Assignment 2

```
0801CS221114
```

0801CS221114

```
import pandas as pd
df = pd.read_csv('st.csv')
df.head()
```

	Unnamed: 0	Gender	EthnicGroup	ParentEduc	LunchType	TestPrep	MathScore	ReadingScore	WritingScore
0	0	female	group B	bachelor's degree	standard	none	72	72	74
1	1	female	group C	some college	standard	completed	69	90	88
2	2	female	group B	master's degree	standard	none	90	95	93
3	3	male	group A	associate's degree	free/reduced	none	47	57	44
4	4	male	group C	some college	standard	none	76	78	75

```
df.duplicated()
```

	0
0	False
1	False
2	False
3	False
4	False
...	...
17068	True
17337	True
23041	True
23578	True
10357	True

30651 rows × 1 columns

dtype: bool

```
df=df.drop_duplicates()
df
```

	Unnamed: 0	Gender	EthnicGroup	ParentEduc	LunchType	TestPrep	MathScore	ReadingScore	WritingScore
0	0	female	group B	bachelor's degree	standard	none	72	72	74
1	1	female	group C	some college	standard	completed	69	90	88
2	2	female	group B	master's degree	standard	none	90	95	93
3	3	male	group A	associate's degree	free/reduced	none	47	57	44
4	4	male	group C	some college	standard	none	76	78	75
...	...	...	...	...	...	...	...	...	...
30636	995	male	group C	some high school	standard	none	56	47	51
30637	996	male	group E	associate's degree	free/reduced	none	74	75	72
30638	997	male	group C	some college	standard	none	36	29	27
30639	998	male	group A	some high school	free/reduced	completed	43	34	39
30640	999	female	group D	associate's degree	standard	none	52	68	66

30641 rows × 9 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df[df.duplicated()]
```

Unnamed: 0	Gender	EthnicGroup	ParentEduc	LunchType	TestPrep	MathScore	ReadingScore	WritingScore

```
import numpy as np
from scipy import stats


import seaborn as sns
import matplotlib.pyplot as plt


z = np.abs(stats.zscore(df['ReadingScore']))
z
```

	ReadingScore
0	0.161882
1	1.388764
2	1.729565
3	0.860520
4	0.570843
...	...
30636	1.542122
30637	0.366362
30638	2.769004
30639	2.428203
30640	0.110759

30641 rows × 1 columns

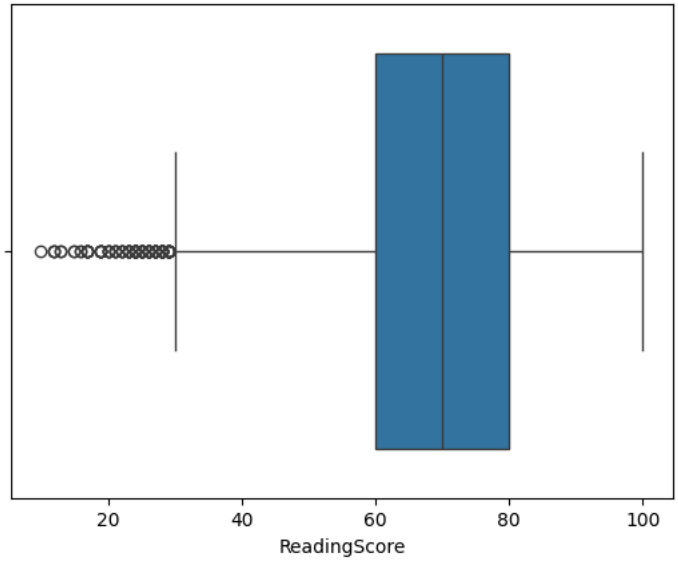
dtype: float64

```
threshold_z = 2
outlier_indices = np.where(z > threshold_z)[0]
no_outliers = df.drop(outlier_indices)
print("Original DataFrame Shape:", df.shape)
print("DataFrame Shape after Removing Outliers:", no_outliers.shape)
```

```
Original DataFrame Shape: (30641, 9)
DataFrame Shape after Removing Outliers: (29295, 9)
```

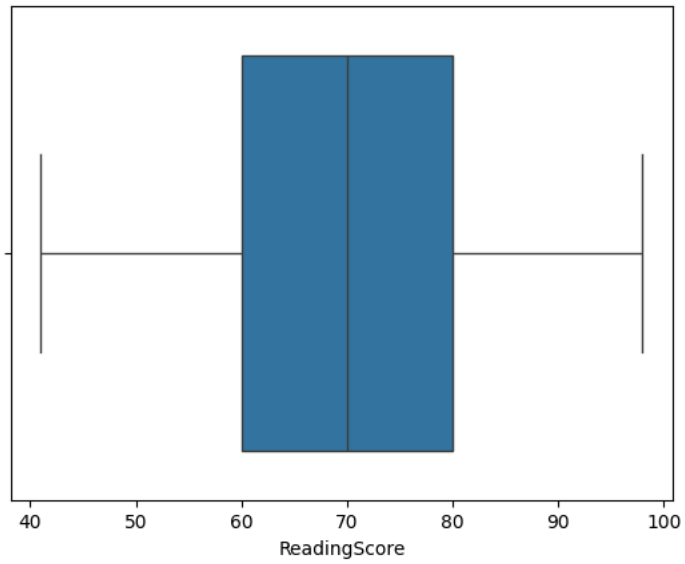
```
sns.boxplot(x=df['ReadingScore'])
```

```
<Axes: xlabel='ReadingScore'>
```




```
sns.boxplot(x=no_outliers['ReadingScore'])
```

```
<Axes: xlabel='ReadingScore'>
```




```
z = np.abs(stats.zscore(df['WritingScore']))
z
```

	WritingScore
0	0.361369
1	1.275949
2	1.602585
3	1.598447
4	0.426696
...	...
30636	1.141157
30637	0.230714
30638	2.709009
30639	1.925083
30640	0.161249

30641 rows × 1 columns

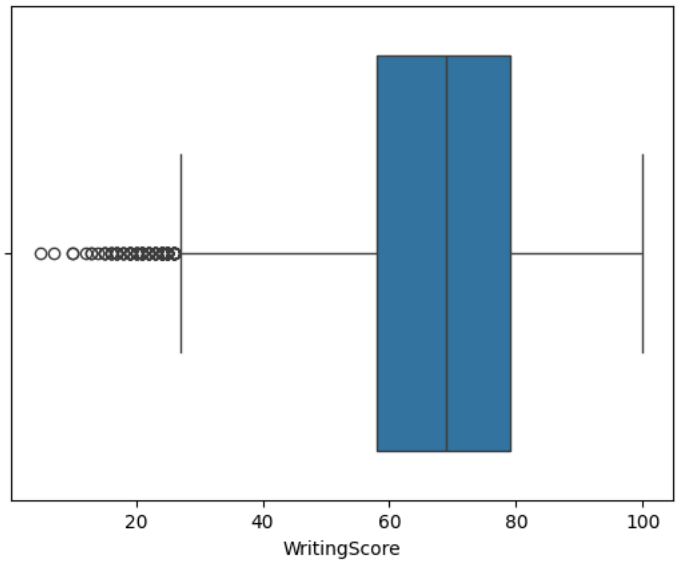
dtype: float64

```
threshold_z = 2
outlier_indices = np.where(z > threshold_z)[0]
no_outliers = df.drop(outlier_indices)
print("Original DataFrame Shape:", df.shape)
print("DataFrame Shape after Removing Outliers:", no_outliers.shape)
```

```
Original DataFrame Shape: (30641, 9)
DataFrame Shape after Removing Outliers: (29463, 9)
```

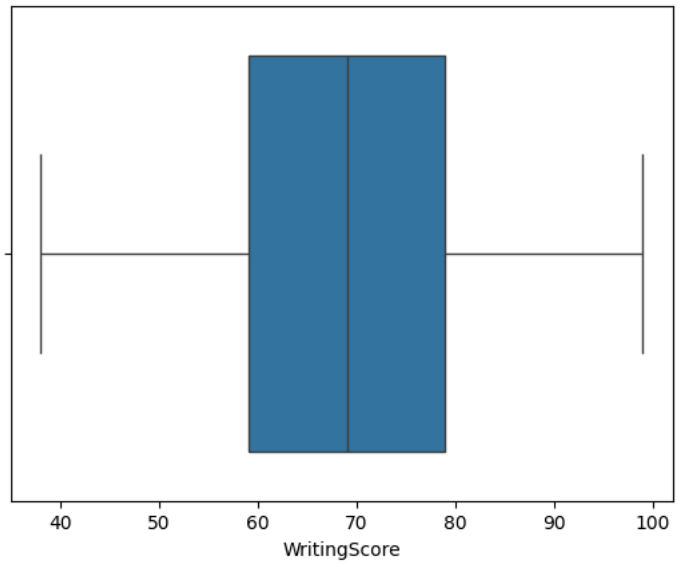
```
sns.boxplot(x=df['WritingScore'])
```

```
<Axes: xlabel='WritingScore'>
```




```
sns.boxplot(x=no_outliers['WritingScore'])
```

```
<Axes: xlabel='WritingScore'>
```



outlier using iqr

```
Q1 = df['MathScore'].quantile(0.25)
Q3 = df['MathScore'].quantile(0.75)
IQR = Q3 - Q1
IQR
```

```
22.0
```

```
lower = Q1 - 1.5*IQR

lower
```

```
23.0
```

```
upper = Q3 + 1.5*IQR
upper
```

```
111.0
```

```
upper_array = np.where(df['MathScore'] >= upper)[0]
upper_array
```

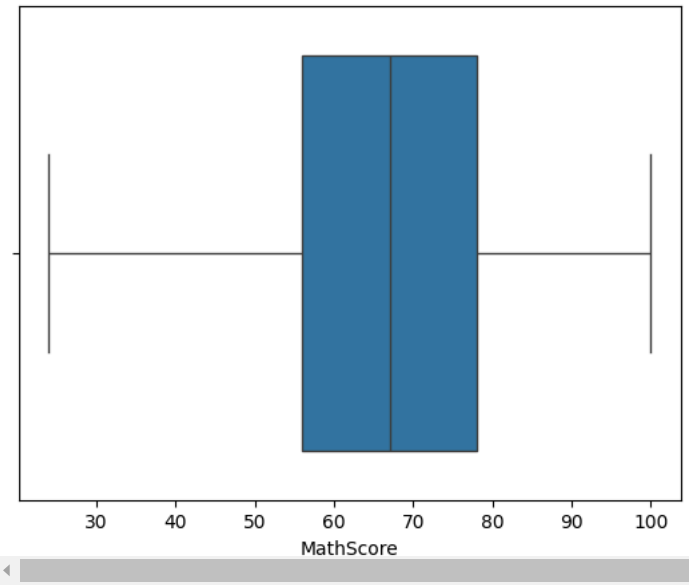
```
array([], dtype=int64)
```

```
lower_array = np.where(df['MathScore'] <= lower)[0]
```

```
df.drop(index=upper_array, inplace=True)
df.drop(index=lower_array, inplace=True)
```

```
sns.boxplot(x=df['MathScore'])
```

```
<Axes: xlabel='MathScore'>
```




```
from sklearn.preprocessing import MinMaxScaler
```

```
numerical_cols = ['MathScore', 'ReadingScore', 'WwritingScore']
scaler = MinMaxScaler()
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])
df.head()
```

	Unnamed: 0	Gender	EthnicGroup	ParentEduc	LunchType	TestPrep	MathScore	ReadingScore	WritingScore
0	0	female	group B	bachelor's degree	standard	none	0.631579	0.688889	0.726316
1	1	female	group C	some college	standard	completed	0.592105	0.888889	0.873684
2	2	female	group B	master's degree	standard	none	0.868421	0.944444	0.926316
3	3	male	group A	associate's degree	free/reduced	none	0.302632	0.522222	0.410526
4	4	male	group C	some college	standard	none	0.684211	0.755556	0.736842

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])
df.head()
```

	Unnamed: 0	Gender	EthnicGroup	ParentEduc	LunchType	TestPrep	MathScore	ReadingScore	WritingScore
0	0	female	group B	bachelor's degree	standard	none	0.340509	0.155575	0.356852
1	1	female	group C	some college	standard	completed	0.140580	1.393526	1.279421
2	2	female	group B	master's degree	standard	none	1.540087	1.737402	1.608910
3	3	male	group A	associate's degree	free/reduced	none	-1.325571	-0.876051	-1.620082
4	4	male	group C	some college	standard	none	0.607082	0.568226	0.422750

one hot encoding

```
categorical_cols = ['Gender', 'EthnicGroup', 'ParentEduc', 'LunchType', 'TestPrep']

df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_first=True)
df_encoded.head()
```

	Unnamed: 0	MathScore	ReadingScore	WritingScore	Gender_male	EthnicGroup_group B	EthnicGroup_group C	EthnicGroup_group D	EthnicGrou
0	0	0.340509	0.155575	0.356852	False	True	False	False	
1	1	0.140580	1.393526	1.279421	False	False	True	False	
2	2	1.540087	1.737402	1.608910	False	True	False	False	
3	3	-1.325571	-0.876051	-1.620082	True	False	False	False	
4	4	0.607082	0.568226	0.422750	True	False	True	False	

label encoding

```
from sklearn.preprocessing import LabelEncoder


df_encoded = df.copy()
df_encoded[categorical_cols] = df_encoded[categorical_cols].apply(LabelEncoder().fit_transform)

df_encoded.head()
```

	Unnamed: 0	Gender	EthnicGroup	ParentEduc	LunchType	TestPrep	MathScore	ReadingScore	WritingScore
0	0	0	1	1	1	1	0.340509	0.155575	0.356852
1	1	0	2	4	1	0	0.140580	1.393526	1.279421
2	2	0	1	3	1	1	1.540087	1.737402	1.608910
3	3	1	0	0	0	1	-1.325571	-0.876051	-1.620082
4	4	1	2	4	1	1	0.607082	0.568226	0.422750

new features

```
df_encoded["TotalScore"] = df_encoded["MathScore"] + df_encoded["ReadingScore"] + df_encoded["WritingScore"]
df_encoded
```

	Unnamed: 0	Gender	EthnicGroup	ParentEduc	LunchType	TestPrep	MathScore	ReadingScore	WritingScore	TotalScore
0	0	0	1	1	1	1	0.340509	0.155575	0.356852	0.852936
1	1	0	2	4	1	0	0.140580	1.393526	1.279421	2.813527
2	2	0	1	3	1	1	1.540087	1.737402	1.608910	4.886399
3	3	1	0	0	0	1	-1.325571	-0.876051	-1.620082	-3.821704
4	4	1	2	4	1	1	0.607082	0.568226	0.422750	1.598057
...	...	...	...	...	...	...	...	...	...	...
3065095	183	1	3	4	0	0	0.473796	0.980876	0.686341	2.141013
3065096	457	1	4	5	1	1	0.407152	-0.188300	-0.499820	-0.280967
3065097	247	0	1	0	1	0	0.807012	1.187201	1.213524	3.207736
3065098	790	1	1	3	1	1	1.406801	0.499451	0.488648	2.394899
3065099	404	1	0	2	1	1	1.473444	0.912101	0.752239	3.137784

3056600 rows × 10 columns

```
df_encoded["AverageScore"] = df_encoded["TotalScore"] / 3
df_encoded
```

	Unnamed: 0	Gender	EthnicGroup	ParentEduc	LunchType	TestPrep	MathScore	ReadingScore	WritingScore	TotalScore	AverageSc
0	0	0	1	1	1	1	0.340509	0.155575	0.356852	0.852936	0.284
1	1	0	2	4	1	0	0.140580	1.393526	1.279421	2.813527	0.937
2	2	0	1	3	1	1	1.540087	1.737402	1.608910	4.886399	1.628

```
def per(score):
    if score >= 1:
        return "High"
    elif score >= 0:
        return "Medium"
    else:
        return "Low"

df["performance"] = df_encoded["AverageScore"].apply(per)
df_encoded['performance']
```

	performance
0	Medium
1	Medium
2	High
3	Low
4	Medium
...	...
3065095	Medium
3065096	Low
3065097	High
3065098	Medium
3065099	High

3056600 rows × 1 columns