In [23]:
```
# Load the data
import pandas as pd

file_path = 'standardized_health_data.csv'
data = pd.read_csv(file_path)

print(data.head())
print(data['Disease_Type'].value_counts())
```

```
  Age_Group  Gender       Race Smoking Alcohol_Consumption  \
0     31-45    Male  Caucasian   Never          Occasional
1       61+    Male  Caucasian  Former               Never
2     46-60  Female  Caucasian   Never               Never
3     46-60    Male   Hispanic   Never          Occasional
4     18-30  Female  Caucasian   Never          Occasional

         Education_Level Insurance_Type Exercise_Frequency Diet_Quality  \
0  Less than high school      Uninsured     1-2 times/week         Fair
1  Less than high school        Private              Never         Poor
2           Some college       Medicaid     1-2 times/week         Poor
3        Graduate degree        Private              Never         Poor
4           Some college      Uninsured     3-4 times/week         Fair

  Sleep_Hours Stress_Level         Occupation Family_History BMI_Category  \
0         6-7         High      Manual Labor             No       Normal
1         7-8       Medium      Manual Labor             No       Normal
2          8+          Low  Service Industry             No   Overweight
3         7-8       Medium   Office/Desk Job            Yes       Normal
4         6-7       Medium  Service Industry            Yes   Overweight

        Income_Level       Disease_Type
0  Less than $30,000  Colorectal Cancer
1   $75,000-$100,000           Diabetes
2    $30,000-$50,000           Diabetes
3    $30,000-$50,000           Diabetes
4    $30,000-$50,000        Sleep Apnea
Chronic Respiratory Disease    1013
Sleep Apnea                    1003
Obesity                         975
Lung Cancer                     515
Diabetes                        510
Colorectal Cancer               485
Hypertension                    219
Heart Disease                   148
Depression                       69
Arthritis                        63
Name: Disease_Type, dtype: int64
```

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

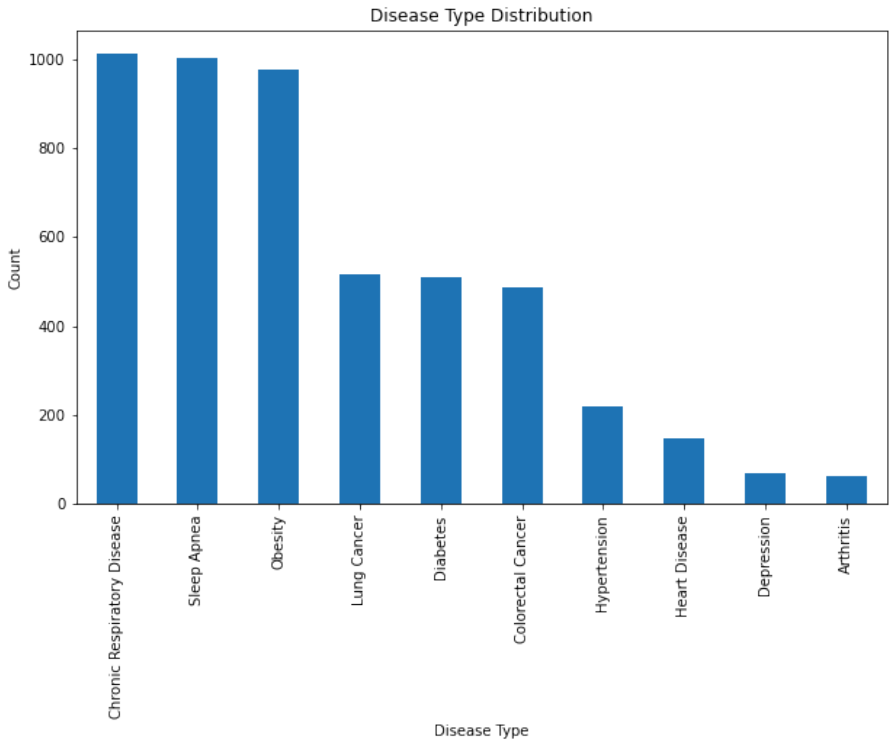
Out[24]:

	Age_Group	Gender	Race	Smoking	Alcohol_Consumption	Education_Level	Insurance_Type	Exercise_Frequency	Diet_Quality	Sleep_Hours	Stress_Level	Occupation	Family_History	BMI_Category	Inc
0	31-45	Male	Caucasian	Never	Occasional	Less than high school	Uninsured	1-2 times/week	Fair	6-7	High	Manual Labor	No	Normal	
1	61+	Male	Caucasian	Former	Never	Less than high school	Private	Never	Poor	7-8	Medium	Manual Labor	No	Normal	
2	46-60	Female	Caucasian	Never	Never	Some college	Medicaid	1-2 times/week	Poor	8+	Low	Service Industry	No	Overweight	
3	46-60	Male	Hispanic	Never	Occasional	Graduate degree	Private	Never	Poor	7-8	Medium	Office/Desk Job	Yes	Normal	
4	18-30	Female	Caucasian	Never	Occasional	Some college	Uninsured	3-4 times/week	Fair	6-7	Medium	Service Industry	Yes	Overweight	

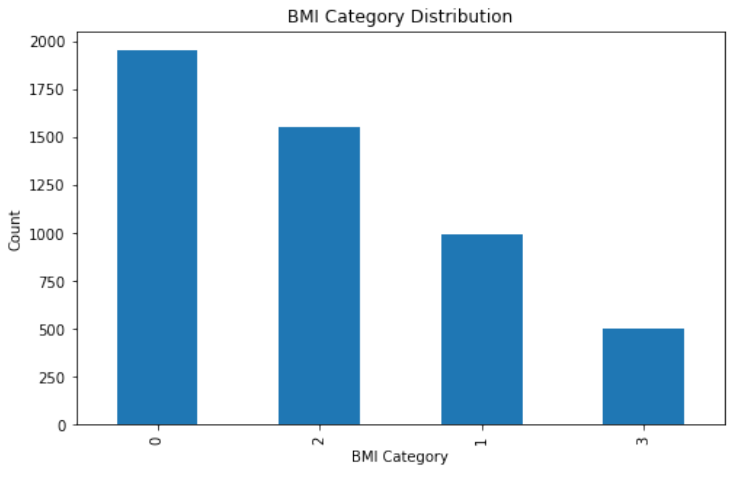
In [25]:
```
# check for missing values
print("missing value: ")
print(data.isnull().sum())

# check data type
print("\ndata type: ")
print(data.dtypes)


from sklearn.preprocessing import LabelEncoder

categorical_features = ['Age_Group', 'Gender', 'Race', 'Smoking', 'Alcohol_Consumption',
                        'Education_Level', 'Insurance_Type', 'Exercise_Frequency', 'Diet_Quality',
                        'Sleep_Hours', 'Stress_Level', 'Occupation', 'Family_History', 'BMI_Category']
encoder = LabelEncoder()

for feature in categorical_features:
    data[feature] = encoder.fit_transform(data[feature])

print(data.head())
```

```
missing value:
Age_Group              0
Gender                 0
Race                   0
Smoking                0
Alcohol_Consumption    0
Education_Level        0
Insurance_Type         0
Exercise_Frequency     0
Diet_Quality           0
Sleep_Hours            0
Stress_Level           0
Occupation             0
Family_History         0
BMI_Category           0
Income_Level           0
Disease_Type           0
dtype: int64

data type:
Age_Group              object
Gender                 object
Race                   object
Smoking                object
Alcohol_Consumption    object
Education_Level        object
Insurance_Type         object
Exercise_Frequency     object
Diet_Quality           object
Sleep_Hours            object
Stress_Level           object
Occupation             object
Family_History         object
BMI_Category           object
Income_Level           object
Disease_Type           object
dtype: object
   Age_Group  Gender  Race  Smoking  Alcohol_Consumption  Education_Level  \
0          1       1     2        2                    1                3
1          3       1     2        1                    0                3
2          2       0     2        2                    0                4
3          2       1     3        2                    1                1
4          0       0     2        2                    1                4

   Insurance_Type  Exercise_Frequency  Diet_Quality  Sleep_Hours  \
0               3                   0             1            0
1               2                   3             3            1
2               0                   0             3            2
3               2                   3             3            1
4               3                   1             1            0

   Stress_Level  Occupation  Family_History  BMI_Category       Income_Level  \
0             0           0               0             0  Less than $30,000
1             2           0               0             0   $75,000-$100,000
2             1           3               0             2    $30,000-$50,000
3             2           1               1             0    $30,000-$50,000
4             2           3               1             2    $30,000-$50,000

       Disease_Type
0  Colorectal Cancer
1           Diabetes
2           Diabetes
3           Diabetes
4        Sleep Apnea
```

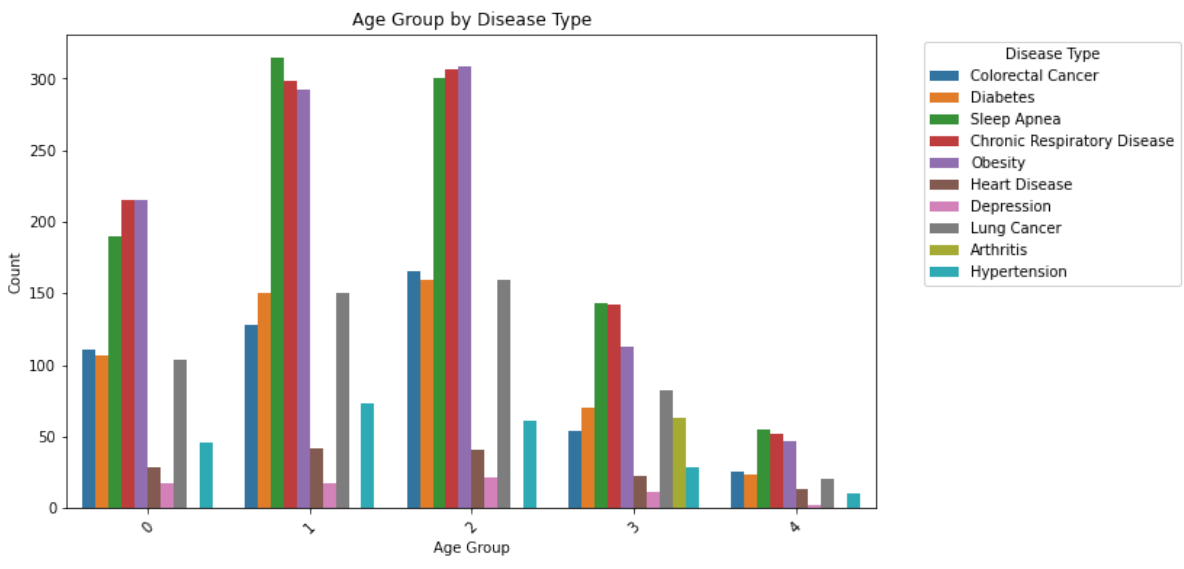
In [26]:
```
# Disease_Type 分布
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
data['Disease_Type'].value_counts().plot(kind='bar')
plt.title('Disease Type Distribution')
plt.xlabel('Disease Type')
plt.ylabel('Count')
plt.show()
```



In [27]:
```
plt.figure(figsize=(8, 5))
data['BMI_Category'].value_counts().plot(kind='bar')
plt.title('BMI Category Distribution')
plt.xlabel('BMI Category')
plt.ylabel('Count')
plt.show()
```

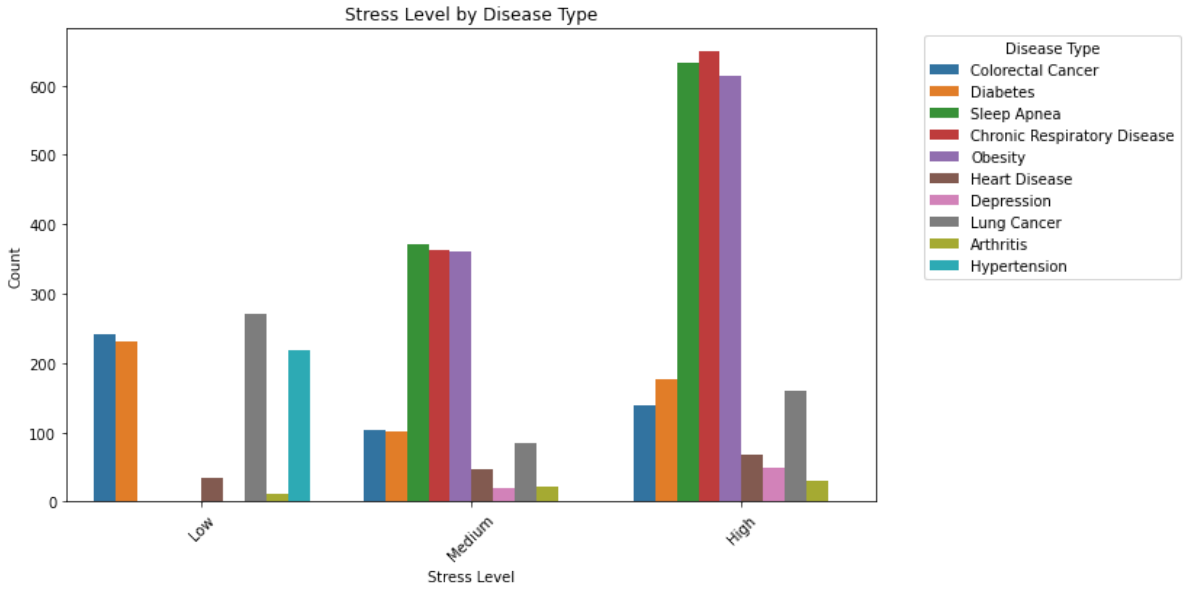


In [28]:
```
plt.figure(figsize=(10, 6))
sns.countplot(data=data, x='Age_Group', hue='Disease_Type')
plt.title('Age Group by Disease Type')
plt.xlabel('Age Group')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend(title='Disease Type', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



In [29]:

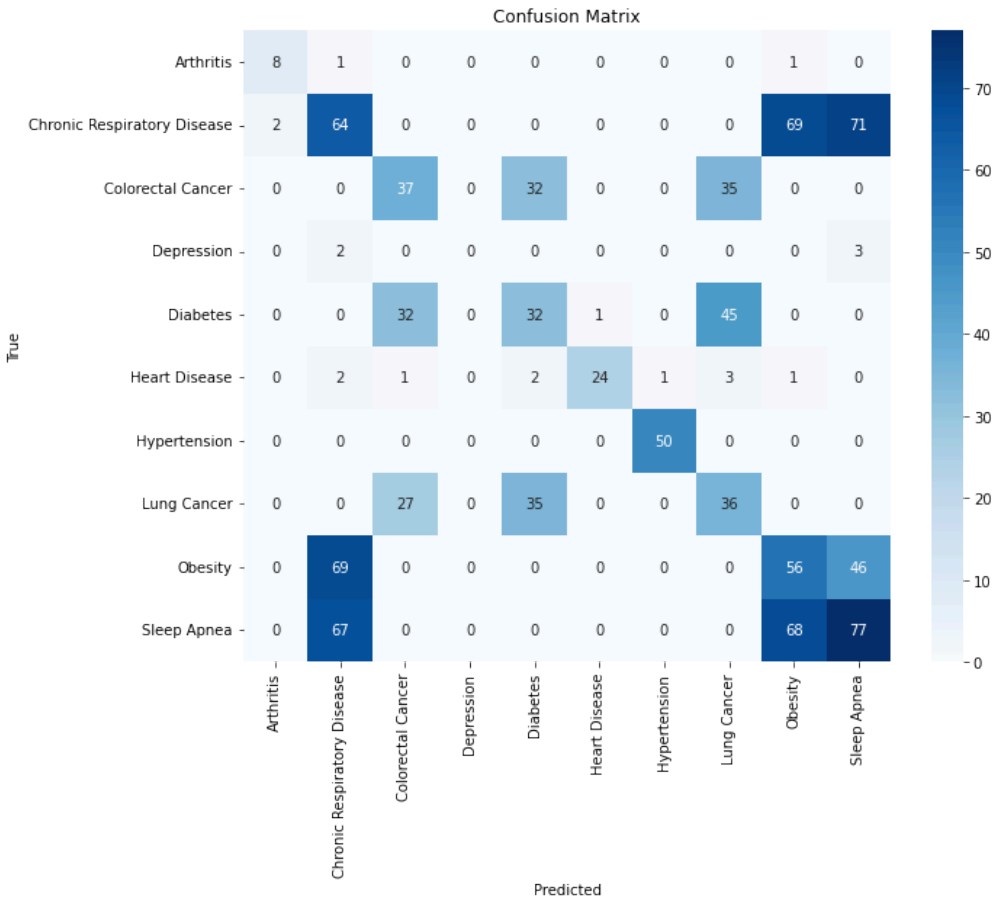
```
stress_mapping = {0: 'Low', 1: 'Medium', 2: 'High'}
data['Stress_Level'] = data['Stress_Level'].map(stress_mapping)


plt.figure(figsize=(10, 6))
sns.countplot(data=data, x='Stress_Level', hue='Disease_Type', order=['Low', 'Medium', 'High'])
plt.title('Stress Level by Disease Type')
plt.xlabel('Stress Level')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend(title='Disease Type', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



In [30]:

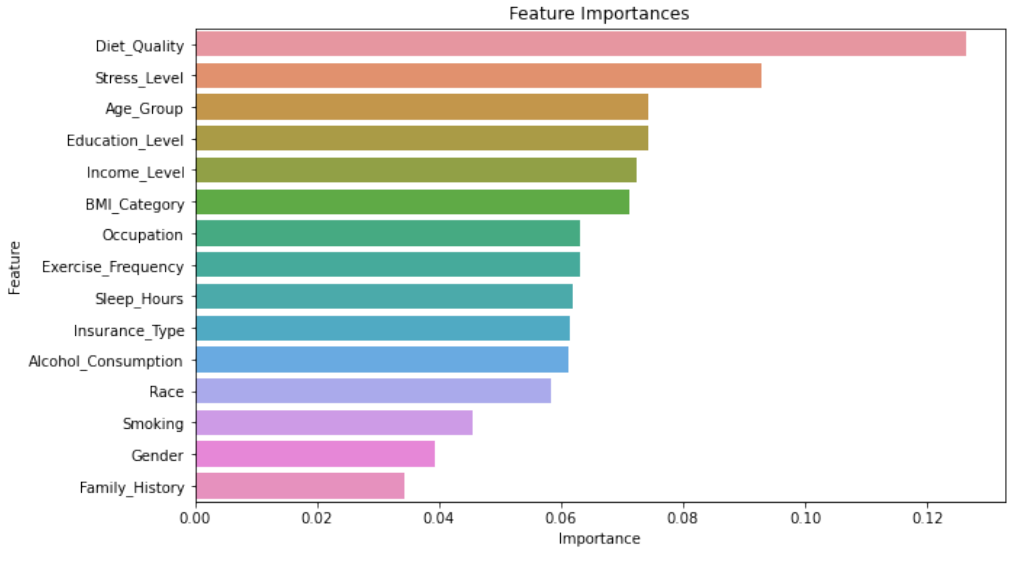
```
# Step 1: Dataset Preparation
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Define features and target variable
features = ['Age_Group', 'Gender', 'Race', 'Smoking', 'Alcohol_Consumption',
            'Education_Level', 'Insurance_Type', 'Exercise_Frequency',
            'Diet_Quality', 'Sleep_Hours', 'Stress_Level', 'Occupation',
            'Family_History', 'BMI_Category', 'Income_Level']
X = data[features]
y = data['Disease_Type']

# Encode features and target variable
label_encoder = LabelEncoder()
X_encoded = X.apply(label_encoder.fit_transform)  # Encode features
y_encoded = label_encoder.fit_transform(y)       # Encode target variable

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y_encoded, test_size=0.2, random_state=42)

print("Training set size:", X_train.shape)
print("Test set size:", X_test.shape)

# Step 2: Model Training and Prediction
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Define Random Forest model
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

# Predict on test set
y_pred = rf_model.predict(X_test)

# Step 3: Model Evaluation
# Print classification report
print("Test set accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=label_encoder.classes_))

# Visualize confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

# Step 4: Feature Importance Analysis
feature_importances = rf_model.feature_importances_
importance_df = pd.DataFrame({'Feature': features, 'Importance': feature_importances}).sort_values(by='Importance', ascending=False)

print("\nFeature Importances:")
print(importance_df)

# Visualize feature importances
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=importance_df)
plt.title('Feature Importances')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```

```
Training set size: (4000, 15)
Test set size: (1000, 15)
Test set accuracy: 0.384

Classification Report:
                              precision    recall  f1-score   support

                   Arthritis       0.80      0.80      0.80        10
Chronic Respiratory Disease       0.31      0.31      0.31       206
          Colorectal Cancer       0.38      0.36      0.37       104
                 Depression       0.00      0.00      0.00         5
                   Diabetes       0.32      0.29      0.30       110
              Heart Disease       0.96      0.71      0.81        34
               Hypertension       0.98      1.00      0.99        50
                Lung Cancer       0.30      0.37      0.33        98
                    Obesity       0.29      0.33      0.31       171
                Sleep Apnea       0.39      0.36      0.38       212

                   accuracy                           0.38      1000
                  macro avg       0.47      0.45      0.46      1000
               weighted avg       0.39      0.38      0.39      1000


C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



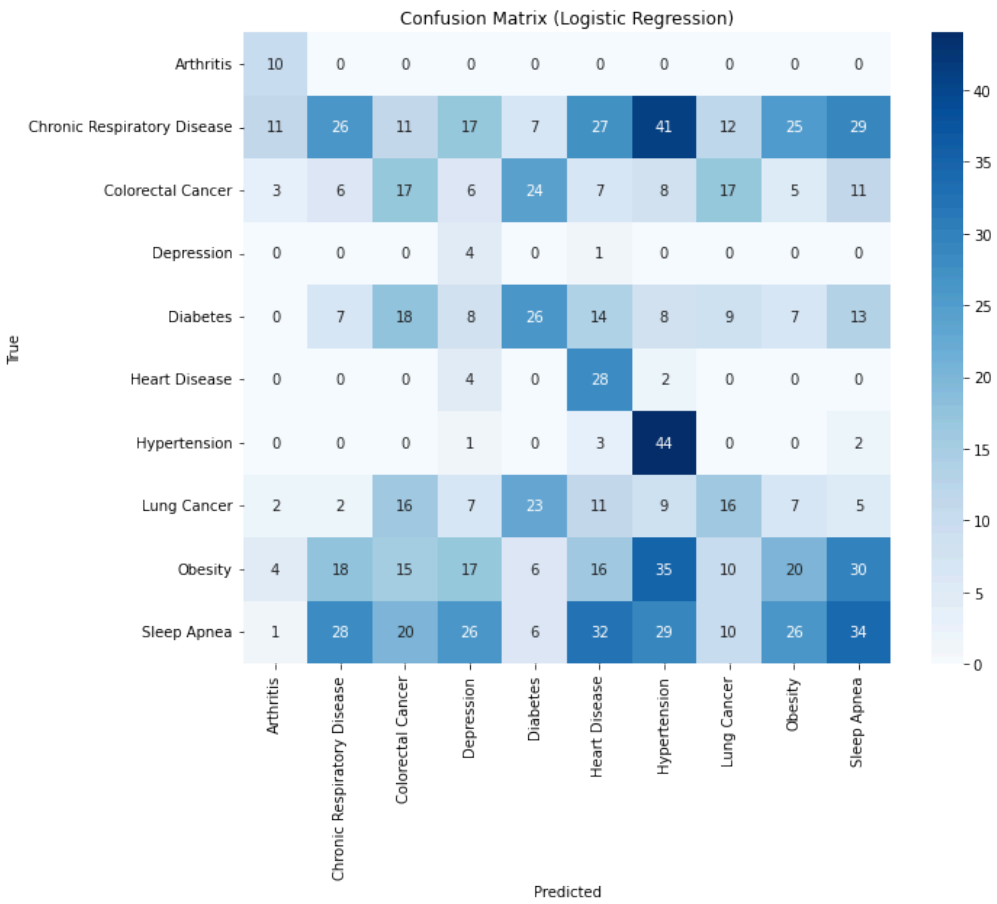
Feature Importances:

	Feature	Importance
8	Diet_Quality	0.126381
10	Stress_Level	0.092838
0	Age_Group	0.074347
5	Education_Level	0.074307
14	Income_Level	0.072473
13	BMI_Category	0.071241
11	Occupation	0.063097
7	Exercise_Frequency	0.063081
9	Sleep_Hours	0.062048
6	Insurance_Type	0.061360
4	Alcohol_Consumption	0.061159
2	Race	0.058314
3	Smoking	0.045526
1	Gender	0.039431
12	Family_History	0.034397



In [31]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Logistic Regression Model
logistic_model = LogisticRegression(max_iter=1000, multi_class='multinomial', class_weight='balanced', random_state=42)
logistic_model.fit(X_train, y_train)

# Predict on the test set
y_pred_logistic = logistic_model.predict(X_test)

# Model Evaluation
print("Logistic Regression Test Accuracy:", accuracy_score(y_test, y_pred_logistic))
print("\nClassification Report:")
print(classification_report(y_test, y_pred_logistic, target_names=label_encoder.classes_))

# Confusion Matrix
conf_matrix_logistic = confusion_matrix(y_test, y_pred_logistic)
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix_logistic, annot=True, fmt='d', cmap='Blues', xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.title('Confusion Matrix (Logistic Regression)')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
Logistic Regression Test Accuracy: 0.225

Classification Report:
                          precision    recall  f1-score   support

                Arthritis       0.32      1.00      0.49        10
Chronic Respiratory Disease       0.30      0.13      0.18       206
         Colorectal Cancer       0.18      0.16      0.17       104
                Depression       0.04      0.80      0.08         5
                 Diabetes       0.28      0.24      0.26       110
            Heart Disease       0.20      0.82      0.32        34
             Hypertension       0.25      0.88      0.39        50
              Lung Cancer       0.22      0.16      0.19        98
                  Obesity       0.22      0.12      0.15       171
              Sleep Apnea       0.27      0.16      0.20       212

                 accuracy                           0.23      1000
                macro avg       0.23      0.45      0.24      1000
             weighted avg       0.25      0.23      0.21      1000
```



In [32]:
```
# Map Disease_Type to Risk Levels
risk_mapping = {
    'Arthritis': 'Low Risk',
    'Depression': 'Low Risk',
    'Obesity': 'Medium Risk',
    'Hypertension': 'Medium Risk',
    'Sleep Apnea': 'Medium Risk',
    'Heart Disease': 'High Risk',
    'Lung Cancer': 'High Risk',
    'Chronic Respiratory Disease': 'High Risk',
    'Diabetes': 'Medium Risk',
    'Colorectal Cancer': 'High Risk'
}

data['Health_Risk'] = data['Disease_Type'].map(risk_mapping)

# Check the distribution of the new target variable
print(data['Health_Risk'].value_counts())
```

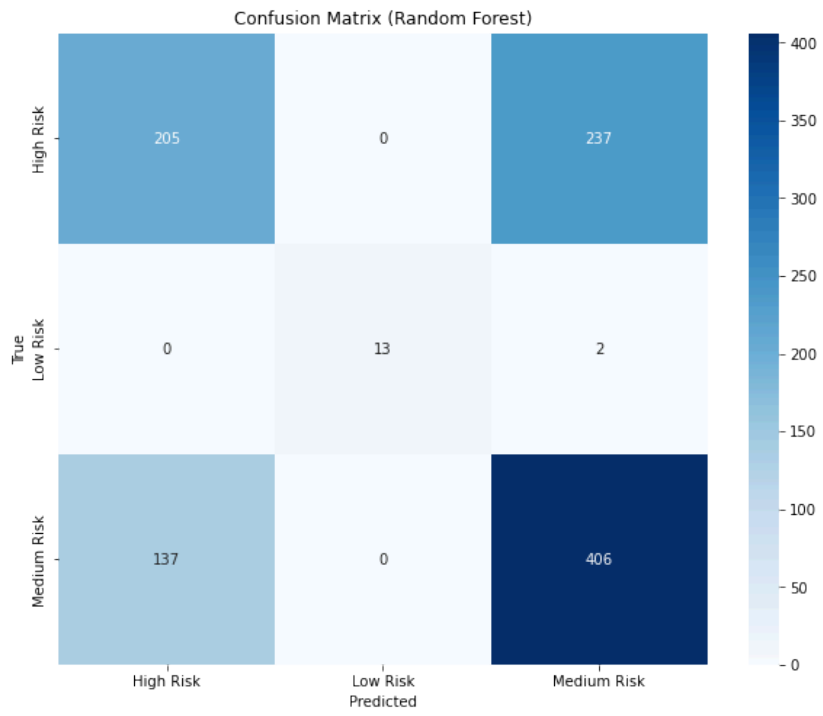
```
Medium Risk    2707
High Risk      2161
Low Risk        132
Name: Health_Risk, dtype: int64
```

In [33]:
```
# New target variable
y = data['Health_Risk']

# Encode the new target variable
y_encoded = label_encoder.fit_transform(y)

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y_encoded, test_size=0.2, random_state=42)

# Logistic Regression Model
logistic_model = LogisticRegression(max_iter=1000, class_weight='balanced', random_state=42)
logistic_model.fit(X_train, y_train)

# Predictions
y_pred_logistic = logistic_model.predict(X_test)

# Model Evaluation
print("Logistic Regression Test Accuracy:", accuracy_score(y_test, y_pred_logistic))
print("\nClassification Report:")
print(classification_report(y_test, y_pred_logistic, target_names=label_encoder.classes_))
```

```
Logistic Regression Test Accuracy: 0.459

Classification Report:
              precision    recall  f1-score   support

   High Risk       0.54      0.45      0.49       442
    Low Risk       0.06      0.93      0.11        15
 Medium Risk       0.62      0.45      0.52       543

    accuracy                           0.46      1000
   macro avg       0.41      0.61      0.38      1000
weighted avg       0.58      0.46      0.50      1000
```

In [34]:
```
# Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(class_weight='balanced', random_state=42)
rf_model.fit(X_train, y_train)

# Predictions
y_pred_rf = rf_model.predict(X_test)

# Model Evaluation
print("Random Forest Test Accuracy:", accuracy_score(y_test, y_pred_rf))
print("\nClassification Report:")
print(classification_report(y_test, y_pred_rf, target_names=label_encoder.classes_))

# Plot Confusion Matrix
conf_matrix_rf = confusion_matrix(y_test, y_pred_rf)
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix_rf, annot=True, fmt='d', cmap='Blues', xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.title('Confusion Matrix (Random Forest)')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
Random Forest Test Accuracy: 0.624

Classification Report:
              precision    recall  f1-score   support

   High Risk       0.60      0.46      0.52       442
    Low Risk       1.00      0.87      0.93        15
 Medium Risk       0.63      0.75      0.68       543

    accuracy                           0.62      1000
   macro avg       0.74      0.69      0.71      1000
weighted avg       0.62      0.62      0.62      1000
```



In [ ]:

In [ ]: