Diabetes dataset to be cleaned using pandas library

```
this dataset shall be  go through with following steps to clean it:
1.checking the number of columns
2.checking misplet columns names to correct them
3.checkingr duplicate and missing values
4.chekcing for outliers
```

In [1]:
```
#import basic libraries and read the datsate

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv(r"C:\Users\verma\Downloads\diabetes_unclean.csv")
df
```

Out[1]:

	ID	No_Pation	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI
0	502	17975	F	50.0	4.7	46.0	4.9	4.2	0.9	2.4	1.4	0.5	24.0
1	735	34221	M	26.0	4.5	62.0	4.9	3.7	1.4	1.1	2.1	0.6	23.0
2	420	47975	F	50.0	4.7	46.0	4.9	4.2	0.9	2.4	1.4	0.5	24.0
3	680	87656	F	50.0	4.7	46.0	4.9	4.2	0.9	2.4	1.4	0.5	24.0
4	504	34223	M	33.0	7.1	46.0	4.9	4.9	1.0	0.8	2.0	0.4	21.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1004	191	454316	M	55.0	NaN	62.0	6.8	5.3	2.0	1.0	3.5	0.9	30.1
1005	192	454316	M	55.0	4.8	88.0	NaN	5.7	4.0	0.9	3.3	1.8	30.0
1006	193	454316	M	62.0	6.3	82.0	6.7	5.3	2.0	1.0	3.5	NaN	30.1
1007	194	454316	F	57.0	4.1	70.0	9.3	5.3	3.3	1.0	1.4	1.3	29.0
1008	195	4543	f	55.0	4.1	34.0	13.9	5.4	1.6	1.6	3.1	0.7	33.0

1009 rows × 14 columns

In [2]:
```
#getting infor mation the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1009 entries, 0 to 1008
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   ID         1009 non-null   int64  
 1   No_Pation  1009 non-null   int64  
 2   Gender     1009 non-null   object 
 3   AGE        1008 non-null   float64
 4   Urea       1008 non-null   float64
 5   Cr         1007 non-null   float64
 6   HbA1c      1006 non-null   float64
 7   Chol       1007 non-null   float64
 8   TG         1007 non-null   float64
 9   HDL        1008 non-null   float64
 10  LDL        1007 non-null   float64
 11  VLDL       1008 non-null   float64
 12  BMI        1009 non-null   float64
 13  CLASS      1009 non-null   object 
dtypes: float64(10), int64(2), object(2)
memory usage: 110.5+ KB
```

In [3]:
```
#check whether any column has blank value or not
df.isnull().any()
```

Out[3]:
```
ID           False
No_Pation    False
Gender       False
AGE           True
Urea          True
Cr            True
HbA1c         True
Chol          True
TG            True
HDL           True
LDL           True
VLDL          True
BMI          False
CLASS        False
dtype: bool
```

In [4]:
```
#top 5 row and column
df.head()
```

Out[4]:

	ID	No_Pation	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI	CL
0	502	17975	F	50.0	4.7	46.0	4.9	4.2	0.9	2.4	1.4	0.5	24.0	
1	735	34221	M	26.0	4.5	62.0	4.9	3.7	1.4	1.1	2.1	0.6	23.0	
2	420	47975	F	50.0	4.7	46.0	4.9	4.2	0.9	2.4	1.4	0.5	24.0	
3	680	87656	F	50.0	4.7	46.0	4.9	4.2	0.9	2.4	1.4	0.5	24.0	
4	504	34223	M	33.0	7.1	46.0	4.9	4.9	1.0	0.8	2.0	0.4	21.0	

In [5]:
```
#check the columns
df.columns
```

Out[5]:
```
Index(['ID', 'No_Pation', 'Gender', 'AGE', 'Urea', 'Cr', 'HbA1c', 'Chol',
'TG',
       'HDL', 'LDL', 'VLDL', 'BMI', 'CLASS'],
      dtype='object')
```

In [6]:
```
#change the column name
df.rename(columns = {"No_Pation":"Patients_No"},inplace = True)
df
```

Out[6]:

	ID	Patients_No	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI
0	502	17975	F	50.0	4.7	46.0	4.9	4.2	0.9	2.4	1.4	0.5	24.0
1	735	34221	M	26.0	4.5	62.0	4.9	3.7	1.4	1.1	2.1	0.6	23.0
2	420	47975	F	50.0	4.7	46.0	4.9	4.2	0.9	2.4	1.4	0.5	24.0
3	680	87656	F	50.0	4.7	46.0	4.9	4.2	0.9	2.4	1.4	0.5	24.0
4	504	34223	M	33.0	7.1	46.0	4.9	4.9	1.0	0.8	2.0	0.4	21.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1004	191	454316	M	55.0	NaN	62.0	6.8	5.3	2.0	1.0	3.5	0.9	30.1
1005	192	454316	M	55.0	4.8	88.0	NaN	5.7	4.0	0.9	3.3	1.8	30.0
1006	193	454316	M	62.0	6.3	82.0	6.7	5.3	2.0	1.0	3.5	NaN	30.1
1007	194	454316	F	57.0	4.1	70.0	9.3	5.3	3.3	1.0	1.4	1.3	29.0
1008	195	4543	f	55.0	4.1	34.0	13.9	5.4	1.6	1.6	3.1	0.7	33.0

1009 rows × 14 columns

In [7]:
```
#check for the missing values
df.isnull().sum()
```

Out[7]:
```
ID             0
Patients_No    0
Gender         0
AGE            1
Urea           1
Cr             2
HbA1c          3
Chol           2
TG             2
HDL            1
LDL            2
VLDL           1
BMI            0
CLASS          0
dtype: int64
```

In [8]:
```
#replace the missing values in "HbA1c"column
mean_value =df["HbA1c"].mean()
mean_value
```

Out[8]:
```
8.284155069582505
```

In [9]:
```
#by using mean_value to repalce the missing values in that column

df["HbA1c"].fillna(mean_value,inplace = True)
#to check for the change
df.isnull().sum()
```

Out[9]:
```
ID             0
Patients_No    0
Gender         0
AGE            1
Urea           1
Cr             2
HbA1c          0
Chol           2
TG             2
HDL            1
LDL            2
VLDL           1
BMI            0
CLASS          0
dtype: int64
```

In [10]:
```
#droping the missing values from other columns
df1 = df.dropna()
df1.isnull().sum()
```

Out[10]:
```
ID             0
Patients_No    0
Gender         0
AGE            0
Urea           0
Cr             0
HbA1c          0
Chol           0
TG             0
HDL            0
LDL            0
VLDL           0
BMI            0
CLASS          0
dtype: int64
```

In [11]:
```
#check the information of the dataset
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 997 entries, 0 to 1008
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   ID           997 non-null    int64  
 1   Patients_No  997 non-null    int64  
 2   Gender       997 non-null    object 
 3   AGE          997 non-null    float64
 4   Urea         997 non-null    float64
 5   Cr           997 non-null    float64
 6   HbA1c        997 non-null    float64
 7   Chol         997 non-null    float64
 8   TG           997 non-null    float64
 9   HDL          997 non-null    float64
 10  LDL          997 non-null    float64
 11  VLDL         997 non-null    float64
 12  BMI          997 non-null    float64
 13  CLASS        997 non-null    object 
dtypes: float64(10), int64(2), object(2)
memory usage: 116.8+ KB
```

In [12]:
```
#to split data into groups based on some criteria. After grouping, you can 
df1.groupby('CLASS')['CLASS'].agg("count")
```

Out[12]:
```
CLASS
N     102
N       1
P      53
Y     832
Y       9
Name: CLASS, dtype: int64
```

In [13]:
```
df1
```

Out[13]:

	ID	Patients_No	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL
0	502	17975	F	50.0	4.7	46.0	4.900000	4.2	0.9	2.4	1.4	0.5
1	735	34221	M	26.0	4.5	62.0	4.900000	3.7	1.4	1.1	2.1	0.6
2	420	47975	F	50.0	4.7	46.0	4.900000	4.2	0.9	2.4	1.4	0.5
3	680	87656	F	50.0	4.7	46.0	4.900000	4.2	0.9	2.4	1.4	0.5
4	504	34223	M	33.0	7.1	46.0	4.900000	4.9	1.0	0.8	2.0	0.4
...	...	...	...	...	...	...	...	...	...	...	...	...
1002	188	454316	F	75.0	10.3	113.0	8.600000	4.2	1.6	0.9	2.6	0.7
1003	189	454316	M	58.0	4.0	55.0	7.900000	4.9	2.0	1.2	1.4	1.1
1005	192	454316	M	55.0	4.8	88.0	8.284155	5.7	4.0	0.9	3.3	1.8
1007	194	454316	F	57.0	4.1	70.0	9.300000	5.3	3.3	1.0	1.4	1.3
1008	195	4543	f	55.0	4.1	34.0	13.900000	5.4	1.6	1.6	3.1	0.7

997 rows × 14 columns

In [ ]:
```
```

In [14]:
```
#check for the unique values in "CLASS"
df['CLASS'].unique()
```

Out[14]:
```
array(['N', 'N ', 'P', 'Y', 'Y '], dtype=object)
```

In [15]:
```
#to clean the column , I replace the unique values
df1.loc[:, 'CLASS'] = df1['CLASS'].str.replace("Y ", "Y")
df1.loc[:, 'CLASS'] = df1['CLASS'].str.replace("N ", "N")
df1
```

Out[15]:

	ID	Patients_No	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL
0	502	17975	F	50.0	4.7	46.0	4.900000	4.2	0.9	2.4	1.4	0.5
1	735	34221	M	26.0	4.5	62.0	4.900000	3.7	1.4	1.1	2.1	0.6
2	420	47975	F	50.0	4.7	46.0	4.900000	4.2	0.9	2.4	1.4	0.5
3	680	87656	F	50.0	4.7	46.0	4.900000	4.2	0.9	2.4	1.4	0.5
4	504	34223	M	33.0	7.1	46.0	4.900000	4.9	1.0	0.8	2.0	0.4
...	...	...	...	...	...	...	...	...	...	...	...	...
1002	188	454316	F	75.0	10.3	113.0	8.600000	4.2	1.6	0.9	2.6	0.7
1003	189	454316	M	58.0	4.0	55.0	7.900000	4.9	2.0	1.2	1.4	1.1
1005	192	454316	M	55.0	4.8	88.0	8.284155	5.7	4.0	0.9	3.3	1.8
1007	194	454316	F	57.0	4.1	70.0	9.300000	5.3	3.3	1.0	1.4	1.3
1008	195	4543	f	55.0	4.1	34.0	13.900000	5.4	1.6	1.6	3.1	0.7

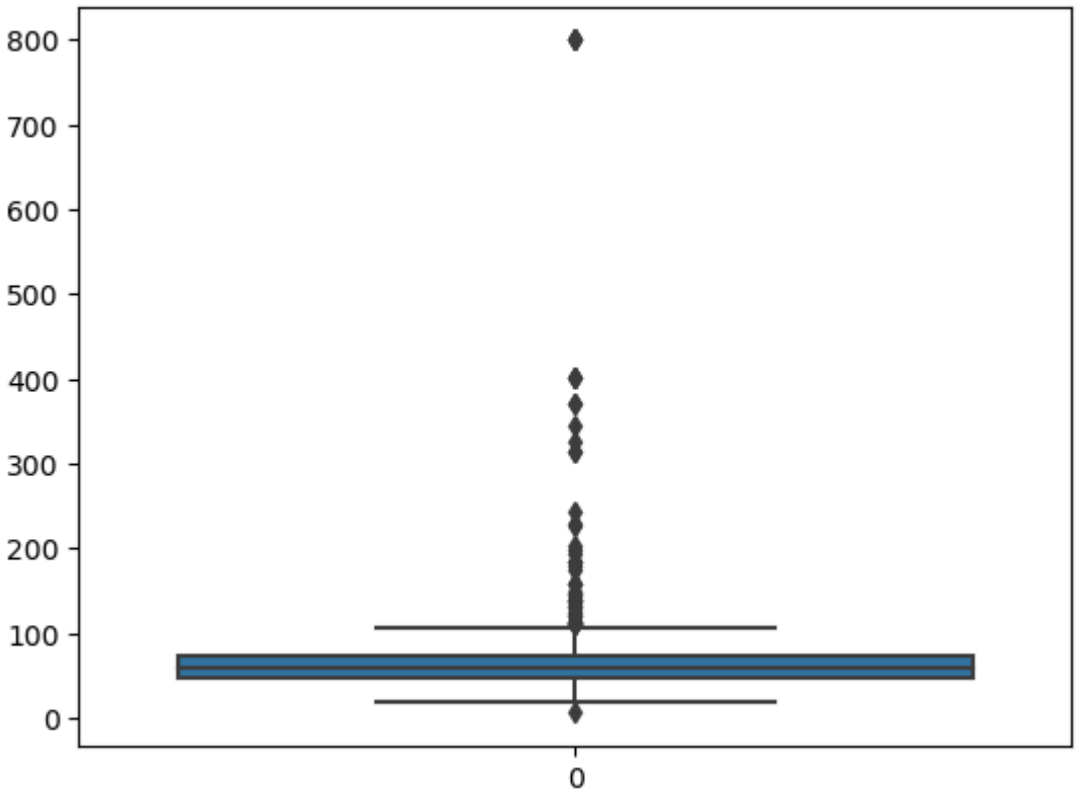
997 rows × 14 columns

In [16]:
```
df1["CLASS"].unique()
```

Out[16]:
```
array(['N', 'P', 'Y'], dtype=object)
```

In [17]:
```
#checking the outliers in the dataset
sns.boxplot(df1["Cr"])
plt.show
```

Out[17]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [18]:
```
#choose the maximum quantile to fill the outliers
max_cr = df1["Cr"].quantile(0.995)
max_cr
```

Out[18]:
```
401.0
```

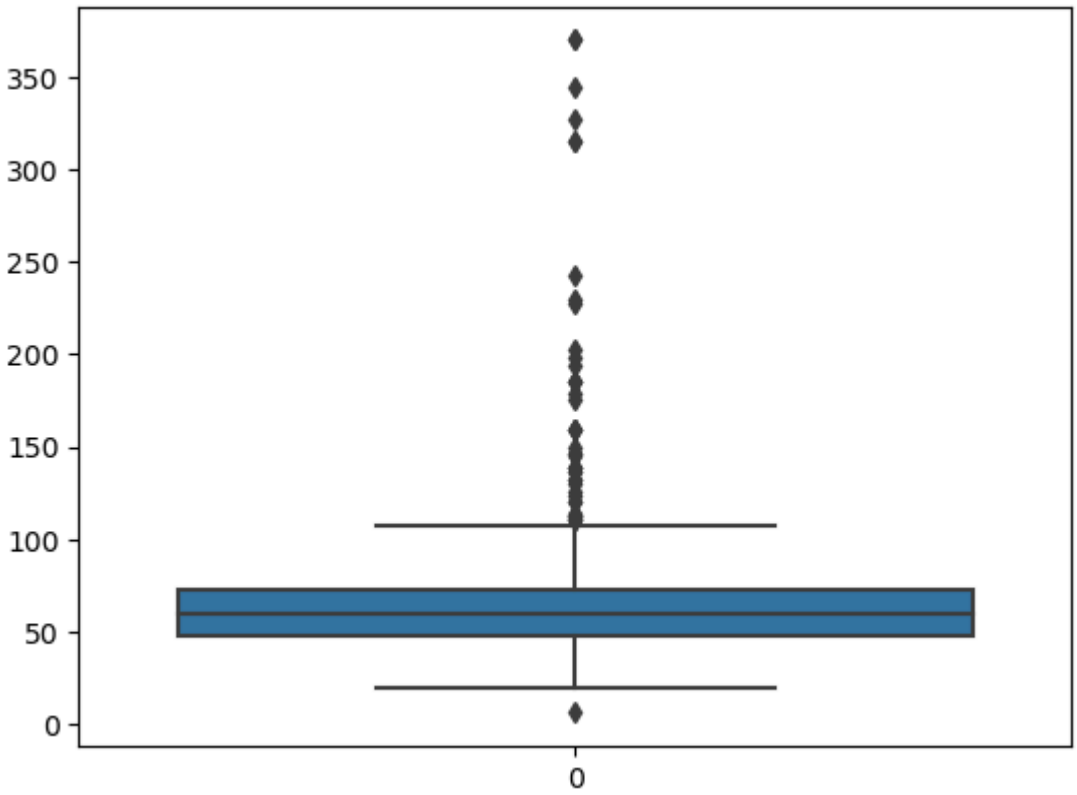
In [19]:
```
df1[df1["Cr"] > max_cr]
```

Out[19]:

	ID	Patients_No	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI
273	1	34325	M	58.0	20.8	800.0	9.1	6.6	2.9	1.1	4.3	1.3	33.0
283	266	24060	M	58.0	20.8	800.0	9.1	6.6	2.9	1.1	4.3	1.3	33.0
846	1	34325	M	56.0	20.8	800.0	9.0	4.6	2.0	1.2	2.5	0.9	35.0
860	19	51623	M	60.0	20.8	800.0	9.0	2.3	1.1	0.9	0.9	0.5	33.0

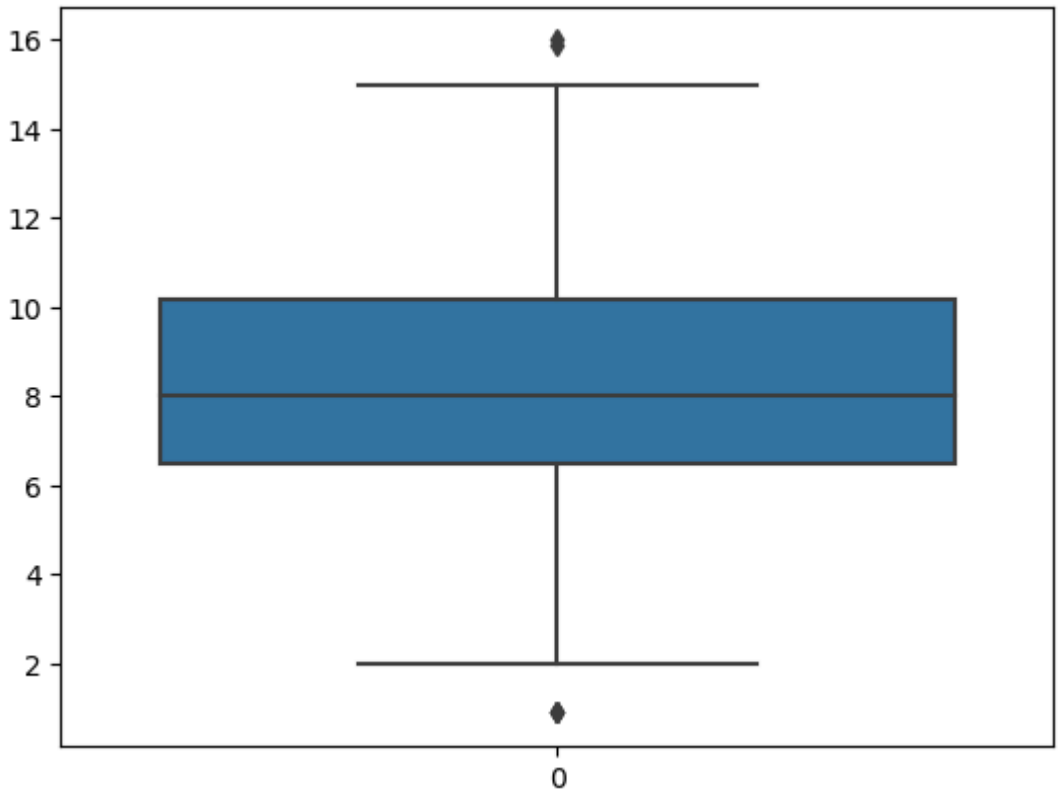
In [20]:
```
#assign a new dataframe
df2 = df1[df1["Cr"] < max_cr]
#to confirm the change
sns.boxplot(df2["Cr"])
plt.show
```

Out[20]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [21]:
```
#check the outliers for "HbA1c" column
sns.boxplot(df2["HbA1c"])
plt.show
```

Out[21]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [22]:
```
#checking for the duplicates values
df2.duplicated()
```

Out[22]:
```
0       False
1       False
2       False
3       False
4       False
        ...
1002     True
1003     True
1005    False
1007    False
1008     True
Length: 990, dtype: bool
```

In [23]:
```
#drop duplicate values
df3 = df2.drop_duplicates()
```

In [24]:
```
#check duplicates
df3.duplicated().sum()
```

Out[24]:
```
0
```

In [25]:
```
#save the new dataset into a csv file
df3.to_csv("Cleaned_data2.csv")
```