Name : Harleen Kaur Taunque

Student Id : 20811951

Name : Somesh Gupta

Student Id: 20817245

Question 1

In [1]:
```
import sys
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
from sklearn.impute import SimpleImputer
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.impute import KNNImputer
from impyute.imputation.cs import mice
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import IsolationForest
```

In [2]:
```
#Load DataA.csv
dataA = pd.read_csv('DataA.csv')
```

In [3]:
```
print("\n dataA: \n", dataA)
```

```
 dataA: 
        Unnamed: 0  fea.1  fea.2  fea.3  fea.4   fea.5  fea.6  fea.7  fea.8  \
0               1 -153.0  414.0  939.0 -161.0  1007.0   99.0 -210.0  948.0
1               2 -150.0  420.0  939.0 -177.0  1008.0  103.0 -207.0  939.0
2               3 -160.0  432.0  941.0 -162.0   982.0   98.0 -198.0  936.0
3               4 -171.0  432.0  911.0 -174.0   999.0  115.0 -187.0  918.0
4               5 -171.0    NaN  929.0 -189.0  1004.0  104.0 -198.0  939.0
...           ...    ...    ...    ...    ...     ...    ...    ...    ...
18995       18996    NaN    NaN    NaN    NaN     NaN    NaN    NaN    NaN
18996       18997    NaN    NaN    NaN    NaN     NaN    NaN    NaN    NaN
18997       18998    NaN    NaN    NaN    NaN     NaN    NaN    NaN    NaN
18998       18999    NaN    NaN    NaN    NaN     NaN    NaN    NaN    NaN
18999       19000    NaN    NaN    NaN    NaN     NaN    NaN    NaN    NaN

       fea.9  ...  fea.72  fea.73  fea.74  fea.75  fea.76  fea.77  fea.78  \
0      333.0  ...   655.0  -316.0  -302.0  -617.0  -955.0  -264.0    23.0
1      316.0  ...   655.0  -309.0  -304.0  -619.0  -955.0  -265.0    19.0
2      315.0  ...   655.0  -302.0  -308.0  -621.0  -966.0  -270.0    10.0
3      338.0  ...   655.0  -293.0  -312.0  -622.0  -964.0  -269.0    14.0
4      350.0  ...   655.0  -284.0  -318.0  -624.0  -966.0  -262.0    24.0
...      ...  ...     ...     ...     ...     ...     ...     ...     ...
18995    NaN  ...     NaN     NaN     NaN     NaN     NaN     NaN     NaN
18996    NaN  ...     NaN     NaN     NaN     NaN     NaN     NaN     NaN
18997    NaN  ...     NaN     NaN     NaN     NaN     NaN     NaN     NaN
18998    NaN  ...     NaN     NaN     NaN     NaN     NaN     NaN     NaN
18999    NaN  ...     NaN     NaN     NaN     NaN     NaN     NaN     NaN

       fea.79  fea.80  fea.81
0       -29.0    36.0    24.0
1       -31.0    47.0     3.0
2       -38.0    20.0     0.0
3       -51.0    33.0    -1.0
4       -40.0     1.0     4.0
...       ...     ...     ...
18995     NaN     NaN     NaN
18996     NaN     NaN     NaN
18997     NaN     NaN     NaN
18998     NaN     NaN     NaN
18999     NaN     NaN     NaN

[19000 rows x 82 columns]
```

As we can see in the above dataset there are many missing values in each of the column.

In [4]:
```
# Display all columns
pd.set_option('display.max_columns', None)
```

Shape of the Dataset

In [5]:
```
print("\n Dimension of DataA: ",dataA.shape)
```

```
 Dimension of DataA:  (19000, 82)
```

The Dataset has 19000 samples and 82 features. However there is a Unamed column 'Unnamed: 0' which is just the count which needs to be removed from the dataset

Checking all the Columns

In [6]:
```
# Columns of DataA
print(" \n Columns of DataA: \n",dataA.columns)
```

```
 Columns of DataA:
 Index(['Unnamed: 0', 'fea.1', 'fea.2', 'fea.3', 'fea.4', 'fea.5', 'fea.6',
       'fea.7', 'fea.8', 'fea.9', 'fea.10', 'fea.11', 'fea.12', 'fea.13',
       'fea.14', 'fea.15', 'fea.16', 'fea.17', 'fea.18', 'fea.19', 'fea.20',
       'fea.21', 'fea.22', 'fea.23', 'fea.24', 'fea.25', 'fea.26', 'fea.27',
       'fea.28', 'fea.29', 'fea.30', 'fea.31', 'fea.32', 'fea.33', 'fea.34',
       'fea.35', 'fea.36', 'fea.37', 'fea.38', 'fea.39', 'fea.40', 'fea.41',
       'fea.42', 'fea.43', 'fea.44', 'fea.45', 'fea.46', 'fea.47', 'fea.48',
       'fea.49', 'fea.50', 'fea.51', 'fea.52', 'fea.53', 'fea.54', 'fea.55',
       'fea.56', 'fea.57', 'fea.58', 'fea.59', 'fea.60', 'fea.61', 'fea.62',
       'fea.63', 'fea.64', 'fea.65', 'fea.66', 'fea.67', 'fea.68', 'fea.69',
       'fea.70', 'fea.71', 'fea.72', 'fea.73', 'fea.74', 'fea.75', 'fea.76',
       'fea.77', 'fea.78', 'fea.79', 'fea.80', 'fea.81'],
      dtype='object')
```

Removing 'Unnamed: 0' columns from the dataset

In [7]:
```
dataA_updated = dataA.drop(['Unnamed: 0'], axis=1)
print("Updated Dataset after removal of 'Unnamed: 0' column: \n", dataA_update
d.head(20))
```

```
Updated Dataset after removal of 'Unnamed: 0' column:
     fea.1  fea.2  fea.3  fea.4   fea.5  fea.6  fea.7   fea.8  fea.9  fea.10  \
0  -153.0  414.0  939.0 -161.0  1007.0   99.0 -210.0   948.0  333.0   -19.0
1  -150.0  420.0  939.0 -177.0  1008.0  103.0 -207.0   939.0  316.0     9.0
2  -160.0  432.0  941.0 -162.0   982.0   98.0 -198.0   936.0  315.0   -10.0
3  -171.0  432.0  911.0 -174.0   999.0  115.0 -187.0   918.0  338.0    34.0
4  -171.0    NaN  929.0 -189.0  1004.0  104.0 -198.0   939.0  350.0    60.0
5  -171.0  432.0  924.0 -179.0  1011.0   85.0 -204.0   945.0  336.0    94.0
6  -169.0  429.0  949.0 -175.0  1007.0  102.0 -188.0   914.0  322.0   154.0
7  -160.0  423.0  927.0 -195.0   996.0  123.0 -213.0   925.0  302.0   128.0
8  -163.0  432.0  929.0 -178.0   994.0  101.0 -186.0   946.0  296.0   166.0
9  -156.0  415.0  936.0 -186.0  1014.0  111.0 -195.0   960.0  280.0   202.0
10 -153.0  413.0  923.0 -187.0   993.0   91.0 -193.0   970.0  282.0   233.0
11 -168.0  412.0  904.0 -194.0   989.0  115.0 -198.0   960.0  269.0   267.0
12 -166.0  442.0  926.0 -191.0  1001.0  114.0 -199.0   950.0  296.0   360.0
13 -162.0  447.0  920.0 -218.0  1000.0  110.0 -235.0   948.0  256.0   339.0
14 -184.0  442.0  941.0 -237.0   992.0  144.0 -238.0   940.0  244.0   390.0
15 -157.0  427.0  925.0 -245.0   986.0  127.0 -228.0   932.0  217.0   410.0
16 -158.0  427.0  905.0 -218.0   990.0  111.0 -216.0   984.0  285.0   454.0
17 -153.0  451.0  889.0 -260.0   967.0  112.0 -213.0  1001.0  253.0   513.0
18 -150.0  443.0  928.0 -243.0   968.0  130.0 -242.0   959.0  309.0   586.0
19 -151.0  442.0  930.0 -260.0   991.0   92.0 -248.0   969.0  304.0   563.0

    fea.11  fea.12  fea.13  fea.14  fea.15  fea.16  fea.17  fea.18  fea.19  \
0   -587.0   810.0   902.0  -140.0  -468.0   -28.0  1016.0  -191.0   358.0
1   -605.0   835.0   897.0  -136.0  -473.0   -38.0   989.0  -160.0   346.0
2   -580.0   802.0   902.0  -139.0  -454.0   -19.0   992.0  -170.0   345.0
3   -579.0   849.0   910.0  -151.0  -460.0   -18.0  1015.0  -185.0   345.0
4   -584.0   843.0   917.0  -151.0  -463.0   -12.0   996.0  -209.0   345.0
5   -583.0   827.0   892.0  -140.0  -455.0   -32.0  1012.0  -204.0   345.0
6   -578.0   805.0   910.0  -161.0  -454.0   -26.0   996.0  -219.0   345.0
7   -566.0   775.0   914.0  -153.0  -466.0   -36.0   980.0  -214.0   336.0
8   -567.0   784.0   907.0  -159.0  -479.0   -54.0   991.0  -233.0   345.0
9   -549.0   810.0   908.0  -149.0  -464.0   -62.0   983.0  -232.0   345.0
10  -537.0   790.0   915.0  -147.0  -444.0   -74.0   981.0  -243.0   342.0
11  -567.0   768.0   906.0  -133.0  -463.0   -69.0   995.0  -237.0   338.0
12  -526.0   752.0   902.0  -148.0  -458.0   -34.0   959.0  -253.0   345.0
13  -491.0   767.0   914.0  -161.0  -465.0   -40.0   962.0  -264.0   349.0
14  -516.0   720.0   896.0  -143.0  -447.0   -89.0   994.0  -249.0   349.0
15  -440.0   726.0   895.0  -142.0  -446.0  -107.0  1016.0  -269.0   348.0
16  -391.0   702.0   916.0  -132.0  -443.0   -78.0   990.0  -260.0   359.0
17  -268.0   693.0   933.0  -147.0  -432.0   -97.0   970.0  -280.0   373.0
18  -268.0   765.0   906.0  -172.0  -465.0   -93.0   979.0  -260.0   352.0
19  -214.0   743.0   901.0  -170.0  -443.0   -79.0   969.0  -283.0   349.0

    fea.20  fea.21  fea.22  fea.23  fea.24  fea.25  fea.26  fea.27  fea.28  \
0    895.0  -189.0  -511.0  -782.0  -212.0  -241.0   472.0   938.0  -882.0
1    889.0  -215.0  -609.0  -773.0  -202.0  -255.0   509.0   934.0  -878.0
2    898.0  -200.0  -606.0  -753.0  -116.0  -272.0   492.0   919.0  -864.0
3    888.0  -220.0  -638.0  -718.0  -354.0  -275.0   517.0   919.0  -847.0
4    886.0  -204.0  -723.0  -679.0  -220.0  -271.0   525.0   923.0  -862.0
5    895.0  -211.0  -787.0  -687.0  -184.0  -298.0   548.0   934.0  -869.0
6    896.0  -212.0  -718.0  -690.0  -227.0  -279.0   541.0   890.0  -851.0
7    895.0  -234.0  -722.0  -611.0  -240.0  -255.0   549.0   918.0  -865.0
8    886.0  -227.0  -746.0  -560.0  -231.0  -263.0   539.0   899.0  -874.0
9    874.0  -202.0  -790.0  -653.0  -118.0  -246.0   530.0   905.0  -870.0
10   872.0  -213.0  -865.0  -562.0   -93.0  -268.0   531.0   858.0  -880.0
11   888.0  -213.0  -868.0  -486.0   -51.0  -295.0   556.0   909.0  -872.0
12   888.0  -223.0  -840.0  -495.0    58.0  -267.0   534.0   897.0  -872.0
13   895.0  -202.0  -845.0  -530.0     4.0  -241.0   523.0   871.0  -872.0
14   878.0  -214.0  -820.0  -483.0    42.0  -258.0   581.0   802.0  -881.0
15   879.0  -217.0  -778.0  -375.0    77.0  -224.0   553.0   867.0  -882.0
16   870.0  -215.0  -917.0  -186.0   241.0  -348.0   535.0   907.0  -893.0
```

In [8]:
```
# Dimension of the updated Dataset
print("\n Dimension of Updated DataA: ",dataA_updated.shape)
```

```
 Dimension of Updated DataA:  (19000, 81)
```

The Column 'Unnamed: 0' has been removed from the dataset as we can see in the above data and now the dimension of the dataset has become (19000, 81) which means it has 19000 samples and 81 features in total

Checking Missing Values in the Dataset

In [9]:
```
# Missing Values in the Dataset
print("\n Missing values in the DataA: \n",dataA_updated.isnull())
```

```
 Missing values in the DataA:
        fea.1  fea.2  fea.3  fea.4  fea.5  fea.6  fea.7  fea.8  fea.9  fea.10
\
0      False  False  False  False  False  False  False  False  False   False
1      False  False  False  False  False  False  False  False  False   False
2      False  False  False  False  False  False  False  False  False   False
3      False  False  False  False  False  False  False  False  False   False
4      False   True  False  False  False  False  False  False  False   False
...      ...    ...    ...    ...    ...    ...    ...    ...    ...     ...
18995   True   True   True   True   True   True   True   True   True    True
18996   True   True   True   True   True   True   True   True   True    True
18997   True   True   True   True   True   True   True   True   True    True
18998   True   True   True   True   True   True   True   True   True    True
18999   True   True   True   True   True   True   True   True   True    True

       fea.11  fea.12  fea.13  fea.14  fea.15  fea.16  fea.17  fea.18  fea.19
\
0       False   False   False   False   False   False   False   False   False
1       False   False   False   False   False   False   False   False   False
2       False   False   False   False   False   False   False   False   False
3       False   False   False   False   False   False   False   False   False
4       False   False   False   False   False   False   False   False   False
...       ...     ...     ...     ...     ...     ...     ...     ...     ...
18995    True    True    True    True    True    True    True    True    True
18996    True    True    True    True    True    True    True    True    True
18997    True    True    True    True    True    True    True    True    True
18998    True    True    True    True    True    True    True    True    True
18999    True    True    True    True    True    True    True    True    True

       fea.20  fea.21  fea.22  fea.23  fea.24  fea.25  fea.26  fea.27  fea.28
\
0       False   False   False   False   False   False   False   False   False
1       False   False   False   False   False   False   False   False   False
2       False   False   False   False   False   False   False   False   False
3       False   False   False   False   False   False   False   False   False
4       False   False   False   False   False   False   False   False   False
...       ...     ...     ...     ...     ...     ...     ...     ...     ...
18995    True    True    True    True    True    True    True    True    True
18996    True    True    True    True    True    True    True    True    True
18997    True    True    True    True    True    True    True    True    True
18998    True    True    True    True    True    True    True    True    True
18999    True    True    True    True    True    True    True    True    True

       fea.29  fea.30  fea.31  fea.32  fea.33  fea.34  fea.35  fea.36  fea.37
\
0       False   False   False   False   False    True    True    True   False
1       False   False   False   False   False    True    True    True   False
2       False   False   False   False   False    True    True    True   False
3       False   False   False   False   False    True    True    True   False
4       False   False   False   False   False    True    True    True   False
...       ...     ...     ...     ...     ...     ...     ...     ...     ...
18995    True    True    True    True    True    True    True    True    True
18996    True    True    True    True    True    True    True    True    True
18997    True    True    True    True    True    True    True    True    True
18998    True    True    True    True    True    True    True    True    True
18999    True    True    True    True    True    True    True    True    True

       fea.38  fea.39  fea.40  fea.41  fea.42  fea.43  fea.44  fea.45  fea.46
\
0       False   False   False   False   False   False   False   False   False
1       False   False   False   False   False   False   False   False   False
2       False   False   False   False   False   False   False   False   False
3       False   False   False   False   False   False   False   False   False
```

Here We can see in the dataset there are many missing values as the dataset contains value as 'True' in some rows and columns after applying 'dataA_updated.isnull()'.

Number of Missing values per columns

In [10]:
```
# Find the Total number of missing values in each column
print("\n Number of missing values in each Column: \n", dataA_updated.isnull().
sum())
```

```
 Number of missing values in each Column: 
 fea.1     1187
fea.2     1188
fea.3     1187
fea.4      800
fea.5      800
          ... 
fea.77     773
fea.78     773
fea.79     773
fea.80     773
fea.81     773
Length: 81, dtype: int64
```

As we can see from the above table that the dataset contains too many missing values and each feature has different number of missing values. As from the table it is clear that feature.1 has 1187 missing values, feature.2 has 1188 missing values while from feature.37 to feature.81 all have a missing value of 773 which is the lowest value missing among all the features. Whereas feature.34 to feature.36 have 18999 missing values which are highest among all the features in the entire dataset.

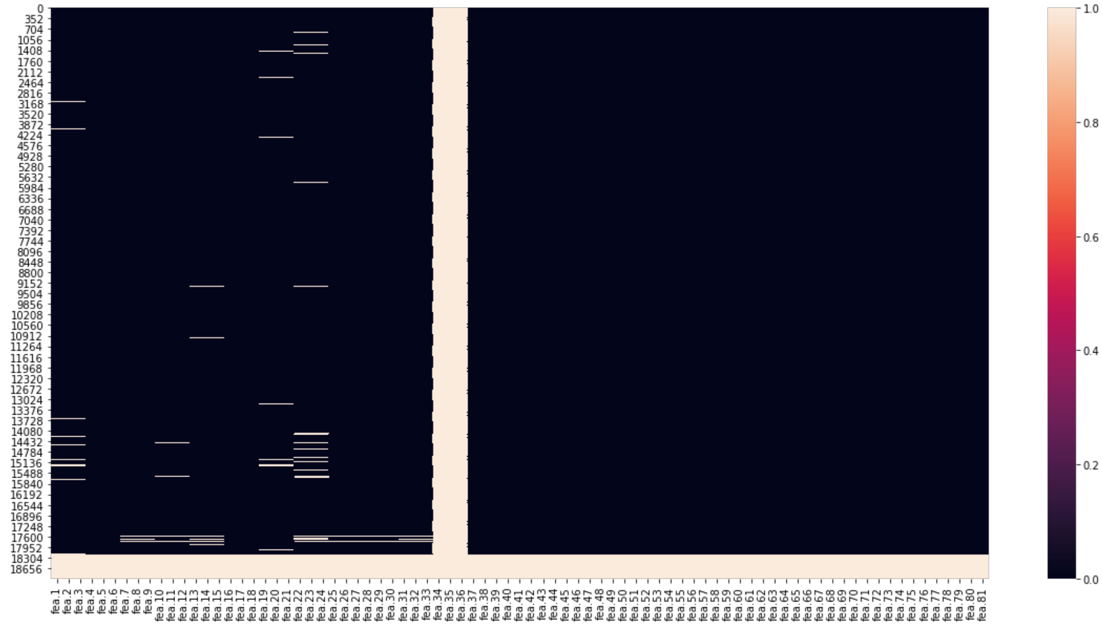
Total number of Missing values in the dataset

In [11]:
```
print("\n Total number of Missing values in the dataset: ", dataA_updated.isnul
l().sum().sum())
```

```
 Total number of Missing values in the dataset:  124053
```

In total there are 124053 missing values in the dataset

In [12]:
```
# Heat Map showing number of Missing Values per Column
plt.subplots(figsize=(20,10))
sns.heatmap(dataA_updated.isnull())
plt.show()
```



As it can be seen from the heat map that Feature 34, 35 and 36 are missing so these features have to be removed

Statistical Description of the Dataset with missing values

In [13]:
```
# Statistical analysis of the Dataset
dataA_updated.describe()
```

Out[13]:

	fea.1	fea.2	fea.3	fea.4	fea.5	fea.6	fea.7
count	17813.000000	17812.000000	17813.000000	18200.000000	18200.000000	18200.000000	18099.000000
mean	-132.812384	698.264485	597.541402	-307.128462	909.548077	-32.760824	61.974363
std	284.183187	375.672475	396.654659	183.151634	193.963300	254.001018	317.393784
min	-2724.000000	-855.000000	-2196.000000	-1365.000000	-245.000000	-920.000000	-1580.000000
25%	-179.000000	360.000000	304.000000	-409.000000	860.000000	-144.000000	-131.000000
50%	-100.000000	811.000000	574.000000	-266.000000	969.500000	-39.000000	70.000000
75%	-15.000000	984.000000	955.000000	-167.000000	1006.000000	45.000000	251.000000
max	1887.000000	2531.000000	2941.000000	609.000000	1833.000000	1215.000000	1490.000000

As we can see from the above statistical description of the dataset it clear that the rows of feature.34, feature.35 and feature.36 is entirely missing so there is no statistical value for those columns. We have to remove the entire feature.34, feature.35 and feature.36 from the dataset.

Drop feature.34, feature.35 and feature.36

In [14]:
```
dataA_updated_redDim = dataA_updated.drop(['fea.34', 'fea.35', 'fea.36'], axis=
1)
print("\n Dimension of Reduced Dataset: ", dataA_updated_redDim.shape)
print("\n Reduced Dataset: \n", dataA_updated_redDim)
```

```
 Dimension of Reduced Dataset:  (19000, 78)

 Reduced Dataset:
        fea.1  fea.2  fea.3  fea.4   fea.5  fea.6  fea.7  fea.8  fea.9  fea.10
\
0     -153.0  414.0  939.0 -161.0  1007.0   99.0 -210.0  948.0  333.0   -19.0
1     -150.0  420.0  939.0 -177.0  1008.0  103.0 -207.0  939.0  316.0     9.0
2     -160.0  432.0  941.0 -162.0   982.0   98.0 -198.0  936.0  315.0   -10.0
3     -171.0  432.0  911.0 -174.0   999.0  115.0 -187.0  918.0  338.0    34.0
4     -171.0    NaN  929.0 -189.0  1004.0  104.0 -198.0  939.0  350.0    60.0
...      ...    ...    ...    ...     ...    ...    ...    ...    ...     ...
18995    NaN    NaN    NaN    NaN     NaN    NaN    NaN    NaN    NaN     NaN
18996    NaN    NaN    NaN    NaN     NaN    NaN    NaN    NaN    NaN     NaN
18997    NaN    NaN    NaN    NaN     NaN    NaN    NaN    NaN    NaN     NaN
18998    NaN    NaN    NaN    NaN     NaN    NaN    NaN    NaN    NaN     NaN
18999    NaN    NaN    NaN    NaN     NaN    NaN    NaN    NaN    NaN     NaN

       fea.11  fea.12  fea.13  fea.14  fea.15  fea.16  fea.17  fea.18  fea.19
\
0      -587.0   810.0   902.0  -140.0  -468.0   -28.0  1016.0  -191.0   358.0
1      -605.0   835.0   897.0  -136.0  -473.0   -38.0   989.0  -160.0   346.0
2      -580.0   802.0   902.0  -139.0  -454.0   -19.0   992.0  -170.0   345.0
3      -579.0   849.0   910.0  -151.0  -460.0   -18.0  1015.0  -185.0   345.0
4      -584.0   843.0   917.0  -151.0  -463.0   -12.0   996.0  -209.0   345.0
...       ...     ...     ...     ...     ...     ...     ...     ...     ...
18995     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
18996     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
18997     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
18998     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
18999     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN

       fea.20  fea.21  fea.22  fea.23  fea.24  fea.25  fea.26  fea.27  fea.28
\
0       895.0  -189.0  -511.0  -782.0  -212.0  -241.0   472.0   938.0  -882.0
1       889.0  -215.0  -609.0  -773.0  -202.0  -255.0   509.0   934.0  -878.0
2       898.0  -200.0  -606.0  -753.0  -116.0  -272.0   492.0   919.0  -864.0
3       888.0  -220.0  -638.0  -718.0  -354.0  -275.0   517.0   919.0  -847.0
4       886.0  -204.0  -723.0  -679.0  -220.0  -271.0   525.0   923.0  -862.0
...       ...     ...     ...     ...     ...     ...     ...     ...     ...
18995     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
18996     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
18997     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
18998     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
18999     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN

       fea.29  fea.30  fea.31  fea.32  fea.33  fea.37  fea.38  fea.39  fea.40
\
0       -61.0   543.0   141.0  -150.0   965.0  -989.0  -158.0    41.0    78.0
1       -50.0   515.0   130.0  -179.0   995.0  -991.0  -157.0    43.0    48.0
2       -56.0   540.0   122.0  -162.0   946.0  -988.0  -171.0    41.0    68.0
3       -66.0   546.0   145.0  -144.0   939.0  -994.0  -159.0    42.0    32.0
4       -70.0   531.0   153.0  -167.0   970.0 -1002.0  -161.0    42.0    58.0
...       ...     ...     ...     ...     ...     ...     ...     ...     ...
18995     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
18996     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
18997     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
18998     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
18999     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN

       fea.41  fea.42  fea.43  fea.44  fea.45  fea.46  fea.47  fea.48  fea.49
\
0       -49.0   -13.0   643.0   104.0   924.0  -717.0    62.0  -691.0    63.0
1       -85.0     1.0   644.0   107.0   922.0  -718.0    61.0  -690.0    63.0
```

As it can be seen from the above dataset that the Feature.34, Feature.35 and feature.36 has been dropped from the dataset.

In [15]:
```
# Total missing value check after removing the Features
print("\n Number of missing values after feature removal: ", dataA_updated_redD
im.isnull().sum().sum())
```

```
 Number of missing values after feature removal:  67056
```

The dataset still contains 67056 missing values in the other Feature sets. So we will apply Imputation of missing values learned during lectures.

Imputing Missing Values

1.A Replace missing data by Mean values for each columns

In [16]:
```
# Using Simple Imputation Technique for replacing the missing values with mean
values
si = SimpleImputer(strategy='mean')
si.fit(dataA_updated_redDim)
dataA_Mean_SiImputation = si.transform(dataA_updated_redDim)
```

In [17]:
```
print("\n Data after Mean Imputation: \n", dataA_Mean_SiImputation)
```

```
 Data after Mean Imputation:
 [[-153.          414.          939.         ...  -29.
    36.           24.        ]
 [-150.          420.          939.         ...  -31.
    47.            3.        ]
 [-160.          432.          941.         ...  -38.
    20.            0.        ]
 ...
 [-132.81238421  698.26448462  597.54140235 ...  -18.09952269
     4.67125693   20.72683382]
 [-132.81238421  698.26448462  597.54140235 ...  -18.09952269
     4.67125693   20.72683382]
 [-132.81238421  698.26448462  597.54140235 ...  -18.09952269
     4.67125693   20.72683382]]
```

1.B Using fillna for mean imputation and Checking for Missing Values after Imputation

In [18]:
```
# Copy data in another variable
data_forMeanImputing = dataA_updated_redDim.copy(deep = True)
```

In [19]:
```
# Mean Method for fill na
data_forMeanImputing.fillna(data_forMeanImputing.mean(), inplace=True)
print("\n Data after Mean imputation: \n",data_forMeanImputing)
```

```
 Data after Mean imputation:
            fea.1       fea.2       fea.3       fea.4        fea.5  \
0     -153.000000  414.000000  939.000000 -161.000000  1007.000000
1     -150.000000  420.000000  939.000000 -177.000000  1008.000000
2     -160.000000  432.000000  941.000000 -162.000000   982.000000
3     -171.000000  432.000000  911.000000 -174.000000   999.000000
4     -171.000000  698.264485  929.000000 -189.000000  1004.000000
...           ...         ...         ...         ...          ...
18995 -132.812384  698.264485  597.541402 -307.128462   909.548077
18996 -132.812384  698.264485  597.541402 -307.128462   909.548077
18997 -132.812384  698.264485  597.541402 -307.128462   909.548077
18998 -132.812384  698.264485  597.541402 -307.128462   909.548077
18999 -132.812384  698.264485  597.541402 -307.128462   909.548077

            fea.6       fea.7       fea.8       fea.9      fea.10      fea.11
\
0       99.000000 -210.000000  948.000000  333.000000  -19.000000 -587.000000
1      103.000000 -207.000000  939.000000  316.000000    9.000000 -605.000000
2       98.000000 -198.000000  936.000000  315.000000  -10.000000 -580.000000
3      115.000000 -187.000000  918.000000  338.000000   34.000000 -579.000000
4      104.000000 -198.000000  939.000000  350.000000   60.000000 -584.000000
...           ...         ...         ...         ...         ...         ...
18995  -32.760824   61.974363  899.313498   81.650478  356.638752  330.971569
18996  -32.760824   61.974363  899.313498   81.650478  356.638752  330.971569
18997  -32.760824   61.974363  899.313498   81.650478  356.638752  330.971569
18998  -32.760824   61.974363  899.313498   81.650478  356.638752  330.971569
18999  -32.760824   61.974363  899.313498   81.650478  356.638752  330.971569

          fea.12      fea.13     fea.14      fea.15     fea.16       fea.17  \
0      810.00000  902.000000 -140.00000 -468.000000 -28.000000  1016.000000
1      835.00000  897.000000 -136.00000 -473.000000 -38.000000   989.000000
2      802.00000  902.000000 -139.00000 -454.000000 -19.000000   992.000000
3      849.00000  910.000000 -151.00000 -460.000000 -18.000000  1015.000000
4      843.00000  917.000000 -151.00000 -463.000000 -12.000000   996.000000
...          ...         ...        ...         ...        ...          ...
18995  584.83313  583.589861  421.62429  -38.719833  71.807768   780.358257
18996  584.83313  583.589861  421.62429  -38.719833  71.807768   780.358257
18997  584.83313  583.589861  421.62429  -38.719833  71.807768   780.358257
18998  584.83313  583.589861  421.62429  -38.719833  71.807768   780.358257
18999  584.83313  583.589861  421.62429  -38.719833  71.807768   780.358257

           fea.18      fea.19      fea.20      fea.21      fea.22      fea.23
\
0     -191.000000  358.000000  895.000000 -189.000000 -511.000000 -782.000000
1     -160.000000  346.000000  889.000000 -215.000000 -609.000000 -773.000000
2     -170.000000  345.000000  898.000000 -200.000000 -606.000000 -753.000000
3     -185.000000  345.000000  888.000000 -220.000000 -638.000000 -718.000000
4     -209.000000  345.000000  886.000000 -204.000000 -723.000000 -679.000000
...           ...         ...         ...         ...         ...         ...
18995 -461.605043  274.476843  893.086451  -89.299933 -622.139334  233.764044
18996 -461.605043  274.476843  893.086451  -89.299933 -622.139334  233.764044
18997 -461.605043  274.476843  893.086451  -89.299933 -622.139334  233.764044
18998 -461.605043  274.476843  893.086451  -89.299933 -622.139334  233.764044
18999 -461.605043  274.476843  893.086451  -89.299933 -622.139334  233.764044

           fea.24      fea.25      fea.26      fea.27      fea.28     fea.29  \
0     -212.000000 -241.000000  472.000000  938.000000 -882.000000  -61.00000
1     -202.000000 -255.000000  509.000000  934.000000 -878.000000  -50.00000
2     -116.000000 -272.000000  492.000000  919.000000 -864.000000  -56.00000
3     -354.000000 -275.000000  517.000000  919.000000 -847.000000  -66.00000
4     -220.000000 -271.000000  525.000000  923.000000 -862.000000  -70.00000
...           ...         ...         ...         ...         ...        ...
18995 -131.400068 -360.661825  875.943278  269.614934 -570.500166  432.33442
```

In [20]:
```
# Check for missing values in the dataset
print("\n Missing Value check in the dataset: ", data_forMeanImputing.isnull().
sum().sum())
```

```
 Missing Value check in the dataset:  0
```

In [21]:
```
# Statistical Table for the Dataset after Mean imputation
print(data_forMeanImputing.describe())
```

```
             fea.1        fea.2        fea.3        fea.4        fea.5  \
count  19000.000000  19000.000000  19000.000000  19000.000000  19000.000000
mean    -132.812384    698.264485    597.541402   -307.128462    909.548077
std      275.162567    363.737566    384.063939    179.254138    189.835729
min    -2724.000000   -855.000000  -2196.000000  -1365.000000   -245.000000
25%     -172.000000    366.000000    322.750000   -402.000000    868.000000
50%     -110.000000    756.000000    597.541402   -277.000000    964.000000
75%      -20.000000    977.000000    950.000000   -169.000000   1005.000000
max     1887.000000   2531.000000   2941.000000    609.000000   1833.000000

             fea.6        fea.7        fea.8        fea.9       fea.10  \
count  19000.000000  19000.000000  19000.000000  19000.000000  19000.000000
mean     -32.760824     61.974363    899.313498     81.650478    356.638752
std      248.595835    309.776406    192.105395    320.034742    334.377778
min     -920.000000  -1580.000000   -149.000000  -1624.000000  -1792.000000
25%     -139.000000   -123.000000    861.000000   -143.000000    170.000000
50%      -32.760824     61.974363    940.000000     68.000000    359.000000
75%       40.000000    239.000000    994.000000    303.000000    570.000000
max     1215.000000   1490.000000   1682.000000   1096.000000   2202.000000

            fea.11       fea.12       fea.13       fea.14       fea.15  \
count  19000.000000  19000.000000  19000.000000  19000.000000  19000.000000
mean     330.971569    584.833130    583.589861    421.624290    -38.719833
std      529.666403    274.889595    298.989418    528.066821    482.015391
min    -1545.000000  -1079.000000  -1710.000000  -1120.000000  -1634.000000
25%     -125.000000    382.000000    361.000000   -110.000000   -456.000000
50%      337.000000    594.000000    594.000000    481.000000    -38.719833
75%      770.000000    804.000000    809.000000    863.000000    245.000000
max     2047.000000   2152.000000   2408.000000   2212.000000   2620.000000

            fea.16       fea.17       fea.18       fea.19       fea.20  \
count  19000.000000  19000.000000  19000.000000  19000.000000  19000.000000
mean      71.807768    780.358257   -461.605043    274.476843    893.086451
std      253.440945    285.451810    255.698513    289.947729    210.202643
min    -1089.000000   -289.000000  -1379.000000  -2269.000000   -786.000000
25%       11.000000    708.000000   -637.000000    259.000000    881.000000
50%       86.000000    895.000000   -412.000000    299.000000    917.000000
75%      174.000000    964.000000   -287.000000    352.000000    943.000000
max      878.000000   1653.000000   1292.000000   2335.000000   2466.000000

            fea.21       fea.22       fea.23       fea.24       fea.25  \
count  19000.000000  19000.000000  19000.000000  19000.000000  19000.000000
mean     -89.299933   -622.139334    233.764044   -131.400068   -360.661825
std      214.959699    346.464727    566.656792    443.772537    218.011487
min    -1944.000000  -3172.000000  -2324.000000  -2734.000000  -1873.000000
25%     -186.000000   -859.000000   -156.000000   -444.000000   -467.000000
50%      -81.000000   -622.139334    233.764044   -131.400068   -353.000000
75%       28.000000   -393.000000    695.000000    156.000000   -212.000000
max     1968.000000   2229.000000   2677.000000   2401.000000    711.000000

            fea.26       fea.27       fea.28       fea.29       fea.30  \
count  19000.000000  19000.000000  19000.000000  19000.000000  19000.000000
mean     875.943278    269.614934   -570.500166    432.334420    427.547277
std      247.719619    362.563895    300.011889    485.647768    199.608760
min     -442.000000  -1430.000000  -1933.000000  -1103.000000   -765.000000
25%      781.000000     24.000000   -819.000000    -54.000000    290.000000
50%      921.000000    247.000000   -623.000000    530.000000    433.000000
75%     1028.000000    561.000000   -318.000000    846.000000    562.000000
max     2463.000000   1570.000000    939.000000   1846.000000   1367.000000

            fea.31       fea.32       fea.33       fea.37       fea.38  \
count  19000.000000  19000.000000  19000.000000  19000.000000  19000.000000
mean    -124.834707    369.141625    646.348062   -862.982169   -209.995885
```

As we can see from the above dataset that the missing values have been filled with mean values after applying simple Imputation

It is clear that all the missing values from the dataset are filled with mean values for each column. However there are merits and dmerits of using Imputaion with Mean values they are shown as:

Merits:

1. It is easy and faster to implement.
2. It works well with small dataset.

Demerits:

1. It doesn't factor the correlation between the features.
2. It doesn't encounter the Uncertainity in the imputation
3. It is not very accurate.
4. It is susceptible to Outliers

2. Median Imputation

In [22]:
```
# Copy Data for Median Imputation
dataA_updatedForMedian = dataA_updated_redDim.copy(deep=True)
```

In [23]:
```
si_mean = SimpleImputer(strategy='median')
si_mean.fit(dataA_updatedForMedian)
dataA_SiMedianImputation = si_mean.transform(dataA_updated_redDim)
```

In [24]:
```
print("\n Data after Median Imputation: \n", dataA_SiMedianImputation)
```

```
 Data after Median Imputation:
 [[-153.  414.  939. ...  -29.   36.   24.]
 [-150.  420.  939. ...  -31.   47.    3.]
 [-160.  432.  941. ...  -38.   20.    0.]
 ...
 [-100.  811.  574. ...  -29.    4.   19.]
 [-100.  811.  574. ...  -29.    4.   19.]
 [-100.  811.  574. ...  -29.    4.   19.]]
```

Merits:

The Median imputation is not susceptible to Outliers so it is a better way for finding missing values

Demerits:

It doesn't factor the correlation between the features.

3. Imputation using KNN

In [25]:
```
knnimpute = KNNImputer(n_neighbors=2)
dataA_knnImputation = knnimpute.fit_transform(dataA_updated_redDim)
```

In [26]:
```
print("\n Data after KNN imputation for missing values: \n ", dataA_knnImputati
on)
```

```
 Data after KNN imputation for missing values: 
  [[-153.          414.          939.         ...  -29.
    36.          24.        ]
 [-150.          420.          939.         ...  -31.
    47.           3.        ]
 [-160.          432.          941.         ...  -38.
    20.           0.        ]
 ...
 [-132.81238421  698.26448462  597.54140235 ...  -18.09952269
     4.67125693   20.72683382]
 [-132.81238421  698.26448462  597.54140235 ...  -18.09952269
     4.67125693   20.72683382]
 [-132.81238421  698.26448462  597.54140235 ...  -18.09952269
     4.67125693   20.72683382]]
```

The Imputed Values obatined from KNN based method is very similar to the Imputed values obtained by using Mean Imputation.

Merits

1. much more accurate than Mean/Median

Demerits

1. It is sensitive to the outliers
2. It is computaionally very Expensive

4. Imputation using Linear Interpolation

In [27]:
```
# Copy the data for Linear Interpolation
data_linInterpolate = dataA_updated_redDim.copy(deep=True)
```

In [28]:
```
dataA_interpolateImpute = data_linInterpolate.interpolate(method='linear', axis
=0)
print("\n Data after Linear Interpolation: \n",dataA_interpolateImpute)
```

```
 Data after Linear Interpolation:
        fea.1  fea.2  fea.3  fea.4   fea.5  fea.6  fea.7  fea.8  fea.9  fea.10
\
0      -153.0  414.0  939.0 -161.0  1007.0   99.0 -210.0  948.0  333.0   -19.0
1      -150.0  420.0  939.0 -177.0  1008.0  103.0 -207.0  939.0  316.0     9.0
2      -160.0  432.0  941.0 -162.0   982.0   98.0 -198.0  936.0  315.0   -10.0
3      -171.0  432.0  911.0 -174.0   999.0  115.0 -187.0  918.0  338.0    34.0
4      -171.0  432.0  929.0 -189.0  1004.0  104.0 -198.0  939.0  350.0    60.0
...       ...    ...    ...    ...     ...    ...    ...    ...    ...     ...
18995     0.0    0.0    0.0    0.0     0.0    0.0    0.0    0.0    0.0     0.0
18996     0.0    0.0    0.0    0.0     0.0    0.0    0.0    0.0    0.0     0.0
18997     0.0    0.0    0.0    0.0     0.0    0.0    0.0    0.0    0.0     0.0
18998     0.0    0.0    0.0    0.0     0.0    0.0    0.0    0.0    0.0     0.0
18999     0.0    0.0    0.0    0.0     0.0    0.0    0.0    0.0    0.0     0.0

       fea.11  fea.12  fea.13  fea.14  fea.15  fea.16  fea.17  fea.18  fea.19
\
0      -587.0   810.0   902.0  -140.0  -468.0   -28.0  1016.0  -191.0   358.0
1      -605.0   835.0   897.0  -136.0  -473.0   -38.0   989.0  -160.0   346.0
2      -580.0   802.0   902.0  -139.0  -454.0   -19.0   992.0  -170.0   345.0
3      -579.0   849.0   910.0  -151.0  -460.0   -18.0  1015.0  -185.0   345.0
4      -584.0   843.0   917.0  -151.0  -463.0   -12.0   996.0  -209.0   345.0
...       ...     ...     ...     ...     ...     ...     ...     ...     ...
18995     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
18996     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
18997     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
18998     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
18999     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0

       fea.20  fea.21  fea.22  fea.23  fea.24  fea.25  fea.26  fea.27  fea.28
\
0       895.0  -189.0  -511.0  -782.0  -212.0  -241.0   472.0   938.0  -882.0
1       889.0  -215.0  -609.0  -773.0  -202.0  -255.0   509.0   934.0  -878.0
2       898.0  -200.0  -606.0  -753.0  -116.0  -272.0   492.0   919.0  -864.0
3       888.0  -220.0  -638.0  -718.0  -354.0  -275.0   517.0   919.0  -847.0
4       886.0  -204.0  -723.0  -679.0  -220.0  -271.0   525.0   923.0  -862.0
...       ...     ...     ...     ...     ...     ...     ...     ...     ...
18995     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
18996     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
18997     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
18998     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
18999     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0

       fea.29  fea.30  fea.31  fea.32  fea.33  fea.37  fea.38  fea.39  fea.40
\
0       -61.0   543.0   141.0  -150.0   965.0  -989.0  -158.0    41.0    78.0
1       -50.0   515.0   130.0  -179.0   995.0  -991.0  -157.0    43.0    48.0
2       -56.0   540.0   122.0  -162.0   946.0  -988.0  -171.0    41.0    68.0
3       -66.0   546.0   145.0  -144.0   939.0  -994.0  -159.0    42.0    32.0
4       -70.0   531.0   153.0  -167.0   970.0 -1002.0  -161.0    42.0    58.0
...       ...     ...     ...     ...     ...     ...     ...     ...     ...
18995     0.0     0.0     0.0     0.0     0.0  -970.0  -249.0   -79.0   -13.0
18996     0.0     0.0     0.0     0.0     0.0  -970.0  -249.0   -79.0   -13.0
18997     0.0     0.0     0.0     0.0     0.0  -970.0  -249.0   -79.0   -13.0
18998     0.0     0.0     0.0     0.0     0.0  -970.0  -249.0   -79.0   -13.0
18999     0.0     0.0     0.0     0.0     0.0  -970.0  -249.0   -79.0   -13.0

       fea.41  fea.42  fea.43  fea.44  fea.45  fea.46  fea.47  fea.48  fea.49
\
0       -49.0   -13.0   643.0   104.0   924.0  -717.0    62.0  -691.0    63.0
1       -85.0     1.0   644.0   107.0   922.0  -718.0    61.0  -690.0    63.0
2       -68.0    -8.0   642.0   109.0   919.0  -719.0    60.0  -689.0    64.0
3       -95.0     4.0   640.0   114.0   915.0  -720.0    60.0  -688.0    65.0
```

In [29]:
```
# Check for missing values
print("\n Missing values after Linear Interpolation: ",dataA_interpolateImpute.
isnull().sum().sum())
```

```
 Missing values after Linear Interpolation:  0
```

In [30]:
```
# Statistical Analysis for the Dataset after Linear Interpolation
print(dataA_interpolateImpute.describe())
```

```
              fea.1         fea.2         fea.3         fea.4         fea.5  \
count  19000.000000  19000.000000  19000.000000  19000.000000  19000.000000
mean    -124.740553    679.817605    563.032474   -294.196737    871.251316
std      279.158229    401.075991    408.463091    189.569822    263.449167
min    -2724.000000   -855.000000  -2196.000000  -1365.000000   -245.000000
25%     -173.000000    334.000000    250.000000   -402.000000    836.000000
50%      -92.000000    794.000000    536.000000   -255.000000    964.000000
75%        0.000000    983.000000    950.000000   -156.000000   1005.000000
max     1887.000000   2531.000000   2941.000000    609.000000   1833.000000

              fea.6         fea.7         fea.8         fea.9        fea.10  \
count  19000.000000  19000.000000  19000.000000  19000.000000  19000.000000
mean     -31.381421     56.161263    859.519211     81.482737    341.387000
std      248.682889    313.189606    264.930665    323.747494    342.941702
min     -920.000000  -1580.000000   -149.000000  -1624.000000  -1792.000000
25%     -139.000000   -126.000000    831.000000   -143.000000    120.000000
50%      -30.000000     49.000000    940.000000     22.000000    362.000000
75%       40.000000    239.000000    994.000000    307.000000    572.000000
max     1215.000000   1490.000000   1682.000000   1096.000000   2202.000000

             fea.11        fea.12        fea.13        fea.14        fea.15  \
count  19000.000000  19000.000000  19000.000000  19000.000000  19000.000000
mean     311.822211    560.280132    555.819816    397.217289    -38.605474
std      539.116818    299.208213    325.365484    545.722719    488.181989
min    -1545.000000  -1079.000000  -1710.000000  -1120.000000  -1634.000000
25%     -132.000000    340.000000    311.000000   -123.000000   -465.000000
50%      339.000000    597.000000    597.000000    493.000000      0.000000
75%      770.000000    805.000000    811.000000    865.000000    255.000000
max     2047.000000   2152.000000   2408.000000   2212.000000   2620.000000

             fea.16        fea.17        fea.18        fea.19        fea.20  \
count  19000.000000  19000.000000  19000.000000  19000.000000  19000.000000
mean      68.791842    747.583211   -442.217632    265.354658    852.433553
std      253.849945    325.555065    271.947856    296.737216    278.939594
min    -1089.000000   -289.000000  -1379.000000  -2269.000000   -786.000000
25%        0.000000    651.000000   -637.000000    229.000000    869.000000
50%       86.000000    895.000000   -385.000000    300.000000    917.000000
75%      174.000000    964.000000   -268.750000    354.000000    943.000000
max      878.000000   1653.000000   1292.000000   2335.000000   2466.000000

             fea.21        fea.22        fea.23        fea.24        fea.25  \
count  19000.000000  19000.000000  19000.000000  19000.000000  19000.000000
mean     -85.254842   -592.546237    217.029447   -125.687132   -342.562079
std      216.916468    375.072241    579.982702    449.465363    233.402454
min    -1944.000000  -3172.000000  -2324.000000  -2734.000000  -1873.000000
25%     -188.000000   -863.000000   -180.000000   -453.000000   -467.000000
50%      -61.000000   -628.000000    219.000000   -111.000000   -330.000000
75%       29.000000   -339.000000    705.000000    167.000000   -184.000000
max     1968.000000   2229.000000   2677.000000   2401.000000    711.000000

             fea.26        fea.27        fea.28        fea.29        fea.30  \
count  19000.000000  19000.000000  19000.000000  19000.000000  19000.000000
mean     838.253211    261.187526   -546.851079    408.188342    408.774105
std      303.878500    368.717572    321.540375    500.265027    217.718417
min     -442.000000  -1430.000000  -1933.000000  -1103.000000   -765.000000
25%      726.000000      0.000000   -820.000000    -56.000000    254.000000
50%      922.000000    198.000000   -627.000000    530.000000    433.000000
75%     1028.000000    570.000000   -273.000000    846.000000    562.000000
max     2463.000000   1570.000000    939.000000   1846.000000   1367.000000

             fea.31        fea.32        fea.33        fea.37        fea.38  \
count  19000.000000  19000.000000  19000.000000  19000.000000  19000.000000
mean    -118.214237    346.969895    619.289711   -867.336105   -211.582737
```

From the Statical Analysis obtained after Imputing with Linear Iterpolation we observe that the mean and 2 standard away mean(50%) for Linear Interpolation method values are different from that obtained after Mean/Median Imputation on the dataset.

5. Multivariate Imputation using Iterative Imputation

In [31]:
```
# Initialize the IterativeImputation
ii = IterativeImputer(initial_strategy = 'median', random_state=42, imputation_
order = 'descending', add_indicator = True)
ii.fit(dataA_updated_redDim)
dataA_IterImpute = ii.transform(dataA_updated_redDim)
```

```
C:\Users\tonkh\anaconda3\lib\site-packages\sklearn\impute\_iterative.py:638: Co
nvergenceWarning: [IterativeImputer] Early stopping criterion not reached.
  " reached.", ConvergenceWarning)
```

In [32]:
```
print("\n Data after Iterative Imputation: \n", dataA_IterImpute)
```

```
 Data after Iterative Imputation:
 [[-153.          414.          939.         ...    0.
     0.           0.        ]
 [-150.          420.          939.         ...    0.
     0.           0.        ]
 [-160.          432.          941.         ...    0.
     0.           0.        ]
 ...
 [-100.08514228  729.70155501  605.13668153 ...    1.
     1.           1.        ]
 [-100.08514228  729.70155501  605.13668153 ...    1.
     1.           1.        ]
 [-100.08514228  729.70155501  605.13668153 ...    1.
     1.           1.        ]]
```

In [33]:
```
# Number of Features with missing values
print("\n Features with missing values before Imputation: ",ii.n_features_with_
missing_)
```

```
 Features with missing values before Imputation:  78
```

It is evident from the output obtained after Multivariate Imputation using Iterative Imputation that the values are distinct from what we have obtained from Mean, Median, KNN and Linear Interpolation.

6. MICE (http://scholar.google.ca/scholar_url?url=https://www.jstatsoft.org/article/view/v045i04/v45i04.pdf&hl=en&sa=X&scisig=AAGBfm2i0J4zMPMI6FJgalzvk2DxTDu65g&nossl=1&oi=scholarr)

It works by filling missing data multiple times. As given in the paper (http://scholar.google.ca/scholar_url?url=https://www.jstatsoft.org/article/view/v045i04/v45i04.pdf&hl=en&sa=X&scisig=AAGBfm2i0J4zMPMI6FJgalzvk2DxTDu65g&nossl=1&oi=scholarr) Multiple Imputation is much better than that of Single Imputation as it improves the measure of Uncertainity of the missing values.

In the library of impyute the algorithm of [MICE] uses LinearRegression for Convergence of missing values for the Dataset.

In [34]:
```
dataA_MICEImputation = mice(dataA_updated_redDim.values)
```

In [35]:
```
print("\n Data after MICE Imputation: \n", dataA_MICEImputation)
```

Output from MICE Algorithm

Data after MICE Imputation: [[-153. 414. 939. ... -29.

```
36.          24.       ]
```

[-150. 420. 939. ... -31.

```
47.           3.       ]
```

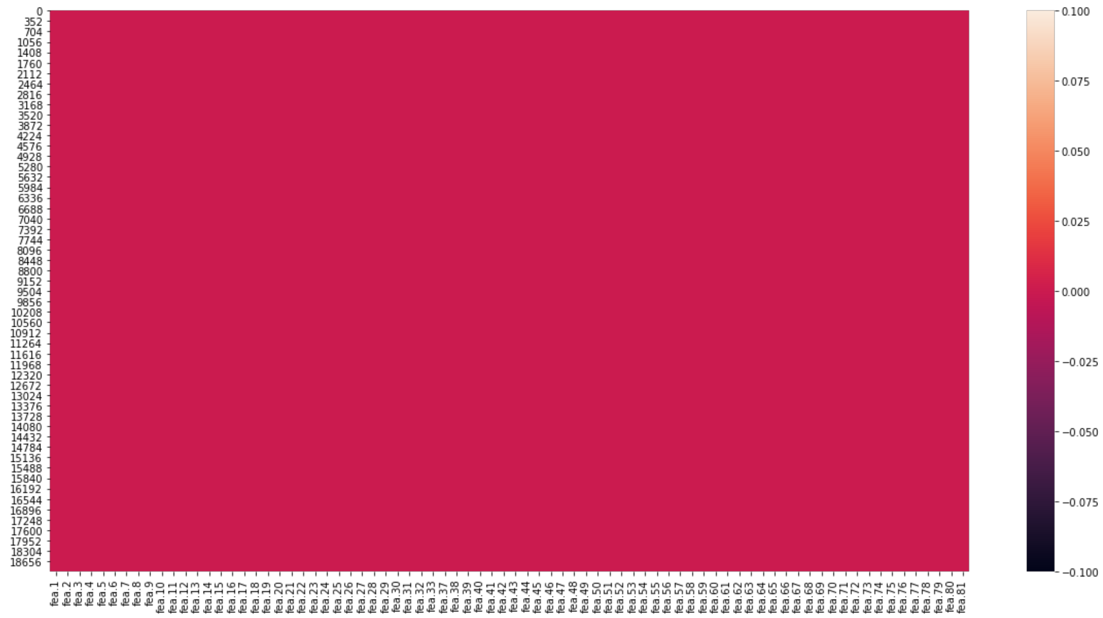
[-160. 432. 941. ... -38.

```
20.           0.       ]
```

... [-130.36492054 705.07225938 589.2263699 ... -18.09952269 4.67125693 20.72683382] [-130.36492054 705.07225938 589.2263699 ... -18.09952269 4.67125693 20.72683382] [-130.36492054 705.07225938 589.2263699 ... -18.09952269 4.67125693 20.72683382]]

The Output obtained from the above algorithm differs from that of all the above methods and since the basis of this algorithm uses LinearRegression it should have more accurate data for a given value. We have mainly used this for comparing our results with other methods. However it is computationally very inefficient. Therefore we have ran it during our leisure time and provided the output above.

In [36]:
```
# To check missing values using Heat Map
plt.subplots(figsize=(20,10))
sns.heatmap(data_forMeanImputing.isnull())
plt.show()
```



As from the heat map it is evident that there are no missing values

Outlier Detection in the Dataset

In [37]:
```
# Convert array to Dataframe
data_knn_dataframe = pd.DataFrame(data=dataA_knnImputation)

# Total Outlier count in the Dataset
dataA_std = data_knn_dataframe.std()
dataA_mean = data_knn_dataframe.mean()

dataA_data_outliers=data_knn_dataframe.apply(lambda y: np.abs(y-dataA_mean)>(3*
dataA_std),axis=1)
print("Total number of ouliers in each feature are: \n",dataA_data_outliers.sum
())
```

```
Total number of ouliers in each feature are:
 0     742
1     160
2     185
3     156
4     735
     ...
73     20
74    228
75    363
76    199
77    441
Length: 78, dtype: int64
```

In [38]:
```
# Total Number of outlier in the dataset
print("Total number of ouliers in the dataset: ",dataA_data_outliers.sum().sum
())
```

```
Total number of ouliers in the dataset:  16154
```

So as seen from the above analysis that there are exactly 16154 outlier in the dataset. In addition, a count for number of outlier in each feature is shown.

1. Using Z-Score for Outlier Detection

In [39]:
```
# Detect Outlier using ZScore
outlier_value = np.abs(stats.zscore(data_knn_dataframe))
print("Outlier: \n",outlier_value)
```

```
Outlier:
 [[8.18201697e-02 7.87385473e-01 8.88211767e-01 ... 1.43050168e-02
  6.65315630e-02 7.34234028e-03]
 [7.10717267e-02 7.71170266e-01 8.88211767e-01 ... 1.69296756e-02
  8.98918105e-02 3.97646923e-02]
 [1.06899870e-01 7.38739853e-01 8.93300603e-01 ... 2.61159814e-02
  3.25530211e-02 4.64942683e-02]
 ...
 [9.49169104e-03 1.91509044e-02 1.93982622e-02 ... 4.66233060e-18
  0.00000000e+00 7.96941899e-18]
 [9.49169104e-03 1.91509044e-02 1.93982622e-02 ... 4.66233060e-18
  0.00000000e+00 7.96941899e-18]
 [9.49169104e-03 1.91509044e-02 1.93982622e-02 ... 4.66233060e-18
  0.00000000e+00 7.96941899e-18]]
```

In [40]:
```
standard_thershold = 3
print("\n Position of Outliers in the Dataset: \n",np.where(outlier_value>stand
ard_thershold))
```

```
 Position of Outliers in the Dataset:
 (array([   60,    60,    61, ..., 18202, 18206, 18206], dtype=int64), array([5
1, 62, 50, ..., 19,  0, 25], dtype=int64))
```

In [41]:
```
print("\n Outlier at row 60 and column 1: ",outlier_value[60][1])
```

```
 Outlier at row 60 and column 1:  0.6711764908392637
```

The above says that there is an outlier at row: 60 in column: 1.

2. IQR Method for Outlier detection

In [42]:
```
# find the Quartiles for the dataset
Q1 = data_knn_dataframe.quantile(0.25)
Q3 = data_knn_dataframe.quantile(0.75)
IQR = Q3 - Q1
print("The InterQuartile Range for Each Features is: \n ", IQR)
```

```
The InterQuartile Range for Each Features is: 
  0     158.000
1     616.000
2     642.125
3     233.000
4     138.000
       ...
73    486.000
74    233.000
75    397.000
76    214.250
77    260.250
Length: 78, dtype: float64
```

In [43]:
```
# Detect the Oultiers using IQR technique
print("\n Outliers in the complete dataset: \n", (data_knn_dataframe < (Q1 - 1.
5 * IQR))|(data_knn_dataframe > (Q3 + 1.5 * IQR)))
```

```
 Outliers in the complete dataset:
           0      1      2      3      4      5      6      7      8      9  \
0      False  False  False  False  False  False  False  False  False  False   
1      False  False  False  False  False  False  False  False  False  False   
2      False  False  False  False  False  False  False  False  False  False   
3      False  False  False  False  False  False  False  False  False  False   
4      False  False  False  False  False  False  False  False  False  False   
...      ...    ...    ...    ...    ...    ...    ...    ...    ...    ...   
18995  False  False  False  False  False  False  False  False  False  False   
18996  False  False  False  False  False  False  False  False  False  False   
18997  False  False  False  False  False  False  False  False  False  False   
18998  False  False  False  False  False  False  False  False  False  False   
18999  False  False  False  False  False  False  False  False  False  False   

          10     11     12     13     14     15     16     17     18     19  \
0      False  False  False  False  False  False  False  False  False  False   
1      False  False  False  False  False  False  False  False  False  False   
2      False  False  False  False  False  False  False  False  False  False   
3      False  False  False  False  False  False  False  False  False  False   
4      False  False  False  False  False  False  False  False  False  False   
...      ...    ...    ...    ...    ...    ...    ...    ...    ...    ...   
18995  False  False  False  False  False  False  False  False  False  False   
18996  False  False  False  False  False  False  False  False  False  False   
18997  False  False  False  False  False  False  False  False  False  False   
18998  False  False  False  False  False  False  False  False  False  False   
18999  False  False  False  False  False  False  False  False  False  False   

          20     21     22     23     24     25     26     27     28     29  \
0      False  False  False  False  False  False  False  False  False  False   
1      False  False  False  False  False  False  False  False  False  False   
2      False  False  False  False  False  False  False  False  False  False   
3      False  False  False  False  False  False  False  False  False  False   
4      False  False  False  False  False  False  False  False  False  False   
...      ...    ...    ...    ...    ...    ...    ...    ...    ...    ...   
18995  False  False  False  False  False  False  False  False  False  False   
18996  False  False  False  False  False  False  False  False  False  False   
18997  False  False  False  False  False  False  False  False  False  False   
18998  False  False  False  False  False  False  False  False  False  False   
18999  False  False  False  False  False  False  False  False  False  False   

          30     31     32     33     34     35     36     37     38     39  \
0      False  False  False  False  False  False  False  False  False  False   
1      False  False  False  False  False  False  False  False  False  False   
2      False  False  False  False  False  False  False  False  False  False   
3      False  False  False  False  False  False  False  False  False  False   
4      False  False  False  False  False  False  False  False  False  False   
...      ...    ...    ...    ...    ...    ...    ...    ...    ...    ...   
18995  False  False  False  False  False  False  False  False  False  False   
18996  False  False  False  False  False  False  False  False  False  False   
18997  False  False  False  False  False  False  False  False  False  False   
18998  False  False  False  False  False  False  False  False  False  False   
18999  False  False  False  False  False  False  False  False  False  False   

          40     41     42     43     44     45     46     47     48     49  \
0      False  False  False  False  False  False   True  False  False  False   
1      False  False  False  False  False  False   True  False  False  False   
2      False  False  False  False  False  False   True  False  False  False   
3      False  False  False  False  False  False   True  False  False  False   
4      False  False  False  False  False  False   True  False  False  False   
...      ...    ...    ...    ...    ...    ...    ...    ...    ...    ...   
18995  False  False  False  False  False  False  False  False  False  False   
18996  False  False  False  False  False  False  False  False  False  False   
18997  False  False  False  False  False  False  False  False  False  False   
```

The value where there is a True label indicated the presence of an Outlier and False when there are no Outliers.

3. Outlier detection using IsolationForest

In [44]:
```
# IsolationForest Initialization
isoforest = IsolationForest(max_samples="auto", n_jobs=-1)
dataA_outliers_isoforest = isoforest.fit_predict(data_knn_dataframe)
```

In [45]:
```
print("\n Outliers from IsolationForest: \n", dataA_outliers_isoforest)
```

```
 Outliers from IsolationForest: 
 [1 1 1 ... 1 1 1]
```

In [46]:
```
print("Shape of IsolationForest Dataset: ",dataA_outliers_isoforest.shape)
```

```
Shape of IsolationForest Dataset:  (19000,)
```

In [47]:
```
# Display all the labels from IsolationForest
count = 0
for value in range(dataA_outliers_isoforest.shape[0]):
    if dataA_outliers_isoforest[value] == -1:
        sys.stderr.write("\n Outliers for each label: {0}".format(dataA_outlier
s_isoforest[value]))
        sys.stderr.flush()
        count += 1

print("\n Total number Outliers detected by IsolationForest is: ", count)
```

```
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
Outliers for each label: -1
```

```
Total number Outliers detected by IsolationForest is:  719
```

The label given by Isolation Forest method gives us whether a Outlier is present or not. If the label is -1 then there is a outlier whereas when the label is +1 then there are no outlier or it is an inlier. As seen from the above analysis that IsolationForest gives us the total number of Outliers in the dataset which is 1231 and it is far less than Z-score and IQR method.

Removing Outliers from the Imputed Dataset

In [48]:
```
# Remove outliers in the KNN Obtained Dataset
dataA_OutlierRemovedknn= data_knn_dataframe[(outlier_value < 3).all(axis=1)]
```

In the above code the outliers have been removed from KNN Imputed dataset.

Name : Harleen Kaur Taunque

Student Id : 20811951

Name : Somesh Gupta

Student Id: 20817245

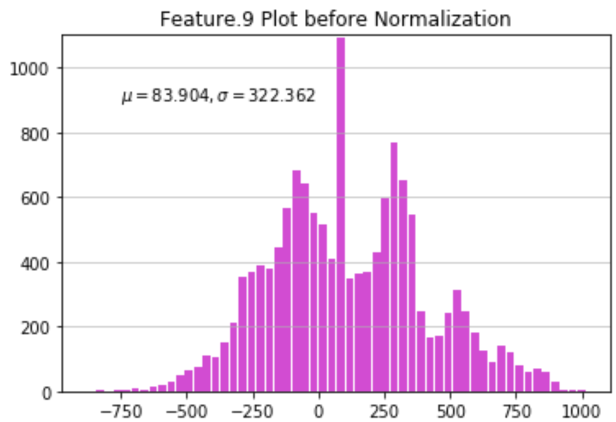
Part 1.3

Feature.9 and Feature.24 plot

Before Normalization

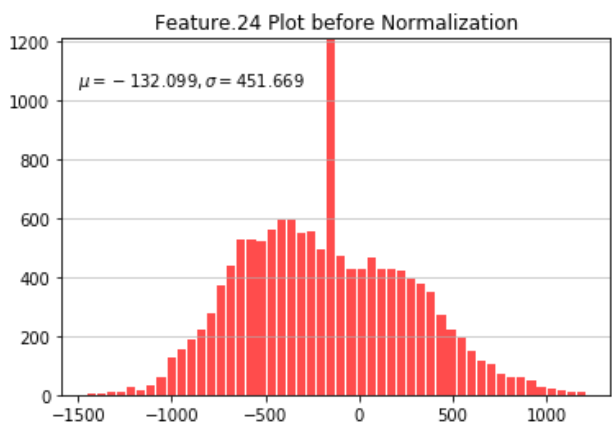
In [49]:
```
# Plot of Feature.9 with KNN imputed dataset
n, bins, patches = plt.hist(x=dataA_OutlierRemovedknn.iloc[:, 8], bins='auto',
color='m', alpha=0.7, rwidth=0.85)
plt.grid(axis='y', alpha=0.75)
plt.title('Feature.9 Plot before Normalization')
plt.text(-750,900, r'$\mu=83.904 , \sigma= 322.362$')
maxfreq = n.max()
plt.ylim(ymax=np.ceil(maxfreq/10) * 10 if maxfreq % 10 else maxfreq + 10)
```

Out[49]: (0.0, 1100.0)



In [50]:
```
# Plot of Feature.24 with KNN imputed dataset
n, bins, patches = plt.hist(x=dataA_OutlierRemovedknn.iloc[:, 23], bins='auto',
color='r', alpha=0.7, rwidth=0.85)
plt.grid(axis='y', alpha=0.75)
plt.text(-1500, 1050, r'$\mu=-132.099, \sigma= 451.669$')
plt.title('Feature.24 Plot before Normalization')
maxfreq = n.max()
plt.ylim(ymax=np.ceil(maxfreq/10) * 10 if maxfreq % 10 else maxfreq + 10)
```

Out[50]: (0.0, 1210.0)



Data Normalization

1. Max-Min Normalization

In [51]:
```
MMscaler = MinMaxScaler()
MMscaler.fit(dataA_OutlierRemovedknn)
data_normalizedMM = MMscaler.transform(dataA_OutlierRemovedknn)
```

In [52]:
```
print("\n Max-Min Normalized dataset: \n", data_normalizedMM)
```

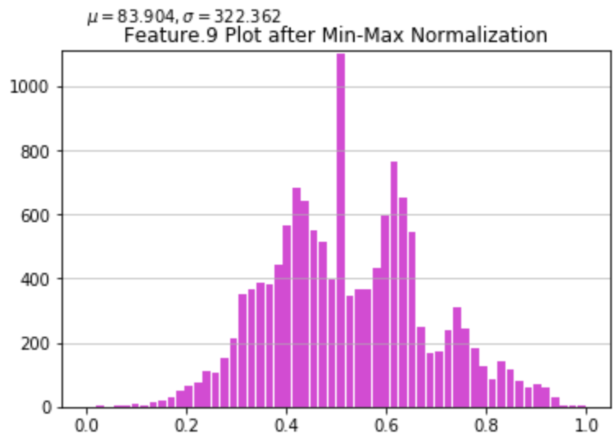
```
 Max-Min Normalized dataset: 
 [[0.48501199 0.36720867 0.6498719  ... 0.49492945 0.50714286 0.50056455]
 [0.48681055 0.3699187  0.6498719  ... 0.49448854 0.5112782  0.4926609 ]
 [0.48081535 0.37533875 0.65072588 ... 0.49294533 0.50112782 0.4915318 ]
 ...
 [0.49711488 0.49560275 0.50407404 ... 0.49733256 0.49536513 0.49933264]
 [0.49711488 0.49560275 0.50407404 ... 0.49733256 0.49536513 0.49933264]
 [0.49711488 0.49560275 0.50407404 ... 0.49733256 0.49536513 0.49933264]]
```

In [53]:
```
print("Data of Feature.9: \n",data_normalizedMM[:,8])
```

```
Data of Feature.9: 
 [0.63962065 0.63066386 0.63013699 ... 0.50719203 0.50719203 0.50719203]
```

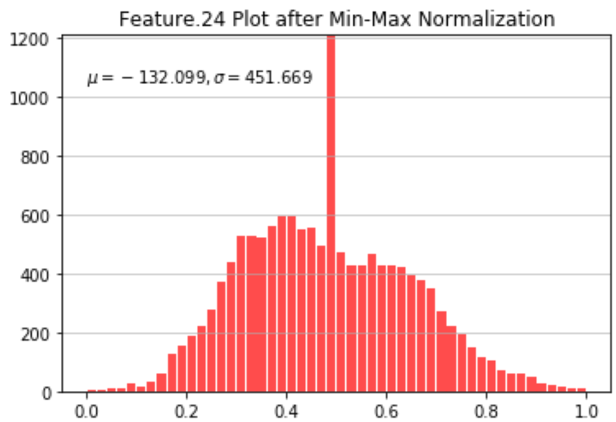
In [54]:
```
# Plot of Feature.9 with KNN imputed dataset
n, bins, patches = plt.hist(x=data_normalizedMM[:, 8], bins='auto', color='m',
alpha=0.7, rwidth=0.85)
plt.grid(axis='y', alpha=0.75)
plt.title('Feature.9 Plot after Min-Max Normalization')
plt.text(0,1200, r'$\mu=83.904 , \sigma= 322.362$')
maxfreq = n.max()
plt.ylim(ymax=np.ceil(maxfreq/10) * 10 if maxfreq % 10 else maxfreq + 10)
```

Out[54]: (0.0, 1110.0)



In [55]:
```
# Plot of Feature.24 with KNN imputed dataset
n, bins, patches = plt.hist(x=data_normalizedMM[:, 23], bins='auto', color='r',
alpha=0.7, rwidth=0.85)
plt.grid(axis='y', alpha=0.75)
plt.title('Feature.24 Plot after Min-Max Normalization')
plt.text(0, 1050, r'$\mu=-132.099, \sigma= 451.669$')
maxfreq = n.max()
plt.ylim(ymax=np.ceil(maxfreq/10) * 10 if maxfreq % 10 else maxfreq + 10)
```

Out[55]: (0.0, 1210.0)



2. Z-Score Normalization (Standardization)

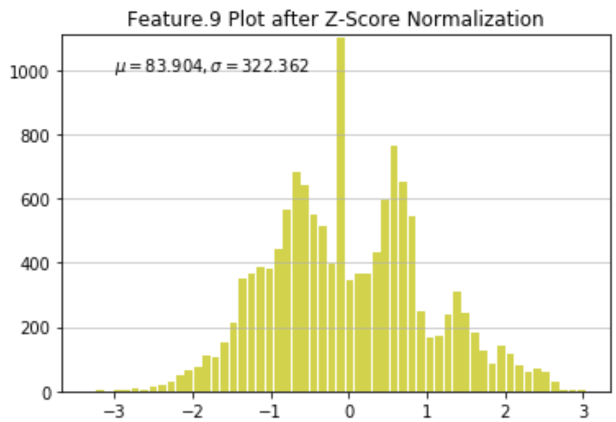
In [56]:
```
# Z-Score Normalization
ss = StandardScaler()
ss.fit(dataA_OutlierRemovedknn)
dataA_standardised = ss.transform(dataA_OutlierRemovedknn)
```

In [57]:
```
print("Standardized Dataset: \n", dataA_standardised)
```

```
Standardized Dataset: 
 [[-3.89907455e-01 -8.24536715e-01  8.62808975e-01 ...  6.90475610e-04
   1.53562436e-01 -1.21843822e-02]
 [-3.70034786e-01 -8.06083774e-01  8.62808975e-01 ... -3.08203888e-03
   1.98633738e-01 -8.05587443e-02]
 [-4.36277014e-01 -7.69177891e-01  8.68672966e-01 ... -1.62858396e-02
   8.80041785e-02 -9.03265103e-02]
 ...
 [-2.56180190e-01  4.97159352e-02 -1.38346117e-01 ...  2.12515799e-02
   2.51963232e-02 -2.28415560e-02]
 [-2.56180190e-01  4.97159352e-02 -1.38346117e-01 ...  2.12515799e-02
   2.51963232e-02 -2.28415560e-02]
 [-2.56180190e-01  4.97159352e-02 -1.38346117e-01 ...  2.12515799e-02
   2.51963232e-02 -2.28415560e-02]]
```

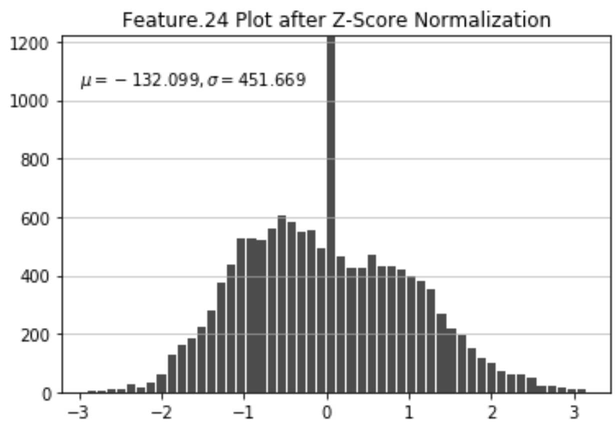
In [58]:
```
# Plot of Feature.9 with KNN imputed dataset
n, bins, patches = plt.hist(x=dataA_standardised[:, 8], bins='auto', color='y',
alpha=0.7, rwidth=0.85)
plt.grid(axis='y', alpha=0.75)
plt.title('Feature.9 Plot after Z-Score Normalization')
plt.text(-3,1000, r'$\mu=83.904 , \sigma= 322.362$')
maxfreq = n.max()
plt.ylim(ymax=np.ceil(maxfreq/10) * 10 if maxfreq % 10 else maxfreq + 10)
```

Out[58]: (0.0, 1110.0)



In [59]:
```
# Plot of Feature.24 with KNN imputed dataset
n, bins, patches = plt.hist(x=dataA_standardised[:, 23], bins='auto', color='k
', alpha=0.7, rwidth=0.85)
plt.grid(axis='y', alpha=0.75)
plt.title('Feature.24 Plot after Z-Score Normalization')
plt.text(-3, 1050, r'$\mu=-132.099, \sigma= 451.669$')
maxfreq = n.max()
plt.ylim(ymax=np.ceil(maxfreq/10) * 10 if maxfreq % 10 else maxfreq + 10)
```

Out[59]: (0.0, 1220.0)



Analysis:

From the plot generated before max min and z-score normalization the x-axis for the plot by feature.9 ranges from 0 to 1000 and that of the feature.24 ranges from -1500 to 1500. After max-min normalization the range of the x-axis changes and it becomes normalized between 0 and 1 for both the plot for feature.9 and feature.24. Where the plot generated after Z-score normalization the x-range ranges from -4 to 4 for both the features. However there is no change in the y label of the plots before and after normalization for both max-min and z-score normalization. From the graphs it can be seen that Feature 24 shows lesser variation than Feature 9 . Also, data is more concentrated at the center in the plots.