In [19]:
```
# DSBDA Practical A-10:
# Download the Iris flower dataset or any other dataset into a DataFrame.
# Scan the dataset and give the inference as:
# 1. List down the features and their types (e.g., numeric, nominal) available in the
# 2. Create a histogram for each feature in the dataset to illustrate the feature dis
# 3. Create a boxplot for each feature in the dataset.
# 4. Compare distributions and identify outliers.
```

In [20]:
```
# https://www.kaggle.com/datasets/uciml/iris?resource=download
```

In [21]:
```
df=pd.read_csv('Iris.csv')
```

In [22]:
```
df
```

Out[22]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [24]:
```
df.dtypes
```

Out[24]:
```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [25]:
```
df.describe()
```

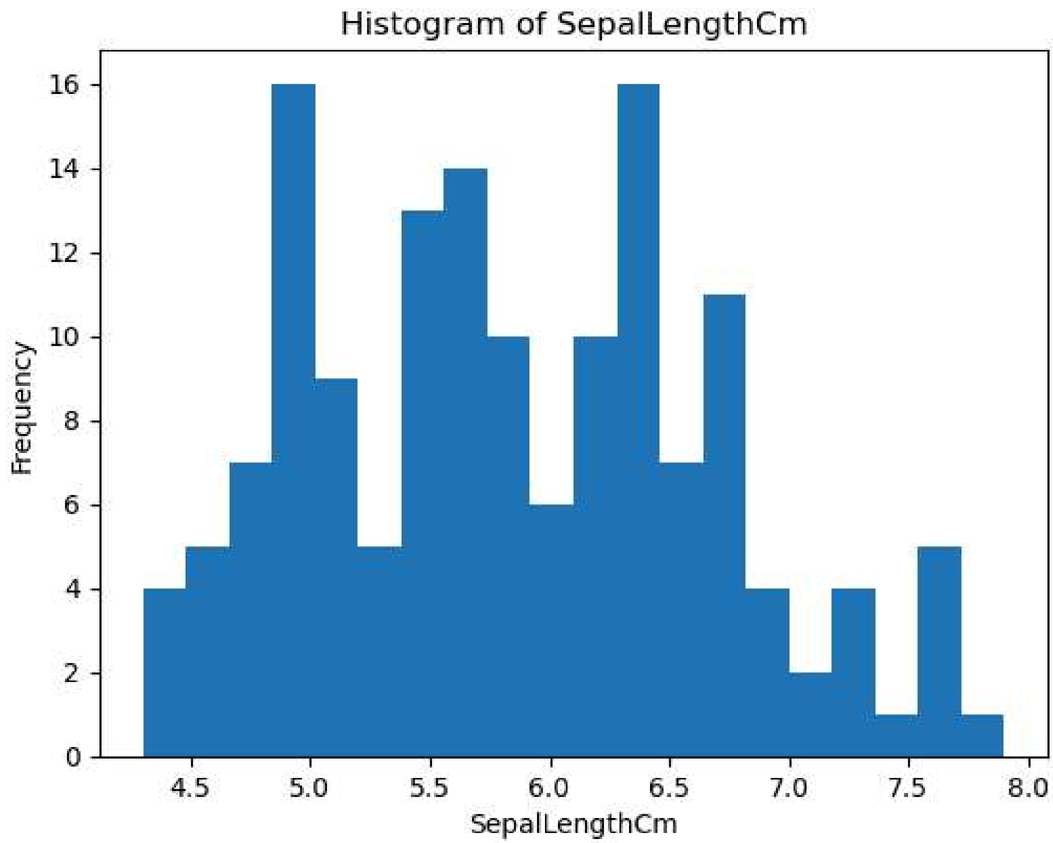
Out[25]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [26]:
```
# Draw Histograms for each feature in dataset to illustrate the feature distributions
import matplotlib.pyplot as plt
```

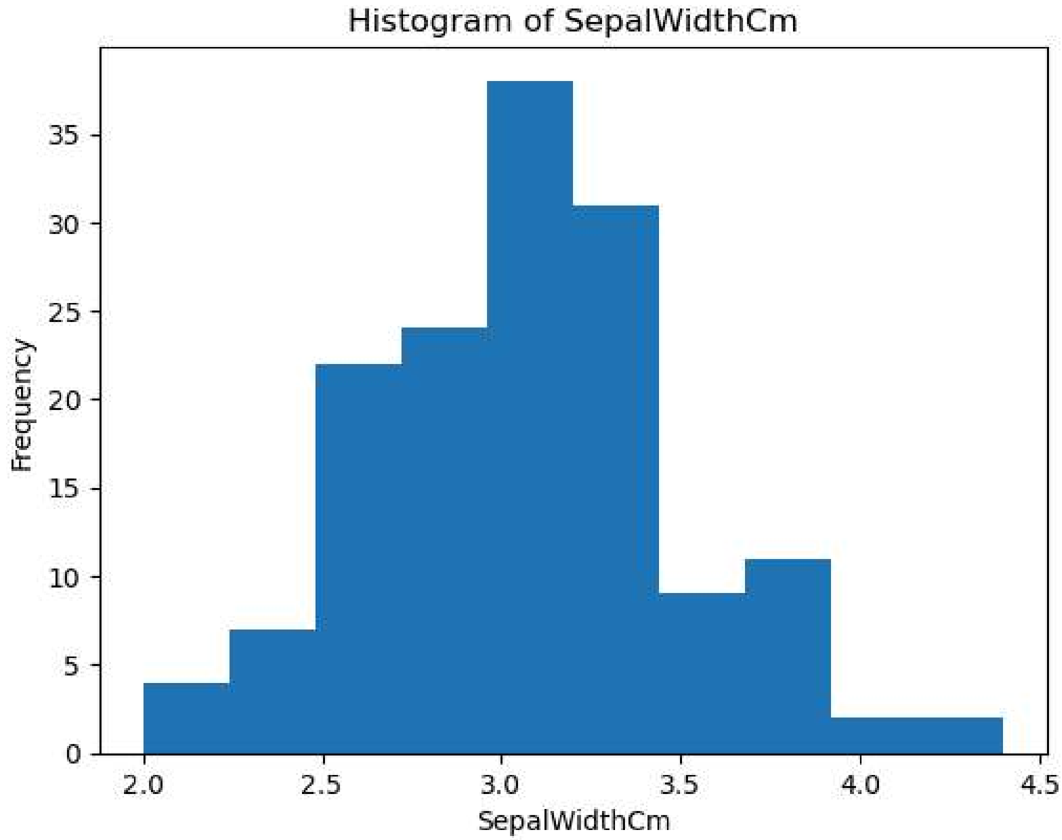
In [27]:
```
plt.hist(df['SepalLengthCm'], bins=20) # Adjust the number of bins as needed
plt.title(f'Histogram of SepalLengthCm')
plt.xlabel('SepalLengthCm')
plt.ylabel('Frequency')
```

Out[27]: `Text(0, 0.5, 'Frequency')`



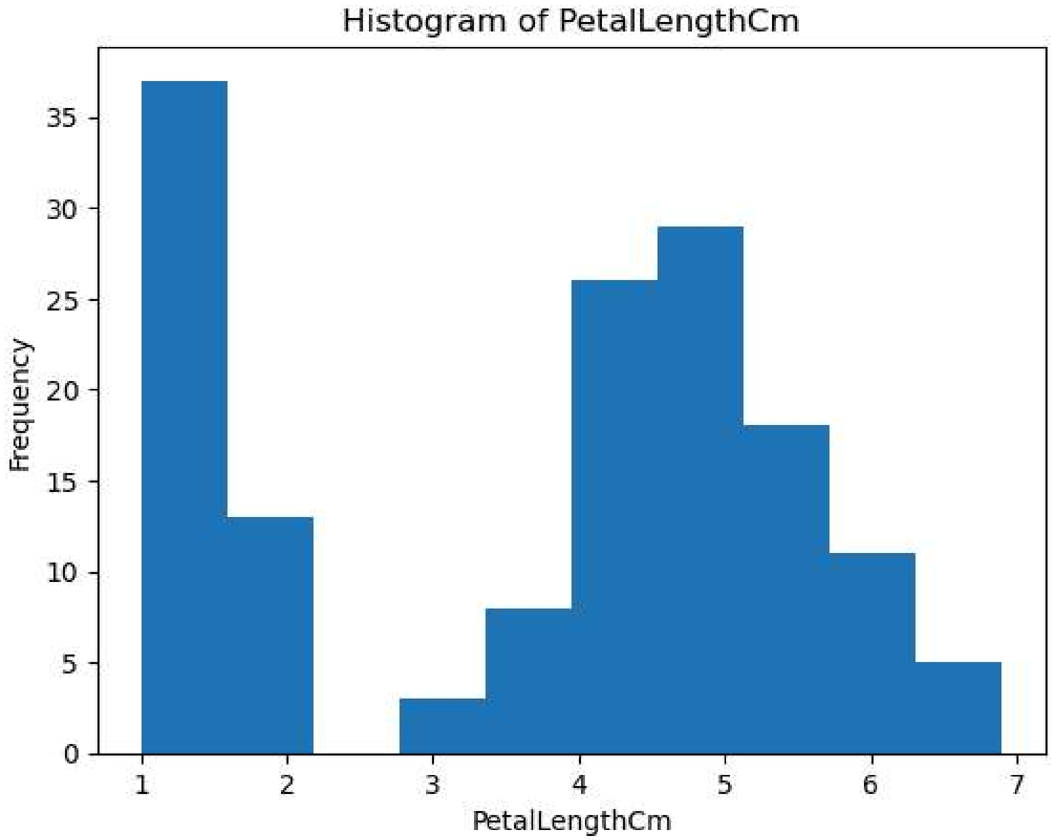
In [28]:
```
plt.hist(df['SepalWidthCm'], bins=10) # Adjust the number of bins as needed
plt.title(f'Histogram of SepalWidthCm')
plt.xlabel('SepalWidthCm')
plt.ylabel('Frequency')
```

Out[28]: Text(0, 0.5, 'Frequency')



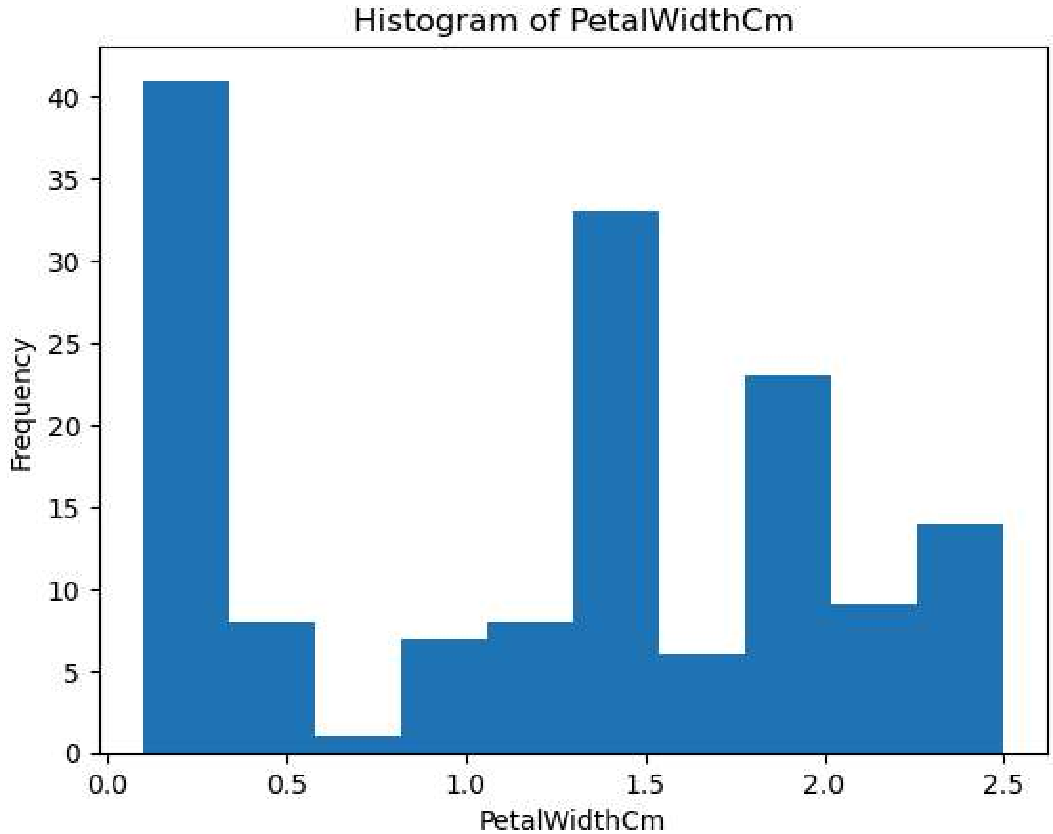
In [29]:
```
plt.hist(df['PetalLengthCm'], bins=10) # Adjust the number of bins as needed
plt.title(f'Histogram of PetalLengthCm')
plt.xlabel('PetalLengthCm')
plt.ylabel('Frequency')
```

Out[29]: Text(0, 0.5, 'Frequency')

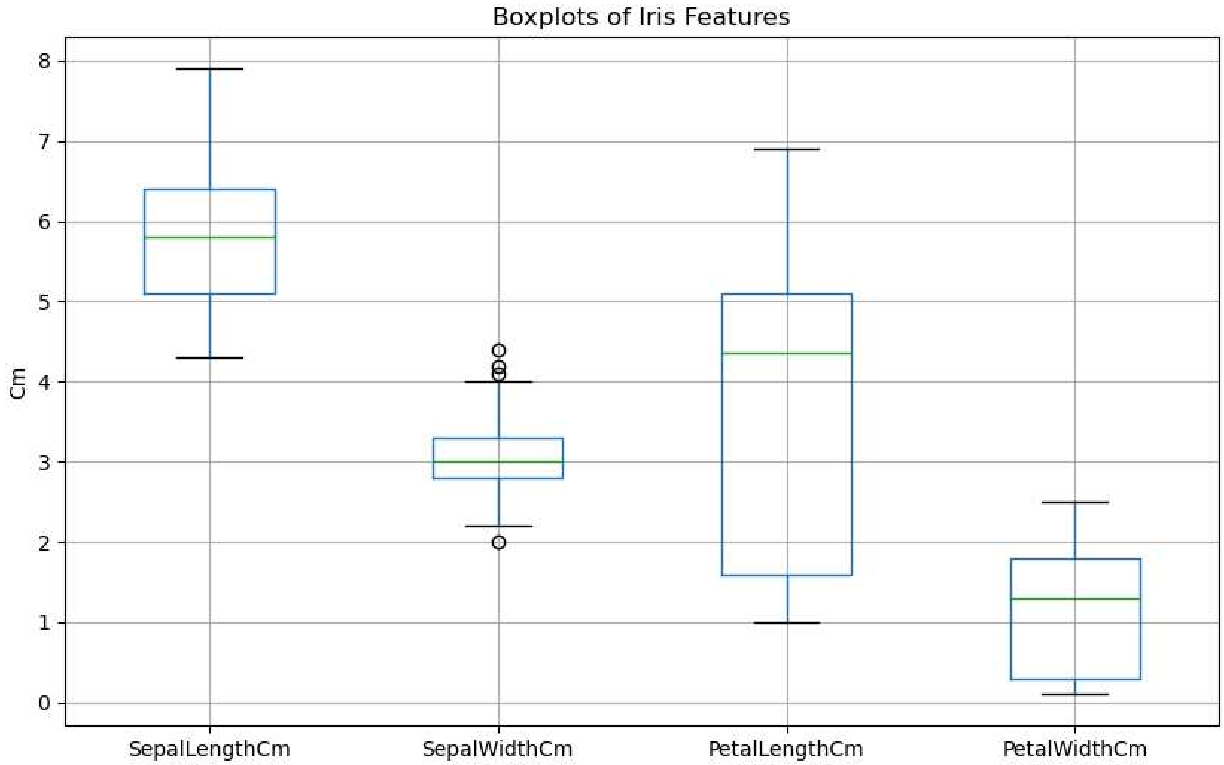


In [30]:
```
plt.hist(df['PetalWidthCm'], bins=10) # Adjust the number of bins as needed
plt.title(f'Histogram of PetalWidthCm')
plt.xlabel('PetalWidthCm')
plt.ylabel('Frequency')
```

Out[30]:
```
Text(0, 0.5, 'Frequency')
```



In [31]:
```
# Create boxplots for SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm
plt.figure(figsize=(10, 6))
df.boxplot(column=['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm'])
plt.title('Boxplots of Iris Features')
plt.ylabel('Cm')
plt.show()
```



In [32]:
```
# Detect outlier in Boxplot of SepalWidthCm Column
# The boxplot itself visually represents outliers.
# We can use IQR to programmatically find them.
```

In [33]:
```
# Calculate Q1, Q3, and IQR
Q1 = df['SepalWidthCm'].quantile(0.25)
Q3 = df['SepalWidthCm'].quantile(0.75)
IQR = Q3 - Q1
```

In [34]:
```
# Define bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
```

In [36]:
```
lower_bound
```

Out[36]: 2.05

In [37]:
```
upper_bound
```

Out[37]: 4.05

In [38]:
```
# Identify outliers
outliers = df[(df['SepalWidthCm'] < lower_bound) | (df['SepalWidthCm'] > upper_bound)
```

In [40]:
```
# Print or further process the outliers
print("Outliers in SepalWidthCm:")
outliers
```

Outliers in SepalWidthCm:

Out[40]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
15	16	5.7	4.4	1.5	0.4	Iris-setosa
32	33	5.2	4.1	1.5	0.1	Iris-setosa
33	34	5.5	4.2	1.4	0.2	Iris-setosa
60	61	5.0	2.0	3.5	1.0	Iris-versicolor

In [44]:
```
# Printing the Row Indexes of Outliers
outlier_indices = outliers.index
print("Outliers RowIndex:")
outlier_indices
```

Outliers RowIndex:

Out[44]:
```
Int64Index([15, 32, 33, 60], dtype='int64')
```