Google Play Store Apps (Exploratory Data Analysis)

1 - Introduction

Project Overview:

Welcome to my Exploratory Data Analysis (EDA) project on the Google Play Store Apps Dataset! In this project, I will dive into a comprehensive analysis of more than 2,000,000 records available on the Google Play Store dataset. My primary objective is to gain valuable insights into the characteristics of these apps and understand the factors that contribute to their popularity and success.

Dataset Description:

The dataset I will use for this analysis contains information about various mobile applications listed on the Google Play Store. It comprises 23 attributes that provide valuable details about each app, such as its name, category, rating, installs, pricing, content rating, and more. Below is a brief description of the key attributes present in the dataset:

1. **App Name:** The name of the application.
2. **App Id:** A unique identifier for each app.
3. **Category:** The category to which the app belongs (e.g., productivity, games, social, etc.).
4. **Rating:** The average user rating of the app.
5. **Rating Count:** The number of user ratings received by the app.
6. **Installs:** The total number of times the app has been installed.
7. **Minimum Installs:** The minimum number of installs required for the app.
8. **Maximum Installs:** The maximum number of installs the app has achieved.
9. **Free:** A binary indicator (True/False) to denote if the app is free to download.
10. **Price:** The price of the app (if not free).
11. **Currency:** The currency used for pricing.
12. **Size:** The size of the app.
13. **Minimum Android:** The minimum Android version required to run the app.
14. **Developer Id:** The unique identifier for the app's developer.
15. **Developer Website:** The website of the app's developer.
16. **Developer Email:** The email address of the app's developer.
17. **Released:** The date when the app was released.
18. **Privacy Policy:** The URL to the app's privacy policy.
19. **Last Updated:** The date when the app was last updated.
20. **Content Rating:** The age group for which the app's content is suitable (e.g., Everyone, Teen, Mature 17+, etc.).

21. **Ad Supported:** A binary indicator (True/False) to denote if the app contains ads.
22. **In-app Purchases:** A binary indicator (True/False) to denote if the app offers in-app purchases.
23. **Editor Choice:** A binary indicator (True/False) to denote if the app has been selected as an editor's choice.

Throughout this EDA project, I will utilize various data analysis and visualization techniques to explore the dataset, extract meaningful patterns, and draw valuable conclusions.

Importing Libraries and Dataset:

In [248…

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

df = pd.read_csv('Google-Playstore.csv')
pd.set_option('display.max_columns', None)
df.head()
```

Out[248]:

	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Pr
0	Gakondo	com.ishakwe.gakondo	Adventure	0.0	0.0	10+	10.0	15	True	
1	Ampere Battery Info	com.webserveis.batteryinfo	Tools	4.4	64.0	5,000+	5000.0	7662	True	
2	Vibook	com.doantiepvien.crm	Productivity	0.0	0.0	50+	50.0	58	True	
3	Smart City Trichy Public Service Vehicles 17UC...	cst.stJoseph.ug17ucs548	Communication	5.0	5.0	10+	10.0	19	True	
4	GROW.me	com.horodyski.grower	Tools	0.0	0.0	100+	100.0	478	True	

In [249…

```
print(f'We have {df.shape[0]} Records, and {df.shape[1]} Columns in the dataset.')
```

```
We have 2312944 Records, and 24 Columns in the dataset.
```

2 - Data Cleaning

Handling Missing Values:

In [250…

```
# missing percentage
(df.isnull().sum() / len(df)) * 100
```

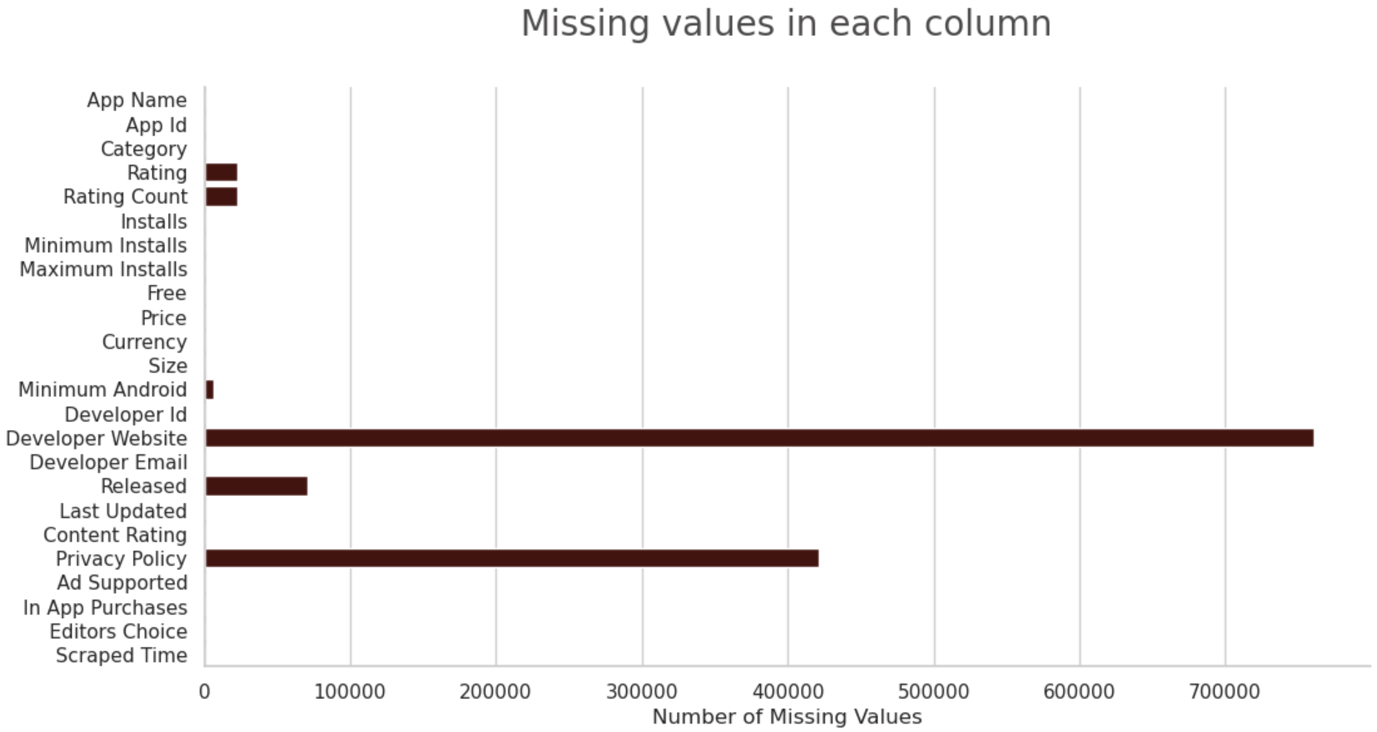
Out[250]:

```
App Name                  0.000086
App Id                    0.000000
Category                  0.000000
Rating                    0.989345
Rating Count              0.989345
Installs                  0.004626
```

```
Minimum Installs       0.004626
Maximum Installs       0.000000
Free                   0.000000
Price                  0.000000
Currency               0.005837
Size                   0.008474
Minimum Android        0.282324
Developer Id           0.001427
Developer Website     32.894657
Developer Email        0.001340
Released               3.071972
Last Updated           0.000000
Content Rating         0.000000
Privacy Policy        18.199879
Ad Supported           0.000000
In App Purchases       0.000000
Editors Choice         0.000000
Scraped Time           0.000000
dtype: float64
```

In [251…

```
# PLOTS RULES.
f_size = (12, 6)
plt_color = '#4a0d08'
title_color = '#4d4b4b'
title_pad = 30
title_size = 20

# bar plot to visualize the missing values
plt.figure(figsize=f_size)

sns.barplot(x=df.isnull().sum().values,
            y=df.isnull().sum().index,
           color=plt_color)

plt.title('Missing values in each column', fontsize=title_size, color=title_color, pad=t
plt.xlabel('Number of Missing Values')

sns.despine(right=True, top=True)
plt.show()
```



Given that the "Developer Website" column contains approximately 33% null values and is not crucial for our analysis, we will drop it from the dataset. Similarly, we will also remove the "Privacy Policy" column. Since our dataset comprises a substantial amount of data, around 2.3 million records, we can safely eliminate rows containing null values, as they do not significantly affect our ultimate objectives.

We can also see that the columns "App Id," "Developer Id," "Developer Email," and "Scraped Time" are not necessary for our analysis. Let's drop them to simplify our dataset and focus on the important information.

In [252…

```
# remove useless columns
df.drop(['Developer Website', 'Privacy Policy', 'App Id', 'Developer Id', 'Scraped Time'

rec_before = df.shape[0]

# drop rows with any null value
df.dropna(inplace=True)

print(f'The original number of records we had was {rec_before}. After dropping rows with
```

```
The original number of records we had was 2312944. After dropping rows with nulls, we en
d up with 2235309 records. This means we removed 3.36% of the records.
```

In [253…

```
# check for missing percentage again
(df.isnull().sum() / len(df)) * 100
```

Out[253]:

```
App Name             0.0
Category             0.0
Rating               0.0
Rating Count         0.0
Installs             0.0
Minimum Installs     0.0
Maximum Installs     0.0
Free                 0.0
Price                0.0
Currency             0.0
Size                 0.0
Minimum Android      0.0
Released             0.0
Last Updated         0.0
Content Rating       0.0
Ad Supported         0.0
In App Purchases     0.0
Editors Choice       0.0
dtype: float64
```

Data Formatting and Conversion:

In [254…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2235309 entries, 0 to 2312943
Data columns (total 18 columns):
 #   Column            Dtype
---  ------            -----
 0   App Name          object
 1   Category          object
 2   Rating            float64
 3   Rating Count      float64
 4   Installs          object
 5   Minimum Installs  float64
 6   Maximum Installs  int64
 7   Free              bool
 8   Price             float64
 9   Currency          object
 10  Size              object
 11  Minimum Android   object
 12  Released          object
 13  Last Updated      object
 14  Content Rating    object
 15  Ad Supported      bool
 16  In App Purchases  bool
 17  Editors Choice    bool
dtypes: bool(4), float64(4), int64(1), object(9)
memory usage: 264.3+ MB
```

"First of all, we need to convert all columns with the `object` data type to the `string` data type to achieve memory efficiency."

In [255…

```
for column in ['App Name', 'Category', 'Installs', 'Currency', 'Size','Minimum Android',
    df[column] = df[column].astype('string')
```

"We can drop the 'Installs' column since we can calculate the average installation number using 'Minimum Installs' and 'Maximum Installs' columns."

In [256…

```
# add calculated average installation
df['Avg Installs'] = np.ceil((df['Minimum Installs'] + df['Maximum Installs']) / 2).asty

# remove useless columns
df.drop(['Installs', 'Minimum Installs', 'Maximum Installs'], axis=1, inplace=True)
```

In [257…

```
df.head(3)
```

Out[257]:

	App Name	Category	Rating	Rating Count	Free	Price	Currency	Size	Minimum Android	Released	Last Updated	Content Rating
0	Gakondo	Adventure	0.0	0.0	True	0.0	USD	10M	7.1 and up	Feb 26, 2020	Feb 26, 2020	Everyone
1	Ampere Battery Info	Tools	4.4	64.0	True	0.0	USD	2.9M	5.0 and up	May 21, 2020	May 06, 2021	Everyone
2	Vibook	Productivity	0.0	0.0	True	0.0	USD	3.7M	4.0.3 and up	Aug 9, 2019	Aug 19, 2019	Everyone

In [258…

```
# remove 'M' letter from 'Size' column and append it in column name
df['Size'] = df['Size'].str.replace(r'[^\d]+', '', regex=True)

# remove 'and up' from 'Minimum Android' column
df['Minimum Android'] = df['Minimum Android'].str.replace(r'[^\d]+', '', regex=True)

df.rename(columns={'Size': 'Size(Megabyte)', 'Released': 'Released Date', 'Last Updated'
```

In [259…

```
# change columns to proper data types
df['Rating Count'] = df['Rating Count'].astype('int')

df['Size(Megabyte)'] = df['Size(Megabyte)'].replace('', '0')
df['Size(Megabyte)'] = df['Size(Megabyte)'].astype('int')

df['Minimum Android'] = df['Minimum Android'].replace('', '0')
df['Minimum Android'] = df['Minimum Android'].apply(lambda x: int(str(x)[0]))
df['Minimum Android'] = df['Minimum Android'].astype('int')

# deal with dates columns
df["Released Date"] = pd.to_datetime(df["Released Date"], format="%b %d, %Y")
df["Last Update Date"] = pd.to_datetime(df["Last Update Date"], format="%b %d, %Y")
```

```
# add a boolean column to indicate whether the app has been updated or not
df['Updated'] = (df['Last Update Date'] > df['Released Date']).astype('bool')
```

In [260…

```
df.head(3)
```

Out[260]:

	App Name	Category	Rating	Rating Count	Free	Price	Currency	Size(Megabyte)	Minimum Android	Released Date	Last Update Date
0	Gakondo	Adventure	0.0	0	True	0.0	USD	10	7	2020-02-26	2020-02-26
1	Ampere Battery Info	Tools	4.4	64	True	0.0	USD	29	5	2020-05-21	2021-05-06
2	Vibook	Productivity	0.0	0	True	0.0	USD	37	4	2019-08-09	2019-08-19

In [261…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2235309 entries, 0 to 2312943
Data columns (total 17 columns):
 #   Column            Dtype         
---  ------            -----         
 0   App Name          string        
 1   Category          string        
 2   Rating            float64       
 3   Rating Count      int64         
 4   Free              bool          
 5   Price             float64       
 6   Currency          string        
 7   Size(Megabyte)    int64         
 8   Minimum Android   int64         
 9   Released Date     datetime64[ns]
 10  Last Update Date  datetime64[ns]
 11  Content Rating    string        
 12  Ad Supported      bool          
 13  In App Purchases  bool          
 14  Editors Choice    bool          
 15  Avg Installs      int64         
 16  Updated           bool          
dtypes: bool(5), datetime64[ns](2), float64(2), int64(4), string(4)
memory usage: 232.4 MB
```

Removing Duplicates:

In [262…

```
# check for duplicated rows
df[df.duplicated()]
```

Out[262]:

	App Name	Category	Rating	Rating Count	Free	Price	Currency	Size(Megabyte)	Minimum Android	Released Date
185370	呆萌鸟2	Casual	0.0	0	True	0.0	USD	13	4	2019-12-20
546688	□□□□□□ □□□□□ - Look Moo Mobile	Business	0.0	0	True	0.0	USD	23	5	2019-03-27
709903	Saraybosna Üniversitesi	Education	0.0	0	True	0.0	USD	64	4	2017-05-23
912802	Bangladesh	Social	0.0	0	True	0.0	USD	78	4	2018-06-

	Jubo Mohila League									05
1400173	Kosova Eğitim	Education	0.0	0	True	0.0	USD	63	4	2018-03-09
1556914	Book, The Attache	Books & Reference	0.0	0	True	0.0	USD	69	6	2020-04-26
1680429	Lou Streetfood	Food & Drink	0.0	0	True	0.0	USD	19	4	2021-03-30
1951985	Baltimore Traveler Map All Amenity & ATM Finder	Maps & Navigation	0.0	0	True	0.0	USD	34	4	2020-03-01
2128928	CONCEPT ACADEMY	Education	0.0	0	True	0.0	USD	37	4	2020-03-30
2199941	VZ | Exprésate Lector Unidad 4	Education	0.0	0	True	0.0	USD	31	4	2020-02-27

In [263…

```
# remove duplicates
df.drop_duplicates(inplace=True, keep='last')

df[df.duplicated()]
```

Out[263]:

	App Name	Category	Rating	Rating Count	Free	Price	Currency	Size(Megabyte)	Minimum Android	Released Date	Last Update Date	Conte Rati

In [264…

```
# save dataset
df.to_csv('Clean_dataset.csv')
```

3 - Exploratory Data Analysis & Data Visualization

Exploratory Data Analysis (EDA) is a preliminary data investigation technique to understand patterns and relationships in the dataset. It involves using statistical tools and visualizations to gain insights and identify potential trends or anomalies, guiding further data exploration and analysis.

As part of your Exploratory Data Analysis (EDA) project, we can explore various aspects of the dataset to gain insights and answer meaningful questions about the mobile apps. Here are some potential questions to consider:

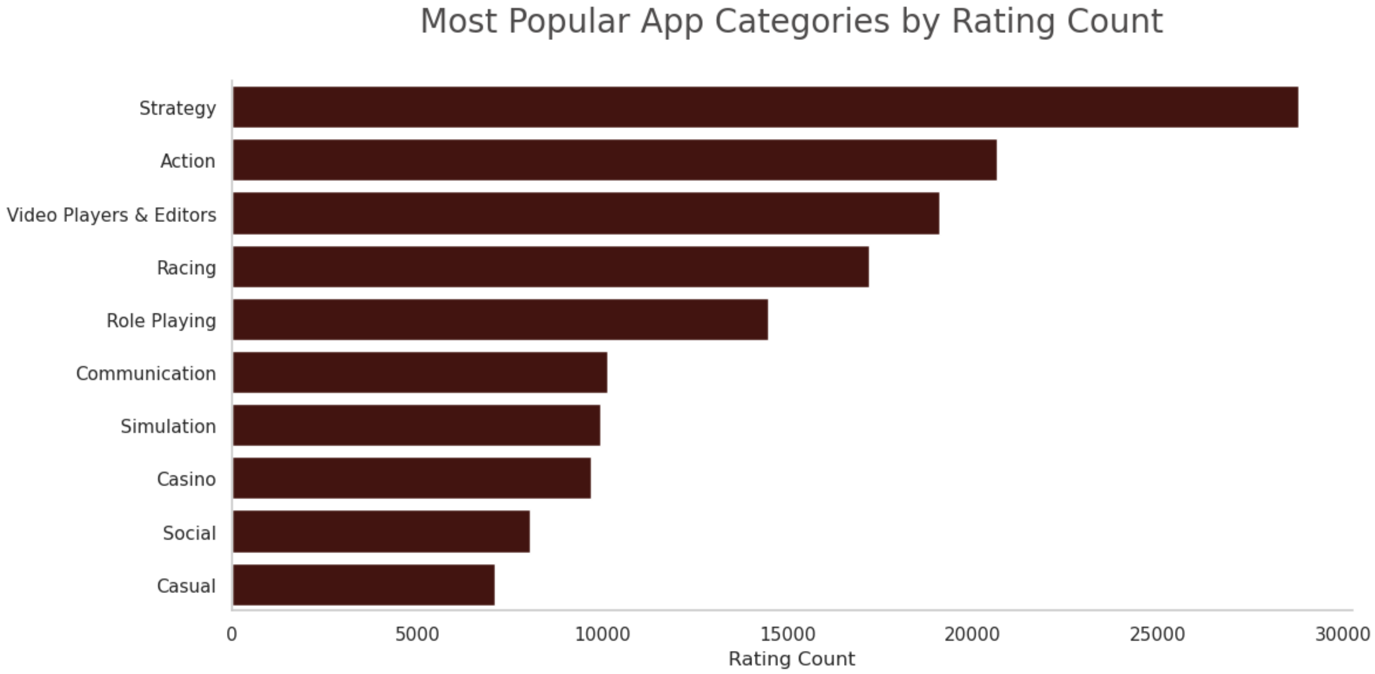
1. Which are the most popular app categories based on rating count or average number of installs?
2. How does the app's rating correlate with its rating count? Is there a positive relationship between the two?
3. What is the distribution of app ratings? Are most apps highly rated, or is there a wide range of ratings?
4. How does the app size (in megabytes) vary across different categories? Are certain categories associated with larger or smaller app sizes?
5. Are apps with in-app purchases more popular or highly rated compared to apps without in-app purchases?

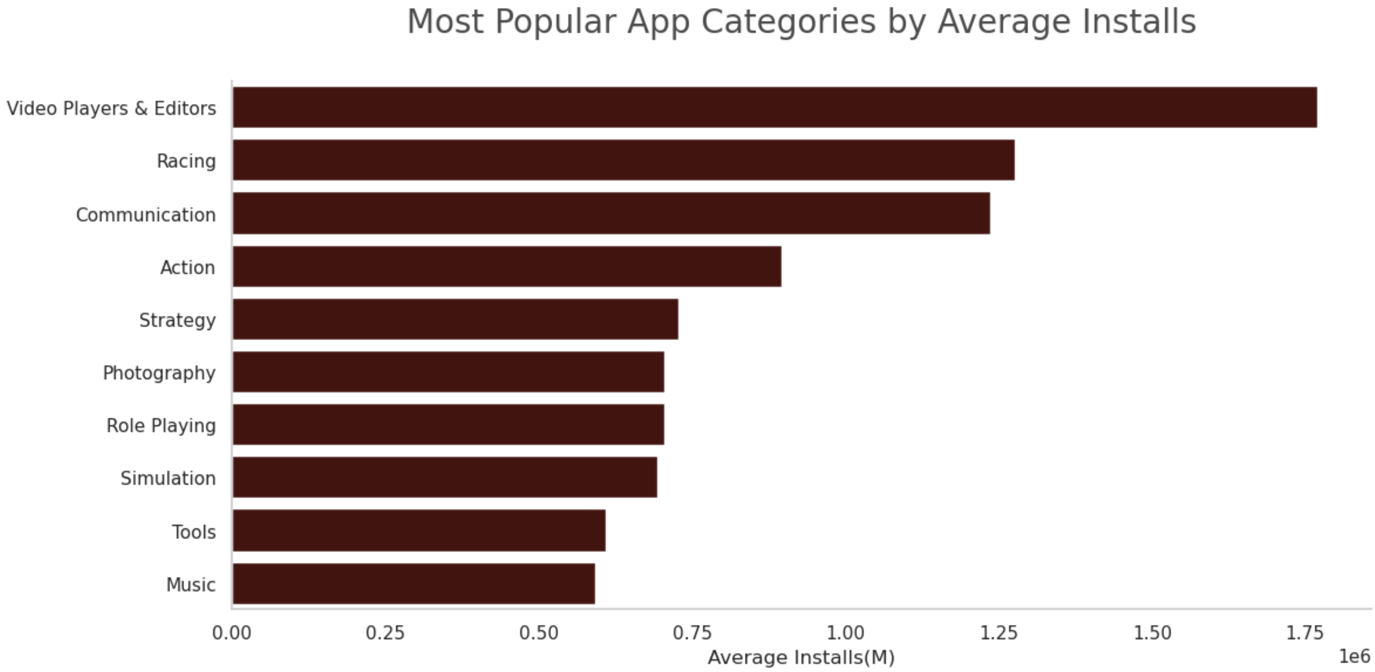
6. Is there a difference in ratings between ad-supported and non-ad-supported apps?
7. What is the trend of app releases over time? Are there specific year with a higher number of app releases?
8. How does the average app rating change over time?
9. Are apps with editor's choice designation more likely to have higher ratings or installs?
10. How many apps in the dataset have each specific minimum Android version requirement?
11. Are free apps more popular in terms of ratings compared to paid apps?
12. Are there any correlations between app attributes such as Rating, Rating Count, Minimum Android or Avg Installs?

1 - Which are the most popular app categories based on rating count or average number of installs?

In [265…

```
# Which are the most popular app categories based on rating count or average number of i
category_avg_rating_count = df.groupby('Category')['Rating Count'].mean()
category_avg_installs = df.groupby('Category')['Avg Installs'].mean()

most_popular_by_rating_count = category_avg_rating_count.sort_values(ascending=False)
most_popular_by_installs = category_avg_installs.sort_values(ascending=False)

plt.figure(figsize=f_size)
sns.barplot(x=most_popular_by_rating_count[:10].values, y=most_popular_by_rating_count[:
plt.ylabel('')
plt.xlabel('Rating Count')
plt.title('Most Popular App Categories by Rating Count', fontsize=title_size, color=titl
sns.despine()
plt.tight_layout()
plt.grid(False)
plt.show()

plt.figure(figsize=f_size)
sns.barplot(x=most_popular_by_installs[:10].values, y=most_popular_by_installs[:10].inde
plt.ylabel('')
plt.xlabel('Average Installs(M)')
plt.title('Most Popular App Categories by Average Installs', fontsize=title_size, color=
sns.despine()
plt.tight_layout()
plt.grid(False)
plt.show()
```

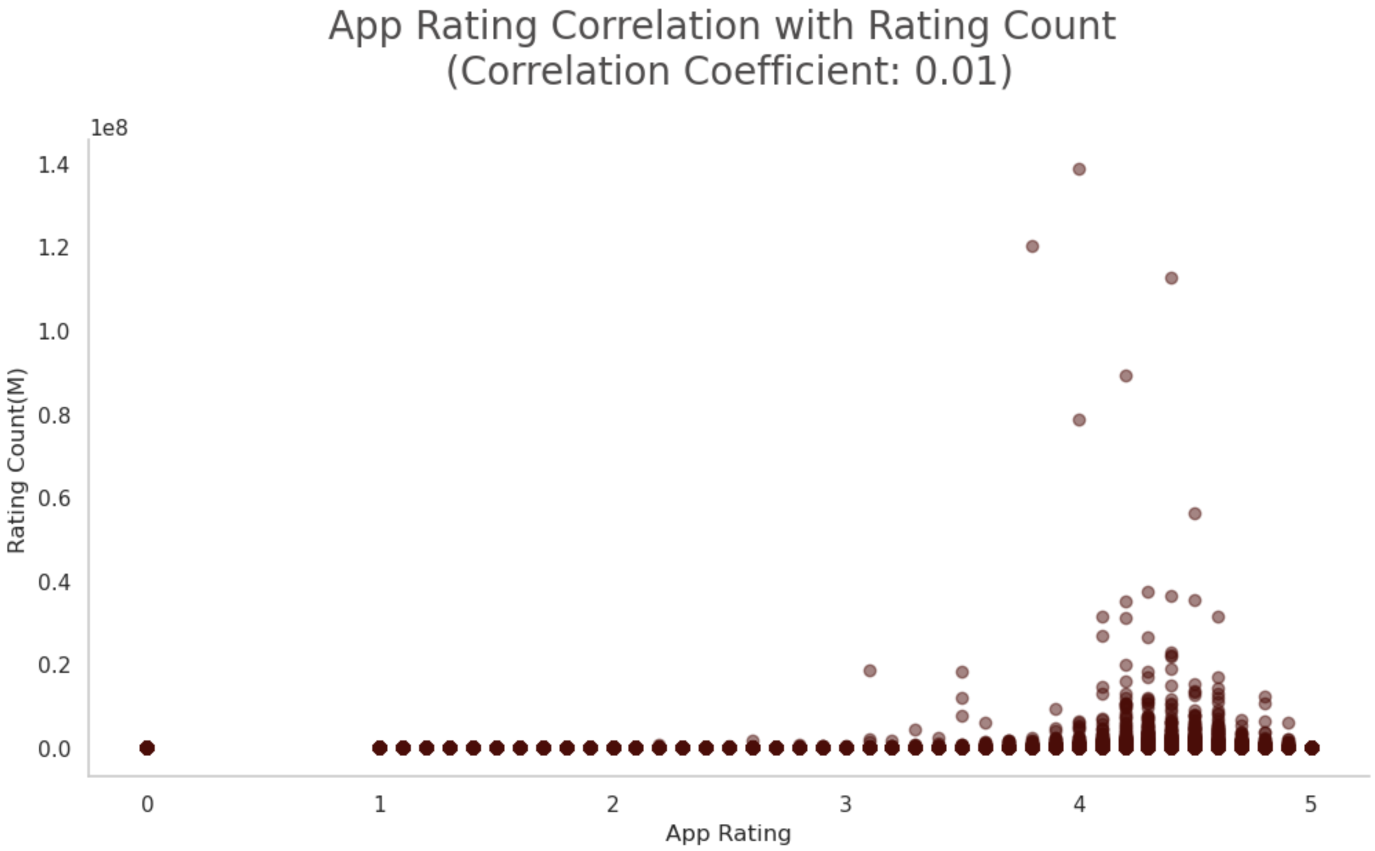




2 - How does the app's rating correlate with its rating count? Is there a positive relationship between the two?

In [266…

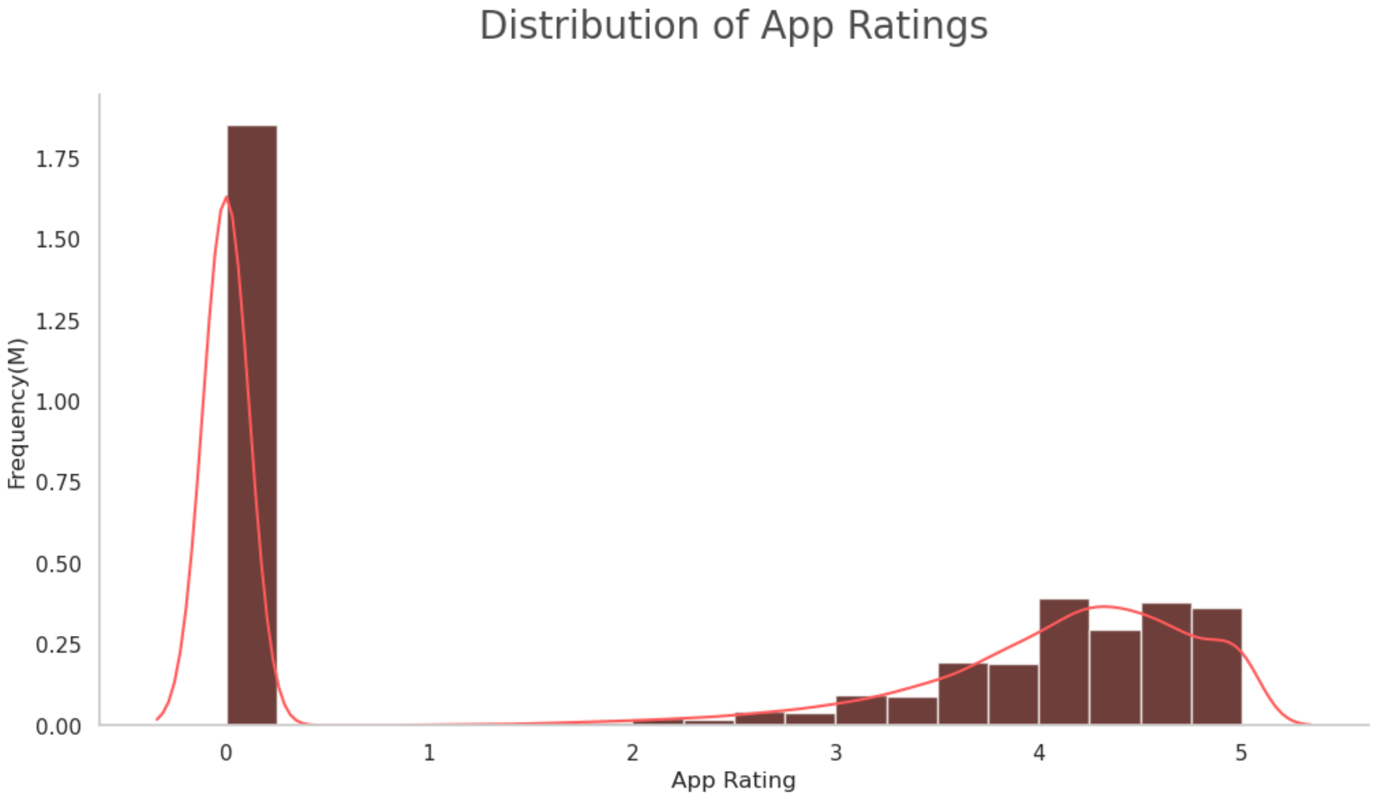
```
# How does the app's rating correlate with its rating count? Is there a positive relatio
from scipy.stats import pearsonr

correlation_coefficient, _ = pearsonr(df['Rating'], df['Rating Count'])

plt.figure(figsize=f_size)
plt.scatter(rating_data, rating_count_data, alpha=0.5, color=plt_color)

plt.xlabel('App Rating')
plt.ylabel('Rating Count(M)')
plt.title(f"App Rating Correlation with Rating Count \n(Correlation Coefficient: {correl
sns.despine(right=True, top=True)
plt.grid(False)
plt.show()
```



3 - What is the distribution of app ratings? Are most apps highly rated, or is there a wide range of ratings?

In [267…

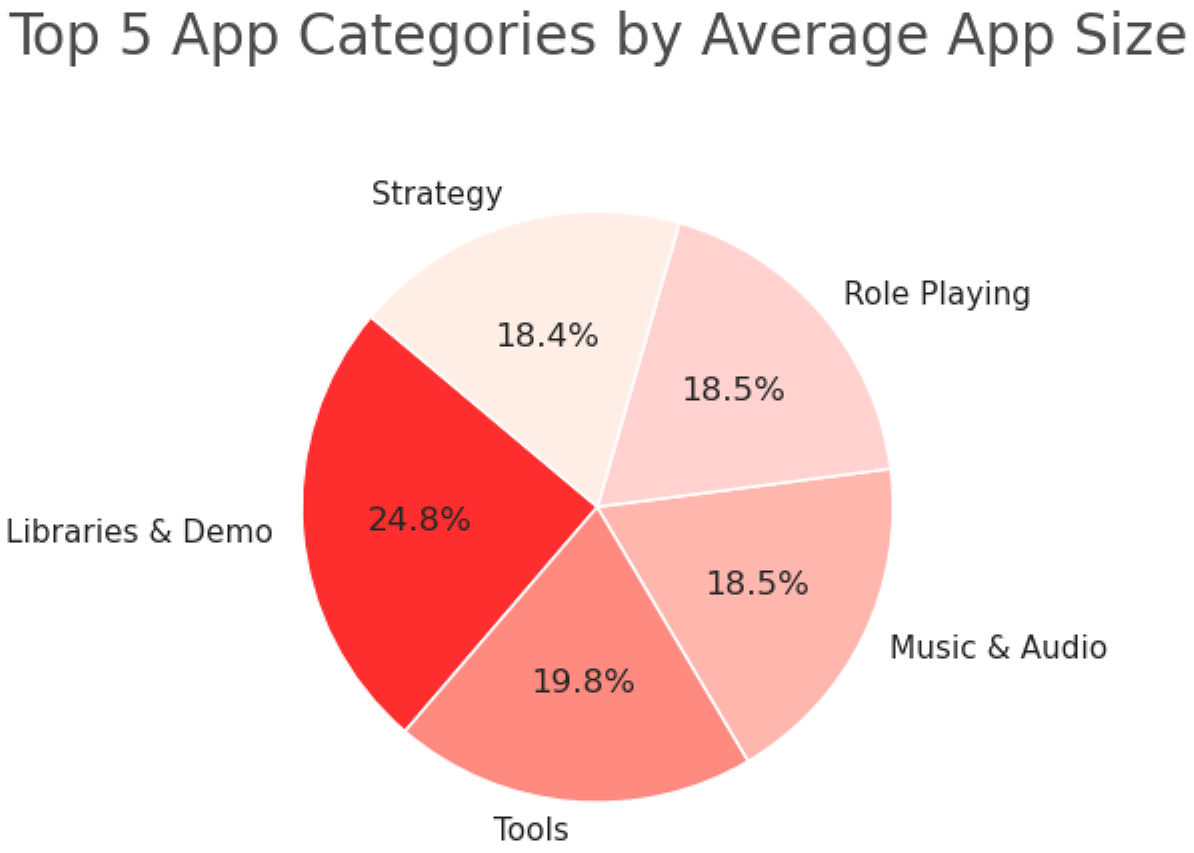
```
# What is the distribution of app ratings? Are most apps highly rated, or is there a wid
plt.figure(figsize=f_size)
plt.hist(df['Rating'], density=True, bins=20, color=plt_color, alpha=0.8)
sns.kdeplot(df['Rating'], color='#FF5C5C')

plt.xlabel('App Rating')
plt.ylabel('Frequency(M)')
plt.title('Distribution of App Ratings', fontsize=title_size, color=title_color, pad=tit
sns.despine(top=True, right=True)
plt.grid(False)
plt.show()
```



4 - How does the app size (in megabytes) vary across different categories? Are certain categories associated with larger app sizes?

In [268…

```
# How does the app size (in megabytes) vary across different categories? Are certain cat
colors = ['#FF2E2E', '#FF8A80', '#FFB6AD', '#FFD2D0', '#FFEEE6']

app_size_mean = df.groupby('Category')['Size(Megabyte)'].mean().reset_index().sort_value
top_5_categories = app_size_mean[:5]

plt.pie(top_5_categories['Size(Megabyte)'], labels=top_5_categories['Category'], autopct
plt.title('Top 5 App Categories by Average App Size', fontsize=title_size, color=title_c
plt.figure(figsize=f_size)
plt.show()
```

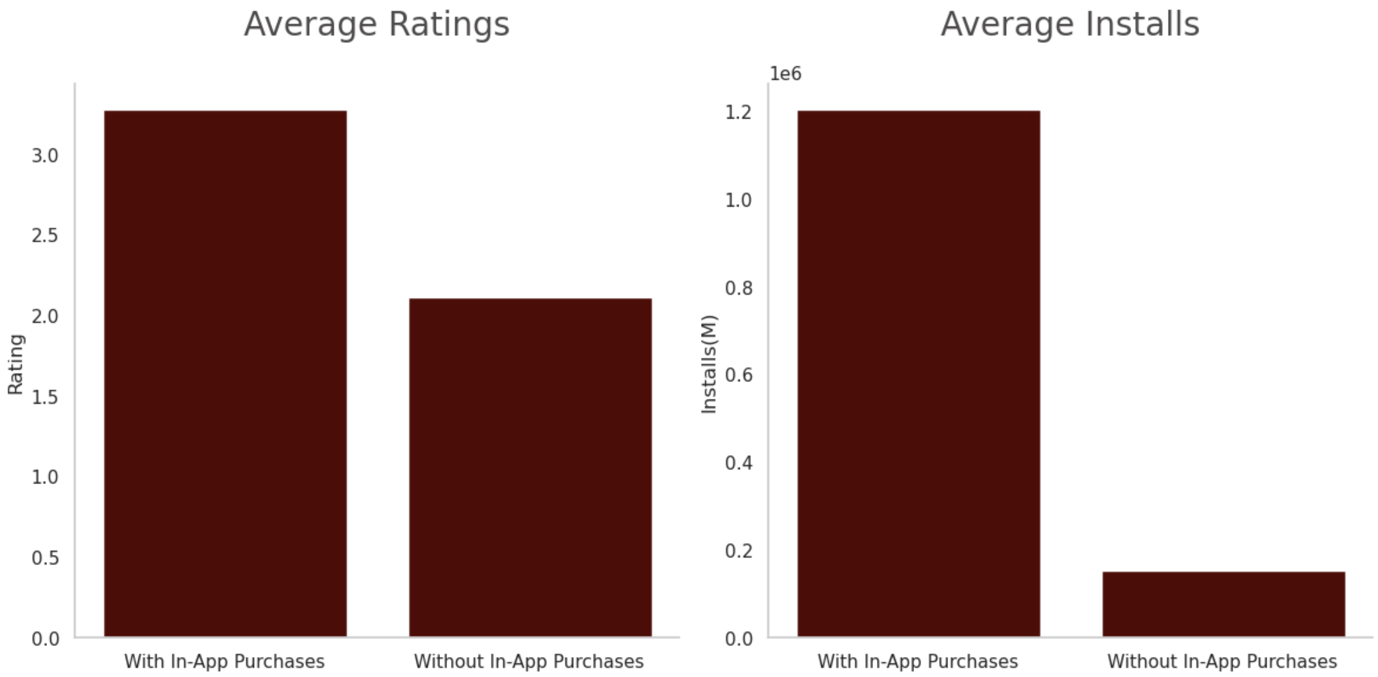


```
<Figure size 1200x600 with 0 Axes>
```

5 - Are apps with in-app purchases more popular or highly rated compared to apps without in-app purchases?

In [269…

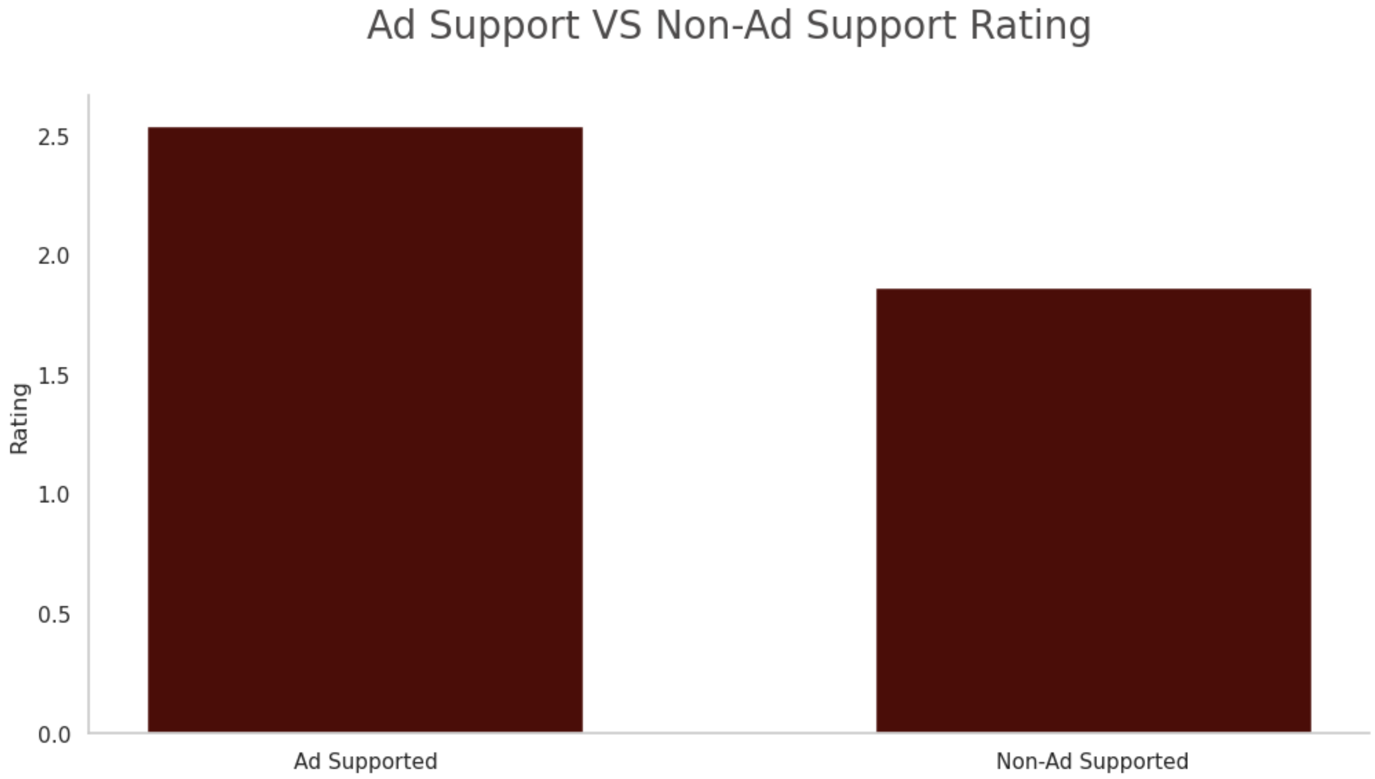
```
# Are apps with in-app purchases more popular or highly rated compared to apps without i
apps_with_in_app_purchases = df[df["In App Purchases"] == True]
apps_without_in_app_purchases = df[df["In App Purchases"] == False]

avg_rating_with_in_app_purchases = apps_with_in_app_purchases["Rating"].mean()
avg_rating_without_in_app_purchases = apps_without_in_app_purchases["Rating"].mean()

avg_installs_with_in_app_purchases = apps_with_in_app_purchases["Avg Installs"].mean()
avg_installs_without_in_app_purchases = apps_without_in_app_purchases["Avg Installs"].me

labels = ["With In-App Purchases", "Without In-App Purchases"]
avg_ratings = [avg_rating_with_in_app_purchases, avg_rating_without_in_app_purchases]
avg_installs = [avg_installs_with_in_app_purchases, avg_installs_without_in_app_purchase

plt.figure(figsize=f_size)

plt.subplot(1, 2, 1)
plt.bar(labels, avg_ratings, color=plt_color)
plt.title("Average Ratings", fontsize=title_size, color=title_color, pad=title_pad)
plt.ylabel("Rating")
sns.despine(top=True, right=True)
plt.grid(False)

plt.subplot(1, 2, 2)
plt.bar(labels, avg_installs, color=plt_color)
plt.title("Average Installs", fontsize=title_size, color=title_color, pad=title_pad)
plt.ylabel("Installs(M)")
sns.despine(top=True, right=True)
plt.tight_layout()
plt.grid(False)
plt.show()
```



6 - Is there a difference in ratings between ad-supported and non-ad-supported apps?

In [270…

```
# Is there a difference in ratings between ad-supported and non-ad-supported apps?
ad_supported_df = df[df['Ad Supported'] == True]
non_ad_supported_df = df[df['Ad Supported'] == False]

avg_rate_ad_supported_df = ad_supported_df['Rating'].mean()
avg_rate_non_ad_supported_df = non_ad_supported_df['Rating'].mean()

labels = ['Ad Supported', 'Non-Ad Supported']
plt.figure(figsize=f_size)
plt.bar(labels, [avg_rate_ad_supported_df, avg_rate_non_ad_supported_df], color=plt_colo
plt.title('Ad Support VS Non-Ad Support Rating', fontsize=title_size, color=title_color,
plt.ylabel("Rating")
sns.despine(top=True, right=True)
plt.grid(False)
plt.show()
```



7 - What is the trend of app releases over time? Are there specific year with a higher number of app releases?

In [271…

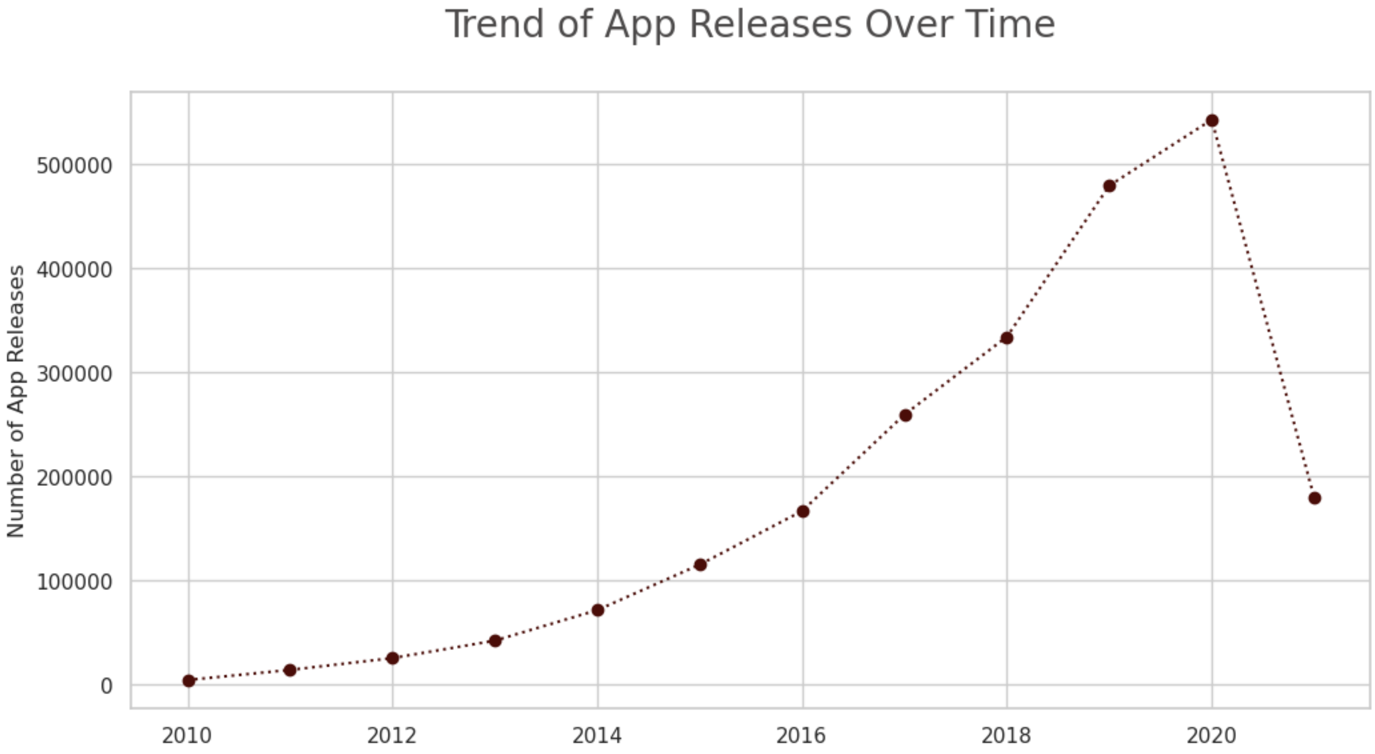
```
df["Released Date"]
```

Out[271]:

```
0          2020-02-26
1          2020-05-21
2          2019-08-09
3          2018-09-10
4          2020-02-21
              ...
2312938    2018-05-22
2312940    2018-01-17
2312941    2018-08-19
2312942    2016-08-01
2312943    2019-08-09
Name: Released Date, Length: 2235299, dtype: datetime64[ns]
```

In [272…

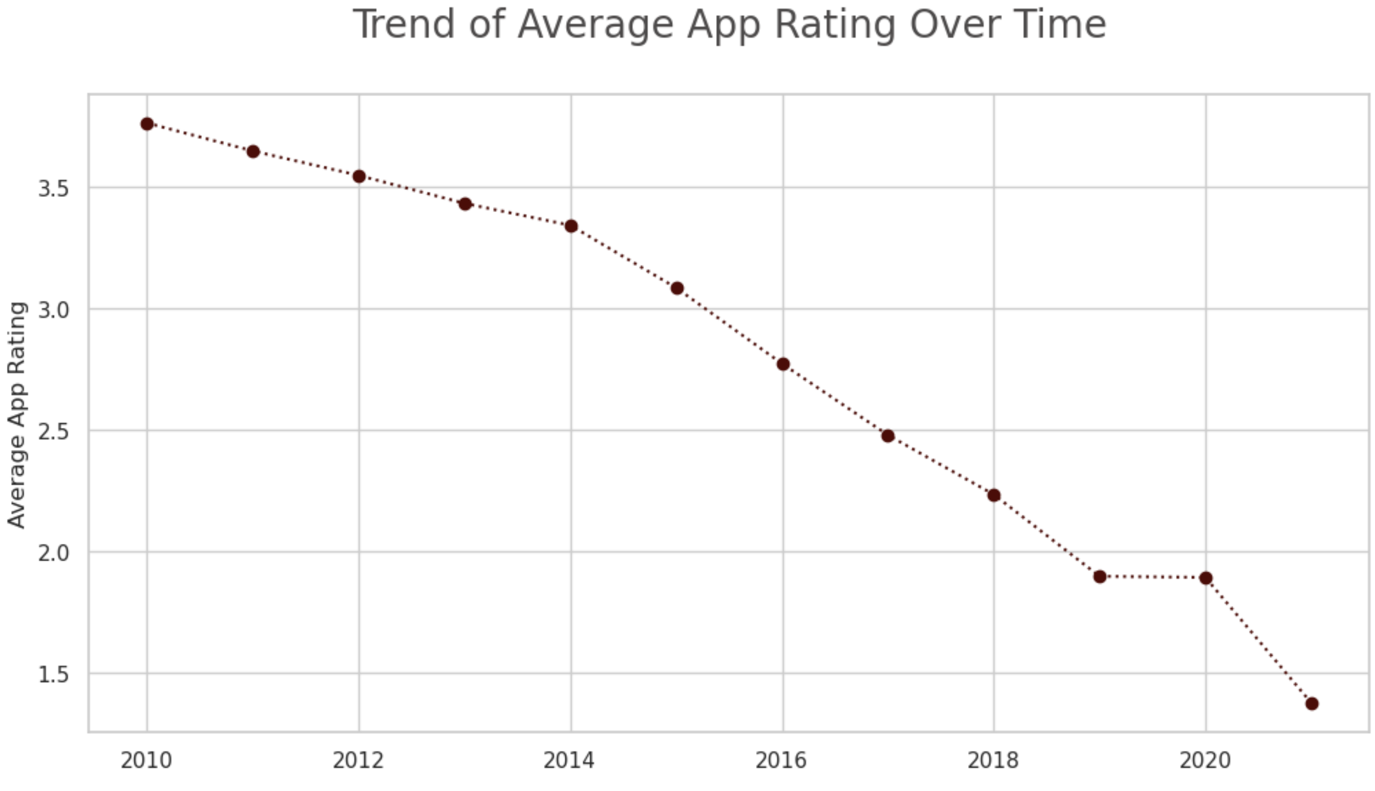
```
# What is the trend of app releases over time? Are there specific year with a higher num
df["Released Date"] = pd.to_datetime(df["Released Date"], errors='coerce')
df["Year"] = df["Released Date"].dt.year

app_releases_by_year = df.groupby(["Year"]).size().reset_index(name="Count")

plt.figure(figsize=f_size)
plt.plot(app_releases_by_year["Year"], app_releases_by_year["Count"], marker='o', linest

plt.xlabel("")
plt.ylabel("Number of App Releases")
plt.title("Trend of App Releases Over Time", fontsize=title_size, color=title_color, pad
plt.grid(True)
plt.show()
```



8 - How does the average app rating change over time?

In [273…

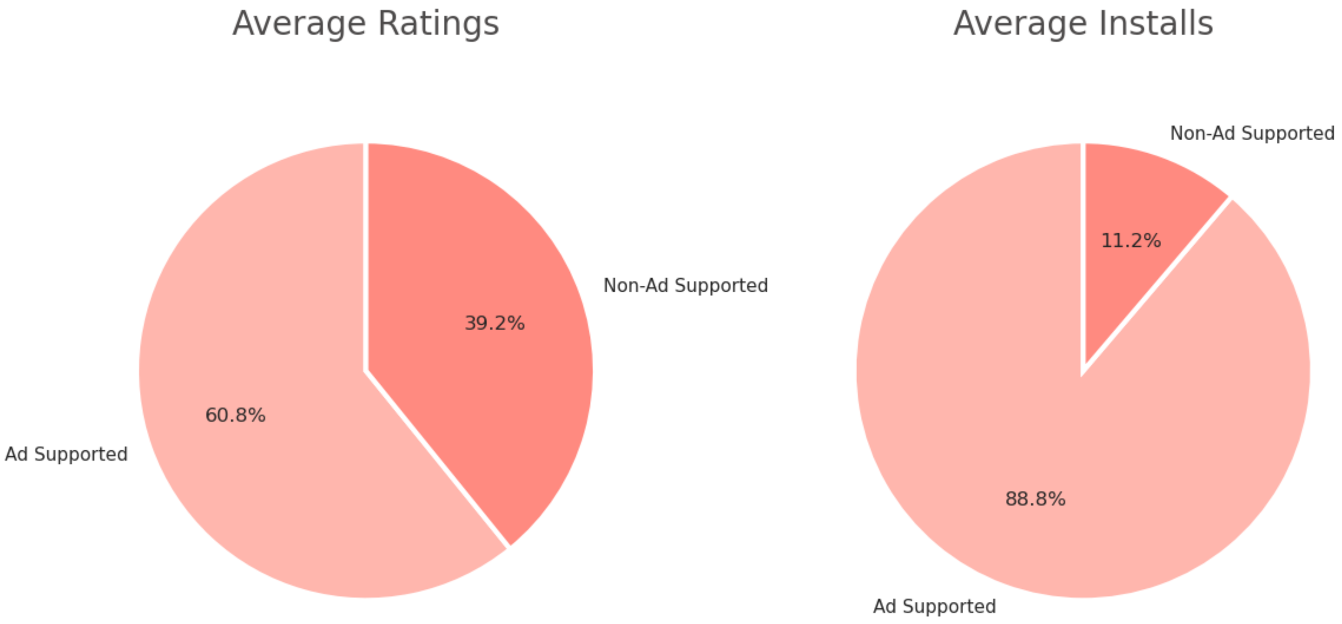
```
# How does the average app rating change over time?
average_rating_by_year = df.groupby(["Year"])["Rating"].mean().reset_index(name="Average

plt.figure(figsize=f_size)
plt.plot(average_rating_by_year["Year"], average_rating_by_year["Average Rating"], marke

plt.xlabel("")
plt.ylabel("Average App Rating")
plt.title("Trend of Average App Rating Over Time", fontsize=title_size, color=title_colo
plt.grid(True)
plt.show()
```



9 - Are apps with editor's choice designation more likely to have higher ratings or installs?

In [274…

```
# Are apps with editor's choice designation more likely to have higher ratings or instal
apps_with_in_app_purchases = df[df["In App Purchases"] == True]
apps_without_in_app_purchases = df[df["In App Purchases"] == False]

avg_rating_with_in_app_purchases = apps_with_in_app_purchases["Rating"].mean()
avg_rating_without_in_app_purchases = apps_without_in_app_purchases["Rating"].mean()

avg_installs_with_in_app_purchases = apps_with_in_app_purchases["Avg Installs"].mean()
avg_installs_without_in_app_purchases = apps_without_in_app_purchases["Avg Installs"].me

sns.set(style="whitegrid")
plt.figure(figsize=f_size)

plt.subplot(1, 2, 1)
plt.pie([avg_rating_with_in_app_purchases, avg_rating_without_in_app_purchases],
        labels=labels,
        colors=["#FFB6AD", "#FF8A80"],
        autopct="%.1f%%",
        wedgeprops={"linewidth": 3},
        startangle=90)
plt.title("Average Ratings", fontsize=title_size, color=title_color, pad=title_pad)

plt.subplot(1, 2, 2)
plt.pie([avg_installs_with_in_app_purchases, avg_installs_without_in_app_purchases],
        labels=labels,
        colors=["#FFB6AD", "#FF8A80"],
        autopct="%.1f%%",
        wedgeprops={"linewidth": 3},
        startangle=90)
plt.title("Average Installs", fontsize=title_size, color=title_color, pad=title_pad)

plt.tight_layout()
plt.show()
```

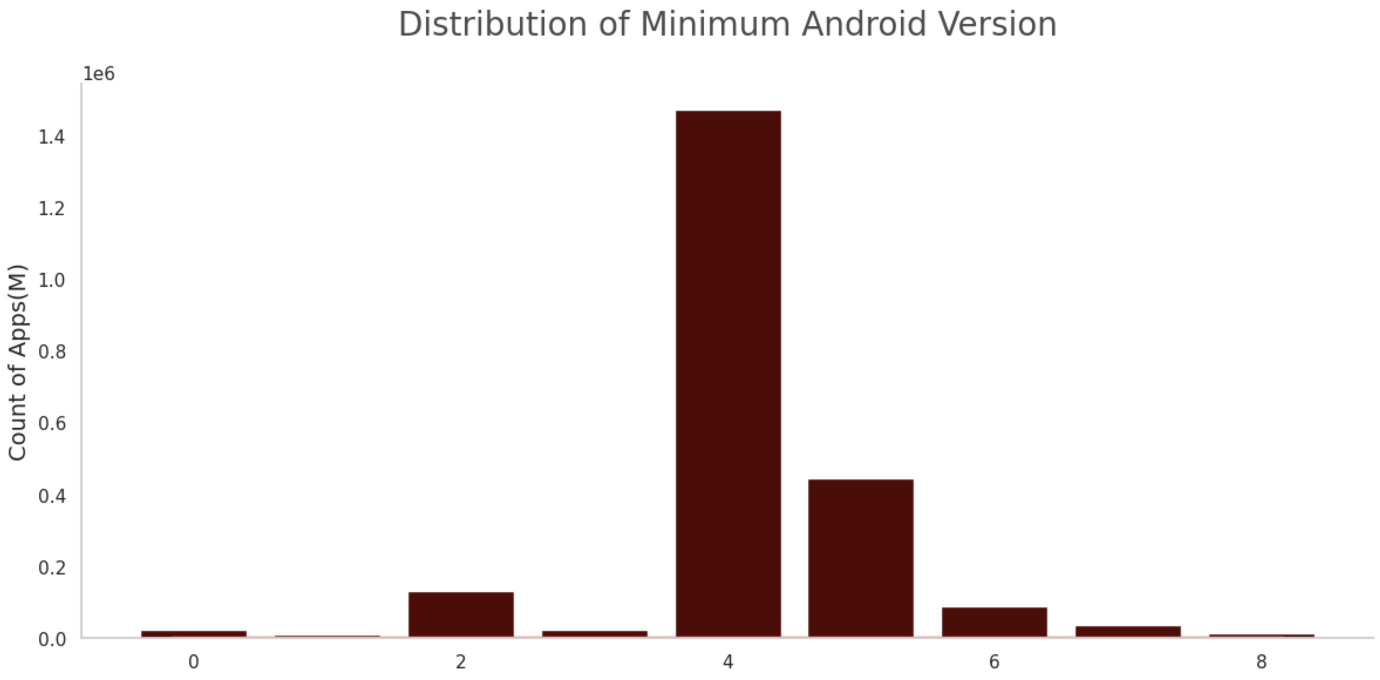


10 - How many apps in the dataset have each specific minimum Android version requirement?

In [275…

```
# How many apps in the dataset have each specific minimum Android version requirement?
```

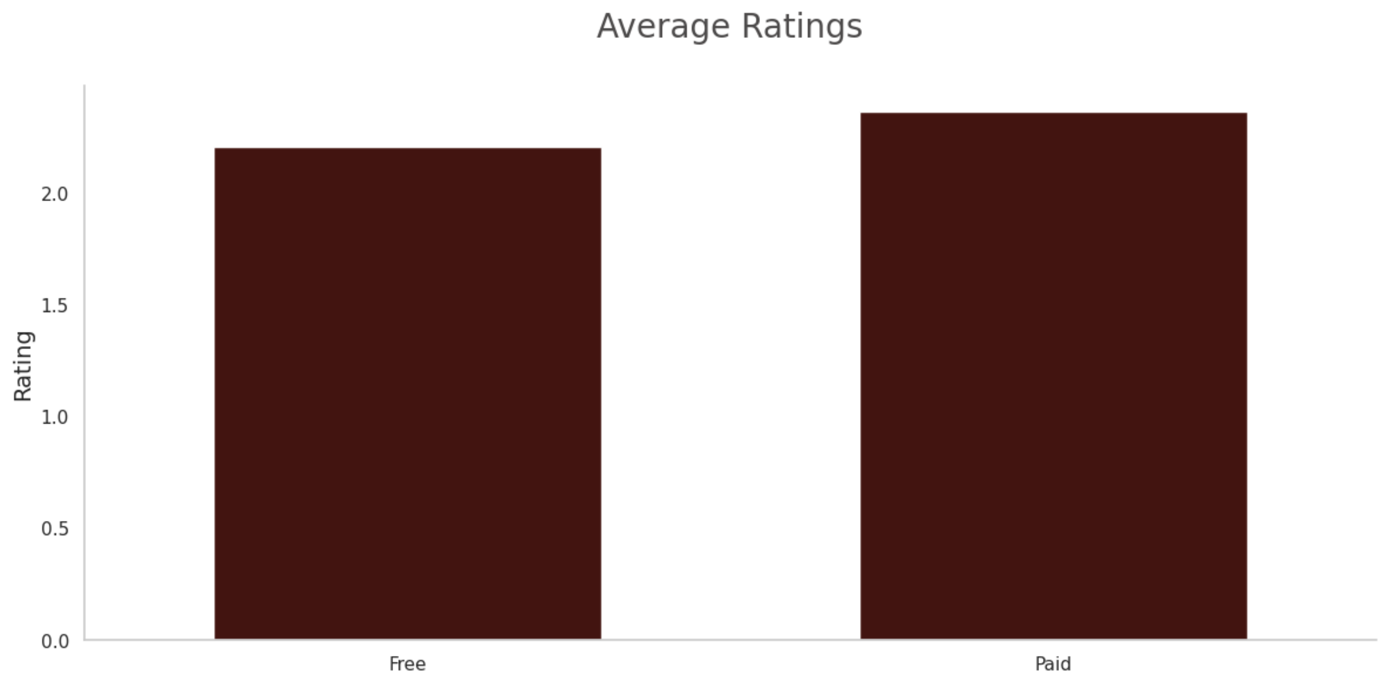
```
android_version_counts = df["Minimum Android"].value_counts().sort_index()

plt.figure(figsize=f_size)
plt.bar(android_version_counts.index, android_version_counts.values, color=plt_color)
sns.kdeplot(df["Minimum Android"], color='#FF5C5C')

plt.title("Distribution of Minimum Android Version", fontsize=title_size, color=title_co
plt.xlabel("")
plt.ylabel("Count of Apps(M)", fontsize=14)
sns.despine(top=True, right=True)
plt.grid(False)
plt.tight_layout()
plt.show()
```



11 - Are free apps more popular in terms of ratings compared to paid apps?

In [276…

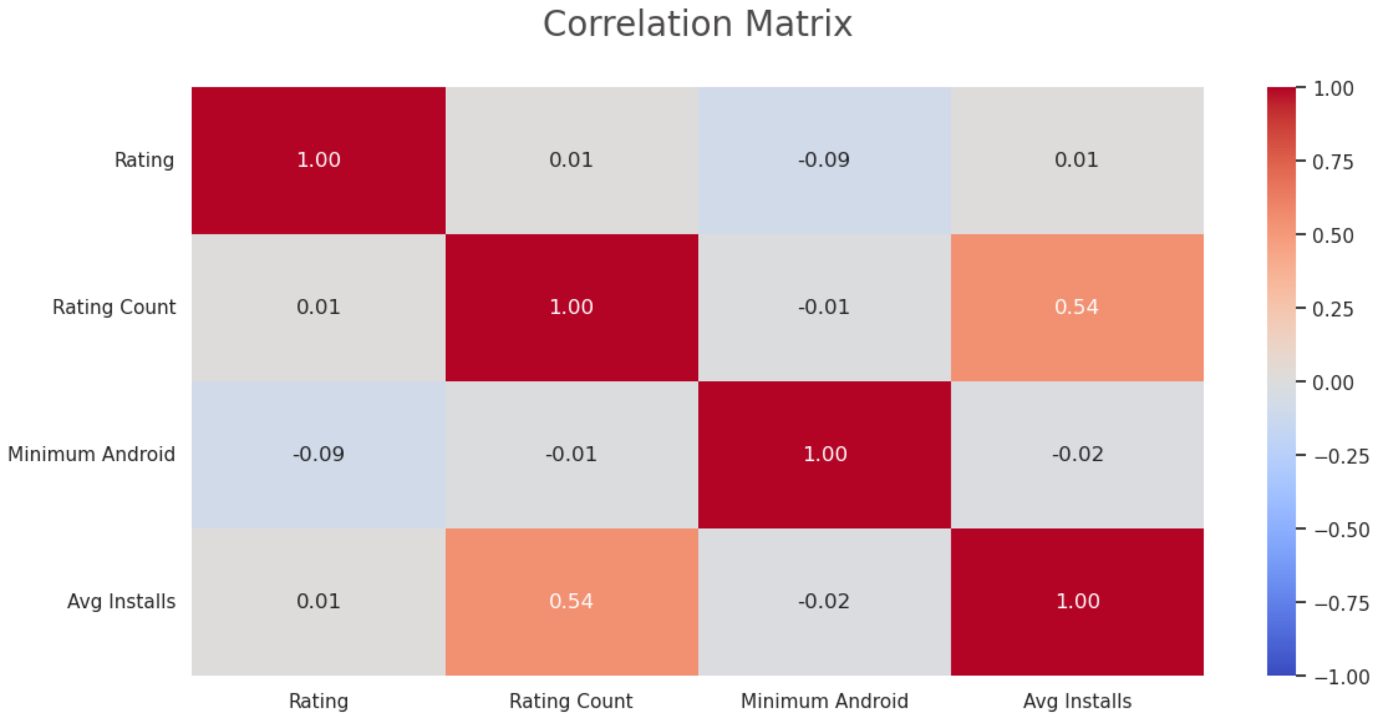
```
# Are free apps more popular in terms of ratings compared to paid apps?
free_apps = df[df["Free"] == True]
paid_apps = df[df["Free"] == False]

avg_rating_free = free_apps["Rating"].mean()
avg_rating_paid = paid_apps["Rating"].mean()

avg_installs_free = free_apps["Avg Installs"].mean()
avg_installs_paid = paid_apps["Avg Installs"].mean()

sns.set(style="whitegrid")
plt.figure(figsize=f_size)
sns.barplot(x=["Free", "Paid"], y=[avg_rating_free, avg_rating_paid], color=plt_color, w
plt.title("Average Ratings", fontsize=title_size, color=title_color, pad=title_pad)
plt.ylabel("Rating", fontsize=14)
sns.despine(top=True, right=True)
plt.tight_layout()
plt.grid(False)
plt.show()
```



12 - Are there any correlations between app attributes such as Rating, Rating Count, Minimum Android or Avg Installs?

In [277…

```python
# Are there any correlations between app attributes such as Rating, Rating Count, Minimu
selected_columns = ["Rating", "Rating Count", "Minimum Android", "Avg Installs"]
correlation_matrix = df[selected_columns].corr()

plt.figure(figsize=f_size)
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", vmin=-1, vmax=1)
plt.title("Correlation Matrix", fontsize=title_size, color=title_color, pad=title_pad)
plt.tight_layout()
plt.show()
```



4 - Summary of Key Findings

1. **Most Popular App Categories**: Based on rating count, the most popular app categories are `Strategy, Action, Video Players & Editors`. On the other hand, based on the average number of installs, the most popular app categories are `Video Players & Editors, Racing, Communication`.

2. **Correlation between Rating and Rating Count**: There is a `weak` correlation between an app's rating and its rating count.

3. **Distribution of App Ratings**: The distribution of app ratings is `skewed to the right`. `most rating between 0-1`.

4. **App Size Variation Across Categories**: App size `does not vary significantly across different categories`. However, there is `one category` that is slightly larger than the others.

5. **In-App Purchases Impact on Popularity and Ratings**: Apps with in-app purchases are `more popular` and `have higher ratings` compared to apps without in-app purchases.

6. **Difference in Ratings Between Ad-Supported and Non-Ad-Supported Apps**: `Ad-supported apps have greater ratings than non-supported apps`.

7. **Trend of App Releases Over Time**: The trend of app releases over time has been `increasing`, but since the `beginning of 2020`, it has `dropped down`.

8. **Average App Rating Over Time**: The average app rating `decreases` over time.

9. **Editor's Choice Impact on Ratings and Installs**: Apps with the "Editor's Choice" designation `have higher ratings and more installs` compared to apps without this designation.

10. **Distribution of Minimum Android Version Requirements**: The minimum Android version requirements for apps in the dataset range from 1 to 8. `The most common minimum Android version requirement among apps is 4`.

11. **Popularity of Free Apps vs. Paid Apps**: Ratings for free apps and paid apps are `nearly equal`, but the number of `paid apps is slightly higher` than the number of free apps.

12. **Correlations Between App Attributes**: There are `positive correlations between the "Avg Installs" and "Rating Count" attributes`, indicating that apps with higher average installs tend to have more ratings. On the other hand, there is a `negative correlation between the "Minimum Android" version and the "Rating" attribute`. This suggests that apps with lower minimum Android versions required to run might receive higher ratings.

In conclusion, the exploratory data analysis (EDA) of the app dataset reveals interesting insights into the app categories, ratings, installs, sizes, and other attributes. These findings can be valuable for understanding user preferences, app trends, and the factors that contribute to app popularity and success.

Whoami

I'm Mohammed Nashaat, a data enthusiast passionate about exploring and analyzing datasets. You can find some of my data analysis projects on GitHub.

Feel free to check out my portfolio to see some of the interesting projects I've worked on.