Name: Saumya Deshmukh

PRN: 21070521082

Department: CSE

Division: B

Symbiosis Institute of Technology

Department of Applied Science

Generative AI

CA-2

Computer Science and Engineering

Batch 2021-2025

Semester - VII

Q.1 Generate a model in Python for representation of a bank account of type savings and balance along with transactions of deposit and withdrawals and currently create a program to generate 100 accounts with Random balance and transactions for no. of months and no. of transactions with a seed value of amount. Print all 100 accounts with the last balance and organize them by lowest to highest balance.

Code:

```python
import random

# The program uses a seed value to ensure reproducibility, simulates transactions for several months,
# and finally sorts and displays the accounts by their final balance, from lowest to highest.

# Seed value for reproducibility
random.seed(42)

class BankAccount:
    def __init__(self, account_id, balance):
        self.account_id = account_id
        self.balance = balance
        self.transactions = []

    def deposit(self, amount):
        self.balance += amount
        self.transactions.append(f"Deposit: +{amount}")

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            self.transactions.append(f"Withdraw: -{amount}")
        else:
            self.transactions.append(f"Failed withdraw: -{amount} (Insufficient funds)")

    def __str__(self):
        return f"Account {self.account_id} | Balance: ${self.balance:.2f}"
```

Class BankAccount**:**

- Represents a savings bank account with methods for depositing and withdrawing money.
- Keeps track of the balance and the list of transactions.

```python
# Function to simulate random transactions
def simulate_transactions(account, months, max_transactions, max_transaction_amount):
    for _ in range(months):
        for _ in range(random.randint(1, max_transactions)):
            amount = random.uniform(1, max_transaction_amount)
            if random.choice([True, False]):
                account.deposit(amount)
            else:
                account.withdraw(amount)
```

Function simulate_transactions**:**

- Simulates transactions (deposits and withdrawals) for a given number of months and a maximum number of transactions per month.

```
# Generate 100 random bank accounts
def generate_accounts(num_accounts, months, max_transactions, seed_balance):
    accounts = []
    for i in range(num_accounts):
        initial_balance = random.uniform(0, seed_balance)
        account = BankAccount(account_id=i+1, balance=initial_balance)
        simulate_transactions(account, months, max_transactions, seed_balance/2)  # Max transaction is half of seed balance
        accounts.append(account)
    return accounts
```

Function generate_accounts**:**

- Generates 100 bank accounts with random initial balances and simulates transactions for a specified number of months.

```
# Function to print accounts sorted by final balance
def print_accounts_sorted_by_balance(accounts):
    sorted_accounts = sorted(accounts, key=lambda acc:acc.balance)
    for account in sorted_accounts:
        print(account)

# Configuration
num_accounts = 100
months = 6 # Number of months to simulate transactions
max_transactions = 10 # Max number of transactions per month
seed_balance = 1000 # Seed value for balance and transactions

# Generate and display the accounts
accounts = generate_accounts(num_accounts, months, max_transactions, seed_balance)
print_accounts_sorted_by_balance(accounts)
```

Function print_accounts_sorted_by_balance**:**

- Prints the list of accounts sorted by their final balance (from lowest to highest).

Output (Generated Data):

There are 100 such rows of the data showing the final account balance of every account in an ascending order

```
C:\Users\LOQ\OneDrive\Desktop\Python_Practice>python -u "c:\Users\LOQ\OneDrive\Desktop\Python_Practice\GenAI_Problem1.py"
Account 19 | Balance: $21.14
Account 13 | Balance: $46.55
Account 10 | Balance: $58.12
Account 66 | Balance: $66.83
Account 94 | Balance: $71.87
Account 6 | Balance: $95.66
Account 99 | Balance: $115.00
Account 91 | Balance: $237.18
Account 96 | Balance: $245.69
Account 15 | Balance: $266.78
Account 16 | Balance: $268.37
Account 41 | Balance: $276.46
Account 72 | Balance: $278.69
Account 59 | Balance: $292.29
Account 9 | Balance: $304.56
Account 69 | Balance: $329.15
Account 8 | Balance: $337.86
```

Q.3 Generate a model for an Insurance company to hold information on the insurer's vehicle, and create a chart of monthly, yearly, and quarterly premiums based on no. of years of insurance where in each year, the value of the vehicle depreciates by 7%.

Code:

```python
import matplotlib.pyplot as plt

# Class to represent the insurer's vehicle
class VehicleInsurance:
    def __init__(self, vehicle_id, initial_value, years_of_insurance, base_premium_rate):
        self.vehicle_id = vehicle_id
        self.initial_value = initial_value
        self.years_of_insurance = years_of_insurance
        self.base_premium_rate = base_premium_rate
        self.vehicle_value = initial_value
        self.premiums = {}

    def calculate_depreciation(self):
        """Calculates the vehicle value depreciation over the years"""
        for year in range(1, self.years_of_insurance + 1):
            self.vehicle_value *= 0.93  # Depreciates by 7% each year

    def calculate_premiums(self):
        """Calculates monthly, quarterly, and yearly premiums based on the current vehicle value"""
        self.calculate_depreciation()
        self.premiums['yearly'] = self.vehicle_value * self.base_premium_rate
        self.premiums['quarterly'] = self.premiums['yearly'] / 4
        self.premiums['monthly'] = self.premiums['yearly'] / 12

    def display_premiums(self):
        """Displays calculated premiums"""
        print(f"Vehicle {self.vehicle_id} | Initial Value: ${self.initial_value:.2f} | Depreciated Value: ${self.vehicle_value:.2f}")
        print(f"Yearly Premium: ${self.premiums['yearly']:.2f}")
        print(f"Quarterly Premium: ${self.premiums['quarterly']:.2f}")
        print(f"Monthly Premium: ${self.premiums['monthly']:.2f}")
```

Class VehicleInsurance**:**

- This class models a vehicle with attributes for the vehicle ID, initial value, number of years of insurance, base premium rate, and a dictionary for storing premium values (monthly, quarterly, yearly).
- It has methods to calculate the depreciation (7% per year), calculate the premiums based on the depreciated value, display the premiums for the vehicle.

```python
# Function to generate vehicle data and calculate premiums
def generate_vehicle_data(num_vehicles, initial_values, years_of_insurance, base_premium_rate):
    vehicles = []
    for i in range(1, num_vehicles + 1):
        initial_value = initial_values[i - 1]  # Get initial value from the list
        vehicle = VehicleInsurance(vehicle_id=i, initial_value=initial_value,
                                   years_of_insurance=years_of_insurance, base_premium_rate=base_premium_rate)
        vehicle.calculate_premiums()
        vehicles.append(vehicle)
    return vehicles
```

Function to generate the vehicle data generate_vehicle_data**:**

- Generates data for multiple vehicles, but for simplicity, it's generating one vehicle here.

```python
# Function to plot premium charts for multiple vehicles
def plot_premiums(vehicles):
    years = list(range(1, vehicles[0].years_of_insurance + 1))

    plt.figure(figsize=(12, 8))

    for vehicle in vehicles:
        yearly_premiums = []
        current_value = vehicle.initial_value

        for year in years:
            current_value *= 0.93  # Depreciate by 7% each year
            yearly_premium = current_value * vehicle.base_premium_rate
            yearly_premiums.append(yearly_premium)

        plt.plot(years, yearly_premiums, label=f"Vehicle {vehicle.vehicle_id} (${vehicle.initial_value:.2f})", marker='o')

    # Plot settings
    plt.title(f"Yearly Premiums for {len(vehicles)} Vehicles Over {vehicles[0].years_of_insurance} Years")
    plt.xlabel("Years")
    plt.ylabel("Yearly Premium ($)")
    plt.legend()
    plt.grid(True)
    plt.show()
```

Function to plot the graphs plot_premiums**:**

- Plots a line chart showing how the premiums (monthly, quarterly, and yearly) change over the years as the vehicle value depreciates.

```python
# Example usage
num_vehicles = 5  # Generate data for 5 vehicles
initial_values = [30000, 25000, 20000, 35000, 40000]  # Initial values for each vehicle
years_of_insurance = 5  # Number of years of insurance
base_premium_rate = 0.05  # 5% premium rate of the vehicle's value

# Generate vehicle insurance data
vehicles = generate_vehicle_data(num_vehicles, initial_values, years_of_insurance, base_premium_rate)

# Display the premium data for each vehicle
for vehicle in vehicles:
    vehicle.display_premiums()
    print("-" * 50)

# Plot the premium chart for all vehicles
plot_premiums(vehicles)
```

Depreciation and Premium Calculation:

- The vehicle's value depreciates by 7% per year.
- Premiums are calculated as a percentage of the vehicle's current value after depreciation (5% in this example).
- Monthly and quarterly premiums are derived from the yearly premium.

Output (Generated Data):

The console will display the initial vehicle value, the depreciated value after 5 years, and the calculated premiums (monthly, quarterly, yearly).

Vehicle 1 | Initial Value: $30000.00 | Depreciated Value: $20870.65

Yearly Premium: $1043.53

Quarterly Premium: $260.88

Monthly Premium: $86.96

Vehicle 2 | Initial Value: $25000.00 | Depreciated Value: $17392.21

Yearly Premium: $869.61

Quarterly Premium: $217.40

Monthly Premium: $72.47

Vehicle 3 | Initial Value: $20000.00 | Depreciated Value: $13913.77

Yearly Premium: $695.69

Quarterly Premium: $173.92

Monthly Premium: $57.97

Vehicle 4 | Initial Value: $35000.00 | Depreciated Value: $24349.09

Yearly Premium: $1217.45

Quarterly Premium: $304.36

Monthly Premium: $101.45

Vehicle 5 | Initial Value: $40000.00 | Depreciated Value: $27827.53

Yearly Premium: $1391.38

Quarterly Premium: $347.84

Monthly Premium: $115.95

Output Graph:

The program will also display a chart that shows how the yearly premiums for each vehicle decrease over time as the vehicle value depreciates. Each vehicle is represented by a unique line on the chart, making it easy to compare the premium trends across different vehicles.