Problem Set

App Store Dataset

- 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 paid vs free gaming apps

In [ ]:

In [2]:

```
import pandas as pd
# Reading CSV files

filepath='DataFiles/AppleStore.csv'

AppleStore=pd.read_csv(filepath)

AppleStore
```

Out[2]:

	Unnamed: 0	id	track_name	size_bytes	currency	price	rating_count_tot	ratir
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	
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	

	Unnamed: 0	id	track_name	size_bytes	currency	price	rating_count_tot	ratir
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	
...	...	...	...	...	...	...	...	
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	

	Unnamed: 0	id	track_name	size_bytes	currency	price	rating_count_tot	ratir
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	
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	

	Unnamed: 0	id	track_name	size_bytes	currency	price	rating_count_tot	ratir
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	
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 [ ]:

In [ ]:

```
def getRowIndex(df,rowkey):
    for i in range(len(df.values)):
        if df.values[i][0]==rowkey or df.values[i][1]==rowkey:
            rowindex=i
    return rowindex

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

In [ ]:

```

```

In [ ]:

```
# List of all unique Prime_Genres(categories) in the dataset
prime_genreList=[]
ColumnIndex=getColumnIndex(AppleStore,'prime_genre')
#prime_genreList.append(AppleStore.values[:][11])
prime_genreList
for x in range (1,len(AppleStore.values)):
    prime_genreList.append(AppleStore.values[x][12])

Unique_prime_genreList=list(set(prime_genreList))
Unique_prime_genreList
```

In [ ]:

```

```

In [2]:

```
# List of all unique Prime_Genres(categories) in the dataset
def UniqueCategory(AppleStore,Category):
    Prime_Genres=[];Unique_Prime_Genres=[]
    Prime_Genres=list(AppleStore.loc[:,Category])
    Unique_Prime_Genres=set(Prime_Genres)
    Unique_Category_Count={}
    for PG in Unique_Prime_Genres:
        Unique_Category_Count[PG]=Prime_Genres.count(PG)
    return Unique_Category_Count

UniqueCategory(AppleStore,'prime_genre')
```

Out[2]:

```
{'Social Networking': 167,
 'Medical': 23,
 'Sports': 114,
 'Health & Fitness': 180,
 'Entertainment': 535,
 'Lifestyle': 144,
 'Utilities': 248,
 'Reference': 64,
 'Catalogs': 10,
 'Photo & Video': 349,
 'News': 75,
 'Navigation': 46,
 'Food & Drink': 63,
 'Shopping': 122,
 'Education': 453,
 'Book': 112,
 'Games': 3862,
 'Finance': 104,
 'Music': 138,
 'Weather': 72,
 'Productivity': 178,
 'Travel': 81,
 'Business': 57}
```

In [ ]:

```

```

In [5]:

```
# Category with highest number of apps
def HiggestPrimeGenres(AppleStore):
    Unique_Prime_Genres_Count=UniqueCategory(AppleStore,'prime_genre')
    for items in Unique_Prime_Genres_Count.items():
        if max(Unique_Prime_Genres_Count.values())==items[1]:
            print(items[0])
HiggestPrimeGenres(AppleStore)
```

```
Games
```

In [ ]:

```

```

In [6]:

```
# Category with lowest number of apps
def LowestPrimeGenres(AppleStore):
    Unique_Prime_Genres_Count=UniqueCategory(AppleStore,'prime_genre')
    for items in Unique_Prime_Genres_Count.items():
        if min(Unique_Prime_Genres_Count.values())==items[1]:
            print(items[0])
LowestPrimeGenres(AppleStore)
```

```
Catalogs
```

In [ ]:

```

```

In [7]:

```
Unique_user_rating_Count={}
Unique_user_rating_Count=UniqueCategory(AppleStore,'user_rating')
MaxUsermax=max((Unique_user_rating_Count.keys()))
MaxUserMaxCat={}
for i in range(0,len(AppleStore.values)):
    if AppleStore.values[i][8]==MaxUsermax:
        if AppleStore.values[i][12] not in MaxUserMaxCat.keys():
            MaxUserMaxCat[AppleStore.values[i][12]]=1
        else:
            MaxUserMaxCat[AppleStore.values[i][12]]+=1
MaxUserMaxCat
#max(MaxUserMaxCat.values())
```

Out[7]:

```
{'Games': 277,
 'Business': 4,
 'Education': 24,
 'Photo & Video': 30,
 'Utilities': 12,
 'Shopping': 12,
 'News': 2,
 'Health & Fitness': 24,
 'Productivity': 13,
 'Food & Drink': 6,
 'Reference': 8,
 'Travel': 5,
 'Lifestyle': 8,
 'Weather': 2,
 'Music': 6,
 'Book': 14,
 'Finance': 4,
 'Sports': 6,
 'Entertainment': 26,
 'Social Networking': 5,
 'Catalogs': 1,
 'Medical': 2,
 'Navigation': 1}
```

In [ ]:

```
```

In [ ]:

```
# App with highest downloads
def HighestDownloads(AppleStore):
    Highest=(AppleStore.nlargest(1, 'rating_count_tot'))
    for i in range(0,len(Highest.values)):
        print(Highest.values[i][2])
    return

HighestDownloads(AppleStore)
```

In [ ]:

```
```

In [3]:

```
# Category with highest average rating count
def HighestAverageRating(AppleStore):

    category=UniqueCategory(AppleStore,'prime_genre')
    highestrating={}
    for cat in category:
        rating=[]
        for i in range(0,len(AppleStore.values)):
            if AppleStore.values[i][12]==cat:
                rating.append(AppleStore.values[i][6])
        highestrating[cat]=sum(rating)//len(rating)
    return highestrating

HighestAverageRating(AppleStore)
```

Out[3]:

```
{'Social Networking': 45498,
 'Medical': 592,
 'Sports': 14026,
 'Health & Fitness': 9913,
 'Entertainment': 7533,
 'Lifestyle': 6161,
 'Utilities': 6863,
 'Reference': 22410,
 'Catalogs': 1732,
 'Photo & Video': 14352,
 'News': 13015,
 'Navigation': 11853,
 'Food & Drink': 13938,
 'Shopping': 18615,
 'Education': 2239,
 'Book': 5125,
 'Games': 13691,
 'Finance': 11047,
 'Music': 28842,
 'Weather': 22181,
 'Productivity': 8051,
 'Travel': 14129,
 'Business': 4788}
```

In [ ]:

```

```

In [45]:

```
# Average user rating for free apps
# Method-1
AvgRating=AppleStore.groupby(['price']).mean()

FreeAvgRating=AvgRating.values[0][5]

print('Highest Average User Rating for Free Apps',FreeAvgRating)
```

```
Highest Average User Rating for Free Apps 3.3767258382642997
```

In [ ]:

```
```

In [43]:

```
# Average user rating for paid apps
# Method-2

FreeAvgRating=AppleStore.groupby(['price']).mean().filter(items=['price','user_rating'
]).values[0]

print('Highest Average User Rating for Free Apps',FreeAvgRating[0])
```

```
Highest Average User Rating for Free Apps 3.3767258382642997
```

In [ ]:

```
```

In [42]:

```
# Category with highest average user rating for paid apps

FreeAvgRating=AppleStore.groupby(['price']).mean().filter(items=['price','user_rating'
])

HighestAverageUserRating=sum(FreeAvgRating.values[1:])/len(FreeAvgRating.values[1:])

print('Highest Average User Rating for Paid Apps',HighestAverageUserRating[0])
```

```
Highest Average User Rating for Paid Apps 3.5900690165853533
```

In [ ]:

```
```

In [13]:

```
# Most frequent Price point > 0

def MostFrequentPrice(AppleStore,Category):
    Price= UniqueCategory(AppleStore,Category)
    res=sorted(list(Price.keys()))
    if res[0]==0 or res[0]==0.0:
        print(res[1],"Dollars with Frequency of",Price[res[1]],"times")
    else:
        print(res[0],"Dollars with Frequency of",Price[res[0]],"times")
    return


MostFrequentPrice(AppleStore,'price')
```

```
0.99 Dollars with Frequency of 728 times
```

In [ ]:

```
```

In [36]:

```
# Compare average user rating for paid vs free gaming apps

y=AppleStore.groupby(['prime_genre','price']).mean()

AverageGameApps=y.loc['Games']
AverageFreeGameApps=AverageGameApps.values[0][5]
AveragePaidGameApps=AverageGameApps.values[1:,5].mean()
print('Average of Free Game Apps: ',AverageFreeGameApps,"\nAverage of Paid Game Apps: "
,AveragePaidGameApps)
```

```
Average of Free Game Apps:  3.5285777580859548
Average of Paid Game Apps:  3.5373871699218817
```