DESCRIPTION Objective: Make a model to predict the app rating, with other information about the app provided. Problem Statement: Google Play Store team is about to launch a new feature wherein, certain apps that are promising, are boosted in visibility. The boost will manifest in multiple ways including higher priority in recommendations sections (“Similar apps”, “You might also like”, “New and updated games”). These will also get a boost in search results visibility. This feature will help bring more attention to newer apps that have the potential.

Domain: General Analysis to be done: The problem is to identify the apps that are going to be good for Google to promote. App ratings, which are provided by the customers, is always a great indicator of the goodness of the app. The problem reduces to: predict which apps will have high ratings.

Fields in the data – App: Application name

Category: Category to which the app belongs

Rating: Overall user rating of the app

Reviews: Number of user reviews for the app

Size: Size of the app

Installs: Number of user downloads/installs for the app

Type: Paid or Free

Price: Price of the app

Content Rating: Age group the app is targeted at - Children / Mature 21+ / Adult

Genres: An app can belong to multiple genres (apart from its main category). For example, a musical family game will belong to Music, Game, Family genres.

Last Updated: Date when the app was last updated on Play Store

Current Ver: Current version of the app available on Play Store

Android Ver: Minimum required Android version

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

Steps to perform:

In [95]:

```
# Load the data file using pandas.
```

In [3]:

```
inp0 = pd.read_csv('Downloads\googleplaystore.csv')
```

In [4]:

```
inp0.head()
```

Out[4]:

	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 [5]:

```
inp0.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        10840 non-null  object 
 2   Rating          9367 non-null   float64
 3   Reviews         10841 non-null  int64  
 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  10841 non-null  object 
 9   Genres          10840 non-null  object 
 10  Last Updated    10841 non-null  object 
 11  Current Ver     10833 non-null  object 
 12  Android Ver     10839 non-null  object 
dtypes: float64(1), int64(1), object(11)
memory usage: 1.1+ MB
```

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

In [6]:

```
inp0.isnull().sum()
```

Out[6]:

```
App                  0
Category             1
Rating            1474
Reviews              0
Size                 0
Installs             0
Type                 1
Price                0
Content Rating       0
Genres               1
Last Updated         0
Current Ver          8
Android Ver          2
dtype: int64
```

Dropping the records with null ratings

- this is done because ratings is our target variable

In [ ]:

```

```

In [7]:

```
inp0.dropna(how='any' , inplace = True) # any row that has a missing value drop it
```

In [8]:

```
inp0.isnull().sum()
```

Out[8]:

```
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
```

```
Confirming that the null records have been dropped
```

Change variable to correct types

In [9]:

```
inp0.dtypes
```

Out[9]:

```
App                object
Category           object
Rating            float64
Reviews             int64
Size               object
Installs           object
Type               object
Price              object
Content Rating     object
Genres             object
Last Updated       object
Current Ver        object
Android Ver        object
dtype: object
```

4.Variables seem to have incorrect type and inconsistent formatting. You need to fix them:

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

2.  Reviews is a numeric field that is loaded as a string field. Convert it to numeric (int/float).
3.  Installs field is currently stored as string and has values like 1,000,000+.
    a.  Treat 1,000,000+ as 1,000,000
    b.  remove ‘+’, ‘,’ from the field, convert it to integer
4.  Price field is a string and has $ symbol. Remove ‘$’ sign, and convert it to numeric.
```

4.4 Price column needs to be cleaned

In [10]:

```
inp0.Price.value_counts()[:5]
```

Out[10]:

```
0        8715
$2.99     114
$0.99     106
$4.99      70
$1.99      59
Name: Price, dtype: int64
```

```
Some have dollars, some have 0
- we need to conditionally handle this
- first, let's modify the column to take 0 if value is 0, else take the first letter onwards
```

In [11]:

```
def clean_price(x):
    if x == "0":
        return 0
    else:
        return float(x[1:])

inp0['Price'] = inp0.Price.map(clean_price)
```

4.2 Converting reviews to numeric

In [12]:

```
inp0['Reviews'] = inp0['Reviews'].astype('int32')
```

In [13]:

```
inp0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9360 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             9360 non-null   object 
 1   Category        9360 non-null   object 
 2   Rating          9360 non-null   float64
 3   Reviews         9360 non-null   int32  
 4   Size            9360 non-null   object 
 5   Installs        9360 non-null   object 
 6   Type            9360 non-null   object 
 7   Price           9360 non-null   float64
 8   Content Rating  9360 non-null   object 
 9   Genres          9360 non-null   object 
 10  Last Updated    9360 non-null   object 
 11  Current Ver     9360 non-null   object 
 12  Android Ver     9360 non-null   object 
dtypes: float64(2), int32(1), object(10)
memory usage: 987.2+ KB
```

```
### 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
```

In [14]:

```
inp0['Installs'] = inp0['Installs'].str.replace('+','').str.replace(',','')
```

In [15]:

```
inp0['Installs']= inp0['Installs'].astype('int32')
```

In [16]:

```
inp0.head(2)
```

Out[16]:

	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	10000	Free	0.0	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	500000	Free	0.0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0	4.0.3 and up

In [17]:

```
inp0.Installs.describe()
```

Out[17]:

```
count    9.360000e+03
mean     1.790875e+07
std      9.126637e+07
min      1.000000e+00
25%      1.000000e+04
50%      5.000000e+05
75%      5.000000e+06
max      1.000000e+09
Name: Installs, dtype: float64
```

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

In [18]:

```
### 4.1 Handling the app size field
```

In [19]:

```
# write a function 'change_size',
# if there is M which is size in MB, delete the last element, mutiply it with 1000 and convert it to float
# if there is k which is size in kB, delete the last element and convert it to float
# otherwise return None

def change_size(size):
    if "M" in size:
        x = size[:-1] # start : stop - 1
        x = float(x)*1000
        return x
    elif 'k' in size[-1]:
        x = size[:-1]
        x = float(x)
        return x
    else:
        return None
```

In [20]:

```
# use map to apply the function to the column as shown earlier
inp0["Size"] = inp0["Size"].map(change_size)
```

In [21]:

```
inp0.Size.describe()
```

Out[21]:

```
count      7723.000000
mean      22970.456105
std       23449.628935
min           8.500000
25%        5300.000000
50%       14000.000000
75%       33000.000000
max      100000.000000
Name: Size, dtype: float64
```

In [22]:

```
inp0["Size"].isnull().sum()
```

Out[22]:

```
1637
```

In [23]:

```
# filling Size which had NA
inp0.Size.fillna(method = 'ffill', inplace = True) # the missing values are introduced when we are working on the data type
```

In [24]:

```
inp0.dtypes
```

Out[24]:

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

5. Some sanity checks

1. 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.
2. Reviews should not be more than installs as only those who installed can review the app. If there are any such records, drop them.
3. For free apps (type = "Free"), the price should not be >0. Drop any such rows.

5.1 Avg. rating should be between 1 and 5, as only these values are allowed on the play store. Drop any rows that have a value outside this range.

In [25]:

```
inp0.Rating.describe()
```

Out[25]:

```
count    9360.000000
mean        4.191838
std         0.515263
min         1.000000
25%         4.000000
50%         4.300000
75%         4.500000
max         5.000000
Name: Rating, dtype: float64
```

```
Min is 1 and max is 5. Looks good.
```

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

In [26]:

```
inp0.loc[inp0.Reviews >= inp0.Installs] ##Checking if reviews are more than installs. Counting total rows like this.
```

Out[26]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
2454	KBA-EZ Health Guide	MEDICAL	5.0	4	25000.0	1	Free	0.00	Everyone	Medical	August 2, 2018	1.0.72	4.0.3 and up
4663	Alarmy (Sleep If U Can) - Pro	LIFESTYLE	4.8	10249	30000.0	10000	Paid	2.49	Everyone	Lifestyle	July 30, 2018	Varies with device	Varies with device
5917	Ra Ga Ba	GAME	5.0	2	20000.0	1	Paid	1.49	Everyone	Arcade	February 8, 2017	1.0.4	2.3 and up
6183	Revita.bg	HEALTH_AND_FITNESS	4.8	10	4000.0	10	Free	0.00	Everyone	Health & Fitness	June 13, 2018	3.55	4.0 and up
6700	Brick Breaker BR	GAME	5.0	7	19000.0	5	Free	0.00	Everyone	Arcade	July 23, 2018	1	4.1 and up
7147	CB Heroes	SOCIAL	5.0	5	1800.0	5	Free	0.00	Everyone	Social	August 4, 2018	1.2.4	5.0 and up
7402	Trovami se ci riesci	GAME	5.0	11	6100.0	10	Free	0.00	Everyone	Arcade	March 11, 2017	0.1	2.3 and up
8591	DN Blog	SOCIAL	5.0	20	4200.0	10	Free	0.00	Teen	Social	July 23, 2018	1	4.0 and up
10697	Mu.F.O.	GAME	5.0	2	16000.0	1	Paid	0.99	Everyone	Arcade	March 3, 2017	1	2.3 and up

In [27]:

```
len(inp0.loc[inp0.Reviews >= inp0.Installs])
```

Out[27]:

9

In [28]:

```
inp0.loc[inp0.Reviews >= inp0.Installs].index
```

Out[28]:

```
Int64Index([2454, 4663, 5917, 6183, 6700, 7147, 7402, 8591, 10697], dtype='int64')
```

In [29]:

```
inp0.drop(index =[2454, 4663, 5917, 6183, 6700, 7147, 7402, 8591, 10697] , axis = 0 ,inplace = True )
```

In [30]:

```
inp0.shape
```

Out[30]:

(9351, 13)

In [31]:

```
inp0.Reviews.shape
```

Out[31]:

(9351,)

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

In [32]:

```
len(inp0[(inp0.Type == "Free") & (inp0.Price>0)])
```

Out[32]:

0

5.A. Performing univariate analysis:

5.A. Performing univariate analysis:

- Boxplot for Price o Are there any outliers? Think about the price of usual apps on Play Store.
- Boxplot for Reviews o Are there any apps with very high number of reviews? Do the values seem right?
- Histogram for Rating o How are the ratings distributed? Is it more toward higher ratings?
- Histogram for Size

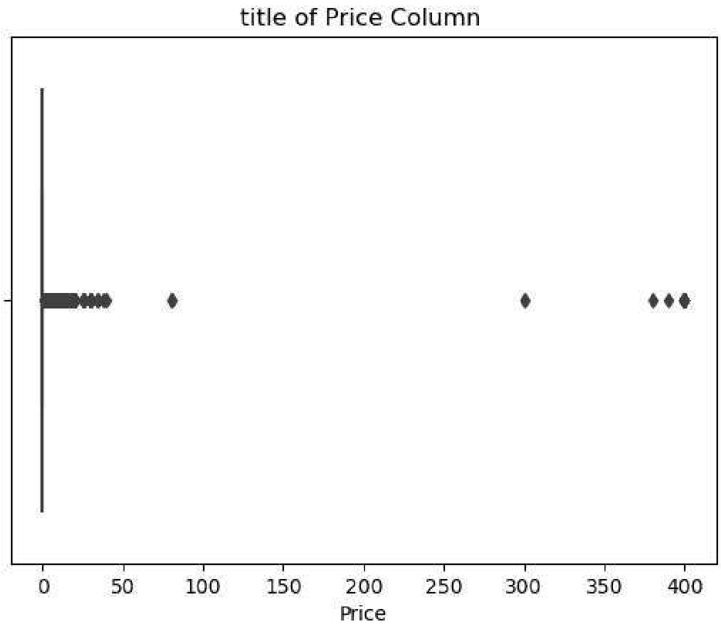
Note down your observations for the plots made above. Which of these seem to have outliers?

Box plot for price

o Are there any outliers? Think about the price of usual apps on Play Store.

In [33]:

```
sns.boxplot(inp0.Price).set(title = "title of Price Column ");
```



In [34]:

```
inp0.Price.describe()
```

Out[34]:

```
count    9351.000000
mean        0.961673
std        15.829226
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max       400.000000
Name: Price, dtype: float64
```

In [35]:

```
Q1 = inp0['Price'].quantile(0.25)
Q2 = inp0["Price"].quantile(0.50)
Q3 = inp0['Price'].quantile(0.75)

# 25% Apps price are less than or equal to zero
# 50% Apps price are less than or equal to zero
# 75% Apps price are less than or equal to zero

IQR = Q3-Q1
print(IQR)
```

```
0.0
```

In [36]:

```
Lower_limit = (Q1 -1.5*IQR) # Data points less than Lowerlimits are outliers
Upper_limit = (Q3 + 1.5*IQR) # Data points higher than upper limits are outliers
print(Lower_limit)
print(Upper_limit)
```

```
0.0
0.0
```

It is observed that price coloumn has outliers, some apps price are significantly high. Most of the app's price are zero

In [ ]:

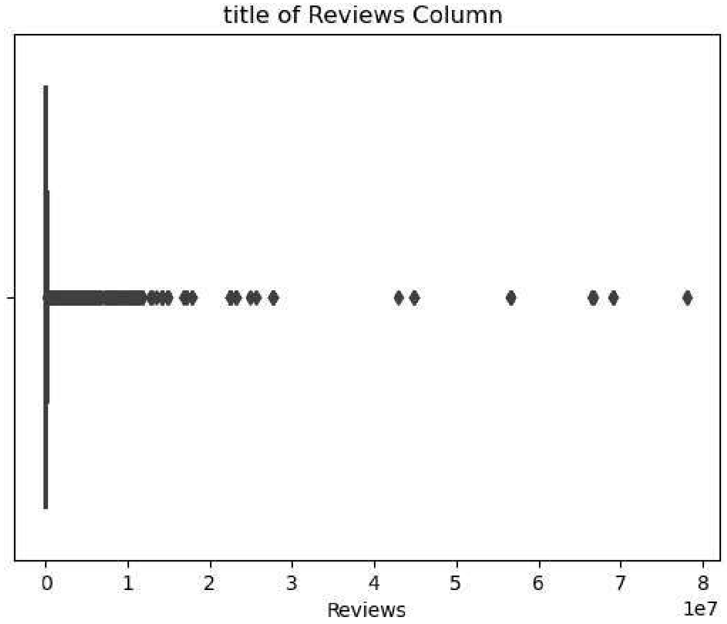
```

```

Box plot for Reviews

o Are there any apps with very high number of reviews? Do the values seem right?

In [37]:

```
sns.boxplot(inp0.Reviews).set(title = "title of Reviews Column ");
```



In [38]:

```
inp0.Reviews.describe()
```

Out[38]:

```
count    9.351000e+03
mean     5.148707e+05
std      3.146496e+06
min      1.000000e+00
25%      1.880000e+02
50%      5.968000e+03
75%      8.187600e+04
max      7.815831e+07
Name: Reviews, dtype: float64
```

In [39]:

```
Q1 = inp0['Reviews'].quantile(0.25)
Q2 = inp0["Reviews"].quantile(0.50)
Q3 = inp0['Reviews'].quantile(0.75)
print(Q1)
print(Q2)
print(Q3)
```

```
188.0
5968.0
81876.0
```

In [40]:

```
IQR = Q3-Q1
print("IQR : " ,IQR)

Lower_limit = (Q1 -1.5*IQR) # Data points less than lowerlimits are outliers
Upper_limit = (Q3 + 1.5*IQR) # Data points higher than upper limits are outliers
print("Lower_limit : ", Lower_limit )
print("Upper_limit : ", Upper_limit )
```

```
IQR :  81688.0
Lower_limit :  -122344.0
Upper_limit :  204408.0
```

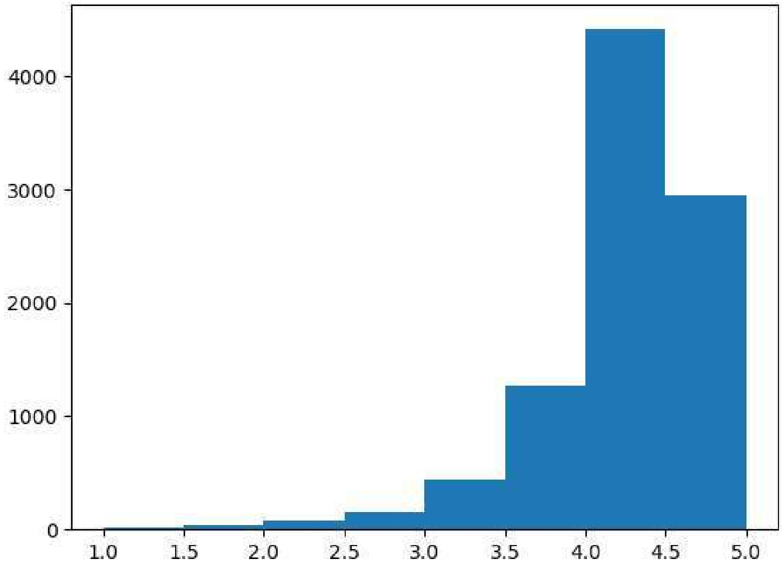
In [ ]:

Histogram for Rating

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

In [41]:

```
plt.hist(inp0.Rating , bins = 8);
```



In [42]:

```
# Rating is towards higher side but it can be seen that skew is on negative side
```

In [43]:

```
inp0.Rating.describe()
```

Out[43]:

```
count    9351.000000
mean        4.191103
std         0.514959
min         1.000000
25%         4.000000
50%         4.300000
75%         4.500000
max         5.000000
Name: Rating, dtype: float64
```

In [44]:

```
inp0.Rating.median()
```

Out[44]:

```
4.3
```

6. Outlier treatment:

#Price: From the box plot, it seems like there are some apps with very high price. A price of $200 for an application on the Play Store is very high and suspicious! a. Check out the records with very high price i. Is 200 indeed a high price? b. Drop these as most seem to be junk apps

#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.

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

6.1. Price: From the box plot, it seems like there are some apps with very high price. A price of $200 for an application on the Play Store is very high and suspicious!

```
    a.    Check out the records with very high price
        i.    Is 200 indeed a high price?
    b.    Drop these as most seem to be junk apps
```

In [45]:

```
len(inp0[(inp0.Price >= 200.00)]) ## Apps which are very high in price (greater than 200)
```

Out[45]:

```
15
```

In [46]:

```
inp0.drop(inp0.loc[(inp0.Price >= 200.00)].index,axis = 0,inplace = True)
```

In [47]:

```
inp0.shape
```

Out[47]:

```
(9336, 13)
```

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.

In [48]:

```
inp0.loc[inp0.Reviews > 2000000]
```

Out[48]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Currer Ve
139	Wattpad ☐ Free Books	BOOKS_AND_REFERENCE	4.6	2914724	3100.0	100000000	Free	0.0	Teen	Books & Reference	August 1, 2018	Varie wit devic
335	Messenger – Text and Video Chat for Free	COMMUNICATION	4.0	56642847	35000.0	1000000000	Free	0.0	Everyone	Communication	August 1, 2018	Varie wit devic
336	WhatsApp Messenger	COMMUNICATION	4.4	69119316	35000.0	1000000000	Free	0.0	Everyone	Communication	August 3, 2018	Varie wit devic
338	Google Chrome: Fast & Secure	COMMUNICATION	4.3	9642995	17000.0	1000000000	Free	0.0	Everyone	Communication	August 1, 2018	Varie wit devic
340	Gmail	COMMUNICATION	4.3	4604324	17000.0	1000000000	Free	0.0	Everyone	Communication	August 2, 2018	Varie wit devic
...	...	...	...	...	...	...	...	...	...	...	...	.
9166	Modern Combat 5: eSports FPS	GAME	4.3	2903386	58000.0	100000000	Free	0.0	Mature 17+	Action	July 24, 2018	3.2.1
9841	Google Earth	TRAVEL_AND_LOCAL	4.3	2339098	63000.0	100000000	Free	0.0	Everyone	Travel & Local	June 18, 2018	9.2.17.1
10186	Farm Heroes Saga	FAMILY	4.4	7615646	71000.0	100000000	Free	0.0	Everyone	Casual	August 7, 2018	5.2.
10190	Fallout Shelter	FAMILY	4.6	2721923	25000.0	10000000	Free	0.0	Teen	Simulation	June 11, 2018	1.13.1
10327	Garena Free Fire	GAME	4.5	5534114	53000.0	100000000	Free	0.0	Teen	Action	August 3, 2018	1.21.

453 rows × 13 columns

In [49]:

```
inp0.drop(inp0.loc[inp0.Reviews > 2000000].index , axis = 0, inplace = True)
```

In [50]:

```
inp0.shape
```

Out[50]:

```
(8883, 13)
```

6.3 Installs: There seems to be some outliers in this field too. Apps having very high number of installs should be dropped from the analysis.

```
a.    Find out the different percentiles – 10, 25, 50, 70, 90, 95, 99
b.    Decide a threshold as cutoff for outlier and drop records having values more than that
```

Dropping very high Installs values

In [51]:

```
inp0.Installs.quantile([0.1, 0.25, 0.5, 0.70, 0.9, 0.95, 0.99])
```

Out[51]:

```
0.10         1000.0
0.25        10000.0
0.50       500000.0
0.70      1000000.0
0.90     10000000.0
0.95     10000000.0
0.99    100000000.0
Name: Installs, dtype: float64
```

In [52]:

```
# It looks like there are just 1% apps having more than 100M installs. These apps might be genuine, but will definitely skew
# We need to drop these.
```

In [53]:

```
inp0.drop(inp0.loc[inp0.Installs >= 100000000].index, axis = 0,inplace=True)
```

In [54]:

```
inp0.shape
```

Out[54]:

```
(8741, 13)
```

7. Bivariate analysis: Let’s look at how the available predictors relate to the variable of interest, i.e., our target variable rating. Make scatter plots (for numeric features) and box plots (for character features) to assess the relations between rating and the other features.

```
1.    Make scatter plot/joinplot for Rating vs. Price
    a.    What pattern do you observe? Does rating increase with price?
2.    Make scatter plot/joinplot for Rating vs. Size
    a.    Are heavier apps rated better?
3.    Make scatter plot/joinplot for Rating vs. Reviews
    a.    Does more review mean a better rating always?
4.    Make boxplot for Rating vs. Content Rating
    a.    Is there any difference in the ratings? Are some types liked better?
5.    Make boxplot for Ratings vs. Category
    a.    Which genre has the best ratings?
```

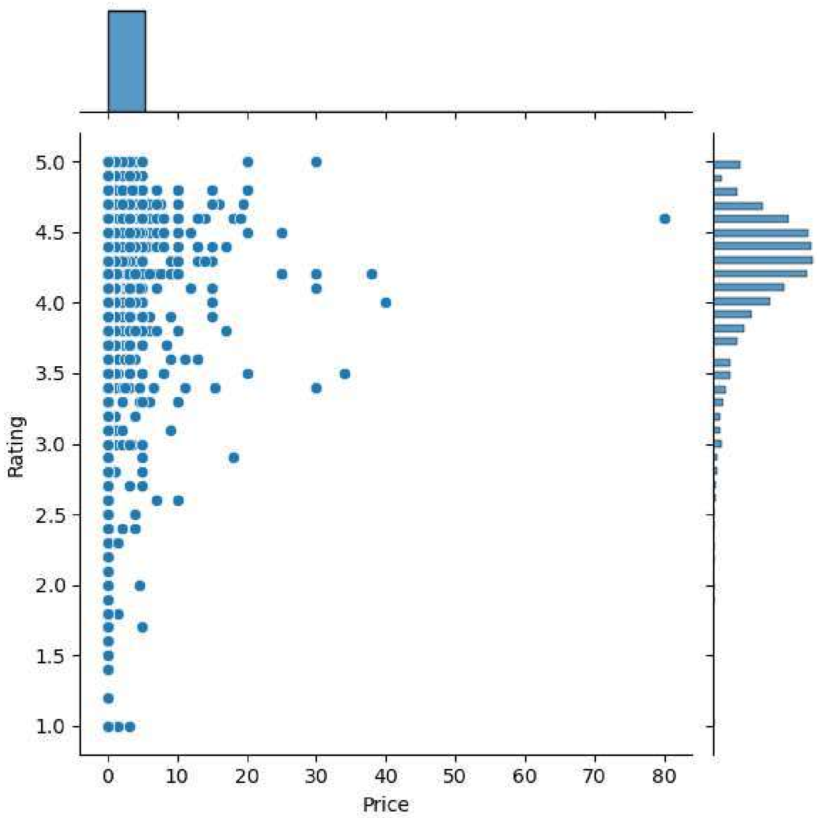
For each of the plots above, note down your observation.

7.1. Make scatter plot/joinplot for Rating vs Price

```
a.    What pattern do you observe? Does rating increase with price?
```

In [55]:

```
sns.jointplot(inp0.Price, inp0.Rating);
```



In [56]:

```
# most of the apps are price are zero. while few apps price are negligible higher with increase of rating
```

In [ ]:

```
```

7.2 Make scatter plot/joinplot for Rating vs Size

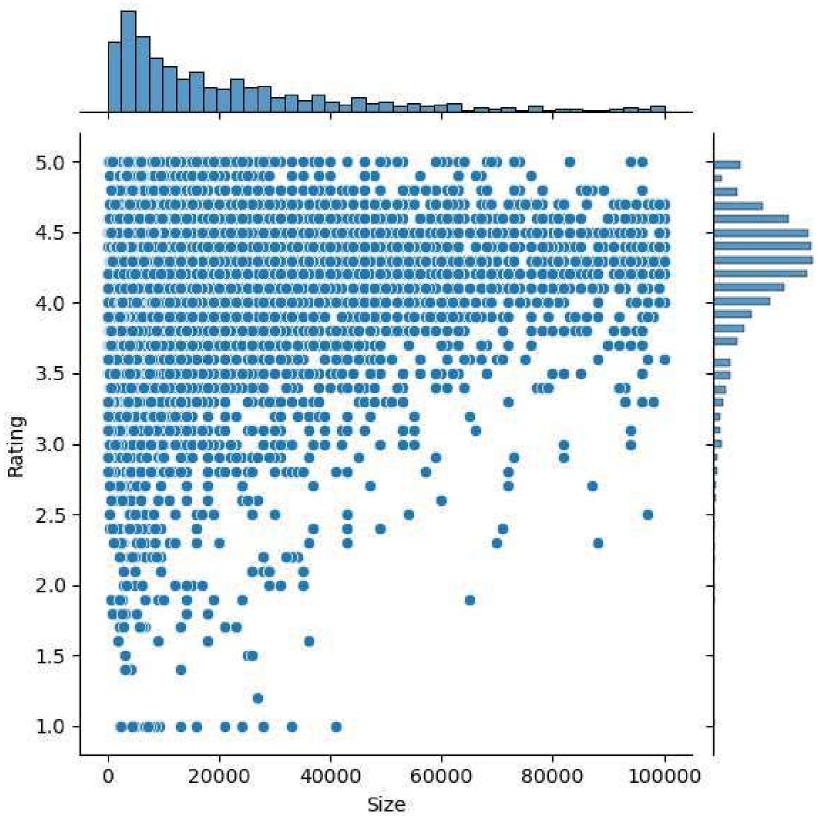
```
a.    Are heavier apps rated better?
```

In [57]:

```
sns.jointplot( x = inp0.Size , y = inp0.Rating , data = inp0)
```

Out[57]:

```
<seaborn.axisgrid.JointGrid at 0x18d03591820>
```



In [58]:

```
# its observed from the above jointplot that higher size app's rating are less that also mean people dont like or use app wh
#
```

In [ ]:

```
```

7.3 Make scatter plot/joinplot for Rating vs Reviews

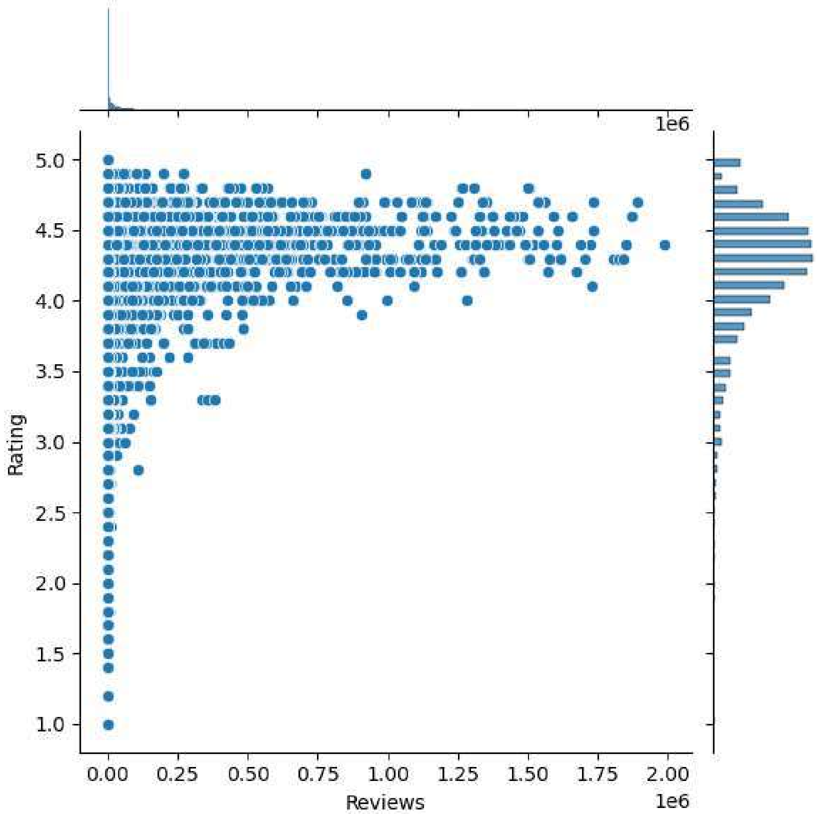
```
a.    Does more review mean a better rating always?
```

In [59]:

```
sns.jointplot( x = inp0.Reviews, y = inp0.Rating , data = inp0)
```

Out[59]:

<seaborn.axisgrid.JointGrid at 0x18d03ac9700>



In [60]:

```
# it does not certainly mean that more review mean a better rating. people give rating according to the precedence of app.
```

7.4 Make boxplot for Rating vs Content Rating

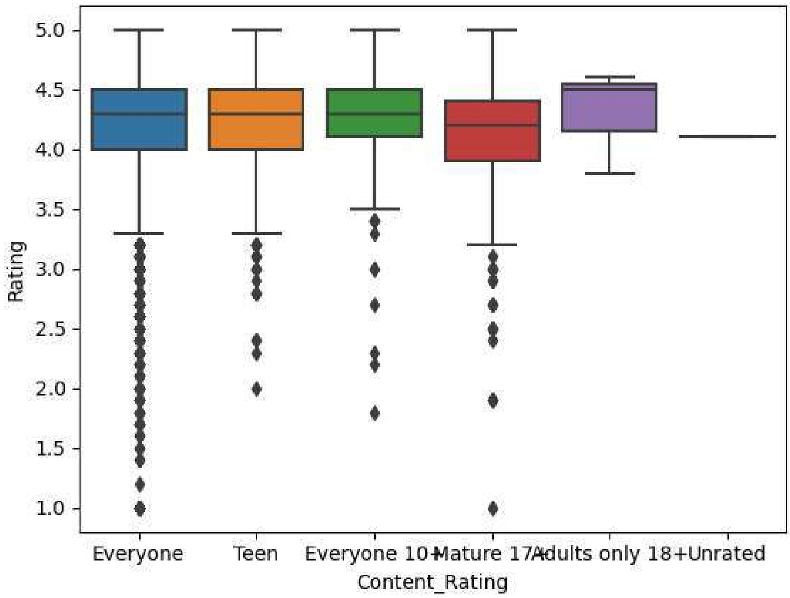
```
a.    Is there any difference in the ratings? Are some types liked better?
```

In [61]:

```
inp0.rename(columns = {"Content Rating" : "Content_Rating"} ,inplace = True)
```

In [62]:

```
sns.boxplot( x = inp0.Content_Rating , y =inp0.Rating , data = inp0 ,width=0.8 );
```



In [63]:

```
# Its noticed from the above boxplot that ratings are significantly different according to content.
# The contents for everyone, teen and Everyone 10+ are quite having same rating that also mean their choices are fairly same
# Rating range is outspread for mature 17+ content , this can also mean that diverse are different, different kind of people
#lastly adults only 18+ content related app remarkably higher.Rating range is also on top side though people are not enough
```

In [64]:

```
inp0.Content_Rating.unique()
```

Out[64]:

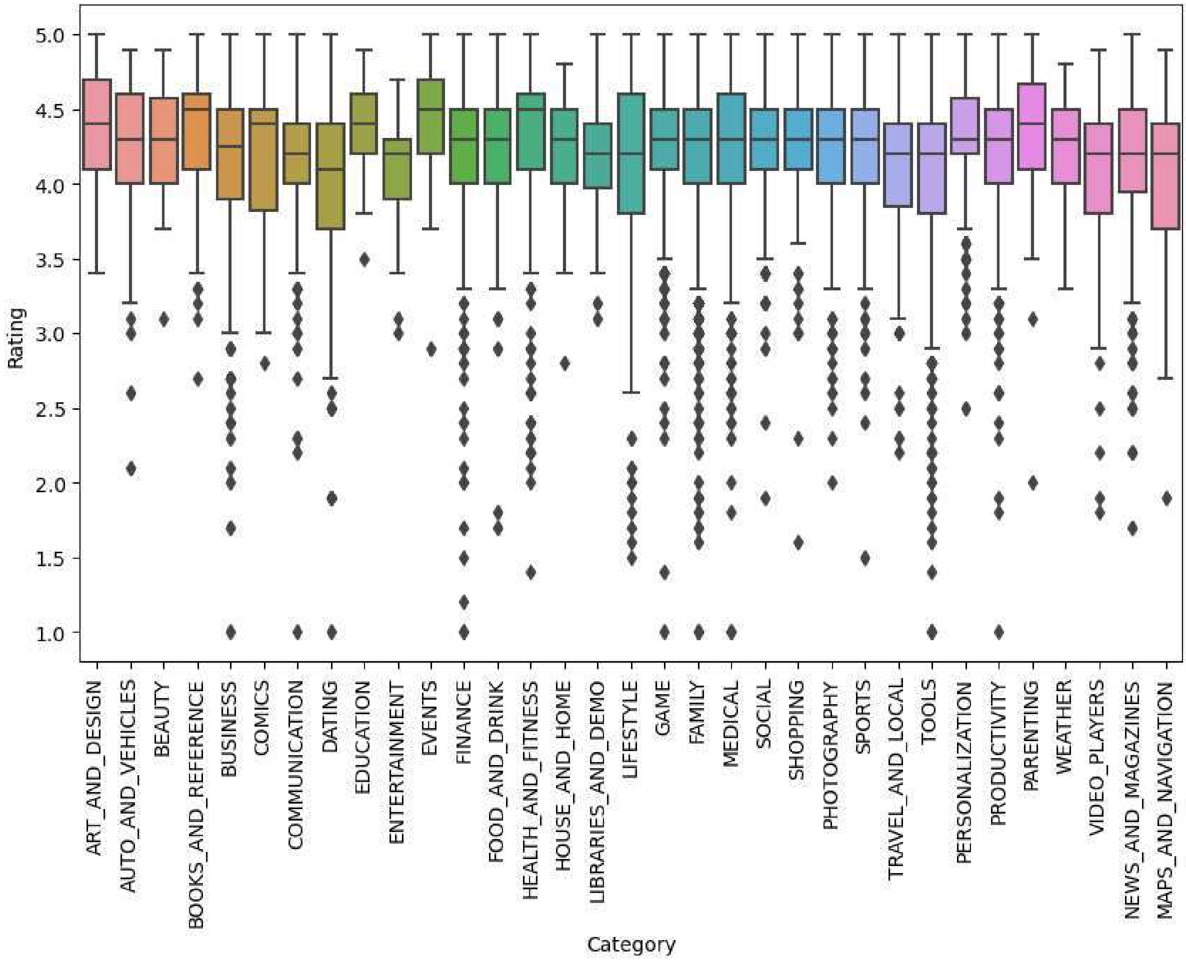
```
array(['Everyone', 'Teen', 'Everyone 10+', 'Mature 17+',
       'Adults only 18+', 'Unrated'], dtype=object)
```

7.5 Make boxplot for Ratings vs. Category

```
a.    Which genre has the best ratings?
```

In [65]:

```
plt.figure(figsize=[10,6]);
plt.xticks(rotation=90);
g=sns.boxplot(x = inp0.Category,y = inp0.Rating, data = inp0)
```



In [66]:

```
# Events Category has the best ratings out of all category
```

In [ ]:

8 Data preprocessing

For the steps below, create a copy of the dataframe to make all the edits. Name it inp1.

1. Reviews and Install have some values that are still relatively very high. Before building a linear regression model, you need to reduce the skew. Apply log transformation (np.log1p) to Reviews and Installs.
2. Drop columns App, Last Updated, Current Ver, and Android Ver. These variables are not useful for our task.
3. Get dummy columns for Category, Genres, and Content Rating. This needs to be done as the models do not understand categorical data, and all data should be numeric. Dummy encoding is one way to convert character fields to numeric. Name of dataframe should be inp2.

In [67]:

```
inp1 = inp0.copy()
```

8.1 Reviews and Install have some values that are still relatively very high. Before building a linear regression model, you need to reduce the skew. Apply log transformation (np.log1p) to Reviews and Installs.

In [68]:

```
inp1.Reviews.describe()
```

Out[68]:

```
count    8.741000e+03
mean     8.959908e+04
std      2.320747e+05
min      1.000000e+00
25%      1.500000e+02
50%      3.878000e+03
75%      5.029400e+04
max      1.986068e+06
Name: Reviews, dtype: float64
```

In [69]:

```
inp1.Reviews = inp1.Reviews.apply(np.log1p)
```

In [70]:

```
inp1.head(2)
```

Out[70]:

	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	5.075174	19000.0	10000	Free	0.0	Everyone	Art & Design	January 7, 2018	1.0.0	4.0.3 and up
1	Coloring book moana	ART_AND_DESIGN	3.9	6.875232	14000.0	500000	Free	0.0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0	4.0.3 and up

In [71]:

```
inp1.drop(columns = ["App", "Last Updated", "Current Ver", "Android Ver","Type"],axis = 1,inplace = True)
```

In [72]:

```
inp1.shape
```

Out[72]:

```
(8741, 8)
```

8.3 Get dummy columns for Category, Genres, and Content Rating. This needs to be done as the models do not understand categorical data, and all data should be numeric. Dummy encoding is one way to convert character fields to numeric. Name of dataframe should be inp2.

In [73]:

```
inp1 = pd.get_dummies(inp1 ,columns = ["Category","Content_Rating","Genres"], drop_first = True , prefix = ["Category","Cont
```

In [74]:

```
inp2 = pd.DataFrame(inp1)
```

In [75]:

```
inp2.shape
```

Out[75]:

```
(8741, 156)
```

9. Train test split and apply 70-30 split. Name the new dataframes df_train and df_test.

In [76]:

```
from sklearn.model_selection import train_test_split
```

In [77]:

```
df_train,df_test = train_test_split(inp2 , train_size = 0.7 , random_state =100)
```

In [78]:

```
df_train.shape, df_test.shape
```

Out[78]:

```
((6118, 156), (2623, 156))
```

10. Separate the dataframes into X_train, y_train, X_test, and y_test.

In [79]:

```
y_train = df_train.pop("Rating")
X_train = df_train
```

In [80]:

```
y_test = df_test.pop("Rating")
X_test = df_test
```

11 . Model building

- Use linear regression as the technique
- Report the R2 on the train set

In [92]:

```
from sklearn.linear_model import LinearRegression

linreg = LinearRegression()

#fit the model to the training data
linreg.fit(X_train,y_train)

#print the intercept and coefficients
print(linreg.intercept_,3) #b0
print(np.round(linreg.coef_,3)) #b1
```

```
3.81728994734337953 3
[ 0.031  0.    -0.     0.008  0.114  0.155  0.161  0.056  0.331  0.034
  0.011  0.056 -0.077  0.254  0.032  0.058  0.071  0.17   0.094  0.061
  0.108  0.072  0.017  0.122  0.08   0.069  0.171  0.042  0.082  0.092
  0.08   0.018  0.004  0.064  0.024  0.098 -0.06  -0.092 -0.068 -0.038
  0.051  0.228 -0.024  0.184  0.505 -0.124  0.026  0.297  0.354  0.407
  0.601 -0.095  0.114  0.155  0.12  -0.047  0.29  -0.     0.161  0.683
  0.056 -0.221  0.12   0.394  0.061 -0.032  0.045  0.31   0.252  0.298
 -0.027  0.065 -0.179  0.51   0.034 -0.     0.011  0.265  0.261  0.218
  0.297  0.25  -0.05   0.319 -0.172 -0.044  0.337 -0.157  0.233  0.113
  0.071  0.081  0.249  0.413 -0.     0.11  -0.032  0.254  0.058  0.071
  0.094 -0.     0.598  0.061  0.108  0.072  0.255 -0.     0.017  0.122
 -0.112  0.28   0.487  0.08   0.244  0.    -0.233  0.058  0.171  0.042
  0.082  0.249  0.114  0.214  0.273  0.59  -0.107  0.157  0.437  0.13
  0.18   0.114 -0.089  0.092  0.047  0.216  0.258  0.276  0.08   0.171
  0.192  0.066  0.488  0.214  0.453  0.004  0.     0.002  0.062  0.043
 -0.045 -0.071 -0.133  0.098  0.145]
```

In [93]:

```
# make predictions on the training data
y_pred = linreg.predict(X_train)
y_pred[:5]
```

Out[93]:

```
array([4.00951879, 4.0639892 , 4.43285835, 4.10598229, 4.37853721])
```

In [94]:

```
from sklearn.metrics import r2_score
print (r2_score( y_train,y_pred))
```

```
0.08318137936398051
```

12. Make predictions on test set and report R2.

In [82]:

```
# make predictions on the testing data
y_pred = linreg.predict(X_test)
y_pred[:5]
```

Out[82]:

```
array([3.97390765, 4.4312996 , 3.95893384, 3.9715273 , 3.96228427])
```

In [83]:

```
# mean squared error
np.square( y_test - y_pred).mean()
```

Out[83]:

```
0.24584213539728636
```

In [84]:

```
# R squared
1- (np.square( y_test - y_pred).mean())/np.square(y_test - y_test.mean()).sum()
```

Out[84]:

```
0.9996382311595329
```

In [ ]:

In [ ]: