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

Problem Statement:

Google Play Store team is about to launch a new feature wherein, certain apps that are promising, are boosted in visibility. The boost will manifest in multiple ways including higher priority in recommendations sections ("Similar apps", "You might also like", "New and updated games"). These will also get a boost in search results visibility.
This feature will help bring more attention to newer apps that have the potential.

Domain: General

Analysis to be done: The problem is to identify the apps that are going to be good for Google to promote. App ratings, which are provided by the customers, is always a great indicator of the goodness of the app. The problem reduces to: predict which apps will have high ratings.

Content: Dataset: Google Play Store data ("googleplaystore.csv")

```
pip install seaborn
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: seaborn in c:\programdata\anaconda3\lib\site-
packages (0.11.2)
Requirement already satisfied: scipy>=1.0 in
c:\programdata\anaconda3\lib\site-packages (from seaborn) (1.7.3)
Requirement already satisfied: numpy>=1.15 in
c:\programdata\anaconda3\lib\site-packages (from seaborn) (1.21.5)
Requirement already satisfied: matplotlib>=2.2 in
c:\programdata\anaconda3\lib\site-packages (from seaborn) (3.5.1)
Requirement already satisfied: pandas>=0.23 in
c:\programdata\anaconda3\lib\site-packages (from seaborn) (1.4.2)
Requirement already satisfied: cycler>=0.10 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn)
(0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn)
(1.3.2)
Requirement already satisfied: pyparsing>=2.2.1 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn)
(3.0.4)
Requirement already satisfied: fonttools>=4.22.0 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn)
(4.25.0)
Requirement already satisfied: packaging>=20.0 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn)
(21.3)
Requirement already satisfied: pillow>=6.2.0 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn)
(9.0.1)
Requirement already satisfied: python-dateutil>=2.7 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in
c:\programdata\anaconda3\lib\site-packages (from pandas>=0.23->seaborn)
(2021.3)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-
packages (from python-dateutil>=2.7->matplotlib>=2.2->seaborn) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

```
pip install sklearn
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting sklearn
  Downloading sklearn-0.0.tar.gz (1.1 kB)
Requirement already satisfied: scikit-learn in
c:\programdata\anaconda3\lib\site-packages (from sklearn) (1.0.2)
Requirement already satisfied: scipy>=1.1.0 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn->sklearn)
```

```
(1.7.3)
Requirement already satisfied: joblib>=0.11 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn->sklearn)
(1.1.0)
Requirement already satisfied: numpy>=1.14.6 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn->sklearn)
(1.21.5)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn->sklearn)
(2.2.0)
Building wheels for collected packages: sklearn
  Building wheel for sklearn (setup.py): started
  Building wheel for sklearn (setup.py): finished with status 'done'
  Created wheel for sklearn: filename=sklearn-0.0-py2.py3-none-any.whl
size=1310
sha256=68ab56c49595bd8a4e5f5dd1e0da972ede244c6884aa46b218264b9f07d5f83b
  Stored in directory:
c:\users\shiwa\appdata\local\pip\cache\wheels\e4\7b\98\b6466d71b8d738a0c54700
8b9eb39bf8676d1ff6ca4b22af1c
Successfully built sklearn
Installing collected packages: sklearn
Successfully installed sklearn-0.0
Note: you may need to restart the kernel to use updated packages.
```

```
pip install matplotlib
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: matplotlib in
c:\programdata\anaconda3\lib\site-packages (3.5.1)
Requirement already satisfied: packaging>=20.0 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (21.3)
Requirement already satisfied: numpy>=1.17 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (1.21.5)
Requirement already satisfied: pillow>=6.2.0 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (9.0.1)
Requirement already satisfied: python-dateutil>=2.7 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: cycler>=0.10 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (1.3.2)
Requirement already satisfied: fonttools>=4.22.0 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: pyparsing>=2.2.1 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (3.0.4)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-
packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

-
  ```
  import numpy as np
  import pandas as pd
  import matplotlib.pyplot as plt
  import seaborn as sns
  import statistics as stc
  from sklearn.model_selection import train_test_split
  from sklearn.linear_model import LinearRegression
  from sklearn.metrics import r2_score
  from warnings import filterwarnings
  filterwarnings('ignore')
  plt.rcParams['figure.figsize'] = [15 , 8]
  ```

Q1. Load the data file using pandas.

-
  ```
  df = pd.read_csv (r'C:\simplilearn\python project
  simplilearn\googleplaystore.csv')
  ```

-
  ```
  df.head()
  ```

```
                                                 App        Category  Rating  
\
0     Photo Editor & Candy Camera & Grid & ScrapBook  ART_AND_DESIGN     4.1   
1                                Coloring book moana  ART_AND_DESIGN     3.9   
2  U Launcher Lite – FREE Live Cool Themes, Hide ...  ART_AND_DESIGN     4.7   
3                              Sketch - Draw & Paint  ART_AND_DESIGN     4.5   
4              Pixel Draw - Number Art Coloring Book  ART_AND_DESIGN     4.3   

  Reviews  Size     Installs  Type Price Content Rating  \
0     159   19M      10,000+  Free     0       Everyone   
1     967   14M     500,000+  Free     0       Everyone   
2   87510  8.7M   5,000,000+  Free     0       Everyone   
3  215644   25M  50,000,000+  Free     0           Teen   
4     967  2.8M     100,000+  Free     0       Everyone   

                     Genres      Last Updated         Current Ver  \
0              Art & Design   January 7, 2018               1.0.0   
1  Art & Design;Pretend Play  January 15, 2018               2.0.0   
2              Art & Design    August 1, 2018               1.2.4   
3              Art & Design      June 8, 2018  Varies with device   
4    Art & Design;Creativity     June 20, 2018                 1.1   

    Android Ver  
0  4.0.3 and up  
1  4.0.3 and up  
2  4.0.3 and up  
3    4.2 and up  
4    4.4 and up  
```

1) Data inspection

- `df.dtypes`

```
App                object
Category           object
Rating            float64
Reviews            object
Size               object
Installs           object
Type               object
Price              object
Content Rating     object
Genres             object
Last Updated       object
Current Ver        object
Android Ver        object
dtype: object
```

- `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
```

- `df.shape`

```
(10841, 13)
```

- `df.describe()`

```
            Rating
count  9367.000000
mean      4.193338
std       0.537431
min       1.000000
25%       4.000000
50%       4.300000
75%       4.500000
max      19.000000
```

- `df.describe(include = object)`

```
           App Category Reviews                Size    Installs   Type  Price
\
count    10841    10841   10841               10841       10841  10840  10841
unique    9660       34    6002                 462          22      3     93
top     ROBLOX   FAMILY       0  Varies with device  1,000,000+   Free      0
freq         9     1972     596                1695        1579  10039  10040

       Content Rating Genres    Last Updated         Current Ver Android Ver
count           10840  10841           10841               10833       10838
unique              6    120            1378                2832          33
top          Everyone  Tools  August 3, 2018  Varies with device  4.1 and up
freq             8714    842             326                1459        2451
```

Q2. Check for null values in the data. Get the number of null values for each column.

- `df.isnull().sum()*100/df.shape[0]`

```
App                0.000000
Category           0.000000
Rating            13.596532
Reviews            0.000000
Size               0.000000
Installs           0.000000
Type               0.009224
Price              0.000000
Content Rating     0.009224
Genres             0.000000
Last Updated       0.000000
Current Ver        0.073794
Android Ver        0.027673
dtype: float64
```

Interpretation: There are null values present in the data. those are Rating = 13.596532, Type = 0.009224, Content Rating = 0.009224, Current Ver = 0.073794, Android Ver = 0.027673,

Q3. Drop records with nulls in any of the columns.

-
  ```
  print("Frame Size before : " , df.shape)
  df.dropna(subset=['Rating', 'Type', 'Content Rating', 'Current Ver','Android Ver'],axis=0, inplace=True)
  print("Frame Size After : " , df.shape)
  df.isnull().sum(axis=0)
  ```

```
Frame Size before :  (10841, 13)
Frame Size After :  (9360, 13)
```

```
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
```

Q4.1 Variables seem to have incorrect type and inconsistent formatting. You need to fix them:

Q4.1. Size column has sizes in Kb as well as Mb. To analyze, you’ll need to convert these to numeric.

Q4.1.1. Extract the numeric value from the column

Q4.1.2. Multiply the value by 1,000, if size is mentioned in Mb

Q4.2. Reviews is a numeric field that is loaded as a string field. Convert it to numeric (int/float).

Q4.3.Installs field is currently stored as string and has values like 1,000,000+. . Q4.3.1. Treat 1,000,000+ as 1,000,000

Q4.3.2. remove ‘+’, ‘,’ from the field, convert it to integer

Q4.4. Price field is a string and has $ symbol. Remove ‘$’ sign, and convert it to numeric.

-
  ```
  df=df[-df['Size'].str.contains('Var')]
  df["Size"]
  ```

```
0         19M
1         14M
2        8.7M
3         25M
4        2.8M
         ...
10833    619k
10834    2.6M
10836     53M
10837    3.6M
10840     19M
Name: Size, Length: 7723, dtype: object
```

Q4.1.1. Extract the numeric value from the column

-
  ```
  df.loc[:,"SizeNum"] = df.Size.str.rstrip("Mk+")
  df["SizeNum"]
  ```

```
0         19
1         14
2        8.7
3         25
4        2.8
        ...
10833    619
10834    2.6
10836     53
10837    3.6
10840     19
Name: SizeNum, Length: 7723, dtype: object
```

-
  ```
  df.SizeNum = pd.to_numeric(df["SizeNum"])
  df.SizeNum.dtype
  ```

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

Q4.1.2 Multiply the value by 1,000, if size is mentioned in Mb

-
  ```
  df['SizeNum']=np.where(df.Size.str.contains('M'),df.SizeNum*1000,
  df.SizeNum)
  ```

-
  ```
  df.Size=df.SizeNum
  df.drop('SizeNum',axis=1,inplace=True)
  ```

Q4.2. Reviews is a numeric field that is loaded as a string field. Convert it to numeric (int/float).

- `df.Reviews = pd.to_numeric(df.Reviews)`
- `df.Reviews.dtype`
- `dtype('int64')`

Q4.3.Installs field is currently stored as string and has values like 1,000,000+. Q4.3.2. remove ‘+’, ‘,’ from the field, convert it to integer

-
  ```
  df['Installs']=df.Installs.str.replace("1000000+","1000000")
  df['Installs']=df.Installs.str.replace("+","")
  ```

-
  ```
  df.Installs=df.Installs.str.replace(",","")
  df.Installs=pd.to_numeric(df.Installs)
  df.Installs.dtype
  ```
- `dtype('int64')`
- `df.head()`

```
                                                 App        Category  Rating  
\
0     Photo Editor & Candy Camera & Grid & ScrapBook  ART_AND_DESIGN     4.1
1                                Coloring book moana  ART_AND_DESIGN     3.9
2  U Launcher Lite – FREE Live Cool Themes, Hide ...  ART_AND_DESIGN     4.7
3                              Sketch - Draw & Paint  ART_AND_DESIGN     4.5
4              Pixel Draw - Number Art Coloring Book  ART_AND_DESIGN     4.3

   Reviews     Size  Installs  Type Price Content Rating  \
0      159  19000.0     10000  Free     0       Everyone
1      967  14000.0    500000  Free     0       Everyone
2    87510   8700.0   5000000  Free     0       Everyone
3   215644  25000.0  50000000  Free     0           Teen
4      967   2800.0    100000  Free     0       Everyone

                     Genres      Last Updated         Current Ver  \
0              Art & Design   January 7, 2018               1.0.0
1  Art & Design;Pretend Play  January 15, 2018               2.0.0
2              Art & Design    August 1, 2018               1.2.4
3              Art & Design      June 8, 2018  Varies with device
4    Art & Design;Creativity     June 20, 2018                 1.1

    Android Ver
0  4.0.3 and up
1  4.0.3 and up
2  4.0.3 and up
```

```
3    4.2 and up
4    4.4 and up
```

Q4.4. Price field is a string and has $ symbol. Remove '$' sign, and convert it to numeric.

- `df.Price.value_counts()`

```
0          7146
$0.99       105
$2.99       101
$4.99        63
$1.99        53
           ... 
$6.49         1
$1.29         1
$299.99       1
$379.99       1
$1.20         1
Name: Price, Length: 68, dtype: int64
```

- `df['Price'] = df['Price'].str.replace('$', '')`

- `df.Price.value_counts()`

```
0         7146
0.99       105
2.99       101
4.99        63
1.99        53
          ... 
6.49         1
1.29         1
299.99       1
379.99       1
1.20         1
Name: Price, Length: 68, dtype: int64
```

- `df['Price'] = df['Price'].astype(float)`

5. Sanity checks:

```
     5.1. Average rating should be between 1 and 5 as only these values
are allowed on the play store.
     Drop the rows that have a value outside this range.

     5.2. Reviews should not be more than installs as only those who
installed can review the app.
     If there are any such records, drop them.

     5.3. For free apps (type = “Free”), the price should not be >0.
Drop any such rows.
```

5.1. Average rating should be between 1 and 5 as only these values are allowed on the play store. Drop the rows that have a value outside this range.

-
  ```
  df=df[(df.Rating>=1) & (df.Rating<=5) ]
  df.head()
  df.tail()
  ```

```
                                                 App             Category  \
10833                                    Chemin (fr)  BOOKS_AND_REFERENCE   
10834                                  FR Calculator               FAMILY   
10836                               Sya9a Maroc - FR               FAMILY   
10837               Fr. Mike Schmitz Audio Teachings               FAMILY   
10840  iHoroscope - 2018 Daily Horoscope & Astrology            LIFESTYLE   

       Rating  Reviews     Size  Installs  Type  Price Content Rating  \
10833     4.8       44    619.0      1000  Free    0.0       Everyone   
10834     4.0        7   2600.0       500  Free    0.0       Everyone   
10836     4.5       38  53000.0      5000  Free    0.0       Everyone   
10837     5.0        4   3600.0       100  Free    0.0       Everyone   
10840     4.5   398307  19000.0  10000000  Free    0.0       Everyone   

                  Genres    Last Updated         Current Ver  \
10833  Books & Reference  March 23, 2014                 0.8   
10834          Education   June 18, 2017               1.0.0   
10836          Education   July 25, 2017                1.48   
10837          Education    July 6, 2018                 1.0   
10840          Lifestyle   July 25, 2018  Varies with device   

              Android Ver  
10833          2.2 and up  
10834          4.1 and up  
10836          4.1 and up  
10837          4.1 and up  
10840  Varies with device  
```

5.2. Reviews should not be more than installs as only those who installed can review the app. If there are any such records, drop them.

-
  ```
  len(df.index)
  ```

7723

-
  ```
  df.drop(df.index[df.Reviews>df.Installs],axis=0,inplace=True)
  len(df.index)
  ```

7717

- **For free apps (type = “Free”), the price should not be >0. Drop any such rows.**

-
  ```
  index_free_and_price_gt_0=df.index[((df.Type=='Free')&(df.Price>0))]
  if len(index_free_and_price_gt_0)>0:
      print("Dropping following indices:",index_free_and_price_gt_0)
      df.drop(index_free_and_price_gt_0,axis=0,inplace=True)
  else:
      print("There is no Free Apps with price >0")
  ```

-
  ```
  There is no Free Apps with price >0
  ```

Interpretation: There is no Free Apps with price >0

5. Performing univariate analysis:

5.1. Boxplot for Price

```
 5.1.1 Are there any outliers? Think about the price of usual apps on
Play Store.
```

5.2. Boxplot for Reviews

```
 5.2.1 Are there any apps with very high number of reviews? Do the
values seem right?
```

5.3. Histogram for Rating

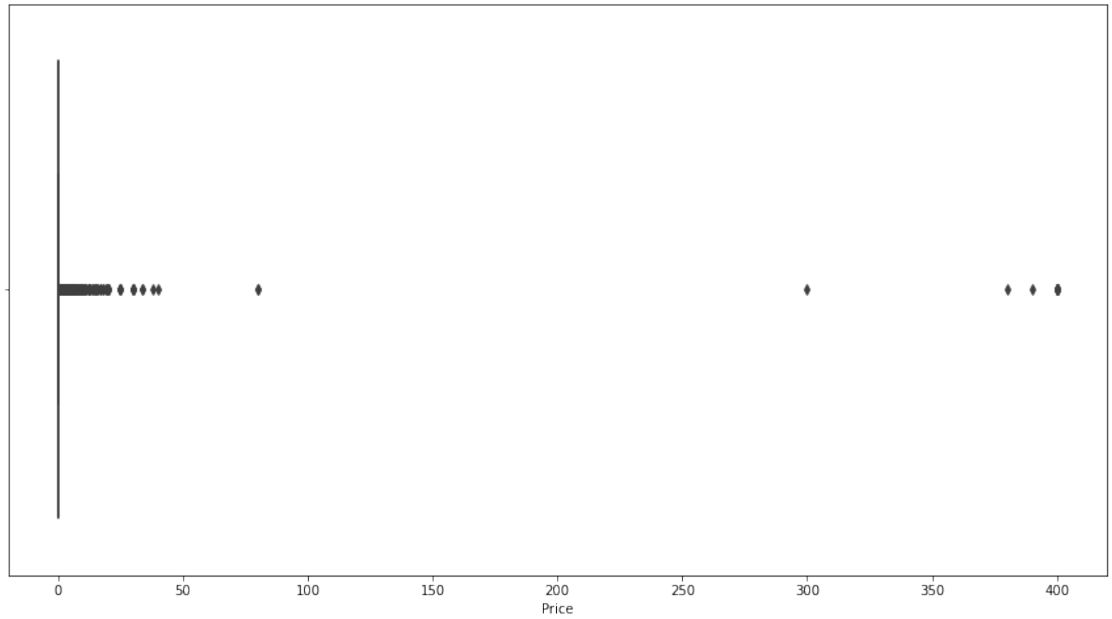
```
 5.3.1 How are the ratings distributed? Is it more toward higher
ratings?
```

5.4. Histogram for Size

```
 5.4.1 Note down your observations for the plots made above. Which of
these seem to have outliers?
```

5.1. Boxplot for Price

- `bprice = sns.boxplot(x='Price', data=df)`



Interpretation: Most of Price values are less than 50 while there is some near concentration around 80. greater than 100 may be considered outliers

-
  ```
  price_standard_deviation=stc.stdev(df.Price)
  price_standard_deviation
  ```

```
17.414783874309933
```

-
  ```
  price_mean=stc.mean(df.Price)
  price_mean
  ```

```
1.128724893093171
```

-
  ```
  price_outlier_uplimit=price_mean+3*price_standard_deviation
  price_outlier_uplimit
  ```

```
53.37307651602297
```

-
  ```
  print("Number of upper outliers is ",len(df[(df.Price>price_outlier_uplimit) ]))
  ```

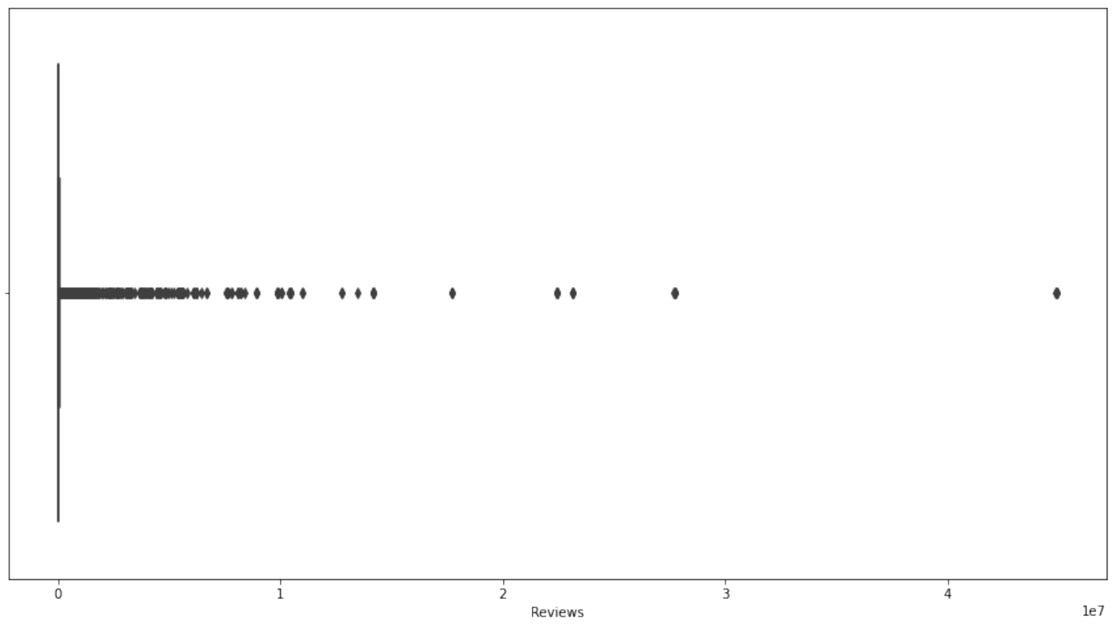
```
Number of upper outliers is  17
```

Interpretation: Number of upper outliers is 17

5.2. Boxplot for Reviews

```
sns.boxplot(x='Reviews',data=df)
```

```
<AxesSubplot:xlabel='Reviews'>
```



Interpretation: Most Apps get about less than 2M review. Roughly, greater than 2M can be considered outliers

-
    ```
    review_standard_deviation=stc.stdev(df.Reviews)
    review_standard_deviation
    ```

```
1864639.6094670836
```

-
    ```
    review_mean=stc.mean(df.Reviews)
    review_mean
    ```

```
295127.5482700531
```

-
    ```
    review_outlier_uplimit=review_mean+3*review_standard_deviation
    rev_outlier_uplimit
    ```

```
5889046.376671304
```

-
    ```
    review_outlier_downlimit=review_mean-3*review_standard_deviation
    review_outlier_downlimit
    ```

```
-5298791.280131198
```

-
    ```
    print("number of upper outliers is
    ",len(df[(df.Reviews>rev_outlier_uplimit) ]))
    ```

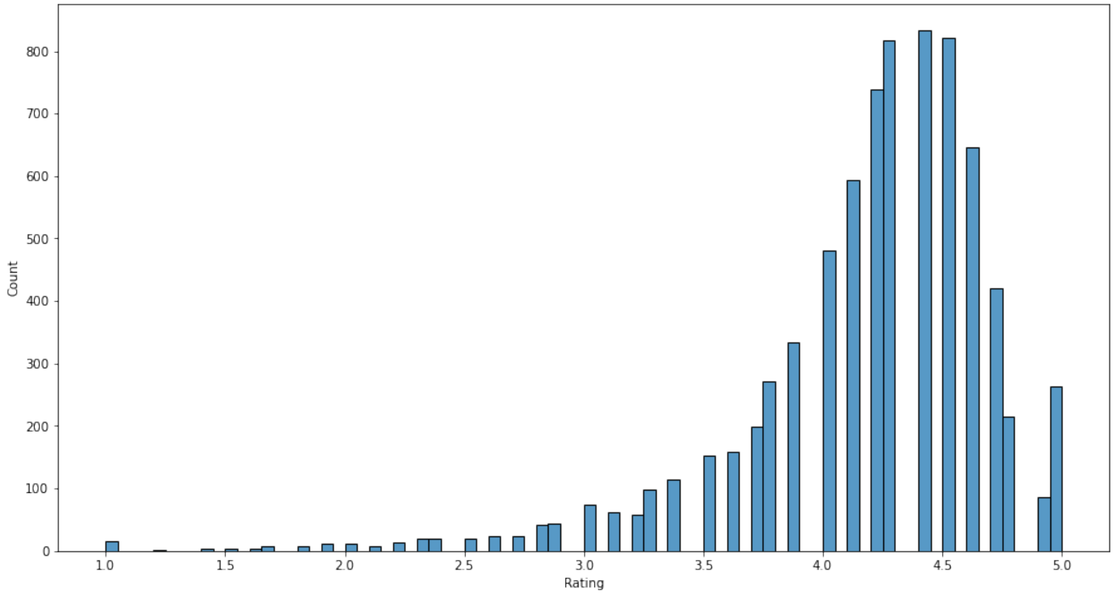
```
number of upper outliers is  89
```

Interpretation: number of upper outliers is 89

5.3. Histogram for Rating

```
sns.histplot(x='Rating',data=df)
```

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



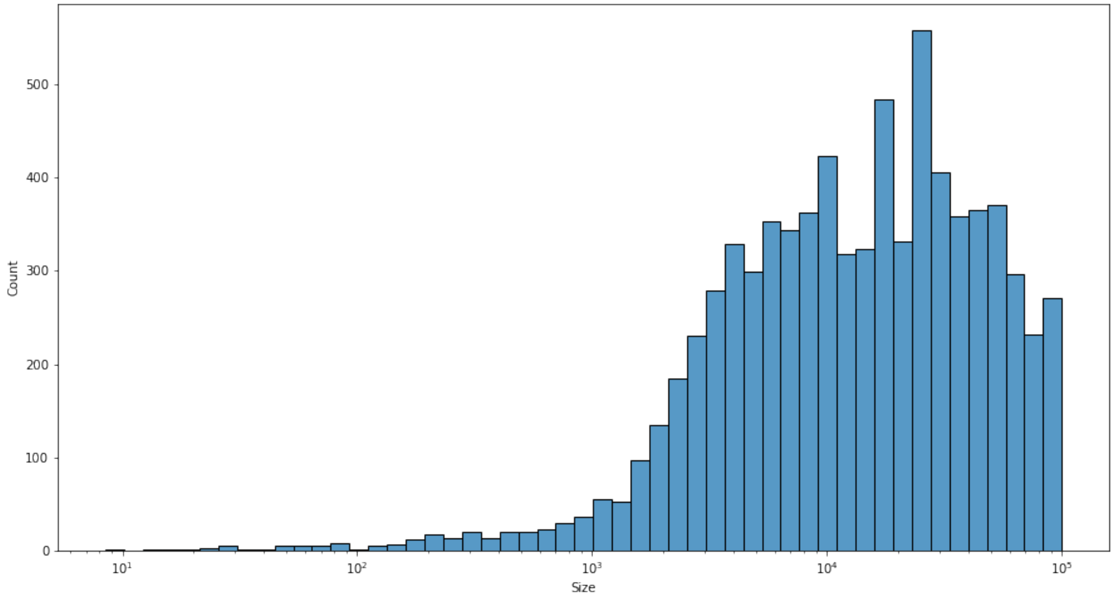
5.3.1 How are the ratings distributed? Is it more toward higher ratings?

Interpretation: ratings distributed towards higher rating

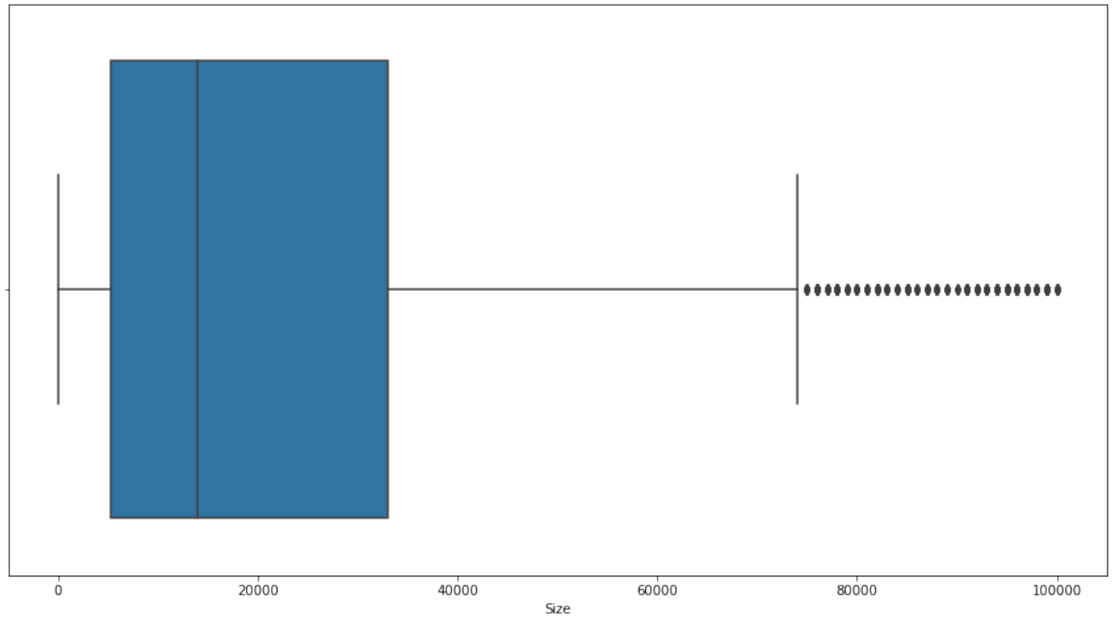
5.4. Histogram for Size

```
sns.histplot(x='Size',data=df,log_scale=True)
```

```
<AxesSubplot:xlabel='Size', ylabel='Count'>
```



```
bsize = sns.boxplot(x='Size', data=df)
```



5.4.1 Note down your observations for the plots made above.

interpretation: most of the app size lies under 100000

-
  ```
  Size_standard_deviation=stc.stdev(df.Size)
  review_standard_deviation
  ```

```
1864639.6094670836
```

-
  ```
  Size_mean=stc.mean(df.Size)
  Size_mean
  ```

```
22976.614293119088
```

-
  ```
  Size_outlier_uplimit=Size_mean+3*Size_standard_deviation
  Size_outlier_uplimit
  ```

```
93346.9260933562
```

-
  ```
  Size_outlier_downlimit=Size_mean-3*Size_standard_deviation
  Size_outlier_downlimit
  ```

```
-47393.697507118035
```

-
  ```
  print("number of upper outliers is ",len(df[(df.Size>Size_outlier_uplimit) ]))
  ```

```
number of upper outliers is  148
```

5.4.1 Note down your observations for the plots made above. Which of these seem to have outliers?

interpretation: App which are more than size of 93346.92 consider as outlier. number of outlier are 148

6. Outlier treatment:

6.1. Price: From the box plot, it seems like there are some apps with very high price. A price of $200 for an application on the Play Store is very high and suspicious!

6.1.1. Check out the records with very high price

6.1.1. Is 200 indeed a high price?

6.1.2 Drop these as most seem to be junk apps

6.2. Reviews: Very few apps have very high number of reviews. These are all star apps that don't help with the analysis and, in fact, will skew it. Drop records having more than 2 million reviews.

6.3 Installs: There seems to be some outliers in this field too. Apps having very high number of installs should be dropped from the analysis.

6.3.1 Find out the different percentiles – 10, 25, 50, 70, 90, 95, 99

6.3.2 Decide a threshold as cutoff for outlier and drop records having values more than that

6.1 Price: From the box plot, it seems like there are some apps with very high price. A price of $200 for an application on the Play Store is very high and suspicious!

6.1.1 Check out the records with very high price

- `df[df.Price>=200]`

```
                                 App   Category  Rating  Reviews     Size  \
4197          most expensive app (H)     FAMILY     4.3        6   1500.0   
4362                      ⍰ I'm rich  LIFESTYLE     3.8      718  26000.0   
4367        I'm Rich - Trump Edition  LIFESTYLE     3.6      275   7300.0   
5351                       I am rich  LIFESTYLE     3.8     3547   1800.0   
5354                  I am Rich Plus     FAMILY     4.0      856   8700.0   
5355                   I am rich VIP  LIFESTYLE     3.8      411   2600.0   
5356               I Am Rich Premium    FINANCE     4.1     1867   4700.0   
5357             I am extremely Rich  LIFESTYLE     2.9       41   2900.0   
5358                      I am Rich!    FINANCE     3.8       93  22000.0   
5359              I am rich(premium)    FINANCE     3.5      472    965.0   
5362                   I Am Rich Pro     FAMILY     4.4      201   2700.0   
5364  I am rich (Most expensive app)    FINANCE     4.1      129   2700.0   
5366                       I Am Rich     FAMILY     3.6      217   4900.0   
5369                       I am Rich    FINANCE     4.3      180   3800.0   
5373              I AM RICH PRO PLUS    FINANCE     4.0       36  41000.0   

      Installs  Type   Price Content Rating         Genres       Last Updated  
\
4197       100  Paid  399.99       Everyone  Entertainment      July 16, 2018   
4362     10000  Paid  399.99       Everyone      Lifestyle     March 11, 2018   
4367     10000  Paid  400.00       Everyone      Lifestyle        May 3, 2018   
5351    100000  Paid  399.99       Everyone      Lifestyle   January 12, 2018   
5354     10000  Paid  399.99       Everyone  Entertainment       May 19, 2018   
5355     10000  Paid  299.99       Everyone      Lifestyle      July 21, 2018   
5356     50000  Paid  399.99       Everyone        Finance  November 12, 2017   
5357      1000  Paid  379.99       Everyone      Lifestyle       July 1, 2018   
5358      1000  Paid  399.99       Everyone        Finance  December 11, 2017   
5359      5000  Paid  399.99       Everyone        Finance        May 1, 2017   
5362      5000  Paid  399.99       Everyone  Entertainment       May 30, 2017   
5364      1000  Paid  399.99           Teen        Finance   December 6, 2017   
5366     10000  Paid  389.99       Everyone  Entertainment      June 22, 2018   
5369      5000  Paid  399.99       Everyone        Finance     March 22, 2018   
5373      1000  Paid  399.99       Everyone        Finance      June 25, 2018   

     Current Ver   Android Ver  
4197         1.0    7.0 and up  
4362       1.0.0    4.4 and up  
4367       1.0.1    4.1 and up  
5351         2.0  4.0.3 and up  
5354         3.0    4.4 and up  
5355       1.1.1    4.3 and up  
5356         1.6    4.0 and up  
5357         1.0    4.0 and up  
5358         1.0    4.1 and up  
5359         3.4    4.4 and up  
```

```
5362        1.54    1.6 and up
5364           2  4.0.3 and up
5366         1.5    4.2 and up
5369         1.0    4.2 and up
5373       1.0.2    4.1 and up
```

-
  ```
  print("Number of Apps with price >= 200 are ",len(df[(df.Price>=200)
  ]))
  ```

```
 Number of Apps with price >= 200 are  15
```

- Is 200 indeed a high price?

- `6.1.2 Drop these as most seem to be junk apps`

-
  ```
  df.drop(df.index[(df.Price>=200)], inplace=True)
  len(df.index)
  ```

```
7702
```

6.2. Reviews: Very few apps have very high number of reviews

These are all star apps that don’t help with the analysis and, in fact, will skew it. Drop records having more than 2 million reviews.

-
  ```
  df.drop(df.index[(df.Reviews>=2000000)], inplace=True)
  len(df.index)
  ```

```
7483
```

- 6.3 Installs: There seems to be some outliers in this field too. Apps having very high number of installs should be dropped from the analysis.

6.3.1 Find out the different percentiles – 10, 25, 50, 70, 90, 95, 99

```
install_10_perc=np.percentile(df.Installs, 10)
install_10_perc
```

```
1000.0
```

```
install_25_perc=np.percentile(df.Installs, 25)
install_25_perc
```

```
10000.0
```

```
install_50_perc=np.percentile(df.Installs, 50)
install_50_perc
```

```
100000.0
```

```
install_70_perc=np.percentile(df.Installs, 70)
install_70_perc
```

```
1000000.0
```

```
install_90_perc=np.percentile(df.Installs, 90)
install_90_perc
```

```
10000000.0
```

```
install_95_perc=np.percentile(df.Installs, 95)
install_95_perc
```

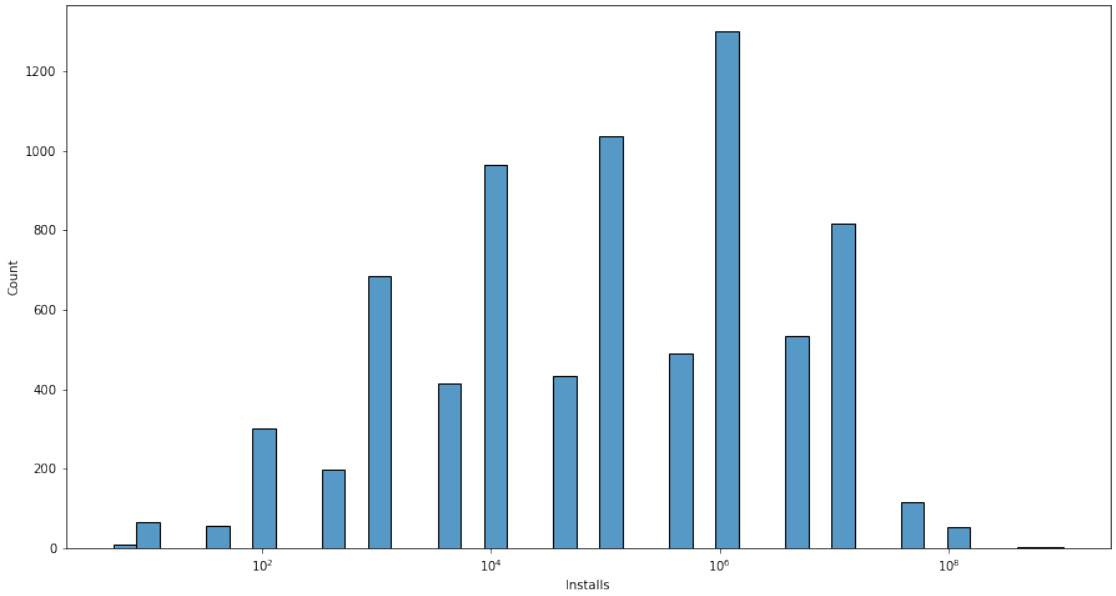
```
10000000.0
```

```
install_99_perc=np.percentile(df.Installs, 99)
install_99_perc
```

```
50000000.0
```

```
sns.histplot(data=df,x='Installs',log_scale=True)
```

```
<AxesSubplot:xlabel='Installs', ylabel='Count'>
```



-
  ```
  Installs_standard_deviation=stc.stdev(df.Installs)
  Installs_standard_deviation
  ```

```
27818305.317482274
```

-
  ```
  Installs_mean=stc.mean(df.Installs)
  Installs_mean
  ```

```
3947464.5449685953
```

-
  ```
  Installs_outlier_uplimit=Installs_mean+3*Installs_standard_deviation
  Installs_outlier_uplimit
  ```

```
87402380.49741541
```

-
  ```
  print("number of upper outliers is ",len(df[(df.Installs>Installs_outlier_uplimit) ]))
  ```

```
number of upper outliers is  60
```

-
  ```
  df.drop(df.index[df.Installs >= 87402380.49741541],inplace=True)
  len(df.index)
  ```

```
7423
```

I Decide more than that of 99% as an threshol cutoff for outlier and drop records having values more than that.

-
  ```
  df.drop(df.index[df.Installs >= install_99_perc],inplace=True)
  len(df.index)
  ```

```
7307
```

7. **Bivariate analysis: Let's look at how the available predictors relate to the variable of interest, i.e., our target variable rating. Make scatter plots (for numeric features) and box plots (for character features) to assess the relations between rating and the other features.**

7.1 Make scatter plot/joinplot for Rating vs. Price

7.1.1 What pattern do you observe? Does rating increase with price?

7.2 Make scatter plot/joinplot for Rating vs. Size

7.2.1 Are heavier apps rated better?

7.3 Make scatter plot/joinplot for Rating vs. Reviews

7.3.1 Does more review mean a better rating always?

7.4 Make boxplot for Rating vs. Content Rating

7.4.1 Is there any difference in the ratings? Are some types liked better?

7.5 Make boxplot for Ratings vs. Category

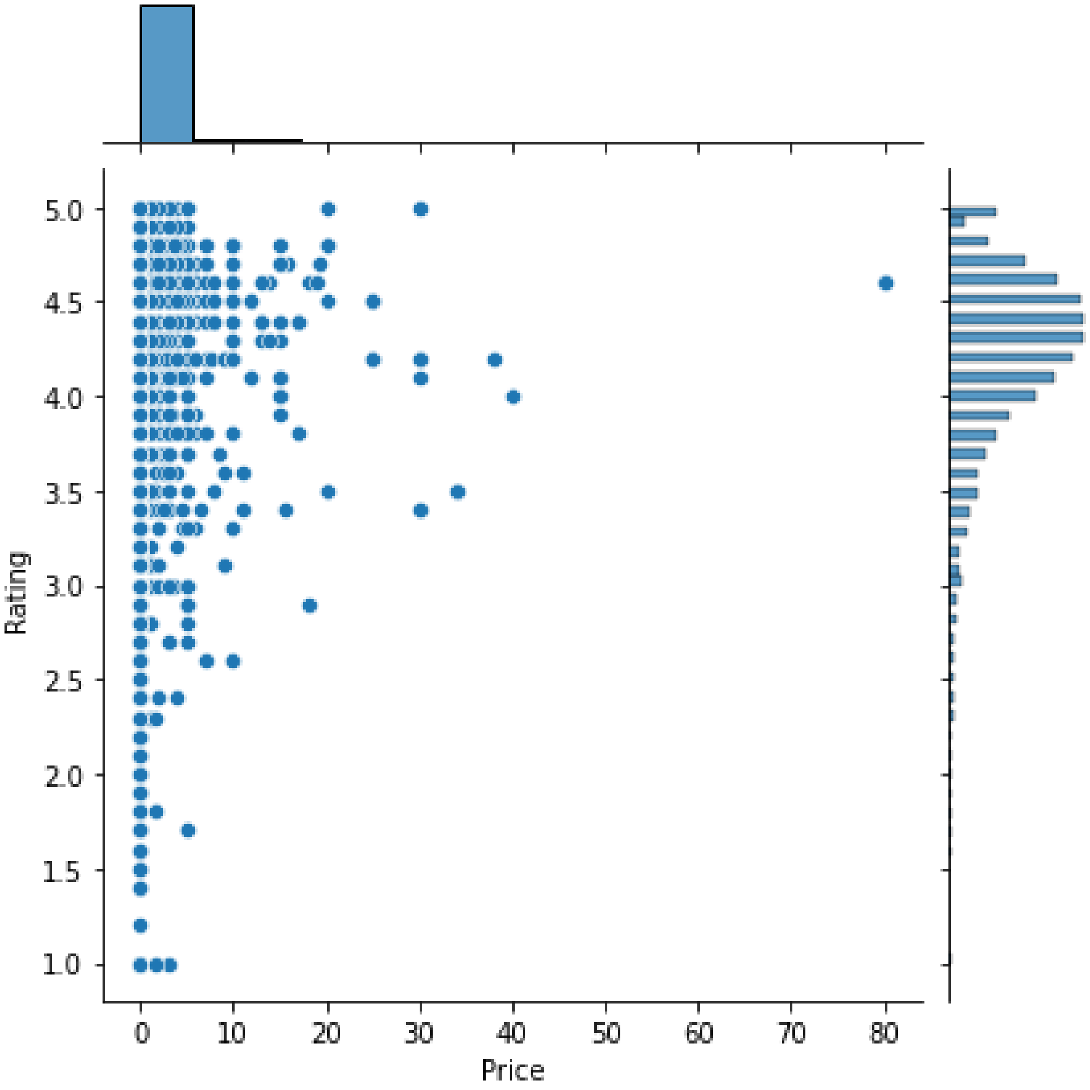
7.5.1 Which genre has the best ratings?

7.6 For each of the plots above, note down your observation

7.1. Make scatter plot/joinplot for Rating vs. Price

```
sns.jointplot(data=df,y='Rating',x='Price')
```

```
<seaborn.axisgrid.JointGrid at 0x19abbee7400>
```



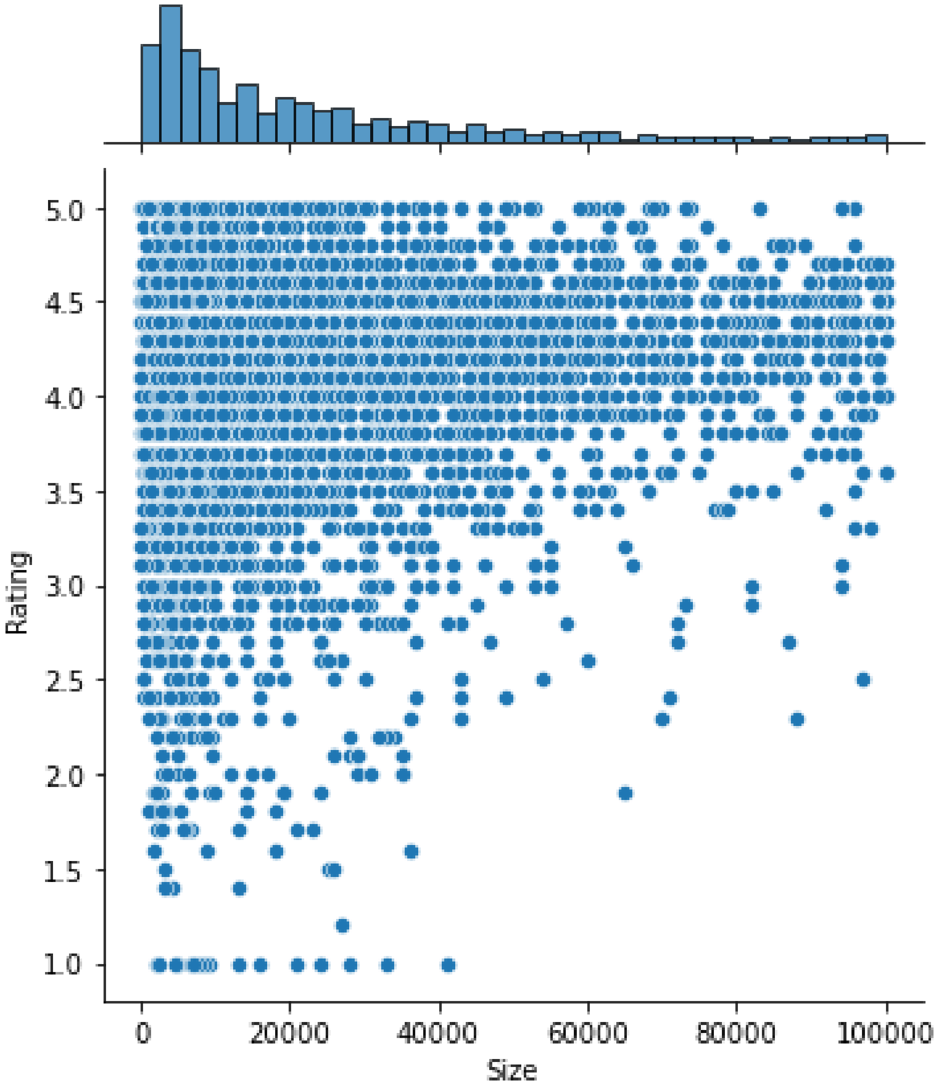
7.1.1 What pattern do you observe? Does rating increase with price? interpretation: Most of Apps with high price get > 3 Rating but this is because majority of apps are with low price. In addition most apps get rating > 3.

Concusion: We cannot consider there is a good relationship between Rating and Price. It seems Price has limited impact on Rating.

7.2 Make scatter plot/joinplot for Rating vs. Size

```
sns.jointplot(data=df,y='Rating',x='Size')
```

```
<seaborn.axisgrid.JointGrid at 0x19ab7cc4a60>
```



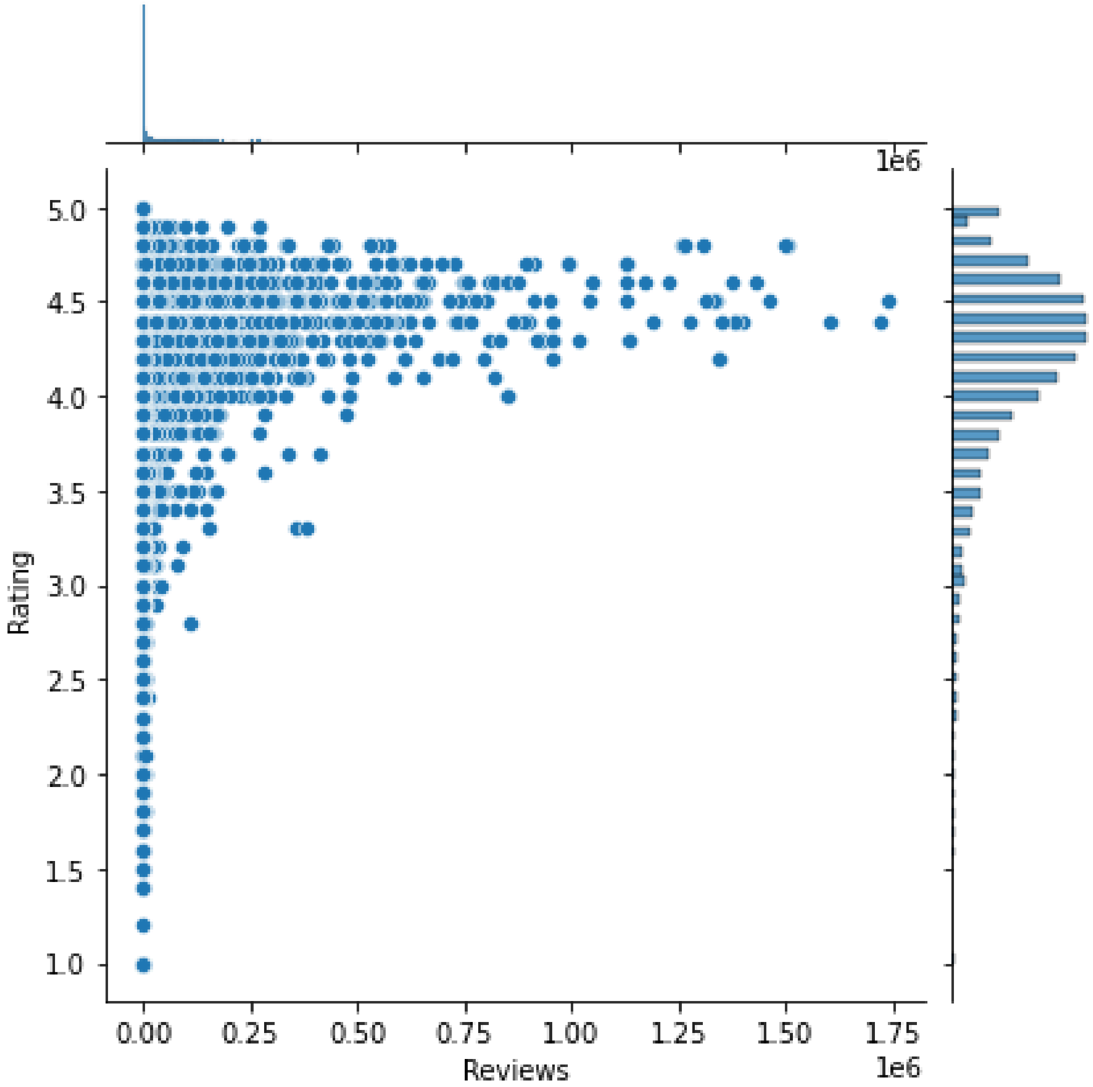
7.2.1 Are heavier apps rated better?

interpretation:if we look to the area where most apps rated > 3 almost the points are evenly distributed The relationship between Size and rating is very weak

7.3. Make scatter plot/joinplot for Rating vs. Reviews

```
sns.jointplot(data=df,y='Rating',x='Reviews')
```

```
<seaborn.axisgrid.JointGrid at 0x19ac176c520>
```



7.3.1 Does more review mean a better rating always?

interpretation: Although the relationship seems also not so strong, but we can notice that there is some concentration of apps with higher reviews in high rating area. It seems good apps get more reviews than others

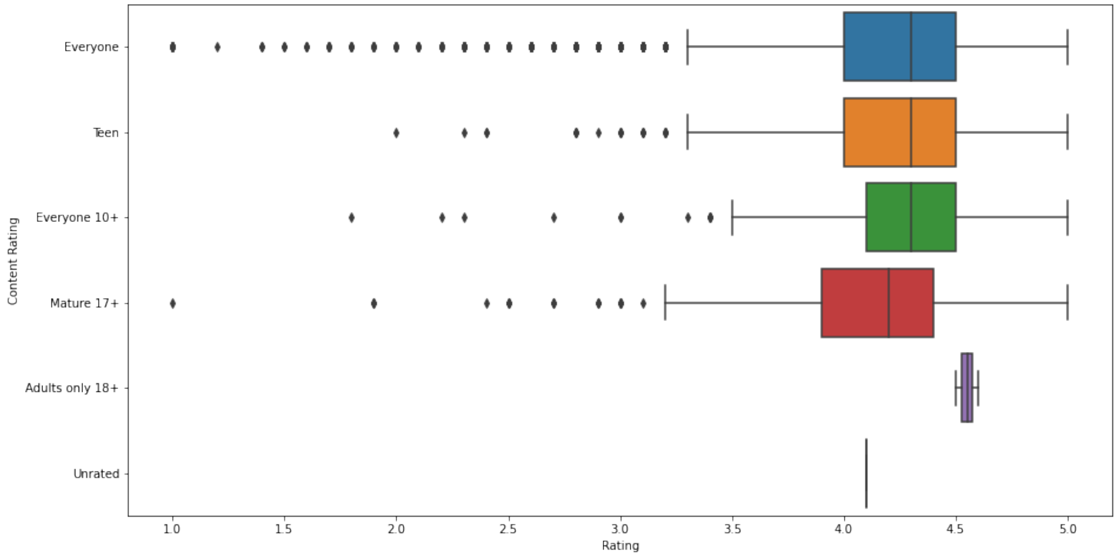
7.4 Make boxplot for Rating vs. Content Rating

```
df['Content Rating'].unique()
```

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

```
sns.boxplot(data=df,x='Rating',y='Content Rating')
```

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



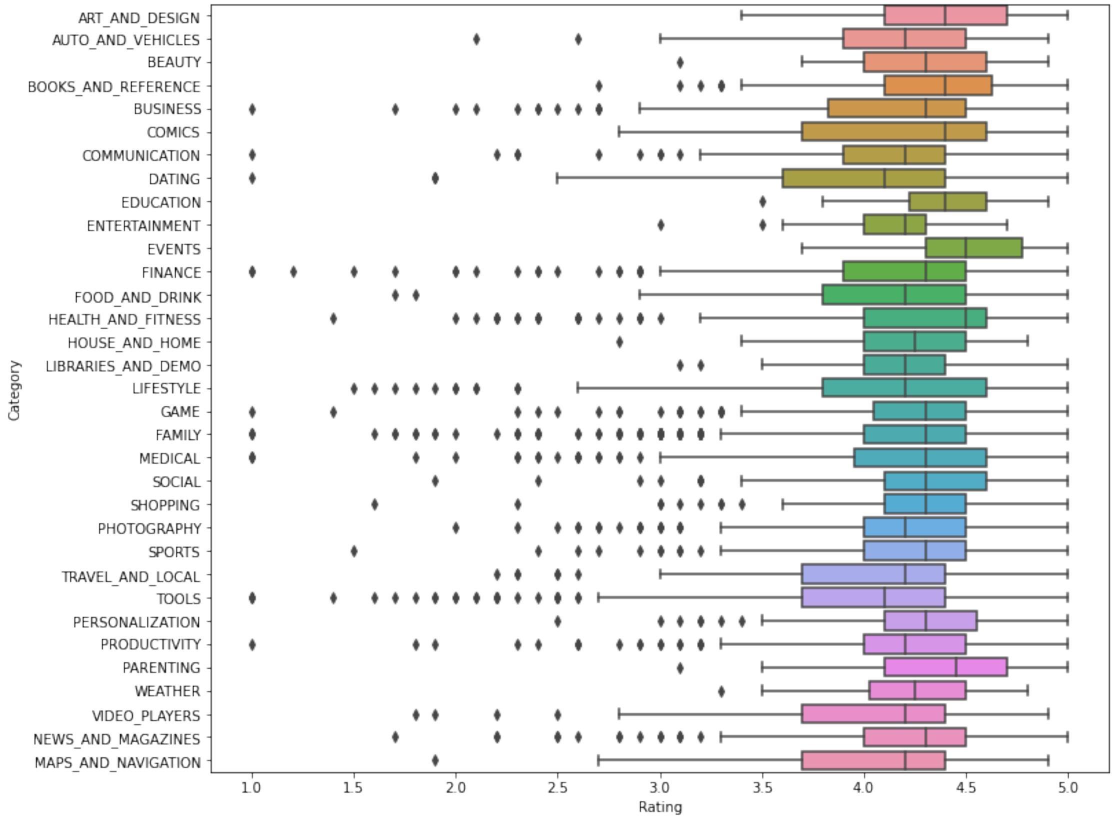
7.4.1 Is there any difference in the ratings? Are some types liked better? Is there any difference in the ratings? Are some types liked better?

interpretation: Apps of Adults only 18+ has higher rating than others while Mature 17+ gets less liks. Others seem to be closed. Content has good impact on Rating

```
#7.5 Make boxplot for Ratings vs. Category
```

-
  ```
  a4_dims = (11.7, 10.27)
  fig, ax = plt.subplots(figsize=a4_dims)
  sns.boxplot(data=df,x='Rating',y='Category',ax=ax)
  ```

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



7.5.1 Which genre has the best ratings?

while observing box plot i found that the Events genere has higest rating.

7.6 For each of the plots above, note down your observation

while observing above plot i foungout that Rating vs. Content Rating and Ratings vs. Category. gives better understanding of app rating

8. Data preprocessing

For the steps below, create a copy of the dataframe to make all the edits. Name it inp1.

8.1Reviews and Install have some values that are still relatively very high.
Before building a linear regression model, you need to reduce the skew. Apply log transformation (np.log1p) to Reviews and Installs.

8.2Drop columns App, Last Updated, Current Ver, and Android Ver.
These variables are not useful for our task.

8.3Get dummy columns for Category, Genres, and Content Rating.
This needs to be done as the models do not understand categorical data, and all data should be numeric.
Dummy encoding is one way to convert character fields to numeric. Name of dataframe should be inp2.

Q 8.1

```
inp1=df.copy()
inp1.Reviews=inp1.Reviews.apply(np.log1p)
```

Q 8.2

```
inp1.drop(columns=['App','Last Updated','Current Ver','Android Ver'],inplace=True)
```

```
inp1.shape
```

```
(7307, 9)
```

Q 8.3

-
  ```
  inp2= pd.get_dummies(inp1)
  ```

  ```
  inp2.shape
  ```

```
(7307, 158)
```

9. **Train test split and apply 70-30 split. Name the new dataframes df_train and df_test.**

Separate the dataframes into X_train, y_train, X_test, and y_test.

-
  ```
  data = inp2.drop(columns='Rating')
  data.shape
  ```

```
(7307, 157)
```

-
  ```
  target = pd.DataFrame(inp2.Rating)
  target.shape
  ```

```
(7307, 1)
```

-
  ```
  x_train, x_test, y_train, y_test = train_test_split(data, target,
  test_size=0.3, random_state=3)
  print("x_train shape is ", x_train.shape)
  print("y_train shape is ", y_train.shape)
  print("x_test shape is ", x_test.shape)
  print("y_test shape is ", y_test.shape)
  ```

```
x_train shape is  (5114, 157)
y_train shape is  (5114, 1)
x_test shape is  (2193, 157)
y_test shape is  (2193, 1)
```

11 . Model building

11.1 Use linear regression as the technique

11.2 Report the R2 on the train set

-
  ```
  model=LinearRegression()
  model.fit(x_train, y_train)
  ```

-
  ```
  LinearRegression()
  ```

-
  ```
  train_predict=model.predict(x_train)
  ```

-
  ```
  print("R2 value of the model(by train) is ", r2_score(y_train, train_predict))
  ```

```
R2 value of the model(by train) is  0.08010678015666617
```

1. **Make predictions on test set and report R2.**

```
test_predict=model.predict(x_test)
```

-
  ```
  print("R2 value of the model(by test) is ", r2_score(y_test, test_predict))
  ```

```
R2 value of the model(by test) is  0.0522779707476938
```

- pandoc C:\Users\shiwa\Downloads\App Rating Prediction.ipynb -s -o App Rating Prediction.docx