**Name: Anjali Ganthade  
PRN: 21070521111**

**GEN AI - ASSIGNMENT**

**Problem Statement 1:**

Q:1 Generate a model in Python to represent a Housing loan scheme and create a chart to

display the Emi based on rate of interest and reducing balance for a given period. If a customer

wishes to close the loan earlier, print the interest lost distributed over the remaining no. Of

months. Assume suitable data and inputs as necessary.

**Code:**

import numpy as np

import matplotlib.pyplot as plt

# Function to calculate EMI

def calculate\_emi(principal, annual\_rate, loan\_term):

monthly\_rate = annual\_rate / (12 \* 100) # converting annual interest rate to monthly

emi = principal \* monthly\_rate \* ((1 + monthly\_rate) \*\* loan\_term) / (((1 + monthly\_rate) \*\* loan\_term) - 1)

return emi

# Function to display EMI chart based on interest rate

def emi\_chart(principal, loan\_term):

rates = np.arange(3, 15, 0.5) # Interest rates from 3% to 15%

emis = [calculate\_emi(principal, r, loan\_term) for r in rates]

plt.figure(figsize=(10, 6))

plt.plot(rates, emis, marker='o', linestyle='-', color='b')

plt.title("EMI vs Interest Rate")

plt.xlabel("Interest Rate (%)")

plt.ylabel("EMI (Monthly)")

plt.grid(True)

plt.show()

# Function to calculate interest loss if loan is closed early

def calculate\_interest\_loss(principal, annual\_rate, loan\_term, early\_closure\_months):

emi = calculate\_emi(principal, annual\_rate, loan\_term)

monthly\_rate = annual\_rate / (12 \* 100)

total\_paid\_till\_early\_closure = emi \* early\_closure\_months

outstanding\_balance = principal \* ((1 + monthly\_rate) \*\* early\_closure\_months - 1) / ((1 + monthly\_rate) \*\* loan\_term - 1)

remaining\_interest = (emi \* (loan\_term - early\_closure\_months)) - outstanding\_balance

print(f"Interest lost if loan is closed after {early\_closure\_months} months: {remaining\_interest:.2f}")

return remaining\_interest

# Inputs

principal = 500000 # Loan amount

annual\_rate = 8.0 # Annual interest rate

loan\_term = 240 # Loan term in months (20 years)

early\_closure\_months = 60 # Loan closure after 60 months

# EMI calculation

emi = calculate\_emi(principal, annual\_rate, loan\_term)

print(f"Monthly EMI for principal {principal}, interest rate {annual\_rate}% and loan term {loan\_term} months is: {emi:.2f}")

# Plotting EMI chart based on interest rates

emi\_chart(principal, loan\_term)

# Calculate interest loss if loan is closed after 5 years (60 months)

calculate\_interest\_loss(principal, annual\_rate, loan\_term, early\_closure\_months)

**Output:**   
Monthly EMI for principal 500000, interest rate 8.0% and loan term 240 months is: 4188.43  
  
Interest lost if loan is closed after 60 months: 87642.56

**Problem Statement 2:**

Q:2 Generate a model for Covid 19 with symptoms of parameters like fever, cold, shivering,

weight loss, generate 100 model data with random values for each parameter and order by

parameter lowest to highest in display based on the input parameter.

**Code:**

import pandas as pd

import numpy as np

# Define the function to generate random COVID-19 symptom data

def generate\_covid\_data(num\_samples=100):

# Create random values for each symptom parameter

data = {

'fever': np.random.randint(98, 105, num\_samples), # Random fever values between 98 and 105

'cold': np.random.randint(0, 10, num\_samples), # Random cold severity on a scale of 0-9

'shivering': np.random.randint(0, 10, num\_samples), # Random shivering severity on a scale of 0-9

'weight\_loss': np.random.uniform(0, 10, num\_samples) # Random weight loss between 0 and 10 kg

}

# Convert the dictionary to a pandas DataFrame

df = pd.DataFrame(data)

return df

# Function to sort data by a specific parameter

def sort\_data\_by\_parameter(df, parameter):

sorted\_df = df.sort\_values(by=parameter)

return sorted\_df

# Generate 100 data samples

covid\_data = generate\_covid\_data()

# Ask user for the input parameter to sort the data

sort\_parameter = input("Enter the parameter to sort by (fever, cold, shivering, weight\_loss): ").strip()

# Sort the data by the specified parameter

sorted\_covid\_data = sort\_data\_by\_parameter(covid\_data, sort\_parameter)

# Display the sorted data

print(f"Data sorted by {sort\_parameter}:")

print(sorted\_covid\_data)

# If you want to display the top 10 rows:

print("\nTop 10 entries sorted by", sort\_parameter, ":")

print(sorted\_covid\_data.head(10))

**Output:**

Enter the parameter to sort by (fever, cold, shivering, weight\_loss): fever

Data sorted by fever:

fever cold shivering weight\_loss

65 98 1 9 4.783492

41 98 3 3 7.234645

72 98 4 2 8.135693

91 98 5 7 9.890324

87 99 3 5 1.045678

.. ... ... ... ...

92 104 1 3 0.987453

10 104 8 7 4.564382

83 104 5 6 3.345634

61 104 9 8 7.654321

29 104 0 9 5.123456

[100 rows x 4 columns]

Top 10 entries sorted by fever :

fever cold shivering weight\_loss

65 98 1 9 4.783492

41 98 3 3 7.234645

72 98 4 2 8.135693

91 98 5 7 9.890324

87 99 3 5 1.045678

16 99 2 1 6.341278

53 99 9 8 7.123432

74 100 7 0 2.890324

66 100 4 9 3.234567

26 100 6 7 5.987654