Mini Project Data Science: Analysis of Google PlayStore Data

In [1]:

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

In [2]:

```
data = pd.read_csv("./googleplaystore.csv")
data.dropna(inplace=True)#removes all the rows with atleast one null value,inplace is lik
e conformation
```

In [3]:

```
data.shape
```

Out[3]:

```
(9360, 13)
```

In [4]:

```
data.head()
```

Out[4]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Cu
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018	
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	June 8, 2018	V de
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	

In [5]:

```
data.describe()
```

Out[5]:

	Rating
count	9360.000000
mean	4.191838
std	0.515263
min	1.000000
25%	4.000000
50%	4.300000
75%	4.500000
max	5.000000

In [6]:

```
print("cleaning review column:-")
```

```
cleaning review column:-
```

In [7]:

```
"""
data of Reviews has string M in it i stands for million
we have to remove the string "M" and "," from values and
miltiply with 1000000 if the values has 'M' in it
finaly convert to int

this method can clean the Reviewa column
"""
def filter(per):
    if "M" in str(per) and "," in str(per):
        per = str(per).replace("M","")
        per = per.replace(",","")
        return int(per)*1000000
    elif "M" in str(per):
        per = int(str(per).replace("M",""))
        return per*1000000
    elif "," in str(per):
        per = str(per).replace(",","")
        return int(per)

    else:
        return int(per)
```

In [8]:

```
data["Reviews"] =data["Reviews"].apply(filter) # all the values of column 'Reviews' are p
assed to filter method
```

In [9]:

```
print("cleaning size column:-")
```

```
cleaning size column:-
```

In [10]:

```
def filter1(per):
    per = str(per)
    if "M" in per:
        per = per.replace("M","")
        return float(per)
    elif per == "Varies with device":
        return np.NaN
    elif "k" in per:
        return float(per.replace("k",""))/1000
    else:
```

```
        return float(per)
```

In [11]:

```
data["Size"]=data["Size"].apply(filter1) #used to apply filter1 function
```

In [12]:

```
print("cleaning install column:-")
```

```
cleaning install column:-
```

In [13]:

```
"""
thid function is used to clean instals column
it remones i]the string "+" and ","
and returns as intiger
"""
def filter2(per):
    per = str(per)
    if "+" in per:
        per = per.replace("+","")
    if "," in per:
        per = per.replace(",","")

    return int(per)
```

In [14]:

```
data["Installs"]=data["Installs"].apply(filter2)# used to apply filter2 function"
```

In [15]:

```
#cleaning price column
"""
used to remove the string "$"
and convert thr price to rupies as floats
"""
def filter3(per):
    per = str(per)
    if "$" in per:
        per=per.split("$")[1]
    return (float(per)*69.44)
```

In [16]:

```
data["Price"]=data["Price"].apply(filter3)# used to apply filter 3 function
```

In [17]:

```
print("DATA VISULIZATION:")
```

```
DATA VISULIZATION:
```

In [18]:

```
import plotly
print(plotly.__version__)
%matplotlib inline
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
```

```
4.11.0
```

In [19]:

```
sns.pairplot(pd.DataFrame(list(zip(data["Rating"],data["Size"], np.log(data["Installs"]),
np.log10(data["Reviews"]),data["Type"], data["Price"])),
```

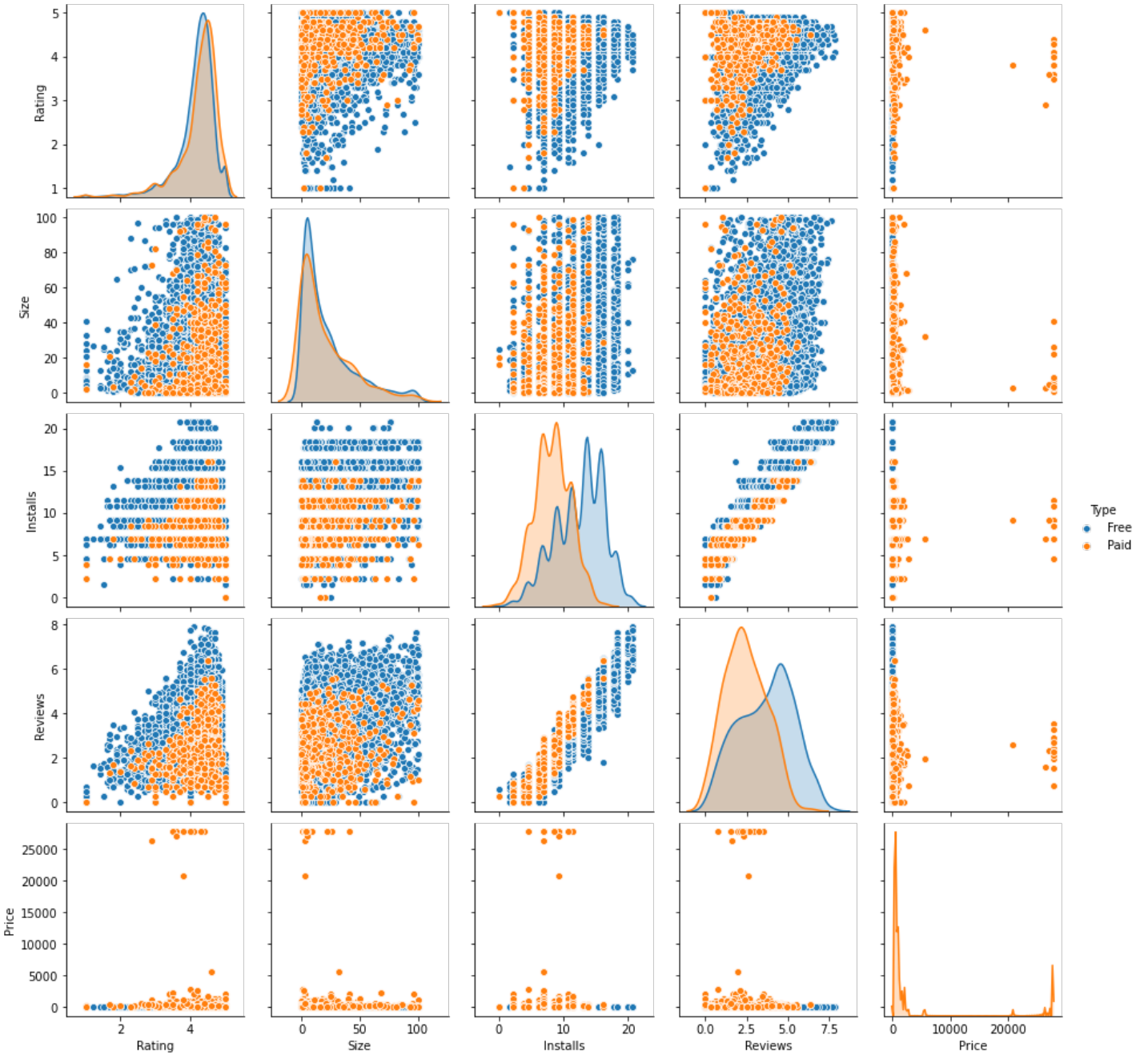
```
                    columns=['Rating','Size', 'Installs', 'Reviews', 'Type', 'Price'
]), hue='Type')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:283: UserWarning:

Data must have variance to compute a kernel density estimate.
```

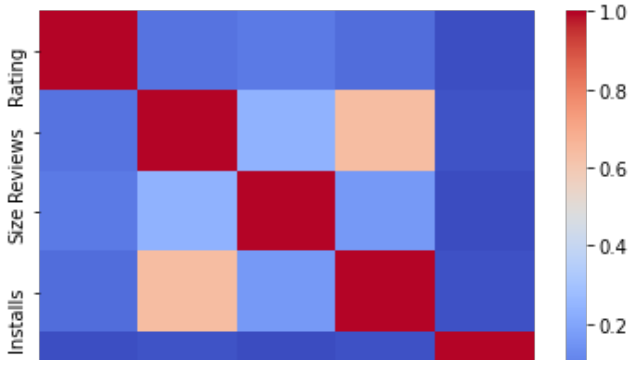
Out[19]:

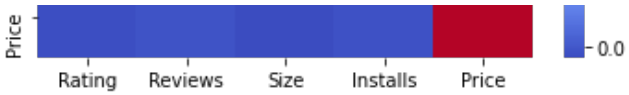
```
<seaborn.axisgrid.PairGrid at 0x24762b73f70>
```

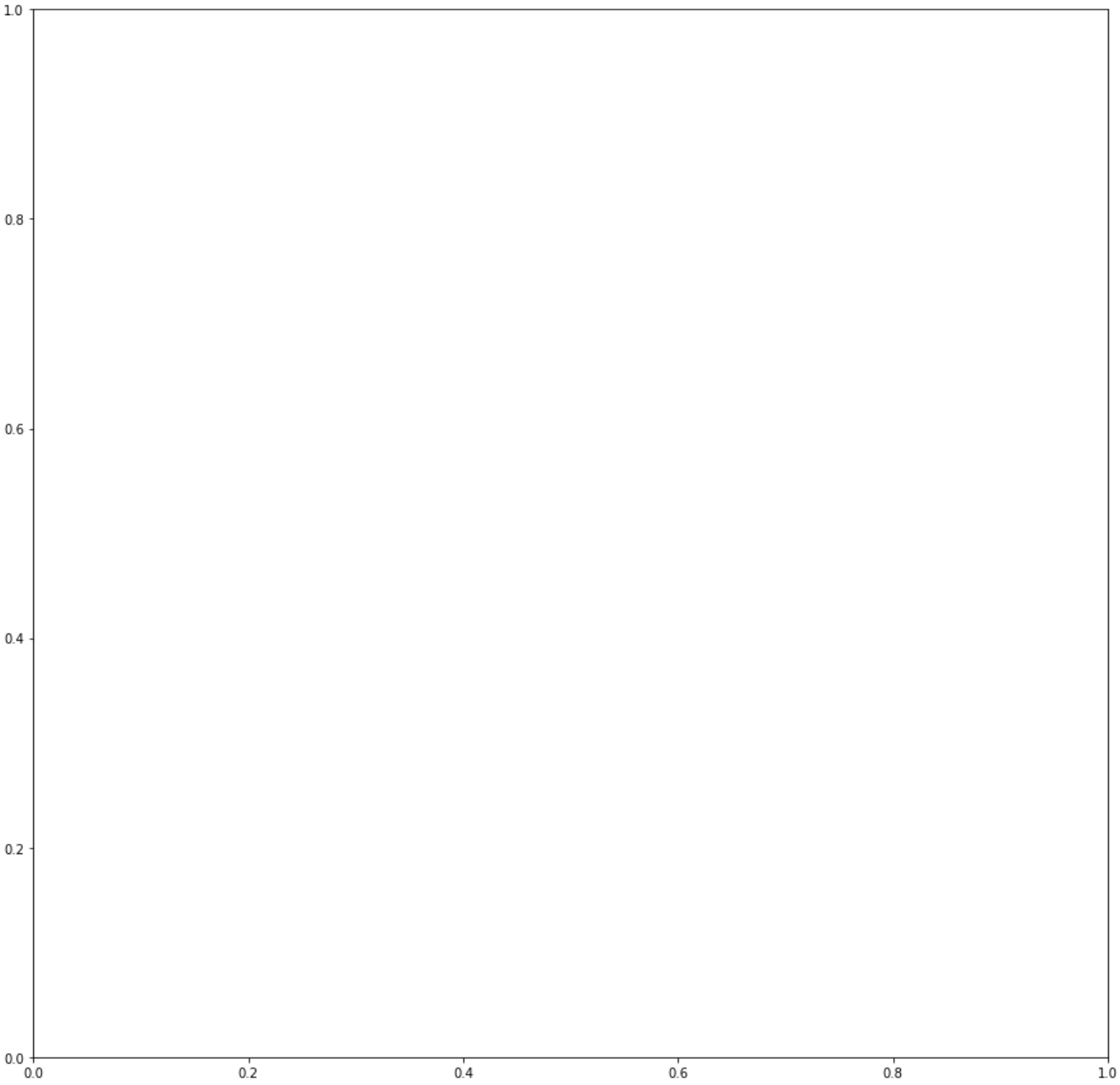


In [20]:

```
sns.heatmap(data.corr(),cmap='coolwarm')
fig, ax = plt.subplots(figsize=(15,15))
```







In [21]:

```
number_of_apps_in_category = data['Category'].value_counts().sort_values(ascending=True)

df = [go.Pie(labels = number_of_apps_in_category.index,values = number_of_apps_in_catego
ry.values,hoverinfo = 'label+value')]

plotly.offline.iplot(df, filename='active_category')
```

In [22]:

```
df = [go.Histogram(
        x = data.Rating,
        xbins = {'start': 1, 'size': 0.1, 'end' :5}
)]

print('Average app rating = ', np.mean(data['Rating']))
plotly.offline.iplot(df, filename='overall_rating_distribution')
```

```
Average app rating =  4.191837606837612
```

In [23]:

```
#print('Junk apps priced above  12800')
data[['Category', 'App',"Price"]][data.Price > 200*64]
```

Out[23]:

	Category	App	Price

4197	[illegible]	most expensive app (H)	27775.3056
4362	LIFESTYLE	💎 I'm rich	27775.3056
4367	LIFESTYLE	I'm Rich - Trump Edition	27776.0000
5351	LIFESTYLE	I am rich	27775.3056
5354	FAMILY	I am Rich Plus	27775.3056
5355	LIFESTYLE	I am rich VIP	20831.3056
5356	FINANCE	I Am Rich Premium	27775.3056
5357	LIFESTYLE	I am extremely Rich	26386.5056
5358	FINANCE	I am Rich!	27775.3056
5359	FINANCE	I am rich(premium)	27775.3056
5362	FAMILY	I Am Rich Pro	27775.3056
5364	FINANCE	I am rich (Most expensive app)	27775.3056
5366	FAMILY	I Am Rich	27080.9056
5369	FINANCE	I am Rich	27775.3056
5373	FINANCE	I AM RICH PRO PLUS	27775.3056

In [24]:

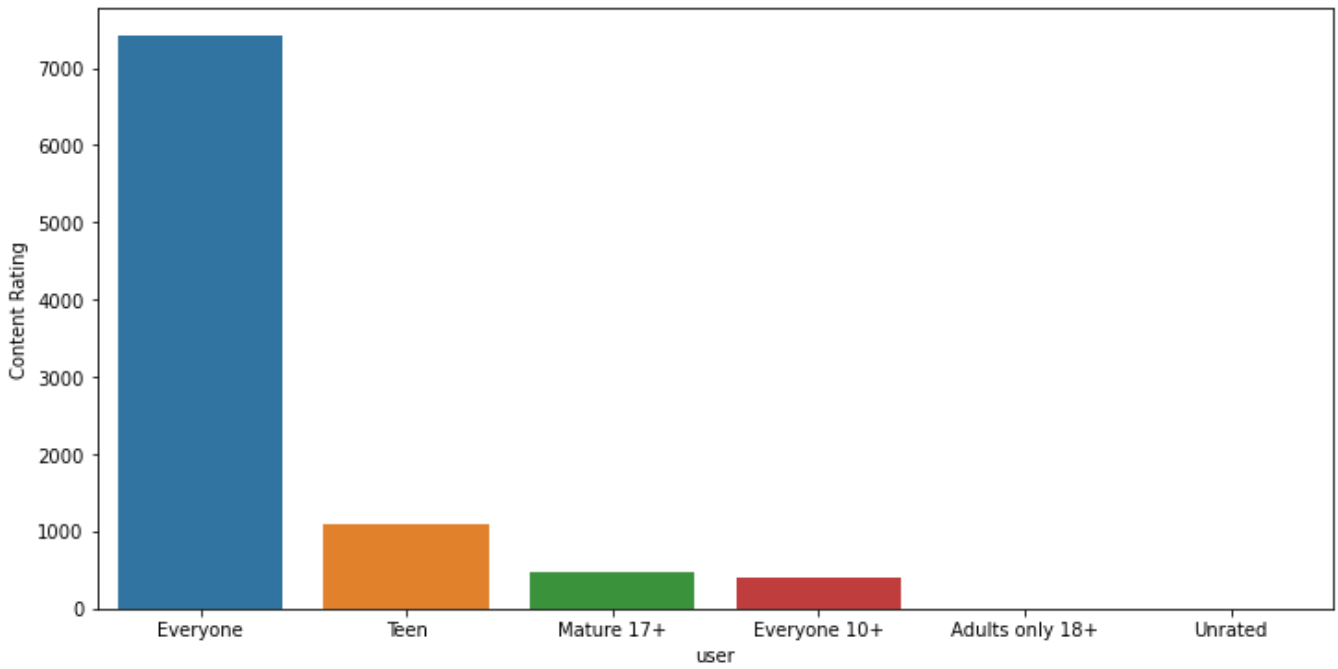
```
temp=pd.DataFrame(data["Content Rating"].value_counts()).reset_index()
temp.columns=['user', 'Content Rating']
```

In [25]:

```
plt.figure(figsize=(12,6))
sns.barplot(data=temp,x="user",y="Content Rating")
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2476290f8b0>
```

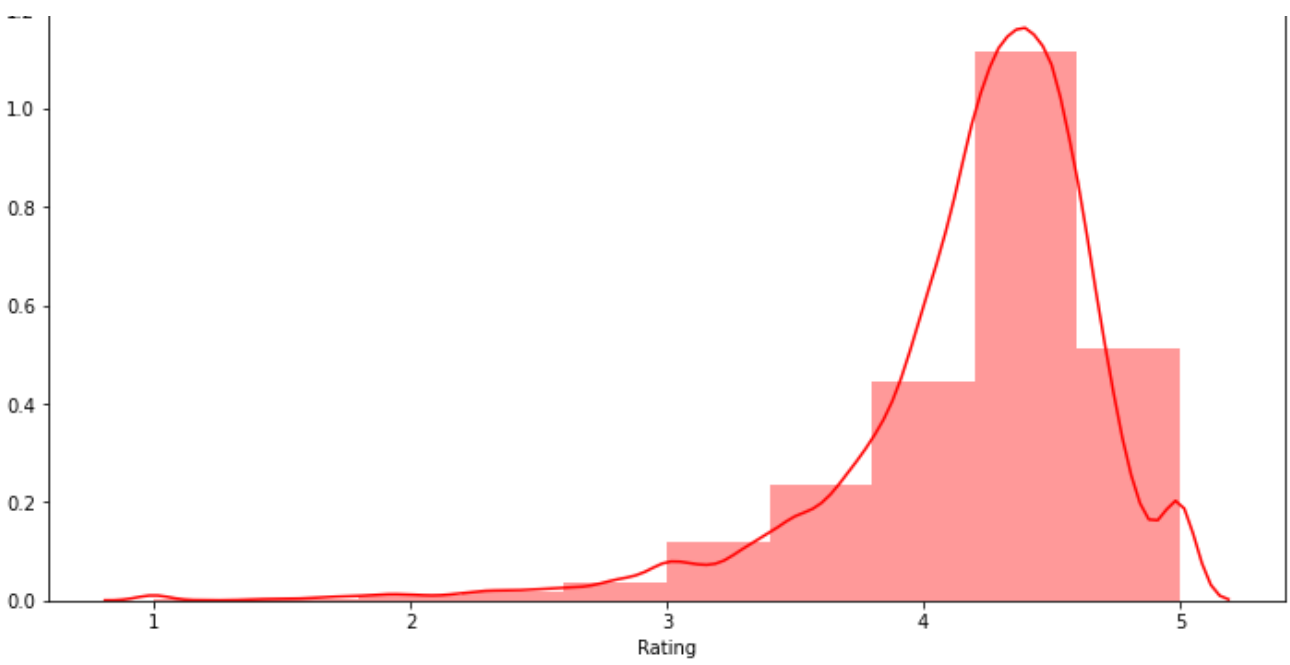


In [26]:

```
#most reviewed app rating
plt.figure(figsize=(12,6))
sns.distplot(data["Rating"],bins=10,color="red")
```

Out[26]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x247659379a0>
```

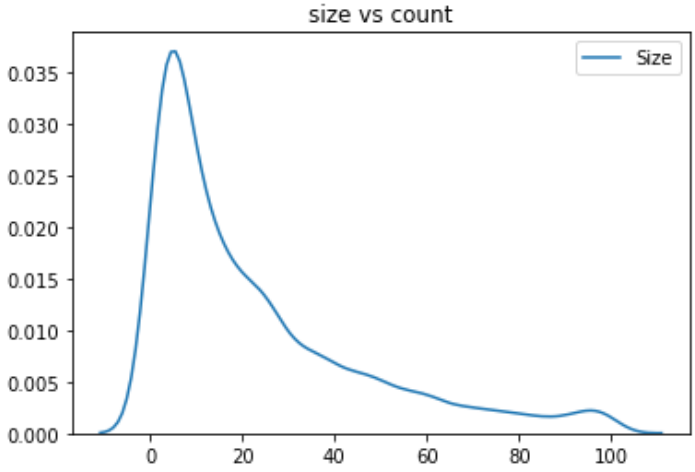


In [27]:

```
sns.kdeplot(data=data["Size"])
plt.title("size vs count")
plt.xlabel("")
```

Out[27]:

```
Text(0.5, 0, '')
```



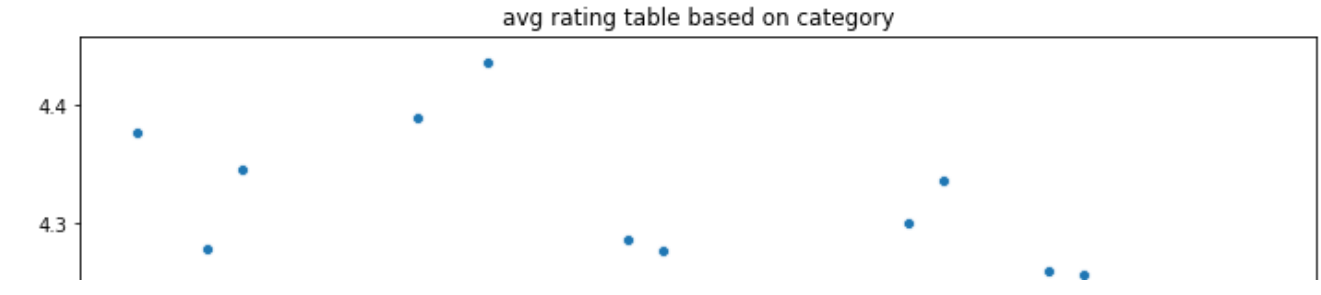
In [28]:

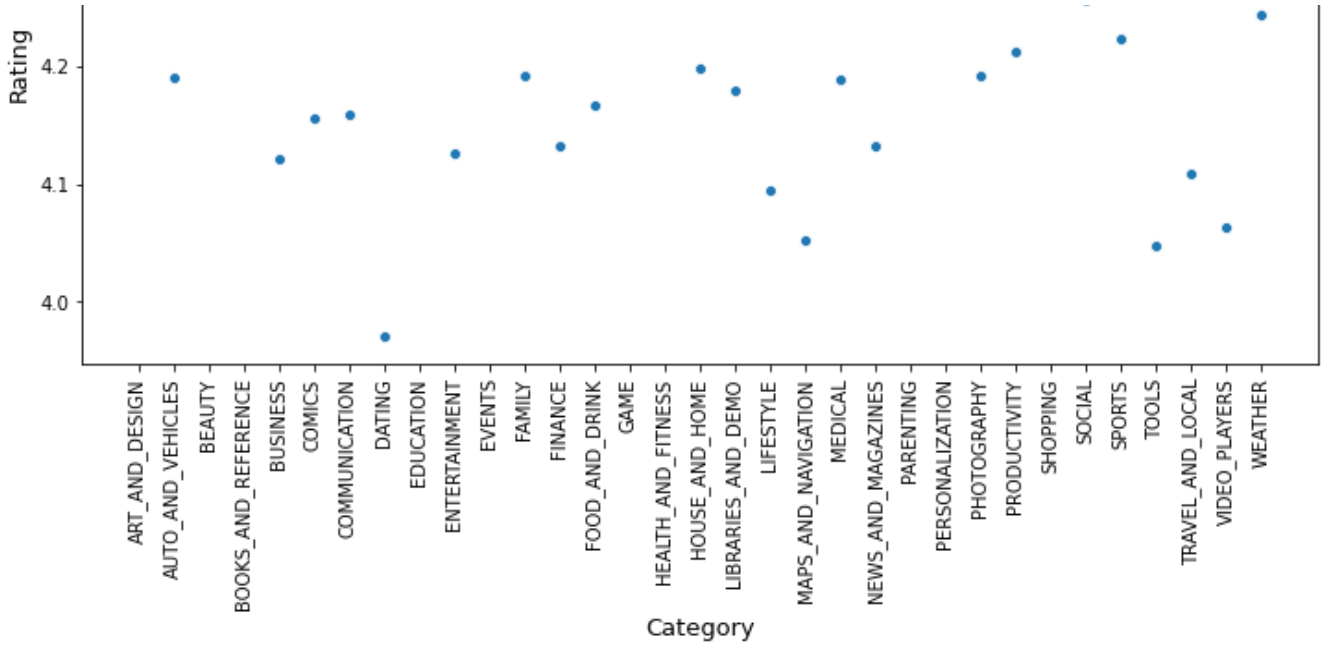
```
plt.figure(figsize=(12,6))

sns.scatterplot(x = data.groupby('Category')['Rating'].mean().index, y = data.groupby('C
ategory')['Rating'].mean().values)
plt.xlabel('Category', fontsize=13)
plt.ylabel('Rating', fontsize=13)
plt.xticks(rotation=90)
plt.title("avg rating table based on category")
```

Out[28]:

```
Text(0.5, 1.0, 'avg rating table based on category')
```





In [29]:

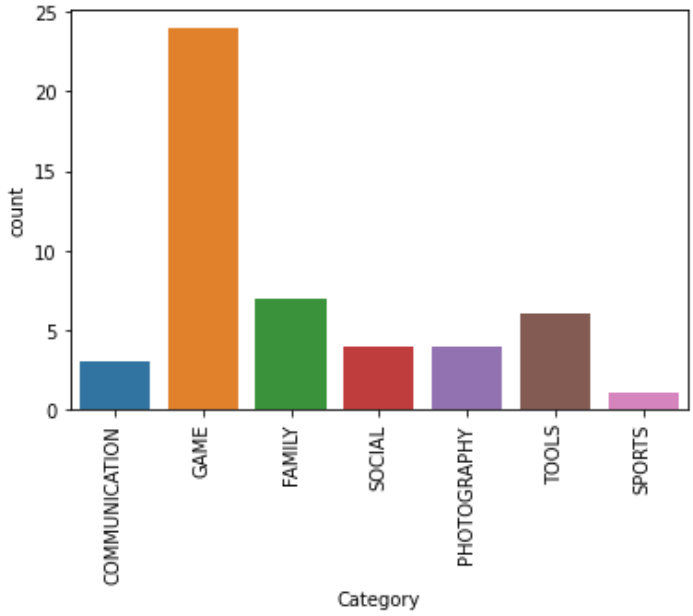
```
most_popular_apps = data[(data["Reviews"]>10000000) ][ (data["Rating"]>=4.5)]
sns.countplot(most_popular_apps["Category"])
plt.xticks(rotation=90)
```

```
<ipython-input-29-88b3bb6fe8d9>:1: UserWarning:

Boolean Series key will be reindexed to match DataFrame index.
```

Out[29]:

```
(array([0, 1, 2, 3, 4, 5, 6]), <a list of 7 Text major ticklabel objects>)
```

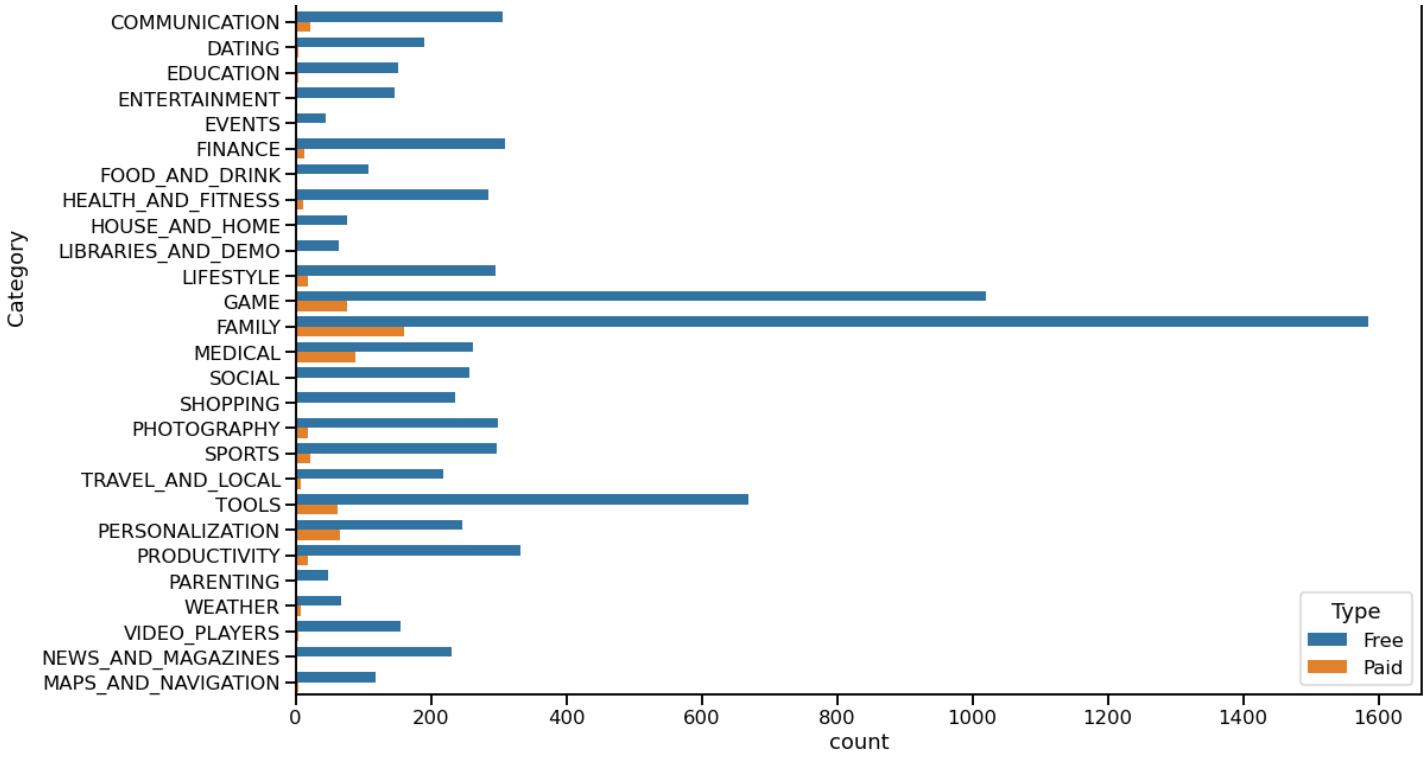


In [30]:

```
sns.set_context('talk',font_scale=1)
plt.figure(figsize=(17,13))
sns.countplot(data=data,y="Category",hue="Type")
```

Out[30]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24765a8ba30>
```

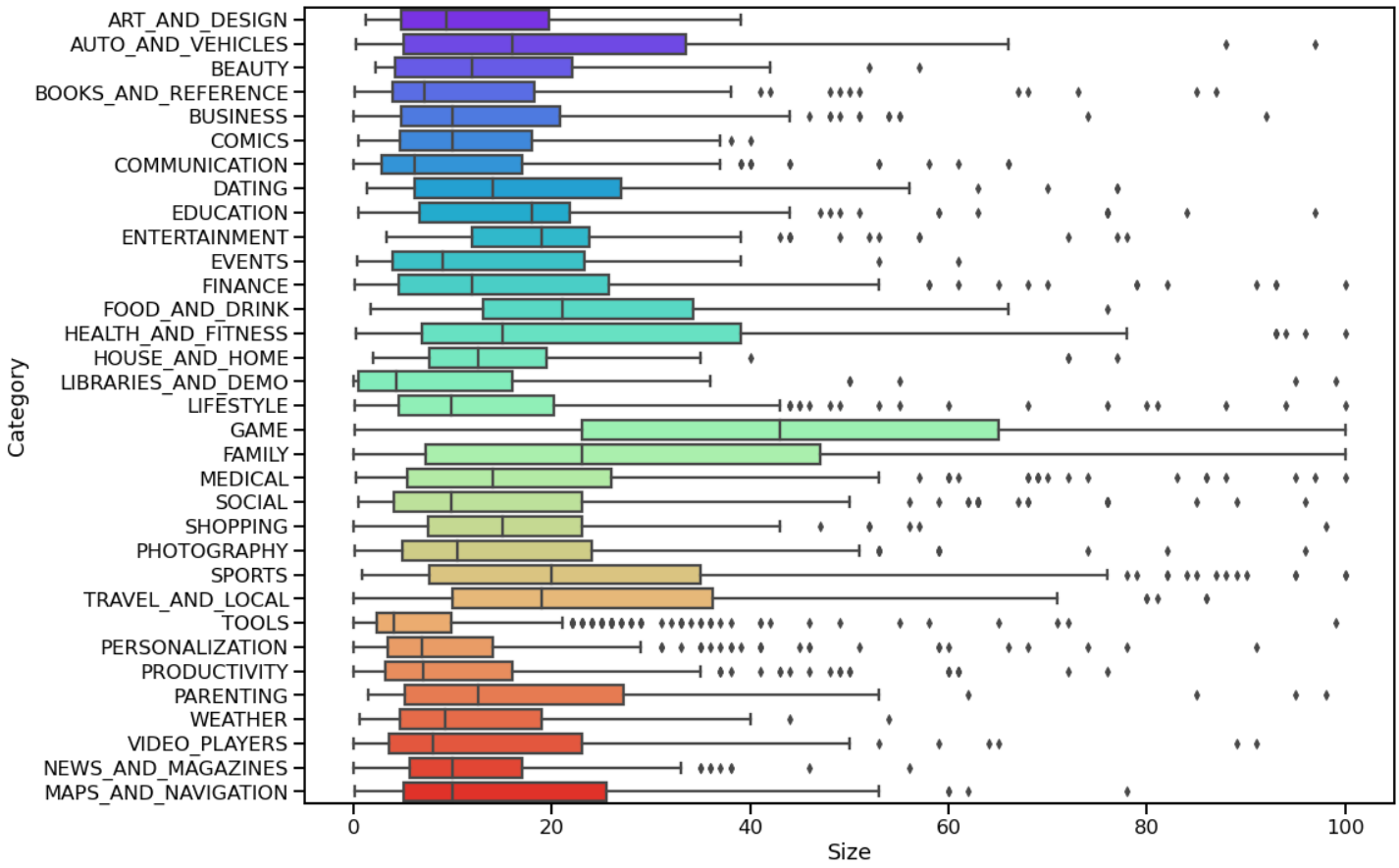


In [31]:

```
plt.figure(figsize=(16,12))
sns.boxplot(data=data,x="Size",y="Category",palette='rainbow')
```

Out[31]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24765a7d8e0>
```

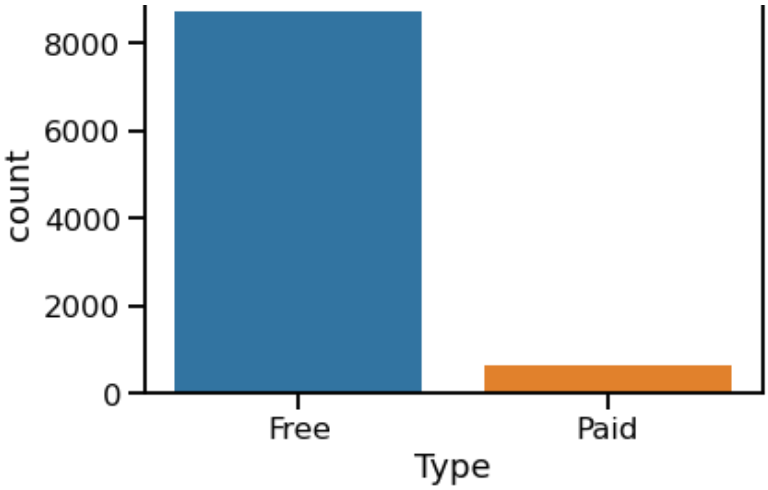


In [32]:

```
sns.countplot(x=data["Type"])#either paid or free
```

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x247667498b0>
```

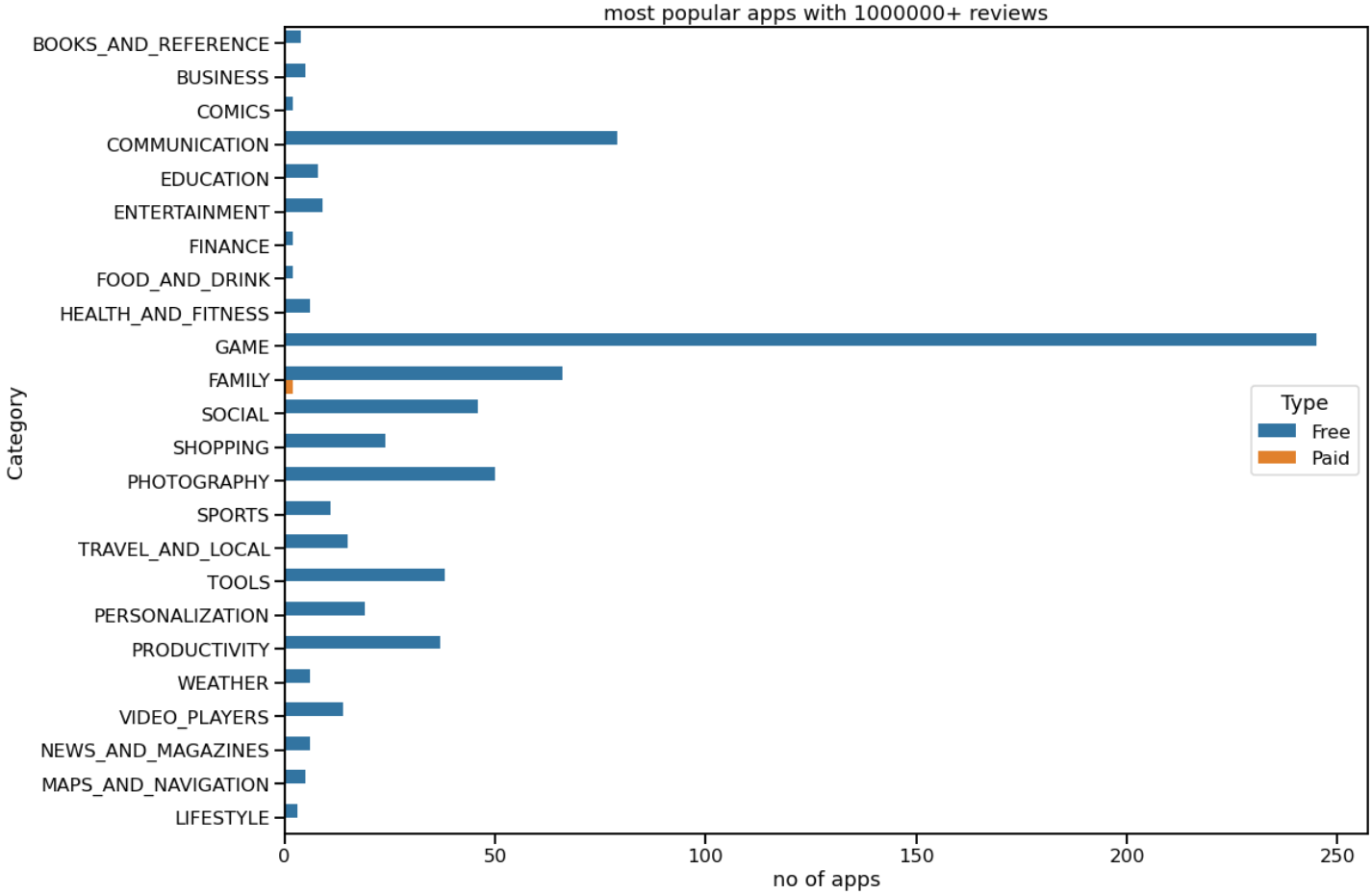


In [33]:

```
plt.figure(figsize=(17,13))
sns.countplot(data=data[data["Reviews"]>1000000],y="Category",hue="Type")
plt.title("most popular apps with 1000000+ reviews")
plt.xlabel("no of apps")
```

Out[33]:

```
Text(0.5, 0, 'no of apps')
```

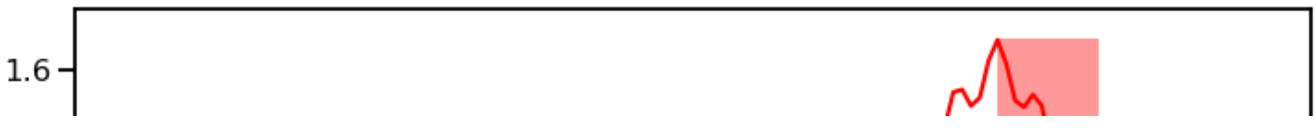


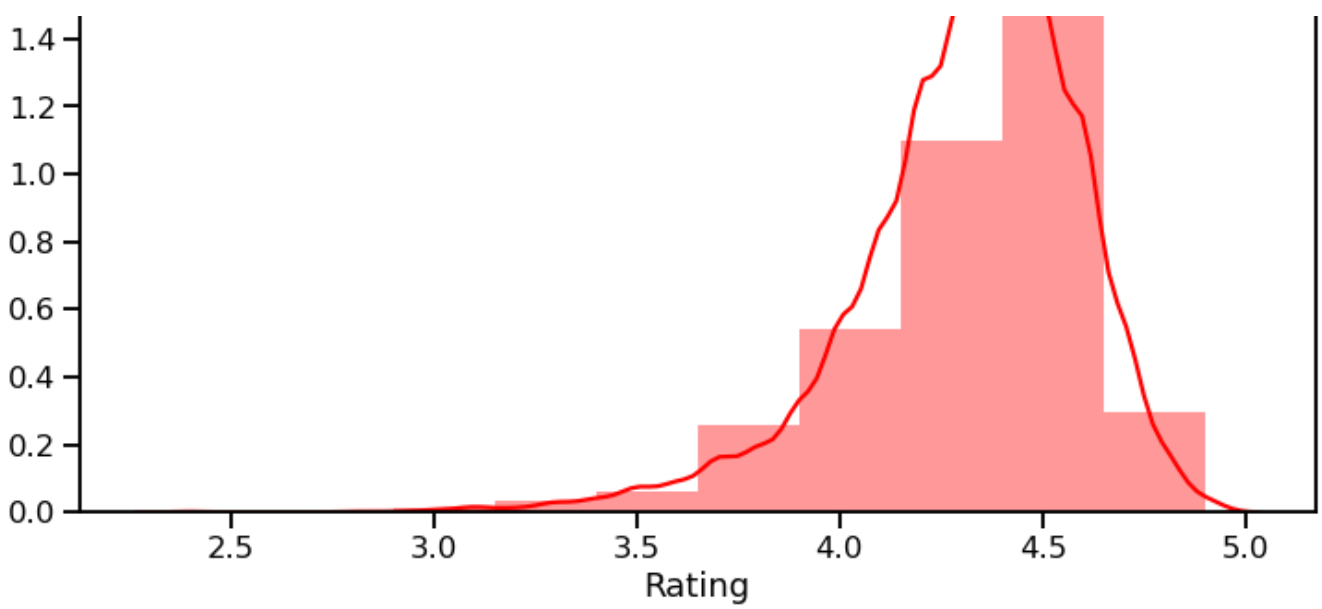
In [34]:

```
#most reviewed app rating
plt.figure(figsize=(12,6))
sns.distplot(data[data["Reviews"]>10000]["Rating"],bins=10,color="red")
```

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x247668f4520>
```



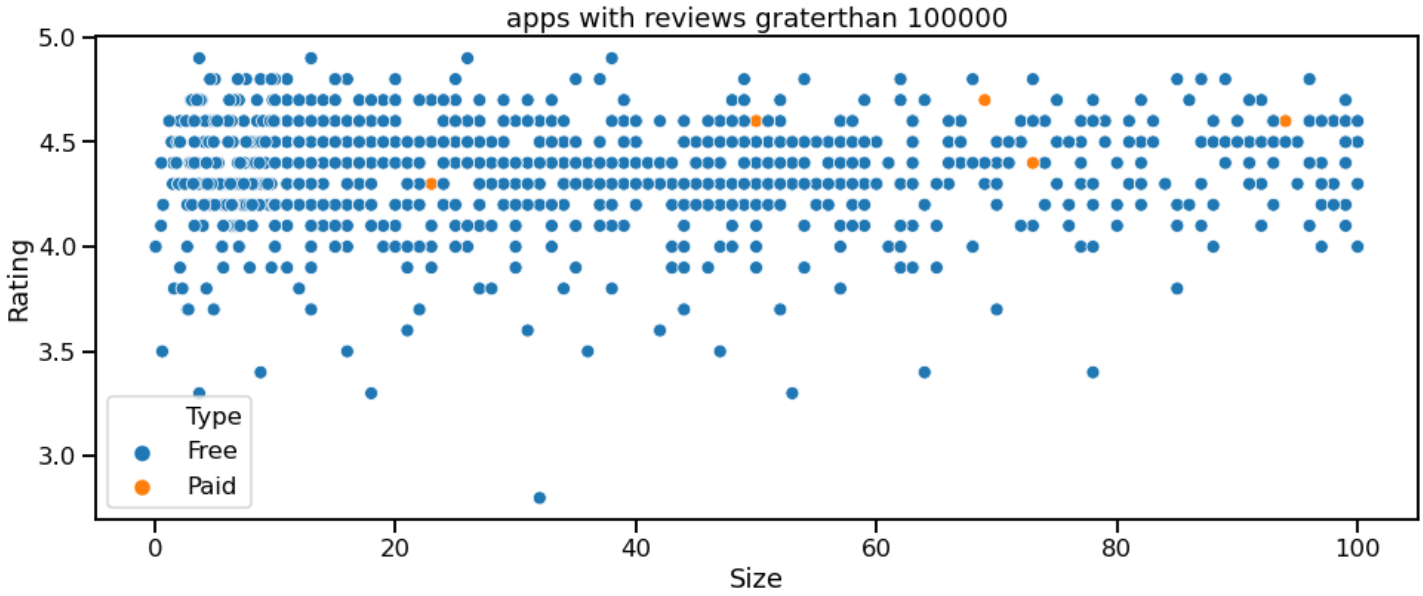


In [35]:

```
plt.figure(figsize=(16,6))
sns.scatterplot(data=data[data["Reviews"]>100000],x="Size",y="Rating",hue="Type")
plt.title("apps with reviews graterthan 100000")
```

Out[35]:

```
Text(0.5, 1.0, 'apps with reviews graterthan 100000')
```



In [36]:

```
x=np.log(data["Installs"])
y=np.log(data["Reviews"])
```

In [37]:

```
popular_apps = data[(data["Installs"]>10000000) & (data["Rating"]>=4.7)]
#the most popular paid apps with decent reviews and ratings
pd.DataFrame(popular_apps[popular_apps["Type"]=="Free"][["App"]])
```

Out[37]:

	App
699	**Duolingo: Learn Languages Free**
784	**Duolingo: Learn Languages Free**
799	**Duolingo: Learn Languages Free**
826	**Duolingo: Learn Languages Free**

	App
1360	Period Tracker - Period Calendar Ovulation App
1677	Bubble Witch 3 Saga
1712	Toy Blast
1720	Bowmasters
1762	Bowmasters
1763	Piano Tiles 2™
1877	Toy Blast
1907	Bowmasters
1922	Bowmasters
1939	CATS: Crash Arena Turbo Stars
1999	Bowmasters
2056	Duolingo: Learn Languages Free
2216	Duolingo: Learn Languages Free
2674	The birth
2684	Mercado Libre: Find your favorite brands
2931	Video Editor Music,Cut,No Crop
3112	Booking.com Travel Deals
3156	Booking.com Travel Deals
3211	Booking.com Travel Deals
3298	Brightest Flashlight Free ®
3365	ZenUI Launcher
3464	Calculator - unit converter
3941	Bible
4005	Clean Master- Space Cleaner & Antivirus
4038	DU Recorder – Screen Recorder, Video Editor, Live
5719	Kaspersky Mobile Antivirus: AppLock & Web Secu...
6077	Bowmasters
7536	Security Master - Antivirus, VPN, AppLock, Boo...
8439	Duolingo: Learn Languages Free

In [38]:

```
popular_apps = data[(data["Installs"]>100000) & (data["Rating"]>4.5)]
#the most popular paid apps with decent reviews and ratings
pd.DataFrame(popular_apps[popular_apps["Type"]=="Paid"][["App","Price"]])
```

Out[38]:

	App	Price
853	Toca Life: City	277.0656
2151	Toca Life: City	277.0656
4034	Hitman Sniper	68.7456
4260	Cut the Rope GOLD	68.7456
5627	Five Nights at Freddy's 2	207.6256
5631	Five Nights at Freddy's	207.6256
6936	Hitman GO	68.7456
8449	Cameringo+ Filters Camera	207.6256
8804	DraStic DS Emulator	346.5056

	App	Price
8860	Bloons TD 5	207.6256
9678	Where's My Water?	138.1856
9785	ES File Explorer/Manager PRO	207.6256
9941	Tasker	207.6256

In [39]:

```
print("On analyzing the data, RATINGS of the app can be concluded as the most important parameter that plays an important role in depicting how better the app performs compared to the other apps in the market. It also hints on how well the company works on implementing the feedback given by the users. After all, users are the key to modern software businesses.")
```

```
On analyzing the data, RATINGS of the app can be concluded as the most important parameter that plays an important role in depicting how better the app performs compared to the other apps in the market. It also hints on how well the company works on implementing the feedback given by the users. After all, users are the key to modern software businesses.
```

In [40]:

```
"""
To predict the ratings of the App (before/after launching it on Play Store).
This is clearly a regression problem.
"""
mldata = data[["Reviews","Size","Installs","Price","Rating"]]
mldata.dropna(inplace=True)

X=mldata.iloc[:,0:-1].values
y = mldata.iloc[:,-1].values

from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(X,y)

from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(n_estimators=300)
```

```
<ipython-input-40-a825040b6fcf>:6: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [41]:

```
rfr.fit(xtrain,ytrain)
ypre = rfr.predict(xtest)

df=pd.DataFrame()

df["ytest"]=pd.Series(ytest)

df["ypre"] =pd.Series(ypre)
df.sample(10)
```

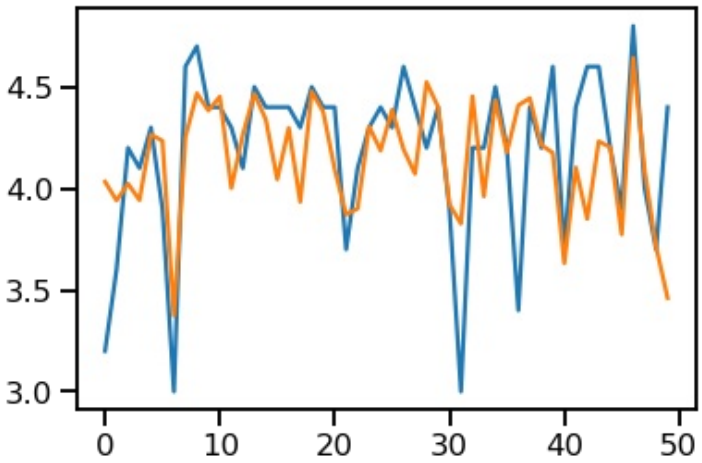
Out[41]:

	ytest	ypre
1671	4.8	4.185000
1626	3.7	3.148667
1295	4.2	3.343000
1361	4.4	4.300333
1325	4.4	4.407000

	ytest	ypre
807	4.8	4.764000
1061	4.8	4.584333
1830	4.3	3.829000
1700	4.4	3.732667
548	4.2	4.149333

In [42]:

```
y=ytest[0:50]
z=ypre[0:50]
fig, ax =plt.subplots()
ax.plot(y)
ax.plot(z)
plt.show()
```



In [43]:

```
from sklearn.metrics import r2_score
r2_score(ytest, ypre)
```

Out[43]:

```
0.025065005739678292
```

In [44]:

```
from sklearn.metrics import mean_squared_error
print(np.sqrt(mean_squared_error(ytest, ypre)))
```

```
0.532332034219816
```