google-play-store

December 10, 2023

1 Google playstore Data

DataSet : https://www.kaggle.com/datasets/lava18/google-play-store-apps

1.1 Libraries

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

2 Loading and exploration

[2]:
```
df = pd.read_csv('googleplaystore.csv')
```

[3]:
```
df.head(5)
```

[3]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	1.0.0
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0
2	U Launcher Lite – FREE Live Cool Themes, Hide …	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	1.2.4
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	June 8, 2018	Varies with device
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design		

```
4    Art & Design;Creativity     June 20, 2018                1.1

    Android Ver  Unnamed: 13
0  4.0.3 and up          NaN
1  4.0.3 and up          NaN
2  4.0.3 and up          NaN
3    4.2 and up          NaN
4    4.4 and up          NaN
```

[4]:
```
#  import warnings
# warnings.filterwarnings('ignore')
```

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

[5]:
```
Index(['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type',
       'Price', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver',
       'Android Ver', 'Unnamed: 13'],
      dtype='object')
```

[6]:
```
df.shape
```

[6]:
```
(10841, 14)
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10841 entries, 0 to 10840
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   App             10840 non-null  object
 1   Category        10841 non-null  object
 2   Rating          9367 non-null   float64
 3   Reviews         10841 non-null  int64
 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  10841 non-null  object
 9   Genres          10840 non-null  object
 10  Last Updated    10841 non-null  object
 11  Current Ver     10833 non-null  object
 12  Android Ver     10839 non-null  object
 13  Unnamed: 13     0 non-null      float64
dtypes: float64(2), int64(1), object(11)
memory usage: 1.2+ MB
```

[8]:
```
df.describe()
```

[8]:

	Rating	Reviews	Unnamed: 13
count	9367.000000	1.084100e+04	0.0
mean	4.191513	4.441119e+05	NaN
std	0.515735	2.927629e+06	NaN
min	1.000000	0.000000e+00	NaN
25%	4.000000	3.800000e+01	NaN
50%	4.300000	2.094000e+03	NaN
75%	4.500000	5.476800e+04	NaN
max	5.000000	7.815831e+07	NaN

[9]:
```
# check for null values
df['Size'].isnull().sum()
```

[9]:
```
0
```

[10]:
```
# check unique values
df['Size'].unique()
```

[10]:
```
array(['19M', '14M', '8.7M', '25M', '2.8M', '5.6M', '29M', '33M', '3.1M',
       '28M', '12M', '20M', '21M', '37M', '2.7M', '5.5M', '17M', '39M',
       '31M', '4.2M', '7.0M', '23M', '6.0M', '6.1M', '4.6M', '9.2M',
       '5.2M', '11M', '24M', 'Varies with device', '9.4M', '15M', '10M',
       '1.2M', '26M', '8.0M', '7.9M', '56M', '57M', '35M', '54M', '201k',
       '3.6M', '5.7M', '8.6M', '2.4M', '27M', '2.5M', '16M', '3.4M',
       '8.9M', '3.9M', '2.9M', '38M', '32M', '5.4M', '18M', '1.1M',
       '2.2M', '4.5M', '9.8M', '52M', '9.0M', '6.7M', '30M', '2.6M',
       '7.1M', '3.7M', '22M', '7.4M', '6.4M', '3.2M', '8.2M', '9.9M',
       '4.9M', '9.5M', '5.0M', '5.9M', '13M', '73M', '6.8M', '3.5M',
       '4.0M', '2.3M', '7.2M', '2.1M', '42M', '7.3M', '9.1M', '55M',
       '23k', '6.5M', '1.5M', '7.5M', '51M', '41M', '48M', '8.5M', '46M',
       '8.3M', '4.3M', '4.7M', '3.3M', '40M', '7.8M', '8.8M', '6.6M',
       '5.1M', '61M', '66M', '79k', '8.4M', '118k', '44M', '695k', '1.6M',
       '6.2M', '18k', '53M', '1.4M', '3.0M', '5.8M', '3.8M', '9.6M',
       '45M', '63M', '49M', '77M', '4.4M', '4.8M', '70M', '6.9M', '9.3M',
       '10.0M', '8.1M', '36M', '84M', '97M', '2.0M', '1.9M', '1.8M',
       '5.3M', '47M', '556k', '526k', '76M', '7.6M', '59M', '9.7M', '78M',
       '72M', '43M', '7.7M', '6.3M', '334k', '34M', '93M', '65M', '79M',
       '100M', '58M', '50M', '68M', '64M', '67M', '60M', '94M', '232k',
       '99M', '624k', '95M', '8.5k', '41k', '292k', '11k', '80M', '1.7M',
       '74M', '62M', '69M', '75M', '98M', '85M', '82M', '96M', '87M',
       '71M', '86M', '91M', '81M', '92M', '83M', '88M', '704k', '862k',
       '899k', '378k', '266k', '375k', '1.3M', '975k', '980k', '4.1M',
       '89M', '696k', '544k', '525k', '920k', '779k', '853k', '720k',
       '713k', '772k', '318k', '58k', '241k', '196k', '857k', '51k',
       '953k', '865k', '251k', '930k', '540k', '313k', '746k', '203k',
```

```
       '26k', '314k', '239k', '371k', '220k', '730k', '756k', '91k',
       '293k', '17k', '74k', '14k', '317k', '78k', '924k', '902k', '818k',
       '81k', '939k', '169k', '45k', '475k', '965k', '90M', '545k', '61k',
       '283k', '655k', '714k', '93k', '872k', '121k', '322k', '1.0M',
       '976k', '172k', '238k', '549k', '206k', '954k', '444k', '717k',
       '210k', '609k', '308k', '705k', '306k', '904k', '473k', '175k',
       '350k', '383k', '454k', '421k', '70k', '812k', '442k', '842k',
       '417k', '412k', '459k', '478k', '335k', '782k', '721k', '430k',
       '429k', '192k', '200k', '460k', '728k', '496k', '816k', '414k',
       '506k', '887k', '613k', '243k', '569k', '778k', '683k', '592k',
       '319k', '186k', '840k', '647k', '191k', '373k', '437k', '598k',
       '716k', '585k', '982k', '222k', '219k', '55k', '948k', '323k',
       '691k', '511k', '951k', '963k', '25k', '554k', '351k', '27k',
       '82k', '208k', '913k', '514k', '551k', '29k', '103k', '898k',
       '743k', '116k', '153k', '209k', '353k', '499k', '173k', '597k',
       '809k', '122k', '411k', '400k', '801k', '787k', '237k', '50k',
       '643k', '986k', '97k', '516k', '837k', '780k', '961k', '269k',
       '20k', '498k', '600k', '749k', '642k', '881k', '72k', '656k',
       '601k', '221k', '228k', '108k', '940k', '176k', '33k', '663k',
       '34k', '942k', '259k', '164k', '458k', '245k', '629k', '28k',
       '288k', '775k', '785k', '636k', '916k', '994k', '309k', '485k',
       '914k', '903k', '608k', '500k', '54k', '562k', '847k', '957k',
       '688k', '811k', '270k', '48k', '329k', '523k', '921k', '874k',
       '981k', '784k', '280k', '24k', '518k', '754k', '892k', '154k',
       '860k', '364k', '387k', '626k', '161k', '879k', '39k', '970k',
       '170k', '141k', '160k', '144k', '143k', '190k', '376k', '193k',
       '246k', '73k', '658k', '992k', '253k', '420k', '404k', '470k',
       '226k', '240k', '89k', '234k', '257k', '861k', '467k', '157k',
       '44k', '676k', '67k', '552k', '885k', '1020k', '582k', '619k'],
      dtype=object)
```

There are several uniques values in the Size column, we have to first make the unit into one common unit from M and K to bytes, and then remove the M and K from the values and convert them into numeric data type.

[11]:
```
# find the values in size column which has 'M' in it
df['Size'].loc[df['Size'].str.contains('M')].value_counts().sum()
```

[11]: 8830

[12]:
```
# find the values in size column which has 'k' in it
df['Size'].loc[df['Size'].str.contains('k')].value_counts().sum()
```

[12]: 316

[13]:
```
# Total Values in Size column
df['Size'].value_counts().sum()
```

[13]: 10841

```
[14]: # taking sum of all the values in size column which has 'M', 'K' and 'varies␣
       ↪with device' in it
      8830+316+1695
```

[14]: 10841

We have 8830 values in M units We have 316 values in k units We have 1695 value in Varies with device

converting the M and K units into bytes and then remove the M and K from the values and convert them into numeric data type.

```
[15]: # convert the size column to numeric by multiplying the values with 1024 if it␣
       ↪has 'k' in it and 1024*1024 if it has 'M' in it
      # this function will convert the size column to numeric
      def convert_size(size):

          if isinstance(size, str):
              if 'k' in size:
                  return float(size.replace('k', '')) * 1024
              elif 'M' in size:
                  return float(size.replace('M', '')) * 1024 * 1024
              elif 'Varies with device' in size:
                  return np.nan
          return size

      df['Size'] = df['Size'].apply(convert_size)
```

```
[16]: # rename the column name 'Size' to 'Size_in_bytes'
      df.rename(columns={'Size': 'Size_in_bytes'}, inplace=True)
```

```
[17]: # making a new column called 'Size in Mb' which will have the size in MB
      df['Size_in_Mb'] = df['Size_in_bytes'].apply(lambda x: x/(1024*1024))
```

Now we have converted every value into bytes and removed the M and K from the values and converted them into numeric data type.

```
[18]: # check the unique values in size column
      df['Installs'].unique()
```

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

[19]:
```
# let's have a values counts
df['Installs'].value_counts()
```

[19]:
```
1,000,000+        1579
10,000,000+       1252
100,000+          1169
10,000+           1054
1,000+             908
5,000,000+         752
100+               719
500,000+           539
50,000+            479
5,000+             477
100,000,000+       409
10+                386
500+               330
50,000,000+        289
50+                205
5+                  82
500,000,000+        72
1+                  67
1,000,000,000+      58
0+                  14
0                    1
Name: Installs, dtype: int64
```

[20]:
```
# find how many values has '+' in it
df['Installs'].loc[df['Installs'].str.contains('\+')].value_counts().sum()
```

[20]:
```
10840
```

[21]:
```
# Total values in Installs column
df['Installs'].value_counts().sum()
```

[21]:
```
10841
```

[22]:
```
df.head() # check the head of the dataframe
```

[22]:

	App	Category	Rating	Reviews	Size_in_bytes	Installs	Type	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19922944.0	10,000+	Free	0	Everyone
1	Coloring book moana	ART_AND_DESIGN	3.9						
2	U Launcher Lite - FREE Live Cool Themes, Hide …	ART_AND_DESIGN	4.7						
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5						
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3						

```
1      967     14680064.0     500,000+  Free     0       Everyone
2    87510      9122611.2   5,000,000+  Free     0       Everyone
3   215644     26214400.0  50,000,000+  Free     0           Teen
4      967      2936012.8     100,000+  Free     0       Everyone

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

    Android Ver  Unnamed: 13  Size_in_Mb
0  4.0.3 and up          NaN        19.0
1  4.0.3 and up          NaN        14.0
2  4.0.3 and up          NaN         8.7
3    4.2 and up          NaN        25.0
4    4.4 and up          NaN         2.8
```

[23]:
```
df['Installs'].dtype # data type of the column
```

[23]:
```
dtype('O')
```

[24]:
```
df['Installs'].max() # this will show the value counts of the column
```

[24]:
```
'500,000,000+'
```

[25]:
```
# check the unique values in the 'Price' column
df['Price'].unique()
```

[25]:
```
array(['0', '$4.99', '$3.99', '$6.99', '$1.49', '$2.99', '$7.99', '$5.99',
       '$3.49', '$1.99', '$9.99', '$7.49', '$0.99', '$9.00', '$5.49',
       '$10.00', '$24.99', '$11.99', '$79.99', '$16.99', '$14.99',
       '$1.00', '$29.99', '$12.99', '$2.49', '$10.99', '$1.50', '$19.99',
       '$15.99', '$33.99', '$74.99', '$39.99', '$3.95', '$4.49', '$1.70',
       '$8.99', '$2.00', '$3.88', '$25.99', '$399.99', '$17.99',
       '$400.00', '$3.02', '$1.76', '$4.84', '$4.77', '$1.61', '$2.50',
       '$1.59', '$6.49', '$1.29', '$5.00', '$13.99', '$299.99', '$379.99',
       '$37.99', '$18.99', '$389.99', '$19.90', '$8.49', '$1.75',
       '$14.00', '$4.85', '$46.99', '$109.99', '$154.99', '$3.08',
       '$2.59', '$4.80', '$1.96', '$19.40', '$3.90', '$4.59', '$15.46',
       '$3.04', '$4.29', '$2.60', '$3.28', '$4.60', '$28.99', '$2.95',
       '$2.90', '$1.97', '$200.00', '$89.99', '$2.56', '$30.99', '$3.61',
       '$394.99', '$1.26', '$1.20', '$1.04'], dtype=object)
```

[26]:
```
df['Price'].isnull().sum()
```

[26]:
```
0
```

- No Null Values

[27]:
```
df['Price'].value_counts() # check the value counts of the 'Price' column
```

[27]:
```
0          10041
$0.99        148
$2.99        129
$1.99         73
$4.99         72
           ...
$19.90         1
$1.75          1
$14.00         1
$4.85          1
$1.04          1
Name: Price, Length: 92, dtype: int64
```

[28]:
```
# count the values having $ in the 'Price' column
df['Price'].loc[df['Price'].str.contains('\$')].value_counts().sum()
```

[28]:
```
800
```

[29]:
```
# This code counts the number of values in the 'Price' column which contains 0␣
 ↪but does not contain $ sign
df['Price'].loc[(df['Price'].str.contains('0')) & (~df['Price'].str.
 ↪contains('\$'))].value_counts().sum()
```

[29]:
```
10041
```

[30]:
```
# remove the dollar sign from the price column and convert it to numeric
df['Price'] = df['Price'].apply(lambda x: x.replace('$', '') if '$' in str(x)␣
 ↪else x)
# convert the price column to numeric (float because this is the price)
df['Price'] = df['Price'].apply(lambda x: float(x))
```

[31]:
```
df['Price'].dtype # this will show the data type of the column
```

[31]:
```
dtype('float64')
```

[32]:
```
# using f string to print the min, max and average price of the apps
print(f"Min price is: {df['Price'].min()} $")
print(f"Max price is: {df['Price'].max()} $")
print(f"Average price is: {df['Price'].mean()} $")
```

```
Min price is: 0.0 $
Max price is: 400.0 $
```

```
Average price is: 1.0272733142699015 $
```

2.0.1 Statistics

[33]:
```
df.describe()
```

[33]:
```
              Rating       Reviews  Size_in_bytes         Price  Unnamed: 13  \
count  9367.000000  1.084100e+04   9.146000e+03  10841.000000          0.0
mean      4.191513  4.441119e+05   2.255921e+07      1.027273          NaN
std       0.515735  2.927629e+06   2.368595e+07     15.948971          NaN
min       1.000000  0.000000e+00   8.704000e+03      0.000000          NaN
25%       4.000000  3.800000e+01   5.138022e+06      0.000000          NaN
50%       4.300000  2.094000e+03   1.363149e+07      0.000000          NaN
75%       4.500000  5.476800e+04   3.145728e+07      0.000000          NaN
max       5.000000  7.815831e+07   1.048576e+08    400.000000          NaN

        Size_in_Mb
count  9146.000000
mean     21.514141
std      22.588679
min       0.008301
25%       4.900000
50%      13.000000
75%      30.000000
max     100.000000
```

3 2.2. Dealing with the missing values

- Let's have a look on the missing values in the dataset

[34]:
```
df.isnull().sum() # this will show the number of null values in each column
```

[34]:
```
App                 1
Category            0
Rating           1474
Reviews             0
Size_in_bytes    1695
Installs            0
Type                1
Price               0
Content Rating      0
Genres              1
Last Updated        0
Current Ver         8
Android Ver         2
Unnamed: 13     10841
Size_in_Mb       1695
```

```
dtype: int64
```

[35]:
```
df.isnull().sum().sort_values(ascending=False) # this will show the number of␣
↪null values in each column in descending order
```

[35]:
```
Unnamed: 13       10841
Size_in_bytes      1695
Size_in_Mb         1695
Rating             1474
Current Ver           8
Android Ver           2
App                   1
Type                  1
Genres                1
Category              0
Reviews               0
Installs              0
Price                 0
Content Rating        0
Last Updated          0
dtype: int64
```

[36]:
```
df.isnull().sum().sum() # this will show the total number of null values in the␣
↪dataframe
```

[36]:
```
15718
```

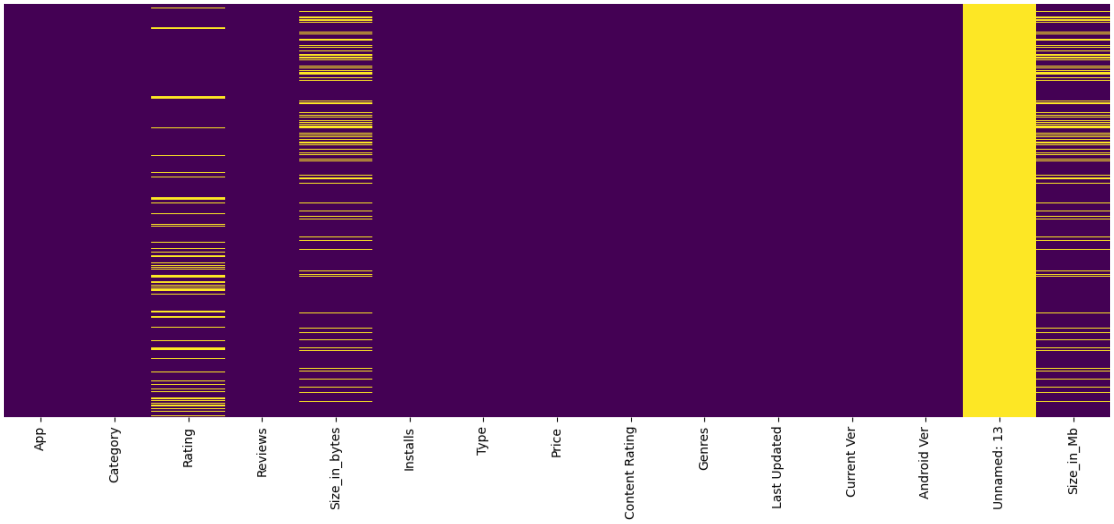
[37]:
```
(df.isnull().sum() / len(df) * 100).sort_values(ascending=False) # this will␣
↪show the percentage of null values in each column
```

[37]:
```
Unnamed: 13       100.000000
Size_in_bytes      15.635089
Size_in_Mb         15.635089
Rating             13.596532
Current Ver         0.073794
Android Ver         0.018448
App                 0.009224
Type                0.009224
Genres              0.009224
Category            0.000000
Reviews             0.000000
Installs            0.000000
Price               0.000000
Content Rating      0.000000
Last Updated        0.000000
dtype: float64
```

- Let's plot the missing values in the dataset

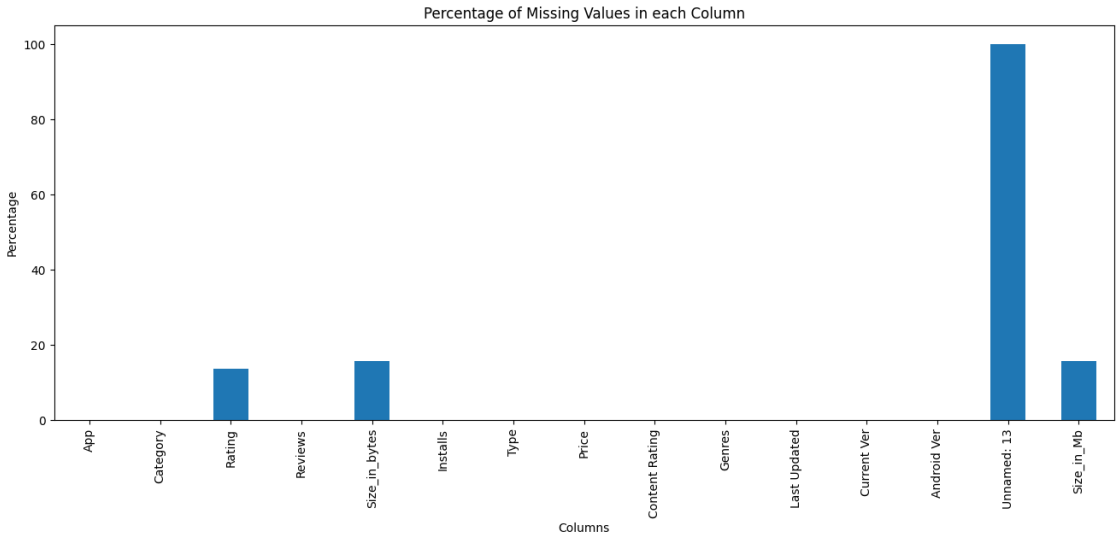
[38]:
```
# make a figure size
plt.figure(figsize=(16, 6))
#plot the null values in each column
sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis') # this␣
 ↪will show the heatmap of null values in the dataframe
```

[38]: <Axes: >



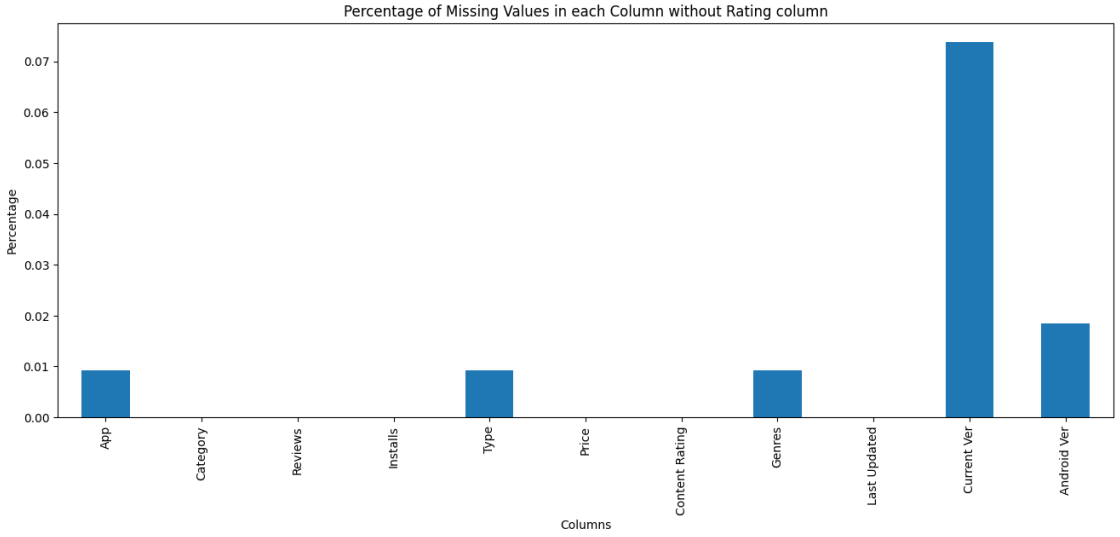
[39]:
```
# make figure size
plt.figure(figsize=(16, 6))
# plot the null values by their percentage in each column
missing_percentage = df.isnull().sum()/len(df)*100
missing_percentage.plot(kind='bar')
# add the labels
plt.xlabel('Columns')
plt.ylabel('Percentage')
plt.title('Percentage of Missing Values in each Column')
```

[39]: Text(0.5, 1.0, 'Percentage of Missing Values in each Column')



[40]:
```
plt.figure(figsize=(16, 6)) # make figure size
missing_percentage[missing_percentage < 1].plot(kind='bar') # plot the null␣
 ↪values by their percentage in each column
plt.xlabel('Columns') # add the x-axis labels
plt.ylabel('Percentage') # add the labels for y-axis
plt.title('Percentage of Missing Values in each Column without Rating column') ␣
 ↪# add the title for the plot
```

[40]:
```
Text(0.5, 1.0, 'Percentage of Missing Values in each Column without Rating
column')
```



[41]:
```
df.isnull().sum().sort_values(ascending=False) # this will show the number of␣
↪null values in each column in descending order
```

[41]:
```
Unnamed: 13       10841
Size_in_bytes      1695
Size_in_Mb         1695
Rating             1474
Current Ver           8
Android Ver           2
App                   1
Type                  1
Genres                1
Category              0
Reviews               0
Installs              0
Price                 0
Content Rating        0
Last Updated          0
dtype: int64
```

[42]:
```
(df.isnull().sum() / len(df) * 100).sort_values(ascending=False) # this will␣
↪show the percentage of null values in each column
```

[42]:
```
Unnamed: 13       100.000000
Size_in_bytes      15.635089
Size_in_Mb         15.635089
Rating             13.596532
Current Ver         0.073794
Android Ver         0.018448
App                 0.009224
Type                0.009224
Genres              0.009224
Category            0.000000
Reviews             0.000000
Installs            0.000000
Price               0.000000
Content Rating      0.000000
Last Updated        0.000000
dtype: float64
```

- Let's run the correlations

[43]:
```
df.describe() # these are numeric columns
```

[43]:
```
              Rating       Reviews  Size_in_bytes         Price  Unnamed: 13  \
count  9367.000000  1.084100e+04   9.146000e+03  10841.000000          0.0
mean      4.191513  4.441119e+05   2.255921e+07      1.027273          NaN
std       0.515735  2.927629e+06   2.368595e+07     15.948971          NaN
```

```
min       1.000000  0.000000e+00   8.704000e+03      0.000000          NaN
25%       4.000000  3.800000e+01   5.138022e+06      0.000000          NaN
50%       4.300000  2.094000e+03   1.363149e+07      0.000000          NaN
75%       4.500000  5.476800e+04   3.145728e+07      0.000000          NaN
max       5.000000  7.815831e+07   1.048576e+08    400.000000          NaN

        Size_in_Mb
count  9146.000000
mean     21.514141
std      22.588679
min       0.008301
25%       4.900000
50%      13.000000
75%      30.000000
max     100.000000
```

[44]:
```
# Make a correlation matrix of numeric columns
plt.figure(figsize=(16, 10)) # make figure size
numeric_cols = ['Rating', 'Reviews', 'Size_in_bytes', 'Installs', 'Price',␣
 ↪'Size_in_Mb'] # make a list of numeric columns
sns.heatmap(df[numeric_cols].corr(), annot=True) # plot the correlation matrix
```

```
/tmp/ipykernel_493/2767317481.py:4: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(df[numeric_cols].corr(), annot=True) # plot the correlation matrix
```

[44]: <Axes: >

[45]:
```
# we can also calculate the correlation matrix using pandas
df[numeric_cols].corr() # this will show the correlation matrix
```

```
/tmp/ipykernel_493/3440762686.py:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  df[numeric_cols].corr() # this will show the correlation matrix
```

[45]:

	Rating	Reviews	Size_in_bytes	Price	Size_in_Mb
Rating	1.000000	0.068147	0.084098	-0.021851	0.084098
Reviews	0.068147	1.000000	0.238218	-0.009666	0.238218
Size_in_bytes	0.084098	0.238218	1.000000	-0.023000	1.000000
Price	-0.021851	-0.009666	-0.023000	1.000000	-0.023000
Size_in_Mb	0.084098	0.238218	1.000000	-0.023000	1.000000

[46]:
```
# length before removing null values
print(f"Length of the dataframe before removing null values: {len(df)}")
```

```
Length of the dataframe before removing null values: 10841
```

[47]:
```
# remove the rows having null values in the 'Current Ver', 'Android Ver',␣
 ↪'Category', 'Type' and 'Genres' column
df.dropna(subset=['Current Ver', 'Android Ver', 'Category', 'Type', 'Genres'],␣
 ↪inplace=True)
```

[48]:
```
# length after removing null values
print(f"Length of the dataframe after removing null values: {len(df)}")
```

```
Length of the dataframe after removing null values: 10829
```

[49]:
```
# let's check the null values again
df.isnull().sum().sort_values(ascending=False)
```

[49]:
```
Unnamed: 13       10829
Size_in_bytes      1694
Size_in_Mb         1694
Rating             1469
App                   0
Category              0
Reviews               0
Installs              0
Type                  0
Price                 0
Content Rating        0
Genres                0
Last Updated          0
Current Ver           0
Android Ver           0
dtype: int64
```

[50]:
```
df.columns
```

[50]:
```
Index(['App', 'Category', 'Rating', 'Reviews', 'Size_in_bytes', 'Installs',
       'Type', 'Price', 'Content Rating', 'Genres', 'Last Updated',
       'Current Ver', 'Android Ver', 'Unnamed: 13', 'Size_in_Mb'],
      dtype='object')
```

[52]:
```
#find the trend of Rating in each Installs_category
#df.groupby('Installs_category')['Rating'].describe()
```

[57]:
```
df['Rating'].isnull().sum()
```

[57]:
```
1469
```

[53]:
```
# in which Install_category the Rating has NaN values
#df['Installs_category'].loc[df['Rating'].isnull()].value_counts()
```

- Let's plot this and have a look

- Let's check if there is any similar link with Reviews as well

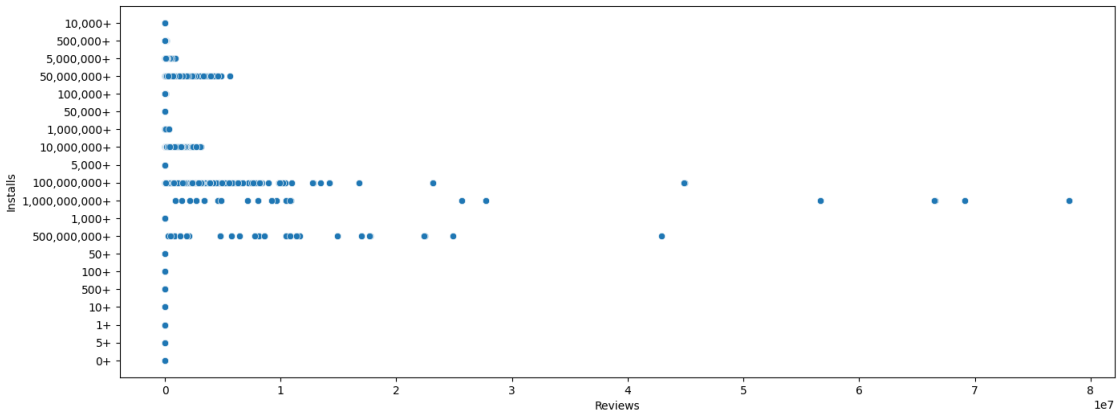
[ ]:
```
# in which Install_category the Rating has NaN values
df['Installs_category'].loc[df['Reviews'].isnull()].value_counts()
```

[ ]:
```
Installs_category
no                      0
Very low                0
Low                     0
Moderate                0
More than moderate      0
High                    0
Very High               0
Top Notch               0
Name: count, dtype: int64
```

- There are no Null values in Reviews
- We also draw the scatter plot of the `Rating` and `Review` columns with the `Installs` column

[62]:
```
# plot reviews and installs in a scatter plot
plt.figure(figsize=(16, 6)) # make figure size
sns.scatterplot(x='Reviews', y='Installs', data=df) # plot the scatter plot
```

[62]:
```
<Axes: xlabel='Reviews', ylabel='Installs'>
```



3.1 2.3. Duplicates

- Removing duplicates is one of the most important part of the data wrangling process, we must remove the duplicates in order to get the correct insights from the data.
- If you do not remove duplicates from a dataset, it can lead to incorrect insights and analysis.
- Duplicates can skew statistical measures such as mean, median, and standard deviation, and can also lead to over-representation of certain data points.

- It is important to remove duplicates to ensure the accuracy and reliability of your data analysis.

[ ]:
```
# find duplicate if any
df.duplicated().sum()
```

[ ]:
```
483
```

This shows us total duplicates, but we can also check based on the app name, as we know that every app has a unique name.

[ ]:
```
# find duplicate if any in the 'App' column
df['App'].duplicated().sum()
```

[ ]:
```
1181
```

[65]:
```
# let's check for number of duplicates
for col in df.columns:
    print(f"Number of duplicates in {col} column are: {df[col].duplicated().
 ↪sum()}")
```

```
Number of duplicates in App column are: 1181
Number of duplicates in Category column are: 10796
Number of duplicates in Rating column are: 10789
Number of duplicates in Reviews column are: 4830
Number of duplicates in Size_in_bytes column are: 10373
Number of duplicates in Installs column are: 10809
Number of duplicates in Type column are: 10827
Number of duplicates in Price column are: 10737
Number of duplicates in Content Rating column are: 10823
Number of duplicates in Genres column are: 10710
Number of duplicates in Last Updated column are: 9453
Number of duplicates in Current Ver column are: 7998
Number of duplicates in Android Ver column are: 10796
Number of duplicates in Unnamed: 13 column are: 10828
Number of duplicates in Size_in_Mb column are: 10373
```

[66]:
```
# print the number of duplicates in df
print(f"Number of duplicates in df are: {df.duplicated().sum()}")
```

```
Number of duplicates in df are: 483
```

[67]:
```
# find exact duplicates and print them
df[df['App'].duplicated(keep=False)].sort_values(by='App')
```

[67]:

	App	Category	Rating	Reviews	\
1393	10 Best Foods for You	HEALTH_AND_FITNESS	4.0	2490	
1407	10 Best Foods for You	HEALTH_AND_FITNESS	4.0	2490	

```
2543      1800 Contacts - Lens Store             MEDICAL     4.7    23160
2322      1800 Contacts - Lens Store             MEDICAL     4.7    23160
2385      2017 EMRA Antibiotic Guide             MEDICAL     4.4       12
…                                 …                 …     …      …
3202        trivago: Hotels & Travel    TRAVEL_AND_LOCAL     4.2   219848
3118        trivago: Hotels & Travel    TRAVEL_AND_LOCAL     4.2   219848
3103        trivago: Hotels & Travel    TRAVEL_AND_LOCAL     4.2   219848
8291  wetter.com - Weather and Radar             WEATHER     4.2   189310
3652  wetter.com - Weather and Radar             WEATHER     4.2   189313

      Size_in_bytes     Installs  Type  Price Content Rating  \
1393     3984588.8     500,000+  Free   0.00   Everyone 10+
1407     3984588.8     500,000+  Free   0.00   Everyone 10+
2543    27262976.0   1,000,000+  Free   0.00       Everyone
2322    27262976.0   1,000,000+  Free   0.00       Everyone
2385     3984588.8       1,000+  Paid  16.99       Everyone
…              …           …   …    …             …
3202           NaN  50,000,000+  Free   0.00       Everyone
3118           NaN  50,000,000+  Free   0.00       Everyone
3103           NaN  50,000,000+  Free   0.00       Everyone
8291    39845888.0  10,000,000+  Free   0.00       Everyone
3652    39845888.0  10,000,000+  Free   0.00       Everyone

               Genres       Last Updated         Current Ver  \
1393  Health & Fitness  February 17, 2017               1.9
1407  Health & Fitness  February 17, 2017               1.9
2543           Medical      July 27, 2018             7.4.1
2322           Medical      July 27, 2018             7.4.1
2385           Medical   January 27, 2017             1.0.5
…                …                …                  …
3202    Travel & Local     August 2, 2018  Varies with device
3118    Travel & Local     August 2, 2018  Varies with device
3103    Travel & Local     August 2, 2018  Varies with device
8291           Weather     August 6, 2018  Varies with device
3652           Weather     August 6, 2018  Varies with device

            Android Ver  Unnamed: 13  Size_in_Mb
1393       2.3.3 and up          NaN         3.8
1407       2.3.3 and up          NaN         3.8
2543         5.0 and up          NaN        26.0
2322         5.0 and up          NaN        26.0
2385       4.0.3 and up          NaN         3.8
…                  …           …          …
3202  Varies with device          NaN         NaN
3118  Varies with device          NaN         NaN
3103  Varies with device          NaN         NaN
8291  Varies with device          NaN        38.0
```

```
3652  Varies with device          NaN        38.0

[1979 rows x 15 columns]
```

- Remove Duplicates

[68]:
```
# remove the duplicates
df.drop_duplicates(inplace=True)
```

[69]:
```
# print the number of rows and columns after removing duplicates
print(f"Number of rows after removing duplicates: {df.shape[0]}")
```

```
Number of rows after removing duplicates: 10346
```

- Now we have removed 483 duplicates from the dataset. and have 10346 rows left.

3.2 Which category has the highest number of apps?

[70]:
```
# which category has highest number of apps
df['Category'].value_counts().head(10) # this will show the top 10 categories␣
↪with highest number of apps
```

[70]:
```
FAMILY             1939
GAME               1121
TOOLS               841
BUSINESS            427
MEDICAL             408
PRODUCTIVITY        407
PERSONALIZATION     386
LIFESTYLE           373
COMMUNICATION       366
FINANCE             360
Name: Category, dtype: int64
```

4 Which category has the highest number of installs?

[ ]:
```
# category with highest number of Installs
df.groupby('Category')['Installs'].sum().sort_values(ascending=False).head(10)
```

[ ]:
```
Category
GAME                 31544024415
COMMUNICATION        24152276251
SOCIAL               12513867902
PRODUCTIVITY         12463091369
TOOLS                11452271905
```

```
FAMILY                10041632405
PHOTOGRAPHY            9721247655
TRAVEL_AND_LOCAL       6361887146
VIDEO_PLAYERS          6222002720
NEWS_AND_MAGAZINES     5393217760
Name: Installs, dtype: int64
```

5 Which category has the highest number of reviews

[ ]:
```
# Category with highest number of Reviews
df.groupby('Category')['Reviews'].sum().sort_values(ascending=False).head(10)
```

[ ]:
```
Category
GAME                1415536650
COMMUNICATION        601273552
SOCIAL               533576829
FAMILY               396771746
TOOLS                273181033
PHOTOGRAPHY          204297410
VIDEO_PLAYERS        110380188
PRODUCTIVITY         102554498
SHOPPING              94931162
PERSONALIZATION       75192744
Name: Reviews, dtype: int64
```

6 Which category has the highest rating?

[ ]:
```
# Category with highest average Rating
df.groupby('Category')['Rating'].mean().sort_values(ascending=False).head(10)
```

[ ]:
```
Category
EVENTS                4.435556
ART_AND_DESIGN        4.377049
EDUCATION             4.375969
BOOKS_AND_REFERENCE   4.347458
PERSONALIZATION       4.333117
PARENTING             4.300000
GAME                  4.281285
BEAUTY                4.278571
HEALTH_AND_FITNESS    4.261450
SOCIAL                4.254918
Name: Rating, dtype: float64
```

[71]:
```
# plot the rating distribution
plt.figure(figsize=(16, 6)) # make figure size
```

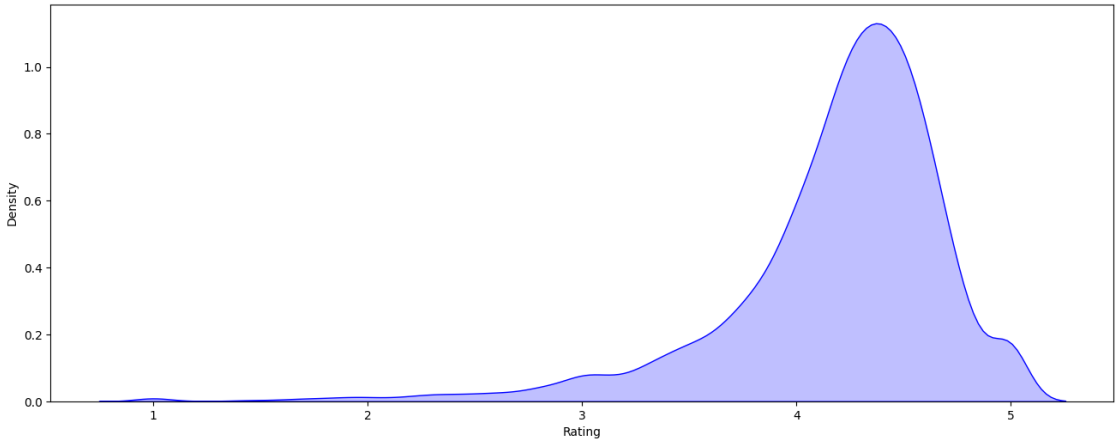
```
sns.kdeplot(df['Rating'], color="blue", shade=True) # plot the distribution plot
```

```
/tmp/ipykernel_884/2564065887.py:3: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df['Rating'], color="blue", shade=True) # plot the distribution
plot
```

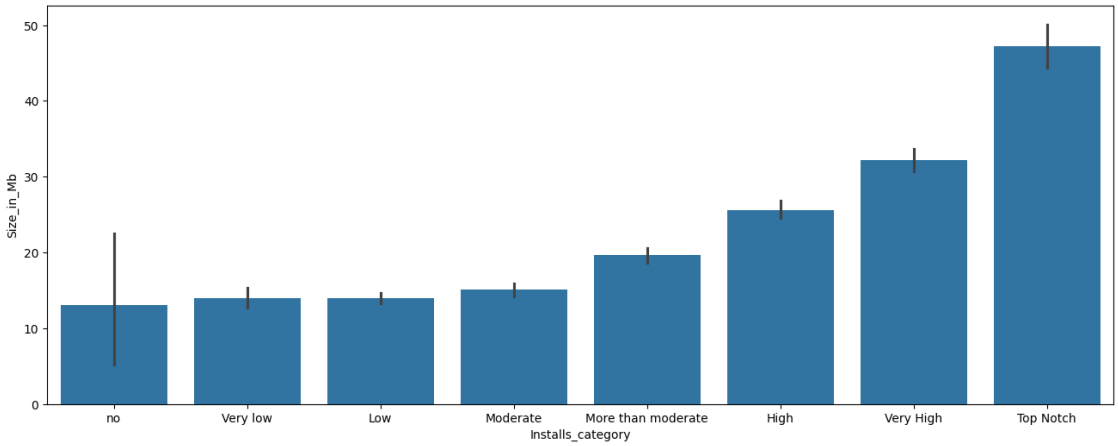
[71]: <Axes: xlabel='Rating', ylabel='Density'>



7 Examples

[ ]:
```
# Check if there is any impact of size on installs
# make a bar plot of Size_in_Mb vs Installs_category
plt.figure(figsize=(16, 6)) # make figure size
sns.barplot(x='Installs_category', y='Size_in_Mb', data=df) # plot the bar plot
```

[ ]: <Axes: xlabel='Installs_category', ylabel='Size_in_Mb'>

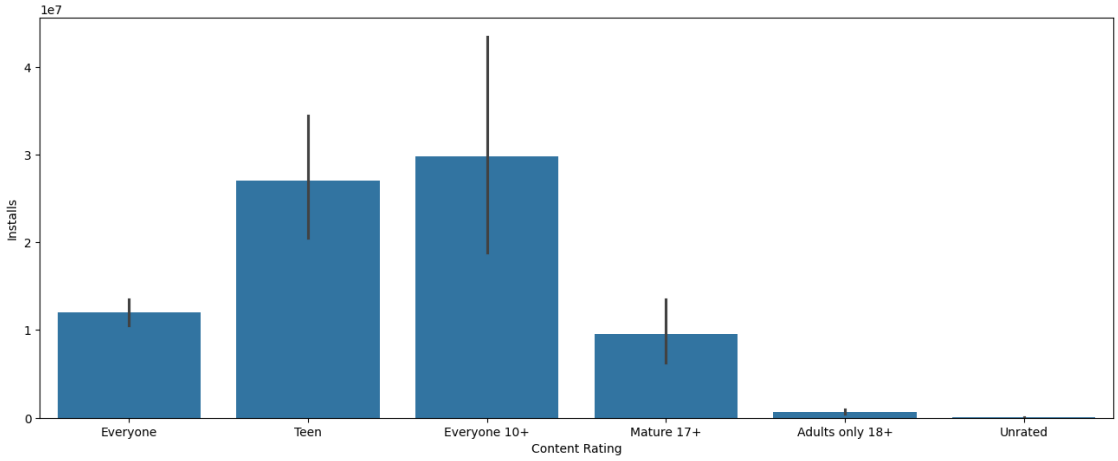


[ ]:
```
# Which content rating is most popular in installs
df['Content Rating'].value_counts() # this will show the value counts of each␣
↪content rating
```

[ ]:
```
Content Rating
Everyone           8372
Teen               1146
Mature 17+          447
Everyone 10+        376
Adults only 18+       3
Unrated               2
Name: count, dtype: int64
```

[ ]:
```
# plot the bar plot of Content Rating vs Installs
plt.figure(figsize=(16, 6)) # make figure size
sns.barplot(x='Content Rating', y='Installs', data=df) # plot the bar plot
```

[ ]:
```
<Axes: xlabel='Content Rating', ylabel='Installs'>
```

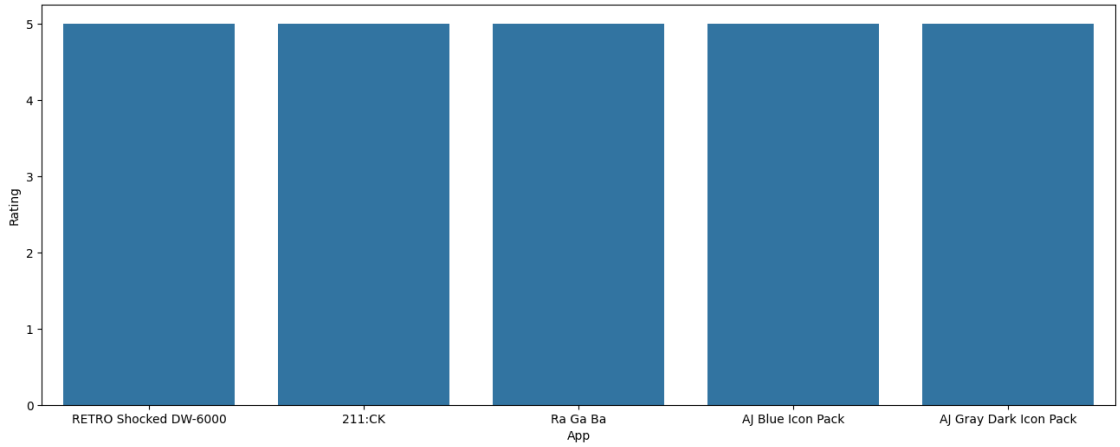


[ ]:
```
# find how many apps are there in Everyone content rating
df['Category'].loc[df['Content Rating'] == 'Everyone'].value_counts()
```

[ ]:
```
Category
FAMILY                 1498
TOOLS                   833
GAME                    595
BUSINESS                412
PRODUCTIVITY            396
MEDICAL                 390
FINANCE                 355
LIFESTYLE               337
COMMUNICATION           325
PERSONALIZATION         318
SPORTS                  318
PHOTOGRAPHY             307
HEALTH_AND_FITNESS      275
TRAVEL_AND_LOCAL        230
BOOKS_AND_REFERENCE     200
SHOPPING                183
NEWS_AND_MAGAZINES      168
VIDEO_PLAYERS           146
MAPS_AND_NAVIGATION     133
EDUCATION               121
FOOD_AND_DRINK          114
SOCIAL                   98
LIBRARIES_AND_DEMO       84
AUTO_AND_VEHICLES        83
HOUSE_AND_HOME           78
WEATHER                  78
ART_AND_DESIGN           60
PARENTING                58
EVENTS                   53
BEAUTY                   45
ENTERTAINMENT            37
COMICS                   26
DATING                   18
Name: count, dtype: int64
```

[ ]:
```
# plot top 5 rated paid apps
plt.figure(figsize=(16, 6)) # make figure size
sns.barplot(x='App', y='Rating', data=df[df['Type'] == 'Paid'].
 ↪sort_values(by='Rating', ascending=False).head(5)) # plot the bar plot
```

[ ]:
```
<Axes: xlabel='App', ylabel='Rating'>
```



[ ]:
```
df[df['Type'] == 'Paid'].sort_values(by='Rating', ascending=False).head(5)
```

[ ]:

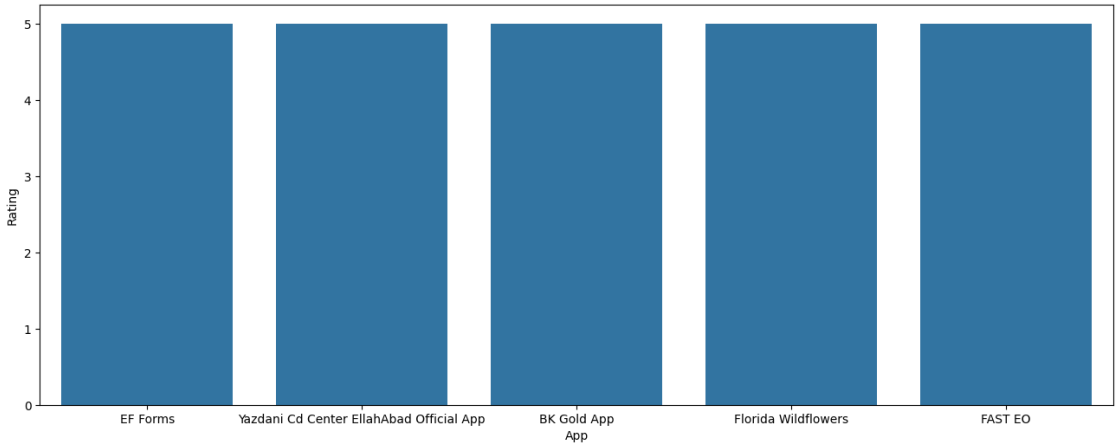
	App	Category	Rating	Reviews	Size_in_bytes	\
9010	RETRO Shocked DW-6000	PERSONALIZATION	5.0	13	512000.0	
7466	211:CK	GAME	5.0	8	39845888.0	
5917	Ra Ga Ba	GAME	5.0	2	20971520.0	
5263	AJ Blue Icon Pack	PERSONALIZATION	5.0	4	32505856.0	
5260	AJ Gray Dark Icon Pack	PERSONALIZATION	5.0	2	36700160.0	

	Installs	Type	Price	Content Rating	Genres	Last Updated	\
9010	100	Paid	1.49	Everyone	Personalization	April 4, 2017	
7466	10	Paid	0.99	Teen	Arcade	April 11, 2018	
5917	1	Paid	1.49	Everyone	Arcade	February 8, 2017	
5263	50	Paid	0.99	Everyone	Personalization	April 27, 2018	
5260	10	Paid	0.99	Everyone	Personalization	April 29, 2018	

	Current Ver	Android Ver	Size_in_Mb	Installs_category
9010	1.2	2.3 and up	0.488281	Low
7466	1.3	4.1 and up	38.000000	Very low
5917	1.0.4	2.3 and up	20.000000	Very low
5263	1.1	4.1 and up	31.000000	Low
5260	1.1	4.1 and up	35.000000	Very low

[ ]:
```
# plot top rated 5 apps in free category
plt.figure(figsize=(16, 6)) # make figure size
sns.barplot(x='App', y='Rating', data=df[df['Type'] == 'Free'].
 ↪sort_values(by='Rating', ascending=False).head(5)) # plot the bar plot
```

[ ]:
```
<Axes: xlabel='App', ylabel='Rating'>
```



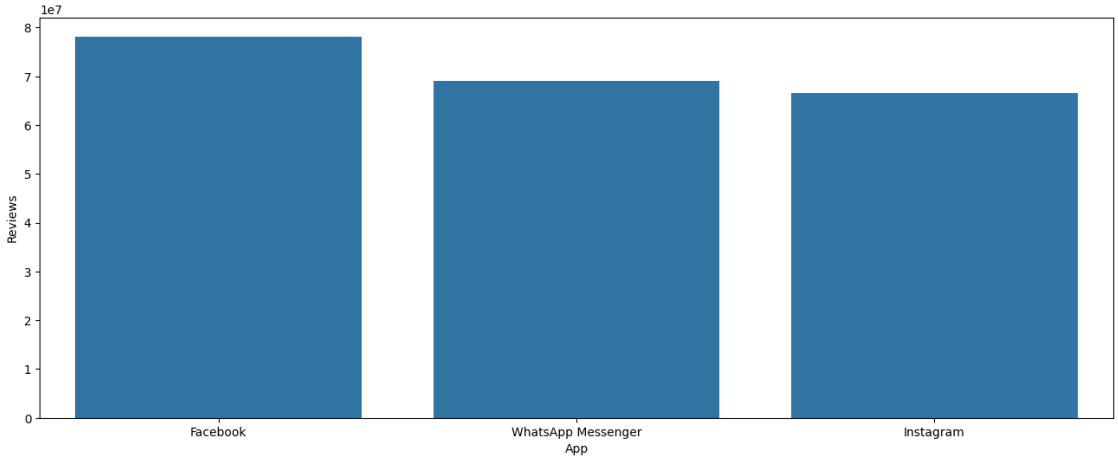
[ ]:
```
df[df['Type'] == 'Free'].sort_values(by='Rating', ascending=False).head(5)
```

[ ]:

	App	Category	Rating	Reviews	Size_in_bytes	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver	Size_in_Mb	Installs_category
9290	EF Forms	BUSINESS	5.0	2	24117248.0	50	Free	0.0	Everyone	Business	July 24, 2018	1.29	4.4 and up	23.0	Low
7170	Yazdani Cd Center EllahAbad Official App	FAMILY	5.0	8	3984588.8	500	Free	0.0	Everyone	Entertainment	January 12, 2018	2.0	4.0 and up	3.8	Low
6398	BK Gold App	FINANCE	5.0	4	11534336.0	50	Free	0.0	Everyone	Finance	May 25, 2018	1.0.0	4.4 and up	11.0	Low
10629	Florida Wildflowers	FAMILY	5.0	5	72351744.0	1000	Free	0.0	Everyone	Education	July 10, 2017	1.5	4.1 and up	69.0	Low
9659	FAST EO	EVENTS	5.0	1	NaN	10	Free	0.0	Everyone	Events	May 15, 2018	1.0.3	4.1 and up	NaN	Very low

[ ]:
```
# Plot top 5 FREE apps with highest number of reviews
plt.figure(figsize=(16, 6)) # make figure size
sns.barplot(x='App', y='Reviews', data=df[df['Type'] == 'Free'].
 ↪sort_values(by='Reviews', ascending=False).head(5)) # plot the bar plot
```

[ ]: <Axes: xlabel='App', ylabel='Reviews'>



[ ]:
```
df[df['Type'] == 'Free'].sort_values(by='Reviews', ascending=False).head(5)
```

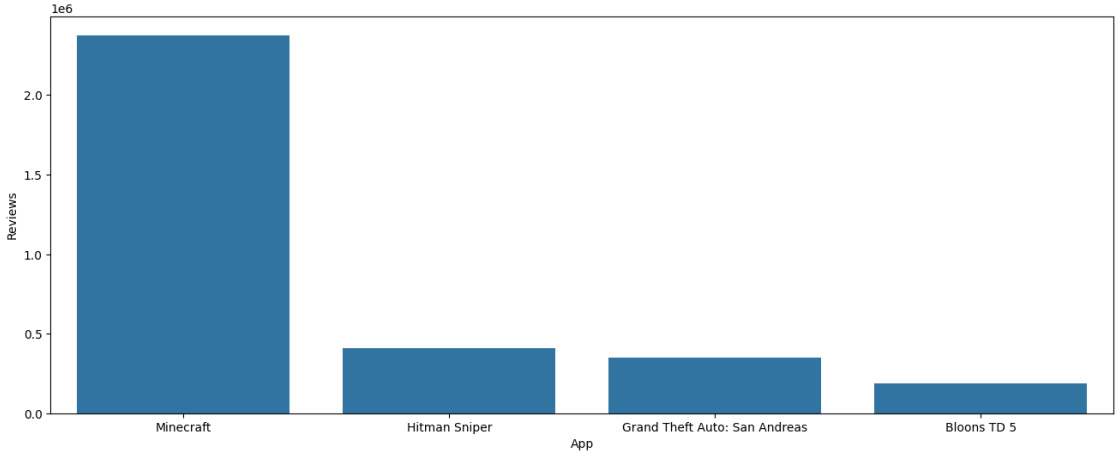
[ ]:
```
                    App       Category  Rating   Reviews  Size_in_bytes  \
2544           Facebook         SOCIAL     4.1  78158306            NaN
3943           Facebook         SOCIAL     4.1  78128208            NaN
336   WhatsApp Messenger  COMMUNICATION     4.4  69119316            NaN
3904  WhatsApp Messenger  COMMUNICATION     4.4  69109672            NaN
2604          Instagram         SOCIAL     4.5  66577446            NaN

        Installs  Type  Price Content Rating         Genres    Last Updated  \
2544  1000000000  Free    0.0           Teen         Social  August 3, 2018
3943  1000000000  Free    0.0           Teen         Social  August 3, 2018
336   1000000000  Free    0.0       Everyone  Communication  August 3, 2018
3904  1000000000  Free    0.0       Everyone  Communication  August 3, 2018
2604  1000000000  Free    0.0           Teen         Social   July 31, 2018

             Current Ver         Android Ver  Size_in_Mb Installs_category
2544  Varies with device  Varies with device         NaN         Top Notch
3943  Varies with device  Varies with device         NaN         Top Notch
336   Varies with device  Varies with device         NaN         Top Notch
3904  Varies with device  Varies with device         NaN         Top Notch
2604  Varies with device  Varies with device         NaN         Top Notch
```

[ ]:
```
# Plot top 5 Paid apps with highest number of reviews
plt.figure(figsize=(16, 6)) # make figure size
sns.barplot(x='App', y='Reviews', data=df[df['Type'] == 'Paid'].
 ↪sort_values(by='Reviews', ascending=False).head(5)) # plot the bar plot
```

[ ]:
```
<Axes: xlabel='App', ylabel='Reviews'>
```


1e6
2.0
1.5
1.0
0.5
0.0
Reviews
Minecraft
Hitman Sniper
Grand Theft Auto: San Andreas
Bloons TD 5
App