`Analyzing Popular App Categories on Google Play Store`

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 earn money through ads. By understanding which app categories are in high demand. We can help our developers create apps that attracts more users and generate more revenue. We'll analyze data from the Google play store to identity patterns and preferences among users. This way, we can make smater decision about the kinds of apps we develope.

In [1]:
```
#Importing required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
#pd.set_option('display.max_columns', None) # this is to display all the columns in the dataframe
#pd.set_option('display.max_rows', None) # this is to display all the rows in the dataframe
```

In [3]:
```
# hide all warnings runtime
import warnings
warnings.filterwarnings('ignore')
```

Data Loading, Exploration and Cleaning

In [4]:
```
#Reading the dataset into pandas DataFrame
df = pd.read_csv('googleplaystore.csv')
df.head()
```

Out[4]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
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	4.0.3 and up
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	4.0.3 and up
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	4.0.3 and up
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 and up
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 and up

In [5]:
```
print(f"No. of Columns {df.shape[1]}")
print(f"No. of Rows {df.shape[0]}")
```

```
No. of Columns 13
No. of Rows 10841
```

In [6]:
```
print(f"The name of the columns are as follows: \
      {df.columns}")
```

```
The name of the columns are as follows:       Index(['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs'
, 'Type',
       'Price', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver',
       'Android Ver'],
      dtype='object')
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10841 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             10841 non-null  object 
 1   Category        10841 non-null  object 
 2   Rating          9367 non-null   float64
 3   Reviews         10841 non-null  object 
 4   Size            10841 non-null  object 
 5   Installs        10841 non-null  object 
 6   Type            10840 non-null  object 
 7   Price           10841 non-null  object 
 8   Content Rating  10840 non-null  object 
 9   Genres          10841 non-null  object 
 10  Last Updated    10841 non-null  object 
 11  Current Ver     10833 non-null  object 
 12  Android Ver     10838 non-null  object 
dtypes: float64(1), object(12)
memory usage: 1.1+ MB
```

Observations

---

1. There are 10841 rows and 13 columns in the dataset
2. The columns are of different data types
3. There are some missing values in the dataset.
4. There are some columns which are of object data type but they should be of numeric data type, we will convert them
    - `'Size', 'Installs', 'Price'`

Target Column: `Size`

In [8]:
```
df['Size'].unique()
```

Out[8]:
```
array(['19M', '14M', '8.7M', '25M', '2.8M', '5.6M', '29M', '33M', '3.1M',
       '28M', '12M', '20M', '21M', '37M', '2.7M', '5.5M', '17M', '39M',
       '31M', '4.2M', '7.0M', '23M', '6.0M', '6.1M', '4.6M', '9.2M',
       '5.2M', '11M', '24M', 'Varies with device', '9.4M', '15M', '10M',
       '1.2M', '26M', '8.0M', '7.9M', '56M', '57M', '35M', '54M', '201k',
       '3.6M', '5.7M', '8.6M', '2.4M', '27M', '2.5M', '16M', '3.4M',
       '8.9M', '3.9M', '2.9M', '38M', '32M', '5.4M', '18M', '1.1M',
       '2.2M', '4.5M', '9.8M', '52M', '9.0M', '6.7M', '30M', '2.6M',
       '7.1M', '3.7M', '22M', '7.4M', '6.4M', '3.2M', '8.2M', '9.9M',
       '4.9M', '9.5M', '5.0M', '5.9M', '13M', '73M', '6.8M', '3.5M',
       '4.0M', '2.3M', '7.2M', '2.1M', '42M', '7.3M', '9.1M', '55M',
       '23k', '6.5M', '1.5M', '7.5M', '51M', '41M', '48M', '8.5M', '46M',
       '8.3M', '4.3M', '4.7M', '3.3M', '40M', '7.8M', '8.8M', '6.6M',
       '5.1M', '61M', '66M', '79k', '8.4M', '118k', '44M', '695k', '1.6M',
       '6.2M', '18k', '53M', '1.4M', '3.0M', '5.8M', '3.8M', '9.6M',
       '45M', '63M', '49M', '77M', '4.4M', '4.8M', '70M', '6.9M', '9.3M',
       '10.0M', '8.1M', '36M', '84M', '97M', '2.0M', '1.9M', '1.8M',
       '5.3M', '47M', '556k', '526k', '76M', '7.6M', '59M', '9.7M', '78M',
       '72M', '43M', '7.7M', '6.3M', '334k', '34M', '93M', '65M', '79M',
       '100M', '58M', '50M', '68M', '64M', '67M', '60M', '94M', '232k',
       '99M', '624k', '95M', '8.5k', '41k', '292k', '11k', '80M', '1.7M',
       '74M', '62M', '69M', '75M', '98M', '85M', '82M', '96M', '87M',
       '71M', '86M', '91M', '81M', '92M', '83M', '88M', '704k', '862k',
       '899k', '378k', '266k', '375k', '1.3M', '975k', '980k', '4.1M',
       '89M', '696k', '544k', '525k', '920k', '779k', '853k', '720k',
       '713k', '772k', '318k', '58k', '241k', '196k', '857k', '51k',
       '953k', '865k', '251k', '930k', '540k', '313k', '746k', '203k',
       '26k', '314k', '239k', '371k', '220k', '730k', '756k', '91k',
       '293k', '17k', '74k', '14k', '317k', '78k', '924k', '902k', '818k',
       '81k', '939k', '169k', '45k', '475k', '965k', '90M', '545k', '61k',
       '283k', '655k', '714k', '93k', '872k', '121k', '322k', '1.0M',
       '976k', '172k', '238k', '549k', '206k', '954k', '444k', '717k',
       '210k', '609k', '308k', '705k', '306k', '904k', '473k', '175k',
       '350k', '383k', '454k', '421k', '70k', '812k', '442k', '842k',
       '417k', '412k', '459k', '478k', '335k', '782k', '721k', '430k',
       '429k', '192k', '200k', '460k', '728k', '496k', '816k', '414k',
       '506k', '887k', '613k', '243k', '569k', '778k', '683k', '592k',
       '319k', '186k', '840k', '647k', '191k', '373k', '437k', '598k',
       '716k', '585k', '982k', '222k', '219k', '55k', '948k', '323k',
       '691k', '511k', '951k', '963k', '25k', '554k', '351k', '27k',
       '82k', '208k', '913k', '514k', '551k', '29k', '103k', '898k',
       '743k', '116k', '153k', '209k', '353k', '499k', '173k', '597k',
       '809k', '122k', '411k', '400k', '801k', '787k', '237k', '50k',
       '643k', '986k', '97k', '516k', '837k', '780k', '961k', '269k',
       '20k', '498k', '600k', '749k', '642k', '881k', '72k', '656k',
       '601k', '221k', '228k', '108k', '940k', '176k', '33k', '663k',
       '34k', '942k', '259k', '164k', '458k', '245k', '629k', '28k',
       '288k', '775k', '785k', '636k', '916k', '994k', '309k', '485k',
       '914k', '903k', '608k', '500k', '54k', '562k', '847k', '957k',
       '688k', '811k', '270k', '48k', '329k', '523k', '921k', '874k',
       '981k', '784k', '280k', '24k', '518k', '754k', '892k', '154k',
       '860k', '364k', '387k', '626k', '161k', '879k', '39k', '970k',
       '170k', '141k', '160k', '144k', '143k', '190k', '376k', '193k',
       '246k', '73k', '658k', '992k', '253k', '420k', '404k', '1,000+',
       '470k', '226k', '240k', '89k', '234k', '257k', '861k', '467k',
       '157k', '44k', '676k', '67k', '552k', '885k', '1020k', '582k',
       '619k'], dtype=object)
```

- There are several uniques values in the `Size` column, we have to first make the unit into one common unit from M and K to bytes, and then remove the `M` and `K` from the values and convert them into numeric data type.

In [9]:
```
# find the values in size column which has 'M' in it
df['Size'].loc[df['Size'].str.contains('M')].value_counts().sum()
```

Out[9]:
```
8829
```

In [10]:
```
# find the values in size column which has 'k' in it
df['Size'].loc[df['Size'].str.contains('k')].value_counts().sum()
```

Out[10]:
```
316
```

In [11]:
```
# find the values in size column which has 'Varies with device' in it
df['Size'].loc[df['Size'].str.contains('Varies with device')].value_counts().sum()
```

Out[11]:
```
1695
```

In [12]:
```
# Total Values in Size column
df['Size'].value_counts().sum()
```

Out[12]:
```
10841
```

In [13]:
```
# taking sum of all the values in size column which has 'M', 'K' and 'varies with device' in it
8830+316+1695
```

Out[13]: 10841

- We have `8830` values in `M` units
- We have `316` values in `k` units
- We have `1695` value in `Varies with device`

> Let's convert the `M` and `K` units into bytes and then remove the `M` and `K` from the values and convert them into numeric data type.

In [14]:
```
# convert the size column to numeric by multiplying the values with 1024 if it has 'k' in it and 1024*1024 if i
# this function will convert the size column to numeric
def convert_size(size):
    if isinstance(size, str):
        if 'k' in size:
            return float(size.replace('k', '')) * 1024
        elif 'M' in size:
            return float(size.replace('M', '')) * 1024 * 1024
        elif 'Varies with device' in size:
            return np.nan
    return size

df['Size'] = df['Size'].apply(convert_size)
```

In [15]:
```
# rename the column name 'Size' to 'Size_in_bytes'
df.rename(columns={'Size': 'Size_in_bytes'}, inplace=True)
```

Target Column: `Category`

In [17]:
```
df['Category'].value_counts()
```

Out[17]:
```
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 [18]:
```
df[df['Category'] == '1.9']
```

Out[18]:

	App	Category	Rating	Reviews	Size_in_bytes	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

In [19]:
```
df[df['Category'] == '1.9'].values
```

Out[19]:
```
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 [20]:
```
clean_list = ['Life Made WI-Fi Touchscreen Photo Frame', 'LIFESTYLE', '1.9', 19.0, '3.0M',
        '1,000+', 'Free', '0', 'Everyone', 'life style', 'February 11, 2018',
```

```
       '1.0.19', '4.0 and up']
clean_list
```

Out[20]:
```
['Life Made WI-Fi Touchscreen Photo Frame',
 'LIFESTYLE',
 '1.9',
 19.0,
 '3.0M',
 '1,000+',
 'Free',
 '0',
 'Everyone',
 'life style',
 'February 11, 2018',
 '1.0.19',
 '4.0 and up']
```

In [21]:
```
df[df['Category'] == '1.9'] = clean_list
```

In [22]:
```
df['Category'].value_counts()
```

Out[22]:
```
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
```

Observations

---

we can see above there is no messy value remaining.

Target Column : App

In [23]:
```
df['App']
```

Out[23]:
```
0           Photo Editor & Candy Camera & Grid & ScrapBook
1                                       Coloring book moana
2        U Launcher Lite – FREE Live Cool Themes, Hide ...
3                                     Sketch - Draw & Paint
4                     Pixel Draw - Number Art Coloring Book
                               ...
10836                                   Sya9a Maroc - FR
10837                     Fr. Mike Schmitz Audio Teachings
10838                            Parkinson Exercices FR
10839                        The SCP Foundation DB fr nn5n
10840       iHoroscope - 2018 Daily Horoscope & Astrology
Name: App, Length: 10841, dtype: object
```

In [24]:
```
app_count = df['App'].value_counts()
app_count
```

Out[24]:
```
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

if we explore the Google Play data enough, we'll find that some of the apps have more than one entry. For instance, the app 'Instagram' has 4 entries.

In [25]:
```
'Instagram' in app_count[app_count > 1].index
```

Out[25]:
```
True
```

In [26]:
```
duplicated_app = df[df.duplicated(subset=['App'], keep= False)]

duplicated_app[duplicated_app['App'] == 'Instagram']
```

Out[26]:

	App	Category	Rating	Reviews	Size_in_bytes	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
2545	Instagram	SOCIAL	4.5	66577313	NaN	1,000,000,000+	Free	0	Teen	Social	July 31, 2018	Varies with device	Varies with device
2604	Instagram	SOCIAL	4.5	66577446	NaN	1,000,000,000+	Free	0	Teen	Social	July 31, 2018	Varies with device	Varies with device
2611	Instagram	SOCIAL	4.5	66577313	NaN	1,000,000,000+	Free	0	Teen	Social	July 31, 2018	Varies with device	Varies with device
3909	Instagram	SOCIAL	4.5	66509917	NaN	1,000,000,000+	Free	0	Teen	Social	July 31, 2018	Varies with device	Varies with device

In [27]:
```
#Number of duplicate apps
duplicated_app_num = duplicated_app['App'].nunique()
duplicated_app_num
```

Out[27]:
```
798
```

In [28]:
```
df.shape[0]
```

Out[28]:
```
10841
```

Observations

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

- If you examive instagram rows have different number of reviews, which means the data was collected at different times. We won't remove rows randomly, rather we'll keep the row with the maximum number of reviews because the higher the reviews, the more reliable the ratings.

In [29]:
```
# Get the numbers of max reviews for each app
rev_max = df.groupby('App')['Reviews'].max()
```

In [30]:
```
rev_max['Instagram']
```

Out[30]:
```
'66577446'
```

In [31]:
```
duplicated_app[duplicated_app['App'] == 'Instagram']
```

Out[31]:

	App	Category	Rating	Reviews	Size_in_bytes	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
2545	Instagram	SOCIAL	4.5	66577313	NaN	1,000,000,000+	Free	0	Teen	Social	July 31, 2018	Varies with device	Varies with device
2604	Instagram	SOCIAL	4.5	66577446	NaN	1,000,000,000+	Free	0	Teen	Social	July 31, 2018	Varies with device	Varies with device
2611	Instagram	SOCIAL	4.5	66577313	NaN	1,000,000,000+	Free	0	Teen	Social	July 31, 2018	Varies with device	Varies with device
3909	Instagram	SOCIAL	4.5	66509917	NaN	1,000,000,000+	Free	0	Teen	Social	July 31, 2018	Varies with device	Varies with device

Now, use the `rev_max` dictionary to keep all the duplicating app which the maximum number of reviews and remove rest duplicating rows

In [32]:
```
# create an empty list
andriod_clean = []

# create an empty list to keep track on already added apps
already_added = []

#Iterate through each row of the DataFrame
for index, row in df.iterrows():
    name=row['App']
    n_review=row['Reviews']

# check if the current app has the maximum number of reviews and has not added before
    if (rev_max[name] == n_review) and (name not in already_added):
        andriod_clean.append(row) # Add the app to the andriod_cleaned list
        already_added.append(name) # add the app name to the already_added list
```

In [33]:
```
andriod_clean = pd.DataFrame(andriod_clean)
```

Removing Non-English Apps

If you explore dataset enought, you'll find out that names of some of the apps suggest they are not directed towards an English-speaking audience. Below we see couple examples of from both data sets.

In [34]:
```
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 [35]:
```
is_english('Instagram')
```

Out[35]:
```
True
```

In [36]:
```
is_english('Jokes 😀')
```

Out[36]:
```
False
```

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

Modifying 'is_english' function:

In [37]:
```
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 [38]:
```
is_english('Jokes 😂')
```

Out[38]:
```
True
```

In [39]:
```
is_english('Jokes 😂😊')
```

Out[39]:
```
False
```

In [40]:
```
andriod_english = andriod_clean[andriod_clean['App'].apply(is_english)]
andriod_english.head()
```

Out[40]:

	App	Category	Rating	Reviews	Size_in_bytes	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	An
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19922944.0	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	1.0.0	a
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	9122611.2	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	1.2.4	a
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	26214400.0	50,000,000+	Free	0	Teen	Art & Design	June 8, 2018	Varies with device	4.
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2936012.8	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.
5	Paper flowers instructions	ART_AND_DESIGN	4.4	167	5872025.6	50,000+	Free	0	Everyone	Art & Design	March 26, 2017	1.0	2.

Isolating the Free Apps

As we metioned in the introduction, we only build apps that are free to download and instal, and our main source of revence consists of in-apps ads. Our datasets contain both free and non-free apps, and we'll need to isolate only the free apps for our analysis for both the datasets

Target Column : `Price`

In [41]:
```
andriod_english['Price'].unique()
```

Out[41]:
```
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', '$1.20', '$1.04'], dtype=object)
```

In [42]:
```
andriod_final = andriod_english[andriod_english['Price'] == '0']
```

In [43]:
```
andriod_final.head()
```

Out[43]:

	App	Category	Rating	Reviews	Size_in_bytes	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	An
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19922944.0	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	1.0.0	a
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	9122611.2	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	1.2.4	a
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	26214400.0	50,000,000+	Free	0	Teen	Art & Design	June 8, 2018	Varies with device	4.
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2936012.8	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.
5	Paper flowers instructions	ART_AND_DESIGN	4.4	167	5872025.6	50,000+	Free	0	Everyone	Art & Design	March 26, 2017	1.0	2.

In [44]:
```
andriod_final.shape
```

Out[44]:
```
(8863, 13)
```

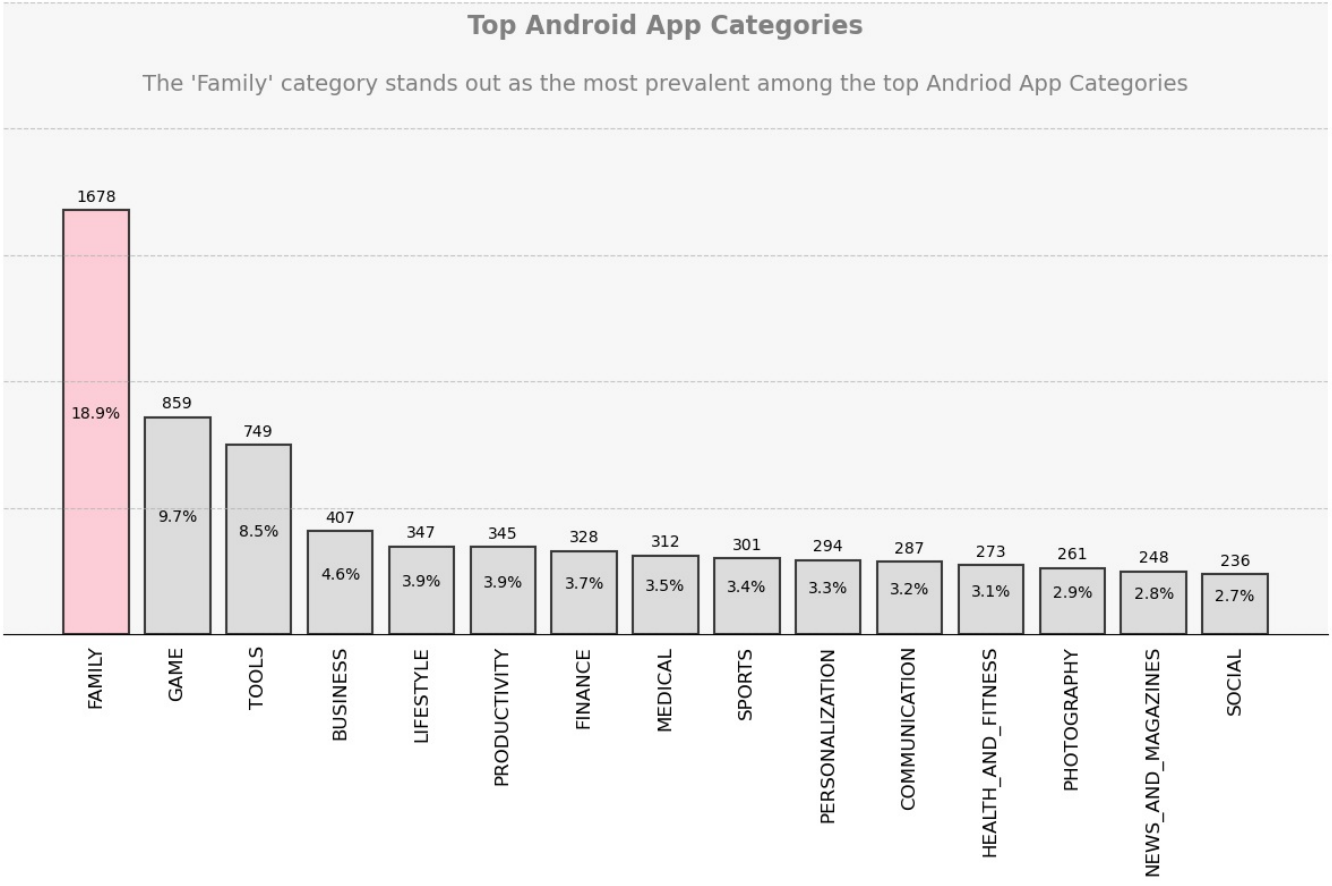
Analysis

In [45]:
```
andriod_final['Category'].value_counts(normalize = True)*100
```

Out[45]:
```
FAMILY                 18.932641
GAME                    9.691978
TOOLS                   8.450863
BUSINESS                4.592125
LIFESTYLE               3.915153
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
Name: Category, dtype: float64
```

In [65]:
```
categories = andriod_final["Category"].value_counts().index[:15]
counts = andriod_final["Category"].value_counts().values[:15]
percentage = round(andriod_final["Category"].value_counts(normalize =True)*100,1)[:15]

#Create a stylish bar chart
plt.figure(figsize=(12, 8))
bars = plt.bar(categories, counts, color='lightgray', alpha=0.75, edgecolor ='black', linewidth = '1.5')
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 customized y-ticks range
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("pink")
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='center', va='bottom', fonts
    plt.text(bar.get_x() + bar.get_width()/ 2,height/2, f'{perc}%', ha='center', va='bottom', fontsize=10, colo

#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 =16, transform=plt.gca
         color = 'gray', fontweight = 'bold')


#Adding a text annotation
plt.text(0.5, 0.86, "The 'Family' category stands out as the most prevalent among the top Andriod App Categorie
         horizontalalignment="center", fontsize=14, transform=plt.gca().transAxes, color='gray')

#Removing spines
for i in ["top", "right", "left"]:
    plt.gca().spines[i].set_visible(False)

#Adjust layout to prevent clipping
plt.tight_layout()

plt.show()
```



In [47]:
```
andriod_final[andriod_final['Category'] == 'FAMILY']
```

Out[47]:

	App	Category	Rating	Reviews	Size_in_bytes	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	An
2017	Jewels Crush-Match 3 Puzzle	FAMILY	4.4	14774	19922944.0	1,000,000+	Free	0	Everyone	Casual;Brain Games	July 23, 2018	1.9.3901	a
2018	Coloring & Learn	FAMILY	4.4	12753	53477376.0	5,000,000+	Free	0	Everyone	Educational;Creativity	July 17, 2018	1.49	a
2019	Mahjong	FAMILY	4.5	33983	23068672.0	5,000,000+	Free	0	Everyone	Puzzle;Brain Games	August 2, 2018	1.24.3181	a
2020	Super ABC! Learning games for kids! Preschool ...	FAMILY	4.6	20267	48234496.0	1,000,000+	Free	0	Everyone	Educational;Education	July 16, 2018	1.1.6.7	4.
2021	Toy Pop Cubes	FAMILY	4.5	5761	22020096.0	1,000,000+	Free	0	Everyone	Casual;Brain Games	July 4, 2018	1.8.3181	a
...	...	...	...	...	...	...	...	...	...	...	...	...	
10821	Poop FR	FAMILY	NaN	6	2621440.0	50+	Free	0	Everyone	Entertainment	May 29, 2018	1.0	a
10827	Fr Agnel Ambarnath	FAMILY	4.2	117	13631488.0	5,000+	Free	0	Everyone	Education	June 13, 2018	2.0.20	a
10834	FR Calculator	FAMILY	4.0	7	2726297.6	500+	Free	0	Everyone	Education	June 18, 2017	1.0.0	4.
10836	Sya9a Maroc - FR	FAMILY	4.5	38	55574528.0	5,000+	Free	0	Everyone	Education	July 25, 2017	1.48	4.
10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3774873.6	100+	Free	0	Everyone	Education	July 6, 2018	1.0	4.

1678 rows × 13 columns

The Most Popular Apps by Genre

For the Google Play market, we actually have data about the 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+,1000+,5000+)

In [48]:
```
andriod_final['Installs'].value_counts(normalize = True)*100
```

Out[48]:
```
1,000,000+        15.739592
100,000+          11.553650
10,000,000+       10.515627
10,000+           10.199707
1,000+             8.405732
100+               6.916394
5,000,000+         6.837414
500,000+           5.573733
50,000+            4.772650
5,000+             4.513145
10+                3.542818
500+               3.249464
50,000,000+        2.290421
100,000,000+       2.121178
50+                1.918086
5+                 0.789800
1+                 0.507729
500,000,000+       0.270789
1,000,000,000+     0.225657
0+                 0.045131
0                  0.011283
Name: Installs, dtype: float64
```

In [49]:
```
andriod_final['Installs_int'] = andriod_final['Installs'].str.replace('+','').str.replace(',','').astype(int)
andriod_final['Installs_int']
```

Out[49]:
```
0           10000
2         5000000
3        50000000
4          100000
5           50000
           ...
10836        5000
10837         100
10838        1000
10839        1000
10840    10000000
Name: Installs_int, Length: 8863, dtype: int32
```

In [50]:
```
install_frq = andriod_final['Installs_int'].value_counts().sort_index()
install_frq = install_frq[install_frq.index > 500]
install_frq
```

Out[50]:
```
1000           745
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 [51]:
```
install_frq_per = round(andriod_final['Installs_int'].value_counts(normalize = True)*100,2).sort_index()
install_frq_per = install_frq_per[install_frq_per.index > 500]
install_frq_per
```

Out[51]:
```
1000           8.41
5000           4.51
10000         10.20
50000          4.77
100000        11.55
500000         5.57
1000000       15.74
5000000        6.84
10000000      10.52
50000000       2.29
100000000      2.12
500000000      0.27
1000000000     0.23
Name: Installs_int, dtype: float64
```

In [52]:
```
#aplha numeric unit

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 [53]:
```
alphanumeric_units(1000000000)
```

Out[53]:
```
'1B'
```

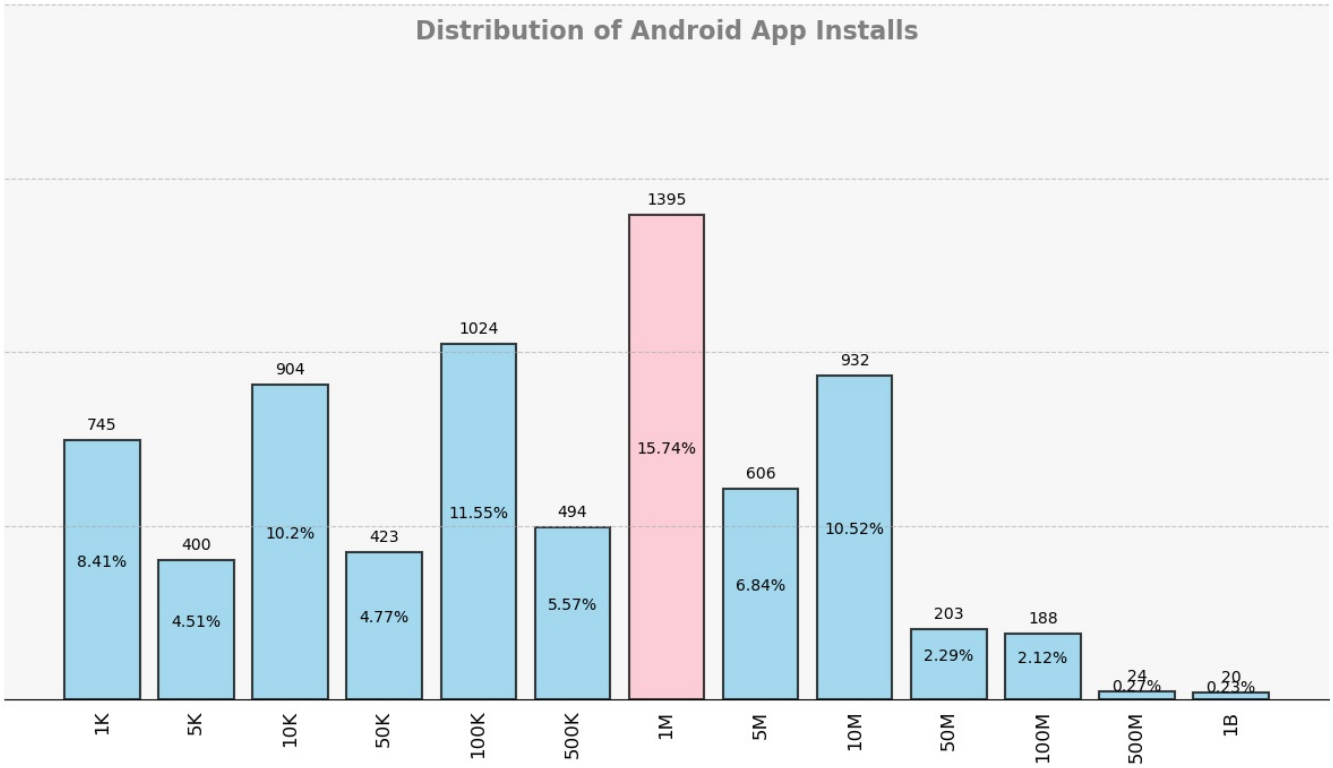
In [54]:
```
install_frq
```

Out[54]:
```
1000           745
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 [55]:
```
install_frq.index = install_frq.index.map(alphanumeric_units)
install_frq
```

Out[55]:
```
1K       745
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 [56]:
```
categories = install_frq.index
counts = install_frq.values
percentage1 = install_frq_per.values

#Create a stylish bar chart
plt.figure(figsize=(12, 7))
bars = plt.bar(categories, counts, color='skyblue', alpha=0.75, edgecolor ='black', linewidth = '1.5')
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 customized y-ticks range
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("pink")
bars[max_count_index].set_edgecolor('black')

# Adding data Labels and percentages inside each bar
for bar, perc in zip(bars, percentage1):
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/ 2, height + 20, "%d" % int(height), ha='center', va='bottom', fonts
    plt.text(bar.get_x() + bar.get_width()/ 2,height/2, f'{perc}%', ha='center', va='bottom', fontsize=10, colo

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

# Adding CHART TITLE inside the chart
plt.text(0.5, 0.95, 'Distribution of Android App Installs', horizontalalignment="center", fontsize =16, \
         transform=plt.gca().transAxes, color = 'gray', fontweight = 'bold')

#Removing spines
for i in ["top", "right", "left"]:
    plt.gca().spines[i].set_visible(False)

#Adjust layout to prevent clipping
plt.tight_layout()

plt.show()
```



Observation

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

In [57]:
```
categories_android = andriod_final['Category'].unique()
categories_android
```

Out[57]:
```
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 [58]:
```
pd.pivot_table(andriod_final, values='Installs_int', index = 'Category', aggfunc = 'mean')
```

Out[58]:

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	1.433676e+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 [59]:
```
# Display dataframe without scientific notation
pd.options.display.float_format = '{:.0f}'.format
```

In [60]:
```
categories_installs = pd.pivot_table(andriod_final, values='Installs_int', index = 'Category', aggfunc = 'mean'
categories_installs = categories_installs.sort_values(by = 'Installs_int', ascending = False)
categories_installs = categories_installs['Installs_int']
categories_installs
```

Out[60]:
```
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              1433676
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
Name: Installs_int, dtype: float64
```

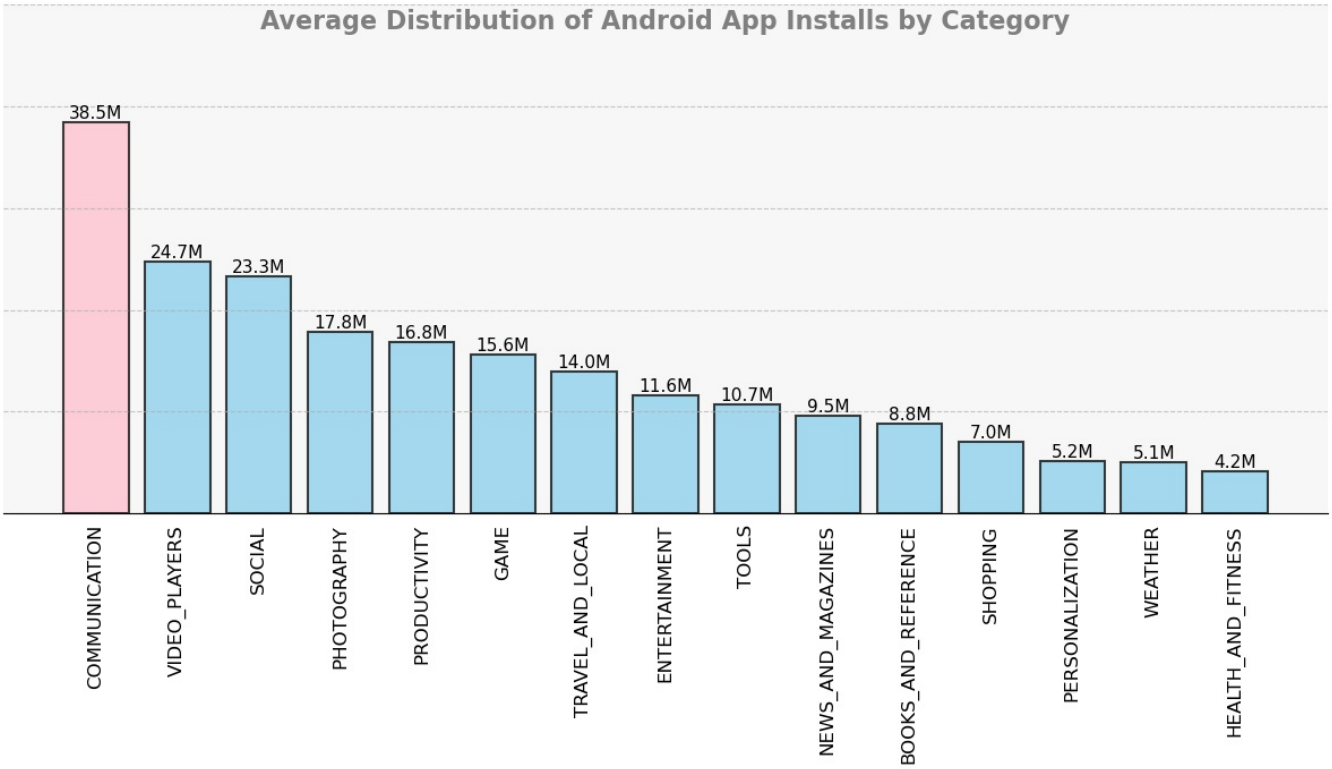
In [61]:
```
def alphanum_units(value):
    if value >= 1e9:
        return f'{value/ 1e9:.1f}B'
    elif value >= 1e6:
        return f'{value / 1e6:.1f}M'
    elif value >= 1e3:
        return f'{value / 1e3:.1f}K'
    else:
        return f'{value:.1f}'
```

In [62]:
```
categories_installs_units = categories_installs.map(alphanum_units)
categories_installs_units
```

Out[62]:
```
Category
COMMUNICATION          38.5M
VIDEO_PLAYERS          24.7M
SOCIAL                 23.3M
PHOTOGRAPHY            17.8M
PRODUCTIVITY           16.8M
GAME                   15.6M
TRAVEL_AND_LOCAL       14.0M
ENTERTAINMENT          11.6M
TOOLS                  10.7M
NEWS_AND_MAGAZINES      9.5M
BOOKS_AND_REFERENCE     8.8M
SHOPPING                7.0M
PERSONALIZATION         5.2M
WEATHER                 5.1M
HEALTH_AND_FITNESS      4.2M
MAPS_AND_NAVIGATION     4.1M
FAMILY                  3.7M
SPORTS                  3.6M
ART_AND_DESIGN          2.0M
FOOD_AND_DRINK          1.9M
EDUCATION               1.8M
BUSINESS                1.7M
LIFESTYLE               1.4M
FINANCE                 1.4M
HOUSE_AND_HOME          1.3M
DATING                854.0K
COMICS                817.7K
AUTO_AND_VEHICLES     647.3K
LIBRARIES_AND_DEMO    638.5K
PARENTING             542.6K
BEAUTY                513.2K
EVENTS                253.5K
MEDICAL               120.6K
Name: Installs_int, dtype: object
```

In [64]:
```
categories = categories_installs.index[:15]
counts = categories_installs.values[:15]
```

```
#Create a stylish bar chart
plt.figure(figsize=(12, 7))
bars = plt.bar(categories, counts, color='skyblue', alpha=0.75, edgecolor ='black', linewidth = '1.5')
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 customized y-ticks range
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("pink")
bars[max_count_index].set_edgecolor('black')

# Adding data Labels and percentages 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', va='bottom', fontsize=11)

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

# Adding CHART TITLE inside the chart
plt.text(0.5, 0.95, 'Average Distribution of Android App Installs by Category', horizontalalignment="center", \
         fontsize =16, transform=plt.gca().transAxes, color = 'gray', fontweight = 'bold')

#Removing spines
for i in ["top", "right", "left"]:
    plt.gca().spines[i].set_visible(False)

#Adjust layout to prevent clipping
plt.tight_layout()

plt.show()
```



Observation

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

Analysis of Photography Category andPotential for Photo Generation App in 2024

Conclusion

The analysis of the photography category reveals a notable trend in the populanty of photoediting and collage-making applications, with several apps garnering 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 development of aphoto generation application in 2024. Such an app, offering prompt and free generation ofpictures and photos, could capitalize on the existing user interest in photography apps. Byproviding innovative features, easy usability and high-quality output, this application couldstand out in the competitive market and attract a large user base. Considering the success of existing photography apps and the evolving preferences ofusers, investing in the development of a photo generation app seems promising for tappinginto this lucrative market segment in 2024.