HEART DISEASE HEALTH INDICATORS:-

TEAM -04

1)AFFAN -02FE22BCS008

2)CHANDRAKALA -02FE22BCS027

3)RUTIKA -02FE22BCS088

4)ROHAN -02FE23BCS040

Feature set description

HighBP : Indicates if the person has been told by a health professional that they have High Blood Pressure.

HighChol : Indicates if the person has been told by a health professional that they have High Blood Cholesterol.

CholCheck : Cholesterol Check, if the person has their cholesterol levels checked within the last 5 years.

BMI : Body Mass Index, calculated by dividing the persons weight (in kilogram) by the square of their height (in meters).

Smoker : Indicates if the person has smoked at least 100 cigarettes.

Stroke : Indicates if the person has a history of stroke.

Diabetes : Indicates if the person has a history of diabetes, or currently in pre-diabetes, or suffers from either type of diabetes.

PhysActivity : Indicates if the person has some form of physical activity in their day-to-day routine.

Fruits : Indicates if the person consumes 1 or more fruit(s) daily.

Veggies : Indicates if the person consumes 1 or more vegetable(s) daily.

HvyAlcoholConsump : Indicates if the person has more than 14 drinks per week.

AnyHealthcare : Indicates if the person has any form of health insurance.

NoDocbcCost : Indicates if the person wanted to visit a doctor within the past 1 year but couldn’t, due to cost.

GenHlth : Indicates the persons response to how well is their general health, ranging from 1 (excellent) to 5 (poor).

Menthlth : Indicates the number of days, within the past 30 days that the person had bad mental health.

PhysHlth : Indicates the number of days, within the past 30 days that the person had bad physical health.

DiffWalk : Indicates if the person has difficulty while walking or climbing stairs.

Sex : Indicates the gender of the person, where 0 is female and 1 is male.

Age : Indicates the age class of the person, where 1 is 18 years to 24 years up till 13 which 80 years or older, each interval between has a 5-year increment.

Education : Indicates the highest year of school completed, with 0 being never attended or kindergarten only and 6 being, having attended 4 years of college or more.

Income : Indicates the total household income, ranging from 1 (at least $10,000) to 6 (75,000+)

Numeric Attributes (7):

BMI

Income

Education

Age

GenHlth

MentHlth

PhysHlth

Binary Attributes (15):

HeartDiseaseorAttack

HighBP

HighChol

CholCheck

Smoker

Stroke

Diabetes

PhysActivity

Fruits

Veggies

HvyAlcoholConsump

AnyHealthcare

NoDocbcCost

DiffWalk

Sex

Importing Libraries

In [27]:
```
import pandas as pd   #providing data structures and functions to efficiently
import numpy as np   #providing powerful tools for array manipulation and math
import matplotlib.pyplot as plt  #Matplotlib is a versatile Python library for
import seaborn as sns  # provides a high-level interface for creating informat
import scipy.stats as stats
```

Loading Dataset

In [28]:
```
data = pd.read_csv(r"D:\modified_dataset2.csv")
data
```

Out[28]:

	HeartDiseaseorAttack	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	Diabetes	P
0	0	1	1	1	40	1	0	0	
1	0	0	0	0	25	1	0	0	
2	0	1	1	1	28	0	0	0	
3	0	1	0	1	27	0	0	0	
4	0	1	1	1	24	0	0	0	
...	...	...	...	...	...	...	...	...	
253675	0	1	1	1	45	0	0	0	
253676	0	1	1	1	18	0	0	2	
253677	0	0	0	1	28	0	0	0	
253678	0	1	0	1	23	0	0	0	
253679	1	1	1	1	25	0	0	2	

253680 rows × 22 columns

Total Number Of Rows And Columns

In [3]:
```
print ("Numbers of rows and columns in dataset: ", data.shape)
```

```
Numbers of rows and columns in dataset:  (253680, 22)
```

Feature information and datatype

In [22]:
```
print("Info of the dataset\n")
data.info()
```

```
Info of the dataset

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 253680 entries, 0 to 253679
Data columns (total 22 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   HeartDiseaseorAttack  253680 non-null  float64
 1   HighBP                253680 non-null  float64
 2   HighChol              253680 non-null  float64
 3   CholCheck             228947 non-null  float64
 4   BMI                   253680 non-null  float64
 5   Smoker                253680 non-null  float64
 6   Stroke                253680 non-null  float64
 7   Diabetes              240996 non-null  float64
 8   PhysActivity          253680 non-null  float64
 9   Fruits                253680 non-null  float64
 10  Veggies               253680 non-null  float64
 11  HvyAlcoholConsump     253680 non-null  float64
 12  AnyHealthcare         253680 non-null  float64
 13  NoDocbcCost           253680 non-null  float64
 14  GenHlth               253680 non-null  float64
 15  MentHlth              253680 non-null  float64
 16  PhysHlth              253680 non-null  float64
 17  DiffWalk              253680 non-null  float64
 18  Sex                   253680 non-null  float64
 19  Age                   253680 non-null  float64
 20  Education             253680 non-null  float64
 21  Income                253680 non-null  float64
dtypes: float64(22)
memory usage: 42.6 MB
```

In [24]:
```
data.head()
```

Out[24]:

	HeartDiseaseorAttack	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	Diabetes	PhysAc
0	0.0	1.0	1.0	1.0	40.0	1.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	25.0	1.0	0.0	0.0	
2	0.0	1.0	1.0	1.0	28.0	0.0	0.0	0.0	
3	0.0	1.0	0.0	1.0	27.0	0.0	0.0	0.0	
4	0.0	1.0	1.0	1.0	24.0	0.0	0.0	0.0	

5 rows × 22 columns

Find the null values

In [45]:
```
data.isnull().sum()
```

Out[45]:
```
HeartDiseaseorAttack      0
HighBP                    0
HighChol                  0
CholCheck                 0
BMI                       0
Smoker                    0
Stroke                    0
Diabetes                  0
PhysActivity              0
Fruits                  200
Veggies                   0
HvyAlcoholConsump         0
AnyHealthcare           200
NoDocbcCost               0
GenHlth                   0
MentHlth                  0
PhysHlth                  0
DiffWalk                  0
Sex                       0
Age                       0
Education                 0
Income                    0
dtype: int64
```

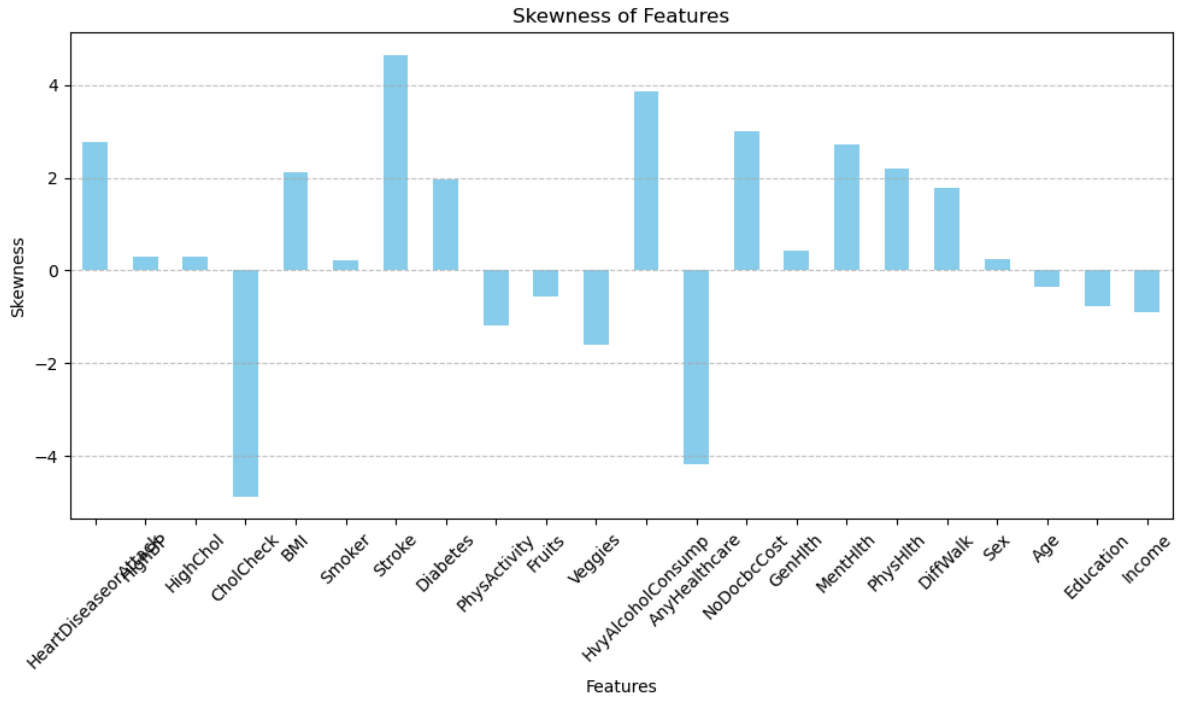
Find the skewness of all the features

Skewness measures the degree and direction of asymmetry in a distribution. A distribution is symmetric if its left and right tails are mirror images of each other. Skewness measures how much a distribution deviates from this symmetry.

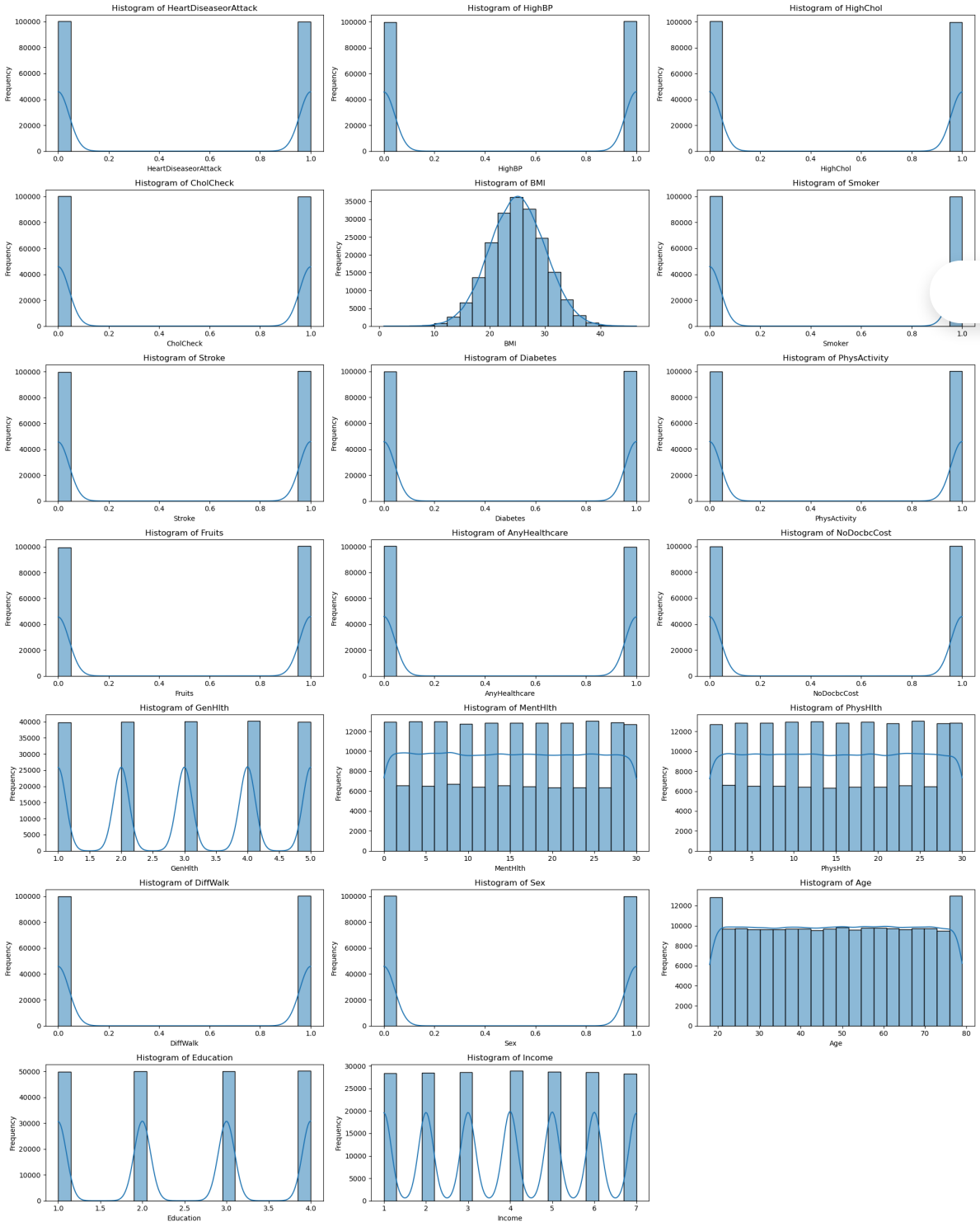
In [5]:
```
#Select only numeric columns
numeric_cols = data.select_dtypes (include=['float64', 'int64'])
# Calculate skewness for each numeric feature
skewness = numeric_cols.skew()
#Display the skewness of each feature

print("Skewness of each numeric feature:")

print(skewness)
```

```
Skewness of each numeric feature:
HeartDiseaseorAttack    2.778742
HighBP                  0.286904
HighChol                0.307075
CholCheck              -4.881271
BMI                     2.122004
Smoker                  0.228810
Stroke                  4.657340
Diabetes                1.976390
PhysActivity           -1.195546
Fruits                 -0.557515
Veggies                -1.592239
HvyAlcoholConsump       3.854132
AnyHealthcare          -4.181157
NoDocbcCost             2.995290
GenHlth                 0.422867
MentHlth                2.721148
PhysHlth                2.207395
DiffWalk                1.773907
```

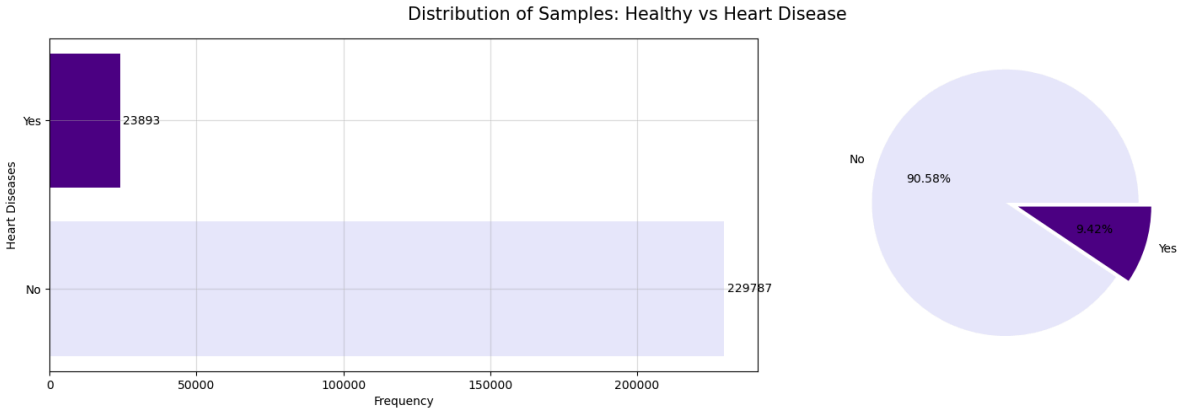
In [6]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
skewness.plot(kind='bar', color='skyblue')
plt.title('Skewness of Features')
plt.xlabel('Features')
plt.ylabel('Skewness')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



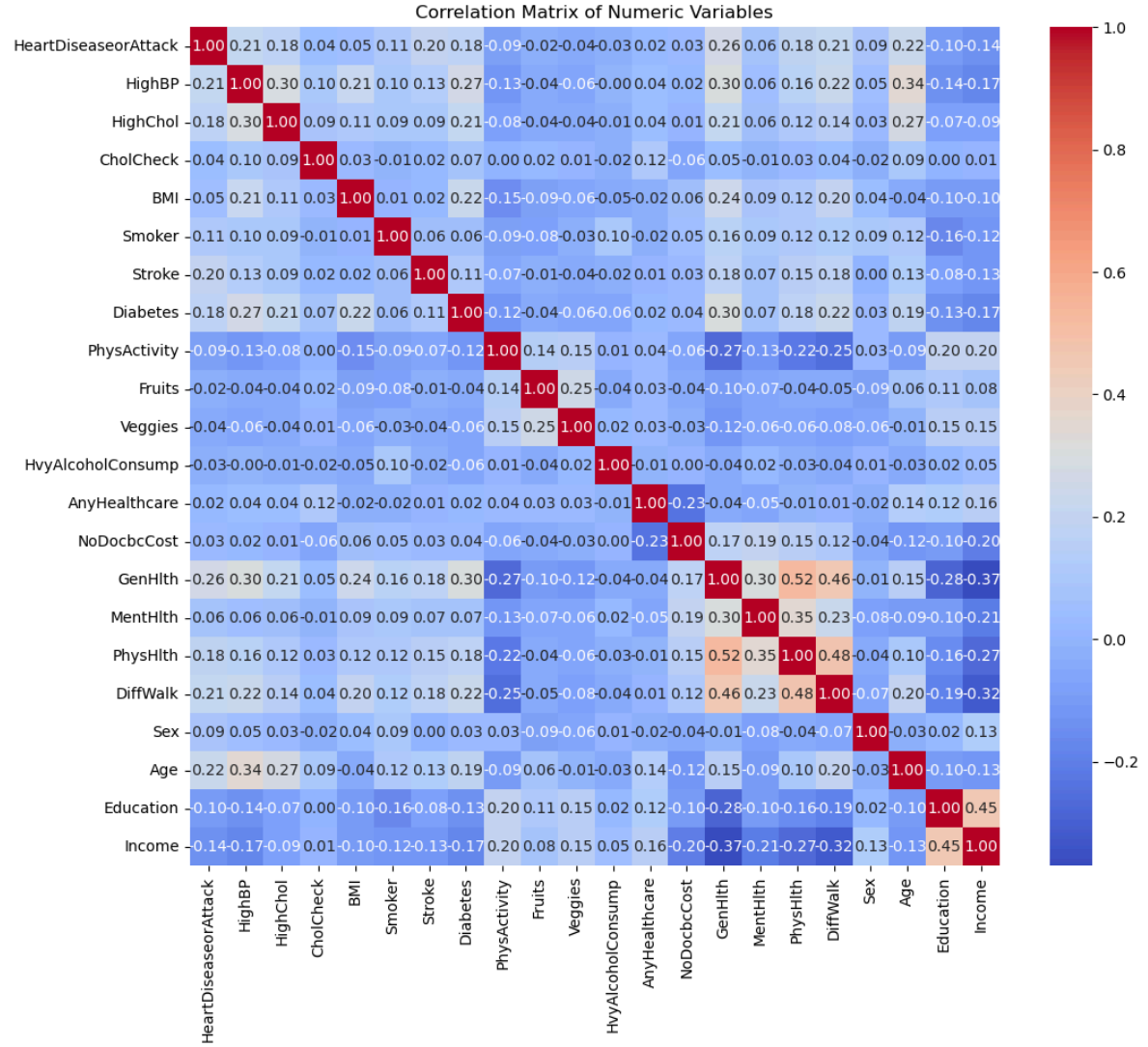
In [31]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming df is your DataFrame containing the dataset
# Replace df with your actual dataset variable

# List of variables to visualize
variables = ['HeartDiseaseorAttack', 'HighBP', 'HighChol', 'CholCheck', 'BMI',
             'Stroke', 'Diabetes', 'PhysActivity', 'Fruits', 'AnyHealthcar
             'GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk', 'Sex', 'Age',

# Plotting histograms for numerical variables and count plots for categorical
plt.figure(figsize=(20, 25))

# Iterate through each variable
for i, var in enumerate(variables):
    plt.subplot(7, 3, i+1)
    if df[var].dtype == 'object':
        # Plot count plot for categorical variables
        sns.countplot(x=var, data=df)
        plt.title(f'Count Plot of {var}')
        plt.xlabel(var)
        plt.ylabel('Count')
        plt.xticks(rotation=45)
    else:
        # Plot histogram for numerical variables
        sns.histplot(df[var], bins=20, kde=True)
        plt.title(f'Histogram of {var}')
        plt.xlabel(var)
        plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```


Histogram of HeartDiseaseorAttack
Frequency
HeartDiseaseorAttack
Histogram of HighBP
HighBP
Histogram of HighChol
HighChol
Histogram of CholCheck
CholCheck
Histogram of BMI
BMI
Histogram of Smoker
Smoker
Histogram of Stroke
Stroke
Histogram of Diabetes
Diabetes
Histogram of PhysActivity
PhysActivity
Histogram of Fruits
Fruits
Histogram of AnyHealthcare
AnyHealthcare
Histogram of NoDocbcCost
NoDocbcCost
Histogram of GenHlth
GenHlth
Histogram of MentHlth
MentHlth
Histogram of PhysHlth
PhysHlth
Histogram of DiffWalk
DiffWalk
Histogram of Sex
Sex
Histogram of Age
Age
Histogram of Education
Education
Histogram of Income
Income

In [29]:
```
# distribution of the classes in training dataset
targets = data.HeartDiseaseorAttack.value_counts()
colors = ['lavender','indigo']

# visualization
figure, axes = plt.subplots(1,2, figsize=(15,5), gridspec_kw={'width_ratios':[
axes[0].barh(y=targets.index, width=targets.values, color=colors)
axes[0].set_xlabel('Frequency')
axes[0].set_ylabel('Heart Diseases')
axes[0].set_yticks([0,1], ['No','Yes'])
axes[0].grid(alpha=0.4)

for index, values in enumerate(targets):
    axes[0].text(values+1000, index, str(values), va='center')

axes[1].pie(targets.values, labels=['No','Yes'], autopct='%.2f%%', explode=([0

figure.suptitle('Distribution of Samples: Healthy vs Heart Disease', fontsize=
plt.tight_layout(pad=1)
plt.show()
```



In [7]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load your dataset
#df = pd.read_csv('liver_cirrhosis_with_null1.csv')

# Select numeric columns only
numeric_columns = data.select_dtypes(include=['float64', 'int64'])

# Fill missing values with median for numeric columns
numeric_columns.fillna(numeric_columns.median(), inplace=True)

# Compute the correlation matrix for numeric columns
correlation_matrix_numeric = numeric_columns.corr()

# Plotting the correlation matrix
plt.figure(figsize=(12, 10), facecolor='white')
sns.heatmap(correlation_matrix_numeric, annot=True, fmt='.2f', cmap='coolwarm'
plt.title('Correlation Matrix of Numeric Variables')
plt.show()
```



Detecting Outliers

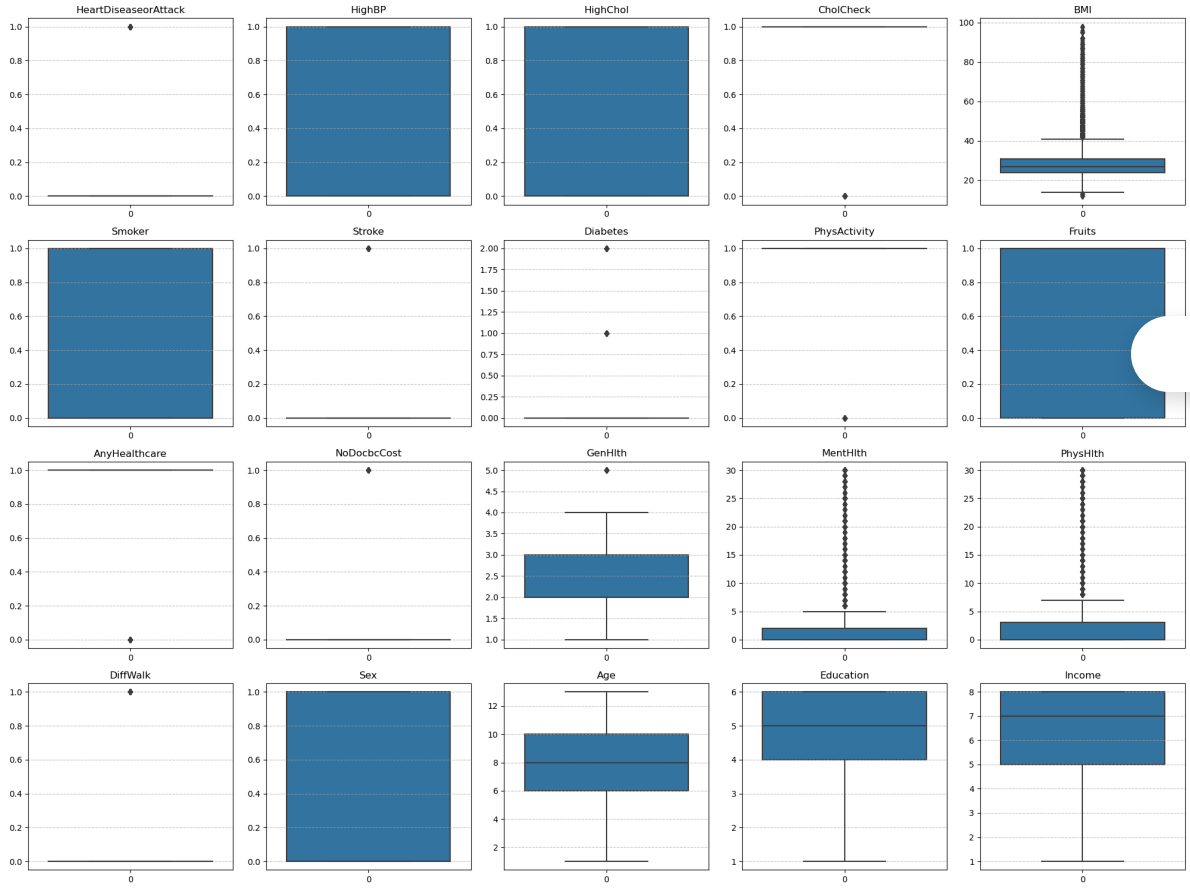
Detecting outliers involves identifying observations in a dataset that significantly deviate from the rest of the data. Outliers can occur due to various reasons such as data entry errors, measurement errors, or genuine anomalies in the data.

BOX PLOT TO REPRESENT THE OUTLIERS

In [13]:
```
import pandas as pd

# Function to count outliers in a column
def count_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return ((column < lower_bound) | (column > upper_bound)).sum()

# List of attributes
attributes = [
    'HeartDiseaseorAttack', 'HighBP', 'HighChol',
    'CholCheck', 'BMI', 'Smoker',
    'Stroke', 'Diabetes', 'PhysActivity',
    'Fruits', 'AnyHealthcare', 'NoDocbcCost',
    'GenHlth', 'MentHlth', 'PhysHlth',
    'DiffWalk', 'Sex', 'Age',
    'Education', 'Income'
]

# Dictionary to store outlier counts
outlier_counts = {}

# Calculate outliers for each attribute and store in the dictionary
for attribute in attributes:
    outlier_counts[attribute] = count_outliers(data[attribute])

# Convert the dictionary to a DataFrame for better visualization
outlier_counts_df = pd.DataFrame(list(outlier_counts.items()), columns=['Attri

# Display the DataFrame
print(outlier_counts_df)
```

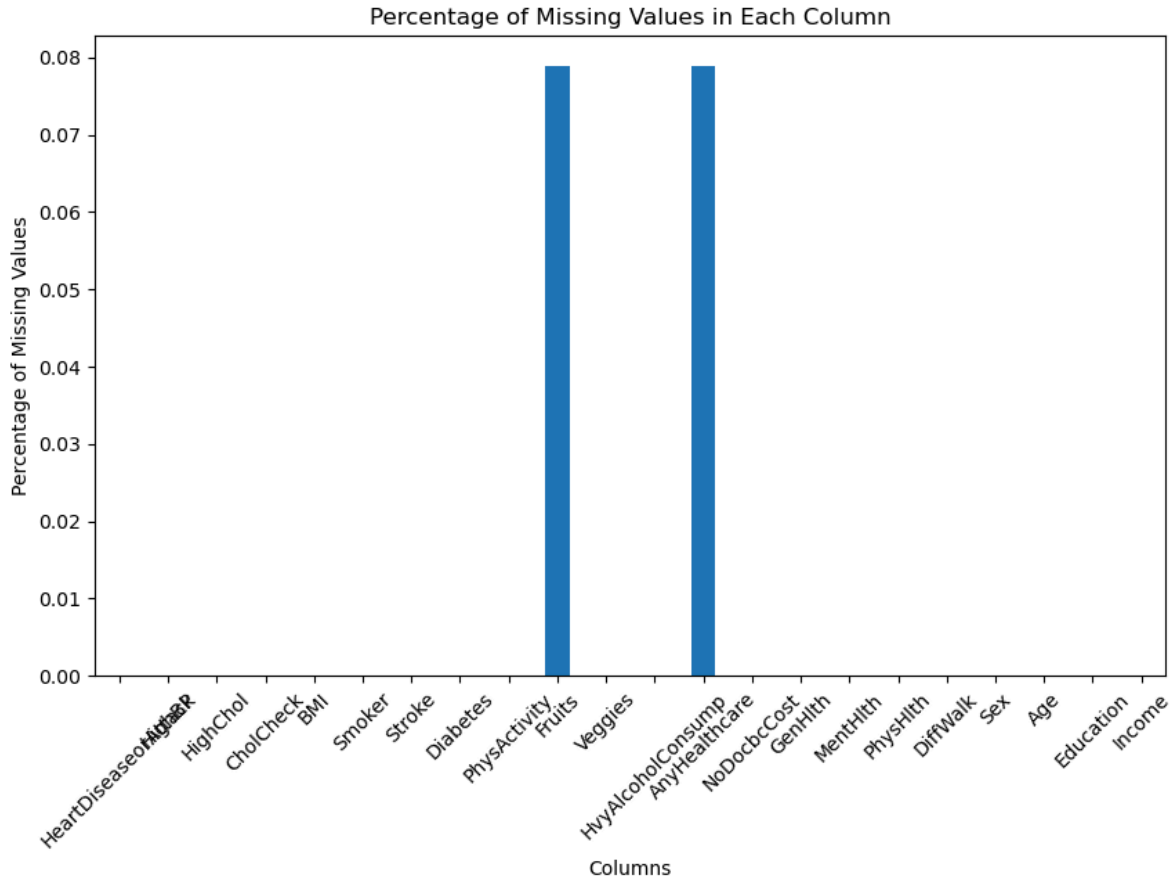
	Attribute	Outlier Count
0	HeartDiseaseorAttack	23893
1	HighBP	0
2	HighChol	0
3	CholCheck	9470
4	BMI	9847
5	Smoker	0
6	Stroke	10292
7	Diabetes	39977
8	PhysActivity	61760
9	Fruits	0
10	AnyHealthcare	12407
11	NoDocbcCost	21354
12	GenHlth	12081
13	MentHlth	36208
14	PhysHlth	40949
15	DiffWalk	42675
16	Sex	0
17	Age	0
18	Education	0
19	Income	0

In [12]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'data' is a DataFrame containing the attributes

# List of attributes to plot
attributes = [
    'HeartDiseaseorAttack', 'HighBP', 'HighChol',
    'CholCheck', 'BMI', 'Smoker',
    'Stroke', 'Diabetes', 'PhysActivity',
    'Fruits', 'AnyHealthcare', 'NoDocbcCost',
    'GenHlth', 'MentHlth', 'PhysHlth',
    'DiffWalk', 'Sex', 'Age',
    'Education', 'Income'
]

# Create a 4x5 grid for the boxplots
fig, axes = plt.subplots(4, 5, figsize=(20, 15))

# Flatten the axes array for easy iteration
axes = axes.flatten()

# Plot each attribute in a separate subplot
for i, attribute in enumerate(attributes):
    sns.boxplot(data[attribute], ax=axes[i])
    axes[i].set_title(attribute)
    axes[i].grid(axis='y', linestyle='--', alpha=0.7)

# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```



REPRESENTATION OF NULL VALUES USING BAR PLOT

In [53]:
```
# Calculate the percentage of missing values in each column
missing_percentage = (data.isnull().sum() / len(data)) * 100

# Create a bar plot of missing values for all columns
plt.figure(figsize=(10, 6))
missing_percentage.plot(kind='bar')
plt.xlabel('Columns')
plt.ylabel('Percentage of Missing Values')
plt.title('Percentage of Missing Values in Each Column')
plt.xticks(rotation=45)
plt.show()
print("                          Only cholestrol check and diabetes has the nu
```



Only cholestrol check and diabetes has the null val
ues

DATA CLEANING

In [36]:
```
import pandas as pd

# Assuming 'data' is your DataFrame and 'Diabetes' is the column with missing
median_Diabetes = data['Fruits'].median()

# Fill missing values in 'Diabetes' column with the median
data['Fruits'].fillna(median_Diabetes, inplace=True)

# Verify if missing values have been filled
print("Number of missing values in Fruits column after filling:", data['Ch
print("Number of missing values in AnyHlthcare column after filling:", dat
```

```
Number of missing values in Fruits column after filling: 0
Number of missing values in AnyHlthcare column after filling: 0
```

In [24]:
```
data.isnull().sum()
```

Out[24]:
```
HeartDiseaseorAttack    0
HighBP                  0
HighChol                0
CholCheck               0
BMI                     0
Smoker                  0
Stroke                  0
Diabetes                0
PhysActivity            0
Fruits                  0
AnyHealthcare           0
NoDocbcCost             0
GenHlth                 0
MentHlth                0
PhysHlth                0
DiffWalk                0
Sex                     0
Age                     0
Education               0
Income                  0
dtype: int64
```

Remove outliers

The method used to remove outliers in the provided code is based on the z-score method.

In [37]:
```
import pandas as pd
import numpy as np

def remove_outliers(data, column, threshold=0.5):
    z_scores = (data[column] - data[column].mean()) / data[column].std()
    filtered_data = data[abs(z_scores) < threshold]
    return filtered_data

# Load the dataset
data = pd.read_csv(r"D:\modified_dataset2.csv")

# Define numerical columns
numerical_columns = ['BMI', 'GenHlth', 'MentHlth', 'PhysHlth','GenHlth','MentH



# Remove outliers for each numerical column
clear_data = data.copy()  # Initialize clear_data with the original data
for column in numerical_columns:
    clear_data = remove_outliers(clear_data, column)

# Print the cleaned dataset
print("Cleaned dataset after removing outliers:")
print(clear_data)
```

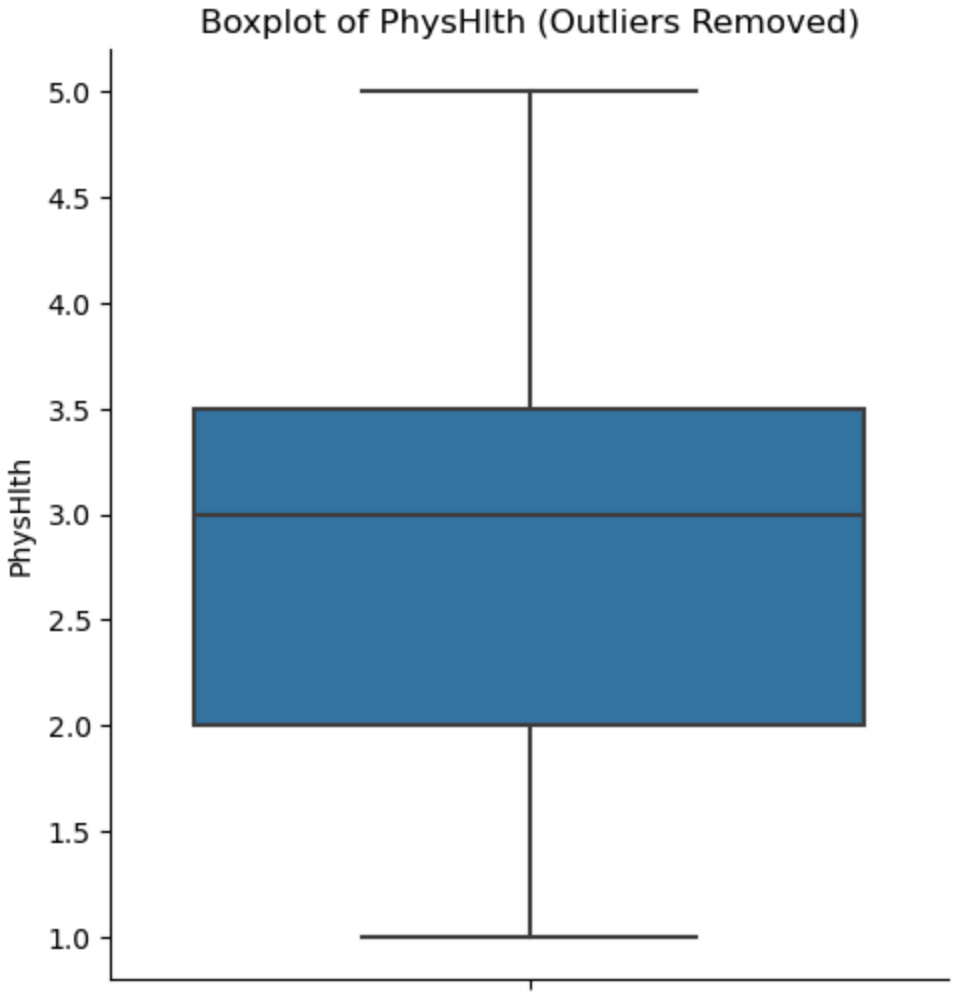
```
Cleaned dataset after removing outliers:
Empty DataFrame
Columns: [HeartDiseaseorAttack, HighBP, HighChol, CholCheck, BMI, Smoker, Str
oke, Diabetes, PhysActivity, Fruits, Veggies, HvyAlcoholConsump, AnyHealthcar
e, NoDocbcCost, GenHlth, MentHlth, PhysHlth, DiffWalk, Sex, Age, Education, I
ncome]
Index: []

[0 rows x 22 columns]
```

In [38]:
```
import pandas as pd

# Function to count outliers in a column
def count_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return ((column < lower_bound) | (column > upper_bound)).sum()

# List of attributes
attributes = [
    'HeartDiseaseorAttack', 'HighBP', 'HighChol',
    'CholCheck', 'BMI', 'Smoker',
    'Stroke', 'Diabetes', 'PhysActivity',
    'Fruits', 'AnyHealthcare', 'NoDocbcCost',
    'GenHlth', 'MentHlth', 'PhysHlth',
    'DiffWalk', 'Sex', 'Age',
    'Education', 'Income'
]

# Dictionary to store outlier counts
outlier_counts = {}

# Calculate outliers for each attribute and store in the dictionary
for attribute in attributes:
    outlier_counts[attribute] = count_outliers(clear_data[attribute])

# Convert the dictionary to a DataFrame for better visualization
outlier_counts_df = pd.DataFrame(list(outlier_counts.items()), columns=['Attri

# Display the DataFrame
print(outlier_counts_df)
```

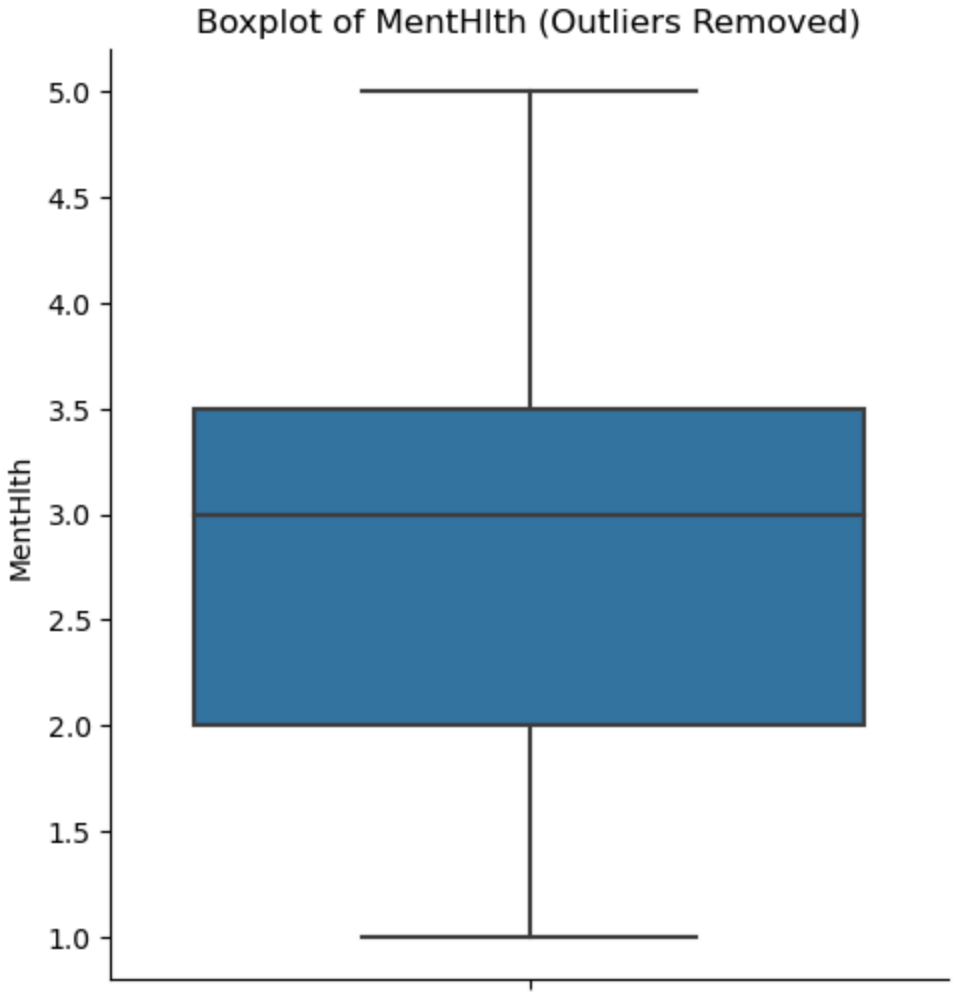
	Attribute	Outlier Count
0	HeartDiseaseorAttack	0
1	HighBP	0
2	HighChol	0
3	CholCheck	0
4	BMI	0
5	Smoker	0
6	Stroke	0
7	Diabetes	0
8	PhysActivity	0
9	Fruits	0
10	AnyHealthcare	0
11	NoDocbcCost	0
12	GenHlth	0
13	MentHlth	0
14	PhysHlth	0
15	DiffWalk	0
16	Sex	0
17	Age	0
18	Education	0
19	Income	0

In [39]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Example data
data = [
    {'PhysHlth': 1, 'MentHlth': 1, 'BMI': 22.0, 'HeartDiseaseorAttack': 1, 'Ch
     'Stroke': 1, 'Diabetes': 1, 'PhysActivity': 1, 'DiffWalk': 1, 'NoDocbcCos
    {'PhysHlth': 2, 'MentHlth': 2, 'BMI': 24.5, 'HeartDiseaseorAttack': 2, 'Ch
     'Stroke': 2, 'Diabetes': 2, 'PhysActivity': 2, 'DiffWalk': 2, 'NoDocb
    {'PhysHlth': 3, 'MentHlth': 3, 'BMI': 26.1, 'HeartDiseaseorAttack': 3,
     'Stroke': 3, 'Diabetes': 3, 'PhysActivity': 3, 'DiffWalk': 3, 'NoDocbcCos
    {'PhysHlth': 4, 'MentHlth': 4, 'BMI': 19.8, 'HeartDiseaseorAttack': 4, 'Ch
     'Stroke': 4, 'Diabetes': 4, 'PhysActivity': 4, 'DiffWalk': 4, 'NoDocbcCos
    {'PhysHlth': 100, 'MentHlth': 100, 'BMI': 45.0, 'HeartDiseaseorAttack': 1,
     'Stroke': 5, 'Diabetes': 0, 'PhysActivity': 5, 'DiffWalk': 5, 'NoDocbcCos
    {'PhysHlth': 5, 'MentHlth': 5, 'BMI': 23.7, 'HeartDiseaseorAttack': 5, 'Ch
     'Stroke': 6, 'Diabetes': 0, 'PhysActivity': 6, 'DiffWalk': 6, 'NoDocbcCos
    {'PhysHlth': 2, 'MentHlth': 2, 'BMI': 27.4, 'HeartDiseaseorAttack': 6, 'Ch
     'Stroke': 0, 'Diabetes': 1, 'PhysActivity': 0, 'DiffWalk': 1, 'NoDocbcCos
    {'PhysHlth': 3, 'MentHlth': 3, 'BMI': 21.9, 'HeartDiseaseorAttack': 0, 'Ch
     'Stroke': 1, 'Diabetes': 0, 'PhysActivity': 1, 'DiffWalk': 0, 'NoDocbcCos
]

# Convert the list of dictionaries to a DataFrame
df = pd.DataFrame(data)

# Function to remove outliers for multiple columns
def remove_outliers(df, columns):
    for column in columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
    return df

# Remove outliers for all attributes
columns_to_clean = ['PhysHlth', 'MentHlth', 'BMI', 'HeartDiseaseorAttack', 'Ch
                    'Stroke', 'Diabetes', 'PhysActivity', 'DiffWalk', 'NoDocbc
filtered_df = remove_outliers(df, columns_to_clean)

# Create boxplots for each attribute
for column in columns_to_clean:
    sns.catplot(y=column, kind='box', data=filtered_df)
    plt.ylabel(column)
    plt.xlabel('')
    plt.title(f'Boxplot of {column} (Outliers Removed)')
    plt.show()

print("Outliers have been removed from all attributes.")
```

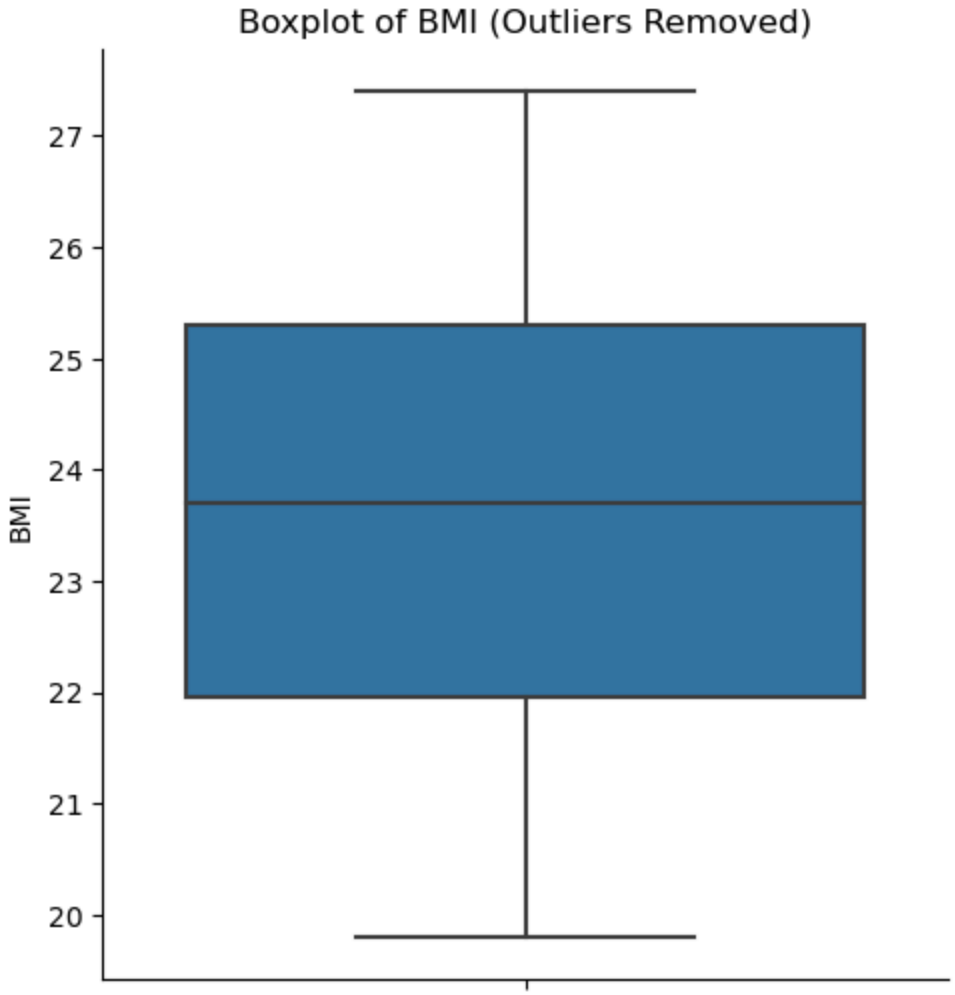
```
C:\Users\rutik\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



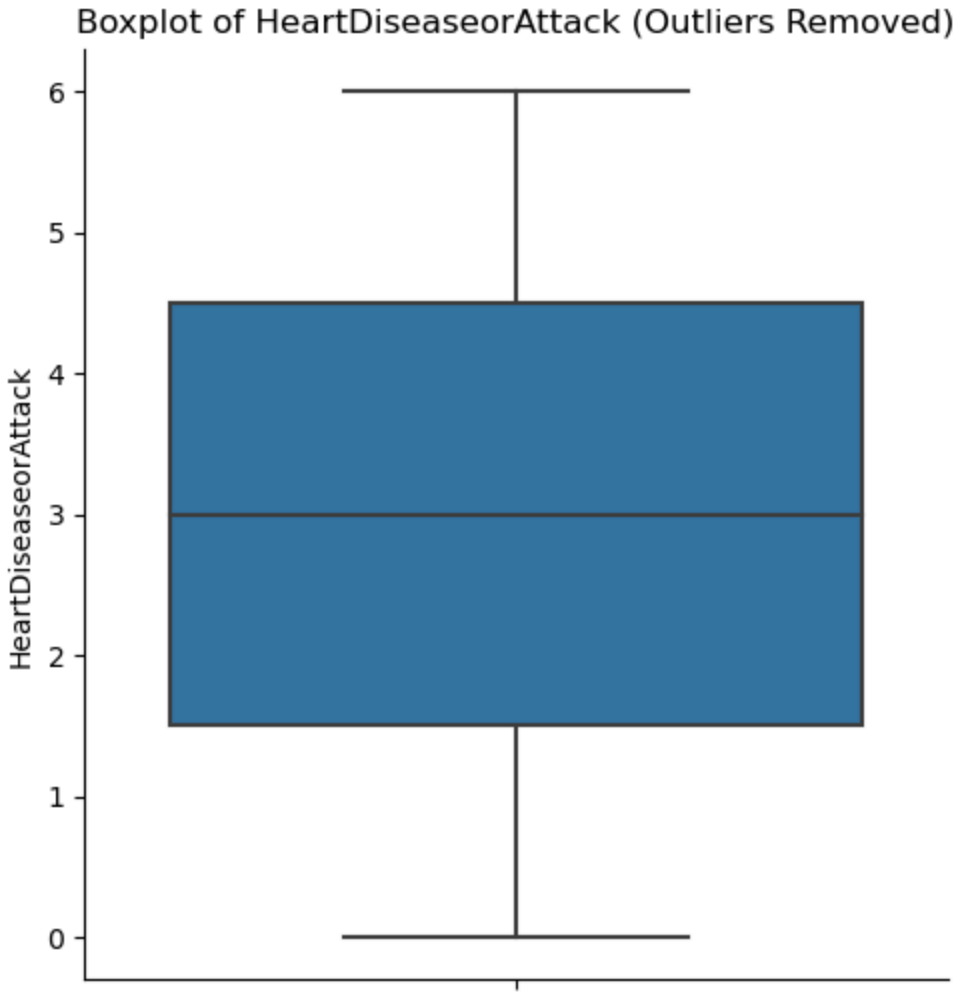

```
C:\Users\rutik\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



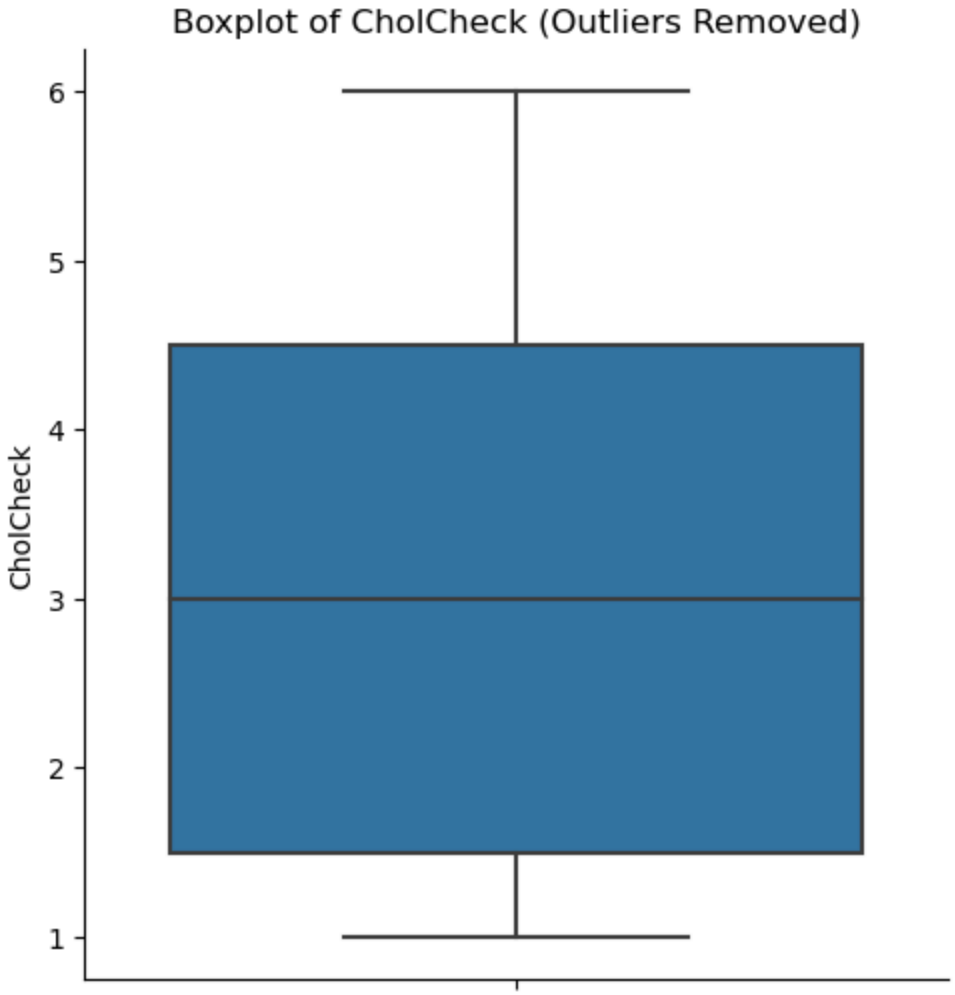

```
C:\Users\rutik\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



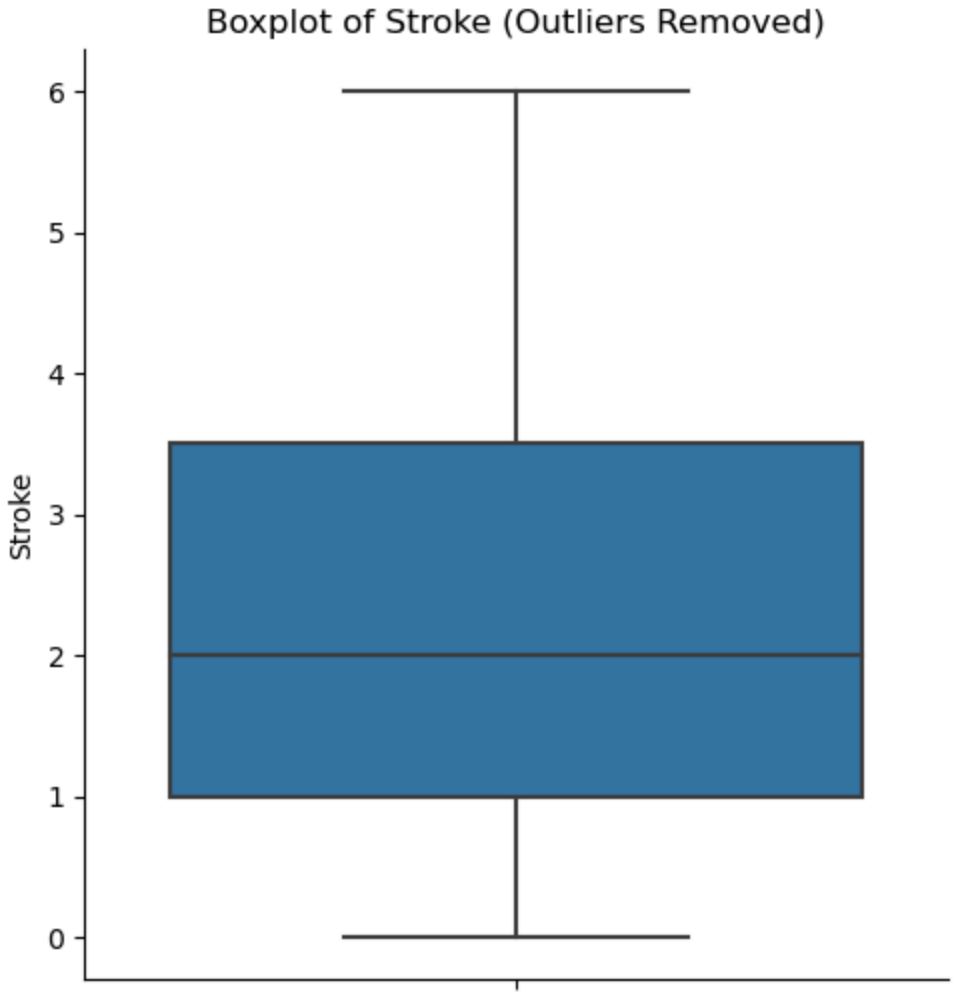

```
C:\Users\rutik\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



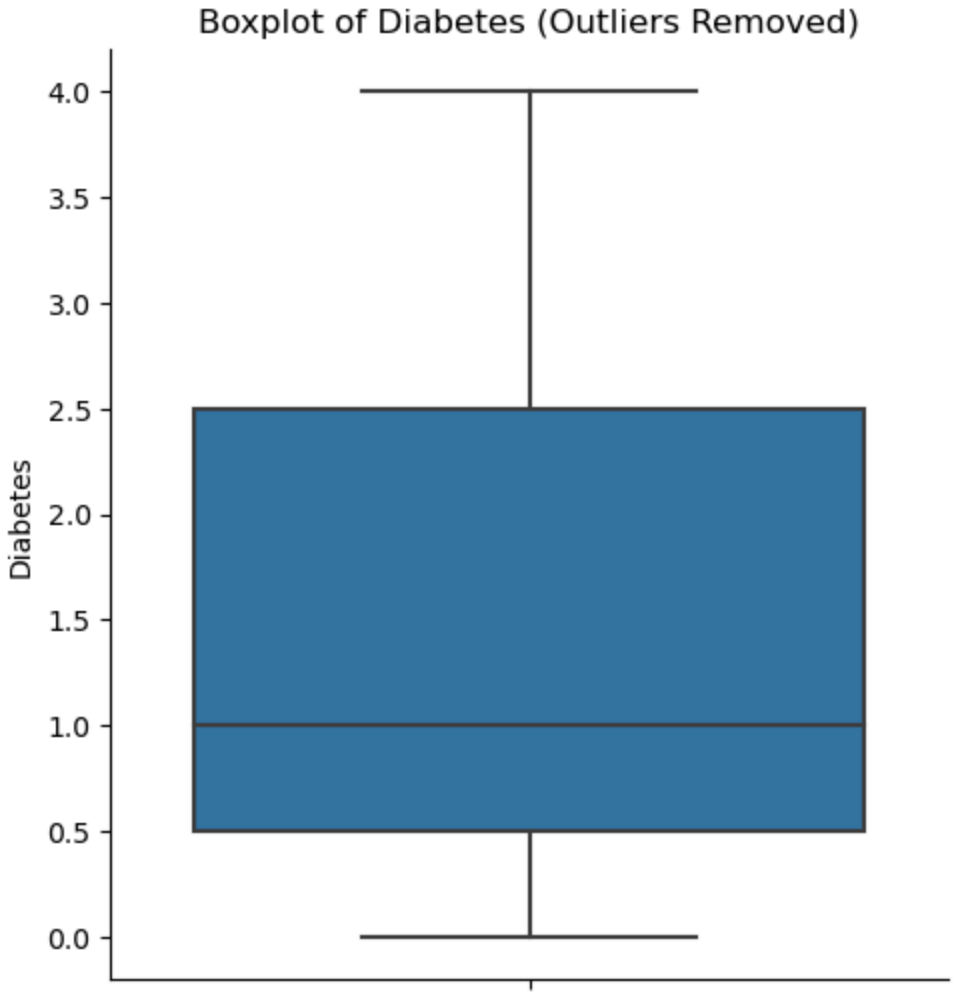

```
C:\Users\rutik\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



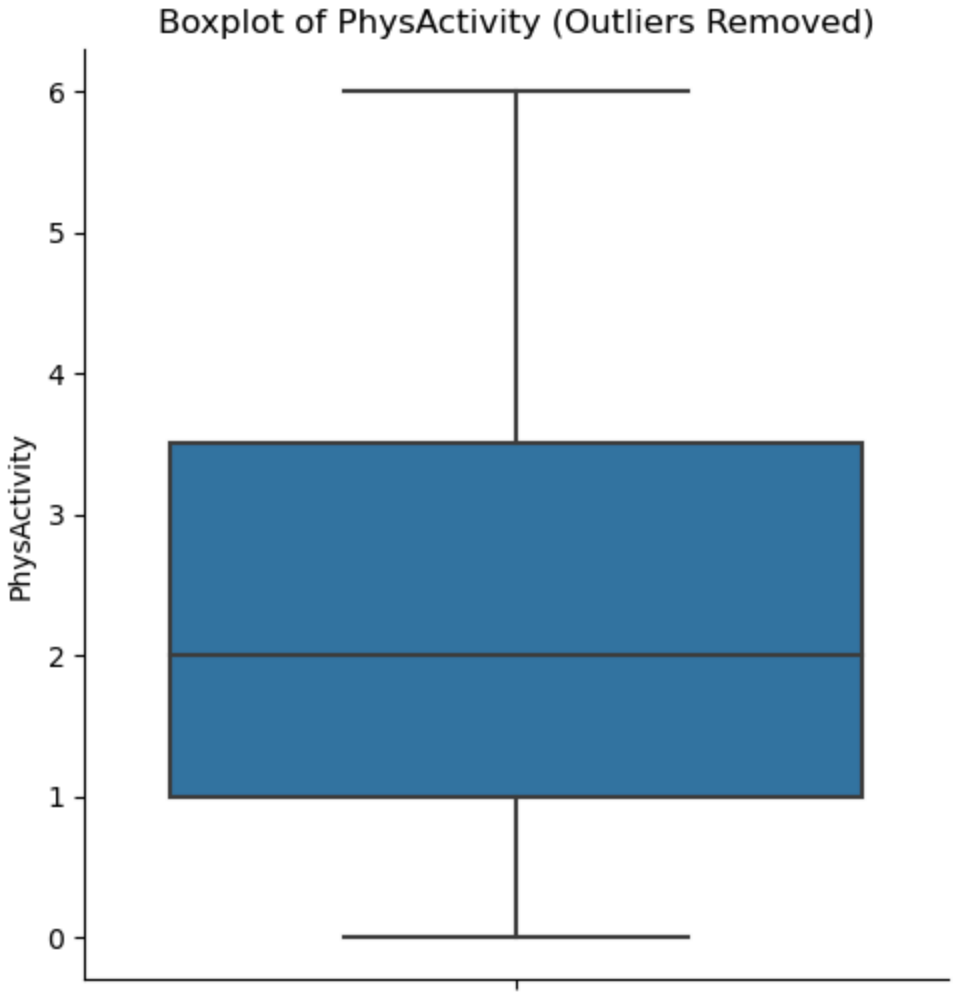

```
C:\Users\rutik\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



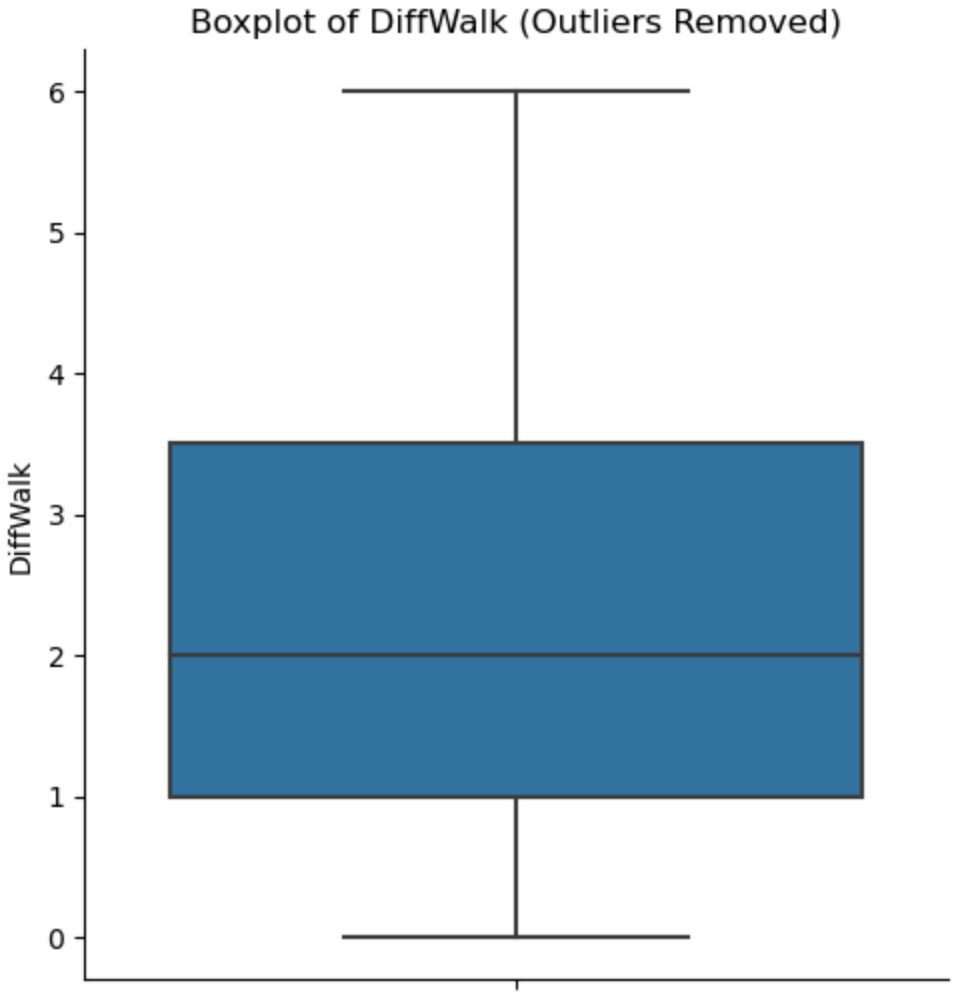

```
C:\Users\rutik\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



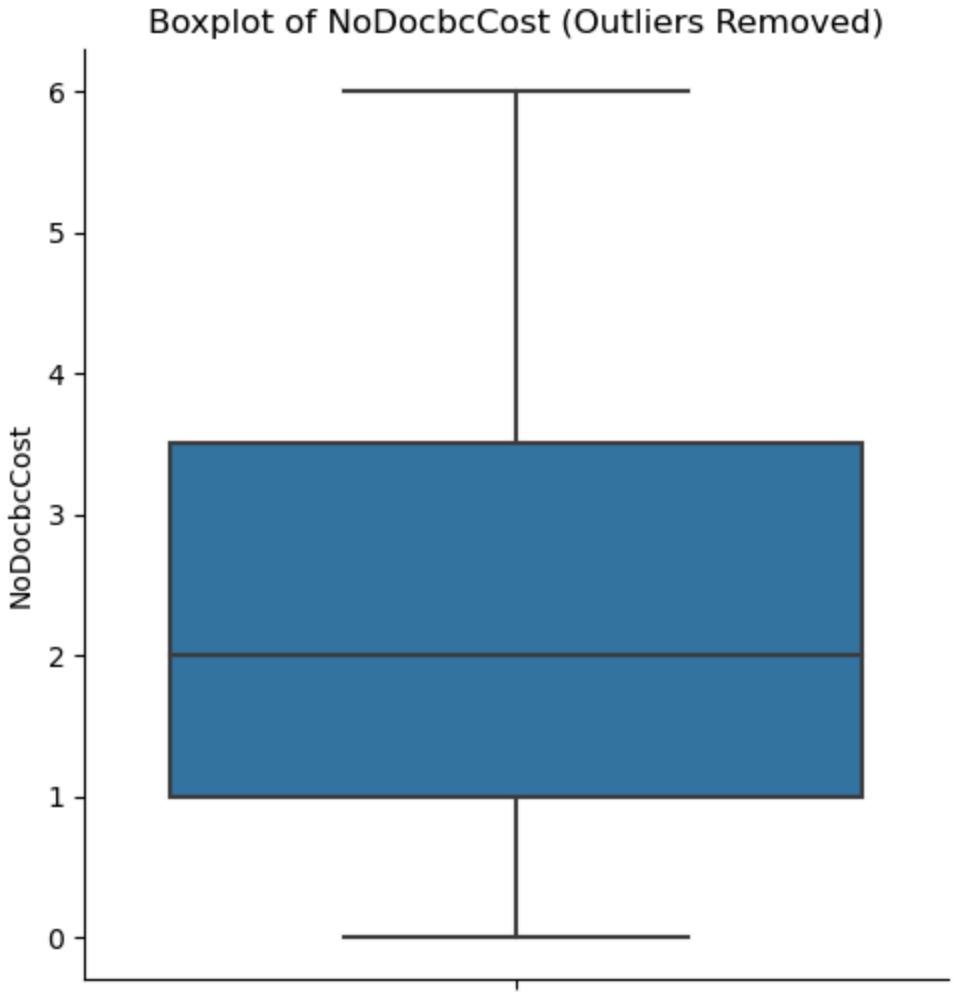

```
C:\Users\rutik\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```




```
C:\Users\rutik\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```




```
C:\Users\rutik\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```




```
Outliers have been removed from all attributes.
```

Standardization and Normalization

To perform standardization, normalization, and string processing for the numerical attributes (BMI, Income, Education, Age, GenHlth, MentHlth, PhysHlth) in the dataset, we'll proceed with the following steps. Since string processing is not applicable to these numerical attributes, we'll focus on standardization and normalization only.

In [34]:

```
numerical_columns = ['BMI', 'Income', 'Education', 'Age', 'GenHlth', 'MentHlth

# Initialize StandardScaler
scaler_standard = StandardScaler()

# Apply StandardScaler to the numerical columns
data_standardized = data.copy()
data_standardized[numerical_columns] = scaler_standard.fit_transform(data[nume

# Print the first few rows of the standardized dataset for verification
print("Standardized Data:")
print(data_standardized.head())

# Initialize MinMaxScaler
scaler_minmax = MinMaxScaler()

# Apply MinMaxScaler to the numerical columns
data_normalized = data.copy()
data_normalized[numerical_columns] = scaler_minmax.fit_transform(data[numerica

# Print the first few rows of the normalized dataset for verification
print("\nNormalized Data:")
print(data_normalized.head())
```

```
Standardized Data:
   HeartDiseaseorAttack  HighBP  HighChol  CholCheck       BMI  Smoker  \
0                     0       1         1          1  1.757936       1
1                     0       0         0          0 -0.511806       1
2                     0       1         1          1 -0.057858       0
3                     0       1         0          1 -0.209174       0
4                     0       1         1          1 -0.663122       0

   Stroke  Diabetes  PhysActivity  Fruits  ...  AnyHealthcare  NoDocbcCost  \
0       0         0             0     0.0  ...            1.0            0
1       0         0             1     0.0  ...            0.0            1
2       0         0             0     1.0  ...            1.0            1
3       0         0             1     1.0  ...            1.0            0
4       0         0             1     1.0  ...            1.0            0

    GenHlth  MentHlth  PhysHlth  DiffWalk  Sex       Age  Education    Income
0  2.329121  1.998592  1.233999         1    0  0.316900  -1.065595 -1.474487
1  0.457294 -0.429630 -0.486592         0    0 -0.337933   0.963272 -2.440138
2  2.329121  3.617407  2.954590         1    0  0.316900  -1.065595  0.939638
3 -0.478619 -0.429630 -0.486592         0    0  0.971733  -2.080028 -0.026012
4 -0.478619 -0.024926 -0.486592         0    0  0.971733  -0.051162 -0.991662

[5 rows x 22 columns]

Normalized Data:
   HeartDiseaseorAttack  HighBP  HighChol  CholCheck       BMI  Smoker  \
0                     0       1         1          1  0.325581       1
1                     0       0         0          0  0.151163       1
2                     0       1         1          1  0.186047       0
3                     0       1         0          1  0.174419       0
4                     0       1         1          1  0.139535       0

   Stroke  Diabetes  PhysActivity  Fruits  ...  AnyHealthcare  NoDocbcCost  \
0       0         0             0     0.0  ...            1.0            0
1       0         0             1     0.0  ...            0.0            1
2       0         0             0     1.0  ...            1.0            1
3       0         0             1     1.0  ...            1.0            0
4       0         0             1     1.0  ...            1.0            0

   GenHlth  MentHlth  PhysHlth  DiffWalk  Sex       Age  Education    Income
0     1.00       0.6       0.5         1    0  0.666667        0.6  0.285714
1     0.50       0.0       0.0         0    0  0.500000        1.0  0.000000
2     1.00       1.0       1.0         1    0  0.666667        0.6  1.000000
3     0.25       0.0       0.0         0    0  0.833333        0.4  0.714286
4     0.25       0.1       0.0         0    0  0.833333        0.8  0.428571

[5 rows x 22 columns]
```

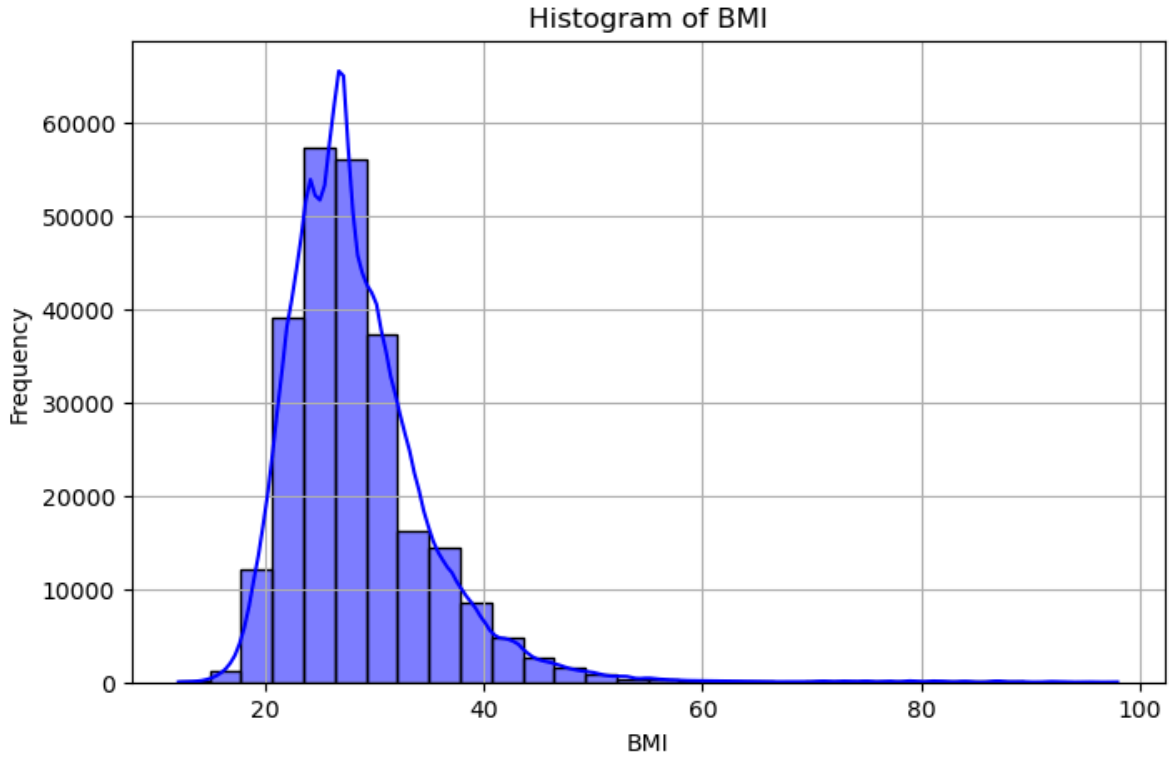
DATA VISUALISATION

UNIVARIATE ANALYSIS

Univariate analysis involves the examination and description of a single variable within a dataset. It is a fundamental aspect of data analysis and helps to understand the basic characteristics of a dataset.

1. What is the distribution of BMI among the population?

In [10]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
# Plot histogram for BMI
plt.figure(figsize=(8, 5))
sns.histplot(data['BMI'], bins=30, kde=True, color='blue')
plt.title('Histogram of BMI')
plt.xlabel('BMI')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()

# Summary statistics for BMI
print(data['BMI'].describe())
```

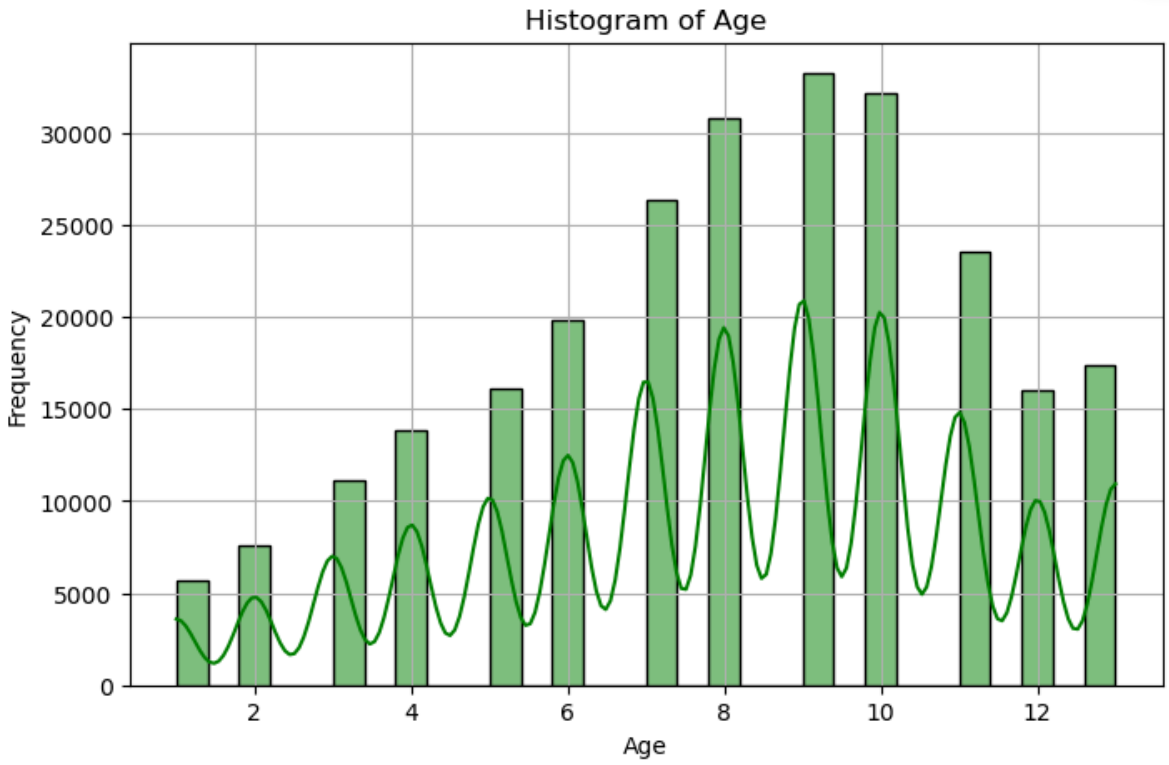



```
count    253680.000000
mean         28.382364
std           6.608694
min          12.000000
25%          24.000000
50%          27.000000
75%          31.000000
max          98.000000
Name: BMI, dtype: float64
```

BMI values appear to be concentrated around 25-30, suggesting a prevalence of overweight individuals.

2. What is the age distribution in the dataset?

In [11]:
```
# Plot histogram for Age
plt.figure(figsize=(8, 5))
sns.histplot(data['Age'], bins=30, kde=True, color='green')
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()

# Summary statistics for Age
print(data['Age'].describe())
```

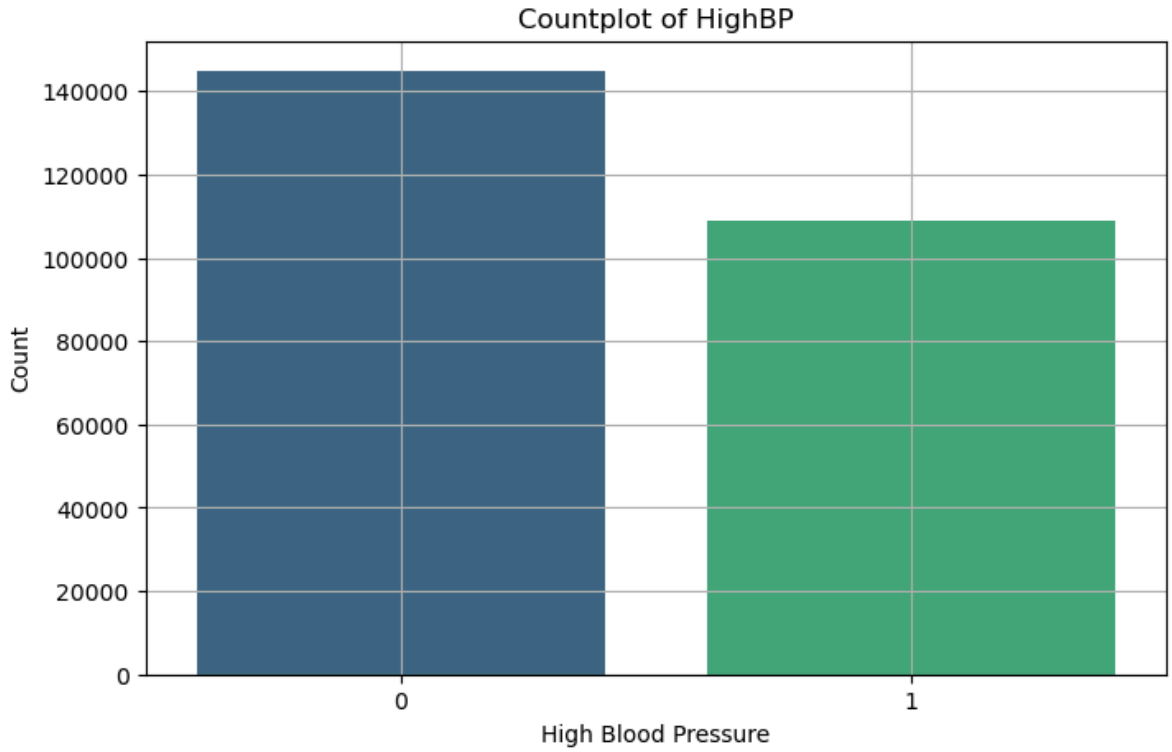



```
count    253680.000000
mean          8.032119
std           3.054220
min           1.000000
25%           6.000000
50%           8.000000
75%          10.000000
max          13.000000
Name: Age, dtype: float64
```

The majority of individuals seem to be between 1 to 9 years old.

3. How many individuals have high blood pressure (HighBP)?

In [12]:
```
# Plot count plot for HighBP
plt.figure(figsize=(8, 5))
sns.countplot(data=data, x='HighBP', palette='viridis')
plt.title('Countplot of HighBP')
plt.xlabel('High Blood Pressure')
plt.ylabel('Count')
plt.grid(True)
plt.show()

# Calculate percentages for HighBP
high_bp_percentage = data['HighBP'].value_counts(normalize=True) * 100
print(f"Percentage with High Blood Pressure:\n{high_bp_percentage}")
```

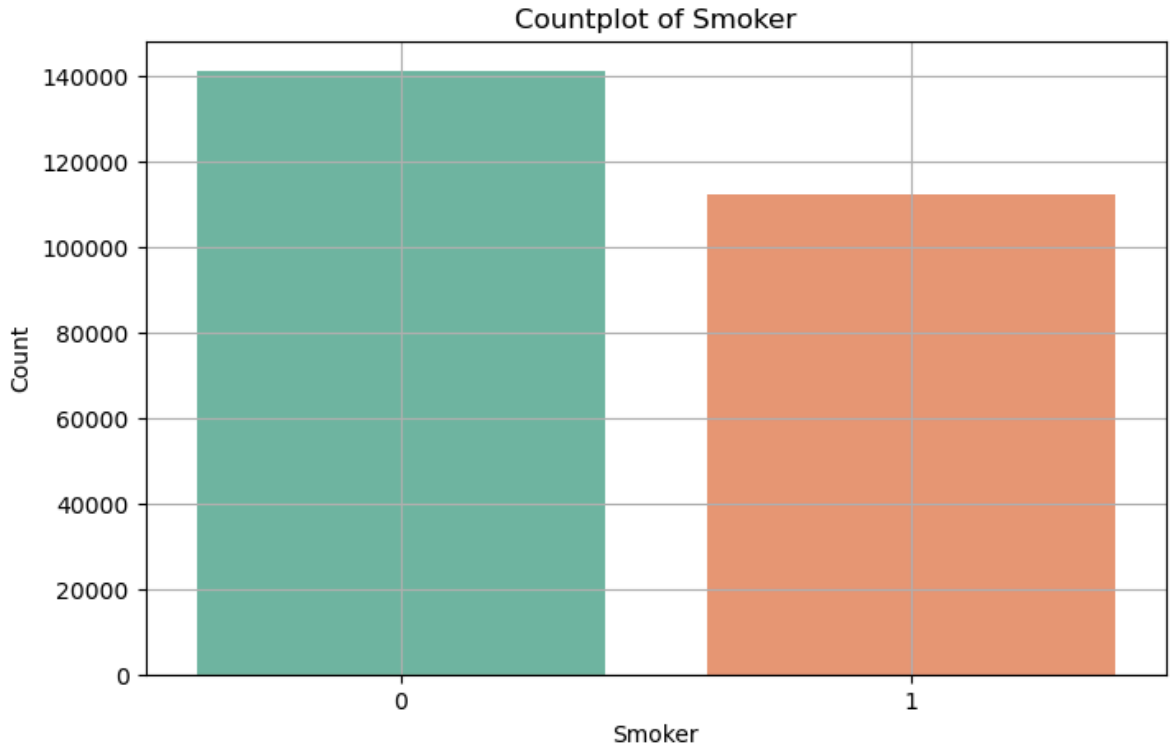



```
Percentage with High Blood Pressure:
HighBP
0    57.09989
1    42.90011
Name: proportion, dtype: float64
```

A substantial portion of the population seems to have high blood pressure.

4. How many individuals are smokers?

In [13]:
```
# Plot count plot for Smoker
plt.figure(figsize=(8, 5))
sns.countplot(data=data, x='Smoker', palette='Set2')
plt.title('Countplot of Smoker')
plt.xlabel('Smoker')
plt.ylabel('Count')
plt.grid(True)
plt.show()

# Calculate percentages for Smoker
smoker_percentage = data['Smoker'].value_counts(normalize=True) * 100
print(f"Percentage of Smokers:\n{smoker_percentage}")
```




```
Percentage of Smokers:
Smoker
0    55.683144
1    44.316856
Name: proportion, dtype: float64
```

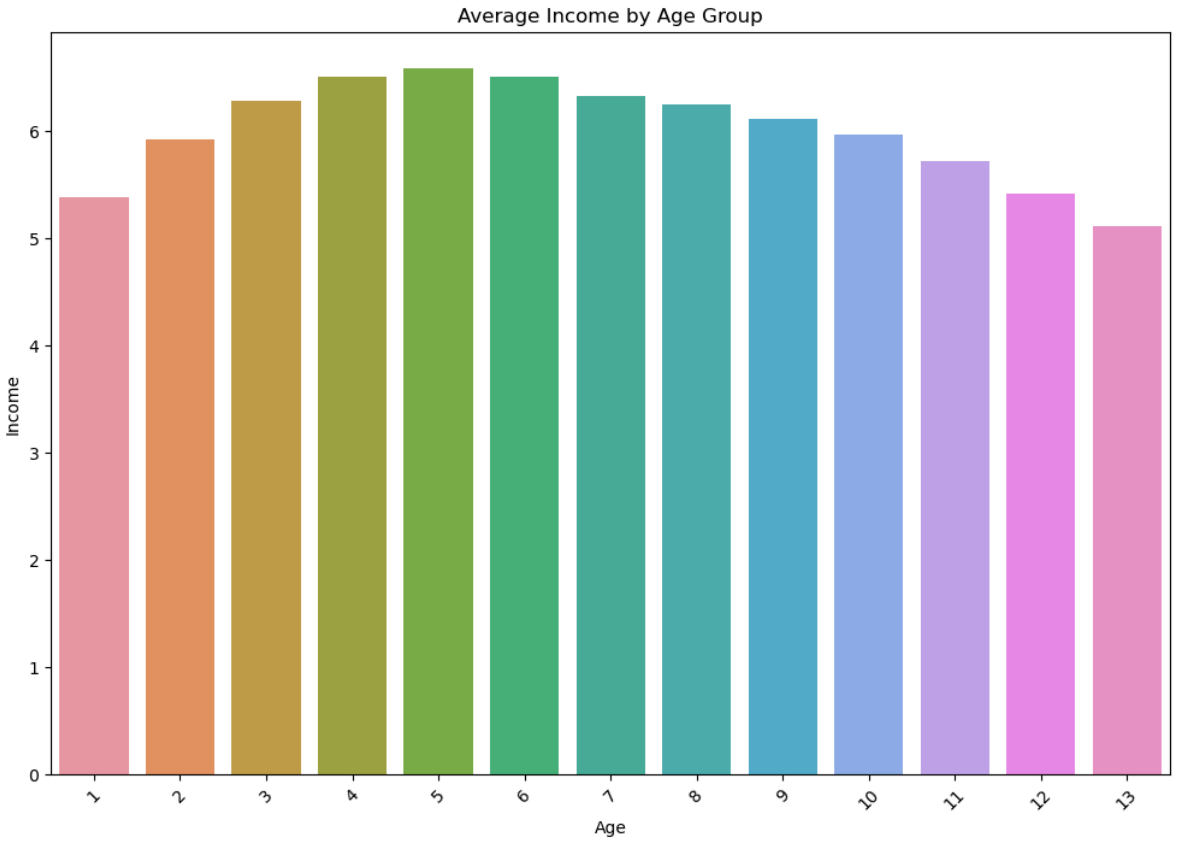
The dataset suggests a minority of individuals are smokers.

BIVARIATE ANALYSIS

Bivariate analysis examines the relationship between two variables. It helps to identify patterns, correlations, and causal relationships between variables.

In [16]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 8))
sns.barplot(x='Age', y='Income', data=data, ci=None)
plt.title('Average Income by Age Group')
plt.xlabel('Age')
plt.ylabel('Income')
plt.xticks(rotation=45)
plt.show()
```

```
C:\Users\rutik\AppData\Local\Temp\ipykernel_26360\3471869384.py:3: FutureWa
ing:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='Age', y='Income', data=data, ci=None)
```



MULTIVARIATE ANALYSIS

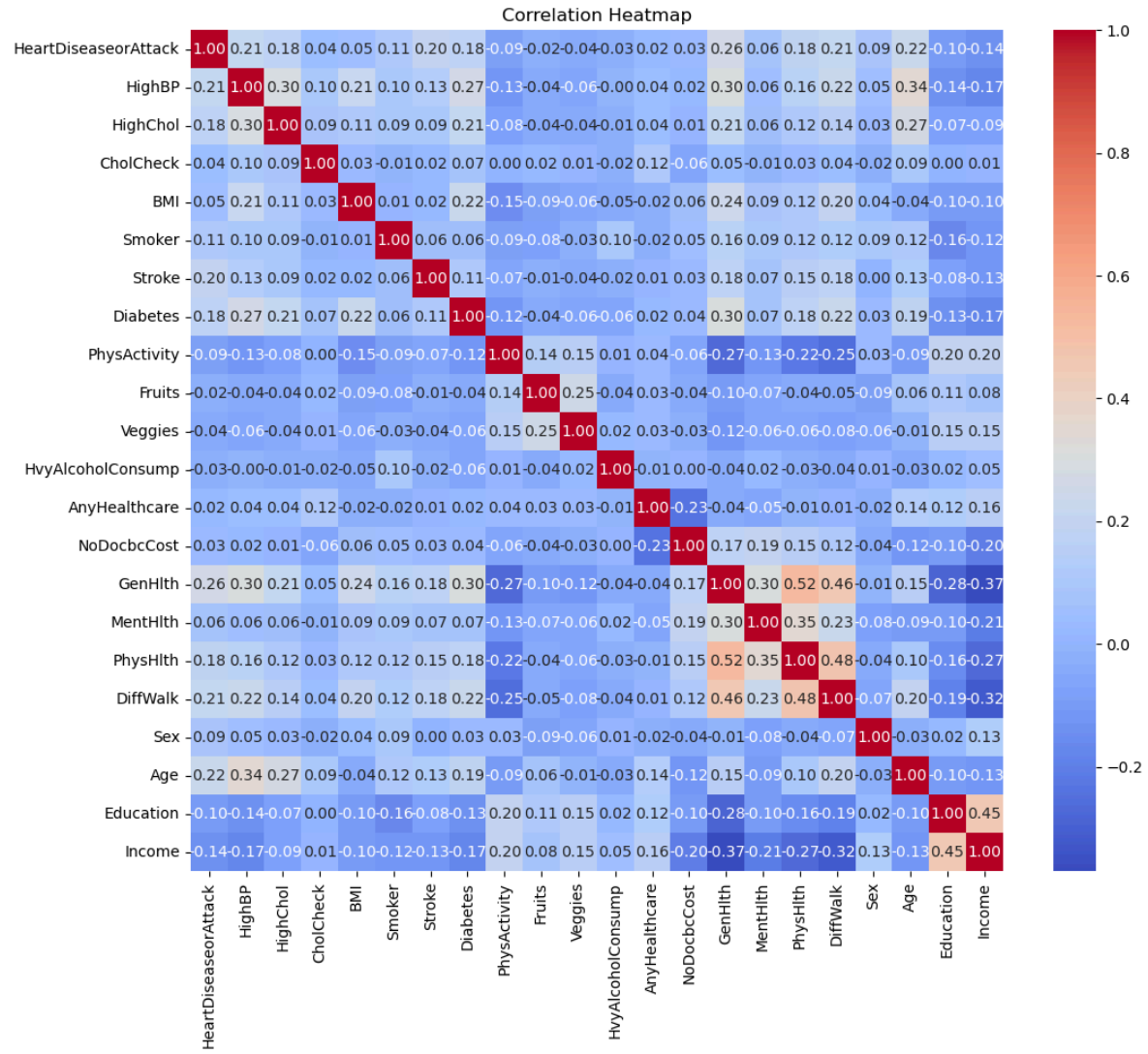
Multivariate analysis involves examining more than two variables simultaneously to understand relationships and patterns among them.

Correlation and Covariance

In [21]:
```
# Select only numeric columns
numeric_cols = data.select_dtypes (include=['float64', 'int64'])

# Calculate correlation between numeric features
correlation_matrix = numeric_cols.corr()

# Calculate covariance between numeric features
covariance_matrix = numeric_cols.cov()

# Display correlation matrix
print("Correlation matrix:")
print(correlation_matrix)

# Display covariance matrix
print("\nCovariance matrix:")
print(covariance_matrix)
```

```
Covariance matrix:
                      HeartDiseaseorAttack    HighBP  HighChol  CholCheck
\
HeartDiseaseorAttack              0.085315  0.030266  0.026094   0.002448
HighBP                            0.030266  0.244960  0.072940   0.009243
HighChol                          0.026094  0.072940  0.244243   0.008024
CholCheck                         0.002448  0.009243  0.008024   0.035937
BMI                               0.102122  0.699142  0.348563   0.043216
Smoker                            0.016605  0.023847  0.022414  -0.000935
Stroke                            0.011698  0.012653  0.009031   0.000904
Diabetes                          0.036762  0.093848  0.072142   0.008940
PhysActivity                     -0.010943 -0.026608 -0.016554   0.000341
Fruits                           -0.002779 -0.009661 -0.009701   0.002173
Veggies                          -0.004475 -0.011862 -0.007708   0.000454
HvyAlcoholConsump                -0.001950 -0.000453 -0.001314  -0.001036
AnyHealthcare                     0.001188  0.004107  0.004508   0.004809
NoDocbcCost                       0.002514  0.002385  0.001826  -0.003066
GenHlth                           0.080639  0.158928  0.110060   0.009437
MentHlth                          0.139918  0.207130  0.227390  -0.011756
```

In [19]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming df is your DataFrame with the transformed data

# Calculate the correlation matrix
correlation_matrix = data.corr()

# Plot the heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```



Here we can infer that "PhysHlth" has strong positive correlation with "GenHlth" whereas it has strong negative correlation with "income". "AnyHlthCare" and "NoDocBcCost" has almost no correlation with "HeartDiseaseorAttack" Since correlation is zero we can infer there is no linear relationship between these two predictors.However it is safe to drop these features in case you're applying Linear Regression model to the dataset.

The heatmap visualizes the correlation between different features in the dataset. Each cell in the heatmap represents the correlation coefficient between two features.

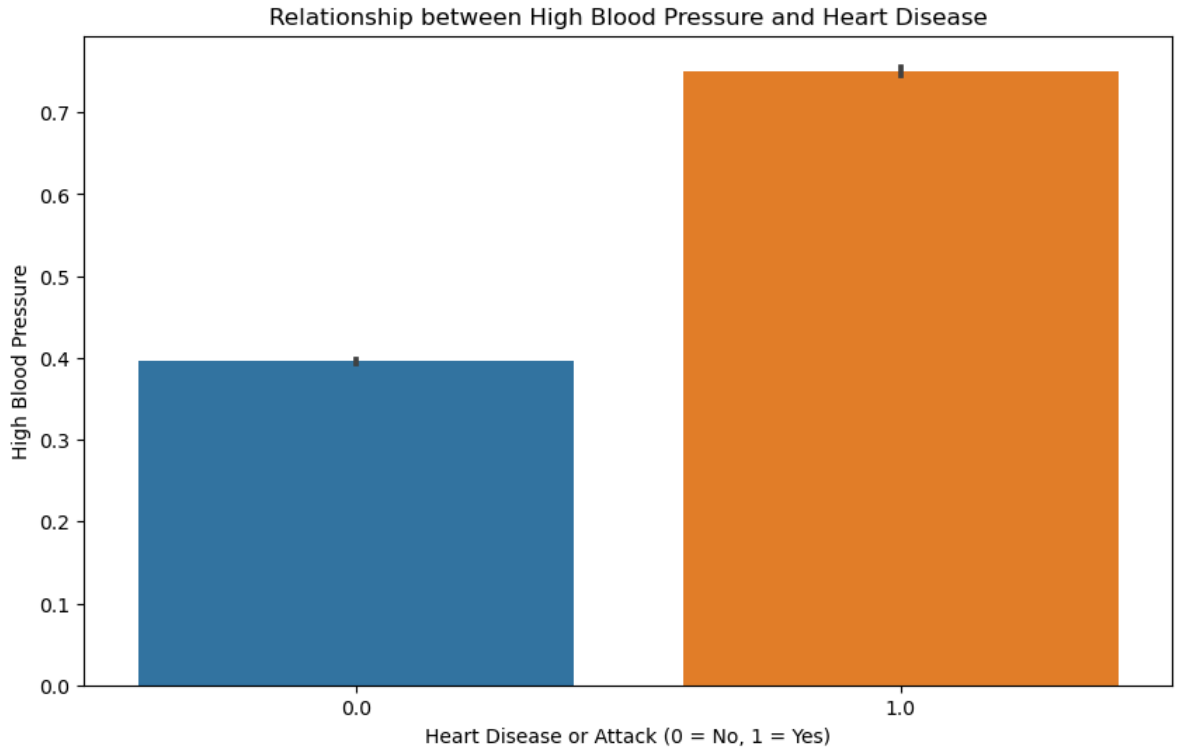
Positive correlations are indicated by lighter colors (closer to red), while negative correlations are indicated by darker colors (closer to blue).

Values close to 1 or -1 indicate strong correlations, while values close to 0 indicate weak correlations or no correlation.

The annotations inside the cells provide the exact correlation values, helping in quantitative

Hypothesis 1: People with High Blood Pressure are more likely to have Heart Disease

In [11]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x='HeartDiseaseorAttack', y='HighBP', data=data)
plt.title('Relationship between High Blood Pressure and Heart Disease')
plt.xlabel('Heart Disease or Attack (0 = No, 1 = Yes)')
plt.ylabel('High Blood Pressure')
plt.show()
```



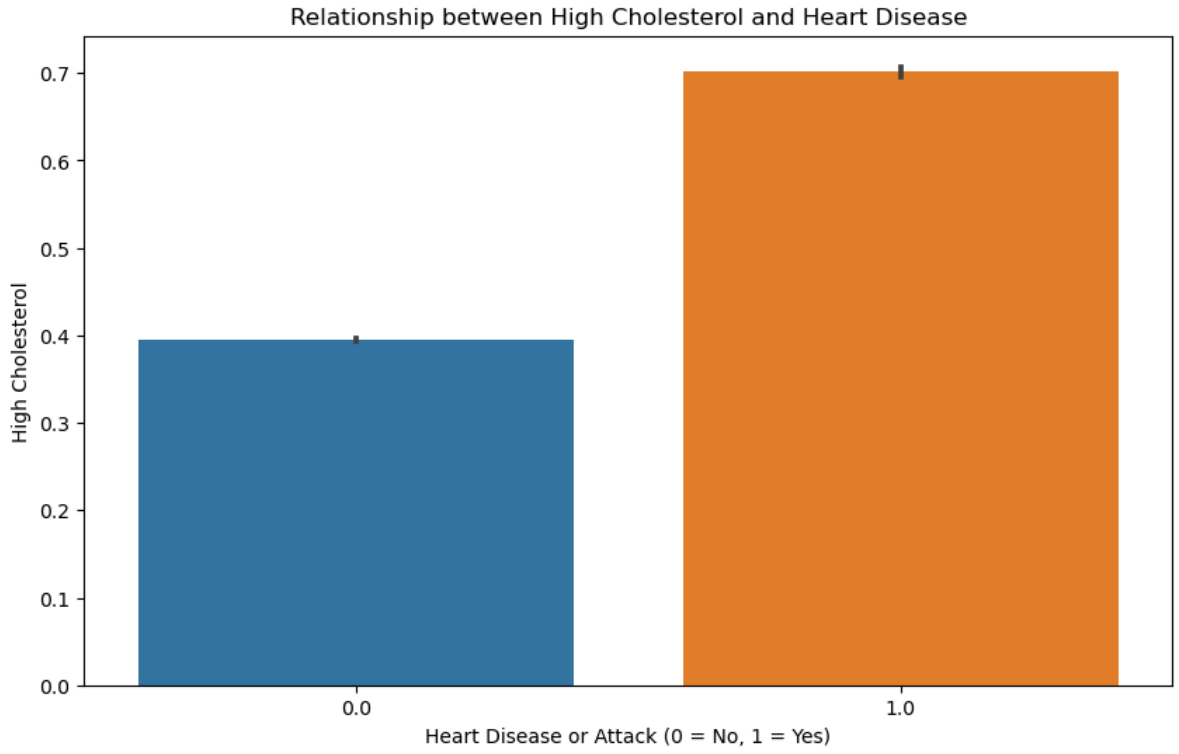
HighBP: A binary feature indicating if an individual has high blood pressure (1) or not (0).

HeartDiseaseorAttack: The target variable indicating if an individual has heart disease or has had a heart attack (1) or not (0).

This plot shows the average occurrence of high blood pressure in individuals with and without heart disease. If individuals with heart disease have a significantly higher average of high blood pressure, it supports the hypothesis.

Hypothesis 2: Higher cholesterol levels (HighChol) are associated with a higher risk of heart disease or a heart attack

In [13]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x='HeartDiseaseorAttack', y='HighChol', data=data)
plt.title('Relationship between High Cholesterol and Heart Disease')
plt.xlabel('Heart Disease or Attack (0 = No, 1 = Yes)')
plt.ylabel('High Cholesterol')
plt.show()
```



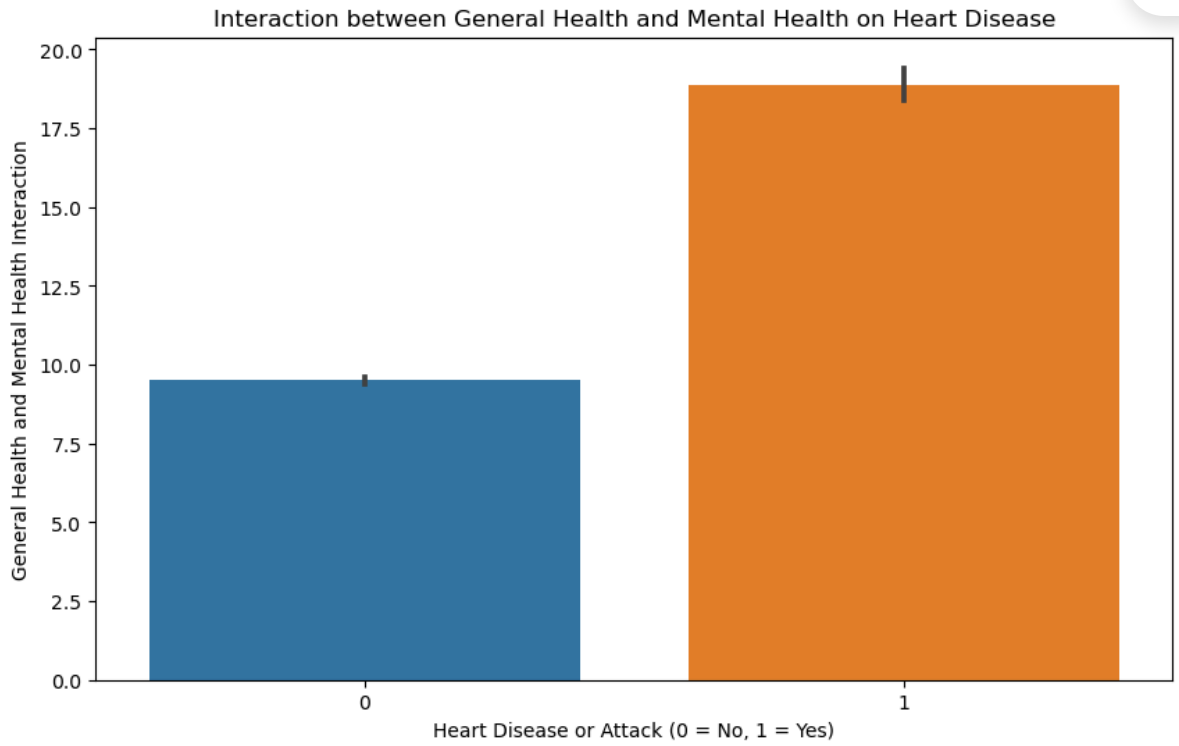
HighChol: A binary feature indicating if an individual has high cholesterol (1) or not (0).

HeartDiseaseorAttack: The target variable indicating if an individual has heart disease or has had a heart attack (1) or not (0).

This plot demonstrates the average occurrence of high cholesterol in individuals with and without heart disease. A higher average for individuals with heart disease would support the hypothesis.

Hypothesis 4: Interaction between General Health (GenHlth), Mental Health (MentHlth), and Heart Disease

In [11]:
```
# Create a new feature combining GenHlth and MentHlth
data['GenHlth_MentHlth'] = data['GenHlth'] * data['MentHlth']

plt.figure(figsize=(10, 6))
sns.barplot(x='HeartDiseaseorAttack', y='GenHlth_MentHlth', data=data)
plt.title('Interaction between General Health and Mental Health on Heart Disea
plt.xlabel('Heart Disease or Attack (0 = No, 1 = Yes)')
plt.ylabel('General Health and Mental Health Interaction')
plt.show()
```



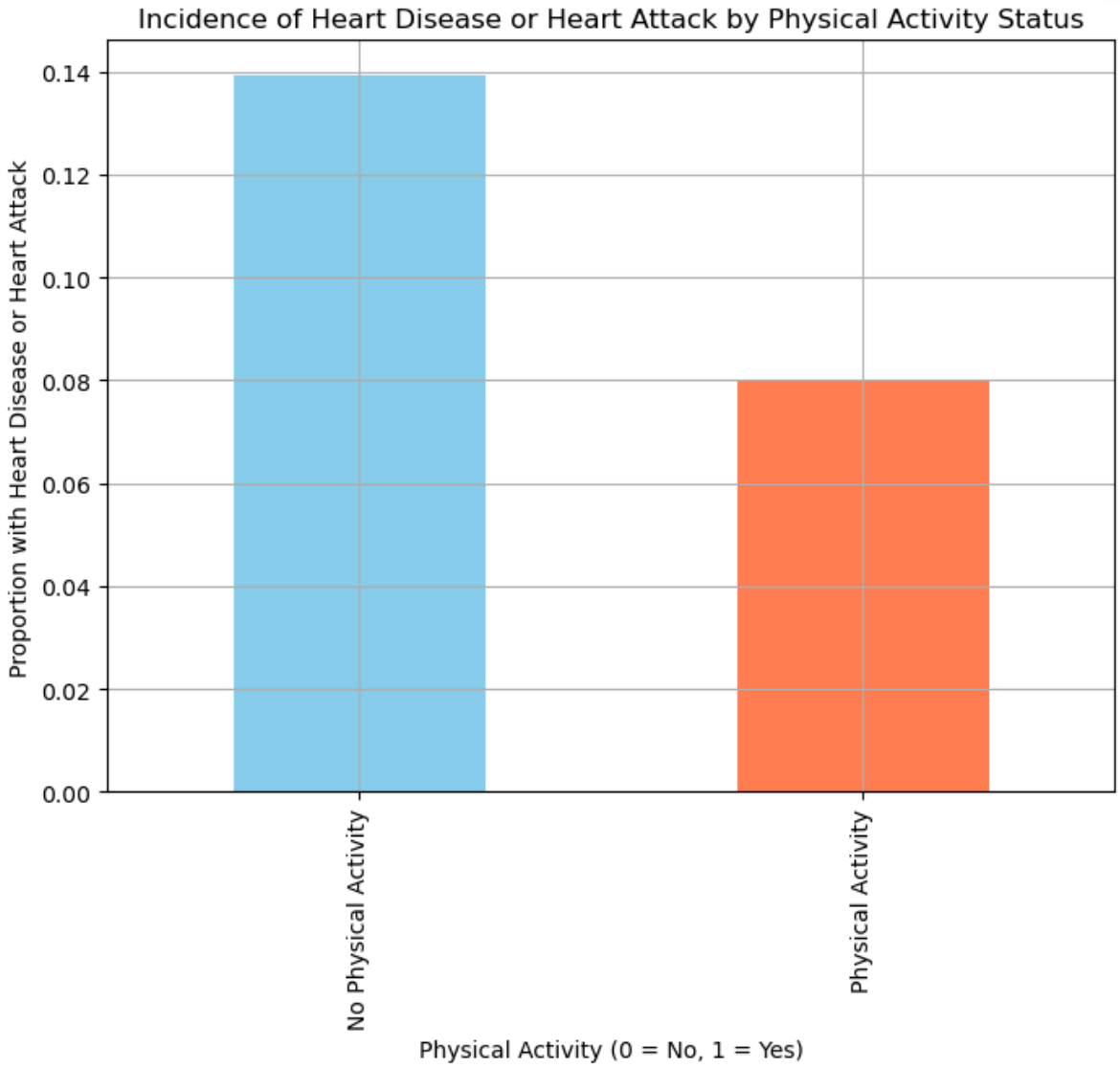
GenHlth_MentHlth: A combined feature representing the interaction between general health and mental health. Higher values indicate worse combined health status.

HeartDiseaseorAttack: The target variable indicating if an individual has heart disease or has had a heart attack (1) or not (0).

This plot shows the relationship between the combined effect of general and mental health and heart disease, illustrating the compounded risk of poor overall health.

HYPOTHESIS 5: Is there a difference in the incidence of heart disease or heart attack between individuals who engage in regular physical activity and those who don't?

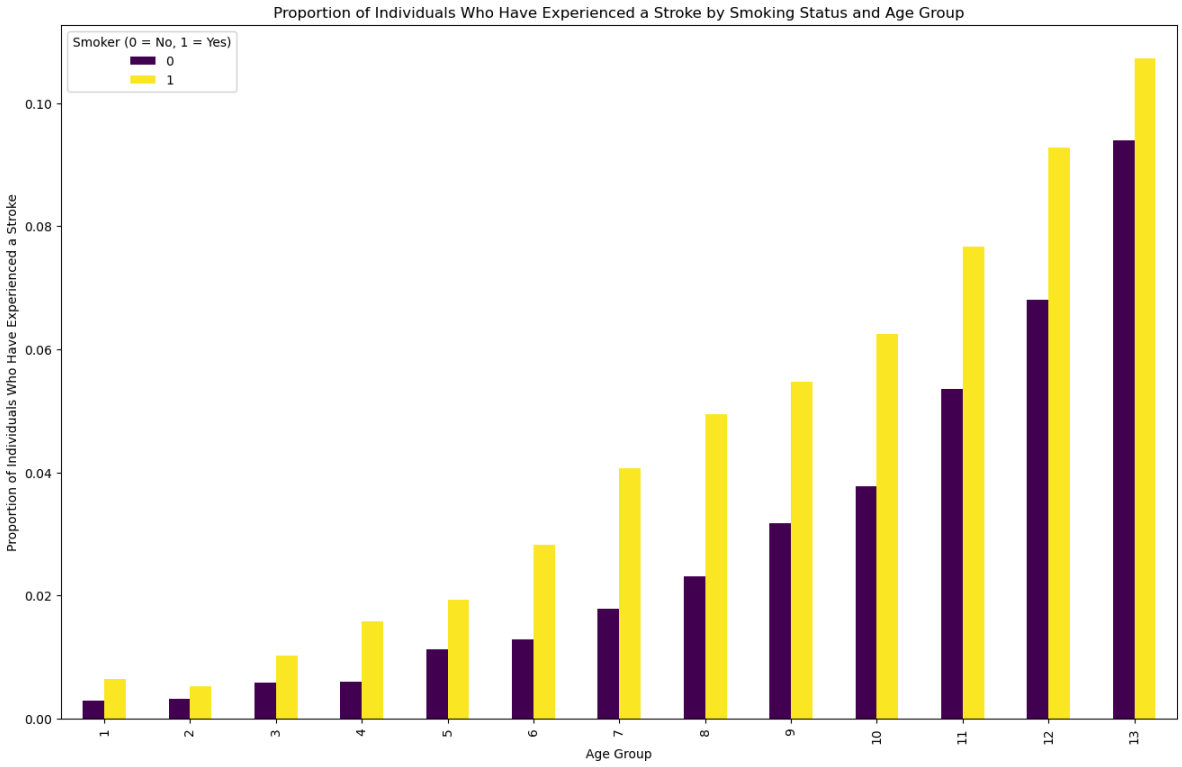
In [7]:
```
heart_disease_activity = data.groupby('PhysActivity')['HeartDiseaseorAttack'].

# Plot the results
plt.figure(figsize=(8, 6))
heart_disease_activity.plot(kind='bar', color=['skyblue', 'coral'])
plt.title('Incidence of Heart Disease or Heart Attack by Physical Activity Sta
plt.xlabel('Physical Activity (0 = No, 1 = Yes)')
plt.ylabel('Proportion with Heart Disease or Heart Attack')
plt.xticks([0, 1], ['No Physical Activity', 'Physical Activity'])
plt.grid(True)
plt.show()
```



=> The analysis indicates that individuals who engage in regular physical activity have a lower incidence of heart disease or heart attack compared to those who do not. This supports the hypothesis that regular physical activity is associated with a reduced risk of heart disease or heart attack. Regular physical activity may contribute to better cardiovascular health and reduce the risk of heart-related conditions.

HYPOTHESIS 6:Are smokers more likely to have experienced a stroke, and does this relationship vary by age group?

In [32]:
```
df = pd.DataFrame(data)

# Calculate the proportion of individuals who have experienced a stroke fo
stroke_proportions = df.groupby(['Age', 'Smoker'])['Stroke'].mean().unstac

# Plot the proportions
stroke_proportions.plot(kind='bar', figsize=(16, 10), colormap='viridis')
plt.title('Proportion of Individuals Who Have Experienced a Stroke by Smoking
plt.xlabel('Age Group')
plt.ylabel('Proportion of Individuals Who Have Experienced a Stroke')
plt.legend(title='Smoker (0 = No, 1 = Yes)')

plt.show()
```

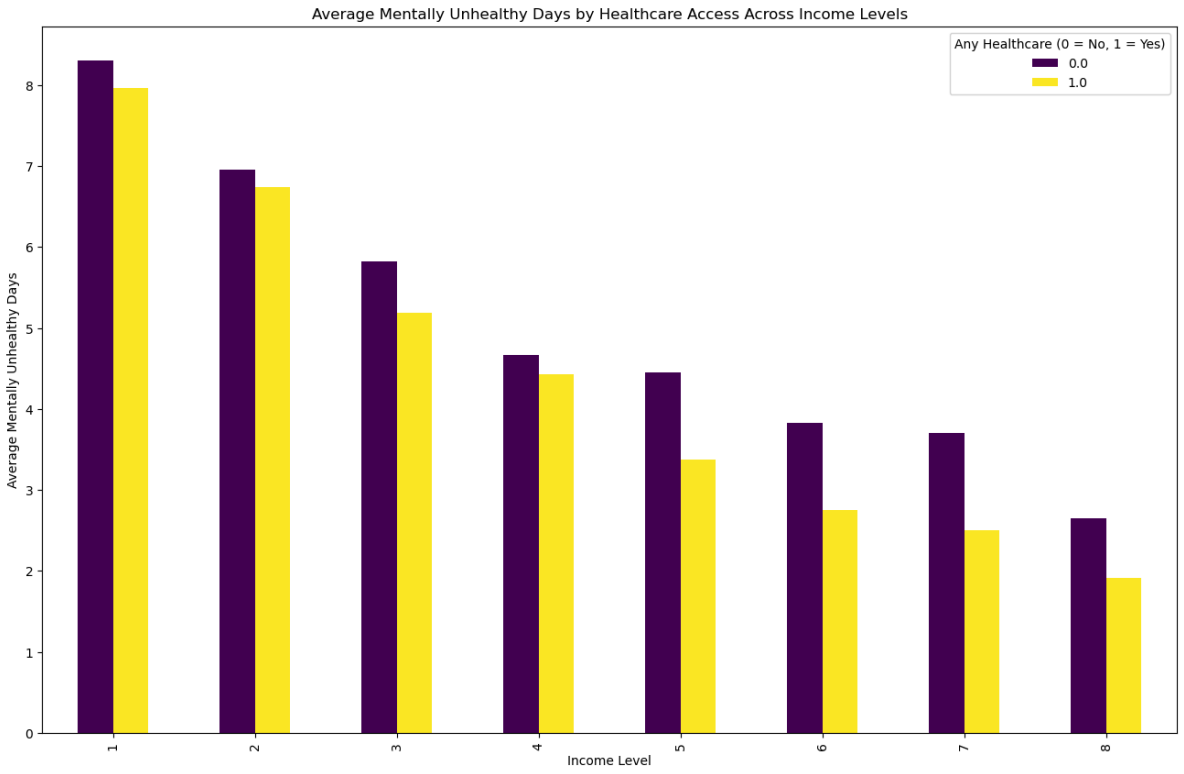


Age Groups 1-4 (Younger Age Groups): In these age groups, the proportion of individuals who have experienced a stroke is relatively low for both smokers and non-smokers. However, smokers still show a slightly higher likelihood of having had a stroke.

Age Groups 5-8 (Middle Age Groups): The difference in stroke prevalence between smokers and non-smokers becomes more apparent. Smokers in these age groups have a noticeably higher proportion of strokes compared to non-smokers.

HYPOTHESIS 8: How the average number of mentally unhealthy days by income level and healthcare coverage using a DataFrame?

In [10]:
```
mental_health_income_healthcare = df.groupby(['Income', 'AnyHealthcare'])['

# Plot the results
mental_health_income_healthcare.plot(kind='bar', figsize=(16, 10), colormap='v
plt.title('Average Mentally Unhealthy Days by Healthcare Access Across Income
plt.xlabel('Income Level')
plt.ylabel('Average Mentally Unhealthy Days')
plt.legend(title='Any Healthcare (0 = No, 1 = Yes)')
plt.show()
```

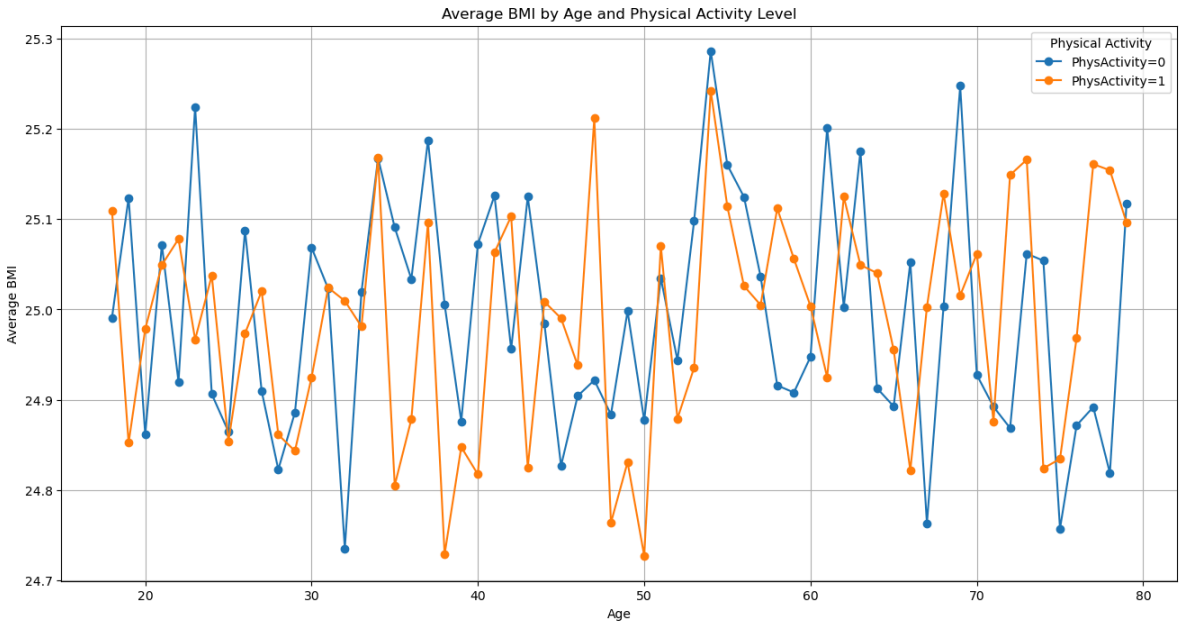


The visualization of average mentally unhealthy days by income level and healthcare coverage reveals significant insights into mental health disparities. Generally, individuals with higher income levels experience fewer mentally unhealthy days, highlighting the positive correlation between higher income and better mental health. Additionally, having healthcare coverage is associated with fewer mentally unhealthy days across almost all income levels, underscoring the critical role of healthcare access in improving mental health outcomes. This suggests that policies aimed at increasing income and expanding healthcare coverage could be effective in reducing mental health issues.

HYPOTHESIS 9: How does the average body mass index (BMI) vary with age, stratified by physical activity level?

In [21]:

```
df = pd.DataFrame(data)

# Calculate the average BMI for each age group and physical activity level
avg_bmi_physact_age = df.groupby(['Age', 'PhysActivity'])['BMI'].mean().unstac

# Plot the results using a line plot
plt.figure(figsize=(16, 8))
for physact in [0, 1]:  # PhysActivity levels (0 = No, 1 = Yes)
    plt.plot(avg_bmi_physact_age.index, avg_bmi_physact_age[physact], marker='
plt.title('Average BMI by Age and Physical Activity Level')
plt.xlabel('Age')
plt.ylabel('Average BMI')
plt.legend(title='Physical Activity')
plt.grid(True)
plt.show()
```



Interpretation:

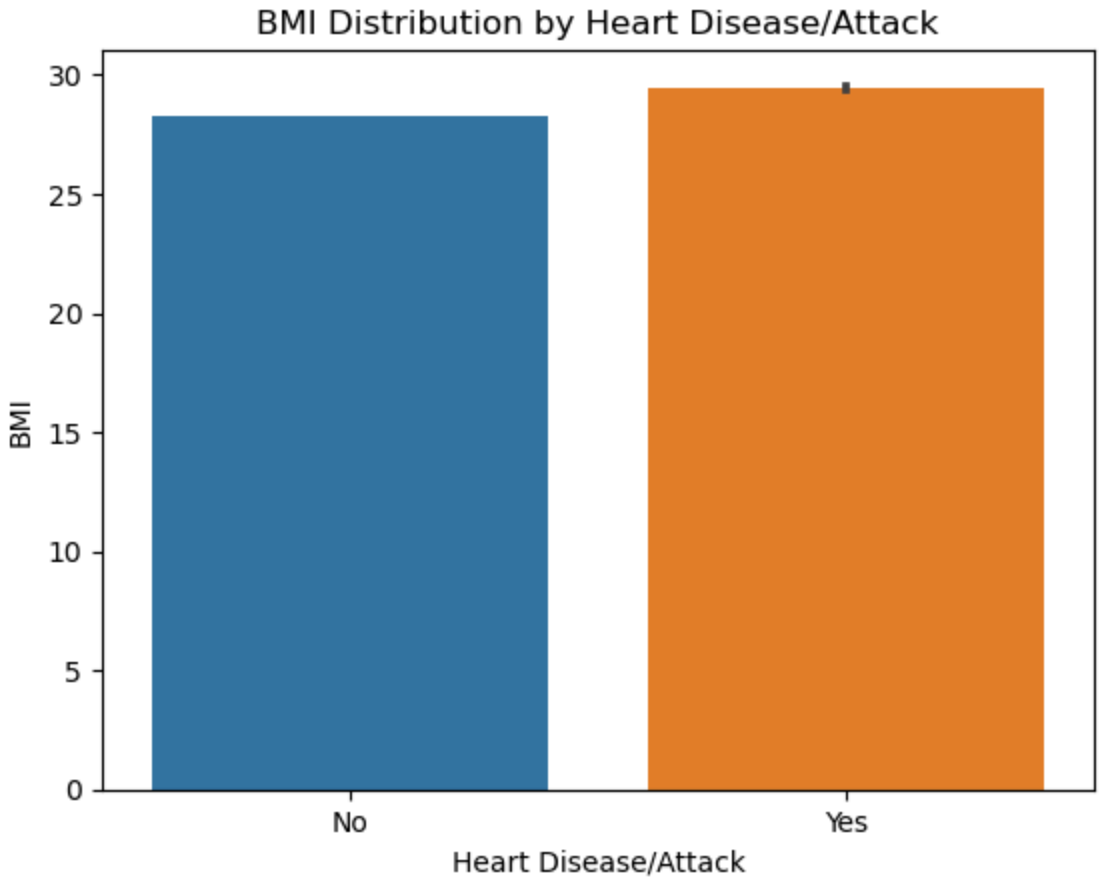
Trend by Age: The plot allows us to observe how average BMI changes across different age groups. Impact of Physical Activity: Comparing the lines for different physical activity levels shows whether and how physical activity influences BMI at different ages. Insights into Health Patterns: Higher BMI averages may indicate potential health risks associated with age and physical activity levels.

Implications:

Understanding these patterns can inform health interventions and policies aimed at promoting physical activity to manage BMI and potentially reduce health risks. Monitoring BMI across age groups helps in identifying trends and patterns that may require targeted health interventions or

Is there a correlation between body mass index (BMI) and the occurrence of heart disease or heart attack?

In [24]:
```
# Create a box plot to compare BMI distribution between individuals with and w
sns.barplot(x='HeartDiseaseorAttack', y='BMI', data=data)
plt.xlabel('Heart Disease/Attack')
plt.ylabel('BMI')
plt.title('BMI Distribution by Heart Disease/Attack')
plt.xticks([0, 1], ['No', 'Yes'])
plt.show()
```

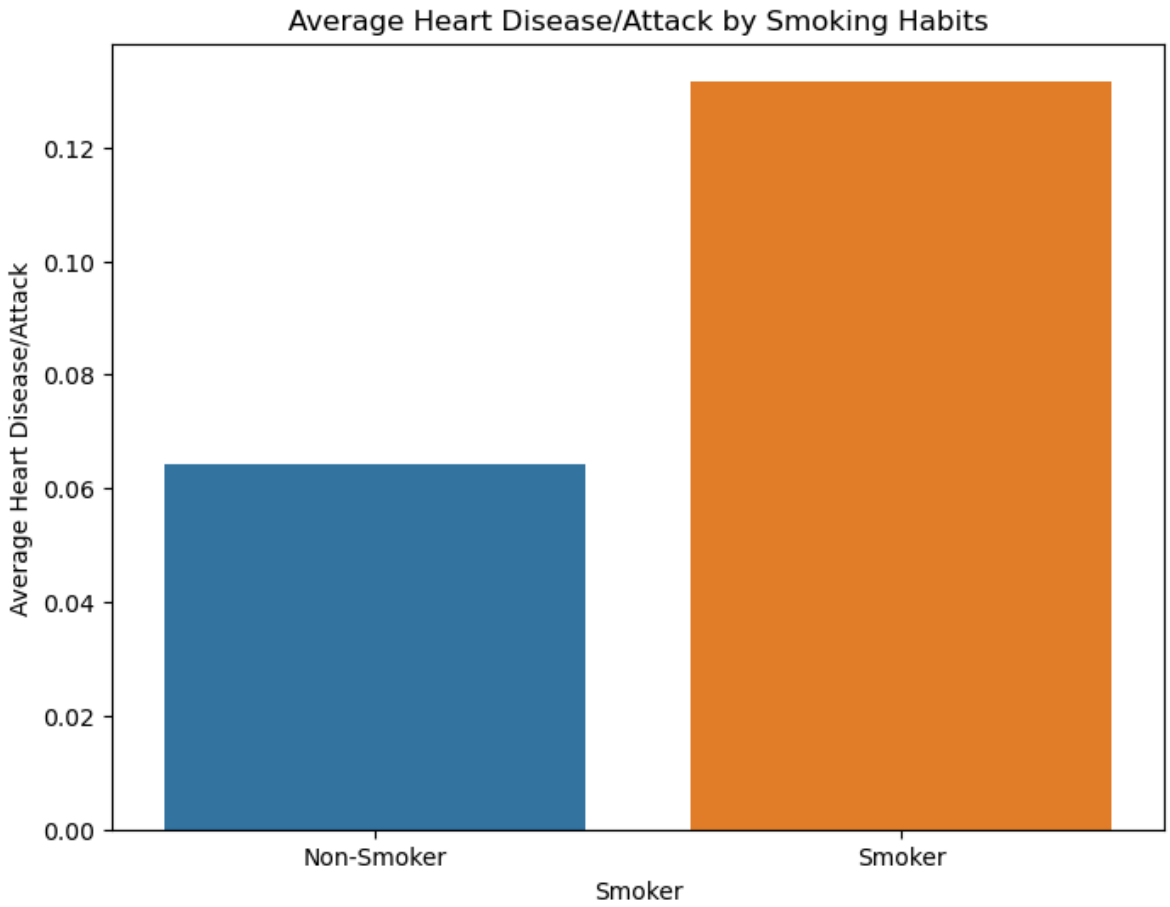


The resulting bar plot will display the distribution of BMI for individuals with and without heart disease or heart attack. Observing any differences in the distributions can give insights into the correlation between BMI and the occurrence of heart disease or heart attack. If there's a notable difference, it may suggest a correlation between higher BMI and a higher risk of heart disease or heart attack.

Is there a relationship between smoking habits and the likelihood of having heart disease or a heart attack?

In [25]:
```
# Grouping data by smoking habits and calculating the average HeartDiseaseorAt
smoking_and_heart_disease = data.groupby('Smoker')['HeartDiseaseorAttack'].mea

# Print the average HeartDiseaseorAttack for smokers and non-smokers
print("Average Heart Disease/Attack for Smokers and Non-Smokers:")
print(smoking_and_heart_disease)

# Visualization
plt.figure(figsize=(8, 6))
sns.barplot(x=smoking_and_heart_disease.index, y=smoking_and_heart_disease.val
plt.title('Average Heart Disease/Attack by Smoking Habits')
plt.xlabel('Smoker')
plt.ylabel('Average Heart Disease/Attack')
plt.xticks([0, 1], ['Non-Smoker', 'Smoker'])
plt.show()
```

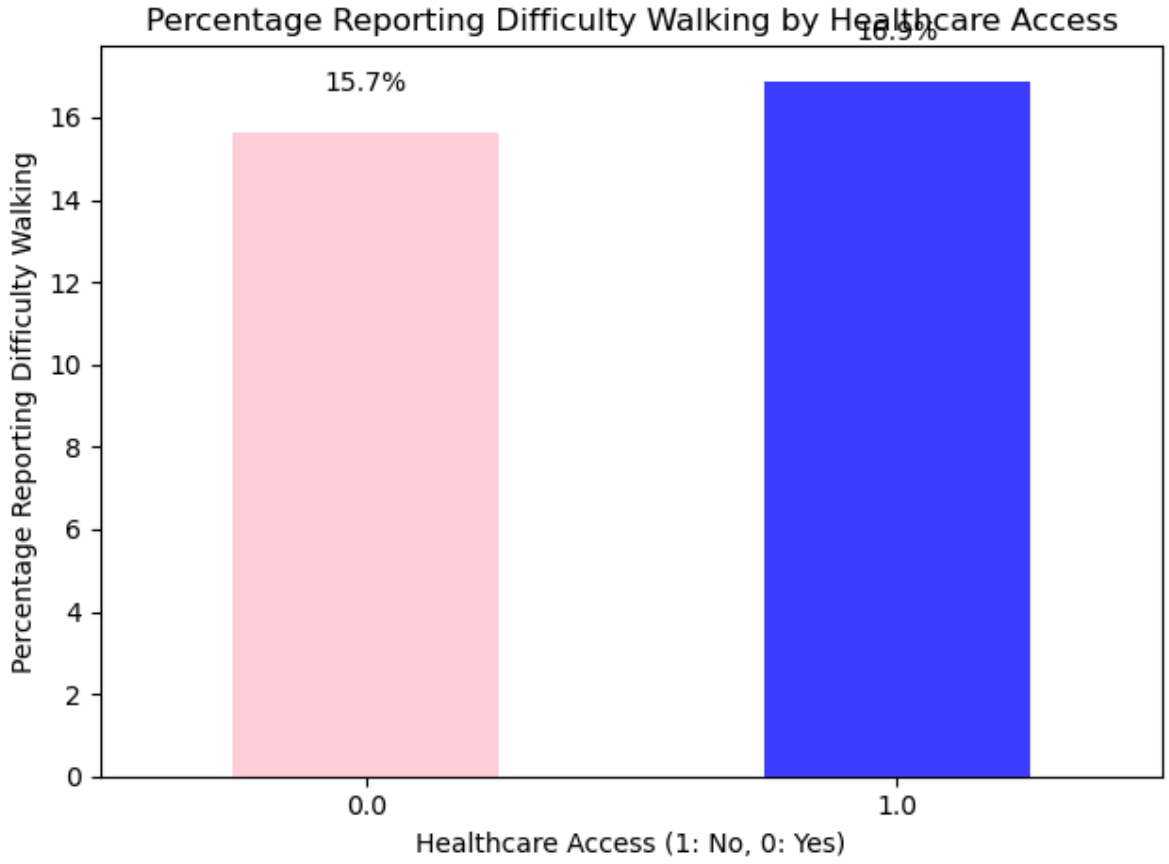
```
Average Heart Disease/Attack for Smokers and Non-Smokers:
Smoker
0    0.064365
1    0.131655
Name: HeartDiseaseorAttack, dtype: float64
```



This code will provide us with insights into the relationship between smoking habits and the likelihood of having heart disease or a heart attack. The visualization will help in understanding the average heart disease/attack among smokers and non-smokers, answering our hypothesis question.

Individuals with healthcare access are less likely to report difficulty walking (DiffWalk).

In [57]:
```
import matplotlib.pyplot as plt
# Group by healthcare access and calculate percentage reporting difficulty wal
diffwalk_percentage = data.groupby('AnyHealthcare')['DiffWalk'].mean() * 100

# Plotting
diffwalk_percentage.plot(kind='bar', color=['pink', 'blue'], alpha=0.75)

# Adding labels and title
plt.title('Percentage Reporting Difficulty Walking by Healthcare Access')
plt.xlabel('Healthcare Access (1: No, 0: Yes)')
plt.ylabel('Percentage Reporting Difficulty Walking')

# Adding percentage labels on bars
for index, value in enumerate(diffwalk_percentage):
    plt.text(index, value + 1, f'{value:.1f}%', ha='center')

# Show plot
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



Calculate the percentage of individuals reporting difficulty walking (DiffWalk = 1) in both groups (with and without healthcare access). A lower percentage in the healthcare access group would support the hypothesis that healthcare access correlates with fewer mobility issues.