```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('/content/Iris.csv')
df
```

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
...	...	...	...	...	...
144	6.7	3.0	5.2	2.3	Iris-virginica
145	6.3	2.5	5.0	1.9	Iris-virginica
146	6.5	3.0	5.2	2.0	Iris-virginica
147	6.2	3.4	5.4	2.3	Iris-virginica
148	5.9	3.0	5.1	1.8	Iris-virginica

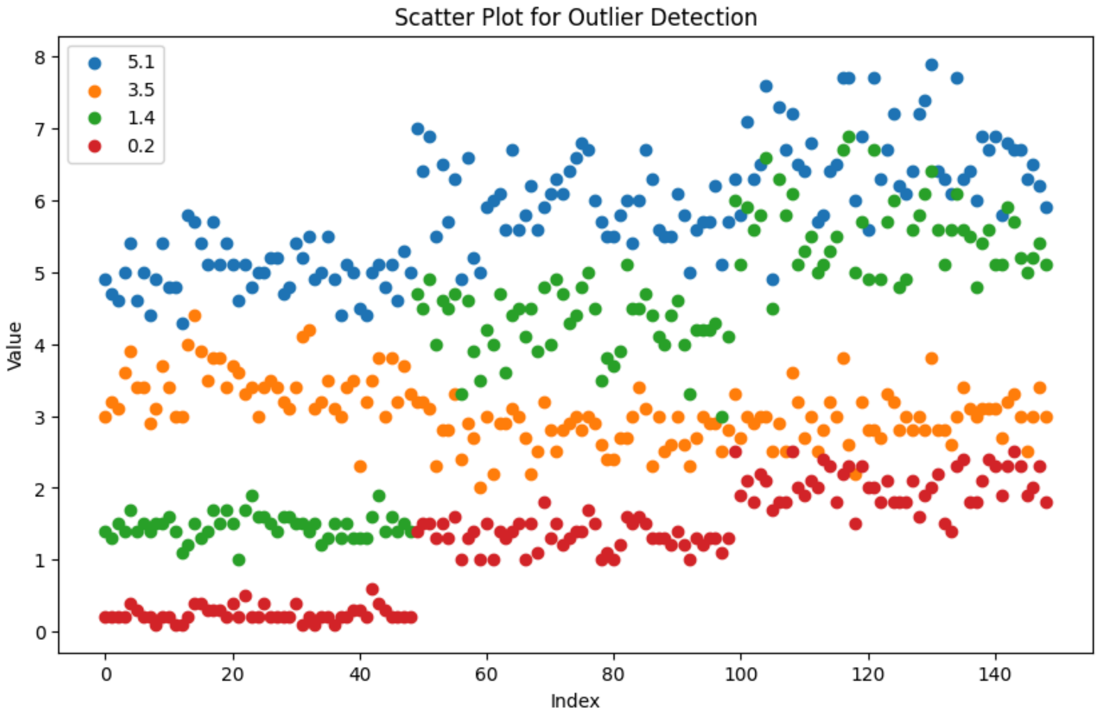
149 rows × 5 columns

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

```
# Select numerical columns for outlier detection
numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns
```

2. Detect outliers using ScatterPlot

```
plt.figure(figsize=(10,6))
for col in numeric_columns:
    plt.scatter(df.index, df[col], label=col)
plt.xlabel("Index")
plt.ylabel("Value")
plt.title("Scatter Plot for Outlier Detection")
plt.legend()
plt.show()
```



3. Handle outliers using Quantile-based Flooring and Capping

```
def cap_outliers(df, column):
    Q1 = df[column].quantile(0.10)  # 10th percentile
    Q3 = df[column].quantile(0.90)  # 90th percentile
    df[column] = np.where(df[column] < Q1, Q1, df[column])  # Flooring
    df[column] = np.where(df[column] > Q3, Q3, df[column])  # Capping

# Apply function to each numerical column
for col in numeric_columns:
    cap_outliers(df, col)

# Print updated dataset
print(df.head())
```

```
   5.1   3.5  1.4  0.2  Iris-setosa
0  4.9  3.00  1.4  0.2  Iris-setosa
1  4.8  3.20  1.4  0.2  Iris-setosa
2  4.8  3.10  1.5  0.2  Iris-setosa
3  5.0  3.60  1.4  0.2  Iris-setosa
4  5.4  3.62  1.7  0.4  Iris-setosa
```