Here I am using the Google Playstore dataset, which contains details about the Apps in playstore, there are more than 10,0000+ Apps in the playstore.The size of the dataset is 210Mb.

The main objective of this project is to deliver insights to understand customer demands better and thus help developers to popularize the product.

In [1]:
```
# import neccessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

DATA PREPARATION & CLEANING

In [2]:
```
# download dataset
df = pd.read_csv('Google-Playstore.csv')
df
```

Out[2]:

	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	...	h
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	...	http:/
4	GROW.me	com.horodyski.grower	Tools	0.0	0.0	100+	100.0	478	True	0.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
2312939	大俠客—熱血歸來	com.rxsj.ssjj	Role Playing	4.3	16775.0	100,000+	100000.0	337109	True	0.0	...	
2312940	ORU Online	com.threedream.oruonline	Education	0.0	0.0	100+	100.0	430	True	0.0	...	
2312941	Data Structure	datastructure.appoworld.datastucture	Education	0.0	0.0	100+	100.0	202	True	0.0	...	
2312942	Devi Suktam	ishan.devi.suktam	Music & Audio	3.5	8.0	1,000+	1000.0	2635	True	0.0	...	https:/
2312943	Biliyor Musun - Sonsuz Yarış	com.yyazilim.biliyormusun	Trivia	5.0	12.0	100+	100.0	354	True	0.0	...	

2312944 rows × 24 columns

In [3]:
```
# see column names
df.columns
```

Out[3]:
```
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')
```

In [4]:
```
# check shape of data
df.shape
```

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

In [5]:
```
# see info about data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2312944 entries, 0 to 2312943
Data columns (total 24 columns):
 #   Column             Dtype
---  ------             -----
 0   App Name           object
 1   App Id             object
 2   Category           object
 3   Rating             float64
 4   Rating Count       float64
 5   Installs           object
 6   Minimum Installs   float64
 7   Maximum Installs   int64
 8   Free               bool
 9   Price              float64
 10  Currency           object
 11  Size               object
 12  Minimum Android    object
 13  Developer Id       object
 14  Developer Website  object
 15  Developer Email    object
 16  Released           object
 17  Last Updated       object
 18  Content Rating     object
 19  Privacy Policy     object
 20  Ad Supported       bool
 21  In App Purchases   bool
 22  Editors Choice     bool
 23  Scraped Time       object
dtypes: bool(4), float64(4), int64(1), object(15)
memory usage: 361.8+ MB
```

Observation:

- The dataset having 2312944 rows and 24 columns
- The columns in the dataset are:
  - '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'
- There are 5 numerical columns in the dataset

In [6]:
```
# descriptive statistics
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

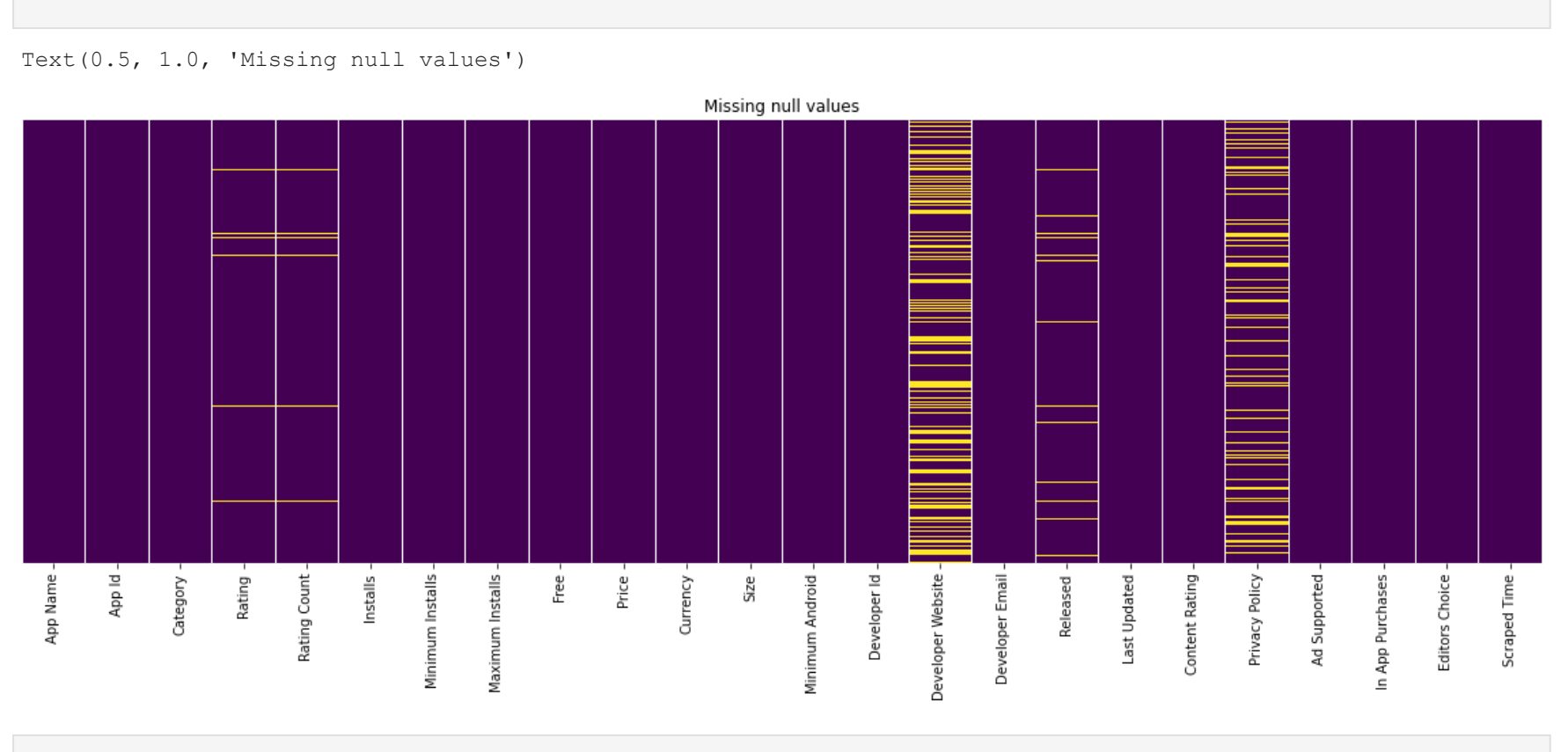
In [7]:
```
# check for missing values
df.isnull().sum().sort_values(ascending = False)
```

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

In [8]:
```
# plot missing null values

import matplotlib
matplotlib.rcParams['figure.figsize'] = (20,6)
sns.heatmap(df.isnull(),yticklabels = False, cbar = False , cmap = 'viridis')
plt.title("Missing null values")
```

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

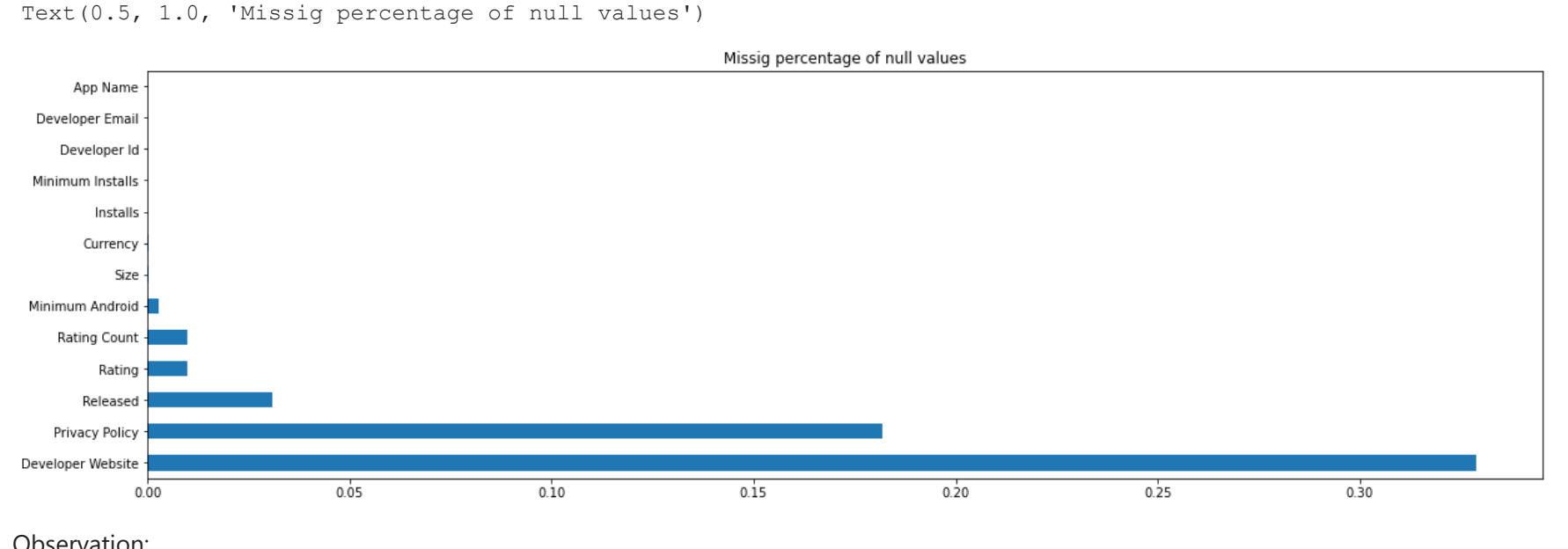


In [9]:
```
# check missing percentage
missing_percentage = df.isnull().sum().sort_values(ascending = False)/len(df)
missing_percentage
```

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

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



Observation:

- The columns with highest percentage of null values are :
  - Developer Website
  - Privacy Policy
- We can drop the small null values for the columns
  - Size
  - Currency
  - Installs
  - Minimum Installs
  - Developer Id
  - Developer Email
- Rating,Rating Count,Released are important column so it is good to fill the null values.
  - Rating
  - Rating count
  - Minimum Andriod
  - Released

In [11]:
```
# Drop small null values
df.dropna(subset = ['App Name','Size','Currency','Installs','Minimum Installs','Developer Id','Developer Email'
df.isnull().sum()
```

Out[11]:
```
App Name                 0
App Id                   0
Category                 0
Rating               22572
Rating Count         22572
Installs                 0
Minimum Installs         0
Maximum Installs         0
Free                     0
Price                    0
Currency                 0
Size                     0
Minimum Android       6530
Developer Id             0
Developer Website   760716
Developer Email          0
Released             70748
Last Updated             0
Content Rating           0
Privacy Policy      420845
Ad Supported             0
In App Purchases         0
Editors Choice           0
Scraped Time             0
dtype: int64
```

In [12]:
```
# Check for duplicates
df['App Name'].duplicated().any()
```

Out[12]:
```
True
```

In [13]:
```
df['App Name'].value_counts()
```

Out[13]:
```
Tic Tac Toe                       382
Calculator                        261
Flashlight                        256
BMI Calculator                    201
Age Calculator                    190
                                 ...
Color Bowl Puzzle                   1
[illegible]                         1
Quest Adventurer                    1
BPM LOCI                            1
Quản lý Trực Tuyến - Công chúc      1
Name: App Name, Length: 2177593, dtype: int64
```

In [14]:
```
# check to see if data is duplicated
df[df['App Name'] == "Tic Tac Toe"]
```

Out[14]:

	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	...	
3918	Tic Tac Toe	com.dev.atte.tic_tac_toe	Puzzle	0.0	0.0	10+	10.0	30	True	0.0	...	https:/
13758	Tic Tac Toe	com.md.tictactoe	Board	0.0	0.0	10+	10.0	31	True	0.0	...	
25351	Tic Tac Toe	com.sthirumuru.tictactoenew	Board	4.2	11.0	5,000+	5000.0	6379	True	0.0	...	
28142	Tic Tac Toe	com.fsk00x.TicTacToe	Board	0.0	0.0	500+	500.0	922	True	0.0	...	
30962	Tic Tac Toe	net.snatchdreams.tictactoe	Puzzle	0.0	0.0	10+	10.0	38	True	0.0	...	h
...	...	...	...	...	...	...	...	...	...	...	...	...
2272273	Tic Tac Toe	com.tictactoe.MobileAppsPro	Board	3.8	9269.0	500,000+	500000.0	800662	True	0.0	...	
2281376	Tic Tac Toe	com.erjainam.tictactoe.release	Board	0.0	0.0	50+	50.0	50	True	0.0	...	
2282023	Tic Tac Toe	com.swapworld.tic_tac_toe	Board	4.7	7.0	10+	10.0	34	True	0.0	...	http://prudhvikchirun
2299121	Tic Tac Toe	com.infuse.tictactoe	Board	4.8	20.0	100+	100.0	462	True	0.0	...	
2310375	Tic Tac Toe	com.justmalloavi.avinashk.threet	Puzzle	4.7	10.0	100+	100.0	173	True	0.0	...	

382 rows × 24 columns

In [15]:
```
# check duplicates in app id
df['App Id'].duplicated().any()
```

Out[15]:
```
False
```

In [16]:
```
df['App Id'].value_counts()
```

Out[16]:
```
kr.co.skills.mmsstreaminghelper        1
com.muzikssizilahi.hzrmusic            1
com.aslulbayt.alshia                   1
com.Chintz.TheChintzBar                1
com.chownow.dialogcafe                 1
                                      ..
com.reno.free                          1
org.contentarcade.apps.sourecipes      1
de.rothsoftware.stadtbildpflege.kl     1
com.UnofficialFleetwoodmac.myapp       1
com.volkion.guessthecharacter          1
Name: App Id, Length: 2312548, dtype: int64
```

Observation:

- We have the Apps with the same names but with the unique App IDs so the Apps are differed based on the App IDs

In [17]:
```
# evaluate numerical columns
df['Installs'].unique()
```

Out[17]:
```
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)
```

In [18]:
```
# Change from objects to integers
df['Installs'] = df['Installs'].str.split('+').str[0]    # removes the + symbol
df['Installs'].replace(',','', regex=True, inplace=True) #replace the commas ',' in the install column
df['Installs'] = df['Installs'].astype('int64')          # converting it to the int type
```

In [19]:
```
df['Installs'].unique()
```

Out[19]:
```
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)
```

In [20]:
```
df['Currency'].unique()
```

Out[20]:
```
array(['USD', 'XXX', 'CAD', 'EUR', 'INR', 'VND', 'GBP', 'BRL', 'KRW',
       'TRY', 'RUB', 'SGD', 'AUD', 'PKR', 'ZAR'], dtype=object)
```

In [21]:
```
df['Size'].unique()
```

Out[21]:
```
array(['10M', '2.9M', '3.7M', ..., '405M', '3.2k', '512M'], dtype=object)
```

In [22]:
```
# convert data in mb from gb and kb
df['Size'] = df['Size'].apply(lambda x: str(x).replace('M', '') if 'M' in str(x) else x)
```

I got a mismatched value with the data, I got the value 1,018 which I can drop it or I can assume as it may be a '.' (dot) the would incorrectly added to the dataset. so let assume it as dot for now and repalce the ',' with the dot '.'

In [23]:
```
df['Size'] = df['Size'].apply(lambda x: str(x).replace(',', '.') if ',' in str(x) else x)
```

In [24]:
```
# convert kbs to mb
df['Size'] = df['Size'].apply(lambda x: float(str(x).replace('k', '')) / 1000 if 'k' in str(x) else x)
```

I got an another missmatch value in the data set which is 'Varies with device', it means the size of the App may vaies with the device---> but here I am assuming it as 0

In [25]:
```
df['Size'] = df['Size'].apply(lambda x: str(x).replace('Varies with device', '0') if 'Varies with device' in st
```

In [26]:
```
# convert gbs to mb
df['Size'] = df['Size'].apply(lambda x: float(str(x).replace('G', '')) * 1000 if 'G' in str(x) else x)
df['Size'] = df['Size'].apply(lambda x: float(x))
df.dtypes['Size']
```

Out[26]:
```
dtype('float64')
```

In [27]:
```
df['Minimum Android']
```

Out[27]:
```
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 [28]:
```
df['Content Rating']
```

Out[28]:
```
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 [29]:
```
df['Released']
```

Out[29]:
```
0          Feb 26, 2020
1          May 21, 2020
2           Aug 9, 2019
3          Sep 10, 2018
4          Feb 21, 2020
               ...
2312939             NaN
2312940    Jan 17, 2018
2312941    Aug 19, 2018
2312942     Aug 1, 2016
2312943     Aug 9, 2019
Name: Released, Length: 2312548, dtype: object
```

In [30]:
```
df['Last Updated']
```

Out[30]:
```
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 [31]:
```
df['Privacy Policy']
```

Out[31]:
```
0                         https://beniyizibyose.tk/projects/
1          https://dev4phones.wordpress.com/licencia-de-uso/
2          https://www.vietnamairlines.com/vn/en/terms-an...
3                                                        NaN
4                                http://www.horodyski.com.pl
                                 ...
2312939                http://a.4399sy.com.hk/user/aggreement
2312940      http://www.oru.edu/about-oru/privacy-policy.php
2312941    https://appoworld.000webhostapp.com/datastruct...
2312942    https://docs.google.com/document/d/1x-9reZuLRX...
2312943    https://biliyor-musun-sons.flycricket.io/priva...
Name: Privacy Policy, Length: 2312548, dtype: object
```

In [32]:
```
df['Scraped Time']
```

Out[32]:
```
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
```

In [33]:
```
df['Free']
```

Out[33]:
```
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
```

In [34]:
```
# Create column type for free and paid apps using free column
df['Type'] = np.where(df['Free'] == True,'Free','Paid')
df.drop(['Free'],axis=1, inplace= True )
```

In [35]:
```
# Clean up content column a little
df['Content Rating'].unique()
```

Out[35]:
```
array(['Everyone', 'Teen', 'Mature 17+', 'Everyone 10+',
       'Adults only 18+', 'Unrated'], dtype=object)
```

In [36]:
```
df["Content Rating"].value_counts()
```

Out[36]:
```
Everyone           2021788
Teen                196311
Mature 17+           60278
Everyone 10+         33881
Unrated                154
Adults only 18+        136
Name: Content Rating, dtype: int64
```

Observations:

- Categories in the content Rating column :
  - Everyone
  - Teen
  - Mature 17+
  - Everyone 10+
  - Unrated
  - Adults only 18+

I am going to simply these for clearer understanding.

In [37]:
```
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 [38]:
```
df['Content Rating'].unique()
```

Out[38]:
```
array(['Everyone', 'Teen', 'Adults'], dtype=object)
```

In [39]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2312548 entries, 0 to 2312943
Data columns (total 24 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   float64
 7   Maximum Installs   int64
 8   Price              float64
 9   Currency           object
 10  Size               float64
 11  Minimum Android    object
 12  Developer Id       object
 13  Developer Website  object
 14  Developer Email    object
 15  Released           object
 16  Last Updated       object
 17  Content Rating     object
 18  Privacy Policy     object
 19  Ad Supported       bool
 20  In App Purchases   bool
 21  Editors Choice     bool
 22  Scraped Time       object
 23  Type               object
dtypes: bool(3), float64(5), int64(2), object(14)
memory usage: 394.8+ MB
```

In [40]:
```
df['Rating'].unique()
```

Out[40]:
```
array([0. , 4.4, 5. , 4.5, 2. , 4.7, 4.9, 3.9, 3.7, 4.2, 3.4, 3.8, 4.6,
       4.1, 2.5, 2.3, 2.1, 2.7, 4.3, 3.1, 4.8, 3.3, 4. , 2.4, 3.2, 3.6,
       2.6, nan, 1.5, 3.5, 2.9, 1.3, 2.8, 3. , 2.2, 1.8, 1.4, 1.2, 1.9,
       1.7, 1.6, 1. , 1.1])
```

In [41]:
```
df['Rating Count'].unique()
```

Out[41]:
```
array([0.0000e+00, 6.4000e+01, 5.0000e+00, ..., 8.7553e+04, 7.5960e+04,
       7.8351e+04])
```

In [42]:
```
# Change rating count to certain ranges for clearer understanding
df['Rating Count'].max()
```

Out[42]:
```
138557570.0
```

In [43]:
```
df['Rating Type'] = 'NoRatingProvided'
df.loc[(df['Rating Count'] > 0) & (df['Rating Count'] <= 10000.0),'Rating Type'] = 'Less than 10K'
df.loc[(df['Rating Count'] > 10000) & (df['Rating Count'] <= 500000.0),'Rating Type'] = 'Between 10K and 500K'
df.loc[(df['Rating Count'] > 500000) & (df['Rating Count'] <= 138557570.0),'Rating Type'] = 'More than 500K'
df['Rating Type'].value_counts()
```

Out[43]:
```
Less than 10K           1192801
NoRatingProvided        1082303
Between 10K and 500K      35779
More than 500K             1665
Name: Rating Type, dtype: int64
```

In [44]:
```
df['Rating Type']
```

Out[44]:
```
0              NoRatingProvided
1                 Less than 10K
2              NoRatingProvided
3                 Less than 10K
4              NoRatingProvided
                   ...
2312939    Between 10K and 500K
2312940        NoRatingProvided
2312941        NoRatingProvided
2312942           Less than 10K
2312943           Less than 10K
Name: Rating Type, Length: 2312548, dtype: object
```

In [45]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2312548 entries, 0 to 2312943
Data columns (total 25 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   float64
 7   Maximum Installs   int64
 8   Price              float64
 9   Currency           object
 10  Size               float64
 11  Minimum Android    object
 12  Developer Id       object
 13  Developer Website  object
 14  Developer Email    object
 15  Released           object
 16  Last Updated       object
 17  Content Rating     object
 18  Privacy Policy     object
 19  Ad Supported       bool
 20  In App Purchases   bool
 21  Editors Choice     bool
 22  Scraped Time       object
 23  Type               object
 24  Rating Type        object
dtypes: bool(3), float64(5), int64(2), object(15)
memory usage: 412.4+ MB
```

QUESTIONS AND ANSWERS:

- What are the top 10 Categories that are installed from the Google Play Store ?
- Which are the Categories that are getting installed the most in top 10 Categories?
- What is the higest rated Category ?
- Which Category has the highest Paid and Free apps ?
- How does the size of the Application impacts the Installation?
- What are the Top 5 Paid Apps based With highest ratings and installs ?
- What are the Top 5 Free Apps based With highest ratings and installs ?

1. What are the top 10 Categories that are installed from the Google Play Store ?

In [46]:
```
df['Category'].unique()
```

Out[46]:
```
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 [47]:
```
top_Category = df.Category.value_counts().reset_index().rename(columns={'Category':'Count','index':'Category'})
top_Category
```

Out[47]:

	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
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 [48]:
```
Category_installs = df.groupby(['Category'])[['Installs']].sum()
Category_installs
```

Out[48]:

Category	Installs
Action	17399855328
Adventure	5390108856
Arcade	14501230855
Art & Design	1116398902
Auto & Vehicles	1594754418
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
Shopping	7108586169
Simulation	11689990564
Social	17161994565
Sports	7954121388
Strategy	4156174163
Tools	71440271217
Travel & Local	9726222850
Trivia	1188962483
Video Players & Editors	18591154109
Weather	2773800062
Word	2029143357

In [49]:
```
top_Category_installs = pd.merge(top_Category, top_Category_installs, on='Category')
top_Category_installs.head(5)
```

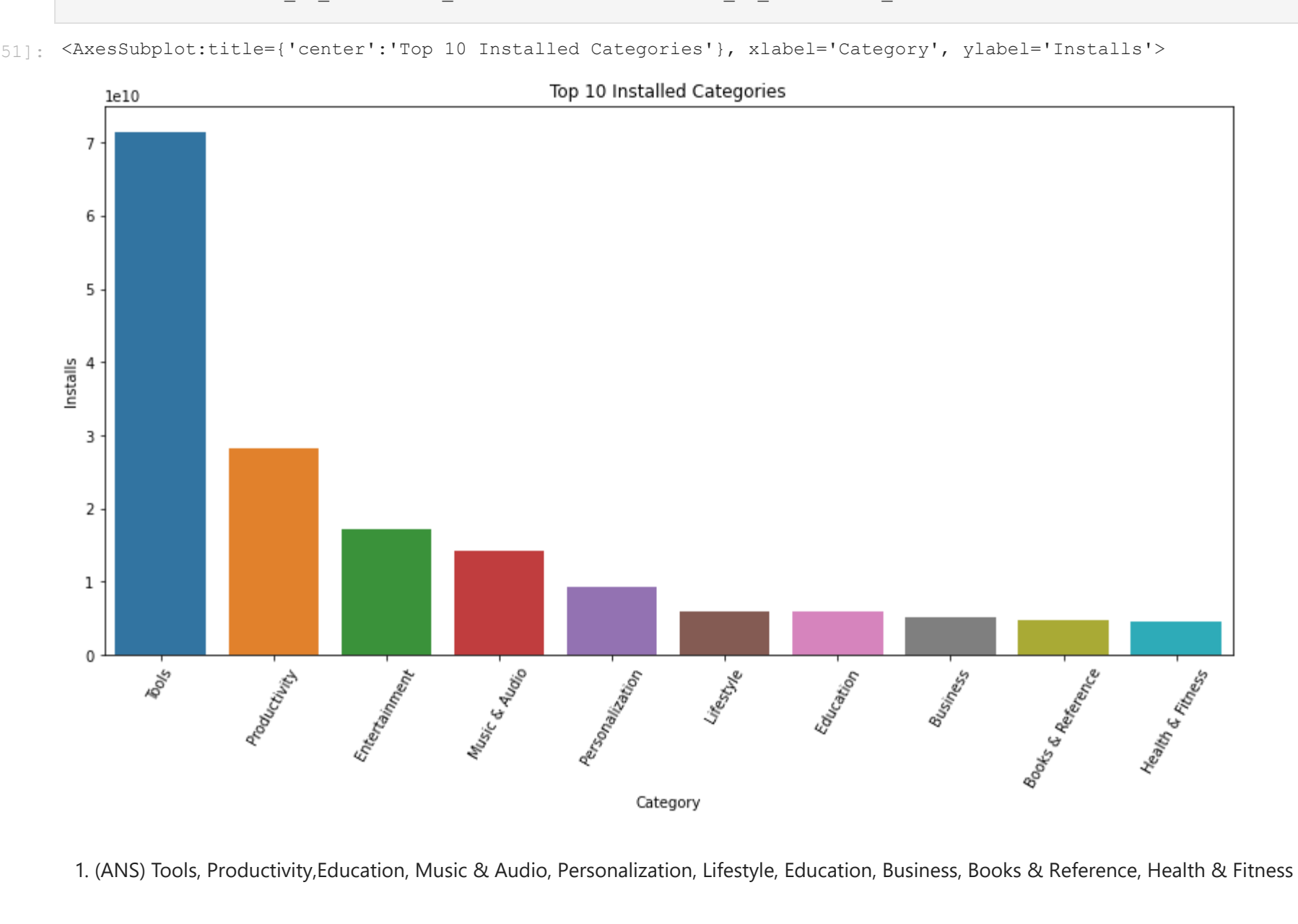
Out[49]:

	Category	Count	Installs
0	Education	241068	5983815847
1	Music & Audio	154897	14239401798
2	Tools	143971	71440271217
3	Business	143749	5236661902
4	Entertainment	138261	17108397933

In [50]:
```
top_10_Categories_installs = top_Category_installs.head(10).sort_values(by = ['Installs'],ascending = False)
```

In [51]:
```
plt.figure(figsize=(14,7))
plt.xticks(rotation=60)
plt.xlabel("Category")
plt.ylabel("Number of applications")
plt.title("Top 10 Installed Categories")
sns.barplot(x = top_10_Categories_installs.Category,y= top_10_Categories_installs.Installs)
```

Out[51]: <AxesSubplot:title={'center':'Top 10 Installed Categories'}, xlabel='Category', ylabel='Installs'>

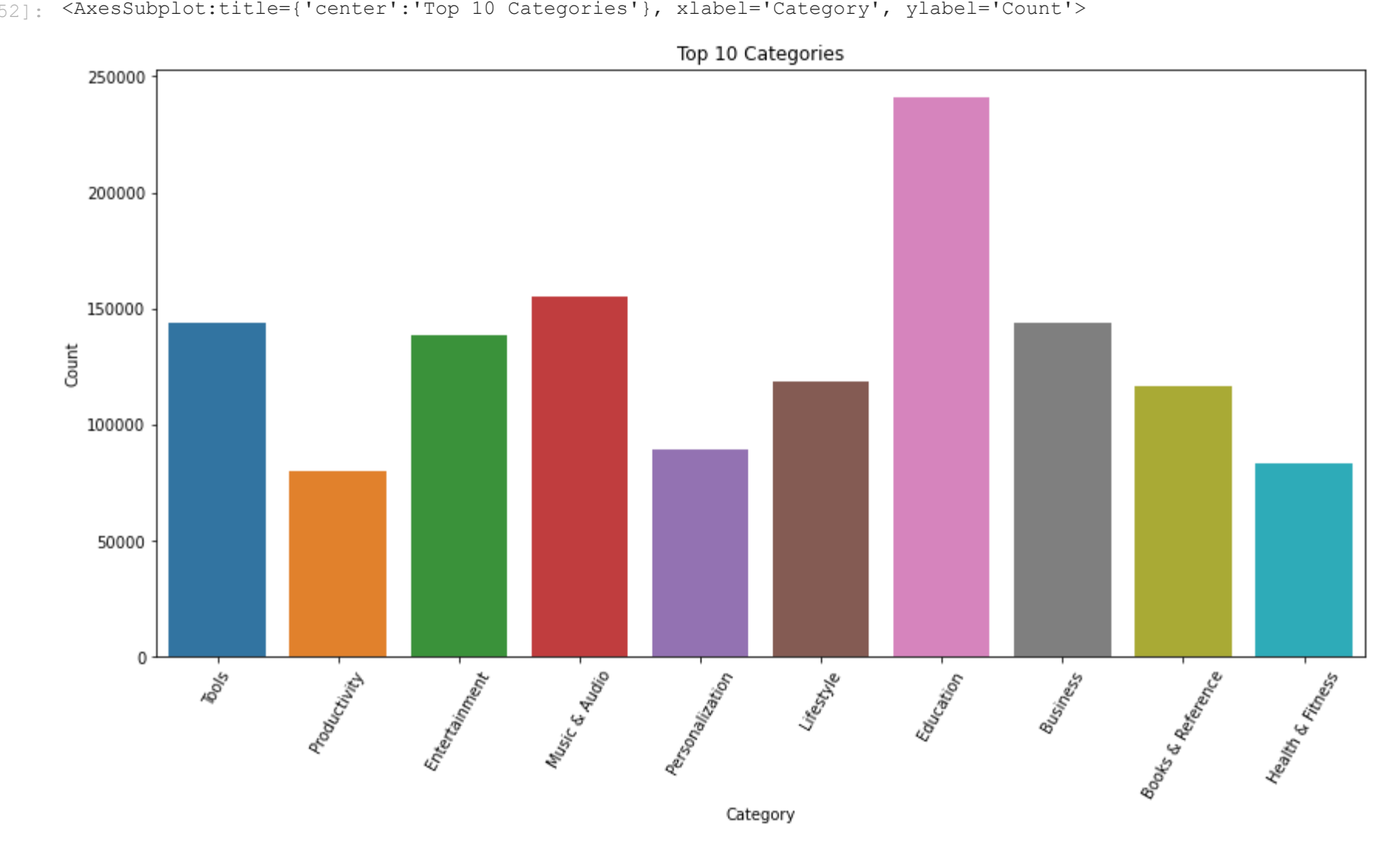


1. (ANS) Tools, Productivity,Education, Music & Audio, Personalization, Lifestyle, Education, Business, Books & Reference, Health & Fitness

1. What are the 10 Categories in playstore as per the count?

In [52]:
```
plt.figure(figsize=(14,7))
plt.xticks(rotation=60)
plt.xlabel("Category")
plt.ylabel("Number of applications")
plt.title("Top 10 Categories")
sns.barplot(x = top_10_Categories_installs.Category, y = top_10_Categories_installs.Count)
```

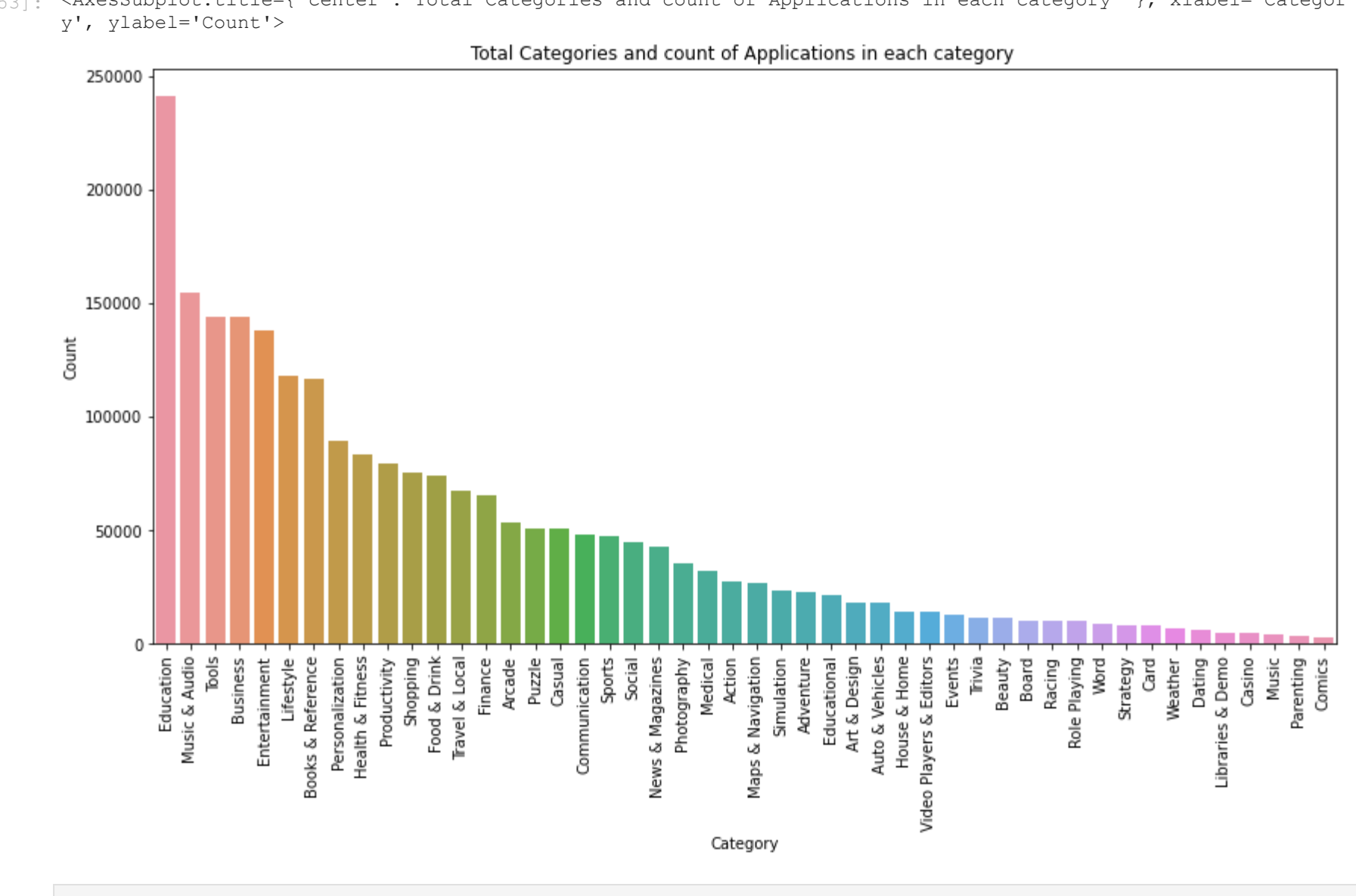
Out[52]: <AxesSubplot:title={'center':'Top 10 Categories'}, xlabel='Category', ylabel='Count'>



1. (ANS) Education, Music & Audio, Tools, Business, Entertainment, Lifestyle, Books & Reference, Personalization, Health & Fitness, Productivity.

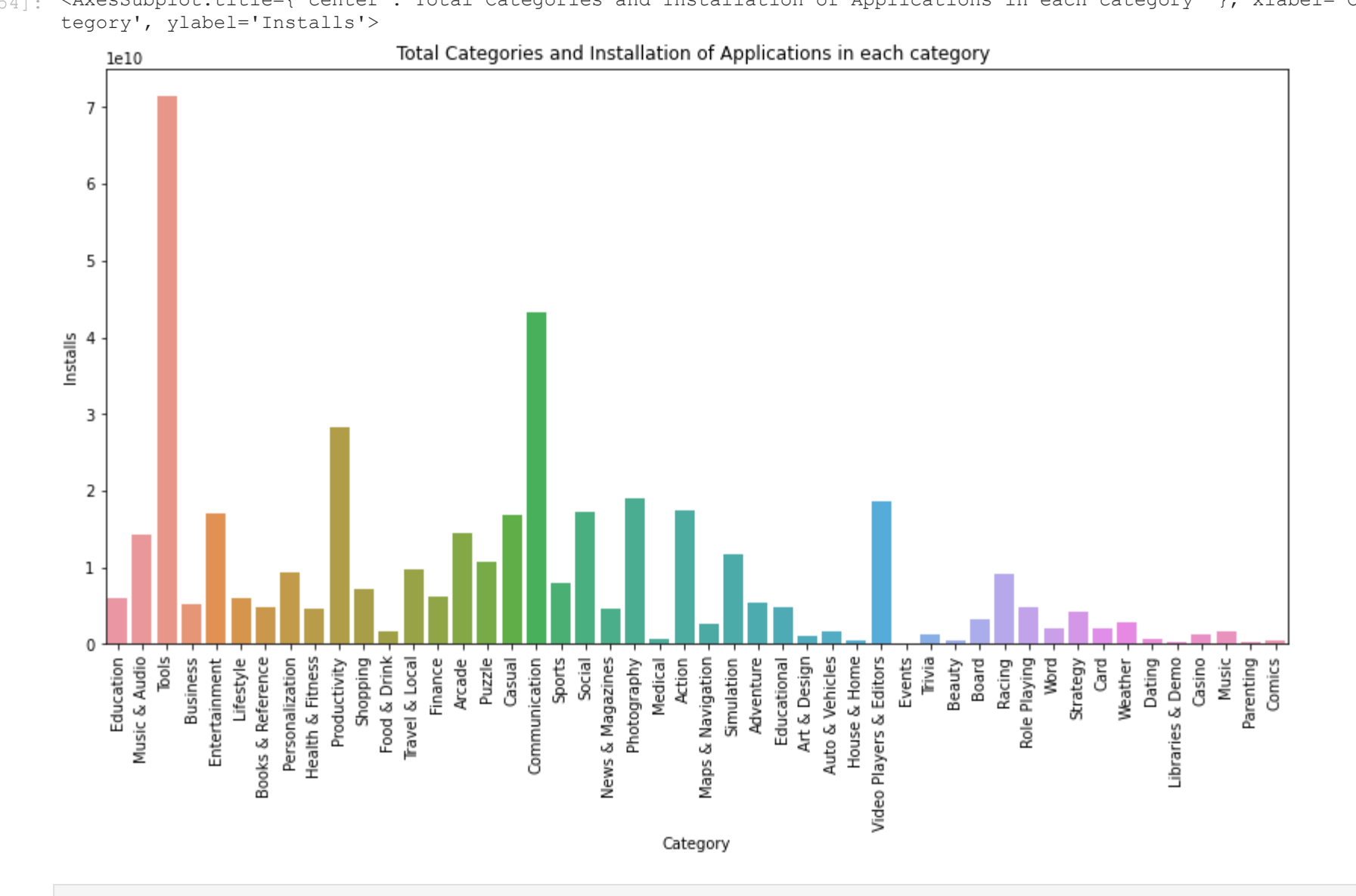
In [53]:
```
# Total Categories and count of Apps in each category.

plt.figure(figsize=(14,7))
plt.xticks(rotation=90)
plt.xlabel("Category")
plt.ylabel("Number of applications")
plt.title("Total Categories and count of Applications in each category ")
sns.barplot(x = top_Category_installs.Category, y = top_Category_installs.Count)
```

Out[53]: <AxesSubplot:title={'center':'Total Categories and count of Applications in each category '}, xlabel='Category', ylabel='Count'>



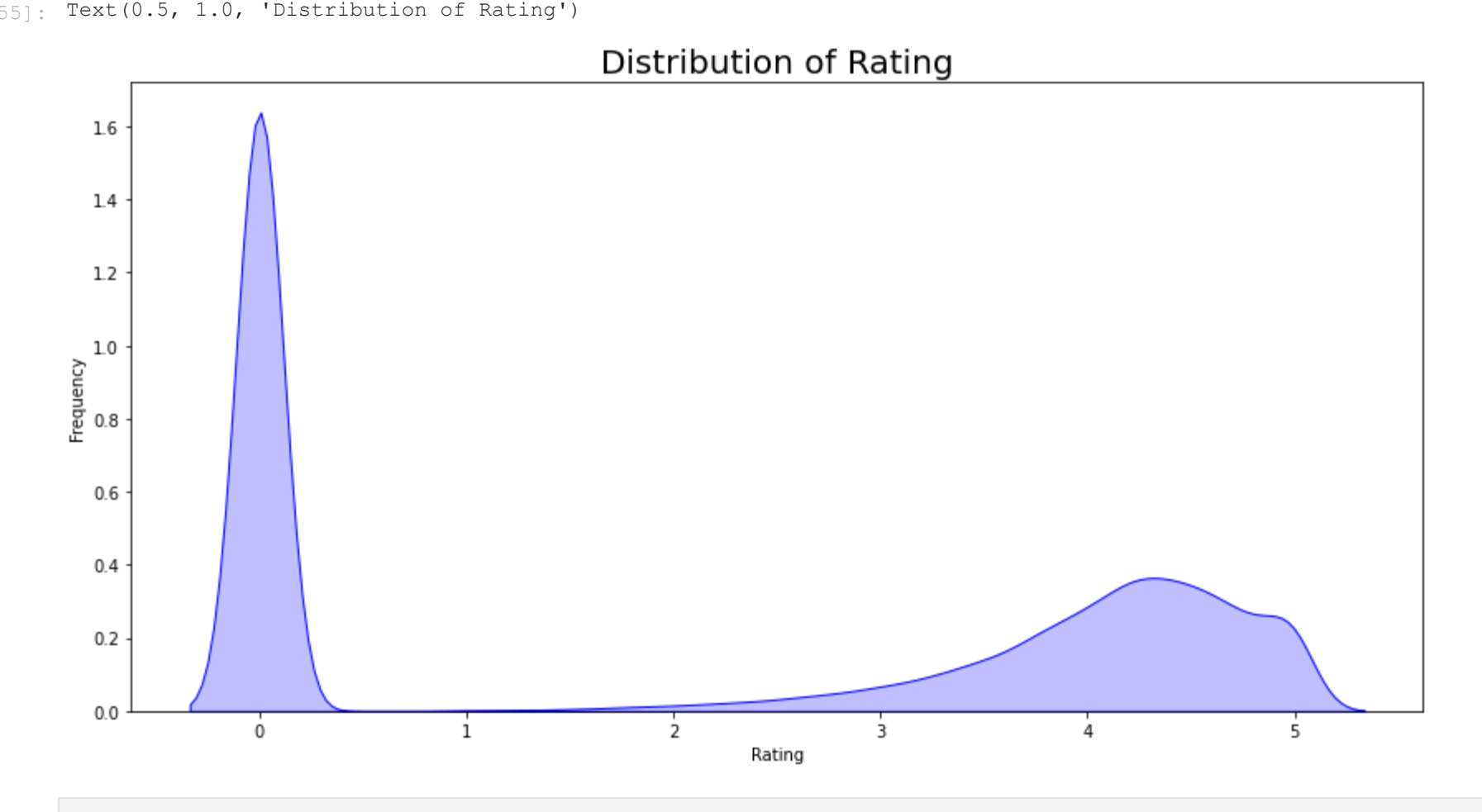
In [54]:
```
# Total Categories and installed Applications in each category

plt.figure(figsize=(14,7))
plt.xticks(rotation=90)
plt.xlabel("Category")
plt.ylabel("Number of applications")
plt.title("Total Categories and Installation of Applications in each category ")
sns.barplot(x = top_Category_installs.Category,y = top_Category_installs.Installs)
```

Out[54]: <AxesSubplot:title={'center':'Total Categories and Installation of Applications in each category '}, xlabel='Category', ylabel='Installs'>



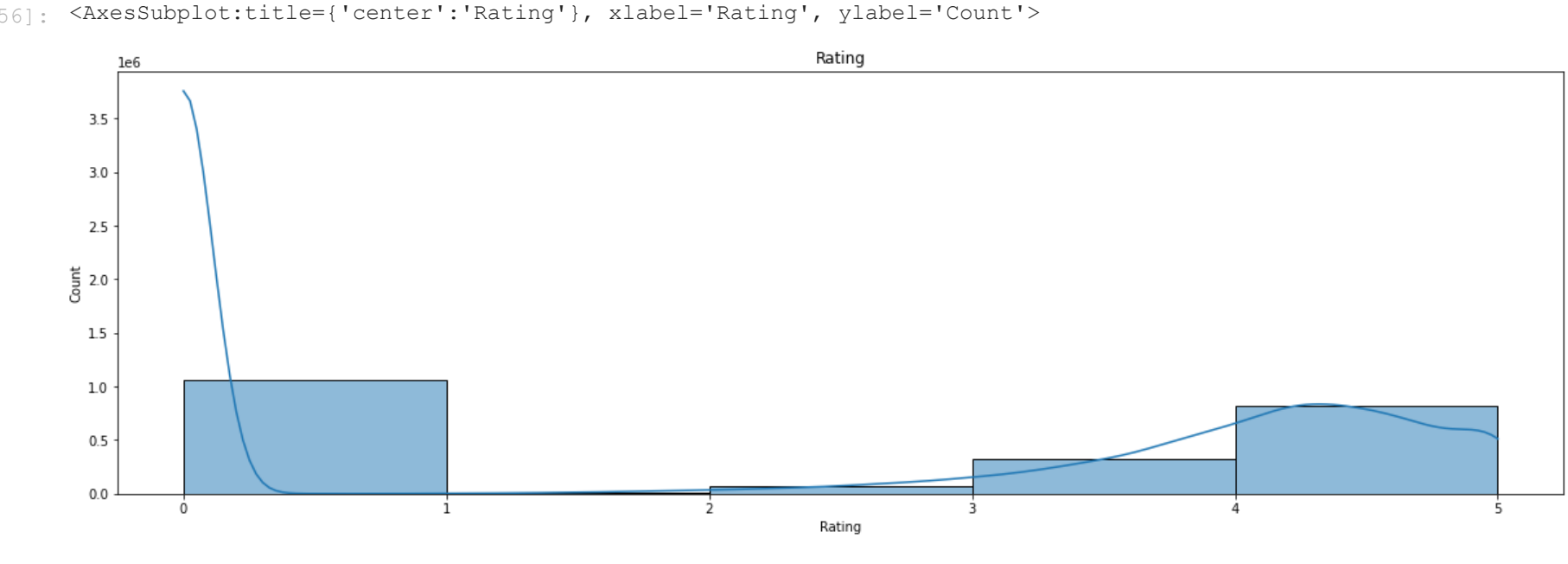
In [55]:
```
# Rating distribution

plt.figure(figsize=(14,7))
g = sns.kdeplot(df.Rating, color="Blue", shade = True)
g.set_xlabel("Rating")
g.set_ylabel("Frequency")
plt.title('Distribution of Rating',size = 20)
```

Out[55]: Text(0.5, 1.0, 'Distribution of Rating')



In [56]:
```
plt.title("Rating")
sns.histplot(df.Rating,kde=True,bins=5)
```

Out[56]: <AxesSubplot:title={'center':'Rating'}, xlabel='Rating', ylabel='Count'>

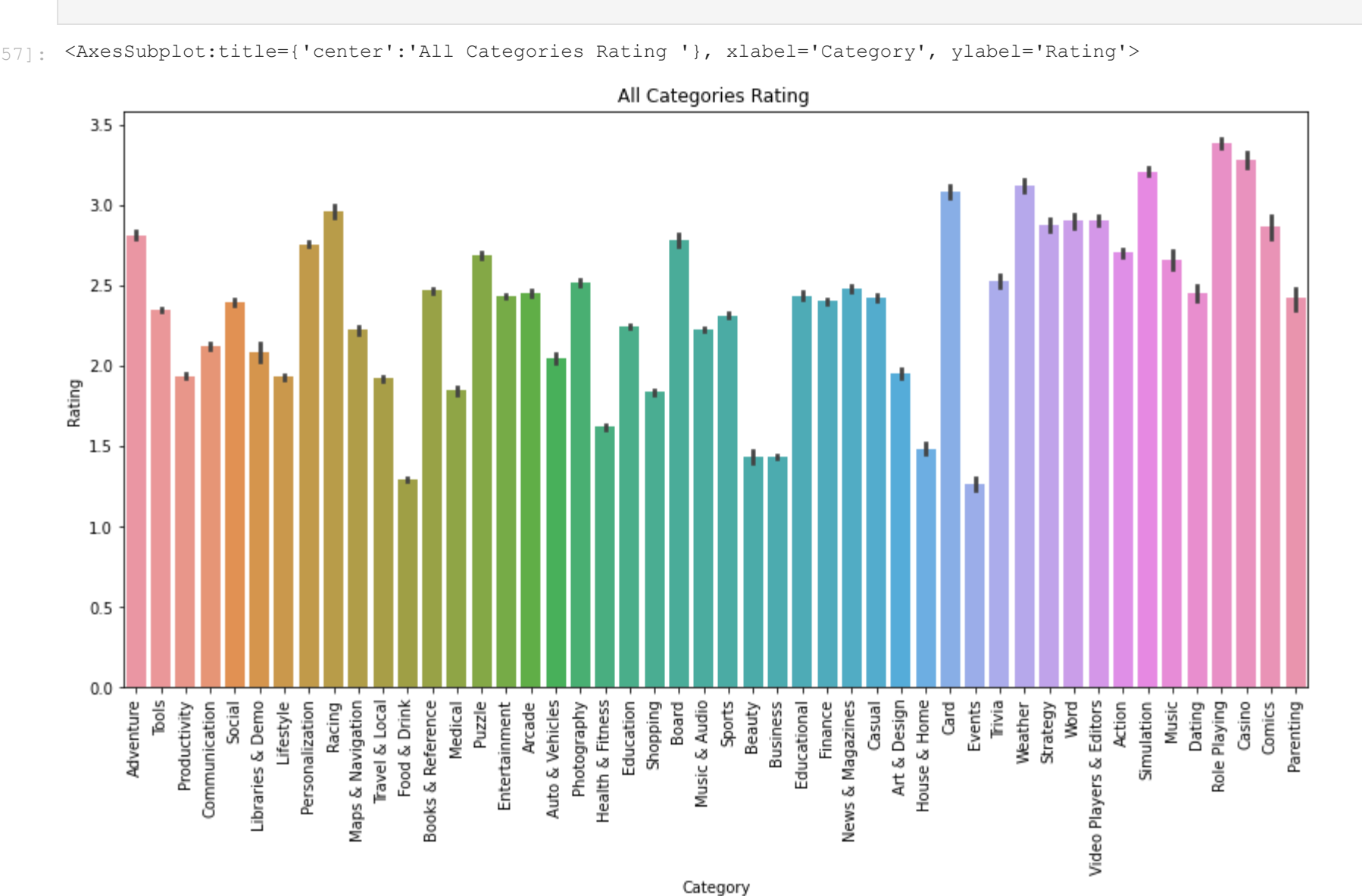


OBSERVATION:

From the above two plots I can see that most people does not give a rating, but, one more thing that comes out from this graph as well is that people tend to give 4+ rating the most.

In [57]:
```
# 3. What is the highest rated Category?
plt.figure(figsize=(14,7))
plt.xticks(rotation=90)
plt.xlabel("Highest Rated Category")
plt.ylabel("Number of applications")
plt.title("All Categories Rating ")
sns.barplot(x = df.Category, y = df.Rating)
```

Out[57]: <AxesSubplot:title={'center':'All Categories Rating '}, xlabel='Category', ylabel='Rating'>



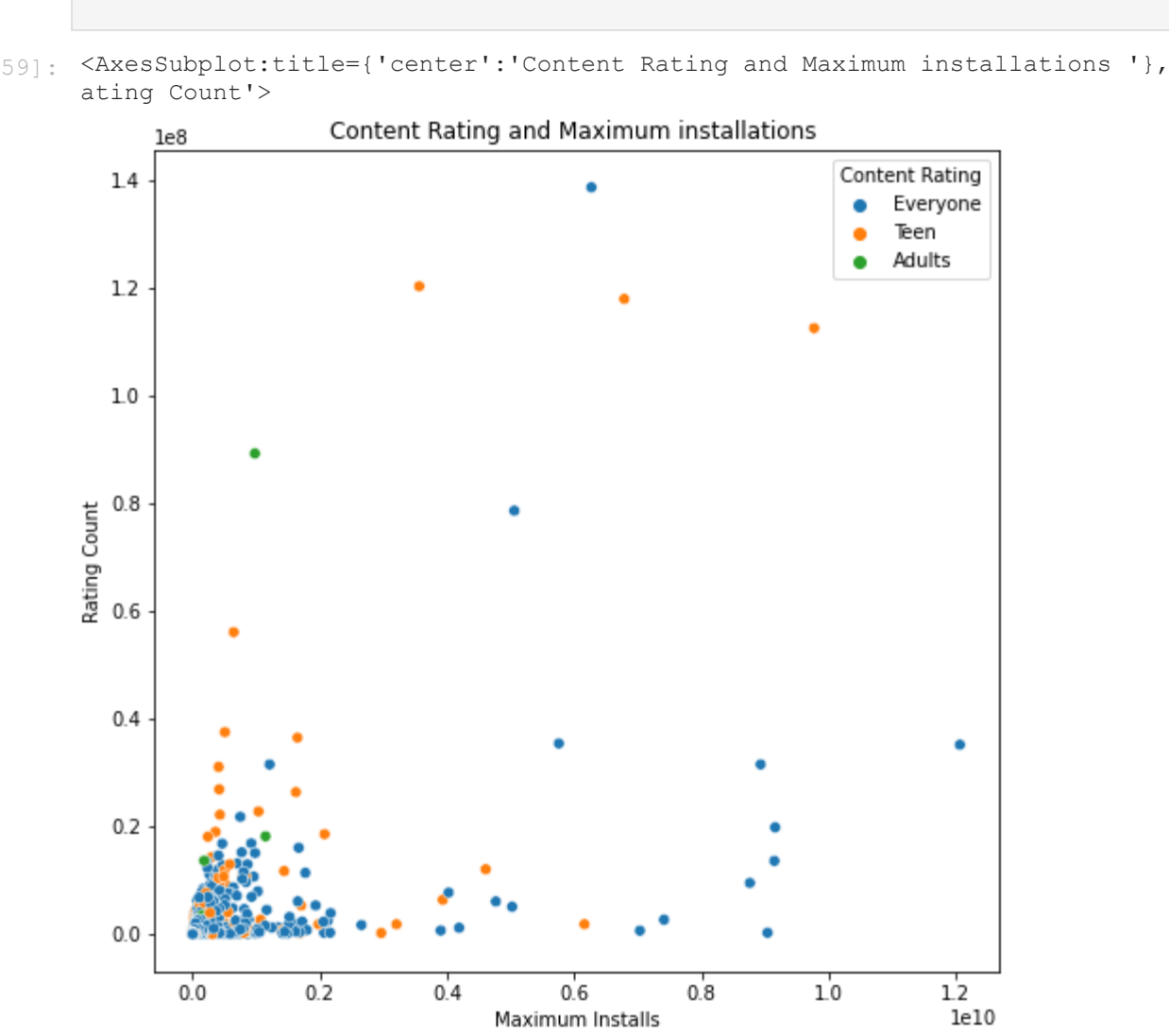
1. (ANS) - Role Playing

In [58]:
```
df['Content Rating'].unique()
```

Out[58]: array(['Everyone', 'Teen', 'Adults'], dtype=object)

In [59]:
```
plt.figure(figsize=(8,6))
plt.title("Content Rating and Maximum installations ")
sns.scatterplot(x='Maximum Installs',y='Rating Count',data=df,hue='Content Rating')
```

Out[59]: <AxesSubplot:title={'center':'Content Rating and Maximum installations '}, xlabel='Maximum Installs', ylabel='Rating Count'>

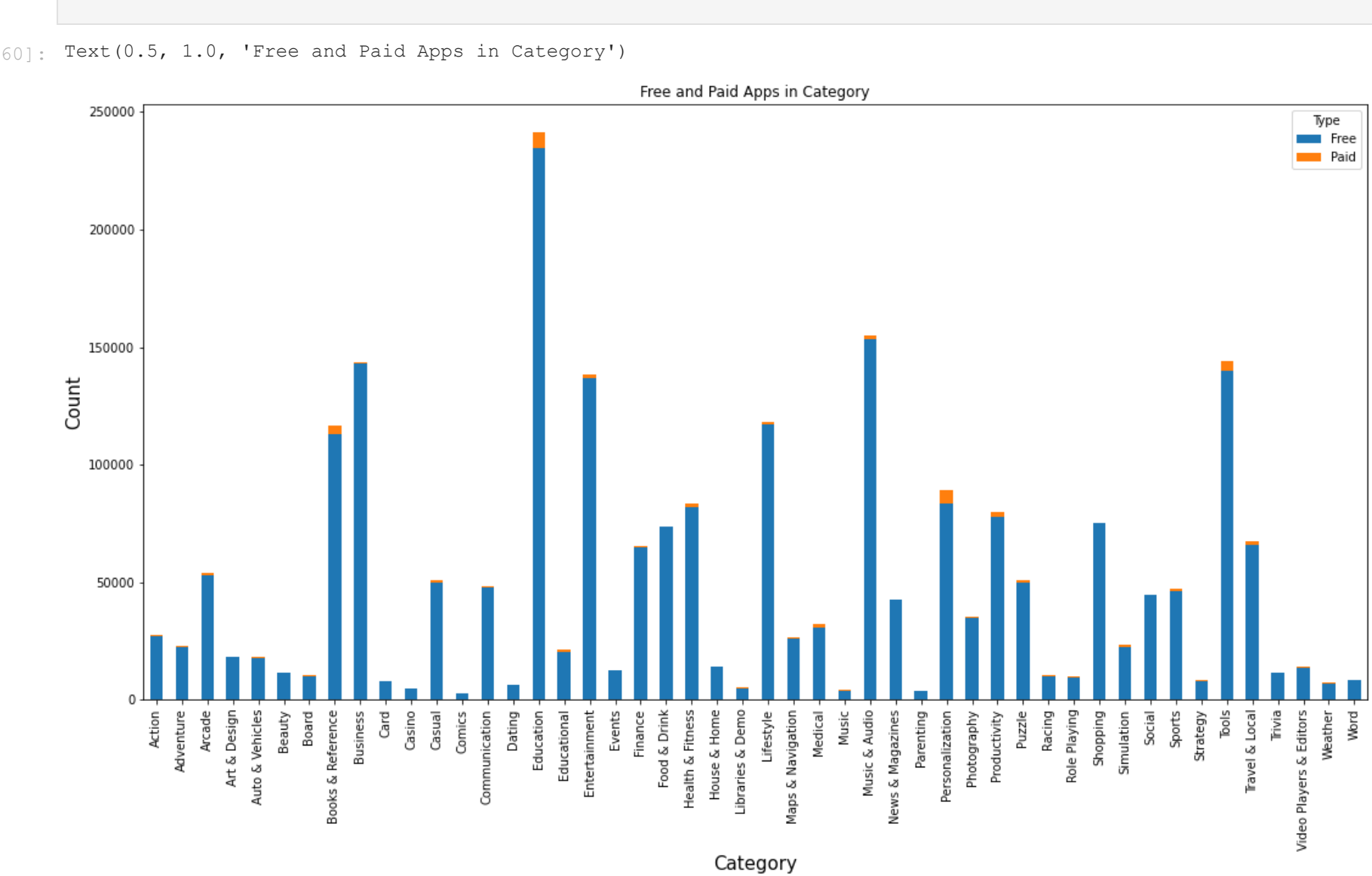


OBSERVATION:

This scatterplot shows that: If I exclude everyone from the plot and focus on Teen and Adults I can see that teens have much engagement in terms of download and rating count.

In [60]:
```
# 4. Which Category has the highest Paid and Free apps ?

app_count = df.groupby(['Category','Type'])[['App Name']].count().reset_index().rename(columns={'App Name':'Co
df_app_count = app_count.pivot('Category', 'Type', 'Count').fillna(0).reset_index()

df_app_count.set_index('Category').plot(kind='bar', stacked=True, figsize=(18,9)) # try df_app_count.plot(kind=
plt.xlabel("Category", fontsize=15)
plt.ylabel("Count", fontsize=15)
plt.title("Free and Paid Apps in Category")
```

Out[60]: Text(0.5, 1.0, 'Free and Paid Apps in Category')



4.(ANS) Education

OBSERVATION:

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 and Social categories were free to install. At the same time Books & References, Education, Tools and Personalization categories had the biggest number of paid apps available for installation.

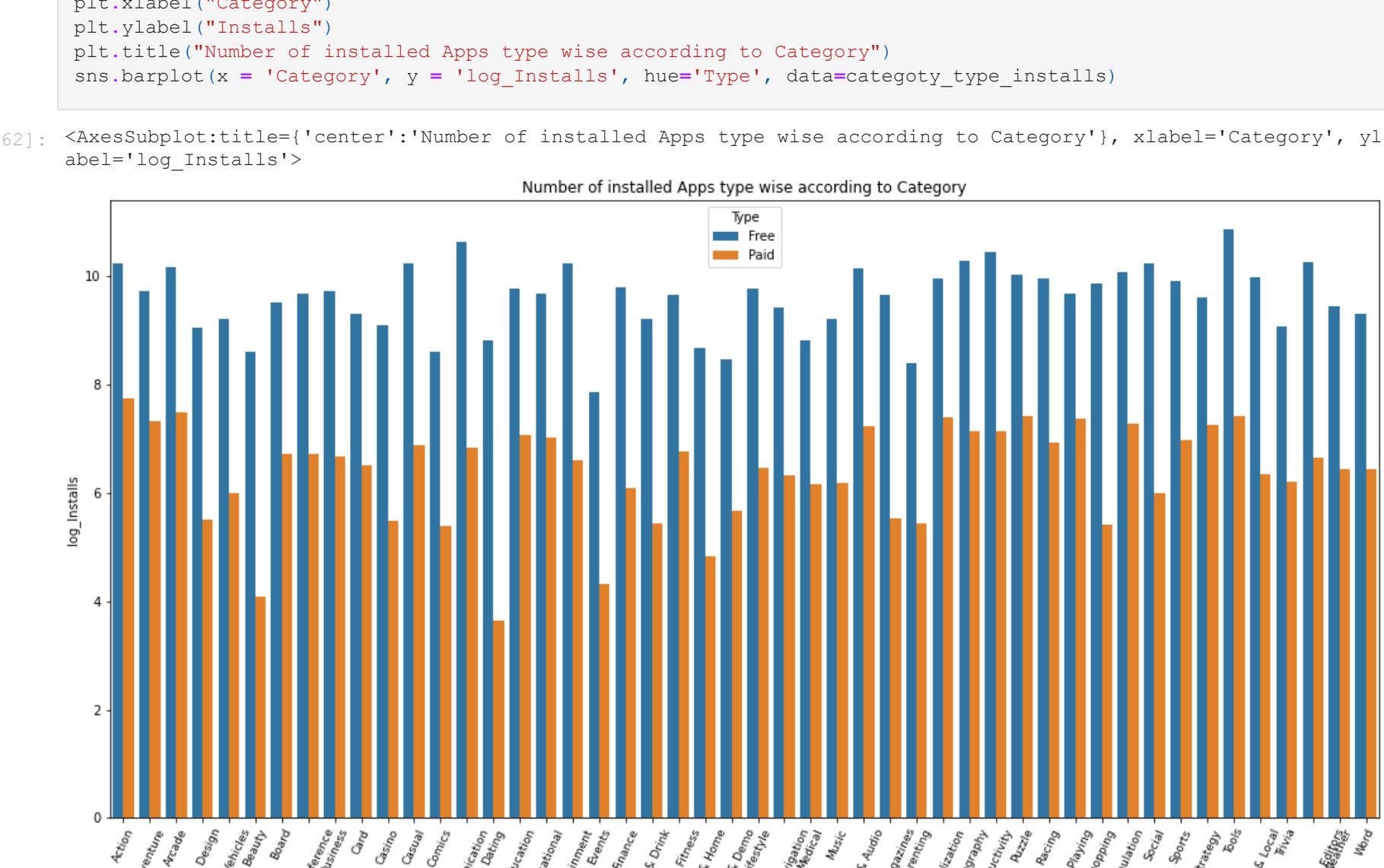
In [61]:
```
# Visualize the installation type in each category
df['Installs'].min(),df['Installs'].max()
```

Out[61]: (0, 10000000000)

There is a high variance in the number of installs, I need to reduce it so I can use a log value for this column, otherwise it would be unable to see the data when visualized.

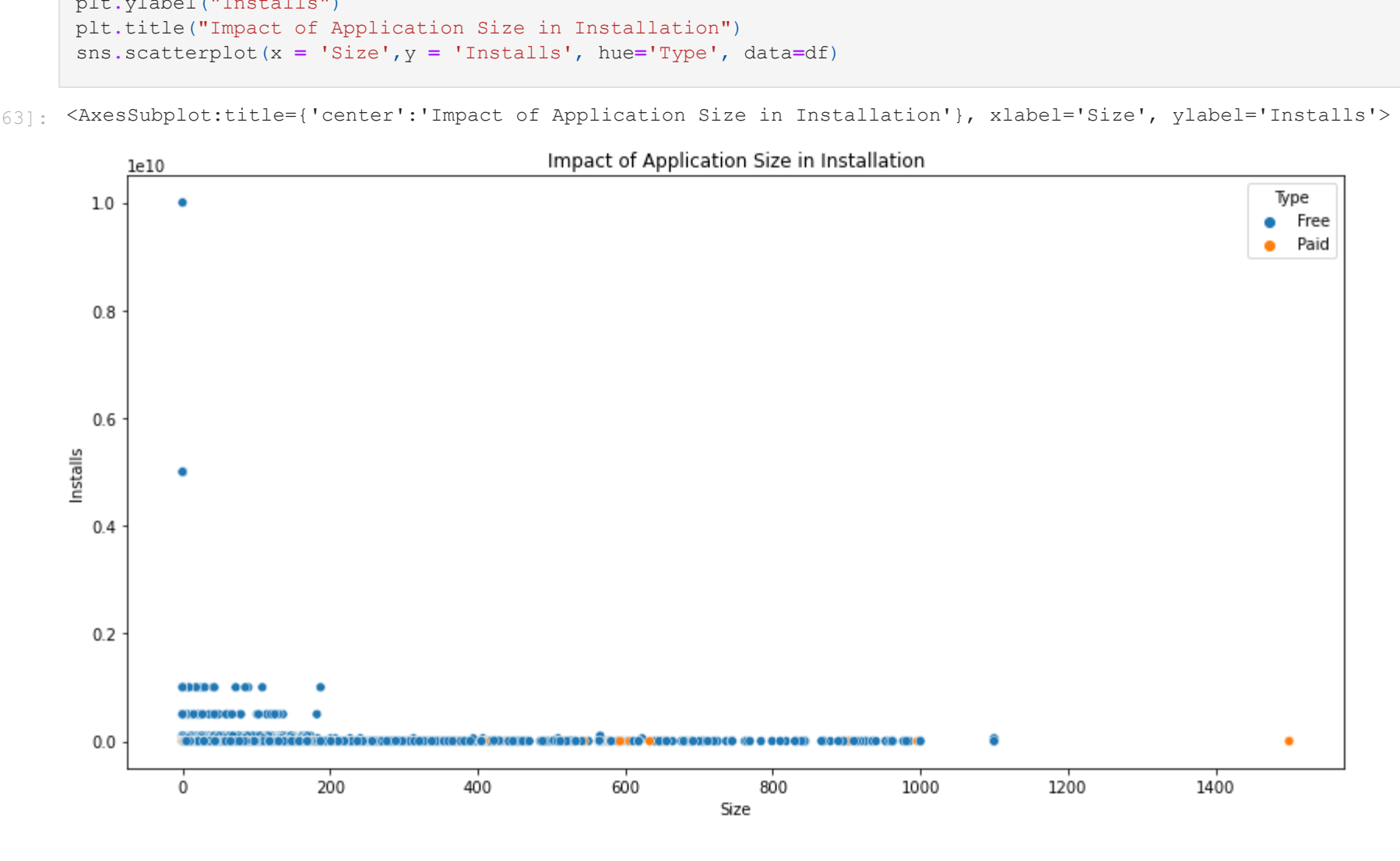
In [62]:
```
category_type_installs = df.groupby(['Category','Type'])[['Installs']].sum().reset_index()
category_type_installs['log_Installs'] = np.log10(category_type_installs['Installs'])

plt.figure(figsize=(18,9))
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[62]: <AxesSubplot:title={'center':'Number of installed Apps type wise according to Category'}, xlabel='Category', ylabel='log_Installs'>



In [63]:
```
# 5. How does the size of the Application impacts the Installation ?
df['log_Installs'] = np.log10(category_type_installs['Installs'])

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[63]: <AxesSubplot:title={'center':'Impact of Application Size in Installation'}, xlabel='Size', ylabel='Installs'>



1. (ANS)From the above plot we can see that size impacts the number of installations. Applications with large size are less installed by the user.

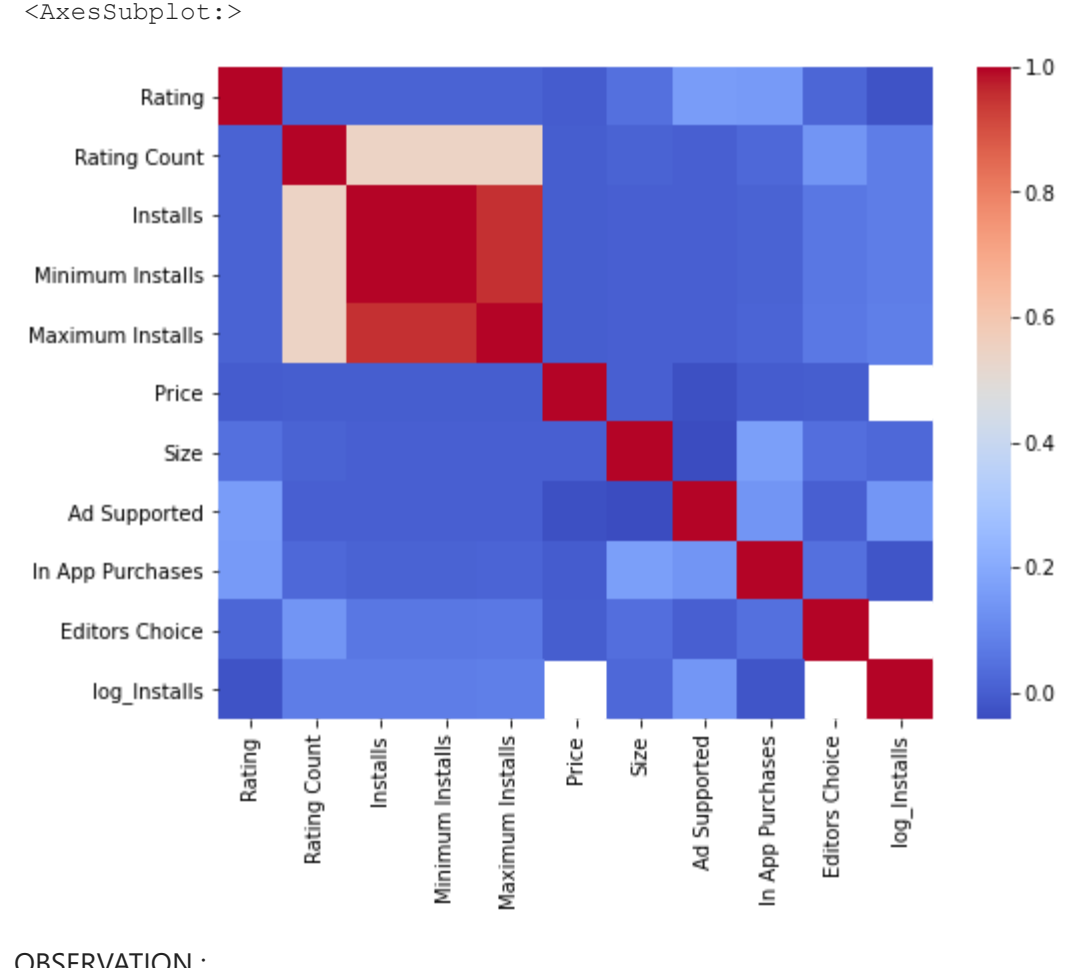
In [64]:
```
# Examing correlations
df.corr()
```

Out[64]:

	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Price	Size	Ad Supported	In App Purchases	Editors Choice	log_Installs
Rating	1.000000	0.013038	0.011215	0.011215	0.012615	-0.003674	0.046099	0.160503	0.153824	0.019485	-0.023632
Rating Count	0.013038	1.000000	0.545281	0.545281	0.547571	-0.000411	0.013600	0.005294	0.025737	0.137311	0.076834
Installs	0.011215	0.545281	1.000000	1.000000	0.954037	-0.000461	0.004240	0.002249	0.014178	0.059932	0.077248
Minimum Installs	0.011215	0.545281	1.000000	1.000000	0.954037	-0.000461	0.004240	0.002249	0.014178	0.059932	0.077248
Maximum Installs	0.012615	0.547571	0.954037	0.954037	1.000000	-0.000515	0.005093	0.002881	0.016100	0.064206	0.079792
Price	-0.003674	-0.000411	-0.000461	-0.000461	-0.000515	1.000000	0.004160	-0.034281	-0.002636	0.000503	NaN
Size	0.046099	0.013600	0.004240	0.004240	0.005093	0.004160	1.000000	-0.042772	0.165194	0.039926	0.024328
Ad Supported	0.160503	0.005294	0.002249	0.002249	0.002881	-0.034281	-0.042772	1.000000	0.138304	0.005701	0.140935
In App Purchases	0.153824	0.025737	0.014178	0.014178	0.016100	-0.002636	0.165194	0.138304	1.000000	0.046078	-0.021114
Editors Choice	0.019485	0.137311	0.059932	0.059932	0.064206	0.000503	0.039926	0.005701	0.046078	1.000000	NaN
log_Installs	-0.023632	0.076834	0.077248	0.077248	0.079792	NaN	0.024328	0.140935	-0.021114	NaN	1.000000

In [65]:
```
plt.figure(figsize=(8,6))
sns.heatmap(df.corr(),cbar= True, cmap='coolwarm')
```

Out[65]: <AxesSubplot:>



OBSERVATION :

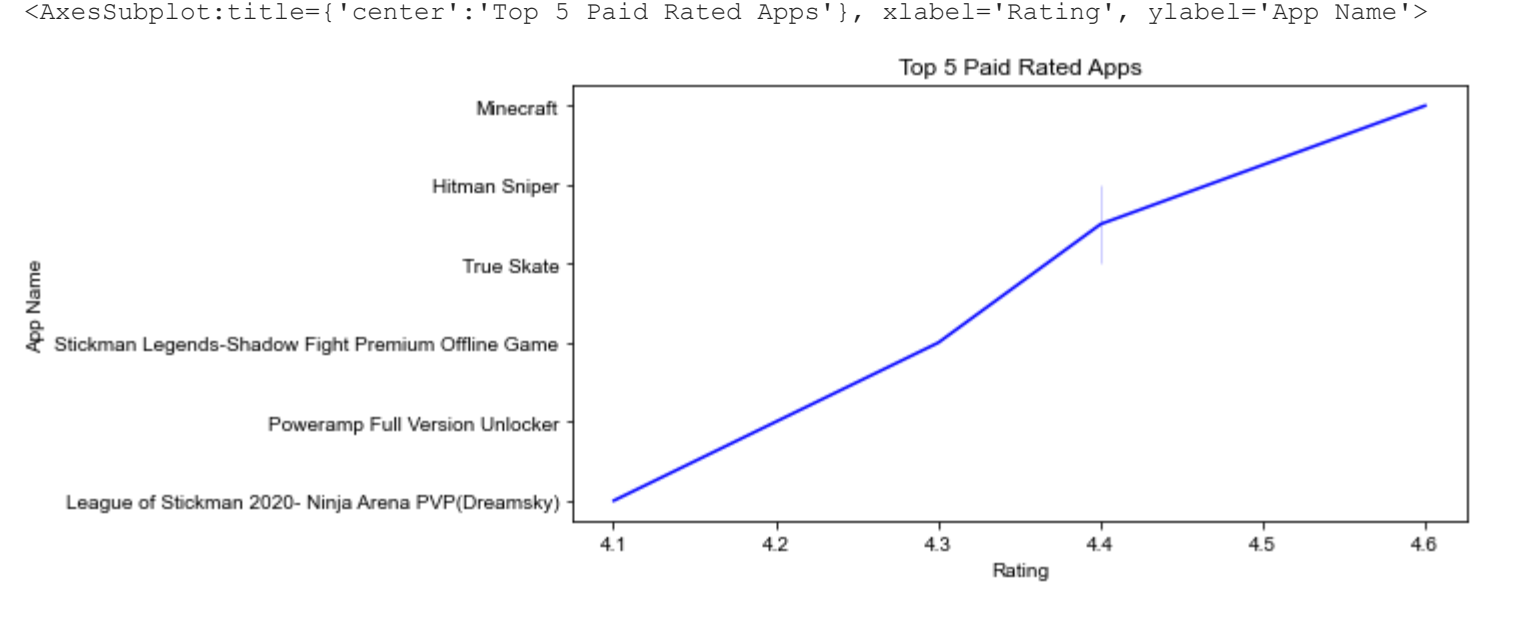
Factors like AdSupport and InappPurchases are correlated to app rating. So I can say that if the app provides customer support and have subscription plans I can engage more customers. I can also see from the same graph that the editor's choice plays a important role as well. With high editor choice I can see high ratings count and high installs.

In [66]:
```
# 6. What are the Top 5 Paid Apps based With highest ratings and installs ?
df['Installs'].max()
```

Out[66]: 10000000000

In [69]:
```
paid_apps = df[(df.Type=='Paid') & (df.Installs >= 5000000)]
paid_apps = paid_apps.groupby('App Name')['Rating'].max().sort_values(ascending = False)

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[69]: <AxesSubplot:title={'center':'Top 5 Paid Rated Apps'}, xlabel='Rating', ylabel='App Name'>

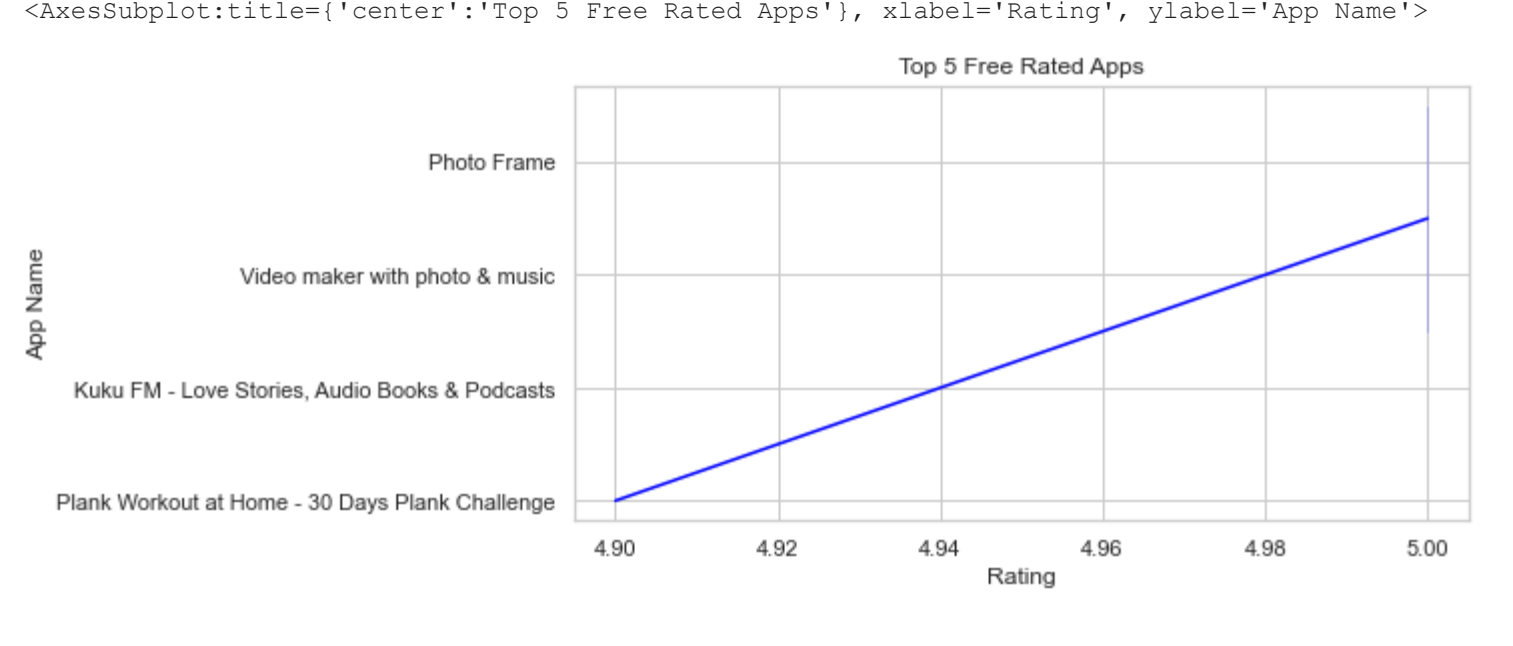


1. (ANS) Minecraft, Hitman Sniper, True Skate, Stickman Legends-Shadow Fight Premium Offline Game, Poweramp Full Version Unlocker, League of Stickman 2020- Ninja Arena PVP(Dreamsky)

In [70]:
```
# 7.  What are the Top 5 Free Apps based with highest ratings and installs ?

free_apps = df[(df.Type=='Free') & (df.Installs >= 5000000)]
free_apps = free_apps.groupby('App Name')['Rating'].max().sort_values(ascending = False)
free_apps = free_apps.head(5)
```

In [71]:
```
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')
```

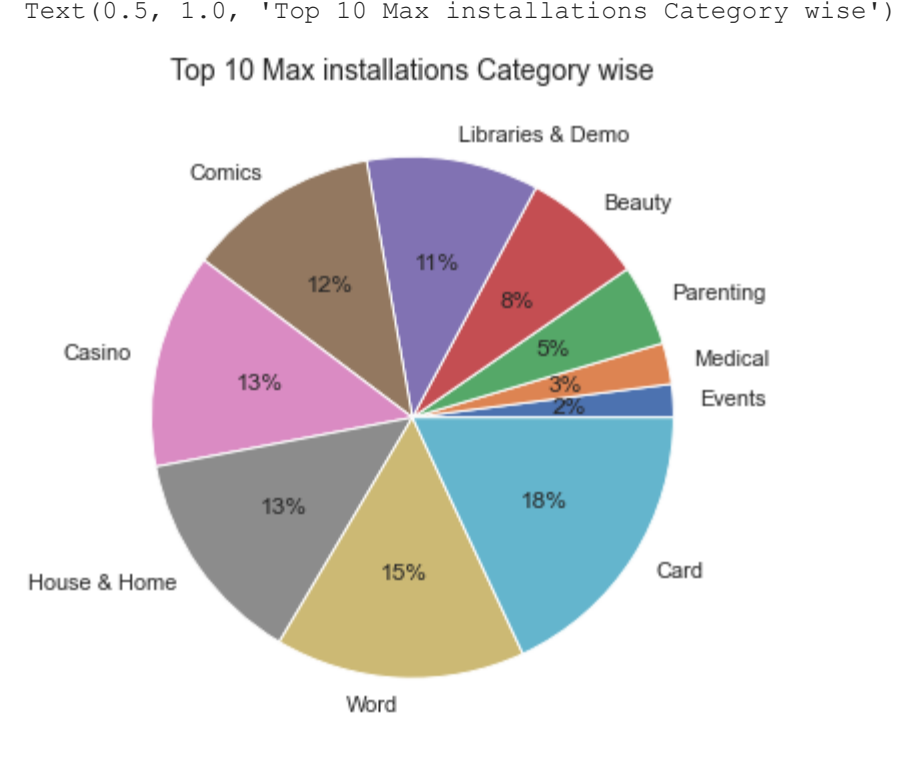
Out[71]: <AxesSubplot:title={'center':'Top 5 Free Rated Apps'}, xlabel='Rating', ylabel='App Name'>



1. (ANS) Photo Frame, Video maker with photo & music, Kuku FM - Love Stories,Audio Books & Podcasts, Plank Workout at Home.

In [72]:
```
# Visualize the categories that have the Top 10 Max Installations
plt.figure(figsize=(8,6))
data = df.groupby('Category')['Maximum Installs'].max().sort_values(ascending = True)
data = data.head(10)
labels = data.keys()
plt.pie(data, labels= labels,autopct='%.0f%%')
plt.title("Top 10 Max installations Category wise", fontsize=14)
```

Out[72]: Text(0.5, 1.0, 'Top 10 Max installations Category wise')



SUMMARY AND CONCLUSION:

```
- People are more interested in installing gaming apps, the top ratings are given to the gaming apps.
- InAppPurchases are correlated to App rating. So we can say that if the app provides customer support
and have subscription plans it will helps to engage customers.
- Most people do not give ratings, but,  the people who are give ratings tend to give 4+ rating most
times.
- Most adults  installed the "Social" and "Communication" apps.
- Most of the installations that are done by teens is "Video Players" and "Editors". Video Players and
Editors has more demand.
- Size of the Application varies the installations.
- People mostly download the free apps the installation of the free apps is high and the availabilty of
the free apps also is very high.
```

In [ ]: