App Rating Prediction

Objective: - Make a model with other information ab

```
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    Con

                                                                                                            Ra

               Photo
             Editor &
               Candy
         0                ART_AND_DESI4G.N1 15919M 10,00F0r+ee 0Eve
            Camera &
              Grid &
           ScrapBook

            Coloring
         1                bookART_AND_DESI3G.N9 96714M 500,00F0r+ee 0Eve moana

                   U
            Launcher
              Lite –
         2                FREE LiveART_AND_DESI4G.N7875180.7M5,000,0F0r0e+e 0Eve Cool
           Themes, Hide ...

            Sketch -
         3                Draw &ART_AND_DESI4G.N521564245M50,000,F0r0e0e+ 0 Paint

          Pixel Draw
            - Number
         4                ArtART_AND_DESI4G.N3 9672.8M100,00F0r+ee 0Eve
            Coloring
                Book
```

```
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 [85]: # # 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      C

                        Photo
                      Editor &
                        Candy

         0                                                    ART_AND_DESI4G.N1 15919000.010,00F0r+ee 0E

                     Camera &
                        Grid &
                    ScrapBook
```

1 Coloring bookART_AND_DESI3G.N9 96714000.0500,00F0r+ee 0E moana

U
Launcher
2 Lite –
FREE LiveART_AND_DESI4G.N7875108700.50,000,000+Free 0E Cool
Themes, Hide ...

Sketch -
3 Draw &ART_AND_DESI4G.N521564245000.050,000,F0r0e0e+ 0
Paint

4 Pixel Draw
- Number
ArtART_AND_DESI4G.N3 9672800.0100,00F0r+ee 0E
Coloring
Book

```
In
[[85]: 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 [[85]: 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 8886.0080.080806000e+038886.0080.080806000e+038886.00000

mean 4.187945.9730928e+19000.6515.961590061e+075 0.963526

std 0.522422.8906007e+23023.4188.668460413e+076 16.194792

min 1.000010.0000000e+000.000010.0000000e+000.000000

25% 4.000010.0640000e+022500.0010.000000000e+040.000000

```
               50%      4.300040.0723000e+039400.0050.000000000e+050.000000

               75%      4.500070.0131325e+27000.0050.000000000e+064                            0.000000

               max      5.000070.0815831e+07100000.010.0000000e+400.0000009
```

```
In [[85]: # 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   Ge Rating

                      KBA-
                        EZ
              2454                                                                MEDICAL5.0 425000.01.0Free0.0E0veryonMe
                     Health
                     Guide

                     Alarmy
                     (Sleep
                                                                        4663 If ULIFESTYLE4.810249 0.010000P.a0id2.4E9veryonLi

                     Can) -
                       Pro
```

```
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                              Cont
                                                                                     Installs     Type     Price      Rat
                     Photo
                    Editor &
                     Candy
             0                                                           ART_AND_DESI4G.N1 159190001.00000.0Free0.0Eve
                   Camera &
                     Grid &
                  ScrapBook
```

```
In [9[85]: # For free apps (type = "Free"), the price should not be >0.

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

Out[98]:

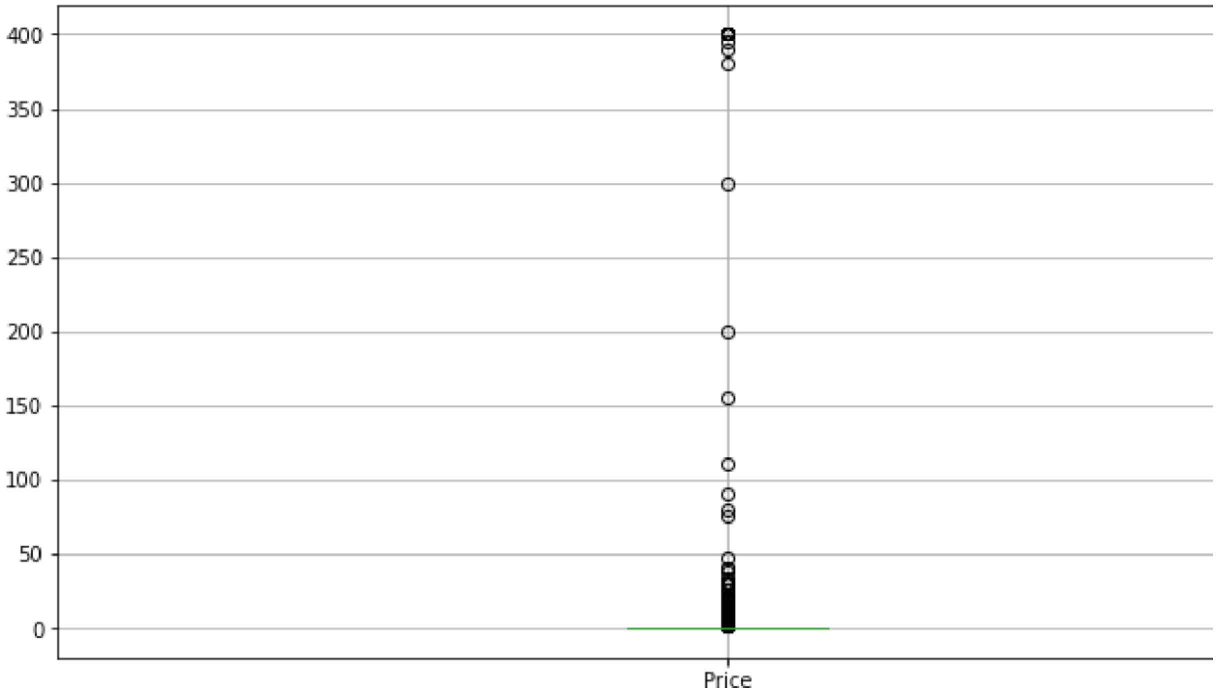
App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Las Updated

```
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')
```

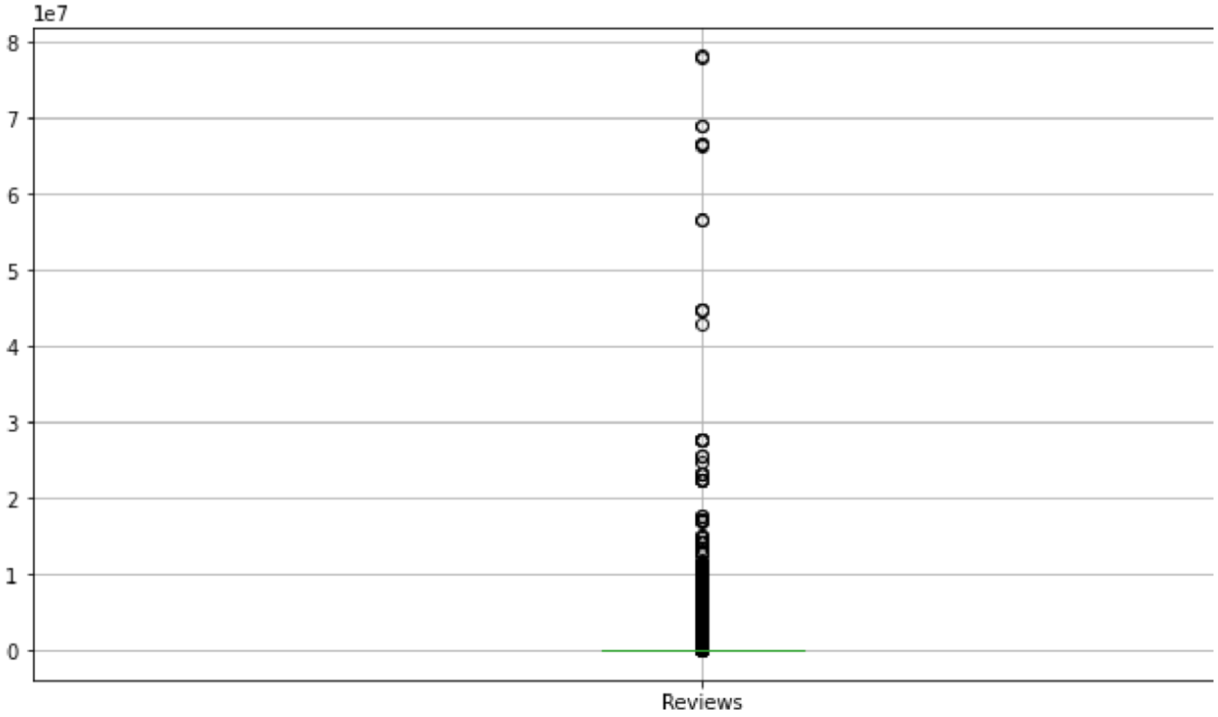
```
Out[34]: <AxesSubplot:>
```



```
In [[85]: # outlier are present in dataset , anything above than 300 will
be co
```

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

```
Out[37]: <AxesSubplot:>
```

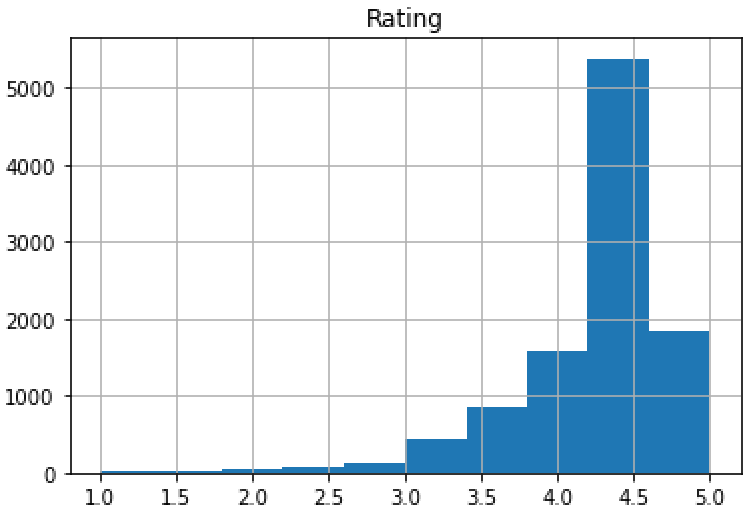


```
In [ [85]: # values above than 3 to le7 are the outliers in box plot shown
abov
```

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

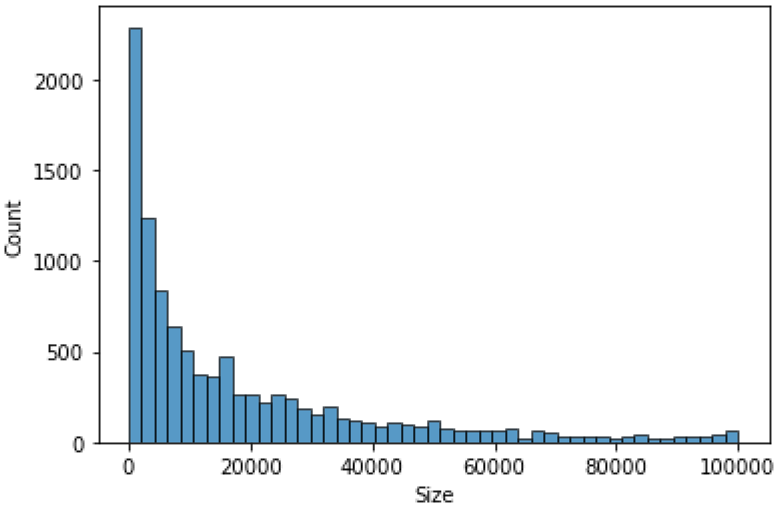
```
Out[43]:    array([[<AxesSubplot:title={'center':'Rating'}>]],    dtype=object)
```

```
 <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[85]: # From the box plot, it seems like there are some apps with very
           high # A price of $200 for an application on the Play Store is very
           high a # 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
                       expensiveFAMILY4.3 6 1500.0100.P0aid399.9E9ve
                          app (H)

         4362           I'LmI FrEiScThYLE3.8 71826000.100000.0Paid399.9E9ve

                          I'm Rich -
         4367
                              TrumpLIFESTYLE3.6 2757300.100000.0Paid400.0E0ve Edition

         5351           I am LrliFcEhSTYLE3.8 35471800.100000.0Paid399.9E9ve

                          I am Rich
         5354
                                              FAMILY4.0 8568700.100000.0Paid399.9E9ve Plus

         5355       I am riLcIhF EVSITPYLE3.8 4112600.100000.0Paid299.9E9ve

                          I Am Rich
         5356
                                            FINANCE4.1 18674700.500000.0Paid399.9E9ve Premium

                               I am
         5357                                                               412900.01000.P0aid379.9E9ve
                       extremelyLIFESTYLE2.9 Rich

         5358          I am RiFcIhN!ANCE3.8                                   9322000.10000.0Paid399.9E9ve

                               I am                                         472 965.05000.P0aid399.9E9ve
         5359     FINANCE3.5 rich(premium)

         5362 I Am RichF APMrIoLY4.4                                        2012700.05000.P0aid399.9E9ve
```

```
                I am rich
                    (Most
5364                                          FINANCE4.11292700.01000.P0aid399.99T
             expensive app)

5366 I Am RicFhAMILY3.6                                2174900.100000.0Paid389.9E9ve

5369 I am RiFcIhNANCE4.3                               1803800.05000.P0aid399.9E9ve

             I AM RICH
5373                                          FINANCE4.0  3641000.10000.0Paid399.9E9ve
           PRO PLUS

9917 Eu Sou RFiIcNoANCE4.4                                0 1400.0 0.0Paid394.9E9ve

           I'm Rich/Eu
9934 sou Rico/LIFESTYLE4.4 b'ᓚ·8
                    /≶ᓗ·ᓀ                                 040000.00.0Paid399.9E9ve
```

```
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 [85]: # 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 R

          Photo
        Editor &
          Candy
     0                   ART_AND_DESI4G.N1 15919000.100000.0Free0.0Ev
       Camera &
          Grid &
      ScrapBook

        Coloring
     1                   bookART_AND_DESI3G.N9 967140005.000000.0Free0.0Ev moana
```

```
In [1[85]: # 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 [[85]: # Keeping 95% value as a threshold/cutoff for outlier and drop
record

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

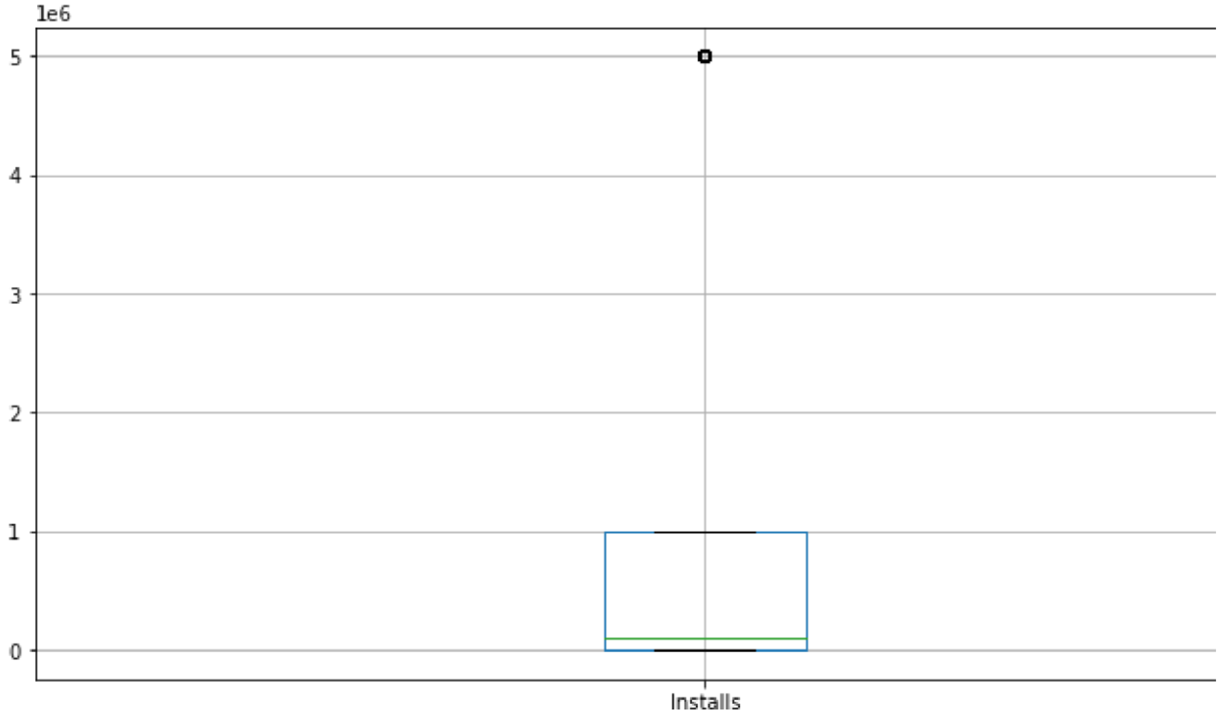
```
Out[103]:                App            Category   Rating   Reviews     Size    Installs   Type   Price    Co

                                                                                                            R

                        Photo                                  15919000.100000.0Free0.0Ev
                      Editor &
                        Candy
          0                              ART_AND_DESI4G.N1
                    Camera &
                        Grid &
                    ScrapBook
                      Coloring                                 967140005.000000.0Free0.0Ev
          1           bookART_AND_DESI3G.N9 moana
```

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

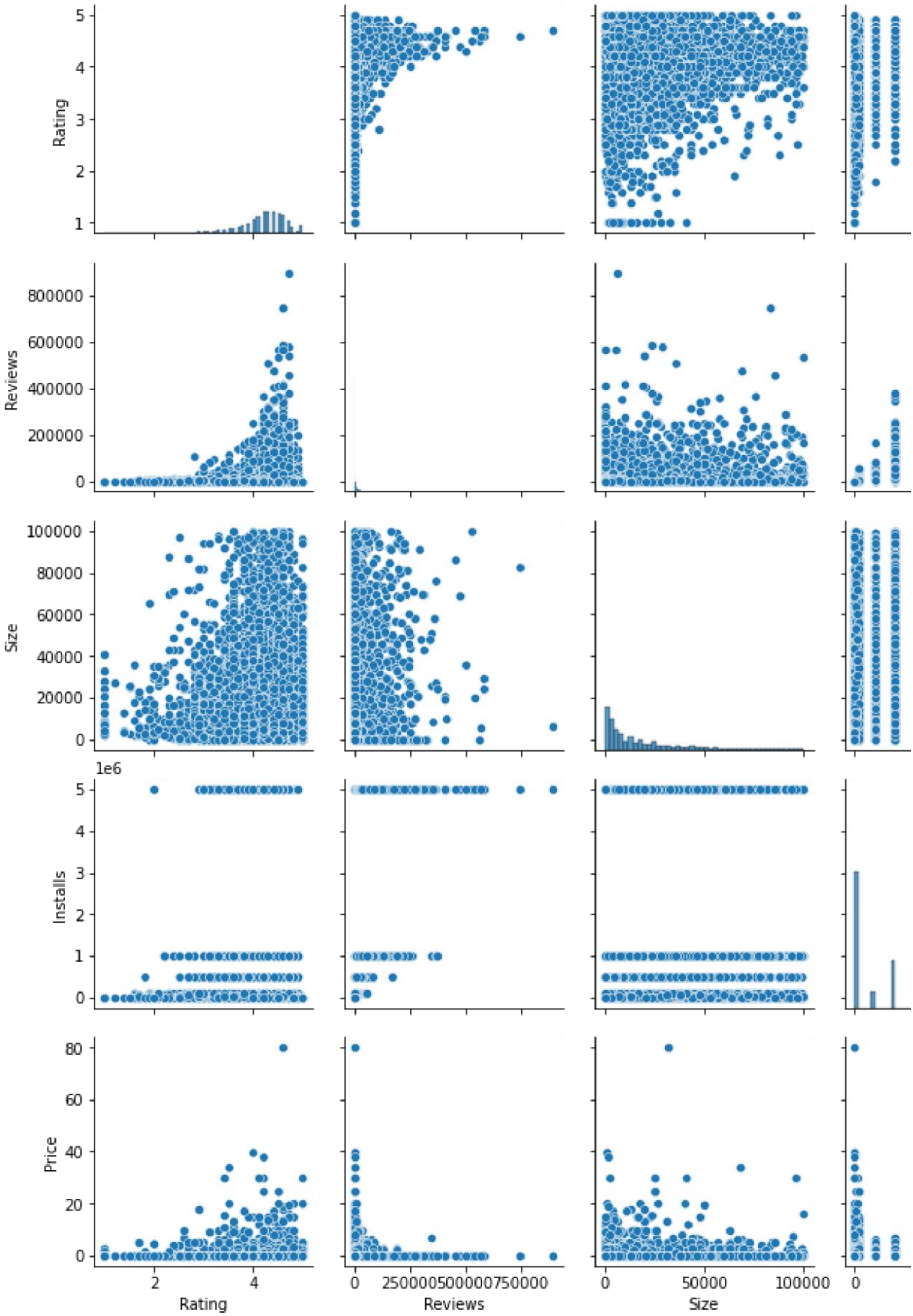
```
Out[104]: (6981, 13)
```

```
In [[85]: # 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
         Rati

         # 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
         ap

         # 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
         mor # 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>
```

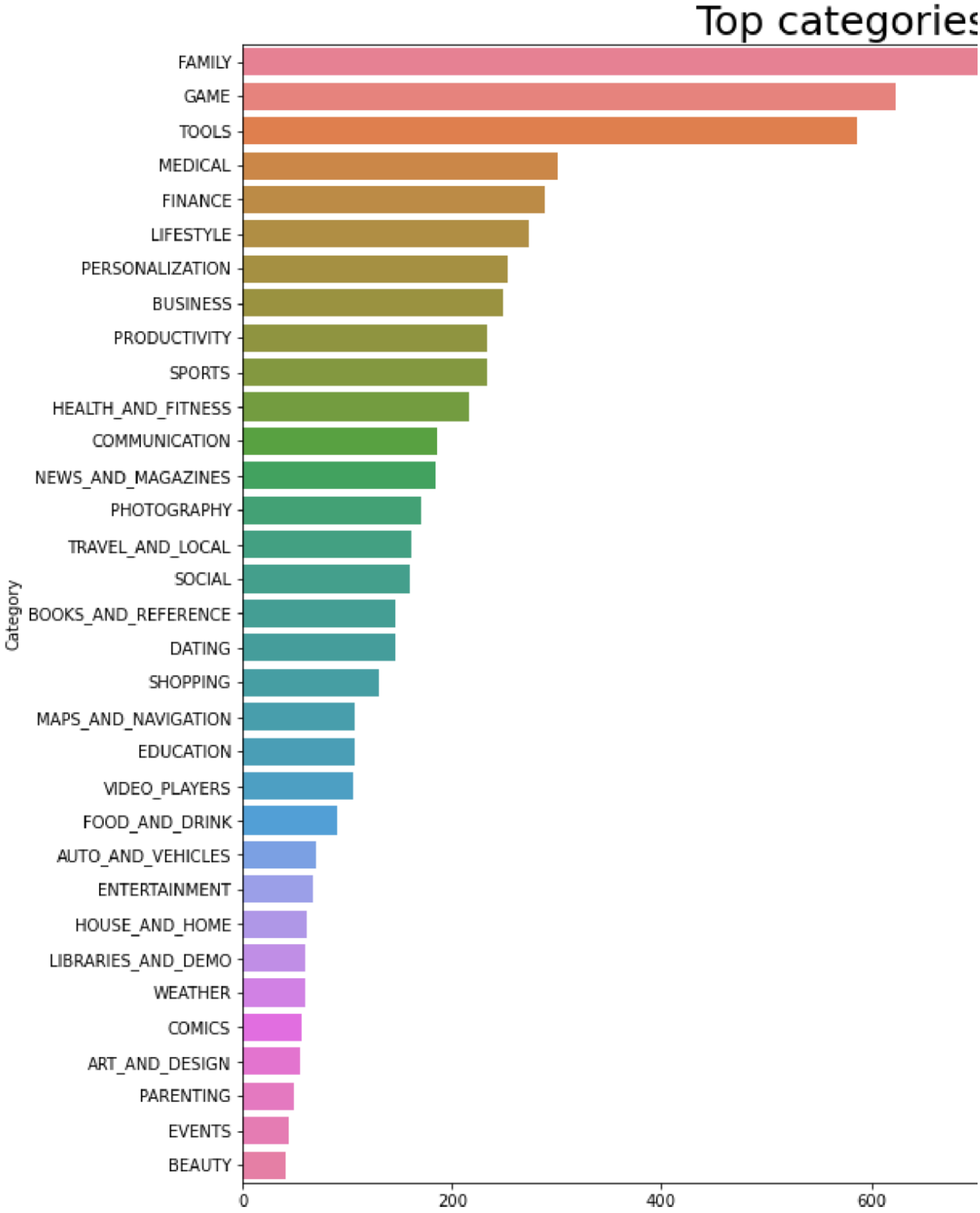


```
In [106]: ##  value count of top most app on google
```

```
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])
```

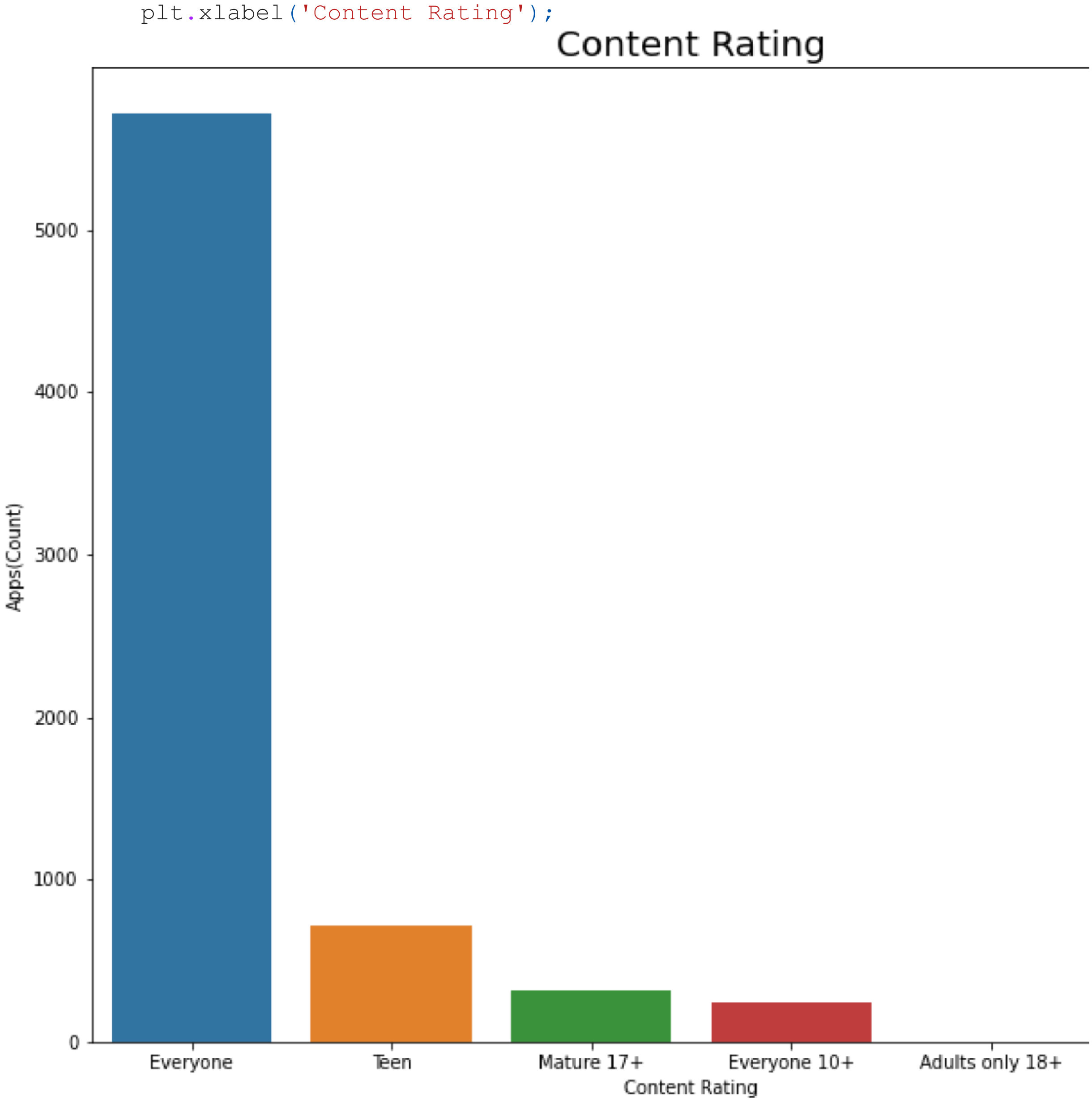
In
[107[85]:

```
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
```

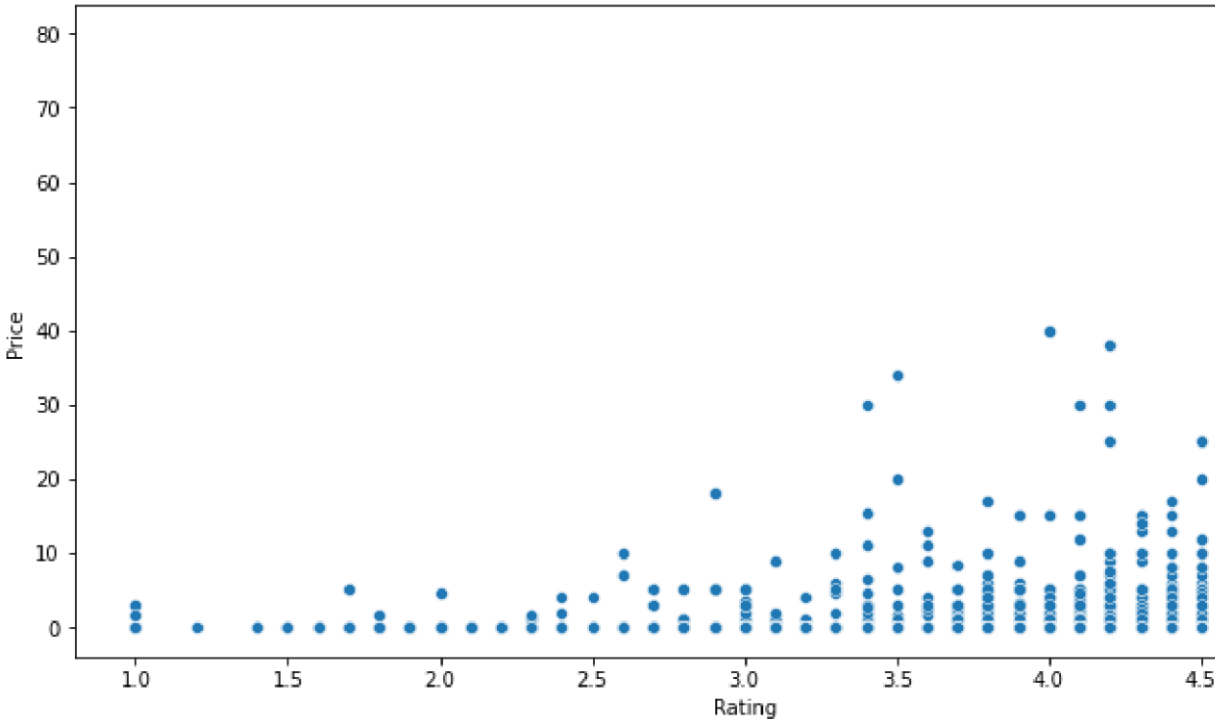

Top categories
Category
FAMILY
GAME
TOOLS
MEDICAL
FINANCE
LIFESTYLE
PERSONALIZATION
BUSINESS
PRODUCTIVITY
SPORTS
HEALTH_AND_FITNESS
COMMUNICATION
NEWS_AND_MAGAZINES
PHOTOGRAPHY
TRAVEL_AND_LOCAL
SOCIAL
BOOKS_AND_REFERENCE
DATING
SHOPPING
MAPS_AND_NAVIGATION
EDUCATION
VIDEO_PLAYERS
FOOD_AND_DRINK
AUTO_AND_VEHICLES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

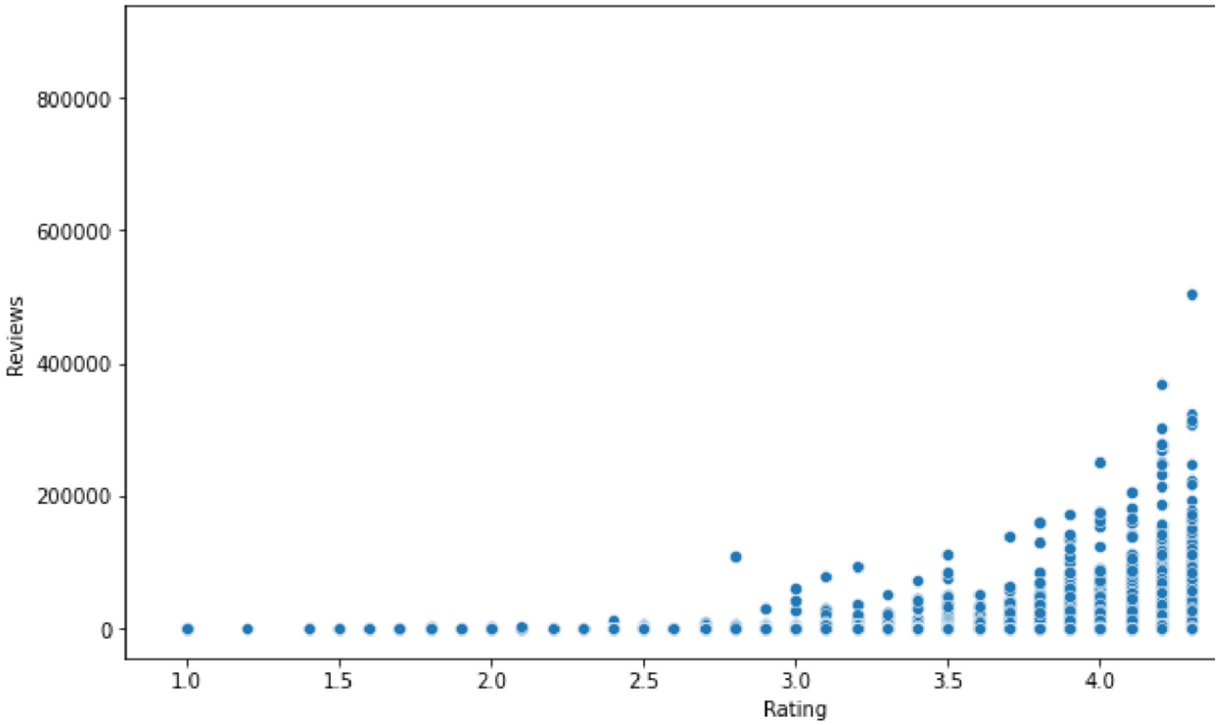
```
In      [110]:      plt.figure(figsize=(12,10))
          sns.barplot(x=  x1_axis,  y=  y1_axis)
          plt.title('Content   Rating',size   =
          20);        plt.ylabel('Apps(Count)');
          plt.xlabel('Content Rating');
```



```
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'>
```

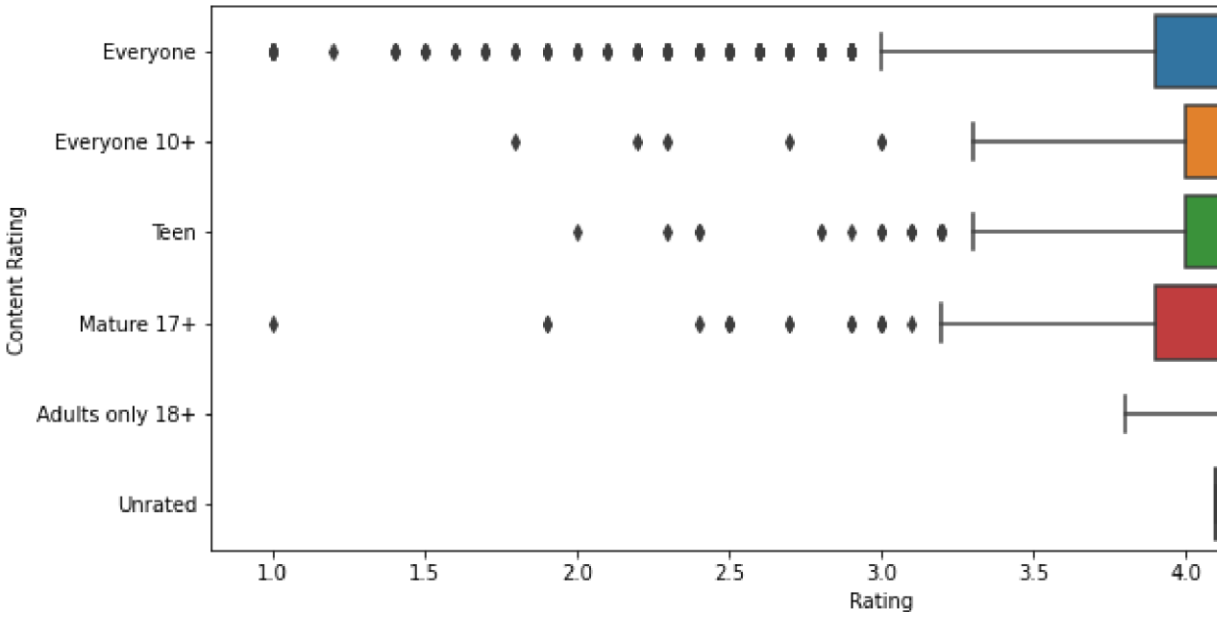


```
In [130]: ## Scatter plot for rating vs Reviews

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

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

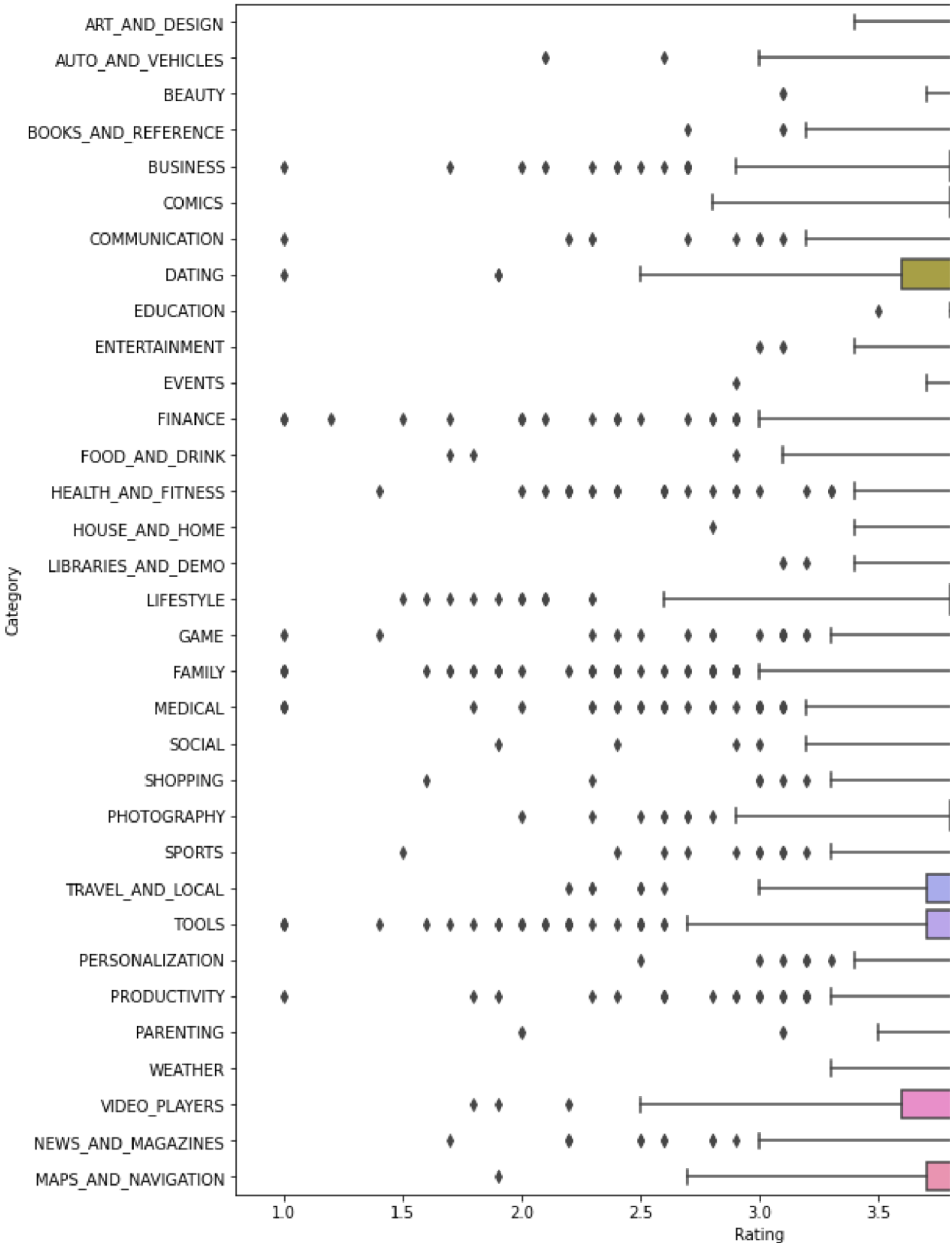


```
In [134]: plt.figure(figsize=[12,5]) sns.boxplot("Rating",
          "Content Rating", data=df )
```

```
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 [85]: # 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 [[85]: # 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     Typ

                           Photo
                        Editor &
                           Candy
              0                         ART_AND_DESI4G.N1 15919000.100000.Fr
                        Camera &
                          Grid &
                       ScrapBook

                        Coloring
              1                         book      ART_AND_DESI3G.N9 96714000.5000000.Fr moana

                               U
                        Launcher
                          Lite –
              2                         FREE LiveART_AND_DESI4G.N7875108700.5000000.Fr
                            Cool
                       Themes, Hide ...

                      Pixel Draw
                        - Number
              4                  Art                      ART_AND_DESI4G.N39672800.1000000.Fr
                        Coloring
                            Book

                           Paper
              5 flowers ART_AND_DESI4G.N4 instructions                   1675600.050000F.r

             ...                 ...                                       ... ...   ... ... ...

          10833 Chemin B(OfOrK)S_AND_REFERENCE4.8                          44 619.01000.F0r

                              FR
          10834                                              FAMILY4.0    7 2600.0500.F0r
                      Calculator
                           Sya9a
          10836                                              FAMILY4.5  3853000.05000.F0r
                      Maroc - FR
                        Fr. Mike
                         Schmitz
          10837                                              FAMILY5.0    4 3600.0100.F0r
                           Audio
                       Teachings
```

The SCP 114 0.0 1000.F0r
10839 FoundationBOOKS_AND_REFERE4N.C5E DB fr nn5n

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 Con R

Photo
Editor &
Candy
0 ART_AND_DESI4G.N15.07511794000.09.210F4r4ee0.0Ev
Camera &
Grid &
ScrapBook

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

In [16[85]: 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_DESI4G.N15.07511794000.09.2104F4r0ee0.0EveryonA

1 ART_AND_DESI3G.N96.87521342000.013.122F3r6e5e0.0EveryonDe

```
In [16[85]: # Get dummy columns for Category, Genres, and Content Rating.

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

Out[165]:
```

Content Content

```
                    Rating      Reviews        Size        Installs     Price     Rating_Adults                              Rating
                                                                                             Rating_Everyone only
                                                                                  18+
                          0 4.15.07511794000.09.2104400.0                                        0                   1
       1         3.96.87521342000.013.122306.50          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[85]: # 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]:                                                          Content              Content                 Co
                      Reviews        Size        Installs    Price    Rating_Adults                              Rating_Eve
                                                                                    Rating_Everyone only
                                                                         18+

                      9588 7.95543295000.013.122306.50                          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 [ ]:
```