DATA ANALYSIS OF GOOGLE APP'S RATINGS

Import the Required Modules

In [547]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import scipy.stats as stats
```

Read data

In [548]:
```
google_data=pd.read_csv('Desktop/resume/googleplaystore.csv')
```

In [549]:
```
google_data.head()
```

Out[549]:

	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	De
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 [550]:
```
google_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10841 entries, 0 to 10840
Data columns (total 13 columns):
App               10841 non-null object
Category          10841 non-null object
Rating            9367 non-null float64
Reviews           10841 non-null object
Size              10841 non-null object
Installs          10841 non-null object
Type              10840 non-null object
Price             10841 non-null object
Content Rating    10840 non-null object
Genres            10841 non-null object
Last Updated      10841 non-null object
Current Ver       10833 non-null object
Android Ver       10838 non-null object
dtypes: float64(1), object(12)
memory usage: 1.1+ MB
```

In [551]:
```
google_data.describe()
```

Out[551]:

	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

Data cleaning

Count the number of missing values in each columns

In [552]:
```
google_data.isnull().sum()
```

Out[552]:
```
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
```

Removing records with nulls in any of the columns.

In [553]:
```
google_data.dropna(inplace = True)
google_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9360 entries, 0 to 10840
Data columns (total 13 columns):
App               9360 non-null object
Category          9360 non-null object
Rating            9360 non-null float64
Reviews           9360 non-null object
Size              9360 non-null object
Installs          9360 non-null object
Type              9360 non-null object
Price             9360 non-null object
Content Rating    9360 non-null object
Genres            9360 non-null object
Last Updated      9360 non-null object
Current Ver       9360 non-null object
Android Ver       9360 non-null object
dtypes: float64(1), object(12)
memory usage: 1023.8+ KB
```

Converting Size to numeric(float)

In [554]:
```
google_data["Size"] = google_data["Size"].apply(lambda x: str(x).replace
(",", "") if "," in str(x) else x)
```

In [555]:
```
google_data["Size"] = google_data["Size"].apply(lambda x: str(x).replace
('k', '') if 'k' in str(x) else x)
```

In [556]:
```
google_data["Size"] = google_data["Size"].apply(lambda x: str(x).replace
("Varies with device","NAN")
                                                    if "Varies with device"
in str(x)else x)
```

In [557]:
```
google_data["Size"] = google_data["Size"].apply(lambda x: float(str(x).r
eplace('M', '')) * 1000 if 'M' in str(x) else x)
```

In [558]:
```
google_data["Size"] = google_data["Size"].apply(lambda x:float(x))
```

In [559]:
```
google_data['Size'].astype('float')
```

Out[559]:
```
0        19000.0
1        14000.0
2         8700.0
3        25000.0
4         2800.0
          ...
10834     2600.0
10836    53000.0
10837     3600.0
10839        NaN
10840    19000.0
Name: Size, Length: 9360, dtype: float64
```

Converting Reviews to numeric(int)

In [560]:
```
google_data['Reviews']=pd.to_numeric(google_data['Reviews'],errors='coer
ce')
```

Converting Installs to numeric(int)

In [561]:
```
google_data['Installs']=google_data['Installs'].apply(lambda x : str(x).
replace('+','')if '+'in str(x) else str(x))
```

In [562]:
```
google_data['Installs']=google_data['Installs'].apply(lambda x : str(x).
replace(',','')if ','in str(x) else str(x))
```

In [563]:
```
google_data['Installs']=google_data['Installs'].apply(lambda x :float(x
))
```

In [564]:
```
google_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9360 entries, 0 to 10840
Data columns (total 13 columns):
App               9360 non-null object
Category          9360 non-null object
Rating            9360 non-null float64
Reviews           9360 non-null int64
Size              7723 non-null float64
Installs          9360 non-null float64
Type              9360 non-null object
Price             9360 non-null object
Content Rating    9360 non-null object
Genres            9360 non-null object
Last Updated      9360 non-null object
Current Ver       9360 non-null object
Android Ver       9360 non-null object
dtypes: float64(3), int64(1), object(9)
memory usage: 1023.8+ KB
```

Converting Price to numeric(int)

In [565]:
```
google_data['Price']=google_data['Price'].apply(lambda x: str(x).replace
(('$'),'')if '$'in str(x) else str(x))
```

In [566]:
```
google_data['Price']=google_data['Price'].apply(lambda x:float(x))
```

In [567]:
```
google_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9360 entries, 0 to 10840
Data columns (total 13 columns):
App               9360 non-null object
Category          9360 non-null object
Rating            9360 non-null float64
Reviews           9360 non-null int64
Size              7723 non-null float64
Installs          9360 non-null float64
Type              9360 non-null object
Price             9360 non-null float64
Content Rating    9360 non-null object
Genres            9360 non-null object
Last Updated      9360 non-null object
Current Ver       9360 non-null object
Android Ver       9360 non-null object
dtypes: float64(4), int64(1), object(8)
memory usage: 1023.8+ KB
```

Sanity checks

In [568]:
```
google_data[(google_data['Rating']<1) & (google_data['Rating']> 5)]
```

Out[568]:

App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Curren Ve

In [569]:
```
google_data[google_data['Reviews']>google_data["Installs"]]
```

Out[569]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	U
2454	KBA-EZ Health Guide	MEDICAL	5.0	4	25000.0	1.0	Free	0.00	Everyone	Medical	
4663	Alarmy (Sleep If U Can) - Pro	LIFESTYLE	4.8	10249	NaN	10000.0	Paid	2.49	Everyone	Lifestyle	
5917	Ra Ga Ba	GAME	5.0	2	20000.0	1.0	Paid	1.49	Everyone	Arcade	
6700	Brick Breaker BR	GAME	5.0	7	19000.0	5.0	Free	0.00	Everyone	Arcade	
7402	Trovami se ci riesci	GAME	5.0	11	6100.0	10.0	Free	0.00	Everyone	Arcade	
8591	DN Blog	SOCIAL	5.0	20	4200.0	10.0	Free	0.00	Teen	Social	
10697	Mu.F.O.	GAME	5.0	2	16000.0	1.0	Paid	0.99	Everyone	Arcade	

In [570]:
```
google_data.drop([2454,4663,5917,6700,7402,8591,10697],inplace=True)
```

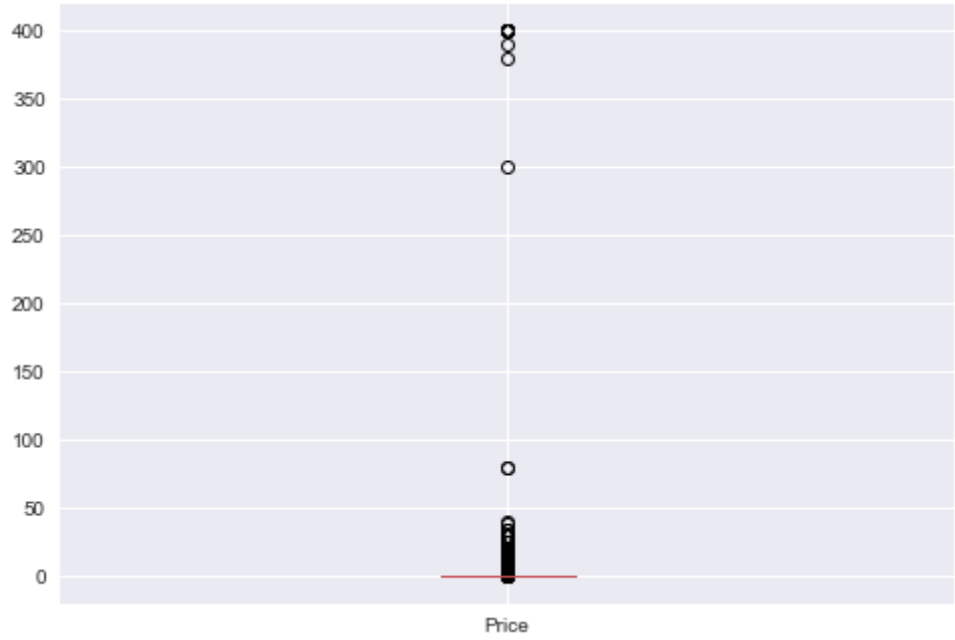
In [571]:
```
google_data[(google_data['Type']=='Free') & (google_data['Price']>0)]
```

Out[571]:

App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Curren Ve

Performing univariate analysis:

In [572]:
```
plt.style.use('seaborn')
google_data.boxplot('Price')
plt.show()
```



In [573]:
```
print("No. of Apps with low price: ",google_data.Price[google_data['Pric
e'] ==  0 ].count())
```

```
No. of Apps with low price:  8711
```

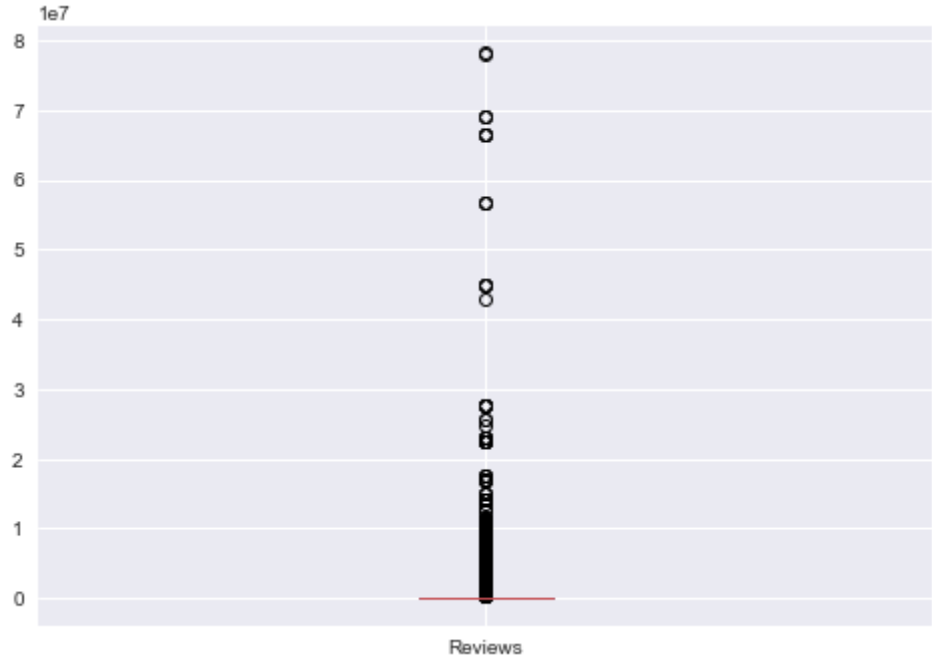
In [574]:
```
google_data[google_data['Price'] == 0 ].head()
```

Out[574]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19000.0	10000.0	Free	0.0	Everyone	
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14000.0	500000.0	Free	0.0	Everyone	
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8700.0	5000000.0	Free	0.0	Everyone	
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25000.0	50000000.0	Free	0.0	Teen	
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2800.0	100000.0	Free	0.0	Everyone	D

Insight: There is outliers . Outliers lies above 200 . The price of the most of the app is 0 i.e free .

In [575]:
```
plt.style.use('seaborn')
google_data.boxplot('Reviews')
plt.show()
```



In [576]:
```
print("No. of Apps with high Reviews: ",google_data.Reviews[google_data[
'Reviews'] > 400000].count())
```

```
No. of Apps with high Reviews:  1163
```

In [577]:
```
google_data[google_data['Reviews'] > 400000]
```

Out[577]:

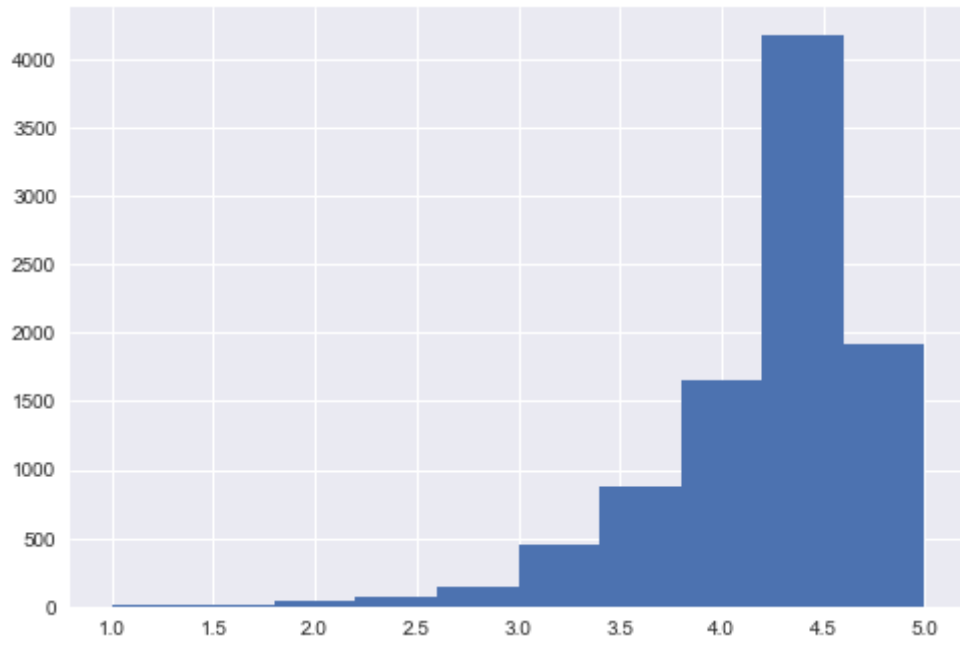
	App	Category	Rating	Reviews	Size	Installs	Type	Pr
139	Wattpad 📖 Free Books	BOOKS_AND_REFERENCE	4.6	2914724	NaN	1.000000e+08	Free	
142	Wikipedia	BOOKS_AND_REFERENCE	4.4	577550	NaN	1.000000e+07	Free	
143	Amazon Kindle	BOOKS_AND_REFERENCE	4.2	814080	NaN	1.000000e+08	Free	
145	Dictionary - Merriam-Webster	BOOKS_AND_REFERENCE	4.5	454060	NaN	1.000000e+07	Free	
152	Google Play Books	BOOKS_AND_REFERENCE	3.9	1433233	NaN	1.000000e+09	Free	
...	...	...	...	...	...	...	...	
10507	Rope Hero: Vice Town	GAME	4.4	452589	99000.0	1.000000e+07	Free	
10513	Flight Simulator: Fly Plane 3D	FAMILY	4.0	660613	21000.0	5.000000e+07	Free	
10636	FRONTLINE COMMANDO	GAME	4.4	1351833	12000.0	1.000000e+07	Free	
10707	Photo Editor Collage Maker Pro	PHOTOGRAPHY	4.5	1519671	NaN	1.000000e+08	Free	
10781	Modern Strike Online	GAME	4.3	834117	44000.0	1.000000e+07	Free	

1163 rows × 13 columns

Insight: There is outliers which indicates there are app which have higher reviews which is above 4 million and it can mislead the analysis.

In [578]:
```
plt.style.use('seaborn')
google_data['Rating'].hist()
```

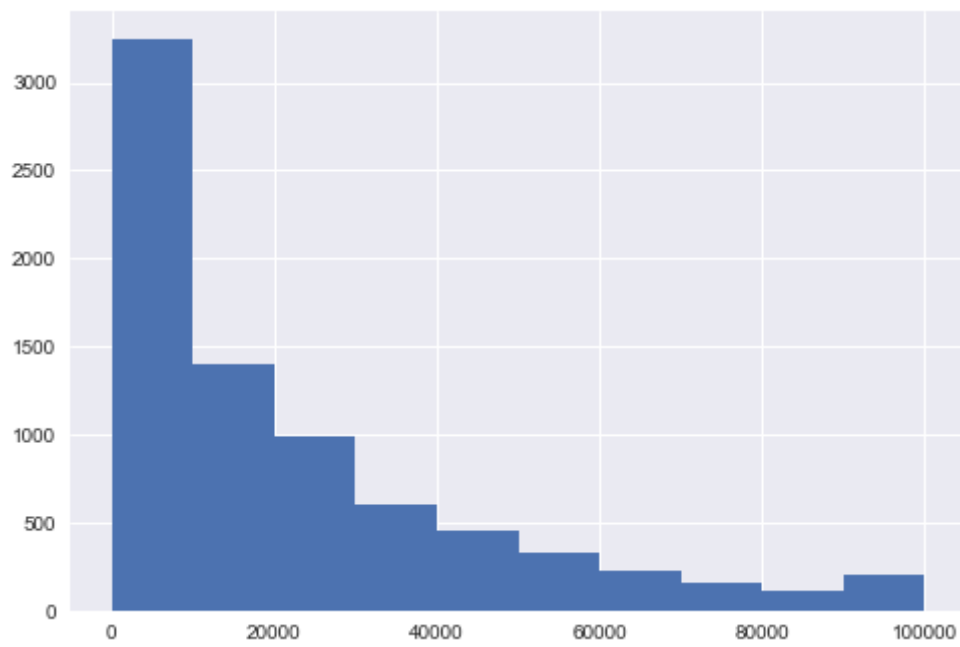
Out[578]: <matplotlib.axes._subplots.AxesSubplot at 0x1a279c5550>



Insight: Rating is distributed towards right between 4 to 5.

In [579]:
```
google_data['Size'].hist()
```

Out[579]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2bd54f10>



Insight: Size is not normally distributed and skewness is towards left.

Outlier Treatment

In [580]:
```
google_data [google_data['Price']>200]
```

Out[580]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	
4197	most expensive app (H)	FAMILY	4.3	6	1500.0	100.0	Paid	399.99	Everyone	Ente
4362	💎 I'm rich	LIFESTYLE	3.8	718	26000.0	10000.0	Paid	399.99	Everyone	
4367	I'm Rich - Trump Edition	LIFESTYLE	3.6	275	7300.0	10000.0	Paid	400.00	Everyone	
5351	I am rich	LIFESTYLE	3.8	3547	1800.0	100000.0	Paid	399.99	Everyone	
5354	I am Rich Plus	FAMILY	4.0	856	8700.0	10000.0	Paid	399.99	Everyone	Ente
5355	I am rich VIP	LIFESTYLE	3.8	411	2600.0	10000.0	Paid	299.99	Everyone	
5356	I Am Rich Premium	FINANCE	4.1	1867	4700.0	50000.0	Paid	399.99	Everyone	
5357	I am extremely Rich	LIFESTYLE	2.9	41	2900.0	1000.0	Paid	379.99	Everyone	
5358	I am Rich!	FINANCE	3.8	93	22000.0	1000.0	Paid	399.99	Everyone	
5359	I am rich(premium)	FINANCE	3.5	472	965.0	5000.0	Paid	399.99	Everyone	
5362	I Am Rich Pro	FAMILY	4.4	201	2700.0	5000.0	Paid	399.99	Everyone	Ente
5364	I am rich (Most expensive app)	FINANCE	4.1	129	2700.0	1000.0	Paid	399.99	Teen	
5366	I Am Rich	FAMILY	3.6	217	4900.0	10000.0	Paid	389.99	Everyone	Ente
5369	I am Rich	FINANCE	4.3	180	3800.0	5000.0	Paid	399.99	Everyone	
5373	I AM RICH PRO PLUS	FINANCE	4.0	36	41000.0	1000.0	Paid	399.99	Everyone	

In [581]:
```
print("No. of Apps with high price: ",google_data.Price[google_data['Pri
ce'] > 200 ].count())
```

```
No. of Apps with high price:  15
```

In [582]:
```
index_price=google_data[google_data['Price']>200].index
```

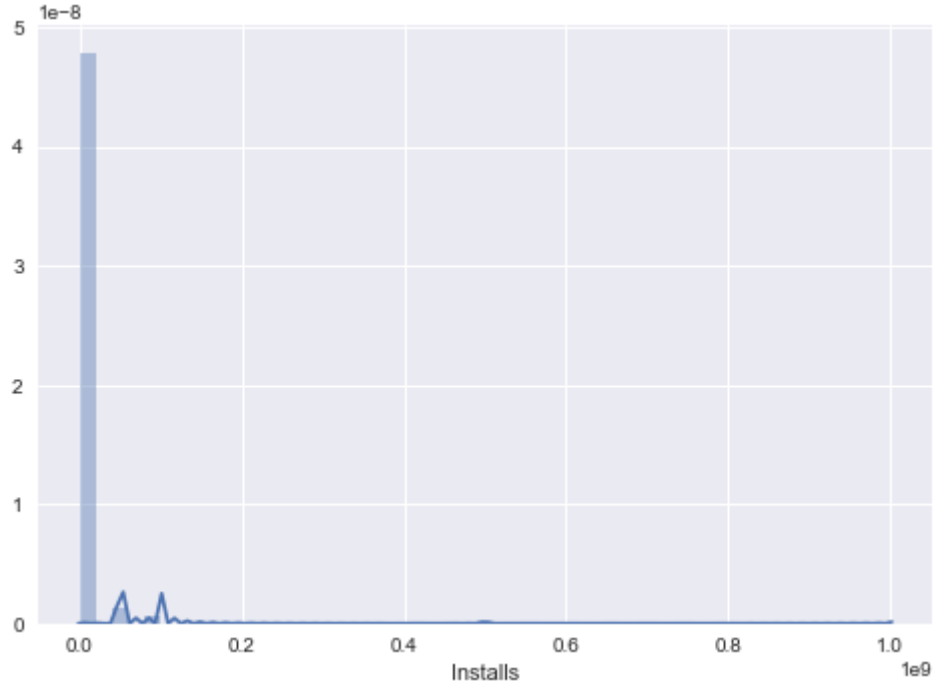
In [583]:
```
google_data.drop(index_price,inplace=True)
```

In [584]:
```
review_index = google_data[google_data['Reviews']> 2000000].index
```

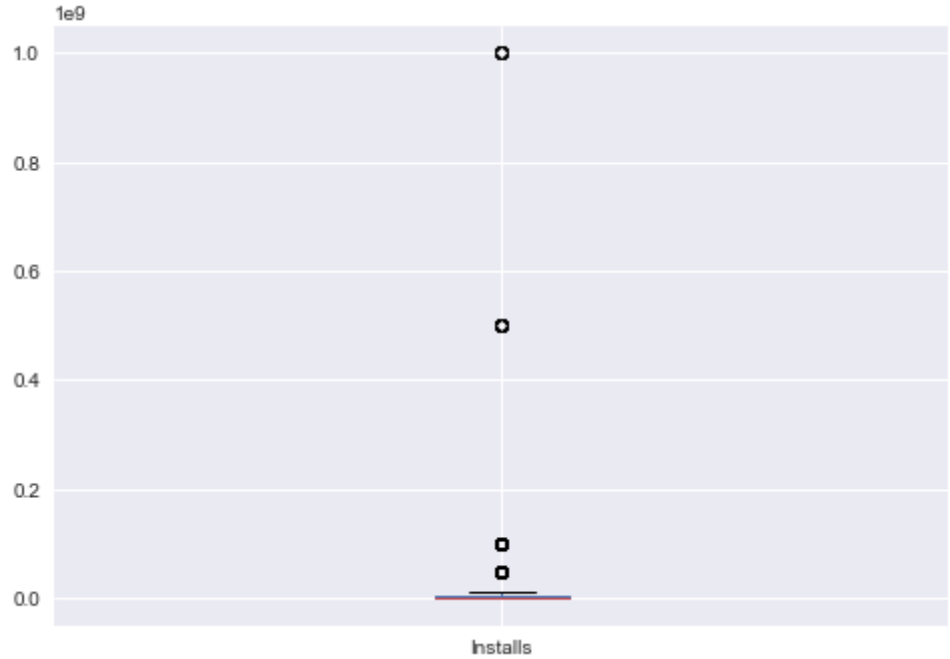
In [585]:
```
google_data.drop(review_index,inplace=True)
```

In [586]:
```
sns.distplot(google_data.Installs)
```

Out[586]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2be76f90>
```



In [587]:
```
plt.style.use('seaborn')
google_data.boxplot('Installs')
plt.show()
```



In [588]:
```
Installs_per=google_data.Installs
```

In [589]:
```
percentile_ten=np.percentile(Installs_per,10)
percentile_twentyfive=np.percentile(Installs_per,25)
percentile_fifty=np.percentile(Installs_per,50)
percentile_seventy=np.percentile(Installs_per,70)
percentile_ninty=np.percentile(Installs_per,90)
percentile_nintyfive=np.percentile(Installs_per,95)
percentile_nintynine=np.percentile(Installs_per,99)
```

In [590]:
```
percentile_ten
```

Out[590]: 1000.0

In [591]:
```
percentile_twentyfive
```

Out[591]: 10000.0

In [592]:
```
percentile_fifty
```

Out[592]: 500000.0

In [593]:
```
percentile_seventy
```

Out[593]: 1000000.0

In [594]:
```
percentile_ninty
```

Out[594]: 10000000.0

In [595]:
```
percentile_nintyfive
```

Out[595]: 10000000.0

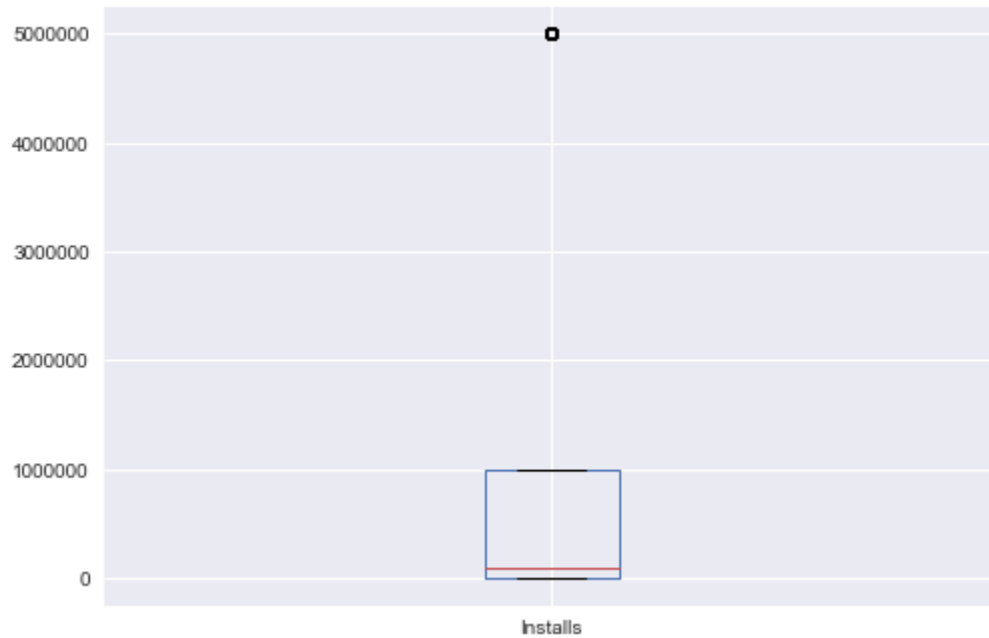
In [596]:
```
percentile_nintynine
```

Out[596]: 100000000.0

In [597]:
```
threshold = percentile_ninty
install_index =  google_data[google_data['Installs']>= threshold].index
```

In [598]:
```
google_data.drop(install_index,inplace=True)
```

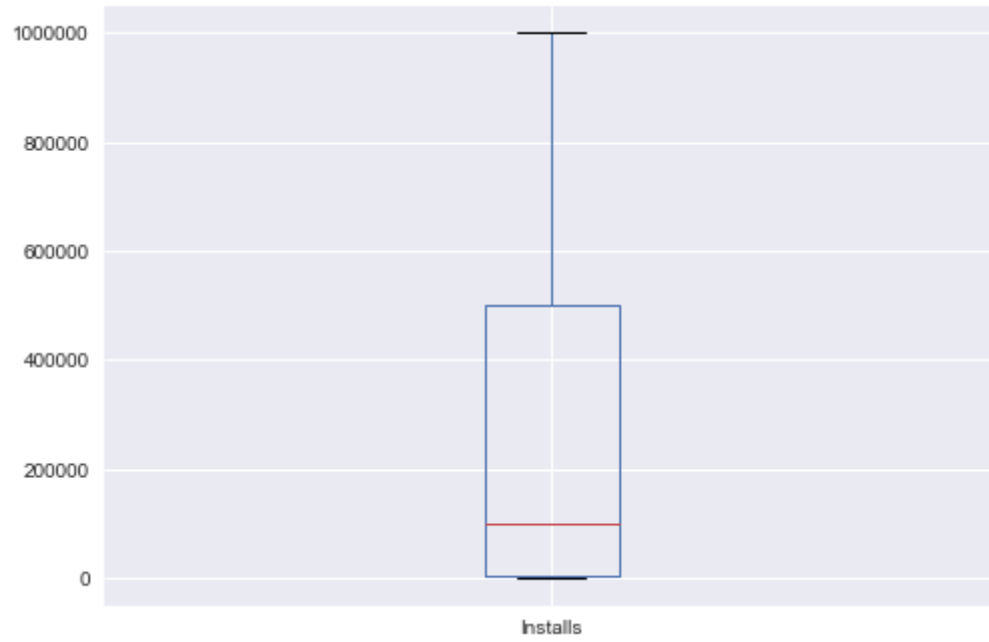
In [599]:
```
plt.style.use('seaborn')
google_data.boxplot('Installs')
plt.show()
```



In [600]:
```
index_outlier=google_data[google_data['Installs']==5000000].index
```

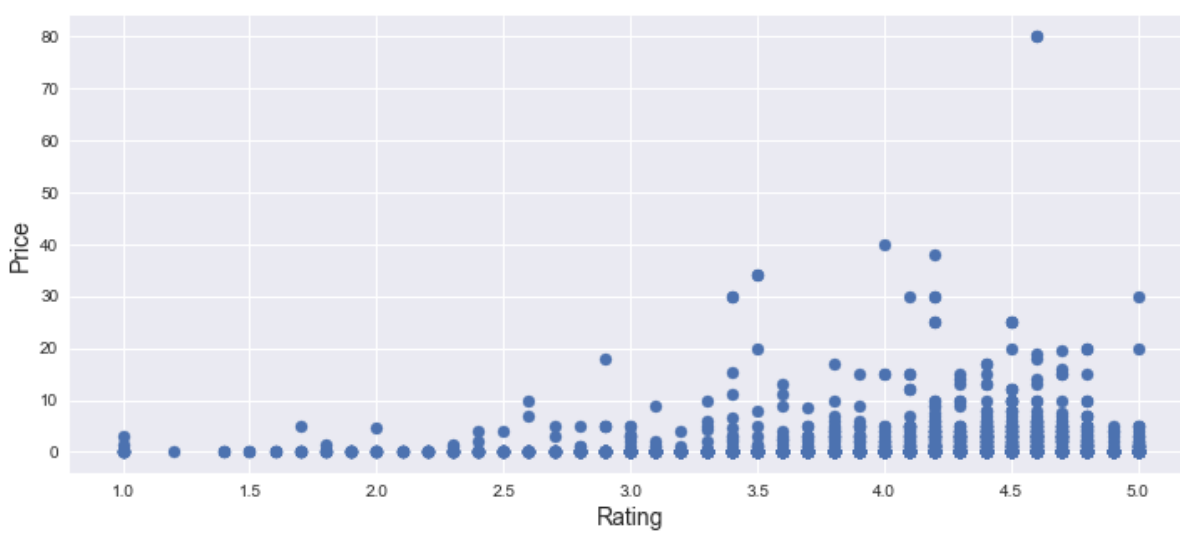
In [601]:
```
google_data.drop(index_outlier,inplace=True)
```

In [602]:
```
plt.style.use('seaborn')
google_data.boxplot('Installs')
plt.show()
```



Bivariate analysis

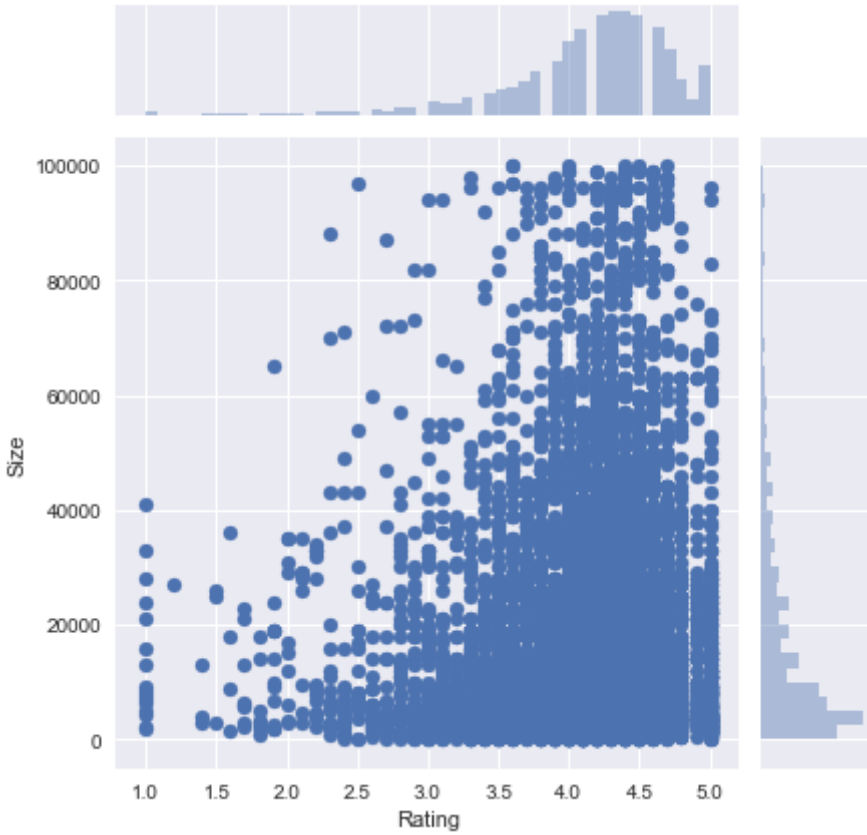
In [603]:
```
plt.figure(figsize=(12,5))
plt.scatter(google_data.Rating,google_data.Price)
plt.xlabel('Rating', size = 14)
plt.ylabel('Price', size = 14)
plt.show()
```



Insight: Rating increases with Price

In [604]:
```
sns.jointplot(data=google_data,x='Rating', y='Size',kind='scatter')
```

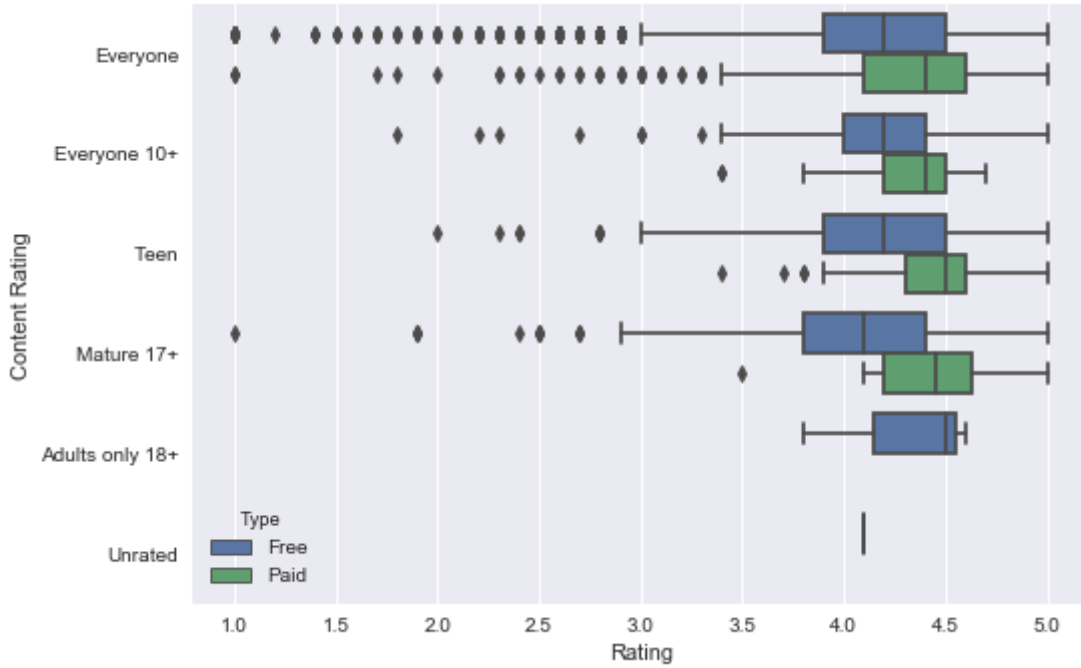
Out[604]:
```
<seaborn.axisgrid.JointGrid at 0x1a2c22b9d0>
```



Insight: Yes ,heavier app are rated from 4 to 5.

In [605]:
```
sns.boxplot(data=google_data,x='Rating', y='Content Rating',hue='Type')
```

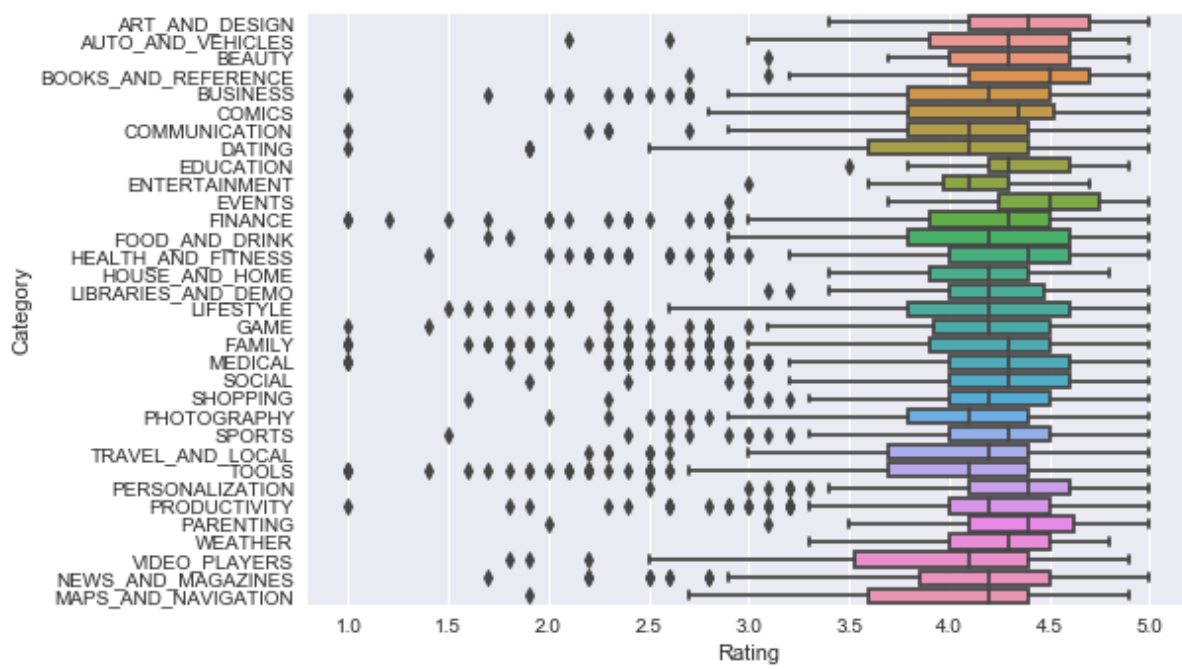
Out[605]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2c391f50>
```



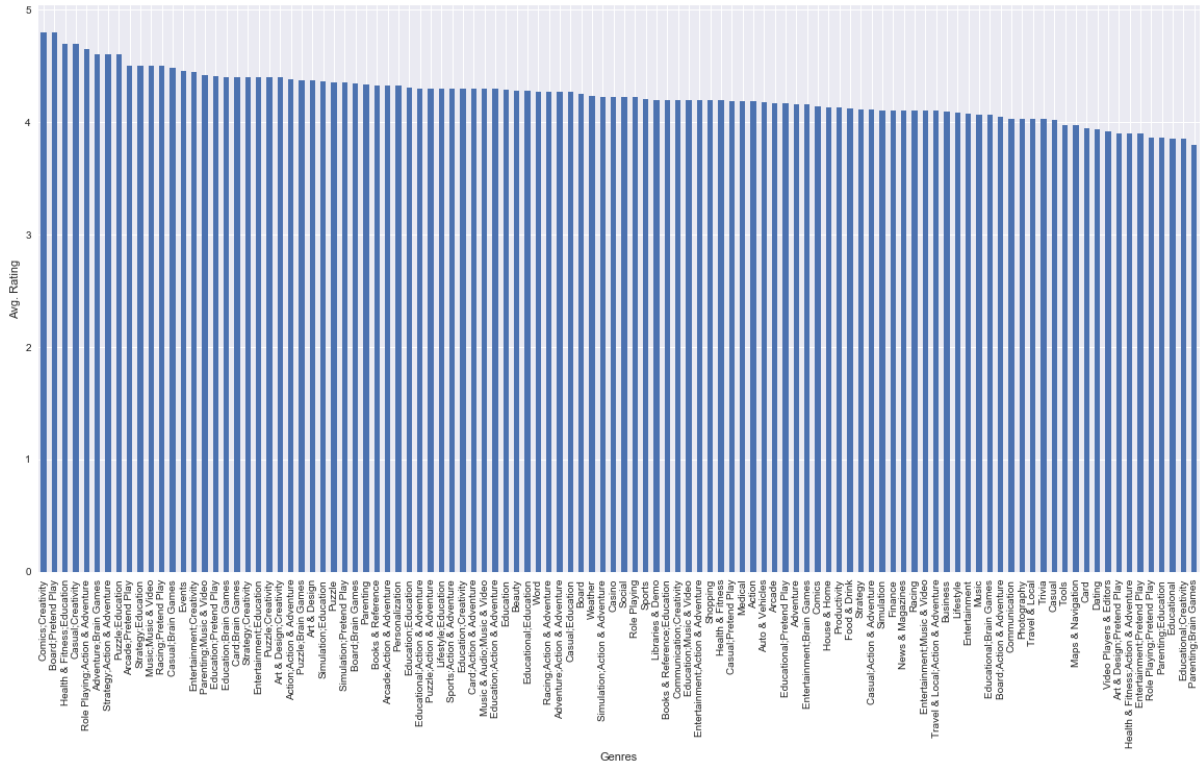
Insight:Yes , there is difference in Rating . Everyone has highest rating than others.

In [606]:
```
sns.boxplot(data=google_data,x='Rating', y='Category')
```

Out[606]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2da885d0>
```



In [607]:
```
google_data.groupby('Genres').Rating.mean().sort_values(ascending=False)
.plot(kind='bar',figsize=(20,10))
plt.ylabel('Avg. Rating')
plt.show()
```



Insight: Comics.,creativity and Board Pretend play have highest rating.

Data preprocessing

In [608]:
```
google_data.to_csv(r'/Users/ajesh_mahto/Desktop/googleStore_project/goog
lestore_input2.csv', index = False)
```

In [609]:
```
intt1=pd.read_csv(r'/Users/ajesh_mahto/Desktop/googleStore_project/googl
estore_input2.csv')
```

In [610]:
```
intt1
```

Out[610]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Co
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19000.0	10000.0	Free	0.0	Ev
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14000.0	500000.0	Free	0.0	Ev
2	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2800.0	100000.0	Free	0.0	Ev
3	Paper flowers instructions	ART_AND_DESIGN	4.4	167	5600.0	50000.0	Free	0.0	Ev
4	Smoke Effect Photo Maker - Smoke Editor	ART_AND_DESIGN	3.8	178	19000.0	50000.0	Free	0.0	Ev
...	...	...	...	...	...	...	...	...	
6501	Chemin (fr)	BOOKS_AND_REFERENCE	4.8	44	619.0	1000.0	Free	0.0	Ev
6502	FR Calculator	FAMILY	4.0	7	2600.0	500.0	Free	0.0	Ev
6503	Sya9a Maroc - FR	FAMILY	4.5	38	53000.0	5000.0	Free	0.0	Ev
6504	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3600.0	100.0	Free	0.0	Ev
6505	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5	114	NaN	1000.0	Free	0.0	

6506 rows × 13 columns

In [611]:
```
intt1.isnull().sum()
```

Out[611]:
```
App                 0
Category            0
Rating              0
Reviews             0
Size              552
Installs            0
Type                0
Price               0
Content Rating      0
Genres              0
Last Updated        0
Current Ver         0
Android Ver         0
dtype: int64
```

In [612]:
```
intt1.dropna(inplace=True)
```

In [613]:
```
intt1.isnull().sum()
```

Out[613]:
```
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 [614]:
```
Reviews_log=np.log1p(intt1['Reviews'])
```

In [615]:
```
Reviews_log
```

Out[615]:
```
0       5.075174
1       6.875232
2       6.875232
3       5.123964
4       5.187386
          ...
6500    7.086738
6501    3.806662
6502    2.079442
6503    3.663562
6504    1.609438
Name: Reviews, Length: 5954, dtype: float64
```

In [616]:
```
intt1['Reviews'] = Reviews_log
```

In [617]:
```
Install_log=np.log1p(intt1['Installs'])
```

In [618]:
```
intt1['Installs']=Install_log
```

In [619]:
```
intt1.head()
```

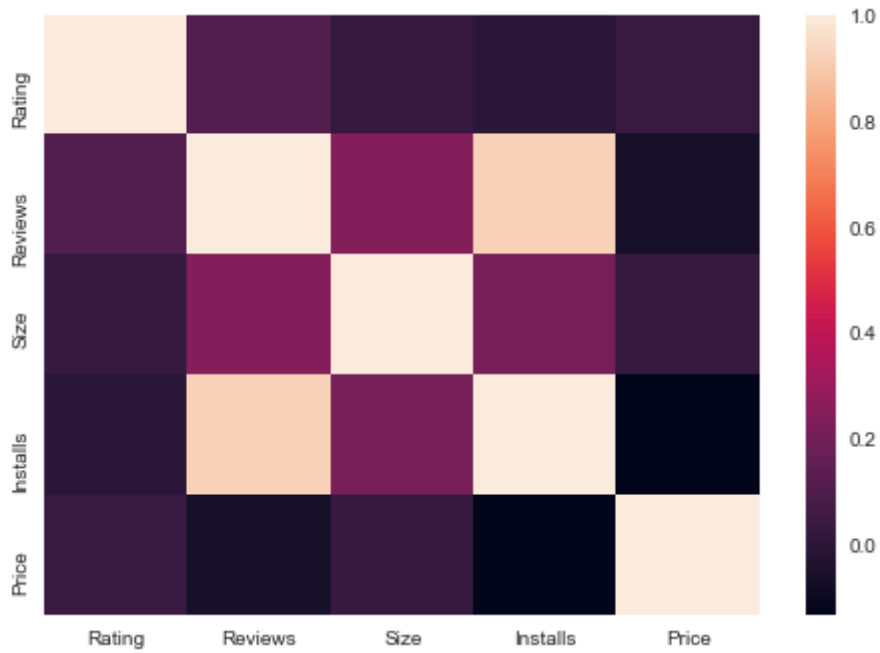
Out[619]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	5.075174	19000.0	9.210440	Free	0.0	Everyone
1	Coloring book moana	ART_AND_DESIGN	3.9	6.875232	14000.0	13.122365	Free	0.0	Everyone
2	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	6.875232	2800.0	11.512935	Free	0.0	Everyone
3	Paper flowers instructions	ART_AND_DESIGN	4.4	5.123964	5600.0	10.819798	Free	0.0	Everyone
4	Smoke Effect Photo Maker - Smoke Editor	ART_AND_DESIGN	3.8	5.187386	19000.0	10.819798	Free	0.0	Everyone

In [620]:
```
intt1.drop(['App', 'Last Updated', 'Current Ver',  'Android Ver'], axis=
1, inplace=True)
```

In [621]:
```
intt1.corr()
sns.heatmap(intt1.corr())
```

Out[621]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2e75de10>



In [622]:
```
 int2=pd.get_dummies(intt1, columns=['Category', 'Genres', 'Content Rati
ng','Type'], drop_first=True)
```

In [623]:
```
int2.head()
```

Out[623]:

	Rating	Reviews	Size	Installs	Price	Category_AUTO_AND_VEHICLES	Category_BEAUT
0	4.1	5.075174	19000.0	9.210440	0.0	0	
1	3.9	6.875232	14000.0	13.122365	0.0	0	
2	4.3	6.875232	2800.0	11.512935	0.0	0	
3	4.4	5.123964	5600.0	10.819798	0.0	0	
4	3.8	5.187386	19000.0	10.819798	0.0	0	

5 rows × 148 columns

In [624]:
```
int2.describe()
```

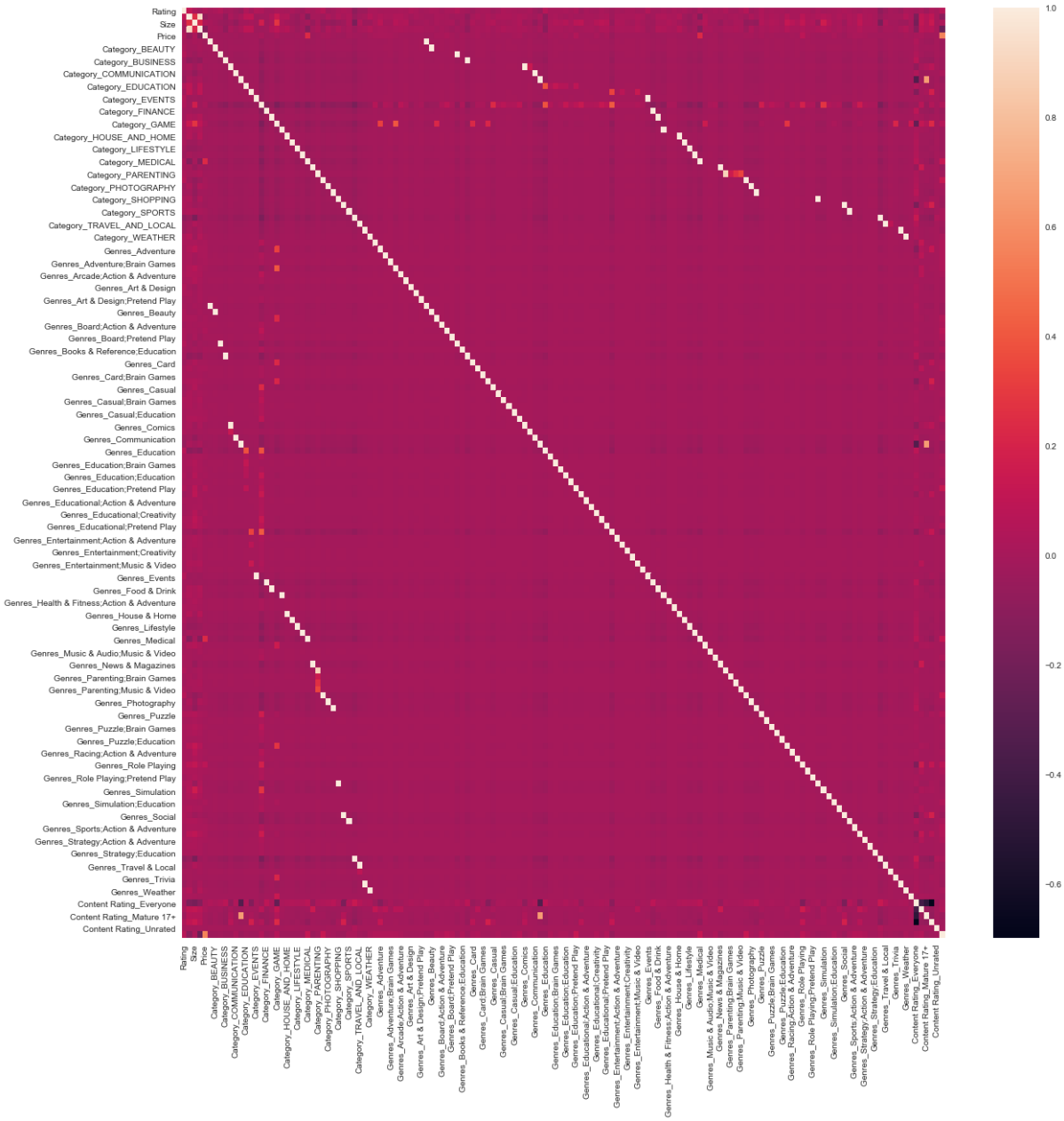
Out[624]:

	Rating	Reviews	Size	Installs	Price	Category_AUTO_AND_
count	5954.000000	5954.000000	5954.000000	5954.000000	5954.000000	5
mean	4.135119	6.330305	19326.390914	10.277487	0.476910	
std	0.594299	2.873851	20816.835745	2.953172	2.661021	
min	1.000000	0.693147	8.500000	1.791759	0.000000	
25%	3.900000	3.988984	4325.000000	8.517393	0.000000	
50%	4.200000	6.355236	11000.000000	10.819798	0.000000	
75%	4.500000	8.770747	27000.000000	13.122365	0.000000	
max	5.000000	12.837940	100000.000000	13.815512	79.990000	

8 rows × 148 columns

In [625]:
```
plt.figure(figsize=(20,20))
int2.corr()
sns.heatmap(int2.corr())
```

Out[625]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2e00fbd0>



Training, Testing and Validation

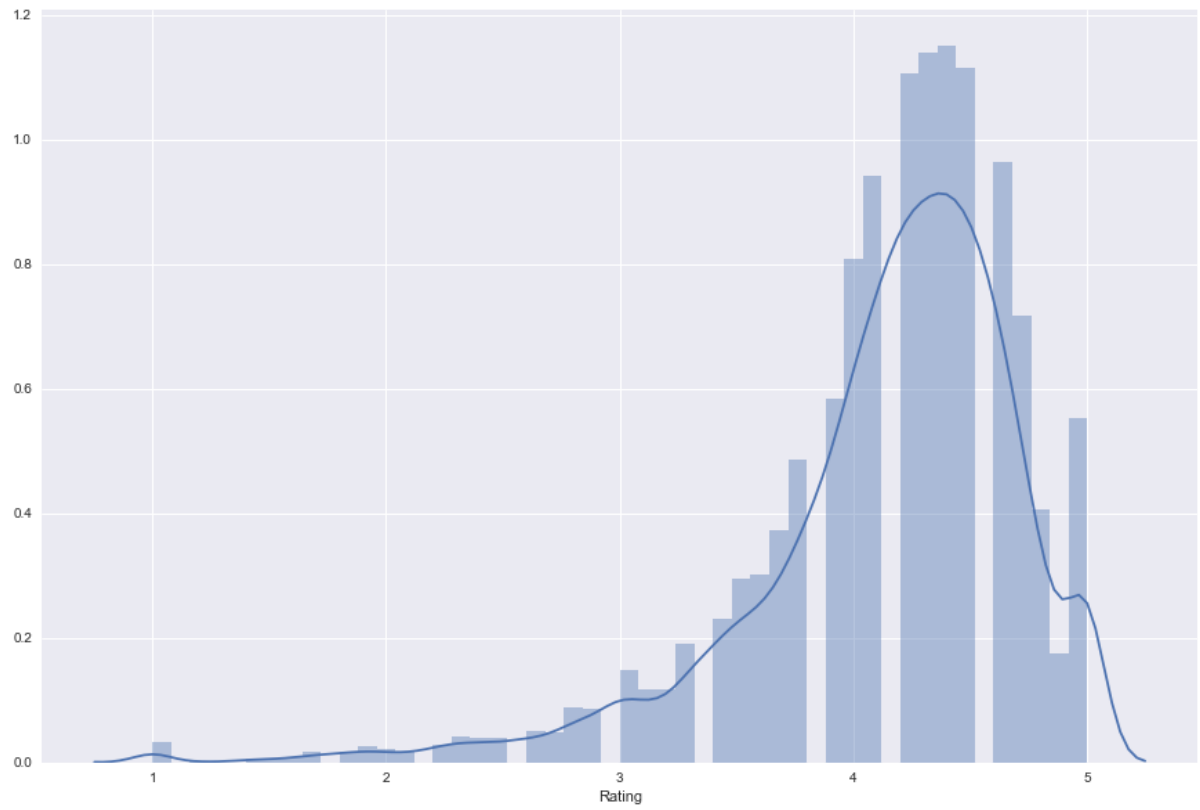
Here we define which columns are features in X and label in Y. The label is Rating, and features are the rest of the columns

In [645]:
```
x = int2.drop(columns=['Rating'], axis=1)

y = int2['Rating']
```

Let's check the average value of the “Rating” column.

In [646]:
```
plt.figure(figsize=(15,10))
plt.tight_layout()
sns.distplot(int2['Rating'])
```

Out[646]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2c398c10>
```



As we can observe that most of the time the value lies between 4.2 to 4.5

Next, we split 70% of the data to the training set while 30% of the data to test set using below code.

In [647]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
%matplotlib inline
```

In [648]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state=0)
```

In [649]:
```
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

Out[649]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normaliz
e=False)
```

In [650]:
```
#To retrieve the intercept:
print(regressor.intercept_)
```

```
4.621647685163875
```

In [651]:
```
import pandas as pd
```

In [652]:
```
coeff_df = pd.DataFrame(regressor.coef_, x.columns, columns=['Coefficien
t'])
coeff_df
```

Out[652]:

	Coefficient
Reviews	1.822981e-01
Size	-5.170894e-07
Installs	-1.655434e-01
Price	-3.276466e-03
Category_AUTO_AND_VEHICLES	1.623444e-01
...	...
Content Rating_Everyone 10+	-2.179903e-01
Content Rating_Mature 17+	-2.045781e-01
Content Rating_Teen	-1.950232e-01
Content Rating_Unrated	0.000000e+00
Type_Paid	-6.334366e-02

147 rows × 1 columns

In [635]:
```
y_pred = regressor.predict(x_test)
```

In [636]:
```
y_pred
```

Out[636]:
```
array([4.15980528, 4.05346446, 4.48029126, ..., 4.89552226, 4.09991528,
       3.91879625])
```

In [637]:
```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
```

In [638]:
```
df
```

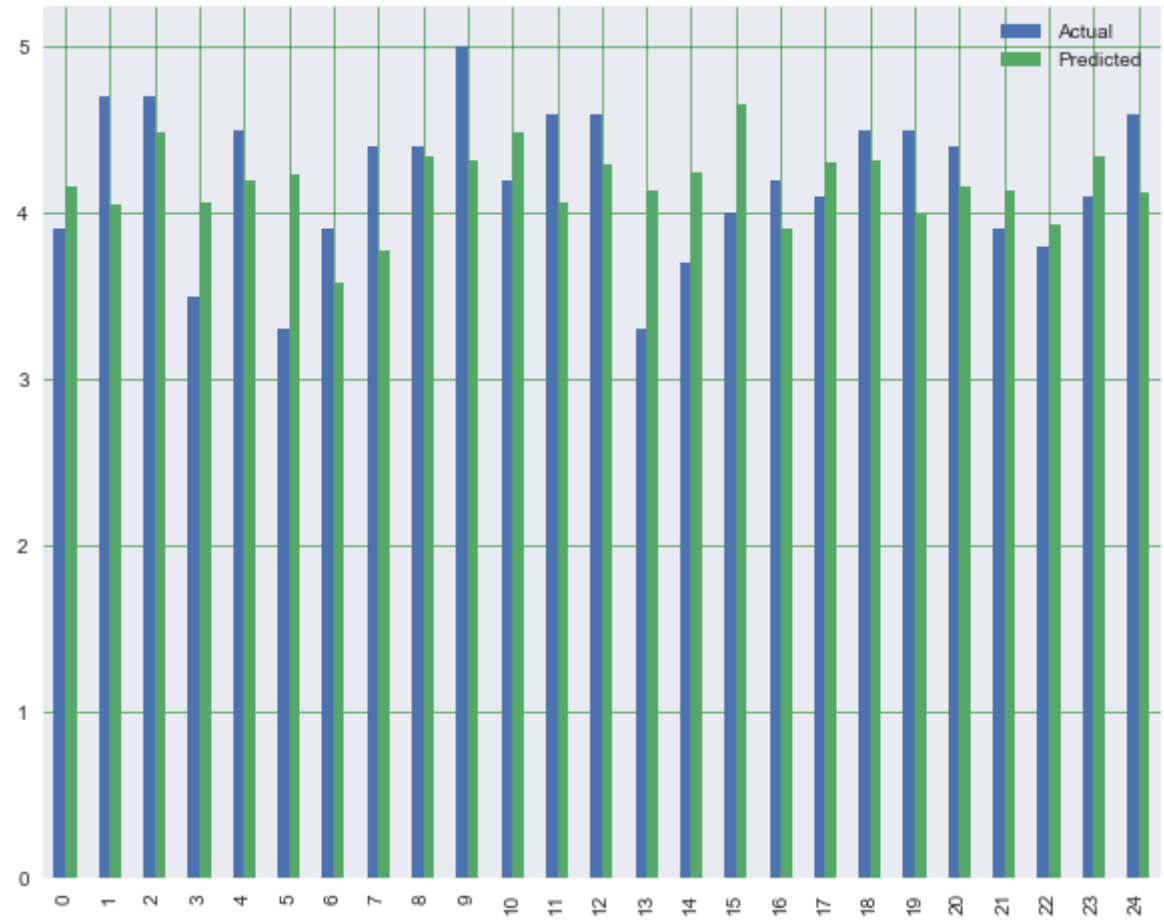
Out[638]:

	Actual	Predicted
0	3.9	4.159805
1	4.7	4.053464
2	4.7	4.480291
3	3.5	4.057624
4	4.5	4.197072
...	...	...
1782	3.4	3.687711
1783	4.6	4.295746
1784	4.5	4.895522
1785	4.5	4.099915
1786	4.1	3.918796

1787 rows × 2 columns

In [639]:
```
df1 = df.head(25)
```

In [640]:
```
df1.plot(kind='bar',figsize=(10,8))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
```

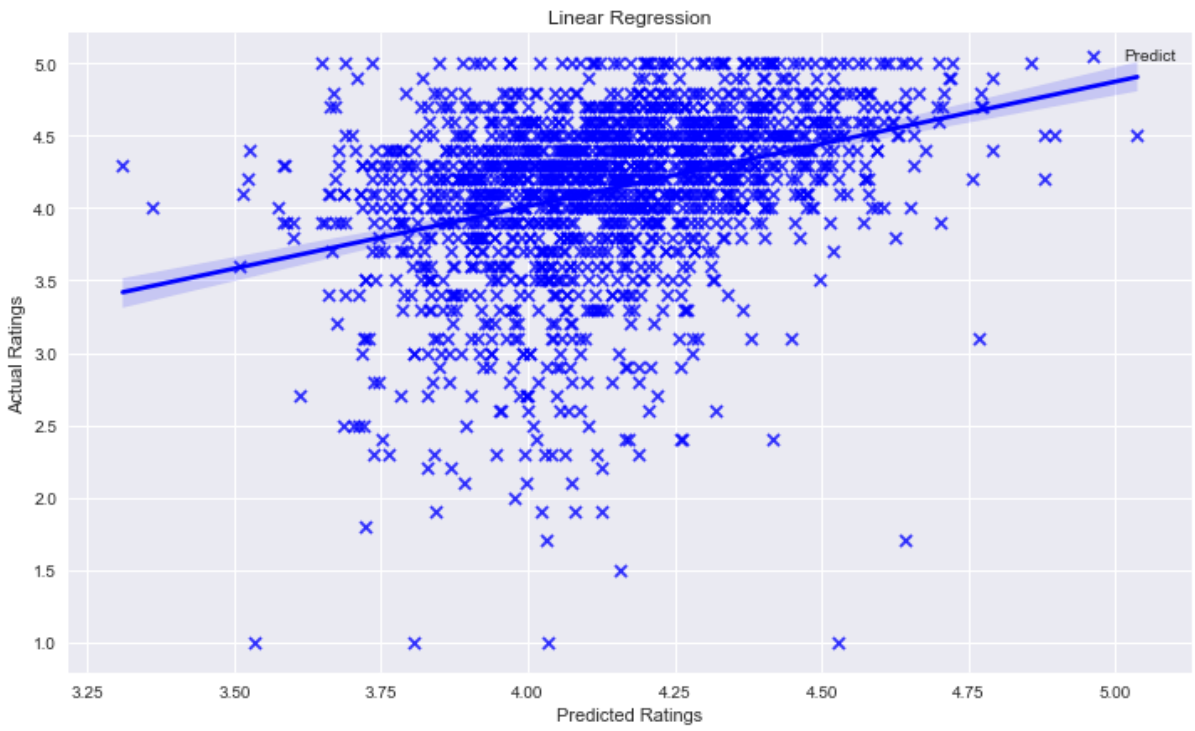


In [641]:
```
from sklearn.metrics import mean_squared_error, r2_score
```

In [642]:
```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred
))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_t
est, y_pred)))
print("R2 Score: ", r2_score(y_test,y_pred ))
```

```
Mean Absolute Error: 0.39929447970192616
Mean Squared Error: 0.3013683705858034
Root Mean Squared Error: 0.548970282060699
R2 Score:  0.11172384810720948
```

In [643]:
```
plt.figure(figsize=(12,7))
sns.regplot(y_pred,y_test,color='blue', label = 'Predict', marker = 'x')
plt.legend()
plt.title('Linear Regression')
plt.xlabel('Predicted Ratings')
plt.ylabel('Actual Ratings')
plt.show()
```



Insight: R2 is usually low in case of human behaviour as associated with rating.

This means that our algorithm was not very accurate but can still make reasonably good predictions.

There are many factors that may have contributed to this inaccuracy, for example :Need more data: We need to have a huge amount of data to get the best possible prediction.Bad assumptions: We made the assumption that this data has a linear relationship, but that might not be the case. Visualizing the data may help you determine that.Poor features: The features we used may not have had a high enough correlation to the values we were trying to predict.

Futher Analysis:

In [656]:
```
 google_rating=pd.read_csv(r'/Users/ajesh_mahto/Desktop/googleStore_proj
ect/googlestore_input2.csv')
```

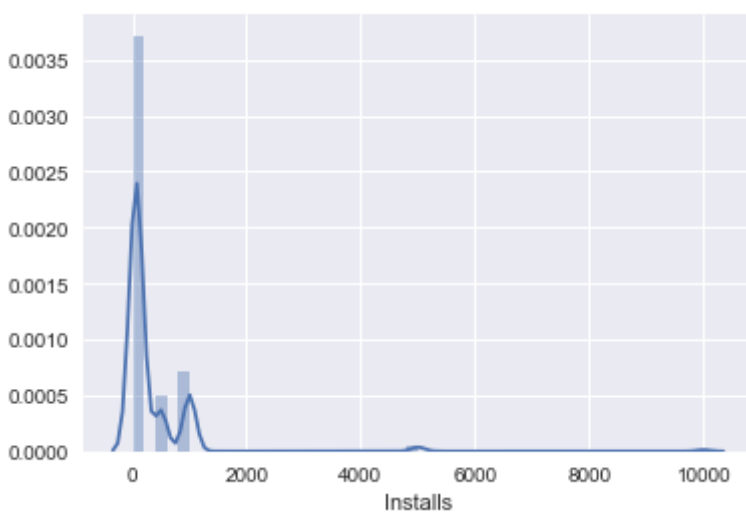
In [658]:
```
df_full = google_rating[google_rating.Rating == 5]

df_full.head()
```

Out[658]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Upd
239	Hojiboy Tojiboyev Life Hacks	COMICS	5.0	15	37000.0	1000.0	Free	0.0	Everyone	Comics	26-
388	American Girls Mobile Numbers	DATING	5.0	5	4400.0	1000.0	Free	0.0	Mature 17+	Dating	17
390	Awake Dating	DATING	5.0	2	70000.0	100.0	Free	0.0	Mature 17+	Dating	24
396	Spine-The dating app	DATING	5.0	5	9300.0	500.0	Free	0.0	Teen	Dating	14
398	Girls Live Talk - Free Text and Video Chat	DATING	5.0	6	5000.0	100.0	Free	0.0	Mature 17+	Dating	1-

In [659]:
```
sns.distplot(df_full.Installs)
```

Out[659]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2e1a9410>



In [660]:
```
df_full.Installs.value_counts().sort_index()
```

Out[660]:
```
5.0          7
10.0        46
50.0        32
100.0      114
500.0       27
1000.0      38
5000.0       3
10000.0      1
Name: Installs, dtype: int64
```

Insight : There are many Apps that have full ratings but less downloads/installs. So we can't really consider those apps as the best ones.

Considering the Apps with 5.0 Ratings and Maximum Installs

In [661]:
```
df_full_maxinstalls = df_full[google_rating.Installs > 1000]

df_full_maxinstalls[['App', 'Category', 'Installs']]
```

```
/Users/ajesh_mahto/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:1: UserWarning: Boolean Series key will be reindexed to mat
ch DataFrame index.
  """Entry point for launching an IPython kernel.
```

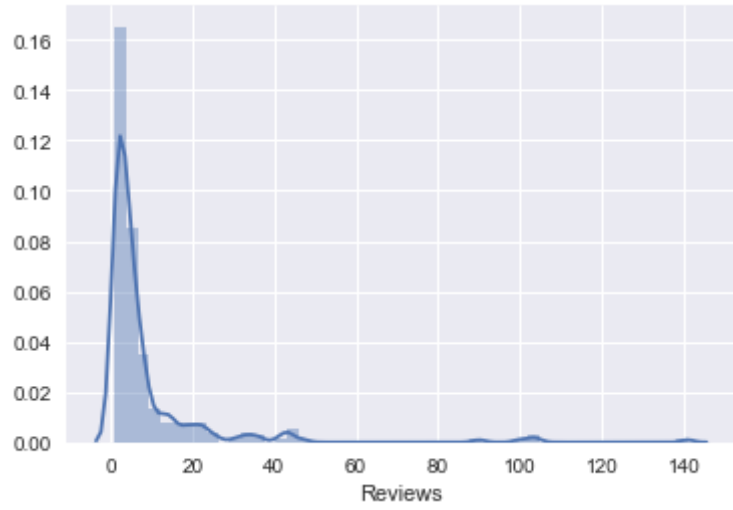
Out[661]:

	App	Category	Installs
4285	CL Keyboard - Myanmar Keyboard (No Ads)	TOOLS	5000.0
4711	Oración CX	LIFESTYLE	5000.0
4828	Superheroes, Marvel, DC, Comics, TV, Movies News	COMICS	5000.0
5668	Ek Bander Ne Kholi Dukan	FAMILY	10000.0

Checking the No. of Reviews of 5.0 Rating Apps

In [662]:
```
sns.distplot(df_full.Reviews)
```

Out[662]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a28f89c10>
```



The above distribution is clearly skewed. Apps with very few reviews easily managed to get 5.0 ratings which can be misleading.

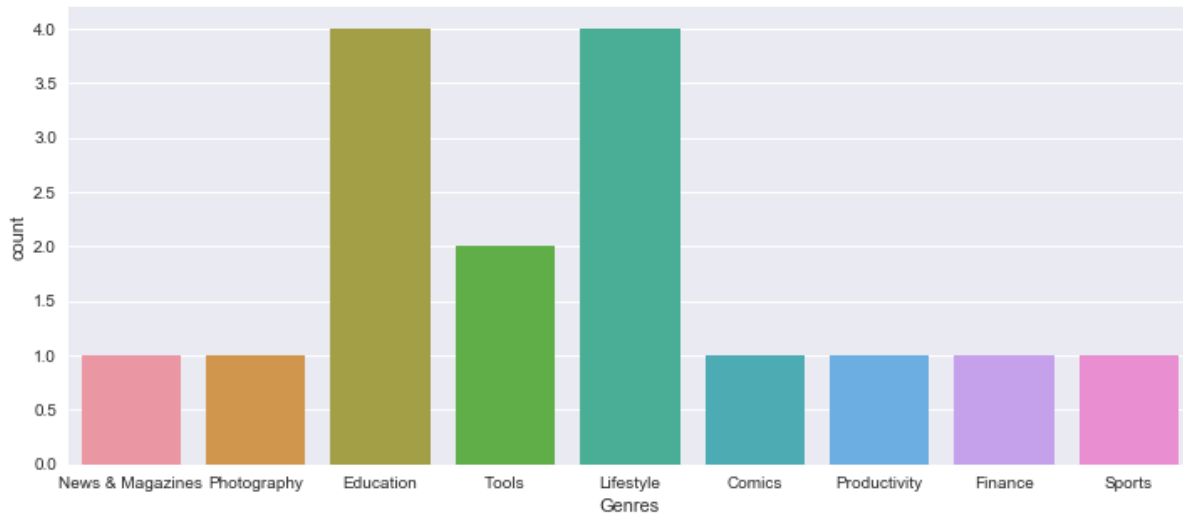
So let's filter out the ones with more than 30 reviews. These filtered ones are the apps that really stand for 5.0 rating.¶

In [664]:
```
df_full = df_full[google_rating.Reviews > 30]

print("No. of Apps having 5.0 Rating with sufficient Reviews: ",df_full.
App.count())
```

```
No. of Apps having 5.0 Rating with sufficient Reviews:  16

/Users/ajesh_mahto/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:1: UserWarning: Boolean Series key will be reindexed to mat
ch DataFrame index.
  """Entry point for launching an IPython kernel.
```

In [666]:
```
plt.figure(figsize=(12,5))
sns.countplot(df_full.Genres)
```

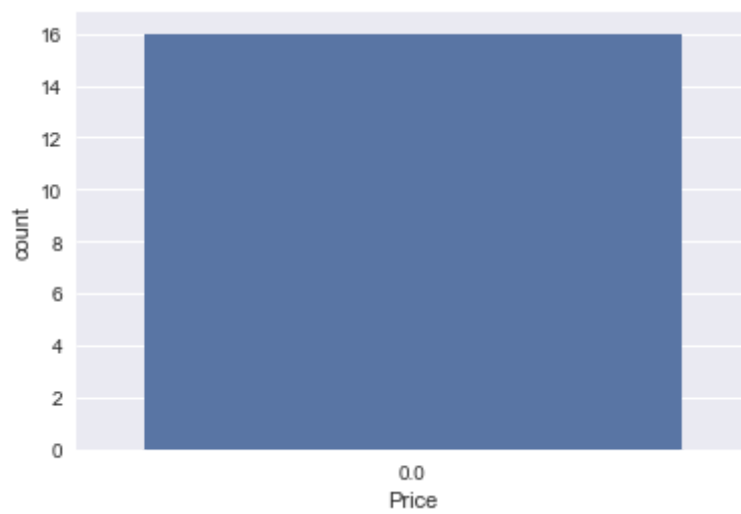
Out[666]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2929e690>



Insight : Apps related to Education, LifeStyle and Tools seem to fetch full Ratings with sufficient number of reviews.¶

In [667]:
```
sns.countplot(df_full.Price)
```

Out[667]: <matplotlib.axes._subplots.AxesSubplot at 0x1a292646d0>



Insight : All the Apps with 5.0 ratings are Free to install.

In [ ]: