Assignment No. 6

Problem Statement: DATA PREPROCESSING: Perform the following operations using Python on the heart diseases data sets a. Data cleaning b. Error-correcting

Objective: To preprocess the heart disease dataset by performing data cleaning and error correction to enhance data quality and reliability. This includes handling missing values and correcting inconsistencies, preparing the dataset for accurate analysis and predictive modeling in diagnosing heart disease.

Prerequisite :

1. Basic understanding of Python programming.
2. Understanding of Data cleaning, Data Preprocessing and error detecting.
3. Understanding of libraries like Pandas, NumPy, Matplotlib, and Seaborn.
4. Knowledge of libraries such as NumPy and Matplotlib for data generation and visualization

Theory :

Data Cleaning:

Data cleaning is essential to improve data quality by handling missing values, removing noise, and resolving inconsistencies. Unprocessed data often leads to unreliable analysis, so data cleaning is crucial for accurate results.

a) Handling Missing Data

- **Removing Rows/Columns:** If a row or column has over 65% missing values or is entirely null, it can be removed.
- **Duplicate Removal:** Duplicate entries should be eliminated to avoid skewed results.
- **Estimating Missing Values:** For minor missing data, gaps can be filled with mean, median, or mode values.

b) Addressing Noisy Data

- **Binning:** Data is grouped into segments, and values are smoothed within each segment.
- **Clustering:** Similar data points are grouped to help identify outliers.
- **Regression:** Data smoothing is achieved by fitting values to a regression model.

Error-Correction:

Error correction in data science applies to the identification, the treatment and the apologizing of facts which are not true or do not present some aspects of the truth. Data errors can occur due to many reasons, such as mistakes in manual input, faults with sensors, software bugs and problems related to the fusion of data. In their uncorrected state, errors are potential threats to the different levels of analysis, machine learning models, and even decisions based on data.

Types of Errors in Data

1. **Measurement errors:** Deficiencies in data collection that occurs in the usage of bad instruments or the recording of measurements that are wrong.
2. **Data entry errors:** Mistakes caused by people working on data compilation, for instance, making a typographical error or lack of some pieces of information.
3. **Outliers:** High values or low values that occurs on account of a data entry mistake or very rare occurrences.
4. **Duplicate data:** Identical observations or data sets that affect the result of analysis in an undesired way.
5. **Inconsistent Databases:** Entries in the Data base that have the same meaning, but different format or name.
6. **Missing Values:** These are values that should have been recorded in a data set and which, if missing, can compromise analysis and other processes in a serious way.

Error Correction Techniques

1. **Data Validation**

   This is the procedure of maintaining constraints on data so that it will always meet the desired conditions, thus avoiding the risk of entering or recording erroneous information.

   Techniques:

   a) **Range Checks:** Guarantees that the value to be provided resides within level constraints, for example age should be a natural number.
   b) **Consistency Checks:** Checks whether or not related items of Data are supposed to relate, for example a start date is not supposed to be later than an end date.
   c) **Format Checks:** cover letters, email addresses, phone numbers and dates etc., are supposed to be presented in acceptable formats.

2. **Data Imputation for Missing Values**:

   The problem of missing values is solved by substituting the absent data with appropriate values so as not to compromise the integrity of the dataset.

   Techniques:

a) **Mean/Median ordering for missing data imputation**: Involves substituting unknown values with average or median values found within the column, helpful for numerical data.
b) **Mode Ordering:** For categorical data, in which unknown values should be substituted with the most frequent category.
c) Predictive Imputation: Relying on models in statistical learning to estimate and substitute the missing cells due to the presence of other variables. K-Nearest
d) Neighbors (KNN) Imputation: Fills missing values with the 'k' nearest neighbors of a data point.

3. Outlier Detection and Correction

Outliers are observations that if not dealt with appropriately, may distort analyses and models. Discrepancy detection and correction involves detecting these anomalies and making the choice on the action that should be taken.

Methods:

a) Z-score Method: It determines how many standard deviations a data point differs from the mean value. Outliers are defined as data points that cross a certain threshold level (usually ±3 or higher).
b) Interquartile Range (IQR): Values outside of the range $[Q1-1.5\times IQR, Q3+1.5\times IQR]$ are considered outliers.
c) Trimming/Capping: It entails eliminating extreme cases or fixing a cut off maximum values to the extreme cases.
d) Winsorizing: It is a statistical practice in which extreme values are substituted with the nearest level of an extreme value within a certain range.

4. Duplicate Data Detection and removal

The inclusion of more datasets into the active datasets causes the arousal of Duplicate datasets that often miscount the counts, bias the statistics, and overtrain the model.

Methods:

a) Exact Duplicate Removal: select and delete information that is identical to other information from the signature.
b) Fuzzy Matching: This entails measuring scenes using a metric, often Levenshtein distance, this is close duplicates of copies based on the similarity of the strings, used sometimes in text data.

Benefits of Error Correction

Effective error correction improves the accuracy, reliability, and consistency of data, enhancing its usability for analytics, modeling, and decision-making. It also reduces biases,

prevents erroneous conclusions, and improves the interpretability and performance of predictive models.

Challenges in Error Correction

a) **Cost and Time**: Error detection and correction can be resource-intensive, especially for large datasets.
b) **Scalability**: As data grows, error correction methods must scale, often necessitating automation.
c) **Subjectivity**: Some corrections depend on assumptions or domain knowledge, potentially introducing bias.
d) **Balancing Correction with Data Integrity**: Excessive error correction, especially through removal, can lead to data loss and reduce representativeness.

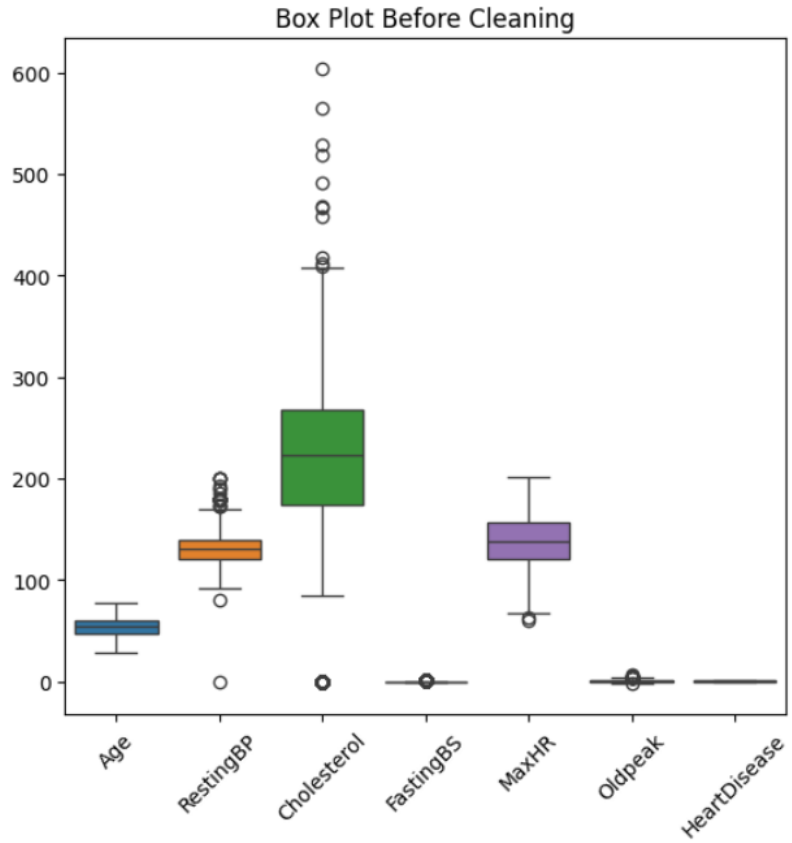
Code & Output:

```
import pandas as pd
# Load the dataset
df = pd.read_csv('C:/Users/dnyan/FODS Assignments/Datasets/heart.csv')
data = df.iloc[:, 4]
print(data.head())
```

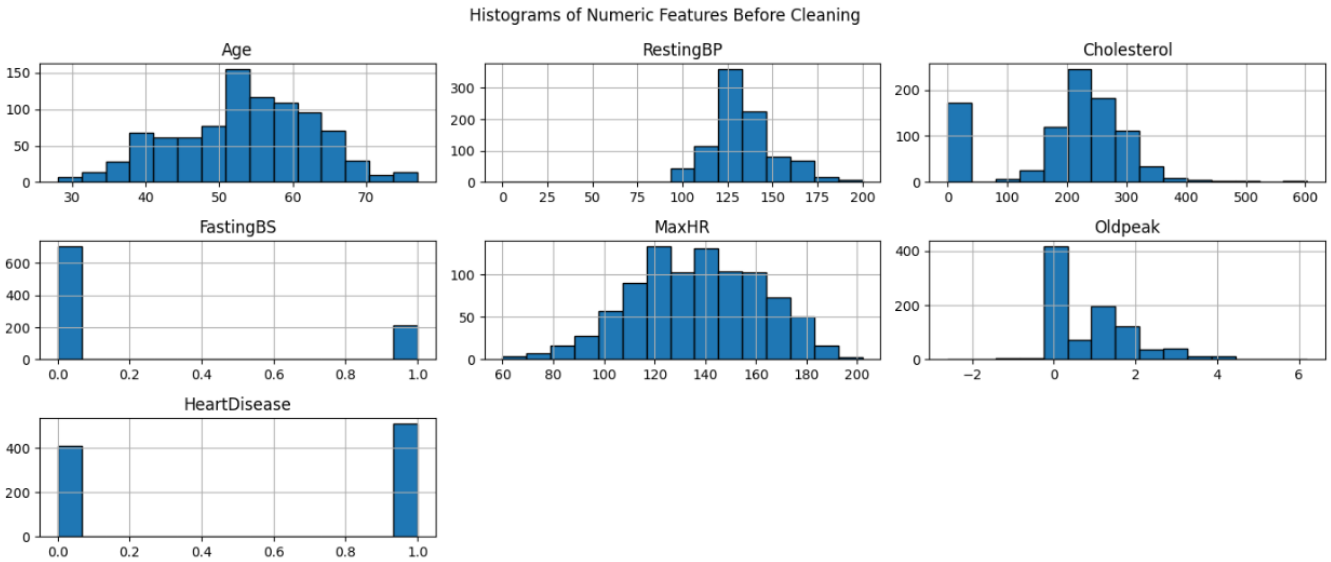
```
0    289
1    180
2    283
3    214
4    195
Name: Cholesterol, dtype: int64
```

```
# Before Cleaning: Visualize Outliers
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.boxplot(data=df[numeric_columns])
plt.title('Box Plot Before Cleaning')
plt.xticks(rotation=45)
```

```
([0, 1, 2, 3, 4, 5, 6],
 [Text(0, 0, 'Age'),
  Text(1, 0, 'RestingBP'),
  Text(2, 0, 'Cholesterol'),
  Text(3, 0, 'FastingBS'),
  Text(4, 0, 'MaxHR'),
  Text(5, 0, 'Oldpeak'),
  Text(6, 0, 'HeartDisease')])
```




```
# Plot Histograms
plt.subplot(1, 2, 2)
df[numeric_columns].hist(bins=15, edgecolor='black', figsize=(14, 6))
plt.suptitle('Histograms of Numeric Features Before Cleaning')

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



```
#Step 1: Data Cleaning
# Check for missing values
print("Missing values in each column:")
print(df.isnull().sum())
```

```
Missing values in each column:
Age             0
Sex             0
ChestPainType   0
RestingBP       0
Cholesterol     0
FastingBS       0
RestingECG      0
MaxHR           0
ExerciseAngina  0
Oldpeak         0
ST_Slope        0
HeartDisease    0
dtype: int64
```

```
#NO missing values in dataset
```

```
dataset_null = df.isnull()
print(dataset_null)
```

```
       Age    Sex  ChestPainType  RestingBP  Cholesterol  FastingBS  \
0    False  False          False      False        False      False
1    False  False          False      False        False      False
2    False  False          False      False        False      False
3    False  False          False      False        False      False
4    False  False          False      False        False      False
..     ...    ...            ...        ...          ...        ...
913  False  False          False      False        False      False
914  False  False          False      False        False      False
915  False  False          False      False        False      False
916  False  False          False      False        False      False
917  False  False          False      False        False      False

     RestingECG  MaxHR  ExerciseAngina  Oldpeak  ST_Slope  HeartDisease
0         False  False           False    False     False         False
1         False  False           False    False     False         False
2         False  False           False    False     False         False
3         False  False           False    False     False         False
4         False  False           False    False     False         False
..          ...    ...             ...      ...       ...           ...
913       False  False           False    False     False         False
914       False  False           False    False     False         False
915       False  False           False    False     False         False
916       False  False           False    False     False         False
917       False  False           False    False     False         False

[918 rows x 12 columns]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   918 non-null    object 
 3   RestingBP       918 non-null    int64  
 4   Cholesterol     918 non-null    int64  
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    object 
 7   MaxHR           918 non-null    int64  
 8   ExerciseAngina  918 non-null    object 
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object 
 11  HeartDisease    918 non-null    int64  
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

```
# 1. Check for inconsistent values in categorical columns and display unique values
for col in ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope']:
    print(f"Unique values in {col} column: {df[col].unique()}")
```

```
Unique values in Sex column: ['M' 'F']
Unique values in ChestPainType column: ['ATA' 'NAP' 'ASY' 'TA']
Unique values in RestingECG column: ['Normal' 'ST' 'LVH']
Unique values in ExerciseAngina column: ['N' 'Y']
Unique values in ST_Slope column: ['Up' 'Flat' 'Down']
```

```
# 2. Encode categorical columns
df['Sex'] = df['Sex'].map({'M': 1, 'F': 0})
df['ChestPainType'] = df['ChestPainType'].map({'TA': 0, 'ATA': 1, 'NAP': 2, 'ASY': 3})
df['RestingECG'] = df['RestingECG'].map({'Normal': 0, 'ST': 1, 'LVH': 2})
df['ExerciseAngina'] = df['ExerciseAngina'].map({'N': 0, 'Y': 1})
df['ST_Slope'] = df['ST_Slope'].map({'Up': 0, 'Flat': 1, 'Down': 2})
```

```
import numpy as np

# 3. Handle outliers in numeric columns using Interquartile Range (IQR) method
def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df[column] = np.where(df[column] < lower_bound, lower_bound, df[column])
    df[column] = np.where(df[column] > upper_bound, upper_bound, df[column])

for col in ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']:
    remove_outliers(df, col)
```

```python
# 4. Verify the data types after cleaning and encoding
print(df.info())

# Display the cleaned data
print(df.head())
```

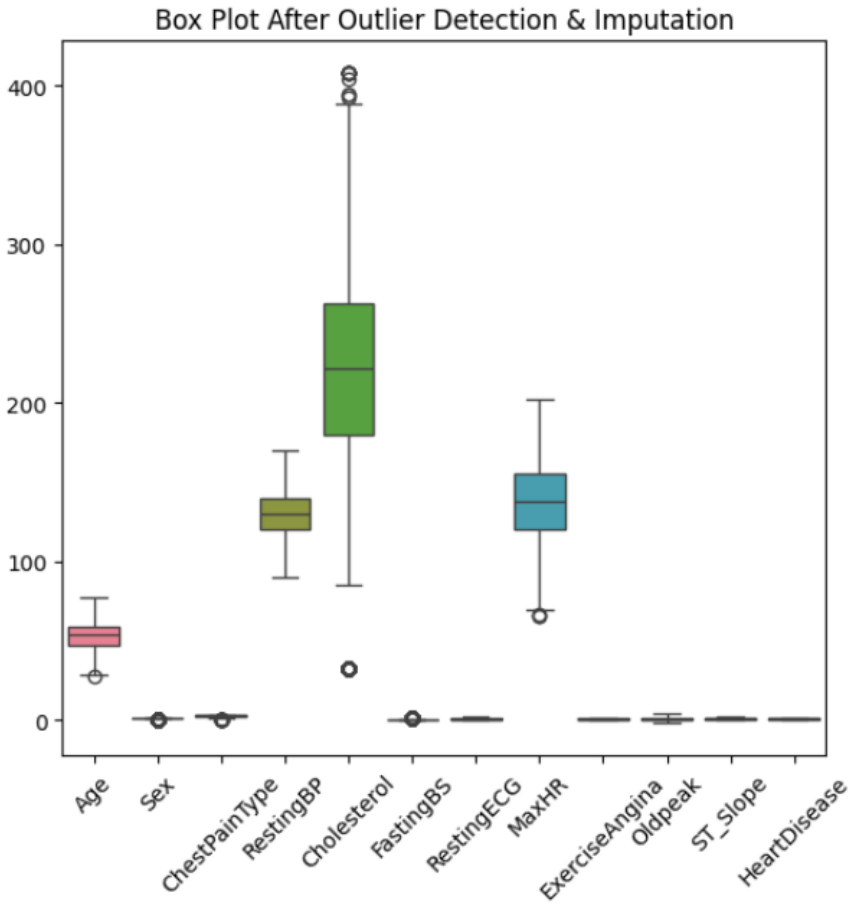
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    float64
 1   Sex             918 non-null    int64  
 2   ChestPainType   918 non-null    int64  
 3   RestingBP       918 non-null    float64
 4   Cholesterol     918 non-null    float64
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    int64  
 7   MaxHR           918 non-null    float64
 8   ExerciseAngina  918 non-null    int64  
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    int64  
 11  HeartDisease    918 non-null    int64  
dtypes: float64(5), int64(7)
memory usage: 86.2 KB
None
    Age  Sex  ChestPainType  RestingBP  Cholesterol  FastingBS  RestingECG  \
0  40.0    1              1      140.0        289.0          0           0   
1  49.0    0              2      160.0        180.0          0           0   
2  37.0    1              1      130.0        283.0          0           1   
3  48.0    0              3      138.0        214.0          0           0   
4  54.0    1              2      150.0        195.0          0           0   

   MaxHR  ExerciseAngina  Oldpeak  ST_Slope  HeartDisease  
0  172.0               0      0.0         0             0  
1  156.0               0      1.0         1             1  
2   98.0               0      0.0         0             0  
3  108.0               1      1.5         1             1  
4  122.0               0      0.0         0             0  
```

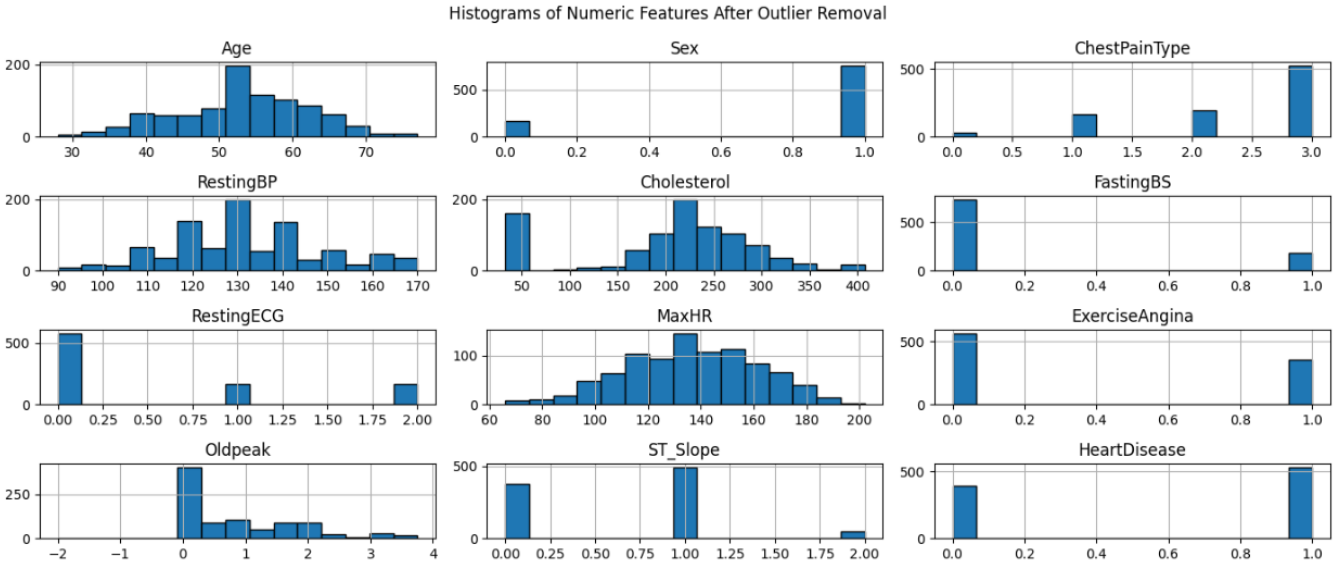
```python
# 1. Data Validation

# Set valid ranges for numerical columns (age, cholesterol, etc.)
valid_age_range = (0, 120)
valid_resting_bp_range = (0, 200)
valid_cholesterol_range = (0, 600)
valid_max_hr_range = (0, 220)
```

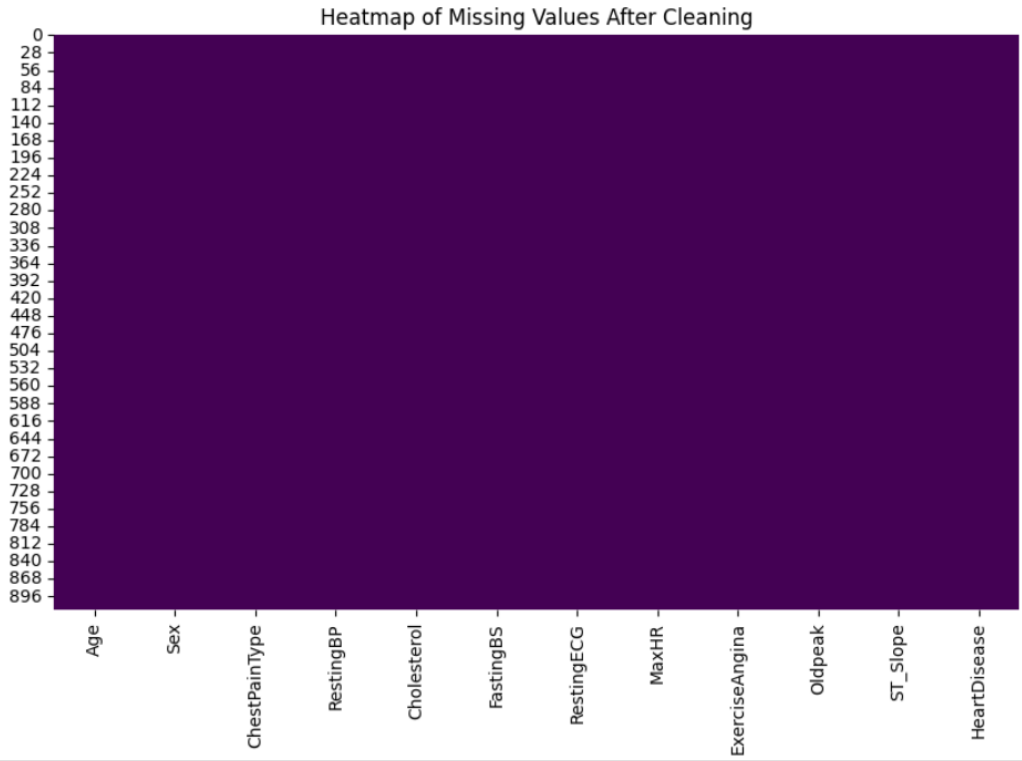
```python
# Apply range checks and replace out-of-range values with NaN
df.loc[~df['Age'].between(*valid_age_range), 'Age'] = np.nan
df.loc[~df['RestingBP'].between(*valid_resting_bp_range), 'RestingBP'] = np.nan
df.loc[~df['Cholesterol'].between(*valid_cholesterol_range), 'Cholesterol'] = np.nan
df.loc[~df['MaxHR'].between(*valid_max_hr_range), 'MaxHR'] = np.nan
```

```
from scipy import stats
# 2. Outlier Detection and Correction

# Using Z-score method for outlier detection in numerical columns
numeric_columns = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
z_scores = np.abs(stats.zscore(df[numeric_columns]))
outliers = (z_scores > 3)  # Threshold set to 3 standard deviations
df[numeric_columns] = df[numeric_columns].mask(outliers, np.nan)
```

```
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.ensemble import IsolationForest
import matplotlib.pyplot as plt
import seaborn as sns

# Define numeric columns
numeric_columns = df.select_dtypes(include=[np.number]).columns.tolist()

# 1. Handle Missing Values - Impute using Median
imputer = SimpleImputer(strategy='median')
df[numeric_columns] = imputer.fit_transform(df[numeric_columns])

# 2. Detect outliers using Isolation Forest
iso_forest = IsolationForest(contamination=0.05)
outliers = iso_forest.fit_predict(df[numeric_columns])

# 3. Mark outliers as NaN for imputation
df.loc[outliers == -1, numeric_columns] = np.nan
# 4. Impute outliers again using median for numeric columns
df[numeric_columns] = imputer.fit_transform(df[numeric_columns])

# Visualization
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.boxplot(data=df[numeric_columns])
plt.title('Box Plot After Outlier Detection & Imputation')
plt.xticks(rotation=45)

# Plot Histograms After Cleaning
plt.subplot(1, 2, 2)
df[numeric_columns].hist(bins=15, edgecolor='black', figsize=(14, 6))
plt.suptitle('Histograms of Numeric Features After Outlier Removal')

plt.tight_layout()
plt.show()

# Heatmap for Missing Values
plt.figure(figsize=(10, 6))
sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
plt.title('Heatmap of Missing Values After Cleaning')
plt.show()
```


Box Plot After Outlier Detection & Imputation
400
300
200
100
0
Age
Sex
ChestPainType
RestingBP
Cholesterol
FastingBS
RestingECG
MaxHR
ExerciseAngina
Oldpeak
ST_Slope
HeartDisease


Histograms of Numeric Features After Outlier Removal
Age
Sex
ChestPainType
RestingBP
Cholesterol
FastingBS
RestingECG
MaxHR
ExerciseAngina
Oldpeak
ST_Slope
HeartDisease

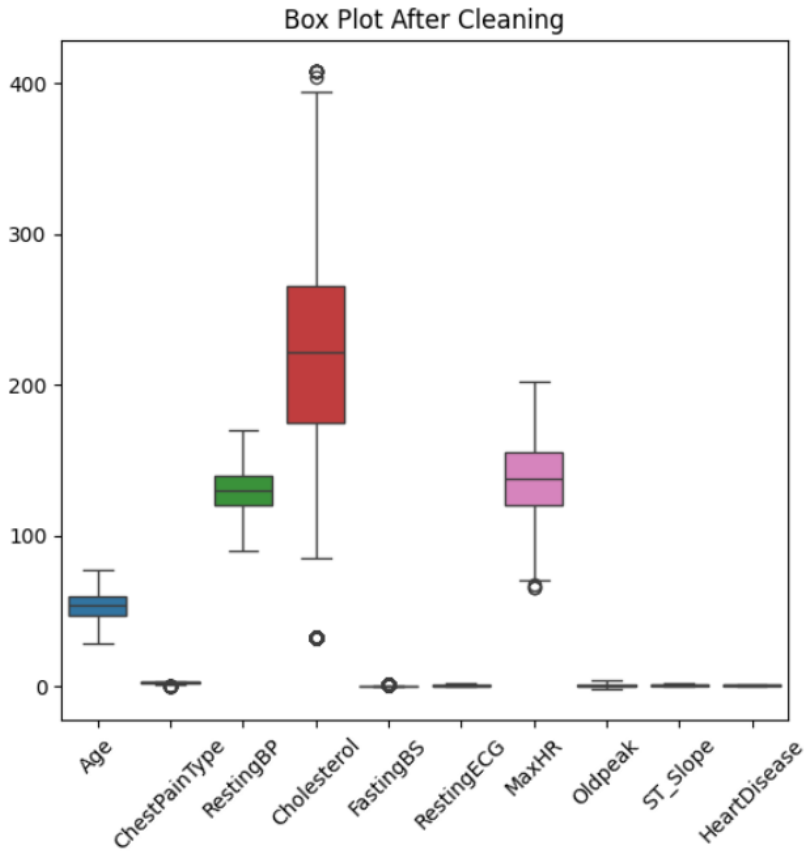



```
#Standardization for Consistency

# Convert Sex and ExerciseAngina to categorical
df['Sex'] = df['Sex'].replace({0: 'Female', 1: 'Male'})
df['ExerciseAngina'] = df['ExerciseAngina'].replace({0: 'No', 1: 'Yes'})
```

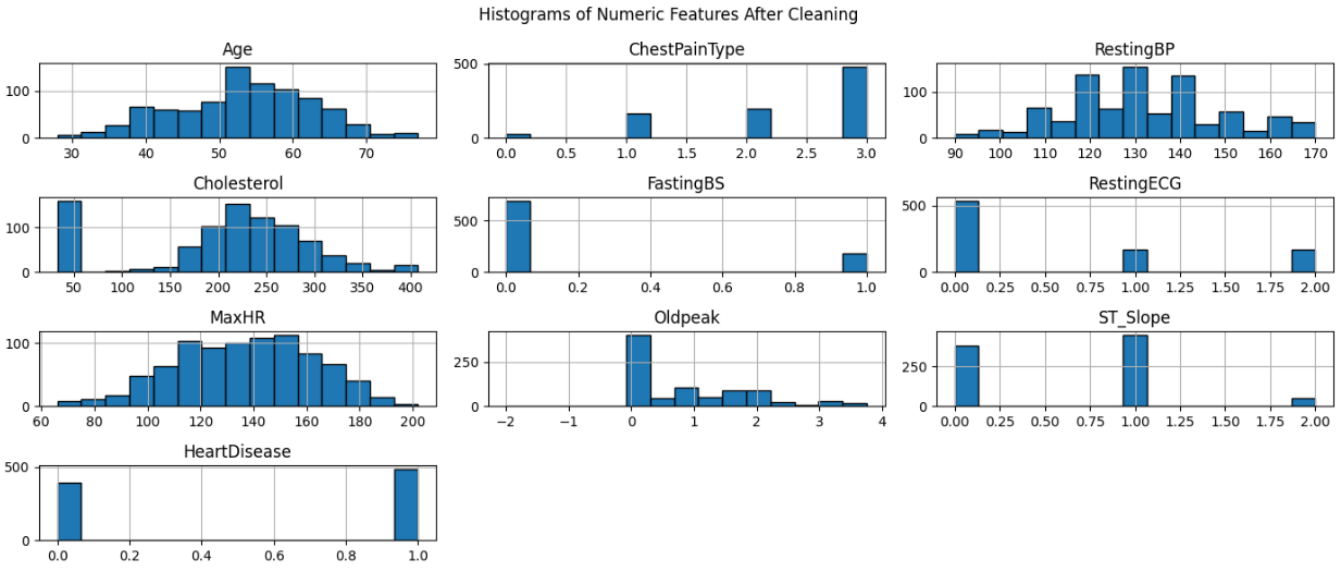
```
# Handling Duplicates

# Check and remove exact duplicates
df.drop_duplicates(inplace=True)
```

```
# After Cleaning: Visualize Outliers Again
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.boxplot(data=df[numeric_columns])
plt.title('Box Plot After Cleaning')
plt.xticks(rotation=45)
```

```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
 [Text(0, 0, 'Age'),
  Text(1, 0, 'ChestPainType'),
  Text(2, 0, 'RestingBP'),
  Text(3, 0, 'Cholesterol'),
  Text(4, 0, 'FastingBS'),
  Text(5, 0, 'RestingECG'),
  Text(6, 0, 'MaxHR'),
  Text(7, 0, 'Oldpeak'),
  Text(8, 0, 'ST_Slope'),
  Text(9, 0, 'HeartDisease')])
```




```
# Plot Histograms Again
plt.subplot(1, 2, 2)
df[numeric_columns].hist(bins=15, edgecolor='black', figsize=(14, 6))
plt.suptitle('Histograms of Numeric Features After Cleaning')

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



References :

https://medium.easyread.co/basics-of-data-preprocessing-71c314bc7188
https://medium.com/womenintechnology/data-preprocessing-steps-for-machine-learning-in-phyton-part-1-18009c6f1153
Github- https://github.com/dnyaneshwardhere/FODS

Conclusion :

In this assignment, we performed data cleaning and error correction on a heart disease dataset containing 918 entries. We addressed missing values through median imputation and detected outliers using the Isolation Forest algorithm, replacing them with NaN for later imputation. Visualization techniques confirmed the effectiveness of our cleaning process, enhancing the dataset's quality for reliable analysis and predictive modeling related to heart disease risk.