Heart Disease Risk Analysis & Predictive Insights

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]:
```
import pandas as pd  #Load dataset
import matplotlib.pyplot as plt
df=pd.read_csv(r"C:\Users\Administrator\Desktop\python\Heart Disease UCI.csv")
print("\nDataset\n\n\n",df.head(10))
print("\nCount of Null Values per column\n\n\n",df.isna().sum()) #checking null va

print("\nData Types of Each Column\n\n\n")
print(df.dtypes)  #Validating data types
print("\nStatistical Summary\n\n\n",df.describe())  # summary statistics
```

Dataset

	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

Count of Null Values per column

```
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
```

Data Types of Each Column

```
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

Statistical Summary
```

	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

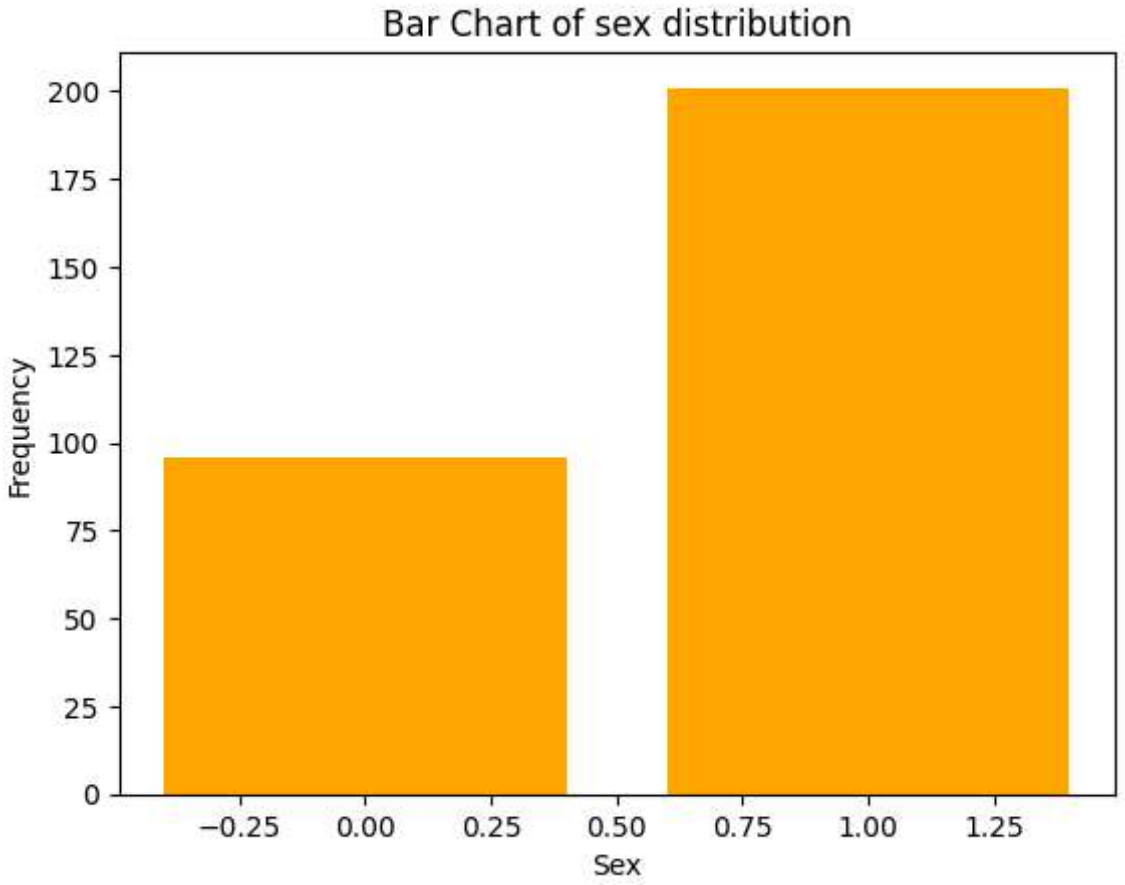
In [150]:
```
# Count number of males and females
sex_count=df['sex'].value_counts()
sex_count
```

Out[150]:
```
sex
1    201
0     96
Name: count, dtype: int64
```

In [3]:
```
# Calculate percentage distribution using NumPy

per = df.groupby('sex').size() / len(df) * 100
print("Percentage Distribution : ",per)
```

```
Percentage Distribution :  sex
0    32.323232
1    67.676768
dtype: float64
```

In [151]:
```
# Plot a bar chart
plt.bar(sex_count.index, sex_count.values, color="orange")
plt.title("Bar Chart of sex distribution")
plt.xlabel("Sex")
plt.ylabel("Frequency")
plt.show()
```



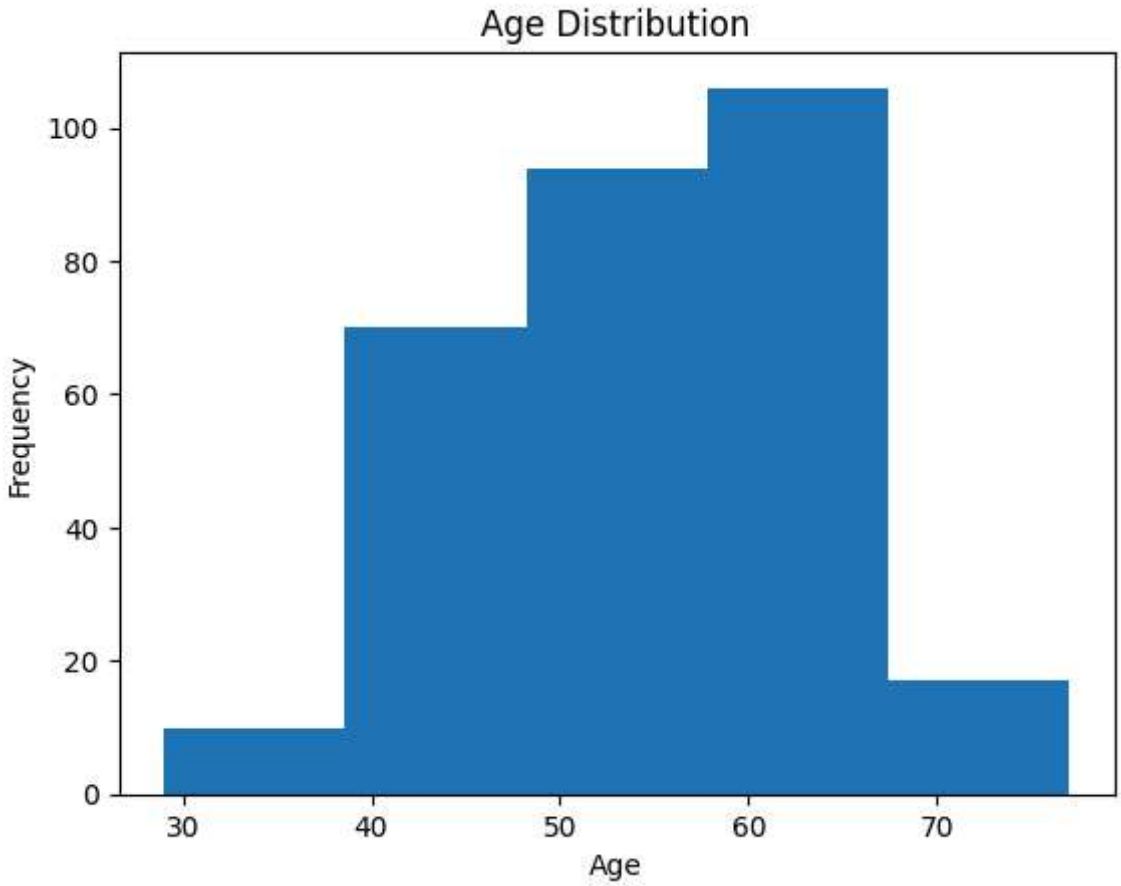
3.Age Analysis

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

In [4]:
```
print("Minimum Age=",df['age'].min()) #minimum age
print("Maximum Age=",df['age'].max()) #maximum age
print("Mean of Age=",df['age'].mean()) #mean age
print("Median of Age=",df['age'].median()) #minimum age

plt.hist(df['age'],bins=5)
plt.title("Age Distribution")
plt.xlabel("Age")
```

```
plt.ylabel("Frequency")
plt.show()
```

```
Minimum Age= 29
Maximum Age= 77
Mean of Age= 54.54208754208754
Median of Age= 56.0
```



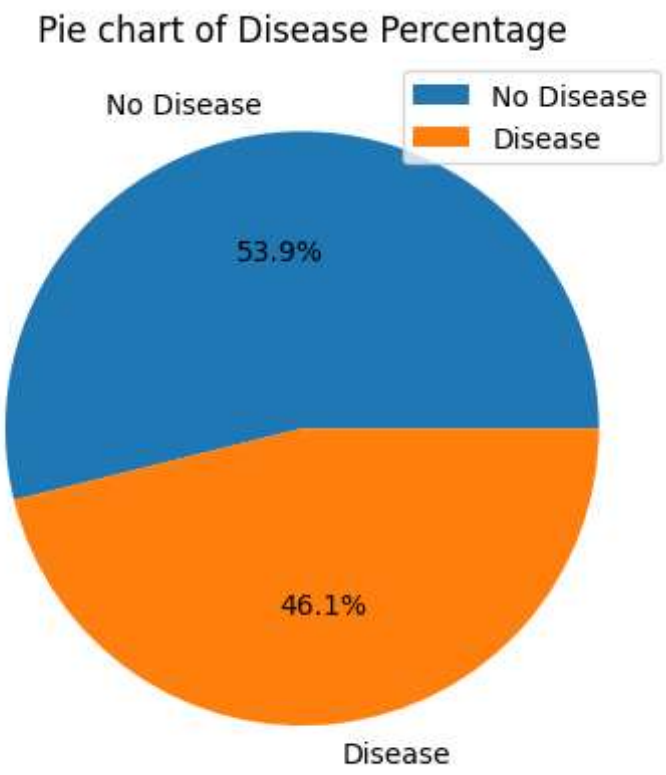
4. Target Variable Analysis

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

In [5]:
```
# Count number of patients with and without heart disease

counts=df['condition'].value_counts()
print("patients with and without heart disease",counts)
```

```
patients with and without heart disease condition
0    160
1    137
Name: count, dtype: int64
```

In [154]:
```
# Plot pie chart
plt.pie(counts,labels=["No Disease","Disease"], autopct="%1.1f%%")
plt.title("Pie chart of Disease Percentage")
plt.legend()

plt.show()
```



In [9]:
```
# Calculate disease percentage
count=df['condition'].value_counts()
per=(count[1]/len(df))*100
print("Disease percentage:",per)
```

```
Disease percentage: 46.12794612794613
```

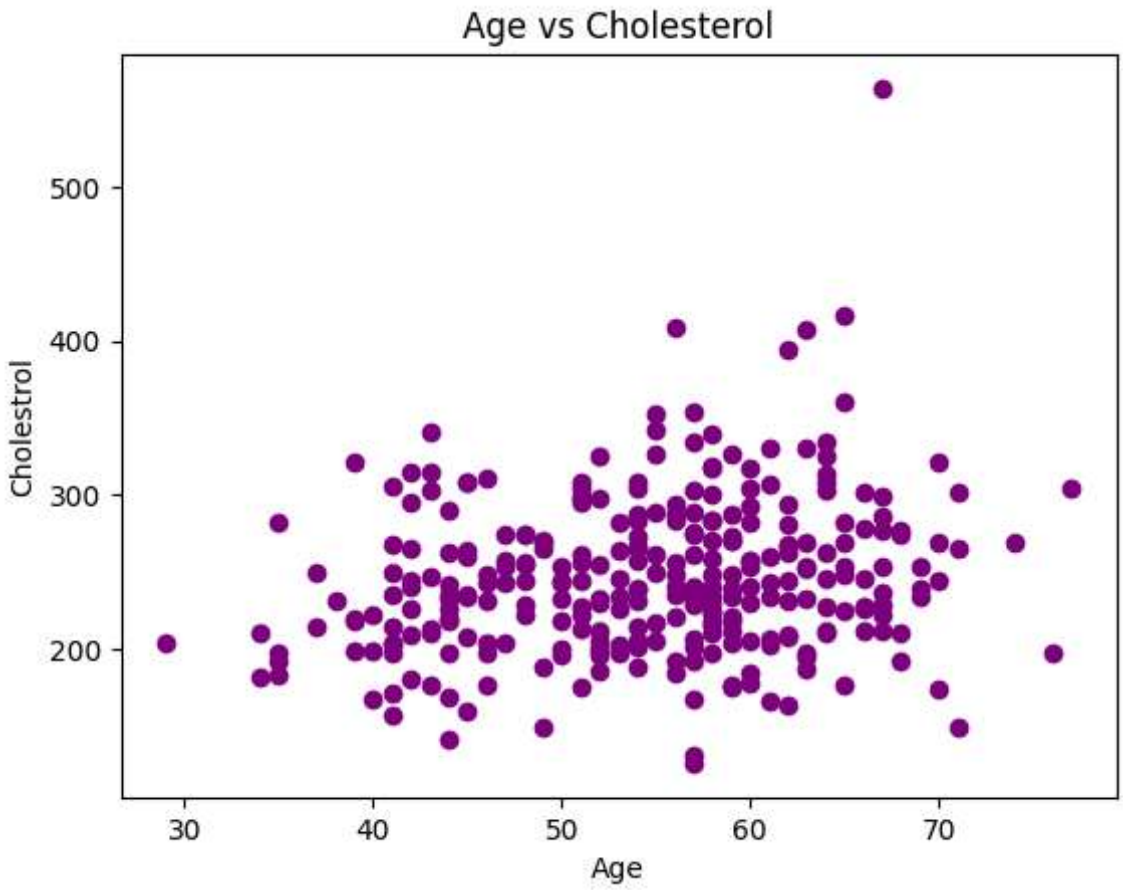
5. Correlation Between Age and Cholesterol

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

In [8]:
```
# Calculate correlation using df.corr()

print("Correlation between age and cholestrol \n:",df[['age','chol']].corr())
```

```
Correlation between age and cholestrol
:           age      chol
age    1.000000  0.202644
chol   0.202644  1.000000
```

In [158]:
```
# Plot scatter plot (Age vs Cholesterol)

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



In [21]:
```
# Interpret relationship

cor = df['age'].corr(df['chol'])
print("Correlation =",cor)

if cor > 0.5:
    print("Strong positive relationship")
elif cor > 0:
    print("Weak positive relationship")
elif cor > -0.5:
    print("Weak negative relationship")
else:
    print("Strong negative relationship")
```

```
Correlation = 0.20264354584662683
Weak positive relationship
```

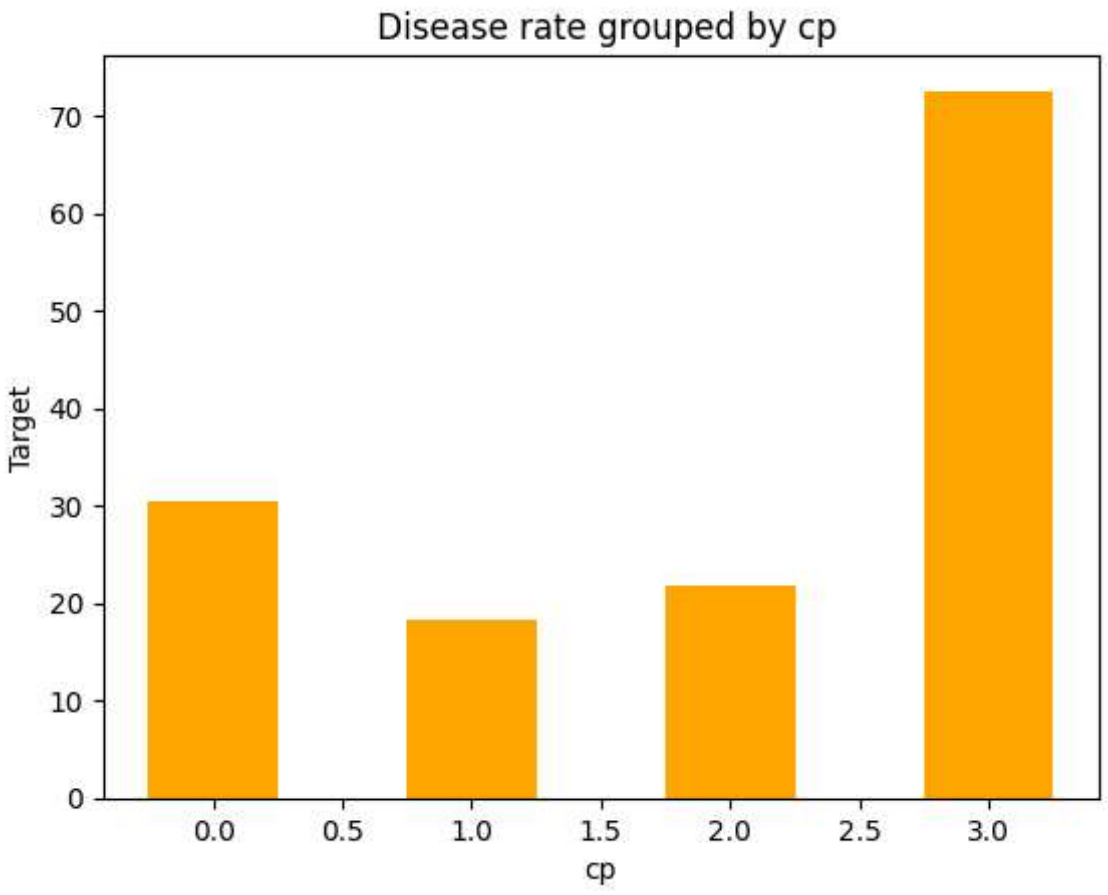
6. Chest Pain Type vs Disease

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

In [10]:
```
# Group by cp and calculate disease rate

grouped=df.groupby('cp')['condition'].mean()*100
print("cp and calculate disease rate:\n",grouped)
```

```
cp and calculate disease rate:
 cp
0    30.434783
1    18.367347
2    21.686747
3    72.535211
Name: condition, dtype: float64
```

In [160]:
```
# Plot grouped bar chart
plt.bar(grouped.index, grouped.values, color="orange", width=0.5)
plt.title(" Disease rate grouped by cp")
plt.xlabel("cp")
plt.ylabel("Target")
plt.show()
```



In [11]:
```
# Identify which chest pain type is most risky

most_risky_cp = grouped.idxmax()

print("Most risky cp type = ",most_risky_cp)
```

```
Most risky cp type =  3
```

7. Average Cholesterol by Gender

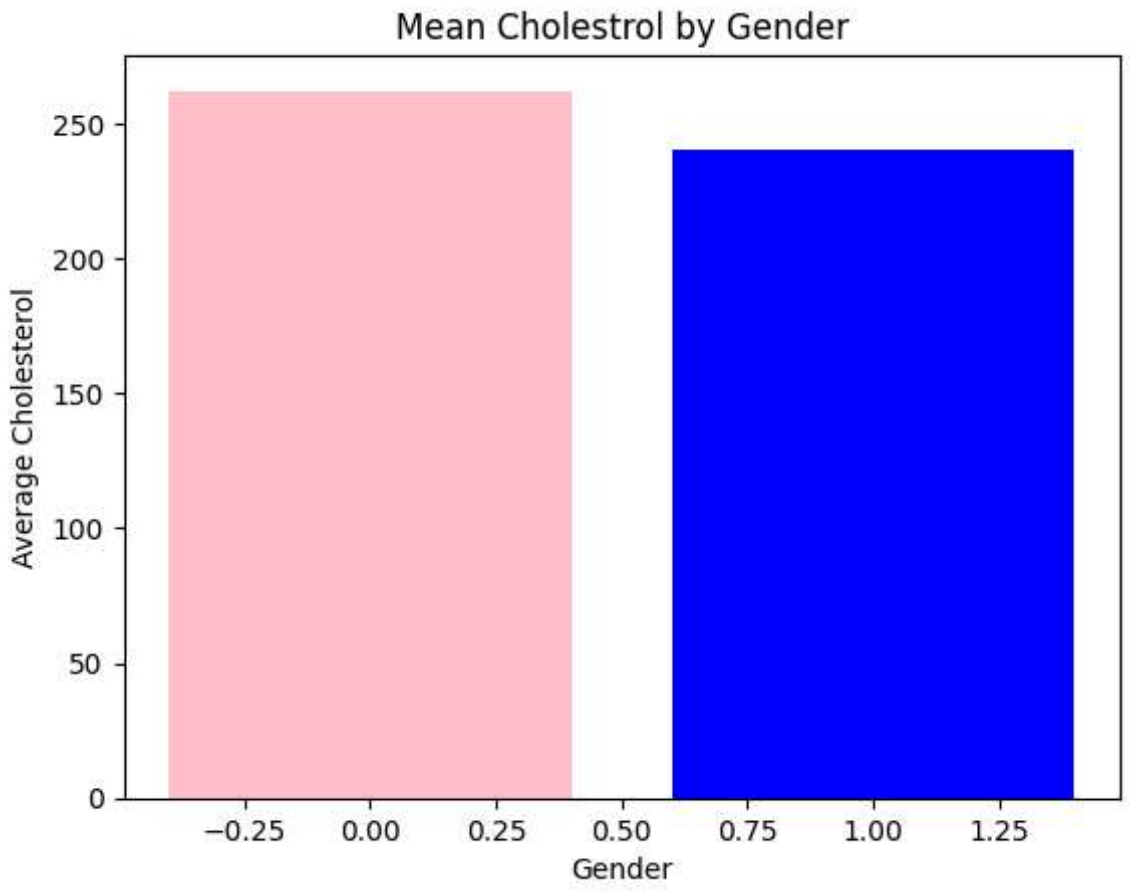
- Group by sex
- Calculate mean cholesterol
- Visualize using bar plot

In [12]:
```
# Group by sex
grp=df.groupby('sex')['chol'].mean()
```

```
grp
```

Out[12]:
```
sex
0    262.229167
1    240.243781
Name: chol, dtype: float64
```

In [17]:
```
#Calculate mean cholesterol Visualize using bar plot
avg_chol_by_gender=df.groupby('sex')['chol'].mean()
plt.bar(avg_chol_by_gender.index, avg_chol_by_gender.values, color=['pink','blue']
plt.title("Mean Cholestrol by Gender")
plt.xlabel("Gender")
plt.ylabel("Average Cholesterol")
plt.show()
```



8. Resting Blood Pressure Analysis

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

In [14]:
```
# Average BP
print("Average BP =",df['trestbps'].mean())
```

```
Average BP = 131.69360269360268
```

In [29]:
```
# Patients with BP > 140

high_bp=df[df['trestbps'] > 140]
```

```
print(high_bp)
print("count =",high_bp.count())
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0     69    1   0       160   234    1        2      131      0      0.1
2     66    0   0       150   226    0        0      114      0      2.6
5     64    1   0       170   227    0        2      155      0      0.6
6     63    1   0       145   233    1        2      150      0      2.3
8     60    0   0       150   240    0        0      171      0      0.9
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
263   50    1   3       150   243    0        2      128      0      2.6
264   50    1   3       144   200    0        2      126      1      0.9
277   45    1   3       142   309    0        2      147      1      0.0
285   43    1   3       150   247    0        0      171      0      1.5
292   40    1   3       152   223    0        0      181      0      0.0

     slope  ca  thal  condition
0        1   1     0          0
2        2   0     0          0
5        1   0     2          0
6        2   0     1          0
8        0   0     0          0
..     ...  ..   ...        ...
263      1   0     2          1
264      1   0     2          1
277      1   3     2          1
285      0   0     0          0
292      0   0     2          1

[66 rows x 14 columns]
count = age          66
sex          66
cp           66
trestbps     66
chol         66
fbs          66
restecg      66
thalach      66
exang        66
oldpeak      66
slope        66
ca           66
thal         66
condition    66
dtype: int64
```

In [30]:
```
# Compare disease presence in high BP group
print(high_bp['condition'].value_counts(normalize=True) * 100)
```

```
condition
1    59.090909
0    40.909091
Name: proportion, dtype: float64
```

9. Maximum Heart Rate vs Disease

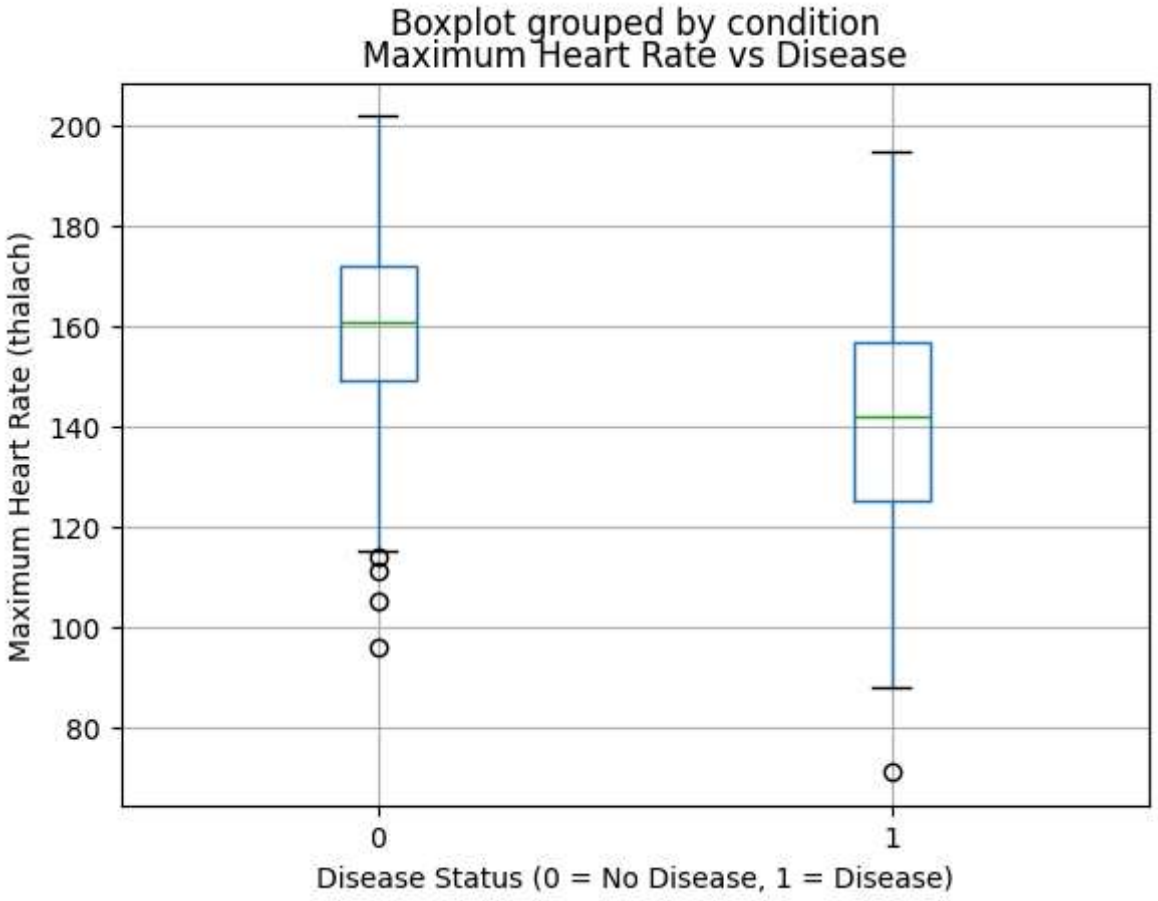
- Compare average thalach for:
  - Disease patients
  - Non-disease patients

- Plot boxplot

In [181]:
```
# Compare average thalach for:

print(df.groupby('condition')['thalach'].mean())
```

```
condition
0    158.581250
1    139.109489
Name: thalach, dtype: float64
```

In [5]:
```
df.boxplot(column='thalach', by='condition')
plt.title("Maximum Heart Rate vs Disease")

plt.xlabel("Disease Status")
plt.ylabel("Maximum Heart Rate")
plt.show()
```



10. Exercise Induced Angina Impact

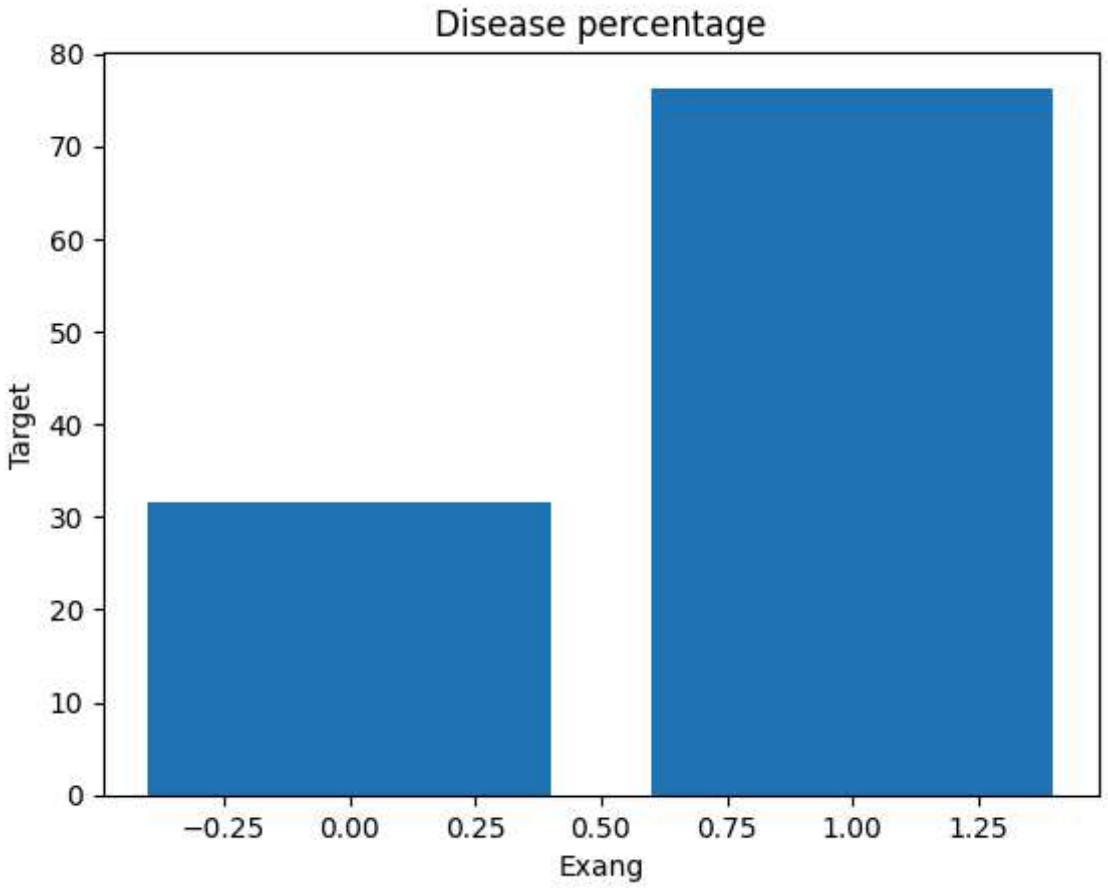
- Calculate disease percentage in:
  - exang = 1
  - exang = 0
- Visualize using bar chart

In [121]:
```
# Calculate disease percentage in:
# exang = 1
disease_percentage = df.groupby('exang')['condition'].mean() * 100

print("Disease percentage by exang:")
```

```
print(disease_percentage)

# exang = 0
```

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

In [187]:
```
plt.bar(disease_percentage.index,disease_percentage.values)
plt.xlabel("Exang")
plt.ylabel("Target")
plt.title("Disease percentage")
plt.show()
```



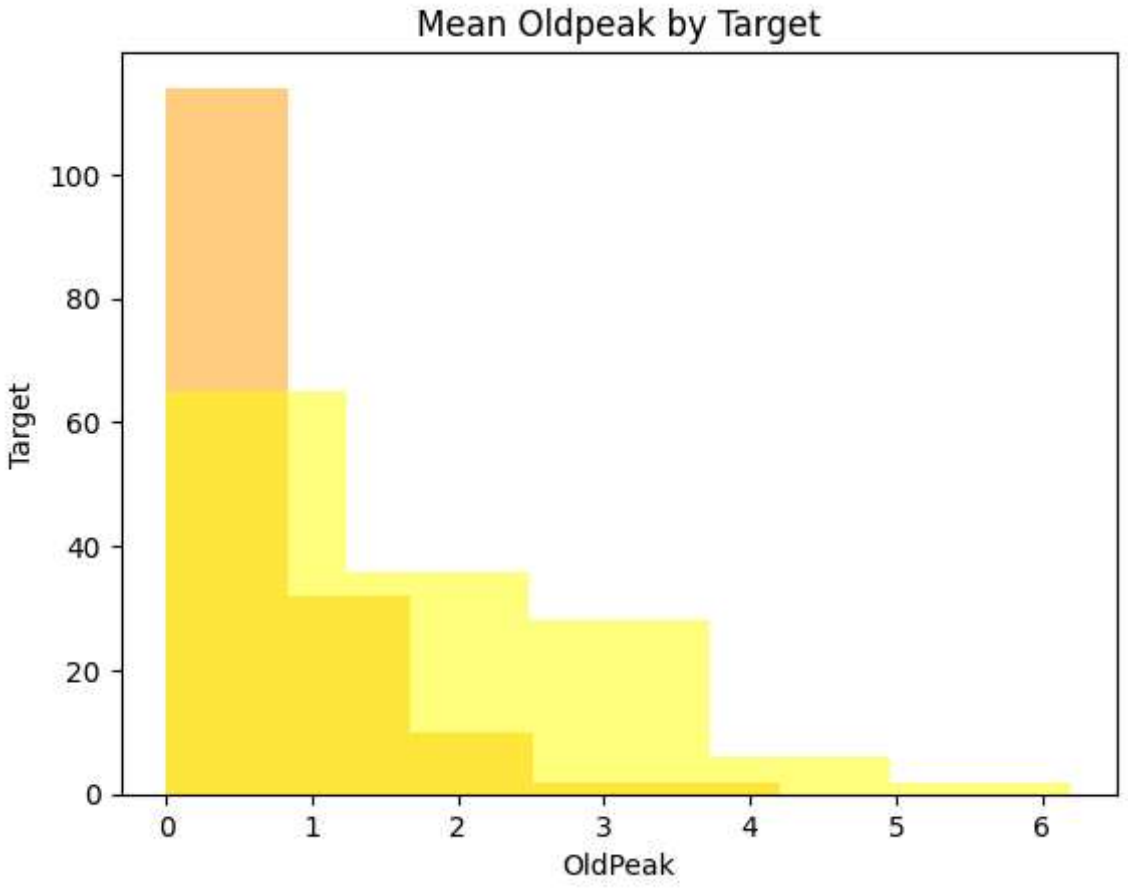
11. ST Depression (oldpeak) Analysis

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

In [34]:
```
# Calculate mean oldpeak by target

oldpeak_mean=df.groupby('condition')['oldpeak'].mean()
oldpeak_mean
```

Out[34]:
```
condition
0    0.598750
1    1.589051
Name: oldpeak, dtype: float64
```

In [64]:
```
# Plot histogram for both classes
plt.hist(df[df['condition'] == 0]['oldpeak'],bins=5,color="orange",alpha=0.5)

plt.hist(df[df['condition'] == 1]['oldpeak'],bins=5,color="yellow",alpha=0.5)
plt.title("Mean Oldpeak by Target")
plt.xlabel("OldPeak")
plt.ylabel("Target")
plt.show()
```



In [35]:
```
if oldpeak_mean[1] > oldpeak_mean[0]:
    print("Trend: Higher ST Depression in patients with Heart Disease.")
elif oldpeak_mean[1] < oldpeak_mean[0]:
    print("Trend: Lower ST Depression in patients with Heart Disease.")
else:
    print("Trend: No significant difference.")
```

```
Trend: Higher ST Depression in patients with Heart Disease.
```

12. Number of Major Vessels (ca) Impact

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

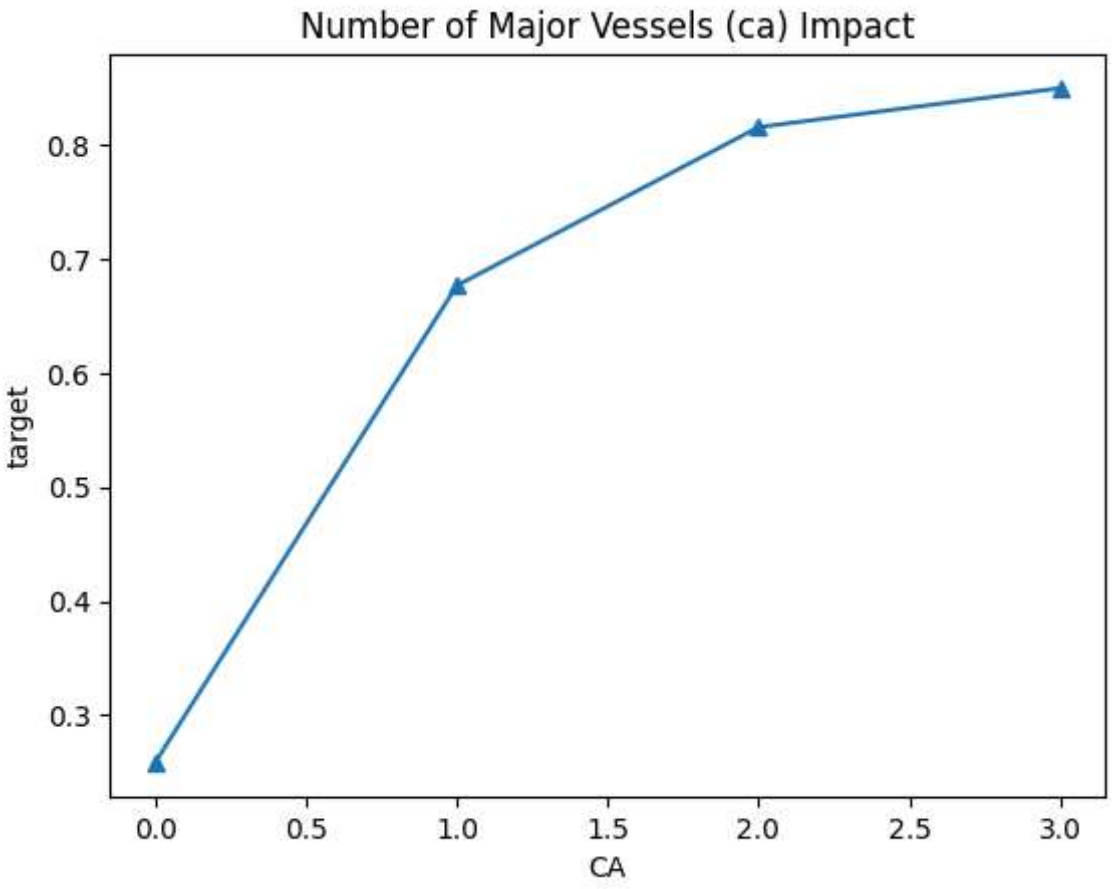
In [36]:
```
# Group by ca
grouped=df.groupby("ca")['condition'].mean()
grouped
```

Out[36]:
```
ca
0    0.258621
1    0.676923
2    0.815789
3    0.850000
Name: condition, dtype: float64
```

In [39]:
```
# Calculate disease probability
print(grouped*100)
```

```
ca
0    25.862069
1    67.692308
2    81.578947
3    85.000000
Name: condition, dtype: float64
```

In [41]:
```
# Plot line chart

plt.plot(grouped.index, grouped.values, marker="^")
plt.xlabel("CA")
plt.ylabel("target")
plt.title("Number of Major Vessels (ca) Impact")
plt.show() #displays the plot
```

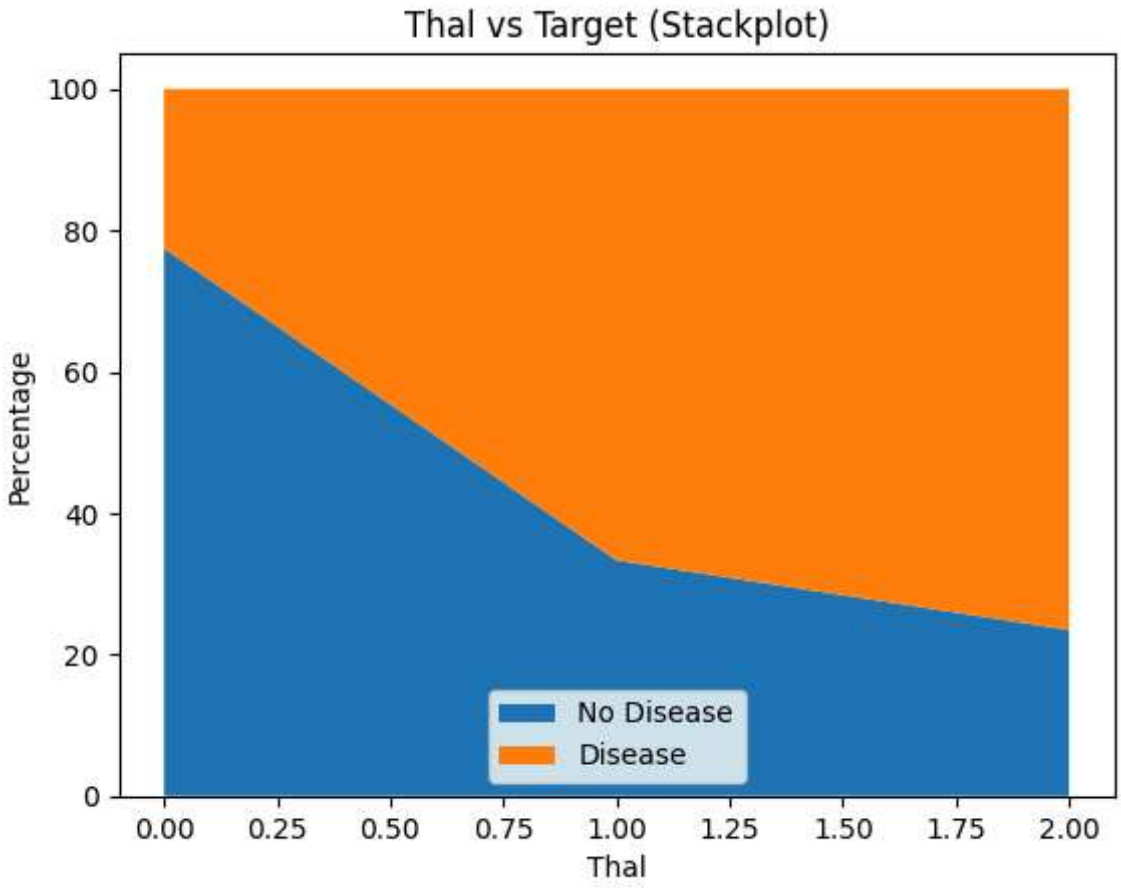


13. Thalassemia vs Disease

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

In [18]:
```
# Cross-tabulate thal and target

ct = pd.crosstab(df['thal'], df['condition'], normalize='index').mul(100)
print(ct)

x = ct.index
y1 = ct[0]
y2 = ct[1]

plt.stackplot(x, y1, y2, labels=['No Disease', 'Disease'])

plt.xlabel('Thal')
plt.ylabel('Percentage')
plt.title('Thal vs Target (Stackplot)')
plt.legend()
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

In [144]:
```
# Find patients with:
# Age > 50
# Cholesterol > 240
# BP > 140


print(df[(df['age']>50) &(df['chol']>240) &(df['trestbps']>140)])
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	\
9	59	1	0	178	270	0	2	145	0	4.2	
10	59	1	0	170	288	0	2	159	0	0.2	
11	59	1	0	160	273	0	2	125	0	0.0	
13	58	0	0	150	283	1	2	162	0	1.0	
16	52	1	0	152	298	1	0	178	0	1.2	
24	71	0	1	160	302	0	0	162	0	0.4	
25	70	1	1	156	245	0	2	143	0	0.0	
26	66	1	1	160	246	0	0	120	1	0.0	
45	54	1	1	192	283	0	2	195	0	0.0	
74	70	1	2	160	269	0	0	112	1	2.9	
76	68	1	2	180	274	1	2	150	1	1.6	
81	67	0	2	152	277	0	0	172	0	0.0	
82	66	0	2	146	278	0	2	152	0	0.0	
84	65	0	2	155	269	0	0	148	0	0.8	
85	65	0	2	160	360	0	2	151	0	0.8	
92	61	1	2	150	243	1	0	137	1	1.0	
160	67	1	3	160	286	0	2	108	1	1.5	
178	64	0	3	180	325	0	0	154	1	0.0	
180	63	0	3	150	407	0	2	154	0	4.0	
190	62	0	3	150	244	0	0	154	1	1.4	
193	61	0	3	145	307	0	2	146	1	1.0	
199	60	1	3	145	282	0	2	142	1	2.8	
201	60	0	3	150	258	0	2	157	0	2.6	
205	60	0	3	158	305	0	2	161	0	0.0	
206	59	1	3	170	326	0	2	140	1	3.4	
209	59	0	3	174	249	0	0	143	1	0.0	
220	58	1	3	150	270	0	2	111	1	0.8	
224	57	1	3	150	276	0	2	112	1	0.6	
225	57	1	3	165	289	1	2	124	0	1.0	
226	57	1	3	152	274	0	0	88	1	1.2	
234	56	0	3	200	288	1	2	133	1	4.0	
241	55	1	3	160	289	0	2	145	1	0.8	
242	55	0	3	180	327	0	1	117	1	3.4	

	slope	ca	thal	condition
9	2	0	2	0
10	1	0	2	1
11	0	0	0	1
13	0	0	0	0
16	1	0	2	0
24	0	2	0	0
25	0	0	0	0
26	1	3	1	1
45	0	1	2	1
74	1	1	2	1
76	1	0	2	1
81	0	1	0	0
82	1	1	0	0
84	0	0	0	0
85	0	0	0	0
92	1	0	0	0
160	1	3	0	1
178	0	0	0	0
180	1	3	2	1
190	1	0	0	1
193	1	0	2	1
199	1	2	2	1
201	1	2	2	1
205	0	0	0	1

```
206    2   0    2        1
209    1   0    0        1
220    0   0    2        1
224    1   1    1        1
225    1   3    2        1
226    1   1    2        1
234    2   2    2        1
241    1   1    2        1
242    1   0    0        1
```

In [52]:
```
# Calculate percentage having disease
data=df[(df['age']>50) & (df['chol']>240) & (df['trestbps']>140)]['condition'].mea
print(round(data,2),"%")
```

66.67 %

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 [44]:
```
df['risk_score']=(df['chol']/200)+(df['trestbps']/120 )+ (df['oldpeak'])
df
```

Out[44]:

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

297 rows × 15 columns

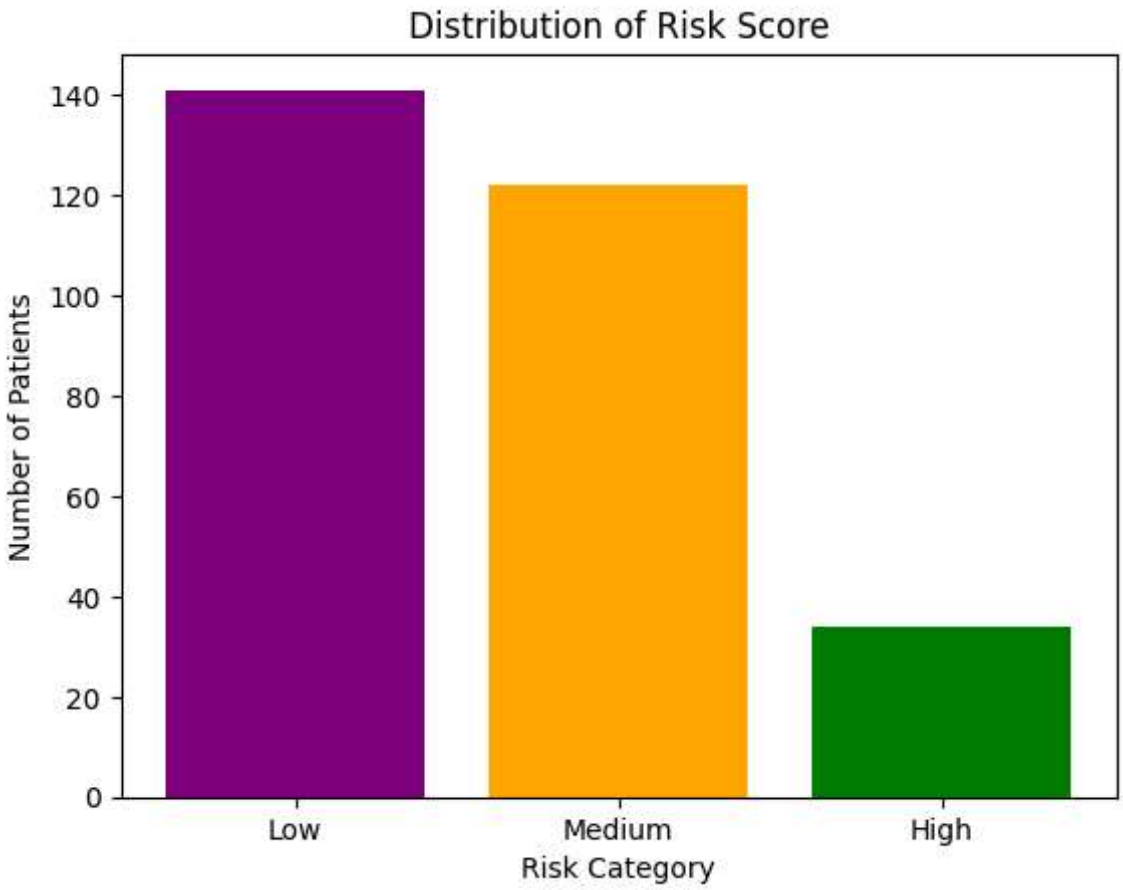
In [57]:
```
def risk_category(score):
    if score < 3:
        return "Low"
    elif score < 5:
        return "Medium"
    else:
        return "High"
df['category'] = df['risk_score'].apply(risk_category)
df
```

Out[57]:

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

297 rows × 17 columns

In [58]:
```
risks=df['category'].value_counts()
plt.bar(risks.index,risks.values, color=['purple','orange','green'])
plt.title("Distribution of Risk Score")
plt.xlabel("Risk Category")
plt.ylabel("Number of Patients")
plt.show()
```



Insights

1. Does cholesterol strongly impact heart disease?

- Correlation between chol and condition is very weak.So cholestrol alone doesn't strongly impact heart disease.

2. Is male population more vulnerable?

- Yes,Disease rate among males is higher.

3. Does exercise-induced angina significantly increase risk?

- Yes,Patients with exang=1 has higher disease percentage.

4. Which feature has strongest correlation with disease?

- No of major vessels and ST depression

In [ ]: