

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [3]: dataset=pd.read_csv("Flyzy Flight Cancellation - Sheet1.csv")
```

```
In [4]: dataset.shape
```

Out[4]: (3000, 14)

```
In [5]: dataset.head()
```

Out[5]:

	Flight ID	Airline	Flight_Distance	Origin_Airport	Destination_Airport	Scheduled_Departure_Time
0	7319483	Airline D	475	Airport 3	Airport 2	4
1	4791965	Airline E	538	Airport 5	Airport 4	12
2	2991718	Airline C	565	Airport 1	Airport 2	17
3	4220106	Airline E	658	Airport 5	Airport 3	1
4	2263008	Airline E	566	Airport 2	Airport 2	19



```
In [6]: dataset.describe()
```

Out[6]:

	Flight ID	Flight_Distance	Scheduled_Departure_Time	Day_of_Week	Month	Weat
count	3.000000e+03	3000.000000	3000.000000	3000.000000	3000.000000	30
mean	4.997429e+06	498.909333	11.435000	3.963000	6.381000	
std	2.868139e+06	98.892266	6.899298	2.016346	3.473979	
min	3.681000e+03	138.000000	0.000000	1.000000	1.000000	
25%	2.520313e+06	431.000000	6.000000	2.000000	3.000000	
50%	5.073096e+06	497.000000	12.000000	4.000000	6.000000	
75%	7.462026e+06	566.000000	17.000000	6.000000	9.000000	
max	9.999011e+06	864.000000	23.000000	7.000000	12.000000	



```
In [7]: dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 14 columns):
 #   Column                                Non-Null Count  Dtype
---  -
 0   Flight ID                            3000 non-null   int64
 1   Airline                             3000 non-null   object
 2   Flight_Distance                      3000 non-null   int64
 3   Origin_Airport                      3000 non-null   object
 4   Destination_Airport                 3000 non-null   object
 5   Scheduled_Departure_Time            3000 non-null   int64
 6   Day_of_Week                        3000 non-null   int64
 7   Month                              3000 non-null   int64
 8   Airplane_Type                      3000 non-null   object
 9   Weather_Score                      3000 non-null   float64
10   Previous_Flight_Delay_Minutes       3000 non-null   float64
11   Airline_Rating                     3000 non-null   float64
12   Passenger_Load                     3000 non-null   float64
13   Flight_Cancelled                   3000 non-null   int64
dtypes: float64(4), int64(6), object(4)
memory usage: 328.3+ KB
```

```
In [8]: dataset.dtypes
```

```
Out[8]: Flight ID                int64
Airline                object
Flight_Distance        int64
Origin_Airport         object
Destination_Airport    object
Scheduled_Departure_Time int64
Day_of_Week            int64
Month                 int64
Airplane_Type          object
Weather_Score          float64
Previous_Flight_Delay_Minutes float64
Airline_Rating         float64
Passenger_Load         float64
Flight_Cancelled       int64
dtype: object
```

```
In [9]: dataset.isnull().sum()
```

```
Out[9]: Flight ID                0
Airline                0
Flight_Distance        0
Origin_Airport         0
Destination_Airport    0
Scheduled_Departure_Time 0
Day_of_Week            0
Month                 0
Airplane_Type          0
Weather_Score          0
Previous_Flight_Delay_Minutes 0
Airline_Rating         0
Passenger_Load         0
Flight_Cancelled       0
dtype: int64
```

```
In [10]: dataset.columns
```

```
Out[10]: Index(['Flight ID', 'Airline', 'Flight_Distance', 'Origin_Airport',  
            'Destination_Airport', 'Scheduled_Departure_Time', 'Day_of_Week',  
            'Month', 'Airplane_Type', 'Weather_Score',  
            'Previous_Flight_Delay_Minutes', 'Airline_Rating', 'Passenger_Load',  
            'Flight_Cancelled'],  
          dtype='object')
```

```
In [15]: type(dataset)
```

```
Out[15]: pandas.core.frame.DataFrame
```

```
In [20]: numeric_columns=dataset.select_dtypes(include=['float64', 'int64']).columns  
        numeric_columns
```

```
Out[20]: Index(['Flight ID', 'Flight_Distance', 'Scheduled_Departure_Time',  
            'Day_of_Week', 'Month', 'Weather_Score',  
            'Previous_Flight_Delay_Minutes', 'Airline_Rating', 'Passenger_Load',  
            'Flight_Cancelled'],  
          dtype='object')
```

```
In [21]: numeric_columns=dataset.select_dtypes(include=['object', 'object']).columns  
        numeric_columns
```

```
Out[21]: Index(['Airline', 'Origin_Airport', 'Destination_Airport', 'Airplane_Type'], dtype  
            ='object')
```

```
In [22]: from sklearn.preprocessing import LabelEncoder
```

```
In [23]: label_encoder=LabelEncoder()
```

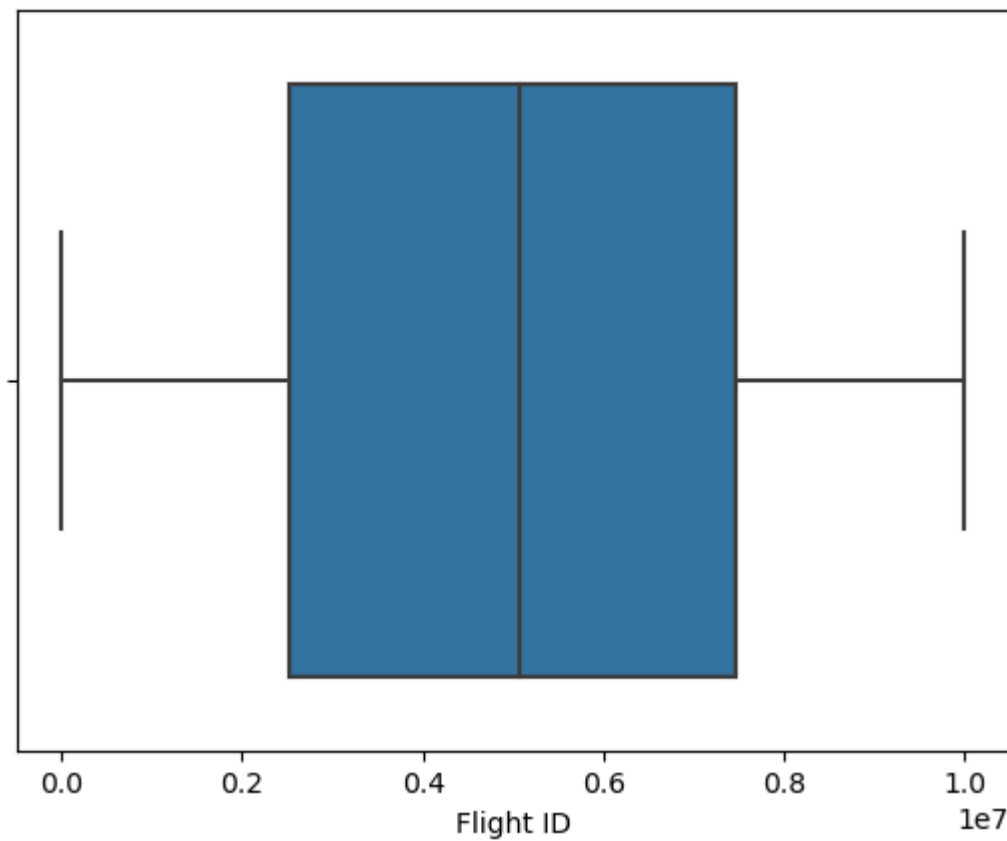
```
In [24]: dataset['Airline']=label_encoder.fit_transform(dataset['Airline'])  
        dataset['Origin_Airport']=label_encoder.fit_transform(dataset['Origin_Airport'])  
        dataset['Destination_Airport']=label_encoder.fit_transform(dataset['Destination_Airport'])  
        dataset['Airplane_Type']=label_encoder.fit_transform(dataset['Airplane_Type'])
```

```
In [25]: dataset.dtypes
```

```
Out[25]: Flight ID                int64  
        Airline                  int32  
        Flight_Distance          int64  
        Origin_Airport           int32  
        Destination_Airport      int32  
        Scheduled_Departure_Time  int64  
        Day_of_Week              int64  
        Month                    int64  
        Airplane_Type            int32  
        Weather_Score            float64  
        Previous_Flight_Delay_Minutes float64  
        Airline_Rating           float64  
        Passenger_Load           float64  
        Flight_Cancelled         int64  
        dtype: object
```

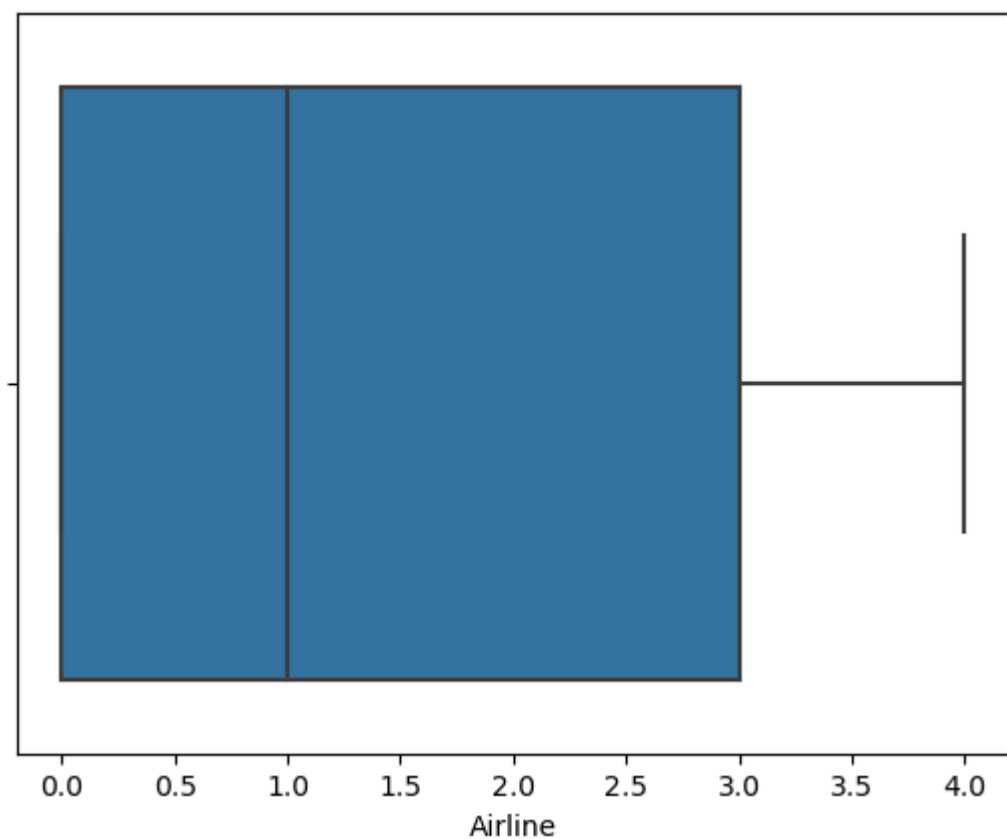
```
In [26]: sns.boxplot(x=dataset['Flight ID'])
```

```
Out[26]: <Axes: xlabel='Flight ID'>
```



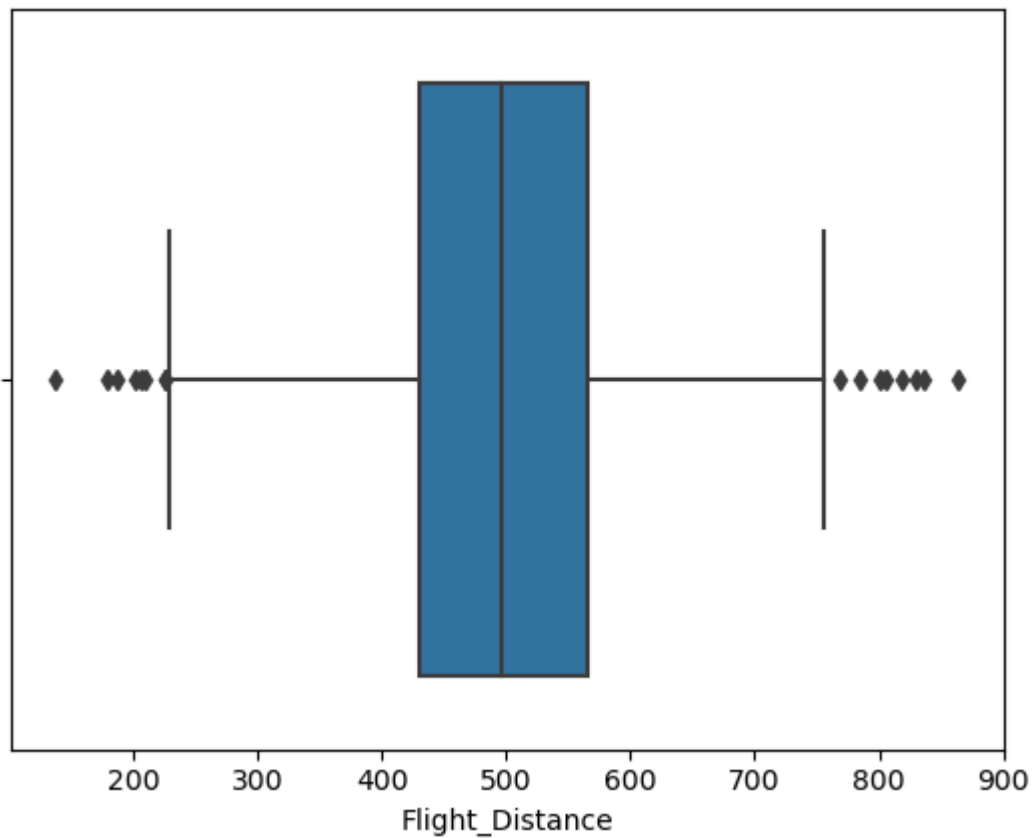
```
In [27]: sns.boxplot(x=dataset['Airline'])
```

```
Out[27]: <Axes: xlabel='Airline'>
```



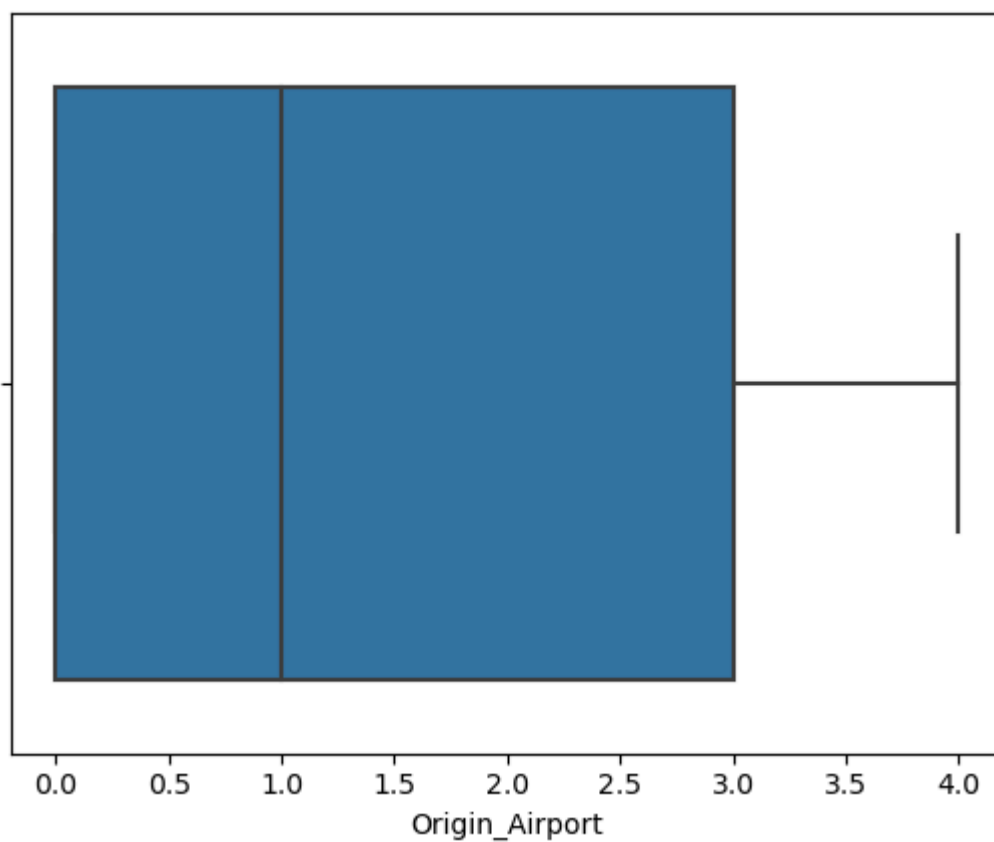
```
In [28]: sns.boxplot(x=dataset['Flight_Distance'])
```

```
Out[28]: <Axes: xlabel='Flight_Distance'>
```



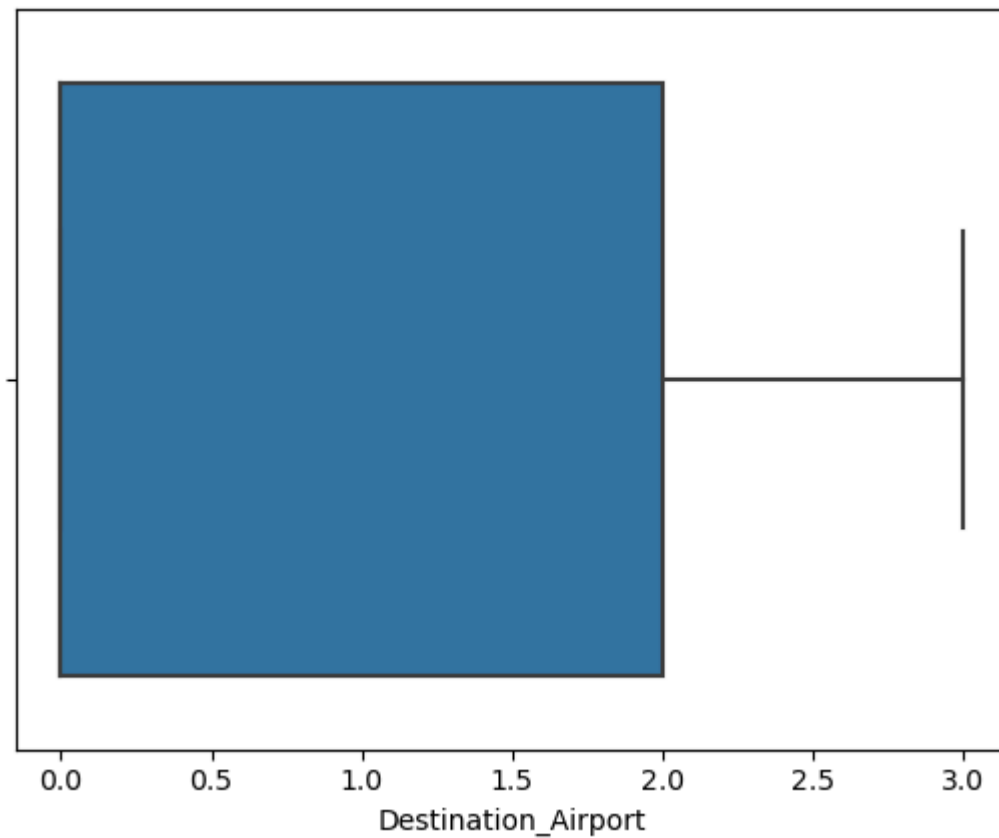
```
In [29]: sns.boxplot(x=dataset['Origin_Airport'])
```

```
Out[29]: <Axes: xlabel='Origin_Airport'>
```



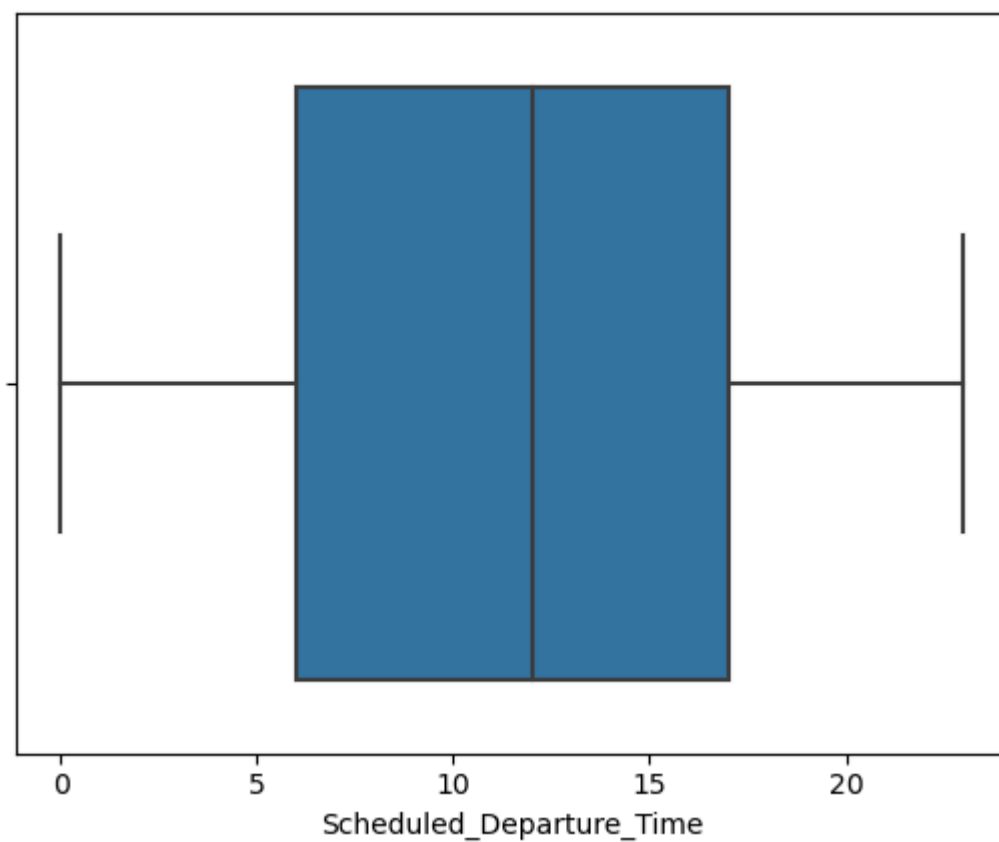
```
In [30]: sns.boxplot(x=dataset['Destination_Airport'])
```

```
Out[30]: <Axes: xlabel='Destination_Airport'>
```



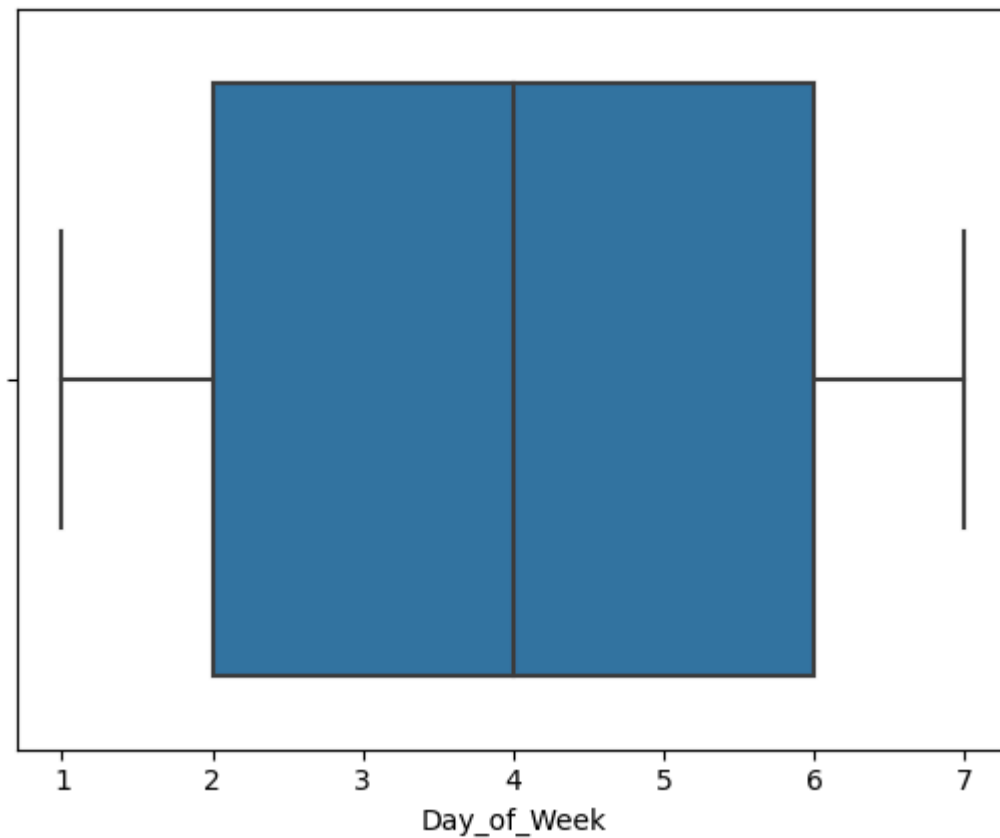
```
In [31]: sns.boxplot(x=dataset['Scheduled_Departure_Time'])
```

```
Out[31]: <Axes: xlabel='Scheduled_Departure_Time'>
```



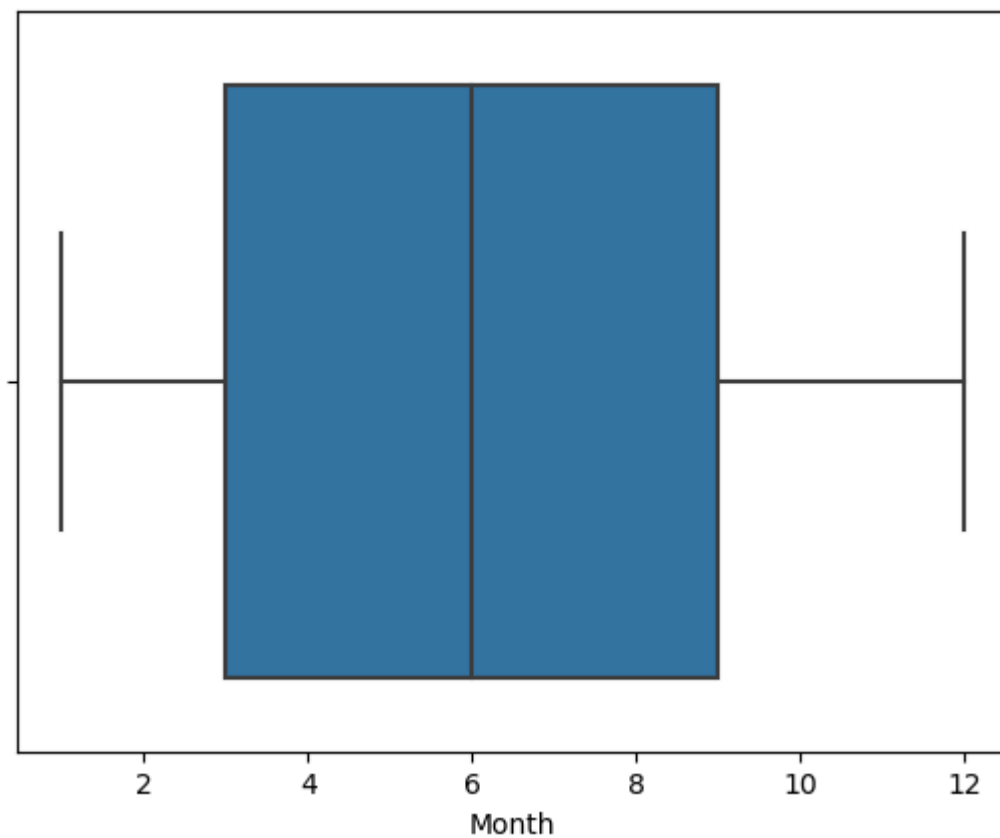
```
In [32]: sns.boxplot(x=dataset['Day_of_Week'])
```

```
Out[32]: <Axes: xlabel='Day_of_Week'>
```



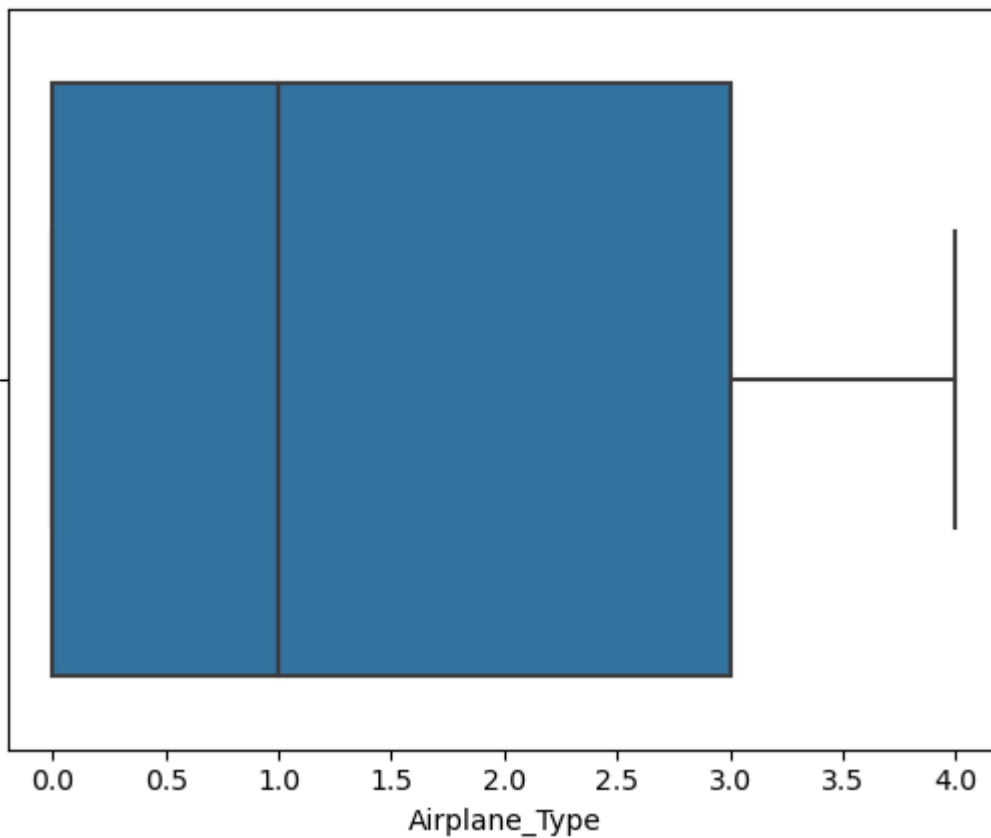
```
In [33]: sns.boxplot(x=dataset['Month'])
```

```
Out[33]: <Axes: xlabel='Month'>
```



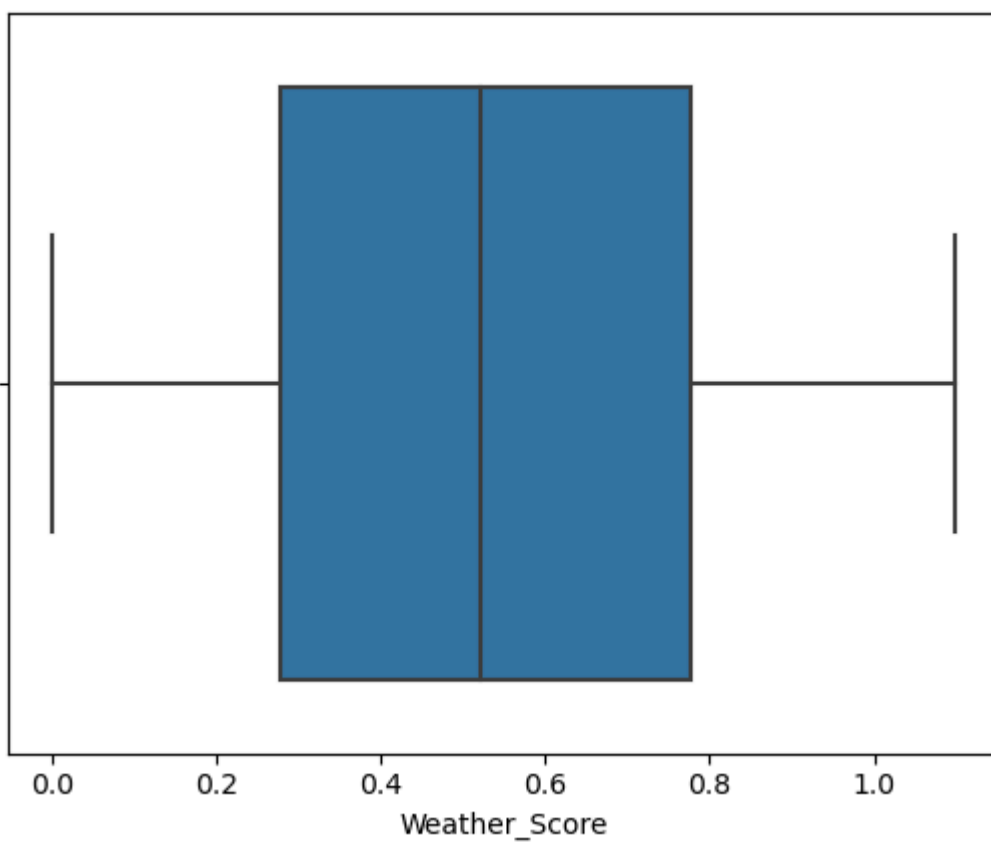
```
In [34]: sns.boxplot(x=dataset['Airplane_Type'])
```

```
Out[34]: <Axes: xlabel='Airplane_Type'>
```



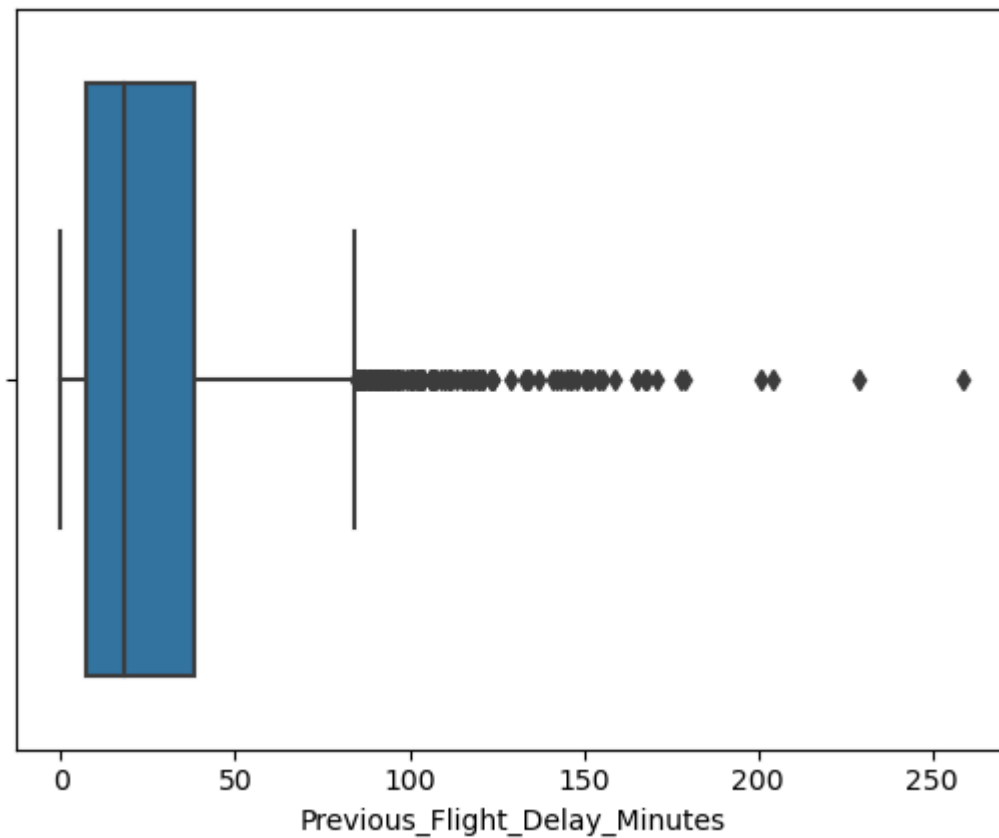
```
In [35]: sns.boxplot(x=dataset['Weather_Score'])
```

```
Out[35]: <Axes: xlabel='Weather_Score'>
```



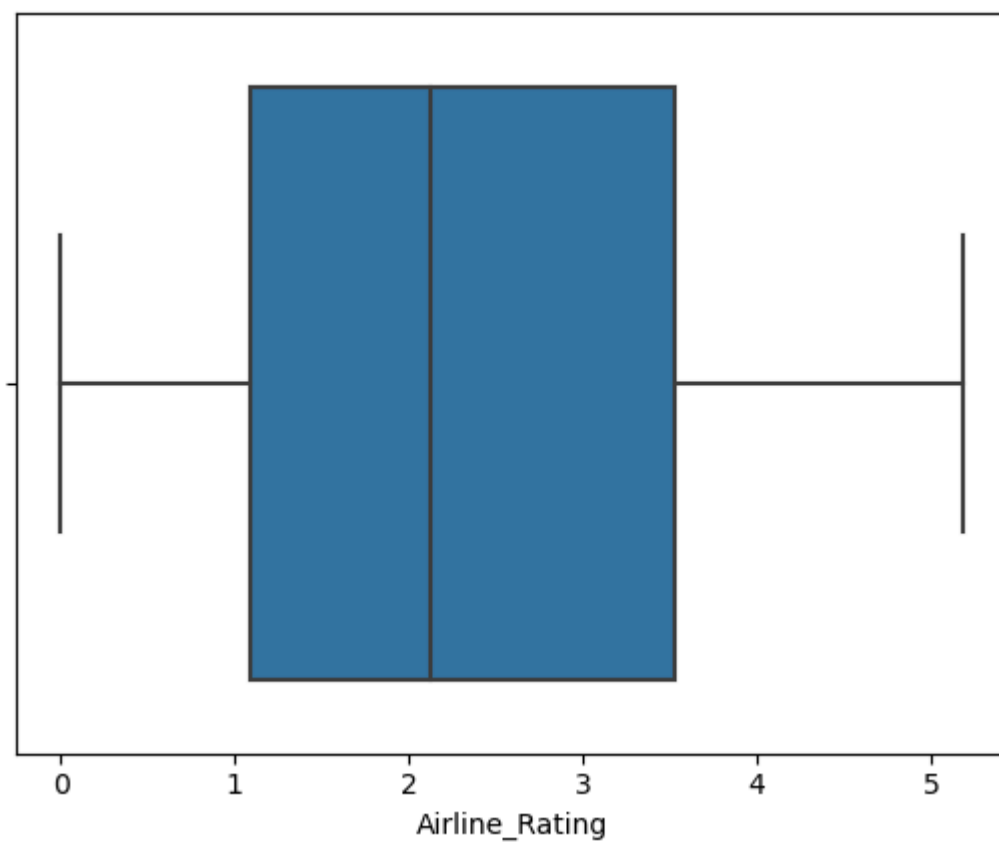
```
In [36]: sns.boxplot(x=dataset['Previous_Flight_Delay_Minutes'])
```

```
Out[36]: <Axes: xlabel='Previous_Flight_Delay_Minutes'>
```

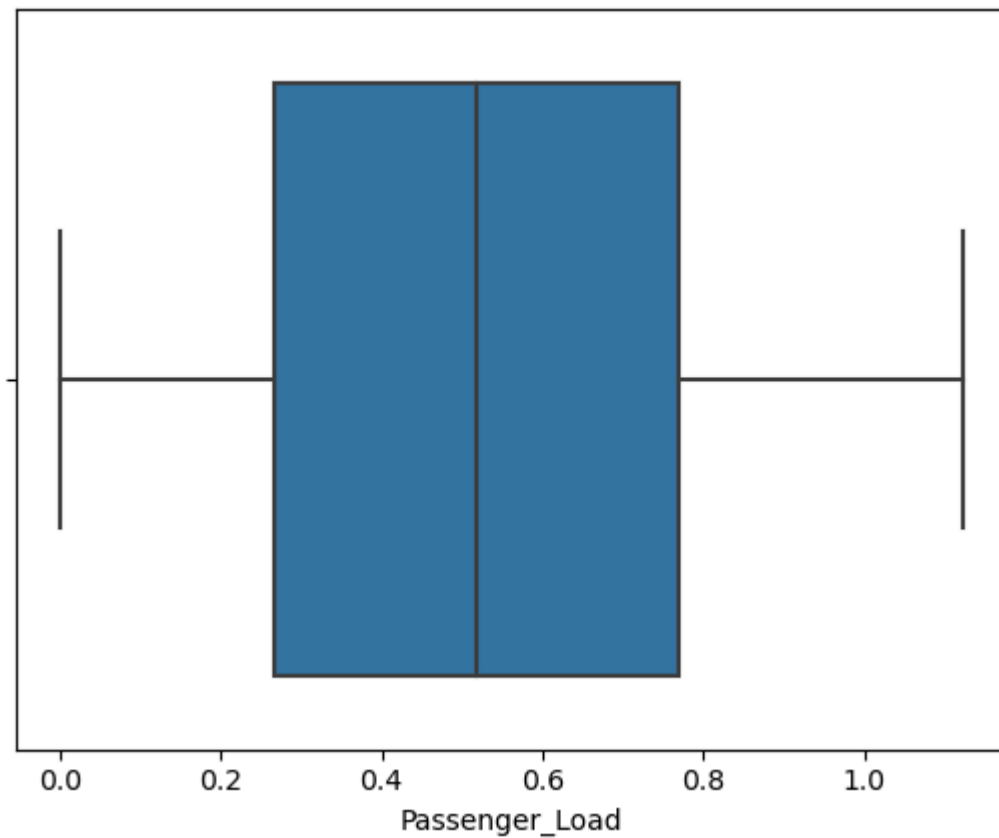
```
In [37]: sns.boxplot(x=dataset['Airline_Rating'])
```

```
Out[37]: <Axes: xlabel='Airline_Rating'>
```



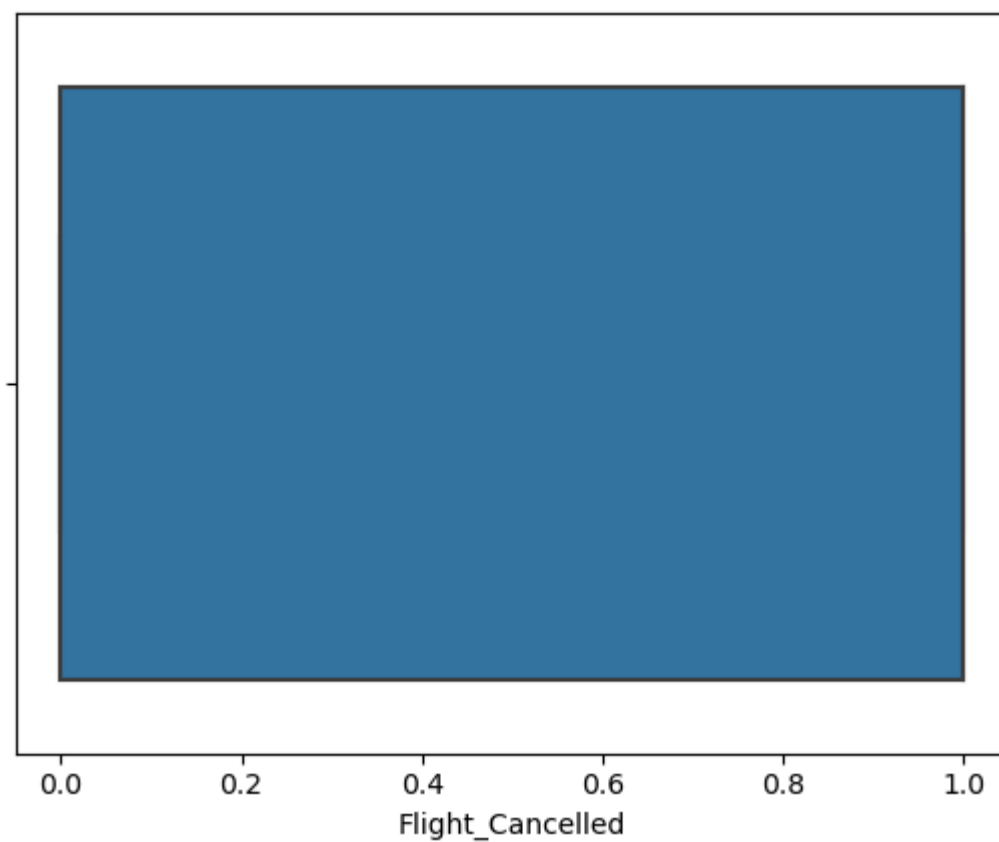
```
In [38]: sns.boxplot(x=dataset['Passenger_Load'])
```

```
Out[38]: <Axes: xlabel='Passenger_Load'>
```



```
In [39]: sns.boxplot(x=dataset['Flight_Cancelled'])
```

```
Out[39]: <Axes: xlabel='Flight_Cancelled'>
```



```
In [42]: def detect_outliers(dataset):  
    outliers = []  
    threshold = 3  
    mean = np.mean(dataset)  
    std = np.std(dataset)
```

```

for value in data:
    z_score = (value - mean) / std
    if np.abs(z_score) > threshold:
        outliers.append(value)
    return outliers

```

```

In [43]: from scipy.stats.mstats import winsorize
import numpy as np

```

```

In [44]: dataset=np.random.normal(loc=0, scale=1, size=100)

```

```

In [47]: dataset_winsorized=winsorize(dataset, limits=[0.05, 0.05])
print(dataset_winsorized)

```

```

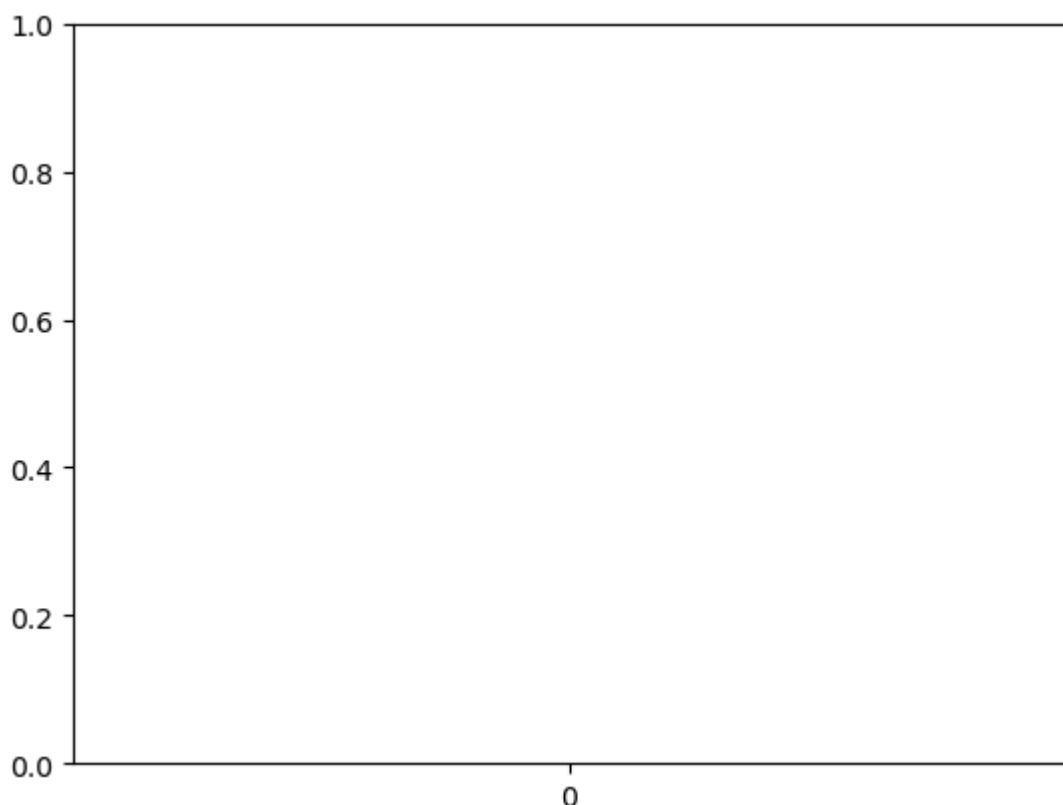
[-0.64965234  0.25563499  0.18341794 -1.58128981  1.65804405 -1.30616738
-0.05996652 -0.41772447 -0.14846198  0.07988675 -1.40303167  1.65804405
-0.48775732 -0.23932739  1.1720029   1.45197459 -0.829427   0.55233677
 0.86807595  1.19937002  0.78821331  0.53706415 -0.02147413  0.3739901
-0.4891361  -1.19349522 -0.07508567  1.65804405 -0.25027876 -1.58013726
 0.39634842 -0.1801127  -0.4419832   0.84428345  0.46998497 -1.03515374
-0.56152248 -0.73826327  0.20186136  1.65804405 -1.09670596  0.46055817
-1.27122344 -0.36201104 -0.43284583  1.65804405 -1.31238608 -0.48293471
-0.16023945 -0.70322171 -0.07881647 -0.18894878 -0.26166639  0.2839005
 1.01395192 -0.82351454 -1.49335014 -0.11829412 -0.66687738 -1.58128981
-0.65725689  0.54252294  0.48814617 -1.58128981  1.27082612 -1.58128981
-0.16003668  0.71797108  0.9492323  -1.05712916 -0.32273816 -0.89456671
-1.06976138  0.08854586 -0.38755384 -0.513144   0.71523408 -1.10819748
 0.60906254 -1.58128981  0.29483511 -0.02213082 -0.43037894  1.48309072
 1.21692373  0.6211766  -0.22740329  0.00834157 -1.21546636  1.65804405
 0.3844207  -0.37414805 -1.58128981  0.76744058 -0.11987383 -0.11998199
-0.23562321  0.13360845  0.70463811 -0.06581468]

```

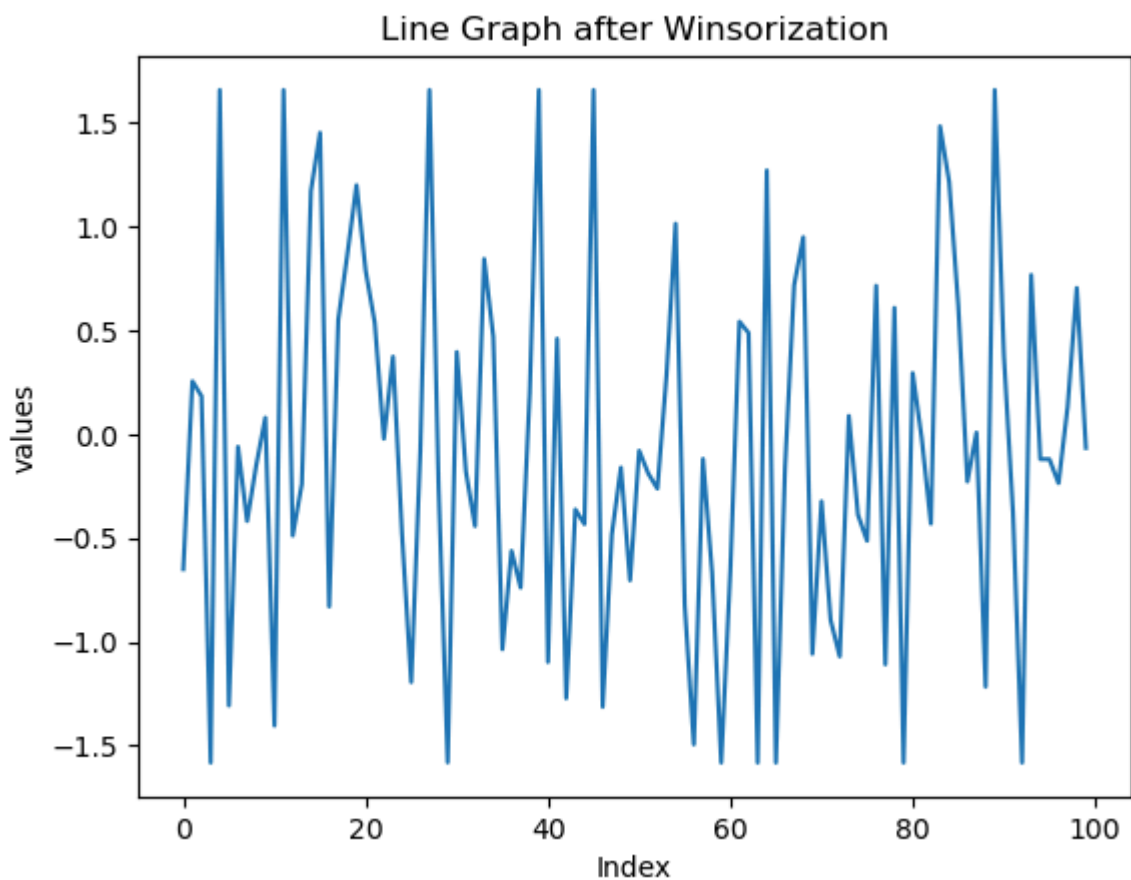
```

In [49]: sns.boxplot(dataset=dataset)
plt.show()

```

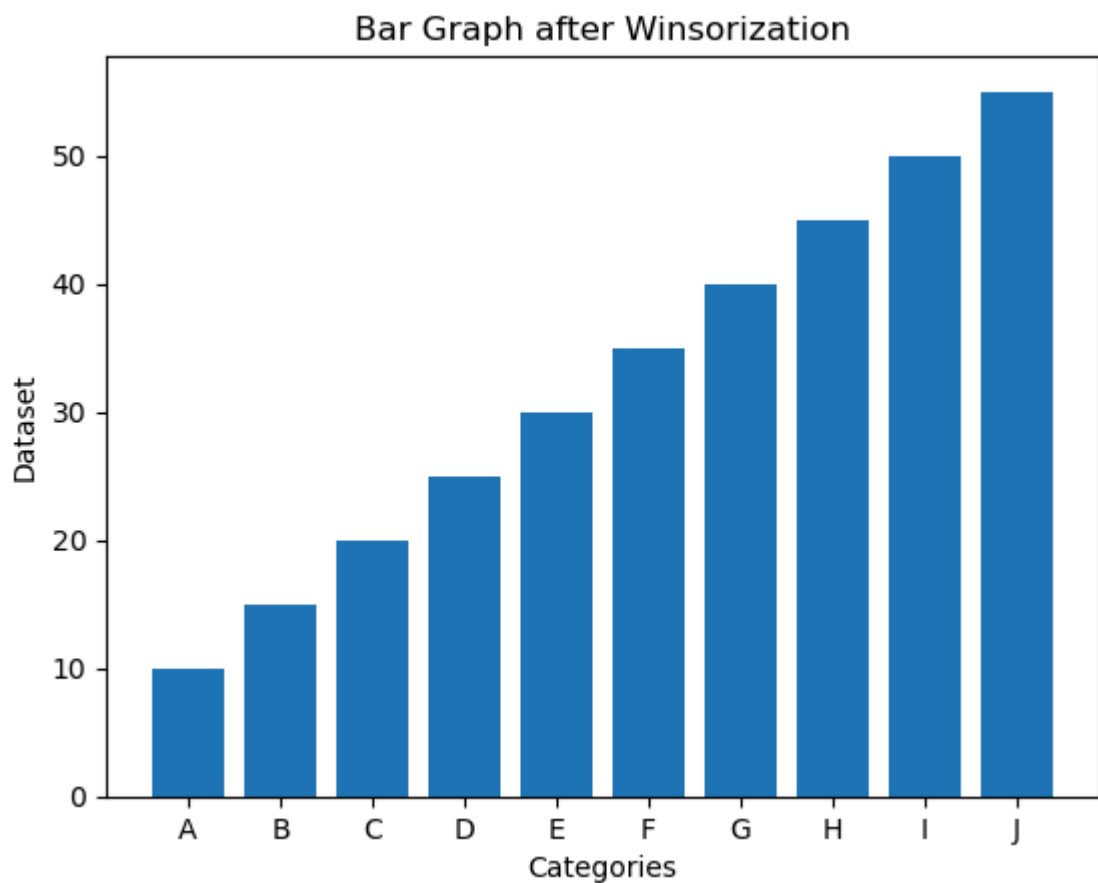


```
In [50]: plt.plot(dataset_winsorized)
plt.xlabel('Index')
plt.ylabel('values')
plt.title('Line Graph after Winsorization')
plt.show()
```



```
In [54]: dataset= [10, 15, 20, 25, 30, 35, 40, 45, 50, 55]
categories= ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']

plt.bar(categories, dataset)
plt.xlabel('Categories')
plt.ylabel('Dataset')
plt.title('Bar Graph after Winsorization')
plt.show()
```



```
In [58]: type(dataset)
```

```
Out[58]: list
```

```
In [62]: data=[dataset]
summary={i: dataset.count(i) for i in dataset}
print(summary)

{10: 1, 15: 1, 20: 1, 25: 1, 30: 1, 35: 1, 40: 1, 45: 1, 50: 1, 55: 1}
```

```
In [70]: dataset=pd.DataFrame(dataset)
print(dataset.dtypes)
```

```
0    int64
dtype: object
```

```
In [66]:
```

```
In [ ]:
```