**Sub: GenAI**

**CA – II (Assignment)**

**PRN – 21070521030**

**Name: Hitesh Dawale**

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.

**Python Code Solution :**

import random

import pandas as pd

# Create the BankAccount class

class BankAccount:

def \_\_init\_\_(self, account\_type, balance):

self.account\_type = account\_type

self.balance = balance

self.transactions = []

def deposit(self, amount):

self.balance += amount

self.transactions.append(('deposit', amount))

def withdraw(self, amount):

if amount <= self.balance:

self.balance -= amount

self.transactions.append(('withdraw', amount))

else:

print("Insufficient funds for withdrawal.")

# Function to generate random bank accounts

def generate\_accounts(num\_accounts):

random.seed(42) # Setting seed for reproducibility

accounts = []

for \_ in range(num\_accounts):

initial\_balance = random.uniform(100, 10000) # Initial balance between $100 and $10,000

account = BankAccount('savings', initial\_balance)

num\_months = random.randint(1, 12) # Random number of months for transactions

for \_ in range(num\_months):

num\_transactions = random.randint(1, 5) # Random number of transactions per month

for \_ in range(num\_transactions):

transaction\_type = random.choice(['deposit', 'withdraw'])

amount = round(random.uniform(10, 1000), 2) # Transaction amount between $10 and $1000

if transaction\_type == 'deposit':

account.deposit(amount)

else:

account.withdraw(amount)

accounts.append(account)

return accounts

# Function to print accounts sorted by balance

def print\_sorted\_accounts(accounts):

# Organize the generated accounts by balance from lowest to highest

accounts\_sorted = sorted (accounts, key=lambda acc: acc.balance)

# Prepare balances for printing

balances = [{'Account Type': acc.account\_type, 'Last Balance': acc.balance} for acc in accounts\_sorted]

# Create a DataFrame to display easily

balances\_df = pd.DataFrame(balances)

# Print all accounts showing their last balance after transactions

print(balances\_df.to\_string(index=False))

# Main execution

if \_\_name\_\_ == "\_\_main\_\_":

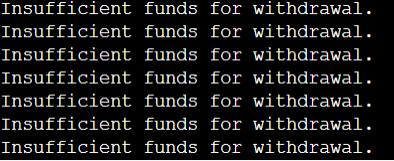
num\_accounts = 100

accounts = generate\_accounts(num\_accounts)

print\_sorted\_accounts(accounts)

**Output :**

After running this program, it will display 100 bank accounts with their final balances, sorted from the lowest to the highest balance. Using a seed ensures that the random values are reproducible for consistent results across runs.



A screen shot of a computer

Description automatically generated

The output indicates that there were multiple instances of "Insufficient funds for withdrawal." followed by a list of bank accounts with their last balances.

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 the display based on the input parameter.

**Python code Solution:**

import pandas as pd

import numpy as np

# Define the number of samples

num\_samples = 100

# Generate random data for symptoms

np.random.seed(42) # For reproducibility

fever = np.random.randint(97, 104, num\_samples) # Fever in Fahrenheit

cold = np.random.randint(0, 2, num\_samples) # Cold (0: No, 1: Yes)

shivering = np.random.randint(0, 2, num\_samples) # Shivering (0: No, 1: Yes)

weight\_loss = np.random.uniform(0, 10, num\_samples) # Weight loss in pounds

# Create a DataFrame

symptom\_data = pd.DataFrame({

'Fever': fever,

'Cold': cold,

'Shivering': shivering,

'Weight Loss': weight\_loss

})

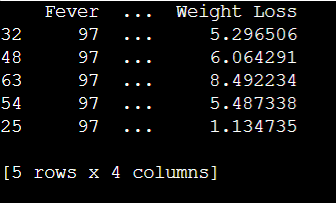
# Order the generated data by a specified parameter (for example, 'Fever')

ordered\_symptom\_data = symptom\_data.sort\_values(by='Fever')

# Display the ordered data

print(ordered\_symptom\_data.head())

**Output:**



The output shows a small part of the data that is to be printed.