1.Import all required libararies

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

2.Import the dataset

```
#import dataset
data = pd.read_csv('/content/Google Apps data.csv')
```

3.We will first check the head of the dataset i.e first five rows

```
data.head()
```

	Unnamed: 0	Unnamed: 0.1	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	La Updat
0	0	0	Photo Editor & Candy Camera & Grid & ScrapBook	Art And Design	4.1	159	19.0	10000	Free	0.0	Others	Janu 7, 2(
1	1	1	Coloring book moana	Art And Design	3.9	967	14.0	500000	Free	0.0	Others	Janu 2(
2	2	5	U Launcher Lite – FREE Live Cool Themes, Hide ...	Art And Design	4.7	87510	8.7	5000000	Free	0.0	Others	Aug 1, 2(
3	3	6	Sketch - Draw & Paint	Art And Design	4.5	215644	25.0	50000000	Free	0.0	Teen	June 2(

4.Lets check no.of rows and columns in the dataset

```
data.shape
```

```
(8276, 15)
```

5.Lets check the information datatypes and is null values in the columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8276 entries, 0 to 8275
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Unnamed: 0          8276 non-null   int64  
 1   Unnamed: 0.1        8276 non-null   int64  
 2   App                 8276 non-null   object 
 3   Category            8276 non-null   object 
 4   Rating              8276 non-null   float64
 5   Reviews             8276 non-null   int64  
 6   Size                8276 non-null   float64
 7   Installs            8276 non-null   int64  
 8   Type                8276 non-null   object 
 9   Price               8276 non-null   float64
 10  Content Rating      7915 non-null   object 
 11  Last Updated        8276 non-null   object 
 12  Current Ver         8276 non-null   object 
 13  Minimum Android Ver 8276 non-null   object 
 14  Genres              8276 non-null   object 
dtypes: float64(3), int64(4), object(8)
memory usage: 970.0+ KB
```

6.Checking the data types of the columns

```
#checking datatypes

data.dtypes
```

```
Unnamed: 0              int64
Unnamed: 0.1            int64
App                    object
Category               object
Rating                float64
Reviews                 int64
Size                  float64
Installs                int64
Type                   object
Price                 float64
Content Rating         object
Last Updated           object
Current Ver            object
Minimum Android Ver    object
Genres                 object
dtype: object
```

7.Check the names of the columns in the dataset

```
data.columns
```

```
Index(['Unnamed: 0', 'Unnamed: 0.1', 'App', 'Category', 'Rating', 'Reviews',
       'Size', 'Installs', 'Type', 'Price', 'Content Rating', 'Last Updated',
       'Current Ver', 'Minimum Android Ver', 'Genres'],
      dtype='object')
```

8. Cleaning the dataset

Lets check if any null values are there in the dataset

```
#checking for null values

data.isnull().sum()
```

```
Unnamed: 0              0
Unnamed: 0.1            0
App                     0
Category                0
Rating                  0
Reviews                 0
Size                    0
Installs                0
Type                    0
Price                   0
Content Rating        361
Last Updated            0
Current Ver             0
Minimum Android Ver     0
Genres                  0
dtype: int64
```

As we can see only Content Rating has null values. We will elimanate those null values

```
data.dropna(how='any',inplace=True)
```

```
data.isnull().sum()
```

```
Unnamed: 0            0
Unnamed: 0.1          0
App                   0
Category              0
Rating                0
Reviews               0
Size                  0
Installs              0
Type                  0
Price                 0
Content Rating        0
Last Updated          0
Current Ver           0
Minimum Android Ver   0
Genres                0
dtype: int64
```

All null values are eliminated

9.Check the unique values in the price column

> Indented block

```
data['Price'].value_counts()
```

```
0.00      7326
2.99       108
0.99       100
4.99        66
1.99        58
          ...
1.59         1
6.49         1
1.29         1
379.99       1
1.20         1
Name: Price, Length: 72, dtype: int64
```

10. After eliminating the null values lets check the size of the dataset

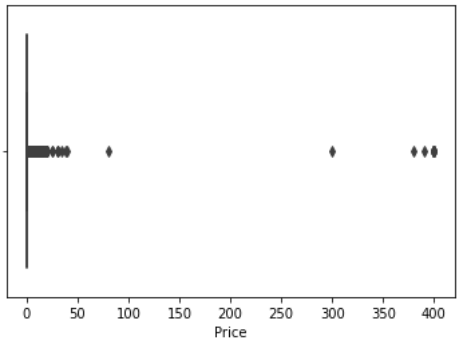
```
data.shape
```

```
(7915, 15)
```

11. Exploratory Data Analysis (EDA) Now lets check outliers present in the dataset

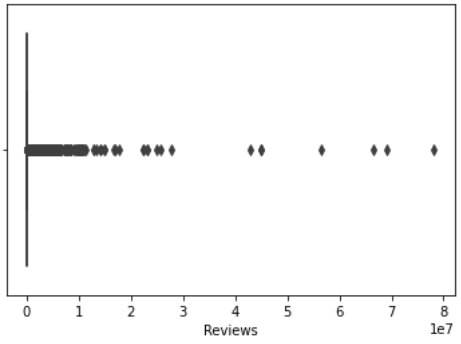
```
#Lets check for outliers

sns.boxplot(data.Price)
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
```



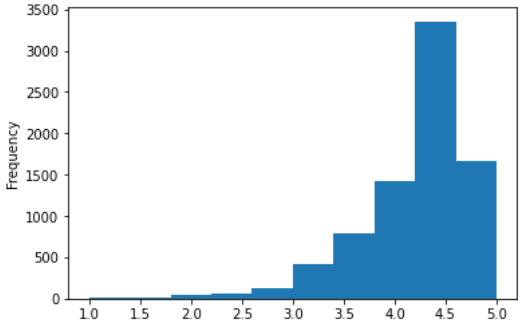

```
sns.boxplot(data.Reviews)
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
```

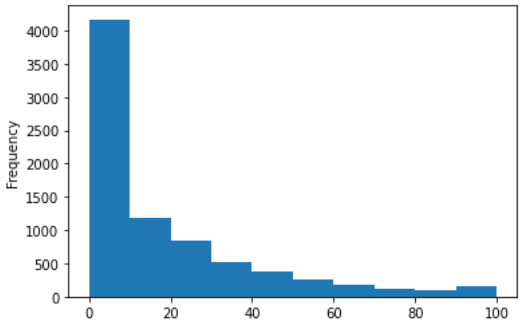



```
#checking distribution and skewness
data.Rating.plot.hist()
```

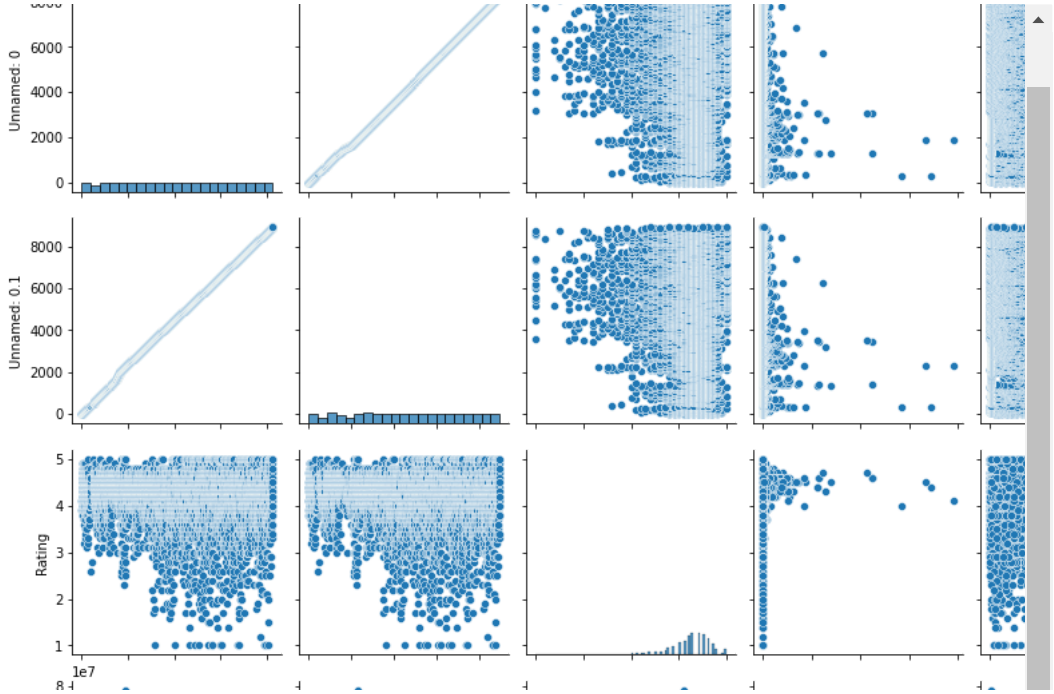
```
plt.show()
```




```
#Size PLOT

data.Size.plot.hist()
plt.show()
```



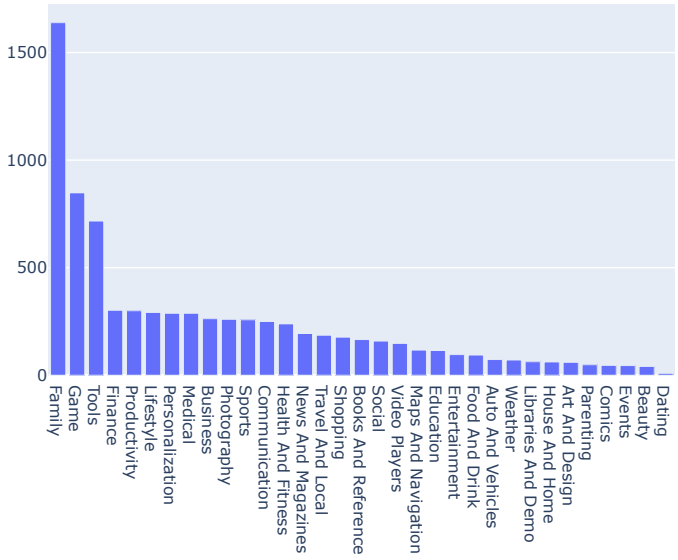

```
#Pair Plot
sns.pairplot(data=data)
```



12.Now well check category vise data of the apps

```
import plotly
import plotly.graph_objs as go
# Print the total number of unique categories
num_categories = data["Category"].nunique()
print('Number of categories = ', num_categories)

# Count the number of apps in each 'Category'. Sort in descending order depending on number of apps in each category
num_apps_in_category = data["Category"].value_counts().sort_values(ascending=False)
data1 = [go.Bar(
        x = num_apps_in_category.index, # index = category name
        y = num_apps_in_category.values, # value = count
)]

plotly.offline.iplot(data1)
```

```
Number of categories =  33
```

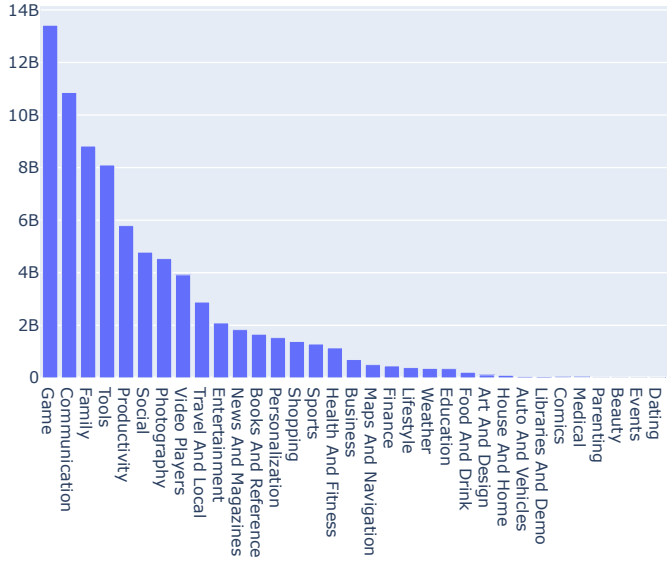


Obeservations:

-33 unique app categories in the dataset or market -Top 5: Family (1832), Game, Tools, Business, Medical -Last 5: Beauty, Parenting, Comic, Art and Design, Events

13.Now we will check category wise apps installed

```
#lets check number of app install by category
num_apps_installed=data.groupby("Category").agg({"Installs":"sum" }).sort_values(by="Installs",ascending=False)

data1 = [go.Bar(
        x = num_apps_installed.index, # index = category name
        y = num_apps_installed["Installs"], # value = count
)]

plotly.offline.iplot(data1)
num_apps_installed["Installs"].head(10)
```



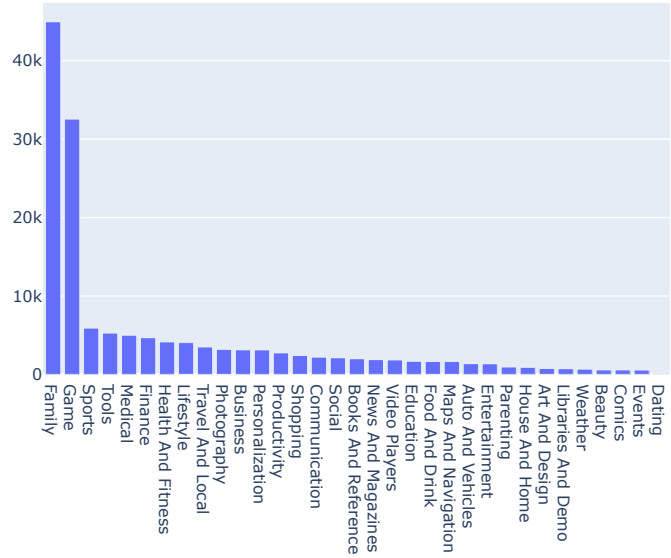

```
Category
Game                13417140507
Communication       10859131530
Family               8821100780
Tools                8100169500
Productivity         5793069180
Social               4783221475
Photography          4538143130
Video Players        3921397200
Travel And Local     2884859300
Entertainment        2090560000
Name: Installs, dtype: int64
```

Observation:

As we can see top 5 Installed apps and there numbers Top 3: 1.Game-13417140507 2.Communication-10859131530 3.Family-8821100780

14.lets check size of app install by category

```
#lets check size of app install by category
num_apps_installed=data.groupby("Category").agg({"Size":"sum" }).sort_values(by="Size",ascending=False)

data1 = [go.Bar(
        x = num_apps_installed.index, # index = category name
        y = num_apps_installed["Size"], # value = count
)]

plotly.offline.iplot(data1)
num_apps_installed["Size"].head(10)
```




```
Category
Family                44968.42852
Game                  32559.61601
```

15.Moving towards rating

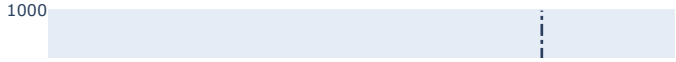
```
Medical                4987.68673
```

```
#moving towards rating

#lets check average rating of all apps

print('Average rating of all applications is:',data['Rating'].mean())
```

```
Average rating of all applications is: 4.177485786481364
```

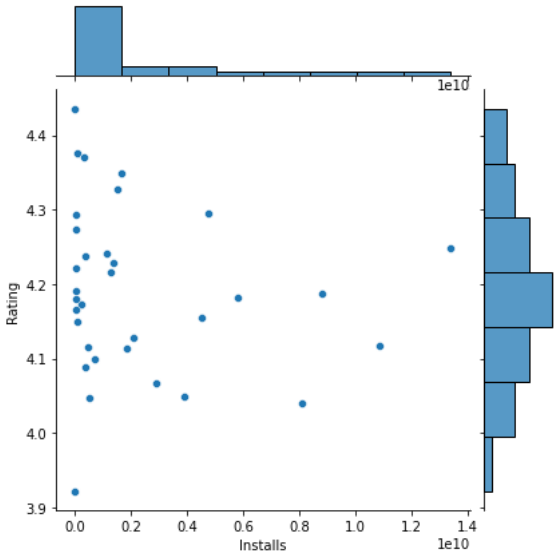
16.Lets see average ratings

```
#lets plot average rating

data1·=·[go.Histogram(x=data['Rating'])]

#·Vertical·dashed·line·to·indicate·the·average·app·rating
layout·=·{'shapes':·[{
              'type' :'line',
              'x0': data['Rating'].mean(),
              'y0': 0,
              'x1': data['Rating'].mean(),
              'y1': 1000,
              'line': { 'dash': 'dashdot'}
          }]
          }
plotly.offline.iplot({"data":data1,"layout":layout})
```

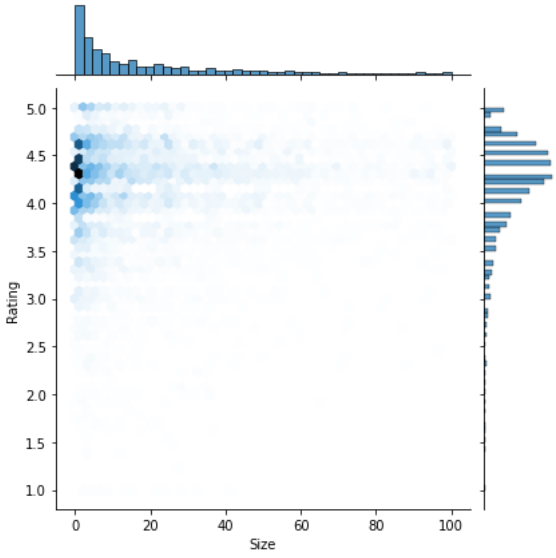



```
# Check the realtion bewteen Installs and Rating. High installments doesn´t necessarily mean high rates.
category_info=data.groupby("Category").agg({"Rating":"mean","Installs":"sum" })
sns.jointplot(x="Installs",y="Rating",data=category_info)
```

```
<seaborn.axisgrid.JointGrid at 0x7f0495126160>
```



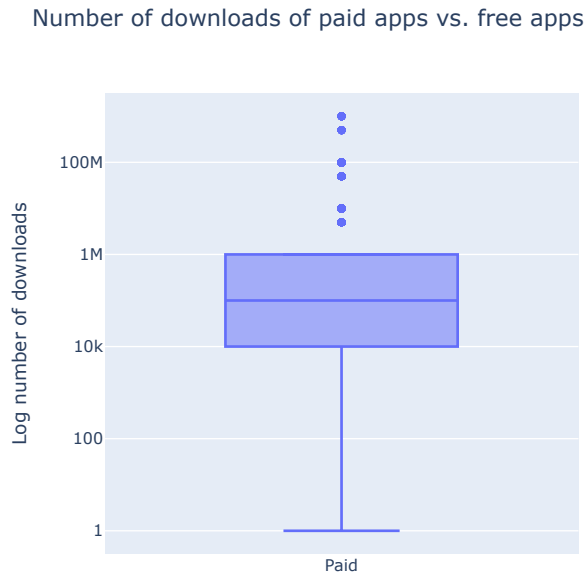

```
#Size and price of an app

plt1 = sns.jointplot(x = data["Size"], y = data["Rating"], kind = 'hex')
```




```
## Examine the price trend by plotting Price vs Category
plt.figure(figsize=(15,10))
ax = sns.stripplot(x = data["Price"], y = data["Category"], jitter=True, linewidth=1)
ax.set_title('App pricing trend across categories')
```

```
Text(0.5, 1.0, 'App pricing trend across categories')
```


```
#Popularity of paid apps vs free apps

trace0 = go.Box(
    # Data for paid apps
    y = data[data['Type'] == "Paid"]['Installs'],
    name = 'Paid'
)

trace1 = go.Box(
    # Data for free apps
    y = data[data['Type'] == "Free"]['Installs'],
    name = 'Free'
)

layout = go.Layout(
    title = "Number of downloads of paid apps vs. free apps",
    yaxis = dict(title = "Log number of downloads",
                 type = 'log',
                 autorange = True)
)

# Add trace0 and trace1 to a list for plotting
data1 = [trace0, trace1]
plotly.offline.iplot({'data': data1, 'layout': layout})
```



✓ 0s completed at 4:05 PM