# Analyzing Popular App Categories on Google Play

In this project, our goal is to figure out what types of apps tend to be popular on the Google Play Store. We work for a company that makes free apps and earns money through ads. By understanding which app categories are in high demand, we can help our developers create apps that attract more users and generate more revenue. We'll analyze data from the Google Play Store to identify patterns and preferences among users. This way, we can make smarter decisions about the kinds of apps we develop.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
# Read the dataset into pandas DataFrame objects
df = pd.read_csv('googleplaystore.csv')
```

In [3]:
```
# Explore the dataset using pandas methods
df.head()
```

Out[3]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price | Content Rating | Genres |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Photo Editor & Candy Camera & Grid & ScrapBook | ART_AND_DESIGN | 4.1 | 159 | 19M | 10,000+ | Free | 0 | Everyone | Art & Design |
| **1** | Coloring book moana | ART_AND_DESIGN | 3.9 | 967 | 14M | 500,000+ | Free | 0 | Everyone | Art & Design;Pretend Play |
| **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 |
| **3** | Sketch - Draw & Paint | ART_AND_DESIGN | 4.5 | 215644 | 25M | 50,000,000+ | Free | 0 | Teen | Art & Design |
| **4** | Pixel Draw - Number Art Coloring Book | ART_AND_DESIGN | 4.3 | 967 | 2.8M | 100,000+ | Free | 0 | Everyone | Art & Design;Creativity |

In [4]:
```
df.Category.value_counts()
```

Out[4]:
```
Category
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: count, dtype: int64
```

In [5]:
```
df[df.Category == '1.9']
```

Out[5]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price | Content Rating | Genres | Last Updated |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 |

In [6]:
```
LST = ['Life Made WI-Fi Touchscreen Photo Frame', '','1.9', 19.0, '3.0M', '1,000+', 'Fre
```

In [7]:
```
df.loc[df.Category == '1.9'] = LST
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_2888\3959563845.py:1: FutureWarning: Setting an
item of incompatible dtype is deprecated and will raise in a future error of pandas. Val
ue '1.9' has dtype incompatible with float64, please explicitly cast to a compatible dty
pe first.
  df.loc[df.Category == '1.9'] = LST
```

In [8]:
```
df.Category.value_counts()
```

Out[8]:
```
Category
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
Name: count, dtype: int64
```

## Clean the data

In [9]:
```
df[df['App'] == 'Instagram']
```

Out[9]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price | Content Rating | Genres | Last Updated | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **2545** | Instagram | SOCIAL | 4.5 | 66577313 | Varies with device | 1,000,000,000+ | Free | 0 | Teen | Social | July 31, 2018 | |
| **2604** | Instagram | SOCIAL | 4.5 | 66577446 | Varies with device | 1,000,000,000+ | Free | 0 | Teen | Social | July 31, 2018 | |
| **2611** | Instagram | SOCIAL | 4.5 | 66577313 | Varies with device | 1,000,000,000+ | Free | 0 | Teen | Social | July 31, 2018 | |
| **3909** | Instagram | SOCIAL | 4.5 | 66509917 | Varies with device | 1,000,000,000+ | Free | 0 | Teen | Social | July 31, 2018 | |

In [10]:
```
df.duplicated(subset=['App'], keep='first').sum()
```

Out[10]:
```
1181
```

In [11]:
```
df.shape
```

Out[11]:
```
(10841, 13)
```

In [12]:
```
10841 - 1181
```

Out[12]:
```
9660
```

## 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 [13]:
```
df[df['App'] == 'Instagram']
```

Out[13]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price | Content Rating | Genres | Last Updated | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **2545** | Instagram | SOCIAL | 4.5 | 66577313 | Varies with device | 1,000,000,000+ | Free | 0 | Teen | Social | July 31, 2018 | |
| **2604** | Instagram | SOCIAL | 4.5 | 66577446 | Varies with device | 1,000,000,000+ | Free | 0 | Teen | Social | July 31, 2018 | |
| **2611** | Instagram | SOCIAL | 4.5 | 66577313 | Varies with device | 1,000,000,000+ | Free | 0 | Teen | Social | July 31, 2018 | |
| **3909** | Instagram | SOCIAL | 4.5 | 66509917 | Varies with device | 1,000,000,000+ | Free | 0 | Teen | Social | July 31, 2018 | |

In [14]:
```
# Check for duplicate rows based on the 'App' column, marking all duplicates as True
duplicate_apps_df = df[df.duplicated(subset=['App'], keep=False)]
duplicate_apps_df.shape
```

Out[14]:
```
(1979, 13)
```

In [15]:
```
1979-798
```

Out[15]:
```
1181
```

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 numbers show that the data was collected at different times. We can use this to build a criterion for keeping rows. We won't 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 more reliable the ratings.

## Part Two

Let's start by building the dictionary.

In [16]:
```
reviews_max = df.groupby('App')['Reviews'].max()
```

In [17]:
```
reviews_max['Instagram']
```

Out[17]:
```
'66577446'
```

In [18]:
```
android_clean = []

already_added = []

for index, row in df.iterrows():
    name = row['App']
    no_of_reviews = row['Reviews']

    if (no_of_reviews == reviews_max[name]) and (name not in already_added):
```

```
        android_clean.append(row)
        already_added.append(name)
```

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 loop through the `android` data set, and for every iteration:
  - We isolate the name of the app and the number of reviews.
  - We add the current row ( `app` ) to the `android_clean` list, and the app name ( `name` ) to the `already_added` list if:
    - The number of reviews of the current app matches the number of reviews of that app as described in the `reviews_max` dictionary; and
    - The name of the app is not already in the `already_added` list. We need to add this supplementary condition to account for those cases where the highest number of reviews of a duplicate app is the same for more than one entry (for example, the Box app has three entries, and the number of reviews is the same). If we just check for `reviews_max[name] == n_reviews`, we'll still end up with duplicate entries for some apps.

In [19]:
```
android_clean = pd.DataFrame(android_clean)
```

In [20]:
```
android_clean.head()
```

Out[20]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price | Content Rating | Genres |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Photo Editor & Candy Camera & Grid & ScrapBook | ART_AND_DESIGN | 4.1 | 159 | 19M | 10,000+ | Free | 0 | Everyone | Art & Design |
| **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 |
| **3** | Sketch - Draw & Paint | ART_AND_DESIGN | 4.5 | 215644 | 25M | 50,000,000+ | Free | 0 | Teen | Art & Design |
| **4** | Pixel Draw - Number Art Coloring Book | ART_AND_DESIGN | 4.3 | 967 | 2.8M | 100,000+ | Free | 0 | Everyone | Art & Design;Creativity |
| **5** | Paper flowers instructions | ART_AND_DESIGN | 4.4 | 167 | 5.6M | 50,000+ | Free | 0 | Everyone | Art & Design |

In [21]:
```
android_clean[android_clean.App == 'Instagram']
```

Out[21]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price | Content Rating | Genres | Last Updated | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **2604** | Instagram | SOCIAL | 4.5 | 66577446 | Varies with device | 1,000,000,000+ | Free | 0 | Teen | Social | July 31, 2018 | |

# 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 [22]:
```
def is_english(app_name):
    list1 = []
    for i in app_name:
        if ord(i) > 127:
            list1.append(False)
        else:
            list1.append(True)

    check = set(list1)

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

In [23]:
```
is_english("Instagram□")
```

Out[23]:
```
False
```

In [24]:
```
is_english("□□□□□□□□")
```

Out[24]:
```
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 [25]:
```
def is_english(app_name):
    list1 = []
    for i in app_name:
        if ord(i) > 127:
            list1.append(False)
        else:
            list1.append(True)

    ascii_count = 0
    for i in list1:
        if i == False:
            ascii_count += 1


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

In [26]:
```
is_english("Instagram□□□□")
```

Out[26]:
```
False
```

In [27]:
```
android_english= android_clean[android_clean['App'].apply(is_english)]
```

In [28]:
```
android_english.head()
```

Out[28]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price | Content Rating | Genres |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Photo Editor & Candy Camera & Grid & ScrapBook | ART_AND_DESIGN | 4.1 | 159 | 19M | 10,000+ | Free | 0 | Everyone | Art & Design |
| **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 |
| **3** | Sketch - Draw & Paint | ART_AND_DESIGN | 4.5 | 215644 | 25M | 50,000,000+ | Free | 0 | Teen | Art & Design |
| **4** | Pixel Draw - Number Art Coloring Book | ART_AND_DESIGN | 4.3 | 967 | 2.8M | 100,000+ | Free | 0 | Everyone | Art & Design;Creativity |
| **5** | Paper flowers instructions | ART_AND_DESIGN | 4.4 | 167 | 5.6M | 50,000+ | Free | 0 | Everyone | Art & Design |

# 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 [29]:
```
android_english.Price.value_counts()
```

Out[29]:
```
Price
0          8863
$0.99       145
$2.99       124
$1.99        73
$4.99        70
          ...
$18.99        1
$389.99       1
$19.90        1
$1.75         1
$1.04         1
Name: count, Length: 92, dtype: int64
```

In [30]:
```
android_clean = android_english[android_english.Price == '0']
```

In [31]:
```
android_clean.head()
```

Out[31]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price | Content Rating | Genres |
|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Photo Editor & Candy Camera & Grid & ScrapBook | ART_AND_DESIGN | 4.1 | 159 | 19M | 10,000+ | Free | 0 | Everyone | Art & Design |
| **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 |
| **3** | Sketch - Draw & Paint | ART_AND_DESIGN | 4.5 | 215644 | 25M | 50,000,000+ | Free | 0 | Teen | Art & Design |
| **4** | Pixel Draw - Number Art Coloring Book | ART_AND_DESIGN | 4.3 | 967 | 2.8M | 100,000+ | Free | 0 | Everyone | Art & Design;Creativity |
| **5** | Paper flowers instructions | ART_AND_DESIGN | 4.4 | 167 | 5.6M | 50,000+ | Free | 0 | Everyone | Art & Design |

In [32]:
```
android_final = android_english[android_english.Price == '0']
```

In [33]:
```
android_final.head()
```

Out[33]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price | Content Rating | Genres |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Photo Editor & Candy Camera & Grid & ScrapBook | ART_AND_DESIGN | 4.1 | 159 | 19M | 10,000+ | Free | 0 | Everyone | Art & Design |
| **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 |
| **3** | Sketch - Draw & Paint | ART_AND_DESIGN | 4.5 | 215644 | 25M | 50,000,000+ | Free | 0 | Teen | Art & Design |
| **4** | Pixel Draw - Number Art Coloring Book | ART_AND_DESIGN | 4.3 | 967 | 2.8M | 100,000+ | Free | 0 | Everyone | Art & Design;Creativity |
| **5** | Paper flowers instructions | ART_AND_DESIGN | 4.4 | 167 | 5.6M | 50,000+ | Free | 0 | Everyone | Art & Design |

# Most Common Apps by Genre

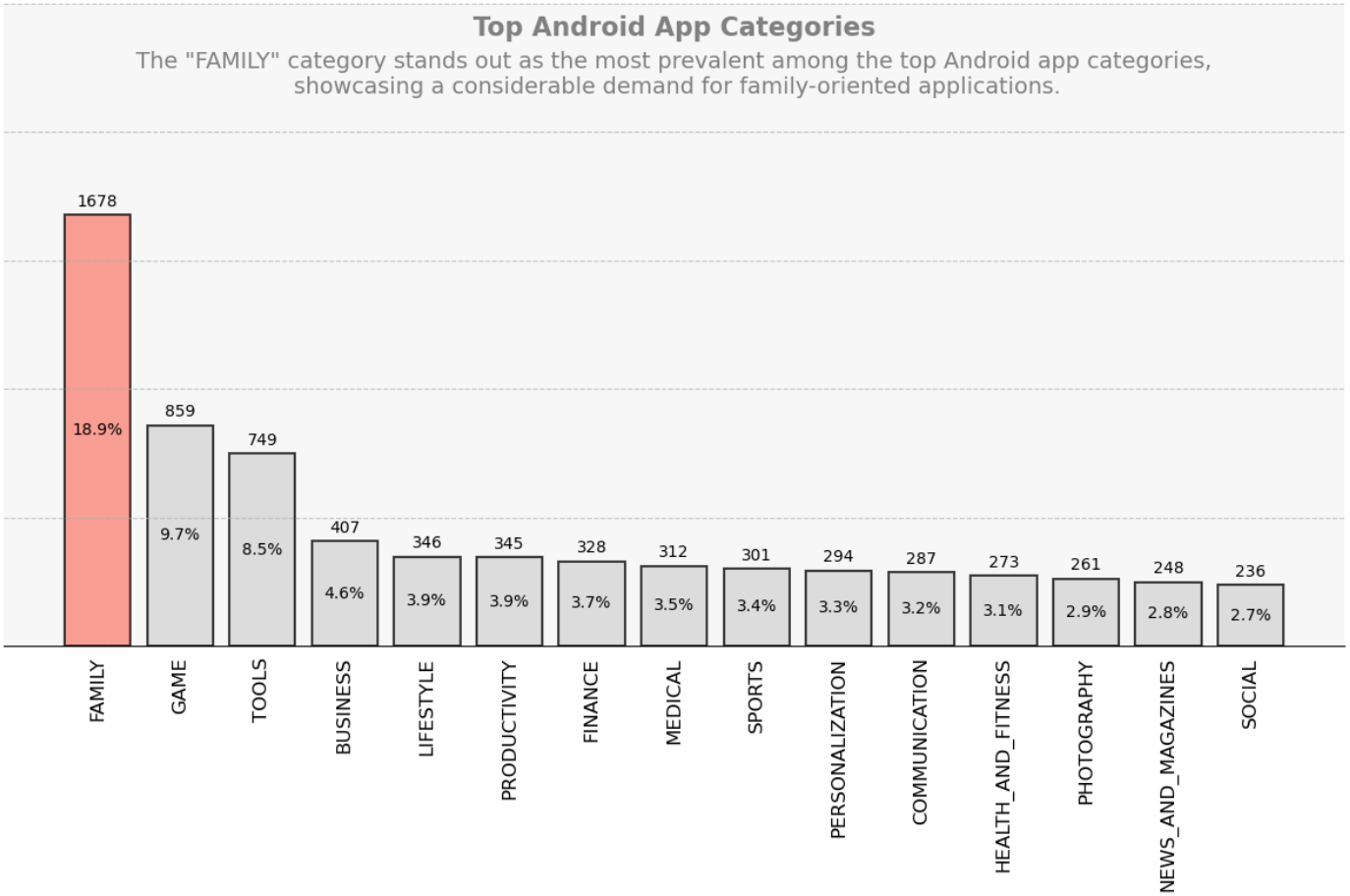
In [34]:
```
android_final['Category'].value_counts(normalize=True) * 100
```

```
Category
```

Out[34]:
```
FAMILY                 18.932641
GAME                    9.691978
TOOLS                   8.450863
BUSINESS                4.592125
LIFESTYLE               3.903870
PRODUCTIVITY            3.892587
FINANCE                 3.700779
MEDICAL                 3.520253
SPORTS                  3.396141
PERSONALIZATION         3.317161
COMMUNICATION           3.238181
HEALTH_AND_FITNESS      3.080221
PHOTOGRAPHY             2.944827
NEWS_AND_MAGAZINES      2.798150
SOCIAL                  2.662755
TRAVEL_AND_LOCAL        2.335552
SHOPPING                2.245289
BOOKS_AND_REFERENCE     2.143744
DATING                  1.861672
VIDEO_PLAYERS           1.793975
MAPS_AND_NAVIGATION     1.399075
FOOD_AND_DRINK          1.241115
EDUCATION               1.173418
ENTERTAINMENT           0.959043
LIBRARIES_AND_DEMO      0.936477
AUTO_AND_VEHICLES       0.925195
HOUSE_AND_HOME          0.823649
WEATHER                 0.801083
EVENTS                  0.710820
PARENTING               0.654406
ART_AND_DESIGN          0.643123
COMICS                  0.620557
BEAUTY                  0.597992
                        0.011283
Name: proportion, dtype: float64
```

In [35]:
```
# 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)*100,1)[:15]

# Create stylish bar chart
plt.figure(figsize=(12, 8))
bars = plt.bar(categories, counts, color='lightgray', alpha=0.75, edgecolor='black', lin
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, 3000, 500), [],fontsize=12)  # Customized tick labels and customised
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 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 + 20, '%d' % int(height), ha='cente
```

```
    plt.text(bar.get_x() + bar.get_width()/2, height/2, f'{perc}%', ha='center', va='cen


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

# Adding chart title inside the chart
plt.text(0.5, 0.95, 'Top Android App Categories', horizontalalignment='center', fontsize
        color='gray', fontweight='bold')

# Adding conclusion inside the chart
plt.text(0.5, 0.86, 'The "FAMILY" category stands out as the most prevalent among the to
         horizontalalignment='center', fontsize=14, transform=plt.gca().transAxes,  colo

# 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 [36]:
```
android_final.head()
```

Out[36]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price | Content Rating | Genres |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Photo Editor & Candy Camera & Grid & ScrapBook | ART_AND_DESIGN | 4.1 | 159 | 19M | 10,000+ | Free | 0 | Everyone | Art & Design |
| **2** | U Launcher | ART_AND_DESIGN | 4.7 | 87510 | 8.7M | 5,000,000+ | Free | 0 | Everyone | Art & Design |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Lite – FREE Live Cool Themes, Hide ... | | | | | | | | | |
| **3** | Sketch - Draw & Paint | ART_AND_DESIGN | 4.5 | 215644 | 25M | 50,000,000+ | Free | 0 | Teen | Art & Design |
| **4** | Pixel Draw - Number Art Coloring Book | ART_AND_DESIGN | 4.3 | 967 | 2.8M | 100,000+ | Free | 0 | Everyone | Art & Design;Creativity |
| **5** | Paper flowers instructions | ART_AND_DESIGN | 4.4 | 167 | 5.6M | 50,000+ | Free | 0 | Everyone | Art & Design |

# 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 clearer picture about genre popularity. However, the install numbers don't seem precise enough — we can see that most values are open-ended (100+, 1,000+, 5,000+, etc.):

In [37]:
```
android_final.Installs.value_counts()
```

Out[37]:
```
Installs
1,000,000+        1395
100,000+          1024
10,000,000+        932
10,000+            904
1,000+             745
100+               613
5,000,000+         606
500,000+           494
50,000+            423
5,000+             400
10+                314
500+               288
50,000,000+        203
100,000,000+       188
50+                170
5+                  70
1+                  45
500,000,000+        24
1,000,000,000+      20
0+                   4
0                    1
Name: count, dtype: int64
```

In [38]:
```
android_final.loc[:, "Installs_int"] = android_final["Installs"].str.replace("+", "").st
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_2888\4184877451.py:1: 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
  android_final.loc[:, "Installs_int"] = android_final["Installs"].str.replace("+", "").
str.replace(",", '').astype(int)
```

In [39]:
```
install_frq = android_final["Installs_int"].value_counts().sort_index(ascending = False)
```

In [40]:
```
install_frq = install_frq[install_frq.index > 500]
```

In [41]:
```
install_frq_per = round(android_final.Installs_int.value_counts(normalize=True) * 100, 2
```

In [42]:
```
install_frq_per
```

Out[42]:
```
Installs_int
1000000000     0.23
500000000      0.27
100000000      2.12
50000000       2.29
10000000      10.52
5000000        6.84
1000000       15.74
500000         5.57
100000        11.55
50000          4.77
10000         10.20
5000           4.51
1000           8.41
500            3.25
100            6.92
50             1.92
10             3.54
5              0.79
1              0.51
0              0.06
Name: proportion, dtype: float64
```

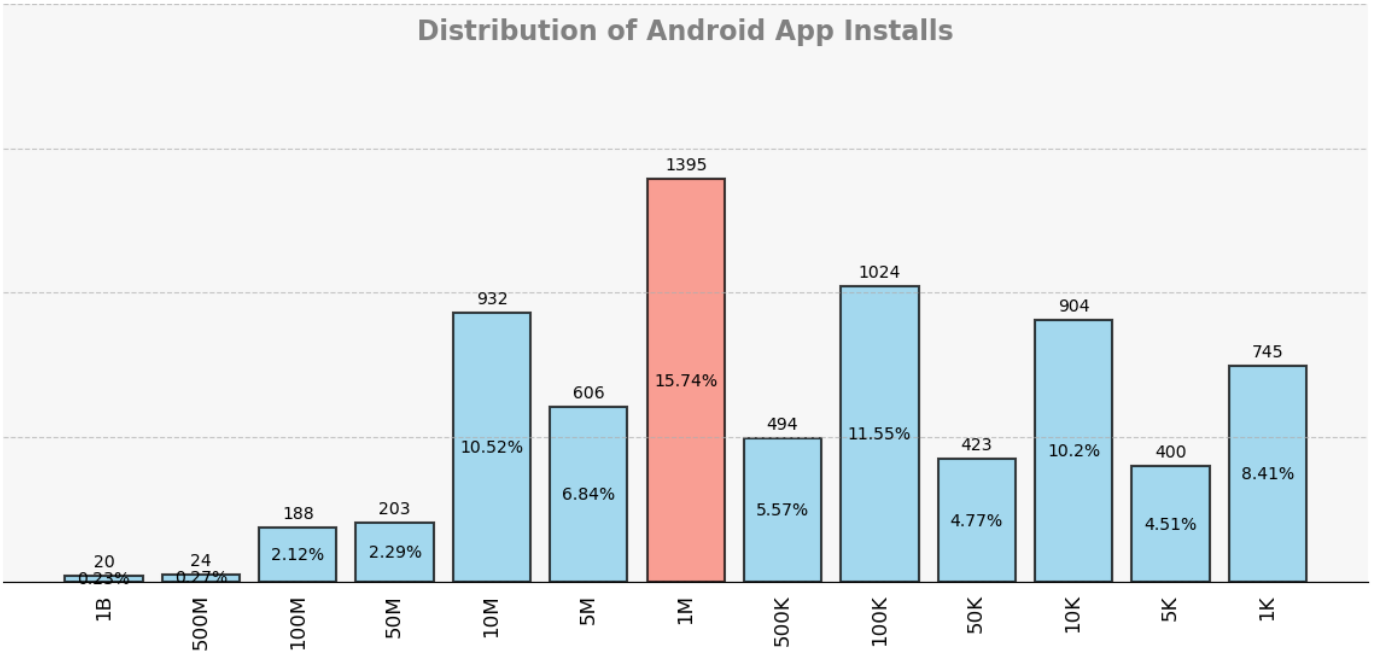
In [43]:
```
# alphanumeric units
def alphanumeric_units(value):
    if value >= 1e9:
        return f'{value / 1e9:.0f}B'
    elif value >= 1e6:
        return f'{value / 1e6:.0f}M'
    elif value >= 1e3:
        return f'{value / 1e3:.0f}K'
    else:
        return f'{value:.0f}'
```

In [44]:
```
install_frq.index = install_frq.index.map(alphanumeric_units)
install_frq
```

Out[44]:
```
Installs_int
1B        20
500M      24
100M     188
50M      203
10M      932
5M       606
1M      1395
500K     494
100K    1024
50K      423
10K      904
5K       400
1K       745
Name: count, dtype: int64
```

In [45]:
```
# 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', linew
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 tick labels and customised
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 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 + 20, '%d' % int(height), ha='cente
    plt.text(bar.get_x() + bar.get_width()/2, height/2, f'{perc}%', ha='center', va='cen


# 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', fontweight='bold')

# Adding conclusion inside the chart
plt.text(0.5, -0.35, '''From the data provided, it's evident that the majority of Androi
         horizontalalignment='center', fontsize=11, transform=plt.gca().transAxes,  colo

# 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()
```




From the data provided, it's evident that the majority of Android app installs fall within the lower range,
with the highest number of installs being in the 1K to 10M range.
Specifically, the 1M install range stands out with 1395 apps, indicating a significant number of apps falling into this category.
As the number of installs increases, the count of apps decreases,
with only a few apps reaching install counts of 500M and 1B.

In [46]:
```
categories_android = android_final["Category"].unique()
categories_android
```

Out[46]:
```
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 [47]:
```
pd.pivot_table(android_final, values = 'Installs_int', index = 'Category', aggfunc = 'me
```

Out[47]:

| Category | Installs_int |
|---|---|
| | 1.000000e+03 |
| 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 [48]:
```
#Display DataFrame without scientific notation
pd.options.display.float_format = '{:.0f}'.format
```

In [49]:
```
categories_installs  = pd.pivot_table(android_final, values = 'Installs_int', index = 'C
categories_installs = categories_installs.sort_values(by="Installs_int", ascending=False
categories_installs = categories_installs["Installs_int"]
categories_installs
```

Out[49]:
```
Category
COMMUNICATION         38456119
VIDEO_PLAYERS         24727872
SOCIAL                23253652
PHOTOGRAPHY           17805628
PRODUCTIVITY          16787331
GAME                  15560966
TRAVEL_AND_LOCAL      13984078
ENTERTAINMENT         11640706
TOOLS                 10682301
NEWS_AND_MAGAZINES     9549178
BOOKS_AND_REFERENCE    8767812
SHOPPING               7036877
PERSONALIZATION        5201483
WEATHER                5074486
HEALTH_AND_FITNESS     4188822
MAPS_AND_NAVIGATION    4056942
FAMILY                 3694276
SPORTS                 3638640
ART_AND_DESIGN         1986335
FOOD_AND_DRINK         1924898
EDUCATION              1820673
```

```
BUSINESS               1712290
LIFESTYLE              1437816
FINANCE                1387692
HOUSE_AND_HOME         1331541
DATING                  854029
COMICS                  817657
AUTO_AND_VEHICLES       647318
LIBRARIES_AND_DEMO      638504
PARENTING               542604
BEAUTY                  513152
EVENTS                  253542
MEDICAL                 120616
                          1000
Name: Installs_int, dtype: float64
```

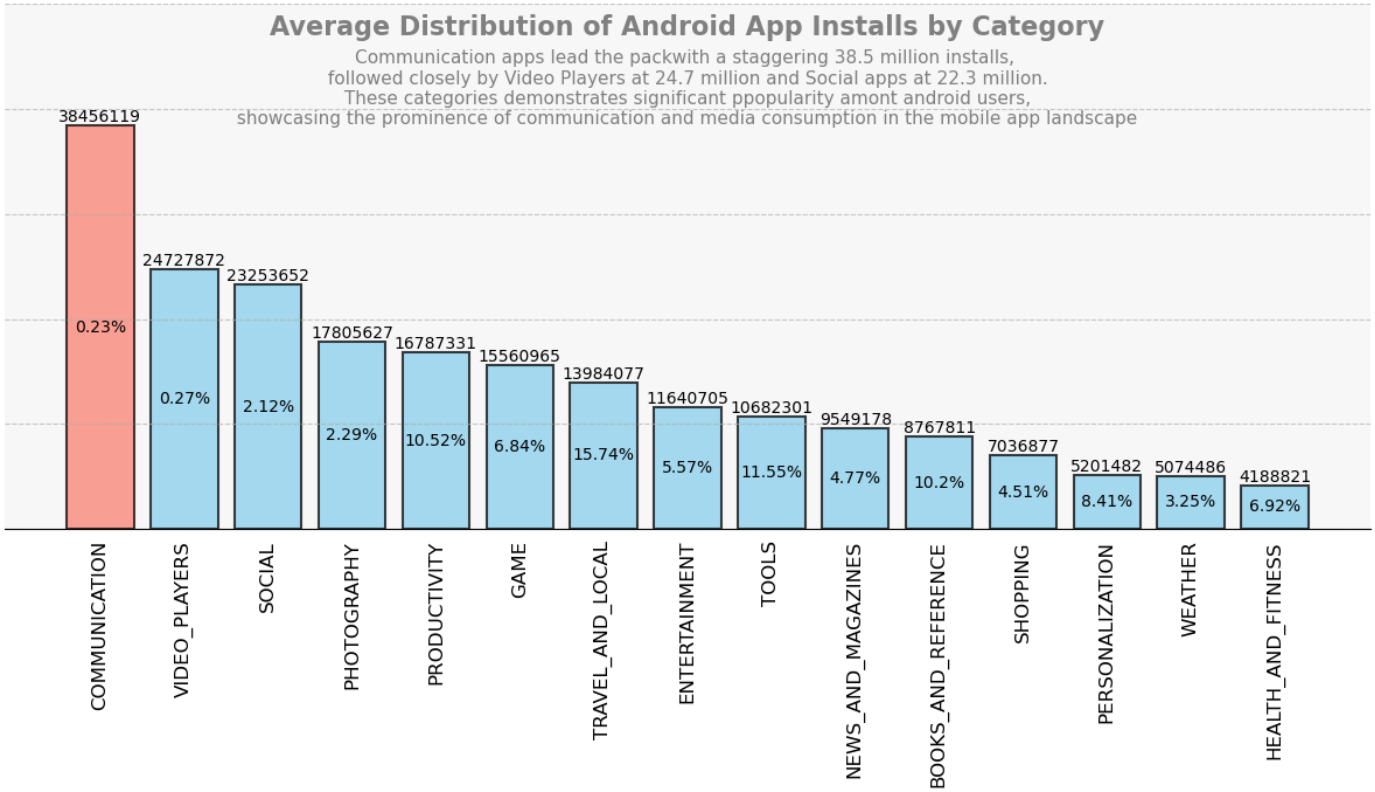
In [50]:
```python
# alphanumeric units
def alphanumeric_units(value):
    if value >= 1e9:
        return f'{value / 1e9:.0f}B'
    elif value >= 1e6:
        return f'{value / 1e6:.0f}M'
    elif value >= 1e3:
        return f'{value / 1e3:.0f}K'
    else:
        return f'{value:.0f}'
```

In [52]:
```python
categories_installs_unit = categories_installs.map(alphanumeric_units)
categories_installs_unit
```

Out[52]:
```
Category
COMMUNICATION           38M
VIDEO_PLAYERS           25M
SOCIAL                  23M
PHOTOGRAPHY             18M
PRODUCTIVITY            17M
GAME                    16M
TRAVEL_AND_LOCAL        14M
ENTERTAINMENT           12M
TOOLS                   11M
NEWS_AND_MAGAZINES      10M
BOOKS_AND_REFERENCE      9M
SHOPPING                 7M
PERSONALIZATION          5M
WEATHER                  5M
HEALTH_AND_FITNESS       4M
MAPS_AND_NAVIGATION      4M
FAMILY                   4M
SPORTS                   4M
ART_AND_DESIGN           2M
FOOD_AND_DRINK           2M
EDUCATION                2M
BUSINESS                 2M
LIFESTYLE                1M
FINANCE                  1M
HOUSE_AND_HOME           1M
DATING                 854K
COMICS                 818K
AUTO_AND_VEHICLES      647K
LIBRARIES_AND_DEMO     639K
PARENTING              543K
BEAUTY                 513K
EVENTS                 254K
MEDICAL                121K
                         1K
Name: Installs_int, dtype: object
```

In [60]:

```
# 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', linew
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, 10000000), [],fontsize=12)  # Customized tick labels and c
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 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 + 20, '%d' % int(height), ha='cente
    plt.text(bar.get_x() + bar.get_width()/2, height/2, f'{perc}%', ha='center', va='cen


# 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', horizont
        color='gray', fontweight='bold')

# Adding conclusion inside the chart
plt.text(0.5, 0.77, '''Communication apps lead the packwith a staggering 38.5 million in
         horizontalalignment='center', fontsize=11, transform=plt.gca().transAxes,  colo

# 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 [61]:
```
category_group = android_final.groupby("Category")
```

In [65]:
```
COMMUNICATION = category_group.get_group("COMMUNICATION").sort_values(by="Installs_int")
COMMUNICATION.head()
```

Out[65]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price | Content Rating | Genres | Upd |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **5921** | BA SALES | COMMUNICATION | NaN | 0 | 2.9M | 1+ | Free | 0 | Everyone | Communication | Jar 31, |
| **6399** | Of the wall Arapaho bk | COMMUNICATION | NaN | 0 | 12M | 5+ | Free | 0 | Everyone | Communication | Au 18, |
| **10169** | Test Server SMS FA | COMMUNICATION | NaN | 0 | 1.8M | 5+ | Free | 0 | Everyone | Communication | Jul |
| **10672** | FNH Payment Info | COMMUNICATION | NaN | 0 | 2.1M | 10+ | Free | 0 | Everyone | Communication | M 17, |
| **9514** | Ek IRA | COMMUNICATION | NaN | 0 | 5.7M | 10+ | Free | 0 | Everyone | Communication | Jul |

In [66]:
```
# 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 [68]:
```
categories_installs.index[:15]
```

Out[68]:
```
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 [70]:
```
df = COMMUNICATION[["App","Installs_int"]].head(15)
df["Installs_int_unit"] = df["Installs_int"].map(alphanumeric_units)
df
```

Out[70]:

| | App | Installs_int | Installs_int_unit |
|---|---|---|---|
| **5921** | BA SALES | 1 | 1 |
| **6399** | Of the wall Arapaho bk | 5 | 5 |
| **10169** | Test Server SMS FA | 5 | 5 |
| **10672** | FNH Payment Info | 10 | 10 |
| **9514** | Ek IRA | 10 | 10 |
| **9455** | EJ messenger | 10 | 10 |
| **9412** | ei | 10 | 10 |
| **8556** | DM - The Offical Messaging App | 10 | 10 |
| **8497** | DK Browser | 10 | 10 |
| **5100** | Oklahoma Ag Co-op Council | 10 | 10 |
| **10670** | FN Web Radio | 10 | 10 |
| **8147** | Hlášenírozhlasu.cz | 10 | 10 |
| **9648** | EO Mumbai | 10 | 10 |
| **7485** | CK Call NEW | 10 | 10 |
| **10748** | FP Live | 10 | 10 |

In [72]:
```
df = category_group.get_group("VIDEO_PLAYERS").sort_values(by="Installs_int", ascending=
df = df[["App","Installs_int"]].head(15)
df["Installs_int_unit"] = df["Installs_int"].map(alphanumeric_units)
df
```

Out[72]:

| | App | Installs_int | Installs_int_unit |
|---|---|---|---|
| **3665** | YouTube | 1000000000 | 1B |
| **3687** | Google Play Movies & TV | 1000000000 | 1B |
| **3711** | MX Player | 500000000 | 500M |
| **3675** | VLC for Android | 100000000 | 100M |
| **4688** | VivaVideo - Video Editor & Photo Movie | 100000000 | 100M |
| **4032** | Dubsmash | 100000000 | 100M |
| **10647** | Motorola FM Radio | 100000000 | 100M |
| **4696** | VideoShow-Video Editor, Video Maker, Beauty Ca... | 100000000 | 100M |
| **3672** | Motorola Gallery | 100000000 | 100M |
| **3691** | Samsung Video Library | 50000000 | 50M |
| **4038** | DU Recorder – Screen Recorder, Video Editor, Live | 50000000 | 50M |
| **3693** | LIKE – Magic Video Maker & Community | 50000000 | 50M |
| **3686** | Vigo Video | 50000000 | 50M |

| | | | |
|---|---|---|---|
| **4049** | KineMaster – Pro Video Editor | 50000000 | 50M |
| **5612** | Ringdroid | 50000000 | 50M |

In [73]:
```
df = category_group.get_group("VIDEO_PLAYERS").sort_values(by="Installs_int", ascending=
df = df[["App","Installs_int"]].head(15)
df["Installs_int_unit"] = df["Installs_int"].map(alphanumeric_units)
df
```

Out[73]:

| | App | Installs_int | Installs_int_unit |
|---|---|---|---|
| **3665** | YouTube | 1000000000 | 1B |
| **3687** | Google Play Movies & TV | 1000000000 | 1B |
| **3711** | MX Player | 500000000 | 500M |
| **3675** | VLC for Android | 100000000 | 100M |
| **4688** | VivaVideo - Video Editor & Photo Movie | 100000000 | 100M |
| **4032** | Dubsmash | 100000000 | 100M |
| **10647** | Motorola FM Radio | 100000000 | 100M |
| **4696** | VideoShow-Video Editor, Video Maker, Beauty Ca... | 100000000 | 100M |
| **3672** | Motorola Gallery | 100000000 | 100M |
| **3691** | Samsung Video Library | 50000000 | 50M |
| **4038** | DU Recorder – Screen Recorder, Video Editor, Live | 50000000 | 50M |
| **3693** | LIKE – Magic Video Maker & Community | 50000000 | 50M |
| **3686** | Vigo Video | 50000000 | 50M |
| **4049** | KineMaster – Pro Video Editor | 50000000 | 50M |
| **5612** | Ringdroid | 50000000 | 50M |

In [74]:
```
df = category_group.get_group("PHOTOGRAPHY").sort_values(by="Installs_int", ascending=Fa
df = df[["App","Installs_int"]].head(15)
df["Installs_int_unit"] = df["Installs_int"].map(alphanumeric_units)
df
```

Out[74]:

| | 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 [75]:
```
df = category_group.get_group("PRODUCTIVITY").sort_values(by="Installs_int", ascending=F
df = df[["App","Installs_int"]].head(15)
df["Installs_int_unit"] = df["Installs_int"].map(alphanumeric_units)
df
```

Out[75]:

| | 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 |

# Analysis of Photography Category and Potential for Photo Generation App in 2024

# Conclusion

The analysis of the photography category reveals a notabe trend in the popularity of photo editing and collage-making applications, with several apps gamering over 100 million installs. This indicates a strong demand for photo-related functionalities among users.

Given this observation, there appears to be significant potential for the developmentof a photo generation in 2024. Such an app, offering prompt and free generation of pictures and photos, could capitalize on the existing user interest in photography apps. By providing innovative features, easy usability, and high-quality output, this application could stand out in the competitive market and attract a large user base.

Considering the success of existing photography apps and the evolving preferences of users, investing in the development of a photo generation app seems promising for tapping into this lucrative market segment in 2024.