App Rating Prediction python

Importing all the libraries

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Steps to perform:

1. Load the data file using pandas.

In [2]:

```
df = pd.read_csv('Dataset\googleplaystore.csv')
```

In [3]:

```
1 df
```

Out[3]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0
...	...	...	...	...	...	...	...	...
10836	Sya9a Maroc - FR	FAMILY	4.5	38	53M	5,000+	Free	0
10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3.6M	100+	Free	0
10838	Parkinson Exercices FR	MEDICAL	NaN	3	9.5M	1,000+	Free	0
10839	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5	114	Varies with device	1,000+	Free	0
10840	iHoroscope - 2018 Daily Horoscope & Astrology	LIFESTYLE	4.5	398307	19M	10,000,000+	Free	0

10841 rows × 13 columns

In [4]:

```
1 df.head()
```

Out[4]:

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

In [5]:

```
1 df.tail()
```

Out[5]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price
10836	Sya9a Maroc - FR	FAMILY	4.5	38	53M	5,000+	Free	0
10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3.6M	100+	Free	0
10838	Parkinson Exercices FR	MEDICAL	NaN	3	9.5M	1,000+	Free	0
10839	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5	114	Varies with device	1,000+	Free	0
10840	iHoroscope - 2018 Daily Horoscope & Astrology	LIFESTYLE	4.5	398307	19M	10,000,000+	Free	0

In [6]:

```
1 df.head(10)
```

Out[6]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	D
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	De
5	Paper flowers instructions	ART_AND_DESIGN	4.4	167	5.6M	50,000+	Free	0	Everyone	
6	Smoke Effect Photo Maker - Smoke Editor	ART_AND_DESIGN	3.8	178	19M	50,000+	Free	0	Everyone	
7	Infinite Painter	ART_AND_DESIGN	4.1	36815	29M	1,000,000+	Free	0	Everyone	
8	Garden Coloring Book	ART_AND_DESIGN	4.4	13791	33M	1,000,000+	Free	0	Everyone	
9	Kids Paint Free - Drawing Fun	ART_AND_DESIGN	4.7	121	3.1M	10,000+	Free	0	Everyone	De

In [7]:

```
1 # To check the shape of a DataFrame i.e. total no of rows & columns
```

In [8]:

```
1 df.shape
```

Out[8]:

```
(10841, 13)
```

In [9]:

```
1 # To check the info of the DataFrame such as index, columns, dtype, non-null values & me
```

In [10]:

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10841 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             10841 non-null  object 
 1   Category        10841 non-null  object 
 2   Rating          9367 non-null   float64
 3   Reviews         10841 non-null  object 
 4   Size            10841 non-null  object 
 5   Installs        10841 non-null  object 
 6   Type            10840 non-null  object 
 7   Price           10841 non-null  object 
 8   Content Rating  10840 non-null  object 
 9   Genres          10841 non-null  object 
 10  Last Updated    10841 non-null  object 
 11  Current Ver     10833 non-null  object 
 12  Android Ver     10838 non-null  object 
dtypes: float64(1), object(12)
memory usage: 1.1+ MB
```

In [11]:

```
1 # To find the description (statistical measures) of the data in the DataFrame such as co
```

In [12]:

```
1 df.describe()
```

Out[12]:

	Rating
count	9367.000000
mean	4.193338
std	0.537431
min	1.000000
25%	4.000000
50%	4.300000
75%	4.500000
max	19.000000

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

In [13]:

```
1 # sum() calculates the sum of elements for each row and column as True=1 and False=0
```

In [14]:

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

Out[14]:

```
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
```

In [15]:

```
df.notnull().sum()
```

Out[15]:

```
App               10841
Category          10841
Rating             9367
Reviews           10841
Size              10841
Installs          10841
Type              10840
Price             10841
Content Rating    10840
Genres            10841
Last Updated      10841
Current Ver       10833
Android Ver       10838
dtype: int64
```

3. Drop records with nulls in any of the columns.

In [16]:

```
# dropna() method removes the Rows/Columns that contains NULL values.
# inplace: It is a boolean which makes the changes in data frame itself if True.
#If (inplace = True) not specified, then returns a new DataFrame object
```

In [17]:

```
df.dropna(inplace = True)
```

In [18]:

```
df.shape
```

Out[18]:

```
(9360, 13)
```

In [19]:

```
# To reset index after droping null records
```

In [20]:

```
df = df.reset_index(drop=True)
df
```

	Draw & Paint	ART_AND_DESIGN	4.6	218044	25M	50,000,000+	Free	0	Teen	
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Desi
...	...	...	...	...	...	...	...	...	...	
9355	FR Calculator	FAMILY	4.0	7	2.6M	500+	Free	0	Everyone	
9356	Sya9a Maroc - FR	FAMILY	4.5	38	53M	5,000+	Free	0	Everyone	
9357	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3.6M	100+	Free	0	Everyone	
9358	The SCP Foundation	BOOKS_AND_REFERENCE	4.5	114	Varies with	1,000+	Free	0	Mature 17+	

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
2. Extract the numeric value from the column
3. Multiply the value by 1,000, if size is mentioned in Mb

In [21]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9360 entries, 0 to 9359
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   object 
 4   Size            9360 non-null   object 
 5   Installs        9360 non-null   object 
 6   Type            9360 non-null   object 
 7   Price           9360 non-null   object 
 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(1), object(12)
memory usage: 950.8+ KB
```

In [22]:

```
# To extract single column
# df["Size"] or df.Size
```

In [23]:

```
df["Size"]
```

Out[23]:

```
0                     19M
1                     14M
2                    8.7M
3                     25M
4                    2.8M
               ...
9355                 2.6M
9356                  53M
9357                 3.6M
9358    Varies with device
9359                  19M
Name: Size, Length: 9360, dtype: object
```

In [24]:

```
df["Size"] = [ (1000 * float(i.split('M')[0])) if 'M' in i else (float(i.split('k')[0])
```

In [25]:

```
1 df
```

Out[25]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19000.0	10,000+	Free	0
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14000.0	500,000+	Free	0
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8700.0	5,000,000+	Free	0
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25000.0	50,000,000+	Free	0
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2800.0	100,000+	Free	0
...	...	...	...	...	...	...	...	...
9355	FR Calculator	FAMILY	4.0	7	2600.0	500+	Free	0
9356	Sya9a Maroc - FR	FAMILY	4.5	38	53000.0	5,000+	Free	0
9357	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3600.0	100+	Free	0
9358	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5	114	0.0	1,000+	Free	0
9359	iHoroscope - 2018 Daily Horoscope & Astrology	LIFESTYLE	4.5	398307	19000.0	10,000,000+	Free	0

9360 rows × 13 columns

2. Reviews is a numeric field that is loaded as a string field. Convert it to numeric (int/float)

In [26]:

```
# df["Reviews"] = pd.to_numeric(df["Reviews"])
```

In [27]:

```
df["Reviews"] = df["Reviews"].astype(float)
```

In [28]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9360 entries, 0 to 9359
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   float64
 4   Size            9360 non-null   float64
 5   Installs        9360 non-null   object 
 6   Type            9360 non-null   object 
 7   Price           9360 non-null   object 
 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(3), object(10)
memory usage: 950.8+ KB
```

3. Installs field is currently stored as string and has values like 1,000,000+

1. Treat 1,000,000+ as 1,000,000
2. remove ‘+’, ‘,’ from the field, convert it to integer

In [29]:

```
# To display dataframe

df
```

Out[29]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159.0	19000.0	10,000+	Free	0	Everyone
1	Coloring book moana	ART_AND_DESIGN	3.9	967.0	14000.0	500,000+	Free	0	Everyone
2	U Launcher Lite – FREE Live Cool	ART_AND_DESIGN	4.7	87510.0	8700.0	5,000,000+	Free	0	Everyone

In [30]:

```
df["Installs"]
```

Out[30]:

```
0           10,000+
1          500,000+
2        5,000,000+
3       50,000,000+
4          100,000+
           ...
9355           500+
9356         5,000+
9357           100+
9358         1,000+
9359    10,000,000+
Name: Installs, Length: 9360, dtype: object
```

In [31]:

```
df["Installs"] = [ (i.replace('+','').replace(',', '')) if '+' in i or ',' in i else fl
```

In [32]:

```
df["Installs"] = df["Installs"].astype(int)
```

In [33]:

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9360 entries, 0 to 9359
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   float64
 4   Size            9360 non-null   float64
 5   Installs        9360 non-null   int32  
 6   Type            9360 non-null   object 
 7   Price           9360 non-null   object 
 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(3), int32(1), object(9)
memory usage: 914.2+ KB
```

In [34]:

```
1 df
```

Out[34]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159.0	19000.0	10000	Free	0	E
1	Coloring book moana	ART_AND_DESIGN	3.9	967.0	14000.0	500000	Free	0	E
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510.0	8700.0	5000000	Free	0	E
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644.0	25000.0	50000000	Free	0	
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967.0	2800.0	100000	Free	0	E
...	...	...	...	...	...	...	...	...	
9355	FR Calculator	FAMILY	4.0	7.0	2600.0	500	Free	0	E
9356	Sya9a Maroc - FR	FAMILY	4.5	38.0	53000.0	5000	Free	0	E
9357	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4.0	3600.0	100	Free	0	E
9358	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5	114.0	0.0	1000	Free	0	
9359	iHoroscope - 2018 Daily Horoscope & Astrology	LIFESTYLE	4.5	398307.0	19000.0	10000000	Free	0	E

9360 rows × 13 columns

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

In [35]:

```
1 df["Price"]
```

Out[35]:

```
0       0
1       0
2       0
3       0
4       0
       ..
9355    0
9356    0
9357    0
9358    0
9359    0
Name: Price, Length: 9360, dtype: object
```

In [36]:

```
1 df['Price'].unique()
```

Out[36]:

```
array(['0', '$4.99', '$3.99', '$6.99', '$7.99', '$5.99', '$2.99', '$3.49',
       '$1.99', '$9.99', '$7.49', '$0.99', '$9.00', '$5.49', '$10.00',
       '$24.99', '$11.99', '$79.99', '$16.99', '$14.99', '$29.99',
       '$12.99', '$2.49', '$10.99', '$1.50', '$19.99', '$15.99', '$33.99',
       '$39.99', '$3.95', '$4.49', '$1.70', '$8.99', '$1.49', '$3.88',
       '$399.99', '$17.99', '$400.00', '$3.02', '$1.76', '$4.84', '$4.77',
       '$1.61', '$2.50', '$1.59', '$6.49', '$1.29', '$299.99', '$379.99',
       '$37.99', '$18.99', '$389.99', '$8.49', '$1.75', '$14.00', '$2.00',
       '$3.08', '$2.59', '$19.40', '$3.90', '$4.59', '$15.46', '$3.04',
       '$13.99', '$4.29', '$3.28', '$4.60', '$1.00', '$2.95', '$2.90',
       '$1.97', '$2.56', '$1.20'], dtype=object)
```

In [37]:

```
1 df['Price'] = [ float (i.split('$')[1]) if '$' in i else float(0) for i in df['Price'] ]
```

In [38]:

```
1 df["Price"] = df["Price"].astype(int)
```

In [39]:

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9360 entries, 0 to 9359
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   float64
 4   Size            9360 non-null   float64
 5   Installs        9360 non-null   int32  
 6   Type            9360 non-null   object 
 7   Price           9360 non-null   int32  
 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(3), int32(2), object(8)
memory usage: 877.6+ KB
```

In [40]:

```
1 df.shape
```

Out[40]:

```
(9360, 13)
```

5. Sanity checks:

1. Average rating should be between 1 and 5 as only these values are allowed on th e 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.

In [41]:

```
df["Rating"]
```

Out[41]:

```
0       4.1
1       3.9
2       4.7
3       4.5
4       4.3
       ...
9355    4.0
9356    4.5
9357    5.0
9358    4.5
9359    4.5
Name: Rating, Length: 9360, dtype: float64
```

In [42]:

```
# Observation: Rating column is already in Range between 1 to 5
```

In [43]:

```
df = df[(df["Reviews"]<= df["Installs"])].reset_index(drop = True)
```

In [44]:

```
df.shape
```

Out[44]:

```
(9353, 13)
```

In [45]:

```
df.drop(df[(df['Type'] =='Free') & (df['Price'] > 0 )].index, inplace = True)
```

In [46]:

```
df.shape
```

Out[46]:

```
(9353, 13)
```

Performing univariate analysis:

• Boxplot for Price: Are there any outliers? Think about the price of usual apps on Play Store.

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

• Histogram for Rating: How are the ratings distributed? Is it more toward higher ratings?

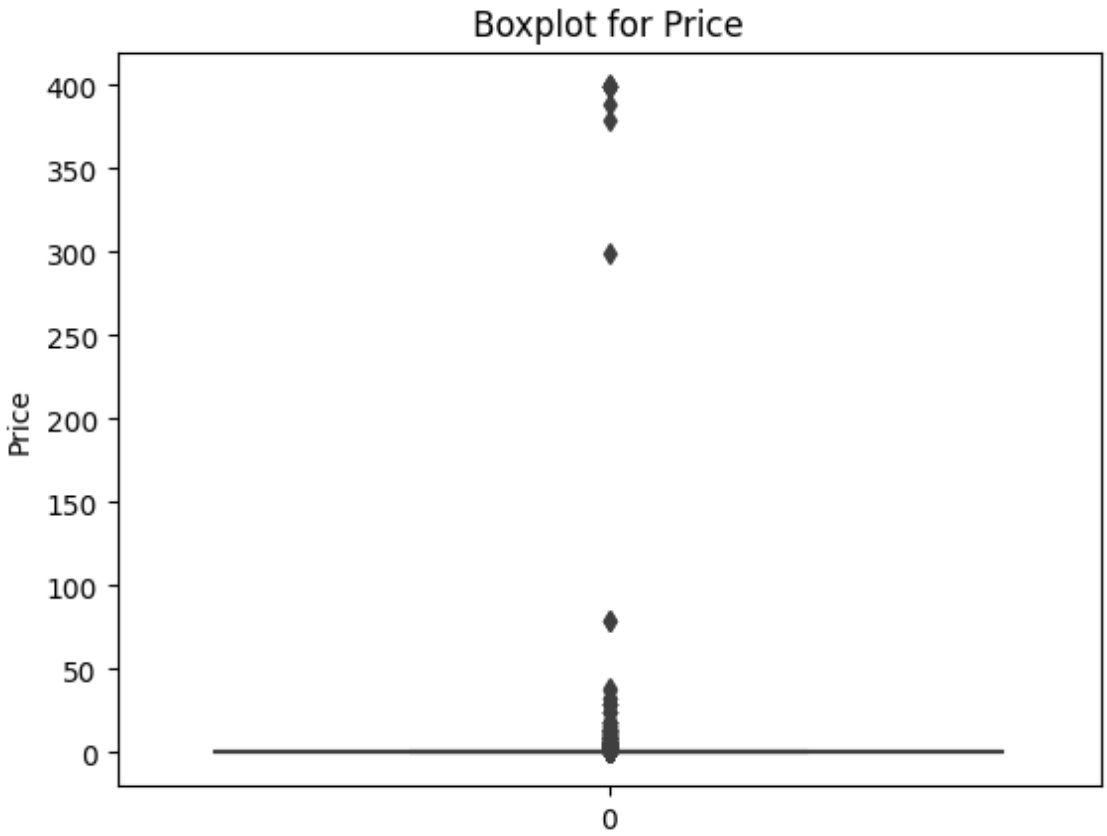
• Histogram for Size

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

Boxplot for Price

In [47]:

```
sns.boxplot(df["Price"])
plt.title("Boxplot for Price")
plt.ylabel("Price")
plt.show()
```



In [48]:

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

Out[48]:

```
count    9353.000000
mean        0.899711
std        15.776941
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max       400.000000
Name: Price, dtype: float64
```

In [49]:

```
# To round off to 2 decimels
```

In [50]:

```
round(df["Price"].describe(),2)
```

Out[50]:

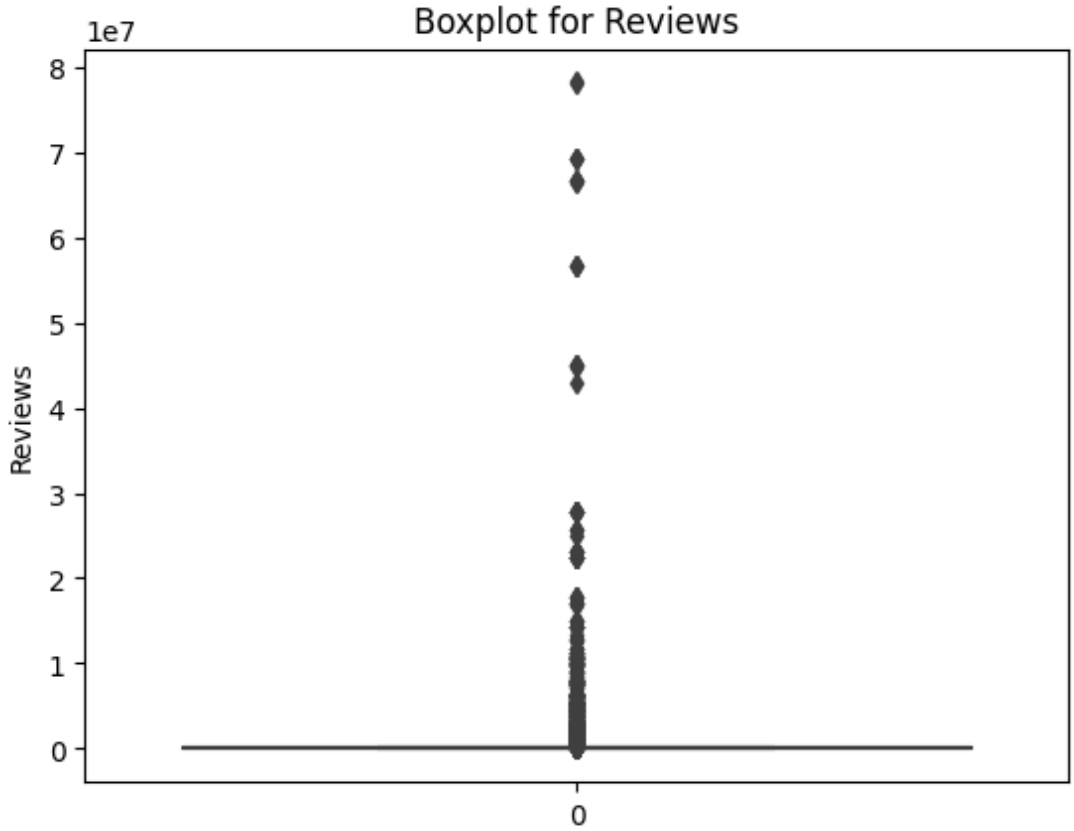
```
count    9353.00
mean        0.90
std        15.78
min         0.00
25%         0.00
50%         0.00
75%         0.00
max       400.00
Name: Price, dtype: float64
```

Observation : There are outliers and the price of usual apps on Play Store is 0.90.

Boxplot for Reviews

In [51]:

```
sns.boxplot(df["Reviews"])
plt.title("Boxplot for Reviews")
plt.ylabel("Reviews")
plt.show()
```



In [52]:

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

Out[52]:

```
count    9.353000e+03
mean     5.147606e+05
std      3.146169e+06
min      1.000000e+00
25%      1.870000e+02
50%      5.967000e+03
75%      8.174700e+04
max      7.815831e+07
Name: Reviews, dtype: float64
```

In [53]:

```
round(df["Reviews"].describe(),2)
```

Out[53]:

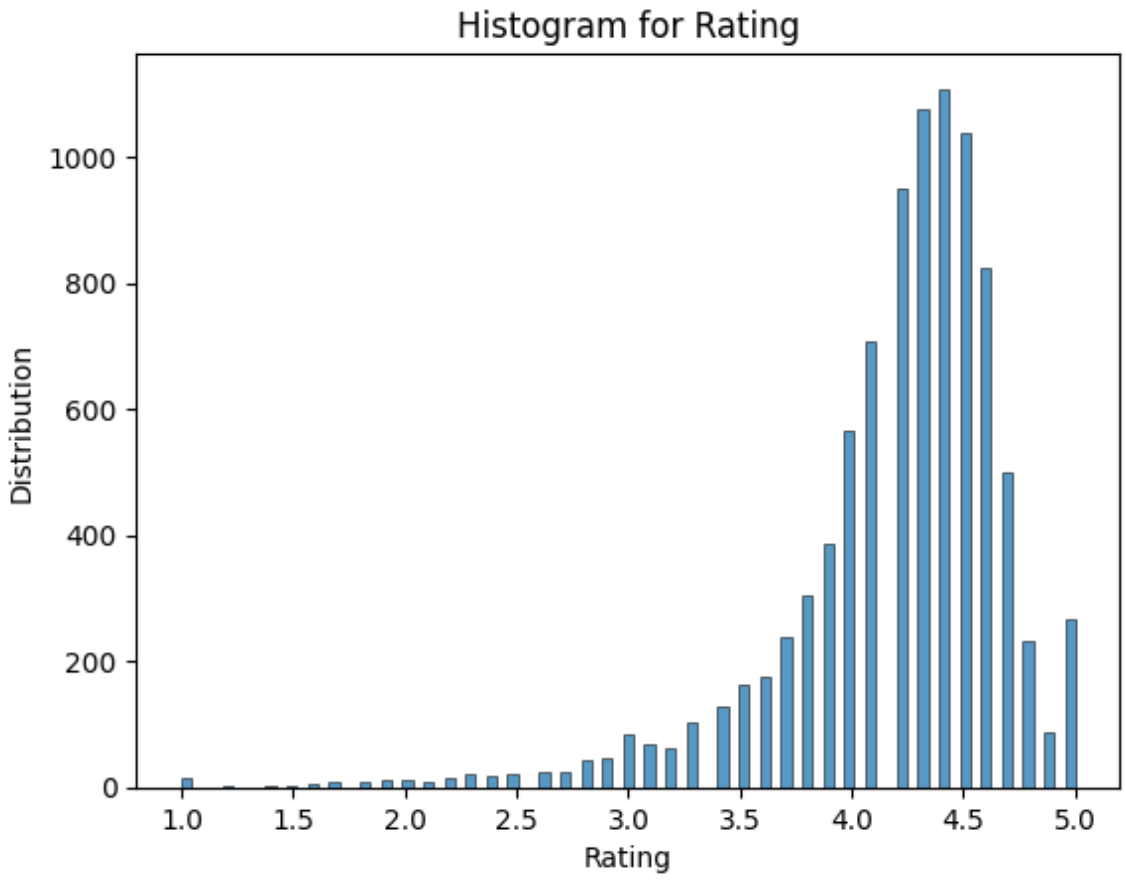
```
count        9353.00
mean       514760.58
std       3146168.75
min             1.00
25%           187.00
50%          5967.00
75%         81747.00
max      78158306.00
Name: Reviews, dtype: float64
```

Observation : There are apps with very high number of Reviews and the value 78158306 don't seem to be correct

Histogram for Rating

In [54]:

```
sns.histplot(df["Rating"])
plt.title("Histogram for Rating")
plt.xlabel("Rating")
plt.ylabel("Distribution")
plt.show()
```

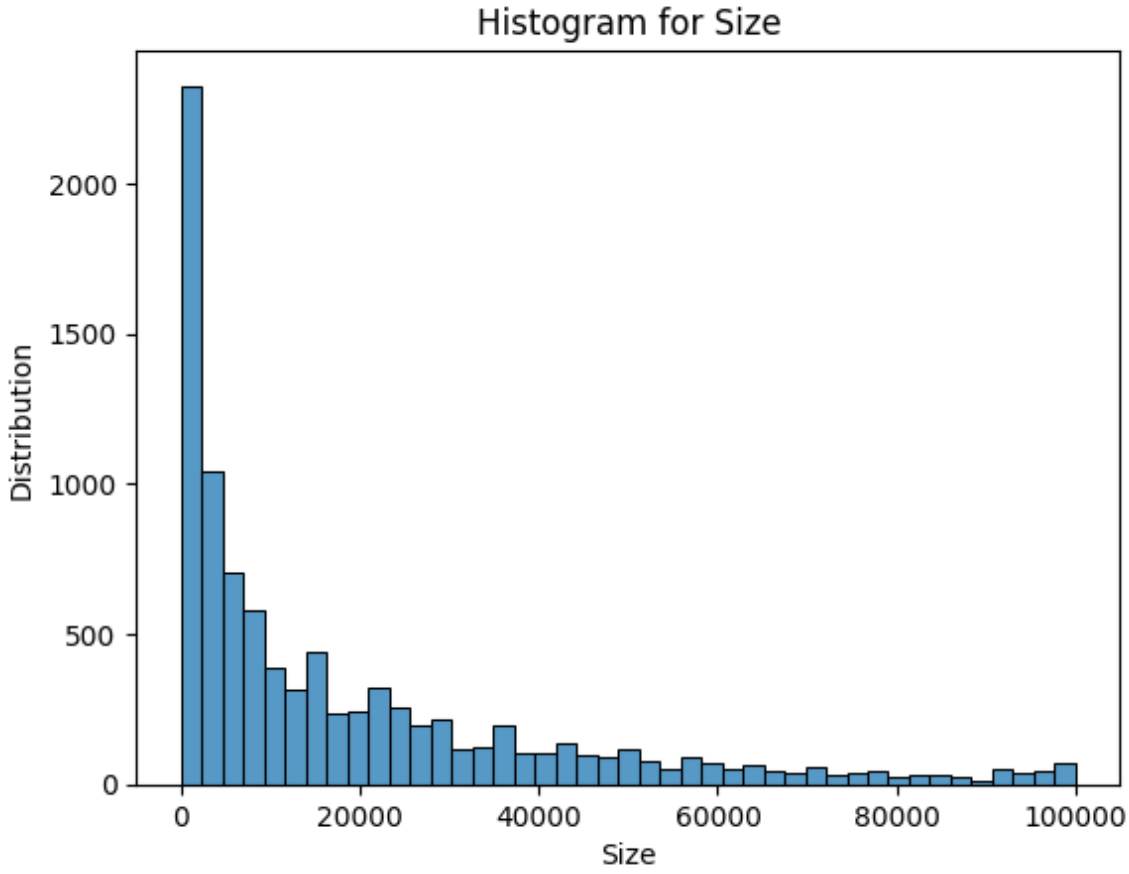


Observation : Ratings are left skewed distributed and it is more toward higher ratings.

Histogram for Size

In [55]:

```
sns.histplot(df["Size"])
plt.title("Histogram for Size")
plt.xlabel("Size")
plt.ylabel("Distribution")
plt.show()
```



Observation : Size are right skewed distributed.

6. Outlier treatment:

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

      a. Is 200 indeed a high price?

         ii. Drop these as most seem to be junk apps

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.

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

In [56]:

```
df.drop(df[df["Price"] > 200].index, inplace = True)
```

In [57]:

```
df.shape
```

Out[57]:

(9338, 13)

In [58]:

```
df.drop(df[df['Reviews'] > 2000000].index, inplace = True)
```

In [59]:

```
df.shape
```

Out[59]:

(8885, 13)

In [60]:

```
df.quantile([.1, .25, .5, .70, .90, .95, .99])
```

```
C:\Users\nidhi\AppData\Local\Temp\ipykernel_26168\2610616657.py:1: FutureWarn
ing: The default value of numeric_only in DataFrame.quantile is deprecated. I
n a future version, it will default to False. Select only valid columns or sp
ecify the value of numeric_only to silence this warning.
  df.quantile([.1, .25, .5, .70, .90, .95, .99])
```

Out[60]:

	Rating	Reviews	Size	Installs	Price
0.10	3.5	18.00	0.0	1000.0	0.0
0.25	4.0	159.00	2600.0	10000.0	0.0
0.50	4.3	4290.00	9500.0	500000.0	0.0
0.70	4.5	35930.40	23000.0	1000000.0	0.0
0.90	4.7	296771.00	50000.0	10000000.0	0.0
0.95	4.8	637298.00	68000.0	10000000.0	1.0
0.99	5.0	1462800.88	95000.0	100000000.0	7.0

In [61]:

```
# dropping more than 10000000 (threshold) Installs value

df.drop(df["Installs"] > 10000000].index, inplace = True)
```

In [62]:

```
df.shape
```

Out[62]:

```
(8496, 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

   b. Are heavier apps rated better?

3. Make scatter plot/joinplot for Rating vs. Reviews

   c. Does more review mean a better rating always?

4. Make boxplot for Rating vs. Content Rating

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

5. Make boxplot for Ratings vs. Category

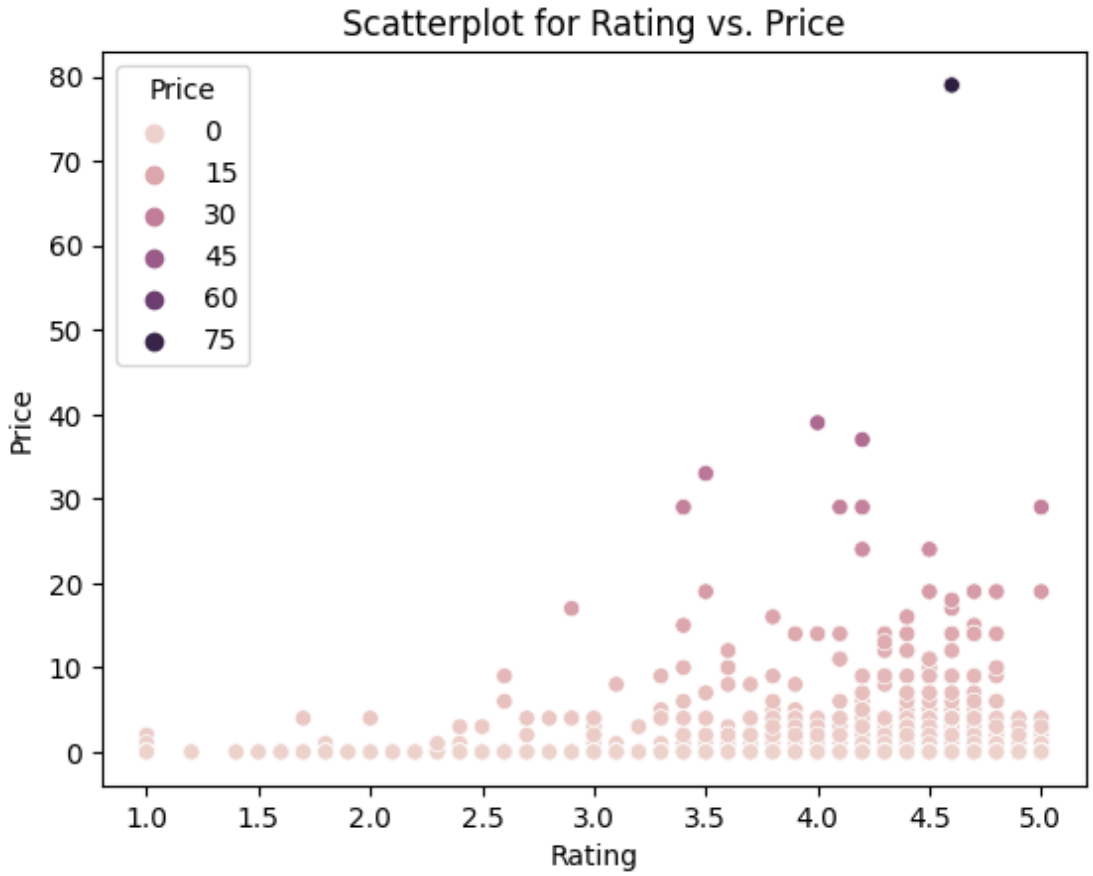
   f. Which genre has the best ratings?

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

Scatterplot for Rating vs. Price

In [63]:

```
sns.scatterplot(data=df, x="Rating", y="Price", hue="Price")
plt.title("Scatterplot for Rating vs. Price")
plt.xlabel("Rating")
plt.ylabel("Price")
plt.show()
```

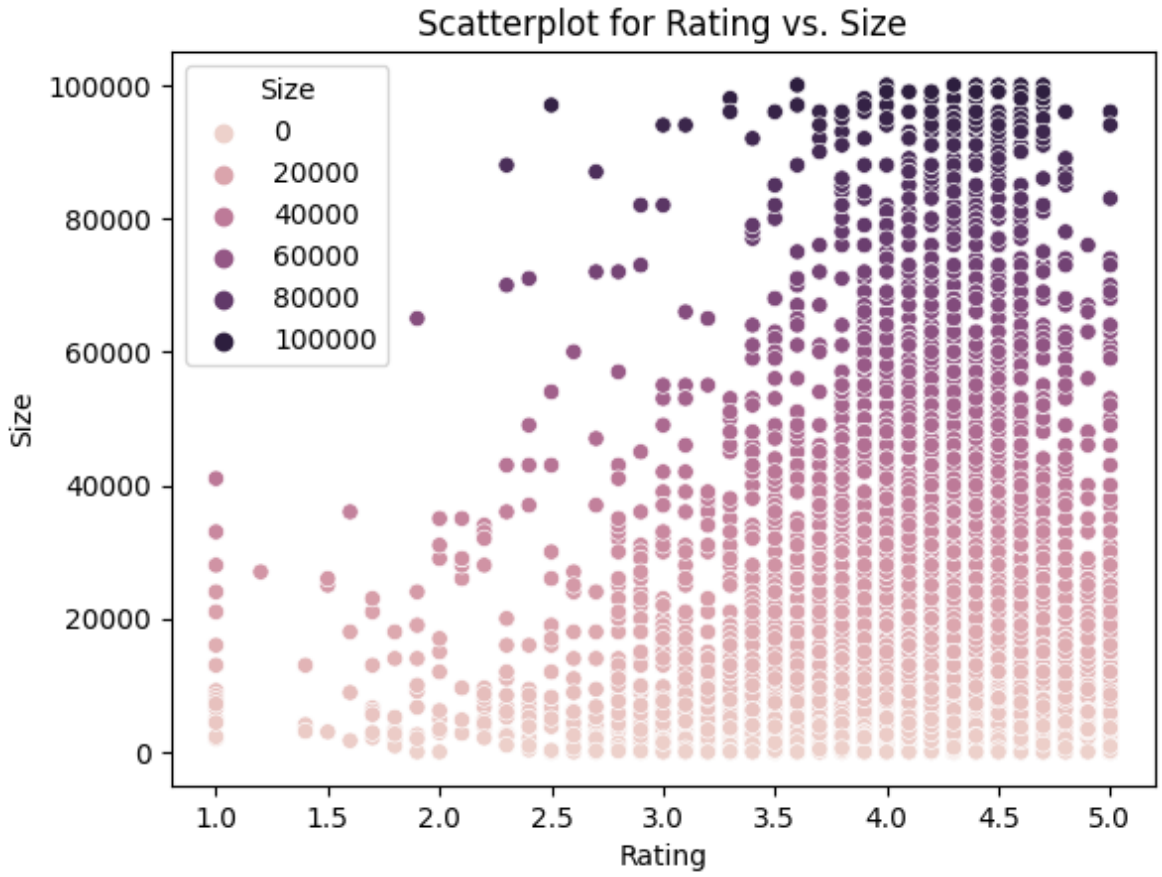


Observation : Paid apps have higher ratings as compared to free apps.

Scatterplot for Rating vs. Size

In [64]:

```
sns.scatterplot(data=df, x="Rating", y="Size", hue="Size")
plt.title("Scatterplot for Rating vs. Size")
plt.xlabel("Rating")
plt.ylabel("Size")
plt.show()
```

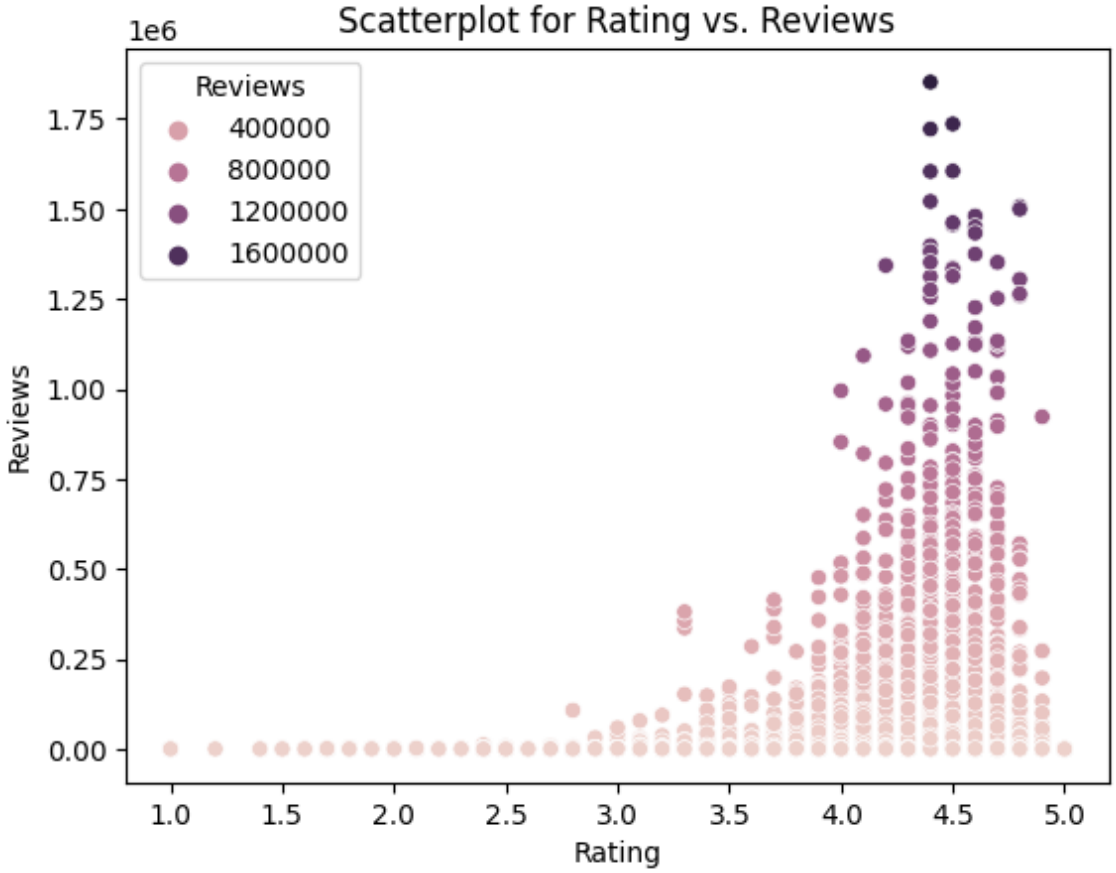


Observation : It is clear that heavier apps are rated better.

Scatterplot for Rating vs. Reviews

In [65]:

```
sns.scatterplot(data=df, x="Rating", y="Reviews", hue="Reviews")
plt.title("Scatterplot for Rating vs. Reviews")
plt.xlabel("Rating")
plt.ylabel("Reviews")
plt.show()
```

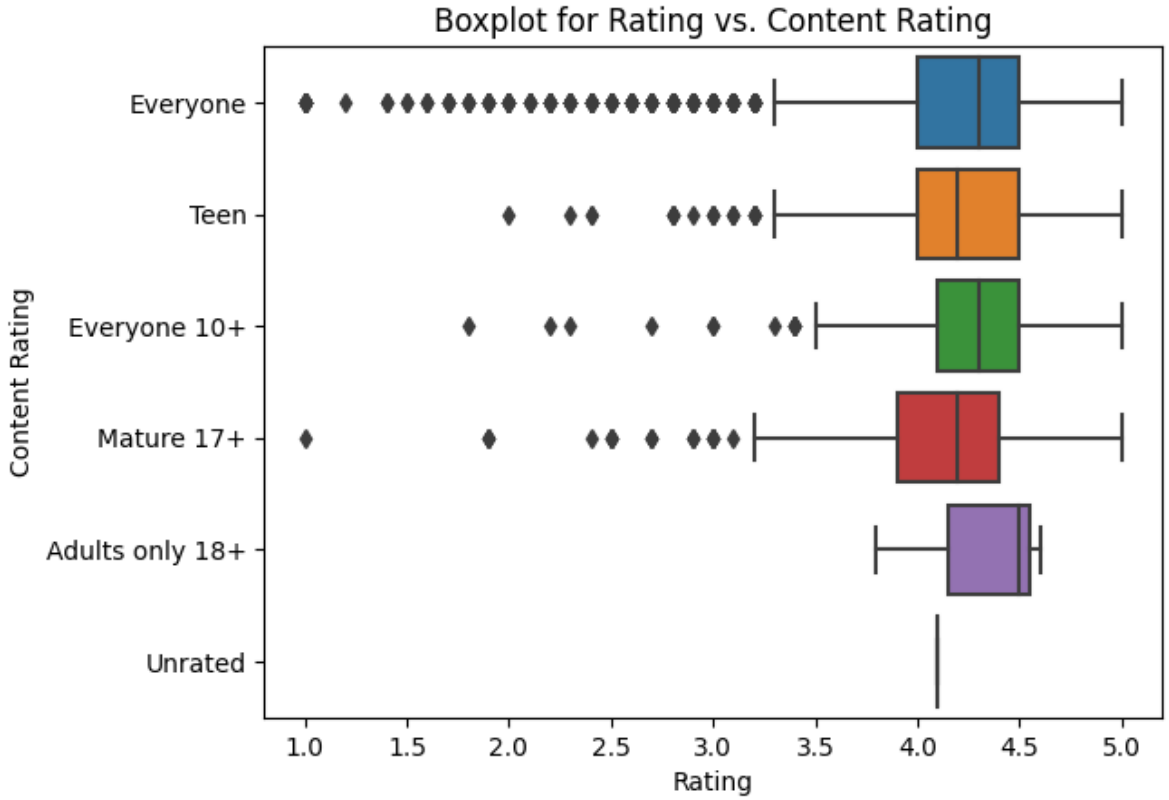


Observation : It is cristal clear that more reviews means better rating always.

Boxplot for Rating vs. Content Rating

In [66]:

```
sns.boxplot(data=df, x="Rating", y="Content Rating")
plt.title("Boxplot for Rating vs. Content Rating")
plt.xlabel("Rating")
plt.ylabel("Content Rating")
plt.show()
```



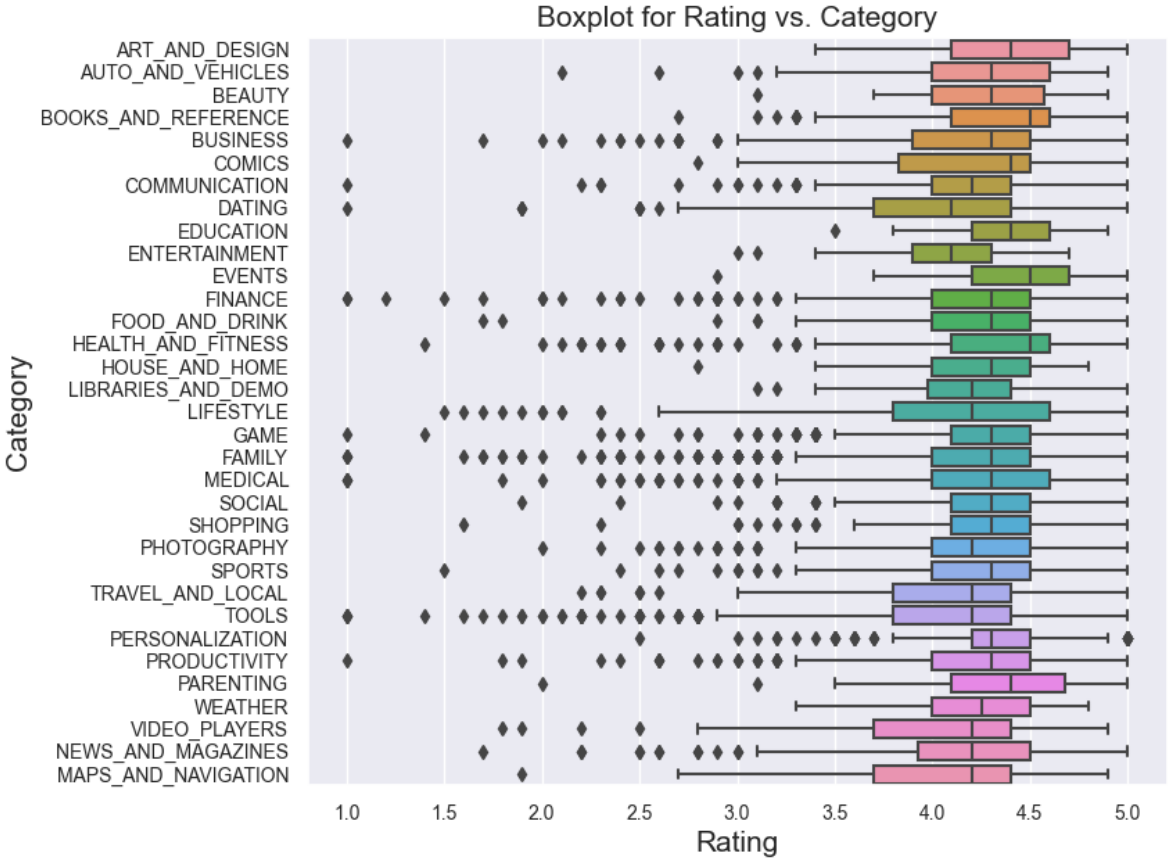
Observation : Apps which are for everyone has more bad ratings compare to other sections as it has so much outliers value, while Adults only 18+ apps have better ratings.

Boxplot for Rating vs. Category

In [67]:

```
sns.set(rc={'figure.figsize':(8,7)})

sns.boxplot(data=df, x="Rating", y="Category")
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.title("Boxplot for Rating vs. Category", fontsize=15)
plt.xlabel("Rating", fontsize=15)
plt.ylabel("Category",fontsize=15)
plt.show()
```



Observation : Events category has best ratings compare to others.

8. Data pre-processing

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 do ne as the models do not understand categorical data, and all data should be num

In [68]:

```
1 inp1 = df.copy()
```

In [69]:

```
1 inp1.skew()
```

```
C:\Users\nidhi\AppData\Local\Temp\ipykernel_26168\3545313420.py:1: FutureWarn
ing: The default value of numeric_only in DataFrame.skew is deprecated. In a
future version, it will default to False. In addition, specifying 'numeric_on
ly=None' is deprecated. Select only valid columns or specify the value of num
eric_only to silence this warning.
  inp1.skew()
```

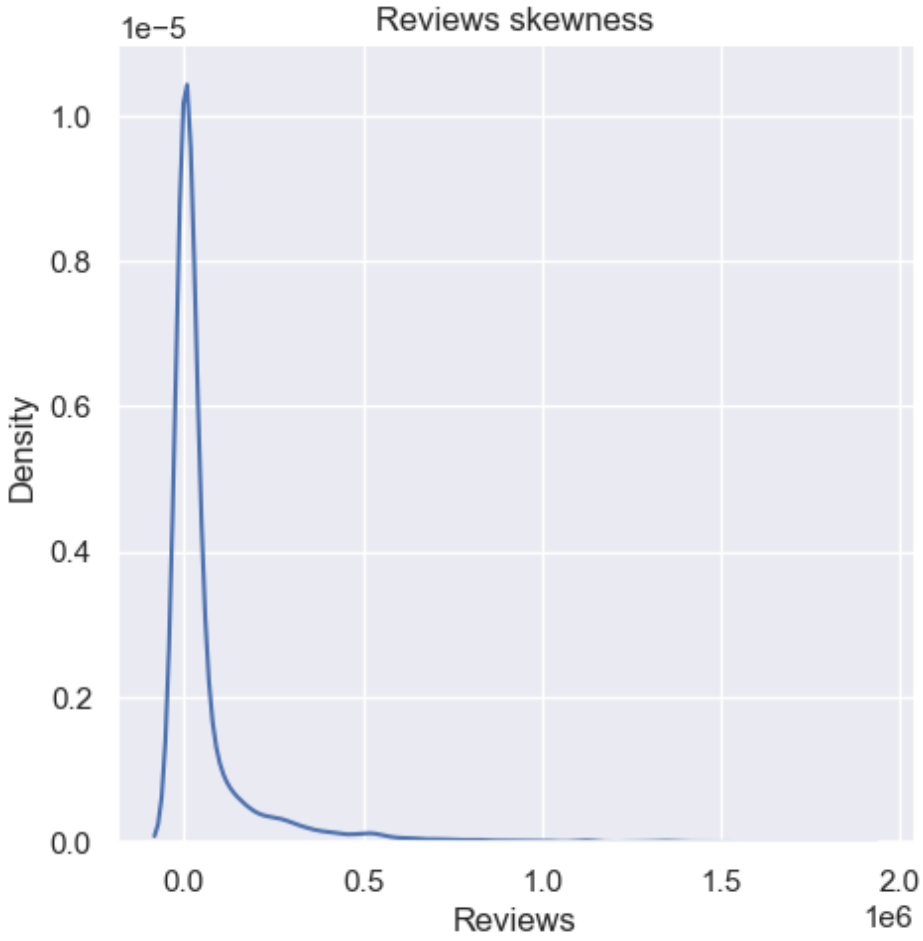
Out[69]:

```
Rating      -1.749753
Reviews      4.576494
Size         1.657766
Installs     1.543697
Price       18.074542
dtype: float64
```

In [70]:

```
sns.set(rc={'figure.figsize':(6,4)})

sns.displot(data=inp1, x="Reviews", kind="kde")
plt.title('Reviews skewness')
plt.xlabel('Reviews')
plt.show()
```



In [71]:

```
# We can see that Reviews is right skewed
```

Applying Log transformation to Reviews

In [72]:

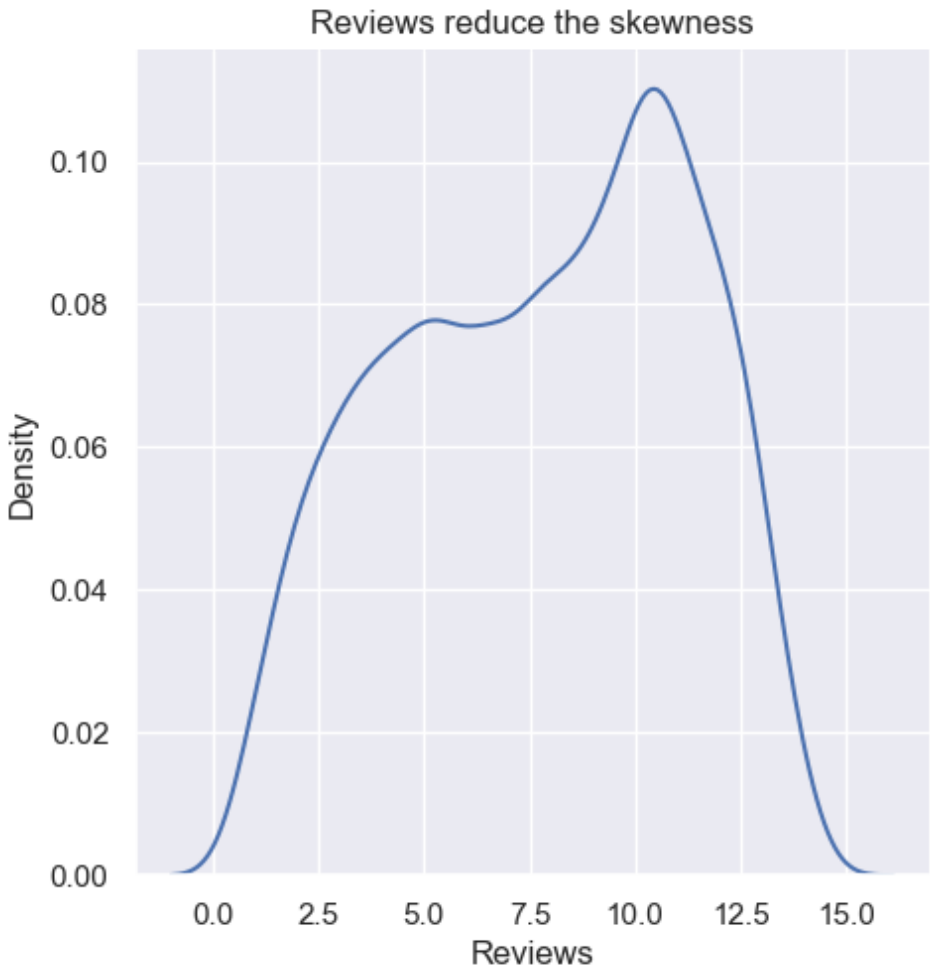
```
reviewskew = np.log1p(inp1['Reviews'])
inp1['Reviews'] = reviewskew
reviewskew.skew()
```

Out[72]:

```
-0.20039949659264134
```

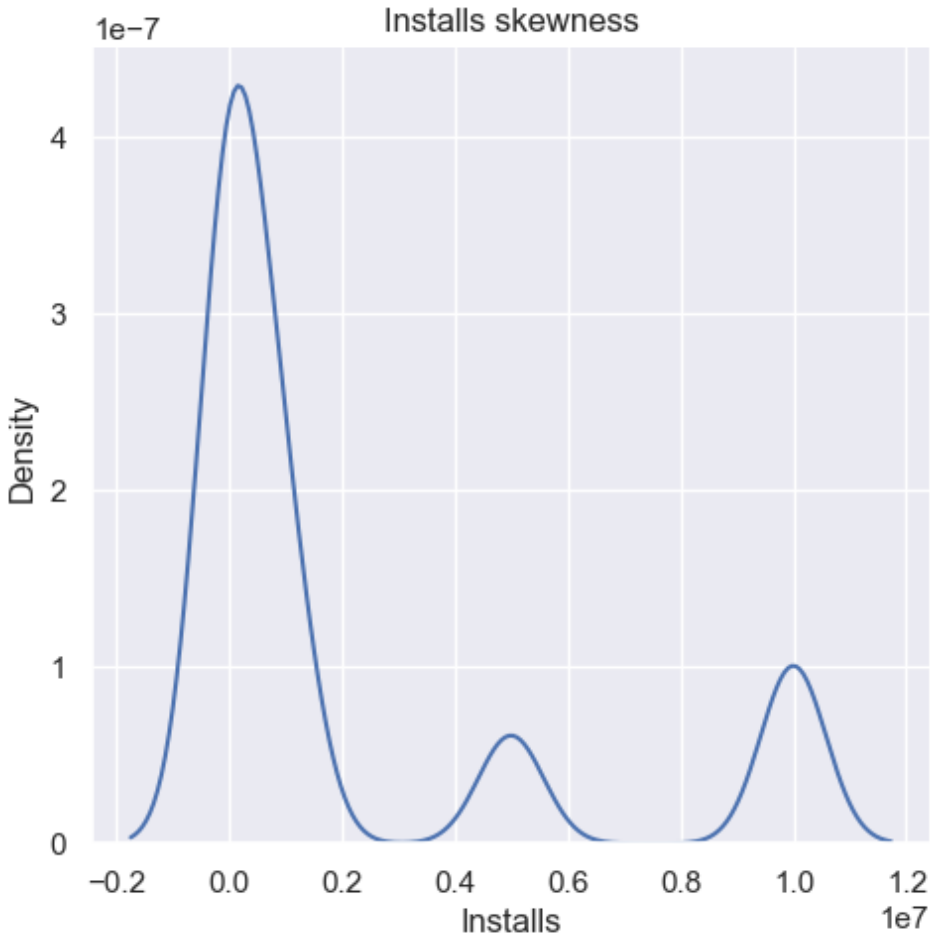
In [73]:

```
sns.displot(data=inp1, x="Reviews", kind="kde")
plt.title('Reviews reduce the skewness')
plt.xlabel('Reviews')
plt.show()
```



In [74]:

```
sns.displot(data=inp1, x="Installs", kind="kde")
plt.title('Installs skewness')
plt.xlabel('Installs')
plt.show()
```



Applying Log transformation to Installs

In [75]:

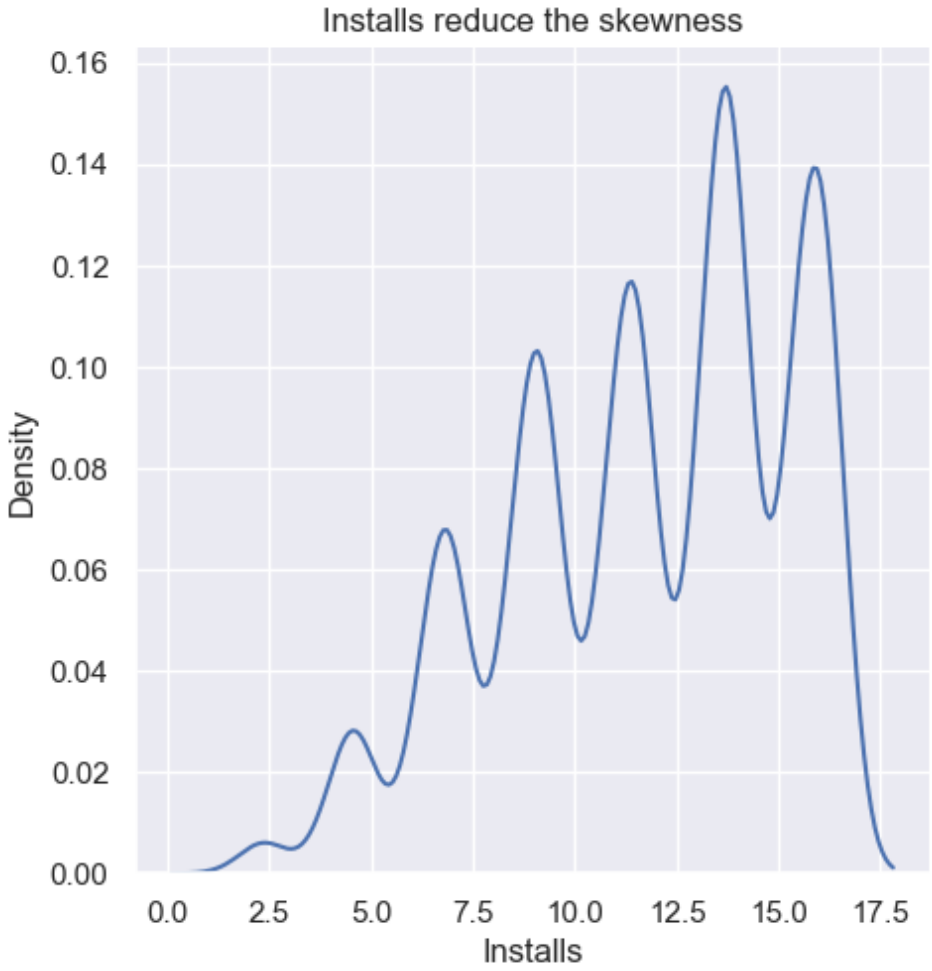
```
installsskew = np.log1p(inp1['Installs'])
inp1['Installs'] = installsskew
installsskew.skew()
```

Out[75]:

```
-0.5097286542754812
```

In [76]:

```
sns.displot(data=inp1, x="Installs", kind="kde")
plt.title('Installs reduce the skewness')
plt.xlabel('Installs')
plt.show()
```



Drop columns App, Last Updated, Current Ver, and Android Ver. These variables are not useful for our task.

In [77]:

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

Out[77]:

	Category	Rating	Reviews	Size	Installs	Price	Content Rating	Genres
0	ART_AND_DESIGN	4.1	5.075174	19000.0	9.210440	0	Everyone	Art & Design
1	ART_AND_DESIGN	3.9	6.875232	14000.0	13.122365	0	Everyone	Art & Design;Pretend Play
2	ART_AND_DESIGN	4.7	11.379520	8700.0	15.424949	0	Everyone	Art & Design
4	ART_AND_DESIGN	4.3	6.875232	2800.0	11.512935	0	Everyone	Art & Design;Creativity
5	ART_AND_DESIGN	4.4	5.123964	5600.0	10.819798	0	Everyone	Art & Design

In [78]:

```
inp2 = inp1.copy()
```

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 [79]:

```
# apply Dummy Encoding on Column "Category"
# get unique values in Column "Category"
inp2['Category'].unique()
```

Out[79]:

```
array(['ART_AND_DESIGN', 'AUTO_AND_VEHICLES', 'BEAUTY',
       'BOOKS_AND_REFERENCE', 'BUSINESS', 'COMICS', 'COMMUNICATION',
       'DATING', 'EDUCATION', 'ENTERTAINMENT', 'EVENTS', 'FINANCE',
       'FOOD_AND_DRINK', 'HEALTH_AND_FITNESS', 'HOUSE_AND_HOME',
       'LIBRARIES_AND_DEMO', 'LIFESTYLE', 'GAME', 'FAMILY', 'MEDICAL',
       'SOCIAL', 'SHOPPING', 'PHOTOGRAPHY', 'SPORTS', 'TRAVEL_AND_LOCAL',
       'TOOLS', 'PERSONALIZATION', 'PRODUCTIVITY', 'PARENTING', 'WEATHER',
       'VIDEO_PLAYERS', 'NEWS_AND_MAGAZINES', 'MAPS_AND_NAVIGATION'],
      dtype=object)
```

In [80]:

```
x = inp2[['Category']]
del inp2['Category']

dummies = pd.get_dummies(x, prefix = 'Category')
inp2 = pd.concat([inp2,dummies], axis=1)
inp2.head()
```

Out[80]:

	Rating	Reviews	Size	Installs	Price	Content Rating	Genres	Category_ART_AND_D
0	4.1	5.075174	19000.0	9.210440	0	Everyone	Art & Design	
1	3.9	6.875232	14000.0	13.122365	0	Everyone	Art & Design;Pretend Play	
2	4.7	11.379520	8700.0	15.424949	0	Everyone	Art & Design	
4	4.3	6.875232	2800.0	11.512935	0	Everyone	Art & Design;Creativity	
5	4.4	5.123964	5600.0	10.819798	0	Everyone	Art & Design	

5 rows × 40 columns

In [81]:

```
inp2.shape
```

Out[81]:

```
(8496, 40)
```

In [82]:

```
# apply Dummy EnCoding on Column "Genres"
# get unique values in Column "Genres"

inp2["Genres"].unique()
```

Out[82]:

```
array(['Art & Design', 'Art & Design;Pretend Play',
       'Art & Design;Creativity', 'Auto & Vehicles', 'Beauty',
       'Books & Reference', 'Business', 'Comics', 'Comics;Creativity',
       'Communication', 'Dating', 'Education', 'Education;Creativity',
       'Education;Education', 'Education;Music & Video',
       'Education;Action & Adventure', 'Education;Pretend Play',
       'Education;Brain Games', 'Entertainment',
       'Entertainment;Brain Games', 'Entertainment;Creativity',
       'Entertainment;Music & Video', 'Events', 'Finance', 'Food & Drink',
       'Health & Fitness', 'House & Home', 'Libraries & Demo',
       'Lifestyle', 'Lifestyle;Pretend Play', 'Card', 'Casual', 'Puzzle',
       'Action', 'Arcade', 'Word', 'Racing', 'Casual;Creativity',
       'Sports', 'Board', 'Simulation', 'Role Playing', 'Adventure',
       'Strategy', 'Simulation;Education', 'Action;Action & Adventure',
       'Trivia', 'Casual;Brain Games', 'Simulation;Action & Adventure',
       'Educational;Creativity', 'Puzzle;Brain Games',
       'Educational;Education', 'Card;Brain Games',
       'Educational;Brain Games', 'Educational;Pretend Play',
       'Casual;Action & Adventure', 'Entertainment;Education',
       'Casual;Education', 'Casual;Pretend Play', 'Music;Music & Video',
       'Racing;Action & Adventure', 'Arcade;Pretend Play',
       'Adventure;Action & Adventure', 'Role Playing;Action & Adventure',
       'Simulation;Pretend Play', 'Puzzle;Creativity',
       'Sports;Action & Adventure', 'Educational;Action & Adventure',
       'Arcade;Action & Adventure', 'Entertainment;Action & Adventure',
       'Puzzle;Action & Adventure', 'Strategy;Action & Adventure',
       'Music & Audio;Music & Video', 'Health & Fitness;Education',
       'Adventure;Education', 'Board;Brain Games',
       'Board;Action & Adventure', 'Board;Pretend Play',
       'Casual;Music & Video', 'Role Playing;Pretend Play',
       'Entertainment;Pretend Play', 'Video Players & Editors;Creativity',
       'Card;Action & Adventure', 'Medical', 'Social', 'Shopping',
       'Photography', 'Travel & Local',
       'Travel & Local;Action & Adventure', 'Tools', 'Tools;Education',
       'Personalization', 'Productivity', 'Parenting',
       'Parenting;Music & Video', 'Parenting;Brain Games',
       'Parenting;Education', 'Weather', 'Video Players & Editors',
       'Video Players & Editors;Music & Video', 'News & Magazines',
       'Maps & Navigation', 'Health & Fitness;Action & Adventure',
       'Music', 'Educational', 'Casino', 'Adventure;Brain Games',
       'Lifestyle;Education', 'Books & Reference;Education',
       'Puzzle;Education', 'Role Playing;Brain Games',
       'Strategy;Education', 'Racing;Pretend Play',
       'Communication;Creativity', 'Strategy;Creativity'], dtype=object)
```

#Since, There are too many categories under Genres. Hence, we will try to reduce some categories which have very few samples under them and put them under one new common category i.e. "Other".

In [83]:

```
pd.set_option("display.max_rows", None)
inp2["Genres"].value_counts()
```

```
Trivia                                  27
Word                                    25
Entertainment;Music & Video             23
Education;Pretend Play                  23
Racing;Action & Adventure               18
Puzzle;Brain Games                      18
Casual;Action & Adventure               17
Action;Action & Adventure               17
Educational;Pretend Play                15
Board;Brain Games                       15
Music                                   15
Casual;Brain Games                      13
Arcade;Action & Adventure               13
Simulation;Action & Adventure           11
Entertainment;Brain Games                8
Education;Creativity                     7
Art & Design;Creativity                  7
Casual;Creativity                        7
Role Playing;Action & Adventure          7
Parenting;Music & Video                  6
```

In [84]:

```
lists = []
for i in inp2['Genres'].value_counts().index:
    if inp2['Genres'].value_counts()[i] < 20:
        lists.append(i)
inp2['Genres'] = ['Other' if i in lists else i for i in inp2['Genres']]
```

In [85]:

```
1 inp2["Genres"].value_counts()
```

Out[85]:

```
Tools                          672
Entertainment                  501
Education                      468
Medical                        349
Finance                        311
Sports                         308
Lifestyle                      305
Other                          303
Business                       293
Health & Fitness               290
Personalization                290
Productivity                   285
Action                         279
Photography                    248
Communication                  243
News & Magazines               222
Social                         213
Shopping                       208
Travel & Local                 204
Dating                         195
Simulation                     184
Books & Reference              171
Arcade                         150
Casual                         142
Video Players & Editors        133
Maps & Navigation              118
Food & Drink                   109
Role Playing                   103
Puzzle                          99
Strategy                        81
House & Home                    76
Auto & Vehicles                 73
Adventure                       70
Weather                         70
Racing                          69
Libraries & Demo                64
Comics                          57
Art & Design                    54
Events                          45
Card                            44
Education;Education             43
Beauty                          42
Parenting                       40
Board                           39
Educational;Education           38
Casino                          36
Educational                     32
Casual;Pretend Play             29
Trivia                          27
Word                            25
Entertainment;Music & Video     23
Education;Pretend Play          23
Name: Genres, dtype: int64
```

In [86]:

```
x = inp2[["Genres"]]
del inp2['Genres']
dummies = pd.get_dummies(x, prefix = 'Genres')
inp2 = pd.concat([inp2,dummies], axis=1)
inp2.head()
```

Out[86]:

	Rating	Reviews	Size	Installs	Price	Content Rating	Category_ART_AND_DESIGN	Category_
0	4.1	5.075174	19000.0	9.210440	0	Everyone	1	
1	3.9	6.875232	14000.0	13.122365	0	Everyone	1	
2	4.7	11.379520	8700.0	15.424949	0	Everyone	1	
4	4.3	6.875232	2800.0	11.512935	0	Everyone	1	
5	4.4	5.123964	5600.0	10.819798	0	Everyone	1	

5 rows × 91 columns

In [87]:

```
inp2.shape
```

Out[87]:

```
(8496, 91)
```

In [88]:

```
# apply Dummy EnCoding on Column "Content Rating"
# get unique values in Column "Content Rating"
inp2["Content Rating"].unique()
```

Out[88]:

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

In [89]:

```
x = inp2[['Content Rating']]
del inp2['Content Rating']

dummies = pd.get_dummies(x, prefix = 'Content Rating')
inp2 = pd.concat([inp2,dummies], axis=1)
inp2.head()
```

Out[89]:

	Rating	Reviews	Size	Installs	Price	Category_ART_AND_DESIGN	Category_AUTO_AN
0	4.1	5.075174	19000.0	9.210440	0	1	
1	3.9	6.875232	14000.0	13.122365	0	1	
2	4.7	11.379520	8700.0	15.424949	0	1	
4	4.3	6.875232	2800.0	11.512935	0	1	
5	4.4	5.123964	5600.0	10.819798	0	1	

5 rows × 96 columns

In [90]:

```
inp2.shape
```

Out[90]:

```
(8496, 96)
```

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

In [94]:

```
1 !pip install scikit-learn
```

```
Collecting scikit-learn
  Downloading scikit_learn-1.3.0-cp311-cp311-win_amd64.whl (9.2 MB)
                                             0.0/9.2 MB ? eta -:--:--
                                             0.0/9.2 MB 1.3 MB/s eta 0:00:08
                                             0.1/9.2 MB 1.3 MB/s eta 0:00:07
                                             0.2/9.2 MB 2.0 MB/s eta 0:00:05
     -                                       0.4/9.2 MB 2.7 MB/s eta 0:00:04
     --                                      0.6/9.2 MB 3.0 MB/s eta 0:00:03
     ----                                    1.1/9.2 MB 4.5 MB/s eta 0:00:02
     ------                                  1.6/9.2 MB 5.2 MB/s eta 0:00:02
     ---------                               2.2/9.2 MB 6.5 MB/s eta 0:00:02
     -------------                           3.1/9.2 MB 7.8 MB/s eta 0:00:01
     ----------------                        3.9/9.2 MB 8.8 MB/s eta 0:00:01
     --------------------                    4.6/9.2 MB 9.5 MB/s eta 0:00:01
     ----------------------------            6.9/9.2 MB 13.3 MB/s eta 0:00:0
1
     --------------------------------        7.7/9.2 MB 13.7 MB/s eta 0:00:0
1
     ----------------------------------------  9.0/9.2 MB 14.8 MB/s eta 0:00:0
1
     ---------------------------------------- 9.2/9.2 MB 13.7 MB/s eta 0:00:0
0
Requirement already satisfied: numpy>=1.17.3 in c:\users\nidhi\appdata\local
\programs\python\python311\lib\site-packages (from scikit-learn) (1.24.2)
Requirement already satisfied: scipy>=1.5.0 in c:\users\nidhi\appdata\local\p
rograms\python\python311\lib\site-packages (from scikit-learn) (1.10.1)
Collecting joblib>=1.1.1 (from scikit-learn)
  Downloading joblib-1.3.1-py3-none-any.whl (301 kB)
                                             0.0/302.0 kB ? eta -:--:--
     ----------------------------------      276.5/302.0 kB ? eta -:--:--
     ---------------------------------------- 302.0/302.0 kB 4.6 MB/s eta 0:00:
00
Collecting threadpoolctl>=2.0.0 (from scikit-learn)
  Downloading threadpoolctl-3.1.0-py3-none-any.whl (14 kB)
Installing collected packages: threadpoolctl, joblib, scikit-learn
Successfully installed joblib-1.3.1 scikit-learn-1.3.0 threadpoolctl-3.1.0
```

In [95]:

```
1 from sklearn.model_selection import train_test_split
2 from sklearn.linear_model import LinearRegression
3 from sklearn.metrics import mean_squared_error
```

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

In [96]:

```
d1 = inp2.copy()

X = d1.drop('Rating', axis=1)
y = d1['Rating']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [97]:

```
X_train.shape[0], X_test.shape[0]
```

Out[97]:

```
(5947, 2549)
```

11 . Model building

• Use linear regression as the technique

• Report the R2 on the train set

In [98]:

```
LR_model = LinearRegression()
LR_model.fit(X_train, y_train)
```

Out[98]:

LinearRegression

LinearRegression()

In [99]:

```
R2_train = LR_model.score(X_train, y_train)
print("The R2 value of the Training Set is : {}".format(R2_train))
```

```
The R2 value of the Training Set is : 0.16437115581874862
```

In [100]:

```
R2_train = round(LR_model.score(X_train, y_train),3)
print("The R2 value of the Training Set is : {}".format(R2_train))
```

```
The R2 value of the Training Set is : 0.164
```

12. Make predictions on test set and report R2.

In [101]:

```
R2_test = round(LR_model.score(X_test, y_test), 3)
print("The R2 value of the Testing Set is : {}".format(R2_test))
```

```
The R2 value of the Testing Set is : 0.125
```

In [ ]:

```

```