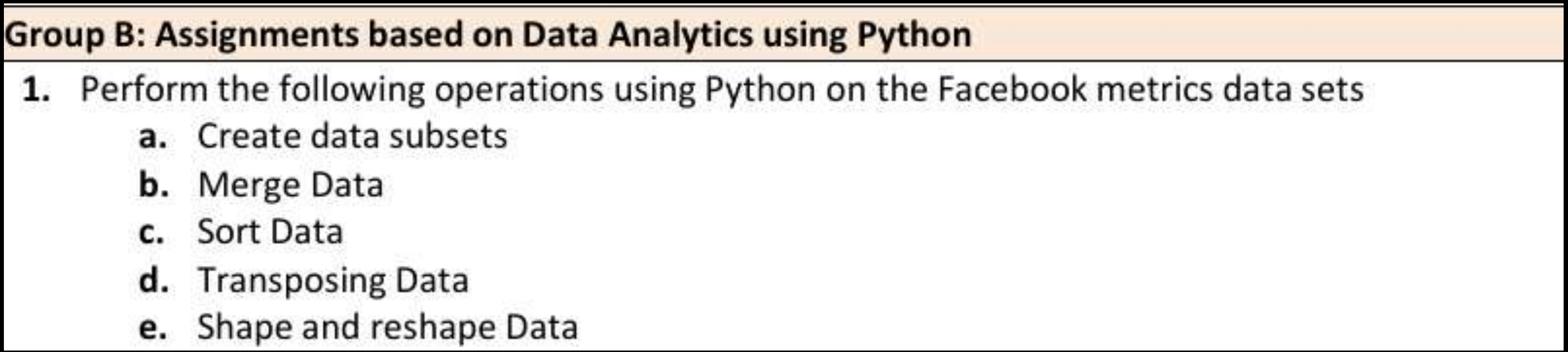
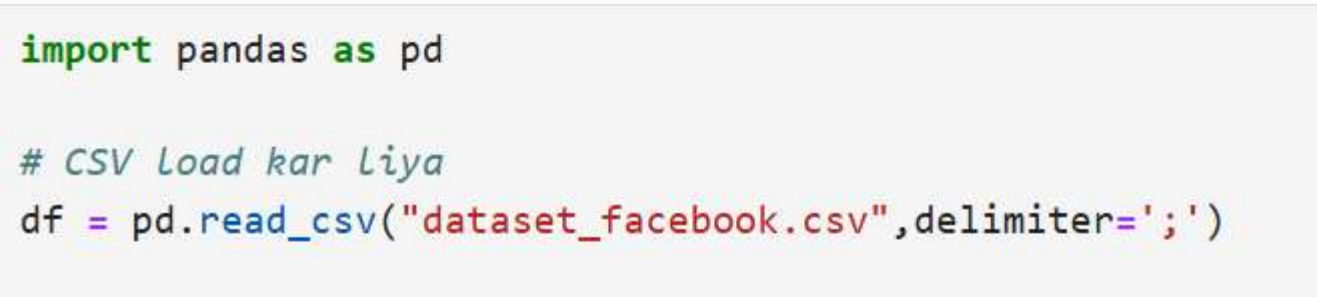


Group B: Assignments based on Data Analytics using Python

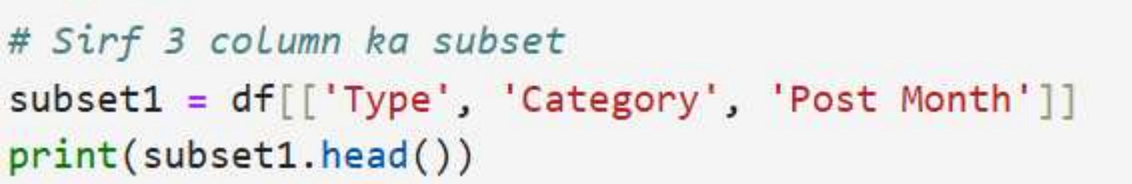
1. Perform the following operations using Python on the Facebook metrics data sets
    a. Create data subsets
    b. Merge Data
    c. Sort Data
    d. Transposing Data
    e. Shape and reshape Data

Step 1: Dataset Load karna

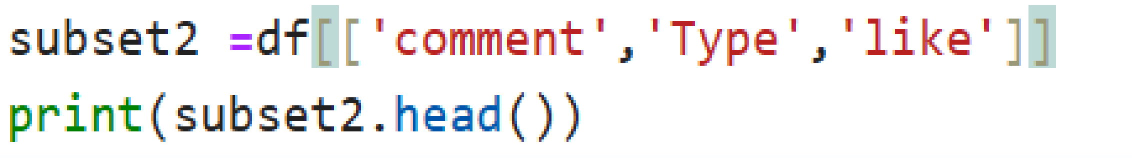


```
import pandas as pd

# CSV Load kar liya
df = pd.read_csv("dataset_facebook.csv",delimiter=';')
```

❖ a. Create data subsets

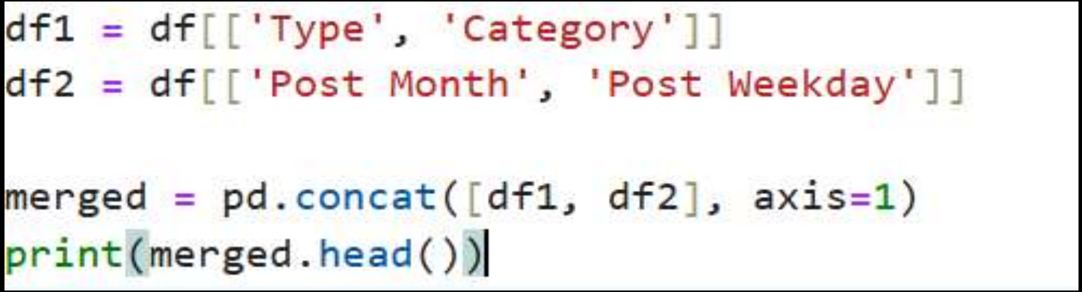


```
# Sirf 3 column ka subset
subset1 = df[['Type', 'Category', 'Post Month']]
print(subset1.head())
```



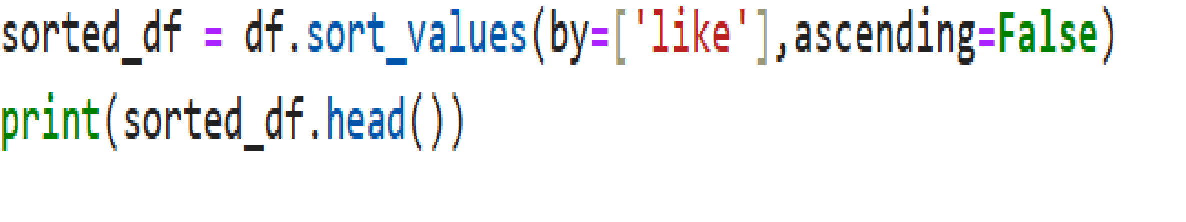
```
subset2 =df[['comment','Type','like']]
print(subset2.head())
```

❖ b. Merge Data

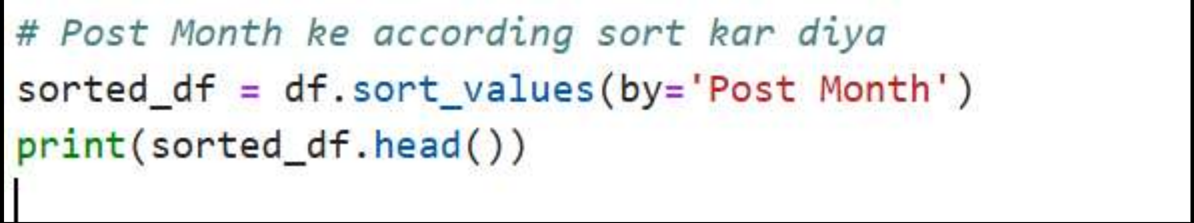


```
df1 = df[['Type', 'Category']]
df2 = df[['Post Month', 'Post Weekday']]

merged = pd.concat([df1, df2], axis=1)
print(merged.head())
```

⬍ c. Sort Data

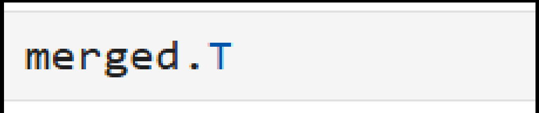


```
sorted_df = df.sort_values(by=['like'],ascending=False)
print(sorted_df.head())
```

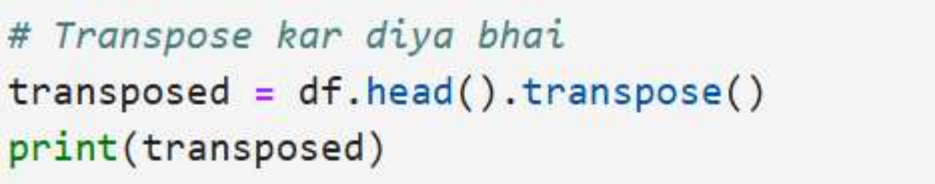


```
# Post Month ke according sort kar diya
sorted_df = df.sort_values(by='Post Month')
print(sorted_df.head())
```

- d. Transposing Data=>(Row ko column aur column ko row bana dena)

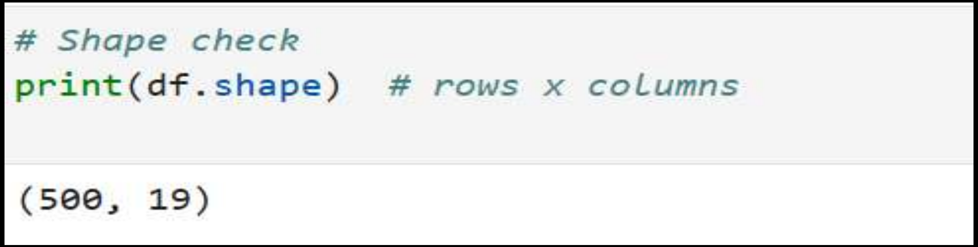


```
merged.T
```



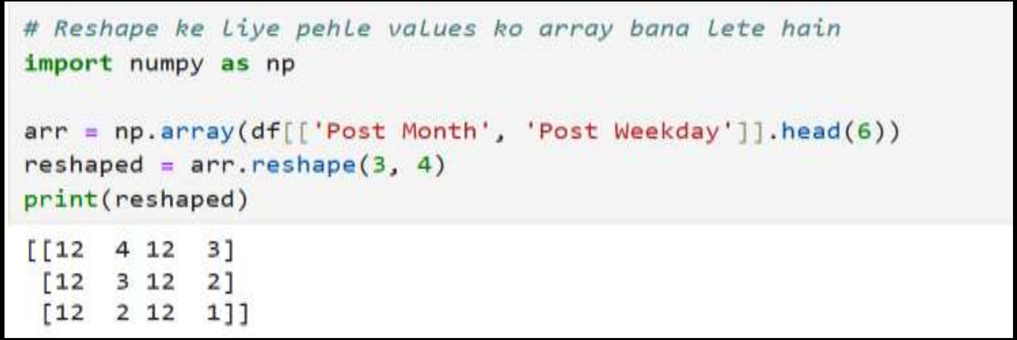
```
# Transpose kar diya bhai
transposed = df.head().transpose()
print(transposed)
```

❖ e. Shape and reshape Data=>(Data ka shape check karna ya usko naya shape dena (reshape karna)



```
# Shape check
print(df.shape)  # rows x columns
```

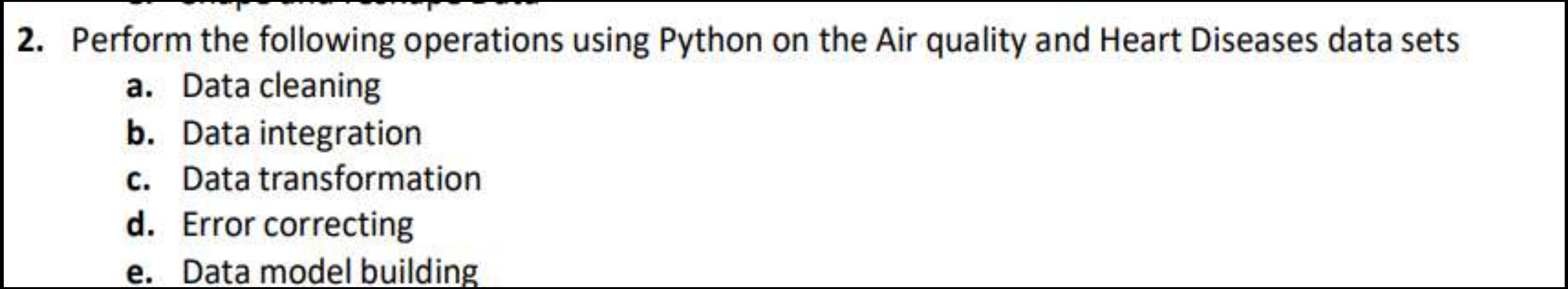
```
(500, 19)
```



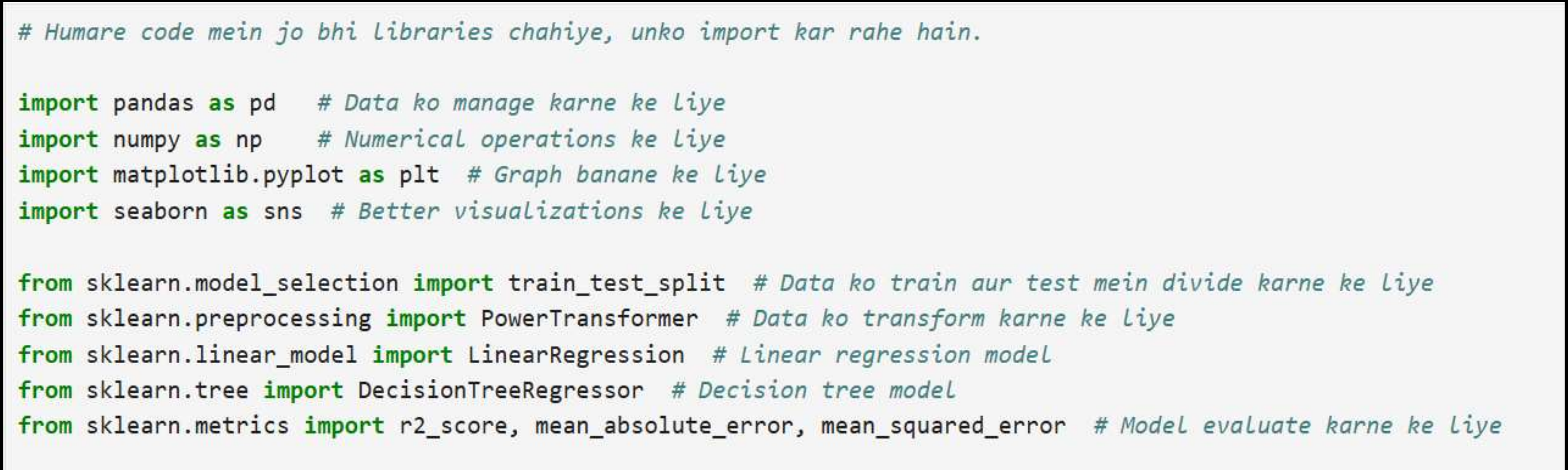
```
# Reshape ke liye pehle values ko array bana lete hain
import numpy as np

arr = np.array(df[['Post Month', 'Post Weekday']].head(6))
reshaped = arr.reshape(3, 4)
print(reshaped)
```

```
[[12  4 12  3]
 [12  3 12  2]
 [12  2 12  1]]
```

TE IT – DSBDA Practical (Group B-2)

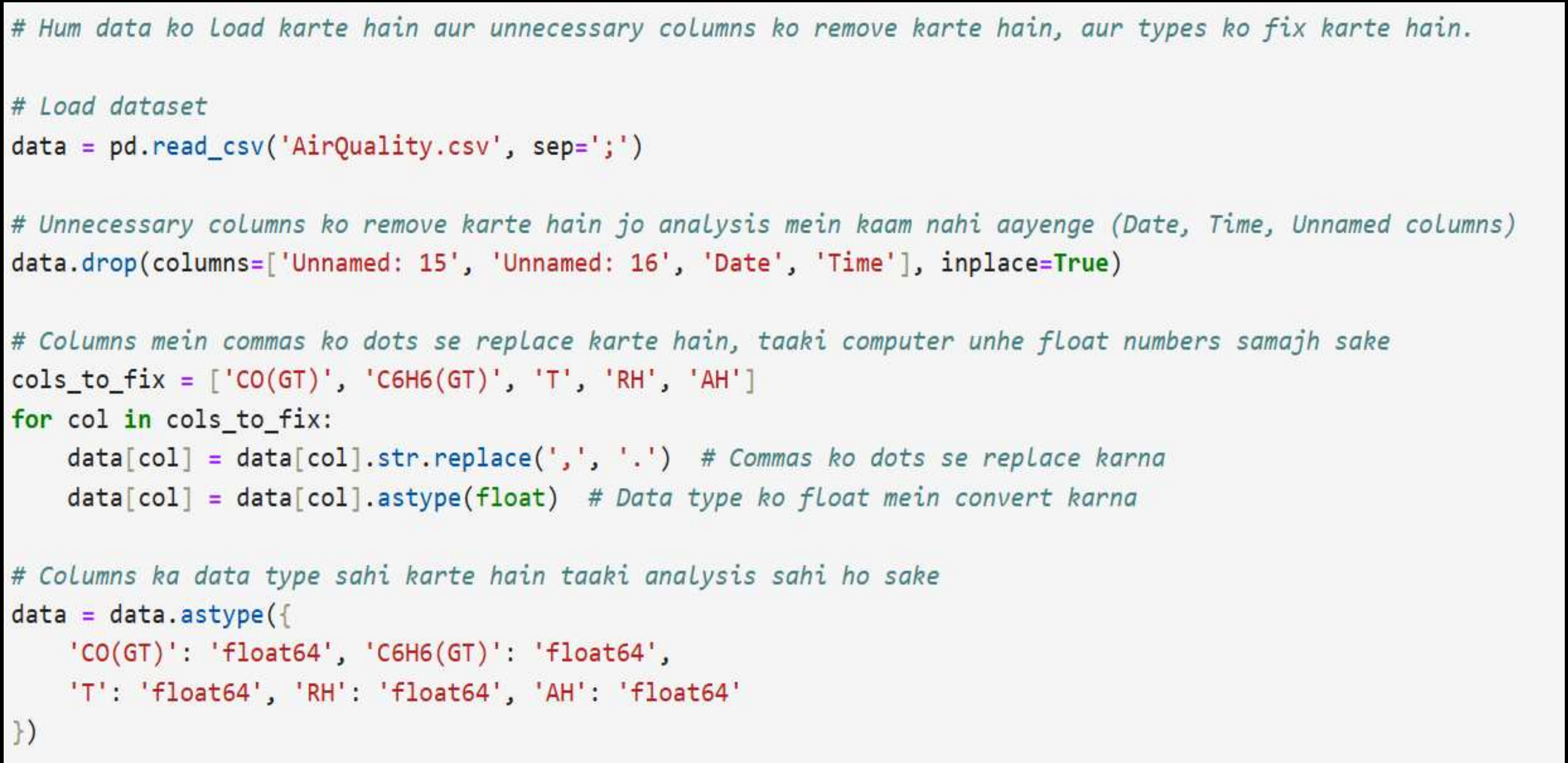


2. Perform the following operations using Python on the Air quality and Heart Diseases data sets
   - a. Data cleaning
   - b. Data integration
   - c. Data transformation
   - d. Error correcting
   - e. Data model building



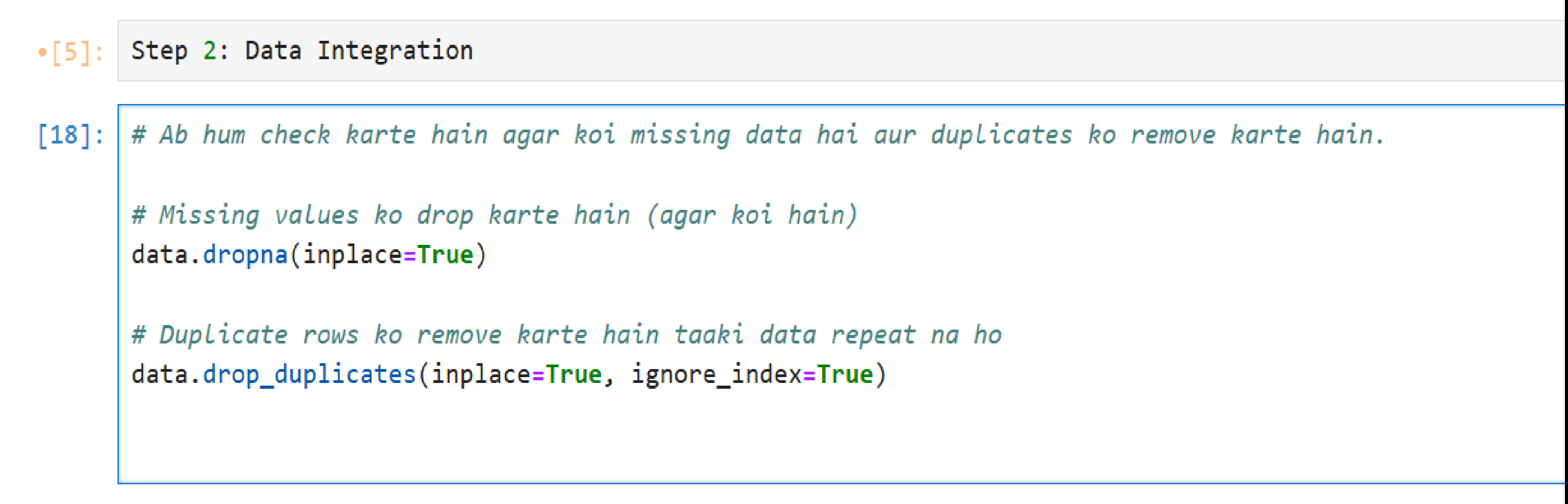
```python
# Humare code mein jo bhi libraries chahiye, unko import kar rahe hain.

import pandas as pd   # Data ko manage karne ke liye
import numpy as np    # Numerical operations ke liye
import matplotlib.pyplot as plt  # Graph banane ke liye
import seaborn as sns  # Better visualizations ke liye

from sklearn.model_selection import train_test_split  # Data ko train aur test mein divide karne ke liye
from sklearn.preprocessing import PowerTransformer  # Data ko transform karne ke liye
from sklearn.linear_model import LinearRegression  # Linear regression model
from sklearn.tree import DecisionTreeRegressor  # Decision tree model
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error  # Model evaluate karne ke liye
```

Step 1: Data Cleaning



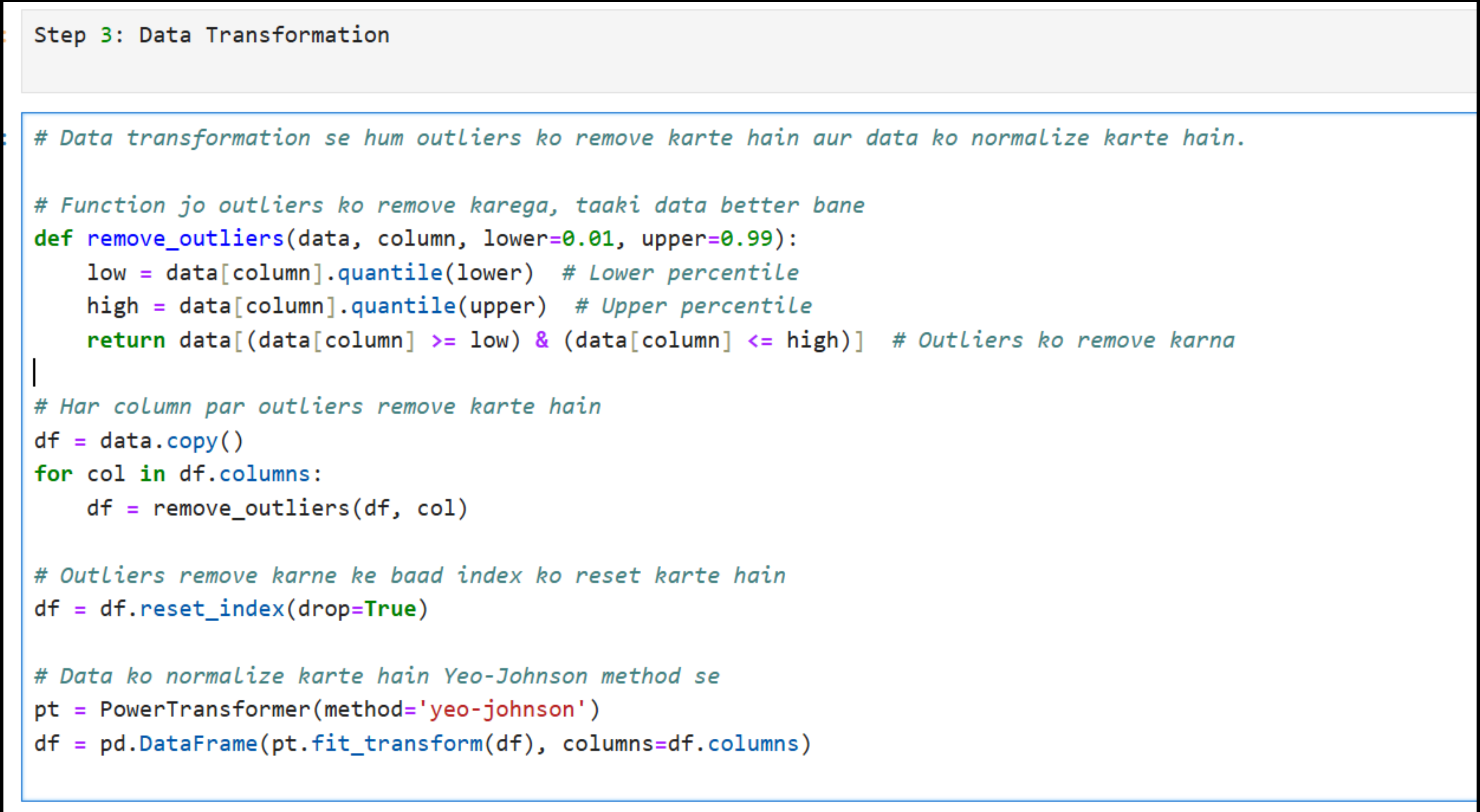
```
# Hum data ko load karte hain aur unnecessary columns ko remove karte hain, aur types ko fix karte hain.

# Load dataset
data = pd.read_csv('AirQuality.csv', sep=';')

# Unnecessary columns ko remove karte hain jo analysis mein kaam nahi aayenge (Date, Time, Unnamed columns)
data.drop(columns=['Unnamed: 15', 'Unnamed: 16', 'Date', 'Time'], inplace=True)

# Columns mein commas ko dots se replace karte hain, taaki computer unhe float numbers samajh sake
cols_to_fix = ['CO(GT)', 'C6H6(GT)', 'T', 'RH', 'AH']
for col in cols_to_fix:
    data[col] = data[col].str.replace(',', '.')  # Commas ko dots se replace karna
    data[col] = data[col].astype(float)  # Data type ko float mein convert karna

# Columns ka data type sahi karte hain taaki analysis sahi ho sake
data = data.astype({
    'CO(GT)': 'float64', 'C6H6(GT)': 'float64',
    'T': 'float64', 'RH': 'float64', 'AH': 'float64'
})
```

Step 2: Data Integration



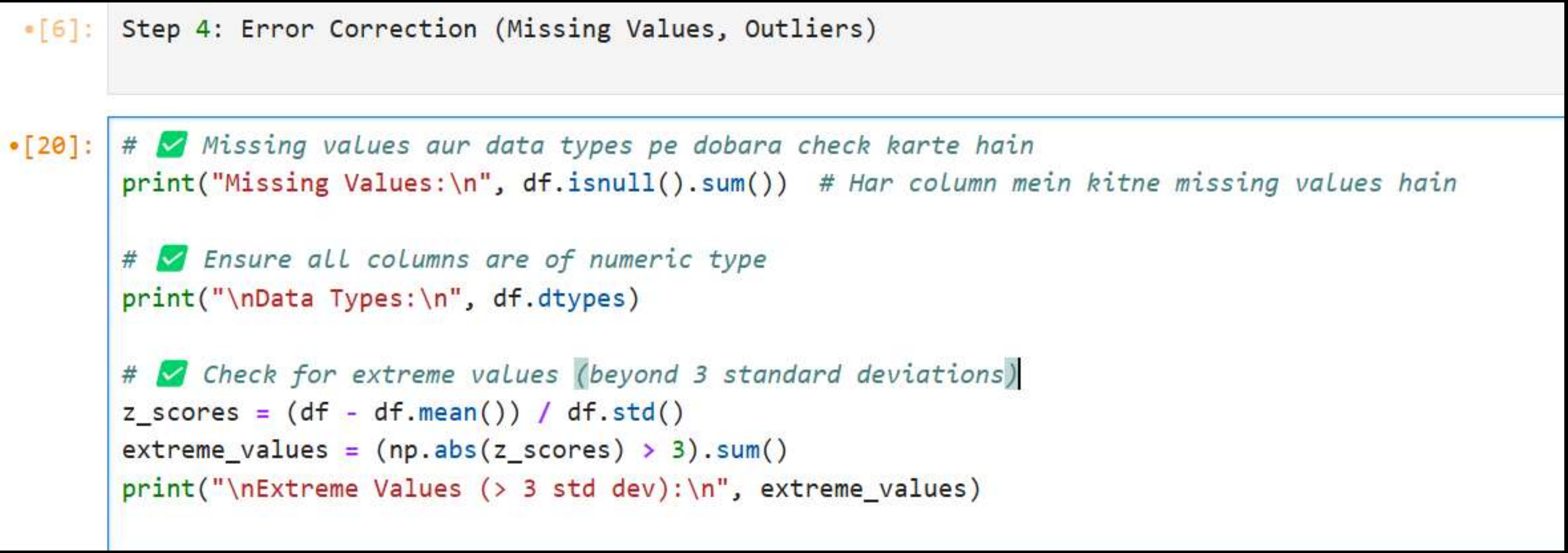
```
Step 2: Data Integration
```

```
# Ab hum check karte hain agar koi missing data hai aur duplicates ko remove karte hain.

# Missing values ko drop karte hain (agar koi hain)
data.dropna(inplace=True)

# Duplicate rows ko remove karte hain taaki data repeat na ho
data.drop_duplicates(inplace=True, ignore_index=True)
```



Step 3: Data Transformation

```python
# Data transformation se hum outliers ko remove karte hain aur data ko normalize karte hain.

# Function jo outliers ko remove karega, taaki data better bane
def remove_outliers(data, column, lower=0.01, upper=0.99):
    low = data[column].quantile(lower)  # Lower percentile
    high = data[column].quantile(upper)  # Upper percentile
    return data[(data[column] >= low) & (data[column] <= high)]  # Outliers ko remove karna

# Har column par outliers remove karte hain
df = data.copy()
for col in df.columns:
    df = remove_outliers(df, col)

# Outliers remove karne ke baad index ko reset karte hain
df = df.reset_index(drop=True)

# Data ko normalize karte hain Yeo-Johnson method se
pt = PowerTransformer(method='yeo-johnson')
df = pd.DataFrame(pt.fit_transform(df), columns=df.columns)
```



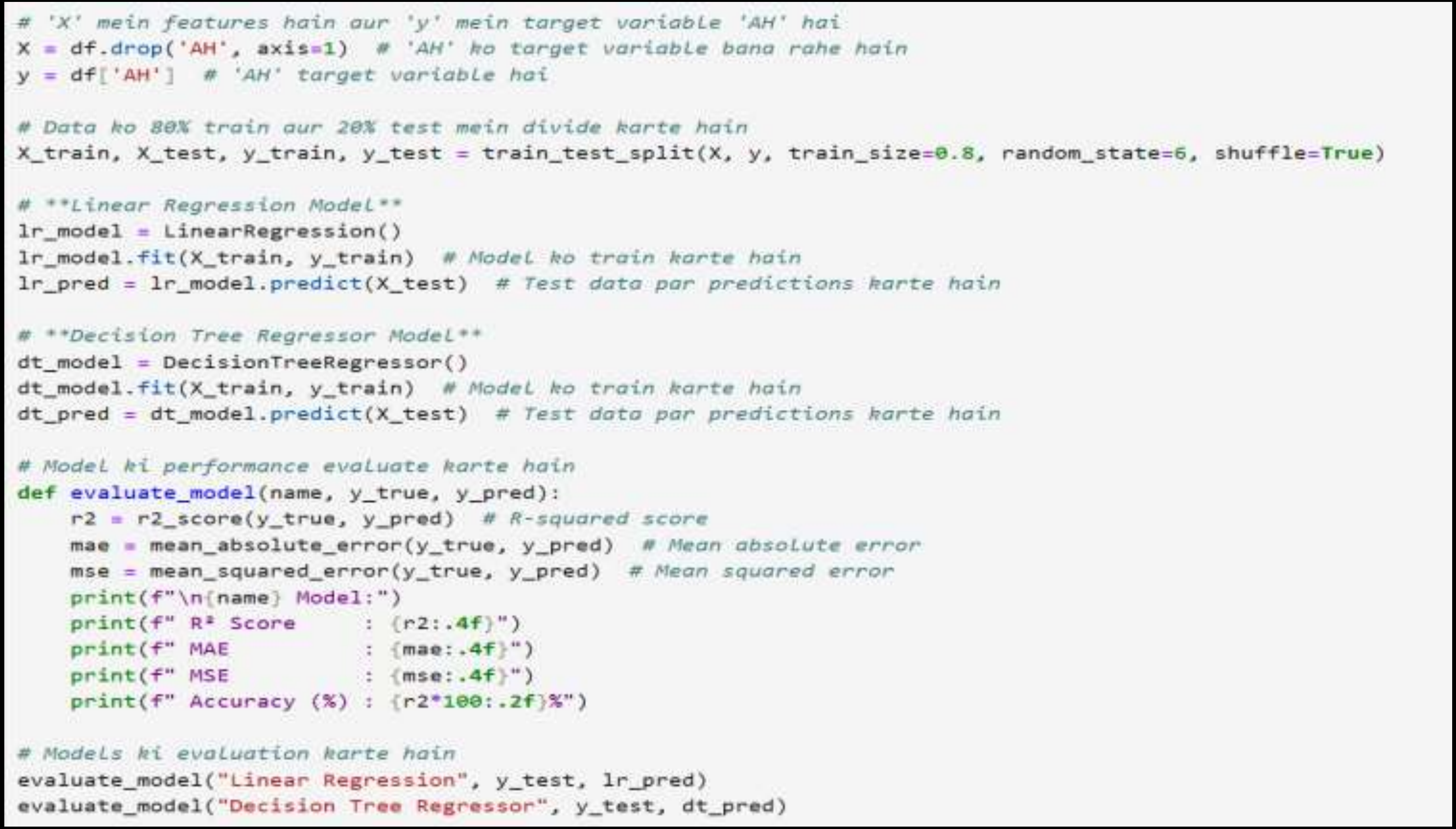
•[6]:

```
Step 4: Error Correction (Missing Values, Outliers)
```

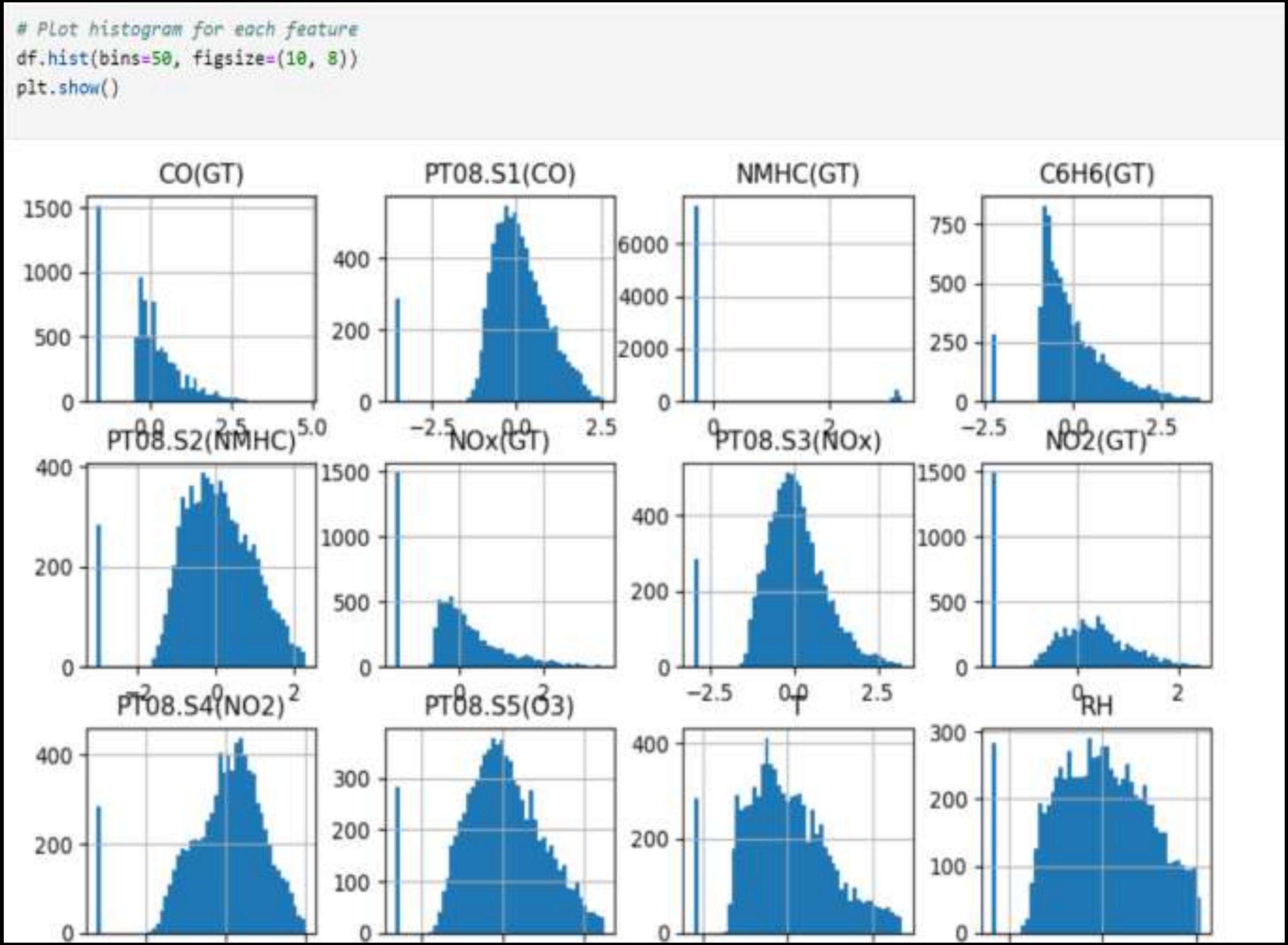
•[20]:

```
# ✅ Missing values aur data types pe dobara check karte hain
print("Missing Values:\n", df.isnull().sum())  # Har column mein kitne missing values hain

# ✅ Ensure all columns are of numeric type
print("\nData Types:\n", df.dtypes)

# ✅ Check for extreme values (beyond 3 standard deviations)
z_scores = (df - df.mean()) / df.std()
extreme_values = (np.abs(z_scores) > 3).sum()
print("\nExtreme Values (> 3 std dev):\n", extreme_values)
```

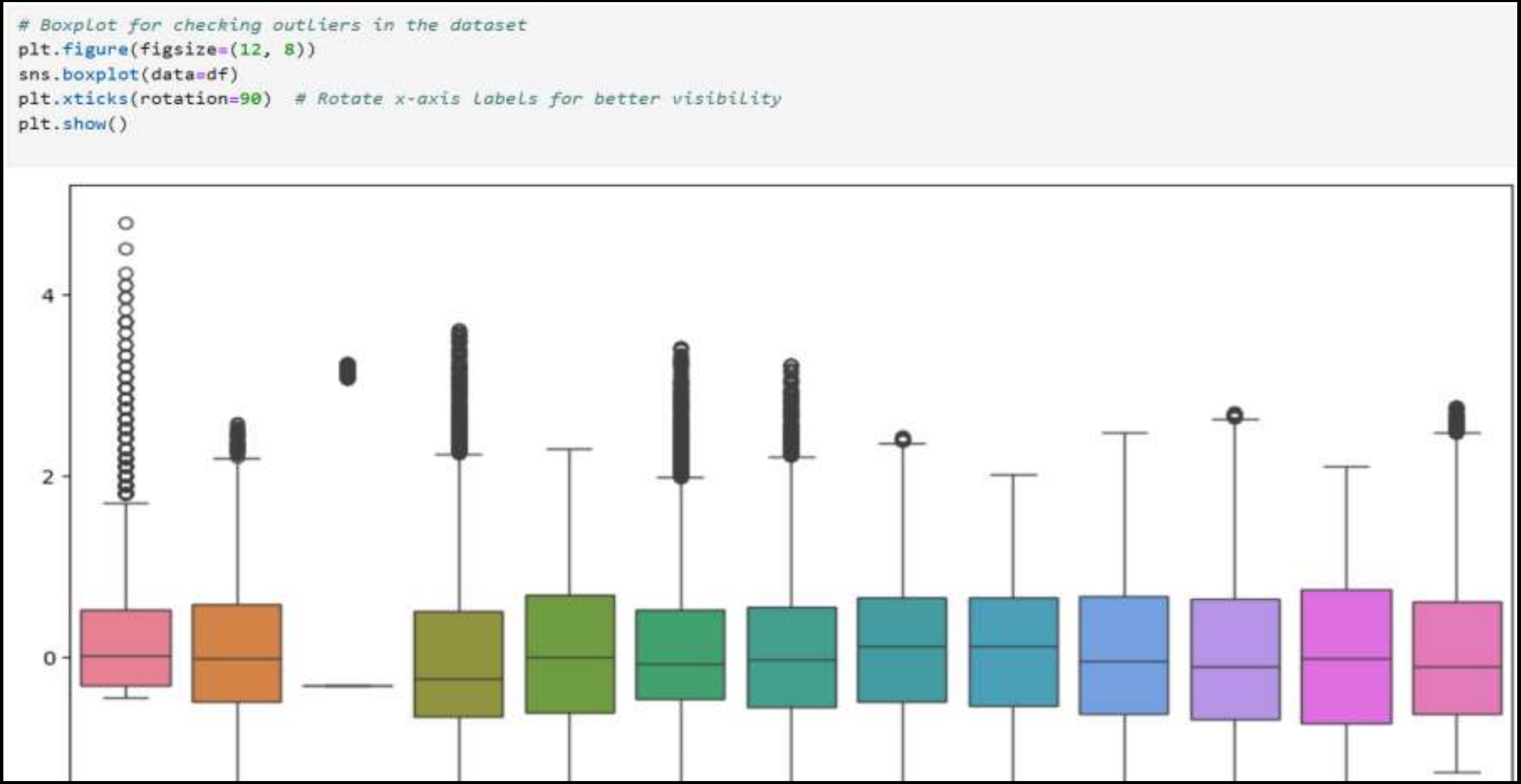
Step 5: Model Building



```
# 'X' mein features hain aur 'y' mein target variable 'AH' hai
X = df.drop('AH', axis=1)  # 'AH' ko target variable bana rahe hain
y = df['AH']  # 'AH' target variable hai

# Data ko 80% train aur 20% test mein divide karte hain
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, random_state=6, shuffle=True)

# **Linear Regression Model**
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)  # Model ko train karte hain
lr_pred = lr_model.predict(X_test)  # Test data par predictions karte hain

# **Decision Tree Regressor Model**
dt_model = DecisionTreeRegressor()
dt_model.fit(X_train, y_train)  # Model ko train karte hain
dt_pred = dt_model.predict(X_test)  # Test data par predictions karte hain

# Model ki performance evaluate karte hain
def evaluate_model(name, y_true, y_pred):
    r2 = r2_score(y_true, y_pred)  # R-squared score
    mae = mean_absolute_error(y_true, y_pred)  # Mean absolute error
    mse = mean_squared_error(y_true, y_pred)  # Mean squared error
    print(f"\n{name} Model:")
    print(f" R² Score     : {r2:.4f}")
    print(f" MAE          : {mae:.4f}")
    print(f" MSE          : {mse:.4f}")
    print(f" Accuracy (%) : {r2*100:.2f}%")

# Models ki evaluation karte hain
evaluate_model("Linear Regression", y_test, lr_pred)
evaluate_model("Decision Tree Regressor", y_test, dt_pred)
```

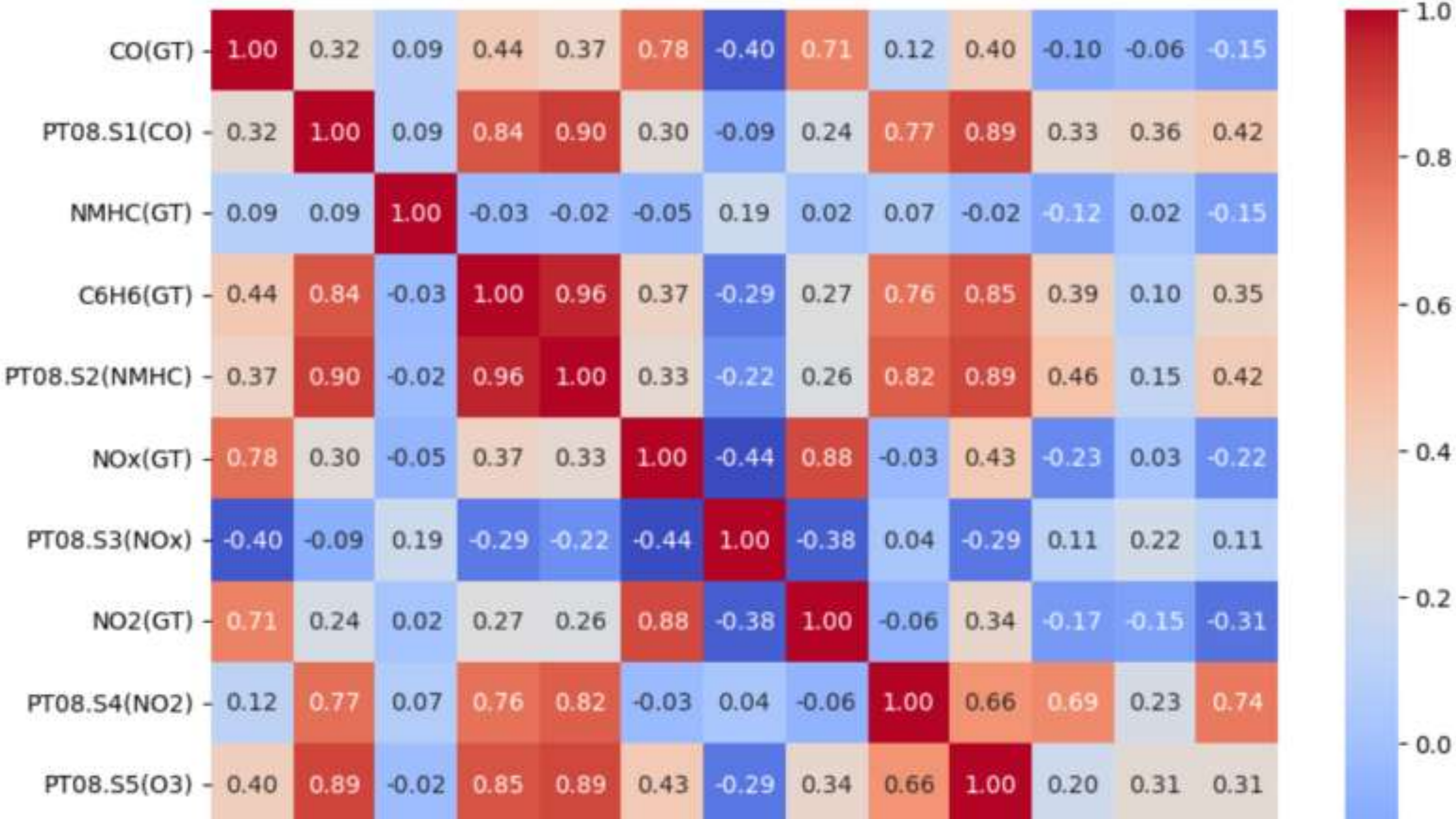


```
# Plot histogram for each feature
df.hist(bins=50, figsize=(10, 8))
plt.show()
```

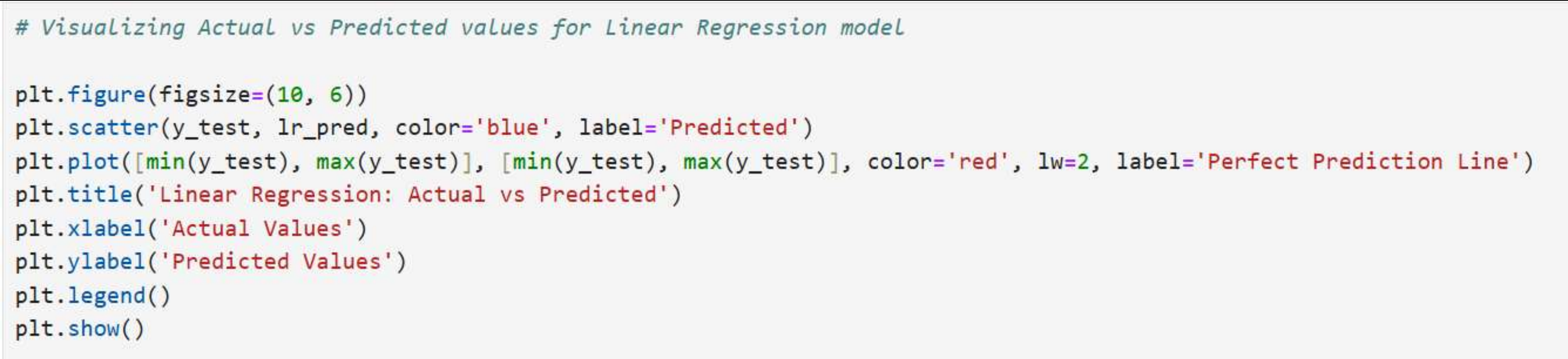


```
# Boxplot for checking outliers in the dataset
plt.figure(figsize=(12, 8))
sns.boxplot(data=df)
plt.xticks(rotation=90)  # Rotate x-axis labels for better visibility
plt.show()
```

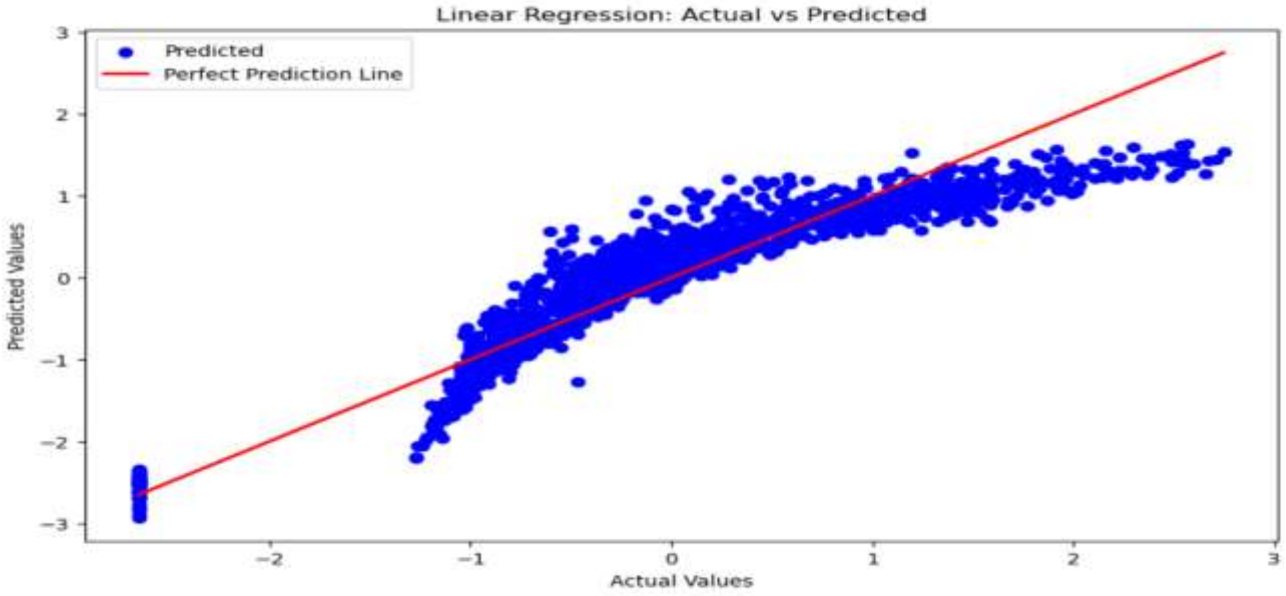
```python
# Correlation heatmap to understand the relationship between features
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.show()
```



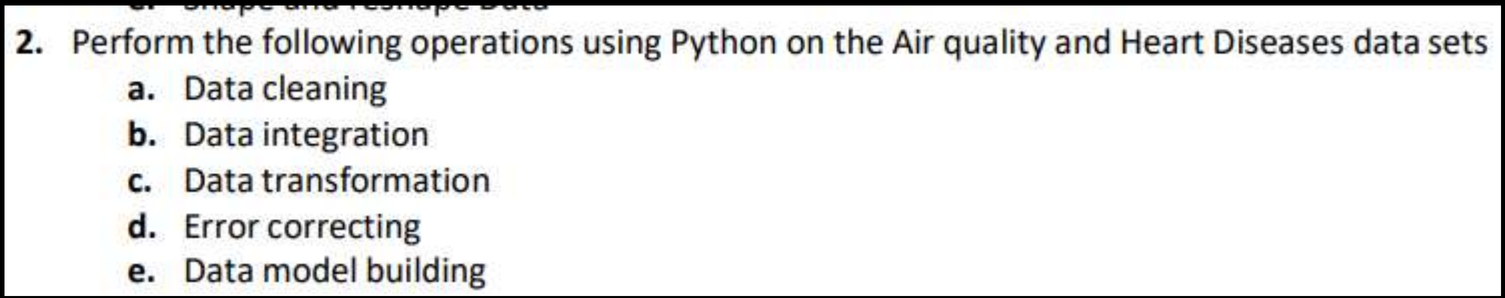
CO(GT)	1.00	0.32	0.09	0.44	0.37	0.78	-0.40	0.71	0.12	0.40	-0.10	-0.06	-0.15
PT08.S1(CO)	0.32	1.00	0.09	0.84	0.90	0.30	-0.09	0.24	0.77	0.89	0.33	0.36	0.42
NMHC(GT)	0.09	0.09	1.00	-0.03	-0.02	-0.05	0.19	0.02	0.07	-0.02	-0.12	0.02	-0.15
C6H6(GT)	0.44	0.84	-0.03	1.00	0.96	0.37	-0.29	0.27	0.76	0.85	0.39	0.10	0.35
PT08.S2(NMHC)	0.37	0.90	-0.02	0.96	1.00	0.33	-0.22	0.26	0.82	0.89	0.46	0.15	0.42
NOx(GT)	0.78	0.30	-0.05	0.37	0.33	1.00	-0.44	0.88	-0.03	0.43	-0.23	0.03	-0.22
PT08.S3(NOx)	-0.40	-0.09	0.19	-0.29	-0.22	-0.44	1.00	-0.38	0.04	-0.29	0.11	0.22	0.11
NO2(GT)	0.71	0.24	0.02	0.27	0.26	0.88	-0.38	1.00	-0.06	0.34	-0.17	-0.15	-0.31
PT08.S4(NO2)	0.12	0.77	0.07	0.76	0.82	-0.03	0.04	-0.06	1.00	0.66	0.69	0.23	0.74
PT08.S5(O3)	0.40	0.89	-0.02	0.85	0.89	0.43	-0.29	0.34	0.66	1.00	0.20	0.31	0.31



```
# Visualizing Actual vs Predicted values for Linear Regression model

plt.figure(figsize=(10, 6))
plt.scatter(y_test, lr_pred, color='blue', label='Predicted')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', lw=2, label='Perfect Prediction Line')
plt.title('Linear Regression: Actual vs Predicted')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.legend()
plt.show()
```

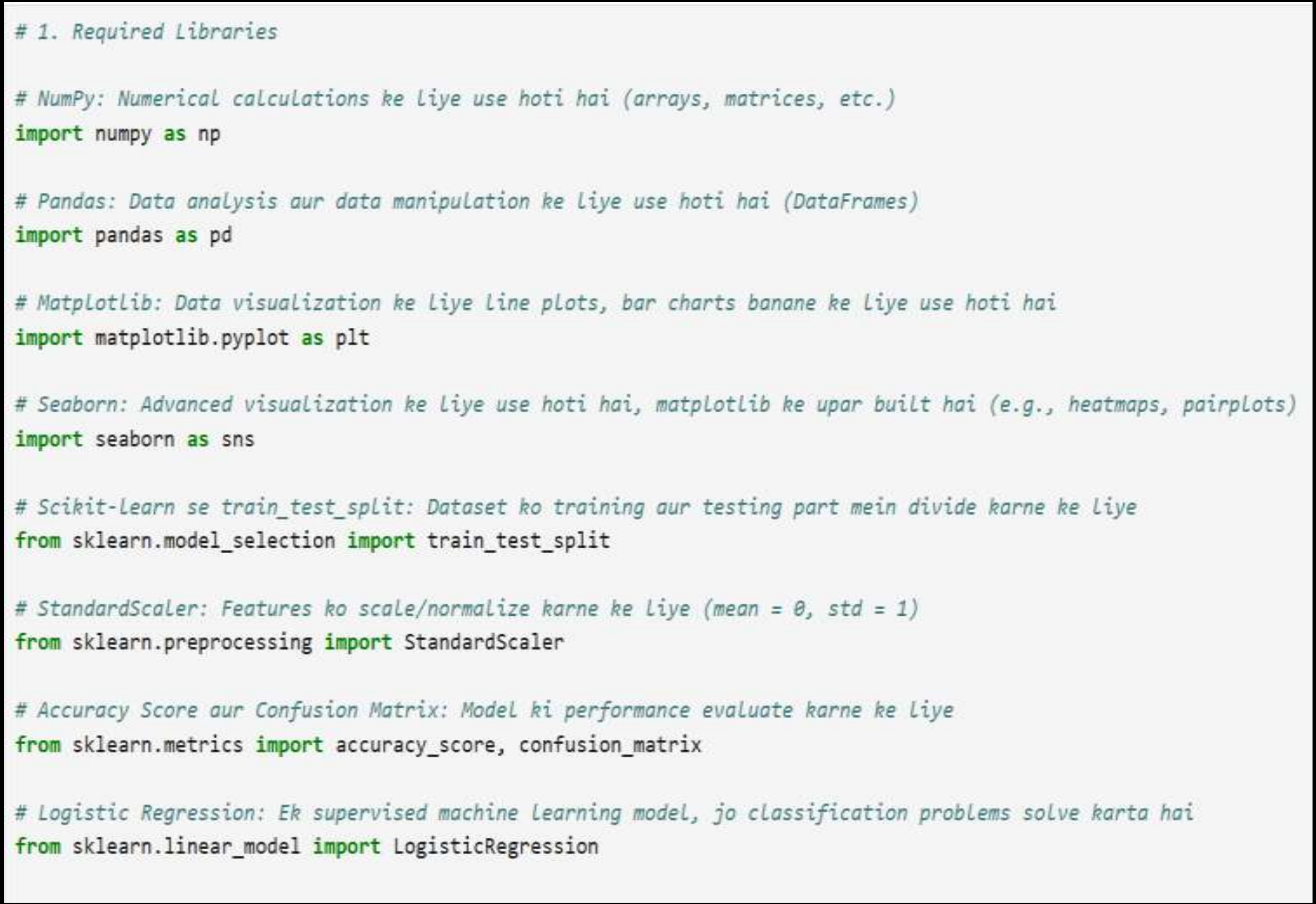


Heart Diseases DataSet

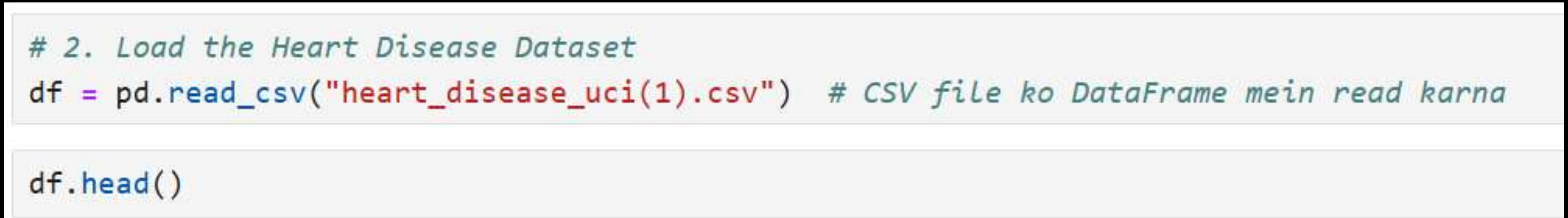


c. Shape and reshape Data

2. Perform the following operations using Python on the Air quality and Heart Diseases data sets
   - **a.** Data cleaning
   - **b.** Data integration
   - **c.** Data transformation
   - **d.** Error correcting
   - **e.** Data model building

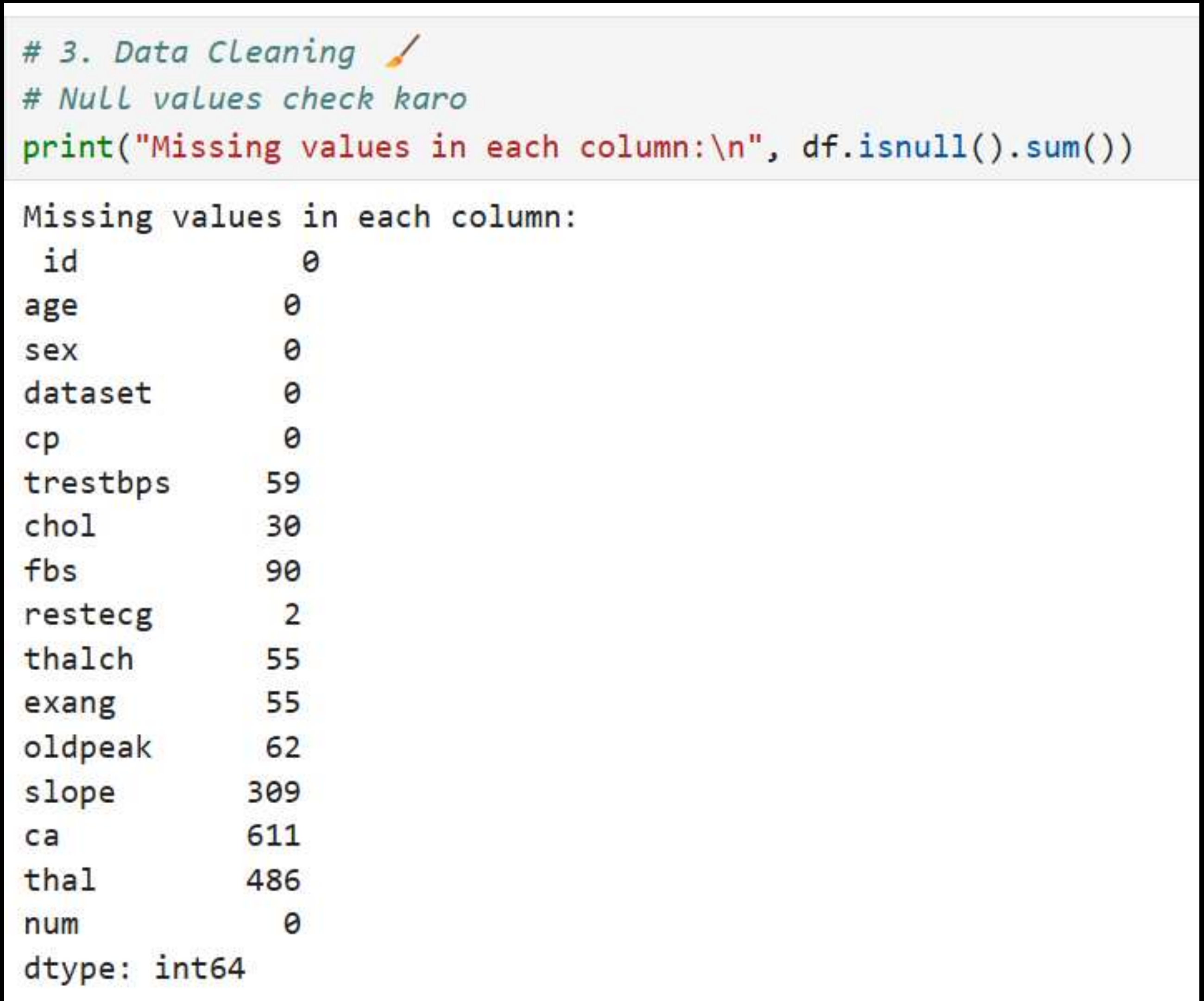


```
# 1. Required Libraries

# NumPy: Numerical calculations ke liye use hoti hai (arrays, matrices, etc.)
import numpy as np

# Pandas: Data analysis aur data manipulation ke liye use hoti hai (DataFrames)
import pandas as pd

# Matplotlib: Data visualization ke liye line plots, bar charts banane ke liye use hoti hai
import matplotlib.pyplot as plt

# Seaborn: Advanced visualization ke liye use hoti hai, matplotlib ke upar built hai (e.g., heatmaps, pairplots)
import seaborn as sns

# Scikit-learn se train_test_split: Dataset ko training aur testing part mein divide karne ke liye
from sklearn.model_selection import train_test_split

# StandardScaler: Features ko scale/normalize karne ke liye (mean = 0, std = 1)
from sklearn.preprocessing import StandardScaler

# Accuracy Score aur Confusion Matrix: Model ki performance evaluate karne ke liye
from sklearn.metrics import accuracy_score, confusion_matrix

# Logistic Regression: Ek supervised machine learning model, jo classification problems solve karta hai
from sklearn.linear_model import LogisticRegression
```



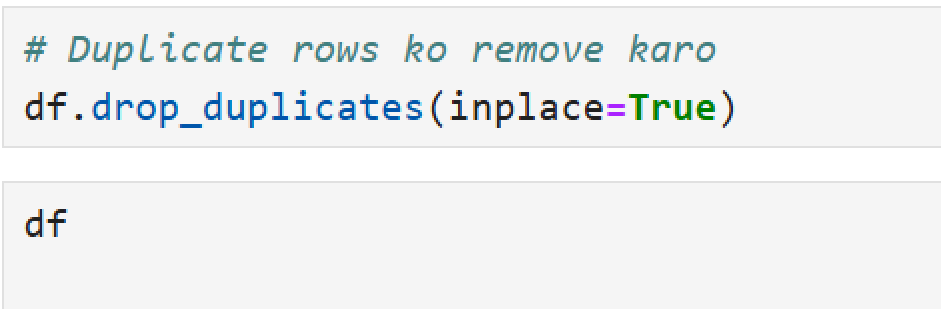
```
# 2. Load the Heart Disease Dataset
df = pd.read_csv("heart_disease_uci(1).csv")  # CSV file ko DataFrame mein read karna
```

```
df.head()
```



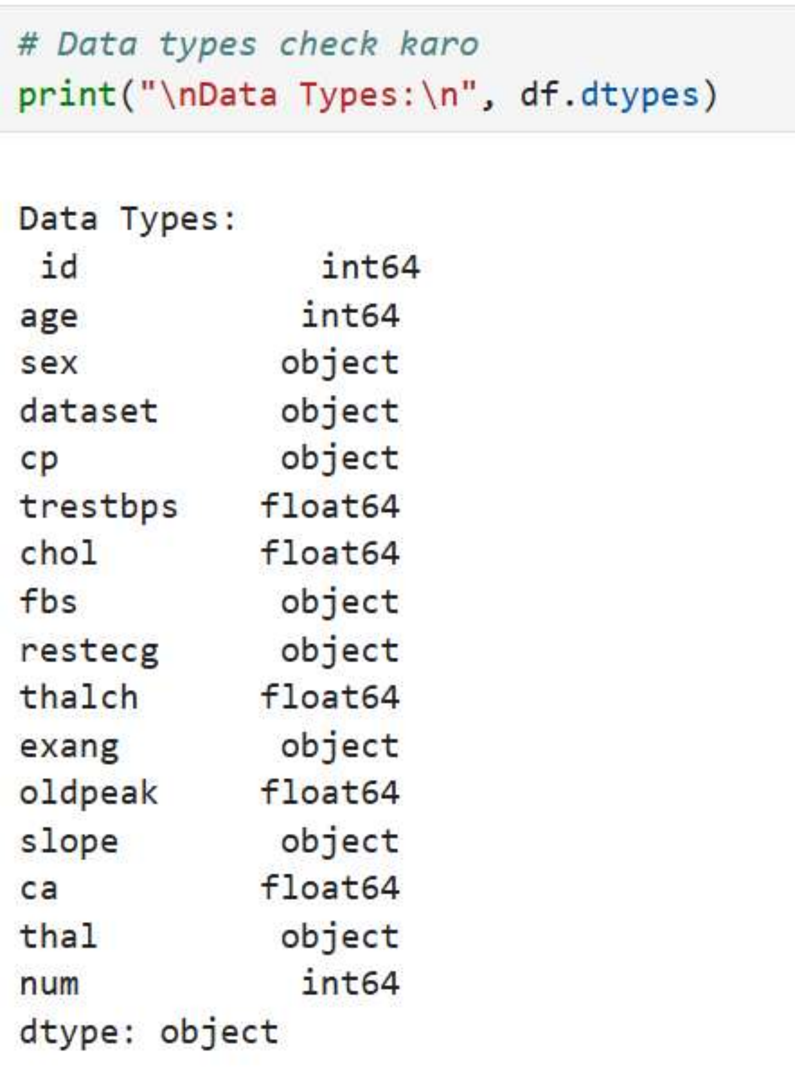
```python
# 3. Data Cleaning 🧹
# Null values check karo
print("Missing values in each column:\n", df.isnull().sum())
```

```
Missing values in each column:
 id            0
age           0
sex           0
dataset       0
cp            0
trestbps     59
chol         30
fbs          90
restecg       2
thalch       55
exang        55
oldpeak      62
slope       309
ca          611
thal        486
num           0
dtype: int64
```



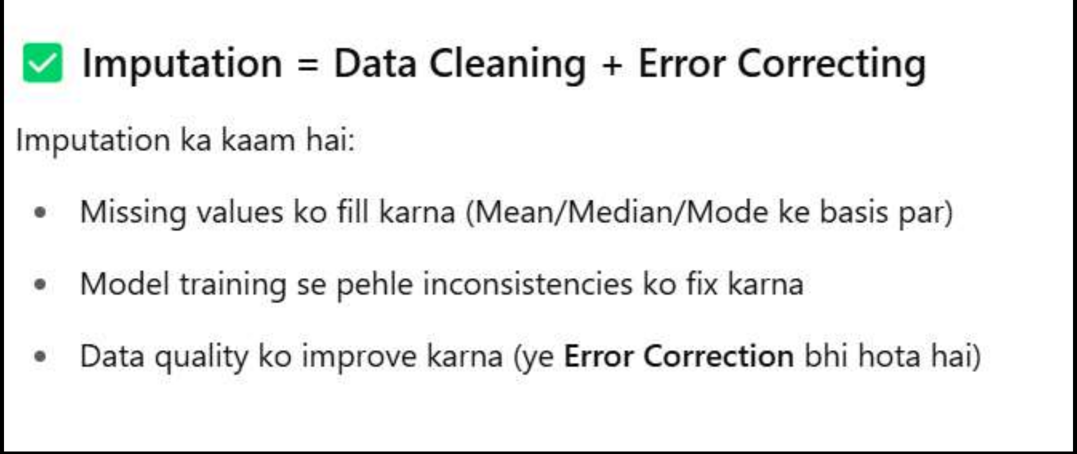
```
# Duplicate rows ko remove karo
df.drop_duplicates(inplace=True)
```

```
df
```



```python
# Data types check karo
print("\nData Types:\n", df.dtypes)
```

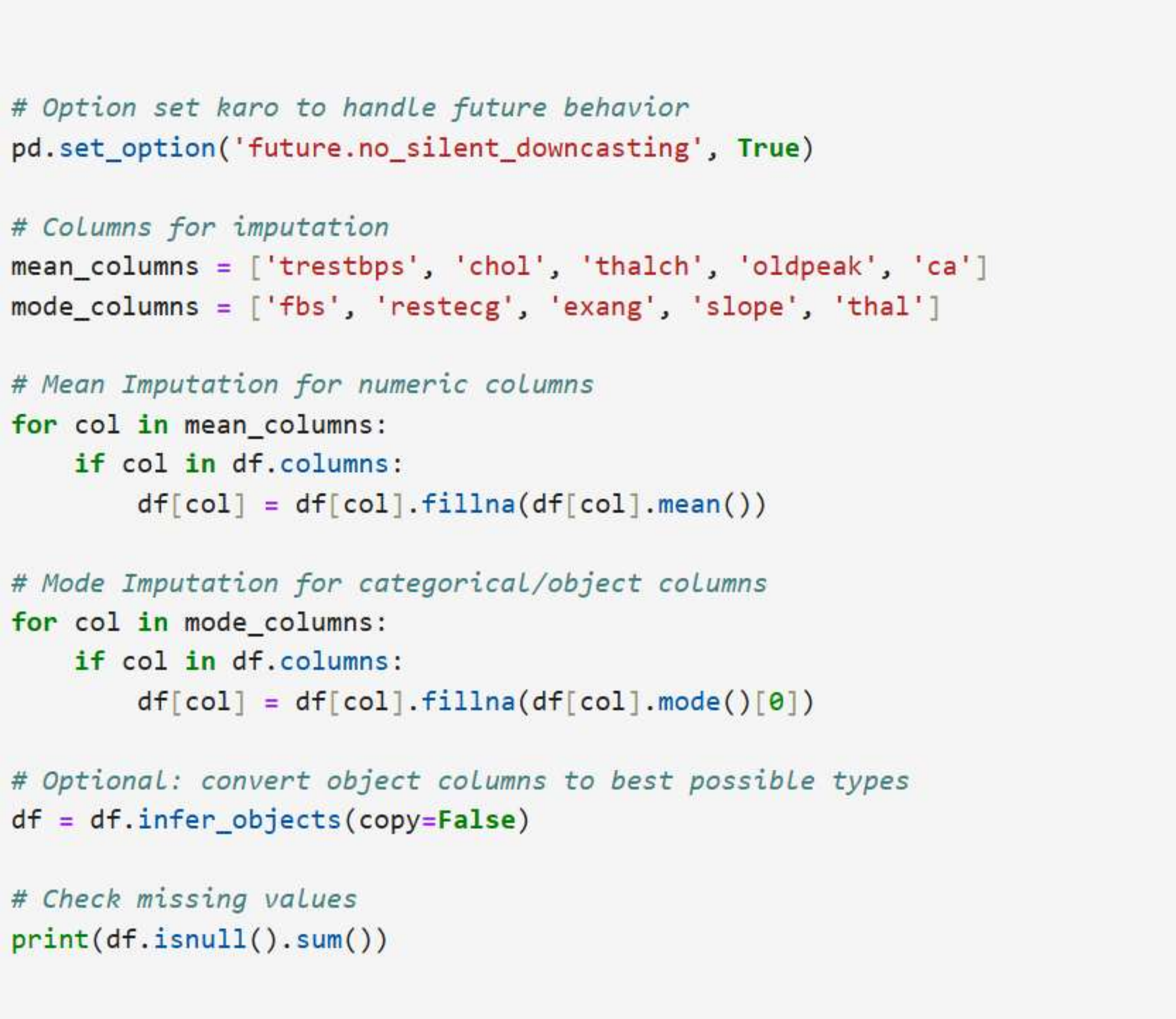
```
Data Types:
 id             int64
age            int64
sex           object
dataset       object
cp            object
trestbps     float64
chol         float64
fbs           object
restecg       object
thalch       float64
exang         object
oldpeak      float64
slope         object
ca           float64
thal          object
num            int64
dtype: object
```



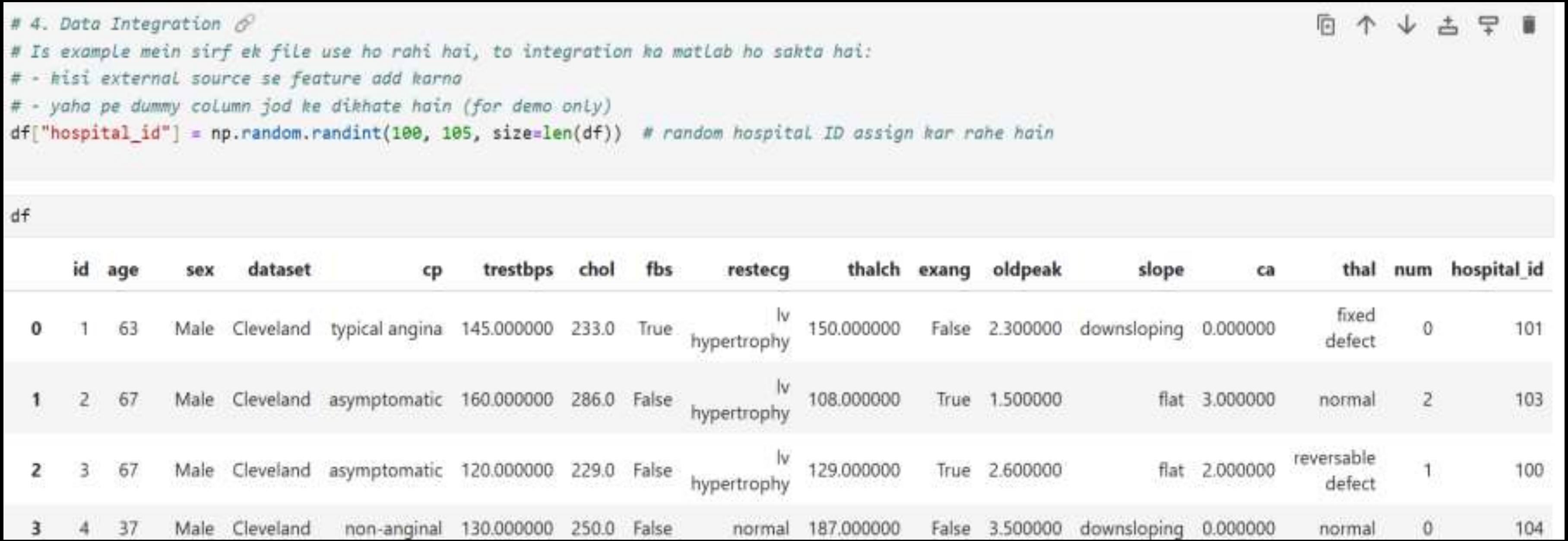
✅ Imputation = Data Cleaning + Error Correcting

Imputation ka kaam hai:

- Missing values ko fill karna (Mean/Median/Mode ke basis par)
- Model training se pehle inconsistencies ko fix karna
- Data quality ko improve karna (ye **Error Correction** bhi hota hai)



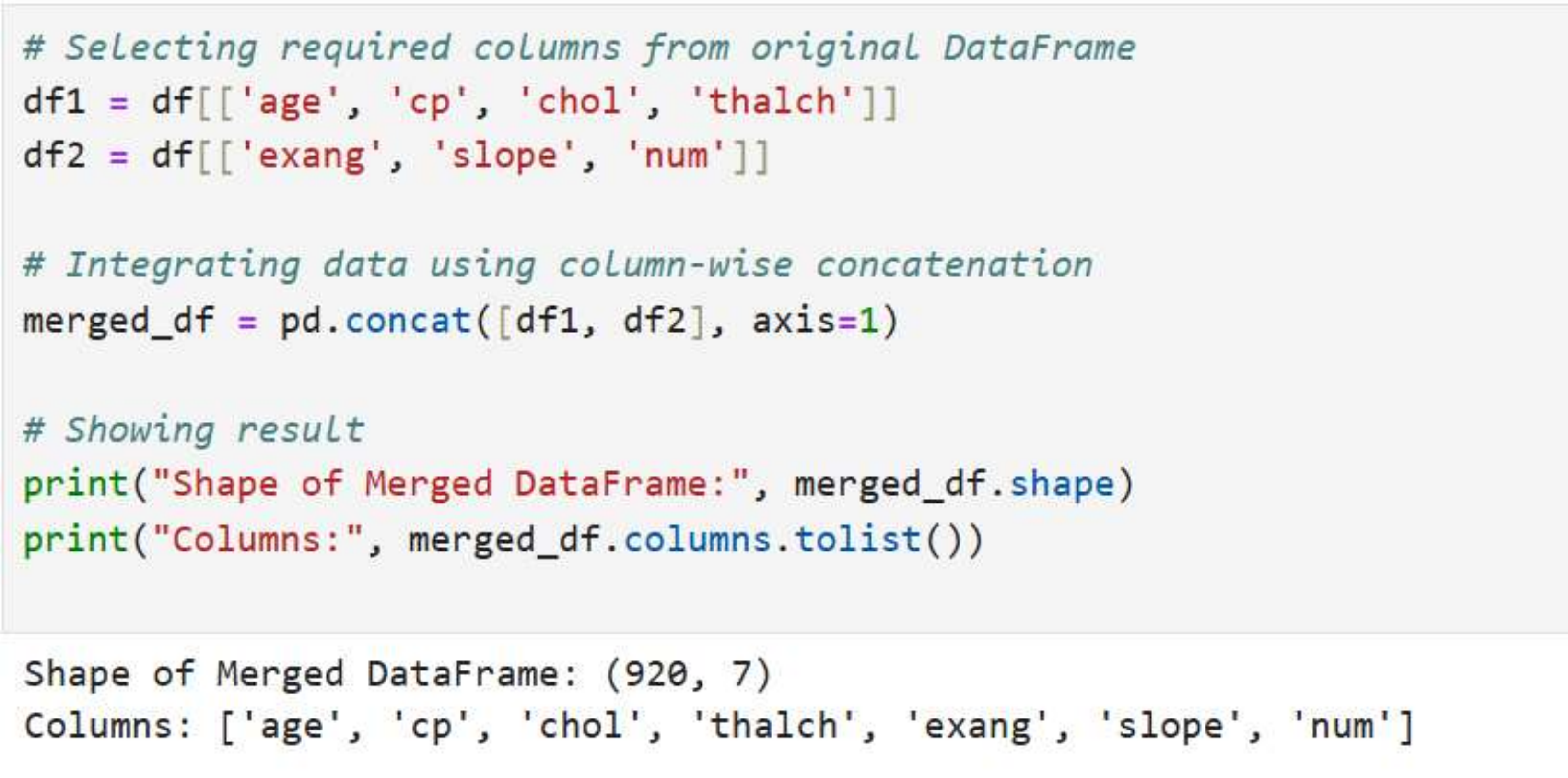
```python
# Option set karo to handle future behavior
pd.set_option('future.no_silent_downcasting', True)

# Columns for imputation
mean_columns = ['trestbps', 'chol', 'thalch', 'oldpeak', 'ca']
mode_columns = ['fbs', 'restecg', 'exang', 'slope', 'thal']

# Mean Imputation for numeric columns
for col in mean_columns:
    if col in df.columns:
        df[col] = df[col].fillna(df[col].mean())

# Mode Imputation for categorical/object columns
for col in mode_columns:
    if col in df.columns:
        df[col] = df[col].fillna(df[col].mode()[0])

# Optional: convert object columns to best possible types
df = df.infer_objects(copy=False)

# Check missing values
print(df.isnull().sum())
```



```python
# 4. Data Integration
# Is example mein sirf ek file use ho rahi hai, to integration ka matlab ho sakta hai:
# - kisi external source se feature add karna
# - yaha pe dummy column jod ke dikhate hain (for demo only)
df["hospital_id"] = np.random.randint(100, 105, size=len(df))  # random hospital ID assign kar rahe hain
```

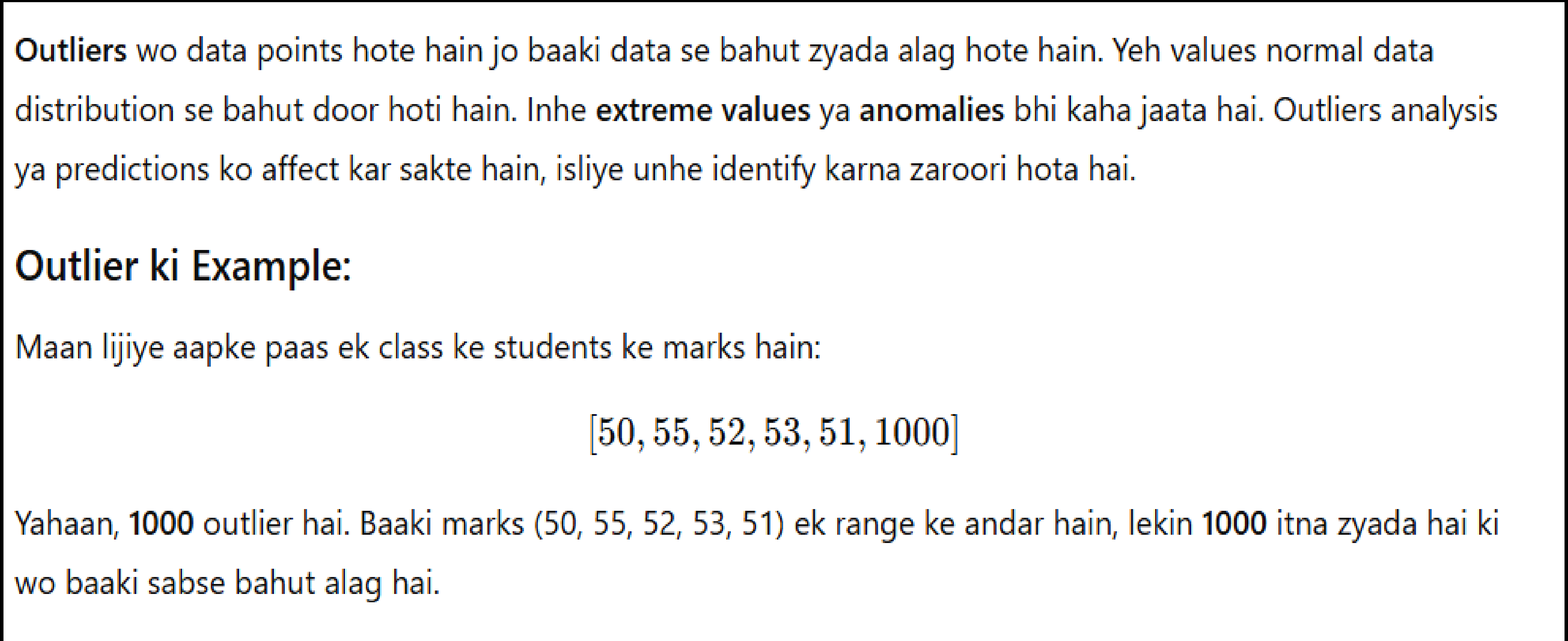
```python
df
```

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num	hospital_id
0	1	63	Male	Cleveland	typical angina	145.000000	233.0	True	lv hypertrophy	150.000000	False	2.300000	downsloping	0.000000	fixed defect	0	101
1	2	67	Male	Cleveland	asymptomatic	160.000000	286.0	False	lv hypertrophy	108.000000	True	1.500000	flat	3.000000	normal	2	103
2	3	67	Male	Cleveland	asymptomatic	120.000000	229.0	False	lv hypertrophy	129.000000	True	2.600000	flat	2.000000	reversable defect	1	100
3	4	37	Male	Cleveland	non-anginal	130.000000	250.0	False	normal	187.000000	False	3.500000	downsloping	0.000000	normal	0	104



```
# Selecting required columns from original DataFrame
df1 = df[['age', 'cp', 'chol', 'thalch']]
df2 = df[['exang', 'slope', 'num']]

# Integrating data using column-wise concatenation
merged_df = pd.concat([df1, df2], axis=1)

# Showing result
print("Shape of Merged DataFrame:", merged_df.shape)
print("Columns:", merged_df.columns.tolist())
```

```
Shape of Merged DataFrame: (920, 7)
Columns: ['age', 'cp', 'chol', 'thalch', 'exang', 'slope', 'num']
```



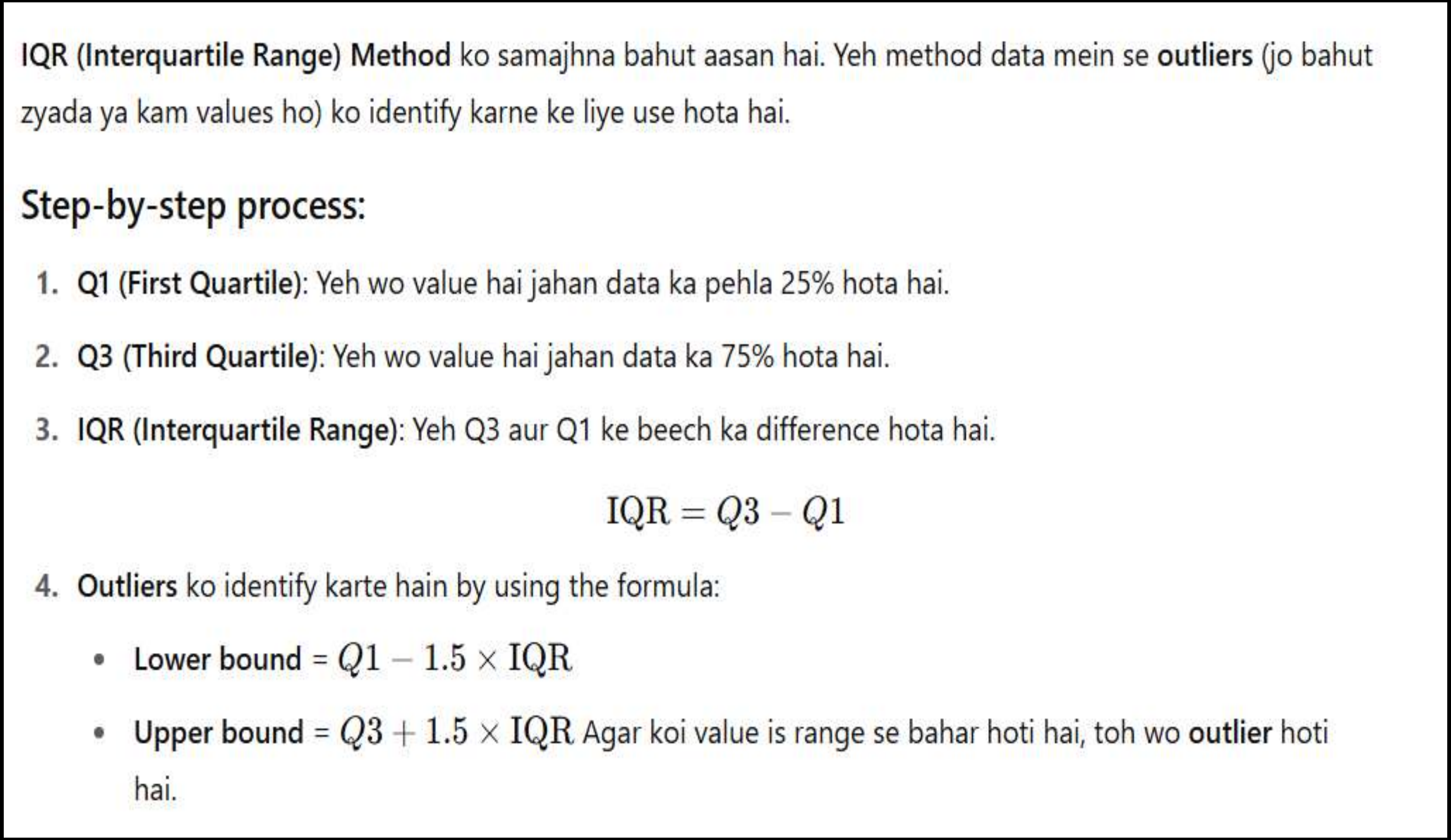
Outliers wo data points hote hain jo baaki data se bahut zyada alag hote hain. Yeh values normal data distribution se bahut door hoti hain. Inhe **extreme values** ya **anomalies** bhi kaha jaata hai. Outliers analysis ya predictions ko affect kar sakte hain, isliye unhe identify karna zaroori hota hai.

Outlier ki Example:

Maan lijiye aapke paas ek class ke students ke marks hain:

$$[50, 55, 52, 53, 51, 1000]$$

Yahaan, **1000** outlier hai. Baaki marks (50, 55, 52, 53, 51) ek range ke andar hain, lekin **1000** itna zyada hai ki wo baaki sabse bahut alag hai.



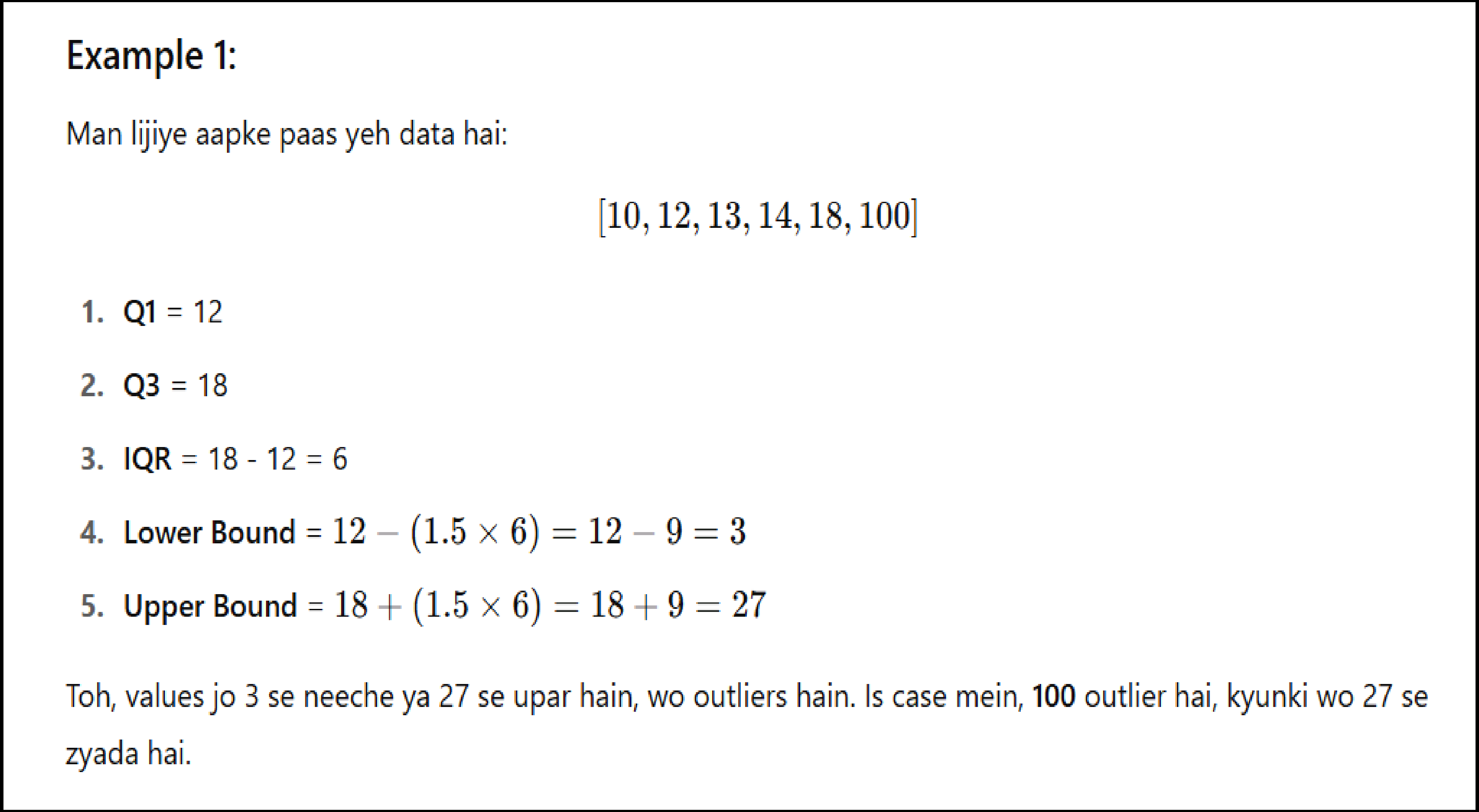
IQR (Interquartile Range) Method ko samajhna bahut aasan hai. Yeh method data mein se **outliers** (jo bahut zyada ya kam values ho) ko identify karne ke liye use hota hai.

Step-by-step process:

1. **Q1 (First Quartile)**: Yeh wo value hai jahan data ka pehla 25% hota hai.
2. **Q3 (Third Quartile)**: Yeh wo value hai jahan data ka 75% hota hai.
3. **IQR (Interquartile Range)**: Yeh Q3 aur Q1 ke beech ka difference hota hai.

$$\text{IQR} = Q3 - Q1$$

4. **Outliers** ko identify karte hain by using the formula:
   - **Lower bound** = $Q1 - 1.5 \times \text{IQR}$
   - **Upper bound** = $Q3 + 1.5 \times \text{IQR}$ Agar koi value is range se bahar hoti hai, toh wo **outlier** hoti hai.



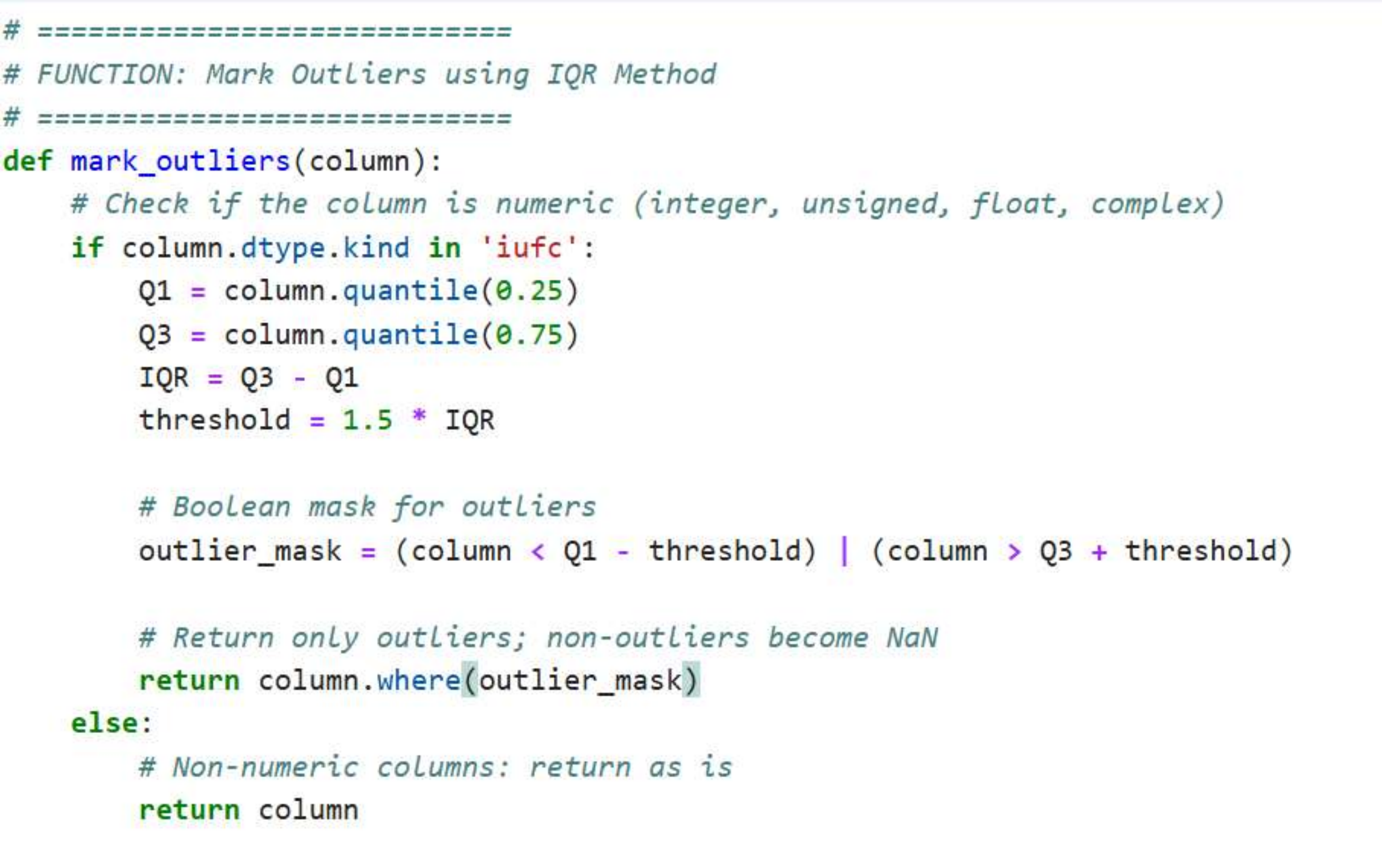
Example 1:

Man lijiye aapke paas yeh data hai:

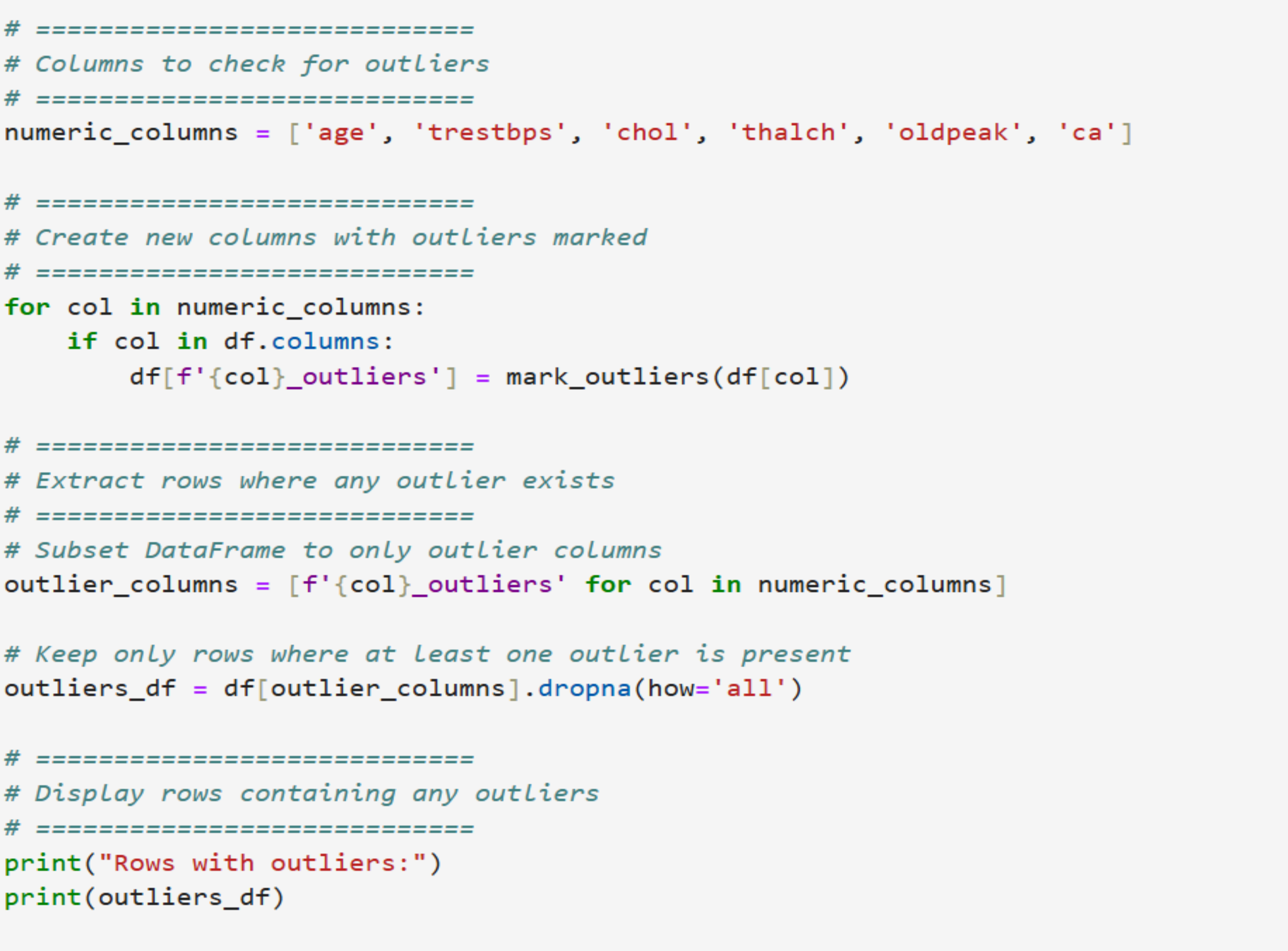
$$[10, 12, 13, 14, 18, 100]$$

1. **Q1** = 12
2. **Q3** = 18
3. **IQR** = 18 - 12 = 6
4. **Lower Bound** = $12 - (1.5 \times 6) = 12 - 9 = 3$
5. **Upper Bound** = $18 + (1.5 \times 6) = 18 + 9 = 27$

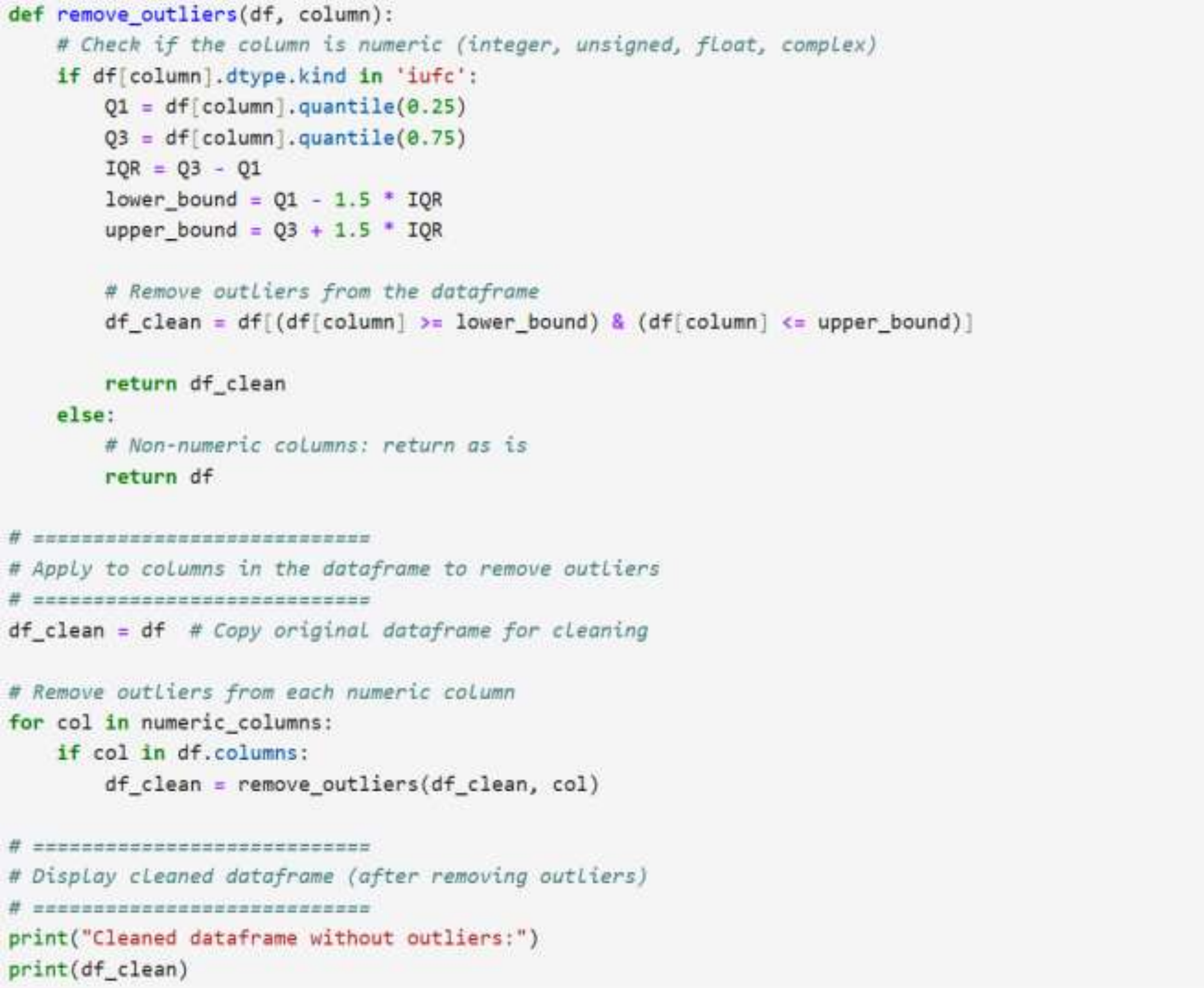
Toh, values jo 3 se neeche ya 27 se upar hain, wo outliers hain. Is case mein, **100** outlier hai, kyunki wo 27 se zyada hai.



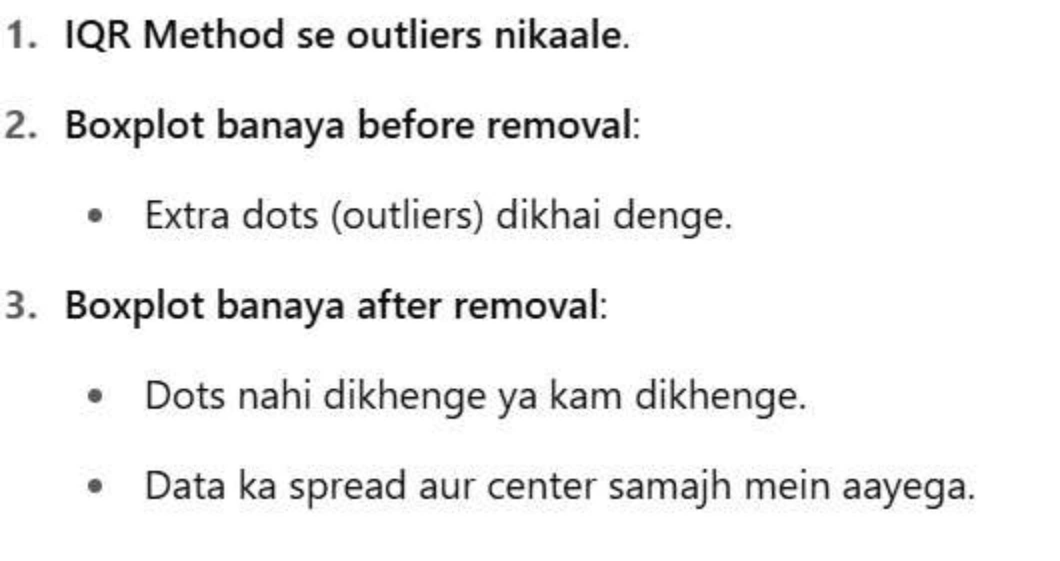
```python
# ============================
# FUNCTION: Mark Outliers using IQR Method
# ============================
def mark_outliers(column):
    # Check if the column is numeric (integer, unsigned, float, complex)
    if column.dtype.kind in 'iufc':
        Q1 = column.quantile(0.25)
        Q3 = column.quantile(0.75)
        IQR = Q3 - Q1
        threshold = 1.5 * IQR

        # Boolean mask for outliers
        outlier_mask = (column < Q1 - threshold) | (column > Q3 + threshold)

        # Return only outliers; non-outliers become NaN
        return column.where(outlier_mask)
    else:
        # Non-numeric columns: return as is
        return column
```



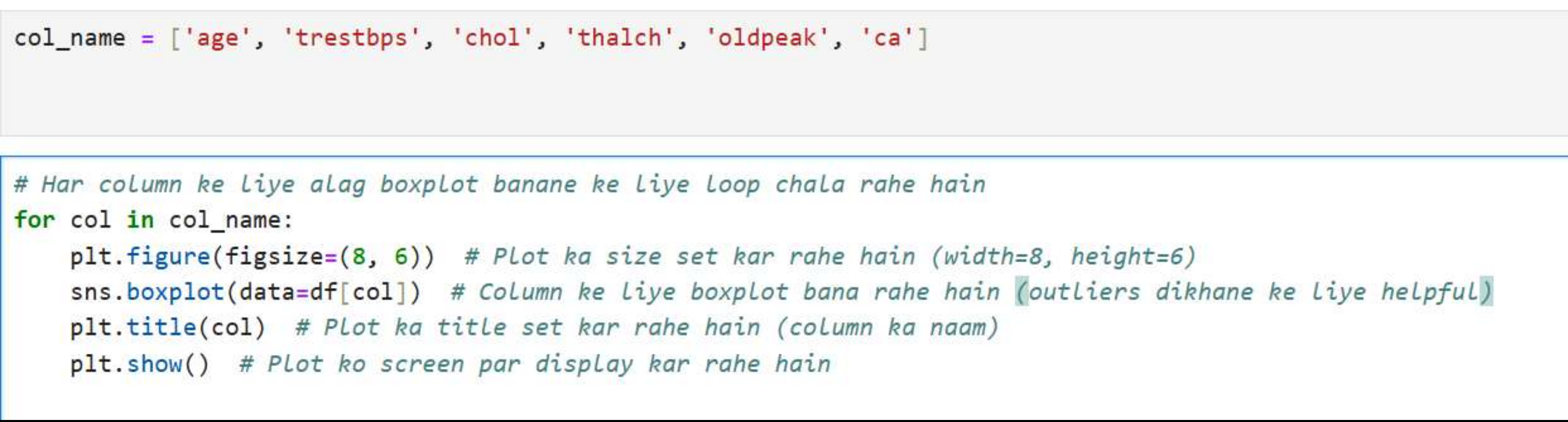
```
# ============================
# Columns to check for outliers
# ============================
numeric_columns = ['age', 'trestbps', 'chol', 'thalch', 'oldpeak', 'ca']

# ============================
# Create new columns with outliers marked
# ============================
for col in numeric_columns:
    if col in df.columns:
        df[f'{col}_outliers'] = mark_outliers(df[col])

# ============================
# Extract rows where any outlier exists
# ============================
# Subset DataFrame to only outlier columns
outlier_columns = [f'{col}_outliers' for col in numeric_columns]

# Keep only rows where at least one outlier is present
outliers_df = df[outlier_columns].dropna(how='all')

# ============================
# Display rows containing any outliers
# ============================
print("Rows with outliers:")
print(outliers_df)
```



```python
def remove_outliers(df, column):
    # Check if the column is numeric (integer, unsigned, float, complex)
    if df[column].dtype.kind in 'iufc':
        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

        # Remove outliers from the dataframe
        df_clean = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

        return df_clean
    else:
        # Non-numeric columns: return as is
        return df

# ============================
# Apply to columns in the dataframe to remove outliers
# ============================
df_clean = df  # Copy original dataframe for cleaning

# Remove outliers from each numeric column
for col in numeric_columns:
    if col in df.columns:
        df_clean = remove_outliers(df_clean, col)

# ============================
# Display cleaned dataframe (after removing outliers)
# ============================
print("Cleaned dataframe without outliers:")
print(df_clean)
```

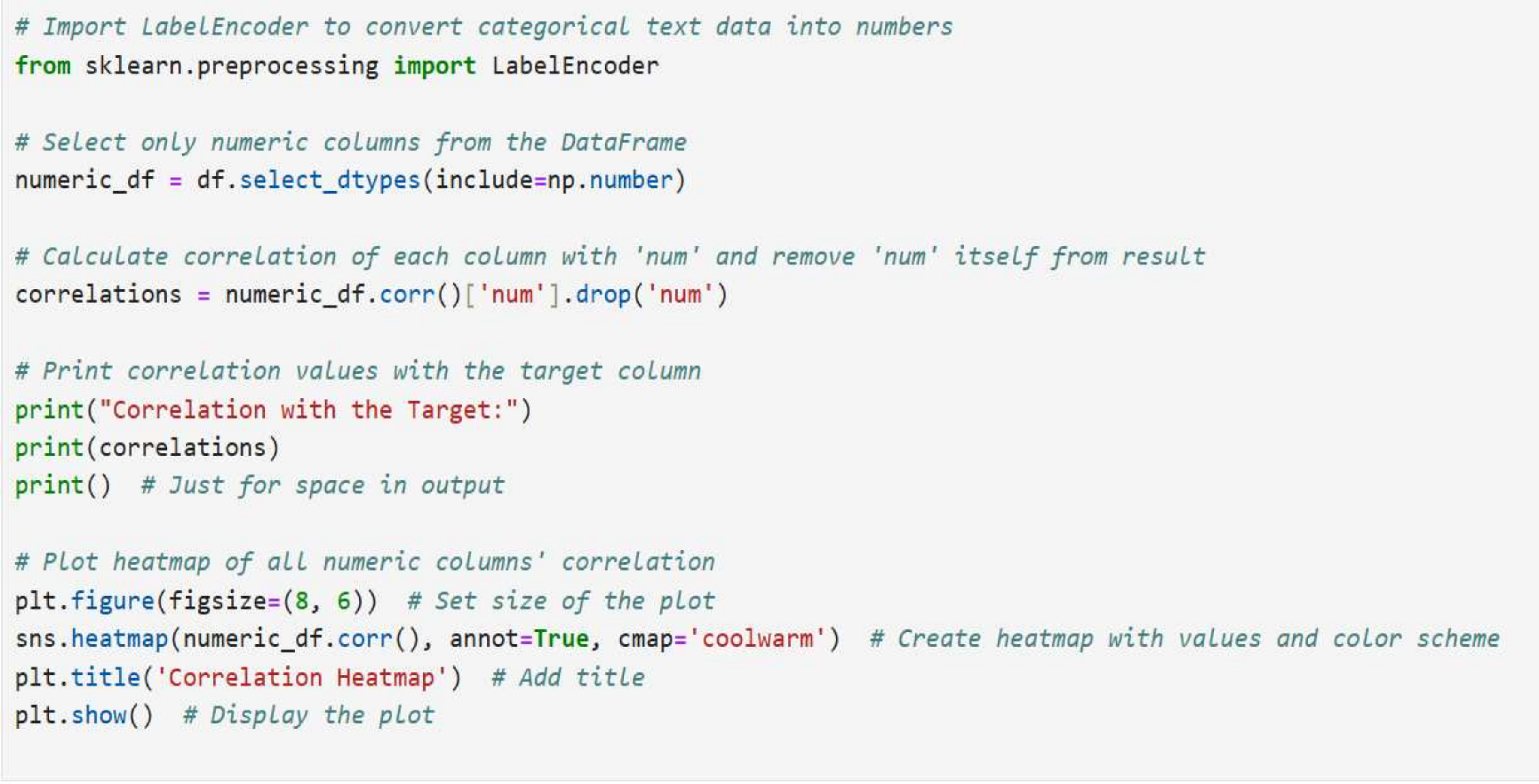


1. **IQR Method se outliers nikaale**.
2. **Boxplot banaya before removal**:
   - Extra dots (outliers) dikhai denge.
3. **Boxplot banaya after removal**:
   - Dots nahi dikhenge ya kam dikhenge.
   - Data ka spread aur center samajh mein aayega.



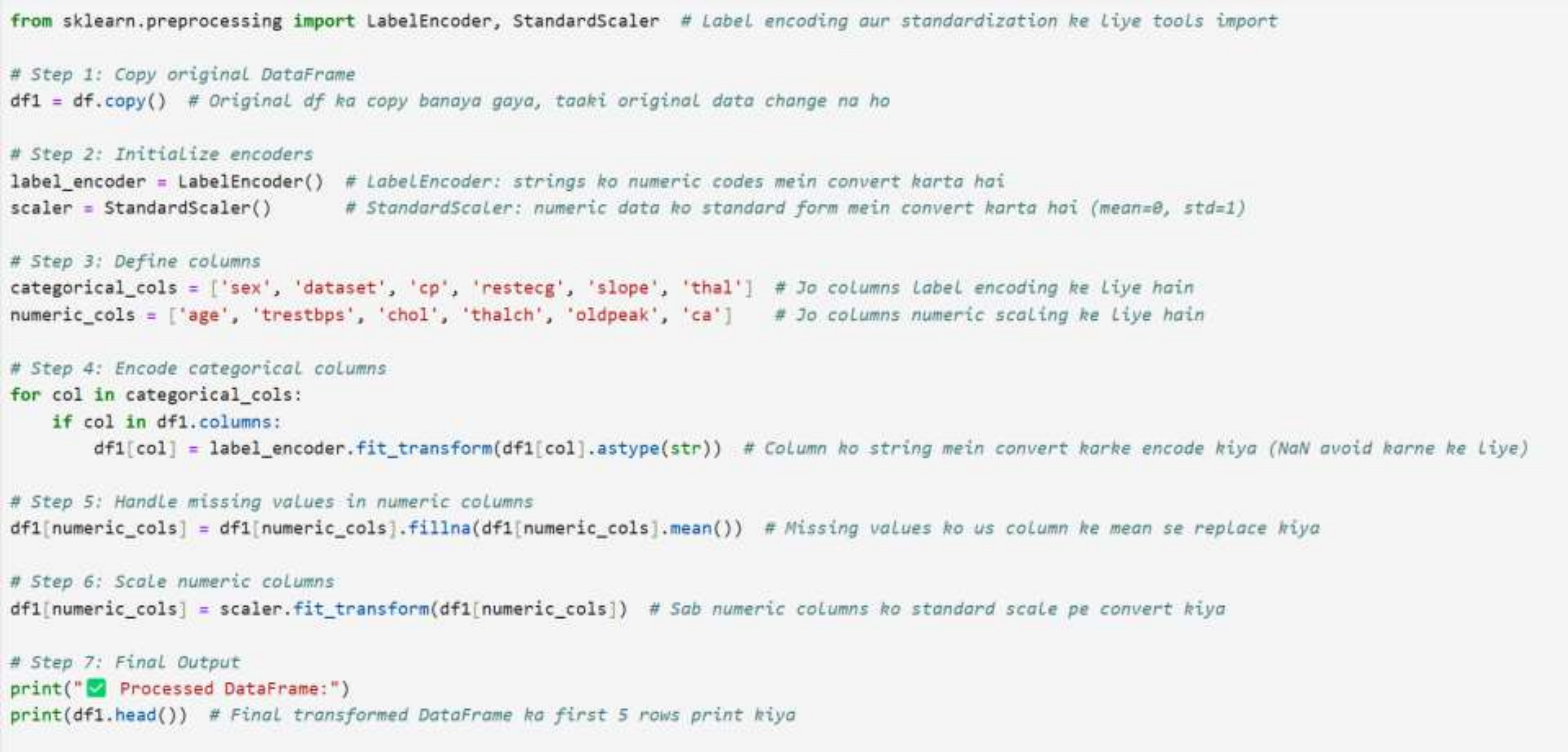
```python
col_name = ['age', 'trestbps', 'chol', 'thalch', 'oldpeak', 'ca']
```

```python
# Har column ke liye alag boxplot banane ke liye loop chala rahe hain
for col in col_name:
    plt.figure(figsize=(8, 6))  # Plot ka size set kar rahe hain (width=8, height=6)
    sns.boxplot(data=df[col])  # Column ke liye boxplot bana rahe hain (outliers dikhane ke liye helpful)
    plt.title(col)  # Plot ka title set kar rahe hain (column ka naam)
    plt.show()  # Plot ko screen par display kar rahe hain
```



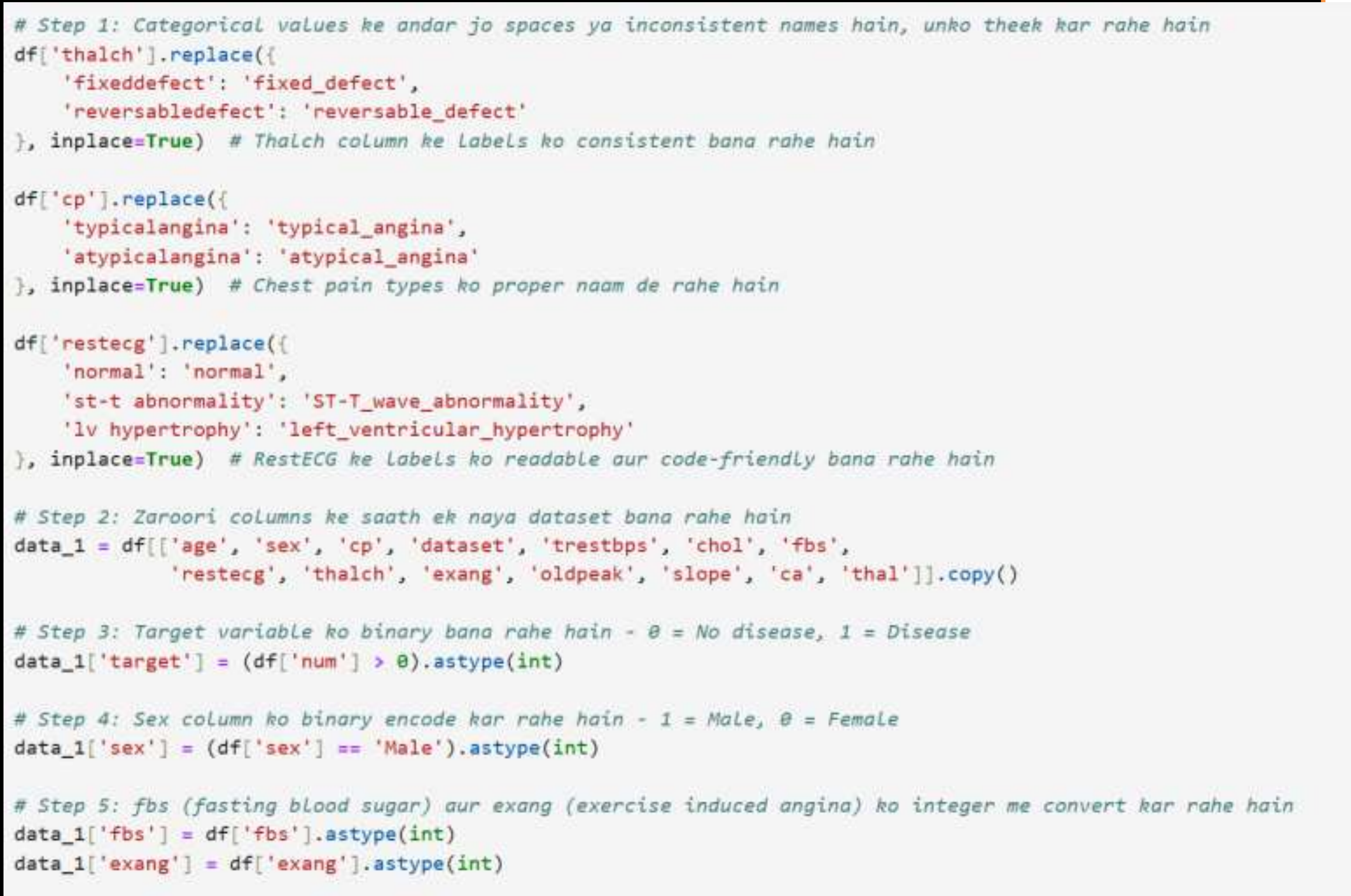
```python
# Import LabelEncoder to convert categorical text data into numbers
from sklearn.preprocessing import LabelEncoder

# Select only numeric columns from the DataFrame
numeric_df = df.select_dtypes(include=np.number)

# Calculate correlation of each column with 'num' and remove 'num' itself from result
correlations = numeric_df.corr()['num'].drop('num')

# Print correlation values with the target column
print("Correlation with the Target:")
print(correlations)
print()  # Just for space in output

# Plot heatmap of all numeric columns' correlation
plt.figure(figsize=(8, 6))  # Set size of the plot
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm')  # Create heatmap with values and color scheme
plt.title('Correlation Heatmap')  # Add title
plt.show()  # Display the plot
```

Data transformation

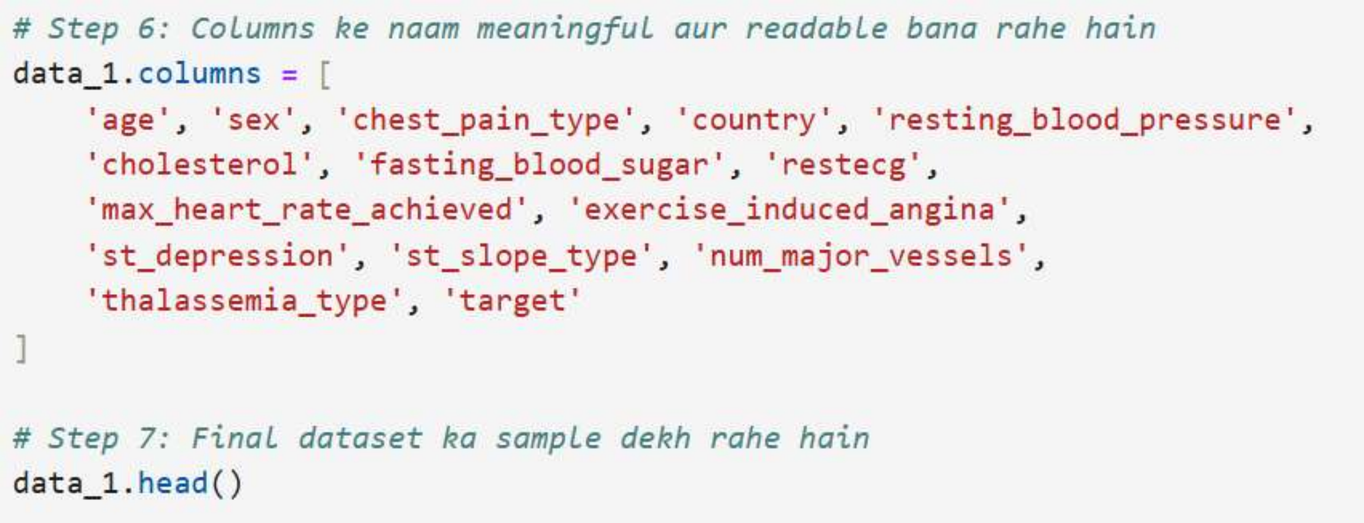


```python
from sklearn.preprocessing import LabelEncoder, StandardScaler  # Label encoding aur standardization ke liye tools import

# Step 1: Copy original DataFrame
df1 = df.copy()  # Original df ka copy banaya gaya, taaki original data change na ho

# Step 2: Initialize encoders
label_encoder = LabelEncoder()  # LabelEncoder: strings ko numeric codes mein convert karta hai
scaler = StandardScaler()       # StandardScaler: numeric data ko standard form mein convert karta hai (mean=0, std=1)

# Step 3: Define columns
categorical_cols = ['sex', 'dataset', 'cp', 'restecg', 'slope', 'thal']  # Jo columns label encoding ke liye hain
numeric_cols = ['age', 'trestbps', 'chol', 'thalch', 'oldpeak', 'ca']    # Jo columns numeric scaling ke liye hain

# Step 4: Encode categorical columns
for col in categorical_cols:
    if col in df1.columns:
        df1[col] = label_encoder.fit_transform(df1[col].astype(str))  # Column ko string mein convert karke encode kiya (NaN avoid karne ke liye)

# Step 5: Handle missing values in numeric columns
df1[numeric_cols] = df1[numeric_cols].fillna(df1[numeric_cols].mean())  # Missing values ko us column ke mean se replace kiya

# Step 6: Scale numeric columns
df1[numeric_cols] = scaler.fit_transform(df1[numeric_cols])  # Sab numeric columns ko standard scale pe convert kiya

# Step 7: Final Output
print("✅ Processed DataFrame:")
print(df1.head())  # Final transformed DataFrame ka first 5 rows print kiya
```

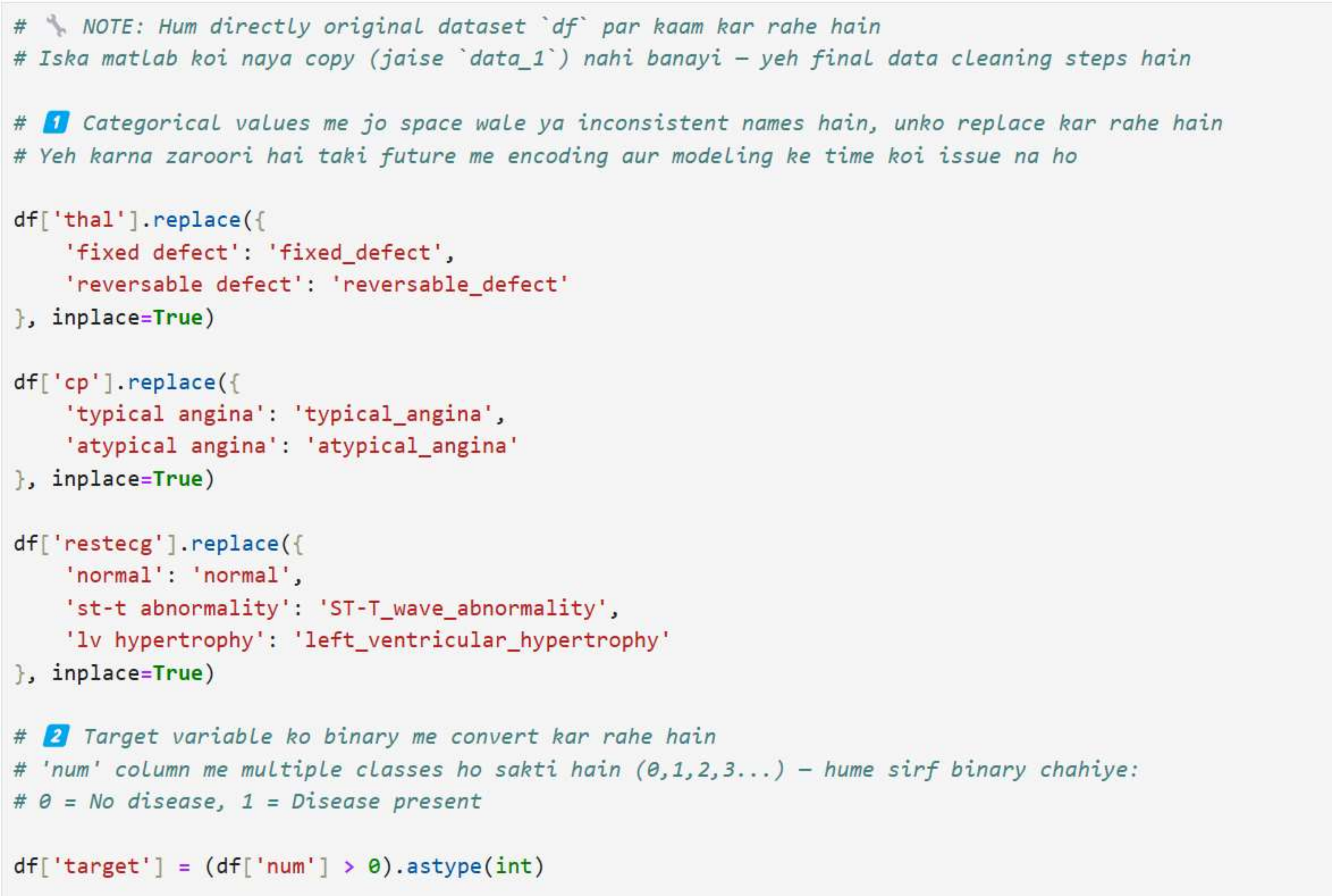
Model Building



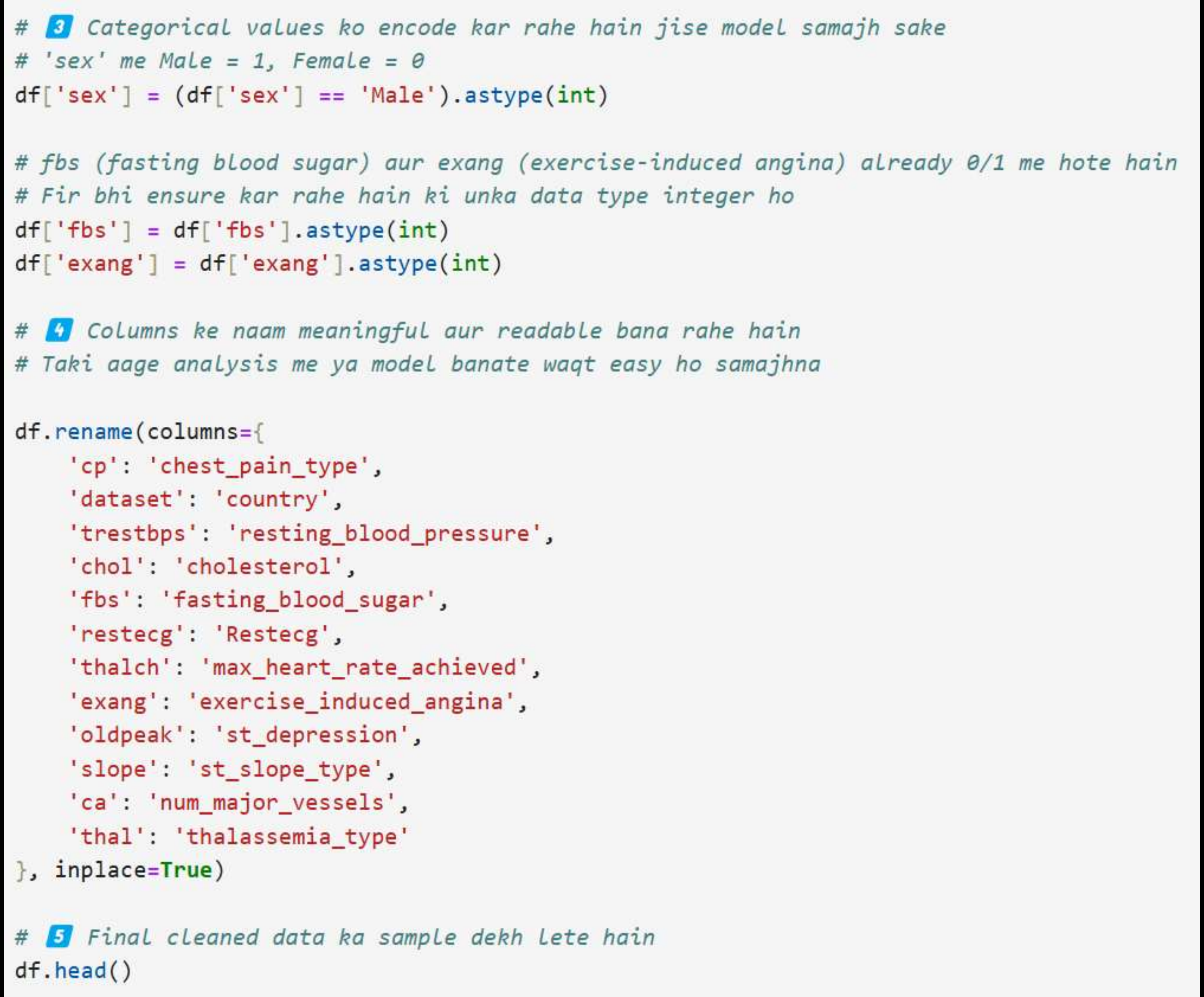
```python
# Step 1: Categorical values ke andar jo spaces ya inconsistent names hain, unko theek kar rahe hain
df['thalch'].replace({
    'fixeddefect': 'fixed_defect',
    'reversabledefect': 'reversable_defect'
}, inplace=True)  # Thalch column ke labels ko consistent bana rahe hain

df['cp'].replace({
    'typicalangina': 'typical_angina',
    'atypicalangina': 'atypical_angina'
}, inplace=True)  # Chest pain types ko proper naam de rahe hain

df['restecg'].replace({
    'normal': 'normal',
    'st-t abnormality': 'ST-T_wave_abnormality',
    'lv hypertrophy': 'left_ventricular_hypertrophy'
}, inplace=True)  # RestECG ke labels ko readable aur code-friendly bana rahe hain

# Step 2: Zaroori columns ke saath ek naya dataset bana rahe hain
data_1 = df[['age', 'sex', 'cp', 'dataset', 'trestbps', 'chol', 'fbs',
             'restecg', 'thalch', 'exang', 'oldpeak', 'slope', 'ca', 'thal']].copy()

# Step 3: Target variable ko binary bana rahe hain - 0 = No disease, 1 = Disease
data_1['target'] = (df['num'] > 0).astype(int)

# Step 4: Sex column ko binary encode kar rahe hain - 1 = Male, 0 = Female
data_1['sex'] = (df['sex'] == 'Male').astype(int)

# Step 5: fbs (fasting blood sugar) aur exang (exercise induced angina) ko integer me convert kar rahe hain
data_1['fbs'] = df['fbs'].astype(int)
data_1['exang'] = df['exang'].astype(int)
```



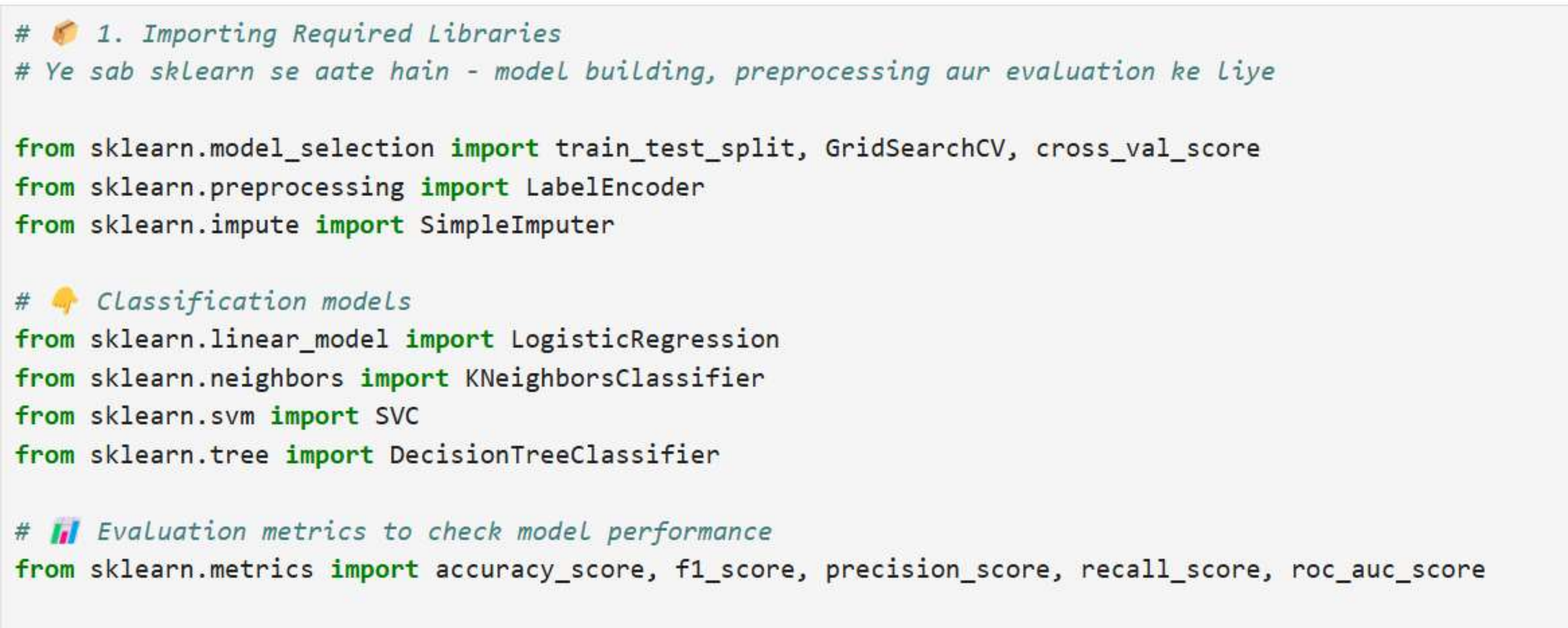
```
# Step 6: Columns ke naam meaningful aur readable bana rahe hain
data_1.columns = [
    'age', 'sex', 'chest_pain_type', 'country', 'resting_blood_pressure',
    'cholesterol', 'fasting_blood_sugar', 'restecg',
    'max_heart_rate_achieved', 'exercise_induced_angina',
    'st_depression', 'st_slope_type', 'num_major_vessels',
    'thalassemia_type', 'target'
]

# Step 7: Final dataset ka sample dekh rahe hain
data_1.head()
```



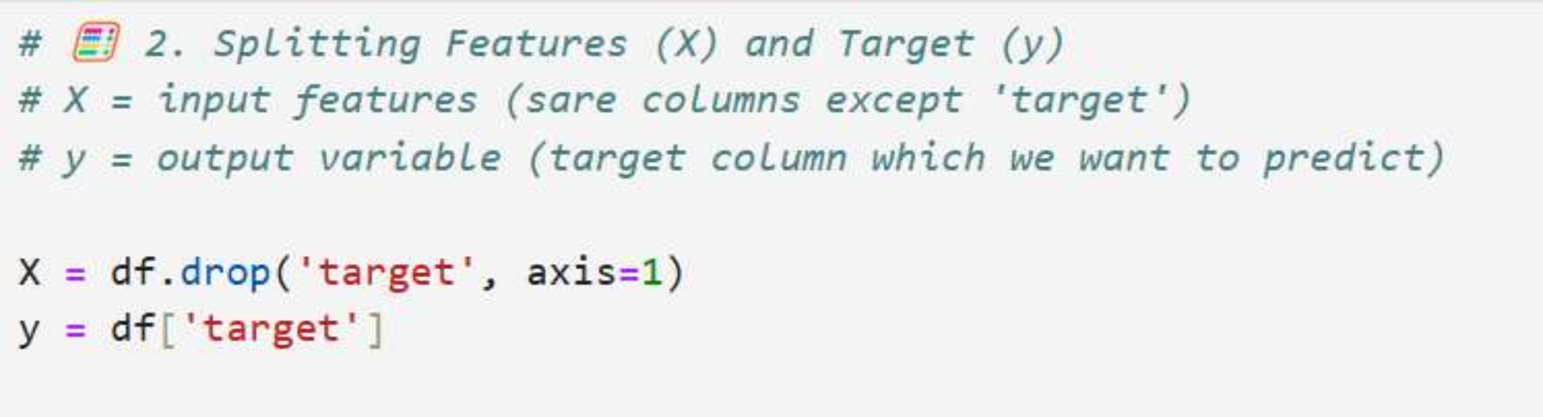
```python
# 🔧 NOTE: Hum directly original dataset `df` par kaam kar rahe hain
# Iska matlab koi naya copy (jaise `data_1`) nahi banayi – yeh final data cleaning steps hain

# 1️⃣ Categorical values me jo space wale ya inconsistent names hain, unko replace kar rahe hain
# Yeh karna zaroori hai taki future me encoding aur modeling ke time koi issue na ho

df['thal'].replace({
    'fixed defect': 'fixed_defect',
    'reversable defect': 'reversable_defect'
}, inplace=True)

df['cp'].replace({
    'typical angina': 'typical_angina',
    'atypical angina': 'atypical_angina'
}, inplace=True)

df['restecg'].replace({
    'normal': 'normal',
    'st-t abnormality': 'ST-T_wave_abnormality',
    'lv hypertrophy': 'left_ventricular_hypertrophy'
}, inplace=True)

# 2️⃣ Target variable ko binary me convert kar rahe hain
# 'num' column me multiple classes ho sakti hain (0,1,2,3...) – hume sirf binary chahiye:
# 0 = No disease, 1 = Disease present

df['target'] = (df['num'] > 0).astype(int)
```



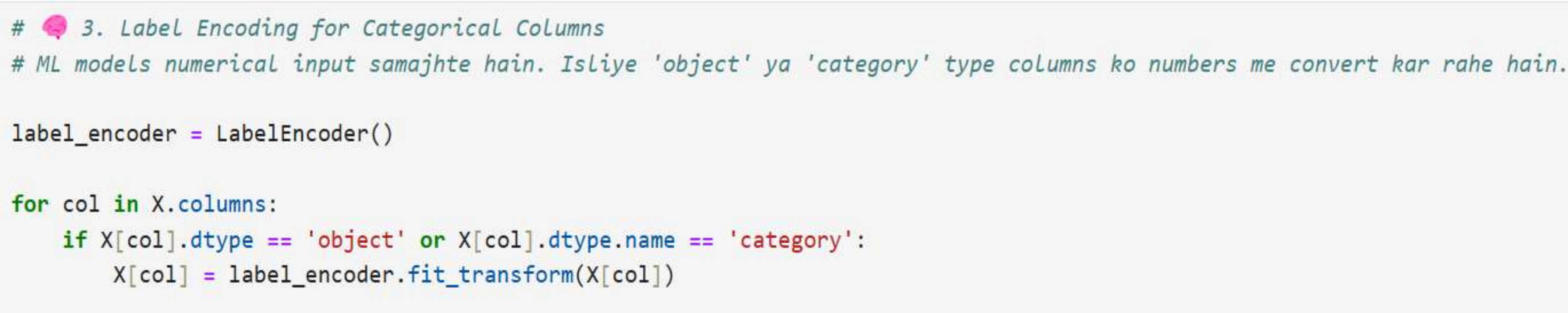
```python
# 3️⃣ Categorical values ko encode kar rahe hain jise model samajh sake
# 'sex' me Male = 1, Female = 0
df['sex'] = (df['sex'] == 'Male').astype(int)

# fbs (fasting blood sugar) aur exang (exercise-induced angina) already 0/1 me hote hain
# Fir bhi ensure kar rahe hain ki unka data type integer ho
df['fbs'] = df['fbs'].astype(int)
df['exang'] = df['exang'].astype(int)

# 4️⃣ Columns ke naam meaningful aur readable bana rahe hain
# Taki aage analysis me ya model banate waqt easy ho samajhna

df.rename(columns={
    'cp': 'chest_pain_type',
    'dataset': 'country',
    'trestbps': 'resting_blood_pressure',
    'chol': 'cholesterol',
    'fbs': 'fasting_blood_sugar',
    'restecg': 'Restecg',
    'thalch': 'max_heart_rate_achieved',
    'exang': 'exercise_induced_angina',
    'oldpeak': 'st_depression',
    'slope': 'st_slope_type',
    'ca': 'num_major_vessels',
    'thal': 'thalassemia_type'
}, inplace=True)

# 5️⃣ Final cleaned data ka sample dekh lete hain
df.head()
```



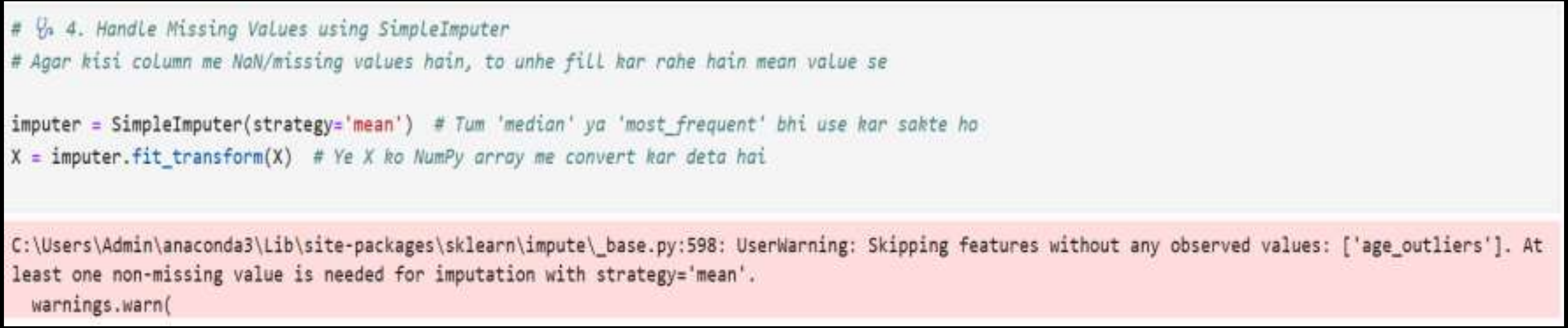
```python
# 📦 1. Importing Required Libraries
# Ye sab sklearn se aate hain - model building, preprocessing aur evaluation ke liye

from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer

# 👇 Classification models
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

# 📊 Evaluation metrics to check model performance
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, roc_auc_score
```



```
# 🧮 2. Splitting Features (X) and Target (y)
# X = input features (sare columns except 'target')
# y = output variable (target column which we want to predict)

X = df.drop('target', axis=1)
y = df['target']
```

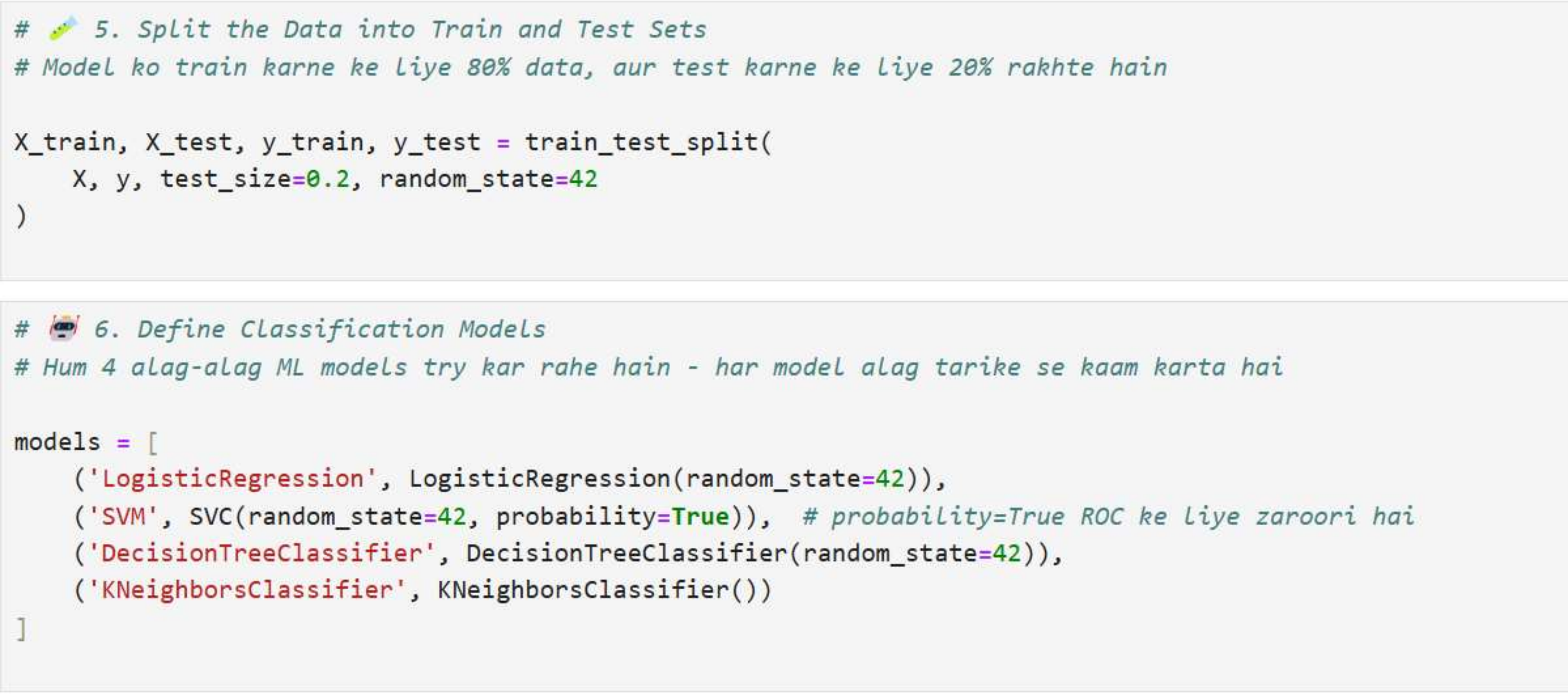


```python
# 🧠 3. Label Encoding for Categorical Columns
# ML models numerical input samajhte hain. Isliye 'object' ya 'category' type columns ko numbers me convert kar rahe hain.

label_encoder = LabelEncoder()

for col in X.columns:
    if X[col].dtype == 'object' or X[col].dtype.name == 'category':
        X[col] = label_encoder.fit_transform(X[col])
```



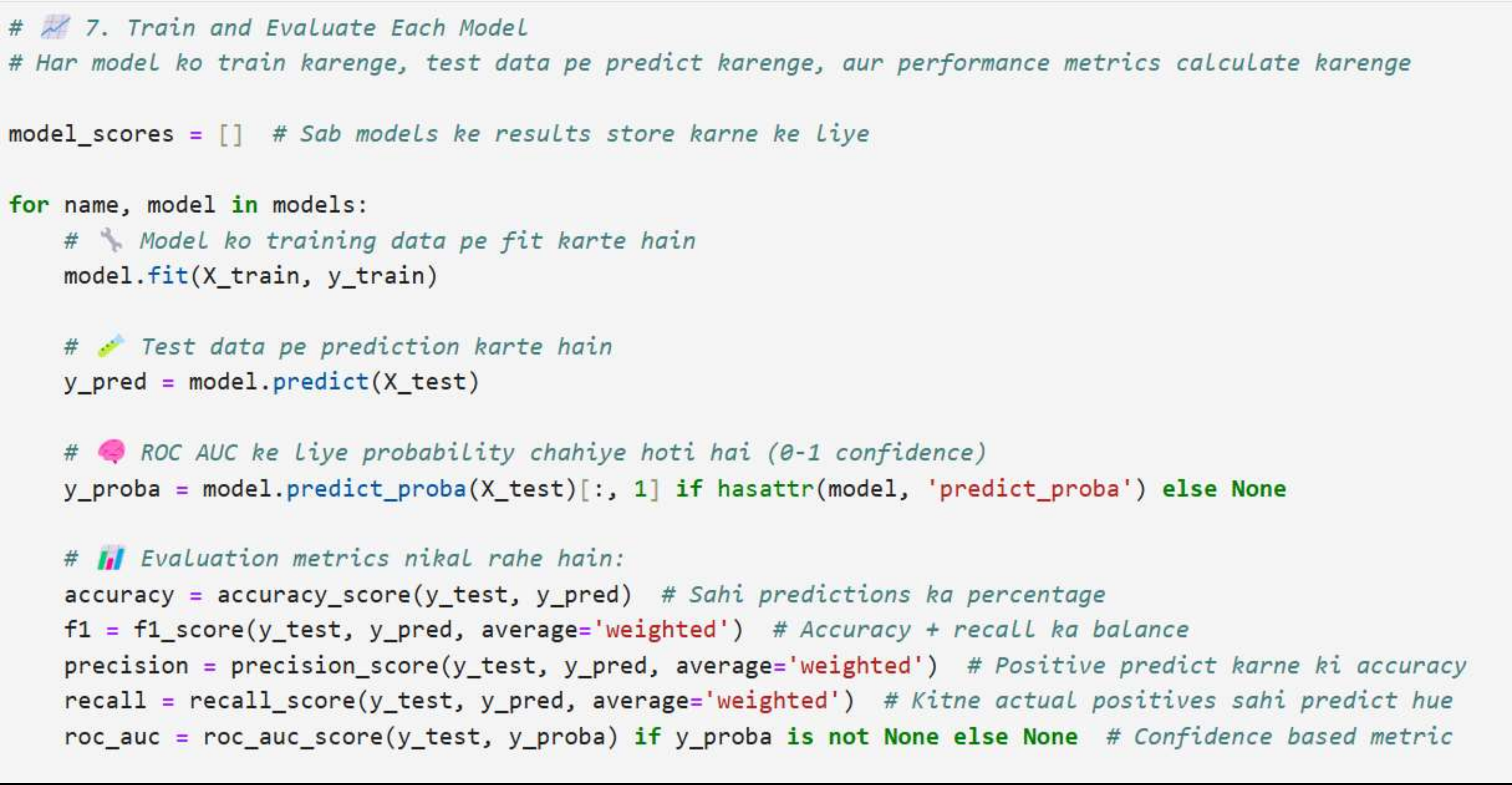
```
# 🩺 4. Handle Missing Values using SimpleImputer
# Agar kisi column me NaN/missing values hain, to unhe fill kar rahe hain mean value se

imputer = SimpleImputer(strategy='mean')  # Tum 'median' ya 'most_frequent' bhi use kar sakte ho
X = imputer.fit_transform(X)  # Ye X ko NumPy array me convert kar deta hai
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\sklearn\impute\_base.py:598: UserWarning: Skipping features without any observed values: ['age_outliers']. At
least one non-missing value is needed for imputation with strategy='mean'.
  warnings.warn(
```

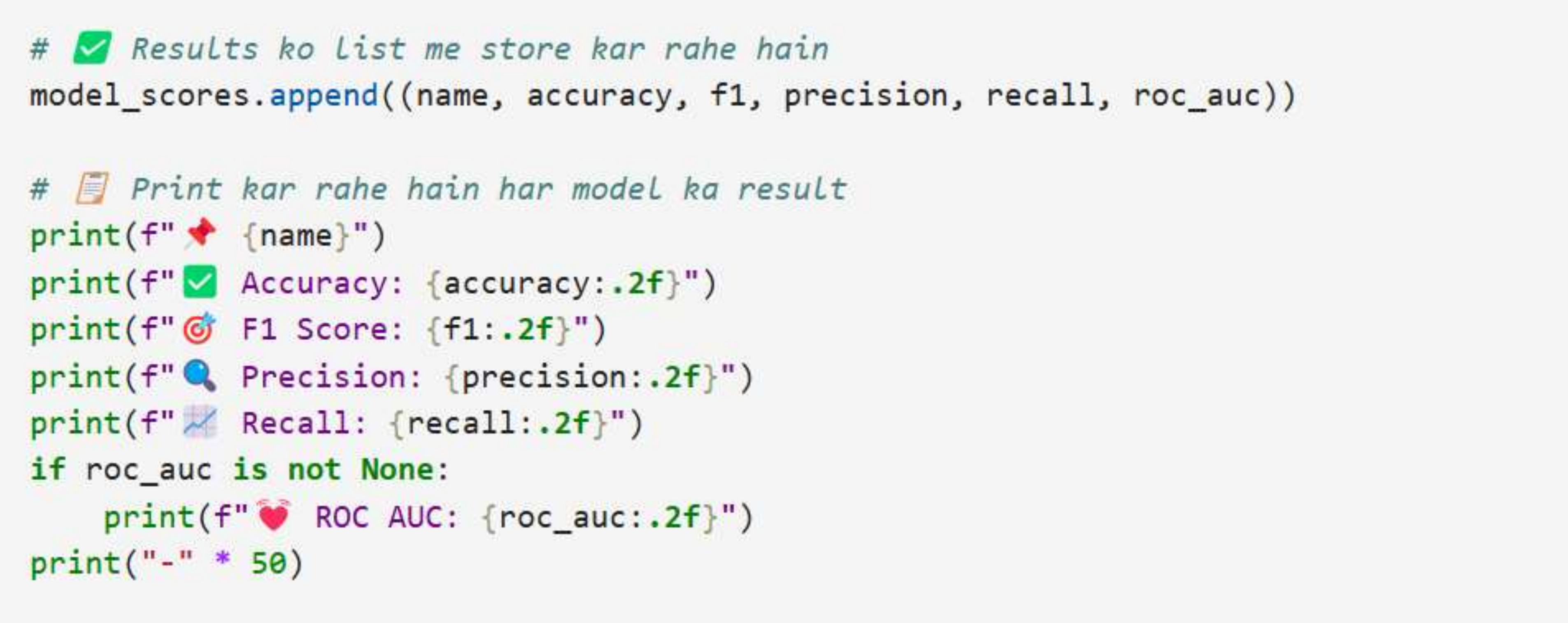


```python
# 🧪 5. Split the Data into Train and Test Sets
# Model ko train karne ke liye 80% data, aur test karne ke liye 20% rakhte hain

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

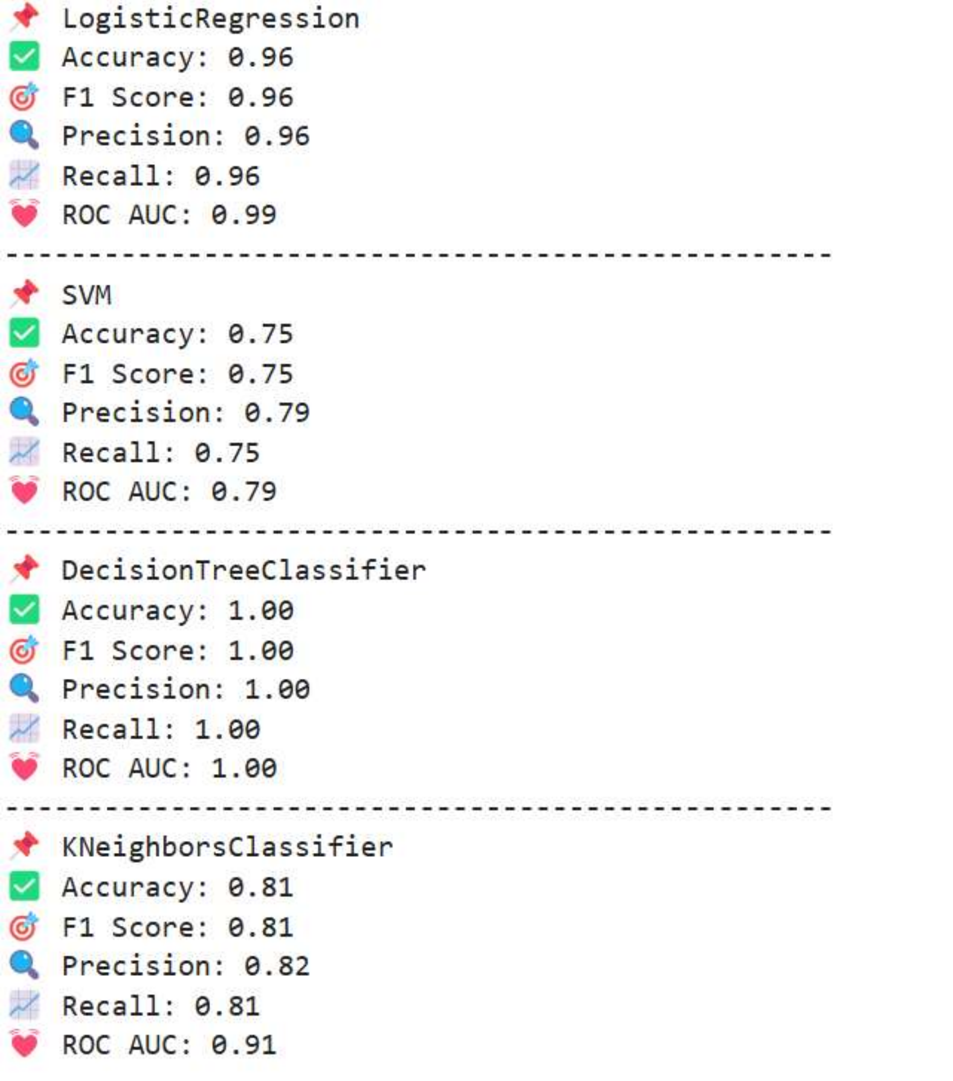
```python
# 🤖 6. Define Classification Models
# Hum 4 alag-alag ML models try kar rahe hain - har model alag tarike se kaam karta hai

models = [
    ('LogisticRegression', LogisticRegression(random_state=42)),
    ('SVM', SVC(random_state=42, probability=True)),  # probability=True ROC ke liye zaroori hai
    ('DecisionTreeClassifier', DecisionTreeClassifier(random_state=42)),
    ('KNeighborsClassifier', KNeighborsClassifier())
]
```



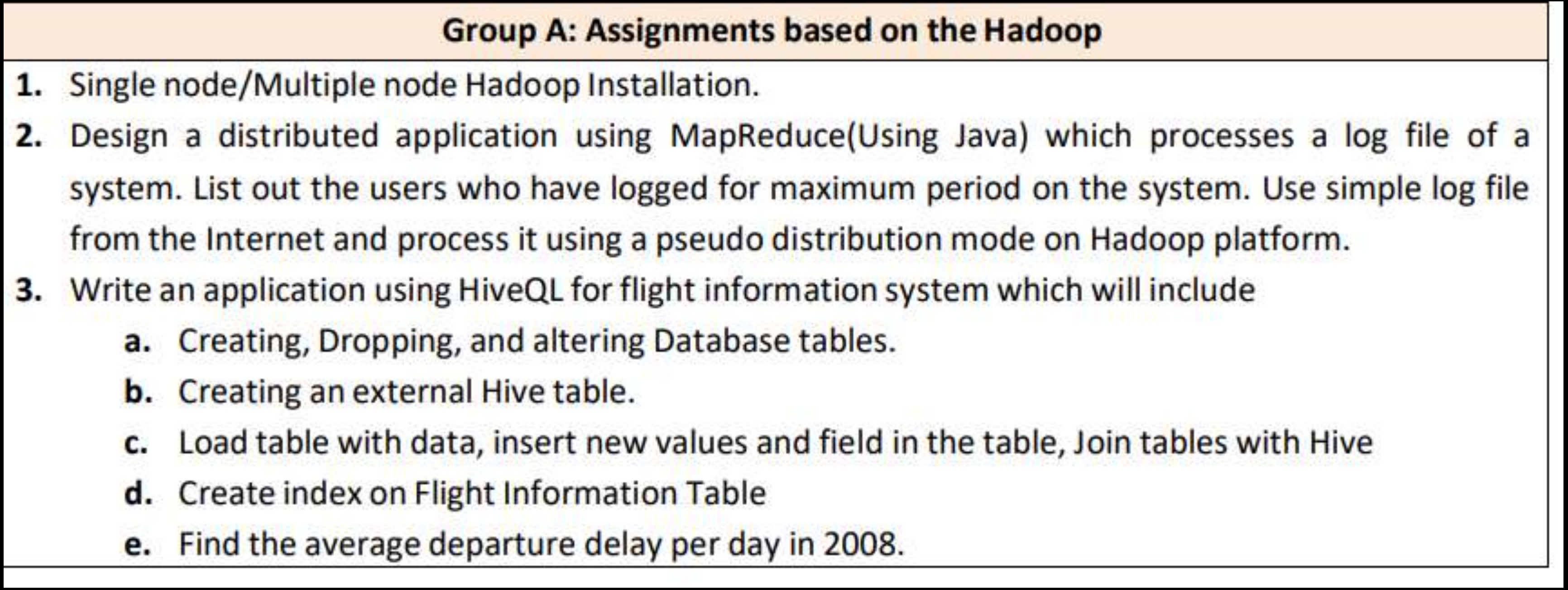
```python
# 📈 7. Train and Evaluate Each Model
# Har model ko train karenge, test data pe predict karenge, aur performance metrics calculate karenge

model_scores = []  # Sab models ke results store karne ke liye

for name, model in models:
    # 🔧 Model ko training data pe fit karte hain
    model.fit(X_train, y_train)

    # 🧪 Test data pe prediction karte hain
    y_pred = model.predict(X_test)

    # 🧠 ROC AUC ke liye probability chahiye hoti hai (0-1 confidence)
    y_proba = model.predict_proba(X_test)[:, 1] if hasattr(model, 'predict_proba') else None

    # 📊 Evaluation metrics nikal rahe hain:
    accuracy = accuracy_score(y_test, y_pred)  # Sahi predictions ka percentage
    f1 = f1_score(y_test, y_pred, average='weighted')  # Accuracy + recall ka balance
    precision = precision_score(y_test, y_pred, average='weighted')  # Positive predict karne ki accuracy
    recall = recall_score(y_test, y_pred, average='weighted')  # Kitne actual positives sahi predict hue
    roc_auc = roc_auc_score(y_test, y_proba) if y_proba is not None else None  # Confidence based metric
```



```python
# ✅ Results ko list me store kar rahe hain
model_scores.append((name, accuracy, f1, precision, recall, roc_auc))

# 📋 Print kar rahe hain har model ka result
print(f"📌 {name}")
print(f"✅ Accuracy: {accuracy:.2f}")
print(f"🎯 F1 Score: {f1:.2f}")
print(f"🔍 Precision: {precision:.2f}")
print(f"📈 Recall: {recall:.2f}")
if roc_auc is not None:
    print(f"💓 ROC AUC: {roc_auc:.2f}")
print("-" * 50)
```



```
📌 LogisticRegression
✅ Accuracy: 0.96
🎯 F1 Score: 0.96
🔍 Precision: 0.96
📈 Recall: 0.96
💓 ROC AUC: 0.99
--------------------------------------------------
📌 SVM
✅ Accuracy: 0.75
🎯 F1 Score: 0.75
🔍 Precision: 0.79
📈 Recall: 0.75
💓 ROC AUC: 0.79
--------------------------------------------------
📌 DecisionTreeClassifier
✅ Accuracy: 1.00
🎯 F1 Score: 1.00
🔍 Precision: 1.00
📈 Recall: 1.00
💓 ROC AUC: 1.00
--------------------------------------------------
📌 KNeighborsClassifier
✅ Accuracy: 0.81
🎯 F1 Score: 0.81
🔍 Precision: 0.82
📈 Recall: 0.81
💓 ROC AUC: 0.91
```



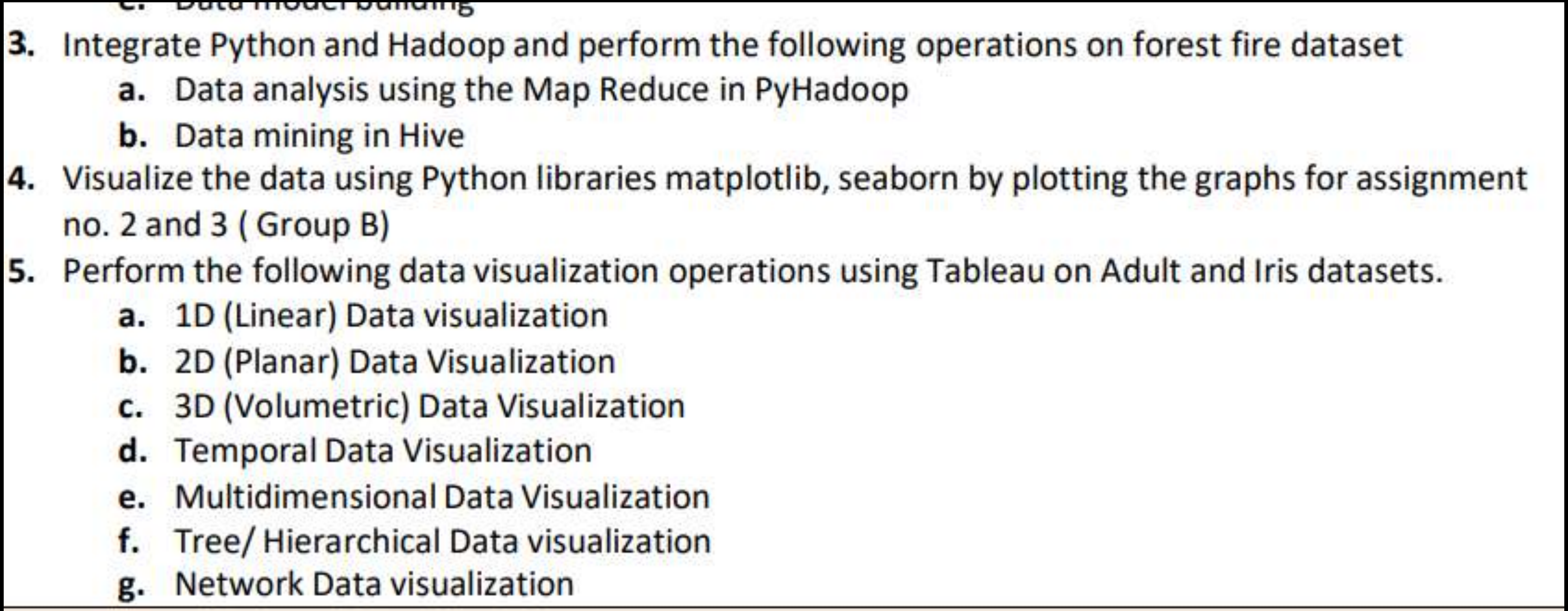
Group A: Assignments based on the Hadoop

1. Single node/Multiple node Hadoop Installation.
2. Design a distributed application using MapReduce(Using Java) which processes a log file of a system. List out the users who have logged for maximum period on the system. Use simple log file from the Internet and process it using a pseudo distribution mode on Hadoop platform.
3. Write an application using HiveQL for flight information system which will include
    a. Creating, Dropping, and altering Database tables.
    b. Creating an external Hive table.
    c. Load table with data, insert new values and field in the table, Join tables with Hive
    d. Create index on Flight Information Table
    e. Find the average departure delay per day in 2008.

1. Perform the following operations using Python on the Facebook metrics data sets
   a. Create data subsets
   b. Merge Data
   c. Sort Data
   d. Transposing Data
   e. Shape and reshape Data
2. Perform the following operations using Python on the Air quality and Heart Diseases data sets
   a. Data cleaning
   b. Data integration
   c. Data transformation
   d. Error correcting
   e. Data model building
3. Integrate Python and Hadoop and perform the following operations on forest fire dataset
   a. Data analysis using the Map Reduce in PyHadoop
   b. Data mining in Hive
4. Visualize the data using Python libraries matplotlib, seaborn by plotting the graphs for assignment no. 2 and 3 (Group B)
5. Perform the following data visualization operations using Tableau on Adult and Iris datasets.
   a. 1D (Linear) Data visualization
   b. 2D (Planar) Data Visualization
   c. 3D (Volumetric) Data Visualization
   d. Temporal Data Visualization
   e. Multidimensional Data Visualization
   f. Tree/ Hierarchical Data visualization
   g. Network Data visualization

Group C: Model Implementation

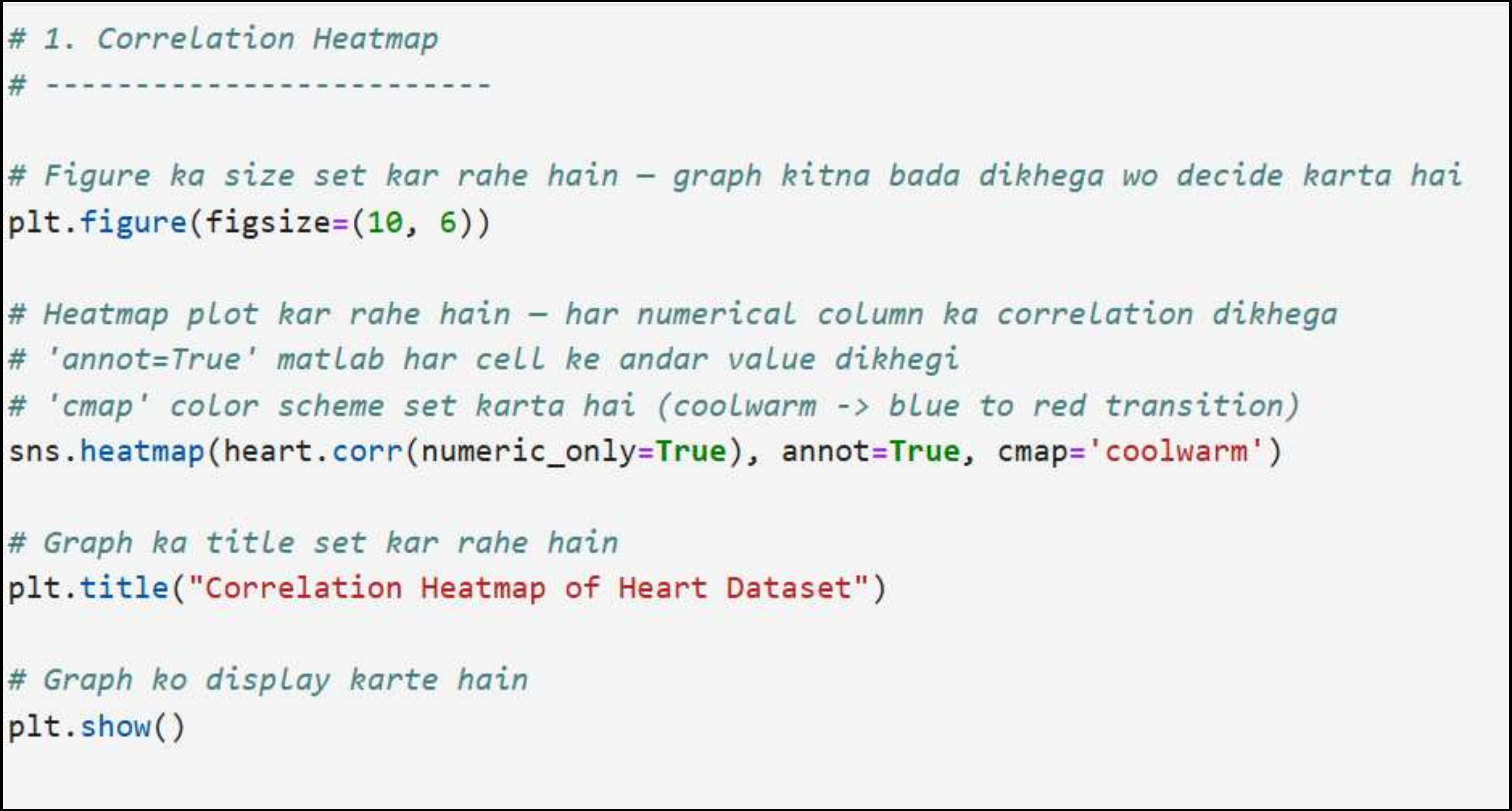
1. Create a review scrapper for any ecommerce website to fetch real time comments, reviews, ratings, comment tags, customer name using Python.
2. Develop a mini project in a group using different predictive models techniques to solve any real life problem. (Refer link dataset- https://www.kaggle.com/tanmoyie/us-graduate-schools- admission-parameters)



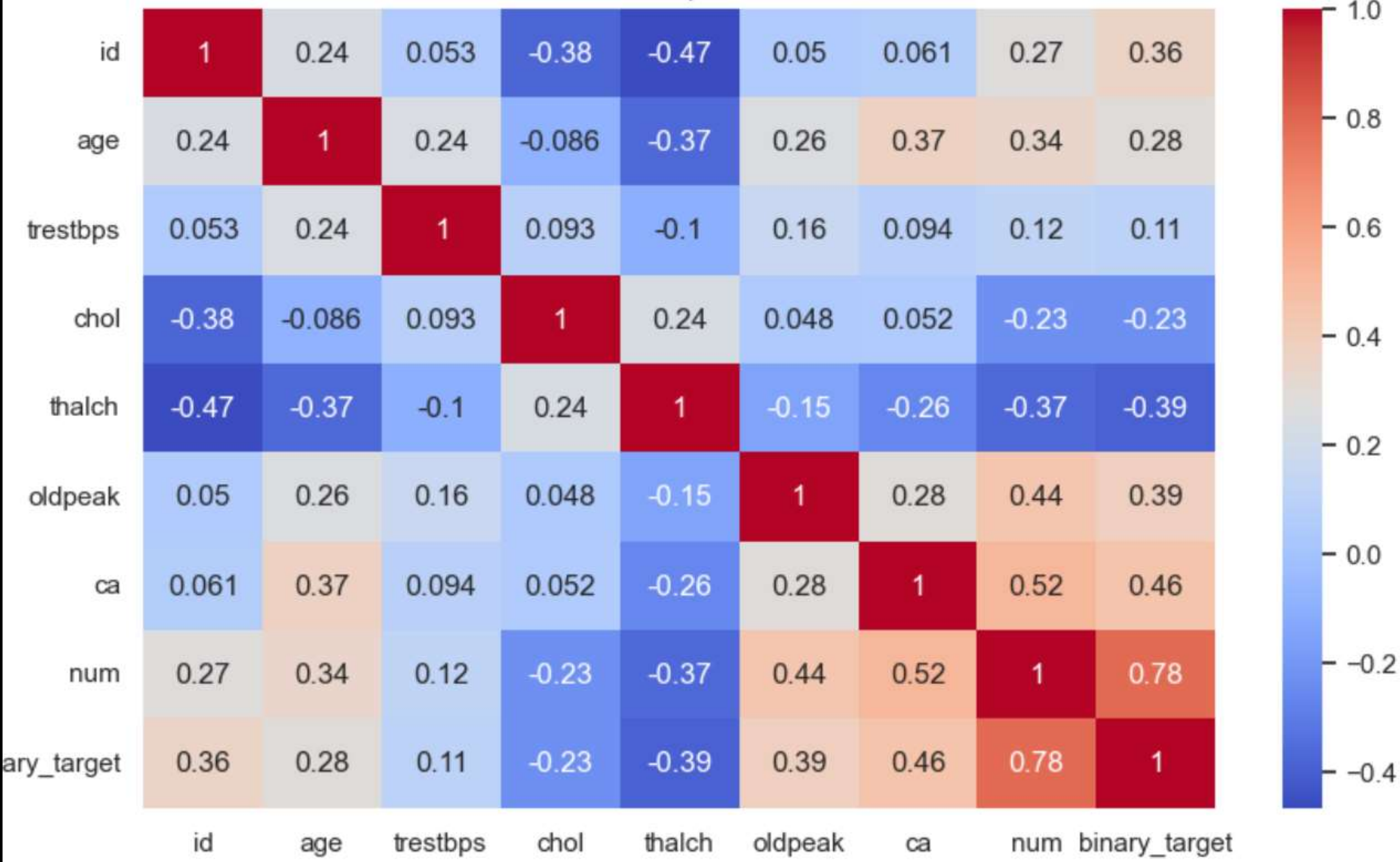
c. Data model building

3. Integrate Python and Hadoop and perform the following operations on forest fire dataset
    a. Data analysis using the Map Reduce in PyHadoop
    b. Data mining in Hive
4. Visualize the data using Python libraries matplotlib, seaborn by plotting the graphs for assignment no. 2 and 3 (Group B)
5. Perform the following data visualization operations using Tableau on Adult and Iris datasets.
    a. 1D (Linear) Data visualization
    b. 2D (Planar) Data Visualization
    c. 3D (Volumetric) Data Visualization
    d. Temporal Data Visualization
    e. Multidimensional Data Visualization
    f. Tree/ Hierarchical Data visualization
    g. Network Data visualization

Assignment 4: Python Data Visualization using Matplotlib & Seaborn



```python
# 1. Correlation Heatmap
# -------------------------

# Figure ka size set kar rahe hain – graph kitna bada dikhega wo decide karta hai
plt.figure(figsize=(10, 6))

# Heatmap plot kar rahe hain – har numerical column ka correlation dikhega
# 'annot=True' matlab har cell ke andar value dikhegi
# 'cmap' color scheme set karta hai (coolwarm -> blue to red transition)
sns.heatmap(heart.corr(numeric_only=True), annot=True, cmap='coolwarm')

# Graph ka title set kar rahe hain
plt.title("Correlation Heatmap of Heart Dataset")

# Graph ko display karte hain
plt.show()
```

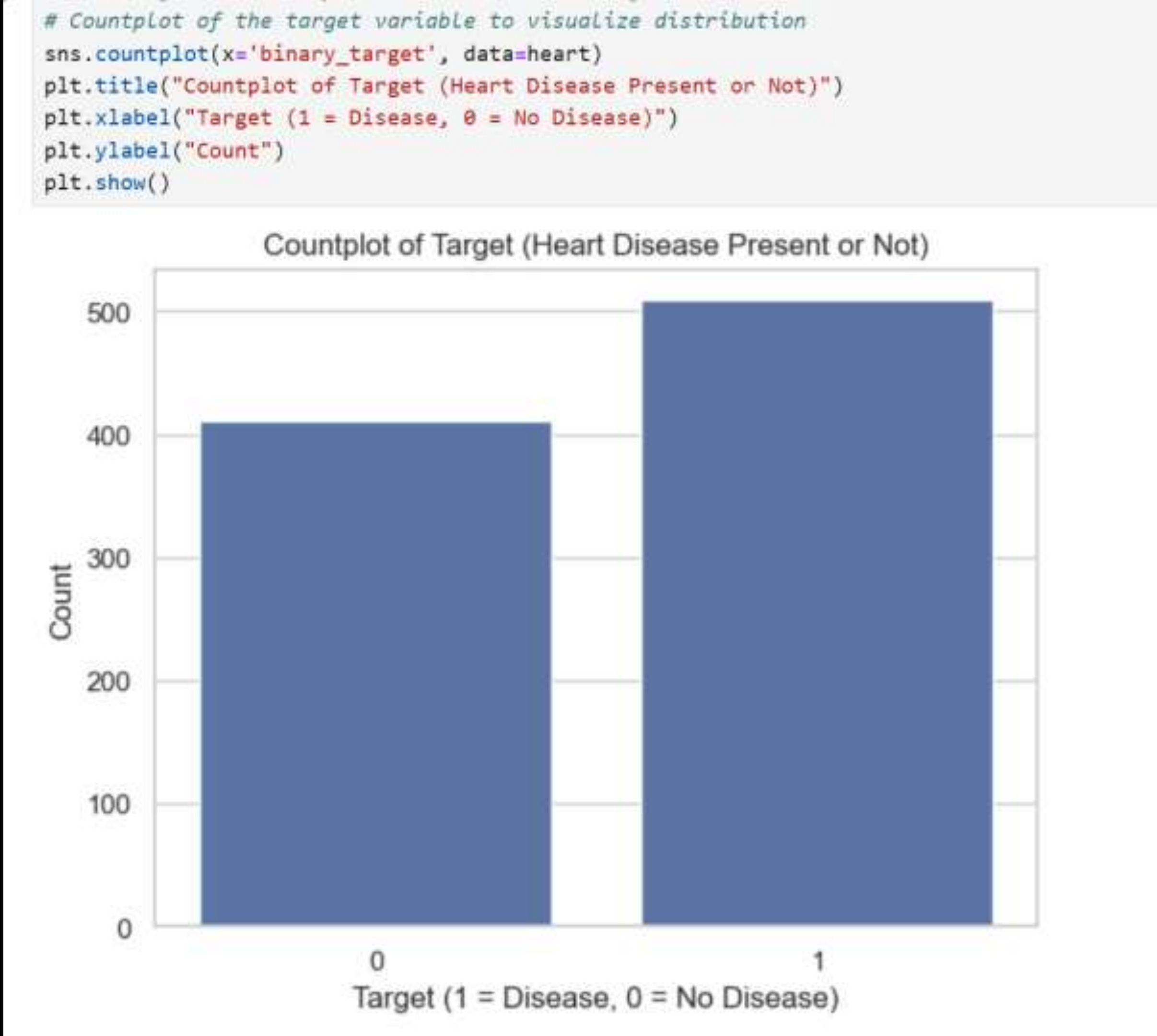


Correlation Heatmap of Heart Dataset

	id	age	trestbps	chol	thalch	oldpeak	ca	num	binary_target
id	1	0.24	0.053	-0.38	-0.47	0.05	0.061	0.27	0.36
age	0.24	1	0.24	-0.086	-0.37	0.26	0.37	0.34	0.28
trestbps	0.053	0.24	1	0.093	-0.1	0.16	0.094	0.12	0.11
chol	-0.38	-0.086	0.093	1	0.24	0.048	0.052	-0.23	-0.23
thalch	-0.47	-0.37	-0.1	0.24	1	-0.15	-0.26	-0.37	-0.39
oldpeak	0.05	0.26	0.16	0.048	-0.15	1	0.28	0.44	0.39
ca	0.061	0.37	0.094	0.052	-0.26	0.28	1	0.52	0.46
num	0.27	0.34	0.12	-0.23	-0.37	0.44	0.52	1	0.78
ary_target	0.36	0.28	0.11	-0.23	-0.39	0.39	0.46	0.78	1

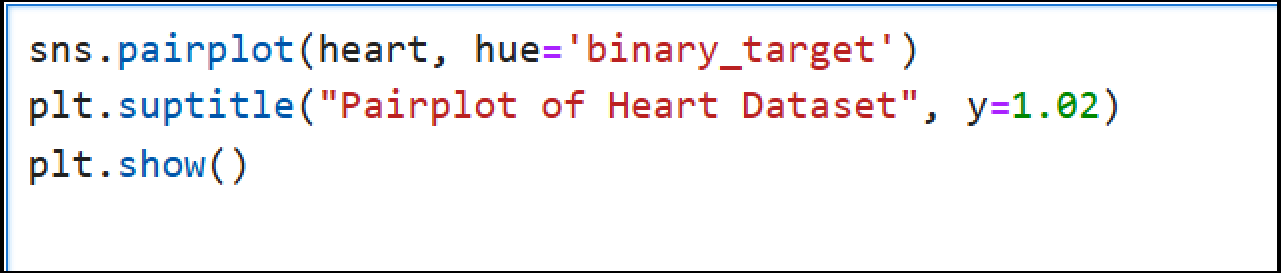
- **Positive correlation** ka matlab: dono values saath-saath badhti hain.
- **Negative correlation** ka matlab: ek value badhti hai to doosri ghatti hai.

🔢 **Feature Pair**	🔁 **Correlation**	⍰ **Explanation with Example Values (Hin+Eng)**
thalch vs binary_target	**-0.39**	Jab thalach (max heart rate) zyada hota hai, to heart disease hone ka chance kam hota hai. ⍰ *Example*: thalach = 170 → target = 0, thalach = 110 → target = 1
ca vs binary_target	**0.46**	Zyada number of major vessels (ca) blocked hone ka matlab heart disease ka high chance. ⍰ *Example*: ca = 0 → target = 0, ca = 3 → target = 1
chol vs binary_target	**-0.23**	Cholesterol badhne par thoda heart disease ka chance badhta hai, but weak relation hai. ⍰ *Example*: chol = 200 → target = 0, chol = 290 → target = 1
oldpeak vs binary_target	**0.39**	ST depression (exercise ke baad) agar zyada hai, to heart disease ka risk bhi zyada. ⍰ *Example*: oldpeak = 0.0 → target = 0, oldpeak = 3.0 → target = 1
age vs thalch	**-0.37**	Jaise jaise age badhta hai, maximum heart rate (thalch) naturally kam hota hai. ⍰ *Example*: age = 30 → thalach = 180, age = 60 → thalach = 120

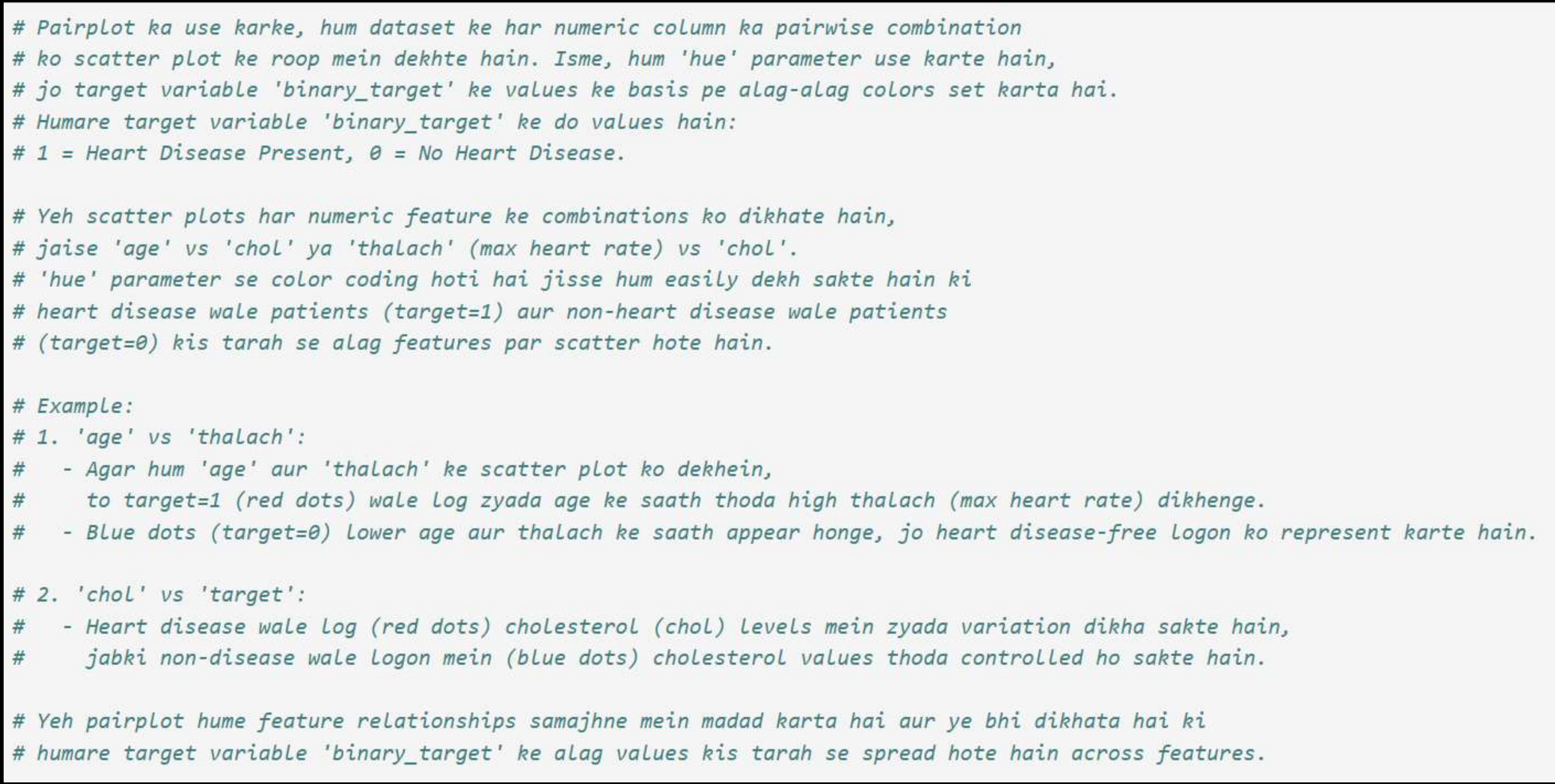


```python
# Countplot of the target variable to visualize distribution
sns.countplot(x='binary_target', data=heart)
plt.title("Countplot of Target (Heart Disease Present or Not)")
plt.xlabel("Target (1 = Disease, 0 = No Disease)")
plt.ylabel("Count")
plt.show()
```

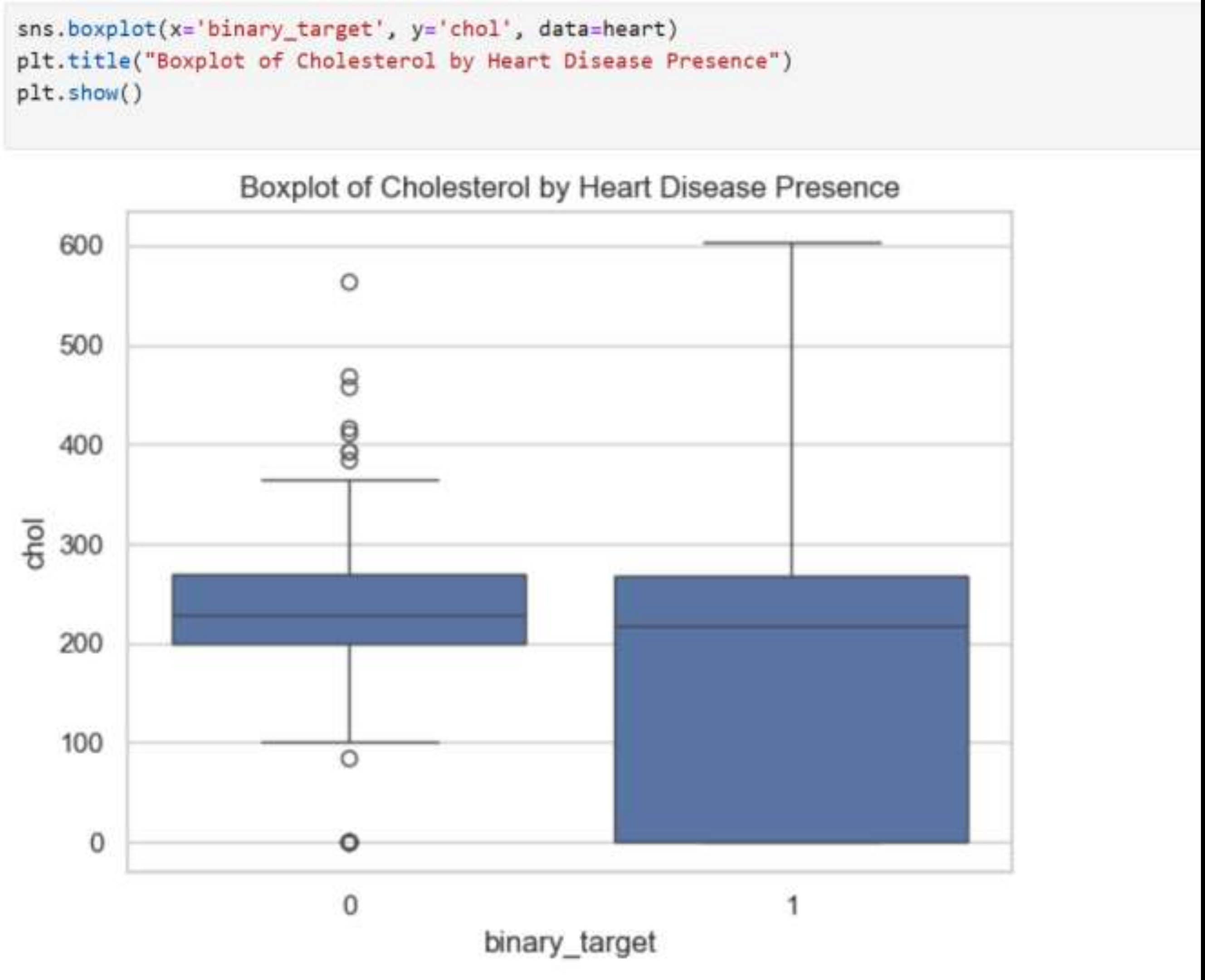
Countplot of Target (Heart Disease Present or Not)
Count
500
400
300
200
100
0
0
1
Target (1 = Disease, 0 = No Disease)



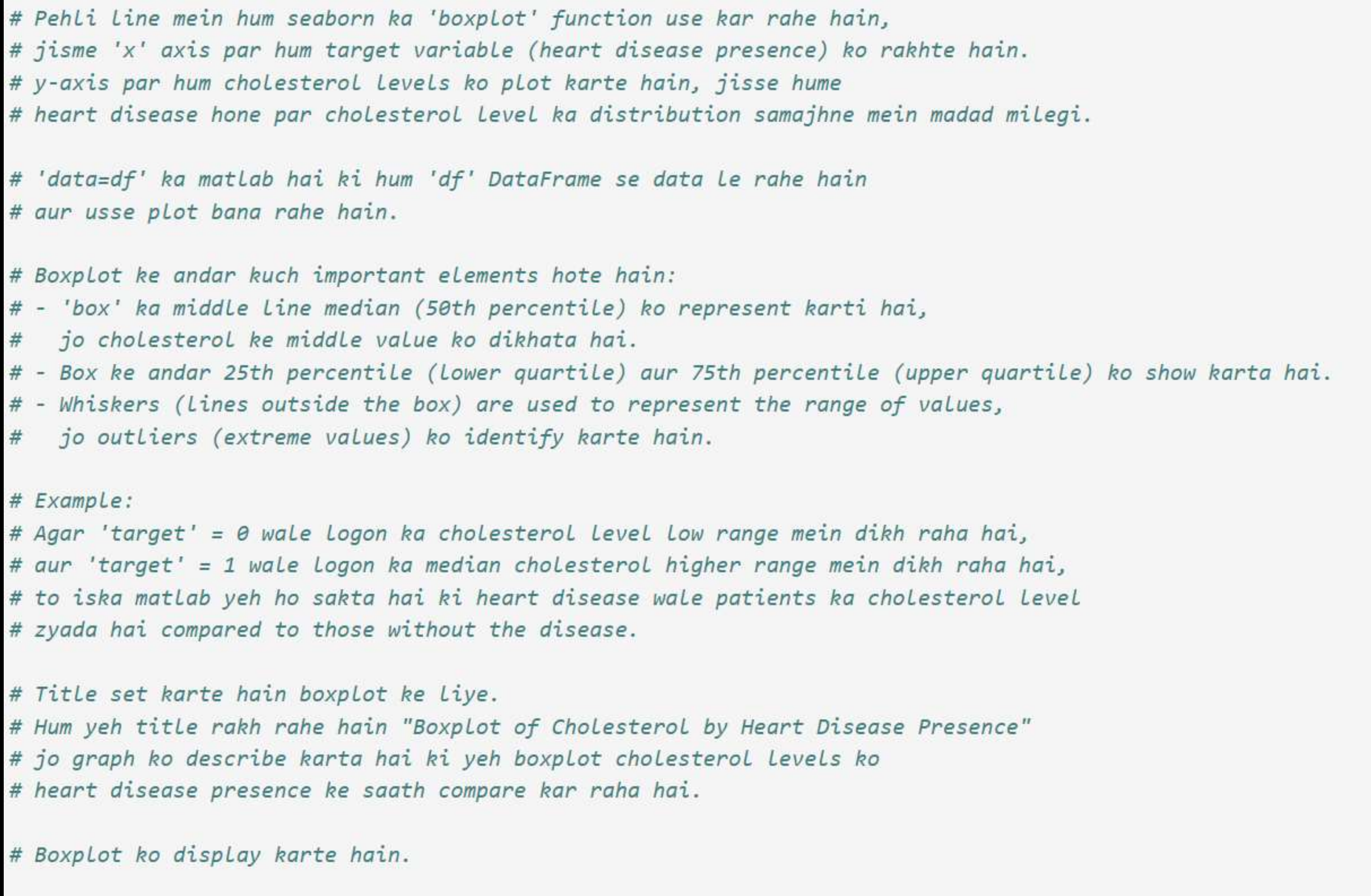
```
sns.pairplot(heart, hue='binary_target')
plt.suptitle("Pairplot of Heart Dataset", y=1.02)
plt.show()
```



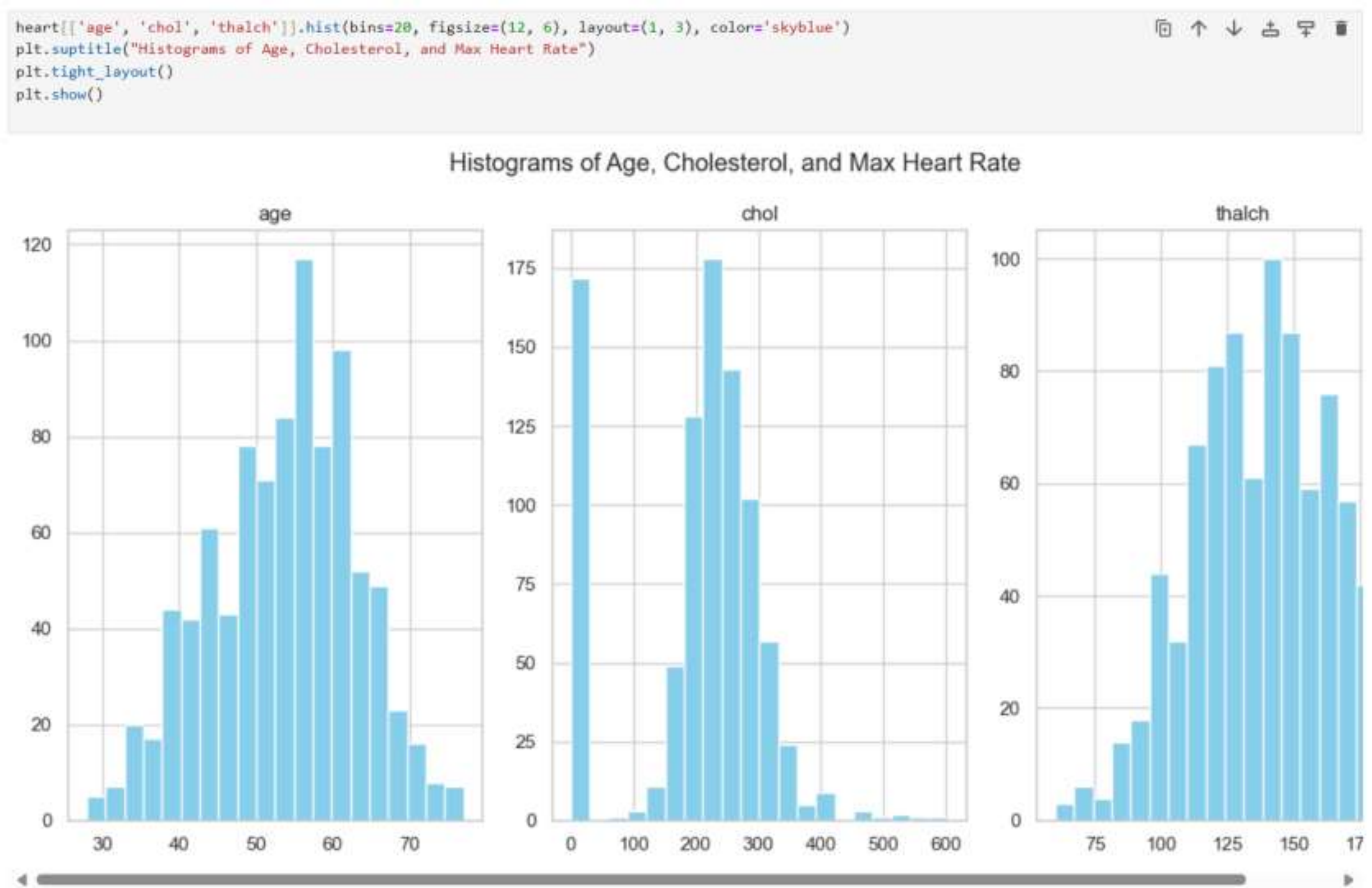
```
# Pairplot ka use karke, hum dataset ke har numeric column ka pairwise combination
# ko scatter plot ke roop mein dekhte hain. Isme, hum 'hue' parameter use karte hain,
# jo target variable 'binary_target' ke values ke basis pe alag-alag colors set karta hai.
# Humare target variable 'binary_target' ke do values hain:
# 1 = Heart Disease Present, 0 = No Heart Disease.

# Yeh scatter plots har numeric feature ke combinations ko dikhate hain,
# jaise 'age' vs 'chol' ya 'thalach' (max heart rate) vs 'chol'.
# 'hue' parameter se color coding hoti hai jisse hum easily dekh sakte hain ki
# heart disease wale patients (target=1) aur non-heart disease wale patients
# (target=0) kis tarah se alag features par scatter hote hain.

# Example:
# 1. 'age' vs 'thalach':
#   - Agar hum 'age' aur 'thalach' ke scatter plot ko dekhein,
#     to target=1 (red dots) wale log zyada age ke saath thoda high thalach (max heart rate) dikhenge.
#   - Blue dots (target=0) lower age aur thalach ke saath appear honge, jo heart disease-free logon ko represent karte hain.

# 2. 'chol' vs 'target':
#   - Heart disease wale log (red dots) cholesterol (chol) levels mein zyada variation dikha sakte hain,
#     jabki non-disease wale logon mein (blue dots) cholesterol values thoda controlled ho sakte hain.

# Yeh pairplot hume feature relationships samajhne mein madad karta hai aur ye bhi dikhata hai ki
# humare target variable 'binary_target' ke alag values kis tarah se spread hote hain across features.
```



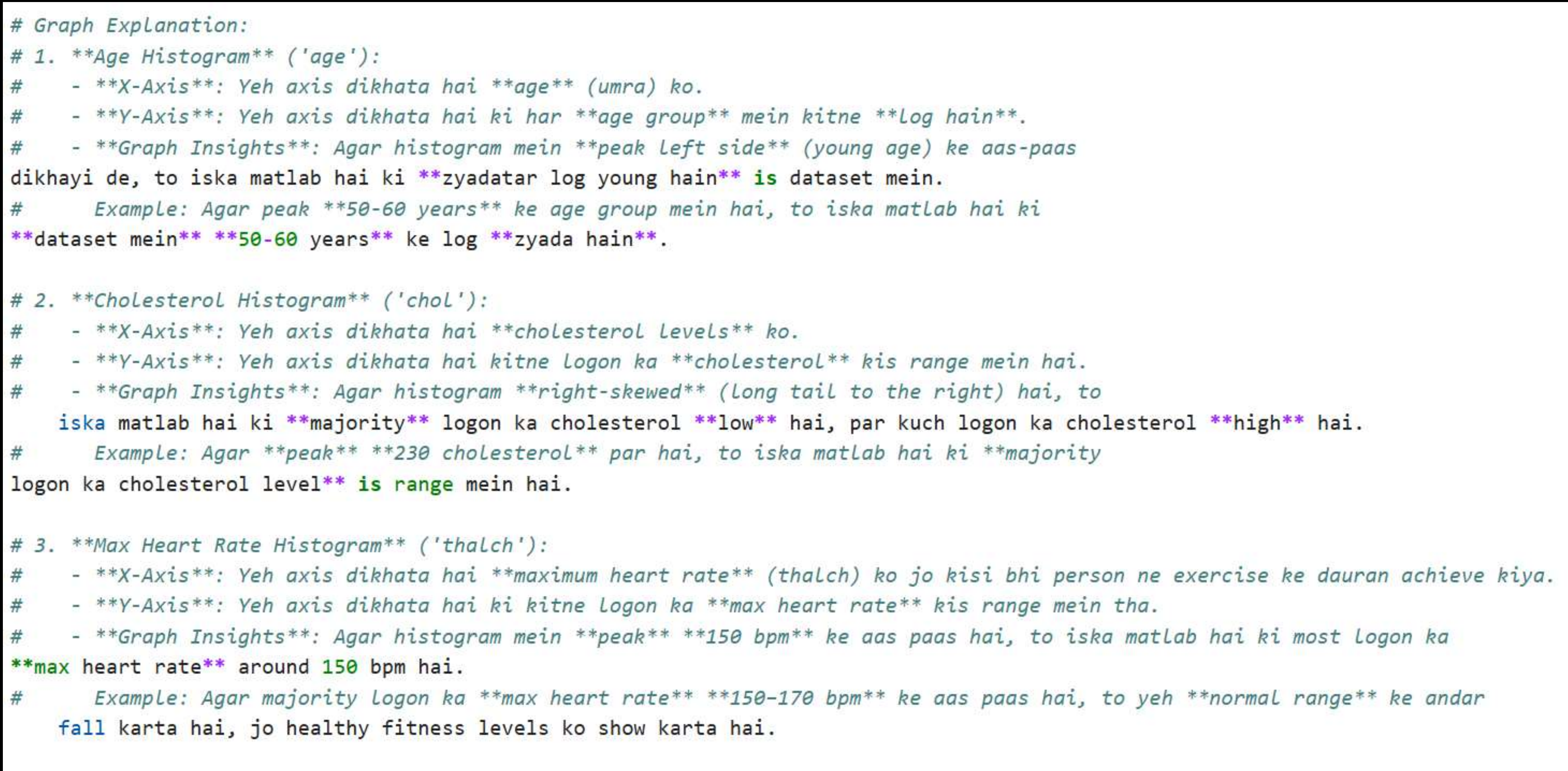
```
sns.boxplot(x='binary_target', y='chol', data=heart)
plt.title("Boxplot of Cholesterol by Heart Disease Presence")
plt.show()
```



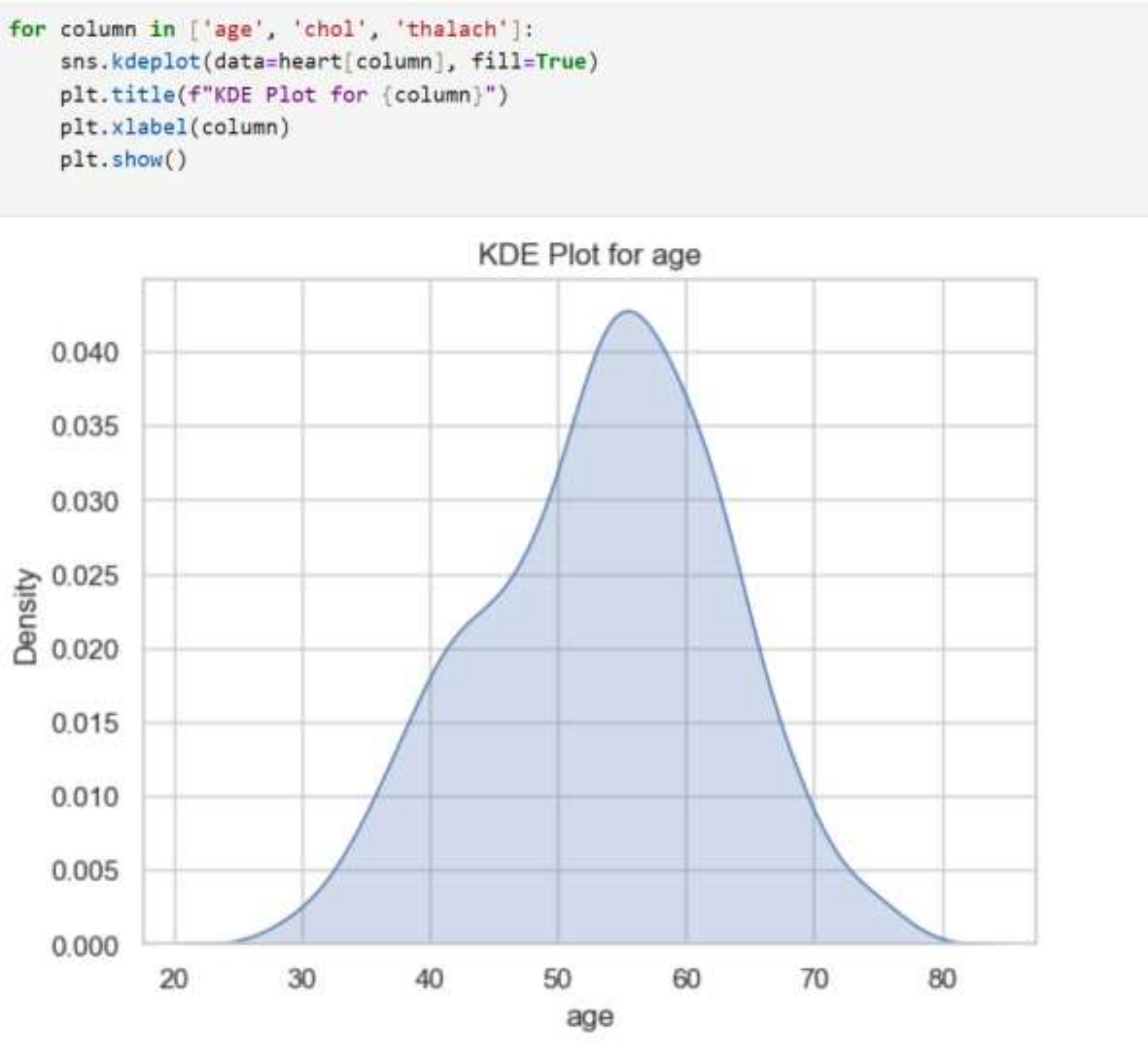
```
# Pehli line mein hum seaborn ka 'boxplot' function use kar rahe hain,
# jisme 'x' axis par hum target variable (heart disease presence) ko rakhte hain.
# y-axis par hum cholesterol levels ko plot karte hain, jisse hume
# heart disease hone par cholesterol level ka distribution samajhne mein madad milegi.

# 'data=df' ka matlab hai ki hum 'df' DataFrame se data le rahe hain
# aur usse plot bana rahe hain.

# Boxplot ke andar kuch important elements hote hain:
# - 'box' ka middle line median (50th percentile) ko represent karti hai,
#   jo cholesterol ke middle value ko dikhata hai.
# - Box ke andar 25th percentile (lower quartile) aur 75th percentile (upper quartile) ko show karta hai.
# - Whiskers (lines outside the box) are used to represent the range of values,
#   jo outliers (extreme values) ko identify karte hain.

# Example:
# Agar 'target' = 0 wale logon ka cholesterol level low range mein dikh raha hai,
# aur 'target' = 1 wale logon ka median cholesterol higher range mein dikh raha hai,
# to iska matlab yeh ho sakta hai ki heart disease wale patients ka cholesterol level
# zyada hai compared to those without the disease.

# Title set karte hain boxplot ke liye.
# Hum yeh title rakh rahe hain "Boxplot of Cholesterol by Heart Disease Presence"
# jo graph ko describe karta hai ki yeh boxplot cholesterol levels ko
# heart disease presence ke saath compare kar raha hai.

# Boxplot ko display karte hain.
```



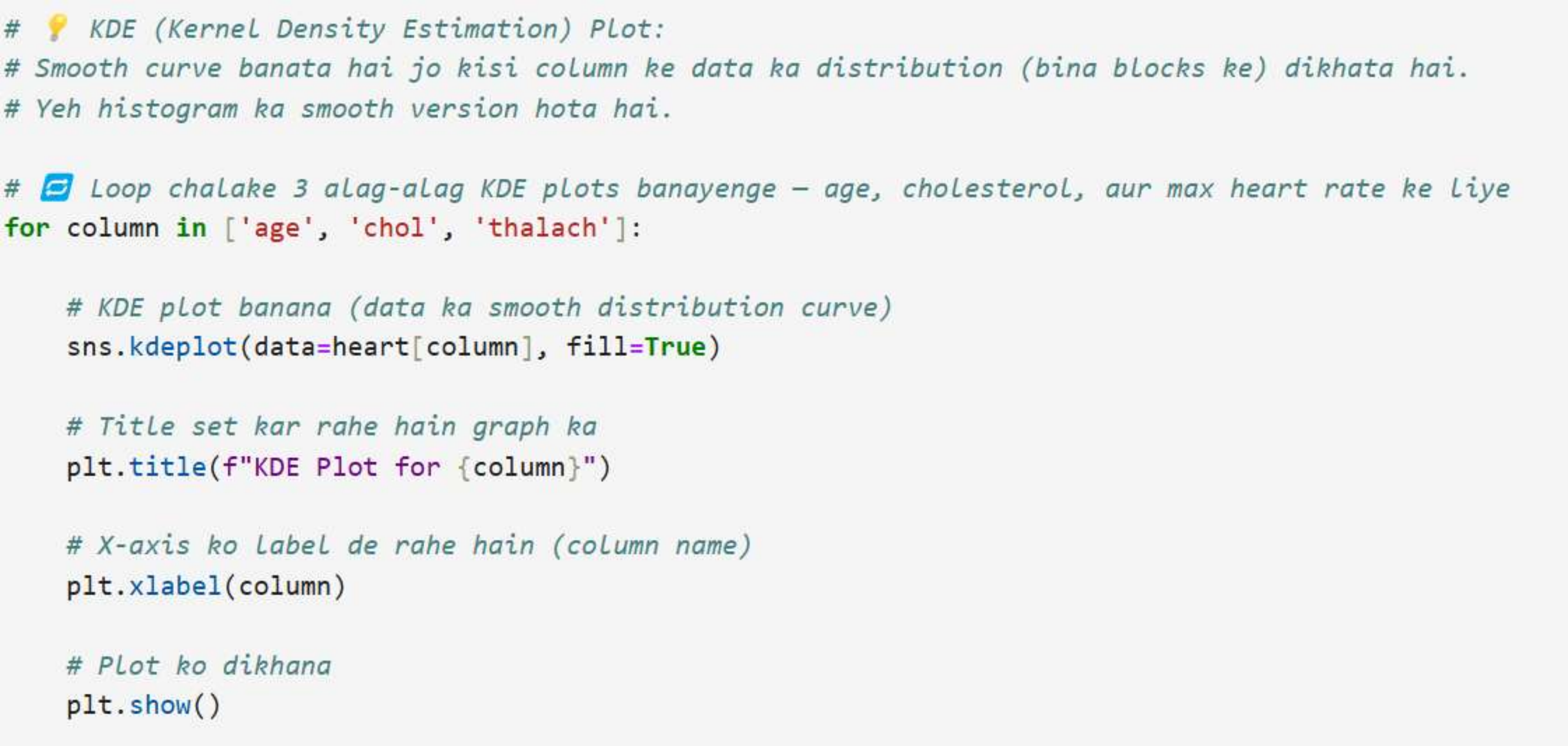
```python
heart[['age', 'chol', 'thalch']].hist(bins=20, figsize=(12, 6), layout=(1, 3), color='skyblue')
plt.suptitle("Histograms of Age, Cholesterol, and Max Heart Rate")
plt.tight_layout()
plt.show()
```



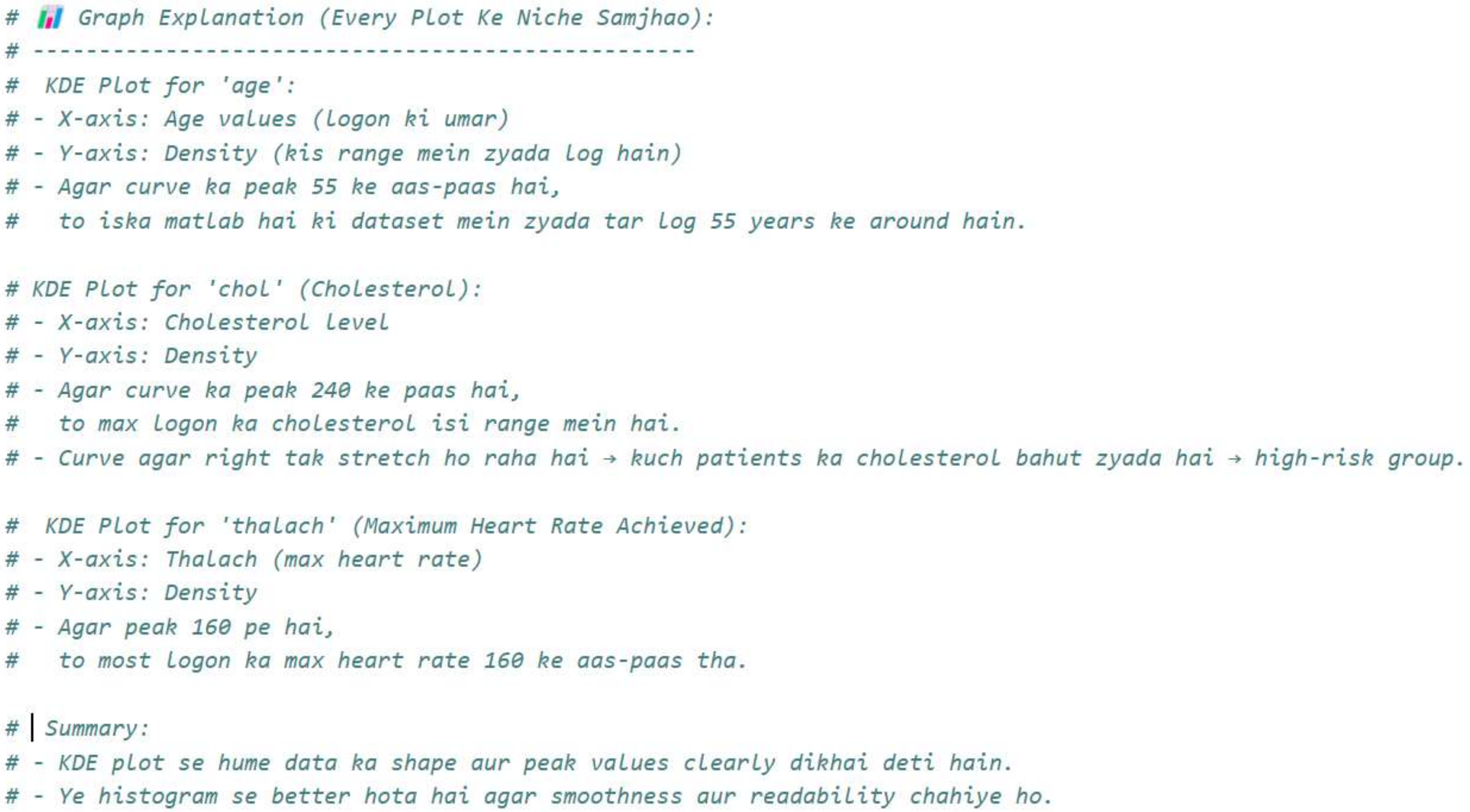
```
# Graph Explanation:
# 1. **Age Histogram** ('age'):
#    - **X-Axis**: Yeh axis dikhata hai **age** (umra) ko.
#    - **Y-Axis**: Yeh axis dikhata hai ki har **age group** mein kitne **log hain**.
#    - **Graph Insights**: Agar histogram mein **peak left side** (young age) ke aas-paas
dikhayi de, to iska matlab hai ki **zyadatar log young hain** is dataset mein.
#      Example: Agar peak **50-60 years** ke age group mein hai, to iska matlab hai ki
**dataset mein** **50-60 years** ke log **zyada hain**.

# 2. **Cholesterol Histogram** ('chol'):
#    - **X-Axis**: Yeh axis dikhata hai **cholesterol levels** ko.
#    - **Y-Axis**: Yeh axis dikhata hai kitne logon ka **cholesterol** kis range mein hai.
#    - **Graph Insights**: Agar histogram **right-skewed** (long tail to the right) hai, to
    iska matlab hai ki **majority** logon ka cholesterol **low** hai, par kuch logon ka cholesterol **high** hai.
#      Example: Agar **peak** **230 cholesterol** par hai, to iska matlab hai ki **majority
logon ka cholesterol level** is range mein hai.

# 3. **Max Heart Rate Histogram** ('thalch'):
#    - **X-Axis**: Yeh axis dikhata hai **maximum heart rate** (thalch) ko jo kisi bhi person ne exercise ke dauran achieve kiya.
#    - **Y-Axis**: Yeh axis dikhata hai ki kitne logon ka **max heart rate** kis range mein tha.
#    - **Graph Insights**: Agar histogram mein **peak** **150 bpm** ke aas paas hai, to iska matlab hai ki most logon ka
**max heart rate** around 150 bpm hai.
#      Example: Agar majority logon ka **max heart rate** **150-170 bpm** ke aas paas hai, to yeh **normal range** ke andar
    fall karta hai, jo healthy fitness levels ko show karta hai.
```



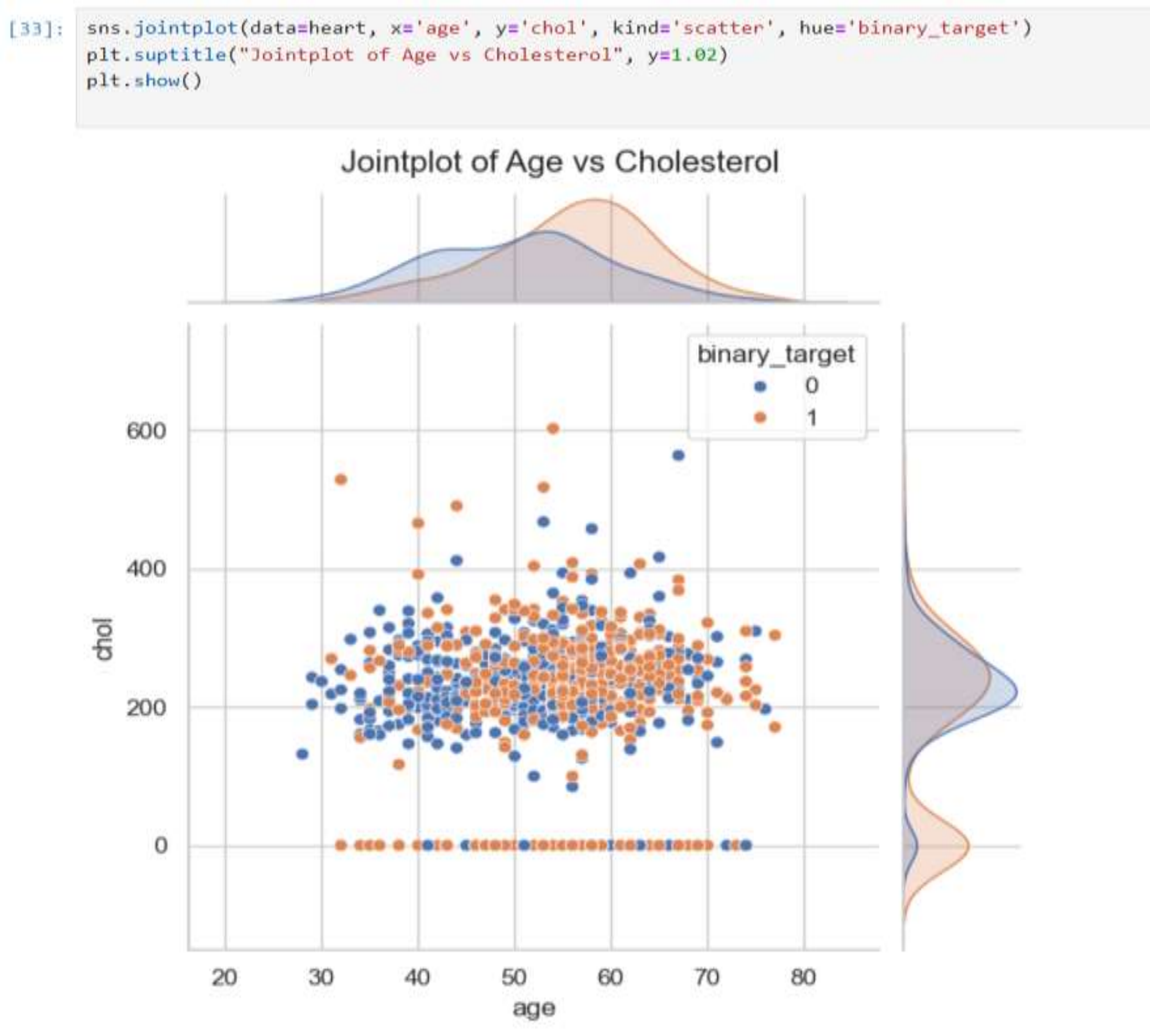
```
for column in ['age', 'chol', 'thalach']:
    sns.kdeplot(data=heart[column], fill=True)
    plt.title(f"KDE Plot for {column}")
    plt.xlabel(column)
    plt.show()
```



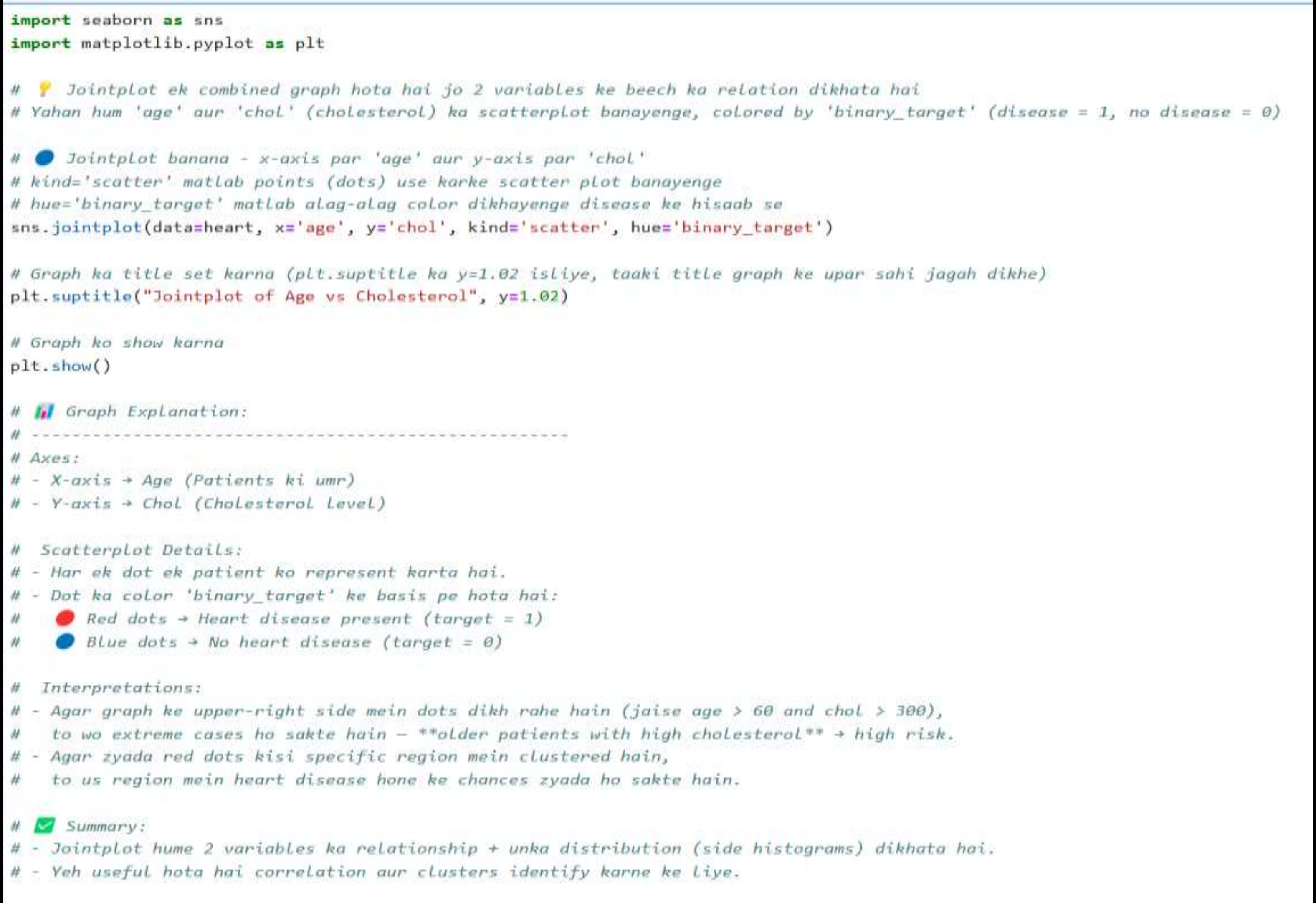
```python
# 💡 KDE (Kernel Density Estimation) Plot:
# Smooth curve banata hai jo kisi column ke data ka distribution (bina blocks ke) dikhata hai.
# Yeh histogram ka smooth version hota hai.

# 🔁 Loop chalake 3 alag-alag KDE plots banayenge — age, cholesterol, aur max heart rate ke liye
for column in ['age', 'chol', 'thalach']:

    # KDE plot banana (data ka smooth distribution curve)
    sns.kdeplot(data=heart[column], fill=True)

    # Title set kar rahe hain graph ka
    plt.title(f"KDE Plot for {column}")

    # X-axis ko label de rahe hain (column name)
    plt.xlabel(column)

    # Plot ko dikhana
    plt.show()
```



```
# 📊 Graph Explanation (Every Plot Ke Niche Samjhao):
# --------------------------------------------------
#  KDE Plot for 'age':
# - X-axis: Age values (logon ki umar)
# - Y-axis: Density (kis range mein zyada log hain)
# - Agar curve ka peak 55 ke aas-paas hai,
#   to iska matlab hai ki dataset mein zyada tar log 55 years ke around hain.

# KDE Plot for 'chol' (Cholesterol):
# - X-axis: Cholesterol level
# - Y-axis: Density
# - Agar curve ka peak 240 ke paas hai,
#   to max logon ka cholesterol isi range mein hai.
# - Curve agar right tak stretch ho raha hai → kuch patients ka cholesterol bahut zyada hai → high-risk group.

#  KDE Plot for 'thalach' (Maximum Heart Rate Achieved):
# - X-axis: Thalach (max heart rate)
# - Y-axis: Density
# - Agar peak 160 pe hai,
#   to most logon ka max heart rate 160 ke aas-paas tha.

# Summary:
# - KDE plot se hume data ka shape aur peak values clearly dikhai deti hain.
# - Ye histogram se better hota hai agar smoothness aur readability chahiye ho.
```



```
[33]: sns.jointplot(data=heart, x='age', y='chol', kind='scatter', hue='binary_target')
      plt.suptitle("Jointplot of Age vs Cholesterol", y=1.02)
      plt.show()
```



```python
import seaborn as sns
import matplotlib.pyplot as plt

# 💡 Jointplot ek combined graph hota hai jo 2 variables ke beech ka relation dikhata hai
# Yahan hum 'age' aur 'chol' (cholesterol) ka scatterplot banayenge, colored by 'binary_target' (disease = 1, no disease = 0)

# 🔵 Jointplot banana - x-axis par 'age' aur y-axis par 'chol'
# kind='scatter' matlab points (dots) use karke scatter plot banayenge
# hue='binary_target' matlab alag-alag color dikhayenge disease ke hisaab se
sns.jointplot(data=heart, x='age', y='chol', kind='scatter', hue='binary_target')

# Graph ka title set karna (plt.suptitle ka y=1.02 isliye, taaki title graph ke upar sahi jagah dikhe)
plt.suptitle("Jointplot of Age vs Cholesterol", y=1.02)

# Graph ko show karna
plt.show()

# 📊 Graph Explanation:
# -------------------------------------------------------
# Axes:
# - X-axis → Age (Patients ki umr)
# - Y-axis → Chol (Cholesterol Level)

#  Scatterplot Details:
# - Har ek dot ek patient ko represent karta hai.
# - Dot ka color 'binary_target' ke basis pe hota hai:
#    🔴 Red dots → Heart disease present (target = 1)
#    🔵 Blue dots → No heart disease (target = 0)

#  Interpretations:
# - Agar graph ke upper-right side mein dots dikh rahe hain (jaise age > 60 and chol > 300),
#   to wo extreme cases ho sakte hain — **older patients with high cholesterol** → high risk.
# - Agar zyada red dots kisi specific region mein clustered hain,
#   to us region mein heart disease hone ke chances zyada ho sakte hain.

# ✅ Summary:
# - Jointplot hume 2 variables ka relationship + unka distribution (side histograms) dikhata hai.
# - Yeh useful hota hai correlation aur clusters identify karne ke liye.
```

📄 Air Quality Data Analysis & Visualizations

Library	Use
pandas (pd)	Data read karne, clean karne aur table jaisa structure manage karne ke liye
seaborn (sns)	Stylish aur statistical visualizations ke liye (matplotlib ka wrapper)
matplotlib.pyplot (plt)	Graph banane ke liye, basic plotting library

Part	Explanation
pd.read_csv("AirQuality.csv")	File ka naam "AirQuality.csv" hai, jise load kar rahe hain
sep=';'	Is CSV file mein data comma , se nahi balki semicolon ; se alag-alag hai
skipinitialspace=True	Columns ke values ke aage jo spaces hote hain unhe ignore karega

Column Name	Reason to Remove
Unnamed: 15	Blank hai
Unnamed: 16	Blank hai

```
# NaN values ko handle karne ke liye (optional step)
df = df.dropna()
```

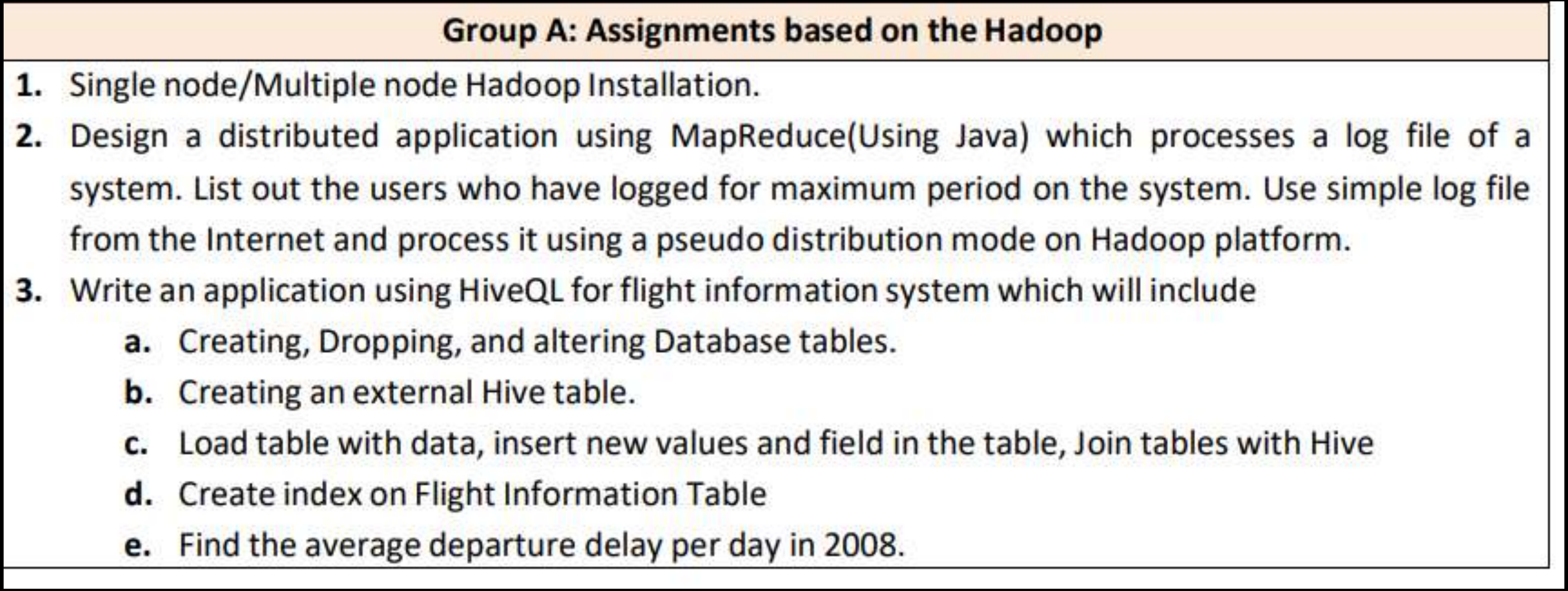
- dropna() ka matlab: jaha bhi missing (NaN) data hai, us row ko hata do

- Ye step optional hai, but agar missing data zyada hai to analysis galat ho sakta hai

⚠️ **Warning:** Bahut zyada rows delete ho sakti hain agar zyada missing values hon.

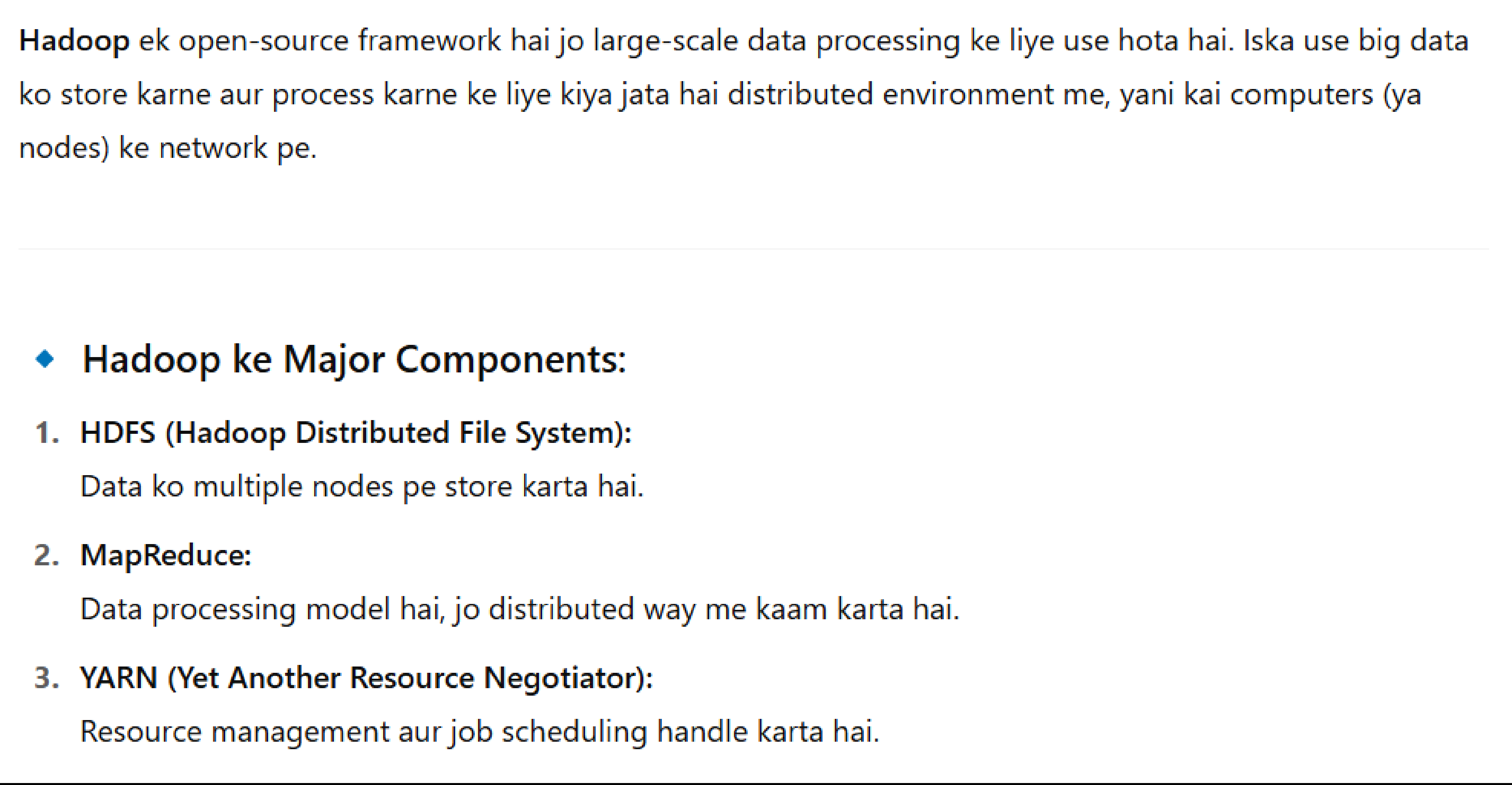
Alternative: fillna() se missing values ko average ya 0 se bhar bhi sakte ho.

"whitegrid" style ka matlab: **graph ke piche halka grey grid aata hai**, jo readability badhata hai



Group A: Assignments based on the Hadoop

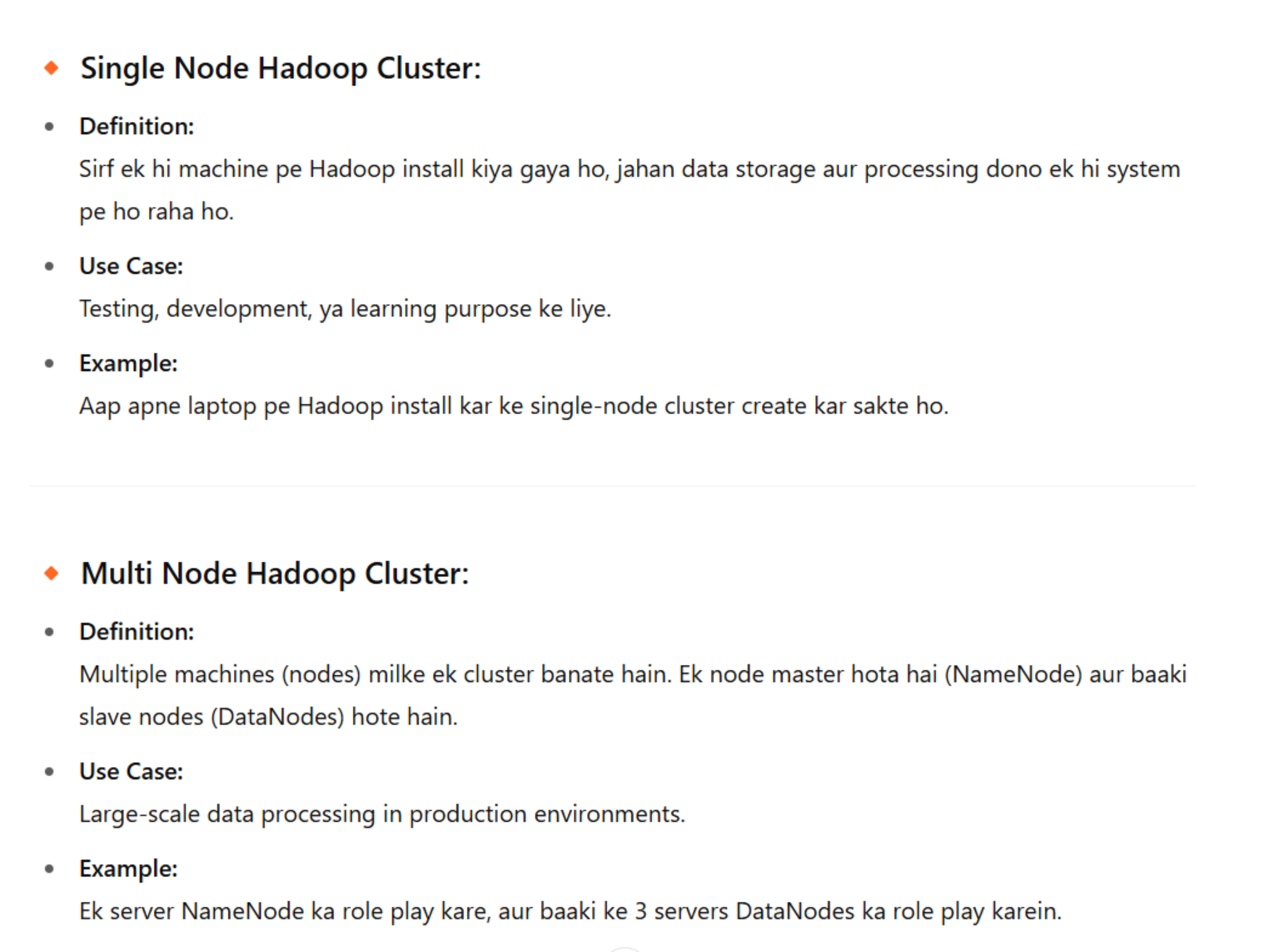
1. Single node/Multiple node Hadoop Installation.
2. Design a distributed application using MapReduce(Using Java) which processes a log file of a system. List out the users who have logged for maximum period on the system. Use simple log file from the Internet and process it using a pseudo distribution mode on Hadoop platform.
3. Write an application using HiveQL for flight information system which will include
   a. Creating, Dropping, and altering Database tables.
   b. Creating an external Hive table.
   c. Load table with data, insert new values and field in the table, Join tables with Hive
   d. Create index on Flight Information Table
   e. Find the average departure delay per day in 2008.



Hadoop ek open-source framework hai jo large-scale data processing ke liye use hota hai. Iska use big data ko store karne aur process karne ke liye kiya jata hai distributed environment me, yani kai computers (ya nodes) ke network pe.

◆ Hadoop ke Major Components:

1. **HDFS (Hadoop Distributed File System):**
   Data ko multiple nodes pe store karta hai.
2. **MapReduce:**
   Data processing model hai, jo distributed way me kaam karta hai.
3. **YARN (Yet Another Resource Negotiator):**
   Resource management aur job scheduling handle karta hai.



◆ Single Node Hadoop Cluster:

- **Definition:**
  Sirf ek hi machine pe Hadoop install kiya gaya ho, jahan data storage aur processing dono ek hi system pe ho raha ho.
- **Use Case:**
  Testing, development, ya learning purpose ke liye.
- **Example:**
  Aap apne laptop pe Hadoop install kar ke single-node cluster create kar sakte ho.

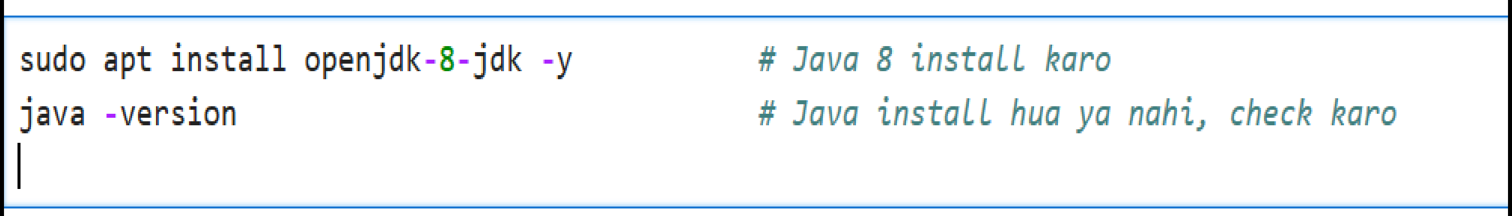
◆ Multi Node Hadoop Cluster:

- **Definition:**
  Multiple machines (nodes) milke ek cluster banate hain. Ek node master hota hai (NameNode) aur baaki slave nodes (DataNodes) hote hain.
- **Use Case:**
  Large-scale data processing in production environments.
- **Example:**
  Ek server NameNode ka role play kare, aur baaki ke 3 servers DataNodes ka role play karein.

📄 Assignment 1: Single Node Hadoop Installation (Hadoop 2.9.0)

Goal: Hadoop ko ek system pe (Single Node/multinode) install karna aur run karna.

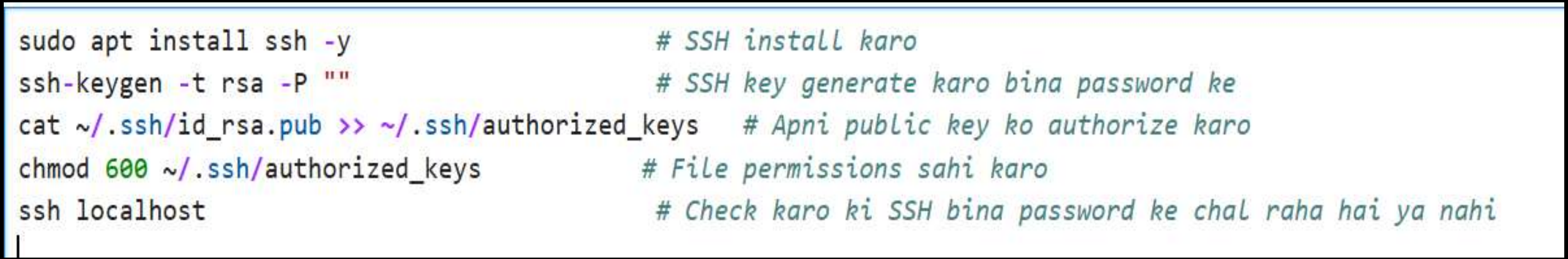
✅ 1. Java 8 Install karna (Hadoop ke liye required hai)



```
sudo apt install openjdk-8-jdk -y           # Java 8 install karo
java -version                               # Java install hua ya nahi, check karo
```

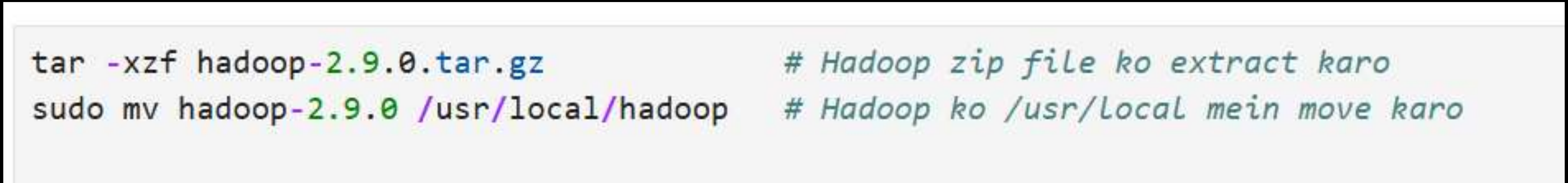
✓ 2. SSH Setup (Hadoop ko khud se connect karne ke liye zaroori hai)

(Hadoop ke liye password-less login chahiye hota hai)



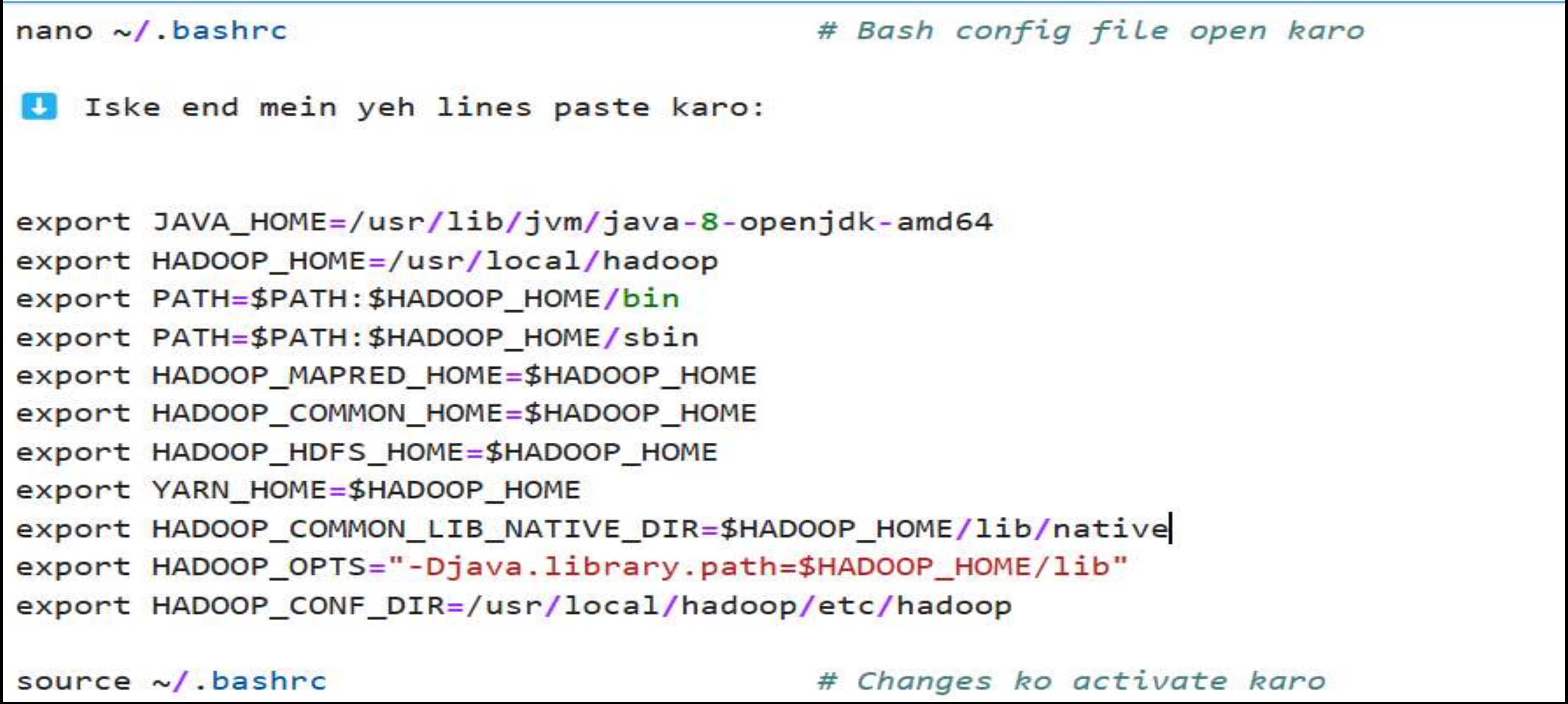
```
sudo apt install ssh -y                     # SSH install karo
ssh-keygen -t rsa -P ""                     # SSH key generate karo bina password ke
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys   # Apni public key ko authorize karo
chmod 600 ~/.ssh/authorized_keys           # File permissions sahi karo
ssh localhost                               # Check karo ki SSH bina password ke chal raha hai ya nahi
```

✅ 3. Hadoop File ko Extract & Move karo



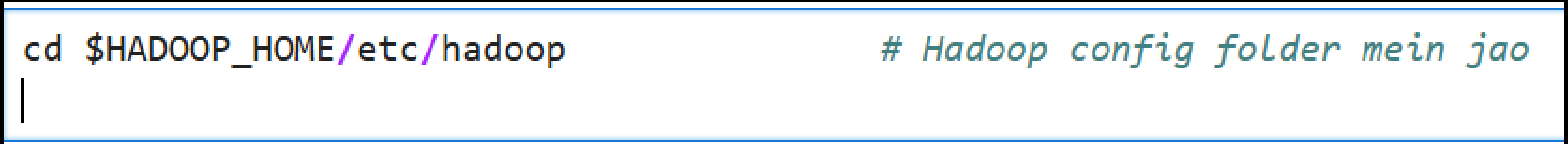
```
tar -xzf hadoop-2.9.0.tar.gz              # Hadoop zip file ko extract karo
sudo mv hadoop-2.9.0 /usr/local/hadoop   # Hadoop ko /usr/local mein move karo
```

🔰 STEP 4: Environment Variables set karo **.bashrc** file mein

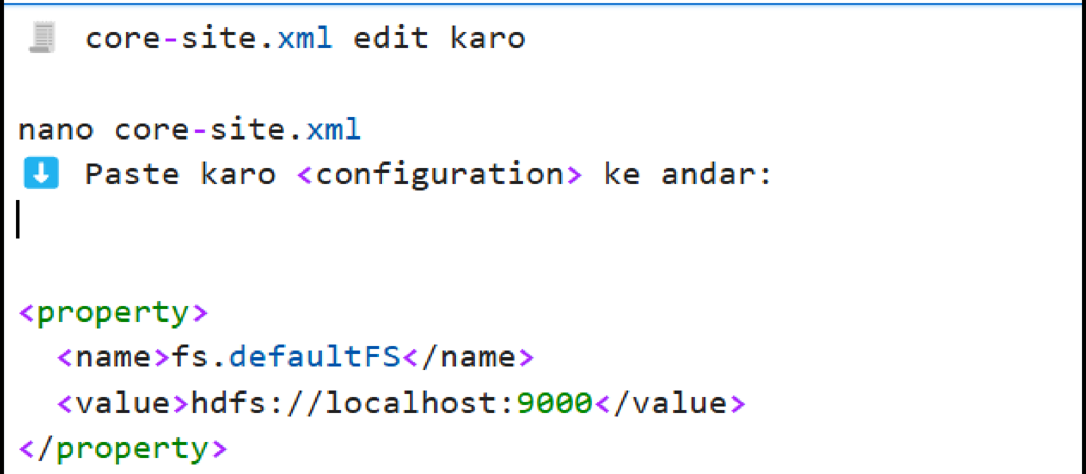


```
nano ~/.bashrc                          # Bash config file open karo

⬇️ Iske end mein yeh lines paste karo:


export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop

source ~/.bashrc                        # Changes ko activate karo
```

🔰 STEP 5: Hadoop Config Files setup karo



```
cd $HADOOP_HOME/etc/hadoop              # Hadoop config folder mein jao
```

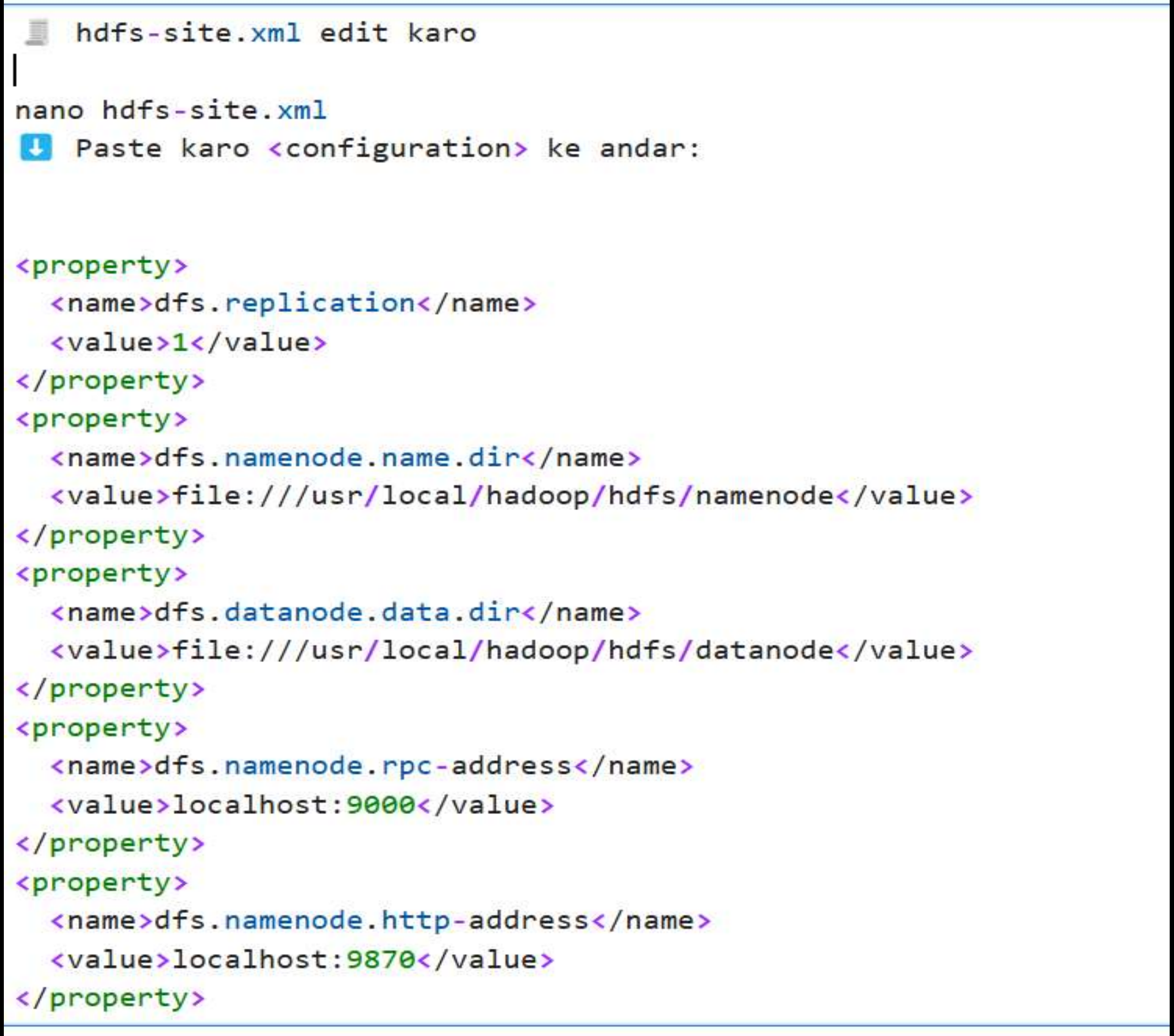


📄 core-site.xml edit karo

```
nano core-site.xml
```

⬇️ Paste karo <configuration> ke andar:

```
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://localhost:9000</value>
</property>
```

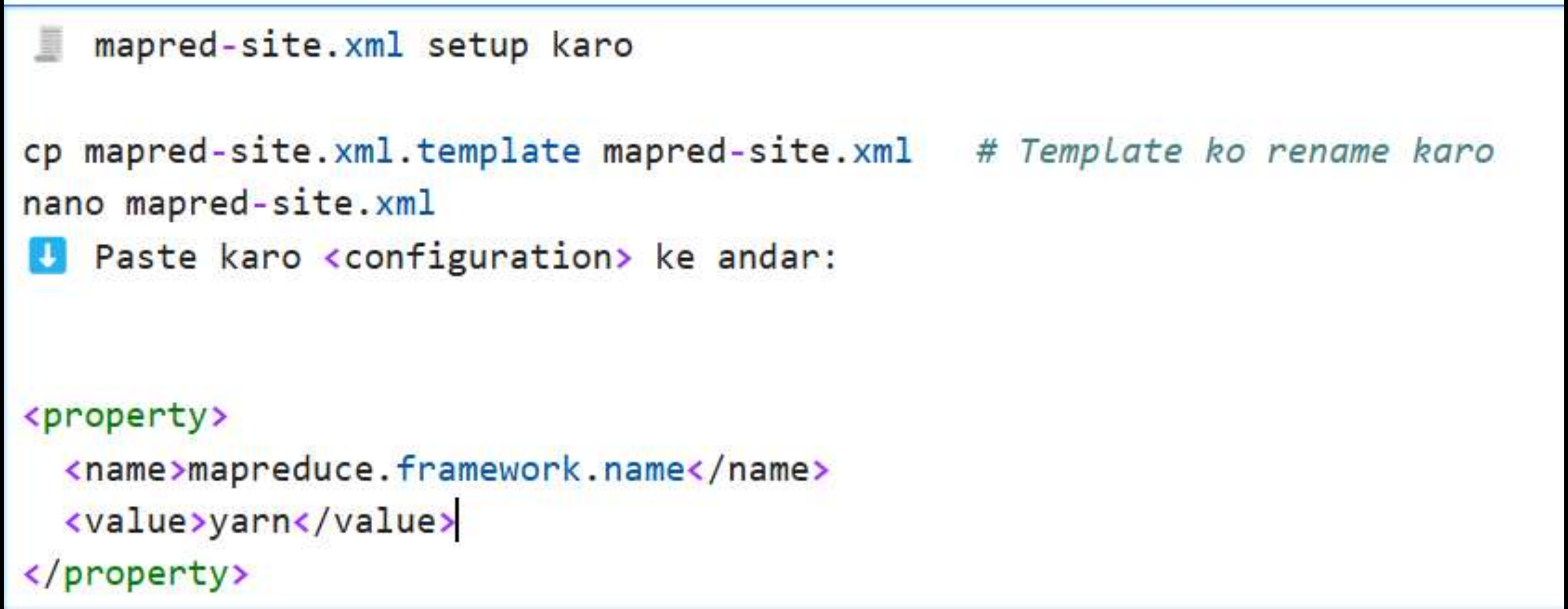


📄 hdfs-site.xml edit karo

```
nano hdfs-site.xml
```

⬇️ Paste karo <configuration> ke andar:

```
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:///usr/local/hadoop/hdfs/namenode</value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:///usr/local/hadoop/hdfs/datanode</value>
</property>
<property>
  <name>dfs.namenode.rpc-address</name>
  <value>localhost:9000</value>
</property>
<property>
  <name>dfs.namenode.http-address</name>
  <value>localhost:9870</value>
</property>
```

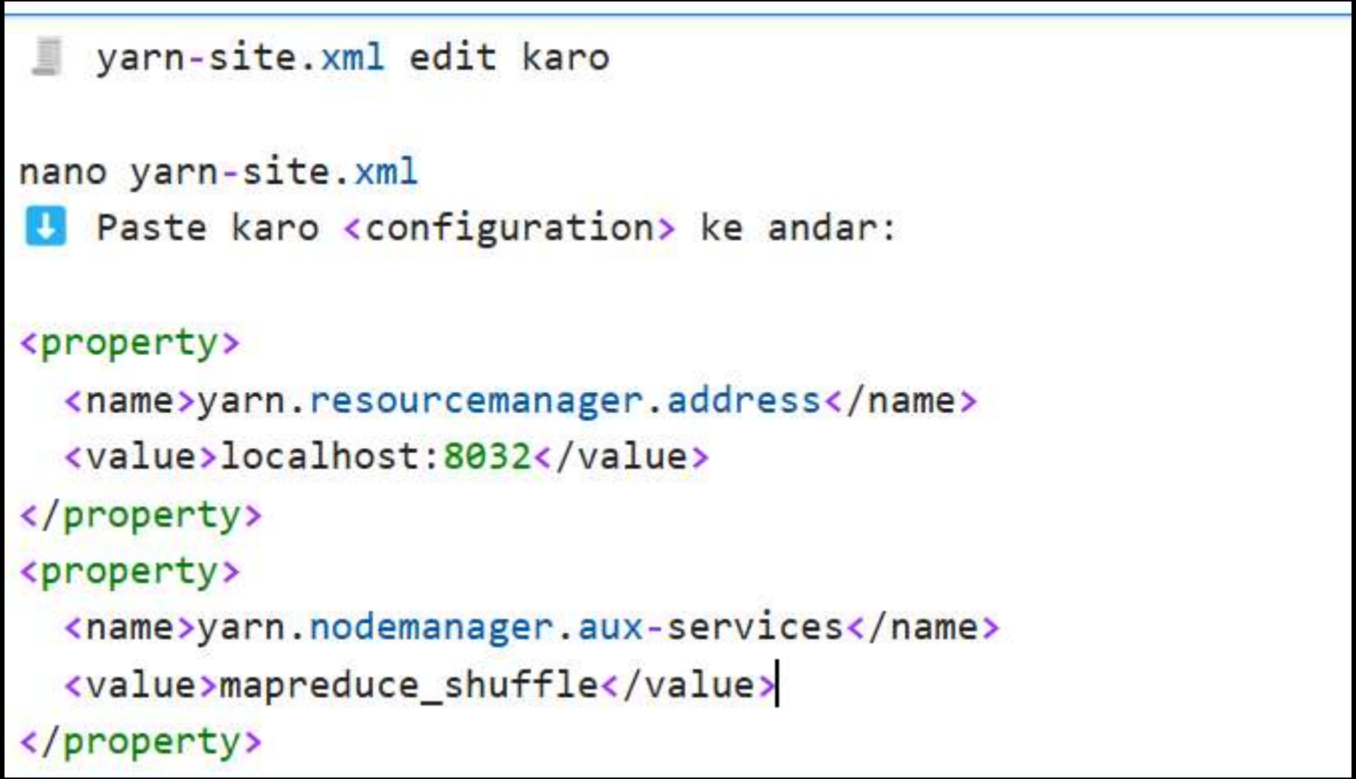


📄 mapred-site.xml setup karo

```
cp mapred-site.xml.template mapred-site.xml   # Template ko rename karo
nano mapred-site.xml
```

⬇️ Paste karo <configuration> ke andar:

```
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
```

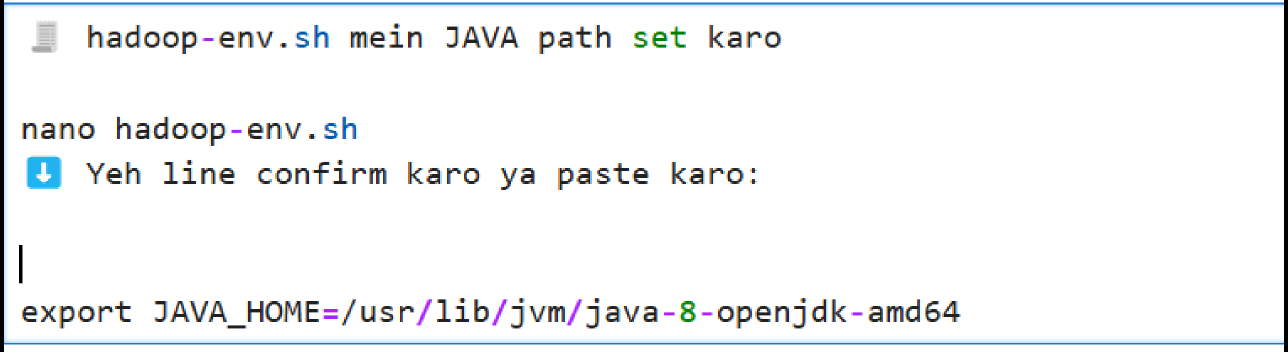


🗒️ yarn-site.xml edit karo

```
nano yarn-site.xml
```

⬇️ Paste karo <configuration> ke andar:

```
<property>
  <name>yarn.resourcemanager.address</name>
  <value>localhost:8032</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
```

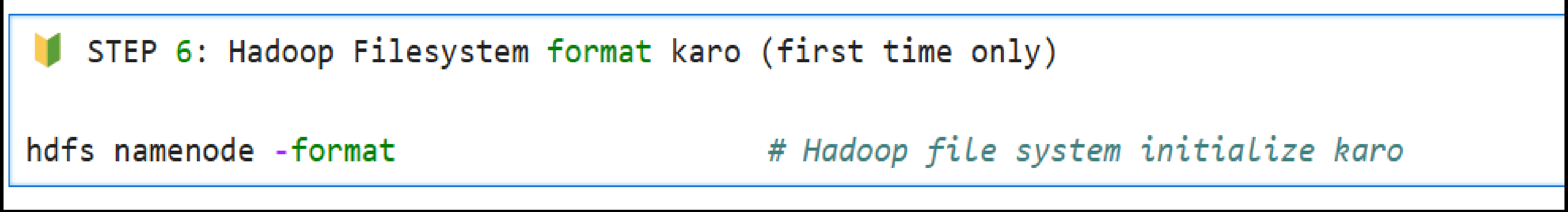


🧾 hadoop-env.sh mein JAVA path set karo

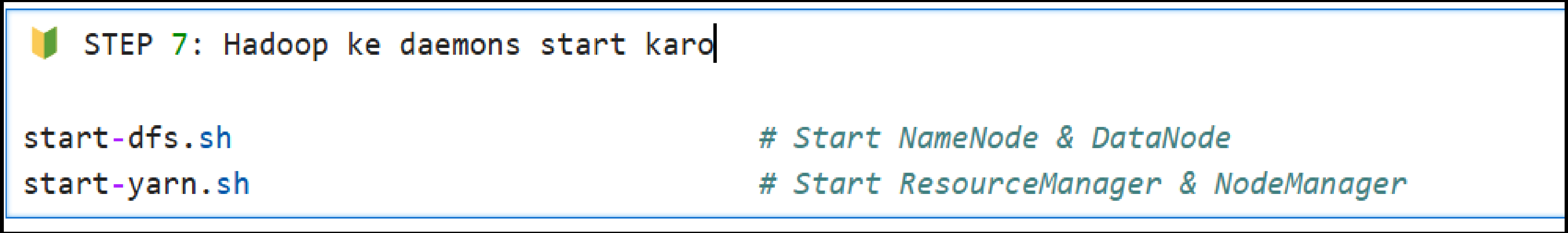
```
nano hadoop-env.sh
```

⬇️ Yeh line confirm karo ya paste karo:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

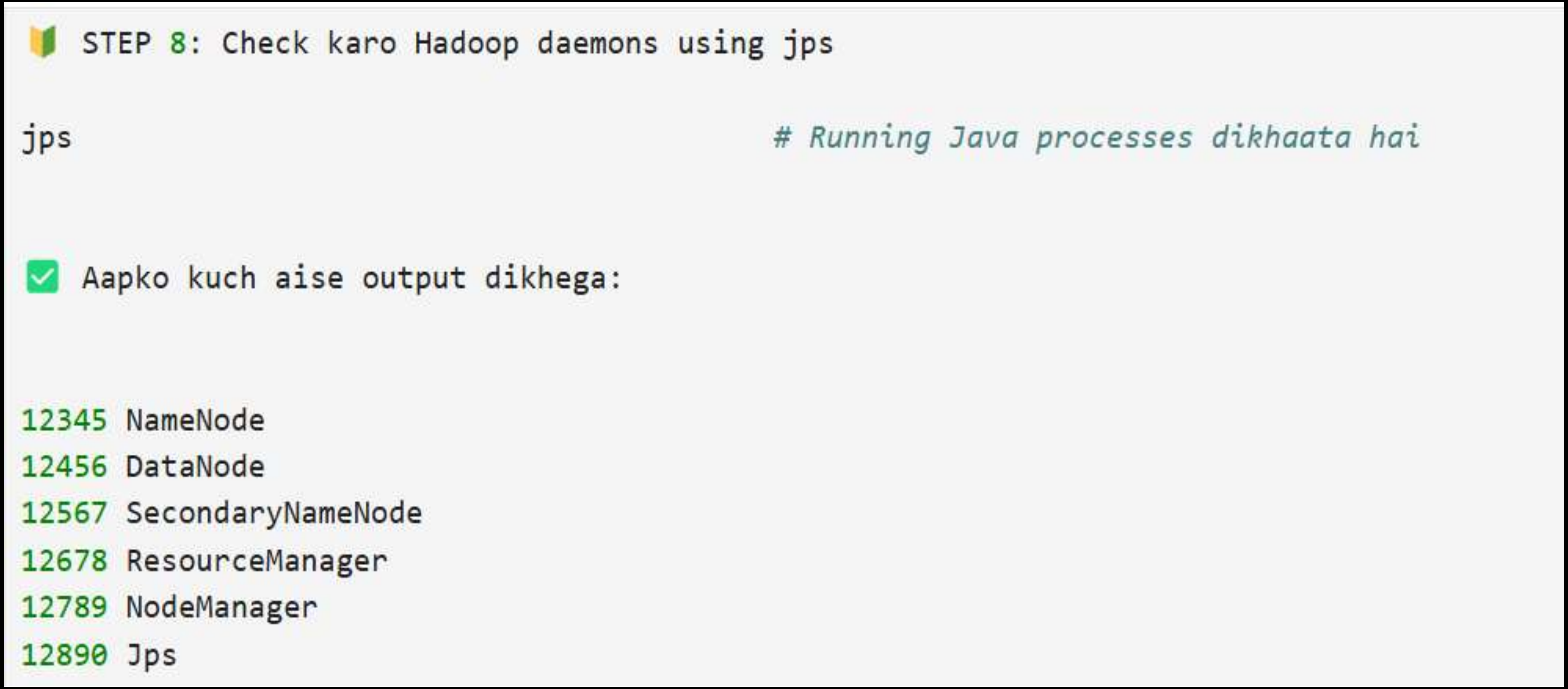


```
🔰 STEP 6: Hadoop Filesystem format karo (first time only)

hdfs namenode -format                     # Hadoop file system initialize karo
```



🔰 STEP 7: Hadoop ke daemons start karo

```
start-dfs.sh                              # Start NameNode & DataNode
start-yarn.sh                             # Start ResourceManager & NodeManager
```



🔰 STEP 8: Check karo Hadoop daemons using jps

```
jps                                        # Running Java processes dikhaata hai
```

✅ Aapko kuch aise output dikhega:

```
12345 NameNode
12456 DataNode
12567 SecondaryNameNode
12678 ResourceManager
12789 NodeManager
12890 Jps
```

Process Name	Explanation
NameNode	Master node of HDFS — file system ka control karta hai
DataNode	Data store karne wala node
SecondaryNameNode	Metadata ka backup rakhta hai
ResourceManager	YARN ka master — job scheduling karta hai
NodeManager	Job execution ke liye responsible
Jps	Bas yeh command khud ko show karta hai