Introduction to Data Analytics SGA -1 (Data Loading, Cleaning, and Exploration)

1. Data Acquisition

1.1 Downloading the Dataset

First the imports are declared and as I am using macOS, it is throwing SSL error when trying to download the dataset from UCI Repository so will be using this code:

```
from ucimlrepo import fetch_ucirepo
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import ssl
import certifi

# For macOS SSL Error
ssl._create_default_https_context = lambda:
ssl.create_default_context(cafile=certifi.where())
```

I will be using the Heart Disease Dataset from UCI Repository as it is a very common data set that contains only Numerical and Categorical Data.

Now, the code to download the dataset:

```
# 1.1 Download the Dataset from UCI Repository
heart_disease = fetch_ucirepo(id=45)  # Heart Disease dataset
```

1.2 Converting the data into a pandas DataFrame

The code to convert the data into DataFrame is as follows and also additionally storing the data as a CSV file “heart_disease.csv”:

```
# 1.2 Convert dataset to DataFrame
X = heart_disease.data.features
y = heart_disease.data.targets
df = pd.concat([X, y], axis=1)

# Additionally saved the data in the form of a CSV File
df.to_csv("heart_disease.csv", index=False)
print("\nDataset saved as 'heart_disease_cleaned.csv'")
```

Output of the Code:

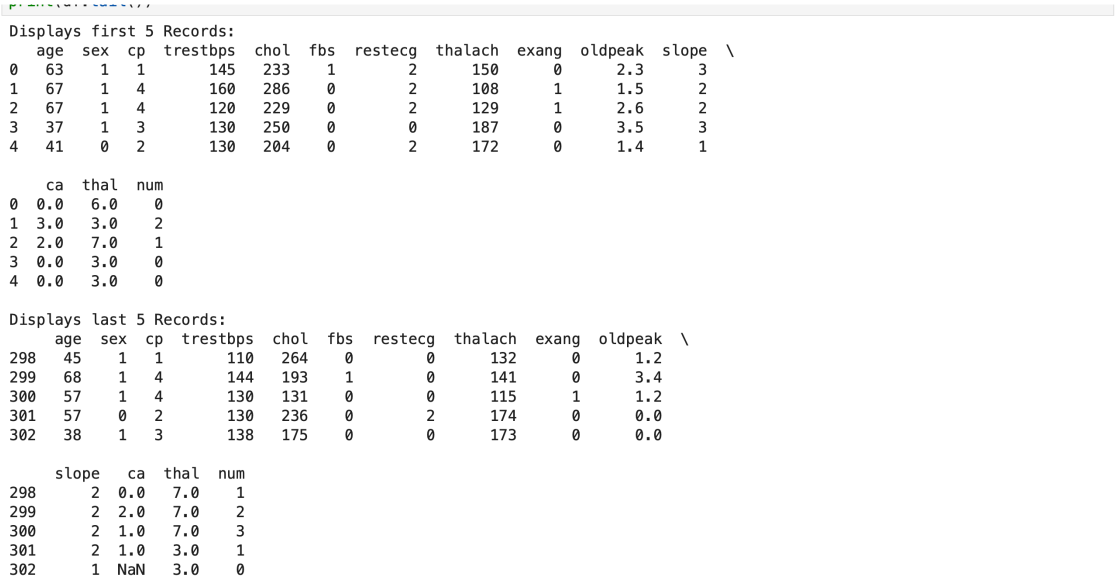
```
print("\nDataset saved as 'heart_disease_cleaned.csv'")

Dataset saved as 'heart_disease_cleaned.csv'
```

1.3 Displaying the First & Last Data of the Dataset

Code to display the First & Last Data of the Dataset:

```
# 1.3 Display first and last five records
print("Displays first 5 Records:")
print(df.head())

print("\nDisplays last 5 Records:")
print(df.tail())
```

Output of the Code:



```
Displays first 5 Records:
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   63    1   1       145   233    1        2      150      0      2.3      3
1   67    1   4       160   286    0        2      108      1      1.5      2
2   67    1   4       120   229    0        2      129      1      2.6      2
3   37    1   3       130   250    0        0      187      0      3.5      3
4   41    0   2       130   204    0        2      172      0      1.4      1

    ca  thal  num
0  0.0   6.0    0
1  3.0   3.0    2
2  2.0   7.0    1
3  0.0   3.0    0
4  0.0   3.0    0

Displays last 5 Records:
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
298   45    1   1       110   264    0        0      132      0      1.2
299   68    1   4       144   193    1        0      141      0      3.4
300   57    1   4       130   131    0        0      115      1      1.2
301   57    0   2       130   236    0        2      174      0      0.0
302   38    1   3       138   175    0        0      173      0      0.0

     slope   ca  thal  num
298      2  0.0   7.0    1
299      2  2.0   7.0    2
300      2  1.0   7.0    3
301      2  1.0   3.0    1
302      1  NaN   3.0    0
```

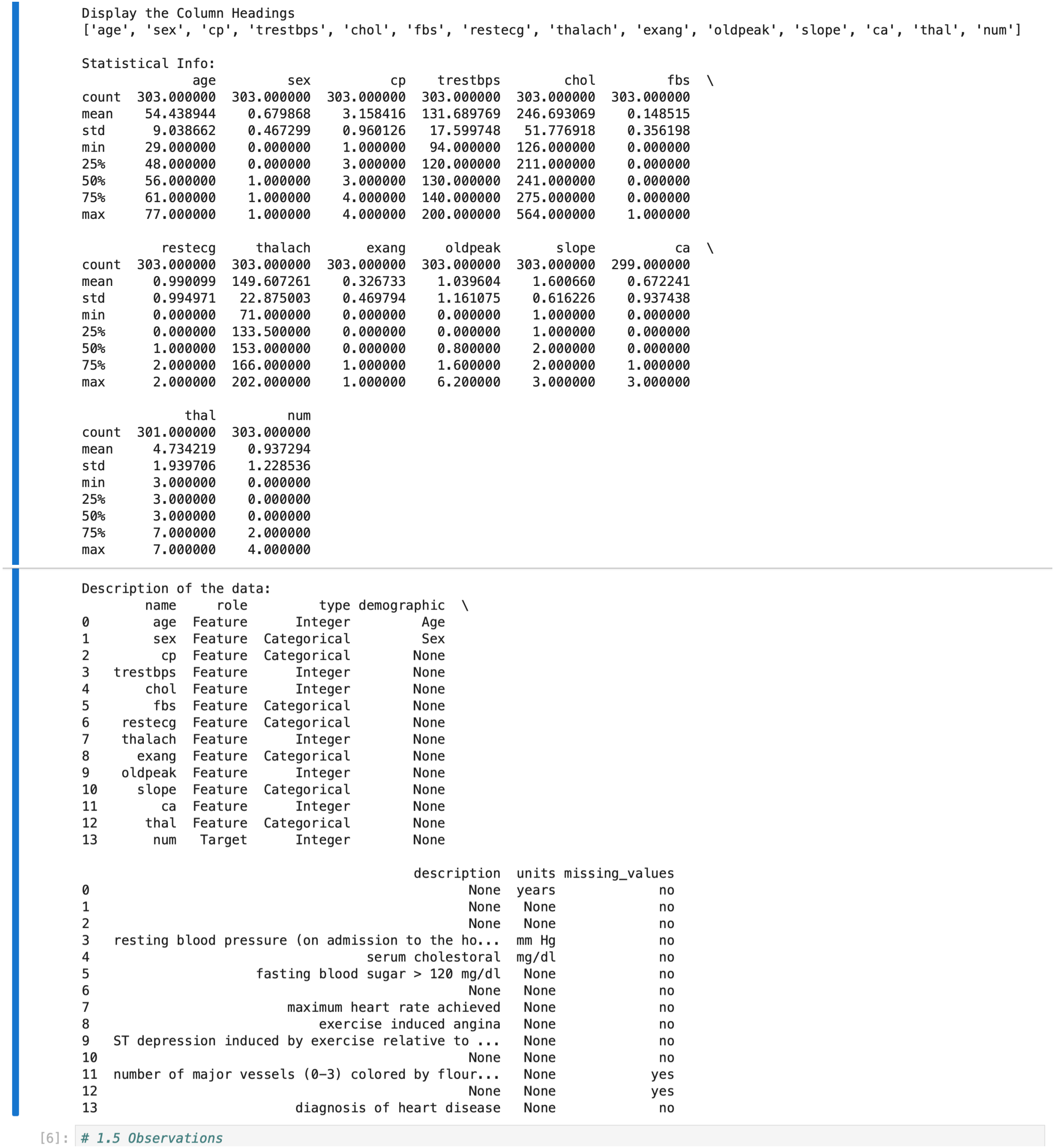
1.4 Using functions to display the given data

Code to display column headings, statistical information, and description of the data:

```
# 1.4 Display column headings, statistical info, and description

# Shows the column headings
print("\nDisplay the Column Headings")
print(df.columns.tolist())

# Shows the Statistical Information (like count, mean, standard
deviation, etc.)
print("\nStatistical Info:")
print(df.describe())

# Shows the Description of the Data (as in structure, meaning,
distributing, types)
```

```
print("\nDescription of the data:")
print(heart_disease.variables)
```

Output of the Code:



```
Display the Column Headings
['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'num']

Statistical Info:
              age         sex          cp    trestbps        chol         fbs  \
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean    54.438944    0.679868    3.158416  131.689769  246.693069    0.148515
std      9.038662    0.467299    0.960126   17.599748   51.776918    0.356198
min     29.000000    0.000000    1.000000   94.000000  126.000000    0.000000
25%     48.000000    0.000000    3.000000  120.000000  211.000000    0.000000
50%     56.000000    1.000000    3.000000  130.000000  241.000000    0.000000
75%     61.000000    1.000000    4.000000  140.000000  275.000000    0.000000
max     77.000000    1.000000    4.000000  200.000000  564.000000    1.000000

          restecg     thalach       exang     oldpeak       slope          ca  \
count  303.000000  303.000000  303.000000  303.000000  303.000000  299.000000
mean     0.990099  149.607261    0.326733    1.039604    1.600660    0.672241
std      0.994971   22.875003    0.469794    1.161075    0.616226    0.937438
min      0.000000   71.000000    0.000000    0.000000    1.000000    0.000000
25%      0.000000  133.500000    0.000000    0.000000    1.000000    0.000000
50%      1.000000  153.000000    0.000000    0.800000    2.000000    0.000000
75%      2.000000  166.000000    1.000000    1.600000    2.000000    1.000000
max      2.000000  202.000000    1.000000    6.200000    3.000000    3.000000

             thal         num
count  301.000000  303.000000
mean     4.734219    0.937294
std      1.939706    1.228536
min      3.000000    0.000000
25%      3.000000    0.000000
50%      3.000000    0.000000
75%      7.000000    2.000000
max      7.000000    4.000000
```

```
Description of the data:
        name     role         type demographic  \
0        age  Feature      Integer         Age
1        sex  Feature  Categorical         Sex
2         cp  Feature  Categorical        None
3   trestbps  Feature      Integer        None
4       chol  Feature      Integer        None
5        fbs  Feature  Categorical        None
6    restecg  Feature  Categorical        None
7    thalach  Feature      Integer        None
8      exang  Feature  Categorical        None
9    oldpeak  Feature      Integer        None
10     slope  Feature  Categorical        None
11        ca  Feature      Integer        None
12      thal  Feature  Categorical        None
13       num   Target      Integer        None

                                          description  units missing_values
0                                                None  years             no
1                                                None   None             no
2                                                None   None             no
3   resting blood pressure (on admission to the ho...  mm Hg             no
4                                   serum cholestoral  mg/dl             no
5                           fasting blood sugar > 120 mg/dl   None             no
6                                                None   None             no
7                         maximum heart rate achieved   None             no
8                             exercise induced angina   None             no
9   ST depression induced by exercise relative to ...   None             no
10                                               None   None             no
11  number of major vessels (0-3) colored by flour...   None            yes
12                                               None   None            yes
13                            diagnosis of heart disease   None             no
```

```
[6]: # 1.5 Observations
```

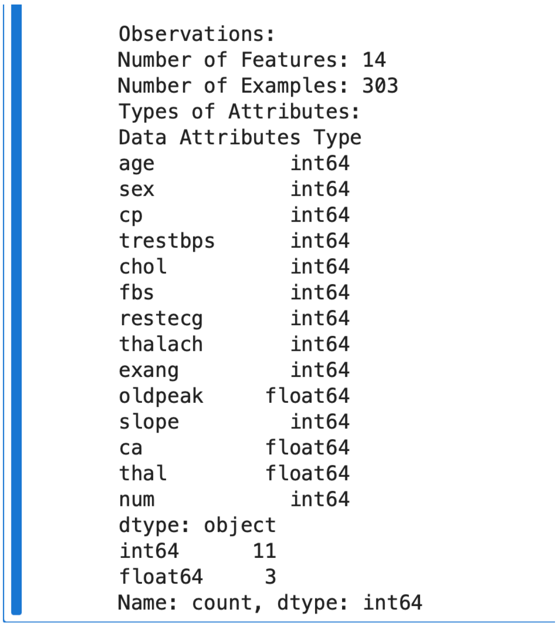
1.5 *Observations from the data*

Code to show the number of features and the examples in the dataset and the types of data attributes:

```
# 1.5 Observations

# Shows No. of Features and Examples in the Dataset
print("\nObservations:")
print(f"Number of Features: {df.shape[1]}")
print(f"Number of Examples: {df.shape[0]}")

# Shows the Types of Data Attributes
print("Types of Attributes:")
print("Data Attributes\tType")
print(df.dtypes)
print(df.dtypes.value_counts())
```

- So, from the dataset it can be concluded that the dataset consist of 303 patients (example) and 14 features related to medicine and heart disease diagnosis.
- The attributes “ca” and “thal” have missing values.
- The dataset includes medically significant features such as blood pressure (trestbps), cholesterol level (chol), chest pain type (cp), and maximum heart rate (thalach), which are crucial for diagnosis.

Output of the Code:



```
Observations:
Number of Features: 14
Number of Examples: 303
Types of Attributes:
Data Attributes Type
age            int64
sex            int64
cp             int64
trestbps       int64
chol           int64
fbs            int64
restecg        int64
thalach        int64
exang          int64
oldpeak      float64
slope          int64
ca           float64
thal         float64
num            int64
dtype: object
int64      11
float64     3
Name: count, dtype: int64
```

2. Data Preparation

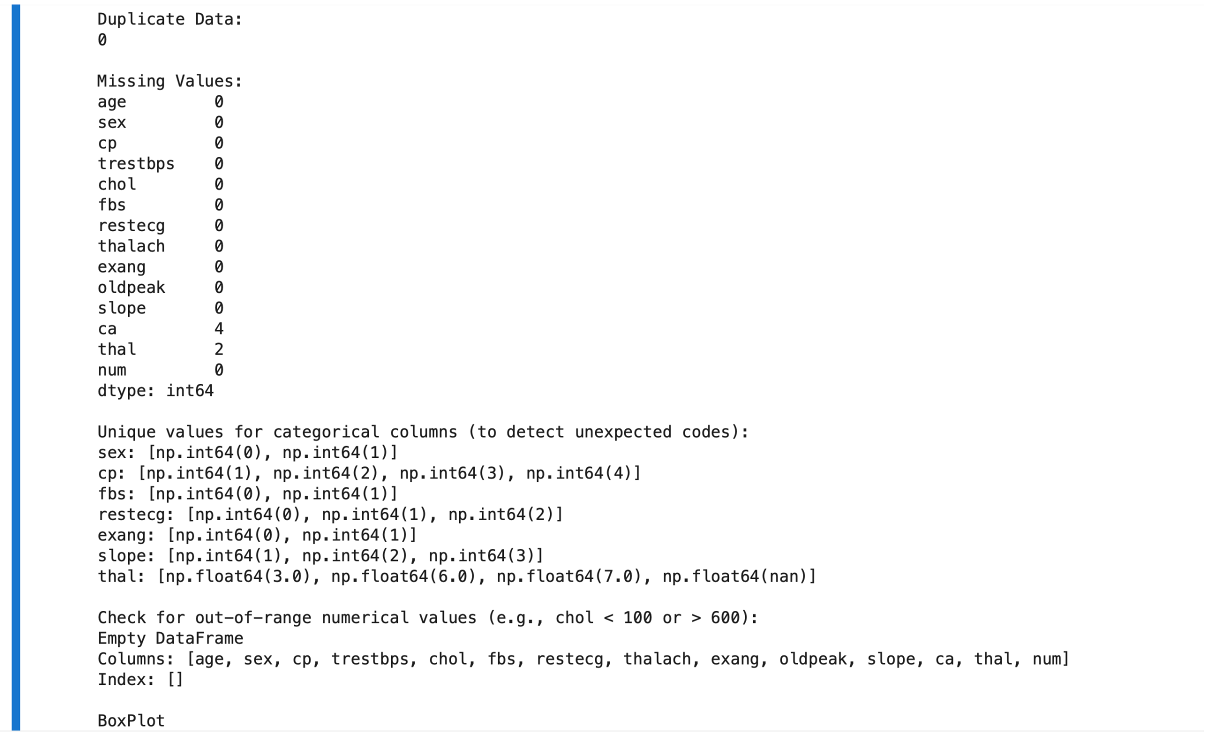
2.1 Checking for Duplicate, missing, inconsistent and Outliers

The python code for checking are:

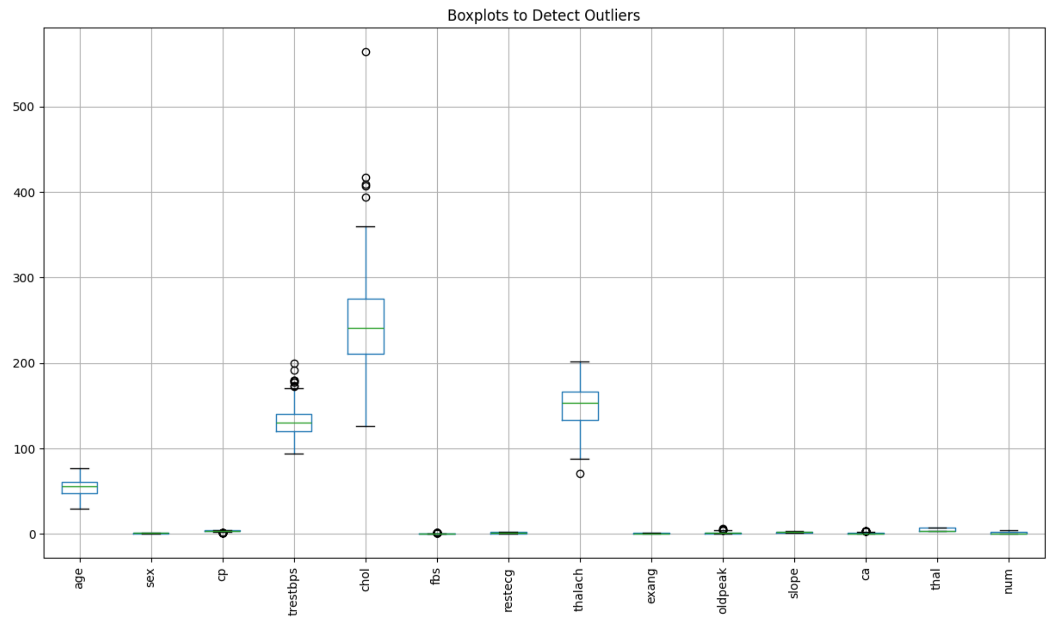
```
# 2.1 Check for duplicates, missing values, inconsistencies

# Check for Duplicate Data
print("\nDuplicate Data:")
print(df.duplicated().sum())

# Check for Missing Values
print("\nMissing Values:")
print(df.isnull().sum())

# Check for inconsistent data (i.e. unexpected values in categorical and numeric columns)
print("\nUnique values for categorical columns (to detect unexpected codes):")
for col in categorical_cols:
    print(f"{col}: {sorted(df[col].unique())}")

print("\nCheck for out-of-range numerical values (e.g., chol < 100 or > 600):")
print(df[(df['chol'] < 100) | (df['chol'] > 600)])


# Check for outliers using boxplot
print("\nBoxPlot")
plt.figure(figsize=(15, 8))
df.select_dtypes(include='number').boxplot(rot=90)
plt.title("Boxplots to Detect Outliers")
plt.show()
```

From the inconsistent data checking and output it is clear that the data is not inconsistent and there is no duplicate data and there is missing data. The outliers can be seen using the Boxplots.

<u>Output of the Code:</u>



```
Duplicate Data:
0

Missing Values:
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          4
thal        2
num         0
dtype: int64

Unique values for categorical columns (to detect unexpected codes):
sex: [np.int64(0), np.int64(1)]
cp: [np.int64(1), np.int64(2), np.int64(3), np.int64(4)]
fbs: [np.int64(0), np.int64(1)]
restecg: [np.int64(0), np.int64(1), np.int64(2)]
exang: [np.int64(0), np.int64(1)]
slope: [np.int64(1), np.int64(2), np.int64(3)]
thal: [np.float64(3.0), np.float64(6.0), np.float64(7.0), np.float64(nan)]

Check for out-of-range numerical values (e.g., chol < 100 or > 600):
Empty DataFrame
Columns: [age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal, num]
Index: []

BoxPlot
```

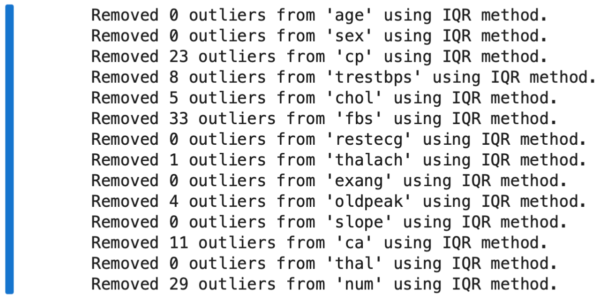


2.2 *Applying techniques to remove duplicate, missing, and outliers data*

The code for removing duplicate, missing and outlier data is:

```
# 2.2 Applying Techniques to remove duplicate, missing, inconsistent
and outlier data

# Remove duplicate data (although from the above data it is clear that
there is no duplicate data)
df = df.drop_duplicates()

# Drop missing data (will remove the thal one)
df = df.dropna()

# Outlier removal using IQR for all numeric columns (excluding
categorical ones)

# To check for outliers — exclude encoded categorical columns if needed
i.e. only numerical columns
numerical_cols = df.select_dtypes(include=[np.number]).columns.tolist()

# Remove outliers for each column using IQR (Inter-Quartile Range)
for col in numerical_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    before = df.shape[0]
    df = df[~((df[col] < Q1 - 1.5 * IQR) | (df[col] > Q3 + 1.5 * IQR))]
    after = df.shape[0]
    print(f"Removed {before - after} outliers from '{col}' using IQR
method.")
```

Output of the Code (only showing the outlier removal using IQR):



```
Removed 0 outliers from 'age' using IQR method.
Removed 0 outliers from 'sex' using IQR method.
Removed 23 outliers from 'cp' using IQR method.
Removed 8 outliers from 'trestbps' using IQR method.
Removed 5 outliers from 'chol' using IQR method.
Removed 33 outliers from 'fbs' using IQR method.
Removed 0 outliers from 'restecg' using IQR method.
Removed 1 outliers from 'thalach' using IQR method.
Removed 0 outliers from 'exang' using IQR method.
Removed 4 outliers from 'oldpeak' using IQR method.
Removed 0 outliers from 'slope' using IQR method.
Removed 11 outliers from 'ca' using IQR method.
Removed 0 outliers from 'thal' using IQR method.
Removed 29 outliers from 'num' using IQR method.
```

2.3 ***Encode Categorical using either One-hot Encoding or Label Encoding***

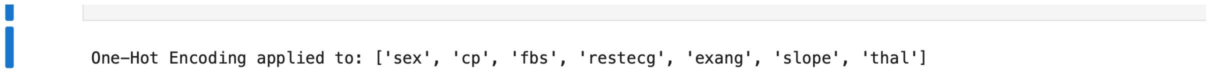
One Hot Encoding is a method for converting categorical variables into a binary format. It creates new columns for each category where 1 means the category is present and 0 means it is not. The primary purpose of One Hot Encoding is to ensure that categorical data can be effectively used in machine learning models.

Therefore, in this case of the chosen dataset, One Hot Encoding have to be used as the Encoding Technique.

The code for the same is :

```
# 2.3 Encode categorical variables

# Manually define categorical columns from the 1.4 chapter data
(Description of the Data)
categorical_cols = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope',
'thal']

# Apply one-hot encoding
df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)

print("\nOne-Hot Encoding applied to:", categorical_cols)
```

Output of the Code:



```
One-Hot Encoding applied to: ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'thal']
```

2.4 ***Report of the Observation***

The final observations that can be made about the dataset is:

- Even though there is no duplicate data, the code is given to ensure clarity of knowledge.

- From the check of data inconsistency, the data is not inconsistent.
- There were few missing values which are removed using the “dropna()” function of pandas.
- There were quite a few outliers, which were removed using the IQR (Inter-Quartile Range) Technique.
- Lastly, for encoding the categorical data, One Hot Encoding Technique is used.

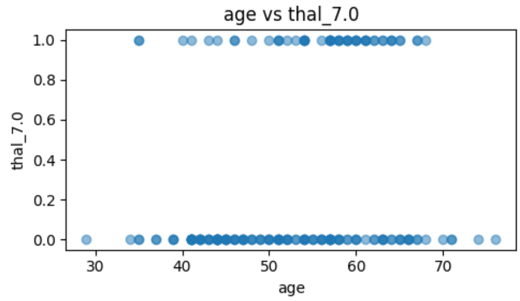
3. *Data Exploration using Visualizations*

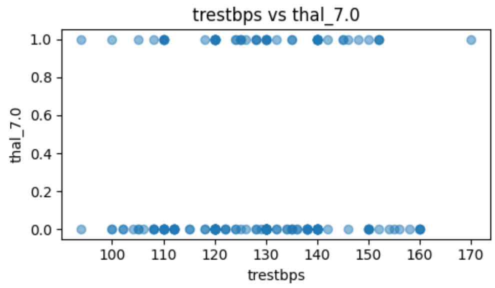
3.1 Creation of Scatter Plots for each feature against the target variable

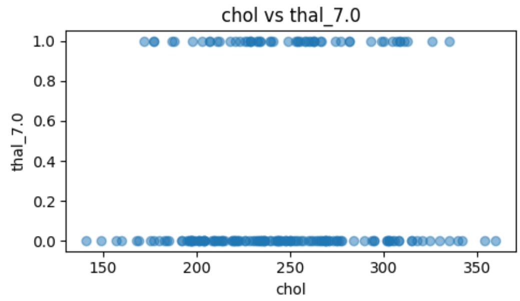
The code for Scatter Plot creation for each feature is as follow:

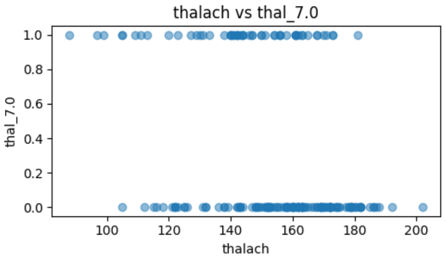
```
# 3.1 Scatter plots of features vs target
target = df.columns[-1]
for col in df.columns[:-1]:
    if pd.api.types.is_numeric_dtype(df[col]):
        plt.figure(figsize=(5, 3))
        plt.scatter(df[col], df[target], alpha=0.5)
        plt.title(f"{col} vs {target}")
        plt.xlabel(col)
        plt.ylabel(target)
        plt.tight_layout()
        plt.show()
```

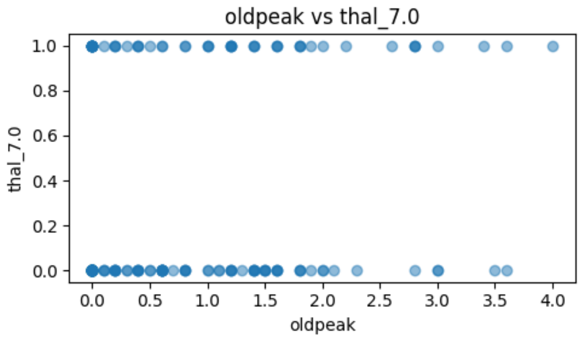
The Scatter Plots are as follows:

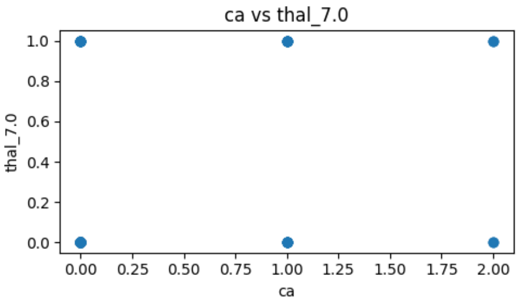


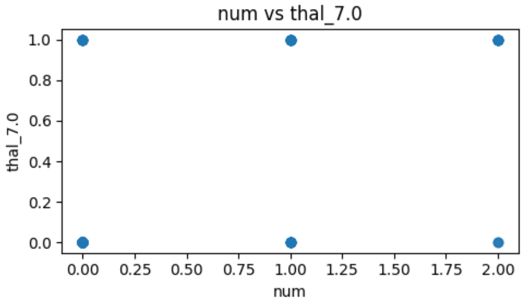


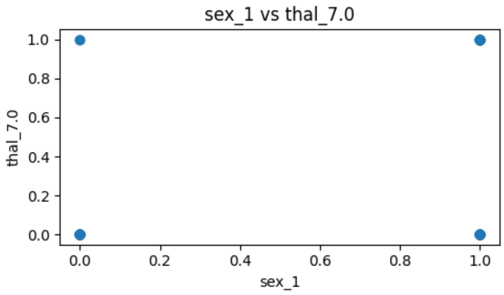


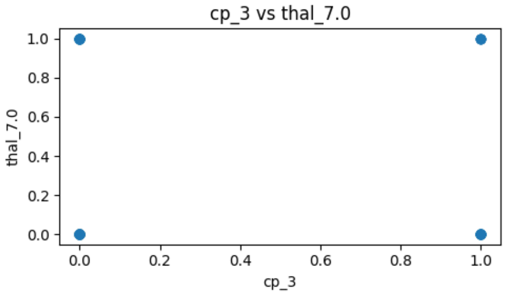


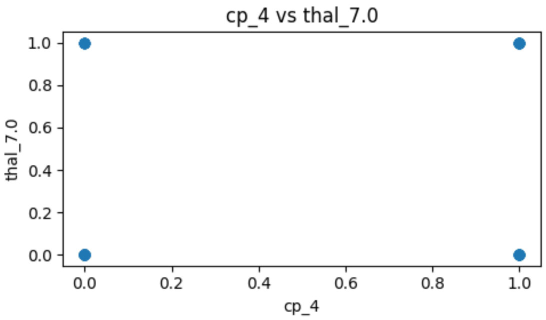

oldpeak vs thal_7.0
thal_7.0
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
oldpeak

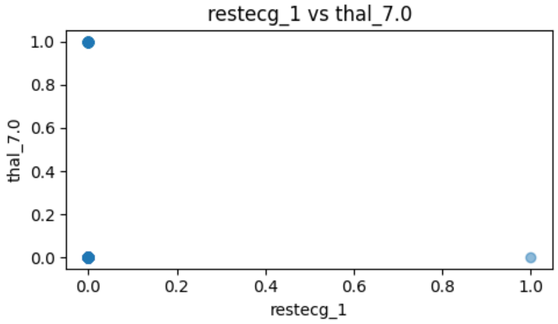

ca vs thal_7.0
thal_7.0
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
ca

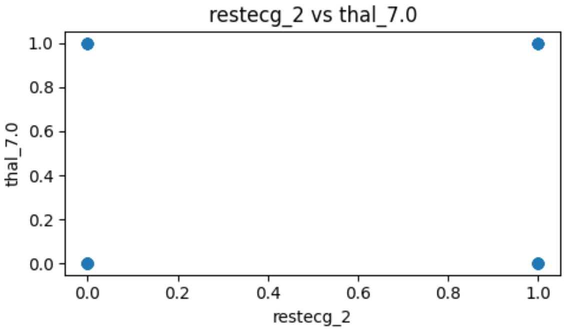

num vs thal_7.0
thal_7.0
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
num

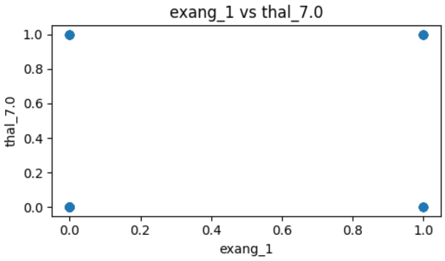

sex_1 vs thal_7.0
thal_7.0
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
sex_1

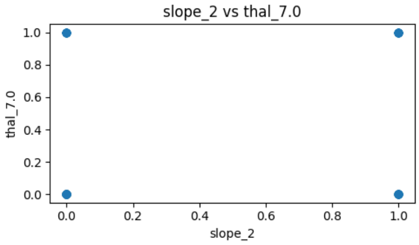

cp_3 vs thal_7.0
thal_7.0
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
cp_3

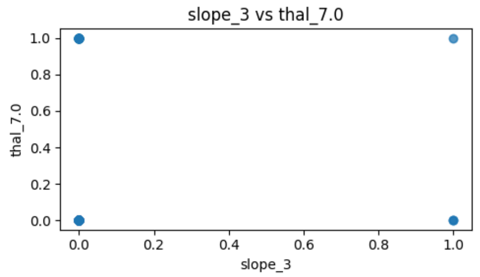

cp_4 vs thal_7.0
thal_7.0
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
cp_4

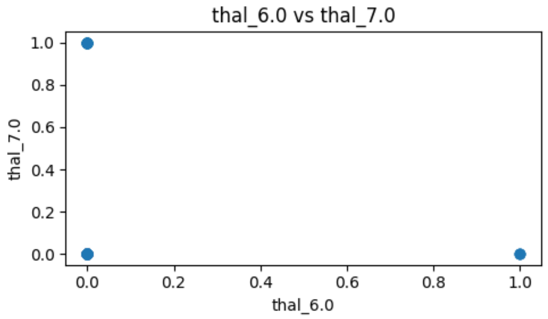

restecg_1 vs thal_7.0
thal_7.0
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
restecg_1


restecg_2 vs thal_7.0
thal_7.0
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
restecg_2


exang_1 vs thal_7.0
thal_7.0
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
exang_1


slope_2 vs thal_7.0
thal_7.0
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
slope_2





3.2 <u>Performing EDA using 2 additional visualization (like pair plot, heat map, correlation plot, regression plot)</u>

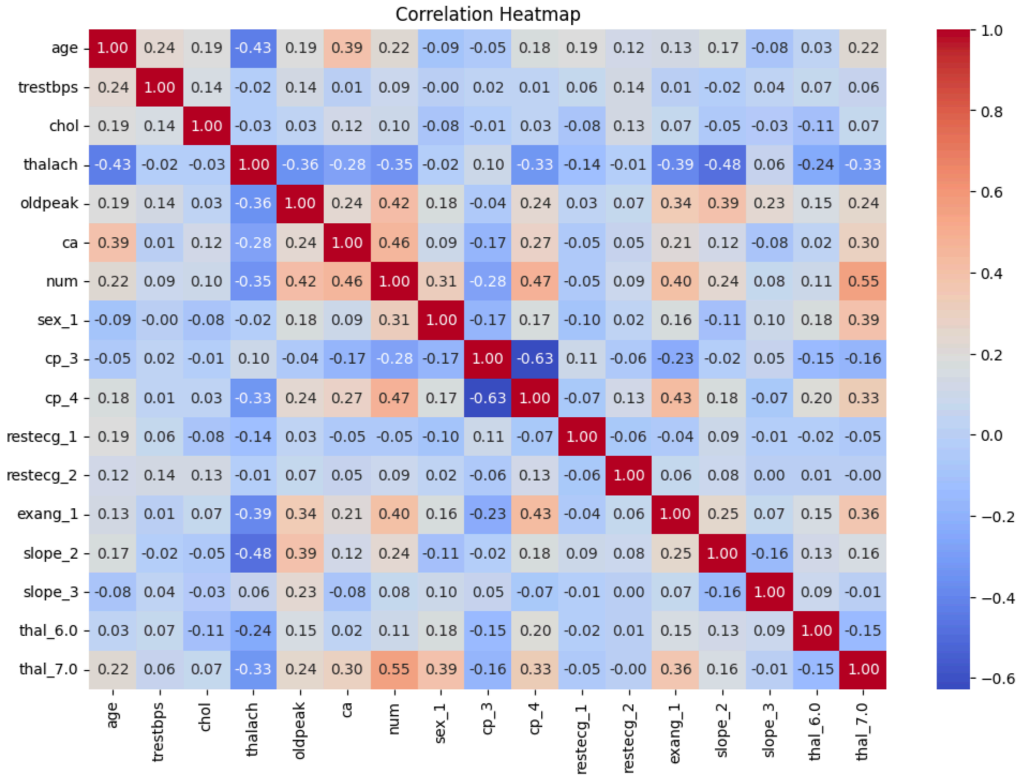
For performing EDA, these 2 visualization can be used namely, Heatmaps and Pair Plots for the following reasons:

- **Correlation Heatmap** helps in identifying which numerical features are strongly correlated with the target.
- **Pair plots** provide a visual understanding of relationships and class separability between top features and the target variable, especially in binary or multiclass problems.
- **Regression plots** are mainly useful for showing linear trends between two continuous variables. Since the target variable is categorical (heart disease presence), regression plots are not viable here.

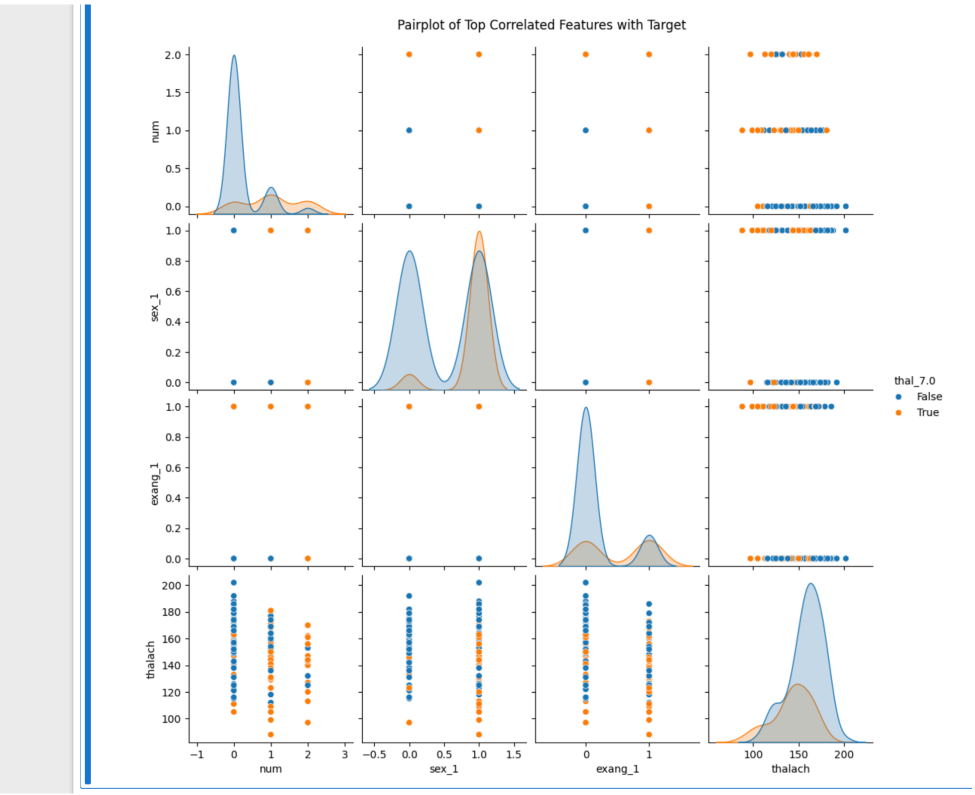
Therefore, the code for the heat maps and pair plot are as follows:

```
# 3.2 Heatmap and pairplot
corr = df.corr()

plt.figure(figsize=(12, 8))
sns.heatmap(corr, annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()

top_corr =
corr[target].abs().sort_values(ascending=False).index[1:5].tolist()
sns.pairplot(df[top_corr + [target]], hue=target)
plt.suptitle("Pairplot of Top Correlated Features with Target", y=1.02)
plt.show()
```

Correlation Heatmap



Pair Plot


Pairplot of Top Correlated Features with Target
num
sex_1
exang_1
thalach
num
sex_1
exang_1
thalach
thal_7.0
False
True