In [1]:
```
## Heart Attck Prediction ##
```

In [3]:
```
# 1. Import Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

In [4]:
```
# 2. Load and Clean Data
df = pd.read_csv("heart.csv")
df.columns = df.columns.str.strip()
df.replace('?', np.nan, inplace=True)
df
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
0	28	1	2	130	132	0	2	185	0	0.0	NaN	NaN	NaN	0
1	29	1	2	120	243	0	0	160	0	0.0	NaN	NaN	NaN	0
2	29	1	2	140	NaN	0	0	170	0	0.0	NaN	NaN	NaN	0
3	30	0	1	170	237	0	1	170	0	0.0	NaN	NaN	6	0
4	31	0	2	100	219	0	1	150	0	0.0	NaN	NaN	NaN	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
289	52	1	4	160	331	0	0	94	1	2.5	NaN	NaN	NaN	1
290	54	0	3	130	294	0	1	100	1	0.0	2	NaN	NaN	1
291	56	1	4	155	342	1	0	150	1	3.0	2	NaN	NaN	1
292	58	0	2	180	393	0	0	110	1	1.0	2	NaN	7	1
293	65	1	4	130	275	0	1	115	1	1.0	2	NaN	NaN	1

294 rows × 14 columns

In [5]:
```
# Convert columns to numeric
for col in ['chol', 'slope', 'ca', 'thal']:
    df[col] = pd.to_numeric(df[col], errors='coerce')
df[col]
```

Out[5]:
```
0      NaN
1      NaN
2      NaN
3      6.0
4      NaN
      ...
289    NaN
290    NaN
291    NaN
292    7.0
293    NaN
Name: thal, Length: 294, dtype: float64
```

In [6]:
```
# Impute missing values with most frequent
imputer = SimpleImputer(strategy='most_frequent')
df_imputed = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)
df_imputed
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
0	28	1	2	130	132.0	0	2	185	0	0.0	2.0	0.0	7.0	0
1	29	1	2	120	243.0	0	0	160	0	0.0	2.0	0.0	7.0	0
2	29	1	2	140	230.0	0	0	170	0	0.0	2.0	0.0	7.0	0
3	30	0	1	170	237.0	0	1	170	0	0.0	2.0	0.0	6.0	0
4	31	0	2	100	219.0	0	1	150	0	0.0	2.0	0.0	7.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
289	52	1	4	160	331.0	0	0	94	1	2.5	2.0	0.0	7.0	1
290	54	0	3	130	294.0	0	1	100	1	0.0	2.0	0.0	7.0	1
291	56	1	4	155	342.0	1	0	150	1	3.0	2.0	0.0	7.0	1
292	58	0	2	180	393.0	0	0	110	1	1.0	2.0	0.0	7.0	1
293	65	1	4	130	275.0	0	1	115	1	1.0	2.0	0.0	7.0	1

294 rows × 14 columns

In [7]:
```
# Create binary target
df_imputed['target'] = df_imputed['num'].astype(int).apply(lambda x: 1 if x > 0 else 0)
df_imputed.drop(columns='num', inplace=True)
```

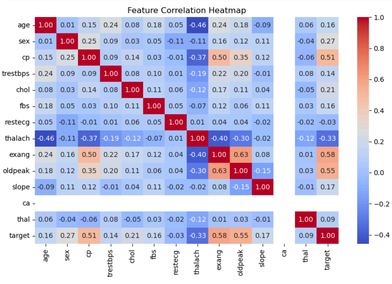
In [8]:
```
# Ensure all columns are numeric
df_imputed = df_imputed.apply(pd.to_numeric)
df_imputed
```

Out[8]:

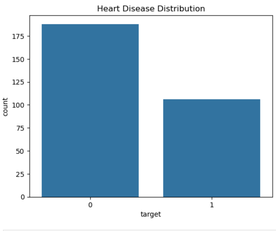
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	28	1	2	130	132.0	0	2	185	0	0.0	2.0	0.0	7.0	0
1	29	1	2	120	243.0	0	0	160	0	0.0	2.0	0.0	7.0	0
2	29	1	2	140	230.0	0	0	170	0	0.0	2.0	0.0	7.0	0
3	30	0	1	170	237.0	0	1	170	0	0.0	2.0	0.0	6.0	0
4	31	0	2	100	219.0	0	1	150	0	0.0	2.0	0.0	7.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
289	52	1	4	160	331.0	0	0	94	1	2.5	2.0	0.0	7.0	1
290	54	0	3	130	294.0	0	1	100	1	0.0	2.0	0.0	7.0	1
291	56	1	4	155	342.0	1	0	150	1	3.0	2.0	0.0	7.0	1
292	58	0	2	180	393.0	0	0	110	1	1.0	2.0	0.0	7.0	1
293	65	1	4	130	275.0	0	1	115	1	1.0	2.0	0.0	7.0	1

294 rows × 14 columns

In [9]:
```
# 3. Visualize Data
plt.figure(figsize=(10, 6))
sns.heatmap(df_imputed.corr(), annot=True, fmt='.2f', cmap='coolwarm')
plt.title("Feature Correlation Heatmap")
plt.show()
```



In [10]:
```
sns.countplot(x='target', data=df_imputed)
plt.title("Heart Disease Distribution")
plt.show()
```

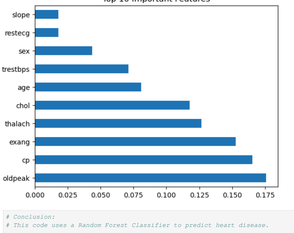


In [11]:
```
# 4. Model Building
X = df_imputed.drop(columns='target')
y = df_imputed['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

In [12]:
```
# 5. Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.8135593220338984

Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.87      0.86        38
           1       0.75      0.71      0.73        21

    accuracy                           0.81        59
   macro avg       0.80      0.79      0.79        59
weighted avg       0.81      0.81      0.81        59


Confusion Matrix:
 [[33  5]
 [ 6 15]]
```

In [26]:
```
# 6. Feature Importance Plot
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.title("Top 10 Important Features")
plt.show()
```



In [28]:
```
# Conclusion:
# This code uses a Random Forest Classifier to predict heart disease.

# It handles missing values using the most frequent value imputation.

# The model's accuracy and feature importance help assess performance.

# Visualizations like heatmaps and countplots give insights into data relationships and class balance.
```

In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:

In [ ]:

In [ ]: