Task 4 - Satisfaction Analysis

Assuming that the satisfaction of a user is dependent on user engagement and experience, you're expected in this section to analyze customer satisfaction in depth. The following tasks will guide you:

Based on the engagement analysis + the experience analysis you conducted above,

Task 4.1 - Write a Python program to assign:

f. engagement score to each user. Consider the engagement score as the Euclidean distance between the user data point & the less engaged cluster (use the first clustering for this) (Euclidean Distance)

g. experience score for each user. Consider the experience score as the Euclidean distance between the user data point & the worst experience cluster.

Task 4.2 - Consider the average of both engagement & experience scores as the satisfaction score & report the top 10 satisfied customer

Task 4.3 - Build a regression model of your choice to predict the satisfaction score of a customer.

Task 4.4 - Run a k-means (k=2) on the engagement & the experience score.

Task 4.5 - Aggregate the average satisfaction & experience score per cluster.

Task 4.6 - Export your final table containing all user id + engagement, experience & satisfaction scores in your local MySQL database. Report a screenshot of a select query output on the exported table.

Task 4.7 Model deployment tracking- deploy the model and monitor your model. Here you can use Docker or other MlOps tools which can help you to track your model's change.

Your model tracking report includes code version, start and end time, source, parameters, metrics (loss convergence) and artefacts or any output file regarding each specific run. (CSV file, screenshot)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings("ignore")

data=pd.read_csv('telcom_data.csv')
```

```
data

            Bearer Id            Start  Start ms              End  End
ms  \
0       1.311448e+19   4/4/2019 12:01     770.0  4/25/2019 14:35
662.0
1       1.311448e+19   4/9/2019 13:04     235.0   4/25/2019 8:15
606.0
2       1.311448e+19   4/9/2019 17:42       1.0  4/25/2019 11:58
652.0
3       1.311448e+19   4/10/2019 0:31     486.0   4/25/2019 7:36
171.0
4       1.311448e+19  4/12/2019 20:10     565.0  4/25/2019 10:40
954.0
...              ...              ...       ...              ...     .
..
149996  7.277826e+18   4/29/2019 7:28     451.0   4/30/2019 6:02
214.0
149997  7.349883e+18   4/29/2019 7:28     483.0  4/30/2019 10:41
187.0
149998  1.311448e+19   4/29/2019 7:28     283.0  4/30/2019 10:46
810.0
149999  1.311448e+19   4/29/2019 7:28     696.0  4/30/2019 10:40
327.0
150000           NaN              NaN       NaN              NaN
NaN

        Dur. (ms)          IMSI  MSISDN/Number          IMEI  \
0       1823652.0  2.082014e+14   3.366496e+10  3.552121e+13
1       1365104.0  2.082019e+14   3.368185e+10  3.579401e+13
2       1361762.0  2.082003e+14   3.376063e+10  3.528151e+13
3       1321509.0  2.082014e+14   3.375034e+10  3.535661e+13
4       1089009.0  2.082014e+14   3.369980e+10  3.540701e+13
...           ...           ...            ...           ...
149996    81230.0  2.082022e+14   3.365069e+10  3.548311e+13
149997    97970.0  2.082019e+14   3.366345e+10  3.566051e+13
149998    98249.0  2.082017e+14   3.362189e+10  3.572121e+13
149999    97910.0  2.082021e+14   3.361962e+10  8.618620e+13
150000        NaN           NaN            NaN           NaN

           Last Location Name  ...  Youtube DL (Bytes)  Youtube UL
(Bytes)  \
0       9.16456699548519E+015  ...        1.585461e+07
2.501332e+06
1                      L77566A  ...        2.024740e+07
1.911173e+07
2                      D42335A  ...        1.972566e+07
1.469958e+07
3                      T21824A  ...        2.138812e+07
1.514664e+07
```

```
4                     D88865A  ...        1.525938e+07
1.896287e+07
...                       ...  ...                 ...
...
149996                D20434A  ...        1.619167e+07
1.176343e+07
149997                D10223C  ...        1.387723e+07
8.288284e+06
149998                T51102A  ...        2.266051e+07
1.855903e+06
149999                L88342B  ...        8.817106e+06
8.305402e+06
150000                    NaN  ...        1.163407e+07
1.100941e+07

        Netflix DL (Bytes)  Netflix UL (Bytes)  Gaming DL (Bytes)  \
0             8.198936e+06        9.656251e+06       2.780823e+08
1             1.833841e+07        1.722713e+07       6.087501e+08
2             1.758779e+07        6.163408e+06       2.295846e+08
3             1.399465e+07        1.097942e+06       7.995382e+08
4             1.712458e+07        4.152180e+05       5.277072e+08
...                    ...                 ...                ...
149996        1.788370e+07        1.967816e+07       5.266097e+08
149997        1.935015e+07        2.129315e+07       6.268931e+08
149998        9.963942e+06        5.065760e+06       5.535395e+08
149999        3.322253e+06        1.317259e+07       3.525370e+08
150000        1.162685e+07        1.100175e+07       4.220447e+08

        Gaming UL (Bytes)  Other DL (Bytes)  Other UL (Bytes)  \
0            1.434415e+07      1.717444e+08      8.814393e+06
1            1.170709e+06      5.269042e+08      1.505514e+07
2            3.956300e+05      4.106926e+08      4.215763e+06
3            1.084972e+07      7.490399e+08      1.279728e+07
4            3.529801e+06      5.507095e+08      1.391032e+07
...                   ...               ...               ...
149996       9.197207e+06      3.264510e+06      1.348742e+07
149997       4.735033e+06      7.121804e+08      2.457758e+06
149998       1.339432e+07      1.211009e+08      1.131473e+07
149999       2.529475e+06      8.147131e+08      1.406930e+06
150000       8.288398e+06      4.211005e+08      8.264799e+06

        Total UL (Bytes)  Total DL (Bytes)
0             36749741.0       308879636.0
1             53800391.0       653384965.0
2             27883638.0       279807335.0
3             43324218.0       846028530.0
4             38542814.0       569138589.0
...                  ...               ...
149996        57628851.0       574175259.0
149997        39135081.0       666648844.0
```

```
149998        34912224.0       592786405.0
149999        29626096.0       371895920.0
150000               NaN               NaN

[150001 rows x 55 columns]
```

```
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
 14  TCP DL Retrans. Vol (Bytes)               61855 non-null
float64
 15  TCP UL Retrans. Vol (Bytes)               53352 non-null
float64
 16  DL TP < 50 Kbps (%)                       149247 non-null
float64
 17  50 Kbps < DL TP < 250 Kbps (%)            149247 non-null
```

```
float64
 18  250 Kbps < DL TP < 1 Mbps (%)              149247 non-null
float64
 19  DL TP > 1 Mbps (%)                         149247 non-null
float64
 20  UL TP < 10 Kbps (%)                        149209 non-null
float64
 21  10 Kbps < UL TP < 50 Kbps (%)              149209 non-null
float64
 22  50 Kbps < UL TP < 300 Kbps (%)             149209 non-null
float64
 23  UL TP > 300 Kbps (%)                       149209 non-null
float64
 24  HTTP DL (Bytes)                            68527 non-null
float64
 25  HTTP UL (Bytes)                            68191 non-null
float64
 26  Activity Duration DL (ms)                  150000 non-null
float64
 27  Activity Duration UL (ms)                  150000 non-null
float64
 28  Dur. (ms).1                                150000 non-null
float64
 29  Handset Manufacturer                       149429 non-null  object

 30  Handset Type                               149429 non-null  object

 31  Nb of sec with 125000B < Vol DL            52463 non-null
float64
 32  Nb of sec with 1250B < Vol UL < 6250B      57107 non-null
float64
 33  Nb of sec with 31250B < Vol DL < 125000B   56415 non-null
float64
 34  Nb of sec with 37500B < Vol UL             19747 non-null
float64
 35  Nb of sec with 6250B < Vol DL < 31250B     61684 non-null
float64
 36  Nb of sec with 6250B < Vol UL < 37500B     38158 non-null
float64
 37  Nb of sec with Vol DL < 6250B              149246 non-null
float64
 38  Nb of sec with Vol UL < 1250B              149208 non-null
float64
 39  Social Media DL (Bytes)                    150001 non-null
float64
 40  Social Media UL (Bytes)                    150001 non-null
float64
 41  Google DL (Bytes)                          150001 non-null
float64
```

```
 42  Google UL (Bytes)                          150001 non-null
float64
 43  Email DL (Bytes)                           150001 non-null
float64
 44  Email UL (Bytes)                           150001 non-null
float64
 45  Youtube DL (Bytes)                         150001 non-null
float64
 46  Youtube UL (Bytes)                         150001 non-null
float64
 47  Netflix DL (Bytes)                         150001 non-null
float64
 48  Netflix UL (Bytes)                         150001 non-null
float64
 49  Gaming DL (Bytes)                          150001 non-null
float64
 50  Gaming UL (Bytes)                          150001 non-null
float64
 51  Other DL (Bytes)                           150001 non-null
float64
 52  Other UL (Bytes)                           150001 non-null
float64
 53  Total UL (Bytes)                           150000 non-null
float64
 54  Total DL (Bytes)                           150000 non-null
float64
dtypes: float64(50), object(5)
memory usage: 62.9+ MB

data.describe()

          Bearer Id       Start ms          End ms     Dur. (ms)
IMSI  \
count  1.490100e+05  150000.000000  150000.000000  1.500000e+05
1.494310e+05
mean   1.013887e+19     499.188200     498.800880  1.046086e+05
2.082016e+14
std    2.893173e+18     288.611834     288.097653  8.103762e+04
2.148809e+10
min    6.917538e+18       0.000000       0.000000  7.142000e+03
2.040471e+14
25%    7.349883e+18     250.000000     251.000000  5.744050e+04
2.082014e+14
50%    7.349883e+18     499.000000     500.000000  8.639900e+04
2.082015e+14
75%    1.304243e+19     749.000000     750.000000  1.324302e+05
2.082018e+14
max    1.318654e+19     999.000000     999.000000  1.859336e+06
2.140743e+14
```

```
       MSISDN/Number          IMEI  Avg RTT DL (ms)  Avg RTT UL
(ms)  \
count   1.489350e+05  1.494290e+05    122172.000000    122189.000000

mean    4.188282e+10  4.847455e+13       109.795706        17.662883

std     2.447443e+12  2.241637e+13       619.782739        84.793524

min     3.360100e+10  4.400152e+11         0.000000         0.000000

25%     3.365130e+10  3.546071e+13        32.000000         2.000000

50%     3.366371e+10  3.572201e+13        45.000000         5.000000

75%     3.368349e+10  8.611970e+13        70.000000        15.000000

max     8.823971e+14  9.900120e+13     96923.000000      7120.000000


       Avg Bearer TP DL (kbps)  ...  Youtube DL (Bytes)  Youtube UL
(Bytes)  \
count            150000.000000  ...        1.500010e+05
1.500010e+05
mean              13300.045927  ...        1.163407e+07
1.100941e+07
std               23971.878541  ...        6.710569e+06
6.345423e+06
min                   0.000000  ...        5.300000e+01
1.050000e+02
25%                  43.000000  ...        5.833501e+06
5.517965e+06
50%                  63.000000  ...        1.161602e+07
1.101345e+07
75%               19710.750000  ...        1.744852e+07
1.651556e+07
max              378160.000000  ...        2.325910e+07
2.201196e+07

       Netflix DL (Bytes)  Netflix UL (Bytes)  Gaming DL (Bytes)  \
count        1.500010e+05        1.500010e+05       1.500010e+05
mean         1.162685e+07        1.100175e+07       4.220447e+08
std          6.725218e+06        6.359490e+06       2.439675e+08
min          4.200000e+01        3.500000e+01       2.516000e+03
25%          5.777156e+06        5.475981e+06       2.104733e+08
50%          1.164222e+07        1.099638e+07       4.234081e+08
75%          1.747048e+07        1.650727e+07       6.331742e+08
max          2.325919e+07        2.201196e+07       8.434419e+08

       Gaming UL (Bytes)  Other DL (Bytes)  Other UL (Bytes)  \
count       1.500010e+05      1.500010e+05      1.500010e+05
```

```
mean         8.288398e+06      4.211005e+08      8.264799e+06
std          4.782700e+06      2.432050e+08      4.769004e+06
min          5.900000e+01      3.290000e+03      1.480000e+02
25%          4.128476e+06      2.101869e+08      4.145943e+06
50%          8.291208e+06      4.218030e+08      8.267071e+06
75%          1.243162e+07      6.316918e+08      1.238415e+07
max          1.655879e+07      8.434425e+08      1.655882e+07

       Total UL (Bytes)  Total DL (Bytes)
count      1.500000e+05      1.500000e+05
mean       4.112121e+07      4.546434e+08
std        1.127639e+07      2.441429e+08
min        2.866892e+06      7.114041e+06
25%        3.322201e+07      2.431068e+08
50%        4.114331e+07      4.558411e+08
75%        4.903424e+07      6.657055e+08
max        7.833131e+07      9.029696e+08

[8 rows x 50 columns]
```

```
data.head()
```

```
      Bearer Id            Start  Start ms              End  End ms  \
0  1.311448e+19   4/4/2019 12:01     770.0  4/25/2019 14:35   662.0
1  1.311448e+19   4/9/2019 13:04     235.0   4/25/2019 8:15   606.0
2  1.311448e+19   4/9/2019 17:42       1.0  4/25/2019 11:58   652.0
3  1.311448e+19   4/10/2019 0:31     486.0   4/25/2019 7:36   171.0
4  1.311448e+19  4/12/2019 20:10     565.0  4/25/2019 10:40   954.0

   Dur. (ms)          IMSI  MSISDN/Number          IMEI  \
0  1823652.0  2.082014e+14   3.366496e+10  3.552121e+13
1  1365104.0  2.082019e+14   3.368185e+10  3.579401e+13
2  1361762.0  2.082003e+14   3.376063e+10  3.528151e+13
3  1321509.0  2.082014e+14   3.375034e+10  3.535661e+13
4  1089009.0  2.082014e+14   3.369980e+10  3.540701e+13

      Last Location Name  ...  Youtube DL (Bytes)  Youtube UL (Bytes)
\
0  9.16456699548519E+015  ...           15854611.0           2501332.0

1                L77566A  ...           20247395.0          19111729.0

2                D42335A  ...           19725661.0          14699576.0

3                T21824A  ...           21388122.0          15146643.0

4                D88865A  ...           15259380.0          18962873.0


   Netflix DL (Bytes)  Netflix UL (Bytes)  Gaming DL (Bytes)  \
0           8198936.0           9656251.0        278082303.0
```

```
1          18338413.0          17227132.0        608750074.0
2          17587794.0           6163408.0        229584621.0
3          13994646.0           1097942.0        799538153.0
4          17124581.0            415218.0        527707248.0

   Gaming UL (Bytes)  Other DL (Bytes)  Other UL (Bytes)  Total UL
(Bytes)  \
0         14344150.0       171744450.0         8814393.0
36749741.0
1          1170709.0       526904238.0        15055145.0
53800391.0
2           395630.0       410692588.0         4215763.0
27883638.0
3         10849722.0       749039933.0        12797283.0
43324218.0
4          3529801.0       550709500.0        13910322.0
38542814.0

   Total DL (Bytes)
0       308879636.0
1       653384965.0
2       279807335.0
3       846028530.0
4       569138589.0

[5 rows x 55 columns]

data.tail()

           Bearer Id           Start  Start ms               End  End
ms  \
149996  7.277826e+18  4/29/2019 7:28     451.0   4/30/2019 6:02
214.0
149997  7.349883e+18  4/29/2019 7:28     483.0  4/30/2019 10:41
187.0
149998  1.311448e+19  4/29/2019 7:28     283.0  4/30/2019 10:46
810.0
149999  1.311448e+19  4/29/2019 7:28     696.0  4/30/2019 10:40
327.0
150000           NaN              NaN       NaN               NaN
NaN

        Dur. (ms)          IMSI  MSISDN/Number          IMEI  \
149996    81230.0  2.082022e+14   3.365069e+10  3.548311e+13
149997    97970.0  2.082019e+14   3.366345e+10  3.566051e+13
149998    98249.0  2.082017e+14   3.362189e+10  3.572121e+13
149999    97910.0  2.082021e+14   3.361962e+10  8.618620e+13
150000        NaN           NaN            NaN           NaN

       Last Location Name  ...  Youtube DL (Bytes)  Youtube UL (Bytes)
```

```
\
149996            D20434A  ...        1.619167e+07        1.176343e+07

149997            D10223C  ...        1.387723e+07        8.288284e+06

149998            T51102A  ...        2.266051e+07        1.855903e+06

149999            L88342B  ...        8.817106e+06        8.305402e+06

150000                NaN  ...        1.163407e+07        1.100941e+07


        Netflix DL (Bytes)  Netflix UL (Bytes)  Gaming DL (Bytes)  \
149996        1.788370e+07        1.967816e+07       5.266097e+08
149997        1.935015e+07        2.129315e+07       6.268931e+08
149998        9.963942e+06        5.065760e+06       5.535395e+08
149999        3.322253e+06        1.317259e+07       3.525370e+08
150000        1.162685e+07        1.100175e+07       4.220447e+08

        Gaming UL (Bytes)  Other DL (Bytes)  Other UL (Bytes)  \
149996       9.197207e+06      3.264510e+06      1.348742e+07
149997       4.735033e+06      7.121804e+08      2.457758e+06
149998       1.339432e+07      1.211009e+08      1.131473e+07
149999       2.529475e+06      8.147131e+08      1.406930e+06
150000       8.288398e+06      4.211005e+08      8.264799e+06

        Total UL (Bytes)  Total DL (Bytes)
149996         57628851.0       574175259.0
149997         39135081.0       666648844.0
149998         34912224.0       592786405.0
149999         29626096.0       371895920.0
150000                NaN               NaN

[5 rows x 55 columns]

data.columns

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)',
       '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')
```

Task 4.1 - Write a Python program to assign:

f. engagement score to each user. Consider the engagement score as the Euclidean distance between the user data point & the less engaged cluster (use the first clustering for this) (Euclidean Distance)

g. experience score for each user. Consider the experience score as the Euclidean distance between the user data point & the worst experience cluster.

Engagement analysis

```
data['Sessions Frequency'] = data.groupby('MSISDN/Number')['Bearer
Id'].transform('count')
data['Duration Of Session'] = data['Dur. (ms)']
data['Session Total Traffic']= data['Total UL (Bytes)'] + data['Total
DL (Bytes)']

data['Sessions Frequency']
```

```
0         2.0
1         2.0
2         1.0
3         1.0
4         1.0
         ...
149996    1.0
149997    2.0
149998    1.0
```

```
149999    1.0
150000    NaN
Name: Sessions Frequency, Length: 150001, dtype: float64
```

```
data['Duration Of Session']
```

```
0         1823652.0
1         1365104.0
2         1361762.0
3         1321509.0
4         1089009.0
            ...
149996      81230.0
149997      97970.0
149998      98249.0
149999      97910.0
150000          NaN
Name: Duration Of Session, Length: 150001, dtype: float64
```

```
data['Session Total Traffic']
```

```
0         345629377.0
1         707185356.0
2         307690973.0
3         889352748.0
4         607681403.0
             ...
149996    631804110.0
149997    705783925.0
149998    627698629.0
149999    401522016.0
150000            NaN
Name: Session Total Traffic, Length: 150001, dtype: float64
```

```
data['Sessions Frequency']
data['Duration Of Session']
data['Session Total Traffic']
```

```
0         345629377.0
1         707185356.0
2         307690973.0
3         889352748.0
4         607681403.0
             ...
149996    631804110.0
149997    705783925.0
149998    627698629.0
149999    401522016.0
150000            NaN
Name: Session Total Traffic, Length: 150001, dtype: float64
```

```
data

            Bearer Id            Start  Start ms              End  End
ms  \
0       1.311448e+19   4/4/2019 12:01     770.0  4/25/2019 14:35
662.0
1       1.311448e+19   4/9/2019 13:04     235.0   4/25/2019 8:15
606.0
2       1.311448e+19   4/9/2019 17:42       1.0  4/25/2019 11:58
652.0
3       1.311448e+19   4/10/2019 0:31     486.0   4/25/2019 7:36
171.0
4       1.311448e+19  4/12/2019 20:10     565.0  4/25/2019 10:40
954.0
...              ...              ...       ...              ...     .
..
149996  7.277826e+18   4/29/2019 7:28     451.0   4/30/2019 6:02
214.0
149997  7.349883e+18   4/29/2019 7:28     483.0  4/30/2019 10:41
187.0
149998  1.311448e+19   4/29/2019 7:28     283.0  4/30/2019 10:46
810.0
149999  1.311448e+19   4/29/2019 7:28     696.0  4/30/2019 10:40
327.0
150000           NaN              NaN       NaN              NaN
NaN

        Dur. (ms)          IMSI  MSISDN/Number          IMEI  \
0       1823652.0  2.082014e+14   3.366496e+10  3.552121e+13
1       1365104.0  2.082019e+14   3.368185e+10  3.579401e+13
2       1361762.0  2.082003e+14   3.376063e+10  3.528151e+13
3       1321509.0  2.082014e+14   3.375034e+10  3.535661e+13
4       1089009.0  2.082014e+14   3.369980e+10  3.540701e+13
...           ...           ...            ...           ...
149996    81230.0  2.082022e+14   3.365069e+10  3.548311e+13
149997    97970.0  2.082019e+14   3.366345e+10  3.566051e+13
149998    98249.0  2.082017e+14   3.362189e+10  3.572121e+13
149999    97910.0  2.082021e+14   3.361962e+10  8.618620e+13
150000        NaN           NaN            NaN           NaN

           Last Location Name  ...  Netflix UL (Bytes)  Gaming DL
(Bytes)  \
0       9.16456699548519E+015  ...        9.656251e+06
2.780823e+08
1                      L77566A  ...        1.722713e+07
6.087501e+08
2                      D42335A  ...        6.163408e+06
2.295846e+08
3                      T21824A  ...        1.097942e+06
7.995382e+08
```

```
4                    D88865A  ...        4.152180e+05
5.277072e+08
...                      ...  ...                 ...
...
149996               D20434A  ...        1.967816e+07
5.266097e+08
149997               D10223C  ...        2.129315e+07
6.268931e+08
149998               T51102A  ...        5.065760e+06
5.535395e+08
149999               L88342B  ...        1.317259e+07
3.525370e+08
150000                   NaN  ...        1.100175e+07
4.220447e+08

        Gaming UL (Bytes)  Other DL (Bytes)  Other UL (Bytes)  \
0            1.434415e+07      1.717444e+08      8.814393e+06
1            1.170709e+06      5.269042e+08      1.505514e+07
2            3.956300e+05      4.106926e+08      4.215763e+06
3            1.084972e+07      7.490399e+08      1.279728e+07
4            3.529801e+06      5.507095e+08      1.391032e+07
...                   ...               ...               ...
149996       9.197207e+06      3.264510e+06      1.348742e+07
149997       4.735033e+06      7.121804e+08      2.457758e+06
149998       1.339432e+07      1.211009e+08      1.131473e+07
149999       2.529475e+06      8.147131e+08      1.406930e+06
150000       8.288398e+06      4.211005e+08      8.264799e+06

        Total UL (Bytes)  Total DL (Bytes)  Sessions Frequency  \
0             36749741.0       308879636.0                 2.0
1             53800391.0       653384965.0                 2.0
2             27883638.0       279807335.0                 1.0
3             43324218.0       846028530.0                 1.0
4             38542814.0       569138589.0                 1.0
...                  ...               ...                 ...
149996        57628851.0       574175259.0                 1.0
149997        39135081.0       666648844.0                 2.0
149998        34912224.0       592786405.0                 1.0
149999        29626096.0       371895920.0                 1.0
150000               NaN               NaN                 NaN

        Duration Of Session  Session Total Traffic
0                 1823652.0            345629377.0
1                 1365104.0            707185356.0
2                 1361762.0            307690973.0
3                 1321509.0            889352748.0
4                 1089009.0            607681403.0
...                     ...                    ...
149996              81230.0            631804110.0
149997              97970.0            705783925.0
```

```
149998              98249.0            627698629.0
149999              97910.0            401522016.0
150000                  NaN                    NaN

[150001 rows x 58 columns]
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150001 entries, 0 to 150000
Data columns (total 58 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
 14  TCP DL Retrans. Vol (Bytes)             61855 non-null
float64
 15  TCP UL Retrans. Vol (Bytes)             53352 non-null
float64
 16  DL TP < 50 Kbps (%)                     149247 non-null
float64
 17  50 Kbps < DL TP < 250 Kbps (%)          149247 non-null
```

```
float64
 18  250 Kbps < DL TP < 1 Mbps (%)              149247 non-null
float64
 19  DL TP > 1 Mbps (%)                         149247 non-null
float64
 20  UL TP < 10 Kbps (%)                        149209 non-null
float64
 21  10 Kbps < UL TP < 50 Kbps (%)              149209 non-null
float64
 22  50 Kbps < UL TP < 300 Kbps (%)             149209 non-null
float64
 23  UL TP > 300 Kbps (%)                       149209 non-null
float64
 24  HTTP DL (Bytes)                            68527 non-null
float64
 25  HTTP UL (Bytes)                            68191 non-null
float64
 26  Activity Duration DL (ms)                  150000 non-null
float64
 27  Activity Duration UL (ms)                  150000 non-null
float64
 28  Dur. (ms).1                                150000 non-null
float64
 29  Handset Manufacturer                       149429 non-null  object

 30  Handset Type                               149429 non-null  object

 31  Nb of sec with 125000B < Vol DL            52463 non-null
float64
 32  Nb of sec with 1250B < Vol UL < 6250B      57107 non-null
float64
 33  Nb of sec with 31250B < Vol DL < 125000B   56415 non-null
float64
 34  Nb of sec with 37500B < Vol UL             19747 non-null
float64
 35  Nb of sec with 6250B < Vol DL < 31250B     61684 non-null
float64
 36  Nb of sec with 6250B < Vol UL < 37500B     38158 non-null
float64
 37  Nb of sec with Vol DL < 6250B              149246 non-null
float64
 38  Nb of sec with Vol UL < 1250B              149208 non-null
float64
 39  Social Media DL (Bytes)                    150001 non-null
float64
 40  Social Media UL (Bytes)                    150001 non-null
float64
 41  Google DL (Bytes)                          150001 non-null
float64
```

```
 42  Google UL (Bytes)                          150001 non-null
float64
 43  Email DL (Bytes)                           150001 non-null
float64
 44  Email UL (Bytes)                           150001 non-null
float64
 45  Youtube DL (Bytes)                         150001 non-null
float64
 46  Youtube UL (Bytes)                         150001 non-null
float64
 47  Netflix DL (Bytes)                         150001 non-null
float64
 48  Netflix UL (Bytes)                         150001 non-null
float64
 49  Gaming DL (Bytes)                          150001 non-null
float64
 50  Gaming UL (Bytes)                          150001 non-null
float64
 51  Other DL (Bytes)                           150001 non-null
float64
 52  Other UL (Bytes)                           150001 non-null
float64
 53  Total UL (Bytes)                           150000 non-null
float64
 54  Total DL (Bytes)                           150000 non-null
float64
 55  Sessions Frequency                         148935 non-null
float64
 56  Duration Of Session                        150000 non-null
float64
 57  Session Total Traffic                      150000 non-null
float64
dtypes: float64(53), object(5)
memory usage: 66.4+ MB

data.columns

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)',
```

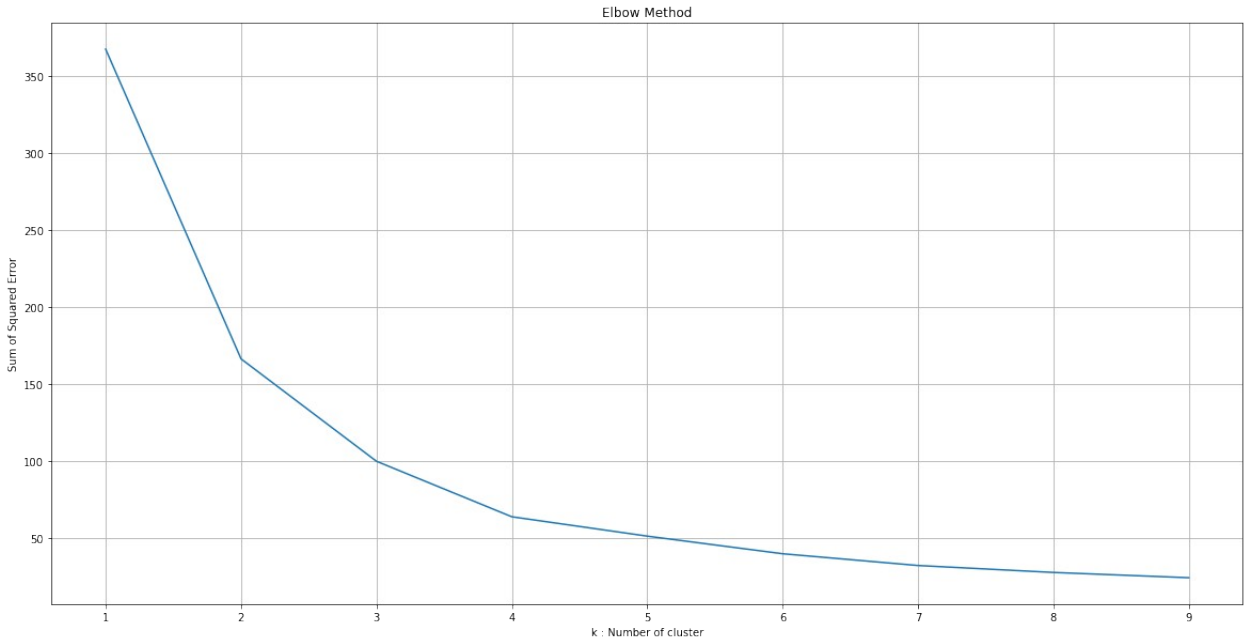
```
       'Activity Duration UL (ms)', 'Dur. (ms).1', 'Handset
Manufacturer',
       'Handset Type', 'Nb of sec with 125000B < Vol DL',
       'Nb of sec with 1250B < Vol UL < 6250B',
       'Nb of sec with 31250B < Vol DL < 125000B',
       'Nb of sec with 37500B < Vol UL',
       'Nb of sec with 6250B < 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)',
       '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)', 'Sessions Frequency',
       'Duration Of Session', 'Session Total Traffic'],
      dtype='object')
```

```
# 1.Aggregate the metrics per customer id (MSISDN)
Aggregate_data= data.groupby('MSISDN/Number').agg({'Sessions
Frequency':'sum',
                                                   'Duration Of
Session':'sum',
                                                   'Session Total
Traffic':'sum'})
Aggregate_data
```

```
                Sessions Frequency  Duration Of Session  Session Total
Traffic
MSISDN/Number

3.360100e+10                   1.0             116720.0
8.786906e+08
3.360100e+10                   1.0             181230.0
1.568596e+08
3.360100e+10                   1.0             134969.0
5.959665e+08
3.360101e+10                   1.0              49878.0
4.223207e+08
3.360101e+10                   4.0              37104.0
1.457411e+09
...                            ...                 ...
...
3.379000e+10                   1.0               8810.0
7.146416e+08
3.379000e+10                   1.0             140988.0
4.803073e+08
3.197021e+12                   1.0             877385.0
```

```
2.321240e+08
3.370000e+14                    1.0              253030.0
5.962878e+08
8.823971e+14                    1.0              869844.0
1.391536e+08

[106856 rows x 3 columns]
```

```
from sklearn.preprocessing import MinMaxScaler,StandardScaler
mms = MinMaxScaler() # Normalization
ss = StandardScaler() # Standardization
from sklearn.cluster import KMeans
kmeans=KMeans()

Aggregate_data['Sessions
Frequency']=mms.fit_transform(Aggregate_data[['Sessions Frequency']])
Aggregate_data['Duration Of
Session']=mms.fit_transform(Aggregate_data[['Duration Of Session']])
Aggregate_data['Session Total
Traffic']=mms.fit_transform(Aggregate_data[['Session Total Traffic']])

sse = {};
kmax = 10
fig = plt.subplots(figsize = (20,10))

# Elbow Method :
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(Aggregate_data)
    sse[k] = kmeans.inertia_ # Inertia: Sum of distances of samples to
their closest cluster center
sns.lineplot(x = list(sse.keys()), y = list(sse.values()));
plt.title('Elbow Method')
plt.xlabel("k : Number of cluster")
plt.ylabel("Sum of Squared Error")
plt.grid()
```




```
#2 Normalize the engagement metrics
Scaler = StandardScaler()
normalized_data = pd.DataFrame(Scaler.fit_transform(Aggregate_data),
columns=Aggregate_data.columns, index=Aggregate_data.index)
normalized_data
```

MSISDN/Number	Sessions Frequency	Duration Of Session	Session Total Traffic
3.360100e+10	-0.321123	-0.158014	0.382297
3.360100e+10	-0.321123	0.188148	-1.087666
3.360100e+10	-0.321123	-0.060090	-0.193453
3.360101e+10	-0.321123	-0.516690	-0.547071
3.360101e+10	0.285601	-0.585235	1.560825
...	...	...	...
3.379000e+10	-0.321123	-0.737061	0.048222
3.379000e+10	-0.321123	-0.027791	-0.428985
3.197021e+12	-0.321123	3.923731	-0.934395
3.370000e+14	-0.321123	0.573428	-0.192798

```
8.823971e+14            -0.321123             3.883266              -
1.123723

[106856 rows x 3 columns]
```

```
kmeans = KMeans(n_clusters=3)
kmeans.fit(normalized_data)
Aggregate_data['cluster']=kmeans.labels_
kmeans.labels_
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
lowest_engagement =
Aggregate_data.groupby('cluster').get_group(0).mean()
lowest_engagement
```

```
Sessions Frequency       0.004787
Duration Of Session      0.005789
Session Total Traffic    0.060390
cluster                  0.000000
dtype: float64
```

```
def get_engagement_score(data, lowest):
    x = float(lowest['Sessions Frequency'])
    y = float(lowest['Duration Of Session'])
    z = float(lowest['Session Total Traffic'])
    new_data = data.copy()
    new_data['engagement score'] = ((data['Sessions Frequency'] -
x)**2 + (data['Duration Of Session'] - y)**2 + (data['Session Total
Traffic'] - z)**2)**0.5
    return new_data
engagement_scored_data = get_engagement_score(Aggregate_data,
lowest_engagement)
engagement_scored_data.head(15)
```

```
                Sessions Frequency  Duration Of Session  Session Total
Traffic  \
MSISDN/Number

3.360100e+10             0.003086             0.005908
0.095931
3.360100e+10             0.003086             0.009387
0.014026
3.360100e+10             0.003086             0.006892
0.063851
3.360101e+10             0.003086             0.002304
0.044148
3.360101e+10             0.012346             0.001615
0.161598
3.360101e+10             0.012346             0.013309
0.066035
```

```
3.360101e+10              0.012346              0.006536
0.070518
3.360101e+10              0.003086              0.004273
0.033974
3.360101e+10              0.012346              0.026342
0.108577
3.360102e+10              0.003086              0.006347
0.079339
3.360102e+10              0.003086              0.013009
0.009273
3.360102e+10              0.003086              0.001686
0.067723
3.360102e+10              0.003086              0.004865
0.050967
3.360102e+10              0.003086              0.012680
0.036953
3.360103e+10              0.003086              0.008780
0.022362
```

```
                cluster  engagement score
MSISDN/Number
3.360100e+10          0          0.035583
3.360100e+10          0          0.046534
3.360100e+10          0          0.004011
3.360101e+10          0          0.016699
3.360101e+10          0          0.101576
3.360101e+10          0          0.012065
3.360101e+10          0          0.012660
3.360101e+10          0          0.026514
3.360101e+10          1          0.052930
3.360102e+10          0          0.019034
3.360102e+10          0          0.051652
3.360102e+10          0          0.008574
3.360102e+10          0          0.009620
3.360102e+10          0          0.024488
3.360103e+10          0          0.038183
```

```
lowest_experiance =
Aggregate_data.groupby('cluster').get_group(0).mean()
lowest_experiance
```

```
Sessions Frequency       0.004787
Duration Of Session      0.005789
Session Total Traffic    0.060390
cluster                  0.000000
dtype: float64
```

```
def get_experiance_score(data, low):
    x = float(low['Sessions Frequency'])
    y = float(low['Duration Of Session'])
```

```
    z = float(low['Session Total Traffic'])
    new_data = data.copy()
    new_data['experience score'] = ((data['Sessions Frequency'] -
x)**2 + (data['Duration Of Session'] - y)**2 \
                                + (data['Session Total Traffic'] - z)**2
)**0.5
    return new_data
experiance_scored_data = get_experiance_score(Aggregate_data,
lowest_experiance)
experiance_scored_data.head(15)
```

MSISDN/Number	Sessions Frequency	Duration Of Session	Session Total Traffic	\
3.360100e+10	0.003086	0.005908	0.095931	
3.360100e+10	0.003086	0.009387	0.014026	
3.360100e+10	0.003086	0.006892	0.063851	
3.360101e+10	0.003086	0.002304	0.044148	
3.360101e+10	0.012346	0.001615	0.161598	
3.360101e+10	0.012346	0.013309	0.066035	
3.360101e+10	0.012346	0.006536	0.070518	
3.360101e+10	0.003086	0.004273	0.033974	
3.360101e+10	0.012346	0.026342	0.108577	
3.360102e+10	0.003086	0.006347	0.079339	
3.360102e+10	0.003086	0.013009	0.009273	
3.360102e+10	0.003086	0.001686	0.067723	
3.360102e+10	0.003086	0.004865	0.050967	
3.360102e+10	0.003086	0.012680	0.036953	
3.360103e+10	0.003086	0.008780	0.022362	

MSISDN/Number	cluster	experience score
3.360100e+10	0	0.035583
3.360100e+10	0	0.046534

```
3.360100e+10          0           0.004011
3.360101e+10          0           0.016699
3.360101e+10          0           0.101576
3.360101e+10          0           0.012065
3.360101e+10          0           0.012660
3.360101e+10          0           0.026514
3.360101e+10          1           0.052930
3.360102e+10          0           0.019034
3.360102e+10          0           0.051652
3.360102e+10          0           0.008574
3.360102e+10          0           0.009620
3.360102e+10          0           0.024488
3.360103e+10          0           0.038183
```

Task 4.2 Consider the average of both engagement & experience scores as the satisfaction score & report the top 10 satisfied customer

```
satisfaction_data= pd.merge(engagement_scored_data["engagement
score"], experiance_scored_data['experience score'],
on='MSISDN/Number')
satisfaction_data['satisfaction score']=
(satisfaction_data['engagement score'] + satisfaction_data['experience
score']) / 2
satisfaction_data['satisfaction score']
```

```
MSISDN/Number
3.360100e+10    0.035583
3.360100e+10    0.046534
3.360100e+10    0.004011
3.360101e+10    0.016699
3.360101e+10    0.101576
                   ...
3.379000e+10    0.017942
3.379000e+10    0.009914
3.197021e+12    0.055905
3.370000e+14    0.008420
8.823971e+14    0.063257
Name: satisfaction score, Length: 106856, dtype: float64
```

```
satisfaction_data.sort_values(by='satisfaction score',
ascending=False).head(10)
```

```
                engagement score  experience score  satisfaction score
MSISDN/Number
3.362578e+10            1.608124          1.608124             1.608124
3.361489e+10            1.397200          1.397200             1.397200
3.362632e+10            1.384011          1.384011             1.384011
3.376054e+10            1.238314          1.238314             1.238314
3.365973e+10            1.147996          1.147996             1.147996
3.367588e+10            1.110050          1.110050             1.110050
3.376041e+10            0.909593          0.909593             0.909593
3.366716e+10            0.901315          0.901315             0.901315
3.366646e+10            0.882150          0.882150             0.882150
3.366471e+10            0.819151          0.819151             0.819151
```

```
top_10_satisfied_customers=satisfaction_data.sort_values(by='satisfact
ion score', ascending=False).head(10)
print(top_10_satisfied_customers)
```

```
                engagement score  experience score  satisfaction score
MSISDN/Number
3.362578e+10            1.608124          1.608124             1.608124
3.361489e+10            1.397200          1.397200             1.397200
3.362632e+10            1.384011          1.384011             1.384011
3.376054e+10            1.238314          1.238314             1.238314
3.365973e+10            1.147996          1.147996             1.147996
3.367588e+10            1.110050          1.110050             1.110050
3.376041e+10            0.909593          0.909593             0.909593
3.366716e+10            0.901315          0.901315             0.901315
3.366646e+10            0.882150          0.882150             0.882150
3.366471e+10            0.819151          0.819151             0.819151
```

Task 4.3 - Build a regression model of your choice to predict the satisfaction score of a customer

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

#1.TCP retransmission
data['avg_tcp_retransmission'] = (data['TCP DL Retrans. Vol (Bytes)']
+ data['TCP UL Retrans. Vol (Bytes)']) / 2
data['avg_tcp_retransmission']
```

```
0        NaN
1        NaN
2        NaN
```

```
3         NaN
4         NaN
           ..
149996    NaN
149997    NaN
149998    NaN
149999    NaN
150000    NaN
Name: avg_tcp_retransmission, Length: 150001, dtype: float64
```

```
data['avg_tcp_retransmission'].isnull().sum()
```

```
99530
```

```
data['avg_rtt'] = (data['Avg RTT DL (ms)']+ data['Avg RTT UL (ms)']) /
2
data['avg_rtt']
```

```
0          23.5
1          35.0
2           NaN
3           NaN
4           NaN
           ...
149996     16.0
149997     14.5
149998     24.5
149999     21.0
150000      NaN
Name: avg_rtt, Length: 150001, dtype: float64
```

```
# Calculate the average RTT
data['avg_rtt'].fillna(data['avg_rtt'].mean(), inplace=True)
data['avg_rtt']
```

```
0          23.500000
1          35.000000
2          63.512443
3          63.512443
4          63.512443
             ...
149996     16.000000
149997     14.500000
149998     24.500000
149999     21.000000
150000     63.512443
Name: avg_rtt, Length: 150001, dtype: float64
```

```
data.columns
```

```
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)',
       '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)', 'Sessions Frequency',
       'Duration Of Session', 'Session Total Traffic',
       'avg_tcp_retransmission', 'avg_rtt'],
      dtype='object')
```

```
# Calculate the average throughput
data['avg_throughput'] = data["Avg Bearer TP DL (kbps)"] + data["Avg
Bearer TP UL (kbps)"]
data['avg_throughput']
```

```
0          67.0
1          42.0
2          15.0
3          88.0
4          15.0
          ...
149996    117.0
149997     77.0
149998     90.0
149999     71.0
```

```
150000      NaN
Name: avg_throughput, Length: 150001, dtype: float64
```

```
data['avg_throughput'].fillna(data['avg_throughput'].mean(),
inplace=True)
data['avg_throughput']
```

```
0            67.000000
1            42.000000
2            15.000000
3            88.000000
4            15.000000
              ...
149996      117.000000
149997       77.000000
149998       90.000000
149999       71.000000
150000    15070.474573
Name: avg_throughput, Length: 150001, dtype: float64
```

```
data['Handset Type'].fillna(data['Handset Type'].mode()[0],
inplace=True)
data['Handset Type']
```

```
0          Samsung Galaxy A5 Sm-A520F
1         Samsung Galaxy J5 (Sm-J530)
2            Samsung Galaxy A8 (2018)
3                            undefined
4                    Samsung Sm-G390F
                     ...
149996    Apple iPhone 8 Plus (A1897)
149997        Apple iPhone Se (A1723)
149998        Apple iPhone Xs (A2097)
149999                  Huawei Fig-Lx1
150000                Huawei B528S-23A
Name: Handset Type, Length: 150001, dtype: object
```

```
# Treat outliers by replacing with the mean of the corresponding
variable
def replace_outliers_with_mean(column):
    mean = column.mean()
    std = column.std()
    column[np.abs(column - mean) > 3 * std] = mean

replace_outliers_with_mean(data['avg_tcp_retransmission'])
replace_outliers_with_mean(data['avg_rtt'])
replace_outliers_with_mean(data['avg_throughput'])

# Aggregate the information per customer
grouped_data = data.groupby('MSISDN/Number').agg({
    'avg_tcp_retransmission': 'mean',
```

```
    'avg_rtt': 'mean',
    'Handset Type': lambda x: x.mode()[0],
    'avg_throughput': 'mean'
})

grouped_data
```

```
                avg_tcp_retransmission    avg_rtt  \
MSISDN/Number
3.360100e+10                       NaN  23.000000
3.360100e+10                       NaN  15.500000
3.360100e+10                       NaN  63.512443
3.360101e+10                       NaN  42.000000
3.360101e+10                 4685416.0  29.750000
...                                ...        ...
3.379000e+10                  109022.5  26.000000
3.379000e+10                       NaN  20.000000
3.197021e+12                       NaN  63.512443
3.370000e+14                       NaN  63.512443
8.823971e+14                       NaN  63.512443

                                   Handset Type  avg_throughput
MSISDN/Number
3.360100e+10     Huawei P20 Lite Huawei Nova 3E            76.0
3.360100e+10              Apple iPhone 7 (A1778)            99.0
3.360100e+10                          undefined            97.0
3.360101e+10             Apple iPhone 5S (A1457)           248.0
3.360101e+10             Apple iPhone Se (A1723)         28422.0
...                                          ...            ...
3.379000e+10                 Huawei Honor 9 Lite         10365.0
3.379000e+10         Apple iPhone 8 Plus (A1897)           116.0
3.197021e+12   Quectel Wireless. Quectel Ec25-E             1.0
3.370000e+14                   Huawei B525S-23A            33.0
8.823971e+14   Quectel Wireless. Quectel Ec21-E             2.0

[106856 rows x 4 columns]
```

```
data
```

```
           Bearer Id            Start  Start ms               End  End
ms  \
0       1.311448e+19   4/4/2019 12:01     770.0  4/25/2019 14:35
662.0
1       1.311448e+19   4/9/2019 13:04     235.0   4/25/2019 8:15
606.0
2       1.311448e+19   4/9/2019 17:42       1.0  4/25/2019 11:58
652.0
3       1.311448e+19   4/10/2019 0:31     486.0   4/25/2019 7:36
171.0
4       1.311448e+19  4/12/2019 20:10     565.0  4/25/2019 10:40
```

```
954.0
...              ...              ...       ...               ...     .
..
149996  7.277826e+18   4/29/2019 7:28     451.0   4/30/2019 6:02
214.0
149997  7.349883e+18   4/29/2019 7:28     483.0  4/30/2019 10:41
187.0
149998  1.311448e+19   4/29/2019 7:28     283.0  4/30/2019 10:46
810.0
149999  1.311448e+19   4/29/2019 7:28     696.0  4/30/2019 10:40
327.0
150000           NaN              NaN       NaN               NaN
NaN

        Dur. (ms)          IMSI  MSISDN/Number          IMEI  \
0       1823652.0  2.082014e+14   3.366496e+10  3.552121e+13
1       1365104.0  2.082019e+14   3.368185e+10  3.579401e+13
2       1361762.0  2.082003e+14   3.376063e+10  3.528151e+13
3       1321509.0  2.082014e+14   3.375034e+10  3.535661e+13
4       1089009.0  2.082014e+14   3.369980e+10  3.540701e+13
...           ...           ...            ...           ...
149996    81230.0  2.082022e+14   3.365069e+10  3.548311e+13
149997    97970.0  2.082019e+14   3.366345e+10  3.566051e+13
149998    98249.0  2.082017e+14   3.362189e+10  3.572121e+13
149999    97910.0  2.082021e+14   3.361962e+10  8.618620e+13
150000        NaN           NaN            NaN           NaN

           Last Location Name  ...  Other DL (Bytes)  Other UL (Bytes)
\
0       9.16456699548519E+015  ...      1.717444e+08      8.814393e+06

1                   L77566A  ...      5.269042e+08      1.505514e+07

2                   D42335A  ...      4.106926e+08      4.215763e+06

3                   T21824A  ...      7.490399e+08      1.279728e+07

4                   D88865A  ...      5.507095e+08      1.391032e+07

...                     ...  ...               ...               ...

149996              D20434A  ...      3.264510e+06      1.348742e+07

149997              D10223C  ...      7.121804e+08      2.457758e+06

149998              T51102A  ...      1.211009e+08      1.131473e+07

149999              L88342B  ...      8.147131e+08      1.406930e+06

150000                  NaN  ...      4.211005e+08      8.264799e+06
```

```
        Total UL (Bytes)  Total DL (Bytes)  Sessions Frequency  \
0             36749741.0       308879636.0                 2.0
1             53800391.0       653384965.0                 2.0
2             27883638.0       279807335.0                 1.0
3             43324218.0       846028530.0                 1.0
4             38542814.0       569138589.0                 1.0
...                  ...               ...                 ...
149996        57628851.0       574175259.0                 1.0
149997        39135081.0       666648844.0                 2.0
149998        34912224.0       592786405.0                 1.0
149999        29626096.0       371895920.0                 1.0
150000               NaN               NaN                 NaN

        Duration Of Session  Session Total Traffic
avg_tcp_retransmission  \
0                 1823652.0            345629377.0
NaN
1                 1365104.0            707185356.0
NaN
2                 1361762.0            307690973.0
NaN
3                 1321509.0            889352748.0
NaN
4                 1089009.0            607681403.0
NaN
...                     ...                    ...
...
149996              81230.0            631804110.0
NaN
149997              97970.0            705783925.0
NaN
149998              98249.0            627698629.0
NaN
149999              97910.0            401522016.0
NaN
150000                  NaN                    NaN
NaN

          avg_rtt  avg_throughput
0       23.500000        67.000000
1       35.000000        42.000000
2       63.512443        15.000000
3       63.512443        88.000000
4       63.512443        15.000000
...           ...              ...
149996  16.000000       117.000000
149997  14.500000        77.000000
149998  24.500000        90.000000
```

```
149999  21.000000       71.000000
150000  63.512443    15070.474573

[150001 rows x 61 columns]
```

```
engagement_data=['Bearer Id', 'Dur. (ms)',' Total UL (Bytes)','Total DL (Bytes)']
engagement_data
```

```
['Bearer Id', 'Dur. (ms)', ' Total UL (Bytes)', 'Total DL (Bytes)']
```

```
experiance_data=['avg_rtt','avg_tcp_retransmission', 'avg_throughput']
experiance_data
```

```
['avg_rtt', 'avg_tcp_retransmission', 'avg_throughput']
```

```
data=satisfaction_data.sort_values(by='satisfaction score', ascending=False).head(10)
data
```

MSISDN/Number	engagement score	experience score	satisfaction score
3.362578e+10	1.608124	1.608124	1.608124
3.361489e+10	1.397200	1.397200	1.397200
3.362632e+10	1.384011	1.384011	1.384011
3.376054e+10	1.238314	1.238314	1.238314
3.365973e+10	1.147996	1.147996	1.147996
3.367588e+10	1.110050	1.110050	1.110050
3.376041e+10	0.909593	0.909593	0.909593
3.366716e+10	0.901315	0.901315	0.901315
3.366646e+10	0.882150	0.882150	0.882150
3.366471e+10	0.819151	0.819151	0.819151

```
X = data[['engagement score', 'experience score']]
y = data['satisfaction score']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a linear regression model
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
# Make predictions on the testing set
y_pred = model.predict(X_test)

y_pred
```

```
array([0.88214999, 1.39720008])
```

```
y_test
```

```
MSISDN/Number
3.366646e+10    0.88215
3.361489e+10    1.39720
Name: satisfaction score, dtype: float64
```

```
# Evaluate the model
MSE = mean_squared_error(y_test, y_pred)
print('Mean Squared Error:', MSE)
```

```
Mean Squared Error: 2.465190328815662e-32
```

```
from sklearn.metrics import mean_absolute_error

# Calculate MAE
MAE = mean_absolute_error(y_test, y_pred)

print("Mean Absolute Error (MAE):", MAE)
```

```
Mean Absolute Error (MAE): 1.1102230246251565e-16
```

```
from sklearn.metrics import r2_score

r2 = r2_score(y_test, y_pred)

print("R-squared:", r2)
```

```
R-squared: 1.0
```

Task 4.4 - Run a k-means (k=2) on the engagement & the experience score .

```
df=data[['engagement score','experience score']]
Z=data.values

cluster_labels = kmeans.labels_

Scaler = StandardScaler()
normalized_data = pd.DataFrame(Scaler.fit_transform(df),
columns=df.columns, index=df.index)
normalized_data
```

```
               engagement score  experience score
MSISDN/Number
3.362578e+10           1.858602          1.858602
3.361489e+10           1.021541          1.021541
3.362632e+10           0.969199          0.969199
```

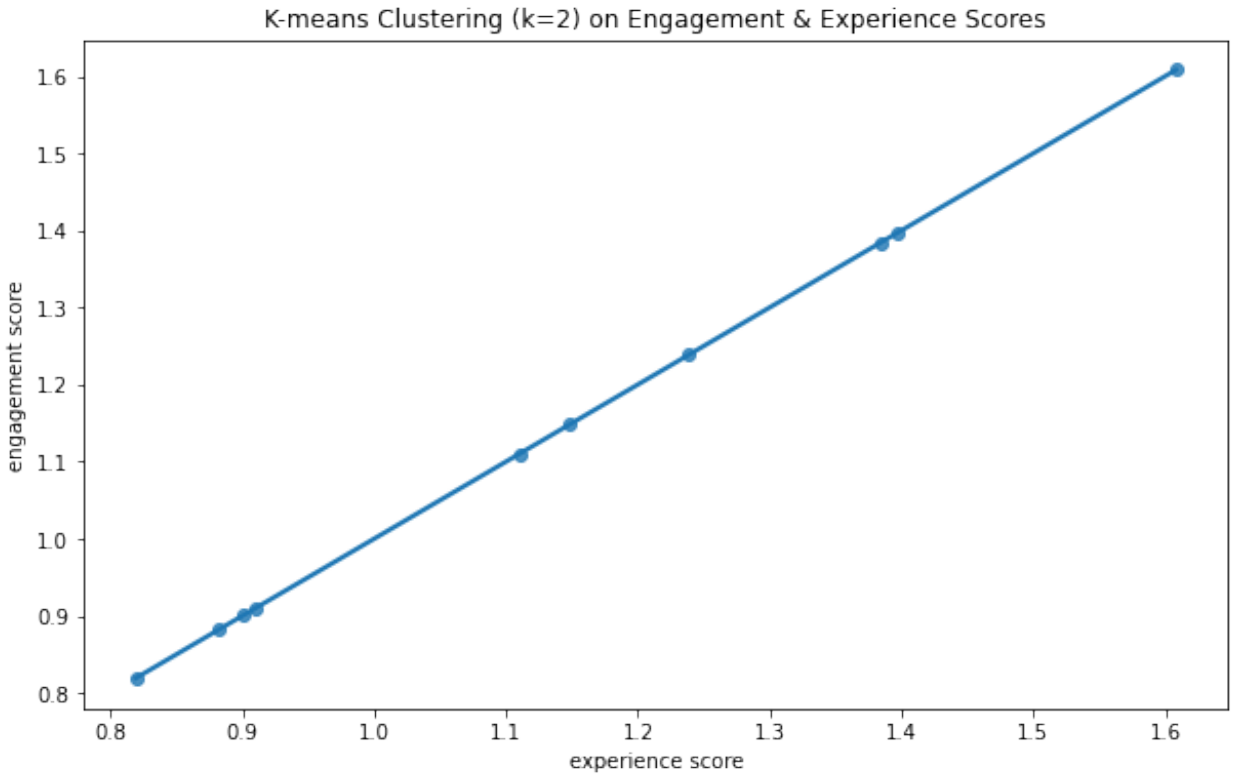
```
3.376054e+10           0.390996          0.390996
3.365973e+10           0.032563          0.032563
3.367588e+10          -0.118027         -0.118027
3.376041e+10          -0.913547         -0.913547
3.366716e+10          -0.946397         -0.946397
3.366646e+10          -1.022456         -1.022456
3.366471e+10          -1.272472         -1.272472
```

```
kmeans = KMeans(n_clusters=2)
kmeans.fit(normalized_data)
df['cluster'] = kmeans.labels_
```

```
df['cluster']
```

```
MSISDN/Number
3.362578e+10    1
3.361489e+10    1
3.362632e+10    1
3.376054e+10    1
3.365973e+10    0
3.367588e+10    0
3.376041e+10    0
3.366716e+10    0
3.366646e+10    0
3.366471e+10    0
Name: cluster, dtype: int32
```

```
plt.figure(figsize=(10,6))
sns.regplot(data=data,y="engagement score",x="experience
score").set(title='K-means Clustering (k=2) on Engagement & Experience
Scores')
plt.show()
```

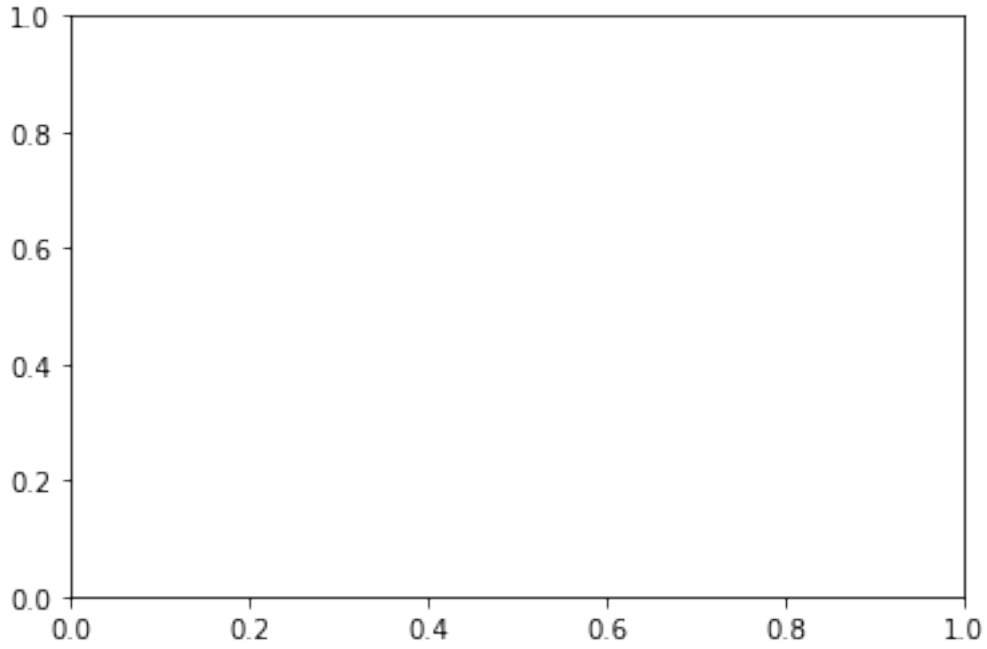



```
# Plotting the scatter plot
plt.scatter(Z[:, 0], Z[:, 1], c=cluster_labels, cmap='viridis')

# Adding labels and title
plt.xlabel('Engagement Score')
plt.ylabel('Experience Score')
plt.title('Scatter Plot: Engagement Score vs Experience Score')

# Displaying the plot
plt.show()
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
~\anaconda3\lib\site-packages\matplotlib\axes\_axes.py in
_parse_scatter_color_args(c, edgecolors, kwargs, xsize,
get_next_color_func)
   4349             try:  # Is 'c' acceptable as PathCollection
facecolors?
-> 4350                 colors = mcolors.to_rgba_array(c)
   4351             except (TypeError, ValueError) as err:

~\anaconda3\lib\site-packages\matplotlib\colors.py in to_rgba_array(c,
alpha)
    384     else:
```

```
--> 385         rgba = np.array([to_rgba(cc) for cc in c])
    386

~\anaconda3\lib\site-packages\matplotlib\colors.py in <listcomp>(.0)
    384     else:
--> 385         rgba = np.array([to_rgba(cc) for cc in c])
    386

~\anaconda3\lib\site-packages\matplotlib\colors.py in to_rgba(c,
alpha)
    205     if rgba is None:  # Suppress exception chaining of cache
lookup failure.
--> 206         rgba = _to_rgba_no_colorcycle(c, alpha)
    207         try:

~\anaconda3\lib\site-packages\matplotlib\colors.py in
_to_rgba_no_colorcycle(c, alpha)
    283     if not np.iterable(c):
--> 284         raise ValueError(f"Invalid RGBA argument: {orig_c!r}")
    285     if len(c) not in [3, 4]:

ValueError: Invalid RGBA argument: 0.0

The above exception was the direct cause of the following exception:

ValueError                                Traceback (most recent call
last)
~\AppData\Local\Temp/ipykernel_9988/3400543450.py in <module>
      1 # Plotting the scatter plot
----> 2 plt.scatter(Z[:, 0], Z[:, 1], c=cluster_labels,
cmap='viridis')
      3
      4 # Adding labels and title
      5 plt.xlabel('Engagement Score')

~\anaconda3\lib\site-packages\matplotlib\pyplot.py in scatter(x, y, s,
c, marker, cmap, norm, vmin, vmax, alpha, linewidths, edgecolors,
plotnonfinite, data, **kwargs)
   3066         vmin=None, vmax=None, alpha=None, linewidths=None, *,
   3067         edgecolors=None, plotnonfinite=False, data=None,
**kwargs):
-> 3068     __ret = gca().scatter(
   3069         x, y, s=s, c=c, marker=marker, cmap=cmap, norm=norm,
   3070         vmin=vmin, vmax=vmax, alpha=alpha,
linewidths=linewidths,

~\anaconda3\lib\site-packages\matplotlib\__init__.py in inner(ax,
data, *args, **kwargs)
   1359     def inner(ax, *args, data=None, **kwargs):
   1360         if data is None:
```

```
-> 1361             return func(ax, *map(sanitize_sequence, args),
**kwargs)
   1362
   1363         bound = new_sig.bind(ax, *args, **kwargs)

~\anaconda3\lib\site-packages\matplotlib\axes\_axes.py in
scatter(self, x, y, s, c, marker, cmap, norm, vmin, vmax, alpha,
linewidths, edgecolors, plotnonfinite, **kwargs)
   4514             orig_edgecolor = kwargs.get('edgecolor', None)
   4515         c, colors, edgecolors = \
-> 4516             self._parse_scatter_color_args(
   4517                 c, edgecolors, kwargs, x.size,
   4518
get_next_color_func=self._get_patches_for_fill.get_next_color)

~\anaconda3\lib\site-packages\matplotlib\axes\_axes.py in
_parse_scatter_color_args(c, edgecolors, kwargs, xsize,
get_next_color_func)
   4354                 else:
   4355                     if not valid_shape:
-> 4356                         raise invalid_shape_exception(c.size,
xsize) from err
   4357                     # Both the mapping *and* the RGBA
conversion failed: pretty
   4358                     # severe failure => one may appreciate a
verbose feedback.

ValueError: 'c' argument has 106856 elements, which is inconsistent
with 'x' and 'y' with size 10.
```



Task 4.5 - Aggregate the average satisfaction & experience score per cluster.

```
df['clusters'] = kmeans.labels_
df['clusters'].value_counts()

result = df.groupby('clusters').mean()
result

df.columns

sns.pairplot(data=data,vars=['engagement score','experience score'])
plt.show()

data

df
```

Task 4.6 - Export your final table containing all user id + engagement, experience & satisfaction scores in your local MySQL database. Report a screenshot of a select query output on the exported table.

```

```