ST PETER'S HOSPITAL PROJECT FOR PREDICTING HEART DISEASE IN PATIENT USING MACHINE LEARNING ALGORITHMS

In [1]:

```
# Import necessary libraries

# For Data Analysis
import pandas as pd
import numpy as np

# For Data Visualisation
import matplotlib.pyplot as plt
import seaborn as sns

# Data Preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

# Classifiers Libraries
from sklearn.linear_model import SGDClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

#!pip install xgboost
from xgboost import XGBClassifier
from sklearn.svm import LinearSVC, SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

# Evaluation Metrics
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
from sklearn.metrics import confusion_matrix
sns.set()
import warnings
warnings.filterwarnings("ignore")
from scipy.stats import skew, kurtosis
```

In [101]:

```
df = pd.read_csv(r"C:\Users\HENRY OKEOMA\Downloads\heart.csv")
```

In [102]:

```
df.head()
```

Out[102]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Features in the dataset and meaning

- age = age in years
- sex = (1= male; 0 = female)
- chest pain type (cp) (1:typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic)
- resting blood pressure(trestbps) (in mm hg on admission to the hospital),
- serum cholesterol (chol), (in mg/dl)
- fasting blood sugar > 120 mg/dl (fbs), (1 = True, 0 = False)
- resting electrocardiographic results (restecg),
- maximum heart rate achieved (thalach),
- exercise-inducedangina (exang), (1=yes, 0= No)
- ST depression induced by exercise relativeto rest (oldpeak),
- The slope of the peak exercise ST segment(slope),

- number of major vessels colored by flourosopy(ca),(0-3)
- thalassemia (thal)(thal-3 = normal, 6 = fixed defect, 7 = reversable defect)
- Target - Heart disease or not (1=yes, 0= no)

In [103]:

```
df.columns
```

Out[103]:

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

In [104]:

```
etter understanding and flow of Analysis, we will rename some of the columns
mns = ['age', 'sex', 'chest_pain_type', 'rest_blood_pressure', 'cholesterol', 'fasting_blood_sugar', 'rest_ecg', 'max_heart_
'excercise_induced_angina', 'st_depression', 'st_slope', 'num_major_bloodvessels', 'thalassemia', 'target']
(2)
```

Out[104]:

	age	sex	chest_pain_type	rest_blood_pressure	cholesterol	fasting_blood_sugar	rest_ecg	max_heart_rate	excercise_induced_angina
0	63	1	3	145	233	1	0	150	0
1	37	1	2	130	250	0	1	187	0

In [105]:

```
# Data verification of data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   age                       303 non-null    int64  
 1   sex                       303 non-null    int64  
 2   chest_pain_type           303 non-null    int64  
 3   rest_blood_pressure       303 non-null    int64  
 4   cholesterol               303 non-null    int64  
 5   fasting_blood_sugar       303 non-null    int64  
 6   rest_ecg                  303 non-null    int64  
 7   max_heart_rate            303 non-null    int64  
 8   excercise_induced_angina  303 non-null    int64  
 9   st_depression             303 non-null    float64
 10  st_slope                  303 non-null    int64  
 11  num_major_bloodvessels    303 non-null    int64  
 12  thalassemia               303 non-null    int64  
 13  target                    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

No Missing data in our data set

In [107]:

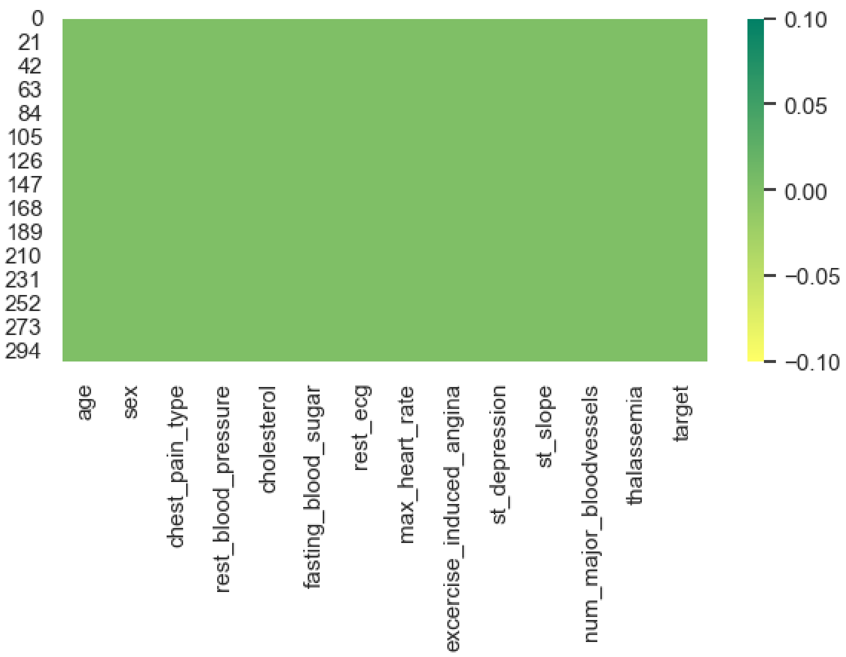
```
# Sttistical Analysis
df.describe().astype(int)
```

Out[107]:

	age	sex	chest_pain_type	rest_blood_pressure	cholesterol	fasting_blood_sugar	rest_ecg	max_heart_rate	excercise_induced_an
count	303	303	303	303	303	303	303	303	
mean	54	0	0	131	246	0	0	149	
std	9	0	1	17	51	0	0	22	
min	29	0	0	94	126	0	0	71	
25%	47	0	0	120	211	0	0	133	
50%	55	1	1	130	240	0	1	153	
75%	61	1	2	140	274	0	1	166	
max	77	1	3	200	564	1	2	202	

In [108]:

```
# Check for Missing values and Visualise
df.isna().sum()
plt.figure(figsize=(7,3))
sns.heatmap(df.isna(), cbar=True, cmap='summer_r');
```

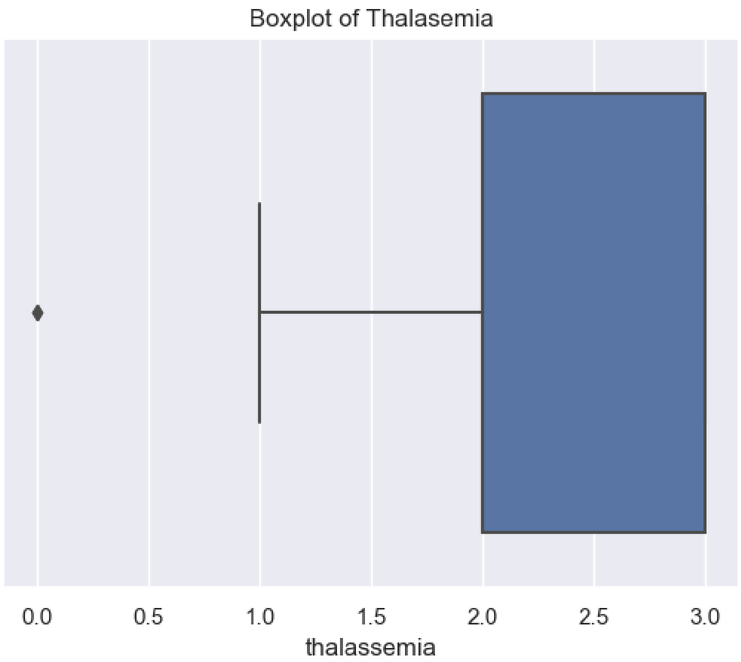


Exploratory Data Analysis

- Univariate Analysis

In [109]:

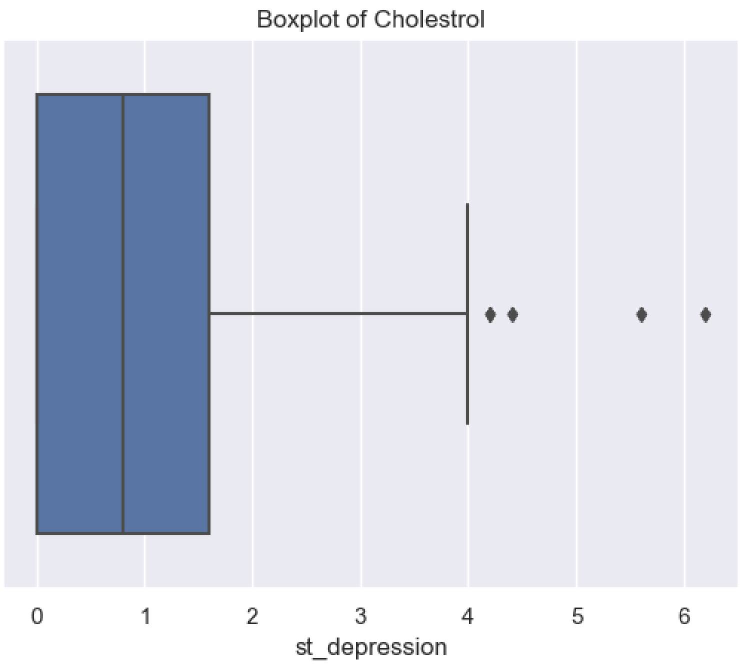
```
# We shall perform Few Univariate analysis on Thalasemia
sns.boxplot(x='thalassemia', data=df)
plt.title('Boxplot of Thalasemia');
```



Insight: We have one outlier in our dataset on thalasemia

In [110]:

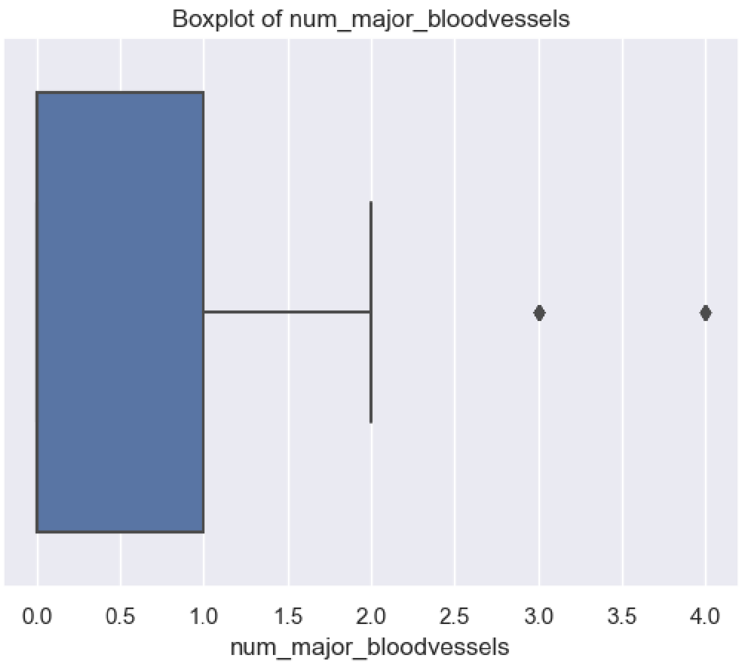
```
# We shall perform Few Univariate analysis on st_depression
sns.boxplot(x='st_depression', data=df)
plt.title('Boxplot of Cholestrol');
```



Insight: We also have few outliers in our dataset on st_depression

In [111]:

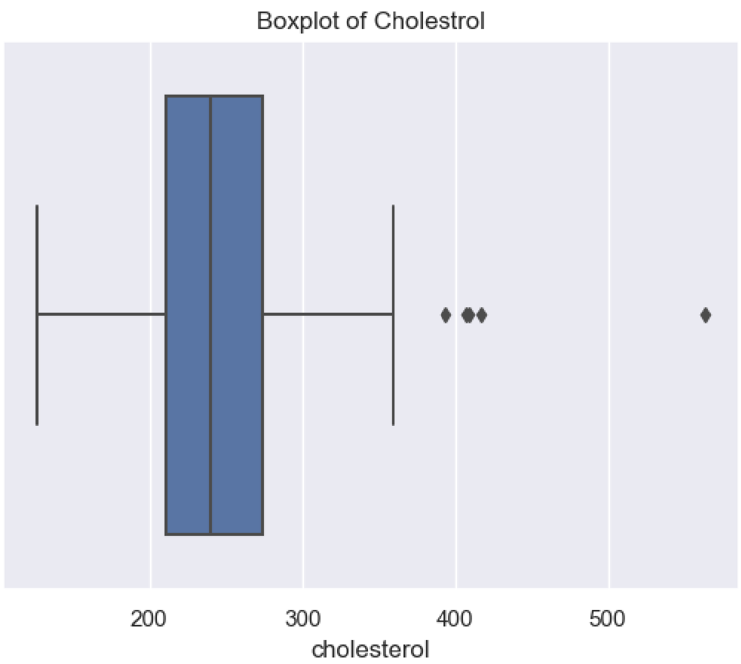
```
# We shall perform Few Univariate analysis on num_major_bloodvessels
sns.boxplot(x='num_major_bloodvessels', data=df)
plt.title('Boxplot of num_major_bloodvessels');
```



Insight: We also have few outliers in our dataset on num of major blood vessels

In [113]:

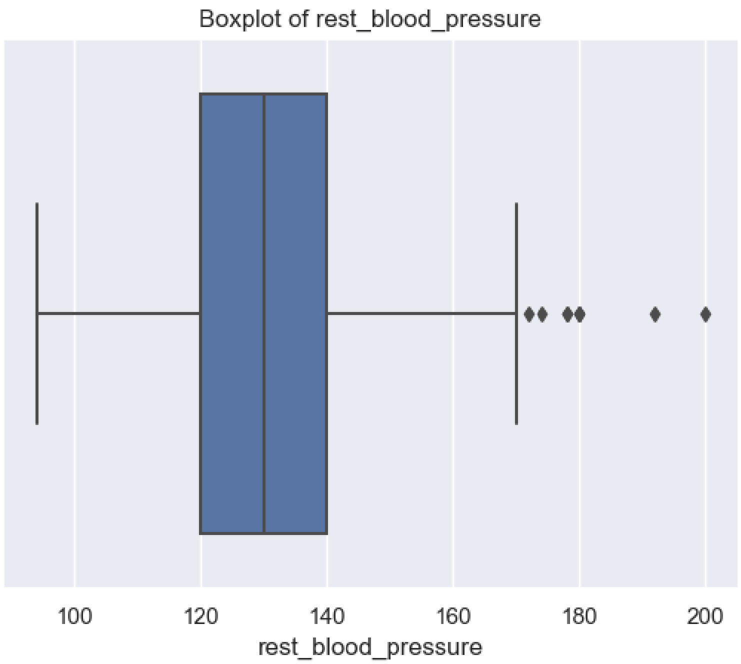
```
# We shall perform Few Univariate analysis on Cholestrol
sns.boxplot(x='cholesterol', data=df)
plt.title('Boxplot of Cholestrol');
```



Insight: Just few outliers on the Cholesterol

In [115]:

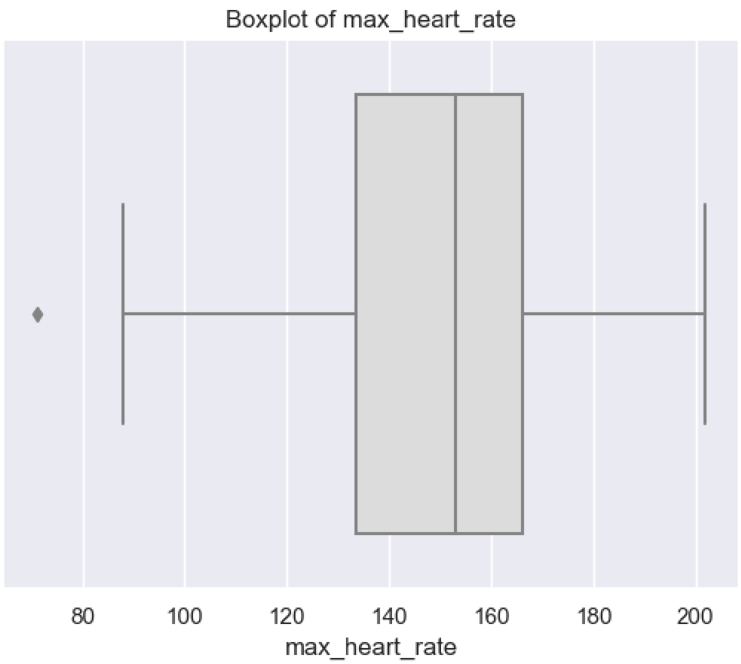
```
# We shall perform Few Univariate analysis on
sns.boxplot(x='rest_blood_pressure', data=df)
plt.title('Boxplot of rest_blood_pressure');
```



Insight: Just few outliers also on the rest blood sugar

In [116]:

```
# We shall perform Few Univariate analysis on
sns.boxplot(x='max_heart_rate', data=df, palette='coolwarm')
plt.title('Boxplot of max_heart_rate');
```



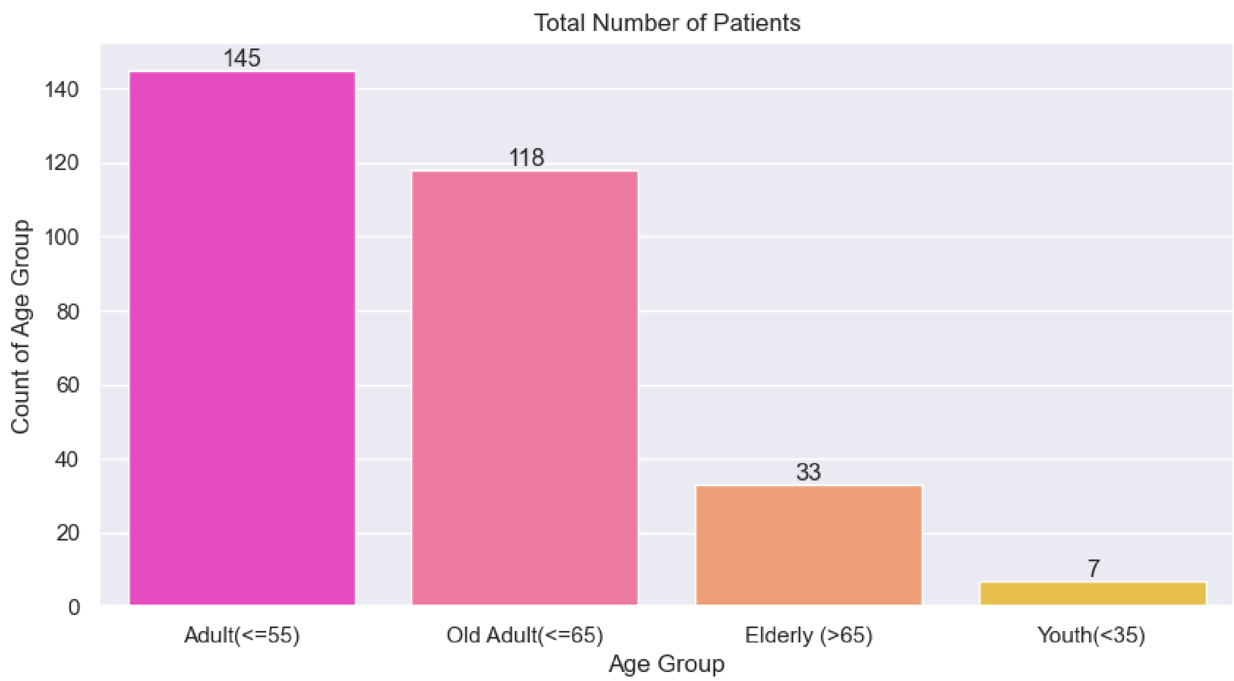
Insight: Just one outlier on the max heart rate

In [117]:

```
# Creating Age Bracket for Age of the Patients
# Age Brackets

def age_bracket(age):
    if age <= 35:
        return 'Youth(<35)'
    elif age <= 55:
        return 'Adult(<=55)'
    elif age <= 65:
        return 'Old Adult(<=65)'
    else:
        return 'Elderly (>65)'

df['age_bracket'] = df['age'].apply(age_bracket)
```

In [118]:

```
# Investigating the age group of patients
plt.figure(figsize=(10,5))
ax = sns.countplot(x='age_bracket', data=df, order=df['age_bracket'].value_counts().index, palette='spring')
values = df['age_bracket'].value_counts().values
ax.bar_label(container = ax.containers[0], labels=values)
plt.title('Total Number of Patients')
plt.xlabel('Age Group')
plt.ylabel('Count of Age Group');
```



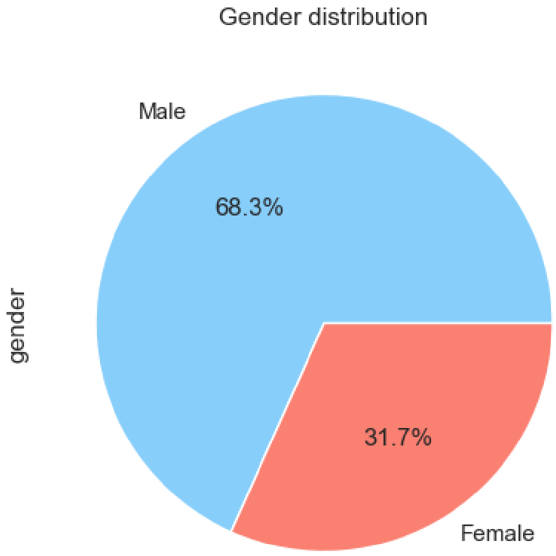
Insight: We have More people between 35 - 55 age brackets, followed by people between 55 - 65 age brackets. The elderly are minimal in our data, while youth od less than 35years are just 7.

In [119]:

```
# Creating a Gender column for Sex of the Patients
# Age Brackets

def gender(sex):
    if sex == 1:
        return 'Male'
    else:
        return 'Female'

df['gender'] = df['sex'].apply(gender)
```

In [120]:

```
# Visualising the Gender as below using a Pie Chart

plt.figure(figsize =(5,5))
df['gender'].value_counts(normalize=False).plot.pie(autopct='%1.1f%%', colors = ['lightskyblue', 'salmon'])
plt.title("Gender distribution");
```



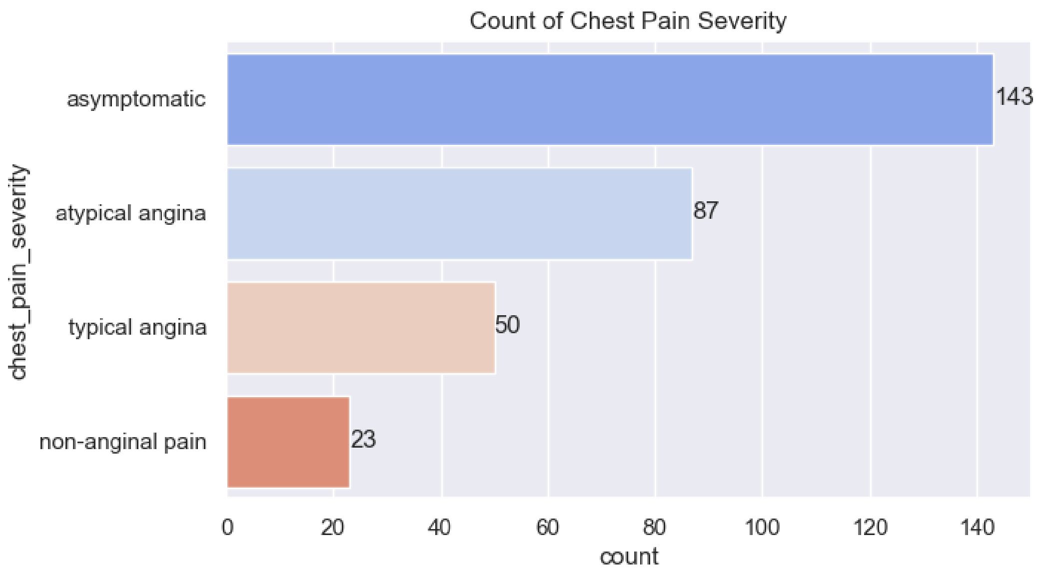
Insight: We have more male than female in our dataset

In [121]:

```
# Creating a Chest pain Category - we shall see how many peoplewith chest pain severity
# Chest pain severity

def chest_pain_severity(cp):
    if cp == 1:
        return 'typical angina'
    elif cp == 2:
        return 'atypical angina'
    elif cp == 3:
        return 'non-anginal pain'
    else:
        return 'asymptomatic'

df['chest_pain_severity'] = df['chest_pain_type'].apply(chest_pain_severity)
```

In [122]:

```
Visualise the  Chest pain severity data column as newly created

lt.figure(figsize=(7,4))
x = sns.countplot(y='chest_pain_severity', data=df, order=df['chest_pain_severity'].value_counts().index, palette='coolwarm'
alues = df['chest_pain_severity'].value_counts().values
x.bar_label(container = ax.containers[0], labels=values)
lt.title('Count of Chest Pain Severity');
```



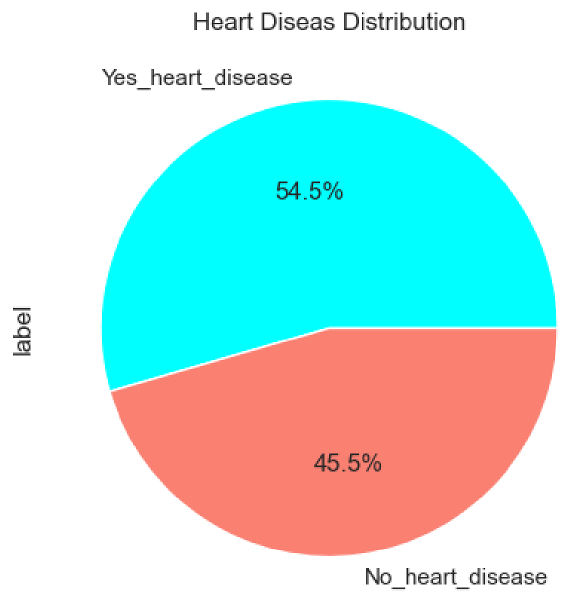
Insight: We have most of the patients asymtomatic and atypical angina, non anginal pain is the least

In [123]:

```
# Creating a Function for the target Heart Disease in patients to see how many actually have the heart disease
# Target - have heart disease (0 = No, 1 = Yes)

def label(tg):
    if tg == 1:
        return 'Yes_heart_disease'
    else:
        return 'No_heart_disease'

df['label'] = df['target'].apply(label)
```

In [124]:

```
# Visualising the Target as below using a Pie Chart

plt.figure(figsize =(5,5))
df['label'].value_counts(normalize=False).plot.pie(autopct='%1.1f%%', colors = ['cyan', 'salmon'])
plt.title("Heart Diseas Distribution");
```

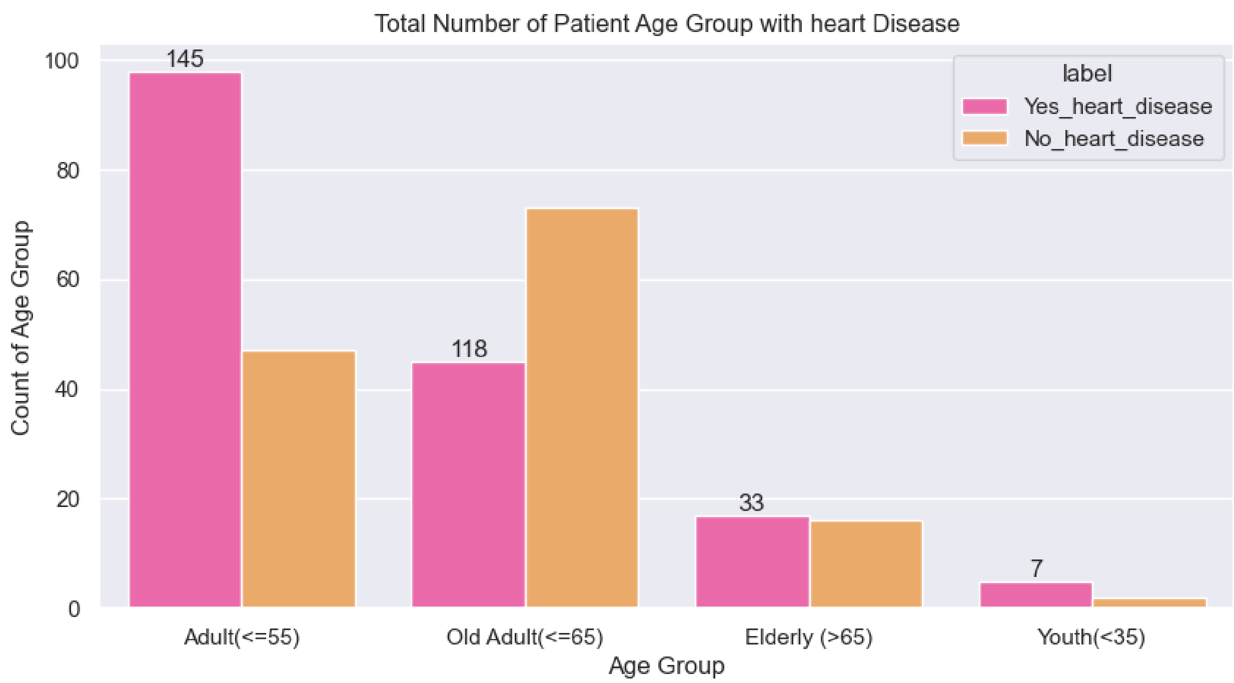


Insight: We have most of the patients with heart disease about 54.5% in our dataset.

BIVARIATE ANALYSIS here we shall target the heart disease

In [125]:

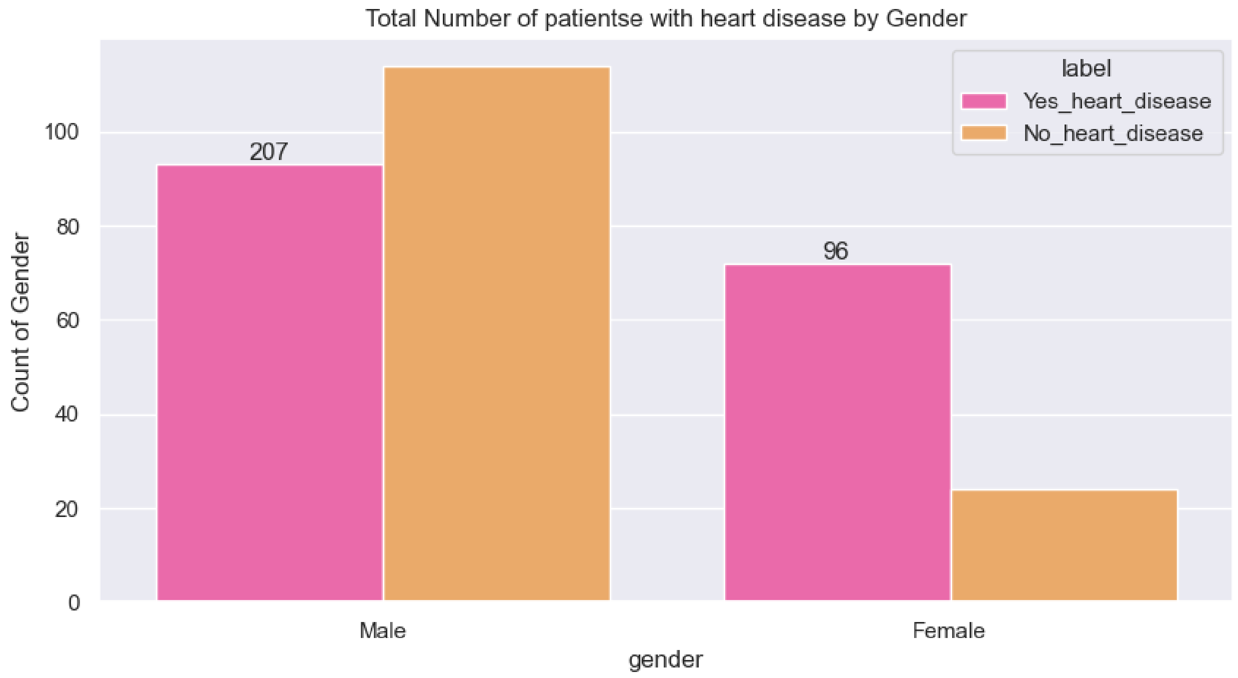
```
# Investigating the age group of patients with heart disease
plt.figure(figsize=(10,5))
ax = sns.countplot(x='age_bracket', data=df, order=df['age_bracket'].value_counts().index, hue='label', palette='spring')
values = df['age_bracket'].value_counts().values
ax.bar_label(container = ax.containers[0], labels=values)
plt.title('Total Number of Patient Age Group with heart Disease')
plt.xlabel('Age Group')
plt.ylabel('Count of Age Group');
```



Insight: Most people between 35-65years have heart disease than the rest of the age group

In [126]:

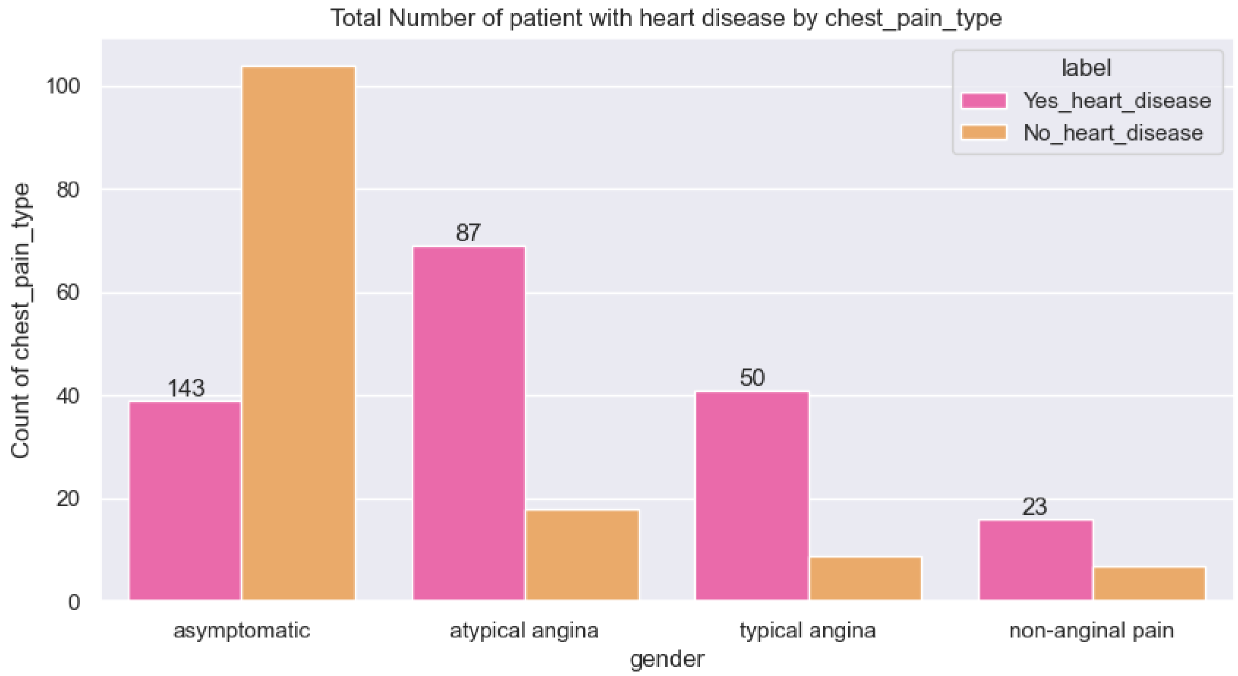
```
# Investigating the Gender of patients with heart disease

plt.figure(figsize=(10,5))
ax = sns.countplot(x='gender', data=df, hue='label', order=df['gender'].value_counts().index, palette='spring')
values = df['gender'].value_counts().values
ax.bar_label(container = ax.containers[0], labels=values)
plt.title('Total Number of patientse with heart disease by Gender')
plt.xlabel('gender')
plt.ylabel('Count of Gender');
```



Insight: In male patients, we have more people with no heart disease. But in the femal gender, we have more patients with heart disease. Female are more proone to heart disease than males in the data set.

In [127]:

```
# Investigating the Chest pain of patients with heart disease

plt.figure(figsize=(10,5))
ax = sns.countplot(x='chest_pain_severity', data=df, hue='label', order=df['chest_pain_severity'].value_counts().index, pal
values = df['chest_pain_severity'].value_counts().values
ax.bar_label(container = ax.containers[0], labels=values)
plt.title('Total Number of patient with heart disease by chest_pain_type')
plt.xlabel('gender')
plt.ylabel('Count of chest_pain_type');
```

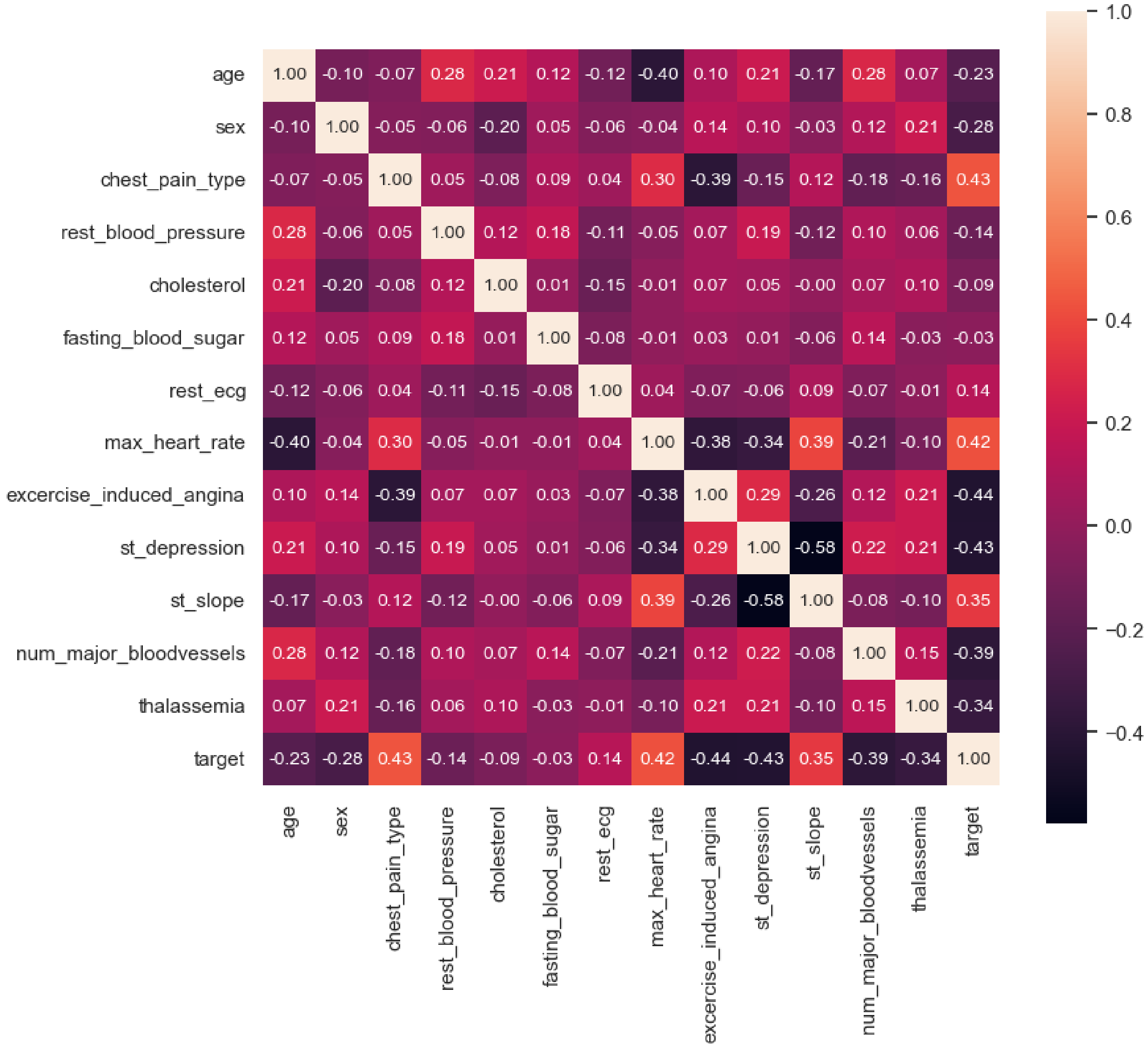


Insight: Asymtomatic patients have higher chances of no heart disease, however all other chest pain types have cases of heart disease with atypical most, then typical and non-agina pain as the least

Multivariate Analysis

In [128]:

```
# We shall use the heatmap to show our correlation
plt.figure(figsize=(9,8))

sns.heatmap(df.corr(), cbar=True, annot=True, square=True, fmt='.2f',annot_kws={'size': 10});
```



Insight: chestpain type, max heart rate and st_slope have a weak positive correlation with heart disease.

Feature Engineering/ Data Pre-Processing

In [129]:

```
# Create a copy of the data (Exclude target/Label alongside other columns that was created) this is because
# we dont need categories in our machine Learning algorithms

df1 = df[['age', 'sex', 'chest_pain_type', 'rest_blood_pressure', 'cholesterol',
       'fasting_blood_sugar', 'rest_ecg', 'max_heart_rate',
       'excercise_induced_angina', 'st_depression', 'st_slope',
       'num_major_bloodvessels', 'thalassemia']]
df1.head(3)
```

Out[129]:

	age	sex	chest_pain_type	rest_blood_pressure	cholesterol	fasting_blood_sugar	rest_ecg	max_heart_rate	excercise_induced_angina
0	63	1	3	145	233	1	0	150	0
1	37	1	2	130	250	0	1	187	0
2	41	0	1	130	204	0	0	172	0

We have dropped all features not required for machine learning

In [130]:

```
label = df[['target']]
label.head(3)
```

Out[130]:

	target
0	1
1	1
2	1

In [131]:

```
# Check the data types to ensure we have all features in only integers and floats all through our dataset
df1.dtypes
```

Out[131]:

```
age                          int64
sex                          int64
chest_pain_type              int64
rest_blood_pressure          int64
cholesterol                  int64
fasting_blood_sugar          int64
rest_ecg                     int64
max_heart_rate               int64
excercise_induced_angina     int64
st_depression              float64
st_slope                     int64
num_major_bloodvessels       int64
thalassemia                  int64
dtype: object
```

In [134]:

```
# We need to do away with outliers in our dataset such as Thalasemia, Cholestrol, Max heart rate,
#rest blood pressure, st_depression and st_slope

# Normalise the date
scaler = MinMaxScaler()

df1['Scaled_RBP'] = scaler.fit_transform(df1['rest_blood_pressure'].values.reshape(-1,1))
df1['Scaled_Chol'] = scaler.fit_transform(df1['cholesterol'].values.reshape(-1,1))
df1['Scaled_Thal'] = scaler.fit_transform(df1['thalassemia'].values.reshape(-1,1))
df1['Scaled_max_heart_rate'] = scaler.fit_transform(df1['max_heart_rate'].values.reshape(-1,1))
df1['Scaled_st_depression'] = scaler.fit_transform(df1['st_depression'].values.reshape(-1,1))
df1['Scaled_st_slope'] = scaler.fit_transform(df1['st_slope'].values.reshape(-1,1))

df1.drop(['rest_blood_pressure','cholesterol','thalassemia','max_heart_rate','st_depression','st_slope'], axis=1, inplace=Tı

df1.head(3)
```

Out[134]:

	age	sex	chest_pain_type	fasting_blood_sugar	rest_ecg	excercise_induced_angina	num_major_bloodvessels	Scaled_RBP	Scaled_C
0	63	1	3	1	0	0	0	0.481132	0.244:
1	37	1	2	0	1	0	0	0.339623	0.283
2	41	0	1	0	0	0	0	0.339623	0.178(

We have scaled our outliers and normalised our outliers to be between 0 and 1 and further dropped the old features while creating a new

MACHINE LEARNING

In [135]:

```
# We have to split the data set into training sets and testing sets
X_train, X_test, y_train, y_test = train_test_split(df1, label, test_size=0.2, random_state=42)
```

In [139]:

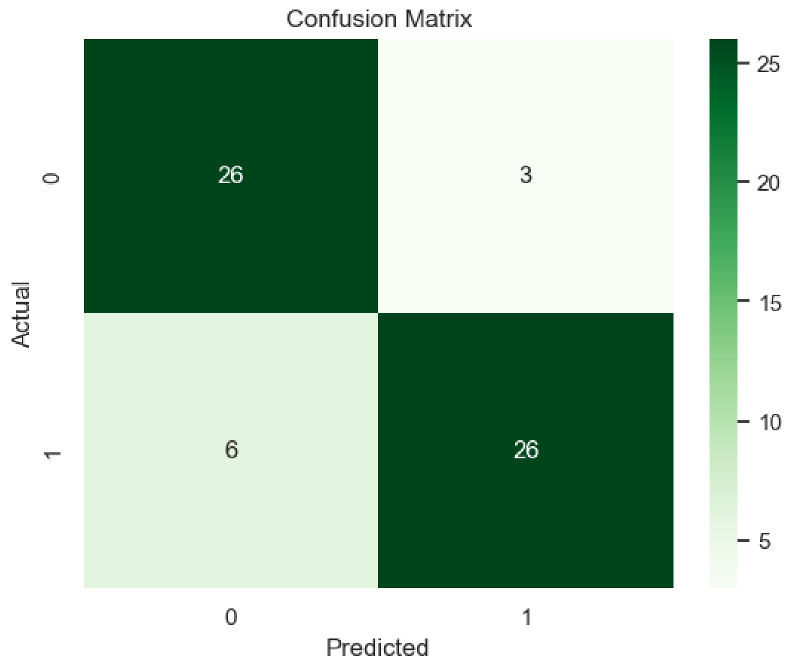
```
# Model Building

# Logistic Regression Model

logreg = LogisticRegression()

logreg.fit(X_train, y_train)

ly_pred = logreg.predict(X_test)

print('Logistic Regression')
print('Accuracy:', accuracy_score(y_test, ly_pred))
print('Precision:', precision_score(y_test, ly_pred))
print('Recall:', recall_score(y_test, ly_pred))
print('F1-score:', f1_score(y_test, ly_pred))
print('AUC-ROC:', roc_auc_score(y_test, ly_pred))
```

```
Logistic Regression
Accuracy: 0.8524590163934426
Precision: 0.896551724137931
Recall: 0.8125
F1-score: 0.8524590163934426
AUC-ROC: 0.8545258620689655
```

We can see that our logistics regression numbers are good, all above 81%

In [140]:

```
# Create a confusion Matrix (lcm = logistics confusion matrix)
lcm = confusion_matrix(y_test, ly_pred)

# Visualise the confusion matrix
sns.heatmap(lcm, annot=True, cmap='Greens', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



Insight: We have 26 patients equally on true positive and true negative. However, we have 6 patients who are false negative (who actually have heart disease but was predicted wrongly as dont have disease) we need to check other models

In [141]:

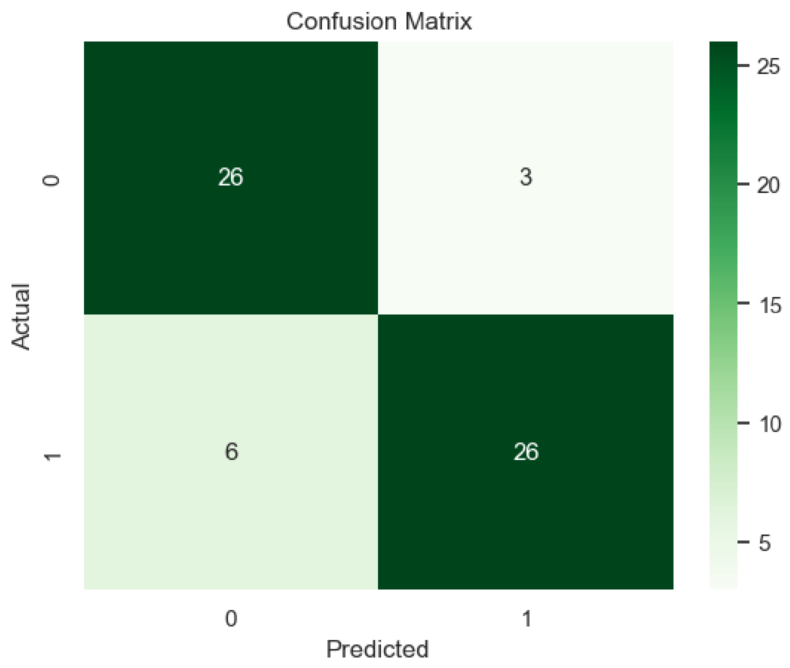
```
# Model Building

#Random Forest Classifier
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
rfy_pred = rfc.predict(X_test)
print('Random Forest Regression')
print('Accuracy:', accuracy_score(y_test, rfy_pred))
print('Precision:', precision_score(y_test, rfy_pred))
print('Recall:', recall_score(y_test, rfy_pred))
print('F1-score:', f1_score(y_test, rfy_pred))
print('AUC-ROC:', roc_auc_score(y_test, rfy_pred))
```

```
Random Forest Regression
Accuracy: 0.8360655737704918
Precision: 0.84375
Recall: 0.84375
F1-score: 0.84375
AUC-ROC: 0.8356681034482758
```

We can see that our random forest regression numbers are also good but is not better than Linear regression

In [142]:

```
# Create a confusion Matrix
rcm = confusion_matrix(y_test, ly_pred)

# Visualise the confusion matrix
sns.heatmap(rcm, annot=True, cmap='Greens', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



Insight: we also have the confusion matrix very comparable with the Linear regression confusion matrix

In [205]:

```
# Machine learning Algorithm will be applied to the dataset

classifiers = [[XGBClassifier(), 'XGB Classifier'],
               [RandomForestClassifier(), 'Random Forest'],
               [KNeighborsClassifier(), 'K-Nearest Neighbours'],
               [SGDClassifier(), 'SDG Classifier'],
               [SVC(), 'SVC'],
               [GaussianNB(), 'Naive Bayes'],
               [DecisionTreeClassifier(random_state = 42), 'Decision tree'],
               [LogisticRegression(), 'Logistic Reression']
               ]
```

In [206]:

```
acc_list = {}
precision_list = {}
recall_list = {}
roc_list = {}


for classifier in classifiers:
    model = classifier[0]
    model.fit(X_train, y_train)
    model_name = classifier[1]

    pred = model.predict(X_test)

    a_score = accuracy_score(y_test, pred)
    p_score = precision_score(y_test, pred)
    r_score = recall_score(y_test, pred)
    roc_score = roc_auc_score(y_test, pred)

    acc_list[model_name] = ([str(round(a_score*100, 2)) + '%'])
    precision_list[model_name] = ([str(round(p_score*100, 2)) + '%'])
    recall_list[model_name] =  ([str(round(r_score*100, 2)) + '%'])
    roc_list[model_name] =  ([str(round(roc_score*100, 2)) + '%'])

    if model_name != classifiers[-1][1]:
        print('')
```

In [207]:

```
acc_list
```

Out[207]:

```
{'XGB Classifier': ['81.97%'],
 'Random Forest': ['86.89%'],
 'K-Nearest Neighbours': ['75.41%'],
 'SDG Classifier': ['60.66%'],
 'SVC': ['65.57%'],
 'Naive Bayes': ['86.89%'],
 'Decision tree': ['83.61%'],
 'Logistic Reression': ['85.25%']}
```

In [217]:

```
table1 = pd.DataFrame(acc_list)
table1.head()
```

Out[217]:

	XGB Classifier	Random Forest	K-Nearest Neighbours	SDG Classifier	SVC	Naive Bayes	Decision tree	Logistic Reression
0	81.97%	86.89%	75.41%	60.66%	65.57%	86.89%	83.61%	85.25%

In [209]:

```
table2 = pd.DataFrame(precision_list)
table2.head()
```

Out[209]:

	XGB Classifier	Random Forest	K-Nearest Neighbours	SDG Classifier	SVC	Naive Bayes	Decision tree	Logistic Reression
0	86.21%	87.5%	75.76%	83.33%	66.67%	90.0%	89.29%	89.66%

In [210]:

```
table3 = pd.DataFrame(recall_list)
table3
```

Out[210]:

	XGB Classifier	Random Forest	K-Nearest Neighbours	SDG Classifier	SVC	Naive Bayes	Decision tree	Logistic Reression
0	78.12%	87.5%	78.12%	31.25%	68.75%	84.38%	78.12%	81.25%

In [211]:

```
table4 = pd.DataFrame(roc_list)
table4
```

Out[211]:

	XGB Classifier	Random Forest	K-Nearest Neighbours	SDG Classifier	SVC	Naive Bayes	Decision tree	Logistic Reression
0	82.17%	86.85%	75.27%	62.18%	65.41%	87.02%	83.89%	85.45%

In [212]:

```
Con = pd.concat([table1, table2, table3, table4])
Con
```

Out[212]:

	XGB Classifier	Random Forest	K-Nearest Neighbours	SDG Classifier	SVC	Naive Bayes	Decision tree	Logistic Reression
0	81.97%	86.89%	75.41%	60.66%	65.57%	86.89%	83.61%	85.25%
0	86.21%	87.5%	75.76%	83.33%	66.67%	90.0%	89.29%	89.66%
0	78.12%	87.5%	78.12%	31.25%	68.75%	84.38%	78.12%	81.25%
0	82.17%	86.85%	75.27%	62.18%	65.41%	87.02%	83.89%	85.45%

Insight: naive bayes approach have the best stats overall when compared to all the other classifiers used to test our model