Programming Basics and Data Analytics with Python

Project 1

App Rating Prediction

Import Libraries

In [1]:

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

1. Load Dataset

In [2]:

```
data=pd.read_csv("googleplaystore.csv")
```

In [3]:

```
data.head()
```

Out[3]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone
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

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

In [4]:

```
data.isna().sum()
```

Out[4]:

```
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.

In [5]:

```
new_data = data.dropna()
```

In [6]:

```
new_data.isna().sum()
```

Out[6]:

```
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
```

In [7]:

```
new_data.columns
```

Out[7]:

```
Index(['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type',
       'Price', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver',
       'Android Ver'],
      dtype='object')
```

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

4.1 Size column has sizes in Kb as well as Mb. To analyze, you'll need to convert these to numeric.

In [8]:

```
new_data["Size"].dtype
```

Out[8]:

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

In [9]:

```
new_data=new_data[-new_data['Size'].str.contains('Var')]
```

In [10]:

```
new_data.loc[:,"SizeNum"] = new_data.Size.str.rstrip("Mk+")
new_data.SizeNum=pd.to_numeric(new_data['SizeNum'])
new_data.SizeNum.dtype
```

Out[10]:

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

In [11]:

```
new_data['SizeNum']=np.where(new_data.Size.str.contains('M'),new_data.SizeNum*1000, new
_data.SizeNum)
```

In [12]:

```
new_data.Size=new_data.SizeNum
new_data.drop('SizeNum',axis=1,inplace=True)
```

In [13]:

```
new_data.Reviews = pd.to_numeric(new_data.Reviews)
```

In [14]:

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

In [15]:

```
new_data.Installs=new_data.Installs.str.replace(",","")
new_data.Installs=pd.to_numeric(new_data.Installs)
new_data.Installs.dtype
```

Out[15]:

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

In [16]:

```
new_data.Price=new_data.Price.str.replace("$","")
new_data.Price=pd.to_numeric(new_data.Price)
new_data.Price.dtype
```

Out[16]:

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

5. Sanity checks:

In [17]:

```
new_data=new_data[(new_data.Rating>=1) & (new_data.Rating<=5) ]
```

In [18]:

```
new_data.drop(new_data.index[new_data.Reviews>new_data.Installs],axis=0,inplace=True)
len(new_data.index)
```

Out[18]:

```
7717
```

In [19]:

```
index_free_and_price_gt_0 = new_data.index[((new_data.Type=='Free')&(new_data.Price>0
))]
```

In [20]:

```
if len(index_free_and_price_gt_0)>0:
    print("Dropping following indices:",index_free_and_price_gt_0)
    new_data.drop(index_free_and_price_gt_0,axis=0,inplace=True)
else:
    print("There is no Free Apps with price >0")
```

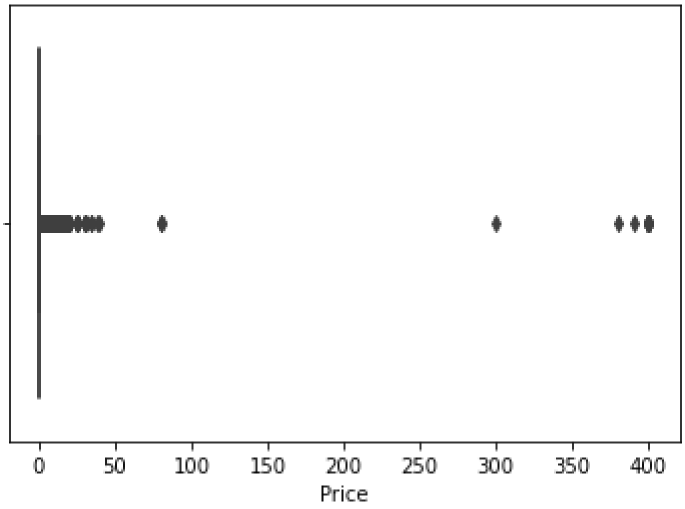
```
There is no Free Apps with price >0
```

Performing univariate analysis:

5.1. Boxplot for Price

In [21]:

```
ax = sns.boxplot(x='Price', data=new_data)
```



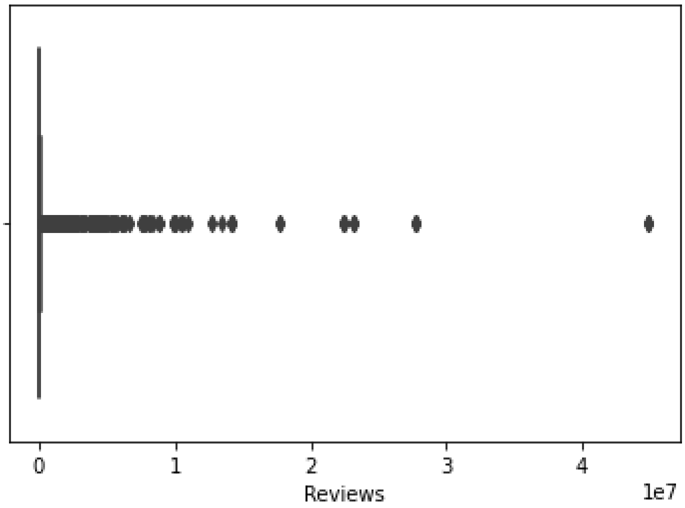
5.2. Boxplot for Reviews

In [22]:

```
sns.boxplot(x='Reviews',data=new_data)
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2065e6fa5e0>
```



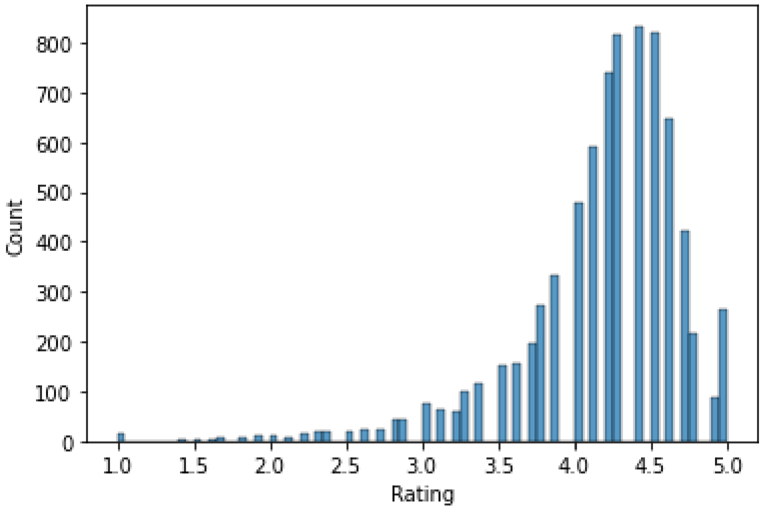
5.3. Histogram for Rating

In [23]:

```
sns.histplot(x='Rating',data=new_data)
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2065e733850>
```



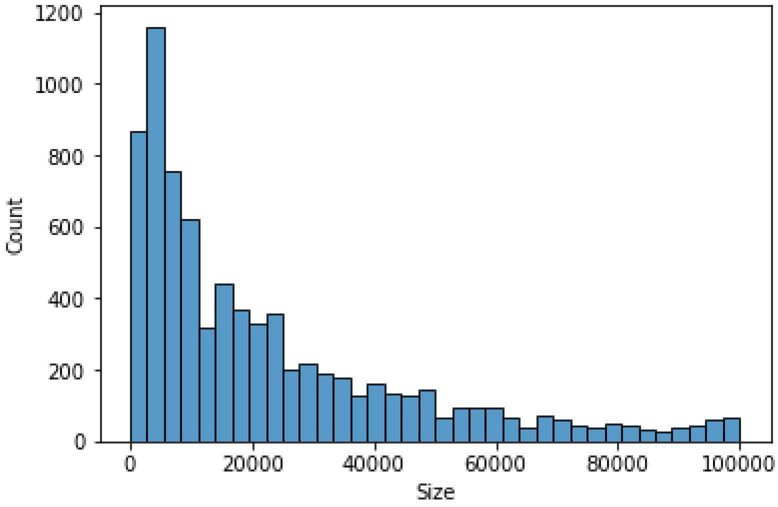
5.4. Histogram for Size

In [24]:

```
sns.histplot(x='Size',data=new_data)
```

Out[24]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2065efb4c10>
```



6. Outlier treatment:

In [25]:

```
new_data[new_data.Price>=200]
```

Out[25]:

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

In [26]:

```
new_data.drop(new_data.index[(new_data.Price>=200)], inplace=True)
len(new_data.index)
```

Out[26]:

```
7702
```

In [27]:

```
new_data.drop(new_data.index[(new_data.Reviews>=2000000)], inplace=True)
len(new_data.index)
```

Out[27]:

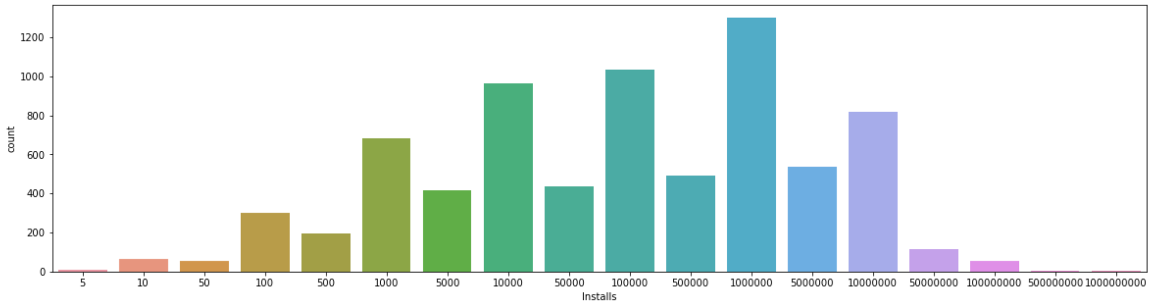
```
7483
```

In [28]:

```
plt.figure(figsize=(20,5))
sns.countplot(x='Installs',data=new_data)
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2065f07d370>
```



In [29]:

```
new_data["Installs"]
```

Out[29]:

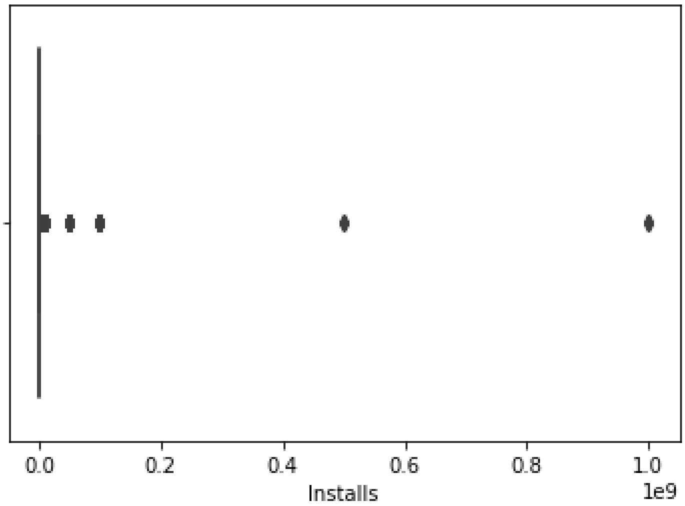
```
0           10000
1          500000
2         5000000
3        50000000
4          100000
           ...
10833        1000
10834         500
10836        5000
10837         100
10840    10000000
Name: Installs, Length: 7483, dtype: int64
```

In [30]:

```
sns.boxplot(x='Installs',data=new_data)
```

Out[30]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2065f322940>
```



In [31]:

```
install_10_perc=np.percentile(new_data.Installs, 10)
install_10_perc
```

Out[31]:

```
1000.0
```

In [32]:

```
install_25_perc=np.percentile(new_data.Installs, 25)
install_25_perc
```

Out[32]:

```
10000.0
```

In [33]:

```
install_50_perc=np.percentile(new_data.Installs, 50)
install_50_perc
```

Out[33]:

```
100000.0
```

In [34]:

```
install_70_perc=np.percentile(new_data.Installs, 70)
install_70_perc
```

Out[34]:

```
1000000.0
```

In [35]:

```
install_90_perc=np.percentile(new_data.Installs, 90)
install_90_perc
```

Out[35]:

```
10000000.0
```

In [36]:

```
install_95_perc=np.percentile(new_data.Installs, 95)
install_95_perc
```

Out[36]:

```
10000000.0
```

In [37]:

```
install_99_perc=np.percentile(new_data.Installs, 99)
install_99_perc
```

Out[37]:

```
50000000.0
```

In [38]:

```
print("As result, ",len(new_data[new_data.Installs >= install_99_perc])," will be dropp
ed")
```

```
As result,  176  will be dropped
```

In [39]:

```
new_data.drop(new_data.index[new_data.Installs >= install_99_perc],inplace=True)
len(new_data.index)
```

Out[39]:

```
7307
```

In [40]:

```
new_data["Reviews"].dtype
```

Out[40]:

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

7. Bivariate analysis

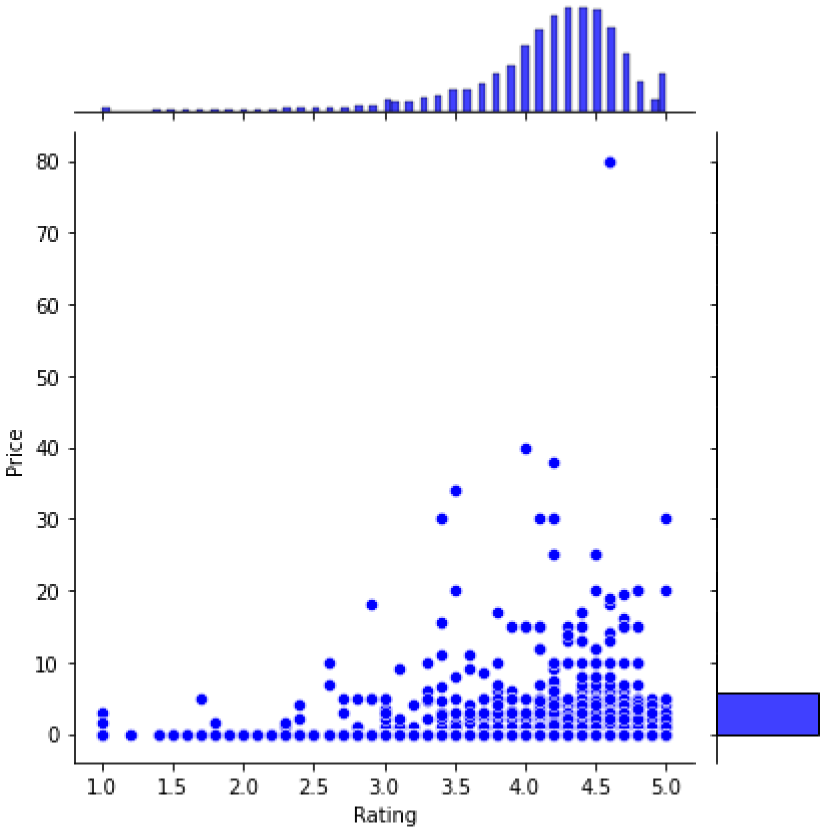
7.1 Rating Vs Price

In [41]:

```
sns.jointplot("Rating" , "Price" ,data=new_data,color="B")
```

Out[41]:

```
<seaborn.axisgrid.JointGrid at 0x2065f1390a0>
```



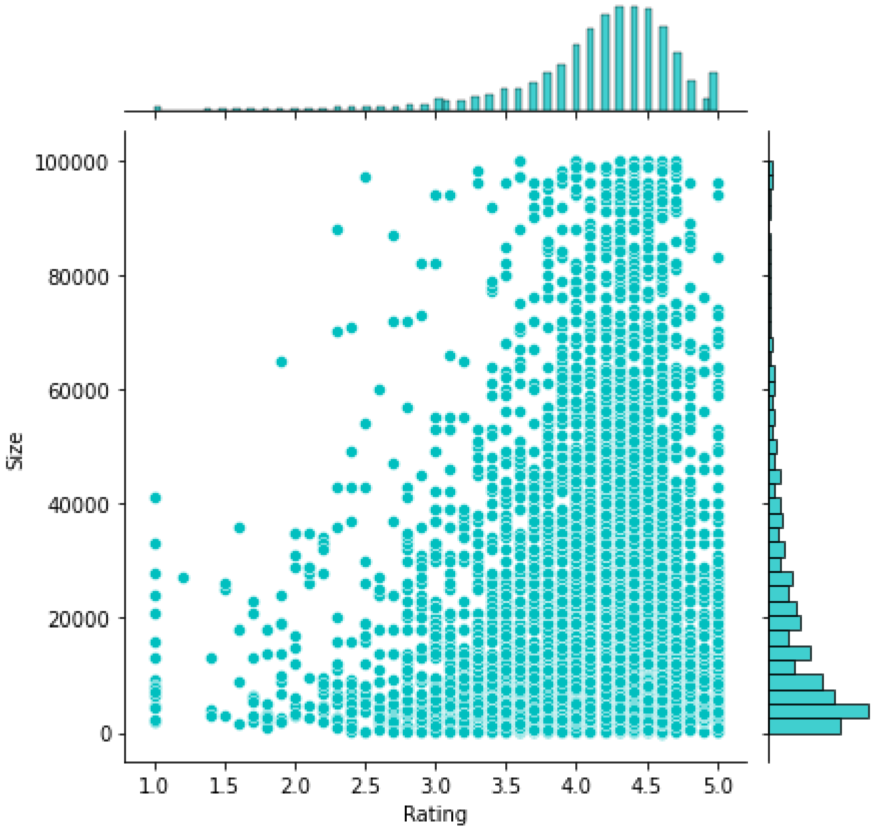
7.2 Rating Vs Size

In [42]:

```
sns.jointplot("Rating" , "Size" ,data=new_data,color="c")
```

Out[42]:

```
<seaborn.axisgrid.JointGrid at 0x2065f310a00>
```



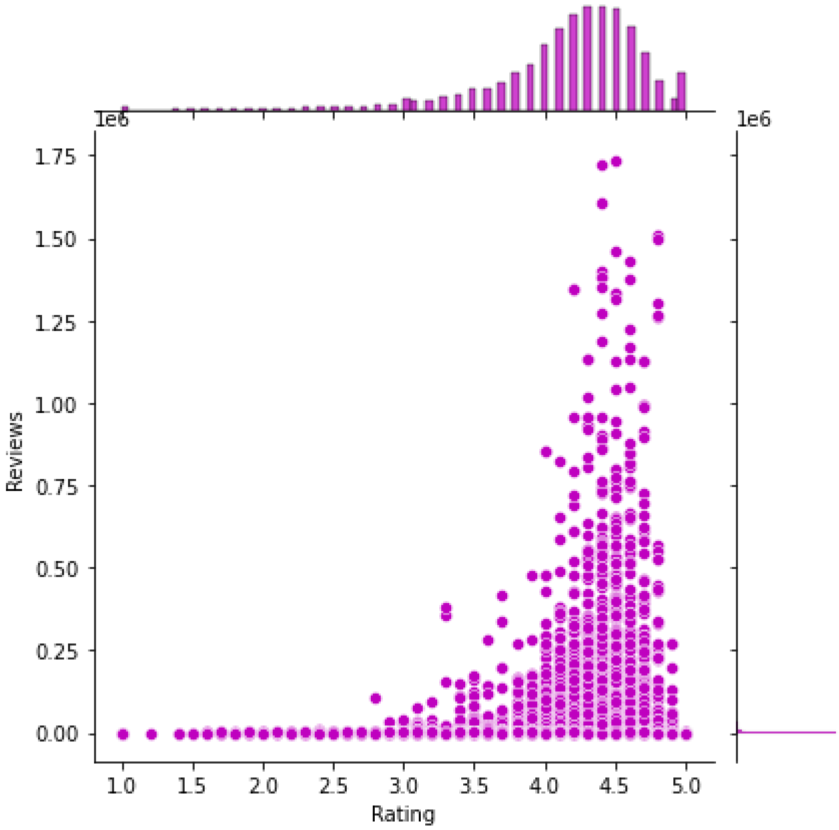
7.3 Rating Vs Reviews

In [43]:

```
sns.jointplot("Rating" , "Reviews" ,data=new_data,color="M")
```

Out[43]:

```
<seaborn.axisgrid.JointGrid at 0x2065f31d460>
```



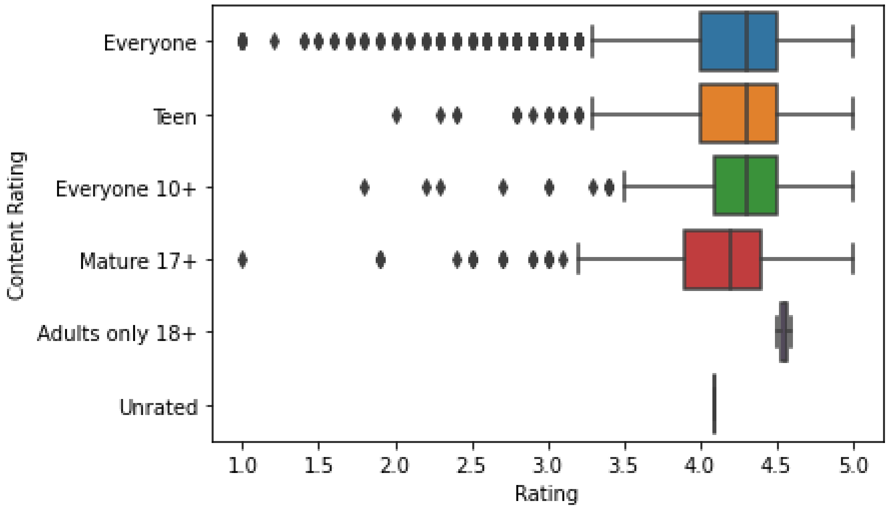
7.4 Rating Vs Content Rating

In [44]:

```
sns.boxplot(x="Rating",y="Content Rating",data=new_data)
```

Out[44]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20660e90640>
```



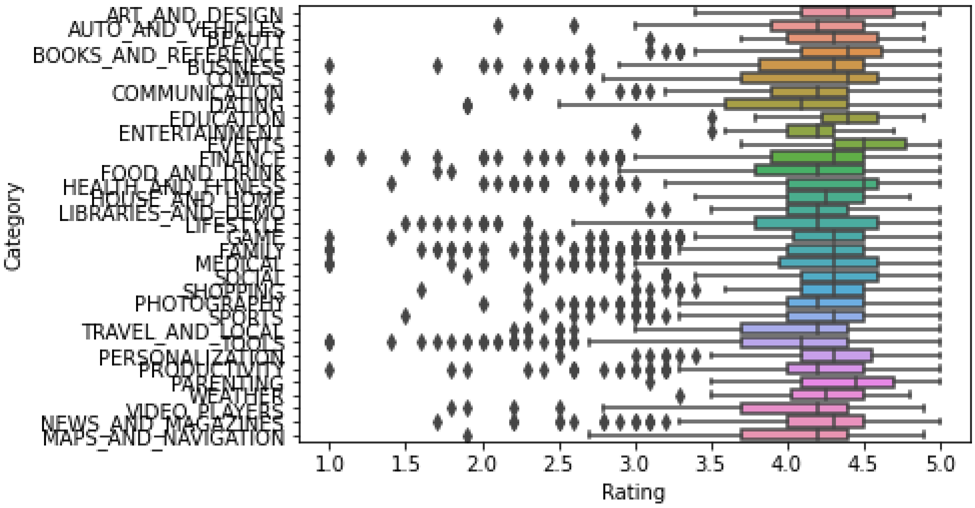
7.5 Rating Vs Category

In [45]:

```
sns.boxplot(x="Rating",y="Category",data=new_data)
```

Out[45]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2065f7d1e50>
```



8. Data preprocessing

In [46]:

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

In [47]:

```
inp1.Installs=inp1.Installs.apply(np.log1p)
inp1.drop(columns=['App','Last Updated','Current Ver','Android Ver'],inplace=True,axis=
1)
```

In [48]:

```
inp1.columns
```

Out[48]:

```
Index(['Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type', 'Pric
e',
       'Content Rating', 'Genres'],
      dtype='object')
```

In [49]:

```
inp1.head()
```

Out[49]:

	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	
0	ART_AND_DESIGN	4.1	5.075174	19000.0	9.210440	Free	0.0	Everyone	Art &
1	ART_AND_DESIGN	3.9	6.875232	14000.0	13.122365	Free	0.0	Everyone	Design;
2	ART_AND_DESIGN	4.7	11.379520	8700.0	15.424949	Free	0.0	Everyone	Art &
4	ART_AND_DESIGN	4.3	6.875232	2800.0	11.512935	Free	0.0	Everyone	Design;C
5	ART_AND_DESIGN	4.4	5.123964	5600.0	10.819798	Free	0.0	Everyone	Art &

In [50]:

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

In [51]:

```
inp2.head()
```

Out[51]:

	Rating	Reviews	Size	Installs	Price	Category_ART_AND_DESIGN	Category_AUTO
0	4.1	5.075174	19000.0	9.210440	0.0	1	
1	3.9	6.875232	14000.0	13.122365	0.0	1	
2	4.7	11.379520	8700.0	15.424949	0.0	1	
4	4.3	6.875232	2800.0	11.512935	0.0	1	
5	4.4	5.123964	5600.0	10.819798	0.0	1	

5 rows × 158 columns

In [52]:

```
set(inp2.columns)
```

Out[52]:

```
{'Category_ART_AND_DESIGN',
 'Category_AUTO_AND_VEHICLES',
 'Category_BEAUTY',
 'Category_BOOKS_AND_REFERENCE',
 'Category_BUSINESS',
 'Category_COMICS',
 'Category_COMMUNICATION',
 'Category_DATING',
 'Category_EDUCATION',
 'Category_ENTERTAINMENT',
 'Category_EVENTS',
 'Category_FAMILY',
 'Category_FINANCE',
 'Category_FOOD_AND_DRINK',
 'Category_GAME',
 'Category_HEALTH_AND_FITNESS',
 'Category_HOUSE_AND_HOME',
 'Category_LIBRARIES_AND_DEMO',
 'Category_LIFESTYLE',
 'Category_MAPS_AND_NAVIGATION',
 'Category_MEDICAL',
 'Category_NEWS_AND_MAGAZINES',
 'Category_PARENTING',
 'Category_PERSONALIZATION',
 'Category_PHOTOGRAPHY',
 'Category_PRODUCTIVITY',
 'Category_SHOPPING',
 'Category_SOCIAL',
 'Category_SPORTS',
 'Category_TOOLS',
 'Category_TRAVEL_AND_LOCAL',
 'Category_VIDEO_PLAYERS',
 'Category_WEATHER',
 'Content Rating_Adults only 18+',
 'Content Rating_Everyone',
 'Content Rating_Everyone 10+',
 'Content Rating_Mature 17+',
 'Content Rating_Teen',
 'Content Rating_Unrated',
 'Genres_Action',
 'Genres_Action;Action & Adventure',
 'Genres_Adventure',
 'Genres_Adventure;Action & Adventure',
 'Genres_Adventure;Brain Games',
 'Genres_Adventure;Education',
 'Genres_Arcade',
 'Genres_Arcade;Action & Adventure',
 'Genres_Arcade;Pretend Play',
 'Genres_Art & Design',
 'Genres_Art & Design;Creativity',
 'Genres_Art & Design;Pretend Play',
 'Genres_Auto & Vehicles',
 'Genres_Beauty',
 'Genres_Board',
 'Genres_Board;Action & Adventure',
 'Genres_Board;Brain Games',
 'Genres_Board;Pretend Play',
 'Genres_Books & Reference',
 'Genres_Books & Reference;Education',
 'Genres_Business',
 'Genres_Card',
 'Genres_Card;Action & Adventure',
 'Genres_Card;Brain Games',
 'Genres_Casino',
 'Genres_Casual',
 'Genres_Casual;Action & Adventure',
 'Genres_Casual;Brain Games',
 'Genres_Casual;Creativity',
 'Genres_Casual;Education',
 'Genres_Casual;Music & Video',
 'Genres_Casual;Pretend Play',
 'Genres_Comics',
 'Genres_Comics;Creativity',
 'Genres_Communication',
 'Genres_Dating',
 'Genres_Education',
 'Genres_Education;Action & Adventure',
 'Genres_Education;Brain Games',
 'Genres_Education;Creativity',
 'Genres_Education;Education',
 'Genres_Education;Music & Video',
 'Genres_Education;Pretend Play',
 'Genres_Educational',
 'Genres_Educational;Action & Adventure',
 'Genres_Educational;Brain Games',
 'Genres_Educational;Creativity',
 'Genres_Educational;Education',
 'Genres_Educational;Pretend Play',
 'Genres_Entertainment',
 'Genres_Entertainment;Action & Adventure',
 'Genres_Entertainment;Brain Games',
 'Genres_Entertainment;Creativity',
 'Genres_Entertainment;Education',
 'Genres_Entertainment;Music & Video',
 'Genres_Entertainment;Pretend Play',
 'Genres_Events',
 'Genres_Finance',
 'Genres_Food & Drink',
 'Genres_Health & Fitness',
 'Genres_Health & Fitness;Action & Adventure',
 'Genres_Health & Fitness;Education',
 'Genres_House & Home',
 'Genres_Libraries & Demo',
 'Genres_Lifestyle',
 'Genres_Lifestyle;Pretend Play',
 'Genres_Maps & Navigation',
 'Genres_Medical',
 'Genres_Music',
 'Genres_Music & Audio;Music & Video',
 'Genres_Music;Music & Video',
 'Genres_News & Magazines',
 'Genres_Parenting',
 'Genres_Parenting;Brain Games',
 'Genres_Parenting;Education',
 'Genres_Parenting;Music & Video',
 'Genres_Personalization',
 'Genres_Photography',
 'Genres_Productivity',
 'Genres_Puzzle',
 'Genres_Puzzle;Action & Adventure',
 'Genres_Puzzle;Brain Games',
 'Genres_Puzzle;Creativity',
 'Genres_Puzzle;Education',
 'Genres_Racing',
 'Genres_Racing;Action & Adventure',
 'Genres_Racing;Pretend Play',
 'Genres_Role Playing',
 'Genres_Role Playing;Action & Adventure',
 'Genres_Role Playing;Brain Games',
 'Genres_Role Playing;Pretend Play',
 'Genres_Shopping',
 'Genres_Simulation',
 'Genres_Simulation;Action & Adventure',
 'Genres_Simulation;Education',
 'Genres_Simulation;Pretend Play',
 'Genres_Social',
 'Genres_Sports',
 'Genres_Sports;Action & Adventure',
 'Genres_Strategy',
 'Genres_Strategy;Action & Adventure',
 'Genres_Strategy;Creativity',
 'Genres_Strategy;Education',
 'Genres_Tools',
 'Genres_Travel & Local',
 'Genres_Travel & Local;Action & Adventure',
 'Genres_Trivia',
 'Genres_Video Players & Editors',
 'Genres_Video Players & Editors;Creativity',
 'Genres_Video Players & Editors;Music & Video',
 'Genres_Weather',
 'Genres_Word',
 'Installs',
 'Price',
 'Rating',
 'Reviews',
 'Size',
 'Type_Free',
 'Type_Paid'}
```

In [53]:

```
data = inp2.drop(columns='Rating')
data.shape
target = pd.DataFrame(inp2.Rating)
target.shape
```

Out[53]:

```
(7307, 1)
```

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

In [54]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(data, target, test_size=0.3, random_state=3)
print("x_train shape is ", x_train.shape)
print("y_train shape is ", y_train.shape)
print("x_test shape is ", x_test.shape)
print("y_test shape is ", y_test.shape)
```

```
x_train shape is  (5114, 157)
y_train shape is  (5114, 1)
x_test shape is  (2193, 157)
y_test shape is  (2193, 1)
```

11 . Model building

In [55]:

```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train, y_train)
```

Out[55]:

```
LinearRegression()
```

In [56]:

```
from sklearn.metrics import r2_score
train_pred=model.predict(x_train)
print("R2 value of the model(by train) is ", r2_score(y_train, train_pred))
```

```
R2 value of the model(by train) is  0.15264772134593874
```

12. Make predictions on test set and report R2.

In [57]:

```
test_pred=model.predict(x_test)
print("R2 value of the model(by test) is ", r2_score(y_test, test_pred))
```

```
R2 value of the model(by test) is  0.14262263030973144
```

```
                        ----- The Project End -----


   Thank you.
```