	NaN	3	9.5M	1,000+	Free	0	Everyone	Medical
10839	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5	114	Varies with device	1,000+	Free	0	Mature 17+	Books & Reference
10840	iHoroscope - 2018 Daily Horoscope & Astrology	LIFESTYLE	4.5	398307	19M	10,000,000+	Free	0	Everyone	Lifestyle

In [8]:

```
# Understand shape of data (Rows , Columns)
df_ps.shape
```

Out[8]: (10841, 13)

In [9]:

```
# Statistics summary of play store data
df_ps.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
```

In [10]:

```
# Understand shape of data (Rows , Columns)
df_ur.shape
```

Out[10]:

```
(64295, 5)
```

In [11]:

```
# Statistics summary of User data
df_ur.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
```

In [12]:

```
#The described method will help to see how data has been spread for numerical values.
df_ps.describe(include='all').T
```

Out[12]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
App	10841	9660	ROBLOX	9	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Category	10841	34	FAMILY	1972	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Rating	9367.0	NaN	NaN	NaN	4.193338	0.537431	1.0	4.0	4.3	4.5	19.0
Reviews	10841	6002	0	596	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Size	10841	462	Varies with device	1695	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Installs	10841	22	1,000,000+	1579	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Type	10840	3	Free	10039	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Price	10841	93	0	10040	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Content Rating	10840	6	Everyone	8714	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Genres	10841	120	Tools	842	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Last Updated	10841	1378	August 3, 2018	326	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Current Ver	10833	2832	Varies with device	1459	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Android Ver	10838	33	4.1 and up	2451	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Data Cleaning

- Knowing how to clean your data is

advantageous for many reasons. Here are just a few:

1. It prevents you from wasting time on

wobbly or even faulty analysis

2. It prevents you from making the wrong

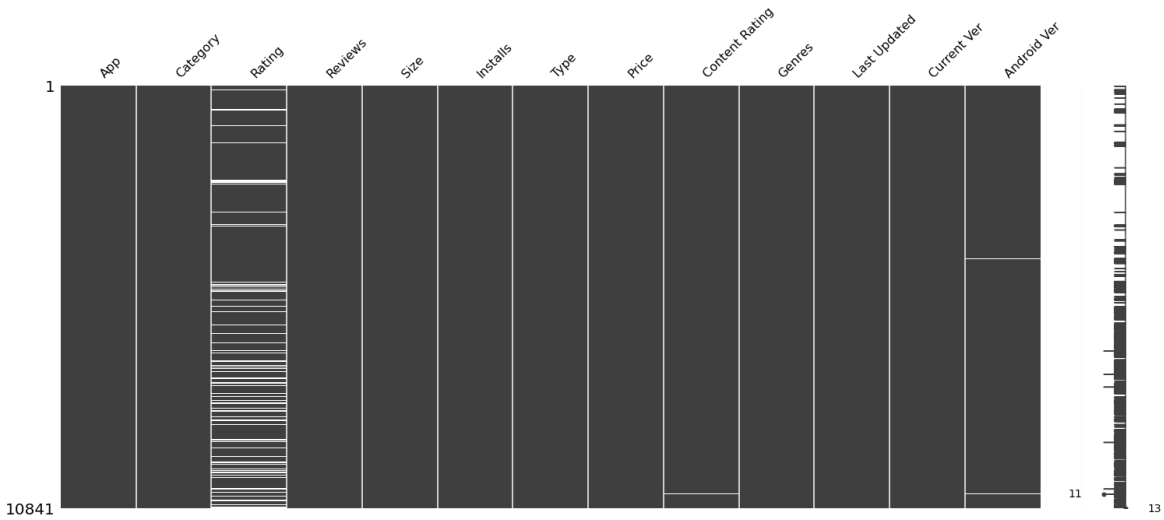
conclusions, which would make you look bad!

In [13]:
```
#Instaaled missingno libraries for better visualizing missing data
! pip install missingno
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: missingno in /usr/local/lib/python3.8/dist-packages (0.5.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-packages (from missingno) (1.
7.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (from missingno) (1.
21.6)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-packages (from missingn
o) (3.2.2)
Requirement already satisfied: seaborn in /usr/local/lib/python3.8/dist-packages (from missingno)
(0.11.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.
8/dist-packages (from matplotlib->missingno) (3.0.9)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.8/dist-packages (from
matplotlib->missingno) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplot
lib->missingno) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from ma
tplotlib->missingno) (1.4.4)
Requirement already satisfied: pandas>=0.23 in /usr/local/lib/python3.8/dist-packages (from seaborn
->missingno) (1.3.5)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dist-packages (from pandas>
=0.23->seaborn->missingno) (2022.7)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-date
util>=2.1->matplotlib->missingno) (1.15.0)
```

In [14]:
```
import missingno as msno
msno.matrix(df_ps)
```

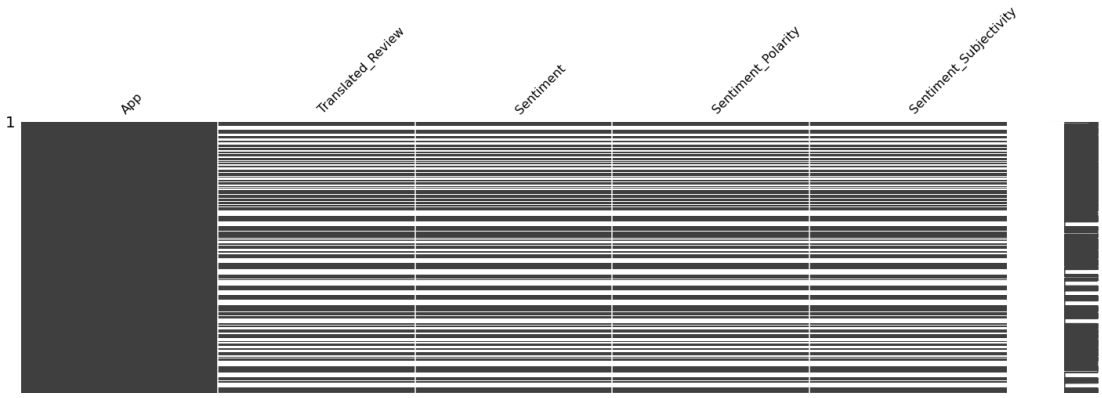
Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff6472728e0>
```

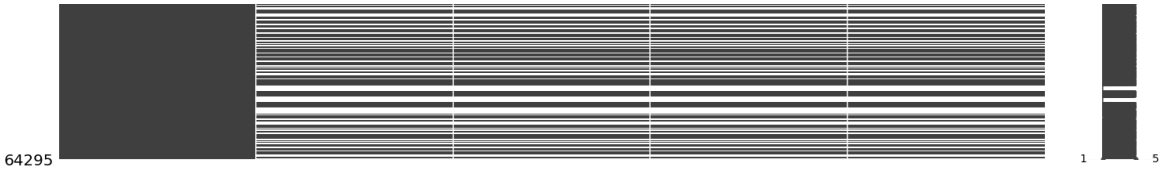


- Rating Coulumn having missing data, Follwed by content rating , Android Ver

In [15]:
```
msno.matrix(df_ur)
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff6449bcc70>
```





In [16]:
```
# Using Fill method and replaced all null data
df_ps['Rating'] = df_ps[['Rating']].fillna(value=df_ps['Rating'].mean())
df_ps['Rating']= round(df_ps[['Rating']],1)
```

In [17]:
```
#Unique method for better understanding the column data
df_ps['Rating'].unique()
```

Out[17]:
```
array([ 4.1,  3.9,  4.7,  4.5,  4.3,  4.4,  3.8,  4.2,  4.6,  3.2,  4. ,
        4.8,  4.9,  3.6,  3.7,  3.3,  3.4,  3.5,  3.1,  5. ,  2.6,  3. ,
        1.9,  2.5,  2.8,  2.7,  1. ,  2.9,  2.3,  2.2,  1.7,  2. ,  1.8,
        2.4,  1.6,  2.1,  1.4,  1.5,  1.2, 19. ])
```

In [18]:
```
# Filter dataframe rows or columns according to the specified index labels
df_ps[df_ps['Rating'] == 19.]
```

Out[18]:

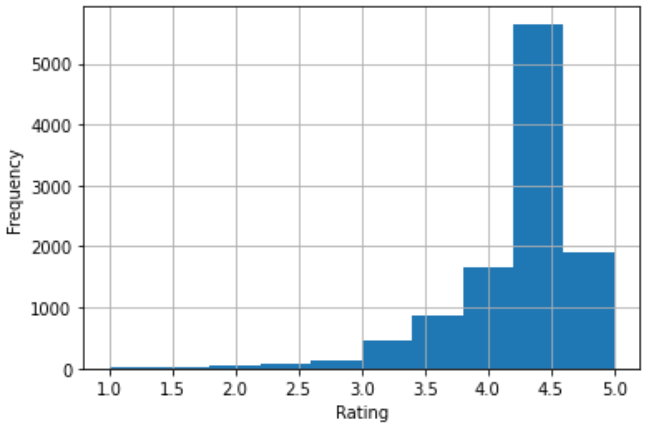
	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Curre Ve
10472	Life Made WI-Fi Touchscreen Photo Frame	1.9	19.0	3.0M	1,000+	Free	0	Everyone	NaN	February 11, 2018	1.0.19	4.0 ar u

- 19.0 is a outlier and replaced with mean value

In [19]:
```
df_ps['Rating'] = df_ps['Rating'].replace(19.,value =df_ps['Rating'].mean())
```

In [20]:
```
# visualizing Rating column values with help of histogram
df_ps.Rating.hist();

plt.xlabel('Rating')
plt.ylabel('Frequency')
```

Out[20]:
```
Text(0, 0.5, 'Frequency')
```



In [21]:
```
# With fill method replaced all null values
df_ps[['Type','Content Rating','Current Ver','Android Ver']] = df_ps[['Type','Content Rating','Cur
df_ps.isnull().sum()
```

Out[21]:
```
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
```

In [22]:

```
# Datatype conversion to float
df_ps.Installs = df_ps.Installs.str.replace(',','')
df_ps.Installs = df_ps.Installs.str.replace('+','')
df_ps.Installs = df_ps.Installs.str.replace('Free','0')

df_ps['Installs'] = pd.to_numeric(df_ps['Installs'])
```

```
<ipython-input-22-c20e01ccda59>:3: FutureWarning: The default value of regex will change from True
to False in a future version. In addition, single character regular expressions will *not* be treat
ed as literal strings when regex=True.
  df_ps.Installs = df_ps.Installs.str.replace('+','')
```

- By using replace , Replace '+' with ''.For calculation purpose

In [23]:

```
df_ps['Installs']
```

Out[23]:

```
0           10000
1          500000
2         5000000
3        50000000
4          100000
           ...
10836        5000
10837         100
10838        1000
10839        1000
10840    10000000
Name: Installs, Length: 10841, dtype: int64
```

In [24]:

```
# Datatype conversion to float
df_ps.Reviews = pd.to_numeric(df_ps.Reviews, errors='coerce')
```

In [25]:

```
df_ps['Reviews'].max()
```

Out[25]:

```
78158306.0
```

In [26]:

```
# Datatype conversion to float
df_ps['Size']=df_ps['Size'].str.replace('M','e+6').str.replace('k','e+3').str.replace('Varies with
```

```
<ipython-input-26-6818ea92e0fb>:2: FutureWarning: The default value of regex will change from True
to False in a future version. In addition, single character regular expressions will *not* be treat
ed as literal strings when regex=True.
  df_ps['Size']=df_ps['Size'].str.replace('M','e+6').str.replace('k','e+3').str.replace('Varies wit
h device','0').str.replace('+','').str.replace('1,000','1000').astype('float')
```

- Replaced 'M'(MB) with e+6(000000)
- Replaced 'K'(KB) with e+3(000)
- Replaced 'Varies with device with 0
- Converted datatype to float

In [27]:

```
df_ps['Size']
```

Out[27]:

```
0        19000000.0
1        14000000.0
2         8700000.0
3        25000000.0
4         2800000.0
             ...
```

```
10836    53000000.0
10837     3600000.0
10838     9500000.0
10839           0.0
10840    19000000.0
Name: Size, Length: 10841, dtype: float64
```

In [28]:
```
# Using replace method
df_ps['Size'].replace(0.0,method = 'bfill',inplace = True)
```

In [29]:
```
df_ps['Size'].min()
```

Out[29]:
```
1000.0
```

In [30]:
```
#Return a Series containing counts of unique values.
df_ps['Type'].value_counts()
```

Out[30]:
```
Free    10040
Paid      800
0           1
Name: Type, dtype: int64
```

In [31]:
```
#Unique method for better understanding the column data
df_ps['Type'].unique()
```

Out[31]:
```
array(['Free', 'Paid', '0'], dtype=object)
```

In [32]:
```
# Using replace method
df_ps['Type'] = df_ps['Type'].replace('0','Free')
```

In [33]:
```
df_ps['Type'].unique()
```

Out[33]:
```
array(['Free', 'Paid'], dtype=object)
```

In [34]:
```
df_ps['Type'].value_counts()
```

Out[34]:
```
Free    10041
Paid      800
Name: Type, dtype: int64
```

In [35]:
```
df_ps['Price'].unique()
```

Out[35]:
```
array(['0', '$4.99', '$3.99', '$6.99', '$1.49', '$2.99', '$7.99', '$5.99',
       '$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',
       '$1.00', '$29.99', '$12.99', '$2.49', '$10.99', '$1.50', '$19.99',
       '$15.99', '$33.99', '$74.99', '$39.99', '$3.95', '$4.49', '$1.70',
       '$8.99', '$2.00', '$3.88', '$25.99', '$399.99', '$17.99',
       '$400.00', '$3.02', '$1.76', '$4.84', '$4.77', '$1.61', '$2.50',
       '$1.59', '$6.49', '$1.29', '$5.00', '$13.99', '$299.99', '$379.99',
       '$37.99', '$18.99', '$389.99', '$19.90', '$8.49', '$1.75',
       '$14.00', '$4.85', '$46.99', '$109.99', '$154.99', '$3.08',
       '$2.59', '$4.80', '$1.96', '$19.40', '$3.90', '$4.59', '$15.46',
       '$3.04', '$4.29', '$2.60', '$3.28', '$4.60', '$28.99', '$2.95',
       '$2.90', '$1.97', '$200.00', '$89.99', '$2.56', '$30.99', '$3.61',
       '$394.99', '$1.26', 'Everyone', '$1.20', '$1.04'], dtype=object)
```

In [36]:
```
# Datatype conversion to float
df_ps['Price'] = df_ps.Price.str.replace('$','').str.replace('Everyone','0').astype(float)
df_ps['Price'].dtype
```

```
<ipython-input-36-a31a49b07455>:2: FutureWarning: The default value of regex will change from True
to False in a future version. In addition, single character regular expressions will *not* be treat
ed as literal strings when regex=True.
  df_ps['Price'] = df_ps.Price.str.replace('$','').str.replace('Everyone','0').astype(float)
```

Out[36]:
```
dtype('float64')
```

In [37]:
```
df_ps['Price'].max()
```

Out[37]:
```
400.0
```

In [38]:
```
df_ps['Content Rating'].unique()
```

Out[38]:
```
array(['Everyone', 'Teen', 'Everyone 10+', 'Mature 17+',
       'Adults only 18+', 'Unrated'], dtype=object)
```

In [39]:
```
df_ps['Content Rating'].value_counts()
```

Out[39]:
```
Everyone           8715
Teen               1208
Mature 17+          499
Everyone 10+        414
Adults only 18+       3
Unrated               2
Name: Content Rating, dtype: int64
```

In [40]:
```
df_ps['Content Rating']= df_ps['Content Rating'].replace(4.1,'Teen')
```

In [41]:
```
df_ps['Genres'].unique()
```

Out[41]:
```
array(['Art & Design', 'Art & Design;Pretend Play',
       'Art & Design;Creativity', 'Art & Design;Action & Adventure',
       'Auto & Vehicles', 'Beauty', 'Books & Reference', 'Business',
       'Comics', 'Comics;Creativity', 'Communication', 'Dating',
       'Education;Education', 'Education', 'Education;Creativity',
       'Education;Music & Video', 'Education;Action & Adventure',
       'Education;Pretend Play', 'Education;Brain Games', 'Entertainment',
       'Entertainment;Music & Video', 'Entertainment;Brain Games',
       'Entertainment;Creativity', 'Events', 'Finance', 'Food & Drink',
       'Health & Fitness', 'House & Home', 'Libraries & Demo',
       'Lifestyle', 'Lifestyle;Pretend Play',
       'Adventure;Action & Adventure', 'Arcade', 'Casual', 'Card',
       'Casual;Pretend Play', 'Action', 'Strategy', 'Puzzle', 'Sports',
       'Music', 'Word', 'Racing', 'Casual;Creativity',
       'Casual;Action & Adventure', 'Simulation', 'Adventure', 'Board',
       'Trivia', 'Role Playing', 'Simulation;Education',
       'Action;Action & Adventure', 'Casual;Brain Games',
       'Simulation;Action & Adventure', 'Educational;Creativity',
       'Puzzle;Brain Games', 'Educational;Education', 'Card;Brain Games',
       'Educational;Brain Games', 'Educational;Pretend Play',
       'Entertainment;Education', 'Casual;Education',
       'Music;Music & Video', 'Racing;Action & Adventure',
       'Arcade;Pretend Play', 'Role Playing;Action & Adventure',
       'Simulation;Pretend Play', 'Puzzle;Creativity',
       'Sports;Action & Adventure', 'Educational;Action & Adventure',
       'Arcade;Action & Adventure', 'Entertainment;Action & Adventure',
       'Puzzle;Action & Adventure', 'Strategy;Action & Adventure',
       'Music & Audio;Music & Video', 'Health & Fitness;Education',
       'Adventure;Education', 'Board;Brain Games',
       'Board;Action & Adventure', 'Board;Pretend Play',
       'Casual;Music & Video', 'Role Playing;Pretend Play',
       'Entertainment;Pretend Play', 'Video Players & Editors;Creativity',
       'Card;Action & Adventure', 'Medical', 'Social', 'Shopping',
       'Photography', 'Travel & Local',
       'Travel & Local;Action & Adventure', 'Tools', 'Tools;Education',
       'Personalization', 'Productivity', 'Parenting',
       'Parenting;Music & Video', 'Parenting;Education',
       'Parenting;Brain Games', 'Weather', 'Video Players & Editors',
       'Video Players & Editors;Music & Video', 'News & Magazines',
       'Maps & Navigation', 'Health & Fitness;Action & Adventure',
       'Educational', 'Casino', 'Adventure;Brain Games',
       'Trivia;Education', 'Lifestyle;Education',
       'Books & Reference;Creativity', 'Books & Reference;Education',
       'Puzzle;Education', 'Role Playing;Education',
       'Role Playing;Brain Games', 'Strategy;Education',
       'Racing;Pretend Play', 'Communication;Creativity',
       'February 11, 2018', 'Strategy;Creativity'], dtype=object)
```

In [42]:
```
# Datatype conversion to datetime
from datetime import datetime,date
```

```
df_ps['Last Updated']=pd.to_datetime(df_ps['Last Updated'],errors='coerce')
df_ps['Last Updated']
```

Out[42]:
```
0       2018-01-07
1       2018-01-15
2       2018-08-01
3       2018-06-08
4       2018-06-20
           ...
10836   2017-07-25
10837   2018-07-06
10838   2017-01-20
10839   2015-01-19
10840   2018-07-25
Name: Last Updated, Length: 10841, dtype: datetime64[ns]
```

In [43]:
```
#Understading the frequency of app updates
frequency = df_ps['Last Updated'].diff()
frequency.value_counts()
```

Out[43]:
```
0 days        155
1 days        102
-3 days        92
3 days         91
-2 days        84
              ...
1160 days       1
-1123 days      1
-740 days       1
-1335 days      1
-1578 days      1
Name: Last Updated, Length: 2404, dtype: int64
```

In [44]:
```
frequency.median().days
```

Out[44]: 0

In [45]:
```
df_ps['Day']=df_ps['Last Updated'].dt.day
df_ps['Month']=df_ps['Last Updated'].dt.month
df_ps['Year']=df_ps['Last Updated'].dt.year
```

In [46]:
```
df_ps['Day']=df_ps['Day'].apply("int64")
df_ps['Month'] =df_ps['Month'].apply("int64")
df_ps['Year']= df_ps['Year'].apply("int64")
```

In [47]:
```
df_ps['Year']=df_ps['Year'].replace(-9223372036854775808,2010)
df_ps['Day']=df_ps['Day'].replace(-9223372036854775808,1)
df_ps['Month']=df_ps['Month'].replace(-9223372036854775808,1)
```

In [48]:
```
df_ps['Year'].unique()
```

Out[48]: array([2018, 2017, 2014, 2016, 2015, 2013, 2012, 2011, 2010])

In [49]:
```
df_ps.head()
```

Out[49]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	L Upda
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159.0	19000000.0	10000	Free	0.0	Everyone	Art & Design	2018-
1	Coloring book moana	ART_AND_DESIGN	3.9	967.0	14000000.0	500000	Free	0.0	Everyone	Art & Design;Pretend Play	2018-

2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510.0	8700000.0	5000000	Free	0.0	Everyone	Art & Design	2018·
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644.0	25000000.0	50000000	Free	0.0	Teen	Art & Design	2018·
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967.0	2800000.0	100000	Free	0.0	Everyone	Art & Design;Creativity	2018·

In [50]:
```
df_ps['Current Ver'].value_counts()
```

Out[50]:
```
Varies with device    1459
1.0                    809
1.1                    264
1.2                    178
2.0                    151
                      ...
1.0.17.3905              1
15.1.2                   1
4.94.19                  1
1.1.11.11                1
2.0.148.0                1
Name: Current Ver, Length: 2832, dtype: int64
```

In [51]:
```
df_ps['Android Ver'].value_counts()
```

Out[51]:
```
4.1 and up            2452
4.0.3 and up          1501
4.0 and up            1376
Varies with device    1362
4.4 and up             981
2.3 and up             652
5.0 and up             601
4.2 and up             394
2.3.3 and up           281
2.2 and up             244
4.3 and up             243
3.0 and up             241
2.1 and up             134
1.6 and up             116
6.0 and up              60
7.0 and up              42
3.2 and up              36
2.0 and up              32
5.1 and up              24
1.5 and up              20
4.4W and up             12
3.1 and up              10
2.0.1 and up             7
8.0 and up               6
7.1 and up               3
4.0.3 - 7.1.1            2
5.0 - 8.0                2
1.0 and up               2
7.0 - 7.1.1              1
4.1 - 7.1.1              1
5.0 - 6.0                1
2.2 - 7.1.1              1
5.0 - 7.1.1              1
Name: Android Ver, dtype: int64
```

In [52]:
```
df_ps['Category'].unique()
```

Out[52]:
```
array(['ART_AND_DESIGN', 'AUTO_AND_VEHICLES', 'BEAUTY',
       'BOOKS_AND_REFERENCE', 'BUSINESS', 'COMICS', 'COMMUNICATION',
       'DATING', 'EDUCATION', 'ENTERTAINMENT', 'EVENTS', 'FINANCE',
```

```
       'FOOD_AND_DRINK', 'HEALTH_AND_FITNESS', 'HOUSE_AND_HOME',
       'LIBRARIES_AND_DEMO', 'LIFESTYLE', 'GAME', 'FAMILY', 'MEDICAL',
       'SOCIAL', 'SHOPPING', 'PHOTOGRAPHY', 'SPORTS', 'TRAVEL_AND_LOCAL',
       'TOOLS', 'PERSONALIZATION', 'PRODUCTIVITY', 'PARENTING', 'WEATHER',
       'VIDEO_PLAYERS', 'NEWS_AND_MAGAZINES', 'MAPS_AND_NAVIGATION',
       '1.9'], dtype=object)
```

In [53]:

```
# Using Replace method
df_ps['Category'].replace(4.1,method = 'ffill',inplace = True)
```

In [54]:

```
df_ps['Category'].unique()
```

Out[54]:

```
array(['ART_AND_DESIGN', 'AUTO_AND_VEHICLES', 'BEAUTY',
       'BOOKS_AND_REFERENCE', 'BUSINESS', 'COMICS', 'COMMUNICATION',
       'DATING', 'EDUCATION', 'ENTERTAINMENT', 'EVENTS', 'FINANCE',
       'FOOD_AND_DRINK', 'HEALTH_AND_FITNESS', 'HOUSE_AND_HOME',
       'LIBRARIES_AND_DEMO', 'LIFESTYLE', 'GAME', 'FAMILY', 'MEDICAL',
       'SOCIAL', 'SHOPPING', 'PHOTOGRAPHY', 'SPORTS', 'TRAVEL_AND_LOCAL',
       'TOOLS', 'PERSONALIZATION', 'PRODUCTIVITY', 'PARENTING', 'WEATHER',
       'VIDEO_PLAYERS', 'NEWS_AND_MAGAZINES', 'MAPS_AND_NAVIGATION',
       '1.9'], dtype=object)
```

In [55]:

```
# Drop the duplicates
df_ps = df_ps.drop_duplicates(subset=['App'], keep = 'first')
df_ps.shape
```

Out[55]:

```
(9660, 16)
```

Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to analyze the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.

- Which category having High Active Apps ?
- How many free and paid apps are published ?
- Content Rating - Detail Analysis
- What are the Top App Genres ?
- What are the Top 10 Review Apps ?
- What are the Top 10 High Profitable Apps ?
- What are the App pricing trends across top categories ?

In [ ]:

```
# Compute pairwise correlation of columns
df_ps.corr()
```

Out[ ]:

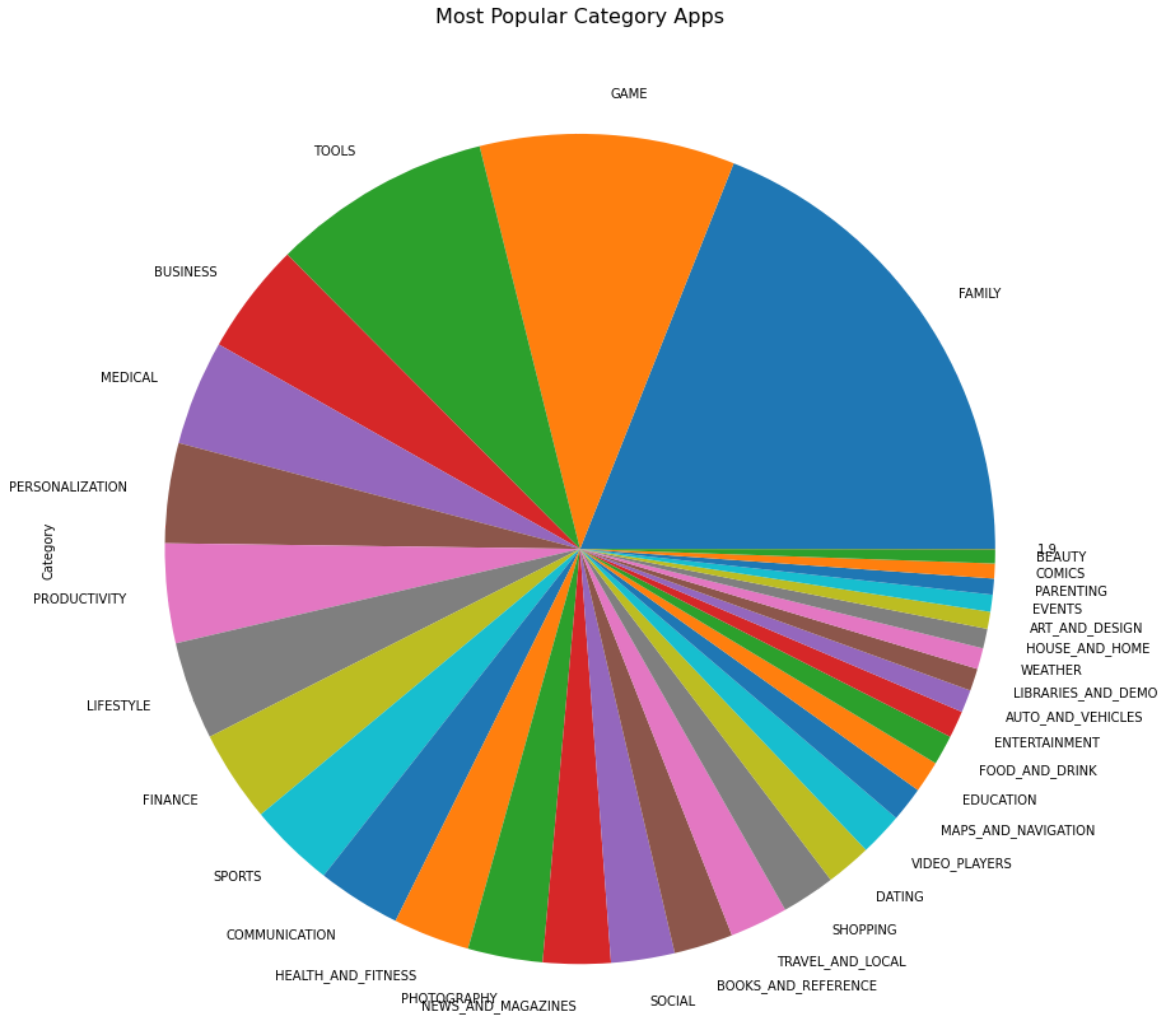
	Rating	Reviews	Size	Installs	Price	Day	Month	Year
Rating	1.000000	0.054032	0.054447	0.038875	-0.019363	-0.000349	-0.000349	-0.000349
Reviews	0.054032	1.000000	0.073463	0.625165	-0.007598	-0.033099	0.036541	0.058073
Size	0.054447	0.073463	1.000000	0.036445	-0.021890	0.009625	0.009625	0.009625
Installs	0.038875	0.625165	0.036445	1.000000	-0.009404	0.001472	0.001472	0.001472
Price	-0.019363	-0.007598	-0.021890	-0.009404	1.000000	0.000664	0.000664	0.000664
Day	-0.000349	-0.033099	0.009625	0.001472	0.000664	1.000000	1.000000	1.000000
Month	-0.000349	0.036541	0.009625	0.001472	0.000664	1.000000	1.000000	1.000000
Year	-0.000349	0.058073	0.009625	0.001472	0.000664	1.000000	1.000000	1.000000

Which category having High Active Apps ?

In [ ]:

```
#Pie Chart For analyzing category app
plt.figure(figsize=(40,15))
```

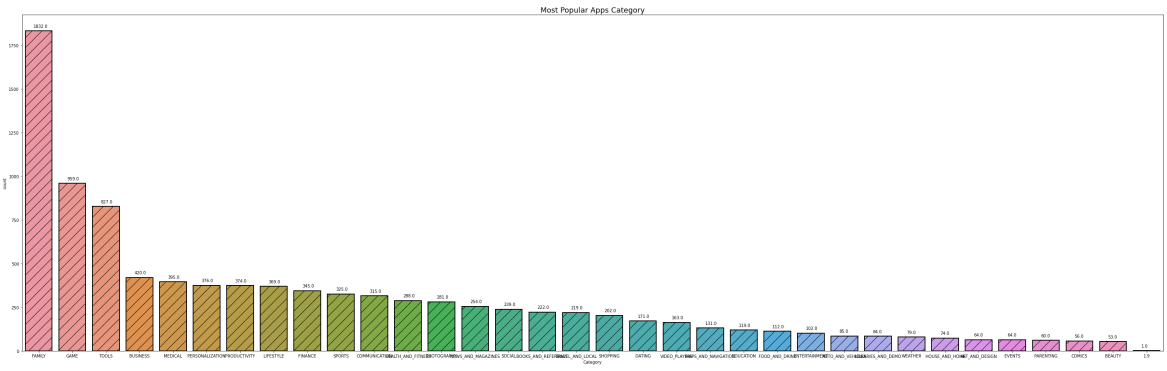
```
df_ps['Category'].value_counts().plot(kind='pie')
plt.title('Most Popular Category Apps',fontsize = 16)
plt.show()
```



In [ ]:

```
#Countplot for popular App Category
plt.figure(figsize=(50,15))
ax = sns.countplot(x = 'Category', data = df_ps, order = df_ps['Category'].value_counts().index,lw
#Data labeling
for p in ax.patches:
   ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+20))

plt.title('Most Popular Apps Category',fontsize = 18)
```

Out[ ]: Text(0.5, 1.0, 'Most Popular Apps Category')



How many free and paid apps are published ??

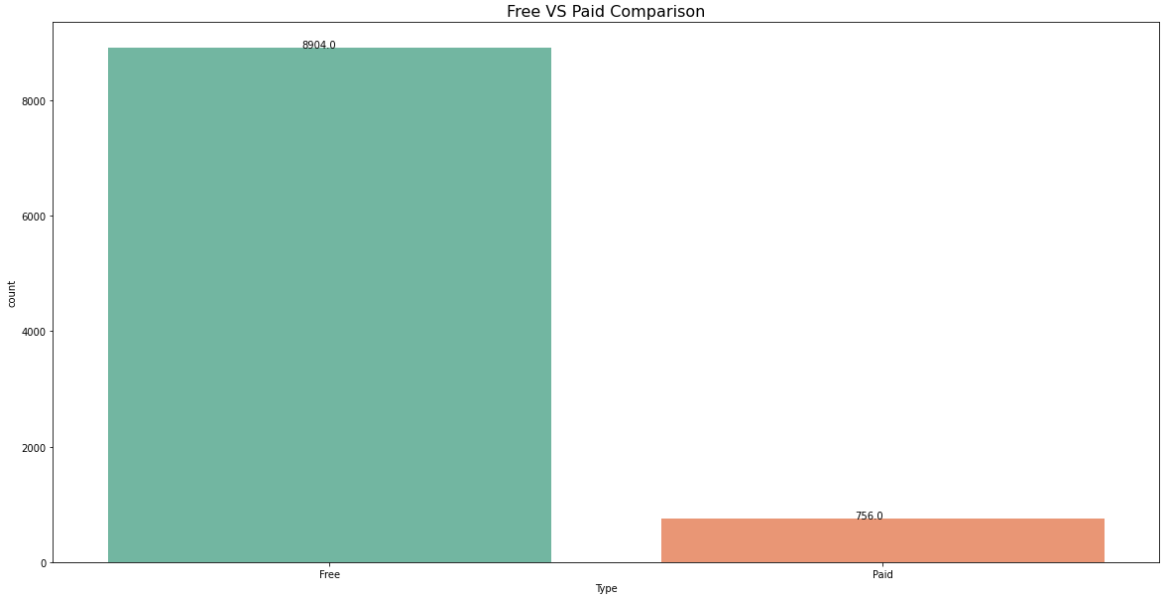
How many free and paid apps are published ?

In [ ]:
```
df_ps['Type'].value_counts(normalize = True)
```

Out[ ]:
```
Free    0.921739
Paid    0.078261
Name: Type, dtype: float64
```

In [56]:
```
#countplot for type of apps
plt.figure(figsize=(20,10))
ax = sns.countplot(x='Type',data=df_ps,palette = "Set2")
#Data labeling
for p in ax.patches:
   ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.35, p.get_height()+1))

plt.title('Free VS Paid Comparison',fontsize = 16)
```

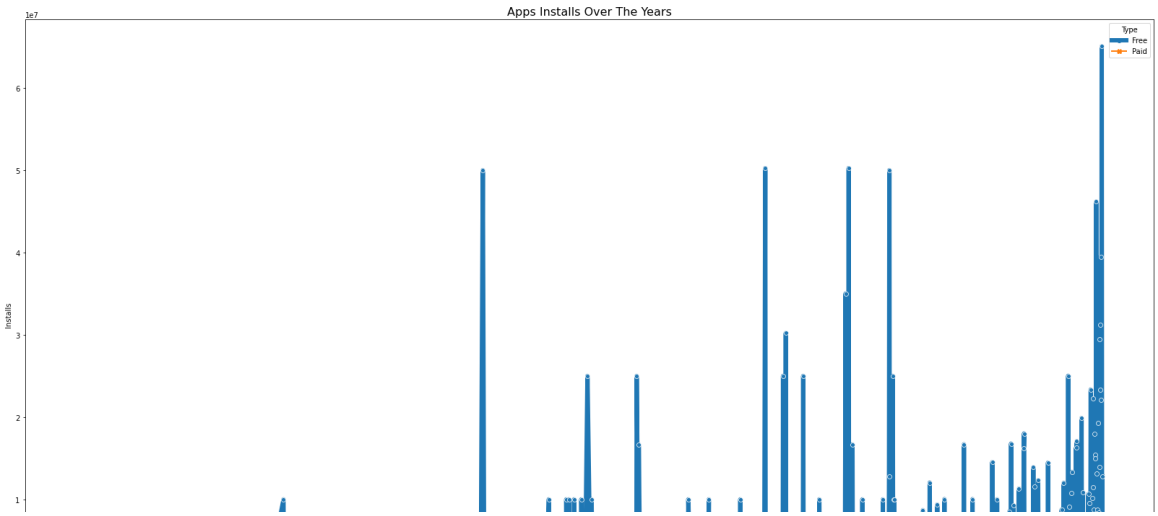
Out[56]:
```
Text(0.5, 1.0, 'Free VS Paid Comparison')
```

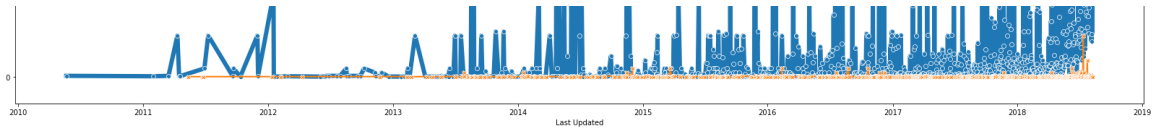


- As per the play store data 92.17 % are free app and only 7.8 % apps are paid.

In [80]:
```
# Lineplot represents numbers of app installs over the past years
plt.figure(figsize=(28,15))
ax = sns.lineplot(x='Last Updated',y='Installs',data=df_ps,hue='Type',style ='Type',ci=False,marke
#Data labeling
for p in ax.patches:
   ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.35, p.get_height()+1))
plt.title('Apps Installs Over The Years',fontsize = 16)
```

Out[80]:
```
Text(0.5, 1.0, 'Apps Installs Over The Years')
```





In [75]:
```
Year = df_ps.groupby(['Year'])['Type']
```

In [76]:
```
Y_17 = Year.get_group(2017).value_counts()
```

In [77]:
```
Y_18 =Year.get_group(2018).value_counts()
```

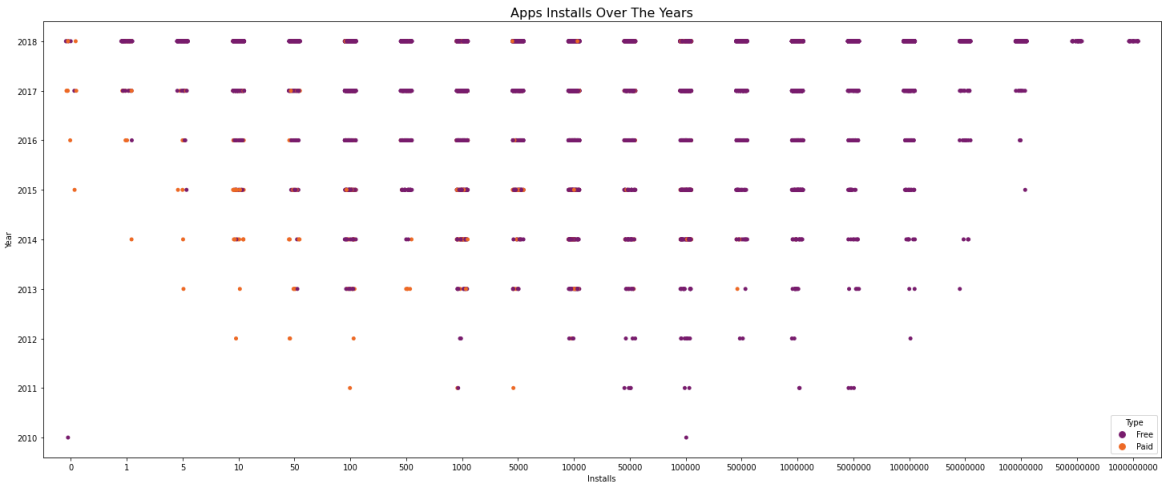
In [78]:
```
YoY_Diff = (Y_18 - Y_17)/Y_17
YoY_Diff
```

Out[78]:
```
Free    2.671596
Paid    0.900585
Name: Type, dtype: float64
```

While analysising last two years of YoY Diffrence in terms of apps published , App store published 2.6 x times more in 2018 compareing to 2017.Its shows they is lot scope for android developers to explored in apps utility and features.

In [79]:
```
# Stripplot represents numbers of app installs over the past years
plt.figure(figsize=(25,10))
sns.stripplot(y='Year',x='Installs',data=df_ps,hue='Type',palette = 'inferno')
plt.title('Apps Installs Over The Years',fontsize = 16)
```

Out[79]:
```
Text(0.5, 1.0, 'Apps Installs Over The Years')
```



In [94]:
```
df_ps.groupby(['Year'])['App'].count()
```

Out[94]:
```
Year
2010       2
2011      15
2012      26
2013     108
2014     203
2015     449
2016     779
2017    1794
2018    6284
Name: App, dtype: int64
```

From above Stripplot, Analyse that In 2010 only 2 numbers of apps was published and apps popularity and useage was gradually increased over the years.In 2018 6284 apps was published.

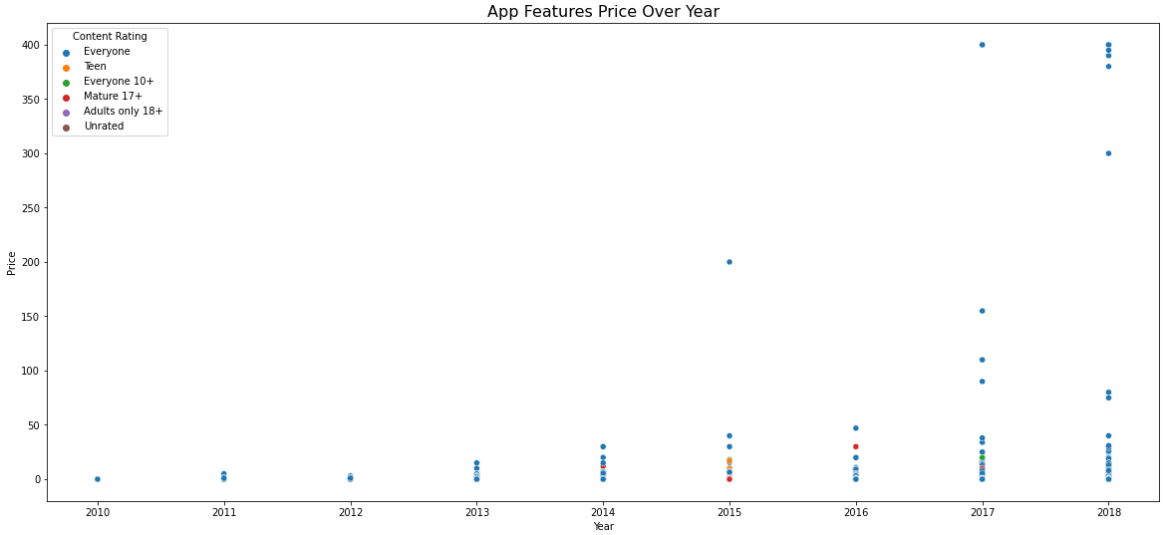
Paid apps are constant growth from past year. But, Apps features plays a major role for paid or premium apps.

Content Rating - Detail Analysis

In [99]:

```
# Scatterplot represents between year and price over content rating
plt.figure(figsize=(20,30))
plt.subplot(3,1,2)
sns.scatterplot(x='Year', y='Price', data=df_ps, hue='Content Rating')
plt.title('App Features Price Over Year',fontsize = 16)
```

Out[99]: Text(0.5, 1.0, 'App Features Price Over Year')



In [102]:

```
df_ps.groupby(['Content Rating'])['Price'].count()
```

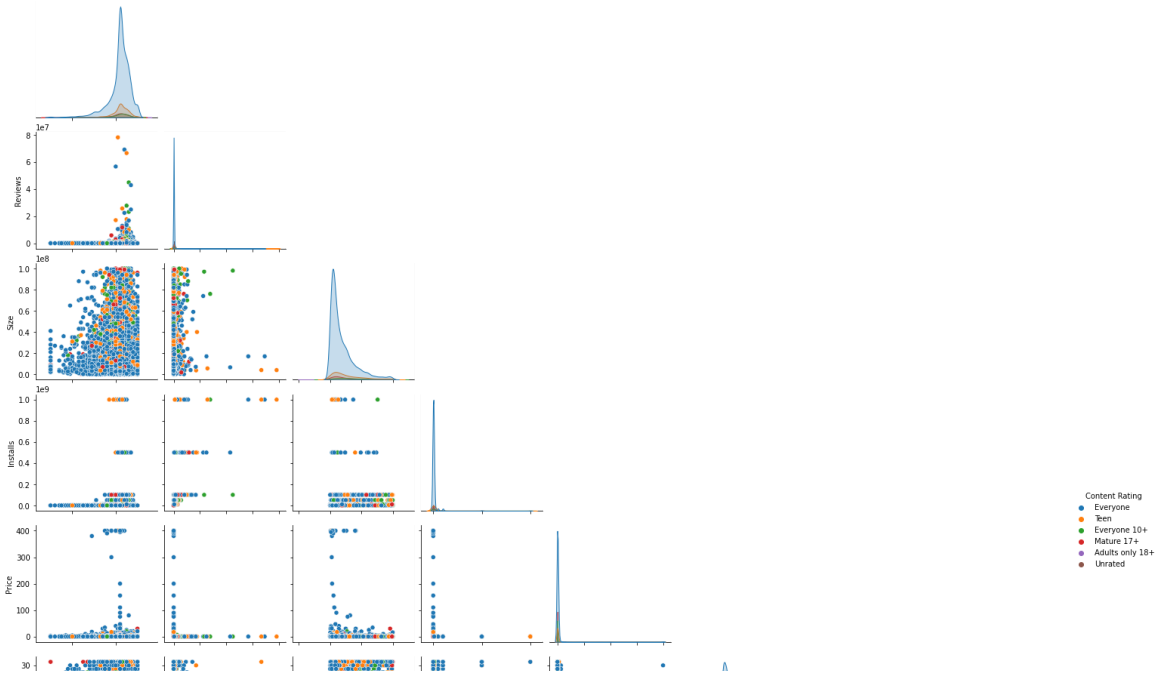
Out[102]:

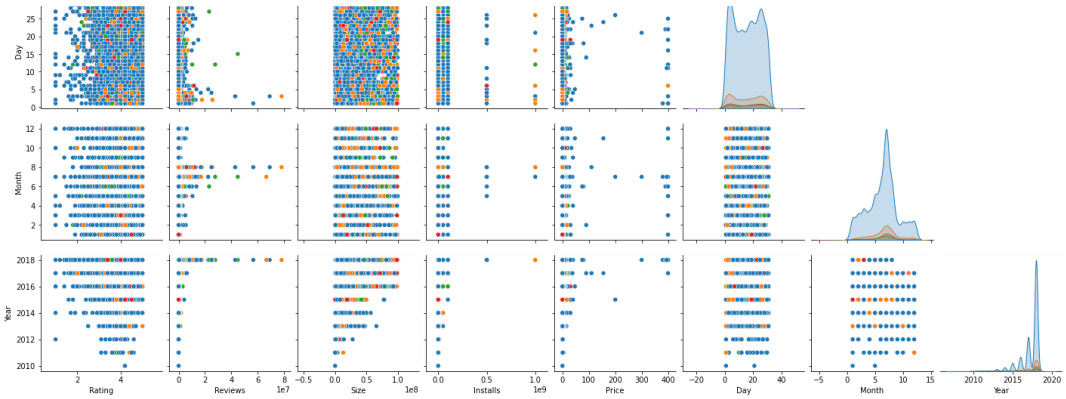
```
Content Rating
Adults only 18+       3
Everyone           7904
Everyone 10+        322
Mature 17+          393
Teen               1036
Unrated               2
Name: Price, dtype: int64
```

Everyone content rating apps having high paid apps and genarating more revenue, Followed by Teen, 10+,17+ content rating apps.

In [101]:

```
# Pairplot represents on numerical columns
sns.pairplot(df_ps, corner=True, hue='Content Rating')
plt.show()
```

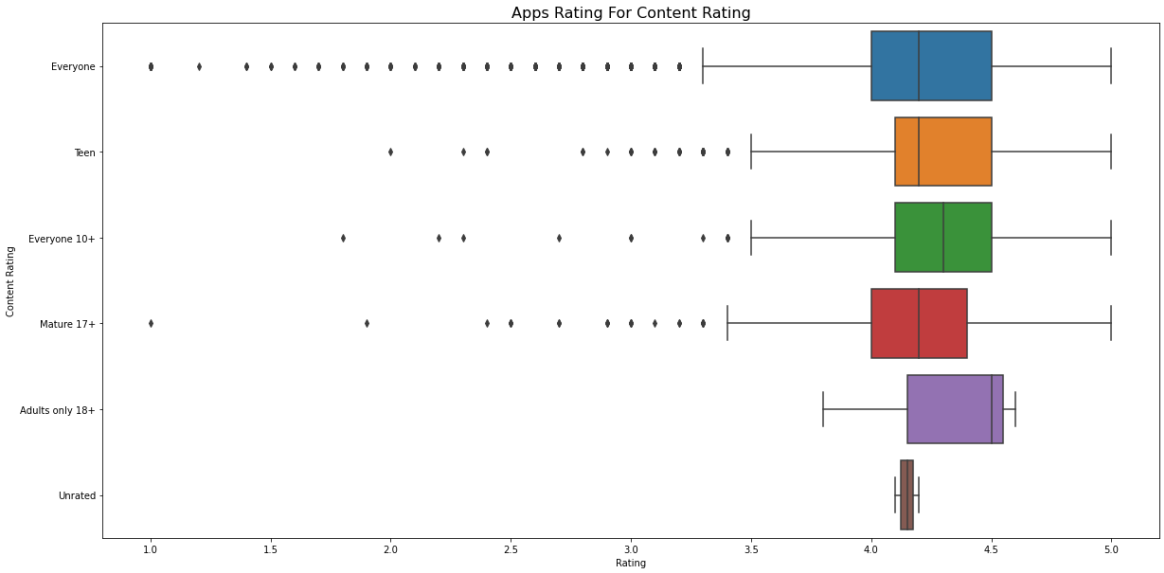




In [100]:

```
#Boxplot represents Content rating VS Rating
plt.figure(figsize=(20,10))
sns.boxplot(x='Rating',y='Content Rating',data = df_ps)
plt.title('Apps Rating For Content Rating',fontsize = 16)
```

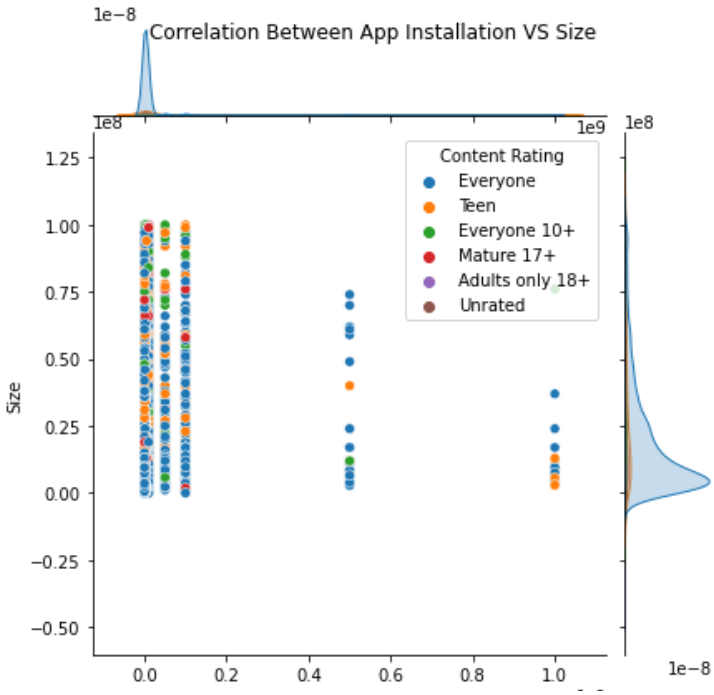
Out[100]: Text(0.5, 1.0, 'Apps Rating For Content Rating')



In [104]:

```
#Jointplot represents App installations VS Size
sns.jointplot(x='Installs',y='Size',data=df_ps,hue='Content Rating')
plt.suptitle('Correlation Between App Installation VS Size')
```

Out[104]: Text(0.5, 0.98, 'Correlation Between App Installation VS Size')



Installs 1e9

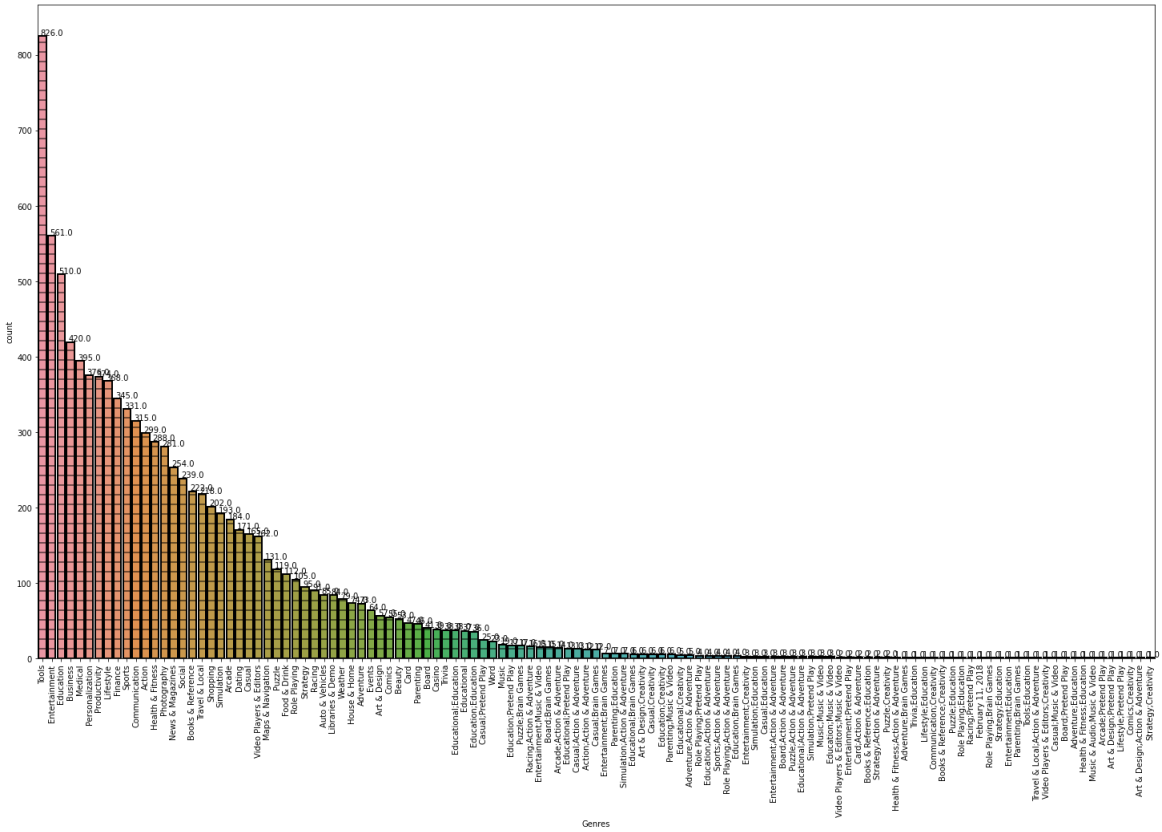
What are the Top App Genres

In [109]:

```
#Countplot represents App Genres
plt.rcParams['figure.figsize'] = (25,15)
x = sns.countplot(x='Genres',data=df_ps,order = df_ps['Genres'].value_counts().index,lw=2,ec='blac

for p in x.patches:
   x.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.15, p.get_height()+0.5))

plt.xticks(rotation=90)
```

Out[109]:

```
(array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118]), <a list of 119 Text major ticklabel objects>)
```



What are the Top 10 Review Apps

In [ ]:

```
#App Reviews sorting
Reviews_Sort = df_ps.sort_values(by=['Reviews'], ascending=False)
Reviews_Sort.head()
```

Out[ ]:

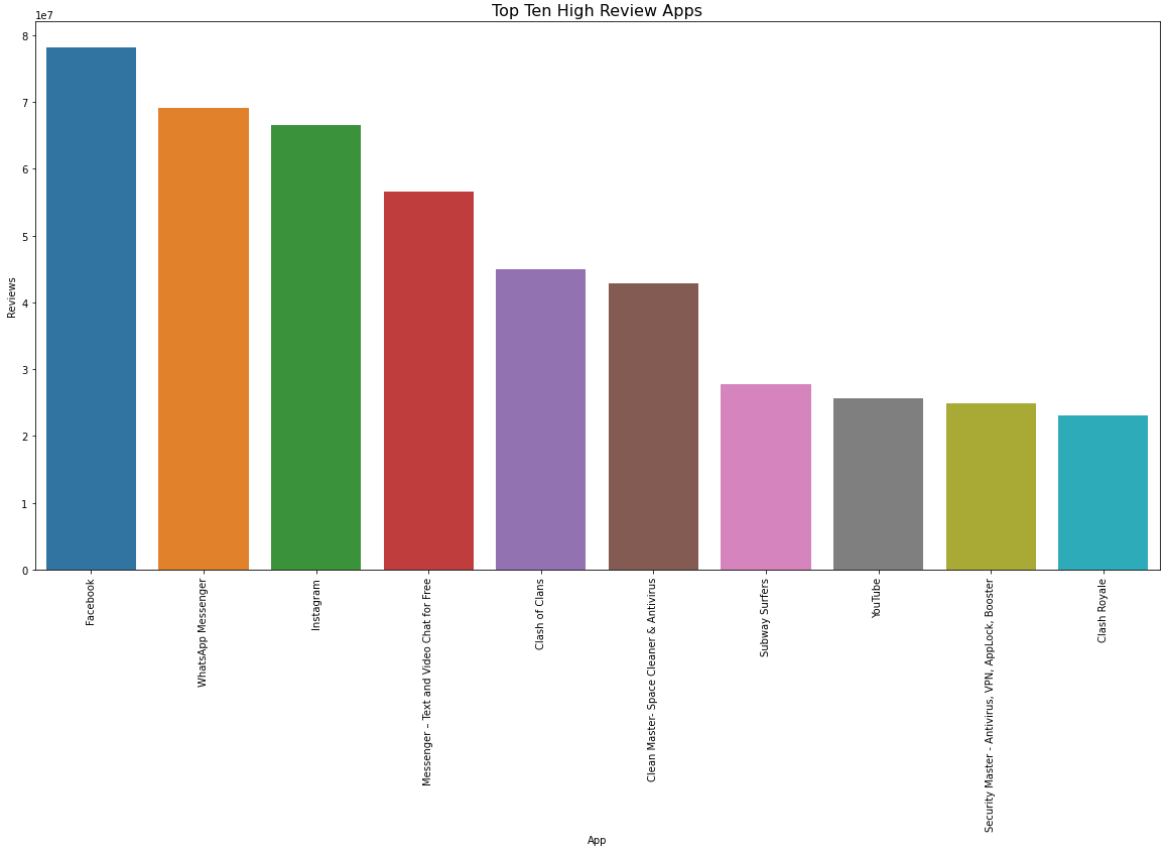
	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genre
2544	Facebook	SOCIAL	4.1	78158306.0	4000000.0	1000000000	Free	0.0	Teen	Soci
336	WhatsApp Messenger	COMMUNICATION	4.4	69119316.0	17000000.0	1000000000	Free	0.0	Everyone	Communicatio

2545	Instagram	SOCIAL	4.5	66577313.0	4000000.0	1000000000	Free	0.0	Teen	Soci
335	Messenger – Text and Video Chat for Free	COMMUNICATION	4.0	56642847.0	17000000.0	1000000000	Free	0.0	Everyone	Communicatio
1670	Clash of Clans	GAME	4.6	44891723.0	98000000.0	100000000	Free	0.0	Everyone 10+	Strateg

In [ ]:

```
# Barplot represents Top App reviews
plt.rcParams['figure.figsize'] = (20, 10)
fig = sns.barplot(x=Reviews_Sort['App'][:10], y=Reviews_Sort['Reviews'][:10])
plt.xticks(rotation=90)
plt.title('Top Ten High Review Apps',fontsize = 16)
```

Out[ ]: Text(0.5, 1.0, 'Top Ten High Review Apps')



What are the Top 10 High Profitable Apps

In [ ]:

```
#Price sorting
Price_Sort = df_ps.sort_values(by=['Price'], ascending=False)
Price_Sort .head()
```

Out[ ]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Cur
4367	I'm Rich - Trump Edition	LIFESTYLE	3.6	275.0	7300000.0	10000	Paid	400.00	Everyone	Lifestyle	2018-05-03	
5351	I am rich	LIFESTYLE	3.8	3547.0	1800000.0	100000	Paid	399.99	Everyone	Lifestyle	2018-01-12	
5359	I am rich(premium)	FINANCE	3.5	472.0	965000.0	5000	Paid	399.99	Everyone	Finance	2017-05-01	
5373	I AM RICH PRO PLUS	FINANCE	4.0	36.0	41000000.0	1000	Paid	399.99	Everyone	Finance	2018-06-25	
											2018-03-	

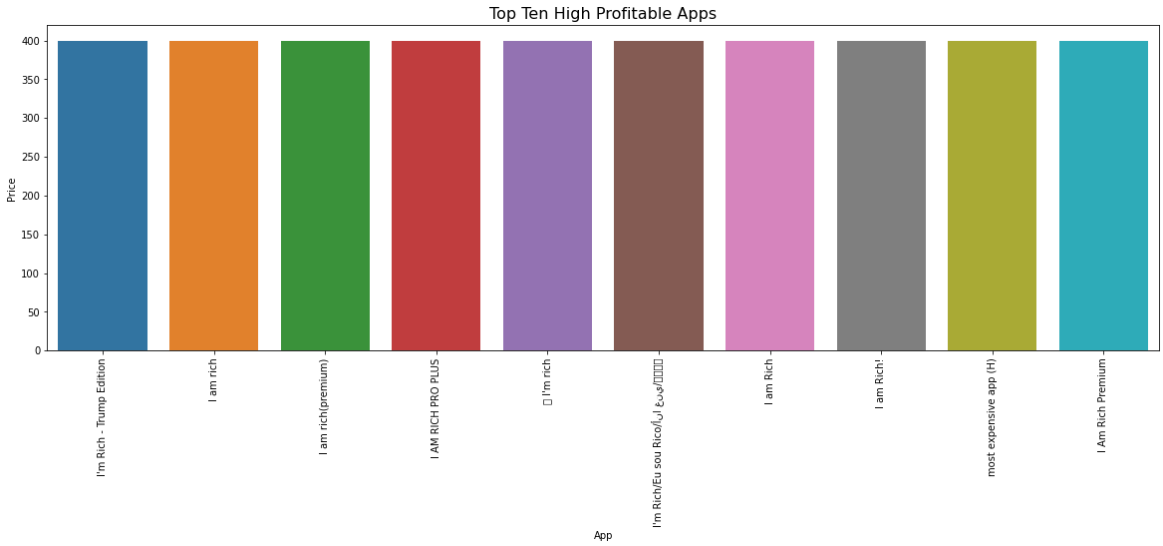
4362	I'm rich	LIFESTYLE	3.8	718.0	26000000.0	10000	Paid	399.99	Everyone	Lifestyle	11

In [ ]:

```
#Barplot represents Top high profitable apps
plt.rcParams['figure.figsize'] = (20, 6)
fig = sns.barplot(x=Price_Sort['App'][:10], y=Price_Sort['Price'][:10])
plt.xticks(rotation=90)
plt.title('Top Ten High Profitable Apps',fontsize = 16)
```

Out[ ]:

```
Text(0.5, 1.0, 'Top Ten High Profitable Apps')

/usr/local/lib/python3.8/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning: Glyp
h 128142 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.8/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning: Glyp
h 25105 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.8/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning: Glyp
h 24456 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.8/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning: Glyp
h 26377 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.8/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning: Glyp
h 37666 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.8/dist-packages/matplotlib/backends/backend_agg.py:183: RuntimeWarning: Glyp
h 128142 missing from current font.
  font.set_text(s, 0, flags=flags)
/usr/local/lib/python3.8/dist-packages/matplotlib/backends/backend_agg.py:183: RuntimeWarning: Glyp
h 25105 missing from current font.
  font.set_text(s, 0, flags=flags)
/usr/local/lib/python3.8/dist-packages/matplotlib/backends/backend_agg.py:183: RuntimeWarning: Glyp
h 24456 missing from current font.
  font.set_text(s, 0, flags=flags)
/usr/local/lib/python3.8/dist-packages/matplotlib/backends/backend_agg.py:183: RuntimeWarning: Glyp
h 26377 missing from current font.
  font.set_text(s, 0, flags=flags)
/usr/local/lib/python3.8/dist-packages/matplotlib/backends/backend_agg.py:183: RuntimeWarning: Glyp
h 37666 missing from current font.
  font.set_text(s, 0, flags=flags)
```



What are the App pricing trends across top categories ??

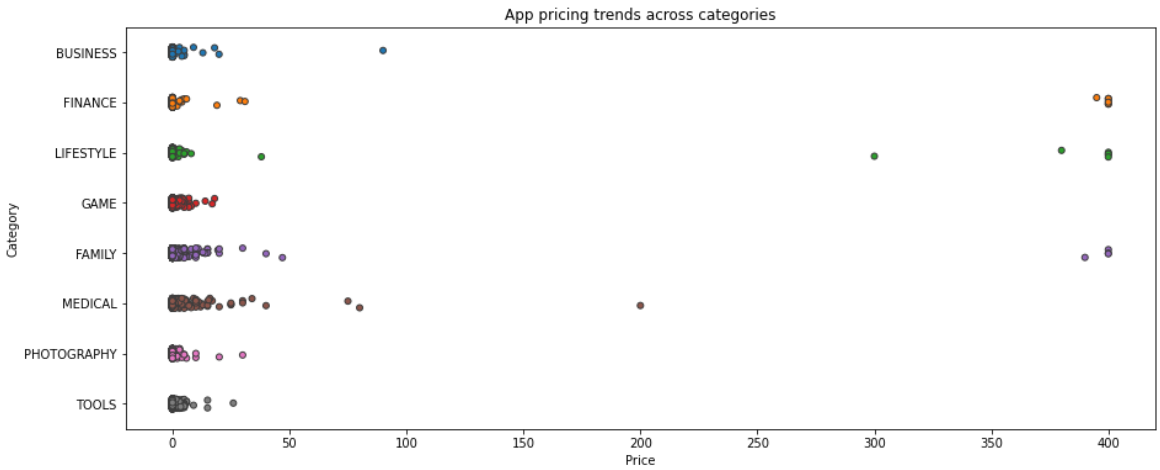
In [ ]:

```
# Top  popular app categories
Top_Category_apps = df_ps[df_ps.Category.isin(['GAME', 'FAMILY', 'PHOTOGRAPHY',
                                                'MEDICAL', 'TOOLS', 'FINANCE',
                                                'LIFESTYLE','BUSINESS'])]

# Plotting Price vs Category
```

```
plt.rcParams['figure.figsize'] = (15, 6)
ax = sns.stripplot(x = Top_Category_apps['Price'], y = Top_Category_apps['Category'], jitter=True,
ax.set_title('App pricing trends across categories')

# Apps whose Price is greater than 200
Above_200 = df_ps[df_ps['Price']>200]
Above_200[['Category', 'App', 'Price']]
```

Out[ ]:

	Category	App	Price
4197	FAMILY	most expensive app (H)	399.99
4362	LIFESTYLE	💎 I'm rich	399.99
4367	LIFESTYLE	I'm Rich - Trump Edition	400.00
5351	LIFESTYLE	I am rich	399.99
5354	FAMILY	I am Rich Plus	399.99
5355	LIFESTYLE	I am rich VIP	299.99
5356	FINANCE	I Am Rich Premium	399.99
5357	LIFESTYLE	I am extremely Rich	379.99
5358	FINANCE	I am Rich!	399.99
5359	FINANCE	I am rich(premium)	399.99
5362	FAMILY	I Am Rich Pro	399.99
5364	FINANCE	I am rich (Most expensive app)	399.99
5366	FAMILY	I Am Rich	389.99
5369	FINANCE	I am Rich	399.99
5373	FINANCE	I AM RICH PRO PLUS	399.99
9917	FINANCE	Eu Sou Rico	394.99
9934	LIFESTYLE	I'm Rich/Eu sou Rico/أنا غني/我很有錢	399.99



After analysing price trends of few popular categorys, price strtegies need to implement and expriment based on app features.

Conclusion

To increase profits and revenue on the Google Play Store, app developers can focus on several key strategies:

1. Optimize app pricing: Developers can experiment with different pricing strategies to find the optimal price point for their app.
2. In-app purchases: Developers can offer in-app purchases such as extra levels or virtual currency to generate additional revenue.
3. Advertising: Developers can earn revenue by integrating ads into their apps and by partnering with brands to offer sponsored content.

4. Subscriptions: Developers can offer in-app subscriptions to provide users with access to premium features or content.
5. App Store Optimization (ASO): Developers can optimize their app's listing on the Google Play Store to increase visibility and downloads.
6. App's features: Developing an app with valuable features that are in high demand, offers a better user experience, and is regularly updated will result in more users and higher revenue.
7. App's Security: Ensuring app's security by implementing features like encryption, authentications, and keeping the app updated with security patches to prevent hacking or data breaches.
8.

User Feedback: Regularly monitoring user feedback and making necessary changes to improve the app's features, user experience and addressing the bugs.

Overall, a combination of these strategies can help developers to increase profits and revenue on the Google Play Store.