Name :- Viashnavi

Project 1

Objective :- Make a model to predict the app rating, with other information about the app provided

In [74]:
```
import numpy as np
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

import os
import warnings
warnings.filterwarnings('ignore')
```

In [75]:
```
df = pd.read_csv('googleplaystore.csv')
```

In [76]:
```
df.head()
```

Out[76]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Cor Ra
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Ever
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Ever
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Ever
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Ever

In [77]:
```
print(f'Number of rows : {df.shape [0]}')
print(f'Number of columns : { df.shape[1]}')
```

```
Number of rows : 10841
Number of columns : 13
```

In [78]:
```
df.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 [79]:
```
df.duplicated().sum()

print(f"DataFrame has {df.duplicated().sum()} duplicate values")
```

```
 DataFrame has 483 duplicate values
```

In [80]:
```
df.drop_duplicates(inplace=True)

print(f" Total duplicate values : {df.duplicated().sum()}")
```

```
  Total duplicate values : 0
```

In [81]:
```
df.isnull().sum()
```

Out[81]:
```
App                  0
Category             0
Rating            1465
Reviews              0
Size                 0
Installs             0
Type                 1
Price                0
Content Rating       1
Genres               0
Last Updated         0
Current Ver          8
Android Ver          3
dtype: int64
```

In [82]:
```
## Drop records with nulls in any of the columns.

df.dropna(inplace=True)
```

In [83]:
```
df.isnull().sum()
```

Out[83]:
```
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 [84]:
```
df.shape
```

Out[84]:
```
(8886, 13)
```

In …
```
# # Variables seem to have incorrect type and inconsistent formatting.
# Size column has sizes in Kb as well as Mb.
# To analyze, you'll need to convert these to numeric.
# Extract the numeric value from the column and Multiply the value by


def size_col_processing(x):
    x= str(x.lower())
    if 'm' in x:

        val=float(x.replace('m', ''))
        val=val*1000

    elif 'k'in x:
        val=float(x.replace('k',''))

    else:
        val=0
    return val
```

In [86]:
```
df['Size']=df['Size'].apply(size_col_processing)
df.head()
```

Out[86]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19000.0	10,000+	Free	0	E
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14000.0	500,000+	Free	0	E
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8700.0	5,000,000+	Free	0	E
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25000.0	50,000,000+	Free	0	
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2800.0	100,000+	Free	0	E

In […
```
df['Price']= df['Price'].apply(lambda x :str(x).replace('$','')if '$'
df['Price']= df['Price'].apply(lambda x : float(x))
df['Reviews']=pd.to_numeric(df['Reviews'], errors ='coerce')
```

In […
```
df['Installs']=df['Installs'].apply(lambda x : str(x).replace('+','')
df['Installs']=df['Installs'].apply(lambda x : str(x).replace(',','')
df['Installs']=df['Installs'].apply(lambda x : float(x))
```

In [89]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8886 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             8886 non-null   object 
 1   Category        8886 non-null   object 
 2   Rating          8886 non-null   float64
 3   Reviews         8886 non-null   int64  
 4   Size            8886 non-null   float64
 5   Installs        8886 non-null   float64
 6   Type            8886 non-null   object 
 7   Price           8886 non-null   float64
 8   Content Rating  8886 non-null   object 
 9   Genres          8886 non-null   object 
 10  Last Updated    8886 non-null   object 
 11  Current Ver     8886 non-null   object 
 12  Android Ver     8886 non-null   object 
dtypes: float64(4), int64(1), object(8)
memory usage: 971.9+ KB
```

In [90]:
```
df.describe()
```

Out[90]:

	Rating	Reviews	Size	Installs	Price
count	8886.000000	8.886000e+03	8886.000000	8.886000e+03	8886.000000
mean	4.187959	4.730928e+05	19000.655919	1.650061e+07	0.963526
std	0.522428	2.906007e+06	23023.418686	8.640413e+07	16.194792
min	1.000000	1.000000e+00	0.000000	1.000000e+00	0.000000
25%	4.000000	1.640000e+02	2500.000000	1.000000e+04	0.000000
50%	4.300000	4.723000e+03	9400.000000	5.000000e+05	0.000000
75%	4.500000	7.131325e+04	27000.000000	5.000000e+06	0.000000
max	5.000000	7.815831e+07	100000.000000	1.000000e+09	400.000000

In […
```
# Reviews should not be more than installs as only those who installe
# If there are any such records, drop them.

df['review_check']=df['Reviews']>df['Installs']
```

In [92]:
```
df.shape
```

Out[92]:
```
(8886, 14)
```

In [93]:
```
df[df['review_check']==True].head(2)
```

Out[93]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Ge
2454	KBA-EZ Health Guide	MEDICAL	5.0	4	25000.0	1.0	Free	0.00	Everyone	Me
4663	Alarmy (Sleep If U Can) - Pro	LIFESTYLE	4.8	10249	0.0	10000.0	Paid	2.49	Everyone	Life

In [94]:
```
df=df[df['review_check']== False]
df.shape
```

Out[94]:
```
(8879, 14)
```

In [95]:
```
df['review_check'].unique()
```

Out[95]:
```
array([False])
```

In [96]:
```
df.drop('review_check',axis=1,inplace=True)
df.head(1)
```

Out[96]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Cont Rat
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19000.0	10000.0	Free	0.0	Every

In [9…
```
# For free apps (type = “Free”), the price should not be >0. Drop an

df[(df['Type']=='Free')&(df['Price']>0)]
```

Out[98]:

App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Las Update

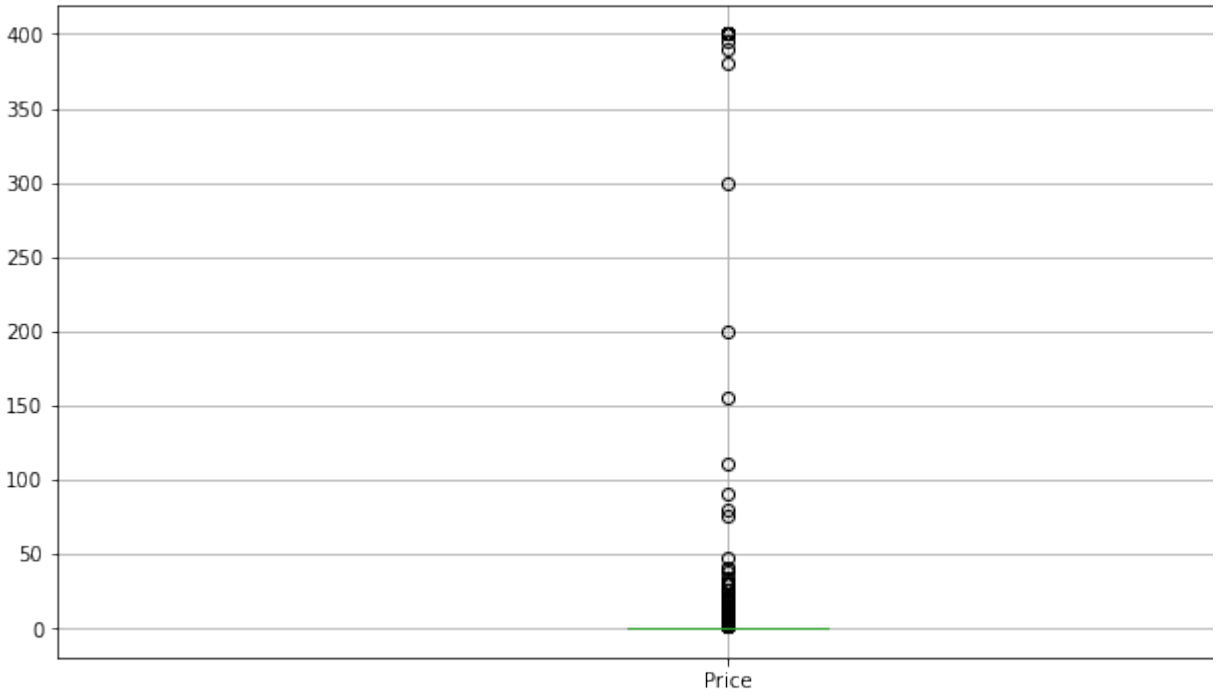
In [174]:
```
# ~ is used to Negate/reverse the df selected using condition

df=df[~((df['Type']=='Free')& (df['Price']>0))]
df.shape
```

Out[174]:
```
(6981, 13)
```

In [34]:
```
plt.figure(figsize=(12,6))
df.boxplot(column ='Price')
```

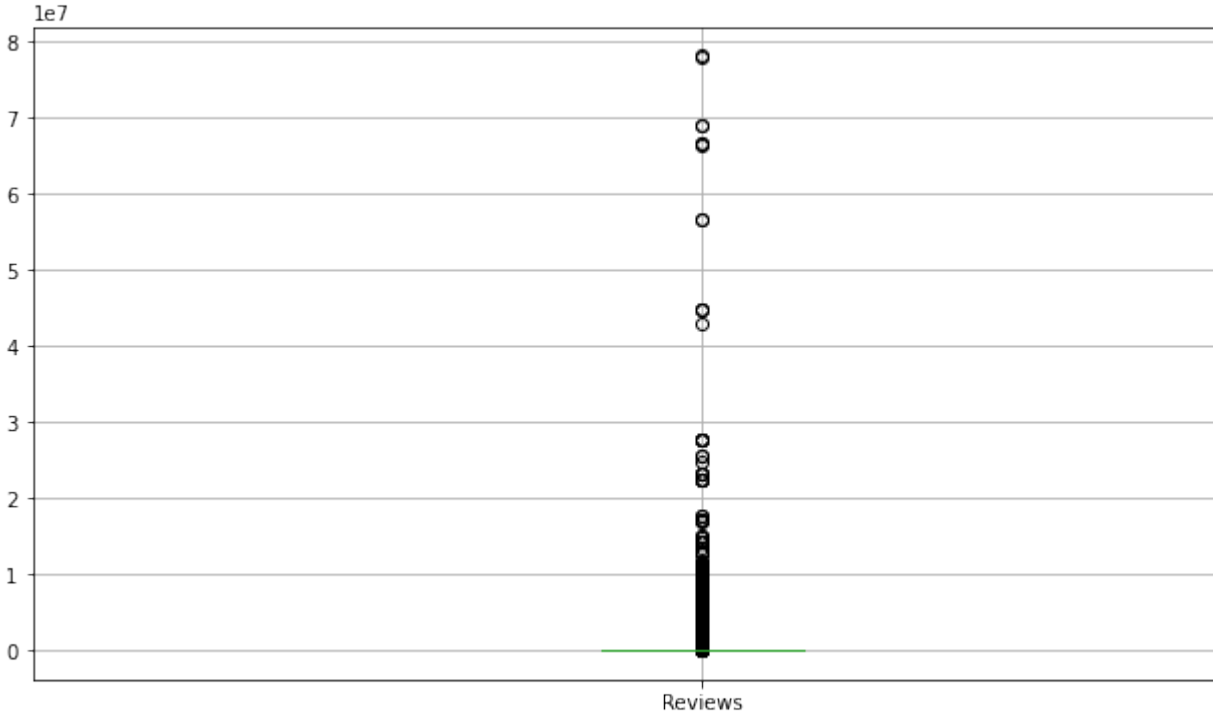
Out[34]:
```
<AxesSubplot:>
```



In [… # outlier are present in dataset , anything above than 300 will be cc

In [37]:
```
plt.figure(figsize=(12,6))
df.boxplot('Reviews')
```

Out[37]: <AxesSubplot:>

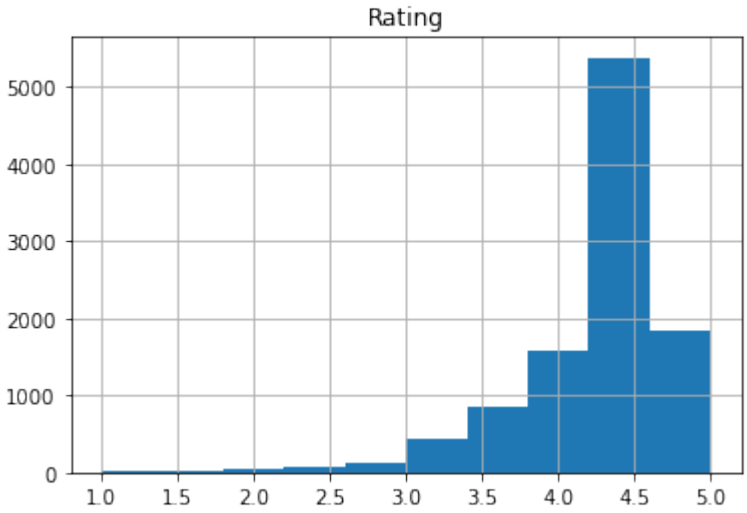


In [… # values above than 3 to le7 are the outliers in box plot shown abov

In [43]:
```
plt.figure(figsize=(12,6))
df.hist('Rating')
```

Out[43]: array([[<AxesSubplot:title={'center':'Rating'}>]], dtype=object)

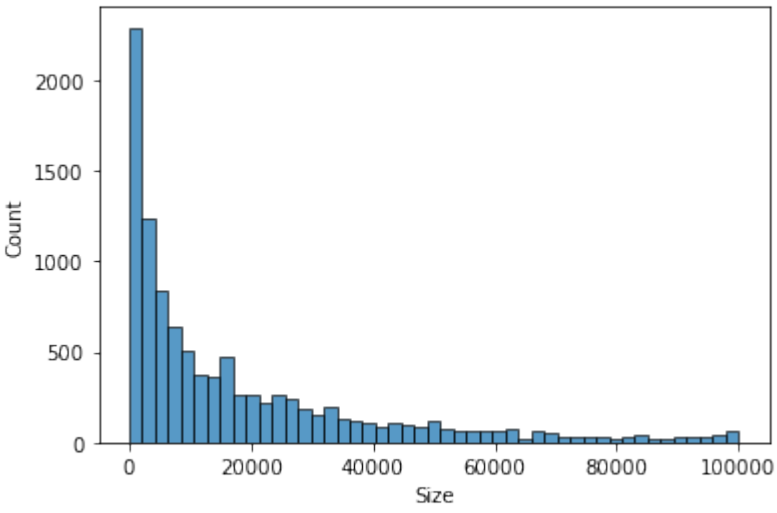
<Figure size 864x432 with 0 Axes>



In [49]:
```
# Histogram for Size

sns.histplot(df['Size'])
```

Out[49]: <AxesSubplot:xlabel='Size', ylabel='Count'>



In [50]:
```
# Most(50%) of the apps are below 20MB of size.
```

In [5…
```
# From the box plot, it seems like there are some apps with very hig
# A price of $200 for an application on the Play Store is very high
# Check out the records with very high price.
# Is 200 indeed a high price?

df=df[df['Price']>200]
df
```

Out[51]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Con Ra
4197	most expensive app (H)	FAMILY	4.3	6	1500.0	100.0	Paid	399.99	Every
4362	I'm rich	LIFESTYLE	3.8	718	26000.0	10000.0	Paid	399.99	Every
4367	I'm Rich - Trump Edition	LIFESTYLE	3.6	275	7300.0	10000.0	Paid	400.00	Every
5351	I am rich	LIFESTYLE	3.8	3547	1800.0	100000.0	Paid	399.99	Every
5354	I am Rich Plus	FAMILY	4.0	856	8700.0	10000.0	Paid	399.99	Every
5355	I am rich VIP	LIFESTYLE	3.8	411	2600.0	10000.0	Paid	299.99	Every
5356	I Am Rich Premium	FINANCE	4.1	1867	4700.0	50000.0	Paid	399.99	Every
5357	I am extremely Rich	LIFESTYLE	2.9	41	2900.0	1000.0	Paid	379.99	Every
5358	I am Rich!	FINANCE	3.8	93	22000.0	1000.0	Paid	399.99	Every
5359	I am rich(premium)	FINANCE	3.5	472	965.0	5000.0	Paid	399.99	Every
5362	I Am Rich Pro	FAMILY	4.4	201	2700.0	5000.0	Paid	399.99	Every
5364	I am rich (Most expensive app)	FINANCE	4.1	129	2700.0	1000.0	Paid	399.99	-
5366	I Am Rich	FAMILY	3.6	217	4900.0	10000.0	Paid	389.99	Every
5369	I am Rich	FINANCE	4.3	180	3800.0	5000.0	Paid	399.99	Every
5373	I AM RICH PRO PLUS	FINANCE	4.0	36	41000.0	1000.0	Paid	399.99	Every
9917	Eu Sou Rico	FINANCE	4.4	0	1400.0	0.0	Paid	394.99	Every
9934	I'm Rich/Eu sou Rico/أنا/غني	LIFESTYLE	4.4	0	40000.0	0.0	Paid	399.99	Every

In [100]:
```
# Yes $200 indeed  is a high price.
# Drop these as most seem to be junk apps

df=df[df['Price']<200]
df['Price'].unique()
```

Out[100]:
```
array([ 0.  ,  4.99,  3.99,  6.99,  7.99,  5.99,  2.99,  3.49,
1.99,
        9.99,  7.49,  0.99,  9.  ,  5.49, 10.  , 24.99, 11.99,
79.99,
       16.99, 14.99, 29.99, 12.99,  2.49, 10.99,  1.5 , 19.99,
15.99,
       33.99, 39.99,  3.95,  4.49,  1.7 ,  8.99,  1.49,  3.88,
17.99,
        3.02,  1.76,  4.84,  4.77,  1.61,  2.5 ,  1.59,  6.49,
1.29,
       37.99, 18.99,  8.49,  1.75, 14.  ,  2.  ,  3.08,  2.59,
19.4 ,
        3.9 ,  4.59, 15.46,  3.04, 13.99,  4.29,  3.28,  4.6 ,
1.  ,
        2.95,  2.9 ,  1.97,  2.56,  1.2 ])
```

In …
```
# Reviews: Very few apps have very high number of reviews. These are a
# in fact, will skew it. Drop records having more than 2 million revie

df=df[df['Reviews']<2000000]
df.head(2)
```

Out[101]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Co
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19000.0	10000.0	Free	0.0	Eve
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14000.0	500000.0	Free	0.0	Eve

In [1…
```
# Installs:  There seems to be some outliers in this field too.
# Find out the different percentiles – 10, 25, 50, 70, 90, 95, 99
# Decide a threshold as cutoff for outlier and drop records having v


df.Installs.quantile([0.10, 0.25, 0.50, 0.70, 0.90, 0.95, 0.99])
```

Out[102]:
```
0.10         1000.0
0.25        10000.0
0.50       100000.0
0.70      1000000.0
0.90     10000000.0
0.95     10000000.0
0.99    100000000.0
Name: Installs, dtype: float64
```

In […
```
# Keeping 95% value as a threshold/cutoff for outlier and drop recorc

df=df[df['Installs']<10000000.0]
df.head(2)
```

Out[103]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Co F
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19000.0	10000.0	Free	0.0	Eve
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14000.0	500000.0	Free	0.0	Eve

In [104]:
```
df.shape
```

Out[104]:
```
(6981, 13)
```

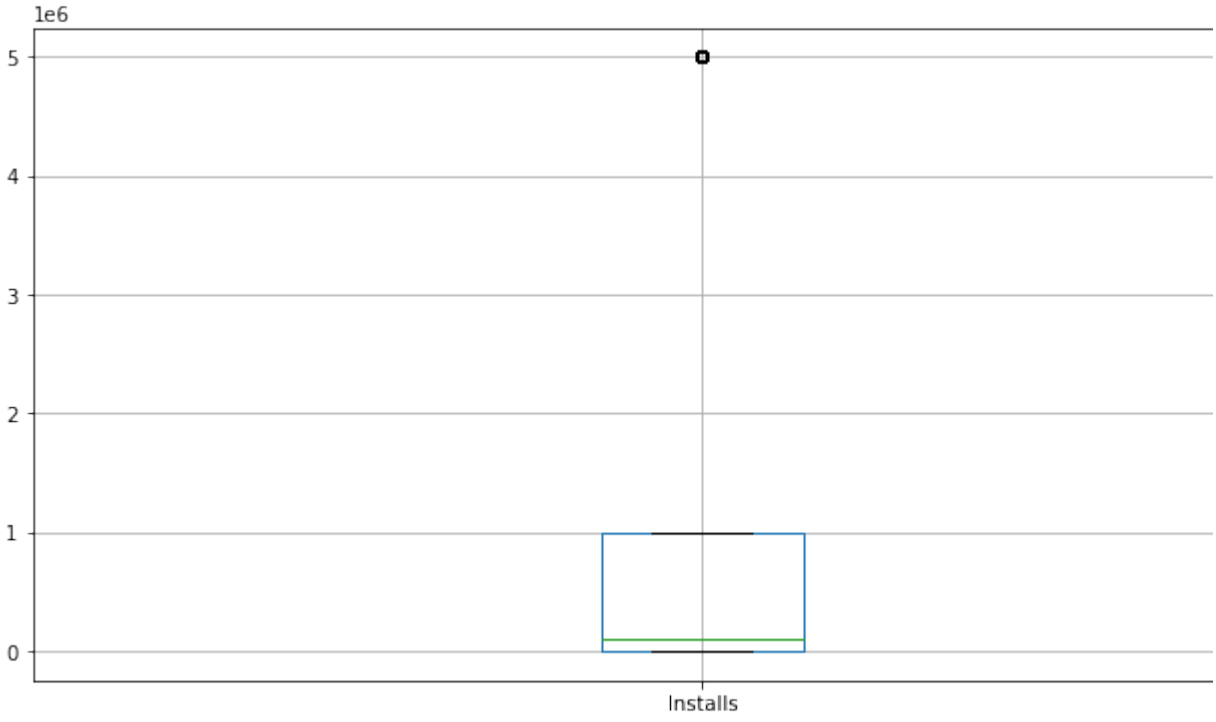
In […
```
# Make scatter plot/joinplot for Rating vs. Price
# What pattern do you observe? Does rating increase with price?
# Yes, it is showing positive correlation as the price increasing Rat

# Make scatter plot/joinplot for Rating vs. Size
# Are heavier apps rated better?
# No relation as we can see everyone is downloading any size of the a

# Make scatter plot/joinplot for Rating vs. Reviews
# Does more review mean a better rating always?
# Apps which are having higher ratings
# The app which are having higher rating are getting somewhat of a mo
# Most of the ratings are on the higher end side of the ratings.
```

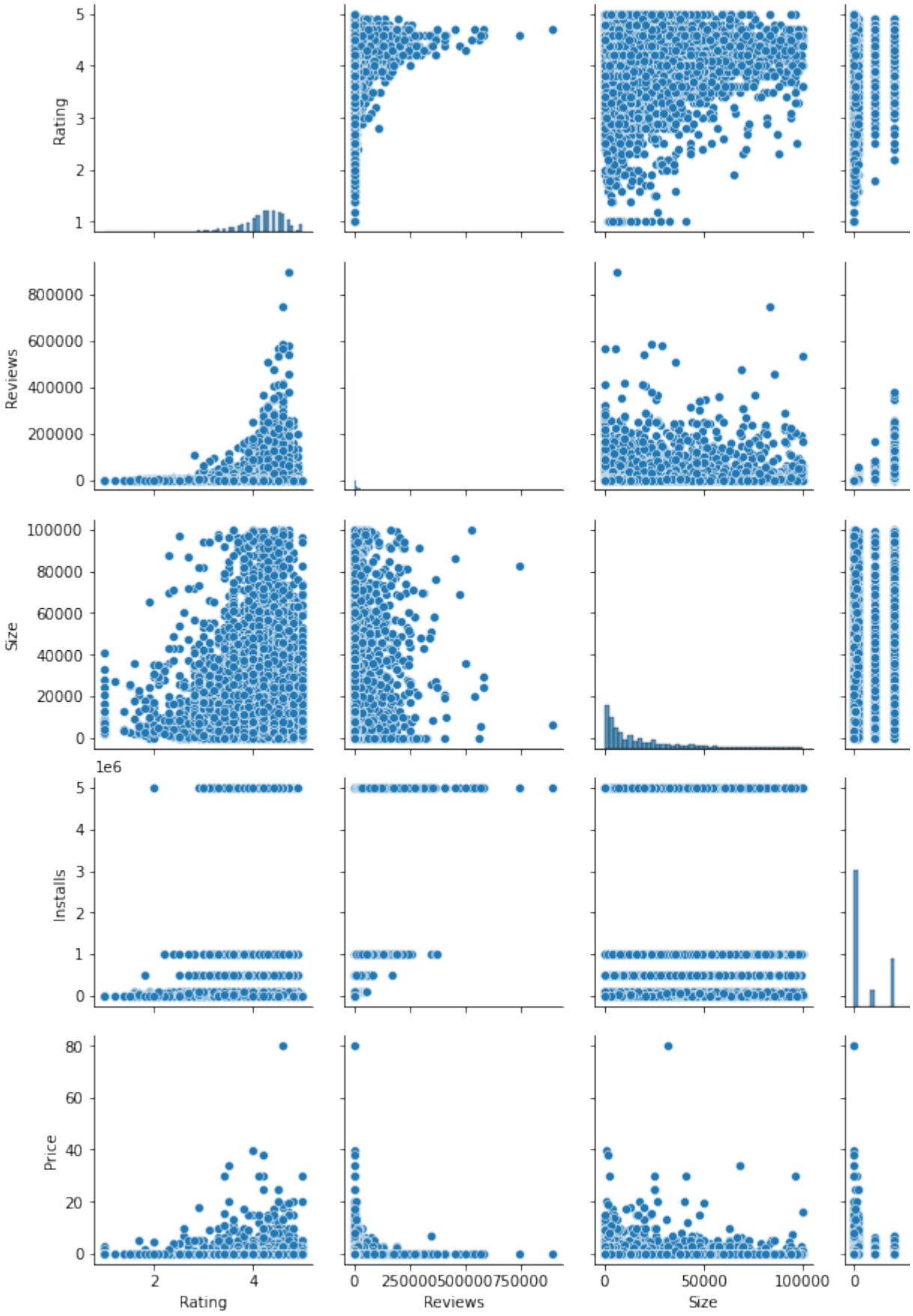
In [105]:
```
plt.figure(figsize=(12,6))
df.boxplot('Installs')
```

Out[105]:
```
<AxesSubplot:>
```



In [125]:
```
sns.pairplot(df)
```

Out[125]:
```
<seaborn.axisgrid.PairGrid at 0x1832d8b8310>
```



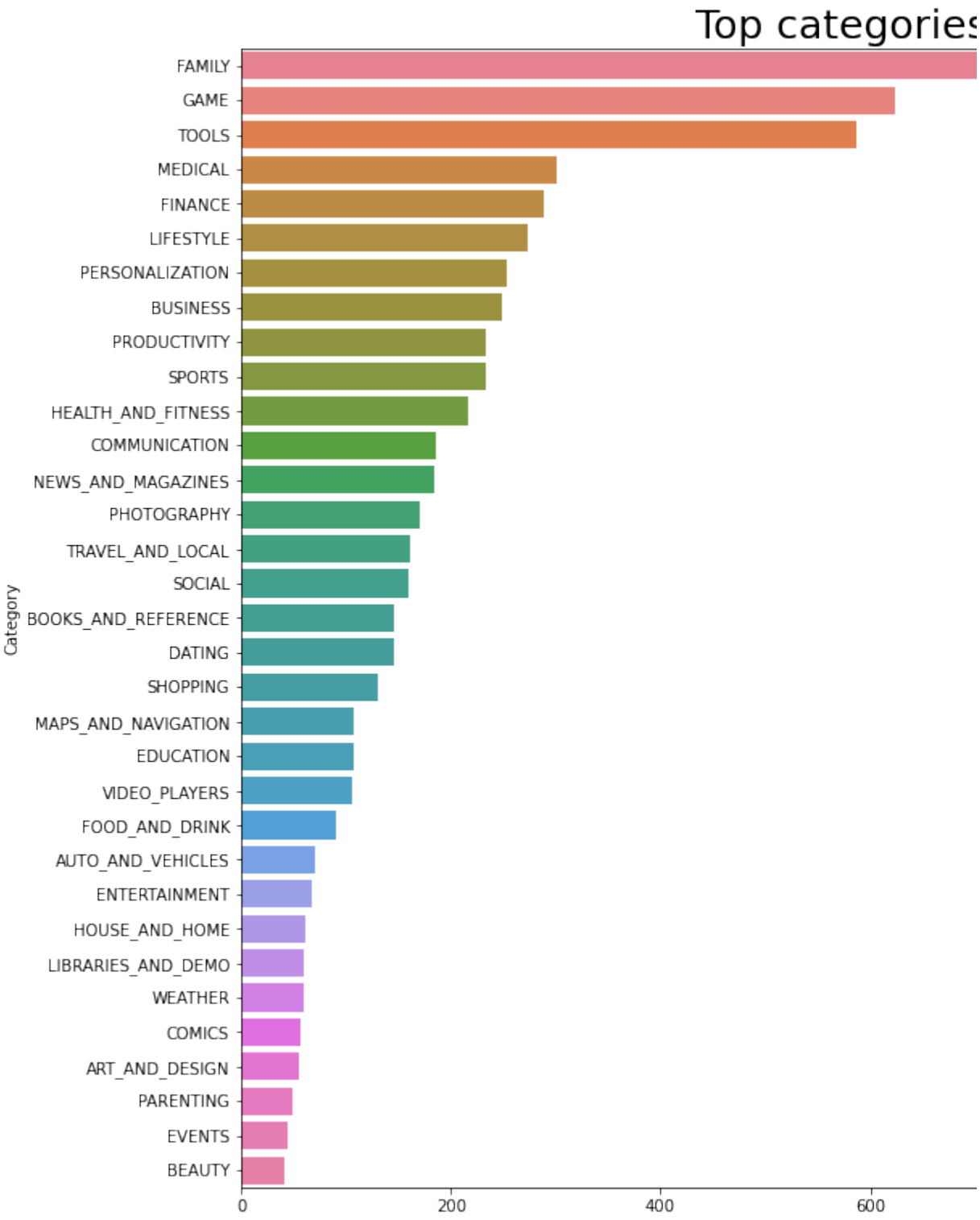
In [106]:
```
##  value count of top most app on google

x = df['Category'].value_counts()
y = df['Category'].value_counts().index

x_axis =[]
y_axis = []
for i in range(len(x)):
    x_axis.append(x[i])
    y_axis.append(y[i])
```

In [107…
```
plt.figure(figsize=(18,13))
plt.xlabel("Count")
plt.ylabel("Category")

graph = sns.barplot(x = x_axis, y = y_axis, palette= "husl")
graph.set_title("Top categories on Google Playstore", fontsize = 2
```


Top categories
FAMILY
GAME
TOOLS
MEDICAL
FINANCE
LIFESTYLE
PERSONALIZATION
BUSINESS
PRODUCTIVITY
SPORTS
HEALTH_AND_FITNESS
COMMUNICATION
NEWS_AND_MAGAZINES
PHOTOGRAPHY
TRAVEL_AND_LOCAL
SOCIAL
BOOKS_AND_REFERENCE
DATING
SHOPPING
MAPS_AND_NAVIGATION
EDUCATION
VIDEO_PLAYERS
FOOD_AND_DRINK
AUTO_AND_VEHICLES
ENTERTAINMENT
HOUSE_AND_HOME
LIBRARIES_AND_DEMO
WEATHER
COMICS
ART_AND_DESIGN
PARENTING
EVENTS
BEAUTY
Category
0
200
400
600

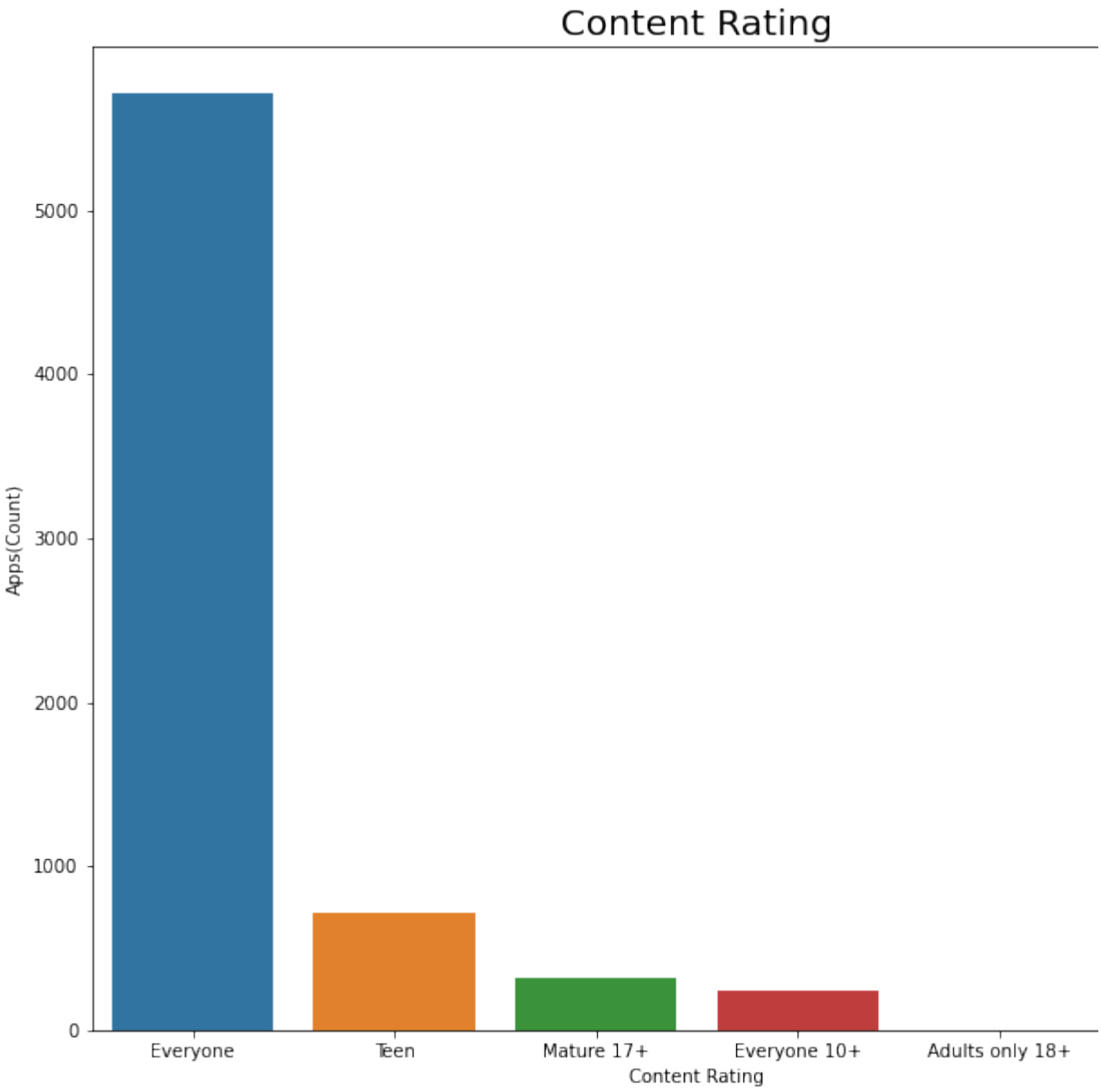
In [126]:
```
## Bar chart for Content Rating

x1 = df['Content Rating'].value_counts().index
y1 = df['Content Rating'].value_counts()

x1_axis = []
y1_axis = []

for i in range(len(x1)):
    x1_axis.append(x1[i])
    y1_axis.append(y1[i])
```

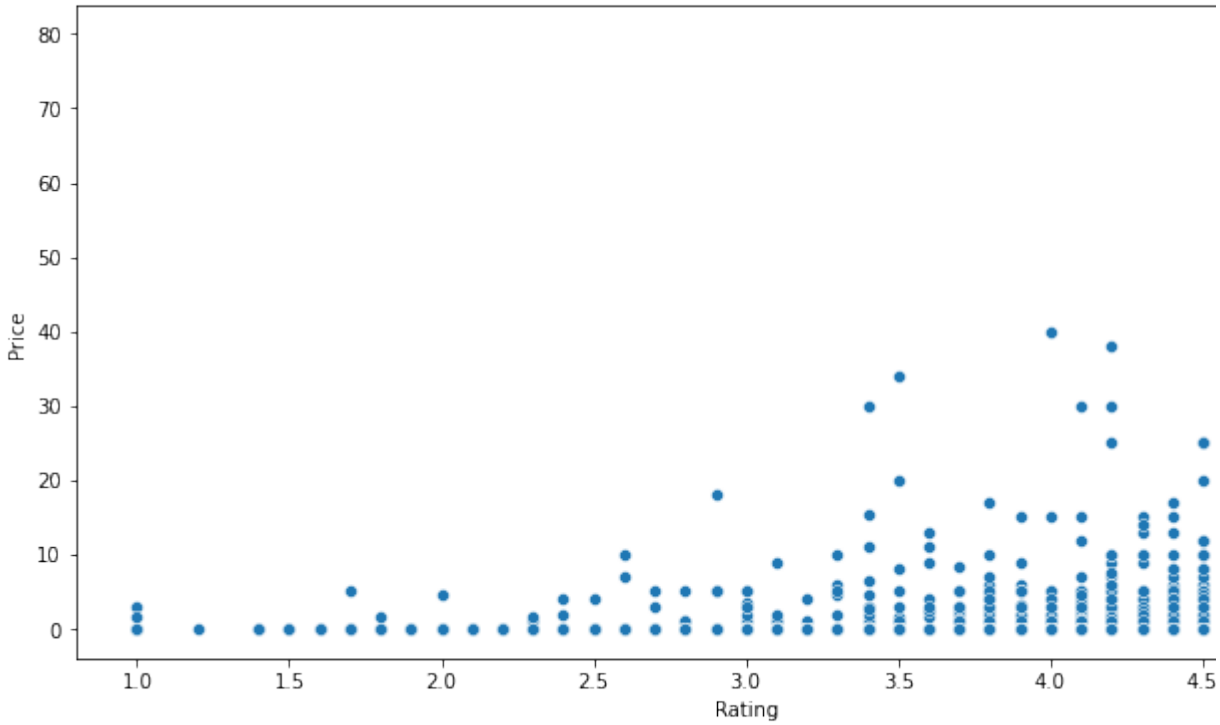
In [110]:
```
plt.figure(figsize=(12,10))
sns.barplot(x= x1_axis, y= y1_axis)
plt.title('Content Rating',size = 20);
plt.ylabel('Apps(Count)');
plt.xlabel('Content Rating');
```



In [127]:
```
## Scatter plot for rating Vs Price
```

In [128]:
```
plt.figure(figsize=(12,6))
sns.scatterplot(x='Rating',y='Price',data=df)
```

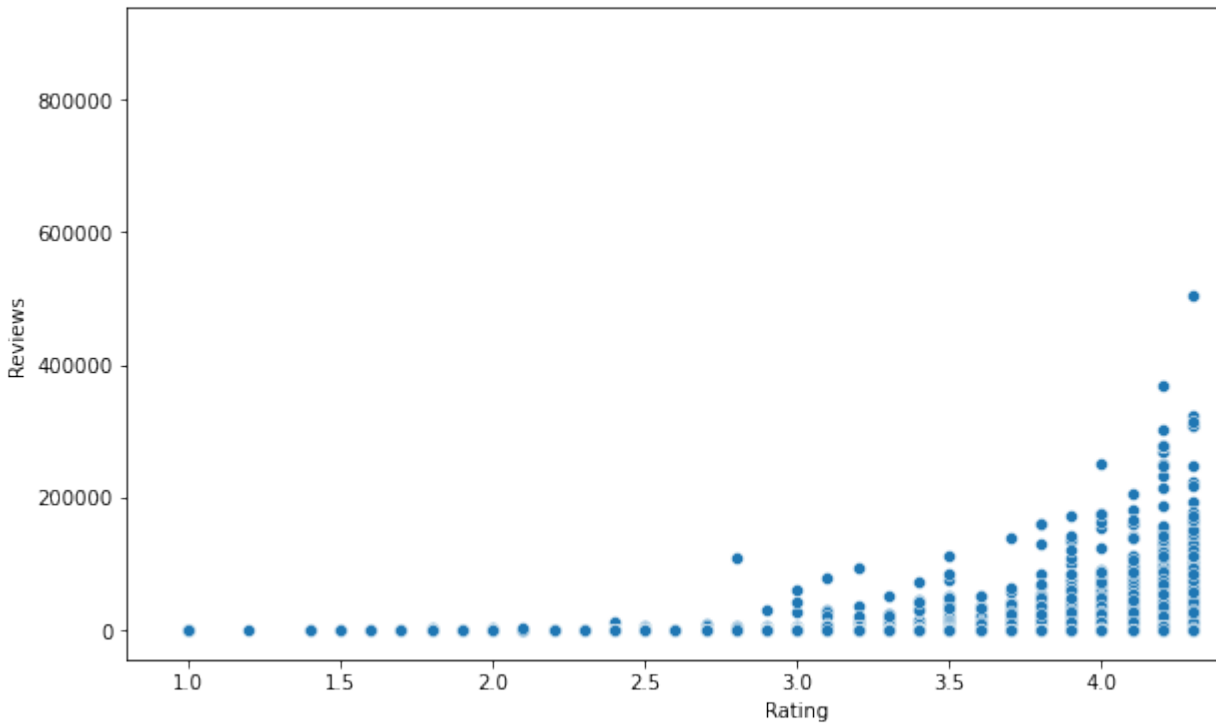
Out[128]:
```
<AxesSubplot:xlabel='Rating', ylabel='Price'>
```



In [130]:
```
## Scatter plot for rating vs Reviews
```

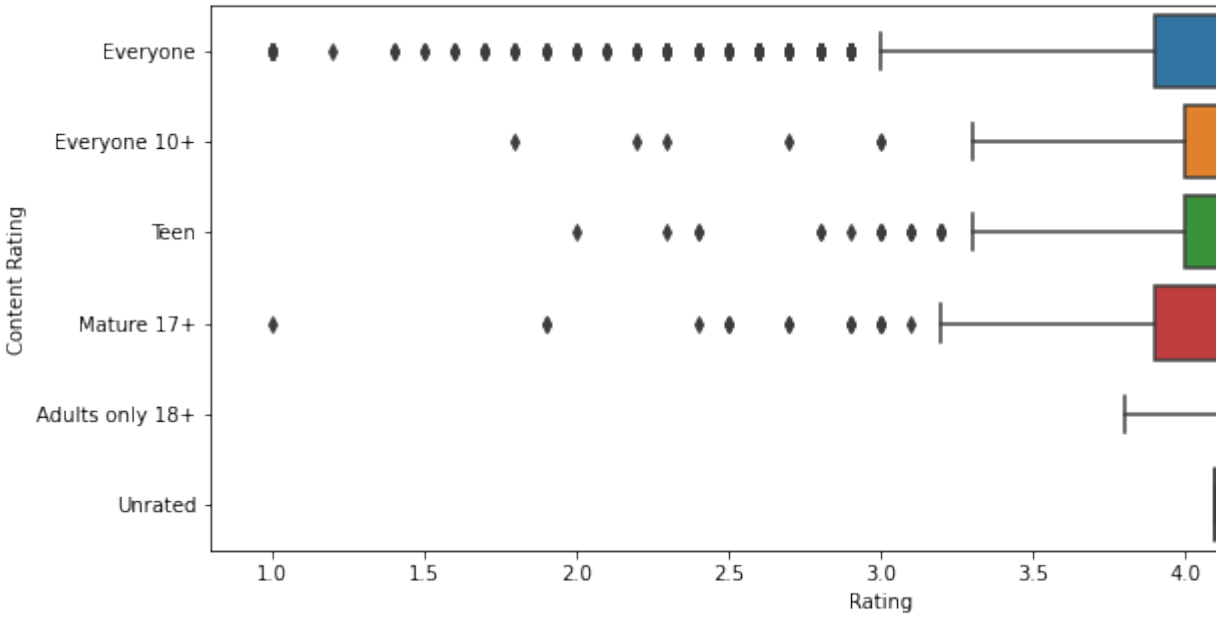
In [131]:
```
plt.figure(figsize=(12,6))
sns.scatterplot(x='Rating',y='Reviews',data=df)
```

Out[131]:
```
<AxesSubplot:xlabel='Rating', ylabel='Reviews'>
```



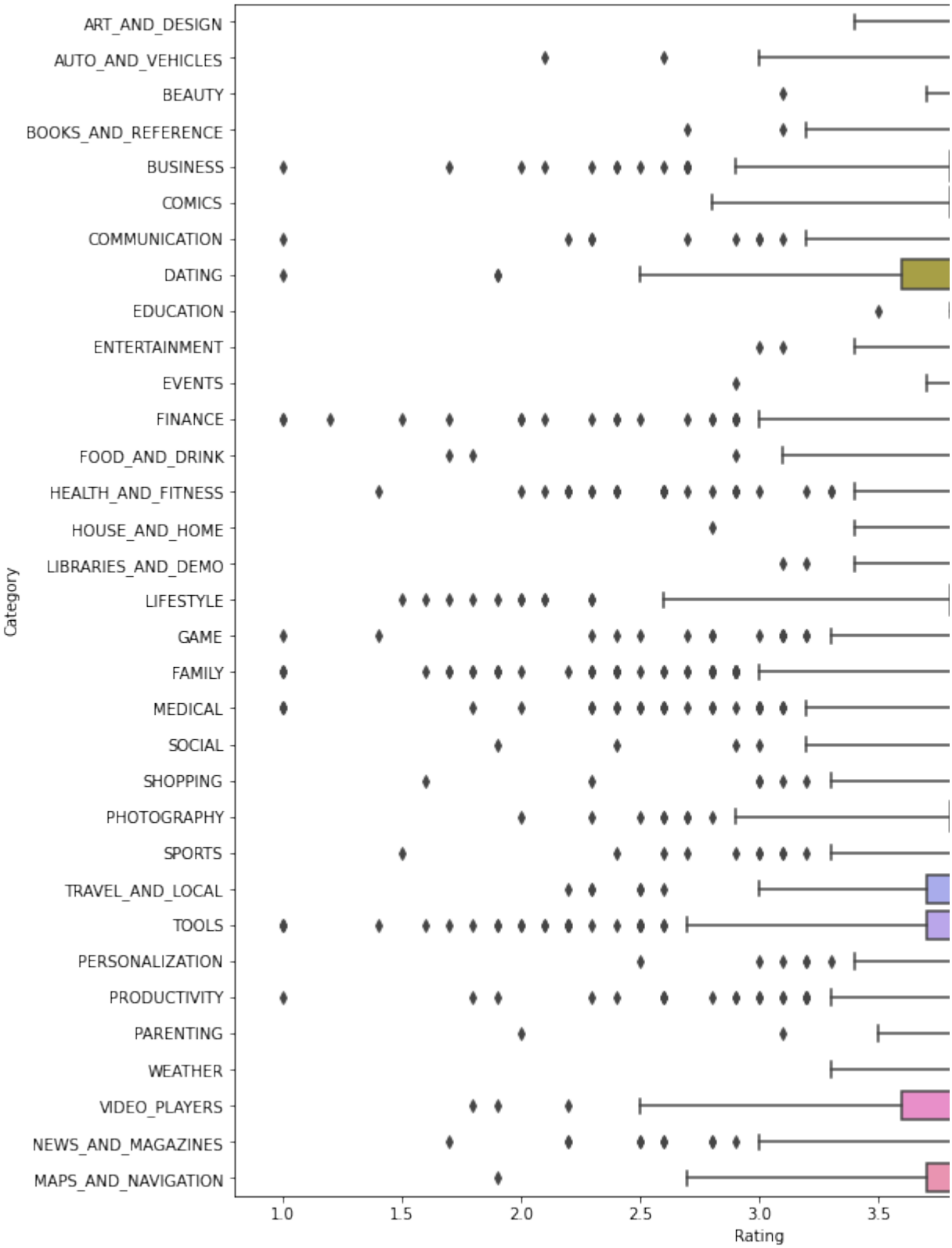
In [134]:
```
plt.figure(figsize=[12,5])
sns.boxplot("Rating", "Content Rating", data=df )
```

Out[134]:
```
<AxesSubplot:xlabel='Rating', ylabel='Content Rating'>
```



In [135]:
```
plt.figure(figsize=[12,14])
sns.boxplot("Rating", "Category", data=df )
```

Out[135]:
```
<AxesSubplot:xlabel='Rating', ylabel='Category'>
```



In …
```
# Make boxplot for Ratings vs. Category
# Which genre has the best ratings?
# Here Q2 (Median) is higher in 'BOOKS_AND_REFERENCES' and 'EVENTS' an
```

In […
```
# For the steps below, create a copy of the dataframe to make all the
```

In [157]:
```
inp1 =df.copy()
```

In [158]:
```
inp1
```

Out[158]:

	App	Category	Rating	Reviews	Size	Installs	Ty
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19000.0	10000.0	Fr
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14000.0	500000.0	Fr
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8700.0	5000000.0	Fr
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2800.0	100000.0	Fr
5	Paper flowers instructions	ART_AND_DESIGN	4.4	167	5600.0	50000.0	Fr
...	...	...	...	...	...	...	
10833	Chemin (fr)	BOOKS_AND_REFERENCE	4.8	44	619.0	1000.0	Fr
10834	FR Calculator	FAMILY	4.0	7	2600.0	500.0	Fr
10836	Sya9a Maroc - FR	FAMILY	4.5	38	53000.0	5000.0	Fr
10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3600.0	100.0	Fr
10839	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5	114	0.0	1000.0	Fr

6981 rows × 13 columns

In [159]:
```
df['Reviews'].describe()
```

Out[159]:
```
count      6981.000000
mean      18564.907606
std       47341.662556
min           1.000000
25%          78.000000
50%        1213.000000
75%       15192.000000
max      896118.000000
Name: Reviews, dtype: float64
```

In [160]:
```
## apply log transformation (np.Log1p) on Reviwes and Installs
```

In [161]:
```
inp1['Reviews']=np.log1p(inp1['Reviews'])
inp1['Installs']=np.log1p(inp1['Installs'])
```

In [162]:
```
inp1.head(1)
```

Out[162]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Co R
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	5.075174	19000.0	9.21044	Free	0.0	Ever

In [163]:
```
## droping the columns which are not usefull for further working
```

In [16…
```
inp1.drop(['App','Last Updated','Current Ver','Android Ver'], axis=1
inp1.head(2)
```

Out[164]:

	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	
0	ART_AND_DESIGN	4.1	5.075174	19000.0	9.210440	Free	0.0	Everyone	
1	ART_AND_DESIGN	3.9	6.875232	14000.0	13.122365	Free	0.0	Everyone	Des

In [16…
```
# Get dummy columns for Category, Genres, and Content Rating.

inp2=pd.get_dummies(inp1,columns=['Content Rating','Genres','Catego
inp2.head(2)
```

Out[165]:

	Rating	Reviews	Size	Installs	Price	Content Rating_Adults only 18+	Content Rating_Everyone	Rating
0	4.1	5.075174	19000.0	9.210440	0.0	0	1	
1	3.9	6.875232	14000.0	13.122365	0.0	0	1	

2 rows × 156 columns

In [166]:
```
x=inp2.drop('Rating',axis=1)   # independent Variable
y=inp2['Rating']                # Dependent Variable
```

In [167]:
```
from sklearn.model_selection import train_test_split
```

In [1…
```
# Train test split  and apply 70-30 split. Name the new dataframes d
# Separate the dataframes into X_train, y_train, X_test, and y_test.


x_train, x_test, y_train, y_test= train_test_split(x, y, test_size=0


from sklearn.linear_model import LinearRegression as LR
```

In [169]:
```
x_train.head(1)
```

Out[169]:

	Reviews	Size	Installs	Price	Content Rating_Adults only 18+	Content Rating_Everyone	Co Rating_Eve
9588	7.955425	39000.0	13.122365	0.0	0	1	

1 rows × 155 columns

In [170]:
```
# Use linear regression as the technique

model=LR()
model.fit(x_train, y_train)
```

Out[170]:
```
LinearRegression()
```

In [171]:
```
# Report the R2 on the train set

model.score(x_train, y_train)
```

Out[171]:
```
0.15268919030909045
```

In [172]:
```
# Make predictions on test set and report R2.

model.score(x_test, y_test)
```

Out[172]:
```
0.11400450481740809
```

In [173]:
```
model.predict(x_test)
```

Out[173]:
```
array([3.74056292, 4.03368424, 4.14940299, ..., 4.21011289,
4.29769173,
       4.27070364])
```

In [ ]:

In [ ]:

In [ ]: