Analysis of Free Mobile Applications and Ad Revenue

The purpose of this project is to analyze the current offering of applications in both the Apple and Android app stores, in an effort to help developers understand what apps are likely to attract more users.

Beyond the goal of user attraction, we are also interested to see what users, and how many, both see and engage in in-app ads. The purpose of this information is to guide development decisions to maximize ad revenue and product usage.

In [3]:
```
def explore_data(dataset, start, end, rows_and_columns=False):
    dataset_slice = dataset[start:end]
    for row in dataset_slice:
        print(row)
        print('\n') # adds a new (empty) line after each row

    if rows_and_columns:
        print('Number of rows:', len(dataset))
        print('Number of columns:', len(dataset[0]))
```

In [4]:
```
from csv import reader
apple_store = open('c:\\users\\daxto\\Jupyter DataSets\\AppleStore.csv', encod
applestoredata = reader(apple_store)
apple_data = list(applestoredata)
google_play = open('c:\\users\\daxto\\Jupyter DataSets\\googleplaystore.csv', e
googlestoredata = reader(google_play)
google_data = list(googlestoredata)
```

In [5]:
```
explore_data(apple_data,1,3,True)
```

```
['1', '281656475', 'PAC-MAN Premium', '100788224', 'USD', '3.99', '21292', '2
6', '4', '4.5', '6.3.5', '4+', 'Games', '38', '5', '10', '1']


['2', '281796108', 'Evernote - stay organized', '158578688', 'USD', '0', '161
065', '26', '4', '3.5', '8.2.2', '4+', 'Productivity', '37', '5', '23', '1']


Number of rows: 7198
Number of columns: 17
```

In [6]:
```
explore_data(google_data,1,3,True)
```

```
['Photo Editor & Candy Camera & Grid & ScrapBook', 'ART_AND_DESIGN', '4.1',
'159', '19M', '10,000+', 'Free', '0', 'Everyone', 'Art & Design', 'January 7,
2018', '1.0.0', '4.0.3 and up']


['Coloring book moana', 'ART_AND_DESIGN', '3.9', '967', '14M', '500,000+', 'F
ree', '0', 'Everyone', 'Art & Design;Pretend Play', 'January 15, 2018', '2.0.
0', '4.0.3 and up']


Number of rows: 10842
Number of columns: 13
```

Getting Familiar

Through the explore_data function, we have been able to see what the table looks like and what we can expect from each column.

Below is printed the column headers for each data set. If any questions arise about the columns, we can refer back to the documentation for either the Apple Store (https://www.kaggle.com/datasets/ramamet4/app-store-apple-data-set-10k-apps) or Google Store (https://www.kaggle.com/datasets/lava18/google-play-store-apps) data sets.

In [7]:
```
print(apple_data[0])
print('\n')
print(google_data[0])
```

```
['', 'id', 'track_name', 'size_bytes', 'currency', 'price', 'rating_count_to
t', 'rating_count_ver', 'user_rating', 'user_rating_ver', 'ver', 'cont_ratin
g', 'prime_genre', 'sup_devices.num', 'ipadSc_urls.num', 'lang.num', 'vpp_li
c']


['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type', 'Price',
'Content Rating', 'Genres', 'Last Updated', 'Current Ver', 'Android Ver']
```

Data Cleaning

An error was reported in the discussion section saying that there is a wrong entry for 10472. They said that the entry is missing a rating column and shifted everything over as a result.

In [8]:
```
print(google_data[10473])
print('\n')
print(google_data[0])
print('\n')
print(google_data[1])
```

```
['Life Made WI-Fi Touchscreen Photo Frame', '1.9', '19', '3.0M', '1,000+', 'F
ree', '0', 'Everyone', '', 'February 11, 2018', '1.0.19', '4.0 and up']


['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type', 'Price',
'Content Rating', 'Genres', 'Last Updated', 'Current Ver', 'Android Ver']


['Photo Editor & Candy Camera & Grid & ScrapBook', 'ART_AND_DESIGN', '4.1',
'159', '19M', '10,000+', 'Free', '0', 'Everyone', 'Art & Design', 'January 7,
2018', '1.0.0', '4.0.3 and up']
```

In [9]:
```
print(len(google_data))
del google_data[10473]
print(len(google_data))
```

```
10842
10841
```

In [10]:
```
print(google_data[10473])
del google_data[10473]
```

```
['osmino Wi-Fi: free WiFi', 'TOOLS', '4.2', '134203', '4.1M', '10,000,000+',
'Free', '0', 'Everyone', 'Tools', 'August 7, 2018', '6.06.14', '4.4 and up']
```

Do Not Run Anything Above This

Testing for Duplicates

Now we want to test for duplicate entries. The Google Play dataset has duplicate entries of many apps including Instagram.

In [11]:
```
for app in google_data[1:]:
    name = app[0]
    if name == 'Instagram':
        print(app)
```

```
['Instagram', 'SOCIAL', '4.5', '66577313', 'Varies with device', '1,000,000,0
00+', 'Free', '0', 'Teen', 'Social', 'July 31, 2018', 'Varies with device',
'Varies with device']
['Instagram', 'SOCIAL', '4.5', '66577446', 'Varies with device', '1,000,000,0
00+', 'Free', '0', 'Teen', 'Social', 'July 31, 2018', 'Varies with device',
'Varies with device']
['Instagram', 'SOCIAL', '4.5', '66577313', 'Varies with device', '1,000,000,0
00+', 'Free', '0', 'Teen', 'Social', 'July 31, 2018', 'Varies with device',
'Varies with device']
['Instagram', 'SOCIAL', '4.5', '66509917', 'Varies with device', '1,000,000,0
00+', 'Free', '0', 'Teen', 'Social', 'July 31, 2018', 'Varies with device',
'Varies with device']
```

We have confirmed many duplicates in the data set. To combat this we have created two lists, one for unique names and one for duplicate names. The loop underneath this checks to see if the name is already listed in the unique list, if not it appends to the unique list. If it is found in the unique list, it appends to the duplicate app names.

In [12]:
```
unique_google_apps = []
duplicate_google_apps = []

for app in google_data[1:]:
    name = app[0]
    if name in unique_google_apps:
        duplicate_google_apps.append(name)
    else:
        unique_google_apps.append(name)

print('Number of duplicate apps:', len(duplicate_google_apps))
print('Number of unique apps:', len(unique_google_apps))
print('\n')
print('Example of duplicates:', duplicate_google_apps[:10])
```

```
Number of duplicate apps: 1180
Number of unique apps: 9659


Example of duplicates: ['Quick PDF Scanner + OCR FREE', 'Box', 'Google My Bus
iness', 'ZOOM Cloud Meetings', 'join.me - Simple Meetings', 'Box', 'Zenefit
s', 'Google Ads', 'Google My Business', 'Slack']
```

We won't remove duplicates at random, it's important to understand what might be different. In this case it's number of reviews as the data was collected. We will want to keep the most reviews an app received.

In [13]:
```
del google_data[10473]
```

In [14]:
```
reviews_max = {}

for app in google_data[1:]:
    name = app[0]
    n_reviews = float(app[3])

    if name in reviews_max and reviews_max[name] < n_reviews:
        reviews_max[name] = n_reviews

    elif name not in reviews_max:
        reviews_max[name] = n_reviews
```

Checking length of dictionaries and lists to match up.

In [15]:
```
print(len(reviews_max))
```

```
9658
```

In [16]:
```
android_clean = []
already_added = []

for app in google_data[1:]:
    name = app[0]
    n_reviews = float(app[3])

    if (n_reviews == reviews_max[name]) and (name not in already_added):
        android_clean.append(app)
        already_added.append(name)
```

After cleaning the google data, I realized that my naming convention wasn't consistent with the lists. In addition, I kept the column headers in the lists up to this point. In the cell below, I have created a new list called `apple_clean` which is the apple list without column headers.

In [17]:
```
apple_clean = []
for row in apple_data[1:]:
    apple_clean.append(row)
```

Checking and viewing data

In [18]:
```
explore_data(android_clean, 0, 3, True)
```

```
['Photo Editor & Candy Camera & Grid & ScrapBook', 'ART_AND_DESIGN', '4.1',
'159', '19M', '10,000+', 'Free', '0', 'Everyone', 'Art & Design', 'January 7,
2018', '1.0.0', '4.0.3 and up']


['U Launcher Lite – FREE Live Cool Themes, Hide Apps', 'ART_AND_DESIGN', '4.
7', '87510', '8.7M', '5,000,000+', 'Free', '0', 'Everyone', 'Art & Design',
'August 1, 2018', '1.2.4', '4.0.3 and up']


['Sketch - Draw & Paint', 'ART_AND_DESIGN', '4.5', '215644', '25M', '50,000,0
00+', 'Free', '0', 'Teen', 'Art & Design', 'June 8, 2018', 'Varies with devic
e', '4.2 and up']


Number of rows: 9658
Number of columns: 13
```

After exploring the data, we can confirm that the number of rows is correct and that the duplicates we previously found have been removed.

Removing Non-English Apps

To check for non-english characters, we can use the function ord() to retreive the ASCII number value for the character. English commonly uses characters with the value 0-127. We can build a function to check for those.

In [19]:
```
def is_english(string):
    non_en_char_count = 0

    for char in string:
        if ord(char) > 127:
            non_en_char_count += 1

    if non_en_char_count > 3:
        return False
    else:
        return True
```

We are testing the function on the following strings:

In [20]:
```
print(is_english('Instagram'))
print(is_english('爱奇艺PPS -《欢乐颂2》电视剧热播'))
print(is_english('Docs To Go™ Free Office Suite'))
print(is_english('Instachat 😜'))
```

```
True
False
True
True
```

It's clear that some primarly english apps use emojis or other characters that don't pass our function that should be kept. For this reason, we will only remove apps that have more than three non-standard english characters.

Changes were made inthe function to include the count for `non_en_char_count`

New lists titled `apple_english` and `android_english` will be used to get the latest cleaned data.

In [21]:
```
apple_english = []
android_english = []
```

We now need to use the function to filter out any non-english apps in both the `android_clean` and `apple_clean`

In [22]:
```
for row in android_clean:
    title = row[0]
    if is_english(title):
        android_english.append(row)

for row in apple_clean:
    title = row[2]
    if is_english(title):
        apple_english.append(row)
```

Checking our work now

In [23]:
```python
explore_data(android_english, 0, 2, True)
print('\n')
explore_data(apple_english, 0, 2, True)
```

```
['Photo Editor & Candy Camera & Grid & ScrapBook', 'ART_AND_DESIGN', '4.1',
'159', '19M', '10,000+', 'Free', '0', 'Everyone', 'Art & Design', 'January 7,
2018', '1.0.0', '4.0.3 and up']


['U Launcher Lite – FREE Live Cool Themes, Hide Apps', 'ART_AND_DESIGN', '4.
7', '87510', '8.7M', '5,000,000+', 'Free', '0', 'Everyone', 'Art & Design',
'August 1, 2018', '1.2.4', '4.0.3 and up']


Number of rows: 9613
Number of columns: 13


['1', '281656475', 'PAC-MAN Premium', '100788224', 'USD', '3.99', '21292', '2
6', '4', '4.5', '6.3.5', '4+', 'Games', '38', '5', '10', '1']


['2', '281796108', 'Evernote - stay organized', '158578688', 'USD', '0', '161
065', '26', '4', '3.5', '8.2.2', '4+', 'Productivity', '37', '5', '23', '1']


Number of rows: 6183
Number of columns: 17
```

Isolating the Free Apps

In this section we would like to isolate the data to only include free applications. After this we will verify the data is clean. If it is, this is the last section we will do prior to analysis. Because of this, our list titles will be called `apple_final` and `android_final`

For ios the price is found in `apple_english index[7]` For andriod, the price is found in `andriod_english index[5]`

In [24]:
```python
apple_final = []
android_final = []
```

In [25]:
```
for app in android_english:
    price = app[7]
    if price == '0':
        android_final.append(app)

for app in apple_english:
    price = app[5]
    if price == '0':
        apple_final.append(app)
```

In [26]:
```
print(len(android_final))

print(len(apple_final))
```

```
8863
3222
```

Analysis

Most Common Apps by Genre

To minimize risks and overhead, our validation strategy for an app idea has three steps:

Build a minimal Android version of the app, and add it to Google Play. If the app has a good response from users, we develop it further. If the app is profitable after six months, we build an iOS version of the app and add it to the App Store. Because our end goal is to add the app on both Google Play and the App Store, we need to find app profiles that are successful in both markets. For instance, a profile that works well for both markets might be a productivity app that makes use of gamification.

Let's begin the analysis by determining the most common genres for each market. For this, we'll need to build frequency tables for a few columns in our datasets.

Genre can be found in the following columns per dataset:

`apple_final index[12]` `android_final index[1]`

First we create a couple functions to assist in the creation of frequency tables.

In [27]:
```
def freq_table(dataset, index):
    freq_table = {}
    total = 0

    for row in dataset:
        key = row[index]
        total += 1
        if key not in freq_table:
            freq_table[key] = 1
        else:
            freq_table[key] += 1

    table_percent = {}
    for key in freq_table:
        percentage = (freq_table[key] / total) * 100
        table_percent[key] = percentage

    return table_percent

def display_table(dataset, index):
    table = freq_table(dataset, index)
    table_display = []
    for key in table:
        key_val_as_tuple = (table[key], key)
        table_display.append(key_val_as_tuple)

    table_sorted = sorted(table_display, reverse = True)
    for entry in table_sorted:
        print(entry[1], ':', entry[0])
```

In [28]:
```
display_table(apple_final, 12)
```

```
Games : 58.16263190564867
Entertainment : 7.883302296710118
Photo & Video : 4.9658597144630665
Education : 3.662321539416512
Social Networking : 3.2898820608317814
Shopping : 2.60707635009311
Utilities : 2.5139664804469275
Sports : 2.1415270018621975
Music : 2.0484171322160147
Health & Fitness : 2.0173805090006205
Productivity : 1.7380509000620732
Lifestyle : 1.5828677839851024
News : 1.3345747982619491
Travel : 1.2414649286157666
Finance : 1.1173184357541899
Weather : 0.8690254500310366
Food & Drink : 0.8069522036002483
Reference : 0.5586592178770949
Business : 0.5276225946617008
Book : 0.4345127250155183
Navigation : 0.186219739292365
Medical : 0.186219739292365
Catalogs : 0.12414649286157665
```

Apple's App Store is dominated by Games which holds a massive 58% of the share given the criteria we have already specified (free, non-duplicated, english only applications). Interesting to see entertainment follow up games in second place with a solid almost 8%.

In [29]:
```
display_table(android_final, 1)
```

```
FAMILY : 18.910075595170937
GAME : 9.725826469592688
TOOLS : 8.462146000225657
BUSINESS : 4.592124562789123
LIFESTYLE : 3.9038700214374367
PRODUCTIVITY : 3.8925871601038025
FINANCE : 3.7007785174320205
MEDICAL : 3.5315355974275078
SPORTS : 3.396141261423897
PERSONALIZATION : 3.317161232088458
COMMUNICATION : 3.226898341419384
HEALTH_AND_FITNESS : 3.0802211440821394
PHOTOGRAPHY : 2.944826808078529
NEWS_AND_MAGAZINES : 2.798149610741284
SOCIAL : 2.6627552747376737
TRAVEL_AND_LOCAL : 2.335552296062281
SHOPPING : 2.245289405393208
BOOKS_AND_REFERENCE : 2.1437436533904997
DATING : 1.8616721200496444
VIDEO_PLAYERS : 1.7939749520478394
MAPS_AND_NAVIGATION : 1.399074805370642
FOOD_AND_DRINK : 1.241114746699763
EDUCATION : 1.1621347173643235
ENTERTAINMENT : 0.9590432133589079
LIBRARIES_AND_DEMO : 0.9364774906916393
AUTO_AND_VEHICLES : 0.9251946293580051
HOUSE_AND_HOME : 0.8236488773552973
WEATHER : 0.8010831546880289
EVENTS : 0.7108202640189552
PARENTING : 0.6544059573507841
ART_AND_DESIGN : 0.6431230960171499
COMICS : 0.6205573733498815
BEAUTY : 0.5979916506826132
```

Google Play's store is very different than the Apple store. Here we see family as the top category, with games at second. However, Games holds a smaller percentage of all apps with a differences of almost a full 50% compared with games in Apple's app store. Let's investigate what family actually means.

In [30]:
```
android_family_apps = []
for apps in android_final:
    genre = apps[1]
    if genre == 'FAMILY':
        android_family_apps.append(apps)

for row in android_family_apps[:4]:
    print(row)
    print('\n')
```

```
['Jewels Crush- Match 3 Puzzle', 'FAMILY', '4.4', '14774', '19M', '1,000,000
+', 'Free', '0', 'Everyone', 'Casual;Brain Games', 'July 23, 2018', '1.9.390
1', '4.0.3 and up']


['Coloring & Learn', 'FAMILY', '4.4', '12753', '51M', '5,000,000+', 'Free',
'0', 'Everyone', 'Educational;Creativity', 'July 17, 2018', '1.49', '4.0.3 an
d up']


['Mahjong', 'FAMILY', '4.5', '33983', '22M', '5,000,000+', 'Free', '0', 'Ever
yone', 'Puzzle;Brain Games', 'August 2, 2018', '1.24.3181', '4.0.3 and up']


['Super ABC! Learning games for kids! Preschool apps', 'FAMILY', '4.6', '2026
7', '46M', '1,000,000+', 'Free', '0', 'Everyone', 'Educational;Education', 'J
uly 16, 2018', '1.1.6.7', '4.1 and up']
```

As seen in the data above, family can typically mean a game designed for children or families. This does help to even the large imbalance of genres between our two lists. However, it is obvious that a practicality is favored far more in the Google Play Store than the App Store.

Most Popular App by Genre in Apple App Store

One way to find out what genres are the most popular (have the most users) is to calculate the average number of installs for each app genre. For the Google Play data set, we can find this information in the Installs column, but this information is missing for the App Store data set. As a workaround, we'll take the total number of user ratings as a proxy, which we can find in the rating_count_tot app which is found on `index[6]`

In [31]:
```
apple_genres = freq_table(apple_final, 12)
print(apple_genres)
```

```
{'Productivity': 1.7380509000620732, 'Weather': 0.8690254500310366, 'Shoppin
g': 2.60707635009311, 'Reference': 0.5586592178770949, 'Finance': 1.117318435
7541899, 'Music': 2.0484171322160147, 'Utilities': 2.5139664804469275, 'Trave
l': 1.2414649286157666, 'Social Networking': 3.2898820608317814, 'Sports': 2.
1415270018621975, 'Health & Fitness': 2.0173805090006205, 'Games': 58.1626319
0564867, 'Food & Drink': 0.8069522036002483, 'News': 1.3345747982619491, 'Boo
k': 0.4345127250155183, 'Photo & Video': 4.9658597144630665, 'Entertainment':
7.883302296710118, 'Business': 0.5276225946617008, 'Lifestyle': 1.58286778398
51024, 'Education': 3.662321539416512, 'Navigation': 0.186219739292365, 'Medi
cal': 0.186219739292365, 'Catalogs': 0.12414649286157665}
```

In [32]:
```
for genre in apple_genres:
    total = 0
    len_genre = 0
    for app in apple_final:
        app_genre = app[12]
        if app_genre == genre:
            num_ratings = float(app[6])
            total += num_ratings
            len_genre += 1
    ave = total / len_genre
    print(genre, ':', ave)
```

```
Productivity : 21028.410714285714
Weather : 52279.892857142855
Shopping : 26919.690476190477
Reference : 74942.11111111111
Finance : 31467.944444444445
Music : 57326.530303030304
Utilities : 18684.456790123455
Travel : 28243.8
Social Networking : 71548.34905660378
Sports : 23008.898550724636
Health & Fitness : 23298.015384615384
Games : 22788.6696905016
Food & Drink : 33333.92307692308
News : 21248.023255813954
Book : 39758.5
Photo & Video : 28441.54375
Entertainment : 14029.830708661417
Business : 7491.117647058823
Lifestyle : 16485.764705882353
Education : 7003.983050847458
Navigation : 86090.33333333333
Medical : 612.0
Catalogs : 4004.0
```

The most popular apps by genre are found in the navigation and social networking genres. This could be influenced heavily by the amount of reviews people have left. After all, we are using number of reviews left as opposed to actual downloads.

Music is also relatively popular and could be a good way to catch a lot of users. Navigation and social media are difficult to get into because the players in the space are so big. Many people don't use anything outside of google maps, apple maps, or waze. Social media is likely even more difficult.

Music could be a great way to go as new ways to interact, share, and listen to music seem to be very popular. Everyone loves sharing their passions, and there is a big user base for it.

Most Popular Apps by Genre on Google Play

To start this we will form a frequency table, using our function, to display what the data would look like using `index[5]` which is the total number of installs.

In [33]:
```
display_table(android_final, 5)
```

```
1,000,000+ : 15.728308699086089
100,000+ : 11.55365000564143
10,000,000+ : 10.549475346947986
10,000+ : 10.199706645605326
1,000+ : 8.383165970890218
100+ : 6.916393997517771
5,000,000+ : 6.826131106848697
500,000+ : 5.562450637481666
50,000+ : 4.772650344127271
5,000+ : 4.513144533453684
10+ : 3.542818458761142
500+ : 3.2494640640866526
50,000,000+ : 2.3017037120613786
100,000,000+ : 2.1324607920568655
50+ : 1.9180864267178157
5+ : 0.7898002933543946
1+ : 0.5077287600135394
500,000,000+ : 0.270788672007221
1,000,000,000+ : 0.2256572266726842
0+ : 0.045131445334536835
0 : 0.011282861333634209
```

As we can see, the data is put into buckets with no exact numbers. For our purposes, this is fine, and we will just take each amount as stated. We will consider an app with 10,000+ installs to have just 10,000 and convert that number to a float so that we can group by genre.

In [49]:
```
for row in android_final:
    users = row[5]
    users = users.replace('+', '')
    users = users.replace(',', '')
    users = users.replace(' ', '')
    row[5] = float(users)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[49], line 3
      1 for row in android_final:
      2     users = row[5]
----> 3     users = users.replace('+', '')
      4     users = users.replace(',', '')
      5     users = users.replace(' ', '')

AttributeError: 'float' object has no attribute 'replace'
```

In [53]:
```
android_cat_freq = freq_table(android_clean, 1)
```

In [54]:
```
for category in android_cat_freq:
    total = 0
    len_category = 0
    for app in android_final:
        app_category = app[1]
        total_users = float(app[5])
        if app_category == category:
            total += total_users
            len_category += 1
    ave = total / len_category
    print(category, ':', ave)
```

```
ART_AND_DESIGN : 1986335.0877192982
AUTO_AND_VEHICLES : 647317.8170731707
BEAUTY : 513151.88679245283
BOOKS_AND_REFERENCE : 8767811.894736841
BUSINESS : 1712290.1474201474
COMICS : 817657.2727272727
COMMUNICATION : 38590577.625874124
DATING : 854028.8303030303
EDUCATION : 1833495.145631068
ENTERTAINMENT : 11640705.88235294
EVENTS : 253542.22222222222
FINANCE : 1387692.475609756
FOOD_AND_DRINK : 1924897.7363636363
HEALTH_AND_FITNESS : 4188821.9853479853
HOUSE_AND_HOME : 1331540.5616438356
LIBRARIES_AND_DEMO : 638503.734939759
LIFESTYLE : 1437816.2687861272
GAME : 15588015.603248259
FAMILY : 3695641.8198090694
MEDICAL : 120550.61980830671
SOCIAL : 23253652.127118643
SHOPPING : 7036877.311557789
PHOTOGRAPHY : 17840110.40229885
SPORTS : 3638640.1428571427
TRAVEL_AND_LOCAL : 13984077.710144928
TOOLS : 10801391.298666667
PERSONALIZATION : 5201482.6122448975
PRODUCTIVITY : 16787331.344927534
PARENTING : 542603.6206896552
WEATHER : 5074486.197183099
VIDEO_PLAYERS : 24727872.452830188
NEWS_AND_MAGAZINES : 9549178.467741935
MAPS_AND_NAVIGATION : 4056941.7741935486
```

The data is heavily skewed towards communication and social platforms. An investigation of those show this information:

In [61]:
```
for row in android_final:
    category = row[1]
    users = row[5]
    name = row[0]
    if category == 'COMMUNICATION':
        print(name, ':', users)
```

```
WhatsApp Messenger : 1000000000.0
Messenger for SMS : 10000000.0
My Tele2 : 5000000.0
imo beta free calls and text : 100000000.0
Contacts : 50000000.0
Call Free – Free Call : 5000000.0
Web Browser & Explorer : 5000000.0
Browser 4G : 10000000.0
MegaFon Dashboard : 10000000.0
ZenUI Dialer & Contacts : 10000000.0
Cricket Visual Voicemail : 10000000.0
TracFone My Account : 1000000.0
Xperia Link™ : 10000000.0
TouchPal Keyboard - Fun Emoji & Android Keyboard : 10000000.0
Skype Lite - Free Video Call & Chat : 5000000.0
My magenta : 1000000.0
Android Messages : 100000000.0
Google Duo - High Quality Video Calls : 500000000.0
Seznam.cz : 1000000.0
```

As shown, the data is blown way out because of apps that have over 1 billion downloads, like WhatsApp. We can assume the same for social media, like in the apple data, because of big players like Facebook, Instagram, Twitter, etc.

Unfortunately, google does not have a music category. This makes it very difficult to compare to our reccomendation of music applications from the apple list. However, we can pull data on similar categories to test the idea of a music, lifestyle, or productivity centered app.

In [69]:
```
for row in android_final:
    category = row[1]
    users = row[5]
    name = row[0]
    if category == 'ENTERTAINMENT':
        print(name, ':', users)
```

```
Complete Spanish Movies : 1000000.0
Pluto TV - It’s Free TV : 1000000.0
Mobile TV : 10000000.0
TV+ : 5000000.0
Digital TV : 5000000.0
Motorola Spotlight Player™ : 10000000.0
Vigo Lite : 5000000.0
Hotstar : 100000000.0
Peers.TV: broadcast TV channels First, Match TV, TNT ... : 5000000.0
The green alien dance : 1000000.0
Spectrum TV : 5000000.0
H TV : 5000000.0
StarTimes - Live International Champions Cup : 1000000.0
Cinematic Cinematic : 1000000.0
MEGOGO - Cinema and TV : 10000000.0
Talking Angela : 100000000.0
DStv Now : 5000000.0
ivi - movies and TV shows in HD : 10000000.0
Radio Javan : 1000000.0
```

In [70]:
```
for row in android_final:
    category = row[1]
    users = row[5]
    name = row[0]
    if category == 'PRODUCTIVITY':
        print(name, ':', users)
```

```
Microsoft Word : 500000000.0
All-In-One Toolbox: Cleaner, Booster, App Manager : 10000000.0
AVG Cleaner – Speed, Battery & Memory Booster : 10000000.0
QR Scanner & Barcode Scanner 2018 : 10000000.0
Chrome Beta : 10000000.0
Microsoft Outlook : 100000000.0
Google PDF Viewer : 10000000.0
My Claro Peru : 5000000.0
Power Booster - Junk Cleaner & CPU Cooler & Boost : 1000000.0
Google Assistant : 10000000.0
Microsoft OneDrive : 100000000.0
Calculator - unit converter : 50000000.0
Microsoft OneNote : 100000000.0
Metro name iD : 10000000.0
Google Keep : 100000000.0
Archos File Manager : 5000000.0
ES File Explorer File Manager : 100000000.0
ASUS SuperNote : 10000000.0
HTC File Manager : 10000000.0
```

Conclusions

In this project, we analyzed data from two large datasets covering the Apple App Store and Google Play Store. After cleaning the data, we determined the frequency and percentage share of each app category to determine what might be popular.

Though there are many directions you could take the data, we determined there could be an opportunity in the music/productivity space. There are many music players and many social medias. However, there is an opportunity to create a unique feature to attract users.

Both apple and android users are big into productivity and family. Features in a music app that could be profitable would be a music app that helps children sleep at night, including a night light on the screen and calming music. Another app that would fit a similar space is a productivity app that can use music to help you focus on a task for a set amount of time/songs. After that amount of time, the user knows it's time to take a break.