In [ ]:
```
import pandas as pd

# Load dataset
df = pd.read_csv("iris_dataset.csv")
print(df.head())

# Explore
print(df.describe())
print("Target classes:", df["species"].unique())
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

  species
0  setosa
1  setosa
2  setosa
3  setosa
4  setosa
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)
count        150.000000
mean           1.199333
std            0.762238
min            0.100000
25%            0.300000
50%            1.300000
75%            1.800000
max            2.500000
Target classes: ['setosa' 'versicolor' 'virginica']
```

In [ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report

# Load dataset
df = pd.read_csv("iris_dataset.csv")

# Features and target
X = df.drop(columns=["species"])
```

```
y = df["species"]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc

# Train model
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Evaluation
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

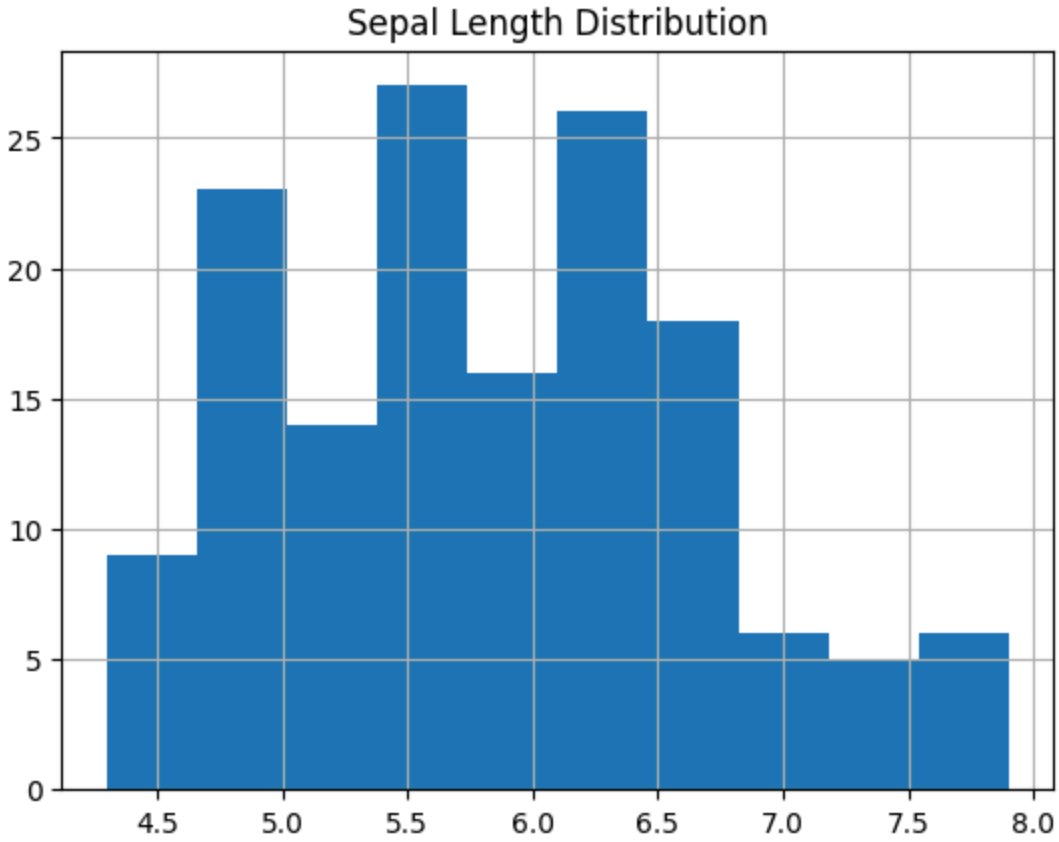
In [5]:

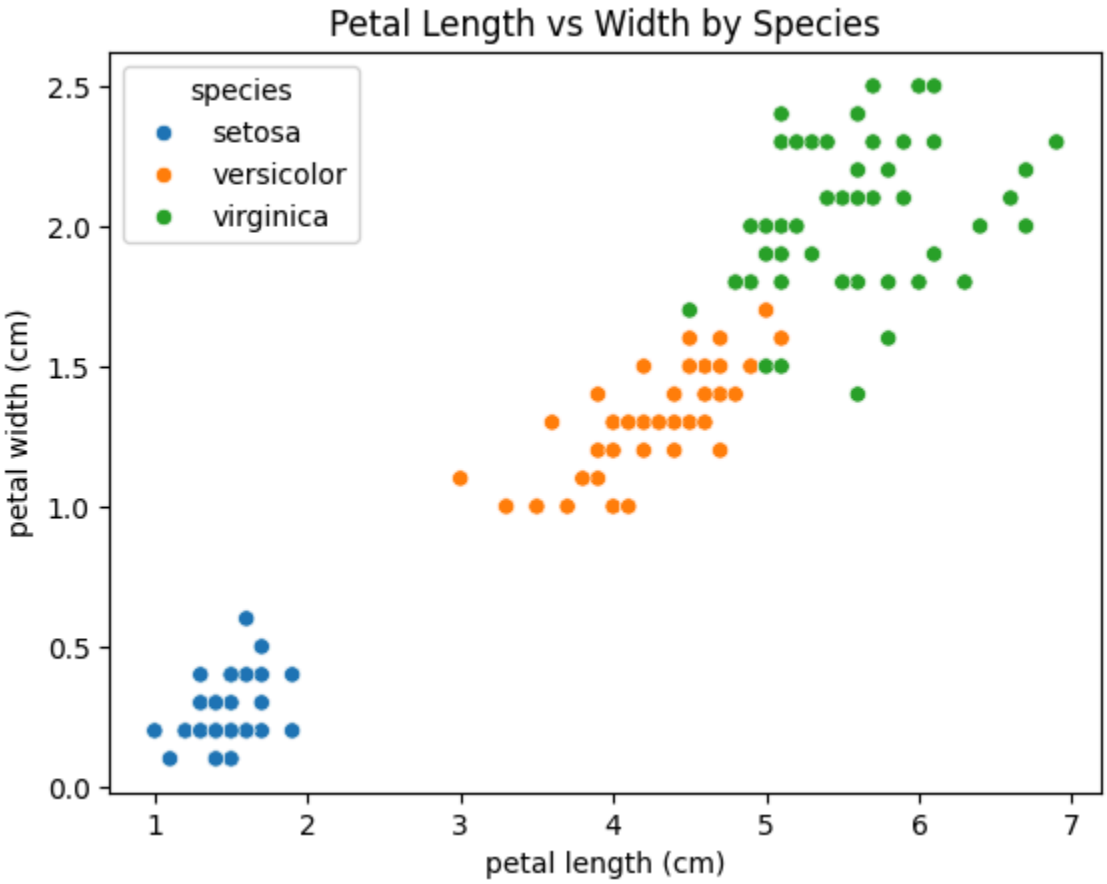
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Load dataset (assuming iris_dataset.csv is the intended dataset)
df = pd.read_csv("iris_dataset.csv")

df['sepal length (cm)'].hist()
plt.title("Sepal Length Distribution")
plt.show()

sns.scatterplot(data=df,x="petal length (cm)",y='petal width (cm)',hue="specie
plt.title("Petal Length vs Width by Species")
plt.show()
```


Sepal Length Distribution
25
20
15
10
5
0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Petal Length vs Width by Species
species
setosa
versicolor
virginica
petal width (cm)
petal length (cm)
0.0
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7

In [ ]:

```python
import pandas as pd

# Load dataset
df = pd.read_csv("housing.csv")
print(df.head())

# Explore
print(df.describe())
print("Columns:", df.columns.tolist())
```

```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                41.0        880.0           129.0
1    -122.22     37.86                21.0       7099.0          1106.0
2    -122.24     37.85                52.0       1467.0           190.0
3    -122.25     37.85                52.0       1274.0           235.0
4    -122.25     37.85                52.0       1627.0           280.0

   population  households  median_income  median_house_value ocean_proximity
0       322.0       126.0         8.3252            452600.0        NEAR BAY
1      2401.0      1138.0         8.3014            358500.0        NEAR BAY
2       496.0       177.0         7.2574            352100.0        NEAR BAY
3       558.0       219.0         5.6431            341300.0        NEAR BAY
4       565.0       259.0         3.8462            342200.0        NEAR BAY
          longitude      latitude  housing_median_age   total_rooms  \
count  20640.000000  20640.000000        20640.000000  20640.000000
mean    -119.569704     35.631861           28.639486   2635.763081
std        2.003532      2.135952           12.585558   2181.615252
min     -124.350000     32.540000            1.000000      2.000000
25%     -121.800000     33.930000           18.000000   1447.750000
50%     -118.490000     34.260000           29.000000   2127.000000
75%     -118.010000     37.710000           37.000000   3148.000000
max     -114.310000     41.950000           52.000000  39320.000000

       total_bedrooms    population    households  median_income  \
count    20433.000000  20640.000000  20640.000000   20640.000000
mean       537.870553   1425.476744    499.539680       3.870671
std        421.385070   1132.462122    382.329753       1.899822
min          1.000000      3.000000      1.000000       0.499900
25%        296.000000    787.000000    280.000000       2.563400
50%        435.000000   1166.000000    409.000000       3.534800
75%        647.000000   1725.000000    605.000000       4.743250
max       6445.000000  35682.000000   6082.000000      15.000100

       median_house_value
count        20640.000000
mean        206855.816909
std         115395.615874
min          14999.000000
25%         119600.000000
50%         179700.000000
75%         264725.000000
max         500001.000000
Columns: ['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'tota
l_bedrooms', 'population', 'households', 'median_income', 'median_house_value',
'ocean_proximity']
```

In [ ]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import numpy as np
```

```
# Load dataset
df = pd.read_csv("housing.csv")

# Drop missing values (if any)
df = df.dropna()

# Features and target (assuming "median_house_value" is the target column)
X = df.drop(columns=["median_house_value"])
y = df["median_house_value"]

# Identify categorical and numerical features
categorical_features = ['ocean_proximity']
numerical_features = X.select_dtypes(include=np.number).columns.tolist()

# Create a column transformer to apply one-hot encoding to categorical feature
preprocessor = ColumnTransformer(
    transformers=[
        ('num', 'passthrough', numerical_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])

# Create a pipeline with the preprocessor and the linear regression model
model = Pipeline(steps=[('preprocessor', preprocessor),
                        ('regressor', LinearRegression())])

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc

# Train model
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Evaluation
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error (RMSE):", rmse)
```

```
Root Mean Squared Error (RMSE): 69297.71669113018
```