```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from statsmodels.distributions.empirical_distribution import ECDF

# Function to display comprehensive univariate analysis for
categorical variables
def categorical_univariate_analysis(feature_name, data_series):
    # Frequency Distribution
    frequency_distribution = data_series.value_counts()

    # Display results
    print(f"\n------ Univariate Analysis for {feature_name} ------")
    print(f"Frequency Distribution:\n{frequency_distribution}\n")

    # Visualization
    plt.figure(figsize=(8, 5))
    sns.countplot(x=data_series,
order=data_series.value_counts().index)  # Added order parameter
    plt.title(f'{feature_name} Distribution')
    plt.show()

def numerical_univariate_analysis(feature_name, data_series):
    # Descriptive Statistics
    descriptive_stats = data_series.describe()

    # Measures of Central Tendency
    mean_value = data_series.mean()
    median_value = data_series.median()
    mode_value = data_series.mode().iloc[0]

    # Measures of Dispersion
    std_deviation = data_series.std()
    range_value = data_series.max() - data_series.min()
    variance_value = data_series.var()

    # Percentiles and Quartiles
    percentiles = np.percentile(data_series, [25, 50, 75])
    quartiles = {'Q1': percentiles[0], 'Q2': percentiles[1], 'Q3':
percentiles[2]}

    # Display results
    print(f"\n------ Univariate Analysis for {feature_name} ------")
    print(f"Descriptive Statistics:\n{descriptive_stats}\n")
    print(f"Measures of Central Tendency:")
    print(f"Mean: {mean_value}")
    print(f"Median: {median_value}")
```

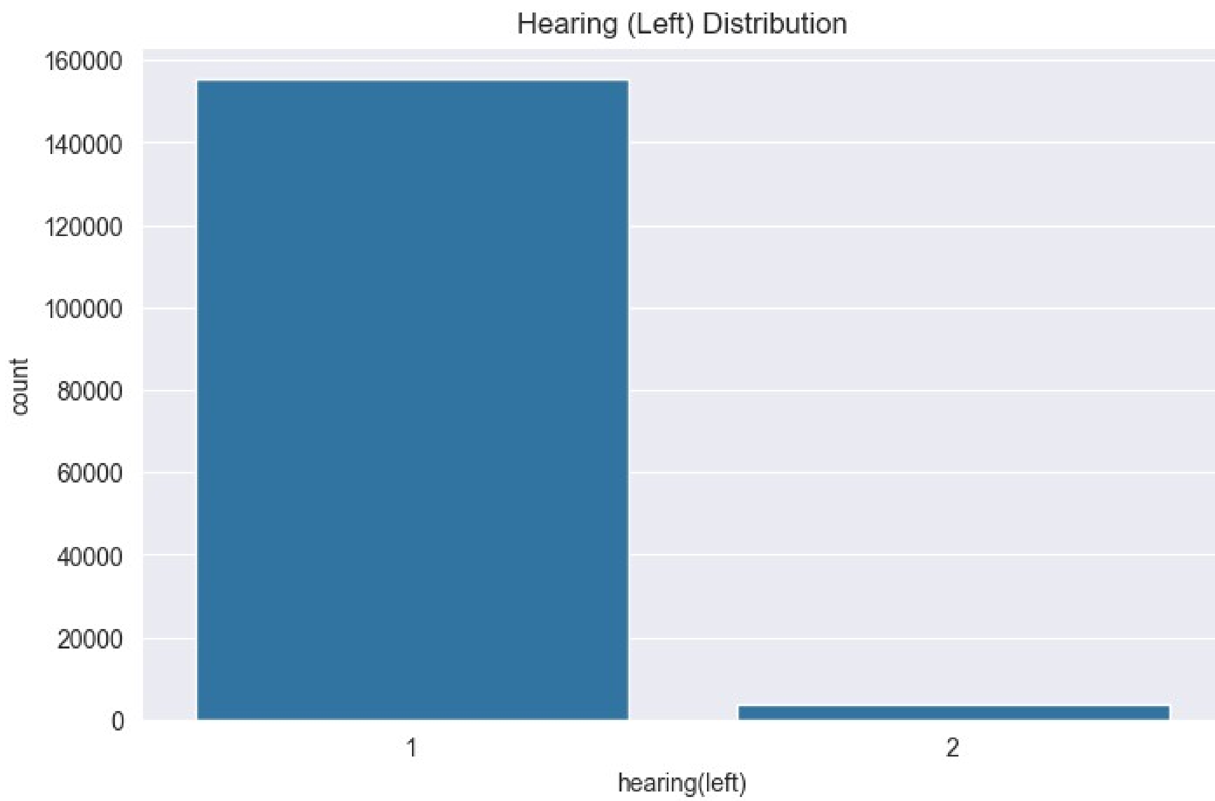
```
    print(f"Mode: {mode_value}\n")
    print(f"Measures of Dispersion:")
    print(f"Standard Deviation: {std_deviation}")
    print(f"Range: {range_value}")
    print(f"Variance: {variance_value}\n")
    print(f"Percentiles and Quartiles:")
    print(f"Q1 (25th percentile): {percentiles[0]}")
    print(f"Q2 (50th percentile - Median): {percentiles[1]}")
    print(f"Q3 (75th percentile): {percentiles[2]}")
    print(f"Interquartile Range (IQR): {percentiles[2] -
percentiles[0]}\n")

    # Visualizations
    plt.figure(figsize=(12, 6))

    # Histogram
    plt.subplot(2, 2, 1)
    sns.histplot(data_series)
    plt.title(f'{feature_name} Distribution (Histogram)')

    # KDE Plot
    plt.subplot(2, 2, 2)
    sns.histplot(data_series, kde=True, color='orange', bins=50,
alpha=0.7)  # Increase bins and add transparency
    plt.title(f'{feature_name} Distribution with KDE')

    # Box Plot
    plt.subplot(2, 2, 3)
    sns.boxplot(x=data_series, color='green')
    plt.title(f'{feature_name} Box Plot')

    # ECDF Plot
    plt.subplot(2, 2, 4)
    ecdf = ECDF(data_series)
    plt.plot(ecdf.x, ecdf.y, marker='o', linestyle='-', color='red')
    plt.title(f'ECDF of {feature_name}')

    plt.tight_layout()
    plt.show()

data_path = 'my_data.csv'

# Read the dataset
df = pd.read_csv(data_path)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159256 entries, 0 to 159255
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype
```

```
---  ------          --------------  -----
 0   id              159256 non-null  int64
 1   ALT             159256 non-null  int64
 2   AST             159256 non-null  int64
 3   hearing(left)   159256 non-null  int64
 4   weight(kg)      159256 non-null  int64
 5   hearing(right)  159256 non-null  int64
 6   relaxation      159256 non-null  int64
 7   waist(cm)       159256 non-null  float64
 8   Cholesterol     159256 non-null  int64
 9   HDL             159256 non-null  int64
 10  systolic        159256 non-null  int64
 11  smoking         159256 non-null  int64
dtypes: float64(1), int64(11)
memory usage: 14.6 MB
```

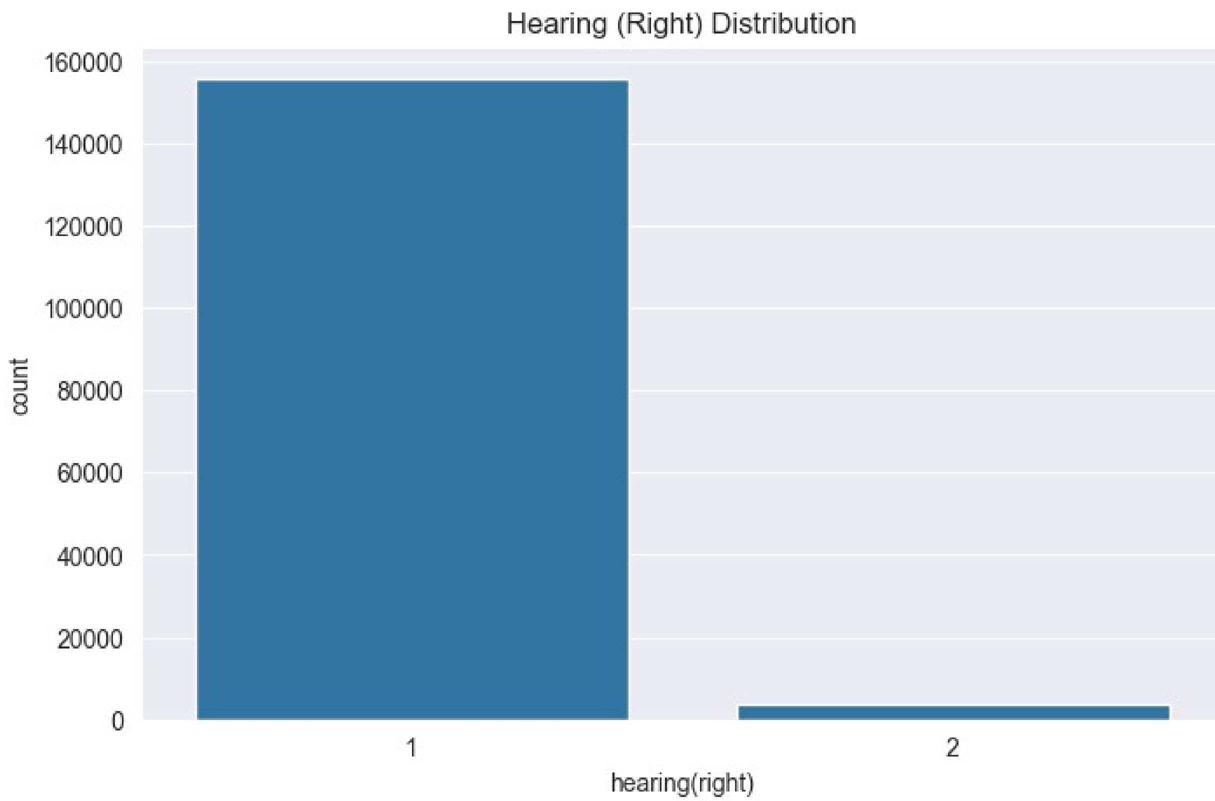
```python
# Apply the functions to each feature
categorical_univariate_analysis('Hearing (Left)', df['hearing(left)'])
```

```
------ Univariate Analysis for Hearing (Left) ------
Frequency Distribution:
hearing(left)
1    155438
2      3818
Name: count, dtype: int64
```




```
categorical_univariate_analysis('Hearing (Right)',
df['hearing(right)'])
```

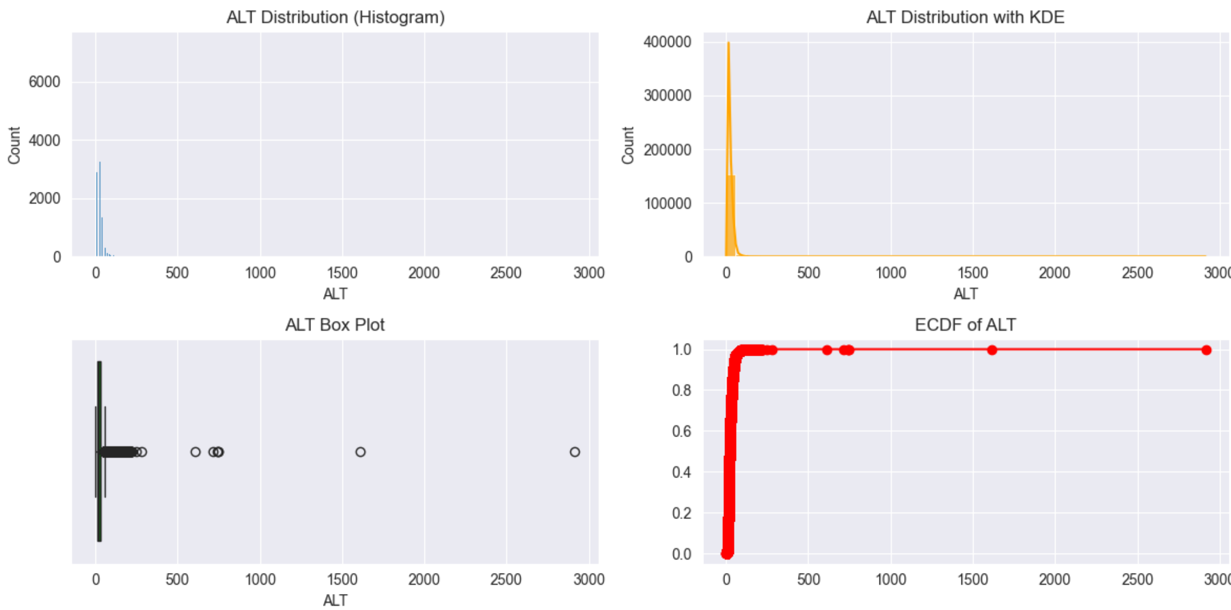
```
------ Univariate Analysis for Hearing (Right) ------
Frequency Distribution:
hearing(right)
1    155526
2      3730
Name: count, dtype: int64
```




```
numerical_univariate_analysis('ALT', df['ALT'])
```

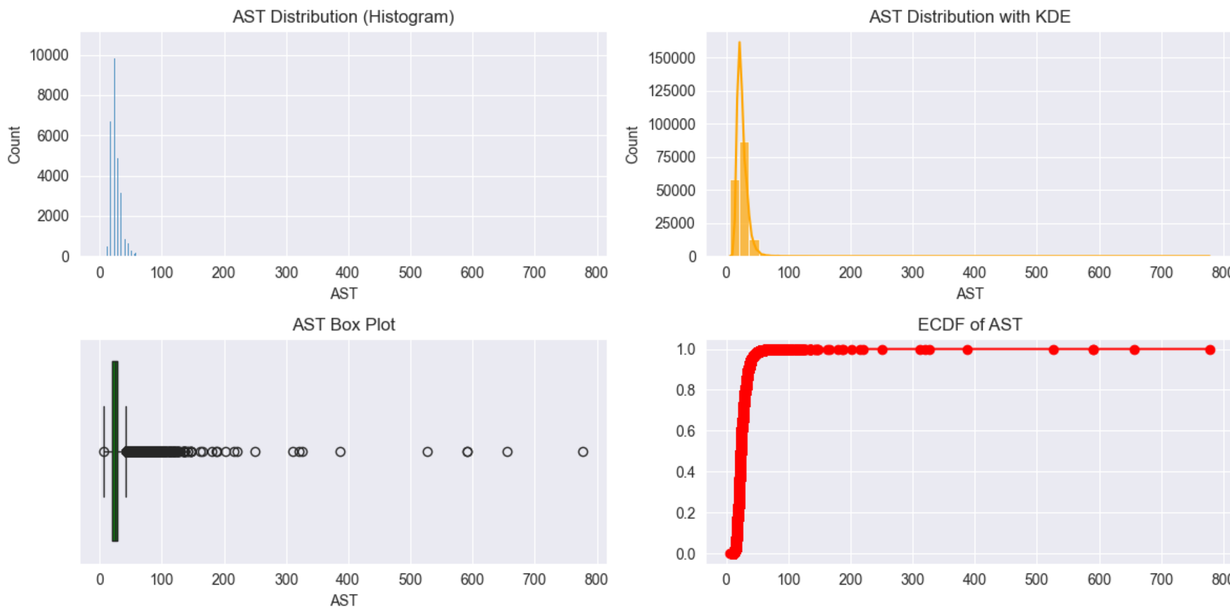
```
------ Univariate Analysis for ALT ------
Descriptive Statistics:
count    159256.000000
mean         26.550296
std          17.753070
min           1.000000
25%          16.000000
50%          22.000000
75%          32.000000
max        2914.000000
Name: ALT, dtype: float64

Measures of Central Tendency:
Mean: 26.550296378158436
Median: 22.0
Mode: 15

Measures of Dispersion:
Standard Deviation: 17.753070138185393
Range: 2913
Variance: 315.17149933132987
```

```
Percentiles and Quartiles:
Q1 (25th percentile): 16.0
Q2 (50th percentile - Median): 22.0
Q3 (75th percentile): 32.0
Interquartile Range (IQR): 16.0
```




```
numerical_univariate_analysis('AST', df['AST'])


------ Univariate Analysis for AST ------
Descriptive Statistics:
count    159256.000000
mean         25.516853
std           9.464882
min           6.000000
25%          20.000000
50%          24.000000
75%          29.000000
max         778.000000
Name: AST, dtype: float64

Measures of Central Tendency:
Mean: 25.516853368161954
Median: 24.0
Mode: 20

Measures of Dispersion:
Standard Deviation: 9.464882078029072
Range: 772
Variance: 89.58399275099592
```

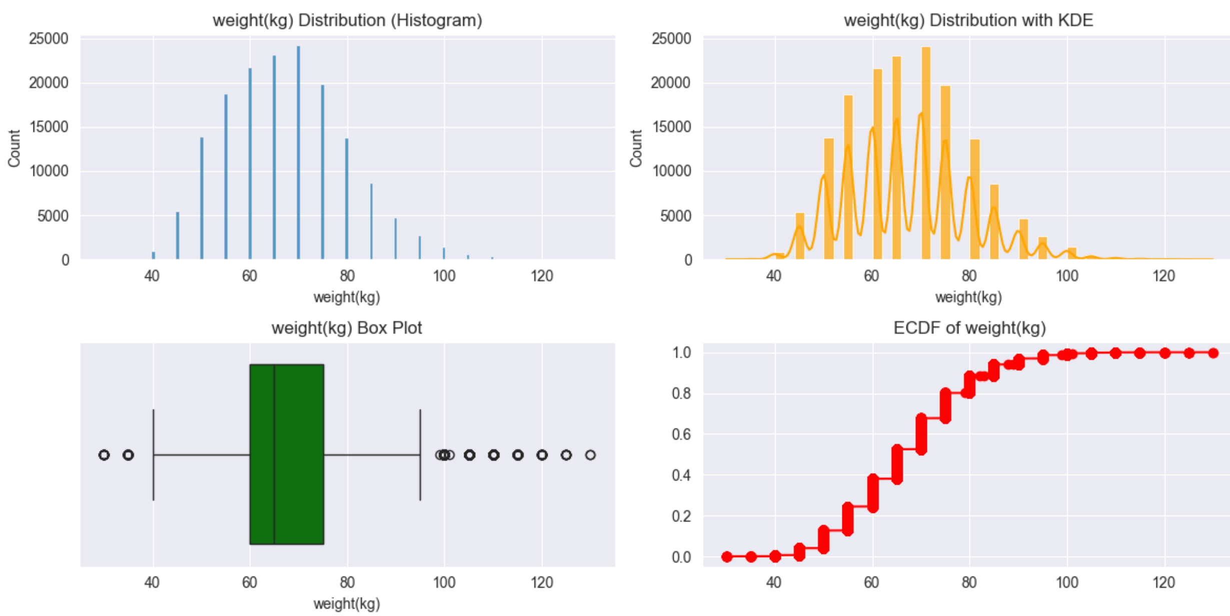
```
Percentiles and Quartiles:
Q1 (25th percentile): 20.0
Q2 (50th percentile - Median): 24.0
Q3 (75th percentile): 29.0
Interquartile Range (IQR): 9.0
```




```
numerical_univariate_analysis('weight(kg)', df['weight(kg)'])
```

```
------ Univariate Analysis for weight(kg) ------
Descriptive Statistics:
count    159256.000000
mean         67.143662
std          12.586198
min          30.000000
25%          60.000000
50%          65.000000
75%          75.000000
max         130.000000
Name: weight(kg), dtype: float64

Measures of Central Tendency:
Mean: 67.14366177726428
Median: 65.0
Mode: 70

Measures of Dispersion:
Standard Deviation: 12.586198142220114
Range: 100
```

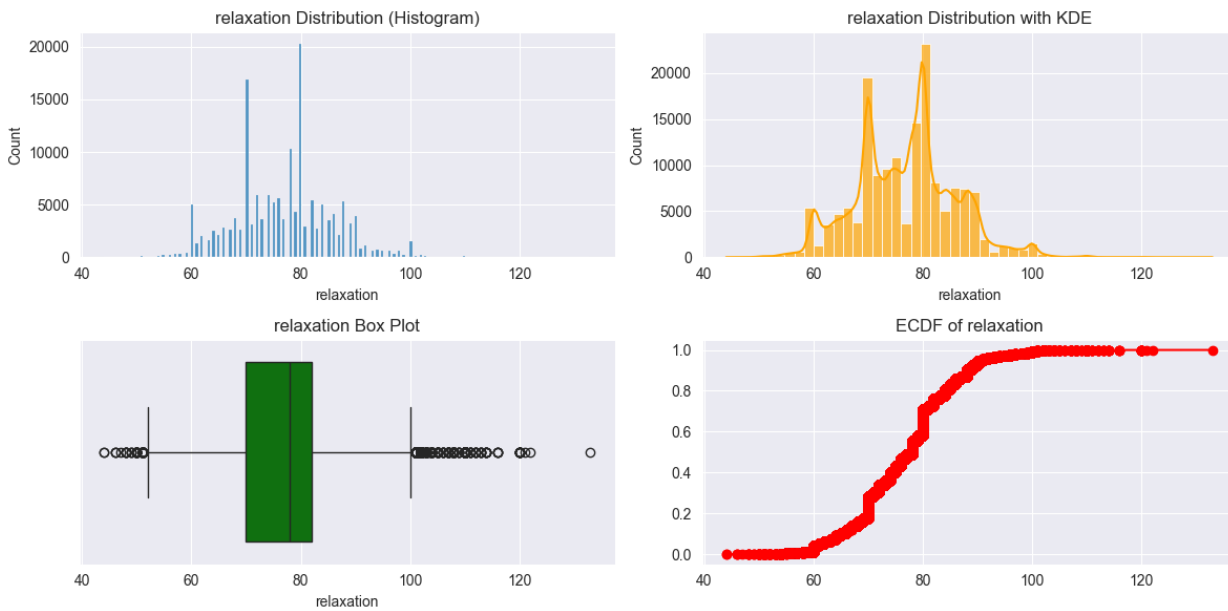
```
Variance: 158.41238367522507

Percentiles and Quartiles:
Q1 (25th percentile): 60.0
Q2 (50th percentile - Median): 65.0
Q3 (75th percentile): 75.0
Interquartile Range (IQR): 15.0
```




```
numerical_univariate_analysis('relaxation', df['relaxation'])
```

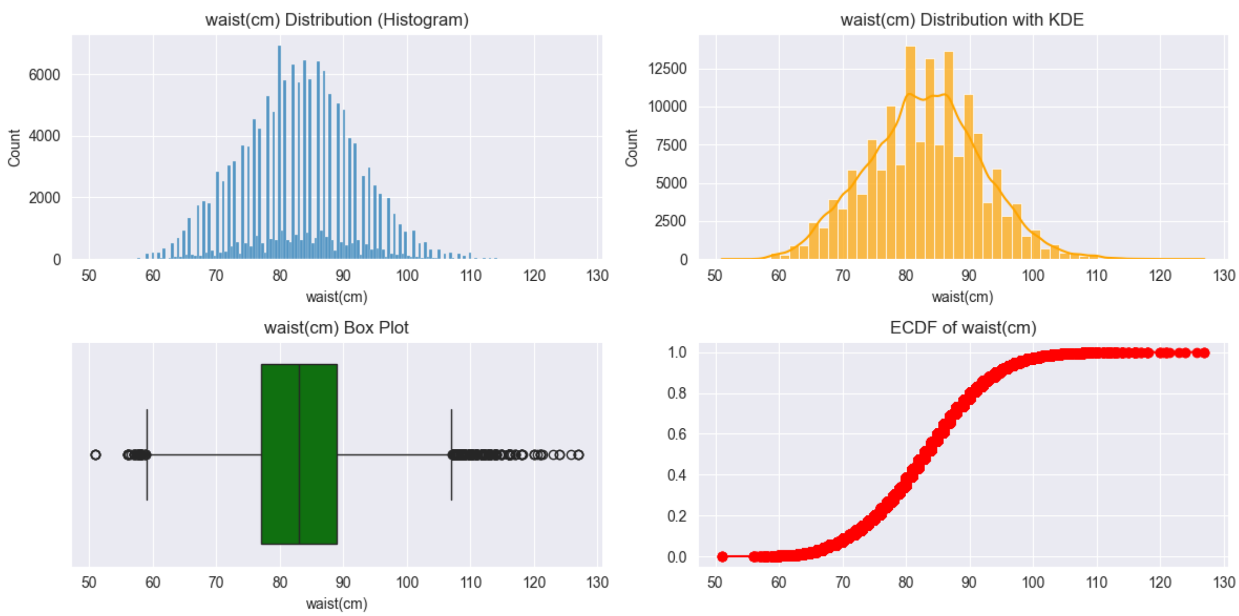
```
------ Univariate Analysis for relaxation ------
Descriptive Statistics:
count    159256.000000
mean         76.874071
std           8.994642
min          44.000000
25%          70.000000
50%          78.000000
75%          82.000000
max         133.000000
Name: relaxation, dtype: float64

Measures of Central Tendency:
Mean: 76.87407067865576
Median: 78.0
Mode: 80

Measures of Dispersion:
Standard Deviation: 8.994641687513207
```

```
Range: 89
Variance: 80.90357908675043

Percentiles and Quartiles:
Q1 (25th percentile): 70.0
Q2 (50th percentile - Median): 78.0
Q3 (75th percentile): 82.0
Interquartile Range (IQR): 12.0
```




```
numerical_univariate_analysis('waist(cm)', df['waist(cm)'])


------ Univariate Analysis for waist(cm) ------
Descriptive Statistics:
count    159256.000000
mean         83.001990
std           8.957937
min          51.000000
25%          77.000000
50%          83.000000
75%          89.000000
max         127.000000
Name: waist(cm), dtype: float64

Measures of Central Tendency:
Mean: 83.00198987793239
Median: 83.0
Mode: 80.0

Measures of Dispersion:
```

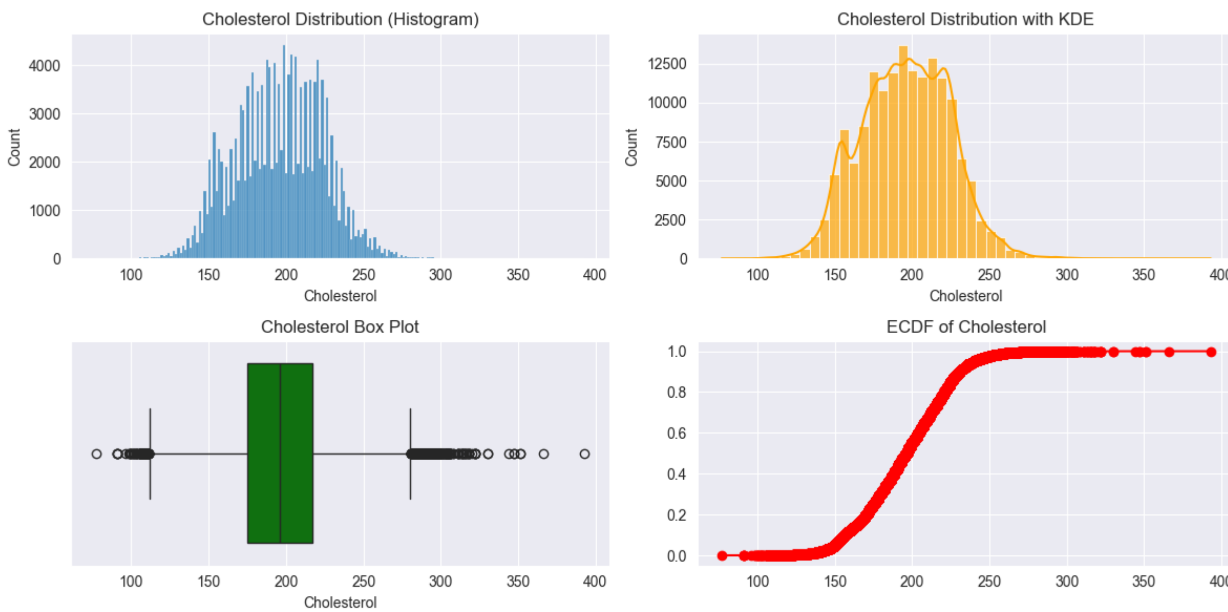
```
Standard Deviation: 8.957937261233033
Range: 76.0
Variance: 80.24463997618716

Percentiles and Quartiles:
Q1 (25th percentile): 77.0
Q2 (50th percentile - Median): 83.0
Q3 (75th percentile): 89.0
Interquartile Range (IQR): 12.0
```




```
numerical_univariate_analysis('Cholesterol', df['Cholesterol'])
```

```
------ Univariate Analysis for Cholesterol ------
Descriptive Statistics:
count    159256.000000
mean        195.796165
std          28.396959
min          77.000000
25%         175.000000
50%         196.000000
75%         217.000000
max         393.000000
Name: Cholesterol, dtype: float64

Measures of Central Tendency:
Mean: 195.79616466569547
Median: 196.0
Mode: 197
```

```
Measures of Dispersion:
Standard Deviation: 28.39695908288623
Range: 316
Variance: 806.3872851551148

Percentiles and Quartiles:
Q1 (25th percentile): 175.0
Q2 (50th percentile - Median): 196.0
Q3 (75th percentile): 217.0
Interquartile Range (IQR): 42.0
```




```
numerical_univariate_analysis('HDL', df['HDL'])
```

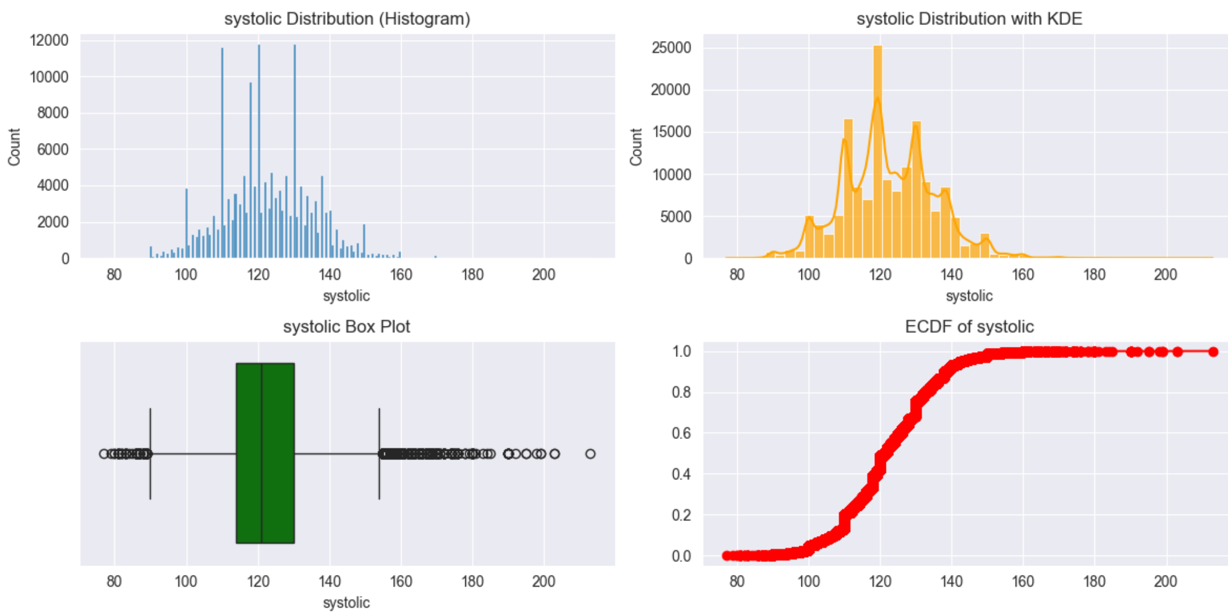
```
------ Univariate Analysis for HDL ------
Descriptive Statistics:
count    159256.000000
mean         55.852684
std          13.964141
min           9.000000
25%          45.000000
50%          54.000000
75%          64.000000
max         136.000000
Name: HDL, dtype: float64

Measures of Central Tendency:
Mean: 55.852683729341436
Median: 54.0
Mode: 47
```

```
Measures of Dispersion:
Standard Deviation: 13.964141074947342
Range: 127
Variance: 194.99723596103152

Percentiles and Quartiles:
Q1 (25th percentile): 45.0
Q2 (50th percentile - Median): 54.0
Q3 (75th percentile): 64.0
Interquartile Range (IQR): 19.0
```


```
numerical_univariate_analysis('systolic', df['systolic'])
```

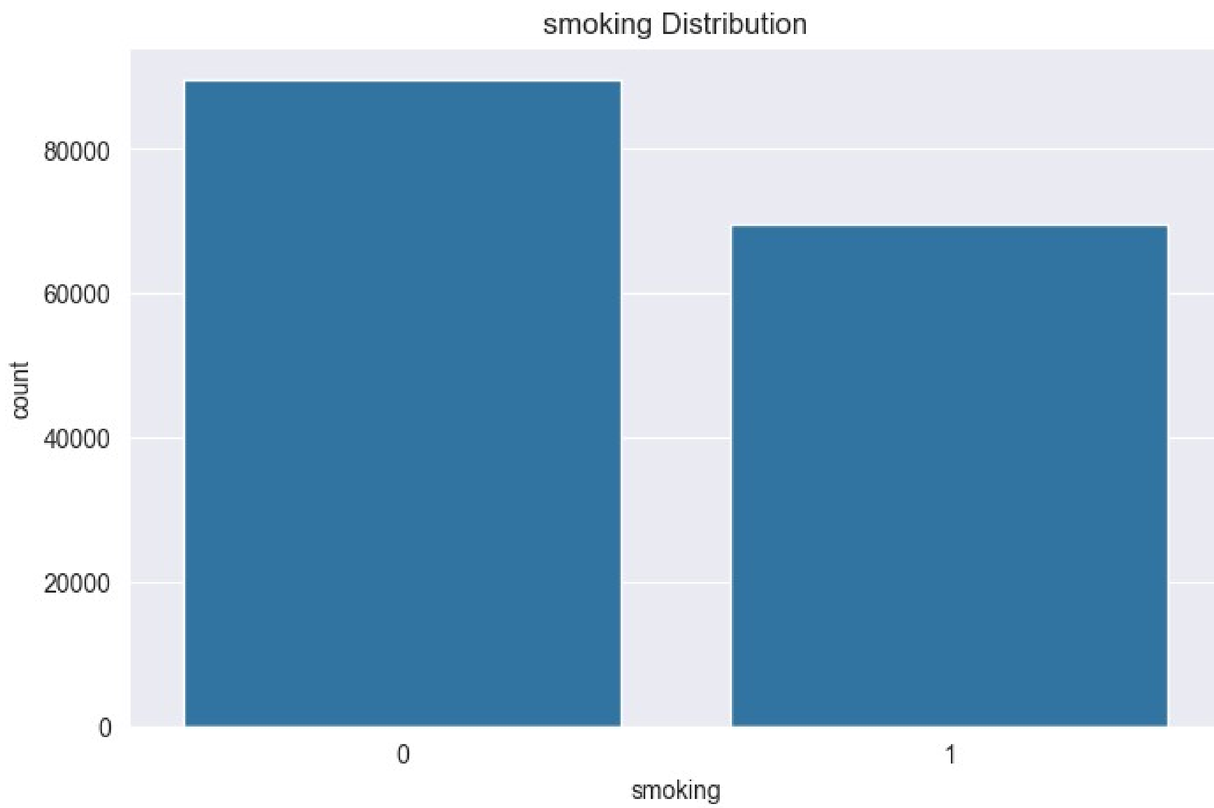
```
------ Univariate Analysis for systolic ------
Descriptive Statistics:
count    159256.000000
mean        122.503648
std          12.729315
min          77.000000
25%         114.000000
50%         121.000000
75%         130.000000
max         213.000000
Name: systolic, dtype: float64

Measures of Central Tendency:
Mean: 122.503648214196
Median: 121.0
```

```
Mode: 130

Measures of Dispersion:
Standard Deviation: 12.729315157676696
Range: 136
Variance: 162.03546438345768

Percentiles and Quartiles:
Q1 (25th percentile): 114.0
Q2 (50th percentile - Median): 121.0
Q3 (75th percentile): 130.0
Interquartile Range (IQR): 16.0
```

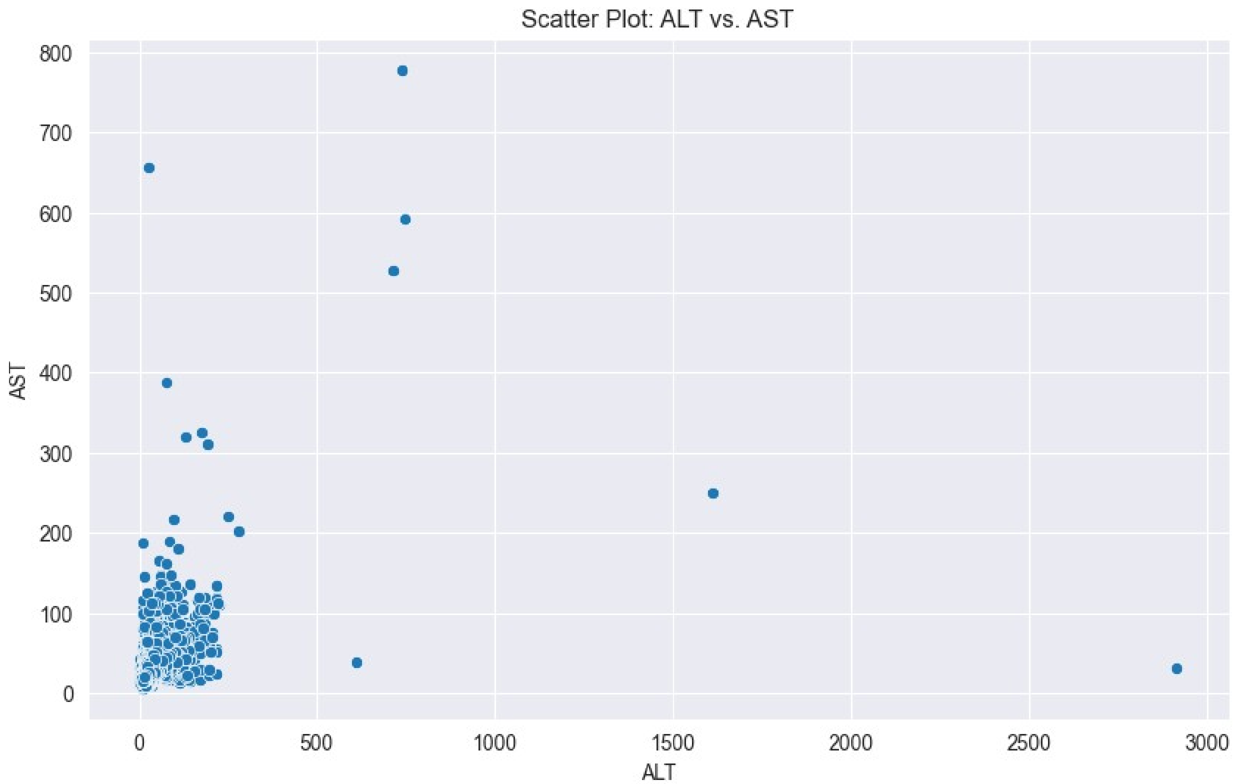


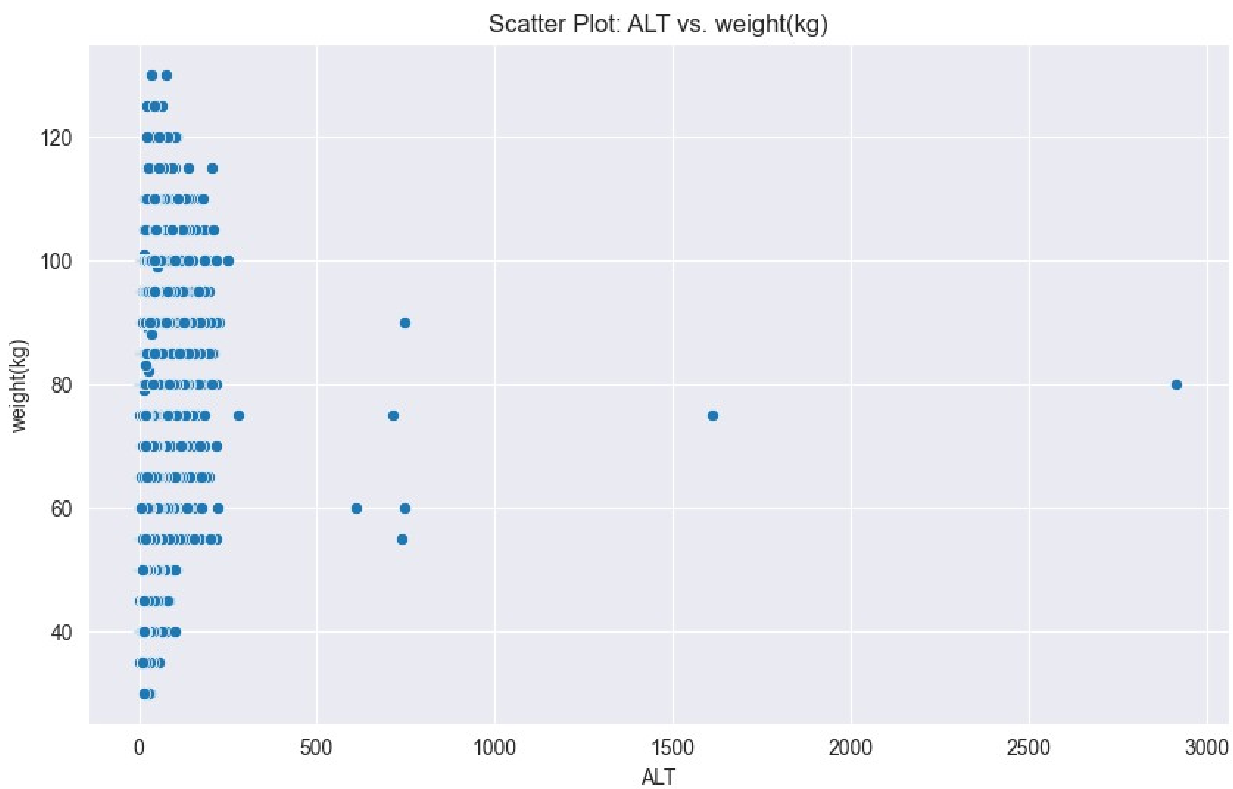

```python
categorical_univariate_analysis('smoking', df['smoking'])
```

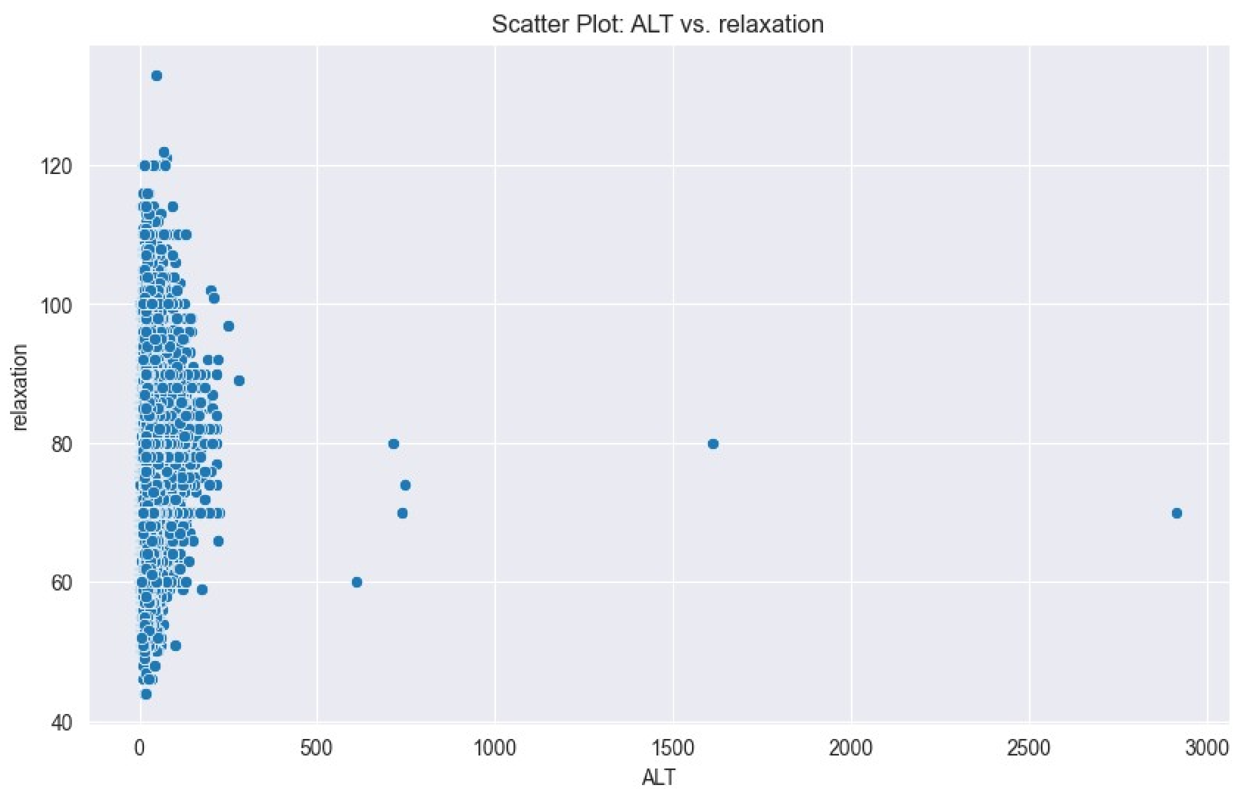
```
------ Univariate Analysis for smoking ------
Frequency Distribution:
smoking
0    89603
1    69653
Name: count, dtype: int64
```

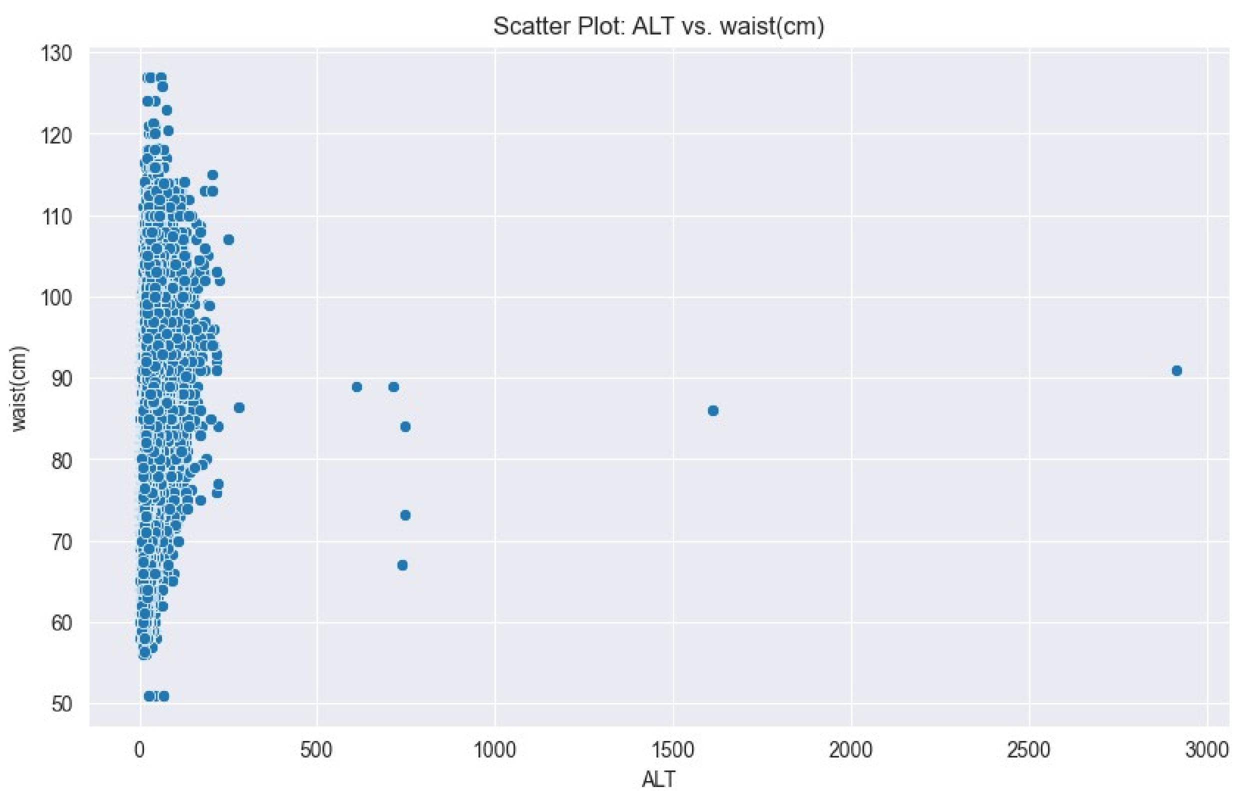


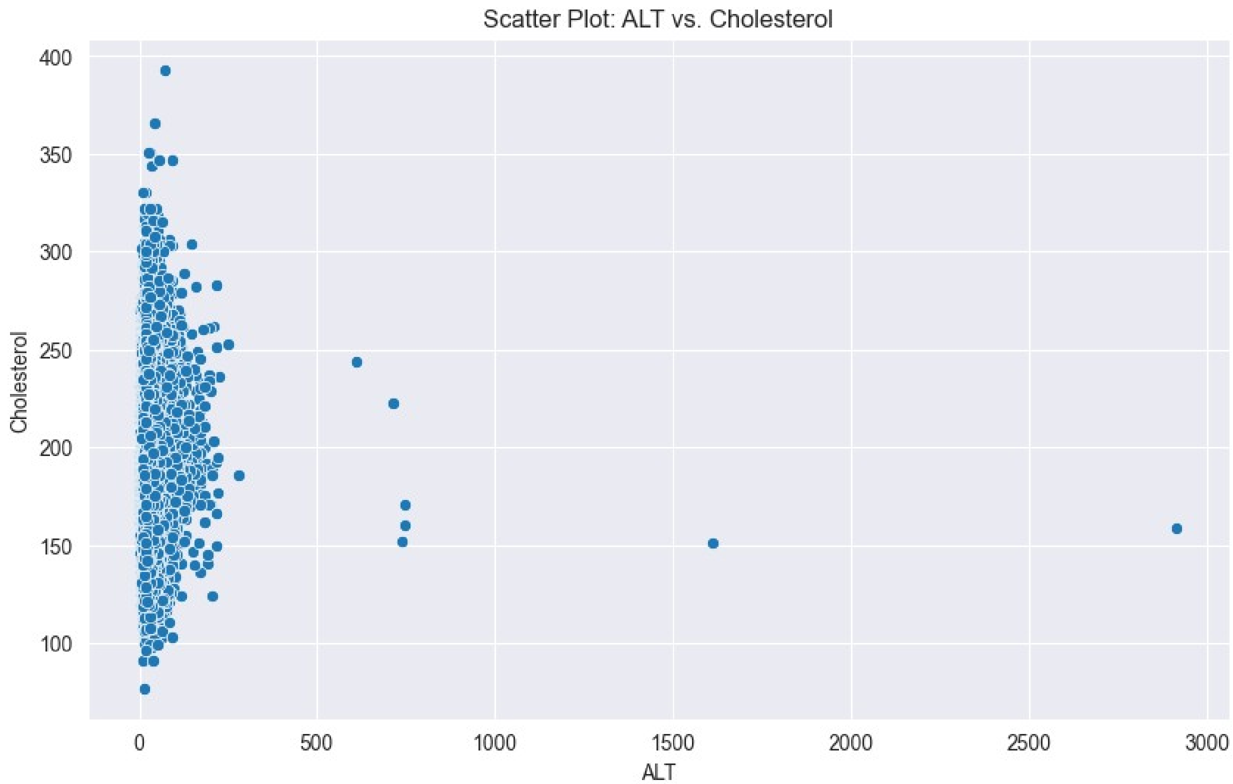

```
# Scatter plots for numeric-numeric relationships
numeric_features = ['ALT', 'AST', 'weight(kg)', 'relaxation',
'waist(cm)', 'Cholesterol', 'HDL', 'systolic']

for i in range(len(numeric_features)):
    for j in range(i+1, len(numeric_features)):
        plt.figure(figsize=(10, 6))
        sns.scatterplot(x=numeric_features[i], y=numeric_features[j],
data=df)
        plt.title(f'Scatter Plot: {numeric_features[i]} vs.
{numeric_features[j]}')
        plt.show()
```


Scatter Plot: ALT vs. AST
AST
ALT
0
100
200
300
400
500
600
700
800
0
500
1000
1500
2000
2500
3000

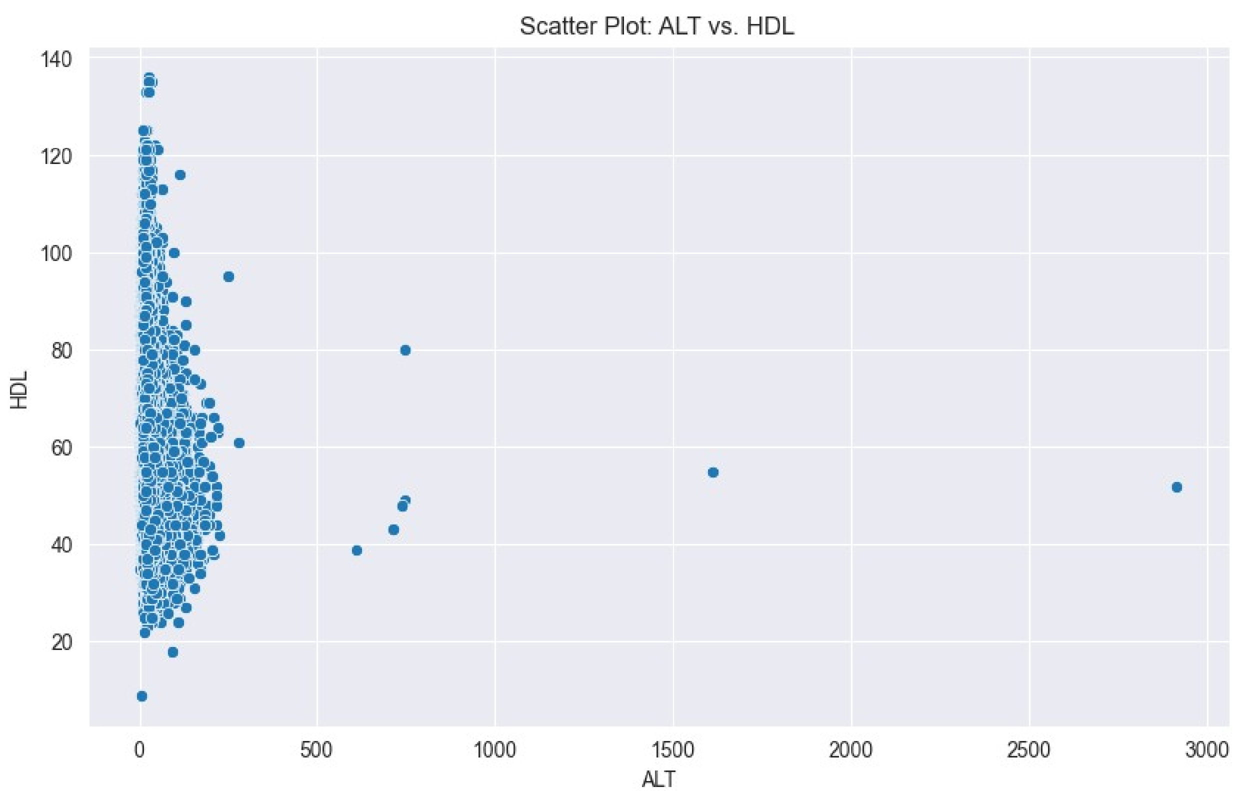

Scatter Plot: ALT vs. weight(kg)
weight(kg)
ALT
40
60
80
100
120
0
500
1000
1500
2000
2500
3000


Scatter Plot: ALT vs. relaxation
relaxation
ALT
40
60
80
100
120
0
500
1000
1500
2000
2500
3000

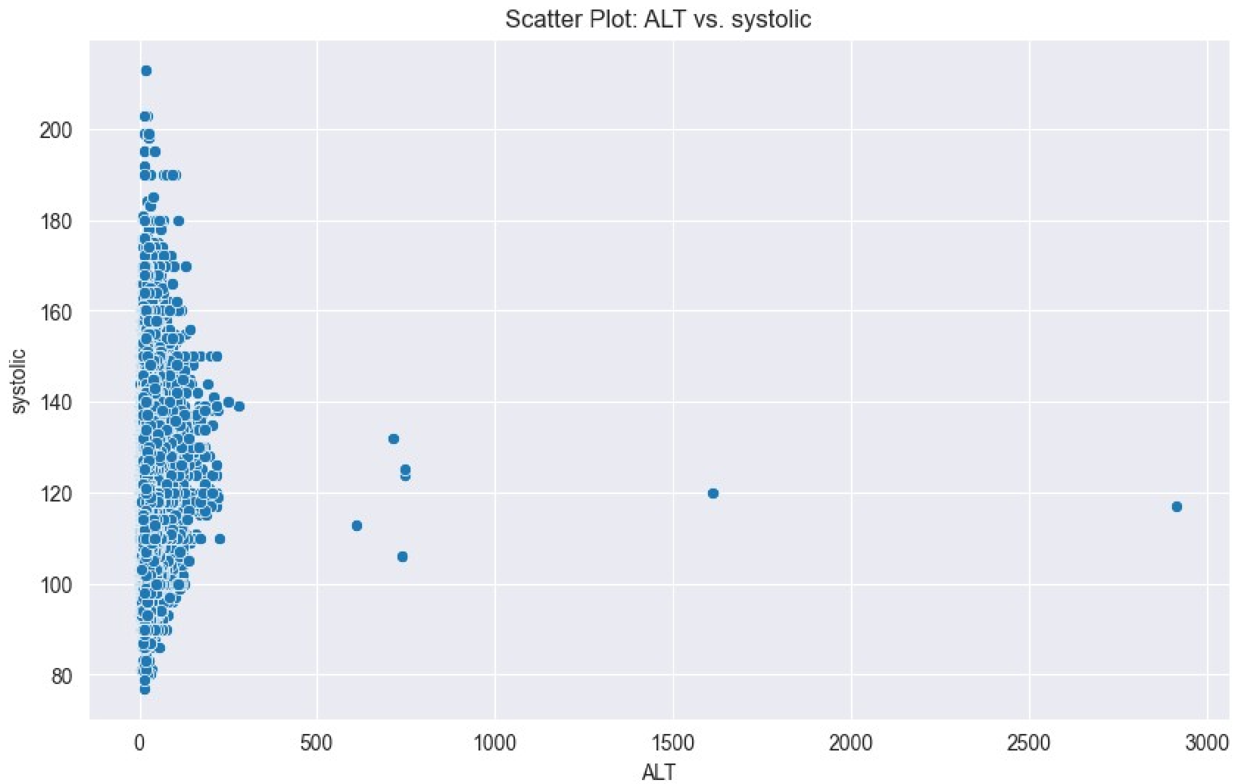

Scatter Plot: ALT vs. waist(cm)
waist(cm)
ALT
50
60
70
80
90
100
110
120
130
0
500
1000
1500
2000
2500
3000



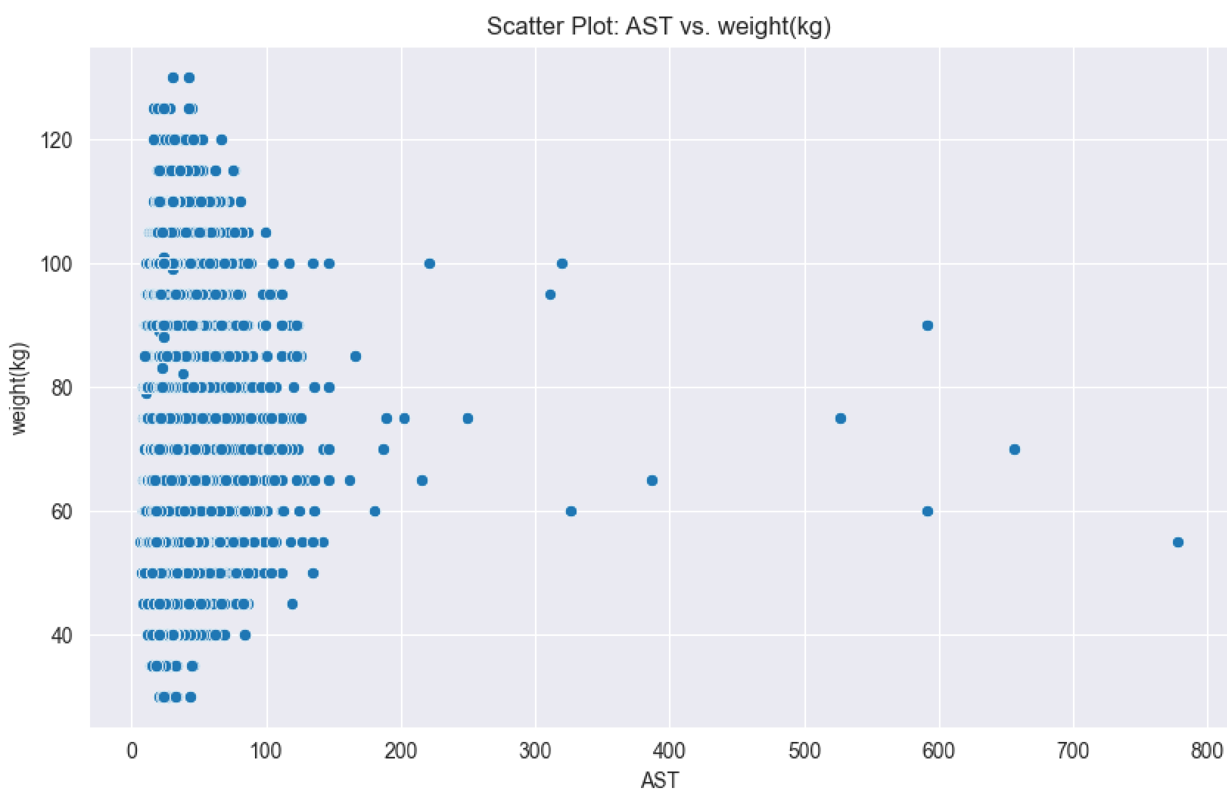
Scatter Plot: ALT vs. Cholesterol
Cholesterol
ALT
400
350
300
250
200
150
100
0
500
1000
1500
2000
2500
3000



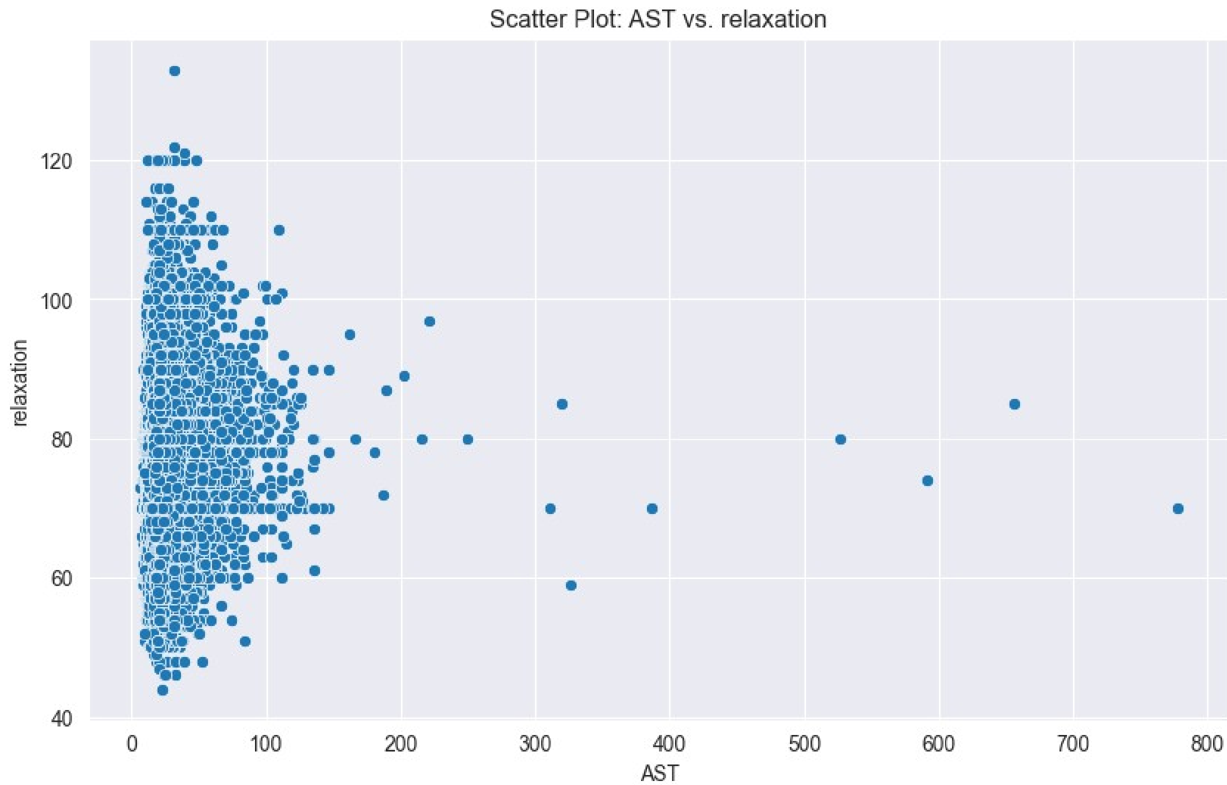
Scatter Plot: ALT vs. HDL
HDL
ALT
140
120
100
80
60
40
20
0
500
1000
1500
2000
2500
3000

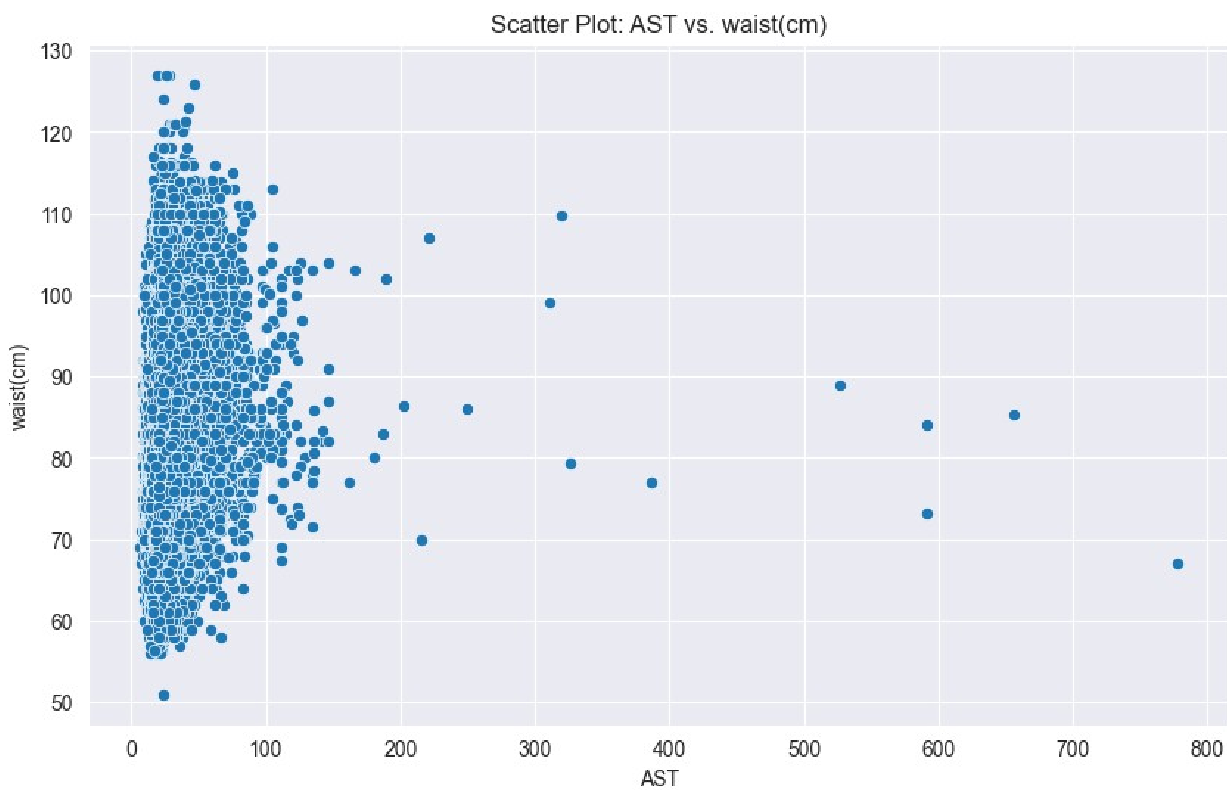


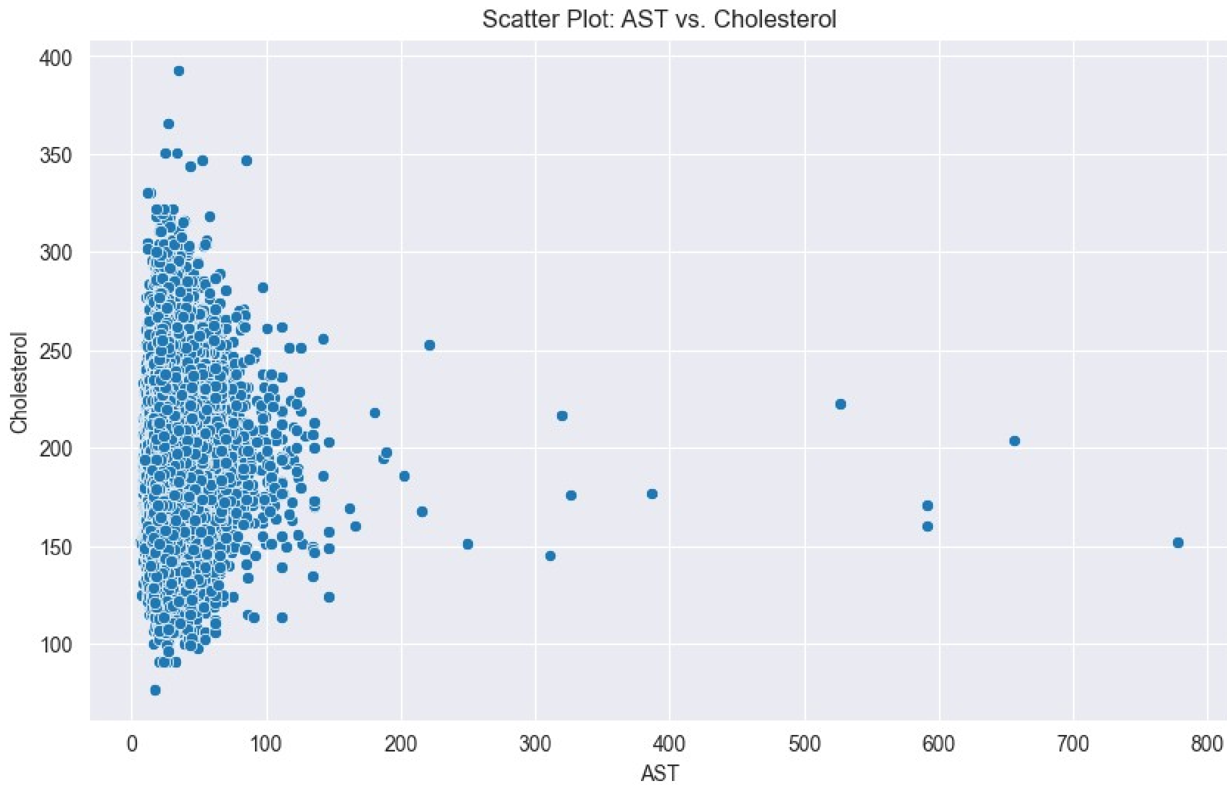
Scatter Plot: ALT vs. systolic
systolic
ALT
80
100
120
140
160
180
200
0
500
1000
1500
2000
2500
3000

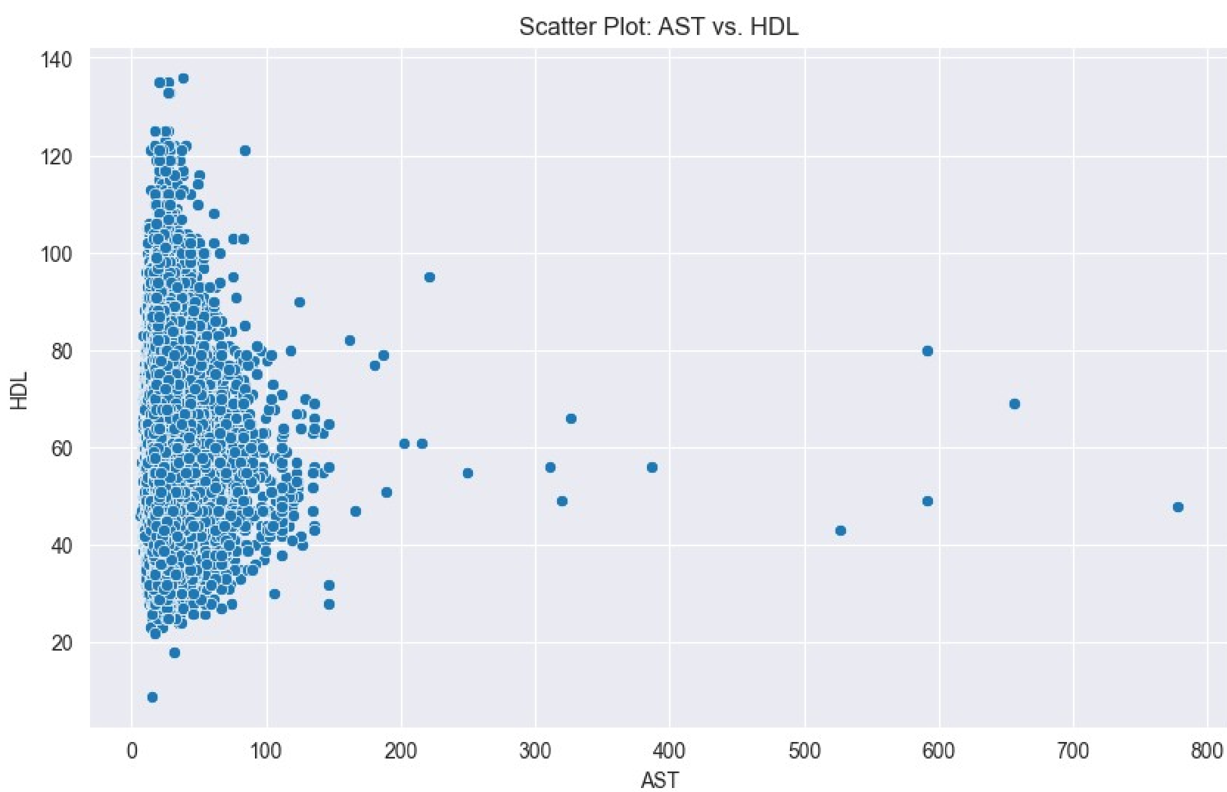


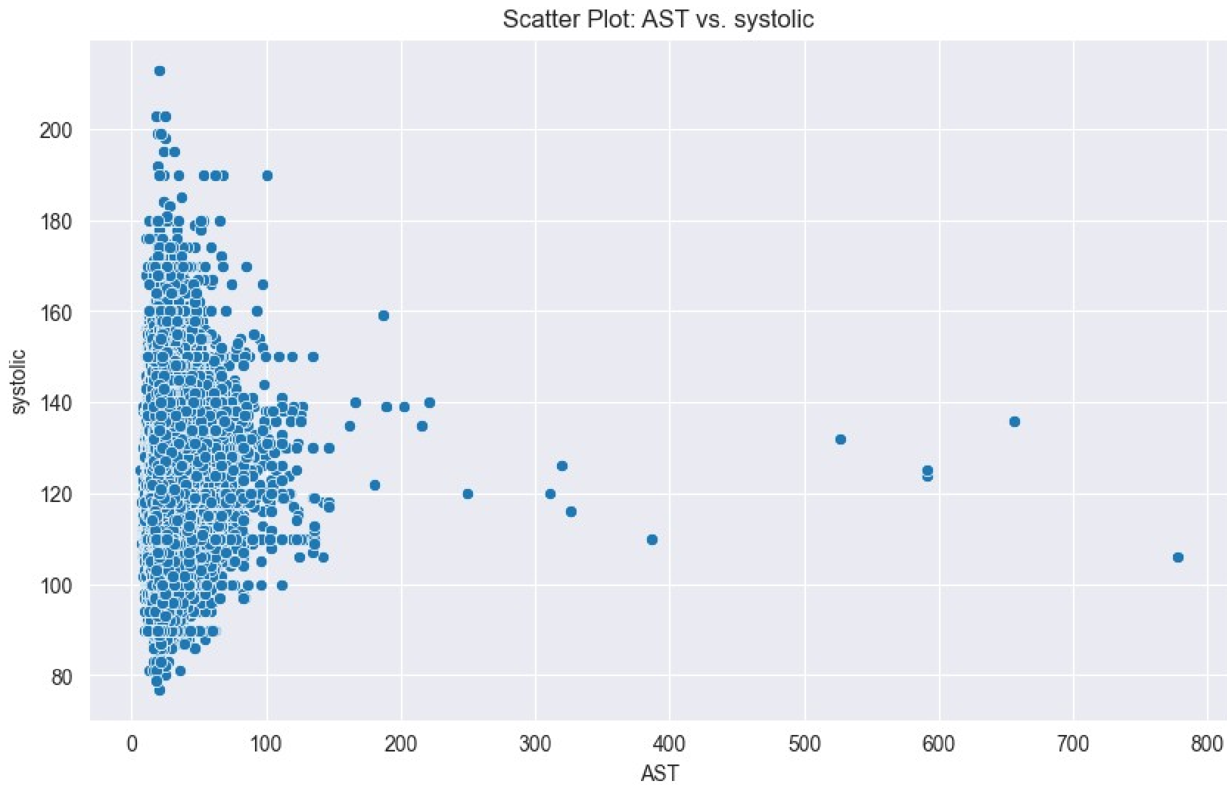
Scatter Plot: AST vs. weight(kg)
weight(kg)
AST
40
60
80
100
120
0
100
200
300
400
500
600
700
800


Scatter Plot: AST vs. relaxation
relaxation
AST
40
60
80
100
120
0
100
200
300
400
500
600
700
800

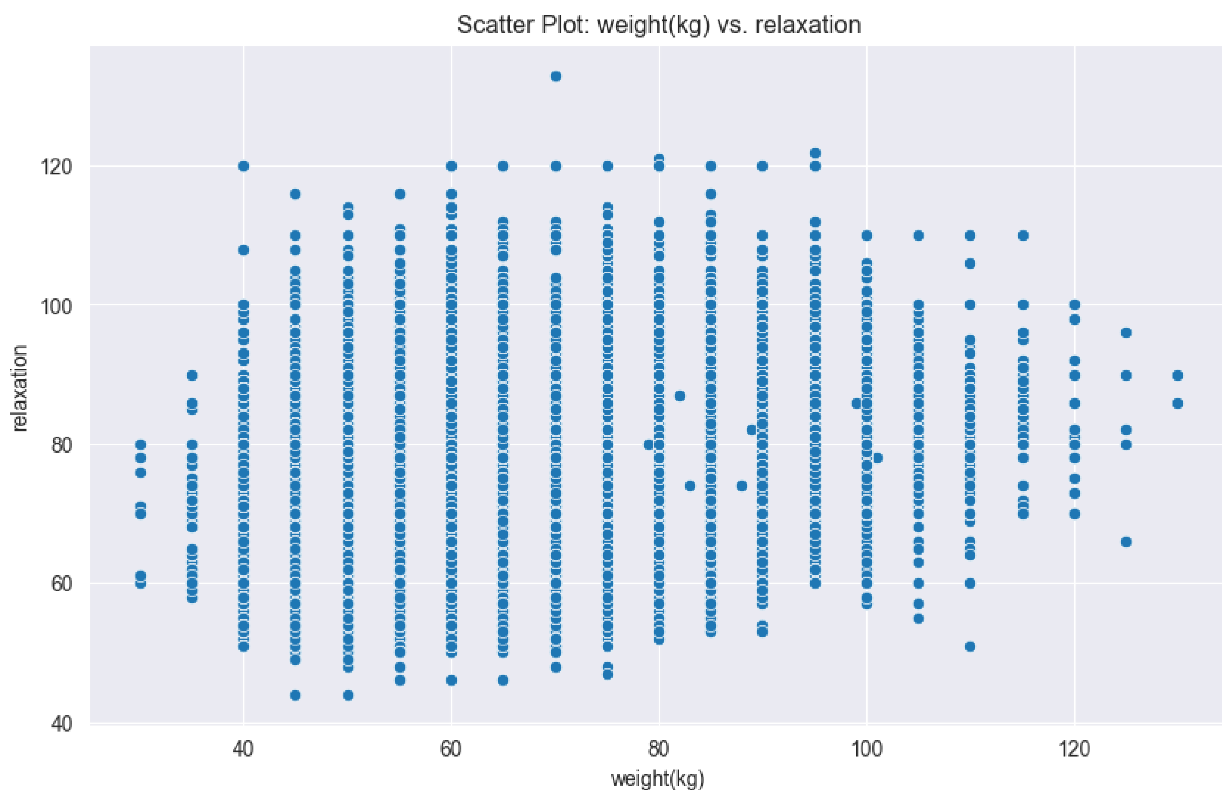

Scatter Plot: AST vs. waist(cm)
waist(cm)
AST
50
60
70
80
90
100
110
120
130
0
100
200
300
400
500
600
700
800


Scatter Plot: AST vs. Cholesterol
Cholesterol
400
350
300
250
200
150
100
0
100
200
300
400
500
600
700
800
AST

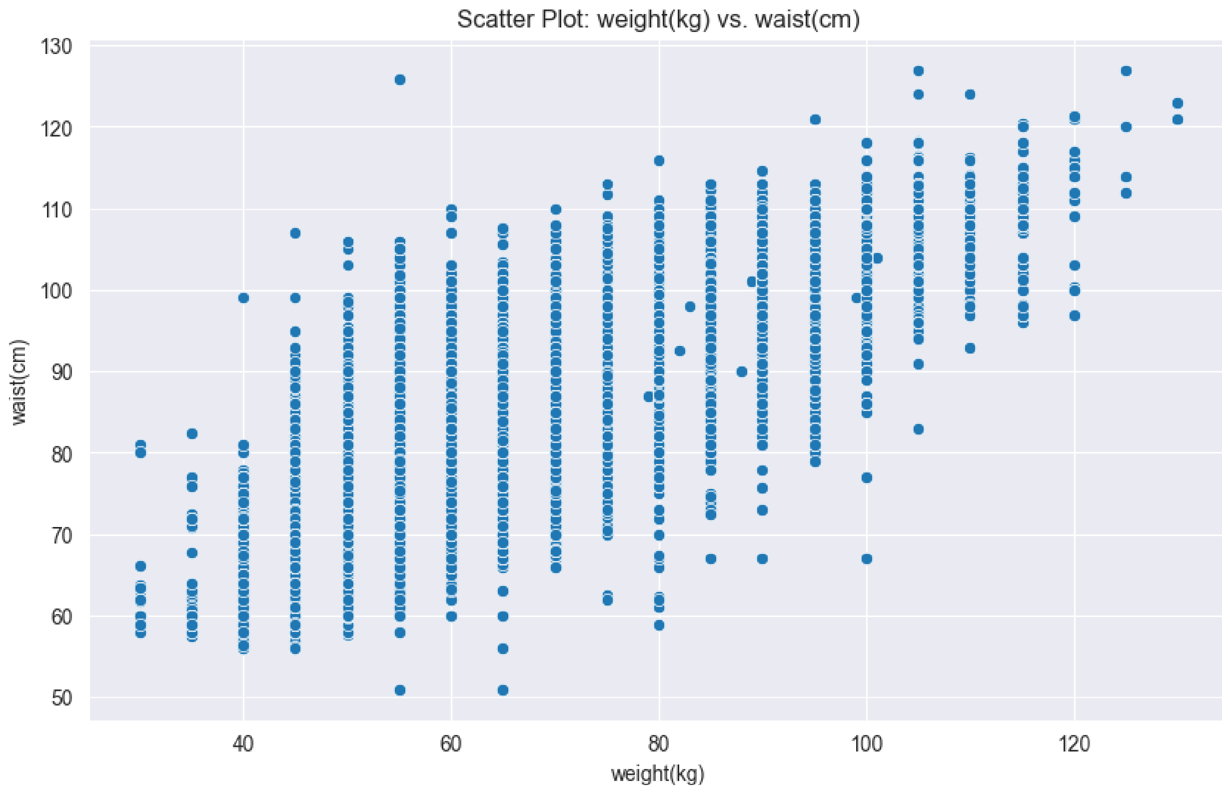

Scatter Plot: AST vs. HDL
HDL
140
120
100
80
60
40
20
0
100
200
300
400
500
600
700
800
AST

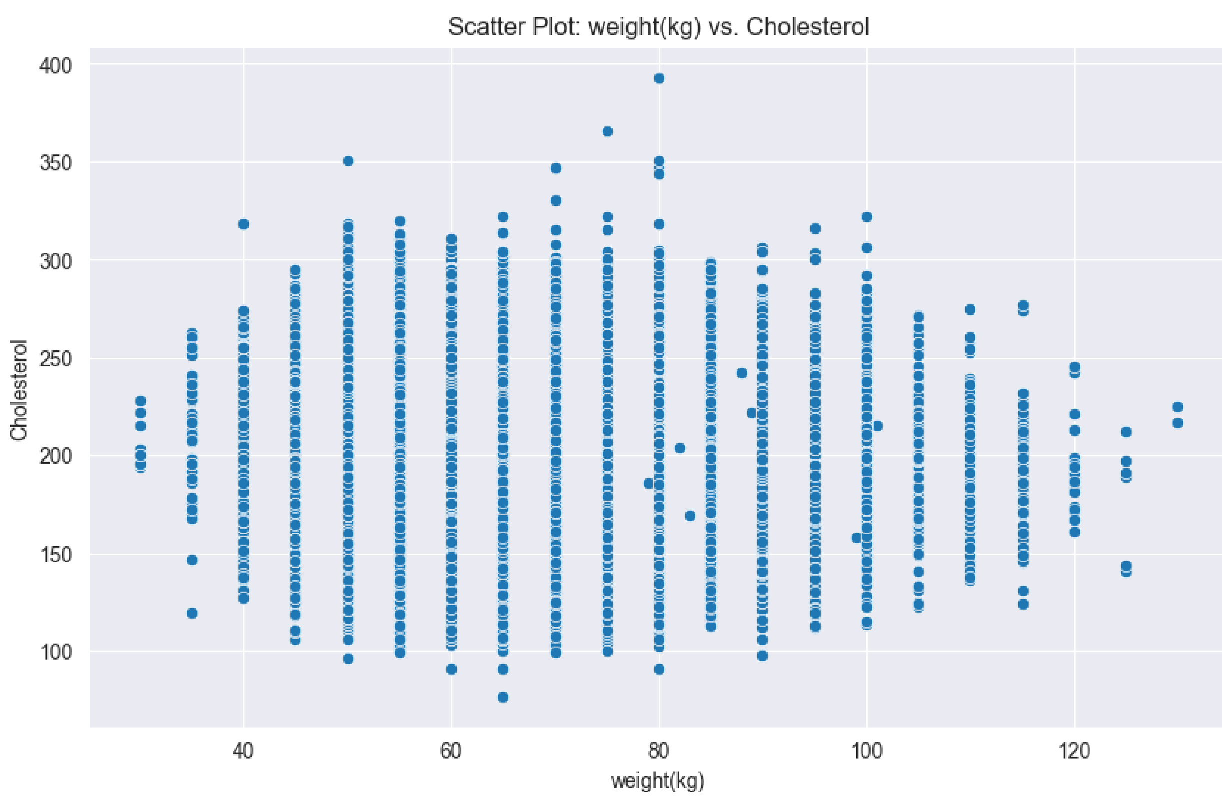


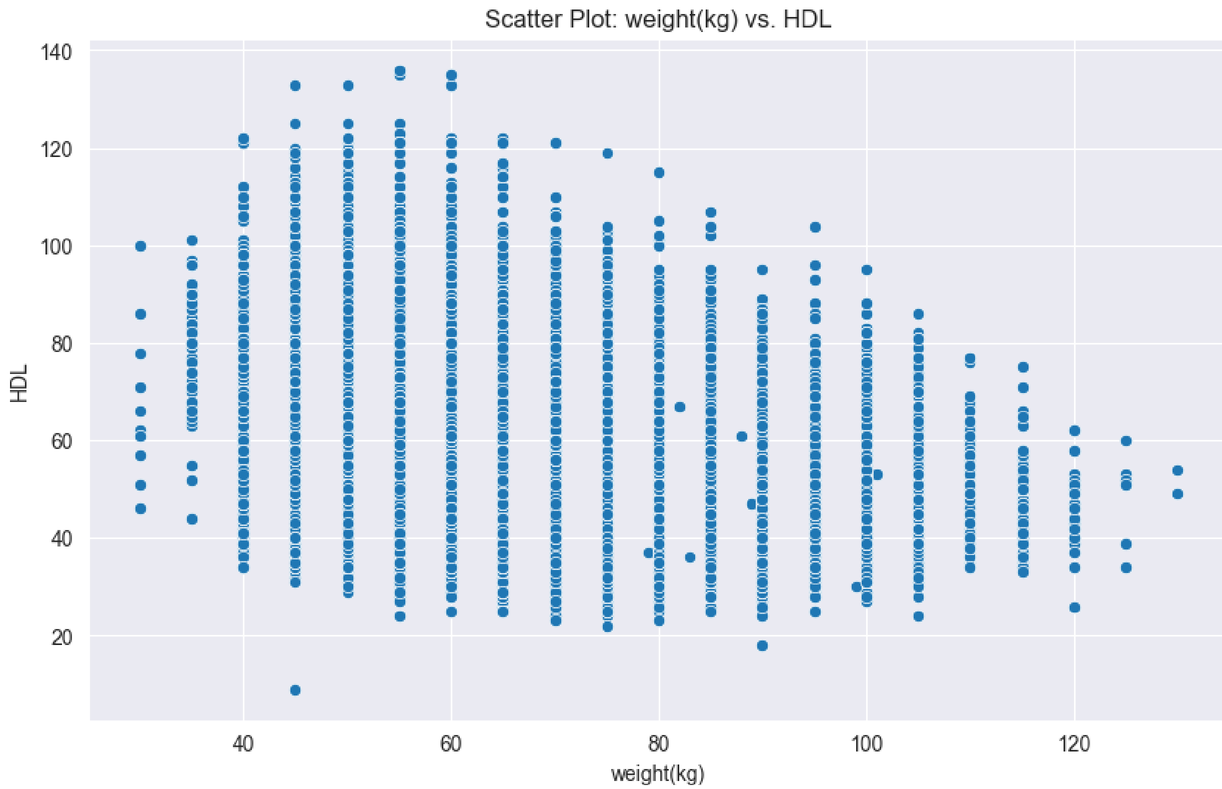
Scatter Plot: AST vs. systolic
systolic
AST
0
100
200
300
400
500
600
700
800
80
100
120
140
160
180
200



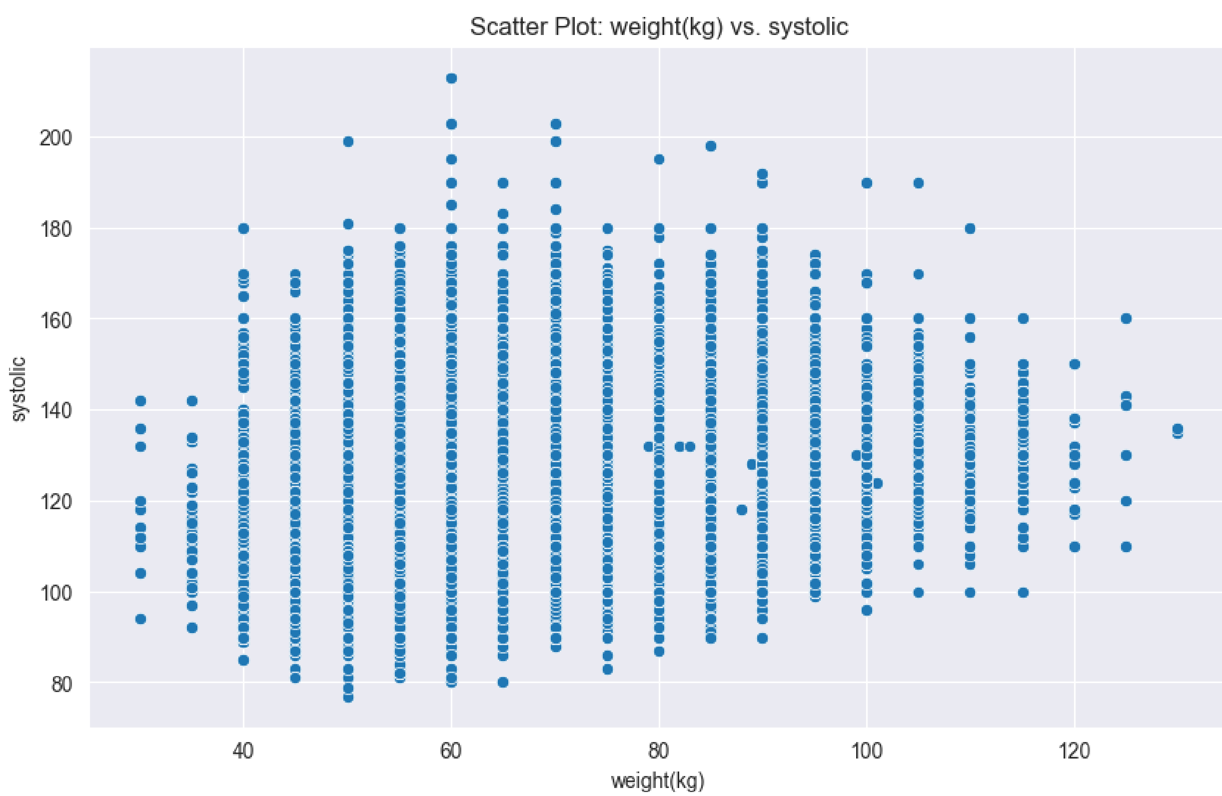
Scatter Plot: weight(kg) vs. relaxation
relaxation
weight(kg)
40
60
80
100
120


Scatter Plot: weight(kg) vs. waist(cm)
waist(cm)
weight(kg)
130
120
110
100
90
80
70
60
50
40
60
80
100
120


Scatter Plot: weight(kg) vs. Cholesterol
Cholesterol
weight(kg)
400
350
300
250
200
150
100
40
60
80
100
120

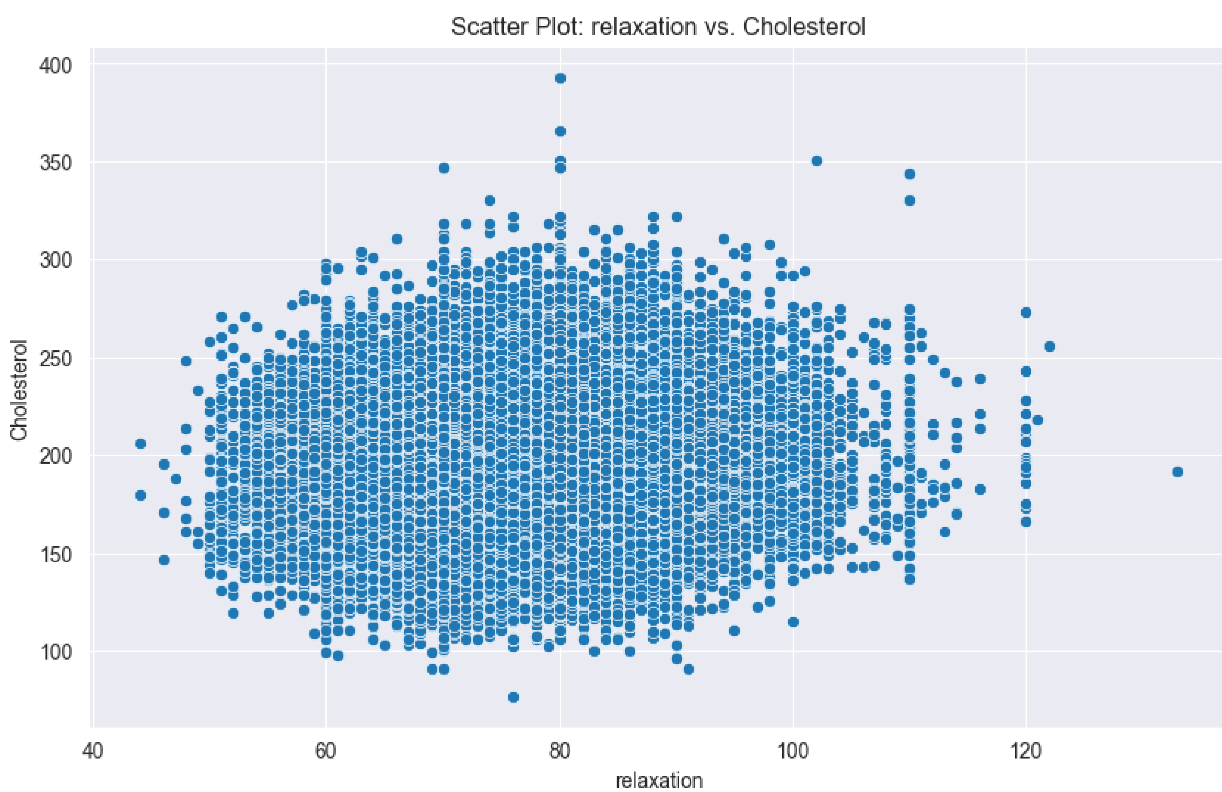


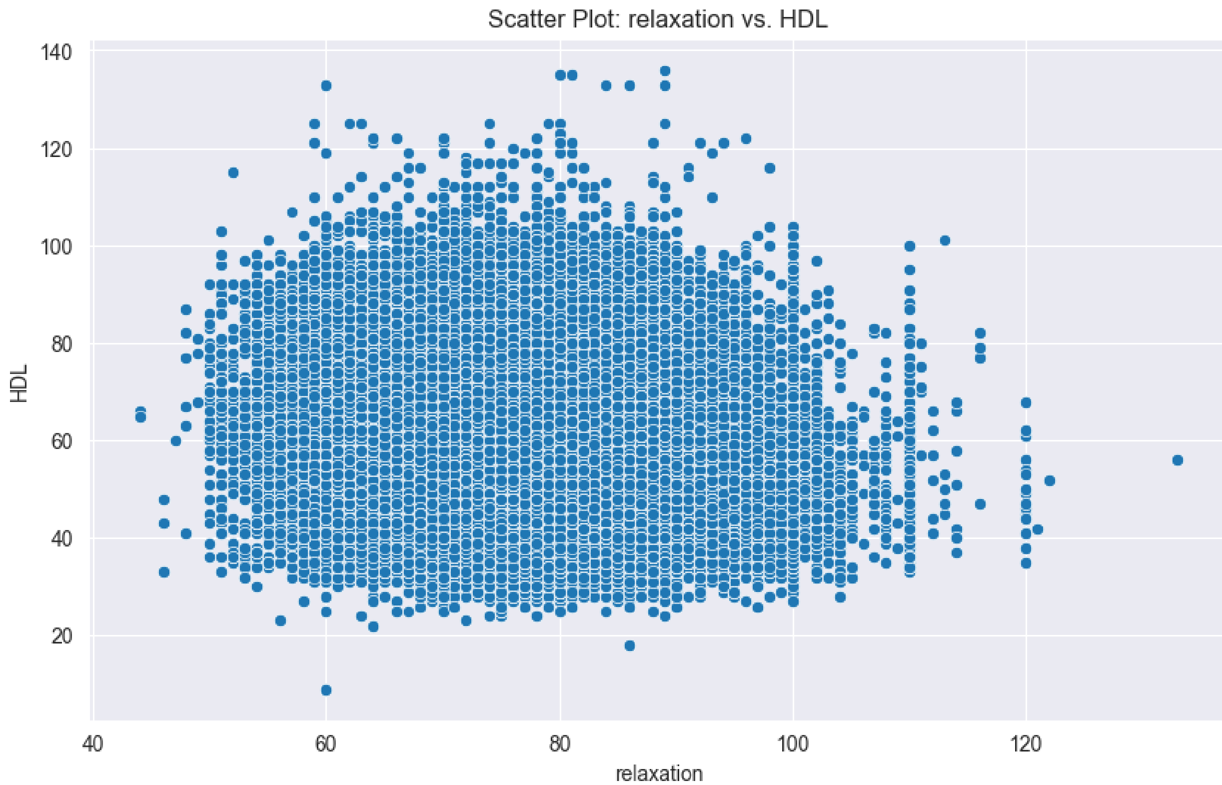
Scatter Plot: weight(kg) vs. HDL
HDL
weight(kg)
140
120
100
80
60
40
20
40
60
80
100
120

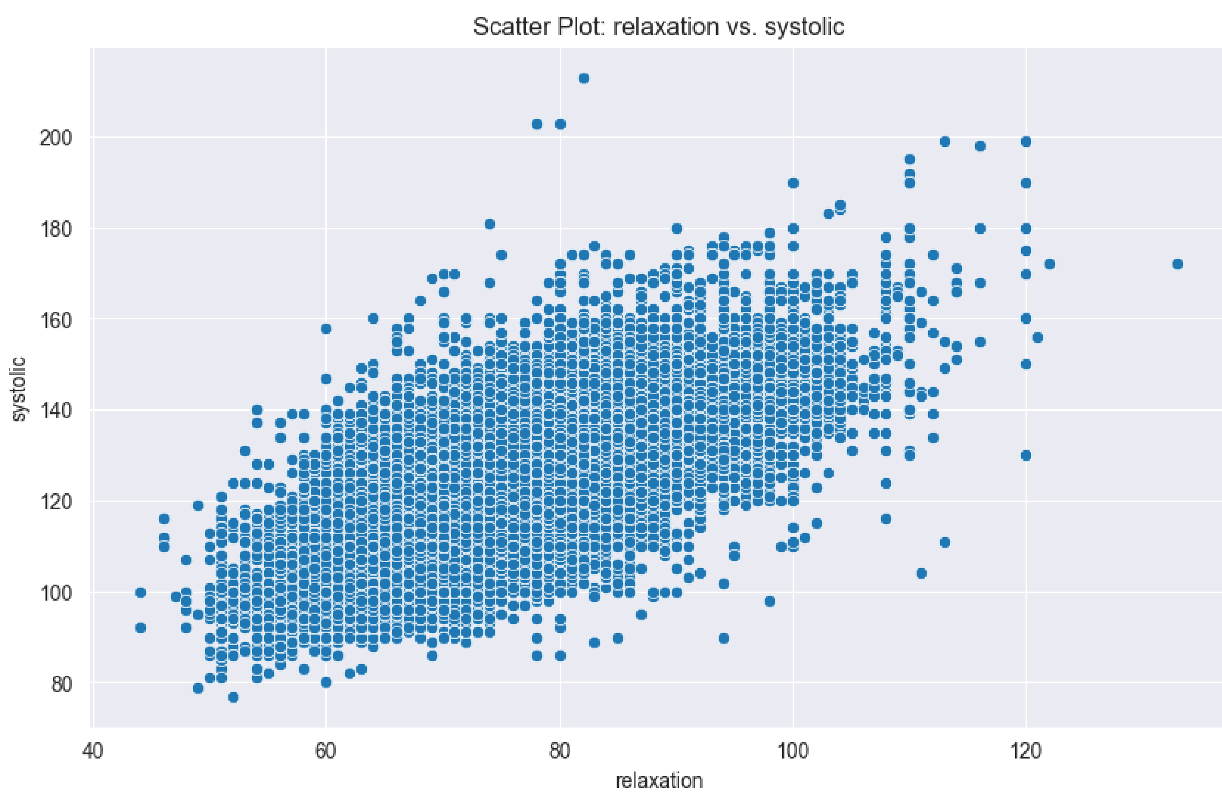


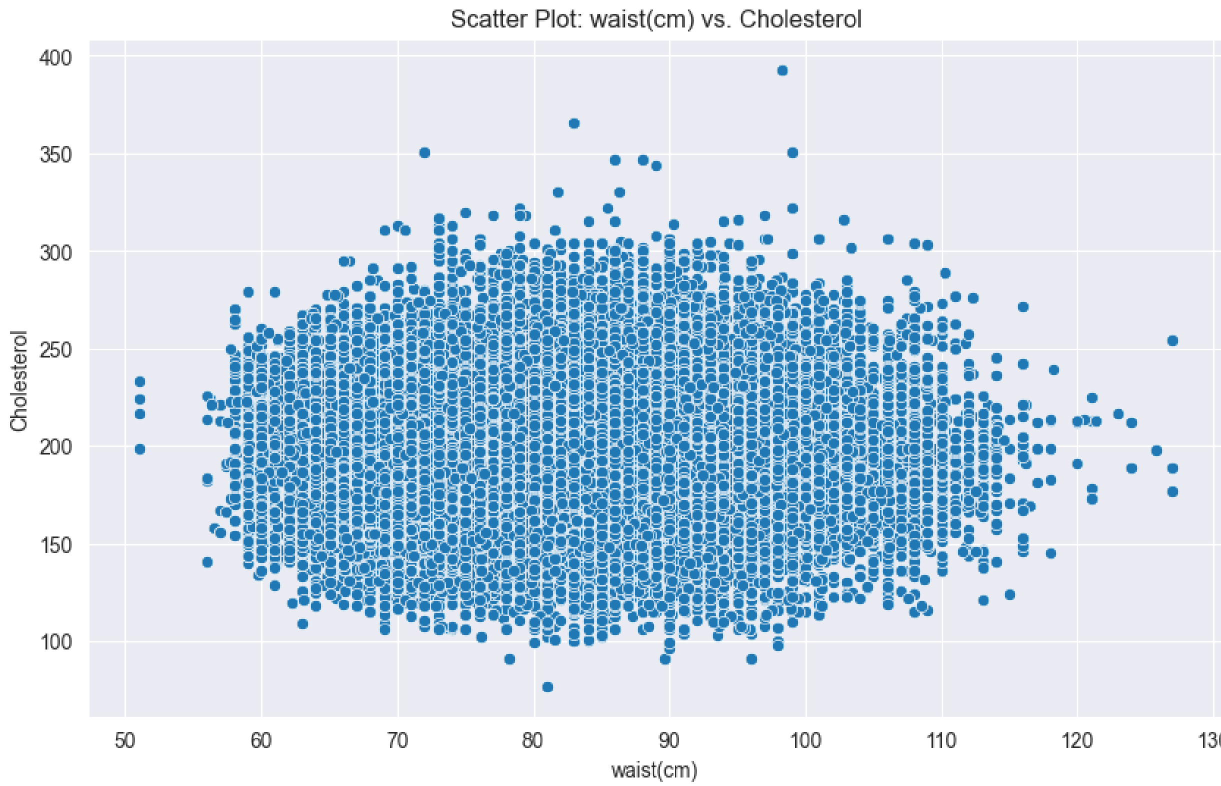
Scatter Plot: weight(kg) vs. systolic
systolic
weight(kg)
200
180
160
140
120
100
80
40
60
80
100
120

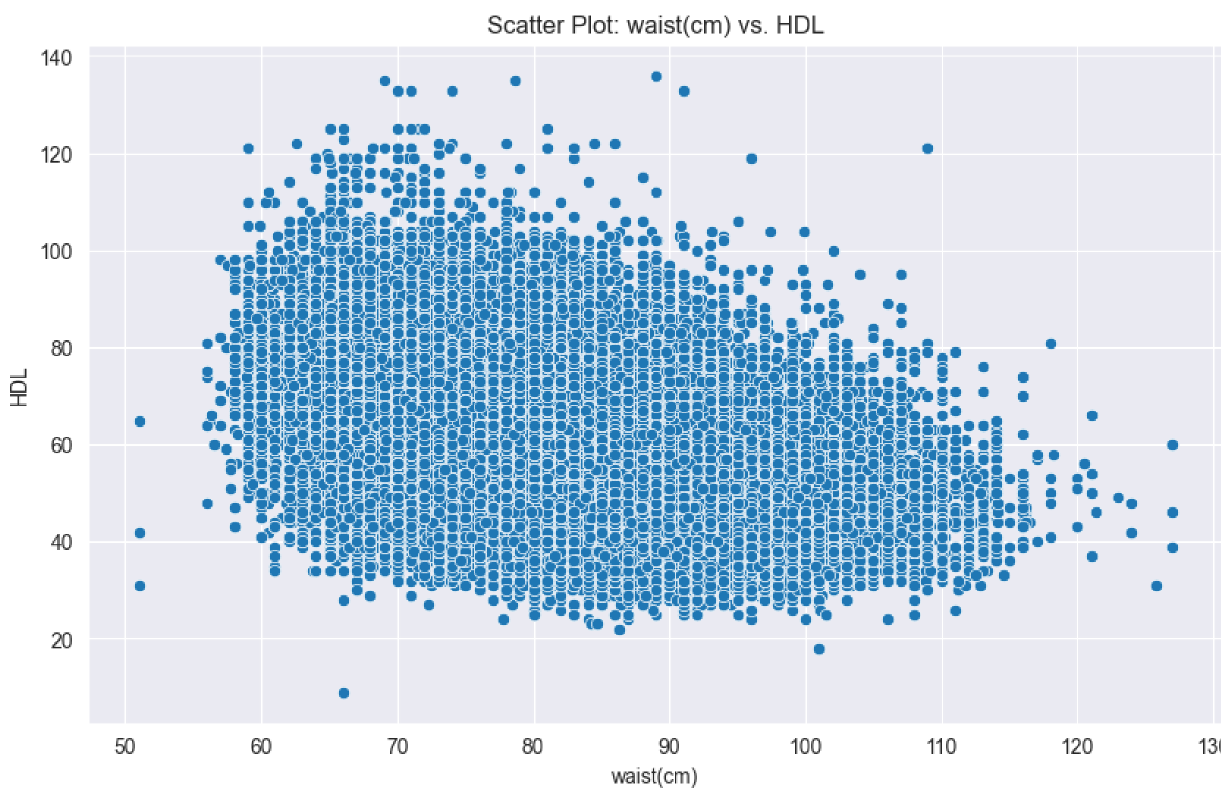

Scatter Plot: relaxation vs. waist(cm)
waist(cm)
relaxation
130
120
110
100
90
80
70
60
50
40
60
80
100
120

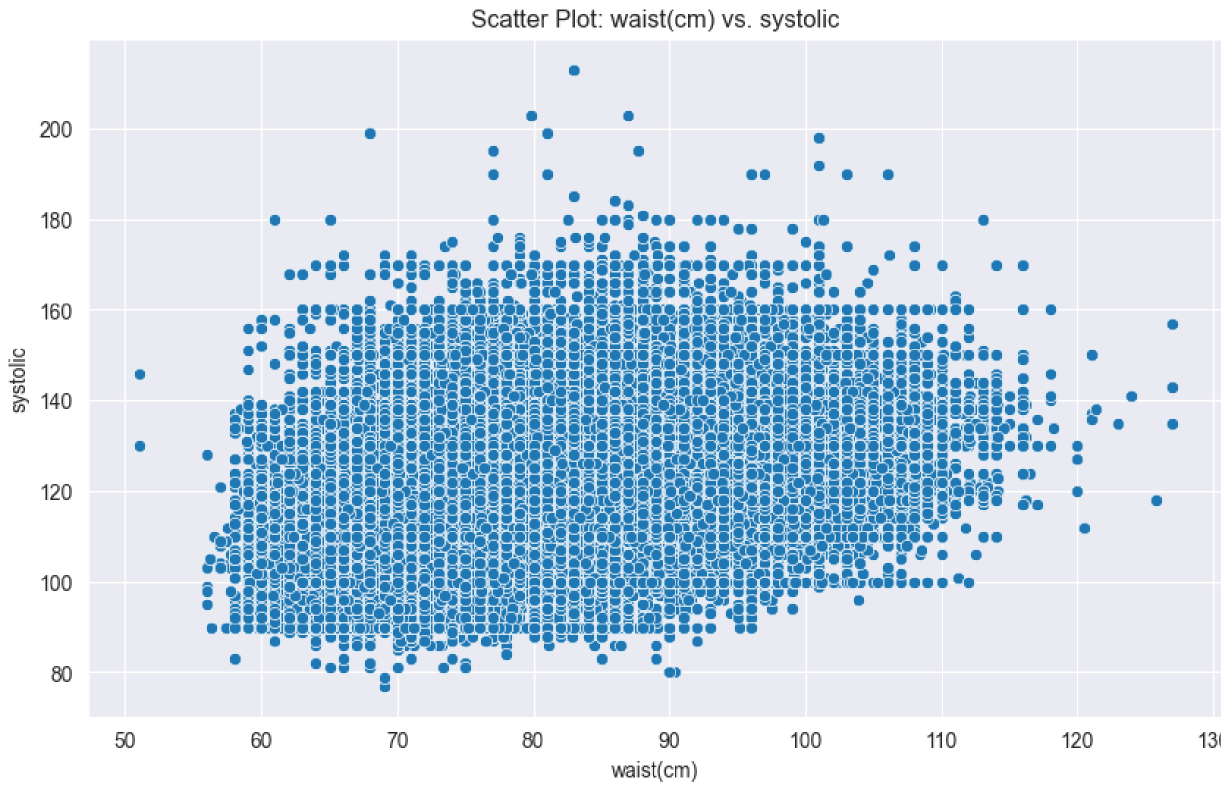

Scatter Plot: relaxation vs. Cholesterol
Cholesterol
relaxation
400
350
300
250
200
150
100
40
60
80
100
120

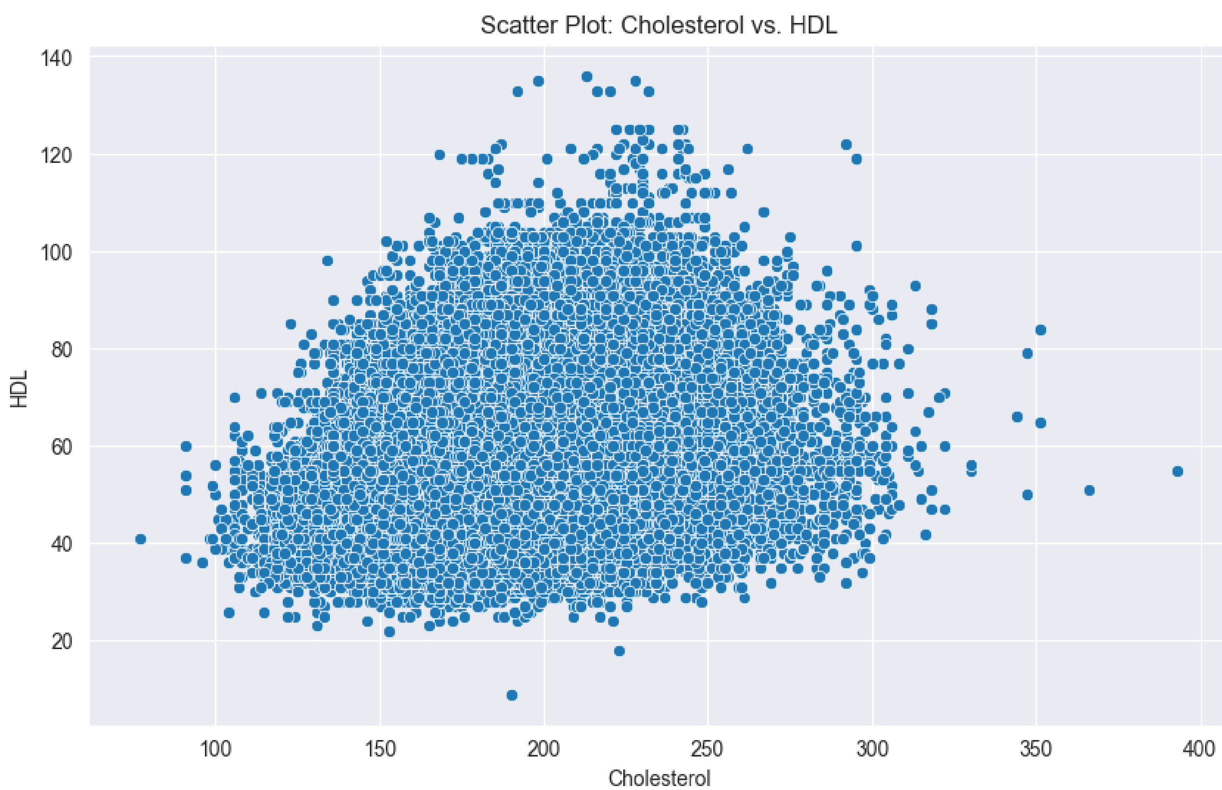

Scatter Plot: relaxation vs. HDL
HDL
140
120
100
80
60
40
20
40
60
80
100
120
relaxation

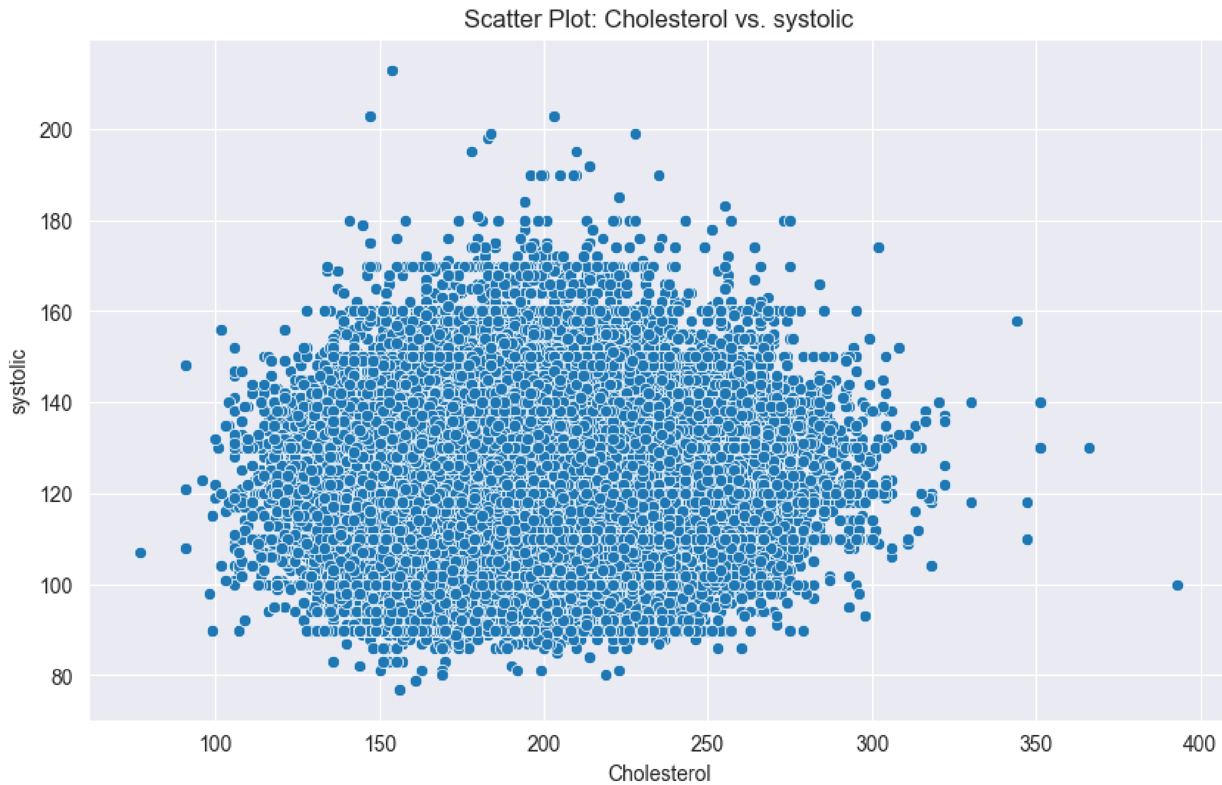

Scatter Plot: relaxation vs. systolic
systolic
200
180
160
140
120
100
80
40
60
80
100
120
relaxation


Scatter Plot: waist(cm) vs. Cholesterol
Cholesterol
waist(cm)
400
350
300
250
200
150
100
50
60
70
80
90
100
110
120
130

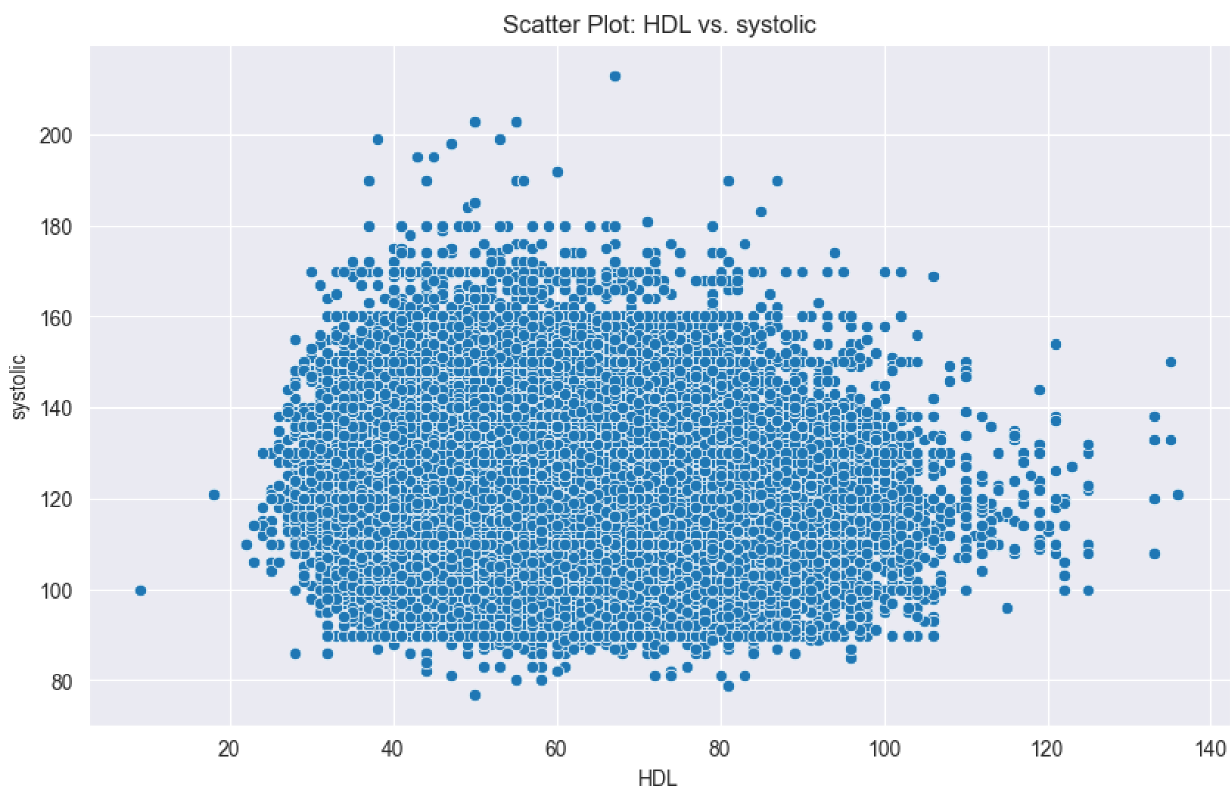

Scatter Plot: waist(cm) vs. HDL
HDL
waist(cm)
140
120
100
80
60
40
20
50
60
70
80
90
100
110
120
130


Scatter Plot: waist(cm) vs. systolic
systolic
waist(cm)
80
100
120
140
160
180
200
50
60
70
80
90
100
110
120
130


Scatter Plot: Cholesterol vs. HDL
HDL
Cholesterol
20
40
60
80
100
120
140
100
150
200
250
300
350
400

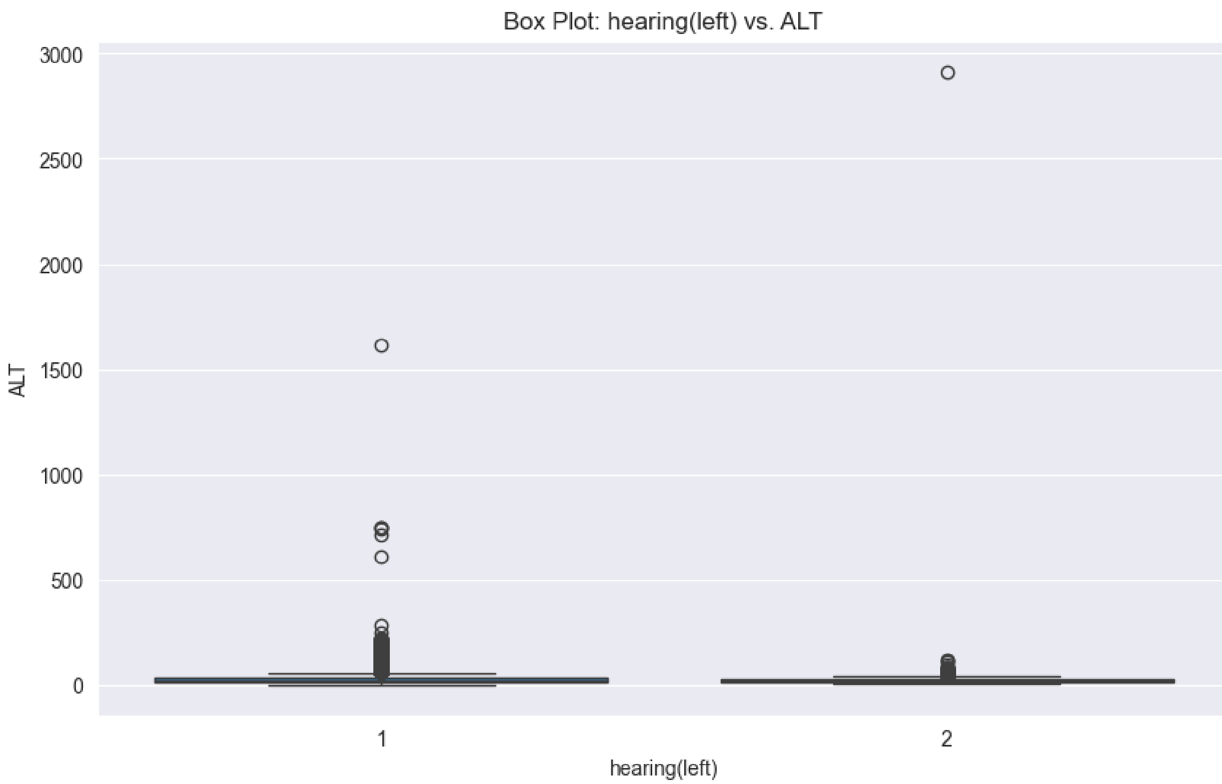


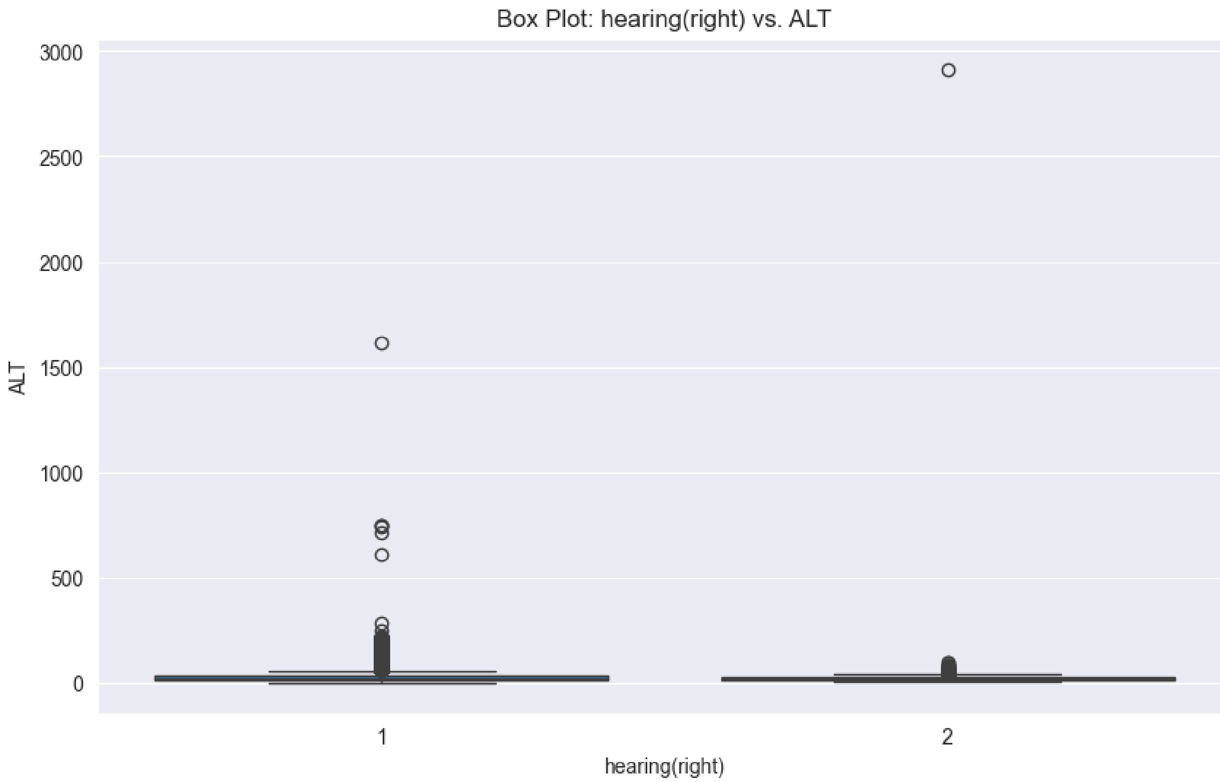
Scatter Plot: Cholesterol vs. systolic
systolic
Cholesterol
80
100
120
140
160
180
200
100
150
200
250
300
350
400

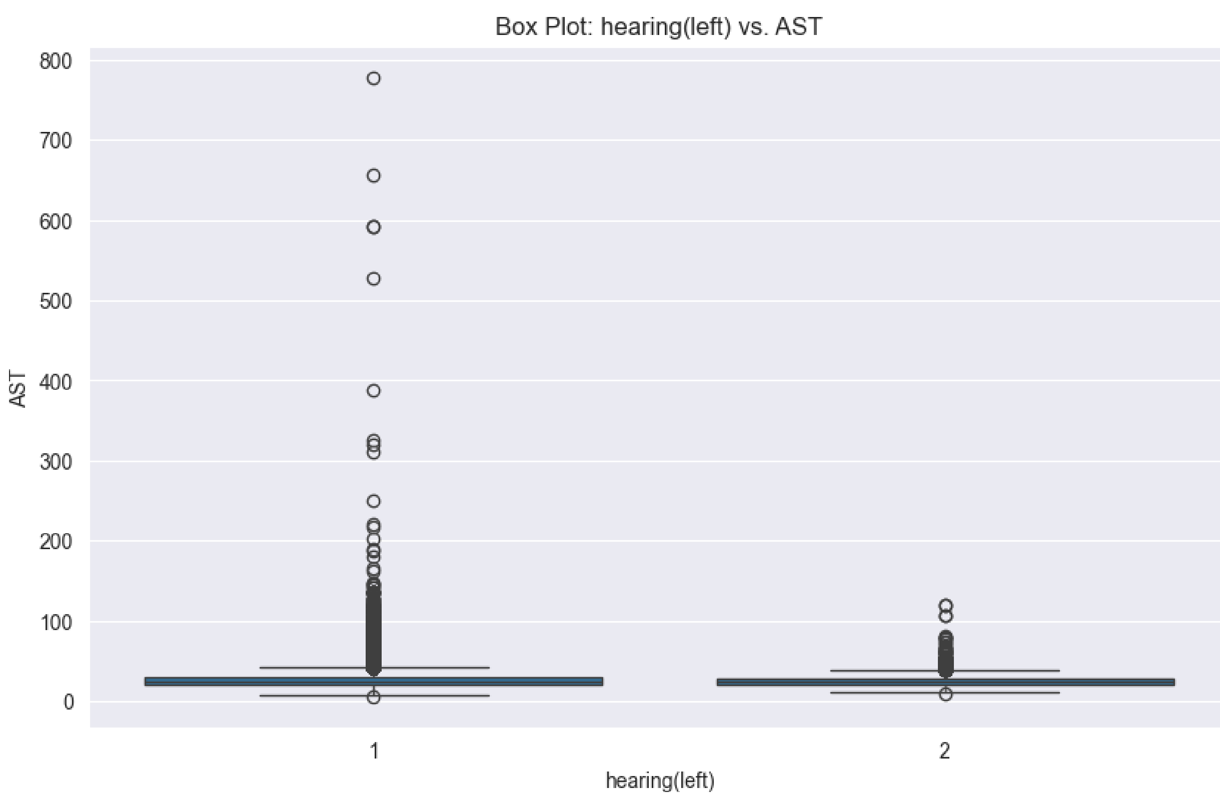


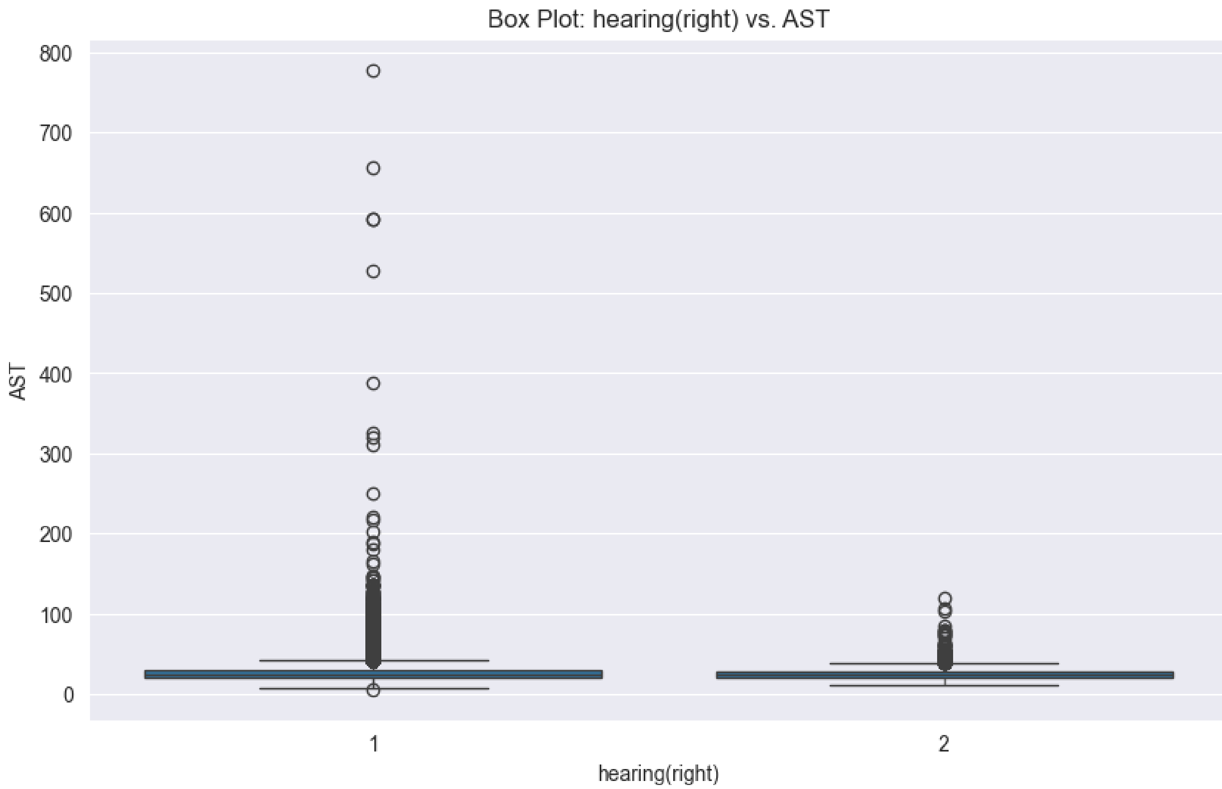
Scatter Plot: HDL vs. systolic
systolic
HDL
80
100
120
140
160
180
200
20
40
60
80
100
120
140

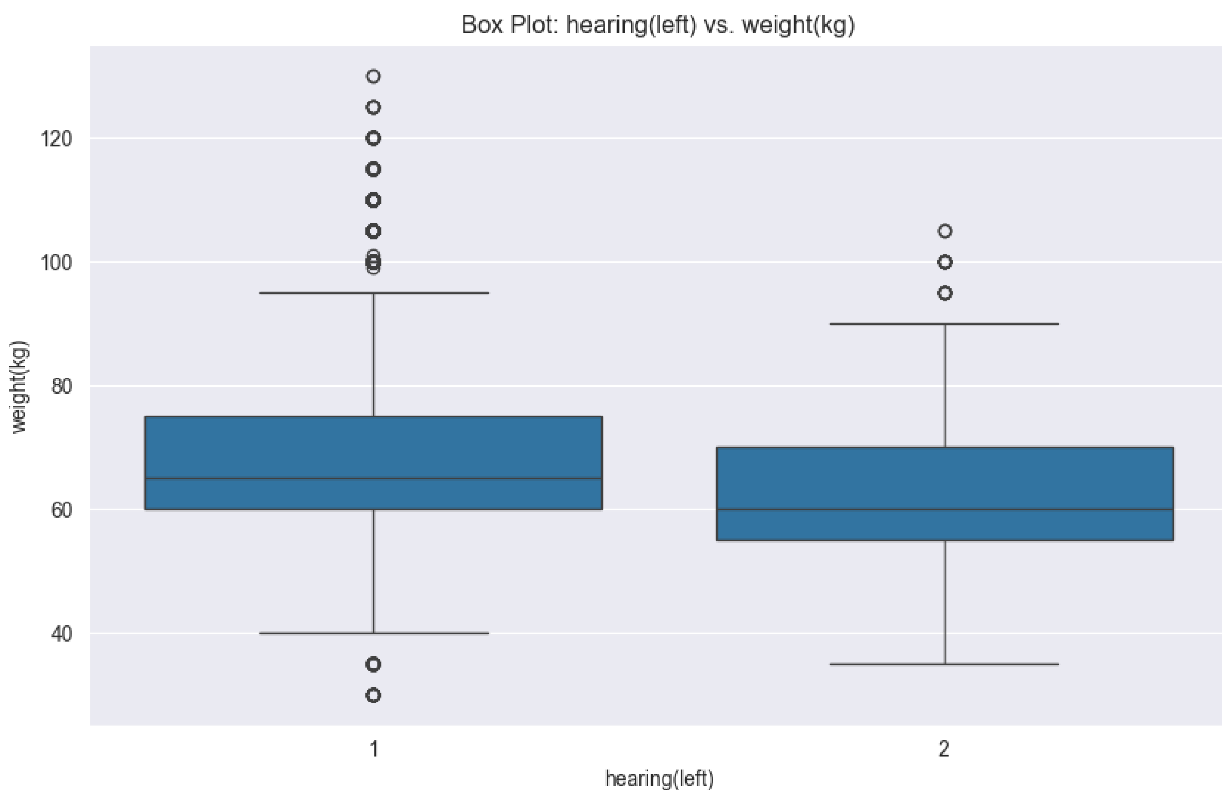
```
# Box plots for categorical-numeric relationships
categorical_features = ['hearing(left)', 'hearing(right)']

for feature in numeric_features:
    for cat_feature in categorical_features:
        plt.figure(figsize=(10, 6))
        sns.boxplot(x=cat_feature, y=feature, data=df,
                    order=df[cat_feature].value_counts().index)  #
Added order parameter
        plt.title(f'Box Plot: {cat_feature} vs. {feature}')
        plt.show()
```

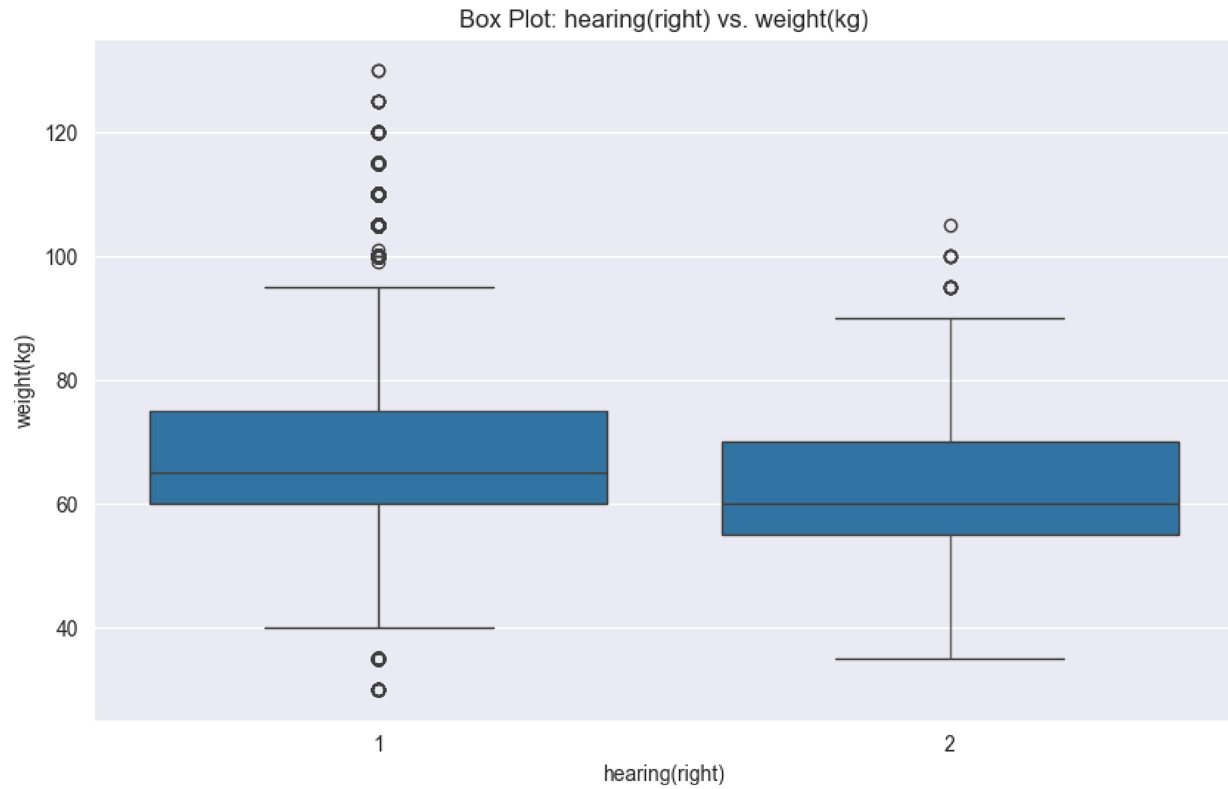


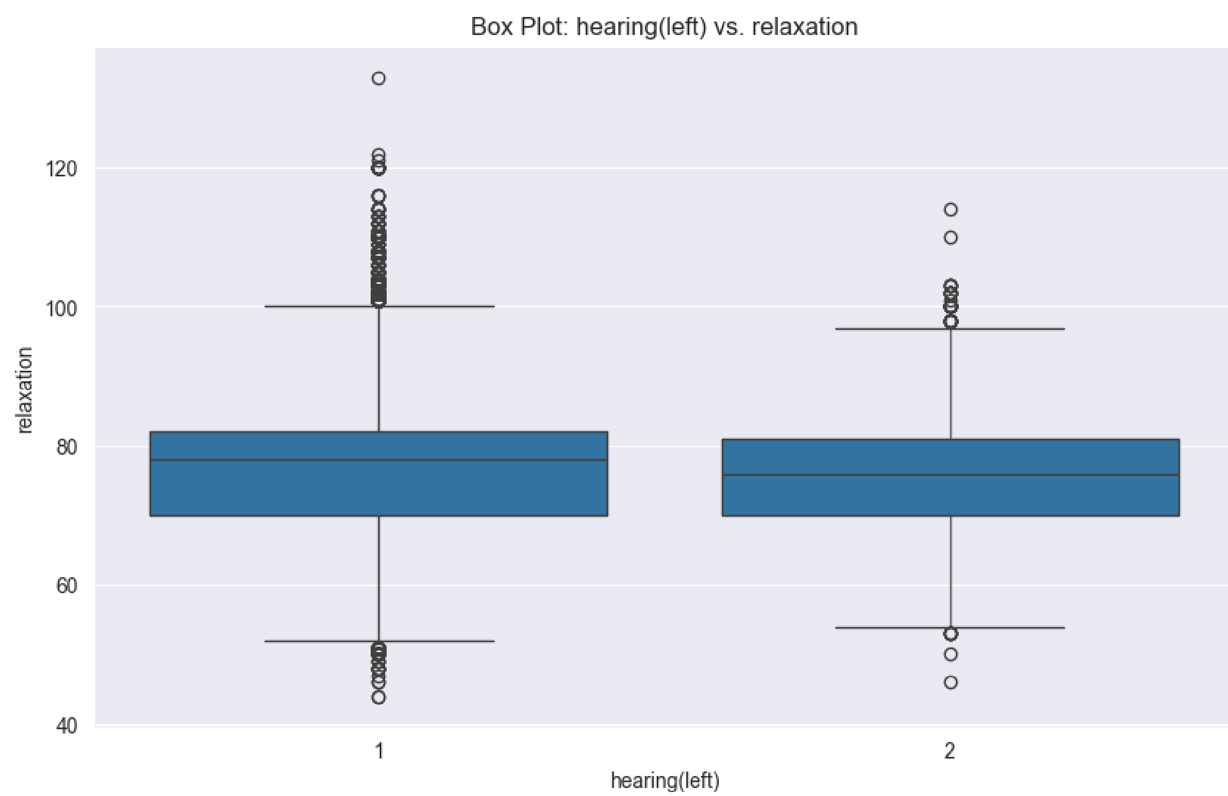

Box Plot: hearing(right) vs. ALT
3000
2500
2000
1500
1000
500
0
ALT
1
2
hearing(right)

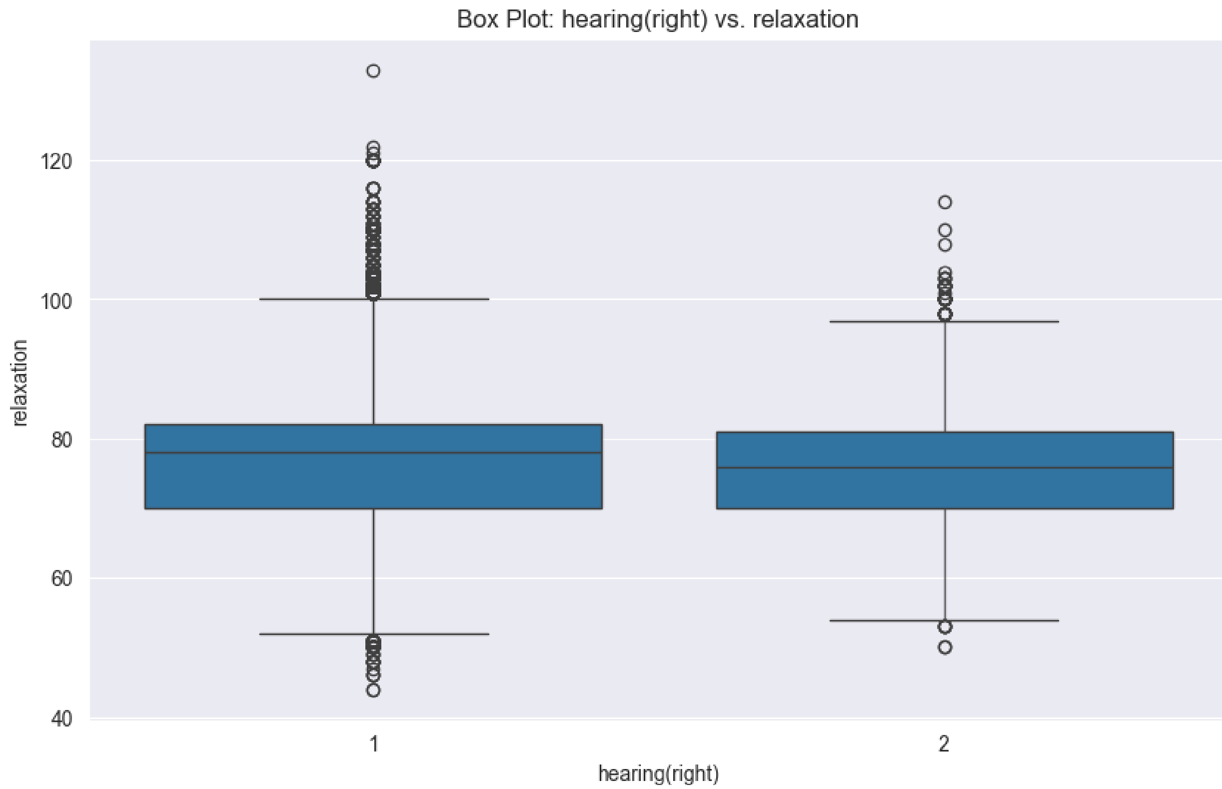

Box Plot: hearing(left) vs. AST
800
700
600
500
400
300
200
100
0
AST
1
2
hearing(left)

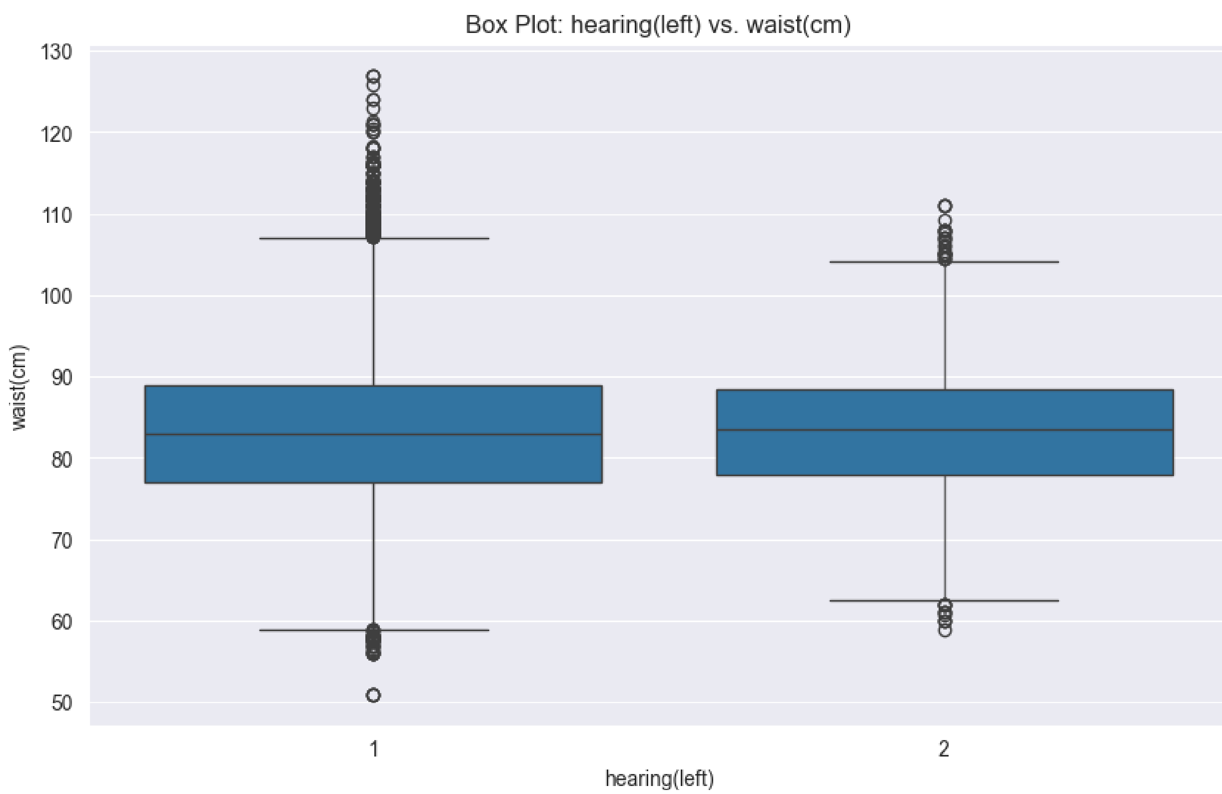

Box Plot: hearing(right) vs. AST
AST
800
700
600
500
400
300
200
100
0
1
2
hearing(right)

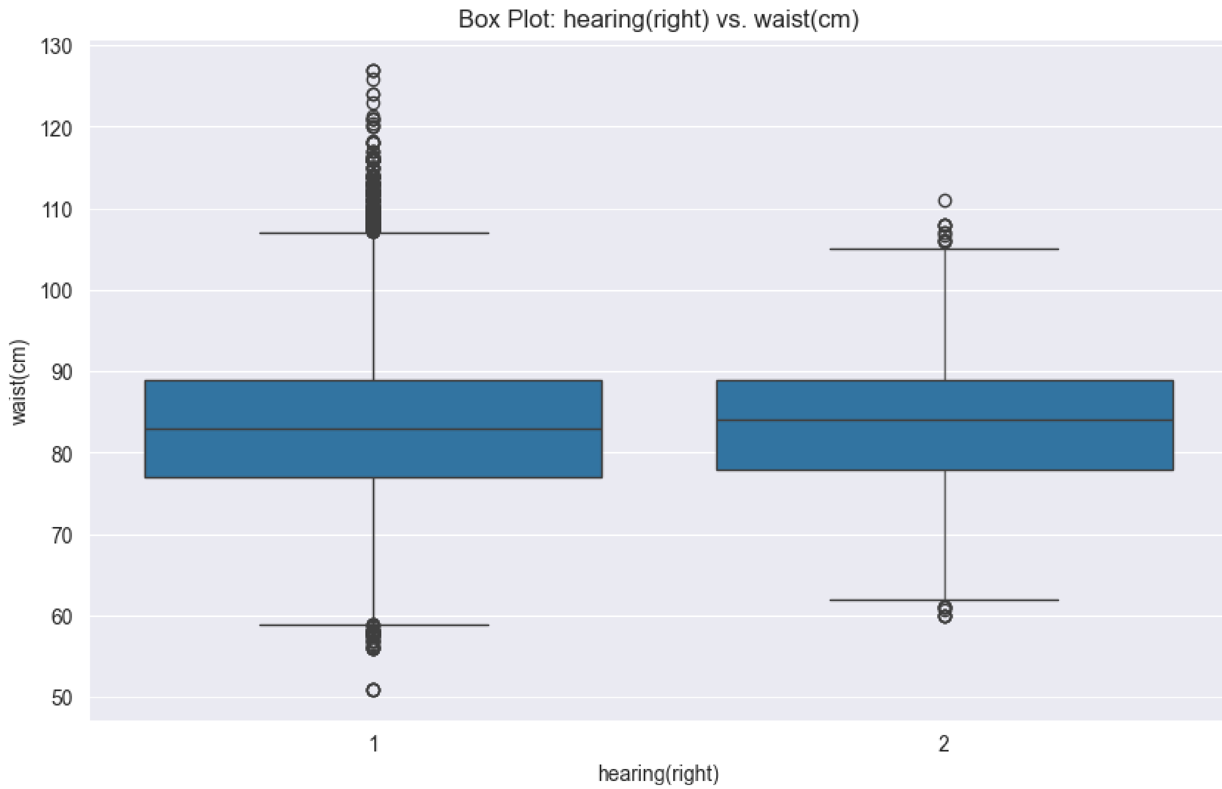

Box Plot: hearing(left) vs. weight(kg)
weight(kg)
120
100
80
60
40
1
2
hearing(left)

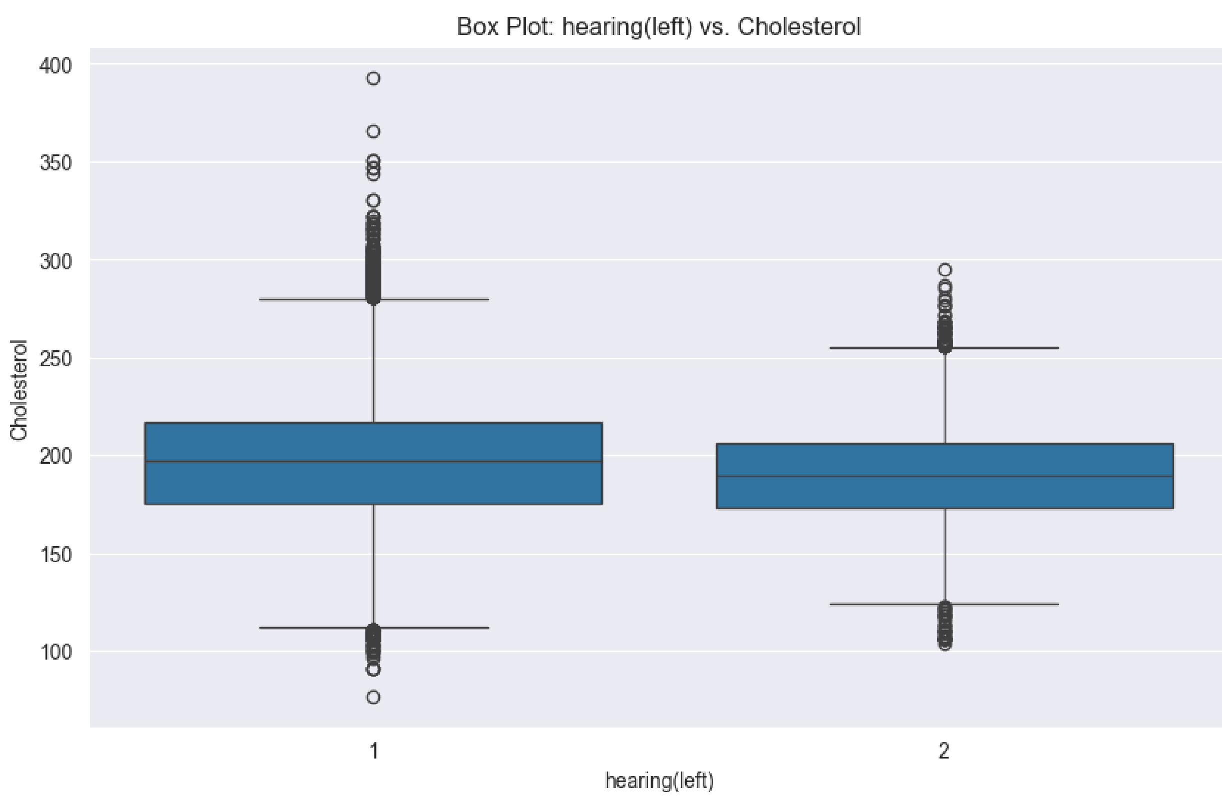

Box Plot: hearing(right) vs. weight(kg)
weight(kg)
hearing(right)
1
2
40
60
80
100
120

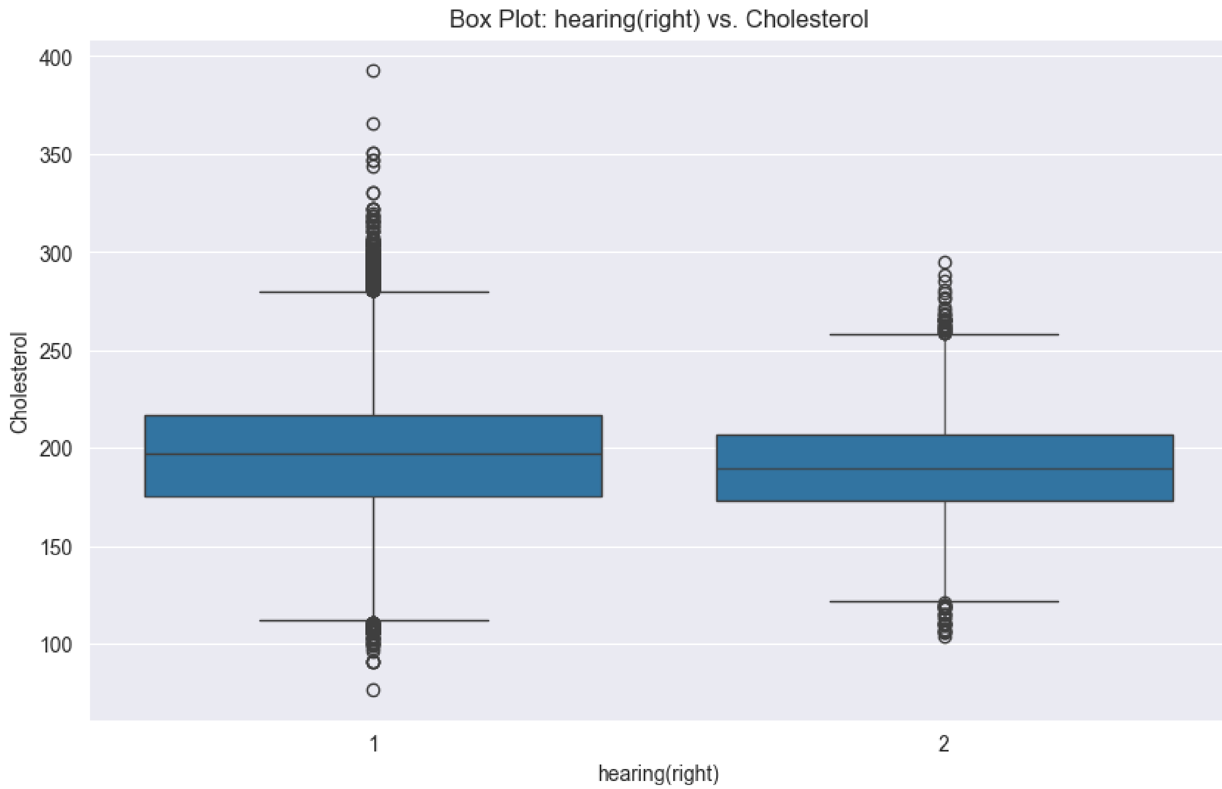

Box Plot: hearing(left) vs. relaxation
relaxation
hearing(left)
1
2
40
60
80
100
120


Box Plot: hearing(right) vs. relaxation
relaxation
hearing(right)
120
100
80
60
40
1
2

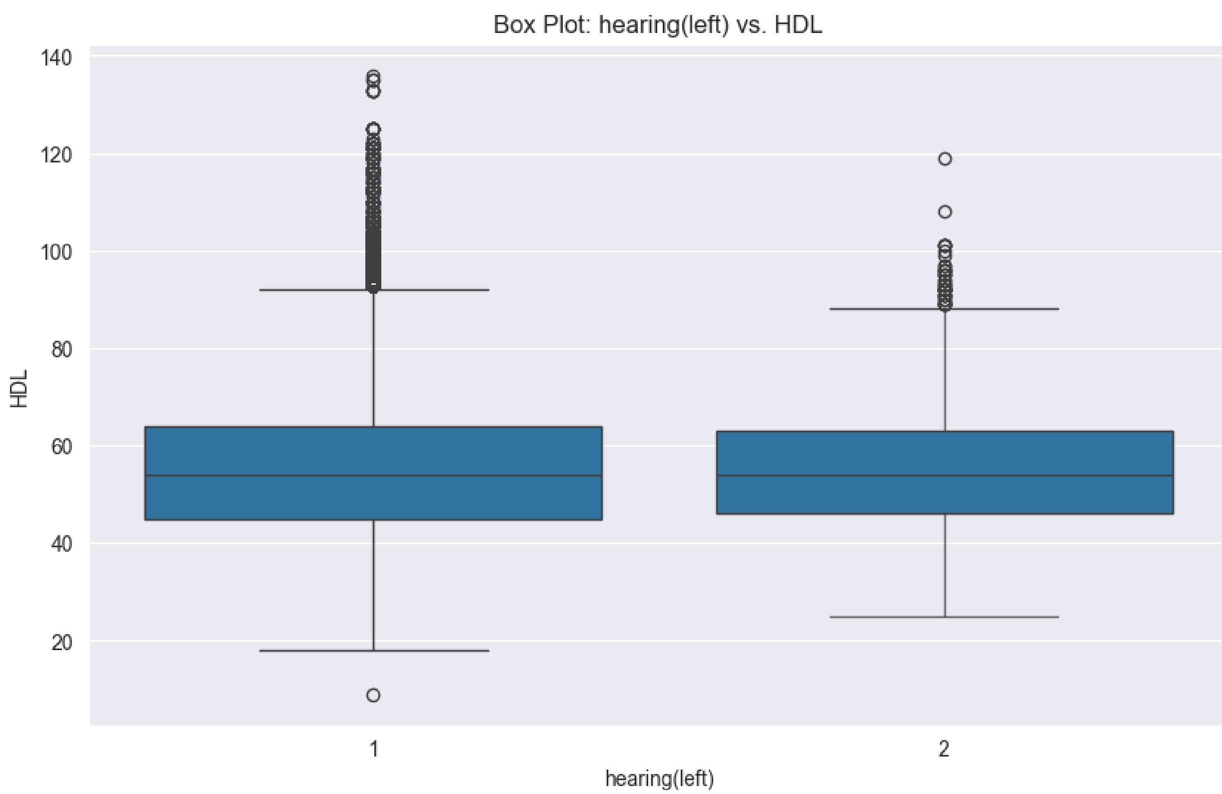

Box Plot: hearing(left) vs. waist(cm)
waist(cm)
hearing(left)
130
120
110
100
90
80
70
60
50
1
2


Box Plot: hearing(right) vs. waist(cm)
waist(cm)
130
120
110
100
90
80
70
60
50
1
2
hearing(right)

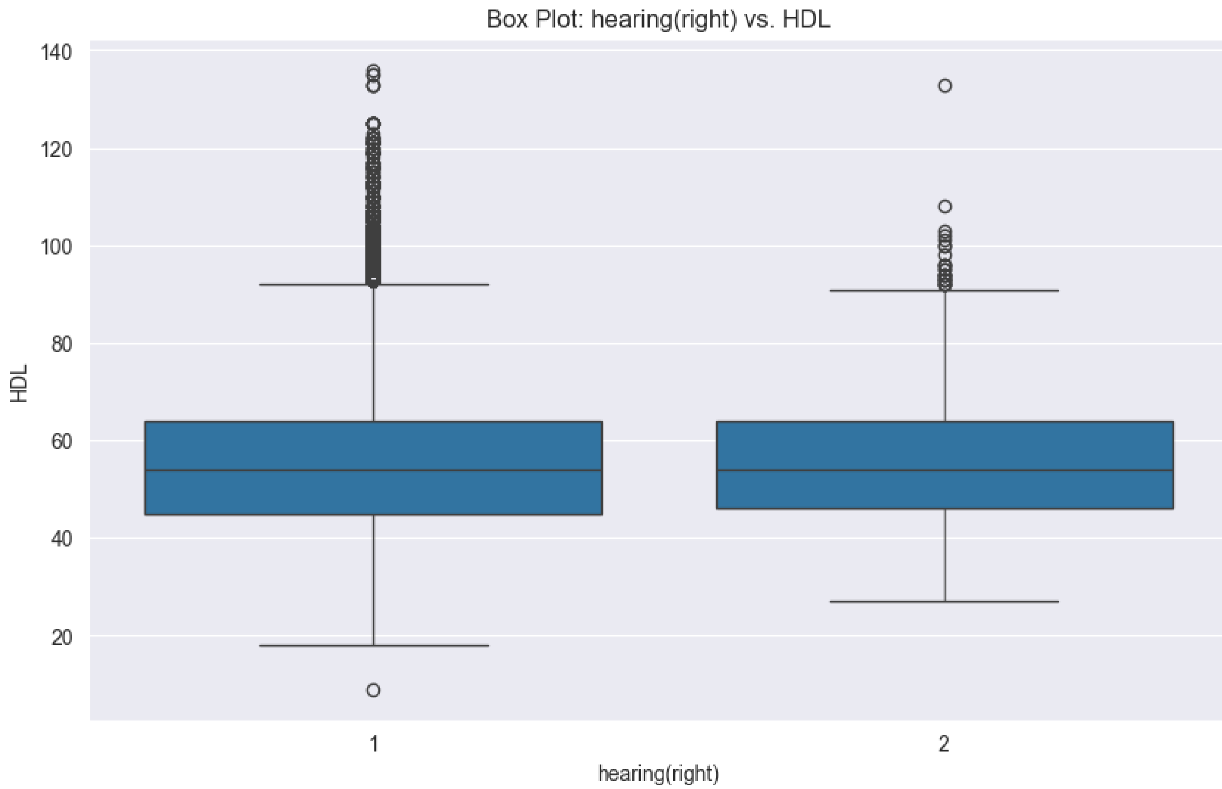

Box Plot: hearing(left) vs. Cholesterol
Cholesterol
400
350
300
250
200
150
100
1
2
hearing(left)



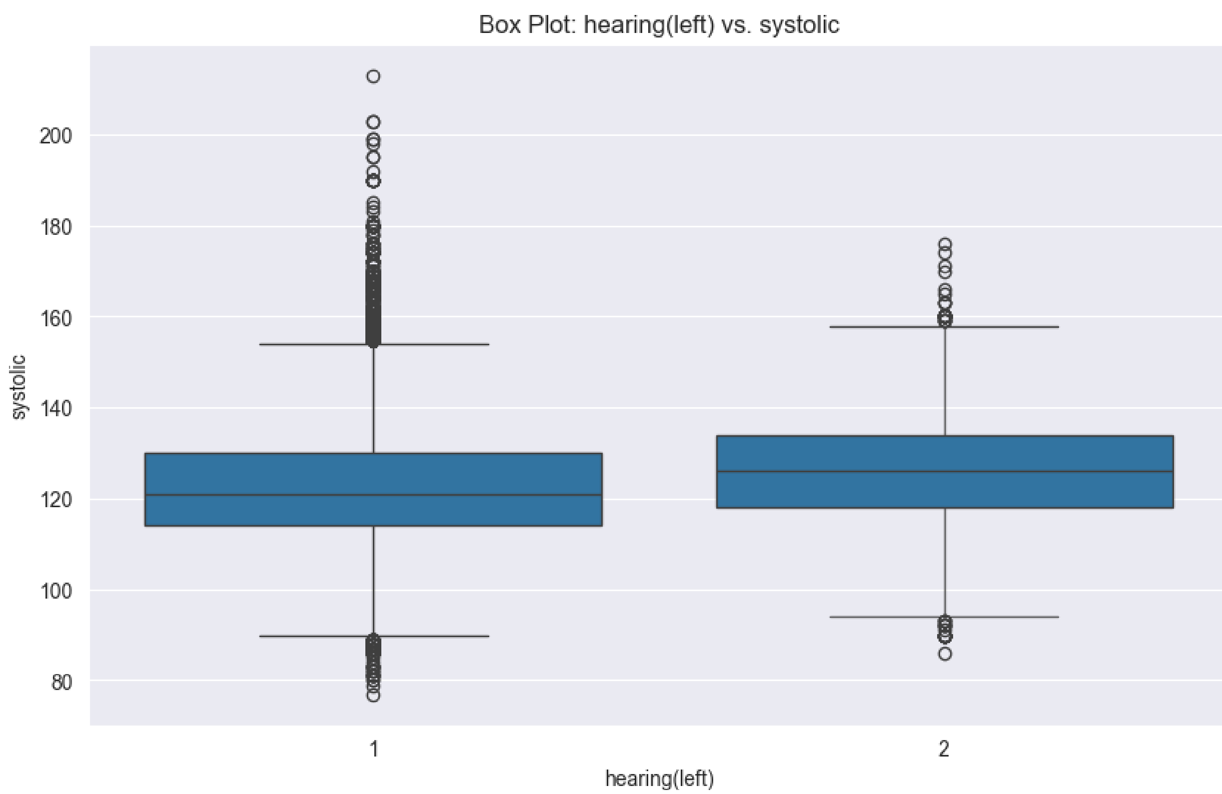
Box Plot: hearing(right) vs. Cholesterol
400
350
300
250
200
150
100
Cholesterol
1
2
hearing(right)



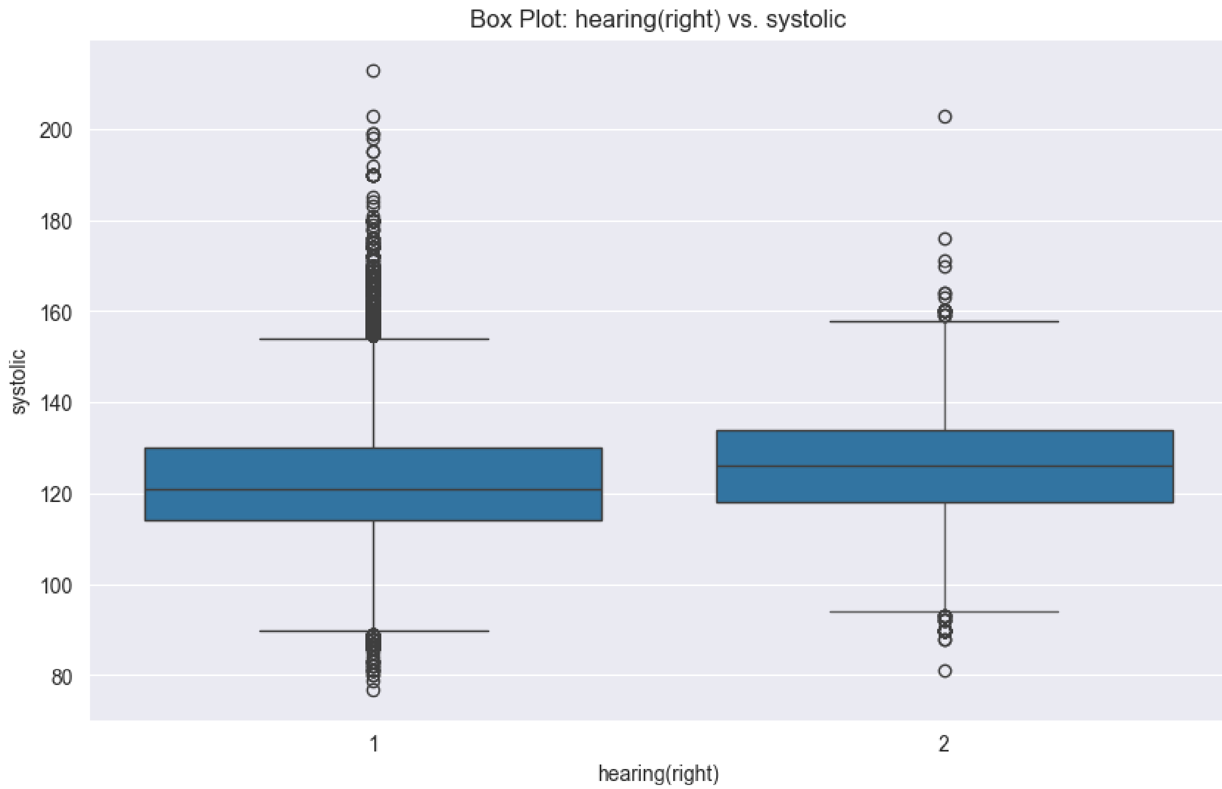
Box Plot: hearing(left) vs. HDL
140
120
100
80
60
40
20
HDL
1
2
hearing(left)



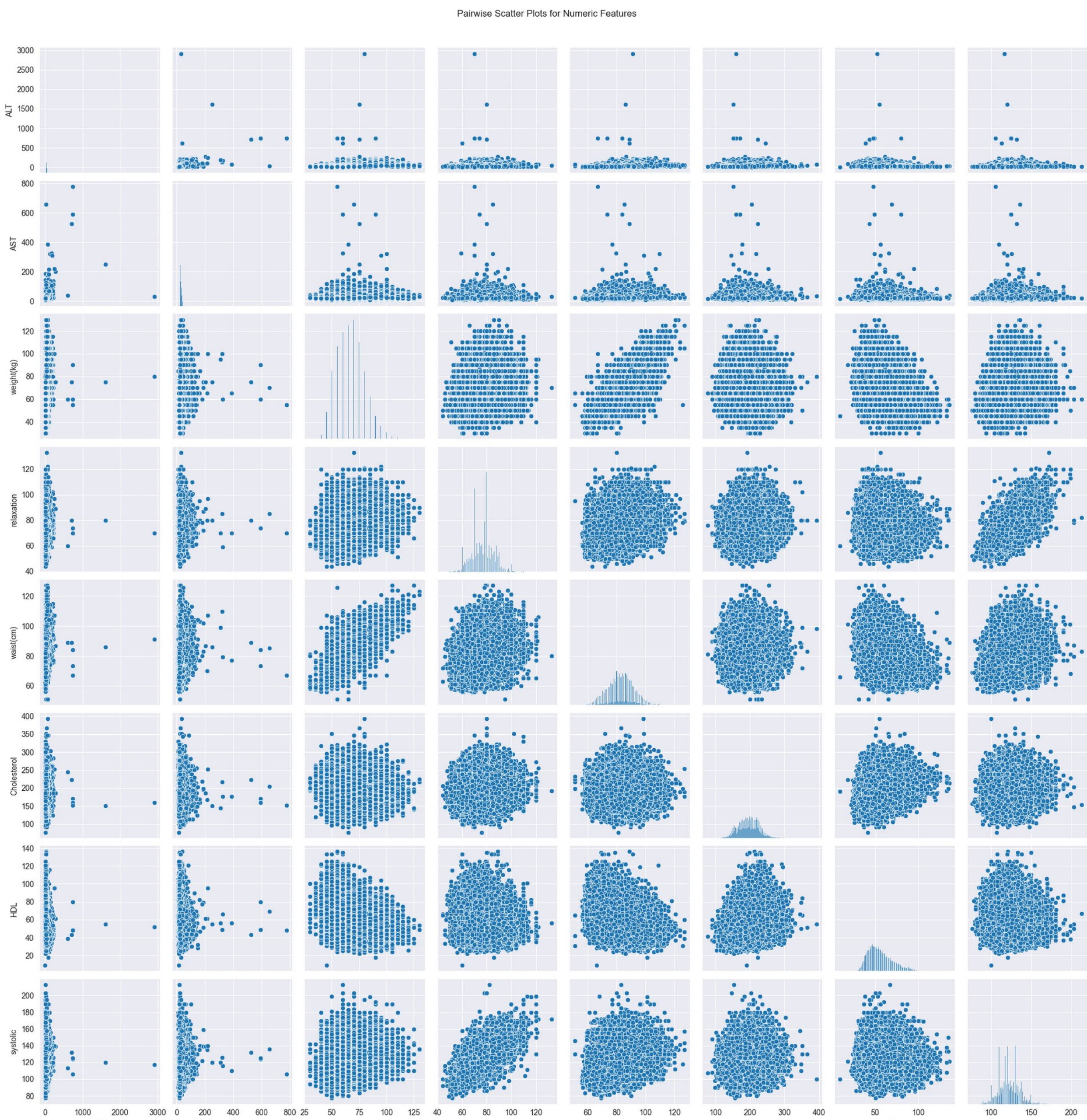
Box Plot: hearing(right) vs. HDL
HDL
140
120
100
80
60
40
20
1
2
hearing(right)



Box Plot: hearing(left) vs. systolic
systolic
200
180
160
140
120
100
80
1
2
hearing(left)

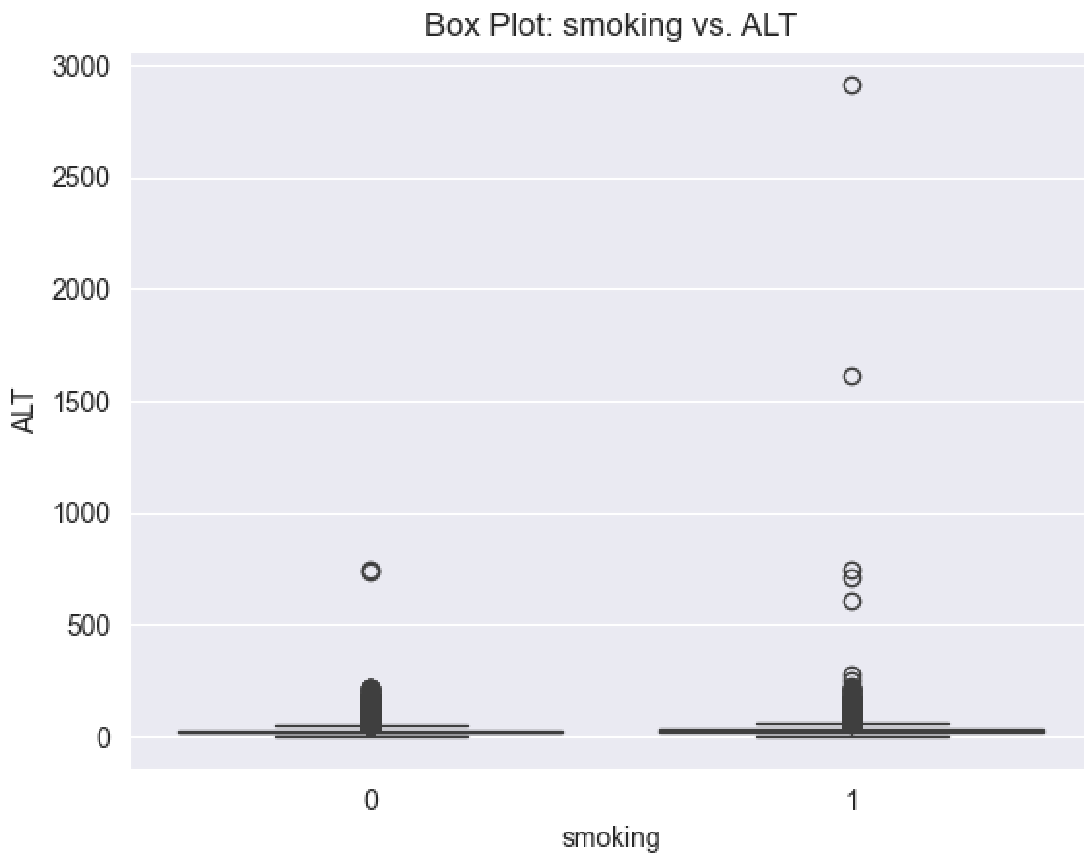


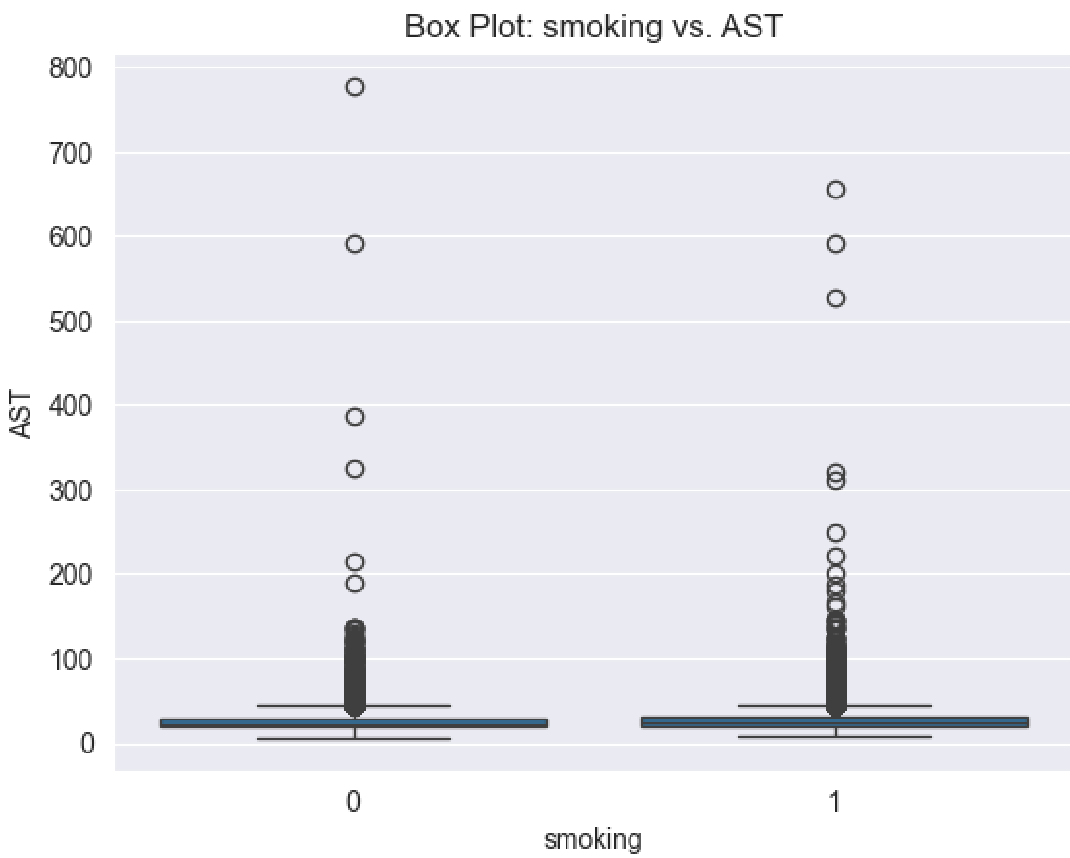

```
# Pairwise scatter plots for numeric-numeric relationships
sns.pairplot(df[['ALT', 'AST', 'weight(kg)', 'relaxation',
'waist(cm)', 'Cholesterol', 'HDL', 'systolic']])
plt.suptitle('Pairwise Scatter Plots for Numeric Features', y=1.02)
plt.show()
```



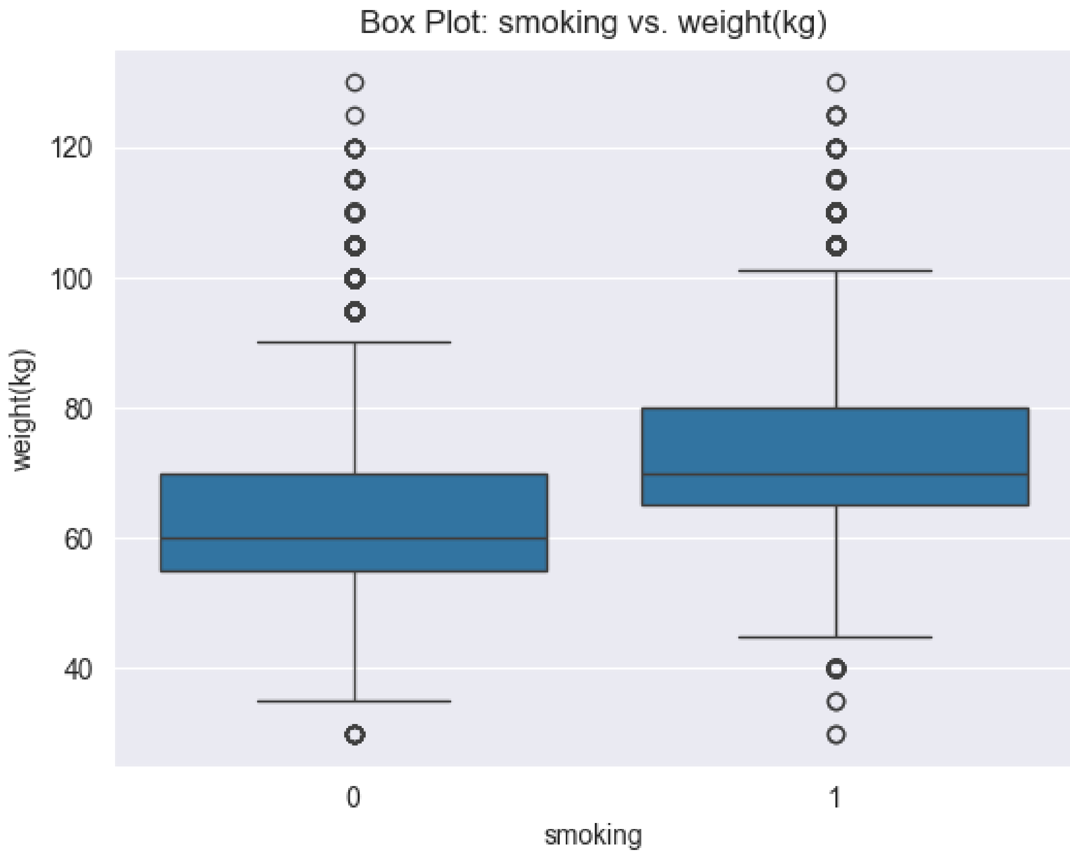

```python
# Box plots for categorical-numeric relationships (e.g., Smoking vs. Numeric Features)
numeric_features = ['ALT', 'AST', 'weight(kg)', 'relaxation', 'waist(cm)', 'Cholesterol', 'HDL', 'systolic']
categorical_feature = 'smoking'

for feature in numeric_features:
    sns.boxplot(x=categorical_feature, y=feature, data=df,
                order=df[categorical_feature].value_counts().index)  # Added order parameter
```

```
    plt.title(f'Box Plot: {categorical_feature} vs. {feature}')
    plt.show()
```

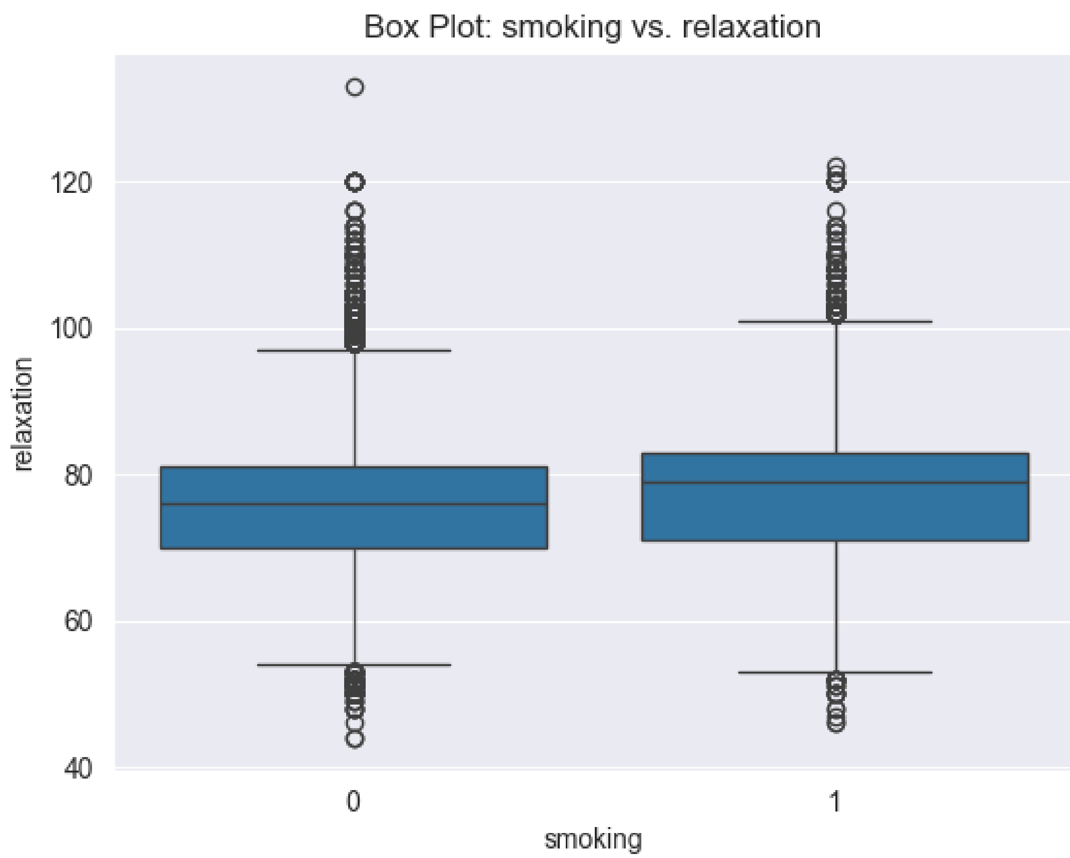


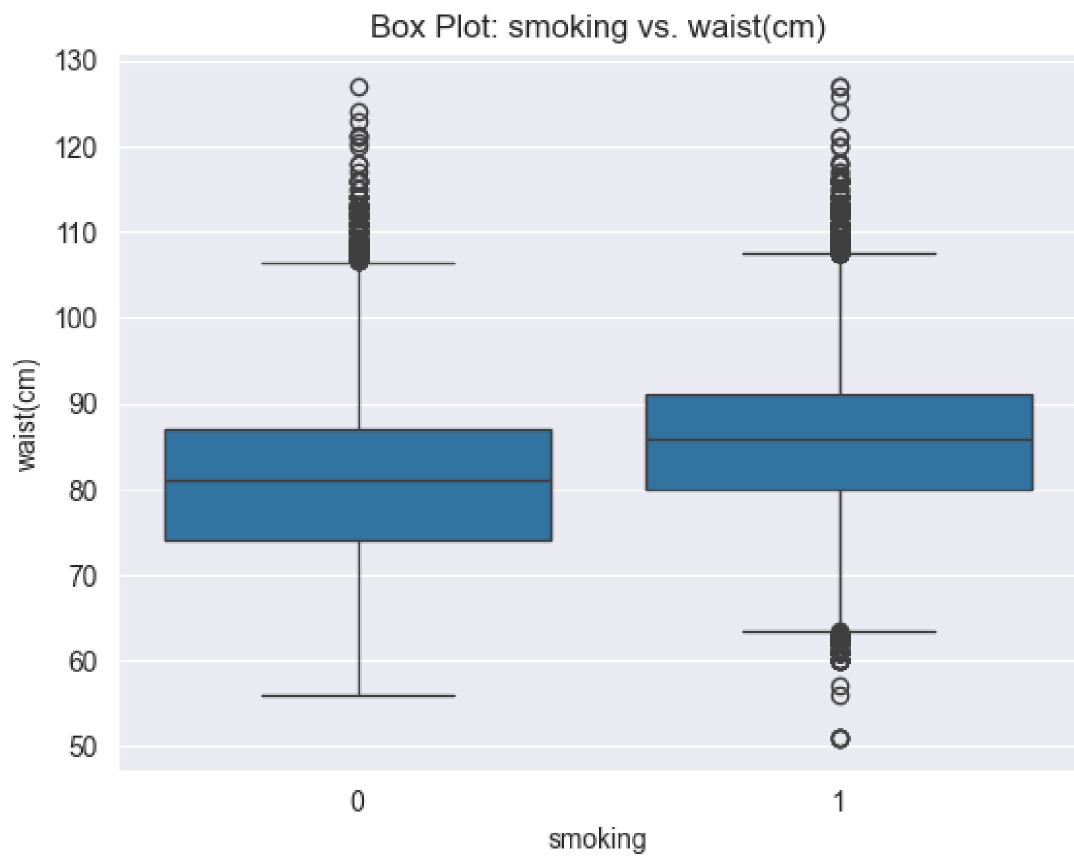


Box Plot: smoking vs. AST
AST
800
700
600
500
400
300
200
100
0
0
1
smoking

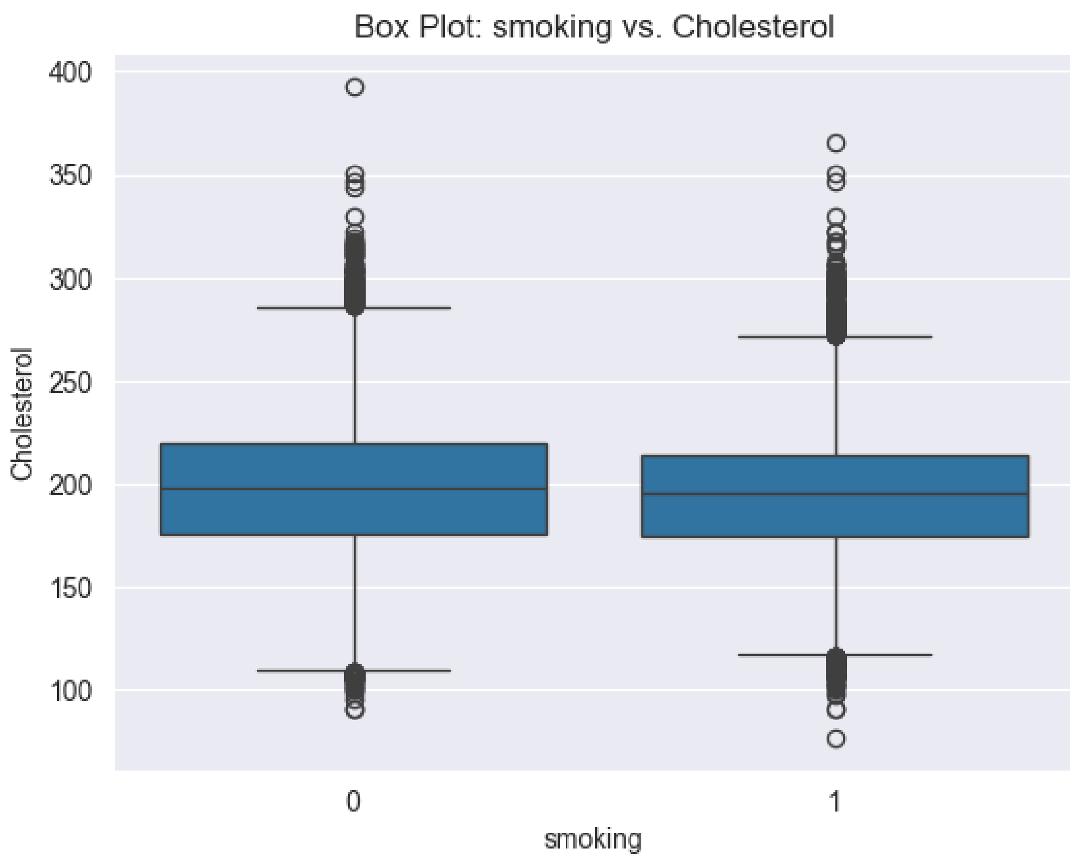


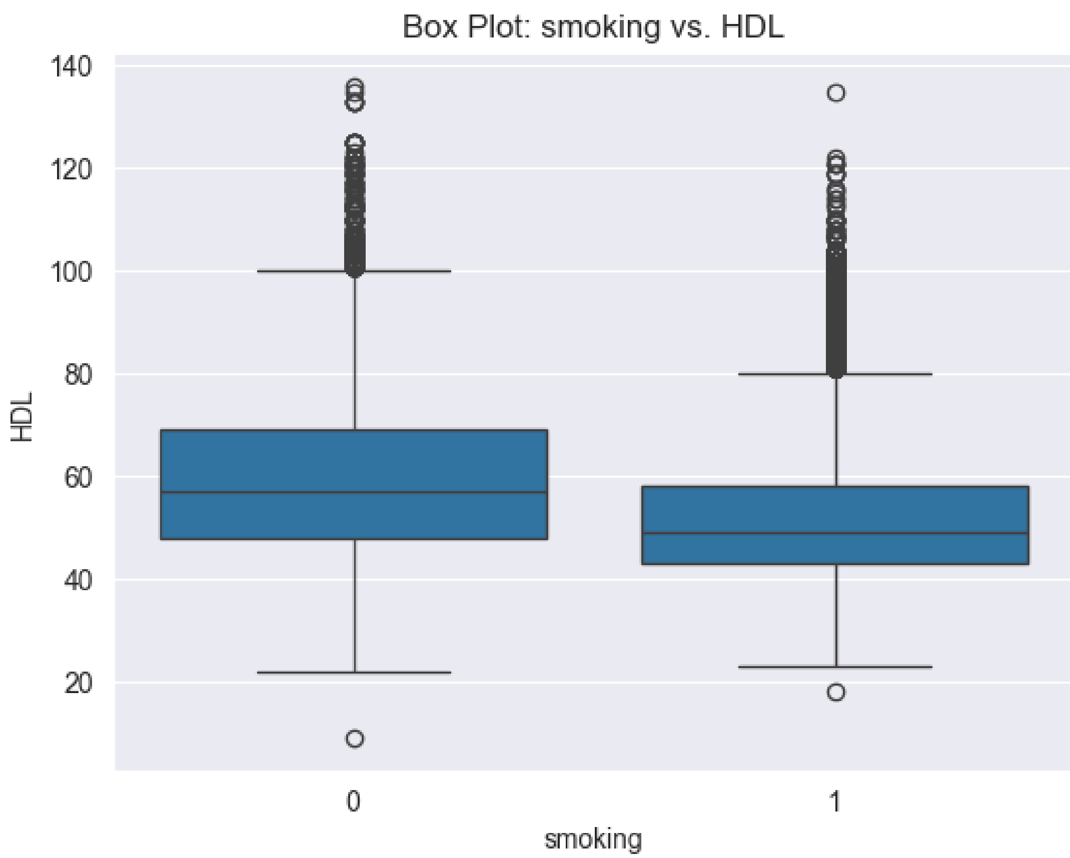
Box Plot: smoking vs. weight(kg)
weight(kg)
120
100
80
60
40
0
1
smoking


Box Plot: smoking vs. relaxation
relaxation
120
100
80
60
40
0
1
smoking

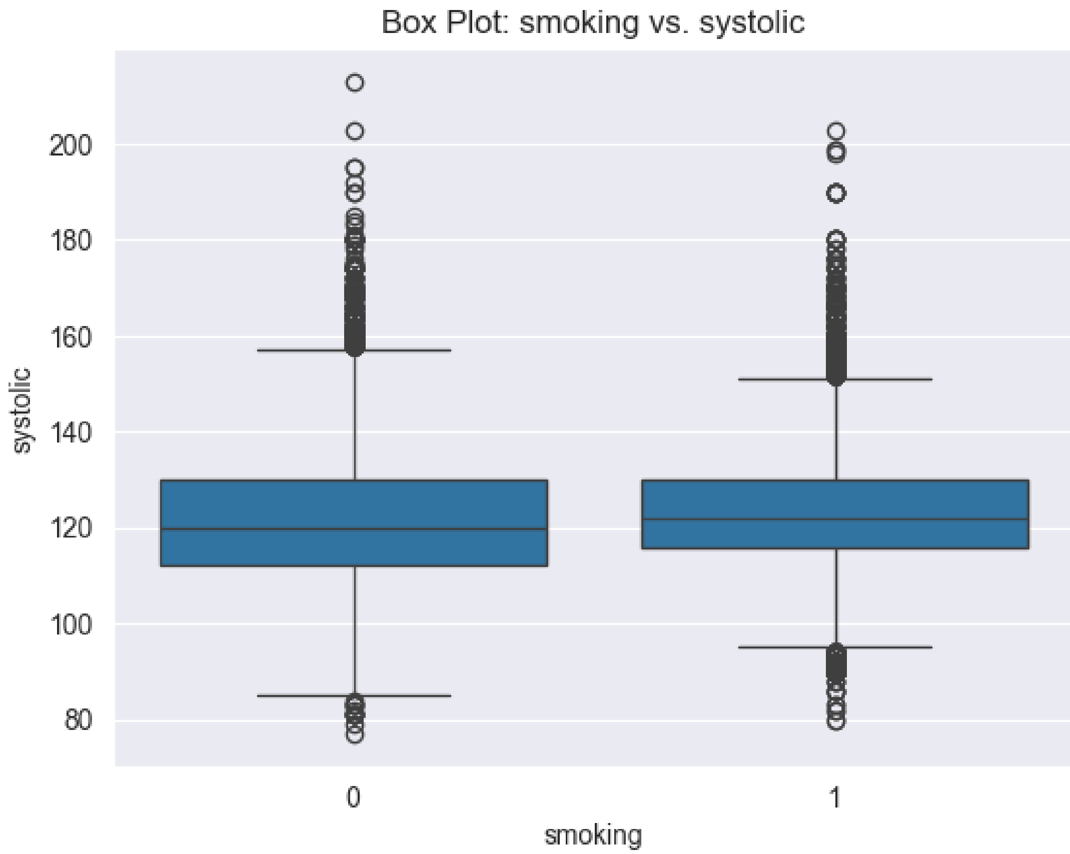


Box Plot: smoking vs. waist(cm)
130
120
110
100
90
80
70
60
50
waist(cm)
0
1
smoking

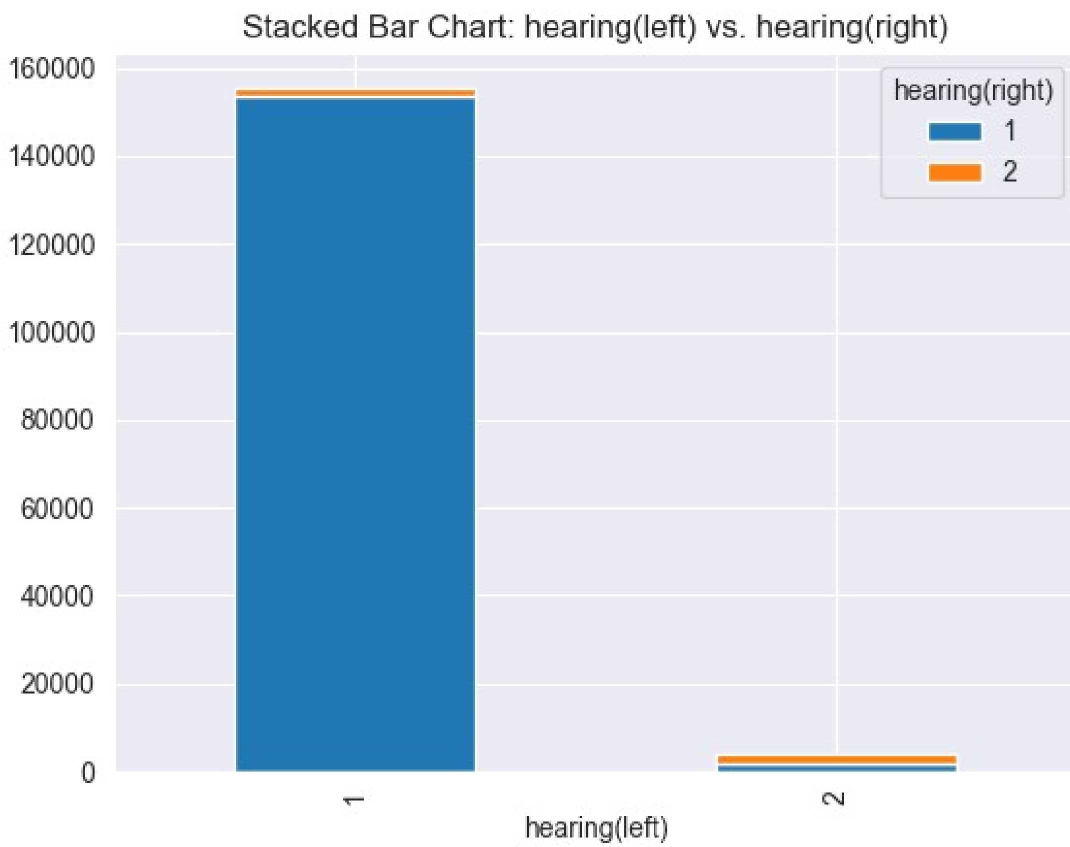

Box Plot: smoking vs. Cholesterol
400
350
300
250
200
150
100
Cholesterol
0
1
smoking



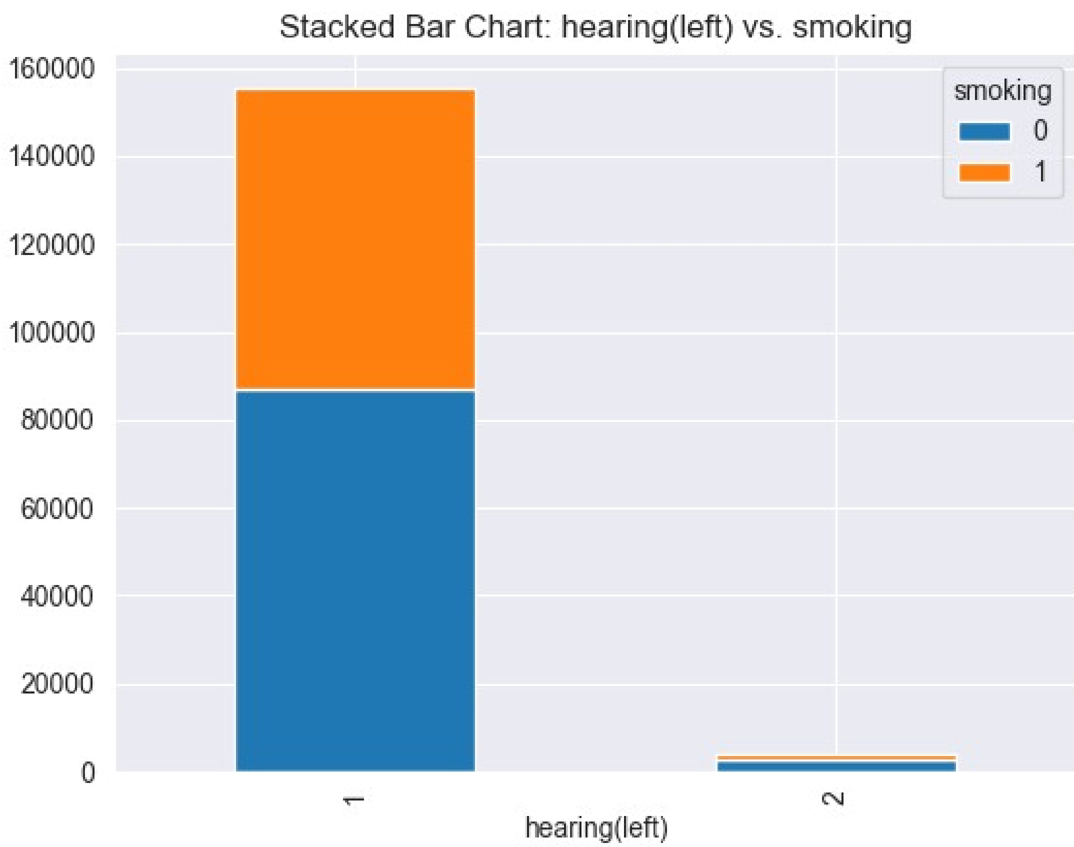
Box Plot: smoking vs. HDL
140
120
100
80
60
40
20
HDL
0
1
smoking

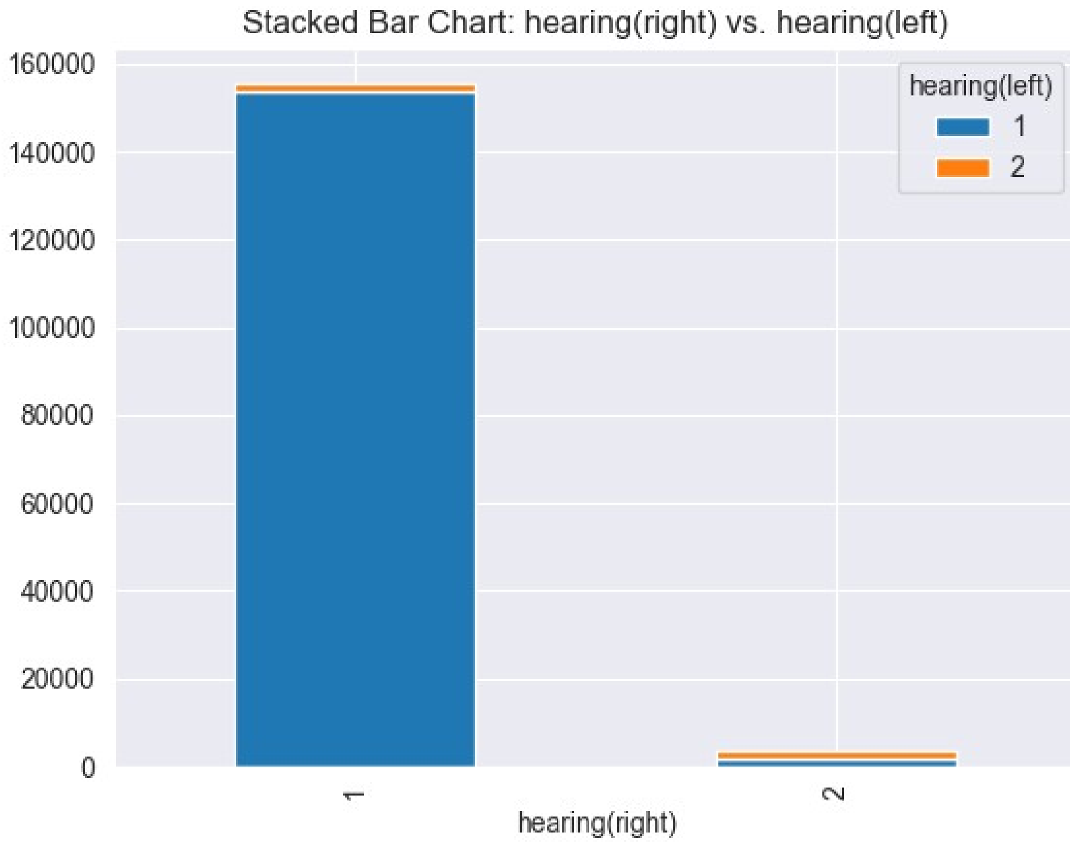



```
# Stacked bar charts for categorical-categorical relationships (e.g.,
Hearing vs. Smoking)
categorical_features = ['hearing(left)', 'hearing(right)', 'smoking']

for feature1 in categorical_features:
    for feature2 in categorical_features:
        if feature1 != feature2:
            cross_tab = pd.crosstab(df[feature1], df[feature2])
            cross_tab.plot(kind='bar', stacked=True)
            plt.title(f'Stacked Bar Chart: {feature1} vs. {feature2}')
            plt.show()
```

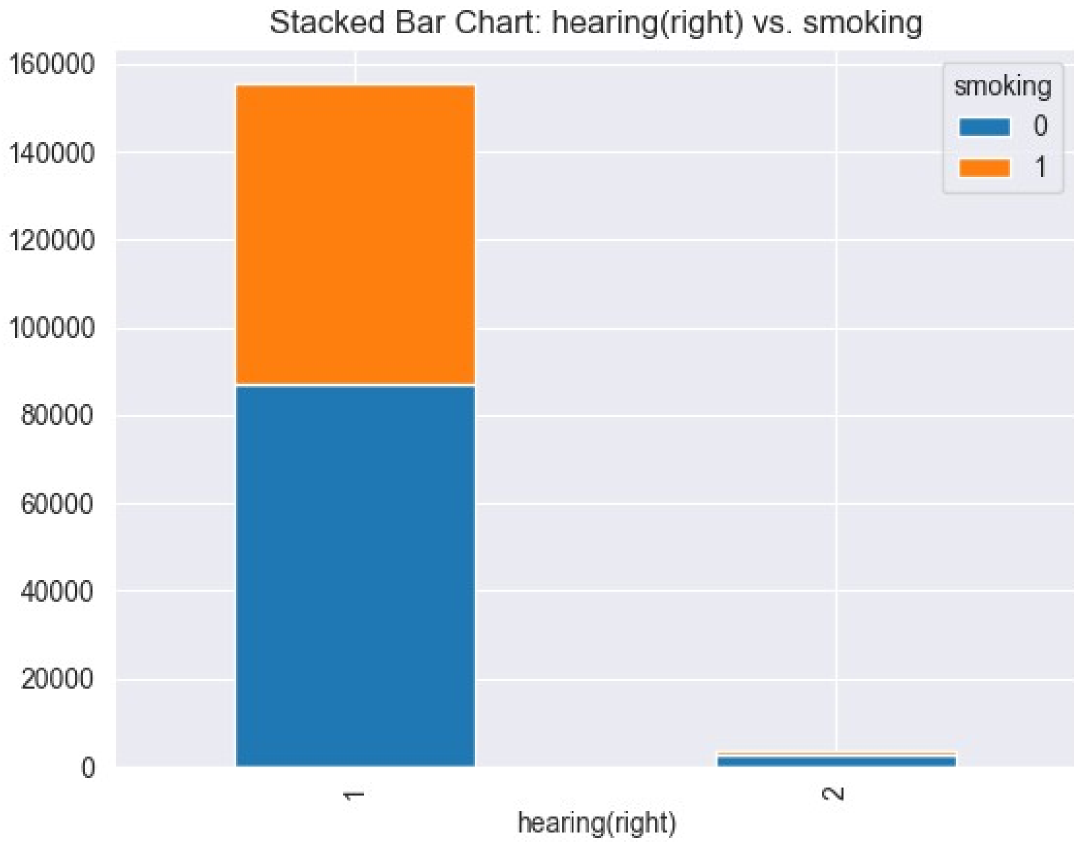


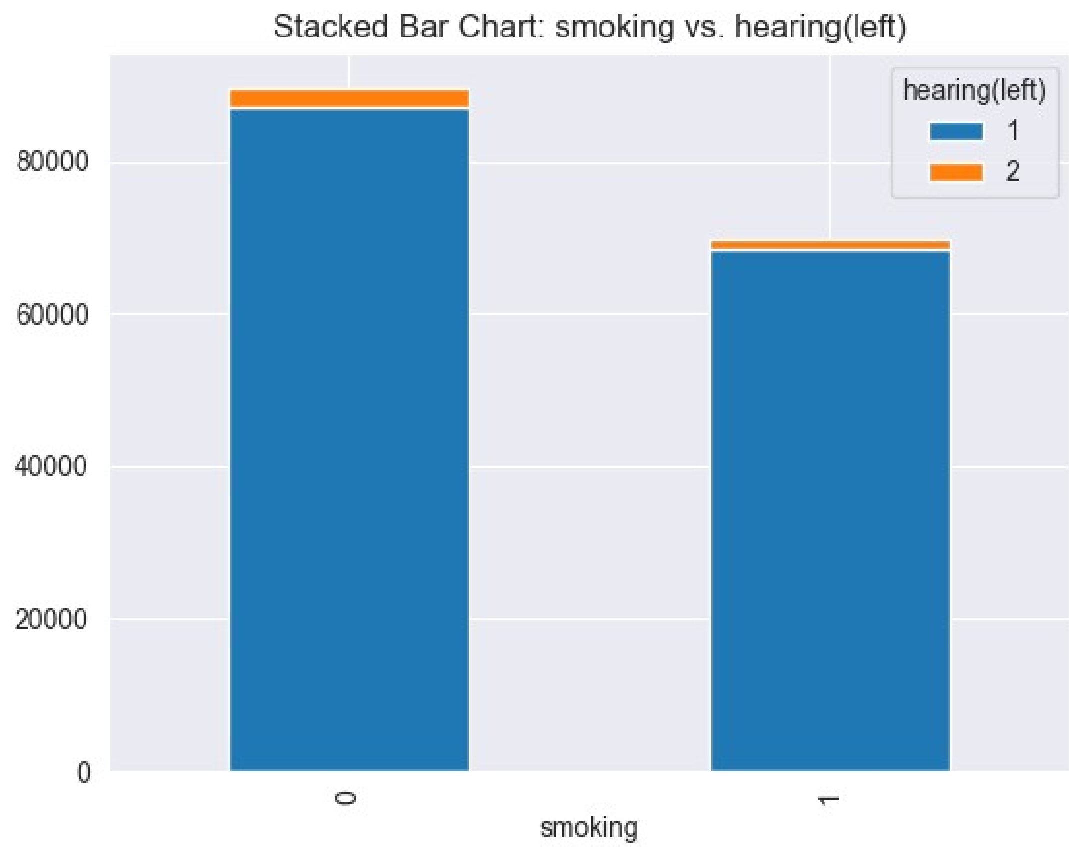
Stacked Bar Chart: hearing(left) vs. hearing(right)
160000
140000
120000
100000
80000
60000
40000
20000
0
hearing(right)
1
2
1
2
hearing(left)

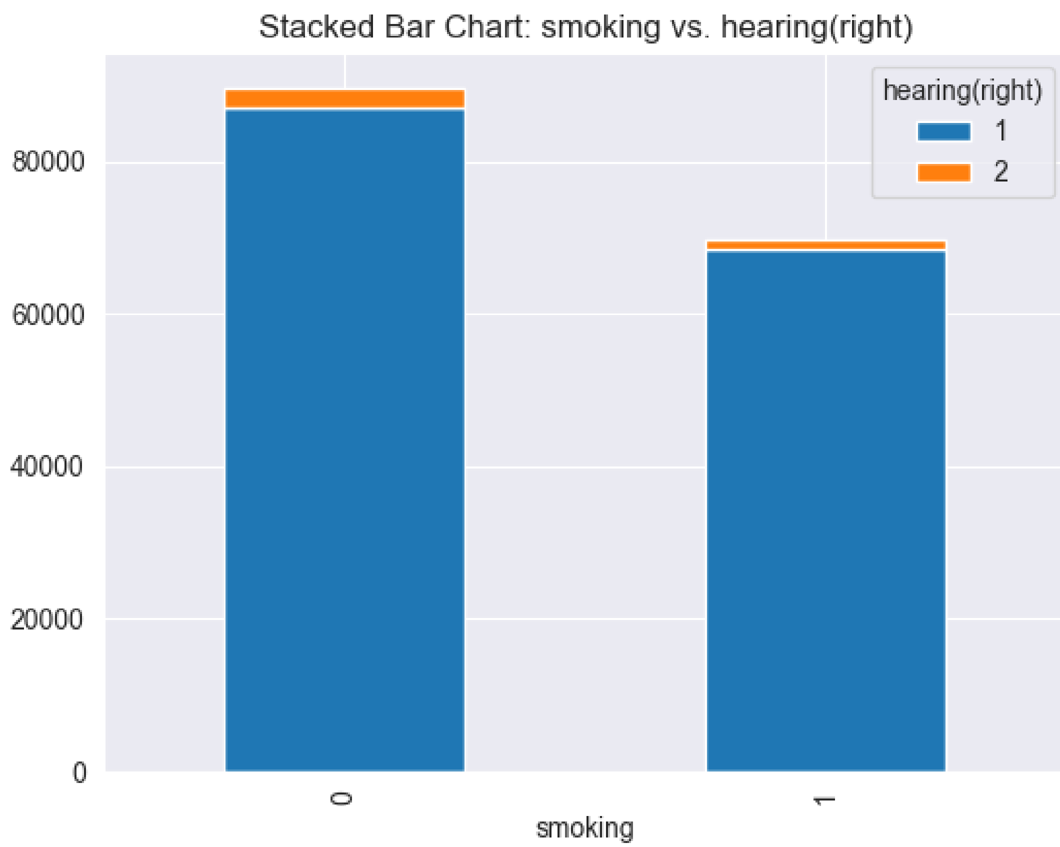

Stacked Bar Chart: hearing(left) vs. smoking
smoking
0
1
160000
140000
120000
100000
80000
60000
40000
20000
0
1
2
hearing(left)



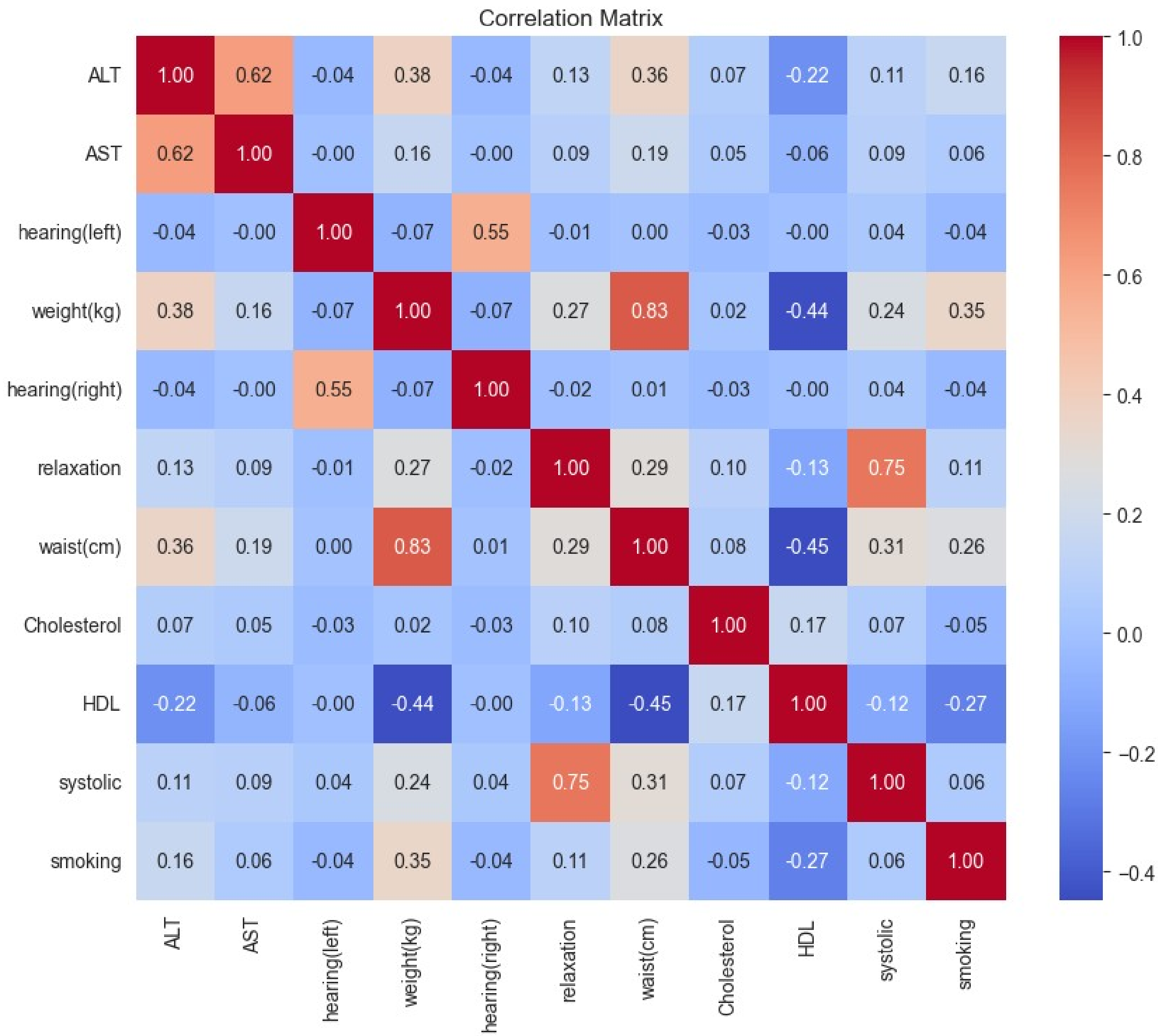
Stacked Bar Chart: hearing(right) vs. hearing(left)
hearing(left)
1
2
0
20000
40000
60000
80000
100000
120000
140000
160000
1
2
hearing(right)


Stacked Bar Chart: hearing(right) vs. smoking
160000
140000
120000
100000
80000
60000
40000
20000
0
smoking
0
1
1
2
hearing(right)


Stacked Bar Chart: smoking vs. hearing(left)
hearing(left)
1
2
80000
60000
40000
20000
0
0
1
smoking




```
df = df.drop(df.columns[0], axis=1)
correlation_matrix = df.corr()

# Visualization using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```

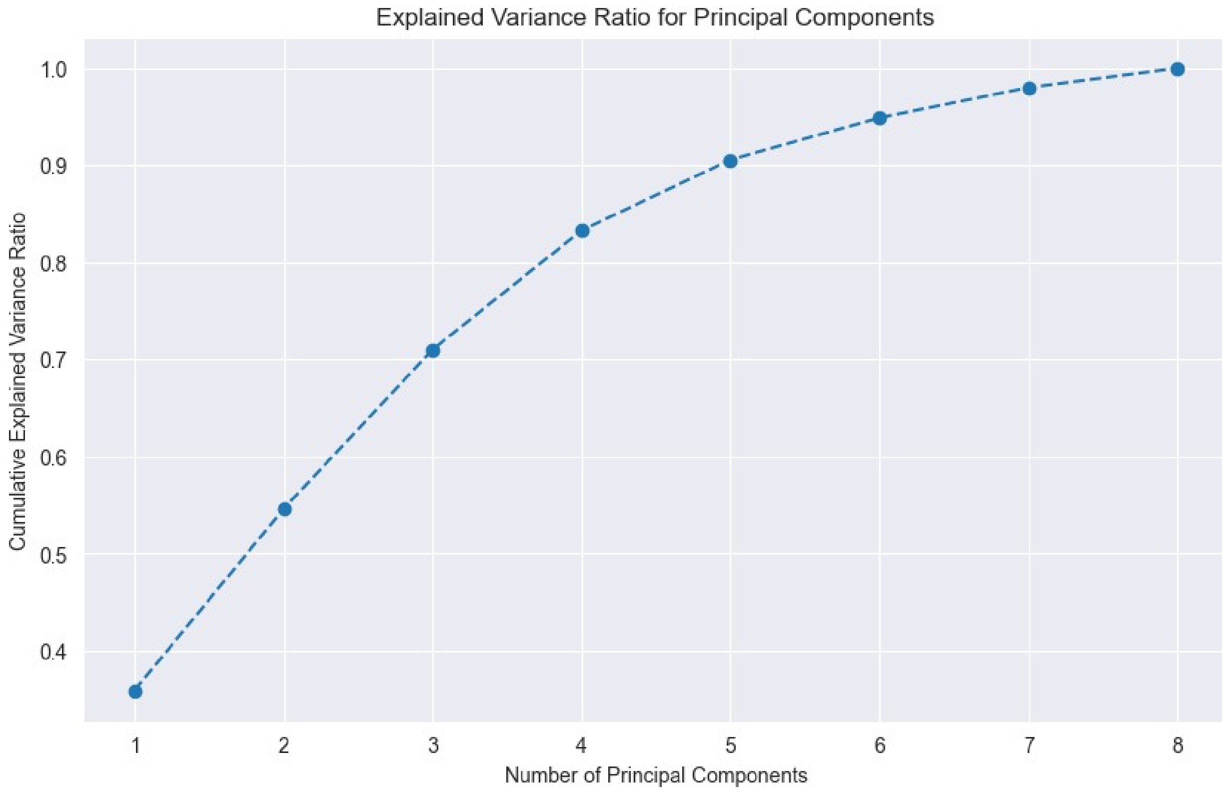



```
# Assuming 'df_normalized_zscore' is your DataFrame with outliers
removed and normalized
numeric_columns_for_pca = ['ALT', 'AST', 'weight(kg)', 'relaxation',
'waist(cm)', 'Cholesterol', 'HDL', 'systolic']

# Standardize the data (important for PCA)
scaler_for_pca = StandardScaler()
features_standardized_for_pca =
scaler_for_pca.fit_transform(df[numeric_columns_for_pca])

# Apply PCA for dimensionality reduction
pca_for_replacement = PCA()
principal_components_for_replacement =
pca_for_replacement.fit_transform(features_standardized_for_pca)

# Variance explained by each principal component
```

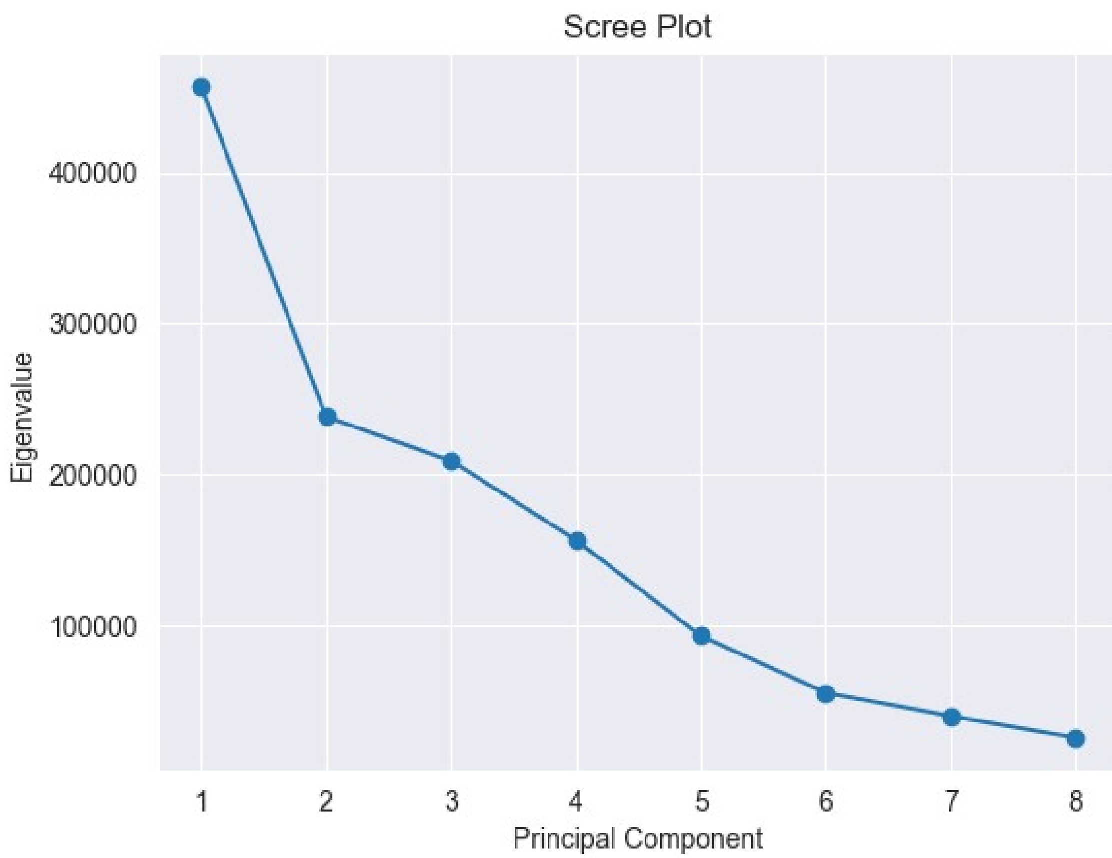
```
explained_variance_ratio =
pca_for_replacement.explained_variance_ratio_

# The variable 'principal_components_for_replacement' contains the
transformed data
print("Principal Components:")
print(pd.DataFrame(principal_components_for_replacement,
columns=[f'PC{i+1}' for i in range(len(numeric_columns_for_pca))]))
```

```
Principal Components:
             PC1       PC2       PC3       PC4       PC5       PC6
PC7  \
0       0.527779 -1.143920 -0.601828  1.329738  0.953990  0.236895 -
0.028690
1       1.038293 -1.424441  0.264738  0.460583 -0.154469 -0.388586
0.865020
2       0.347802  0.750286 -0.637003  0.461389  0.227296  0.087159 -
0.220740
3       3.162670 -0.513503 -2.026887 -0.413436 -0.649651 -0.040555 -
0.373648
4      -0.728714 -0.186675 -1.475829  1.142401  0.392687 -0.073297 -
0.000232
...          ...       ...       ...       ...       ...       ...
...
159251 -1.702009 -1.171266  2.062232 -0.342960  0.747525  0.351459
0.098358
159252 -0.114058 -0.519037  0.007815 -0.847765 -0.457909  0.124668 -
0.477046
159253 -3.436666 -0.658320  0.547927  0.067779 -0.940183  0.471583
0.016600
159254  0.933586 -0.786251 -1.098256  0.531261 -0.812568 -0.135296 -
1.094893
159255 -1.822833 -1.762676  1.232753  0.719901 -1.004129  0.304516 -
0.495064

             PC8
0       0.235281
1       0.333520
2      -0.530824
3       0.283176
4       0.236666
...          ...
159251  0.054403
159252 -0.423042
159253 -0.277518
159254  0.542136
159255  0.897648

[159256 rows x 8 columns]
```

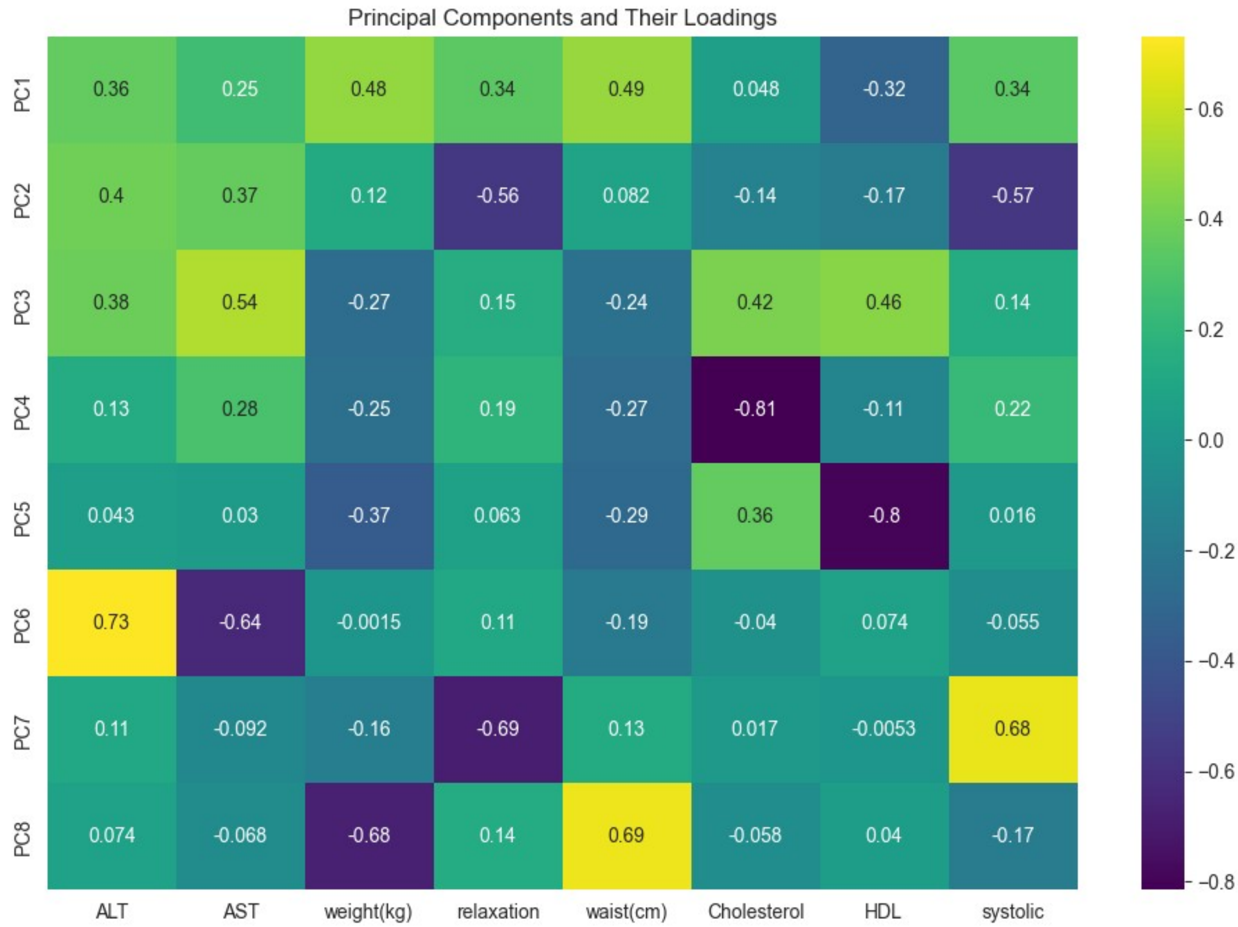
```python
# Plotting the explained variance ratio
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(explained_variance_ratio) + 1),
explained_variance_ratio.cumsum(), marker='o', linestyle='--')
plt.title('Explained Variance Ratio for Principal Components')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance Ratio')
plt.grid(True)
plt.show()
```




```python
# Scree Plot
eigenvalues = pca_for_replacement.singular_values_ ** 2
plt.plot(range(1, len(eigenvalues) + 1), eigenvalues, marker='o')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Eigenvalue')
plt.show()
```




```python
# Heatmap for loadings
plt.figure(figsize=(12, 8))
sns.heatmap(pca_for_replacement.components_, cmap='viridis',
annot=True, xticklabels=numeric_columns_for_pca,
yticklabels=[f'PC{i+1}' for i in range(len(numeric_columns_for_pca))])
plt.title('Principal Components and Their Loadings')
plt.show()
```



Principal Components and Their Loadings
PC1
PC2
PC3
PC4
PC5
PC6
PC7
PC8
ALT
AST
weight(kg)
relaxation
waist(cm)
Cholesterol
HDL
systolic
0.36 0.25 0.48 0.34 0.49 0.048 -0.32 0.34
0.4 0.37 0.12 -0.56 0.082 -0.14 -0.17 -0.57
0.38 0.54 -0.27 0.15 -0.24 0.42 0.46 0.14
0.13 0.28 -0.25 0.19 -0.27 -0.81 -0.11 0.22
0.043 0.03 -0.37 0.063 -0.29 0.36 -0.8 0.016
0.73 -0.64 -0.0015 0.11 -0.19 -0.04 0.074 -0.055
0.11 -0.092 -0.16 -0.69 0.13 0.017 -0.0053 0.68
0.074 -0.068 -0.68 0.14 0.69 -0.058 0.04 -0.17
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8