

```
# Steps1: Load the data file using pandas.
```

```
# Import pandas library
```

```
import pandas as pd
```

```
import csv
```

```
excel_csv_file_path = 'googleplaystore.csv'
```

```
# Read the CSV-formatted Excel file into a pandas DataFrame
```

```
df = pd.read_csv(excel_csv_file_path,encoding='latin-1')
```

```
df
```

```
df.info()
```

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

```
# Step 2: Check for null values in the data. Get the number of null values for each column.
```

```
missing_data = df.isnull()
```

```
for column in missing_data.columns.values.tolist():
```

```
    print(column)
```

```
    print (missing_data[column].value_counts())
```

```
    print("")
```

```
App
App
False    10841
Name: count, dtype: int64

Category
Category
False    10841
Name: count, dtype: int64

Rating
Rating
False    9367
True     1474
Name: count, dtype: int64

Reviews
Reviews
False    10841
Name: count, dtype: int64

Size
Size
False    10841
Name: count, dtype: int64

Installs
Installs
False    10841
Name: count, dtype: int64

Type
Type
False    10840
True      1
Name: count, dtype: int64
```

```

Price
Price
False    10841
Name: count, dtype: int64

```

```

Content Rating
Content Rating
False    10840
True      1
Name: count, dtype: int64

```

```

Genres
Genres
False    10841
Name: count, dtype: int64

```

```

Last Updated
Last Updated
False    10841
Name: count, dtype: int64

```

Step 3: Drop records with nulls in any of the columns.

```
df_cleaned = df.dropna()
```

```
print("DataFrame after dropping rows with NaN values:")
```

```
print(df_cleaned)
```

```
# now check again if there is a null value
```

```
has_null = df_cleaned.isnull().values.any()
```

```
print(f"Does the DataFrame contain any null values? {has_null}")
```

→ DataFrame after dropping rows with NaN values:

	App	Category \
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN
1	Coloring book moana	ART_AND_DESIGN
2	U Launcher Lite â FREE Live Cool Themes, Hid...	ART_AND_DESIGN
3	Sketch - Draw & Paint	ART_AND_DESIGN
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN
...
10834	FR Calculator	FAMILY
10836	Sya9a Maroc - FR	FAMILY
10837	Fr. Mike Schmitz Audio Teachings	FAMILY
10839	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE
10840	iHoroscope - 2018 Daily Horoscope & Astrology	LIFESTYLE

	Rating	Reviews	Size	Installs	Type	Price \
0	4.1	159	19M	10,000+	Free	0
1	3.9	967	14M	500,000+	Free	0
2	4.7	87510	8.7M	5,000,000+	Free	0
3	4.5	215644	25M	50,000,000+	Free	0
4	4.3	967	2.8M	100,000+	Free	0
...
10834	4.0	7	2.6M	500+	Free	0
10836	4.5	38	53M	5,000+	Free	0
10837	5.0	4	3.6M	100+	Free	0
10839	4.5	114	Varies with device	1,000+	Free	0
10840	4.5	398307	19M	10,000,000+	Free	0

	Content Rating	Genres	Last Updated \
0	Everyone	Art & Design	January 7, 2018
1	Everyone	Art & Design;Pretend Play	January 15, 2018
2	Everyone	Art & Design	August 1, 2018
3	Teen	Art & Design	June 8, 2018
4	Everyone	Art & Design;Creativity	June 20, 2018
...
10834	Everyone	Education	June 18, 2017
10836	Everyone	Education	July 25, 2017
10837	Everyone	Education	July 6, 2018
10839	Mature 17+	Books & Reference	January 19, 2015
10840	Everyone	Lifestyle	July 25, 2018

	Current Ver	Android Ver
0	1.0.0	4.0.3 and up
1	2.0.0	4.0.3 and up
2	1.2.4	4.0.3 and up
3	Varies with device	4.2 and up
4	1.1	4.4 and up
...
10834	1.0.0	4.1 and up
10836	1.48	4.1 and up
10837	1.0	4.1 and up

```
10839 Varies with device Varies with device
10840 Varies with device Varies with device
```

```
[9360 rows x 13 columns]
Does the DataFrame contain any null values? False
```

#_ Step 4:

Variables seem to have incorrect type and inconsistent formatting. You need to fix them:

Size column has sizes in Kb as well as Mb. To analyze, you'll need to convert these to numeric.

Extract the numeric value from the column

Multiply the value by 1,000, if size is mentioned in Mb

Convert 'Size' column to numeric (Mb to Kb)

```
df_cleaned['Size'] = df_cleaned['Size'].astype(str).str.replace('M', '000', regex=False).str.replace('K', '', regex=False)
df_cleaned['Size'] = pd.to_numeric(df_cleaned['Size'], errors='coerce')
```

Convert 'Reviews' to numeric

```
df_cleaned['Reviews'] = pd.to_numeric(df_cleaned['Reviews'], errors='coerce')
```

Clean 'Installs' column (remove ',' and '+', convert to int)


```
df_cleaned['Installs'] = df_cleaned['Installs'].astype(str).str.replace('[+,]', '', regex=True)
```

```
df_cleaned['Installs'] = pd.to_numeric(df_cleaned['Installs'], errors='coerce', downcast='integer')
```

Clean 'Price' column (remove '\$' sign, convert 'Free' to 0, and make numeric)

```
df_cleaned['Price'] = df_cleaned['Price'].astype(str).str.replace('$', '', regex=False).str.replace('Free', '0', regex=False)
```

```
df_cleaned['Price'] = pd.to_numeric(df_cleaned['Price'], errors='coerce')
```

 <ipython-input-8-1af4a40f9f8f>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df_cleaned['Size'] = df_cleaned['Size'].astype(str).str.replace('M', '000', regex=False).str.replace('K', '', regex=False)
```

<ipython-input-8-1af4a40f9f8f>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df_cleaned['Size'] = pd.to_numeric(df_cleaned['Size'], errors='coerce')
```

<ipython-input-8-1af4a40f9f8f>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df_cleaned['Reviews'] = pd.to_numeric(df_cleaned['Reviews'], errors='coerce')
```

<ipython-input-8-1af4a40f9f8f>:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df_cleaned['Installs'] = df_cleaned['Installs'].astype(str).str.replace('[+,]', '', regex=True)
```

<ipython-input-8-1af4a40f9f8f>:18: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df_cleaned['Installs'] = pd.to_numeric(df_cleaned['Installs'], errors='coerce', downcast='integer')
```

<ipython-input-8-1af4a40f9f8f>:21: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df_cleaned['Price'] = df_cleaned['Price'].astype(str).str.replace('$', '', regex=False).str.replace('Free', '0', regex=False)
```

<ipython-input-8-1af4a40f9f8f>:22: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df_cleaned['Price'] = pd.to_numeric(df_cleaned['Price'], errors='coerce')
```

Step 5. Sanity checks:

Average rating should be between 1 and 5 as only these values are allowed on the play store. Drop the rows that have a value outside this

Drop rows where 'Average Rating' is outside the allowed range (1 to 5)

```
df_cleaned = df_cleaned[(df_cleaned['Rating'] >= 1) & (df_cleaned['Rating'] <= 5)]
```

```
# Drop rows where 'Reviews' are greater than 'Installs'
df_cleaned = df_cleaned[df_cleaned['Reviews'] <= df_cleaned['Installs']]

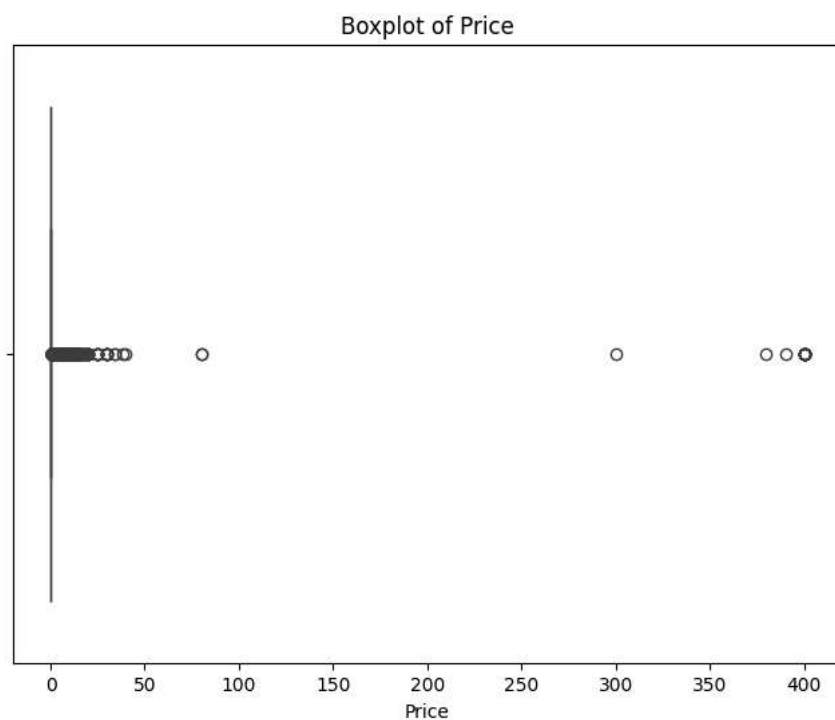
# Drop rows where 'Type' is "Free" but 'Price' is greater than 0
df_cleaned = df_cleaned[~((df_cleaned['Type'] == "Free") & (df_cleaned['Price'] > 0))]

# Step 6:
# Performing univariate analysis:

# Boxplot for Price. Are there any outliers? Think about the price of usual apps on Play Store.
```

```
import seaborn as sns
import matplotlib.pyplot as plt
```

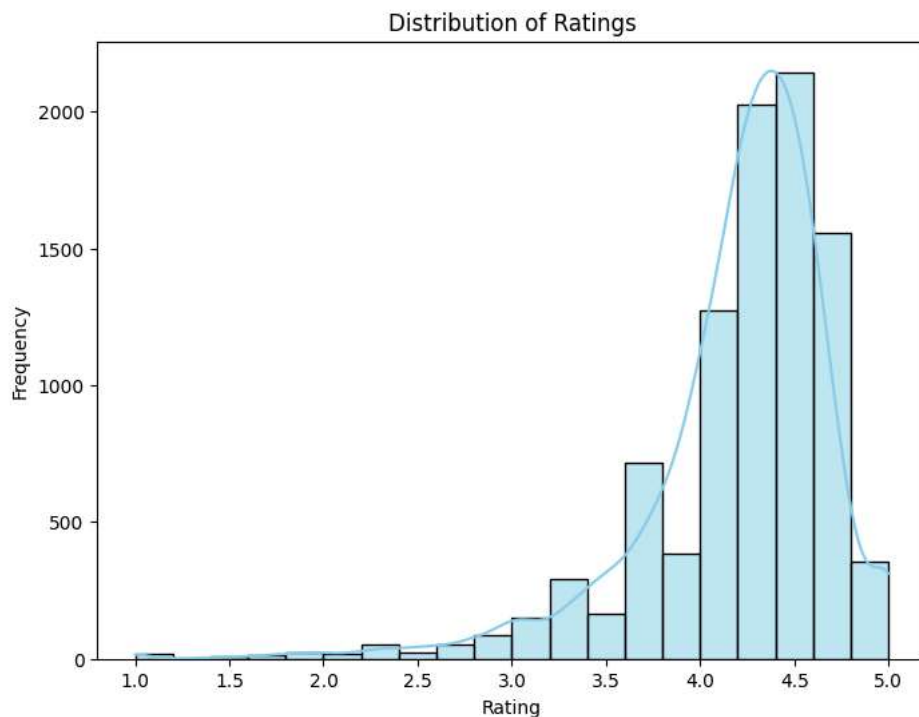
```
# Boxplot for Price column
plt.figure(figsize=(8, 6))
sns.boxplot(x=df_cleaned['Price'])
plt.title('Boxplot of Price')
plt.xlabel('Price')
plt.show()
```



```
# Plotting histogram for the 'Rating' column
plt.figure(figsize=(8, 6))
sns.histplot(df_cleaned['Rating'], bins=20, kde=True, color='skyblue', edgecolor='black')

# Adding title and labels
plt.title('Distribution of Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')

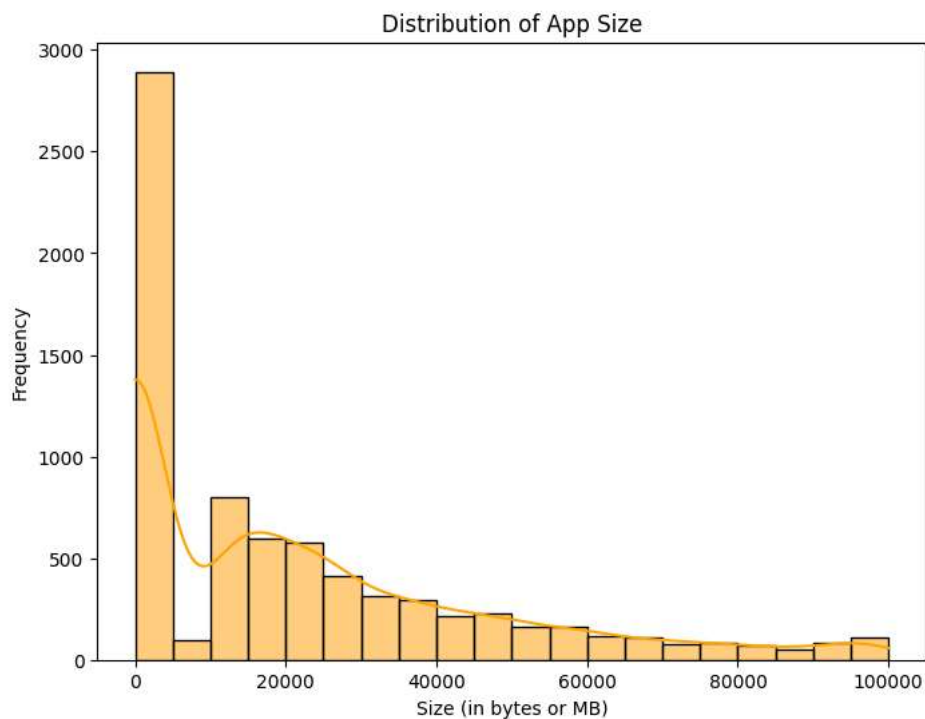
# Show the plot
plt.show()
```



```
# Plotting histogram for the 'Size' column
plt.figure(figsize=(8, 6))
sns.histplot(df_cleaned['Size'], bins=20, kde=True, color='orange', edgecolor='black')

# Adding title and labels
plt.title('Distribution of App Size')
plt.xlabel('Size (in bytes or MB)')
plt.ylabel('Frequency')

# Show the plot
plt.show()
```



Step 6. Outlier treatment:

Price: From the box plot, it seems like there are some apps with very high price. A price of \$200 for an application on the Play Store is
 # Check out the records with very high price

```
# Filter records where Price is greater than or equal to $200
high_price_apps = df_cleaned[df_cleaned['Price'] >= 200]

# Display records with high price
print("High priced apps (Price >= $200):")
print(high_price_apps)

# Drop the rows with Price >= $200
df_cleaned = df_cleaned[df_cleaned['Price'] < 200]

# Verify that the rows with high price are dropped
print("\nDataframe after dropping high priced apps:")
print(df_cleaned.head())

plt.figure(figsize=(8, 6))
sns.boxplot(x=df_cleaned['Price'])
plt.title('Boxplot of Price')
plt.xlabel('Price')
plt.show()
```

High priced apps (Price >= \$200):

	App	Category	Rating	Reviews	Size \
4197	most expensive app (H)	FAMILY	4.3	6	1.5
4362	ðŸŒŒ I'm rich	LIFESTYLE	3.8	718	26000.0
4367	I'm Rich - Trump Edition	LIFESTYLE	3.6	275	7.3
5351	I am rich	LIFESTYLE	3.8	3547	1.8
5354	I am Rich Plus	FAMILY	4.0	856	8.7
5355	I am rich VIP	LIFESTYLE	3.8	411	2.6
5356	I Am Rich Premium	FINANCE	4.1	1867	4.7
5357	I am extremely Rich	LIFESTYLE	2.9	41	2.9
5358	I am Rich!	FINANCE	3.8	93	22000.0
5359	I am rich(premium)	FINANCE	3.5	472	NaN
5362	I Am Rich Pro	FAMILY	4.4	201	2.7
5364	I am rich (Most expensive app)	FINANCE	4.1	129	2.7
5366	I Am Rich	FAMILY	3.6	217	4.9
5369	I am Rich	FINANCE	4.3	180	3.8
5373	I AM RICH PRO PLUS	FINANCE	4.0	36	41000.0

	Installs	Type	Price	Content Rating	Genres	Last Updated \
4197	100	Paid	399.99	Everyone	Entertainment	July 16, 2018
4362	10000	Paid	399.99	Everyone	Lifestyle	March 11, 2018
4367	10000	Paid	400.00	Everyone	Lifestyle	May 3, 2018
5351	100000	Paid	399.99	Everyone	Lifestyle	January 12, 2018
5354	10000	Paid	399.99	Everyone	Entertainment	May 19, 2018
5355	10000	Paid	299.99	Everyone	Lifestyle	July 21, 2018
5356	50000	Paid	399.99	Everyone	Finance	November 12, 2017
5357	1000	Paid	379.99	Everyone	Lifestyle	July 1, 2018
5358	1000	Paid	399.99	Everyone	Finance	December 11, 2017
5359	5000	Paid	399.99	Everyone	Finance	May 1, 2017
5362	5000	Paid	399.99	Everyone	Entertainment	May 30, 2017
5364	1000	Paid	399.99	Teen	Finance	December 6, 2017
5366	10000	Paid	389.99	Everyone	Entertainment	June 22, 2018
5369	5000	Paid	399.99	Everyone	Finance	March 22, 2018
5373	1000	Paid	399.99	Everyone	Finance	June 25, 2018

	Current Ver	Android Ver
4197	1.0	7.0 and up
4362	1.0.0	4.4 and up
4367	1.0.1	4.1 and up
5351	2.0	4.0.3 and up
5354	3.0	4.4 and up
5355	1.1.1	4.3 and up
5356	1.6	4.0 and up
5357	1.0	4.0 and up
5358	1.0	4.1 and up
5359	3.4	4.4 and up
5362	1.54	1.6 and up
5364	2	4.0.3 and up
5366	1.5	4.2 and up
5369	1.0	4.2 and up
5373	1.0.2	4.1 and up

Dataframe after dropping high priced apps:

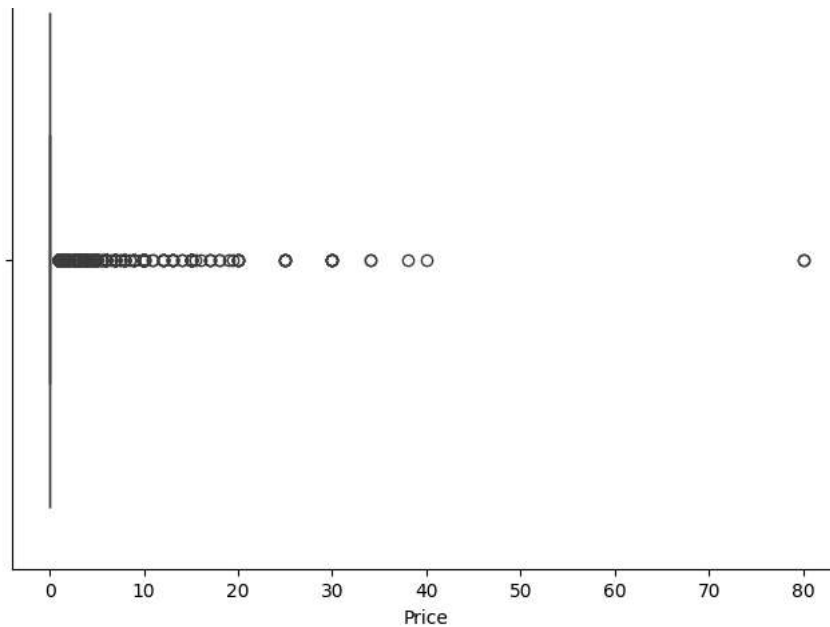
	App	Category	Rating \
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1
1	Coloring book moana	ART_AND_DESIGN	3.9
2	U Launcher Lite âŸŽŒ FREE Live Cool Themes, Hid...	ART_AND_DESIGN	4.7
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3

	Reviews	Size	Installs	Type	Price	Content Rating \
0	159	19000.0	10000	Free	0.0	Everyone
1	967	14000.0	500000	Free	0.0	Everyone
2	87510	8.7	5000000	Free	0.0	Everyone
3	215644	25000.0	50000000	Free	0.0	Teen
4	967	2.8	100000	Free	0.0	Everyone

	Genres	Last Updated	Current Ver \
0	Art & Design	January 7, 2018	1.0.0
1	Art & Design;Pretend Play	January 15, 2018	2.0.0
2	Art & Design	August 1, 2018	1.2.4
3	Art & Design	June 8, 2018	Varies with device
4	Art & Design;Creativity	June 20, 2018	1.1

	Android Ver
0	4.0.3 and up
1	4.0.3 and up
2	4.0.3 and up
3	4.2 and up
4	4.4 and up

Boxplot of Price



```
# Reviews: Very few apps have very high number of reviews.
#These are all star apps that don't help with the analysis and, in fact, will skew it. Drop records having more than 2 million reviews.
# Filter records where Reviews are greater than 2 million
high_review_apps = df_cleaned[df_cleaned['Reviews'] > 2000000]

# Display records with more than 2 million reviews
print("Apps with more than 2 million reviews:")
print(high_review_apps)

# Drop the rows where Reviews > 2 million
df_cleaned = df_cleaned[df_cleaned['Reviews'] <= 2000000]

# Verify that the rows with more than 2 million reviews are dropped
print("\nDataframe after dropping apps with more than 2 million reviews:")
print(df_cleaned.head())
```

```
10327      4.5  5534114  53000.0  100000000  Free    0.0      Teen
      Genres      Last Updated      Current Ver \
139  Books & Reference  August 1, 2018  Varies with device
335  Communication    August 1, 2018  Varies with device
336  Communication    August 3, 2018  Varies with device
338  Communication    August 1, 2018  Varies with device
340  Communication    August 2, 2018  Varies with device
```



```

1           coloring book moana ART_AND_DESIGN 3.9
2 U Launcher Lite â FREE Live Cool Themes, Hid... ART_AND_DESIGN 4.7
3           Sketch - Draw & Paint ART_AND_DESIGN 4.5
4           Pixel Draw - Number Art Coloring Book ART_AND_DESIGN 4.3

```

```

Reviews    Size  Installs  Type  Price  Content  Rating \
0      159  19000.0    10000  Free    0.0      Everyone
1      967  14000.0   500000  Free    0.0      Everyone
2     87510    8.7  5000000  Free    0.0      Everyone
3    215644  25000.0  5000000  Free    0.0      Teen
4       967    2.8   100000  Free    0.0      Everyone

```

```

Genres      Last Updated      Current Ver \
0      Art & Design  January 7, 2018      1.0.0
1 Art & Design;Pretend Play  January 15, 2018      2.0.0
2      Art & Design  August 1, 2018      1.2.4
3      Art & Design  June 8, 2018  Varies with device
4 Art & Design;Creativity  June 20, 2018      1.1

```

```

Android Ver
0 4.0.3 and up
1 4.0.3 and up
2 4.0.3 and up
3 4.2 and up
4 4.4 and up

```

Installs: There seems to be some outliers in this field too. Apps having very high number of installs should be dropped from the analysis

Find out the different percentiles - 10, 25, 50, 70, 90, 95, 99

Calculate the percentiles for the 'Installs' column

```
percentiles = df_cleaned['Installs'].quantile([0.1, 0.25, 0.5, 0.7, 0.9, 0.95, 0.99])
```

Display the calculated percentiles

```
print("Percentiles for 'Installs' column:")
```

```
print(percentiles)
```

```
↗ Percentiles for 'Installs' column:
```

```

0.10      1000.0
0.25     10000.0
0.50    500000.0
0.70   1000000.0
0.90  10000000.0
0.95  10000000.0
0.99 100000000.0

```

```
Name: Installs, dtype: float64
```

Decide a threshold as cutoff for outlier and drop records having values more than that

Calculate the 99th percentile for 'Installs'

```
percentile_99 = df_cleaned['Installs'].quantile(0.99)
```

Set the threshold for dropping outliers (values beyond the 99th percentile)

```
print(f"99th Percentile for Installs: {percentile_99}")
```

Drop records with 'Installs' greater than the 99th percentile

```
df_cleaned = df_cleaned[df_cleaned['Installs'] <= percentile_99]
```

Check the shape of the DataFrame after dropping outliers

```
print(f"DataFrame shape after removing outliers: {df_cleaned.shape}")
```

```
↗ 99th Percentile for Installs: 100000000.0
```

```
Dataframe shape after removing outliers: (8865, 13)
```

Step 7. Bivariate analysis: Let's look at how the available predictors relate to the variable of interest, i.e., our target variable rating

Make scatter plot/joinplot for Rating vs. Price

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

Scatter plot between 'Rating' and 'Price'

```
plt.figure(figsize=(8, 6))
```

```
sns.scatterplot(data=df_cleaned, x='Price', y='Rating', color='blue')
```

```
plt.title('Scatter Plot: Rating vs Price')
```

```
plt.xlabel('Price ($)')
```

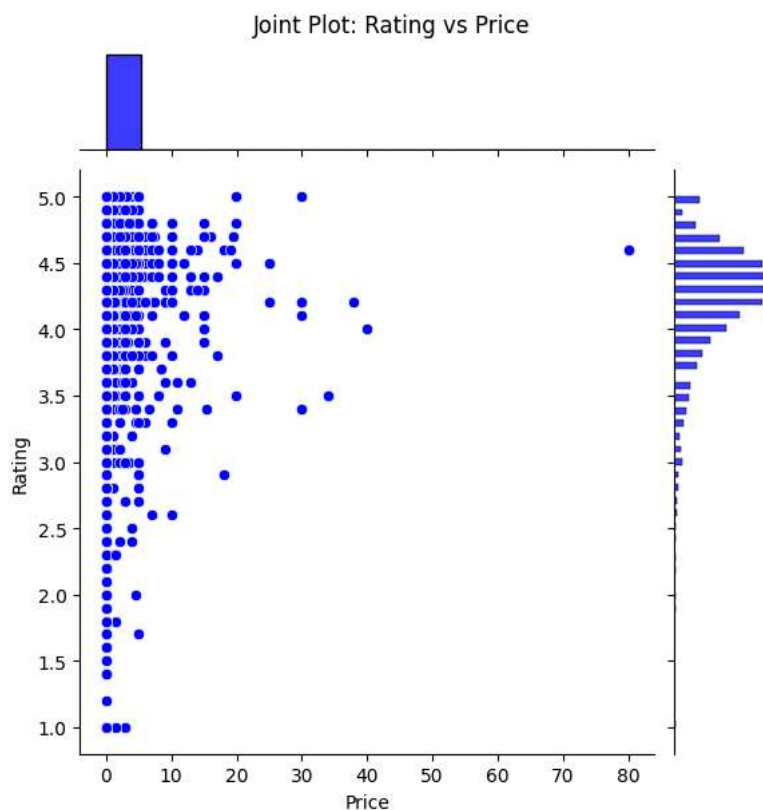
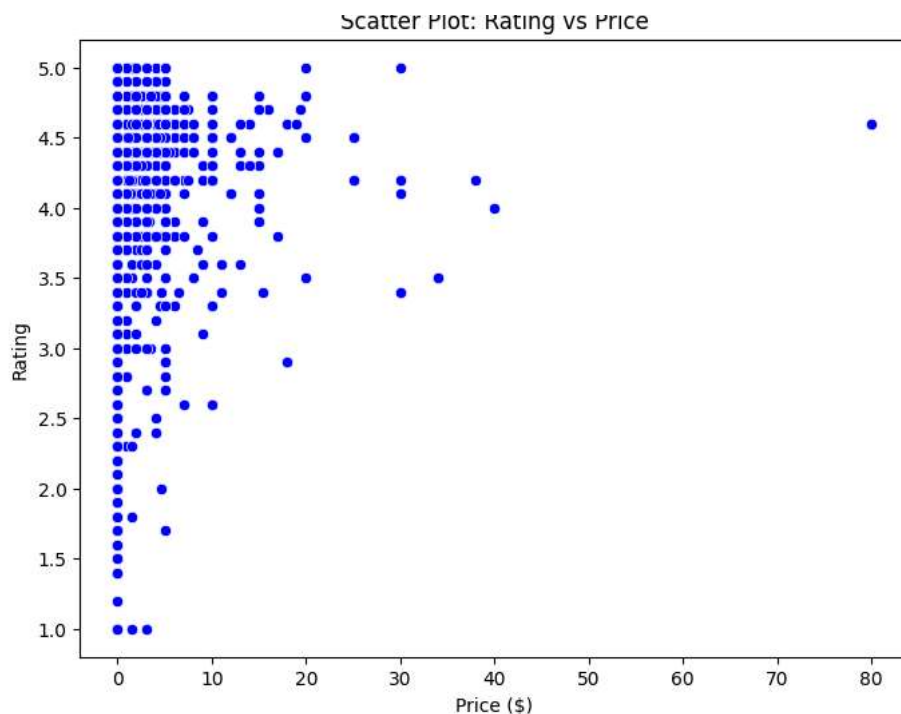
```
plt.ylabel('Rating')
```

```
plt.show()
```

Alternatively, using a joint plot

```
sns.jointplot(data=df_cleaned, x='Price', y='Rating', kind='scatter', color='blue')
plt.suptitle('Joint Plot: Rating vs Price', y=1.02)
plt.show()
```

#What pattern do you observe? Does rating increase with price?
 #price doesn't influence ratings significantly.



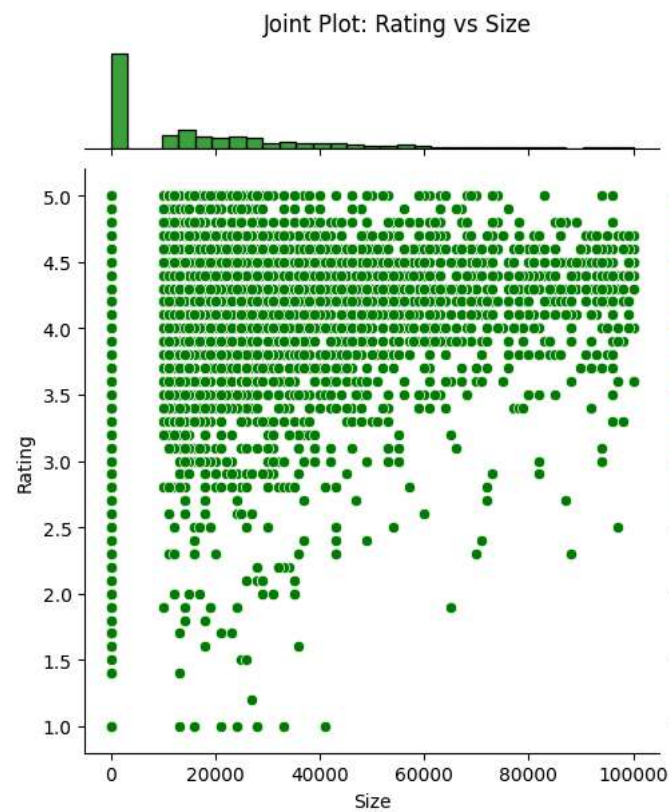
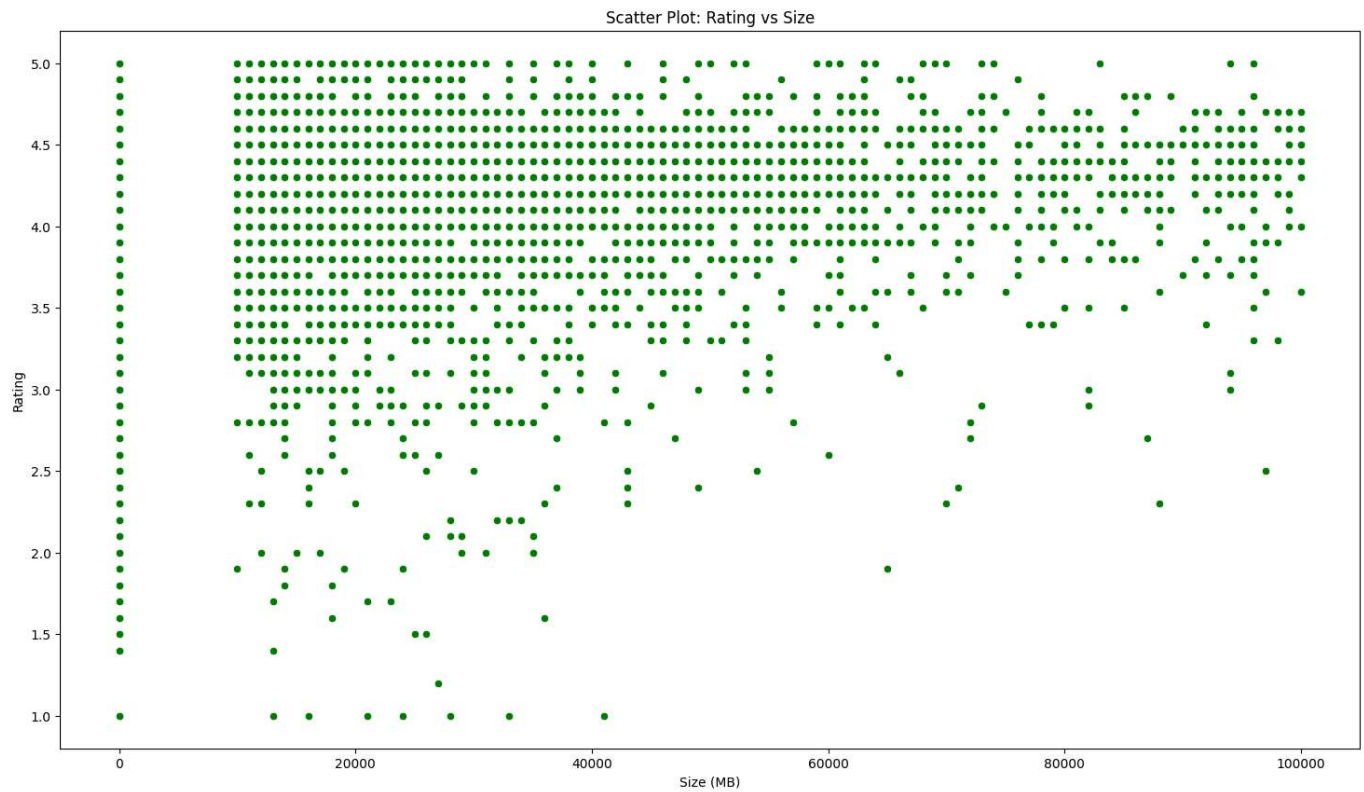
```
# Make scatter plot/joinplot for Rating vs. Size
```

```
#Are heavier apps rated better
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Scatter plot between 'Rating' and 'Size'
```

```
plt.figure(figsize=(18, 10))
sns.scatterplot(data=df_cleaned, x='Size', y='Rating', color='green')
plt.title('Scatter Plot: Rating vs Size')
plt.xlabel('Size (MB)')
plt.ylabel('Rating')
plt.show()

# Alternatively, using a joint plot
sns.jointplot(data=df_cleaned, x='Size', y='Rating', kind='scatter', color='green')
plt.suptitle('Joint Plot: Rating vs Size', y=1.02)
plt.show()
```



```
#Make scatter plot/joinplot for Rating vs. Reviews
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
# Scatter plot between 'Rating' and 'Reviews'
```

```
plt.figure(figsize=(8, 6))
```

```
sns.scatterplot(data=df_cleaned, x='Reviews', y='Rating', color='blue')
```

```
plt.title('Scatter Plot: Rating vs Reviews')
```

```
plt.xlabel('Reviews')
```

```
plt.ylabel('Rating')
```

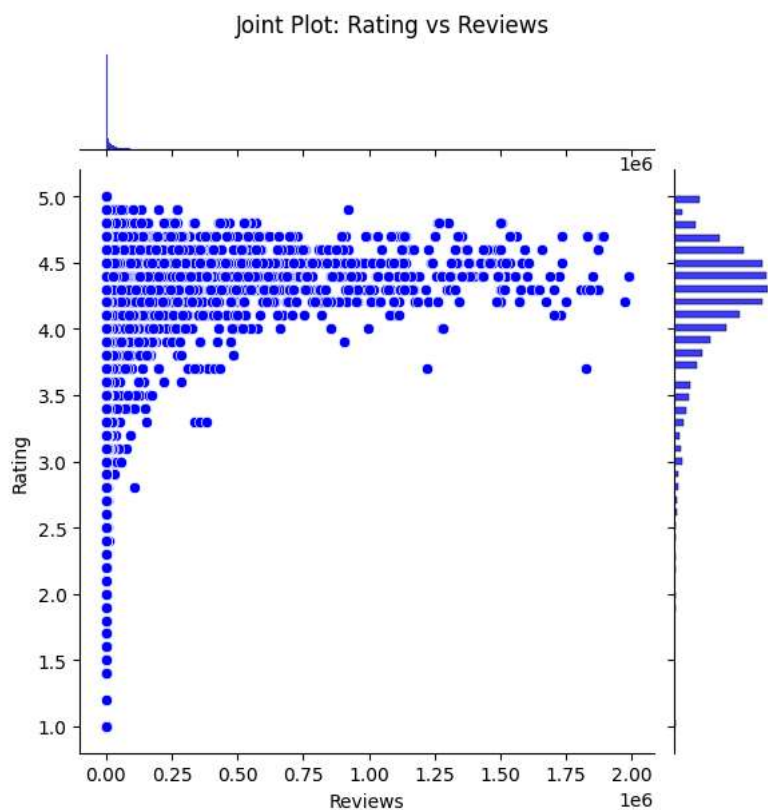
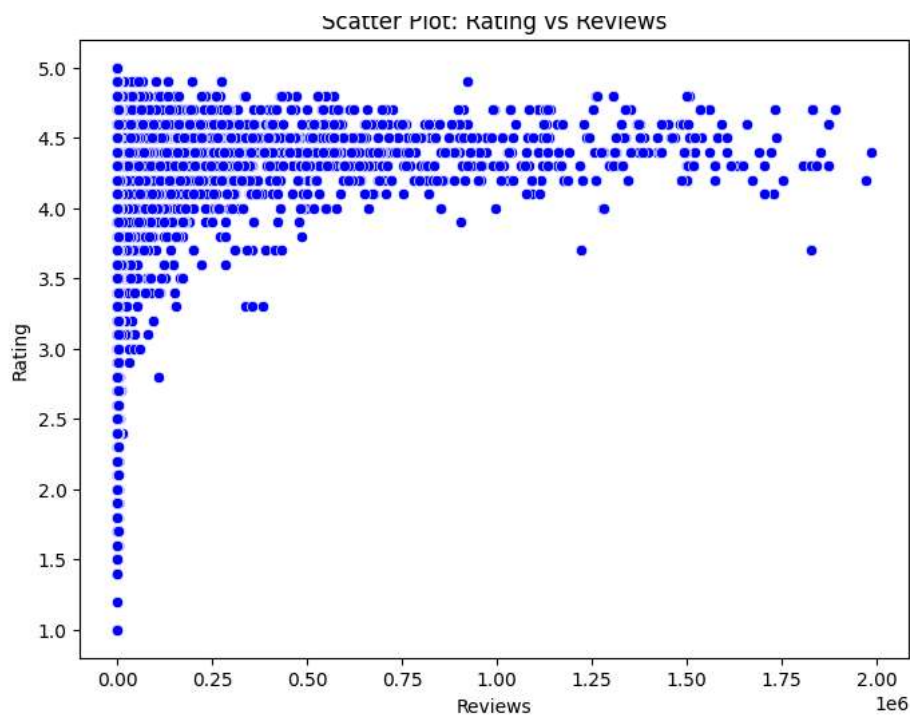
```
plt.show()
```

```
# Alternatively, using a joint plot
```

```
sns.jointplot(data=df_cleaned, x='Reviews', y='Rating', kind='scatter', color='blue')
```

```
plt.suptitle('Joint Plot: Rating vs Reviews', y=1.02)
```

```
plt.show()
```



```
# Make boxplot for Rating vs. Content Rating
```

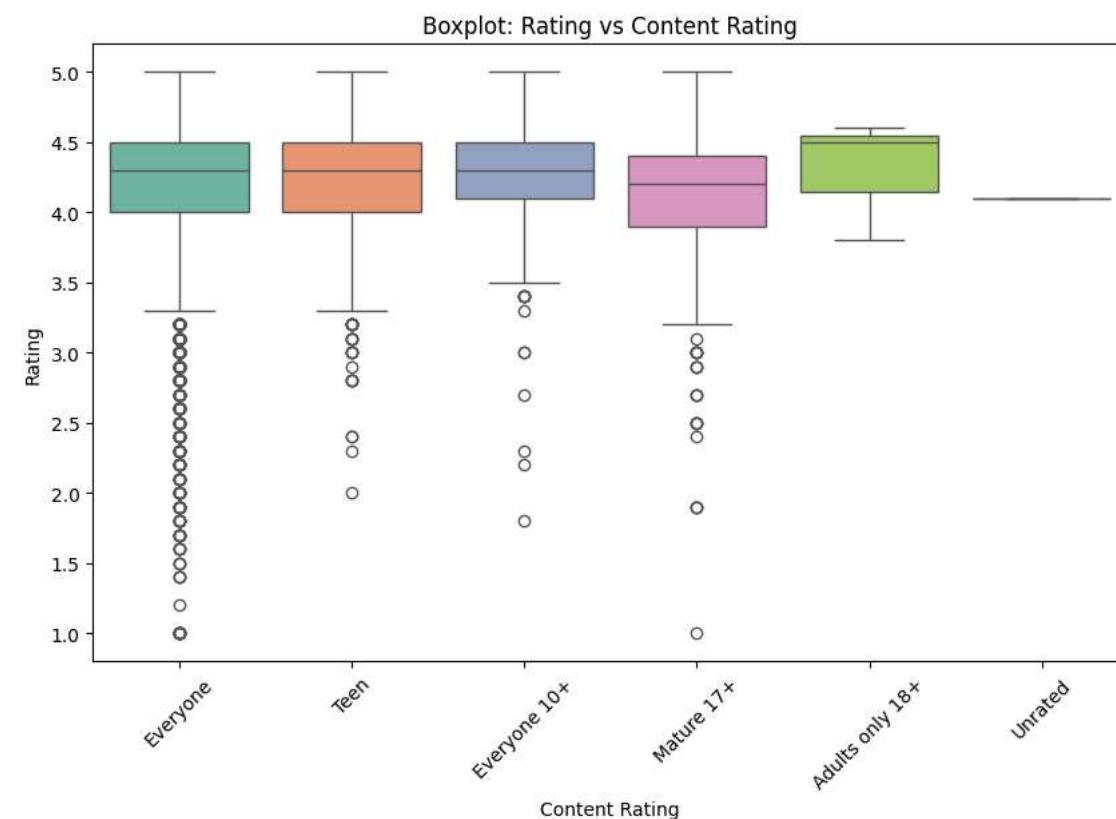
```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Boxplot between 'Rating' and 'Content Rating'
plt.figure(figsize=(10, 6))
sns.boxplot(data=df_cleaned, x='Content Rating', y='Rating', palette='Set2')
plt.title('Boxplot: Rating vs Content Rating')
plt.xlabel('Content Rating')
plt.ylabel('Rating')
plt.xticks(rotation=45)
plt.show()
```

⚡ <ipython-input-20-f063192508fa>:9: 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`

```
sns.boxplot(data=df_cleaned, x='Content Rating', y='Rating', palette='Set2')
```



```
#Make boxplot for Ratings vs. Category
```

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Boxplot between 'Rating' and 'Category'
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_cleaned, x='Category', y='Rating', palette='Set3')
plt.title('Boxplot: Rating vs Category')
plt.xlabel('Category')
plt.ylabel('Rating')
plt.xticks(rotation=90)
plt.show()
```