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

In [2]:
```
data=pd.read_csv("telcom_data1.csv")
```

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

Out[3]:

	Bearer Id	Start	Start ms	End	End ms	Dur. (ms)	IMSI	MSISDN/Number	IMEI	Last Location Name	...	Youtube DL (Bytes)	UL
0	1.310000e+19	04-04-19 12:01	770.0	4/25/2019 14:35	662.0	1823652.0	2.082010e+14	3.366496e+10	3.552120e+13	9.16E+15	...	15854611.0	25
1	1.310000e+19	04-09-19 13:04	235.0	4/25/2019 8:15	606.0	1365104.0	2.082020e+14	3.368185e+10	3.579400e+13	L77566A	...	20247395.0	19
2	1.310000e+19	04-09-19 17:42	1.0	4/25/2019 11:58	652.0	1361762.0	2.082000e+14	3.376063e+10	3.528150e+13	D42335A	...	19725661.0	146
3	1.310000e+19	04-10-19 0:31	486.0	4/25/2019 7:36	171.0	1321509.0	2.082010e+14	3.375034e+10	3.535660e+13	T21824A	...	21388122.0	151
4	1.310000e+19	04-12-19 20:10	565.0	4/25/2019 10:40	954.0	1089009.0	2.082010e+14	3.369980e+10	3.540700e+13	D88865A	...	15259380.0	189

5 rows × 55 columns

In [4]:
```
data.shape
```

Out[4]:
```
(150001, 55)
```

'Bearer Id','Dur. (ms)','Activity Duration DL (ms)','Activity Duration UL (ms)','Social Media DL (Bytes)', 'Social Media UL (Bytes)', 'Google DL (Bytes)', 'Google UL (Bytes)', 'Email DL (Bytes)', 'Email UL (Bytes)', 'Youtube DL (Bytes)', 'Youtube UL (Bytes)', 'Netflix DL (Bytes)', 'Netflix UL (Bytes)', 'Gaming DL (Bytes)', 'Gaming UL (Bytes)', 'Other DL (Bytes)', 'Other UL (Bytes)', 'Total UL (Bytes)', 'Total DL (Bytes)'

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

Out[5]:
```
Index(['Bearer Id', 'Start', 'Start ms', 'End', 'End ms', 'Dur. (ms)', 'IMSI',
       'MSISDN/Number', 'IMEI', 'Last Location Name', 'Avg RTT DL (ms)',
       'Avg RTT UL (ms)', 'Avg Bearer TP DL (kbps)', 'Avg Bearer TP UL (kbps)',
       'TCP DL Retrans. Vol (Bytes)', 'TCP UL Retrans. Vol (Bytes)',
       'DL TP < 50 Kbps (%)', '50 Kbps < DL TP < 250 Kbps (%)',
       '250 Kbps < DL TP < 1 Mbps (%)', 'DL TP > 1 Mbps (%)',
       'UL TP < 10 Kbps (%)', '10 Kbps < UL TP < 50 Kbps (%)',
       '50 Kbps < UL TP < 300 Kbps (%)', 'UL TP > 300 Kbps (%)',
       'HTTP DL (Bytes)', 'HTTP UL (Bytes)', 'Activity Duration DL (ms)',
       'Activity Duration UL (ms)', 'Dur. (ms).1', 'Handset Manufacturer',
       'Handset Type', 'Nb of sec with 125000B < Vol DL',
       'Nb of sec with 1250B < Vol UL < 6250B',
       'Nb of sec with 31250B < Vol DL < 125000B',
       'Nb of sec with 37500B < Vol UL',
       'Nb of sec with 6250B < Vol DL < 31250B',
       'Nb of sec with 6250B < Vol UL < 37500B',
       'Nb of sec with Vol DL < 6250B', 'Nb of sec with Vol UL < 1250B',
       'Social Media DL (Bytes)', 'Social Media UL (Bytes)',
       'Google DL (Bytes)', 'Google UL (Bytes)', 'Email DL (Bytes)',
```

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150001 entries, 0 to 150000
Data columns (total 55 columns):
 #   Column                                    Non-Null Count   Dtype
---  ------                                    --------------   -----
 0   Bearer Id                                 149010 non-null  float64
 1   Start                                     150000 non-null  object
 2   Start ms                                  150000 non-null  float64
 3   End                                       150000 non-null  object
 4   End ms                                    150000 non-null  float64
 5   Dur. (ms)                                 150000 non-null  float64
 6   IMSI                                      149431 non-null  float64
 7   MSISDN/Number                             148935 non-null  float64
 8   IMEI                                      149429 non-null  float64
 9   Last Location Name                        148848 non-null  object
 10  Avg RTT DL (ms)                           122172 non-null  float64
 11  Avg RTT UL (ms)                           122189 non-null  float64
 12  Avg Bearer TP DL (kbps)                   150000 non-null  float64
 13  Avg Bearer TP UL (kbps)                   150000 non-null  float64
```

In [7]:
```
null_pct= data.isnull().sum() / len(data) * 100
null_pct
```

Out[7]:

Bearer Id	0.660662
Start	0.000667
Start ms	0.000667
End	0.000667
End ms	0.000667
Dur. (ms)	0.000667
IMSI	0.379997
MSISDN/Number	0.710662
IMEI	0.381331
Last Location Name	0.768662
Avg RTT DL (ms)	18.552543
Avg RTT UL (ms)	18.541210
Avg Bearer TP DL (kbps)	0.000667
Avg Bearer TP UL (kbps)	0.000667
TCP DL Retrans. Vol (Bytes)	58.763608
TCP UL Retrans. Vol (Bytes)	64.432237
DL TP < 50 Kbps (%)	0.502663
50 Kbps < DL TP < 250 Kbps (%)	0.502663
250 Kbps < DL TP < 1 Mbps (%)	0.502663
DL TP > 1 Mbps (%)	0.502663
UL TP < 10 Kbps (%)	0.527996
10 Kbps < UL TP < 50 Kbps (%)	0.527996
50 Kbps < UL TP < 300 Kbps (%)	0.527996
UL TP > 300 Kbps (%)	0.527996
HTTP DL (Bytes)	54.315638
HTTP UL (Bytes)	54.539636
Activity Duration DL (ms)	0.000667
Activity Duration UL (ms)	0.000667
Dur. (ms).1	0.000667
Handset Manufacturer	0.381331
Handset Type	0.381331
Nb of sec with 125000B < Vol DL	65.024900
Nb of sec with 1250B < Vol UL < 6250B	61.928920
Nb of sec with 31250B < Vol DL < 125000B	62.390251
Nb of sec with 37500B < Vol UL	86.835421
Nb of sec with 6250B < Vol DL < 31250B	58.877607
Nb of sec with 6250B < Vol UL < 37500B	74.561503
Nb of sec with Vol DL < 6250B	0.503330
Nb of sec with Vol UL < 1250B	0.528663
Social Media DL (Bytes)	0.000000
Social Media UL (Bytes)	0.000000
Google DL (Bytes)	0.000000
Google UL (Bytes)	0.000000

```
Email DL (Bytes)                    0.000000
Email UL (Bytes)                    0.000000
Youtube DL (Bytes)                  0.000000
Youtube UL (Bytes)                  0.000000
Netflix DL (Bytes)                  0.000000
Netflix UL (Bytes)                  0.000000
Gaming DL (Bytes)                   0.000000
Gaming UL (Bytes)                   0.000000
Other DL (Bytes)                    0.000000
Other UL (Bytes)                    0.000000
Total UL (Bytes)                    0.000667
Total DL (Bytes)                    0.000667
dtype: float64
```

In [8]:
```
# lets consider user as Bearer Id
user_df=data.groupby('MSISDN/Number').agg({
                                           'Dur. (ms)':'sum',
                                           'Activity Duration DL (ms)':'sum',
                                           'Activity Duration UL (ms)':'sum',
                                           'Social Media DL (Bytes)':'sum',
                                           'Social Media UL (Bytes)':'sum',
                                           'Google DL (Bytes)':'sum',
                                           'Google UL (Bytes)':'sum',
                                           'Email DL (Bytes)':'sum',
                                           'Email UL (Bytes)':'sum',
                                           'Youtube DL (Bytes)':'sum',
                                           'Youtube UL (Bytes)':'sum',

                                            'Netflix DL (Bytes)':'sum',
                                             'Netflix UL (Bytes)':'sum',
                                             'Gaming DL (Bytes)':'sum',
                                             'Gaming UL (Bytes)':'sum',
                                             'Other DL (Bytes)':'sum',
                                             'Other UL (Bytes)':'sum',
                                             'Total UL (Bytes)':'sum',
    'Total DL (Bytes)':'sum'}).reset_index()
```

In [9]:
```
# the user behaviour dataframe (if we consider user as bearer Id)
print(user_df)
```

```
        MSISDN/Number  Dur. (ms)  Activity Duration DL (ms)  \
0        3.360100e+10   116720.0                    26588.0
1        3.360100e+10   181230.0                    49283.0
2        3.360100e+10   134969.0                    16793.0
3        3.360101e+10    49878.0                    12097.0
4        3.360101e+10    37104.0                  4642908.0
...               ...        ...                        ...
106851   3.379000e+10     8810.0                   259862.0
106852   3.379000e+10   140988.0                    33435.0
106853   3.197020e+12   877385.0                        0.0
106854   3.370000e+14   253030.0                     8829.0
106855   8.823970e+14   869844.0                        0.0

        Activity Duration UL (ms)  Social Media DL (Bytes)  \
0                         33662.0                2206504.0
1                         54751.0                2598548.0
2                         18434.0                3148004.0
3                          4497.0                 251469.0
4                       3133148.0                2861230.0
```

In [10]:
```
user_df
```

Out[10]:

	MSISDN/Number	Dur. (ms)	Activity Duration DL (ms)	Activity Duration UL (ms)	Social Media DL (Bytes)	Social Media UL (Bytes)	Google DL (Bytes)	Google UL (Bytes)	Email DL (Bytes)	Email UL (Bytes)	Youtu DL (Byt
0	3.360100e+10	116720.0	26588.0	33662.0	2206504.0	25631.0	3337123.0	1051882.0	837400.0	493962.0	1490020
1	3.360100e+10	181230.0	49283.0	54751.0	2598548.0	62017.0	4197697.0	1137166.0	2828821.0	478960.0	532425
2	3.360100e+10	134969.0	16793.0	18434.0	3148004.0	47619.0	3343483.0	99643.0	2436500.0	768880.0	213727
3	3.360101e+10	49878.0	12097.0	4497.0	251469.0	28825.0	5937765.0	3740728.0	2178618.0	106052.0	439312
4	3.360101e+10	37104.0	4642908.0	3133148.0	2861230.0	51312.0	13728668.0	4770948.0	2247808.0	1057661.0	1033997
...	...	...	...	...	...	...	...	...	...	...	
106851	3.379000e+10	8810.0	259862.0	241044.0	234320.0	65863.0	6834178.0	697091.0	480946.0	525969.0	829431
106852	3.379000e+10	140988.0	33435.0	39006.0	442214.0	56355.0	1472406.0	3957299.0	2513433.0	664.0	559686
106853	3.197020e+12	877385.0	0.0	0.0	668596.0	46628.0	8572779.0	1865881.0	842279.0	678492.0	983988

In [11]:
```
df=pd.DataFrame(user_df)
```

In [12]:
```
df.columns
```

Out[12]:
```
Index(['MSISDN/Number', 'Dur. (ms)', 'Activity Duration DL (ms)',
       'Activity Duration UL (ms)', 'Social Media DL (Bytes)',
       'Social Media UL (Bytes)', 'Google DL (Bytes)', 'Google UL (Bytes)',
       'Email DL (Bytes)', 'Email UL (Bytes)', 'Youtube DL (Bytes)',
       'Youtube UL (Bytes)', 'Netflix DL (Bytes)', 'Netflix UL (Bytes)',
       'Gaming DL (Bytes)', 'Gaming UL (Bytes)', 'Other DL (Bytes)',
       'Other UL (Bytes)', 'Total UL (Bytes)', 'Total DL (Bytes)'],
      dtype='object')
```

In [13]:
```
df=pd.DataFrame(user_df)
```

In [14]:
```
df
```

Out[14]:

	MSISDN/Number	Dur. (ms)	Activity Duration DL (ms)	Activity Duration UL (ms)	Social Media DL (Bytes)	Social Media UL (Bytes)	Google DL (Bytes)	Google UL (Bytes)	Email DL (Bytes)	Email UL (Bytes)	Youtube DL (Bytes)
0	3.360100e+10	116720.0	26588.0	33662.0	2206504.0	25631.0	3337123.0	1051882.0	837400.0	493962.0	14900201.0
1	3.360100e+10	181230.0	49283.0	54751.0	2598548.0	62017.0	4197697.0	1137166.0	2828821.0	478960.0	5324251.0
2	3.360100e+10	134969.0	16793.0	18434.0	3148004.0	47619.0	3343483.0	99643.0	2436500.0	768880.0	2137272.0
3	3.360101e+10	49878.0	12097.0	4497.0	251469.0	28825.0	5937765.0	3740728.0	2178618.0	106052.0	4393123.0
4	3.360101e+10	37104.0	4642908.0	3133148.0	2861230.0	51312.0	13728668.0	4770948.0	2247808.0	1057661.0	10339971.0
...	...	...	...	...	...	...	...	...	...	...	...
106851	3.379000e+10	8810.0	259862.0	241044.0	234320.0	65863.0	6834178.0	697091.0	480946.0	525969.0	8294310.0
106852	3.379000e+10	140988.0	33435.0	39006.0	442214.0	56355.0	1472406.0	3957299.0	2513433.0	664.0	5596862.0
106853	3.197020e+12	877385.0	0.0	0.0	668596.0	46628.0	8572779.0	1865881.0	842279.0	678492.0	9839889.0
106854	3.370000e+14	253030.0	8829.0	10540.0	496337.0	25229.0	8215537.0	1615080.0	2989663.0	328919.0	16690728.0
106855	8.823970e+14	869844.0	0.0	0.0	1500145.0	45943.0	5985089.0	3233558.0	2518425.0	812549.0	18980320.0

106856 rows × 20 columns

Task 1.2

Task 1.2 - Conduct exploratory data analysis on those data & communicate useful insights. Ensure that you identify and treat all missing values and outliers in the dataset by replacing them with the mean of the corresponding column. You're expected to report about the following using Python script and slide : • Describe all relevant variables and associated data types (slide). • Analyze the basic metrics (mean, median, etc) in the Dataset (explain) & their importance for the global objective. • Conduct a Non-Graphical Univariate Analysis by computing dispersion parameters for each quantitative variable and provide useful interpretation. • Conduct a Graphical Univariate Analysis by identifying the most suitable plotting options for each variable and interpreting your findings. • Bivariate Analysis – explore the relationship between each application & the total DL+UL data using appropriate methods and interpret your findings. • Variable transformations – segment the users into the top five decile classes based on the total duration for all sessions and compute the total data (DL+UL) per decile class. • Correlation Analysis – compute a correlation matrix for the following

variables and interpret your findings: Social Media data, Google data, Email data, Youtube data, Netflix data, Gaming data, and Other data • Dimensionality Reduction – perform a principal component analysis to reduce the dimensions of your data and provide a useful interpretation of the results (Provide your interpretation in four (4) bullet points maximum)

In [15]:
```
df
```

Out[15]:

	MSISDN/Number	Dur. (ms)	Activity Duration DL (ms)	Activity Duration UL (ms)	Social Media DL (Bytes)	Social Media UL (Bytes)	Google DL (Bytes)	Google UL (Bytes)	Email DL (Bytes)	Email UL (Bytes)	Youtube DL (Bytes)
0	3.360100e+10	116720.0	26588.0	33662.0	2206504.0	25631.0	3337123.0	1051882.0	837400.0	493962.0	14900201.0
1	3.360100e+10	181230.0	49283.0	54751.0	2598548.0	62017.0	4197697.0	1137166.0	2828821.0	478960.0	5324251.0
2	3.360100e+10	134969.0	16793.0	18434.0	3148004.0	47619.0	3343483.0	99643.0	2436500.0	768880.0	2137272.0
3	3.360101e+10	49878.0	12097.0	4497.0	251469.0	28825.0	5937765.0	3740728.0	2178618.0	106052.0	4393123.0
4	3.360101e+10	37104.0	4642908.0	3133148.0	2861230.0	51312.0	13728668.0	4770948.0	2247808.0	1057661.0	10339971.0
...	...	...	...	...	...	...	...	...	...	...	...
106851	3.379000e+10	8810.0	259862.0	241044.0	234320.0	65863.0	6834178.0	697091.0	480946.0	525969.0	8294310.0
106852	3.379000e+10	140988.0	33435.0	39006.0	442214.0	56355.0	1472406.0	3957299.0	2513433.0	664.0	5596862.0
106853	3.197020e+12	877385.0	0.0	0.0	668596.0	46628.0	8572779.0	1865881.0	842279.0	678492.0	9839889.0
106854	3.370000e+14	253030.0	8829.0	10540.0	496337.0	25229.0	8215537.0	1615080.0	2989663.0	328919.0	16690728.0
106855	8.823970e+14	869844.0	0.0	0.0	1500145.0	45943.0	5985089.0	3233558.0	2518425.0	812549.0	18980320.0

106856 rows × 20 columns

In [16]:
```
df.shape
```

Out[16]:
```
(106856, 20)
```

In [17]:
```
df.dtypes
```

Out[17]:
```
MSISDN/Number                float64
Dur. (ms)                    float64
Activity Duration DL (ms)    float64
Activity Duration UL (ms)    float64
Social Media DL (Bytes)      float64
Social Media UL (Bytes)      float64
Google DL (Bytes)            float64
Google UL (Bytes)            float64
Email DL (Bytes)             float64
Email UL (Bytes)             float64
Youtube DL (Bytes)           float64
Youtube UL (Bytes)           float64
Netflix DL (Bytes)           float64
Netflix UL (Bytes)           float64
Gaming DL (Bytes)            float64
Gaming UL (Bytes)            float64
Other DL (Bytes)             float64
Other UL (Bytes)             float64
Total UL (Bytes)             float64
Total DL (Bytes)             float64
dtype: object
```

In [18]:
```
df.isnull().sum()
```

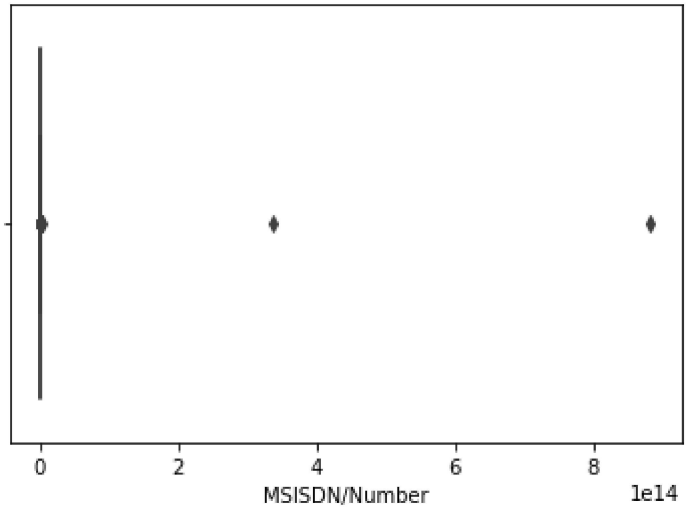
Out[18]:
```
MSISDN/Number                0
Dur. (ms)                    0
Activity Duration DL (ms)    0
Activity Duration UL (ms)    0
Social Media DL (Bytes)      0
Social Media UL (Bytes)      0
Google DL (Bytes)            0
Google UL (Bytes)            0
Email DL (Bytes)             0
Email UL (Bytes)             0
Youtube DL (Bytes)           0
Youtube UL (Bytes)           0
Netflix DL (Bytes)           0
Netflix UL (Bytes)           0
Gaming DL (Bytes)            0
Gaming UL (Bytes)            0
Other DL (Bytes)             0
Other UL (Bytes)             0
Total UL (Bytes)             0
Total DL (Bytes)             0
dtype: int64
```

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 106856 entries, 0 to 106855
Data columns (total 20 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   MSISDN/Number               106856 non-null  float64
 1   Dur. (ms)                   106856 non-null  float64
 2   Activity Duration DL (ms)   106856 non-null  float64
 3   Activity Duration UL (ms)   106856 non-null  float64
 4   Social Media DL (Bytes)     106856 non-null  float64
 5   Social Media UL (Bytes)     106856 non-null  float64
 6   Google DL (Bytes)           106856 non-null  float64
 7   Google UL (Bytes)           106856 non-null  float64
 8   Email DL (Bytes)            106856 non-null  float64
 9   Email UL (Bytes)            106856 non-null  float64
 10  Youtube DL (Bytes)          106856 non-null  float64
 11  Youtube UL (Bytes)          106856 non-null  float64
 12  Netflix DL (Bytes)          106856 non-null  float64
 13  Netflix UL (Bytes)          106856 non-null  float64
 14  Gaming DL (Bytes)           106856 non-null  float64
 15  Gaming UL (Bytes)           106856 non-null  float64
 16  Other DL (Bytes)            106856 non-null  float64
 17  Other UL (Bytes)            106856 non-null  float64
 18  Total UL (Bytes)            106856 non-null  float64
 19  Total DL (Bytes)            106856 non-null  float64
dtypes: float64(20)
memory usage: 16.3 MB
```

In [20]:
```
# fill missing values
# there are no missing values on this data
```

In [21]:
```
# Univariate Analysis using Boxplot
for i in df.columns:
    sns.boxplot(x=i,data=data)
    plt.show()
```



In [22]:
```
# Analysing Non-graphical univariate analysis

df.describe()
```

Out[22]:

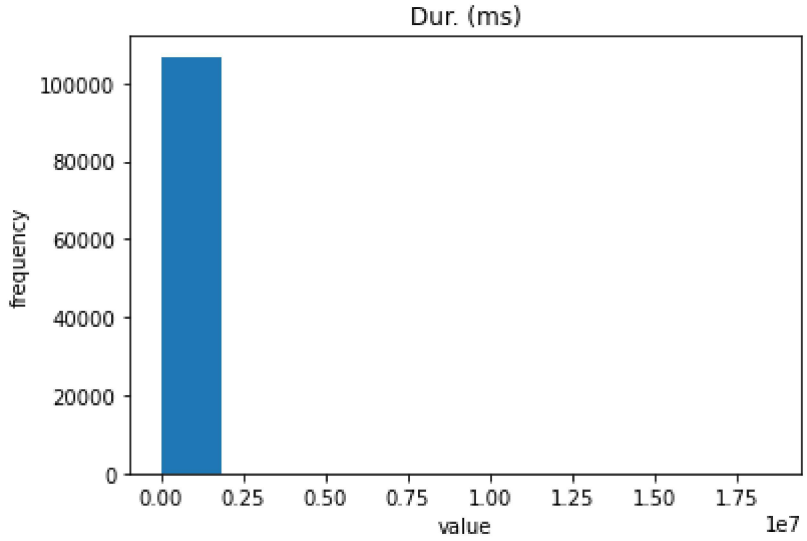
	MSISDN/Number	Dur. (ms)	Activity Duration DL (ms)	Activity Duration UL (ms)	Social Media DL (Bytes)	Social Media UL (Bytes)	Google DL (Bytes)	Google UL (Bytes)	Ema (B
count	1.068560e+05	1.068560e+05	1.068560e+05	1.068560e+05	1.068560e+05	106856.000000	1.068560e+05	1.068560e+05	1.068560
mean	4.511474e+10	1.461672e+05	2.556969e+06	1.968787e+06	2.502081e+06	45886.012802	8.016496e+06	2.865938e+06	2.497352
std	2.889423e+12	1.863587e+05	9.937870e+06	7.997885e+06	1.887588e+06	34717.044775	6.065160e+06	2.172787e+06	1.897063
min	3.360100e+10	7.142000e+03	0.000000e+00	0.000000e+00	1.200000e+01	0.000000	2.070000e+02	3.000000e+00	9.700000
25%	3.365088e+10	7.130800e+04	1.943400e+04	2.783700e+04	1.175902e+06	21600.750000	3.802894e+06	1.341100e+06	1.184544
50%	3.366365e+10	1.027400e+05	5.477550e+04	6.508100e+04	2.265000e+06	41559.500000	7.256742e+06	2.593314e+06	2.266259
75%	3.368344e+10	1.727990e+05	7.697832e+05	6.903182e+05	3.267238e+06	59919.250000	1.043106e+07	3.743934e+06	3.250651
max	8.823970e+14	1.855375e+07	3.406566e+08	3.347255e+08	4.274384e+07	630942.000000	1.161065e+08	3.608540e+07	3.360721

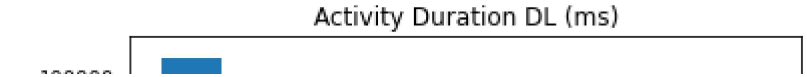
In [23]:
```
df.columns
```

Out[23]:
```
Index(['MSISDN/Number', 'Dur. (ms)', 'Activity Duration DL (ms)',
       'Activity Duration UL (ms)', 'Social Media DL (Bytes)',
       'Social Media UL (Bytes)', 'Google DL (Bytes)', 'Google UL (Bytes)',
       'Email DL (Bytes)', 'Email UL (Bytes)', 'Youtube DL (Bytes)',
       'Youtube UL (Bytes)', 'Netflix DL (Bytes)', 'Netflix UL (Bytes)',
       'Gaming DL (Bytes)', 'Gaming UL (Bytes)', 'Other DL (Bytes)',
       'Other UL (Bytes)', 'Total UL (Bytes)', 'Total DL (Bytes)'],
      dtype='object')
```

In [24]:
```
df.drop(['MSISDN/Number'], axis=1,inplace=True)
```

In [25]:
```
for i in df.columns:
    plt.hist(df[i],bins=10)
    plt.title(i)
    plt.xlabel('value')
    plt.ylabel('frequency')
    plt.savefig(f"{i}_histogram.png")
    plt.show()
```



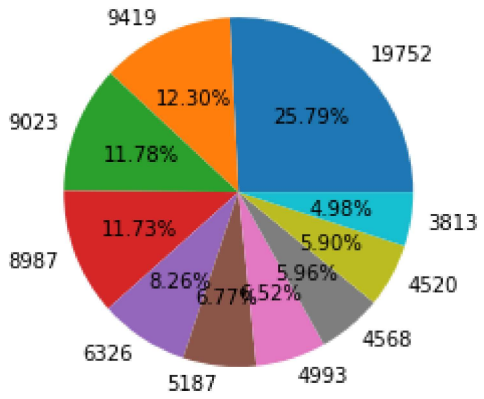


In [26]:
```
# top 10 Handset Type use by customers
Handset_type=data["Handset Type"].value_counts(ascending=False).head(10)
Handset_type
```

Out[26]:
```
Huawei B528S-23A                19752
Apple iPhone 6S (A1688)          9419
Apple iPhone 6 (A1586)           9023
undefined                        8987
Apple iPhone 7 (A1778)           6326
Apple iPhone Se (A1723)          5187
Apple iPhone 8 (A1905)           4993
Apple iPhone Xr (A2105)          4568
Samsung Galaxy S8 (Sm-G950F)     4520
Apple iPhone X (A1901)           3813
Name: Handset Type, dtype: int64
```

In [27]:
```
data['Handset_type']=Handset_type
```

In [28]:
```
# pieplot
plt.pie(x=Handset_type,labels=Handset_type, autopct = '%1.2f%%')
plt.show();
```

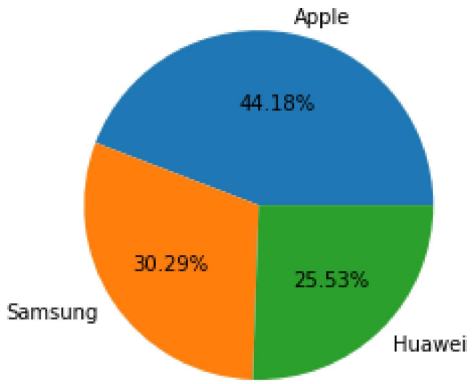


In [29]:
```
# top 3 handset manufacturer
top_3_manufacturer=data["Handset Manufacturer"].value_counts(ascending=False).head(3)
top_3_manufacturer
```

Out[29]:
```
Apple      59565
Samsung    40839
Huawei     34423
Name: Handset Manufacturer, dtype: int64
```

In [30]:
```
data['top_3_manufacturer']=top_3_manufacturer
```

In [31]:
```
# pieplot for top 3 Handset Manufacturer
plt.pie(labels=top_3_manufacturer.index,
        x=top_3_manufacturer, autopct = '%1.2f%%')
plt.show();
```



In [32]:
```
df.columns
```

Out[32]:
```
Index(['Dur. (ms)', 'Activity Duration DL (ms)', 'Activity Duration UL (ms)',
       'Social Media DL (Bytes)', 'Social Media UL (Bytes)',
       'Google DL (Bytes)', 'Google UL (Bytes)', 'Email DL (Bytes)',
       'Email UL (Bytes)', 'Youtube DL (Bytes)', 'Youtube UL (Bytes)',
       'Netflix DL (Bytes)', 'Netflix UL (Bytes)', 'Gaming DL (Bytes)',
       'Gaming UL (Bytes)', 'Other DL (Bytes)', 'Other UL (Bytes)',
       'Total UL (Bytes)', 'Total DL (Bytes)'],
      dtype='object')
```

In [33]:
```
df.skew
```

Out[33]:
```
<bound method NDFrame._add_numeric_operations.<locals>.skew of         Dur. (ms)  Activity Duration DL (m
s)  Activity Duration UL (ms)  \
0        116720.0                   26588.0                   33662.0
1        181230.0                   49283.0                   54751.0
2        134969.0                   16793.0                   18434.0
3         49878.0                   12097.0                    4497.0
4         37104.0                 4642908.0                 3133148.0
...           ...                       ...                       ...
106851     8810.0                  259862.0                  241044.0
106852   140988.0                   33435.0                   39006.0
106853   877385.0                       0.0                       0.0
106854   253030.0                    8829.0                   10540.0
106855   869844.0                       0.0                       0.0

        Social Media DL (Bytes)  Social Media UL (Bytes)  Google DL (Bytes)  \
0                     2206504.0                  25631.0          3337123.0
1                     2598548.0                  62017.0          4197697.0
2                     3148004.0                  47619.0          3343483.0
3                      251469.0                  28825.0          5937765.0
4                     2861220.0                  51312.0         12728668.0
```

In [34]:
```
# outlier detection by IQR
Q1 = np.percentile(df,25)
Q2 = np.percentile(df,50)
Q3 = np.percentile(df,75)
print(Q1,Q2,Q3)
IQR=Q3-Q1
print('IQR=',Q3-Q1)
```

```
720665.0 6159763.5 23869452.25
IQR= 23148787.25
```

In [35]:
```
# upper range and lower range
upperrange=(Q3+IQR)*1.5
print('upperrange=',(Q3+IQR)*1.5)
lowerrange=(Q1-IQR)*1.5
print('lowerrange=',(Q1-IQR)*1.5)
```

```
upperrange= 70527359.25
lowerrange= -33642183.375
```

In [43]:
```
column=df.columns
```

In [60]:
```
# from above upper and lower values we can find outlier
def find_outlier(df,threshold=1.5):
    outlier=pd.DataFrame()
    for i in df.columns:
        if df[i].dtype!='object':     #exclude category
            i_outlier=df[(df[i]<lowerrange)|(df[i]>upperrange)]
            outlier=pd.DataFrame()
            outliers=pd.concat([outlier,i_outlier])
#replcace outlier with mean

            df[i]=np.where((df[i]<lowerrange)|(df[i]>upperrange),df[i].mean(),df[i])
    return outliers
```

In [62]:
```
find_outlier(df)
```

Out[62]:

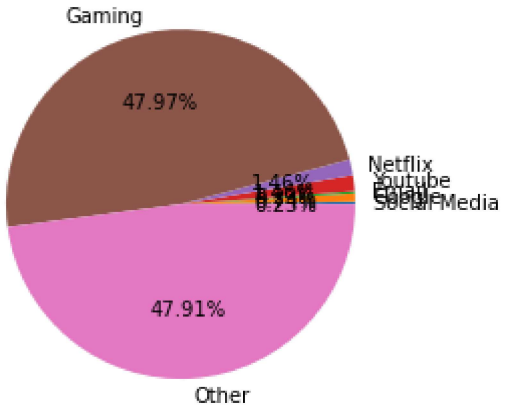
	Dur. (ms)	Activity Duration DL (ms)	Activity Duration UL (ms)	Social Media DL (Bytes)	Social Media UL (Bytes)	Google DL (Bytes)	Google UL (Bytes)	Email DL (Bytes)	Email UL (Bytes)	Youtube DL (Bytes)	Youtube UL (Bytes)	Net
0	116720.0	26588.0	33662.0	2206504.0	25631.0	3337123.0	1051882.0	837400.0	493962.0	14900201.0	6724347.0	1026
1	181230.0	49283.0	54751.0	2598548.0	62017.0	4197697.0	1137166.0	2828821.0	478960.0	5324251.0	7107972.0	77
2	134969.0	16793.0	18434.0	3148004.0	47619.0	3343483.0	99643.0	2436500.0	768880.0	2137272.0	19196298.0	1652
3	49878.0	12097.0	4497.0	251469.0	28825.0	5937765.0	3740728.0	2178618.0	106052.0	4393123.0	2584198.0	115
4	37104.0	4642908.0	3133148.0	2861230.0	51312.0	13728668.0	4770948.0	2247808.0	1057661.0	10339971.0	31193031.0	2497
...	...	...	...	...	...	...	...	...	...	...	...	
106851	8810.0	259862.0	241044.0	234320.0	65863.0	6834178.0	697091.0	480946.0	525969.0	8294310.0	18353533.0	1475
106852	140988.0	33435.0	39006.0	442214.0	56355.0	1472406.0	3957299.0	2513433.0	664.0	5596862.0	14254710.0	692
106853	877385.0	0.0	0.0	668596.0	46628.0	8572779.0	1865881.0	842279.0	678492.0	9839889.0	2120016.0	1034
106854	253030.0	8829.0	10540.0	496337.0	25229.0	8215537.0	1615080.0	2989663.0	328919.0	16690728.0	20044212.0	998
106855	869844.0	0.0	0.0	1500145.0	45943.0	5985089.0	3233558.0	2518425.0	812549.0	18980320.0	21960390.0	2162

103359 rows × 19 columns

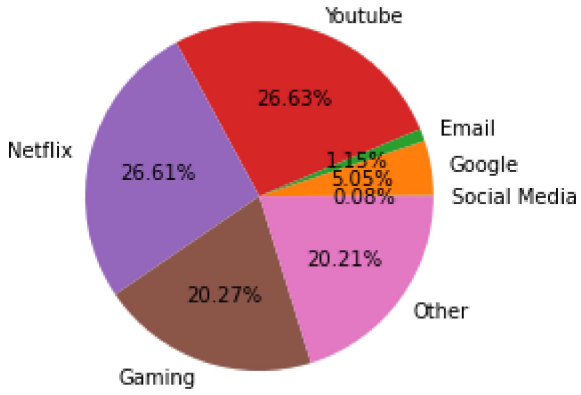
In [63]:
```
# total DL data for each category
dl_data = [
    df["Social Media DL (Bytes)"].sum(),
    df["Google DL (Bytes)"].sum(),
    df["Email DL (Bytes)"].sum(),
    df["Youtube DL (Bytes)"].sum(),
    df["Netflix DL (Bytes)"].sum(),
    df["Gaming DL (Bytes)"].sum(),
    df["Other DL (Bytes)"].sum()
]

# total UL data for each category
ul_data = [
    df["Social Media UL (Bytes)"].sum(),
    df["Google UL (Bytes)"].sum(),
    df["Email UL (Bytes)"].sum(),
    df["Youtube UL (Bytes)"].sum(),
    df["Netflix UL (Bytes)"].sum(),
    df["Gaming UL (Bytes)"].sum(),
    df["Other UL (Bytes)"].sum()
]
```

In [64]:
```
labels = ["Social Media", "Google", "Email", "Youtube", "Netflix", "Gaming", "Other"]
```

In [65]:
```
plt.pie(labels=labels,
        x=dl_data, autopct = '%1.2f%%')
plt.show();
```



In [66]:
```
plt.pie(labels=labels,
        x=ul_data, autopct = '%1.2f%%')
plt.show();
```

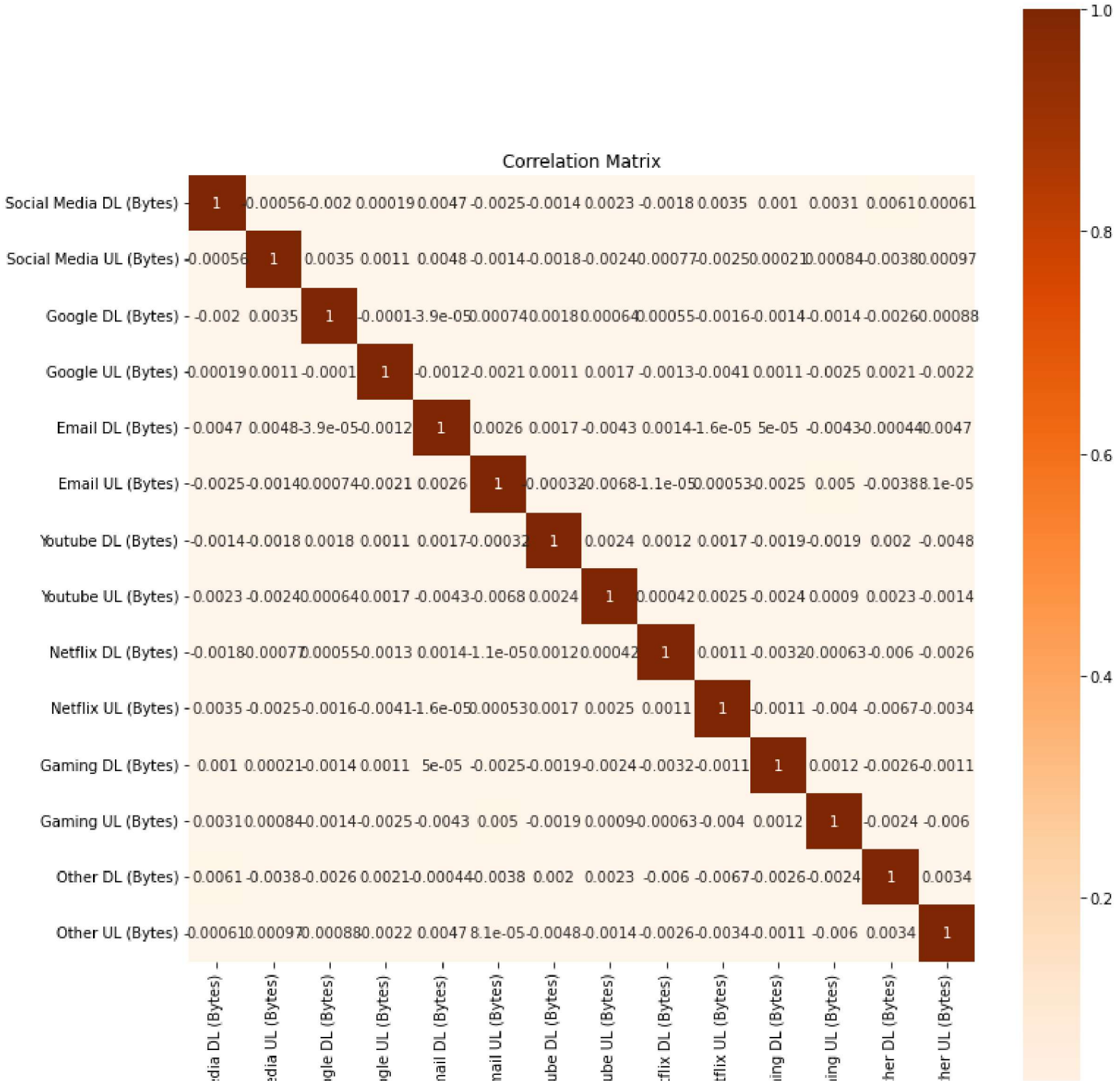


In [67]:
```
# correlation matrix
corr_matrix = data[['Social Media DL (Bytes)', 'Social Media UL (Bytes)',
       'Google DL (Bytes)', 'Google UL (Bytes)', 'Email DL (Bytes)',
       'Email UL (Bytes)', 'Youtube DL (Bytes)', 'Youtube UL (Bytes)',
       'Netflix DL (Bytes)', 'Netflix UL (Bytes)', 'Gaming DL (Bytes)',
       'Gaming UL (Bytes)', 'Other DL (Bytes)', 'Other UL (Bytes)',
       ]].corr()
```

In [68]:
```
print(corr_matrix)
```

```
                        Social Media DL (Bytes)  Social Media UL (Bytes)  \
Social Media DL (Bytes)                1.000000                -0.000555
Social Media UL (Bytes)               -0.000555                 1.000000
Google DL (Bytes)                     -0.001954                 0.003542
Google UL (Bytes)                      0.000186                 0.001085
Email DL (Bytes)                       0.004745                 0.004823
Email UL (Bytes)                      -0.002518                -0.001427
Youtube DL (Bytes)                    -0.001389                -0.001786
Youtube UL (Bytes)                     0.002345                -0.002368
Netflix DL (Bytes)                    -0.001817                -0.000772
Netflix UL (Bytes)                     0.003457                -0.002493
Gaming DL (Bytes)                      0.001018                 0.000210
Gaming UL (Bytes)                      0.003095                 0.000844
Other DL (Bytes)                       0.006126                -0.003850
Other UL (Bytes)                       0.000610                 0.000971

                        Google DL (Bytes)  Google UL (Bytes)  \
Social Media DL (Bytes)         -0.001954           0.000186
Social Media UL (Bytes)          0.003542           0.001085
```

In [69]:
```
# understand correlation matrix with heatmap
plt.figure(figsize=(12,14))
sns.heatmap(corr_matrix, annot=True, cmap='Oranges', square=True)
plt.title('Correlation Matrix')
plt.show()
```


Correlation Matrix
Social Media DL (Bytes)
Social Media UL (Bytes)
Google DL (Bytes)
Google UL (Bytes)
Email DL (Bytes)
Email UL (Bytes)
Youtube DL (Bytes)
Youtube UL (Bytes)
Netflix DL (Bytes)
Netflix UL (Bytes)
Gaming DL (Bytes)
Gaming UL (Bytes)
Other DL (Bytes)
Other UL (Bytes)
1.0
0.8
0.6
0.4
0.2

In [70]:
```
### Principle component Analysis
'''Before  PCA we have to do scaling on the Data,so we will use standard scaler method'''

from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
df_scaled=scaler.fit_transform(df)
df_scaled
```

Out[70]:
```
array([[-0.15801408, -0.30810187, -0.29870521, ..., -0.23527226,
        -0.78492406,  0.18388442],
       [ 0.18814798, -0.30477133, -0.29494446, ..., -1.10014688,
        -0.78074599,  0.18388442],
       [-0.06008955, -0.30953931, -0.30142079, ..., -0.57391072,
        -0.5201922 ,  0.18388442],
       ...,
       [ 3.92373087, -0.31200373, -0.30470809, ..., -1.01792953,
        -0.68380552,  0.18388442],
       [ 0.57342835, -0.31070805, -0.30282851, ..., -0.78332651,
         0.89113247,  0.18388442],
       [ 3.8832657 , -0.31200373, -0.30470809, ...,  0.08684532,
         1.20057316,  0.18388442]])
```

In [81]:
```
# now lets perform pca on the scaled data
from sklearn.decomposition import PCA
pca=PCA(n_components=2)  # here we take 95% of info from data
a=pca.fit(df_scaled)
a
```

Out[81]:
```
PCA(n_components=2)
```

In [82]:
```
b=pca.transform(df_scaled)
```

In [83]:
```
b
```

Out[83]:
```
array([[-1.92080617, -0.43343817],
       [-1.54859929, -0.39458342],
       [-1.27257224, -0.43005073],
       ...,
       [-0.72152904, -0.53018306],
       [-0.59159579, -0.56136748],
       [ 1.01003499,  1.89973986]])
```

In [84]:
```
b.shape
```

Out[84]:
```
(106856, 2)
```

In [85]:
```
df_scaled.shape
```

Out[85]:
```
(106856, 19)
```

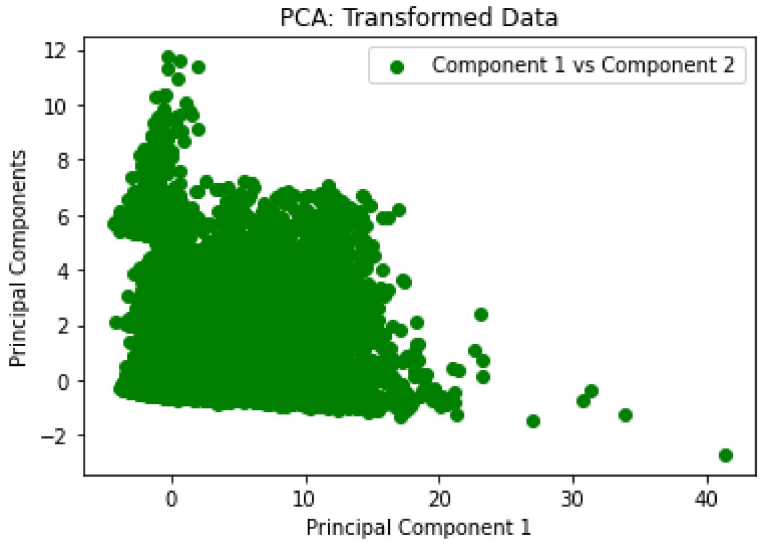
In [86]:
```
pca.components_
```

Out[86]:
```
array([[ 0.20903712,  0.12681533,  0.12804991,  0.27556875,  0.2756798 ,
         0.2742192 ,  0.27661254,  0.27523654,  0.27567202,  0.25010054,
         0.2627152 ,  0.25013885,  0.26221344,  0.05522797,  0.27372049,
         0.0506069 ,  0.27333553,  0.21963949,  0.04520406],
       [-0.02558963,  0.35188147,  0.35244863,  0.00310841,  0.00467125,
         0.0012246 ,  0.00397704,  0.00151671,  0.00528372, -0.01949092,
        -0.00857476, -0.0197542 , -0.01098173, -0.61039516, -0.00203946,
        -0.01755697, -0.0020324 , -0.05250189, -0.61205071]])
```

In [87]:
```
pca.explained_variance_ratio_
```

Out[87]:
```
array([0.41026888, 0.0908695 ])
```

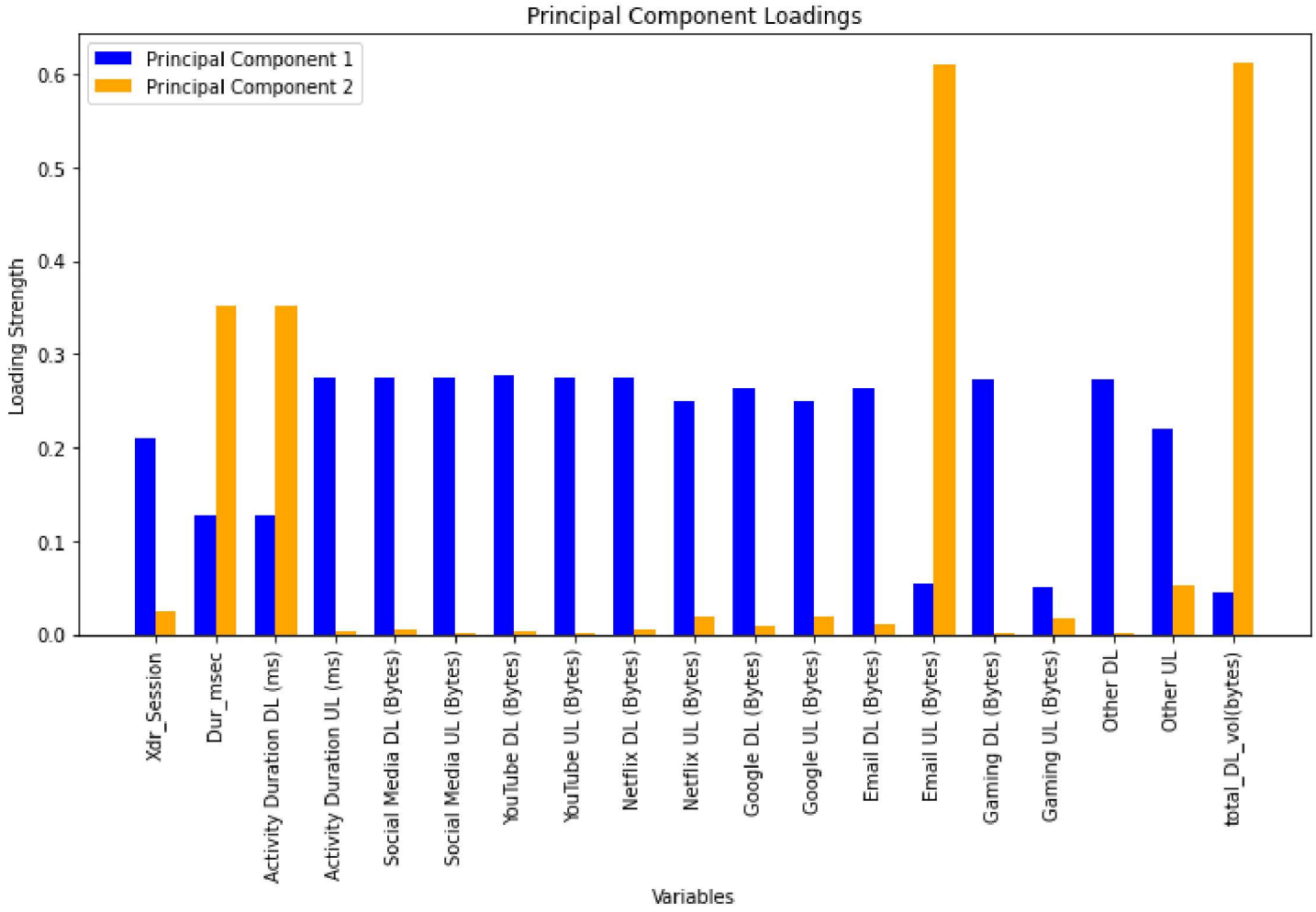
In [89]:
```
plt.scatter(b[:, 0], b[:, 1], c='green', label='Component 1 vs Component 2')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Components')
plt.title('PCA: Transformed Data')
plt.legend()
plt.show()
```



In [90]:
```
# Create a DataFrame to store the variable names
variable_names = pd.DataFrame(columns=['Variable'], data=['Xdr_Session', 'Dur_msec', 'Activity Duration DL (m
       'Activity Duration UL (ms)', 'Social Media DL (Bytes)',
       'Social Media UL (Bytes)', 'YouTube DL (Bytes)', 'YouTube UL (Bytes)',
       'Netflix DL (Bytes)', 'Netflix UL (Bytes)', 'Google DL (Bytes)',
       'Google UL (Bytes)', 'Email DL (Bytes)', 'Email UL (Bytes)',
       'Gaming DL (Bytes)', 'Gaming UL (Bytes)', 'Other DL', 'Other UL',
       'total_DL_vol(bytes)', 'total_Ul_vol(bytes)'])
# Get the loadings from the PCA
loadings = pca.components_

# Determine the number of principal components and variables
num_components = loadings.shape[0]
num_variables = loadings.shape[1]

# Create a list to store the component names
component_names = ['Principal Component 1', 'Principal Component 2']

# Create lists to store the interpretation of loadings
variable_names_list = []
loading_values_pc1 = []
loading_values_pc2 = []

# Iterate through each variable
for variable in range(num_variables):
    # Get the loading for the current variable and component
    loading_pc1 = loadings[0, variable]
    loading_pc2 = loadings[1, variable]
# Determine the strength and direction of the association for PC1
    strength_pc1 = abs(loading_pc1)
    direction_pc1 = "Positive" if loading_pc1 >= 0 else "Negative"
# Determine the strength and direction of the association for PC2
    strength_pc2 = abs(loading_pc2)
    direction_pc2 = "Positive" if loading_pc2 >= 0 else "Negative"
# Get the variable name from the DataFrame
    variable_name = variable_names['Variable'][variable]

# Add the variable name and loading values to the lists
    variable_names_list.append(variable_name)
    loading_values_pc1.append(strength_pc1)
    loading_values_pc2.append(strength_pc2)
# Set the width of the bars
    bar_width = 0.35
```

```
# Set the positions of the x-axis ticks
r1 = np.arange(len(variable_names_list))
r2 = [x + bar_width for x in r1]
```

In [91]:
```
# Create the bar plot
plt.figure(figsize=(12, 6))
plt.bar(r1, loading_values_pc1, color='blue', width=bar_width, label='Principal Component 1')
plt.bar(r2, loading_values_pc2, color='orange', width=bar_width, label='Principal Component 2')
plt.xlabel('Variables')
plt.ylabel('Loading Strength')
plt.title('Principal Component Loadings')
plt.xticks([r + bar_width/2 for r in range(len(variable_names_list))], variable_names_list, rotation=90)
plt.legend()
plt.show()
```



In [ ]:

In [ ]: