GooglePlayStore_EDA

September 17, 2024

0.0.1 Exploratory Data Analysis

1. Import the necessary libraries
2. read the files (CSv or Excel)
3. Data inspection
4. check for null values in the rows
5. Data handling and cleaning
6. describe function
7. check for inconsistencies in your data (missing vales, NAN, incorrect spellings in the rows)
8. Impute these inconsistencies (missing data or NAN values , replace it with mean, median or mode statistics)
9. Visualize the data , check for outliers in each columns
10. Inferences in for the data (on your understanding of the data set)

[2]:
```
# Importing necessary library
import numpy as np, pandas as pd
import seaborn as sns, matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore")
```

[3]:
```
# Reading the csv file
data = pd.read_csv("googleplaystore_v2.csv")
data.head()
```

[3]:

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

[4]:
```
# data inspection
data.shape
```

[4]:
```
(10841, 13)
```

[5]:
```
# checking the rows for null values
data.isnull().sum()
```

[5]:
```
App                0
Category           0
Rating          1474
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
```

1 Data handling and cleaning

For missing values we can do the following

1. dropping them
2. imputing them with statistical values
3. keep them as zeroes

Incorrect data 1. Clean them with correct values 2. clean and convert the entire columns

[7]:
```
data.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  float64
 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(2), object(11)
memory usage: 1.1+ MB
```

[8]:
```
# Since Rating column has lot of missing values i will delete those rows
data = data[-data.Rating.isnull()]
data.shape
```

[8]:
```
(9367, 13)
```

[9]:
```
# fetching the 3 null rows in the "Android Ver" column
data[data['Android Ver'].isnull()]
```

[9]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
4453	[substratum] Vacuum: P	PERSONALIZATION	4.4	230	11000.000000	1,000+	Paid	$1.49	Everyone	Personalization	July 20, 2018	4.4	NaN
4490	Pi Dark [substratum]	PERSONALIZATION	4.5	189	2100.000000	10,000+	Free	0	Everyone	Personalization	March 27, 2018	1.1	NaN
10472	Life Made WI-Fi Touchscreen Photo Frame		19.0	3.0M	21516.529524	Free	0	Everyone	NaN	February 11, 2018	1.0.19	4.0 and up	NaN

[10]:
```
# 10472 row is having shifted values so will drop this row
data.loc[10472,:]
data[(data['Android Ver'].isnull() & (data.Category == '1.9'))]
```

[10]:
```
                                          App Category  Rating Reviews  \
10472  Life Made WI-Fi Touchscreen Photo Frame      1.9    19.0    3.0M

               Size Installs Type     Price Content Rating             Genres  \
10472  21516.529524     Free    0  Everyone            NaN  February 11, 2018

      Last Updated Current Ver Android Ver
10472       1.0.19  4.0 and up         NaN
```

[11]:
```
data = data[-(data['Android Ver'].isnull() & (data.Category == '1.9'))]
```

[12]:
```
# Cross checking if its dropped
data[data['Android Ver'].isnull()]
```

[12]:
```
                        App         Category  Rating Reviews     Size  \
4453  [substratum] Vacuum: P  PERSONALIZATION     4.4     230  11000.0
4490    Pi Dark [substratum]  PERSONALIZATION     4.5     189   2100.0

     Installs  Type  Price Content Rating           Genres    Last Updated  \
4453   1,000+  Paid  $1.49       Everyone  Personalization   July 20, 2018
4490  10,000+  Free      0       Everyone  Personalization  March 27, 2018

     Current Ver Android Ver
4453         4.4         NaN
4490         1.1         NaN
```

[13]:
```
data["Android Ver"].value_counts()
```

[13]:
```
4.1 and up            2059
Varies with device    1319
4.0.3 and up          1240
4.0 and up            1131
4.4 and up             875
2.3 and up             582
5.0 and up             535
4.2 and up             338
2.3.3 and up           240
3.0 and up             211
2.2 and up             208
4.3 and up             207
2.1 and up             113
1.6 and up              87
6.0 and up              48
```

```
7.0 and up             41
3.2 and up             31
2.0 and up             27
5.1 and up             18
1.5 and up             16
3.1 and up              8
2.0.1 and up            7
4.4W and up             6
8.0 and up              5
7.1 and up              3
4.0.3 - 7.1.1           2
5.0 - 8.0               2
1.0 and up              2
7.0 - 7.1.1             1
4.1 - 7.1.1             1
5.0 - 6.0               1
Name: Android Ver, dtype: int64
```

2 Imputing the missing values using statistical techinique

For numerical columns we can use mean and median statistics

For categorical columns we use mode statistics

[15]:
```
data["Android Ver"] = data["Android Ver"].fillna(data["Android Ver"].mode()[0])
```

[16]:
```
data.isnull().sum()
```

[16]:
```
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       4
Android Ver       0
dtype: int64
```

[17]:
```
data[data["Current Ver"].isnull()]
```

[17]:
```
                                App             Category  Rating Reviews  \
15    Learn To Draw Kawaii Characters      ART_AND_DESIGN     3.2      55
```

```
1553            Market Update Helper  LIBRARIES_AND_DEMO     4.1   20145
6322          Virtual DJ Sound Mixer               TOOLS     4.2    4010
7333                      Dots puzzle              FAMILY     4.0     179

         Size    Installs  Type  Price Content Rating            Genres  \
15     2700.0      5,000+  Free      0       Everyone      Art & Design
1553     11.0  1,000,000+  Free      0       Everyone  Libraries & Demo
6322   8700.0    500,000+  Free      0       Everyone             Tools
7333  14000.0     50,000+  Paid  $0.99       Everyone            Puzzle

           Last Updated Current Ver Android Ver
15         June 6, 2018         NaN  4.2 and up
1553  February 12, 2013         NaN  1.5 and up
6322       May 10, 2017         NaN  4.0 and up
7333     April 18, 2018         NaN  4.0 and up
```

[18]:
```
data["Current Ver"].value_counts()
```

[18]:
```
Varies with device    1415
1.0                    458
1.1                    195
1.2                    126
1.3                    120
                      ...
2.9.10                   1
3.18.5                   1
1.3.A.2.9                1
9.9.1.1910               1
0.3.4                    1
Name: Current Ver, Length: 2638, dtype: int64
```

[19]:
```
data["Current Ver"] = data["Current Ver"].fillna(data["Current Ver"].mode()[0])
```

[20]:
```
data.isnull().sum()
```

[20]:
```
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
```

[21]:
```
# Handling incorrect data types
data.dtypes
```

[21]:
```
App                object
Category           object
Rating            float64
Reviews            object
Size              float64
Installs           object
Type               object
Price              object
Content Rating     object
Genres             object
Last Updated       object
Current Ver        object
Android Ver        object
dtype: object
```

[22]:
```
data.Price.value_counts()
```

[22]:
```
0          8719
$2.99       114
$0.99       107
$4.99        70
$1.99        59
           ...
$1.29          1
$299.99        1
$379.99        1
$37.99         1
$1.20          1
Name: Price, Length: 73, dtype: int64
```

[23]:
```
data.Price = data.Price.apply(lambda x: 0 if x == '0' else float(x[1:]))
data.dtypes
```

[23]:
```
App                object
Category           object
Rating            float64
Reviews            object
Size              float64
Installs           object
Type               object
Price             float64
```

```
Content Rating     object
Genres             object
Last Updated       object
Current Ver        object
Android Ver        object
dtype: object
```

[24]:
```
data.Reviews.value_counts()
```

[24]:
```
2         83
3         78
4         74
5         74
1         67
          ..
49657      1
41420      1
7146       1
44706      1
398307     1
Name: Reviews, Length: 5992, dtype: int64
```

[25]:
```
data.Reviews = data.Reviews.astype("int32")
data.dtypes
```

[25]:
```
App                object
Category           object
Rating            float64
Reviews             int32
Size              float64
Installs           object
Type               object
Price             float64
Content Rating     object
Genres             object
Last Updated       object
Current Ver        object
Android Ver        object
dtype: object
```

[26]:
```
data.Reviews.describe()
```

[26]:
```
count    9.366000e+03
mean     5.140498e+05
std      3.144042e+06
min      1.000000e+00
25%      1.862500e+02
```

```
50%      5.930500e+03
75%      8.153275e+04
max      7.815831e+07
Name: Reviews, dtype: float64
```

[27]:
```
data.describe()
```

[27]:
```
             Rating       Reviews           Size        Price
count  9366.000000  9.366000e+03    9366.000000  9366.000000
mean      4.191757  5.140498e+05   22705.733753     0.960928
std       0.515219  3.144042e+06   21305.040123    15.816585
min       1.000000  1.000000e+00       8.500000     0.000000
25%       4.000000  1.862500e+02    6600.000000     0.000000
50%       4.300000  5.930500e+03   21000.000000     0.000000
75%       4.500000  8.153275e+04   27000.000000     0.000000
max       5.000000  7.815831e+07  100000.000000   400.000000
```

[28]:
```
data.Installs.value_counts()
```

[28]:
```
1,000,000+        1577
10,000,000+       1252
100,000+          1150
10,000+           1010
5,000,000+         752
1,000+             713
500,000+           538
50,000+            467
5,000+             432
100,000,000+       409
100+               309
50,000,000+        289
500+               201
500,000,000+        72
10+                 69
1,000,000,000+      58
50+                 56
5+                   9
1+                   3
Name: Installs, dtype: int64
```

[29]:
```
# clean function
def clean_installs(val):
    return int(val.replace(",", "").replace("+",""))
```

[30]:
```
type(clean_installs("3,000+"))
data.Installs = data.Installs.apply(clean_installs)
```

[31]:
```
data.Installs.value_counts()
```

[31]:
```
1000000       1577
10000000      1252
100000        1150
10000         1010
5000000        752
1000           713
500000         538
50000          467
5000           432
100000000      409
100            309
50000000       289
500            201
500000000       72
10              69
1000000000      58
50              56
5                9
1                3
Name: Installs, dtype: int64
```

[32]:
```
data.describe()
```

[32]:

	Rating	Reviews	Size	Installs	Price
count	9366.000000	9.366000e+03	9366.000000	9.366000e+03	9366.000000
mean	4.191757	5.140498e+05	22705.733753	1.789744e+07	0.960928
std	0.515219	3.144042e+06	21305.040123	9.123822e+07	15.816585
min	1.000000	1.000000e+00	8.500000	1.000000e+00	0.000000
25%	4.000000	1.862500e+02	6600.000000	1.000000e+04	0.000000
50%	4.300000	5.930500e+03	21000.000000	5.000000e+05	0.000000
75%	4.500000	8.153275e+04	27000.000000	5.000000e+06	0.000000
max	5.000000	7.815831e+07	100000.000000	1.000000e+09	400.000000

[33]:
```
data.dtypes
```

[33]:
```
App               object
Category          object
Rating           float64
Reviews            int32
Size             float64
Installs           int64
Type              object
Price            float64
Content Rating    object
Genres            object
```

```
Last Updated       object
Current Ver        object
Android Ver        object
dtype: object
```

[34]:
```
data.Type.value_counts()
```

[34]:
```
Free    8719
Paid     647
Name: Type, dtype: int64
```

3 Lets do some sanity check

1. Ratings should have values 1 to 5
2. Reviews should be less than or equal to Installs
3. If Type column shows Free apps then should show 0 and paid should have some value in the Price column

[36]:
```
data[(data.Type == "Free") & (data.Price > 0)].shape

# Here its showing 0 which means data points are sitting correctly
```

[36]:
```
(0, 13)
```

[37]:
```
data[data.Reviews > data.Installs].shape
```

[37]:
```
(7, 13)
```

[38]:
```
data.Rating.describe()
```

[38]:
```
count    9366.000000
mean        4.191757
std         0.515219
min         1.000000
25%         4.000000
50%         4.300000
75%         4.500000
max         5.000000
Name: Rating, dtype: float64
```

[39]:
```
data[data["Reviews"] > data["Installs"]]
```

[39]:
```
                                App   Category  Rating  Reviews  \
2454             KBA-EZ Health Guide    MEDICAL     5.0        4
4663   Alarmy (Sleep If U Can) - Pro  LIFESTYLE     4.8    10249
5917                          Ra Ga Ba       GAME     5.0        2
6700                Brick Breaker BR       GAME     5.0        7
```

```
7402            Trovami se ci riesci       GAME     5.0       11
8591                        DN Blog     SOCIAL     5.0       20
10697                       Mu.F.O.       GAME     5.0        2

               Size  Installs  Type  Price Content Rating     Genres  \
2454   25000.000000         1  Free   0.00       Everyone    Medical
4663   21516.529524     10000  Paid   2.49       Everyone  Lifestyle
5917   20000.000000         1  Paid   1.49       Everyone     Arcade
6700   19000.000000         5  Free   0.00       Everyone     Arcade
7402    6100.000000        10  Free   0.00       Everyone     Arcade
8591    4200.000000        10  Free   0.00           Teen     Social
10697  16000.000000         1  Paid   0.99       Everyone     Arcade

          Last Updated         Current Ver         Android Ver
2454     August 2, 2018               1.0.72        4.0.3 and up
4663      July 30, 2018  Varies with device  Varies with device
5917   February 8, 2017               1.0.4         2.3 and up
6700      July 23, 2018                 1.0         4.1 and up
7402     March 11, 2017                 0.1         2.3 and up
8591      July 23, 2018                 1.0         4.0 and up
10697     March 3, 2017                 1.0         2.3 and up
```

[40]:
```
condition = data["Reviews"] > data["Installs"]
data = data.drop(data[condition].index)
data.shape
```

[40]: (9359, 13)

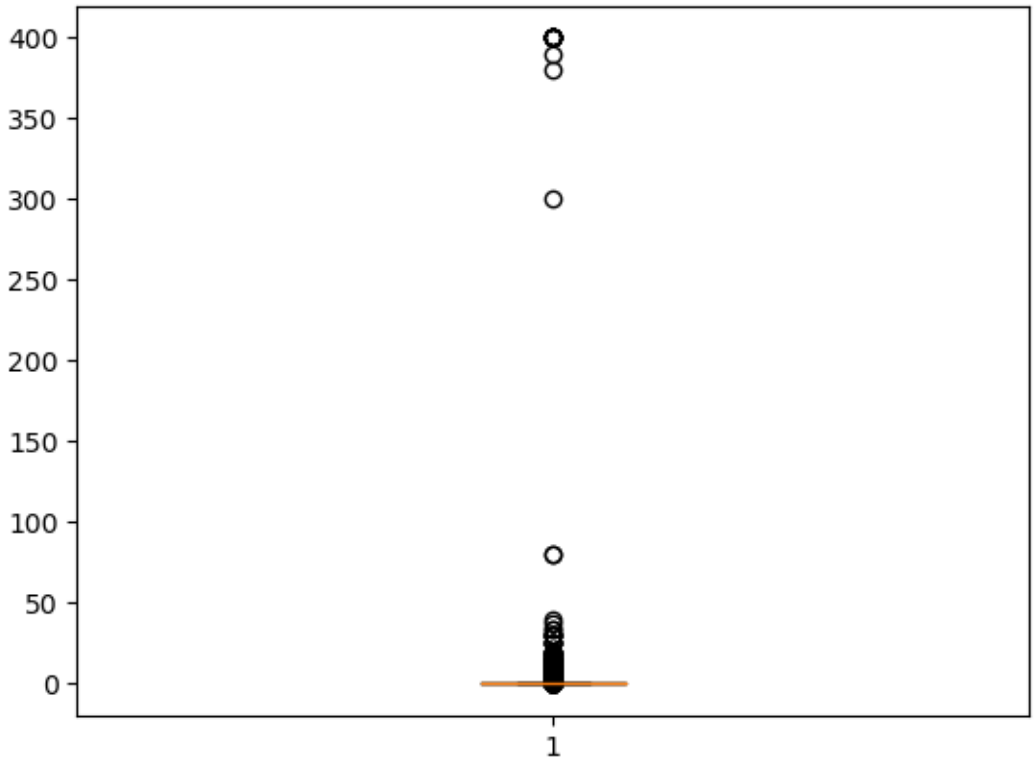
[41]:
```
data[data["Reviews"] > data["Installs"]].shape
```

[41]: (0, 13)

4 Data Visulizations

4.0.1 Univariate analysis

[81]:
```
# Outlier Analysis
plt.boxplot(data.Price)
plt.show()
```



[83]:
```
data[data.Price>200].describe()
```

[83]:

	Rating	Reviews	Size	Installs	Price
count	15.000000	15.000000	15.000000	15.000000	15.000000
mean	3.866667	603.266667	8904.333333	14606.666667	391.324000
std	0.381101	943.841729	11580.105919	26563.521353	25.875398
min	2.900000	6.000000	965.000000	100.000000	299.990000
25%	3.700000	111.000000	2650.000000	1000.000000	399.990000
50%	3.800000	217.000000	3800.000000	5000.000000	399.990000
75%	4.100000	595.000000	8000.000000	10000.000000	399.990000
max	4.400000	3547.000000	41000.000000	100000.000000	400.000000

[85]:
```
data[data.Price<200].describe()
```

[85]:

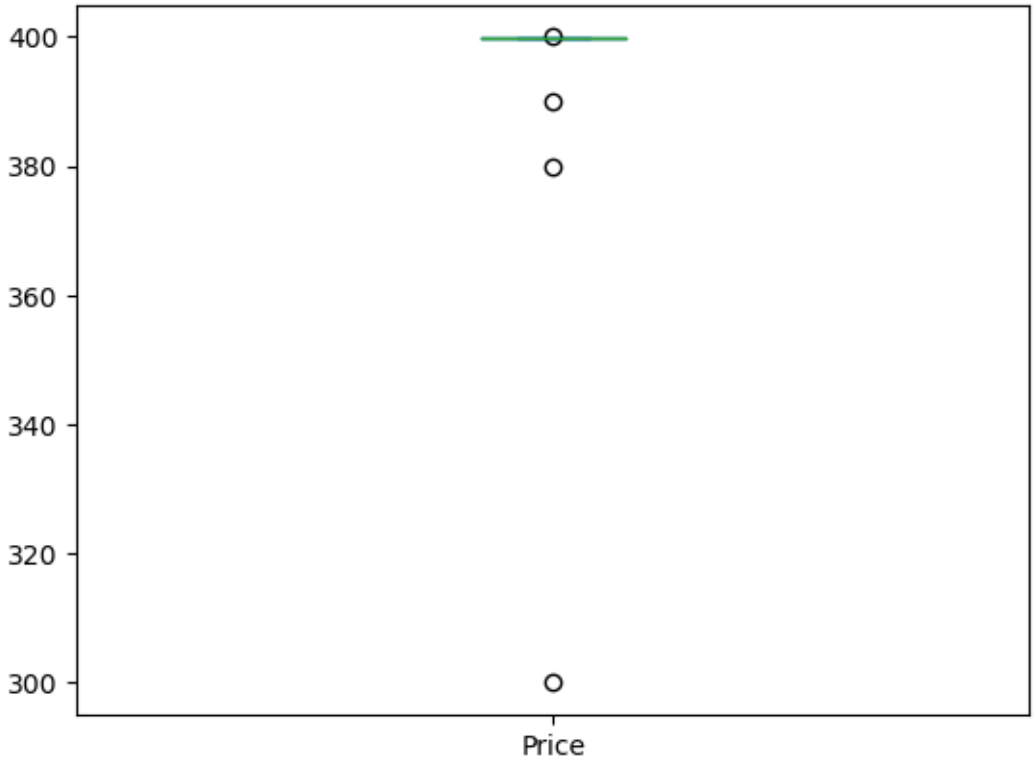
	Rating	Reviews	Size	Installs	Price
count	9344.000000	9.344000e+03	9344.000000	9.344000e+03	9344.000000
mean	4.191695	5.152581e+05	22732.932449	1.793956e+07	0.334463
std	0.515004	3.147643e+06	21316.475007	9.134144e+07	2.169925
min	1.000000	1.000000e+00	8.500000	5.000000e+00	0.000000
25%	4.000000	1.880000e+02	6600.000000	1.000000e+04	0.000000
50%	4.300000	5.998500e+03	21000.000000	5.000000e+05	0.000000
75%	4.500000	8.222650e+04	27000.000000	5.000000e+06	0.000000

```
max       5.000000  7.815831e+07  100000.000000  1.000000e+09    79.990000
```

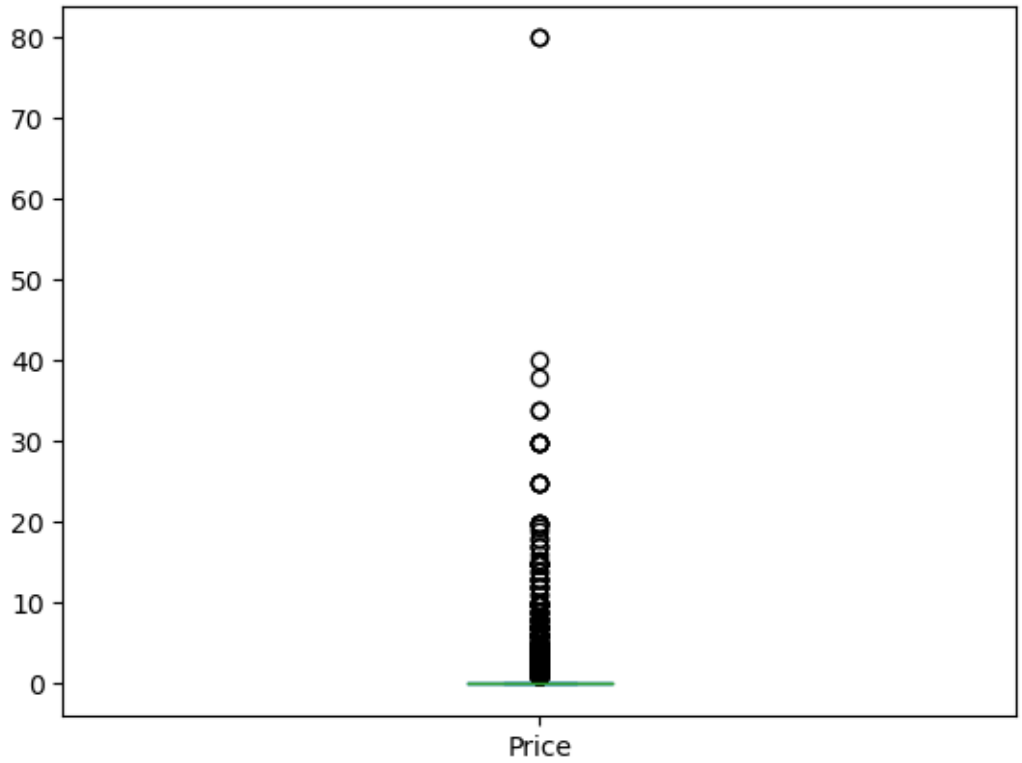
[87]:
```
data.Price.describe()
```

[87]:
```
count    9359.000000
mean        0.961116
std        15.822478
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max       400.000000
Name: Price, dtype: float64
```

[91]:
```
data[data.Price>200].Price.plot.box()
plt.show()
```



[93]:
```
data[data.Price<200].Price.plot.box()
plt.show()
```



[95]:
```
data[data.Price>30].shape
```

[95]:
```
(21, 13)
```

[97]:
```
data[data.Price>30].describe()
```

[97]:

	Rating	Reviews	Size	Installs	Price
count	21.000000	21.000000	21.000000	21.000000	21.000000
mean	3.923810	466.714286	16031.666667	10676.190476	294.085714
std	0.414614	821.064135	21178.196532	23119.708146	159.424893
min	2.900000	6.000000	965.000000	100.000000	33.990000
25%	3.600000	92.000000	2600.000000	1000.000000	79.990000
50%	4.000000	201.000000	4700.000000	1000.000000	399.990000
75%	4.200000	411.000000	26000.000000	10000.000000	399.990000
max	4.600000	3547.000000	68000.000000	100000.000000	400.000000

[99]:
```
data[data.Price<=30].shape
```

[99]:
```
(9338, 13)
```

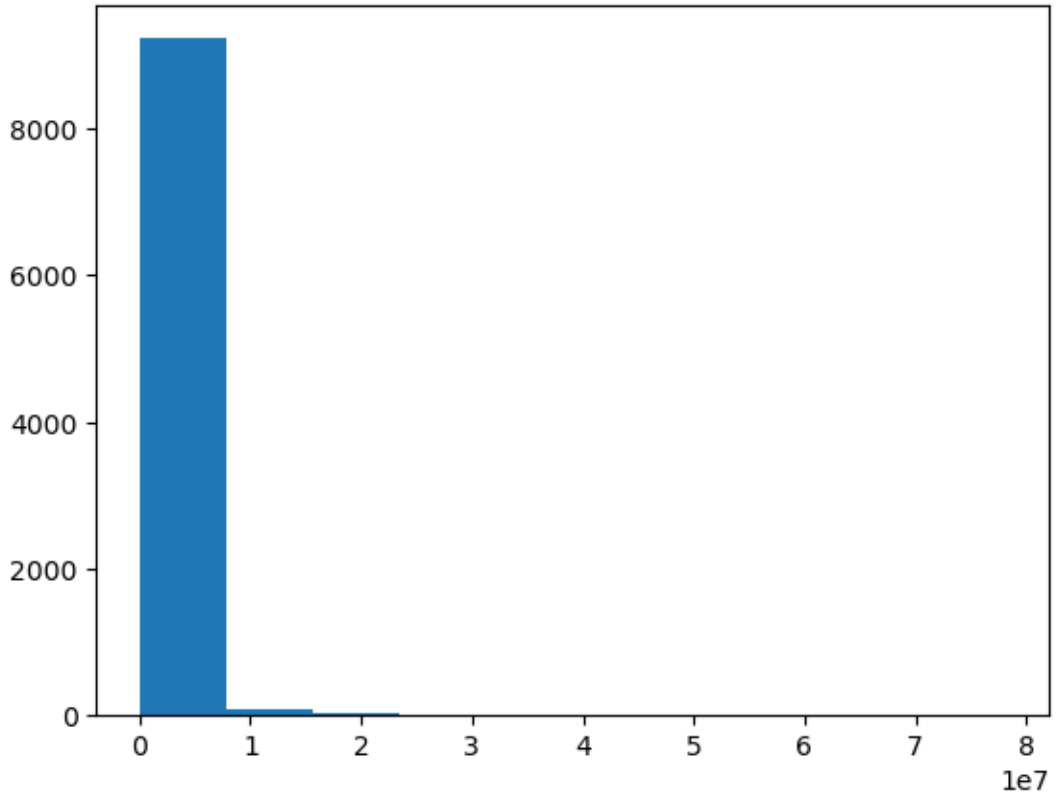
[101]:
```
data[data.Price<=30].describe()
```

[101]:

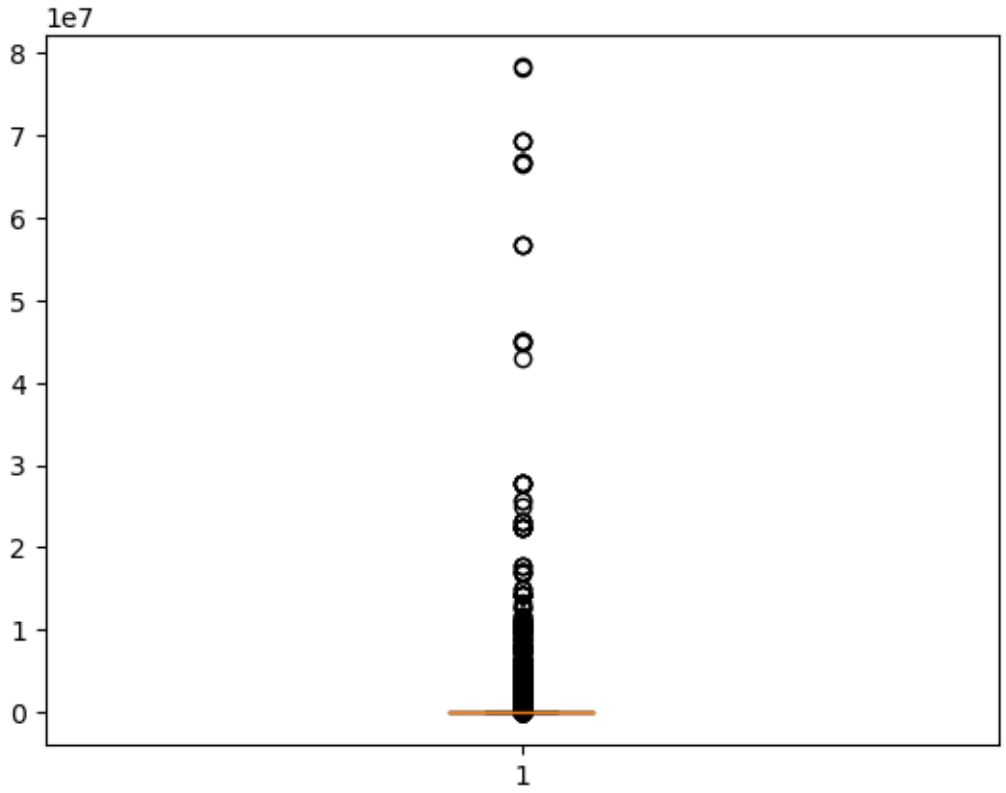
	Rating	Reviews	Size	Installs	Price
count	9338.000000	9.338000e+03	9338.000000	9.338000e+03	9338.000000
mean	4.191776	5.155891e+05	22725.789334	1.795108e+07	0.301915
std	0.515031	3.148627e+06	21310.340299	9.136965e+07	1.669887
min	1.000000	1.000000e+00	8.500000	5.000000e+00	0.000000
25%	4.000000	1.890000e+02	6600.000000	1.000000e+04	0.000000
50%	4.300000	6.011500e+03	21000.000000	5.000000e+05	0.000000
75%	4.500000	8.247100e+04	27000.000000	5.000000e+06	0.000000
max	5.000000	7.815831e+07	100000.000000	1.000000e+09	29.990000

4.0.2 Inference : Public choose to install applications from the Google Playstore that have nominal price.

[107]:
```
plt.hist(data.Reviews)
plt.show()
```



[109]:
```
plt.boxplot(data.Reviews)
plt.show()
```



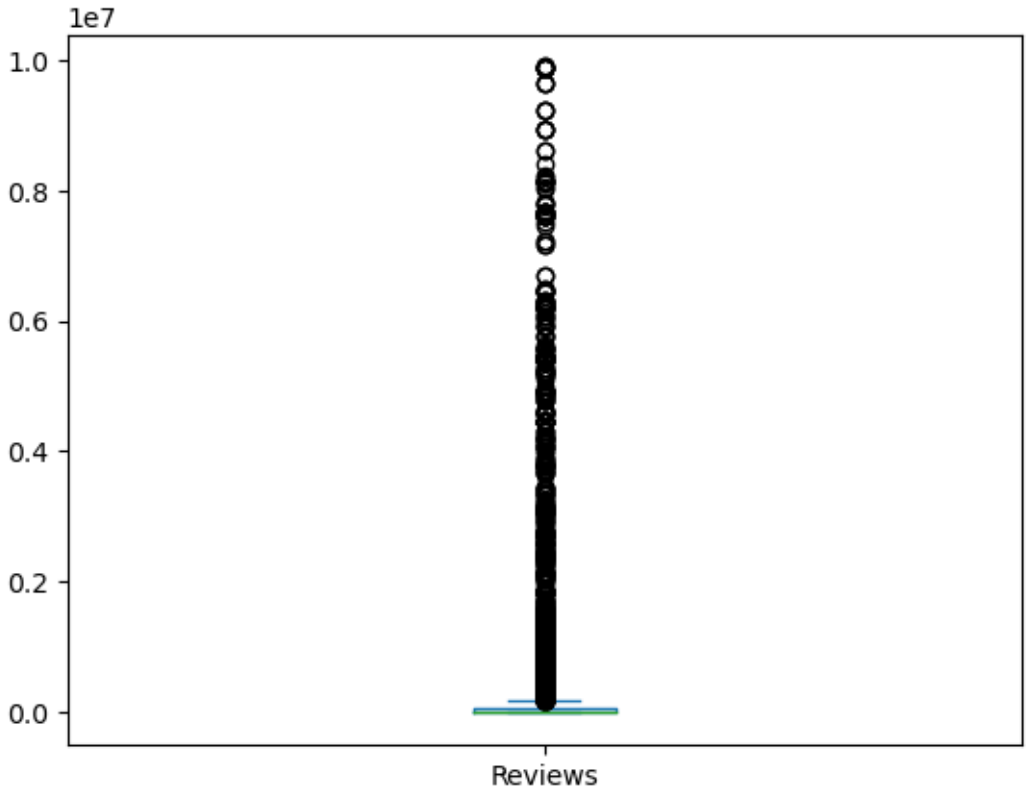
[111]:
```
data[data.Reviews >= 10000000].shape
```

[111]: (92, 13)

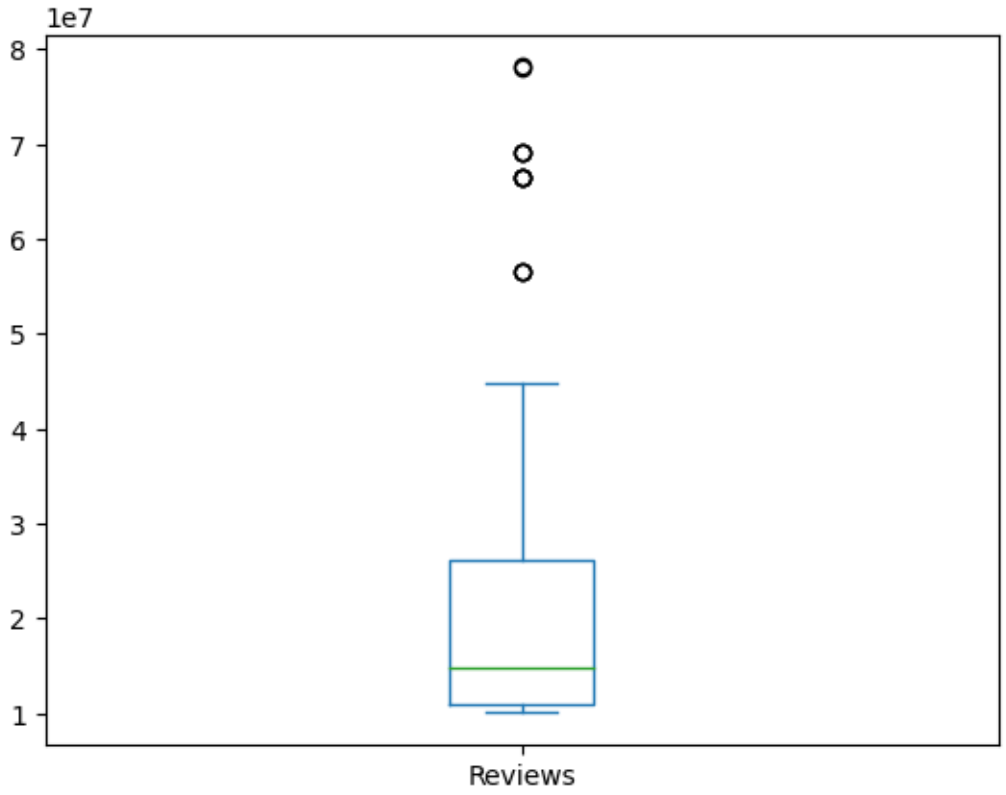
[113]:
```
data[data.Reviews <= 10000000].shape
```

[113]: (9267, 13)

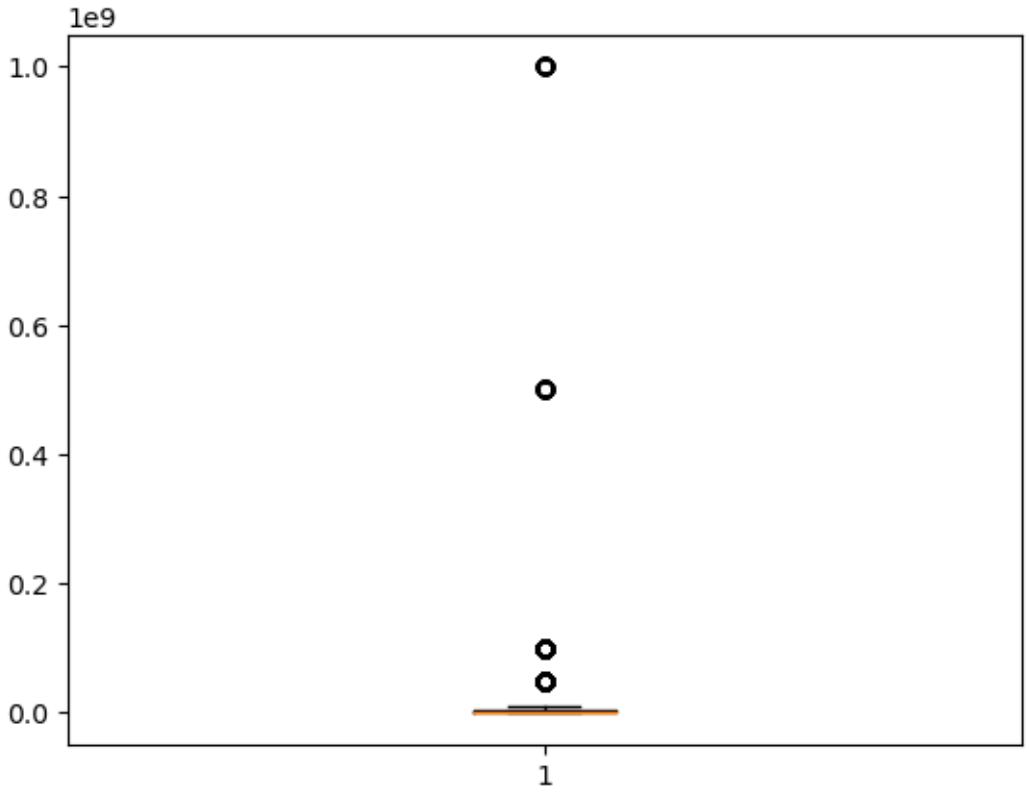
[125]:
```
data[data.Reviews <= 10000000].Reviews.plot.box()
plt.show()
```



[127]:
```
data[data.Reviews >= 10000000].Reviews.plot.box()
plt.show()
```



[117]:
```
plt.boxplot(data.Installs)
plt.show()
```



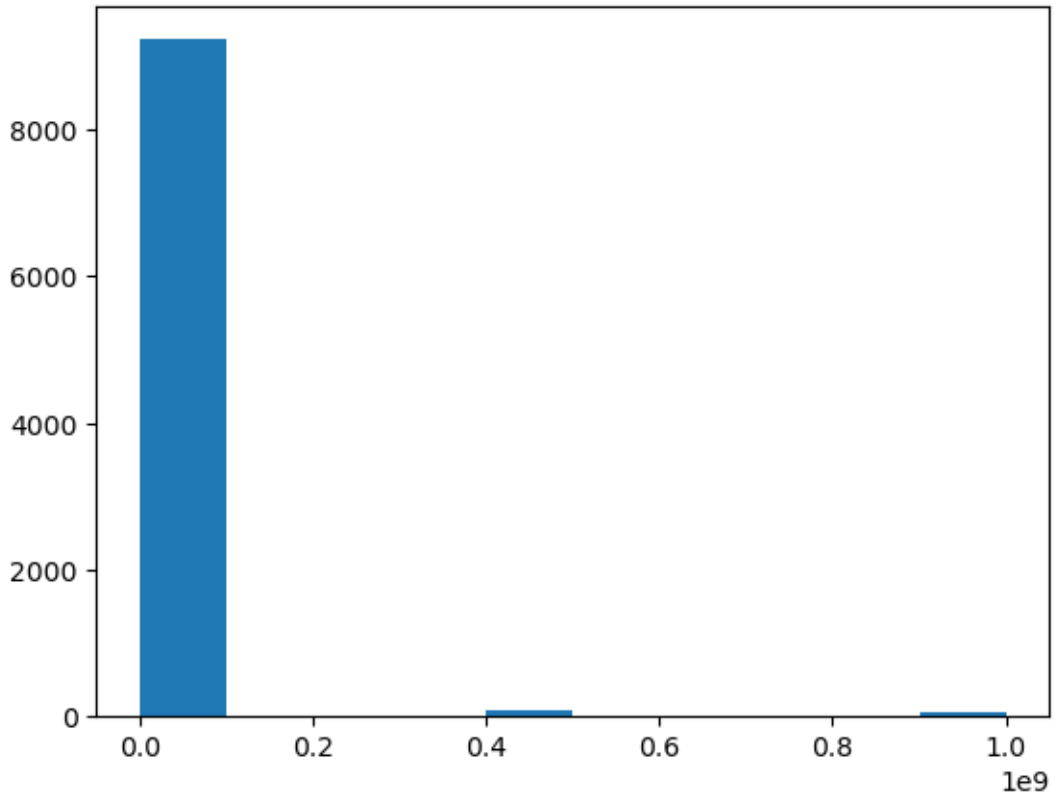
[129]:
```
data.Installs.describe()
```

[129]:
```
count    9.359000e+03
mean     1.791083e+07
std      9.127102e+07
min      5.000000e+00
25%      1.000000e+04
50%      5.000000e+05
75%      5.000000e+06
max      1.000000e+09
Name: Installs, dtype: float64
```

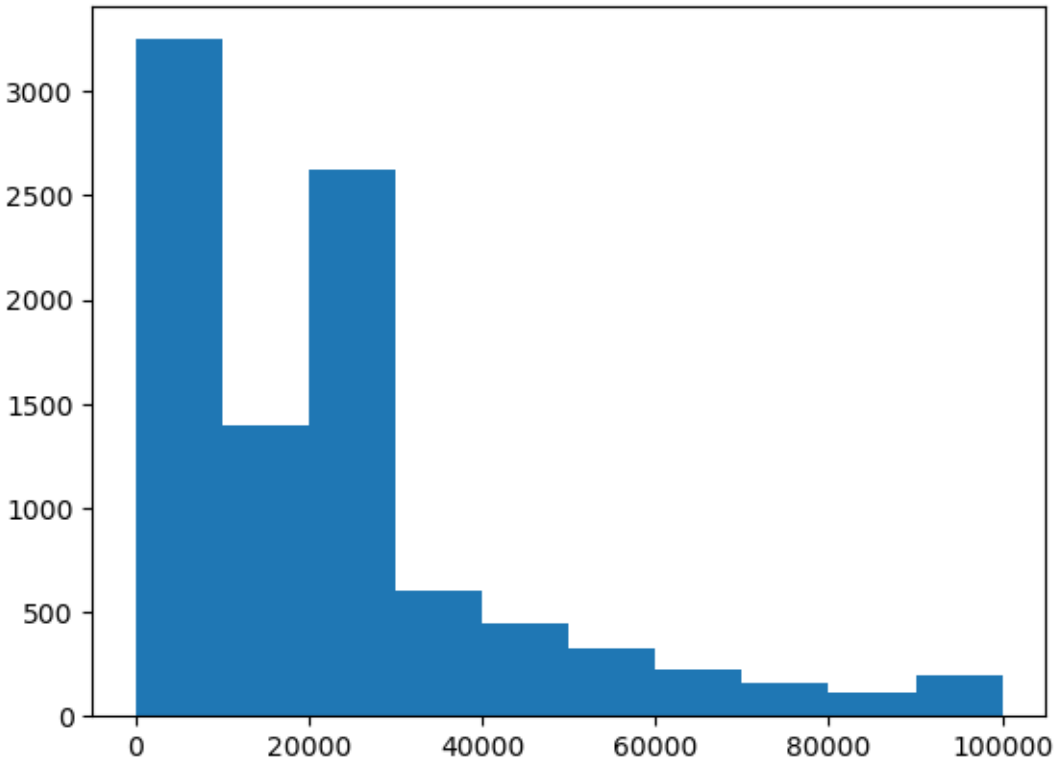
[131]:
```
data[data.Installs <= 10000000].shape
```

[131]:
```
(8531, 13)
```

[133]:
```
plt.hist(data.Installs)
plt.show()
```



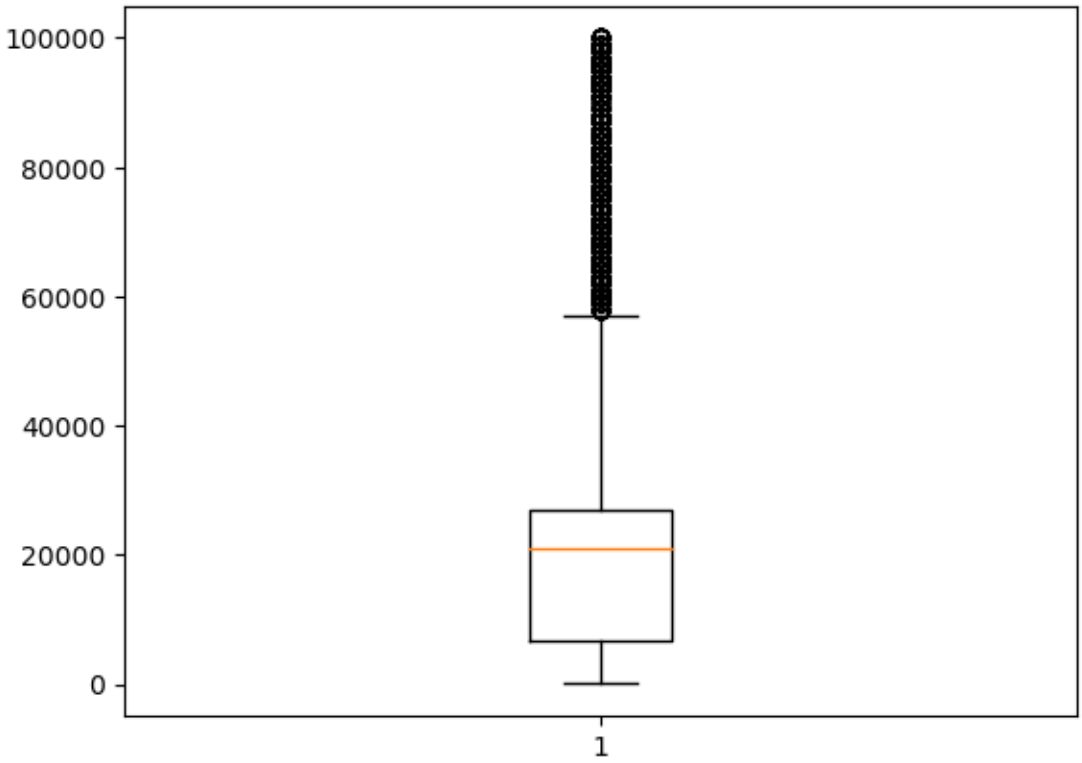
[135]:
```
plt.hist(data.Size)
plt.show()
```



[137]:
```
data.Size.describe()
```

[137]:
```
count      9359.000000
mean      22710.768864
std       21311.274234
min           8.500000
25%        6600.000000
50%       21000.000000
75%       27000.000000
max      100000.000000
Name: Size, dtype: float64
```

[139]:
```
plt.boxplot(data.Size)
plt.show()
```

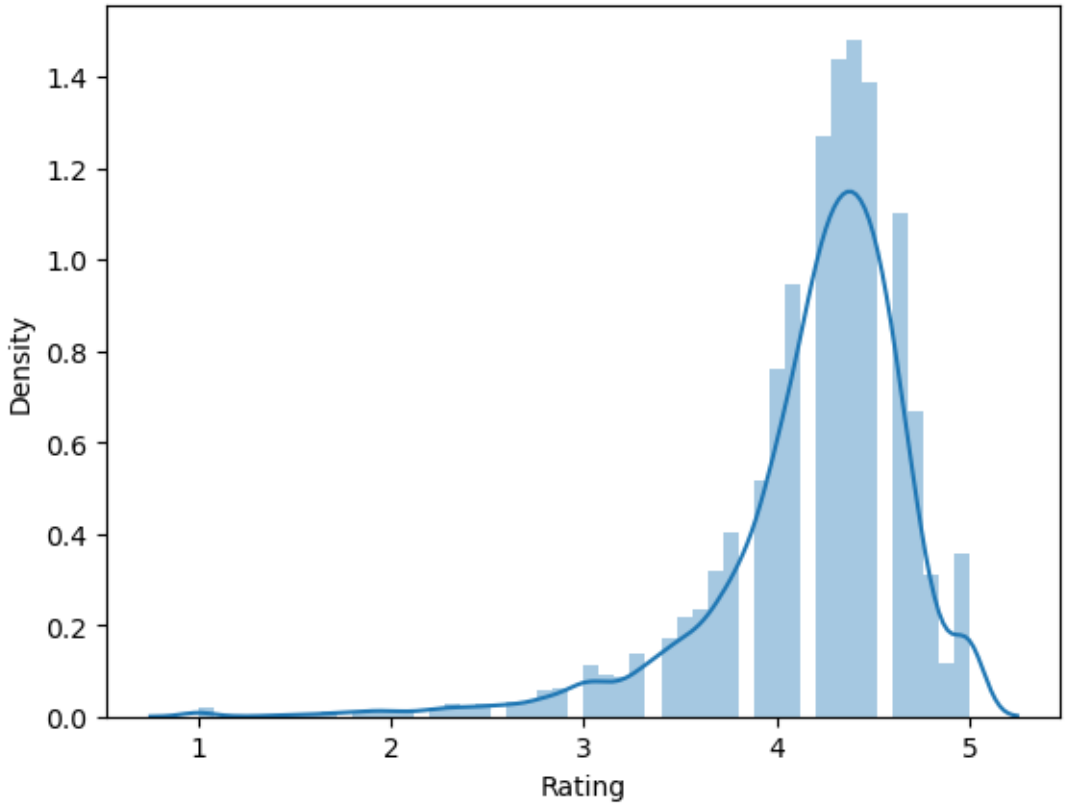


[145]:
```
plt.style.available
```

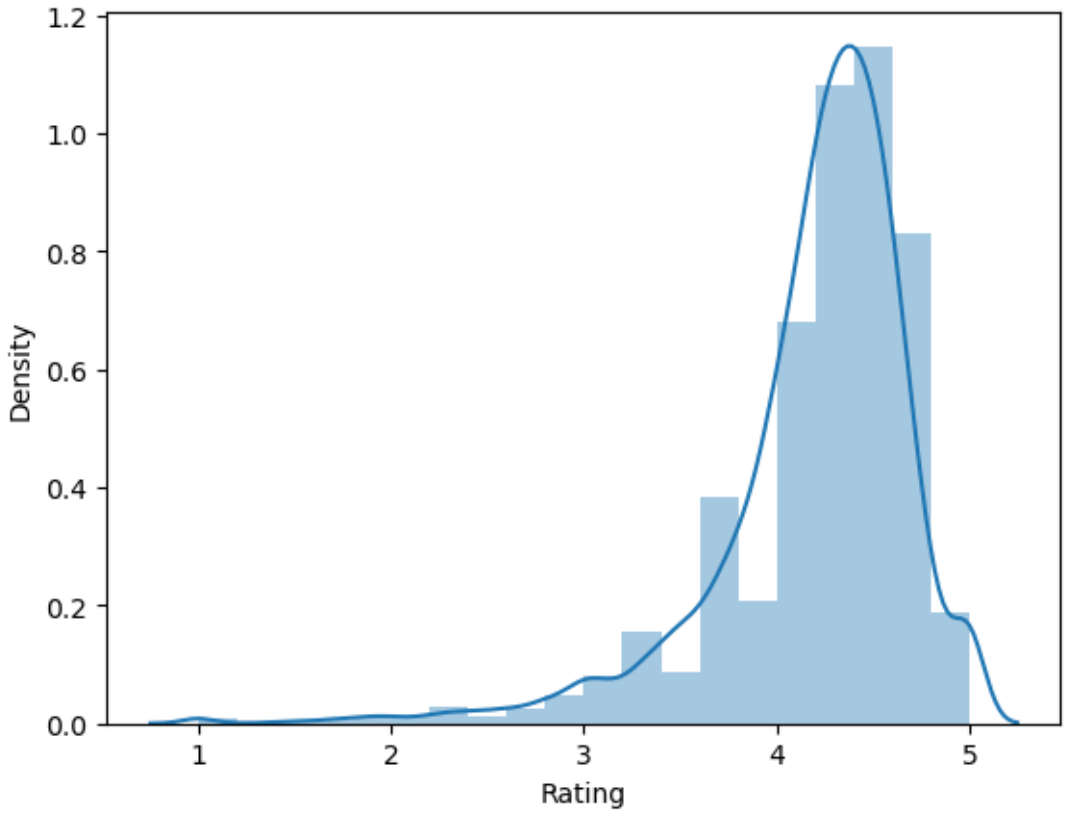
[145]:
```
['Solarize_Light2',
 '_classic_test_patch',
 '_mpl-gallery',
 '_mpl-gallery-nogrid',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn-v0_8',
 'seaborn-v0_8-bright',
 'seaborn-v0_8-colorblind',
 'seaborn-v0_8-dark',
 'seaborn-v0_8-dark-palette',
 'seaborn-v0_8-darkgrid',
 'seaborn-v0_8-deep',
 'seaborn-v0_8-muted',
 'seaborn-v0_8-notebook',
```

```
 'seaborn-v0_8-paper',
 'seaborn-v0_8-pastel',
 'seaborn-v0_8-poster',
 'seaborn-v0_8-talk',
 'seaborn-v0_8-ticks',
 'seaborn-v0_8-white',
 'seaborn-v0_8-whitegrid',
 'tableau-colorblind10']
```

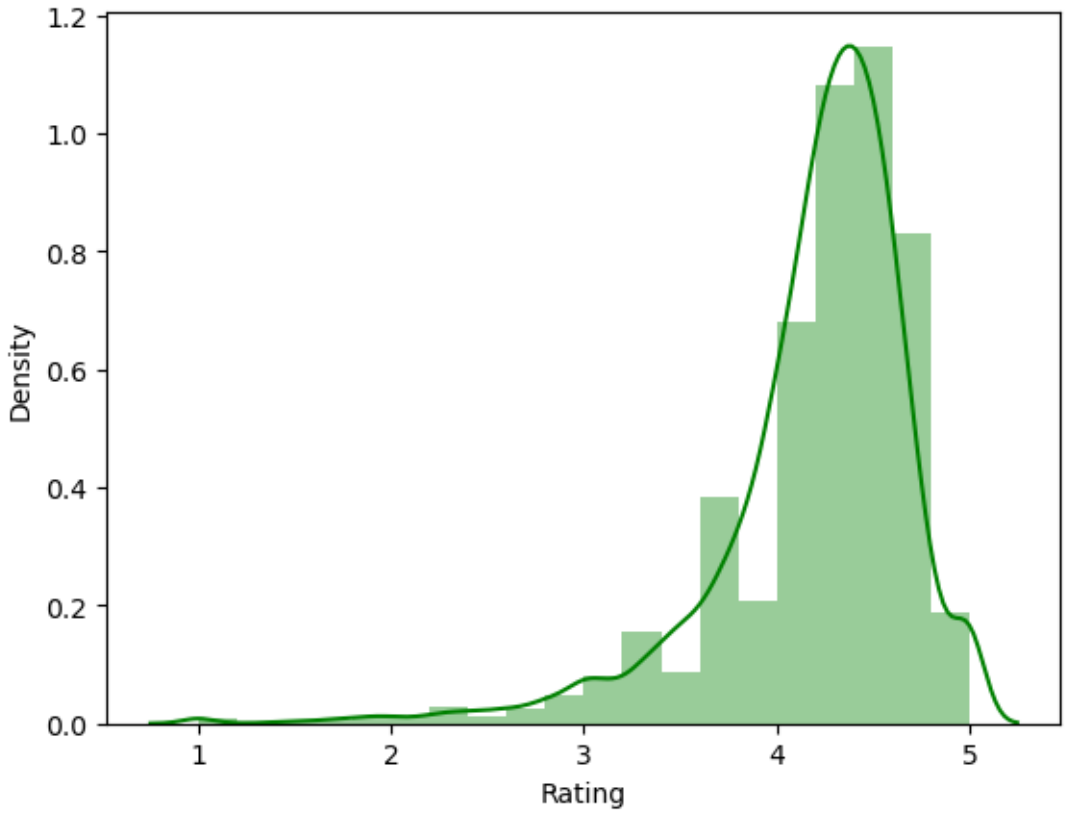
[147]:
```
sns.distplot(data.Rating)
plt.show()
```



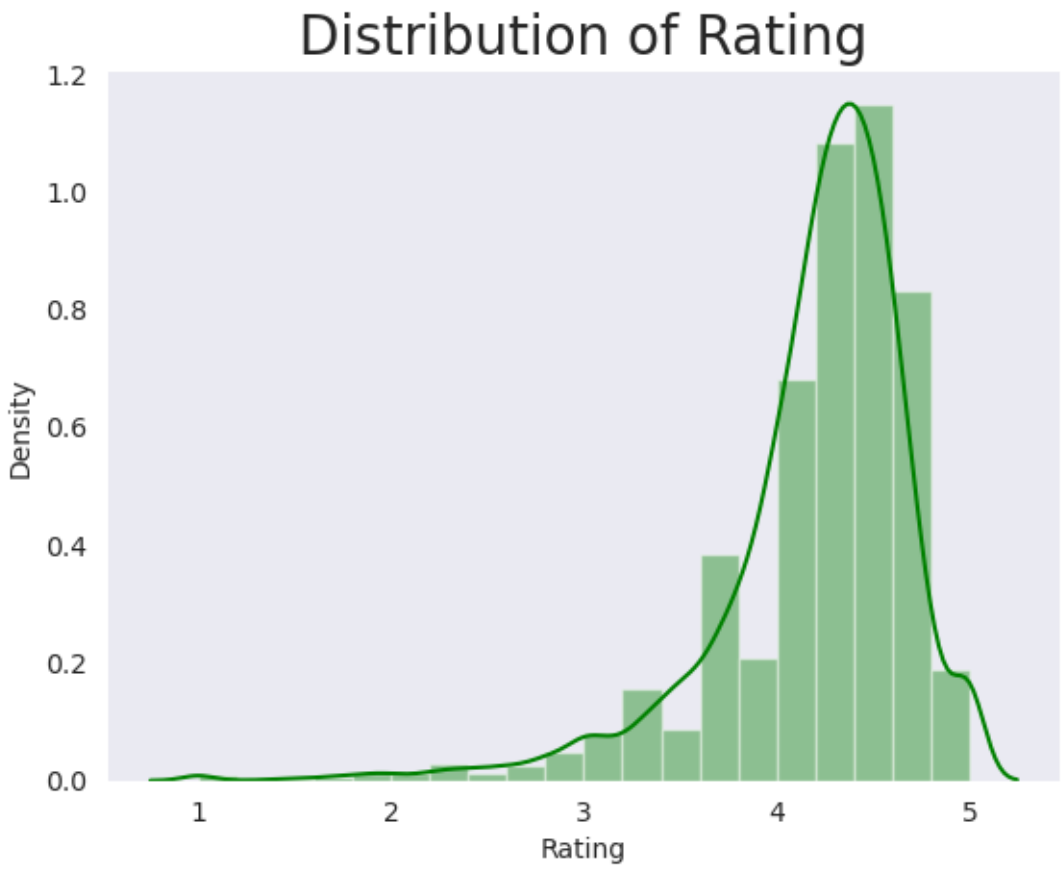
[149]:
```
sns.distplot(data.Rating, bins = 20)
plt.show()
```



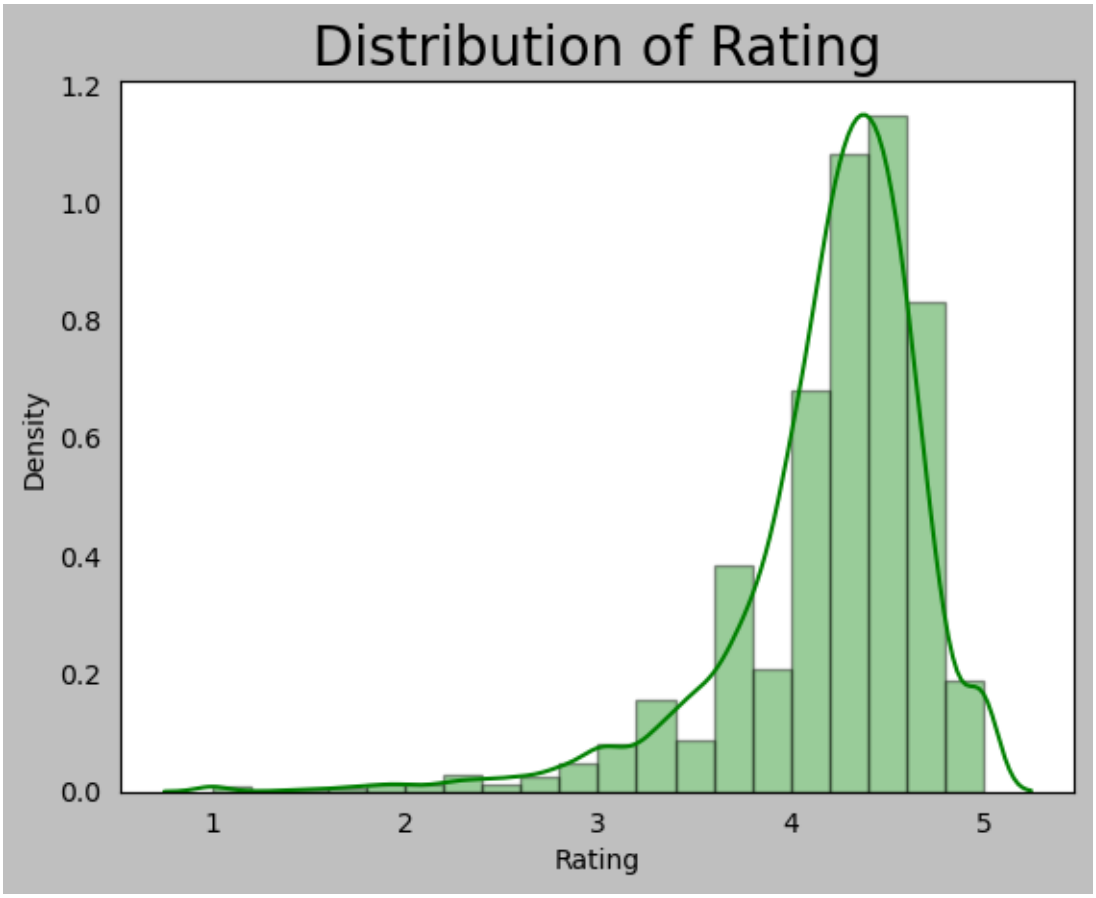
[151]:
```
sns.distplot(data.Rating, bins = 20, color = 'g')
plt.show()
```



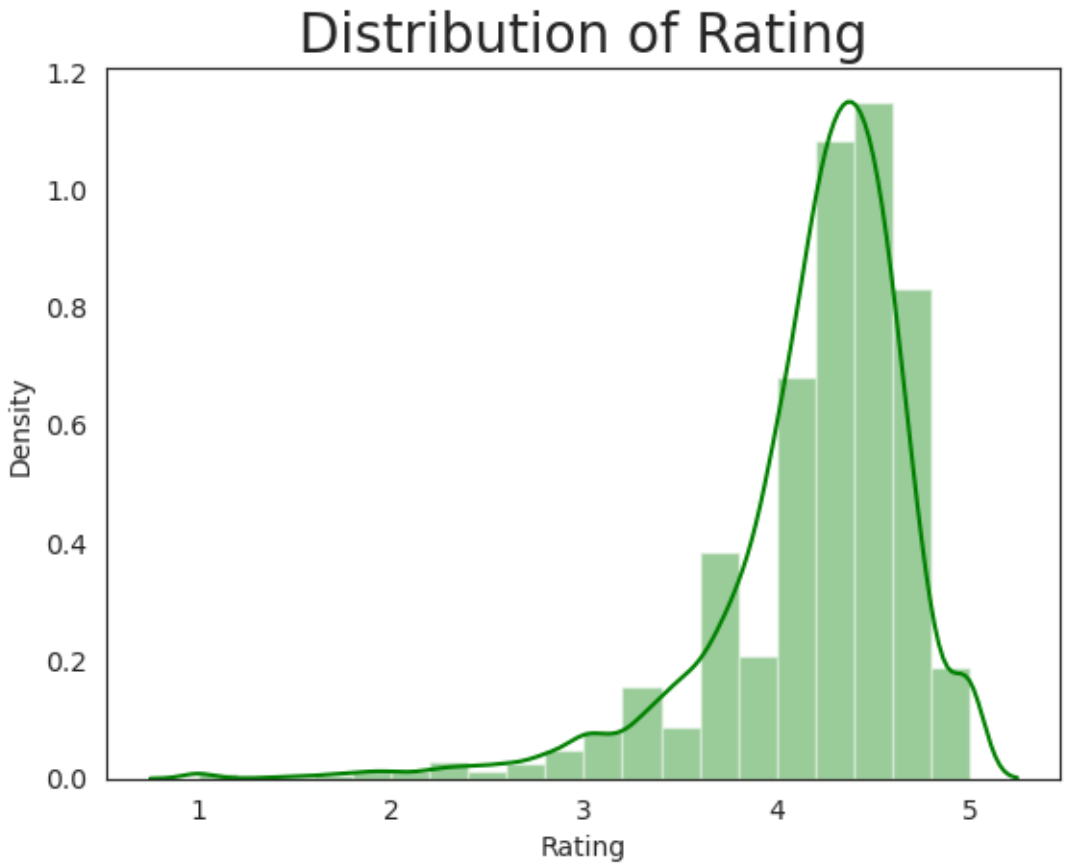
[153]:
```
sns.set_style
sns.set_style("dark")
sns.distplot(data.Rating, bins = 20, color = 'g')
plt.title("Distribution of Rating", fontsize = 20)
plt.show()
```



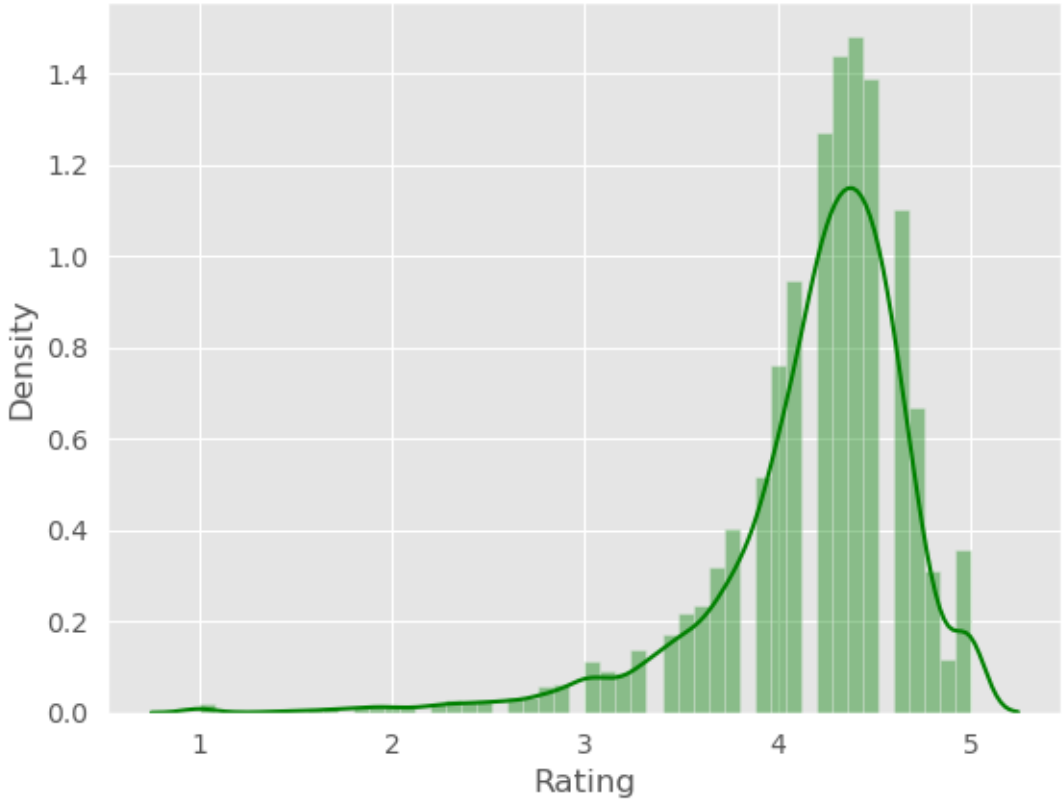
[157]:
```
plt.style.use("grayscale")
sns.distplot(data.Rating, bins = 20, color = 'g')
plt.title("Distribution of Rating", fontsize = 20)
plt.show()
```



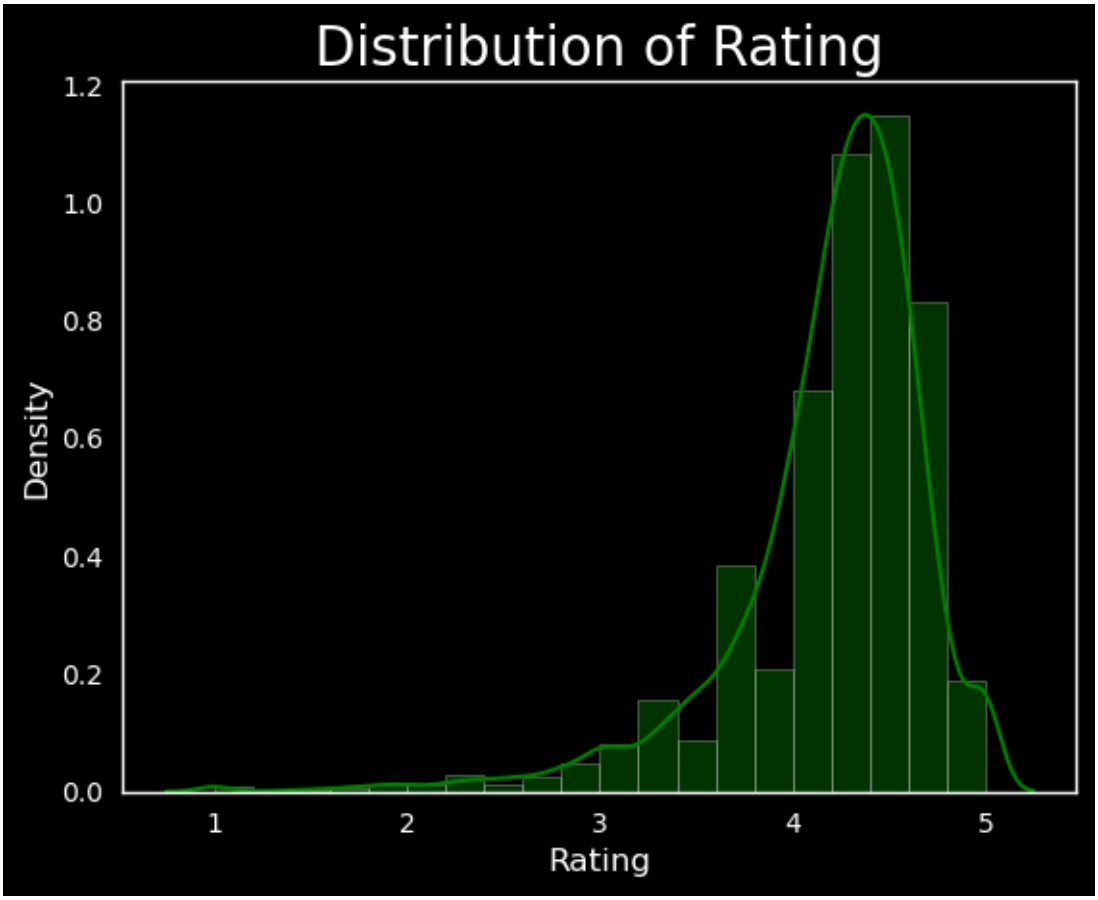
[161]:
```
plt.style.use("grayscale")
sns.set_style("white")
sns.distplot(data.Rating, bins = 20, color = 'g')
plt.title("Distribution of Rating", fontsize = 20)
plt.show()
```



[171]:
```
plt.style.use("ggplot")
#sns.set_style("white")
sns.distplot(data.Rating, color = 'g')
plt.title("Distribution of Rating", fontsize = 20)
plt.show()
```



[169]:
```
plt.style.use("dark_background")

sns.distplot(data.Rating, bins = 20,color = 'g')
plt.title("Distribution of Rating", fontsize = 20)
plt.show()
```



[175]:
```
data["Content Rating"].value_counts()
```

[175]:
```
Everyone           7414
Teen               1083
Mature 17+          461
Everyone 10+        397
Adults only 18+       3
Unrated               1
Name: Content Rating, dtype: int64
```

[177]:
```
data = data[-data["Content Rating"].isin(["Adults only 18+","Unrated"])]
```

[179]:
```
data["Content Rating"].value_counts()
```

[179]:
```
Everyone        7414
Teen            1083
Mature 17+       461
Everyone 10+     397
Name: Content Rating, dtype: int64
```

[181]:
```
data.shape
```

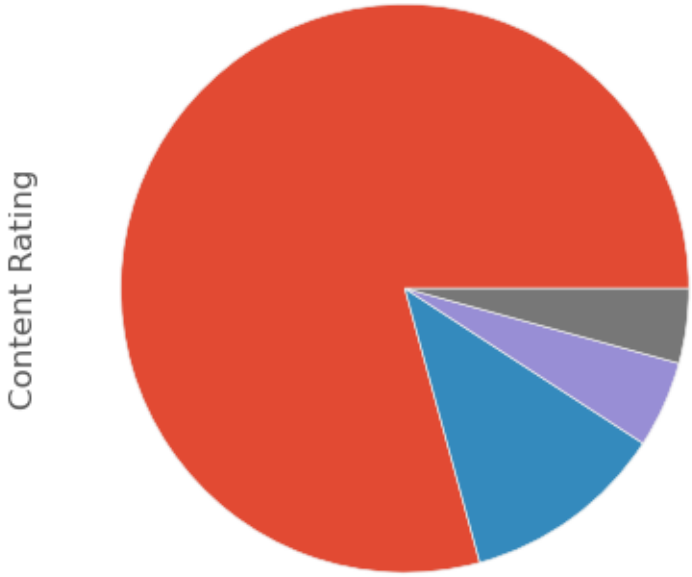
[181]:
```
(9355, 13)
```

[183]:
```
data.reset_index(inplace = True, drop = True)
```

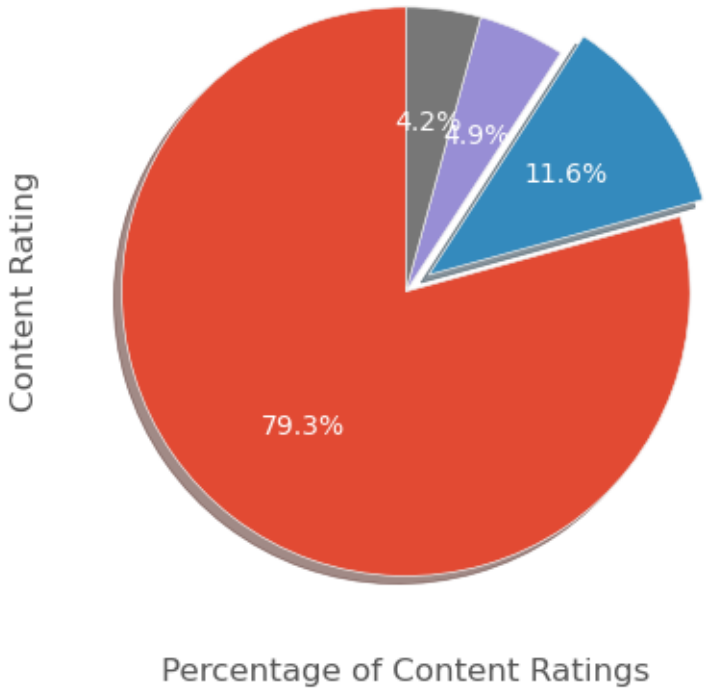
[187]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9355 entries, 0 to 9354
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   App             9355 non-null   object
 1   Category        9355 non-null   object
 2   Rating          9355 non-null   float64
 3   Reviews         9355 non-null   int32
 4   Size            9355 non-null   float64
 5   Installs        9355 non-null   int64
 6   Type            9355 non-null   object
 7   Price           9355 non-null   float64
 8   Content Rating  9355 non-null   object
 9   Genres          9355 non-null   object
 10  Last Updated    9355 non-null   object
 11  Current Ver     9355 non-null   object
 12  Android Ver     9355 non-null   object
dtypes: float64(3), int32(1), int64(1), object(8)
memory usage: 913.7+ KB
```

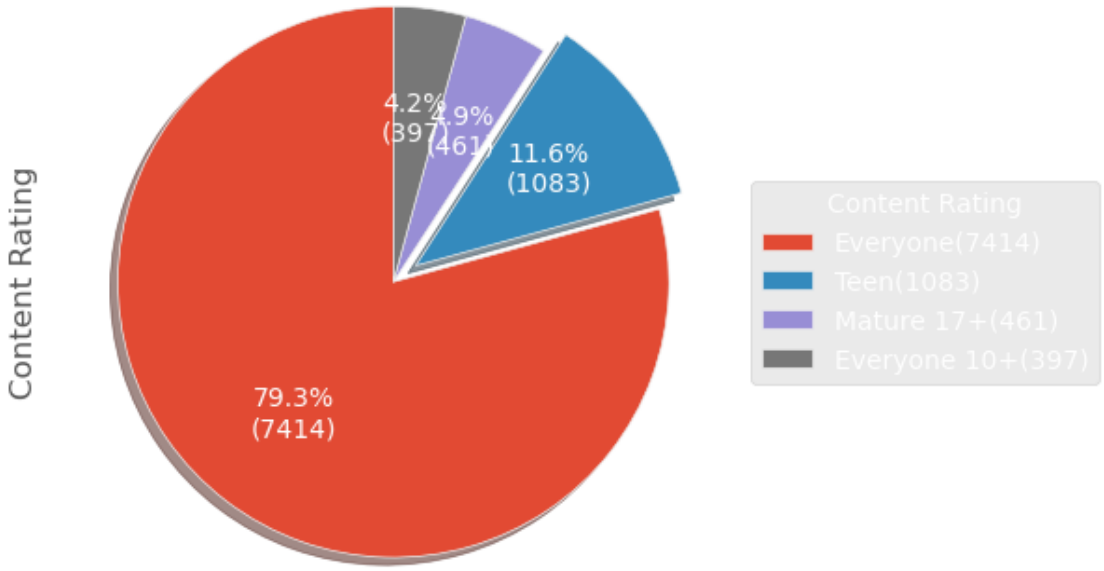
[199]:
```
data["Content Rating"].value_counts().plot.pie()
plt.show()
```



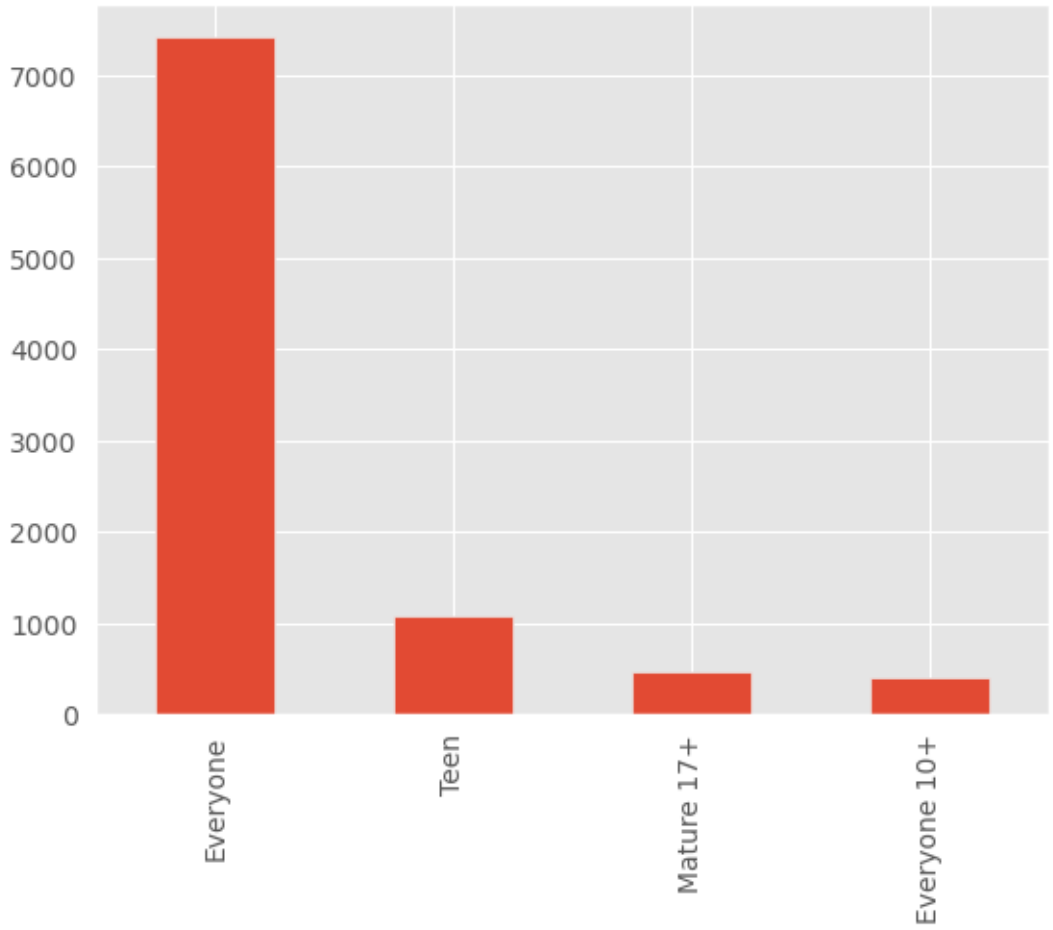
[205]:
```
data["Content Rating"].value_counts().plot.pie(labels = data["Content Rating"].
 ↪value_counts().index,
                                                autopct = '%1.1f%%',
                                                explode = (0,0.1,0,0),
                                                startangle = 90,
                                                shadow = True)
plt.xlabel("Percentage of Content Ratings")

plt.show()
```



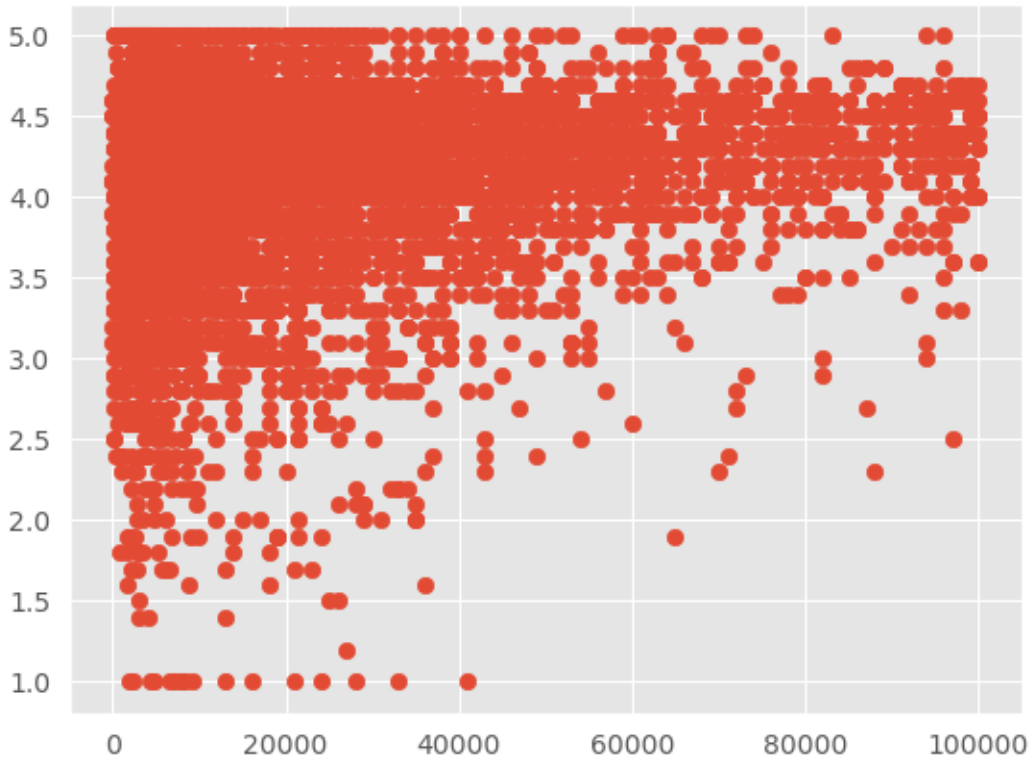
[215]:
```
values = data["Content Rating"].value_counts()
total = values.sum()

def auctpct_format(pct):
    absolute = int(round(pct/100 * total))
    return f'{pct:.1f}%\n({absolute})'

values.plot.pie(
    labels = values.index,
    autopct = auctpct_format,
    explode = (0,0.1,0,0),
    startangle = 90,
    shadow = True)

plt.legend(
    labels = [f'{label}({count})' for label, count in zip(values.index,values)],
    title = 'Content Rating',
    loc = 'center left',
    bbox_to_anchor = (1,0.5)
)

plt.show()
```



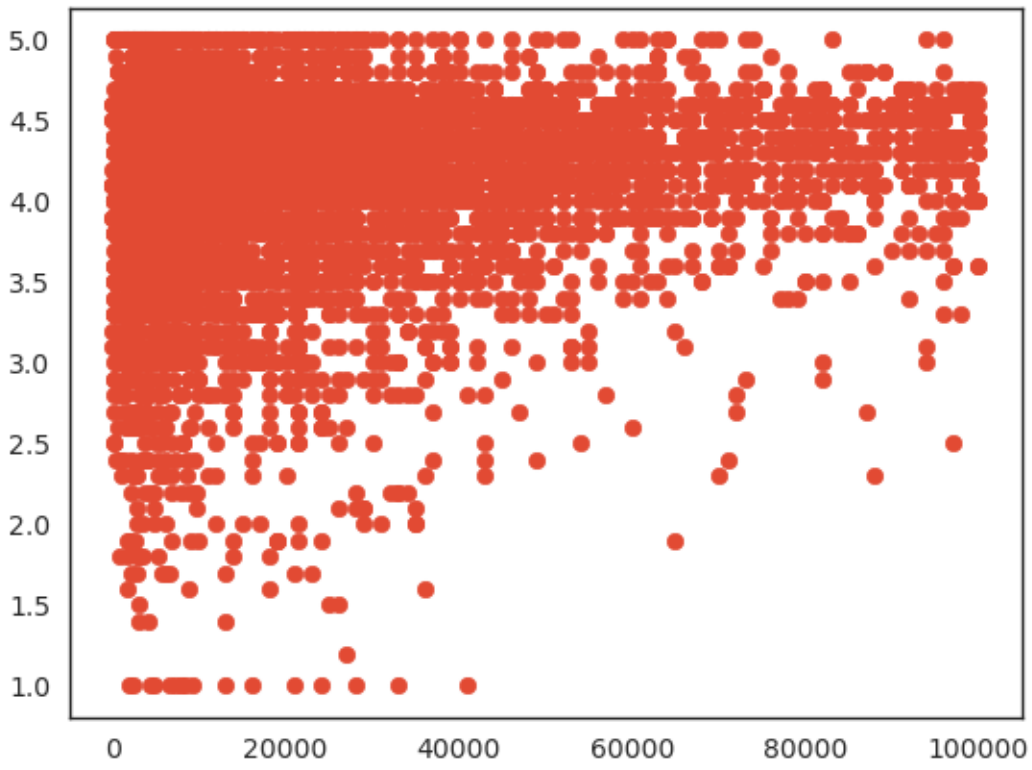
[191]:
```
data["Content Rating"].value_counts().plot.bar()
plt.show()
```



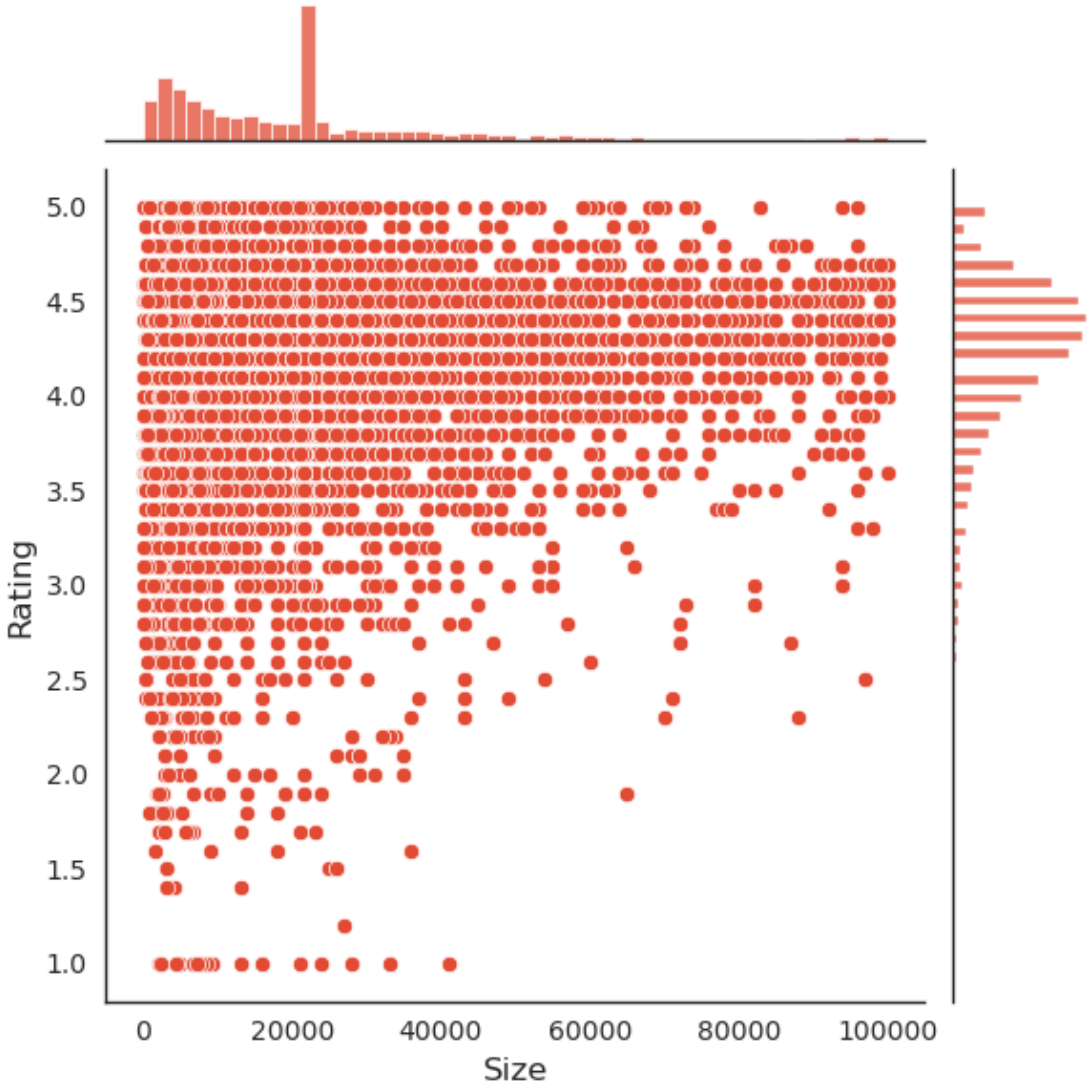
[217]:
```
# Bivariate Analysis

plt.scatter(data.Size, data.Rating)
plt.show()
```



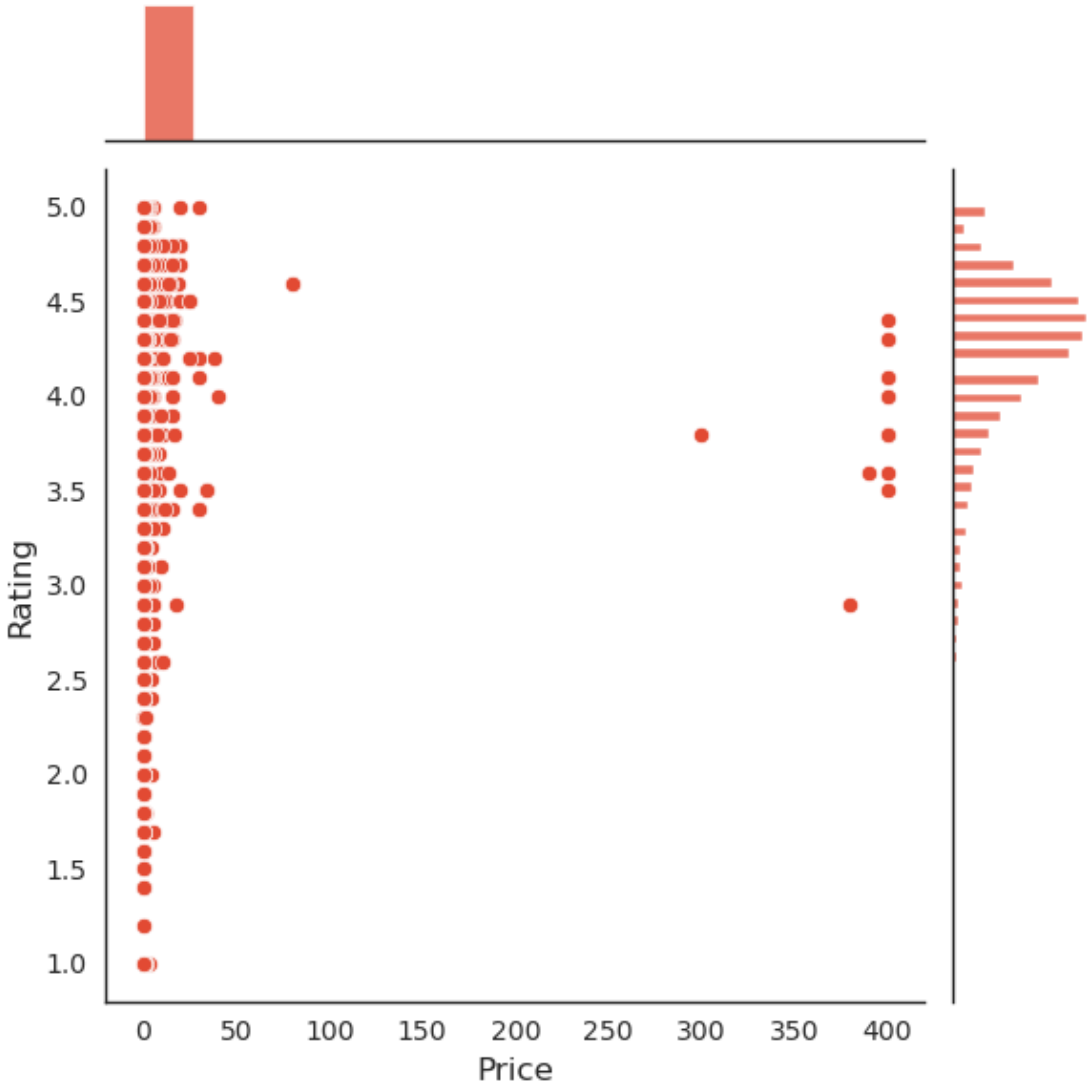
[219]:
```
sns.set_style("white")

plt.scatter(data.Size, data.Rating)
plt.show()
```



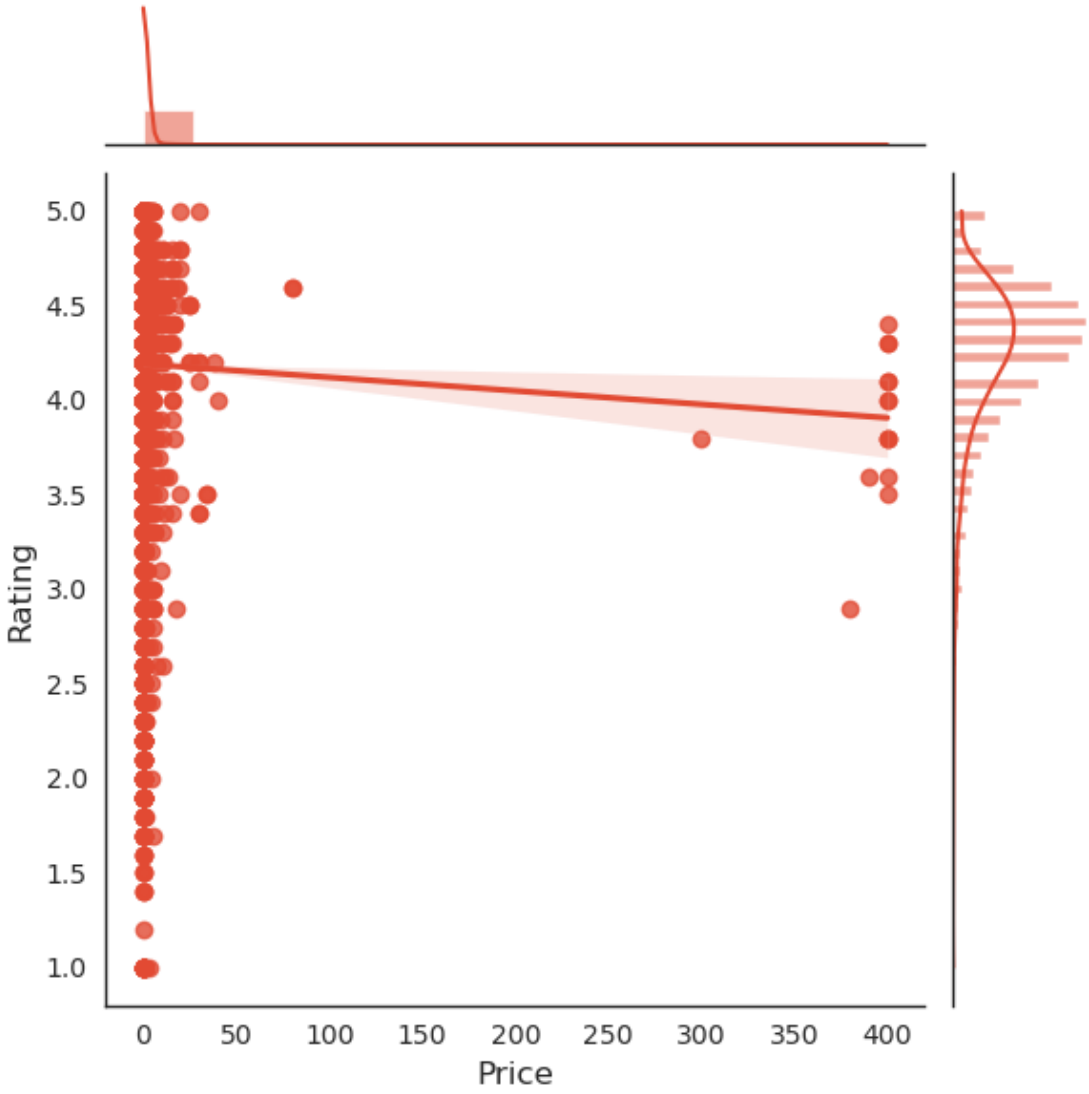
[221]:
```
sns.jointplot(x= 'Size', y = 'Rating', data = data)
plt.show()
```



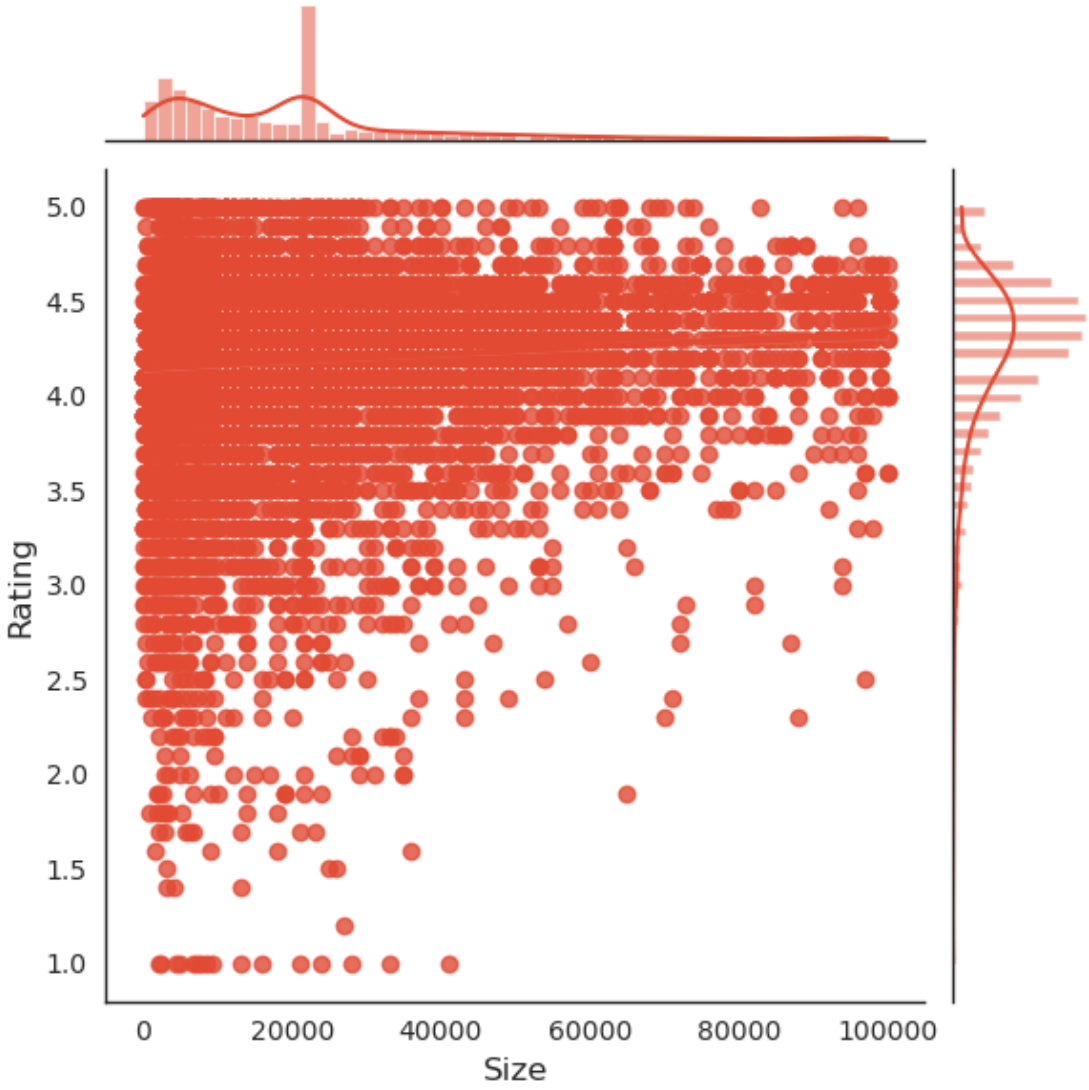
[223]:
```
sns.jointplot(x= 'Price', y = 'Rating', data = data)
plt.show()
```



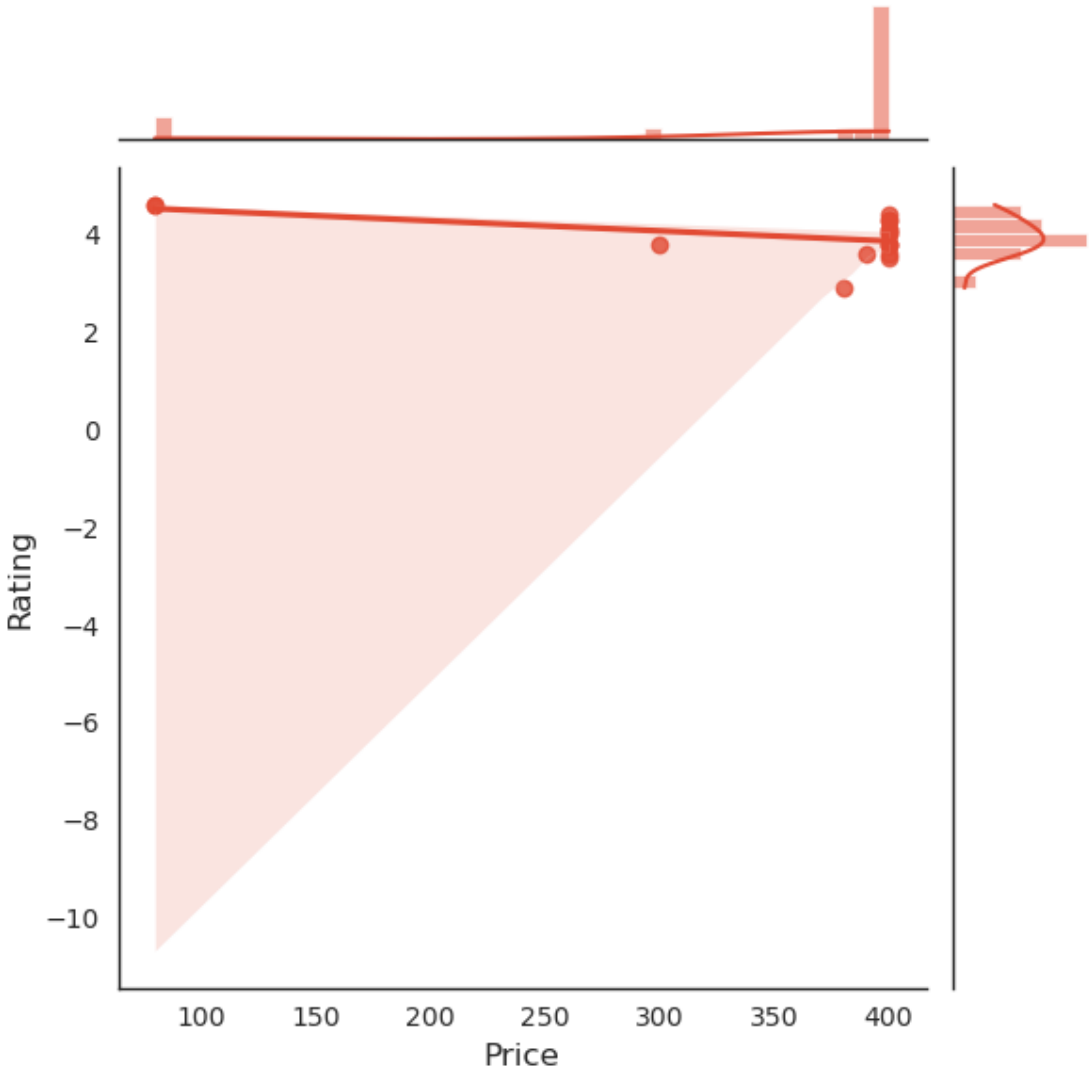
[225]:
```
sns.jointplot(x= 'Price', y = 'Rating', data = data, kind = 'reg')
plt.show()
```



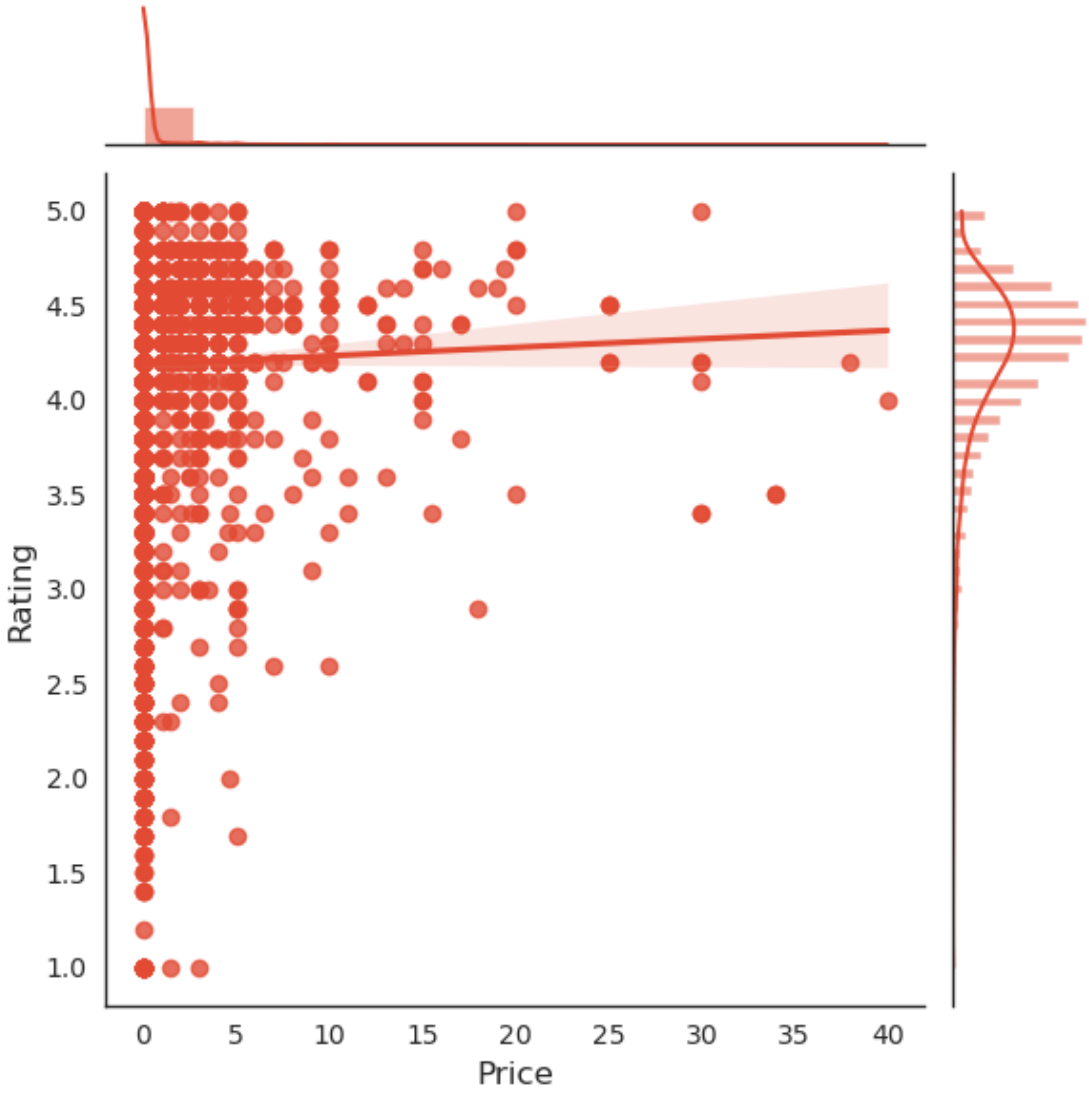
[227]:
```
sns.jointplot(x= 'Size', y = 'Rating', data = data, kind = 'reg')
plt.show()
```



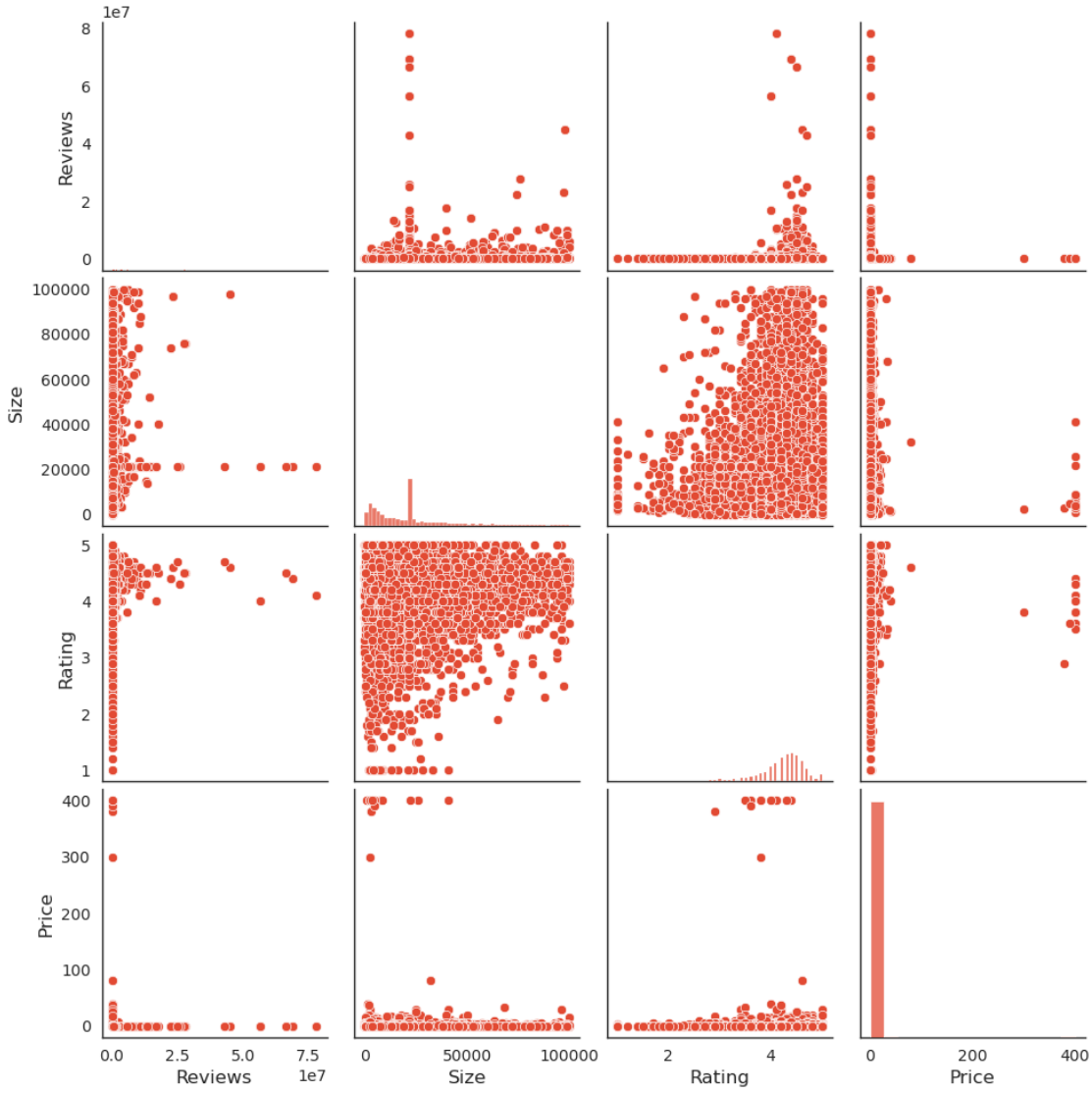
[229]:
```
sns.jointplot(x= 'Price', y = 'Rating', data = data[data.Price>50], kind =␣
↪'reg')
plt.show()
```



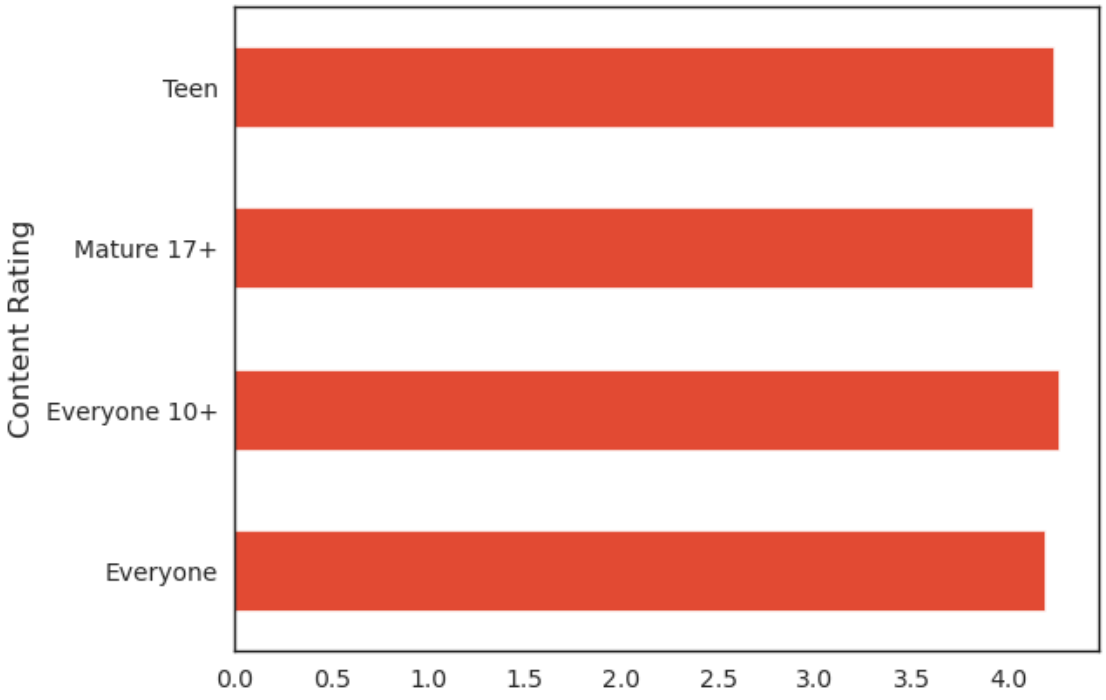
[231]:
```
sns.jointplot(x= 'Price', y = 'Rating', data = data[data.Price<50], kind =␣
↪'reg')
plt.show()
```



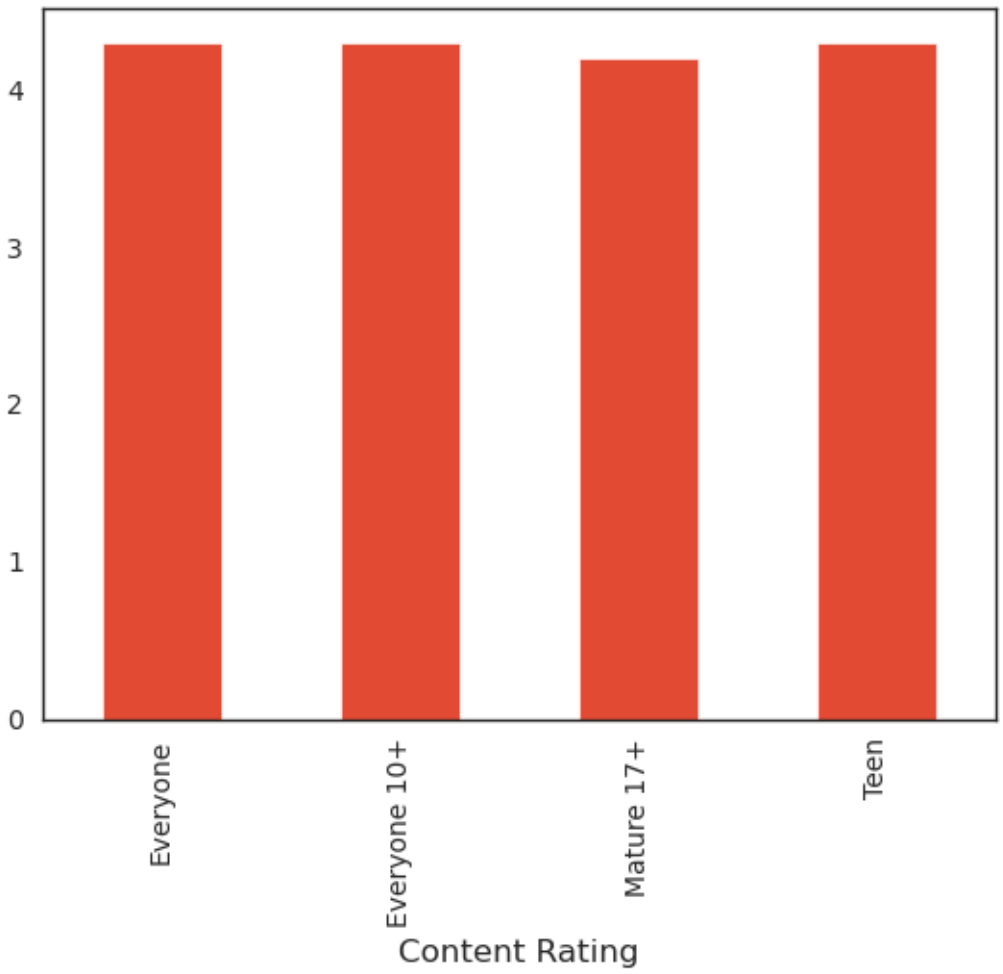
[233]:
```
# Pairplots
sns.pairplot(data[["Reviews", "Size","Rating","Price"]])
plt.show()
```



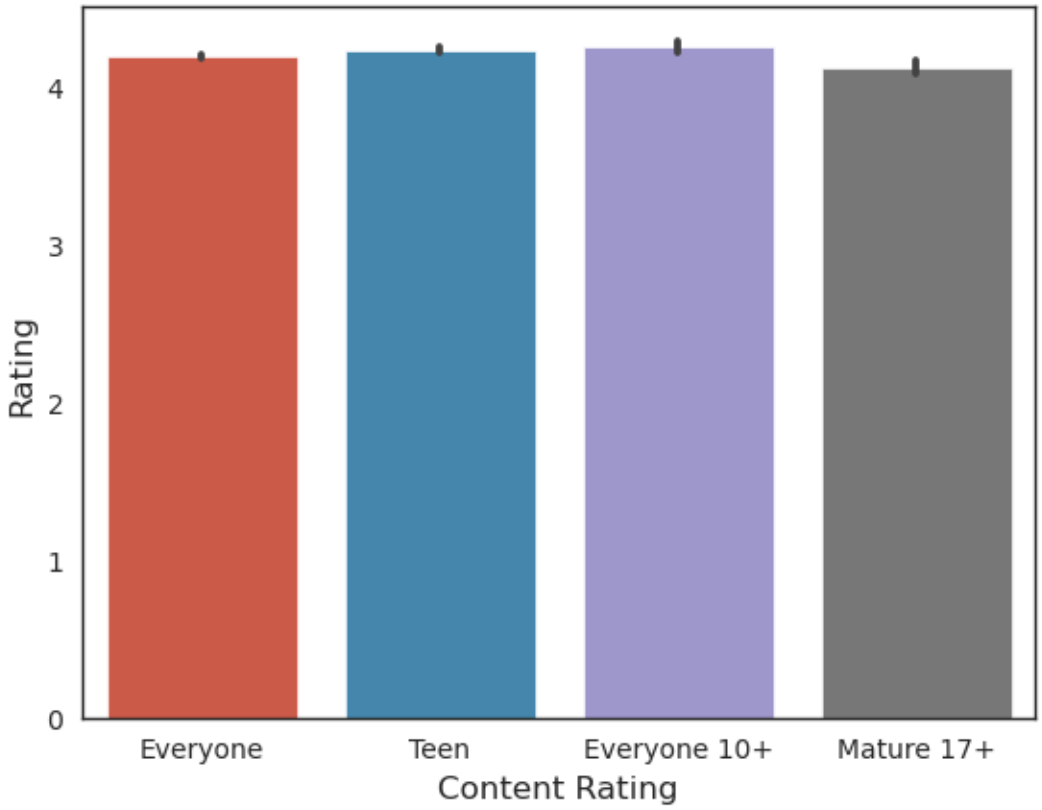
[235]:
```
data.groupby(["Content Rating"])["Rating"].mean().plot.barh()
plt.show()
```



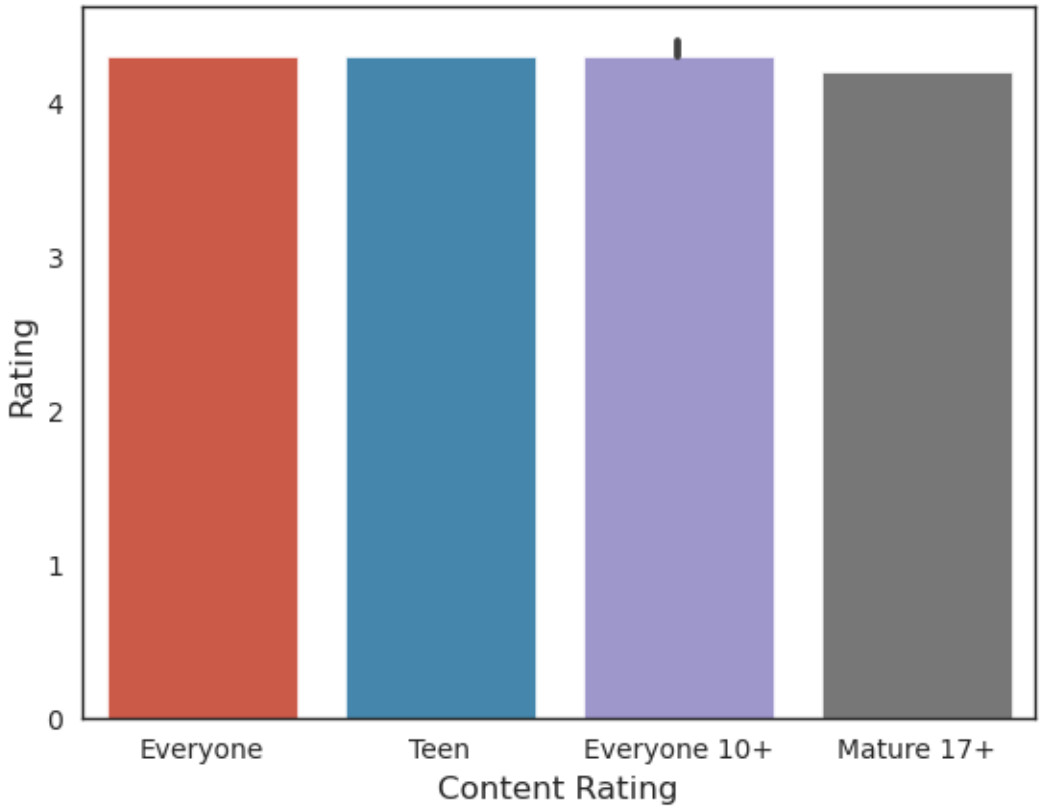
[241]:
```
data.groupby(["Content Rating"])["Rating"].median().plot.bar()
plt.show()
```



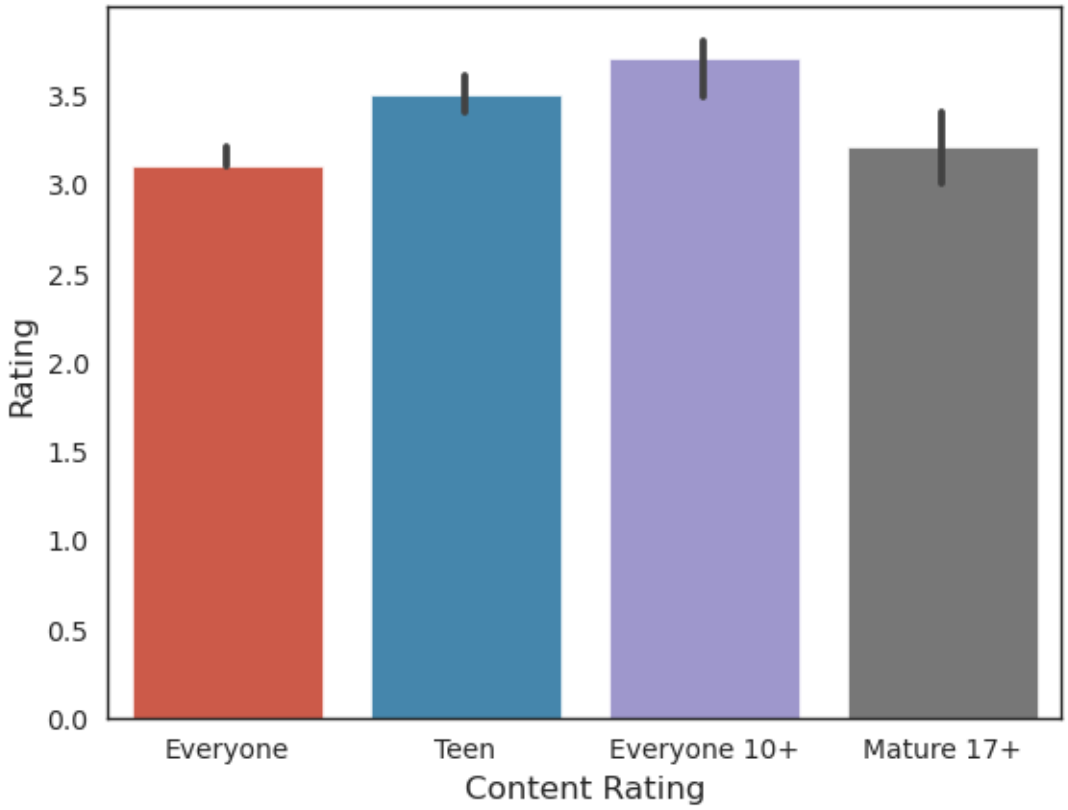
[239]:
```
sns.barplot(data = data , x = "Content Rating", y = "Rating")
plt.show()
```



[245]:
```
sns.barplot(data = data , x = "Content Rating", y = "Rating", estimator = np.
  ↪median)
plt.show()
```



[247]:
```
sns.barplot(data = data , x = "Content Rating", y = "Rating", estimator =␣
 ↪lambda x : np.quantile(x,0.05))
plt.show()
```


3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Rating
Everyone
Teen
Everyone 10+
Mature 17+
Content Rating

[ ]: