Day Objectives:-

Date 03/07/2019 to 04/07/2019

- 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

App Store Data

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_cou
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	
4	5	282935706	Bible	92774400	USD	0.00	985920	
5	6	283619399	Shanghai Mahjong	10485713	USD	0.99	8253	
6	7	283646709	PayPal - Send and request money safely	227795968	USD	0.00	119487	
7	8	284035177	Pandora - Music & Radio	130242560	USD	0.00	1126879	
8	9	284666222	PCalc - The Best Calculator	49250304	USD	9.99	1117	
9	10	284736660	Ms. PAC-MAN	70023168	USD	3.99	7885	
10	11	284791396	Solitaire by MobilityWare	49618944	USD	4.99	76720	
11	12	284815117	SCRABBLE Premium	227547136	USD	7.99	105776	
12	13	284815942	Google – Search made just for mobile	179979264	USD	0.00	479440	

	Unnamed: 0	id	track_name	size_bytes	currency	price	rating_count_tot	rating_cou
13	14	284847138	Bank of America - Mobile Banking	160925696	USD	0.00	119773	
14	15	284862767	FreeCell	55153664	USD	4.99	6340	
15	16	284876795	TripAdvisor Hotels Flights Restaurants	207907840	USD	0.00	56194	
16	17	284882215	Facebook	389879808	USD	0.00	2974676	
17	18	284910350	Yelp - Nearby Restaurants, Shopping & Services	167407616	USD	0.00	223885	
18	20	284993459	Shazam - Discover music, artists, videos & lyrics	147093504	USD	0.00	402925	
19	21	285005463	Crash Bandicoot Nitro Kart 3D	10735026	USD	2.99	31456	
20	22	285946052	iQuran	70707916	USD	1.99	2929	
21	23	285994151	:) Sudoku +	6169600	USD	2.99	11447	
22	24	286058814	Yahoo Sports - Teams, Scores, News & Highlights	130583552	USD	0.00	137951	
23	25	286070473	Mileage Log | Fahrtenbuch	71203840	USD	5.99	8	
24	27	286799607	Cleartune - Chromatic Tuner	11423008	USD	3.99	3241	
25	28	286906691	Lifesum – Inspiring healthy lifestyle app	188017664	USD	0.00	5795	
26	29	286911400	Hangman.	4765696	USD	0.00	42316	
27	31	288113403	iTranslate - Language Translator & Dictionary	287933440	USD	0.00	123215	
28	32	288120394	TouchOSC	4263936	USD	4.99	782	
29	33	288419283	RadarScope	172772352	USD	9.99	3449	

	Unnamed: 0	id	track_name	size_bytes	currency	price	rating_count_tot	rating_cou
...	...	...	...	...	...	...	...	
7167	10995	1182265441	脱出ゲーム　わたしをみつけて　-おじいさんとわたしの物語-	177498112	USD	0.00	1	
7168	10998	1182331762	Escape from the frigid Igloo.	89188352	USD	0.00	3	
7169	11002	1182568288	Talking Santa - Video santa claus calls you	32685056	USD	2.99	9	
7170	11010	1183234072	CTFxCmoji	26077184	USD	0.00	39	
7171	11013	1183260922	Room Escape Game - Santa's Room	143346688	USD	0.00	10	
7172	11016	1183548754	Rescue the Enchanter	242505728	USD	3.99	55	
7173	11019	1183709176	My Diary - 你的名字非官方	18164736	USD	0.99	0	
7174	11022	1183856228	VR Thrills: Roller Coaster 360 (Google Cardboard)	169535488	USD	0.00	14	
7175	11024	1183986102	Santa Kids Hair Salon - Christmas Makeover Games	64244736	USD	0.00	41	
7176	11027	1184711626	Human Juggling Cup	184324096	USD	0.00	0	
7177	11031	1184800011	Again - room escape game	33946624	USD	0.00	11	
7178	11033	1185209084	Saloons Unleashed	327731200	USD	0.99	0	
7179	11035	1185328193	Fam — Group video calling for iMessage	113382400	USD	0.00	279	

	Unnamed: 0	id	track_name	size_bytes	currency	price	rating_count_tot	rating_cou
7180	11036	1185365336	Laurie Hernandez the Human Emoji	94008320	USD	0.00	26	
7181	11038	1185428381	剑倚手游	178160640	USD	0.99	0	
7182	11040	1185538497	camera for filter	9362432	USD	0.00	0	
7183	11041	1185580782	Survivalcraft 2	57349120	USD	3.99	292	
7184	11042	1185731859	剑客情缘-高爆率高掉落天天疯玩	171944960	USD	0.00	0	
7185	11043	1185777521	问仙奇遇-新玩法新套装嗨到爆	208026624	USD	0.99	0	
7186	11050	1186108496	脱出ゲーム - 書道教室 - "漢字"の謎に満ちた部屋からの 脱出	85580800	USD	0.00	1	
7187	11051	1186126548	Escape Game: illumination	52342784	USD	0.00	23	
7188	11060	1186384912	Demolition Derby Virtual Reality (VR) Racing	168774656	USD	0.00	18	
7189	11074	1187128255	飞刀传奇-动作武侠热血江湖即时PK传奇（登录爆金装）	537462784	USD	0.99	0	
7190	11077	1187279979	Add-Ons Studio for Minecraft	22999040	USD	2.99	97	
7191	11079	1187282363	Plead the Fifth - The Game	27853824	USD	2.99	11	
7192	11081	1187617475	Kubik	126644224	USD	0.00	142	
7193	11082	1187682390	VR Roller-Coaster	120760320	USD	0.00	30	
7194	11087	1187779532	Bret Michaels Emojis + Lyric Keyboard	111322112	USD	1.99	15	

	Unnamed: 0	id	track_name	size_bytes	currency	price	rating_count_tot	rating_cou
7195	11089	1187838770	VR Roller Coaster World - Virtual Reality	97235968	USD	0.00	85	
7196	11097	1188375727	Escape the Sweet Shop Series	90898432	USD	0.00	3	

7197 rows × 17 columns

In [ ]:

Prime generes Categories

In [6]:
```
# 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 [ ]:

Prime generes with highest no. of apps values

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 [ ]:
```

```

Category with lowest no. of apps values

In [9]:
```
# 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 [ ]:
```

```

Category with highest user rating

In [16]:
```
# Function to find the category with highest user rating

Appdata=readCSVdata(filepath)
def Highest_user_Rating(df):
    lis=[]
    for row in df.values:
        lis.append(row[8])
    val=max(lis)
    print("max",':',val,'\n')
    p=[]
    for row in df.values:
        if val == row[8]:
            p.append(row[12])
    print(set(p),'\n')
    print(len(set(p)))
Highest_user_Rating(Appdata)
```

```
max : 5.0

{'Productivity', 'Catalogs', 'Games', 'Health & Fitness', 'Food & Drink', 'Life
style', 'Travel', 'Music', 'Photo & Video', 'News', 'Utilities', 'Navigation',
'Reference', 'Education', 'Sports', 'Business', 'Book', 'Medical', 'Social Netw
orking', 'Weather', 'Entertainment', 'Finance', 'Shopping'}

23
```

In [ ]:
```
```

App with highest downloads

In [17]:
```
# Function to find 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 [ ]:
```

```

Category with highest average rating count

In [18]:
```
# 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 [ ]:
```

```

Average User Rating for Free apps

In [6]:
```
# 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("min value for free apps",':',a)
    s=[]
    for j in df.values:
        if a==j[CI1]:
            s.append(j[CI])
    b=(sum(s)/len(s))
    print("Average user rating free apps",':',b)
Average_user_rating_free_apps(Appdata,'user_rating','price')
```

```
min value for free apps : 0.0
Average user rating free apps : 3.3767258382642997
```

In [ ]:
```

```

Average user rating for paid apps

In [7]:
```
# 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 [ ]:
```

```

Category with highest average user rating for paid apps

In [8]:
```
# 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  : {'Catalogs', 'Reference', 'Lifestyle', 'Sh
opping', 'News', 'Productivity', 'Book', 'Finance', 'Music', 'Photo & Video',
'Games', 'Food & Drink', 'Weather', 'Sports', 'Medical', 'Travel', 'Entertainme
nt', 'Navigation', 'Health & Fitness', 'Business', 'Education', 'Utilities', 'S
ocial Networking'}


23
```

In [ ]:
```

```

Most Frequent Price Point greater than zero

In [9]:
```
# 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 [ ]:

Comparing the average user rating for paid vs free gaming apps

In [10]:
```
# 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 [ ]: