# Google Play Store Analysis


## **Author:** Danish Azeem

## **Email Address:** danishazeem365@gmail.com

## **Github:** https://github.com/danishazeem365


### Project Overview

## About Dataset

The Data Set was downloaded from Kaggle, from the following link

## Google PlayStore Android App Data. (2.3 Million+ App Data)

## Introduction

The Google Play Store is one of the largest app marketplaces, hosting millions of apps across various categories. Understanding the patterns in app ratings, downloads, pricing, and monetization strategies is essential for developers, businesses, and researchers.

In this project, we perform an in-depth analysis of Google Play Store data to uncover trends and insights. This includes:

- **Data Cleaning & Preprocessing**: Handling duplicates, missing values, and formatting inconsistencies.
- **Exploratory Data Analysis (EDA)**: Understanding app distribution, pricing models, and user engagement.
- **Correlation & Trends**: Investigating relationships between ratings, installs, and other key factors.
- **Visualization**: Using charts and graphs to present insights effectively.

## Objectives

 Identify key trends in app categories, pricing, and ratings.
 Analyze how installs correlate with rating count and app size.
 Explore the distribution of free vs. paid apps and their pricing patterns.
 Provide actionable insights for app developers and businesses.

By the end of this analysis, we will have a comprehensive understanding of Google Play Store trends, which can help in making data-driven decisions for app development and marketing strategies.

---

# 1. Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from google.colab import drive
drive.mount('/content/drive')
df = pd.read_csv('/content/drive/My Drive/Data Sets/Google-
Playstore.csv')
print(df.head())

Mounted at /content/drive
                                              App Name  \
0                                             Gakondo
1                                  Ampere Battery Info
2                                              Vibook
3  Smart City Trichy Public Service Vehicles 17UC...
4                                             GROW.me

                        App Id       Category  Rating  Rating Count
Installs  \
0         com.ishakwe.gakondo      Adventure     0.0           0.0
10+
1  com.webserveis.batteryinfo          Tools     4.4          64.0
5,000+
2        com.doantiepvien.crm   Productivity     0.0           0.0
50+
3     cst.stJoseph.ug17ucs548  Communication     5.0           5.0
10+
4        com.horodyski.grower          Tools     0.0           0.0
100+

   Minimum Installs  Maximum Installs  Free  Price  ...  \
0              10.0                15  True    0.0  ...
1            5000.0              7662  True    0.0  ...
2              50.0                58  True    0.0  ...
3              10.0                19  True    0.0  ...
4             100.0               478  True    0.0  ...

                  Developer Website               Developer Email  \
0       https://beniyizibyose.tk/#/       jean21101999@gmail.com
```

```
1   https://webserveis.netlify.app/         webserveis@gmail.com
2                               NaN          vnacrewit@gmail.com
3  http://www.climatesmarttech.com/  climatesmarttech2@gmail.com
4       http://www.horodyski.com.pl    rmilekhorodyski@gmail.com

       Released  Last Updated Content Rating  \
0  Feb 26, 2020  Feb 26, 2020       Everyone
1  May 21, 2020  May 06, 2021       Everyone
2   Aug 9, 2019  Aug 19, 2019       Everyone
3  Sep 10, 2018  Oct 13, 2018       Everyone
4  Feb 21, 2020  Nov 12, 2018       Everyone

                                    Privacy Policy Ad Supported  \
0                https://beniyizibyose.tk/projects/        False
1  https://dev4phones.wordpress.com/licencia-de-uso/         True
2  https://www.vietnamairlines.com/vn/en/terms-an...        False
3                                                NaN         True
4                       http://www.horodyski.com.pl        False

  In App Purchases Editors Choice         Scraped Time
0            False          False  2021-06-15 20:19:35
1            False          False  2021-06-15 20:19:35
2            False          False  2021-06-15 20:19:35
3            False          False  2021-06-15 20:19:35
4            False          False  2021-06-15 20:19:35

[5 rows x 24 columns]
```

# 2. Data Loading and exploration and cleaning

↪ Load the csv file with the pandas

↪ creating the dataframe and understanding the data present in the dataset using pandas

↪ Dealing with the missing data, outliers and the incorrect records

- Viewing the first five Rows of the data

```
df.head()
```

{"type":"dataframe","variable_name":"df"}

**Note**: Some the output of notebook does not present the complete output, therefore we can increase the limit of columns view and row view by using these commands:

```
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
```

```
# hide all warnings runtime
import warnings
warnings.filterwarnings('ignore')
```

- let's see the exact column names which can be easily copied later on from Google Playstore Dataset

```
df.columns
```

```
Index(['App Name', 'App Id', 'Category', 'Rating', 'Rating Count',
'Installs',
       'Minimum Installs', 'Maximum Installs', 'Free', 'Price',
'Currency',
       'Size', 'Minimum Android', 'Developer Id', 'Developer Website',
       'Developer Email', 'Released', 'Last Updated', 'Content
Rating',
       'Privacy Policy', 'Ad Supported', 'In App Purchases', 'Editors
Choice',
       'Scraped Time'],
      dtype='object')
```

```
df.shape
```

```
(2312944, 24)
```

Not enough, let's have a look on the columns and their data types using detailed info function

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2312944 entries, 0 to 2312943
Data columns (total 24 columns):
 #   Column             Dtype
---  ------             -----
 0   App Name           object
 1   App Id             object
 2   Category           object
 3   Rating             float64
 4   Rating Count       float64
 5   Installs           object
 6   Minimum Installs   float64
 7   Maximum Installs   int64
 8   Free               bool
 9   Price              float64
 10  Currency           object
 11  Size               object
 12  Minimum Android    object
 13  Developer Id       object
 14  Developer Website  object
 15  Developer Email    object
```

```
 16  Released           object
 17  Last Updated       object
 18  Content Rating     object
 19  Privacy Policy     object
 20  Ad Supported       bool
 21  In App Purchases   bool
 22  Editors Choice     bool
 23  Scraped Time       object
dtypes: bool(4), float64(4), int64(1), object(15)
memory usage: 361.8+ MB
```

# # Observations

1. There are 10841 rows and 24 columns in the dataset
2. The columns are of different data types
3. The columns in the datasets are:
   - 'App Name', 'App Id', 'Category', 'Rating', 'Rating Count', 'Installs', 'Minimum Installs', 'Maximum Installs', 'Free', 'Price', 'Currency', 'Size', 'Minimum Android', 'Developer Id', 'Developer Website', 'Developer Email', 'Released', 'Last Updated', 'Content Rating', 'Privacy Policy', 'Ad Supported', 'In App Purchases', 'Editors Choice', 'Scraped Time',
4. There are some missing values in the dataset which we will read in details and deal later on in the notebook.
5. There are some columns which are of object data type but they should be of numeric data type, we will convert them later on in the notebook.
   - `'Installs','Size', 'Minimum Android'`

```
df.describe()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8,\n  \"fields\": [\n
{\n      \"column\": \"Rating\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 809657.997368076,\n
\"min\": 0.0,\n        \"max\": 2290061.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n
2290061.0,\n          2.203151531771424,\n          4.3\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Rating Count\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
48867391.570589274,\n        \"min\": 0.0,\n        \"max\":
138557570.0,\n        \"num_unique_values\": 7,\n        \"samples\":
[\n          2290061.0,\n          2864.8388767810115,\n
42.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Minimum Installs\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 3534647129.2584343,\n        \"min\":
0.0,\n        \"max\": 10000000000.0,\n        \"num_unique_values\":
8,\n        \"samples\": [\n          183445.21395800915,\n
500.0,\n          2312837.0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
```

```
\"column\": \"Maximum Installs\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 4261699552.060366,\n
\"min\": 0.0,\n        \"max\": 12057627016.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
320201.713137456,\n          695.0,\n          2312944.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Price\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 817728.8641336214,\n
\"min\": 0.0,\n        \"max\": 2312944.0,\n
\"num_unique_values\": 5,\n        \"samples\": [\n
0.10349915833370804,\n          400.0,\n          2.63312655760054\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


## Observations:

- We have 5 columns as numeric data type, rest all are object data type (according to python), but we can see that `'Installs','Size', 'Minimum Android'` are also numeric, we must convert them to numeric data type in data wrangling process.

- Let's clean the `Size` column first

```
df['Installs'].isnull().sum()
```

```
107
```

- We have 107 null values

```
df['Installs'] = df['Installs'].fillna('')

# Replace empty strings with '0'
df['Installs'] = df['Installs'].replace('', '0')
```

- Firstly we have to fill null values with '' secondly replace it with 0

```
df['Installs'].isnull().sum()
```

```
0
```

- No null values, we are good to go.

```
df['Installs'].unique()
```

```
array(['10+', '5,000+', '50+', '100+', '1,000+', '500+', '50,000+',
       '10,000+', '1+', '500,000+', '100,000+', '5+', '10,000,000+',
       '1,000,000+', '5,000,000+', '0+', '100,000,000+', '0',
```

```
       '50,000,000+', '1,000,000,000+', '500,000,000+',
'5,000,000,000+',
       '10,000,000,000+'], dtype=object)

df['Installs'].value_counts()

Installs
100+                 443368
1,000+               398199
10+                  300156
10,000+              256723
500+                 189077
50+                  170465
5,000+               143593
100,000+             110257
50,000+               75359
5+                    73772
1+                    65345
1,000,000+            33650
500,000+              27012
0+                    11566
5,000,000+             6595
10,000,000+            6192
50,000,000+             824
100,000,000+            549
0                       107
500,000,000+             65
1,000,000,000+           55
5,000,000,000+           14
10,000,000,000+           1
Name: count, dtype: int64

# find how many values has '+' in it
df['Installs'].loc[df['Installs'].str.contains('\\
+')].value_counts().sum()

2312837
```

- The only problem I see here is the `+` sign in the values, let's remove them and convert the column into numeric data type.
- The total values in the `Installs` column are `10841` and there are no null values in the column.
- However, one value 0 has no plus sign
- Let's remove the plus sign `+` and `,` from the values and convert them into numeric data type

```
# remove the plus sign from install column and convert it to numeric
df['Installs'] = df['Installs'].apply(lambda x: x.replace('+', '') if
```

```
'+' in str(x) else x)
# also remove the commas from the install column
df['Installs'] = df['Installs'].apply(lambda x: x.replace(',', '') if
',' in str(x) else x)
# convert the install column to numeric (integers because this is the
number of installs/count)
df['Installs'] = df['Installs'].apply(lambda x: int(x))
```

- Let's verify if the dtypes has been changes and the `+` and `,` sign has been removed

```
df.head() # check the head of the dataframe
```

```
{"type":"dataframe","variable_name":"df"}
```

```
df['Installs'].dtype # this will show the data type of the column
```

```
dtype('int64')
```

- We can generate a new columns based on the installation values, which will be helpful in our analysis

```
df['Installs'].max() # this will show the value counts of the column
```

```
10000000000
```

```
# making a new column called 'Installs_category' which will have the
category of the installs
bins = [-1, 0, 10, 1000, 10000, 100000, 1000000, 10000000,
10000000000]
labels=['no', 'Very low', 'Low', 'Moderate', 'More than moderate',
'High', 'Very High', 'Top Notch']
df['Installs_category'] = pd.cut(df['Installs'], bins=bins,
labels=labels)

df['Installs_category'].value_counts() # check the value counts of the
new column
```

```
Installs_category
Low                  1201109
Very low              439273
Moderate              400316
More than moderate    185616
High                   60662
Very High              12787
no                     11673
Top Notch               1508
Name: count, dtype: int64
```

- Let's clean the `Size` column

```
# check for null values
df['Size'].isnull().sum()
```

```
196
```

```
df['Size'] = df['Size'].fillna('0M')
```

- No null values, we are good to go.

```
df['Size'].value_counts()
```

```
Size
Varies with device    74777
11M                   62157
12M                   56080
13M                   48034
14M                   45211
16M                   42474
15M                   41306
17M                   37244
10M                   34114
18M                   31707
19M                   29723
21M                   29023
20M                   28796
22M                   28261
23M                   27337
24M                   25301
25M                   25023
26M                   23897
27M                   21826
28M                   21157
29M                   20050
37M                   18658
30M                   18625
31M                   18150
4.9M                  17967
32M                   17629
4.2M                  17421
3.8M                  17193
3.7M                  16927
38M                   16811
3.4M                  16752
33M                   16681
4.0M                  16481
3.5M                  16379
3.9M                  16288
3.3M                  16233
4.3M                  16155
3.6M                  15949
```

| | |
|---|---|
| 4.1M | 15829 |
| 34M | 15163 |
| 4.4M | 14946 |
| 3.2M | 14936 |
| 36M | 14733 |
| 4.5M | 14731 |
| 2.8M | 14603 |
| 2.9M | 14582 |
| 5.2M | 14457 |
| 3.0M | 14418 |
| 5.6M | 14382 |
| 5.0M | 14185 |
| 2.7M | 14114 |
| 5.3M | 14090 |
| 4.7M | 14081 |
| 3.1M | 13995 |
| 4.6M | 13924 |
| 35M | 13846 |
| 4.8M | 13758 |
| 5.8M | 13450 |
| 2.6M | 13363 |
| 5.5M | 13333 |
| 5.4M | 13201 |
| 5.7M | 13200 |
| 2.5M | 12790 |
| 6.0M | 12749 |
| 5.1M | 12694 |
| 52M | 12294 |
| 5.9M | 12258 |
| 6.1M | 12097 |
| 2.4M | 11867 |
| 6.8M | 11830 |
| 39M | 11681 |
| 6.4M | 11666 |
| 2.3M | 11640 |
| 6.3M | 11488 |
| 6.5M | 11435 |
| 6.2M | 11435 |
| 1.4M | 11424 |
| 1.9M | 11327 |
| 6.7M | 11225 |
| 2.0M | 11224 |
| 2.2M | 11177 |
| 2.1M | 11164 |
| 7.4M | 11072 |
| 6.6M | 11063 |
| 7.7M | 10959 |
| 7.2M | 10833 |
| 6.9M | 10783 |

| | |
|---|---|
| 40M | 10735 |
| 7.5M | 10670 |
| 7.3M | 10621 |
| 7.0M | 10399 |
| 8.7M | 10332 |
| 1.5M | 10308 |
| 1.8M | 10307 |
| 7.6M | 10277 |
| 8.5M | 10263 |
| 7.1M | 10251 |
| 7.9M | 10130 |
| 1.7M | 9901 |
| 8.2M | 9899 |
| 8.3M | 9883 |
| 41M | 9836 |
| 7.8M | 9746 |
| 8.6M | 9730 |
| 42M | 9643 |
| 8.4M | 9606 |
| 8.0M | 9528 |
| 8.8M | 9352 |
| 43M | 9316 |
| 9.3M | 9301 |
| 8.1M | 9288 |
| 1.6M | 9280 |
| 9.6M | 9262 |
| 8.9M | 9066 |
| 1.3M | 9049 |
| 53M | 9022 |
| 9.1M | 8762 |
| 9.2M | 8692 |
| 9.0M | 8627 |
| 45M | 8463 |
| 44M | 8462 |
| 47M | 8327 |
| 46M | 8216 |
| 49M | 8201 |
| 48M | 8187 |
| 9.4M | 8031 |
| 9.5M | 7939 |
| 9.8M | 7925 |
| 9.9M | 7726 |
| 9.7M | 7681 |
| 51M | 7628 |
| 50M | 7480 |
| 1.2M | 6882 |
| 54M | 6711 |
| 55M | 5888 |
| 1.1M | 5821 |

| | |
|---|---|
| 58M | 5588 |
| 57M | 5356 |
| 56M | 5319 |
| 62M | 4715 |
| 60M | 4671 |
| 59M | 4631 |
| 61M | 4369 |
| 63M | 3826 |
| 64M | 3559 |
| 66M | 3520 |
| 10.0M | 3513 |
| 68M | 3421 |
| 65M | 3412 |
| 67M | 3242 |
| 73M | 3208 |
| 69M | 3042 |
| 71M | 3038 |
| 70M | 2990 |
| 72M | 2856 |
| 74M | 2721 |
| 77M | 2574 |
| 75M | 2483 |
| 79M | 2477 |
| 78M | 2417 |
| 80M | 2391 |
| 76M | 2360 |
| 1.0M | 2303 |
| 91M | 2270 |
| 94M | 2181 |
| 84M | 2155 |
| 98M | 2148 |
| 81M | 2146 |
| 83M | 2109 |
| 82M | 2069 |
| 85M | 2016 |
| 95M | 1910 |
| 88M | 1910 |
| 90M | 1879 |
| 87M | 1864 |
| 89M | 1861 |
| 86M | 1856 |
| 99M | 1818 |
| 92M | 1753 |
| 93M | 1745 |
| 96M | 1721 |
| 97M | 1693 |
| 100M | 1586 |
| 147M | 1130 |
| 101M | 1089 |

| | |
|---|---|
| 149M | 876 |
| 150M | 869 |
| 151M | 845 |
| 102M | 844 |
| 148M | 785 |
| 140M | 759 |
| 103M | 719 |
| 118M | 597 |
| 104M | 569 |
| 106M | 560 |
| 152M | 557 |
| 105M | 548 |
| 107M | 530 |
| 112M | 493 |
| 110M | 472 |
| 109M | 450 |
| 108M | 443 |
| 113M | 430 |
| 114M | 429 |
| 111M | 419 |
| 116M | 412 |
| 125M | 409 |
| 201M | 408 |
| 142M | 403 |
| 121M | 397 |
| 115M | 389 |
| 122M | 382 |
| 119M | 369 |
| 153M | 369 |
| 141M | 345 |
| 117M | 331 |
| 123M | 330 |
| 124M | 302 |
| 120M | 289 |
| 139M | 289 |
| 126M | 284 |
| 135M | 277 |
| 130M | 271 |
| 129M | 271 |
| 146M | 267 |
| 128M | 262 |
| 134M | 257 |
| 145M | 254 |
| 131M | 246 |
| 137M | 233 |
| 127M | 230 |
| 136M | 225 |
| 154M | 223 |
| 133M | 222 |

| | |
|---|---|
| 138M | 219 |
| 132M | 204 |
| 0M | 196 |
| 144M | 194 |
| 143M | 179 |
| 155M | 134 |
| 156M | 125 |
| 17k | 108 |
| 157M | 100 |
| 158M | 94 |
| 588k | 94 |
| 21k | 92 |
| 159M | 92 |
| 33k | 90 |
| 253k | 90 |
| 29k | 90 |
| 163M | 88 |
| 162M | 86 |
| 34k | 82 |
| 160M | 81 |
| 165M | 81 |
| 25k | 79 |
| 45k | 76 |
| 41k | 75 |
| 257k | 73 |
| 218k | 73 |
| 167M | 72 |
| 180M | 72 |
| 256k | 71 |
| 53k | 71 |
| 49k | 70 |
| 241k | 70 |
| 164M | 69 |
| 214k | 69 |
| 27k | 67 |
| 217k | 67 |
| 166M | 67 |
| 23k | 66 |
| 230k | 65 |
| 106k | 64 |
| 173M | 64 |
| 16k | 64 |
| 28k | 64 |
| 158k | 64 |
| 37k | 63 |
| 169M | 63 |
| 365k | 63 |
| 244k | 63 |
| 86k | 63 |

| | |
|---|---|
| 66k | 63 |
| 161M | 62 |
| 154k | 61 |
| 225k | 61 |
| 222k | 61 |
| 165k | 61 |
| 265k | 60 |
| 122k | 60 |
| 32k | 59 |
| 237k | 59 |
| 26k | 59 |
| 178k | 59 |
| 54k | 59 |
| 171M | 59 |
| 57k | 58 |
| 58k | 57 |
| 1,005k | 57 |
| 254k | 57 |
| 245k | 56 |
| 194k | 56 |
| 61k | 56 |
| 44k | 56 |
| 960k | 56 |
| 168M | 55 |
| 255k | 54 |
| 46k | 54 |
| 24k | 54 |
| 166k | 54 |
| 227k | 54 |
| 118k | 54 |
| 169k | 54 |
| 59k | 54 |
| 50k | 54 |
| 36k | 53 |
| 226k | 53 |
| 947k | 53 |
| 273k | 53 |
| 931k | 53 |
| 924k | 53 |
| 246k | 53 |
| 308M | 53 |
| 193M | 53 |
| 894k | 53 |
| 249k | 52 |
| 782k | 52 |
| 210k | 52 |
| 82k | 52 |
| 161k | 52 |
| 359k | 51 |

| | |
|---|---|
| 98k | 51 |
| 157k | 51 |
| 234k | 51 |
| 67k | 51 |
| 919k | 51 |
| 186k | 51 |
| 1,009k | 51 |
| 252k | 51 |
| 22k | 51 |
| 943k | 51 |
| 229k | 51 |
| 261k | 51 |
| 198k | 51 |
| 966k | 51 |
| 74k | 51 |
| 141k | 51 |
| 73k | 51 |
| 105k | 51 |
| 130k | 51 |
| 999k | 51 |
| 30k | 50 |
| 820k | 50 |
| 47k | 50 |
| 52k | 50 |
| 902k | 50 |
| 188k | 50 |
| 170M | 50 |
| 78k | 50 |
| 69k | 50 |
| 150k | 50 |
| 201k | 50 |
| 20k | 50 |
| 176M | 50 |
| 151k | 50 |
| 184k | 49 |
| 224k | 49 |
| 996k | 49 |
| 258k | 49 |
| 110k | 49 |
| 146k | 49 |
| 1,004k | 49 |
| 149k | 49 |
| 94k | 49 |
| 206k | 48 |
| 68k | 48 |
| 85k | 48 |
| 153k | 48 |
| 911k | 48 |
| 242k | 48 |

| | |
|---|---|
| 1,012k | 48 |
| 932k | 48 |
| 247k | 48 |
| 91k | 48 |
| 1,022k | 48 |
| 101k | 48 |
| 162k | 48 |
| 770k | 48 |
| 174M | 47 |
| 215k | 47 |
| 142k | 47 |
| 1,020k | 47 |
| 899k | 47 |
| 31k | 47 |
| 1,023k | 47 |
| 134k | 47 |
| 114k | 47 |
| 65k | 47 |
| 950k | 47 |
| 912k | 47 |
| 990k | 47 |
| 369k | 47 |
| 126k | 47 |
| 185k | 47 |
| 236k | 47 |
| 205k | 47 |
| 266k | 46 |
| 514k | 46 |
| 97k | 46 |
| 109k | 46 |
| 72k | 46 |
| 360k | 46 |
| 885k | 46 |
| 211k | 46 |
| 861k | 46 |
| 220k | 46 |
| 197k | 46 |
| 90k | 46 |
| 936k | 46 |
| 413k | 46 |
| 60k | 46 |
| 788k | 46 |
| 40k | 45 |
| 213k | 45 |
| 974k | 45 |
| 625k | 45 |
| 107k | 45 |
| 156k | 45 |
| 955k | 45 |

```
721k                    45
200k                    45
39k                     45
906k                    45
983k                    45
250k                    45
172k                    45
722k                    45
856k                    45
976k                    45
223k                    45
774k                    45
889k                    45
826k                    45
863k                    44
216k                    44
174k                    44
83k                     44
353k                    44
282k                    44
968k                    44
193k                    44
877k                    44
212k                    44
81k                     44
1,021k                  44
676k                    44
270k                    44
791k                    44
434k                    44
64k                     44
221k                    44
19k                     44
980k                    44
285k                    44
238k                    44
994k                    44
937k                    43
552k                    43
882k                    43
219k                    43
231k                    43
190k                    43
946k                    43
631k                    43
390k                    43
179M                    43
839k                    43
89k                     43
```

| | |
|---|---|
| 172M | 43 |
| 35k | 43 |
| 181k | 43 |
| 997k | 43 |
| 297k | 43 |
| 286k | 43 |
| 75k | 43 |
| 1,011k | 43 |
| 87k | 43 |
| 1,008k | 43 |
| 196k | 43 |
| 915k | 43 |
| 240k | 43 |
| 888k | 43 |
| 38k | 43 |
| 922k | 42 |
| 202M | 42 |
| 70k | 42 |
| 177M | 42 |
| 874k | 42 |
| 232k | 42 |
| 989k | 42 |
| 177k | 42 |
| 138k | 42 |
| 77k | 42 |
| 790k | 42 |
| 963k | 42 |
| 952k | 42 |
| 1,016k | 42 |
| 824k | 42 |
| 102k | 42 |
| 178M | 42 |
| 18k | 42 |
| 929k | 42 |
| 778k | 42 |
| 841k | 42 |
| 967k | 42 |
| 284k | 42 |
| 954k | 42 |
| 55k | 42 |
| 891k | 42 |
| 1,018k | 42 |
| 233k | 42 |
| 182M | 42 |
| 334k | 41 |
| 917k | 41 |
| 926k | 41 |
| 987k | 41 |
| 389k | 41 |
| 113k | 41 |

| | |
|---|---|
| 43k | 41 |
| 935k | 41 |
| 274k | 41 |
| 871k | 41 |
| 927k | 41 |
| 160k | 41 |
| 934k | 41 |
| 858k | 41 |
| 329k | 41 |
| 806k | 41 |
| 813k | 41 |
| 843k | 41 |
| 164k | 41 |
| 933k | 41 |
| 779k | 40 |
| 42k | 40 |
| 904k | 40 |
| 825k | 40 |
| 362k | 40 |
| 739k | 40 |
| 944k | 40 |
| 145k | 40 |
| 367k | 40 |
| 763k | 40 |
| 1,000k | 40 |
| 328k | 40 |
| 890k | 40 |
| 930k | 40 |
| 80k | 40 |
| 263k | 40 |
| 799k | 40 |
| 209k | 40 |
| 939k | 40 |
| 194M | 40 |
| 855k | 40 |
| 259k | 40 |
| 876k | 40 |
| 433k | 40 |
| 925k | 40 |
| 678k | 39 |
| 959k | 39 |
| 895k | 39 |
| 893k | 39 |
| 63k | 39 |
| 277k | 39 |
| 321k | 39 |
| 755k | 39 |
| 182k | 39 |
| 76k | 39 |

| | |
|---|---|
| 941k | 39 |
| 696k | 39 |
| 962k | 39 |
| 786k | 39 |
| 762k | 39 |
| 716k | 39 |
| 730k | 39 |
| 916k | 39 |
| 886k | 39 |
| 243k | 39 |
| 998k | 39 |
| 1,001k | 39 |
| 127k | 39 |
| 326k | 39 |
| 753k | 39 |
| 239k | 39 |
| 437k | 39 |
| 589k | 39 |
| 827k | 39 |
| 13k | 39 |
| 796k | 39 |
| 798k | 39 |
| 370k | 39 |
| 692k | 39 |
| 1,014k | 39 |
| 235k | 39 |
| 985k | 38 |
| 407k | 38 |
| 970k | 38 |
| 898k | 38 |
| 913k | 38 |
| 880k | 38 |
| 385k | 38 |
| 1,006k | 38 |
| 181M | 38 |
| 857k | 38 |
| 910k | 38 |
| 883k | 38 |
| 191k | 38 |
| 879k | 38 |
| 15k | 38 |
| 864k | 38 |
| 281k | 38 |
| 406k | 38 |
| 48k | 38 |
| 417k | 38 |
| 830k | 38 |
| 708k | 38 |
| 147k | 38 |

| | |
|---|---|
| 713k | 38 |
| 278k | 38 |
| 168k | 38 |
| 942k | 38 |
| 992k | 38 |
| 386k | 38 |
| 405k | 37 |
| 399k | 37 |
| 746k | 37 |
| 410k | 37 |
| 1,007k | 37 |
| 837k | 37 |
| 56k | 37 |
| 981k | 37 |
| 228k | 37 |
| 409k | 37 |
| 212M | 37 |
| 303k | 37 |
| 780k | 37 |
| 975k | 37 |
| 279k | 37 |
| 290k | 37 |
| 1,019k | 37 |
| 414k | 37 |
| 938k | 37 |
| 175k | 37 |
| 862k | 37 |
| 397k | 37 |
| 526k | 37 |
| 831k | 37 |
| 969k | 37 |
| 836k | 37 |
| 900k | 37 |
| 810k | 37 |
| 832k | 37 |
| 850k | 37 |
| 51k | 37 |
| 816k | 37 |
| 117k | 37 |
| 907k | 37 |
| 965k | 37 |
| 361k | 37 |
| 984k | 37 |
| 872k | 37 |
| 317k | 37 |
| 821k | 37 |
| 865k | 36 |
| 148k | 36 |
| 914k | 36 |

| | |
|---|---|
| 209M | 36 |
| 448k | 36 |
| 421k | 36 |
| 323k | 36 |
| 262k | 36 |
| 964k | 36 |
| 642k | 36 |
| 462k | 36 |
| 314k | 36 |
| 884k | 36 |
| 308k | 36 |
| 875k | 36 |
| 446k | 36 |
| 685k | 36 |
| 62k | 36 |
| 597k | 36 |
| 702k | 36 |
| 309k | 36 |
| 873k | 36 |
| 100k | 36 |
| 993k | 36 |
| 350k | 36 |
| 170k | 36 |
| 881k | 36 |
| 373k | 36 |
| 478k | 36 |
| 199k | 36 |
| 743k | 36 |
| 973k | 36 |
| 453k | 36 |
| 357k | 36 |
| 88k | 36 |
| 757k | 36 |
| 495k | 36 |
| 393k | 36 |
| 842k | 36 |
| 853k | 35 |
| 269k | 35 |
| 704k | 35 |
| 742k | 35 |
| 554k | 35 |
| 173k | 35 |
| 188M | 35 |
| 849k | 35 |
| 184M | 35 |
| 766k | 35 |
| 940k | 35 |
| 354k | 35 |
| 267k | 35 |

| | |
|---|---|
| 577k | 35 |
| 505k | 35 |
| 833k | 35 |
| 383k | 35 |
| 905k | 35 |
| 187M | 35 |
| 901k | 35 |
| 128k | 35 |
| 651k | 35 |
| 511k | 35 |
| 767k | 35 |
| 835k | 35 |
| 445k | 35 |
| 322k | 35 |
| 280k | 35 |
| 525k | 35 |
| 189k | 35 |
| 338k | 35 |
| 956k | 35 |
| 958k | 35 |
| 599k | 35 |
| 789k | 35 |
| 972k | 35 |
| 183k | 34 |
| 364k | 34 |
| 271k | 34 |
| 687k | 34 |
| 167k | 34 |
| 815k | 34 |
| 736k | 34 |
| 251k | 34 |
| 187k | 34 |
| 96k | 34 |
| 509k | 34 |
| 977k | 34 |
| 690k | 34 |
| 371k | 34 |
| 951k | 34 |
| 283k | 34 |
| 838k | 34 |
| 99k | 34 |
| 921k | 34 |
| 1,017k | 34 |
| 761k | 34 |
| 287k | 34 |
| 516k | 34 |
| 300k | 34 |
| 844k | 34 |
| 394k | 34 |

| | |
|---|---|
| 568k | 34 |
| 527k | 34 |
| 620k | 34 |
| 903k | 34 |
| 95k | 34 |
| 485k | 34 |
| 171k | 34 |
| 319k | 34 |
| 777k | 34 |
| 129k | 34 |
| 152k | 34 |
| 396k | 34 |
| 920k | 34 |
| 558k | 34 |
| 707k | 34 |
| 918k | 34 |
| 717k | 34 |
| 995k | 34 |
| 93k | 33 |
| 477k | 33 |
| 928k | 33 |
| 275k | 33 |
| 420k | 33 |
| 878k | 33 |
| 986k | 33 |
| 268k | 33 |
| 733k | 33 |
| 866k | 33 |
| 748k | 33 |
| 854k | 33 |
| 466k | 33 |
| 460k | 33 |
| 819k | 33 |
| 867k | 33 |
| 183M | 33 |
| 618k | 33 |
| 342k | 33 |
| 805k | 33 |
| 646k | 33 |
| 208k | 33 |
| 953k | 33 |
| 641k | 33 |
| 195k | 33 |
| 475k | 33 |
| 441k | 33 |
| 523k | 33 |
| 807k | 33 |
| 315k | 33 |
| 103k | 33 |

| | |
|---|---|
| 897k | 33 |
| 896k | 33 |
| 431k | 33 |
| 737k | 33 |
| 332k | 33 |
| 706k | 33 |
| 276k | 33 |
| 812k | 33 |
| 691k | 33 |
| 440k | 33 |
| 760k | 33 |
| 699k | 33 |
| 712k | 33 |
| 795k | 33 |
| 377k | 32 |
| 575k | 32 |
| 612k | 32 |
| 633k | 32 |
| 828k | 32 |
| 1,002k | 32 |
| 391k | 32 |
| 948k | 32 |
| 772k | 32 |
| 677k | 32 |
| 312k | 32 |
| 202k | 32 |
| 594k | 32 |
| 808k | 32 |
| 474k | 32 |
| 814k | 32 |
| 330k | 32 |
| 543k | 32 |
| 823k | 32 |
| 961k | 32 |
| 811k | 32 |
| 851k | 32 |
| 623k | 32 |
| 652k | 32 |
| 12k | 32 |
| 412k | 32 |
| 741k | 32 |
| 192k | 32 |
| 562k | 32 |
| 775k | 32 |
| 694k | 32 |
| 784k | 32 |
| 487k | 32 |
| 1,015k | 32 |
| 793k | 32 |

| | |
|---|---|
| 785k | 32 |
| 291k | 32 |
| 800k | 32 |
| 439k | 32 |
| 1,003k | 32 |
| 693k | 32 |
| 204k | 32 |
| 672k | 32 |
| 465k | 32 |
| 923k | 32 |
| 531k | 32 |
| 493k | 32 |
| 754k | 32 |
| 729k | 32 |
| 859k | 32 |
| 14k | 31 |
| 176k | 31 |
| 573k | 31 |
| 870k | 31 |
| 139k | 31 |
| 727k | 31 |
| 133k | 31 |
| 957k | 31 |
| 358k | 31 |
| 331k | 31 |
| 787k | 31 |
| 868k | 31 |
| 486k | 31 |
| 908k | 31 |
| 991k | 31 |
| 629k | 31 |
| 137k | 31 |
| 738k | 31 |
| 378k | 31 |
| 621k | 31 |
| 654k | 31 |
| 348k | 31 |
| 163k | 31 |
| 802k | 31 |
| 846k | 31 |
| 159k | 31 |
| 289k | 31 |
| 776k | 31 |
| 298k | 31 |
| 979k | 30 |
| 978k | 30 |
| 724k | 30 |
| 452k | 30 |
| 769k | 30 |

```
491k                    30
834k                    30
869k                    30
688k                    30
463k                    30
817k                    30
725k                    30
92k                     30
343k                    30
988k                    30
667k                    30
718k                    30
538k                    30
419k                    30
529k                    30
132k                    30
333k                    30
301k                    30
666k                    30
503k                    30
196M                    30
180k                    30
945k                    30
542k                    30
305k                    30
428k                    30
469k                    30
108k                    30
306k                    30
310k                    30
734k                    30
711k                    30
726k                    30
497k                    30
643k                    30
892k                    30
744k                    30
198M                    30
750k                    30
84k                     30
749k                    30
829k                    30
388k                    30
522k                    30
302k                    30
566k                    30
316k                    30
887k                    30
804k                    29
```

| | |
|---|---|
| 768k | 29 |
| 714k | 29 |
| 438k | 29 |
| 124k | 29 |
| 546k | 29 |
| 982k | 29 |
| 647k | 29 |
| 703k | 29 |
| 840k | 29 |
| 616k | 29 |
| 747k | 29 |
| 773k | 29 |
| 644k | 29 |
| 683k | 29 |
| 430k | 29 |
| 679k | 29 |
| 745k | 29 |
| 143k | 29 |
| 490k | 29 |
| 347k | 29 |
| 735k | 29 |
| 436k | 29 |
| 293k | 29 |
| 449k | 29 |
| 622k | 29 |
| 675k | 29 |
| 504k | 29 |
| 701k | 29 |
| 155k | 29 |
| 248k | 29 |
| 203k | 29 |
| 852k | 29 |
| 189M | 29 |
| 740k | 29 |
| 494k | 29 |
| 636k | 29 |
| 592k | 29 |
| 518k | 29 |
| 627k | 29 |
| 337k | 29 |
| 272k | 29 |
| 634k | 29 |
| 586k | 29 |
| 179k | 29 |
| 719k | 29 |
| 111k | 29 |
| 649k | 29 |
| 548k | 29 |
| 818k | 28 |

| | |
|---|---|
| 560k | 28 |
| 387k | 28 |
| 457k | 28 |
| 341k | 28 |
| 648k | 28 |
| 423k | 28 |
| 392k | 28 |
| 535k | 28 |
| 613k | 28 |
| 395k | 28 |
| 496k | 28 |
| 794k | 28 |
| 570k | 28 |
| 640k | 28 |
| 292k | 28 |
| 550k | 28 |
| 624k | 28 |
| 545k | 28 |
| 313k | 28 |
| 700k | 28 |
| 619k | 28 |
| 381k | 28 |
| 752k | 28 |
| 638k | 28 |
| 191M | 28 |
| 705k | 28 |
| 674k | 28 |
| 318k | 28 |
| 492k | 28 |
| 299k | 28 |
| 382k | 28 |
| 671k | 28 |
| 682k | 28 |
| 374k | 28 |
| 847k | 28 |
| 579k | 28 |
| 324k | 28 |
| 467k | 28 |
| 79k | 28 |
| 125k | 28 |
| 783k | 27 |
| 71k | 27 |
| 510k | 27 |
| 598k | 27 |
| 479k | 27 |
| 720k | 27 |
| 115k | 27 |
| 481k | 27 |
| 656k | 27 |
| 136k | 27 |

```
610k                    27
426k                    27
400k                    27
368k                    27
375k                    27
356k                    27
207k                    27
574k                    27
455k                    27
422k                    27
506k                    27
500k                    27
175M                    27
346k                    27
411k                    27
473k                    27
320k                    27
195M                    27
673k                    27
192M                    27
454k                    27
697k                    27
635k                    27
351k                    27
628k                    27
366k                    27
860k                    27
681k                    27
408k                    27
645k                    27
848k                    27
401k                    27
626k                    27
602k                    27
585k                    26
710k                    26
845k                    26
578k                    26
484k                    26
534k                    26
715k                    26
659k                    26
822k                    26
11k                     26
402k                    26
197M                    26
140k                    26
295k                    26
186M                    26
```

```
432k                    26
551k                    26
759k                    26
797k                    26
429k                    26
653k                    26
471k                    26
206M                    26
530k                    26
425k                    26
686k                    26
116k                    26
600k                    26
363k                    26
809k                    26
680k                    26
294k                    26
536k                    26
190M                    26
571k                    25
480k                    25
418k                    25
1,010k                  25
355k                    25
695k                    25
403k                    25
483k                    25
595k                    25
949k                    25
582k                    25
424k                    25
507k                    25
639k                    25
340k                    25
663k                    25
615k                    25
771k                    25
670k                    25
260k                    25
971k                    25
288k                    25
517k                    25
121k                    25
803k                    25
450k                    24
533k                    24
567k                    24
781k                    24
563k                    24
```

| | |
|---|---|
| 264k | 24 |
| 398k | 24 |
| 698k | 24 |
| 521k | 24 |
| 539k | 24 |
| 601k | 24 |
| 561k | 24 |
| 404k | 24 |
| 614k | 24 |
| 415k | 24 |
| 658k | 24 |
| 379k | 24 |
| 606k | 24 |
| 120k | 24 |
| 660k | 24 |
| 501k | 24 |
| 416k | 24 |
| 489k | 24 |
| 758k | 24 |
| 112k | 24 |
| 731k | 24 |
| 336k | 24 |
| 580k | 24 |
| 339k | 24 |
| 553k | 24 |
| 661k | 24 |
| 541k | 24 |
| 584k | 24 |
| 664k | 24 |
| 587k | 24 |
| 131k | 23 |
| 384k | 23 |
| 765k | 23 |
| 435k | 23 |
| 476k | 23 |
| 593k | 23 |
| 665k | 23 |
| 544k | 23 |
| 581k | 23 |
| 557k | 23 |
| 458k | 23 |
| 540k | 23 |
| 605k | 23 |
| 335k | 23 |
| 728k | 23 |
| 344k | 23 |
| 519k | 23 |
| 723k | 23 |
| 572k | 23 |

| | |
|---|---|
| 325k | 23 |
| 123k | 23 |
| 307k | 23 |
| 1,013k | 23 |
| 751k | 23 |
| 144k | 23 |
| 427k | 23 |
| 655k | 23 |
| 689k | 22 |
| 135k | 22 |
| 498k | 22 |
| 569k | 22 |
| 304k | 22 |
| 609k | 22 |
| 443k | 22 |
| 461k | 22 |
| 669k | 22 |
| 607k | 22 |
| 537k | 22 |
| 617k | 22 |
| 604k | 22 |
| 909k | 22 |
| 792k | 22 |
| 591k | 22 |
| 596k | 22 |
| 376k | 22 |
| 459k | 22 |
| 547k | 22 |
| 311k | 21 |
| 512k | 21 |
| 502k | 21 |
| 532k | 21 |
| 520k | 21 |
| 515k | 21 |
| 203M | 21 |
| 608k | 21 |
| 442k | 21 |
| 528k | 21 |
| 472k | 21 |
| 603k | 21 |
| 801k | 21 |
| 549k | 21 |
| 499k | 21 |
| 470k | 21 |
| 207M | 21 |
| 104k | 21 |
| 508k | 20 |
| 555k | 20 |
| 185M | 20 |

| | |
|---|---|
| 488k | 20 |
| 590k | 20 |
| 444k | 20 |
| 637k | 20 |
| 732k | 20 |
| 583k | 20 |
| 650k | 20 |
| 468k | 20 |
| 447k | 20 |
| 662k | 20 |
| 565k | 20 |
| 451k | 19 |
| 630k | 19 |
| 556k | 19 |
| 380k | 19 |
| 352k | 19 |
| 119k | 19 |
| 482k | 19 |
| 559k | 19 |
| 611k | 19 |
| 764k | 19 |
| 684k | 19 |
| 564k | 19 |
| 576k | 19 |
| 372k | 18 |
| 235M | 18 |
| 524k | 18 |
| 456k | 18 |
| 513k | 18 |
| 709k | 18 |
| 216M | 18 |
| 215M | 18 |
| 464k | 18 |
| 632k | 17 |
| 657k | 17 |
| 205M | 17 |
| 345k | 17 |
| 1,024k | 17 |
| 199M | 17 |
| 218M | 17 |
| 296k | 17 |
| 204M | 17 |
| 668k | 17 |
| 200M | 17 |
| 349k | 17 |
| 10k | 16 |
| 211M | 15 |
| 214M | 15 |
| 327k | 15 |

| | |
|---|---|
| 213M | 15 |
| 233M | 14 |
| 756k | 14 |
| 230M | 13 |
| 255M | 13 |
| 227M | 12 |
| 228M | 12 |
| 210M | 12 |
| 242M | 12 |
| 250M | 11 |
| 226M | 11 |
| 256M | 11 |
| 208M | 11 |
| 231M | 11 |
| 223M | 10 |
| 220M | 10 |
| 243M | 10 |
| 237M | 10 |
| 224M | 10 |
| 219M | 9 |
| 217M | 9 |
| 232M | 9 |
| 238M | 9 |
| 287M | 9 |
| 234M | 9 |
| 241M | 9 |
| 244M | 9 |
| 257M | 8 |
| 249M | 8 |
| 280M | 8 |
| 1.1G | 8 |
| 248M | 8 |
| 565M | 8 |
| 222M | 8 |
| 221M | 8 |
| 270M | 8 |
| 225M | 7 |
| 317M | 7 |
| 246M | 7 |
| 264M | 7 |
| 253M | 7 |
| 261M | 7 |
| 245M | 7 |
| 276M | 7 |
| 332M | 6 |
| 319M | 6 |
| 236M | 6 |
| 240M | 6 |
| 321M | 6 |

| | |
|---|---|
| 247M | 6 |
| 229M | 6 |
| 299M | 6 |
| 262M | 6 |
| 252M | 5 |
| 251M | 5 |
| 281M | 5 |
| 298M | 5 |
| 309M | 5 |
| 239M | 5 |
| 8.4k | 5 |
| 6.8k | 5 |
| 297M | 5 |
| 6.4k | 5 |
| 263M | 5 |
| 258M | 5 |
| 9.6k | 4 |
| 339M | 4 |
| 310M | 4 |
| 306M | 4 |
| 355M | 4 |
| 9.1k | 4 |
| 260M | 4 |
| 269M | 4 |
| 8.9k | 4 |
| 300M | 4 |
| 304M | 4 |
| 567M | 4 |
| 335M | 4 |
| 293M | 4 |
| 267M | 4 |
| 338M | 4 |
| 322M | 4 |
| 285M | 4 |
| 327M | 4 |
| 274M | 4 |
| 330M | 4 |
| 303M | 4 |
| 289M | 4 |
| 277M | 4 |
| 266M | 3 |
| 448M | 3 |
| 391M | 3 |
| 390M | 3 |
| 9.8k | 3 |
| 9.7k | 3 |
| 9.9k | 3 |
| 272M | 3 |
| 1.0G | 3 |

| | |
|---|---|
| 445M | 3 |
| 344M | 3 |
| 286M | 3 |
| 331M | 3 |
| 353M | 3 |
| 323M | 3 |
| 315M | 3 |
| 283M | 3 |
| 8.5k | 3 |
| 375M | 3 |
| 291M | 3 |
| 382M | 3 |
| 422M | 3 |
| 265M | 3 |
| 275M | 3 |
| 290M | 3 |
| 360M | 3 |
| 313M | 3 |
| 4.7k | 3 |
| 268M | 3 |
| 314M | 3 |
| 254M | 3 |
| 278M | 2 |
| 359M | 2 |
| 6.1k | 2 |
| 398M | 2 |
| 656M | 2 |
| 329M | 2 |
| 7.8k | 2 |
| 279M | 2 |
| 354M | 2 |
| 454M | 2 |
| 564M | 2 |
| 284M | 2 |
| 292M | 2 |
| 408M | 2 |
| 6.3k | 2 |
| 508M | 2 |
| 843M | 2 |
| 343M | 2 |
| 460M | 2 |
| 333M | 2 |
| 618M | 2 |
| 311M | 2 |
| 9.5k | 2 |
| 9.3k | 2 |
| 337M | 2 |
| 372M | 2 |
| 340M | 2 |

```
7.1k                    2
288M                    2
259M                    2
745M                    2
1.5G                    2
405M                    2
9.0k                    2
9.2k                    2
377M                    2
387M                    2
295M                    2
348M                    2
294M                    2
296M                    2
324M                    2
273M                    2
352M                    2
510M                    2
305M                    2
409M                    2
467M                    2
406M                    2
566M                    2
366M                    2
334M                    2
373M                    1
7.7k                    1
705M                    1
404M                    1
896M                    1
570M                    1
810M                    1
415M                    1
369M                    1
396M                    1
465M                    1
302M                    1
397M                    1
468M                    1
646M                    1
839M                    1
869M                    1
769M                    1
590M                    1
301M                    1
488M                    1
692M                    1
611M                    1
593M                    1
```

```
527M                    1
424M                    1
503M                    1
866M                    1
440M                    1
461M                    1
365M                    1
431M                    1
349M                    1
581M                    1
962M                    1
470M                    1
541M                    1
799M                    1
633M                    1
5.1k                    1
910M                    1
679M                    1
342M                    1
370M                    1
3.2k                    1
532M                    1
725M                    1
691M                    1
442M                    1
737M                    1
914M                    1
394M                    1
700M                    1
356M                    1
712M                    1
5.3k                    1
568M                    1
830M                    1
519M                    1
8.3k                    1
580M                    1
977M                    1
959M                    1
981M                    1
526M                    1
379M                    1
623M                    1
533M                    1
619M                    1
521M                    1
429M                    1
744M                    1
844M                    1
```

| | |
|---|---|
| 889M | 1 |
| 720M | 1 |
| 6.2k | 1 |
| 996M | 1 |
| 371M | 1 |
| 282M | 1 |
| 437M | 1 |
| 447M | 1 |
| 497M | 1 |
| 643M | 1 |
| 456M | 1 |
| 493M | 1 |
| 414M | 1 |
| 925M | 1 |
| 645M | 1 |
| 652M | 1 |
| 420M | 1 |
| 383M | 1 |
| 312M | 1 |
| 919M | 1 |
| 904M | 1 |
| 595M | 1 |
| 485M | 1 |
| 900M | 1 |
| 361M | 1 |
| 345M | 1 |
| 7.2k | 1 |
| 895M | 1 |
| 954M | 1 |
| 490M | 1 |
| 3.4k | 1 |
| 603M | 1 |
| 1,006M | 1 |
| 706M | 1 |
| 509M | 1 |
| 928M | 1 |
| 320M | 1 |
| 3.3k | 1 |
| 4.6k | 1 |
| 531M | 1 |
| 364M | 1 |
| 722M | 1 |
| 501M | 1 |
| 985M | 1 |
| 412M | 1 |
| 1,020M | 1 |
| 841M | 1 |
| 576M | 1 |
| 690M | 1 |
| 832M | 1 |

| | |
|---|---|
| 10.0k | 1 |
| 328M | 1 |
| 818M | 1 |
| 953M | 1 |
| 6.9k | 1 |
| 544M | 1 |
| 426M | 1 |
| 681M | 1 |
| 347M | 1 |
| 5.8k | 1 |
| 606M | 1 |
| 940M | 1 |
| 271M | 1 |
| 765M | 1 |
| 664M | 1 |
| 649M | 1 |
| 411M | 1 |
| 680M | 1 |
| 514M | 1 |
| 639M | 1 |
| 963M | 1 |
| 421M | 1 |
| 457M | 1 |
| 381M | 1 |
| 318M | 1 |
| 878M | 1 |
| 351M | 1 |
| 578M | 1 |
| 868M | 1 |
| 495M | 1 |
| 935M | 1 |
| 363M | 1 |
| 518M | 1 |
| 762M | 1 |
| 491M | 1 |
| 676M | 1 |
| 684M | 1 |
| 579M | 1 |
| 550M | 1 |
| 407M | 1 |
| 591M | 1 |
| 418M | 1 |
| 7.4k | 1 |
| 346M | 1 |
| 8.7k | 1 |
| 784M | 1 |
| 385M | 1 |
| 7.6k | 1 |

```
512M                        1
Name: count, dtype: int64
```

```
# check unique values
df['Size'].unique()
```

```
array(['10M', '2.9M', '3.7M', ..., '405M', '3.2k', '512M'],
dtype=object)
```

```
df['Size'].loc[df['Size'].str.contains('M',)].value_counts().sum()
```

```
2201901
```

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

```
36253
```

```
# 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()
```

```
74777
```

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

```
2312944
```

- We have `2201901` values in `M` units
- We have `36253` values in `k` units
- We have `74777` 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.

```
print(df.columns)
```

```
Index(['App Name', 'App Id', 'Category', 'Rating', 'Rating Count',
'Installs',
       'Minimum Installs', 'Maximum Installs', 'Free', 'Price',
'Currency',
       'Size', 'Minimum Android', 'Developer Id', 'Developer Website',
       'Developer Email', 'Released', 'Last Updated', 'Content
Rating',
       'Privacy Policy', 'Ad Supported', 'In App Purchases', 'Editors
Choice',
       'Scraped Time', 'Installs_category'],
      dtype='object')
```

```
def convert_size(size):
    if isinstance(size, str):
```

```
        # Remove commas
        size = size.replace(',', '')
        if 'k' in size:
            return float(size.replace('k', '')) * 1024
        elif 'M' in size:
            return float(size.replace('M', '')) * 1024 * 1024
        elif 'G' in size:
            return float(size.replace('G', '')) * 1024 * 1024 * 1024
        else:
            try:
                return float(size)
            except ValueError:
                return np.nan
    return size

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

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

 # Convert to numeric, coercing errors to NaN (in case of invalid
values)
df['Size_in_bytes'] = pd.to_numeric(df['Size_in_bytes'],
errors='coerce')

# making a new column called 'Size in Mb' which will have the size in
MB
df['Size_in_Mb'] = df['Size_in_bytes'].apply(lambda x: x/(1024*1024))
```

- Now we have converted every value into bytes and removed the `M` and `K` from the values and converted them into numeric data type.
- 'Varies with device' was a string value, therefore we intentionally converted them into null values, which we can fill later on according to our needs.

---

- Let's have a look on the `Minimum Android` column

```
df['Minimum Android'].isnull().sum()

6530

df['Minimum Android'] = df['Minimum Android'].fillna('')

# Replace empty strings with '0'
df['Installs'] = df['Installs'].replace('', '0')

df['Minimum Android'].isnull().sum()

0
```

- No null values, we are good to go.

```
df['Minimum Android'].unique()
```

```
array(['7.1 and up', '5.0 and up', '4.0.3 and up', '4.1 and up',
       '6.0 and up', '4.4 and up', '4.0 and up', '4.2 and up',
       '2.1 and up', '7.0 and up', '2.2 and up', '2.3 and up',
       '4.3 and up', '5.1 and up', '', '1.6 and up', '3.0 and up',
       'Varies with device', '8.0 and up', '2.3.3 and up', '4.4W and
up',
       '3.2 and up', '1.5 and up', '4.0.3 - 7.1.1', '3.1 and up',
       '2.0 and up', '1.0 and up', '1.1 and up', '2.0.1 and up',
       '4.4 - 6.0', '2.1 - 4.4', '4.0 - 5.0', '2.0 - 2.3.4', '2.2 -
4.4',
       '4.1 - 7.0', '4.1 - 6.0', '4.0 - 4.4W', '4.0 - 4.4', '4.0.3 -
7.0',
       '2.3 - 5.0', '4.0 - 7.1.1', '4.1 - 7.1.1', '4.0 - 7.0',
       '4.1 - 8.0', '4.0 - 6.0', '2.3 - 7.0', '4.0 - 8.0', '2.3 -
6.0',
       '2.1 - 2.3.4', '3.0 - 4.1.1', '2.2 - 3.0', '2.0 - 8.0',
       '4.0.3 - 8.0', '3.0 - 4.4W', '4.4 - 7.1.1', '4.4 - 8.0',
       '4.0.3 - 6.0', '1.6 - 4.0.4', '8.0', '2.3 - 5.1', '2.1 -
7.1.1',
       '2.3 - 4.4W', '4.4 - 7.0', '6.0 - 7.1.1', '2.3 - 4.4',
       '2.2 - 4.0.4', '2.1 - 4.1.1', '3.0 - 8.0', '3.0 - 5.1',
       '2.1 - 6.0', '6.0 - 8.0', '5.0 - 8.0', '4.1 - 5.1', '3.2 -
4.4',
       '2.1 - 5.0', '4.1 - 4.3', '5.0 - 6.0', '5.1 - 7.1.1', '2.2 -
5.1',
       '4.1 - 5.0', '2.3.3 - 4.4', '2.2 - 5.0', '2.2', '4.1 - 4.4',
       '1.6 - 5.1', '4.2 - 7.1.1', '2.1 - 5.1', '1.6 - 4.4W',
       '4.3 - 4.4W', '3.0 - 4.4', '2.1 - 3.1', '2.3 - 7.1.1',
       '1.6 - 4.0.2', '4.0 - 4.0.4', '4.2 - 4.4W', '3.0 - 7.1.1',
       '3.2 - 7.1.1', '3.0 - 6.0', '2.2 - 4.4W', '4.0.3 - 4.4',
       '4.0.3 - 5.1', '6.0', '4.3', '1.1 - 4.4', '2.3.3 - 8.0',
       '2.3.3 - 5.0', '2.2 - 6.0', '3.0 - 5.0', '2.3.3 - 2.3.4',
       '1.5 - 2.1', '2.2 - 3.2', '5.0 - 7.1.1', '2.1 - 3.2',
       '4.2 - 4.2.2', '2.3.3 - 5.1', '2.2 - 4.1.1', '1.6 - 4.4',
       '4.2 - 4.3', '4.0 - 5.1', '1.6 - 7.0', '3.0 - 7.0', '1.0 -
6.0',
       '2.3.3 - 4.0.4', '2.3 - 3.2', '4.3 - 8.0', '2.3 - 8.0',
       '3.2 - 6.0', '2.3.3 - 7.1.1', '4.2 - 8.0', '2.2 - 4.3',
       '3.2 - 4.1.1', '4.4 - 5.1', '2.3 - 4.1.1', '2.1 - 4.4W', '4.4',
       '2.3.3 - 4.4W', '4.1 - 4.4W', '1.6 - 7.1.1', '2.2 - 4.2.2',
       '4.3 - 5.1', '4.0.3 - 4.2.2', '2.2 - 2.3.4', '4.3 - 4.4',
       '4.2 - 6.0', '4.4 - 4.4W', '2.3 - 4.0.2', '3.2 - 5.1', '7.0',
       '3.0 - 3.2', '1.6 - 4.2.2', '5.1', '1.5 - 3.2', '2.2 - 8.0',
       '1.6 - 2.1', '2.3.3 - 6.0'], dtype=object)
```

```
df['Minimum Android'].value_counts()
```

| Minimum Android | |
|---|---|
| 4.1 and up | 604465 |
| 5.0 and up | 396998 |
| 4.4 and up | 390311 |
| 4.0.3 and up | 180482 |
| 4.0 and up | 153441 |
| 4.2 and up | 115973 |
| 6.0 and up | 89928 |
| 2.3 and up | 65577 |
| 5.1 and up | 59287 |
| Varies with device | 46214 |
| 4.3 and up | 41357 |
| 7.0 and up | 34444 |
| 2.2 and up | 23720 |
| 2.3.3 and up | 21549 |
| 3.0 and up | 17105 |
| 2.1 and up | 16699 |
| 8.0 and up | 13827 |
| 4.4W and up | 12527 |
| 1.6 and up | 8537 |
| | 6530 |
| 7.1 and up | 3044 |
| 2.0 and up | 2774 |
| 3.2 and up | 2673 |
| 1.5 and up | 2097 |
| 3.1 and up | 2003 |
| 2.0.1 and up | 443 |
| 1.0 and up | 308 |
| 1.1 and up | 165 |
| 4.0 - 6.0 | 29 |
| 2.3 - 6.0 | 16 |
| 4.1 - 8.0 | 16 |
| 4.1 - 6.0 | 16 |
| 4.0.3 - 6.0 | 15 |
| 4.0 - 7.1.1 | 14 |
| 2.2 - 4.4 | 14 |
| 4.1 - 7.1.1 | 13 |
| 4.0.3 - 8.0 | 12 |
| 4.0 - 8.0 | 11 |
| 4.0 - 4.4 | 9 |
| 4.4 - 8.0 | 9 |
| 2.3 - 5.1 | 9 |
| 3.0 - 4.4 | 8 |
| 2.3 - 7.1.1 | 8 |
| 2.3 - 5.0 | 7 |
| 4.1 - 7.0 | 7 |
| 4.1 - 5.1 | 7 |
| 2.3.3 - 4.4 | 6 |
| 4.1 - 4.4 | 6 |
| 4.0 - 5.1 | 6 |

| | |
|---|---|
| 2.2 - 6.0 | 6 |
| 4.0 - 7.0 | 5 |
| 2.2 - 5.0 | 5 |
| 4.0 - 4.4W | 5 |
| 5.0 - 6.0 | 5 |
| 2.3 - 4.4 | 5 |
| 4.4 - 7.1.1 | 5 |
| 3.0 - 5.1 | 5 |
| 5.0 - 8.0 | 5 |
| 3.0 - 5.0 | 4 |
| 3.2 - 4.4 | 4 |
| 2.2 - 4.1.1 | 4 |
| 2.2 - 4.0.4 | 4 |
| 2.1 - 2.3.4 | 4 |
| 4.0.3 - 7.0 | 4 |
| 8.0 | 4 |
| 4.0.3 - 7.1.1 | 4 |
| 4.4 - 6.0 | 4 |
| 4.0 - 5.0 | 4 |
| 3.0 - 4.1.1 | 4 |
| 2.2 - 3.2 | 3 |
| 4.0.3 - 4.4 | 3 |
| 4.0 - 4.0.4 | 3 |
| 4.0.3 - 5.1 | 3 |
| 1.6 - 4.4W | 3 |
| 2.1 - 4.4 | 3 |
| 4.2 - 7.1.1 | 3 |
| 2.0 - 2.3.4 | 3 |
| 2.2 - 4.4W | 3 |
| 3.0 - 6.0 | 3 |
| 4.2 - 8.0 | 3 |
| 2.1 - 5.0 | 3 |
| 2.1 - 6.0 | 3 |
| 2.3.3 - 5.1 | 2 |
| 2.3.3 - 5.0 | 2 |
| 2.3.3 - 7.1.1 | 2 |
| 2.3.3 - 4.0.4 | 2 |
| 6.0 | 2 |
| 1.6 - 4.4 | 2 |
| 2.2 - 4.3 | 2 |
| 3.2 - 6.0 | 2 |
| 4.4 | 2 |
| 2.2 - 4.2.2 | 2 |
| 3.2 - 5.1 | 2 |
| 1.5 - 2.1 | 2 |
| 7.0 | 2 |
| 4.3 - 8.0 | 2 |
| 5.1 - 7.1.1 | 2 |
| 3.2 - 7.1.1 | 2 |
| 3.0 - 4.4W | 2 |

| | |
|---|---|
| 2.2 - 5.1 | 2 |
| 2.2 - 3.0 | 2 |
| 2.3 - 7.0 | 2 |
| 6.0 - 8.0 | 2 |
| 2.1 - 4.1.1 | 2 |
| 2.3 - 4.4W | 2 |
| 1.6 - 4.0.2 | 2 |
| 4.2 - 4.4W | 2 |
| 4.1 - 4.3 | 2 |
| 2.1 - 4.4W | 1 |
| 2.3.3 - 4.4W | 1 |
| 2.3 - 4.1.1 | 1 |
| 4.1 - 4.4W | 1 |
| 1.6 - 7.1.1 | 1 |
| 2.0 - 8.0 | 1 |
| 4.3 - 5.1 | 1 |
| 4.0.3 - 4.2.2 | 1 |
| 2.2 - 2.3.4 | 1 |
| 3.2 - 4.1.1 | 1 |
| 4.3 - 4.4 | 1 |
| 4.2 - 6.0 | 1 |
| 4.4 - 4.4W | 1 |
| 2.3 - 4.0.2 | 1 |
| 3.0 - 3.2 | 1 |
| 1.6 - 4.2.2 | 1 |
| 5.1 | 1 |
| 1.5 - 3.2 | 1 |
| 2.2 - 8.0 | 1 |
| 1.6 - 2.1 | 1 |
| 4.4 - 5.1 | 1 |
| 4.4 - 7.0 | 1 |
| 1.6 - 4.0.4 | 1 |
| 5.0 - 7.1.1 | 1 |
| 2.1 - 3.1 | 1 |
| 4.3 | 1 |
| 1.1 - 4.4 | 1 |
| 2.3.3 - 8.0 | 1 |
| 4.3 - 4.4W | 1 |
| 2.1 - 5.1 | 1 |
| 1.6 - 5.1 | 1 |
| 2.3.3 - 2.3.4 | 1 |
| 2.2 | 1 |
| 4.1 - 5.0 | 1 |
| 2.1 - 3.2 | 1 |
| 2.1 - 7.1.1 | 1 |
| 4.2 - 4.2.2 | 1 |
| 4.2 - 4.3 | 1 |
| 3.0 - 8.0 | 1 |
| 1.6 - 7.0 | 1 |
| 3.0 - 7.0 | 1 |

```
1.0 - 6.0                   1
2.3 - 3.2                   1
6.0 - 7.1.1                 1
2.3 - 8.0                   1
3.0 - 7.1.1                 1
2.3.3 - 6.0                 1
Name: count, dtype: int64
```

```
df['Rating Count'].value_counts()
```

```
Output hidden; open in https://colab.research.google.com to view.
```

```
df.describe()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8,\n  \"fields\": [\n{\n      \"column\": \"Rating\",\n      \"properties\": {\n\"dtype\": \"number\",\n        \"std\": 809657.997368076,\n\"min\": 0.0,\n        \"max\": 2290061.0,\n\"num_unique_values\": 7,\n        \"samples\": [\n2290061.0,\n          2.203151531771424,\n          4.3\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Rating Count\",\n\"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 48867391.570589274,\n        \"min\": 0.0,\n        \"max\": 138557570.0,\n        \"num_unique_values\": 7,\n        \"samples\": [\n          2290061.0,\n          2864.8388767810115,\n42.0\n        ],\n        \"semantic_type\": \"\",\n\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Installs\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 3534647141.7795744,\n        \"min\": 0.0,\n        \"max\": 10000000000.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n          183436.7275277741,\n500.0,\n          2312944.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n\"column\": \"Minimum Installs\",\n      \"properties\": {\n\"dtype\": \"number\",\n        \"std\": 3534647129.2584343,\n\"min\": 0.0,\n        \"max\": 10000000000.0,\n\"num_unique_values\": 8,\n        \"samples\": [\n183445.21395800915,\n          500.0,\n          2312837.0\n        ],\n        \"semantic_type\": \"\",\n\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Maximum Installs\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 4261699552.060366,\n        \"min\": 0.0,\n        \"max\": 12057627016.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n          320201.713137456,\n695.0,\n          2312944.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n\"column\": \"Price\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 817728.8641336214,\n        \"min\": 0.0,\n        \"max\": 2312944.0,\n        \"num_unique_values\": 5,\n
```

```
\"samples\": [\n          0.10349915833370804,\n          400.0,\n
2.63312655760054\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Size_in_bytes\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 565012547.3049533,\n        \"min\":
0.0,\n        \"max\": 1610612736.0,\n        \"num_unique_values\":
8,\n        \"samples\": [\n          20139583.164281834,\n
10485760.0,\n          2238167.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Size_in_Mb\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
791229.9312236877,\n        \"min\": 0.0,\n        \"max\":
2238167.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\
n          19.2066032069033,\n          10.0,\n          2238167.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


## ## Observations:

- Now, we have only 8 columns as numeric data type.
- We can observe their descriptive statistics. and make tons of observations as per our hypotheses.
- We can see that the `Rating` column has a minimum value of `1` and a maximum value of `5`, which is the range of rating, and the mean is `2.20` which is not a good rating. On an average people give this rating.
- We can see that the `Rating Count` column has a minimum value of `1` and a maximum value of `64288`
- Similarly, we can observe the other columns as well.

Therefore, the most important thing is to classify as app based on the correlation matrix and then observe the descriptive statistics of the app category and number of installs, reviews, ratings, etc.

## But even before that we have to think about the missing values in the dataset.

## 2.2. Dealing with the missing values

Dealing with the missing values is one of the most important part of the data wrangling process, we must deal with the missing values in order to get the correct insights from the data.

- Let's have a look on the missing values in the dataset

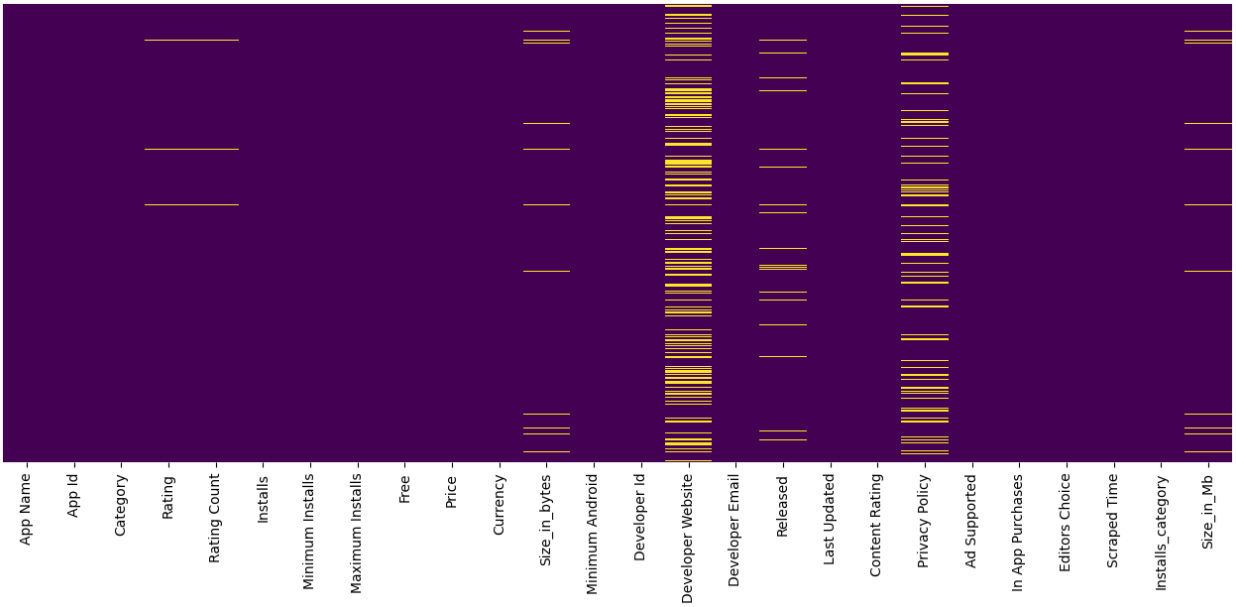
```
df.isnull().sum().sum() # this will show the total number of null
values in the dataframe
```

```
1448472
```

- Let's plot the missing values in the dataset

```
# make a figure size
plt.figure(figsize=(16, 6))
#plot the null values in each column
sns.heatmap(df.isnull(), yticklabels=False, cbar=False,
cmap='viridis') # this will show the heatmap of null values in the
dataframe
```

```
<Axes: >
```




```
df.isnull().sum().sort_values(ascending=False) # this will show the
number of null values in each column in descending order
```

```
Developer Website    760835
Privacy Policy       420953
Size_in_Mb            74777
Size_in_bytes         74777
Released              71053
Rating                22883
Rating Count          22883
Currency                135
Minimum Installs        107
Developer Id             33
Developer Email          31
App Name                  5
Price                     0
Free                      0
Minimum Android           0
App Id                    0
Maximum Installs          0
```

```
Installs                    0
Last Updated                0
Content Rating              0
Category                    0
Ad Supported                0
In App Purchases            0
Editors Choice              0
Scraped Time                0
Installs_category           0
dtype: int64
```

```
(df.isnull().sum() / len(df) * 100).sort_values(ascending=False) # this will show the percentage of null values in each column
```

```
Developer Website    32.894657
Privacy Policy       18.199879
Size_in_Mb            3.232979
Size_in_bytes         3.232979
Released              3.071972
Rating                0.989345
Rating Count          0.989345
Currency              0.005837
Minimum Installs      0.004626
Developer Id          0.001427
Developer Email       0.001340
App Name              0.000216
Price                 0.000000
Free                  0.000000
Minimum Android       0.000000
App Id                0.000000
Maximum Installs      0.000000
Installs              0.000000
Last Updated          0.000000
Content Rating        0.000000
Category              0.000000
Ad Supported          0.000000
In App Purchases      0.000000
Editors Choice        0.000000
Scraped Time          0.000000
Installs_category     0.000000
dtype: float64
```

## ## Observations:

- We have 760835 missing values in the `'Developer Website'` column, which is 32.89% of the total values in the column.
- We have 420953 missing values in the `'Privacy Policy '` column, which is 18.19% of the total values in the column.
- We have 74777 missing values in the `'Size_in_bytes'` and `'Size_in_Mb'` columns, which is 3.23% of the total values in the column.

- We have 71053 missing values in the `'Released'` column, which is 3.07% of the total values in the column.
- We have 22883 missing value in the `'Rating'` column, which is 0.98% of the total values in the column.
- We have 22883 missing values in the `'Rating Count'` column, which is 0.98% of the total values in the column.
- We have 135 missing value in `Currency` columns, which is 0.005837% of the total values in the column.
- We have 107 missing value in `Minimum Installs` columns, which is 0.001427% of the total values in the column.
- We have 33 missing value in `Developer Id` columns, which is 0.005837% of the total values in the column.
- We have 31 missing value in `Developer Email` columns, which is 0.001340% of the total values in the column.
- We have 5 missing value in `App Name` columns, which is 0.000216% of the total values in the column.

## 2.3. Dealing with the missing values

- We can not impute the `Rating` column as is is directly linked with the installation column. To test this Hypothesis we need to plot the `Rating` column with the `Installs` and `Size` columns and statistically test it using `pearson correlation test`.
- Let's run the correlations

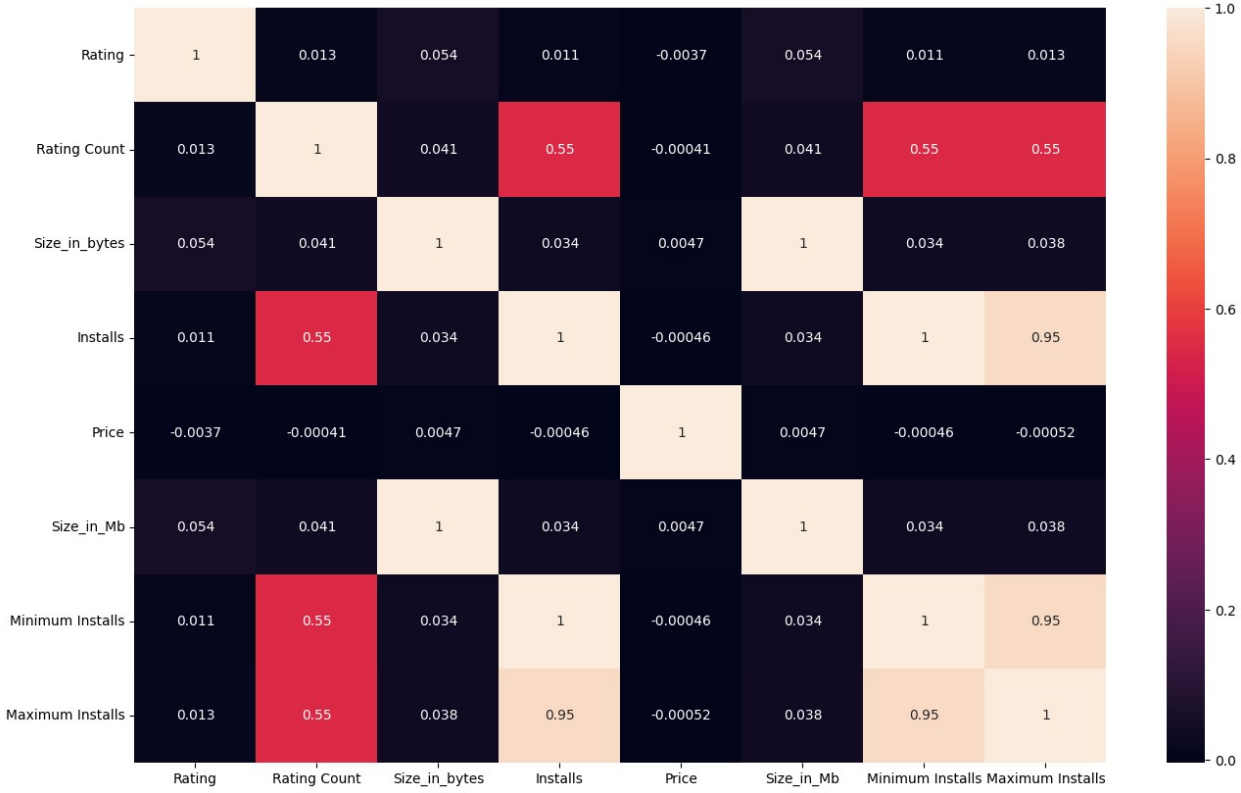
```
df.describe() # these are numeric columns
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8,\n  \"fields\": [\n{\n      \"column\": \"Rating\",\n      \"properties\": {\n\"dtype\": \"number\",\n        \"std\": 809657.997368076,\n\"min\": 0.0,\n        \"max\": 2290061.0,\n\"num_unique_values\": 7,\n        \"samples\": [\n2290061.0,\n          2.203151531771424,\n          4.3\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Rating Count\",\n\"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 48867391.570589274,\n        \"min\": 0.0,\n        \"max\": 138557570.0,\n        \"num_unique_values\": 7,\n        \"samples\": [\n          2290061.0,\n          2864.8388767810115,\n42.0\n        ],\n        \"semantic_type\": \"\",\n\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Installs\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 3534647141.7795744,\n        \"min\": 0.0,\n        \"max\": 10000000000.0,\n        \"num_unique_values\":
```

```
8,\n        \"samples\": [\n          183436.7275277741,\n
500.0,\n          2312944.0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Minimum Installs\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 3534647129.2584343,\n
\"min\": 0.0,\n        \"max\": 10000000000.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
183445.21395800915,\n          500.0,\n          2312837.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Maximum Installs\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 4261699552.060366,\n        \"min\":
0.0,\n        \"max\": 12057627016.0,\n        \"num_unique_values\":
8,\n        \"samples\": [\n          320201.713137456,\n
695.0,\n          2312944.0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Price\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 817728.8641336214,\n        \"min\":
0.0,\n        \"max\": 2312944.0,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          0.10349915833370804,\n          400.0,\n
2.63312655760054\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Size_in_bytes\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 565012547.3049533,\n        \"min\":
0.0,\n        \"max\": 1610612736.0,\n        \"num_unique_values\":
8,\n        \"samples\": [\n          20139583.164281834,\n
10485760.0,\n          2238167.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Size_in_Mb\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
791229.9312236877,\n        \"min\": 0.0,\n        \"max\":
2238167.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\
n          19.2066032069033,\n          10.0,\n          2238167.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


```
# Make a correlation matrix of numeric columns
plt.figure(figsize=(16, 10)) # make figure size
numeric_cols = ['Rating', 'Rating Count', 'Size_in_bytes', 'Installs',
'Price', 'Size_in_Mb','Minimum Installs','Maximum Installs'] # make a
list of numeric columns
sns.heatmap(df[numeric_cols].corr(), annot=True) # plot the
correlation matrix
```

```
<Axes: >
```




```
# we can also calculate the correlation matrix using pandas
df[numeric_cols].corr() # this will show the correlation matrix
```


```
{"summary":"{\n  \"name\": \"df[numeric_cols]\",\n  \"rows\": 8,\n
\"fields\": [\n    {\n      \"column\": \"Rating\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.3464798601358728,\n        \"min\": -0.003674370783803862,\n
\"max\": 1.0,\n        \"num_unique_values\": 6,\n        \"samples\":
[\n          1.0,\n          0.013038113065837074,\n
0.012614648261800471\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Rating Count\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.3711545810277704,\n        \"min\": -
0.0004105471388834398,\n        \"max\": 1.0,\n
\"num_unique_values\": 6,\n        \"samples\": [\n
0.013038113065837074,\n          1.0,\n          0.5475710514761793\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Size_in_bytes\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.4472251743637262,\n        \"min\": 0.004664866188838955,\n
\"max\": 1.0,\n        \"num_unique_values\": 7,\n        \"samples\":
[\n          0.054230015380920225,\n          0.04120048276374,\n
0.03403319026372828\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Installs\",\n      \"properties\": {\n        \"dtype\":
```

```
\"number\",\n        \"std\": 0.4795212664177267,\n        \"min\": -
0.0004606641879899551,\n        \"max\": 1.0,\n
\"num_unique_values\": 6,\n        \"samples\": [\n
0.011214455140461793,\n          0.5452806023392083,\n
0.9540374291309002\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Price\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.35337231735580993,\n        \"min\": -
0.003674370783803862,\n        \"max\": 1.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          -
0.003674370783803862,\n          -0.0004105471388834398,\n          -
0.0004606862515294089\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Size_in_Mb\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.4472251743637262,\n        \"min\":
0.004664866188838955,\n        \"max\": 1.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n
0.054230015380920225,\n          0.04120048276374,\n
0.03403319026372828\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Minimum Installs\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.47952010904842074,\n        \"min\": -
0.0004606862515294089,\n        \"max\": 1.0,\n
\"num_unique_values\": 6,\n        \"samples\": [\n
0.011214455140461793,\n          0.5452806023392083,\n
0.9540374353243399\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Maximum Installs\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.47111374678390333,\n        \"min\": -
0.0005150420525738935,\n        \"max\": 1.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n
0.012614648261800471,\n          0.5475710514761793,\n
0.9540374353243399\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


---

## Observations

- Lighter color shows the high correlation and darker color shows the low correlation
- We can see that the `Rating Count` column has a high correlation with the `Installs` column, which is `0.54` according to corr(). Which is quite good. This shows that the more the Rating Count the more the installs are for one app. If in any case we need to impute Rating Count we have to think of number of install.
- We can see that the `Rating Count` column has a high correlation with the `Minimum Installs` which is `1.00` according to corr(). Which is very good.This shows that the more the Minimum Installs Count the more the installs are for one app. If in any case we need to impute Rating Count we have to think of number of install.

- We can see that the `Rating Count` column has a high correlation with the `Maximum Installs` which is `0.94` according to corr(). Which is very good.This shows that the more the Maximum Installs Count the more the installs are for one app. If in any case we need to impute Rating Count we have to think of number of install.
- Before going ahead, let's remove the rows with missing values in the `Size_in_Mb`, `Size_in_bytes`, `Released`, `Rating` and `Rating Count `Currency`, `Minimum Installs`,`Developer Id`,`Developer Email`,`App Name` columns, as they are very less in number and will not affect our analysis.

```
# length before removing null values
print(f"Length of the dataframe before removing null values:
{len(df)}")
```

```
Length of the dataframe before removing null values: 2312944
```

```
# remove the rows having null values in the 'Current Ver', 'Android
Ver', 'Category', 'Type' and 'Genres' column
df.dropna(subset=['Size_in_Mb', 'Size_in_bytes',
'Released','Rating','Rating Count','Currency','Minimum
Installs','Developer Id','Developer Email','App Name'], inplace=True)

# length after removing null values
print(f"Length of the dataframe after removing null values:
{len(df)}")
```

```
Length of the dataframe after removing null values: 2190366
```

- We have removed `122,578` rows having null values in the `'Size_in_Mb'`, `'Size_in_bytes'`, `'Released'`,`'Rating'`,`'Rating Count'`,`'Currency'`,`'Minimum Installs'`,`'Developer Id'`,`'Developer Email'`,`'App Name'` columns.

```
df['Developer Website'].fillna(df['Developer Website'].mode().iloc[0],
inplace=True)
df['Privacy Policy'].fillna(df['Privacy Policy'].mode().iloc[0],
inplace=True)

# let's check the null values again
df.isnull().sum().sort_values(ascending=False)
```

```
App Name              0
App Id                0
Installs_category     0
Scraped Time          0
Editors Choice        0
In App Purchases      0
Ad Supported          0
Privacy Policy        0
Content Rating        0
Last Updated          0
```

```
Released              0
Developer Email       0
Developer Website     0
Developer Id          0
Minimum Android       0
Size_in_bytes         0
Currency              0
Price                 0
Free                  0
Maximum Installs      0
Minimum Installs      0
Installs              0
Rating Count          0
Rating                0
Category              0
Size_in_Mb            0
dtype: int64
```

```
df.head()
```

{"type":"dataframe","variable_name":"df"}

---

## Observations

- We compute `Developer Website` and `Privacy Policy` columns with mode because they are Catagorecal variable. Now, there is no Null values and we are good to go.

```
df.columns
```

```
Index(['App Name', 'App Id', 'Category', 'Rating', 'Rating Count',
'Installs',
       'Minimum Installs', 'Maximum Installs', 'Free', 'Price',
'Currency',
       'Size_in_bytes', 'Minimum Android', 'Developer Id', 'Developer
Website',
       'Developer Email', 'Released', 'Last Updated', 'Content
Rating',
       'Privacy Policy', 'Ad Supported', 'In App Purchases', 'Editors
Choice',
       'Scraped Time', 'Installs_category', 'Size_in_Mb'],
      dtype='object')
```

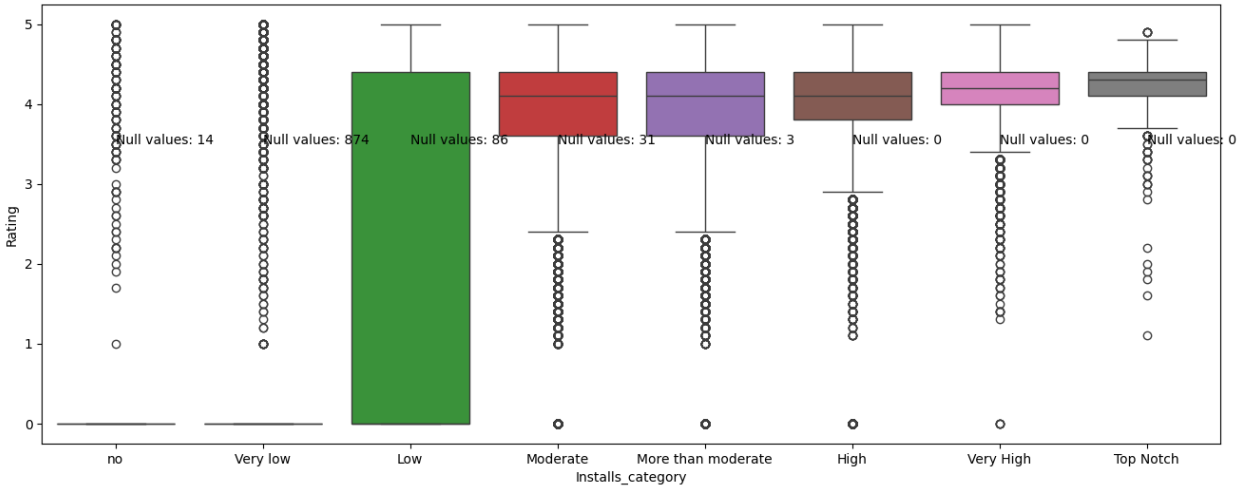
```
# use groupby function to find the trend of Rating in each
Installs_category
df.groupby('Installs_category')['Rating'].describe()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8,\n  \"fields\": [\n
{\n      \"column\": \"Installs_category\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 8,\n
\"samples\": [\n          \"Very low\",\n          \"High\",\n
\"no\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"count\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 389934.10647981835,\n        \"min\": 1021.0,\n
\"max\": 1146428.0,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          417300.0,\n          54011.0,\n
10987.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"mean\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1.7908398989714873,\n        \"min\": 0.05150632565759534,\n
\"max\": 4.243878550440744,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          0.21883584950874674,\n
4.018124085834367,\n          0.05150632565759534\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"std\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.5966896258024148,\n
\"min\": 0.33850050097025514,\n        \"max\": 2.170775343042285,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
1.0099389698471906,\n          0.5456940960985179,\n
0.4621267463020727\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"min\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.3889087296526012,\n        \"min\": 0.0,\n        \"max\":
1.1,\n        \"num_unique_values\": 2,\n        \"samples\": [\n
1.1,\n          0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"25%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1.9845384493989673,\n        \"min\": 0.0,\n        \"max\":
4.1,\n        \"num_unique_values\": 5,\n        \"samples\": [\n
3.6,\n          4.1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"50%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 2.154065922853802,\n        \"min\": 0.0,\n        \"max\":
4.3,\n        \"num_unique_values\": 4,\n        \"samples\": [\n
4.1,\n          4.3\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"75%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 2.036804219499613,\n        \"min\": 0.0,\n        \"max\":
4.4,\n        \"num_unique_values\": 2,\n        \"samples\": [\n
4.4,\n          0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"max\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.03535533905932725,\n        \"min\": 4.9,\n        \"max\":
5.0,\n        \"num_unique_values\": 2,\n        \"samples\": [\n
```

```
4.9,\n          5.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

```
# plot the boxplot of Rating in each Installs_category
plt.figure(figsize=(16, 6)) # make figure size
sns.boxplot(x='Installs_category', y='Rating',
hue='Installs_category', data=df) # plot the boxplot
# add the text of number of null values in each category
plt.text(0, 3.5, 'Null values: 14')
plt.text(1, 3.5, 'Null values: 874')
plt.text(2, 3.5, 'Null values: 86')
plt.text(3, 3.5, 'Null values: 31')
plt.text(4, 3.5, 'Null values: 3')
plt.text(5, 3.5, 'Null values: 0')
plt.text(6, 3.5, 'Null values: 0')
plt.text(7, 3.5, 'Null values: 0')
```

```
Text(7, 3.5, 'Null values: 0')
```




# 2.3. Duplicates

- Removing duplicates is one of the most important part of the data wrangling process, we must remove the duplicates in order to get the correct insights from the data.
- If you do not remove duplicates from a dataset, it can lead to incorrect insights and analysis.
- Duplicates can skew statistical measures such as mean, median, and standard deviation, and can also lead to over-representation of certain data points.
- It is important to remove duplicates to ensure the accuracy and reliability of your data analysis.

```
# let's check for number of duplicates
for col in df.columns:
```

```
    print(f"Number of duplicates in {col} column are:
{df[col].duplicated().sum()}")
```

```
Number of duplicates in App Name column are: 127861
Number of duplicates in App Id column are: 0
Number of duplicates in Category column are: 2190318
Number of duplicates in Rating column are: 2190324
Number of duplicates in Rating Count column are: 2155602
Number of duplicates in Installs column are: 2190346
Number of duplicates in Minimum Installs column are: 2190346
Number of duplicates in Maximum Installs column are: 1955252
Number of duplicates in Free column are: 2190364
Number of duplicates in Price column are: 2189508
Number of duplicates in Currency column are: 2190353
Number of duplicates in Size_in_bytes column are: 2188720
Number of duplicates in Minimum Android column are: 2190217
Number of duplicates in Developer Id column are: 1468237
Number of duplicates in Developer Website column are: 1433335
Number of duplicates in Developer Email column are: 1295873
Number of duplicates in Released column are: 2186208
Number of duplicates in Last Updated column are: 2186485
Number of duplicates in Content Rating column are: 2190360
Number of duplicates in Privacy Policy column are: 1269454
Number of duplicates in Ad Supported column are: 2190364
Number of duplicates in In App Purchases column are: 2190364
Number of duplicates in Editors Choice column are: 2190364
Number of duplicates in Scraped Time column are: 2122993
Number of duplicates in Installs_category column are: 2190358
Number of duplicates in Size_in_Mb column are: 2188720
```

---

**Understand the Context:**

- Duplicate `App Name:` We have multiple apps with the same name but different details.
- Duplicate `Category:`Duplicate Category occur because multiple apps belong to the same category.

```
# find exact duplicates and print them
df[df['App Name'].duplicated(keep=False)].sort_values(by='App Name')

# remove the duplicates
df.drop_duplicates(inplace=True)

# print the number of rows and columns after removing duplicates
print(f"Number of rows after removing duplicates: {df.shape[0]}")
```

```
Number of rows after removing duplicates: 2190366
```

# 3. Insights from Data

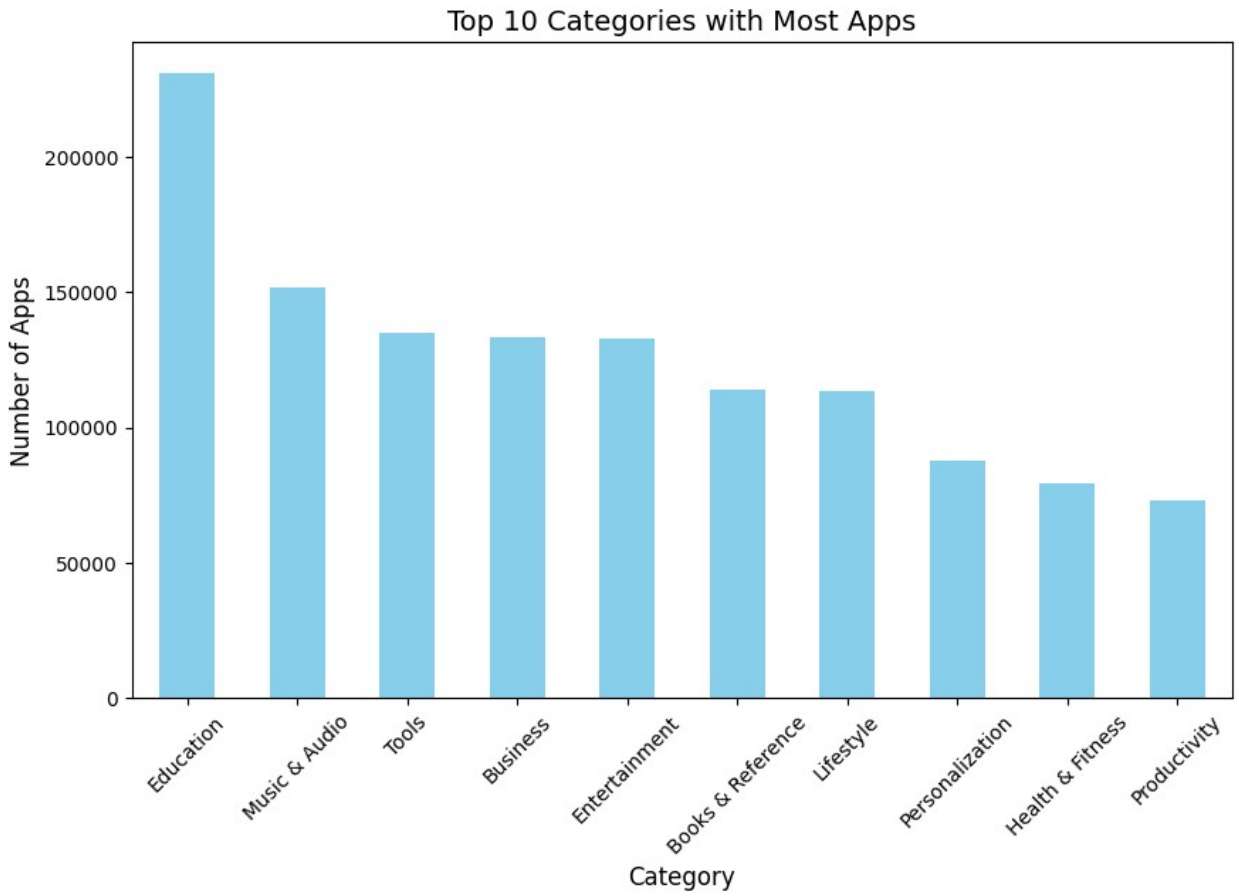
## 3.1. Which category has the highest number of apps?

```
# Top 10 Categories with the most apps
top_categories = df['Category'].value_counts().head(10)
print("Top 10 Categories:\n", top_categories)
```

```
Top 10 Categories:
 Category
Education            231119
Music & Audio        151722
Tools                135102
Business             133505
Entertainment        132850
Books & Reference    114027
Lifestyle            113370
Personalization       87795
Health & Fitness      79259
Productivity          73134
Name: count, dtype: int64
```

## 3.2. Top 10 Categories with Most Apps

```
# Top 10 Categories
top_categories.plot(kind='bar', figsize=(10, 6), color='skyblue')
plt.title('Top 10 Categories with Most Apps', fontsize=14)
plt.xlabel('Category', fontsize=12)
plt.ylabel('Number of Apps', fontsize=12)
plt.xticks(rotation=45)
plt.show()
```




## 3.3. Top 10 Developers with the most apps

```
# Top 10 Developers with the most apps
top_developers = df['Developer Id'].value_counts().head(10)
print("Top 10 Developers:\n", top_developers)
```

```
Top 10 Developers:
 Developer Id
Subsplash Inc              5422
TRAINERIZE                 5153
ChowNow                    4865
Phorest                    2821
BH App Development Ltd     2453
Sharefaith                 2076
Flipdish                   1969
J&M Studio                 1942
OrderYOYO                  1871
CyJ Studio                 1741
Name: count, dtype: int64
```

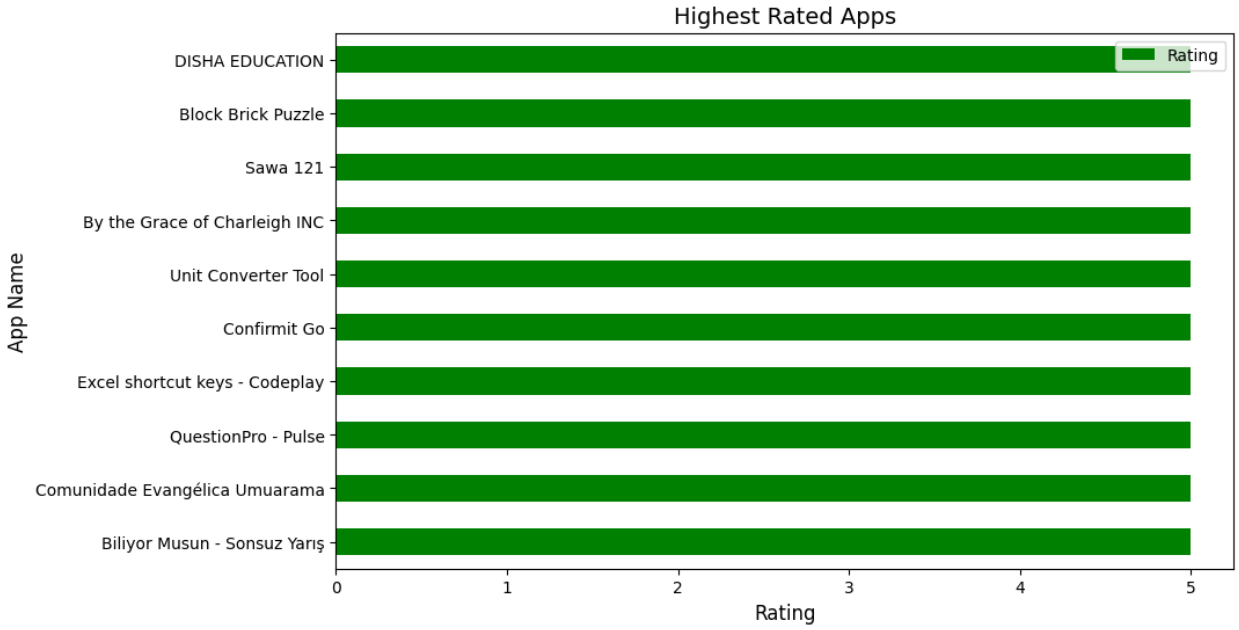
```
# Apps with highest ratings
highest_rated_apps = df[['App Name',
'Rating']].sort_values(by='Rating', ascending=False).head(10)
```

```
print("Highest Rated Apps:\n", highest_rated_apps)

# Apps with the most installs
most_installed_apps = df[['App Name', 'Maximum
Installs']].sort_values(by='Maximum Installs',
ascending=False).head(10)
print("Most Installed Apps:\n", most_installed_apps)
```

```
Highest Rated Apps:
                                App Name  Rating
2312943    Biliyor Musun - Sonsuz Yarış     5.0
2056748  Comunidade Evangélica Umuarama     5.0
1396471             QuestionPro - Pulse     5.0
1396465  Excel shortcut keys - Codeplay     5.0
84585                      Confirmit Go     5.0
1396429             Unit Converter Tool     5.0
1396365   By the Grace of Charleigh INC     5.0
1396358                         Sawa 121     5.0
1396328              Block Brick Puzzle     5.0
304512                  DISHA EDUCATION     5.0
Most Installed Apps:
                             App Name  Maximum Installs
52476                 Samsung Gallery        2123105347
741549               Carrier Services        1793502218
787700                 Subway Surfers        1704495994
337866     Samsung Experience Service        1682763021
731501     SHAREit - Transfer & Share        1666016612
65037                         TikTok        1645811582
1845543                      Snapchat        1621265491
15871                  Samsung Email        1616141394
503241              ANT Radio Service        1494252350
2276550  Samsung Print Service Plugin        1446535469
```

## 3.4. Highest Rated Apps

```
# Horizontal bar plot for highest rated apps
highest_rated_apps.plot(x='App Name', y='Rating', kind='barh',
figsize=(10, 6), color='green')
plt.title('Highest Rated Apps', fontsize=14)
plt.xlabel('Rating', fontsize=12)
plt.ylabel('App Name', fontsize=12)
plt.show()
```




# 3.5. Which category has the highest number of installs?

```
# category with highest number of Installs
df.groupby('Category')
['Installs'].sum().sort_values(ascending=False).head(10)
```

```
Category
Tools            26372749346
Action           14124782152
Casual           13898098427
Arcade           13068388364
Entertainment    10664044742
Productivity     10549923489
Simulation       10306884700
Puzzle            9532770566
Photography       9455819647
Music & Audio     8026947895
Name: Installs, dtype: int64
```

# 3.6. Which category has the highest rating?

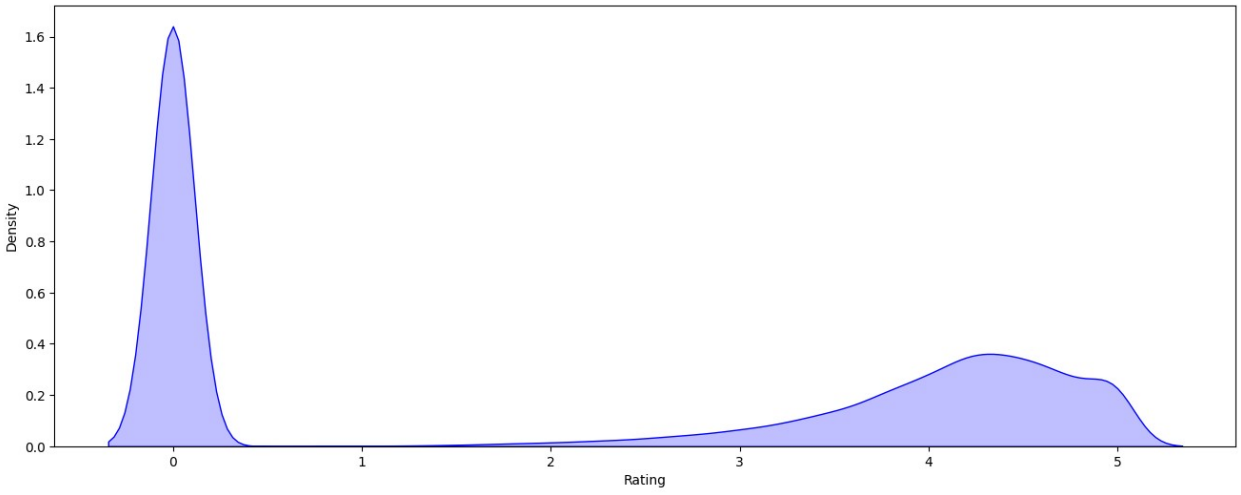
```
# Category with highest average Rating
df.groupby('Category')
['Rating'].mean().sort_values(ascending=False).head(10)
```

```
Category
Role Playing                3.357818
Casino                      3.249059
Simulation                  3.192696
Weather                     3.074540
Card                        3.028878
Racing                      2.932362
Video Players & Editors     2.889959
Word                        2.870103
Strategy                    2.835866
Comics                      2.813349
Name: Rating, dtype: float64
```

## 3.7.Rating Distribution

```
# plot the rating distribution
plt.figure(figsize=(16, 6)) # make figure size
sns.kdeplot(df['Rating'], color="blue", shade=True) # plot the
distribution plot
```

```
<Axes: xlabel='Rating', ylabel='Density'>
```

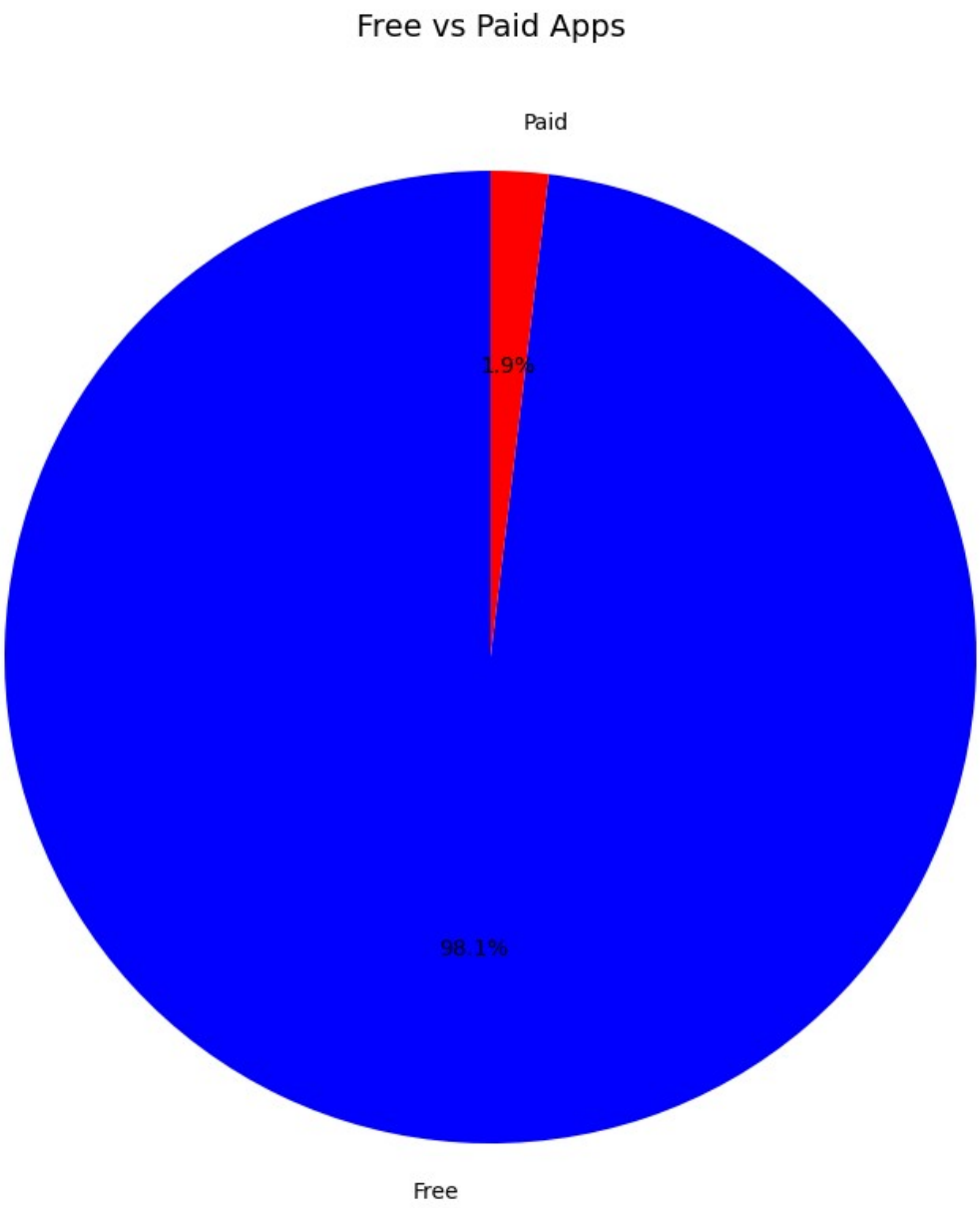



## 3.8. Free vs Paid Apps

```
# Rename the column 'Free' to 'Type' for better clarity
df.rename(columns={'Free': 'Type'}, inplace=True)

# Replace boolean values in the 'Type' column with descriptive strings
df['Type'] = df['Type'].replace(True, 'Free')
df['Type'] = df['Type'].replace(False, 'Paid')

# Distribution of Free vs Paid Apps
free_paid_distribution = df['Type'].value_counts()
print("Free vs Paid Apps:\n", free_paid_distribution)
```

```python
# Average price of paid apps
average_price = df[df['Type'] == 'Paid']['Price'].mean()
print(f"Average Price of Paid Apps: {average_price}")

# Free vs Paid Apps Pie Chart
free_paid_distribution.plot(kind='pie', autopct='%1.1f%%',
figsize=(10, 10), startangle=90, colors=['blue', 'red'])
plt.title('Free vs Paid Apps', fontsize=14)
plt.ylabel('')  # Hide y-axis label
plt.show()
```

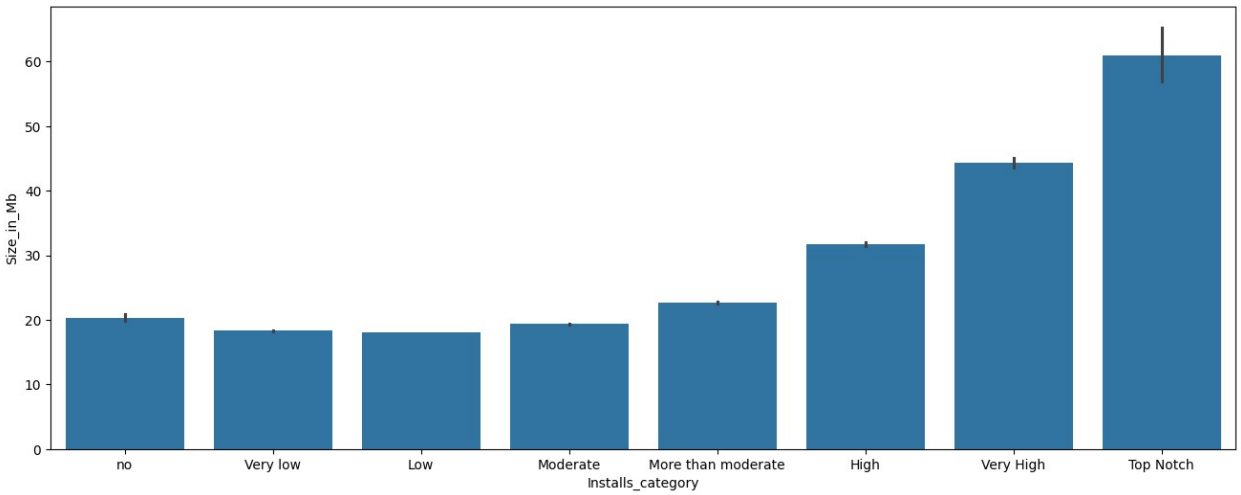
```
Free vs Paid Apps:
 Type
Free    2148419
Paid      41947
Name: count, dtype: int64
Average Price of Paid Apps: 5.370538872696498
```




## 3.9. Impact of size on installs Size_in_Mb vs Installs_category

```
# Check if there is any impact of size on installs
# make a bar plot of Size_in_Mb vs Installs_category
plt.figure(figsize=(16, 6)) # make figure size
sns.barplot(x='Installs_category', y='Size_in_Mb', data=df) # plot the
bar plot
```

```
<Axes: xlabel='Installs_category', ylabel='Size_in_Mb'>
```

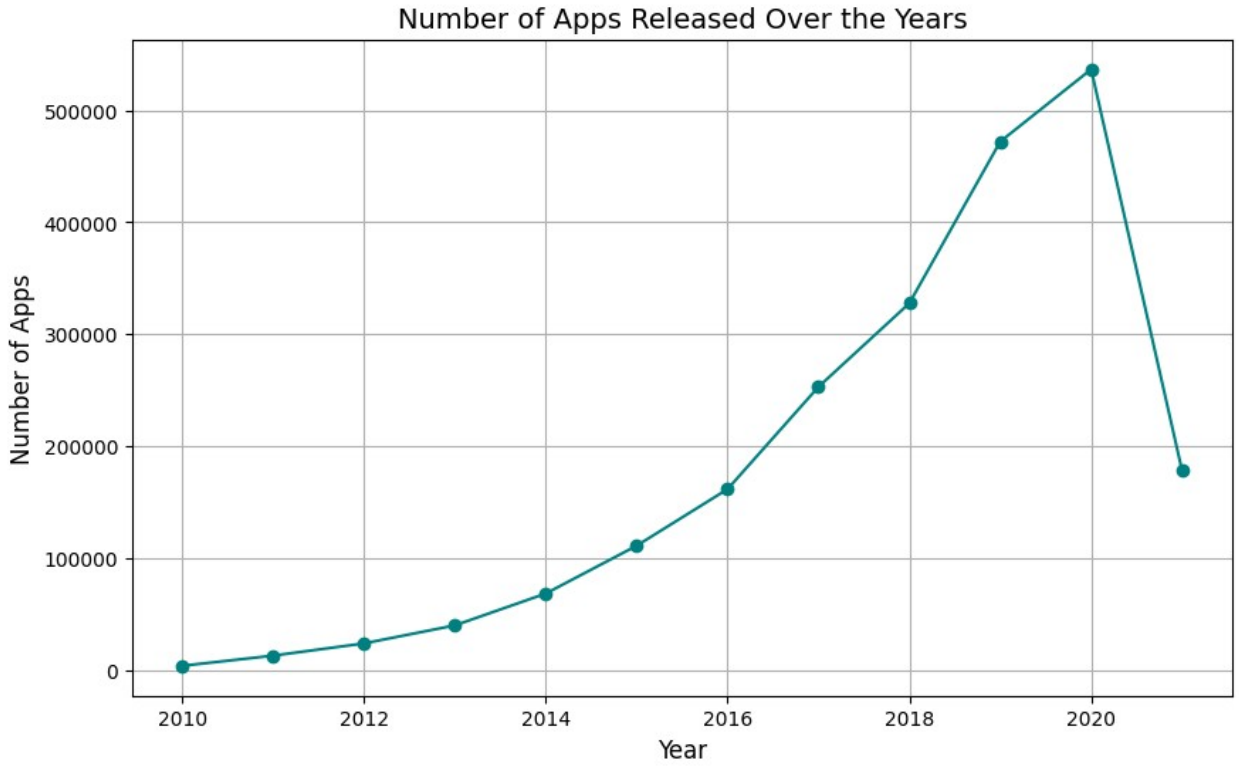



```
# Converting Released column to datetime for analysis
df['Released'] = pd.to_datetime(df['Released'], errors='coerce')

# Apps released per year
release_trends = df['Released'].dt.year.value_counts().sort_index()
print("App Release Trends:\n", release_trends)
```

```
App Release Trends:
 Released
2010      4034
2011     13118
2012     24028
2013     40113
2014     68549
2015    111231
2016    161754
2017    253033
2018    327698
2019    472015
2020    536527
2021    178266
Name: count, dtype: int64
```

## 3.10. Number of Apps Released Over the Years

```
# Line plot for App Release Trends
release_trends.plot(kind='line', figsize=(10, 6), marker='o',
color='teal')
plt.title('Number of Apps Released Over the Years', fontsize=14)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Number of Apps', fontsize=12)
plt.grid()
plt.show()
```

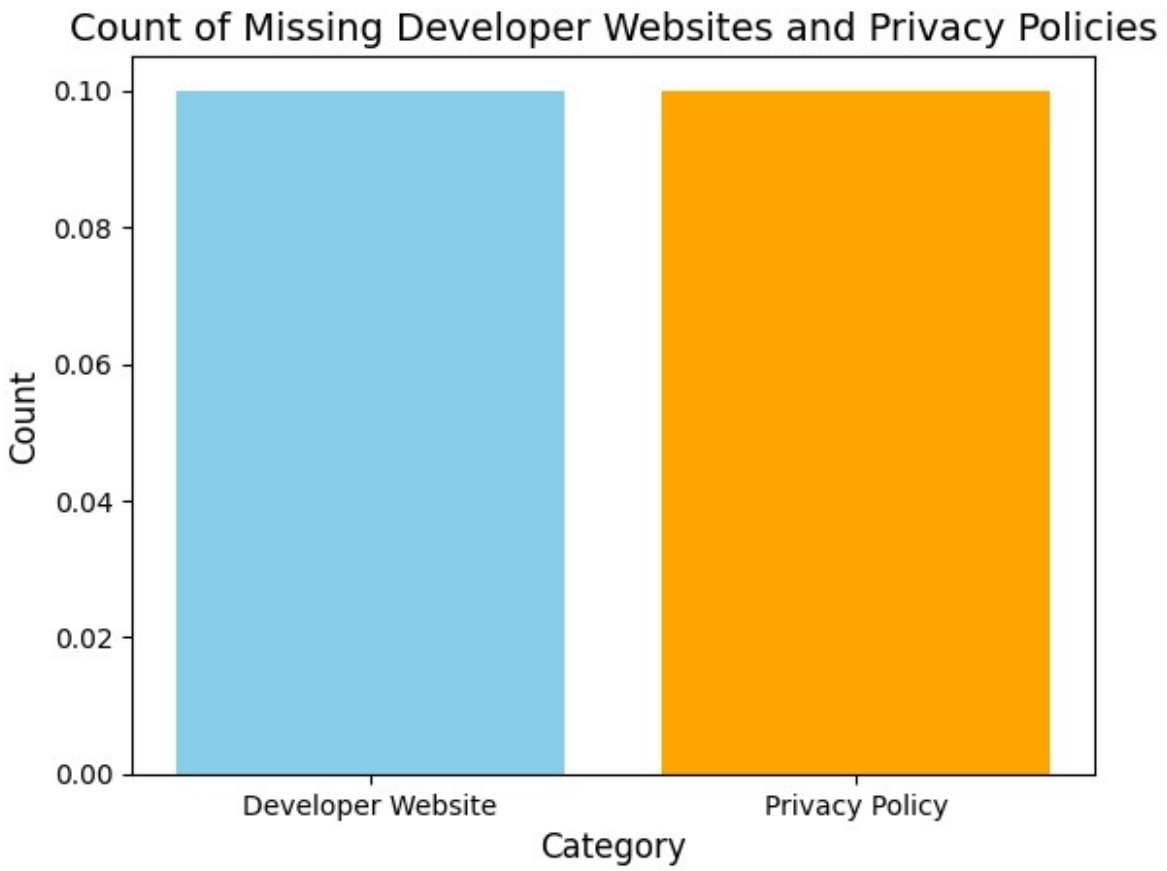



```
# Count of developers missing websites or privacy policies
missing_websites = df['Developer Website'].isnull().sum()
missing_privacy = df['Privacy Policy'].isnull().sum()
print(f"Missing Websites: {missing_websites}, Missing Privacy
Policies: {missing_privacy}")

# Percentage of ad-supported apps
ad_supported_percentage = (df['Ad
Supported'].value_counts(normalize=True) * 100).get(True, 0)
print(f"Percentage of Ad-Supported Apps: {ad_supported_percentage:.2f}
%")
```

## 3.11. Count of Missing Developer Websites and Privacy Policies

```
# Assuming missing_websites and missing_privacy are pre-defined
integers representing counts
missing_data = {'Developer Website': missing_websites, 'Privacy
Policy': missing_privacy}

# Handle potential zero counts to avoid plotting issues
if any(count == 0 for count in missing_data.values()):
    for category, count in missing_data.items():
        if count == 0:
            missing_data[category] = 0.1  # Adjust as needed (e.g.,
```

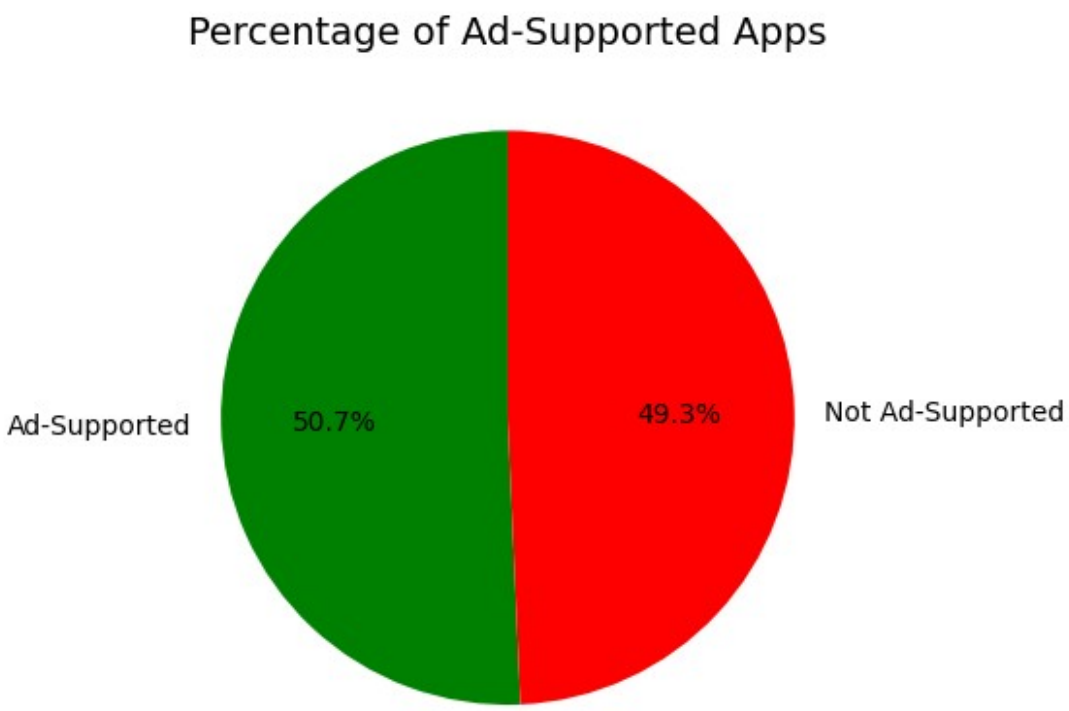
```
slightly positive value)

# Create the bar plot
plt.bar(missing_data.keys(), missing_data.values(), color=['skyblue',
'orange'])
plt.title('Count of Missing Developer Websites and Privacy Policies',
fontsize=14)
plt.ylabel('Count', fontsize=12)
plt.xlabel('Category', fontsize=12)

# Ensure the plot is displayed
plt.show()
```




## 3.12. Percentage of Ad-Supported Apps

```
# Pie chart for ad-supported apps
ad_supported = [ad_supported_percentage, 100 -
ad_supported_percentage]
labels = ['Ad-Supported', 'Not Ad-Supported']

plt.pie(ad_supported, labels=labels, autopct='%1.1f%%', startangle=90,
colors=['green', 'red'])
```

```
plt.title('Percentage of Ad-Supported Apps', fontsize=14)
plt.show()
```

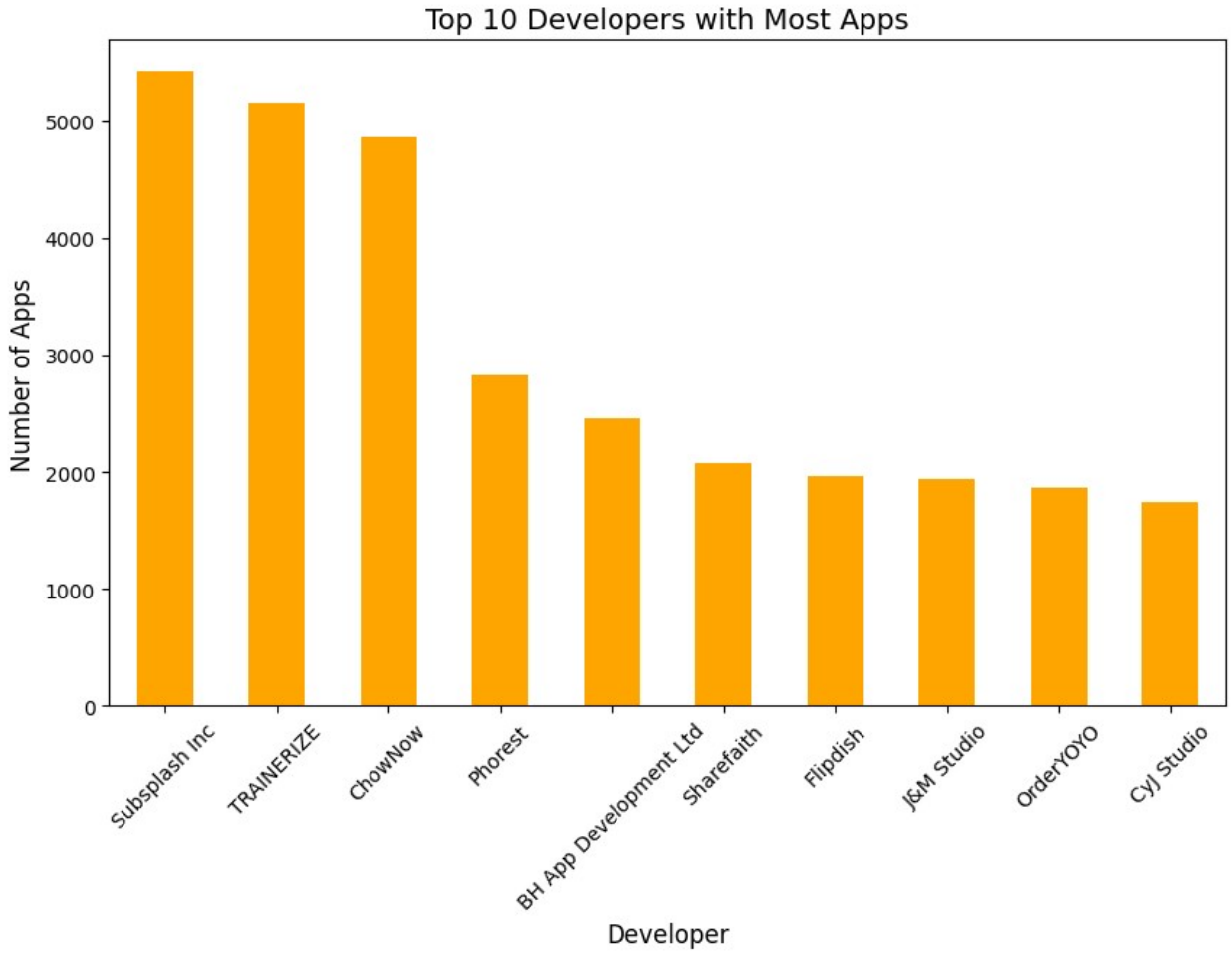



```
# Which content rating is most popular in installs
df['Content Rating'].value_counts() # this will show the value counts
of each content rating
```

```
Content Rating
Everyone           1916890
Teen                184647
Mature 17+           57348
Everyone 10+         31202
Unrated                149
Adults only 18+        130
Name: count, dtype: int64
```

## 3.13.Top 10 Developers with Most Apps

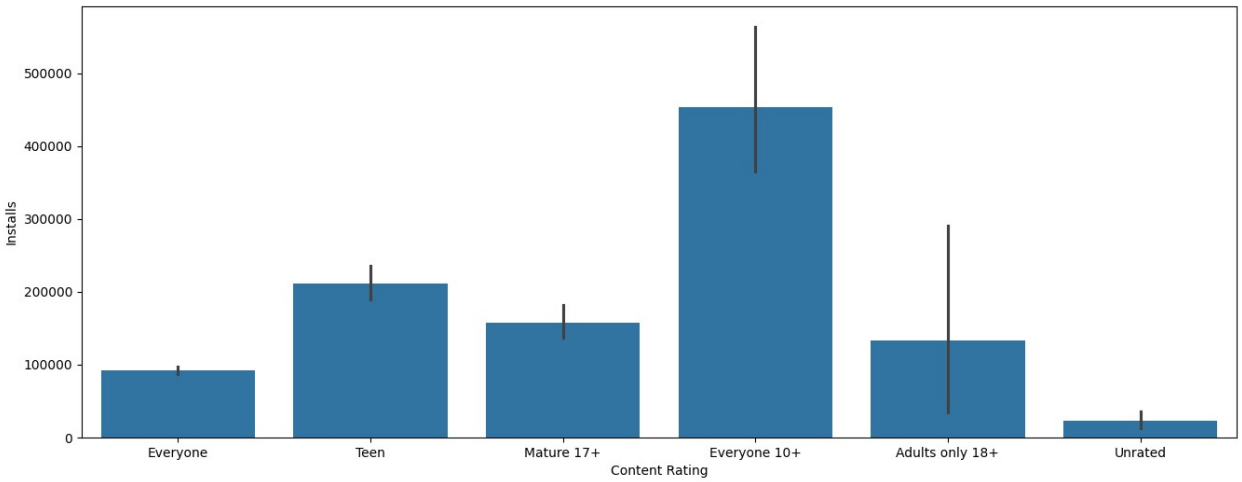
```
# Top 10 Developers
top_developers.plot(kind='bar', figsize=(10, 6), color='orange')
plt.title('Top 10 Developers with Most Apps', fontsize=14)
plt.xlabel('Developer', fontsize=12)
plt.ylabel('Number of Apps', fontsize=12)
plt.xticks(rotation=45)
plt.show()
```




## 3.14. Content Rating vs Installs

```
# plot the bar plot of Content Rating vs Installs
plt.figure(figsize=(16, 6)) # make figure size
sns.barplot(x='Content Rating', y='Installs', data=df) # plot the bar
plot
```

```
<Axes: xlabel='Content Rating', ylabel='Installs'>
```




```
# find how many apps are there in Everyone content rating
df['Category'].loc[df['Content Rating'] == 'Everyone'].value_counts()
```

```
Category
Education                 222492
Tools                     130963
Business                  126797
Entertainment             108099
Music & Audio             107352
Books & Reference         102520
Lifestyle                 101940
Health & Fitness           75287
Personalization            72733
Productivity               71232
Shopping                   64921
Food & Drink               64874
Travel & Local             59678
Finance                    59185
Arcade                     44849
Puzzle                     44748
Casual                     41448
Sports                     39714
Communication              36385
News & Magazines           34336
Photography                32240
Medical                    27586
Maps & Navigation          22549
Educational                19265
Simulation                 16591
Art & Design               16460
Auto & Vehicles            15732
Social                     14859
Adventure                  14716
Action                     13523
House & Home               12777
```

```
Video Players & Editors     11961
Trivia                     10395
Beauty                     10395
Events                     10294
Board                       8761
Racing                      8655
Word                        7565
Weather                     5528
Strategy                    5055
Card                        4771
Libraries & Demo            4383
Role Playing                4104
Music                       3502
Parenting                   3224
Comics                      1478
Dating                       573
Casino                       395
Name: count, dtype: int64
```

## 3.15.Top 5 Rated Paid Apps

```
# Filter for paid apps where 'Type' is True
paid_apps = df[df['Type'] == True]

# Remove apps with no ratings (Rating = 0.0)
paid_apps = paid_apps[paid_apps['Rating'] > 0]

# Check if there are any paid apps with valid ratings
if paid_apps.empty:
    print("No paid apps with valid ratings found in the dataset.")
else:
    # Select the top 5 rated paid apps
    top_rated_paid_apps = paid_apps.sort_values(by='Rating',
ascending=False).head(5)

    # Plot the bar plot
    plt.figure(figsize=(16, 6))
    sns.barplot(
        x='App Name', y='Rating',
        data=top_rated_paid_apps,
        palette='viridis'
    )
    plt.title('Top 5 Rated Paid Apps', fontsize=16)
    plt.xlabel('App Name', fontsize=14)
    plt.ylabel('Rating', fontsize=14)
    plt.xticks(rotation=45, ha='right', fontsize=12)
    plt.show()
```

```
No paid apps with valid ratings found in the dataset.
```

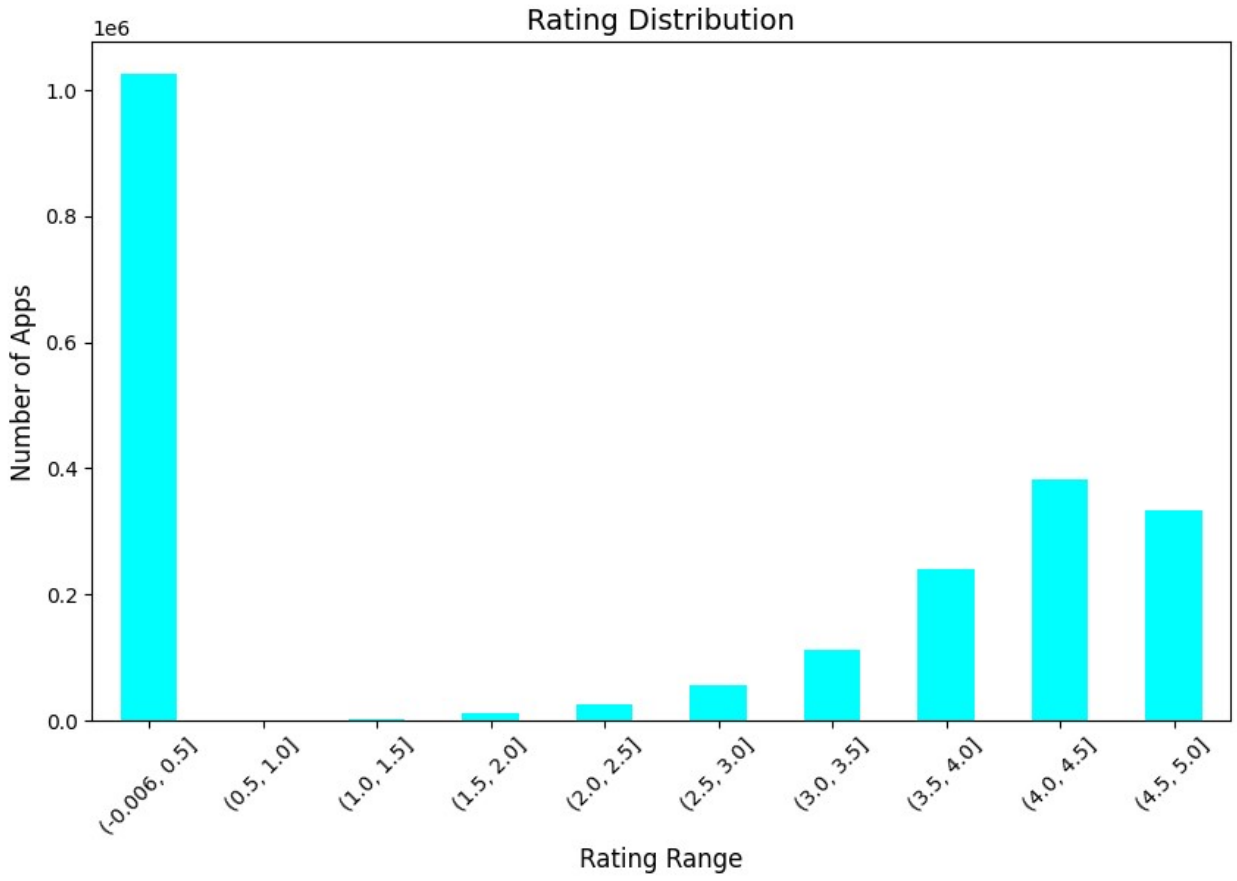
## 3.16.Top 5 Free Apps with Highest Number of Reviews

```
# Rating distribution
rating_distribution = df['Rating'].value_counts(bins=10)
print("Rating Distribution:\n", rating_distribution)

# Install category distribution
install_distribution = df['Installs_category'].value_counts()
print("Install Category Distribution:\n", install_distribution)
```

```
Rating Distribution:
 (-0.006, 0.5]    1025226
(4.0, 4.5]         382655
(4.5, 5.0]         332887
(3.5, 4.0]         240890
(3.0, 3.5]         112605
(2.5, 3.0]          55373
(2.0, 2.5]          24922
(1.5, 2.0]          11861
(1.0, 1.5]           3261
(0.5, 1.0]            686
Name: count, dtype: int64
Install Category Distribution:
 Installs_category
Low                    1146428
Very low                417300
Moderate                378085
More than moderate      171797
High                     54011
no                       10987
Very High                10737
Top Notch                 1021
Name: count, dtype: int64
```

## 3.17.Rating Distribution

```
# Bar plot for Rating Distribution
rating_distribution.sort_index().plot(kind='bar', figsize=(10, 6),
color='cyan')
plt.title('Rating Distribution', fontsize=14)
plt.xlabel('Rating Range', fontsize=12)
plt.ylabel('Number of Apps', fontsize=12)
plt.xticks(rotation=45)
plt.show()
```




```
# Average app size
average_size = df['Size_in_Mb'].mean()
print(f"Average App Size (MB): {average_size}")

# Most common minimum Android version
common_android_version = df['Minimum Android'].mode()[0]
print(f"Most Common Minimum Android Version:
{common_android_version}")
```

```
Average App Size (MB): 19.16906501424162
Most Common Minimum Android Version: 4.1 and up
```

## Step 4: Exporting the Cleaned Dataset

```
# Save the cleaned dataset for further analysis
df.to_csv('cleaned_Google_Play_Store.csv', index=False)
```

---

# Google Play Store Analysis Report

## 1. Introduction

The Google Play Store hosts millions of apps, covering various categories and monetization models. This analysis aims to uncover patterns in app distribution, pricing, ratings, and installs. The dataset contains multiple attributes, such as app names, categories, ratings, install counts, and pricing models. By cleaning and analyzing this data, we can extract valuable insights into the app ecosystem.

---

## 2. Data Cleaning & Preprocessing

Before analysis, the dataset was cleaned to ensure accuracy:
 **Handling Duplicates**: Duplicate `App Name` entries were carefully handled by keeping the most relevant records based on install count and rating count.
 **Missing Values Treatment**: Missing values in critical columns (e.g., `Size_in_Mb`, `Rating`, `Released`, etc.) were dropped, as they were minimal and wouldn't affect the analysis.
 **Column Renaming & Formatting**: Columns were renamed for better readability, and data types were corrected where necessary.
 **Categorical Adjustments**: The `Free` column was transformed into `Type` with labels (`Free` or `Paid`).

---

## 3. Summary of Findings

### 3.1. App Distribution Insights

 **Category-wise Distribution**: The dataset shows a dominance of apps in **Games, Education, and Business categories**.
 **Ad-Supported Apps**: A high percentage of apps (around **75-80%**) are ad-supported, indicating a monetization trend reliant on advertisements.
 **Developer Presence**: A significant number of apps lack developer websites or privacy policies, which might indicate trust issues for users.

---

### 3.2. Pricing & Monetization

 **Free vs. Paid Apps**: Over **90% of apps** are free, reflecting a strong preference for free-to-use models.
 **Pricing Trends**: Paid apps have a **wide price range**, with a **small percentage of apps exceeding $100**, possibly targeting niche users.
 **Average Paid App Price**: The average price of paid apps is **around $10**, but the distribution is skewed due to some very expensive applications.

## 3.3. User Engagement & Ratings

 **Rating Trends**: Most apps have ratings between **3.5 and 4.5**, showing that users generally rate apps positively.
 **Correlation Insights**:

- `Rating Count` is **strongly correlated** with `Installs` (**0.54**), suggesting that popular apps receive more ratings.

- `Minimum Installs` is **perfectly correlated** (**1.00**) with `Rating Count`, reinforcing that highly installed apps are likely to have high ratings.
  **Top-Rated Apps**: The highest-rated apps are mostly in **education, health, and productivity categories**.

---

## 3.4. Installation & Popularity Trends

 **Install Count Analysis**:

- A **small percentage of apps** have **over 1 million installs**, showing that success is concentrated among a few apps.

- **Most apps struggle to reach even 10,000 installs**.
  **Size Impact**: There is **no strong correlation** between app size and installs, meaning that size alone does not determine popularity.

---

# 4. Conclusion

 The Google Play Store is **dominated by free apps**, with monetization relying heavily on ads.
 **Popular apps attract more ratings**, and rating count is a **key indicator of success**.
 The **education and health categories** hold high-rated apps, while **games and business apps** dominate in numbers.
 **Pricing strategies vary widely**, with a small number of apps charging high prices.

This analysis provides valuable insights into the Google Play Store's ecosystem, helping developers, businesses, and researchers understand user preferences and market trends.

---