In [1]:

```
#importing all necessary libraries
```

In [2]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```
#reading the dataset
df = pd.read_csv('E:\Datascience\EDA my present class\EDA Dataset\googleplaystore
```

In [4]:

```
#first five rows of dataset
df.head()
```

Out[4]:

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

In [5]:
```
#finding the last five rows of dataset
df.tail()
```

Out[5]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	
10836	Sya9a Maroc - FR	FAMILY	4.5	38	53M	5,000+	Free	0	E
10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3.6M	100+	Free	0	E
10838	Parkinson Exercices FR	MEDICAL	NaN	3	9.5M	1,000+	Free	0	E
10839	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5	114	Varies with device	1,000+	Free	0	
10840	iHoroscope - 2018 Daily Horoscope & Astrology	LIFESTYLE	4.5	398307	19M	10,000,000+	Free	0	E

In [6]:
```
#finding the information about dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10841 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             10841 non-null  object 
 1   Category        10841 non-null  object 
 2   Rating          9367 non-null   float64
 3   Reviews         10841 non-null  object 
 4   Size            10841 non-null  object 
 5   Installs        10841 non-null  object 
 6   Type            10840 non-null  object 
 7   Price           10841 non-null  object 
 8   Content Rating  10840 non-null  object 
 9   Genres          10841 non-null  object 
 10  Last Updated    10841 non-null  object 
 11  Current Ver     10833 non-null  object 
 12  Android Ver     10838 non-null  object 
dtypes: float64(1), object(12)
memory usage: 1.1+ MB
```

In [7]:
```
#finding the statistical analysis of numerical columns in dataset
df.describe().T
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
Rating	9367.0	4.193338	0.537431	1.0	4.0	4.3	4.5	19.0

In [8]:
```
#finding the columns names from dataset
df.columns
```

Out[8]:
```
Index(['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type',
       'Price', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver',
       'Android Ver'],
      dtype='object')
```

In [9]:
```
#finding the null values
df.isnull().sum()
```

Out[9]:
```
App                  0
Category             0
Rating            1474
Reviews              0
Size                 0
Installs             0
Type                 1
Price                0
Content Rating       1
Genres               0
Last Updated         0
Current Ver          8
Android Ver          3
dtype: int64
```

Now doing preprocessing of the data step by step

In [10]:
```
#finding the unique values in App column
df['App'].unique()
```

Out[10]:
```
array(['Photo Editor & Candy Camera & Grid & ScrapBook',
       'Coloring book moana',
       'U Launcher Lite – FREE Live Cool Themes, Hide Apps', ...,
       'Parkinson Exercices FR', 'The SCP Foundation DB fr nn5n',
       'iHoroscope - 2018 Daily Horoscope & Astrology'], dtype=object)
```

In [11]:
```
#finding the unique values in Category column
df['Category'].unique()
```

Out[11]:
```
array(['ART_AND_DESIGN', 'AUTO_AND_VEHICLES', 'BEAUTY',
       'BOOKS_AND_REFERENCE', 'BUSINESS', 'COMICS', 'COMMUNICATION',
       'DATING', 'EDUCATION', 'ENTERTAINMENT', 'EVENTS', 'FINANCE',
       'FOOD_AND_DRINK', 'HEALTH_AND_FITNESS', 'HOUSE_AND_HOME',
       'LIBRARIES_AND_DEMO', 'LIFESTYLE', 'GAME', 'FAMILY', 'MEDICAL',
       'SOCIAL', 'SHOPPING', 'PHOTOGRAPHY', 'SPORTS', 'TRAVEL_AND_LOCAL',
       'TOOLS', 'PERSONALIZATION', 'PRODUCTIVITY', 'PARENTING', 'WEATHER',
       'VIDEO_PLAYERS', 'NEWS_AND_MAGAZINES', 'MAPS_AND_NAVIGATION',
       '1.9'], dtype=object)
```

In [13]:
```
#finding the unique values in Ratings column
df['Rating'].unique()
```

Out[13]:
```
array([ 4.1,  3.9,  4.7,  4.5,  4.3,  4.4,  3.8,  4.2,  4.6,  3.2,  4. ,
        nan,  4.8,  4.9,  3.6,  3.7,  3.3,  3.4,  3.5,  3.1,  5. ,  2.6,
        3. ,  1.9,  2.5,  2.8,  2.7,  1. ,  2.9,  2.3,  2.2,  1.7,  2. ,
        1.8,  2.4,  1.6,  2.1,  1.4,  1.5,  1.2, 19. ])
```

In [14]:
```
#describing the dataset including all columns
df.describe(include='all')
```

Out[14]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Ge
count	10841	10841	9367.000000	10841	10841	10841	10840	10841	10840	1
unique	9660	34	NaN	6002	462	22	3	93	6	
top	ROBLOX	FAMILY	NaN	0	Varies with device	1,000,000+	Free	0	Everyone	
freq	9	1972	NaN	596	1695	1579	10039	10040	8714	
mean	NaN	NaN	4.193338	NaN	NaN	NaN	NaN	NaN	NaN	
std	NaN	NaN	0.537431	NaN	NaN	NaN	NaN	NaN	NaN	
min	NaN	NaN	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	
25%	NaN	NaN	4.000000	NaN	NaN	NaN	NaN	NaN	NaN	
50%	NaN	NaN	4.300000	NaN	NaN	NaN	NaN	NaN	NaN	
75%	NaN	NaN	4.500000	NaN	NaN	NaN	NaN	NaN	NaN	
max	NaN	NaN	19.000000	NaN	NaN	NaN	NaN	NaN	NaN	

In [15]:
```
df_copy =df.copy()
```

In [16]:
```
# finding the shape of datasets
df_copy.shape
```

Out[16]:
```
(10841, 13)
```

In [17]:
```
# finding the reviews column head
df['Reviews'].head()
```

Out[17]:
```
0       159
1       967
2     87510
3    215644
4       967
Name: Reviews, dtype: object
```

In [18]:
```
# finding the reviews column shape
df['Reviews'].shape
```

Out[18]:
```
(10841,)
```

In [19]:
```
#finding is reviews column is having all numerica values or not
df.Reviews.str.isnumeric().sum()
```

Out[19]:
```
10840
```

In [21]:
```
df.Reviews.str.isnumeric()
```

Out[21]:
```
0        True
1        True
2        True
3        True
4        True
         ... 
10836    True
10837    True
10838    True
10839    True
10840    True
Name: Reviews, Length: 10841, dtype: bool
```

In [20]:
```
#~ symbole represents the inversing of values ex:if its True, ~gives us False
~df.Reviews.str.isnumeric()
```

Out[20]:
```
0        False
1        False
2        False
3        False
4        False
         ...  
10836    False
10837    False
10838    False
10839    False
10840    False
Name: Reviews, Length: 10841, dtype: bool
```

In [22]:
```
#finding in which row we are having this object
df[~df.Reviews.str.isnumeric()]
```

Out[22]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres
10472	Life Made WI-Fi Touchscreen Photo Frame	1.9	19.0	3.0M	1,000+	Free	0	Everyone	NaN	February 11, 2018

In [23]:
```
#from above information Reviews should be in numerical but in given dataset it wo
# here we need to handle it
```

#findings:

here in 10472 row we have an object i'e 3.0M Reviews,

#here we need to handle this

hence it is only one row, we can drop this row for now

In [24]:
```
df_copy = df_copy.drop(df_copy.index[10472])
```

In [25]:
```
# now we can find out the shape of our reviews columns
df_copy['Reviews'].shape
```

Out[25]:
```
(10840,)
```

#Observations: Here we got 10840, hence that row has been dropped or deleted

In [26]:
```
#finding the shape of entire dataframe
df_copy.shape
```

Out[26]:
```
(10840, 13)
```

Observations: Hence one row got deleted

In [27]:
```
#checking the type of  Reviews column
df_copy['Reviews'].dtype
```

Out[27]:
```
dtype('O')
```

Observation: It's showing as an object. But infact its a interger

In [28]:
```
#now converting that reviews column in to interger
df_copy['Reviews']=df_copy['Reviews'].astype('int')
```

In [29]:
```
#now we can check the datatype of Reviews column
df_copy['Reviews'].dtype
```

Out[29]:
```
dtype('int32')
```

Observation: Now its successfully converted into interger

In [30]:
```
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10840 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             10840 non-null  object 
 1   Category        10840 non-null  object 
 2   Rating          9366 non-null   float64
 3   Reviews         10840 non-null  int32  
 4   Size            10840 non-null  object 
 5   Installs        10840 non-null  object 
 6   Type            10839 non-null  object 
 7   Price           10840 non-null  object 
 8   Content Rating  10840 non-null  object 
 9   Genres          10840 non-null  object 
 10  Last Updated    10840 non-null  object 
 11  Current Ver     10832 non-null  object 
 12  Android Ver     10838 non-null  object 
dtypes: float64(1), int32(1), object(11)
memory usage: 1.1+ MB
```

Now we can perform the Analysis on Size Column

In [31]:
```
df_copy['Size'].unique()
```

Out[31]:
```
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)
```

In [32]:

```
#replacing M with 000 in size column
df_copy['Size']= df_copy['Size'].str.replace('M','000')
```

```
In [33]: df_copy['Size'].unique()
```

```
Out[33]: array(['19000', '14000', '8.7000', '25000', '2.8000', '5.6000', '29000',
       '33000', '3.1000', '28000', '12000', '20000', '21000', '37000',
       '2.7000', '5.5000', '17000', '39000', '31000', '4.2000', '7.0000',
       '23000', '6.0000', '6.1000', '4.6000', '9.2000', '5.2000', '11000',
       '24000', 'Varies with device', '9.4000', '15000', '10000',
       '1.2000', '26000', '8.0000', '7.9000', '56000', '57000', '35000',
       '54000', '201k', '3.6000', '5.7000', '8.6000', '2.4000', '27000',
       '2.5000', '16000', '3.4000', '8.9000', '3.9000', '2.9000', '38000',
       '32000', '5.4000', '18000', '1.1000', '2.2000', '4.5000', '9.8000',
       '52000', '9.0000', '6.7000', '30000', '2.6000', '7.1000', '3.7000',
       '22000', '7.4000', '6.4000', '3.2000', '8.2000', '9.9000',
       '4.9000', '9.5000', '5.0000', '5.9000', '13000', '73000', '6.8000',
       '3.5000', '4.0000', '2.3000', '7.2000', '2.1000', '42000',
       '7.3000', '9.1000', '55000', '23k', '6.5000', '1.5000', '7.5000',
       '51000', '41000', '48000', '8.5000', '46000', '8.3000', '4.3000',
       '4.7000', '3.3000', '40000', '7.8000', '8.8000', '6.6000',
       '5.1000', '61000', '66000', '79k', '8.4000', '118k', '44000',
       '695k', '1.6000', '6.2000', '18k', '53000', '1.4000', '3.0000',
       '5.8000', '3.8000', '9.6000', '45000', '63000', '49000', '77000',
       '4.4000', '4.8000', '70000', '6.9000', '9.3000', '10.0000',
       '8.1000', '36000', '84000', '97000', '2.0000', '1.9000', '1.8000',
       '5.3000', '47000', '556k', '526k', '76000', '7.6000', '59000',
       '9.7000', '78000', '72000', '43000', '7.7000', '6.3000', '334k',
       '34000', '93000', '65000', '79000', '100000', '58000', '50000',
       '68000', '64000', '67000', '60000', '94000', '232k', '99000',
       '624k', '95000', '8.5k', '41k', '292k', '11k', '80000', '1.7000',
       '74000', '62000', '69000', '75000', '98000', '85000', '82000',
       '96000', '87000', '71000', '86000', '91000', '81000', '92000',
       '83000', '88000', '704k', '862k', '899k', '378k', '266k', '375k',
       '1.3000', '975k', '980k', '4.1000', '89000', '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', '90000', '545k', '61k', '283k', '655k',
       '714k', '93k', '872k', '121k', '322k', '1.0000', '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)
```

```
In [34]: #replacing k with ''
         df_copy['Size']= df_copy['Size'].str.replace('k','')
```

```
In [35]: df_copy['Size'].unique()
```

```
Out[35]: array(['19000', '14000', '8.7000', '25000', '2.8000', '5.6000', '29000',
                '33000', '3.1000', '28000', '12000', '20000', '21000', '37000',
                '2.7000', '5.5000', '17000', '39000', '31000', '4.2000', '7.0000',
                '23000', '6.0000', '6.1000', '4.6000', '9.2000', '5.2000', '11000',
                '24000', 'Varies with device', '9.4000', '15000', '10000',
                '1.2000', '26000', '8.0000', '7.9000', '56000', '57000', '35000',
                '54000', '201', '3.6000', '5.7000', '8.6000', '2.4000', '27000',
                '2.5000', '16000', '3.4000', '8.9000', '3.9000', '2.9000', '38000',
                '32000', '5.4000', '18000', '1.1000', '2.2000', '4.5000', '9.8000',
                '52000', '9.0000', '6.7000', '30000', '2.6000', '7.1000', '3.7000',
                '22000', '7.4000', '6.4000', '3.2000', '8.2000', '9.9000',
                '4.9000', '9.5000', '5.0000', '5.9000', '13000', '73000', '6.8000',
                '3.5000', '4.0000', '2.3000', '7.2000', '2.1000', '42000',
                '7.3000', '9.1000', '55000', '23', '6.5000', '1.5000', '7.5000',
                '51000', '41000', '48000', '8.5000', '46000', '8.3000', '4.3000',
                '4.7000', '3.3000', '40000', '7.8000', '8.8000', '6.6000',
                '5.1000', '61000', '66000', '79', '8.4000', '118', '44000', '695',
                '1.6000', '6.2000', '18', '53000', '1.4000', '3.0000', '5.8000',
                '3.8000', '9.6000', '45000', '63000', '49000', '77000', '4.4000',
                '4.8000', '70000', '6.9000', '9.3000', '10.0000', '8.1000',
                '36000', '84000', '97000', '2.0000', '1.9000', '1.8000', '5.3000',
                '47000', '556', '526', '76000', '7.6000', '59000', '9.7000',
                '78000', '72000', '43000', '7.7000', '6.3000', '334', '34000',
                '93000', '65000', '79000', '100000', '58000', '50000', '68000',
                '64000', '67000', '60000', '94000', '232', '99000', '624', '95000',
                '8.5', '41', '292', '11', '80000', '1.7000', '74000', '62000',
                '69000', '75000', '98000', '85000', '82000', '96000', '87000',
                '71000', '86000', '91000', '81000', '92000', '83000', '88000',
                '704', '862', '899', '378', '266', '375', '1.3000', '975', '980',
                '4.1000', '89000', '696', '544', '525', '920', '779', '853', '720',
                '713', '772', '318', '58', '241', '196', '857', '51', '953', '865',
                '251', '930', '540', '313', '746', '203', '26', '314', '239',
                '371', '220', '730', '756', '91', '293', '17', '74', '14', '317',
                '78', '924', '902', '818', '81', '939', '169', '45', '475', '965',
                '90000', '545', '61', '283', '655', '714', '93', '872', '121',
                '322', '1.0000', '976', '172', '238', '549', '206', '954', '444',
                '717', '210', '609', '308', '705', '306', '904', '473', '175',
                '350', '383', '454', '421', '70', '812', '442', '842', '417',
                '412', '459', '478', '335', '782', '721', '430', '429', '192',
                '200', '460', '728', '496', '816', '414', '506', '887', '613',
                '243', '569', '778', '683', '592', '319', '186', '840', '647',
                '191', '373', '437', '598', '716', '585', '982', '222', '219',
                '55', '948', '323', '691', '511', '951', '963', '25', '554', '351',
                '27', '82', '208', '913', '514', '551', '29', '103', '898', '743',
                '116', '153', '209', '353', '499', '173', '597', '809', '122',
                '411', '400', '801', '787', '237', '50', '643', '986', '97', '516',
                '837', '780', '961', '269', '20', '498', '600', '749', '642',
                '881', '72', '656', '601', '221', '228', '108', '940', '176', '33',
                '663', '34', '942', '259', '164', '458', '245', '629', '28', '288',
                '775', '785', '636', '916', '994', '309', '485', '914', '903',
                '608', '500', '54', '562', '847', '957', '688', '811', '270', '48',
                '329', '523', '921', '874', '981', '784', '280', '24', '518',
                '754', '892', '154', '860', '364', '387', '626', '161', '879',
                '39', '970', '170', '141', '160', '144', '143', '190', '376',
                '193', '246', '73', '658', '992', '253', '420', '404', '470',
                '226', '240', '89', '234', '257', '861', '467', '157', '44', '676',
                '67', '552', '885', '1020', '582', '619'], dtype=object)
```

In [36]:

```
#replacing one string variable with nan value using numpy
df_copy['Size']= df_copy['Size'].str.replace('Varies with device',str(np.nan))
```

In [37]:

```
df_copy['Size'].dtype
```

Out[37]:

```
dtype('O')
```

In [38]:

```
#converting Object to float data type
df_copy['Size']=df_copy['Size'].astype('float')
```

In [40]:

```
df_copy['Size'].dtype
```

Out[40]:

```
dtype('float64')
```

In [41]:

```
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10840 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             10840 non-null  object 
 1   Category        10840 non-null  object 
 2   Rating          9366 non-null   float64
 3   Reviews         10840 non-null  int32  
 4   Size            9145 non-null   float64
 5   Installs        10840 non-null  object 
 6   Type            10839 non-null  object 
 7   Price           10840 non-null  object 
 8   Content Rating  10840 non-null  object 
 9   Genres          10840 non-null  object 
 10  Last Updated    10840 non-null  object 
 11  Current Ver     10832 non-null  object 
 12  Android Ver     10838 non-null  object 
dtypes: float64(2), int32(1), object(10)
memory usage: 1.1+ MB
```

In [42]:

```
#finding the null values in size column
df_copy['Size'].isnull().sum()
```

Out[42]:

```
1695
```

In [43]:

```
df_copy['Size'].mean()
```

Out[43]:

```
19579.41991252059
```

In [44]:

```
#finding the 2nd index of Size column
df_copy['Size'][2]*1000
```

Out[44]:

```
8700.0
```

In [45]:

```
for i in df_copy['Size']:
    if i<10:
        df_copy['Size']=df_copy['Size'].replace(i, i*1000)
```

In [47]:

```
df_copy['Size'].unique()
```

Out[47]:

```
array([1.90e+04, 1.40e+04, 8.70e+03, 2.50e+04, 2.80e+03, 5.60e+03,
       2.90e+04, 3.30e+04, 3.10e+03, 2.80e+04, 1.20e+04, 2.00e+04,
       2.10e+04, 3.70e+04, 2.70e+03, 5.50e+03, 1.70e+04, 3.90e+04,
       3.10e+04, 4.20e+03, 7.00e+03, 2.30e+04, 6.00e+03, 6.10e+03,
       4.60e+03, 9.20e+03, 5.20e+03, 1.10e+04, 2.40e+04,      nan,
       9.40e+03, 1.50e+04, 1.00e+04, 1.20e+03, 2.60e+04, 8.00e+03,
       7.90e+03, 5.60e+04, 5.70e+04, 3.50e+04, 5.40e+04, 2.01e+02,
       3.60e+03, 5.70e+03, 8.60e+03, 2.40e+03, 2.70e+04, 2.50e+03,
       1.60e+04, 3.40e+03, 8.90e+03, 3.90e+03, 2.90e+03, 3.80e+04,
       3.20e+04, 5.40e+03, 1.80e+04, 1.10e+03, 2.20e+03, 4.50e+03,
       9.80e+03, 5.20e+04, 9.00e+03, 6.70e+03, 3.00e+04, 2.60e+03,
       7.10e+03, 3.70e+03, 2.20e+04, 7.40e+03, 6.40e+03, 3.20e+03,
       8.20e+03, 9.90e+03, 4.90e+03, 9.50e+03, 5.00e+03, 5.90e+03,
       1.30e+04, 7.30e+04, 6.80e+03, 3.50e+03, 4.00e+03, 2.30e+03,
       7.20e+03, 2.10e+03, 4.20e+04, 7.30e+03, 9.10e+03, 5.50e+04,
       2.30e+01, 6.50e+03, 1.50e+03, 7.50e+03, 5.10e+04, 4.10e+04,
       4.80e+04, 8.50e+03, 4.60e+04, 8.30e+03, 4.30e+03, 4.70e+03,
       3.30e+03, 4.00e+04, 7.80e+03, 8.80e+03, 6.60e+03, 5.10e+03,
       6.10e+04, 6.60e+04, 7.90e+01, 8.40e+03, 1.18e+02, 4.40e+04,
       6.95e+02, 1.60e+03, 6.20e+03, 1.80e+01, 5.30e+04, 1.40e+03,
       3.00e+03, 5.80e+03, 3.80e+03, 9.60e+03, 4.50e+04, 6.30e+04,
       4.90e+04, 7.70e+04, 4.40e+03, 4.80e+03, 7.00e+04, 6.90e+03,
       9.30e+03, 1.00e+01, 8.10e+03, 3.60e+04, 8.40e+04, 9.70e+04,
       2.00e+03, 1.90e+03, 1.80e+03, 5.30e+03, 4.70e+04, 5.56e+02,
       5.26e+02, 7.60e+04, 7.60e+03, 5.90e+04, 9.70e+03, 7.80e+04,
       7.20e+04, 4.30e+04, 7.70e+03, 6.30e+03, 3.34e+02, 3.40e+04,
       9.30e+04, 6.50e+04, 7.90e+04, 1.00e+05, 5.80e+04, 5.00e+04,
       6.80e+04, 6.40e+04, 6.70e+04, 6.00e+04, 9.40e+04, 2.32e+02,
       9.90e+04, 6.24e+02, 9.50e+04, 4.10e+01, 2.92e+02, 1.10e+01,
       8.00e+04, 1.70e+03, 7.40e+04, 6.20e+04, 6.90e+04, 7.50e+04,
       9.80e+04, 8.50e+04, 8.20e+04, 9.60e+04, 8.70e+04, 7.10e+04,
       8.60e+04, 9.10e+04, 8.10e+04, 9.20e+04, 8.30e+04, 8.80e+04,
       7.04e+02, 8.62e+02, 8.99e+02, 3.78e+02, 2.66e+02, 3.75e+02,
       1.30e+03, 9.75e+02, 9.80e+02, 4.10e+03, 8.90e+04, 6.96e+02,
       5.44e+02, 5.25e+02, 9.20e+02, 7.79e+02, 8.53e+02, 7.20e+02,
       7.13e+02, 7.72e+02, 3.18e+02, 5.80e+01, 2.41e+02, 1.96e+02,
       8.57e+02, 5.10e+01, 9.53e+02, 8.65e+02, 2.51e+02, 9.30e+02,
       5.40e+02, 3.13e+02, 7.46e+02, 2.03e+02, 2.60e+01, 3.14e+02,
       2.39e+02, 3.71e+02, 2.20e+02, 7.30e+02, 7.56e+02, 9.10e+01,
       2.93e+02, 1.70e+01, 7.40e+01, 1.40e+01, 3.17e+02, 7.80e+01,
       9.24e+02, 9.02e+02, 8.18e+02, 8.10e+01, 9.39e+02, 1.69e+02,
       4.50e+01, 4.75e+02, 9.65e+02, 9.00e+04, 5.45e+02, 6.10e+01,
       2.83e+02, 6.55e+02, 7.14e+02, 9.30e+01, 8.72e+02, 1.21e+02,
       3.22e+02, 1.00e+03, 9.76e+02, 1.72e+02, 2.38e+02, 5.49e+02,
       2.06e+02, 9.54e+02, 4.44e+02, 7.17e+02, 2.10e+02, 6.09e+02,
       3.08e+02, 7.05e+02, 3.06e+02, 9.04e+02, 4.73e+02, 1.75e+02,
       3.50e+02, 3.83e+02, 4.54e+02, 4.21e+02, 7.00e+01, 8.12e+02,
       4.42e+02, 8.42e+02, 4.17e+02, 4.12e+02, 4.59e+02, 4.78e+02,
       3.35e+02, 7.82e+02, 7.21e+02, 4.30e+02, 4.29e+02, 1.92e+02,
       2.00e+02, 4.60e+02, 7.28e+02, 4.96e+02, 8.16e+02, 4.14e+02,
       5.06e+02, 8.87e+02, 6.13e+02, 2.43e+02, 5.69e+02, 7.78e+02,
       6.83e+02, 5.92e+02, 3.19e+02, 1.86e+02, 8.40e+02, 6.47e+02,
       1.91e+02, 3.73e+02, 4.37e+02, 5.98e+02, 7.16e+02, 5.85e+02,
       9.82e+02, 2.22e+02, 2.19e+02, 5.50e+01, 9.48e+02, 3.23e+02,
       6.91e+02, 5.11e+02, 9.51e+02, 9.63e+02, 2.50e+01, 5.54e+02,
       3.51e+02, 2.70e+01, 8.20e+01, 2.08e+02, 9.13e+02, 5.14e+02,
       5.51e+02, 2.90e+01, 1.03e+02, 8.98e+02, 7.43e+02, 1.16e+02,
       1.53e+02, 2.09e+02, 3.53e+02, 4.99e+02, 1.73e+02, 5.97e+02,
       8.09e+02, 1.22e+02, 4.11e+02, 4.00e+02, 8.01e+02, 7.87e+02,
       2.37e+02, 5.00e+01, 6.43e+02, 9.86e+02, 9.70e+01, 5.16e+02,
       8.37e+02, 7.80e+02, 9.61e+02, 2.69e+02, 2.00e+01, 4.98e+02,
       6.00e+02, 7.49e+02, 6.42e+02, 8.81e+02, 7.20e+01, 6.56e+02,
       6.01e+02, 2.21e+02, 2.28e+02, 1.08e+02, 9.40e+02, 1.76e+02,
       3.30e+01, 6.63e+02, 3.40e+01, 9.42e+02, 2.59e+02, 1.64e+02,
       4.58e+02, 2.45e+02, 6.29e+02, 2.80e+01, 2.88e+02, 7.75e+02,
       7.85e+02, 6.36e+02, 9.16e+02, 9.94e+02, 3.09e+02, 4.85e+02,
       9.14e+02, 9.03e+02, 6.08e+02, 5.00e+02, 5.40e+01, 5.62e+02,
       8.47e+02, 9.57e+02, 6.88e+02, 8.11e+02, 2.70e+02, 4.80e+01,
       3.29e+02, 5.23e+02, 9.21e+02, 8.74e+02, 9.81e+02, 7.84e+02,
       2.80e+02, 2.40e+01, 5.18e+02, 7.54e+02, 8.92e+02, 1.54e+02,
       8.60e+02, 3.64e+02, 3.87e+02, 6.26e+02, 1.61e+02, 8.79e+02,
       3.90e+01, 9.70e+02, 1.70e+02, 1.41e+02, 1.60e+02, 1.44e+02,
       1.43e+02, 1.90e+02, 3.76e+02, 1.93e+02, 2.46e+02, 7.30e+01,
       6.58e+02, 9.92e+02, 2.53e+02, 4.20e+02, 4.04e+02, 4.70e+02,
       2.26e+02, 2.40e+02, 8.90e+01, 2.34e+02, 2.57e+02, 8.61e+02,
       4.67e+02, 1.57e+02, 4.40e+01, 6.76e+02, 6.70e+01, 5.52e+02,
       8.85e+02, 1.02e+03, 5.82e+02, 6.19e+02])
```

In [48]:
```
df_copy.columns
```

Out[48]:
```
Index(['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type',
       'Price', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver',
       'Android Ver'],
      dtype='object')
```

In [50]:
```
df_copy['Installs'].head()
```

Out[50]:
```
0        10,000+
1       500,000+
2     5,000,000+
3    50,000,000+
4       100,000+
Name: Installs, dtype: object
```

In [53]:
```
df_copy['Installs'].unique()
```

Out[53]:
```
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)
```

```
In [52]: df_copy['Price'].unique()
```

```
Out[52]: 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)
```

```
In [58]: #replacing the +, '', $ with ''
         char_to_remove=['+','','$']
         cols_to_clean = ['Installs', 'Price']
         for item in char_to_remove:
             for col in cols_to_clean:
                 df_copy[col]=df_copy[col].str.replace(item, '')
```

```
In [59]: df_copy['Price'].unique()
```

```
Out[59]: 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)
```

Observations: Here we got cleaned data without $ symbol

```
In [60]: df_copy['Installs'].unique()
```

```
Out[60]: 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'], dtype=object)
```

Observations: Here we got cleaned data without + symbol

In [63]:

```
#converting datatype in to float
df_copy['Price']=df_copy['Price'].astype(float)
```

```
In [64]: df_copy['Installs']=df_copy['Installs'].astype(int)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [64], in <cell line: 1>()
----> 1 df_copy['Installs']=df_copy['Installs'].astype(int)

File ~\anaconda3\lib\site-packages\pandas\core\generic.py:5912, in NDFrame.asty
pe(self, dtype, copy, errors)
   5905     results = [
   5906         self.iloc[:, i].astype(dtype, copy=copy)
   5907         for i in range(len(self.columns))
   5908     ]
   5910 else:
   5911     # else, only a single dtype is given
-> 5912     new_data = self._mgr.astype(dtype=dtype, copy=copy, errors=errors)
   5913     return self._constructor(new_data).__finalize__(self, method="astyp
e")
   5915 # GH 33113: handle empty frame or series

File ~\anaconda3\lib\site-packages\pandas\core\internals\managers.py:419, in Ba
seBlockManager.astype(self, dtype, copy, errors)
    418 def astype(self: T, dtype, copy: bool = False, errors: str = "raise") -
> T:
--> 419     return self.apply("astype", dtype=dtype, copy=copy, errors=errors)

File ~\anaconda3\lib\site-packages\pandas\core\internals\managers.py:304, in Ba
seBlockManager.apply(self, f, align_keys, ignore_failures, **kwargs)
    302         applied = b.apply(f, **kwargs)
    303     else:
--> 304         applied = getattr(b, f)(**kwargs)
    305 except (TypeError, NotImplementedError):
    306     if not ignore_failures:

File ~\anaconda3\lib\site-packages\pandas\core\internals\blocks.py:580, in Bloc
k.astype(self, dtype, copy, errors)
    562 """
    563 Coerce to the new dtype.
    564
   (...)
    576 Block
    577 """
    578 values = self.values
--> 580 new_values = astype_array_safe(values, dtype, copy=copy, errors=errors)
    582 new_values = maybe_coerce_values(new_values)
    583 newb = self.make_block(new_values)

File ~\anaconda3\lib\site-packages\pandas\core\dtypes\cast.py:1292, in astype_a
rray_safe(values, dtype, copy, errors)
   1289     dtype = dtype.numpy_dtype
   1291 try:
-> 1292     new_values = astype_array(values, dtype, copy=copy)
   1293 except (ValueError, TypeError):
   1294     # e.g. astype_nansafe can fail on object-dtype of strings
   1295     #  trying to convert to float
   1296     if errors == "ignore":
```

```
File ~\anaconda3\lib\site-packages\pandas\core\dtypes\cast.py:1237, in astype_a
rray(values, dtype, copy)
   1234     values = values.astype(dtype, copy=copy)
   1236 else:
-> 1237     values = astype_nansafe(values, dtype, copy=copy)
   1239 # in pandas we don't store numpy str dtypes, so convert to object
   1240 if isinstance(dtype, np.dtype) and issubclass(values.dtype.type, str):

File ~\anaconda3\lib\site-packages\pandas\core\dtypes\cast.py:1154, in astype_n
ansafe(arr, dtype, copy, skipna)
   1150 elif is_object_dtype(arr.dtype):
   1151 
   1152     # work around NumPy brokenness, #1987
   1153     if np.issubdtype(dtype.type, np.integer):
-> 1154         return lib.astype_intsafe(arr, dtype)
   1156     # if we have a datetime/timedelta array of objects
   1157     # then coerce to a proper dtype and recall astype_nansafe
   1159     elif is_datetime64_dtype(dtype):

File ~\anaconda3\lib\site-packages\pandas\_libs\lib.pyx:668, in pandas._libs.li
b.astype_intsafe()

ValueError: invalid literal for int() with base 10: '10,000'
```

In [65]:
```
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10840 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             10840 non-null  object 
 1   Category        10840 non-null  object 
 2   Rating          9366 non-null   float64
 3   Reviews         10840 non-null  int32  
 4   Size            9145 non-null   float64
 5   Installs        10840 non-null  object 
 6   Type            10839 non-null  object 
 7   Price           10840 non-null  float64
 8   Content Rating  10840 non-null  object 
 9   Genres          10840 non-null  object 
 10  Last Updated    10840 non-null  object 
 11  Current Ver     10832 non-null  object 
 12  Android Ver     10838 non-null  object 
dtypes: float64(3), int32(1), object(9)
memory usage: 1.4+ MB
```

In [68]:
```
df_copy['Last Updated'].unique()
```

Out[68]:
```
array(['January 7, 2018', 'January 15, 2018', 'August 1, 2018', ...,
       'January 20, 2014', 'February 16, 2014', 'March 23, 2014'],
      dtype=object)
```

In [70]:
```
df_copy['Last Updated'].dtype
```

Out[70]:
```
dtype('O')
```

In [72]:
```
#converting date and time using pandas
df_copy['Last Updated']=pd.to_datetime(df_copy['Last Updated'])
```

In [73]:
```
df_copy['Last Updated']
```

Out[73]:
```
0       2018-01-07
1       2018-01-15
2       2018-08-01
3       2018-06-08
4       2018-06-20
           ...
10836   2017-07-25
10837   2018-07-06
10838   2017-01-20
10839   2015-01-19
10840   2018-07-25
Name: Last Updated, Length: 10840, dtype: datetime64[ns]
```

In [79]:
```
#finding only day
df_copy['day']=df_copy['Last Updated'].dt.day
```

In [78]:
```
#finding only month
df_copy['month']=df_copy['Last Updated'].dt.month
```

In [77]:
```
#finding only year
df_copy['year']=df_copy['Last Updated'].dt.year
```

In [80]:
```
df_copy.head()
```

Out[80]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19000.0	10,000	Free	0.0	Everyone	
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14000.0	500,000	Free	0.0	Everyone	D
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8700.0	5,000,000	Free	0.0	Everyone	
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25000.0	50,000,000	Free	0.0	Teen	
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2800.0	100,000	Free	0.0	Everyone	Des

In [81]:
```
#saving the data in our computer
df_copy.to_csv('Google cleaned', index = False)
```

In [83]:
```
pwd
```

Out[83]:
```
'C:\\Users\\lenovo'
```

Observations: Here we can find our cleaned data

In [ ]:
```

```