```
# Importing the important Python modules for our project
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


df_store = pd.read_csv("/content/Google Play Store Data/googleplaystore.csv")


df_store.head()
```

	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

```
df_store.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
```

```
df_store.duplicated()
```

```
0        False
1        False
2        False
3        False
4        False
         ...
10836    False
10837    False
10838    False
10839    False
10840    False
Length: 10841, dtype: bool
```

```
# Handling Missing Values
df_store.dropna(inplace=True)  # Drop rows with missing values

# Dealing with Duplicates
df_store.drop_duplicates(inplace=True)  # Remove duplicate rows


df_store.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   object
 4   Size            8886 non-null   object
 5   Installs        8886 non-null   object
 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), object(12)
memory usage: 971.9+ KB
```

```
df_store.duplicated().sum()
```

```
0
```

```
# Removing Outliers
z_scores = np.abs((df_store['Rating'] - df_store['Rating'].mean()) / df_store['Rating'].std())
df_store = df_store[z_scores < 3]  # Keep rows within 3 standard deviations
```

```
df_store.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   object
 4   Size            8886 non-null   object
 5   Installs        8886 non-null   object
 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), object(12)
memory usage: 971.9+ KB
```

```
# Change the data type of Installs column to numrical data type and remove the commma and the plus symbols:
df_store['Installs'] = df_store['Installs'].str.replace('+', '')  # Remove '+' sign
df_store['Installs'] = df_store['Installs'].str.replace(',', '')  # Remove ',' sign

df_store['Installs'] = df_store['Installs'].astype(int)  # Convert data type to int

# Change the data type of Reviews column to numrical data type:
df_store['Reviews'] = df_store['Reviews'].astype(int)  # Convert data type to int+
```

```
<ipython-input-10-4ad26d07a1d6>:2: FutureWarning: The default value of regex will change from True to False in a future version. In adc
  df_store['Installs'] = df_store['Installs'].str.replace('+', '')  # Remove '+' sign
```

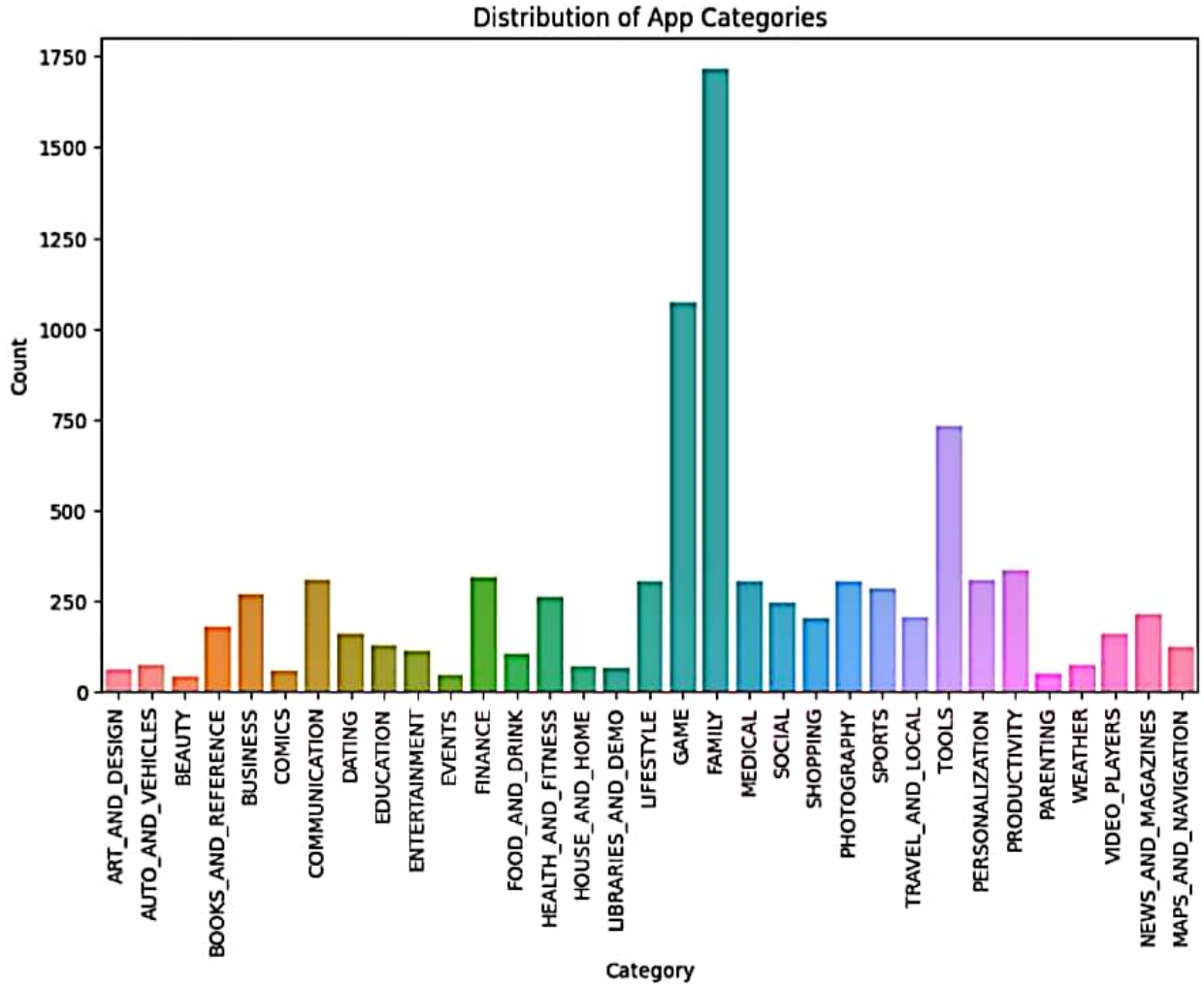
```
df_store.head()
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10000	Free	0	Everyone

```
# Bar Plot
plt.figure(figsize=(10, 6))
sns.countplot(data=df_store, x='Category')
plt.xlabel('Category')
plt.xticks( rotation=90)
```

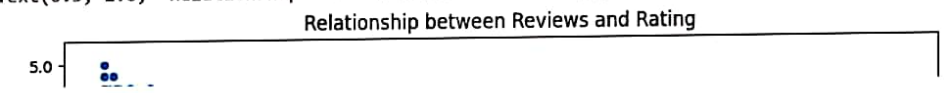
```
plt.ylabel('Count')
plt.title('Distribution of App Categories')
```

```
Text(0.5, 1.0, 'Distribution of App Categories')
```



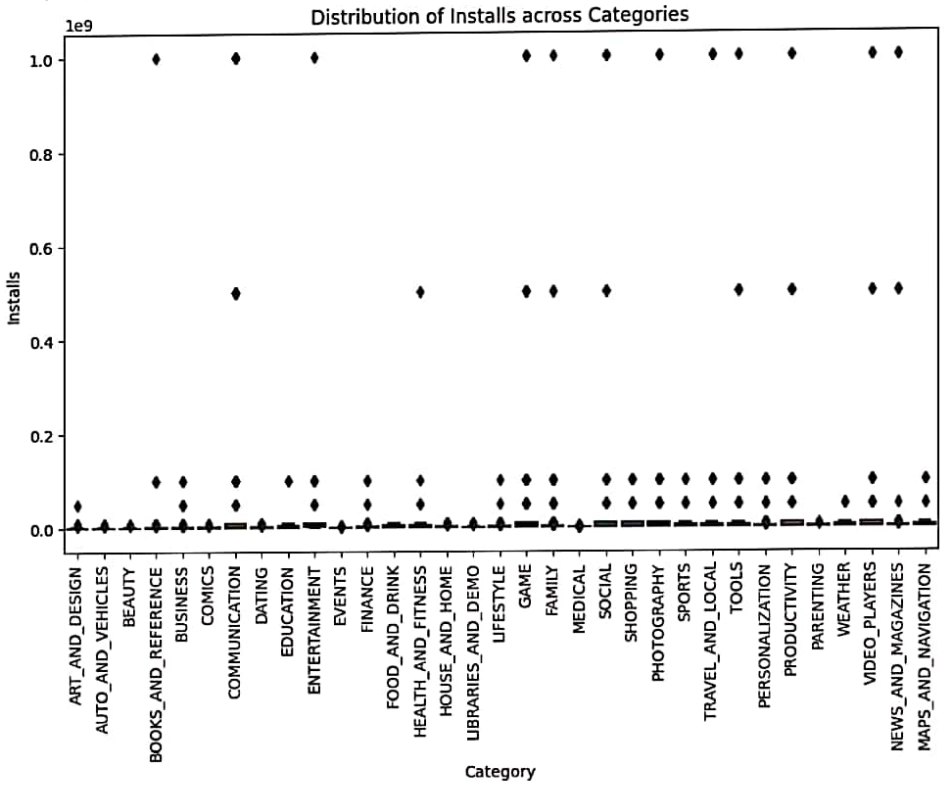

```
# Scatter Plot
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df_store, x='Reviews', y='Rating')
plt.xlabel('Number of Reviews')
plt.ylabel('Rating')
plt.title('Relationship between Reviews and Rating')
```

```
Text(0.5, 1.0, 'Relationship between Reviews and Rating')
```




```
# Box Plot
plt.figure(figsize=(10, 6))
sns.boxplot(data=df_store, x='Category', y='Installs')
plt.xlabel('Category')
plt.xticks(rotation=90)
plt.ylabel('Installs')
plt.title('Distribution of Installs across Categories')
```

```
Text(0.5, 1.0, 'Distribution of Installs across Categories')
```




```
df_store.head()
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10000	Free	0	Everyone

```
from scipy.stats import ttest_ind

# Splitting the dataset into free and paid apps
free_apps = df_store[df_store['Type'] == 'Free']
paid_apps = df_store[df_store['Type'] == 'Paid']


# Performing t-test
t_stat, p_value = ttest_ind(free_apps['Rating'], paid_apps['Rating'])
```

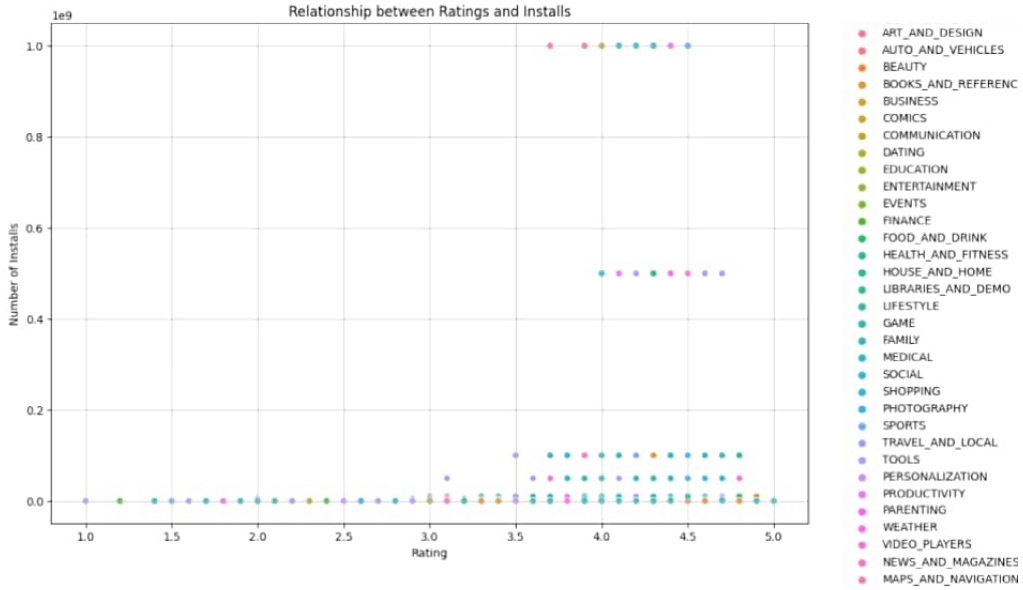
```
p_value
```

```
0.0002984015230732988
```

```
# Checking the p-value
if p_value < 0.05:
    print("Reject null hypothesis: There is a significant difference in ratings.")
else:
    print("Fail to reject null hypothesis: There is no significant difference in ratings.")
```

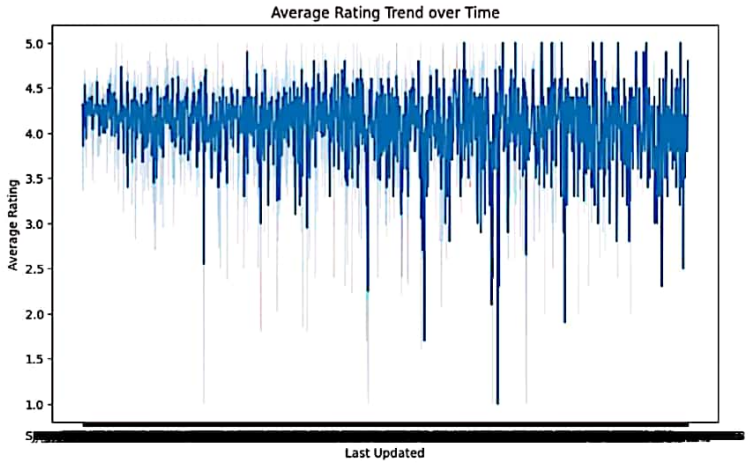
```
Reject null hypothesis: There is a significant difference in ratings.
```

```
plt.figure(figsize=(12, 8))
sns.scatterplot(data=df_store, x='Rating', y='Installs', hue='Category')
plt.grid(True)
plt.xlabel('Rating')
plt.ylabel('Number of Installs')
plt.title('Relationship between Ratings and Installs')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.show()
```




```
# Line plot of Average Rating over Time
plt.figure(figsize=(10, 6))
sns.lineplot(data=df_store, x='Last Updated', y='Rating')
plt.xlabel('Last Updated')
plt.ylabel('Average Rating')
plt.title('Average Rating Trend over Time')
```

```
Text(0.5, 1.0, 'Average Rating Trend over Time')
```




```
df_store.head()
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10000	Free	0	Everyone

```
# Convert 'Last Updated' to datetime
df_store['Last Updated'] = pd.to_datetime(df_store['Last Updated'])

# Set 'Last Updated' as the index
df_store.set_index('Last Updated', inplace=True)

# Resample 'Rating' to monthly frequency
df_store_monthly = df_store['Rating'].resample('M').mean()


plt.figure(figsize=(10, 6))
sns.lineplot(data=df_store_monthly)
plt.xlabel('Date of Last Update (Monthly)')
plt.xticks(rotation=90)
plt.ylabel('Average Rating')
plt.title('Average Rating Trend over Time (Monthly)')
plt.show()
```