```
import pandas as pd

df = pd.read_csv('/content/HeartDiseaseTrain-Test.csv')

# 3□Data Exploration
print("\nFirst 5 rows of the dataset:")
print(df.head())

print("\nShape of the dataset:", df.shape)
print("\nColumns:", df.columns.tolist())
print("\nData Types & Missing Values:")
df.info()

print("\nSummary Statistics:")
print(df.describe())

print("\nMissing values per column:\n", df.isnull().sum())
print("\nDuplicate rows:", df.duplicated().sum())

# 4□Data Visualization (Optional, for better understanding)
import seaborn as sns
import matplotlib.pyplot as plt

# Example: Plot distribution of target variable (assuming 'target' is the disease indicator)
sns.countplot(x='target', data=df)
plt.title('Distribution of Disease Presence (0=No, 1=Yes)')
plt.show()

# 5□Prepare Features and Target
target = 'target'  # Change to actual target column if named differently
features = df.columns.drop(target)
print("\nFeatures:", features.tolist())

# 6□Convert Categorical Columns
categorical_cols = df.select_dtypes(include=['object']).columns.tolist()
print("\nCategorical Columns:", categorical_cols)

df_encoded = pd.get_dummies(df, drop_first=True)

# 3□Check for Missing Values
print("\n□ Missing values per column:\n", df.isnull().sum())

# Fill missing numeric columns with median
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
for col in numeric_cols:
    if df[col].isnull().sum() > 0:
        median_val = df[col].median()
        df[col].fillna(median_val, inplace=True)
        print(f"Filled missing values in '{col}' with median: {median_val}")

# Fill missing categorical columns with mode
categorical_cols = df.select_dtypes(include=['object']).columns
for col in categorical_cols:
    if df[col].isnull().sum() > 0:
        mode_val = df[col].mode()[0]
        df[col].fillna(mode_val, inplace=True)
        print(f"Filled missing values in '{col}' with mode: {mode_val}")

# 4️⃣ Check for Duplicates
duplicates = df.duplicated().sum()
print(f"\n□ Duplicate rows: {duplicates}")
if duplicates > 0:
    df.drop_duplicates(inplace=True)
    print("□ Duplicates removed.")

# 5️⃣ Detect and Handle Outliers (Optional: here we cap them using IQR method)
def cap_outliers(column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    df[column] = df[column].clip(lower, upper)
    print(f"□ Outliers capped for '{column}'")

for col in numeric_cols:
    cap_outliers(col)

# 6️⃣ Encode Categorical Features
print("\n□ Categorical Columns:", categorical_cols.tolist())
df_encoded = pd.get_dummies(df, drop_first=True)

# 7️⃣ Feature Scaling
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X = df_encoded.drop('target', axis=1)  # Replace 'target' with your actual target column
y = df_encoded['target']

X_scaled = scaler.fit_transform(X)

# 8️⃣ Train-Test Split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)

# 9□ Model Training
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# □ Predictions
y_pred = model.predict(X_test)

# □ Evaluation
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# □ Predicting a new patient
# Example: Replace with actual input values
new_patient = {
    'age': 55,
    'sex': 1,
    'cp': 3,
    'trestbps': 140,
    'chol': 250,
    'fbs': 0,
    'restecg': 1,
    'thalach': 150,
    'exang': 0,
    'oldpeak': 2.3,
    'slope': 0,
    'ca': 0,
    'thal': 2
}

# Convert input to DataFrame
new_df = pd.DataFrame([new_patient])

# Combine with original data to ensure same columns
df_temp = pd.concat([df.drop(target, axis=1), new_df], ignore_index=True)
df_temp_encoded = pd.get_dummies(df_temp, drop_first=True)

# Reindex to match training columns
df_temp_encoded = df_temp_encoded.reindex(columns=df_encoded.drop(target, axis=1).columns, fill_value=0)

# Scale new input
new_input_scaled = scaler.transform(df_temp_encoded.tail(1))
```

```python
# Make prediction
predicted_disease = model.predict(new_input_scaled)

print("\nPredicted Disease Presence (1=Yes, 0=No):", predicted_disease[0])
import gradio as gr
import joblib
import pandas as pd

# Load the trained model
#model = joblib.load("heart_disease_model.pkl")

# Prediction function
def predict_heart_disease(age, sex, chest_pain_type, resting_blood_pressure, cholestoral,
                          fasting_blood_sugar, rest_ecg, max_heart_rate, exercise_induced_angina,
                          oldpeak, slope, vessels_colored_by_flourosopy, thalassemia):

    input_data = pd.DataFrame({
        "age": [age],
        "sex": [sex],
        "chest_pain_type": [chest_pain_type],
        "resting_blood_pressure": [resting_blood_pressure],
        "cholestoral": [cholestoral],
        "fasting_blood_sugar": [fasting_blood_sugar],
        "rest_ecg": [rest_ecg],
        "Max_heart_rate": [max_heart_rate],
        "exercise_induced_angina": [exercise_induced_angina],
        "oldpeak": [oldpeak],
        "slope": [slope],
        "vessels_colored_by_flourosopy": [vessels_colored_by_flourosopy],
        "thalassemia": [thalassemia]
    })

    prediction = model.predict(input_data)[0]
    return "□ High Risk of Heart Disease" if prediction == 1 else "□ Low Risk of Heart Disease"

# Gradio interface
demo = gr.Interface(
    fn=predict_heart_disease,
    inputs=[
        gr.Number(label="Age"),
        gr.Radio(["Male", "Female"], label="Sex"),
        gr.Dropdown(["Typical angina", "Atypical angina", "Non-anginal pain", "Asymptomatic"], label="Chest Pain Type"),
        gr.Number(label="Resting Blood Pressure"),
        gr.Number(label="Cholesterol"),
        gr.Radio(["Lower than 120 mg/ml", "Greater than 120 mg/ml"], label="Fasting Blood Sugar"),
        gr.Dropdown(["Normal", "ST-T wave abnormality", "Left ventricular hypertrophy"], label="Rest ECG"),
        gr.Number(label="Max Heart Rate"),
        gr.Radio(["Yes", "No"], label="Exercise Induced Angina"),
        gr.Number(label="Oldpeak"),
        gr.Dropdown(["Upsloping", "Flat", "Downsloping"], label="Slope"),
        gr.Dropdown(["Zero", "One", "Two", "Three"], label="Vessels Colored by Fluoroscopy"),
        gr.Dropdown(["Fixed Defect", "Normal", "Reversable Defect"], label="Thalassemia")
    ],
    outputs="text",
    title="Heart Disease Risk Predictor",
    description="This app predicts the risk of heart disease based on patient data."
)

# Launch the app
demo.launch()
```