```
In [1]: #calling all the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

1. Loading & Checking the Data

```
In [3]: #loading data
df=pd.read_csv(r"C:\Users\USER\Downloads\App Rating Prediction @ Python\googleplaystore.csv")
df.head()
```

Out[3]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	1.0.0	4.0.3 and up
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0	4.0.3 and up
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	1.2.4	4.0.3 and up
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	June 8, 2018	Varies with device	4.2 and up
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.4 and up

```
In [4]: #to find out the no. of rows & columns
df.shape

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

```
In [5]: #to find the not null values & data type for each column
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 [6]: #description of the data in the DataFrame
df.describe()
```

Out[6]:

	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

1. Checking for Null Values: Identifying and counting null values in each column.

```
In [7]: # Checking for null values
null_counts = df.isnull().sum()
print(null_counts)

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. Dropping Records with Null Values: Remove records with null values in any column.

```
In [20]: # Droping records with null values
df.dropna(inplace=True)
null_counts = df.isnull().sum()
print(null_counts)

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
```

1. Fixing Variable Types and Formatting:

```
In [10]: # Size column: Convert to numeric
df['Size'] = df['Size'].apply(lambda x: str(x).replace('M', '') if 'M' in str(x) else x)
df['Size'] = df['Size'].apply(lambda x: str(x).replace('Varies with device', 'nan') if 'Varies with device' in str(x) else x)

# Scaling all the values to Millions format (means that 19.0 => 19x10^6 => 19M)
df['Size'] = df['Size'].apply(lambda x: float(str(x).replace('k', ''))/1000 if 'k' in str(x) else x)
df['Size'] = df['Size'].apply(lambda x : float(x))
df = df[pd.notnull(df['Size'])]
df['Size'].dtype

Out[10]: dtype('float64')
```

```
In [14]: # Reviews column: Convert to numeric
df['Reviews'] = pd.to_numeric(df['Reviews'], errors='coerce')
df['Reviews'].dtype

Out[14]: dtype('int64')
```

```
In [18]: # Installs column: Convert to integer
df['Installs'] = df['Installs'].apply (lambda x: str(x).replace('+','') if '+' in str(x) else x)
df['Installs'] = df['Installs'].apply(lambda x: str(x).replace(',', '') if ',' in str(x) else x)
df['Installs'] = df['Installs'].apply(lambda x: int (x))
df['Installs'].dtype

Out[18]: dtype('int64')
```

```
In [21]: # Price column: Convert to numeric
df['Price'] = df['Price'].apply(lambda x: float(x.strip('$')))
df['Price'].dtype

Out[21]: dtype('float64')
```

1. Sanity Checking

```
In [24]: # Checking rows with rating outside the range [1, 5]
df['Rating'].unique()
df[df['Rating']>5].shape[0]

Out[24]: 0
```

```
In [25]: # checking rows where reviews are greater than installs
df[df['Reviews']>df['Installs']].shape[0]

Out[25]: 6
```

```
In [26]: # droping rows where reviews are greater than installs
df.drop(df[df['Reviews']>df['Installs']].index, inplace = True)
```

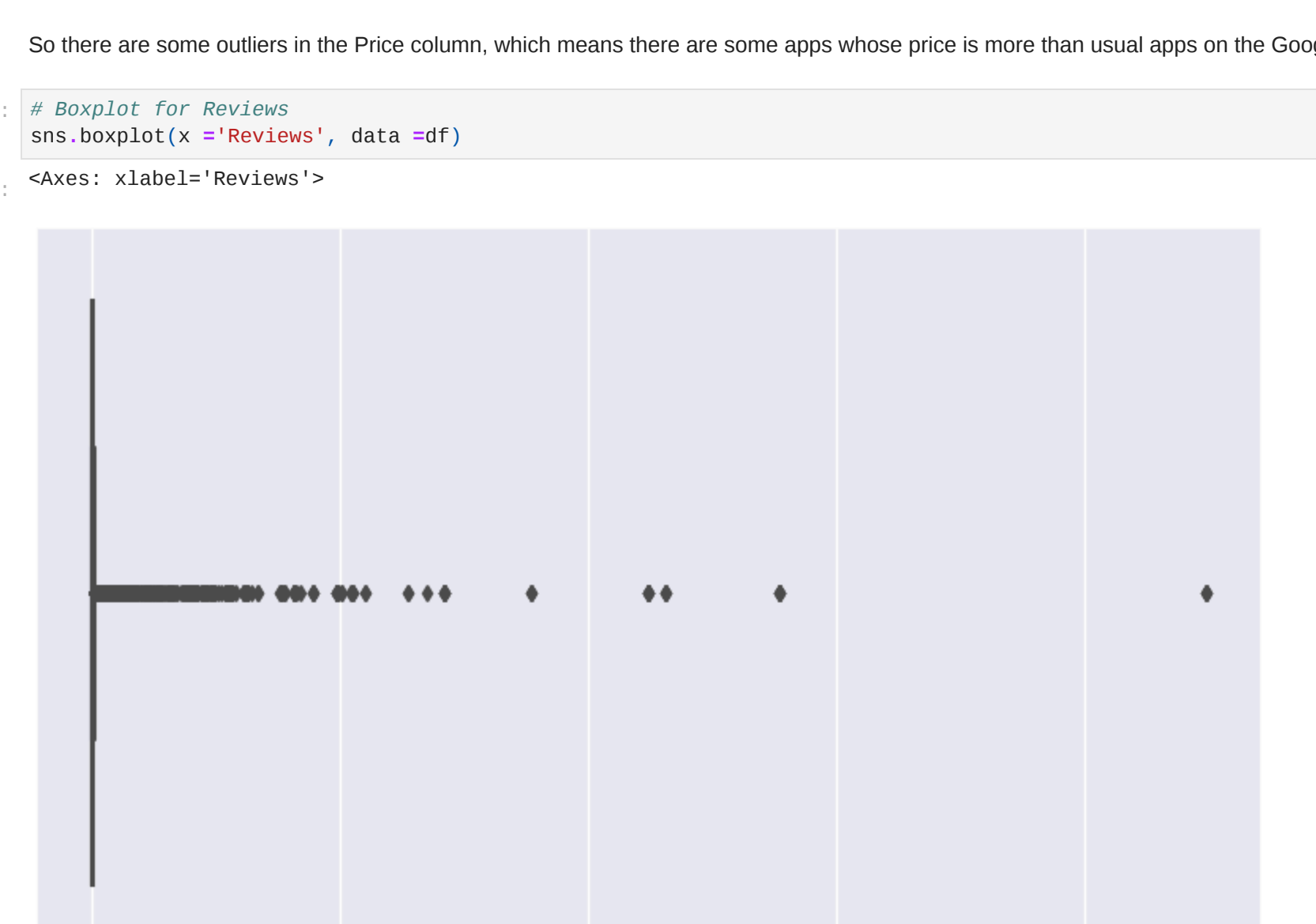
```
In [27]: # For free apps, price should be 0
df[(df['Type']=='free')& (df['Price'] ==0)].shape[0]

Out[27]: 0
```

1. Performing Univariate Analysis:

```
In [29]: # Boxplot for Price
sns.set(rc={'figure.figsize':(10,6)})
sns.boxplot(x= 'Price',data= df)

Out[29]: <Axes: xlabel='Price'>
```

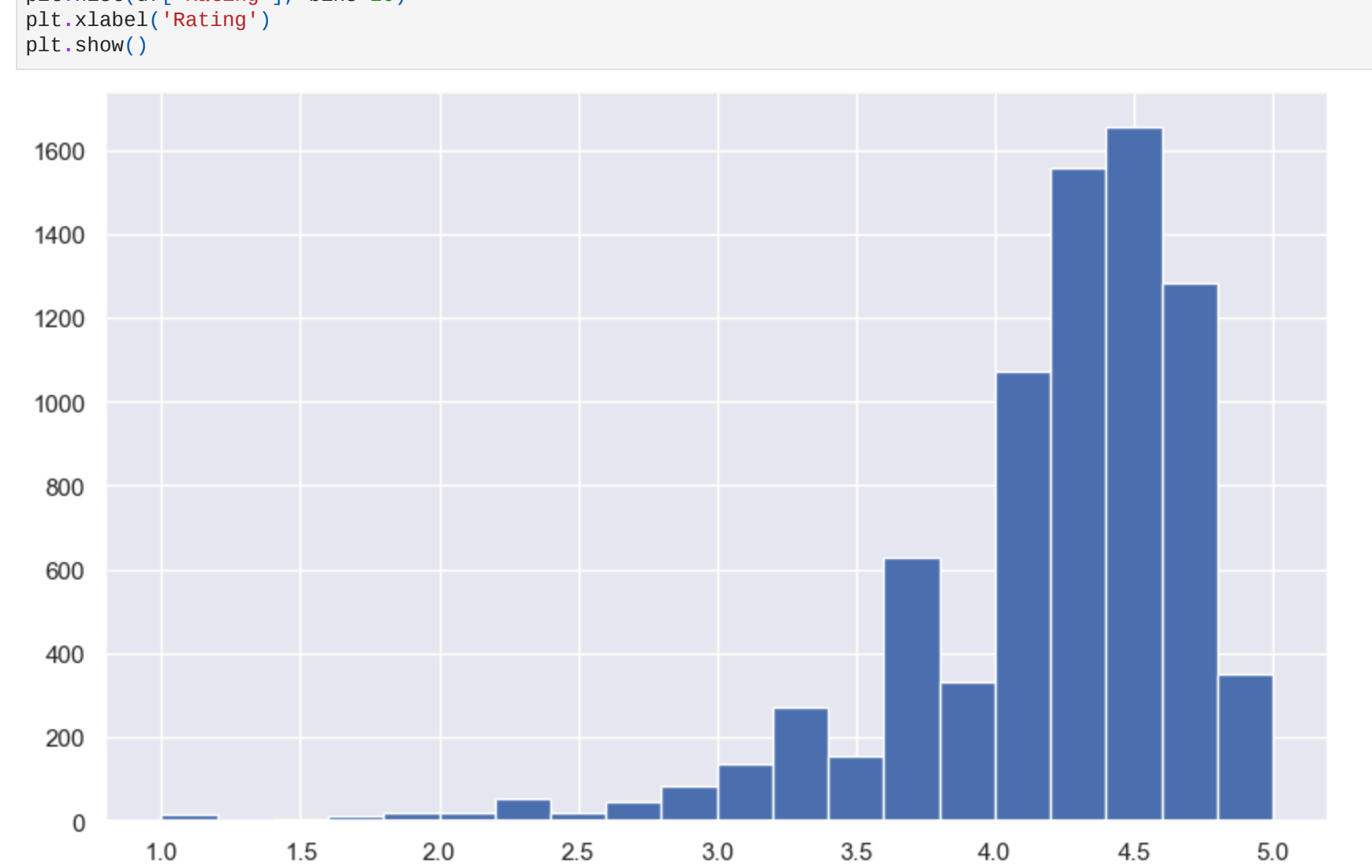
So there are some outliers in the Price column, which means there are some apps whose price is more than usual apps on the Googleplaystore.

```
In [30]: # Boxplot for Reviews
sns.boxplot(x ='Reviews', data =df)

Out[30]: <Axes: xlabel='Reviews'>
```



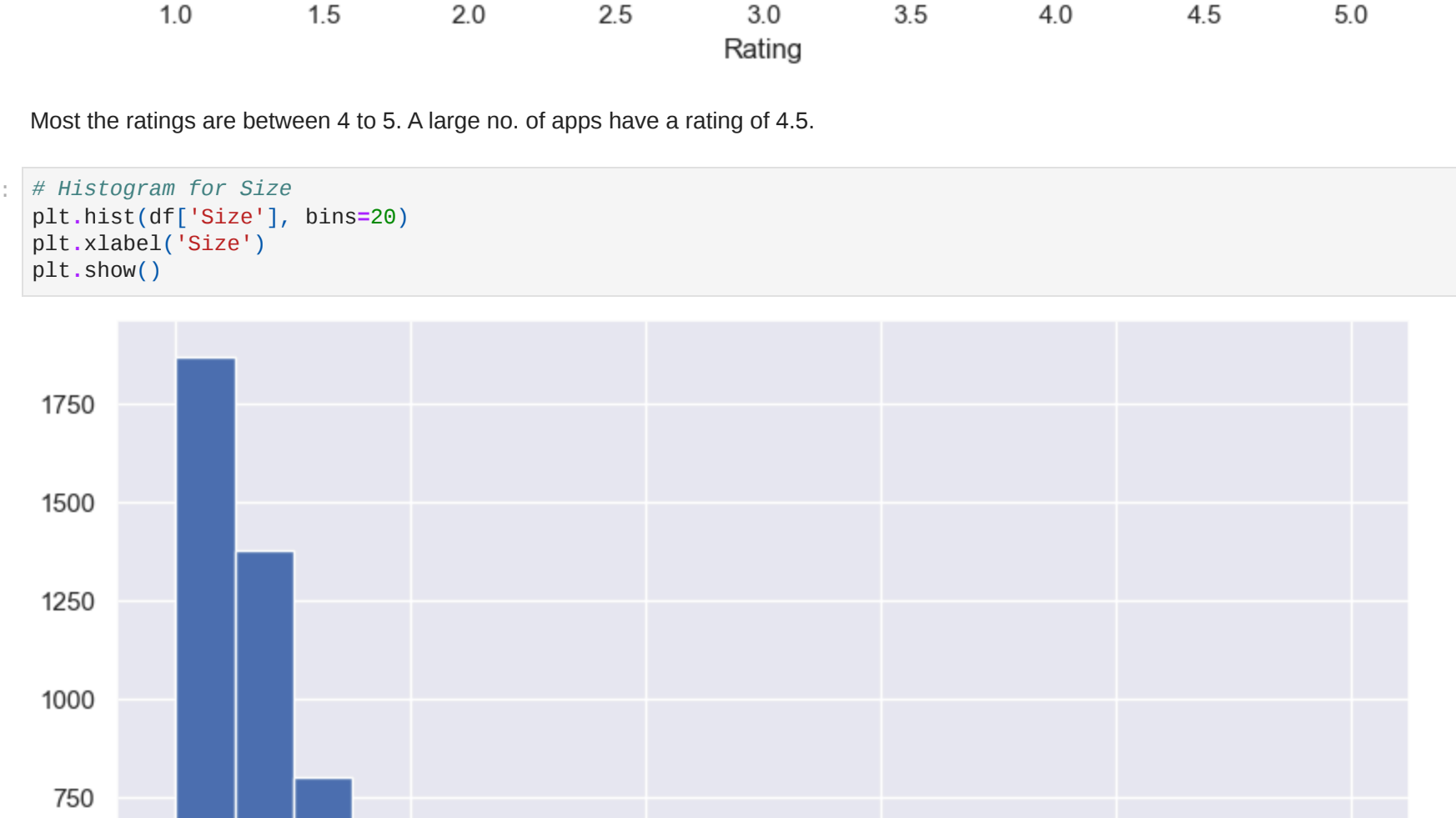
So there are some apps that have a very high number of Reviews.

```
In [31]: # Histogram for Rating
plt.hist(df['Rating'], bins=20)
plt.xlabel('Rating')
plt.show()
```



Most the ratings are between 4 to 5. A large no. of apps have a rating of 4.5.

```
In [32]: # Histogram for Size
plt.hist(df['Size'], bins=20)
plt.xlabel('Size')
plt.show()
```




```
In [ ]: Most of the apps takes very less memory space to install.
```

1. Outlier Treatment

```
In [33]: # Price outliers: Checking if price of $200 and above for an application
df[df['Price']>200].index.shape[0]

Out[33]: 15
```

```
In [34]: #Dropping these apps as they are Junk apps
df.drop(df[df['Price']>200].index, inplace= True)
```

```
In [36]: # Reviews outliers: Checking records having more than 2 million reviews
df.loc[df['Reviews']>2000000].shape[0]

Out[36]: 219
```

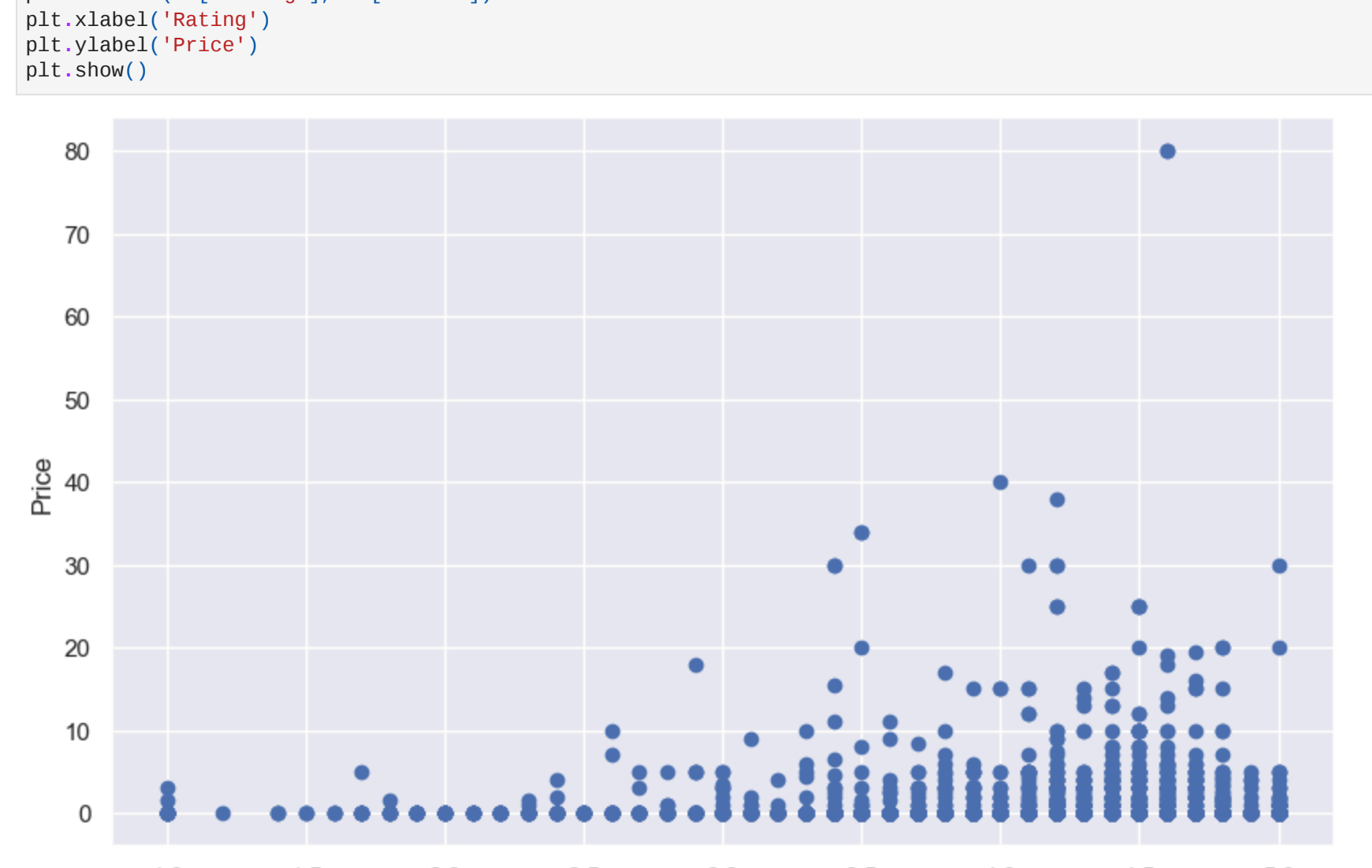
```
In [37]: #Dropping these apps as they these will skew the analysis
df.drop(df[df['Reviews']>2000000].index, inplace= True)
```

```
In [42]: # Installs outliers: Finding out the different percentiles
percentiles = df['Installs'].quantile([0.1, 0.25, 0.5, 0.7, 0.9, 0.95, 0.99])
print(percentiles)

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    50000000.0
Name: Installs, dtype: float64
```

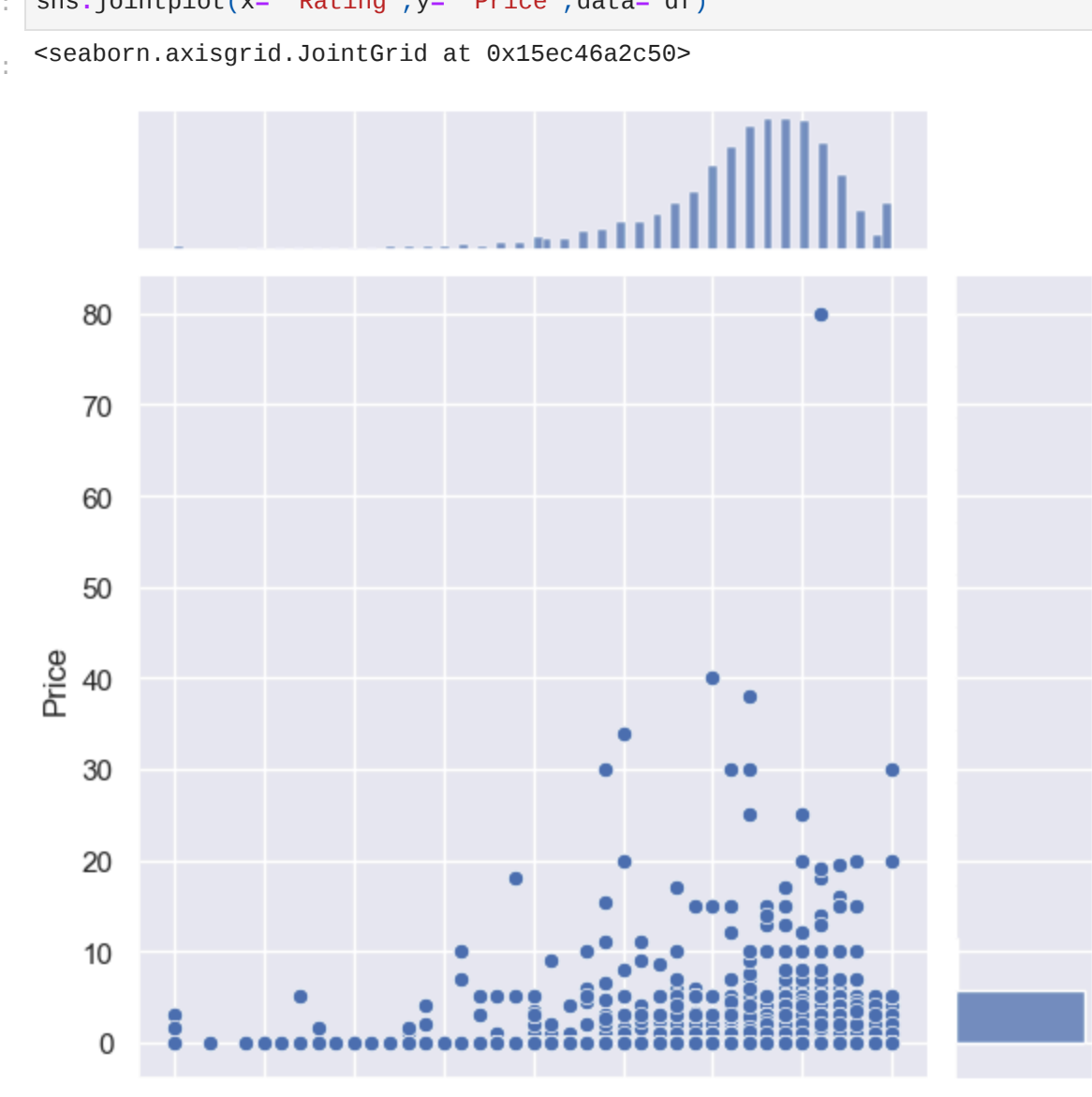
```
In [43]: #dropping the value more than the cutoff(threshold -95th percentile)
df.drop(df[df['Installs']>10000000].index, inplace= True)
```

1. Bivariate Analysis

```
In [46]: # Scatterplot/jointplot for Rating vs. Price
plt.scatter(df['Rating'], df['Price'])
plt.xlabel('Rating')
plt.ylabel('Price')
plt.show()
```

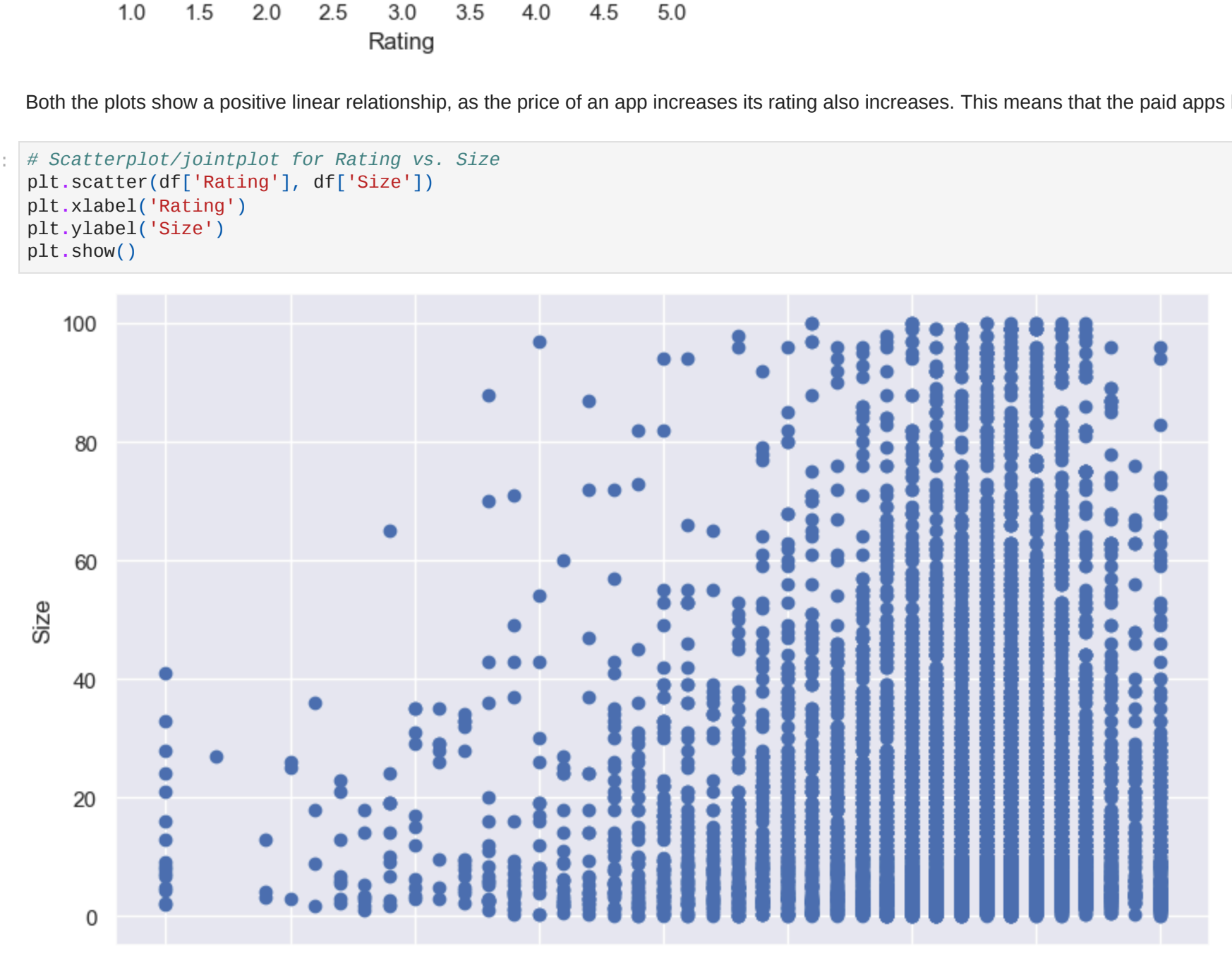



```
In [50]: sns.jointplot(x= 'Rating',y= 'Price',data= df)

Out[50]: <seaborn.axisgrid.JointGrid at 0x15ec46a2c50>
```

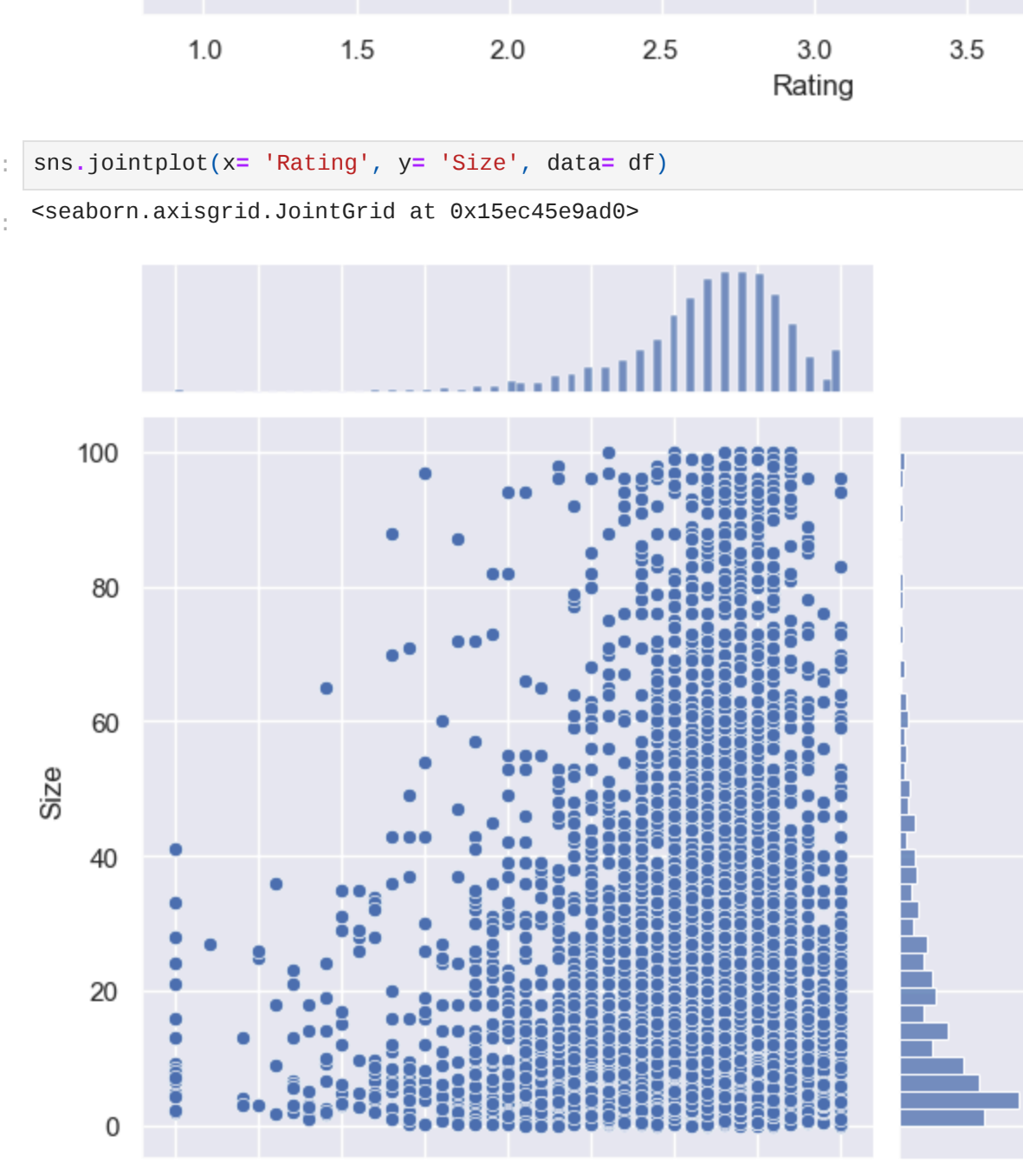


Both the plots show a positive linear relationship, as the price of an app increases its rating also increases. This means that the paid apps have the highest of Ratings.

```
In [49]: # Scatterplot/jointplot for Rating vs. Size
plt.scatter(df['Rating'], df['Size'])
plt.xlabel('Rating')
plt.ylabel('Size')
plt.show()
```




```
In [51]: sns.jointplot(x= 'Rating', y= 'Size', data= df)

Out[51]: <seaborn.axisgrid.JointGrid at 0x15ec45e9ad0>
```

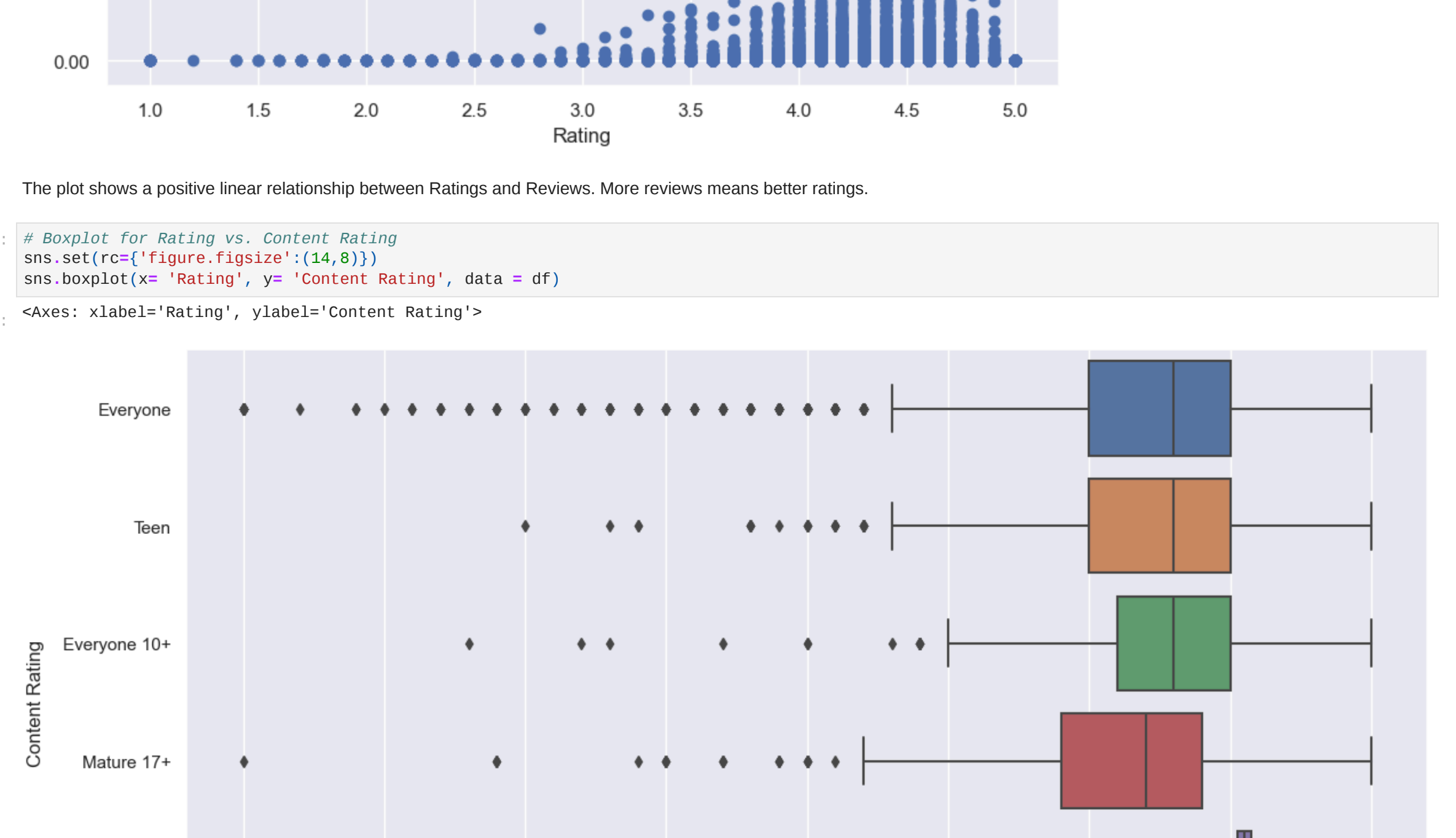


The plots show a positive linear relationship; as the Size increases the Ratings increases.

```
In [52]: #scatter plot for Rating vs. Reviews
plt.scatter(df['Rating'], df['Reviews'])
plt.xlabel('Rating')
plt.ylabel('Reviews')
plt.show()
```

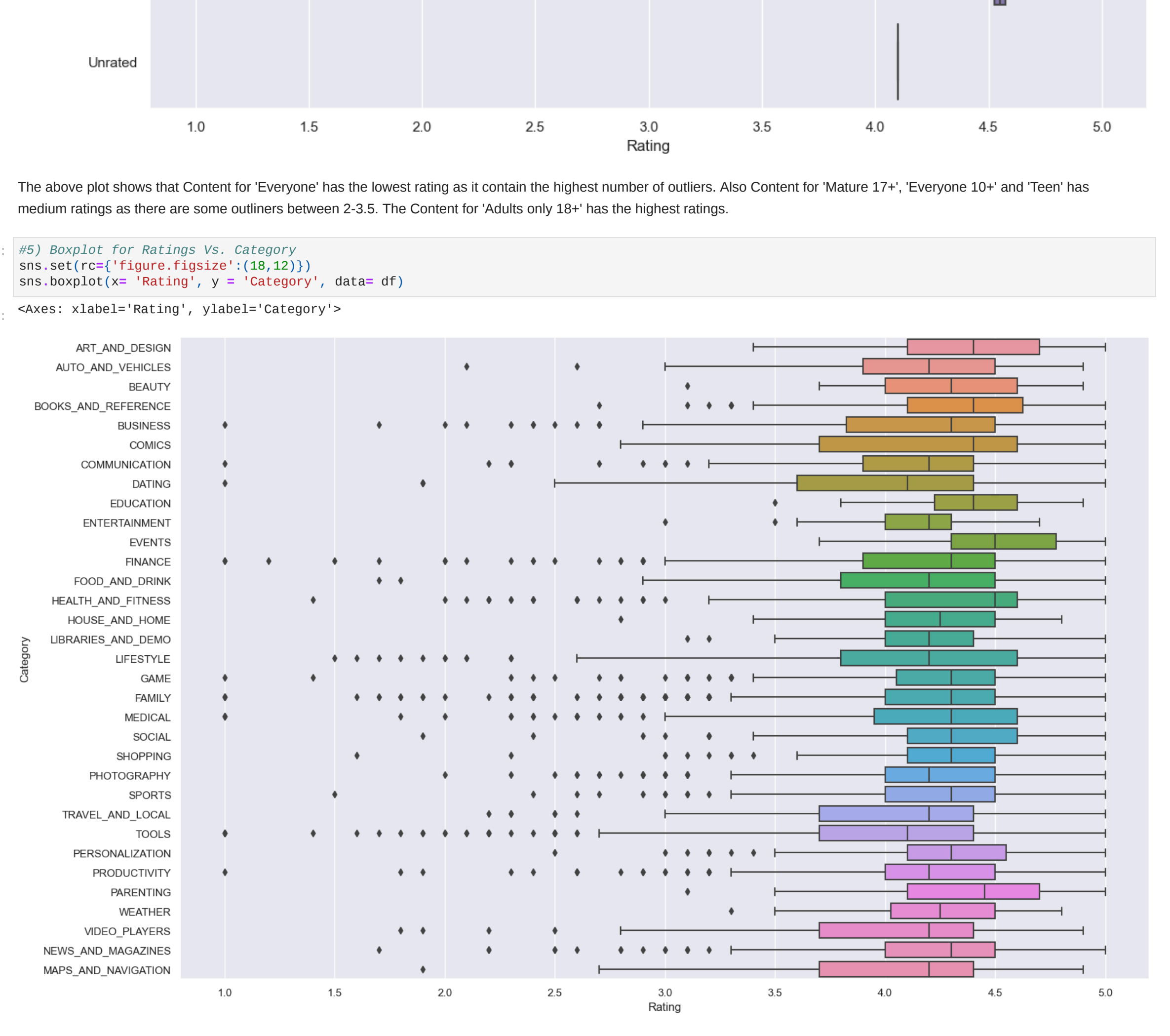
The plot shows a positive linear relationship between Ratings and Reviews. More reviews means better ratings.

```
In [54]: # Boxplot for Rating vs. Content Rating
sns.set(rc={'figure.figsize':(14,8)})
sns.boxplot(x= 'Rating', y= 'Content Rating', data = df)

Out[54]: <Axes: xlabel='Rating', ylabel='Content Rating'>
```



The above plot shows that Content for 'Everyone' has the lowest rating as it contain the highest number of outliers. Also Content for 'Mature 17+', 'Everyone 10+' and 'Teen' has medium ratings as there are some outliners between 2-3.5. The Content for 'Adults only 18+' has the highest ratings.

```
In [55]: #5) Boxplot for Ratings Vs. Category
sns.set(rc={'figure.figsize':(18,12)})
sns.boxplot(x= 'Rating', y = 'Category', data= df)

Out[55]: <Axes: xlabel='Rating', ylabel='Category'>
```



ART_AND_DESIGN
AUTO_AND_VEHICLES
BEAUTY
BOOKS_AND_REFERENCE
BUSINESS
COMICS
COMMUNICATION
DATING
EDUCATION
ENTERTAINMENT
EVENTS
FINANCE
FOOD_AND_DRINK
HEALTH_AND_FITNESS
HOUSE_AND_HOME
LIBRARIES_AND_DEMO
LIFESTYLE
GAME
FAMILY
MEDICAL
SOCIAL
SHOPPING
PHOTOGRAPHY
SPORTS
TRAVEL_AND_LOCAL
TOOLS
PERSONALIZATION
PRODUCTIVITY
PARENTING
WEATHER
VIDEO_PLAYERS
NEWS_AND_MAGAZINES
MAPS_AND_NAVIGATION
Category
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Rating

The 'Events' category has the highest Ratings out of all other app genres