Day Objectives:-

03-07-2019

- List of all unique Prime_Genres (categories) in the dataset
- Category with highest number of apps
- Category with lowest number of apps
- Category with highest user rating
- App with highest downloads
- Category with highest average rating count
- Average user rating for free apps
- Average user rating for paid apps
- Category with highest average user rating for paid apps
- Most Frequent Price point >0
- Compare average user rating for piad vs free gaming apps

In [2]:
```
import pandas as pd

  # comma seprated values all spreads are csv files
def readCSVdata(filepath):
    return pd.read_csv(filepath)
filepath='DataFiles\AP.csv'
readCSVdata(filepath)
```

Out[2]:

	Unnamed: 0	id	track_name	size_bytes	currency	price	rating_count_tot	rating_
0	1	281656475	PAC-MAN Premium	100788224	USD	3.99	21292	
1	2	281796108	Evernote - stay organized	158578688	USD	0.00	161065	
2	3	281940292	WeatherBug - Local Weather, Radar, Maps, Alerts	100524032	USD	0.00	188583	
3	4	282614216	eBay: Best App to Buy, Sell, Save! Online Shop...	128512000	USD	0.00	262241	

In [2]:
```
# function to find the columns
Appdata=readCSVdata(filepath)
def columns(df):
    columns=df.columns
    for i in columns:
        print(i,end="\n")
    return
columns(Appdata)
```

```
Unnamed: 0
id
track_name
size_bytes
currency
price
rating_count_tot
rating_count_ver
user_rating
user_rating_ver
ver
cont_rating
prime_genre
sup_devices.num
ipadSc_urls.num
lang.num
vpp_lic
```

In [3]:
```
# Function to list our  the prime_generes values

def Prime_geners_Column(df):
    for i in range(len(df.values)):
        for j in range(12,len(df.columns)-4):
            print(df.values[i][j])
Prime_geners_Column(Appdata)
```

```
Games
Productivity
Weather
Shopping
Reference
Games
Finance
Music
Utilities
Games
Games
Games
Utilities
Finance
Games
Travel
Social Networking
Travel
Music
```

In [ ]:
```
# Function to find the prime_generes values
Appdata=readCSVdata(filepath)
def Prime_geners_Unique(df):
    u=[]
    for i in range(len(df.values)):
        for j in range(12,len(df.columns)-4):
            a=df.values[i][j]
            if a not in u:
                u.append(a)
    print(u)
Prime_geners_Unique(Appdata)
```

```
['Games', 'Productivity', 'Weather', 'Shopping', 'Reference', 'Finance', 'Musi
c', 'Utilities', 'Travel', 'Social Networking', 'Sports', 'Business', 'Health &
Fitness', 'Entertainment', 'Photo & Video', 'Navigation', 'Education', 'Lifesty
le', 'Food & Drink', 'News', 'Book', 'Medical', 'Catalogs']
```

In [3]:
```
# Function to find the prime_generes with highest no.of apps values
Appdata=readCSVdata(filepath)
def Prime_geners_Highest(df):
    u={}
    for i in range(len(df.values)):
        for j in range(12,len(df.columns)-4):
            a=df.values[i][j]
            if a not in u.keys():
                u[a]=1
            else:
                u[a]+=1
    print(u)
    m=sorted(u.values(),reverse=True)
    print(m)
    print('\n')
    max1=max(m)
    print(max1)
    for item in u.items():
        if item[1]==max1:
            print('\n')
            print(item[0],':',max1)


Prime_geners_Highest(Appdata)
```

```
{'Games': 3862, 'Productivity': 178, 'Weather': 72, 'Shopping': 122, 'Referenc
e': 64, 'Finance': 104, 'Music': 138, 'Utilities': 248, 'Travel': 81, 'Social N
etworking': 167, 'Sports': 114, 'Business': 57, 'Health & Fitness': 180, 'Enter
tainment': 535, 'Photo & Video': 349, 'Navigation': 46, 'Education': 453, 'Life
style': 144, 'Food & Drink': 63, 'News': 75, 'Book': 112, 'Medical': 23, 'Catal
ogs': 10}
[3862, 535, 453, 349, 248, 180, 178, 167, 144, 138, 122, 114, 112, 104, 81, 75,
72, 64, 63, 57, 46, 23, 10]


3862


Games : 3862
```

In [4]:
```
# Function to find the Category with lowest no. of apps values
Appdata=readCSVdata(filepath)
def Prime_geners_Lowest(df):
    u={}
    for i in range(len(df.values)):
        for j in range(12,len(df.columns)-4):
            a=df.values[i][j]
            if a not in u.keys():
                u[a]=1
            else:
                u[a]+=1
    print(u)
    print('\n')
    m=sorted(u.values(),reverse=True)
    print(m)
    print('\n')
    min1=min(m)
    for item in u.items():
        if item[1]==min1:
            print('\n')
            print(item[0],':',min1)



Prime_geners_Lowest(Appdata)
```

```
{'Games': 3862, 'Productivity': 178, 'Weather': 72, 'Shopping': 122, 'Referenc
e': 64, 'Finance': 104, 'Music': 138, 'Utilities': 248, 'Travel': 81, 'Social N
etworking': 167, 'Sports': 114, 'Business': 57, 'Health & Fitness': 180, 'Enter
tainment': 535, 'Photo & Video': 349, 'Navigation': 46, 'Education': 453, 'Life
style': 144, 'Food & Drink': 63, 'News': 75, 'Book': 112, 'Medical': 23, 'Catal
ogs': 10}


[3862, 535, 453, 349, 248, 180, 178, 167, 144, 138, 122, 114, 112, 104, 81, 75,
72, 64, 63, 57, 46, 23, 10]



Catalogs : 10
```

In [53]:
```
# Function to find the category with highest user rating

Appdata=readCSVdata(filepath)

def columnIndex(df,key):
    for i in range(len(df.columns)):
        if df.columns[i]==key:
            CI=i
    return CI
columnIndex(Appdata,'price')

def Index1(df,key):
    for j in range(len(df.columns)):
        if df.columns[j]==key:
            return j
Index1(Appdata,'prime_genre')


def Highest_user_Rating(df,key,key1):
    lis=[]
    index=columnIndex(df,key)
    index1=Index1(df,key1)
    for row in df.values:
        lis.append(row[index])
    val=max(lis)
    p=[]
    for row in df.values:
        if val == row[index]:
            p.append(row[index1])
    print(set(p))
    print(len(set(p)))
Highest_user_Rating(Appdata,"user_rating","prime_genre")
```

```
{'Utilities', 'Catalogs', 'Medical', 'Education', 'Reference', 'Health & Fitnes
s', 'News', 'Book', 'Travel', 'Entertainment', 'Weather', 'Business', 'Sports',
'Games', 'Social Networking', 'Lifestyle', 'Food & Drink', 'Shopping', 'Music',
'Finance', 'Productivity', 'Photo & Video', 'Navigation'}
23
```

In [4]:
```
# Function  to find the app with highest downloads

Appdata=readCSVdata(filepath)

def ColInd(df,key):
    for i in range(len(df.values)):
        if df.columns[i]==key:
            return i
ColInd(Appdata,'rating_count_tot')

def Index1(df,key):
    for i in range(len(df.values)):
        if df.columns[i]==key:
            return i
Index1(Appdata,'track_name')

def AppHighestDownload(df,key,key1):
    CI=ColInd(df,key)
    CI1=Index1(df,key1)
    li=[]
    for i in df.values:
        li.append(i[CI])
    m=max(li)
    print(m)
    s=[]
    for j in df.values:
        if m==j[CI]:
            s.append(j[CI1])
    print(s,':',m)
AppHighestDownload(Appdata,'rating_count_tot','track_name')
```

```
2974676
['Facebook'] : 2974676
```

In [71]:
```
# Function to find the category with highest average  rating count

def ci(df,key):
    for i in range(len(df.values)):
        if df.columns[i]==key:
            return i
ci(Appdata,'rating_count_tot')

def ci1(df,key1):
    for i in range(len(df.values)):
        if df.columns[i]==key1:
            return i
ci1(Appdata,'prime_genre')

def HighRatingCount(df,key,key1):
    CI=ci(df,key)
    CI1=ci1(df,key1)
    u=[]
    for i in df.values:
        u.append(i[CI])
    a=max(u)
    print(a)
    print('\n')
    s=[]
    for i in df.values:
        if a==i[CI]:
            s.append(i[CI1])
    print(s,':',a)
HighRatingCount(Appdata,'rating_count_tot','prime_genre')
```

```
2974676


['Social Networking'] : 2974676
```

In [11]:
```
# Function to find the Average user rating for free apps


def ci(df,key):
    for i in range(len(df.values)):
        if df.columns[i]==key:
            return i
ci(Appdata,'user_rating')

def ci1(df,key1):
    for i in range(len(df.values)):
        if df.columns[i]==key1:
            return i
ci1(Appdata,'price')

def Average_user_rating_free_apps(df,key,key1):
    u=[]
    CI=ci(df,key)
    CI1=ci1(df,key1)
    for i in df.values:
        u.append(i[CI1])
    a=min(u)
    print(a)
    s=[]
    for j in df.values:
        if a==j[CI1]:
            s.append(j[CI])
    b=(sum(s)/len(s))
    print(b)
Average_user_rating_free_apps(Appdata,'user_rating','price')
```

```
0.0
3.3767258382642997
```

In [28]:
```
# Average user rating for paid apps

def ci(df,key):
    for i in range(len(df.values)):
        if df.columns[i]==key:
            return i
def ci1(df,key1):
    for i in range(len(df.values)):
        if df.columns[i]==key1:
            return i

def Average_userrating_paid_apps(df,key,key1):
    CI=ci(df,key)
    CI1=ci1(df,key1)
    u=[]
    s=0
    c=0
    for i in df.values:
        if i[CI1]>0:
            s=s+i[CI]
            c=c+1
    print("Average user rating for paid apps",':',s/c)
Average_userrating_paid_apps(Appdata,'user_rating','price')
```

```
Average user rating for paid apps : 3.720948742438714
```

In [49]:
```
# Function to find the category with highest average user rating for paid apps

def ci(df,key):
    for i in range(len(df.values)):
        if df.columns[i]==key:
            return i
def ci1(df,key1):
    for i in range(len(df.values)):
        if df.columns[i]==key1:
            return i
def ci2(df,key2):
    for i in range(len(df.values)):
        if df.columns[i]==key2:
            return i

def Highest_average_userrating_Paid_apps(df,key,key1,key2):
    CI=ci(df,key)
    CI1=ci1(df,key1)
    CI2=ci2(df,key2)
    u=[]
    for i in df.values:
        if i[CI1]>0:
            u.append(i[CI])
    m=max(u)
    print("max user rating",':',m,'\n')
    u1=[]
    for k in df.values:
        if k[CI]==m:
            u1.append(k[CI2])
    print("Category with highest user rating ",':',set(u1))
    print('\n')
    print(len(set(u1)))
Highest_average_userrating_Paid_apps(Appdata,'user_rating','price','prime_genre'
```

```
max user rating : 5.0

Category with highest user rating  : {'Lifestyle', 'Sports', 'Entertainment',
'Productivity', 'News', 'Health & Fitness', 'Food & Drink', 'Photo & Video', 'C
atalogs', 'Music', 'Shopping', 'Travel', 'Reference', 'Utilities', 'Medical',
'Finance', 'Weather', 'Education', 'Social Networking', 'Book', 'Games', 'Navig
ation', 'Business'}


23
```

In [59]:
```
# Most Frequent Price point >0

def ci(df,key):
    for i in range(len(df.values)):
        if df.columns[i]==key:
            return i

def ci1(df,key1):
    for i in range(len(df.values)):
        if df.columns[i]==key1:
            return i

def Most_frequent_price_greater_zero(df,key):
    CI=ci(df,key)
    u={}
    u1=[]
    for i in df.values:
        if i[CI]>0:
            u1.append(i[CI])
    for i in u1:
        if i not in u:
            u[i]=1
        else:
            u[i]+=1
    print(u,'\n')
    m=max(u.values())
    print("maximum price frequent",m,'\n')
    for item in u.items():
        if item[1]==m:
            print(item[0],"is the most frequent price having ':'",m)
Most_frequent_price_greater_zero(Appdata,'price')
```

```
{3.99: 277, 0.99: 728, 9.99: 81, 4.99: 394, 7.99: 33, 2.99: 683, 1.99: 621, 5.9
9: 52, 12.99: 5, 21.99: 1, 249.99: 1, 6.99: 166, 74.99: 1, 19.99: 13, 8.99: 9,
24.99: 8, 13.99: 6, 14.99: 21, 16.99: 2, 47.99: 1, 11.99: 6, 59.99: 3, 15.99:
4, 27.99: 2, 17.99: 3, 299.99: 1, 49.99: 2, 23.99: 2, 20.99: 2, 39.99: 2, 99.9
9: 1, 29.99: 6, 34.99: 1, 18.99: 1, 22.99: 2}

maximum price frequent 728

0.99 is the most frequent price having : 728
```

In [86]:
```
# Function to find the compare the average user rating for paid vs free gaming ap

def ci(df,key):
    for i in range(len(df.values)):
        if df.columns[i]==key:
            return i

def ci1(df,key1):
    for i in range(len(df.values)):
        if df.columns[i]==key1:
            return i

def ci2(df,key2):
    for i in range(len(df.values)):
        if df.columns[i]==key2:
            return i

def Paid_vs_Free_Gaming_apps(df,key,key1,key2):
    CI=ci(df,key)
    CI1=ci1(df,key1)
    CI2=ci2(df,key2)
    sum1=0
    count1=0
    s1=0
    c1=0
    for i in range(len(df.values)):
        if df.values[i][CI]=='Games':
            if df.values[i][CI1]>0:
                sum1=sum1+df.values[i][CI2]
                count1=count1+1
            else:
                s1=s1+df.values[i][CI2]
                c1=c1+1
    avg1=sum1/count1
    avg2=s1/c1
    print(avg1,avg2,'\n')
    if avg1>avg2:
        print("Paid apps are having highest avg  than Free appps",':',avg1,'\n')
    else:
        print("Free apps are having highest avg than Paid apps",':'.avg2)

Paid_vs_Free_Gaming_apps(Appdata,'prime_genre','price','user_rating')
```

```
3.9049844236760123 3.5285777580859548

Paid apps are having highest avg  than Free appps : 3.9049844236760123
```

In [74]:
```
n=int(input())
n1=input().split()
c=0
for i in n1:
    if  int(i)==1:
        c=c+1
print(c)
```

```
5
1 1 10 1
3
```

In [ ]:
```

```