Dealing the missing value & Outlier Treatment

Importing the required libraries

In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
df=pd.read_csv('C:\\Users\\dell\\Music\\claimants sample.csv')
```

In [3]:

```
df
```

Out[3]:

	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
0	5	0	1.0	0	50.0	34.940	0
1	3	1	0.0	0	18.0	0.891	1
2	66	0	1.0	0	5.0	0.330	1
3	70	1	1.0	1	31.0	0.037	0
4	96	0	1.0	0	30.0	NaN	1
5	97	1	1.0	0	35.0	0.309	0
6	10	0	NaN	0	9.0	3.538	0
7	36	1	NaN	0	34.0	4.881	0
8	51	1	1.0	0	60.0	0.874	1
9	55	1	1.0	0	NaN	0.350	1

step-1---find whether in the given data missing value is there or not ? If there seperate continues & discriptive stats

In [7]:

```
df.isnull().sum()
```

Out[7]:

```
CASENUM     0
CLMSEX      0
CLMINSUR    2
SEATBELT    0
CLMAGE      1
LOSS        1
ATTORNEY    0
dtype: int64
```

step-2---seperate continues & discriptive stats

In [11]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   CASENUM   10 non-null     int64  
 1   CLMSEX    10 non-null     int64  
 2   CLMINSUR  8 non-null      float64
 3   SEATBELT  10 non-null     int64  
 4   CLMAGE    9 non-null      float64
 5   LOSS      9 non-null      float64
 6   ATTORNEY  10 non-null     int64  
dtypes: float64(3), int64(4)
memory usage: 688.0 bytes
```

In [12]:

```
df
```

Out[12]:

	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
0	5	0	1.0	0	50.0	34.940	0
1	3	1	0.0	0	18.0	0.891	1
2	66	0	1.0	0	5.0	0.330	1
3	70	1	1.0	1	31.0	0.037	0
4	96	0	1.0	0	30.0	NaN	1
5	97	1	1.0	0	35.0	0.309	0
6	10	0	NaN	0	9.0	3.538	0
7	36	1	NaN	0	34.0	4.881	0
8	51	1	1.0	0	60.0	0.874	1
9	55	1	1.0	0	NaN	0.350	1

step-3--Remove --dropna

1.we can remove 5% if data is big

In [16]:

```
df1=df.dropna()
```

In [17]:

```
df1
```

Out[17]:

	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
0	5	0	1.0	0	50.0	34.940	0
1	3	1	0.0	0	18.0	0.891	1
2	66	0	1.0	0	5.0	0.330	1
3	70	1	1.0	1	31.0	0.037	0
5	97	1	1.0	0	35.0	0.309	0
8	51	1	1.0	0	60.0	0.874	1

step -4 Replace the nan values

1.Mean

2.Median

3.Mode

4.fill with some values

Contonous Variable---> AGE , LOSS --Replace with either mean or meadian

Discrete Variable----> INSUR ---> Mode is used for discrete variable

In [1]:

```
df
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [1], in <cell line: 1>()
----> 1 df

NameError: name 'df' is not defined
```

In [ ]:

```
df['CLMAGE'].fillna(df['CLMAGE'].mean(),inplace=True)
df
```

In [31]:

```
df['LOSS'].fillna(df['LOSS'].median(),inplace=True)
df
```

Out[31]:

	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
0	5	0	1.0	0	50.000000	34.940	0
1	3	1	0.0	0	18.000000	0.891	1
2	66	0	1.0	0	5.000000	0.330	1
3	70	1	1.0	1	31.000000	0.037	0
4	96	0	1.0	0	30.000000	0.874	1
5	97	1	1.0	0	35.000000	0.309	0
6	10	0	NaN	0	9.000000	3.538	0
7	36	1	NaN	0	34.000000	4.881	0
8	51	1	1.0	0	60.000000	0.874	1
9	55	1	1.0	0	30.222222	0.350	1

In [33]:

```
df['CLMINSUR'].fillna(df['CLMINSUR'].mode()[0],inplace=True)
df
```

Out[33]:

	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
0	5	0	1.0	0	50.000000	34.940	0
1	3	1	0.0	0	18.000000	0.891	1
2	66	0	1.0	0	5.000000	0.330	1
3	70	1	1.0	1	31.000000	0.037	0
4	96	0	1.0	0	30.000000	0.874	1
5	97	1	1.0	0	35.000000	0.309	0
6	10	0	1.0	0	9.000000	3.538	0
7	36	1	1.0	0	34.000000	4.881	0
8	51	1	1.0	0	60.000000	0.874	1
9	55	1	1.0	0	30.222222	0.350	1

step 5---Simple Imputer using Sklearn

REplace with sklearn

In [35]:

```
from sklearn.impute import SimpleImputer
```

In [36]:

```
df
```

Out[36]:

	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
0	5	0	1.0	0	50.000000	34.940	0
1	3	1	0.0	0	18.000000	0.891	1
2	66	0	1.0	0	5.000000	0.330	1
3	70	1	1.0	1	31.000000	0.037	0
4	96	0	1.0	0	30.000000	0.874	1
5	97	1	1.0	0	35.000000	0.309	0
6	10	0	1.0	0	9.000000	3.538	0
7	36	1	1.0	0	34.000000	4.881	0
8	51	1	1.0	0	60.000000	0.874	1
9	55	1	1.0	0	30.222222	0.350	1

In [37]:

```
df2=pd.read_csv('C:\\Users\\dell\\Music\\claimants sample.csv')
```

In [39]:

```
df2
```

Out[39]:

	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
0	5	0	1.0	0	50.0	34.940	0
1	3	1	0.0	0	18.0	0.891	1
2	66	0	1.0	0	5.0	0.330	1
3	70	1	1.0	1	31.0	0.037	0
4	96	0	1.0	0	30.0	NaN	1
5	97	1	1.0	0	35.0	0.309	0
6	10	0	NaN	0	9.0	3.538	0
7	36	1	NaN	0	34.0	4.881	0
8	51	1	1.0	0	60.0	0.874	1
9	55	1	1.0	0	NaN	0.350	1

In [41]:

```
mean_imp=SimpleImputer(strategy='mean')
df2['CLMAGE']=mean_imp.fit_transform(df2[['CLMAGE']])
df2
```

Out[41]:

	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
0	5	0	1.0	0	50.000000	34.940	0
1	3	1	0.0	0	18.000000	0.891	1
2	66	0	1.0	0	5.000000	0.330	1
3	70	1	1.0	1	31.000000	0.037	0
4	96	0	1.0	0	30.000000	NaN	1
5	97	1	1.0	0	35.000000	0.309	0
6	10	0	NaN	0	9.000000	3.538	0
7	36	1	NaN	0	34.000000	4.881	0
8	51	1	1.0	0	60.000000	0.874	1
9	55	1	1.0	0	30.222222	0.350	1

In [42]:

```
median_imp=SimpleImputer(strategy='median')
df2['LOSS']=median_imp.fit_transform(df2[["LOSS"]])
df2
```

Out[42]:

	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
0	5	0	1.0	0	50.000000	34.940	0
1	3	1	0.0	0	18.000000	0.891	1
2	66	0	1.0	0	5.000000	0.330	1
3	70	1	1.0	1	31.000000	0.037	0
4	96	0	1.0	0	30.000000	0.874	1
5	97	1	1.0	0	35.000000	0.309	0
6	10	0	NaN	0	9.000000	3.538	0
7	36	1	NaN	0	34.000000	4.881	0
8	51	1	1.0	0	60.000000	0.874	1
9	55	1	1.0	0	30.222222	0.350	1

In [44]:

```
mode_imp=SimpleImputer(strategy='most_frequent')
df2['CLMINSUR']=mode_imp.fit_transform(df2[['CLMINSUR']])
df2
```

Out[44]:

	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
0	5	0	1.0	0	50.000000	34.940	0
1	3	1	0.0	0	18.000000	0.891	1
2	66	0	1.0	0	5.000000	0.330	1
3	70	1	1.0	1	31.000000	0.037	0
4	96	0	1.0	0	30.000000	0.874	1
5	97	1	1.0	0	35.000000	0.309	0
6	10	0	1.0	0	9.000000	3.538	0
7	36	1	1.0	0	34.000000	4.881	0
8	51	1	1.0	0	60.000000	0.874	1
9	55	1	1.0	0	30.222222	0.350	1

In [45]:

```
df.isnull().sum()
```

Out[45]:

```
CASENUM     0
CLMSEX      0
CLMINSUR    0
SEATBELT    0
CLMAGE      0
LOSS        0
ATTORNEY    0
dtype: int64
```

In [46]:

```
df1.isnull().sum()
```

Out[46]:

```
CASENUM     0
CLMSEX      0
CLMINSUR    0
SEATBELT    0
CLMAGE      0
LOSS        0
ATTORNEY    0
dtype: int64
```

In [47]:

```
df2.isnull().sum()
```

Out[47]:

```
CASENUM     0
CLMSEX      0
CLMINSUR    0
SEATBELT    0
CLMAGE      0
LOSS        0
ATTORNEY    0
dtype: int64
```

In [ ]:

```

```

Outliers

Outliers Adversly affect at the time of ML also on mean and standard daviation

reasons --Data entry,Measurement error,Instrumental error

TYPES--1. Univariate (for single variable) 2. Bivariate (for double variable)

In [1]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```python
df=pd.read_csv('C:\\Users\\dell\\Music\\claimants.csv')
```

In [3]:

```python
df
```

Out[3]:

	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
0	5	0.0	1.0	0.0	50.0	34.940	0
1	3	1.0	0.0	0.0	18.0	0.891	1
2	66	0.0	1.0	0.0	5.0	0.330	1
3	70	0.0	1.0	1.0	31.0	0.037	0
4	96	0.0	1.0	0.0	30.0	0.038	1
...	...	...	...	...	...	...	...
1335	34100	0.0	1.0	0.0	NaN	0.576	1
1336	34110	1.0	1.0	0.0	46.0	3.705	0
1337	34113	1.0	1.0	0.0	39.0	0.099	1
1338	34145	1.0	0.0	0.0	8.0	3.177	0
1339	34153	1.0	1.0	0.0	30.0	0.688	1

1340 rows × 7 columns

In [4]:

```python
df.isnull().sum()
```

Out[4]:

```
CASENUM       0
CLMSEX       12
CLMINSUR     41
SEATBELT     48
CLMAGE      189
LOSS          0
ATTORNEY      0
dtype: int64
```

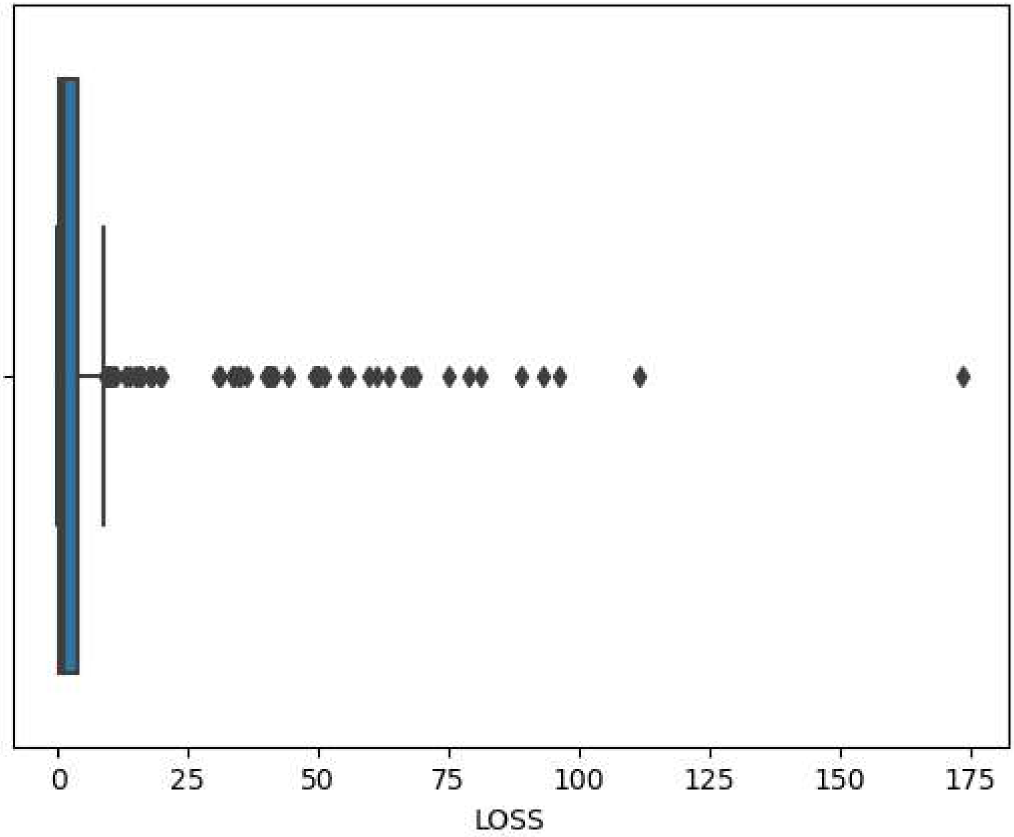
In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1340 entries, 0 to 1339
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   CASENUM   1340 non-null   int64  
 1   CLMSEX    1328 non-null   float64
 2   CLMINSUR  1299 non-null   float64
 3   SEATBELT  1292 non-null   float64
 4   CLMAGE    1151 non-null   float64
 5   LOSS      1340 non-null   float64
 6   ATTORNEY  1340 non-null   int64  
dtypes: float64(5), int64(2)
memory usage: 73.4 KB
```

In [6]:

```
sns.boxplot(x=df['LOSS'])
plt.show()
```



Detection of outlier (Based on IQR)

Calculate Q1 & Q3 ,then IQR , and finally Upper & lower limit

In [7]:

```
Q1=df['LOSS'].quantile(0.25)
Q1
```

Out[7]: 0.4

In [8]:

```
Q3=df['LOSS'].quantile(0.75)
Q3
```

Out[8]:

```
3.7815
```

In [9]:

```
IQR=Q3-Q1
IQR
```

Out[9]:

```
3.3815
```

In [10]:

```
upper_lim=Q3+(1.5*IQR)
lower_lim=Q1-(1.5*IQR)
print(upper_lim,lower_lim)
```

```
8.85375 -4.67225
```

In [11]:

```
df[df['LOSS']>8.85]
```

Out[11]:

	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
0	5	0.0	1.0	0.0	50.0	34.940	0
11	148	0.0	1.0	0.0	41.0	19.610	0
22	550	0.0	0.0	0.0	38.0	16.161	0
24	580	0.0	1.0	0.0	54.0	10.040	1
43	941	1.0	1.0	0.0	55.0	13.100	1
...	...	...	...	...	...	...	...
1179	30430	1.0	1.0	0.0	37.0	13.789	0
1188	30588	1.0	1.0	0.0	44.0	13.000	0
1256	31159	0.0	1.0	1.0	30.0	30.640	0
1286	33037	1.0	1.0	1.0	44.0	55.709	0
1312	33661	1.0	1.0	0.0	45.0	14.884	0

66 rows × 7 columns

Dealing with Outlier

.1.REMOVE/TRIMMING

In [12]:

```
df_trimmed=df[(df['LOSS']>lower_lim) & (df['LOSS']<upper_lim)]
df_trimmed
```

Out[12]:

	CASENUM	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS	ATTORNEY
1	3	1.0	0.0	0.0	18.0	0.891	1
2	66	0.0	1.0	0.0	5.0	0.330	1
3	70	0.0	1.0	1.0	31.0	0.037	0
4	96	0.0	1.0	0.0	30.0	0.038	1
5	97	1.0	1.0	0.0	35.0	0.309	0
...	...	...	...	...	...	...	...
1335	34100	0.0	1.0	0.0	NaN	0.576	1
1336	34110	1.0	1.0	0.0	46.0	3.705	0
1337	34113	1.0	1.0	0.0	39.0	0.099	1
1338	34145	1.0	0.0	0.0	8.0	3.177	0
1339	34153	1.0	1.0	0.0	30.0	0.688	1

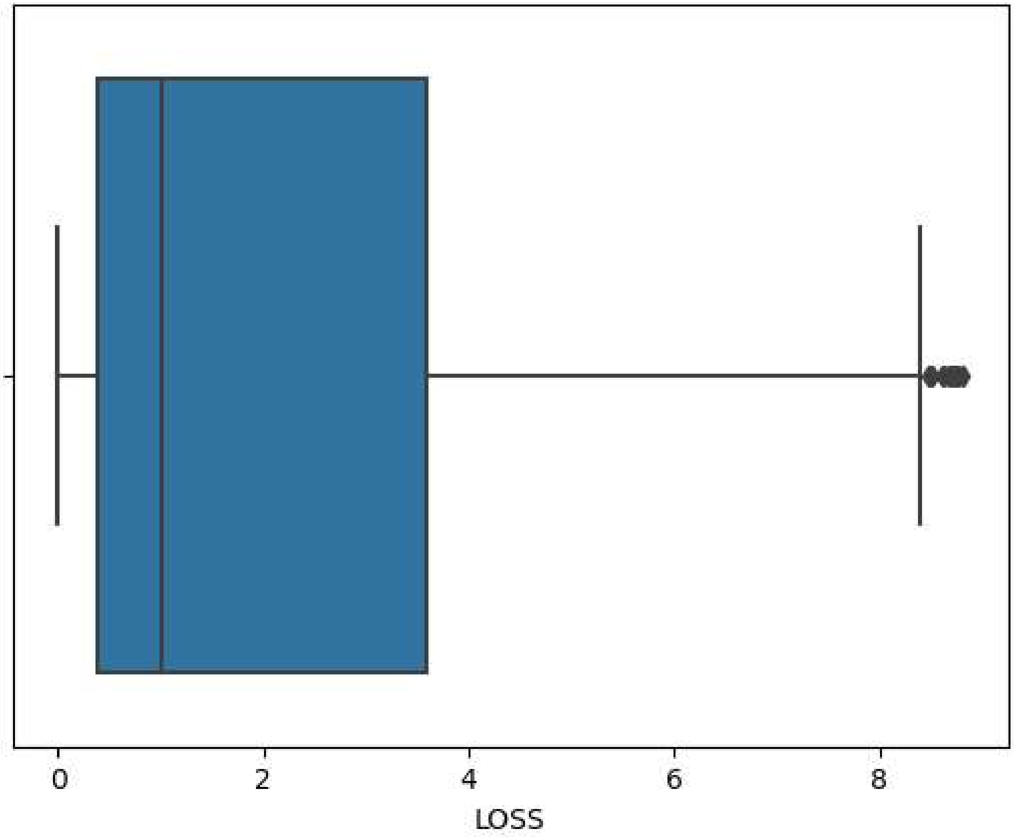
1274 rows × 7 columns

In [13]:

```
sns.boxplot(x=df_trimmed['LOSS'])
plt.show
```

Out[13]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



we can still see some outliers now again we have to #Calculate Q1 & Q3 ,then IQR , and finally Upper & lower limit

In [14]:

```
#best TECHINIQUES---2. REPLACE the Outliers
```

In [17]:

```
pip install feature_engine
```

```
Collecting feature_engine
  Downloading feature_engine-1.5.1-py2.py3-none-any.whl (285 kB)
     -------------------------------------- 285.3/285.3 kB 4.4 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn>=1.0.0 in c:\users\dell\anaconda3\envs\ten
sorflow\lib\site-packages (from feature_engine) (1.1.3)
Requirement already satisfied: statsmodels>=0.11.1 in c:\users\dell\anaconda3\envs\ten
sorflow\lib\site-packages (from feature_engine) (0.13.2)
Requirement already satisfied: scipy>=1.4.1 in c:\users\dell\anaconda3\envs\tensorflow
\lib\site-packages (from feature_engine) (1.7.3)
Requirement already satisfied: numpy>=1.18.2 in c:\users\dell\anaconda3\envs\tensorflo
w\lib\site-packages (from feature_engine) (1.21.5)
Requirement already satisfied: pandas>=1.0.3 in c:\users\dell\anaconda3\envs\tensorflo
w\lib\site-packages (from feature_engine) (1.4.4)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\dell\anaconda3\envs
\tensorflow\lib\site-packages (from pandas>=1.0.3->feature_engine) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\dell\anaconda3\envs\tensorflow
\lib\site-packages (from pandas>=1.0.3->feature_engine) (2022.1)
Requirement already satisfied: joblib>=1.0.0 in c:\users\dell\anaconda3\envs\tensorflo
w\lib\site-packages (from scikit-learn>=1.0.0->feature_engine) (1.1.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\dell\anaconda3\envs\te
nsorflow\lib\site-packages (from scikit-learn>=1.0.0->feature_engine) (2.2.0)
Requirement already satisfied: patsy>=0.5.2 in c:\users\dell\anaconda3\envs\tensorflow
\lib\site-packages (from statsmodels>=0.11.1->feature_engine) (0.5.2)
Requirement already satisfied: packaging>=21.3 in c:\users\dell\anaconda3\envs\tensorf
low\lib\site-packages (from statsmodels>=0.11.1->feature_engine) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\dell\anaconda3\env
s\tensorflow\lib\site-packages (from packaging>=21.3->statsmodels>=0.11.1->feature_eng
ine) (3.0.9)
Requirement already satisfied: six in c:\users\dell\anaconda3\envs\tensorflow\lib\site
-packages (from patsy>=0.5.2->statsmodels>=0.11.1->feature_engine) (1.16.0)
Installing collected packages: feature_engine
Successfully installed feature_engine-1.5.1
Note: you may need to restart the kernel to use updated packages.
```

In [18]:

```
from feature_engine.outliers import Winsorizer
```

In [19]:

```python
from feature_engine.outliers import Winsorizer
win=Winsorizer(capping_method='iqr',tail='both',fold=1.5,variables=['LOSS'])
df_win=win.fit_transform(df[['LOSS']])
df_win
```

Out[19]:

	LOSS
0	8.85375
1	0.89100
2	0.33000
3	0.03700
4	0.03800
...	...
1335	0.57600
1336	3.70500
1337	0.09900
1338	3.17700
1339	0.68800

1340 rows × 1 columns

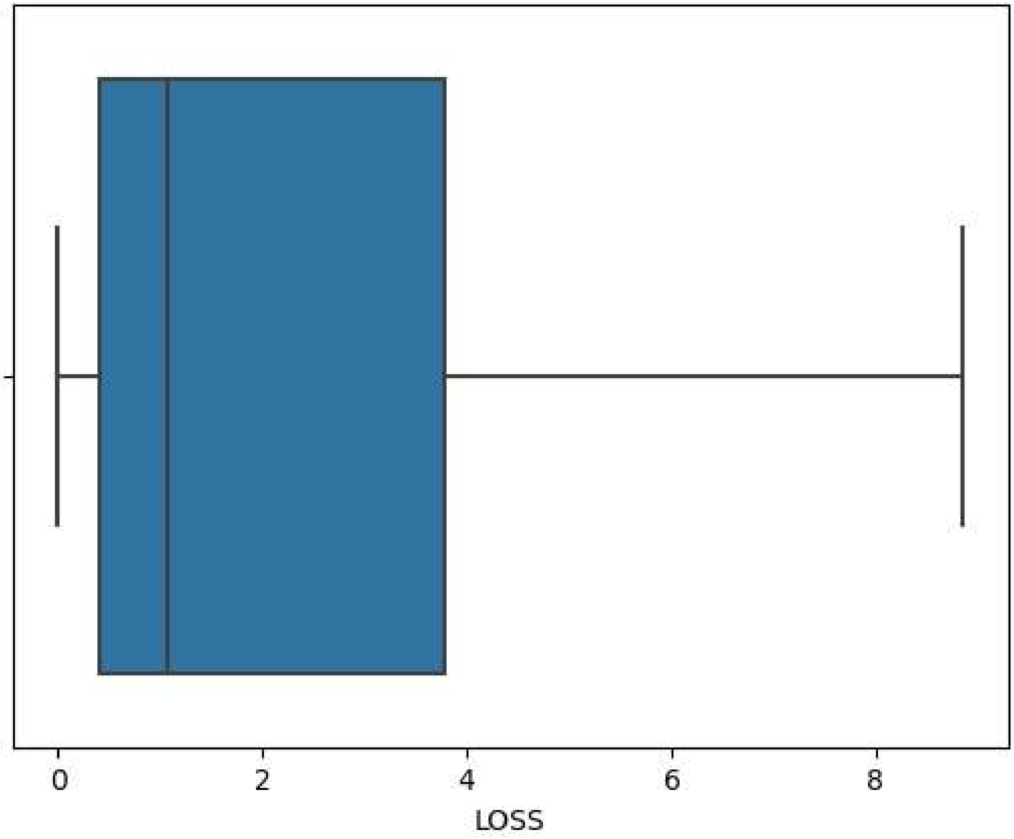
In [20]:

```python
print(win.left_tail_caps_,win.right_tail_caps_)
```

```
{'LOSS': -4.67225} {'LOSS': 8.85375}
```

In [21]:

```python
sns.boxplot(x=df_win['LOSS'])
plt.show()
```



step-3 REPLACE Arnitrary Outlier Capper(minimum & maximum values are

Step 3 REPLACE Arbitrary Outlier Capper(minimum & maximum values are determined by the user)

In [22]:

```python
#we get min & max values by domain expert
```

In [23]:

```python
from feature_engine.outliers import ArbitraryOutlierCapper
capper=ArbitraryOutlierCapper(max_capping_dict= {'LOSS':6},
                              min_capping_dict= {'LOSS':0.03})
df_c=capper.fit_transform(df[['LOSS']])
df_c
```

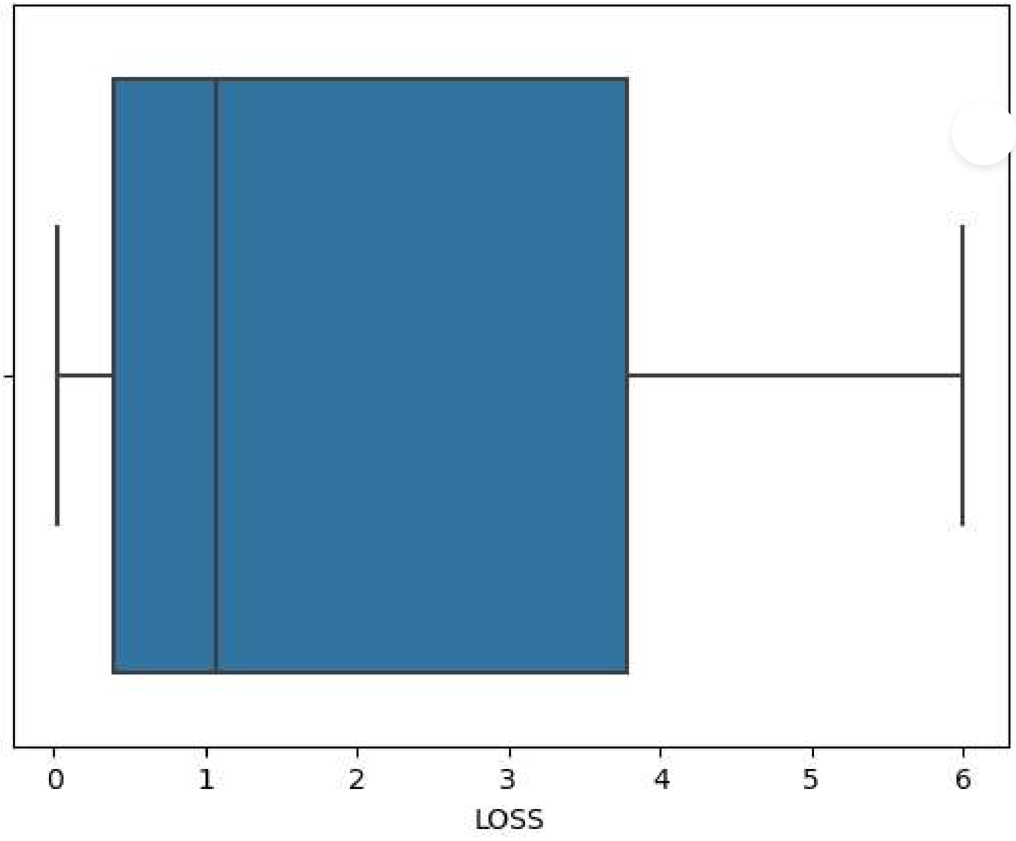
Out[23]:

	LOSS
0	6.000
1	0.891
2	0.330
3	0.037
4	0.038
...	...
1335	0.576
1336	3.705
1337	0.099
1338	3.177
1339	0.688

1340 rows × 1 columns

In [24]:

```python
sns.boxplot(x=df_c['LOSS'])
plt.show()
```


LOSS