EXERCISE 2

AIM: -

Apply the following Pre-processing techniques for a given dataset.

a. Attribute selection

b. Handling Missing Values

c. Discretization

d. Elimination of Outliers

a. Attribute Selection Methods: -

1.Unvariate Selection:

In [11]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
data=pd.read_csv("train.csv")
```

In [13]:
```
data.head()
```

Out[13]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobil
0	842	0	2.2	0	1	0	7	0.6	
1	1021	1	0.5	1	0	1	53	0.7	
2	563	1	0.5	1	2	1	41	0.9	
3	615	1	2.5	0	0	0	10	0.8	
4	1821	1	1.2	0	13	1	44	0.6	

5 rows × 21 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   battery_power  2000 non-null   int64  
 1   blue           2000 non-null   int64  
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64  
 4   fc             2000 non-null   int64  
 5   four_g         2000 non-null   int64  
 6   int_memory     2000 non-null   int64  
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64  
 9   n_cores        2000 non-null   int64  
 10  pc             2000 non-null   int64  
 11  px_height      2000 non-null   int64  
 12  px_width       2000 non-null   int64  
 13  ram            2000 non-null   int64  
 14  sc_h           2000 non-null   int64  
 15  sc_w           2000 non-null   int64  
 16  talk_time      2000 non-null   int64  
 17  three_g        2000 non-null   int64  
 18  touch_screen   2000 non-null   int64  
 19  wifi           2000 non-null   int64  
 20  price_range    2000 non-null   int64  
dtypes: float64(2), int64(19)
memory usage: 328.3 KB
```

In [21]:
```
x=data.iloc[:, :-1]
y=data.iloc[:, -1]
```

In [23]:
```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   battery_power  2000 non-null   int64  
 1   blue           2000 non-null   int64  
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64  
 4   fc             2000 non-null   int64  
 5   four_g         2000 non-null   int64  
 6   int_memory     2000 non-null   int64  
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64  
 9   n_cores        2000 non-null   int64  
 10  pc             2000 non-null   int64  
 11  px_height      2000 non-null   int64  
 12  px_width       2000 non-null   int64  
 13  ram            2000 non-null   int64  
 14  sc_h           2000 non-null   int64  
 15  sc_w           2000 non-null   int64  
 16  talk_time      2000 non-null   int64  
 17  three_g        2000 non-null   int64  
 18  touch_screen   2000 non-null   int64  
 19  wifi           2000 non-null   int64  
dtypes: float64(2), int64(18)
memory usage: 312.6 KB
```

In [25]:
```
y.info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2000 entries, 0 to 1999
Series name: price_range
Non-Null Count  Dtype
--------------  -----
2000 non-null   int64
dtypes: int64(1)
memory usage: 15.8 KB
```

In [35]:
```
bestfeatures=SelectKBest(score_func=chi2,k=10)
fit=bestfeatures.fit(x,y)
```

In [37]:
```
dfscores=pd.DataFrame(fit.scores_)
dfcolumns=pd.DataFrame(x.columns)
```

In [39]:
```
featureScores=pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns=['Feature','Score']
```

In [43]:
```
featureScores
```

Out[43]:

	Feature	Score
0	battery_power	14129.866576
1	blue	0.723232
2	clock_speed	0.648366
3	dual_sim	0.631011
4	fc	10.135166
5	four_g	1.521572
6	int_memory	89.839124
7	m_dep	0.745820
8	mobile_wt	95.972863
9	n_cores	9.097556
10	pc	9.186054
11	px_height	17363.569536
12	px_width	9810.586750
13	ram	931267.519053
14	sc_h	9.614878
15	sc_w	16.480319
16	talk_time	13.236400
17	three_g	0.327643
18	touch_screen	1.928429
19	wifi	0.422091

In [45]:

```
print(featureScores.nlargest(10,'Score'))
```

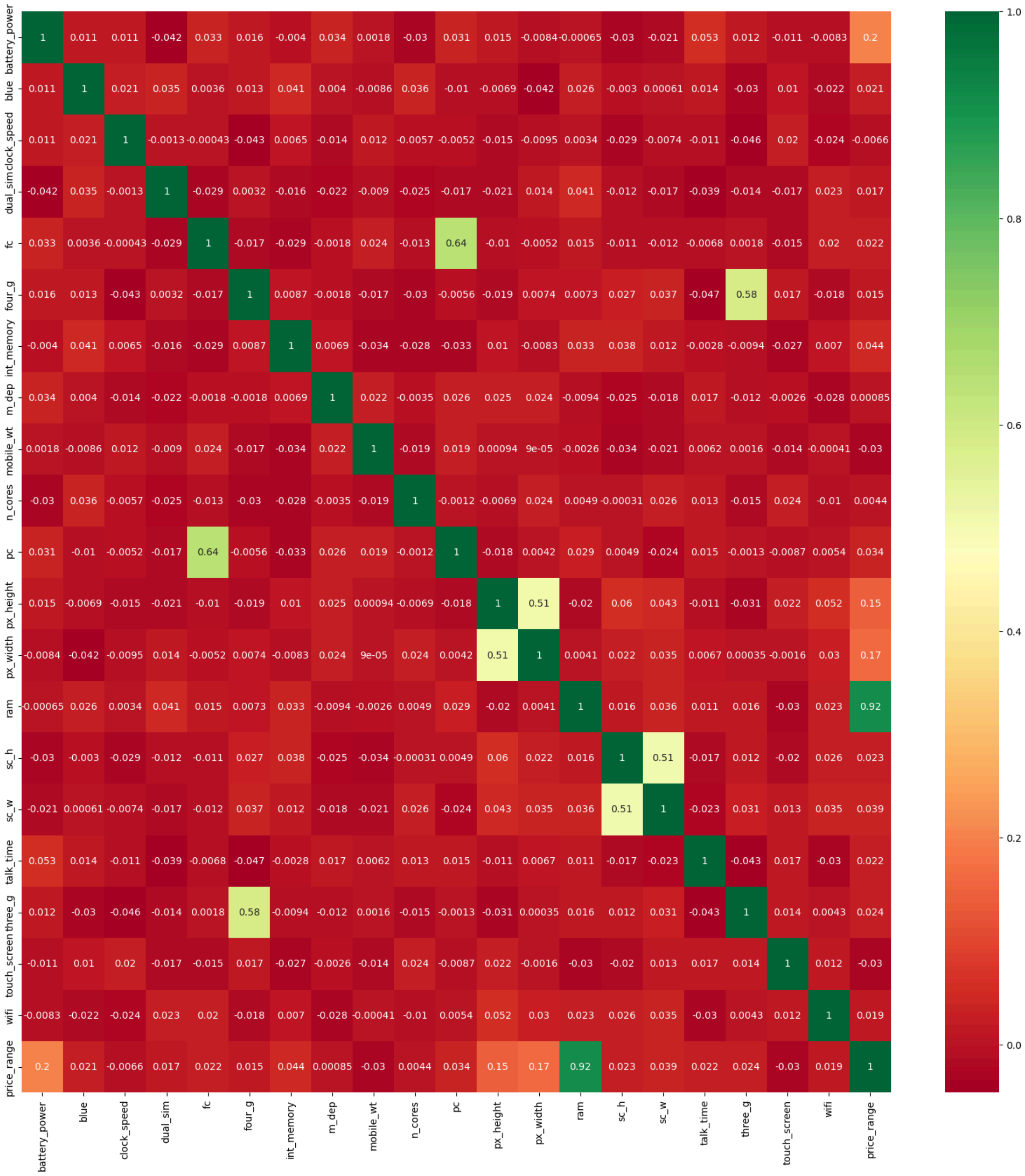
```
          Feature          Score
13            ram  931267.519053
11      px_height   17363.569536
0   battery_power   14129.866576
12       px_width    9810.586750
8       mobile_wt      95.972863
6      int_memory      89.839124
15           sc_w      16.480319
16      talk_time      13.236400
4              fc      10.135166
14           sc_h       9.614878
```

Correlation Matrix with Heatmap: -

In [48]:

```
import seaborn as sns
corrmat = data.corr()
top_corr_features = corrmat.index
```

```python
plt.figure(figsize=(20,20))
g=sns.heatmap(data[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



Feature Selection: Removing Low-Impact Features: -

In [63]:
```python
data_cleaned = data.drop(['px_width','sc_w','three_g','wifi','dual_sim','clock_s
print("Remaining Features:", data_cleaned.columns)
```

```
Remaining Features: Index(['battery_power', 'fc', 'four_g', 'int_memory', 'm_de
p', 'mobile_wt',
       'n_cores', 'pc', 'px_height', 'ram', 'sc_h', 'talk_time',
       'touch_screen', 'price_range'],
      dtype='object')
```

In [65]:
```python
data_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   battery_power  2000 non-null   int64  
 1   fc             2000 non-null   int64  
 2   four_g         2000 non-null   int64  
 3   int_memory     2000 non-null   int64  
 4   m_dep          2000 non-null   float64
 5   mobile_wt      2000 non-null   int64  
 6   n_cores        2000 non-null   int64  
 7   pc             2000 non-null   int64  
 8   px_height      2000 non-null   int64  
 9   ram            2000 non-null   int64  
 10  sc_h           2000 non-null   int64  
 11  talk_time      2000 non-null   int64  
 12  touch_screen   2000 non-null   int64  
 13  price_range    2000 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 218.9 KB
```

b. Handling Missing Values: -

Load the Dataset: -

In [77]:
```
df = pd.read_csv('Loan_Prediction_Dataset .csv')
df.head()
```

Out[77]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001002	Male	No	0	Graduate	No	584
1	LP001003	Male	Yes	1	Graduate	No	458
2	LP001005	Male	Yes	0	Graduate	Yes	300
3	LP001006	Male	Yes	0	Not Graduate	No	258
4	LP001008	Male	No	0	Graduate	No	600

In [79]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

Check the missing values: -

In [81]:
```python
df.isnull().sum()
```

Out[81]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

Fill missing value with mean, median and mode: -

First let us copy the existing data frame to new data frame: -

In [86]:
```python
new_df=df.copy()
```

In [89]:
```python
df['LoanAmount'].mean()
```

Out[89]:
```
146.41216216216216
```

In [91]:
```python
new_df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].mean())
new_df.isnull().sum()
```

Out[91]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [93]:
```
df['Loan_Amount_Term'].median()
```

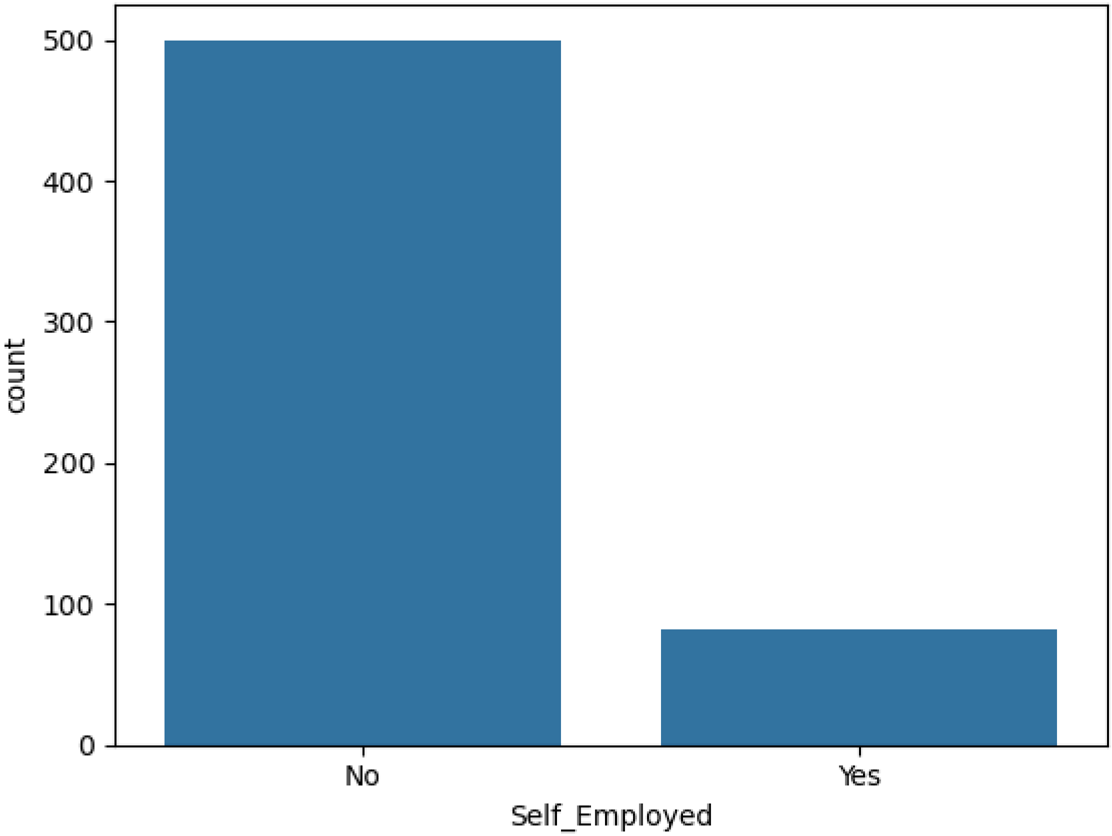
Out[93]:
```
360.0
```

In [95]:
```
new_df['Loan_Amount_Term'] = df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'
new_df.isnull().sum()
```

Out[95]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [97]:
```
sns.countplot(x=df['Self_Employed'])
```

Out[97]:
```
<Axes: xlabel='Self_Employed', ylabel='count'>
```



In [99]:
```
df['Self_Employed'].mode()[0]
```
Out[99]: 'No'

In [101…
```
new_df['Self_Employed'] = df['Self_Employed'].fillna(df['Self_Employed'].mode()[
new_df.isnull().sum()
```
Out[101…
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```
In [103…
```
df=pd.read_csv('Titanic-Dataset.csv')
df.head()
```

Out[103…

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	I
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0

In [105…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Handle missing values (important): -

In [107…

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

Out[107…

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [110…

```
df['Age'] = df['Age'].fillna(df['Age'].median())
```

In [112…

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

Out[112…

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

1. Equal-Width Discretization (Using pd.cut): -

In [115…

```
df['Age_bin_width'] = pd.cut(
    df['Age'],
    bins=3,
    labels=['Young', 'Adult', 'Senior']
)
```

In [117…

```
df[['Age', 'Age_bin_width']].head(10)
```

Out[117...

	Age	Age_bin_width
0	22.0	Young
1	38.0	Adult
2	26.0	Young
3	35.0	Adult
4	35.0	Adult
5	28.0	Adult
6	54.0	Senior
7	2.0	Young
8	27.0	Adult
9	14.0	Young

2. Equal-Frequency Discretization (Using pd.qcut): -

In [120...

```
df['Age_bin_freq'] = pd.qcut(
    df['Age'],
    q=4,
    labels=['Q1', 'Q2', 'Q3', 'Q4']
)
```

In [122...

```
df[['Age', 'Age_bin_freq']].head(10)
```

Out[122...

	Age	Age_bin_freq
0	22.0	Q1
1	38.0	Q4
2	26.0	Q2
3	35.0	Q3
4	35.0	Q3
5	28.0	Q2
6	54.0	Q4
7	2.0	Q1
8	27.0	Q2
9	14.0	Q1

3. Discretization using sklearn (KBinsDiscretizer): -

In [125...

```
from sklearn.preprocessing import KBinsDiscretizer
kb = KBinsDiscretizer(
    n_bins=4,
    encode='ordinal',
```

```
    strategy='uniform'
)
df['Age_kbins'] = kb.fit_transform(df[['Age']])

df[['Age', 'Age_kbins']].head(10)
```

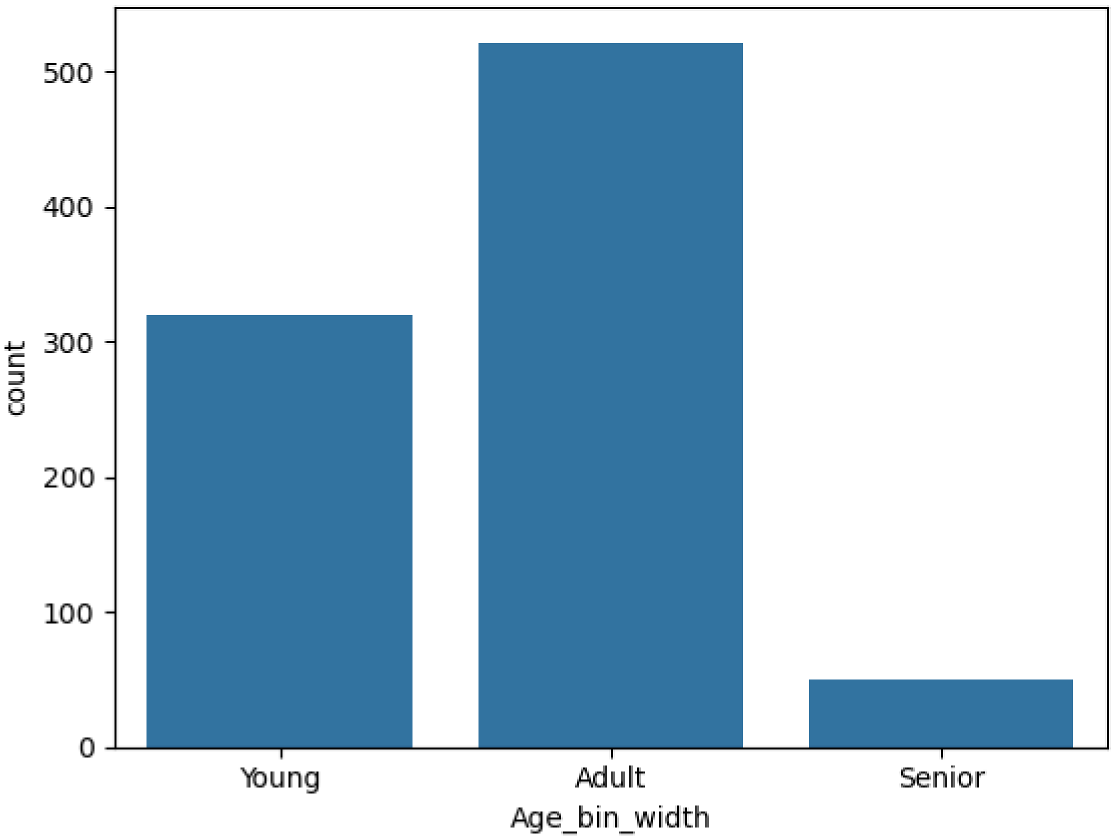
Out[125…

	Age	Age_kbins
0	22.0	1.0
1	38.0	1.0
2	26.0	1.0
3	35.0	1.0
4	35.0	1.0
5	28.0	1.0
6	54.0	2.0
7	2.0	0.0
8	27.0	1.0
9	14.0	0.0

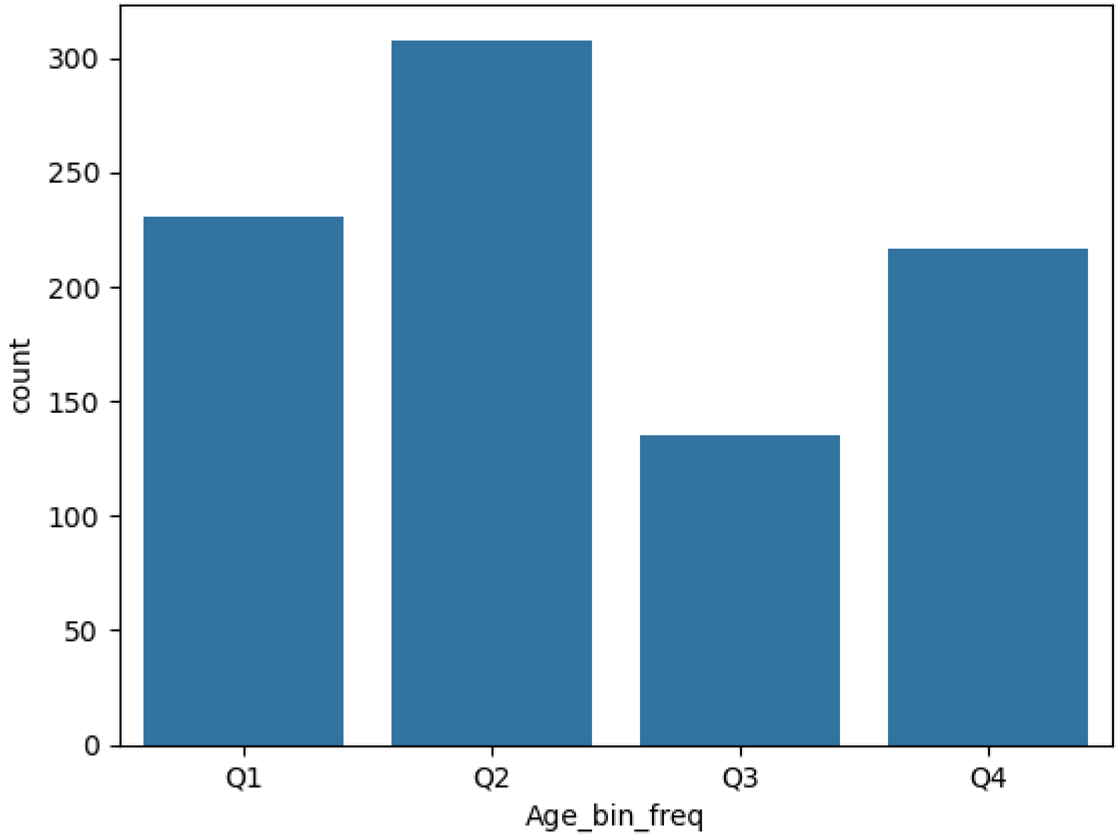
Visualize Discretized Data: -

In [128…

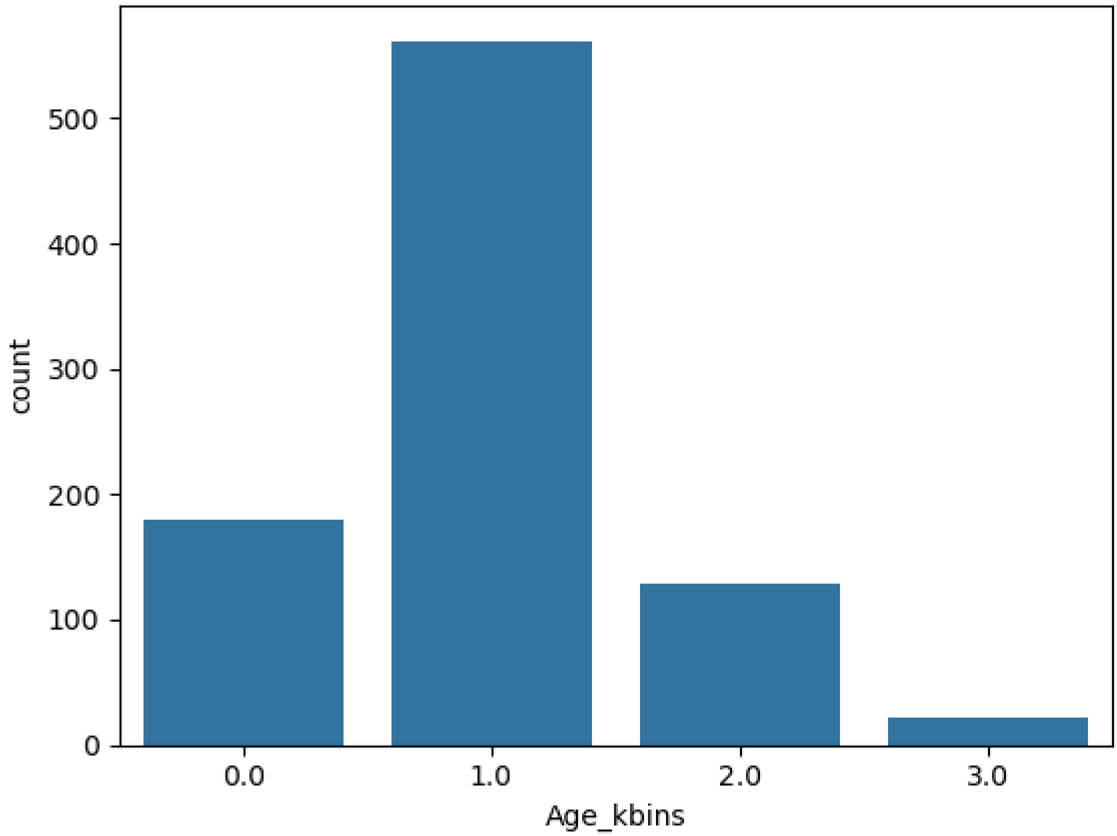
```
sns.countplot(x='Age_bin_width', data=df)
plt.show()
```



In [130…
```
sns.countplot(x='Age_bin_freq', data=df)
plt.show()
```



In [132…
```
sns.countplot(x='Age_kbins', data=df)
plt.show()
```



d. Elimination of Outliers: -

In [137…
```
df = pd.read_csv('winequality.csv')
df.head()
```

Out[137…

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	su
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	

In [139…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   type                  6497 non-null   object 
 1   fixed acidity         6487 non-null   float64
 2   volatile acidity      6489 non-null   float64
 3   citric acid           6494 non-null   float64
 4   residual sugar        6495 non-null   float64
 5   chlorides             6495 non-null   float64
 6   free sulfur dioxide   6497 non-null   float64
 7   total sulfur dioxide  6497 non-null   float64
 8   density               6497 non-null   float64
 9   pH                    6488 non-null   float64
 10  sulphates             6493 non-null   float64
 11  alcohol               6497 non-null   float64
 12  quality               6497 non-null   int64  
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
```

In [141…
```
df.isnull().sum()
```

Out[141…
```
type                     0
fixed acidity           10
volatile acidity         8
citric acid              3
residual sugar           2
chlorides                2
free sulfur dioxide      0
total sulfur dioxide     0
density                  0
pH                       9
sulphates                4
alcohol                  0
quality                  0
dtype: int64
```

In [143…

```
df.describe()
```

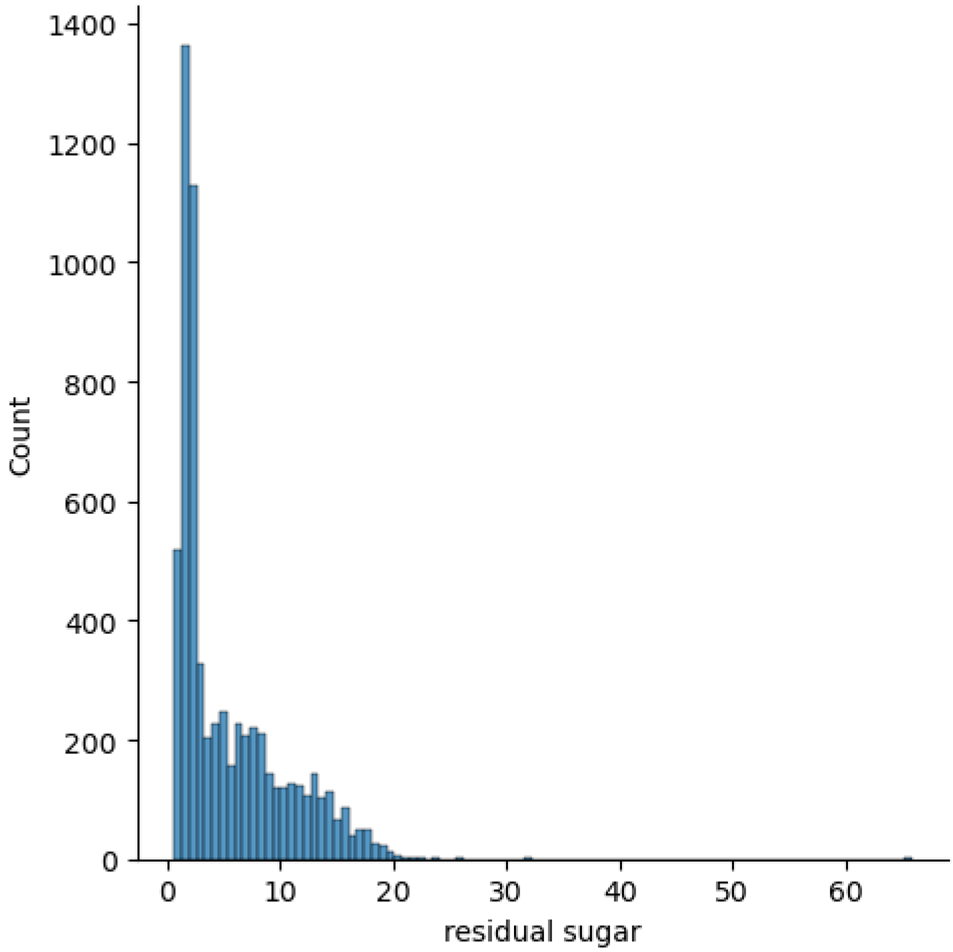
Out[143…

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide
count	6487.000000	6489.000000	6494.000000	6495.000000	6495.000000	6497.000000
mean	7.216579	0.339691	0.318722	5.444326	0.056042	30.525319
std	1.296750	0.164649	0.145265	4.758125	0.035036	17.749400
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000

Visualize the Data: -

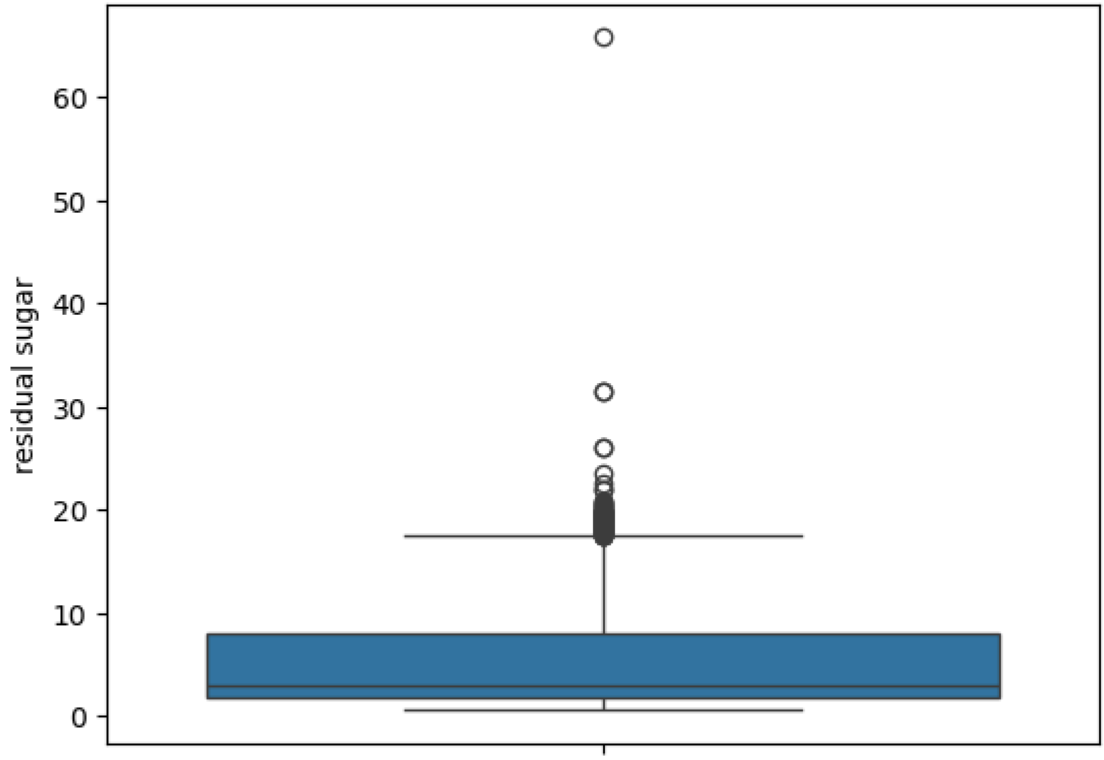
In [156…

```
sns.displot(df['residual sugar'])
plt.show()
```



In [150…

```
sns.boxplot(df['residual sugar'])
plt.show()
```



Methods to remove Outliers: -

First we will get the upper and lower limits

In [158…

```
upper_limit = df['residual sugar'].mean() + 3 * df['residual sugar'].std()
lower_limit = df['residual sugar'].mean() - 3 * df['residual sugar'].std()

print('Upper limit:', upper_limit)
print('Lower limit:', lower_limit)
```

```
Upper limit: 19.718700632944987
Lower limit: -8.830047823091254
```

In [162…

```
# Find the outliers
df.loc[
    (df['residual sugar'] > upper_limit) |
    (df['residual sugar'] < lower_limit)
]
```

Out[162…

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH
0	white	7.0	0.270	0.36	20.70	0.045	45.0	170.0	1.00100	3.00
7	white	7.0	0.270	0.36	20.70	0.045	45.0	170.0	1.00100	3.00
182	white	6.8	0.280	0.40	22.00	0.048	48.0	167.0	1.00100	2.93
191	white	6.8	0.280	0.40	22.00	0.048	48.0	167.0	1.00100	2.93
292	white	7.4	0.280	0.42	19.80	0.066	53.0	195.0	1.00000	2.96
444	white	6.9	0.240	0.36	20.80	0.031	40.0	139.0	0.99750	3.20
1454	white	8.3	0.210	0.49	19.80	0.054	50.0	231.0	1.00120	2.99
1608	white	6.9	0.270	0.49	23.50	0.057	59.0	235.0	1.00240	2.98
1653	white	7.9	0.330	0.28	31.60	0.053	35.0	176.0	1.01030	3.15
1663	white	7.9	0.330	0.28	31.60	0.053	35.0	176.0	1.01030	3.15
2489	white	6.1	0.280	0.24	19.95	0.074	32.0	174.0	0.99922	3.19
2492	white	6.1	0.280	0.24	19.95	0.074	32.0	174.0	0.99922	3.19
2620	white	6.5	0.280	0.28	20.40	0.041	40.0	144.0	1.00020	3.14
2781	white	7.8	0.965	0.60	65.80	0.074	8.0	160.0	1.03898	3.39
2785	white	6.4	0.240	0.25	20.20	0.083	35.0	157.0	0.99976	3.17
2787	white	6.4	0.240	0.25	20.20	0.083	35.0	157.0	0.99976	3.17
3014	white	7.0	0.450	0.34	19.80	0.040	12.0	67.0	0.99760	3.07
3023	white	7.0	0.450	0.34	19.80	0.040	12.0	67.0	0.99760	3.07
3420	white	7.6	0.280	0.49	20.15	0.060	30.0	145.0	1.00196	3.01
3497	white	7.7	0.430	1.00	19.95	0.032	42.0	164.0	0.99742	3.29
3547	white	7.3	0.200	0.29	19.90	0.039	69.0	237.0	1.00037	3.10
3619	white	6.8	0.450	0.28	26.05	0.031	27.0	122.0	1.00295	3.06
3623	white	6.8	0.450	0.28	26.05	0.031	27.0	122.0	1.00295	3.06
3730	white	6.2	0.220	0.20	20.80	0.035	58.0	184.0	1.00022	3.11
4107	white	6.8	0.300	0.26	20.30	0.037	45.0	150.0	0.99727	3.04
4480	white	5.9	0.220	0.45	22.60	0.120	55.0	122.0	0.99636	3.10

In [164…

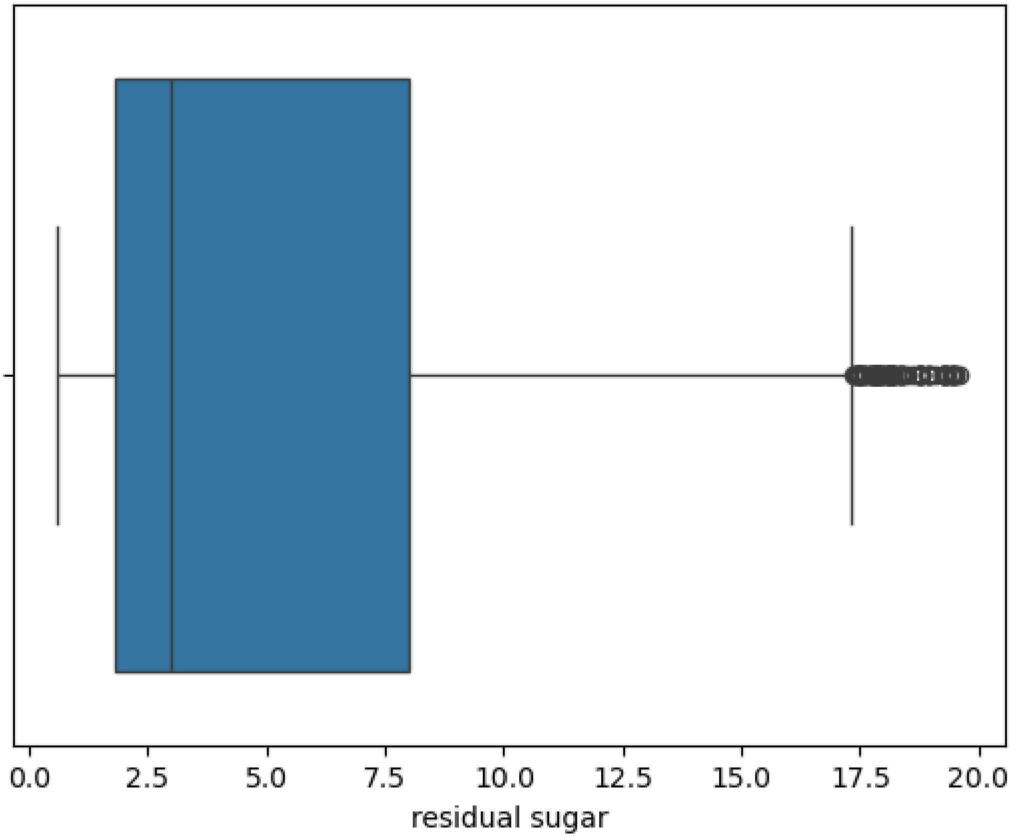
```
# Trimming - delete the outlier data
new_df = df.loc[
    (df['residual sugar'] <= upper_limit) &
    (df['residual sugar'] >= lower_limit)
]

print('Before removing outliers:', len(df))
```

```
print('After removing outliers:', len(new_df))
print('Outliers:', len(df) - len(new_df))
```

```
Before removing outliers: 6497
After removing outliers: 6469
Outliers: 28
```

In [166…
```
sns.boxplot(x=new_df['residual sugar'])
```

Out[166…
```
<Axes: xlabel='residual sugar'>
```



Inter Quartile Range Method: -

In [169…
```
q1 = df['residual sugar'].quantile(0.25)
q3 = df['residual sugar'].quantile(0.75)
iqr = q3 - q1
```

In [171…
```
q1,q3,iqr
```

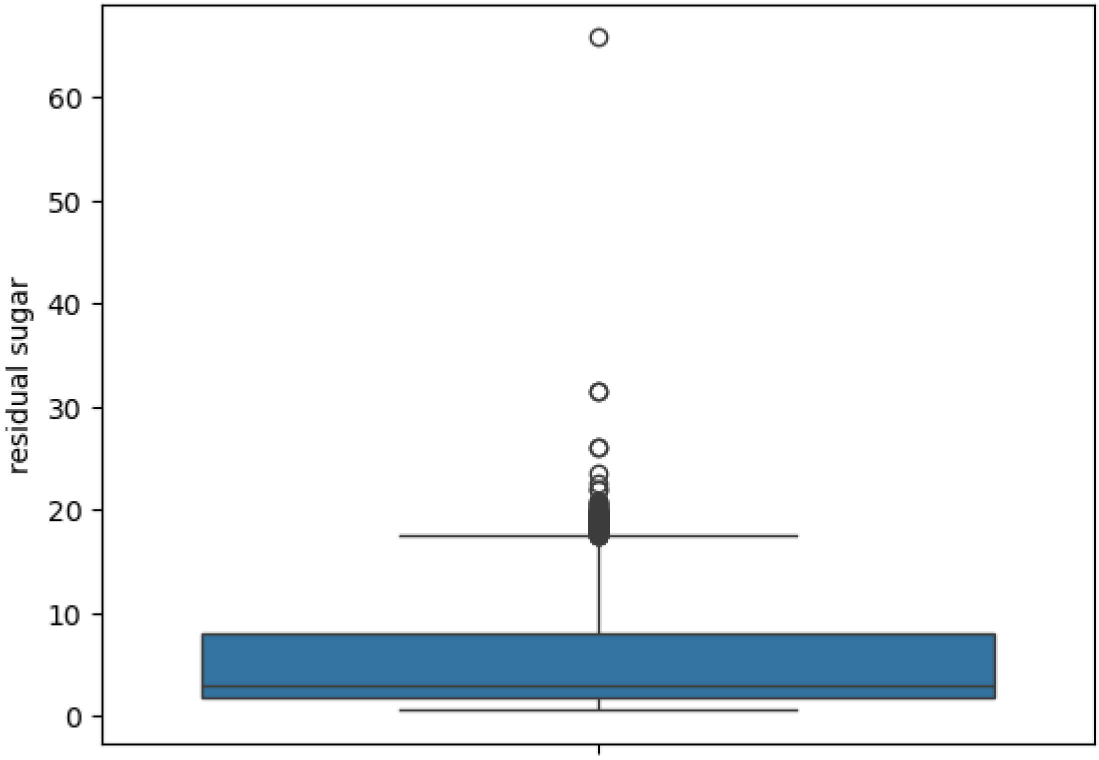
Out[171…
```
(1.8, 8.1, 6.3)
```

In [173…
```
upper_limit = q3 + (1.5 * iqr)
lower_limit = q1 - (1.5 * iqr)

lower_limit, upper_limit
```

Out[173…
```
(-7.6499999999999995, 17.549999999999997)
```

In [175…
```
sns.boxplot(df['residual sugar'])
```

Out[175…
```
<Axes: ylabel='residual sugar'>
```



In [177…

```
# Find the outliers
df.loc[
    (df['residual sugar'] > upper_limit) |
    (df['residual sugar'] < lower_limit)
]
```

Out[177…

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH
0	white	7.0	0.270	0.36	20.70	0.045	45.0	170.0	1.00100	3.00
7	white	7.0	0.270	0.36	20.70	0.045	45.0	170.0	1.00100	3.00
14	white	8.3	0.420	0.62	19.25	0.040	41.0	172.0	1.00020	2.98
38	white	7.3	0.240	0.39	17.95	0.057	45.0	149.0	0.99990	3.21
39	white	7.3	0.240	0.39	17.95	0.057	45.0	149.0	0.99990	3.21
...	...	...	...	...	...	...	...	...	...	...
4691	white	6.9	0.190	0.31	19.25	0.043	38.0	167.0	0.99954	2.93
4694	white	6.9	0.190	0.31	19.25	0.043	38.0	167.0	0.99954	2.93
4748	white	6.1	0.340	0.24	18.35	0.050	33.0	184.0	0.99943	3.12
4749	white	6.2	0.350	0.25	18.40	0.051	28.0	182.0	0.99946	3.13
4778	white	5.8	0.315	0.19	19.40	0.031	28.0	106.0	0.99704	2.97

118 rows × 13 columns

In [179…

```
# Trimming - delete the outlier data
new_df = df.loc[
```

```
    (df['residual sugar'] <= upper_limit) &
    (df['residual sugar'] >= lower_limit)
]

print('Before removing outliers:', len(df))
print('After removing outliers:', len(new_df))
print('Outliers:', len(df) - len(new_df))
```

```
Before removing outliers: 6497
After removing outliers: 6377
Outliers: 120
```

In [181…

```
sns.boxplot(x=new_df['residual sugar'])
plt.show()
```