import pandas as pd

csv = pd.read\_csv("/content/Heart\_Disease\_.csv")

print(csv)

csv.dtypes

csv.describe(include='object')

csv.tail()

csv.columns

print(csv.isnull()) # Count missing values

csv.info()

csv.head()

csv.info()

#Findout missing value in rows

missing\_rows = csv[csv.isnull().any(axis=1)]

print("\nRows with Missing Values:")

print(missing\_rows)

#Findout missing value in column

missing\_columns = csv.columns[csv.isnull().any()].tolist()

display("Columns with missing values:")

missing\_columns

# Remove rows with missing values

csv\_drop=csv.dropna()

csv\_drop

csv = pd.read\_csv('Heart\_Disease\_.csv')

csv\_fillna=csv.fillna(0)

csv\_fillna

#Forward Filling the value

csv = pd.read\_csv('Heart\_Disease\_.csv')

csv\_methfor=csv.fillna(method="ffill")

csv\_methfor

#Backward filling the values

csv = pd.read\_csv('Heart\_Disease\_.csv')

csv\_methbac=csv.fillna(method="bfill")

csv\_methbac

#duplicate rows

csv = pd.read\_csv('Heart\_Disease\_.csv')

duplicates = csv[csv.duplicated()]

print(duplicates)

#count duplicate

duplicate\_counts = csv.duplicated().sum()

print("Number of Duplicate Rows:", duplicate\_counts)

#Remove duplicate

csv\_no\_duplicates = csv.drop\_duplicates()

print("\nDataFrame After Removing Duplicates:")

print(csv\_no\_duplicates)

# Select Using Label-Based Indexing (loc)

print(csv.loc[2, "StressLevel"])

# Select an entire row

print(csv.loc[5])

print(csv.iloc[5]) # Selects the second row (index 1)

# Select multiple rows and columns

print(csv.loc[[1, 5], ["SmokingStatus", "StressLevel"]])

print(csv.iloc[1:3]) # Selects rows with index 1 and 2 (excluding 3)

print(csv.iloc[:, 12]) # Selects all rows but only column at index 1 (sex)

print(csv.iloc[2, 0]) # Gets the value at row index 2 and column index 0

print(csv.iloc[0:5, 1:5]) # Selects first two rows and columns at index 1 and 2

csv[csv["ChestPainType"] > 1]

csv.sort\_values(by="ChestPainType") #Sort by a column.

csv.sort\_index()

#deleting first four

csv\_drop = csv.drop(csv.index[:4])

csv\_drop

#Syntax of Lambda Function

#lambda arguments: expression

#arguments â†’ Input variables (can be multiple, separated by commas)

#expression â†’ Single line of code that is executed and returned

add = lambda x, y: x + y

print(add(5, 3))

#using map fun

numbers = [1, 2, 3, 4, 5]

squared = list(map(lambda x: x\*\*2, numbers))

print(squared)

"""map(function, iterable)

function â†’ A function (can be a built-in function, lambda function, or user-defined function)

iterable â†’ A sequence (list, tuple, etc.)

"""

numbers = ["1", "2", "3", "4"]

converted = list(map(int, numbers)) # Convert strings to integers

print(converted)

list1 = [1, 2, 3]

list2 = [4, 5, 6]

summed = list(map(lambda x, y: x + y, list1, list2))

print(summed)

words = ["hello","ddd"]

upper\_words = list(map(str.upper, words))

print(upper\_words) # Output: ['HELLO', 'WORLD', 'PYTHON']

def categorize\_cholesterol(value):

if value < 200:

return "Normal"

elif 200 <= value < 240:

return "Borderline High"

else:

return "High"

csv["Cholesterol\_Level"] = csv["Cholesterol"].apply(categorize\_cholesterol)

print(csv)

csv = pd.read\_csv('Heart\_Disease\_.csv')

def new\_cholesterol\_category(value):

if value < 180:

return "Low"

elif 180 <= value < 220:

return "Normal"

elif 220 <= value < 250:

return "Borderline High"

else:

return "High"

# Update the same column with new values

csv["Cholesterol\_Level"] = csv["Cholesterol"].apply(new\_cholesterol\_category)

print(csv)

csv["Cardio\_Index"] = (csv["Cholesterol"] \* 0.6) + (csv["RestingBP"] \* 0.4)

print(csv)

"""\*\*Using df.groupby("column") in a Heart Disease Dataset\*\*

The groupby() function in pandas groups data based on a column and allows us to compute statistics like mean, sum, count, min, max, etc.

"""

csv = pd.read\_csv('Heart\_Disease\_.csv')

grouped = csv.groupby("StressLevel").mean()

print(grouped)

csv\_grp=csv.groupby("SmokingStatus").count()

print(csv\_grp)

pivot\_df = csv.pivot\_table(index='Sex', columns='MaxHeartRate', values='BMI', aggfunc='mean')

print(pivot\_df)

csv = pd.read\_csv("/content/Heart\_Disease\_.csv")

csv1 = pd.read\_csv("/content/Heart Prediction Quantum Dataset (1).csv")

csv1

csv\_concat=pd.concat([csv,csv1])

csv\_concat

csv.merge(csv1, on='Cholesterol', how='inner')

csv\_merge=csv.merge(csv1, on='Cholesterol', how='outer')

csv\_merge

duplicates =csv\_merge[csv\_merge.duplicated()]

print(duplicates)

duplicate\_counts = csv\_merge.duplicated().sum()

print("Number of Duplicate Rows:", duplicate\_counts)

"""Left - Dataset 1(Eg: Age-left)

Right - Dataset 2(Eg:Age\_Right)

"""

csv\_join=csv.join(csv1, lsuffix='\_left', rsuffix='\_right')

csv\_join

csv.merge(csv1, on='Age', how='left')

csv\_half=csv\_concat.fillna(csv\_concat[['Cholesterol','BloodPressure','RestingBP','MaxHeartRate','BMI','HeartRate']].mean())

csv\_half

csv2=csv\_half.fillna(csv\_half[['Sex','ChestPainType','ExerciseInducedAngina','FastingBloodSugar','SmokingStatus','Diabetes','PhysicalActivity','QuantumPatternFeature','Gender',]].median())

csv2

csv2["Cholesterol"].hist() # Histogram

csv2.boxplot(column="Cholesterol") # Box plot