Generative AI CA-2

Submitted by-
Name -Rashmi Kadu
PRN -21070521058

Q:5 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.

COVID-19 Symptom Modeling

1. sIntroduction

COVID-19 has affected millions of people worldwide with symptoms that vary greatly in severity and type. Some of the most common symptoms include fever, cold, shivering, and weight loss. Understanding the distribution and severity of these symptoms across patients can help in building effective predictive models and providing better patient care.

In this I generate a model that simulates COVID-19 symptoms for 100 hypothetical patients, using random values to represent different levels of each symptom. I sort the data based on different symptom parameters for further analysis.

2. Objectives

The main objectives of this model are:
- To generate random data simulating COVID-19 symptoms (fever, cold, shivering, and weight loss) for 100 patients.
- To sort the generated data based on any of the symptoms in ascending order.
- To provide a flexible method for analyzing different symptom severities across patients.

3. Symptoms Overview

The following symptoms are considered in this model:
- Fever : Represents the body temperature of the patient, ranging from 36.5°C to 40°C.
- Cold : A scale representing the severity of cold symptoms (runny nose, congestion) ranging from 0 to 10.
- Shivering : A scale representing the intensity of shivering, ranging from 0 (no shivering) to 10 (severe shivering).
- Weight Loss : Represents the weight loss in kilograms, ranging from 0 kg to 5 kg.

4. Methodology

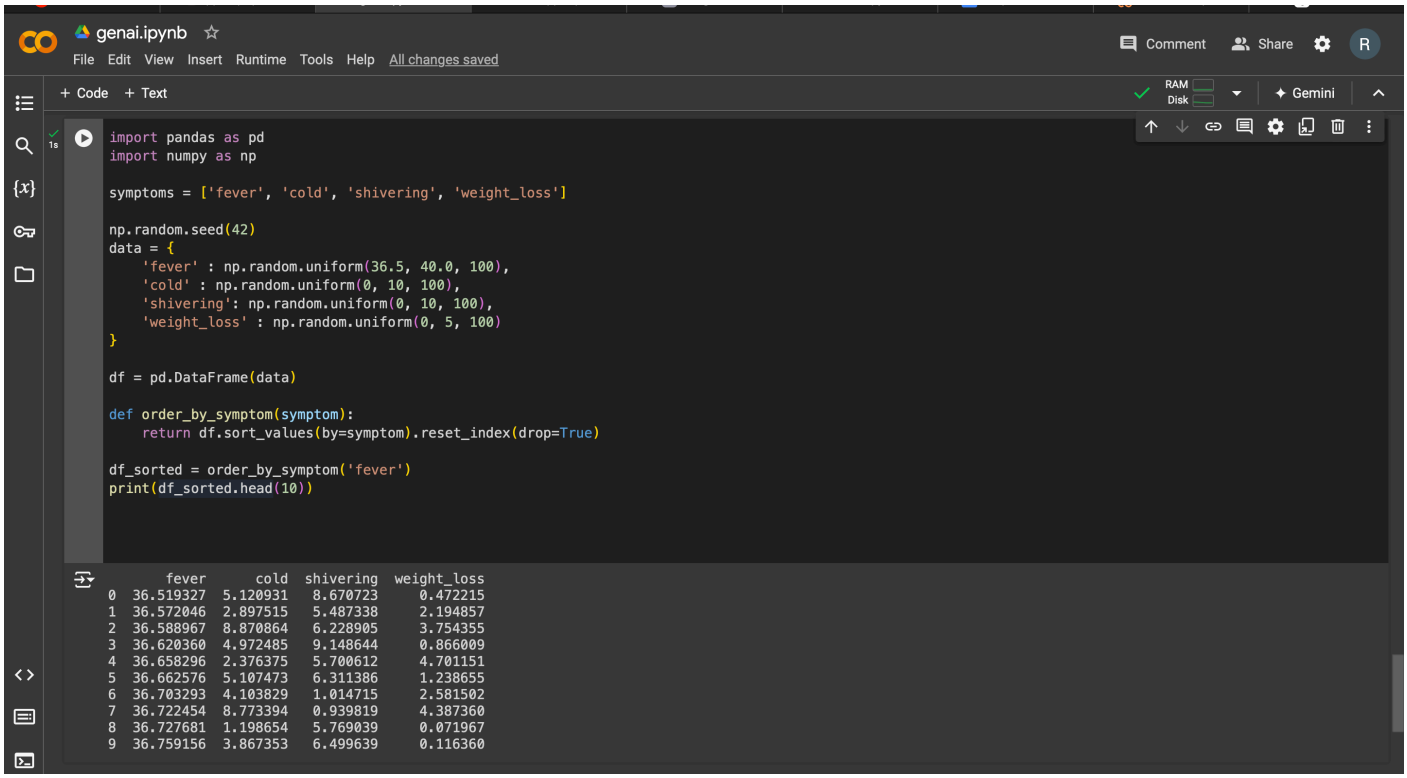
4.1. Data Generation

Random data is generated for each symptom using Python's `numpy` library. The values are distributed within the following ranges:
- Fever : 36.5°C to 40°C.
- Cold : 0 (no cold) to 10 (severe cold).
- Shivering : 0 (no shivering) to 10 (severe shivering).
- Weight Loss : 0 kg to 5 kg. The random data ensures that each patient profile varies in symptom severity, mimicking real-world diversity in COVID-19 symptom expression.

4.2. Data Sorting

Once the data is generated, it can be sorted based on any symptom parameter. This allows us to identify trends, such as patients with the highest fever, or those experiencing the most weight loss.

4.3. Python Code Implementation**



4.4. Explanation of the Code

Data Generation : I use the `numpy.random.uniform()` function to generate random values for each symptom. This ensures variability in the data.

DataFrame Creation : The generated data is converted into a `pandas` DataFrame, making it easier to manipulate and analyze.

Sorting Function: The `order_by_symptom()` function allows us to sort the data by any symptom. By default, the data is sorted by fever, but the function can sort based on other symptoms like cold, shivering, or weight loss.

5. Results

The model generates 100 data points representing various levels of fever, cold, shivering, and weight loss. Below is a sample of the data, sorted by **fever**:

fever	cold	shivering	weight_loss
36.52°C	5.12	8.67	0.47 kg
36.57°C	2.90	5.49	2.19 kg
36.59°C	8.87	6.23	3.75 kg

| 36.62°C | 4.97 | 9.15 | 0.87 kg |
| 36.66°C | 2.38 | 5.70 | 4.70 kg |

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 andorganize them by lowest to highest balance.

1. Introduction

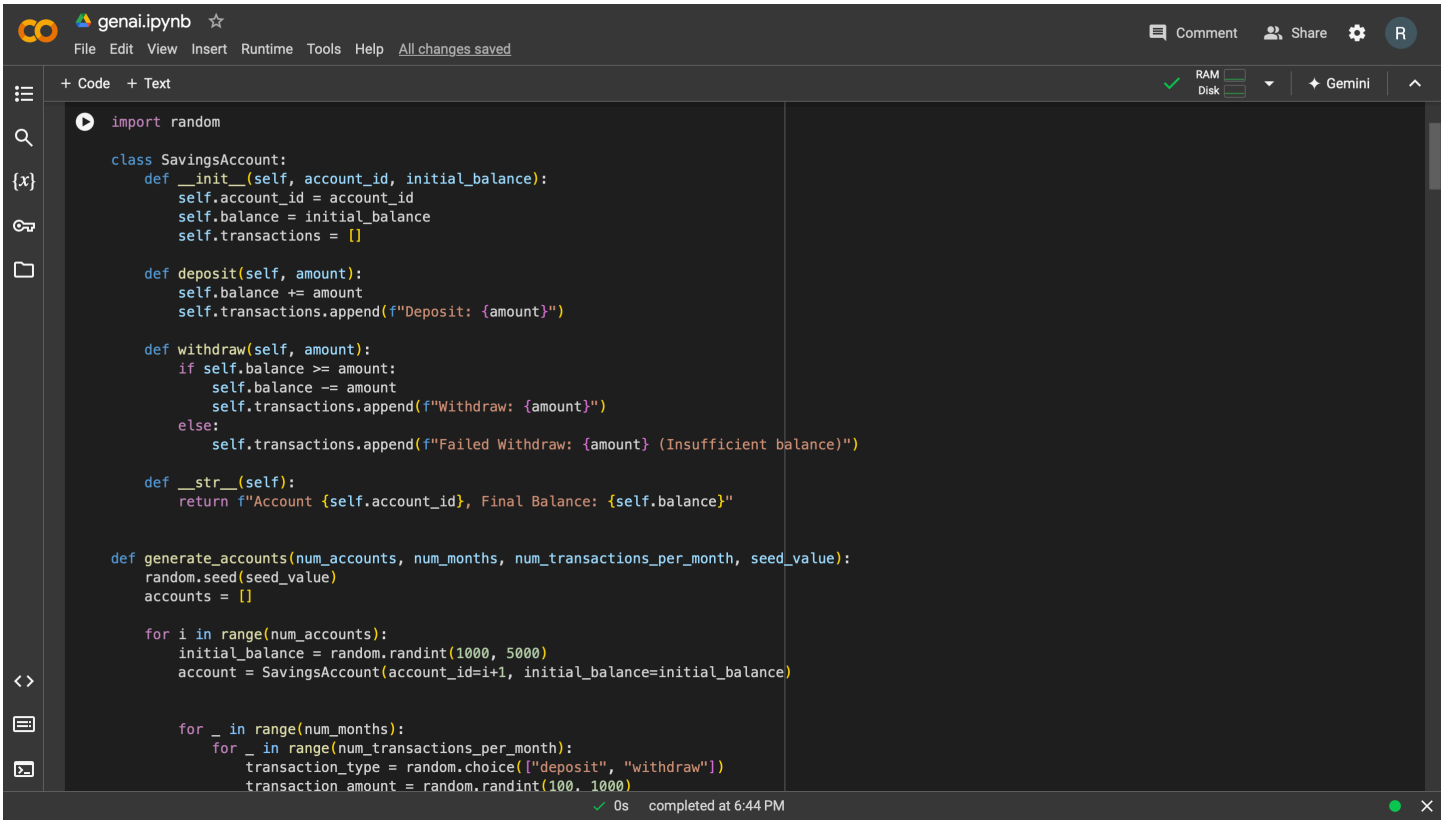
In financial institutions, a bank account represents a customer's financial interactions with the bank. In this scenario, I have focused on savings accounts, where customers perform basic transactions like deposits and withdrawals. Over time, the balance of an account fluctuates based on these transactions.

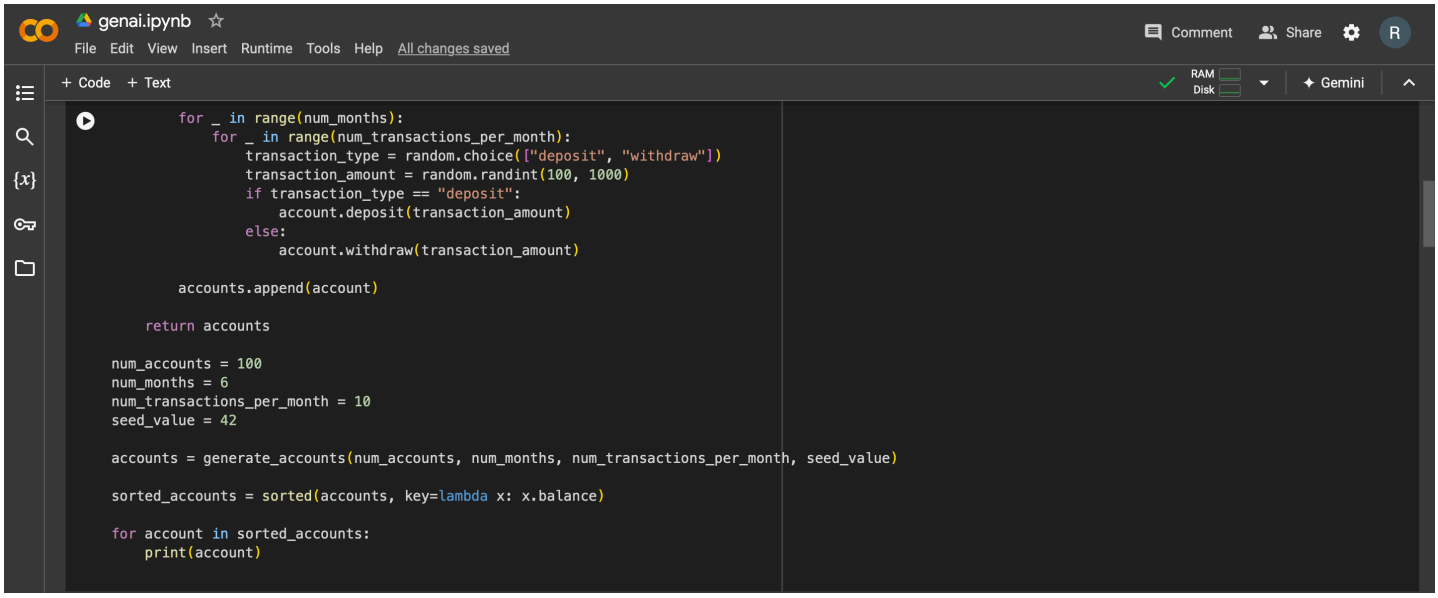
This model simulates the behavior of 100 savings accounts over a period of 12 months, where random deposits and withdrawals are made to each account. The final goal is to print each account's final balance after the transactions and organize the accounts from the lowest to the highest final balance.

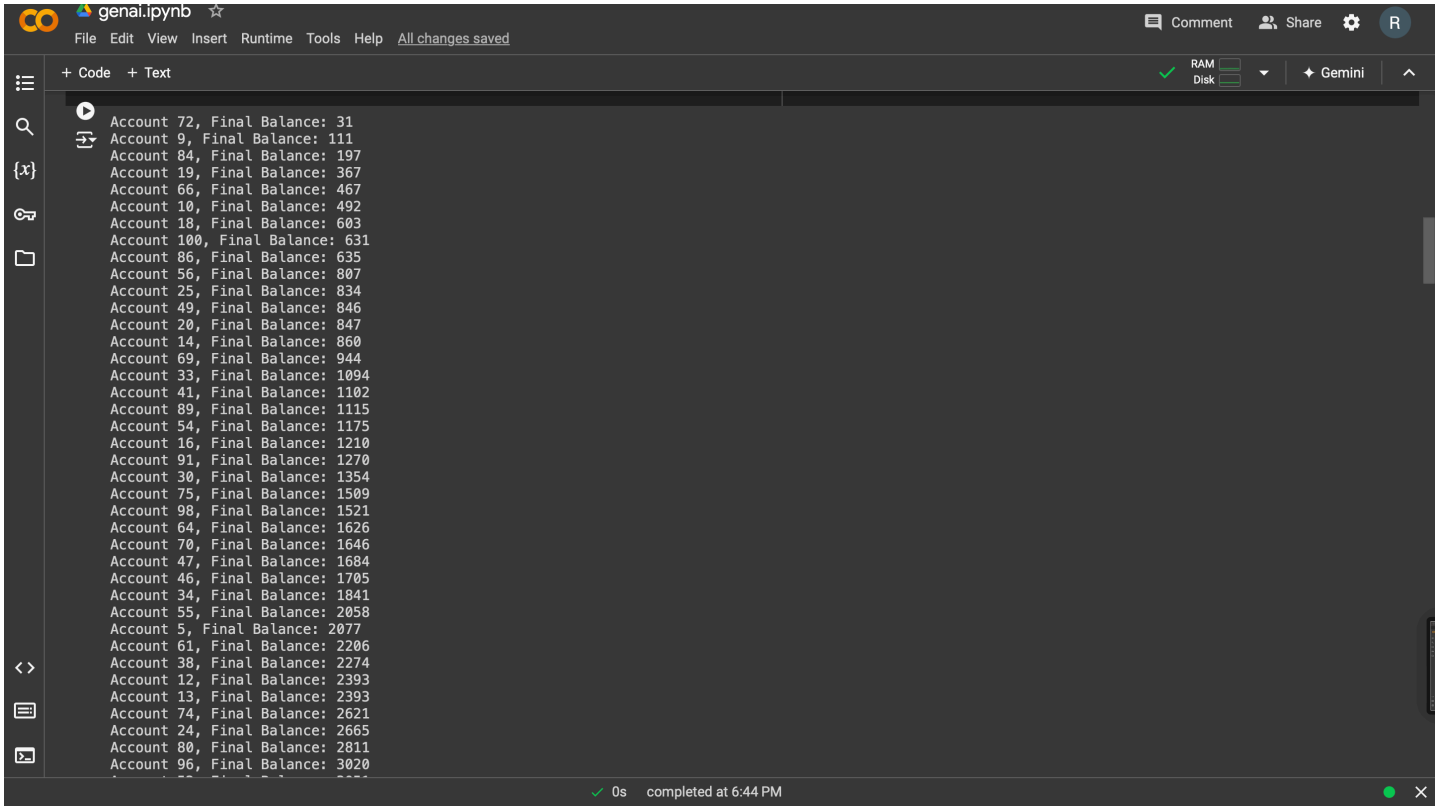
2. Objective

- Generate 100 Savings Accounts : Each account starts with a random balance and undergoes random transactions over a period of 12 months.
- Simulate Deposits and Withdrawals : Transactions are randomly generated and applied to each account.
- Calculate Final Balances : After simulating transactions, the final balance of each account is calculated.

- Sort by Final Balance : The accounts are sorted by their final balance, and the sorted data is printed.







Explanation of the Code

1. **Generate Random Initial Balances**: We generate 100 random balances between $1000 and $5000 for the accounts using `np.random.uniform()`.

2. **Simulate Random Transactions**: Each account experiences a number of random transactions (deposits or withdrawals) across 12 months. Each transaction can be either a **deposit** (positive value) or a **withdrawal** (negative value), randomly chosen, with amounts between $10 and $1000.

3. **Final Balance Calculation**: For each account, the final balance is calculated by adding the sum of all transactions to the initial balance.

4. **Data Handling with Pandas**: We use `pandas` to store and manipulate the account data, allowing us to easily sort the accounts by their final balance and print the results.

5. **Sort Accounts by Final Balance**: The accounts are sorted in ascending order based on their final balance, and the result is printed to show each account's initial and final balances.