Cardiovascular Disease Data EDA and Machine Learning Prediction

In [9]:
```
# Loading the dataset
df = pd.read_csv('C:/Users/zizhe/Desktop/Leo Zhao/06Cardiovascular Disease Prediction/Data.csv')
df.head(8)
```

Out[9]:

	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression	Diabetes	Arthritis	Sex	Age_Category	Heig
0	Poor	Within the past 2 years	No	No	No	No	No	No	Yes	Female	70-74	
1	Very Good	Within the past year	No	Yes	No	No	No	Yes	No	Female	70-74	
2	Very Good	Within the past year	Yes	No	No	No	No	Yes	No	Female	60-64	
3	Poor	Within the past year	Yes	Yes	No	No	No	Yes	No	Male	75-79	
4	Good	Within the past year	No	No	No	No	No	No	No	Male	80+	
5	Good	Within the past year	No	No	No	No	Yes	No	Yes	Male	60-64	
6	Fair	Within the past year	Yes	Yes	No	No	No	No	Yes	Male	60-64	
7	Good	Within the past year	Yes	No	No	No	No	No	Yes	Female	65-69	

Data Preprocessing

In [10]:
```
# Checking the shape of the dataset
df.shape
```

Out[10]:
```
(308854, 19)
```

In [11]:
```
# Checking for null/missing values
df.isnull().sum()
```

Out[11]:
```
General_Health                  0
Checkup                         0
Exercise                        0
Heart_Disease                   0
Skin_Cancer                     0
Other_Cancer                    0
Depression                      0
Diabetes                        0
Arthritis                       0
Sex                             0
Age_Category                    0
Height_(cm)                     0
Weight_(kg)                     0
BMI                             0
Smoking_History                 0
Alcohol_Consumption             0
Fruit_Consumption               0
Green_Vegetables_Consumption    0
FriedPotato_Consumption         0
dtype: int64
```

In [12]:
```
# Checking the datatypes
df.dtypes
```

Out[12]:
```
General_Health                 object
Checkup                        object
Exercise                       object
Heart_Disease                  object
Skin_Cancer                    object
Other_Cancer                   object
Depression                     object
Diabetes                       object
Arthritis                      object
Sex                            object
Age_Category                   object
Height_(cm)                   float64
Weight_(kg)                   float64
BMI                           float64
Smoking_History                object
Alcohol_Consumption           float64
Fruit_Consumption             float64
Green_Vegetables_Consumption  float64
FriedPotato_Consumption       float64
dtype: object
```

In [13]:
```
#Check the number of unique value from all of the object datatype
df.select_dtype(include='object').nunique()
```

Out[13]:
```
General_Health      5
Checkup             5
Exercise            2
Heart_Disease       2
Skin_Cancer         2
Other_Cancer        2
Depression          2
Diabetes            4
Arthritis           2
Sex                 2
Age_Category       13
Smoking_History     2
dtype: int64
```

The dataset has columns for weight, height, and BMI. However, the BMI column is calculated using the weight and height columns. Hence, the weight and height columns are dropped from the dataset.

In [14]:
```
# Drop Column
df.drop(column=['Weight_(kg)', 'Height_(cm)'], inplace=True)
```

In [15]:
```
# Unique values in each column
for i in df.columns:
    print(i, df[i].unique())
```

```
General_Health ['Poor' 'Very Good' 'Good' 'Fair' 'Excellent']
Checkup ['Within the past 2 years' 'Within the past year' '5 or more years ago'
 'Within the past 5 years' 'Never']
Exercise ['No' 'Yes']
Heart_Disease ['No' 'Yes']
Skin_Cancer ['No' 'Yes']
Other_Cancer ['No' 'Yes']
Depression ['No' 'Yes']
Diabetes ['No' 'Yes' 'No, pre-diabetes or borderline diabetes'
 'Yes, but female told only during pregnancy']
Arthritis ['Yes' 'No']
Sex ['Female' 'Male']
Age_Category ['70-74' '60-64' '75-79' '80+' '65-69' '50-54' '45-49' '18-24' '30-34'
 '55-59' '35-39' '40-44' '25-29']
BMI [14.54 28.29 33.47 ... 63.83 19.09 56.32]
Smoking_History ['Yes' 'No']
Alcohol_Consumption [ 0.  4.  3.  8. 30.  2. 12.  1.  5. 10. 20. 17. 16.  6. 25. 28. 15.  7.
  9. 24. 11. 29. 27. 14. 21. 23. 18. 26. 22. 13. 19.]
Fruit_Consumption [ 30.  12.   8.  16.   2.   1.  60.   0.   7.   5.   3.   6.  90.  28.
  20.   4.  80.  24.  15.  10.  25.  14. 120.  32.  40.  17.  45. 100.
   9.  99.  96.  35.  50.  56.  48.  27.  72.  36.  84.  26.  23.  18.
  21.  42.  22.  11. 112.  29.  64.  70.  33.  76.  44.  39.  75.  31.
  92. 104.  88.  65.  55.  13.  38.  63.  97. 108.  19.  52.  98.  37.
  68.  34.  41. 116.  54.  62.  85.]
Green_Vegetables_Consumption [ 16.   0.   3.  30.   4.  12.   8.  20.   1.  10.   5.   2.   6.  60.
  28.  25.  14.  40.   7.  22.  24.  15. 120.  90.  19.  13.  11.  80.
  27.  17.  56.  18.   9.  21.  99.  29.  31.  45.  23. 100. 104.  32.
  48.  75.  36.  35. 112.  26.  50.  33.  96.  52.  76.  84.  34.  97.
  88.  98.  68.  92.  55.  95.  64. 124.  61.  65.  77.  85.  44.  39.
  70.  93. 128.  37.  53.]
FriedPotato_Consumption [ 12.   4.  16.   8.   0.   1.   2.  30.  20.  15.  10.   3.   7.  28.
   5.   9.   6. 120.  32.  14.  60.  33.  48.  25.  24.  21.  90.  13.
  99.  17.  18.  40.  56.  34.  36.  44. 100.  11.  64.  45.  80.  29.
  68.  26.  50.  22.  95.  23.  27. 112.  35.  31.  98.  96.  88.  92.
  19.  76.  49.  97. 128.  41.  37.  42.  52.  72.  46. 124.  84.]
```

The diabetes column contains four values: 'Yes,' 'No,' 'No pre-diabetes or borderline diabetes,' and 'Yes, but female told only during pregnancy.' We will replace the last two values with 'pre-diabetes' and 'gestational diabetes,' respectively.

In [16]:
```
df['Diabetes'] = df['Diabetes']map({'No, pre-diabetes or borderline diabetes': 'Pre-Diabetes' ,'Yes, but female told on
```

Descriptive Statistics

In [19]:
```
df.describe()
```

Out[19]:

	BMI	Alcohol_Consumption	Fruit_Consumption	Green_Vegetables_Consumption	FriedPotato_Consumption
count	186777.000000	186777.000000	186777.000000	186777.000000	186777.000000
mean	28.303577	2.505287	18.446104	11.893440	4.899565
std	5.433758	3.777076	10.898445	9.604871	4.261893
min	12.870000	0.000000	0.000000	0.000000	0.000000
25%	24.370000	0.000000	8.000000	4.000000	2.000000
50%	27.550000	0.000000	16.000000	8.000000	4.000000
75%	31.750000	4.000000	30.000000	16.000000	8.000000
max	43.280000	15.000000	56.000000	44.000000	17.000000

In [20]:
```
df.head()
```

Out[20]:

	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression	Diabetes	Arthritis	Sex	Age_Category	BM
0	Poor	Within the past 2 years	No	No	No	No	No	No	Yes	Female	70-74	14.5
1	Very Good	Within the past year	No	Yes	No	No	No	Yes	No	Female	70-74	28.2
2	Very Good	Within the past year	Yes	No	No	No	No	Yes	No	Female	60-64	33.4
3	Poor	Within the past year	Yes	Yes	No	No	No	Yes	No	Male	75-79	28.7
4	Good	Within the past year	No	No	No	No	No	No	No	Male	80+	24.3

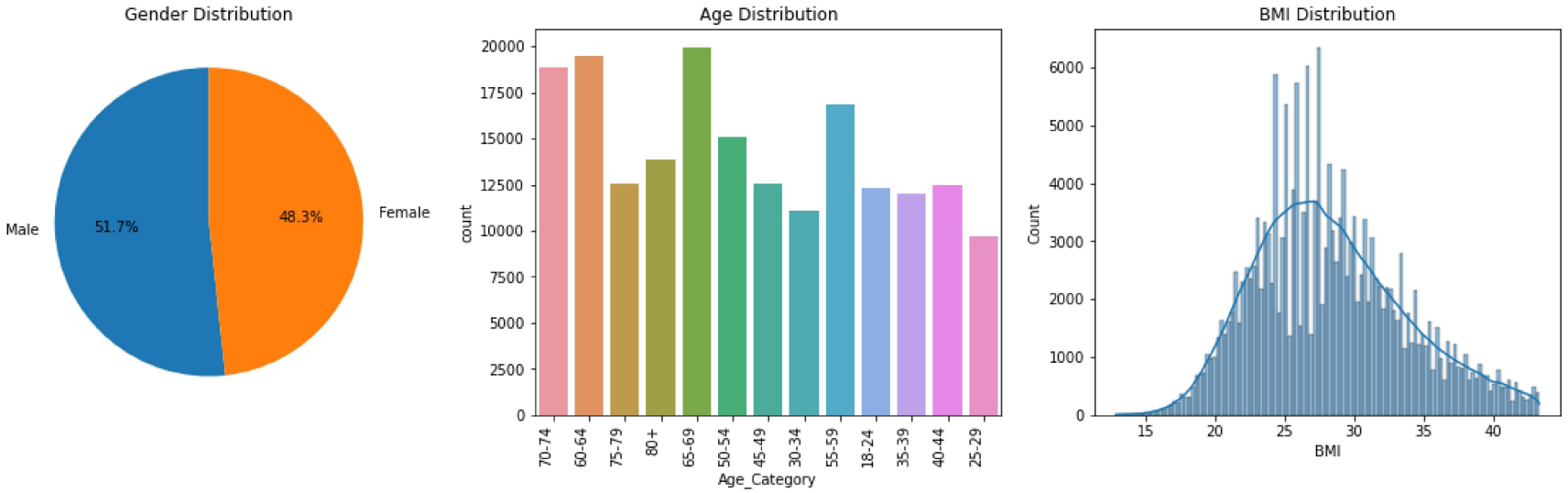
Exploratory Data Analysis

In the exploratory data analysis, I will examine the dataset to understand it better and analyze the relationship between the features and the target variable. I'll start by assessing the data's distribution across all variables and then investigate the relationship between the features and the target variable.

Patient demographics

In [21]:
```
fig, ax = plt.subplots(1,3,figsize=(20, 5))
ax[0].pie(df['Sex'].value_counts(), labels = ['Male', 'Female'], startangle=90)
ax[0].set_title('Gender Distribution')
sns.countplot(x = 'Age_Category', data = df, ax = ax[1]).set_title('Age Distribution')
ax[1].set_xticklabels(ax[1].get_xticklabels(),)
sns.histplot(x = 'BMI', data = df, ax = ax[2]).set_title('BMI Distribution')
```

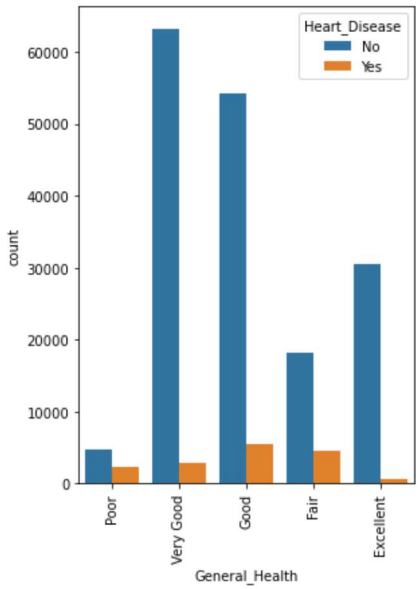
Out[21]:
```
Text(0.5, 1.0, 'BMI Distribution')
```

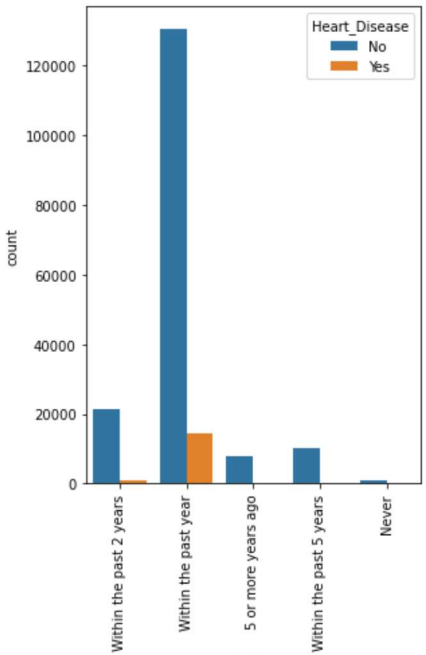


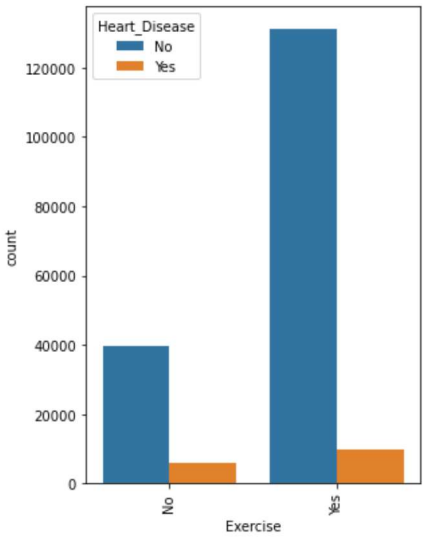
The three graphs above illustrate the patient demographics in the dataset. The pie chart indicates that the majority of patients are male (51.7%), followed closely by females (48.3%). Examining the age distribution, we observe that most patients are older than 45 years, indicating a skew toward older individuals in the dataset. The histogram of BMI reveals that most patients fall within the BMI range of 25 to 30, indicating that a majority are overweight. Based on these observations, I hypothesize that patients with higher BMI values are more likely to have cardiovascular disease.

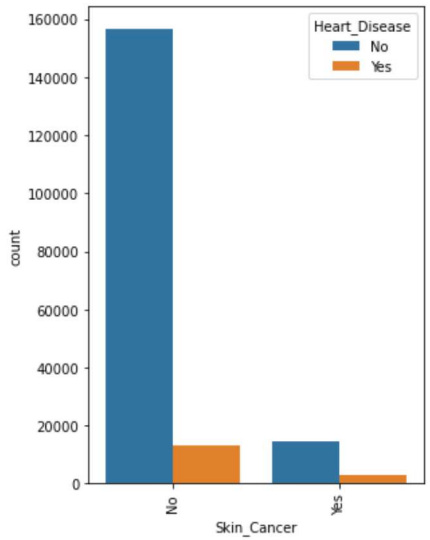
In [22]:
```
# list of categorical variables to plot
cat_vars = ['General_Health', 'Checkup', 'Exercise', 'Skin_Cancer', 'Other_Cancer',
            'Depression', 'Diabetes', 'Arthritis', 'Sex', 'Age_Category', 'Smoking_History']

# create figure with subplots
fig, axs = plt.subplots(figsize=(18, 20))
axs = axs.flatten()

# create barplot for each categorical variable
for i, var in enumerate(cat_vars):
    sns.countplot(x=var, hue='Heart_Disease', data=df)
    axs[i].set_xticklabels(axs[i].get_xticklabels(), rotation=60)

# adjust spacing between subplots
fig.tight_layout()

# remove the 12th subplot
fig.delaxes(axs[11])
```

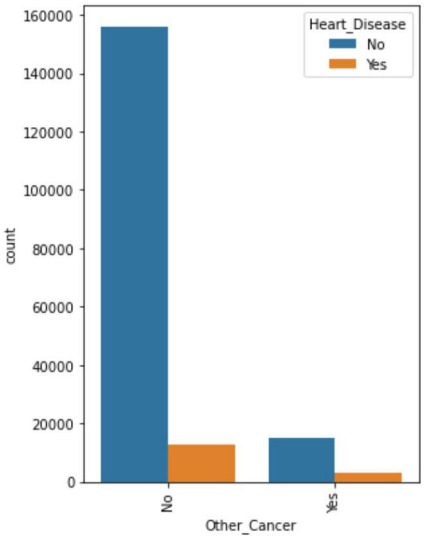
```
# show plot
plt.show()
```

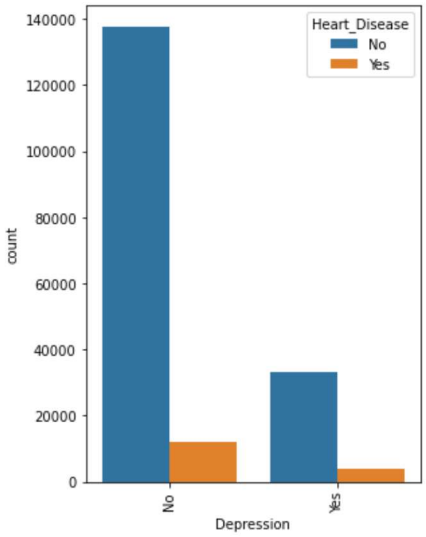

Heart_Disease
No
Yes
count
General_Health
Poor
Very Good
Good
Fair
Excellent

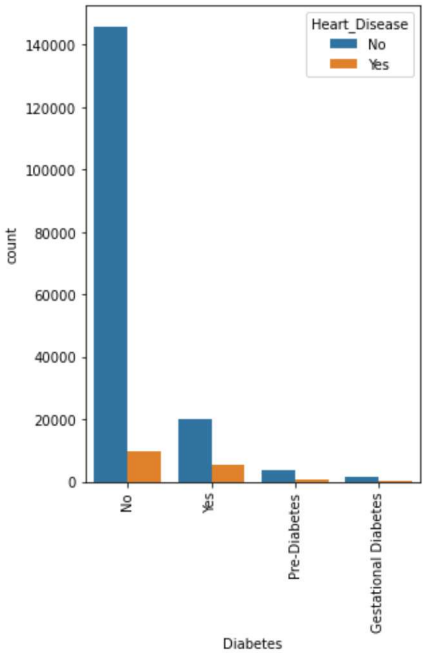

Heart_Disease
No
Yes
count
Checkup
Within the past 2 years
Within the past year
5 or more years ago
Within the past 5 years
Never

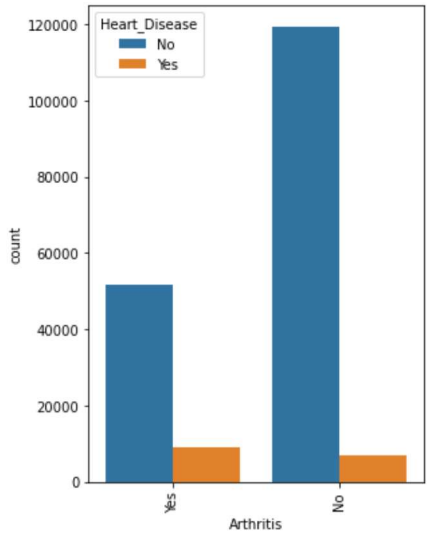

Heart_Disease
No
Yes
count
Exercise
No
Yes


Heart_Disease
No
Yes
count
Skin_Cancer
No
Yes


Heart_Disease
No
Yes
count
Other_Cancer
No
Yes

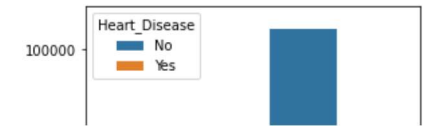

Heart_Disease
No
Yes
count
Depression
No
Yes

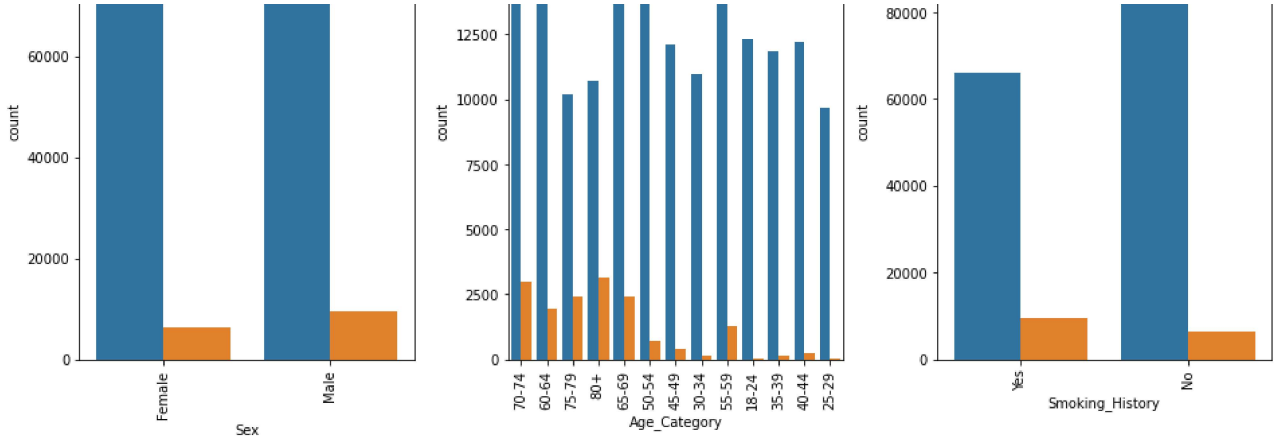

Heart_Disease
No
Yes
count
Diabetes
No
Yes
Pre-Diabetes
Gestational Diabetes


Heart_Disease
No
Yes
count
Arthritis
Yes
No


Heart_Disease
No
Yes


Heart_Disease
No
Yes

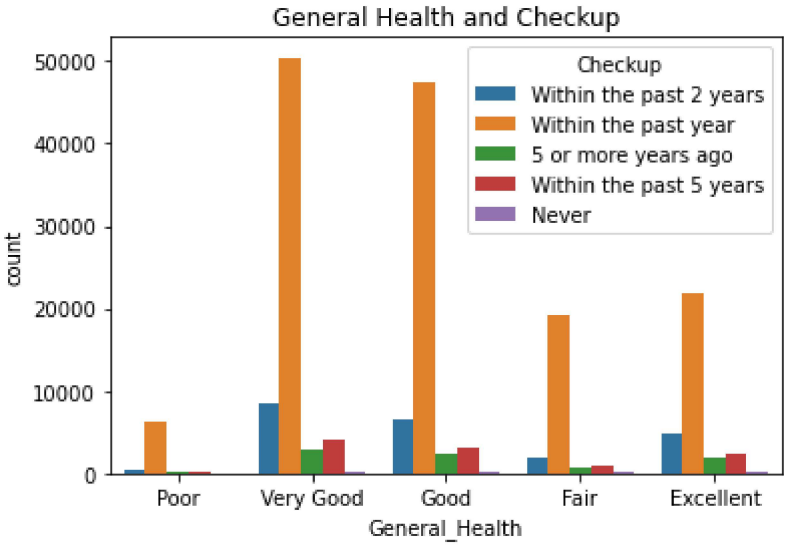

Heart_Disease
No
Yes



General Health and Last Checkup

In [17]:
```
sns.countplot(x = 'General_Health', data = df).set_title('General Health and Checkup')
```

Out[17]:
```
Text(0.5, 1.0, 'General Health and Checkup')
```



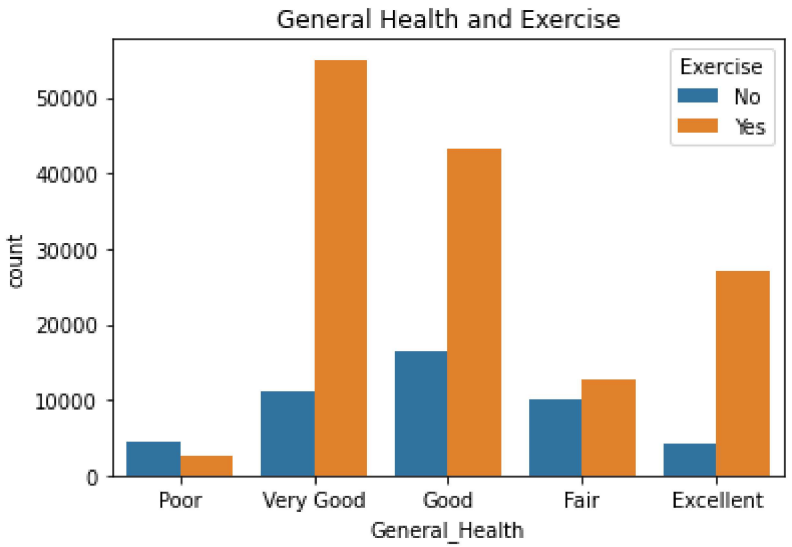
Based on this graph, the majority of people in the dataset report either good or very good general health, with a notable number reporting excellent general health. This suggests that most individuals in the dataset are in good health, while very few report poor general health.

When examining the timing of their last checkup across different general health statuses, it becomes apparent that, for all categories of general health, the majority of people have had their last checkup within the past year. However, there is still a significant portion of the population who have not had a checkup in the last 5 years or more. This increases the likelihood of potential cardiovascular disease.

Excersise and General Health

In [18]:
```
sns.countplot(x = 'General_Health', data = df).set_title('General Health and Exercise')
```

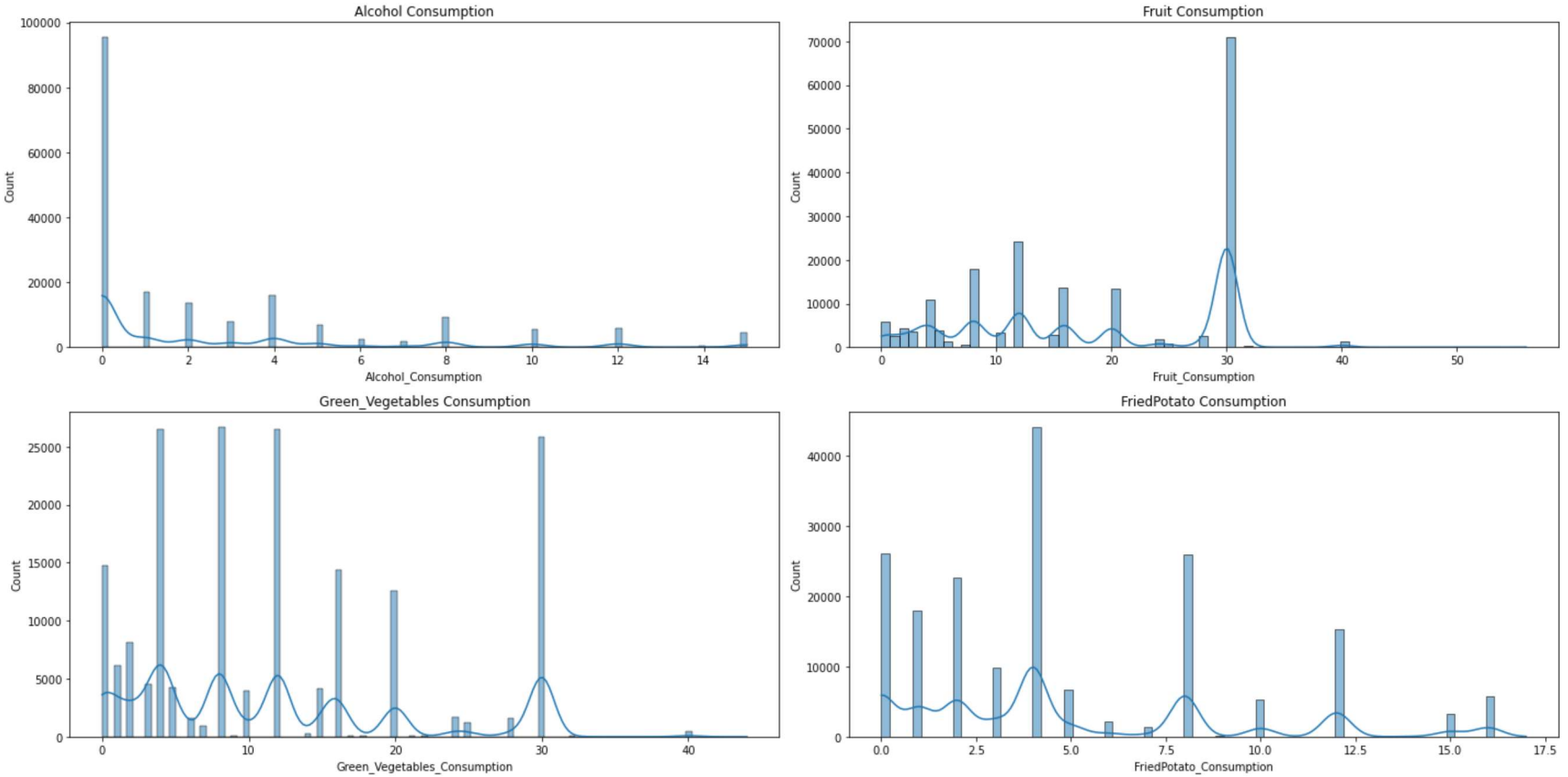
Out[18]:
```
Text(0.5, 1.0, 'General Health and Exercise')
```



The role of exercise in general health is evident from this graph. Individuals who exercise regularly are more likely to report good, very good, or excellent health. Conversely, those who do not exercise are more likely to report poor health. This underscores the importance of exercise in maintaining good health.

Food Consumption

In [19]:
```
fig, ax = plt.subplots(2,2,figsize=(20, 10))
sns.histplot(x = 'Alcohol_Consumption', data = df, kde = True).set_title('Alcohol Consumption')
sns.histplot(x = 'Fruit_Consumption', data = df, kde = True).set_title('Fruit Consumption')
sns.histplot(x = 'Green_Vegetables_Consumption', data = df, kde = True).set_title('Green_Vegetables Consumption')
sns.histplot(x = 'FriedPotato_Consumption', data = df, kde = True).set_title('FriedPotato Consumption')
plt.tight_layout()
```

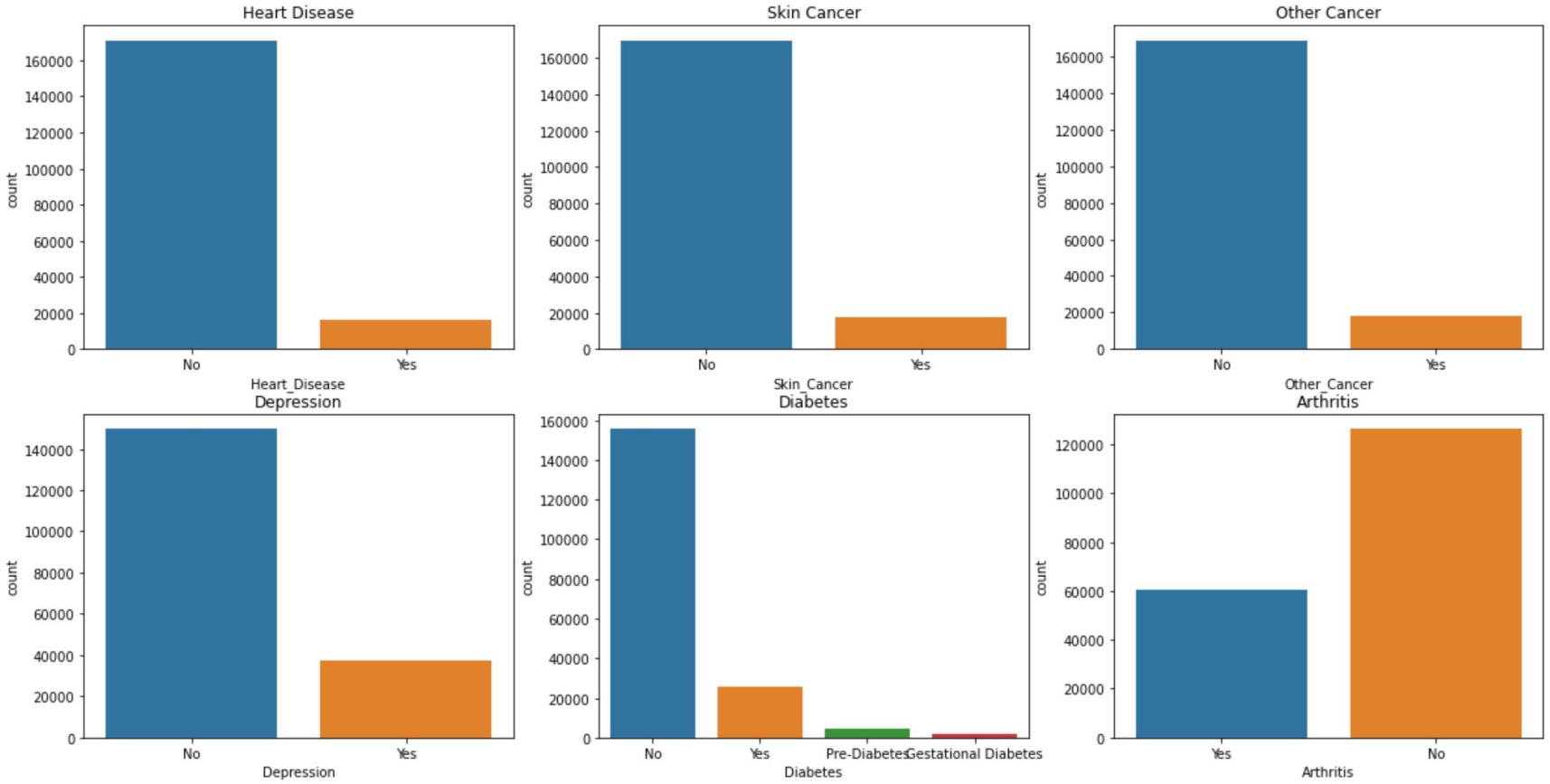


These plots visualize the food and drinking habits of the patients. It's evident that the majority of patients do not consume alcohol. Regarding food habits, most patients consume a higher amount of fruits and green vegetables, which is beneficial for health. However, a significant number of patients also consume fried potatoes, which is less healthy. This suggests that patients who consume fried potatoes and alcohol may be more likely to have cardiovascular disease.

Medical History

In [20]:
```
fig, ax = plt.subplots(2,3,figsize=(20, 10))
sns.countplot(x = 'Heart_Disease', data = df).set_title('Heart Disease')
sns.countplot(x = 'Skin_Cancer', data = df).set_title('Skin Cancer')
sns.countplot(x = 'Other_Cancer', data = df).set_title('Other Cancer')
sns.countplot(x = 'Depression', data = df).set_title('Depression')
sns.countplot(x = 'Diabetes', data = df).set_title('Diabetes')
sns.countplot(x = 'Arthritis', data = df).set_title('Arthritis')
```

Out[20]: Text(0.5, 1.0, 'Arthritis')

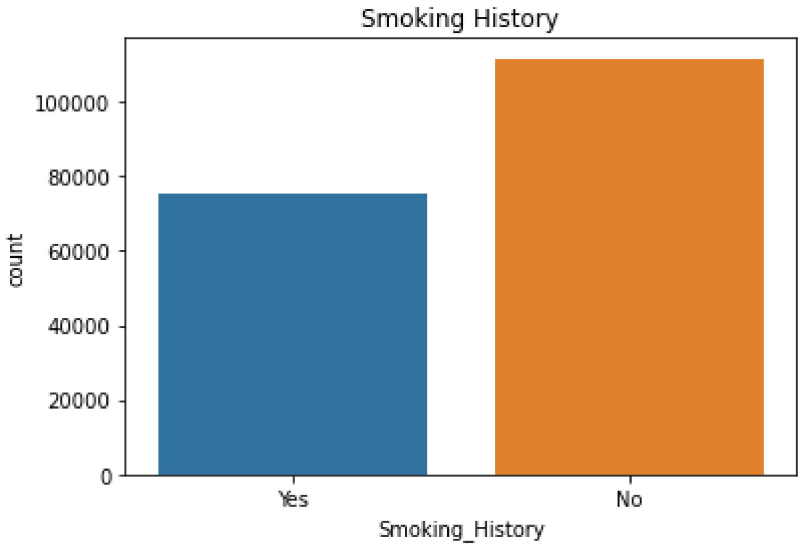


Most patients have no medical conditions, but some do have conditions such as heart disease, skin cancer, other cancers, depression, diabetes, and arthritis. Notably, there is an increased number of patients suffering from depression compared to other medical conditions. This suggests the importance of focusing on mental health in addition to physical health. Additionally, there are patients who are pre-diabetic, and some females suffer from gestational diabetes

Patient's Smoking History

In [21]:
```
sns.countplot(x = 'Smoking_History', data = df ).set_title('Smoking History')
```

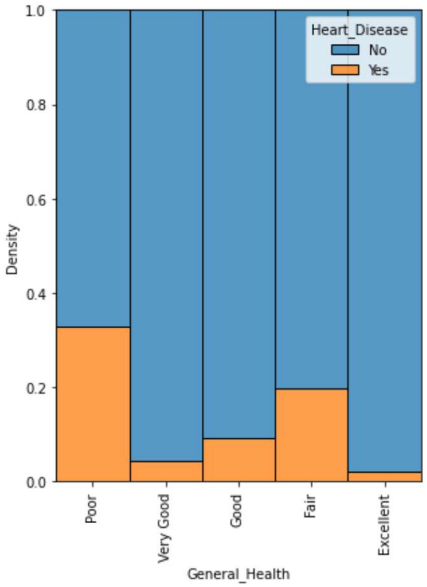
Out[21]: Text(0.5, 1.0, 'Smoking History')

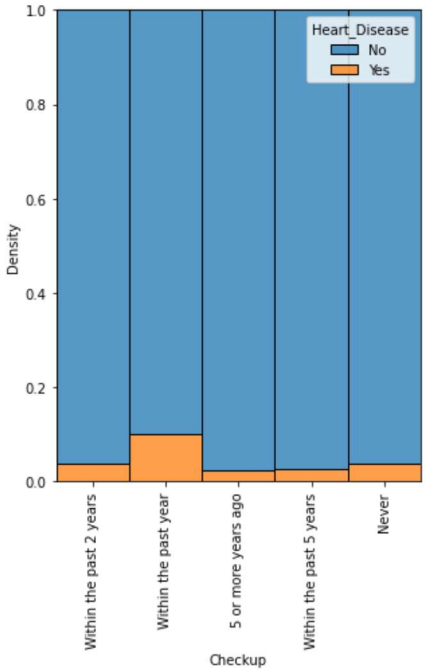


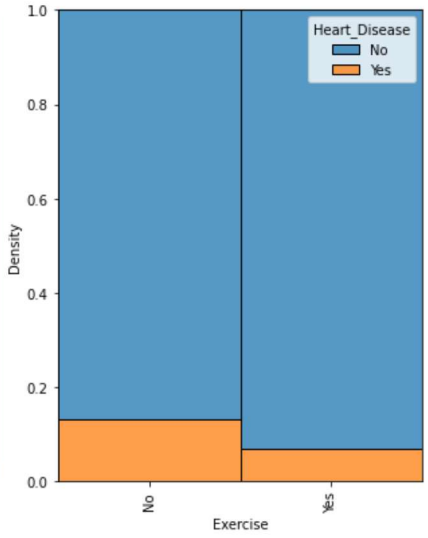
In [42]:
```
#This graph displays the smoking history of the patients in the dataset.
#The majority of patients have never smoked, but there is a significant number of current smokers.
#This suggests that patients who are current smokers may be more likely to have cardiovascular disease.
```

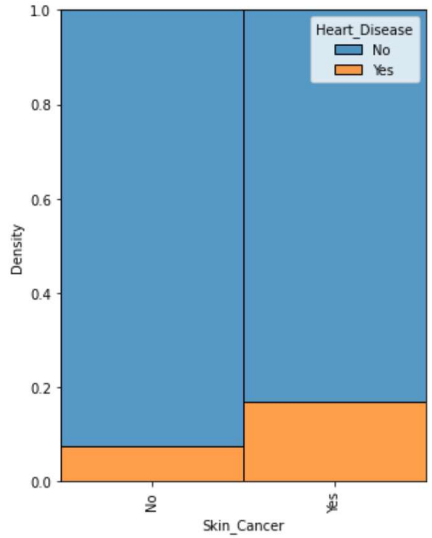
In [23]:
```
import warnings
warnings.filterwarnings("ignore")
# get list of categorical variables
cat_vars = ['General_Health', 'Checkup', 'Exercise', 'Skin_Cancer', 'Other_Cancer',
            'Depression', 'Diabetes', 'Arthritis', 'Sex', 'Age_Category', 'Smoking_History']

# create figure with subplots
fig, axs = plt.subplots(nrows=3,figsize=(18, 20))
axs = axs.flatten()

# create histplot for each categorical variable
for i, var in enumerate(cat_vars):
    sns.histplot(x=var, hue='Heart_Disease', data=df, ax=axs[i], element="bars", fill=True, stat='density')
    axs[i].set_xticklabels(df[var].unique(), rotation=90)
    axs[i].set_xlabel(var)

# adjust spacing between subplots
fig.tight_layout()

# remove the 12th subplot
fig.delaxes(axs[11])
```

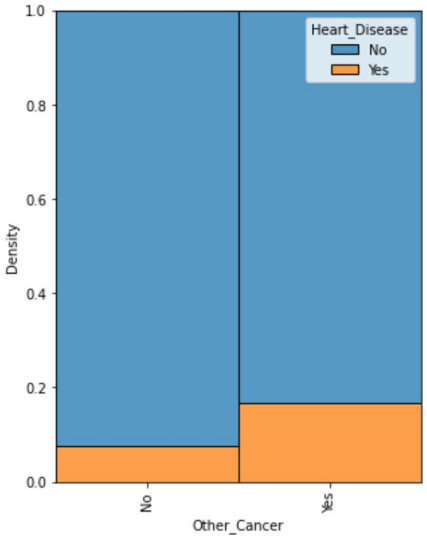
```
# show plot
plt.show()
```

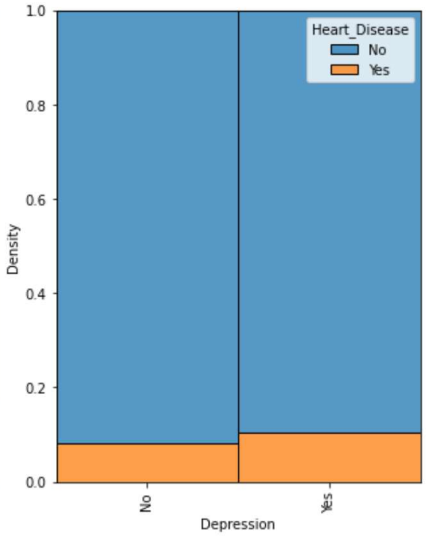

Heart_Disease
No
Yes
Density
1.0
0.8
0.6
0.4
0.2
0.0
Poor
Very Good
Good
Fair
Excellent
General_Health

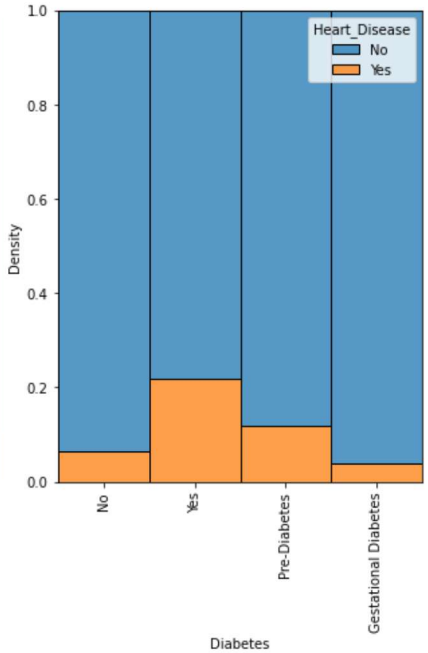

Heart_Disease
No
Yes
Density
1.0
0.8
0.6
0.4
0.2
0.0
Within the past 2 years
Within the past year
5 or more years ago
Within the past 5 years
Never
Checkup

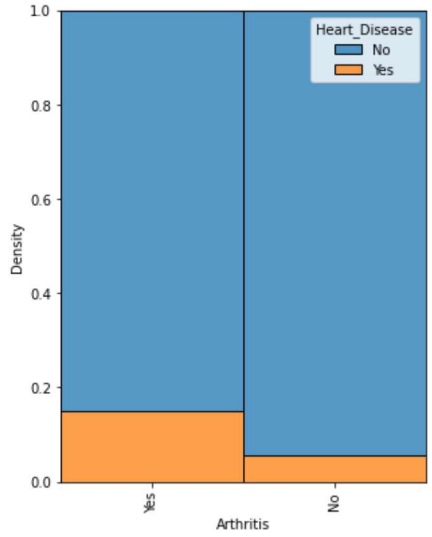

Heart_Disease
No
Yes
Density
1.0
0.8
0.6
0.4
0.2
0.0
No
Yes
Exercise

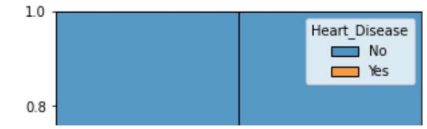

Heart_Disease
No
Yes
Density
1.0
0.8
0.6
0.4
0.2
0.0
No
Yes
Skin_Cancer

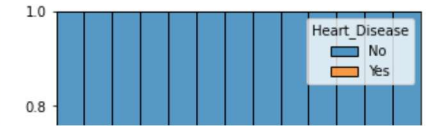

Heart_Disease
No
Yes
Density
1.0
0.8
0.6
0.4
0.2
0.0
No
Yes
Other_Cancer

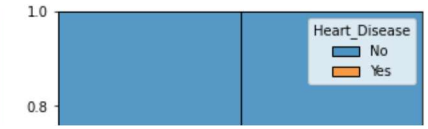

Heart_Disease
No
Yes
Density
1.0
0.8
0.6
0.4
0.2
0.0
No
Yes
Depression

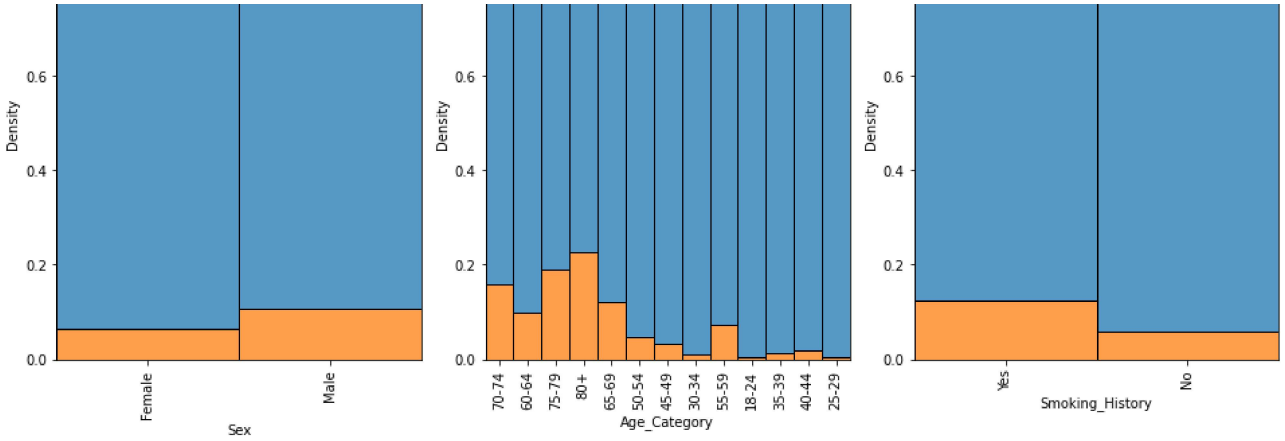

Heart_Disease
No
Yes
Density
1.0
0.8
0.6
0.4
0.2
0.0
No
Yes
Pre-Diabetes
Gestational Diabetes
Diabetes


Heart_Disease
No
Yes
Density
1.0
0.8
0.6
0.4
0.2
0.0
Yes
No
Arthritis


Heart_Disease
No
Yes
1.0
0.8


Heart_Disease
No
Yes
1.0
0.8

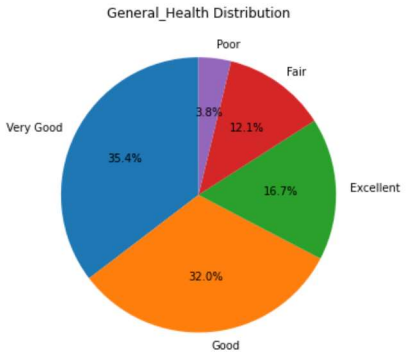

Heart_Disease
No
Yes
1.0
0.8

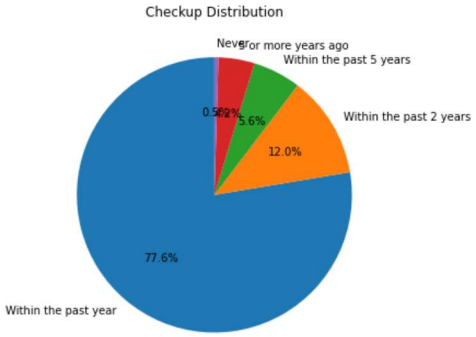


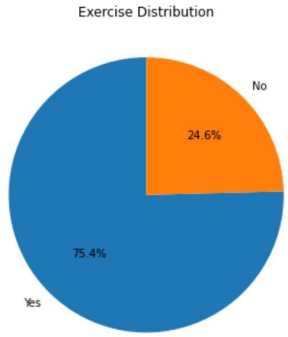
In [24]:

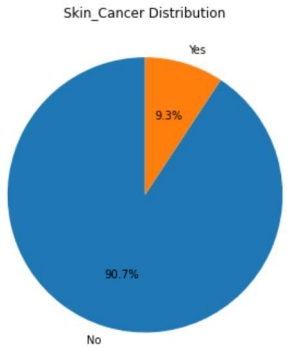
```
# Specify the maximum number of categories to show individually
max_categories = 5

cat_vars = ['General_Health', 'Checkup', 'Exercise', 'Skin_Cancer', 'Other_Cancer',
            'Depression', 'Diabetes', 'Arthritis', 'Sex', 'Age_Category', 'Smoking_History', 'Heart_Disease']

# Create a figure and axes
fig, axs = plt.subplots(nrows=3, figsize=(22, 22))

# Create a pie chart for each categorical variable
for i, var in enumerate(cat_vars):
    if i < len(axs.flat):
        # Count the number of occurrences for each category
        cat_counts = df[var].value_counts()

        # Group categories beyond the top max_categories as 'Other'
        if len(cat_counts) > max_categories:
            cat_counts_other = pd.Series(cat_counts[max_categories:].sum(), index=['Other'])
            cat_counts = cat_counts_top.append(cat_counts_other)

        # Create a pie chart
        axs.flat[i].pie(cat_counts, labels=cat_counts, startangle=90)

        # Set a title for each subplot
        axs.flat[i].set_title(f'{var} Distribution')

# Adjust spacing between subplots
fig.tight_layout()
```

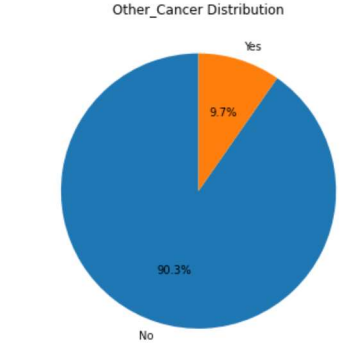
```
# Show the plot
plt.show()
```

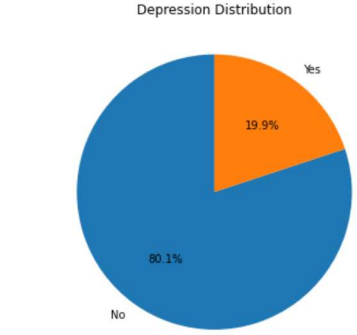

General_Health Distribution
Poor
Fair
Very Good
Excellent
3.8%
12.1%
35.4%
16.7%
32.0%
Good

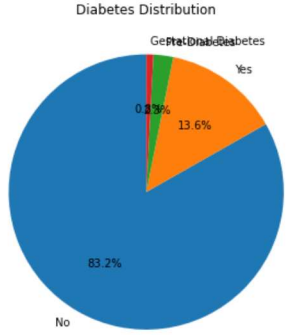

Checkup Distribution
Never
For more years ago
Within the past 5 years
Within the past 2 years
5.6%
12.0%
77.6%
Within the past year

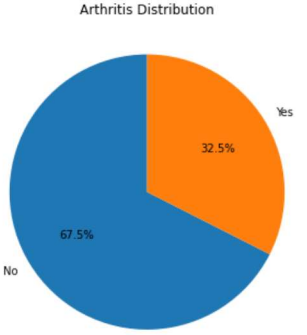

Exercise Distribution
No
24.6%
75.4%
Yes

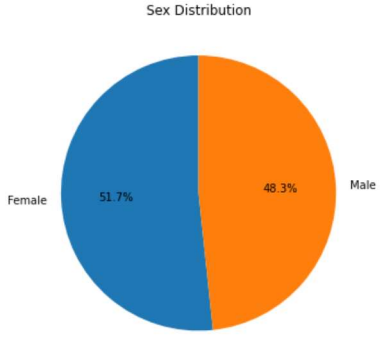

Skin_Cancer Distribution
Yes
9.3%
90.7%
No

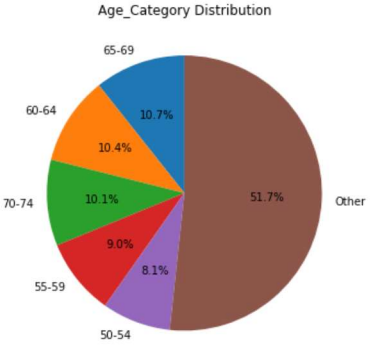

Other_Cancer Distribution
Yes
9.7%
90.3%
No

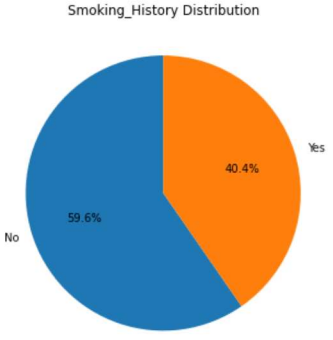

Depression Distribution
Yes
19.9%
80.1%
No

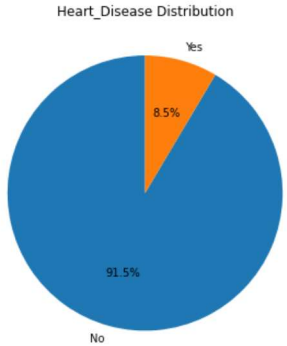

Diabetes Distribution
Yes
13.6%
83.2%
No


Arthritis Distribution
Yes
32.5%
67.5%
No

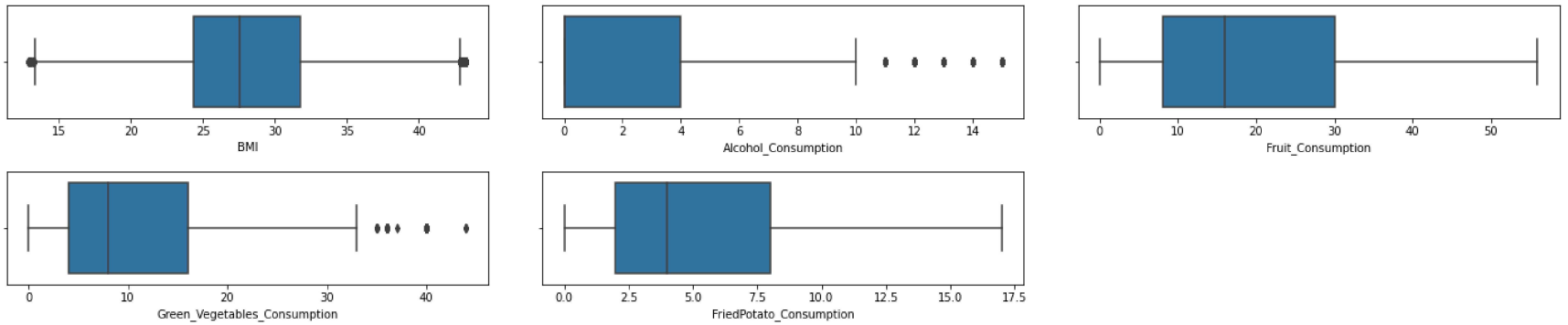

Sex Distribution
Female
51.7%
48.3%
Male


Age_Category Distribution
65-69
60-64
10.7%
10.4%
70-74
10.1%
51.7%
Other
9.0%
55-59
8.1%
50-54


Smoking_History Distribution
Yes
40.4%
No
59.6%

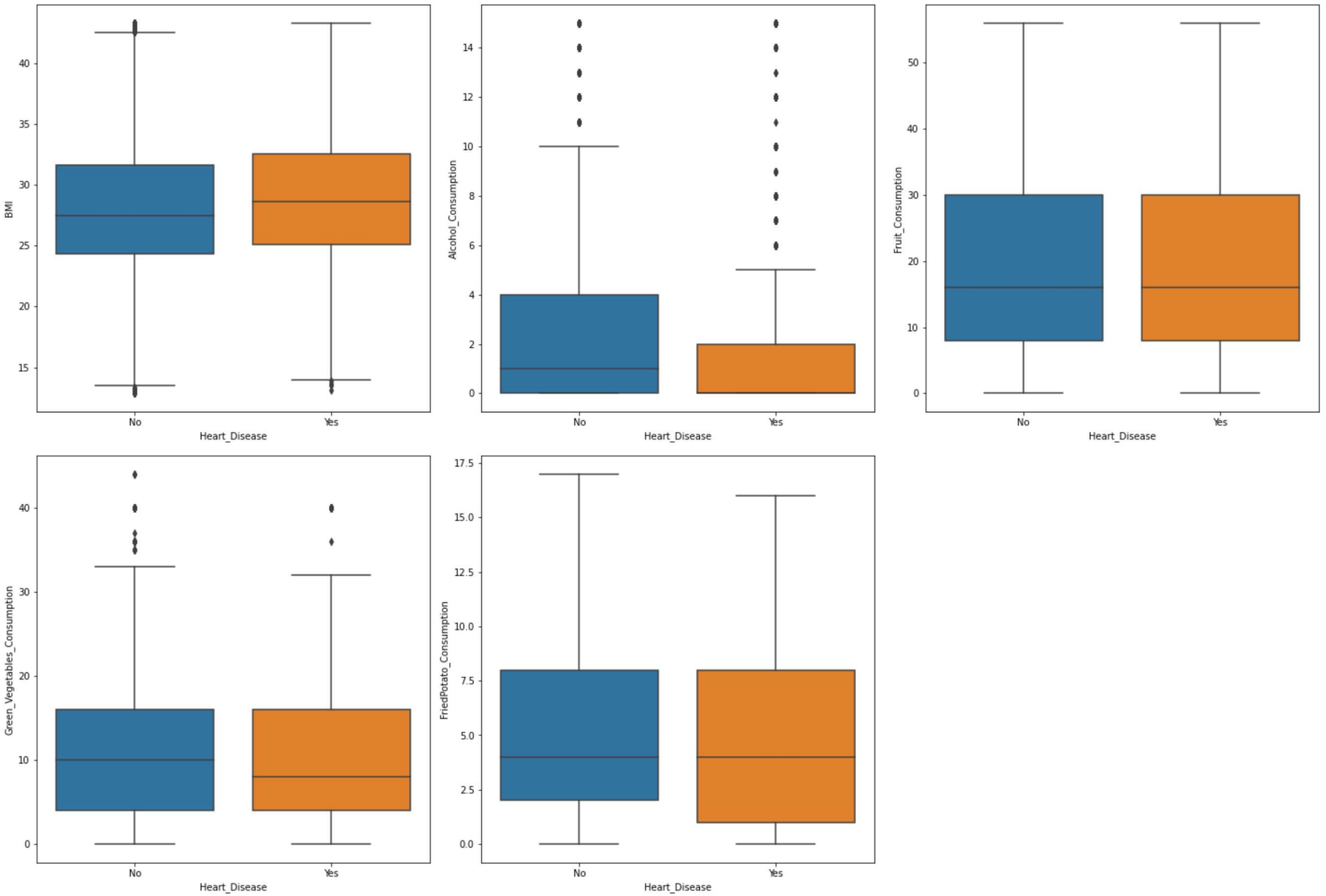

Heart_Disease Distribution
Yes
8.5%
91.5%
No

In [37]:
```
num_vars = [ 'BMI', 'Alcohol_Consumption',
            'Fruit_Consumption','Green_Vegetables_Consumption', 'FriedPotato_Consumption']

fig, axs = plt.subplots(nrows=3,figsize=(20, 6))
axs = axs.flatten()

for i, var in enumerate(num_vars):
    sns.boxplot(x=var, data=df, ax=axs[i])

fig.tight_layout()

# remove the 6th subplot
fig.delaxes(axs[5])

# remove the 9th subplot
fig.delaxes(axs[8])

plt.show()
```



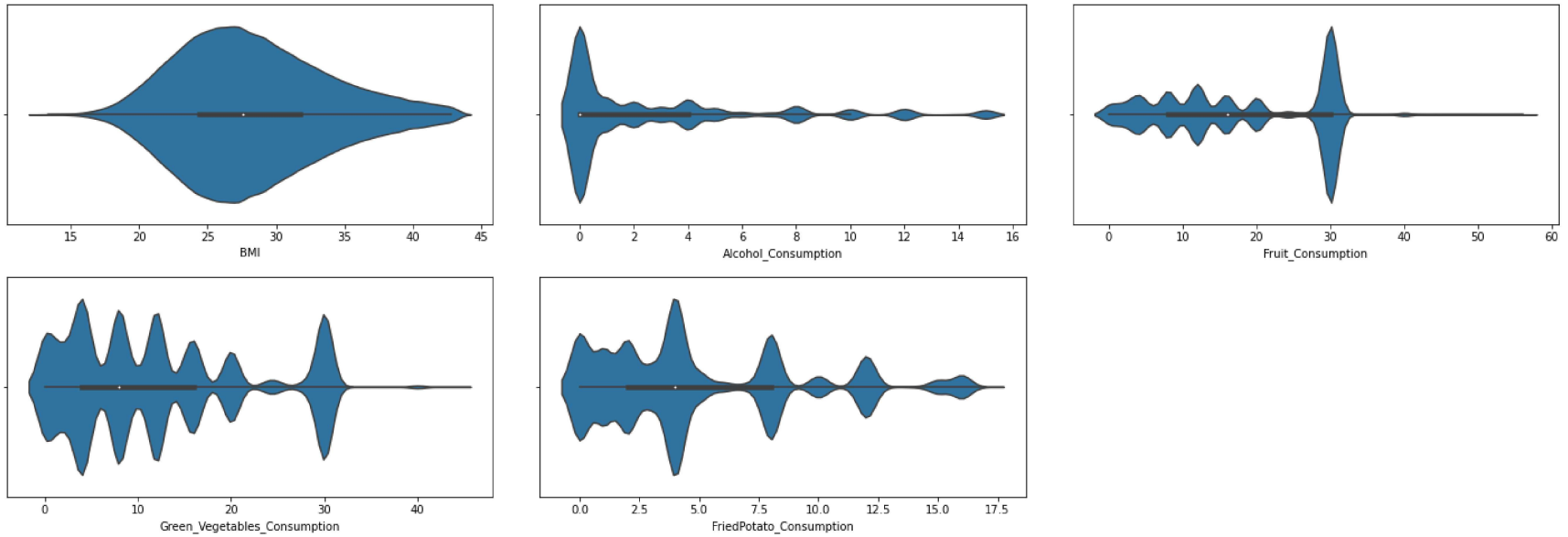
In [38]:
```
num_vars = ['BMI', 'Alcohol_Consumption',
            'Fruit_Consumption','Green_Vegetables_Consumption', 'FriedPotato_Consumption']

fig, axs = plt.subplots(nrows=3, ncols=3, figsize=(20, 20))
axs = axs.flatten()

for i, var in enumerate(num_vars):
    sns.boxplot(y=var, x='Heart_Disease', data=df, ax=axs[i])

fig.tight_layout()

# remove the 6th subplot
fig.delaxes(axs[5])
```

```
# remove the 9th subplot
fig.delaxes(axs[8])
plt.show()
```



In [39]:
```
num_vars = ['BMI', 'Alcohol_Consumption',
            'Fruit_Consumption','Green_Vegetables_Consumption', 'FriedPotato_Consumption']
```

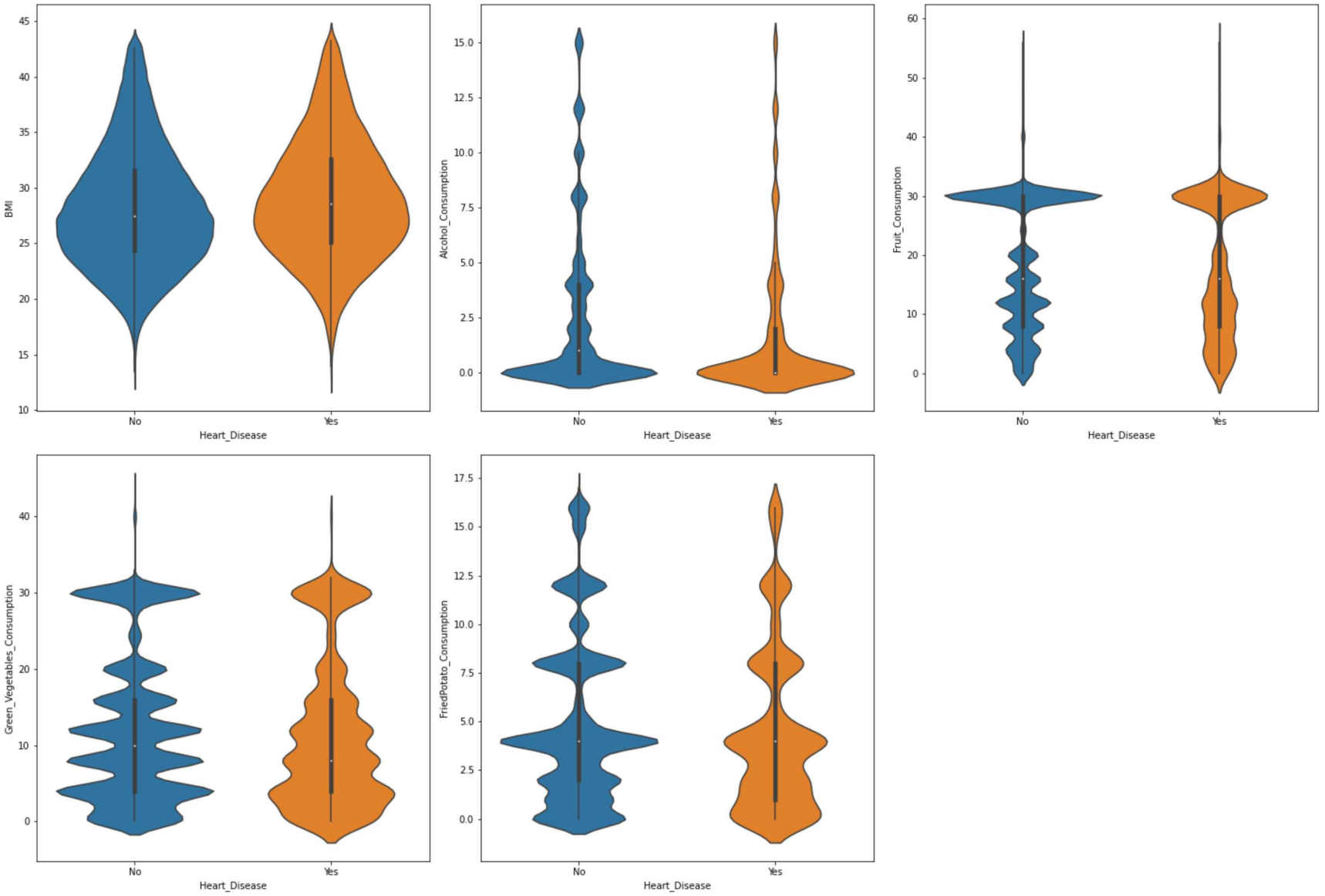
```
fig, axs = plt.subplots(nrows=3, ncols=3, figsize=(20, 10))
axs = axs.flatten()

for i, var in enumerate(num_vars):
    sns.violinplot(x=var, data=df, ax=axs[i])

fig.tight_layout()

# remove the 8th subplot
fig.delaxes(axs[7])

# remove the 9th subplot
fig.delaxes(axs[8])
plt.show()
```



In [40]:

```
num_vars = ['BMI', 'Alcohol_Consumption',
            'Fruit_Consumption','Green_Vegetables_Consumption', 'FriedPotato_Consumption']

fig, axs = plt.subplots(nrows=3, ncols=3, figsize=(20, 20))
axs = axs.flatten()

for i, var in enumerate(num_vars):
    sns.violinplot(y=var, data=df, x='Heart_Disease', ax=axs[i])

fig.tight_layout()
```

```
# remove the 8th subplot
fig.delaxes(axs[7])

# remove the 9th subplot
fig.delaxes(axs[8])

plt.show()
```

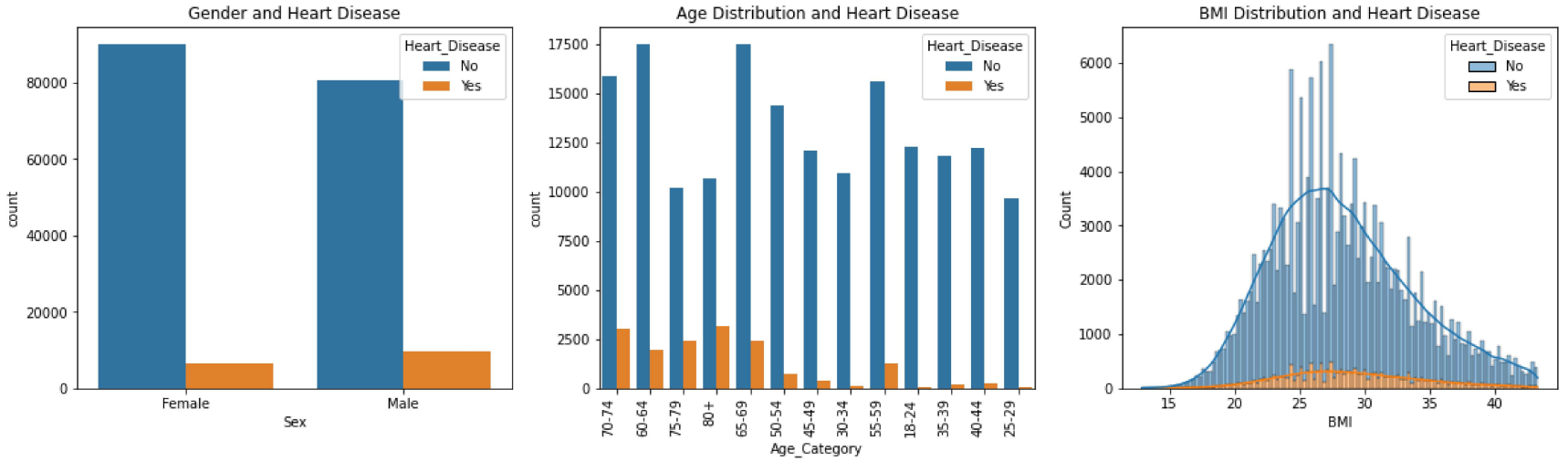


Target Variable and Independent Variables Visualization

Patient's Demographics and Heart Disease

In [43]:
```
fig, ax = plt.subplots(1,3,figsize=(20, 5))
sns.countplot(x = 'Sex', data = df, hue  = 'Heart_Disease', ax = ax[0]).set_title('Gender and Heart Disease')
sns.countplot(x = 'Age_Category', data = df, hue = 'Heart_Disease').set_title('Age Distribution and Heart Disease')
ax[1].set_xticklabels(ax[1].get_xticklabels())
sns.histplot(x = 'BMI', data = df, ax = ax[2], kde = True, hue = 'Heart_Disease', multiple = 'stack').set_title('BMI Di
```

Out[43]:
```
Text(0.5, 1.0, 'BMI Distribution and Heart Disease')
```



Visualizing patient demographics alongside heart disease data provides insights into the relationship between cardiovascular disease and patient characteristics.

First, the Gender graph indicates that males are more susceptible to heart disease compared to females.

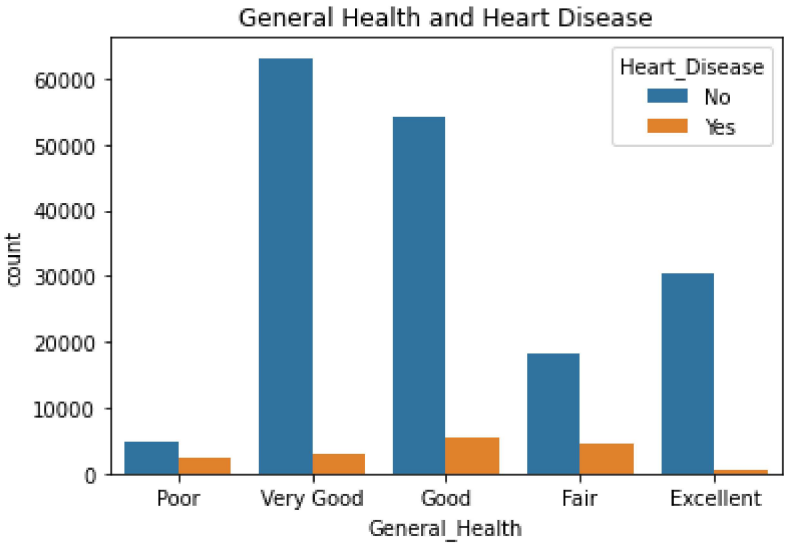
Second, the Age graph reveals that patients over 55 years of age, particularly those in the 80+ age group, have a higher incidence of heart disease, indicating that cardiovascular disease risk increases with age.

Third, the BMI graph shows that patients with a BMI between 25-30 (overweight) have a higher likelihood of heart disease.

General Health and Heart Disease

In [44]:
```
sns.countplot(x = 'General_Health', data = df, hue  = 'Heart_Disease').title('General Health and Heart Disease')
```

Out[44]:
```
Text(0.5, 1.0, 'General Health and Heart Disease')
```

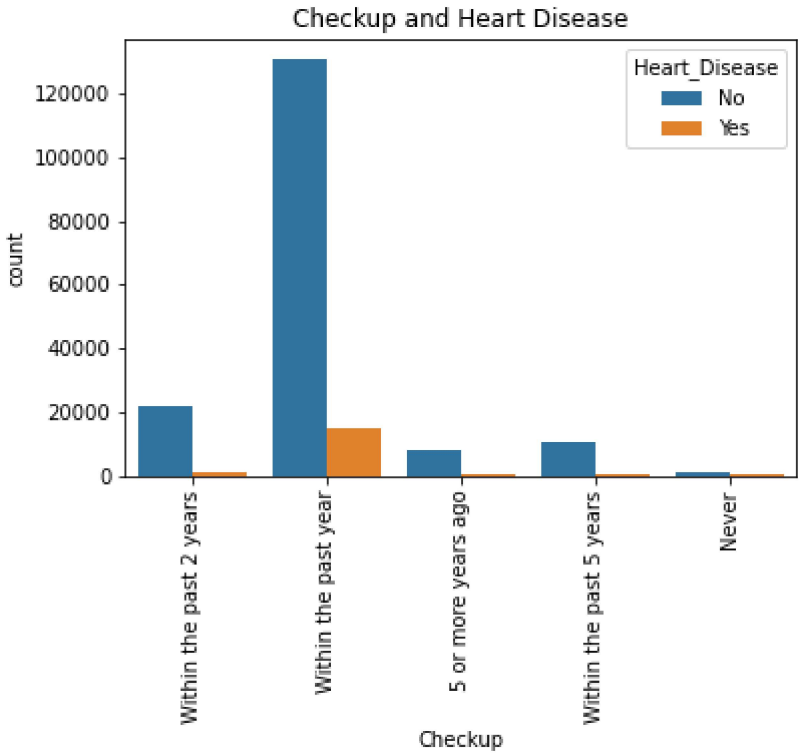


This graph contradicts the belief that healthy patients are less prone to heart disease. Instead, it reveals that patients with very good or good general health have a higher likelihood of heart disease compared to patients with poor general health.

Checkup and Heart Disease

In [45]:
```
sns.countplot(x = 'Checkup', data = df, hue  = 'Heart_Disease').title('Checkup and Heart Disease')
plt.xticks(rotation=90)
```

Out[45]:
```
(array([0, 1, 2, 3, 4]),
 [Text(0, 0, 'Within the past 2 years'),
  Text(1, 0, 'Within the past year'),
  Text(2, 0, '5 or more years ago'),
  Text(3, 0, 'Within the past 5 years'),
  Text(4, 0, 'Never')])
```

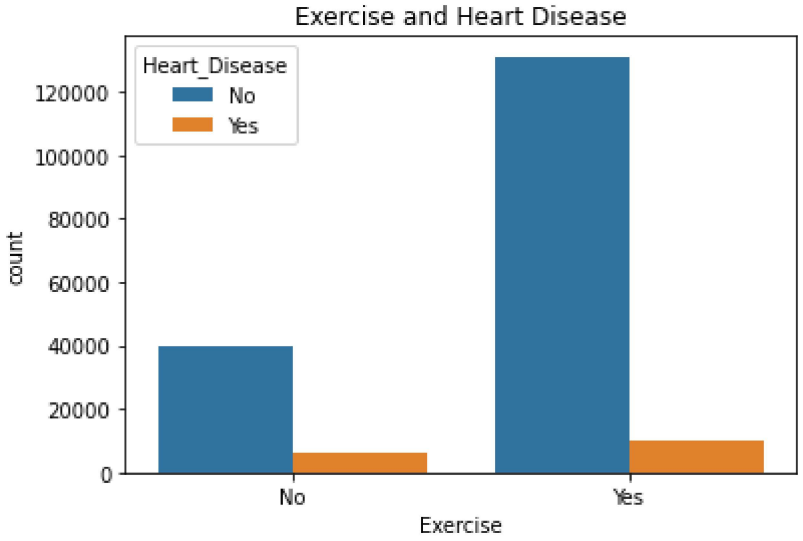


According to this graph, patients who had a checkup in the last year have a higher likelihood of having heart disease. This suggests that patients who undergo regular checkups are more likely to detect cardiovascular disease at an early stage compared to those who do not.

Excercise and Heart Disease

In [46]:
```
sns.countplot(x = 'Exercise', data = df, hue  = 'Heart_Disease').title('Exercise and Heart Disease')
```

Out[46]:
```
Text(0.5, 1.0, 'Exercise and Heart Disease')
```

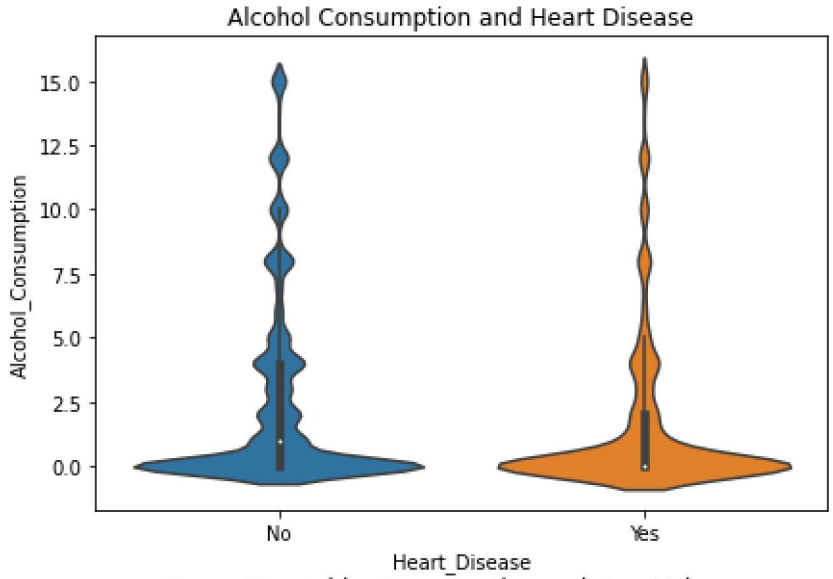


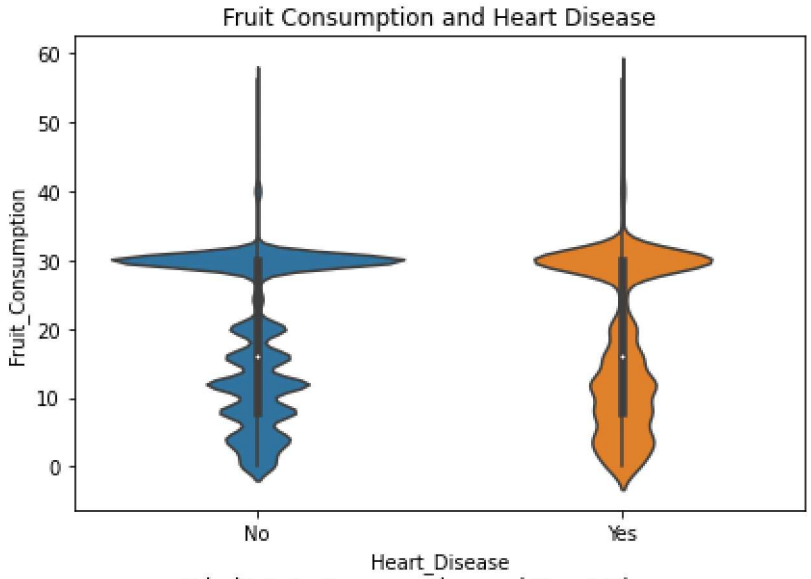
Interestingly, patients who exercise tend to have higher rates of heart disease, contrary to the belief that regular exercise reduces the risk of heart disease. Conversely, this graph indicates that patients who do not exercise are less prone to heart disease. It's possible that patients with weakened hearts exert excessive pressure on their hearts through exercise, potentially contributing to heart disease.

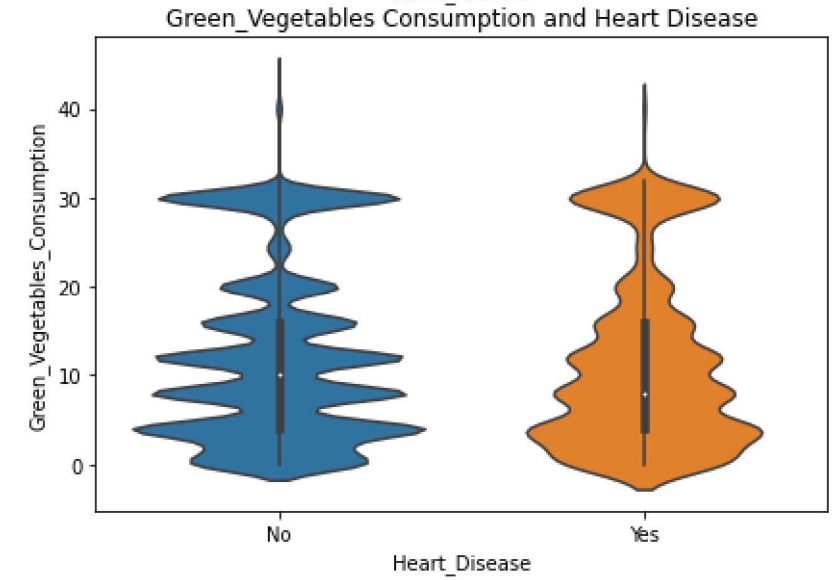
Food Consumption and Heart Disease

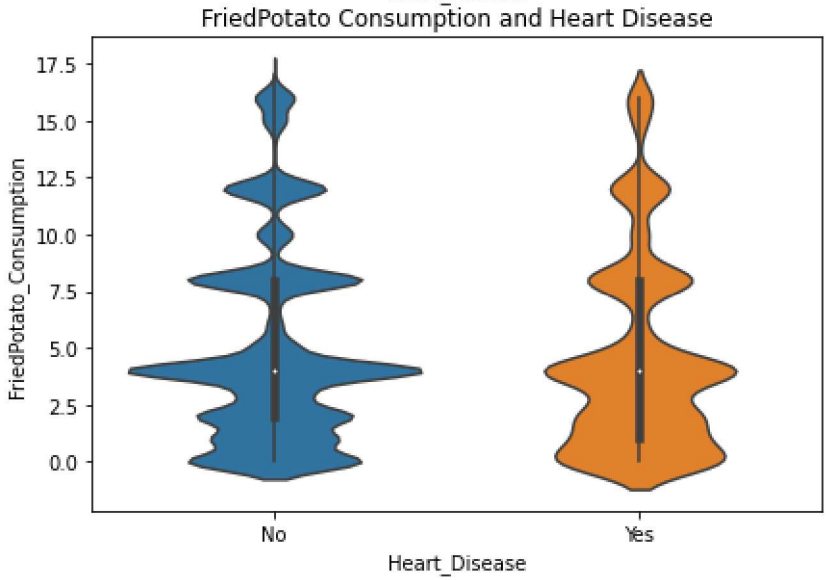
In [47]:
```
fig, ax = plt.subplots(2,2,figsize=(15, 10))
sns.violinplot(x = 'Heart_Disease', y = 'Alcohol_Consumption', data = df).set_title('Alcohol Consumption and Heart Dise
sns.violinplot(x = 'Heart_Disease', y = 'Fruit_Consumption', data = df).set_title('Fruit Consumption and Heart Disease'
sns.violinplot(x = 'Heart_Disease', y = 'Green_Vegetables_Consumption', data = df).set_title('Green_Vegetables Consumpt
sns.violinplot(x = 'Heart_Disease', y = 'FriedPotato_Consumption', data = df).set_title('FriedPotato Consumption and He
```

Out[47]:
```
Text(0.5, 1.0, 'FriedPotato Consumption and Heart Disease')
```









These graphs visualize the relationship between patient food and drink habits and heart disease.

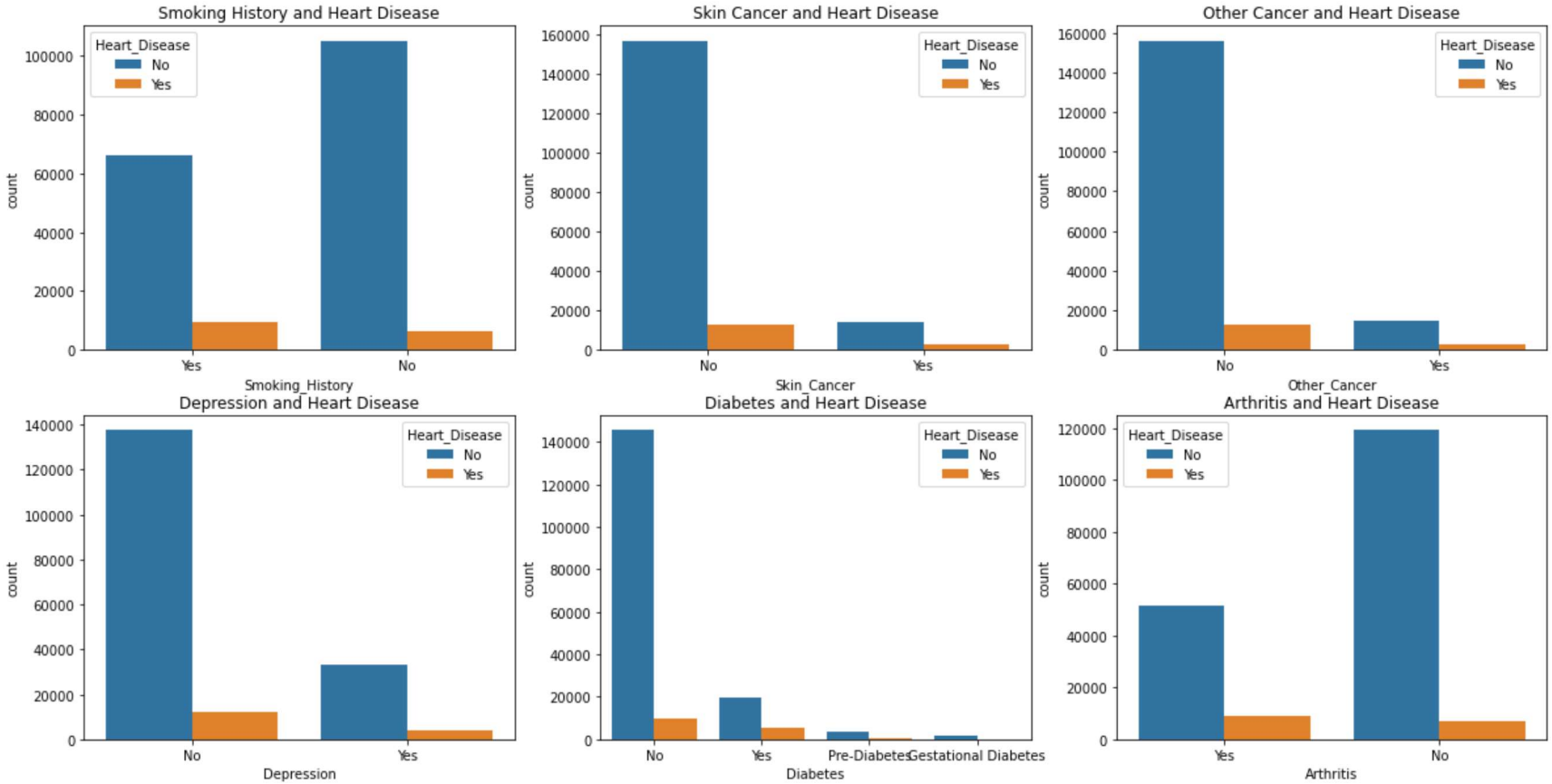
In the alcohol consumption graph, patients with higher alcohol consumption have lower chances of heart disease.

Regarding food habits, patients with increased consumption of fruits and green vegetables tend to have a lower risk of heart disease. Conversely, those with higher consumption of fried potatoes tend to have a higher risk of heart disease.

Medical History and Heart Disease

In [48]:
```
fig, ax = plt.subplots(2,3,figsize=(20, 10))
sns.countplot(x = 'Smoking_History', data = df,  hue = 'Heart_Disease').set_title('Smoking History and Heart Disease')
sns.countplot(x = 'Skin_Cancer', data = df,  hue = 'Heart_Disease').set_title('Skin Cancer and Heart Disease')
sns.countplot(x = 'Other_Cancer', data = df,  hue = 'Heart_Disease').set_title('Other Cancer and Heart Disease')
sns.countplot(x = 'Depression', data = df,  hue = 'Heart_Disease').set_title('Depression and Heart Disease')
sns.countplot(x = 'Diabetes', data = df,  hue = 'Heart_Disease').set_title('Diabetes and Heart Disease')
sns.countplot(x = 'Arthritis', data = df,  hue = 'Heart_Disease').set_title('Arthritis and Heart Disease')
```

Out[48]:
```
Text(0.5, 1.0, 'Arthritis and Heart Disease')
```



These graphs illustrate the relationship between patient medical history and heart disease.

1,In the first graph on smoking history, patients who smoke or used to smoke tend to have a higher incidence of cardiovascular disease.

2,The second graph shows that patients without skin cancer have a higher likelihood of having heart disease compared to those with skin cancer.

3,The third graph indicates that patients without any type of cancer are more likely to have cardiovascular disease.

4,In the fourth graph, patients without depression have a higher incidence of heart disease compared to those with depression.

5,The fifth graph reveals that patients without diabetes are more likely to have heart disease, while pre-diabetes or gestational diabetes seem to have little or no effect on heart disease.

6,The final graph demonstrates that patients without arthritis are more likely to have heart disease compared to those with arthritis.

Based on the information presented above, it appears that patients' medical histories do not have a major effect on the likelihood of having cardiovascular disease. However, it's important to note that this conclusion is based on the data available and may not account for all possible factors or interactions.

Data Preprocessing 2

Label Encoding the Categorical Variables

In [49]:
```
#Check missing value
check_missing = df.isnull()* 100 / df.shape[0]
check_missing[check_missing > 0]values(ascending=False)
```

Out[49]:
```
Series([], dtype: float64)
```

In [50]:
```
# Loop over each column in the DataFrame where dtype is 'object'
for col in df.select_dtypes(include=['object'])columns:

    # Print the column name and the unique values
    print(f"{col}: {df[col]unique()}")
```

```
General_Health: ['Poor' 'Very Good' 'Good' 'Fair' 'Excellent']
Checkup: ['Within the past 2 years' 'Within the past year' '5 or more years ago'
 'Within the past 5 years' 'Never']
Exercise: ['No' 'Yes']
Heart_Disease: ['No' 'Yes']
Skin_Cancer: ['No' 'Yes']
Other_Cancer: ['No' 'Yes']
Depression: ['No' 'Yes']
Diabetes: ['No' 'Yes' 'Pre-Diabetes' 'Gestational Diabetes']
Arthritis: ['Yes' 'No']
Sex: ['Female' 'Male']
Age_Category: ['70-74' '60-64' '75-79' '80+' '65-69' '50-54' '45-49' '30-34' '55-59'
 '18-24' '35-39' '40-44' '25-29']
Smoking_History: ['Yes' 'No']
```

In [51]:

```python
from sklearn import preprocessing

# Loop over each column in the DataFrame where dtype is 'object'
for col in df.select_dtypes(include=['object']).columns:

    # Initialize a LabelEncoder object
    label_encoder = preprocessing.LabelEncoder()

    # Fit the encoder to the unique values in the column
    label_encoder.fit(df[col].unique())

    # Transform the column using the encoder
    df[col] = label_encodert ransform(df[col])

    # Print the column name and the unique encoded values
    print(f"{col}: {df[col]unique()}")
```

```
General_Health: [3 4 2 1 0]
Checkup: [2 4 0 3 1]
Exercise: [0 1]
Heart_Disease: [0 1]
Skin_Cancer: [0 1]
Other_Cancer: [0 1]
Depression: [0 1]
Diabetes: [1 3 2 0]
Arthritis: [1 0]
Sex: [0 1]
Age_Category: [10  8 11 12  9  6  5  2  7  0  3  4  1]
Smoking_History: [1 0]
```

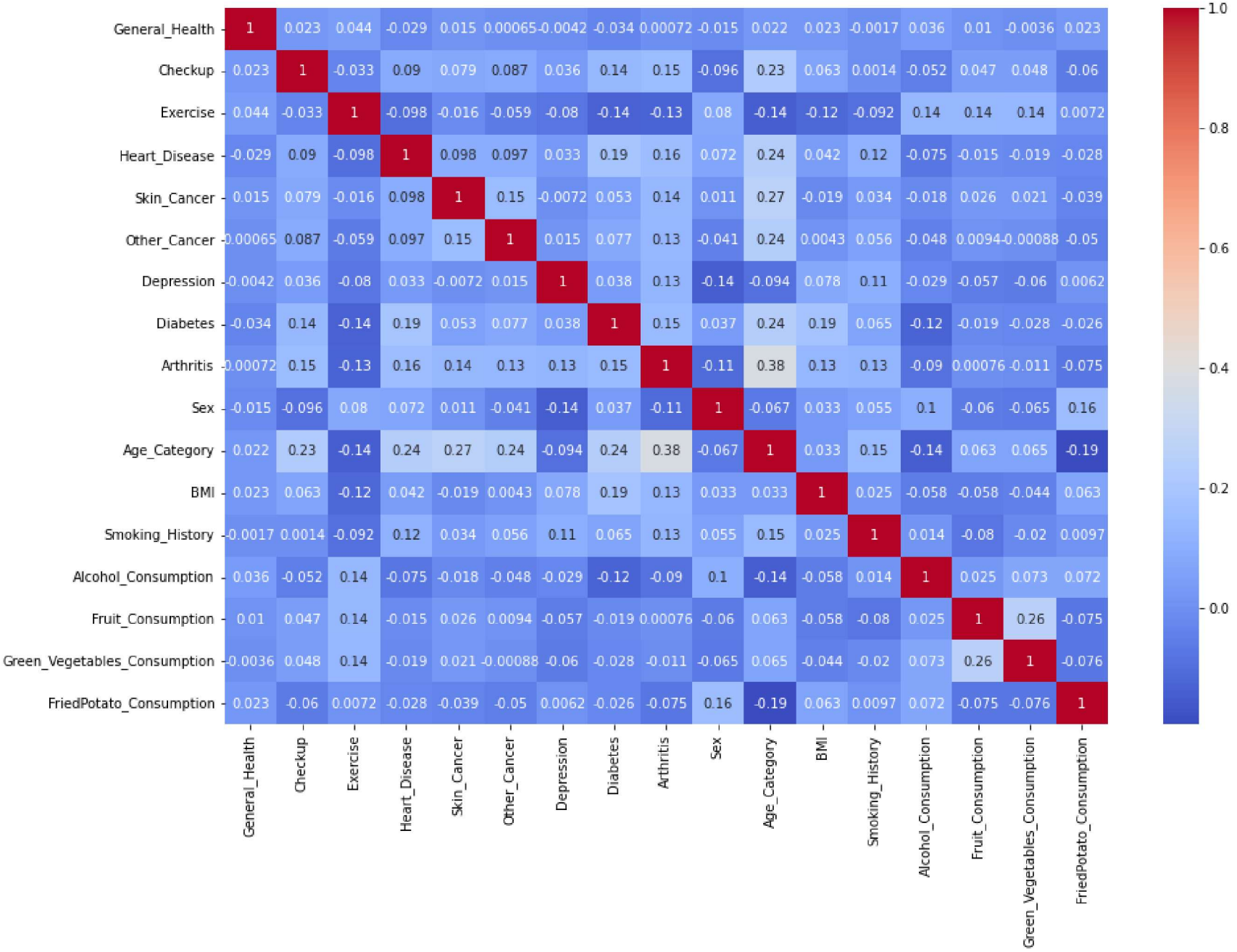
In [28]:
```
from sklearn.preprocessing import LabelEncoder

# List of categorical variables
cols = ['General_Health','Checkup','Exercise','Heart_Disease','Skin_Cancer','Other_Cancer','Depression','Diabetes','Art

# Label encoding object
le = Encoder()

for ii in cols:
    le.fit(df[ii])
    df[ii] = le.transform(df[ii])
    print(ii, df[ii].unique())
```

```
General_Health [3 4 2 1 0]
Checkup [2 4 0 3 1]
Exercise [0 1]
Heart_Disease [0 1]
Skin_Cancer [0 1]
Other_Cancer [0 1]
Depression [0 1]
Diabetes [1 3 2 0]
Arthritis [1 0]
Sex [0 1]
Age_Category [10  8 11 12  9  6  5  2  7  0  3  4  1]
Smoking_History [1 0]
```

Correlation Matrix Heatmap

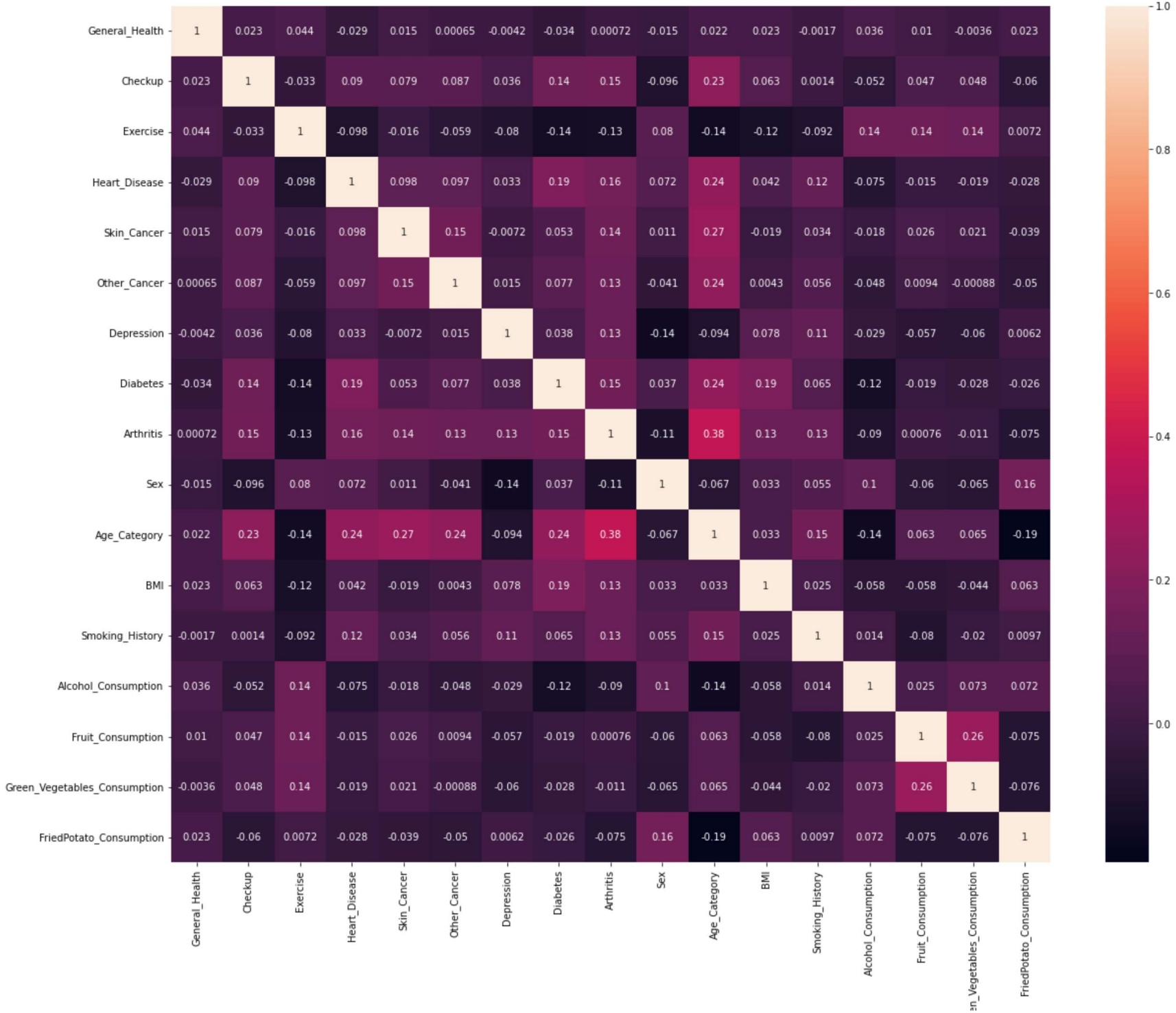
In [52]:
```
plt.figure(figsize=(15,10))
sns.heatmap(df.cor(), annot = True, map = 'coolwarm')
```

Out[52]:
```
<AxesSubplot:>
```



In [53]:
```
#Correlation Heatmap (print the correlation score each variables)
plt.figure(figsize=(20, 16))
sns.heatmap(df.cor(), fmt='2g', annot=True)
```

Out[53]: <AxesSubplot:>


General_Health 1 0.023 0.044 -0.029 0.015 0.00065 -0.0042 -0.034 0.00072 -0.015 0.022 0.023 -0.0017 0.036 0.01 -0.0036 0.023
Checkup 0.023 1 -0.033 0.09 0.079 0.087 0.036 0.14 0.15 -0.096 0.23 0.063 0.0014 -0.052 0.047 0.048 -0.06
Exercise 0.044 -0.033 1 -0.098 -0.016 -0.059 -0.08 -0.14 -0.13 0.08 -0.14 -0.12 -0.092 0.14 0.14 0.14 0.0072
Heart_Disease -0.029 0.09 -0.098 1 0.098 0.097 0.033 0.19 0.16 0.072 0.24 0.042 0.12 -0.075 -0.015 -0.019 -0.028
Skin_Cancer 0.015 0.079 -0.016 0.098 1 0.15 -0.0072 0.053 0.14 0.011 0.27 -0.019 0.034 -0.018 0.026 0.021 -0.039
Other_Cancer 0.00065 0.087 -0.059 0.097 0.15 1 0.015 0.077 0.13 -0.041 0.24 0.0043 0.056 -0.048 0.0094 -0.00088 -0.05
Depression -0.0042 0.036 -0.08 0.033 -0.0072 0.015 1 0.038 0.13 -0.14 -0.094 0.078 0.11 -0.029 -0.057 -0.06 0.0062
Diabetes -0.034 0.14 -0.14 0.19 0.053 0.077 0.038 1 0.15 0.037 0.24 0.19 0.065 -0.12 -0.019 -0.028 -0.026
Arthritis 0.00072 0.15 -0.13 0.16 0.14 0.13 0.13 0.15 1 -0.11 0.38 0.13 0.13 -0.09 0.00076 -0.011 -0.075
Sex -0.015 -0.096 0.08 0.072 0.011 -0.041 -0.14 0.037 -0.11 1 -0.067 0.033 0.055 0.1 -0.06 -0.065 0.16
Age_Category 0.022 0.23 -0.14 0.24 0.27 0.24 -0.094 0.24 0.38 -0.067 1 0.033 0.15 -0.14 0.063 0.065 -0.19
BMI 0.023 0.063 -0.12 0.042 -0.019 0.0043 0.078 0.19 0.13 0.033 0.033 1 0.025 -0.058 -0.058 -0.044 0.063
Smoking_History -0.0017 0.0014 -0.092 0.12 0.034 0.056 0.11 0.065 0.13 0.055 0.15 0.025 1 0.014 -0.08 -0.02 0.0097
Alcohol_Consumption 0.036 -0.052 0.14 -0.075 -0.018 -0.048 -0.029 -0.12 -0.09 0.1 -0.14 -0.058 0.014 1 0.025 0.073 0.072
Fruit_Consumption 0.01 0.047 0.14 -0.015 0.026 0.0094 -0.057 -0.019 0.00076 -0.06 0.063 -0.058 -0.08 0.025 1 0.26 -0.075
Green_Vegetables_Consumption -0.0036 0.048 0.14 -0.019 0.021 -0.00088 -0.06 -0.028 -0.011 -0.065 0.065 -0.044 -0.02 0.073 0.26 1 -0.076
FriedPotato_Consumption 0.023 -0.06 0.0072 -0.028 -0.039 -0.05 0.0062 -0.026 -0.075 0.16 -0.19 0.063 0.0097 0.072 -0.075 -0.076 1
General_Health
Checkup
Exercise
Heart_Disease
Skin_Cancer
Other_Cancer
Depression
Diabetes
Arthritis
Sex
Age_Category
BMI
Smoking_History
Alcohol_Consumption
Fruit_Consumption
Green_Vegetables_Consumption
FriedPotato_Consumption
1.0
0.8
0.6
0.4
0.2
0.0

There is no strong or significant correlation among the variables.

Train Test Split

In [54]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.drop(columns = ['Heart_Disease']), df['Heart_Disease'], test_siz
```

In [71]:
```
from sklearn.model_selection import train_test_split
# Select the features (X) and the target variable (y)
X = df.drop('Heart_Disease')
y = df['Heart_Disease']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [72]:
```
#Remove the Outlier from train data using IQR
```

In [73]:
```
# Define the columns for which you want to remove outliers
selected_columns = ['BMI', 'Alcohol_Consumption',
            'Fruit_Consumption','Green_Vegetables_Consumption', 'FriedPotato_Consumption']

# Calculate the IQR for the selected columns in the training data
Q1 = X_train[selected_columns].quantile(0.25)
Q3 = X_train[selected_columns].quantile(0.75)

# Set a threshold value for outlier detection (e.g., 1.5 times the IQR)
threshold = 1.5

# Find the indices of outliers based on the threshold
outlier_indices = np.where(
    (X_train[selected_columns] < (Q1 - threshold * IQR)) |
    (X_train[selected_columns] > (Q3 + threshold * IQR))
)[0]

# Remove the outliers from the training data and corresponding labels
X_train = X_train.drop(X_train.index[outlier_indices])
y_train = y_train.drop(y_train.index[outlier_indices])
```

Cardiovascular Disease Prediction

For predicting the cardiovascular disease, I have used the following classification models:

1. Random Forest Classifier
2. Decision Tree Classifier
3. Logistic Regression

Random Forest Classifier

In [74]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(random_state=0, max_features='sqrt', class_weight='balanced')
rfc.fit(X_train, y_train)
```

Out[74]:
```
RandomForestClassifier(class_weight='balanced', max_features='sqrt',
                       n_estimators=200, random_state=0)
```

In [76]:
```
from sklearn.metrics import accuracy_score

y_pred = rfc.predict(X_test)
print("Accuracy Score:", round(accuracy_score(y_test, y_pred) * 100)
```

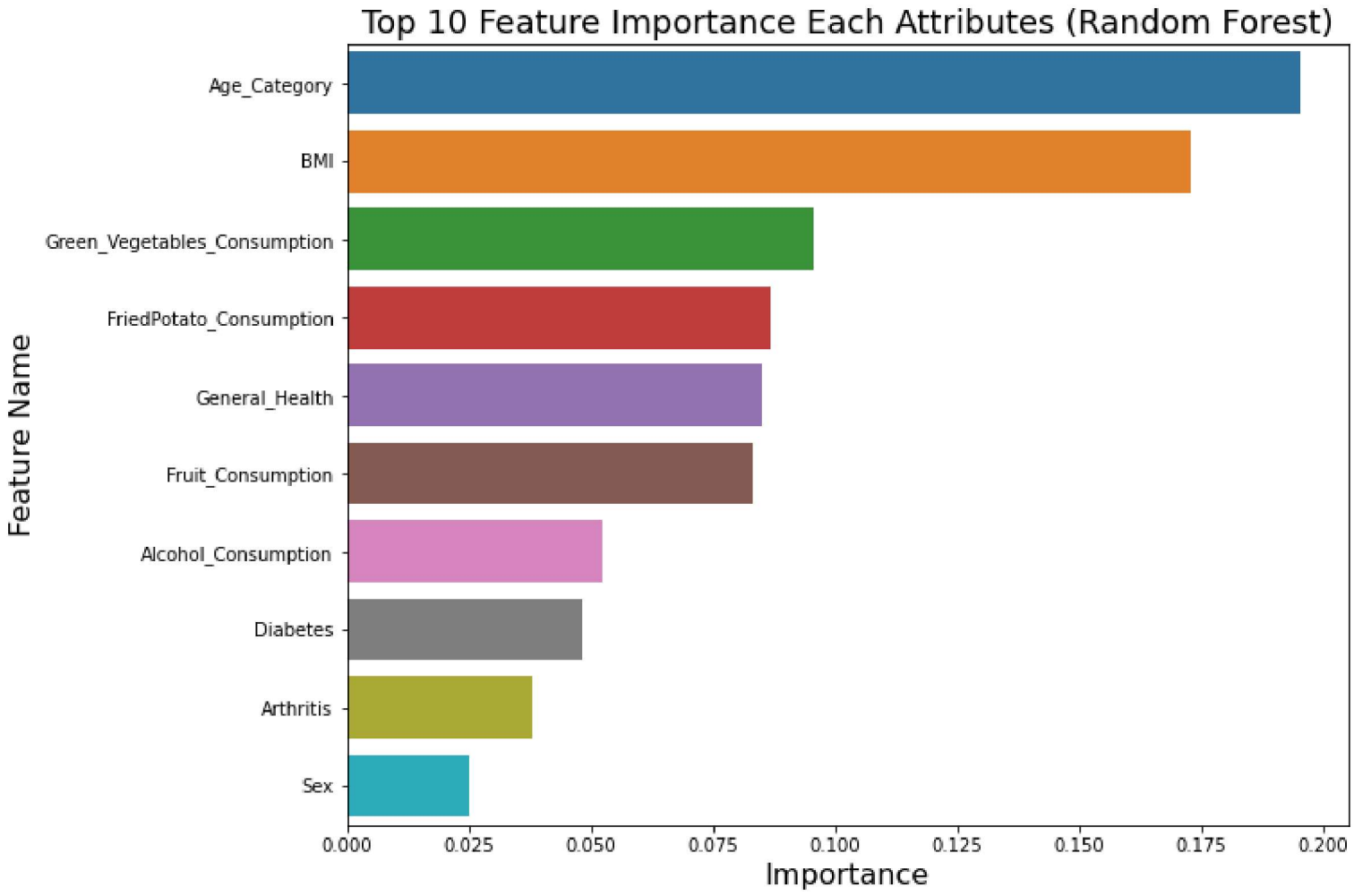
```
Accuracy Score: 91.39 %
```

In [77]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, jaccard_score
print('F-1 Score : ',(f1_score(y_test, y_pred, average='micro')))
print('Precision Score : ',(precision_score(y_test, y_pred, average='micro')))
print('Recall Score : ',(recall_score(y_test, y_pred, average='micro')))
print('Jaccard Score : ',(jaccard_score(y_test, y_pred, average='micro')))
print('Log Loss : ',(log_loss(y_test, y_pred)))
```

```
F-1 Score :  0.913855873219831
Precision Score :  0.9138558732198309
Recall Score :  0.9138558732198309
Jaccard Score :  0.8413762508010055
Log Loss :  2.9753161573619074
```

In [78]:
```
imp_df = pd.DataFrame({
    "Feature Name": X_train.columns,
    "Importance": rfc.feature_importances_
})
```

```
fi = imp_df.sort_values(by="Importance", ascending=False)

fi2 = fi.head(10)
plt.figure(figsize=(10,8))
sns.barplot(data=fi2, x='Importance', y='Feature Name')
plt.title('Top 10 Feature Importance Each Attributes (Random Forest)', fontsize=18)
```



Decision Tree Classifier

In [79]:
```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(random_state=0, min_samples_leaf=2, min_samples_split=6, class_weight='balanced')
dtree.fit(X_train, y_train)
```

Out[79]:
```
DecisionTreeClassifier(class_weight='balanced', max_depth=12,
                       min_samples_leaf=2, random_state=0)
```

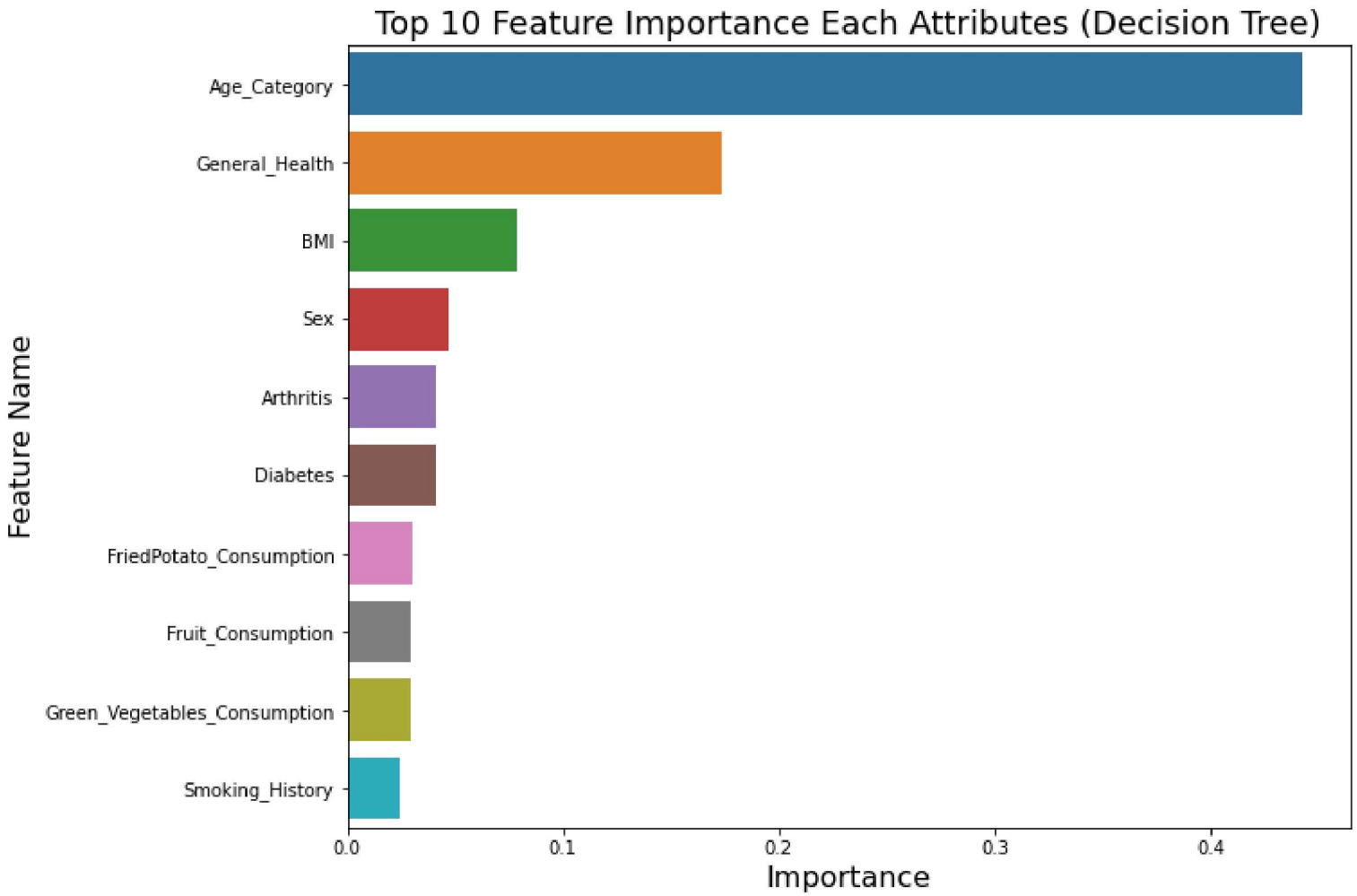
In [80]:
```
from sklearn.metrics import accuracy_score
y_pred = dtree.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2)
```

```
Accuracy Score : 71.87 %
```

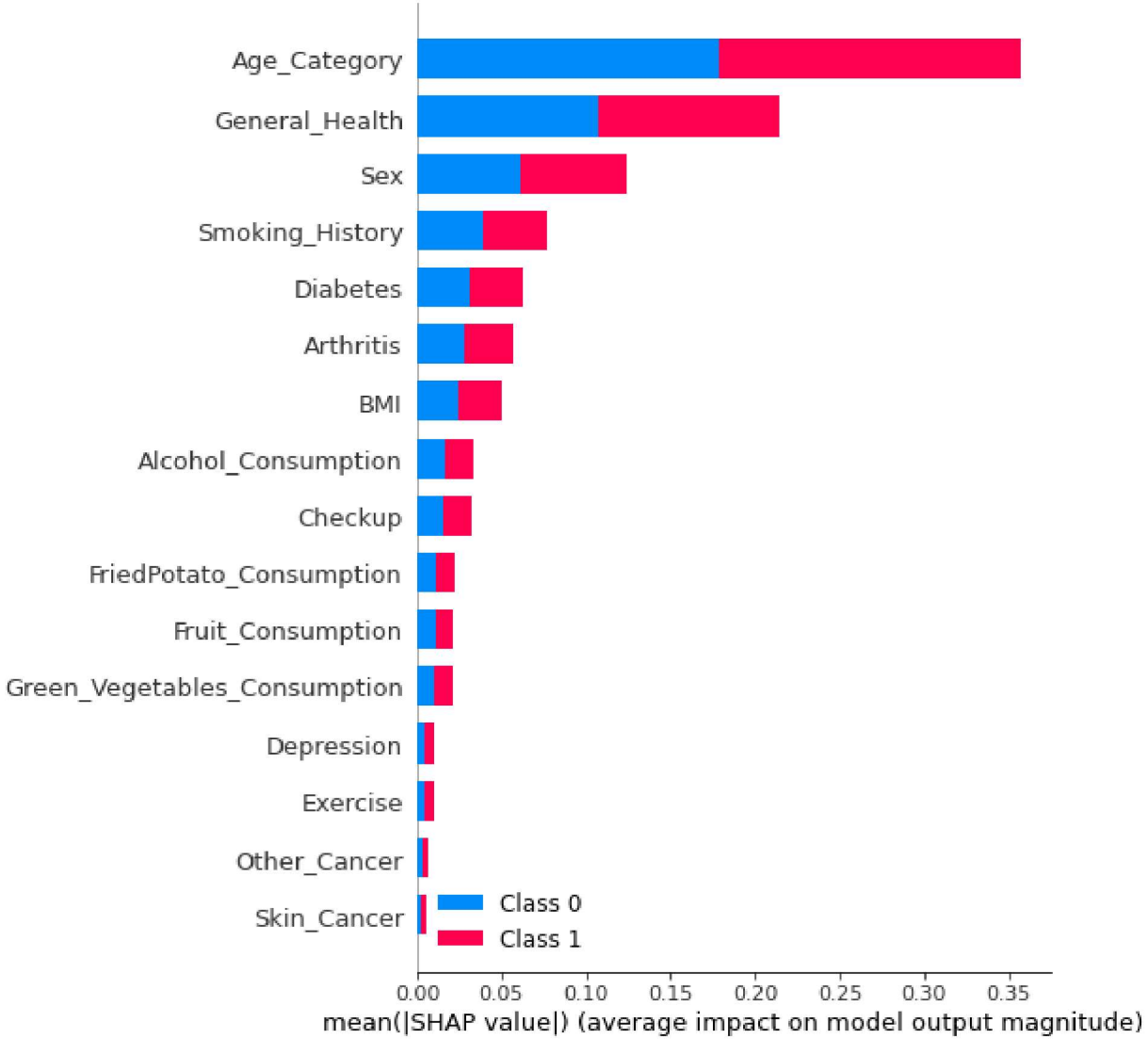
In [81]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, jaccard_score
print('F-1 Score : ',(f1_score(y_test, y_pred, average='micro')))
print('Precision Score : ',(precision_score(y_test, y_pred, average='micro')))
print('Recall Score : ',(recall_score(y_test, y_pred, average='micro')))
print('Jaccard Score : ',(jaccard_score(y_test, y_pred, average='micro')))
```

```
F-1 Score :  0.7186529607024307
Precision Score :  0.7186529607024307
Recall Score :  0.7186529607024307
Jaccard Score :  0.5608573935570134
```

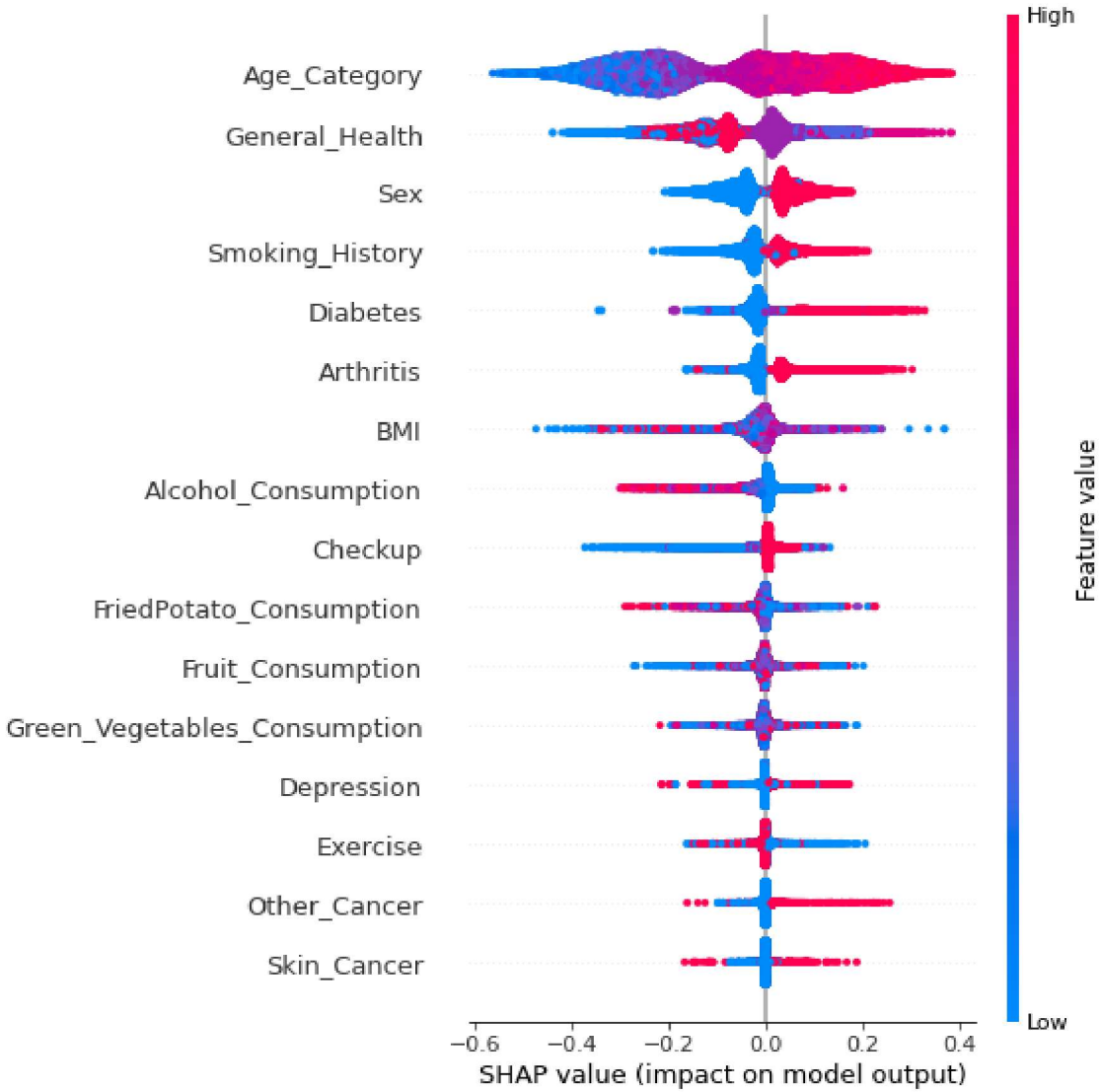
In [82]:
```
imp_df = pd.DataFrame({
    "Feature Name": X_train.columns,
    "Importance": dtree.feature_importances_
})
fi = imp_df.sort_values(by="Importance", ascending=False)

fi2 = fi.head(10)
plt.figure(figsize=(10,8))
sns.barplot(data=fi2, x='Importance', y='Feature Name')
plt.title('Top 10 Feature Importance Each Attributes (Decision Tree)')
plt.xlabel ('Importance')
plt.ylabel ('Feature Name')
plt.show()
```



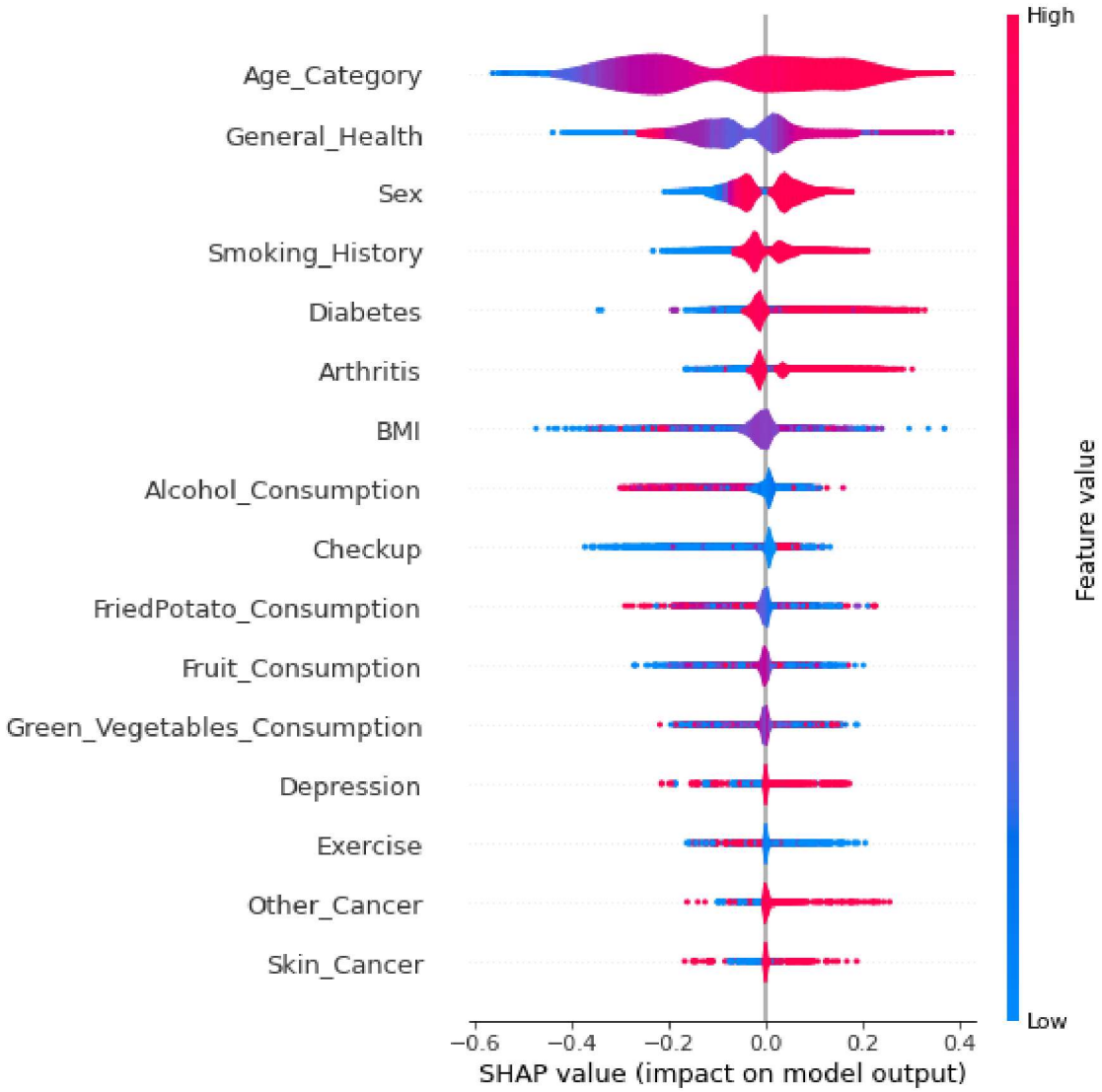
In [83]:
```
explainer = shap.TreeExplainer(dtree)
shap_values = explainer.shap_values(X_test)
shap.plot(shap_values, X_test)
```



In [84]:
```
explainer = shap.TreeExplainer(dtree)
shap_values = explainer.shap_values(X_test)
shap.plot(shap_values[1], X_test.values, feature_names = X_test.columns)
```

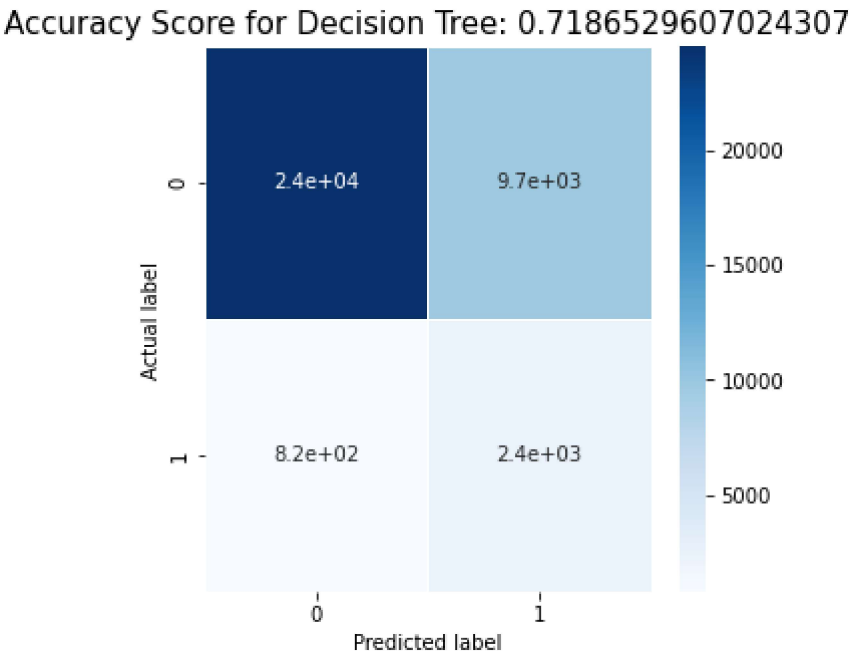


In [85]:
```
explainer = shap.TreeExplainer(dtree)
shap_values = explainer.shap_values(X_test)
shap.plot(shap_values[1], X_test.values, feature_names = X_test, plot_type="violin")
```



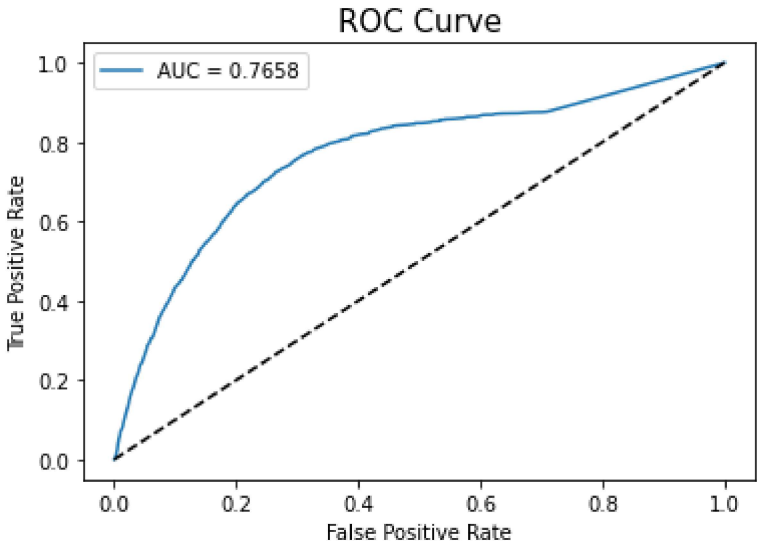
In [86]:
```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, cmap = 'Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score for Decision Tree: {0}'.format(dtree.score(X_test, y_test))
plt.title(all_sample_title, size = 15)
```

Out[86]:
```
Text(0.5, 1.0, 'Accuracy Score for Decision Tree: 0.7186529607024307')
```



In [87]:
```
from sklearn.metrics import roc_curve, roc_auc_score
y_pred_proba = dtree.predict_proba(X_test)

df_actual_predicted = pd.concat([pd.DataFrame(np.array(y_test), column=['y_actual']), pd.DataFrame(y_pred_proba, column
df_actual_predicted.index = y_test.index

fpr, tpr, tr = roc_curve(df_actual_predicted['y_actual'], df_actual_predicted['y_pred_proba'])
auc = roc_auc_score(df_actual_predicted['y_actual'], df_actual_predicted['y_pred_proba'])

plt.plot(fpr, tpr, label='AUC = %0.4f' %auc)
plt.plot(fpr, fpr, linestyle = '--', color='k')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

Out[87]:
```
<matplotlib.legend.Legend at 0x26b8b4af4f0>
```



Logistic Regression

In [88]:
```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
```

In [89]:
```
#Training the model
lr.fit(X_train, y_train)
```

Out[89]:
```
LogisticRegression()
```

In [90]:
```
#Training accuracy
lr.score(X_train, y_train)
```

Out[90]:
```
0.911942754919499
```

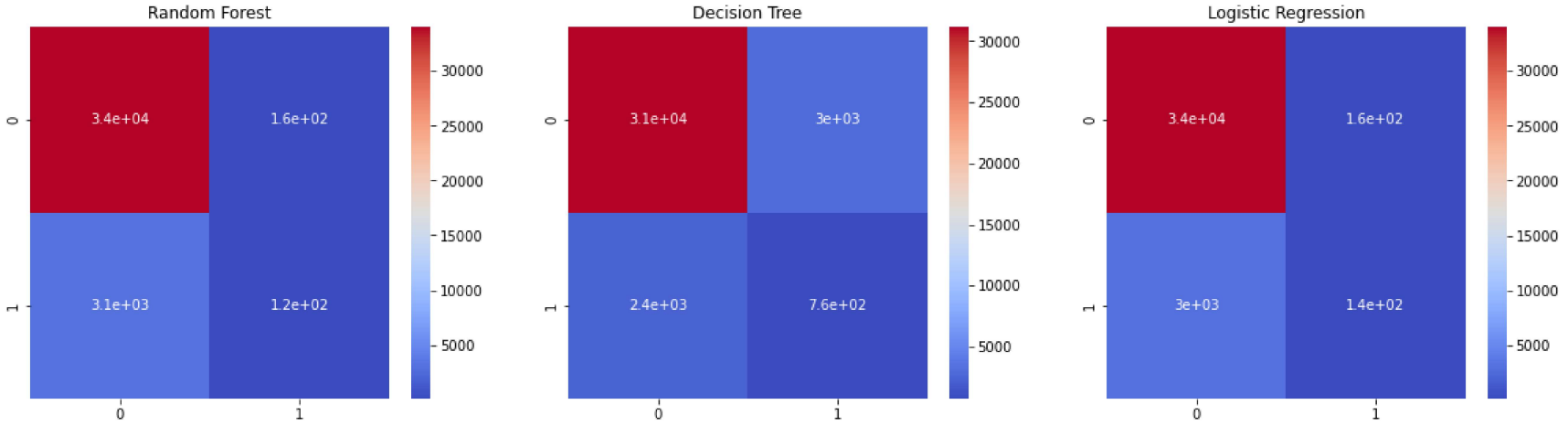
In [91]:
```
#Predicting the test set results
lr_pred = lr.predict(X_test)
```

Model Evalution

Confusion Matrix

In [93]:
```
from sklearn.tree import DecisionTreeClassifier

# Create a Decision Tree classifier
dtc = DecisionTreeClassifier()

# Train the Decision Tree classifier on your training data
dtc.fit(X_train, y_train)

# Make predictions on the test data
dtc_pred = dtc.predict(X_test)

# Now you can plot the confusion matrix for the Decision Tree classifier
fig, ax = plt.subplots(figsize=(20, 5))
sns.heatmap(confusion_matrix(y_test, rfc_pred), annot=True, cmap='coolwarm', ax=ax[0]).set_title('Random Forest')
sns.heatmap(confusion_matrix(y_test, dtc_pred), annot=True, cmap='coolwarm', ax=ax[1]).set_title('Decision Tree')
sns.heatmap(confusion_matrix(y_test, lr_pred), annot=True, cmap='coolwarm', ax=ax[2]).set_title('Logistic Regression')
```

Out[93]:
```
Text(0.5, 1.0, 'Logistic Regression')
```



In [94]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
print('Random Forest')
print('Accuracy Score: ', accuracy_score(y_test, rfc_pred))
print('Precision Score: ', precision_score(y_test, rfc_pred))
print('Recall Score: ', recall_score(y_test, rfc_pred))
print('F1 Score: ', f1_score(y_test, rfc_pred))
```

```
Random Forest
Accuracy Score:  0.9138558732198309
Precision Score:  0.4223826714801444
Recall Score:  0.0368503937007874
F1 Score:  0.06778679026651217
```

In [95]:
```
print('Decision Tree')
print('Accuracy Score: ', accuracy_score(y_test, dtc_pred))
print('Precision Score: ', precision_score(y_test, dtc_pred))
print('Recall Score: ', recall_score(y_test, dtc_pred))
print('F1 Score: ', f1_score(y_test, dtc_pred))
```

```
Decision Tree
Accuracy Score:  0.8547221329906842
Precision Score:  0.2021164021164021
Recall Score:  0.24062992125984253
F1 Score:  0.21969805895039538
```

In [96]:
```
print('Logistic Regression')
print('Accuracy Score: ', accuracy_score(y_test, lr_pred))
print('Precision Score: ', precision_score(y_test, lr_pred))
print('Recall Score: ', recall_score(y_test, lr_pred))
print('F1 Score: ', f1_score(y_test, lr_pred))
```

```
Logistic Regression
Accuracy Score:  0.9144983402933933
Precision Score:  0.46735395189003437
Recall Score:  0.042834645669291335
F1 Score:  0.07847663012117714
```

Conclusion

From the exploratory data analysis, we observed that the risk of cardiovascular disease increases with age, with individuals above 55 years being particularly prone to this condition. The highest number of cardiovascular disease cases was found in patients aged 80 and above. Additionally, higher BMI levels were associated with an increased likelihood of cardiovascular disease.

Interestingly, older patients who exercise appeared to be more prone to cardiovascular disease, possibly due to the strain on the heart. Dietary habits also played a role, as patients who consumed larger quantities of fruits and vegetables had a lower risk, while those who consumed fried potatoes had a higher risk.

Furthermore, smoking or a history of smoking was linked to an increased risk of cardiovascular disease. Surprisingly, previous medical history related to cancer, arthritis, diabetes, or depression had no major effect on the likelihood of cardiovascular disease.

In [ ]: