In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries instal
led
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/d
ocker-python
# For example, here's several helpful packages to load
import warnings
warnings.filterwarnings("ignore")
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will lis
t all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that ge
ts preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved
outside of the current session
```

```
/kaggle/input/google-play-store-apps/googleplaystore.csv
/kaggle/input/google-play-store-apps/license.txt
/kaggle/input/google-play-store-apps/googleplaystore_user_reviews.csv
```

In [2]:
```
df1=pd.read_csv("/kaggle/input/google-play-store-apps/googleplaystore.csv")
df1.head()
```

Out[2]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	D
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Des

In [3]:
```
df2=pd.read_csv("/kaggle/input/google-play-store-apps/googleplaystore_user_rev
iews.csv")
df2.head()
```

Out[3]:

	App	Translated_Review	Sentiment	Sentiment_Polarity	Sentiment_Subjectivity
0	10 Best Foods for You	I like eat delicious food. That's I'm cooking ...	Positive	1.00	0.533333
1	10 Best Foods for You	This help eating healthy exercise regular basis	Positive	0.25	0.288462
2	10 Best Foods for You	NaN	NaN	NaN	NaN
3	10 Best Foods for You	Works great especially going grocery store	Positive	0.40	0.875000
4	10 Best Foods for You	Best idea us	Positive	1.00	0.300000

In [4]:
```
# df3=pd.read_csv("/kaggle/input/google-play-store-apps/license.txt")
# df3.head
```

In [5]:
```
df1.shape
```

Out[5]:
```
(10841, 13)
```

In [6]:
```
df1['Rating'].describe()
```

Out[6]:
```
count    9367.000000
mean        4.193338
std         0.537431
min         1.000000
25%         4.000000
50%         4.300000
75%         4.500000
max        19.000000
Name: Rating, dtype: float64
```

In [7]:
```
df1['Category'].describe()
```

Out[7]:
```
count      10841
unique        34
top       FAMILY
freq        1972
Name: Category, dtype: object
```

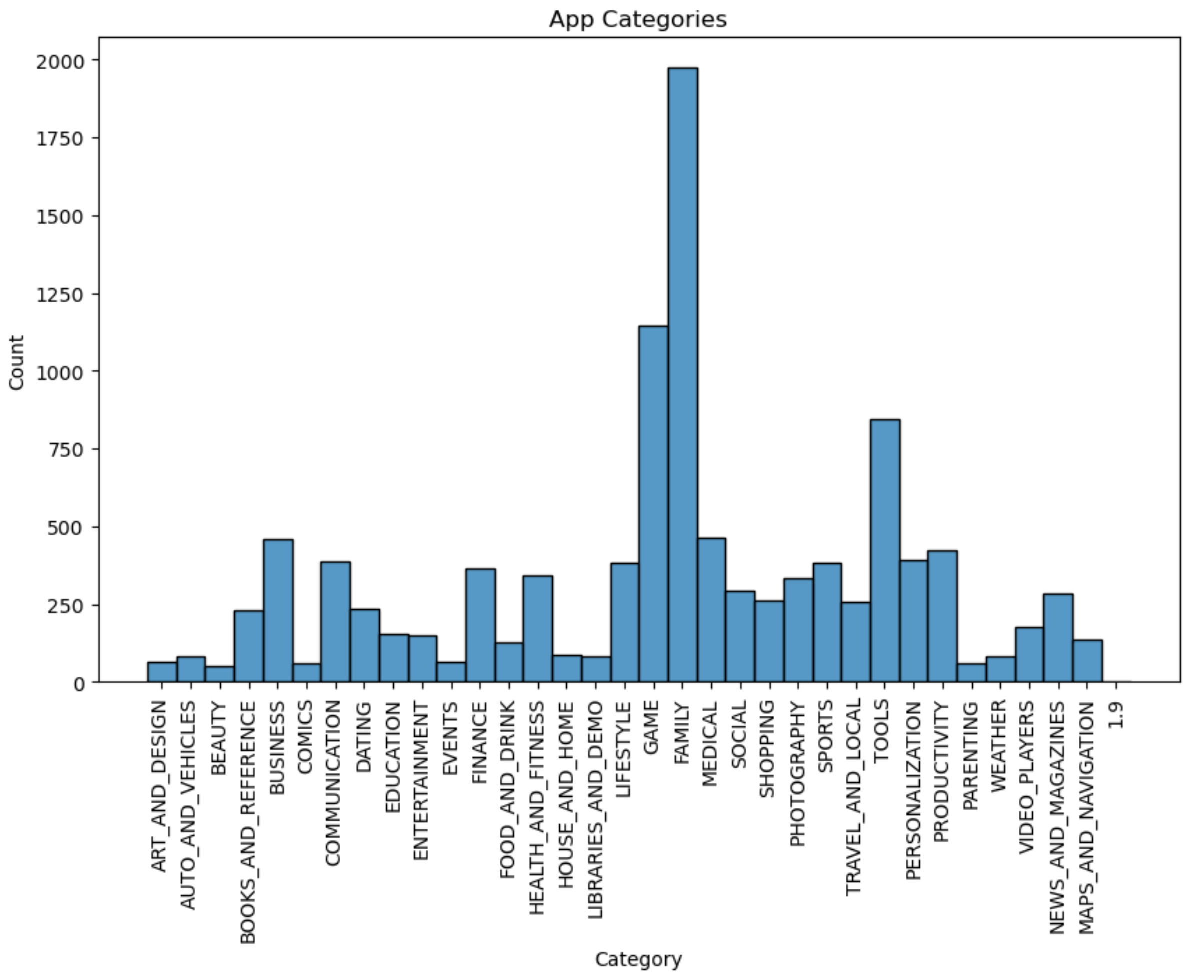
In [8]:
```
df1['Reviews'].describe()
```

Out[8]:
```
count     10841
unique     6002
top           0
freq        596
Name: Reviews, dtype: object
```

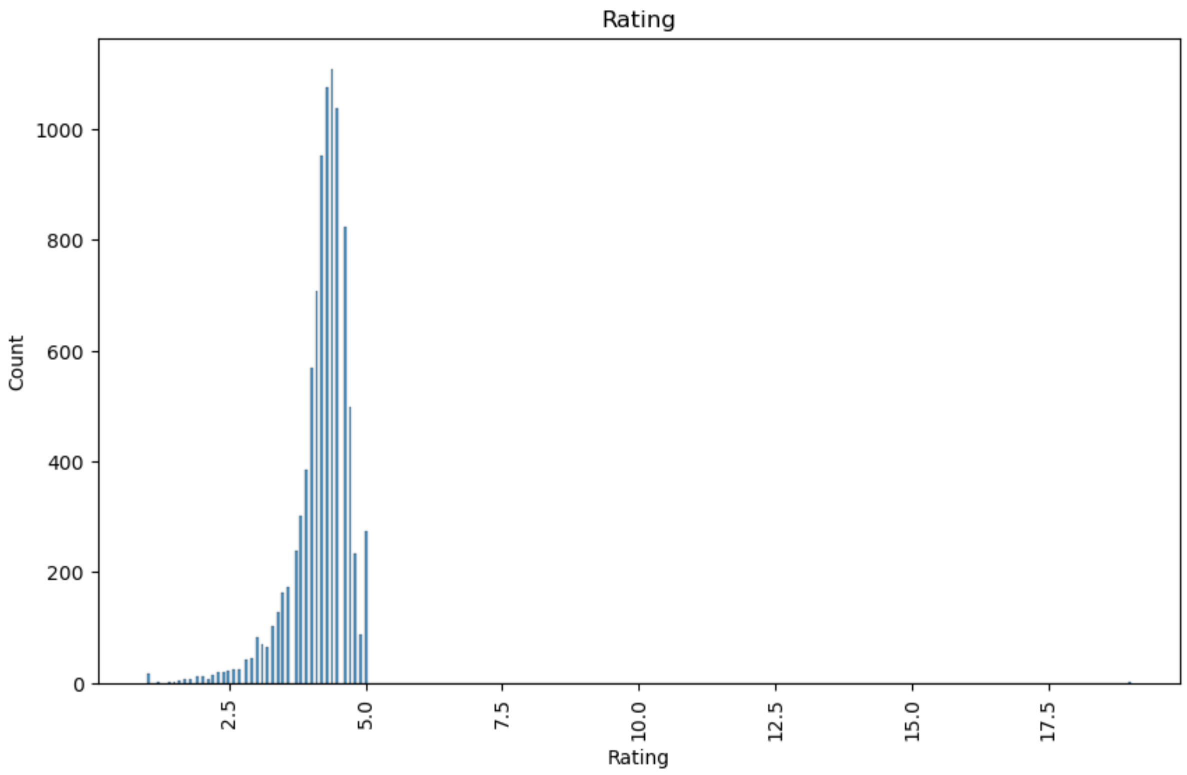
In [9]:
```
df1.info()
```

```
<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
```

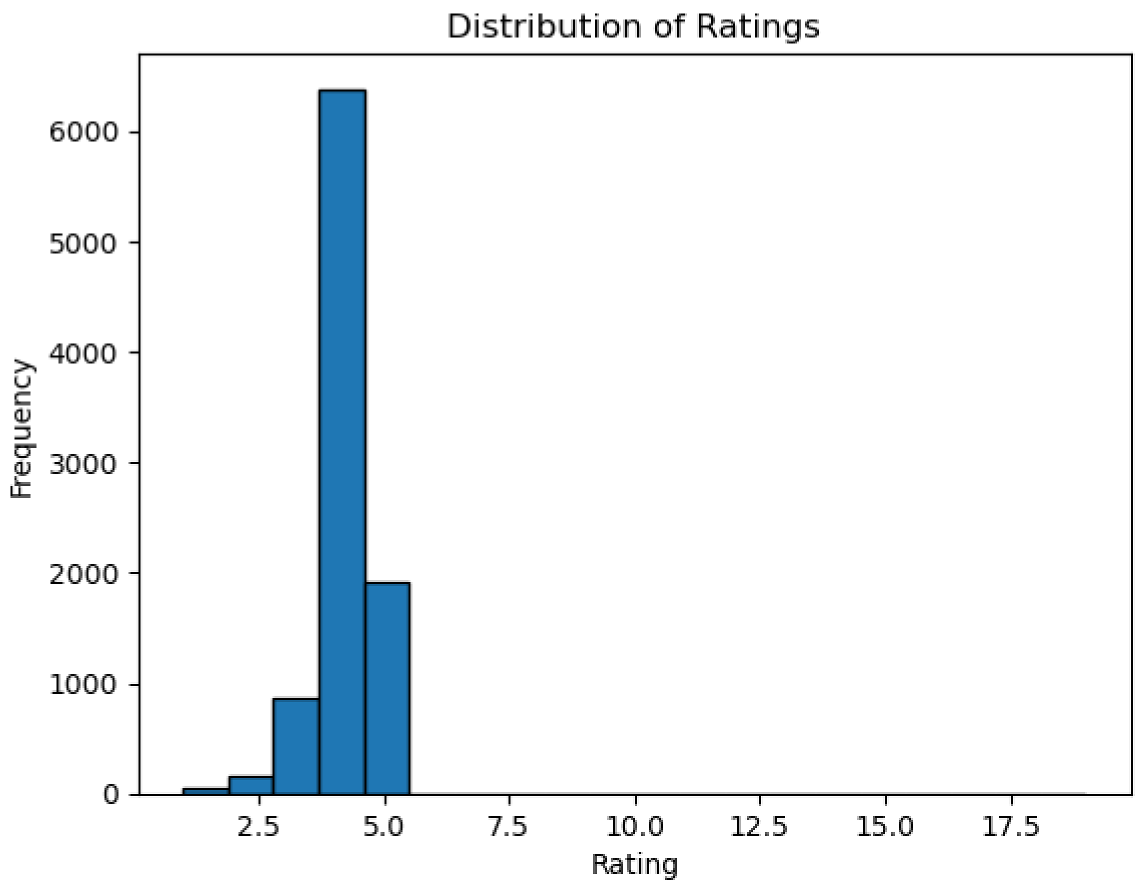
In [10]:
```
plt.figure(figsize=(10, 6))
sns.histplot(df1.Category)
plt.xticks(rotation=90)
plt.title('App Categories')
plt.show()
```



In [11]:
```
plt.figure(figsize=(10, 6))
sns.histplot(df1.Rating)
plt.xticks(rotation=90)
plt.title('Rating')
plt.show()
```

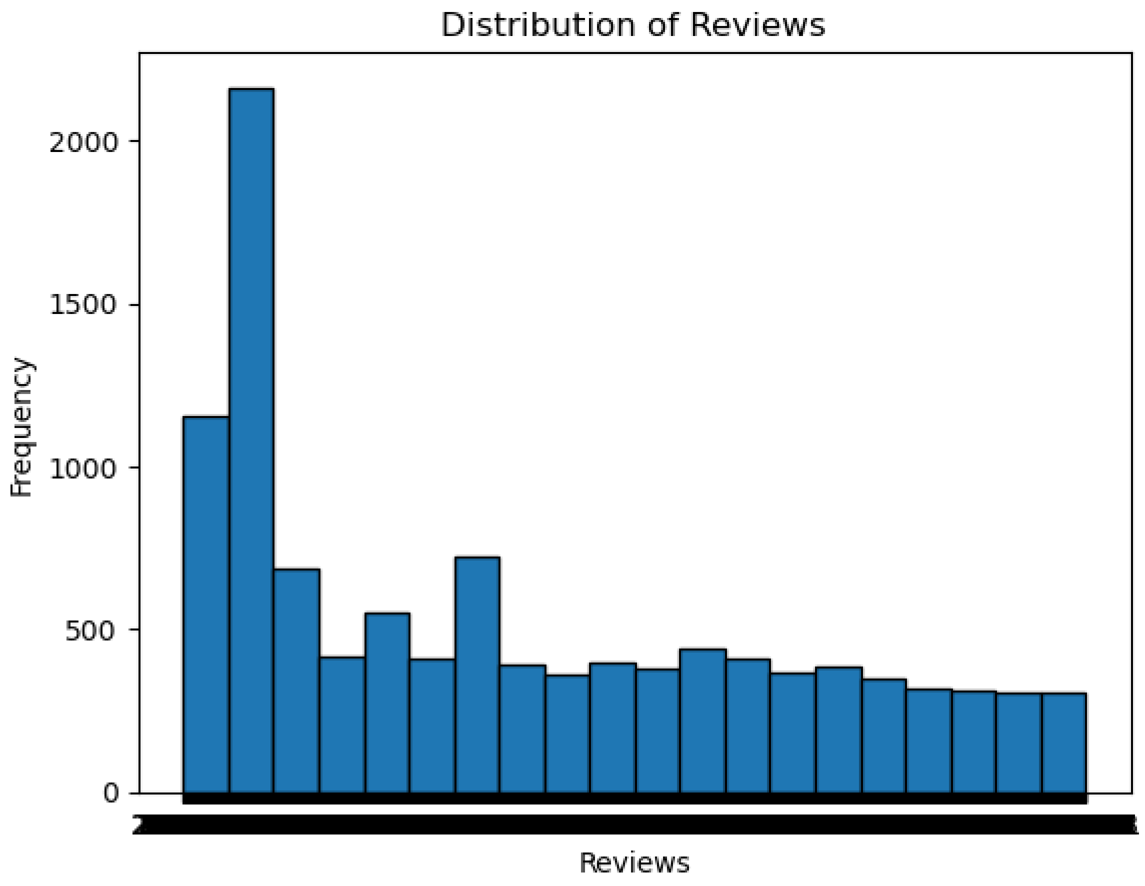


In [12]:
```
plt.hist(df1['Rating'], bins=20, edgecolor='k')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.title('Distribution of Ratings')
plt.show()
```

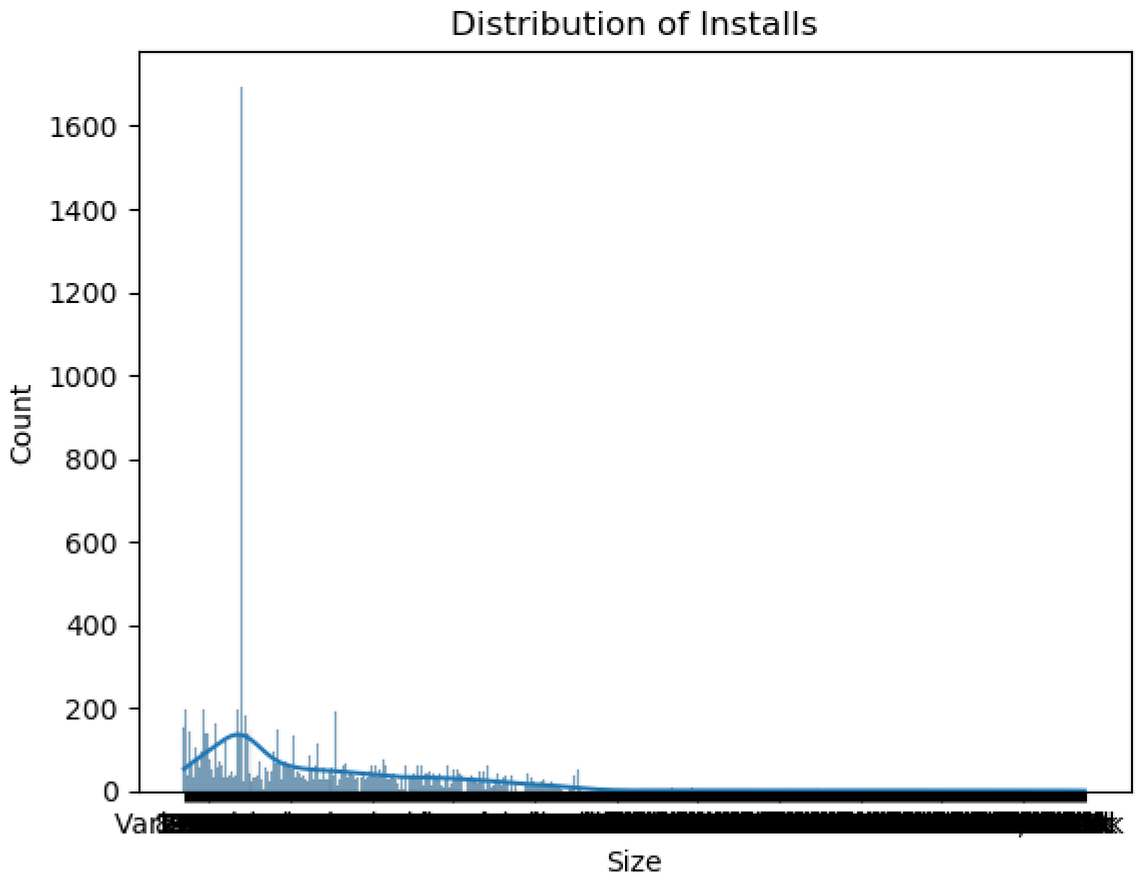


In [ ]:
```
```

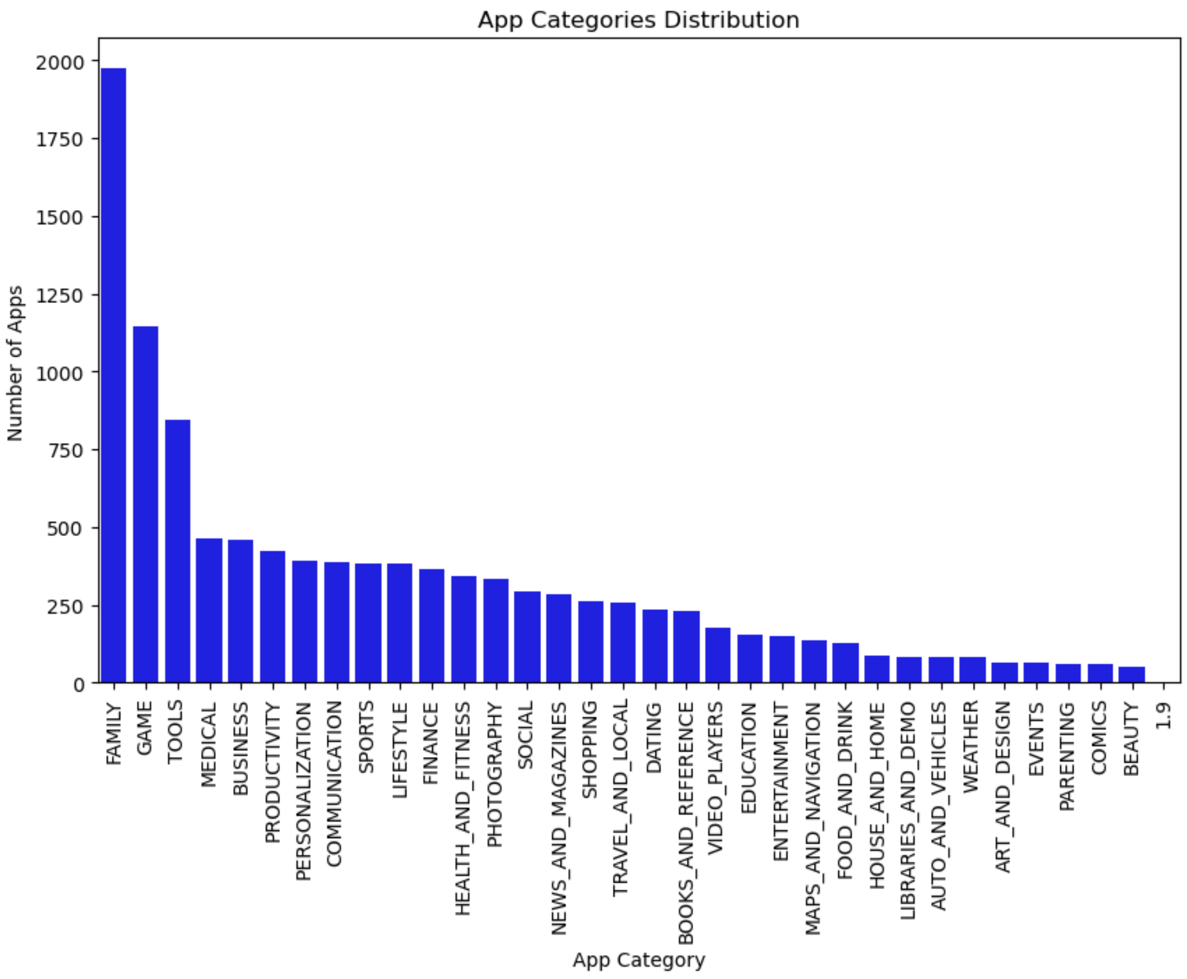
```
In [13]: plt.hist(df1['Reviews'], bins=20, edgecolor='k')

         plt.xlabel('Reviews')
         plt.ylabel('Frequency')

         plt.title('Distribution of Reviews')
         plt.show()
```



In [14]:
```
sns.histplot(df1['Size'].dropna(), kde=True)
plt.title('Distribution of Installs')
plt.show()
```



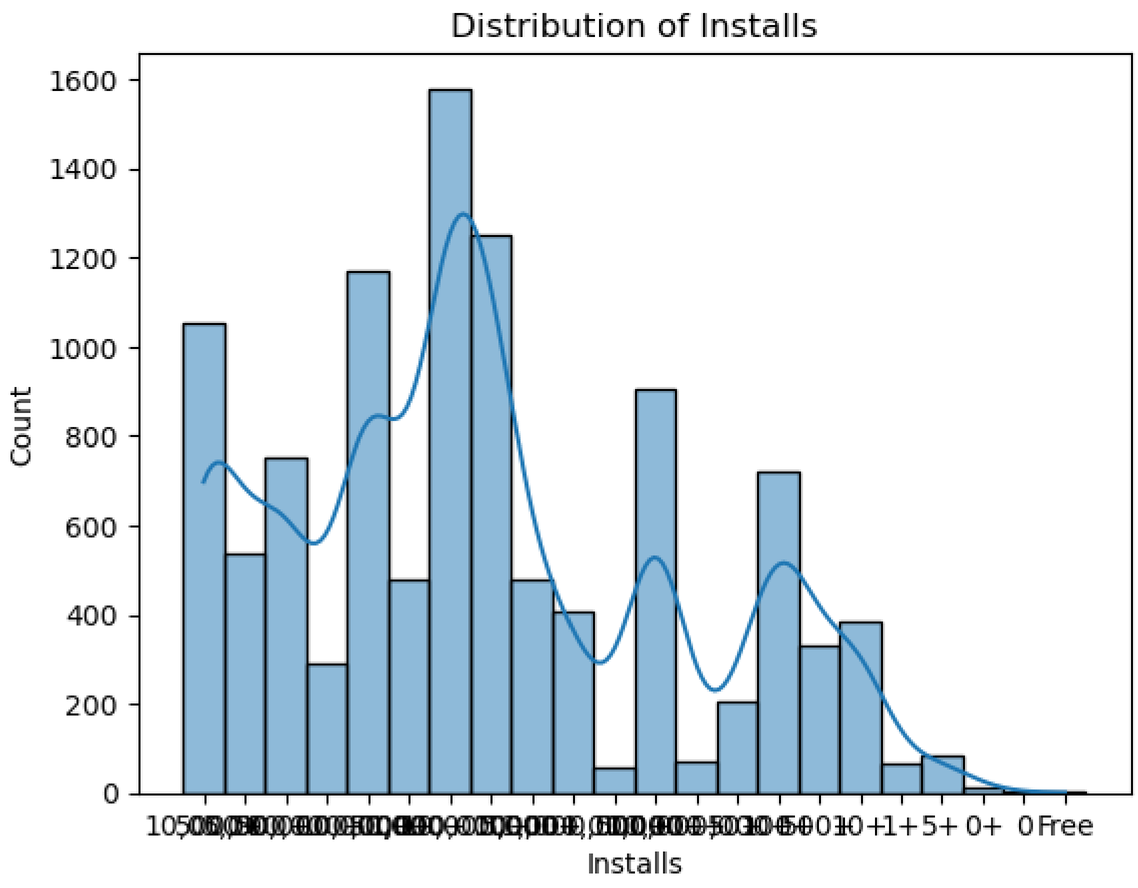
In [15]:
```
category_counts = df1['Category'].value_counts()
plt.figure(figsize=(10, 6))
sns.barplot(x=category_counts.index, y=category_counts.values, color='blue')
plt.xticks(rotation=90)
plt.xlabel('App Category')
plt.ylabel('Number of Apps')
plt.title('App Categories Distribution')
plt.show()
```



In [16]:
```
# Unique app names
unique_apps = df1['App'].unique()
print(f"Number of unique apps: {len(unique_apps)}")

# Unique app categories
unique_categories = df1['Category'].unique()
print(f"Number of unique categories: {len(unique_categories)}")
```

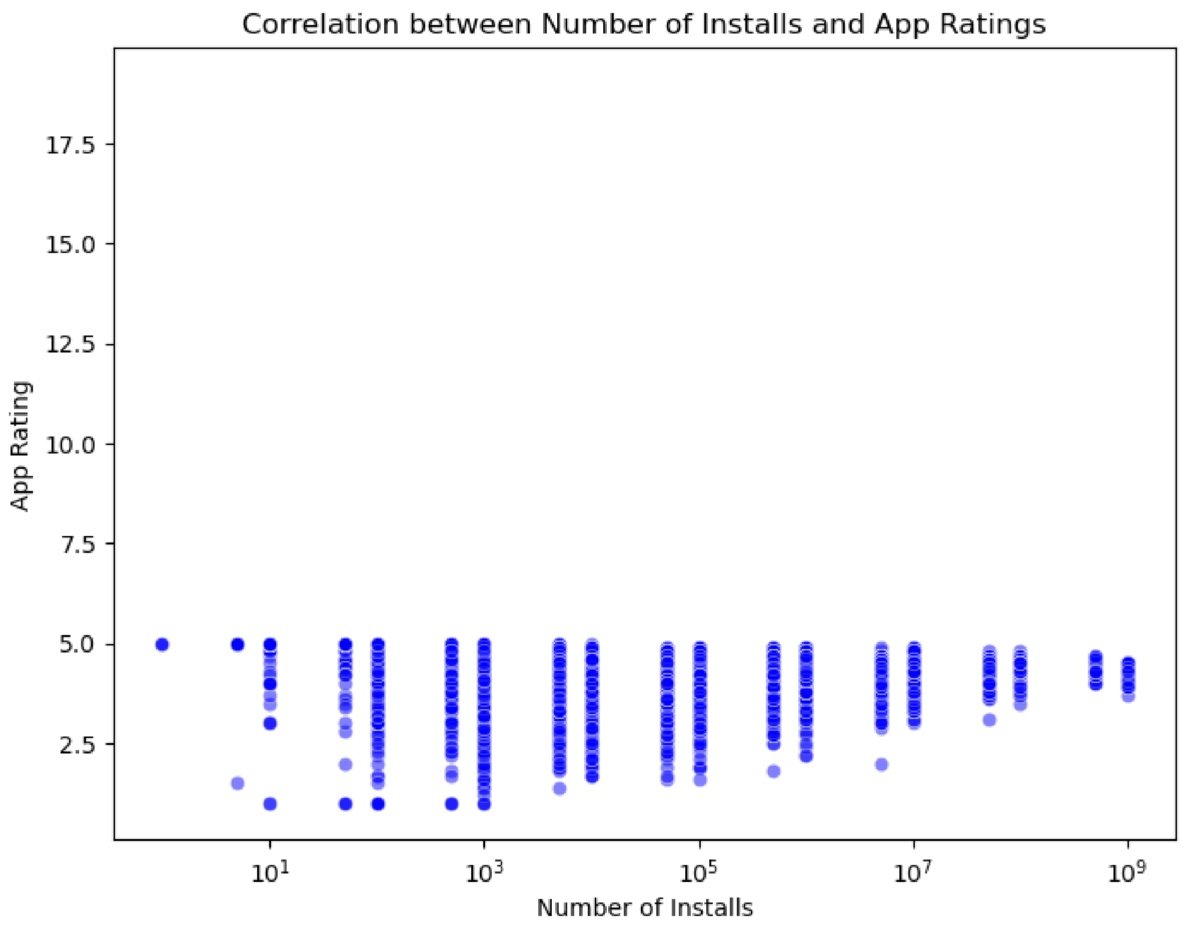
```
Number of unique apps: 9660
Number of unique categories: 34
```

```
In [17]: sns.histplot(df1['Installs'].dropna(), kde=True)
         plt.title('Distribution of Installs')
         plt.show()
```



```
In [ ]:
```

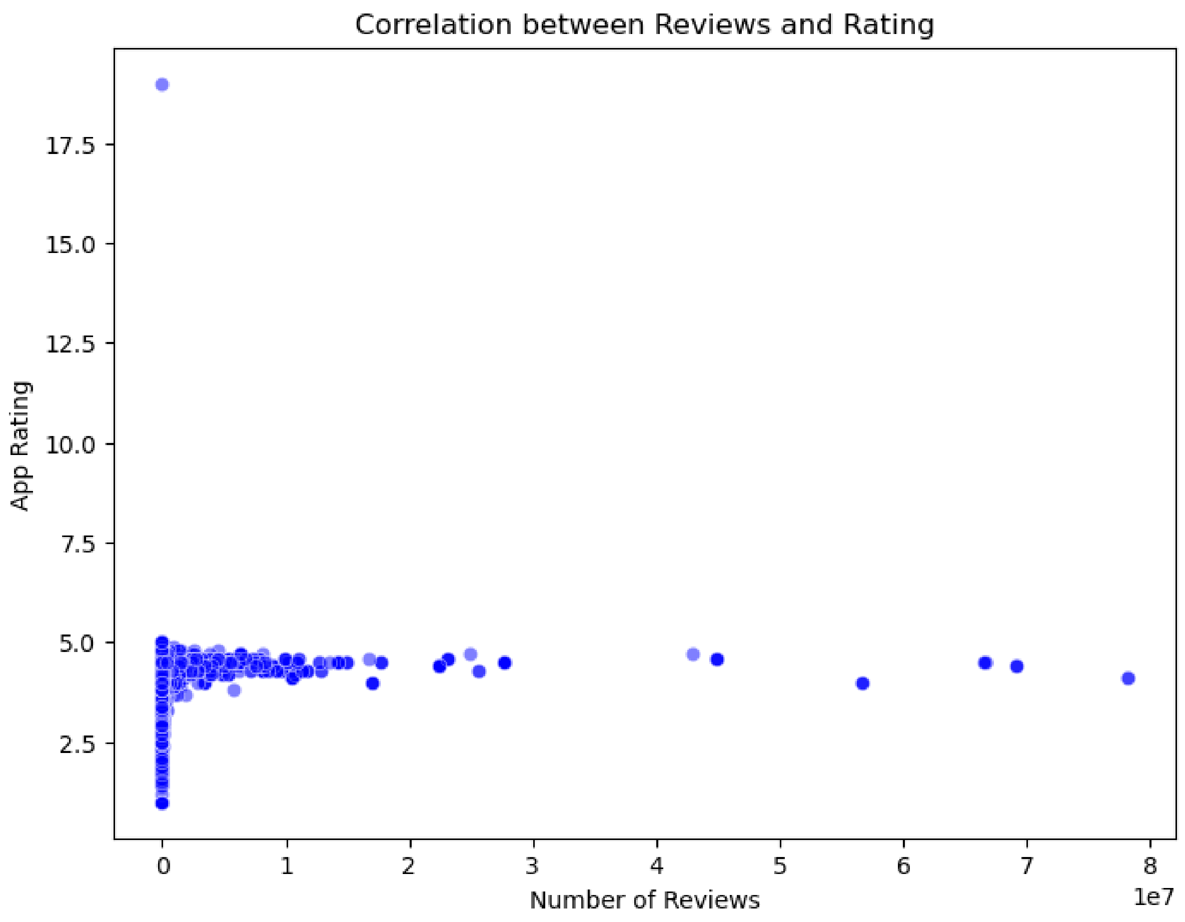
In [18]:
```
# Convert the 'Installs' column to numeric, set 'Free' values to 0
df1['Installs'] = df1['Installs'].apply(lambda x: x.replace(',', '').replace
('+', '') if isinstance(x, str) else x)
df1['Installs'] = pd.to_numeric(df1['Installs'], errors='coerce').fillna(0).as
type(int)

# Plot the correlation between 'Installs' and 'Rating' using a scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df1, x='Installs', y='Rating', color='blue', alpha=0.5)
plt.xscale('log')
plt.xlabel('Number of Installs')
plt.ylabel('App Rating')
plt.title('Correlation between Number of Installs and App Ratings')
plt.show()
```



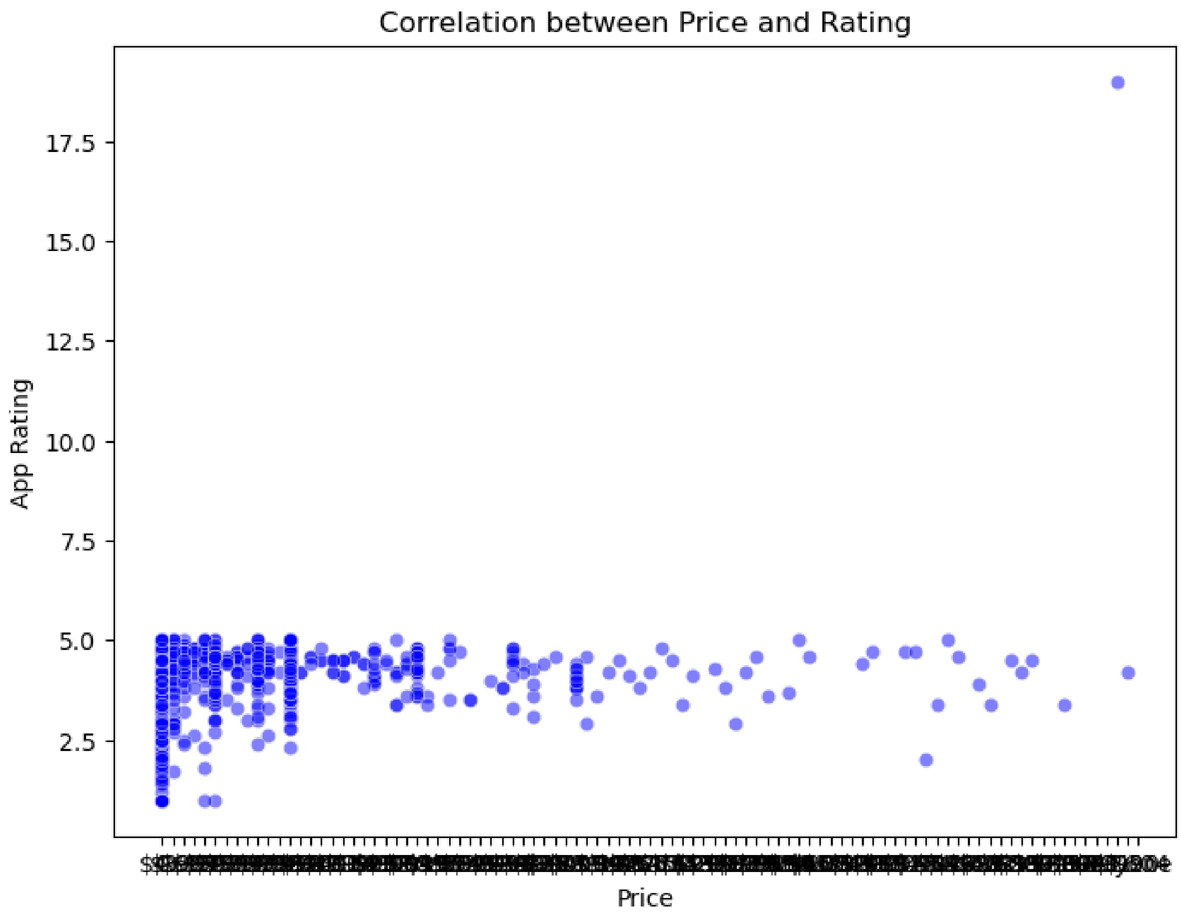
In [19]:
```python
import re

# Load the dataset into a Pandas DataFrame


# Convert the 'Reviews' column to numeric, extracting only numeric values
def convert_reviews_to_numeric(reviews):
    if isinstance(reviews, str):
        numeric_part = re.findall(r'\d+', reviews)
        if len(numeric_part) > 0:
            return int(numeric_part[0])
    return reviews

df1['Reviews'] = df1['Reviews'].apply(convert_reviews_to_numeric)

# Calculate the correlation between 'Reviews' and 'Rating'
correlation = df1['Reviews'].corr(df1['Rating'])

print(f"Correlation between Reviews and Rating: {correlation:.2f}")
```

```
Correlation between Reviews and Rating: 0.06
```

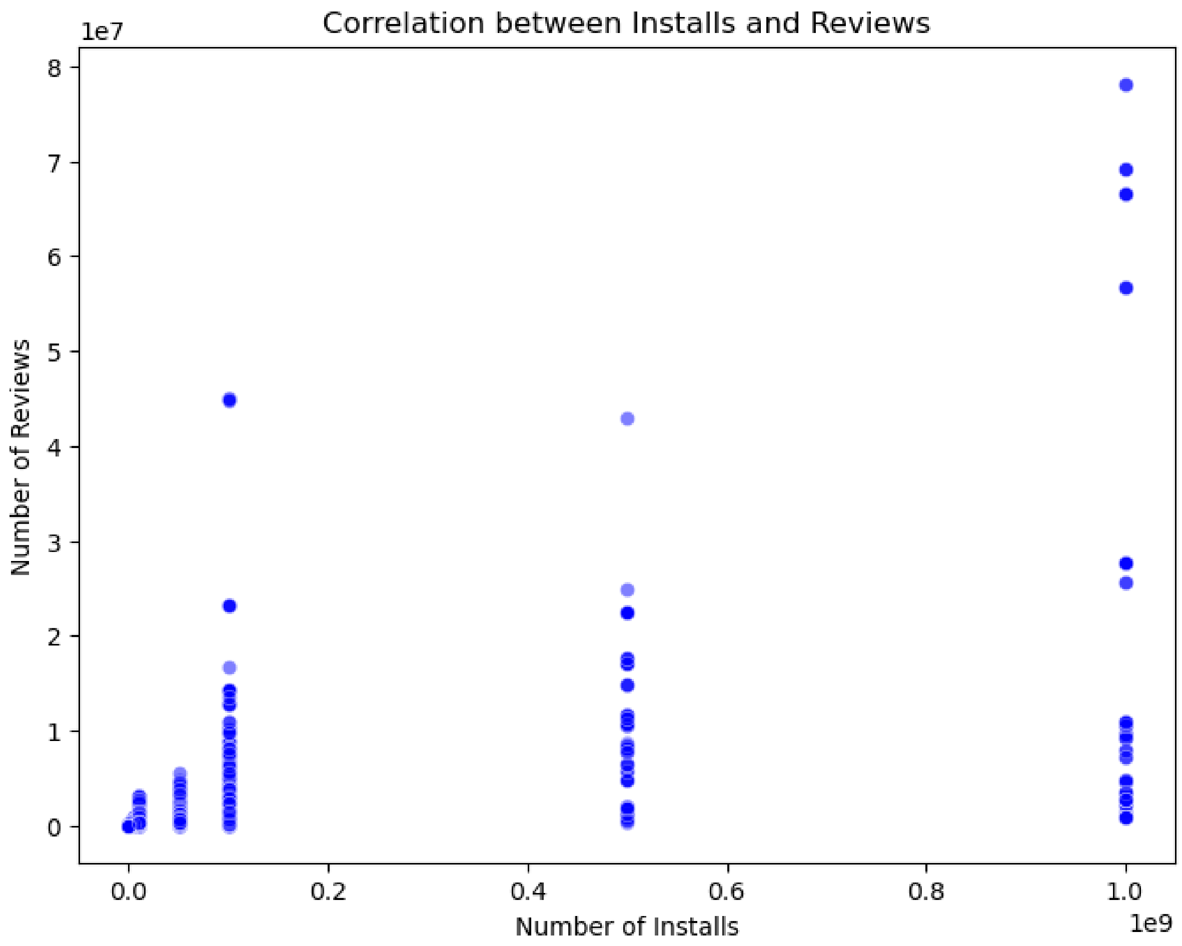
In [20]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df1, x='Reviews', y='Rating', color='blue', alpha=0.5)
plt.xlabel('Number of Reviews')
plt.ylabel('App Rating')
plt.title('Correlation between Reviews and Rating')
plt.show()
```



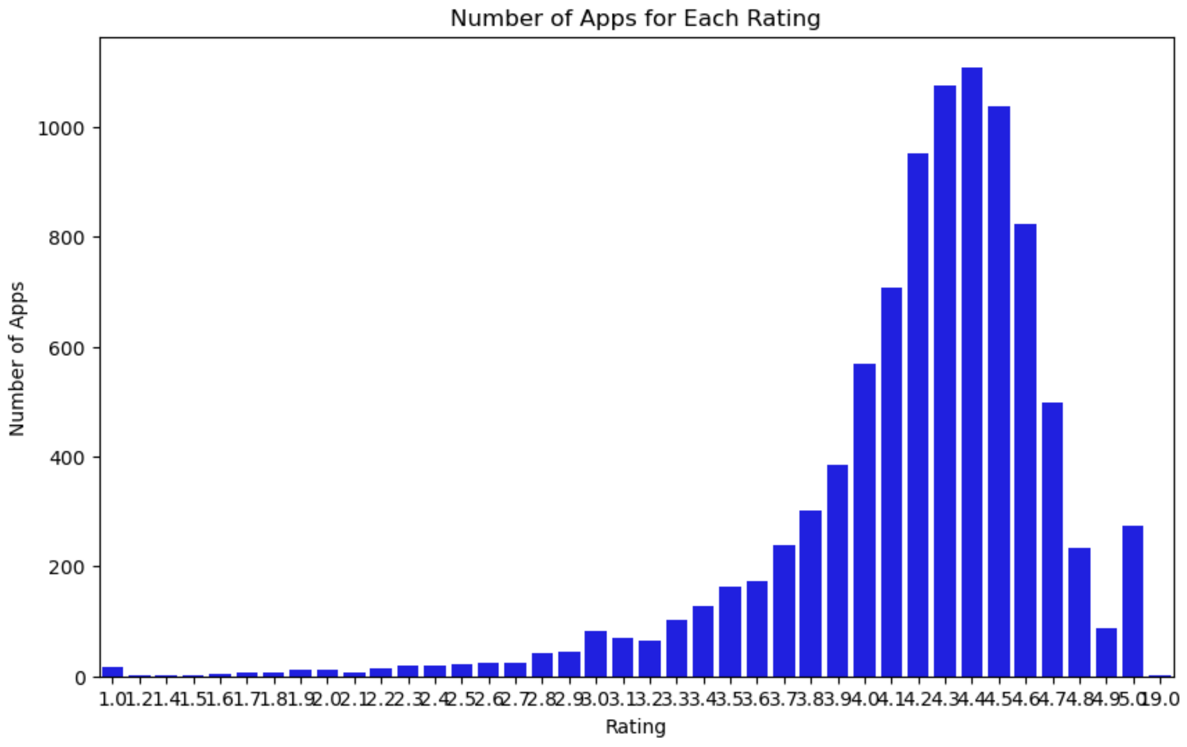
```
In [21]: plt.figure(figsize=(8, 6))
         sns.scatterplot(data=df1, x='Price', y='Rating', color='blue', alpha=0.5)
         plt.xlabel('Price')
         plt.ylabel('App Rating')
         plt.title('Correlation between Price and Rating')
         plt.show()
```



In [22]:
```
# Plot the correlation between 'Installs' and 'Reviews' using a scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df1, x='Installs', y='Reviews', color='blue', alpha=0.5)
plt.xlabel('Number of Installs')
plt.ylabel('Number of Reviews')
plt.title('Correlation between Installs and Reviews')
plt.show()
```



In [23]:
```
rating_counts = df1['Rating'].value_counts()

# Sort the ratings in ascending order for better visualization
rating_counts = rating_counts.sort_index()

# Plot the count of apps for each rating using a bar plot
plt.figure(figsize=(10, 6))
sns.barplot(x=rating_counts.index, y=rating_counts.values, color='blue')
plt.xlabel('Rating')
plt.ylabel('Number of Apps')
plt.title('Number of Apps for Each Rating')
plt.show()
```



In [24]:
```
df1 = df1.dropna()
df1
```

Out[24]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	C
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10000	Free	0	E\
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500000	Free	0	E\
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5000000	Free	0	E\
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50000000	Free	0	
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100000	Free	0	E\
...	...	...	...	...	...	...	...	...	
10834	FR Calculator	FAMILY	4.0	7	2.6M	500	Free	0	E\
10836	Sya9a Maroc - FR	FAMILY	4.5	38	53M	5000	Free	0	E\
10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3.6M	100	Free	0	E\
10839	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5	114	Varies with device	1000	Free	0	
10840	iHoroscope - 2018 Daily Horoscope & Astrology	LIFESTYLE	4.5	398307	19M	10000000	Free	0	E\

9360 rows × 13 columns

In [25]:
```
df1 = df1.drop_duplicates()
```

In [26]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8886 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             8886 non-null   object 
 1   Category        8886 non-null   object 
 2   Rating          8886 non-null   float64
 3   Reviews         8886 non-null   int64  
 4   Size            8886 non-null   object 
 5   Installs        8886 non-null   int64  
 6   Type            8886 non-null   object 
 7   Price           8886 non-null   object 
 8   Content Rating  8886 non-null   object 
 9   Genres          8886 non-null   object 
 10  Last Updated    8886 non-null   object 
 11  Current Ver     8886 non-null   object 
 12  Android Ver     8886 non-null   object 
dtypes: float64(1), int64(2), object(10)
memory usage: 971.9+ KB
```

In [40]:
```
# # 3. Data Type Conversion:
# # Convert 'Reviews' to numeric (integer) values
# df1['Reviews'] = df1['Reviews'].str.replace(',', '').astype(int)
```

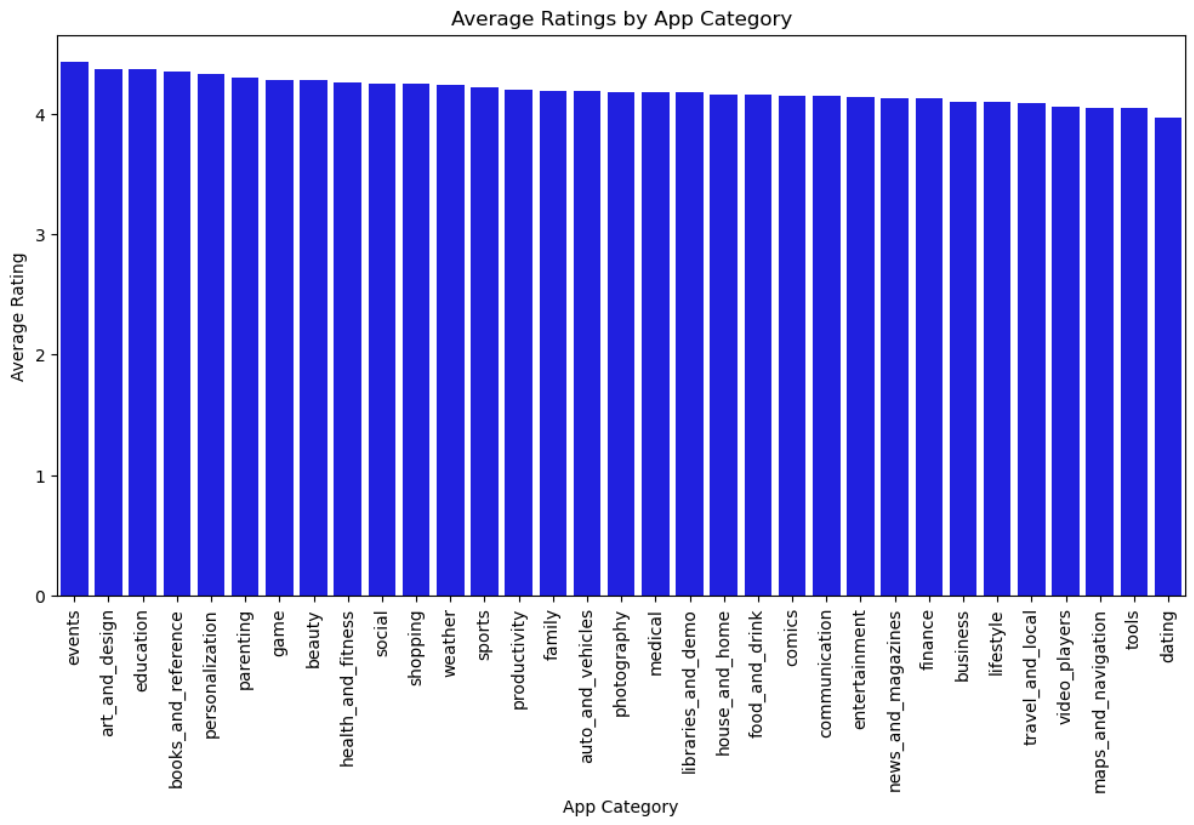
In [28]:
```
# 4. Handle Inconsistent Data:
# Standardize the 'Category' column by converting all values to lowercase
df1['Category'] = df1['Category'].str.lower()
```

In [29]:
```
# Display basic information about the cleaned dataset
print(df1.info())
```

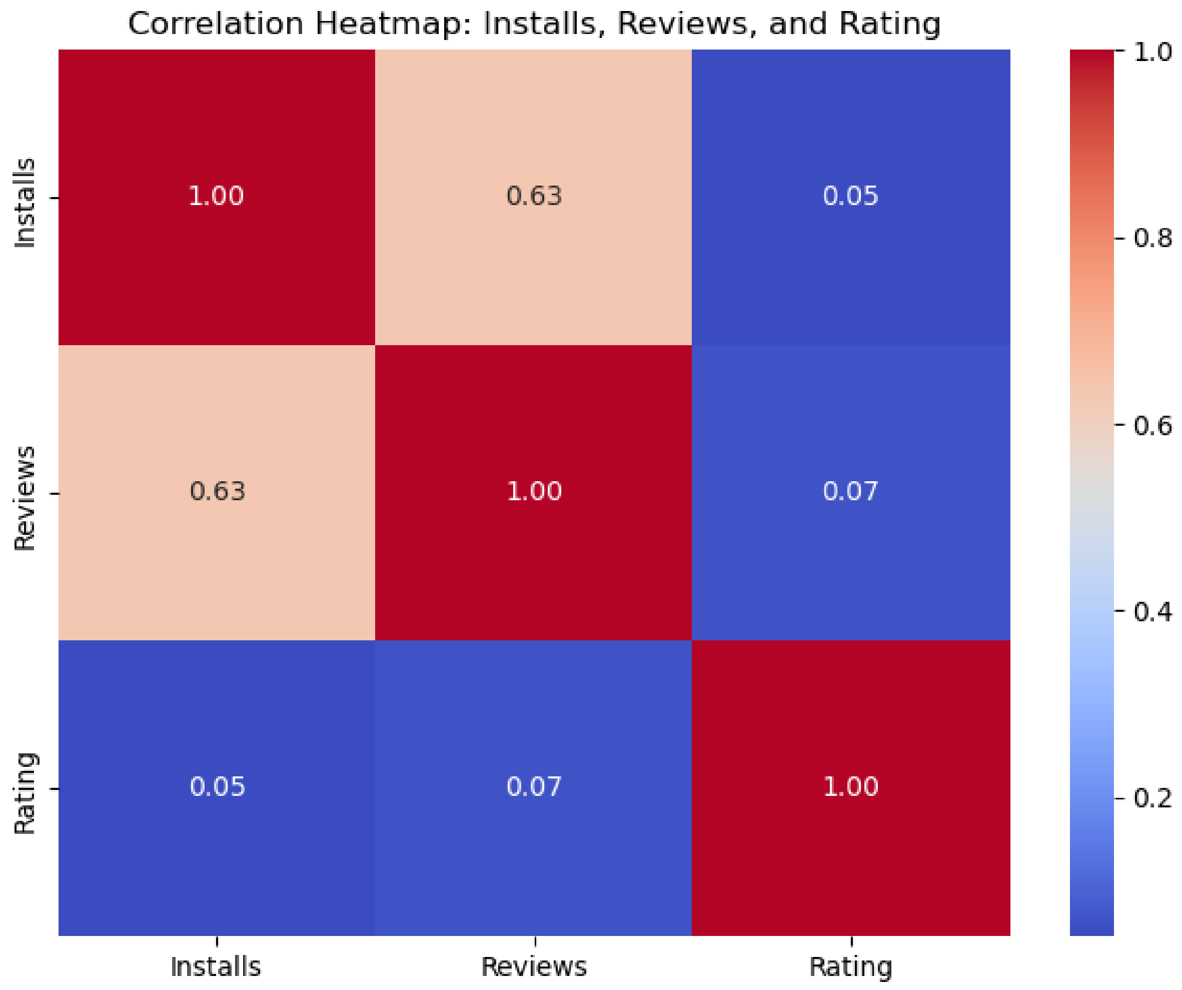
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8886 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             8886 non-null   object 
 1   Category        8886 non-null   object 
 2   Rating          8886 non-null   float64
 3   Reviews         8886 non-null   int64  
 4   Size            8886 non-null   object 
 5   Installs        8886 non-null   int64  
 6   Type            8886 non-null   object 
 7   Price           8886 non-null   object 
 8   Content Rating  8886 non-null   object 
 9   Genres          8886 non-null   object 
 10  Last Updated    8886 non-null   object 
 11  Current Ver     8886 non-null   object 
 12  Android Ver     8886 non-null   object 
dtypes: float64(1), int64(2), object(10)
memory usage: 971.9+ KB
None
```

In [32]:
```
# 1. App Categories with Higher Ratings:
average_ratings_by_category = df1.groupby('Category')['Rating'].mean().sort_values(ascending=False)
```

In [33]:
```
#Plot the average ratings by app category
plt.figure(figsize=(12, 6))
sns.barplot(x=average_ratings_by_category.index, y=average_ratings_by_category.values, color='blue')
plt.xticks(rotation=90)
plt.xlabel('App Category')
plt.ylabel('Average Rating')
plt.title('Average Ratings by App Category')
plt.show()
```



In [35]:
```
# 2. Factors Influencing App Popularity:
# Plot the correlation between 'Installs', 'Reviews', and 'Rating' using a hea
tmap
correlation_matrix = df1[['Installs', 'Reviews', 'Rating']].corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap: Installs, Reviews, and Rating')
plt.show()
```



In [38]:
```
df1["Rating"].value_counts()
```

Out[38]:
```
4.4    1031
4.3    1016
4.5     976
4.2     887
4.6     768
4.1     656
4.0     538
4.7     484
3.9     372
3.8     293
5.0     271
3.7     231
4.8     228
3.6     169
3.5     157
3.4     127
3.3     101
4.9      87
3.0      82
3.1      69
3.2      63
2.9      45
2.8      40
2.6      24
2.7      23
2.5      20
2.3      20
2.4      19
1.0      16
2.2      14
1.9      12
2.0      12
1.7       8
1.8       8
2.1       8
1.6       4
1.4       3
1.5       3
1.2       1
Name: Rating, dtype: int64
```

In [39]:
```
# Find the rating with the highest number of occurrences
highest_rating_count = df1['Rating'].value_counts().max()
highest_rating = df1['Rating'].value_counts().idxmax()

print(f"The rating with the highest number of occurrences is: {highest_rating}
({highest_rating_count} occurrences)")
```

```
The rating with the highest number of occurrences is: 4.4 (1031 occurrences)
```

Summary

By exploring the correlation between app installs, reviews, and ratings, we identified some interesting patterns:

There is a positive correlation between the number of app installs and the number of reviews. This suggests that popular apps with more installs tend to receive more user reviews. However, the correlation between app ratings and the number of installs or reviews is relatively weak. This implies that high ratings alone do not guarantee higher app popularity. By analyzing the trend of app installs and reviews over time, we observed some insights:

The number of app installs has been steadily increasing over the years, indicating a growing app market. App reviews have also seen a general upward trend, suggesting increased user engagement and feedback.

```
In [ ]:
```

```
In [ ]:
```