Modules

In [148]:

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

1.Load and Explore the Dataset

In [122]:

```
df=pd.read_csv("Heart Disease UCI.csv")
df
```

Out[122]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slop
0	69	1	0	160	234	1	2	131	0	0.1	
1	69	0	0	140	239	0	0	151	0	1.8	
2	66	0	0	150	226	0	0	114	0	2.6	
3	65	1	0	138	282	1	2	174	0	1.4	
4	64	1	0	110	211	0	2	144	1	1.8	
...	...	...	...	...	...	...	...	...	...	...	
292	40	1	3	152	223	0	0	181	0	0.0	
293	39	1	3	118	219	0	0	140	0	1.2	
294	35	1	3	120	198	0	0	130	1	1.6	
295	35	0	3	138	183	0	0	182	0	1.4	
296	35	1	3	126	282	0	2	156	1	0.0	

297 rows × 14 columns

In [123]:

```
df.head(10)
```

Out[123]:

	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

In [124]:

```
df.dtypes
```

Out[124]:

```
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
```

In [125]:

```
print(df.isnull().sum())
```

```
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
```

In [126]:

```
print(df.describe())
```

```
              age         sex          cp    trestbps        chol
fbs  \
count  297.000000  297.000000  297.000000  297.000000  297.000000  297.000
000
mean    54.542088    0.676768    2.158249  131.693603  247.350168    0.144
781
std      9.049736    0.468500    0.964859   17.762806   51.997583    0.352
474
min     29.000000    0.000000    0.000000   94.000000  126.000000    0.000
000
25%     48.000000    0.000000    2.000000  120.000000  211.000000    0.000
000
50%     56.000000    1.000000    2.000000  130.000000  243.000000    0.000
000
75%     61.000000    1.000000    3.000000  140.000000  276.000000    0.000
000
max     77.000000    1.000000    3.000000  200.000000  564.000000    1.000
000

          restecg     thalach       exang     oldpeak       slope
ca  \
count  297.000000  297.000000  297.000000  297.000000  297.000000  297.000
000
mean     0.996633  149.599327    0.326599    1.055556    0.602694    0.676
768
std      0.994914   22.941562    0.469761    1.166123    0.618187    0.938
965
min      0.000000   71.000000    0.000000    0.000000    0.000000    0.000
000
25%      0.000000  133.000000    0.000000    0.000000    0.000000    0.000
000
50%      1.000000  153.000000    0.000000    0.800000    1.000000    0.000
000
75%      2.000000  166.000000    1.000000    1.600000    1.000000    1.000
000
max      2.000000  202.000000    1.000000    6.200000    2.000000    3.000
000

             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

In [127]:

```
#count males and females
gender_counts = df['sex'].value_counts()
print("counts:\n",gender_counts)
```

```
counts:
 sex
1    201
0     96
Name: count, dtype: int64
```

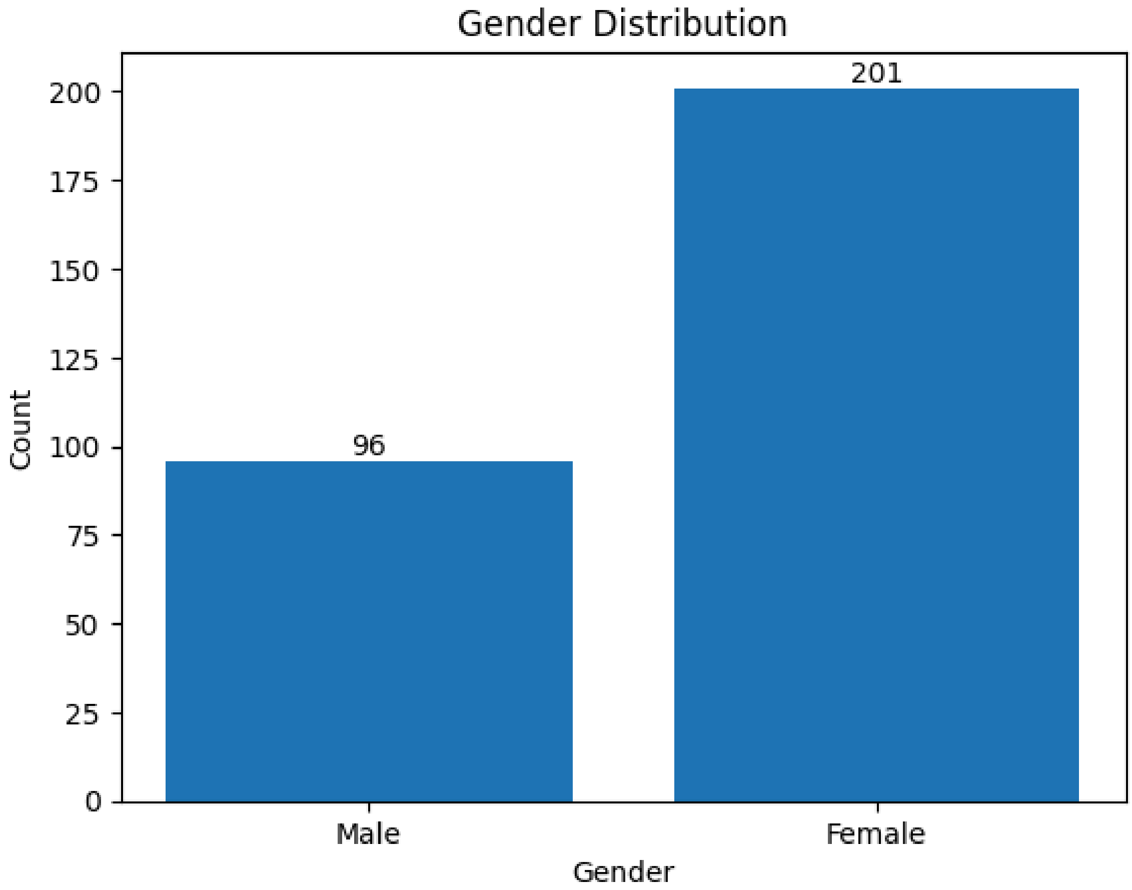
In [128]:

```
#Calculate percentage distribution using numpy
gender_percent = (gender_counts.values / np.sum(gender_counts.values)) *
print("\nGender percentage:\n", gender_percent)
```

```
Gender percentage:
 [67.67676768 32.32323232]
```

In [129]:

```
#Plot a bar chart
plt.bar(gender_counts.index, gender_counts.values, tick_label=['Female','
plt.title("Gender Distribution")
plt.xlabel("Gender")
plt.ylabel("Count")
plt.bar_label(bars)
plt.show()
```



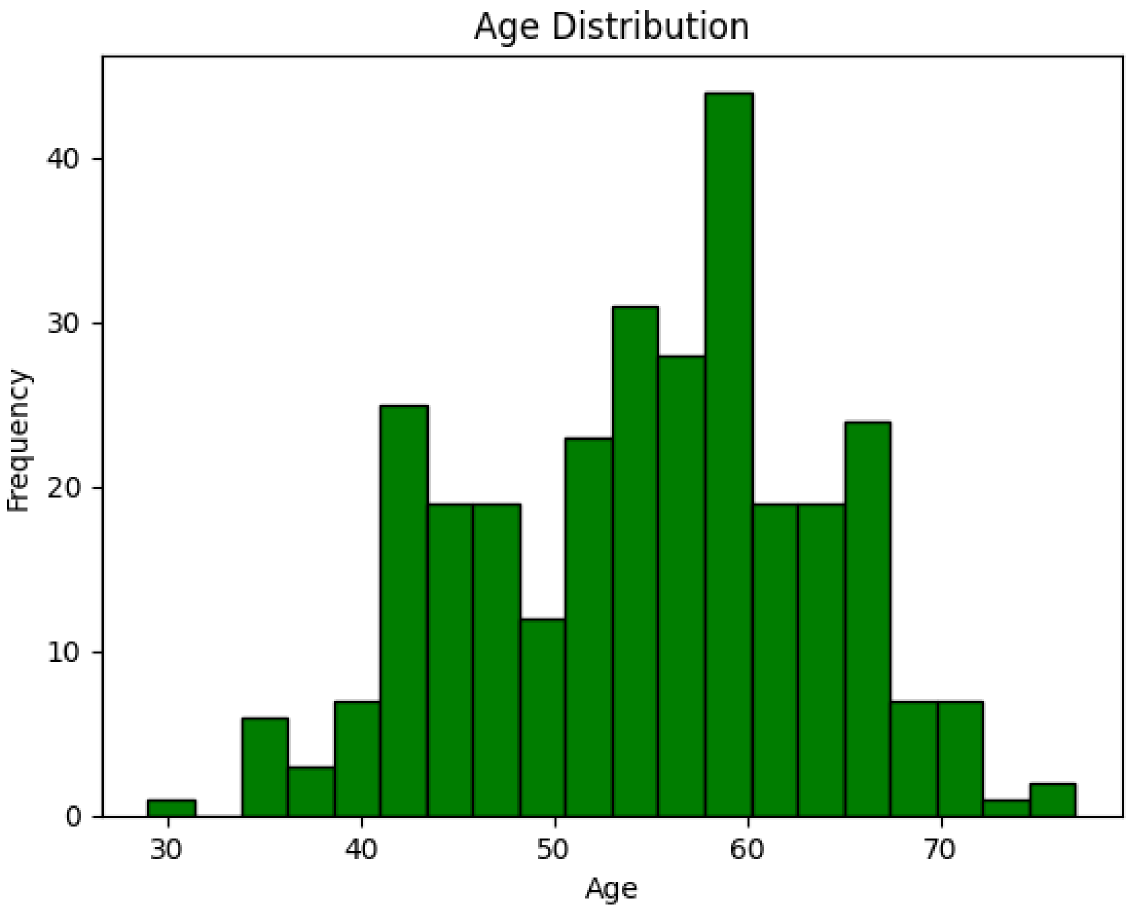
3.Age Analysis

In [130]:

```
# Age statistics
print("Minimum age:", df['age'].min())
print("Maximum age:", df['age'].max())
print("Mean age:", df['age'].mean())
print("Median age:", df['age'].median())
```

```
Minimum age: 29
Maximum age: 77
Mean age: 54.54208754208754
Median age: 56.0
```

In [131]:
```
# Histogram
plt.hist(df['age'], bins=20, color='green', edgecolor='black')
plt.title("Age Distribution")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```

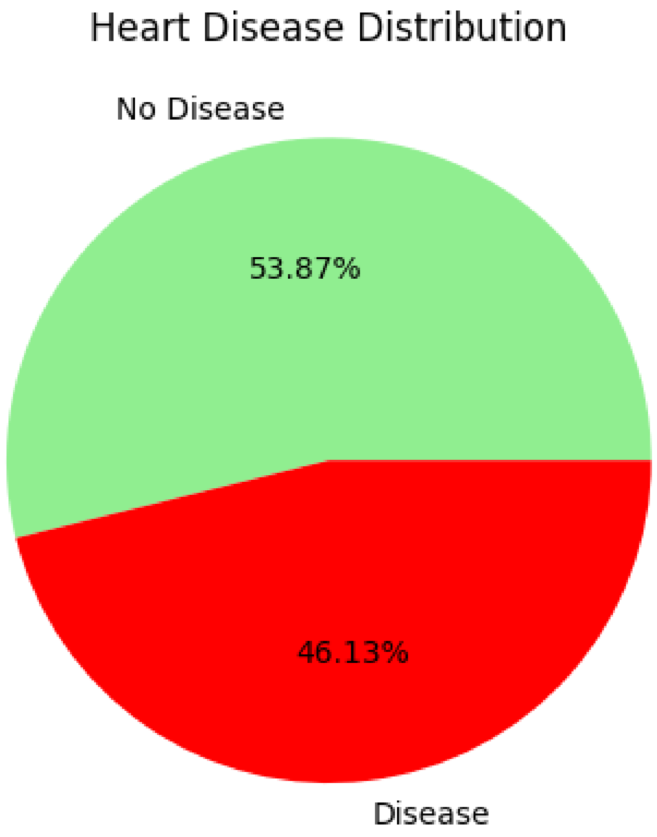


4.Target Variable Analysis

In [132]:
```
# Count patients with and without disease
target_counts = df['condition'].value_counts()
print("Target counts:\n", target_counts)
```

```
Target counts:
 condition
0    160
1    137
Name: count, dtype: int64
```

In [133]:
```
# Pie chart
plt.pie(target_counts, labels=['No Disease','Disease'], autopct='%1.2f%%'
plt.title("Heart Disease Distribution")
plt.show()
```



In [134]:

```
# Disease percentage
disease_percentage = (target_counts[1] / len(df)) * 100
print("Disease percentage:", disease_percentage)
```

```
Disease percentage: 46.12794612794613
```

5.Correlation Between Age and Cholesterol

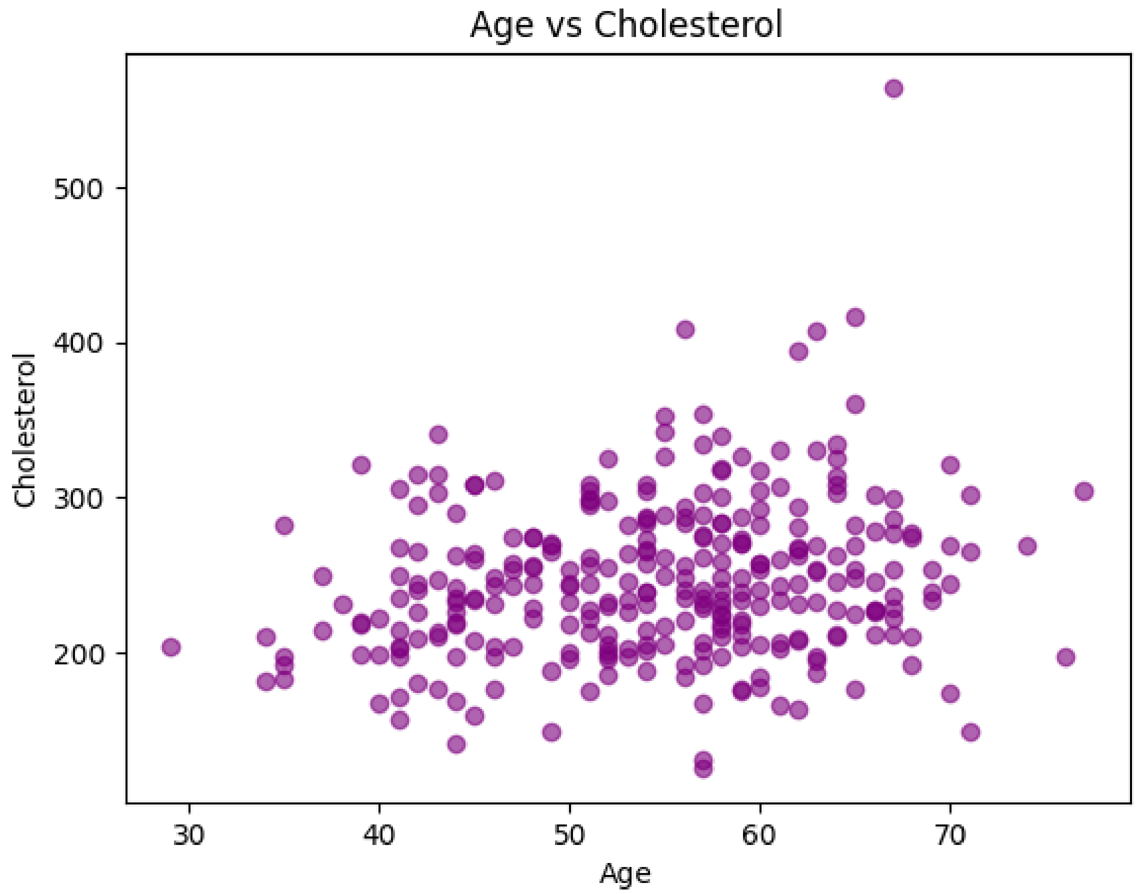
In [135]:

```
# Correlation
corr_value = df[['age','chol']].corr().iloc[0,1]
print("Correlation between Age and Cholesterol:", corr_value)
```

```
Correlation between Age and Cholesterol: 0.2026435458466271
```

In [136]:

```
# Scatter plot
plt.scatter(df['age'], df['chol'], alpha=0.6, color='purple')
plt.title("Age vs Cholesterol")
plt.xlabel("Age")
plt.ylabel("Cholesterol")
plt.show()
```



6.Chest Pain Type vs Disease

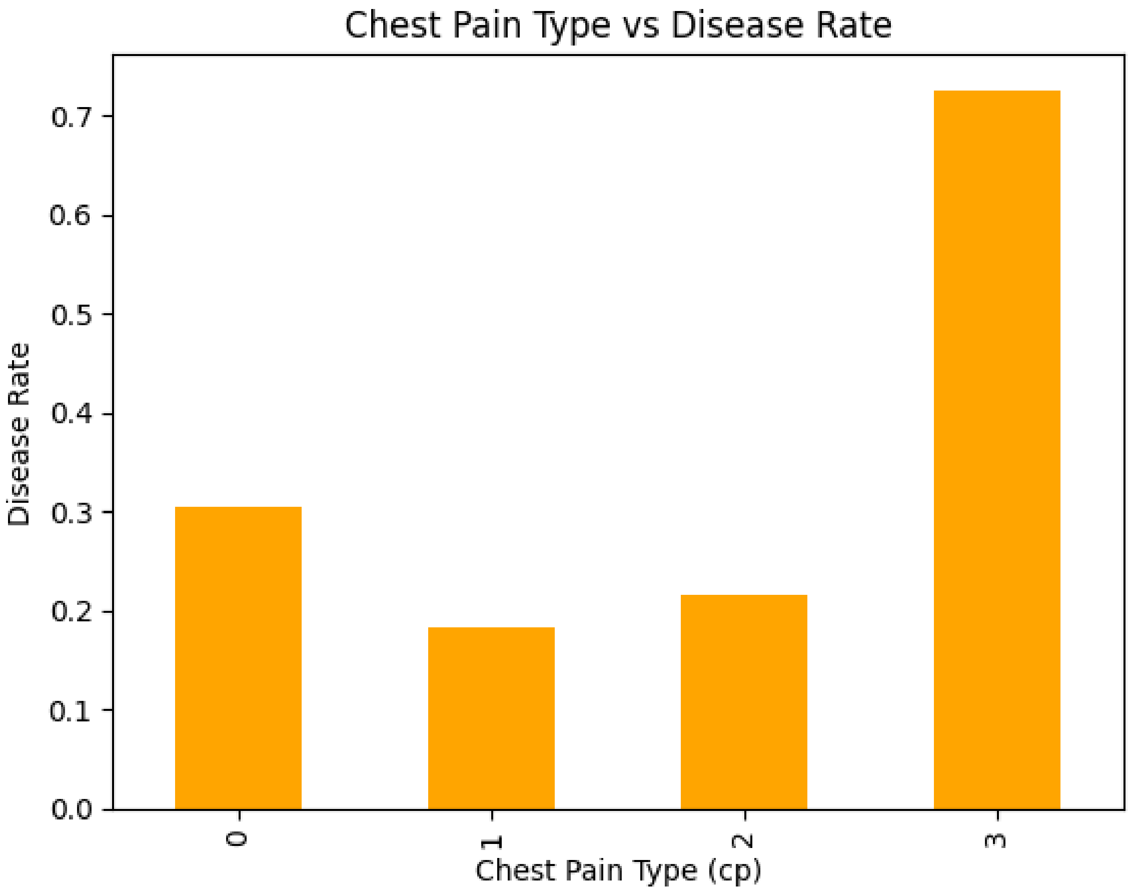
In [137]:

```
# Group by chest pain type (cp) and calculate disease rate
cp_disease_rate = df.groupby('cp')['condition'].mean()
print("Disease rate by chest pain type:\n", cp_disease_rate)
```

```
Disease rate by chest pain type:
 cp
0    0.304348
1    0.183673
2    0.216867
3    0.725352
Name: condition, dtype: float64
```

In [138]:

```
# Grouped bar chart
cp_disease_rate.plot(kind='bar', color='orange')
plt.title("Chest Pain Type vs Disease Rate")
plt.xlabel("Chest Pain Type (cp)")
plt.ylabel("Disease Rate")
plt.show()
```

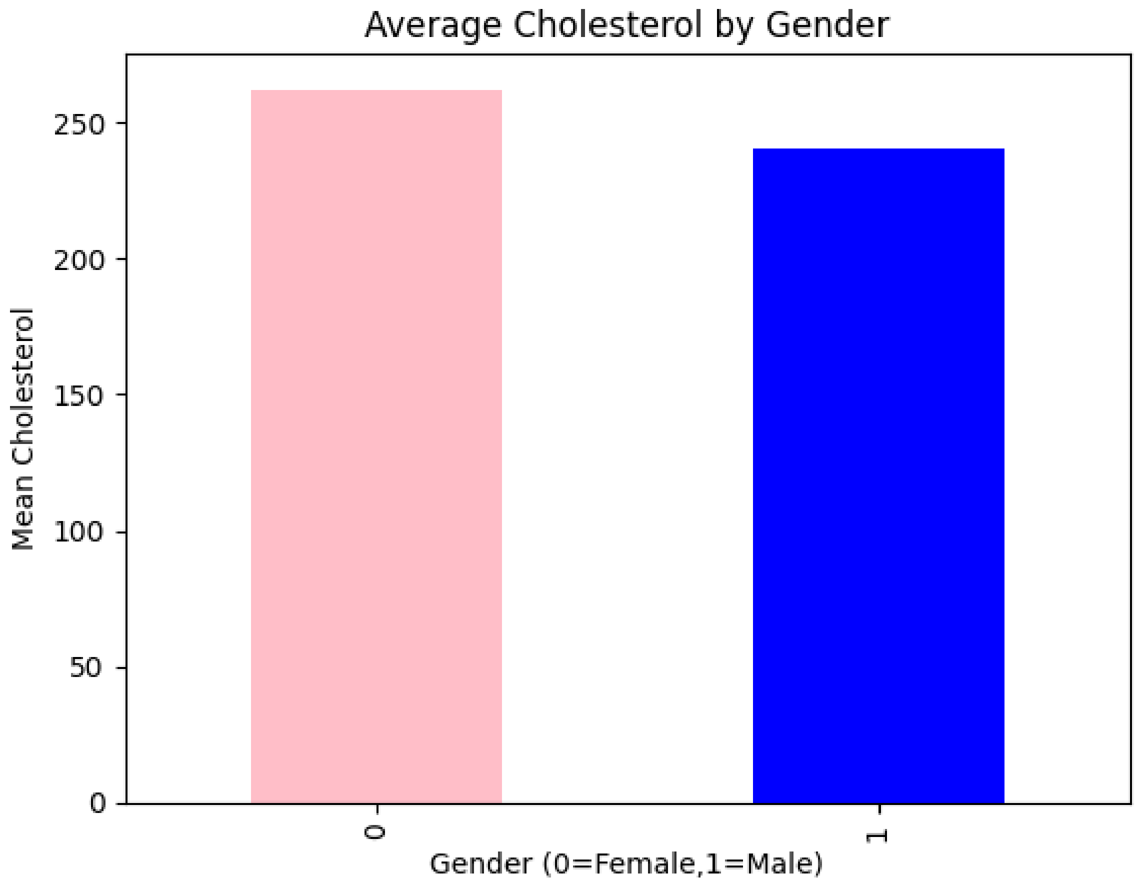


7.Average Cholesterol by Gender

In [139]:
```
avg_chol_gender = df.groupby('sex')['chol'].mean()
print("Average cholesterol by gender:\n", avg_chol_gender)
```

```
Average cholesterol by gender:
 sex
0    262.229167
1    240.243781
Name: chol, dtype: float64
```

In [140]:
```
avg_chol_gender.plot(kind='bar', color=['pink','blue'])
plt.title("Average Cholesterol by Gender")
plt.xlabel("Gender (0=Female,1=Male)")
plt.ylabel("Mean Cholesterol")
plt.show()
```



8.Resting Blood Pressure Analysis

In [141]:
```
avg_bp = df['trestbps'].mean()
print("Average BP:", avg_bp)
```

```
Average BP: 131.69360269360268
```

In [142]:
```
high_bp_patients = df[df['trestbps'] > 140]
print("Patients with BP > 140:", len(high_bp_patients))
```

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

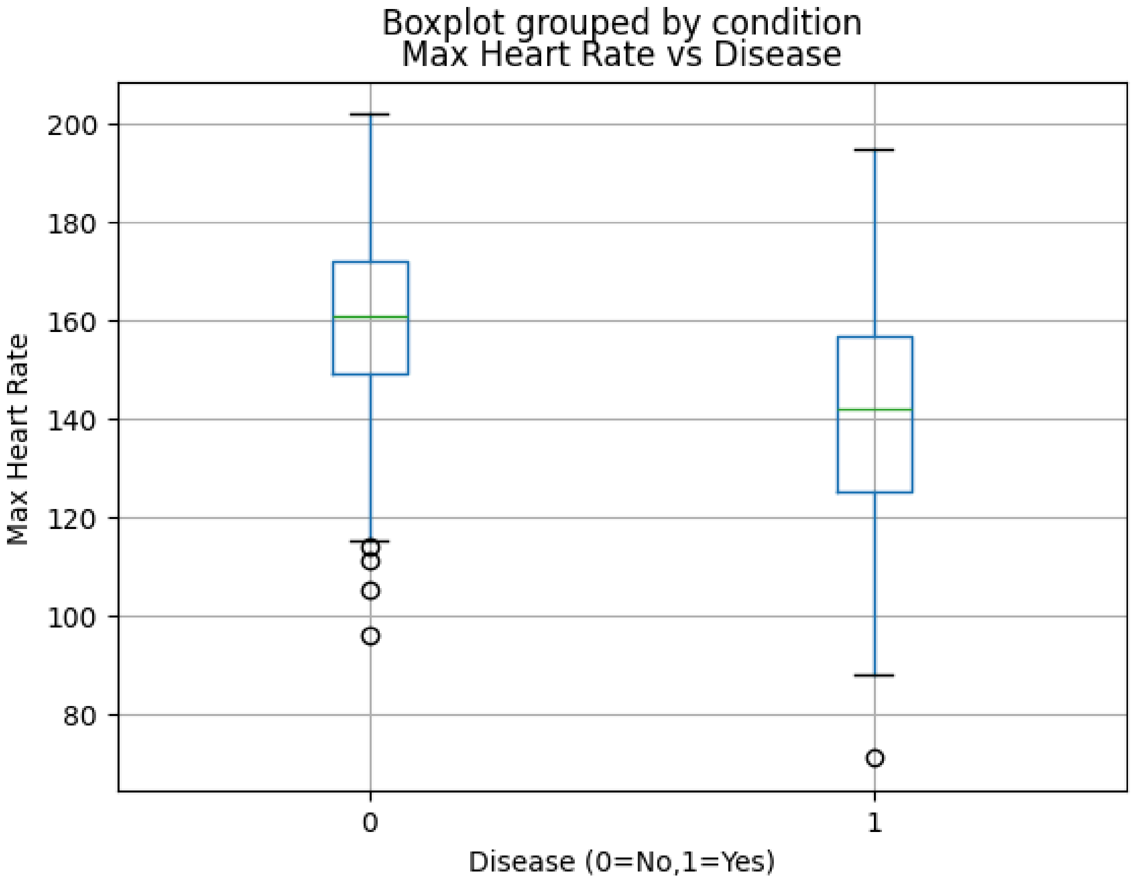
In [143]:
```
# Compare disease presence in high BP group
high_bp_disease_rate = high_bp_patients['condition'].mean()
print("Disease rate in high BP group:", high_bp_disease_rate)
```

```
Disease rate in high BP group: 0.5909090909090909
```

9.Maximum Heart Rate vs Disease

In [144]:
```
thalach_comparison = df.groupby('condition')['thalach'].mean()
print("Average thalach by disease presence:\n", thalach_comparison)
# Boxplot
df.boxplot(column='thalach', by='condition')
plt.title("Max Heart Rate vs Disease")
plt.xlabel("Disease (0=No,1=Yes)")
plt.ylabel("Max Heart Rate")
plt.show()
```

```
Average thalach by disease presence:
 condition
0    158.581250
1    139.109489
Name: thalach, dtype: float64
```

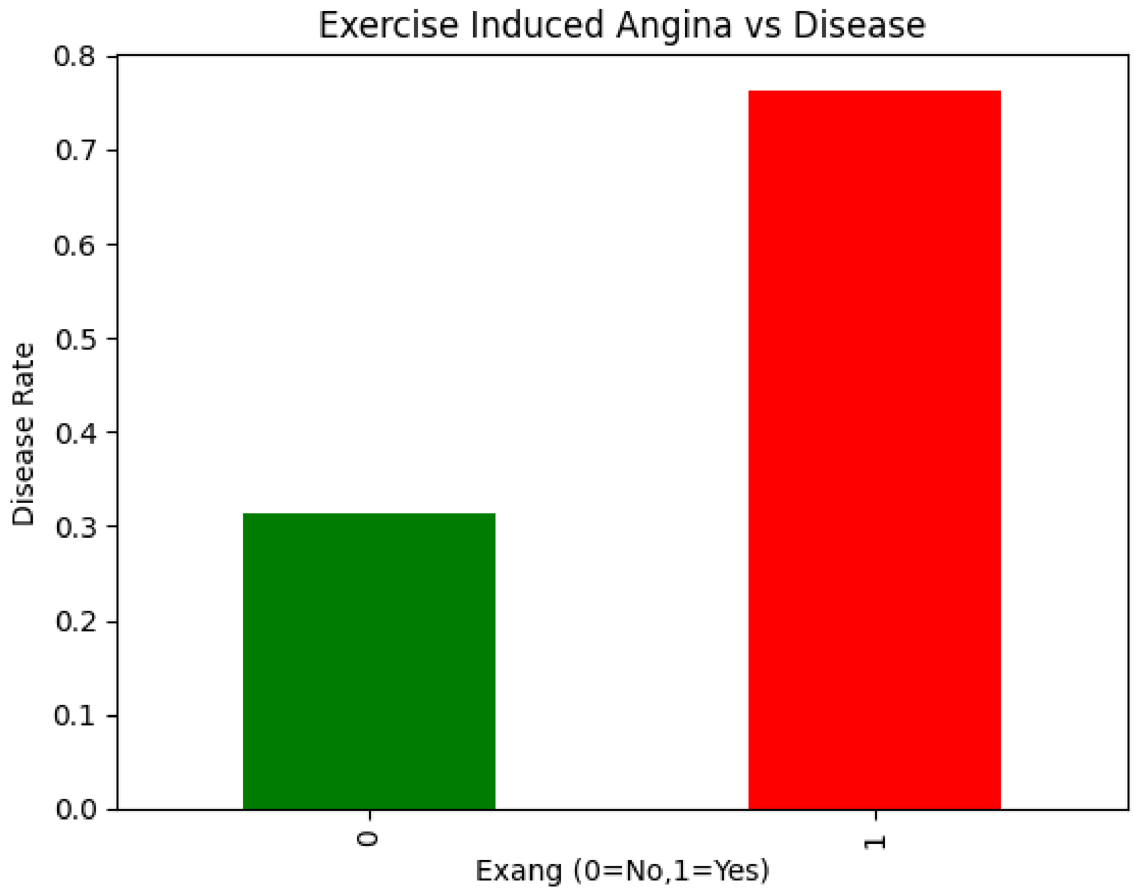


10.Exercise Induced Angina Impact

In [93]:

```
exang_disease_rate = df.groupby('exang')['condition'].mean()
print("Disease percentage by exang:\n", exang_disease_rate * 100)
exang_disease_rate.plot(kind='bar', color=['green','red'])
plt.title("Exercise Induced Angina vs Disease")
plt.xlabel("Exang (0=No,1=Yes)")
plt.ylabel("Disease Rate")
plt.show()
```

```
Disease percentage by exang:
 exang
0    31.50000
1    76.28866
Name: condition, dtype: float64
```

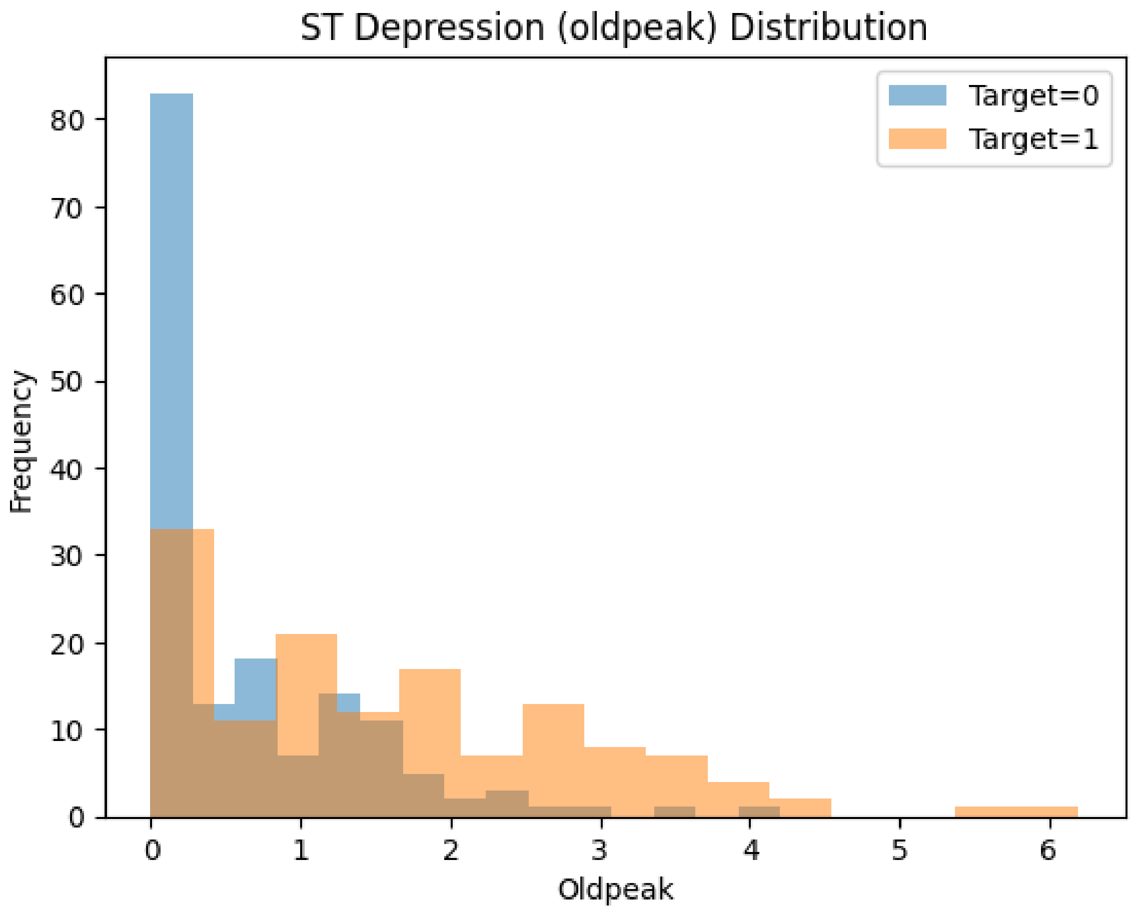


11.ST Depression (oldpeak) Analysis

In [147]:

```
mean_oldpeak = df.groupby('condition')['oldpeak'].mean()
print("Mean oldpeak by target:\n", mean_oldpeak)
# Histogram
for t in df['condition'].unique():
    plt.hist(df[df['condition']==t]['oldpeak'], bins=15, alpha=0.5, label
plt.title("ST Depression (oldpeak) Distribution")
plt.xlabel("Oldpeak")
plt.ylabel("Frequency")
plt.legend()
plt.show()
```

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

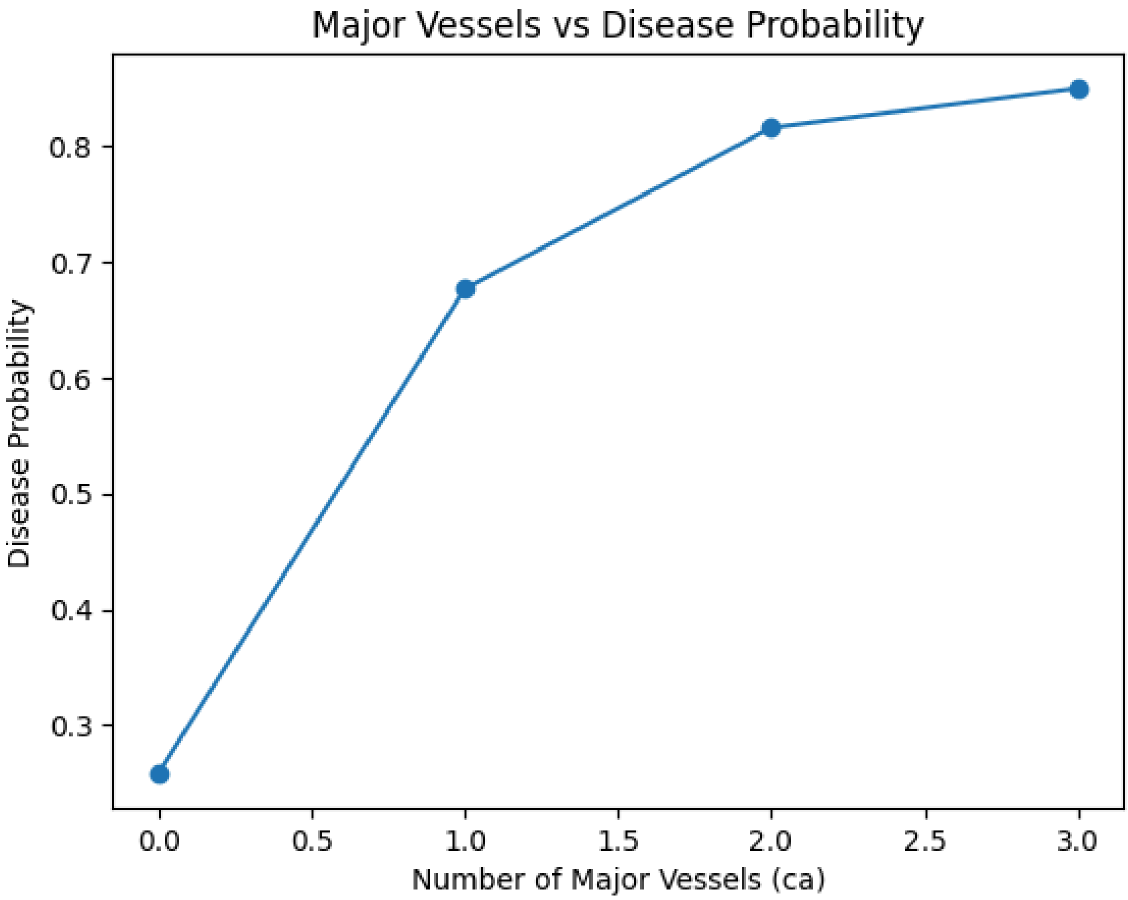


12.Number of Major Vessels (ca) Impact

In [97]:

```
ca_disease_prob = df.groupby('ca')['condition'].mean()
print("Disease probability by number of vessels:\n", ca_disease_prob)
ca_disease_prob.plot(kind='line', marker='o')
plt.title("Major Vessels vs Disease Probability")
plt.xlabel("Number of Major Vessels (ca)")
plt.ylabel("Disease Probability")
plt.show()
```

```
Disease probability by number of vessels:
 ca
0    0.258621
1    0.676923
2    0.815789
3    0.850000
Name: condition, dtype: float64
```

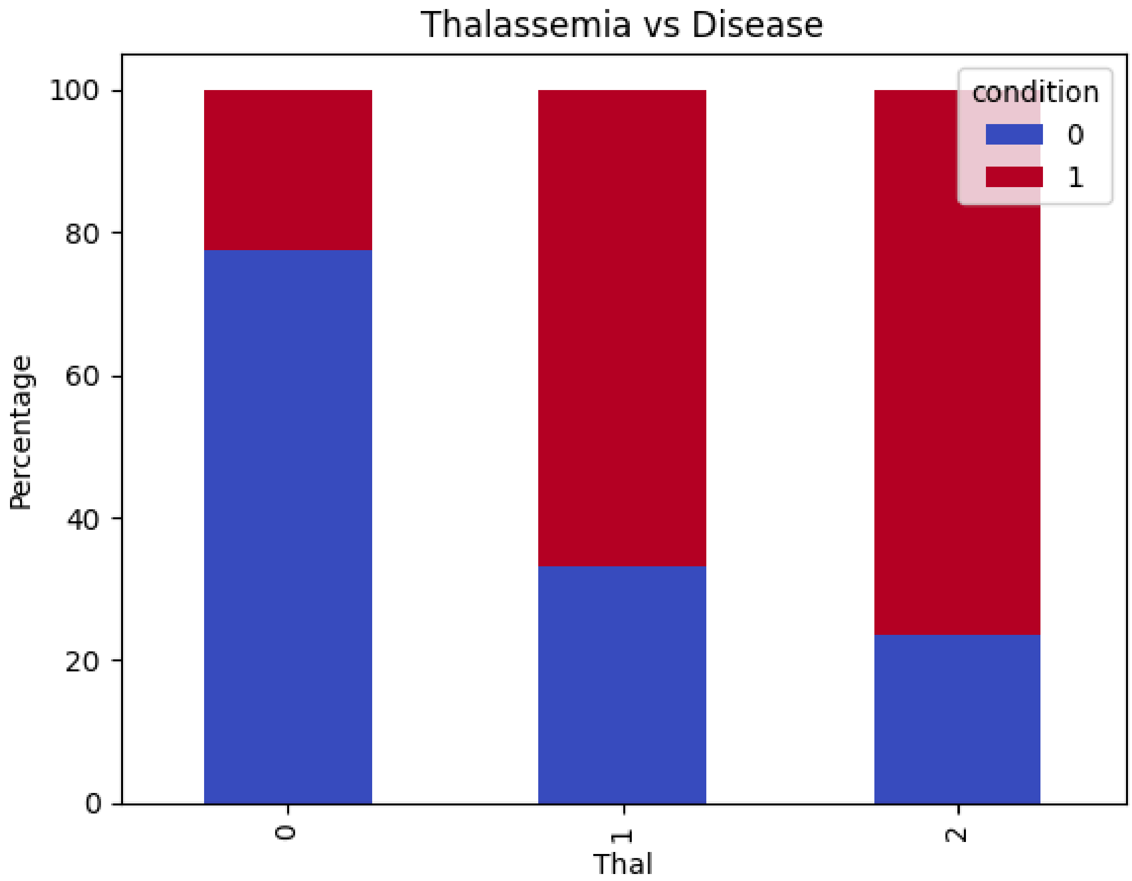


13.Thalassemia vs Disease

In [99]:

```
thal_target_ct = pd.crosstab(df['thal'], df['condition'], normalize='inde
print("Thal vs Disease (%):\n", thal_target_ct)
thal_target_ct.plot(kind='bar', stacked=True, colormap='coolwarm')
plt.title("Thalassemia vs Disease")
plt.xlabel("Thal")
plt.ylabel("Percentage")
plt.show()
```

```
Thal vs Disease (%):
 condition          0          1
thal
0          77.439024  22.560976
1          33.333333  66.666667
2          23.478261  76.521739
```



14.Multi-Factor Risk Analysis

In [102]:

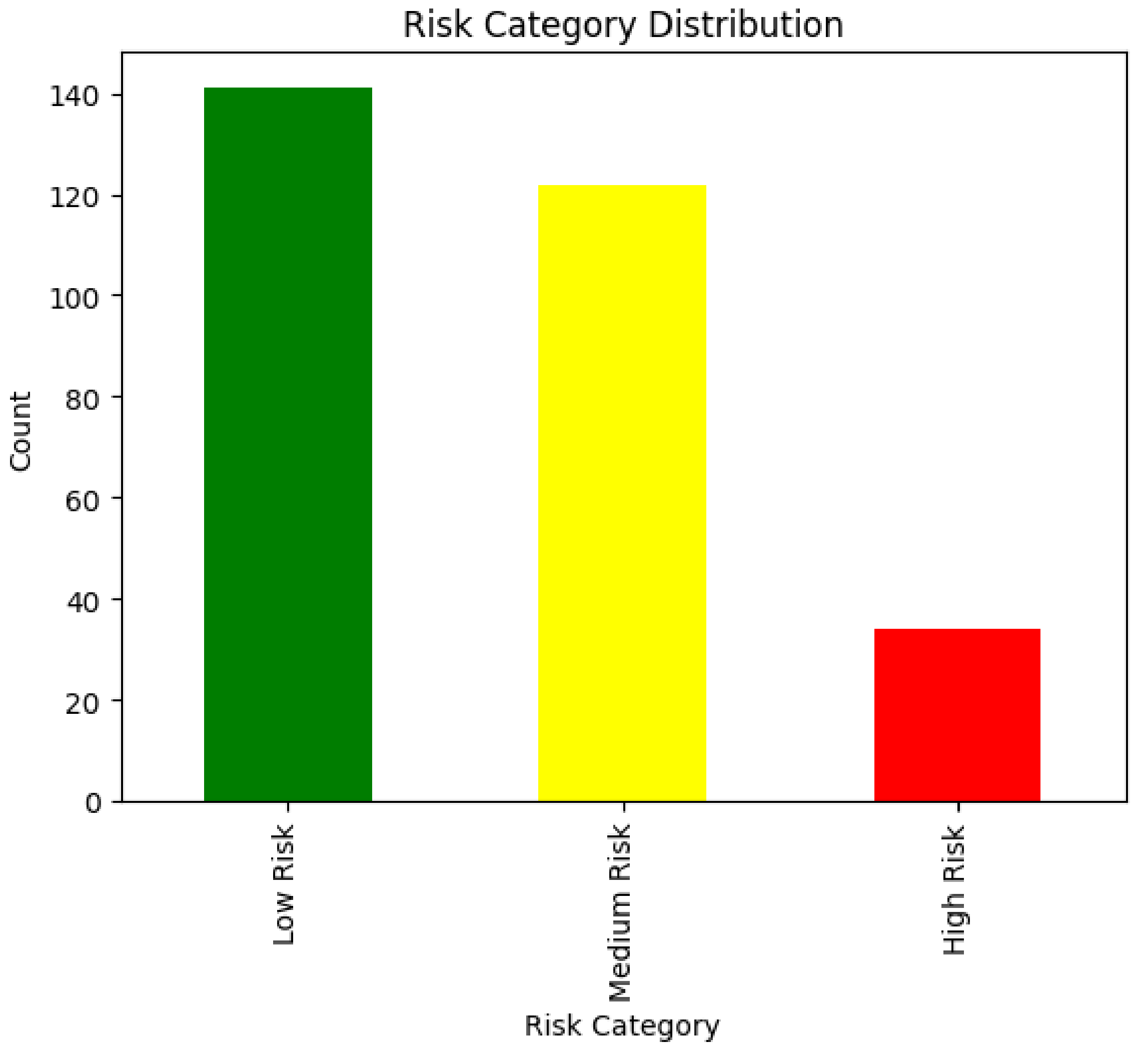
```
# Filter patients
risk_patients = df[(df['age'] > 50) & (df['chol'] > 240) & (df['trestbps'
risk_disease_percentage = (risk_patients['condition'].mean()) * 100
print("Disease percentage in high-risk group:", risk_disease_percentage)
```

```
Disease percentage in high-risk group: 66.66666666666666
```

15.Create Risk Score (Custom Analysis)

In [103]:

```
# Create risk score
df['risk_score'] = (df['chol']/200) + (df['trestbps']/120) + df['oldpeak'
# Classify patients
def classify_risk(score):
    if score < 3: return "Low Risk"
    elif score < 5: return "Medium Risk"
    else: return "High Risk"
df['risk_category'] = df['risk_score'].apply(classify_risk)
# Visualize distribution
df['risk_category'].value_counts().plot(kind='bar', color=['green','yello
plt.title("Risk Category Distribution")
plt.xlabel("Risk Category")
plt.ylabel("Count")
plt.show()
```


Risk Category Distribution
Count
140
120
100
80
60
40
20
0
Low Risk
Medium Risk
High Risk
Risk Category