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

Step 1: Load and Explore the Dataset

```
dataset = pd.read_csv("googleplaystore.csv")
```

Step 2: Data Cleaning:

1. Convert data types for better analysis.

```
print(dataset.dtypes)
```

```
App                object
Category           object
Rating            float64
Reviews            object
Size               object
Installs           object
Type               object
Price              object
Content Rating     object
Genres             object
Last Updated       object
Current Ver        object
Android Ver        object
dtype: object
```

```
dataset['Installs'] = dataset['Installs'].str.replace("Free", "0")

dataset['Installs'] = dataset['Installs'].str.replace(r"[+,]", "",
regex=True).astype(float)

dataset['Price'] = pd.to_numeric(dataset['Price'], errors='coerce') #
converted into numeric

dataset['Last Updated'] = pd.to_datetime(dataset['Last Updated'],
errors='coerce') #converted into date type
```

2. Handle missing values.

```
dataset.isna().sum()  #count missing data
```

```
App                  0
Category             0
Rating            1474
Reviews              0
Size                 0
Installs             0
Type                 1
Price              801
Content Rating       1
Genres               0
Last Updated         1
Current Ver          8
Android Ver          3
dtype: int64
```

```
dataset['Rating'].fillna(dataset['Rating'].median(),inplace =True)
#filled null values with mdedian

dataset['Price'].fillna(dataset['Rating'].median(),inplace =True)
#filled null values with mdedian

mode_cversion = dataset['Current Ver'].mode()[0] #find most frequency
in current version column

dataset['Current Ver'].fillna(mode_cversion, inplace =True) # filled
missing values with most frequency column

mode_andr_version = dataset['Android Ver'].mode()[0] #find most
frequency in android version column

dataset['Android Ver'].fillna(mode_andr_version, inplace =True) #
filled missing values with most frequency column

mode_content_rating = dataset['Content Rating'].mode()[0]

dataset['Content Rating'].fillna(mode_content_rating,inplace=True)
#filled missing values with most frequency column

dataset['Type'].fillna(dataset['Type'].mode()[0],inplace=True) #filled
missing values with most frequency column

dataset['Last Updated'].fillna(dataset['Last Updated'].mode()
[0],inplace=True) #filled missing values with most frequency column

dataset.isna().sum()
```

```
App                0
Category           0
Rating             0
Reviews            0
Size               0
Installs           0
```

```
Type              0
Price             0
Content Rating    0
Genres            0
Last Updated      0
Current Ver       0
Android Ver       0
dtype: int64
```

3. Remove duplicates

```
dataset.duplicated().sum() #gives number of duplicate data
```

```
483
```

```
dataset =dataset.drop_duplicates(keep="first") # keep the first occurrence
```

```
dataset.duplicated().sum() #check its all clear now
```

```
0
```

```
dataset.dtypes
```

```
App                        object
Category                   object
Rating                    float64
Reviews                    object
Size                       object
Installs                  float64
Type                       object
Price                     float64
Content Rating             object
Genres                     object
Last Updated      datetime64[ns]
Current Ver                object
Android Ver                object
dtype: object
```

4. Detect Outliers and Remove Them

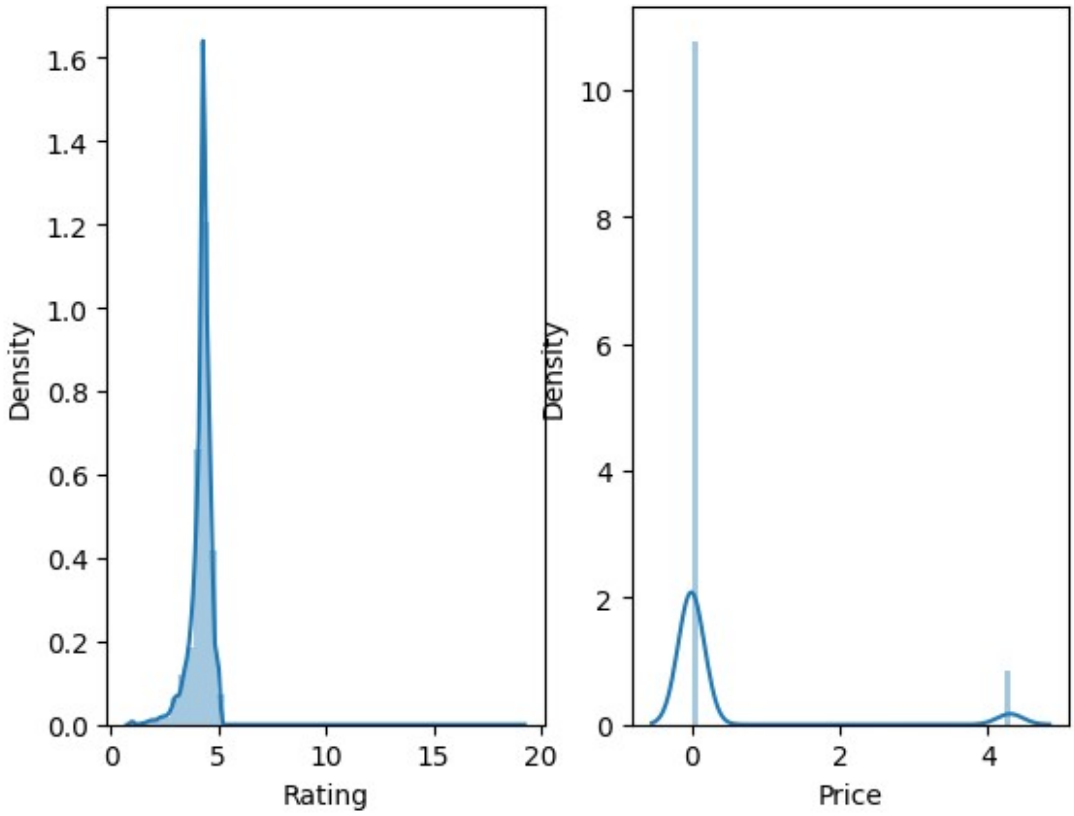
```
dataset.describe()
```

```
             Rating      Installs         Price                    Last Updated
count  10358.000000  1.035800e+04  10358.000000                           10358
mean       4.205165  1.415639e+07      0.317996  2017-11-14 10:01:41.486773504
```

```
min        1.000000  0.000000e+00      0.000000             2010-05-21
00:00:00
25%        4.100000  1.000000e+03      0.000000             2017-09-03
00:00:00
50%        4.300000  1.000000e+05      0.000000             2018-05-20
00:00:00
75%        4.500000  1.000000e+06      0.000000             2018-07-19
00:00:00
max       19.000000  1.000000e+09      4.300000             2018-08-08
00:00:00
std        0.506868  8.023580e+07      1.125337
NaN
```

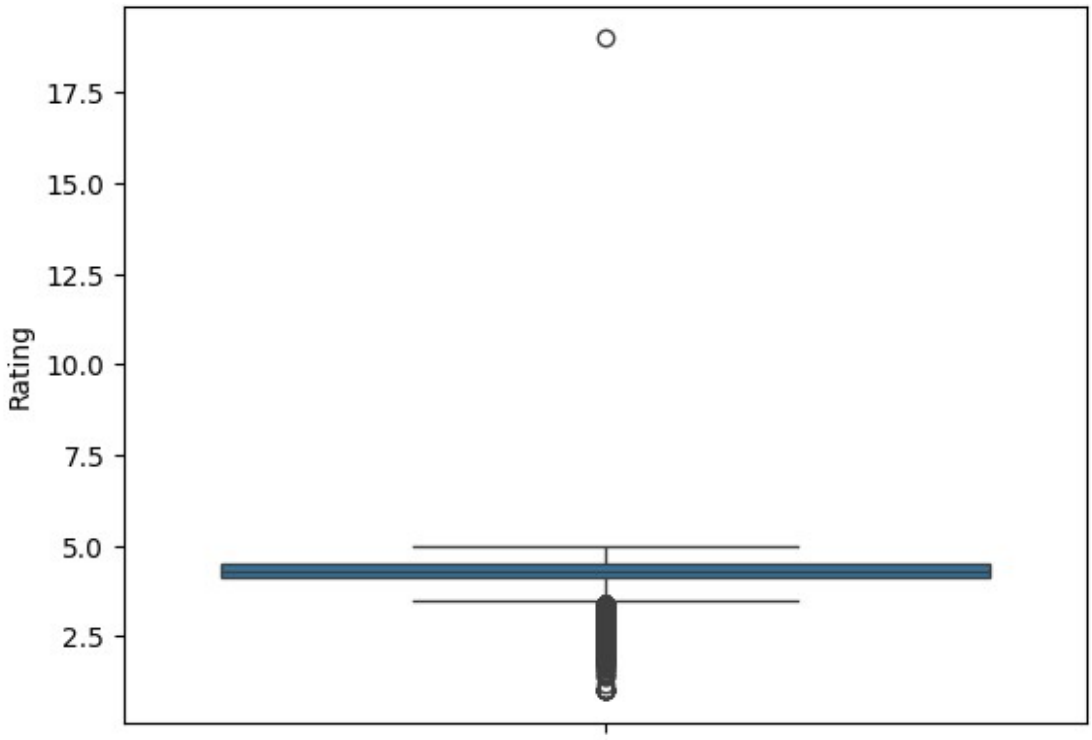
```
plt.subplot(1,2,1)
sns.distplot(dataset['Rating'])
plt.subplot(1,2,2)
sns.distplot(dataset['Price'])

plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_12092\2011596515.py:2:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataset['Rating'])
C:\Users\HP\AppData\Local\Temp\ipykernel_12092\2011596515.py:4:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataset['Price'])
```




```
sns.boxplot(dataset["Rating"]) # viewing outlier usign boxplot for
more clarity
```

```
<Axes: ylabel='Rating'>
```




```
sns.boxplot(dataset['Price'])
```

```
<Axes: ylabel='Price'>
```

Lets understand and handle RATING column first for Outlier detection

```
dataset_backup=dataset.copy() # creating backup file here

(dataset['Rating']>5).sum() # only one value is greater than 5 and its outlier, we will remove it
```

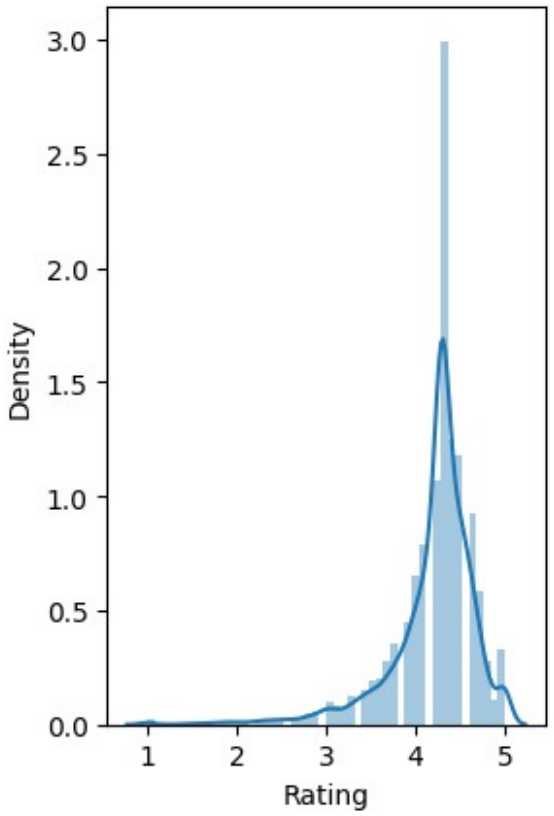
```
1
```

```
dataset = dataset[dataset["Rating"] <= 5]

dataset['Rating'].describe()
```

```
count    10357.000000
mean         4.203737
std          0.485594
min          1.000000
25%          4.100000
50%          4.300000
75%          4.500000
max          5.000000
Name: Rating, dtype: float64
```

```
plt.subplot(1,2,1)
sns.distplot(dataset['Rating'])
plt.show()
```

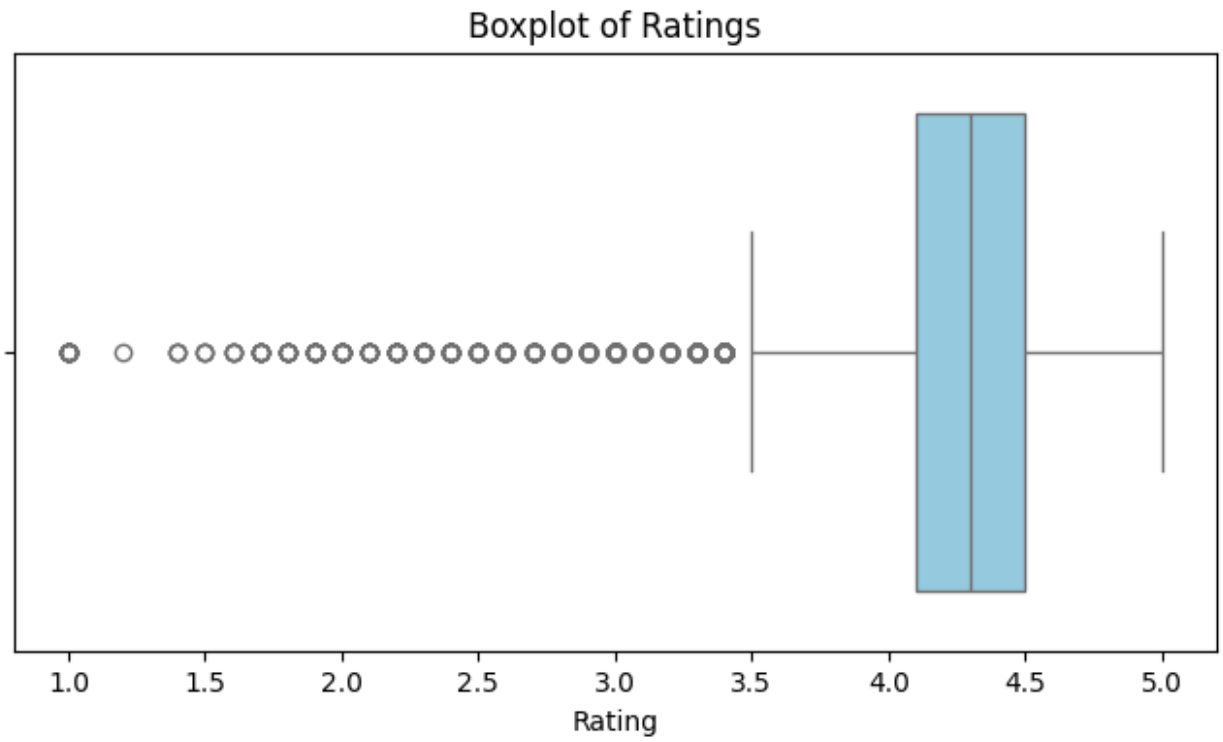
```
C:\Users\HP\AppData\Local\Temp\ipykernel_12092\3074010019.py:2:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataset['Rating'])
```




```
dataset['Rating'].describe()
```

```
count    10357.000000
mean         4.203737
std          0.485594
min          1.000000
25%          4.100000
50%          4.300000
75%          4.500000
max          5.000000
Name: Rating, dtype: float64
```

```
plt.figure(figsize=(8, 4))
sns.boxplot(x=dataset["Rating"], color="skyblue")
plt.title("Boxplot of Ratings")
plt.show()
```



These are not outliers since these are Rating between 1 to 5

Now understand the PRICE column for Outlier detection

```
dataset['Price'].describe()
```

```
count    10357.000000
mean         0.317611
std          1.124710
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max          4.300000
Name: Price, dtype: float64
```

Capping Outliers using Percentile

```
upper_cap = dataset["Price"].quantile(0.95)  # Set upper limit at 95th percentile
dataset["Price"] = dataset["Price"].clip(upper=upper_cap)  # Cap values above 95th percentile

dataset['Price'].describe()
```

```
count    10357.000000
mean         0.317611
std          1.124710
```

```
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max           4.300000
Name: Price, dtype: float64
```

Step 3. Business Questions for Analysis

1. Univariate Analysis (Non-Graphical)

Q1: What is the average rating of apps on the Play Store?

```
avg_rating =dataset['Rating'].mean()
print(f"average rating of apps on the Play Store : {avg_rating:.2f}")
```

```
average rating of apps on the Play Store : 4.20
```

Q2: What percentage of apps are free vs paid?

```
free_apps = dataset[dataset["Type"] == "Free"].shape[0]
paid_apps = dataset[dataset["Type"] == "Paid"].shape[0]
total_apps = dataset.shape[0]

free_percentage = (free_apps / total_apps) * 100
paid_percentage = (paid_apps / total_apps) * 100

print(f"Free Apps: {free_percentage:.2f}%")
print(f"Paid Apps: {paid_percentage:.2f}%")
```

```
Free Apps: 92.61%
Paid Apps: 7.39%
```

Q3: What is the most common app category?

```
most_common_category = dataset["Category"].value_counts()

most_common_category
```

```
Category
FAMILY                 1943
GAME                   1121
TOOLS                   843
BUSINESS                427
MEDICAL                 408
PRODUCTIVITY            407
PERSONALIZATION         388
```

```
LIFESTYLE                373
COMMUNICATION            366
FINANCE                  360
SPORTS                   351
PHOTOGRAPHY              322
HEALTH_AND_FITNESS       306
SOCIAL                   280
NEWS_AND_MAGAZINES       264
TRAVEL_AND_LOCAL         237
BOOKS_AND_REFERENCE      230
SHOPPING                 224
DATING                   196
VIDEO_PLAYERS            175
MAPS_AND_NAVIGATION      137
EDUCATION                130
FOOD_AND_DRINK           124
ENTERTAINMENT            111
AUTO_AND_VEHICLES         85
LIBRARIES_AND_DEMO        85
WEATHER                   82
HOUSE_AND_HOME            80
ART_AND_DESIGN            65
EVENTS                    64
PARENTING                 60
COMICS                    60
BEAUTY                    53
Name: count, dtype: int64
```

```
most_common_category = dataset["Category"].value_counts().idxmax()
```

```
print(f"The most common category is : {most_common_category}")
```

```
The most common category is : FAMILY
```

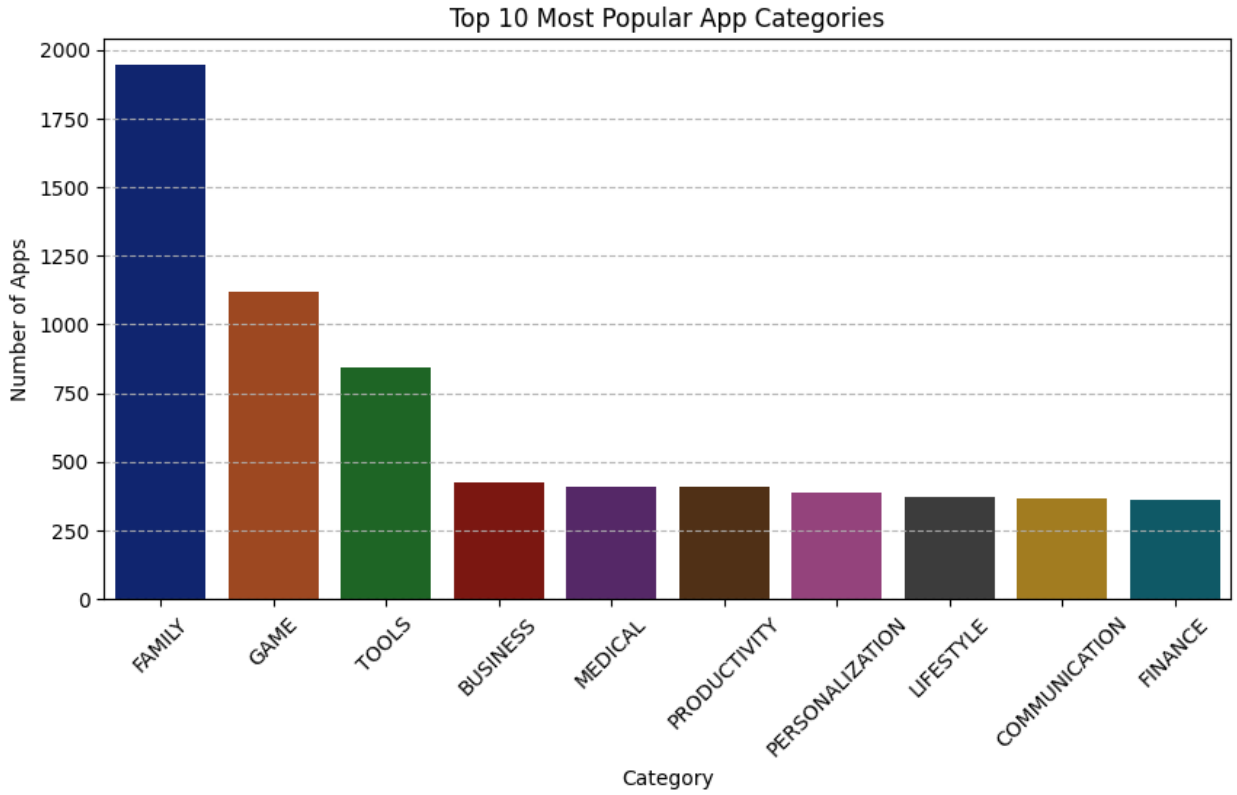
2. Univariate Visualizations (Categorical Variables)

Q4: Which app category has the highest number of apps?

```
plt.figure(figsize=(10,5))
sns.barplot(x=top_categories.index , y=top_categories.values,
palette='dark')
plt.xticks(rotation=45)
plt.title("Top 10 Most Popular App Categories")
plt.xlabel("Category")
plt.ylabel("Number of Apps")
```

```
plt.grid(axis='y', linestyle='--', alpha=0.9)
plt.show()
```

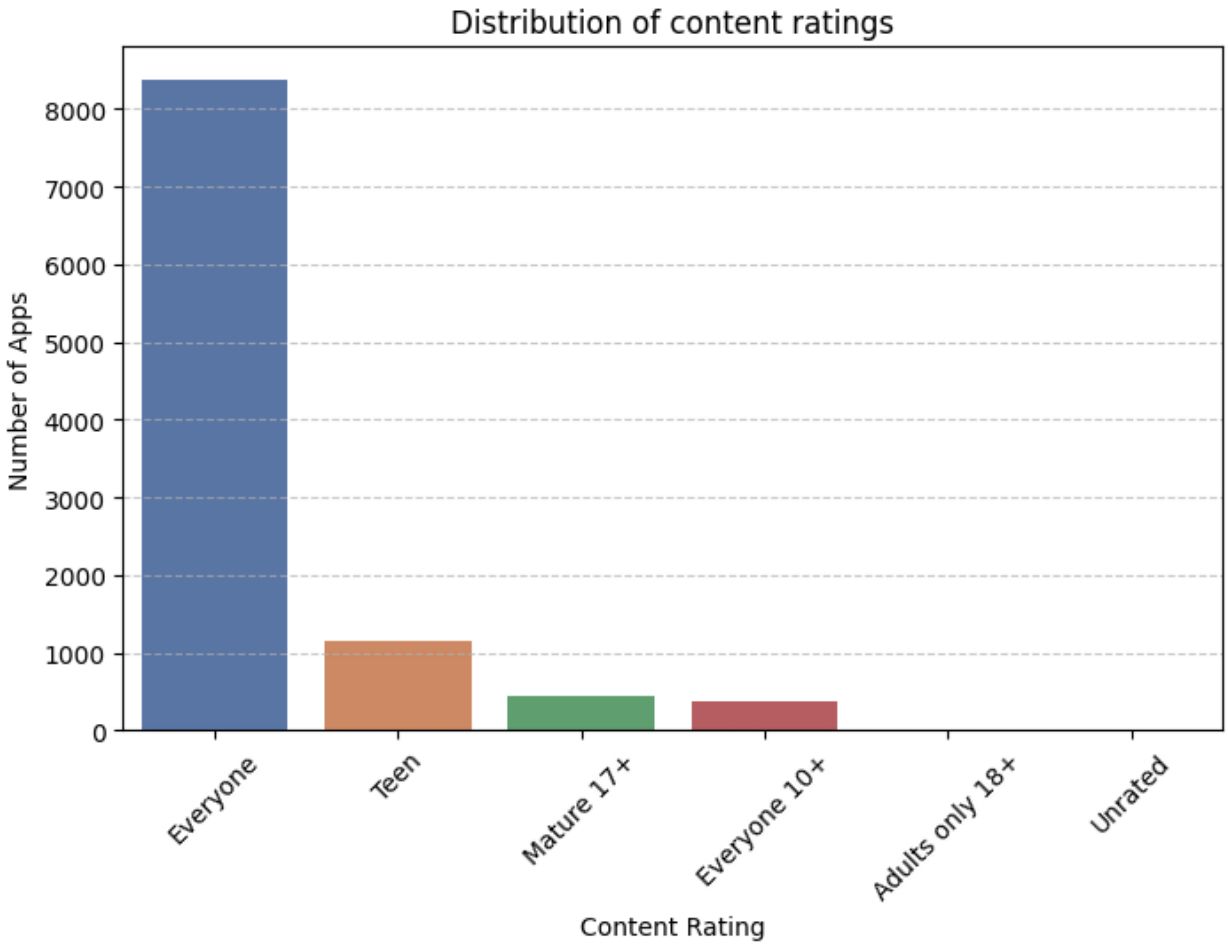
```
C:\Users\HP\AppData\Local\Temp\ipykernel_12092\2022196268.py:2:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=top_categories.index , y=top_categories.values,
palette='dark')
```



Q5: What is the distribution of content ratings?

```
content_rating_counts = dataset["Content Rating"].value_counts()
plt.figure(figsize=(8,5))
sns.barplot(x=content_rating_counts.index ,y=content_rating_counts.val
ues,palette='deep')
plt.title("Distribution of content ratings")
plt.xlabel("Content Rating")
plt.ylabel("Number of Apps")
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show() #The "Everyone" rating is the most common.
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_12092\84067620.py:3:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.


sns.barplot(x=content_rating_counts.index ,y=content_rating_counts.val
ues,palette='deep')
```

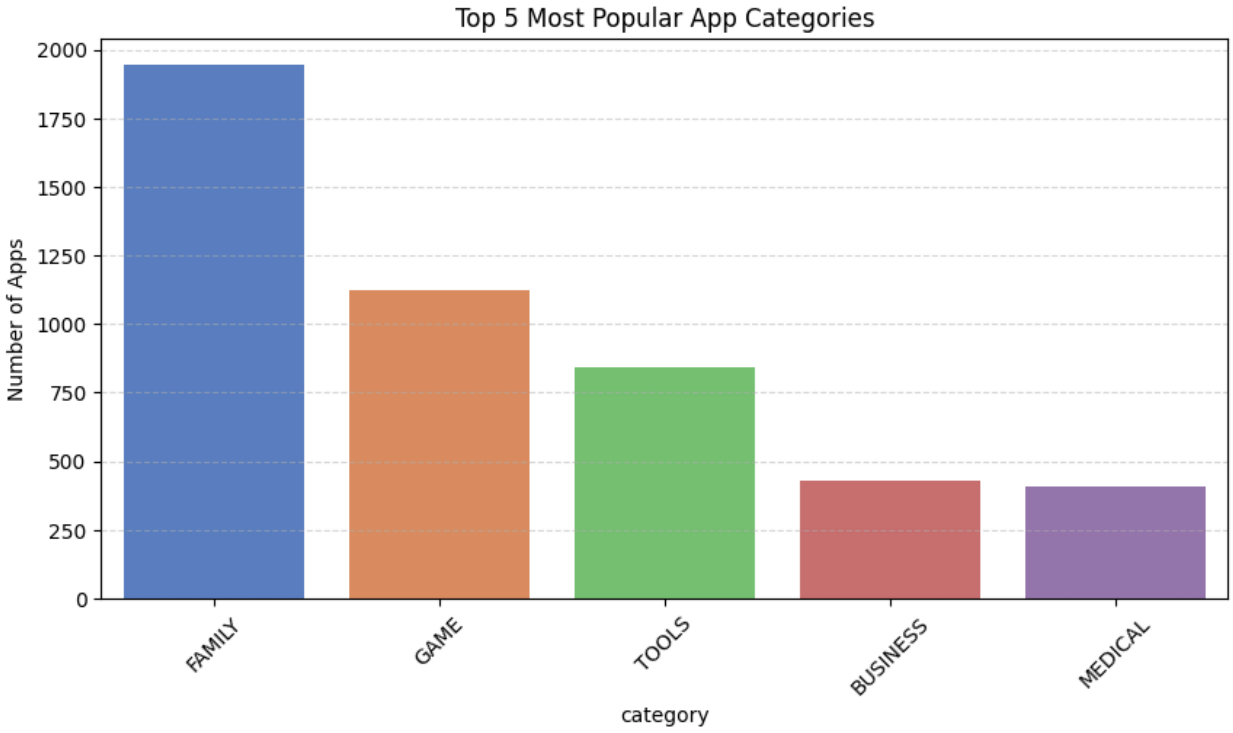


Q6: How many apps belong to the top 5 most popular categories?

```
category_counts = dataset["Category"].value_counts().head(5)
plt.figure(figsize=(10,5))
sns.barplot(x =category_counts.index,
y=category_counts.values,palette='muted')
plt.xticks(rotation=45)
plt.title("Top 5 Most Popular App Categories")
```

```
plt.xlabel("category")
plt.ylabel("Number of Apps")
plt.grid(axis='y',linestyle='--',alpha=0.5)
plt.show()
```

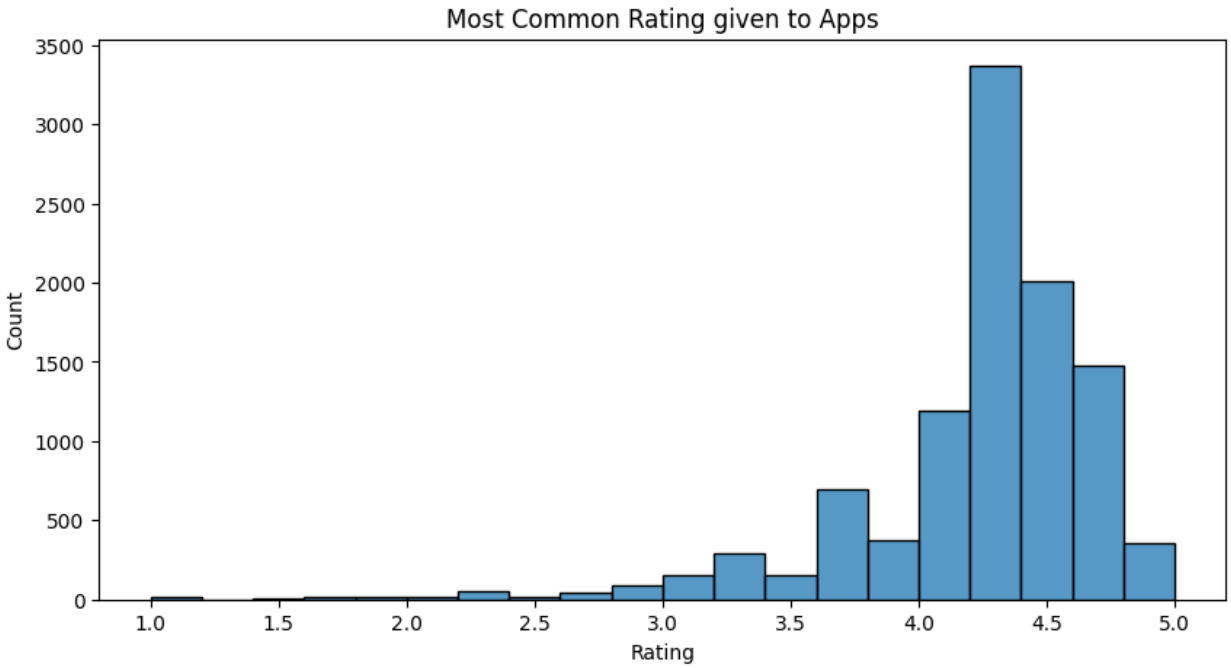
```
C:\Users\HP\AppData\Local\Temp\ipykernel_12092\239612301.py:3:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x =category_counts.index,
y=category_counts.values,palette='muted')
```



3. Univariate Visualizations (Numerical Variables)

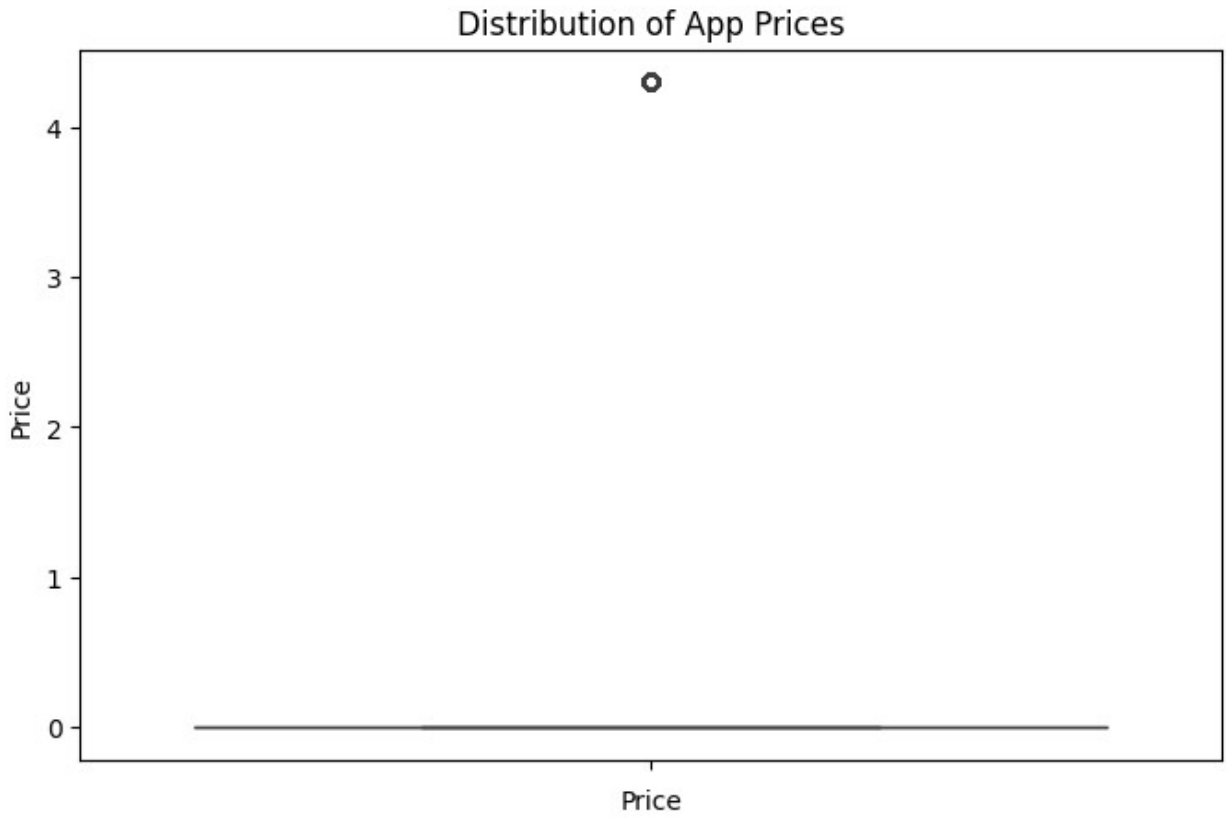
Q7: What is the most common rating given to apps?

```
plt.figure(figsize=(10,5))
sns.histplot(dataset['Rating'],bins=20)
plt.title("Most Common Rating given to Apps")
plt.show()
```



Q8: How are app prices distributed?

```
plt.figure(figsize=(8,5))
sns.boxplot(dataset["Price"])
plt.title("Distribution of App Prices")
plt.xlabel("Price")
plt.show() #Most apps are free or cheap.
```



4. Bivariate Analysis (Numerical vs Categorical)

Q9: Do free apps have better ratings than paid apps?

```
free_avg_rating = dataset[dataset["Type"] == "Free"]["Rating"].mean()
paid_avg_rating = dataset[dataset["Type"] == "Paid"]["Rating"].mean()

print(f"Average Rating (Free Apps): {free_avg_rating:.2f}")
print(f"Average Rating (Paid Apps): {paid_avg_rating:.2f}")
```

```
Average Rating (Free Apps): 4.20
Average Rating (Paid Apps): 4.27
```

Q10: Which app categories have the highest average ratings?

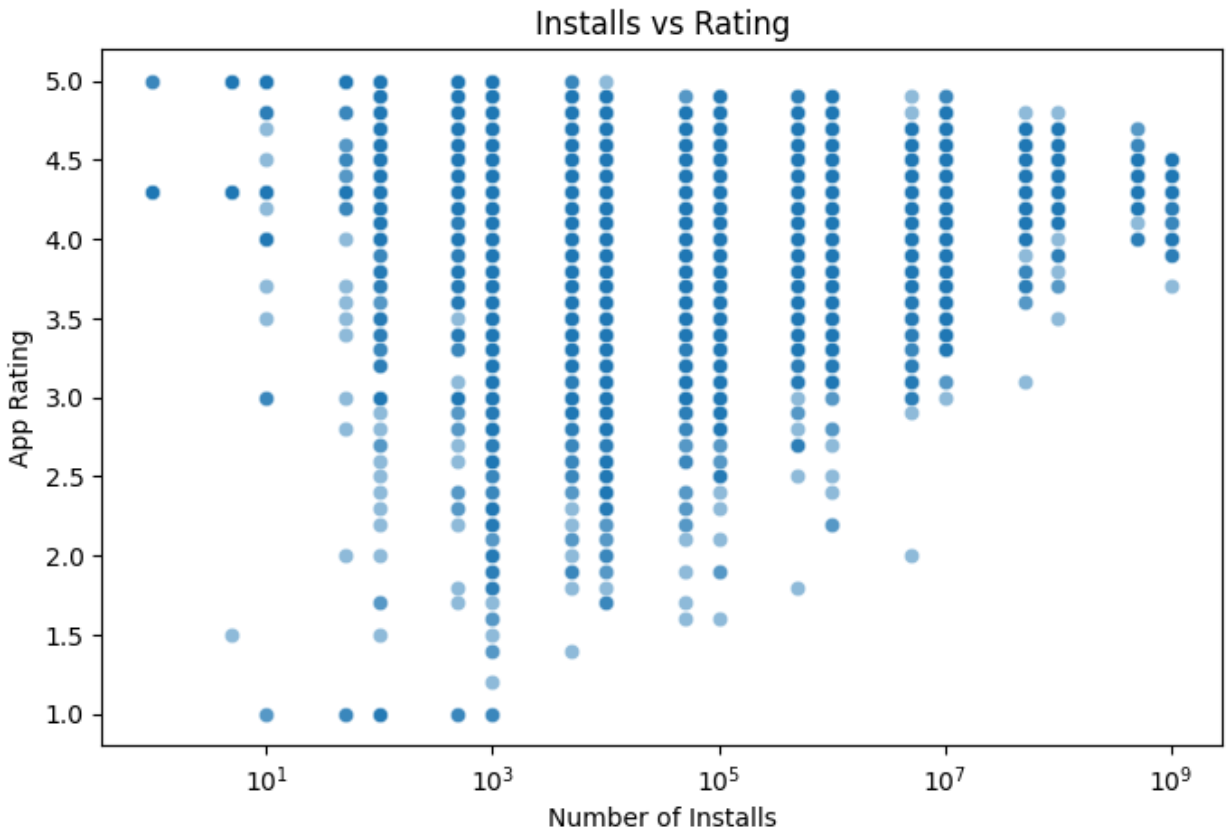
```
category_avg_rating = dataset.groupby("Category")
["Rating"].mean().sort_values(ascending=False)
print(category_avg_rating.head(10)) #The highest-rated categories are
"Events" and "Education".
```

```
Category
EVENTS                 4.395313
EDUCATION              4.375385
ART_AND_DESIGN         4.355385
BOOKS_AND_REFERENCE    4.336522
PERSONALIZATION        4.327062
PARENTING              4.300000
BEAUTY                 4.283019
GAME                   4.282070
HEALTH_AND_FITNESS     4.266993
SOCIAL                 4.260714
Name: Rating, dtype: float64
```

5. Bivariate Visualizations (Numerical vs Numerical)

Q11: Does a higher number of installs correlate with higher ratings?

```
plt.figure(figsize=(8,5))
sns.scatterplot(x=dataset["Installs"], y=dataset["Rating"], alpha=0.5)
plt.xscale("log")
plt.title("Installs vs Rating")
plt.xlabel("Number of Installs")
plt.ylabel("App Rating")
plt.show() #No strong correlation between installs and ratings.
```



Q12: Do paid apps generate more installs than free apps?

```
free_installs = dataset[dataset["Type"] == "Free"]["Installs"].mean()
paid_installs = dataset[dataset["Type"] == "Paid"]["Installs"].mean()

print(f"Average Installs (Free Apps): {free_installs:.2f}")
print(f"Average Installs (Paid Apps): {paid_installs:.2f}") #Free apps
get significantly more installs.
```

```
Average Installs (Free Apps): 15279679.80
Average Installs (Paid Apps): 90491.35
```

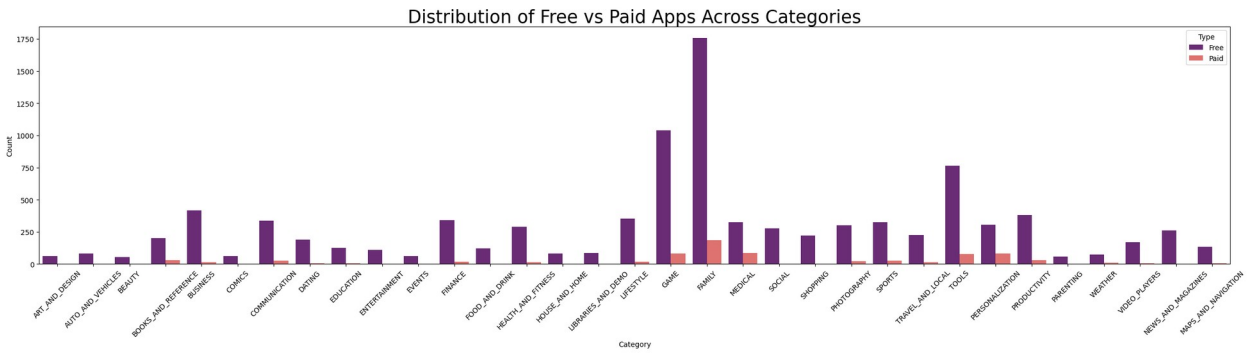
Q13: Are expensive apps rated higher than free apps?

```
plt.figure(figsize=(8,5))
sns.scatterplot(x=dataset["Price"], y=dataset["Rating"], alpha=0.5)
plt.xscale("log")
plt.title("Price vs Rating")
plt.xlabel("Price")
plt.ylabel("App Rating")
plt.show()
```

6. Bivariate Visualizations (Categorical vs Categorical)

Q14: Which categories have the most paid apps?

```
plt.figure(figsize=(30,6))
sns.countplot(x="Category", hue="Type", data=dataset, palette="magma")
plt.xticks(rotation=45)
plt.title("Distribution of Free vs Paid Apps Across Categories",
fontsize=25)
plt.xlabel("Category")
plt.ylabel("Count")
plt.legend(title="Type")
plt.show() #Games,medical and Family have the most paid apps.
```



THANKYOU SO MUCH