

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/343111126>

# Python Projects, with Explanation of Code: Frequency Tables, Loops, and other Calculations and Methods; Project 1: Profitable Apps on the iOS and the Android Markets; Project 2: An...

Preprint · July 2020

CITATIONS

0

READS

579

1 author:



[Aleksandra Zaba](#)

University of Utah

24 PUBLICATIONS 11 CITATIONS

SEE PROFILE

# Project On New Profitable App Profiles for the App Store and Google Play Markets, with Explanation of Code

This project aims to suggest new profitable apps to developers on the Android and iOS mobile app markets. To determine what sort of apps may be profitable, we investigate apps that have been successful in the past. Success is operationalized as the number of users of a given app because the revenue at our company stems from in-app ads. Apps are free to download and to install.

Two free data sets with  $n = 10000$  (Android) and  $n = 7000$  (iOS) respectively are used as samples for the ca. 4 million apps that were available in 2018: 2 million iOS and 2.1 million Android apps.

The links to the data sets are:

<https://dq-content.s3.amazonaws.com/350/googleplaystore.csv> (<https://dq-content.s3.amazonaws.com/350/googleplaystore.csv>)

<https://dq-content.s3.amazonaws.com/350/AppStore.csv> (<https://dq-content.s3.amazonaws.com/350/AppStore.csv>)

Let us open the two data sets in Python for further exploration and analysis.

## 1. Data Exploration

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

#### The Google Play data set (Android apps) ####
opened_file = open('googleplaystore.csv') # Open the file using the open
() command. Save the output to a variable named 'opened_file'.
read_file = reader(opened_file) # Read in the opened file using the read
er() command. Save the output to a variable named 'read_file'.
android = list(read_file) # Transform the read-in file to a list of list
s using the list() command. Save the list of lists to a variable named
'android'.
android_header = android[0] # The first row is the header
android = android[1:] # Data begins in row 2, after the header

#### The App Store data set (iOS apps) ####
opened_file = open('AppStore.csv')
read_file = reader(opened_file)
ios = list(read_file)
ios_header = ios[0]
ios = ios[1:]
```

The following creates a function that allows one to explore the data sets.

```
In [2]: def explore_data(dataset, start, end, rows_and_columns=False): # We call
the function 'explore_data'
dataset_slice = dataset[start:end]
for row in dataset_slice: # Looping through data set
    print(row)
    print('\n') # Adds a new (empty) line after each row

    if rows_and_columns: # If rows_and_columns is True
        print('Number of rows:', len(dataset)) # Print this for any data
set
        print('Number of columns:', len(dataset[0])) # Print this for an
y data set

print(android_header)
print('\n') # Insert an empty line
explore_data(android, 0, 3, True)
```

```
['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type', 'P
rice', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver', 'Andr
oid Ver']
```

```
['Photo Editor & Candy Camera & Grid & ScrapBook', 'ART_AND_DESIGN',
'4.1', '159', '19M', '10,000+', 'Free', '0', 'Everyone', 'Art & Desig
n', 'January 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,
2018', '2.0.0', '4.0.3 and up']
```

```
['U Launcher Lite – FREE Live Cool Themes, Hide Apps', 'ART_AND_DESIG
N', '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: 10841
```

```
Number of columns: 13
```

We printed the first few rows of the Google Play data set, and found the number of rows and columns of the data set. The data sets should not have a header row as the function `explore_data()` assumes that there is no header row. This data set contains 10841 rows and 13 columns. Columns useful to investigate are 'App', 'Category', 'Reviews', 'Installs', 'Type', 'Price', and 'Genres'. Next, let us use the new function in [2] on the App store data set.

```
In [3]: print(ios_header)
print('\n')
explore_data(ios, 0, 3, True)

['id', 'track_name', 'size_bytes', 'currency', 'price', 'rating_count_tot', 'rating_count_ver', 'user_rating', 'user_rating_ver', 'ver', 'content_rating', 'prime_genre', 'sup_devices.num', 'ipadSc_urls.num', 'lang.num', 'vpp_license']

['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', '579', '4.5', '4.5', '9.24.12', '9+', 'Games', '38', '5', '18', '1']

Number of rows: 7197
Number of columns: 16
```

This data set has 7197 rows and 16 columns. The columns that might help with this analysis are 'track\_name', 'currency', 'price', 'rating\_count\_tot', 'rating\_count\_ver', and 'prime\_genre'. The documentation of these can be found in <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>).

## 2. Data Cleaning

### 2.1 Data Cleaning: Removing Incorrect Data

The discussion section for the Google Play data set at <https://www.kaggle.com/lava18/google-play-store-apps/discussion/66015> (<https://www.kaggle.com/lava18/google-play-store-apps/discussion/66015>) reveals that line 10472 contains an error. We print that line in the following as well as the header and a correct row, in order to be able to compare them to find the error.

```
In [4]: print(android[10472]) # incorrect row
print('\n')
print(android_header) # header
print('\n')
print(android[0]) # correct 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']

['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']
```

As noted in the link cited in the text immediately preceding [4], the 'Category' column is missing a value in the row with an error, which is the app 'Life Made WI-Fi Touchscreen Photo Frame'. This leads to the rating to be 19, whereas the maximum rating for the Android apps is actually 5. This row could be deleted, which is what is done in the following.

```
In [5]: print(len(android)) # Number of elements, here, rows, in the data set
del android[10472] # Don't run this more than once
print(len(android))

10841
10840
```

We went from 10841 to 10840, which indicates that we succeeded in eliminating one row.

## 2.2 Data Cleaning Continued: Removing Duplicates

The Google Play data set contains some duplicate rows:

```
In [6]: for app in android:
        name = app[0] # 'Name' is the first column
        if name == 'Instagram':
            print(app)

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

The example above is for the app 'Instagram'. What about the other apps in this data set? Let us explore this question in the following.

```
In [7]: duplicate_apps = [] # Creating new empty list
        unique_apps = [] # Creating new empty list

        for app in android: # Looping through the 'Android' data set
            name = app[0]
            if name in unique_apps: # For each repetition, if column 1 is in 'unique_apps'
                duplicate_apps.append(name) # Add name to list 'duplicate_apps'
            else: # If not yet in 'unique_apps'
                unique_apps.append(name) # Add name to list 'unique_apps'

        print('Number of duplicate apps:', len(duplicate_apps))
        print('\n')
        print('Examples of duplicate apps:', duplicate_apps[:15]) # Print 15 entries
```

Number of duplicate apps: 1181

Examples of duplicate apps: ['Quick PDF Scanner + OCR FREE', 'Box', 'Google My Business', 'ZOOM Cloud Meetings', 'join.me - Simple Meetings', 'Box', 'Zenefits', 'Google Ads', 'Google My Business', 'Slack', 'FreshBooks Classic', 'Insightly CRM', 'QuickBooks Accounting: Invoicing & Expenses', 'HipChat - Chat Built for Teams', 'Xero Accounting Software']

This data set thus contains 1181 duplicate data sets. To remove them and to only keep one row per app, one way is to only keep that row per app that has the highest number of ratings. Rating numbers vary across the duplicates.

A dictionary is created below in which each app corresponds to a separate key and its value is the highest number of reviews of that app. This creates a data set with only one app entry each, the highest one.

```
In [8]: reviews_max = {} # Create new dictionary, empty for now

for app in android: # If app exists as dictionary key in android list
    name = app[0] # Name is column 1
    n_reviews = float(app[3]) # Number of reviews appears in column 4 and we convert it to floats

    if name in reviews_max and reviews_max[name] < n_reviews: # Getting rid of all below the max
        reviews_max[name] = n_reviews

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

The length of this new dictionary should be the length of the entire data set minus 1,181, which corresponds to the number of entries that have multiple ratings (see above). Printing the result of this difference and comparing it to the number of rows in the new dictionary supports this assumption.

```
In [9]: print('Expected length:', len(android) - 1181)
print('Actual length:', len(reviews_max))
```

```
Expected length: 9659
Actual length: 9659
```

Next, it would be good to remove the duplicates using the new dictionary 'max\_reviews'.

```
In [10]: android_clean = [] # Creating empty list
already_added = [] # Creating empty list

for app in android: # Loop started
    name = app[0]
    n_reviews = float(app[3])

    if (reviews_max[name] == n_reviews) and (name not in
                                                already_added): # If number
of reviews = as in 'reviews_max' dictionary and if name of app not yet
in 'already_added' list
        android_clean.append(app) # Add row to 'android_clean' list
        already_added.append(name) # Add name to 'already_added' list
```

In this step, we added the current row (app) to a list called 'android\_clean' and the app name (name) to another list ('already\_added') if the number of reviews of the current app is equal to the number of reviews as specified in the new dictionary ('reviews\_max'), and if the name of the app is not already in the 'already\_added' list. The latter condition refers to cases in which there are multiple identical max review numbers. The android\_clean list should have 9659 rows. Let us test this.

```
In [11]: explore_data(android_clean, 0, 3, True) # Using our function from [2] to
         explore the new data set/list

['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', '50,000,000+',
'Free', '0', 'Teen', 'Art & Design', 'June 8, 2018', 'Varies with device',
'4.2 and up']

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

The number looks correct.

## 2.3 Data Cleaning Continued: Removing Non-English Apps

In the following, let us clean the data further by removing apps that have a non-English name, for the sake of an easy preliminary analysis and because the present project tries to work with apps designed for an English audience. We can accomplish this for example by removing apps containing non-English symbols, that is, symbols that are not encoded using the ASCII standard (numbers 0 - 127 that encode a symbol each).

Let us build a function that tells us whether an app name contains ASCII characters. First, let us look at examples with non-English names.

```
In [12]: print(ios[813][1])
         print(ios[6731][1])

         print(android_clean[4412][0])
         print(android_clean[7940][0])

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～
中国語 AQリスニング
لعبة تقدر تريح DZ
```

Now, to a function that isolates English names.



```
In [13]: def is_english(string): # Defining new function

    for character in string:
        if ord(character) > 127: # Built-in function: find encoding numbers
            return False

    return True # Else return 'true'

print(is_english('Instagram'))
print(is_english('爱奇艺PPS - 《欢乐颂2》电视剧热播'))
```

True  
False

The function 'is\_english' shows correctly that the non-English app tested in the second function run is not ASCII encoded. However, note that the current function will also dub as false those apps that contain symbols other than non-ASCII symbols, for example, smileys (example [14] below). This can cause data loss. The function thus requires some tweaking. One way to keep app names that are English but contain say a smiley symbol is to specify we only get rid of an app if its name has > 3 non-ASCII characters. The code in [15] below does exactly that.

```
In [14]: print(is_english('Docs To Go™ Free Office Suite'))
print(is_english('Instachat 😊'))

print(ord('™'))
print(ord('😊'))
```

False  
False  
8482  
128540

```
In [15]: def is_english(string):
    non_ascii = 0

    for character in string:
        if ord(character) > 127:
            non_ascii += 1

    if non_ascii > 3:
        return False
    else:
        return True

print(is_english('Docs To Go™ Free Office Suite'))
print(is_english('Instachat 😊'))
```

True  
True

We leave optimization of this function for a later exercise. Let us in the following test the function 'is\_english(string)' on both data sets under investigation in the present project.

```
In [16]: android_english = [] # Initiating an empty list
ios_english = [] # Initiating another empty list

for app in android_clean:
    name = app[0] # First column
    if is_english(name):
        android_english.append(app) # Append row if name is English

for app in ios:
    name = app[1] # Second column
    if is_english(name):
        ios_english.append(app)

explore_data(android_english, 0, 3, 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 & Design', 'August 1, 2018', '1.2.4', '4.0.3 and up']

['Sketch - Draw & Paint', 'ART\_AND\_DESIGN', '4.5', '215644', '25M', '50,000,000+', 'Free', '0', 'Teen', 'Art & Design', 'June 8, 2018', 'Varies with device', '4.2 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', '579', '4.5', '4.5', '9.24.12', '9+', 'Games', '38', '5', '18', '1']

Number of rows: 6183  
Number of columns: 16

## 3. Data Analysis

### 3.1 Extracting Free Apps

We cleaned the data a bit. Now, on to analyzing it. We would like to focus on free apps only since, as mentioned, our company only produces free apps. Let us attempt to isolate them in the following code.

```
In [17]: android_final = [] # Empty list
ios_final = [] # Empty list

for app in android_english: # Looping through data set to isolate the fr
ee apps in a separate list
    price = app[7]
    if price == '0':
        android_final.append(app) # Append row if price is 0

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

print(len(android_final)) # Print number of items in list 1
print(len(ios_final)) # Print number of items in list 2

8864
3222
```

The remaining number of Android apps left for analysis is 8864, and that of iOS apps is 3222.

### 3.2 Frequencies of Apps by Genre

We want to determine what type of app pays. For this, we need to know which apps have attracted many customers. Our goal is to build a minimal Android version of the app and add it to Google Play. We will then develop it further if it has a good response from users. And we build an iOS version of the app and add it to the App Store if it is profitable after six months. In fact we want to be able to recommend app profiles that will be successful on both markets so that they can be added to both the Google Play and to the App Store.

For this purpose, let us first determine the relative frequencies of highly popular apps on both markets. At first, let us build frequency tables for a few columns in our data sets, specifically, for the 'prime\_genre' column of the App Store data set and for the 'Genres' and 'Category' columns of the Google Play data set.

We will build a function to generate frequency tables showing percentages, and a function to display the percentages in descending order.

```

In [18]: def freq_table(dataset, index): # Defining a function that takes a given
column (integer) of a given data set (list of lists) as input
    table = {} # Initiating an empty dictionary
    total = 0 # Initiating an empty variable

    for row in dataset:
        total += 1 # Loop through the data set and increment total variable by 1 for each iteration
        value = row[index]
        if value in table:
            table[value] += 1 # If the value is already in the 'table' dictionary, increment the frequency of that value by 1
        else:
            table[value] = 1 # Else, initialize the value with a value of 1 inside the 'table' dictionary

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

    return table_percentages

# The function above returns the frequency table (as a dictionary) for any column we want; frequencies are expressed as percentages

def display_table(dataset, index):
    table = freq_table(dataset, index) # Assign frequency table to new variable 'table'
    table_display = [] # Initializing an empty list
    for key in table:
        key_val_as_tuple = (table[key], key)
        table_display.append(key_val_as_tuple) # Extract only the value you want

# The display_table() function you see above:

# Takes in two parameters: dataset and index. Dataset is expected to be a list of lists, and index is expected to be an integer.
# Generates a frequency table using the freq_table() function.
# Transforms the frequency table into a list of tuples.
# The below sorts the list in a descending order and it
# prints the entries of the frequency table in descending order.

    table_sorted = sorted(table_display, reverse = True) # Assign sorted table_display (in descending order) to new variable name
    for entry in table_sorted:
        print(entry[1], ': ', entry[0]) # Print app name: Associated frequency per entry in sorted table

```

```
In [19]: display_table(ios_final, -5) # Using new function on ios_final data set
```

```
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
```

As shown above, the 'prime\_genre' column thus has many apps in the Games category (ca. 58%), and the rest of the major apps are Entertainment (7.88%) and Photo & Video (4.97%). The remaining positions are filled by, as an example, Education (3.66%) and Social Networking (3.29%). All other apps, such as Shopping, Utilities, and News, amount to only a little each.

The App Store hence seems to contain predominantly free apps that are designed for leisure, for example, games, while more daily life-oriented apps such as those pertaining to education are less frequent. Now, what about the Google Play data set?

```
In [20]: display_table(android_final, 1) # column 'Category'
```

```
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
```

On Google Play, practical apps (for example, 'Family') are better represented than fun related ones such as 'Games'. This is confirmed by the column 'Genres':

```
In [21]: display_table(android_final, -4) # column '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  
Simulation : 2.0419675090252705  
Dating : 1.861462093862816  
Arcade : 1.8501805054151623  
Video Players & Editors : 1.7712093862815883  
Casual : 1.7599277978339352  
Maps & Navigation : 1.3989169675090252  
Food & Drink : 1.2409747292418771  
Puzzle : 1.128158844765343  
Racing : 0.9927797833935018  
Role Playing : 0.9363718411552346  
Libraries & Demo : 0.9363718411552346  
Auto & Vehicles : 0.9250902527075812  
Strategy : 0.9138086642599278  
House & Home : 0.8235559566787004  
Weather : 0.8009927797833934  
Events : 0.7107400722021661  
Adventure : 0.6768953068592057  
Comics : 0.6092057761732852  
Beauty : 0.5979241877256317  
Art & Design : 0.5979241877256317  
Parenting : 0.4963898916967509  
Card : 0.45126353790613716  
Casino : 0.42870036101083037  
Trivia : 0.41741877256317694  
Educational;Education : 0.39485559566787  
Board : 0.3835740072202166  
Educational : 0.3722924187725632  
Education;Education : 0.33844765342960287  
Word : 0.2594765342960289  
Casual;Pretend Play : 0.236913357400722  
Music : 0.2030685920577617  
Racing;Action & Adventure : 0.16922382671480143  
Puzzle;Brain Games : 0.16922382671480143  
Entertainment;Music & Video : 0.16922382671480143  
Casual;Brain Games : 0.13537906137184114  
Casual;Action & Adventure : 0.13537906137184114  
Arcade;Action & Adventure : 0.12409747292418773  
Action;Action & Adventure : 0.10153429602888085



Educational;Pretend Play : 0.09025270758122744  
Simulation;Action & Adventure : 0.078971119133574  
Parenting;Education : 0.078971119133574  
Entertainment;Brain Games : 0.078971119133574  
Board;Brain Games : 0.078971119133574  
Parenting;Music & Video : 0.06768953068592057  
Educational;Brain Games : 0.06768953068592057  
Casual;Creativity : 0.06768953068592057  
Art & Design;Creativity : 0.06768953068592057  
Education;Pretend Play : 0.056407942238267145  
Role Playing;Pretend Play : 0.04512635379061372  
Education;Creativity : 0.04512635379061372  
Role Playing;Action & Adventure : 0.033844765342960284  
Puzzle;Action & Adventure : 0.033844765342960284  
Entertainment;Creativity : 0.033844765342960284  
Entertainment;Action & Adventure : 0.033844765342960284  
Educational;Creativity : 0.033844765342960284  
Educational;Action & Adventure : 0.033844765342960284  
Education;Music & Video : 0.033844765342960284  
Education;Brain Games : 0.033844765342960284  
Education;Action & Adventure : 0.033844765342960284  
Adventure;Action & Adventure : 0.033844765342960284  
Video Players & Editors;Music & Video : 0.02256317689530686  
Sports;Action & Adventure : 0.02256317689530686  
Simulation;Pretend Play : 0.02256317689530686  
Puzzle;Creativity : 0.02256317689530686  
Music;Music & Video : 0.02256317689530686  
Entertainment;Pretend Play : 0.02256317689530686  
Casual;Education : 0.02256317689530686  
Board;Action & Adventure : 0.02256317689530686  
Video Players & Editors;Creativity : 0.01128158844765343  
Trivia;Education : 0.01128158844765343  
Travel & Local;Action & Adventure : 0.01128158844765343  
Tools;Education : 0.01128158844765343  
Strategy;Education : 0.01128158844765343  
Strategy;Creativity : 0.01128158844765343  
Strategy;Action & Adventure : 0.01128158844765343  
Simulation;Education : 0.01128158844765343  
Role Playing;Brain Games : 0.01128158844765343  
Racing;Pretend Play : 0.01128158844765343  
Puzzle;Education : 0.01128158844765343  
Parenting;Brain Games : 0.01128158844765343  
Music & Audio;Music & Video : 0.01128158844765343  
Lifestyle;Pretend Play : 0.01128158844765343  
Lifestyle;Education : 0.01128158844765343  
Health & Fitness;Education : 0.01128158844765343  
Health & Fitness;Action & Adventure : 0.01128158844765343  
Entertainment;Education : 0.01128158844765343  
Communication;Creativity : 0.01128158844765343  
Comics;Creativity : 0.01128158844765343  
Casual;Music & Video : 0.01128158844765343  
Card;Action & Adventure : 0.01128158844765343  
Books & Reference;Education : 0.01128158844765343  
Art & Design;Pretend Play : 0.01128158844765343  
Art & Design;Action & Adventure : 0.01128158844765343  
Arcade;Pretend Play : 0.01128158844765343  
Adventure;Education : 0.01128158844765343

The results of the 'Genres' column has a lot of subcategories. It might be a better idea to focus on the 'Category' column for now because its granularity is sufficient for the purposes of this project.

Just because some categories of apps are more frequent than others, we cannot conclude that these will be the ones used more frequently by users. Therefore, we will next calculate the average number of installs for each app genre. We will look at the 'Installs' column for the Google Play data set and for the total number of user ratings in the rating\_count\_tot app to get a similar kind of info for the App Store data set.

### 3.3 Most Popular Apps by Genre, App Store

First, we want the average number for the App Store. This implies isolating the apps of each genre, summing up the user ratings for the apps of that genre, and dividing the sum by the number of apps belonging to that genre (vs. by the total number of apps).

```
In [22]: genres_ios = freq_table(ios_final, -5) # Using freq_table function on iOS data to generate a frequency table for the 'prime_genre' column

for genre in genres_ios: # Looping over the unique genres of the App Store data set
    total = 0
    len_genre = 0
    for app in ios_final:
        genre_app = app[-5]
        if genre_app == genre:
            n_ratings = float(app[5])
            total += n_ratings
            len_genre += 1

# For each iteration (it was assumed that the iteration variable is named 'genre'):
# We initiated a variable named 'total' with a value of 0. This variable stores the sum of user ratings (the number of ratings, not the actual ratings) specific to each genre.
# We initiated a variable named 'len_genre' with a value of 0. This variable stores the number of apps specific to each genre.
# We looped over the App Store data set, and for each iteration:
# We saved the app genre to a variable named 'genre_app'.
# If 'genre_app' was the same as 'genre' (the iteration variable of the main loop), then:
# We saved the number of user ratings of the app as a float.
# We added up the number of user ratings to the total variable.
# We incremented the len_genre variable by 1.

    avg_n_ratings = total / len_genre
    print(genre, ': ', avg_n_ratings)
```

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

Navigation apps seem to have the most reviews on average. Specifically, Waze and Google Maps are the most popular:

```
In [23]: for app in ios_final:
          if app[-5] == 'Navigation': # If column with index -5 (i.e., 5th from
m end) is 'Navigation'
          print(app[1], ': ', app[5]) # Name and number of ratings
```

```
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
```

A similar pattern, where certain apps have a lot of users, occurs in the 'Social Networking' category:

```
In [24]: for app in ios_final:
          if app[-5] == 'Social Networking':
              print(app[1], ': ', app[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

And also in the 'Music' and 'Reference' categories, the average numbers are heavily influenced by some apps.

```
In [25]: for app in ios_final:
          if app[-5] == 'Music':
              print(app[1], ': ', app[5])
```



Pandora - Music & Radio : 1126879  
Spotify Music : 878563  
Shazam - Discover music, artists, videos & lyrics : 402925  
iHeartRadio - Free Music & Radio Stations : 293228  
SoundCloud - Music & Audio : 135744  
Magic Piano by Smule : 131695  
Smule Sing! : 119316  
TuneIn Radio - MLB NBA Audiobooks Podcasts Music : 110420  
Amazon Music : 106235  
SoundHound Song Search & Music Player : 82602  
Sonos Controller : 48905  
Bandsintown Concerts : 30845  
Karaoke - Sing Karaoke, Unlimited Songs! : 28606  
My Mixtapez Music : 26286  
Sing Karaoke Songs Unlimited with StarMaker : 26227  
Ringtones for iPhone & Ringtone Maker : 25403  
Musi - Unlimited Music For YouTube : 25193  
AutoRap by Smule : 18202  
Spinrilla - Mixtapes For Free : 15053  
Napster - Top Music & Radio : 14268  
edjing Mix:DJ turntable to remix and scratch music : 13580  
Free Music - MP3 Streamer & Playlist Manager Pro : 13443  
Free Piano app by Yokee : 13016  
Google Play Music : 10118  
Certified Mixtapes - Hip Hop Albums & Mixtapes : 9975  
TIDAL : 7398  
YouTube Music : 7109  
Nicki Minaj: The Empire : 5196  
Sounds app - Music And Friends : 5126  
SongFlip - Free Music Streamer : 5004  
Simple Radio - Live AM & FM Radio Stations : 4787  
Deezer - Listen to your Favorite Music & Playlists : 4677  
Ringtones for iPhone with Ringtone Maker : 4013  
Bose SoundTouch : 3687  
Amazon Alexa : 3018  
DatPiff : 2815  
Trebel Music - Unlimited Music Downloader : 2570  
Free Music Play - Mp3 Streamer & Player : 2496  
Acapella from PicPlayPost : 2487  
Coach Guitar - Lessons & Easy Tabs For Beginners : 2416  
Musiccloud - MP3 and FLAC Music Player for Cloud Platforms. : 2211  
Piano - Play Keyboard Music Games with Magic Tiles : 1636  
Boom: Best Equalizer & Magical Surround Sound : 1375  
Music Freedom - Unlimited Free MP3 Music Streaming : 1246  
AmpMe - A Portable Social Party Music Speaker : 1047  
Medly - Music Maker : 933  
Bose Connect : 915  
Music Memos : 909  
UE BOOM : 612  
LiveMixtapes : 555  
NOISE : 355  
MP3 Music Player & Streamer for Clouds : 329  
Musical Video Maker - Create Music clips lip sync : 320  
Cloud Music Player - Downloader & Playlist Manager : 319  
Remixlive - Remix loops with pads : 288  
QQ音乐HD : 224  
Blocs Wave - Make & Record Music : 158

PlayGround • Music At Your Fingertips : 150  
 Music and Chill : 135  
 The Singing Machine Mobile Karaoke App : 130  
 radio.de - Der Radioplayer : 64  
 Free Music - Player & Streamer for Dropbox, OneDrive & Google Drive : 46  
 NRJ Radio : 38  
 Smart Music: Streaming Videos and Radio : 17  
 BOSS Tuner : 13  
 PetitLyrics : 0

```
In [26]: 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 : 17588  
 Merriam-Webster Dictionary : 16849  
 Night Sky : 12122  
 City Maps for Minecraft PE - The Best Maps for Minecraft Pocket Edition (MCPE) : 8535  
 LUCKY BLOCK MOD ™ for Minecraft PC Edition - The Best Pocket Wiki & Mod s Installer 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 Pocket 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

Here, the bible and dictionaries contribute a lot to the average rating. In a future project, one could try to remove those very frequently reviewed apps for each category. Based on the pattern above, the App Store's market seems saturated on the 'fun' front, so that a more educational app might be a good new idea, such as one based on another popular category like 'Reference'. Often, ads shown in, for example, documentations about famous movies based on books are watched a lot. One might suggest new apps based on popular books (Quran based apps seemed popular), but instead of 'just' turning the book into an app (libraries are already present frequently), add, for example, quizzes on it or documentation on its text, such as explanations of words or concepts or descriptions of famous places mentioned in them. In addition, we might also recommend some more new games based on new popular books, especially in tandem with books and quizzes on them to complement them, just because the 'fun' category has proven so successful.

Some of the other popular genres in the iOS category are 'weather', 'book', 'food', 'drink', and 'finance'. Of these, books seem the most interesting based on the previous suggestions. The others are either too specialized ('finance'), they require services such as delivery or cooking ('Starbucks'), or they are not likely to keep people watching any ads, and are thus likely not profitable for us ('weather').

Next, let us pinpoint the most popular app genres for the Google Play market.

## 3.4 Most Popular Apps by Genre, Google Play

This data set contains data about the number of installs.

```
In [27]: display_table(android_final, 5) # This is the 'Installs' column
```

```
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
```

While the data are not very precise in that 100,000+ installs is not a very clear piece of info, it is still possible to make categories, for example, 100,000+ vs. 1,000,000+ installs. To perform averages (see the following), we will convert these numbers to floats. I.e., we will assume 100,000+ means 100,000, etc.

```
In [28]: categories_android = freq_table(android_final, 1)

for category in categories_android:
    total = 0 # Initiating new empty variable
    len_category = 0 # Initiating new empty variable
    for app in android_final:
        category_app = app[1] # 'category_app' is in column 2
        if category_app == category:
            n_installs = app[5] # 'n_installs' is in column 6
            n_installs = n_installs.replace(',', '') # Symbol replacement instructions
            n_installs = n_installs.replace('+', '')
            total += float(n_installs) # Adding on
            len_category += 1 # Adding 1 to column 'len_category'
    avg_n_installs = total / len_category # Average number of installs by genre
    print(category, ': ', avg_n_installs)
```

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

Let us look at a more fine grained picture of the main category, 'Communication'.

```
In [29]: 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+'): # If
                column 2 ('Genres') = 'Communication' and the number of downloads is on
                e of the three specified categories, then print name : installs
                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+
```

As we can see, the main apps in that category are WhatsApp, Messenger, and several others, with hundreds of millions of installs. Let us remove some of those extra influential apps:

```
In [30]: under_100_m = [] # Initializing new empty list

for app in android_final: # Looping
    n_installs = app[5]
    n_installs = n_installs.replace(',', '') # Replacing commas with blanks
    n_installs = n_installs.replace('+', '') # Replacing plus symbols with blanks
    if (app[1] == 'COMMUNICATION') and (float(n_installs) < 100000000):
        under_100_m.append(float(n_installs)) # Isolating apps under 100 million installs


sum(under_100_m) / len(under_100_m) # Average
```

Out[30]: 3603485.3884615386

This step reduces the average. The same pattern, namely, that some of the apps are very prominent can also be found in the other categories, for example, in the video players (for example, YouTube), social apps (for example, Instagram), photography (for example, Google Photos), and productivity apps (for example, Dropbox). These highly popular apps may skew the perception of the app genres in that these may appear as more relevant than they are. In the following, let us explore one of the popular categories, namely books and reference, to compare them to the same category in the App Store. We want to suggest apps that may be of profit on both markets.

```
In [31]: for app in android_final:
          if app[1] == 'BOOKS_AND_REFERENCE': # If column 2 = 'Books_and_refer
          ence'
          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+  
Litnet - E-books : 100,000+  
Read books online : 5,000,000+  
English to Urdu Dictionary : 500,000+  
eBoox: book reader fb2 epub zip : 1,000,000+  
English Persian Dictionary : 500,000+  
Flybook : 500,000+  
All Maths Formulas : 1,000,000+  
Ancestry : 5,000,000+  
HTC Help : 10,000,000+  
English translation from Bengali : 100,000+  
Pdf Book Download - Read Pdf Book : 100,000+  
Free Book Reader : 100,000+  
eBoox new: Reader for fb2 epub zip books : 50,000+  
Only 30 days in English, the guideline is guaranteed : 500,000+  
Moon+ Reader : 10,000,000+  
SH-02J Owner's Manual (Android 8.0) : 50,000+  
English-Myanmar Dictionary : 1,000,000+  
Golden Dictionary (EN-AR) : 1,000,000+  
All Language Translator Free : 1,000,000+  
Azpen eReader : 500,000+  
URBANO V 02 instruction manual : 100,000+  
Bible : 100,000,000+  
C Programs and Reference : 50,000+  
C Offline Tutorial : 1,000+  
C Programs Handbook : 50,000+  
Amazon Kindle : 100,000,000+  
Aab e Hayat Full Novel : 100,000+  
Aldiko Book Reader : 10,000,000+  
Google I/O 2018 : 500,000+  
R Language Reference Guide : 10,000+  
Learn R Programming Full : 5,000+  
R Programing Offline Tutorial : 1,000+  
Guide for R Programming : 5+  
Learn R Programming : 10+  
R Quick Reference Big Data : 1,000+  
V Made : 100,000+  
Wattpad  Free Books : 100,000,000+  
Dictionary - WordWeb : 5,000,000+

Guide (for X-MEN) : 100,000+  
 AC Air condition Troubleshoot,Repair,Maintenance : 5,000+  
 AE Bulletins : 1,000+  
 Ae Allah na Dai (Rasa) : 10,000+  
 50000 Free eBooks & Free AudioBooks : 5,000,000+  
 Ag PhD Field Guide : 10,000+  
 Ag PhD Deficiencis : 10,000+  
 Ag PhD Planting Population Calculator : 1,000+  
 Ag PhD Soybean Diseases : 1,000+  
 Fertilizer Removal By Crop : 50,000+  
 A-J Media Vault : 50+  
 Al-Quran (Free) : 10,000,000+  
 Al Quran (Tafsir & by Word) : 500,000+  
 Al Quran Indonesia : 10,000,000+  
 Al'Quran Bahasa Indonesia : 10,000,000+  
 Al Quran Al karim : 1,000,000+  
 Al-Muhaffiz : 50,000+  
 Al Quran : EALim - Translations & MP3 Offline : 5,000,000+  
 Al-Quran 30 Juz free copies : 500,000+  
 Koran Read &MP3 30 Juz Offline : 1,000,000+  
 Hafizi Quran 15 lines per page : 1,000,000+  
 Quran for Android : 10,000,000+  
 Surah Al-Waqiah : 100,000+  
 Hisnul Al Muslim - Hisn Invocations & Adhkaar : 100,000+  
 Satellite AR : 1,000,000+  
 Audiobooks from Audible : 100,000,000+  
 Kinot & Eichah for Tisha B'Av : 10,000+  
 AW Tozer Devotionals - Daily : 5,000+  
 Tozer Devotional -Series 1 : 1,000+  
 The Pursuit of God : 1,000+  
 AY Sing : 5,000+  
 Ay Hasnain k Nana Milad Naat : 10,000+  
 Ay Mohabbat Teri Khatir Novel : 10,000+  
 Arizona Statutes, ARS (AZ Law) : 1,000+  
 Oxford A-Z of English Usage : 1,000,000+  
 BD Fishpedia : 1,000+  
 BD All Sim Offer : 10,000+  
 Youboox - Livres, BD et magazines : 500,000+  
 B&H Kids AR : 10,000+  
 B y H Niños ES : 5,000+  
 Dictionary.com: Find Definitions for English Words : 10,000,000+  
 English Dictionary - Offline : 10,000,000+  
 Bible KJV : 5,000,000+  
 Borneo Bible, BM Bible : 10,000+  
 MOD Black for BM : 100+  
 BM Box : 1,000+  
 Anime Mod for BM : 100+  
 NOOK: Read eBooks & Magazines : 10,000,000+  
 NOOK Audiobooks : 500,000+  
 NOOK App for NOOK Devices : 500,000+  
 Browser by Barnes & Noble : 5,000+  
 bp e-store : 1,000+  
 Brilliant Quotes: Life, Love, Family & Motivation : 1,000,000+  
 BR Ambedkar Biography & Quotes : 10,000+  
 BU Alsace : 100+  
 Catholic La Bu Zo Kam : 500+  
 Khrifa Hla Bu (Solfa) : 10+

Kristian Hla Bu : 10,000+  
SA HLA BU : 1,000+  
Learn SAP BW : 500+  
Learn SAP BW on HANA : 500+  
CA Laws 2018 (California Laws and Codes) : 5,000+  
Bootable Methods(USB-CD-DVD) : 10,000+  
cloudLibrary : 100,000+  
SDA Collegiate Quarterly : 500+  
Sabbath School : 100,000+  
Cypress College Library : 100+  
Stats Royale for Clash Royale : 1,000,000+  
GATE 21 years CS Papers(2011-2018 Solved) : 50+  
Learn CT Scan Of Head : 5,000+  
Easy Cv maker 2018 : 10,000+  
How to Write CV : 100,000+  
CW Nuclear : 1,000+  
CY Spray nozzle : 10+  
BibleRead En Cy Zh Yue : 5+  
CZ-Help : 5+  
Modlitební knížka CZ : 500+  
Guide for DB Xenoverse : 10,000+  
Guide for DB Xenoverse 2 : 10,000+  
Guide for IMS DB : 10+  
DC HSEMA : 5,000+  
DC Public Library : 1,000+  
Painting Lulu DC Super Friends : 1,000+  
Dictionary : 10,000,000+  
Fix Error Google Playstore : 1,000+  
D. H. Lawrence Poems FREE : 1,000+  
Bilingual Dictionary Audio App : 5,000+  
DM Screen : 10,000+  
wikiHow: how to do anything : 1,000,000+  
Dr. Doug's Tips : 1,000+  
Bible du Semeur-BDS (French) : 50,000+  
La citadelle du musulman : 50,000+  
DV 2019 Entry Guide : 10,000+  
DV 2019 - EDV Photo & Form : 50,000+  
DV 2018 Winners Guide : 1,000+  
EB Annual Meetings : 1,000+  
EC - AP & Telangana : 5,000+  
TN Patta Citta & EC : 10,000+  
AP Stamps and Registration : 10,000+  
CompactiMa EC pH Calibration : 100+  
EGW Writings 2 : 100,000+  
EGW Writings : 1,000,000+  
Bible with EGW Comments : 100,000+  
My Little Pony AR Guide : 1,000,000+  
SDA Sabbath School Quarterly : 500,000+  
Duaa Ek Ibaadat : 5,000+  
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+  
EP Research Service : 1,000+  
Hymnes et Louanges : 100,000+

```

EU Charter : 1,000+
EU Data Protection : 1,000+
EU IP Codes : 100+
EW PDF : 5+
BakaReader EX : 100,000+
EZ Quran : 50,000+
FA Part 1 & 2 Past Papers Solved Free – Offline : 5,000+
La Fe de Jesus : 1,000+
La Fe de Jesús : 500+
Le Fe de Jesus : 500+
Florida – Pocket Brainbook : 1,000+
Florida Statutes (FL Code) : 1,000+
English To Shona Dictionary : 10,000+
Greek Bible FP (Audio) : 1,000+
Golden Dictionary (FR-AR) : 500,000+
Fanfic-FR : 5,000+
Bulgarian French Dictionary Fr : 10,000+
Chemin (fr) : 1,000+
The SCP Foundation DB fr nn5n : 1,000+

```

In the above, we looked at the number of installs per app in this category. The categories include libraries, dictionaries, software for processing and reading ebooks, and so on. Libraries are very frequent. A few very popular apps may still skew the average. Let us test whether that is the case and if so, remove them prior to further analysis.

```

In [32]: for app in android_final:
          if app[1] == 'BOOKS_AND_REFERENCE' and (app[5] == '1,000,000,000+'
                                                    or app[5] == '500,000,000+'
                                                    or app[5] == '100,000,000+'):
          # Restricting apps displayed in the print command based on their number
          of installs
          print(app[0], ': ', app[5])

```

```

Google Play Books : 1,000,000,000+
Bible : 100,000,000+
Amazon Kindle : 100,000,000+
Wattpad 📖 Free Books : 100,000,000+
Audiobooks from Audible : 100,000,000+

```

The code above yields not too many highly popular apps. It seems as if it would be a good idea to focus on the apps that have between 1,000,000 and 100,000,000 downloads, as they appear to be relatively middle of the road.

```
In [33]: 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+
eBook: 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+
```

There are a lot of software types for processing and reading ebooks, and collections of libraries. We may suggest more similar apps although that market seems relatively saturated. Thus maybe we will instead try to push slightly different apps, to introduce some innovation that may prove more profitable than creating more apps in the same old (albeit successful) categories. A lot of the apps relate to the book Quran, thus indicating that building an app around a popular book may be profitable. Again, as on the App Store, perhaps making apps around fluff relating to a new popular book may be a way to go. As an example, audio versions of the book may be of interest to customers.

## 4. Summary and Conclusion

This project focused on the App Store and Google Play free mobile apps. The goal was to recommend a new app profile profitable for both markets.

The niche of 'books/reference' seemed profitable on both markets, and in particular an app based on a new popular book. In addition, extra products based on the book such as daily quotes from the book may create new interest. There are many libraries available already, so that market may be a bit saturated. Therefore, creating fluff-based apps such as quizzes about the book may prove profitable for both markets.

(Author: Aleksandra Zaba; version date: July 21, 2020; guided project, part of Dataquest's Data Scientist in Python specialization)