

```
# Project: App rating Prediction

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# 1.Load the data file using Pandas
df=pd.read_csv('/content/Google store App rating_Python.csv')
df.shape

(10841, 13)
```

```
df.head(5)
```



	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	07-Jan-18	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	15-Jan-18	2.0.0	4.0.3 and up
	U Launcher Lite AS												

```
df.info()
```

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

```
<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.isnull().sum()
```

```
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
```

```
## 3. Drop records with nulls in any of the columns.
df=df.dropna()
```

```
#new shape after the drop
df.shape

(9360, 13)
```

```
df.isnull().sum()

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
```

```
df = df[df.Size != 'Varies with device']
```

df

	App	Category	Rating	Reviews	Size	Installs	Type
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free
2	U Launcher Lite â€” FREE Live Cool Themes, Hid...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free
4	Pixel Draw - Number Art	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free

```
# Size column has sizes in Kb as well as Mb. To analyze, you'll need to convert these to numeric.
# Extract the numeric value from the column
# multiply the value by 1,000, if size is mentioned in Mb
```

```
def Mtok(b):
    if b[len(b) -1: ]=='M':
        return (float(b[0:len(b)-1])*1000)
    elif b[len(b) -1: ]=='K' or b[len(b) -1: ]=='k':
        return (float(b[0:len(b)-1]))
    else:
        return b
```

```
df['Size']=pd.Series(df['Size'])
```

```
df['Size'] = df['Size'].apply(Mtok)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7723 entries, 0 to 10840
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  -
0   App                    7723 non-null   object
1   Category               7723 non-null   object
2   Rating                 7723 non-null   float64
3   Reviews                7723 non-null   object
4   Size                   7723 non-null   float64
5   Installs               7723 non-null   object
6   Type                   7723 non-null   object
7   Price                  7723 non-null   object
8   Content Rating        7723 non-null   object
9   Genres                 7723 non-null   object
10  Last Updated           7723 non-null   object
11  Current Ver            7723 non-null   object
12  Android Ver            7723 non-null   object
dtypes: float64(2), object(11)
memory usage: 1.1+ MB
```

```
df.Size
```

```
0      19000.0
1      14000.0
2       8700.0
3      25000.0
4       2800.0
...
10833     619.0
10834     2600.0
10836    53000.0
10837     3600.0
10840    19000.0
Name: Size, Length: 7723, dtype: float64
```

```
df=df[df.Size!= "Varies with device"]
```

```
df.shape
```

```
(7723, 13)
```

```
df['Reviews'].dtype
```

```
dtype('O')
```

```
df['Reviews']=df['Reviews'].astype("int64")
```

```
df['Reviews'].dtype
```

```
dtype('int64')
```

```
""" 4.3 Installs field is currently stored as string and has values like 1,000,000+.
Treat 1,000,000+ as 1,000,000
remove '+', ',' from the field, convert it to integer"""
```

```
def remove_char(val):
    return(int(val.replace(',','').replace('+','')))
```

```
df.Installs=df.Installs.map(remove_char)
```

```
df.Installs
```

```
0      10000
1     500000
2     5000000
3     50000000
4      100000
...
10833    1000
10834     500
10836     500
10837     100
10840    10000000
Name: Installs, Length: 7723, dtype: int64
```

```
# 4.4 Price field is a string and has $ symbol. Remove '$' sign,
# and convert it to numeric.
```

```
def remove_symbol(val):
    return(float(val.replace("$","")))
```

```
df.Price=df.Price.map(remove_symbol)
```

```
df['Price'].dtype
```

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

```
df.Price
```

```
0      0.0
1      0.0
2      0.0
3      0.0
4      0.0
...
10833    0.0
10834    0.0
10836    0.0
10837    0.0
10840    0.0
Name: Price, Length: 7723, dtype: float64
```

```
"""5.1 Sanity checks:
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.
"""
```

```
#way 1 to check
```

```
df[(df.Rating<1) | (df.Rating>5)]
```



```
# Way 2 to check
```

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

```
# Result : no rating is out side the range
```



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

```
df.loc[df.Reviews>df.Installs]
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genre
2454	KBA-EZ Health Guide	MEDICAL	5.0	4	25000.0	1	Free	0.00	Everyone	M
5917	Ra Ga Ba Brick	GAME	5.0	2	20000.0	1	Paid	1.49	Everyone	A

```
# Created a column to easily identify results for the syntax

df["RAI"]=np.where((df['Reviews'])<=(df["Installs"]),df['Installs'],np.nan)

df=df.dropna()

#now we have dropped all the reviews which were having greater than installs bu filling the NA.
#now lets drop the extra column RAI.

# to drop RAI column
df=df.drop(['RAI'],axis=1)

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

# To determine the rows with Price above $0
df.loc[df.Price > 0]
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price
234	TurboScan: scan documents and receipts in PDF	BUSINESS	4.7	11442	6800.0	100000	Paid	4
235	Tiny Scanner Pro: PDF Doc Scan	BUSINESS	4.8	10295	39000.0	100000	Paid	4
290	TurboScan: scan documents and receipts in PDF	BUSINESS	4.7	11442	6800.0	100000	Paid	4

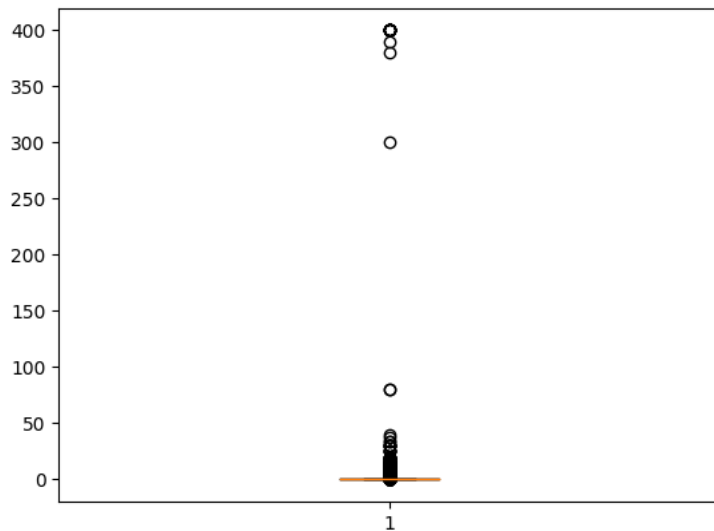
```
# check to confirm any free app with Price > 0
df[np.logical_and(df['Type'] == 'Free', df['Price'] > 0)]
```



```
#Performing univariate analysis:
#Boxplot for Price
#Are there any outliers? Think about the price of usual apps on Play Store.
```

```
plt.boxplot(df['Price'])
# most app prices are less that $100
# some outlier prices above $200
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7eba38acab90>,
<matplotlib.lines.Line2D at 0x7eba38ac8a60>],
'caps': [<matplotlib.lines.Line2D at 0x7eba38aca2c0>,
<matplotlib.lines.Line2D at 0x7eba38ac8460>],
'boxes': [<matplotlib.lines.Line2D at 0x7eba38aca920>],
'medians': [<matplotlib.lines.Line2D at 0x7eba38ac8910>],
'fliers': [<matplotlib.lines.Line2D at 0x7eba38ac9e10>],
'means': []}
```



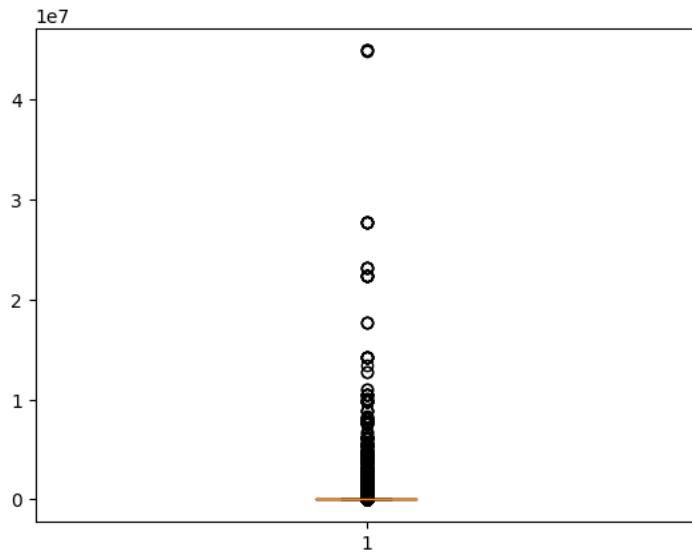
```
df['Price'].describe()
```

```
count    7717.000000
mean       1.128725
std       17.414784
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max       400.000000
Name: Price, dtype: float64
```

```
#Boxplot for Reviews
#Are there any apps with very high number of reviews? Do the values seem right?
```

```
plt.boxplot(df['Reviews'])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7eba388fb1f0>,
<matplotlib.lines.Line2D at 0x7eba388fb490>],
'caps': [<matplotlib.lines.Line2D at 0x7eba388fb730>,
<matplotlib.lines.Line2D at 0x7eba388fb9d0>],
'boxes': [<matplotlib.lines.Line2D at 0x7eba388faf50>],
'medians': [<matplotlib.lines.Line2D at 0x7eba388fbc70>],
'fliers': [<matplotlib.lines.Line2D at 0x7eba388fbf10>],
'means': []}
```



#Histogram for Rating

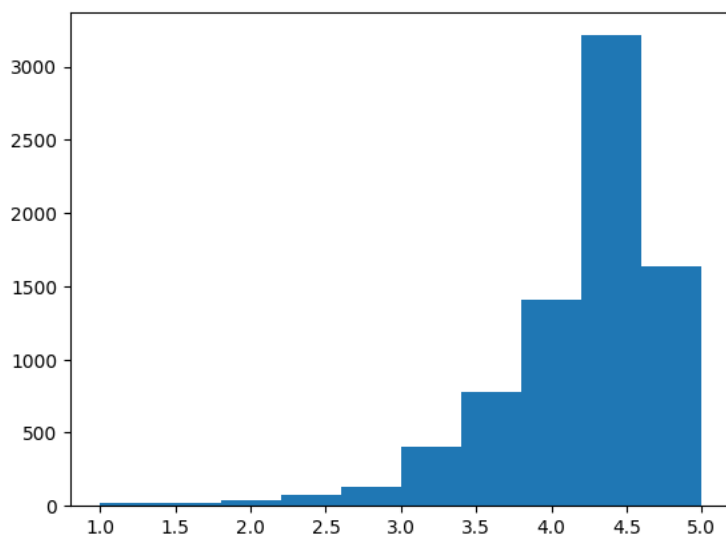
#How are the ratings distributed? Is it more toward higher ratings?

```
df['Reviews'].describe()
```

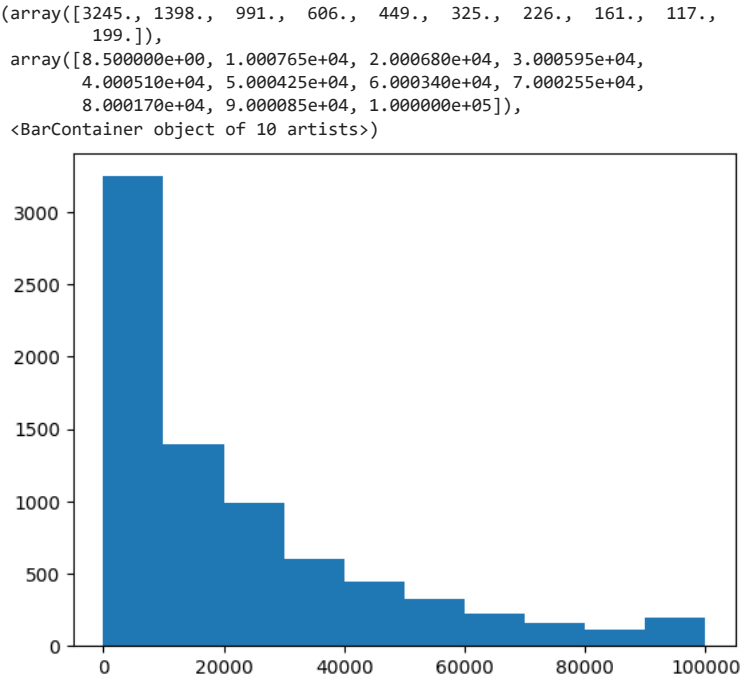
```
count    7.717000e+03
mean     2.951275e+05
std      1.864640e+06
min      1.000000e+00
25%      1.090000e+02
50%      2.351000e+03
75%      3.910900e+04
max      4.489389e+07
Name: Reviews, dtype: float64
```

```
plt.hist(df['Rating'])
```

```
(array([ 17.,  18.,  39.,  72., 132., 408., 781., 1406., 3212.,
        1632.]),
array([1. , 1.4, 1.8, 2.2, 2.6, 3. , 3.4, 3.8, 4.2, 4.6, 5. ]),
<BarContainer object of 10 artists>)
```



```
plt.hist(df['Size'])
```



```
# To determine the rows with Price above $100
# for my DF, I assume any app price greater than $100 is too high and should be dropped(outlier)
df.loc[df.Price > 100]
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Conte Rati
4197	most expensive app (H)	FAMILY	4.3	6	1500.0	100	Paid	399.99	Everyc
4362	ðŸŽŹ I'm rich	LIFESTYLE	3.8	718	26000.0	10000	Paid	399.99	Everyc
4367	I'm Rich - Trump Edition	LIFESTYLE	3.6	275	7300.0	10000	Paid	400.00	Everyc
5351	I am rich	LIFESTYLE	3.8	3547	1800.0	100000	Paid	399.99	Everyc
5354	I am Rich Plus	FAMILY	4.0	856	8700.0	10000	Paid	399.99	Everyc
5355	I am rich VIP	LIFESTYLE	3.8	411	2600.0	10000	Paid	299.99	Everyc
5356	I Am Rich Premium	FINANCE	4.1	1867	4700.0	50000	Paid	399.99	Everyc
5357	I am extremely Rich	LIFESTYLE	2.9	41	2900.0	1000	Paid	379.99	Everyc


```
# Second check and same result
df[df.Price>100]
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Conte Rati
4197	most expensive app (H)	FAMILY	4.3	6	1500.0	100	Paid	399.99	Everyc
4362	ðŸŽŹ I'm rich	LIFESTYLE	3.8	718	26000.0	10000	Paid	399.99	Everyc
4367	I'm Rich - Trump Edition	LIFESTYLE	3.6	275	7300.0	10000	Paid	400.00	Everyc
5351	I am rich	LIFESTYLE	3.8	3547	1800.0	100000	Paid	399.99	Everyc
5354	I am Rich Plus	FAMILY	4.0	856	8700.0	10000	Paid	399.99	Everyc
5355	I am rich VIP	LIFESTYLE	3.8	411	2600.0	10000	Paid	299.99	Everyc
5356	I Am Rich Premium	FINANCE	4.1	1867	4700.0	50000	Paid	399.99	Everyc
5357	I am extremely Rich	LIFESTYLE	2.9	41	2900.0	1000	Paid	379.99	Everyc

```
df[df.Price>100].shape
# 15 rows with outlier prices

(15, 13)

#df shape before the drop
df.shape

(7717, 13)

#df shape after the drop
df[df.Price <=100].shape

(7702, 13)

df = df[df.Price <=100]

#new df after the dropp
df.shape

(7702, 13)

# df accommodates all the required conditions on Price, Rating Reviews and Insatl1s
df.describe()
```

	Rating	Reviews	Size	Installs	Price
count	7702.000000	7.702000e+03	7702.000000	7.702000e+03	7702.000000
mean	4.173890	2.957011e+05	23004.020709	8.447011e+06	0.368802
std	0.544481	1.866409e+06	23466.178824	5.022383e+07	2.348127
min	1.000000	1.000000e+00	8.500000	5.000000e+00	0.000000
25%	4.000000	1.090000e+02	5300.000000	1.000000e+04	0.000000
50%	4.300000	2.374500e+03	14000.000000	1.000000e+05	0.000000
75%	4.500000	3.949125e+04	33000.000000	1.000000e+06	0.000000
max	5.000000	4.489389e+07	100000.000000	1.000000e+09	79.990000

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['Reviews'].describe()

count    7.702000e+03
mean     2.957011e+05
std      1.866409e+06
min      1.000000e+00
25%      1.090000e+02
50%      2.374500e+03
75%      3.949125e+04
max      4.489389e+07
Name: Reviews, dtype: float64
```

rows with Reviews more that 2million
df.loc[df.Reviews > 2000000]

	App	Category	Rating	Reviews	Size	Installs	Type	Price
345	Yahoo Mail & Stay Organized	COMMUNICATION	4.3	4187998	16000.0	100000000	Free	0.0
347	imo free video calls and chat	COMMUNICATION	4.3	4785892	11000.0	500000000	Free	0.0
366	UC Browser Mini -Tiny Fast Private & Secure	COMMUNICATION	4.4	3648120	3300.0	100000000	Free	0.0
370	UC Browser - Fast	COMMUNICATION	4.5	17710000	10000.0	500000000	Free	0.0

```
# second check
df[df.Reviews > 2000000]
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price
345	Yahoo Mail & Stay Organized	COMMUNICATION	4.3	4187998	16000.0	100000000	Free	0.0
347	imo free video calls and chat	COMMUNICATION	4.3	4785892	11000.0	500000000	Free	0.0
366	UC Browser Mini -Tiny Fast Private & Secure	COMMUNICATION	4.4	3648120	3300.0	100000000	Free	0.0
370	UC Browser - Fast	COMMUNICATION	4.5	4774000	4000.0	500000000	Free	0.0

```
# 219 rows to be dropped
df[df.Reviews > 2000000].shape

(219, 13)

df.shape

(7702, 13)

# new df
df = df[df.Reviews <= 2000000]

df.shape

(7483, 13)

df.describe()
```

```

    Rating      Reviews      Size      Installs      Price
count 7483.000000 7.483000e+03 7483.000000 7.483000e+03 7483.000000
mean   4.165789 7.260651e+04 22027.284177 3.947465e+06 0.379595
std    0.549946 2.123720e+05 22582.977041 2.781831e+07 2.381384
min    1.000000 1.000000e+00      8.500000 5.000000e+00 0.000000
25%    4.000000 9.900000e+01 5100.000000 1.000000e+04 0.000000
50%    4.300000 2.026000e+03 14000.000000 1.000000e+05 0.000000
75%    4.500000 3.238600e+04 31000.000000 1.000000e+06 0.000000
max    5.000000 1.986068e+06 100000.000000 1.000000e+09 79.990000

# Installs: There seems to be some outliers in this field too.
#Apps having very high number of installs should be dropped from the analysis.
#Find out the different percentiles - 10, 25, 50, 70, 90, 95, 99
#Decide a threshold as cutoff for outlier and drop records having values more than that

df['Installs'].describe()

count      7.483000e+03
mean       3.947465e+06
std        2.781831e+07
min        5.000000e+00
25%        1.000000e+04
50%        1.000000e+05
75%        1.000000e+06
max        1.000000e+09
Name: Installs, dtype: float64

np.arange(0,1,0.05)

array([0. , 0.05, 0.1 , 0.15, 0.2 , 0.25, 0.3 , 0.35, 0.4 , 0.45, 0.5 ,
       0.55, 0.6 , 0.65, 0.7 , 0.75, 0.8 , 0.85, 0.9 , 0.95])

# To get the percentiles by 0.05
df['Installs'].quantile(q = np.arange(0,1,0.05))

0.00      5.0
0.05     100.0
0.10     1000.0
0.15     1000.0
0.20     5000.0
0.25     10000.0
0.30     10000.0
0.35     10000.0
0.40     50000.0
0.45     100000.0
0.50     100000.0
0.55     100000.0
0.60     500000.0
0.65     1000000.0
0.70     1000000.0
0.75     1000000.0
0.80     5000000.0
0.85     5000000.0
0.90     10000000.0
0.95     10000000.0
Name: Installs, dtype: float64

df['Installs'].quantile(0.99)

50000000.0

# Installations above 10,000,000 are outliers and should be dropped
df[df.Installs > 10000000].head()

App      Category  Rating  Reviews  Size  Installs  Type  Price
3  Sketch - Draw & Paint  ART_AND_DESIGN  4.5  215644  25000.0  50000000  Free  0.0
194  OfficeSuite : Free Office +  BUSINESS  4.3  1002861  35000.0  100000000  Free  0.0
```

```
#second Check
df.loc[df.Installs>10000000]
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25000.0	50000000	Free	0.0
194	OfficeSuite : Free Office + PDF Editor	BUSINESS	4.3	1002861	35000.0	100000000	Free	0.0
225	Secure Folder	BUSINESS	3.8	14760	8600.0	50000000	Free	0.0
293	OfficeSuite : Free Office + PDF Editor	BUSINESS	4.3	1002859	35000.0	100000000	Free	0.0

imo beta

```
df = df[df.Installs <= 10000000]

df.describe()
```

	Rating	Reviews	Size	Installs	Price
count	7307.000000	7.307000e+03	7307.000000	7.307000e+03	7307.000000
mean	4.162899	5.091109e+04	21687.801765	1.716009e+06	0.388738
std	0.555276	1.457407e+05	22460.971012	3.205978e+06	2.409159
min	1.000000	1.000000e+00	8.500000	5.000000e+00	0.000000
25%	4.000000	9.100000e+01	5000.000000	1.000000e+04	0.000000
50%	4.300000	1.749000e+03	14000.000000	1.000000e+05	0.000000
75%	4.500000	2.755850e+04	30000.000000	1.000000e+06	0.000000
max	5.000000	1.736105e+06	100000.000000	1.000000e+07	79.990000

```

# Make scatter plot/joinplot for Rating vs. Price
# What pattern do you observe? Does rating increase with price?

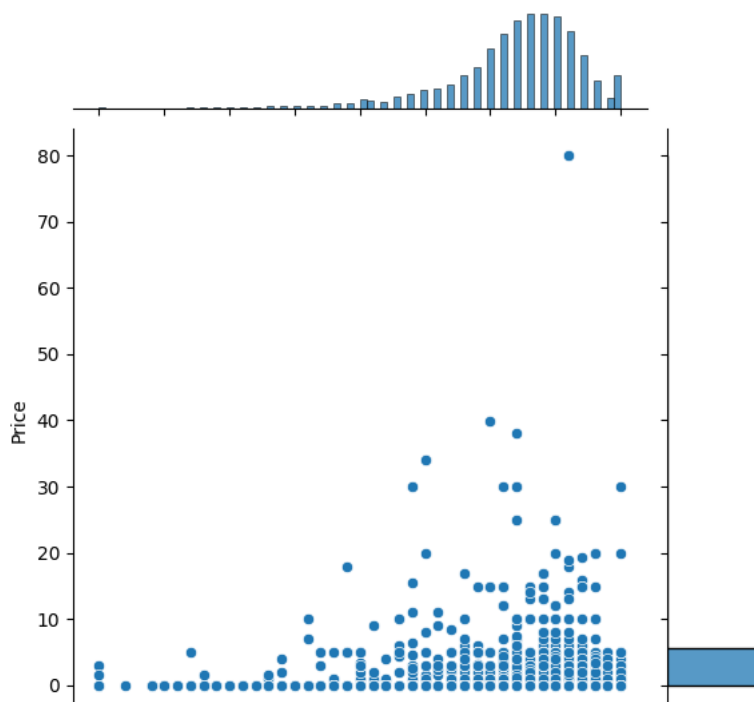
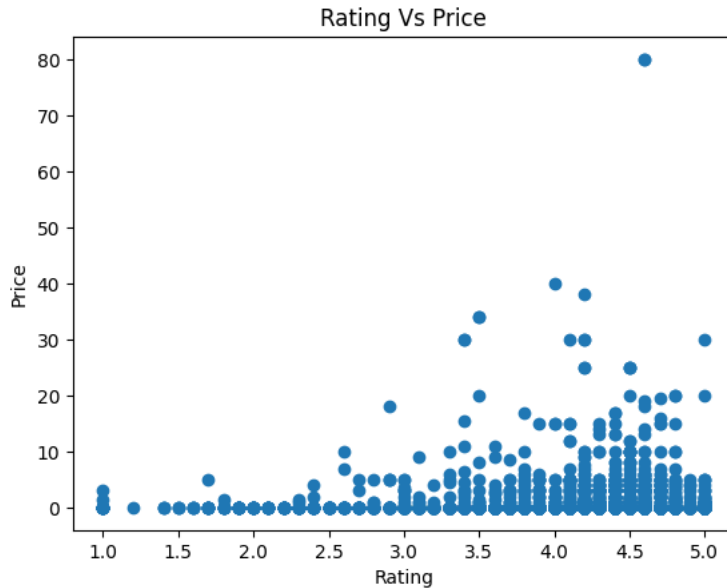
plt.scatter(x,y)

#x --> Rating
#y --> Price

plt.scatter(df['Rating'], df['Price'])
plt.xlabel('Rating')
plt.ylabel('Price')
plt.title('Rating Vs Price')
sns.jointplot(x=df['Rating'], y=df['Price'],data=df)
print('form the plot below, rating does not increase with price')

```

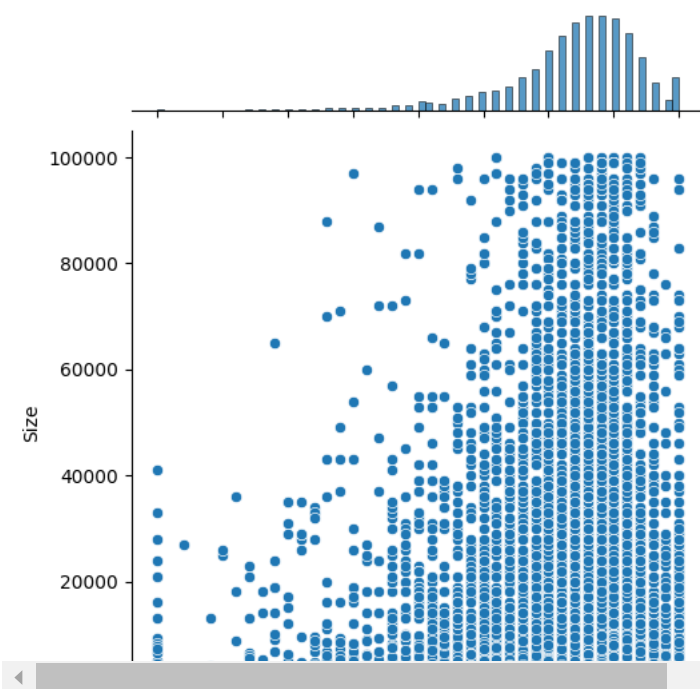
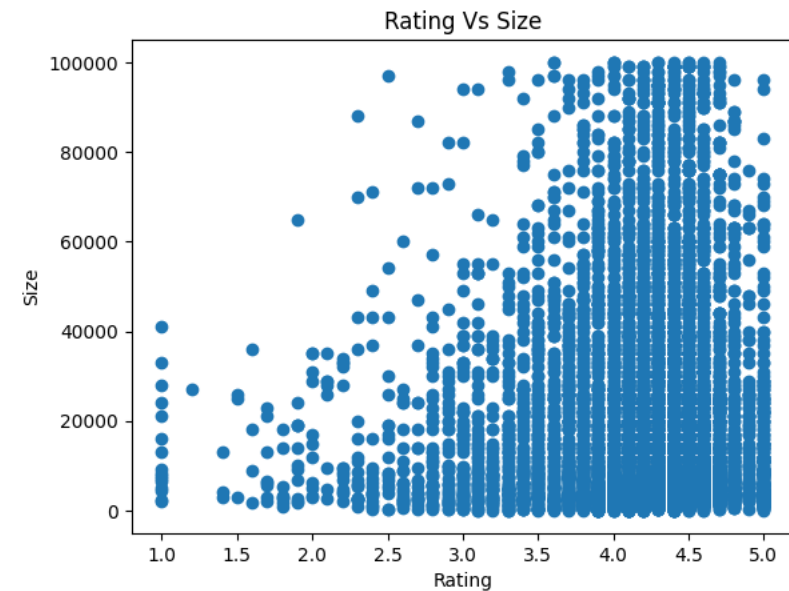
from the plot below, rating does not increase with price



```
# Make scatter plot/joinplot for Rating vs. Size
# Are heavier apps rated better?
```

```
plt.scatter(df['Rating'], df['Size'])
plt.xlabel('Rating')
plt.ylabel('Size')
plt.title('Rating Vs Size')
sns.jointplot(x=df['Rating'], y=df['Size'], data=df)
print('from the plot, lighter apps have less ratings than the heavier apps and are likely to be rated lower')
```

from the plot, lighter apps have less ratings than the heavier apps and are likely



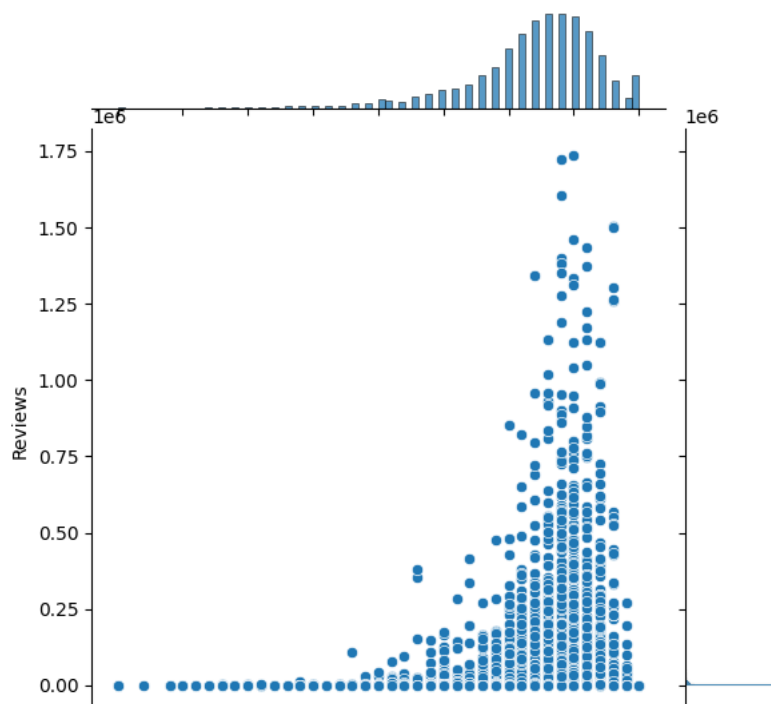
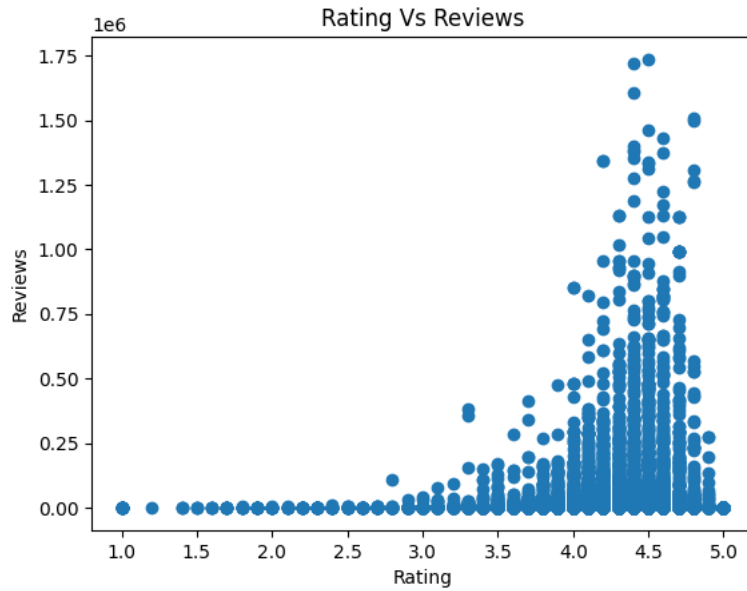
```
# Make scatter plot/joinplot for Rating vs. Reviews
# Does more review mean a better rating always?

plt.scatter(x,y)

#x --> Rating
#y --> Reviews

plt.scatter(df['Rating'], df['Reviews'])
plt.xlabel('Rating')
plt.ylabel('Reviews')
plt.title('Rating Vs Reviews')
sns.jointplot(x=df['Rating'], y=df['Reviews'],data=df)
print('from the plot, apps with the most reviews are rated highly')
```


from the plot, apps with the most reviews are rated highly



```
# 7.4 Make boxplot for Rating vs. Content Rating
```

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

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

```
print('Apps for Teens Content Rating are generally rated higher than others, while the apps for Everyone show a large variance in rating')
```

```
Apps for Teens Content Rating are generally rated higher than others, while the apps

#Make boxplot for Ratings vs. Category
# Which genre has the best ratings?
```

```
fig, axis = plt.subplots(figsize=(9, 15))
sns.boxplot(x=df['Rating'], y=df['Category'],data=df)
print('Apps for parenting and events show the highest ratings')
```

