Assignment 1

```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score,
ConfusionMatrixDisplay
import matplotlib.pyplot as plt
import seaborn as sns

#
=======================================================================
========
# Part 1: Heart Disease Dataset Analysis
#
=======================================================================
========
print("--- Part 1: Heart Disease Dataset Analysis ---")

try:
    # Load the dataset (make sure Heart.csv is in the same folder as this script)
    df = pd.read_csv('Heart.csv')

    # a) Find Shape of Data
    print("\na) Shape of the data:")
    print(df.shape)

    # b) Find Missing Values
    print("\nb) Missing values in each column:")
    print(df.isnull().sum())

    # c) Find data type of each column
    print("\nc) Data type of each column:")
    print(df.dtypes)

    # d) Find number of zeros in each column
    print("\nd) Count of zeros in each column:")
    print((df == 0).sum())

    # e) Find Mean age of patients
    print("\ne) Mean age of patients:")
    print(df['Age'].mean())

    # f) Extract specific columns and split dataset into train/test
    print("\nf) Extracting 'Age', 'Sex', 'ChestPain', 'RestBP', 'Chol' and splitting data...")
    data = df[['Age', 'Sex', 'ChestPain', 'RestBP', 'Chol']]

    # Split data into 75% training and 25% testing
    train, test = train_test_split(data, test_size=0.25, random_state=42)

    print("Shape of the training data:", train.shape)
    print("Shape of the testing data:", test.shape)

except FileNotFoundError:
    # Error message if dataset is missing
    print("\nError: 'Heart.csv' not found. Please download it from")
    print("https://www.kaggle.com/zhaoyingzhu/heartcsv and place it in the same folder.")


#
======================================================================
========
# Part 2: Confusion Matrix and Performance Metrics
#
======================================================================
========
# Scenario details:
# - Total samples = 500
# - Predicted 100 as positive, but only 45 were actually positive
# - Total actual positives = 50
print("\n\n--- Part 2: Confusion Matrix and Performance Metrics ---")

# Calculate TP, FP, FN, TN values from given scenario
total_samples = 500
actual_positives = 50
predicted_positives = 100
true_positives = 45

# Actual labels: 50 positives (1) and 450 negatives (0)
y_actual = np.array([1] * actual_positives + [0] * (total_samples - actual_positives))

# Predicted labels: 45 TP, 5 FN, 55 FP, remaining TN
y_predicted = np.array(
    [1] * true_positives +                              # True Positives
    [0] * (actual_positives - true_positives) +           # False Negatives
    [1] * (predicted_positives - true_positives) +        # False Positives
    [0] * ((total_samples - actual_positives) - (predicted_positives - true_positives))  # True
Negatives
)
np.random.shuffle(y_predicted)  # Shuffle predictions to avoid ordered data

# Confusion matrix
print("\nConfusion Matrix:")
```

```
cm = confusion_matrix(y_actual, y_predicted)
print(cm)

# Plot confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.title("Confusion Matrix for COVID-19 Test Predictions")
# plt.show()  # Uncomment if running interactively

# Performance metrics
print("\nPerformance Metrics:")
accuracy = accuracy_score(y_actual, y_predicted)
print(f"I.   Accuracy: {accuracy:.2f}")

# Precision, Recall, and F1-score from classification report
report = classification_report(y_actual, y_predicted, target_names=['Negative', 'Positive'],
output_dict=True)

precision = report['Positive']['precision']
print(f"II.  Precision: {precision:.2f}")

recall = report['Positive']['recall']
print(f"III. Recall: {recall:.2f}")

f1_score = report['Positive']['f1-score']
print(f"IV.  F-1 Score: {f1_score:.2f}")

# Full classification report
print("\nFull Classification Report:")
print(classification_report(y_actual, y_predicted, target_names=['Negative', 'Positive']))
```