Analyzing Popular App Categories on Google Play Store

In this project, Our goal to figure out what types of apps tend to be popular on the google play store. we work for a a company that makes free app and earn money through ads. By understanding which app category are in high demand, we can help our developers create apps that attract more users and generate more revenue. well analyze data from the google play store to identify patterns and preference among users. this way we can make smater desicion about the kinds of apps we develop.

In [42]:
```python
import pandas as pd
import matplotlib.pyplot as plt
```

In [43]:
```python
# Read the dataset into pandas dataframe. objects
android_df = pd.read_csv('googleplaystore.csv')
android_df
```

Out[43]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Cu
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018	
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	June 8, 2018	
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	
...	...	...	...	...	...	...	...	...	...	...	...	
10836	Sya9a Maroc - FR	FAMILY	4.5	38	53M	5,000+	Free	0	Everyone	Education	July 25, 2017	
10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3.6M	100+	Free	0	Everyone	Education	July 6, 2018	
10838	Parkinson Exercices FR	MEDICAL	NaN	3	9.5M	1,000+	Free	0	Everyone	Medical	January 20, 2017	
10839	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5	114	Varies with device	1,000+	Free	0	Mature 17+	Books & Reference	January 19, 2015	
10840	iHoroscope - 2018 Daily Horoscope & Astrology	LIFESTYLE	4.5	398307	19M	10,000,000+	Free	0	Everyone	Lifestyle	July 25, 2018	

10841 rows × 13 columns

In [44]:

```python
# Explore the data using pandas methods
android_df.head()
```

Out[44]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Androi Ve
0	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. and u
1	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. and u
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	1.2.4	4.0. and u
3	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 an u
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.4 an u

In [45]:

```python
android_df["Category"].value_counts()
```

Out[45]:

```
FAMILY                 1972
GAME                   1144
TOOLS                   843
MEDICAL                 463
BUSINESS                460
PRODUCTIVITY            424
PERSONALIZATION         392
COMMUNICATION           387
SPORTS                  384
LIFESTYLE               382
FINANCE                 366
HEALTH_AND_FITNESS      341
PHOTOGRAPHY             335
SOCIAL                  295
NEWS_AND_MAGAZINES      283
SHOPPING                260
TRAVEL_AND_LOCAL        258
DATING                  234
BOOKS_AND_REFERENCE     231
VIDEO_PLAYERS           175
EDUCATION               156
ENTERTAINMENT           149
MAPS_AND_NAVIGATION     137
FOOD_AND_DRINK          127
HOUSE_AND_HOME           88
LIBRARIES_AND_DEMO       85
AUTO_AND_VEHICLES        85
WEATHER                  82
ART_AND_DESIGN           65
EVENTS                   64
PARENTING                60
COMICS                   60
BEAUTY                   53
1.9                       1
Name: Category, dtype: int64
```

In [46]:
```
android_df[android_df["Category"] == "1.9"]
```

Out[46]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
10472	Life Made WI-Fi Touchscreen Photo Frame	1.9	19.0	3.0M	1,000+	Free	0	Everyone	NaN	February 11, 2018	1.0.19	4.0 and up	NaN

Clean the data

In [47]:
```
android_df[android_df["Category"] == "1.9"].values
```

Out[47]:
```
array([['Life Made WI-Fi Touchscreen Photo Frame', '1.9', 19.0, '3.0M',
        '1,000+', 'Free', '0', 'Everyone', nan, 'February 11, 2018',
        '1.0.19', '4.0 and up', nan]], dtype=object)
```

In [48]:
```
Clean_lst = ['Life Made WI-Fi Touchscreen Photo Frame', 'LIFESTYLE','1.9', 19.0,
             '1,000+', 'Free', '0', 'Everyone', 'LIFESTYLE', 'Febuary 11', '2018',
             '1.0.19', '4.0 and up']
Clean_lst
```

Out[48]:
```
['Life Made WI-Fi Touchscreen Photo Frame',
 'LIFESTYLE',
 '1.9',
 19.0,
 '1,000+',
 'Free',
 '0',
 'Everyone',
 'LIFESTYLE',
 'Febuary 11',
 '2018',
 '1.0.19',
 '4.0 and up']
```

In [49]:
```
android_df[android_df["Category"] == "1.9"] = Clean_lst
```

In [50]:
```
android_category = android_df["Category"].value_counts()
android_category
```

Out[50]:
```
FAMILY                 1972
GAME                   1144
TOOLS                   843
MEDICAL                 463
BUSINESS                460
PRODUCTIVITY            424
PERSONALIZATION         392
COMMUNICATION           387
SPORTS                  384
LIFESTYLE               383
FINANCE                 366
HEALTH_AND_FITNESS      341
PHOTOGRAPHY             335
SOCIAL                  295
NEWS_AND_MAGAZINES      283
SHOPPING                260
TRAVEL_AND_LOCAL        258
DATING                  234
BOOKS_AND_REFERENCE     231
VIDEO_PLAYERS           175
EDUCATION               156
ENTERTAINMENT           149
MAPS_AND_NAVIGATION     137
FOOD_AND_DRINK          127
HOUSE_AND_HOME           88
AUTO_AND_VEHICLES        85
LIBRARIES_AND_DEMO       85
WEATHER                  82
ART_AND_DESIGN           65
EVENTS                   64
PARENTING                60
COMICS                   60
BEAUTY                   53
Name: Category, dtype: int64
```

In [51]:
```
app_count = android_df["App"].value_counts()
"Instagram" in app_count[app_count > 1].index
```

Out[51]:
```
True
```

In [52]:
```
app_count
```

Out[52]:
```
ROBLOX                                                9
CBS Sports App - Scores, News, Stats & Watch Live     8
ESPN                                                  7
Duolingo: Learn Languages Free                        7
Candy Crush Saga                                      7
                                                     ..
Meet U - Get Friends for Snapchat, Kik & Instagram    1
U-Report                                              1
U of I Community Credit Union                         1
Waiting For U Launcher Theme                          1
iHoroscope - 2018 Daily Horoscope & Astrology         1
Name: App, Length: 9660, dtype: int64
```

Removing Duplicate Entries

Part One

if we explore the google play data set long enough, we'll find that some apps have more than one entry. For instance, the application instagram has four entries:

In [53]:
```python
android_df[android_df["App"] == "Instagram"]
```

Out[53]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
2545	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
2604	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
2611	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
3909	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

In [54]:
```python
# Number of dublicates apps name
android_df.duplicated(subset=["App"], keep="first").sum()
```

Out[54]:
```
1181
```

In [55]:
```python
duplicate_apps_df = android_df[android_df.duplicated(subset=["App"], keep=False)]

# Number of duplicate_apps
num_duplicate_apps = duplicate_apps_df["App"].nunique()
num_duplicate_apps
```

Out[55]:
```
798
```

In [56]:
```python
duplicate_apps_df[duplicate_apps_df['App'] == "Instagram"]
```

Out[56]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
2545	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
2604	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
2611	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
3909	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

In [57]:
```python
android_df.shape[0]
```

Out[57]:
```
10841
```

In [58]:
```python
10841 - 1181
```

Out[58]:
```
9660
```

In [59]:
```python
android_df["App"].nunique()
```

Out[59]:
```
9660
```

We don't want 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 we could probably find a better way.

If you examine the rows we printed two cells above for the instagram app, the main difference happens on the fourth position of each row, which corresponds to the number of reviews. The different number shows that the data was collected at different times. We can use this to build a criterion for keeping rows. We wont remove rows randomly, but rather we'll keep the rows that have the highest number of reviews because the higher the number of reviews, the most reliable the ratings.

In [60]:
```
# Group by 'App and get the maximum number of reviews for each app'
reviews_max = android_df.groupby('App')['Reviews'].max()
```

In [61]:
```
reviews_max["Instagram"]
```

Out[61]:
```
'66577446'
```

In [62]:
```
duplicate_apps_df[duplicate_apps_df['App'] == "Instagram"]
```

Out[62]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
2545	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
2604	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
2611	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
3909	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

Now, let's use the reviews_max dictionary to remove the duplicates. For the duplicate cases, we'll only keep the entries with the highest number of reviews. In the code cell below:

. We start by initializing two empty lists, android_clean and already_added . We add the current row (app) to the android_clean list, and the number of reviews. to the already_added list if: . The number of reviews of 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 were 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 [63]:
```
# Create an empty List to store cleaned data
android_clean = []

# Create an empty List to keep track of already added apps
already_added = []
# Iterate through each row in the DataFrame
for index, row in android_df.iterrows():
    name = row['App']
    n_reviews = row['Reviews']
    # Check if the current app has the maximum number of reviews and has not
    if (reviews_max[name] == n_reviews) and (name not in already_added):
        android_clean.append(row) # add the app name to the list of already added
        already_added.append(name) # add the app name to the list of already added
```

In [64]:
```
len(android_clean)
```

Out[64]:
```
9660
```

In [65]:
```
android_clean = pd.DataFrame(android_clean)
```

Removing Non-English Apps

Part One

If you explore the data sets enough, you'll notice the names of some of the apps suggest they are not directed toward an English-Speaking audience. Below, we see a couple of examples from both data sets:

In [66]:
```python
def is_english(app_name):
    lst = []
    for i in app_name:
        if ord(i) > 127:
            lst.append(False)
        else:
            lst.append(True)

        check = set(lst)

        if False in check:
            return False
        else:
            return True
```

In [67]:
```python
is_english("Instagram😜")
```

Out[67]:
```
True
```

In [68]:
```python
is_english("أنستغرام")
```

Out[68]:
```
False
```

Part two

To minimize the impact of data loss, we'll only remove an app if its name has more than three non-ASCII characters

In [69]:
```python
def is_english(app_name):
    lst = []
    for i in app_name:
        if ord(i) > 127:
            lst.append(False)
        else:
            lst.append(True)
    non_ascii = 0
    for j in lst:
        if j == False:
            non_ascii += 1

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

In [70]:
```python
is_english("Instagram 😜😜😜😜")
```

Out[70]:
```
False
```

In [71]:
```python
android_english = android_clean[android_clean["App"].apply(is_english)]
```

In [72]:
```
android_english.head()
```

Out[72]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Androi Ve
0	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. and u
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	1.2.4	4.0. and u
3	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 an u
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.4 an u
5	Paper flowers instructions	ART_AND_DESIGN	4.4	167	5.6M	50,000+	Free	0	Everyone	Art & Design	March 26, 2017	1.0	2.3 an u

Isolating The Free Apps

As we mentioned in the introduction, we only build apps that are free to download and install, and our main source of revenue consists of in-app ads. Our data sets contain both free and non free apps, and we'll need to isolate only the free apps for our analysis. Below we isolate the free apps for both our data sets.

In [73]:
```
android_english["Price"].unique()
```

Out[73]:
```
array(['0', '$4.99', '$3.99', '$6.99', '$1.49', '$2.99', '$7.99', '$5.99',
       '$3.49', '$1.99', '$9.99', '$7.49', '$0.99', '$9.00', '$5.49',
       '$10.00', '$11.99', '$79.99', '$16.99', '$14.99', '$1.00',
       '$29.99', '$12.99', '$2.49', '$24.99', '$10.99', '$1.50', '$19.99',
       '$15.99', '$33.99', '$74.99', '$39.99', '$3.95', '$4.49', '$1.70',
       '$8.99', '$2.00', '$3.88', '$25.99', '$399.99', '$17.99',
       '$400.00', '$3.02', '$1.76', '$4.84', '$4.77', '$1.61', '$2.50',
       '$1.59', '$6.49', '$1.29', '$5.00', '$13.99', '$299.99', '$379.99',
       '$37.99', '$18.99', '$389.99', '$19.90', '$8.49', '$1.75',
       '$14.00', '$4.85', '$46.99', '$109.99', '$154.99', '$3.08',
       '$2.59', '$4.80', '$1.96', '$19.40', '$3.90', '$4.59', '$15.46',
       '$3.04', '$4.29', '$2.60', '$3.28', '$4.60', '$28.99', '$2.95',
       '$2.90', '$1.97', '$200.00', '$89.99', '$2.56', '$30.99', '$3.61',
       '$394.99', '$1.26', 'Everyone', '$1.20', '$1.04'], dtype=object)
```

In [74]:
```
android_final = android_english[android_english["Price"] == "0"]
```

In [75]:

```python
android_final.head()
```

Out[75]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Androi Ve
0	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. and u
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	1.2.4	4.0. and u
3	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 an u
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.4 an u
5	Paper flowers instructions	ART_AND_DESIGN	4.4	167	5.6M	50,000+	Free	0	Everyone	Art & Design	March 26, 2017	1.0	2.3 an u

Most Common Apps by Genre

In [76]:

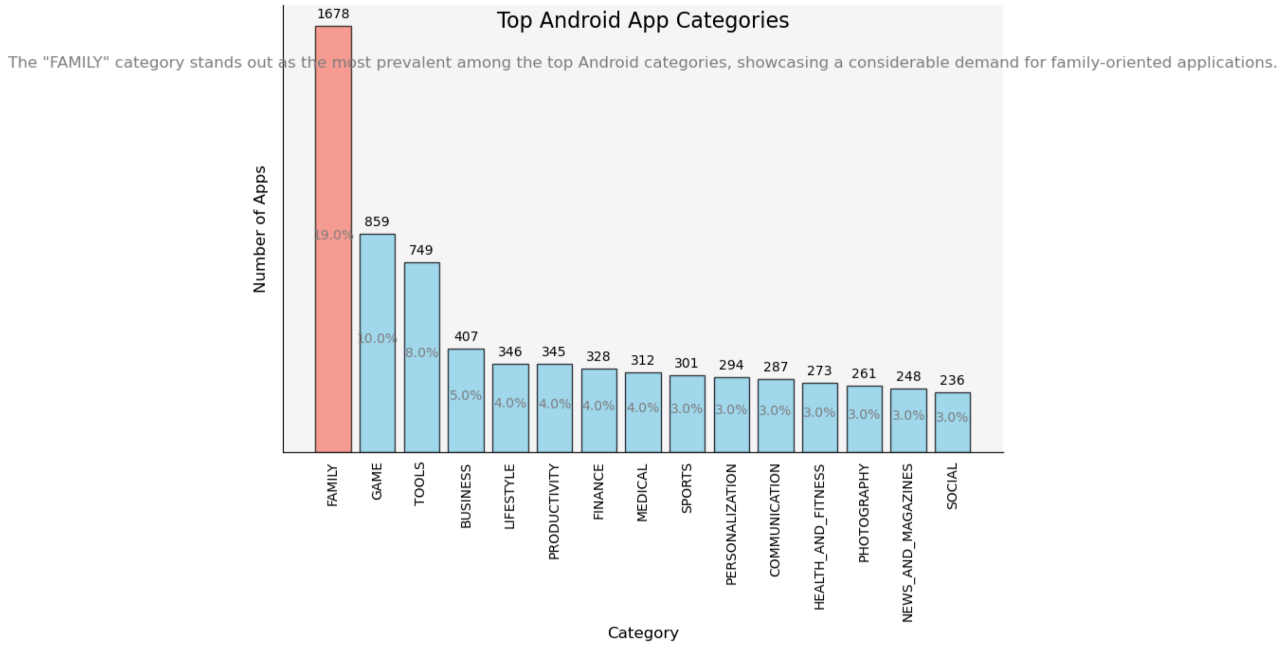
```python
android_final["Category"].value_counts(normalize = True)*100
```

Out[76]:

```
FAMILY                 18.934778
GAME                    9.693072
TOOLS                   8.451817
BUSINESS                4.592643
LIFESTYLE               3.904311
PRODUCTIVITY            3.893026
FINANCE                 3.701196
MEDICAL                 3.520650
SPORTS                  3.396524
PERSONALIZATION         3.317536
COMMUNICATION           3.238547
HEALTH_AND_FITNESS      3.080569
PHOTOGRAPHY             2.945159
NEWS_AND_MAGAZINES      2.798465
SOCIAL                  2.663056
TRAVEL_AND_LOCAL        2.335816
SHOPPING                2.245543
BOOKS_AND_REFERENCE     2.143986
DATING                  1.861882
VIDEO_PLAYERS           1.794177
MAPS_AND_NAVIGATION     1.399233
FOOD_AND_DRINK          1.241255
EDUCATION               1.173550
ENTERTAINMENT           0.959151
LIBRARIES_AND_DEMO      0.936583
AUTO_AND_VEHICLES       0.925299
HOUSE_AND_HOME          0.823742
WEATHER                 0.801174
EVENTS                  0.710900
PARENTING               0.654480
ART_AND_DESIGN          0.643196
COMICS                  0.620627
BEAUTY                  0.598059
Name: Category, dtype: float64
```

In [85]:

```python

# data
categories = android_final["Category"].value_counts().index[:15]
counts = android_final["Category"].value_counts().values[:15]
percentage = round(android_final["Category"].value_counts(normalize=True)* 100)

#create stylish bar chart
plt.figure(figsize=(12, 7))
bars = plt.bar(categories, counts, color='skyblue', alpha=0.75, edgecolor='black')
plt.xticks(rotation=90, fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--',alpha=0.7)
plt.grid(axis='x',linestyle='')
plt.xticks(fontsize=12) # customized tick labels
plt.yticks(range(0, 60000000, 100000000), [],fontsize=12) #customized tick labels and columns
plt.tick_params(bottom = 0, left = 0)

#find the category with the highest count
max_count_category = categories[counts.argmax()]

#highlight the bar for the category with the highest count
max_count_index = list(categories).index(max_count_category)
bars[max_count_index].set_color('salmon')
bars[max_count_index].set_edgecolor('black')

#adding data labels and percentage inside each bar
for bar, perc in zip(bars, percentage):
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, height + 30, f'{int(height)}', ha='center', fontsize=10)
    plt.text(bar.get_x() + bar.get_width() / 2, height / 2, f'{perc}%', ha='center', fontsize=10, color='

# Adding conclusion inside the chart
plt.text(0.5, 0.86, 'The "FAMILY" category stands out as the most prevalent among the top Android categor
         horizontalalignment='center', fontsize=12, transform=plt.gca().transAxes, color='gray')

# Adding chart title inside the chart
plt.text(0.5, 0.95, 'Top Android App Categories', horizontalalignment='center', fontsize=16, transform=pl

# Remove tick marks and spines
plt.tick_params(bottom=False, left=False)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

# Set axis labels and ticks
plt.xlabel('Category', fontsize=12, color='black')
plt.ylabel('Number of Apps', fontsize=12, color='black')
plt.xticks(rotation=90, fontsize=10, color='black')
plt.yticks(fontsize=10, color='black')

# Set background color
plt.gca().set_facecolor('#f7f7f7')

plt.tight_layout()  # Adjust layout to prevent clipping

plt.show()
```



```
In [86]:  android_final[android_final["Category"] == "FAMILY"]
```

Out[86]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Androi Ve
2017	Jewels Crush- Match 3 Puzzle	FAMILY	4.4	14774	19M	1,000,000+	Free	0	Everyone	Casual;Brain Games	July 23, 2018	1.9.3901	4.0. and u
2018	Coloring & Learn	FAMILY	4.4	12753	51M	5,000,000+	Free	0	Everyone	Educational;Creativity	July 17, 2018	1.49	4.0. and u
2019	Mahjong	FAMILY	4.5	33983	22M	5,000,000+	Free	0	Everyone	Puzzle;Brain Games	August 2, 2018	1.24.3181	4.0. and u
2020	Super ABC! Learning games for kids! Preschool ...	FAMILY	4.6	20267	46M	1,000,000+	Free	0	Everyone	Educational;Education	July 16, 2018	1.1.6.7	4.1 an u
2021	Toy Pop Cubes	FAMILY	4.5	5761	21M	1,000,000+	Free	0	Everyone	Casual;Brain Games	July 4, 2018	1.8.3181	4.0. and u
...	...	...	...	...	...	...	...	...	...	...	...	...	.
10821	Poop FR	FAMILY	NaN	6	2.5M	50+	Free	0	Everyone	Entertainment	May 29, 2018	1.0	4.0. and u
10827	Fr Agnel Ambarnath	FAMILY	4.2	117	13M	5,000+	Free	0	Everyone	Education	June 13, 2018	2.0.20	4.0. and u
10834	FR Calculator	FAMILY	4.0	7	2.6M	500+	Free	0	Everyone	Education	June 18, 2017	1.0.0	4.1 an u
10836	Sya9a Maroc - FR	FAMILY	4.5	38	53M	5,000+	Free	0	Everyone	Education	July 25, 2017	1.48	4.1 an u
10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3.6M	100+	Free	0	Everyone	Education	July 6, 2018	1.0	4.1 an u

1678 rows × 13 columns

Most Popular Apps by Genre on Google Play

For the google play market, we actually have data about the number of installs, so we should be able to get a cleaner picture about genre popularity. However, the installs numbers don't seem precise enough - we can see that most values are open ended(100+,

In [88]:
```
android_final["Installs"].value_counts(normalize = True)* 100
```

Out[88]:
```
1,000,000+        15.741368
100,000+          11.554954
10,000,000+       10.516813
10,000+           10.200858
1,000+             8.395396
100+               6.917174
5,000,000+         6.838186
500,000+           5.574362
50,000+            4.773189
5,000+             4.513654
10+                3.543218
500+               3.249831
50,000,000+        2.290679
100,000,000+       2.121417
50+                1.918303
5+                 0.789889
1+                 0.507786
500,000,000+       0.270819
1,000,000,000+     0.225683
0+                 0.045137
0                  0.011284
Name: Installs, dtype: float64
```

In [89]:
```
# Remove non-numeric characters and convert to integers
android_final["Installs_int"] = android_final["Installs"].str.replace("[^\d]", "", regex=True).astype(int)
```

```
C:\Users\dell\AppData\Local\Temp\ipykernel_13780\2758109644.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  android_final["Installs_int"] = android_final["Installs"].str.replace("[^\d]", "", regex=True).astype(int)
```

In [90]:
```
android_final["Installs_int"]
```

Out[90]:
```
0           10000
2         5000000
3        50000000
4          100000
5           50000
           ...
10836        5000
10837         100
10838        1000
10839        1000
10840    10000000
Name: Installs_int, Length: 8862, dtype: int32
```

In [91]:
```python
install_frq = android_final["Installs_int"].value_counts().sort_index()
install_frq = install_frq[install_frq.index > 500]
install_frq
```

Out[91]:
```
1000           744
5000           400
10000          904
50000          423
100000        1024
500000         494
1000000       1395
5000000        606
10000000       932
50000000       203
100000000      188
500000000       24
1000000000      20
Name: Installs_int, dtype: int64
```

In [97]:
```python
install_frq_per = round(android_final["Installs_int"].value_counts(normalize=True)*100)
install_frq_per = install_frq_per[install_frq_per.index > 500]
install_frq_per
```

Out[97]:
```
1000000       16.0
100000        12.0
10000000      11.0
10000         10.0
1000           8.0
5000000        7.0
500000         6.0
50000          5.0
5000           5.0
50000000       2.0
100000000      2.0
500000000      0.0
1000000000     0.0
Name: Installs_int, dtype: float64
```

In [98]:
```python
install_frq
```

Out[98]:
```
1K       744
5K       400
10K      904
50K      423
100K    1024
500K     494
1M      1395
5M       606
10M      932
50M      203
100M     188
500M      24
1B        20
Name: Installs_int, dtype: int64
```

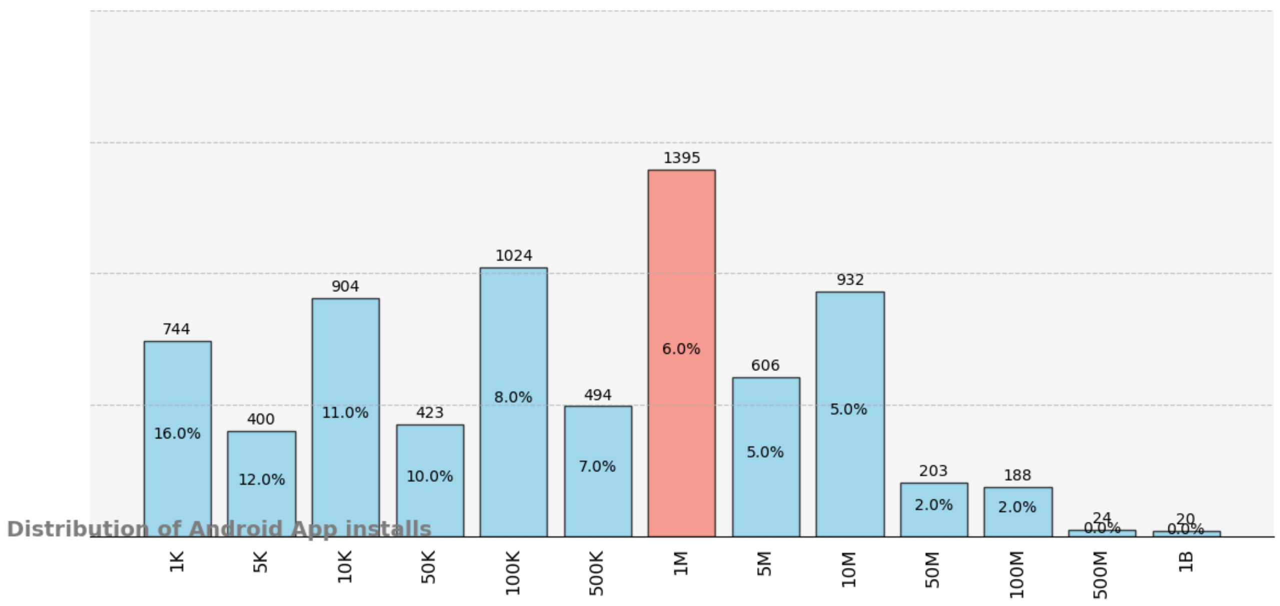
In [99]:
```python
# alphanumeric units
def alphanumeric_units(value):
    if value >= 1e9:
        return f'{value / 1e9:.0f}B'
    elif value >= 1e6:
        return f'{value / 1e6:.0f}M'
    elif value >= 1e3:
        return f'{value / 1e3:.0f}K'
    else:
        return f'{value:.0f}'
```

```
In [100]:  install_frq.index = install_frq.index.map(alphanumeric_units)
           install_frq
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[100], line 1
----> 1 install_frq.index = install_frq.index.map(alphanumeric_units)
      2 install_frq

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:6361, in Index.map(self, mapper, na_action)
   6341 """
   6342 Map values using an input mapping or function.
   6343
   (...)
   6357     a MultiIndex will be returned.
   6358 """
   6359 from pandas.core.indexes.multi import MultiIndex
-> 6361 new_values = self._map_values(mapper, na_action=na_action)
   6363 # we can return a MultiIndex
   6364 if new_values.size and isinstance(new_values[0], tuple):

File ~\anaconda3\Lib\site-packages\pandas\core\base.py:890, in IndexOpsMixin._map_values(self, mapper, na_ac
tion)
    887         raise ValueError(msg)
    889 # mapper is a function
--> 890 new_values = map_f(values, mapper)
    892 return new_values

File ~\anaconda3\Lib\site-packages\pandas\_libs\lib.pyx:2924, in pandas._libs.lib.map_infer()

Cell In[99], line 3, in alphanumeric_units(value)
      2 def alphanumeric_units(value):
----> 3     if value >= 1e9:
      4         return f'{value / 1e9:.0f}B'
      5     elif value >= 1e6:

TypeError: '>=' not supported between instances of 'str' and 'float'
```

In [101]:

```
# Data
categories = install_frq.index
counts = install_frq.values
percentage = install_frq_per.values

#create stylish bar chart
plt.figure(figsize=(12, 7))
bars = plt.bar(categories, counts, color='skyblue', alpha=0.75, edgecolor='black')
plt.xticks(rotation=90, fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='y' , linestyle='--',alpha=0.7)
plt.grid(axis='x', linestyle='')
plt.xticks(fontsize=12) # customized tick labels
plt.yticks(range(0,2500,500), [],fontsize=12) # customized trick labels and columns
plt.tick_params(bottom = 0, left = 0)

# find the category with the highest count
max_count_category = categories[counts.argmax()]

# Highlight the bar of the category with the highest count
max_count_index = list(categories).index(max_count_category)
bars[max_count_index].set_color('salmon')
bars[max_count_index].set_edgecolor('black')

#adding data labels and percentages inside each bar
for bar, perc in zip(bars, percentage):
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, height + 28, '%d' % int(height), ha='center', fontsize=10
    plt.text(bar.get_x() + bar.get_width() / 2, height / 2, f'{perc}%', ha='center', fontsize=10)

# adding a background color
ax = plt.gca()
ax.set_facecolor('#f7f7f7')

#adding chart title inside the chart
plt.text(0.5, 0.94, 'Distribution of Android App installs', horizontalalignment='center',
         color='gray', fontsize=14, weight='bold')

# adding cunclusion in the chart
plt.text(0.5, -0.35, "The data shows a long-tail distribution of app installations on the Google Play Stor
         horizontalalignment='center', fontsize=11, transform=plt.gca().transAxes)
# Remove spines
for i in ["top","right","left"]:
    plt.gca().spines[i].set_visible(False)

plt.tight_layout() # adjust layout to prevent clipping

plt.show()

```



The data shows a long-tail distribution of app installations on the Google Play Store.

In [102]:

```
categories_android = android_final["Category"].unique()
categories_android
```

Out[102]:

```
array(['ART_AND_DESIGN', 'AUTO_AND_VEHICLES', 'BEAUTY',
       'BOOKS_AND_REFERENCE', 'BUSINESS', 'COMICS', 'COMMUNICATION',
       'DATING', 'EDUCATION', 'ENTERTAINMENT', 'EVENTS', 'FINANCE',
       'FOOD_AND_DRINK', 'HEALTH_AND_FITNESS', 'HOUSE_AND_HOME',
       'LIBRARIES_AND_DEMO', 'LIFESTYLE', 'GAME', 'FAMILY', 'MEDICAL',
       'SOCIAL', 'SHOPPING', 'PHOTOGRAPHY', 'SPORTS', 'TRAVEL_AND_LOCAL',
       'TOOLS', 'PERSONALIZATION', 'PRODUCTIVITY', 'PARENTING', 'WEATHER',
       'VIDEO_PLAYERS', 'NEWS_AND_MAGAZINES', 'MAPS_AND_NAVIGATION'],
      dtype=object)
```

In [103]:
```
pd.pivot_table(android_final, values='Installs_int', index='Category', aggfunc='mean')
```

Out[103]:

Category	Installs_int
ART_AND_DESIGN	1.986335e+06
AUTO_AND_VEHICLES	6.473178e+05
BEAUTY	5.131519e+05
BOOKS_AND_REFERENCE	8.767812e+06
BUSINESS	1.712290e+06
COMICS	8.176573e+05
COMMUNICATION	3.845612e+07
DATING	8.540288e+05
EDUCATION	1.820673e+06
ENTERTAINMENT	1.164071e+07
EVENTS	2.535422e+05
FAMILY	3.694276e+06
FINANCE	1.387692e+06
FOOD_AND_DRINK	1.924898e+06
GAME	1.556097e+07
HEALTH_AND_FITNESS	4.188822e+06
HOUSE_AND_HOME	1.331541e+06
LIBRARIES_AND_DEMO	6.385037e+05
LIFESTYLE	1.437816e+06
MAPS_AND_NAVIGATION	4.056942e+06
MEDICAL	1.206165e+05
NEWS_AND_MAGAZINES	9.549178e+06
PARENTING	5.426036e+05
PERSONALIZATION	5.201483e+06
PHOTOGRAPHY	1.780563e+07
PRODUCTIVITY	1.678733e+07
SHOPPING	7.036877e+06
SOCIAL	2.325365e+07
SPORTS	3.638640e+06
TOOLS	1.068230e+07
TRAVEL_AND_LOCAL	1.398408e+07
VIDEO_PLAYERS	2.472787e+07
WEATHER	5.074486e+06

In [104]:
```
# Display Dataframe without scientific notation
pd.options.display.float_format = '{:,0f}'.format
```

In [105]:
```
categories_installs = pd.pivot_table(android_final, values='Installs_int', index='Category', aggfunc='mea
categories_installs = categories_installs.sort_values(by="Installs_int", ascending=False)
categories_installs = categories_installs["Installs_int"]

```

In [106]:
```
categories_installs = categories_installs.map('{:,.0f}'.format)
```

In [107]:
```
categories_installs
```

Out[107]:
```
Category
COMMUNICATION          38,456,119
VIDEO_PLAYERS          24,727,872
SOCIAL                 23,253,652
PHOTOGRAPHY            17,805,628
PRODUCTIVITY           16,787,331
GAME                   15,560,966
TRAVEL_AND_LOCAL       13,984,078
ENTERTAINMENT          11,640,706
TOOLS                  10,682,301
NEWS_AND_MAGAZINES      9,549,178
BOOKS_AND_REFERENCE     8,767,812
SHOPPING                7,036,877
PERSONALIZATION         5,201,483
WEATHER                 5,074,486
HEALTH_AND_FITNESS      4,188,822
MAPS_AND_NAVIGATION     4,056,942
FAMILY                  3,694,276
SPORTS                  3,638,640
ART_AND_DESIGN          1,986,335
FOOD_AND_DRINK          1,924,898
EDUCATION               1,820,673
BUSINESS                1,712,290
LIFESTYLE               1,437,816
FINANCE                 1,387,692
HOUSE_AND_HOME          1,331,541
DATING                    854,029
COMICS                    817,657
AUTO_AND_VEHICLES         647,318
LIBRARIES_AND_DEMO        638,504
PARENTING                 542,604
BEAUTY                    513,152
EVENTS                    253,542
MEDICAL                   120,616
Name: Installs_int, dtype: object
```

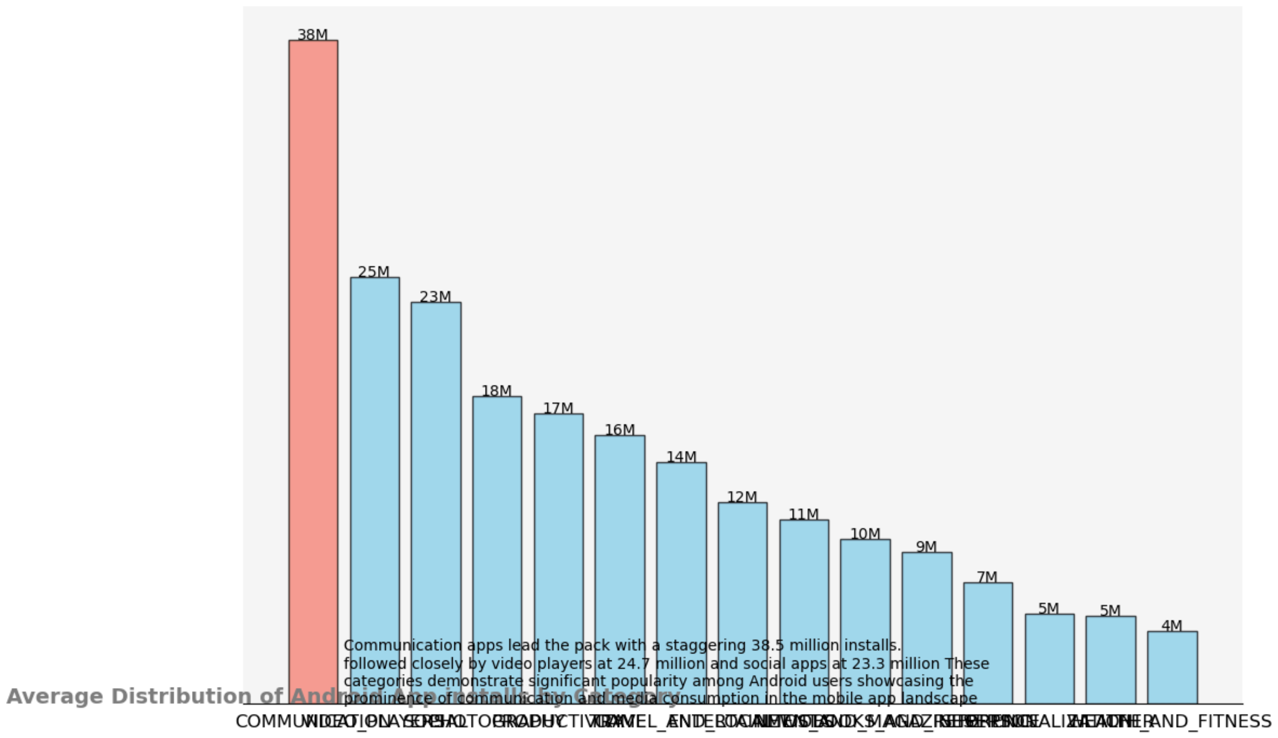
In [108]:
```python
# alphanumeric units
def alphanumeric_units(value):
    if value >= 1e9:
        return f'{value / 1e9:.0f}B'
    elif value >= 1e6:
        return f'{value / 1e6:.0f}M'
    elif value >= 1e3:
        return f'{value / 1e3:.0f}K'
    else:
        return f'{value:.0f}'
```

```
In [112]:
categories_installs = categories_installs.str.replace(',', '').astype(float)  # Remove commas and convert
categories_installs_units = categories_installs.map(alphanumeric_units)
categories_installs_units
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[112], line 1
----> 1 categories_installs = categories_installs.str.replace(',', '').astype(float)  # Remove commas and co
nvert to float
      2 categories_installs_units = categories_installs.map(alphanumeric_units)
      3 categories_installs_units

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:5902, in NDFrame.__getattr__(self, name)
   5895 if (
   5896     name not in self._internal_names_set
   5897     and name not in self._metadata
   5898     and name not in self._accessors
   5899     and self._info_axis._can_hold_identifiers_and_holds_name(name)
   5900 ):
   5901     return self[name]
-> 5902 return object.__getattribute__(self, name)

File ~\anaconda3\Lib\site-packages\pandas\core\accessor.py:182, in CachedAccessor.__get__(self, obj, cls)
    179 if obj is None:
    180     # we're accessing the attribute of the class, i.e., Dataset.geo
    181     return self._accessor
--> 182 accessor_obj = self._accessor(obj)
    183 # Replace the property with the accessor object. Inspired by:
    184 # https://www.pydanny.com/cached-property.html (https://www.pydanny.com/cached-property.html)
    185 # We need to use object.__setattr__ because we overwrite __setattr__ on
    186 # NDFrame
    187 object.__setattr__(obj, self._name, accessor_obj)

File ~\anaconda3\Lib\site-packages\pandas\core\strings\accessor.py:181, in StringMethods.__init__(self, dat
a)
    178 def __init__(self, data) -> None:
    179     from pandas.core.arrays.string_ import StringDtype
--> 181     self._inferred_dtype = self._validate(data)
    182     self._is_categorical = is_categorical_dtype(data.dtype)
    183     self._is_string = isinstance(data.dtype, StringDtype)

File ~\anaconda3\Lib\site-packages\pandas\core\strings\accessor.py:235, in StringMethods._validate(data)
    232 inferred_dtype = lib.infer_dtype(values, skipna=True)
    234 if inferred_dtype not in allowed_types:
--> 235     raise AttributeError("Can only use .str accessor with string values!")
    236 return inferred_dtype

AttributeError: Can only use .str accessor with string values!
```

In [115]:

```
# data
categories = categories_installs.index[:15]
counts = categories_installs.values[:15]

#create stylish bar chart
plt.figure(figsize=(12, 7))
bars = plt.bar(categories, counts, color='skyblue', alpha=0.75, edgecolor='black')
plt.yticks(rotation=90, fontsize=12)
plt.yticks(rotation=90,fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--',alpha=0.7)
plt.grid(axis='x',linestyle='')
plt.xticks(fontsize=12) # customized tick labels
plt.yticks(range(0, 60000000, 100000000), [],fontsize=12) #customized tick labels and columns
plt.tick_params(bottom = 0, left = 0)

#find the category with the highest count
max_count_category = categories[counts.argmax()]

#highlight the bar for the category with the highest count
max_count_index = list(categories).index(max_count_category)
bars[max_count_index].set_color('salmon')
bars[max_count_index].set_edgecolor('black')

#adding data labels and percentage inside each bar
for bar, units in zip(bars, categories_installs_units.values):
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height + 25, units , ha='center')

# adding a background color
ax = plt.gca()
ax.set_facecolor('#f7f7f7')

# adding chart title inside the chart
plt.text(0.5, 0.94, 'Average Distribution of Android App installs by Category', horizontalalignment='cent
         color='gray', fontsize=14, weight='bold')
# adding cunclusion in the chart
plt.text(0.5, 0.77, """Communication apps lead the pack with a staggering 38.5 million installs.
followed closely by video players at 24.7 million and social apps at 23.3 million These
categories demonstrate significant popularity among Android users showcasing the
prominence of communication and media consumption in the mobile app landscape""")
# Remove spines
for i in ["top","right","left"]:
    plt.gca().spines[i].set_visible(False)

plt.tight_layout() # adjust layout to prevent clipping

plt.show()
```



In [136]:

```
print(category_group.groups.keys())

# Check the unique values in the "Category" column
print(category_group['Category'].unique())

# Assuming 'COMMUNICATION' is the correct category name
try:
    # Extract the subgroup corresponding to the "COMMUNICATION" category
    communication_group = category_group.get_group("COMMUNICATION")
    # Sort the extracted subgroup by the "Installs" column
    communication_sorted = communication_group.sort_values(by="Installs")
    # Display the first few rows of the sorted DataFrame
    print(communication_sorted.head())
except KeyError:
    print("The 'COMMUNICATION' category is not found in the DataFrameGroupBy object.")
```

```
dict_keys(['ART_AND_DESIGN', 'AUTO_AND_VEHICLES', 'BEAUTY', 'BOOKS_AND_REFERENCE', 'BUSINESS', 'COMICS',
'COMMUNICATION', 'DATING', 'EDUCATION', 'ENTERTAINMENT', 'EVENTS', 'FAMILY', 'FINANCE', 'FOOD_AND_DRINK',
'GAME', 'HEALTH_AND_FITNESS', 'HOUSE_AND_HOME', 'LIBRARIES_AND_DEMO', 'LIFESTYLE', 'MAPS_AND_NAVIGATION',
'MEDICAL', 'NEWS_AND_MAGAZINES', 'PARENTING', 'PERSONALIZATION', 'PHOTOGRAPHY', 'PRODUCTIVITY', 'SHOPPIN
G', 'SOCIAL', 'SPORTS', 'TOOLS', 'TRAVEL_AND_LOCAL', 'VIDEO_PLAYERS', 'WEATHER'])
Category
ART_AND_DESIGN                [ART_AND_DESIGN]
AUTO_AND_VEHICLES          [AUTO_AND_VEHICLES]
BEAUTY                                [BEAUTY]
BOOKS_AND_REFERENCE      [BOOKS_AND_REFERENCE]
BUSINESS                            [BUSINESS]
COMICS                                [COMICS]
COMMUNICATION                  [COMMUNICATION]
DATING                                [DATING]
EDUCATION                          [EDUCATION]
ENTERTAINMENT                  [ENTERTAINMENT]
EVENTS                                [EVENTS]
FAMILY                                [FAMILY]
FINANCE                              [FINANCE]
```

In [137]:
```python
# alphanumeric units
def alphanumeric_units(value):
    if value >= 1e9:
        return f'{value / 1e9:.0f}B'
    elif value >= 1e6:
        return f'{value / 1e6:.0f}M'
    elif value >= 1e3:
        return f'{value / 1e3:.0f}K'
    else:
        return f'{value:.0f}'
```

In [138]:
```python
categories_installs.index[:15]
```

Out[138]:
```
Index(['COMMUNICATION', 'VIDEO_PLAYERS', 'SOCIAL', 'PHOTOGRAPHY',
       'PRODUCTIVITY', 'GAME', 'TRAVEL_AND_LOCAL', 'ENTERTAINMENT', 'TOOLS',
       'NEWS_AND_MAGAZINES', 'BOOKS_AND_REFERENCE', 'SHOPPING',
       'PERSONALIZATION', 'WEATHER', 'HEALTH_AND_FITNESS'],
      dtype='object', name='Category')
```

In [143]:
```python
# Convert "Installs" column to numeric
df["Installs"] = pd.to_numeric(df["Installs"], errors='coerce')

# Map the values of "Installs" to their corresponding units using the alphanumeric_units function
df["Installs_unit"] = df["Installs"].map(alphanumeric_units)

# Display the DataFrame
print(df)

```

```
                                                 App  Installs Installs_unit
336                                WhatsApp Messenger       NaN           nan
337                                 Messenger for SMS       NaN           nan
343                                         My Tele2       NaN           nan
346                         imo beta free calls and text       NaN           nan
348                                          Contacts       NaN           nan
349                                Call Free – Free Call       NaN           nan
350                               Web Browser & Explorer       NaN           nan
352                                        Browser 4G       NaN           nan
353                                 MegaFon Dashboard       NaN           nan
354                             ZenUI Dialer & Contacts       NaN           nan
355                             Cricket Visual Voicemail       NaN           nan
357                                 TracFone My Account       NaN           nan
361                                        Xperia Link™       NaN           nan
362  TouchPal Keyboard - Fun Emoji & Android Keyboard       NaN           nan
364                  Skype Lite - Free Video Call & Chat       NaN           nan
```

In [146]:
```
df = category_group.get_group("VIDEO_PLAYERS").sort_values(by="Installs_int",)
df = df[["App","Installs_int"]].head(15)
df["Installs_int_unit"] = df["Installs_int"].map(alphanumeric_units)
df
```

Out[146]:

	App	Installs_int	Installs_int_unit
10519	Art of F J Taylor	10	10
7394	CI Stream	10	10
10823	List iptv FR	100	100
5244	AJ Player	100	100
5308	AK Lodi Films	100	100
5985	Bc Vod	100	100
9237	EC MANAGER	100	100
6153	BG MUSIC PLAYER - MUSIC PLAYER	100	100
6950	Bx-WiFi-GI	100	100
5882	A-Z Screen Recorder -	500	500
6163	YourTube Video Views BG	500	500
7420	CJ Camcorder	500	500
3705	Video Wallpaper Show	500	500
8373	DG Screen Recorder	500	500
8070	CX Monthly Tech News	500	500

In [152]:
```
df = category_group.get_group("SOCIAL").sort_values(by="Installs_int",)
df = df[["App","Installs_int"]].head(15)
df["Installs_int_unit"] = df["Installs_int"].map(alphanumeric_units)
df
```

Out[152]:

	App	Installs_int	Installs_int_unit
7434	Pekalongan CJ	0	0
6257	BH Connect	1	1
10101	Amleen Ey	1	1
7147	CB Heroes	5	5
7712	C.P. CERVANTES (TOBARRA)	5	5
5951	BA 3 Banjarmasin	10	10
9113	News Dz	10	10
6470	bm-Events	10	10
9137	quran-DZ	10	10
8591	DN Blog	10	10
9651	EO RAIPUR	10	10
8579	Otto DM	10	10
9925	Reisedealz.eu	10	10
9512	Hum Ek Hain 2.02	10	10
9330	EG Way Life	50	50

In [151]:

```
df = category_group.get_group("PHOTOGRAPHY").sort_values(by="Installs_int", ascending = False)
df = df[["App","Installs_int"]].head(15)
df["Installs_int_unit"] = df["Installs_int"].map(alphanumeric_units)
df

```

Out[151]:

	App	Installs_int	Installs_int_unit
2884	Google Photos	1000000000	1B
4574	S Photo Editor - Collage Maker , Photo Collage	100000000	100M
2949	Camera360: Selfie Photo Editor with Funny Sticker	100000000	100M
2908	Retrica	100000000	100M
8307	LINE Camera - Photo editor	100000000	100M
2921	Photo Editor Pro	100000000	100M
2847	Sweet Selfie - selfie camera, beauty cam, phot...	100000000	100M
2937	BeautyPlus - Easy Photo Editor & Selfie Camera	100000000	100M
2938	PicsArt Photo Studio: Collage Maker & Pic Editor	100000000	100M
5057	AR effect	100000000	100M
2833	YouCam Makeup - Magic Selfie Makeovers	100000000	100M
2942	Z Camera - Photo Editor, Beauty Selfie, Collage	100000000	100M
2943	PhotoGrid: Video & Pic Collage Maker, Photo Ed...	100000000	100M
2944	Candy Camera - selfie, beauty camera, photo ed...	100000000	100M
2945	YouCam Perfect - Selfie Photo Editor	100000000	100M

In [155]:

```
df = category_group.get_group("PRODUCTIVITY").sort_values(by="Installs_int", ascending = False)
df = df[["App","Installs_int"]].head(15)
df["Installs_int_unit"] = df["Installs_int"].map(alphanumeric_units)
df
```

Out[155]:

	App	Installs_int	Installs_int_unit
3523	Google Drive	1000000000	1B
3450	Microsoft Word	500000000	500M
3562	Google Calendar	500000000	500M
3574	Cloud Print	500000000	500M
3473	Dropbox	500000000	500M
3524	Adobe Acrobat Reader	100000000	100M
3489	Samsung Notes	100000000	100M
3477	Google Docs	100000000	100M
3493	SwiftKey Keyboard	100000000	100M
7808	CamScanner - Phone PDF Creator	100000000	100M
3469	ES File Explorer File Manager	100000000	100M
3486	Microsoft PowerPoint	100000000	100M
3467	Google Keep	100000000	100M
3465	Microsoft OneNote	100000000	100M
3526	Google Sheets	100000000	100M

Conclusion

In summary, our analysis of Google Play Store data has revealed key insights into user preferences, guiding our app development strategy. By focusing on popular categories and maximizing user engagement, we aim to drive revenue through targeted ad placement. Continuous market monitoring will be critical for sustaining success.

In [ ]:

```

```