Engineer new features and select relevant features for model training.

- *Generating meaningful features from existing data.*
- *Using techniques like PCA or feature importance to select the most important features.*
- *Optimizing feature sets for improved model performance.*

Suchithra

```
# Importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
# Load the data
data = pd.read_csv("/content/heart.csv")
```

```
# Display the first few rows
data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2

```
# Display the last few rows
data.tail()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tha
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	

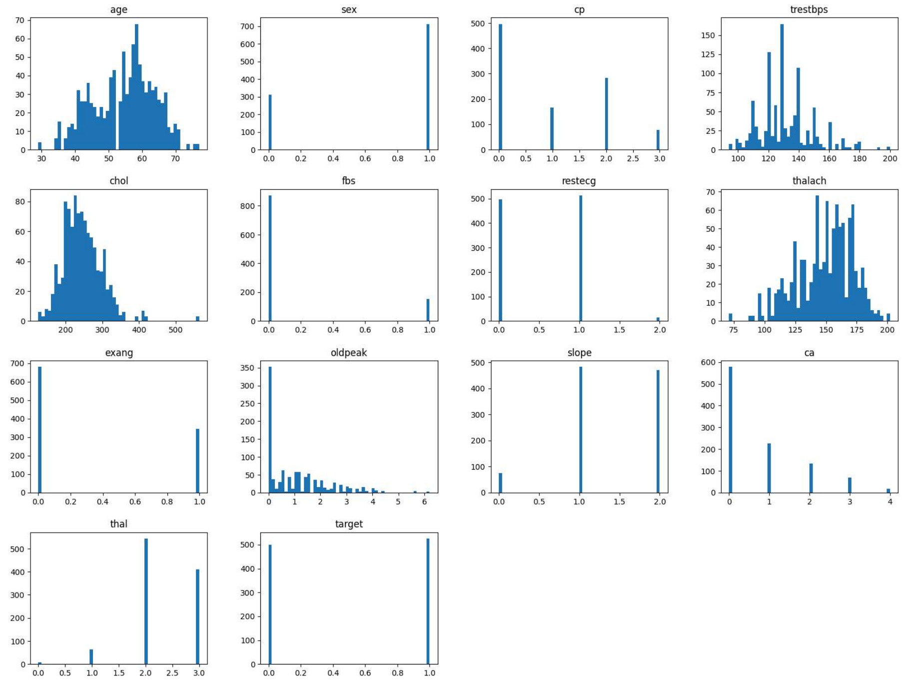
```
# Display column names
data.columns.values
```

```
array(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg',
       'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype=object)
```

```
# Check for missing values
data.isna().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
target      0
dtype: int64
```

```
# Plot histograms for the dataset
data.hist(bins=50, grid=False, figsize=(20,15))
plt.show()
```




```
# Display basic statistics
data.describe()
```

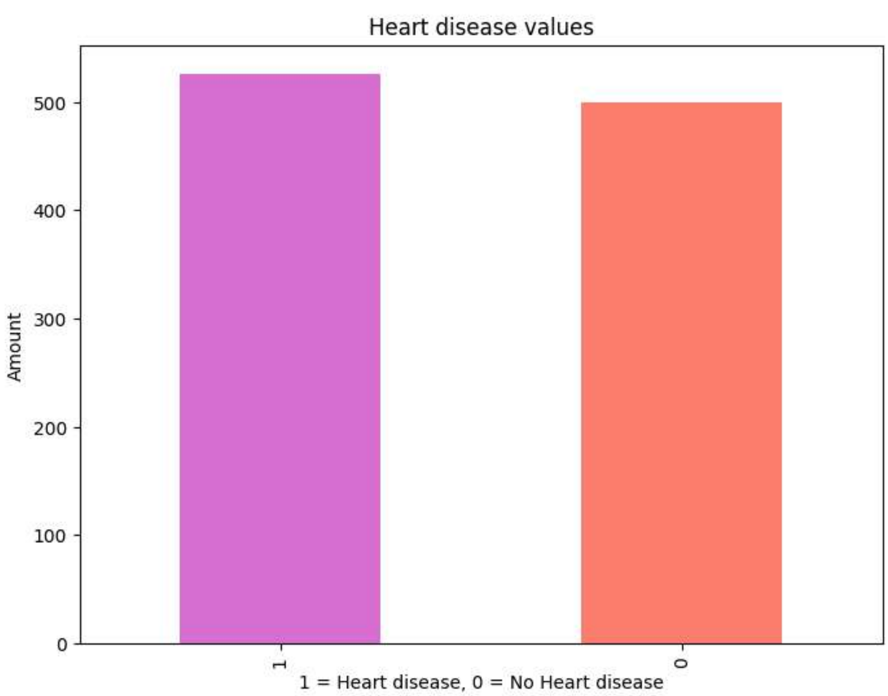
	age	sex	cp	trestbps	chol	fbs	re:
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.0(
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.5:
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.5:
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.0(
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.0(
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.0(
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.0(
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.0(

```
# Existing Questions
explained_questions = [
    "1. How many have heart disease and how many people doesn't have heart disease?",
    "2. People of which sex have the most heart disease?",
    "3. People of which sex have which type of chest pain most?",
    "4. Are people with chest pain more prone to have heart disease?"
]
```

```
# 1. How many have heart disease and how many people don't have heart disease?
data.target.value_counts()
```

```
target
1    526
0    499
Name: count, dtype: int64
```

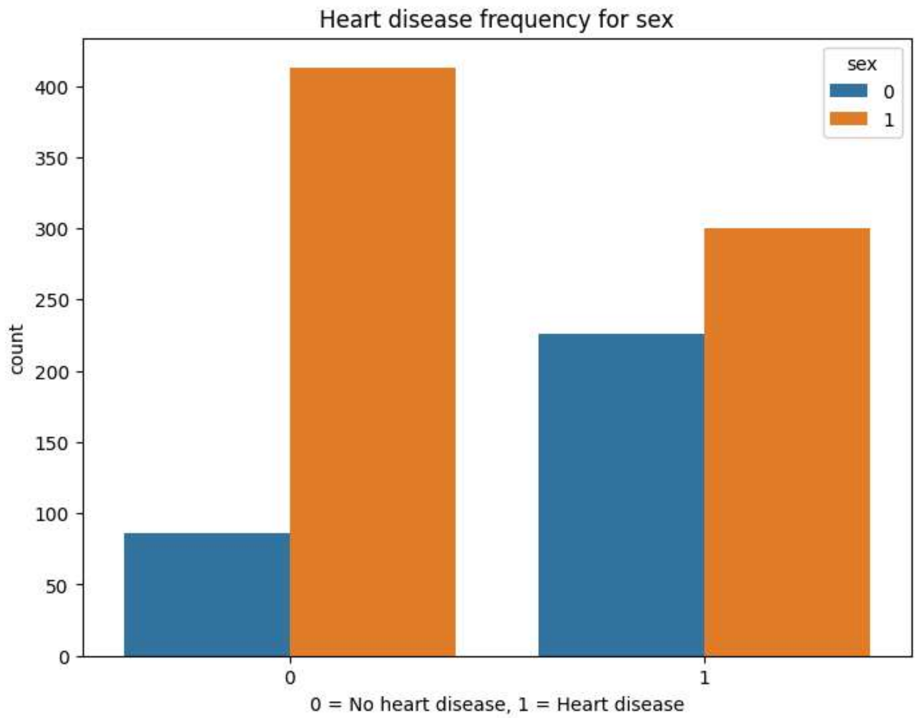
```
# Plotting bar chart
plt.figure(figsize=(8, 6))
data.target.value_counts().plot(kind="bar", color=["orchid", "salmon"])
plt.title("Heart disease values")
plt.xlabel("1 = Heart disease, 0 = No Heart disease")
plt.ylabel("Amount")
plt.show()
```




```
# 2. People of which sex have the most heart disease?
pd.crosstab(data.target, data.sex)
```

sex	0	1
target		
0	86	413
1	226	300

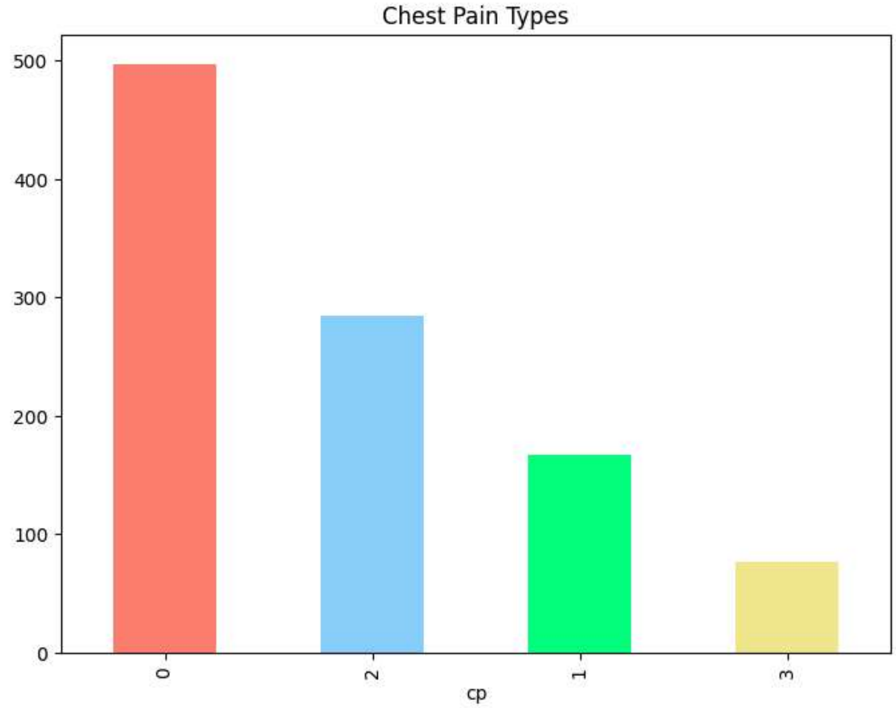
```
plt.figure(figsize=(8, 6))
sns.countplot(x="target", data=data, hue="sex")
plt.title("Heart disease frequency for sex")
plt.xlabel("0 = No heart disease, 1 = Heart disease")
plt.show()
```




```
# 3. People of which sex have which type of chest pain most?
data.cp.value_counts()
```

```
cp
0    497
2    284
1    167
3     77
Name: count, dtype: int64
```

```
plt.figure(figsize=(8, 6))
data.cp.value_counts().plot(kind="bar", color=["salmon", "lightskyblue", "springgreen", "khaki"])
plt.title("Chest Pain Types")
plt.show()
```

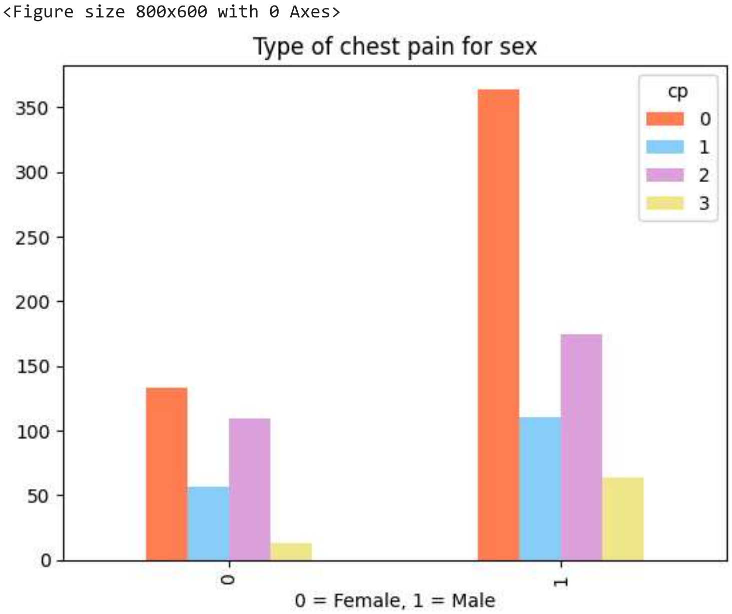



```
pd.crosstab(data.sex, data.cp)
```

cp	0	1	2	3
sex				
0	133	57	109	13
1	364	110	175	64

```
plt.figure(figsize=(8, 6))
pd.crosstab(data.sex, data.cp).plot(kind="bar", color=["coral", "lightskyblue", "plum", "khaki"])
plt.title("Type of chest pain for sex")
plt.xlabel("0 = Female, 1 = Male")
plt.show()
# Most of the male have O-type chest pain and least of them have 3-type chest pain
# In female 0-type and 1-type are almost same
```

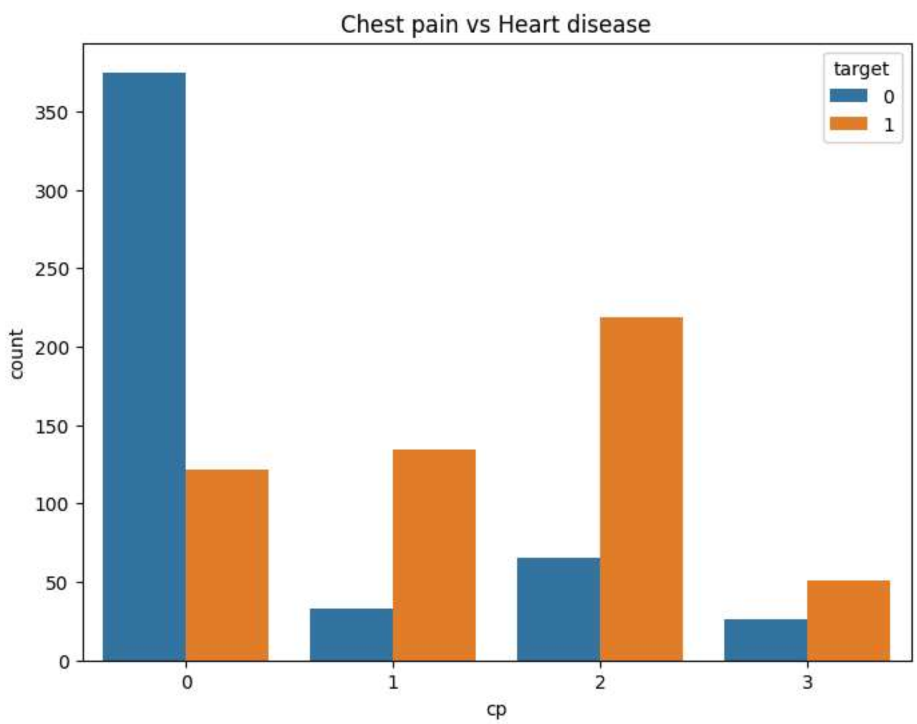
<Figure size 800x600 with 0 Axes>



```
# 4. Are people with chest pain more prone to have heart disease?
pd.crosstab(data.cp, data.target)
```

target	0	1
cp		
0	375	122
1	33	134
2	65	219
3	26	51

```
plt.figure(figsize=(8, 6))
sns.countplot(x="cp", data=data, hue="target")
plt.title("Chest pain vs Heart disease")
plt.show()
# People with chest pain (cp 3) is the most prone have heart disease, followed by (cp 2). The least is (cp 0).
```

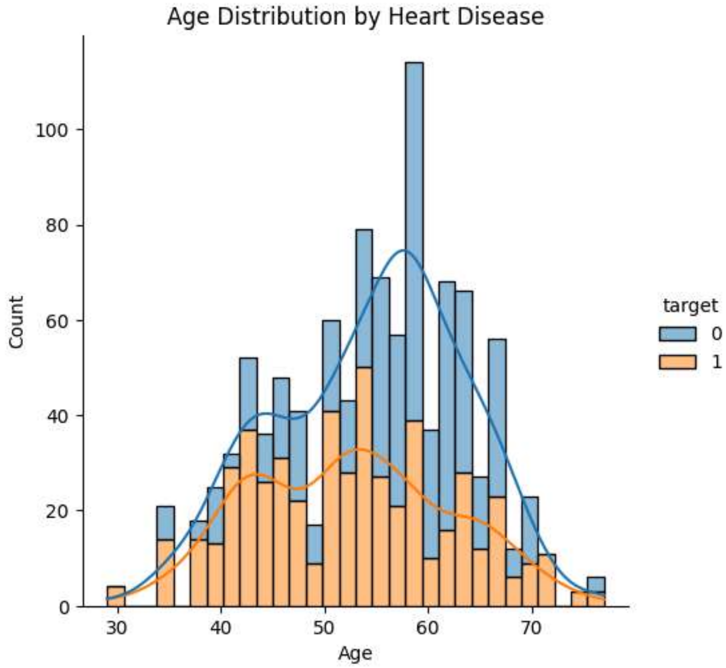



```
data.target.value_counts()
```

```
target
1    526
0    499
Name: count, dtype: int64
```

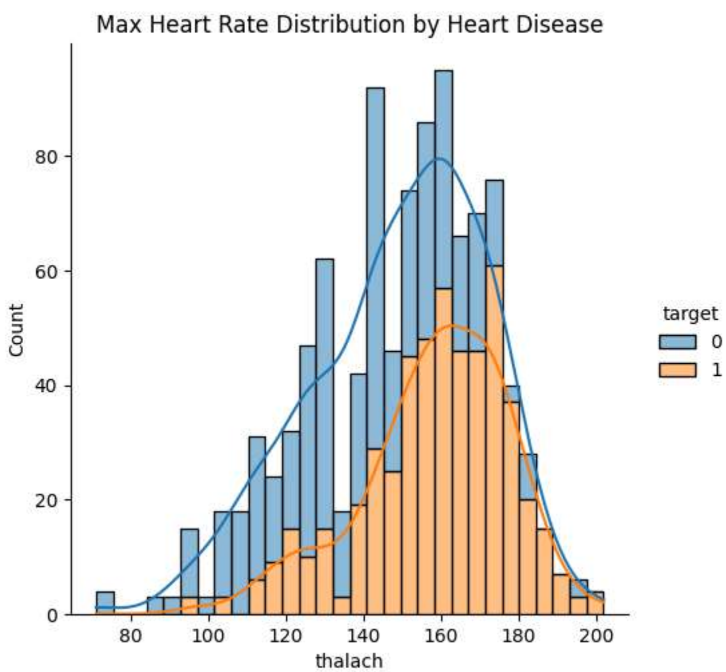
```
# Some of the other questions based on the dataset

# 5. Distribution of age among people with and without heart disease
plt.figure(figsize=(8, 6))
sns.displot(data=data, x="age", hue="target", multiple="stack", bins=30, kde=True)
plt.title("Age Distribution by Heart Disease")
plt.xlabel("Age")
plt.show()
#Higher fasting blood sugar (>120 mg/dl) is slightly more common in people with heart disease.
```

```
<Figure size 800x600 with 0 Axes>
```

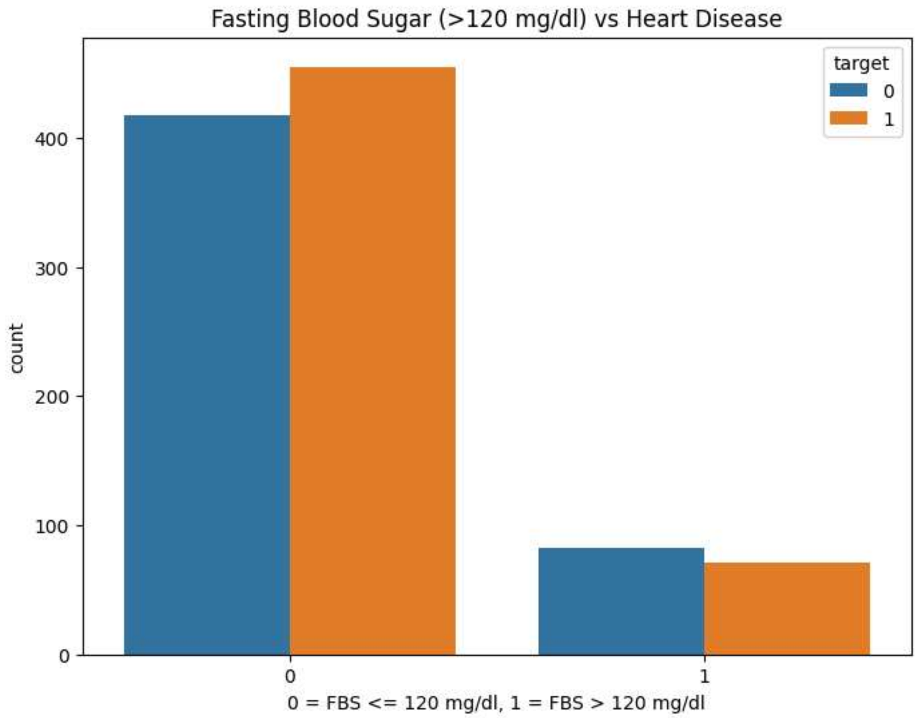



```
# 6. Distribution of maximum heart rate (thalach) among people with and without heart disease
plt.figure(figsize=(8, 6))
sns.displot(data=data, x="thalach", hue="target", multiple="stack", bins=30, kde=True, color="chocolate")
plt.title("Max Heart Rate Distribution by Heart Disease")
plt.show()
#Those without heart disease tend to have higher maximum heart rates.
```

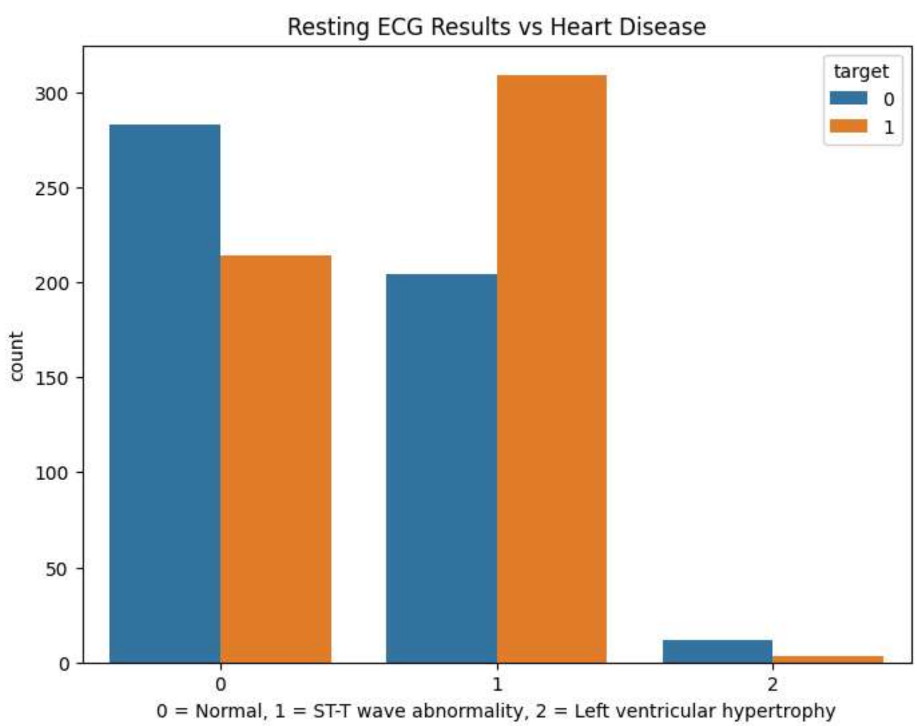
```
<Figure size 800x600 with 0 Axes>
```



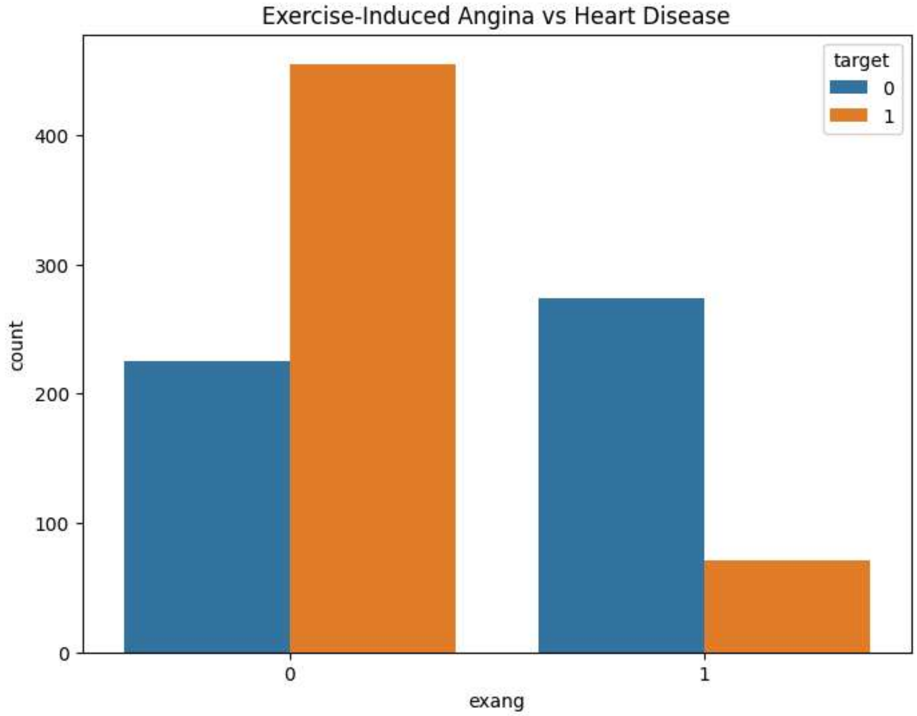

```
# 7. Relationship between fasting blood sugar and heart disease
plt.figure(figsize=(8, 6))
sns.countplot(x="fbs", data=data, hue="target")
plt.title("Fasting Blood Sugar (>120 mg/dl) vs Heart Disease")
plt.xlabel("0 = FBS <= 120 mg/dl, 1 = FBS > 120 mg/dl")
plt.show()
#Higher fasting blood sugar (>120 mg/dl) is slightly more common in people with heart disease.
```




```
# 8. Distribution of electrocardiographic results (restecg) among people with and without heart disease
plt.figure(figsize=(8, 6))
sns.countplot(x="restecg", data=data, hue="target")
plt.title("Resting ECG Results vs Heart Disease")
plt.xlabel("0 = Normal, 1 = ST-T wave abnormality, 2 = Left ventricular hypertrophy")
plt.show()
#ST-T wave abnormalities (restecg 1) are more common in people with heart disease
```

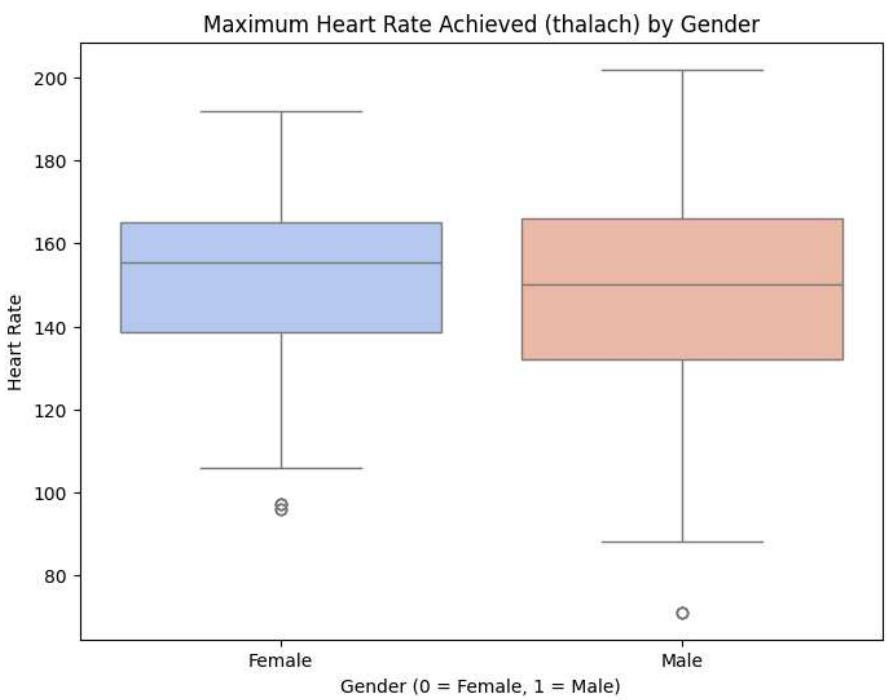



```
# 8. Relationship between exercise-induced angina (exang) and heart disease
plt.figure(figsize=(8, 6))
sns.countplot(x="exang", data=data, hue="target")
plt.title("Exercise-Induced Angina vs Heart Disease")
plt.show()
# Exercise-induced angina (exang 1) is significantly more common in people with heart disease.
```

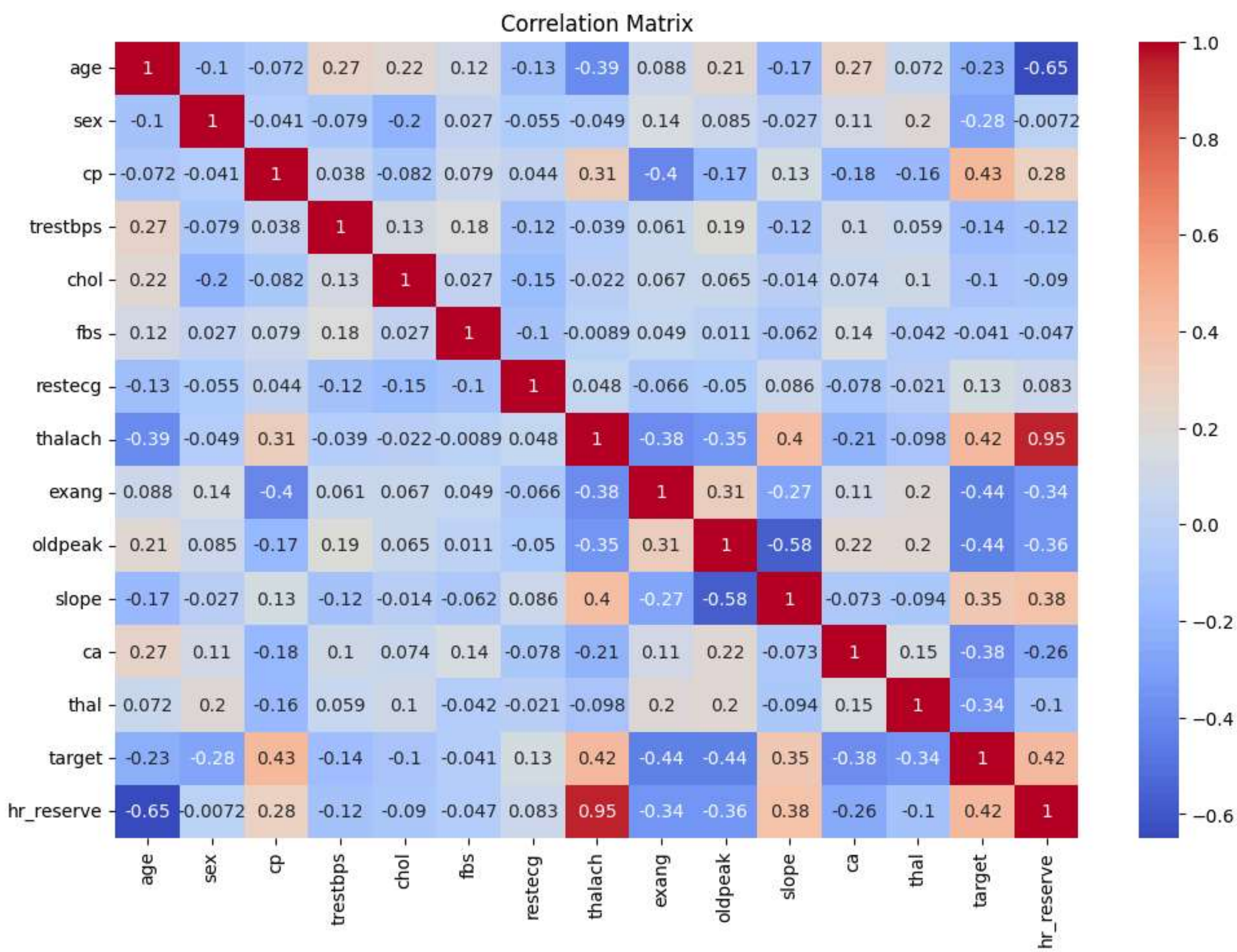



```
# Is there a significant difference in maximum heart rate (thalach) between males and females?
plt.figure(figsize=(8, 6))
sns.boxplot(x="sex", y="thalach", data=data, palette="coolwarm")
plt.title("Maximum Heart Rate Achieved (thalach) by Gender")
plt.xlabel("Gender (0 = Female, 1 = Male)")
plt.ylabel("Heart Rate")
plt.xticks([0, 1], ['Female', 'Male'])
plt.show()
```

```
<ipython-input-26-6e3072f29519>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.boxplot(x="sex", y="thalach", data=data, palette="coolwarm")
```



```
#Feature Engineering and Selection
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split


# Creating new features
data['age_category'] = pd.cut(data['age'], bins=[29, 40, 50, 60, 70, 80], labels=['30-40', '40-50', '50-60', '60-70', '70-80'])
data['cp_sex'] = data['cp'].astype(str) + "_" + data['sex'].astype(str)
data['hr_reserve'] = data['thalach'] - data['age']


# Exclude non-numeric columns
numeric_data = data.select_dtypes(include=[np.number])
corr_matrix = numeric_data.corr()

# Plot the correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Matrix")
plt.show()
```




```
X = data.drop(columns=['target'])
y = data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


pca = PCA(n_components=5)
X_train_pca = pca.fit_transform(X_train.select_dtypes(include=[np.number]))
X_test_pca = pca.transform(X_test.select_dtypes(include=[np.number]))
```

```
# Check the data types of the columns
print(X_train.dtypes)
# Convert categorical columns to numeric using one-hot encoding
X_train_encoded = pd.get_dummies(X_train)
X_test_encoded = pd.get_dummies(X_test)

# Ensure the columns in the training and test sets match
X_train_encoded, X_test_encoded = X_train_encoded.align(X_test_encoded, join='left', axis=1, fill_value=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
age_category   category
cp_sex           object
hr_reserve        int64
dtype: object
```

```
from sklearn.ensemble import RandomForestClassifier

# Initialize and fit the Random Forest model
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train_encoded, y_train)

# Get feature importances
feature_importances = rf.feature_importances_
importance_df = pd.DataFrame({
    'Feature': X_train_encoded.columns,
    'Importance': feature_importances
}).sort_values(by='Importance', ascending=False)

print(importance_df)
```

```
               Feature  Importance
9              oldpeak    0.107396
2                   cp    0.098500
11                  ca    0.096054
13          hr_reserve    0.090098
12                thal    0.089724
7              thalach    0.080136
20           cp_sex_0_1    0.067350
4                 chol    0.059853
0                  age    0.059159
3             trestbps    0.056645
8                exang    0.039667
10               slope    0.037078
1                  sex    0.016322
6              restecg    0.015281
24           cp_sex_2_1    0.013111
17  age_category_60-70    0.011003
16  age_category_50-60    0.010951
23           cp_sex_2_0    0.009668
26           cp_sex_3_1    0.007905
15  age_category_40-50    0.007813
5                  fbs    0.007304
19           cp_sex_0_0    0.005139
22           cp_sex_1_1    0.004429
14  age_category_30-40    0.003971
21           cp_sex_1_0    0.002721
25           cp_sex_3_0    0.001546
18  age_category_70-80    0.001177
```

```
# Print the feature ranking
importances = rf.feature_importances_
indices = np.argsort(importances)[::-1]  # Sort the feature importances in descending order

print("Feature ranking:")
for f in range(X_train_encoded.shape[1]):
    print(f"{f + 1}. feature {X_train_encoded.columns[indices[f]]} ({importances[indices[f]]})")
```

```
Feature ranking:
1. feature oldpeak (0.10739582344765432)
2. feature cp (0.09849986710261464)
3. feature ca (0.09605366079676292)
4. feature hr_reserve (0.09009771136146892)
5. feature thal (0.08972390856045026)
6. feature thalach (0.08013622985389504)
7. feature cp_sex_0_1 (0.06735046452032004)
8. feature chol (0.059852564912944564)
9. feature age (0.0591585037725649)
10. feature trestbps (0.05664469300517228)
11. feature exang (0.03966699660685052)
12. feature slope (0.037078155184638605)
13. feature sex (0.016321883230633782)
14. feature restecg (0.01528119147090576)
15. feature cp_sex_2_1 (0.013111225825423761)
16. feature age_category_60-70 (0.011002613197031191)
17. feature age_category_50-60 (0.010951473072433073)
18. feature cp_sex_2_0 (0.009668036258707716)
19. feature cp_sex_3_1 (0.007904614401300664)
```

```
# Plot the feature importances
plt.figure(figsize=(12, 8))
plt.title("Feature Importances")
plt.bar(range(X_train_encoded.shape[1]), importances[indices], align="center")
plt.xticks(range(X_train_encoded.shape[1]), X_train_encoded.columns[indices], rotation=90)
plt.show()
```