

Avinash A Godi

Python Assignment01

In [26]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
pl_ana=pd.read_csv("playstore-analysis (2) (1).csv")
pl_ana #reading dataframe as pl_ana
```

Out[2]:

	App	Category	Rating	Reviews	Size	Installs	Type
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19000.000000	10,000+	Free
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14000.000000	500,000+	Free
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8700.000000	5,000,000+	Free
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25000.000000	50,000,000+	Free
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2800.000000	100,000+	Free
...
10836	Sya9a Maroc - FR	FAMILY	4.5	38	53000.000000	5,000+	Free
10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3600.000000	100+	Free
10838	Parkinson Exercices FR	MEDICAL	NaN	3	9500.000000	1,000+	Free
10839	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5	114	21516.529524	1,000+	Free
10840	iHoroscope - 2018 Daily Horoscope & Astrology	LIFESTYLE	4.5	398307	19000.000000	10,000,000+	Free

10841 rows × 13 columns

In [3]:

```
p1_ana.head()
```

Out[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

Tasks

1. Data clean up – Missing value treatment

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

In [4]:

```
p1_ana.dropna(how='any', subset=['Rating'], axis=0, inplace = True)
```

In [5]:

```
p1_ana.Rating.isnull().sum()
```

Out[5]:

0

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

i. Are all 3 records having the same problem?

In [6]:

```
pl_ana.loc[pl_ana['Android Ver'].isnull()]
```

Out[6]:

	App	Category	Rating	Reviews	Size	Installs	Type	Pric
4453	[substratum] Vacuum: P	PERSONALIZATION	4.4	230	11000.000000	1,000+	Paid	\$1.4
4490	Pi Dark [substratum]	PERSONALIZATION	4.5	189	2100.000000	10,000+	Free	
10472	Life Made Wi-Fi Touchscreen Photo Frame	1.9	19.0	3.0M	21516.529524	Free	0	Everyon

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

In [7]:

```
pl_ana.drop([10472], inplace = True)
```

In [8]:

```
pl_ana.loc[pl_ana['Android Ver'].isnull()]
```

Out[8]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Conter Ratin
4453	[substratum] Vacuum: P	PERSONALIZATION	4.4	230	11000.0	1,000+	Paid	\$1.49	Everyon
4490	Pi Dark [substratum]	PERSONALIZATION	4.5	189	2100.0	10,000+	Free	0	Everyon

iii. Replace remaining missing values with the mode

In [9]:

```
pl_ana['Android Ver'].fillna(pl_ana['Android Ver'].mode()[0], inplace=True)
```

c. Current ver – replace with most common value

In [10]:

```
pl_ana['Current Ver'].fillna(pl_ana['Current Ver'].mode()[0], inplace=True)
```

2. Data clean up – correcting the data types

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

Reviews and installs need to be brought to numeric types.

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

In [11]:

```
price = []
for i in pl_ana['Price']:
    if i[0]=='$':
        price.append(i[1:])
    else:
        price.append(i)
```

In [12]:

```
pl_ana.drop(labels=pl_ana[pl_ana['Price']=='Everyone'].index, inplace = True)
pl_ana['Price'] = price
pl_ana['Price'] = pl_ana['Price'].astype('float')
```

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

In [13]:

```
install = []
for j in pl_ana['Installs']:
    install.append(j.replace(',','').replace('+','').strip())

pl_ana['Installs'] = install
pl_ana['Installs'] = pl_ana['Installs'].astype('int')
```

d. Convert all other identified columns to numeric

In [14]:

```
pl_ana['Reviews'] = pl_ana['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 [15]:

```
pl_ana.loc[pl_ana.Rating < 1] & pl_ana.loc[pl_ana.Rating > 5]
```

Out[15]:

App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Cur
<div> <div></div> <div></div> </div>											

There are no such records with rating less than 1 or greater than 5.

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.

Yes, there are 7 records where Review is greater than Installs.

In [17]:

```
pl_ana.loc[pl_ana['Reviews'] > pl_ana['Installs']]
```

Out[17]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Ge
2454	KBA-EZ Health Guide	MEDICAL	5.0	4	25000.000000	1	Free	0.00	Everyone	Me
4663	Alarmy (Sleep If U Can) - Pro	LIFESTYLE	4.8	10249	21516.529524	10000	Paid	2.49	Everyone	Lif
5917	Ra Ga Ba	GAME	5.0	2	20000.000000	1	Paid	1.49	Everyone	A
6700	Brick Breaker BR	GAME	5.0	7	19000.000000	5	Free	0.00	Everyone	A
7402	Trovami se ci riesci	GAME	5.0	11	6100.000000	10	Free	0.00	Everyone	A
8591	DN Blog	SOCIAL	5.0	20	4200.000000	10	Free	0.00	Teen	5
10697	Mu.F.O.	GAME	5.0	2	16000.000000	1	Paid	0.99	Everyone	A

In [18]:

```
temp = pl_ana[pl_ana['Reviews'] > pl_ana['Installs']].index
pl_ana.drop(labels=temp, inplace=True)
```

In [19]:

```
pl_ana.loc[pl_ana['Reviews'] > pl_ana['Installs']]
```

Out[19]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Cur
--	-----	----------	--------	---------	------	----------	------	-------	----------------	--------	--------------	-----

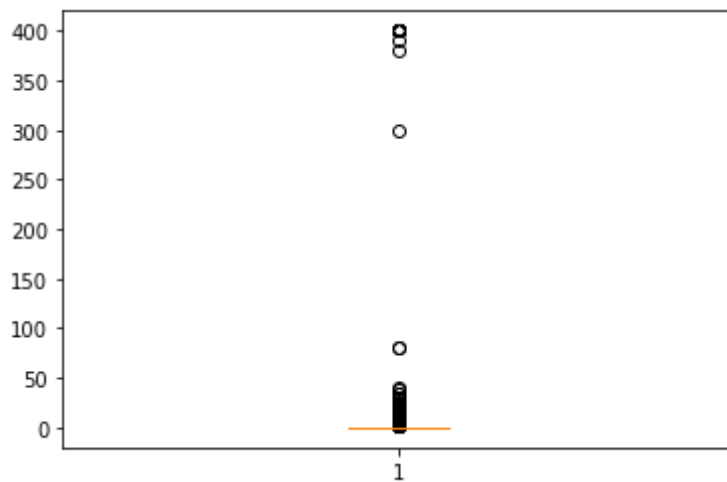
4. Identify and handle outliers –

a. Price column

i. Make suitable plot to identify outliers in price

In [20]:

```
plt.boxplot(pl_ana['Price'])  
plt.show()
```




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

In [21]:

```
print('Yes we can expect apps on the play store to cost $200')
pl_ana.loc[pl_ana['Price'] > 200]
```

Yes we can expect apps on the play store to cost \$200

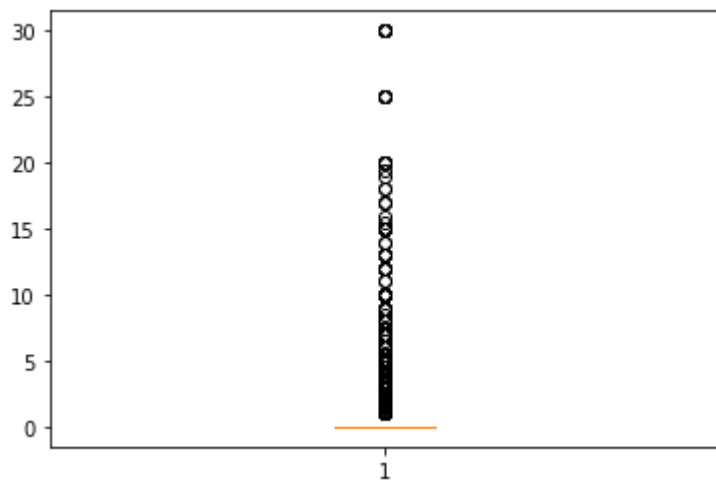
Out[21]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	
4197	most expensive app (H)	FAMILY	4.3	6	1500.0	100	Paid	399.99	Everyone	Ent
4362	 I'm rich	LIFESTYLE	3.8	718	26000.0	10000	Paid	399.99	Everyone	
4367	I'm Rich - Trump Edition	LIFESTYLE	3.6	275	7300.0	10000	Paid	400.00	Everyone	
5351	I am rich	LIFESTYLE	3.8	3547	1800.0	100000	Paid	399.99	Everyone	
5354	I am Rich Plus	FAMILY	4.0	856	8700.0	10000	Paid	399.99	Everyone	Ent
5355	I am rich VIP	LIFESTYLE	3.8	411	2600.0	10000	Paid	299.99	Everyone	
5356	I Am Rich Premium	FINANCE	4.1	1867	4700.0	50000	Paid	399.99	Everyone	
5357	I am extremely Rich	LIFESTYLE	2.9	41	2900.0	1000	Paid	379.99	Everyone	
5358	I am Rich!	FINANCE	3.8	93	22000.0	1000	Paid	399.99	Everyone	
5359	I am rich(premium)	FINANCE	3.5	472	965.0	5000	Paid	399.99	Everyone	
5362	I Am Rich Pro	FAMILY	4.4	201	2700.0	5000	Paid	399.99	Everyone	Ent
5364	I am rich (Most expensive app)	FINANCE	4.1	129	2700.0	1000	Paid	399.99	Teen	
5366	I Am Rich	FAMILY	3.6	217	4900.0	10000	Paid	389.99	Everyone	Ent
5369	I am Rich	FINANCE	4.3	180	3800.0	5000	Paid	399.99	Everyone	
5373	I AM RICH PRO PLUS	FINANCE	4.0	36	41000.0	1000	Paid	399.99	Everyone	

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

In [24]:

```
plt.boxplot(pl_ana['Price'])  
plt.show()
```



iv. Limit data to records with price < \$30

In [22]:

```
gt_30 = pl_ana[pl_ana['Price'] > 30].index  
pl_ana.drop(labels=gt_30, inplace=True)
```

In [23]:

```
count = pl_ana.loc[pl_ana['Price'] > 30].index  
count.value_counts().sum()
```

Out[23]:

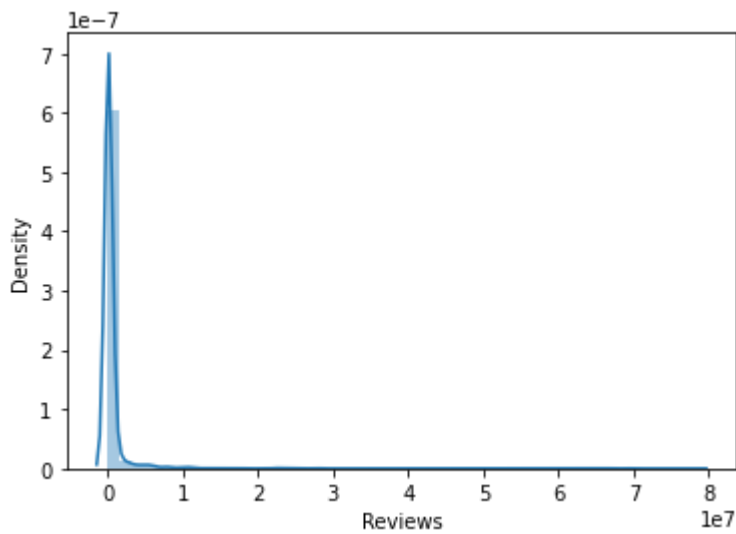
0

b. Reviews column

i. Make suitable plot

In [27]:

```
sns.distplot(pl_ana['Reviews'])  
plt.show()
```



ii. Limit data to apps with < 1 Million reviews

In [28]:

```
gt_1m = pl_ana[pl_ana['Reviews'] > 1000000 ].index  
pl_ana.drop(labels = gt_1m, inplace=True)  
print(gt_1m.value_counts().sum(), 'cols dropped')
```

704 cols dropped

c. Installs

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

In [29]:

```
percentile = pl_ana.Installs.quantile(0.95)  
print(percentile, "is 95th percentile of Installs")
```

10000000.0 is 95th percentile of Installs

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

In [30]:

```
for i in range(0,101,1):  
    print(' the {} percentile of installs is {}'.format(i,np.percentile(pl_ana['Installs']
```

```
the 0 percentile of installs is 5.0  
the 1 percentile of installs is 50.0  
the 2 percentile of installs is 100.0  
the 3 percentile of installs is 100.0  
the 4 percentile of installs is 100.0  
the 5 percentile of installs is 100.0  
the 6 percentile of installs is 500.0  
the 7 percentile of installs is 500.0  
the 8 percentile of installs is 1000.0  
the 9 percentile of installs is 1000.0  
the 10 percentile of installs is 1000.0  
the 11 percentile of installs is 1000.0  
the 12 percentile of installs is 1000.0  
the 13 percentile of installs is 1000.0  
the 14 percentile of installs is 1000.0  
the 15 percentile of installs is 1000.0  
the 16 percentile of installs is 5000.0  
the 17 percentile of installs is 5000.0  
the 18 percentile of installs is 5000.0  
the 19 percentile of installs is 5000.0  
the 20 percentile of installs is 5000.0  
the 21 percentile of installs is 10000.0  
the 22 percentile of installs is 10000.0  
the 23 percentile of installs is 10000.0  
the 24 percentile of installs is 10000.0  
the 25 percentile of installs is 10000.0  
the 26 percentile of installs is 10000.0  
the 27 percentile of installs is 10000.0  
the 28 percentile of installs is 10000.0  
the 29 percentile of installs is 10000.0  
the 30 percentile of installs is 10000.0  
the 31 percentile of installs is 10000.0  
the 32 percentile of installs is 10000.0  
the 33 percentile of installs is 50000.0  
the 34 percentile of installs is 50000.0  
the 35 percentile of installs is 50000.0  
the 36 percentile of installs is 50000.0  
the 37 percentile of installs is 50000.0  
the 38 percentile of installs is 100000.0  
the 39 percentile of installs is 100000.0  
the 40 percentile of installs is 100000.0  
the 41 percentile of installs is 100000.0  
the 42 percentile of installs is 100000.0  
the 43 percentile of installs is 100000.0  
the 44 percentile of installs is 100000.0  
the 45 percentile of installs is 100000.0  
the 46 percentile of installs is 100000.0  
the 47 percentile of installs is 100000.0  
the 48 percentile of installs is 100000.0  
the 49 percentile of installs is 100000.0  
the 50 percentile of installs is 100000.0  
the 51 percentile of installs is 500000.0  
the 52 percentile of installs is 500000.0  
the 53 percentile of installs is 500000.0  
the 54 percentile of installs is 500000.0  
the 55 percentile of installs is 500000.0
```

```
the 56 percentile of installs is 500000.0
the 57 percentile of installs is 500000.0
the 58 percentile of installs is 1000000.0
the 59 percentile of installs is 1000000.0
the 60 percentile of installs is 1000000.0
the 61 percentile of installs is 1000000.0
the 62 percentile of installs is 1000000.0
the 63 percentile of installs is 1000000.0
the 64 percentile of installs is 1000000.0
the 65 percentile of installs is 1000000.0
the 66 percentile of installs is 1000000.0
the 67 percentile of installs is 1000000.0
the 68 percentile of installs is 1000000.0
the 69 percentile of installs is 1000000.0
the 70 percentile of installs is 1000000.0
the 71 percentile of installs is 1000000.0
the 72 percentile of installs is 1000000.0
the 73 percentile of installs is 1000000.0
the 74 percentile of installs is 1000000.0
the 75 percentile of installs is 1000000.0
the 76 percentile of installs is 5000000.0
the 77 percentile of installs is 5000000.0
the 78 percentile of installs is 5000000.0
the 79 percentile of installs is 5000000.0
the 80 percentile of installs is 5000000.0
the 81 percentile of installs is 5000000.0
the 82 percentile of installs is 5000000.0
the 83 percentile of installs is 5000000.0
the 84 percentile of installs is 5000000.0
the 85 percentile of installs is 10000000.0
the 86 percentile of installs is 10000000.0
the 87 percentile of installs is 10000000.0
the 88 percentile of installs is 10000000.0
the 89 percentile of installs is 10000000.0
the 90 percentile of installs is 10000000.0
the 91 percentile of installs is 10000000.0
the 92 percentile of installs is 10000000.0
the 93 percentile of installs is 10000000.0
the 94 percentile of installs is 10000000.0
the 95 percentile of installs is 10000000.0
the 96 percentile of installs is 10000000.0
the 97 percentile of installs is 10000000.0
the 98 percentile of installs is 50000000.0
the 99 percentile of installs is 50000000.0
the 100 percentile of installs is 1000000000.0
```

In [31]:

```
temp1 = pl_ana[pl_ana["Installs"] > percentile].index
pl_ana.drop(labels = temp1, inplace = True)
print(temp1.value_counts().sum())
```

199

Data analysis to answer business questions

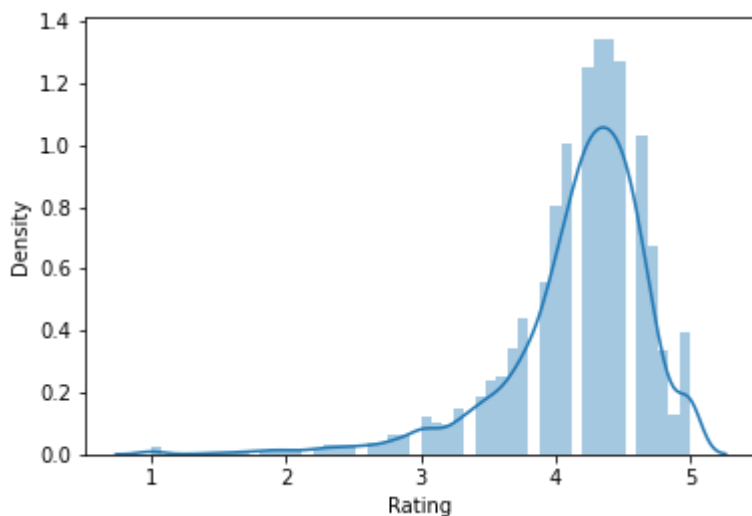
5. What is the distribution of ratings like? (use Seaborn) More skewed towards higher/lower

values?

a. How do you explain this?

In [32]:

```
plot(pl_ana['Rating'])
# The skewness of this distribution is',pl_ana['Rating'].skew())
# The Median of this distribution {} is greater than mean {} of this distribution'.format(pl_ana
```



The skewness of this distribution is -1.7434270330647985

The Median of this distribution 4.3 is greater than mean 4.170800237107298 of this distribution

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

In [33]:

```
pl_ana['Rating'].mode()
```

Out[33]:

```
0    4.3
dtype: float64
```

6. What are the top Content Rating values?

a. Are there any values with very few records?

In [34]:

```
pl_ana['Content Rating'].value_counts()
```

Out[34]:

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

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

In [35]:

```
cr = []
for k in pl_ana['Content Rating']:
    cr.append(k.replace('Adults only 18+', 'NaN').replace('Unrated', 'NaN'))

pl_ana['Content Rating']=cr
```

In [36]:

```
temp2 = pl_ana[pl_ana["Content Rating"] == 'NaN'].index
pl_ana.drop(labels=temp2, inplace=True)
print('dropped cols',temp2)
```

```
dropped cols Int64Index([298, 3043, 6424, 8266], dtype='int64')
```

In [37]:

```
pl_ana['Content Rating'].value_counts()
```

Out[37]:

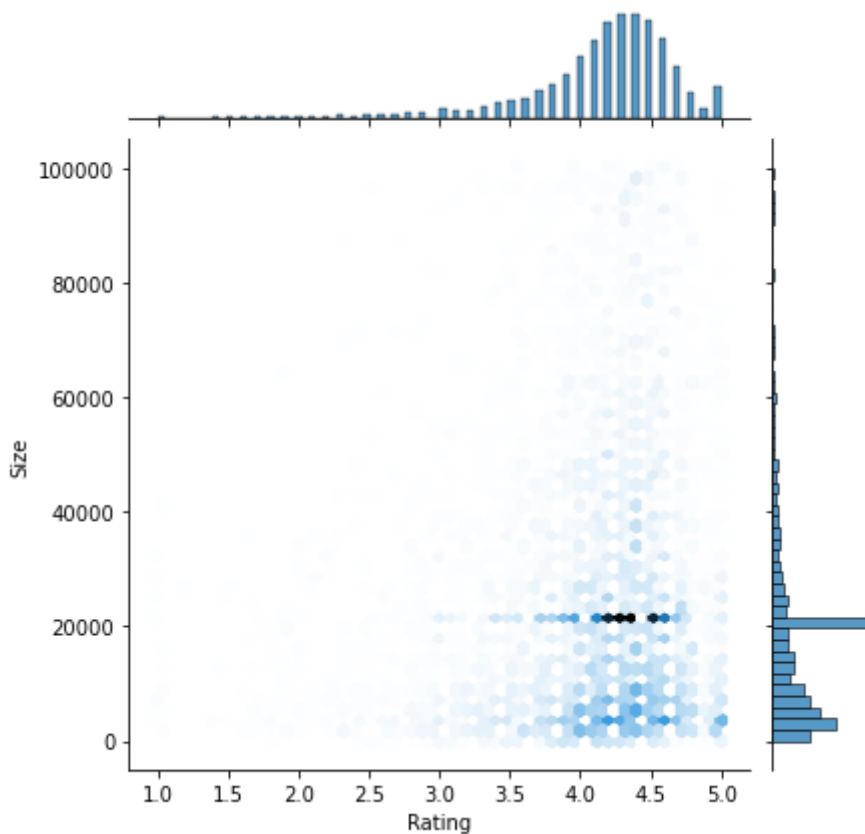
```
Everyone          6782
Teen              900
Mature 17+        417
Everyone 10+      332
Name: Content Rating, dtype: int64
```

7. Effect of size on rating

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

In [38]:

```
sns.jointplot(y = 'Size', x = 'Rating', data = pl_ana, kind = 'hex')  
plt.show()
```



b. Do you see any patterns?

Yes, patterns can be observed between Size and Rating ie. there is correlation between Size and Rating.

c. How do you explain the pattern?

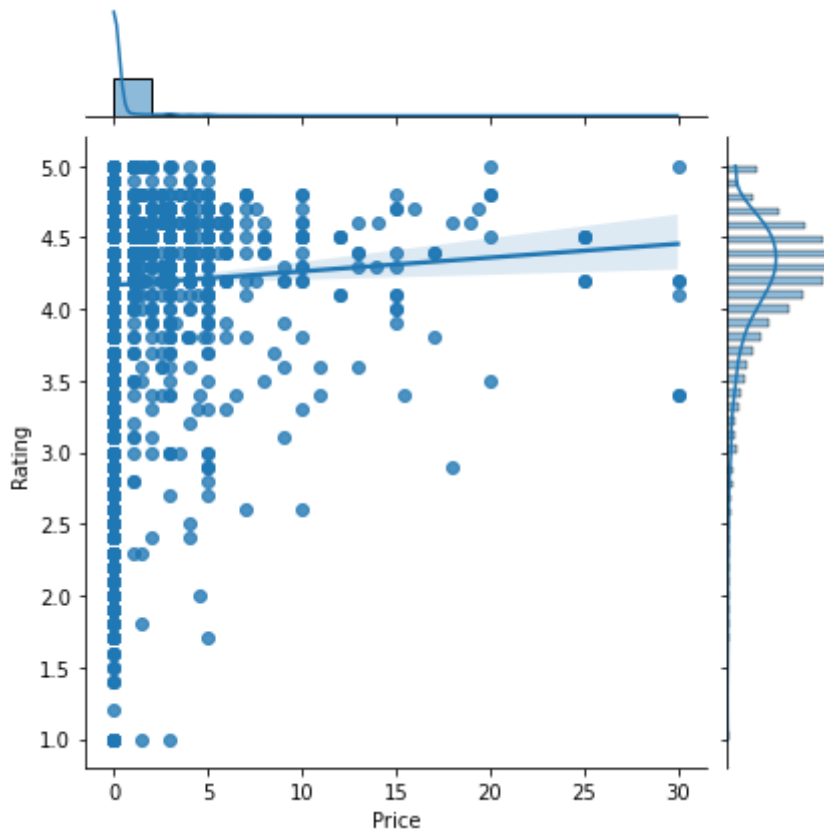
Generally on increasing Rating, Size of App also increases. But this is not always true i.e. for higher Rating, there is constant Size. Thus we can conclude that there is positive correlation between Size and Rating.

8. Effect of price on rating

a. Make a jointplot (with regression line)

In [40]:

```
sns.jointplot(x='Price', y='Rating', data=pl_ana, kind='reg')
plt.show()
```



b. What pattern do you see?

Generally on increasing the Price, Rating remains almost constant greater than 4.

c. How do you explain the pattern?

Since on increasing the Price, Rating remains almost constant greater than 4. Thus it can be concluded that there is very weak Positive correlation between Rating and Price.

In [41]:

```
pl_ana.corr()
```

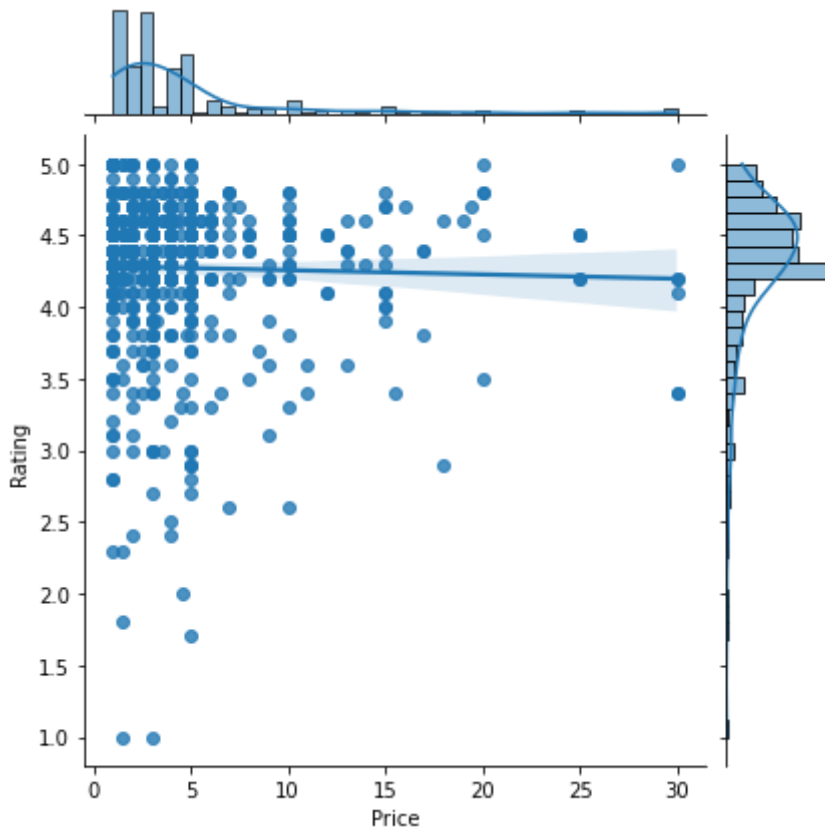
Out[41]:

	Rating	Reviews	Size	Installs	Price
Rating	1.000000	0.158547	0.058076	0.118414	0.031479
Reviews	0.158547	1.000000	0.204667	0.736038	-0.073446
Size	0.058076	0.204667	1.000000	0.190741	-0.001054
Installs	0.118414	0.736038	0.190741	1.000000	-0.110507
Price	0.031479	-0.073446	-0.001054	-0.110507	1.000000

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

In [42]:

```
ps1=pl_ana.loc[pl_ana.Price>0]
sns.jointplot(x='Price', y='Rating', data=ps1, kind='reg')
plt.show()
```



e. Does the pattern change?

Yes, On limiting the record with Price > 0, the overall pattern changed a slight ie, their is very weakly Negative Correlation between Price and Rating.

In [43]:

```
ps1.corr()
```

Out[43]:

	Rating	Reviews	Size	Installs	Price
Rating	1.000000	0.095986	0.117943	0.063960	-0.025975
Reviews	0.095986	1.000000	0.163959	0.787628	-0.049764
Size	0.117943	0.163959	1.000000	0.119255	0.024912
Installs	0.063960	0.787628	0.119255	1.000000	-0.057710
Price	-0.025975	-0.049764	0.024912	-0.057710	1.000000

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

Generally increasing the Prices, doesn't have signifcant effect on Higher Rating. For Higher Price, Rating is

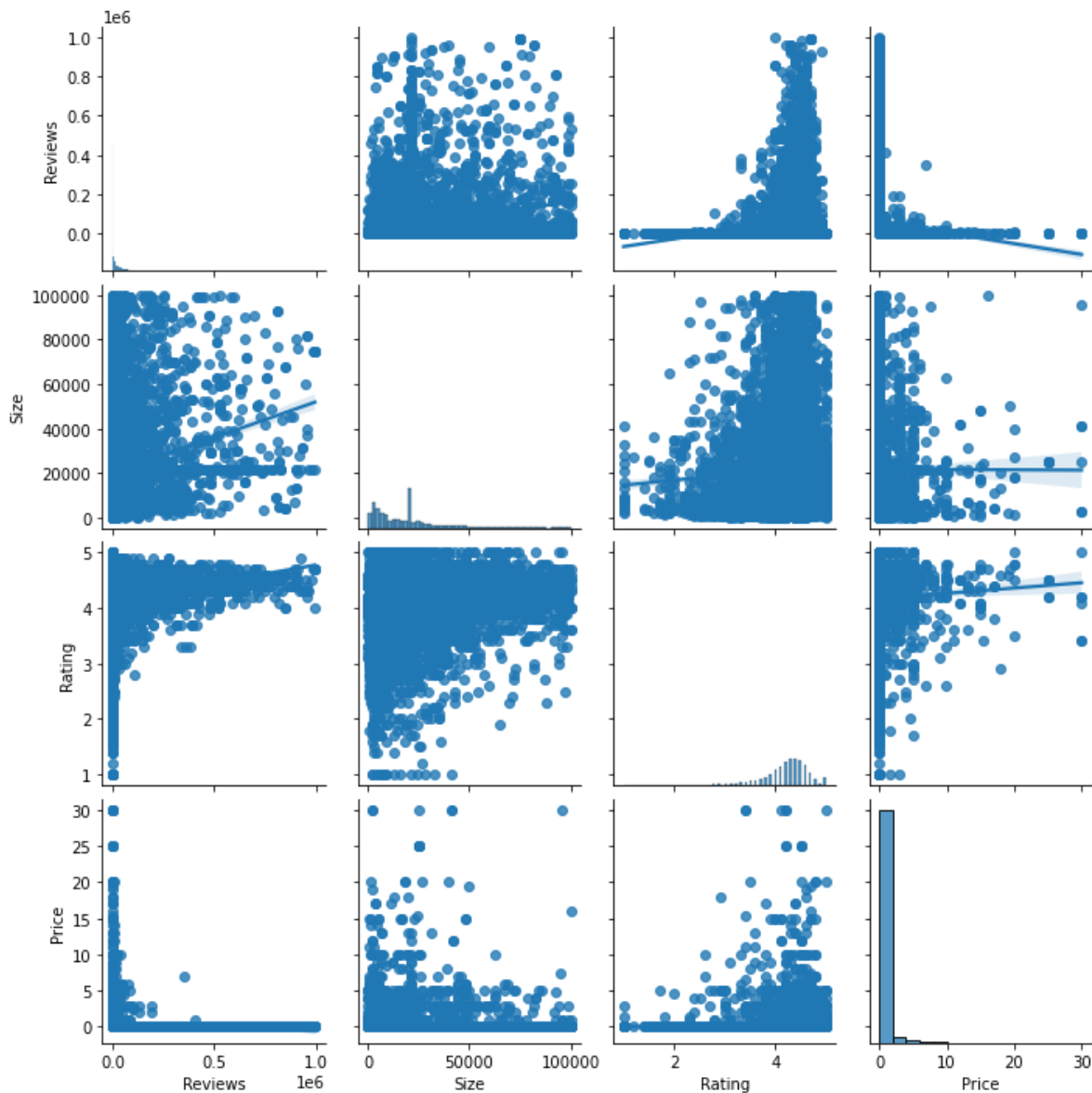
High and almost constant ie greater than 4

9. Look at all the numeric interactions together –

a. Make a pairplot with the columns - 'Reviews', 'Size', 'Rating', 'Price'

In [44]:

```
sns.pairplot(pl_ana, vars=['Reviews', 'Size', 'Rating', 'Price'], kind='reg')  
plt.show()
```

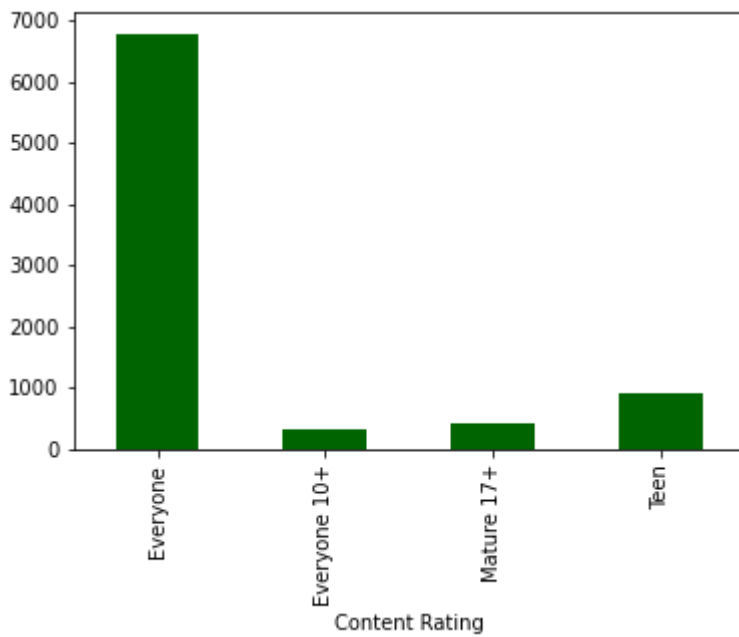


10. Rating vs. content rating

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

In [45]:

```
pl_ana.groupby(['Content Rating'])['Rating'].count().plot.bar(color="darkgreen")  
plt.show()
```

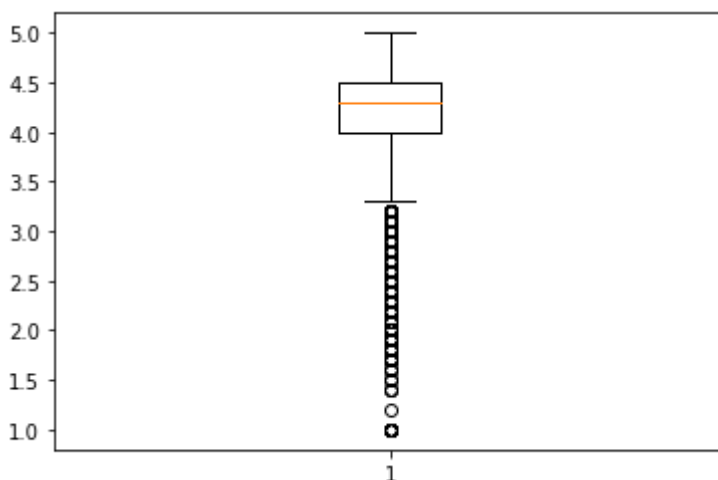


b. Which metric would you use? Mean? Median? Some other quantile?

We must use Median in this case as we are having Outliers in Rating. Because in case of Outliers , median is the best measure of central tendency.

In [46]:

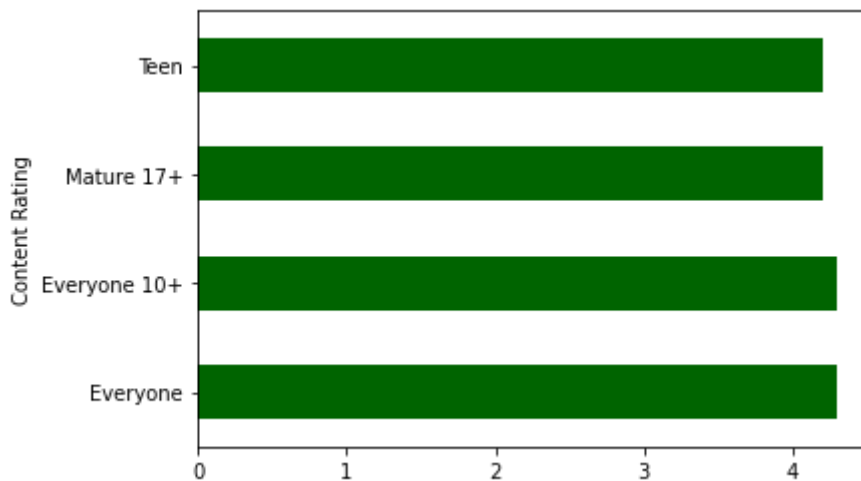
```
plt.boxplot(pl_ana['Rating'])  
plt.show()
```



c. Choose the right metric and plot

In [47]:

```
pl_ana.groupby(['Content Rating'])['Rating'].median().plot.barh(color="darkgreen")
plt.show()
```



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

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

In [49]:

```
bins=[0, 20000, 40000, 60000, 80000, 100000]
pl_ana['Bucket Size'] = pd.cut(pl_ana['Size'], bins, labels=['0-20k', '20k-40k', '40k-60k', '60k-80k', '80k-100k'])
pd.pivot_table(pl_ana, values='Rating', index='Bucket Size', columns='Content Rating')
```

Out[49]:

Content Rating	Everyone	Everyone 10+	Mature 17+	Teen
Bucket Size				
0-20k	4.145730	4.247561	4.010582	4.182240
20k-40k	4.200195	4.169811	4.156291	4.170432
40k-60k	4.167083	4.263636	4.190476	4.237383
60k-80k	4.245408	4.280769	4.200000	4.274194
80k-100k	4.260127	4.304762	4.252632	4.270313

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

In [50]:

```
temp3=pd.pivot_table(pl_ana, values='Rating', index='Bucket Size', columns='Content Rating'  
temp3
```

Out[50]:

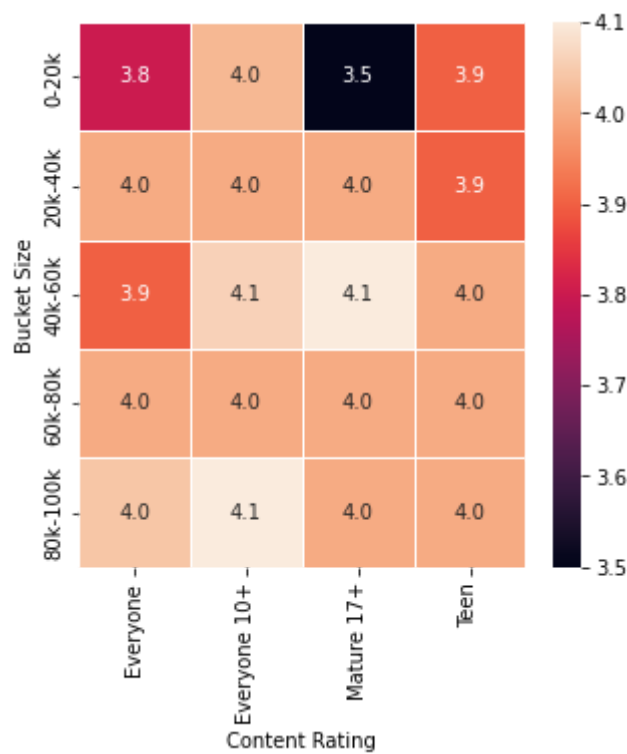
Content Rating	Everyone	Everyone 10+	Mature 17+	Teen
Bucket Size				
0-20k	3.80	4.02	3.5	3.9
20k-40k	4.00	4.00	4.0	3.9
40k-60k	3.90	4.06	4.1	4.0
60k-80k	4.00	4.00	4.0	4.0
80k-100k	4.04	4.10	4.0	4.0

c. Make a heatmap of this

i. Annotated

In [51]:

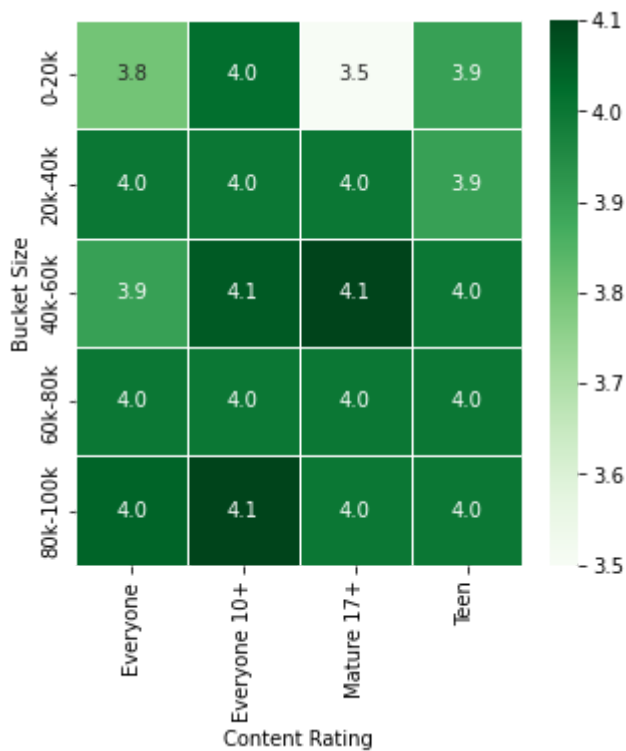
```
f,ax = plt.subplots(figsize=(5, 5))  
sns.heatmap(temp3, annot=True, linewidths=.5, fmt='.1f',ax=ax)  
plt.show()
```



ii. Greens color map

In [53]:

```
f,ax = plt.subplots(figsize=(5, 5))
sns.heatmap(temp3, annot=True, linewidths=.5, cmap='Greens',fmt='.1f',ax=ax)
plt.show()
```



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

Based on analysis, its not true that lighter apps are preferred in all categories. Because apps with size 40k-60k and 80k-100k have got the highest rating in all cateegories. So, in general we can conclude that heavier apps are preferred in all categories.

Thank You