Importing Library

In [1]:
```
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [118]:
```
df = pd.read_csv(r'C:\Users\Ritik\Downloads\CASESTUDY.csv')
```

Descriptive Analytics

In [162]:
```
print("First few rows of the dataset")
df.head(4)
```

```
First few rows of the dataset
```

Out[162]:

	Year	State Abbreviation	Indian States	Chronic Diseases	Family Living Status	GENDER	Age in years	Education	Income	Psychological Health	Psychosocial Factors	Sleep duration(Hrs)
0	2021	PB	Punjab	Asthma	Nuclear	Female	25-34	Graduation	2186613.0	Good	Alcohol	7
1	2019	PB	Punjab	Chronic Obstructive Pulmonary Disease	Roomates	Male	65 and above	Post Graduation	4712953.0	Good	Prescriptive Drugs	6
2	2018	MH	Maharashtra	Cardiovascular Disease	Nuclear	Male	55-64	High School	704213.0	Bad	Prescriptive Drugs	11
3	2018	RJ	Rajasthan	Chronic Obstructive Pulmonary Disease	Nuclear	Female	25-34	High School	704213.0	Good	Prescriptive Drugs	12

In [120]:
```
print("\nDescriptive stats of dataset")
df.describe()
```

```
Descriptive stats of dataset
```

Out[120]:

	Year	Income	Sleep duration(Hrs)	Follows a Diet Plan	Obesity / Weight Status	Physical Activity
count	233875.000000	1.964100e+05	233875.000000	233875.000000	233875.000000	233875.000000
mean	2018.696265	1.355391e+06	7.691339	1.674613	0.042420	0.070012
std	1.365107	1.394714e+06	2.555662	0.835842	0.201546	0.255167
min	2017.000000	1.000000e+03	1.000000	1.000000	0.000000	0.000000
25%	2017.000000	3.594412e+05	6.000000	1.000000	0.000000	0.000000
50%	2019.000000	7.042130e+05	8.000000	1.000000	0.000000	0.000000
75%	2020.000000	2.109548e+06	9.000000	2.000000	0.000000	0.000000
max	2021.000000	4.999998e+06	15.000000	3.000000	1.000000	1.000000

In [121]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 233875 entries, 0 to 233874
Data columns (total 16 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   Year                            233875 non-null  int64  
 1   State Abbreviation              233875 non-null  object 
 2   Indian States                   233875 non-null  object 
 3   Chronic Diseases                233875 non-null  object 
 4   Family Living Status            233875 non-null  object 
 5   GENDER                          233875 non-null  object 
 6   Age in years                    233875 non-null  object 
 7   Education                       233875 non-null  object 
 8   Income                          196410 non-null  float64
 9   Psychological Health            233875 non-null  object 
 10  Psychosocial Factors            191895 non-null  object 
 11  Sleep duration(Hrs)             233875 non-null  int64  
 12  Frequency of healthcare visits  233875 non-null  object 
 13  Follows a Diet Plan             233875 non-null  int64  
 14  Obesity / Weight Status         233875 non-null  int64  
 15  Physical Activity               233875 non-null  int64  
dtypes: float64(1), int64(5), object(10)
memory usage: 28.5+ MB
```

In [122]:
```
df.isnull()
```

Out[122]:

	Year	State Abbreviation	Indian States	Chronic Diseases	Family Living Status	GENDER	Age in years	Education	Income	Psychological Health	Psychosocial Factors	Sleep duration(Hrs)	Frequency of healthcare visits
0	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	True	False	False	False	False
3	False	False	False	False	False	False	False	False	True	False	False	False	False
4	False	False	False	False	False	False	False	False	True	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
233870	False	False	False	False	False	False	False	False	False	False	False	False	False
233871	False	False	False	False	False	False	False	False	False	False	False	False	False
233872	False	False	False	False	False	False	False	False	True	False	False	False	False
233873	False	False	False	False	False	False	False	False	False	False	False	False	False
233874	False	False	False	False	False	False	False	False	False	False	False	False	False

233875 rows × 16 columns

EDA - for unclean data

Pre-Cleaned Data EDA:

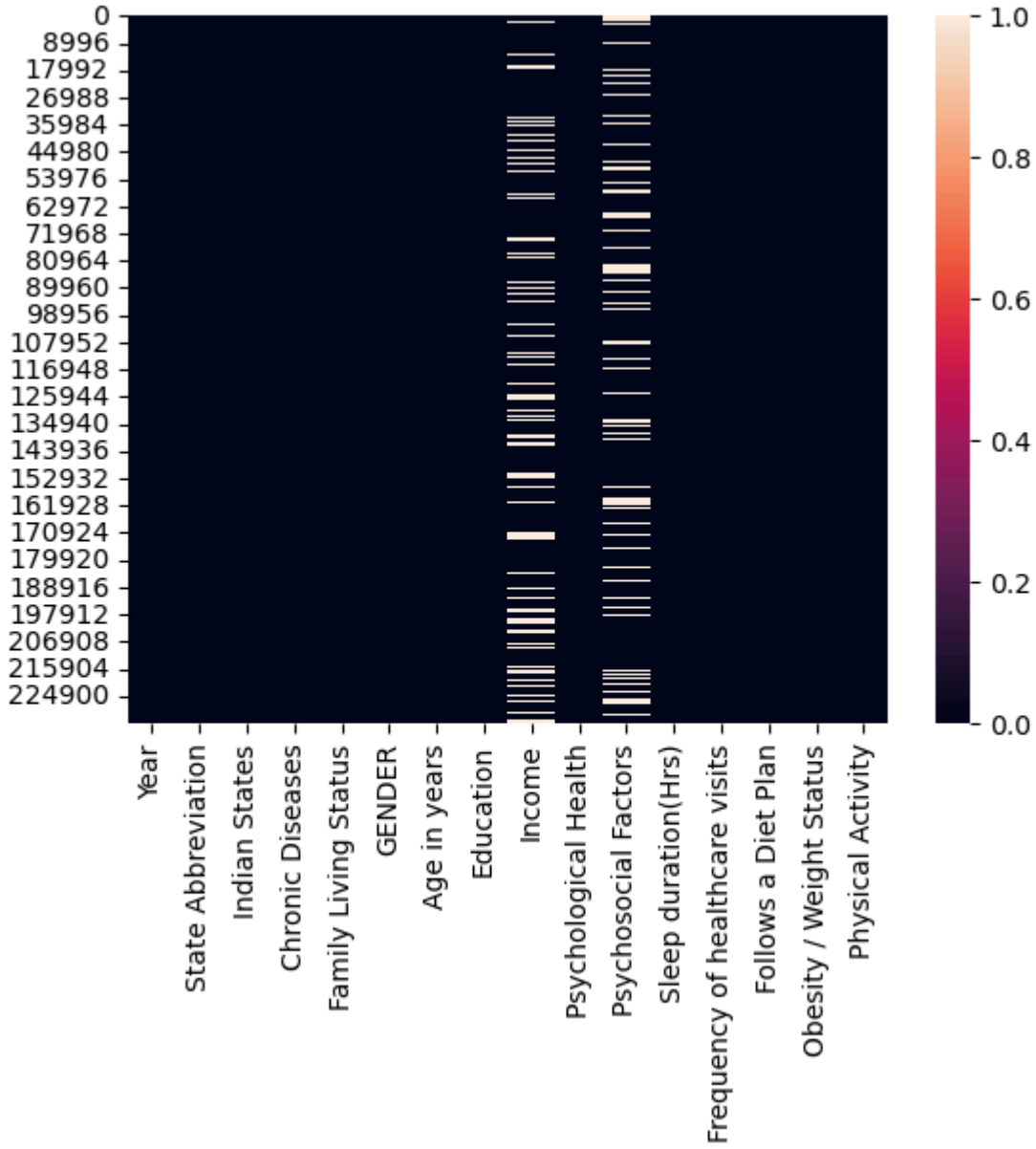
Missing Values Heatmap: Visualize missing values.

Distributions and Counts: Plot distributions for numerical columns like 'Income' and counts for categorical columns.

Pair Plots and Correlation: Examine relationships and correlations between numerical variables.

In [125]:
```
sns.heatmap(df.isnull())
```

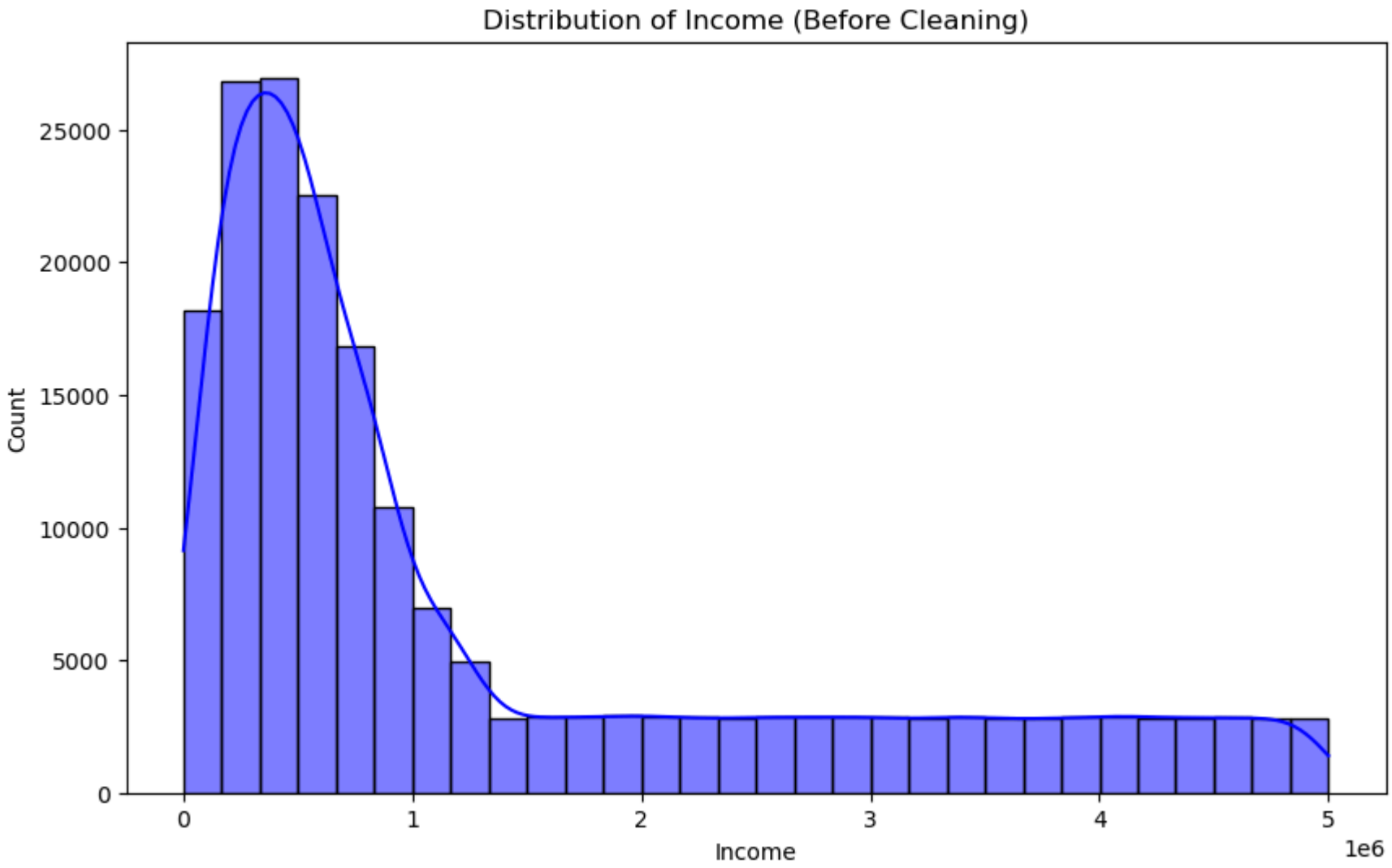
Out[125]: <Axes: >



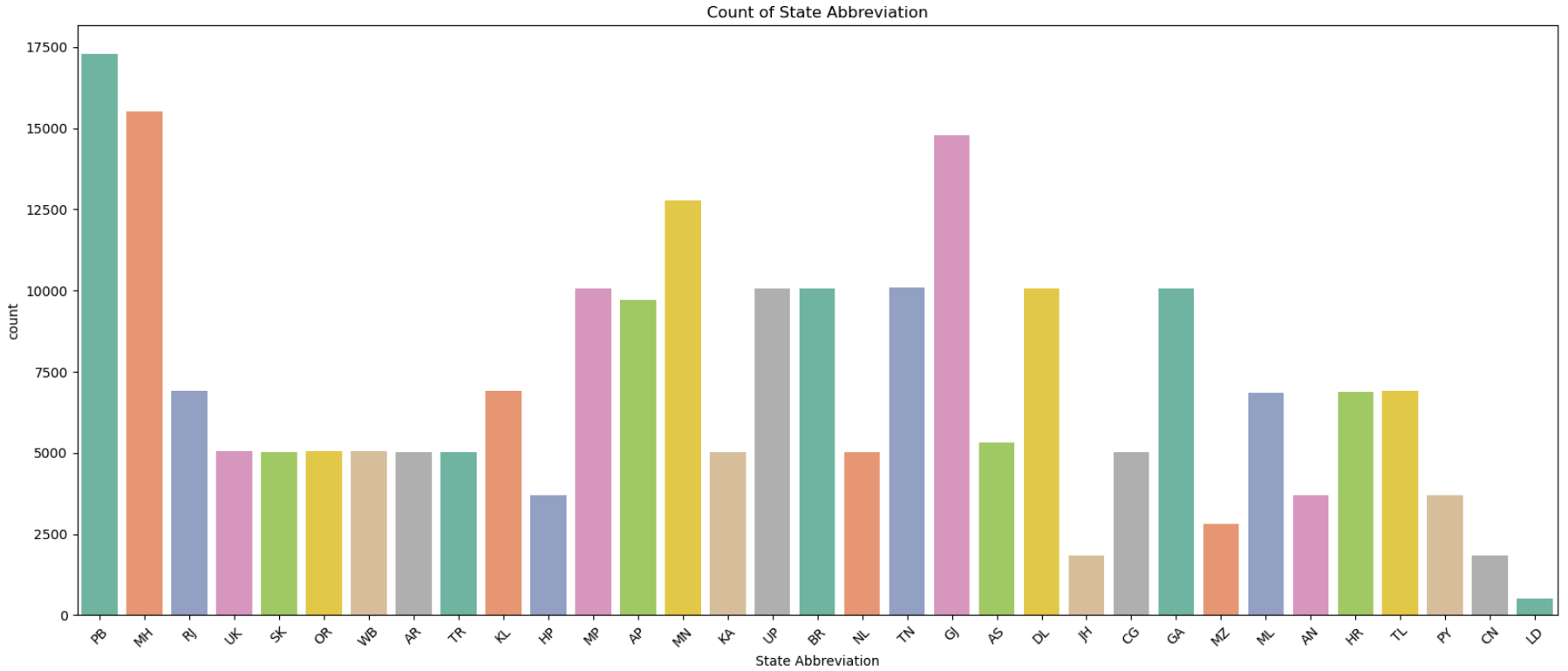
In [126]:
```
# Check for missing values
missing_values = df.isnull().sum()
print("Missing values in each column:\n", missing_values)
```

```
Missing values in each column:
 Year                                0
State Abbreviation                   0
Indian States                        0
Chronic Diseases                     0
Family Living Status                 0
GENDER                               0
Age in years                         0
Education                            0
Income                           37465
Psychological Health                 0
Psychosocial Factors             41980
Sleep duration(Hrs)                  0
Frequency of healthcare visits       0
Follows a Diet Plan                  0
Obesity / Weight Status              0
Physical Activity                    0
dtype: int64
```

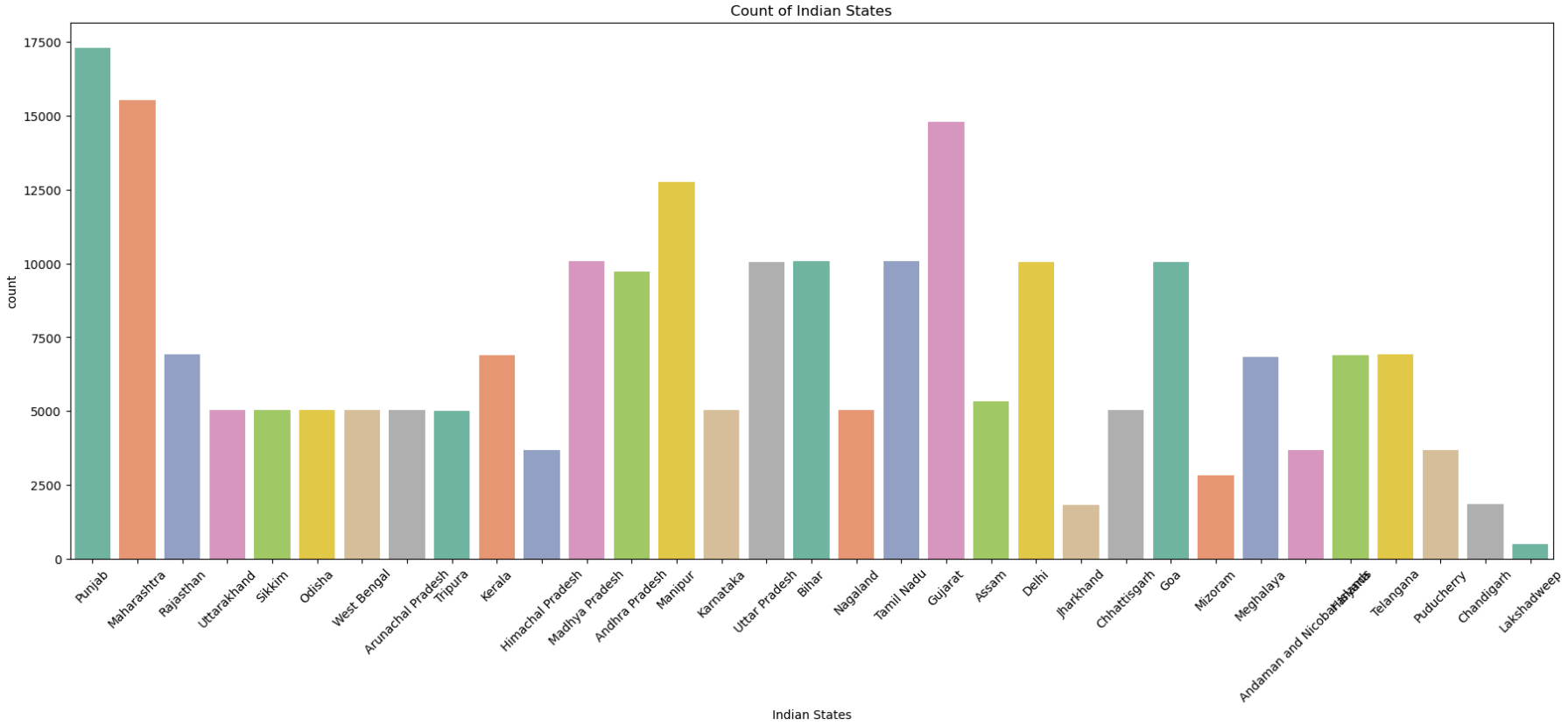
In [127]:
```
# Distribution of 'Income' (before handling missing values)
plt.figure(figsize=(10, 6))
sns.histplot(df['Income'], kde=True, bins=30, color='blue')
plt.title('Distribution of Income (Before Cleaning)')
plt.show()
```



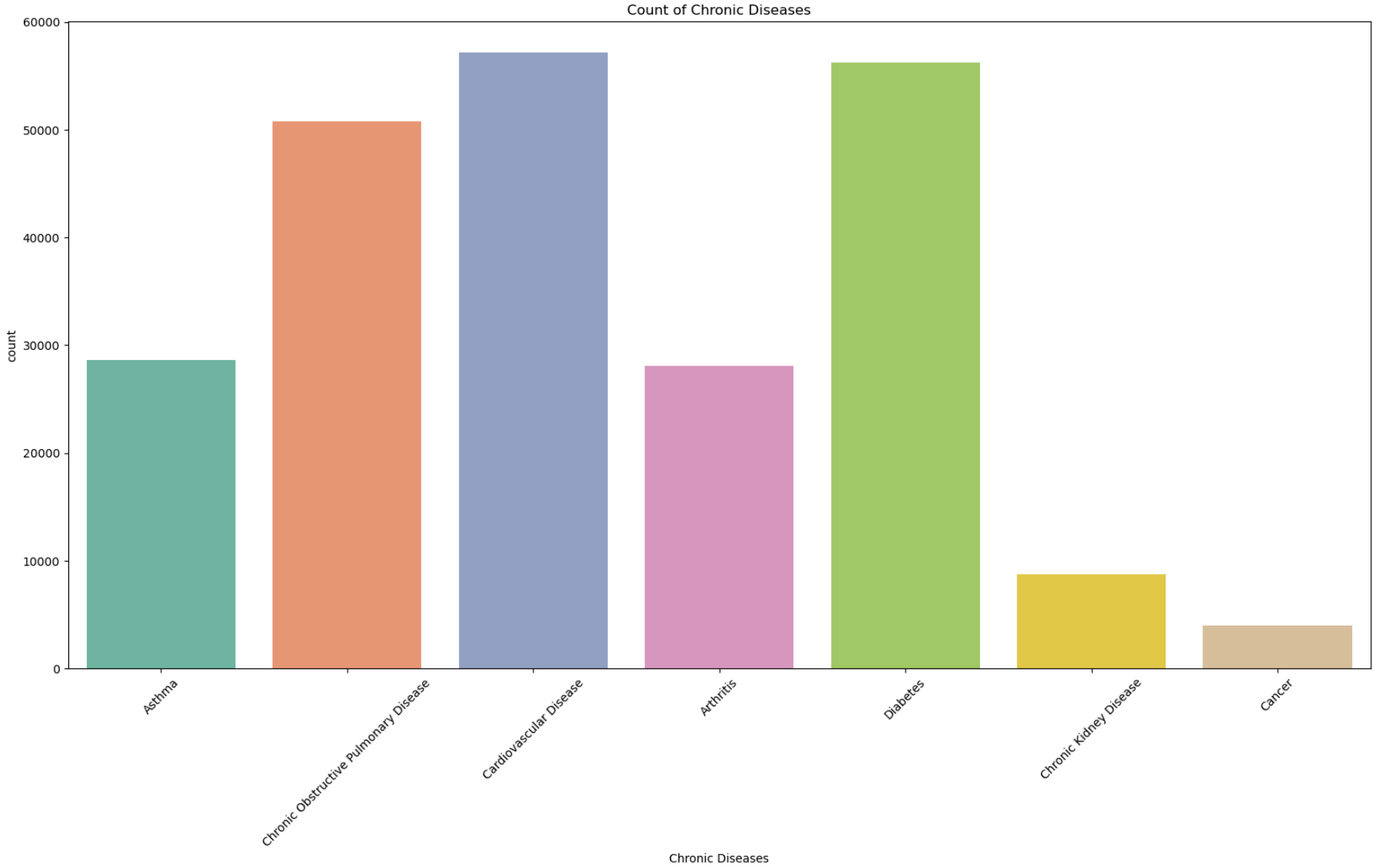
In [174]:
```
# Visualize categorical variables
plt.figure(figsize=(20, 8))
sns.countplot(x='State Abbreviation', data=df, palette='Set2')
plt.title('Count of State Abbreviation')
plt.xticks(rotation=45)
plt.show()
```



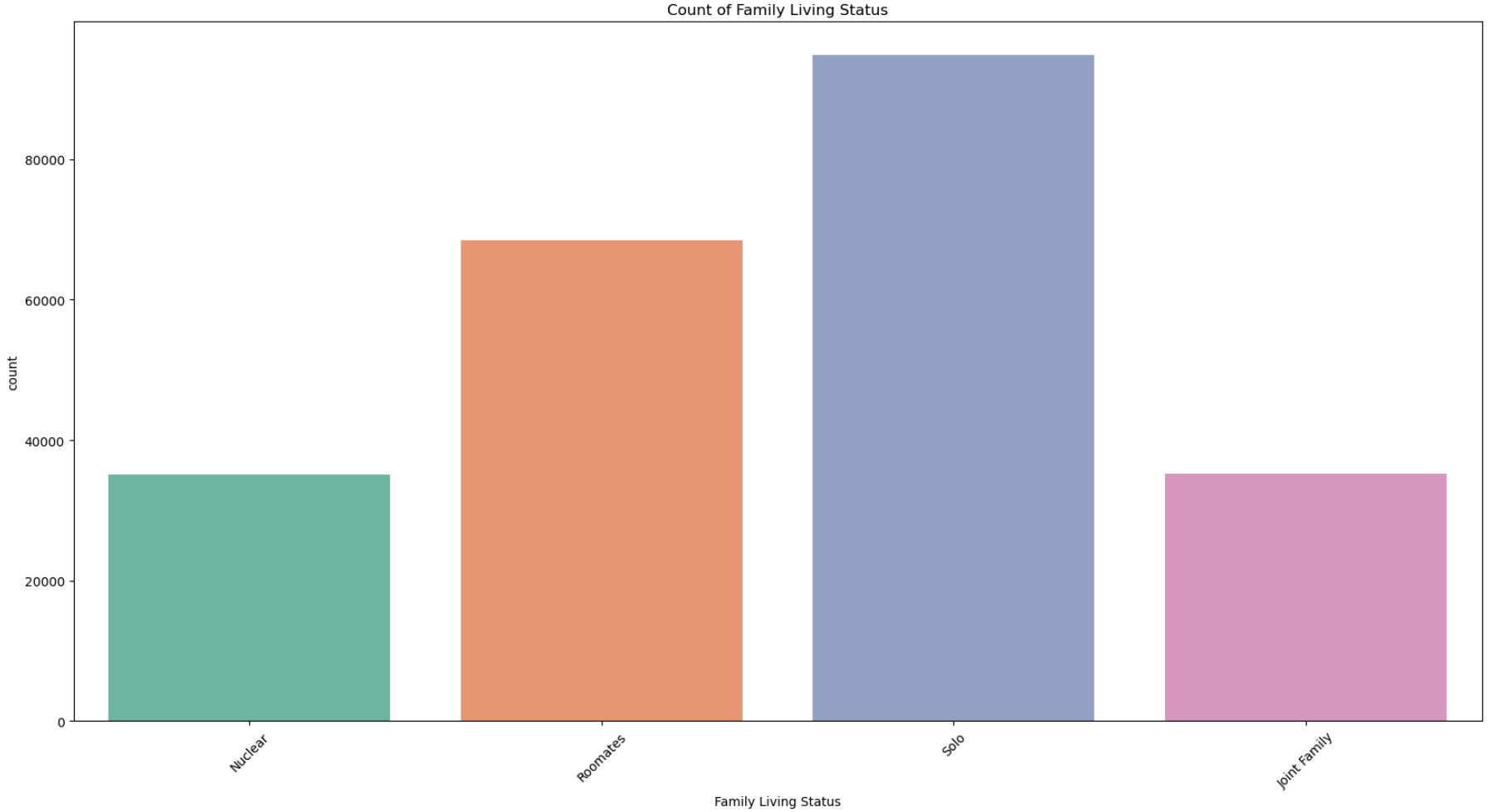
In [172]:
```
# Visualize categorical variables
categorical_columns = ['Indian States']
for col in categorical_columns:
    plt.figure(figsize=(22, 8))
    sns.countplot(x=col, data=df, palette='Set2')  # Use x=col and specify data=df
    plt.title(f'Count of {col}')
    plt.xticks(rotation=45)
    plt.show()
```



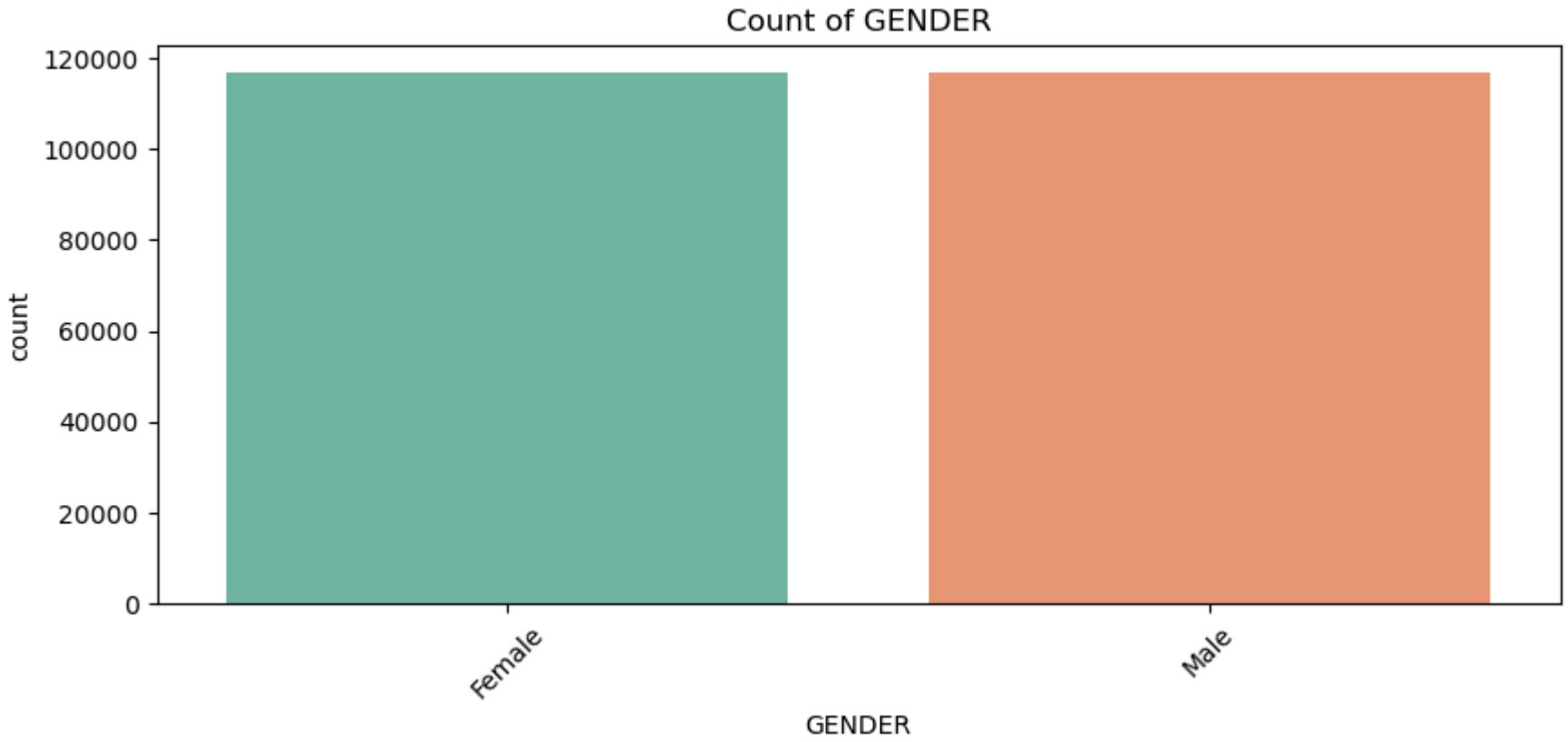
In [176]:
```
# Visualize categorical variables
categorical_columns1 = ['Chronic Diseases']
for col in categorical_columns1:
    plt.figure(figsize=(20, 10))
    sns.countplot(x=col, data=df, palette='Set2')  # Use x=col and specify data=df
    plt.title(f'Count of {col}')
    plt.xticks(rotation=45)
    plt.show()
```


Count of Chronic Diseases
60000
50000
40000
30000
20000
10000
0
count
Asthma
Chronic Obstructive Pulmonary Disease
Cardiovascular Disease
Arthritis
Diabetes
Chronic Kidney Disease
Cancer
Chronic Diseases

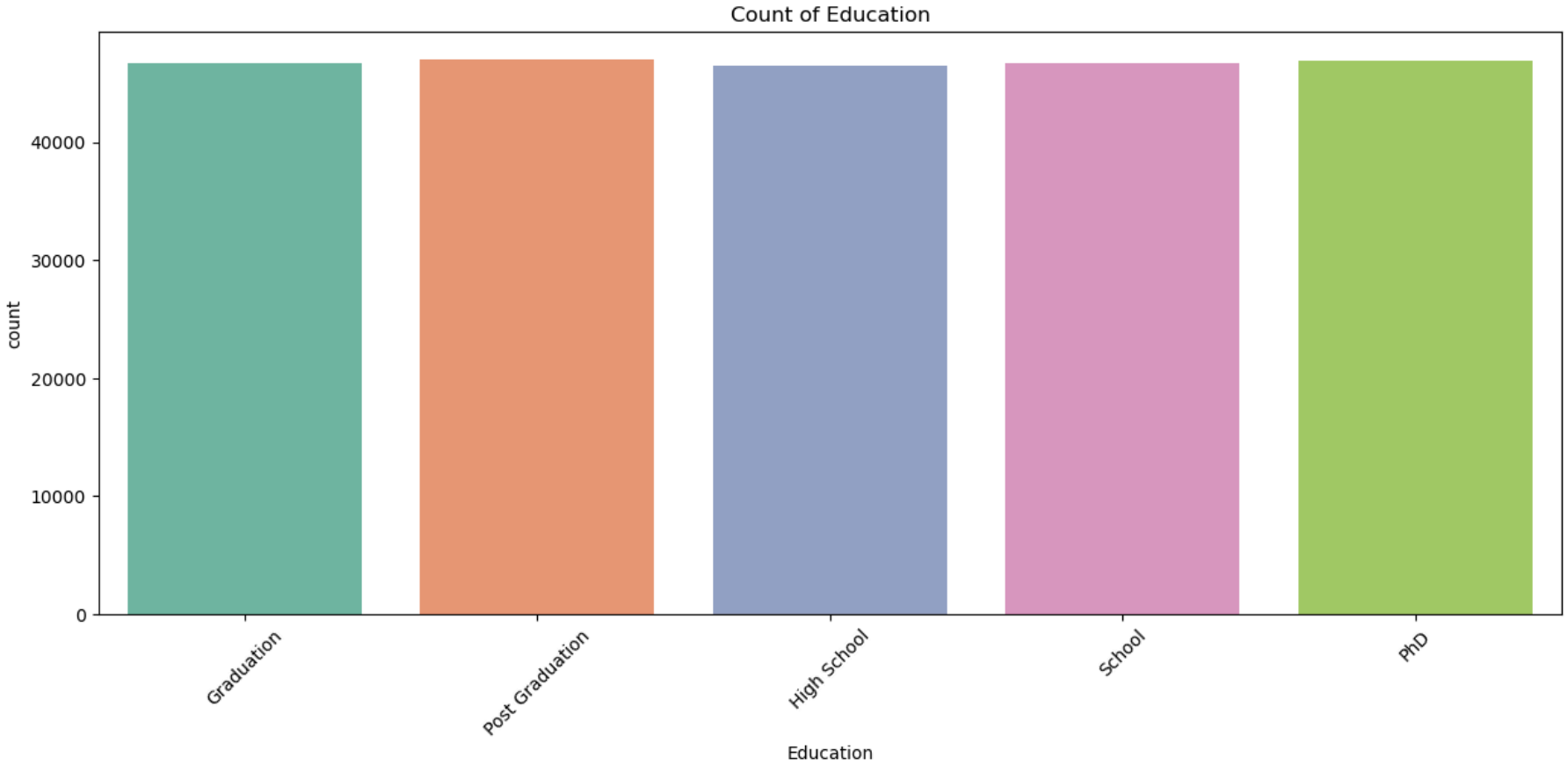
In [177]:
```
# Visualize categorical variables
categorical_columns2 = ['Family Living Status']
for col in categorical_columns2:
    plt.figure(figsize=(20, 10))
    sns.countplot(x=col, data=df, palette='Set2')  # Use x=col and specify data=df
    plt.title(f'Count of {col}')
    plt.xticks(rotation=45)
    plt.show()
```



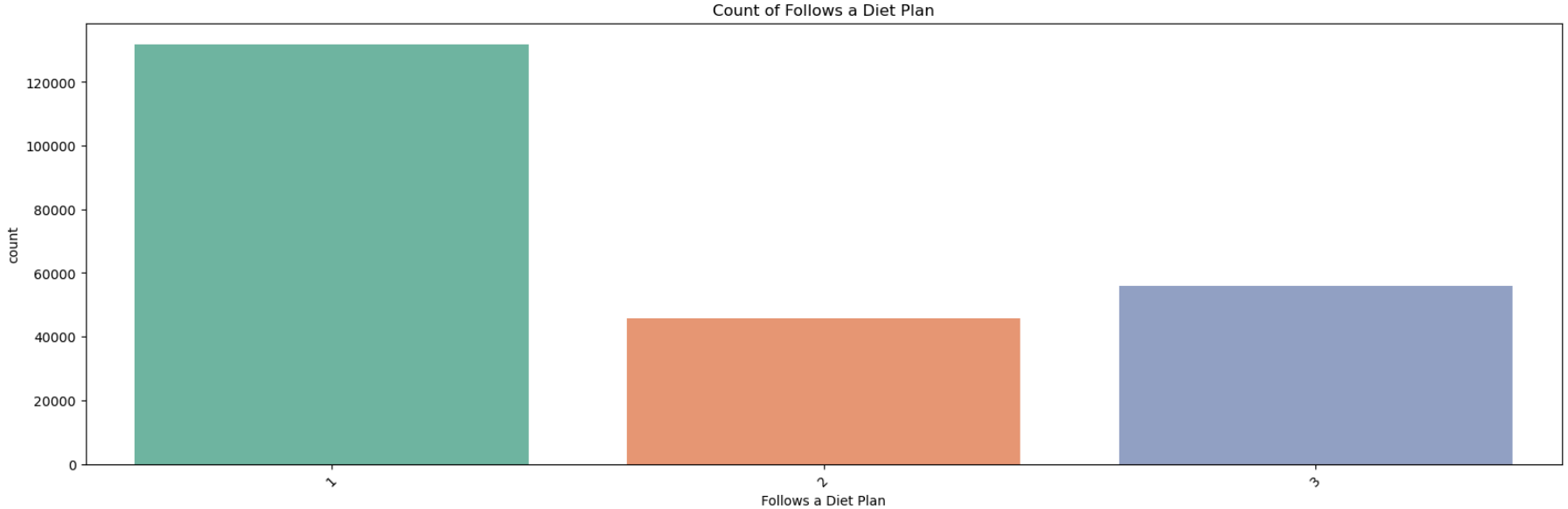
In [178]:
```
# Visualize categorical variables
categorical_columns3 = [ 'GENDER']
for col in categorical_columns3:
    plt.figure(figsize=(10, 4))
    sns.countplot(x=col, data=df, palette='Set2')  # Use x=col and specify data=df
    plt.title(f'Count of {col}')
    plt.xticks(rotation=45)
    plt.show()
```



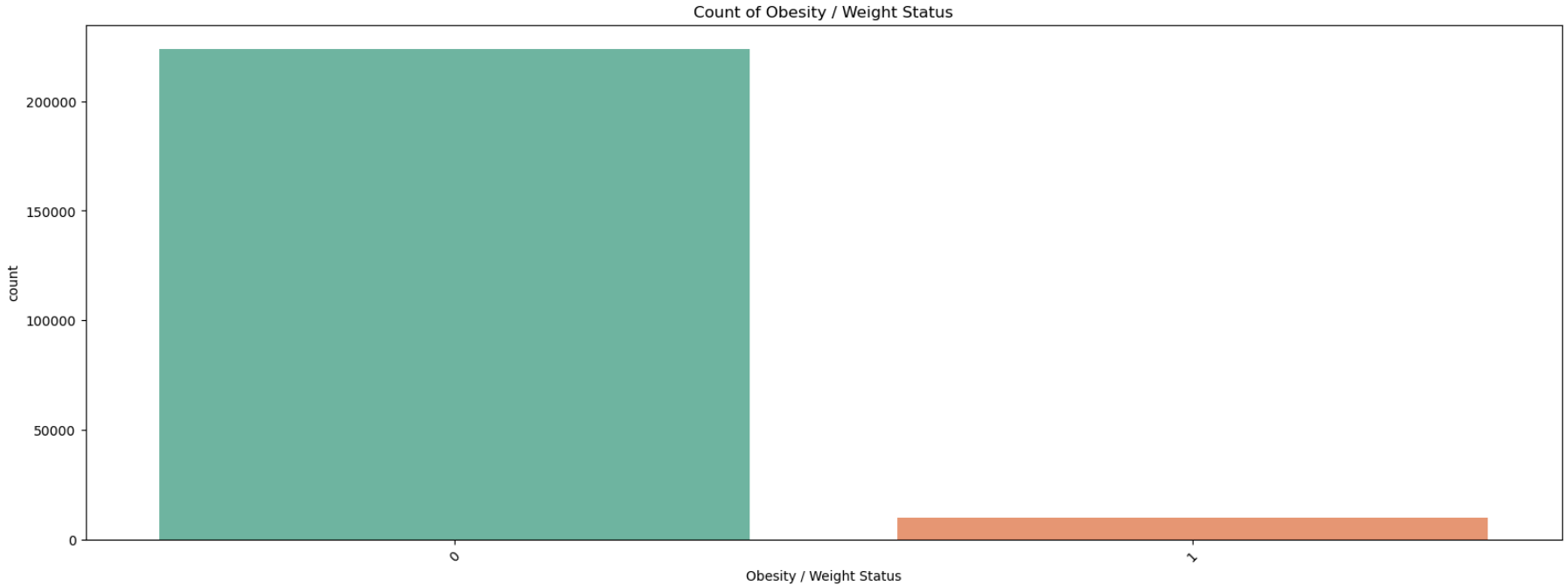
In [179]:
```
# Visualize categorical variables
categorical_columns4 = ['Education']
for col in categorical_columns4:
    plt.figure(figsize=(15, 6))
    sns.countplot(x=col, data=df, palette='Set2')  # Use x=col and specify data=df
    plt.title(f'Count of {col}')
    plt.xticks(rotation=45)
    plt.show()
```



In [180]:
```
# Visualize categorical variables
categorical_columns5 = ['Follows a Diet Plan']
for col in categorical_columns5:
    plt.figure(figsize=(20, 6))
    sns.countplot(x=col, data=df, palette='Set2')  # Use x=col and specify data=df
    plt.title(f'Count of {col}')
    plt.xticks(rotation=45)
    plt.show()
```

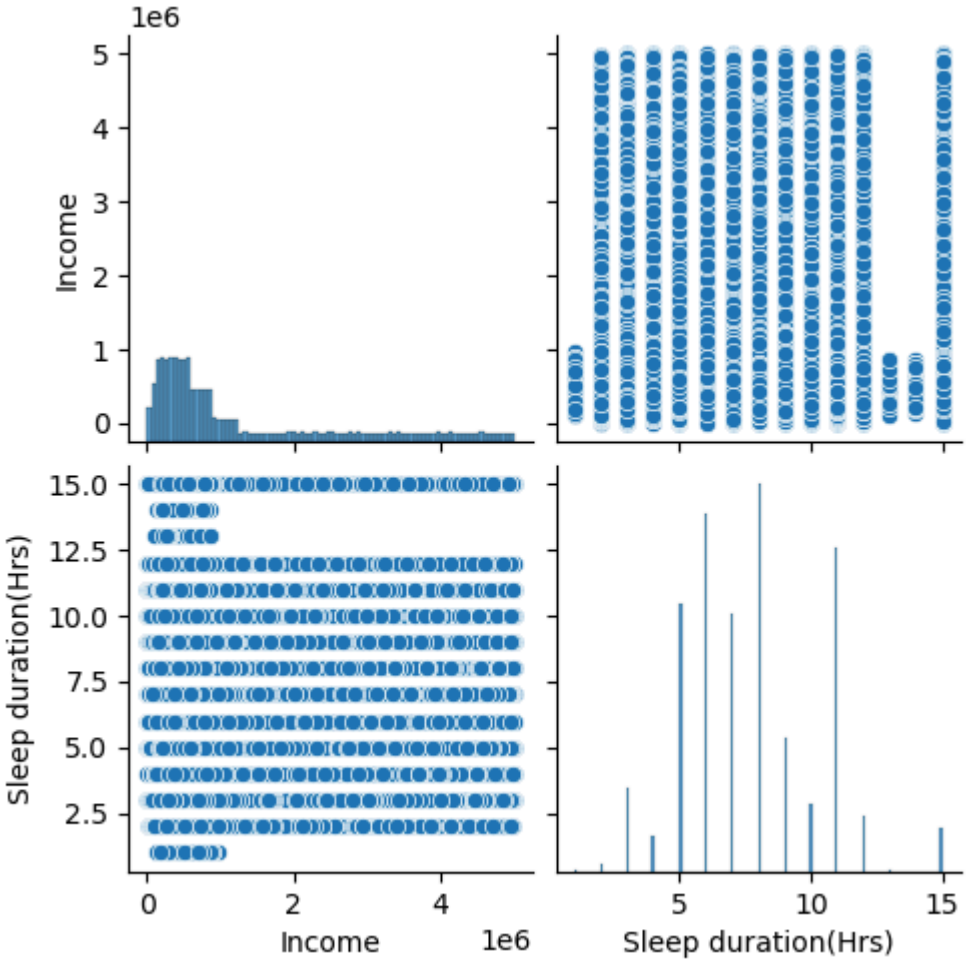


In [181]:
```
# Visualize categorical variables
categorical_columns6 = ['Obesity / Weight Status']
for col in categorical_columns6:
    plt.figure(figsize=(20, 7))
    sns.countplot(x=col, data=df, palette='Set2')  # Use x=col and specify data=df
    plt.title(f'Count of {col}')
    plt.xticks(rotation=45)
    plt.show()
```

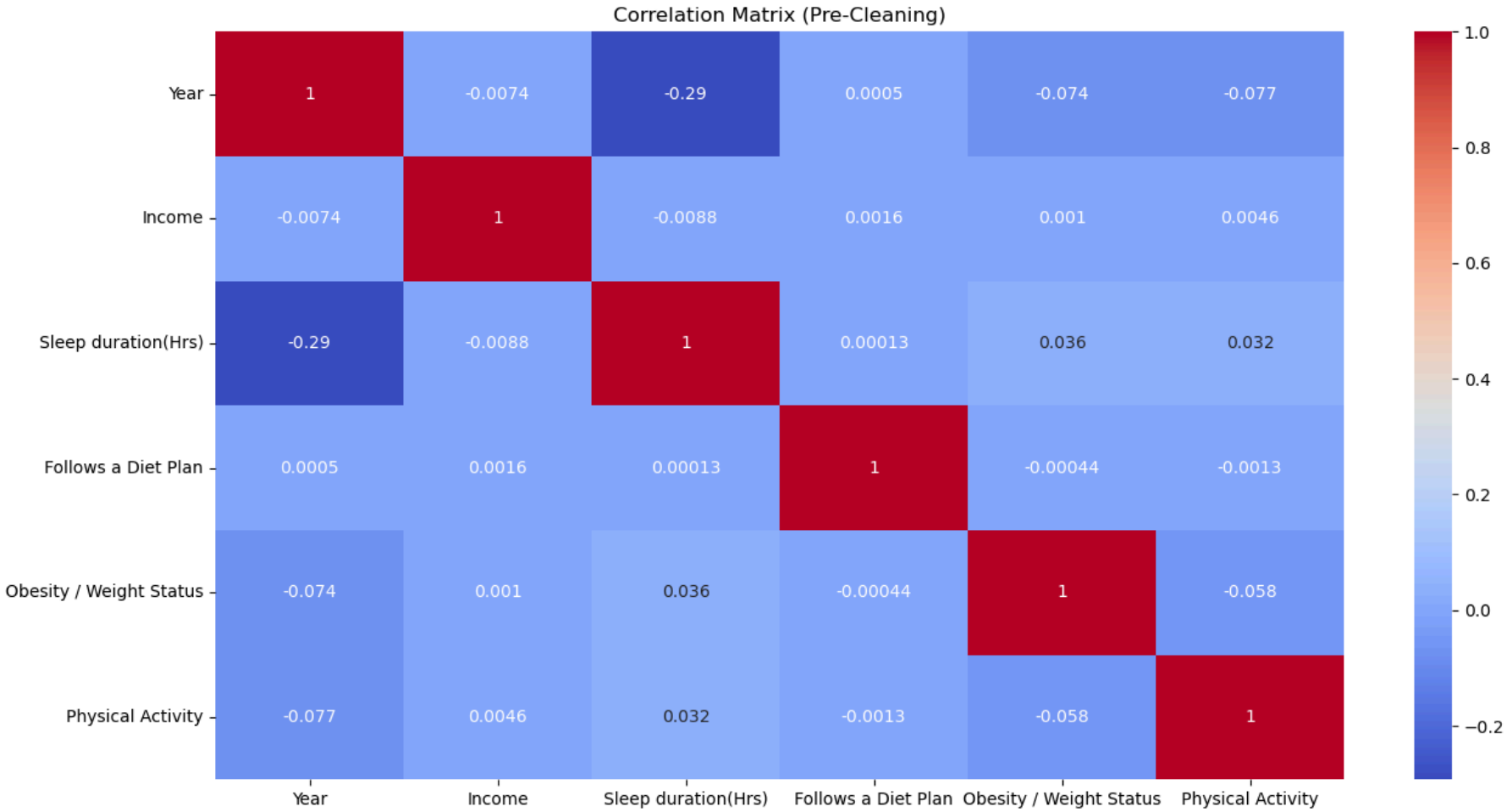


In [138]:
```
# Pair plot to identify relationships between numerical variables
sns.pairplot(df[['Age in years', 'Income', 'Sleep duration(Hrs)', 'Frequency of healthcare visits']])
plt.show()
```

```
C:\Users\Ritik\anaconda3\ane\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed t
o tight
  self._figure.tight_layout(*args, **kwargs)
```



In [189]:
```
# Select only numeric columns for correlation
numeric_columns = df.select_dtypes(include=[np.number])

# Correlation matrix
correlation_matrix = numeric_columns.corr()
plt.figure(figsize=(15, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix (Pre-Cleaning)')
plt.show()
```



Data Cleaning

In [140]:
```
# Checking for missing values in columns
missing_data = df.isnull().sum()
missing_data
```

Out[140]:
```
Year                                0
State Abbreviation                  0
Indian States                       0
Chronic Diseases                    0
Family Living Status                0
GENDER                              0
Age in years                        0
Education                           0
Income                          37465
Psychological Health                0
Psychosocial Factors            41980
Sleep duration(Hrs)                 0
Frequency of healthcare visits      0
Follows a Diet Plan                 0
Obesity / Weight Status             0
Physical Activity                   0
dtype: int64
```

In [142]:
```
#Check for duplicate rows
duplicate_rows = df.duplicated()
print(duplicate_rows.any())
print(duplicate_rows.sum())
```

```
True
230
```

In [143]:
```
df = df.drop_duplicates()
```

In [144]:
```
df.isna().sum()
```

Out[144]:
```
Year                              0
State Abbreviation                0
Indian States                     0
Chronic Diseases                  0
Family Living Status              0
GENDER                            0
Age in years                      0
Education                         0
Income                        37235
Psychological Health              0
Psychosocial Factors          41932
Sleep duration(Hrs)               0
Frequency of healthcare visits    0
Follows a Diet Plan               0
Obesity / Weight Status           0
Physical Activity                 0
dtype: int64
```

Data imputation for income by median

In [145]:
```
#Calculate the median of the income column
median_income = df['Income'].median()
```

In [146]:
```
# Step 3: Impute missing values with the median
df['Income'].fillna(median_income, inplace=True)
```

In [147]:
```
# Display the DataFrame after imputation
df.head(7)
```

Out[147]:

	Year	State Abbreviation	Indian States	Chronic Diseases	Family Living Status	GENDER	Age in years	Education	Income	Psychological Health	Psychosocial Factors	Sleep duration(Hrs)
0	2021	PB	Punjab	Asthma	Nuclear	Female	25-34	Graduation	2186613.0	Good	Alcohol	7
1	2019	PB	Punjab	Chronic Obstructive Pulmonary Disease	Roomates	Male	65 and above	Post Graduation	4712953.0	Good	Prescriptive Drugs	6
2	2018	MH	Maharashtra	Cardiovascular Disease	Nuclear	Male	55-64	High School	704213.0	Bad	Prescriptive Drugs	11
3	2018	RJ	Rajasthan	Chronic Obstructive Pulmonary Disease	Nuclear	Female	25-34	High School	704213.0	Good	Prescriptive Drugs	12
4	2021	UK	Uttarakhand	Arthritis	Nuclear	Female	18-24	High School	704213.0	Good	Prescriptive Drugs	10
5	2018	SK	Sikkim	Diabetes	Roomates	Female	18-24	School	400257.0	Good	Alcohol and Smoking	8
6	2017	PB	Punjab	Asthma	Nuclear	Male	55-64	Post Graduation	1495797.0	Bad	Smoking	9

Data Imputation for Psychological factors

In [23]:
```
# # Impute missing values with the mode
## mode_value = df['Psychosocial Factors'].mode()[0]
## df['Psychosocial Factors'].fillna(mode_value, inplace=True)
```

In [148]:
```
df['Psychosocial Factors'] = df['Psychosocial Factors'].fillna('NA')
```

In [149]:
```
df.head(7)
```

Out[149]:

	Year	State Abbreviation	Indian States	Chronic Diseases	Family Living Status	GENDER	Age in years	Education	Income	Psychological Health	Psychosocial Factors	Sleep duration(Hrs)
0	2021	PB	Punjab	Asthma	Nuclear	Female	25-34	Graduation	2186613.0	Good	Alcohol	7
1	2019	PB	Punjab	Chronic Obstructive Pulmonary Disease	Roomates	Male	65 and above	Post Graduation	4712953.0	Good	Prescriptive Drugs	6
2	2018	MH	Maharashtra	Cardiovascular Disease	Nuclear	Male	55-64	High School	704213.0	Bad	Prescriptive Drugs	11
3	2018	RJ	Rajasthan	Chronic Obstructive Pulmonary Disease	Nuclear	Female	25-34	High School	704213.0	Good	Prescriptive Drugs	12
4	2021	UK	Uttarakhand	Arthritis	Nuclear	Female	18-24	High School	704213.0	Good	Prescriptive Drugs	10
5	2018	SK	Sikkim	Diabetes	Roomates	Female	18-24	School	400257.0	Good	Alcohol and Smoking	8
6	2017	PB	Punjab	Asthma	Nuclear	Male	55-64	Post Graduation	1495797.0	Bad	Smoking	9

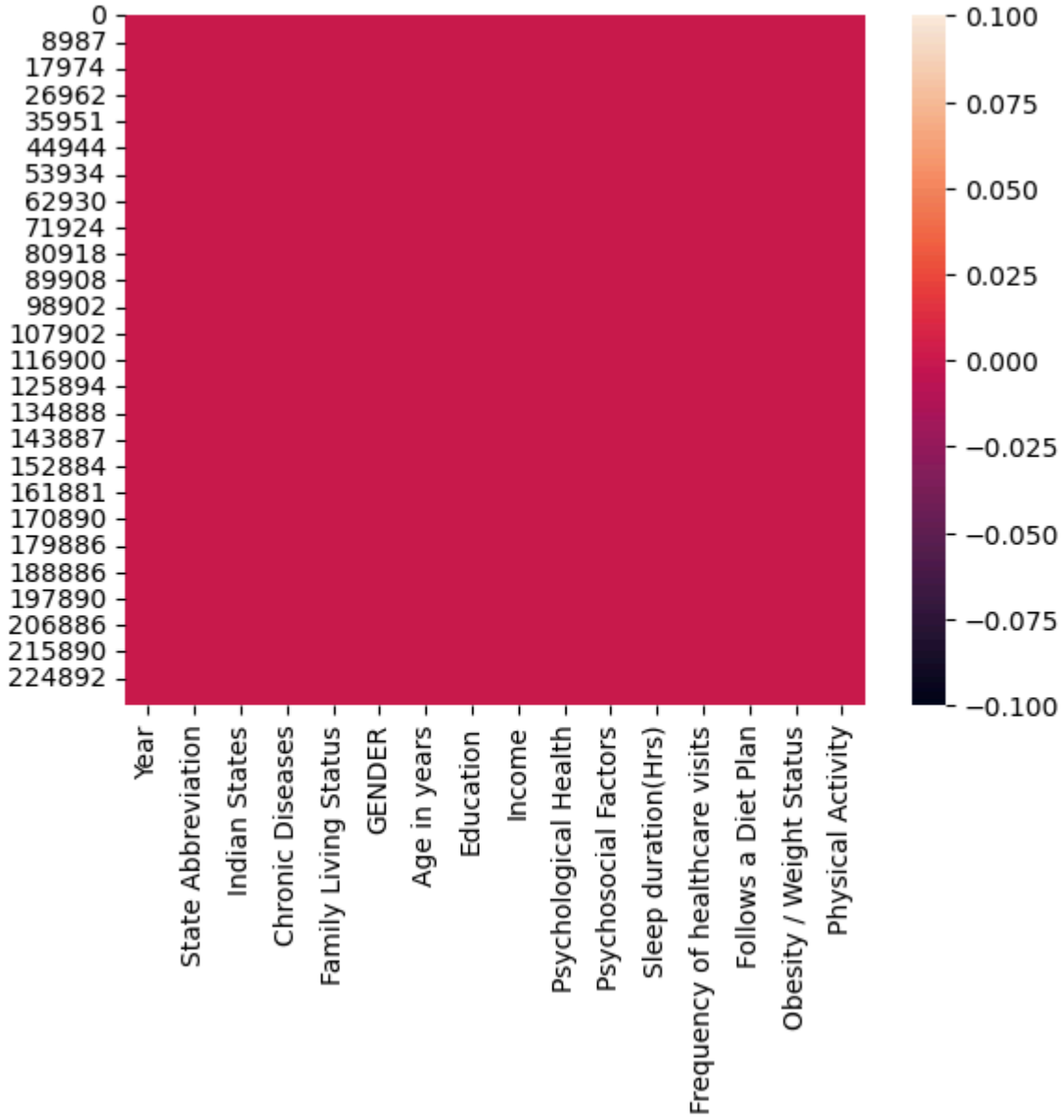
Post data clean -eda

In [150]:
```
df.isna().sum()
```

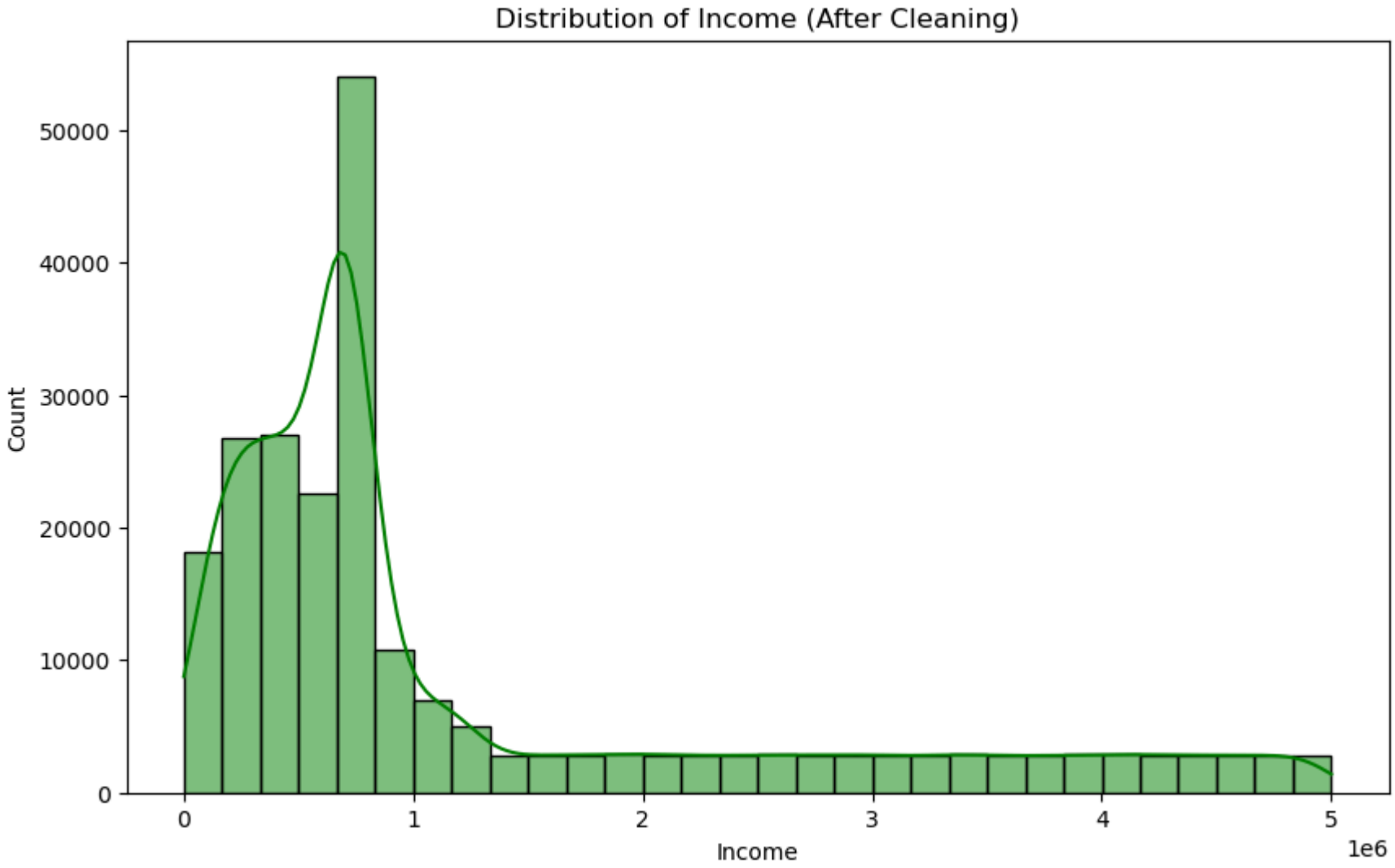
Out[150]:
```
Year                              0
State Abbreviation                0
Indian States                     0
Chronic Diseases                  0
Family Living Status              0
GENDER                            0
Age in years                      0
Education                         0
Income                            0
Psychological Health              0
Psychosocial Factors              0
Sleep duration(Hrs)               0
Frequency of healthcare visits    0
Follows a Diet Plan               0
Obesity / Weight Status           0
Physical Activity                 0
dtype: int64
```

In [151]:
```
sns.heatmap(df.isnull())
```

Out[151]: <Axes: >

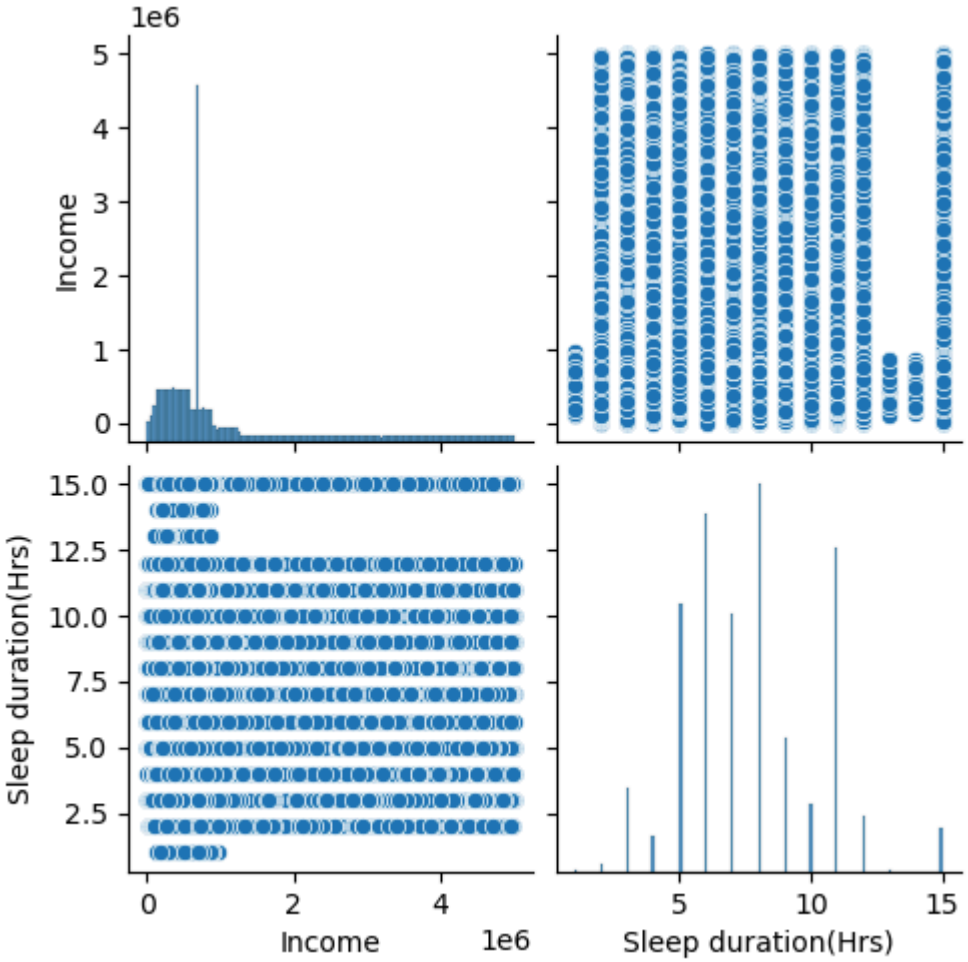


In [152]:
```
# Distribution of 'Income' (after handling missing values)
plt.figure(figsize=(10, 6))
sns.histplot(df['Income'], kde=True, bins=30, color='green')
plt.title('Distribution of Income (After Cleaning)')
plt.show()
```

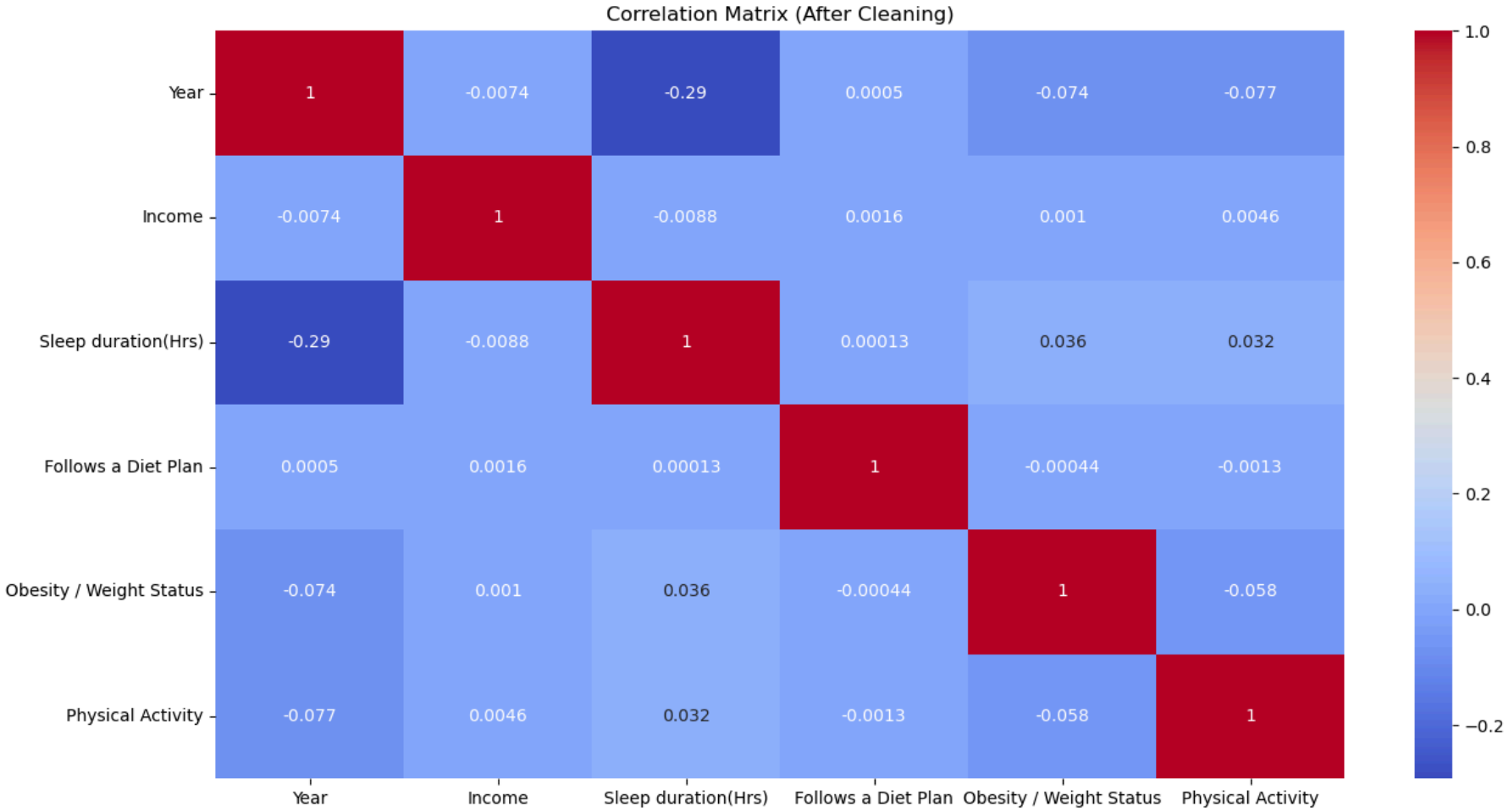


In [153]:
```
# Pair plot to identify relationships between numerical variables after cleaning
sns.pairplot(df[['Age in years', 'Income', 'Sleep duration(Hrs)', 'Frequency of healthcare visits']])
plt.show()
```

```
C:\Users\Ritik\anaconda3\ane\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed t
o tight
  self._figure.tight_layout(*args, **kwargs)
```

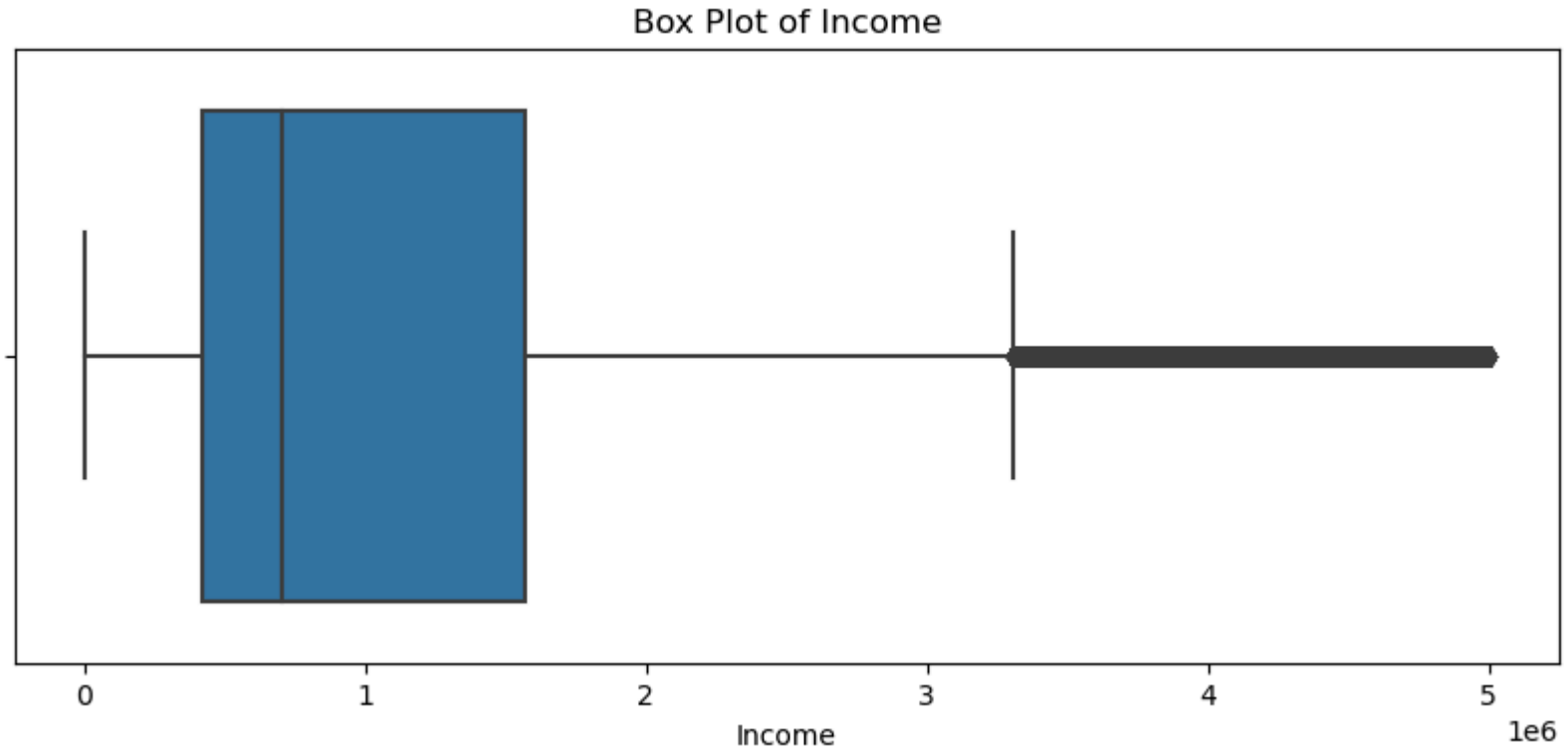


In [188]:
```
# Select only numeric columns for correlation matrix
numeric_df = df.select_dtypes(include=[float, int])

# Correlation matrix after cleaning
correlation_matrix_cleaned = numeric_df.corr()

# Plot heatmap of the correlation matrix
plt.figure(figsize=(15, 8))
sns.heatmap(correlation_matrix_cleaned, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix (After Cleaning)')
plt.show()
```

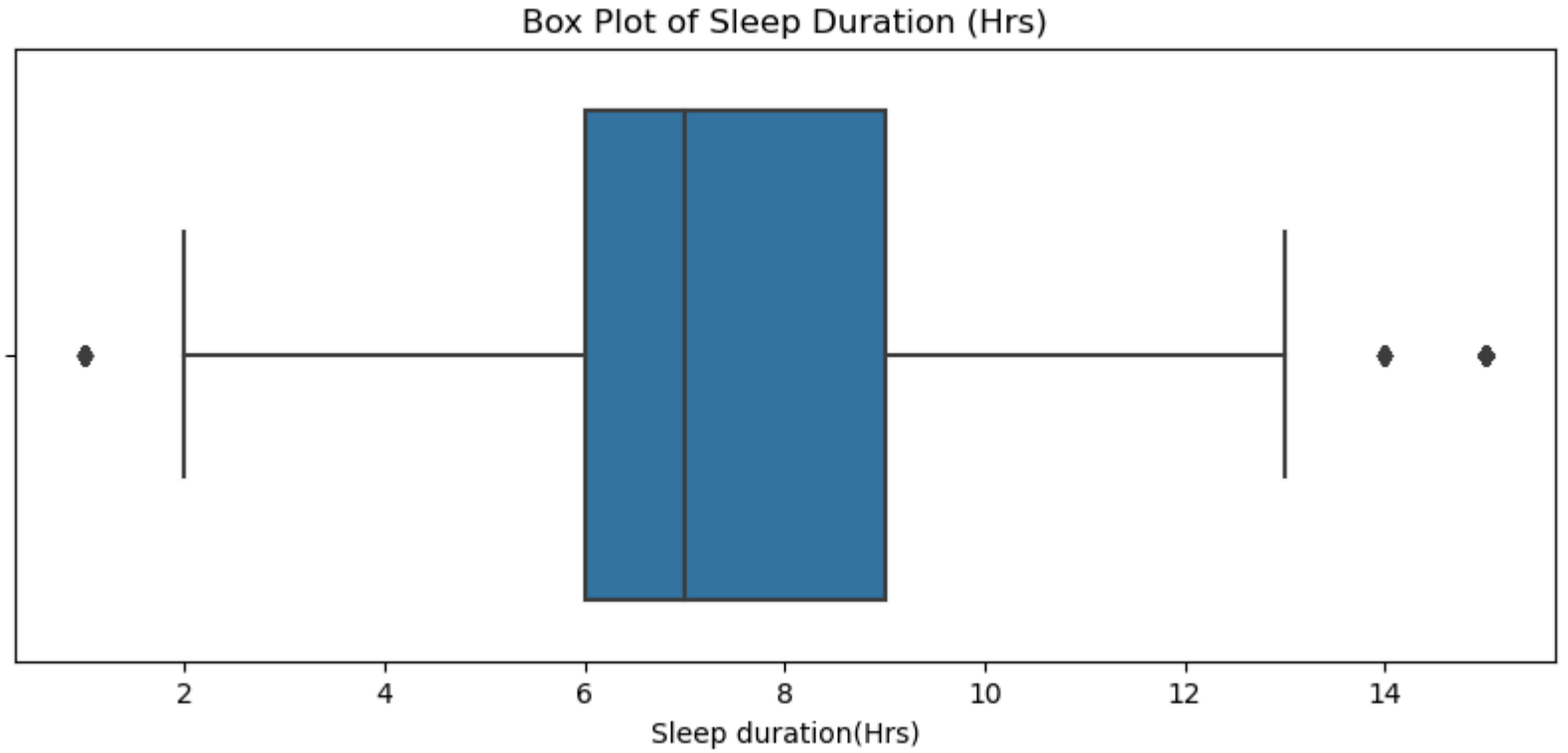


Box plot for outlier detection after cleaning

In [185]:
```
plt.figure(figsize=(10, 4))
sns.boxplot(x=df['Income'])
plt.title('Box Plot of Income')
plt.show()
```



In [186]:
```
plt.figure(figsize=(10, 4))
sns.boxplot(x=df['Sleep duration(Hrs)'])
plt.title('Box Plot of Sleep Duration (Hrs)')
plt.show()
```



Saving clean dataset from dataframe

In [35]:
```
# df.to_excel('output_file.xlsx', index=False)
# # Confirm the file has been saved
# print("DataFrame has been saved to 'output_file.xlsx'")
```

In [36]:
```
import os
# Get the current working directory
# print(os.getcwd())
```

In [187]:
```
df.head(5)
```

Out[187]:

	Year	State Abbreviation	Indian States	Chronic Diseases	Family Living Status	GENDER	Age in years	Education	Income	Psychological Health	Psychosocial Factors	Sleep duration(Hrs)
0	2021	PB	Punjab	Asthma	Nuclear	Female	25-34	Graduation	2186613.0	Good	Alcohol	7
1	2019	PB	Punjab	Chronic Obstructive Pulmonary Disease	Roomates	Male	65 and above	Post Graduation	4712953.0	Good	Prescriptive Drugs	6
2	2018	MH	Maharashtra	Cardiovascular Disease	Nuclear	Male	55-64	High School	704213.0	Bad	Prescriptive Drugs	11
3	2018	RJ	Rajasthan	Chronic Obstructive Pulmonary Disease	Nuclear	Female	25-34	High School	704213.0	Good	Prescriptive Drugs	12
4	2021	UK	Uttarakhand	Arthritis	Nuclear	Female	18-24	High School	704213.0	Good	Prescriptive Drugs	10

In [159]:
```
#!pip install numpy scipy matplotlib seaborn
```

In [160]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skew, kurtosis
```

In [40]:
```
df['Psychological Health'].value_counts()
```

Out[40]:
```
Psychological Health
Bad     123051
Good    110594
Name: count, dtype: int64
```

In [41]:
```
df['Age in years'].value_counts()
```

Out[41]:
```
Age in years
18-24          39323
25-34          39161
45-54          38987
55-64          38831
35-44          38710
65 and above    38633
Name: count, dtype: int64
```

In [42]:
```
df['Psychosocial Factors'].value_counts()
```

Out[42]:
```
Psychosocial Factors
Prescriptive Drugs     111519
NA                      41932
Alcohol                 31594
Alcohol and Smoking     28511
Smoking                 20089
Name: count, dtype: int64
```

In [43]:
```
df['Chronic Diseases'].value_counts()
```

Out[43]:
```
Chronic Diseases
Cardiovascular Disease                  57177
Diabetes                                56222
Chronic Obstructive Pulmonary Disease   50807
Asthma                                  28626
Arthritis                               28088
Chronic Kidney Disease                   8772
Cancer                                   3953
Name: count, dtype: int64
```

In [44]:
```
df['Family Living Status'].value_counts()
```

Out[44]:
```
Family Living Status
Solo            94834
Roomates        68444
Joint Family    35224
Nuclear         35143
Name: count, dtype: int64
```

In [45]:
```
df['Sleep duration(Hrs)'].value_counts()
```

Out[45]:
```
Sleep duration(Hrs)
8     44634
6     41262
11    37370
5     30937
7     29813
9     15467
3      9733
10     7888
12     6458
15     4910
4      4092
2       922
1        68
13       62
14       29
Name: count, dtype: int64
```

In [46]:
```
df['GENDER'].value_counts()
```

Out[46]:
```
GENDER
Male      116852
Female    116793
Name: count, dtype: int64
```

Label Encoding

In [47]:
```
from sklearn.preprocessing import LabelEncoder

# List of columns that need label encoding
label_cols = [
    'Psychosocial Factors','Family Living Status'
]

# Initialize LabelEncoder
le = LabelEncoder()

# Apply LabelEncoder to each column
for col in label_cols:
    df[col] = le.fit_transform(df[col])
```

In [48]:
```
df['Psychological Health'] = df['Psychological Health'].replace({'Bad': 1, 'Good': 0})
```

Hot encoding

In [49]:
```
# Apply one-hot encoding to multiple columns
categorical_columns = [ "Indian States",  "Family Living Status", "GENDER", "Age in years", "Education", "Psychosocial
df = pd.get_dummies(df, columns=categorical_columns)
```

In [50]:
```
df.head(10)
```

Out[50]:

	Year	State Abbreviation	Income	Psychological Health	Sleep duration(Hrs)	Follows a Diet Plan	Obesity / Weight Status	Physical Activity	Indian States_Andaman and Nicobar Islands	Indian States_Andhra Pradesh	...	Frequency of healthcare visits_Monthly	vi
0	2021	PB	2186613.0	0	7	1	0	0	False	False	...	False	
1	2019	PB	4712953.0	0	6	1	0	0	False	False	...	False	
2	2018	MH	704213.0	1	11	1	0	0	False	False	...	False	
3	2018	RJ	704213.0	0	12	1	0	0	False	False	...	False	
4	2021	UK	704213.0	0	10	1	0	0	False	False	...	False	
5	2018	SK	400257.0	0	8	1	0	0	False	False	...	False	
6	2017	PB	1495797.0	1	9	2	0	0	False	False	...	False	
7	2019	SK	804682.0	1	5	1	0	0	False	False	...	True	
8	2017	OR	784668.0	1	8	3	0	0	False	False	...	False	
9	2017	WB	698622.0	0	8	1	0	0	False	False	...	False	

10 rows × 77 columns

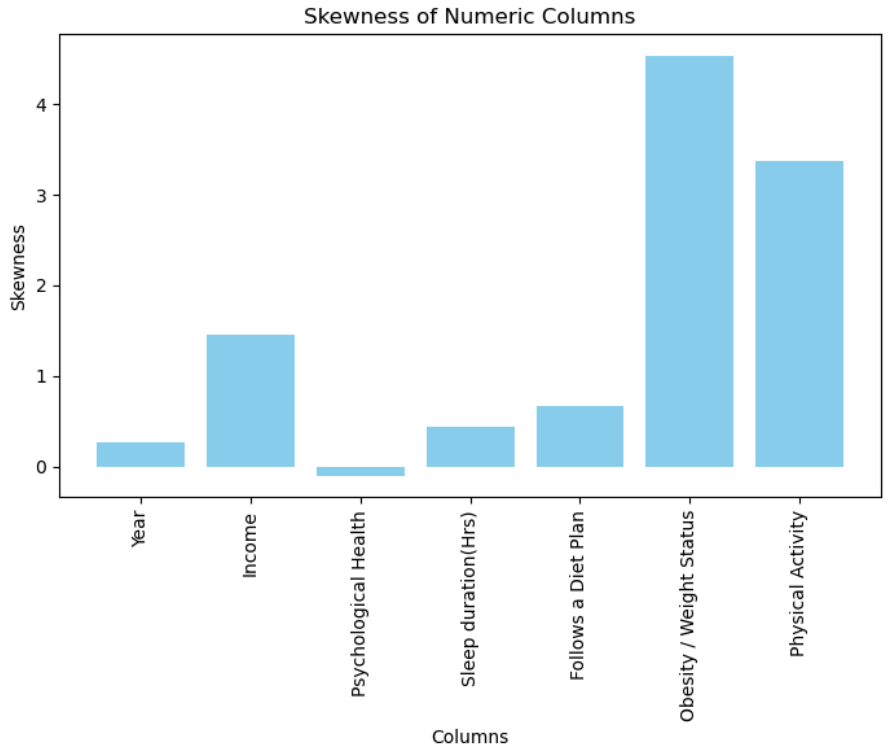
In [184]:
```
#df.info()
```

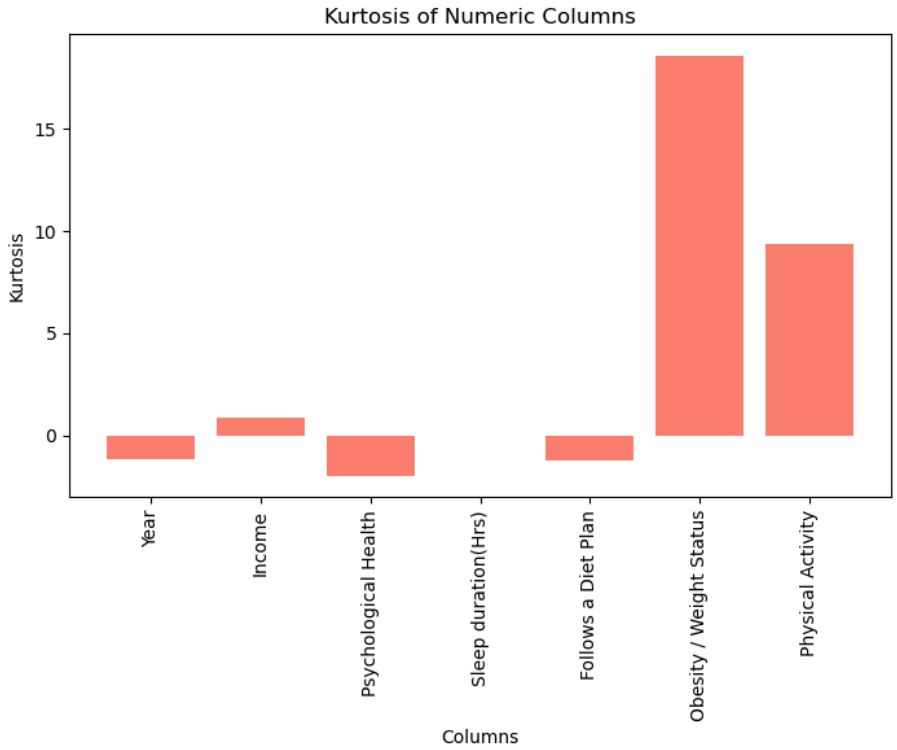
In [54]:
```
import pandas as pd
from scipy.stats import skew, kurtosis

# Assuming df is your DataFrame
# Select only numeric columns
numeric_df = df.select_dtypes(include=[np.number])

# Initialize dictionaries to store results
skewness_results = {}
kurtosis_results = {}

# Calculate skewness and kurtosis for each numeric column
for column in numeric_df.columns:
    skewness_results[column] = skew(numeric_df[column])
    kurtosis_results[column] = kurtosis(numeric_df[column])

# Print the results
for column in numeric_df.columns:
    print(f'Column: {column} - Skewness: {skewness_results[column]}, Kurtosis: {kurtosis_results[column]}')
```

```
Column: Year - Skewness: 0.26680430219148926, Kurtosis: -1.1709937284263394
Column: Income - Skewness: 1.4600874918305562, Kurtosis: 0.8744752181145841
Column: Psychological Health - Skewness: -0.1067837351217476, Kurtosis: -1.9885972339134486
Column: Sleep duration(Hrs) - Skewness: 0.44186879130958473, Kurtosis: -0.028320063221142
Column: Follows a Diet Plan - Skewness: 0.6691688361851331, Kurtosis: -1.2417176430809604
Column: Obesity / Weight Status - Skewness: 4.53842081971628, Kurtosis: 18.597263536834195
Column: Physical Activity - Skewness: 3.368565045459563, Kurtosis: 9.347230465491986
```

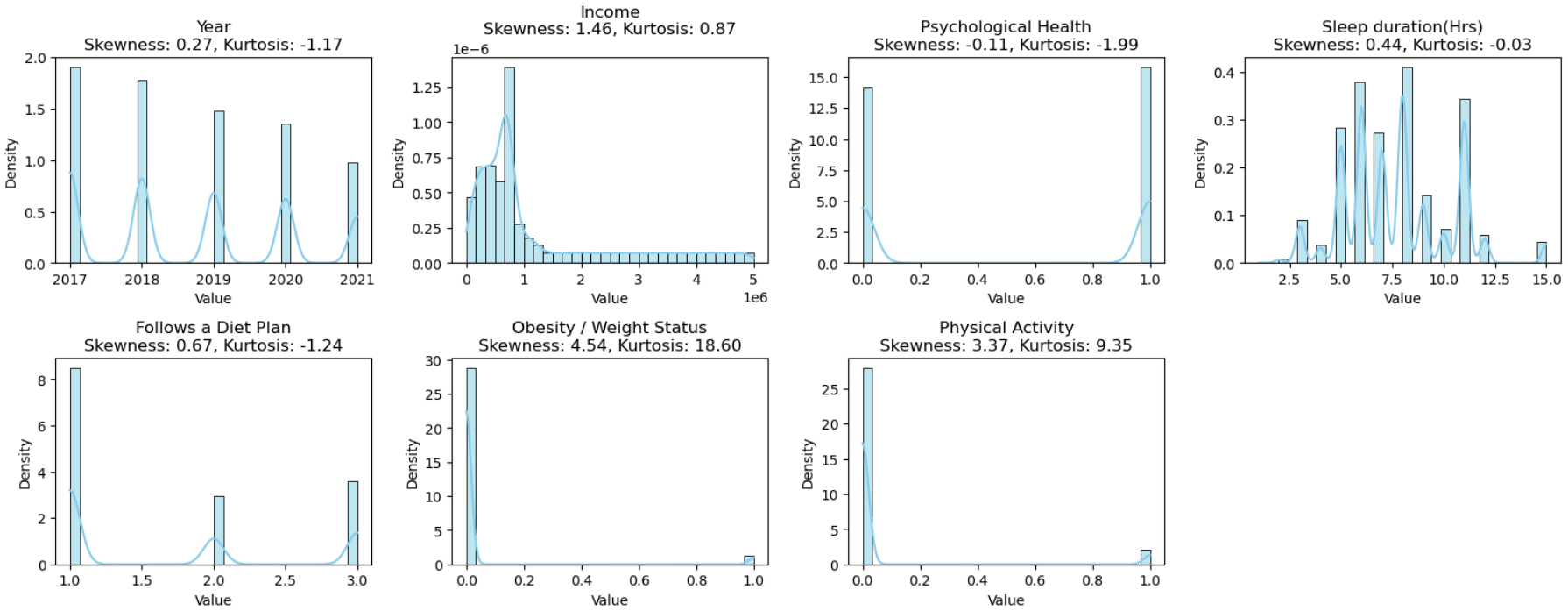
In [55]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import skew, kurtosis

# Assuming df is your DataFrame
# Select only numeric columns
numeric_df = df.select_dtypes(include=[np.number])

# Initialize dictionaries to store results
skewness_results = {}
kurtosis_results = {}

# Calculate skewness and kurtosis for each numeric column
for column in numeric_df.columns:
    skewness_results[column] = skew(numeric_df[column])
    kurtosis_results[column] = kurtosis(numeric_df[column])

# Convert results to DataFrames for easier plotting
skewness_df = pd.DataFrame(list(skewness_results.items()), columns=['Column', 'Skewness'])
kurtosis_df = pd.DataFrame(list(kurtosis_results.items()), columns=['Column', 'Kurtosis'])

# Plotting
plt.figure(figsize=(14, 6))

# Skewness plot
plt.subplot(1, 2, 1)
plt.bar(skewness_df['Column'], skewness_df['Skewness'], color='skyblue')
plt.xticks(rotation=90)
plt.title('Skewness of Numeric Columns')
plt.xlabel('Columns')
plt.ylabel('Skewness')

# Kurtosis plot
plt.subplot(1, 2, 2)
plt.bar(kurtosis_df['Column'], kurtosis_df['Kurtosis'], color='salmon')
plt.xticks(rotation=90)
plt.title('Kurtosis of Numeric Columns')
plt.xlabel('Columns')
plt.ylabel('Kurtosis')
```

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





In [56]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skew, kurtosis

# Assuming df is your DataFrame
# Select only numeric columns
numeric_df = df.select_dtypes(include=[np.number])

# Create a figure
plt.figure(figsize=(16, 12))

# Loop through each numeric column to plot
for i, column in enumerate(numeric_df.columns, 1):
    plt.subplot(4, 4, i)  # Adjust the number of rows and columns as needed
    sns.histplot(numeric_df[column], kde=True, stat="density", color='skyblue', bins=30)

    # Calculate skewness and kurtosis
    col_skewness = skew(numeric_df[column])
    col_kurtosis = kurtosis(numeric_df[column])

    plt.title(f'{column}\nSkewness: {col_skewness:.2f}, Kurtosis: {col_kurtosis:.2f}')
    plt.xlabel('Value')
    plt.ylabel('Density')

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



In [57]:
```
## df.to_excel('cleandata_with_labelencoding.xlsx', index=False)

# # Confirm the file has been saved
## print("DataFrame has been saved to 'cleandata_with_labelencoding.xlsx'")
```

In [58]:
```
import os
# Get the current working directory
## print(os.getcwd())
```

MODEL BUILDING

In [59]:
```
# Drop columns directly
df = df.drop(columns=['State Abbreviation'])
```

In [60]:
```
X = df.drop('Psychological Health', axis=1)
y = df['Psychological Health']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Logistic Regression

In [93]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Logistic Regression
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
y_pred_lr = log_reg.predict(X_test)
y_proba_lr = log_reg.predict_proba(X_test)[:, 1]
```

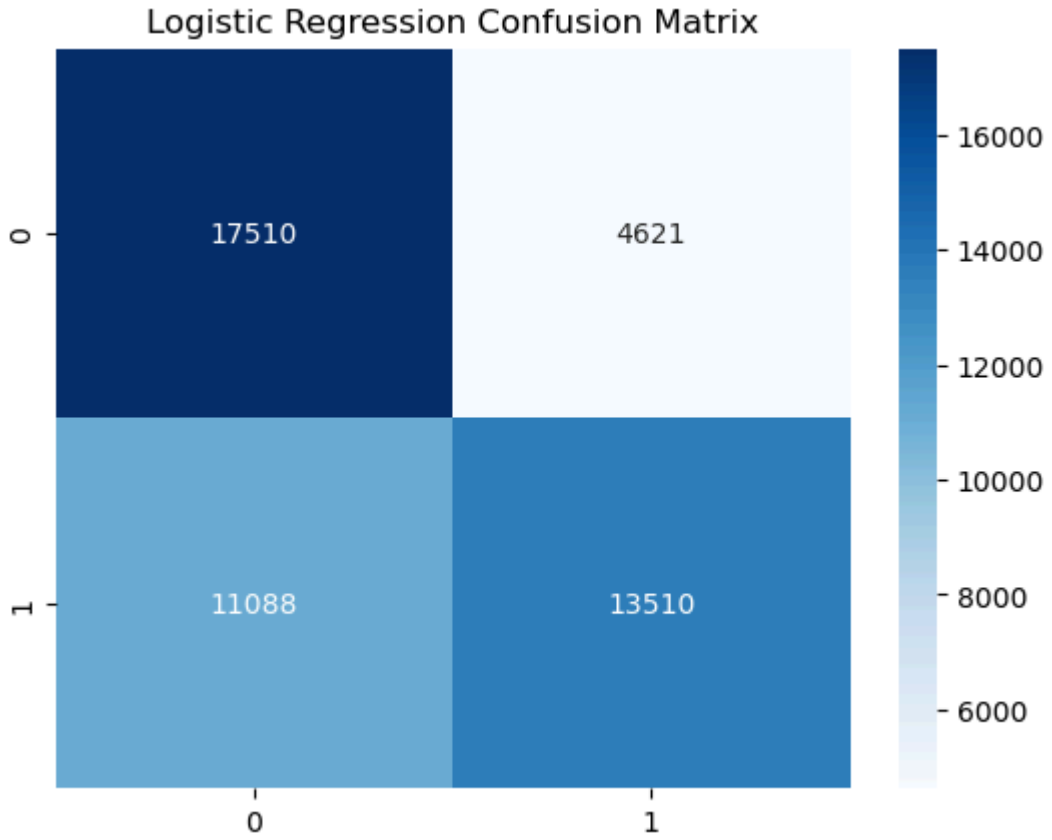
In [94]:
```
# Accuracy
accuracy_lr = accuracy_score(y_test, y_pred_lr)
print(f'Logistic Regression Accuracy: {accuracy_lr}')
```

```
Logistic Regression Accuracy: 0.663827601703439
```

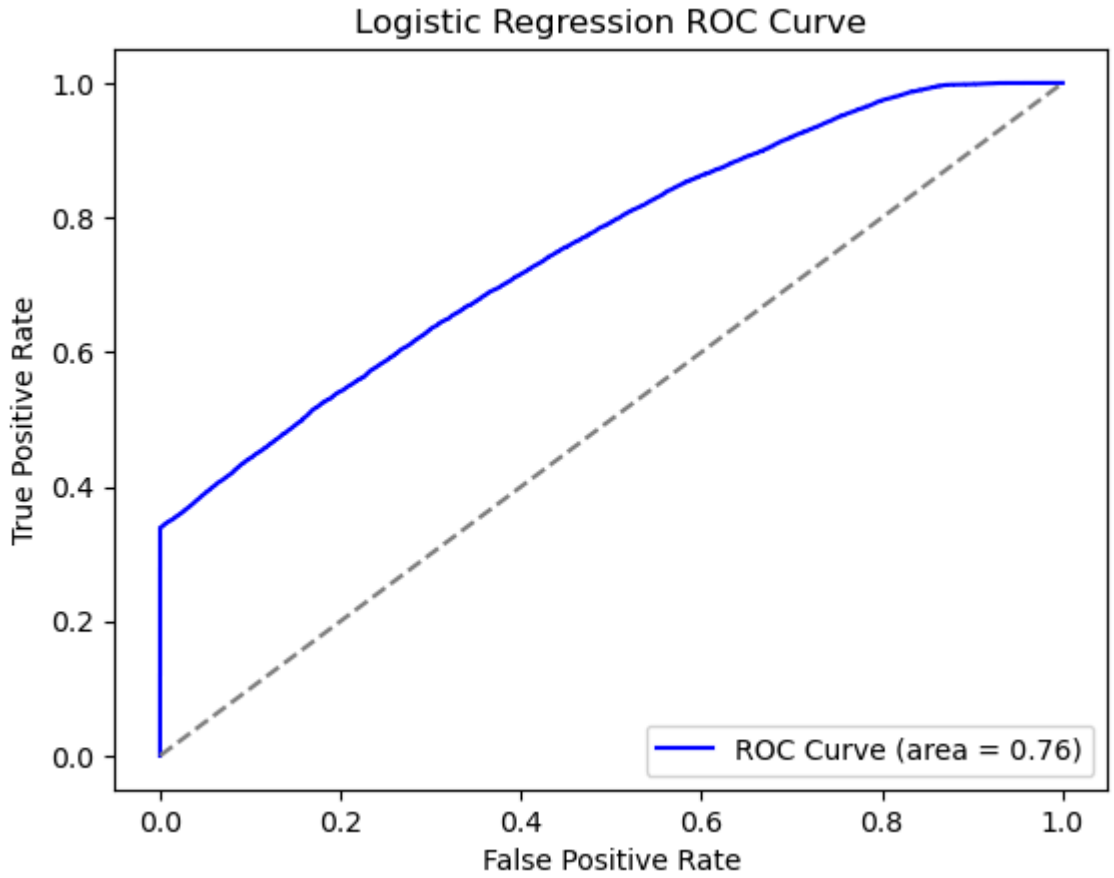
In [105]:
```
# Classification Report
print("Logistic Regression Classification Report:")
print(classification_report(y_test, y_pred_lr))
```

```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.61      0.79      0.69     22131
           1       0.75      0.55      0.63     24598

    accuracy                           0.66     46729
   macro avg       0.68      0.67      0.66     46729
weighted avg       0.68      0.66      0.66     46729
```

In [95]:
```
# Confusion Matrix
cm_lr = confusion_matrix(y_test, y_pred_lr)
sns.heatmap(cm_lr, annot=True, fmt='d', cmap='Blues')
plt.title('Logistic Regression Confusion Matrix')
plt.show()
```



In [96]:
```
# ROC Curve
fpr_lr, tpr_lr, _ = roc_curve(y_test, y_proba_lr)
roc_auc_lr = auc(fpr_lr, tpr_lr)
plt.plot(fpr_lr, tpr_lr, color='blue', label=f'ROC Curve (area = {roc_auc_lr:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.title('Logistic Regression ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.show()
```



Decision Tree

In [99]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

# Decision Tree
dt_clf = DecisionTreeClassifier(random_state=42)
dt_clf.fit(X_train, y_train)
y_pred_dt = dt_clf.predict(X_test)
y_proba_dt = dt_clf.predict_proba(X_test)[:, 1]
```

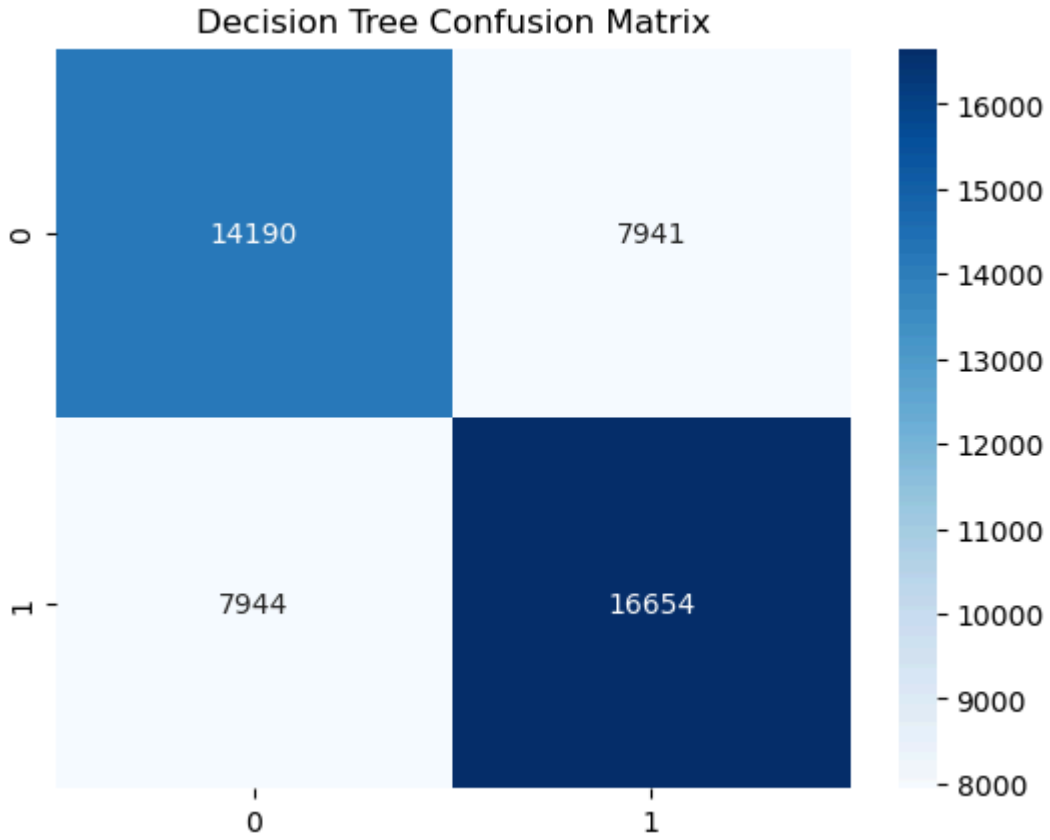
In [100]:
```
# Accuracy
accuracy_dt = accuracy_score(y_test, y_pred_dt)
print(f'Decision Tree Accuracy: {accuracy_dt}')
```

```
Decision Tree Accuracy: 0.6600612039632776
```

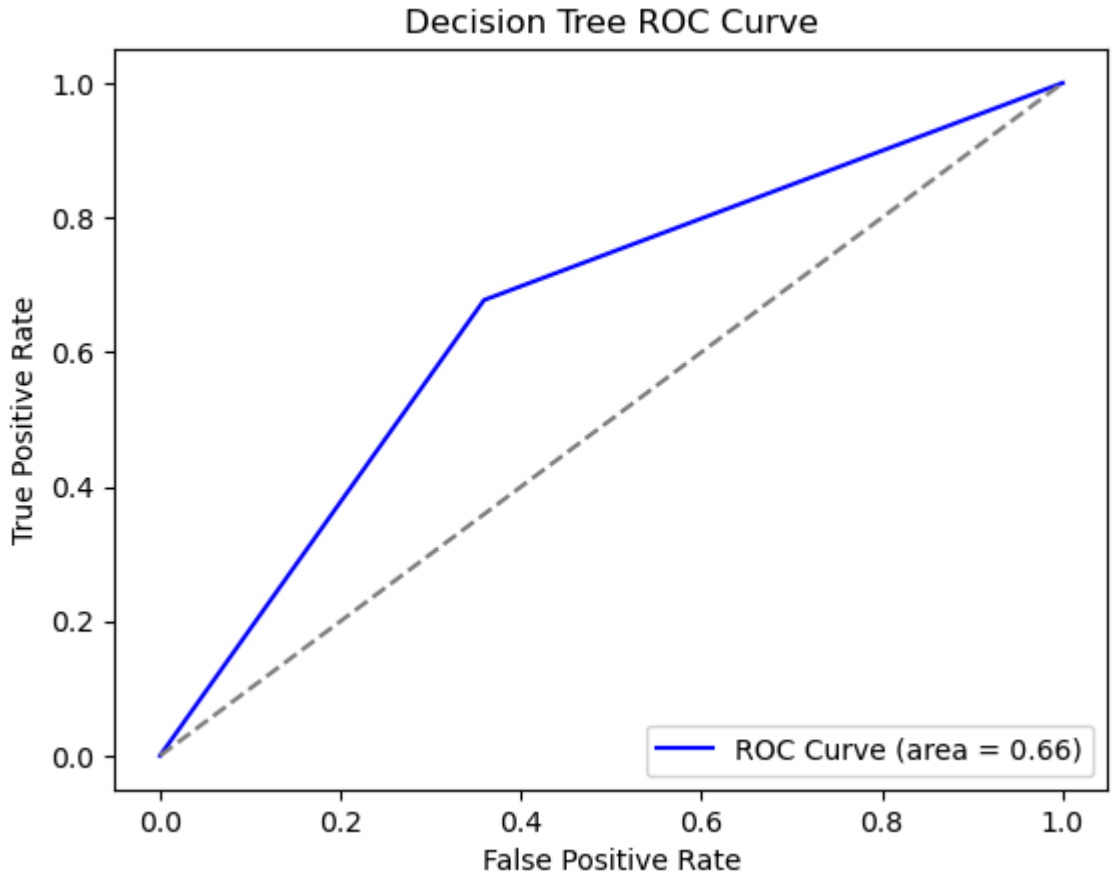
In [104]:
```
# Classification Report
print("Decision Tree Classification Report:")
print(classification_report(y_test, y_pred_dt))
```

```
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       0.64      0.64      0.64     22131
           1       0.68      0.68      0.68     24598

    accuracy                           0.66     46729
   macro avg       0.66      0.66      0.66     46729
weighted avg       0.66      0.66      0.66     46729
```

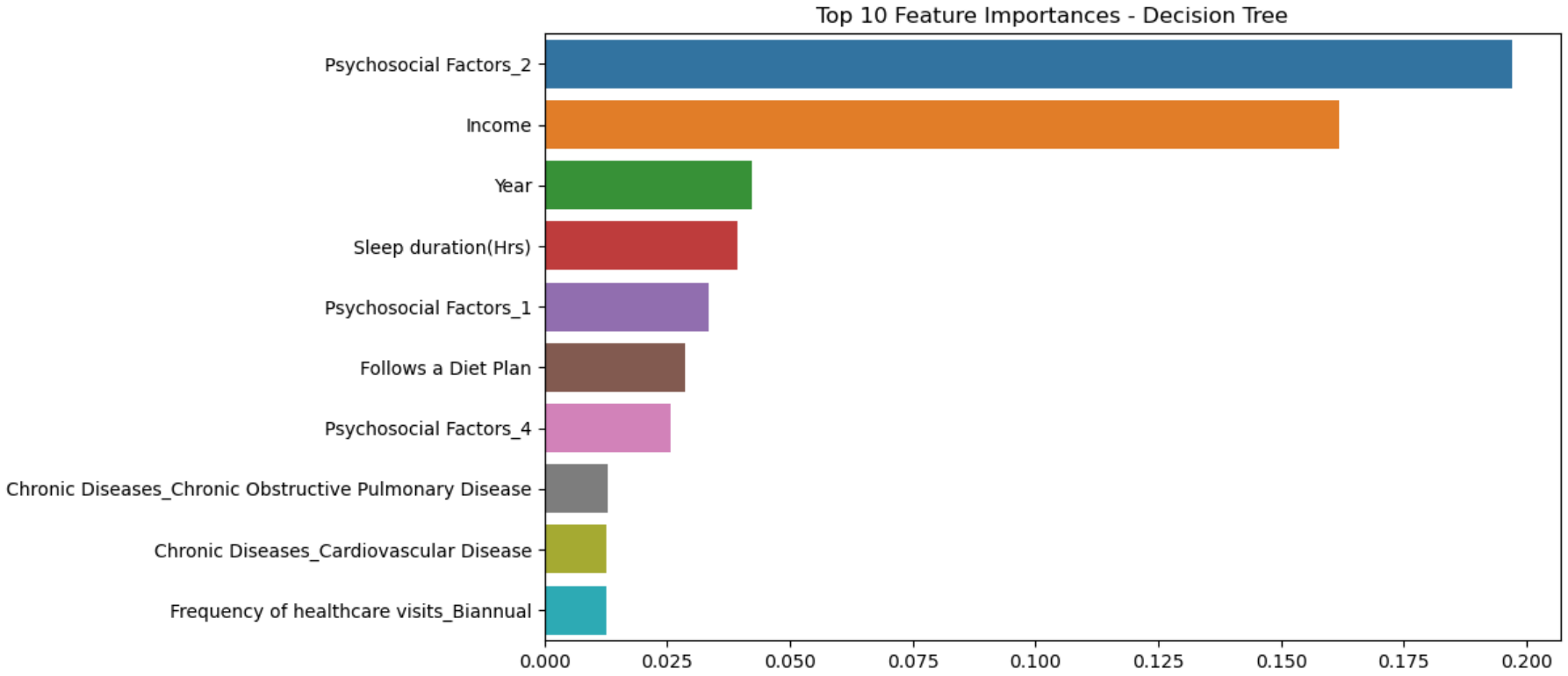
In [101]:
```
# Confusion Matrix
cm_dt = confusion_matrix(y_test, y_pred_dt)
sns.heatmap(cm_dt, annot=True, fmt='d', cmap='Blues')
plt.title('Decision Tree Confusion Matrix')
plt.show()
```



In [102]:
```
# ROC Curve
fpr_dt, tpr_dt, _ = roc_curve(y_test, y_proba_dt)
roc_auc_dt = auc(fpr_dt, tpr_dt)
plt.plot(fpr_dt, tpr_dt, color='blue', label=f'ROC Curve (area = {roc_auc_dt:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.title('Decision Tree ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.show()
```



In [103]:
```
# Feature Importance
importances = dt_clf.feature_importances_
indices = np.argsort(importances)[::-1]
features = X.columns

plt.figure(figsize=(10, 6))
sns.barplot(x=importances[indices][:10], y=features[indices][:10])
plt.title('Top 10 Feature Importances - Decision Tree')
plt.show()
```



Random Forest

In [106]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Random Forest
rf_clf = RandomForestClassifier(random_state=42)
rf_clf.fit(X_train, y_train)
y_pred_rf = rf_clf.predict(X_test)
y_proba_rf = rf_clf.predict_proba(X_test)[:, 1]
```

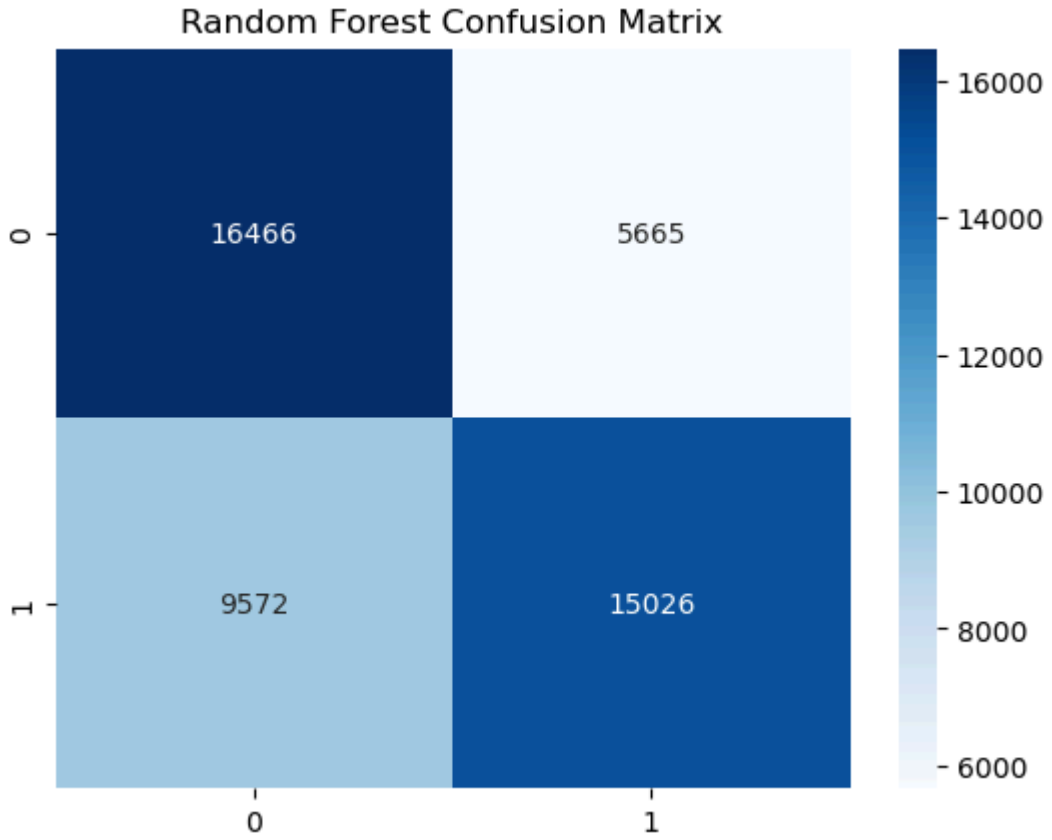
In [107]:
```
# Accuracy
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print(f'Random Forest Accuracy: {accuracy_rf}')
```

```
Random Forest Accuracy: 0.6739283956429626
```

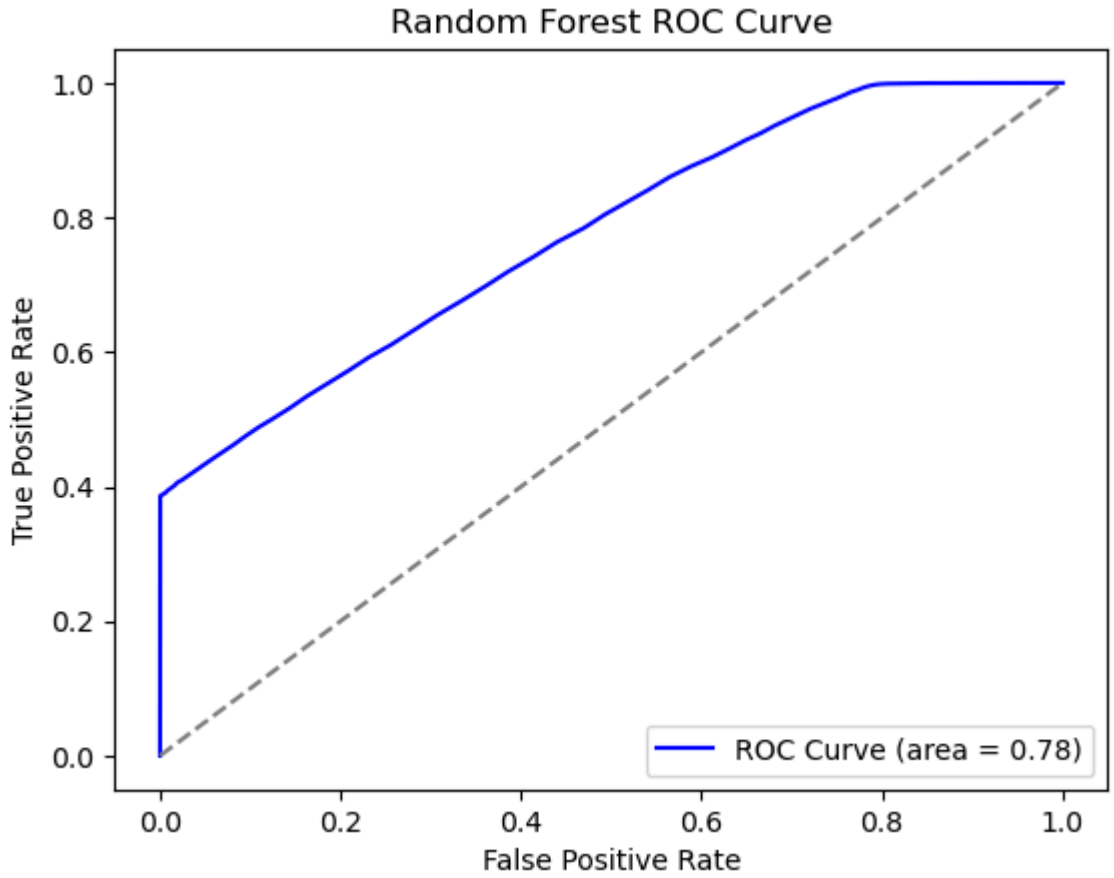
In [108]:
```
# Classification Report
print("Random Forest Classification Report:")
print(classification_report(y_test, y_pred_rf))
```

```
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.74      0.68     22131
           1       0.73      0.61      0.66     24598

    accuracy                           0.67     46729
   macro avg       0.68      0.68      0.67     46729
weighted avg       0.68      0.67      0.67     46729
```

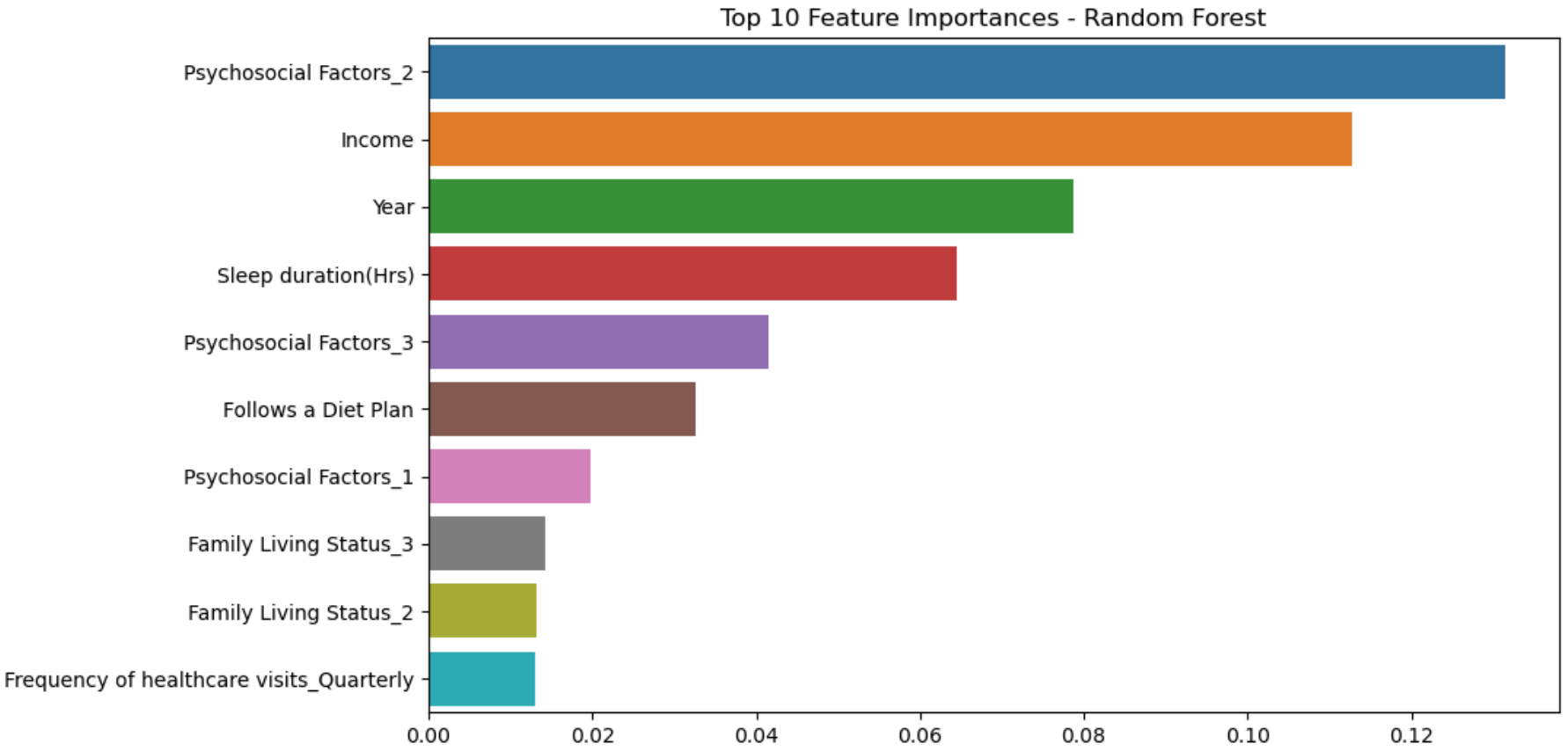
In [109]:
```
# Confusion Matrix
cm_rf = confusion_matrix(y_test, y_pred_rf)
sns.heatmap(cm_rf, annot=True, fmt='d', cmap='Blues')
plt.title('Random Forest Confusion Matrix')
plt.show()
```



In [110]:
```
# ROC Curve
fpr_rf, tpr_rf, _ = roc_curve(y_test, y_proba_rf)
roc_auc_rf = auc(fpr_rf, tpr_rf)
plt.plot(fpr_rf, tpr_rf, color='blue', label=f'ROC Curve (area = {roc_auc_rf:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.title('Random Forest ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.show()
```



In [111]:
```
# Feature Importance
importances = rf_clf.feature_importances_
indices = np.argsort(importances)[::-1]
features = X.columns

plt.figure(figsize=(10, 6))
sns.barplot(x=importances[indices][:10], y=features[indices][:10])
plt.title('Top 10 Feature Importances - Random Forest')
plt.show()
```



XGBoost (Extreme Gradient Boosting)

In [112]:
```
from sklearn.ensemble import GradientBoostingClassifier

# XGBoost
xgb_clf = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
xgb_clf.fit(X_train, y_train)
y_pred_xgb = xgb_clf.predict(X_test)
y_proba_xgb = xgb_clf.predict_proba(X_test)[:, 1]
```

```
C:\Users\Ritik\anaconda3\ane\Lib\site-packages\xgboost\core.py:158: UserWarning: [23:43:10] WARNING: C:\buildkite-ag
ent\builds\buildkite-windows-cpu-autoscaling-group-i-0015a694724fa8361-1\xgboost\xgboost-ci-windows\src\learner.cc:7
40:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

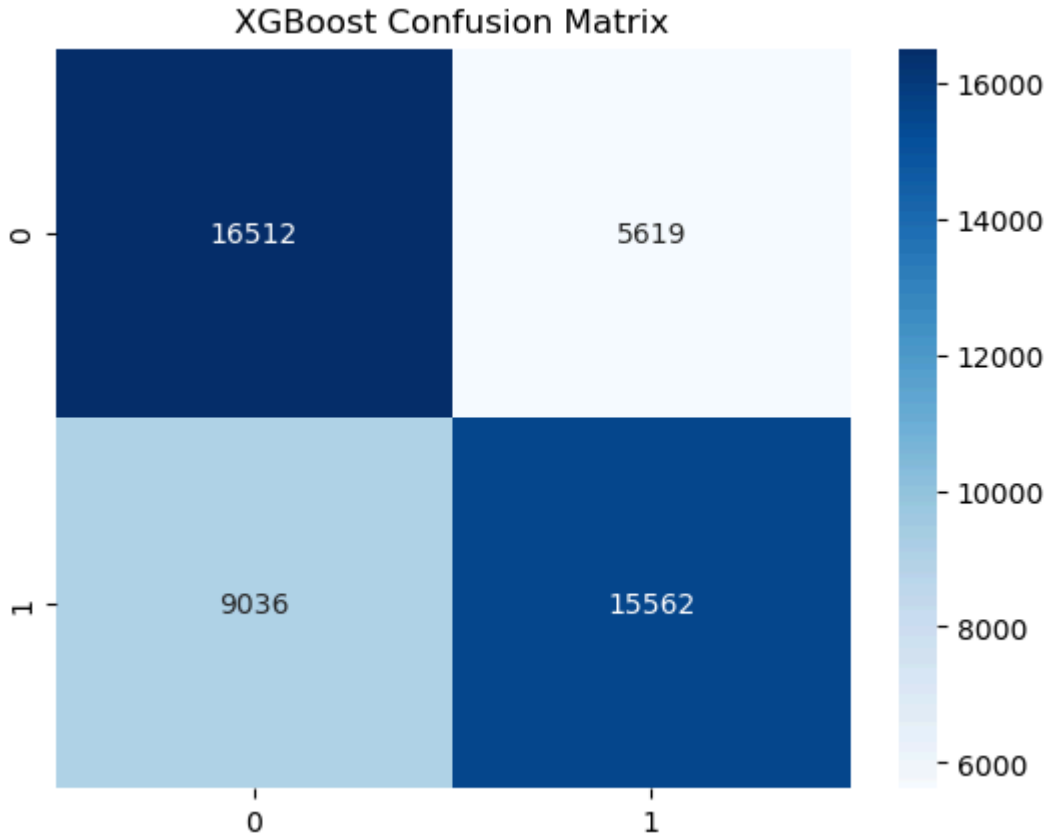
In [113]:
```
# Accuracy
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
print(f'XGBoost Accuracy: {accuracy_xgb}')
```

```
XGBoost Accuracy: 0.6863831881700871
```

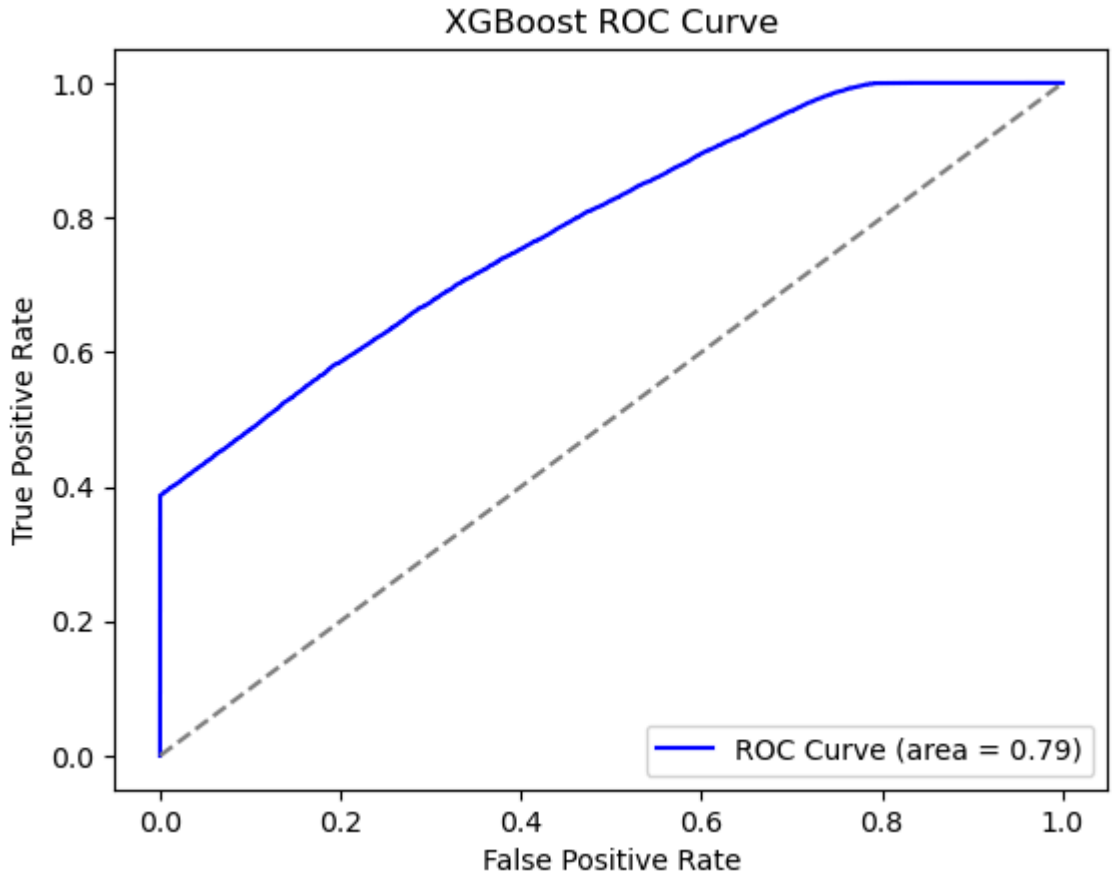
In [114]:
```
# Classification Report
print("XGBoost Classification Report:")
print(classification_report(y_test, y_pred_xgb))
```

```
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.75      0.69     22131
           1       0.73      0.63      0.68     24598

    accuracy                           0.69     46729
   macro avg       0.69      0.69      0.69     46729
weighted avg       0.69      0.69      0.69     46729
```

In [115]:
```
# Confusion Matrix
cm_xgb = confusion_matrix(y_test, y_pred_xgb)
sns.heatmap(cm_xgb, annot=True, fmt='d', cmap='Blues')
plt.title('XGBoost Confusion Matrix')
plt.show()
```



In [116]:
```
# ROC Curve
fpr_xgb, tpr_xgb, _ = roc_curve(y_test, y_proba_xgb)
roc_auc_xgb = auc(fpr_xgb, tpr_xgb)
plt.plot(fpr_xgb, tpr_xgb, color='blue', label=f'ROC Curve (area = {roc_auc_xgb:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.title('XGBoost ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.show()
```



In [117]:
```
# Feature Importance
importances = xgb_clf.feature_importances_
indices = np.argsort(importances)[::-1]
features = X.columns

plt.figure(figsize=(10, 6))
sns.barplot(x=importances[indices][:10], y=features[indices][:10])
plt.title('Top 10 Feature Importances - XGBoost')
plt.show()
```