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

```
In [2]:
df=pd.read_excel('playstore_file.xlsx')
```

```
In [3]:
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	19000.0	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	14000.0	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0	4.0.3 and up
2	U Launcher Lite â€“ FREE Live Cool Themes, Hid...	ART_AND_DESIGN	4.7	87510	8700.0	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	25000.0	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	2800.0	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.4 and up

```
In [4]:
df.isnull().sum()
```

```
Out[4]:
App                  1
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
```

```
In [5]:
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             10840 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
```

1. Data clean up – Missing value treatment

a. Drop records where rating is missing since rating is our target/study variable

```
In [6]:
df.dropna(subset=['Rating'],inplace=True)
```

b. Check the null values for the Android Ver column.

i. Are all 3 records having the same problem?

ii. Drop the 3rd record i.e. record for “Life Made WIFI …”

iii. Replace remaining missing values with the mode

```
In [7]:
df[df['Android Ver'].isna()]

#Are all 3 records having the same problem?
#The 3rd record is slightly different from the first two.
```

Out[7]:

	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	1.9	19.0	3.0M	21516.529524	Free	0	Everyone	NaN	February 11, 2018	1.0.19	4.0 and up	NaN

```
In [8]:
#Drop the 3rd record i.e. record for “Life Made WIFI …”
df.drop(10472,axis=0,inplace=True)
```

```
In [9]:
# Replace remaining missing values with the mode
df['Android Ver']=df['Android Ver'].fillna(df['Android Ver'].mode()[0])
```

c. Current ver – replace with most common value

```
In [10]:
df['Current Ver']=df['Current Ver'].fillna(df['Current Ver'].mode()[0])
```

2. Data clean up – correcting the data types

a. Which all variables need to be brought to numeric types?

Ans.- Reviews,Installs and Price need to be brought to numeric types.

b. Price variable – remove $ sign and convert to float

```
In [11]:
df['Price']=df['Price'].str.replace('$','')
df['Price']=np.where(df['Price'].isna(),0,df['Price'])
df['Price']=df['Price'].astype(float)
```

```
C:\Users\humza\AppData\Local\Temp/ipykernel_11760/1493807749.py:1: FutureWarning: The default value of regex will change from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings when regex=True.
  df['Price']=df['Price'].str.replace('$','')
```

c. Installs – remove ‘,’ and ‘+’ sign, convert to integer

```
In [12]:
df['Installs']=df['Installs'].str.replace(',','').str.replace('+','')
df['Installs']=df['Installs'].astype(int)
```

```
C:\Users\humza\AppData\Local\Temp/ipykernel_11760/2672956651.py:1: FutureWarning: The default value of regex will change from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings when regex=True.
  df['Installs']=df['Installs'].str.replace(',','').str.replace('+','')
```

d. Convert all other identified columns to numeric

```
In [13]:
df['Reviews']=df['Reviews'].astype(int)
```

3. Sanity checks – check for the following and handle accordingly

a. Avg. rating should be between 1 and 5, as only these values are allowed on the play store.

i. Are there any such records? Drop if so.

```
In [14]:
df['Rating'].unique()
```

```
Out[14]:
array([4.1, 3.9, 4.7, 4.5, 4.3, 4.4, 3.8, 4.2, 4.6, 3.2, 4. , 4.8, 4.9,
       3.6, 3.7, 3.3, 3.4, 3.5, 3.1, 5. , 2.6, 3. , 1.9, 2.5, 2.8, 2.7,
       1. , 2.9, 2.3, 2.2, 1.7, 2. , 1.8, 2.4, 1.6, 2.1, 1.4, 1.5, 1.2])
```

There is no such record having Rating>5 or <1.

b. Reviews should not be more than installs as only those who installed can review the app.

i. Are there any such records? Drop if so

```
In [15]:
x=list(df[df['Reviews']>df['Installs']].index)
df.drop(x,inplace=True)
```

4. Identify and handle outliers –

a. Price column

i. Make suitable plot to identify outliers in price

ii. Do you expect apps on the play store to cost $200? Check out these cases

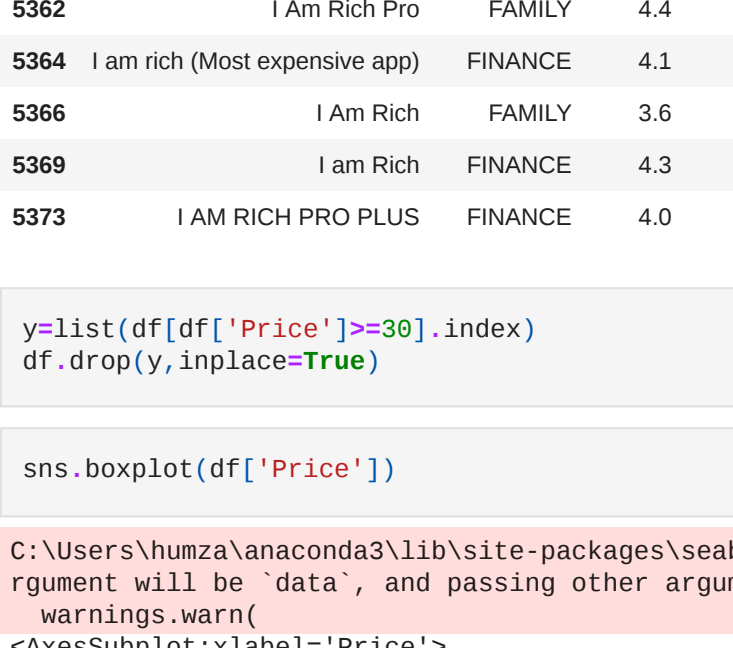
iii. After dropping the useless records, make the suitable plot again to identify outliers

iv. Limit data to records with price < $30

```
In [16]:
sns.boxplot(df['Price'])
```

```
C:\Users\humza\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[16]: <AxesSubplot:xlabel='Price'>




```
In [17]:
df[df['Price']>=200]
```

Out[17]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
4197	most expensive app (H)	FAMILY	4.3	6	1500.0	100	Paid	399.99	Everyone	Entertainment	July 16, 2018	1	7.0 and up
4362	ðŸ'Ž I'm rich	LIFESTYLE	3.8	718	26000.0	10000	Paid	399.99	Everyone	Lifestyle	March 11, 2018	1.0.0	4.4 and up
4367	I'm Rich - Trump Edition	LIFESTYLE	3.6	275	7300.0	10000	Paid	400.00	Everyone	Lifestyle	May 3, 2018	1.0.1	4.1 and up
5351	I am rich	LIFESTYLE	3.8	3547	1800.0	100000	Paid	399.99	Everyone	Lifestyle	January 12, 2018	2	4.0.3 and up
5354	I am Rich Plus	FAMILY	4.0	856	8700.0	10000	Paid	399.99	Everyone	Entertainment	May 19, 2018	3	4.4 and up
5355	I am rich VIP	LIFESTYLE	3.8	411	2600.0	10000	Paid	299.99	Everyone	Lifestyle	July 21, 2018	1.1.1	4.3 and up
5356	I Am Rich Premium	FINANCE	4.1	1867	4700.0	50000	Paid	399.99	Everyone	Finance	November 12, 2017	1.6	4.0 and up
5357	I am extremely Rich	LIFESTYLE	2.9	41	2900.0	1000	Paid	379.99	Everyone	Lifestyle	July 1, 2018	1	4.0 and up
5358	I am Rich!	FINANCE	3.8	93	22000.0	1000	Paid	399.99	Everyone	Finance	December 11, 2017	1	4.1 and up
5359	I am rich(premium)	FINANCE	3.5	472	965.0	5000	Paid	399.99	Everyone	Finance	May 1, 2017	3.4	4.4 and up
5362	I Am Rich Pro	FAMILY	4.4	201	2700.0	5000	Paid	399.99	Everyone	Entertainment	May 30, 2017	1.54	1.6 and up
5364	I am rich (Most expensive app)	FINANCE	4.1	129	2700.0	1000	Paid	399.99	Teen	Finance	December 6, 2017	2	4.0.3 and up
5366	I Am Rich	FAMILY	3.6	217	4900.0	10000	Paid	389.99	Everyone	Entertainment	June 22, 2018	1.5	4.2 and up
5369	I am Rich	FINANCE	4.3	180	3800.0	5000	Paid	399.99	Everyone	Finance	March 22, 2018	1	4.2 and up
5373	I AM RICH PRO PLUS	FINANCE	4.0	36	41000.0	1000	Paid	399.99	Everyone	Finance	June 25, 2018	1.0.2	4.1 and up

```
In [18]:
y=list(df[df['Price']>=30].index)
df.drop(y,inplace=True)
```

```
In [19]:
sns.boxplot(df['Price'])
```

```
C:\Users\humza\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[19]: <AxesSubplot:xlabel='Price'>

b. Reviews column

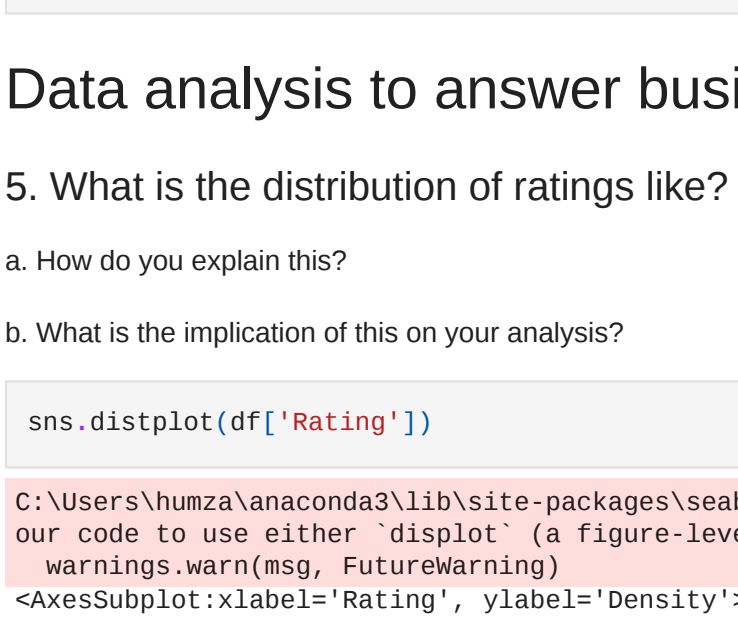
i. Make suitable plot

ii. Limit data to apps with < 1 Million reviews

```
In [20]:
sns.boxplot(df['Reviews'])
```

```
C:\Users\humza\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[20]: <AxesSubplot:xlabel='Reviews'>




```
In [21]:
z=list(df[df['Reviews']>=1000000].index)
df.drop(z,inplace=True)
```

c. Installs

i. What is the 95th percentile of the installs?

ii. Drop records having a value more than the 95th percentile

```
In [22]:
ninety_fifth=df['Installs'].quantile(0.95)
ninety_fifth
```

```
Out[22]:
10000000.0
```

10000000 is the 95th percentile of the installs.

```
In [23]:
df.drop(df[df['Installs']>ninety_fifth].index,inplace=True)
```

Data analysis to answer business questions

5. What is the distribution of ratings like? (use Seaborn) More skewed towards higher/lower values?

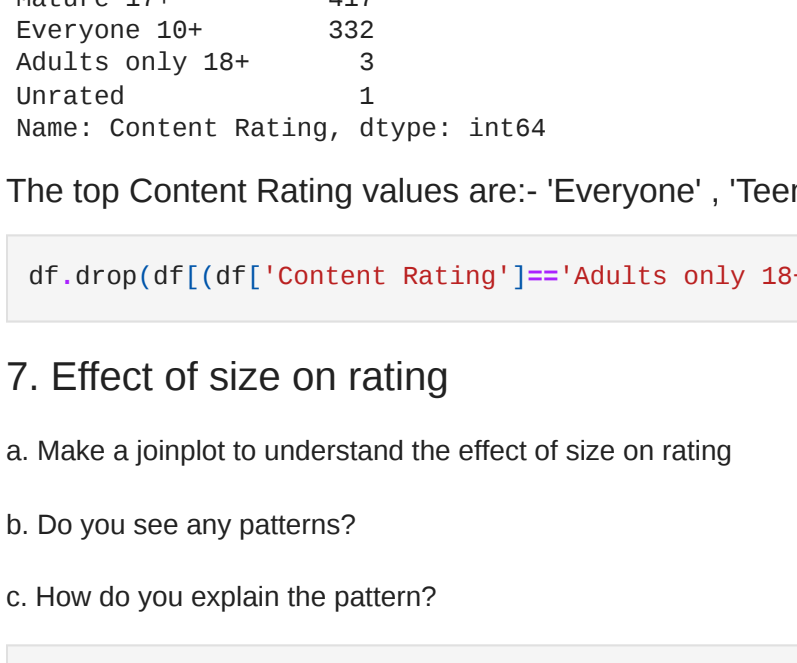
a. How do you explain this?

b. What is the implication of this on your analysis?

```
In [24]:
sns.distplot(df['Rating'])
```

```
C:\Users\humza\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[24]: <AxesSubplot:xlabel='Rating', ylabel='Density'>



The Ratings are more skewed towards higher values i.e. between 4 and 5.

a) This is because most of the apps are performing well and getting a good average rating.

b) Its implication on the analysis will be that all those apps lying in the higher rating zone will have a good user visibility while those lying in the lower rating bracket will eventually have a low user visibility.

6. What are the top Content Rating values?

a. Are there any values with very few records?

b. If yes, drop those as they won’t help in the analysis

```
In [25]:
df['Content Rating'].value_counts()
```

```
Out[25]:
Everyone           6782
Teen                900
Mature 17+          417
Everyone 10+        332
Adults only 18+       3
Unrated               1
Name: Content Rating, dtype: int64
```

The top Content Rating values are:- 'Everyone' , 'Teen' , 'Mature 17+' , 'Everyone 10+'.

```
In [26]:
df.drop(df[(df['Content Rating']=='Adults only 18+')|(df['Content Rating']=='Unrated')].index,inplace=True)
```

7. Effect of size on rating

a. Make a joinplot to understand the effect of size on rating

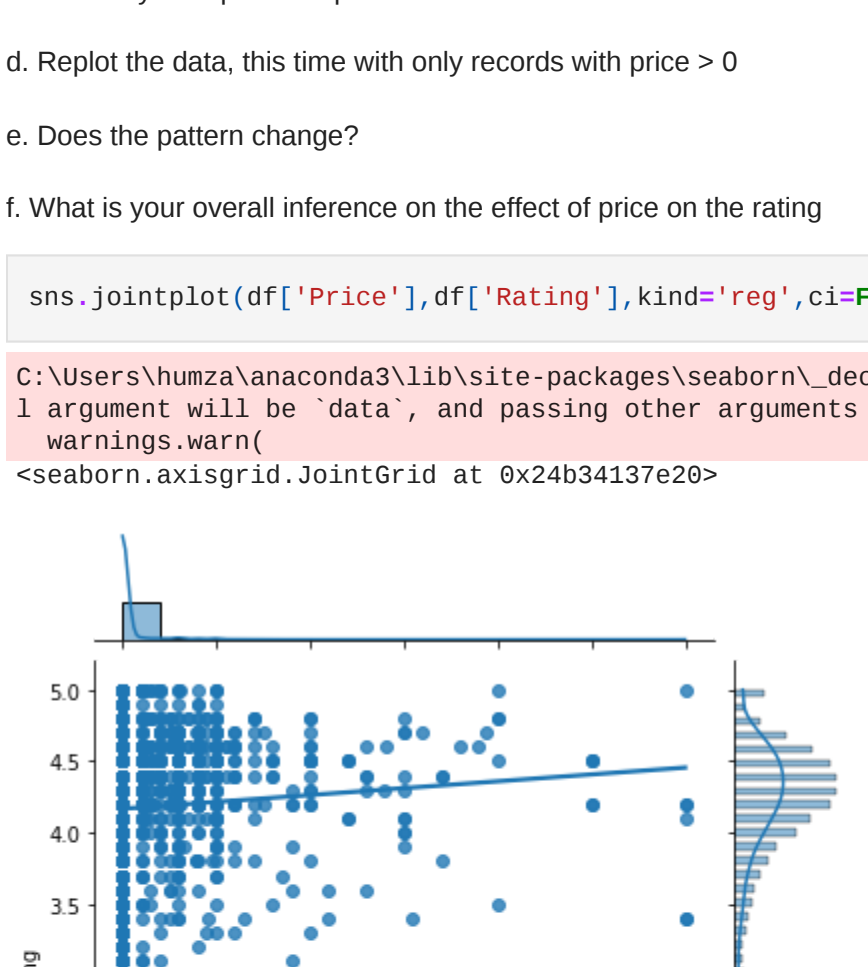
b. Do you see any patterns?

c. How do you explain the pattern?

```
In [27]:
sns.jointplot(df['Size'],df['Rating'])
```

```
C:\Users\humza\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[27]: <seaborn.axisgrid.JointGrid at 0x24b340d02b0>



The pattern obtained reveals that most of the apps having the size less than 40000 have got the highest ratings. Majority of the users use a low end device because of which they are not able to install very large apps , that is the major reason that most of the highly rated apps are small in size.

8. Effect of price on rating

a. Make a jointplot (with regression line)

b. What pattern do you see?

c. How do you explain the pattern?

d. Replot the data, this time with only records with price > 0

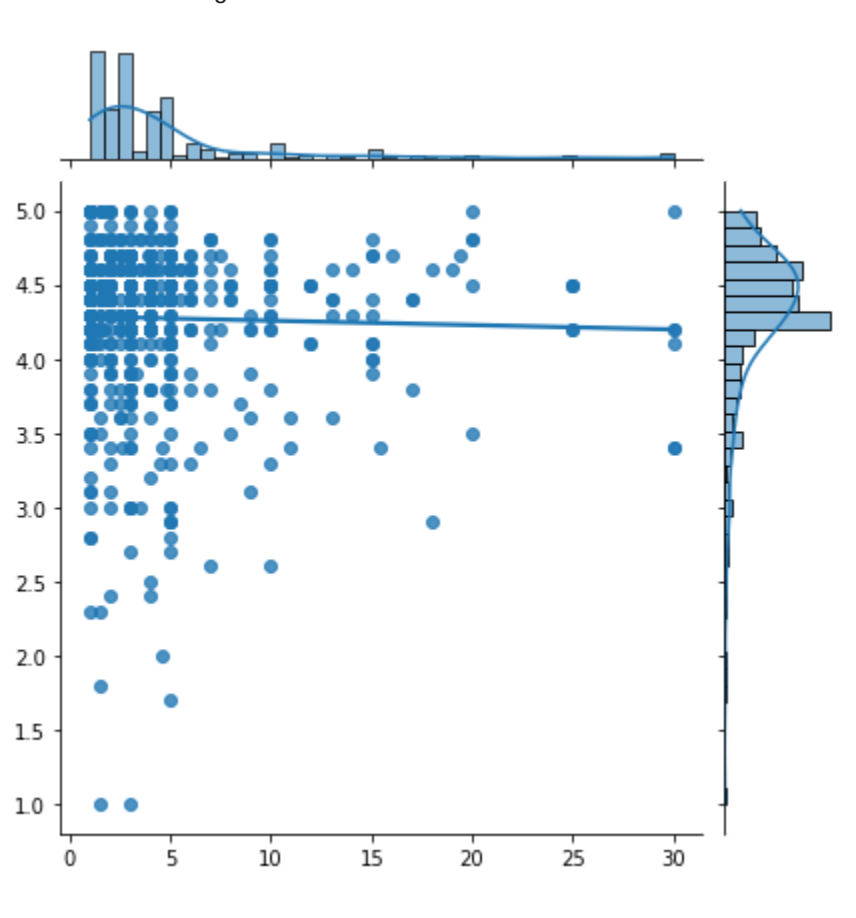
e. Does the pattern change?

f. What is your overall inference on the effect of price on the rating

```
In [28]:
sns.jointplot(df['Price'],df['Rating'],kind='reg',ci=False)
```

```
C:\Users\humza\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[28]: <seaborn.axisgrid.JointGrid at 0x24b34137e20>

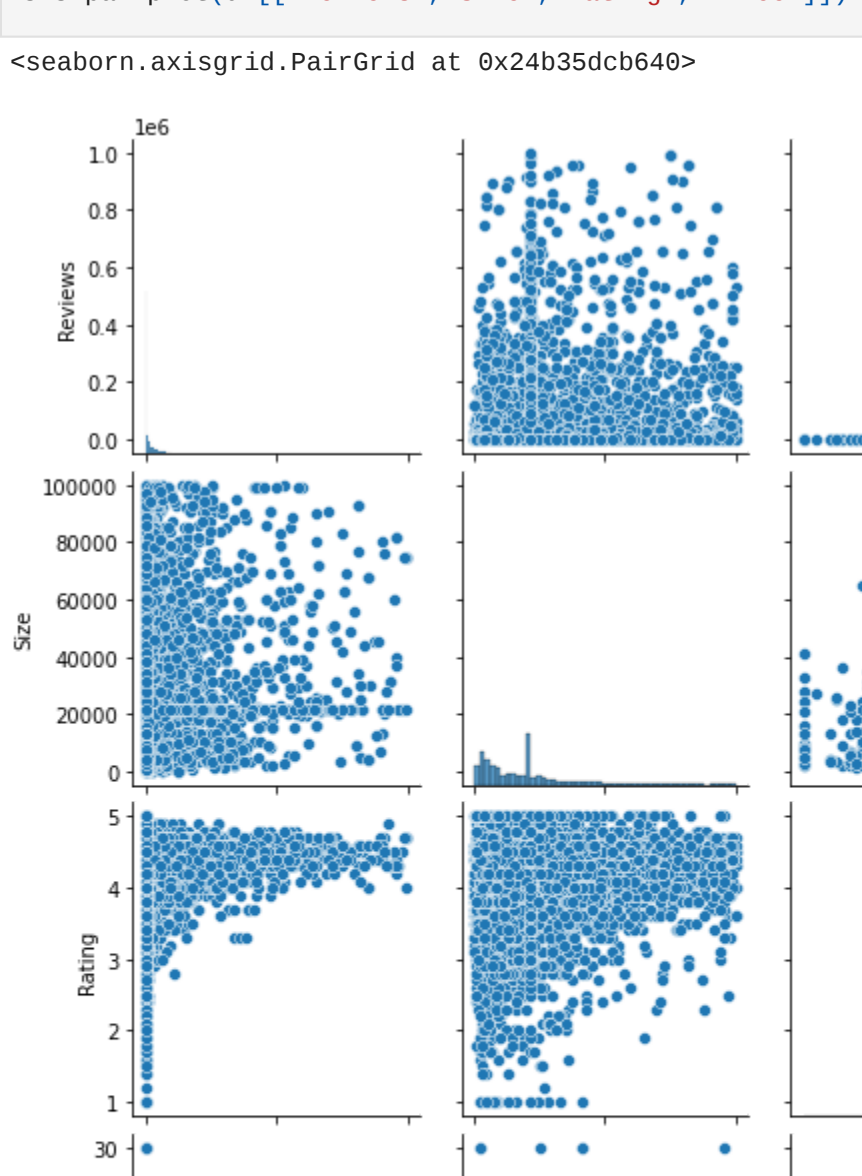


The Rating seems to be increasing gradually with the rise in price.

```
In [29]:
q=df.loc[df['Price']>0,'Price'].values
w=df.loc[df['Price']>0,'Rating'].values
sns.jointplot(q,w,kind='reg',ci=False)
```

```
C:\Users\humza\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[29]: <seaborn.axisgrid.JointGrid at 0x24b35dd75e0>



As soon as the apps with price 0 are excluded, the rating seems to be decreasing very gradually with the increase in price.

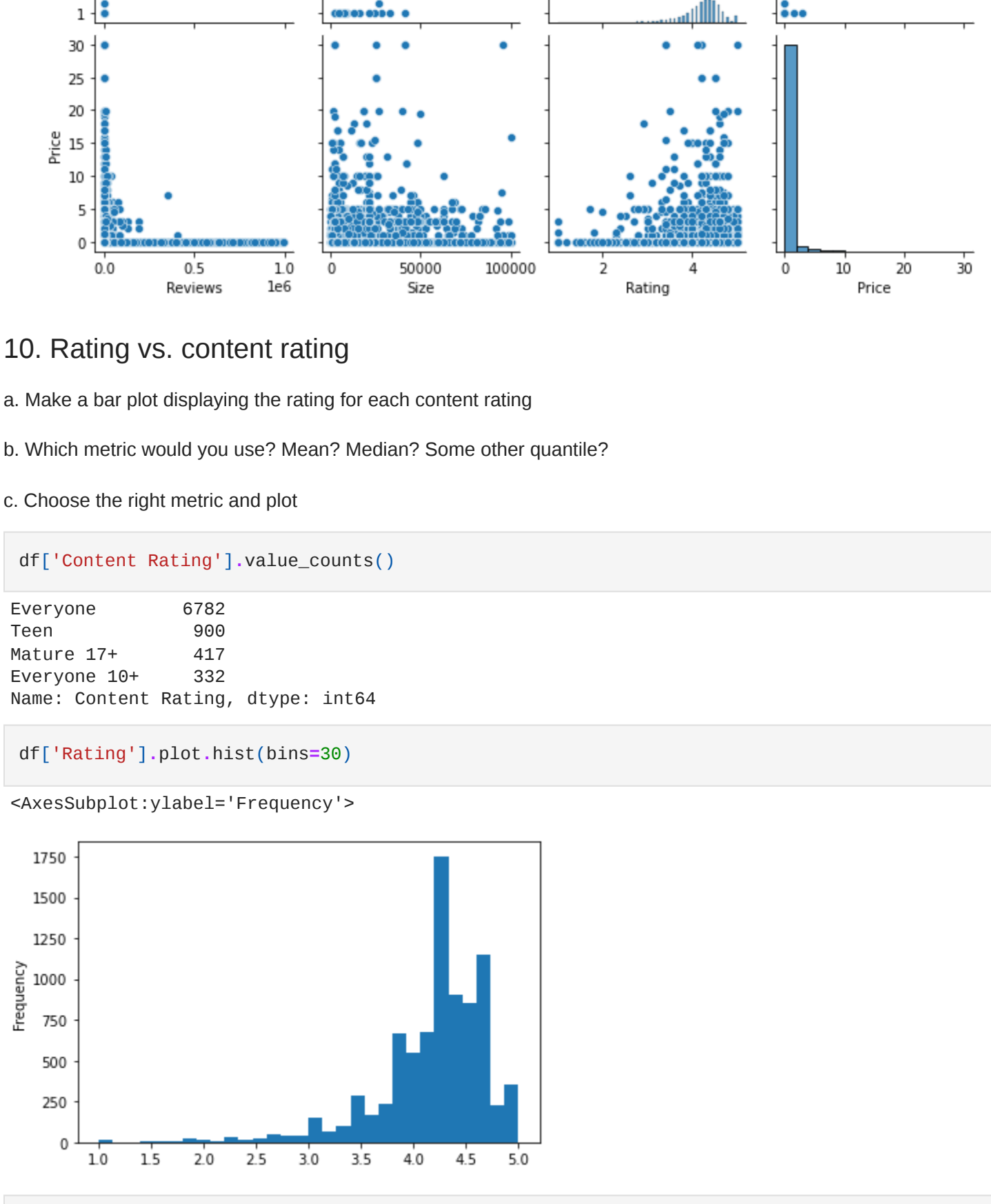
From the above observation we can finally infer that the apps having low price but not 0, are mostly liked and rated high by the users.

9. Look at all the numeric interactions together –

a. Make a pairplort with the colulmns - 'Reviews', 'Size', 'Rating', 'Price'

```
In [30]:
sns.pairplot(df[['Reviews','Size','Rating','Price']])
```

Out[30]: <seaborn.axisgrid.PairGrid at 0x24b35dcb640>



10. Rating vs. content rating

a. Make a bar plot displaying the rating for each content rating

b. Which metric would you use? Mean? Median? Some other quantile?

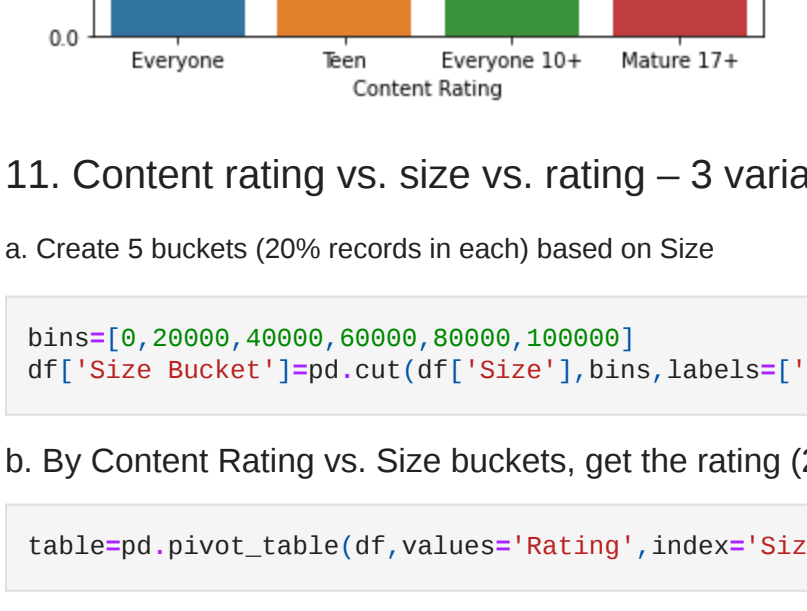
c. Choose the right metric and plot

```
In [31]:
df['Content Rating'].value_counts()
```

```
Out[31]:
Everyone        6782
Teen             900
Mature 17+       417
Everyone 10+     332
Name: Content Rating, dtype: int64
```

```
In [32]:
df['Rating'].plot.hist(bins=30)
```

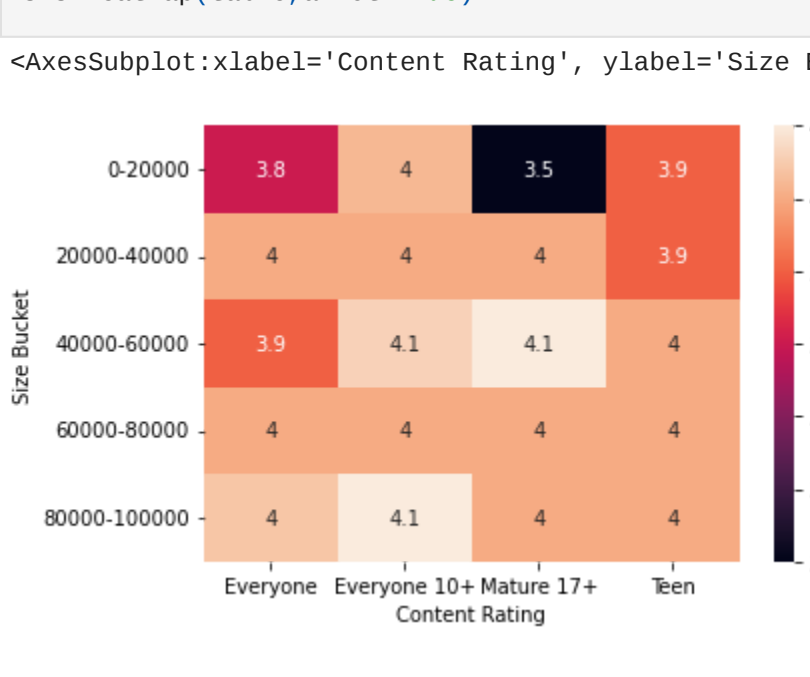
Out[32]: <AxesSubplot:ylabel='Frequency'>




```
In [33]:
from numpy import mean
sns.barplot('Content Rating','Rating',data=df,estimator=mean)
```

```
C:\Users\humza\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[33]: <AxesSubplot:xlabel='Content Rating', ylabel='Rating'>



11. Content rating vs. size vs. rating – 3 variables at a time

a. Create 5 buckets (20% records in each) based on Size

```
In [34]:
bins=[0,20000,40000,60000,80000,100000]
df['Size Bucket']=pd.cut(df['Size'],bins,labels=['0-20000','20000-40000','40000-60000','60000-80000','80000-100000'])
```

b. By Content Rating vs. Size buckets, get the rating (20th percentile) for each combination

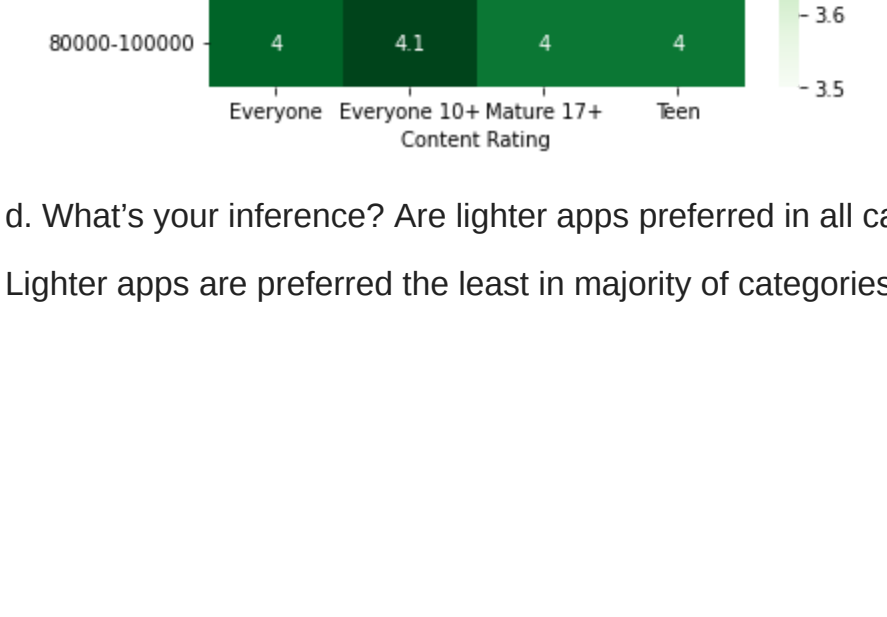
```
In [35]:
table=pd.pivot_table(df,values='Rating',index='Size Bucket',columns='Content Rating',aggfunc=lambda x:np.quantile(x,0.2))
```

c. Make a heatmap of this

i. Annotated

```
In [36]:
sns.heatmap(table,annot=True)
```

Out[36]: <AxesSubplot:xlabel='Content Rating', ylabel='Size Bucket'>



c. Make a heatmap of this

ii. Greens color map

```
In [37]:
sns.heatmap(table,annot=True,cmap='Greens')
```

Out[37]: <AxesSubplot:xlabel='Content Rating', ylabel='Size Bucket'>

d. What’s your inference? Are lighter apps preferred in all categories? Heavier? Some?

Lighter apps are preferred the least in majority of categories while heavier apps are preferred the most.