In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
```

In [2]:
```
df1 = pd.read_csv("app_info.csv")
```

In [3]:
```
df2 = pd.read_csv("app_reviews.csv")
```

In [4]:
```
df1.describe()
```

Out[4]:

	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

In [5]:
```
df2.describe()
```

Out[5]:

	Sentiment_Polarity	Sentiment_Subjectivity
count	37432.000000	37432.000000
mean	0.182146	0.492704
std	0.351301	0.259949
min	-1.000000	0.000000
25%	0.000000	0.357143
50%	0.150000	0.514286
75%	0.400000	0.650000
max	1.000000	1.000000

In [6]:
```
df1.head()
```

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	[
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

In [108]:
```
df2.head()
```

Out[108]:

	App	Translated_Review	Sentiment	Sentiment_Polarity	Sentiment_Subjectivity
0	10 Best Foods for You	I like eat delicious food. That's I'm cooking ...	Positive	1.00	0.533333
1	10 Best Foods for You	This help eating healthy exercise regular basis	Positive	0.25	0.288462
2	10 Best Foods for You	NaN	NaN	NaN	NaN
3	10 Best Foods for You	Works great especially going grocery store	Positive	0.40	0.875000
4	10 Best Foods for You	Best idea us	Positive	1.00	0.300000

In [8]:
```
df1.isnull().sum()
```

Out[8]:
```
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 [9]:
```
alldata = df1.merge(df2, on = 'App')
```

In [10]:
```
alldata.head()
```

Out[10]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	G
0	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Design;F
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Design;F
2	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Design;F
3	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Design;F
4	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Design;F

In [11]:
```
alldata.Size.value_counts().head()
```

Out[11]:
```
Varies with device    47230
11M                    2500
14M                    2436
97M                    2360
52M                    2240
Name: Size, dtype: int64
```

In [12]:
```
alldata['Size'].unique()
```

Out[12]:
```
array(['14M', '33M', '37M', '39M', '12M', '25M', '6.1M', '11M',
       'Varies with device', '15M', '24M', '17M', '2.4M', '27M', '16M',
       '2.9M', '38M', '21M', '52M', '8.2M', '4.2M', '2.8M', '5.0M',
       '5.9M', '19M', '73M', '6.8M', '4.0M', '10M', '2.1M', '42M', '30M',
       '29M', '3.9M', '18M', '3.2M', '20M', '26M', '48M', '22M', '8.5M',
       '28M', '46M', '23M', '7.1M', '40M', '7.8M', '6.6M', '5.1M', '79k',
       '32M', '2.2M', '4.7M', '2.7M', '13M', '118k', '44M', '7.3M',
       '695k', '6.2M', '56M', '3.1M', '31M', '8.0M', '7.9M', '1.4M',
       '7.2M', '3.8M', '41M', '9.0M', '4.9M', '8.1M', '3.7M', '4.5M',
       '2.6M', '6.9M', '7.4M', '1.9M', '1.8M', '2.3M', '5.2M', '1.2M',
       '59M', '5.6M', '72M', '9.6M', '43M', '3.6M', '9.5M', '8.9M', '65M',
       '79M', '8.4M', '58M', '50M', '45M', '53M', '68M', '66M', '35M',
       '76M', '9.4M', '4.3M', '67M', '60M', '5.5M', '3.3M', '7.5M',
       '9.1M', '8.3M', '9.2M', '7.0M', '77M', '5.7M', '5.3M', '232k',
       '99M', '624k', '95M', '8.5k', '292k', '55M', '80M', '4.4M', '36M',
       '3.0M', '1.7M', '4.6M', '7.7M', '74M', '97M', '98M', '78M', '85M',
       '63M', '70M', '71M', '49M', '82M', '57M', '86M', '81M', '96M',
       '93M', '83M', '54M', '61M', '75M', '51M', '91M', '1.6M', '47M',
       '6.4M', '34M', '2.0M', '1.5M', '62M', '4.1M', '9.3M', '9.7M',
       '8.7M', '9.8M', '6.0M', '6.5M', '88M', '7.6M', '10.0M', '5.8M',
       '1.3M', '9.9M', '3.5M', '5.4M', '853k', '720k', '3.4M', '6.3M',
       '92M', '94M'], dtype=object)
```

In [13]:
```
alldata.Size=alldata.Size.str.replace('k','e+3')
alldata.Size=alldata.Size.str.replace('M','e+6')
```

In [14]:
```
alldata['Size'].unique()
```

Out[14]:
```
array(['14e+6', '33e+6', '37e+6', '39e+6', '12e+6', '25e+6', '6.1e+6',
       '11e+6', 'Varies with device', '15e+6', '24e+6', '17e+6', '2.4e+6',
       '27e+6', '16e+6', '2.9e+6', '38e+6', '21e+6', '52e+6', '8.2e+6',
       '4.2e+6', '2.8e+6', '5.0e+6', '5.9e+6', '19e+6', '73e+6', '6.8e+6',
       '4.0e+6', '10e+6', '2.1e+6', '42e+6', '30e+6', '29e+6', '3.9e+6',
       '18e+6', '3.2e+6', '20e+6', '26e+6', '48e+6', '22e+6', '8.5e+6',
       '28e+6', '46e+6', '23e+6', '7.1e+6', '40e+6', '7.8e+6', '6.6e+6',
       '5.1e+6', '79e+3', '32e+6', '2.2e+6', '4.7e+6', '2.7e+6', '13e+6',
       '118e+3', '44e+6', '7.3e+6', '695e+3', '6.2e+6', '56e+6', '3.1e+6',
       '31e+6', '8.0e+6', '7.9e+6', '1.4e+6', '7.2e+6', '3.8e+6', '41e+6',
       '9.0e+6', '4.9e+6', '8.1e+6', '3.7e+6', '4.5e+6', '2.6e+6',
       '6.9e+6', '7.4e+6', '1.9e+6', '1.8e+6', '2.3e+6', '5.2e+6',
       '1.2e+6', '59e+6', '5.6e+6', '72e+6', '9.6e+6', '43e+6', '3.6e+6',
       '9.5e+6', '8.9e+6', '65e+6', '79e+6', '8.4e+6', '58e+6', '50e+6',
       '45e+6', '53e+6', '68e+6', '66e+6', '35e+6', '76e+6', '9.4e+6',
       '4.3e+6', '67e+6', '60e+6', '5.5e+6', '3.3e+6', '7.5e+6', '9.1e+6',
       '8.3e+6', '9.2e+6', '7.0e+6', '77e+6', '5.7e+6', '5.3e+6',
       '232e+3', '99e+6', '624e+3', '95e+6', '8.5e+3', '292e+3', '55e+6',
       '80e+6', '4.4e+6', '36e+6', '3.0e+6', '1.7e+6', '4.6e+6', '7.7e+6',
       '74e+6', '97e+6', '98e+6', '78e+6', '85e+6', '63e+6', '70e+6',
       '71e+6', '49e+6', '82e+6', '57e+6', '86e+6', '81e+6', '96e+6',
       '93e+6', '83e+6', '54e+6', '61e+6', '75e+6', '51e+6', '91e+6',
       '1.6e+6', '47e+6', '6.4e+6', '34e+6', '2.0e+6', '1.5e+6', '62e+6',
       '4.1e+6', '9.3e+6', '9.7e+6', '8.7e+6', '9.8e+6', '6.0e+6',
       '6.5e+6', '88e+6', '7.6e+6', '10.0e+6', '5.8e+6', '1.3e+6',
       '9.9e+6', '3.5e+6', '5.4e+6', '853e+3', '720e+3', '3.4e+6',
       '6.3e+6', '92e+6', '94e+6'], dtype=object)
```

In [15]:
```
alldata['Size'].dtype
```

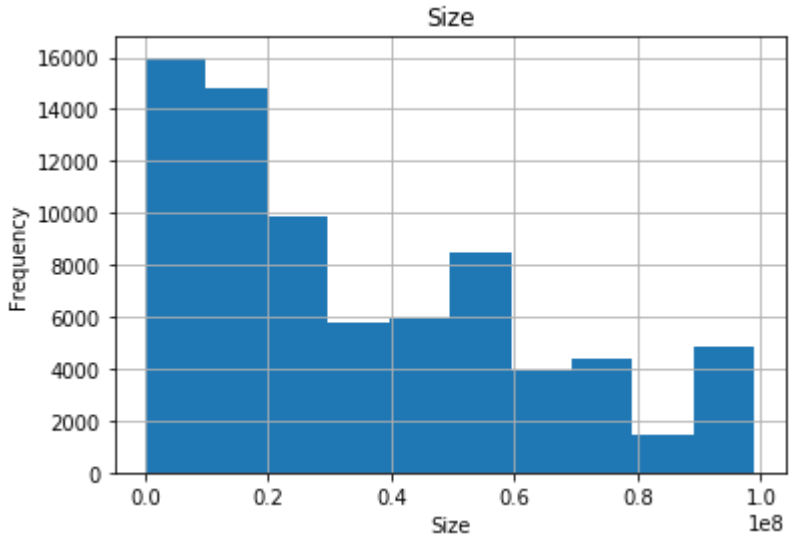
Out[15]:
```
dtype('O')
```

In [16]:
```
alldata.Size=alldata.Size.replace('Varies with device',np.nan)
```

In [17]:
```
alldata.Size=pd.to_numeric(alldata.Size)
```

In [18]:
```
alldata.hist(column='Size')
plt.xlabel('Size')
plt.ylabel('Frequency')
```

Out[18]: Text(0, 0.5, 'Frequency')



Check if Reviews and ratings are correlated

In [127]:
```
grouped=alldata.groupby(['Reviews'], sort=False)['Rating'].mean()
grouped.sort_values(ascending=False).head(10)
```

Out[127]:
```
Reviews
28945      4.9
7774       4.9
776        4.9
6090       4.9
39480      4.8
2588730    4.8
110        4.8
1264084    4.8
1259075    4.8
12304      4.8
Name: Rating, dtype: float64
```

observation: rating is not affected by reviews.

Correlation Matrix

In [20]:
```
corr = alldata.corr()
sns.heatmap(corr,  xticklabels=corr.columns,  yticklabels=corr.columns)
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x1fac0efd748>

In [21]:
```
alldata.Installs.value_counts()
```

Out[21]:
```
10,000,000+       28076
100,000,000+      24740
1,000,000+        18040
5,000,000+        16880
50,000,000+        9420
100,000+           8064
500,000+           6488
1,000,000,000+     3960
500,000,000+       3334
10,000+            1709
50,000+            1471
1,000+              400
5,000+               80
Name: Installs, dtype: int64
```

In [22]:
```
alldata.Installs=alldata.Installs.apply(lambda x: x.strip('+'))
alldata.Installs=alldata.Installs.apply(lambda x: x.replace(',',''))
alldata.Installs=alldata.Installs.replace('Free',np.nan)
alldata.Installs.value_counts()
```

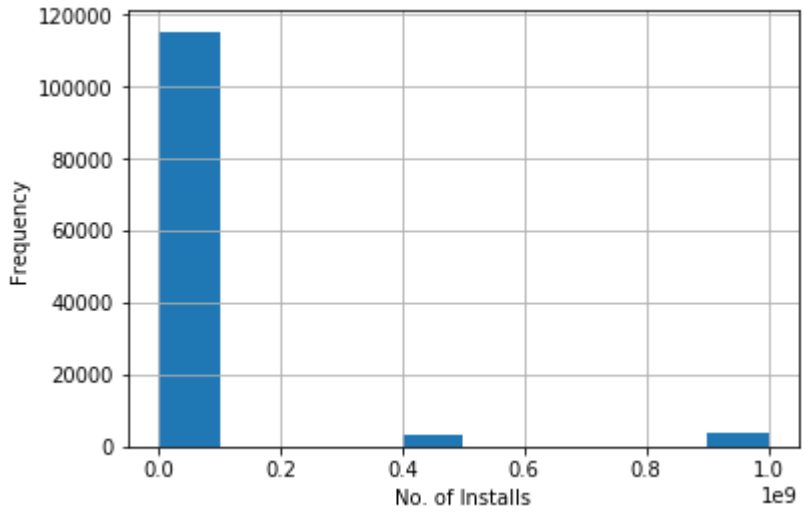
Out[22]:
```
10000000      28076
100000000     24740
1000000       18040
5000000       16880
50000000       9420
100000         8064
500000         6488
1000000000     3960
500000000      3334
10000          1709
50000          1471
1000            400
5000             80
Name: Installs, dtype: int64
```

In [23]:
```
alldata.Installs.str.isnumeric().sum()
```

Out[23]:
```
122662
```

In [24]:
```
alldata.Installs=pd.to_numeric(alldata.Installs)
alldata.Installs.hist();
plt.xlabel('No. of Installs')
plt.ylabel('Frequency')
```

Out[24]:
```
Text(0, 0.5, 'Frequency')
```



In [25]:
```
alldata.Reviews.str.isnumeric().sum()
```

Out[25]:
```
122662
```

In [26]:
```
print("Range: ", alldata.Rating.min(),"-",alldata.Rating.max())
```

```
Range:  2.5 - 4.9
```

In [28]:
```
alldata.Rating.dtype
```

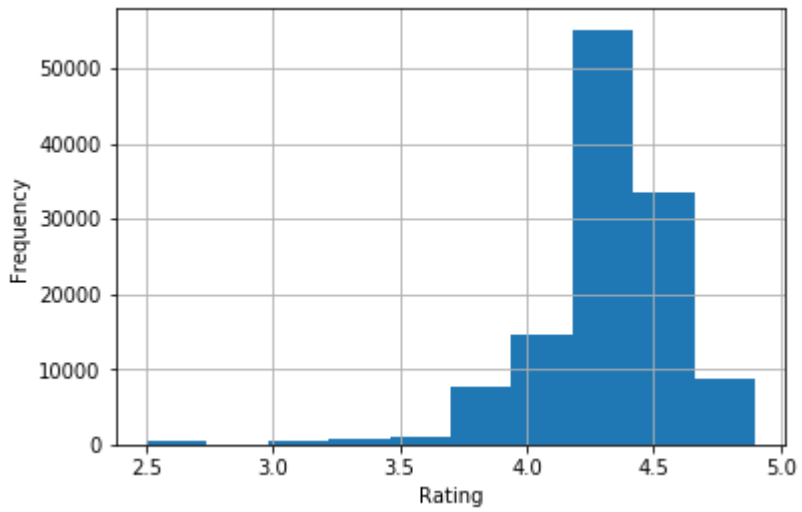
Out[28]:
```
dtype('float64')
```

In [29]:
```
alldata.Rating.isna().sum()
```

Out[29]:
```
40
```

In [30]:
```
alldata.Rating.hist();
plt.xlabel('Rating')
plt.ylabel('Frequency')
```

Out[30]:
```
Text(0, 0.5, 'Frequency')
```



In [31]:
```
alldata.Type.value_counts()
```

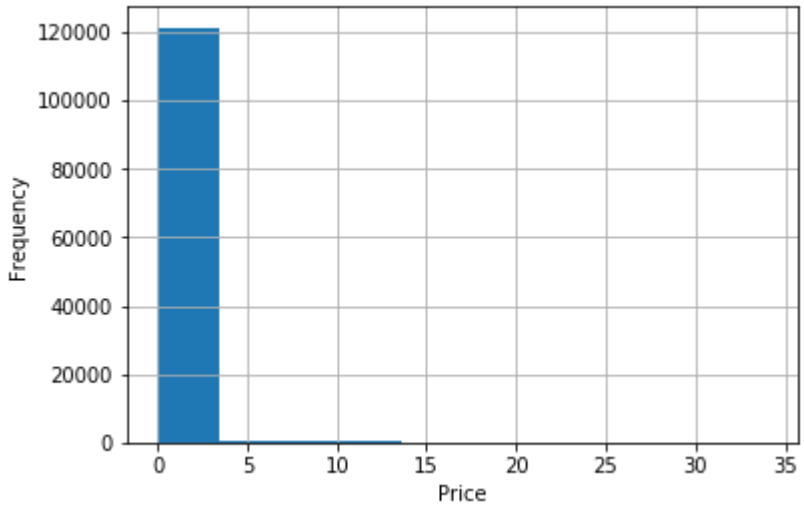
Out[31]:
```
Free    121108
Paid      1554
Name: Type, dtype: int64
```

In [32]:
```
alldata.Price.unique()
```

Out[32]:
```
array(['0', '$6.99', '$1.99', '$4.99', '$3.99', '$2.99', '$11.99',
       '$9.99', '$7.99', '$33.99', '$3.95', '$29.99', '$2.49', '$0.99'],
      dtype=object)
```

In [33]:
```
alldata.Price=alldata.Price.apply(lambda x: x.strip('$'))
alldata.Price=pd.to_numeric(alldata.Price)
alldata.Price.hist();
plt.xlabel('Price')
plt.ylabel('Frequency')
```

Out[33]: Text(0, 0.5, 'Frequency')



In a single SQL query, find the ratings of the top 10 applications which have the most installs.

In [34]:
```
alldata.groupby('Installs',as_index=False)['Rating'].mean().sort_values(by='Installs',ascending=False).head(10)
```

Out[34]:

	Installs	Rating
12	1000000000	4.213131
11	500000000	4.362807
10	100000000	4.431366
9	50000000	4.517622
8	10000000	4.319404
7	5000000	4.322749
6	1000000	4.261086
5	500000	4.123644
4	100000	4.128968
3	50000	4.241400

Write a SQL statement to return the percentage of positive, negative, and neutral sentiment reviews for each application. Show your results in a visualization of your choice (ie. matplotlib, seaborn, D3.js, etc).

In [73]:
```
alldata.groupby('Sentiment',as_index=False)['App'].count()
```

Out[73]:

	Sentiment	App
0	Negative	18134
1	Neutral	8286
2	Positive	46195

In [109]:
```
alldata.isnull().sum()
```

Out[109]:
```
App                        0
Category                   0
Rating                    40
Reviews                    0
Size                   47230
Installs                   0
Type                       0
Price                      0
Content Rating             0
Genres                     0
Last Updated               0
Current Ver                0
Android Ver                0
Translated_Review      50057
Sentiment              50047
Sentiment_Polarity     50047
Sentiment_Subjectivity 50047
dtype: int64
```

In [111]:
```
alldata.shape
```

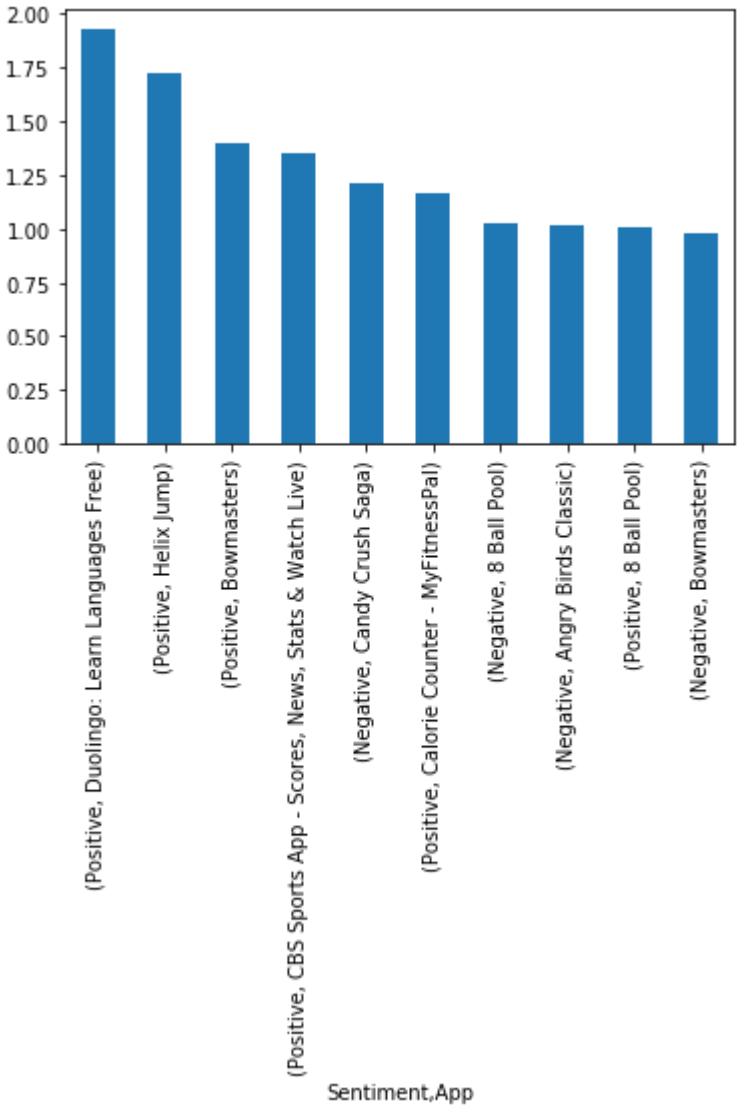
Out[111]:
```
(122662, 17)
```

In [116]:
```
df=alldata.groupby(['Sentiment','App'])['App'].count()*100/(122662-50047)
df=df.sort_values(ascending=False)
df=df.head(10)
df
```

Out[116]:
```
Sentiment  App
Positive   Duolingo: Learn Languages Free                    1.927976
           Helix Jump                                        1.726916
           Bowmasters                                        1.396406
           CBS Sports App - Scores, News, Stats & Watch Live 1.355092
Negative   Candy Crush Saga                                  1.214625
Positive   Calorie Counter - MyFitnessPal                    1.163671
Negative   8 Ball Pool                                       1.021827
           Angry Birds Classic                               1.012188
Positive   8 Ball Pool                                       1.002548
Negative   Bowmasters                                        0.983268
Name: App, dtype: float64
```

In [119]:
```
df.plot.bar(y='Sentiment')
```

Out[119]: <matplotlib.axes._subplots.AxesSubplot at 0x1fac6063518>



If a company is looking to develop the next top trending application on the Google Play Store, what kind of app should they focus on building

In [122]:
```
alldata.groupby('Category',as_index=False)['Rating'].mean().sort_values(by='Rating',ascending=False)
```

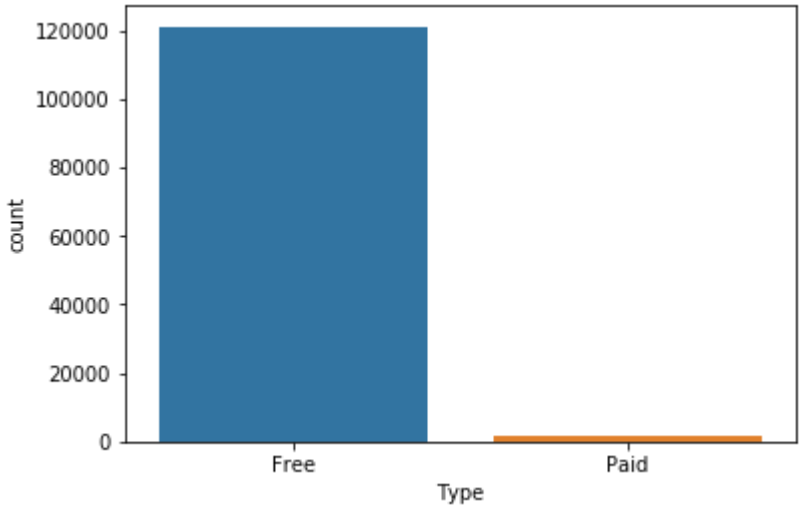
Out[122]:

	Category	Rating
1	AUTO_AND_VEHICLES	4.542123
14	GAME	4.456276
15	HEALTH_AND_FITNESS	4.427217
8	EDUCATION	4.411494
22	PARENTING	4.390909
25	PRODUCTIVITY	4.387197
19	MAPS_AND_NAVIGATION	4.381818
23	PERSONALIZATION	4.354545
11	FAMILY	4.353252
13	FOOD_AND_DRINK	4.347451
24	PHOTOGRAPHY	4.336749
3	BOOKS_AND_REFERENCE	4.336735
12	FINANCE	4.316234
26	SHOPPING	4.300463
29	TOOLS	4.300309
32	WEATHER	4.300000
0	ART_AND_DESIGN	4.299635
4	BUSINESS	4.282278
28	SPORTS	4.280317
16	HOUSE_AND_HOME	4.263717
10	EVENTS	4.257143
30	TRAVEL_AND_LOCAL	4.247489
2	BEAUTY	4.236041
6	COMMUNICATION	4.230493
31	VIDEO_PLAYERS	4.216000
27	SOCIAL	4.208889
21	NEWS_AND_MAGAZINES	4.208750
18	LIFESTYLE	4.177318
20	MEDICAL	4.168456
17	LIBRARIES_AND_DEMO	4.161538
5	COMICS	4.100000
7	DATING	4.095344
9	ENTERTAINMENT	3.996714

The applications with AUTO_AND_VEHICLES, GAME, HEALTH_AND_FITNESS, EDUCATION, PARENTING are the most rated and likely to be turn into a successful projects.

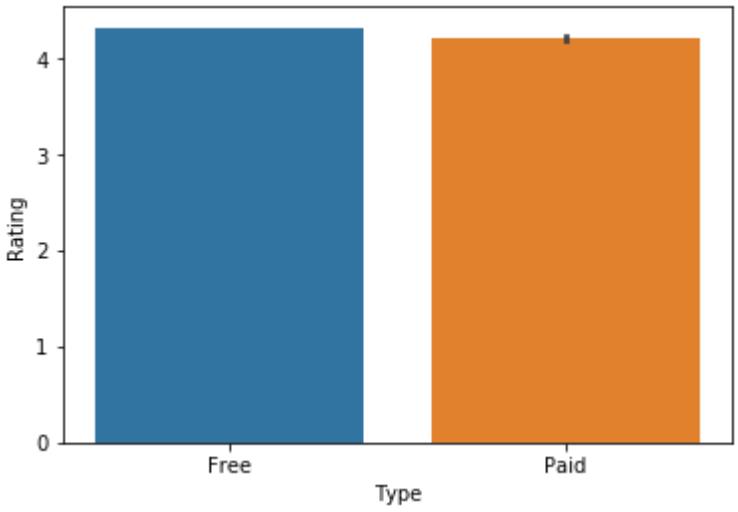
In [138]:
```
sns.countplot(x='Type',data=alldata)
```

Out[138]: <matplotlib.axes._subplots.AxesSubplot at 0x1fac6853588>



In [137]:
```
sns.barplot(x='Type', y='Rating', data=alldata)
```

Out[137]: <matplotlib.axes._subplots.AxesSubplot at 0x1fac67f3e80>



The above two graphs show that- the number of free apps are way more than paid apps. The rating of the application is not affected by its type (paid or free)

In [ ]:
```

```

In [140]:
```
dt_av=alldata.groupby('Android Ver',as_index=False)['Rating'].mean()
dt_av.head(10)
```

Out[140]:

	Android Ver	Rating
0	1.5 and up	4.500000
1	1.6 and up	4.139640
2	2.0 and up	4.200000
3	2.1 and up	4.200000
4	2.2 and up	4.027778
5	2.3 and up	4.313494
6	2.3.3 and up	4.228947
7	3.0 and up	4.208696
8	3.2 and up	4.200000
9	4.0 and up	4.328877

rating is also somehow related to Android version- between version 1.6 to 2.3 the rating of application is low.

Use dataframes, for instance, pandas: Find the top rated application name and its' rating for each category.

In [150]:
```
alldata.groupby(['Category','App'],as_index=False)['Rating'].mean().sort_values(by='Rating',ascending=False).head()
```

Out[150]:

	Category	App	Rating
550	HEALTH_AND_FITNESS	Down Dog: Great Yoga Anywhere	4.9
18	AUTO_AND_VEHICLES	DMV Permit Practice Test 2018 Edition	4.9
148	DATING	FREE LIVE TALK	4.9
16	AUTO_AND_VEHICLES	CDL Practice Test 2018 Edition	4.9
1049	VIDEO_PLAYERS	DU Recorder – Screen Recorder, Video Editor, Live	4.8

In [ ]:

In [ ]: