In [1]:
```
filepath='DataFiles/AppleStore.csv'

import pandas as pd
def readCSVdata(filepath):
    return pd.read_csv(filepath)
readCSVdata(filepath)
```

Out[1]:

price	rating_count_tot	rating_count_ver	user_rating	user_rating_ver	ver	cont_rating	prime_ge
3.99	21292	26	4.0	4.5	6.3.5	4+	Gai
0.00	161065	26	4.0	3.5	8.2.2	4+	Product
0.00	188583	2822	3.5	4.5	5.0.0	4+	Wea
0.00	262241	649	4.0	4.5	5.10.0	12+	Shopp
0.00	985920	5320	4.5	5.0	7.5.1	4+	Refere
0.99	8253	5516	4.0	4.0	1.8	4+	Gai
0.00	119487	879	4.0	4.5	6.12.0	4+	Fina
0.00	1126879	3594	4.0	4.5	8.4.1	12+	M
9.99	1117	4	4.5	5.0	3.6.6	4+	Util
3.99	7885	40	4.0	4.0	4.0.4	4+	Gai
4.99	76720	4017	4.5	4.5	4.10.1	4+	Gai
7.99	105776	166	3.5	2.5	5.19.0	4+	Gai
0.00	479440	203	3.5	4.0	27.0	17+	Util
0.00	119773	2336	3.5	4.5	7.3.8	4+	Fina
4.99	6340	668	4.5	4.5	4.0.3	4+	Gai

price	rating_count_tot	rating_count_ver	user_rating	user_rating_ver	ver	cont_rating	prime_ge
0.00	56194	87	4.0	3.5	21.1	4+	Tr
0.00	2974676	212	3.5	3.5	95.0	4+	Sc Networ
0.00	223885	3726	4.0	4.5	11.15.0	12+	Tr
0.00	402925	136	4.0	4.5	11.0.3	12+	M
2.99	31456	4178	4.0	3.5	1.0.0	4+	Ga
1.99	2929	966	4.5	4.5	3.3	4+	Refere
2.99	11447	781	5.0	5.0	5.2.6	4+	Ga
0.00	137951	131	4.0	4.5	6.9	4+	Sp
5.99	8	0	4.5	0.0	9.0.5	4+	Busir
3.99	3241	297	4.0	4.0	2.1.3	4+	M
0.00	5795	12	3.5	4.0	8.4.1	4+	Heal Fitr
0.00	42316	248	3.0	3.5	2.0.6	9+	Ga
0.00	123215	25	3.5	5.0	10.5.4	4+	Product
4.99	782	7	4.0	3.5	1.9.8	4+	M
9.99	3449	23	4.0	4.5	3.4.1	4+	Wea
...	...	...	...	...	...	...	

price	rating_count_tot	rating_count_ver	user_rating	user_rating_ver	ver	cont_rating	prime_ge
0.00	1	1	1.0	1.0	1.0.1	4+	Gar
0.00	3	3	4.0	4.0	1.0.1	4+	Gar
2.99	9	3	3.0	3.5	1.0.1	4+	Entertainn
0.00	39	4	5.0	5.0	1.3	9+	So Networ
0.00	10	10	5.0	5.0	1.0	4+	Gar
3.99	55	29	4.5	4.5	1.4	4+	Gar
0.99	0	0	0.0	0.0	1.1	4+	Util
0.00	14	4	4.0	3.5	1.3.0	4+	Gar
0.00	41	19	4.5	4.5	1.3	4+	Gar
0.00	0	0	0.0	0.0	1.0.1	4+	Gar
0.00	11	8	4.0	4.0	1.1.0	4+	Gar
0.99	0	0	0.0	0.0	1.1	4+	Gar
0.00	279	5	3.5	3.0	0.6.41	4+	So Networ
0.00	26	3	5.0	5.0	1.0.5	9+	Util

price	rating_count_tot	rating_count_ver	user_rating	user_rating_ver	ver	cont_rating	prime_ge
0.99	0	0	0.0	0.0	1.0	9+	Gar
0.00	0	0	0.0	0.0	1.0.3	4+	Pho Vi
3.99	292	292	4.0	4.0	2.0.20.1	9+	Gar
0.00	0	0	0.0	0.0	1.0	9+	Gar
0.99	0	0	0.0	0.0	1.0	9+	Gar
0.00	1	1	2.0	2.0	1.0.1	4+	Gar
0.00	23	23	4.5	4.5	1.0	4+	Gar
0.00	18	18	4.0	4.0	1.0.0	12+	Gar
0.99	0	0	0.0	0.0	2.1.0	9+	Gar
2.99	97	97	3.0	3.0	1.0	4+	Gar
2.99	11	0	4.0	0.0	1.1.1	17+	Gar
0.00	142	75	4.5	4.5	1.3	4+	Gar
0.00	30	30	4.5	4.5	0.9	4+	Gar
1.99	15	0	4.5	0.0	1.0.2	9+	Util
0.00	85	32	4.5	4.5	1.0.15	12+	Gar
0.00	3	3	5.0	5.0	1.0	4+	Gar

In [3]:
```
# List of all unique Prime_Genres(categories) in the dataset

appstore=readCSVdata(filepath)

def getColumnIndex(df,columnkey):
    for i in range(len(df.columns)):
        if df.columns[i]==columnkey:
            columnindex=i
    return columnindex

def categories(df,columnkey):
    columnindex=getColumnIndex(df,columnkey)
    unique=[]
    for i in range(len(df.values)):
        category=df.values[i][columnindex]
        if category not in unique:
            unique.append(category)
    print(len(unique))
    return unique
categories(appstore,'prime_genre')

```

```
23
```

Out[3]:
```
['Games',
 'Productivity',
 'Weather',
 'Shopping',
 'Reference',
 'Finance',
 'Music',
 'Utilities',
 'Travel',
 'Social Networking',
 'Sports',
 'Business',
 'Health & Fitness',
 'Entertainment',
 'Photo & Video',
 'Navigation',
 'Education',
 'Lifestyle',
 'Food & Drink',
 'News',
 'Book',
 'Medical',
 'Catalogs']
```

In [4]:
```
# Category with highest number of apps


def categoryWithHigestapps(df,columnkey):
    columnindex=getColumnIndex(df,columnkey)
    all={}
    for i in range(len(df.values)):
        category=df.values[i][columnindex]
        if category in all:
            all[category]+=1
        else:
            all[category]=1
    number=all.values()
    highestapps=max(number)
    for item in all.items():
        if item[1]==highestapps:
            print(item[0],":",item[1])
categoryWithHigestapps(appstore,'prime_genre')

```

```
Games : 3862
```

In [10]:
```
def categoryWithLowestapps(df,columnkey):
    columnindex=getColumnIndex(df,columnkey)
    all={}
    for i in range(len(df.values)):
        category=df.values[i][columnindex]
        if category in all:
            all[category]+=1
        else:
            all[category]=1
    number=all.values()
    lowestapps=min(number)
    for item in all.items():
        if item[1]==lowestapps:
            print(item[0],":",item[1])
categoryWithLowestapps(appstore,'prime_genre')

```

```
Catalogs : 10
```

In [12]:
```
def categoryWithHighestUserRating(df,userratingkey,categorykey):
    userratingindex=getColumnIndex(df,userratingkey)
    categoryindex=getColumnIndex(df,categorykey)
    all=[]
    for i in range(len(df.values)):
        all.append(df.values[i][userratingindex])
    maxuserrating=max(all)
    unique={}
    for i in range(len(df.values)):
        if df.values[i][userratingindex]==maxuserrating:
            category=df.values[i][categoryindex]
            if category not in unique:
                unique[category]=maxuserrating

    print(unique)

categoryWithHighestUserRating(appstore,'user_rating','prime_genre')
```

```
{'Games': 5.0, 'Business': 5.0, 'Education': 5.0, 'Photo & Video': 5.0, 'Utilit
ies': 5.0, 'Shopping': 5.0, 'News': 5.0, 'Health & Fitness': 5.0, 'Productivit
y': 5.0, 'Food & Drink': 5.0, 'Reference': 5.0, 'Travel': 5.0, 'Lifestyle': 5.
0, 'Weather': 5.0, 'Music': 5.0, 'Book': 5.0, 'Finance': 5.0, 'Sports': 5.0, 'E
ntertainment': 5.0, 'Social Networking': 5.0, 'Catalogs': 5.0, 'Medical': 5.0,
'Navigation': 5.0}
```

In [13]:
```
# App with highest downloads

def appWithDownloads(df,ratingcountkey,tracknamekey):
    ratingcountindex=getColumnIndex(df,ratingcountkey)
    tracknameindex=getColumnIndex(df,tracknamekey)
    ratingcount=[]
    for i in range(len(df.values)):
        rating=df.values[i][ratingcountindex]
        ratingcount.append(rating)
    maxratingcount=max(ratingcount)
    unique={}
    for i in range(len(df.values)):
        if df.values[i][ratingcountindex]==maxratingcount:
            trackname=df.values[i][tracknameindex]
            if trackname not in unique:
                unique[trackname]=maxratingcount
    return unique

appWithDownloads(appstore,'rating_count_tot','track_name')
```

Out[13]:
```
{'Facebook': 2974676}
```

In [15]:
```
#Category with highest average rating count

def appWithHighestavgrating(df,ratingcountkey,categorykey):
    ratingcountindex=getColumnIndex(df,ratingcountkey)
    categoryindex=getColumnIndex(df,categorykey)
    ratingcount=[]
    for i in range(len(df.values)):
        rating=df.values[i][ratingcountindex]
        ratingcount.append(rating)
    maxratingcount=max(ratingcount)
    unique={}
    for i in range(len(df.values)):
        if df.values[i][ratingcountindex]==maxratingcount:
            category=df.values[i][categoryindex]
            if category not in unique:
                unique[category]=maxratingcount
    return unique

appWithHighestavgrating(appstore,'rating_count_tot','prime_genre')

```

Out[15]: {'Social Networking': 2974676}

In [18]:
```
# Average user rating for free apps and paid apps

def avguserRatingforfreeapps(df,pricekey,userratingkey):
    priceindex=getColumnIndex(df,pricekey)
    userratingindex=getColumnIndex(df,userratingkey)
    sum1=0
    count1=0
    sum2=0
    count2=0
    for i in range(len(df.values)):
        if df.values[i][priceindex]==0:
            sum1=sum1+df.values[i][userratingindex]
            count1=count1+1
        else:
            sum2=sum2+df.values[i][userratingindex]
            count2=count2+1
    avg1=sum1/count1
    avg2=sum2/count2
    print(avg1,avg2)
avguserRatingforfreeapps(appstore,'price','user_rating')


```

```
3.3767258382642997 3.720948742438714
```

In [31]:
```
# Category with highest average user rating for paid apps

def categoryWithHighestUserratingpaidapps(df,userratingkey,categorykey,price
    userratingindex=getColumnIndex(df,userratingkey)
    categoryindex=getColumnIndex(df,categorykey)
    priceindex=getColumnIndex(df,pricekey)

    categories=[]
    for i in range(len(df.values)):
        if df.values[i][priceindex]!=0:
            if df.values[i][userratingindex]==5:
                category=df.values[i][categoryindex]
                if category not in categories:
                    categories.append(category)
    return categories
categoryWithHighestUserratingpaidapps(appstore,'user_rating','prime_genre','
```

Out[31]:
```
['Games',
 'Business',
 'Education',
 'Photo & Video',
 'Health & Fitness',
 'Productivity',
 'Reference',
 'Lifestyle',
 'Weather',
 'Music',
 'Book',
 'Finance',
 'Sports',
 'Entertainment',
 'Travel',
 'Food & Drink',
 'Utilities',
 'Navigation',
 'Social Networking',
 'News']
```

In [21]:
```
# Most frequent Price point > 0

def freqPricePoint(df,pricekey):
    priceindex=getColumnIndex(df,pricekey)
    li=[]
    for i in range(len(df.values)):
        if df.values[i][priceindex]!=0:
            li.append(df.values[i][priceindex])
    unique={}
    for i in li:
        if i not in unique:
            unique[i]=1
        else:
            unique[i]+=1
    freq=unique.values()
    maxfreq=max(freq)
    for item in unique.items():
        if item[1]==maxfreq:
            print(item[0],":",item[1])
freqPricePoint(appstore,'price')

```

```
0.99 : 728
```

In [32]:
```
# Compare average user rating for paid vs free gaming apps

def paidVsfreeGaming(df,ratingkey,pricekey,categorykey):
    priceindex=getColumnIndex(df,pricekey)
    ratingindex=getColumnIndex(df,ratingkey)
    categoryindex=getColumnIndex(df,categorykey)
    sum1=0
    count1=0
    sum2=0
    count2=0
    for i in range(len(df.values)):
        if df.values[i][categoryindex]=='Games':
            if df.values[i][priceindex]==0:
                sum1=sum1+df.values[i][ratingindex]
                count1=count1+1
            else:
                sum2=sum2+df.values[i][ratingindex]
                count2=count2+1
    avg1=sum1/count1
    avg2=sum2/count2
    print(avg1,avg2)
    print(max(avg1,avg2),'is the Highest value')



paidVsfreeGaming(appstore,'user_rating','price','prime_genre')

```

```
3.5285777580859548 3.9049844236760123
3.9049844236760123 is the Highest value
```

In [ ]:
```
1
```