```
Student Names: Chirag Handa & Bhavesh Waghela
Student IDs:     N01604260  &    N01639685
```

Heart Disease Analysis

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, roc_curve, classification_report
```

Loading the dataset

In [2]:
```
df = pd.read_csv('heart.csv')
df.head()
```

Out[2]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

Data Exploration

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   918 non-null    object 
 3   RestingBP       918 non-null    int64  
 4   Cholesterol     918 non-null    int64  
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    object 
 7   MaxHR           918 non-null    int64  
 8   ExerciseAngina  918 non-null    object 
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object 
 11  HeartDisease    918 non-null    int64  
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

In [4]:
```
df.describe()
```

Out[4]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

Data Cleaning and Preprocessing

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Age              0
Sex              0
ChestPainType    0
RestingBP        0
Cholesterol      0
FastingBS        0
RestingECG       0
MaxHR            0
ExerciseAngina   0
Oldpeak          0
ST_Slope         0
HeartDisease     0
dtype: int64
```

In [6]:
```
mean_ch = df[df['Cholesterol'] != 0].mean()

df['Cholesterol'].replace(0, mean_ch['Cholesterol'], inplace = True)
df['RestingBP'].replace(0, mean_ch['RestingBP'], inplace = True)
```

```
C:\Users\Chirag Handa\AppData\Local\Temp\ipykernel_564\3118447426.py:1: FutureWarning: The default value of numeric_on
ly in DataFrame.mean is deprecated. In a future version, it will default to False. In addition, specifying 'numeric_on
ly=None' is deprecated. Select only valid columns or specify the value of numeric_only to silence this warning.
  mean_ch = df[df['Cholesterol'] != 0].mean()
```

In [7]:
```
df
```

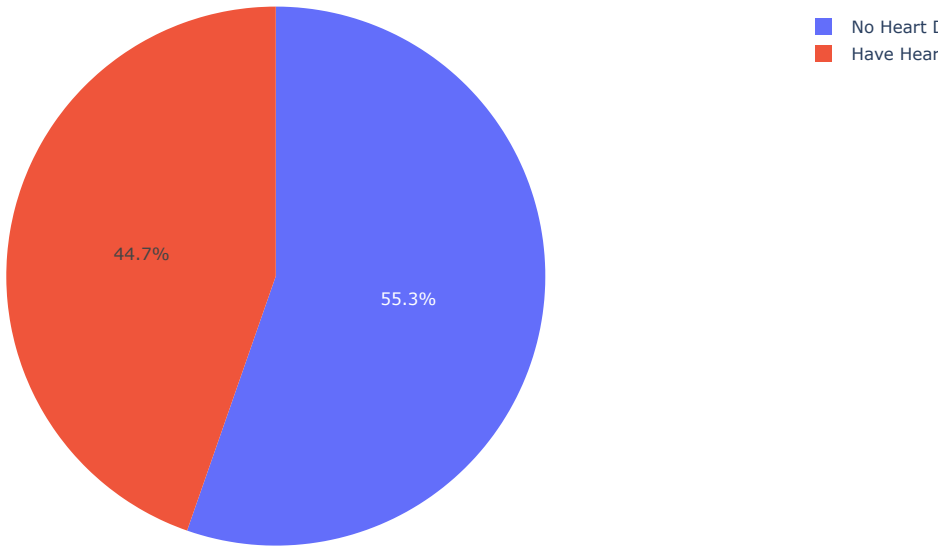
Out[7]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140.0	289.0	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160.0	180.0	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130.0	283.0	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138.0	214.0	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150.0	195.0	0	Normal	122	N	0.0	Up	0
...	...	...	...	...	...	...	...	...	...	...	...	...
913	45	M	TA	110.0	264.0	0	Normal	132	N	1.2	Flat	1
914	68	M	ASY	144.0	193.0	1	Normal	141	N	3.4	Flat	1
915	57	M	ASY	130.0	131.0	0	Normal	115	Y	1.2	Flat	1
916	57	F	ATA	130.0	236.0	0	LVH	174	N	0.0	Flat	1
917	38	M	NAP	138.0	175.0	0	Normal	173	N	0.0	Up	0

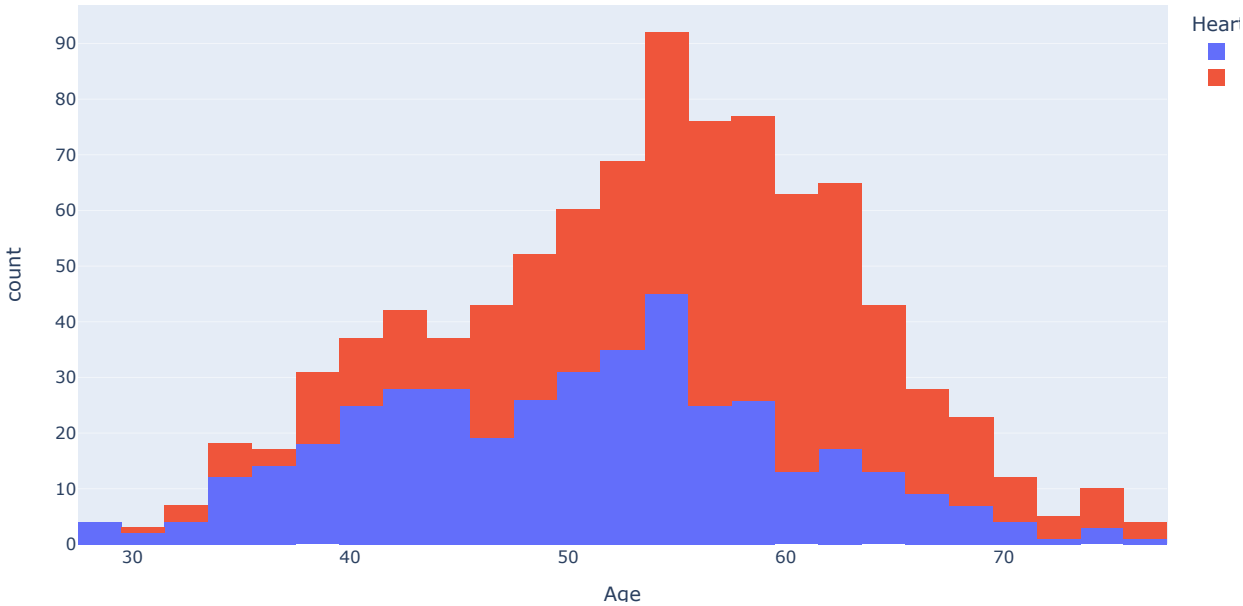
918 rows × 12 columns

Data Analysis

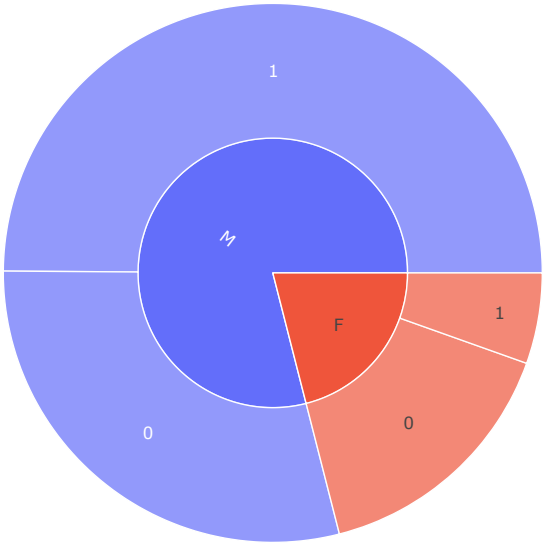
In [8]:
```
px.pie(values=df['HeartDisease'].value_counts(),names={ 0 : 'No Heart Disease',1:'Have Heart Disease'})
```



In [9]:
```
px.histogram(df,x='Age',color='HeartDisease')
```



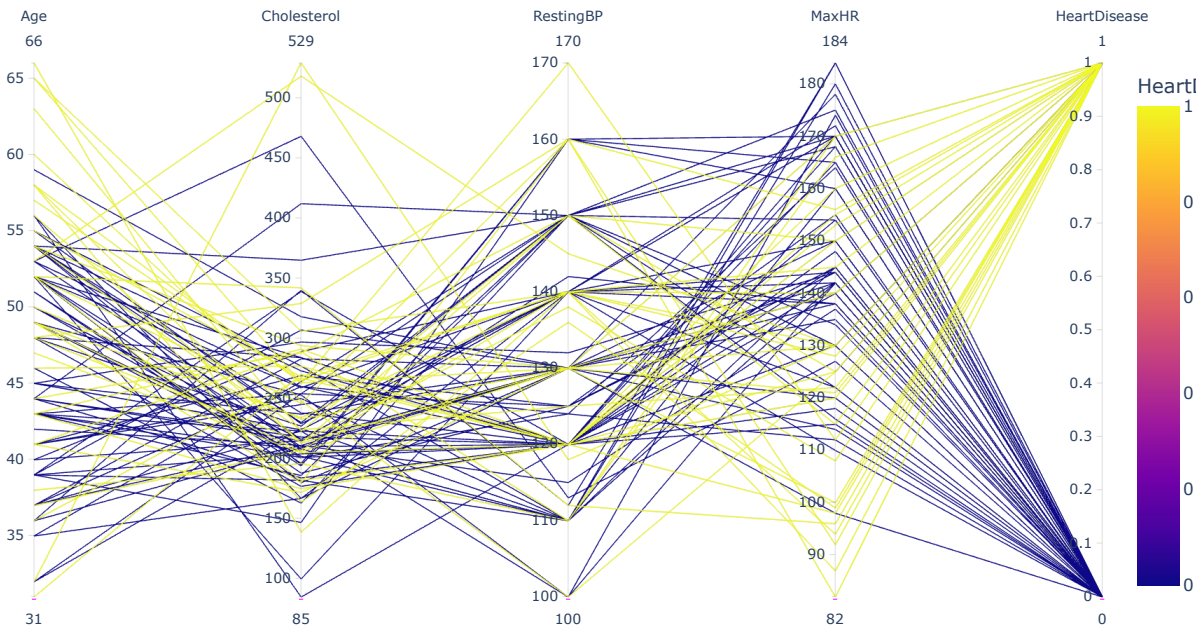
In [10]:
```
px.sunburst(df, path=['Sex','HeartDisease'])
```



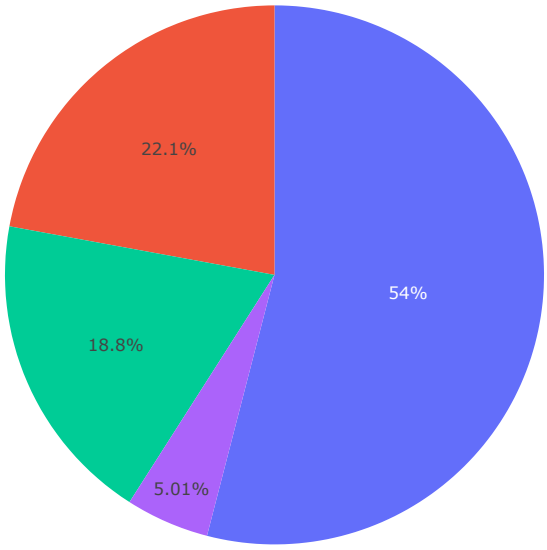
In [11]:
```
cols = ['Age', 'Cholesterol', 'RestingBP', 'MaxHR', 'HeartDisease']
df_new = df[cols]
```

In [12]:
```
px.parallel_coordinates(df_new[0:100], color="HeartDisease")
```

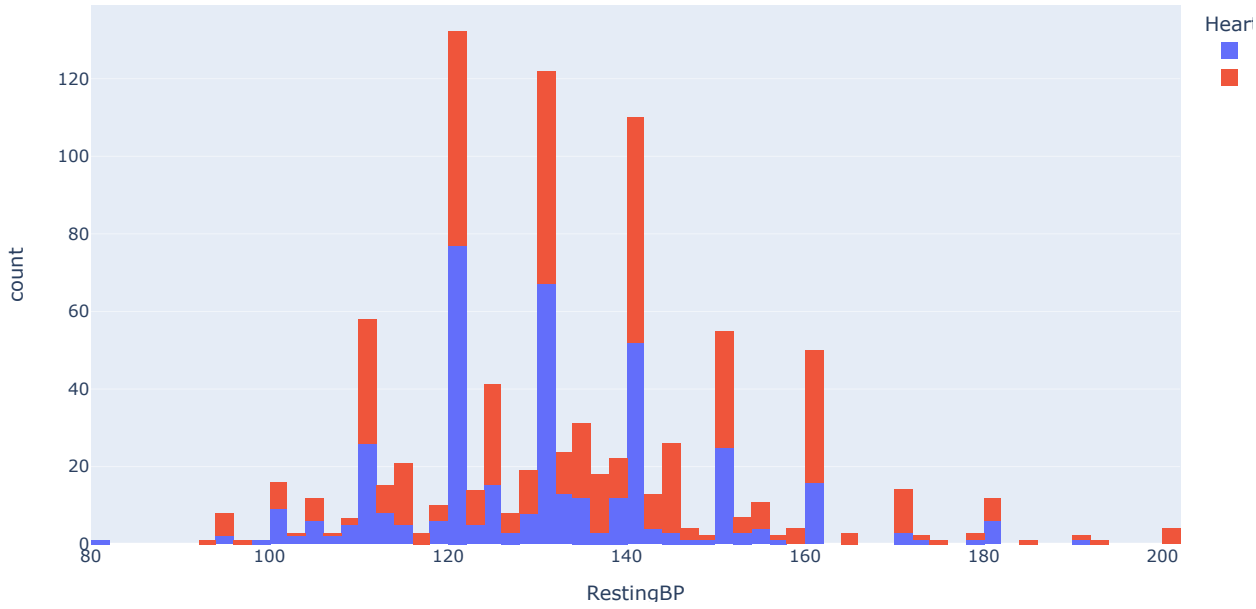
```
C:\Users\Chirag Handa\anaconda3\Lib\site-packages\plotly\express\_core.py:279: FutureWarning:

iteritems is deprecated and will be removed in a future version. Use .items instead.
```



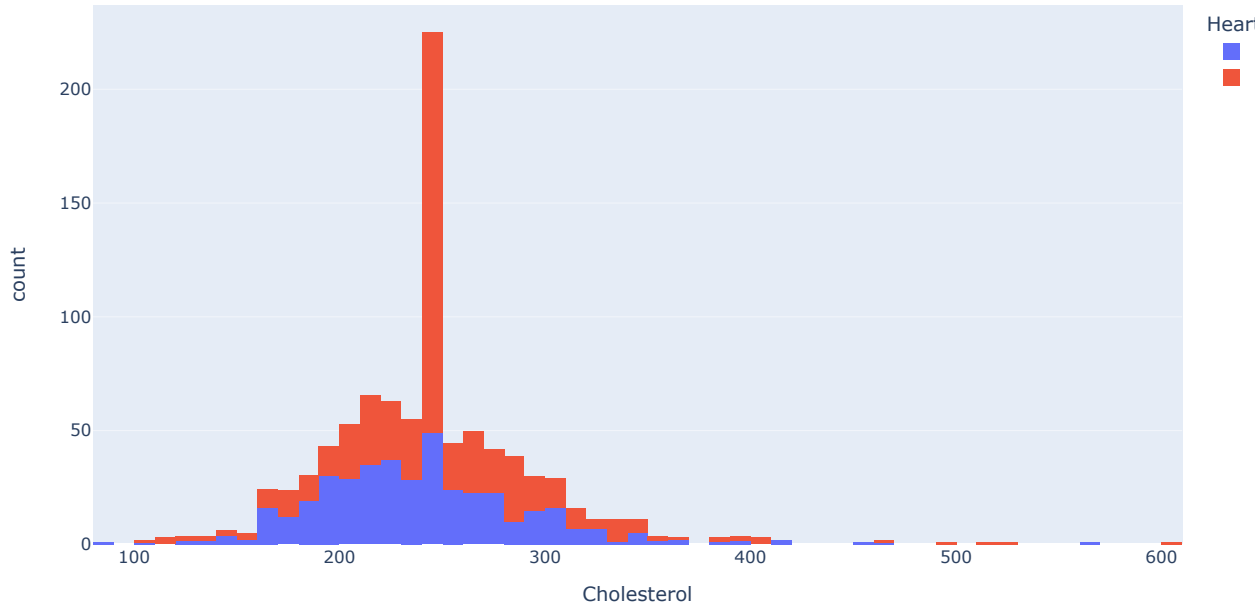
In [13]:
```
px.pie(values=df['ChestPainType'].value_counts(),names =df['ChestPainType'].value_counts().index)
```



In [14]:
```
px.histogram(df, x='RestingBP', color='HeartDisease',nbins=100)
```

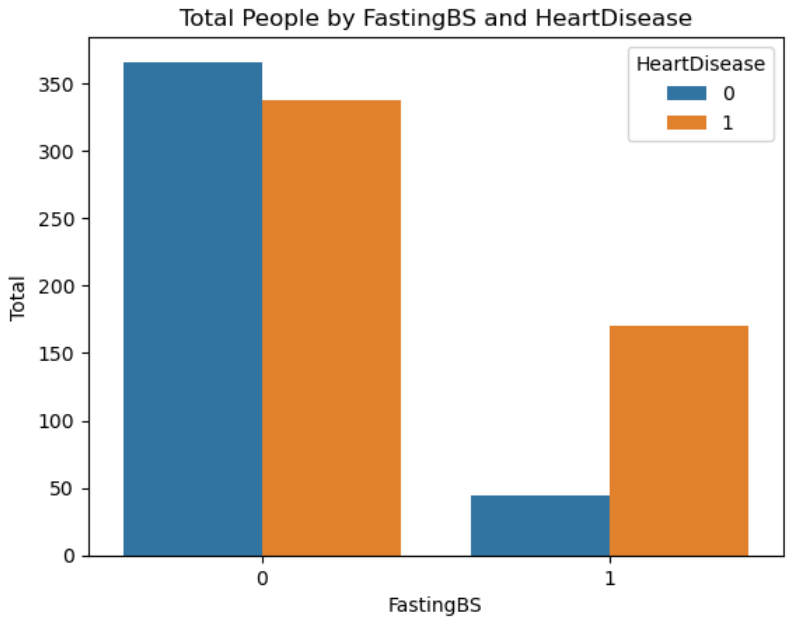


In [15]:
```
px.histogram(df,x='Cholesterol',color='HeartDisease',nbins=100)
```

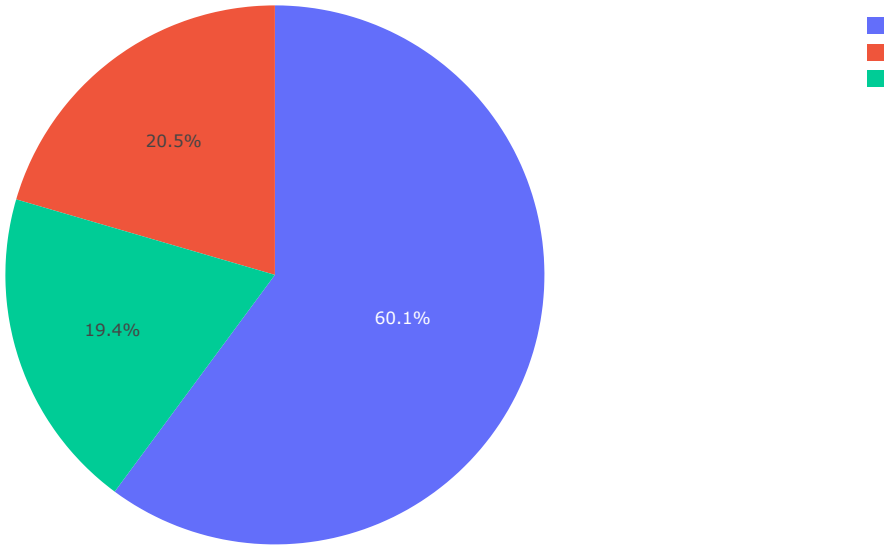


In [16]:
```
df_fasting_heart = df.groupby(['FastingBS','HeartDisease'])['FastingBS'].count().reset_index(name='Total')
sns.barplot(df_fasting_heart,x='FastingBS',y='Total',hue='HeartDisease')
plt.title('Total People by FastingBS and HeartDisease')
```

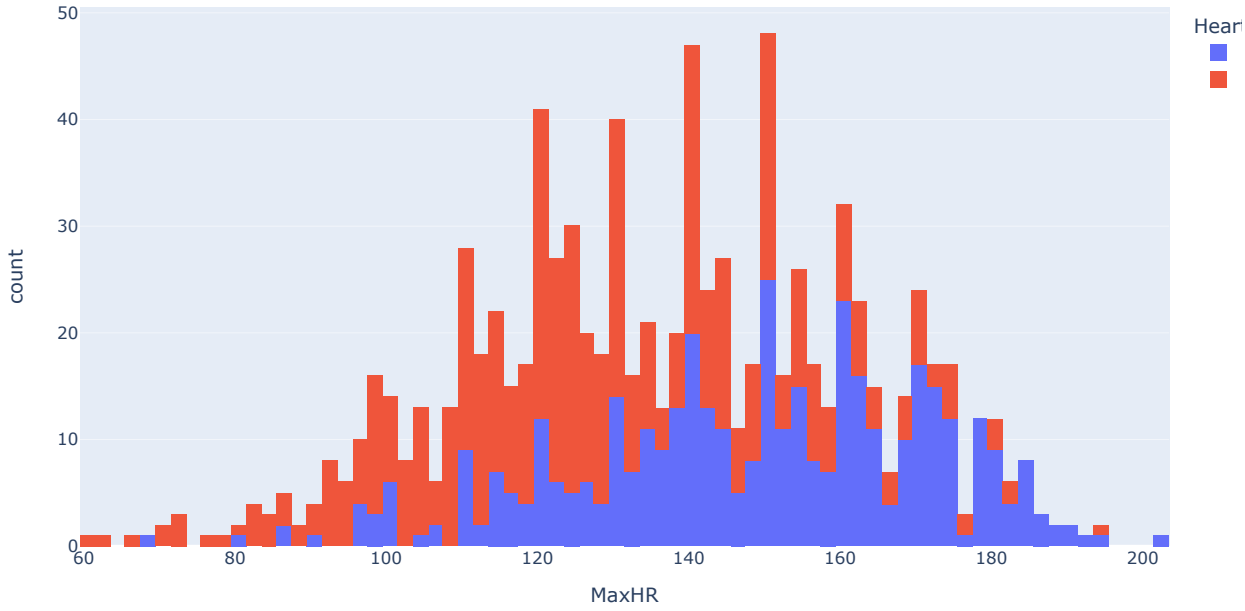
Out[16]: Text(0.5, 1.0, 'Total People by FastingBS and HeartDisease')



In [17]:
```
px.pie(values=df['RestingECG'].value_counts(),names=df['RestingECG'].value_counts().index)
```

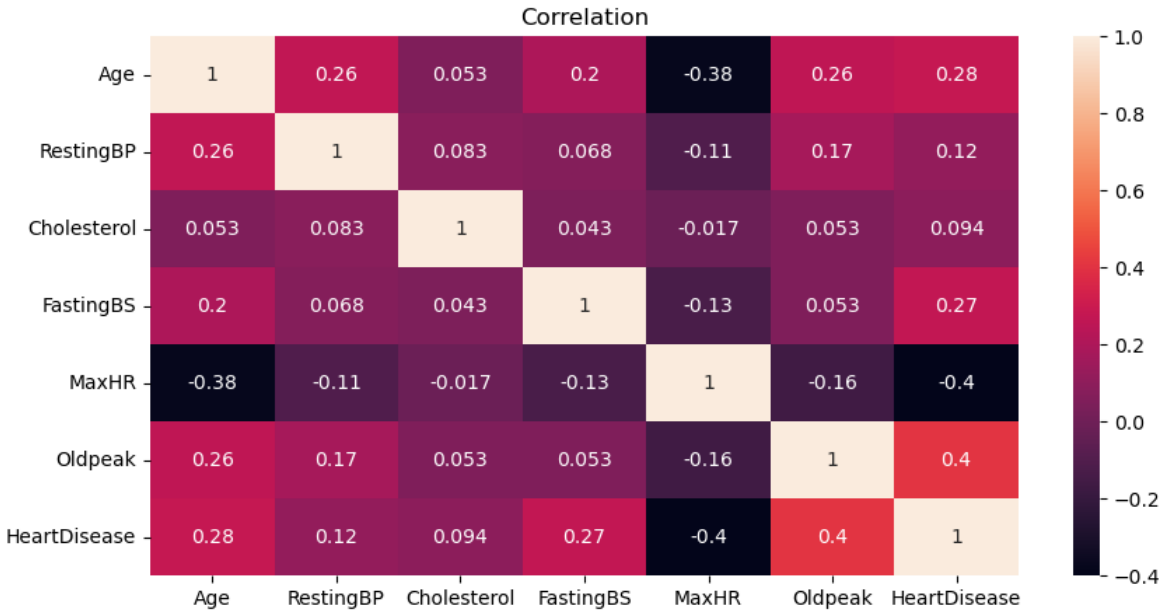


In [18]:
```
px.histogram(df,x='MaxHR',color='HeartDisease',nbins=100)
```



In [19]:
```
plt.figure(figsize=(10,5))
sns.heatmap(df.select_dtypes('number').corr(),annot=True)
plt.title('Correlation')
```

Out[19]:
```
Text(0.5, 1.0, 'Correlation')
```



Identifying Dependent and Independent Variables

In [20]:
```
df_encoded = pd.get_dummies(df, columns=['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope'])
df_encoded
```

Out[20]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease	Sex_F	Sex_M	ChestPainType_ASY	...	ChestPainType_NAP	ChestPa
0	40	140.0	289.0	0	172	0.0	0	0	1	0	...	0	
1	49	160.0	180.0	0	156	1.0	1	1	0	0	...	1	
2	37	130.0	283.0	0	98	0.0	0	0	1	0	...	0	
3	48	138.0	214.0	0	108	1.5	1	1	0	1	...	0	
4	54	150.0	195.0	0	122	0.0	0	0	1	0	...	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	
913	45	110.0	264.0	0	132	1.2	1	0	1	0	...	0	
914	68	144.0	193.0	1	141	3.4	1	0	1	1	...	0	
915	57	130.0	131.0	0	115	1.2	1	0	1	1	...	0	
916	57	130.0	236.0	0	174	0.0	1	1	0	0	...	0	
917	38	138.0	175.0	0	173	0.0	0	0	1	0	...	1	

918 rows × 21 columns

In [21]:
```
X = df_encoded.iloc[:,:-1].values
y = df_encoded.iloc[:,-1].values
```

Splitting the data into training and testing sets

In [22]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 1)
```

In [23]:
```
sc = StandardScaler()
X_train[:, [9,10,11,13,14]] = sc.fit_transform(X_train[:, [9,10,11,13,14]])
X_test[:, [9,10,11,13,14]] = sc.transform(X_test[:, [9,10,11,13,14]])
```

Training the Logistic Regression Model

In [24]:
```
lr = LogisticRegression()
lr.fit(X_train,y_train)
```

```
C:\Users\Chirag Handa\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning:

lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
```

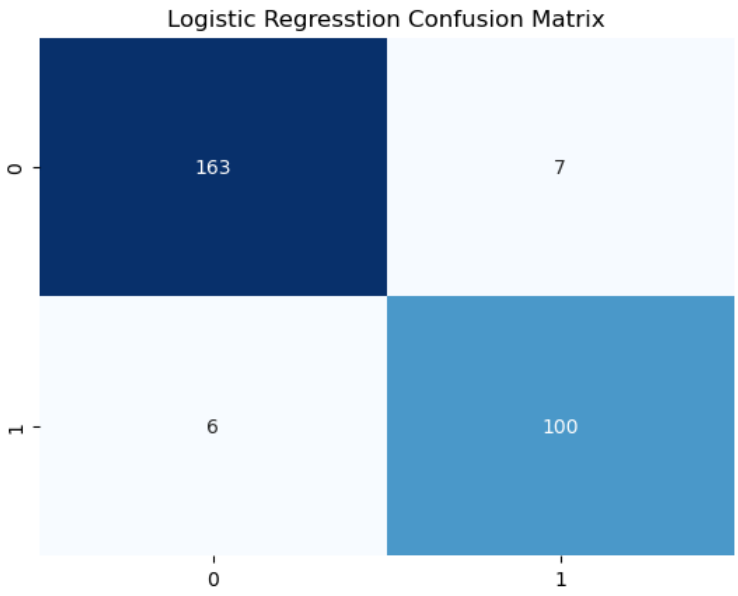
Out[24]:
```
▾ LogisticRegression

LogisticRegression()
```

Evaluation Metrics - Logistic Regression

In [25]:
```
y_pred = lr.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
lr_train_acc = round(accuracy_score(y_train,lr.predict(X_train))*100,2)
lr_test_acc = round(accuracy_score(y_test,y_pred)*100,2)
print('Accuracy = ' , lr_test_acc,' %')
sns.heatmap(cm,annot=True, fmt='d', cmap='Blues', cbar=False,)
plt.title('Logistic Regresstion Confusion Matrix')
```

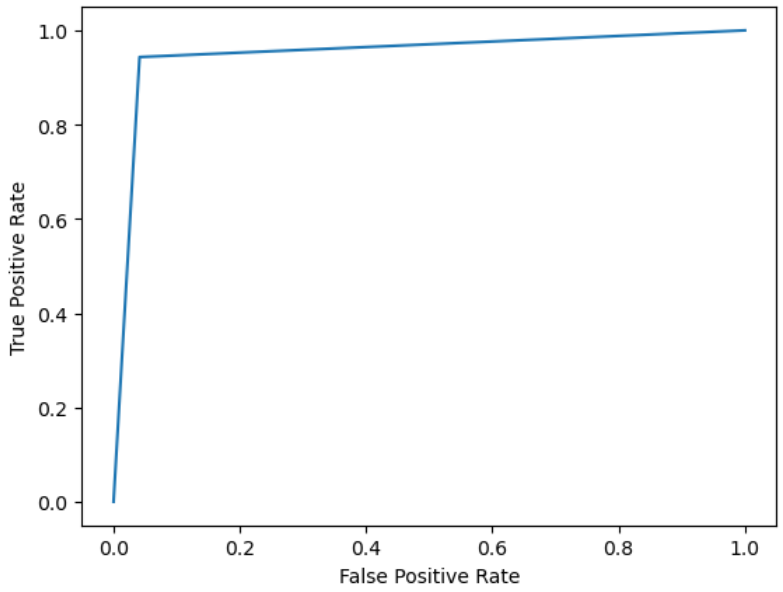
```
Accuracy =  95.29  %
```

Out[25]: Text(0.5, 1.0, 'Logistic Regresstion Confusion Matrix')



ROC Curve

In [26]:
```
def plot_roc_curve(y_test, y_pred):
    fpr, tpr, thresholds = roc_curve(y_test, y_pred)
    plt.plot(fpr, tpr)
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')

plot_roc_curve(y_test, y_pred)
```



Classification Report

In [27]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       170
           1       0.93      0.94      0.94       106

    accuracy                           0.95       276
   macro avg       0.95      0.95      0.95       276
weighted avg       0.95      0.95      0.95       276
```

Conclusion & Inference

From the Correlation Matrix we can observe that MaxHR has a highly negative correlation with heart disease. This means that with high Heart Rate, a person is at a greater risk of getting a heart disease. Stress is a big factor in determining Heart Rate so we should try to remain as calm and stress-free as possible for a healthy life.