Heart Disease Dataset – Problem Statements (Python)

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

1. Load and Explore the Dataset

- Load the dataset using pandas
- Display first 10 rows
- Check data types
- Find missing values
- Display summary statistics

In [2]:
```
df = pd.read_csv("heart.csv")
```

In [3]:
```
# First 10 rows
print(df.head(10))

# Data types
print(df.dtypes)

#Missing Values
print(df.isnull().sum())

# Summary statistics
print(df.describe())
```

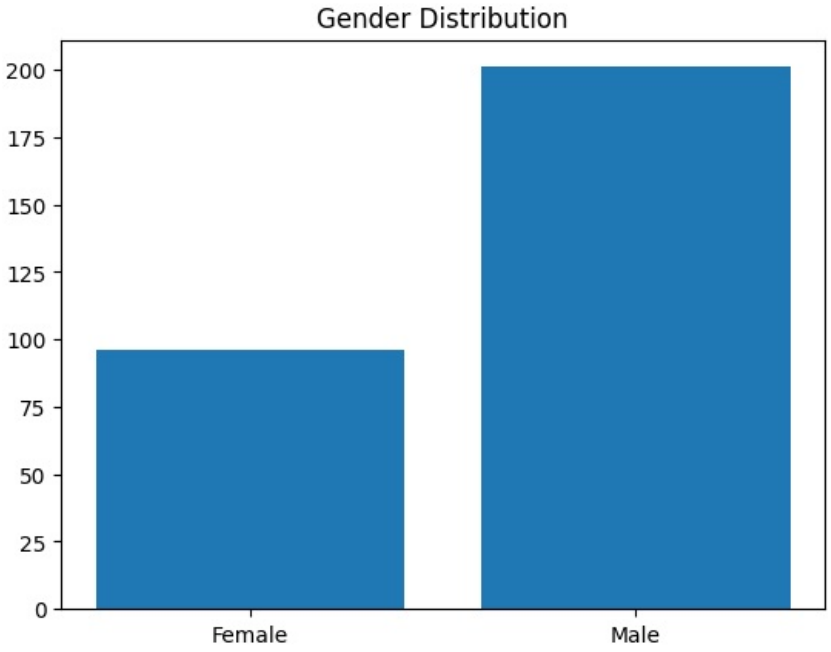
```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   69    1   0       160   234    1        2      131      0      0.1      1
1   69    0   0       140   239    0        0      151      0      1.8      0
2   66    0   0       150   226    0        0      114      0      2.6      2
3   65    1   0       138   282    1        2      174      0      1.4      1
4   64    1   0       110   211    0        2      144      1      1.8      1
5   64    1   0       170   227    0        2      155      0      0.6      1
6   63    1   0       145   233    1        2      150      0      2.3      2
7   61    1   0       134   234    0        0      145      0      2.6      1
8   60    0   0       150   240    0        0      171      0      0.9      0
9   59    1   0       178   270    0        2      145      0      4.2      2

   ca  thal  condition
0   1     0          0
1   2     0          0
2   0     0          0
3   1     0          1
4   0     0          0
5   0     2          0
6   0     1          0
7   2     0          1
8   0     0          0
9   0     2          0
age            int64
sex            int64
cp             int64
trestbps       int64
chol           int64
fbs            int64
restecg        int64
thalach        int64
exang          int64
oldpeak      float64
slope          int64
ca             int64
thal           int64
condition      int64
dtype: object
age          0
sex          0
cp           0
trestbps     0
chol         0
fbs          0
restecg      0
thalach      0
exang        0
oldpeak      0
slope        0
ca           0
thal         0
condition    0
dtype: int64
              age         sex          cp    trestbps        chol         fbs  \
count  297.000000  297.000000  297.000000  297.000000  297.000000  297.000000
mean    54.542088    0.676768    2.158249  131.693603  247.350168    0.144781
std      9.049736    0.468500    0.964859   17.762806   51.997583    0.352474
min     29.000000    0.000000    0.000000   94.000000  126.000000    0.000000
25%     48.000000    0.000000    2.000000  120.000000  211.000000    0.000000
50%     56.000000    1.000000    2.000000  130.000000  243.000000    0.000000
75%     61.000000    1.000000    3.000000  140.000000  276.000000    0.000000
max     77.000000    1.000000    3.000000  200.000000  564.000000    1.000000

          restecg     thalach       exang     oldpeak       slope          ca  \
count  297.000000  297.000000  297.000000  297.000000  297.000000  297.000000
mean     0.996633  149.599327    0.326599    1.055556    0.602694    0.676768
std      0.994914   22.941562    0.469761    1.166123    0.618187    0.938965
min      0.000000   71.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000  133.000000    0.000000    0.000000    0.000000    0.000000
50%      1.000000  153.000000    0.000000    0.800000    1.000000    0.000000
75%      2.000000  166.000000    1.000000    1.600000    1.000000    1.000000
max      2.000000  202.000000    1.000000    6.200000    2.000000    3.000000

            thal   condition
count  297.000000  297.000000
mean     0.835017    0.461279
std      0.956690    0.499340
min      0.000000    0.000000
25%      0.000000    0.000000
50%      0.000000    0.000000
75%      2.000000    1.000000
max      2.000000    1.000000
```

2. Gender Distribution Analysis

- Count number of males and females
- Calculate percentage distribution using NumPy
- Plot a bar chart using Matplotlib

In [4]:
```
gender_count = df["sex"].value_counts()
print(gender_count)

gender_percent = (gender_count / len(df)) * 100
print(gender_percent)

plt.bar(gender_count.index, gender_count.values)
plt.xticks([0,1], ["Female", "Male"])
plt.title("Gender Distribution")
plt.show()
```

```
sex
1    201
0     96
Name: count, dtype: int64
sex
1    67.676768
0    32.323232
Name: count, dtype: float64
```

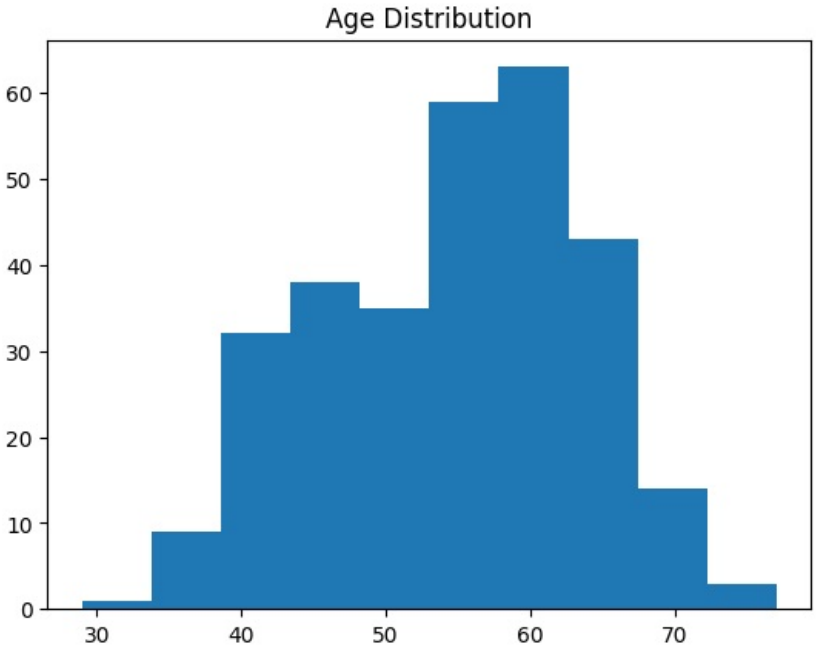


3. Age Analysis Find:

- Minimum age
- Maximum age
- Mean age
- Median age
- Plot histogram of age distribution

In [5]:
```
#3. Age analytics

print("Min Age:", df["age"].min())
print("Max Age:", df["age"].max())
print("Mean Age:", df["age"].mean())
print("Median Age:", df["age"].median())

plt.hist(df["age"], bins=10)
plt.title("Age Distribution")
plt.show()
```

```
Min Age: 29
Max Age: 77
Mean Age: 54.54208754208754
Median Age: 56.0
```

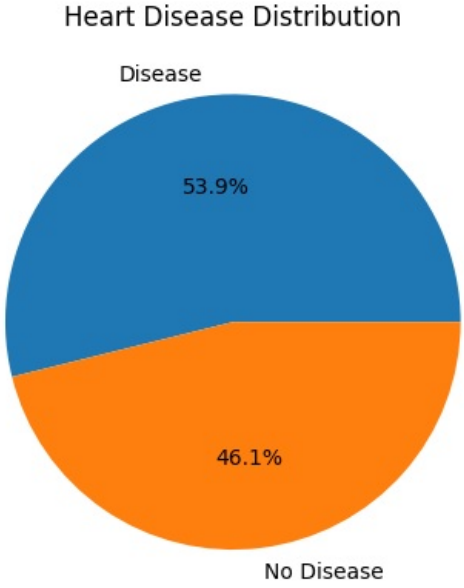


4. Target Variable Analysis

- Count number of patients with and without heart disease
- Plot pie chart
- Calculate disease percentage

In [6]:
```
#4. Target Variable Analysis

target_count = df["condition"].value_counts()
print(target_count)

disease_percent = (target_count[1] / len(df)) * 100
print("Disease Percentage:", disease_percent)

plt.pie(target_count, labels=["Disease","No Disease"], autopct='%1.1f%%')
plt.title("Heart Disease Distribution")
plt.show()
```

```
condition
0    160
1    137
Name: count, dtype: int64
Disease Percentage: 46.12794612794613
```

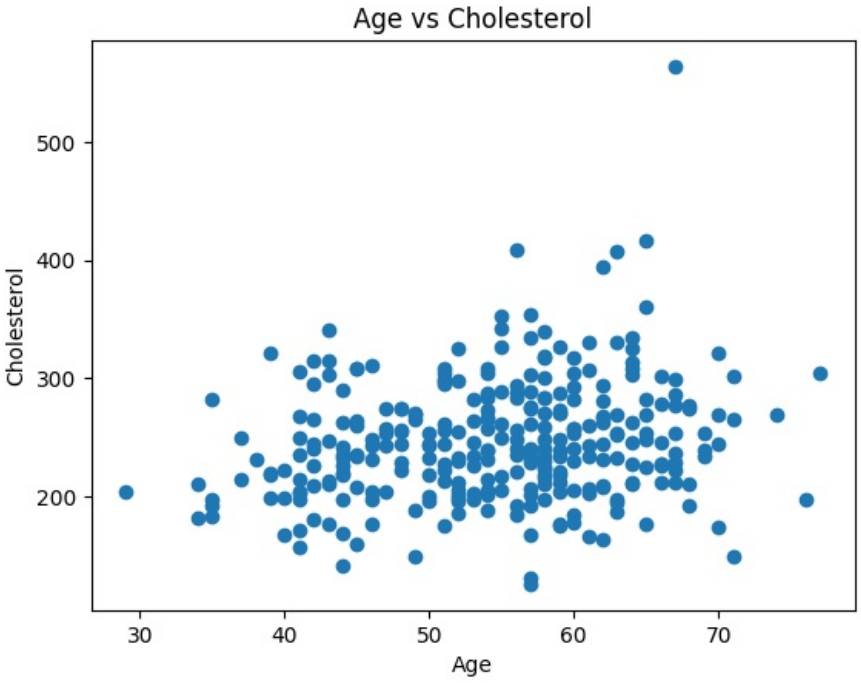


5. Correlation Between Age and Cholesterol

- Calculate correlation using df.corr()
- Plot scatter plot (Age vs Cholesterol)
- Interpret relationship

In [7]:
```
#Correlation Between Age and Cholesterol

print(df[["age","chol"]].corr())

plt.scatter(df["age"], df["chol"])
plt.xlabel("Age")
plt.ylabel("Cholesterol")
plt.title("Age vs Cholesterol")
plt.show()
```

```
          age      chol
age   1.000000  0.202644
chol  0.202644  1.000000
```



In [8]:
```
# Calculate correlation
corr_value = df['age'].corr(df['chol'])

print("Correlation between Age and Cholesterol:", round(corr_value, 3))

if corr_value >= 0.7:
    interpretation = "Strong positive correlation: Cholesterol increases significantly with age."
elif corr_value >= 0.3:
    interpretation = "Moderate positive correlation: Cholesterol tends to increase with age."
elif corr_value > -0.3:
    interpretation = "Weak negative correlation: Cholesterol slightly decreases with age."
elif corr_value > -0.7:
    interpretation = "Moderate negative correlation: Cholesterol tends to decrease with age."
else:
    interpretation = "Strong negative correlation: Cholesterol decreases significantly with age."

print("Interpretation:", interpretation)
```

```
Correlation between Age and Cholesterol: 0.203
Interpretation: Weak negative correlation: Cholesterol slightly decreases with age.
```

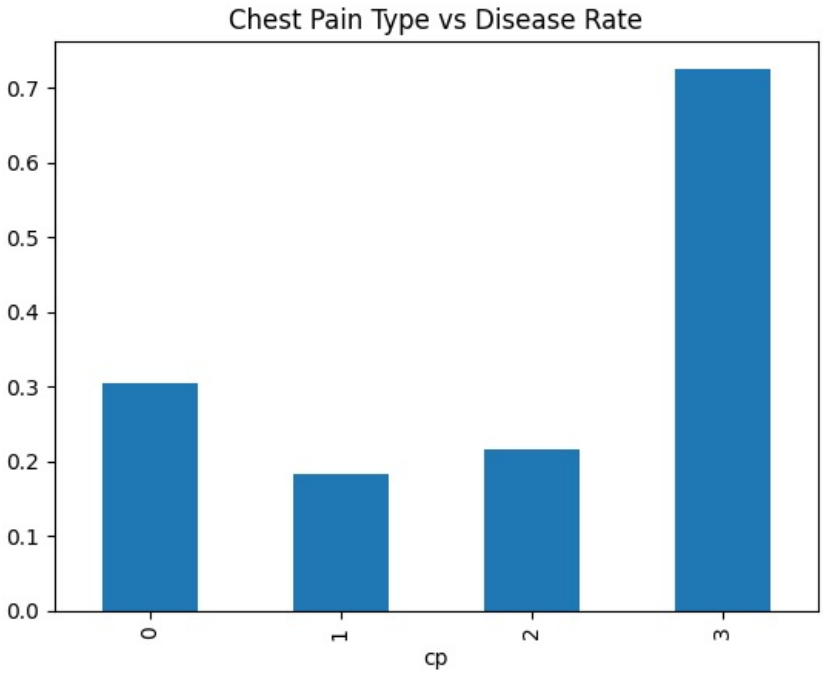
6. Chest Pain Type vs Disease

- Group by cp and calculate disease rate
- Plot grouped bar chart
- Identify which chest pain type is most risky

In [9]:
```
# Chest Pain Type vs Disease

cp_group = df.groupby("cp")["condition"].mean()
print(cp_group)

cp_group.plot(kind="bar")
plt.title("Chest Pain Type vs Disease Rate")
```

```
plt.show()
```

```
cp
0    0.304348
1    0.183673
2    0.216867
3    0.725352
Name: condition, dtype: float64
```



In [10]:
```
most_risky_cp = cp_group.idxmax()
highest_rate = cp_group.max()

print("Most Risky Chest Pain Type:", most_risky_cp)
print("Disease Rate:", round(highest_rate * 100, 2), "%")
```

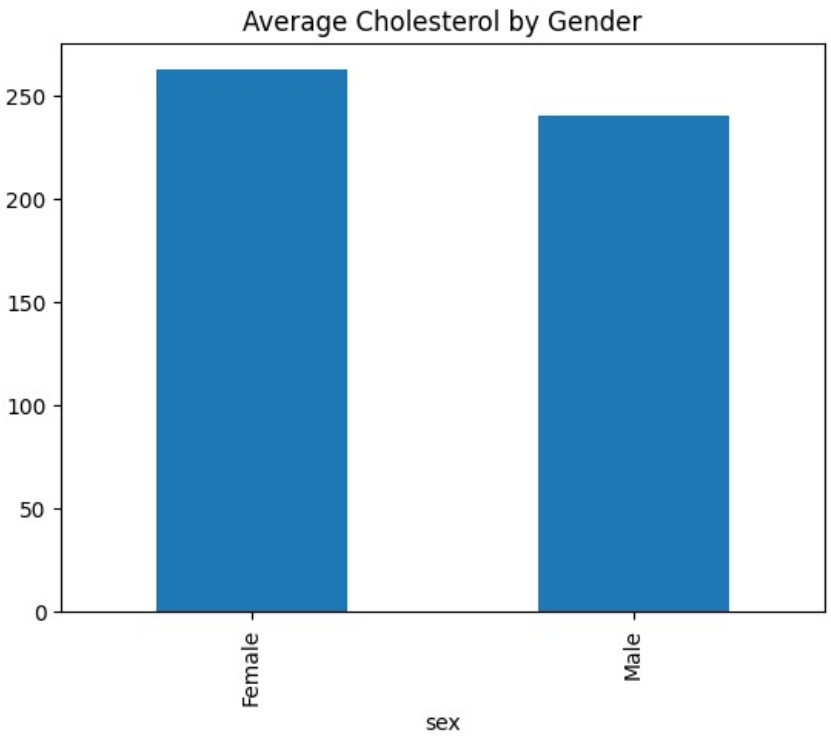
```
Most Risky Chest Pain Type: 3
Disease Rate: 72.54 %
```

7. Average Cholesterol by Gender

- Group by sex
- Calculate mean cholesterol
- Visualize using bar plot

In [11]:
```
#Average Cholesterol by Gender

chol_gender = df.groupby("sex")["chol"].mean()
print(chol_gender)

chol_gender.plot(kind="bar")
plt.xticks([0,1], ["Female","Male"])
plt.title("Average Cholesterol by Gender")
plt.show()
```

```
sex
0    262.229167
1    240.243781
Name: chol, dtype: float64
```



8. Resting Blood Pressure Analysis Find:

- Average BP
- Patients with BP > 140
- Compare disease presence in high BP group

In [12]:
```
#8.Resting Blood Pressure Analysis

avg_bp = df["trestbps"].mean()
print("Average BP:", round(avg_bp, 2))

high_bp = df[df["trestbps"] > 140]
print("Patients with BP > 140:", len(high_bp))

print("\nDisease Count in High BP Group:")
print(high_bp["condition"].value_counts())

high_bp_disease_rate = high_bp["condition"].mean() * 100
print("\nDisease Rate in High BP Group:", round(high_bp_disease_rate, 2), "%")

overall_disease_rate = df["condition"].mean() * 100
print("Overall Disease Rate:", round(overall_disease_rate, 2), "%")
```

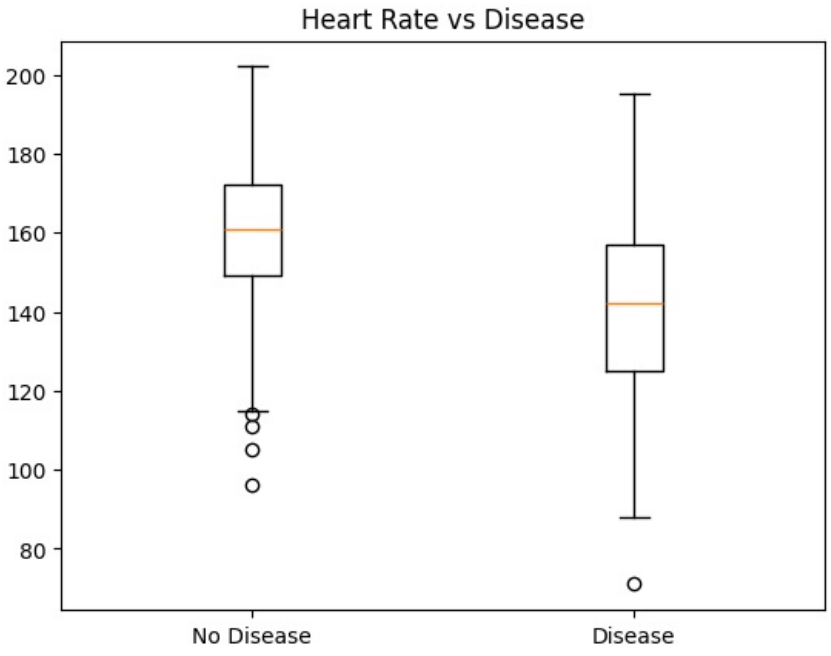
```
Average BP: 131.69
Patients with BP > 140: 66

Disease Count in High BP Group:
condition
1    39
0    27
Name: count, dtype: int64

Disease Rate in High BP Group: 59.09 %
Overall Disease Rate: 46.13 %
```

9. Maximum Heart Rate vs Disease

- Compare average thalach for:
- Disease patients
- Non-disease patients
- Plot boxplot

In [13]:
```
#9. Maximum Heart Rate vs Disease

print("Avg HR (Disease):", df[df["condition"]==1]["thalach"].mean())
print("Avg HR (No Disease):", df[df["condition"]==0]["thalach"].mean())

plt.boxplot([df[df["condition"]==0]["thalach"],
             df[df["condition"]==1]["thalach"]])
plt.xticks([1,2], ["No Disease","Disease"])
plt.title("Heart Rate vs Disease")
plt.show()
```

```
Avg HR (Disease): 139.1094890510949
Avg HR (No Disease): 158.58125
```



10. Exercise Induced Angina Impact

- Calculate disease percentage in:

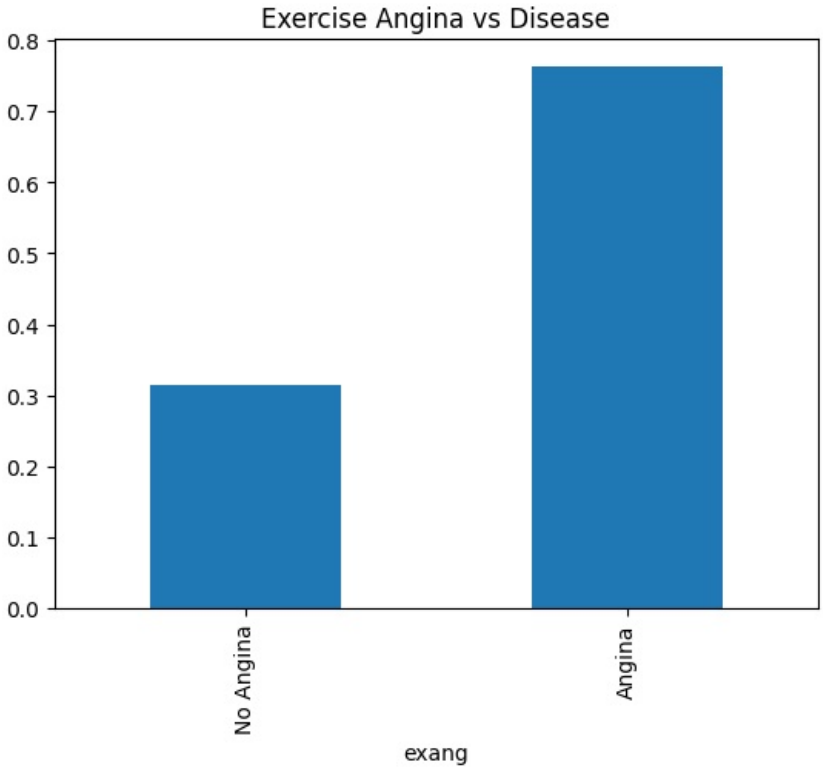
  exang = 1

  exang = 0

- Visualize using bar chart

In [14]:
```
#10. Exercise Induced Angina Impact

exang_group = df.groupby("exang")["condition"].mean()
print(exang_group)

exang_group.plot(kind="bar")
plt.xticks([0,1], ["No Angina","Angina"])
plt.title("Exercise Angina vs Disease")
plt.show()
```

```
exang
0    0.315000
1    0.762887
Name: condition, dtype: float64
```

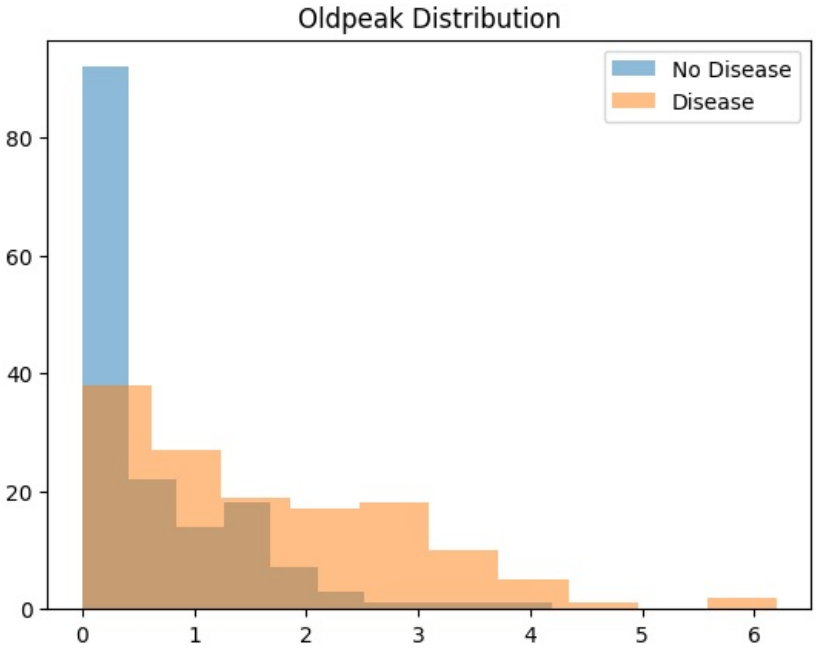


11. ST Depression (oldpeak) Analysis

- Calculate mean oldpeak by target
- Plot histogram for both classes
- Identify trend

In [15]:
```
#11. ST Depression (oldpeak) Analysis
oldpeak_mean = df.groupby("condition")["oldpeak"].mean()
print(oldpeak_mean)

plt.hist(df[df["condition"]==0]["oldpeak"], alpha=0.5, label="No Disease")
plt.hist(df[df["condition"]==1]["oldpeak"], alpha=0.5, label="Disease")
plt.legend()
plt.title("Oldpeak Distribution")
plt.show()
```

```
condition
0    0.598750
1    1.589051
Name: oldpeak, dtype: float64
```



In [16]:
```
mean_no_disease = oldpeak_mean[0]
mean_disease = oldpeak_mean[1]

if mean_disease > mean_no_disease:
    print("\nTrend: Patients with heart disease have higher ST depression values.")
elif mean_disease < mean_no_disease:
    print("\nTrend: Patients without heart disease have higher ST depression values.")
else:
    print("\nTrend: No significant difference in ST depression between groups.")
```

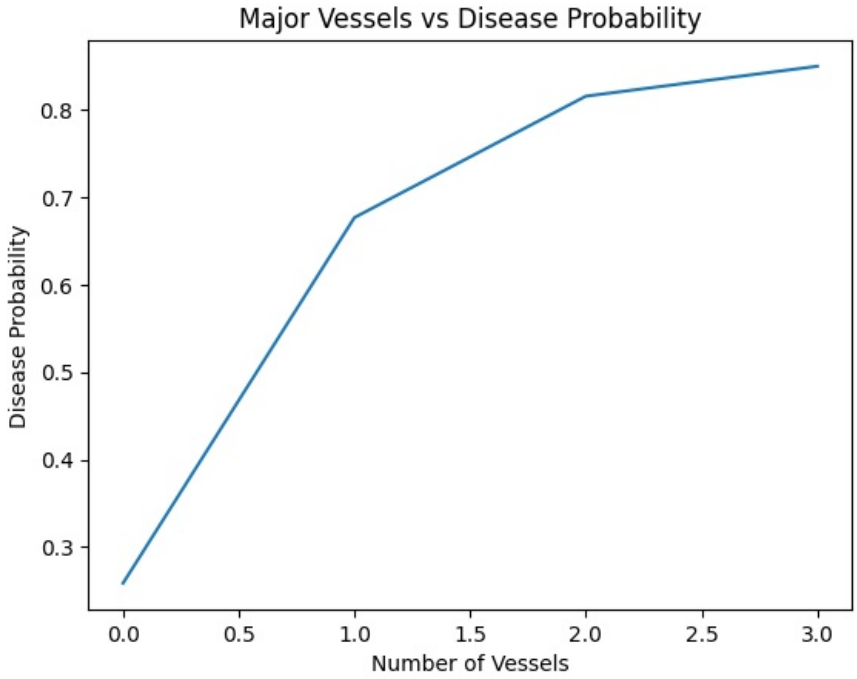
```
Trend: Patients with heart disease have higher ST depression values.
```

12. Number of Major Vessels (ca) Impact

- Group by ca
- Calculate disease probability
- Plot line chart

In [17]:
```
#12. Number of Major Vessels (ca) Impact

ca_group = df.groupby("ca")["condition"].mean()
print(ca_group)

plt.plot(ca_group.index, ca_group.values)
plt.title("Major Vessels vs Disease Probability")
plt.xlabel("Number of Vessels")
plt.ylabel("Disease Probability")
plt.show()
```

```
ca
0    0.258621
1    0.676923
2    0.815789
3    0.850000
Name: condition, dtype: float64
```

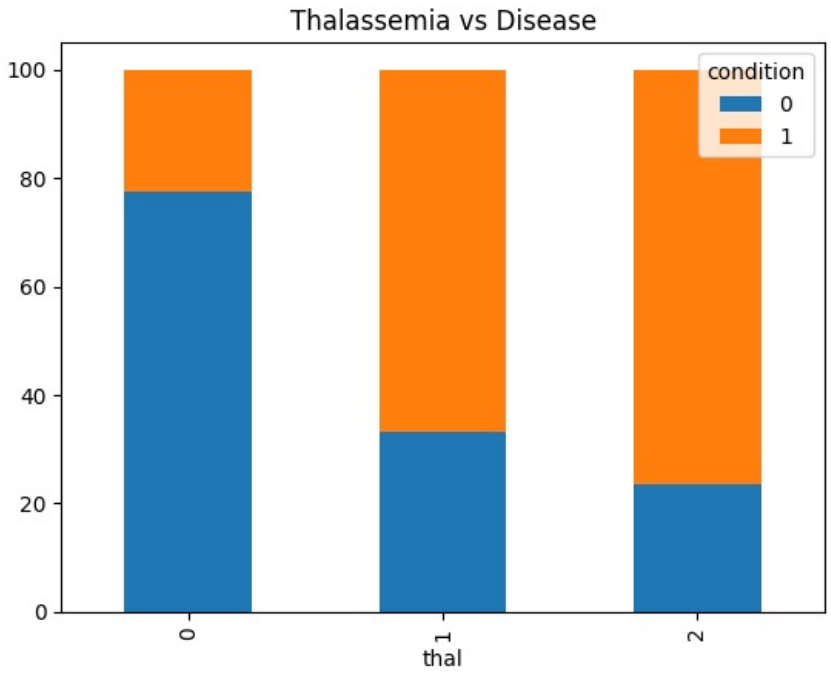


13. Thalassemia vs Disease

- Cross-tabulate thal and target
- Convert to percentage
- Plot stacked bar chart

In [18]:
```
#13. Thalassemia vs Disease
thal_tab = pd.crosstab(df["thal"], df["condition"], normalize="index") * 100
print(thal_tab)

thal_tab.plot(kind="bar", stacked=True)
plt.title("Thalassemia vs Disease")
plt.show()
```

```
condition          0          1
thal
0          77.439024  22.560976
1          33.333333  66.666667
2          23.478261  76.521739
```



14. Multi-Factor Risk Analysis

Find patients with:

- Age > 50
- Cholesterol > 240
- BP > 140
- Calculate percentage having disease . Use NumPy filtering

In [19]:
```
# 14. Multi-Factor Risk Analysis

high_risk = df[
    (df["age"] > 50) &
    (df["chol"] > 240) &
    (df["trestbps"] > 140)
]

percentage = (high_risk["condition"].sum() / len(high_risk)) * 100
print("High Risk Disease Percentage:", percentage)
```

```
High Risk Disease Percentage: 66.66666666666666
```

15. Create Risk Score (Custom Analysis)

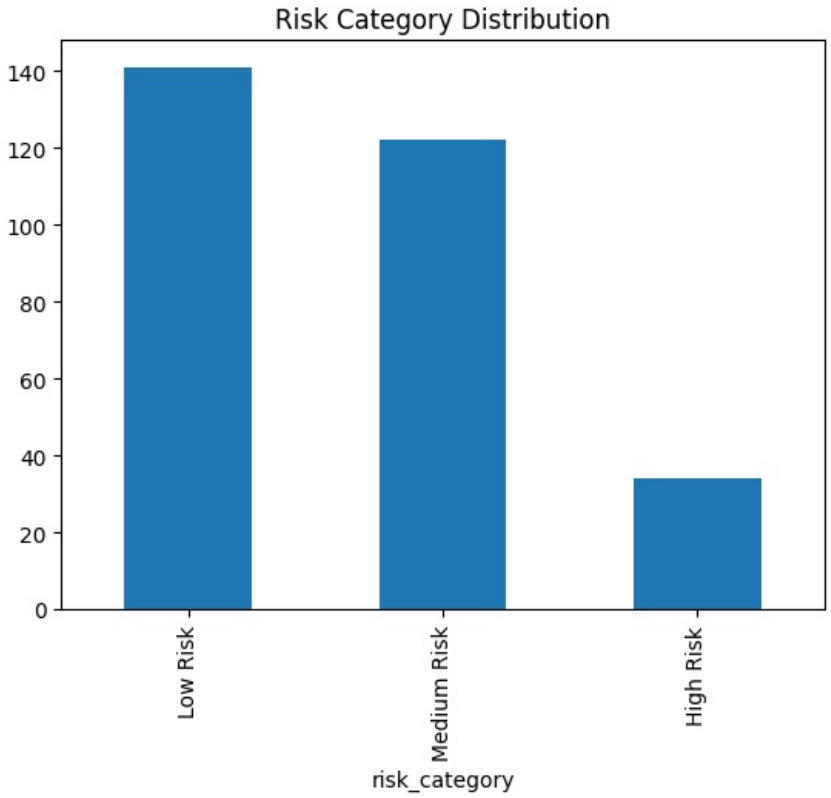
- Create new column: risk_score = (chol/200) + (trestbps/120) + (oldpeak)
- Classify patients as:

  ```
  Low Risk

  Medium Risk

  High Risk
  ```
- Visualize distribution

In [20]:
```
# 15. Create Risk Score (Feature Engineering)

df["risk_score"] = (df["chol"]/200) + \
                   (df["trestbps"]/120) + \
                   (df["oldpeak"])

df["risk_score"] = df["risk_score"].fillna(0)

conditions = [
    (df["risk_score"] < 3),
    (df["risk_score"] >= 3) & (df["risk_score"] < 5),
    (df["risk_score"] >= 5)
```

```
]

choices = ["Low Risk", "Medium Risk", "High Risk"]

df["risk_category"] = np.select(conditions, choices, default="Low Risk")

print(df["risk_category"].value_counts())

df["risk_category"].value_counts().plot(kind="bar")
plt.title("Risk Category Distribution")
plt.show()
```

```
risk_category
Low Risk       141
Medium Risk    122
High Risk       34
Name: count, dtype: int64
```



Insights

1. **Does cholesterol strongly impact heart disease?** → No, because in the correlation analysis 'chol' showed a relatively low correlation value with the 'condition'.
2. **Is male population more vulnerable?** → Yes, because disease percentage in males is more than females.
3. **Does exercise-induced angina significantly increase risk?** → Yes, since patients with `exang = 1` had higher disease occurrence in the analysis.
4. **Which feature has strongest correlation with disease?** → `oldpeak` has the strongest correlation because it showed the highest correlation value with the condition.