In [23]:
```
#importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [24]:
```
#loading data
df = pd.read_csv("Flyzy Flight Cancellation.csv")
```

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

Out[5]:

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

In [6]:
```
df.tail()
```

Out[6]:

	Flight ID	Airline	Flight_Distance	Origin_Airport	Destination_Airport	Scheduled_I
2995	1265781	Airline D	395	Airport 2	Airport 3	
2996	5440150	Airline E	547	Airport 1	Airport 4	
2997	779080	Airline C	461	Airport 1	Airport 3	
2998	4044431	Airline B	464	Airport 3	Airport 3	
2999	2806578	Airline A	369	Airport 1	Airport 2	

In [7]:
```
df.shape
```

Out[7]:
```
(3000, 14)
```

In [8]:
```
df.columns
```

Out[8]:
```
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')
```

Observation: Most of the column names consist of multiple words seperated by underscores,but 'Flight ID' does not follow this format, therefore we need to change it to keep consistancy.

In [9]:
```
#Changing column name
df.rename(columns={'Flight ID' : 'Flight_ID'}, inplace =True)
```

In [10]:
```
df.head(2)
```

Out[10]:

	Flight_ID	Airline	Flight_Distance	Origin_Airport	Destination_Airport	Scheduled_Dep
0	7319483	Airline D	475	Airport 3	Airport 2	
1	4791965	Airline E	538	Airport 5	Airport 4	

CHECKING DATA TYPES OF EACH COLUMN

In [11]:
```
df.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
```

Observation: This results indicate that all columns have the correct data types according to the data they contain

In [12]:
```
df.describe()
```

Out[12]:

	Flight_ID	Flight_Distance	Scheduled_Departure_Time	Day_of_Week	Mo
count	3.000000e+03	3000.000000	3000.000000	3000.000000	3000.000
mean	4.997429e+06	498.909333	11.435000	3.963000	6.381
std	2.868139e+06	98.892266	6.899298	2.016346	3.473
min	3.681000e+03	138.000000	0.000000	1.000000	1.000
25%	2.520313e+06	431.000000	6.000000	2.000000	3.000
50%	5.073096e+06	497.000000	12.000000	4.000000	6.000
75%	7.462026e+06	566.000000	17.000000	6.000000	9.000
max	9.999011e+06	864.000000	23.000000	7.000000	12.000

In [13]:
```
#Checking for duplicates entries
duplicates = df[df.duplicated()]
```

In [16]:
```
duplicates
```

Out[16]:

Flight_ID	Airline	Flight_Distance	Origin_Airport	Destination_Airport	Scheduled_Depa

No duplicates on the dataset

CHECKING FOR MISSING VALUES

In [18]:
```
df.isnull().sum()
```

Out[18]:
```
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
```

There are no missing values

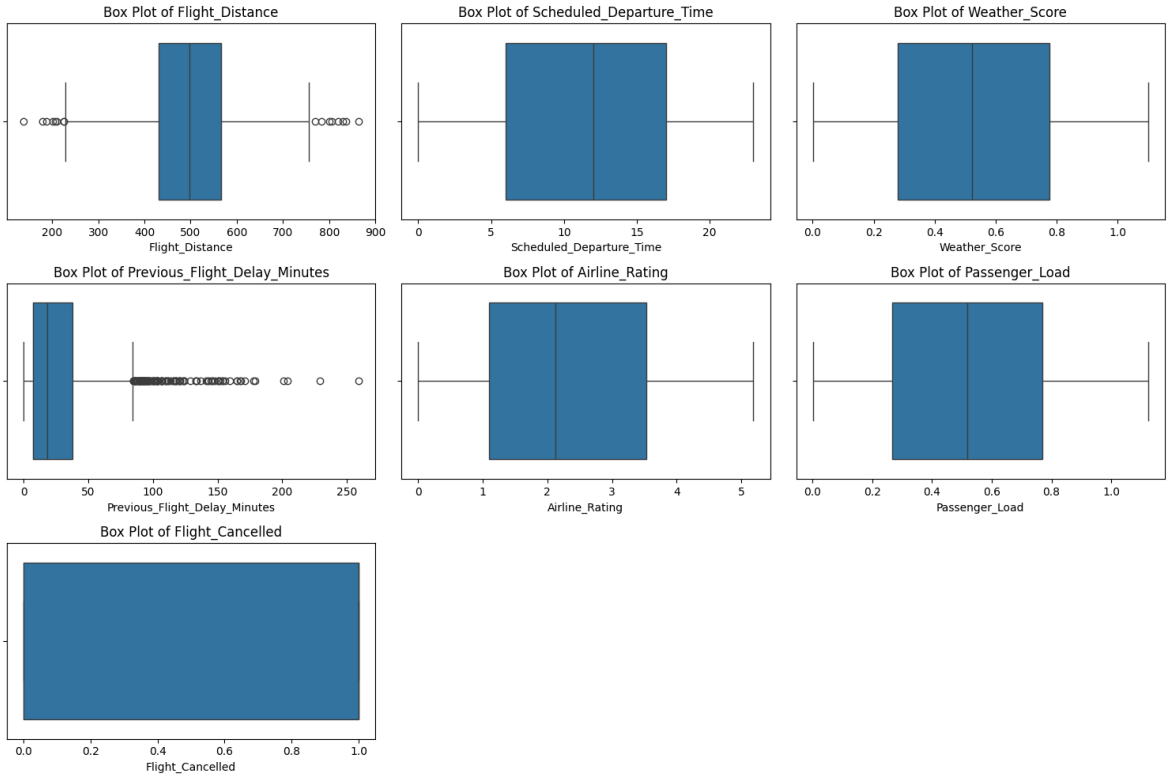
CHECKING FOR OUTLIERS

In [22]:
```
#Used boxplot to visually check outliers
```

In [20]:
```
columns_to_check =['Flight_Distance',
          'Scheduled_Departure_Time',
```

```
            'Weather_Score',
            'Previous_Flight_Delay_Minutes',
            'Airline_Rating', 'Passenger_Load',
            'Flight_Cancelled'
           ]
```

In [21]:
```
plt.figure(figsize =(15,10))
for i, col in enumerate(columns_to_check, 1):
    plt.subplot(3,3, i)
    sns.boxplot(x=df[col])
    plt.title(f'Box Plot of {col}')
plt.tight_layout()
plt.show()
```



This plots shows that the following columns have outliers and have to be handled

1. Flight_Distance
2. Previous_Flight_Delay_Minutes

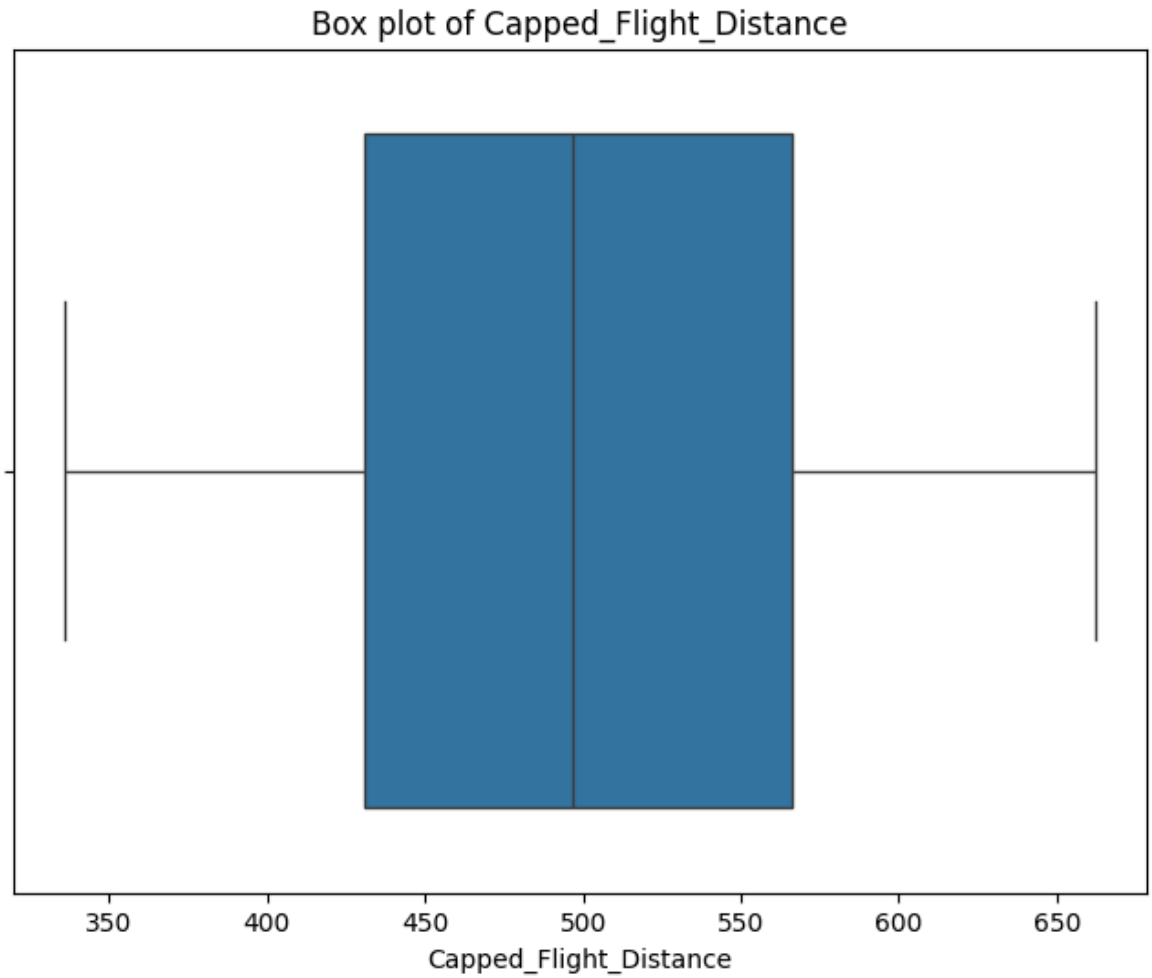
1. Handling outliers for Flight_Distance column using Capping method Because it reduces the impact of extreme outliers, which can distort the analysis.

In [30]:
```
#Handling outliers on Flight Distance using Capping approach

#applying threshold
cap_max = df['Flight_Distance'].quantile(0.95)
cap_min =df['Flight_Distance'].quantile(0.05)
#Apply capping
df['Capped_Flight_Distance'] = np.clip(df['Flight_Distance'], cap_min, cap_max)
```

In [31]:
```
#Plotting transformed column
plt.figure(figsize =(8,6))
sns.boxplot(x=df['Capped_Flight_Distance'])
```

```
plt.title('Box plot of Capped_Flight_Distance')
plt.show()
```

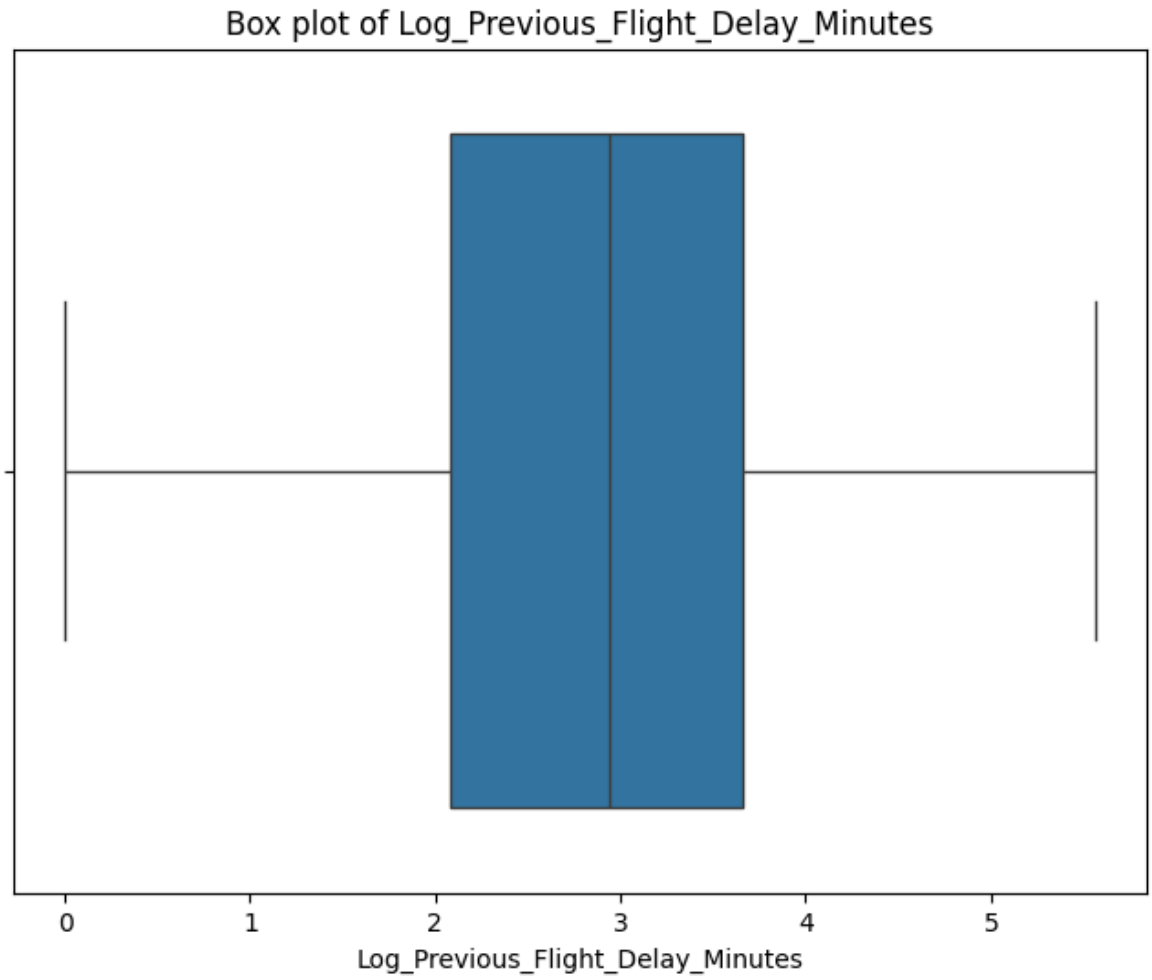


The results shows no more outliers for Flight_Distance

2. Handling Outliers for Previous_Flight_Delay_Minutes Using Log Transformation because data is skewed, compressing the range of delay times, reducing the impact of extreme values.

In [25]:
```
#creating a new column and applying the log
df['Log_Previous_Flight_Delay_Minutes'] = np.log1p(df['Previous_Flight_Delay_Min
```

In [26]:
```
#Plotting transformed column
plt.figure(figsize =(8,6))
sns.boxplot(x=df['Log_Previous_Flight_Delay_Minutes'])
plt.title('Box plot of Log_Previous_Flight_Delay_Minutes')
plt.show()
```



Now the outliers were handled and not showing on the plot

In [32]:
```
df.head(2)
```

Out[32]:

	Flight ID	Airline	Flight_Distance	Origin_Airport	Destination_Airport	Scheduled_Dep
0	7319483	Airline D	475	Airport 3	Airport 2	
1	4791965	Airline E	538	Airport 5	Airport 4	

In [ ]: