# Best Profitable app profiles that generate high revenue

We are working for a company that builds Android and iOS mobile apps, that are free to download and install. The main source of their revenue is in-app ads which depends upon the number of users who use apps - more the users that see and engage with the ads, better the revenue.

The project is to find the best app profiles on Google playstore and Apple store that are likely to attract more users.

Our goal for this project is to analyze sample datasets from Google playstore and Apple store, analyze the datasets, find the best profiles that attracts the users and help our developers understand what type of apps are likely to attract more users on Google Playstore and the Apple store and help them in developing such apps.

Let's open our datasets and convert them into lists of lists and start exploring the data.

In [1]:
```
from csv import reader

#The AppleStore data
open_file = open('AppleStore.csv')
reader_file = reader(open_file)
apple_apps = list(reader_file)
ios_header = apple_apps[0]
ios_data = apple_apps[1:]

#The Google Playstore data
open_file1 = open('googleplaystore.csv')
reader_file1 = reader(open_file1)
android_apps =list(reader_file1)
android_header = android_apps[0]
android_data = android_apps[1:]
```

## Exploring the Datasets

As we have two datasets with 7000 and 10000 apps, to explore our datasets in a convenient way, let's create a function named `explore_data()` that we can use repeatedly to explore rows in a more readable way. Let's also add an option for our function to show the number of rows and columns for any data set.

In [2]:
```python
def explore_data(dataset, start, end, rows_and_columns = False):
    dataset_slice = dataset[start:end]
    for row in dataset_slice:
        print(row) #Print each row in sliced dataset
        print('\n') #This leaves an empty line

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

Let's print a few observations in both the datasets along with their headers

In [3]:
```python
#To print a few observations in android apps
print(android_header)
print('\n')
explore_data(android_data, 0, 4, True)

#To print a few observations in iOS apps
print(ios_header)
print('\n')
explore_data(ios_data, 0, 4, True)
```

```
['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type', 'Pri
ce', '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', 'Ja
nuary 7, 2018', '1.0.0', '4.0.3 and up']


['Coloring book moana', 'ART_AND_DESIGN', '3.9', '967', '14M', '500,000
+', 'Free', '0', 'Everyone', 'Art & Design;Pretend Play', 'January 15, 20
18', '2.0.0', '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 & Des
ign', 'August 1, 2018', '1.2.4', '4.0.3 and up']
```

It is found that `Google playstore` has 10841 observations and 13 columns. Please check https://www.kaggle.com/lava18/google-play-store-apps/home (https://www.kaggle.com/lava18/google-play-store-apps/home) to know more about the columns. Of all the 13 columns, the columns that could help us for our analysis are, `['App', 'Category', 'Rating', 'Reviews', 'Installs', 'Type', 'Price', 'Genres']`

Similarly, `Apple Store` has 7197 observations and 16 columns. Please check https://www.kaggle.com/ramamet4/app-store-apple-data-set-10k-apps/home (https://www.kaggle.com/ramamet4/app-store-apple-data-set-10k-apps/home) for more information oabout the columns. Of all the 16 columns, the columns that could help us for our analysis are, `['id', 'track_name', 'currency', 'price', 'rating_count_tot', 'user_rating', 'cont_rating', 'prime_genre']`

In [4]:
```
#To check if all the observations in android apps have all the values or if a
for row in android_apps:
    header_length = len(android_header)
    row_length = len(row)
    if row_length != header_length:
        print(row)
        print('\n')
        print(android_apps.index(row))
```

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


10473
```

## Removing the wrong data

We found that data point at index# 10473 has an incorrect data. So we need to remove it.

In [5]:
```
#Index 10473 has incorrected data. So we should remove this
del android_apps[10473]
```

In [6]:
```
#To check if all the observations in iOS apps have all the values or if any v
for row in apple_apps:
    header_length = len(ios_header)
    row_length = len(row)
    if row_length != header_length:
        print(row)
        print('\n')
        print(apple_apps.index(row))

#It is found that no value is missed in ios_data.
```

## Checking for Duplicate entries

As per the discussion on Google Play data, it was found that there are a few apps with multiple entries in the dataset. For analyzing, one entry would be enough. So, let's check if we have any duplicates in Google Play dataset.

In [7]:

```
#Checking for Instagram duplicate entries and will check for the entire datas
for app in android_data: #Considering our dataset without header_row
    name = app[0] #The index of App Name in our dataset is '0'
    if name == "Instagram": #Checking if the name of the app is Instagram
        print(app) #If the name is Instagram, print all the values of that ro
```

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

## Finding all the duplicate entries in the dataset

It is confirmed that we have a few duplicate entries. So let's find them by writing a piece of code.

In [8]:

```
#Let's check for duplicate entries

unique_apps = [] # It will have all the unique app names
duplicate_apps = [] #It will have all the app names with multiple entries in

for app in android_apps[1:]: #Because we removed the row in this dataset
    name = app[0]
    if name in unique_apps:
        duplicate_apps.append(name)#Adds name of the app as duplicate entries
    else:
        unique_apps.append(name)#Adds name of the app as unique entries

print("# of unique_apps : ", len(unique_apps))
print('\n')
print("# of duplicate_apps : ", len(duplicate_apps))
print('\n')
print("Examples of duplicate_apps : ", duplicate_apps[:10])
```

```
# of unique_apps :  9659


# of duplicate_apps :  1181


Examples of duplicate_apps :  ['Quick PDF Scanner + OCR FREE', 'Box', 'Goog
le My Business', 'ZOOM Cloud Meetings', 'join.me - Simple Meetings', 'Box',
'Zenefits', 'Google Ads', 'Google My Business', 'Slack']
```

## Remove the Duplicate entries

We don't need to count certain apps more than once when we analyze data, so we need to remove the duplicate entries and keep only one entry per app. One thing we could do is remove the duplicate rows randomly, but let's try to find a better way to remove them.

If we examine the rows we printed for Instagram app, the main difference happens on the fourth position of each row, which corresponds to the number of reviews. The different numbers shows that the data was collected at different times. We can use this to build a criterion for keeping rows. Let's not remove the rows randomly, instead but we'll keep the rows that have the highest number of reviews because the higher the number of reviews, the more reliable the ratings are and it should be the latest version of our dataset.

In [9]:
```
#To cross check the unique app names
print("Expected Length :" , len(android_apps[1:]) - 1181) #Header is not cons
```

```
Expected Length : 9659
```

To remove the duplicate entries, we will

- Create a dictionary where each key is the unique app name and its corresponding value will be the highest number of the reviews of that app.
- Use the dictionary to create a new data set, which will have only one entry per app (and we only select the apps with the highest number of reviews)

In [10]:
```
#Create a dictionary that will have unique app names with value as number of

reviews_max = {}

for app in android_apps[1:]:
    name = app[0]
    n_reviews = float(app[3])
    if name in reviews_max and reviews_max[name] < n_reviews:# To update if c
        reviews_max[name] = n_reviews
    elif name not in reviews_max: #To add if it doesn't exists and give it's
        reviews_max[name] = n_reviews

len(reviews_max) #this should be equal to the Expected length in the previous
```

Out[10]:
```
9659
```

By using the `reviews_max` dictionary, let's make a cleaned dataset with no duplicates and with number of reviews as the latest. This can be done as mentioned below:

- Let's start by creating two empty lists - `android_clean` and `already_added` .
- Loop through our dataset without header and isolate the name of the app and the number of reviews, which is at index# 3.
- Now we add the current row(app) to `android_clean` and app name to `already_added` if:
    - The number of reviews of the current app matches the number of reviews of that app as described in the `reviews_max` dictionary and

- The name of the app is not already in the `already_added` list. We need to add this supplementary condition to account for those cases where the highest number of reviews of a duplicate app is the same for more than one entry (for example, the Box app has three entries, and the number of reviews is the same). If we just check for `reviews_max[name] == n_reviews` , we'll still end up with duplicate entries for some apps.

In [11]:
```
#To remove the duplicate rows
android_clean = []
already_added = []

for app in android_apps[1:]:
    name = app[0]
    n_reviews = float(app[3])
    if reviews_max[name] == n_reviews and name not in already_added:
        android_clean.append(app)
        already_added.append(name)
```

Let's check our cleaned dataset, `android_clean` which should have the expected length as mentioned two cells above i.e. 9659

In [12]:
```
#Crosschecking the new cleaned dataset
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 & Desig
n', 'August 1, 2018', '1.2.4', '4.0.3 and up']


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


Number of rows: 9659
Number of columns: 13
```

## Remove non-english Apps

As we are working for English speaking audience, we can remove the apps that are not directed towards english. Let's check a few observations where we have non - english apps.

In [13]:
```
#Checking for non-english apps
print(apple_apps[814][1])
print(apple_apps[6732][1])
print('\n')
```

```
爱奇艺PPS -《欢乐颂2》电视剧热播
【脱出ゲーム】絶対に最後までプレイしないで 〜謎解き＆ブロックパズル〜
```

We're not interested in keeping these kind of apps, so we'll remove them. One way to go about this is to remove each app whose name contains a symbol that is not commonly used in English text — English text usually includes letters from the English alphabet, numbers composed of digits from 0 to 9, punctuation marks (., !, ?, ;, etc.), and other symbols (+, *, /, etc.).

All these characters that are specific to English texts are encoded using the *ASCII* standard. Each *ASCII* character has a corresponding number between *0 and 127* associated with it and we can take advantage of that to build a function that checks an app name and tells us whether it contains non-ASCII characters.

We built this function below, and we use the built-in *ord()* function to find out the corresponding encoding number of each character.

In [14]:
```
#Function to detect if app name is english

def is_english(string):
    for i in string:
        if ord(i) > 127:
            return False
    return True

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
False
False
```

From the above code cell's result, it is found that the third and fourth, though they are english apps, because of the usage of smileys or words like ™ resulted them as non english apps. Let's check their *ASCII* values.

In [15]:
```
#To check ASCII values of 😜 and ™
print(ord('™'))
print(ord('😜'))
```

```
8482
128540
```

As these values fall out of ASCII range which is >127, these are not considered as english apps as per our function. Because of this, we'll remove useful apps if we use the function in its current form as many English apps will be incoreectly labelled as non-English.

To minimize the impact of data loss, we'll only remove an app if its name has more than three characters with corresponding numbers falling outside the ASCII range. It means all English apps with upto three non-ASCII characters will still be labelled as English. Let's change our function to work for the same.

In [16]:
```
def is_english(string):
    non_ASCII_char = 0
    for i in string: # Loop to check the # of non ASCII chars in a string
        if ord(i) > 127:
            non_ASCII_char += 1

    if non_ASCII_char > 3: #Loop to check the app
        return False
    else:
        return True

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

```
True
True
False
```

The function is still not perfect and very few non-English apps might get past our filter, because an app with two non-ASCII characters will be removed stating that they are not English apps. But for now let's proceed with this.

Let's check for each app name in our `android_clean` (without header) and `apple_apps` (with header) and append the entire row to the new list if an app belongs to English.

In [17]:
```
android_english = []
ios_english = []

for i in android_clean:
    name = i[0]
    if is_english(name):
        android_english.append(i)

for i in apple_apps[1:]:
    name = i[1] #Because in apple_apps index# 1 corresponds to the app name
    if is_english(name):
        ios_english.append(i)

explore_data(android_english, 0, 4, True)
print('\n')
explore_data(ios_english, 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 & Desig
n', 'August 1, 2018', '1.2.4', '4.0.3 and up']


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


['Pixel Draw - Number Art Coloring Book', 'ART_AND_DESIGN', '4.3', '967',
'2.8M', '100,000+', 'Free', '0', 'Everyone', 'Art & Design;Creativity', 'Ju
ne 20, 2018', '1.1', '4.4 and up']


Number of rows: 9614
Number of columns: 13


['284882215', 'Facebook', '389879808', 'USD', '0.0', '2974676', '212', '3.
5', '3.5', '95.0', '4+', 'Social Networking', '37', '1', '29', '1']


['389801252', 'Instagram', '113954816', 'USD', '0.0', '2161558', '1289',
'4.5', '4.0', '10.23', '12+', 'Photo & Video', '37', '0', '29', '1']


['529479190', 'Clash of Clans', '116476928', 'USD', '0.0', '2130805', '57
9', '4.5', '4.5', '9.24.12', '9+', 'Games', '38', '5', '18', '1']


Number of rows: 6183
Number of columns: 16
```

We are now left with 9614 Android apps and 6183 apple apps. Till now as part of data cleaning process, we

```
* Removed inaccurate or wrong data
* Removed duplicate app entries with some proper approach
* Removed non-English apps
```

As we mentioned in the introduction, our caompany will only build apps that are free to download and install. Our main source of revenue is in-app ads. Our datasets contain both free and non-free apps. Let's isolate only the free apps for our analysis.

This will be our last step in the data cleaning process.

In [18]:

```
#print(len(android_english)) -- 9614
#print(len(ios_english)) -- 6183

android_final = []
ios_final = []

for app in android_english:
    price = app[7] #Index 7 in android header
    if price == '0':
        android_final.append(app)

for app in ios_english:
    price = app[4] #Index 4 in ios header
    if price == '0.0':
        ios_final.append(app)

print(len(android_final))
print(len(ios_final))
```

```
8864
3222
```

# Cleaned datasets

Now that we have a cleaned datasets with 8864 and 3222 for android and ios respectively. Let's analyze our datasets with various possible ways.

As we mentioned in the introduction, our aim is to determine the kinds of apps that are likely to attract more # of the users because our revenue is highly influenced by the number of people using our apps.

To minimize risks and overhead, our validation strategy for an app idea is comprised of 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.

Our ultimate goal is to ad dour app on both Google Play and the App Store. We need to find app profiles that are successful on both markets. For instance, a profile that works well for both markets might be a productivity app that makes use of gamification, an application of typical elements of game playing.

Let's begin the analysis by getting a sense of the most common genres for each market. For this, we'll build a frequency table for the `prime_genre` column of the App Store data set, and the `Genres and Category` columns of the Google Play data set.

Let's build two functions to analyze the frequency tables,

```
* One function to generate frequency tables that show percentages
* Another function we can use to display the percentages in a descending
order
```

In [19]:

```
#Function to generate frequency table that show percentages

def freq_table(dataset, index):
    table = {}
    total = 0

    for row in dataset: #each row is a list in a dataset
        total += 1
        value = row[index] # To choose that particular index to generate freq
        if value in table:
            table[value] += 1
        else:
            table[value] = 1


# To calculate the percentage of each Genre/Category
    table_percentages = {}

    for key in table:
        percentage = (table[key]/total) * 100
        table_percentages[key] = percentage

    return table_percentages

#Function to display the percentages
def display_table(dataset, index):
    table = freq_table(dataset, index) #Table will have `Genre` as key with i
    table_display = [] #This is a list of tuples
    for i in table:
        key_value_tuple = (table[i], i) #Coverting into tuple
        table_display.append(key_value_tuple)

#To sort the values in descending order
    table_sorted = sorted(table_display, reverse = True)# Descending order
    for entry in table_sorted: #tuple will be in the form of (58.1234 , Games
        print(entry[1], ':', entry[0]) #To change the above form
```

Let's start by examining the frequency table for the `prime_genre` column of the App Store data set.

In [20]:
```
display_table(ios_final, -5)
```

```
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
```

From the above values, we can conclude that among the free English apps, `Games` constitute for more than 58%, then `Entertainment` with nearly 8%, followed by `Photo & Video` which is close to 5%. Only 3.66% apps are designed for `Education` and 3.28% apps are designed for `Social Networking` in the App Store Dataset.

The general conclusion is that App Store (at least the part containing free English apps) is dominated by apps that are designed for fun (games, entertainment, photo and video, social networking, sports, music, etc.) constituting almost 75 % - 80% of the App Store, while apps with practical purposes such as, education, shopping, utilities, productivity, lifestyle, etc. are rare.

However, the fact that fun apps are the most numerous but that doesn't also imply that they also have the greatest number of users — the demand might not be the same as the percentage seems to be.

In [21]:
```
display_table(android_final, 1) #Category in Google Play dataset
```

```
FAMILY : 18.907942238267147
GAME : 9.724729241877256
TOOLS : 8.461191335740072
BUSINESS : 4.591606498194946
LIFESTYLE : 3.9034296028880866
PRODUCTIVITY : 3.892148014440433
FINANCE : 3.7003610108303246
MEDICAL : 3.531137184115524
SPORTS : 3.395758122743682
PERSONALIZATION : 3.3167870036101084
COMMUNICATION : 3.2378158844765346
HEALTH_AND_FITNESS : 3.0798736462093865
PHOTOGRAPHY : 2.944494584837545
NEWS_AND_MAGAZINES : 2.7978339350180503
SOCIAL : 2.6624548736462095
TRAVEL_AND_LOCAL : 2.33528880866426
SHOPPING : 2.2450361010830324
BOOKS_AND_REFERENCE : 2.1435018050541514
DATING : 1.861462093862816
VIDEO_PLAYERS : 1.7937725631768955
MAPS_AND_NAVIGATION : 1.3989169675090252
FOOD_AND_DRINK : 1.2409747292418771
EDUCATION : 1.1620036101083033
ENTERTAINMENT : 0.9589350180505415
LIBRARIES_AND_DEMO : 0.9363718411552346
AUTO_AND_VEHICLES : 0.9250902527075812
HOUSE_AND_HOME : 0.8235559566787004
WEATHER : 0.8009927797833934
EVENTS : 0.7107400722021661
PARENTING : 0.6543321299638989
ART_AND_DESIGN : 0.6430505415162455
COMICS : 0.6204873646209386
BEAUTY : 0.5979241877256317
```

The landscape seems significantly different on Google Play. There are not that many apps designed for fun, and it seems that a good number of apps are designed for practical purposes (family, tools, business, lifestyle, productivity, etc.). However, if we investigate this further, we can see that the family category (which accounts for almost 19% of the apps) means mostly games for kids.

Even so, practical apps seem to have a better representation on Google Play compared to App Store. This picture is also confirmed by the frequency table we see for the Genres column:

```
In [22]: display_table(android_final, -4) #Genres
```

```
Tools : 8.449909747292418
Entertainment : 6.069494584837545
Education : 5.347472924187725
Business : 4.591606498194946
Productivity : 3.892148014440433
Lifestyle : 3.892148014440433
Finance : 3.7003610108303246
Medical : 3.531137184115524
Sports : 3.463447653429603
Personalization : 3.3167870036101084
Communication : 3.2378158844765346
Action : 3.1024368231046933
Health & Fitness : 3.0798736462093865
Photography : 2.944494584837545
News & Magazines : 2.7978339350180503
Social : 2.6624548736462095
Travel & Local : 2.3240072202166067
Shopping : 2.2450361010830324
Books & Reference : 2.1435018050541514
```

The difference between the Genres and the `Category` columns is not crystal clear, but one thing we can notice is that the `Genres` column is much more granular, it has more categories. We're only looking for the bigger picture at the moment, so we'll only work with the Category column moving forward.

Up to this point, we found that the `App Store` is dominated by apps designed for `fun`, while `Google Play` shows a more balanced landscape of both practical and for-fun apps. Now we'd like to get an idea about the kind of apps that have most users.

## Most Popular Apps by Genre

One way to find out what genres are the most popular or 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 for the `App Store` data set this information is missing. 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.

Below, we calculate the average number of user ratings per app genre on the App Store:

In [23]:
```
ios_genres = freq_table(ios_final, -5)#last 5th position is genres column
#ios_genres is a dictionary with Genres and their percentages in App store

for genre in ios_genres:
    total = 0
    len_genre = 0
    for i in ios_final:
        genre_app = i[-5] #app genre will be stored in genre_app
        if genre_app == genre:
            n_ratings = float(i[5]) #ratings_count_tot column is at index 5
            total += n_ratings
            len_genre += 1
    avg_n_ratings = total/len_genre
    print(genre, ':', avg_n_ratings)
```

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

In [24]:
```
print(ios_header) #Index 5 is 'rating_count_tot' and Index 11(Position -5) i
```

```
['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']
```

From the above results, it is very clear that on an average `Navigation` genre has more number of ratings in `App Store` data. Let's check the apps that comes under 'Navigation' genre along with their ratings.

In [25]:

```
#Checking for apps that comes under Navigation genre along with their user ro
for app in ios_final:
    if app[-5] == "Navigation":
        print(app[1], ':', app[5])
print('\n')
#Checking for apps that comes under Social Networking genre along with their
for app in ios_final:
    if app[-5] == "Social Networking":
        print(app[1], ':', app[5])
```

```
Waze - GPS Navigation, Maps & Real-time Traffic : 345046
Google Maps - Navigation & Transit : 154911
Geocaching® : 12811
CoPilot GPS – Car Navigation & Offline Maps : 3582
ImmobilienScout24: Real Estate Search in Germany : 187
Railway Route Search : 5


Facebook : 2974676
Pinterest : 1061624
Skype for iPhone : 373519
Messenger : 351466
Tumblr : 334293
WhatsApp Messenger : 287589
Kik : 260965
ooVoo – Free Video Call, Text and Voice : 177501
TextNow - Unlimited Text + Calls : 164963
Viber Messenger – Text & Call : 164249
Followers - Social Analytics For Instagram : 112778
MeetMe - Chat and Meet New People : 97072
We Heart It - Fashion, wallpapers, quotes, tattoos : 90414
InsTrack for Instagram - Analytics Plus More : 85535
Tango - Free Video Call, Voice and Chat : 75412
LinkedIn : 71856
Match™ - #1 Dating App. : 60659
Skype for iPad : 60163
POF - Best Dating App for Conversations : 52642
Timehop : 49510
Find My Family, Friends & iPhone - Life360 Locator : 43877
Whisper - Share, Express, Meet : 39819
Hangouts : 36404
LINE PLAY - Your Avatar World : 34677
WeChat : 34584
Badoo - Meet New People, Chat, Socialize. : 34428
Followers + for Instagram - Follower Analytics : 28633
GroupMe : 28260
Marco Polo Video Walkie Talkie : 27662
Miitomo : 23965
SimSimi : 23530
Grindr - Gay and same sex guys chat, meet and date : 23201
Wishbone - Compare Anything : 20649
imo video calls and chat : 18841
After School - Funny Anonymous School News : 18482
Quick Reposter - Repost, Regram and Reshare Photos : 17694
Weibo HD : 16772
Repost for Instagram : 15185
```

```
Live.me – Live Video Chat & Make Friends Nearby : 14724
Nextdoor : 14402
Followers Analytics for Instagram - InstaReport : 13914
YouNow: Live Stream Video Chat : 12079
FollowMeter for Instagram - Followers Tracking : 11976
LINE : 11437
eHarmony™ Dating App - Meet Singles : 11124
Discord - Chat for Gamers : 9152
QQ : 9109
Telegram Messenger : 7573
Weibo : 7265
Periscope - Live Video Streaming Around the World : 6062
Chat for Whatsapp - iPad Version : 5060
QQ HD : 5058
Followers Analysis Tool For Instagram App Free : 4253
live.ly - live video streaming : 4145
Houseparty - Group Video Chat : 3991
SOMA Messenger : 3232
Monkey : 3060
Down To Lunch : 2535
Flinch - Video Chat Staring Contest : 2134
Highrise - Your Avatar Community : 2011
LOVOO - Dating Chat : 1985
PlayStation®Messages : 1918
BOO! - Video chat camera with filters & stickers : 1805
Qzone : 1649
Chatous - Chat with new people : 1609
Kiwi - Q&A : 1538
GhostCodes - a discovery app for Snapchat : 1313
Jodel : 1193
FireChat : 1037
Google Duo - simple video calling : 1033
Fiesta by Tango - Chat & Meet New People : 885
Google Allo — smart messaging : 862
Peach — share vividly : 727
Hey! VINA - Where Women Meet New Friends : 719
Battlefield™ Companion : 689
All Devices for WhatsApp - Messenger for iPad : 682
Chat for Pokemon Go - GoChat : 500
IAmNaughty – Dating App to Meet New People Online : 463
Qzone HD : 458
Zenly - Locate your friends in realtime : 427
League of Legends Friends : 420
豆瓣 : 407
Candid - Speak Your Mind Freely : 398
知乎 : 397
Selfeo : 366
Fake-A-Location Free ™ : 354
Popcorn Buzz - Free Group Calls : 281
Fam — Group video calling for iMessage : 279
QQ International : 274
Ameba : 269
SoundCloud Pulse: for creators : 240
Tantan : 235
Cougar Dating & Life Style App for Mature Women : 213
Rawr Messenger - Dab your chat : 180
WhenToPost: Best Time to Post Photos for Instagram : 158
```

```
Inke—Broadcast an amazing life : 147
Mustknow - anonymous video Q&A : 53
CTFxCmoji : 39
Lobi : 36
Chain: Collaborate On MyVideo Story/Group Video : 35
botman - Real time video chat : 7
BestieBox : 0
MATCH ON LINE chat : 0
niconico ch : 0
LINE BLOG : 0
bit-tube - Live Stream Video Chat : 0
```

It is clear that on an average `navigation` apps have the highest number of user reviews, but this figure is greatly influenced by `Waze and Google Maps` , which almost have close to half a million user reviews together.

Similarly, the second highest number of user reviews on an average is `Social Networking` apps. But `Facebook, Pinterest, Skype, Messenger, Tumblr, Whatsapp, Kik` combinely has almost has 5.6 Million user ratings.

Same applies to music apps, where a few big players like Pandora, Spotify, and Shazam heavily influence the average number.

Our aim is to find popular genres, but `navigation, social networking or music` apps might seem more popular than they really are. The average number of ratings seems to be skewed by very few apps which have hundreds of thousands of user ratings, while the other apps may struggle to get past the 10,000+ . We could get a better picture by removing these extremely popular apps(outliers) for each genre and then rework the averages to get better ideas on the best app profiles.

When we go through the other genres, say `Reference` has 74942 user rating reviews on an average. Let's check `Reference` genre for detailed information.

In [26]:
```
#Checking for apps that comes under Reference genre along with their user rat
for app in ios_final:
    if app[-5] == "Reference":
        print(app[1], ':', app[5])
```

```
Bible : 985920
Dictionary.com Dictionary & Thesaurus : 200047
Dictionary.com Dictionary & Thesaurus for iPad : 54175
Google Translate : 26786
Muslim Pro: Ramadan 2017 Prayer Times, Azan, Quran : 18418
New Furniture Mods - Pocket Wiki & Game Tools for Minecraft PC Edition : 17
588
Merriam-Webster Dictionary : 16849
Night Sky : 12122
City Maps for Minecraft PE - The Best Maps for Minecraft Pocket Edition (MC
PE) : 8535
LUCKY BLOCK MOD ™ for Minecraft PC Edition - The Best Pocket Wiki & Mods In
staller Tools : 4693
GUNS MODS for Minecraft PC Edition - Mods Tools : 1497
Guides for Pokémon GO - Pokemon GO News and Cheats : 826
WWDC : 762
Horror Maps for Minecraft PE - Download The Scariest Maps for Minecraft Poc
ket Edition (MCPE) Free : 718
VPN Express : 14
Real Bike Traffic Rider Virtual Reality Glasses : 8
教えて!goo : 0
Jishokun-Japanese English Dictionary & Translator : 0
```

When we check the `Reference` genre, `Bible` alone has almost a million user rating reviews followed by `Dictionary` with 0.25 million user rating reviews and then `Google Translate` by 26000 user rating reviews.

If we can take another popular book and turn it into an app where we could add different features besides the raw version of the book. This might include daily quotes from the book, an audio version of the book, quizzes about the book, etc. On top of that, we could also embed a `dictionary` and a `Google Translate` within the app, so that users don't need to exit our app to look up words in an external app or check for translation if needed.

This idea seems to fit well, because we know that `App Store` is dominated by for-fun apps unlike `Google Play` dataset. This suggests the market might be a bit saturated with for-fun apps and a practical app might have more chance to stand out among the huge number of apps on the App Store.

Other genres that seem popular include weather, book, food and drink, or finance. The book genre seem to overlap a bit with the app idea we described above, but the other genres don't seem too interesting to us. The suggestions may be like,

```
*Weather apps — people generally don't spend too much time in-app, and t
he chances of making profit from in-app adds are low and also, getting r
eliable live weather data may require us to connect our apps to non-free
APIs.

*Food and drink — examples here include Starbucks, Dunkin' Donuts, McDon
ald's, etc. So making a popular food and drink app requires actual cooki
ng and a delivery service, which is outside the scope of our company.

*Finance apps — these apps involve banking, paying bills, money transfe
r, etc. Building a finance app requires domain knowledge, and we don't w
ant to hire a finance expert just to build an app.
```

Now let's analyze the Google Play market a bit.

## Most Popular apps by Genre in Google Play market

Unlike the number of user ratings in App Store, we have number of installs in Google Play dataset.So based upon the number of installs let's try to find the popularity of the genre. However the installs column doesn't provide us a precise number - we can see that most values are open-ended (100+, 1,000+, 5,000+, etc.)

In [27]:
```python
#Installs column
display_table(android_final, 5)
```

```
1,000,000+ : 15.726534296028879
100,000+ : 11.552346570397113
10,000,000+ : 10.548285198555957
10,000+ : 10.198555956678701
1,000+ : 8.393501805054152
100+ : 6.915613718411552
5,000,000+ : 6.825361010830325
500,000+ : 5.561823104693141
50,000+ : 4.7721119133574
5,000+ : 4.512635379061372
10+ : 3.5424187725631766
500+ : 3.2490974729241873
50,000,000+ : 2.3014440433213
100,000,000+ : 2.1322202166064983
50+ : 1.917870036101083
5+ : 0.78971119133574
1+ : 0.5076714801444043
500,000,000+ : 0.2707581227436823
1,000,000,000+ : 0.22563176895306858
0+ : 0.04512635379061372
0 : 0.01128158844765343
```

One problem with this data is that it is not precise. For instance, we don't know whether an app with 100,000+ installs has 100,000 installs, 200,000, or 350,000. However, we don't need very precise data for our purposes — we only want to get an idea which app genres attract the most users and we don't need perfect precision with respect to the number of users.

We're going to leave the numbers as they are, which means that let's consider that an app with 100,000+ installs has 100,000 installs, and an app with 1,000,000+ installs has 1,000,000 installs, and so on.

To perform computations, however, we'll need to convert each install number to float — this means that we need to remove the commas and the plus characters, otherwise the conversion will fail and raise an error. We'll do this directly in the loop below, where we also compute the average number of installs for each genre (category).

In [30]:
```
android_categories = freq_table(android_final, 1)

for category in android_categories:
    total = 0
    len_category = 0
    for app in android_final:
        app_category = app[1]
        if app_category == category:
            n_installs = app[5]
            n_installs = n_installs.replace(',','')
            n_installs = n_installs.replace('+', '')
            total += float(n_installs)
            len_category += 1
    avg_n_installs = total / len_category
    print(category, ':', avg_n_installs)
```

```
PHOTOGRAPHY : 17840110.40229885
EVENTS : 253542.22222222222
BOOKS_AND_REFERENCE : 8767811.894736841
SOCIAL : 23253652.127118643
PRODUCTIVITY : 16787331.344927534
AUTO_AND_VEHICLES : 647317.8170731707
TRAVEL_AND_LOCAL : 13984077.710144928
ART_AND_DESIGN : 1986335.0877192982
MEDICAL : 120550.61980830671
HOUSE_AND_HOME : 1331540.5616438356
MAPS_AND_NAVIGATION : 4056941.7741935486
ENTERTAINMENT : 11640705.88235294
FOOD_AND_DRINK : 1924897.7363636363
PERSONALIZATION : 5201482.6122448975
GAME : 15588015.603248259
WEATHER : 5074486.197183099
HEALTH_AND_FITNESS : 4188821.9853479853
COMMUNICATION : 38456119.167247385
SPORTS : 3638640.1428571427
BUSINESS : 1712290.1474201474
BEAUTY : 513151.88679245283
COMICS : 817657.2727272727
SHOPPING : 7036877.311557789
LIFESTYLE : 1437816.2687861272
LIBRARIES_AND_DEMO : 638503.734939759
VIDEO_PLAYERS : 24727872.452830188
FINANCE : 1387692.475609756
NEWS_AND_MAGAZINES : 9549178.467741935
TOOLS : 10801391.298666667
EDUCATION : 1833495.145631068
PARENTING : 542603.6206896552
FAMILY : 3695641.8198090694
DATING : 854028.8303030303
```

On average, communication apps have the most installs: 38,456,119. This number is heavily skewed up by a few apps that have over one billion installs (WhatsApp, Facebook Messenger, Skype, Google Chrome, Gmail, and Hangouts), and a few others with over 100 and 500 million installs.

Let's check for `Communication` category:

In [31]:
```
for app in android_final:
    if app[1] == 'COMMUNICATION' and (app[5] == '1,000,000,000+'
                                      or app[5] == '500,000,000+'
                                      or app[5] == '100,000,000+'):
        print(app[0], ':', app[5])
```

```
WhatsApp Messenger : 1,000,000,000+
imo beta free calls and text : 100,000,000+
Android Messages : 100,000,000+
Google Duo - High Quality Video Calls : 500,000,000+
Messenger – Text and Video Chat for Free : 1,000,000,000+
imo free video calls and chat : 500,000,000+
Skype - free IM & video calls : 1,000,000,000+
Who : 100,000,000+
GO SMS Pro - Messenger, Free Themes, Emoji : 100,000,000+
LINE: Free Calls & Messages : 500,000,000+
Google Chrome: Fast & Secure : 1,000,000,000+
Firefox Browser fast & private : 100,000,000+
UC Browser - Fast Download Private & Secure : 500,000,000+
Gmail : 1,000,000,000+
Hangouts : 1,000,000,000+
Messenger Lite: Free Calls & Messages : 100,000,000+
Kik : 100,000,000+
KakaoTalk: Free Calls & Text : 100,000,000+
Opera Mini - fast web browser : 100,000,000+
Opera Browser: Fast and Secure : 100,000,000+
Telegram : 100,000,000+
Truecaller: Caller ID, SMS spam blocking & Dialer : 100,000,000+
UC Browser Mini -Tiny Fast Private & Secure : 100,000,000+
Viber Messenger : 500,000,000+
WeChat : 100,000,000+
Yahoo Mail – Stay Organized : 100,000,000+
BBM - Free Calls & Messages : 100,000,000+
```

Let's try to remove all the `communication` apps that have over 100 million installs, the average would be reduced roughly ten times.

In [35]:
```
apps_under_100million_installs = []

for app in android_final:
    n_installs = app[5]
    n_installs = n_installs.replace(',','')
    n_installs = n_installs.replace('+', '')
    if (app[1] == 'COMMUNICATION') and (float(n_installs) < 100000000):
        apps_under_100million_installs.append(float(n_installs))

sum(apps_under_100million_installs) / len(apps_under_100million_installs)
```

Out[35]:
```
3603485.3884615386
```

We see the same pattern for the video players category, which is the runner-up with 24,727,872 installs. The market is dominated by apps like Youtube, Google Play Movies & TV, or MX Player. The pattern is repeated for social apps (where we have giants like Facebook, Instagram, Google+, etc.), photography apps (Google Photos and other popular photo editors), or productivity apps (Microsoft Word, Dropbox, Google Calendar, Evernote, etc.).

Again, the main concern is that these app genres might seem more popular than they really are. Moreover, these niches seem to be dominated by a few giants who are hard to compete against.

The game genre seems pretty popular, but previously we found out this part of the market seems a bit saturated, so we'd like to come up with a different app recommendation if possible.

The books and reference genre looks fairly popular as well, with an average number of installs of 8,767,811. It's interesting to explore this in more depth, since we found this genre has some potential to work well on the App Store, and our aim is to recommend an app genre that shows potential for being profitable on both the App Store and Google Play.

Let's take a look at some of the apps from this genre and their number of installs.

In [36]:

```
for app in android_final:
    if app[1] == 'BOOKS_AND_REFERENCE':
        print(app[0], ':', app[5])
```

```
E-Book Read - Read Book for free : 50,000+
Download free book with green book : 100,000+
Wikipedia : 10,000,000+
Cool Reader : 10,000,000+
Free Panda Radio Music : 100,000+
Book store : 1,000,000+
FBReader: Favorite Book Reader : 10,000,000+
English Grammar Complete Handbook : 500,000+
Free Books - Spirit Fanfiction and Stories : 1,000,000+
Google Play Books : 1,000,000,000+
AlReader -any text book reader : 5,000,000+
Offline English Dictionary : 100,000+
Offline: English to Tagalog Dictionary : 500,000+
FamilySearch Tree : 1,000,000+
Cloud of Books : 1,000,000+
Recipes of Prophetic Medicine for free : 500,000+
ReadEra – free ebook reader : 1,000,000+
Anonymous caller detection : 10,000+
Ebook Reader : 5,000,000+
```

The book and reference genre includes a variety of apps such as, software for processing and reading ebooks, various collections of libraries, dictionaries, tutorials on programming or languages, etc. It seems there's still a small number of extremely popular apps that skew the average. Let's check the ones that are skewing our average.

In [37]:
```
for app in android_final:
    if app[1] == 'BOOKS_AND_REFERENCE' and (app[5] == '1,000,000+'
                                            or app[5] == '5,000,000+'
                                            or app[5] == '10,000,000+'
                                            or app[5] == '50,000,000+'):
        print(app[0], ':', app[5])
```

```
Wikipedia : 10,000,000+
Cool Reader : 10,000,000+
Book store : 1,000,000+
FBReader: Favorite Book Reader : 10,000,000+
Free Books - Spirit Fanfiction and Stories : 1,000,000+
AlReader -any text book reader : 5,000,000+
FamilySearch Tree : 1,000,000+
Cloud of Books : 1,000,000+
ReadEra – free ebook reader : 1,000,000+
Ebook Reader : 5,000,000+
Read books online : 5,000,000+
eBoox: book reader fb2 epub zip : 1,000,000+
All Maths Formulas : 1,000,000+
Ancestry : 5,000,000+
HTC Help : 10,000,000+
Moon+ Reader : 10,000,000+
English-Myanmar Dictionary : 1,000,000+
Golden Dictionary (EN-AR) : 1,000,000+
All Language Translator Free : 1,000,000+
Aldiko Book Reader : 10,000,000+
Dictionary - WordWeb : 5,000,000+
50000 Free eBooks & Free AudioBooks : 5,000,000+
Al-Quran (Free) : 10,000,000+
Al Quran Indonesia : 10,000,000+
Al'Quran Bahasa Indonesia : 10,000,000+
Al Quran Al karim : 1,000,000+
Al Quran : EAlim - Translations & MP3 Offline : 5,000,000+
Koran Read &MP3 30 Juz Offline : 1,000,000+
Hafizi Quran 15 lines per page : 1,000,000+
Quran for Android : 10,000,000+
Satellite AR : 1,000,000+
Oxford A-Z of English Usage : 1,000,000+
Dictionary.com: Find Definitions for English Words : 10,000,000+
English Dictionary - Offline : 10,000,000+
Bible KJV : 5,000,000+
NOOK: Read eBooks & Magazines : 10,000,000+
Brilliant Quotes: Life, Love, Family & Motivation : 1,000,000+
Stats Royale for Clash Royale : 1,000,000+
Dictionary : 10,000,000+
wikiHow: how to do anything : 1,000,000+
EGW Writings : 1,000,000+
My Little Pony AR Guide : 1,000,000+
Spanish English Translator : 10,000,000+
Dictionary - Merriam-Webster : 10,000,000+
JW Library : 10,000,000+
Oxford Dictionary of English : Free : 10,000,000+
English Hindi Dictionary : 10,000,000+
English to Hindi Dictionary : 5,000,000+
```

This niche seems to be dominated by software for processing and reading ebooks, as well as various collections of libraries and dictionaries, so it's probably not a good idea to build similar apps since there'll be some significant competition.

We also notice there are quite a few apps built around the book Quran, which suggests that building an app around a popular book can be profitable. It seems that taking a popular book (perhaps a more recent book) and turning it into an app could be profitable for both the Google Play and the App Store markets.

However, it looks like the market is already full of libraries, so we need to add some special features besides the raw version of the book. This might include daily quotes from the book, an audio version of the book, quizzes on the book, a forum where people can discuss the book and share their perceptions and etc.

# Conclusion

In this project, we analyzed data about the App Store and Google Play mobile apps with a goal of recommending an app profile that can be profitable for both markets.

We concluded that taking a popular book (perhaps a more recent book) and turning it into an app could be profitable for both the Google Play and the App Store markets. The markets are already full of libraries, so we need to add some special features besides the raw version of the book. This might include daily quotes from the book, an audio version of the book, quizzes on the book, a forum where people can discuss the book, etc.