In [1]:

```
# Import statements
# Import necessary python libraries and packages


# For data analysis & manipulation
import pandas as pd
import numpy as np

# For visualising distributional values
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
# Python version

import sys
print ("The Python version is: {}".format(sys.version))
```

```
The Python version is: 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915
64 bit (AMD64)]
```

In [3]:

```
# Generating the version of a wide variety of packages/libraries used

pd.__version__
pd.show_versions(as_json=False)
```

```
INSTALLED VERSIONS
------------------
commit           : None
python           : 3.7.4.final.0
python-bits      : 64
OS               : Windows
OS-release       : 10
machine          : AMD64
processor        : Intel64 Family 6 Model 126 Stepping 5, GenuineIntel
byteorder        : little
LC_ALL           : None
LANG             : None
LOCALE           : None.None

pandas           : 0.25.1
numpy            : 1.16.5
pytz             : 2019.3
dateutil         : 2.8.0
pip              : 19.2.3
setuptools       : 41.4.0
Cython           : 0.29.13
pytest           : 5.2.1
hypothesis       : None
sphinx           : 2.2.0
blosc            : None
feather          : None
xlsxwriter       : 1.2.1
lxml.etree       : 4.4.1
html5lib         : 1.0.1
pymysql          : None
psycopg2         : None
jinja2           : 2.10.3
IPython          : 7.8.0
pandas_datareader: None
bs4              : 4.8.0
bottleneck       : 1.2.1
fastparquet      : None
gcsfs            : None
lxml.etree       : 4.4.1
matplotlib       : 3.1.1
numexpr          : 2.7.0
odfpy            : None
openpyxl         : 3.0.0
pandas_gbq       : None
pyarrow          : None
pytables         : None
s3fs             : None
scipy            : 1.3.1
sqlalchemy       : 1.3.9
tables           : 3.5.2
xarray           : None
xlrd             : 1.2.0
xlwt             : 1.3.0
xlsxwriter       : 1.2.1
```

In [4]:

```
# Assigning the dataset with the name: "app"

app= pd.read_csv(r"C:\Users\dimit\Google-Playstore-Full.csv", low_memory=False)
```

In [5]:

```
# The type of this dataset is a dataframe
type(app)
```

Out[5]:

```
pandas.core.frame.DataFrame
```

In [6]:

```
# The columns of this dataframe are "series"
type(app["Installs"])
```

Out[6]:

```
pandas.core.series.Series
```

In [7]:

```
# First 5 rows of the dataframe

app.head()
```

Out[7]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Co R
0	DoorDash - Food Delivery	FOOD_AND_DRINK	4.548561573	305034	5,000,000+	Varies with device	0	Ever
1	TripAdvisor Hotels Flights Restaurants Attract...	TRAVEL_AND_LOCAL	4.400671482	1207922	100,000,000+	Varies with device	0	Ever
2	Peapod	SHOPPING	3.656329393	1967	100,000+	1.4M	0	Ever
3	foodpanda - Local Food Delivery	FOOD_AND_DRINK	4.107232571	389154	10,000,000+	16M	0	Ever
4	My CookBook Pro (Ad Free)	FOOD_AND_DRINK	4.647752285	2291	10,000+	Varies with device	$5.99	Ever

In [8]:

```
# Getting the last five rows

app.tail()
```

Out[8]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Co R
267047	Community Healthplex	HEALTH_AND_FITNESS	5	1	100+	4.2M	0	Evei
267048	Pet ads: Buy & Sell	BUSINESS	2.599999905	5	500+	8.4M	0	Evei
267049	Collectors Market: Buy & Sell	BUSINESS	3.285714388	7	1,000+	7.9M	0	Evei
267050	Car Market, Buy & Sell	BUSINESS	5	1	1,000+	8.2M	0	Evei
267051	Selfie with Ariana Grande	PHOTOGRAPHY	4.611111164	18	1,000+	7.8M	0	Evei

In [9]:

```
# Getting the number of rows and columns of the dataframe

app.shape
```

Out[9]:

```
(267052, 15)
```

In [10]:

```
# Removing the columns with index position: 11, 12, 13, 14. They do not seem to offer a
ny substantial value to the data analysis

app=app.drop("Unnamed: 11", axis=1)
app=app.drop("Unnamed: 12", axis=1)
app=app.drop("Unnamed: 13", axis=1)
app=app.drop("Unnamed: 14", axis=1)
```

In [11]:

```
# Number of rows and columns after removing the useless columns

app.shape
```

Out[11]:

```
(267052, 11)
```

In [12]:

```
# Columns after removing the useless ones

app.columns
```

Out[12]:

```
Index(['App Name', 'Category', 'Rating', 'Reviews', 'Installs', 'Size',
       'Price', 'Content Rating', 'Last Updated', 'Minimum Version',
       'Latest Version'],
      dtype='object')
```

In [13]:

```
# Number of app categories

app["Category"].nunique()
```

Out[13]:

```
67
```

In [14]:

```
# The app categories

app.Category.unique()
```

Out[14]:

```
array(['FOOD_AND_DRINK', 'TRAVEL_AND_LOCAL', 'SHOPPING', 'LIFESTYLE',
       'GAME_ACTION', 'GAME_CASUAL', 'GAME_ROLE_PLAYING', 'GAME_PUZZLE',
       'GAME_RACING', 'GAME_ADVENTURE', 'GAME_ARCADE', 'GAME_STRATEGY',
       'GAME_SPORTS', 'GAME_SIMULATION', 'GAME_MUSIC', 'MUSIC_AND_AUDIO',
       'FINANCE', 'EVENTS', 'ENTERTAINMENT', 'EDUCATION',
       'GAME_EDUCATIONAL', 'BOOKS_AND_REFERENCE', 'NEWS_AND_MAGAZINES',
       'PHOTOGRAPHY', 'VIDEO_PLAYERS', 'GAME_WORD', 'ART_AND_DESIGN',
       'GAME_TRIVIA', 'GAME_BOARD', 'BUSINESS', 'PRODUCTIVITY',
       'COMMUNICATION', 'HEALTH_AND_FITNESS', 'HOUSE_AND_HOME', 'SOCIAL',
       'BEAUTY', 'GAME_CASINO', 'MAPS_AND_NAVIGATION', 'PERSONALIZATION',
       'GAME_CARD', 'TOOLS', 'SPORTS', 'AUTO_AND_VEHICLES',
       'LIBRARIES_AND_DEMO', 'COMICS', 'PARENTING', 'DATING', 'WEATHER',
       'MEDICAL', ' Podcasts', ')', ' Channel 2 News', nan,
       ' Breaking News', '6', 'Gate ALARM', ' Alfabe \ufffd?ren',
       ' T\ufffdrk Alfabesi', ' not notified you follow -', ' Mexpost)',
       ' Romantic Song Music Love Songs', ' ETEA & MDCAT', ' Tour Guide',
       'TRAVEL', ' Speaker Pro 2019', ' Islamic Name Boy & Girl+Meaning',
       ' Accounting', ' super loud speaker booster'], dtype=object)
```

In [15]:

```
# Viewing the number of classes (gradation) of the number of installations
# There are 38 different classes

app["Installs"].nunique()
```

Out[15]:

```
38
```

In [16]:

```
# The gradation of installations in the dataframe

# There seem to be some input mistakes, such as "EDUCATION", which should not belong in
this column. They will be edited


app.Installs.unique()
```

Out[16]:

```
array(['5,000,000+', '100,000,000+', '100,000+', '10,000,000+', '10,000+',
       '1,000,000+', '50,000,000+', '500,000+', '50,000+', '5,000+',
       '1,000+', '500,000,000+', '1,000,000,000+', '5,000,000,000+',
       '100+', '500+', '50+', '5+', '10+', '1+', 'EDUCATION', '6',
       '11976', '0+', '20', '156', ' Xmax X', '166', '1', '54', '71',
       '59', '13', '117', '511', '927', '4.823529243', '10'], dtype=objec
t)
```

In [17]:

```
# There are a lot of app sizes

app["Size"].nunique()
```

Out[17]:

```
1248
```

In [18]:

```
# Viewing the content rating; who is permitted to download these apps

# There are some invalid contents. They will be edited

app["Content Rating"].unique()
```

Out[18]:

```
array(['Everyone', 'Teen', 'Everyone 10+', 'Mature 17+',
       'Adults only 18+', '100,000+', 'Unrated', '$0.99', '0', '3702',
       '$2.49', '17M'], dtype=object)
```

In [19]:

```
# the number of categories of the age content rating

len(app["Content Rating"].unique())
```

Out[19]:

```
12
```

In [20]:

```
#current first five rows

app.head()
```

Out[20]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Co R
0	DoorDash - Food Delivery	FOOD_AND_DRINK	4.548561573	305034	5,000,000+	Varies with device	0	Ever
1	TripAdvisor Hotels Flights Restaurants Attract...	TRAVEL_AND_LOCAL	4.400671482	1207922	100,000,000+	Varies with device	0	Ever
2	Peapod	SHOPPING	3.656329393	1967	100,000+	1.4M	0	Ever
3	foodpanda - Local Food Delivery	FOOD_AND_DRINK	4.107232571	389154	10,000,000+	16M	0	Ever
4	My CookBook Pro (Ad Free)	FOOD_AND_DRINK	4.647752285	2291	10,000+	Varies with device	$5.99	Ever

In [21]:

```
app.isnull().sum()
```

Out[21]:

```
App Name           1
Category           1
Rating             0
Reviews            1
Installs           0
Size               0
Price              0
Content Rating     0
Last Updated       0
Minimum Version    1
Latest Version     3
dtype: int64
```

In [22]:

```
# There are totally 11 empty data entries which will be dropped
len(app.isnull().sum())
```

Out[22]:

```
11
```

In [23]:

```
# Dropping the entries where there are missing values

app=app.dropna()
```

In [24]:

```
app.isnull().any()

# False for every category means that there are no longer missing values
```

Out[24]:

```
App Name           False
Category           False
Rating             False
Reviews            False
Installs           False
Size               False
Price              False
Content Rating     False
Last Updated       False
Minimum Version    False
Latest Version     False
dtype: bool
```

In [25]:

```
# Ensuring there are no missing values in any column, in any data of every column

app.isnull().any().any()
```

Out[25]:

```
False
```

In [26]:

```
##########################################################################################
################################
```

In [27]:

```
##########################################################################################
################################
```

In [28]:

```
##########################################################################################
################################
```

Cleaning of the Data - Exploring and Managing the Data

In [29]:

```
# Start of cleaning
# There were given some commands to locate any invalid data
# I noticed that are some misplacing, e.g. here, "4" should move to "Rating", and "GAME
_STRATEGY" should move to "Category"

# Wherever the data are misplaced but valid, the data will be kept and edited (correcti
ng the entry positions)
# Wherever the data are misplaced and invalid too (with lot's of mistakes), the data wi
ll be removed

app[app["Rating"]== "GAME_STRATEGY"]
```

Out[29]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	Last Updated
13504	Never have I ever 18+	)	GAME_STRATEGY	4	6	100+	2.4M	$0.99	Mature 17+

In [30]:

```
# dropping the invalid entry

app.drop(index=13504, inplace=True)
```

In [31]:

```
# Now the column "Rating" is fixed

app[app["Rating"]== "GAME_STRATEGY"]
```

Out[31]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	Last Updated	Minimum Version	Lat Vers

In [32]:

```
# Noticing the same pattern. Wrong entry data in the columns

app[app["Rating"]== "NEWS_AND_MAGAZINES"]
```

Out[32]:

	App Name	Category	Rating	Reviews	Installs	Size	Price
23457	Israel News	Channel 2 News	NEWS_AND_MAGAZINES	3.857798815	11976	1,000,000+	Varies with device
48438	Mojo Times: Bihar Hindi Video News	Breaking News	NEWS_AND_MAGAZINES	4.775640965	156	10,000+	6.9M

In [33]:

```
# Here the data are misplaced but valid

# I am manually fixing the misplacing data values

app.loc[23457, ["Category", "Rating", "Reviews", "Installs", "Size", "Price", "Content
 Rating", "Last Updated", "Minimum Version", "Latest Version"]] = "NEWS_AND_MAGAZINES",
"3.857798815", "11976", "1,000,000+", "Varies with device", "0", "Everyone 10+", "March
16, 2019", "Varies with device", "NaN"
```

In [34]:

```
app.loc[23457]
```

Out[34]:

```
App Name                  Israel News
Category           NEWS_AND_MAGAZINES
Rating                    3.857798815
Reviews                         11976
Installs                  1,000,000+
Size               Varies with device
Price                               0
Content Rating            Everyone 10+
Last Updated           March 16, 2019
Minimum Version    Varies with device
Latest Version                    NaN
Name: 23457, dtype: object
```

In [35]:

```
app.loc[48438, ["Category", "Rating", "Reviews", "Installs", "Size", "Price", "Content
 Rating", "Last Updated", "Minimum Version", "Latest Version"]] = "NEWS_AND_MAGAZINES",
"4.775640965", "156", "10,000+", "6.9M", "0", "Teen", "March 30, 2019","4.1 and up", "N
aN"
```

In [36]:

```
app.loc[48438]
```

Out[36]:

```
App Name           Mojo Times: Bihar Hindi Video News
Category                             NEWS_AND_MAGAZINES
Rating                                     4.775640965
Reviews                                            156
Installs                                       10,000+
Size                                              6.9M
Price                                                0
Content Rating                                   Teen
Last Updated                           March 30, 2019
Minimum Version                           4.1 and up
Latest Version                                   NaN
Name: 48438, dtype: object
```

In [ ]:

```

```

In [37]:

```
# Here is an example of misplaced data with a lot of mistakes. It does not seem importa
nt to be fixed, it will be dropped

app[app["Rating"]== "ENTERTAINMENT"]
```

Out[37]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	Upd
113151	Steins	Gate ALARM	ENTERTAINMENT	4.716867447	166	500+	67M	$0.99	

In [38]:

```
app.drop(index=113151, inplace=True)
```

In [39]:

```
# Ensuring that there are no longer wrong entries in the column "Rating"

app[app["Rating"]== "ENTERTAINMENT"]
```

Out[39]:

App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	Last Updated	Minimum Version	Lat Vers

In [ ]:

```

```

In [40]:

```
app[app["Rating"]== "EDUCATION"]
```

Out[40]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	Upd
125479	2-6 Ya? E?itici ?ocuk Zeka Oyunlar?	Alfabe ??ren	EDUCATION	5	1	10+	57M	$2.49	Eve
125480	2-6 Ya? E?itici ?ocuk Zeka Oyunlar?	T?rk Alfabesi	EDUCATION	4.333333492	54	50,000+	43M	0	Eve
180371	eShagird - Online academy	ETEA & MDCAT	EDUCATION	4.504273415	117	10,000+	6.9M	0	Eve

In [41]:

```
# Dropping these data entries which do not seem important and they have a lot of mistak
es

app.drop(index=125479, inplace=True)
app.drop(index=125480, inplace=True)
app.drop(index=180371, inplace=True)
app[app["Rating"]== "EDUCATION"]
```

Out[41]:

App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	Last Updated	Minimum Version	Lat Vers

In [ ]:

```
```

In [42]:

```
# In this line respecting the column "Rating" there are misplaced but valid data

# Data will be fixed manually, putting them in the correct position

app[app["Rating"]== "SOCIAL"]
```

Out[42]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	Last Updated
165230	Shytter - Twitter client	not notified you follow -	SOCIAL	4.098591328	71	5,000+	7.7M	0	Everyone

In [43]:

```
# Fixing the data entry positions manually

app.loc[165230, ["Category", "Rating", "Reviews", "Installs", "Size", "Price", "Content Rating", "Last Updated", "Minimum Version", "Latest Version"]] = "SOCIAL", "4.098591328", "71", "5,000+", "7.7M", "0", "Everyone", "March 30, 2019","4.1 and up", "NaN"
```

In [44]:

```
app[app["Rating"]== "SOCIAL"]
```

Out[44]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	Last Updated	Minimum Version	Lat Vers

In [ ]:

```
```

In [45]:

```
app[app["Rating"]== "PRODUCTIVITY"]
```

Out[45]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Conte Ratir
168914	CorreosTrack 2.0 (Correos de M&#65533;xico	Mexpost)	PRODUCTIVITY	4.389830589	59	10,000+	16M	

In [46]:

```
# Fixing the data entry positions manually for the index position 168914



app.loc[168914, ["Category", "Rating", "Reviews", "Installs", "Size", "Price", "Content
Rating", "Last Updated", "Minimum Version", "Latest Version"]] = "PRODUCTIVITY", "4.389
830589", "59", "10,000+", "16M", "0", "Everyone", "December 21, 2018","4.1 and up", "Na
N"
app[app["Rating"]== "PRODUCTIVITY"] # Ensuring that column "Rating" is fixed from this
 kind of data entry
```

Out[46]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	Last Updated	Minimum Version	Lat Vers

In [ ]:

In [ ]:

In [47]:

```
app[app["Rating"]== "MUSIC_AND_AUDIO"]
```

Out[47]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Cont Rat
177165	Music Love Song	Romantic Song Music Love Songs	MUSIC_AND_AUDIO	4.538461685	13	1,000+	Varies with device	
193869	Equalizer & Volume Booster	Speaker Pro 2019	MUSIC_AND_AUDIO	4.632093906	511	10,000+	2.5M	
257773	High Volume Booster	super loud speaker booster	MUSIC_AND_AUDIO	4.400000095	10	1,000+	3.5M	

In [48]:

```
# Same logic here. Misplaced but valid data. They will be edited manually

app.loc[177165, ["Category", "Rating", "Reviews", "Installs", "Size", "Price", "Content
Rating", "Last Updated", "Minimum Version", "Latest Version"]] = "MUSIC_AND_AUDIO", "4.
538461685", "13", "1,000+", "Varies with device", "0", "Teen", "October 24, 2018","Vari
es with device", "NaN"
app.loc[193869, ["Category", "Rating", "Reviews", "Installs", "Size", "Price", "Content
Rating", "Last Updated", "Minimum Version", "Latest Version"]] = "MUSIC_AND_AUDIO", "4.
632093906", "511", "10,000+", "2.5M", "0", "Everyone", "September 25, 2018","2.3 and u
p", "NaN"
app.loc[257773, ["Category", "Rating", "Reviews", "Installs", "Size", "Price", "Content
Rating", "Last Updated", "Minimum Version", "Latest Version"]] = "MUSIC_AND_AUDIO", "4.
400000095", "10", "1,000+", "3.5M", "0", "Everyone", "November 7, 2018","4.0 and up",
"NaN"
app[app["Rating"]== "PRODUCTIVITY"]
```

Out[48]:

App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	Last Updated	Minimum Version	Lat Vers

In [ ]:

```

```

In [49]:

```
app[app["Rating"]== "TRAVEL_AND_LOCAL"]
```

Out[49]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	I
190759	Friend in Iceland	Tour Guide	TRAVEL_AND_LOCAL	5	6	1,000+	27M	0	E

In [50]:

```
# Fixing the entries in index position 190759 manually (misplaced but substantial value
s)

app.loc[190759, ["Category", "Rating", "Reviews", "Installs", "Size", "Price", "Content
Rating", "Last Updated", "Minimum Version", "Latest Version"]] = "TRAVEL_AND_LOCAL",
"5", "6", "1,000+", "27M", "0", "Everyone", "October 16, 2017", "4.0 and up", "NaN"

app[app["Rating"]== "TRAVEL_AND_LOCAL"]
```

Out[50]:

App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	Last Updated	Minimum Version	Lat Vers

In [ ]:

```
```

In [51]:

```
app[app["Rating"]== "LIFESTYLE"]
```

Out[51]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	U
194165	Muslim Baby Names	Islamic Name Boy & Girl+Meaning	LIFESTYLE	4.388349533	927	100,000+	3.7M	0	Ev

In [52]:

```
# same logic as previously

app.loc[194165, ["Category", "Rating", "Reviews", "Installs", "Size", "Price", "Content Rating", "Last Updated", "Minimum Version", "Latest Version"]] = "LIFESTYLE", "4.388349533", "927", "100,000+", "3.7M", "0", "Everyone", "May 23, 2018", "4.0 and up", "NaN"
app[app["Rating"]== "LIFESTYLE"]
```

Out[52]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	Last Updated	Minimum Version	Lat Vers

In [ ]:

```
```

In [53]:

```
app[app["Rating"]== " Economics"]
```

Out[53]:

	App Name	Category	Rating	Reviews	Installs	Size	Pric
232811	Learn Accounts - Finance	Accounting	Economics	BOOKS_AND_REFERENCE	4.823529243	17	1,000

In [54]:

```
# Applying the same logic. Correting the misplaced (but valid) data

app.loc[232811, ["Category", "Rating", "Reviews", "Installs", "Size", "Price", "Content Rating", "Last Updated", "Minimum Version", "Latest Version"]] = " Economics", "4.823529243", "17", "1,000+", "17M", "0", "Everyone", "October 22, 2018", "NaN", "NaN"
app[app["Rating"]== " Economics"]
```

Out[54]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	Last Updated	Minimum Version	Lat Vers

In [55]:

```
# Here we had an entry in column "Rating" which was 7. But we want "Rating<=5".
# It was fixed so now there is no longer rating with numbers more than "5"

app[app["Rating"]==7.000000]
```

Out[55]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	Last Updated	Minimum Version	Lat Vers

In [56]:

```
app.drop(index=99584, inplace=True)
app[app["Rating"]==7.000000]
```

Out[56]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating	Last Updated	Minimum Version	Lat Vers

In [57]:

```
# Converting the column "Rating" to float so that we can apply statistics

app.Rating= app.Rating.astype(float)
```

In [58]:

```
app.describe()
```

Out[58]:

	Rating
count	267040.000000
mean	4.269390
std	0.586244
min	1.000000
25%	4.017699
50%	4.382165
75%	4.648649
max	5.000000

In [59]:

```
# Converting the data in the column "Reviews" to float to that we can apply statistics

app.Reviews= app.Reviews.astype(float)
```

In [60]:

```
app.describe()
```

Out[60]:

	Rating	Reviews
count	267040.000000	2.670400e+05
mean	4.269390	1.459628e+04
std	0.586244	4.110638e+05
min	1.000000	1.000000e+00
25%	4.017699	1.600000e+01
50%	4.382165	9.300000e+01
75%	4.648649	6.560000e+02
max	5.000000	8.621429e+07

In [61]:

```
# I want to convert the column "Installs" into float

# I firstly have to remove the "+"

app.Installs= app["Installs"].str.replace("+", "")
```

In [62]:

```
# I had some problems converting the column into float, even when i removed "+"

# I am removing the commas

app.Installs= app["Installs"].str.replace(",", "")
```

In [63]:

```
app["Installs"] = app.Installs.astype(float)
```

In [64]:

```
# Removing "$" from the data entries in the column "Price" so that it can be converted
 to float

app["Price"]= app["Price"].str.replace("$", "")
```

In [65]:

```
# Convert the data in "Price" to float


app["Price"]= app.Price.astype(float)
```

In [66]:

```
# the data in the column "Prics" successfully converted to float

# In these columns i can do various statistical applications
app.describe()
```

Out[66]:

	Rating	Reviews	Installs	Price
count	267040.000000	2.670400e+05	2.670400e+05	267040.000000
mean	4.269390	1.459628e+04	6.410638e+05	0.227872
std	0.586244	4.110638e+05	2.046801e+07	3.559421
min	1.000000	1.000000e+00	0.000000e+00	0.000000
25%	4.017699	1.600000e+01	1.000000e+03	0.000000
50%	4.382165	9.300000e+01	1.000000e+04	0.000000
75%	4.648649	6.560000e+02	5.000000e+04	0.000000
max	5.000000	8.621429e+07	5.000000e+09	399.990000

In [67]:

```
# procedure for converting the column "Size" to float
# there are sizes counted in mb, kb, in numbers without measurement unit and with "vari
es with device"
app.Size.unique()
```

Out[67]:

```
array(['Varies with device', '1.4M', '16M', ..., '601k', '715k', '311k'],
      dtype=object)
```

In [68]:

```
# removing the "m" which is the mb for the size

app.Size= app["Size"].str.replace("M", "")
```

In [69]:

```
# assigning "Varies with device" with a number like "-1" so that i can seperate it late
r

# app.Size= app["Size"].str.replace("Varies with device", "-1")
```

In [70]:

```
# Segmenting the column of the size

y= app.iloc[:, 5:6]
```

In [71]:

```
# I tried to fix the last problems in converting the column "Size" to float
# Here i am trying to remove "k" (kbs) and the blanks, and to convert kbs to mbs
# It keeps giving me errors and i cannot fix it
# i will not use the column "Size" for statistical applications

#for x in y:
#    x = str(x)
#    x= x.replace(" ", "")
#    x= x.replace(",", ".")
#    if "k" in x:
#        x= x.replace("k", "")
#        x=x.replace(" k", "")
#        x=x.replace("k ", "")
#        x= float(x)
#        x= x/1024
```

In [72]:

```
# There are 11,728 apps whose size varies with device

len(app[app["Size"]== "Varies with device"])
```

Out[72]:

```
11728
```

In [73]:

```
app.Size.describe()
```

Out[73]:

```
count                 267040
unique                  1236
top       Varies with device
freq                   11728
Name: Size, dtype: object
```

In [74]:

```
print ("Apps whose size varies with device are {}% of the dataset".format(11728/267040*
100))
```

```
Apps whose size varies with device are 4.391851408028759% of the dataset
```

In [75]:

```
##########################################################################################
#################
```

In [76]:

```
##########################################################################################
#################
```

In [77]:

```
##########################################################################################
#################
```

Statistical Analysis

In [78]:

```
#ensuring the shape of dataframe before proceeding to further statistics and visualizat
ion

app.shape
```

Out[78]:

```
(267040, 11)
```

In [79]:

```
# the columns, the data of which we can do statistic manipulation

app.describe()
```

Out[79]:

	Rating	Reviews	Installs	Price
count	267040.000000	2.670400e+05	2.670400e+05	267040.000000
mean	4.269390	1.459628e+04	6.410638e+05	0.227872
std	0.586244	4.110638e+05	2.046801e+07	3.559421
min	1.000000	1.000000e+00	0.000000e+00	0.000000
25%	4.017699	1.600000e+01	1.000000e+03	0.000000
50%	4.382165	9.300000e+01	1.000000e+04	0.000000
75%	4.648649	6.560000e+02	5.000000e+04	0.000000
max	5.000000	8.621429e+07	5.000000e+09	399.990000

In [ ]:

```

```

In [80]:

```
app.info() # data type for each column
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 267040 entries, 0 to 267051
Data columns (total 11 columns):
App Name            267040 non-null object
Category            267040 non-null object
Rating              267040 non-null float64
Reviews             267040 non-null float64
Installs            267040 non-null float64
Size                267040 non-null object
Price               267040 non-null float64
Content Rating      267040 non-null object
Last Updated        267040 non-null object
Minimum Version     267040 non-null object
Latest Version      267040 non-null object
dtypes: float64(4), object(7)
memory usage: 24.4+ MB
```

In [81]:

```
#reinsuring there are no any missing values
app.isnull().any().any()
```

Out[81]:

```
False
```

In [82]:

```
#*********************************************************************************
***************
# Reviewing the unique values and the number of unique values in each column after the
 cleaning process
```

In [83]:

```
# Values in "Category"
app["Category"].unique()
```

Out[83]:

```
array(['FOOD_AND_DRINK', 'TRAVEL_AND_LOCAL', 'SHOPPING', 'LIFESTYLE',
       'GAME_ACTION', 'GAME_CASUAL', 'GAME_ROLE_PLAYING', 'GAME_PUZZLE',
       'GAME_RACING', 'GAME_ADVENTURE', 'GAME_ARCADE', 'GAME_STRATEGY',
       'GAME_SPORTS', 'GAME_SIMULATION', 'GAME_MUSIC', 'MUSIC_AND_AUDIO',
       'FINANCE', 'EVENTS', 'ENTERTAINMENT', 'EDUCATION',
       'GAME_EDUCATIONAL', 'BOOKS_AND_REFERENCE', 'NEWS_AND_MAGAZINES',
       'PHOTOGRAPHY', 'VIDEO_PLAYERS', 'GAME_WORD', 'ART_AND_DESIGN',
       'GAME_TRIVIA', 'GAME_BOARD', 'BUSINESS', 'PRODUCTIVITY',
       'COMMUNICATION', 'HEALTH_AND_FITNESS', 'HOUSE_AND_HOME', 'SOCIAL',
       'BEAUTY', 'GAME_CASINO', 'MAPS_AND_NAVIGATION', 'PERSONALIZATION',
       'GAME_CARD', 'TOOLS', 'SPORTS', 'AUTO_AND_VEHICLES',
       'LIBRARIES_AND_DEMO', 'COMICS', 'PARENTING', 'DATING', 'WEATHER',
       'MEDICAL', 'TRAVEL', ' Economics'], dtype=object)
```

In [84]:

```
# There are 51 different categories

app["Category"].nunique()
```

Out[84]:

```
51
```

In [85]:

```
# Unique values of Rating
app["Rating"].unique()
```

Out[85]:

```
array([4.54856157, 4.40067148, 3.65632939, ..., 4.06342983, 3.9687779 ,
       4.60038614])
```

In [86]:

```
# There are 99,845 unique values of Rating

app["Rating"].nunique()
```

Out[86]:

```
99845
```

In [87]:

```
# Unique values of the column "Reviews"
app["Reviews"].unique()
```

Out[87]:

```
array([ 305034., 1207922.,    1967., ...,  296774.,    7974.,   69123.])
```

In [88]:

```
# There are 24,531 different reviews

app["Reviews"].nunique()
```

Out[88]:

```
24531
```

In [89]:

```
# Unique values of installations

app["Installs"].unique()
```

Out[89]:

```
array([5.e+06, 1.e+08, 1.e+05, 1.e+07, 1.e+04, 1.e+06, 5.e+07, 5.e+05,
       5.e+04, 5.e+03, 1.e+03, 5.e+08, 1.e+09, 5.e+09, 1.e+02, 5.e+02,
       5.e+01, 5.e+00, 1.e+01, 1.e+00, 0.e+00])
```

In [90]:

```
# There are 21 different classes of installations

app["Installs"].nunique()
```

Out[90]:

```
21
```

In [91]:

```
# Unique values in the column "Size"
app["Size"].unique()
```

Out[91]:

```
array(['Varies with device', '1.4', '16', ..., '601k', '715k', '311k'],
      dtype=object)
```

In [92]:

```
# There are 1,236 different sizes for the apps

app["Size"].nunique()
```

Out[92]:

```
1236
```

In [93]:

```
# There are 488 different prices
app["Price"].nunique()
```

Out[93]:

```
488
```

In [94]:

```
# Unique values of the column "Content Rating"
app["Content Rating"].unique()
```

Out[94]:

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

In [95]:

```
# There are 6 different content ratings

len(app["Content Rating"].unique())
```

Out[95]:

```
6
```

In [96]:

```
print("**************************************************")

print("Minimum number of ratings: %.2f" %app["Rating"].min())
print("Maximum number of ratings: %.2f" %app["Rating"].max())

print("**************************************************")

print("Minimum number of reviews: %.2f" %app["Reviews"].min())
print("Maximum number of reviews: %.2f" %app["Reviews"].max())

print("**************************************************")

print("Minimum number of installs: %.2f" %app["Installs"].min())
print("Maximum number of installs: %.2f" %app["Installs"].max())

print("**************************************************")


print("Minimum number of prices: %.2f" %app["Price"].min())
print("Maximum number of prices: %.2f" %app["Price"].max())

print("**************************************************")
```

```
**************************************************
Minimum number of ratings: 1.00
Maximum number of ratings: 5.00
**************************************************
Minimum number of reviews: 1.00
Maximum number of reviews: 86214292.00
**************************************************
Minimum number of installs: 0.00
Maximum number of installs: 5000000000.00
**************************************************
Minimum number of prices: 0.00
Maximum number of prices: 399.99
**************************************************
```

In [97]:

```
# Getting the measures of central tendency for all the installation grouped by "Categor
y"

app.groupby("Category").Installs.agg(["min", "mean", "median", "max"])
```

Out[97]:

Category	min	mean	median	max
Economics	1000.0	1.000000e+03	1000.0	1.000000e+03
ART_AND_DESIGN	1.0	2.105946e+05	5000.0	1.000000e+08
AUTO_AND_VEHICLES	1.0	1.824824e+05	5000.0	1.000000e+07
BEAUTY	0.0	1.409585e+05	10000.0	1.000000e+07
BOOKS_AND_REFERENCE	0.0	1.008366e+05	10000.0	1.000000e+08
BUSINESS	0.0	1.455845e+05	1000.0	1.000000e+08
COMICS	5.0	3.832805e+05	50000.0	1.000000e+07
COMMUNICATION	0.0	2.419759e+06	10000.0	1.000000e+09
DATING	1.0	4.595702e+05	10000.0	1.000000e+07
EDUCATION	0.0	7.466526e+04	5000.0	1.000000e+08
ENTERTAINMENT	0.0	3.573559e+05	10000.0	1.000000e+09
EVENTS	1.0	4.602683e+04	1000.0	1.000000e+07
FINANCE	0.0	2.236191e+05	10000.0	1.000000e+08
FOOD_AND_DRINK	1.0	1.930530e+05	5000.0	5.000000e+07
GAME_ACTION	0.0	4.063549e+06	100000.0	5.000000e+08
GAME_ADVENTURE	1.0	1.155658e+06	10000.0	1.000000e+08
GAME_ARCADE	1.0	2.960377e+06	10000.0	1.000000e+09
GAME_BOARD	5.0	1.159017e+06	10000.0	1.000000e+08
GAME_CARD	1.0	8.000250e+05	100000.0	1.000000e+08
GAME_CASINO	5.0	1.769368e+06	100000.0	5.000000e+07
GAME_CASUAL	0.0	2.696059e+06	100000.0	5.000000e+08
GAME_EDUCATIONAL	0.0	6.160472e+05	10000.0	1.000000e+08
GAME_MUSIC	5.0	2.268880e+06	100000.0	1.000000e+08
GAME_PUZZLE	0.0	1.184301e+06	10000.0	1.000000e+08
GAME_RACING	1.0	5.540269e+06	500000.0	5.000000e+08
GAME_ROLE_PLAYING	10.0	1.139481e+06	100000.0	5.000000e+07
GAME_SIMULATION	5.0	1.807155e+06	100000.0	1.000000e+08
GAME_SPORTS	10.0	3.580514e+06	100000.0	1.000000e+08
GAME_STRATEGY	1.0	2.544039e+06	100000.0	5.000000e+08
GAME_TRIVIA	1.0	6.068273e+05	10000.0	1.000000e+08
GAME_WORD	0.0	1.046894e+06	100000.0	5.000000e+07
HEALTH_AND_FITNESS	0.0	2.837032e+05	10000.0	5.000000e+08
HOUSE_AND_HOME	1.0	3.353328e+05	5000.0	1.000000e+08
LIBRARIES_AND_DEMO	1.0	1.630496e+05	5000.0	1.000000e+07
LIFESTYLE	0.0	1.463108e+05	10000.0	1.000000e+08
MAPS_AND_NAVIGATION	1.0	2.986137e+05	10000.0	1.000000e+08

Category	min	mean	median	max
MEDICAL	1.0	5.832896e+04	5000.0	5.000000e+06
MUSIC_AND_AUDIO	0.0	3.264714e+05	10000.0	1.000000e+09
NEWS_AND_MAGAZINES	0.0	4.623790e+05	10000.0	1.000000e+09
PARENTING	5.0	1.447583e+05	10000.0	1.000000e+07
PERSONALIZATION	0.0	3.970350e+05	10000.0	1.000000e+08
PHOTOGRAPHY	0.0	1.292555e+06	50000.0	1.000000e+09
PRODUCTIVITY	0.0	1.391869e+06	10000.0	1.000000e+09
SHOPPING	0.0	6.134706e+05	10000.0	1.000000e+08
SOCIAL	1.0	1.490510e+06	10000.0	1.000000e+09
SPORTS	1.0	1.497417e+05	10000.0	5.000000e+07
TOOLS	0.0	1.104303e+06	10000.0	5.000000e+09
TRAVEL	10000.0	1.000000e+04	10000.0	1.000000e+04
TRAVEL_AND_LOCAL	0.0	1.222950e+06	10000.0	5.000000e+09
VIDEO_PLAYERS	1.0	3.554788e+06	10000.0	5.000000e+09
WEATHER	5.0	5.188793e+05	10000.0	1.000000e+08

In [98]:

```
# Sorting (descending sorting) the dataframe by number of installs

app.sort_values(by="Installs", ascending= False)
```

Out[98]:

	App Name	Category	Rating	Reviews	Installs	Size	Price
813	YouTube	VIDEO_PLAYERS	4.368428	41919102.0	5.000000e+09	Varies with device	0.00
2177	Maps - Navigate & Explore	TRAVEL_AND_LOCAL	4.342798	10083666.0	5.000000e+09	Varies with device	0.00
821	Google	TOOLS	4.408893	10870728.0	5.000000e+09	Varies with device	0.00
842	Google Chrome: Fast & Secure	COMMUNICATION	4.335205	13636591.0	1.000000e+09	Varies with device	0.00
539	Subway Surfers	GAME_ARCADE	4.498131	29834812.0	1.000000e+09	85	0.00
...	...	...	...	...	...	...	...
253182	La Barbiera Sasso Marconi	BEAUTY	5.000000	2.0	0.000000e+00	10	0.00
249287	Rich & Rich	LIFESTYLE	5.000000	1.0	0.000000e+00	16	369.99
24784	DUA KE QURAN AMHARIC	EDUCATION	4.730337	89.0	0.000000e+00	7.9	0.00
166479	??? LED ??????	ENTERTAINMENT	1.941176	17.0	0.000000e+00	2.4	0.00
260161	Flugpreise Vergleichen & G�nstige Fl�ge Low Cost	TRAVEL_AND_LOCAL	5.000000	1.0	0.000000e+00	7.9	0.00

267040 rows × 11 columns

In [99]:

```
top_installed_apps=app.sort_values(by="Installs", ascending= False)
```

In [100]:

```
#************************************************************

# top 10 apps based on the number of installations

#************************************************************

top_installed_apps.head(10)
```

Out[100]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	C
813	YouTube	VIDEO_PLAYERS	4.368428	41919102.0	5.000000e+09	Varies with device	0.0	
2177	Maps - Navigate & Explore	TRAVEL_AND_LOCAL	4.342798	10083666.0	5.000000e+09	Varies with device	0.0	Ev
821	Google	TOOLS	4.408893	10870728.0	5.000000e+09	Varies with device	0.0	Ev
842	Google Chrome: Fast & Secure	COMMUNICATION	4.335205	13636591.0	1.000000e+09	Varies with device	0.0	Ev
539	Subway Surfers	GAME_ARCADE	4.498131	29834812.0	1.000000e+09	85	0.0	Ev
2147	Google Street View	TRAVEL_AND_LOCAL	4.215697	2171998.0	1.000000e+09	Varies with device	0.0	Ev
28676	Samsung Print Service Plugin	PRODUCTIVITY	4.204499	322275.0	1.000000e+09	Varies with device	0.0	Ev
704	Facebook	SOCIAL	4.087946	85766433.0	1.000000e+09	Varies with device	0.0	
6411	Google Duo - High Quality Video Calls	COMMUNICATION	4.596404	3641252.0	1.000000e+09	20	0.0	Ev
6412	Google Play Movies & TV	VIDEO_PLAYERS	3.703356	1048972.0	1.000000e+09	Varies with device	0.0	

In [101]:

```
# Apps with 5 billion installations (5b is the 1st greater class of installations in th
e dataset)

len(app[app["Installs"]>= 5000000000])
```

Out[101]:

```
3
```

In [102]:

```
# Apps with more than 1 billion installations (1b is the 2nd greater class of installat
ions in the dataset)

len(app[app["Installs"]>= 1000000000])
```

Out[102]:

```
27
```

In [ ]:

```

```

In [103]:

```
top_installed_and_rated_apps = app.sort_values(by=["Installs", "Rating"], ascending=False)
top_installed_and_rated_apps # main top apps
```

Out[103]:

	App Name	Category	Rating	Reviews	Installs	Size	Pric
821	Google	TOOLS	4.408893	10870728.0	5.000000e+09	Varies with device	0
813	YouTube	VIDEO_PLAYERS	4.368428	41919102.0	5.000000e+09	Varies with device	0
2177	Maps - Navigate & Explore	TRAVEL_AND_LOCAL	4.342798	10083666.0	5.000000e+09	Varies with device	0
7064	Clean Master - Antivirus, Applock & Cleaner	TOOLS	4.657038	44171776.0	1.000000e+09	20	0
6411	Google Duo - High Quality Video Calls	COMMUNICATION	4.596404	3641252.0	1.000000e+09	20	0
...	...	...	...	...	...	...	...
226376	Say My Text (Speech synthesizer)	TOOLS	3.000000	2.0	0.000000e+00	5.4	0
39673	Vadaa Hunt	GAME_CASUAL	2.333333	3.0	0.000000e+00	20	0
225594	PIP Collage Maker Professional	PHOTOGRAPHY	2.130435	46.0	0.000000e+00	8.3	0
166479	??? LED ??????	ENTERTAINMENT	1.941176	17.0	0.000000e+00	2.4	0
184817	Diabetes Ratgeber AR	HEALTH_AND_FITNESS	1.250000	4.0	0.000000e+00	60	0

267040 rows × 11 columns

In [104]:

```
#*************************************************************************
# top 10 apps based on the number of installations and rating together (main top apps)
#*************************************************************************



top_installed_and_rated_apps.head(10)
```

Out[104]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	C
821	Google	TOOLS	4.408893	10870728.0	5.000000e+09	Varies with device	0.0	Eve
813	YouTube	VIDEO_PLAYERS	4.368428	41919102.0	5.000000e+09	Varies with device	0.0	
2177	Maps - Navigate & Explore	TRAVEL_AND_LOCAL	4.342798	10083666.0	5.000000e+09	Varies with device	0.0	Eve
7064	Clean Master - Antivirus, Applock & Cleaner	TOOLS	4.657038	44171776.0	1.000000e+09	20	0.0	Eve
6411	Google Duo - High Quality Video Calls	COMMUNICATION	4.596404	3641252.0	1.000000e+09	20	0.0	Eve
3269	SHAREit - Transfer & Share	TOOLS	4.579340	10450444.0	1.000000e+09	20	0.0	Eve
1259	Google Photos	PHOTOGRAPHY	4.542380	16278468.0	1.000000e+09	Varies with device	0.0	Eve
653	Instagram	SOCIAL	4.519560	79726403.0	1.000000e+09	Varies with device	0.0	
539	Subway Surfers	GAME_ARCADE	4.498131	29834812.0	1.000000e+09	85	0.0	Eve
3267	Samsung Internet Browser	COMMUNICATION	4.424015	832714.0	1.000000e+09	Varies with device	0.0	Eve

In [105]:

```
top_installed_and_reviewed_apps = app.sort_values(by=["Installs", "Reviews"], ascending
=False)
top_installed_and_reviewed_apps
```

Out[105]:

	App Name	Category	Rating	Reviews	Installs	Size
813	YouTube	VIDEO_PLAYERS	4.368428	41919102.0	5.000000e+09	Varies with device
821	Google	TOOLS	4.408893	10870728.0	5.000000e+09	Varies with device
2177	Maps - Navigate & Explore	TRAVEL_AND_LOCAL	4.342798	10083666.0	5.000000e+09	Varies with device
671	WhatsApp Messenger	COMMUNICATION	4.417610	86214292.0	1.000000e+09	Varies with device
704	Facebook	SOCIAL	4.087946	85766433.0	1.000000e+09	Varies with device
...	...	...	...	...	...	...
249287	Rich & Rich	LIFESTYLE	5.000000	1.0	0.000000e+00	16
251001	Tamil Dictionary Offline & Multilingual Transl...	BOOKS_AND_REFERENCE	5.000000	1.0	0.000000e+00	11
260161	Flugpreise Vergleichen & G�nstige Fl�ge Low Cost	TRAVEL_AND_LOCAL	5.000000	1.0	0.000000e+00	7.9
264007	Fm Resplandecer Misiones	MUSIC_AND_AUDIO	5.000000	1.0	0.000000e+00	2.1
265585	DEUTSCH WITZE 2019	COMMUNICATION	5.000000	1.0	0.000000e+00	4.3

267040 rows × 11 columns

In [106]:

```
#*************************************************************************
# top 10 apps based on the number of installations and reviews together
#*************************************************************************



top_installed_and_reviewed_apps.head(10)
```

Out[106]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	C
813	YouTube	VIDEO_PLAYERS	4.368428	41919102.0	5.000000e+09	Varies with device	0.0	
821	Google	TOOLS	4.408893	10870728.0	5.000000e+09	Varies with device	0.0	E\
2177	Maps - Navigate & Explore	TRAVEL_AND_LOCAL	4.342798	10083666.0	5.000000e+09	Varies with device	0.0	E\
671	WhatsApp Messenger	COMMUNICATION	4.417610	86214292.0	1.000000e+09	Varies with device	0.0	E\
704	Facebook	SOCIAL	4.087946	85766433.0	1.000000e+09	Varies with device	0.0	
653	Instagram	SOCIAL	4.519560	79726403.0	1.000000e+09	Varies with device	0.0	
632	Messenger – Text and Video Chat for Free	COMMUNICATION	4.085856	65469531.0	1.000000e+09	Varies with device	0.0	E\
7064	Clean Master - Antivirus, Applock & Cleaner	TOOLS	4.657038	44171776.0	1.000000e+09	20	0.0	E\
539	Subway Surfers	GAME_ARCADE	4.498131	29834812.0	1.000000e+09	85	0.0	E\
1259	Google Photos	PHOTOGRAPHY	4.542380	16278468.0	1.000000e+09	Varies with device	0.0	E\

In [107]:

```
top_10_installed_and_rated_apps= top_installed_and_rated_apps.head(10)
```

In [108]:

```
top_10_installed_and_rated_apps.Category.sort_values(ascending=False)
```

Out[108]:

```
813        VIDEO_PLAYERS
2177    TRAVEL_AND_LOCAL
3269               TOOLS
7064               TOOLS
821                TOOLS
653               SOCIAL
1259         PHOTOGRAPHY
539          GAME_ARCADE
3267       COMMUNICATION
6411       COMMUNICATION
Name: Category, dtype: object
```

In [109]:

```
# There are totally 244,396 apps
app["App Name"].nunique()
```

Out[109]:

```
244396
```

In [110]:

```
# here i will see the number of apps which belong to the categories of the most install
ed and rated apps


count_VIDEO_PLAYERS=0
count_TRAVEL_AND_LOCAL=0
count_TOOLS=0
count_SOCIAL=0
count_PHOTOGRAPHY=0
count_GAME_ARCADE=0
count_COMMUNICATION=0

for x in app["Category"]:
    if x== "VIDEO_PLAYERS":
        count_VIDEO_PLAYERS=count_VIDEO_PLAYERS+1
    elif x== "TRAVEL_AND_LOCAL":
        count_TRAVEL_AND_LOCAL= count_TRAVEL_AND_LOCAL+1
    elif x== "TOOLS":
        count_TOOLS= count_TOOLS+1
    elif x== "SOCIAL":
        count_SOCIAL= count_SOCIAL+1
    elif x== "PHOTOGRAPHY":
        count_PHOTOGRAPHY= count_PHOTOGRAPHY+1
    elif x== "GAME_ARCADE":
        count_GAME_ARCADE= count_GAME_ARCADE+1
    elif x== "COMMUNICATION":
        count_COMMUNICATION= count_COMMUNICATION+1
print ("**********************************************************************
**********")
print ("**********************************************************************
**********")
print ("Number of apps that belong in category: \"Video Players\" is: {}".format(count_
VIDEO_PLAYERS))
print ("**********************************************************************
**********")
print ("Number of apps that belong in category: \"Travel and Local\" is: {}".format(cou
nt_TRAVEL_AND_LOCAL))
print ("**********************************************************************
**********")
print ("Number of apps that belong in category: \"Tools\" is: {}".format(count_TOOLS))
print ("**********************************************************************
**********")
print ("Number of apps that belong in category: \"Social\" is: {}".format(count_SOCIAL
))
print ("**********************************************************************
**********")
print ("Number of apps that belong in category: \"Photography\" is: {}".format(count_PH
OTOGRAPHY))
print ("**********************************************************************
**********")
print ("Number of apps that belong in category: \"Game Arcade\" is: {}".format(count_GA
ME_ARCADE))
print ("**********************************************************************
**********")
print ("Number of apps that belong in category: \"Communication\" is: {}".format(count_
COMMUNICATION))
print ("**********************************************************************
**********")
```

```
print ("**********************************************************************************
**********************")
```

```
**************************************************************************
**************************
**************************************************************************
**************************
Number of apps that belong in category: "Video Players" is: 2717
**************************************************************************
**************************
Number of apps that belong in category: "Travel and Local" is: 6650
**************************************************************************
**************************
Number of apps that belong in category: "Tools" is: 21591
**************************************************************************
**************************
Number of apps that belong in category: "Social" is: 4745
**************************************************************************
**************************
Number of apps that belong in category: "Photography" is: 7240
**************************************************************************
**************************
Number of apps that belong in category: "Game Arcade" is: 2343
**************************************************************************
**************************
Number of apps that belong in category: "Communication" is: 5486
**************************************************************************
**************************
**************************************************************************
**************************
```

In [111]:

```
top_10_installed_and_rated_apps["Content Rating"].sort_values(ascending=False)
```

Out[111]:

```
653            Teen
813            Teen
539     Everyone 10+
3267        Everyone
1259        Everyone
3269        Everyone
6411        Everyone
7064        Everyone
2177        Everyone
821         Everyone
Name: Content Rating, dtype: object
```

In [112]:

```
app["Content Rating"].nunique()
```

Out[112]:

```
6
```

In [113]:

```
# There are totally 6 categories of content rating

# In the top 10 installed and rated apps, there are 3 different content ratings

# I will now see their performance in the whole dataset, along with the other 3 remaini
ng content ratings in the whole dataset

count_Teen=0
count_Everyone_10 = 0
count_Everyone=0

count_Mature_17=0
count_Adults_only=0
count_Unrated=0


for x in app["Content Rating"]:
    if x== "Teen":
        count_Teen= count_Teen+1
    elif x== "Everyone 10+":
        count_Everyone_10= count_Everyone_10+1
    elif x== "Everyone":
        count_Everyone= count_Everyone+1
    elif x== "Mature 17+":
        count_Mature_17 = count_Mature_17+1
    elif x== "Adults only 18+":
        count_Adults_only= count_Adults_only+1
    elif x== "Unrated":
        count_Unrated= count_Unrated+1
print ("**********************************************************************
**********************")
print ("Number of apps of all the dataset, having the content rating which belong the t
op apps:")
print ("*")
print ("*")
print ("Number of apps that belong to the content rating \"Teen\" is: {}".format(count_
Teen))
print ("**********************************************************************
**********************")
print ("Number of apps that belong to the content rating \"Everyone 10+\" is: {}".forma
t(count_Everyone_10))
print ("**********************************************************************
**********************")
print ("Number of apps that belong to the content rating \"Everyone\" is: {}".format(co
unt_Everyone))
print ("**********************************************************************
**********************")
print ("**********************************************************************
**********************")
print ("##########################################################################
######################")
print ("Number of apps having content rating not included in the top apps")
print ("*")
print ("*")
print ("Number of apps that belong to the content rating \"Mature 17+\" is: {}".format(
count_Mature_17))
print ("Number of apps that belong to the content rating \"Adults only 18+\" is: {}".fo
rmat(count_Adults_only))
```

```
print ("Number of apps that belong to the content rating \"Unrated\" is: {}".format(cou
nt_Unrated))
```

```
**************************************************************************
*************************
Number of apps of all the dataset, having the content rating which belong
the top apps:
*
*
Number of apps that belong to the content rating "Teen" is: 17263
**************************************************************************
*************************
Number of apps that belong to the content rating "Everyone 10+" is: 4661
**************************************************************************
*************************
Number of apps that belong to the content rating "Everyone" is: 241582
**************************************************************************
*************************
**************************************************************************
*************************
##########################################################################
############################
Number of apps having content rating not included in the top apps
*
*
Number of apps that belong to the content rating "Mature 17+" is: 3489
Number of apps that belong to the content rating "Adults only 18+" is: 12
Number of apps that belong to the content rating "Unrated" is: 33
```

In [114]:

```
# The aforementioned can be found more easily with the below command

app["Content Rating"].value_counts(ascending=False)
```

Out[114]:

```
Everyone           241582
Teen                17263
Everyone 10+         4661
Mature 17+           3489
Unrated                33
Adults only 18+        12
Name: Content Rating, dtype: int64
```

In [115]:

```
# In this and in the next 2 commands, i will try to see if there is any correlation bet
ween installations, Rating and Reviews

top_10_installed_and_rated_apps
```

Out[115]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	C
821	Google	TOOLS	4.408893	10870728.0	5.000000e+09	Varies with device	0.0	Eve
813	YouTube	VIDEO_PLAYERS	4.368428	41919102.0	5.000000e+09	Varies with device	0.0	
2177	Maps - Navigate & Explore	TRAVEL_AND_LOCAL	4.342798	10083666.0	5.000000e+09	Varies with device	0.0	Eve
7064	Clean Master - Antivirus, Applock & Cleaner	TOOLS	4.657038	44171776.0	1.000000e+09	20	0.0	Eve
6411	Google Duo - High Quality Video Calls	COMMUNICATION	4.596404	3641252.0	1.000000e+09	20	0.0	Eve
3269	SHAREit - Transfer & Share	TOOLS	4.579340	10450444.0	1.000000e+09	20	0.0	Eve
1259	Google Photos	PHOTOGRAPHY	4.542380	16278468.0	1.000000e+09	Varies with device	0.0	Eve
653	Instagram	SOCIAL	4.519560	79726403.0	1.000000e+09	Varies with device	0.0	
539	Subway Surfers	GAME_ARCADE	4.498131	29834812.0	1.000000e+09	85	0.0	Eve
3267	Samsung Internet Browser	COMMUNICATION	4.424015	832714.0	1.000000e+09	Varies with device	0.0	Eve

In [116]:

```
#*****************************************************************************************
*************************
# It seems that none of the best rated apps belong to the top installed (filtered by ra
ting too) apps
#*****************************************************************************************
*************************
app.sort_values(by="Rating", ascending= False).head(10)
```

Out[116]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Content Rating
194506	F-pics	GAME_CASUAL	5.0	3.0	500.0	7.8	0.0	Everyone
193239	Smartler Smart Home	BUSINESS	5.0	2.0	100.0	19	0.0	Everyone
108020	NHK World News Reader - Chinese version	NEWS_AND_MAGAZINES	5.0	3.0	100.0	2.6	0.0	Everyone
193255	UNE Safe	EDUCATION	5.0	5.0	500.0	15	0.0	Everyone
193254	VandySafe	EDUCATION	5.0	1.0	500.0	24	0.0	Everyone
193253	TigerSafe	EDUCATION	5.0	1.0	500.0	16	0.0	Everyone
193249	keypad lock screen 2019	TOOLS	5.0	3.0	100.0	9.8	0.0	Everyone
38405	Corrigo	BUSINESS	5.0	5.0	5000.0	12	0.0	Everyone
38416	S�ivu	GAME_ACTION	5.0	1.0	50.0	65	0.0	Everyone
193014	Vadii	ENTERTAINMENT	5.0	10.0	500.0	4.3	0.0	Everyone

In [117]:

```
#*****************************************************************************************
************************

# Relationship between Reviews and main top apps

# It seems that Instagram, Clean Master - Antivirus, Youtube and Subway Surfers

# The above 4 apps belong to the top installed (filtered by rating too) and simultaneou
sly to the top reviewed apps

# So there is correlation of 4 out of 10 apps respecting top installed-rated apps and t
op reviewed apps
#*****************************************************************************************
************************

app.sort_values(by="Reviews", ascending= False).head(10)
```

Out[117]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	Co R
671	WhatsApp Messenger	COMMUNICATION	4.417610	86214292.0	1.000000e+09	Varies with device	0.0	Eve
704	Facebook	SOCIAL	4.087946	85766433.0	1.000000e+09	Varies with device	0.0	
653	Instagram	SOCIAL	4.519560	79726403.0	1.000000e+09	Varies with device	0.0	
632	Messenger – Text and Video Chat for Free	COMMUNICATION	4.085856	65469531.0	1.000000e+09	Varies with device	0.0	Eve
628	Clash of Clans	GAME_STRATEGY	4.606215	48401470.0	5.000000e+08	103	0.0	Eve
7064	Clean Master - Antivirus, Applock & Cleaner	TOOLS	4.657038	44171776.0	1.000000e+09	20	0.0	Eve
813	YouTube	VIDEO_PLAYERS	4.368428	41919102.0	5.000000e+09	Varies with device	0.0	
539	Subway Surfers	GAME_ARCADE	4.498131	29834812.0	1.000000e+09	85	0.0	Eve
12638	Security Master - Antivirus, VPN, AppLock, Boo...	TOOLS	4.652842	25532160.0	5.000000e+08	Varies with device	0.0	Eve
1542	Clash Royale	GAME_STRATEGY	4.545474	25449254.0	1.000000e+08	81	0.0	Eve

In [118]:

```
top_10_installed_and_rated_apps
```

Out[118]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	C
821	Google	TOOLS	4.408893	10870728.0	5.000000e+09	Varies with device	0.0	Eve
813	YouTube	VIDEO_PLAYERS	4.368428	41919102.0	5.000000e+09	Varies with device	0.0	
2177	Maps - Navigate & Explore	TRAVEL_AND_LOCAL	4.342798	10083666.0	5.000000e+09	Varies with device	0.0	Eve
7064	Clean Master - Antivirus, Applock & Cleaner	TOOLS	4.657038	44171776.0	1.000000e+09	20	0.0	Eve
6411	Google Duo - High Quality Video Calls	COMMUNICATION	4.596404	3641252.0	1.000000e+09	20	0.0	Eve
3269	SHAREit - Transfer & Share	TOOLS	4.579340	10450444.0	1.000000e+09	20	0.0	Eve
1259	Google Photos	PHOTOGRAPHY	4.542380	16278468.0	1.000000e+09	Varies with device	0.0	Eve
653	Instagram	SOCIAL	4.519560	79726403.0	1.000000e+09	Varies with device	0.0	
539	Subway Surfers	GAME_ARCADE	4.498131	29834812.0	1.000000e+09	85	0.0	Eve
3267	Samsung Internet Browser	COMMUNICATION	4.424015	832714.0	1.000000e+09	Varies with device	0.0	Eve

In [119]:

```
# Prices of the apps

app["Price"].value_counts().sort_values(ascending=False).head(10)
```

Out[119]:

```
0.00    255434
0.99      2317
1.99      1552
2.99      1351
4.99       883
3.99       767
1.49       761
2.49       518
3.49       339
9.99       275
Name: Price, dtype: int64
```

In [120]:

```
app.Price.nunique()
```

Out[120]:

```
488
```

In [121]:

```
##########################################################################################
#################
```

In [122]:

```
##########################################################################################
#################
```

In [123]:

```
##########################################################################################
#################
```

Visualising Data

In [124]:

```
app.head(2)
```

Out[124]:

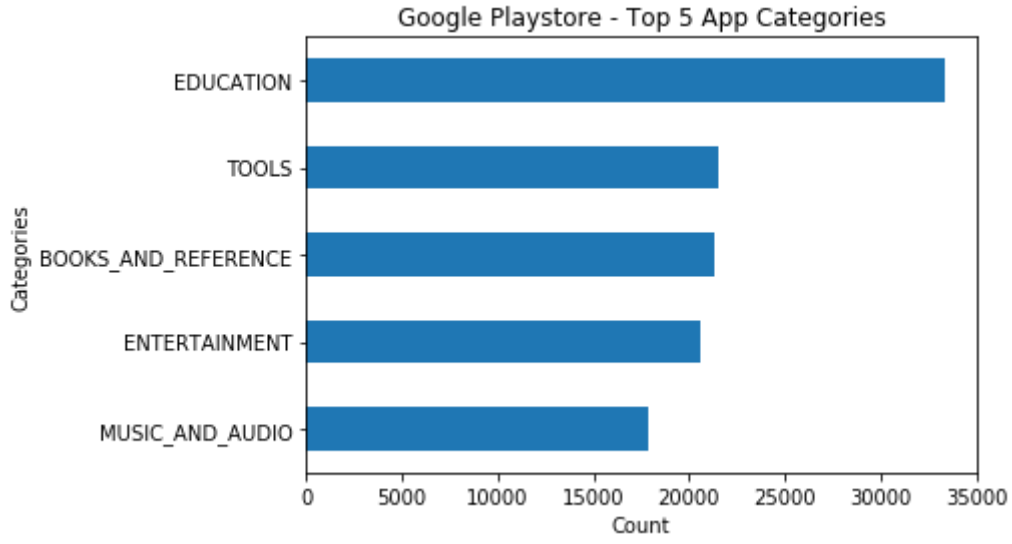
	App Name	Category	Rating	Reviews	Installs	Size	Price	Conte Ratir
0	DoorDash - Food Delivery	FOOD_AND_DRINK	4.548562	305034.0	5000000.0	Varies with device	0.0	Everyor
1	TripAdvisor Hotels Flights Restaurants Attract...	TRAVEL_AND_LOCAL	4.400671	1207922.0	100000000.0	Varies with device	0.0	Everyor

In [125]:

```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [126]:

```
# Top 5 app Categories of all the dataset


app["Category"].value_counts().nlargest(5).sort_values(ascending=True).plot.barh()
plt.ylabel("Categories")
plt.xlabel("Count")
plt.title("Google Playstore - Top 5 App Categories")
plt.show()
```



In [127]:

```
app["Category"].value_counts().nlargest(5).sort_values(ascending=False)
```

Out[127]:

```
EDUCATION              33394
TOOLS                  21591
BOOKS_AND_REFERENCE    21377
ENTERTAINMENT          20603
MUSIC_AND_AUDIO        17879
Name: Category, dtype: int64
```

In [128]:

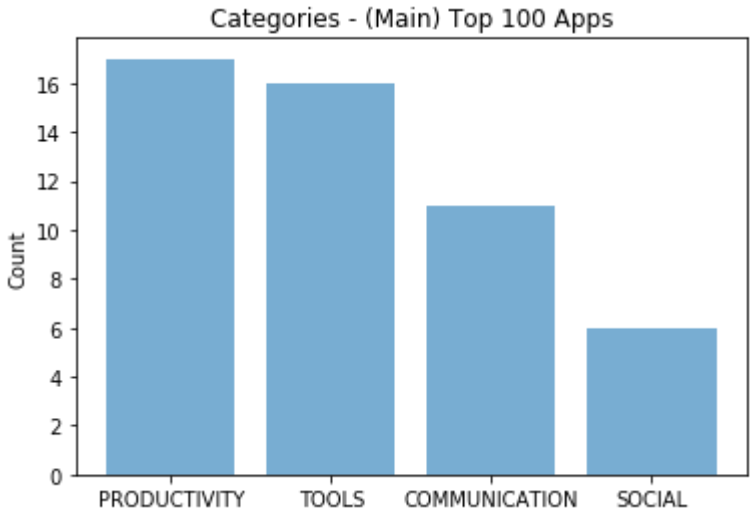
```
# In which category do main 100 top apps belong

top_installed_and_rated_apps["Category"].head(100).value_counts()
```

Out[128]:

```
PRODUCTIVITY           17
TOOLS                  16
COMMUNICATION          11
SOCIAL                  6
VIDEO_PLAYERS           6
GAME_ACTION             4
GAME_CASUAL             4
PHOTOGRAPHY             4
NEWS_AND_MAGAZINES      4
ENTERTAINMENT           3
GAME_RACING             3
PERSONALIZATION         3
TRAVEL_AND_LOCAL        3
GAME_ARCADE             2
MUSIC_AND_AUDIO         2
BOOKS_AND_REFERENCE     2
HEALTH_AND_FITNESS      2
SHOPPING                2
GAME_SPORTS             2
GAME_STRATEGY           1
LIFESTYLE               1
GAME_SIMULATION         1
EDUCATION               1
Name: Category, dtype: int64
```

In [129]:

```
status= ("PRODUCTIVITY", "TOOLS", "COMMUNICATION", "SOCIAL")
y_pos= np.arange(len(status))
numbers= [17,16,11,6]
plt.bar(y_pos, numbers, align="center", alpha=0.6)
plt.xticks(y_pos, status)
plt.ylabel("Count")
plt.title("Categories - (Main) Top 100 Apps")
plt.show()
```

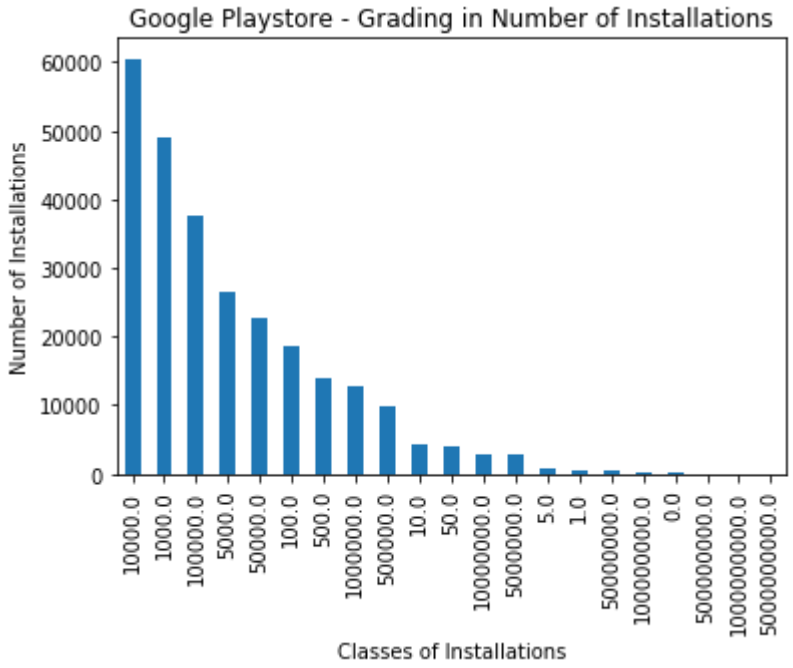


In [130]:

```
x=top_installed_and_rated_apps.head(100)
```

In [131]:

```
# Relationship betweeen: Classes and number of Installations


app["Installs"].value_counts().sort_values(ascending=False).plot.bar()
plt.ylabel("Number of Installations")
plt.xlabel("Classes of Installations")
plt.title("Google Playstore - Grading in Number of Installations")
plt.show()
```



In [132]:

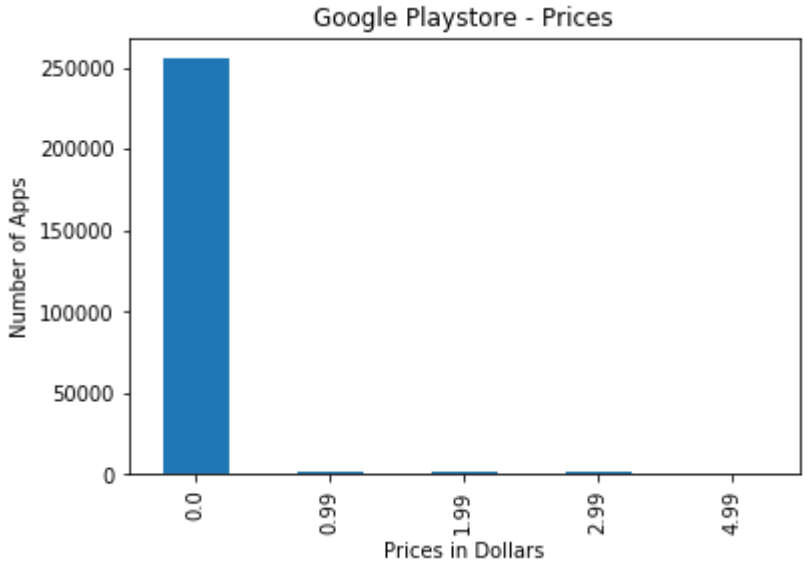
```
# Top 5 Gradings in the number of installations

app["Installs"].value_counts().nlargest(5)
```

Out[132]:

```
10000.0     60533
1000.0      48884
100000.0    37499
5000.0      26361
50000.0     22794
Name: Installs, dtype: int64
```

In [133]:

```
app["Price"].value_counts().nlargest(5).sort_values(ascending=False).plot.bar()
plt.ylabel("Number of Apps")
plt.xlabel("Prices in Dollars")
plt.title("Google Playstore - Prices")
plt.show()
```



In [134]:

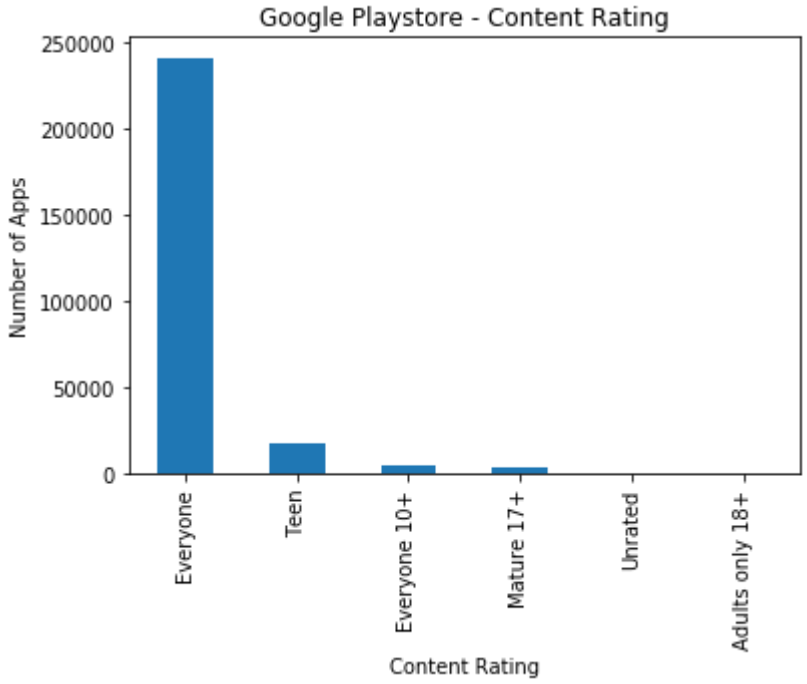
```
app["Price"].value_counts().nlargest(5)
```

Out[134]:

```
0.00    255434
0.99      2317
1.99      1552
2.99      1351
4.99       883
Name: Price, dtype: int64
```

In [135]:

```
app["Content Rating"].value_counts().sort_values(ascending=False).plot.bar()
plt.ylabel("Number of Apps")
plt.xlabel("Content Rating")
plt.title("Google Playstore - Content Rating")
plt.show()
```



In [136]:

```
app["Content Rating"].value_counts()
```

Out[136]:

```
Everyone           241582
Teen                17263
Everyone 10+         4661
Mature 17+           3489
Unrated                33
Adults only 18+        12
Name: Content Rating, dtype: int64
```

In [137]:

```
top_installed_and_rated_apps["Content Rating"].head(100).value_counts()
```

Out[137]:

```
Everyone        69
Teen            24
Everyone 10+     5
Mature 17+       2
Name: Content Rating, dtype: int64
```

In [138]:

```
####################
```

In [139]:

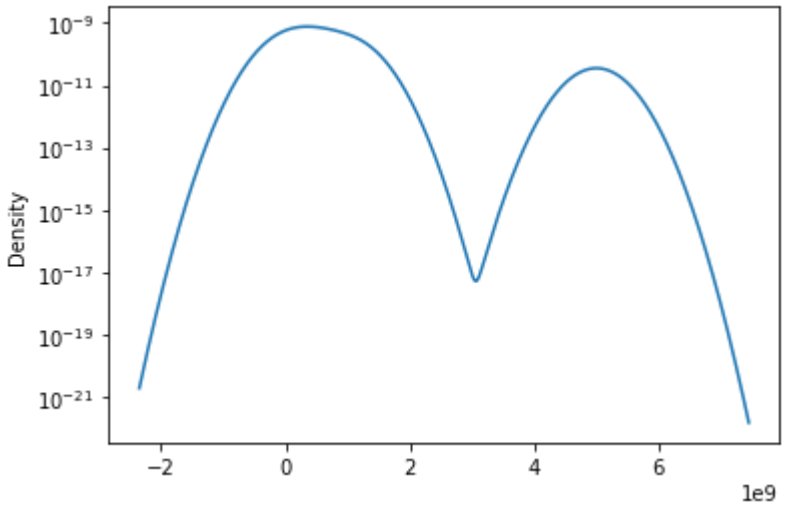
```
top_installed_and_rated_apps.head(100).Installs.value_counts(ascending=False)
```

Out[139]:

```
1.000000e+08    38
5.000000e+08    35
1.000000e+09    24
5.000000e+09     3
Name: Installs, dtype: int64
```

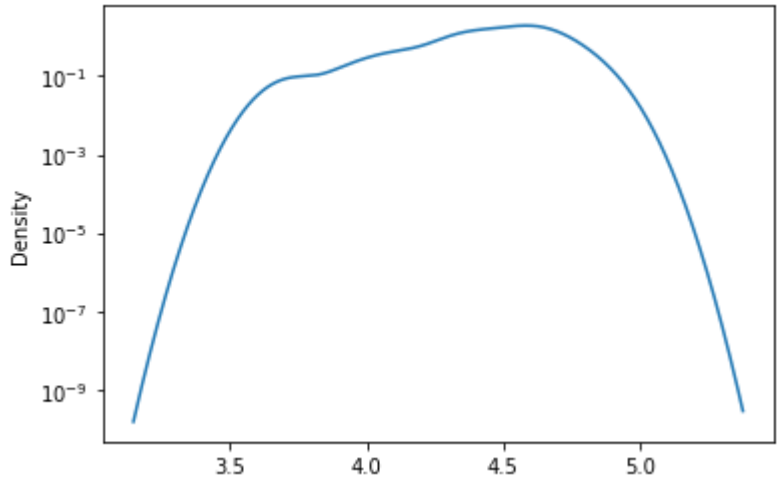
In [140]:

```
app_category= top_installed_and_rated_apps.head(100).Installs
app_category.plot.density().set_yscale("log")
```



In [141]:

```
app_category= top_installed_and_rated_apps.head(100).Rating
app_category.plot.density().set_yscale("log")
```



In [142]:

```
top_installed_and_rated_apps.head(100).Rating.value_counts(ascending=False)
```

Out[142]:

```
4.582568    1
4.703391    1
4.666019    1
4.657038    1
4.463741    1
           ..
4.351907    1
4.491666    1
4.355916    1
4.626627    1
4.652842    1
Name: Rating, Length: 100, dtype: int64
```

In [143]:

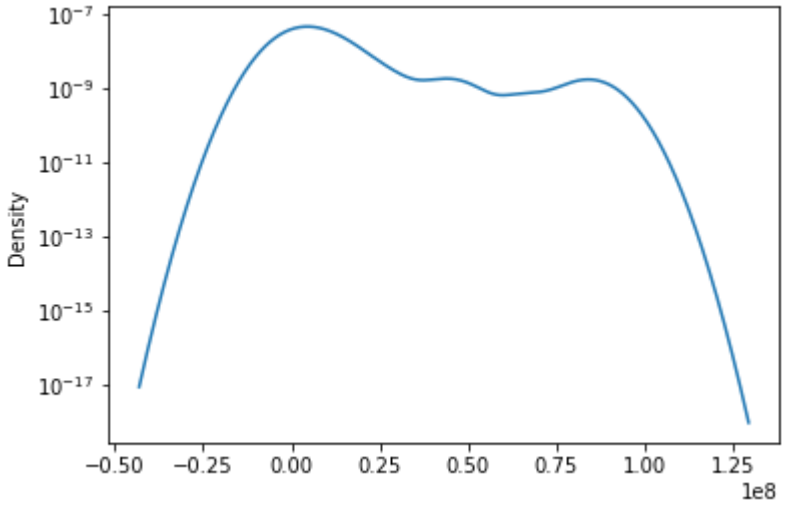
```
top_installed_and_rated_apps.head(100).Reviews.value_counts()
```

Out[143]:

```
1745403.0     1
19573637.0    1
1888392.0     1
2608408.0     1
966728.0      1
             ..
2920141.0     1
1976168.0     1
58506.0       1
99127.0       1
24657922.0    1
Name: Reviews, Length: 100, dtype: int64
```

In [144]:

```
app_category= top_installed_and_rated_apps.head(100).Reviews
app_category.plot.density().set_yscale("log")
```



In [145]:

```
####################
```

In [146]:

```
app["Rating"].value_counts()
```

Out[146]:

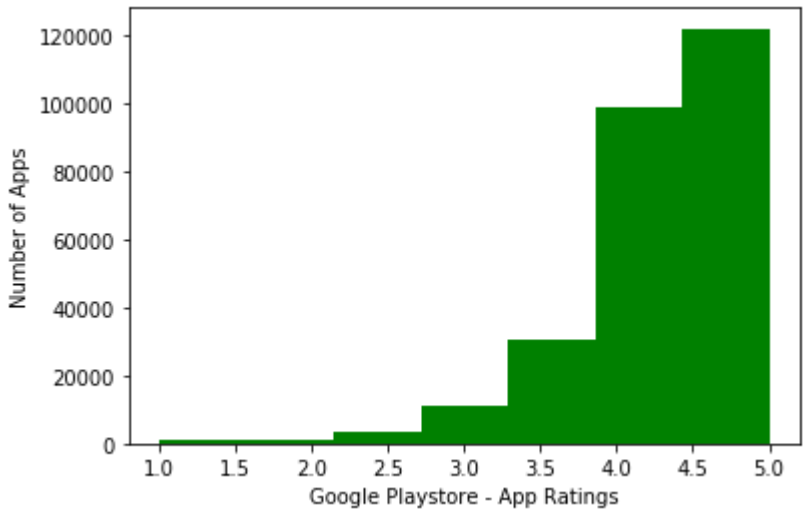
```
5.000000    23805
4.000000     5469
4.500000     3519
3.000000     2581
4.333333     2204
            ...
4.664837        1
4.386788        1
4.727144        1
4.428835        1
3.945443        1
Name: Rating, Length: 99845, dtype: int64
```

In [147]:

```
app_rating= app["Rating"]
num_bins=7
plt.hist(app_rating, num_bins, facecolor="green", alpha = 1)
plt.xlabel("Google Playstore - App Ratings")
plt.ylabel("Number of Apps")
plt.show
```

Out[147]:

```
<function matplotlib.pyplot.show(*args, **kw)>
```

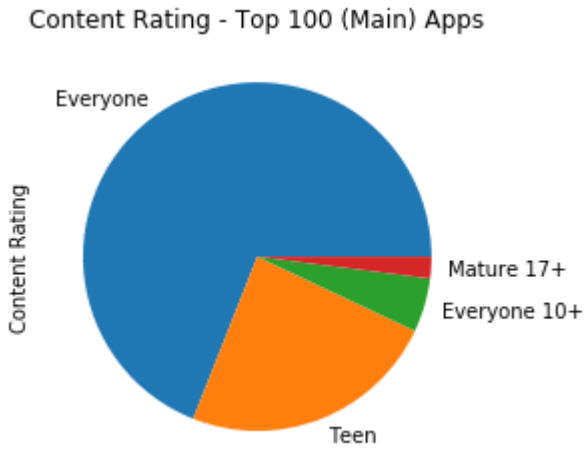


In [148]:

```
################################
```

In [149]:

```
app1=top_installed_and_rated_apps.head(100)
app1["Content Rating"].value_counts().plot.pie()
plt.title("Content Rating - Top 100 (Main) Apps")
plt.show()
```



In [150]:

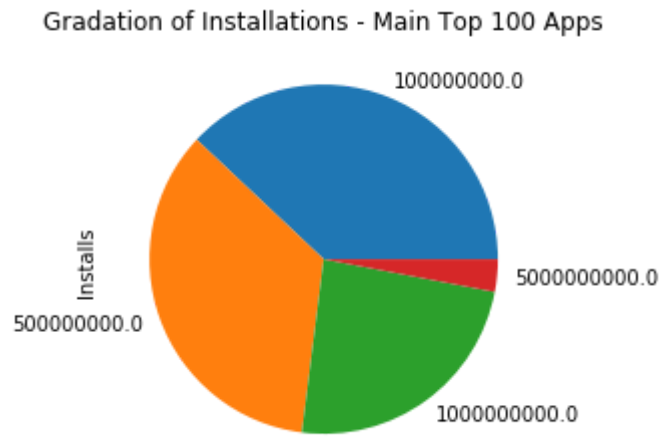
```
app1["Content Rating"].value_counts()
```

Out[150]:

```
Everyone        69
Teen            24
Everyone 10+     5
Mature 17+       2
Name: Content Rating, dtype: int64
```

In [151]:

```
app2= top_installed_and_rated_apps.head(100)
app2["Installs"].value_counts().plot.pie()
plt.title("Gradation of Installations - Main Top 100 Apps")
plt.show()
```



In [152]:

```
app2["Installs"].value_counts()
```

Out[152]:

```
1.000000e+08    38
5.000000e+08    35
1.000000e+09    24
5.000000e+09     3
Name: Installs, dtype: int64
```

In [153]:

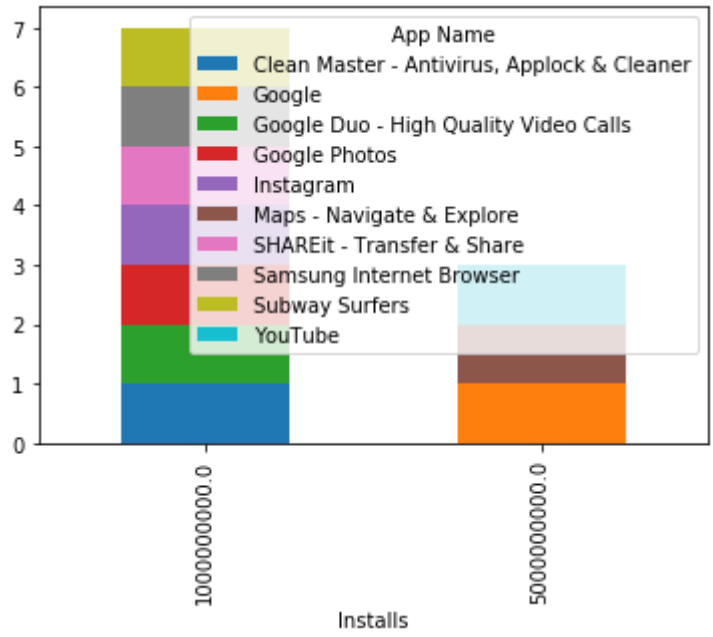
```
top_10_installed_and_rated_apps
```

Out[153]:

	App Name	Category	Rating	Reviews	Installs	Size	Price	C
821	Google	TOOLS	4.408893	10870728.0	5.000000e+09	Varies with device	0.0	Eve
813	YouTube	VIDEO_PLAYERS	4.368428	41919102.0	5.000000e+09	Varies with device	0.0	
2177	Maps - Navigate & Explore	TRAVEL_AND_LOCAL	4.342798	10083666.0	5.000000e+09	Varies with device	0.0	Eve
7064	Clean Master - Antivirus, Applock & Cleaner	TOOLS	4.657038	44171776.0	1.000000e+09	20	0.0	Eve
6411	Google Duo - High Quality Video Calls	COMMUNICATION	4.596404	3641252.0	1.000000e+09	20	0.0	Eve
3269	SHAREit - Transfer & Share	TOOLS	4.579340	10450444.0	1.000000e+09	20	0.0	Eve
1259	Google Photos	PHOTOGRAPHY	4.542380	16278468.0	1.000000e+09	Varies with device	0.0	Eve
653	Instagram	SOCIAL	4.519560	79726403.0	1.000000e+09	Varies with device	0.0	
539	Subway Surfers	GAME_ARCADE	4.498131	29834812.0	1.000000e+09	85	0.0	Eve
3267	Samsung Internet Browser	COMMUNICATION	4.424015	832714.0	1.000000e+09	Varies with device	0.0	Eve

In [154]:

```
# top 10 main apps

app4= top_10_installed_and_rated_apps
top_apps=app4.groupby(["Installs", "App Name"]).size().unstack()
top_apps.plot(kind="bar", stacked=True)
ax=plt.gca()
plt.show()
```



In [155]:

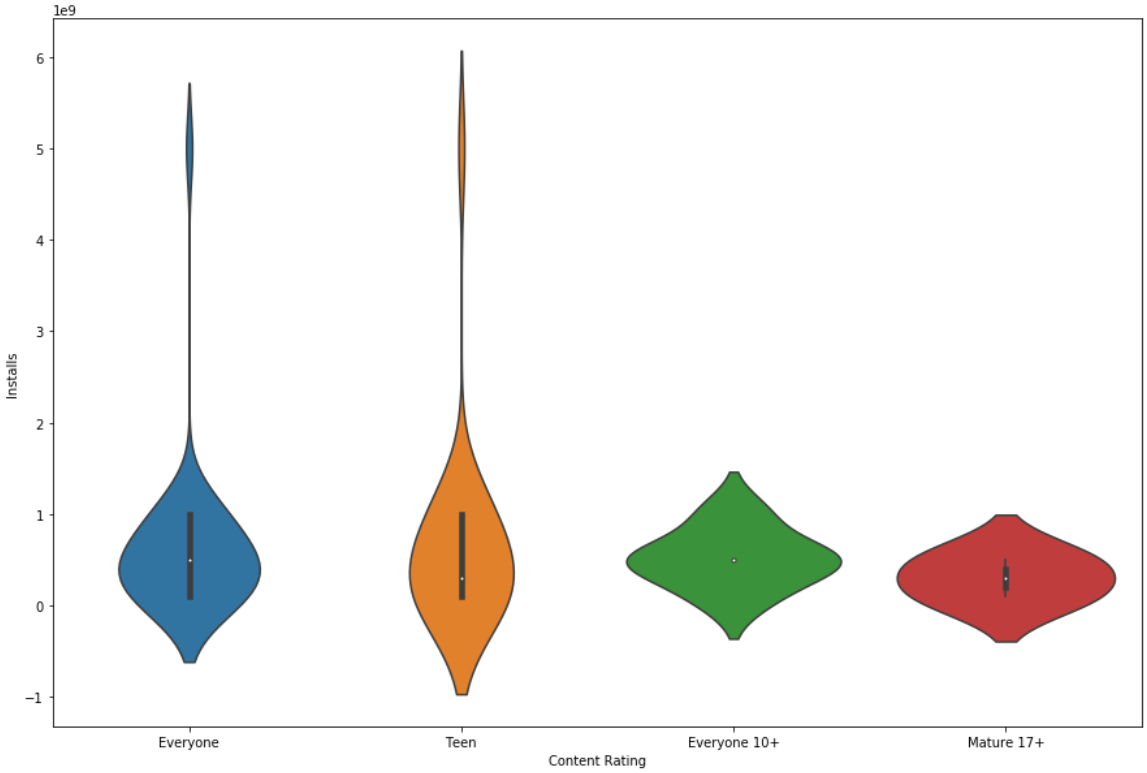
```
top_installed_and_rated_apps.head(100)["Content Rating"].value_counts(ascending= False)
```

Out[155]:

```
Everyone        69
Teen            24
Everyone 10+     5
Mature 17+       2
Name: Content Rating, dtype: int64
```

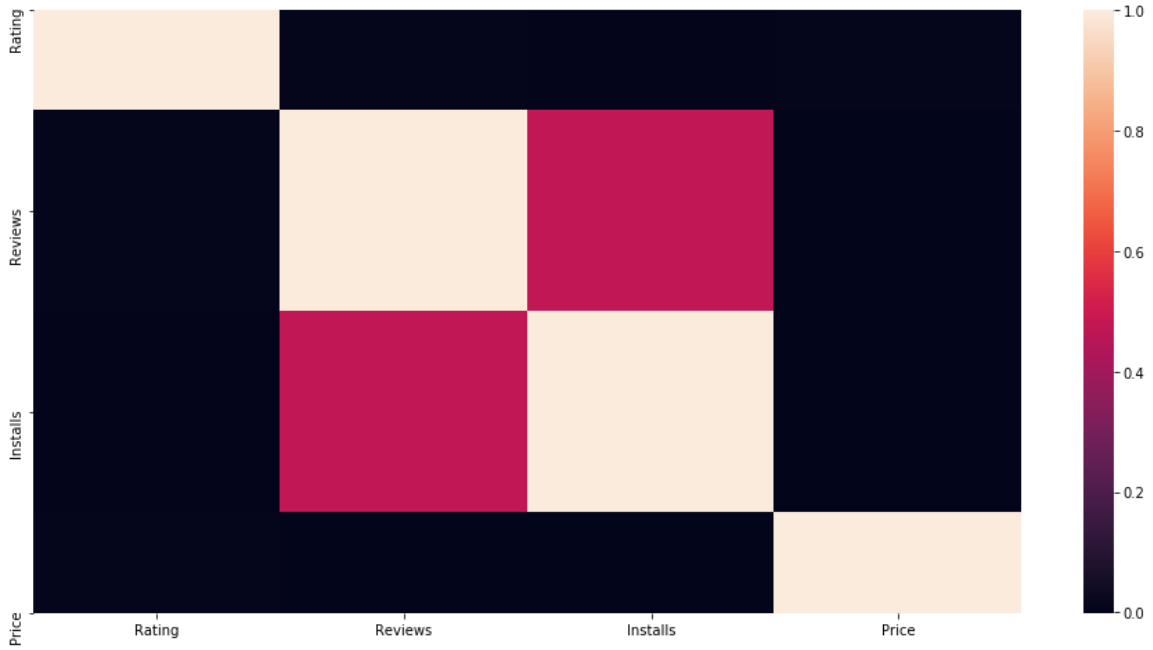
In [156]:

```
# violin plot - main top 100 apps - content rating vs installs
app5=top_installed_and_rated_apps.head(100)
plt.figure(figsize=(15,10))
sns.violinplot(x= "Content Rating", y="Installs", data= app5)
plt.show()
```



In [157]:

```
plt.figure(figsize=(16,8))
corr= app.corr()
sns.heatmap(corr)
plt.show()
```



In [158]:

```
##########################################################################################
#################
```

In [159]:

```
##########################################################################################
#################
```

In [160]:

```
##########################################################################################
#################
```

Unsupervised Methods

In [161]:

```
new_app = app.head(1000) # taking a part(sample) from the data set to apply supervided
 and unsupervised
# i did not take a bigger sample because of memory crashes
```

In [162]:

```
new_app.shape
```

Out[162]:

```
(1000, 11)
```

In [163]:

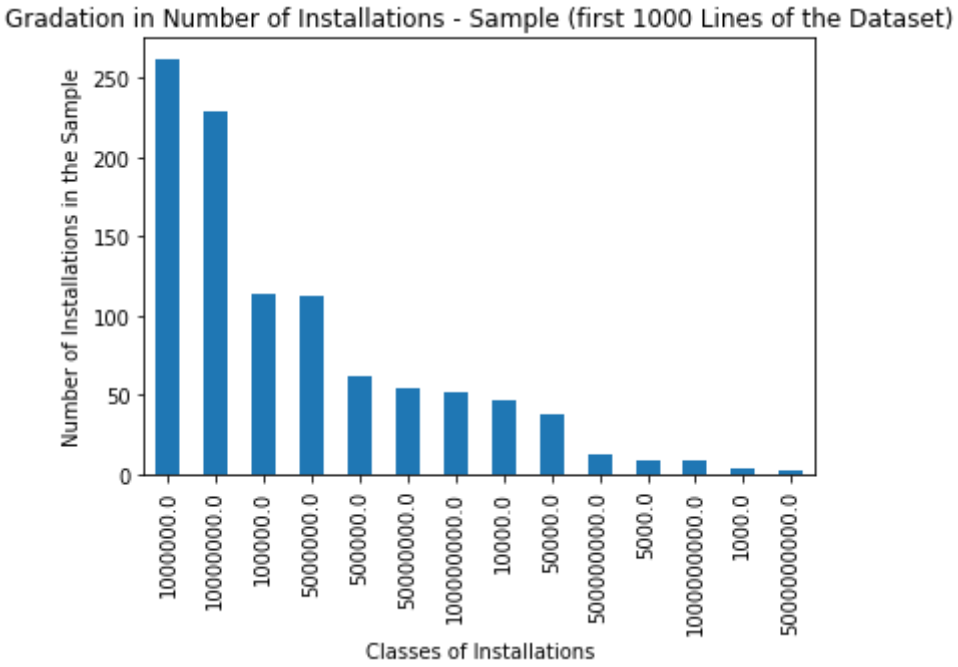
```
new_app["Installs"].value_counts().sort_values(ascending=False)
```

Out[163]:

```
1.000000e+06    262
1.000000e+07    229
1.000000e+05    113
5.000000e+06    112
5.000000e+05     62
5.000000e+07     54
1.000000e+08     52
1.000000e+04     46
5.000000e+04     37
5.000000e+08     12
5.000000e+03      8
1.000000e+09      8
1.000000e+03      3
5.000000e+09      2
Name: Installs, dtype: int64
```

In [164]:

```
# I want to see the gradation of the number of installations of the new dataframe(sampl
e), so as to compare later
# so as to compare the unsupervised and supervised methods results

new_app["Installs"].value_counts().sort_values(ascending=False).plot.bar()
plt.ylabel("Number of Installations in the Sample")
plt.xlabel("Classes of Installations")
plt.title("Gradation in Number of Installations - Sample (first 1000 Lines of the Datas
et)")
plt.show()
```



In [165]:

```
#**********************************
# Import packages from Scikit Learn

from sklearn import cluster # in unsupervised method we have clusters/ data ara grouped
into clusters
from sklearn import metrics # for the distances between the data
from sklearn.preprocessing import scale # for scaling
from sklearn.preprocessing import LabelEncoder # for converting strings to floats
from sklearn.preprocessing import OrdinalEncoder # for converting strings to floats whe
n x(attributes) are strings
```

In [166]:

```
#*****************************************************************************

# Segmenting the data i have chosen into attributes (features)=x, and target=(y)

# y will be the number of installations

# x will be: Category, Rating, Reviews and Content Rating


x= new_app[['Category', 'Rating', 'Reviews', 'Content Rating']] # attributes

y= new_app["Installs"] # y included the classes of installations. e.g. 100,000 in the d
ataset means more than 100,000 installations
```

In [167]:

```
# x has strings. This command is for converting strings to floats

x_transformed= OrdinalEncoder().fit_transform(x)
```

In [168]:

```
# Preparing the data- Scaling/ Handling the data in such way they can belong in a spesi
fic range
# and represent the same degree of difference
#*****************************************************************************
*********************

scaled_data= scale(x_transformed)
```

In [169]:

```
scaled_data
```

Out[169]:

```
array([[-0.77036381,  1.06735713,  0.86717825, -0.57690942],
       [ 2.36819293,  0.28702897,  1.37371577, -0.57690942],
       [ 1.96321786, -1.66552549, -1.2428125 , -0.57690942],
       ...,
       [ 2.1657054 , -1.68980237,  0.0043178 , -0.57690942],
       [ 2.1657054 , -1.72448362,  0.30125358, -0.57690942],
       [ 2.1657054 ,  1.16099651, -0.64894093, -0.57690942]])
```

In [170]:

```
# import python libraries for creating clusters, for converting and for scaling

from sklearn import cluster
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import scale
```

In [171]:

```
# i have taken a sample, so now the clusters of installations are 14 from 21 that norma
lly are for the whole dataset
# creating  clusters using Agglomerative Clustering
len(np.unique(y))
```

Out[171]:

```
14
```

In [172]:

```
y.unique()
```

Out[172]:

```
array([5.e+06, 1.e+08, 1.e+05, 1.e+07, 1.e+04, 1.e+06, 5.e+07, 5.e+05,
       5.e+04, 5.e+03, 1.e+03, 5.e+08, 1.e+09, 5.e+09])
```

In [173]:

```
#**************************************************************************************
***************
# Hierarchical agglomerative clustering - bottom-up approach
#**************************************************************************************
***************
#**************************************************************************************
***************
#**************************************************************************************
***************

# Using average in linkage means that i use the average of the distances of each observ
ation
from sklearn.cluster import AgglomerativeClustering
n_samples, n_features = scaled_data.shape
n_digits = len(np.unique(y))
model = cluster.AgglomerativeClustering(n_clusters=n_digits, linkage="average", affinit
y="cosine")
model.fit(scaled_data)
# this is the model created
```

Out[173]:

```
AgglomerativeClustering(affinity='cosine', compute_full_tree='auto',
                        connectivity=None, distance_threshold=None,
                        linkage='average', memory=None, n_clusters=14)
```

In [174]:

```
print (model.labels_)
```

```
[ 0  8  4  5  9  6  3  7  4  7  7  6  4  6  0  0  6  7  7  6  7  5  4  4
 10  7  8  7  5  7  5  9  6  5  5  8  6  7  3  6  6  7  9  3  5  4  2  9
  1  9 10  9  6 10  2 10  6  9  1  6  9  1  6  3  9  9 10  1 10  9  1  7
  9  3  6  3  2  2 10  9  1  6  7  2  1  6  6  6  9  9  9 10  1  6  9  6
 10  6  1  6  6  6  1  6 10  7  3  7  6  6  7  6  5  5  6  7  9  6  7  7
  7  9  7  0  6  5  9  5  9  9  6  7  6  6  6  6  6  6  5  5  5  5  6  9
  6  6  6  7  6  7  5  6  7  6  6  6  3  7  6  6  5  6  7  0  8  7  7  6
  0  6  7  9  6  6  9  7  6  6  6  6  6  6  6  6  6  6  6  6  7  6  6  6
  6  9  9  6  6  9  7  6  4  6  9  6  6  6  6  6  6  6  6  6  6  6  9  5
  6  6  6  9  6  9  6  2  6  6  6  6  9  6  6  6  6  9  6  6  6  6  6  6
  6  9  6  6  0  6  6  6  4  6  6  6  6  9  9  7  5  5  3  3  3  3  7  6
  6  2 11  6  3  0  2  2  9  3  3  4  6  2  2  2  2  1  6  2  3  3  2  7
  6  2  2  9  3  5  3  2  1 13  3  0  5  7  9  5  3  3  2  6  2  2  3 11
  4  3  9  0  9 12  0  0 11  9  8  4  0  0  0  6  7  0  4 11  4 11 11  5
  7  5  5 12 11  5 11  0 11  6 12  6  8  5 12  9  6 12 11  9  8 11 11  6
  0 11  6  8  4  8  6  7 11  5  8  6  6  5  9  8  8  7  6  4  6  4  8  4
  6  8  4  2  7  7  5  8  9  6  6  9  6  5  6  7  8  8  7  6  7  5  5  4
  8  9  6  7  6  8  6  7  5  7  6  7  8  7  5 11  8  6  5  6  6  8  6  6
  5  8  6  6  9 11  7  6  8  8  6  4  9  6  7  8  4 11  2  1  8 11  9  4
  8  9  8  4  4  8  8  3 10  8 11  8  8  9  8  4  8  7  1  8  8  4  8  8
  4 10  4 11  8 10  7  8  4  8  1  1  5 10  2  8  4  4  8  8 11  8  8  4
  4 10  8  3  8  7  3  4 10  0  4  4  8 10  8  4  3  8  9  3  8  8  8  8
  7  4  4 11  8  8  4  9  0  4  8  0  2  3  5  5  5  3  6  7  6  2  6  2
  2  0  5  3  3  5  7  6  3  3  7  6  0  9  9  2  6  3  2  7  3  5  0  9
  6  0  8  2  9  6  5  2  5  3  0  1  6  5  3  6  9  3  5  5  5  3  3  2
  0  2  3  0  3  5  5  2  5  5  5  5  3  9  0  9  5  2  7  5  3  3  0  5
 10  8  0  0  0 12  8 10  5  3  0 10  3  0  3 11  8  0  3  7 11  4  3  6
  0 12  1  3  5  3  9  4  5  1  3  4  3  0  6  1  3 11  5  3 12  3  5  5
  1  9  5  3  3  3  3  8  0  3  5  3  6  5  5  6 11  8  0  3  7  0  3  7
  4  8  8 13  0  3  3  0  3  3  3  6  5  6  9  2  6  6  6  6  7  9 11  0
  6 12  9  4  3  0  3  0  3  0  3 11  3  0  3  3  0  3  0  5  0  3  0  0
  0  3  3  0  1  0  3  3  3  3  0  3  0  3  3  3  3  6  6  7  6  6  3  1
  3  9  9 12  7  7  5  6  6  6  6  6  9  2  2  5  6  5  6  6  3  3  6  0
  0  9  7  5  6  6  6  6  2  3  2  0  6  7  3  9  0  3 13  9  7  3  3  5
  0  0  8 12 12  8 11  7  8  3  8  8 11  7  0  3  5  3  2  8  9  6  1  3
  5  0  5  8  8  3  0  4  7  1 13  0  5  6 11  1  3  3  7  5  9  2  3  1
  0  0  1  2  1  3  1  7  6  5  0  4  3  3 11  9  7  7 11  3  5  3 12 11
  2  5  3  7  3  6  3  2  5  6  6  7  6 11  9 11  0  4  2  1  4  3  3  0
 12  3  0  3  3 12 11  0 11  0  6 11  0 11  3 11  0  3 12  0  6  0 11 11
  3  1 12 12  0 11  1 11 12  4 11 10  3  1  0 11 11  6 10  0 10  9 11 12
  1 11 11 12  8 10  1  0 12 10  3  0  0  8 11  3  4  4  8  8 11  6  8  4
  8  8  8  4  4  8  4  4  4  8  8  4  4  4  4  8]
```

In [175]:

```
# Silhouette score: comprares the similarity of an object to its own cluster with that
 to other clusters

# models labels= models assigned to the model
#
#
print (metrics.silhouette_score(scaled_data,model.labels_))
print (metrics.completeness_score(y, model.labels_))
print (metrics.homogeneity_score(y, model.labels_))
```

```
0.2046255768331342
0.18939644179827697
0.21934387424949217
```

In [176]:

```
len(np.unique(y))
```

Out[176]:

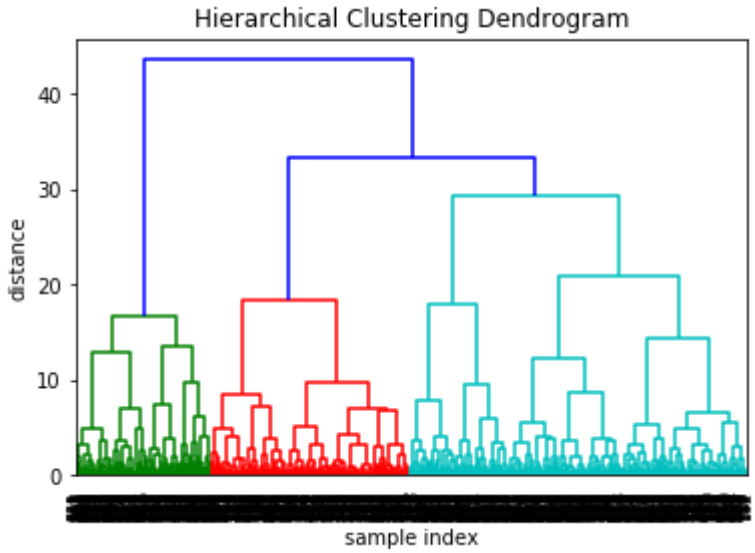
```
14
```

In [177]:

```
from scipy.cluster.hierarchy import dendrogram, linkage
```

In [178]:

```
# Creating Hierarchical Clustering Dendrogram

model= linkage(scaled_data, "ward")
plt.figure()
plt.title("Hierarchical Clustering Dendrogram")
plt.xlabel("sample index")
plt.ylabel("distance")
dendrogram(model, leaf_rotation=90.,  leaf_font_size=8.)
plt.show()
```



In [179]:

```
len(np.unique(y))
```

Out[179]:

14

In [180]:

```
#*****************************************************************************************
***************
#*****************************************************************************************
***************
#*****************************************************************************************
***************

# Clustering using K-means
# need for spesification of numbers of clusters
# clusters in this sample are 14
#*****************************************************************************************
***************
#*****************************************************************************************
***************
#*****************************************************************************************
***************
from sklearn import cluster
from sklearn.preprocessing import LabelEncoder
n_samples, n_features = scaled_data.shape
n_digits = len(np.unique(y))
for k in range(2, 15):
    kmeans = cluster.KMeans(n_clusters=k)
    kmeans.fit(scaled_data)
    print(k)
    print(metrics.silhouette_score(scaled_data, kmeans.labels_))
    print(metrics.completeness_score(y, kmeans.labels_))
    print(metrics.homogeneity_score(y, kmeans.labels_))

# different results on every iteration because we are using random starting points# bes
t score seems to be when k=13 (sometimes when k=14)
```

```
2
0.3160804206956437
0.01748956052802014
0.004283511685507621
3
0.2712690890288309
0.09333830946090167
0.047178895119585326
4
0.27141363588810574
0.08157230707935778
0.05349742834246447
5
0.26420861010801616
0.15649240959090796
0.11997134854556231
6
0.27390111430441044
0.17666325277367087
0.1471235112545505
7
0.27935616993890894
0.16840774995574004
0.15390791322286954
8
0.27189303628879835
0.17349055113129452
0.1693564129408031
9
0.27636384532520186
0.15546437107899316
0.1579033281066116
10
0.2680718309420629
0.1669413476320833
0.1793341334755354
11
0.27453516667893507
0.17091282733423438
0.1913787338995597
12
0.26572485130427015
0.19508680951906274
0.22718056827487962
13
0.2760251845062307
0.1920415193290409
0.23058338476265422
14
0.27617409404156035
0.19510601825779625
0.24205033996401362
```

In [181]:

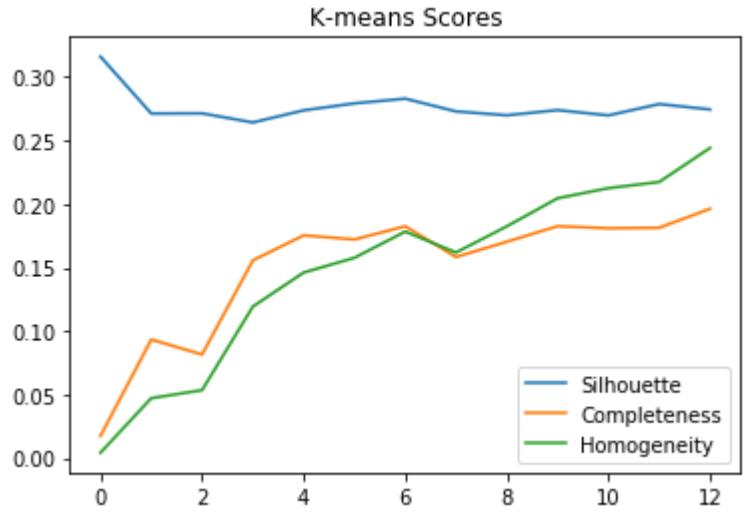
```
# same command with above, but now creating a list for every score in order to show it
 to a graph

n_samples, n_features = scaled_data.shape
n_digits = len(np.unique(y))

y_silhouette=[]
y_completeness=[]
y_homogeneity=[]


for k in range(2, 15):
    kmeans = cluster.KMeans(n_clusters=k)
    kmeans.fit(scaled_data)
    print(k)
    print(metrics.silhouette_score(scaled_data, kmeans.labels_))
    y_silhouette.append(metrics.silhouette_score(scaled_data, kmeans.labels_))

    print(metrics.completeness_score(y, kmeans.labels_))
    y_completeness.append(metrics.completeness_score(y, kmeans.labels_))


    print(metrics.homogeneity_score(y, kmeans.labels_))
    y_homogeneity.append(metrics.homogeneity_score(y, kmeans.labels_))



print("**********************************************************************************
***********************************")
print("**********************************************************************************
***********************************")
print ("silhouette scores are:\n{}".format(y_silhouette))
print("**********************************************************************************
***********************************")
print ("completeness scores are:\n{}".format(y_completeness))
print("**********************************************************************************
***********************************")
print ("homogeneity scores are:\n{}".format(y_homogeneity))
print("**********************************************************************************
***********************************")
```

```
2
0.3160804206956437
0.01748956052802014
0.004283511685507621
3
0.2712690890288309
0.09333830946090167
0.047178895119585326
4
0.27141363588810574
0.0815723070793578
0.05349742834246449
5
0.2642547194013099
0.15565564337600316
0.11930418035003615
6
0.2737759336017376
0.17537426928827313
0.1460896332532604
7
0.27927171868182055
0.17213460723563243
0.15769134622467323
8
0.2829838924573161
0.18251985704139956
0.17832669937991202
9
0.2729292227632848
0.15831407429314642
0.16187857268888395
10
0.26990294756387284
0.17033174149822533
0.18247612036672514
11
0.27396305435078866
0.1825909504226814
0.20457499933675968
12
0.2698134417178773
0.18093404335761015
0.21255223733293446
13
0.27873044639023054
0.18133142321873366
0.21737860532362838
14
0.27450134576148183
0.19628169712239674
0.24415514347729841
**************************************************************************
****************************************
**************************************************************************
****************************************
silhouette scores are:
[0.3160804206956437, 0.2712690890288309, 0.27141363588810574, 0.2642547194
013099, 0.2737759336017376, 0.27927171868182055, 0.2829838924573161, 0.272
9292227632848, 0.26990294756387284, 0.27396305435078866, 0.269813441717877
3, 0.27873044639023054, 0.27450134576148183]
```

```
**************************************************************************
****************************************
completeness scores are:
[0.01748956052802014, 0.09333830946090167, 0.0815723070793578, 0.155655643
37600316, 0.17537426928827313, 0.17213460723563243, 0.18251985704139956,
0.15831407429314642, 0.17033174149822533, 0.1825909504226814, 0.1809340433
5761015, 0.18133142321873366, 0.19628169712239674]
**************************************************************************
****************************************
homogeneity scores are:
[0.004283511685507621, 0.047178895119585326, 0.05349742834246449, 0.119304
18035003615, 0.1460896332532604, 0.15769134622467323, 0.17832669937991202,
0.16187857268888395, 0.18247612036672514, 0.20457499933675968, 0.212552237
33293446, 0.21737860532362838, 0.24415514347729841]
**************************************************************************
****************************************
```

In [182]:

```
plt.plot(y_silhouette)
plt.plot(y_completeness)
plt.plot(y_homogeneity)
plt.legend(["Silhouette", "Completeness", "Homogeneity"])
plt.title("K-means Scores")
plt.show()
```



In [183]:

```
##########################################################################################
#################
```

In [184]:

```
##########################################################################################
#################
```

In [185]:

```
##########################################################################################
#################
```

Supervised Methods

In [186]:

```
new_app.shape
```

Out[186]:

```
(1000, 11)
```

In [187]:

```
supervised_app_x= new_app[['Category', 'Rating', 'Reviews', 'Content Rating']]
supervised_app_y= new_app["Installs"]
```

In [188]:

```
supervised_x=supervised_app_x.values # attributes
supervised_y= supervised_app_y.values #target
```

In [189]:

```
supervised_x_transformed= OrdinalEncoder().fit_transform(supervised_x) # conversting th
e string values to floats for applying distance metrics
```

In [190]:

```
supervised_y= supervised_y.astype(int)
```

In [191]:

```
# segmenting the data in a training and test set of a 60/40 split
```

In [192]:

```
from sklearn.model_selection import train_test_split
```

In [193]:

```
supervised_x_transformed_train, supervised_x_transformed_test, supervised_y_train, supe
rvised_y_test= train_test_split(supervised_x_transformed, supervised_y, test_size=0.4)
```

In [194]:

```
# Creating classifiers to predict the class of installations, using:
# i. Logistic regression
# ii. KNN
```

In [195]:

```
from sklearn import preprocessing
print("LOGISTIC REGRESSION")
print("************************************")
from sklearn.linear_model import LogisticRegression
lm = LogisticRegression()
lm.fit(supervised_x_transformed_train, supervised_y_train)
lm.predict_proba(supervised_x_transformed_test)
```

```
LOGISTIC REGRESSION
************************************

C:\Users\dimit\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
\linear_model\_logistic.py:940: ConvergenceWarning: lbfgs failed to conver
ge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[195]:

```
array([[6.88215036e-04, 4.28690798e-04, 2.56963655e-01, ...,
        2.09783605e-03, 8.07355891e-04, 3.69505186e-04],
       [2.82419004e-02, 1.89734645e-03, 1.18768201e-03, ...,
        3.10061724e-02, 2.73504383e-02, 1.92816034e-02],
       [2.57423693e-03, 2.57358790e-15, 8.02701426e-22, ...,
        7.60655221e-02, 1.41682882e-02, 5.15635449e-03],
       ...,
       [5.22268539e-05, 6.86235757e-05, 5.82835755e-01, ...,
        2.10375843e-04, 6.15619819e-05, 2.24502157e-05],
       [7.04410207e-04, 1.25827617e-18, 1.18513277e-26, ...,
        8.02178509e-02, 8.00578630e-03, 2.90224468e-03],
       [2.09285234e-03, 1.49813438e-18, 9.08855217e-30, ...,
        1.06414820e-01, 2.43348188e-02, 2.00784125e-02]])
```

In [196]:

```
print(lm.intercept_)
```

```
[-1.12000322e-03 -2.80917011e-05  2.86894961e-06  2.00291138e-03
  1.07045911e-03  4.99490549e-03  1.50678660e-03  6.97358065e-03
 -1.71187150e-04 -2.66314927e-03 -4.20359244e-03 -4.37873460e-03
 -2.29040725e-03 -1.69634655e-03]
```

In [197]:

```
print(lm.coef_)
```

```
[[-0.00680975 -0.00490878  0.00937331 -0.00136087]
 [-0.00027147 -0.00448946 -0.02613723 -0.0001261 ]
 [ 0.00215759  0.00784595 -0.05464859  0.00054986]
 [ 0.0215561   0.00612494 -0.00878226  0.00524104]
 [ 0.03377881  0.00546081 -0.0080147   0.0012453 ]
 [ 0.08623461  0.0039118  -0.00095206  0.00970735]
 [ 0.00134441  0.00319068  0.00397368  0.00157408]
 [ 0.03578764  0.00188402  0.00842434  0.00917421]
 [ 0.00855565 -0.00073122  0.01102006 -0.00275734]
 [-0.04549406 -0.00169095  0.01455029 -0.00260518]
 [-0.06221393 -0.00320284  0.01430965 -0.00722089]
 [-0.02864605 -0.00286649  0.01308354 -0.00821582]
 [-0.01924009 -0.00460439  0.01186104 -0.00290489]
 [-0.02673946 -0.00592407  0.01193893 -0.00230072]]
```

In [198]:

```
predicted = lm.predict(supervised_x_transformed_test)
print(metrics.classification_report(supervised_y_test, predicted))
print(metrics.confusion_matrix(supervised_y_test, predicted))
```

```
              precision    recall  f1-score   support

        1000       0.00      0.00      0.00         2
        5000       0.50      0.25      0.33         4
       10000       0.26      0.24      0.25        21
       50000       0.00      0.00      0.00        17
      100000       0.34      0.45      0.39        47
      500000       0.00      0.00      0.00        26
     1000000       0.51      0.59      0.55       110
     5000000       0.50      0.02      0.04        45
    10000000       0.37      0.85      0.51        81
    50000000       0.00      0.00      0.00        17
   100000000       0.00      0.00      0.00        23
   500000000       0.00      0.00      0.00         4
  1000000000       0.00      0.00      0.00         3

    accuracy                           0.41       400
   macro avg       0.19      0.18      0.16       400
weighted avg       0.33      0.41      0.32       400

[[ 0  1  0  0  1  0  0  0  0  0  0  0  0]
 [ 0  1  1  0  2  0  0  0  0  0  0  0  0]
 [ 0  0  5  0 14  0  2  0  0  0  0  0  0]
 [ 0  0  5  0 10  0  2  0  0  0  0  0  0]
 [ 0  0  7  0 21  0 16  0  3  0  0  0  0]
 [ 0  0  1  0  8  0 13  0  4  0  0  0  0]
 [ 0  0  0  0  6  0 65  1 38  0  0  0  0]
 [ 0  0  0  0  0  0 15  1 29  0  0  0  0]
 [ 0  0  0  0  0  0 12  0 69  0  0  0  0]
 [ 0  0  0  0  0  0  2  0 15  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 23  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  4  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  3  0  0  0  0]]

C:\Users\dimit\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
\metrics\_classification.py:1272: UndefinedMetricWarning: Precision and F-
score are ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [199]:

```
#K nearest neighbours

print("KNN")
print("************************************")
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
model.fit(supervised_x_transformed_train, supervised_y_train)
print(model)
```

```
KNN
************************************
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [200]:

```
predicted= model.predict(supervised_x_transformed_test)
print (metrics.classification_report(supervised_y_test, predicted))
print (metrics.confusion_matrix(supervised_y_test, predicted))
```

```
              precision    recall  f1-score   support

        1000       0.00      0.00      0.00         2
        5000       0.00      0.00      0.00         4
       10000       0.30      0.29      0.29        21
       50000       0.20      0.29      0.24        17
      100000       0.37      0.45      0.40        47
      500000       0.28      0.19      0.23        26
     1000000       0.63      0.69      0.66       110
     5000000       0.28      0.20      0.23        45
    10000000       0.56      0.69      0.62        81
    50000000       0.12      0.06      0.08        17
   100000000       0.60      0.39      0.47        23
   500000000       0.00      0.00      0.00         4
  1000000000       0.50      0.33      0.40         3

    accuracy                           0.47       400
   macro avg       0.30      0.28      0.28       400
weighted avg       0.45      0.47      0.45       400

[[ 0  0  1  0  1  0  0  0  0  0  0  0  0]
 [ 0  0  1  2  1  0  0  0  0  0  0  0  0]
 [ 0  0  6  7  8  0  0  0  0  0  0  0  0]
 [ 0  0  4  5  8  0  0  0  0  0  0  0  0]
 [ 0  0  6  8 21  6  6  0  0  0  0  0  0]
 [ 0  0  2  2  9  5  8  0  0  0  0  0  0]
 [ 0  0  0  1  9  7 76  9  8  0  0  0  0]
 [ 0  0  0  0  0  0 22  9 14  0  0  0  0]
 [ 0  0  0  0  0  0  8 14 56  1  2  0  0]
 [ 0  0  0  0  0  0  0  0 11  1  3  2  0]
 [ 0  0  0  0  0  0  0  0  7  5  9  1  1]
 [ 0  0  0  0  0  0  0  0  3  0  1  0  0]
 [ 0  0  0  0  0  0  0  0  1  1  0  0  1]]

C:\Users\dimit\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
\metrics\_classification.py:1272: UndefinedMetricWarning: Precision and F-
score are ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [201]:

```
print (metrics.accuracy_score(supervised_y_test, predicted))
```

```
0.4725
```

In [ ]:

```

```