GOOGLE APPS DATA PROJECT

In [1]:

```
import numpy as np # Linear algebra
import pandas as pd # data processing, csvfile I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from random import randrange
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
df = pd.read_csv("e:Google Apps Data.csv")
df
```

Out[2]:

	Unnamed: 0.1	Unnamed: 0	App	Category	Rating	Reviews	Size	Installs	Type	Pric
0	0	0	Photo Editor & Candy Camera & Grid & ScrapBook	Art And Design	4.1	159	19.0	10000	Free	0.
1	1	1	Coloring book moana	Art And Design	3.9	967	14.0	500000	Free	0.
2	2	5	U Launcher Lite – FREE Live Cool Themes, Hide ...	Art And Design	4.7	87510	8.7	5000000	Free	0.
3	3	6	Sketch - Draw & Paint	Art And Design	4.5	215644	25.0	50000000	Free	0.
4	4	7	Pixel Draw - Number Art Coloring Book	Art And Design	4.3	967	2.8	100000	Free	0.
...	...	...	...	...	...	...	...	...	...	.
8271	8271	8912	FR Calculator	Family	4.0	7	2.6	500	Free	0.
8272	8272	8913	Sya9a Maroc - FR	Family	4.5	38	53.0	5000	Free	0.
8273	8273	8914	Fr. Mike Schmitz Audio Teachings	Family	5.0	4	3.6	100	Free	0.
8274	8274	8915	The SCP Foundation DB fr nn5n	Books And Reference	4.5	114	1.0	1000	Free	0.
8275	8275	8916	iHoroscope - 2018 Daily Horoscope & Astrology	Lifestyle	4.5	398307	19.0	10000000	Free	0.

8276 rows × 15 columns

In [3]:

```
df.head()
```

Out[3]:

	Unnamed: 0.1	Unnamed: 0	App	Category	Rating	Reviews	Size	Installs	Type	Price	C
0	0	0	Photo Editor & Candy Camera & Grid & ScrapBook	Art And Design	4.1	159	19.0	10000	Free	0.0	
1	1	1	Coloring book moana	Art And Design	3.9	967	14.0	500000	Free	0.0	
2	2	5	U Launcher Lite – FREE Live Cool Themes, Hide ...	Art And Design	4.7	87510	8.7	5000000	Free	0.0	
3	3	6	Sketch - Draw & Paint	Art And Design	4.5	215644	25.0	50000000	Free	0.0	
4	4	7	Pixel Draw - Number Art Coloring Book	Art And Design	4.3	967	2.8	100000	Free	0.0	

In [4]:

```
# Cleaning the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8276 entries, 0 to 8275
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Unnamed: 0.1        8276 non-null   int64  
 1   Unnamed: 0          8276 non-null   int64  
 2   App                 8276 non-null   object 
 3   Category            8276 non-null   object 
 4   Rating              8276 non-null   float64
 5   Reviews             8276 non-null   int64  
 6   Size                8276 non-null   float64
 7   Installs            8276 non-null   int64  
 8   Type                8276 non-null   object 
 9   Price               8276 non-null   float64
 10  Content Rating      7915 non-null   object 
 11  Last Updated        8276 non-null   object 
 12  Current Ver         8276 non-null   object 
 13  Minimum Android Ver 8276 non-null   object 
 14  Genres              8276 non-null   object 
dtypes: float64(3), int64(4), object(8)
memory usage: 970.0+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	Unnamed: 0.1	Unnamed: 0	Rating	Reviews	Size	Installs	
count	8276.000000	8276.000000	8276.000000	8.276000e+03	8276.000000	8.276000e+03	8276.0
mean	4137.500000	4560.609957	4.175121	2.803270e+05	18.897761	9.658206e+06	1.02
std	2389.219747	2560.879748	0.534762	2.096170e+06	22.376521	5.986505e+07	16.77
min	0.000000	0.000000	1.000000	1.000000e+00	0.008300	1.000000e+00	0.0
25%	2068.750000	2459.750000	4.000000	1.290000e+02	2.800000	1.000000e+04	0.0
50%	4137.500000	4613.500000	4.300000	3.213500e+03	9.500000	1.000000e+05	0.0
75%	6206.250000	6765.250000	4.500000	4.627800e+04	27.000000	1.000000e+06	0.0
max	8275.000000	8916.000000	5.000000	7.815831e+07	100.000000	1.000000e+09	400.0

In [6]:
```
df.shape
```

Out[6]:
```
(8276, 15)
```

In [7]:
```
df.isnull().any()
```

Out[7]:
```
Unnamed: 0.1             False
Unnamed: 0               False
App                      False
Category                 False
Rating                   False
Reviews                  False
Size                     False
Installs                 False
Type                     False
Price                    False
Content Rating            True
Last Updated             False
Current Ver              False
Minimum Android Ver      False
Genres                   False
dtype: bool
```

In [8]:

```
#Lets Check out the null value
df.isnull().sum()
```

Out[8]:

```
Unnamed: 0.1            0
Unnamed: 0              0
App                     0
Category                0
Rating                  0
Reviews                 0
Size                    0
Installs                0
Type                    0
Price                   0
Content Rating        361
Last Updated            0
Current Ver             0
Minimum Android Ver     0
Genres                  0
dtype: int64
```

In [9]:

```
df = df.dropna()
```

In [10]:

```
df.isnull().any()
```

Out[10]:

```
Unnamed: 0.1          False
Unnamed: 0            False
App                   False
Category              False
Rating                False
Reviews               False
Size                  False
Installs              False
Type                  False
Price                 False
Content Rating        False
Last Updated          False
Current Ver           False
Minimum Android Ver   False
Genres                False
dtype: bool
```

In [11]:

```
df.shape
```

Out[11]:

```
(7915, 15)
```

In [14]:

```
df["Size"] = 1000 * df["Size"]
```

In [15]:
```
df
```

Out[15]:

	Unnamed: 0.1	Unnamed: 0	App	Category	Rating	Reviews	Size	Installs	Type
0	0	0	Photo Editor & Candy Camera & Grid & ScrapBook	Art And Design	4.1	159	19000.00	10000	Free
1	1	1	Coloring book moana	Art And Design	3.9	967	14000.00	500000	Free
2	2	5	U Launcher Lite – FREE Live Cool Themes, Hide ...	Art And Design	4.7	87510	8700.00	5000000	Free
3	3	6	Sketch - Draw & Paint	Art And Design	4.5	215644	25000.00	50000000	Free
4	4	7	Pixel Draw - Number Art Coloring Book	Art And Design	4.3	967	2800.00	100000	Free
...	...	...	...	...	...	...	...	...	...
8270	8270	8911	Chemin (fr)	Books And Reference	4.8	44	604.49	1000	Free
8271	8271	8912	FR Calculator	Family	4.0	7	2600.00	500	Free
8272	8272	8913	Sya9a Maroc - FR	Family	4.5	38	53000.00	5000	Free
8273	8273	8914	Fr. Mike Schmitz Audio Teachings	Family	5.0	4	3600.00	100	Free
8275	8275	8916	iHoroscope - 2018 Daily Horoscope & Astrology	Lifestyle	4.5	398307	19000.00	10000000	Free

7915 rows × 15 columns

In [16]:

```
df["Reviews"] = df["Reviews"].astype(float)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7915 entries, 0 to 8275
Data columns (total 15 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Unnamed: 0.1         7915 non-null   int64  
 1   Unnamed: 0           7915 non-null   int64  
 2   App                  7915 non-null   object 
 3   Category             7915 non-null   object 
 4   Rating               7915 non-null   float64
 5   Reviews              7915 non-null   float64
 6   Size                 7915 non-null   float64
 7   Installs             7915 non-null   int64  
 8   Type                 7915 non-null   object 
 9   Price                7915 non-null   float64
 10  Content Rating       7915 non-null   object 
 11  Last Updated         7915 non-null   object 
 12  Current Ver          7915 non-null   object 
 13  Minimum Android Ver  7915 non-null   object 
 14  Genres               7915 non-null   object 
dtypes: float64(4), int64(3), object(8)
memory usage: 989.4+ KB
```

In [17]:

```
df["Price"] = df["Price"].astype(int)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7915 entries, 0 to 8275
Data columns (total 15 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Unnamed: 0.1         7915 non-null   int64  
 1   Unnamed: 0           7915 non-null   int64  
 2   App                  7915 non-null   object 
 3   Category             7915 non-null   object 
 4   Rating               7915 non-null   float64
 5   Reviews              7915 non-null   float64
 6   Size                 7915 non-null   float64
 7   Installs             7915 non-null   int64  
 8   Type                 7915 non-null   object 
 9   Price                7915 non-null   int32  
 10  Content Rating       7915 non-null   object 
 11  Last Updated         7915 non-null   object 
 12  Current Ver          7915 non-null   object 
 13  Minimum Android Ver  7915 non-null   object 
 14  Genres               7915 non-null   object 
dtypes: float64(3), int32(1), int64(3), object(8)
memory usage: 958.5+ KB
```

In [18]:

```
df["Installs"] = df["Installs"].astype(int)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7915 entries, 0 to 8275
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Unnamed: 0.1        7915 non-null   int64  
 1   Unnamed: 0          7915 non-null   int64  
 2   App                 7915 non-null   object 
 3   Category            7915 non-null   object 
 4   Rating              7915 non-null   float64
 5   Reviews             7915 non-null   float64
 6   Size                7915 non-null   float64
 7   Installs            7915 non-null   int32  
 8   Type                7915 non-null   object 
 9   Price               7915 non-null   int32  
 10  Content Rating      7915 non-null   object 
 11  Last Updated        7915 non-null   object 
 12  Current Ver         7915 non-null   object 
 13  Minimum Android Ver 7915 non-null   object 
 14  Genres              7915 non-null   object 
dtypes: float64(3), int32(2), int64(2), object(8)
memory usage: 927.5+ KB
```

In [19]:

```
df.shape
```

Out[19]:

```
(7915, 15)
```

In [20]:

```
df.drop(df[(df['Reviews'] < 1) & (df['Reviews'] > 5 )].index, inplace = True)
```

In [21]:

```
df.shape
```

Out[21]:

```
(7915, 15)
```

In [22]:

```
df.drop(df[df['Installs'] < df['Reviews'] ].index, inplace = True)
```

In [23]:

```
df.shape
```

Out[23]:

```
(7908, 15)
```

In [24]:

```
df.drop(df[(df['Type'] =='Free') & (df['Price'] > 0 )].index, inplace = True)
```

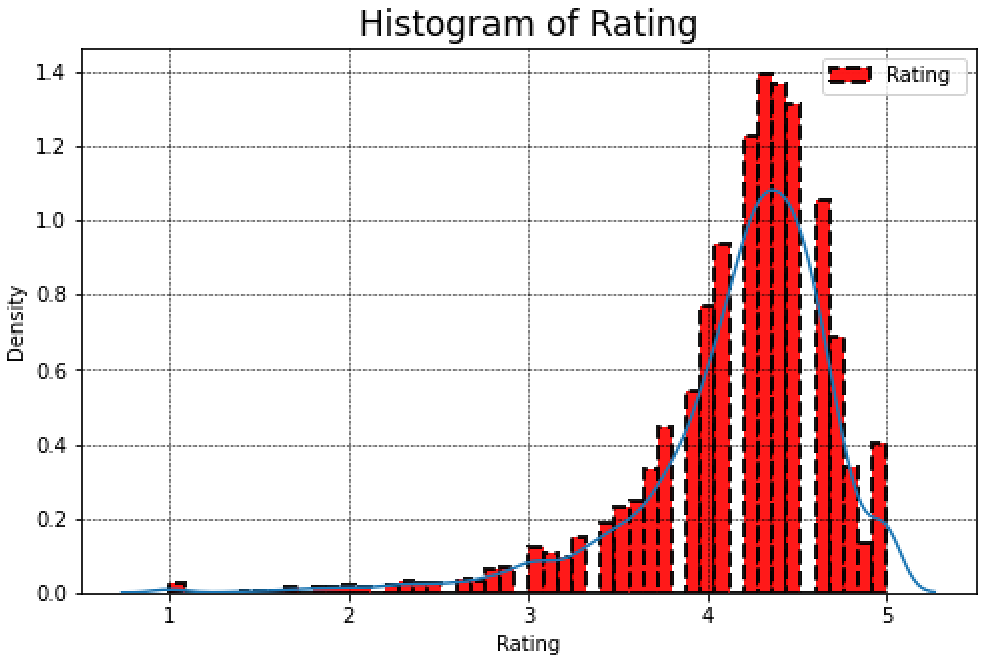
In [26]:

```
df.shape
```

Out[26]:

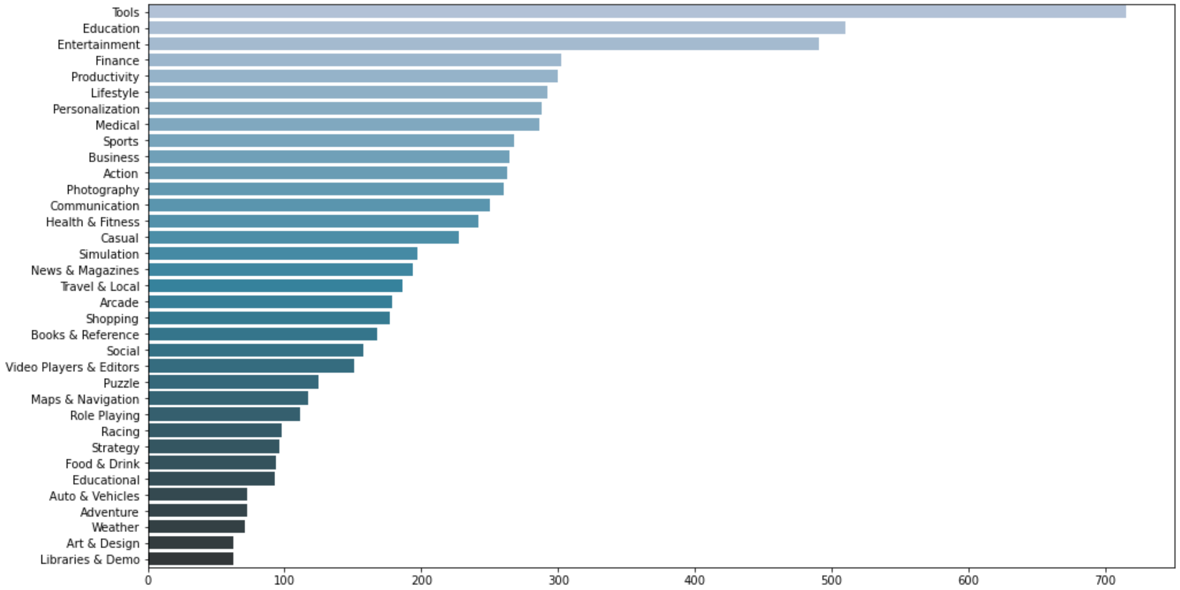
```
(7908, 15)
```

In [27]:
```python
import matplotlib.pyplot as plt #visualisation
%matplotlib inline

import seaborn as sns

plt.figure(figsize = (8,5))


sns.distplot(df['Rating'],  kde = True,
             hist_kws = {'color':'r','edgecolor':'k','linewidth':2,'linestyle'
             label = 'Rating ')

plt.title("Histogram of Rating", fontsize = 17)
plt.xticks()
plt.grid(color = 'k', linestyle = '--', linewidth = 0.5)
plt.legend()
plt.show()
print('- Total number of ratings:', len(df['Rating']))
print('- Mean of distribution of rating :', np.mean(df['Rating']))
print('- Standard deviation:', np.std(df['Rating']))
```

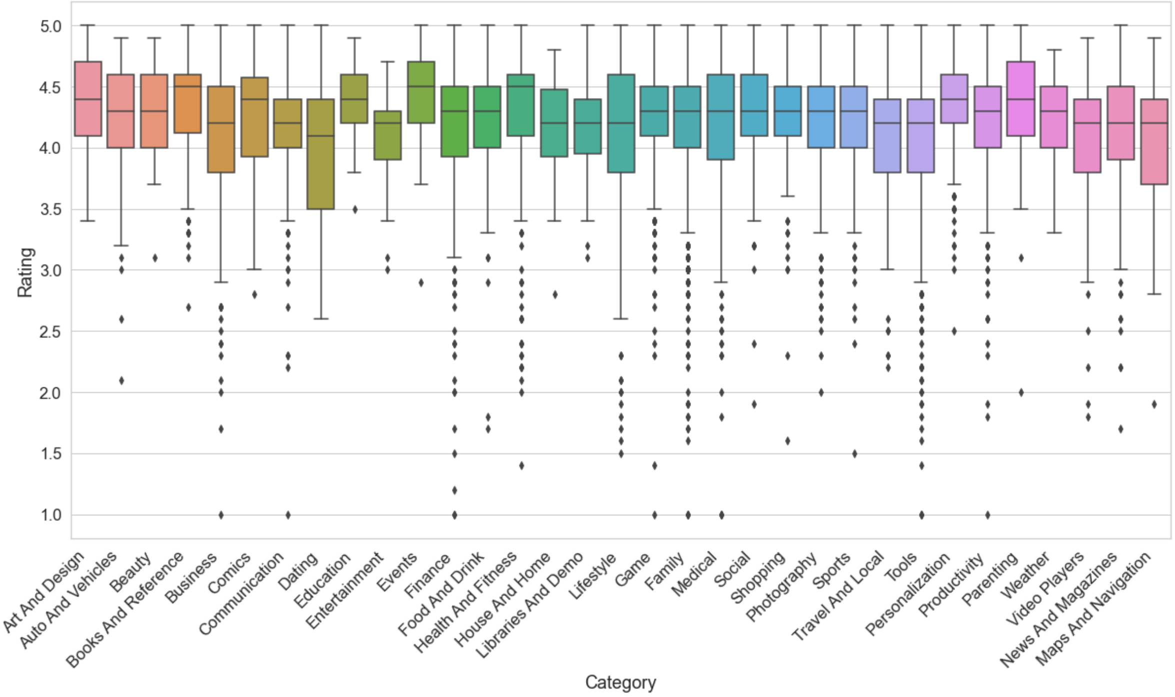


```
- Total number of ratings: 7908
- Mean of distribution of rating : 4.17678300455235
- Standard deviation: 0.5355492691441196
```

In [28]:
```
#Show top 35 app genres
plt.figure(figsize=(16, 9))
genres = df["Genres"].value_counts()[:35]
ax = sns.barplot(x=genres.values, y=genres.index, palette="PuBuGn_d")
```

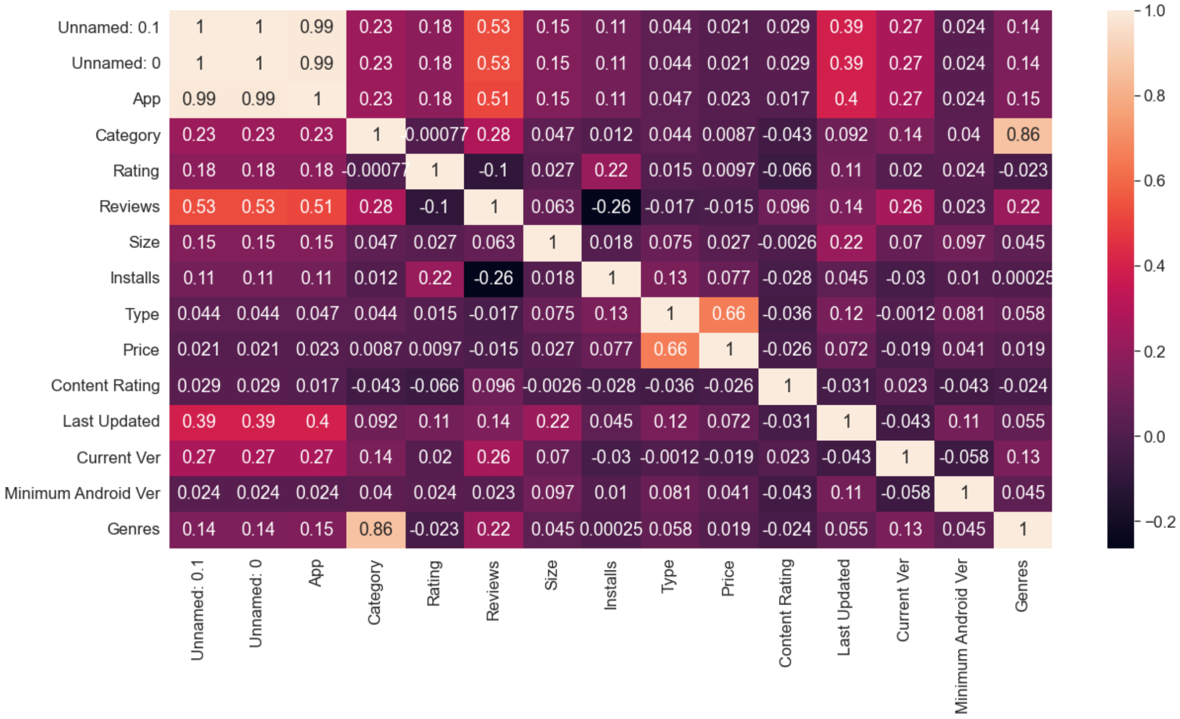


In [29]:
```
#Which categories have the best overall rating? Also, which category had the m
import seaborn as sns

sns.set(rc={'figure.figsize':(20,10)}, font_scale=1.5, style='whitegrid')
ax = sns.boxplot(x="Category",y="Rating",data=df)
labels = ax.set_xticklabels(ax.get_xticklabels(), rotation=45,ha='right')
# All of the categories have close rating averages.Events category has best ra
```



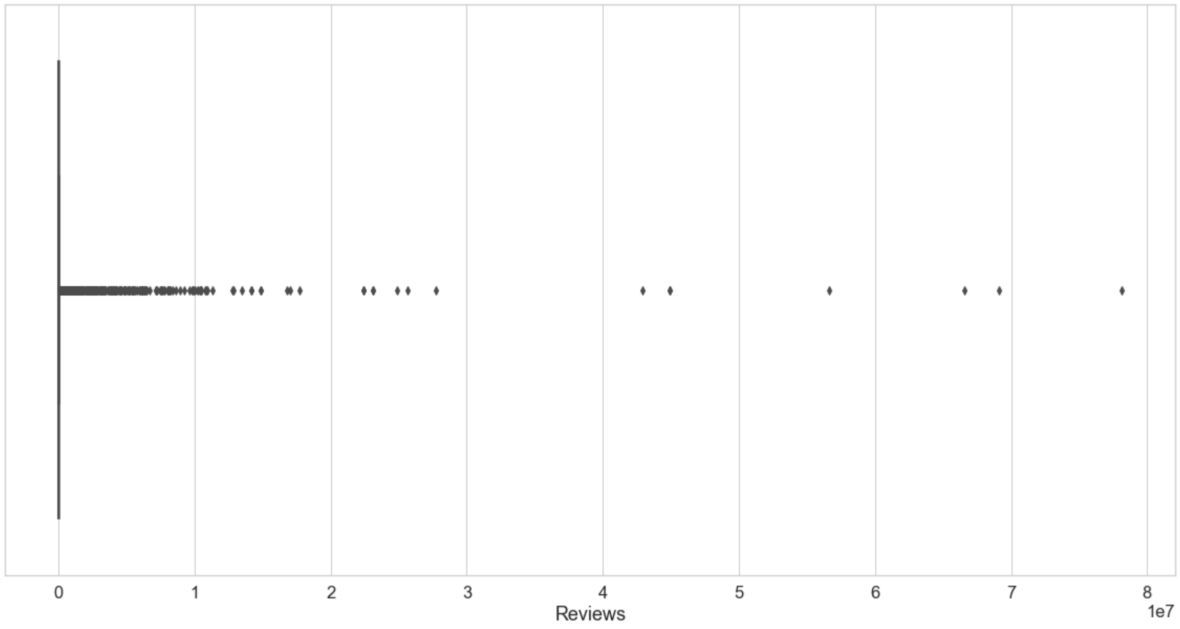
In [30]:
```
#df.dtypes
df["Type"] = (df["Type"] == "Paid").astype(int)
corr = df.apply(lambda x: x.factorize()[0]).corr()
sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns,annot=Tru
# we can see that Installs and Reviews has the strongest inverse correlation.
```

Out[30]:
```
<AxesSubplot:>
```

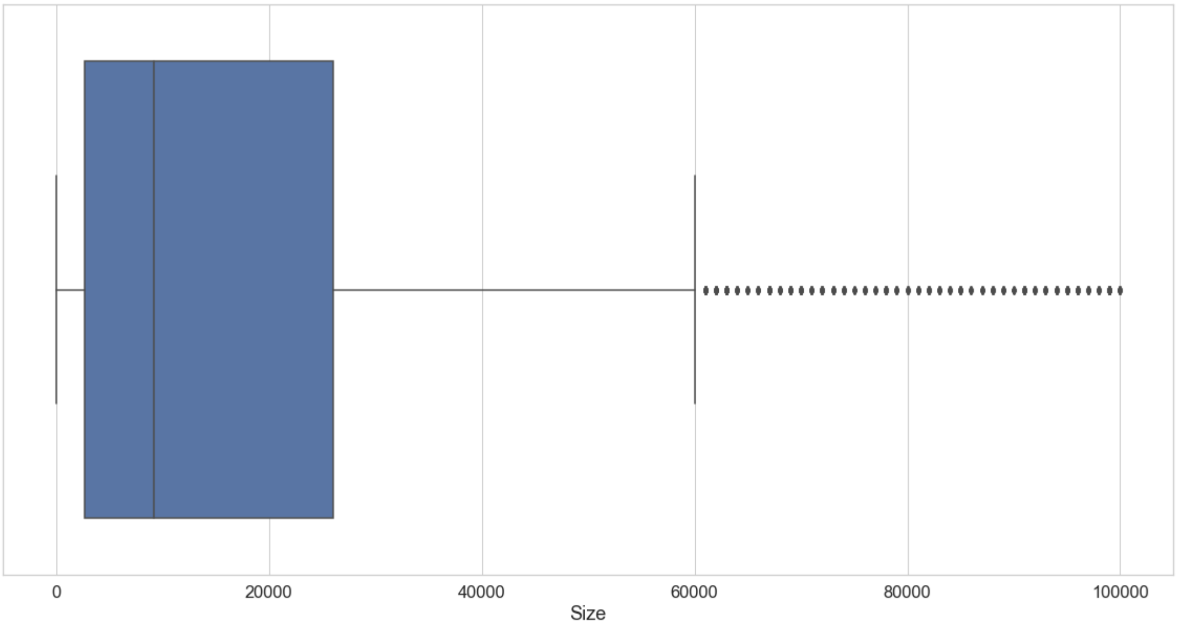


In [31]:
```
sns.boxplot(df['Reviews'])
```

Out[31]:
```
<AxesSubplot:xlabel='Reviews'>
```



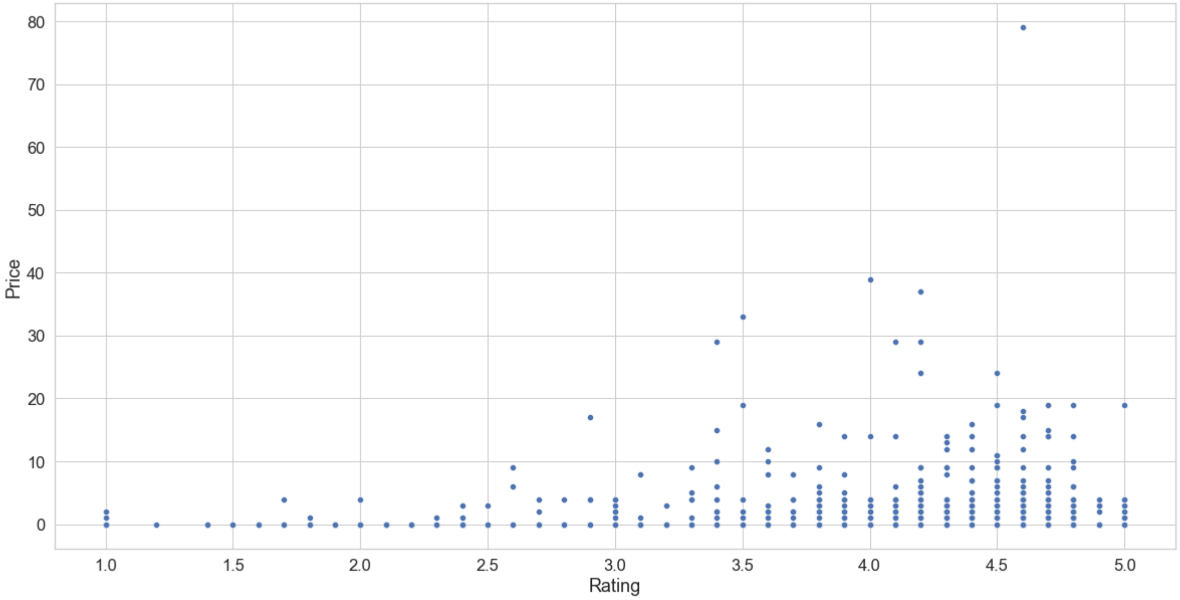
```
In [33]: sns.boxplot(df['Size'])

Out[33]: <AxesSubplot:xlabel='Size'>
```



```
In [35]: more = df.apply(lambda x : True
                     if x['Price'] > 200 else False, axis = 1)

In [36]: more_count = len(more[more == True].index)

In [38]: df.shape

Out[38]: (7908, 15)

In [39]: df.drop(df[df['Price'] > 200].index, inplace = True)
         df.shape

Out[39]: (7893, 15)

In [40]: df.drop(df[df['Reviews'] > 2000000].index, inplace = True)
         df.shape

Out[40]: (7689, 15)

In [41]: # dropping more than 10000000 Installs value
         df.drop(df[df['Installs'] > 10000000].index, inplace = True)
         df.shape

Out[41]: (7439, 15)
```

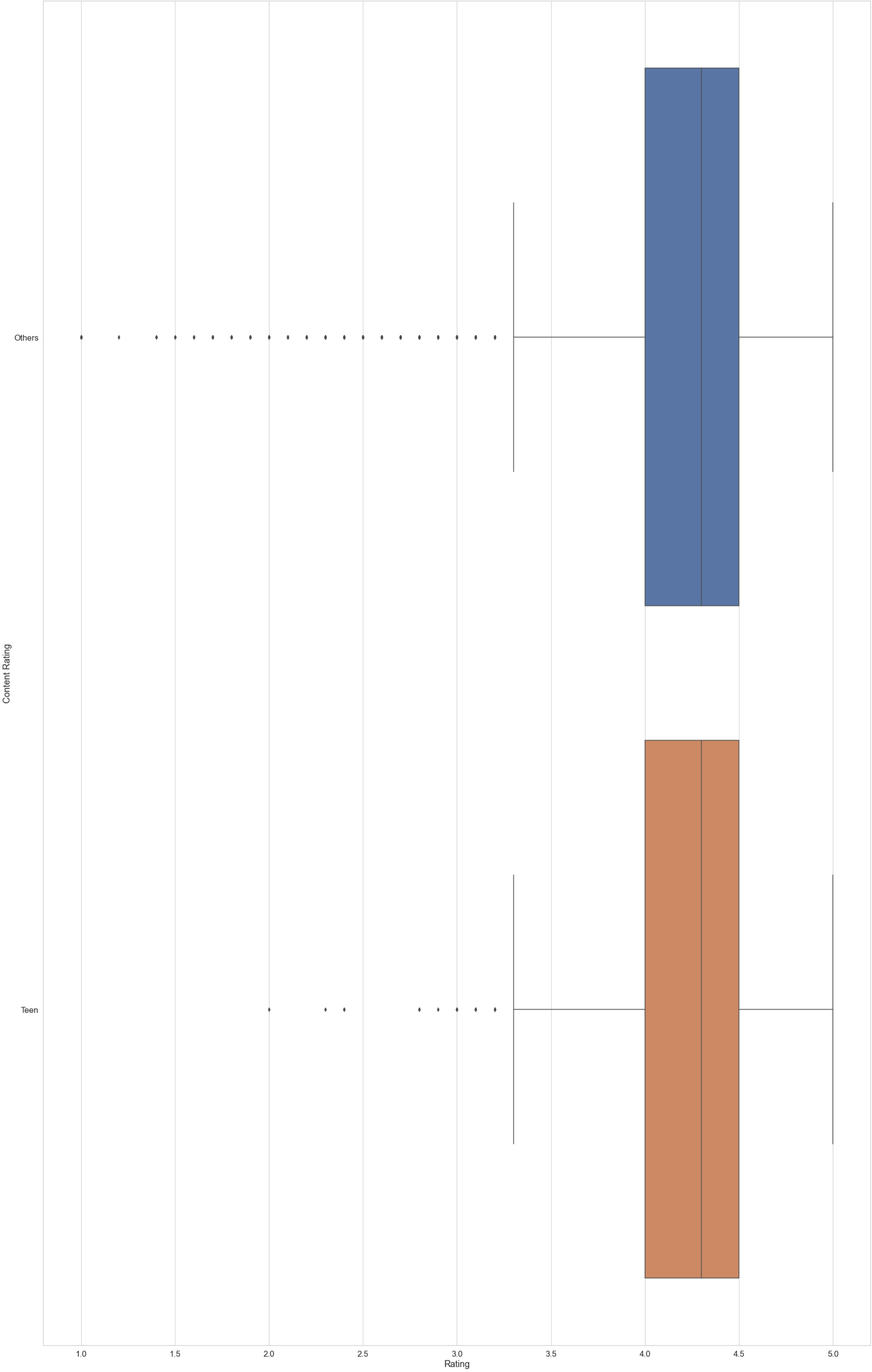
```
In [42]: sns.scatterplot(x='Rating',y='Price',data=df)
         #heavior apps are rated better.
```

```
Out[42]: <AxesSubplot:xlabel='Rating', ylabel='Price'>
```



In [47]:
```
sns.boxplot(x="Rating", y="Content Rating", data=df)
```

Out[47]:
```
<AxesSubplot:xlabel='Rating', ylabel='Content Rating'>
```

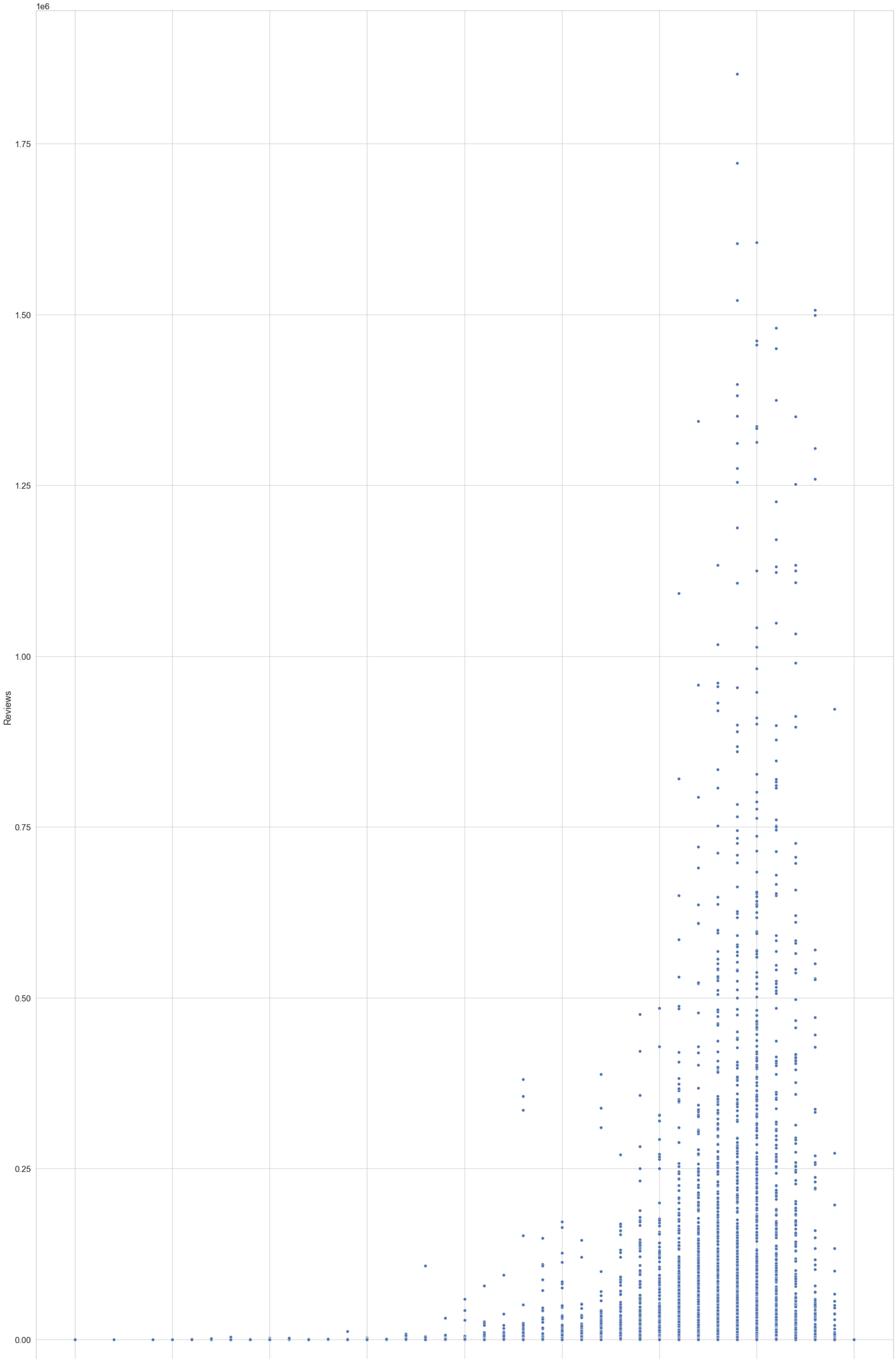

Others
Teen
Content Rating
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Rating

In [48]:

```
sns.scatterplot(x='Rating',y='Reviews',data=df)
# more reviews makes app rating better.
```

Out[48]:

```
<AxesSubplot:xlabel='Rating', ylabel='Reviews'>
```


1e6
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Reviews
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Rating

In [49]:
```
#Drop the columns that are not depend on rating values
df.drop(['App','Current Ver','Minimum Android Ver' ,'Unnamed: 0.1','Unnamed: 0
df
```

Out[49]:

	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Last Updated	Genre
0	Art And Design	4.1	159.0	19000.00	10000	0	0	Others	January 7, 2018	Art & Desig
1	Art And Design	3.9	967.0	14000.00	500000	0	0	Others	January 15, 2018	Art & Desig
2	Art And Design	4.7	87510.0	8700.00	5000000	0	0	Others	August 1, 2018	Art & Desig
4	Art And Design	4.3	967.0	2800.00	100000	0	0	Others	June 20, 2018	Art & Desig
5	Art And Design	4.4	167.0	5600.00	50000	0	0	Others	March 26, 2017	Art & Desig
...	...	...	...	...	...	...	...	...	...	.
8270	Books And Reference	4.8	44.0	604.49	1000	0	0	Others	March 23, 2014	Books & Referenc
8271	Family	4.0	7.0	2600.00	500	0	0	Others	June 18, 2017	Educatio
8272	Family	4.5	38.0	53000.00	5000	0	0	Others	July 25, 2017	Educatio
8273	Family	5.0	4.0	3600.00	100	0	0	Others	July 6, 2018	Educatio
8275	Lifestyle	4.5	398307.0	19000.00	10000000	0	0	Others	July 25, 2018	Lifestyl

7439 rows × 10 columns

In [53]:
```
#Let's apply Dummy EnCoding on Column "Category"
df1 =df
```

In [54]:
```
#get unique values in Column "Category"
df1.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'],
      dtype=object)
```

In [55]:
```
df1.Category = pd.Categorical(df1.Category)

x = df1[['Category']]
del df1['Category']

dummies = pd.get_dummies(x, prefix = 'Category')
df1 = pd.concat([df1,dummies], axis=1)
df1.head()
```

Out[55]:

	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Last Updated	Genres	Category_Art And Design	...
0	4.1	159.0	19000.0	10000	0	0	Others	January 7, 2018	Art & Design	1	...
1	3.9	967.0	14000.0	500000	0	0	Others	January 15, 2018	Art & Design	1	...
2	4.7	87510.0	8700.0	5000000	0	0	Others	August 1, 2018	Art & Design	1	...
4	4.3	967.0	2800.0	100000	0	0	Others	June 20, 2018	Art & Design	1	...
5	4.4	167.0	5600.0	50000	0	0	Others	March 26, 2017	Art & Design	1	...

5 rows × 42 columns

In [56]:
```
df1.shape
```

Out[56]:
```
(7439, 42)
```

In [57]:
```
#Let's apply Dummy EnCoding on Column "Genres"
#get unique values in Column "Genres"
df1["Genres"].unique()
```

Out[57]:
```
array(['Art & Design', 'Auto & Vehicles', 'Beauty', 'Books & Reference',
       'Business', 'Comics', 'Communication', 'Dating', 'Education',
       'Entertainment', 'Events', 'Finance', 'Food & Drink',
       'Health & Fitness', 'House & Home', 'Libraries & Demo',
       'Lifestyle', 'Card', 'Casual', 'Puzzle', 'Arcade', 'Word',
       'Racing', 'Sports', 'Action', 'Board', 'Simulation',
       'Role Playing', 'Adventure', 'Strategy', 'Trivia', 'Educational',
       'Music', 'Music & Audio', 'Video Players & Editors', 'Medical',
       'Social', 'Shopping', 'Photography', 'Travel & Local', 'Tools',
       'Personalization', 'Productivity', 'Parenting', 'Weather',
       'News & Magazines', 'Maps & Navigation', 'Casino'], dtype=object)
```

In [59]:
```
df1.Genres = pd.Categorical(df1['Genres'])
x = df1[["Genres"]]
del df1['Genres']
dummies = pd.get_dummies(x, prefix = 'Genres')
df1 = pd.concat([df1,dummies], axis=1)
```

In [61]:

```
df1.head()
```

Out[61]:

	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Last Updated	Category_Art And Design	Category_A And Vehi
0	4.1	159.0	19000.0	10000	0	0	Others	January 7, 2018	1	
1	3.9	967.0	14000.0	500000	0	0	Others	January 15, 2018	1	
2	4.7	87510.0	8700.0	5000000	0	0	Others	August 1, 2018	1	
4	4.3	967.0	2800.0	100000	0	0	Others	June 20, 2018	1	
5	4.4	167.0	5600.0	50000	0	0	Others	March 26, 2017	1	

5 rows × 86 columns

In [62]:

```
df1.shape
```

Out[62]:

```
(7439, 86)
```

In [63]:

```
#get unique values in Column "Content Rating"
df1["Content Rating"].unique()
```

Out[63]:

```
array(['Others', 'Teen'], dtype=object)
```

In [64]:

```
df1['Content Rating'] = pd.Categorical(df1['Content Rating'])

x = df1[['Content Rating']]
del df1['Content Rating']

dummies = pd.get_dummies(x, prefix = 'Content Rating')
df1 = pd.concat([df1,dummies], axis=1)
df1.head()
```

Out[64]:

	Rating	Reviews	Size	Installs	Type	Price	Last Updated	Category_Art And Design	Category_Auto And Vehicles	Cate
0	4.1	159.0	19000.0	10000	0	0	January 7, 2018	1	0	
1	3.9	967.0	14000.0	500000	0	0	January 15, 2018	1	0	
2	4.7	87510.0	8700.0	5000000	0	0	August 1, 2018	1	0	
4	4.3	967.0	2800.0	100000	0	0	June 20, 2018	1	0	
5	4.4	167.0	5600.0	50000	0	0	March 26, 2017	1	0	

5 rows × 87 columns

In [65]:

```
df1.shape
```

Out[65]:

```
(7439, 87)
```

In [66]:

```
df1.skew()
```

Out[66]:

```
Rating                        -1.705801
Reviews                        5.178585
Size                           1.699823
Installs                       1.795538
Type                           3.184014
                                 ...
Genres_Trivia                 16.511575
Genres_Video Players & Editors  7.307233
Genres_Weather                10.396275
Content Rating_Others         -2.438105
Content Rating_Teen            2.438105
Length: 86, dtype: float64
```

In [67]:

```
reviewskew = np.log1p(df1['Reviews'])
df1['Reviews'] = reviewskew
```

In [68]:

```
reviewskew.skew()
```

Out[68]:

```
-0.11368967711566853
```

In [69]:
```
installsskew = np.log1p(df1['Installs'])
df1['Installs']
```

Out[69]:
```
0          10000
1         500000
2        5000000
4         100000
5          50000
          ...   
8270        1000
8271         500
8272        5000
8273         100
8275    10000000
Name: Installs, Length: 7439, dtype: int32
```

In [70]:
```
installsskew.skew()
```

Out[70]:
```
-0.4286169799756433
```

In [71]:
```
df1.head()
```

Out[71]:

	Rating	Reviews	Size	Installs	Type	Price	Last Updated	Category_Art And Design	Category_Auto And Vehicles	Ca
0	4.1	5.075174	19000.0	10000	0	0	January 7, 2018	1	0	
1	3.9	6.875232	14000.0	500000	0	0	January 15, 2018	1	0	
2	4.7	11.379520	8700.0	5000000	0	0	August 1, 2018	1	0	
4	4.3	6.875232	2800.0	100000	0	0	June 20, 2018	1	0	
5	4.4	5.123964	5600.0	50000	0	0	March 26, 2017	1	0	

5 rows × 87 columns

In [73]:

```
#drop lastupdated column
df1.drop(['Last Updated'],axis=1,inplace=True)
df1
```

Out[73]:

	Rating	Reviews	Size	Installs	Type	Price	Category_Art And Design	Category_Auto And Vehicles	Catego
0	4.1	5.075174	19000.00	10000	0	0	1	0	
1	3.9	6.875232	14000.00	500000	0	0	1	0	
2	4.7	11.379520	8700.00	5000000	0	0	1	0	
4	4.3	6.875232	2800.00	100000	0	0	1	0	
5	4.4	5.123964	5600.00	50000	0	0	1	0	
...	...	...	...	...	...	...	...	...	
8270	4.8	3.806662	604.49	1000	0	0	0	0	
8271	4.0	2.079442	2600.00	500	0	0	0	0	
8272	4.5	3.663562	53000.00	5000	0	0	0	0	
8273	5.0	1.609438	3600.00	100	0	0	0	0	
8275	4.5	12.894981	19000.00	10000000	0	0	0	0	

7439 rows × 86 columns

In [74]:

```
df1 = df1.values
df1
```

Out[74]:

```
array([[4.10000000e+00, 5.07517382e+00, 1.90000000e+04, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00],
       [3.90000000e+00, 6.87523209e+00, 1.40000000e+04, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00],
       [4.70000000e+00, 1.13795198e+01, 8.70000000e+03, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00],
       ...,
       [4.50000000e+00, 3.66356165e+00, 5.30000000e+04, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00],
       [5.00000000e+00, 1.60943791e+00, 3.60000000e+03, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00],
       [4.50000000e+00, 1.28949809e+01, 1.90000000e+04, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00]])
```

In [76]:

```
X = df1[:,1:87] #Predictors
y = df1[:,0] #Target
print(X)
print(y)
```

```
[[5.07517382e+00 1.90000000e+04 1.00000000e+04 ... 0.00000000e+00
  1.00000000e+00 0.00000000e+00]
 [6.87523209e+00 1.40000000e+04 5.00000000e+05 ... 0.00000000e+00
  1.00000000e+00 0.00000000e+00]
 [1.13795198e+01 8.70000000e+03 5.00000000e+06 ... 0.00000000e+00
  1.00000000e+00 0.00000000e+00]
 ...
 [3.66356165e+00 5.30000000e+04 5.00000000e+03 ... 0.00000000e+00
  1.00000000e+00 0.00000000e+00]
 [1.60943791e+00 3.60000000e+03 1.00000000e+02 ... 0.00000000e+00
  1.00000000e+00 0.00000000e+00]
 [1.28949809e+01 1.90000000e+04 1.00000000e+07 ... 0.00000000e+00
  1.00000000e+00 0.00000000e+00]]
[4.1 3.9 4.7 ... 4.5 5.  4.5]
```

In [77]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, rar
```

In [78]:

```
print(X_train.shape)
print(X_test.shape)
```

```
(5951, 85)
(1488, 85)
```

In [82]:

```
from sklearn.model_selection import train_test_split as tts
from sklearn.linear_model import LinearRegression as LR
from sklearn.metrics import mean_squared_error as mse
```

In [85]:

```
reg_all = LR()
reg_all.fit(X_train,y_train)
```

Out[85]:

```
LinearRegression()
```

In [87]:

```
R2_train = round(reg_all.score(X_train,y_train),2)
print("The R2 value of the Training Set is : {}".format(R2_train))
```

```
The R2 value of the Training Set is : 0.07
```

In [88]:

```
R2_test = round(reg_all.score(X_test,y_test),2)
print("The R2 value of the Testing Set is : {}".format(R2_test))
```

```
The R2 value of the Testing Set is : 0.06
```

In [96]:

```
In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [78]:

In [ ]:

In [ ]:

In [71]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```