Performing EDA on Google Play Store Dataset

Exploratory data analysis (EDA):

- EDA is an important step in any Data Analysis or Data Science project.
- It is the process of investigating the dataset to discover patterns, and anomalies (outliers), and form hypotheses based on our understanding of the dataset.
- It involves generating summary statistics for numerical data in the dataset and creating various graphical representations to understand the data better.

About Dataset:

- In this dataset, we will use Pandas, Numpy, matplotlib and seaborn libraries.
- The dataset we are using here is the Google Playstore dataset, which contains details about the Apps in the play store, there are more than 10,0000+ Apps in the play store. The size of the dataset is 210Mb, which is taken from Kaggle.

Importing the libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

Data Preparation and Cleaning :

- Here we will load the dataset into the data frame, view the columns and rows of the data, perform descriptive statistics to know better about the features inside the dataset, write the observations, finding the missing values and duplicate rows.

In [2]:
```
# Loading the dataset into a data frame and reading the dataset using pandas.
df = pd.read_csv('Google-Playstore.csv')
```

In [3]:
```
df.head()
```

Out[3]:

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

5 rows × 24 columns

In [4]:
```
#The dataset having 2312944 rows and 24 columns
df.shape
```

Out[4]:
```
(2312944, 24)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['App Name', 'App Id', 'Category', 'Rating', 'Rating Count', 'Installs',
       'Minimum Installs', 'Maximum Installs', 'Free', 'Price', 'Currency',
       'Size', 'Minimum Android', 'Developer Id', 'Developer Website',
       'Developer Email', 'Released', 'Last Updated', 'Content Rating',
       'Privacy Policy', 'Ad Supported', 'In App Purchases', 'Editors Choice',
       'Scraped Time'],
      dtype='object')
```

Descriptive Statistics:

In [6]:
```
df.describe()
```

Out[6]:

	Rating	Rating Count	Minimum Installs	Maximum Installs	Price
count	2.290061e+06	2.290061e+06	2.312837e+06	2.312944e+06	2.312944e+06
mean	2.203152e+00	2.864839e+03	1.834452e+05	3.202017e+05	1.034992e-01
std	2.106223e+00	2.121626e+05	1.513144e+07	2.355495e+07	2.633127e+00
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	0.000000e+00	0.000000e+00	5.000000e+01	8.400000e+01	0.000000e+00
50%	2.900000e+00	6.000000e+00	5.000000e+02	6.950000e+02	0.000000e+00
75%	4.300000e+00	4.200000e+01	5.000000e+03	7.354000e+03	0.000000e+00
max	5.000000e+00	1.385576e+08	1.000000e+10	1.205763e+10	4.000000e+02

Check for Null values:

In [7]:
```
df.isnull().sum().sort_values(ascending = False)
```

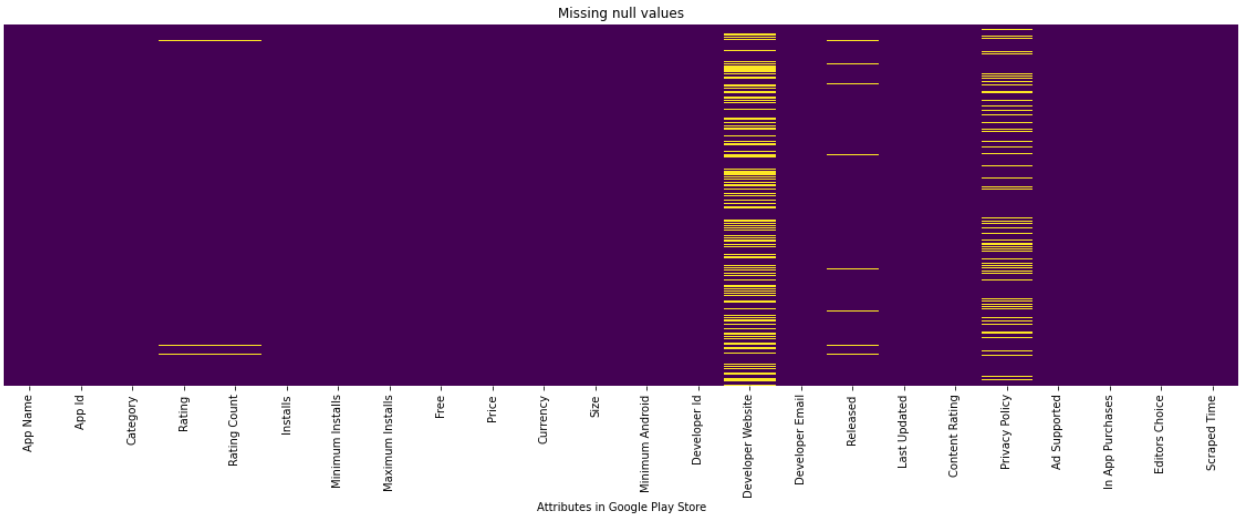
Out[7]:
```
Developer Website    760835
Privacy Policy       420953
Released              71053
Rating                22883
Rating Count          22883
Minimum Android        6530
Size                    196
Currency                135
Installs                107
Minimum Installs        107
Developer Id             33
Developer Email          31
App Name                  2
App Id                    0
Price                     0
Free                      0
Maximum Installs          0
Last Updated              0
Content Rating            0
Category                  0
Ad Supported              0
In App Purchases          0
Editors Choice            0
Scraped Time              0
dtype: int64
```

Exploratory Analysis and Visualizations :

- Here we will explore each feature(both numerical and Categorical), plot visualizations, deal with the null values and write the observations.

In [8]:
```
# Missing values visualization
plt.figure(figsize = (20,6))
sns.heatmap(df.isnull(),yticklabels = False, cbar = False, cmap = 'viridis')
plt.xlabel("Attributes in Google Play Store")
plt.title("Missing null values")
```

Out[8]:
```
Text(0.5, 1.0, 'Missing null values')
```

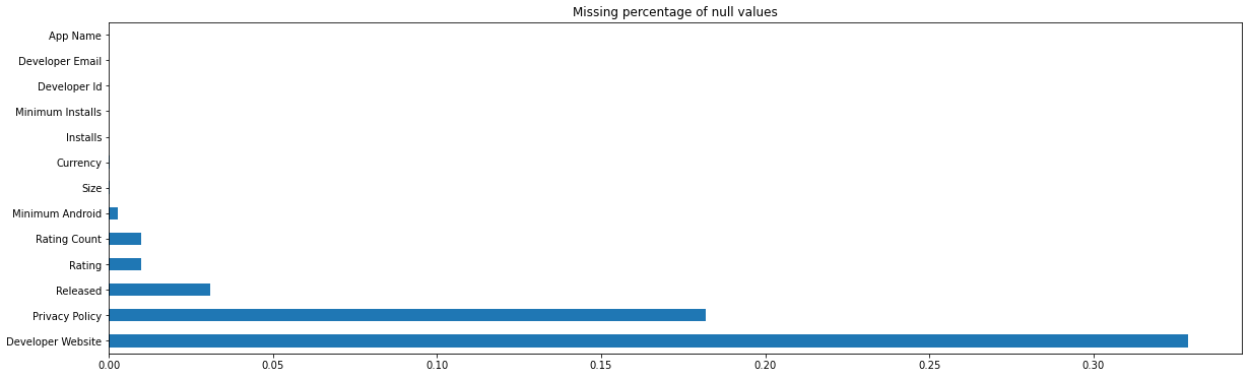


In [9]:
```
# Percentage of the missing values in the dataframe
missing_percentage = df.isnull().sum().sort_values(ascending = False)/len(df)
missing_percentage
```

Out[9]:
```
Developer Website    3.289466e-01
Privacy Policy       1.819988e-01
Released             3.071972e-02
Rating               9.893452e-03
Rating Count         9.893452e-03
Minimum Android      2.823242e-03
Size                 8.474049e-05
Currency             5.836717e-05
Installs             4.626139e-05
Minimum Installs     4.626139e-05
Developer Id         1.426753e-05
Developer Email      1.340283e-05
App Name             8.646988e-07
App Id               0.000000e+00
Price                0.000000e+00
Free                 0.000000e+00
Maximum Installs     0.000000e+00
Last Updated         0.000000e+00
Content Rating       0.000000e+00
Category             0.000000e+00
Ad Supported         0.000000e+00
In App Purchases     0.000000e+00
Editors Choice       0.000000e+00
Scraped Time         0.000000e+00
dtype: float64
```

In [10]:
```
# Missing values plot in percentage
missing_percentage = missing_percentage[missing_percentage != 0] # Only the missing data
plt.figure(figsize = (20,6))
missing_percentage.plot(kind = 'barh')
plt.title("Missing percentage of null values")
```

Out[10]:
```
Text(0.5, 1.0, 'Missing percentage of null values')
```



Observation :

Dealing with the null values

1. The columns having the highest percentage of null values are:

   - Developer Website
   - Privacy Policy

   These two columns have highest percentage of null values, we can drop these columns as these are not so important for the analysis.

2. Lowest null value columns: Since the count is not so high for missing values and can be safely neglected, we can either drop or safely fill the null values. So, let's drop the lowest null values from the below listed columns which can be used in future steps or for analysis:
   - App Name
   - Size
   - Currency
   - Installs
   - Minimum installs
   - Developer id
   - Developer Email

3. Rating, Rating count, Minimum Android, Released are important columns so it is good to fill the null values.
   - Rating
   - Rating count
   - Minimum Android
   - Released

Treating the Null values:

In [11]:
```
# From Observation 1, dropping the columns which has highest percentage of null values.
df.drop(['Developer Website','Privacy Policy'], axis=1, inplace = True)
df.head()
```

Out[11]:

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

5 rows × 22 columns

In [12]:
```
# From observation 2 dropping the null values of the columns that are listed.
df.dropna(subset=['App Name', 'Installs','Minimum Installs', 'Currency','Size', 'Develop
          inplace = True)
```

- From Observation 3, we will fill the null values for Rating, Rating Count, Minimum Android, Released columns by their mode values to perform the analysis in the future steps.

In [13]:
```
# Filling null values for 'Rating' column by mode value
df['Rating'].mode()
```

Out[13]:
```
0    0.0
Name: Rating, dtype: float64
```

In [14]:
```
df['Rating'].fillna(df['Rating'].mode()[0], inplace = True)
df.Rating.isnull().sum()
```

Out[14]:
```
0
```

In [15]:
```
# Filling null values for 'Rating Count' column by mode value
df['Rating Count'].mode()
```

Out[15]:
```
0    0.0
Name: Rating Count, dtype: float64
```

In [16]:
```
df['Rating Count'].fillna(df['Rating Count'].mode()[0], inplace = True)
df['Rating Count'].isnull().sum()
```

Out[16]:
```
0
```

In [17]:
```
# Filling null values for 'Minimum Android' column by mode value
df['Minimum Android'].mode()
```

Out[17]:
```
0    4.1 and up
Name: Minimum Android, dtype: object
```

In [18]:
```
df['Minimum Android'].fillna(df['Minimum Android'].mode()[0], inplace = True)
df['Minimum Android'].isnull().sum()
```

Out[18]:
```
0
```

In [19]:
```
# Filling null values for 'Released' column by mode value
df['Released'].mode()
```

Out[19]:
```
0    Jun 16, 2020
Name: Released, dtype: object
```

In [20]:
```
df['Released'].fillna(df['Released'].mode()[0], inplace = True)
df['Released'].isnull().sum()
```

Out[20]:
```
0
```

In [21]:
```
df.isnull().sum()
```

Out[21]:
```
App Name            0
App Id              0
Category            0
Rating              0
Rating Count        0
Installs            0
Minimum Installs    0
Maximum Installs    0
Free                0
Price               0
Currency            0
Size                0
Minimum Android     0
Developer Id        0
Developer Email     0
Released            0
Last Updated        0
Content Rating      0
Ad Supported        0
In App Purchases    0
Editors Choice      0
Scraped Time        0
dtype: int64
```

Checking for duplicates

In [22]:
```
df.duplicated().sum()
```

Out[22]:
```
0
```

In [23]:
```
df[df.duplicated()]
```

Out[23]:

App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	...	Minimum Android	Developer Id	Devel E

0 rows × 22 columns

- No duplicate values found in the dataframe.

Let's explore the numerical columns :

Installs Column:

- 'Installs' feature, this feature has the information about how many installations has done for each Application in dataset.

In [24]:
```
df['Installs'].unique()
```

Out[24]:
```
array(['10+', '5,000+', '50+', '100+', '1,000+', '500+', '50,000+',
       '10,000+', '1+', '500,000+', '100,000+', '5+', '10,000,000+',
       '1,000,000+', '5,000,000+', '0+', '100,000,000+', '50,000,000+',
       '1,000,000,000+', '500,000,000+', '5,000,000,000+',
       '10,000,000,000+'], dtype=object)
```

- This column is in the object type, we need to change it into the int type

In [25]:
```
# removes the + symbol
df.Installs = df.Installs.str.replace('+','')
# replace the commas in the install column
df.Installs = df.Installs.str.replace(',','')
# converting it to the int type
df['Installs'] = pd.to_numeric(df['Installs'])
```

In [26]:
```
df["Installs"].unique()
```

Out[26]:
```
array([         10,        5000,          50,         100,        1000,
                500,       50000,       10000,           1,      500000,
             100000,           5,    10000000,     1000000,     5000000,
                  0,   100000000,    50000000,  1000000000,   500000000,
         5000000000, 10000000000], dtype=int64)
```

- Hence, we have successfully removed the "+" symbol, ","(comma) and then converted it into the integer type.

Currency Column:

In [27]:
```
df["Currency"].unique()
```

Out[27]:
```
array(['USD', 'XXX', 'CAD', 'EUR', 'INR', 'VND', 'GBP', 'BRL', 'KRW',
       'TRY', 'RUB', 'SGD', 'AUD', 'PKR', 'ZAR'], dtype=object)
```

Size Column:

- 'Size' feature, this feature has the information about the size of each Application in the dataset

In [28]:
```
df["Size"].unique()
```

Out[28]:
```
array(['10M', '2.9M', '3.7M', ..., '405M', '3.2k', '512M'], dtype=object)
```

- The size of data can be in GB, MB and KB. So, let's convert the data into MB's size

In [29]:
```
# Removing the 'M'
df['Size'] = df['Size'].apply(lambda x: str(x).replace("M", "") if "M" in str(x) else x)
```

- It shows that there is a mismatched value with the data i.e. it contains the value '1,018'. We can drop it or We can assume it as may be a "." (dot). So, let assume it as "." (dot) for now and replace the "," with the "." (dot).

In [30]:
```
df['Size'] = df['Size'].apply(lambda x: str(x).replace(",", ".") if "," in str(x) else x
```

In [31]:
```
# Conversion of KB to MB
df['Size'] = df['Size'].apply(lambda x: float(str(x).replace("k", "")) / 1000 if "k" in
```

- It shows that there is another mismatch value in the dataset which is 'Varies with device' - it means the size of the App may vary with the device. So, let's assume it as 0.

In [32]:
```
df['Size'] = df['Size'].apply(lambda x: str(x).replace("Varies with device", "0") if "Va
```

In [33]:
```
# Removing 'G' and converting GB's to MB's
df['Size'] = df['Size'].apply(lambda x: float(str(x).replace("G", "")) * 1000 if "G" in
```

In [34]:
```
# Converting the Size column into float type
df['Size'] = df['Size'].apply(lambda x: float(x))
df.dtypes['Size']
```

Out[34]:
```
dtype('float64')
```

Let's explore other columns:

In [35]:
```
df["Minimum Installs"].head()
```

Out[35]:
```
0      10.0
1    5000.0
2      50.0
3      10.0
4     100.0
Name: Minimum Installs, dtype: float64
```

In [36]:
```
# Changing float type to integer (makes more sense)
df['Minimum Installs'] = df['Minimum Installs'].astype(int)
df["Minimum Installs"].head()
```

Out[36]:
```
0      10
1    5000
2      50
3      10
4     100
Name: Minimum Installs, dtype: int32
```

In [37]:
```
df['Currency'].head()
```

Out[37]:
```
0    USD
1    USD
2    USD
3    USD
4    USD
Name: Currency, dtype: object
```

In [38]:
```
df['Currency'].value_counts()
```

Out[38]:
```
USD    2311287
XXX       1236
EUR          6
INR          5
GBP          3
CAD          2
VND          1
BRL          1
KRW          1
TRY          1
RUB          1
SGD          1
AUD          1
PKR          1
ZAR          1
Name: Currency, dtype: int64
```

- Since USD is significantly dominated buying the apps from Google Playstore. So, we can drop this column because it doesn't add any value.

In [39]:
```
df.drop('Currency', axis=1, inplace=True)
df.columns
```

Out[39]:
```
Index(['App Name', 'App Id', 'Category', 'Rating', 'Rating Count', 'Installs',
       'Minimum Installs', 'Maximum Installs', 'Free', 'Price', 'Size',
       'Minimum Android', 'Developer Id', 'Developer Email', 'Released',
       'Last Updated', 'Content Rating', 'Ad Supported', 'In App Purchases',
       'Editors Choice', 'Scraped Time'],
      dtype='object')
```

In [40]:
```
df["Minimum Android"]
```

Out[40]:
```
0            7.1 and up
1            5.0 and up
2          4.0.3 and up
3          4.0.3 and up
4            4.1 and up
                ...
2312939      4.1 and up
2312940      4.1 and up
2312941      5.0 and up
2312942      5.0 and up
2312943      5.0 and up
Name: Minimum Android, Length: 2312548, dtype: object
```

In [41]:
```
df['Content Rating']
```

Out[41]:
```
0          Everyone
1          Everyone
2          Everyone
3          Everyone
4          Everyone
              ...
2312939        Teen
2312940    Everyone
2312941    Everyone
2312942    Everyone
2312943    Everyone
Name: Content Rating, Length: 2312548, dtype: object
```

In [42]:
```
df["Released"]
```

Out[42]:
```
0          Feb 26, 2020
1          May 21, 2020
2           Aug 9, 2019
3          Sep 10, 2018
4          Feb 21, 2020
                ...
2312939    Jun 16, 2020
2312940    Jan 17, 2018
2312941    Aug 19, 2018
2312942     Aug 1, 2016
2312943     Aug 9, 2019
Name: Released, Length: 2312548, dtype: object
```

In [43]:
```
df["Last Updated"]
```

Out[43]:
```
0          Feb 26, 2020
1          May 06, 2021
2          Aug 19, 2019
3          Oct 13, 2018
4          Nov 12, 2018
               ...
2312939    Jun 01, 2021
2312940    Feb 02, 2018
2312941    Aug 19, 2018
2312942    May 05, 2021
2312943    Aug 19, 2019
Name: Last Updated, Length: 2312548, dtype: object
```

In [44]:
```
df["Scraped Time"]
```

Out[44]:
```
0          2021-06-15 20:19:35
1          2021-06-15 20:19:35
2          2021-06-15 20:19:35
3          2021-06-15 20:19:35
4          2021-06-15 20:19:35
                  ...
2312939    2021-06-16 12:59:18
2312940    2021-06-16 12:59:19
2312941    2021-06-16 12:59:19
2312942    2021-06-16 12:59:19
2312943    2021-06-16 12:59:19
Name: Scraped Time, Length: 2312548, dtype: object
```

Free Column:

- This column has the information about the type of each Application in the dataset which says whether it is Paid or Free for installation.

In [45]:
```
df['Free']
```

Out[45]:
```
0          True
1          True
2          True
3          True
4          True
           ...
2312939    True
2312940    True
2312941    True
2312942    True
2312943    True
Name: Free, Length: 2312548, dtype: bool
```

Observation:

- This column is a bool type, so let's create the column 'Type' for free and paid Apps by using the Free Column for further analysis, so that it would be easy for us to use this feature while dealing with the Paid and Free Apps.
- Drop the column 'Free' as we already created the column 'Type' using 'Free' column information.

In [46]:
```
# Creating 'Type' column using 'Free' column information.
df['Type'] = np.where(df['Free'] == True,'Free','Paid')
df.drop(["Free"],axis = 1, inplace = True)
```

In [47]:
```
df.Type.unique()
```

Out[47]:
```
array(['Free', 'Paid'], dtype=object)
```

Content Rating Column:

- 'Content Rating' column has some Categories like — Teenagers, Adults, Matures , Everyone, Everyone10+, Unrated, which gives the information of various age groups.

In [48]:
```
df['Content Rating'].unique()
```

Out[48]:
```
array(['Everyone', 'Teen', 'Mature 17+', 'Everyone 10+',
       'Adults only 18+', 'Unrated'], dtype=object)
```

- Now, we can make these Categories to a simple 3 Categories for better understanding: Everyone, Teen, Adults.

In [49]:
```
df["Content Rating"]=df["Content Rating"].replace("Unrated","Everyone")
df["Content Rating"]=df["Content Rating"].replace("Everyone 10+","Teen")
df["Content Rating"]=df["Content Rating"].replace("Mature 17+","Adults")
df["Content Rating"]=df["Content Rating"].replace("Adults only 18+","Adults")
```

In [50]:
```
df['Content Rating'].unique()
```

Out[50]:
```
array(['Everyone', 'Teen', 'Adults'], dtype=object)
```

In [51]:
```
df["Content Rating"].value_counts()
```

Out[51]:
```
Everyone    2021942
Teen         230192
Adults        60414
Name: Content Rating, dtype: int64
```

Rating Count:

- This column gives the information about how much rating count for each application

In [52]:
```
df['Rating Count'].unique()
```

Out[52]:
```
array([0.0000e+00, 6.4000e+01, 5.0000e+00, ..., 8.7553e+04, 7.5960e+04,
       7.8351e+04])
```

- Let's change the Rating Count to certain ranges for better understanding of data.

In [53]:
```
df["Rating Count"].max()
```

Out[53]:
```
138557570.0
```

- Here we have the count of rating, we can categorize the count like 0 counts as 'No rating', <1000 as 'less than 10k', 'between 10k and 500k', 'more than 500k'.

In [54]:
```
df["Rating Type"] = "No Rating Provided"
df.loc[(df["Rating Count"] > 0) & (df["Rating Count"] <= 10000.0), "Rating Type"] = "Les
df.loc[(df["Rating Count"] > 10000) & (df["Rating Count"] <= 500000.0), "Rating Type"] =
df.loc[(df["Rating Count"] > 500000) & (df["Rating Count"] <= 138557570.0), "Rating Type
df["Rating Type"].value_counts()
```

Out[54]:
```
Less than 10k           1192801
No Rating Provided      1082303
Between 10k and 500k      35779
More than 500k             1665
Name: Rating Type, dtype: int64
```

In [55]:
```
df["Rating Type"]
```

Out[55]:
```
0            No Rating Provided
1                 Less than 10k
2            No Rating Provided
3                 Less than 10k
4            No Rating Provided
                     ...
2312939    Between 10k and 500k
2312940      No Rating Provided
2312941      No Rating Provided
2312942           Less than 10k
2312943           Less than 10k
Name: Rating Type, Length: 2312548, dtype: object
```

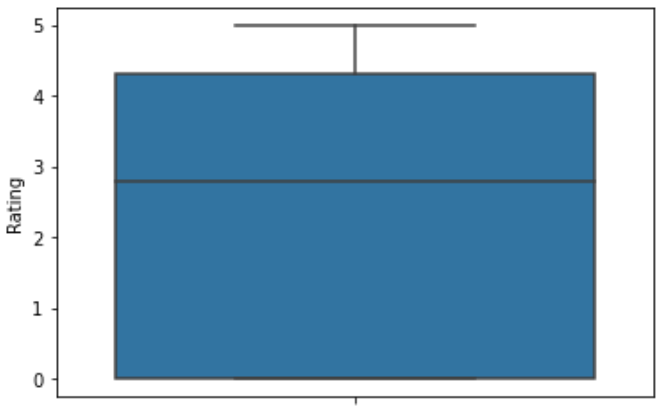
In [56]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2312548 entries, 0 to 2312943
Data columns (total 22 columns):
 #   Column            Dtype  
---  ------            -----  
 0   App Name          object 
 1   App Id            object 
 2   Category          object 
 3   Rating            float64
 4   Rating Count      float64
 5   Installs          int64  
 6   Minimum Installs  int32  
 7   Maximum Installs  int64  
 8   Price             float64
 9   Size              float64
 10  Minimum Android   object 
 11  Developer Id      object 
 12  Developer Email   object 
 13  Released          object 
 14  Last Updated      object 
 15  Content Rating    object 
 16  Ad Supported      bool   
 17  In App Purchases  bool   
 18  Editors Choice    bool   
 19  Scraped Time      object 
 20  Type              object 
 21  Rating Type       object 
dtypes: bool(3), float64(4), int32(1), int64(2), object(12)
memory usage: 350.7+ MB
```

Outliers Detection:

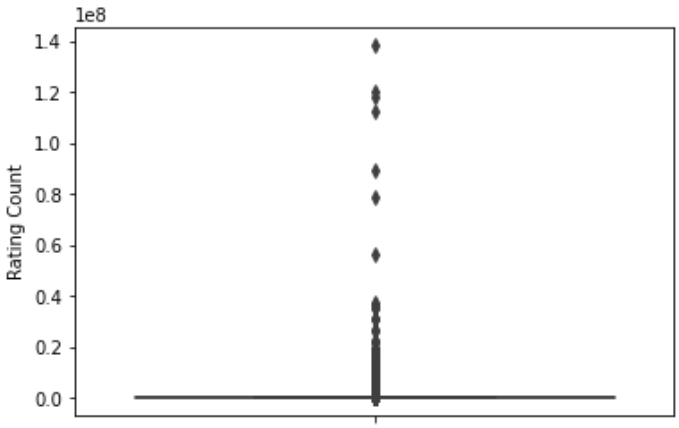
- Outliers are observations in a dataset that don't fit in some way.
- These are the observations that are far from the rest of the observations or the center of mass of observations.

In [57]:
```
sns.boxplot(y='Rating',data=df)

plt.show()
```



- 'Rating' column has no outliers.

In [58]:
```
sns.boxplot(y='Rating Count',data=df)

plt.show()
```

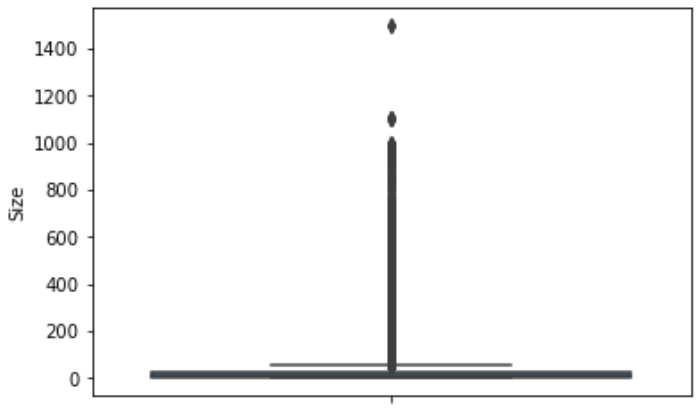


In [59]:
```
df['Rating Count'].describe()
```

Out[59]:
```
count    2.312548e+06
mean     2.836975e+03
std      2.111287e+05
min      0.000000e+00
25%      0.000000e+00
50%      6.000000e+00
75%      4.100000e+01
max      1.385576e+08
Name: Rating Count, dtype: float64
```

- Clearly we can see that 'Rating Count' column has many outliers.

In [60]:
```
sns.boxplot(y='Size',data=df)

plt.show()
```



In [61]:
```
df['Size'].describe()
```

Out[61]:
```
count    2.312548e+06
mean     1.858562e+01
std      2.387231e+01
min      0.000000e+00
25%      4.500000e+00
50%      9.700000e+00
75%      2.400000e+01
max      1.500000e+03
Name: Size, dtype: float64
```

- Clearly we can see that 'Size' column has outliers.

Outliers Removal:

- There are several ways to treat outliers in a dataset, depending on the nature of the outliers and the problem being solved. The most common ways of treating outlier values are by Trimming, Capping and Discretization or Binning.
- We will use capping method to treat outliers for this dataset. Capping is a technique where we cap our outliers data and make the limit i.e, above a particular value or less than that value, all the values will be considered as outliers, and the number of outliers in the dataset gives that capping number.
- For example, if we're working on the price feature, we might find that apps above a certain price level behave similarly to those with a lower price. In this case, we can cap the price value at a level that keeps that intact and accordingly treat the outliers.

IQR Method:

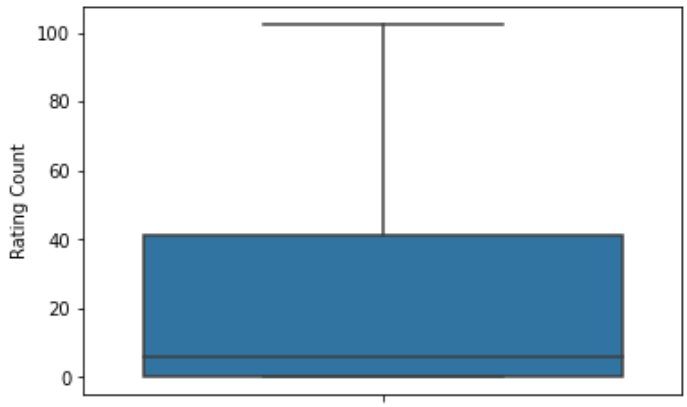
- We cap the values between the upper bound and lower bound
- upper bound= q3 + 1.5*iqr
- lower bound=q1 - 1.5*iqr
- iqr = q3 - q1
- q3 = 75th percentile value
- q1 = 25th percentile value

In [62]:
```
def outlier(data):

    q1=data.quantile(0.25)
    q3=data.quantile(0.75)

    iqr=q3-q1
    upper_bound=q3+1.5*iqr
    lower_bound=q1-1.5*iqr

    return data.clip(upper_bound,lower_bound)
```

In [63]:
```
df['Rating Count']=outlier(df['Rating Count'])
```

In [64]:
```
sns.boxplot(y='Rating Count',data=df)
```

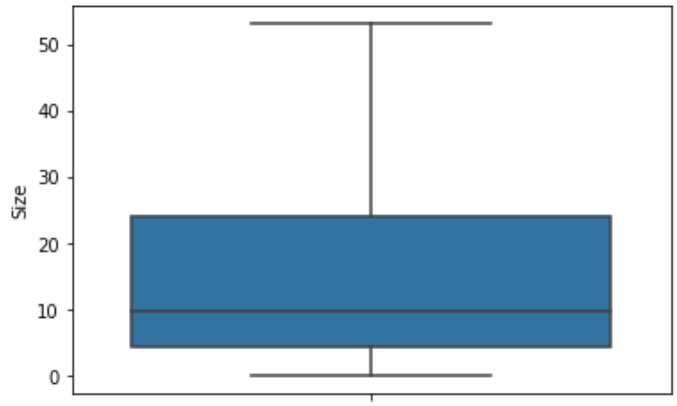
Out[64]:
```
<AxesSubplot:ylabel='Rating Count'>
```



- We have successfully removed outliers in 'Rating Count' column

In [65]:
```
df['Size']=outlier(df['Size'])
sns.boxplot(y='Size',data=df)
```

Out[65]:
```
<AxesSubplot:ylabel='Size'>
```



- Therefore we have successfully removed all the outliers for the 'Rating Count' and 'Size' column. Hence, we can now proceed to perform analysis for this dataset.

Analysis:

1. Top 10 Categories that are installed from the Google Play Store.

In [66]:
```
df["Category"].unique()
```

Out[66]:
```
array(['Adventure', 'Tools', 'Productivity', 'Communication', 'Social',
       'Libraries & Demo', 'Lifestyle', 'Personalization', 'Racing',
       'Maps & Navigation', 'Travel & Local', 'Food & Drink',
       'Books & Reference', 'Medical', 'Puzzle', 'Entertainment',
       'Arcade', 'Auto & Vehicles', 'Photography', 'Health & Fitness',
       'Education', 'Shopping', 'Board', 'Music & Audio', 'Sports',
       'Beauty', 'Business', 'Educational', 'Finance', 'News & Magazines',
       'Casual', 'Art & Design', 'House & Home', 'Card', 'Events',
       'Trivia', 'Weather', 'Strategy', 'Word', 'Video Players & Editors',
       'Action', 'Simulation', 'Music', 'Dating', 'Role Playing',
       'Casino', 'Comics', 'Parenting'], dtype=object)
```

In [67]:
```
top_category = df.Category.value_counts().reset_index().rename(columns = {"Category":"Co
top_category
```

Out[67]:

	Category	Count
0	Education	241068
1	Music & Audio	154897
2	Tools	143971
3	Business	143749
4	Entertainment	138261
5	Lifestyle	118321
6	Books & Reference	116716
7	Personalization	89207
8	Health & Fitness	83497
9	Productivity	79681
10	Shopping	75240
11	Food & Drink	73918
12	Travel & Local	67279
13	Finance	65450
14	Arcade	53779
15	Puzzle	51151
16	Casual	50790
17	Communication	48157
18	Sports	47473
19	Social	44724
20	News & Magazines	42799
21	Photography	35552
22	Medical	32063
23	Action	27539
24	Maps & Navigation	26721
25	Simulation	23268
26	Adventure	23193
27	Educational	21302
28	Art & Design	18536
29	Auto & Vehicles	18276
30	House & Home	14368
31	Video Players & Editors	14014
32	Events	12839
33	Trivia	11793
34	Beauty	11771
35	Board	10587
36	Racing	10360
37	Role Playing	10008

	Category	Count
38	Word	8627
39	Strategy	8517
40	Card	8175
41	Weather	7244
42	Dating	6522
43	Libraries & Demo	5196
44	Casino	5076
45	Music	4201
46	Parenting	3810
47	Comics	2862

In [68]:
```
category_installs = df.groupby(["Category"])[["Installs"]].sum()
category_installs
```

Out[68]:

Category	Installs
Action	17399855328
Adventure	5390108856
Arcade	14501230855
Art & Design	1116398902
Auto & Vehicles	1594745418
Beauty	404328787
Board	3290826164
Books & Reference	4819175179
Business	5236661902
Card	1989318718
Casino	1249613989
Casual	16836783725
Comics	397602344
Communication	43216592414
Dating	643267317
Education	5983815847
Educational	4730801093
Entertainment	17108397933
Events	73016139
Finance	6158477105
Food & Drink	1623441588
Health & Fitness	4586600201
House & Home	463801748
Libraries & Demo	289118035
Lifestyle	5997921096
Maps & Navigation	2683038830
Medical	657119650
Music	1630600862
Music & Audio	14239401798
News & Magazines	4548786357
Parenting	244133966
Personalization	9252971243
Photography	18998958963
Productivity	28313922253
Puzzle	10796411002
Racing	9208081547
Role Playing	4765977334

Category	Installs
Shopping	7108586169
Simulation	11689990564
Social	17165994565
Sports	7954121388
Strategy	4156174163
Tools	71440271217
Travel & Local	9726222850
Trivia	1188962483
Video Players & Editors	18591154109
Weather	2773800062
Word	2029143957

In [69]:
```
top_category_installs = pd.merge(top_category, category_installs, on = "Category")
top_category_installs.head(10)
```

Out[69]:

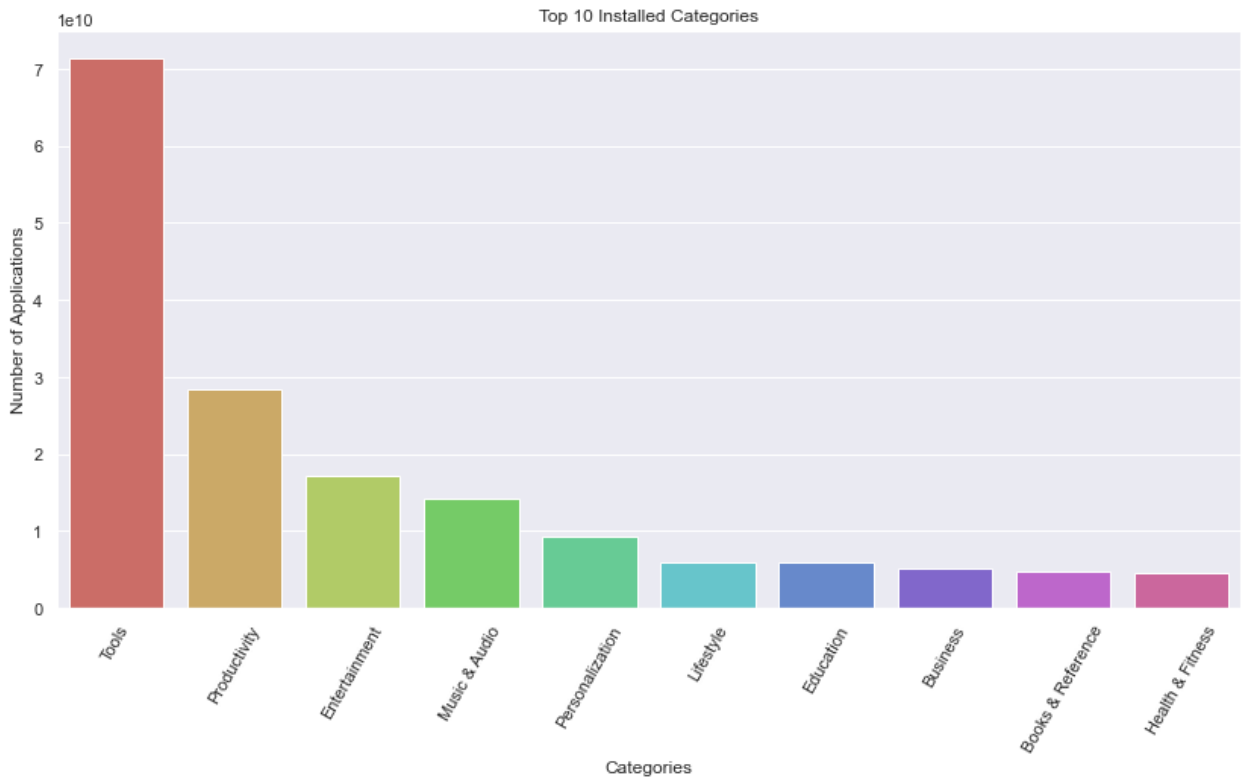
	Category	Count	Installs
0	Education	241068	5983815847
1	Music & Audio	154897	14239401798
2	Tools	143971	71440271217
3	Business	143749	5236661902
4	Entertainment	138261	17108397933
5	Lifestyle	118321	5997921096
6	Books & Reference	116716	4819175179
7	Personalization	89207	9252971243
8	Health & Fitness	83497	4586600201
9	Productivity	79681	28313922253

In [70]:
```
top_10_categories_installs = top_category_installs.head(10).sort_values(by = ["Installs"]
top_10_categories_installs
```

Out[70]:

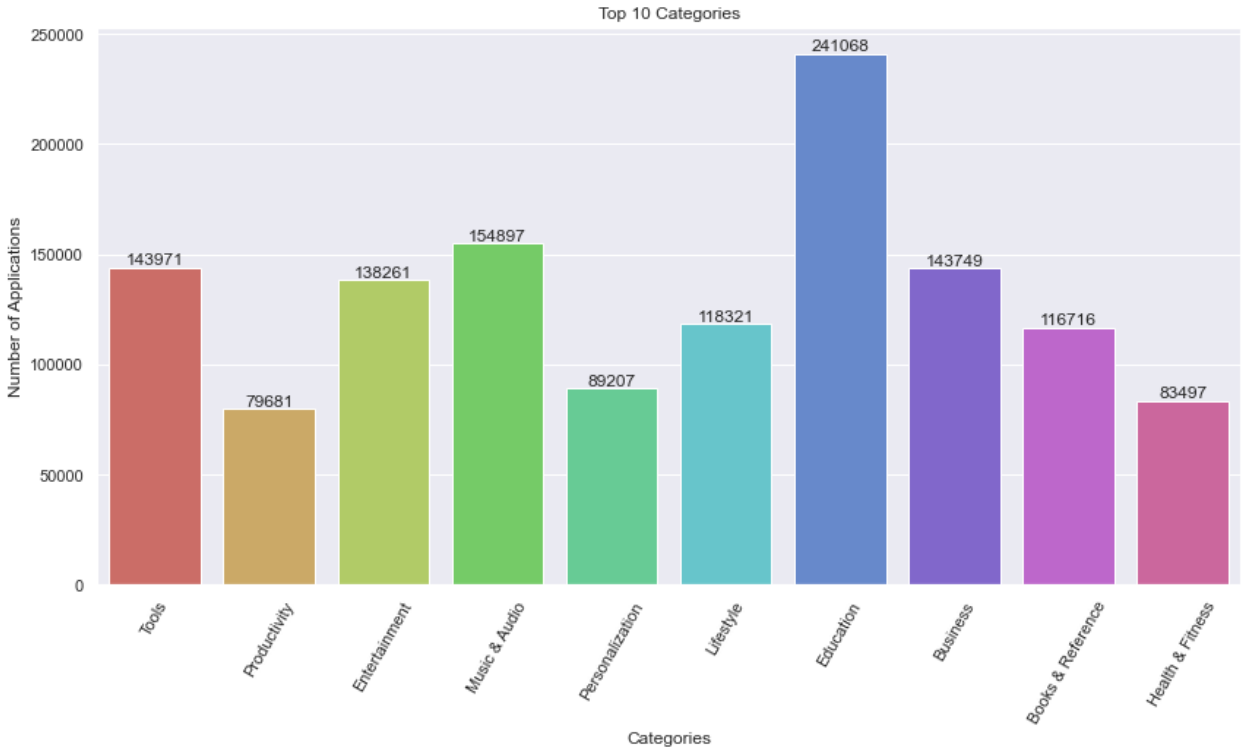
	Category	Count	Installs
2	Tools	143971	71440271217
9	Productivity	79681	28313922253
4	Entertainment	138261	17108397933
1	Music & Audio	154897	14239401798
7	Personalization	89207	9252971243
5	Lifestyle	118321	5997921096
0	Education	241068	5983815847
3	Business	143749	5236661902
6	Books & Reference	116716	4819175179
8	Health & Fitness	83497	4586600201

In [71]:
```
sns.set_theme(style="darkgrid")
plt.figure(figsize = (14,7))
plt.xticks(rotation = 60)
sns.barplot(x = top_10_categories_installs.Category, y = top_10_categories_installs.Insta
plt.xlabel("Categories")
plt.ylabel("Number of Applications")
plt.title("Top 10 Installed Categories")
plt.show()
```



2. Categories that are mostly installed in the top 10 Categories.

In [72]:
```
sns.set_theme(style="darkgrid")
plt.figure(figsize = (14,7))
plt.xticks(rotation = 60)
ax=sns.barplot(x = top_10_categories_installs.Category, y = top_10_categories_installs.C
plt.xlabel("Categories")
plt.ylabel("Number of Applications")
plt.title("Top 10 Categories")
for i in ax.containers:
    ax.bar_label(i)
plt.show()
```

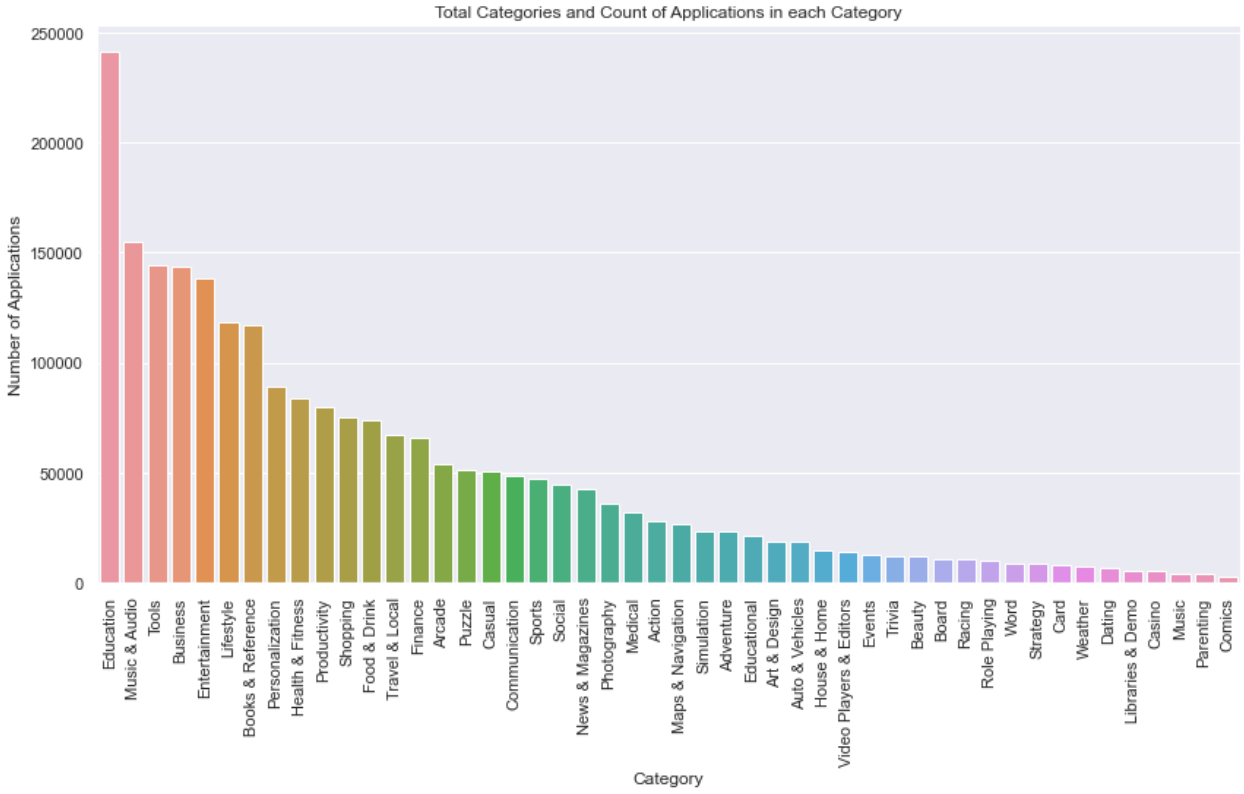


- From the above two plots, we can see that, maximum number of Apps present in google play store comes under Education, Music & Audio, Tools, Business and Entertainment etc. categories but for the installation scenario it is different. Maximum installed Apps comes under Tools, Productivity, Entertainment, Music & Audio etc.

Visualization of Total Categories and the Count Apps in each category.

In [73]:
```
sns.set_theme(style="darkgrid")
plt.figure(figsize = (14,7))
plt.xticks(rotation = 90)
sns.barplot(x = top_category_installs.Category, y = top_category_installs.Count)
plt.xlabel("Category")
plt.ylabel("Number of Applications")
plt.title("Total Categories and Count of Applications in each Category")
```

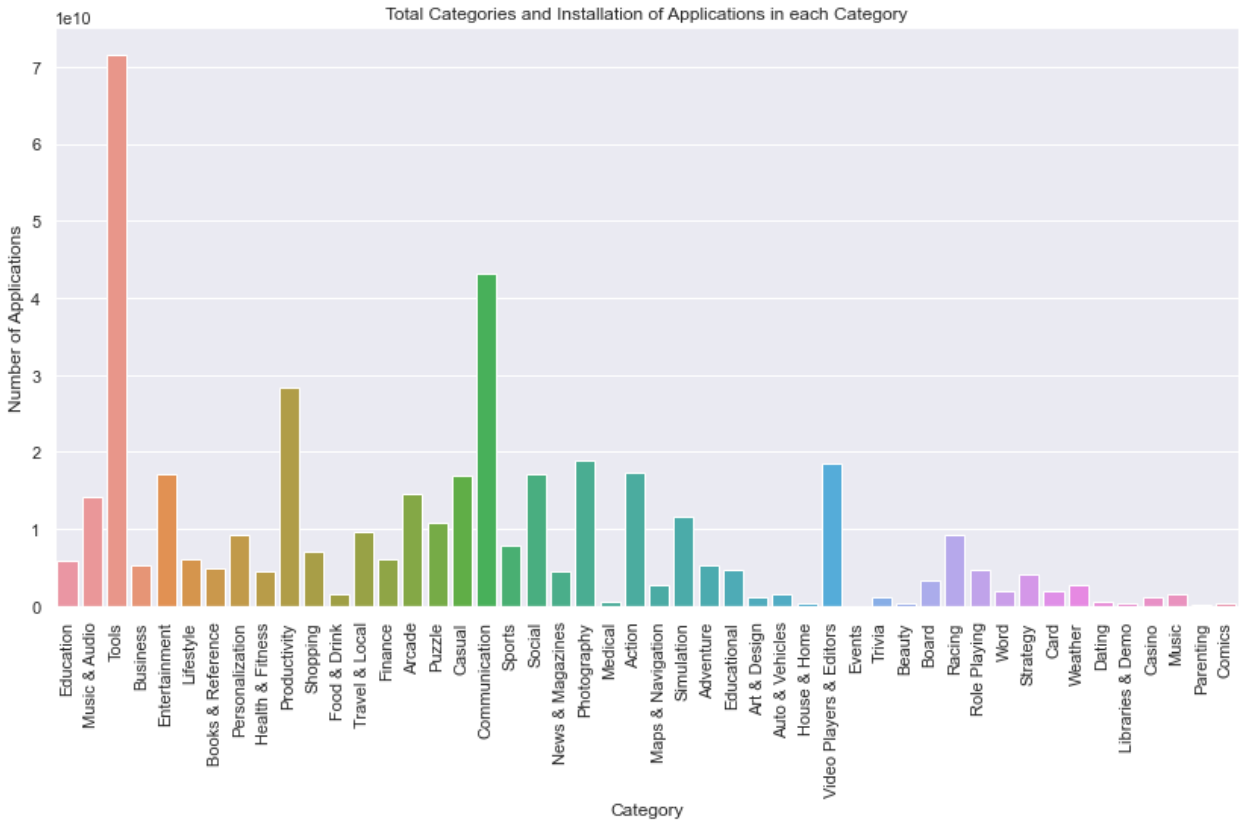
Out[73]:
```
Text(0.5, 1.0, 'Total Categories and Count of Applications in each Category')
```



Visualization of Total Categories and Installed Application in each category

In [74]:
```
plt.figure(figsize = (14,7))
plt.xticks(rotation = 90)
sns.barplot(x = top_category_installs.Category, y = top_category_installs.Installs)
plt.xlabel("Category")
plt.ylabel("Number of Applications")
plt.title("Total Categories and Installation of Applications in each Category")
```

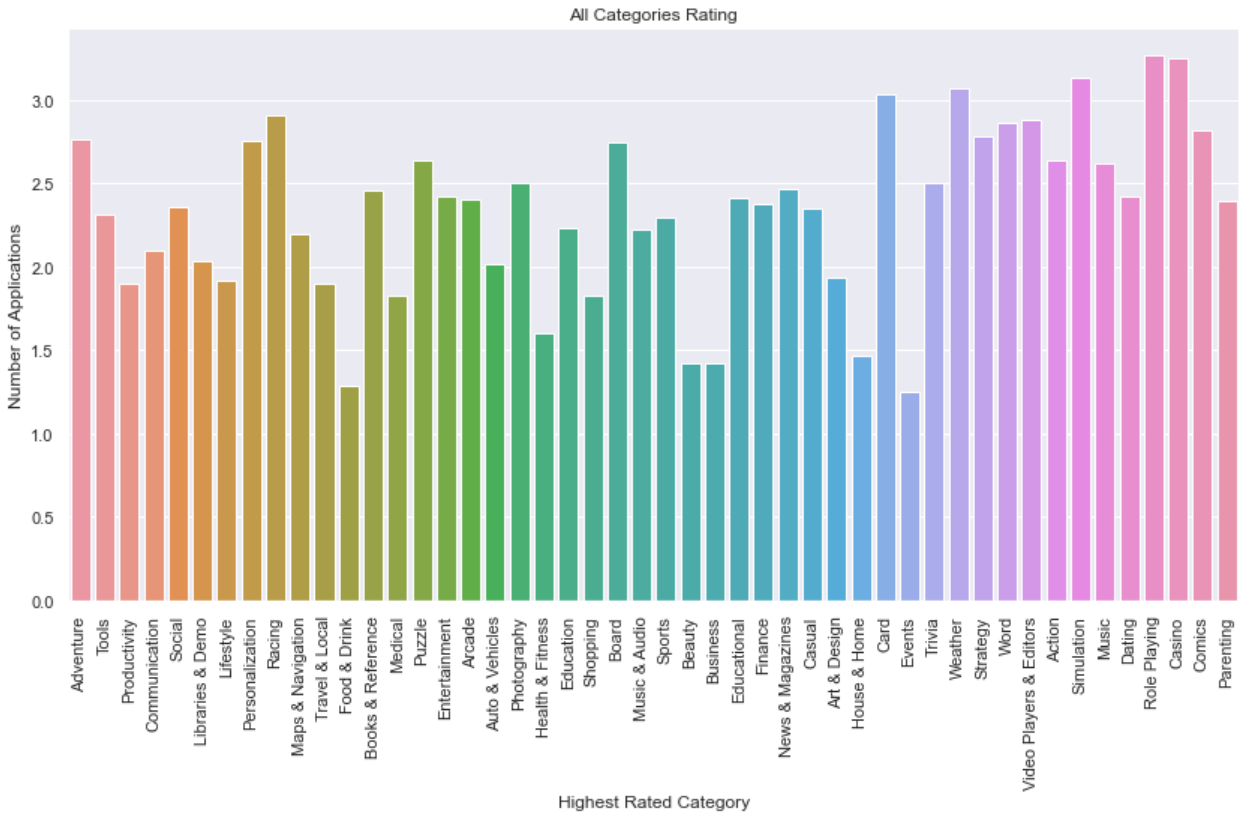
Out[74]:
```
Text(0.5, 1.0, 'Total Categories and Installation of Applications in each Category')
```



3. The highest rated Category

In [75]:
```
plt.figure(figsize = (14,7))
plt.xticks(rotation = 90)
sns.barplot(x = df.Category, y = df.Rating, ci=False)
plt.xlabel("Highest Rated Category")
plt.ylabel("Number of Applications")
plt.title("All Categories Rating")
```

Out[75]:
```
Text(0.5, 1.0, 'All Categories Rating')
```



- From the above plot we can see that 'Role Playing' is the highest Rated category.

4. Which Category has the highest Paid and Free Apps ?

In [76]:
```
app_count = df.groupby(["Category","Type"])[["App Name"]].count().reset_index().rename(c
app_count.head()
```

Out[76]:

	Category	Type	Count
0	Action	Free	26953
1	Action	Paid	586
2	Adventure	Free	22211
3	Adventure	Paid	982
4	Arcade	Free	53016

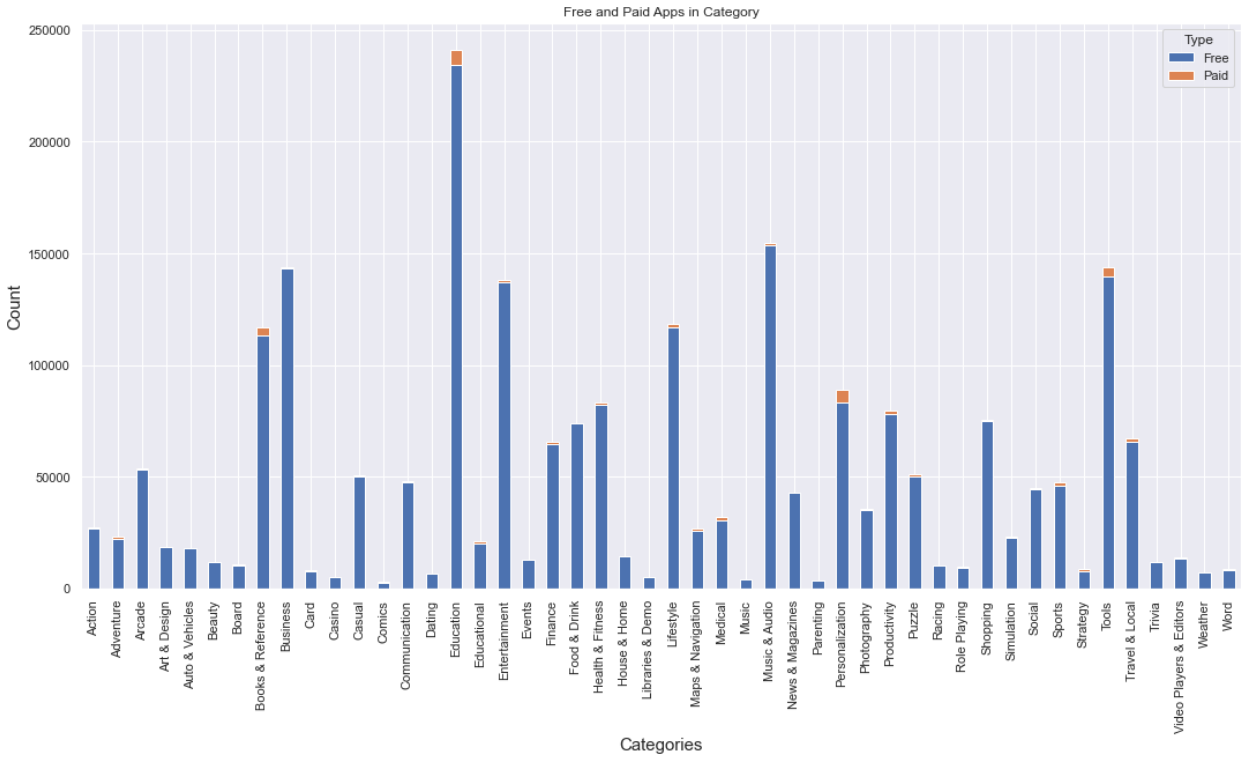
In [77]:
```
app_count_df = app_count.pivot("Category", "Type", "Count").reset_index()
app_count_df.head()
```

Out[77]:

Type	Category	Free	Paid
0	Action	26953	586
1	Adventure	22211	982
2	Arcade	53016	763
3	Art & Design	18364	172
4	Auto & Vehicles	17994	282

In [78]:
```
app_count_df.set_index('Category').plot(kind='bar', stacked=True, figsize=(18,9))
plt.xlabel("Categories", fontsize=15)
plt.ylabel("Count", fontsize=15)
plt.title("Free and Paid Apps in Category")
```

Out[78]:
```
Text(0.5, 1.0, 'Free and Paid Apps in Category')
```



- From the above plot, we can see that, 'Education' Category has the highest in paid and free apps.
- It looks like certain app categories have more free apps available for download than others. In our dataset, the majority of apps in Business, Education, Music & Audio, Tools as well as Entertainment categories were free to install. At the same time Books & References, Education, Tools and Personalization categories has the highest paid apps available for Installation.

5. Visualizing the Installation types in each category

In [79]:
```
df["Installs"].min(), df["Installs"].max()
```

Out[79]:
```
(0, 10000000000)
```

- There is a high variance in the number of installs, so we need to reduce it. To do that, we can use a log value for this column, otherwise it would be difficult to see the data when we visualize.

In [80]:
```
category_type_installs = df.groupby(['Category','Type'])[['Installs']].sum().reset_index
category_type_installs.head()
```

Out[80]:

	Category	Type	Installs
0	Action	Free	17344854020
1	Action	Paid	55001308
2	Adventure	Free	5368844127
3	Adventure	Paid	21264729
4	Arcade	Free	14470221258

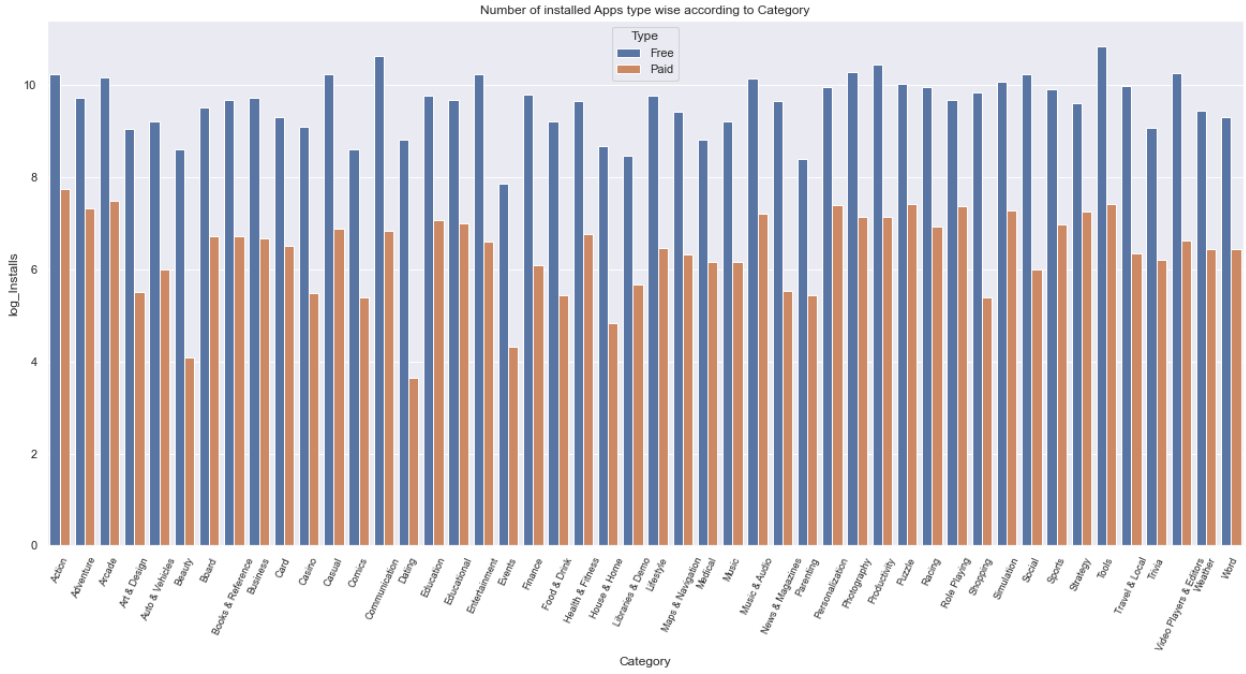
In [81]:
```
category_type_installs['log_Installs'] = np.log10(category_type_installs['Installs'])
category_type_installs.head()
```

Out[81]:

	Category	Type	Installs	log_Installs
0	Action	Free	17344854020	10.239171
1	Action	Paid	55001308	7.740373
2	Adventure	Free	5368844127	9.729881
3	Adventure	Paid	21264729	7.327660
4	Arcade	Free	14470221258	10.160475

In [82]:
```
plt.figure(figsize=(20,9))
plt.xticks(rotation=65,fontsize=9)
plt.xlabel("Category")
plt.ylabel("Installs")
plt.title("Number of installed Apps type wise according to Category")
sns.barplot(x = 'Category', y = 'log_Installs', hue='Type', data=category_type_installs)
```

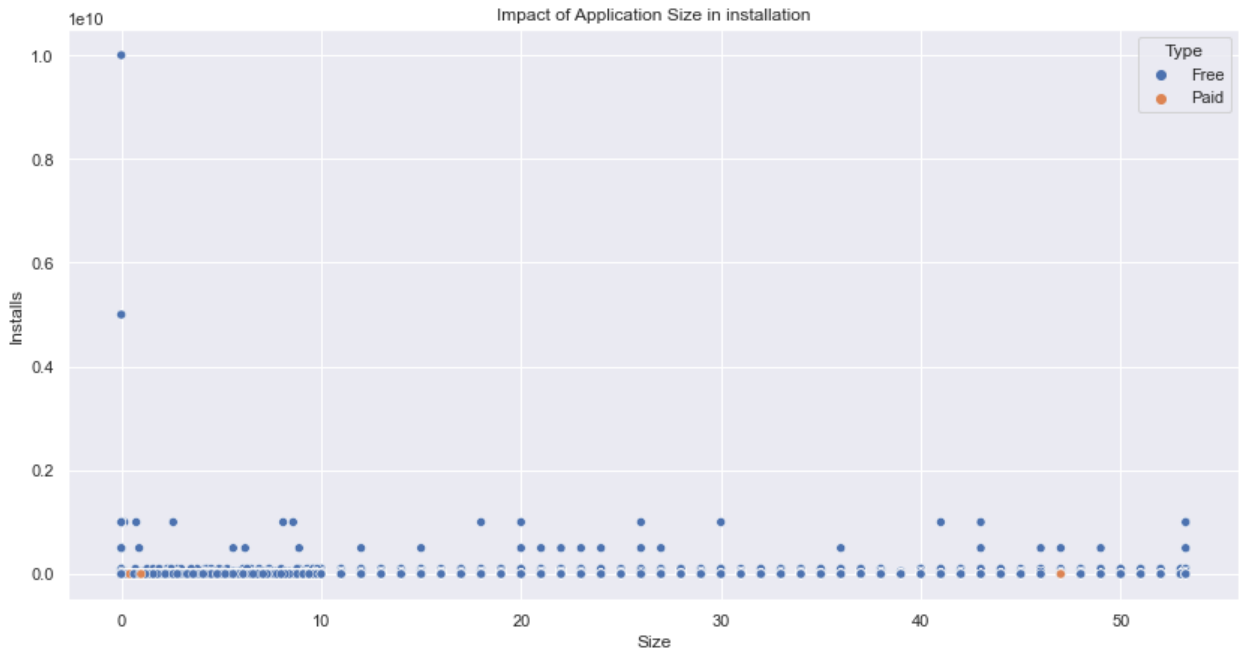
Out[82]:
```
<AxesSubplot:title={'center':'Number of installed Apps type wise according to Categor
y'}, xlabel='Category', ylabel='log_Installs'>
```



6. Impact of Installation with the size of the Application:

In [83]:
```
plt.figure(figsize = (14,7))
plt.xlabel("Size")
plt.ylabel("Installs")
plt.title("Impact of Application Size in installation")
sns.scatterplot(x = "Size", y = "Installs", hue = "Type", data = df)
```

Out[83]:
```
<AxesSubplot:title={'center':'Impact of Application Size in installation'}, xlabel='Size', ylabel='Installs'>
```



- From the above plot we can see that size impacts the number of installations. Application with large size are less installed by the users.

7. Top 5 Paid Apps based with highest Ratings and Installs:

In [84]:
```
df["Installs"].max()
```

Out[84]:
```
10000000000
```

In [85]:
```
paid_apps = df[(df.Type=="Paid") & (df.Installs >= 5000000)]
paid_apps
```

Out[85]:

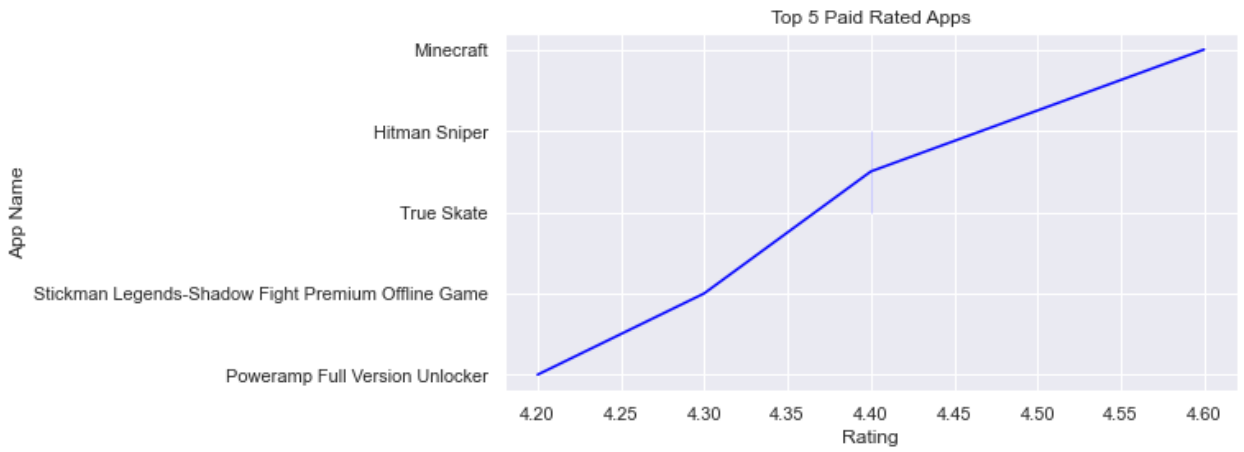
	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs
6302	True Skate	com.trueaxis.trueskate	Sports	4.4	102.5	5000000	5000000
508696	Hitman Sniper	com.squareenixmontreal.hitmansniperandroid	Action	4.4	102.5	10000000	10000000
837351	Minecraft	com.mojang.minecraftpe	Arcade	4.6	102.5	10000000	10000000
1859983	Stickman Legends- Shadow Fight Premium Offline ...	com.zitga.ninja.stickman.legends.shadow.wars	Action	4.3	102.5	10000000	10000000
1933739	Poweramp Full Version Unlocker	com.maxmpz.audioplayer.unlock	Music & Audio	4.2	102.5	5000000	5000000
2052997	League of Stickman 2020- Ninja Arena PVP(Dream...	me.dreamsky.stickman	Action	4.1	102.5	5000000	5000000

6 rows × 22 columns

In [86]:
```
paid_apps = paid_apps.groupby("App Name")["Rating"].max().sort_values(ascending = False)
paid_apps = paid_apps.head(5)
```

In [87]:
```
plt.figure(figsize = (8,4))
plt.xlabel("Rating")
sns.set_theme(style = "whitegrid")
plt.title("Top 5 Paid Rated Apps")
sns.lineplot(x = paid_apps.values, y = paid_apps.index, color = "Blue")
```

Out[87]:
```
<AxesSubplot:title={'center':'Top 5 Paid Rated Apps'}, xlabel='Rating', ylabel='App Nam
e'>
```



- From the above plot, we can see that the top 5 Paid apps based on highest rating and Installs are from Minecraft, Hitman Sniper, True Skate, Stickman Legends-Shadow Fight Premium Offline Game and Poweramp Full Version Unlocker apps.

8. Top 5 Free Apps based with highest ratings and Installs:

In [88]:
```
free_apps = df[(df.Type=="Free") & (df.Installs >= 5000000)]
free_apps.head()
```

Out[88]:

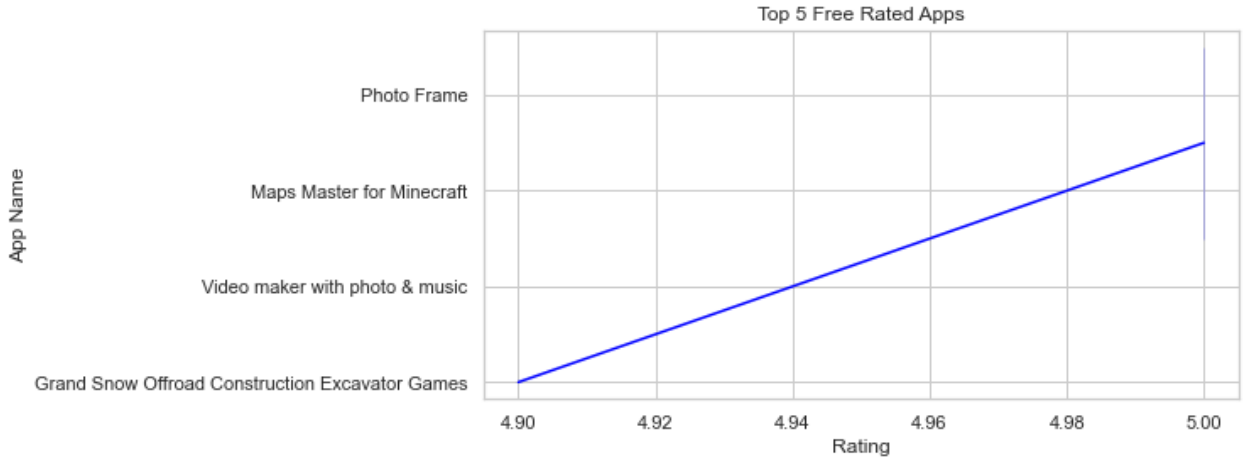
	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs	I
92	Car Games Revival: Car Racing Games for Kids	com.lf.real.extreme.suv.offroad.drive.games.free	Racing	3.9	102.5	10000000	10000000	3
291	Taxsee Driver	com.taxsee.driver	Maps & Navigation	4.4	102.5	5000000	5000000	
472	MONOMAX บริการดูหนังออนไลน์	com.doonung.activity	Entertainment	3.7	102.5	5000000	5000000	
561	Web Browser & Fast Explorer	fast.explorer.web.browser	Social	4.4	102.5	10000000	10000000	1
631	Piano Melody	com.veitch.themelodymaster.pmf	Education	4.2	102.5	10000000	10000000	1

5 rows × 22 columns

In [89]:
```
free_apps = free_apps.groupby("App Name")["Rating"].max().sort_values(ascending = False)
free_apps = free_apps.head(5)
```

In [90]:
```
plt.figure(figsize = (8,4))
plt.xlabel("Rating")
sns.set_theme(style = "whitegrid")
plt.title("Top 5 Free Rated Apps")
sns.lineplot(x = free_apps.values, y = free_apps.index, color = "Blue")
```

Out[90]:
```
<AxesSubplot:title={'center':'Top 5 Free Rated Apps'}, xlabel='Rating', ylabel='App Nam
e'>
```

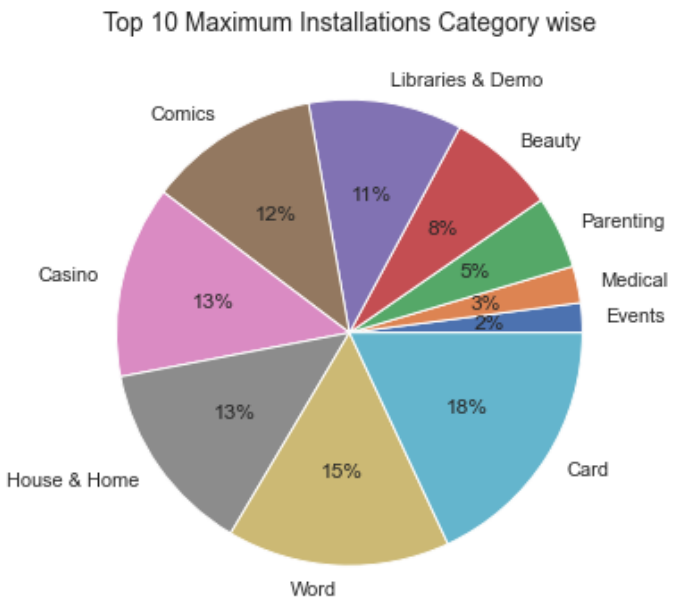


- From the above plot, we can see that top 5 free rated apps are Photo Frame, Maps Master for Minecraft, Video maker with photo & music, Grand Snow Offroad Construction Excavator Games.

8. Categories that have the Top 10 Maximum Installations:

In [91]:
```
plt.figure(figsize = (8,6))
data = df.groupby(["Category"])["Maximum Installs"].max().sort_values(ascending = True)
data = data.head(10)
labels = data.keys()
plt.title("Top 10 Maximum Installations Category wise", fontsize = 14)
plt.pie(data, labels = labels, autopct = "%.0f%%")
```

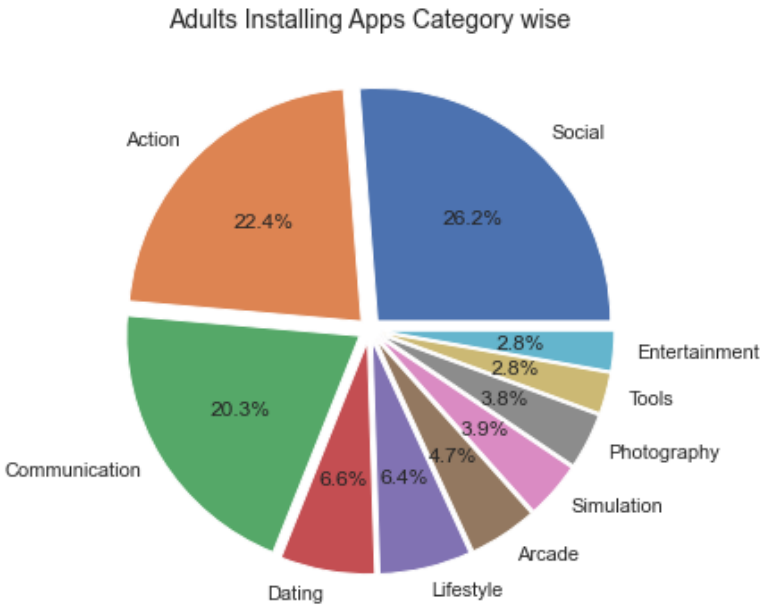
Out[91]:
```
([<matplotlib.patches.Wedge at 0x1d44e350340>,
  <matplotlib.patches.Wedge at 0x1d44e350ac0>,
  <matplotlib.patches.Wedge at 0x1d44e35e220>,
  <matplotlib.patches.Wedge at 0x1d44e35e940>,
  <matplotlib.patches.Wedge at 0x1d44e36c0a0>,
  <matplotlib.patches.Wedge at 0x1d44e36c7c0>,
  <matplotlib.patches.Wedge at 0x1d44e36cee0>,
  <matplotlib.patches.Wedge at 0x1d44e378640>,
  <matplotlib.patches.Wedge at 0x1d44e378d60>,
  <matplotlib.patches.Wedge at 0x1d44e3864c0>],
 [Text(1.0977172374940003, 0.07082984193502413, 'Events'),
  Text(1.0760887493637732, 0.22810743848614537, 'Medical'),
  Text(0.9918973390366911, 0.4755414480483606, 'Parenting'),
  Text(0.7330703143522412, 0.8201267671619471, 'Beauty'),
  Text(0.1733461741584647, 1.0862555426347078, 'Libraries & Demo'),
  Text(-0.5752224409509276, 0.93761353628585, 'Comics'),
  Text(-1.0714062402971944, 0.2491759784815357, 'Casino'),
  Text(-0.8997519509913849, -0.6328083648998303, 'House & Home'),
  Text(-0.053943762140281536, -1.0986765085893813, 'Word'),
  Text(0.9278470196960145, -0.590846772049424, 'Card')],
 [Text(0.5987548568149093, 0.03863445923728588, '2%'),
  Text(0.5869574996529672, 0.1244222391742611, '3%'),
  Text(0.5410349122018314, 0.25938624439001484, '5%'),
  Text(0.39985653510122243, 0.4473418729974256, '8%'),
  Text(0.09455245863188982, 0.592503023255295, '11%'),
  Text(-0.3137576950641423, 0.5114255652468271, '12%'),
  Text(-0.5844034037984696, 0.13591417008083764, '13%'),
  Text(-0.4907737914498463, -0.345168199036271, '13%'),
  Text(-0.02942387025833538, -0.599278095594208, '15%'),
  Text(0.5060983743796442, -0.32228005748150396, '18%')])
```



9. Top 10 installation categories that Adults have installed the most:

In [92]:
```
plt.figure(figsize = (8,6))
Adult = df[(df["Content Rating"] == "Adults")]
Adult = Adult.groupby(["Category"])["Maximum Installs"].max().sort_values(ascending = Fa
Adult = Adult.head(10)
labels = Adult.keys()
plt.title("Adults Installing Apps Category wise", fontsize = 14)
myexplode = [0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05]
plt.pie(Adult, labels = labels, autopct = "%.1f%%", explode = myexplode)
```

Out[92]:
```
([<matplotlib.patches.Wedge at 0x1d44e3cc820>,
  <matplotlib.patches.Wedge at 0x1d44e3ccf40>,
  <matplotlib.patches.Wedge at 0x1d44e3dc6a0>,
  <matplotlib.patches.Wedge at 0x1d44e3dcdc0>,
  <matplotlib.patches.Wedge at 0x1d44e3e8520>,
  <matplotlib.patches.Wedge at 0x1d44e3e8c40>,
  <matplotlib.patches.Wedge at 0x1d44e3f53a0>,
  <matplotlib.patches.Wedge at 0x1d44e3f5ac0>,
  <matplotlib.patches.Wedge at 0x1d44e403220>,
  <matplotlib.patches.Wedge at 0x1d44e403940>],
 [Text(0.7810708955082571, 0.8440546523714738, 'Social'),
  Text(-0.8095312771749034, 0.8167980847648761, 'Action'),
  Text(-0.9793125021117178, -0.6028656759243198, 'Communication'),
  Text(-0.19969844104111364, -1.132528380503442, 'Dating'),
  Text(0.2636808226769595, -1.119362507748228, 'Lifestyle'),
  Text(0.6303702078781013, -0.9618385524711096, 'Arcade'),
  Text(0.866451400627151, -0.7561494365211472, 'Simulation'),
  Text(1.0237700836504646, -0.5238270858043912, 'Photography'),
  Text(1.1102703176214743, -0.2996661839592693, 'Tools'),
  Text(1.1456044660766762, -0.1004510194332223, 'Entertainment')],
 [Text(0.44147485398292785, 0.47707436873170256, '26.2%'),
  Text(-0.4575611566640758, 0.46166848269319083, '22.4%'),
  Text(-0.5535244577153187, -0.3407501646528764, '20.3%'),
  Text(-0.11287303189280336, -0.6401247368062933, '6.6%'),
  Text(0.14903698673045535, -0.6326831565533463, '6.4%'),
  Text(0.3562962044528399, -0.5436478774836706, '4.7%'),
  Text(0.4897334003544766, -0.42738881194673534, '3.9%'),
  Text(0.5786526559763495, -0.29607617893291677, '3.8%'),
  Text(0.6275440925686594, -0.169376538759587, '2.8%'),
  Text(0.6475155677824691, -0.056776663157908254, '2.8%')])
```

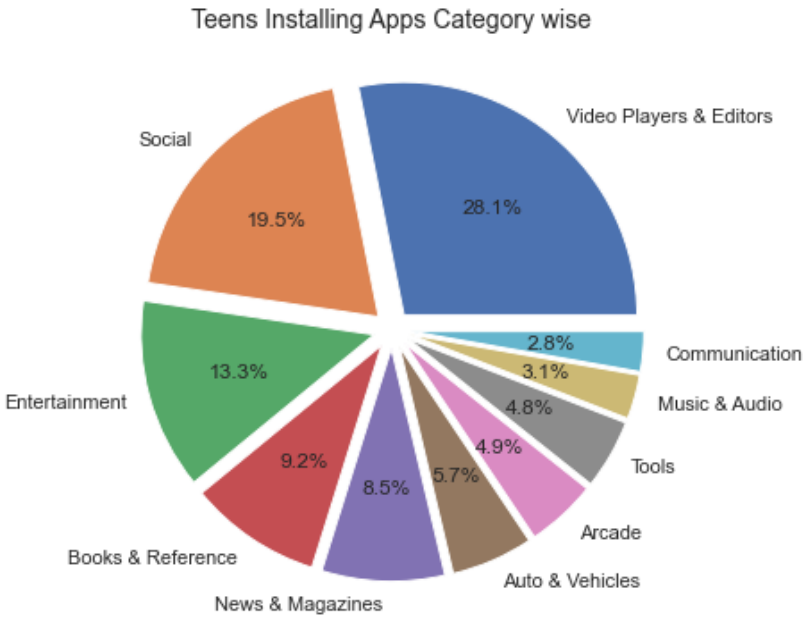


- Most of the Adults showing interest in downloading the Social, Action and Communication categories.

10. Visualization of Teens Installing the Apps in terms of category:

In [93]:
```
plt.figure(figsize = (8,6))
Teen = df[(df["Content Rating"] == "Teen")]
Teen = Teen.groupby(["Category"])["Maximum Installs"].max().sort_values(ascending = Fals
Teen = Teen.head(10)
labels = Teen.keys()
plt.title("Teens Installing Apps Category wise", fontsize = 14)
myexplode = [0.08,0.08,0.08,0.08,0.08,0.08,0.08,0.08,0.08,0.08]
plt.pie(Teen, labels = labels, autopct = "%.1f%%", explode = myexplode)
```

Out[93]:
```
([<matplotlib.patches.Wedge at 0x1d44ea8d850>,
  <matplotlib.patches.Wedge at 0x1d44ea8df70>,
  <matplotlib.patches.Wedge at 0x1d44ea996d0>,
  <matplotlib.patches.Wedge at 0x1d44ea99df0>,
  <matplotlib.patches.Wedge at 0x1d44eaa7550>,
  <matplotlib.patches.Wedge at 0x1d44eaa7c70>,
  <matplotlib.patches.Wedge at 0x1d44ef343d0>,
  <matplotlib.patches.Wedge at 0x1d44ef34af0>,
  <matplotlib.patches.Wedge at 0x1d44ef42250>,
  <matplotlib.patches.Wedge at 0x1d44ef42970>],
 [Text(0.7483652542006793, 0.91233187289777, 'Video Players & Editors'),
  Text(-0.8551751619194701, 0.8130654601174546, 'Social'),
  Text(-1.1371139093607474, -0.3152331789934528, 'Entertainment'),
  Text(-0.6601746973164577, -0.9780436437210375, 'Books & Reference'),
  Text(-0.04232812586343994, -1.179240573318646, 'News & Magazines'),
  Text(0.47147844677583406, -1.0817153388141665, 'Auto & Vehicles'),
  Text(0.7992816962802519, -0.8680718691395105, 'Arcade'),
  Text(1.0239705033439646, -0.5864165825430826, 'Tools'),
  Text(1.136864883180553, -0.3161300956740243, 'Music & Audio'),
  Text(1.1755048123189402, -0.10290012738093687, 'Communication')],
 [Text(0.43126133292920504, 0.5257505708224436, '28.1%'),
  Text(-0.49281280517393183, 0.4685461973558213, '19.5%'),
  Text(-0.6552859816655153, -0.18165979806402366, '13.3%'),
  Text(-0.3804396560806705, -0.5636183709578859, '9.2%'),
  Text(-0.02439247931113488, -0.6795623642853214, '8.5%'),
  Text(0.2716994439047179, -0.6233613816895197, '5.7%'),
  Text(0.46060301141573834, -0.5002448059448026, '4.9%'),
  Text(0.5900846968422846, -0.33793497977058995, '4.8%'),
  Text(0.6551424750532, -0.182176665303675, '3.1%'),
  Text(0.6774095528617621, -0.05929837849070938, '2.8%')])
```



- Most of the Teens showing interest in downloading the Video Players & Editors, Social and Entertainment category apps.

Correlation:

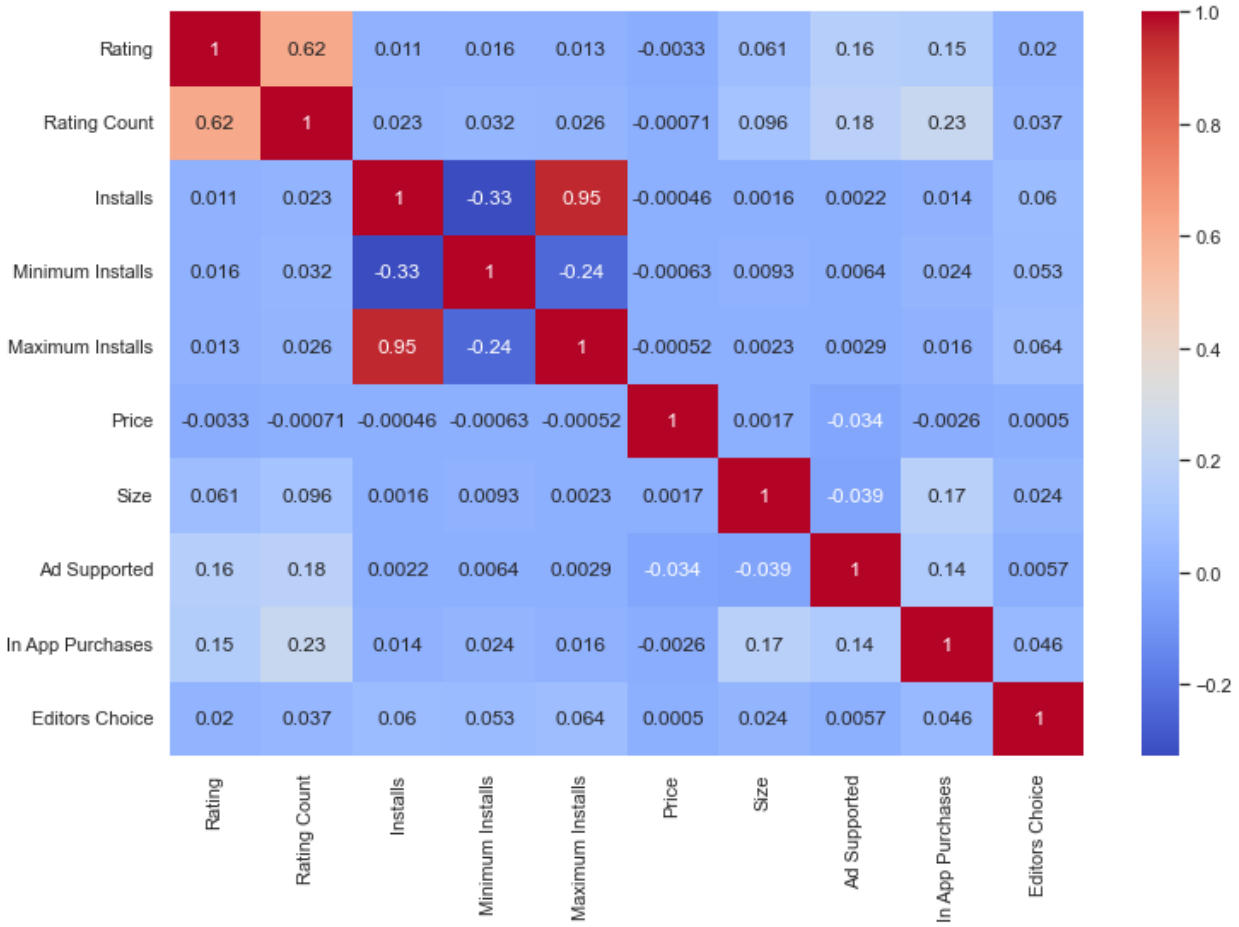
In [94]:
```
df.corr()
```

Out[94]:

	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Price	Size	Ad Supported	In App Purchases	
Rating	1.000000	0.615568	0.011268	0.015959	0.012671	-0.003314	0.060710	0.162928	0.150973	0
Rating Count	0.615568	1.000000	0.023195	0.032064	0.025997	-0.000711	0.096147	0.179890	0.233216	0
Installs	0.011268	0.023195	1.000000	-0.328609	0.954037	-0.000461	0.001647	0.002249	0.014178	0
Minimum Installs	0.015959	0.032064	-0.328609	1.000000	-0.240484	-0.000631	0.009348	0.006410	0.024430	0
Maximum Installs	0.012671	0.025997	0.954037	-0.240484	1.000000	-0.000515	0.002263	0.002881	0.016100	0
Price	-0.003314	-0.000711	-0.000461	-0.000631	-0.000515	1.000000	0.001694	-0.034281	-0.002636	0
Size	0.060710	0.096147	0.001647	0.009348	0.002263	0.001694	1.000000	-0.039437	0.172449	0
Ad Supported	0.162928	0.179890	0.002249	0.006410	0.002881	-0.034281	-0.039437	1.000000	0.138304	0
In App Purchases	0.150973	0.233216	0.014178	0.024430	0.016100	-0.002636	0.172449	0.138304	1.000000	0
Editors Choice	0.019578	0.037047	0.059932	0.052986	0.064206	0.000503	0.024383	0.005701	0.046078	1

In [95]:
```
plt.figure(figsize = (12,8))
sns.heatmap(df.corr(), annot = True, cbar = True, cmap = "coolwarm")
```

Out[95]: <AxesSubplot:>



- Factors like 'Ad Support' and 'In app Purchases' are correlated to app rating. So we can say that if the app provides customer support and have subscription plans we can engage more customers.
- We can also see from the same graph that the 'Editor's choice' plays an important role as well. With high editor choice we can see high ratings count and high installs.

Summary and Conclusion:

- People are more interested to install the gaming apps and the top rating is given to the gaming apps.
- In-Apps-Purchases are highly correlated to app rating. So, we can say that if the app provide customer support and have subscription plans it will helps to engage customers.
- Size of the Application varies the installation.
- People mostly downloaded the free Apps and their installation is high.So availability of Free Apps also very high.
- Most people do not give a rating, but the people who gave the rating, tend to give 4+ rating mostly.
- Most of the Adults installed the Social, Action and Communication Apps.
- Most of the app installation done by Teens are of Video Players and Editors, Social and Entertainment categories.

Recommendation:

Here are some ways I can recommend to do more analysis and visualization on this dataset:

1. Analysis of each Category and find which App gives the top rating in each category.
2. Explore the Minimum Android, Last updated and Released columns and analyze these columns with the rating.
3. Explore the developer email and developer website columns by comparing the developer's email provided. We can find some insights from the ones without the developer email related to Rating and customer service.