PART A

In [5]:
```
import pandas as pd
import numpy as np

# Read the Excel .xlsx file using openpyxl
data = pd.read_excel("C:/Users/AKSHATA/Downloads/prescription_fraud_detection_dataset.xlsx", engine='openpyxl
data
```

Out[5]:

	Prescription_ID	Patient_ID	Doctor_ID	Pharmacy_ID	Drug_Name	Drug_Category	Dosage	Prescription_Date	Refill_Count	Insuranc
0	1	9107	103	678	Insulin	Antibiotic	98mg	2024-06-07	4	
1	2	3997	917	253	Hydrocodone	Antibiotic	26mg	2024-02-04	1	
2	3	7561	896	623	Metformin	Hypertension	471mg	2024-03-15	0	
3	4	1509	704	998	Lisinopril	Hypertension	342mg	2024-02-06	5	
4	5	1804	853	551	Oxycodone	Antibiotic	180mg	2024-01-11	5	
...	...	...	...	...	...	...	...	...	...	
495	496	7360	698	977	Insulin	Hypertension	38mg	2024-01-06	4	
496	497	1626	715	815	Lisinopril	Diabetes	162mg	2024-01-05	4	
497	498	7093	439	303	Oxycodone	Antibiotic	115mg	2024-10-22	3	
498	499	1101	751	435	Oxycodone	Hypertension	195mg	2024-03-04	2	

```
# describing columns

Prescription_ID: Unique identifier for each prescription.
Patient_ID: Unique identifier for each patient.
Doctor_ID: Unique identifier for each prescribing doctor.
Pharmacy_ID: Unique identifier for the pharmacy dispensing the prescription.
Drug_Name: Name of the prescribed drug.
Drug_Category: The category or type of the drug (e.g., opioid, antibiotic).
Dosage: The prescribed dosage of the medication (e.g., mg).
Prescription_Date: Date when the prescription was issued.
Refill_Count: Number of times the prescription has been refilled.
Insurance_Claim: Binary indicator (0 or 1) of whether the prescription was claimed through insurance.
Total_Cost: Total cost of the prescribed medication.
Doctor_Specialty: Medical specialty of the prescribing doctor (e.g., cardiologist, general practitioner).
Patient_Age: Age of the patient at the time of prescription.
Patient_Gender: Gender of the patient (e.g., Male, Female, Other).
Patient_Health_Condition: Health condition of the patient related to the prescription (e.g., diabetes,
hypertension).
Prescription_Status: Current status of the prescription (e.g., filled, pending, canceled).
Drug_Schedule: Classification of the drug based on potential for abuse (e.g., Schedule II, III).
Pharmacy_Location: Location or city of the pharmacy.
Prescription_Filled_Date: Date when the prescription was filled by the pharmacy.
Days_Supply: Number of days the medication supply will last.
Fraudulent_Claim: Binary indicator (0 or 1) denoting whether the prescription is flagged as fraudulent.
Patient_Region: Geographic region where the patient resides (e.g., North, South).
Doctor_Region: Geographic region where the doctor practices.
Pharmacy_Region: Geographic region where the pharmacy is located.
Doctor_Flagged: Binary indicator (0 or 1) showing if the doctor has been flagged for suspicious prescribing
behavior.
```

Data Understanding

In [10]:
```
# Check the first few rows to understand the structure
data.head()
```

Out[10]:

	Prescription_ID	Patient_ID	Doctor_ID	Pharmacy_ID	Drug_Name	Drug_Category	Dosage	Prescription_Date	Refill_Count	Insurance_Clai
0	1	9107	103	678	Insulin	Antibiotic	98mg	2024-06-07	4	
1	2	3997	917	253	Hydrocodone	Antibiotic	26mg	2024-02-04	1	
2	3	7561	896	623	Metformin	Hypertension	471mg	2024-03-15	0	
3	4	1509	704	998	Lisinopril	Hypertension	342mg	2024-02-06	5	
4	5	1804	853	551	Oxycodone	Antibiotic	180mg	2024-01-11	5	

5 rows × 25 columns

In [11]:
```
# Get basic statistics for numerical columns
data.describe()
```

Out[11]:

	Prescription_ID	Patient_ID	Doctor_ID	Pharmacy_ID	Refill_Count	Insurance_Claim	Total_Cost	Patient_Age	Days_Supply	Fraudule
count	500.000000	500.00000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	50
mean	250.500000	5501.74800	550.192000	603.568000	2.528000	0.494000	513.826040	52.226000	49.400000	
std	144.481833	2514.85357	262.374114	225.583939	1.703826	0.500465	279.881182	20.767607	23.983712	
min	1.000000	1000.00000	101.000000	201.000000	0.000000	0.000000	10.760000	18.000000	7.000000	
25%	125.750000	3414.00000	325.000000	425.500000	1.000000	0.000000	281.707500	34.000000	29.000000	
50%	250.500000	5411.00000	551.500000	603.000000	3.000000	0.000000	510.445000	51.000000	50.000000	
75%	375.250000	7593.00000	789.500000	788.750000	4.000000	1.000000	748.252500	69.000000	71.000000	
max	500.000000	9982.00000	998.000000	998.000000	5.000000	1.000000	999.630000	90.000000	90.000000	

In [12]:
```
# Get a summary of your dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 25 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Prescription_ID           500 non-null    int64  
 1   Patient_ID                500 non-null    int64  
 2   Doctor_ID                 500 non-null    int64  
 3   Pharmacy_ID               500 non-null    int64  
 4   Drug_Name                 500 non-null    object 
 5   Drug_Category             500 non-null    object 
 6   Dosage                    500 non-null    object 
 7   Prescription_Date         500 non-null    object 
 8   Refill_Count              500 non-null    int64  
 9   Insurance_Claim           500 non-null    int64  
 10  Total_Cost                500 non-null    float64
 11  Doctor_Specialty          500 non-null    object 
 12  Patient_Age               500 non-null    int64  
 13  Patient_Gender            500 non-null    object 
 14  Patient_Health_Condition  500 non-null    object 
 15  Prescription_Status       500 non-null    object 
 16  Drug_Schedule             500 non-null    object 
 17  Pharmacy_Location         500 non-null    object 
 18  Prescription_Filled_Date  500 non-null    object 
 19  Days_Supply               500 non-null    int64  
 20  Fraudulent_Claim          500 non-null    int64  
 21  Patient_Region            500 non-null    object 
 22  Doctor_Region             500 non-null    object 
 23  Pharmacy_Region           500 non-null    object 
 24  Doctor_Flagged            500 non-null    int64  
dtypes: float64(1), int64(10), object(14)
memory usage: 70.4+ KB
```

DATA PREPROCESSING

In [14]:
```
# Check for missing values in the dataset
missing_values = data.isnull().sum()
print(missing_values)

# Handling missing values - either drop or impute missing values
data = data.dropna()  # OR impute missing values
# Example: For numerical columns, you can fill missing values with the median or mean
data['Total_Cost'] = data['Total_Cost'].fillna(data['Total_Cost'].median())

```

```
Prescription_ID            0
Patient_ID                 0
Doctor_ID                  0
Pharmacy_ID                0
Drug_Name                  0
Drug_Category              0
Dosage                     0
Prescription_Date          0
Refill_Count               0
Insurance_Claim            0
Total_Cost                 0
Doctor_Specialty           0
Patient_Age                0
Patient_Gender             0
Patient_Health_Condition   0
Prescription_Status        0
Drug_Schedule              0
Pharmacy_Location          0
Prescription_Filled_Date   0
Days_Supply                0
Fraudulent_Claim           0
Patient_Region             0
Doctor_Region              0
Pharmacy_Region            0
Doctor_Flagged             0
dtype: int64
```

DATA AGGREGATION

In [28]:
```
# Aggregation by doctor to see fraud patterns
doctor_summary = df.groupby('Doctor_ID').agg({
    'Total_Cost': 'sum',
    'Prescription_ID': 'count',
    'Fraudulent_Claim': 'sum'
}).reset_index()

doctor_summary.head()

```

Out[28]:

	Doctor_ID	Total_Cost	Prescription_ID	Fraudulent_Claim
0	101	1022.84	2	0
1	103	272.94	1	0
2	105	1725.94	2	1
3	106	235.62	1	1
4	110	389.34	1	0

In [29]:
```
# Convert dates to datetime for easier manipulation
df['Prescription_Date'] = pd.to_datetime(df['Prescription_Date'])
df['Prescription_Filled_Date'] = pd.to_datetime(df['Prescription_Filled_Date'])

# Example: Total fraud claims by month
df['Month'] = df['Prescription_Date'].dt.to_period('M')
monthly_fraud = df.groupby('Month')['Fraudulent_Claim'].sum().reset_index()

print(monthly_fraud)

```

```
     Month  Fraudulent_Claim
0  2024-01                30
1  2024-02                23
2  2024-03                21
3  2024-04                23
4  2024-05                19
5  2024-06                16
6  2024-07                21
7  2024-08                18
8  2024-09                23
9  2024-10                19
10 2024-11                24
11 2024-12                25
```

BASIC STATS

In [30]:
```
mean_value = df['Total_Cost'].mean()
print(f"Mean of Total Cost: {mean_value}")
```

```
Mean of Total Cost: 513.82604
```

In [32]:
```
median_value = df['Total_Cost'].median()
print(f"Median of Total Cost: {median_value}")

```

```
Median of Total Cost: 510.445
```

In [31]:
```
mode_value = df['Total_Cost'].mode()[0]
print(f"Mode of Total Cost: {mode_value}")
```

```
Mode of Total Cost: 164.62
```

In [43]:
```
# IQR (Interquartile Range)
Q1 = df['Total_Cost'].quantile(0.25)  # 25th percentile
Q3 = df['Total_Cost'].quantile(0.75)  # 75th percentile
IQR = Q3 - Q1
print("IQR (Interquartile Range) of Total Cost: {IQR}")
```

```
IQR (Interquartile Range) of Total Cost: 466.5450000000001
```

In [44]:
```
variance = df['Total_Cost'].var()
print("Variance of Total Cost: {variance}")
```

```
Variance of Total Cost: 78333.47627566973
```

In [45]:
```
std_dev = df['Total_Cost'].std()
print("Standard Deviation of Total Cost: {std_dev}")
```

```
Standard Deviation of Total Cost: 279.88118242509574
```

In [46]:
```
cv = (std_dev / mean_value) * 100
print("Coefficient of Variation (CV) of Total Cost: {cv:.2f}%")
```

```
Coefficient of Variation (CV) of Total Cost: 54.47%
```

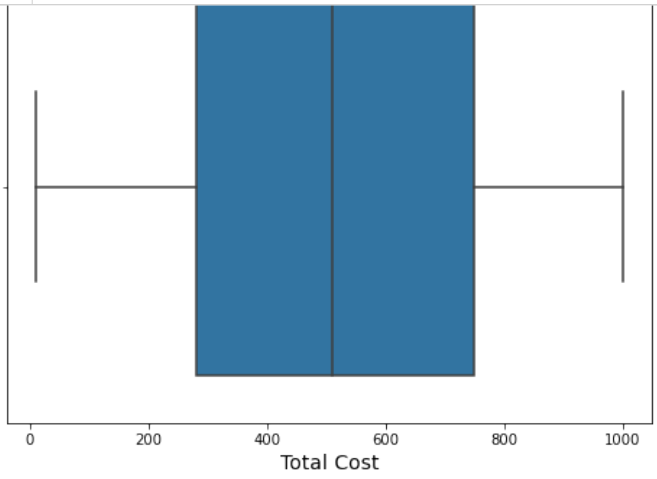
In [47]:
```
mad = df['Total_Cost'].mad()
print(F"Mean Absolute Deviation (MAD) of Total Cost: {mad}")
```

```
Mean Absolute Deviation (MAD) of Total Cost: 241.07819247999996
```

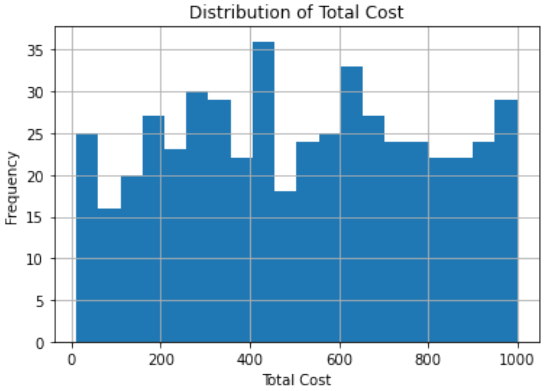
PART B

BOX PLOT

In [26]:
```
# Import necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Step 1: Load your dataset (replace 'your_file.xlsx' with your actual file path)
df = pd.read_excel('C:/Users/AKSHATA/Downloads/prescription_fraud_detection_dataset.xlsx',engine='openpyxl')

# Step 2: Create a box plot for 'Total_Cost'
plt.figure(figsize=(8, 6))
sns.boxplot(x=df['Total_Cost'])

# Step 3: Add labels and title
plt.title('Box Plot of Total Cost', fontsize=16)
plt.xlabel('Total Cost', fontsize=14)

# Step 4: Display the box plot
plt.show()

```



In [40]:
```
import matplotlib.pyplot as plt
# Histogram of 'Total_Cost'
df['Total_Cost'].hist(bins=20)
plt.title('Distribution of Total Cost')
plt.xlabel('Total Cost')
plt.ylabel('Frequency')
plt.show()
```



PART C

In [110]:
```
df = pd.read_excel("C:/Users/AKSHATA/Downloads/prescription_fraud_detection_dataset.xlsx", engine='openpyxl')

df['Dosage'] = df['Dosage'].apply(lambda row: int(row[:-2]))
df['Dosage']
```

Out[110]:
```
0      98
1      26
2     471
3     342
4     180
     ...
495    38
496   162
497   115
498   195
499   415
Name: Dosage, Length: 500, dtype: int64
```

In [111]:
```
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer

# Handling missing values
num_cols = ['Total_Cost', 'Dosage', 'Patient_Age', 'Days_Supply']  # Numerical columns
imputer = SimpleImputer(strategy='median')
df[num_cols] = imputer.fit_transform(df[num_cols])

cat_cols = ['Drug_Name', 'Doctor_Specialty', 'Patient_Health_Condition', 'Pharmacy_Location']  # Categorical
imputer_cat = SimpleImputer(strategy='most_frequent')
df[cat_cols] = imputer_cat.fit_transform(df[cat_cols])

# Encoding categorical variables
# Use One-Hot Encoding for multi-class categorical variables
df = pd.get_dummies(df, columns=cat_cols, drop_first=True)

# Label encode binary categorical variables if necessary
# Example: df['Patient_Gender'] = LabelEncoder().fit_transform(df['Patient_Gender'])

# Scaling numerical features
scaler = StandardScaler()
df[['Total_Cost', 'Dosage', 'Patient_Age', 'Days_Supply']] = scaler.fit_transform(df[['Total_Cost', 'Dosage',

# Final dataset ready for model training
df.head()

```

Out[111]:

	Prescription_ID	Patient_ID	Doctor_ID	Pharmacy_ID	Drug_Category	Dosage	Prescription_Date	Refill_Count	Insurance_Claim	Total_Co
0	1	9107	103	678	Antibiotic	-1.113393	2024-06-07	4	0	-0.8615
1	2	3997	917	253	Antibiotic	-1.630021	2024-02-04	1	0	1.4011
2	3	7561	896	623	Hypertension	1.563032	2024-03-15	0	1	-0.2465
3	4	1509	704	998	Hypertension	0.637405	2024-02-06	5	1	-0.2542
4	5	1804	853	551	Antibiotic	-0.525010	2024-01-11	5	1	-1.2596

5 rows × 132 columns

In [ ]:
```

```