Name : Shreyash Rodge

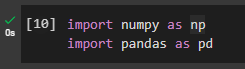
PRN : 21707521096

Subject : Generative AI (CA2)

Section : B (B2)

Problem Statement : 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.

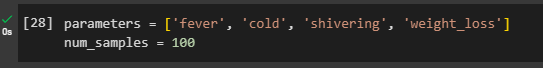
**1. Importing Libraries**



 **numpy (np)**: This library is used for generating random numerical data. In this code, it helps create random values representing symptoms for each parameter.

 **pandas (pd)**: This library is used for creating and managing a tabular dataset (DataFrame), which is a structured way of organizing and manipulating the data.

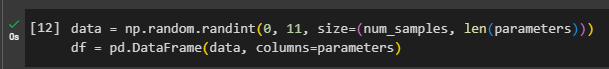
**2. Define Parameters and Number of Samples**



 **parameters**: This list contains four different symptoms (fever, cold, shivering, and weight\_loss), which will be the column names in the dataset.

 **num\_samples**: Specifies that the code will generate 100 records (rows) representing 100 individuals, each having random values for these symptoms.

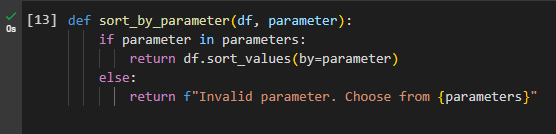
**3. Generating and Structuring Random Symptom Data**



 **Generating Random Data**: The first line creates a 2D array with random integer values between 0 and 10. Each row represents an individual, and each column represents a different symptom, based on the number of samples and parameters specified.

 **Creating a DataFrame**: The second line converts this 2D array into a pandas DataFrame, where each column is labeled according to the symptom names (fever, cold, shivering, weight\_loss), providing a structured and easily accessible format for data analysis.

**4. Sorting Data by a Specified Parameter**



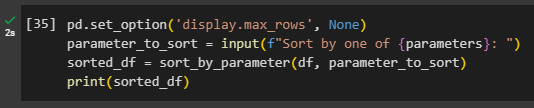
 **Function Definition**: sort\_by\_parameter(df, parameter):

* **Purpose**: To sort the DataFrame (df) by a column specified by the parameter.

 **Parameter Check**:

* **if parameter in parameters**: Checks if the provided parameter is a valid column name in the DataFrame, based on the predefined list of parameters (parameters).
  + **Valid Parameter**: If the parameter is valid, the function sorts the DataFrame by this column in ascending order using df.sort\_values(by=parameter) and returns the sorted DataFrame.
  + **Invalid Parameter**: If the parameter is not found in the list, the function returns an error message indicating the valid parameters that can be chosen.

**5. Configuring Display Options and Sorting Data**



 **Setting Display Options**:

* pd.set\_option('display.max\_rows', None) ensures that all rows of the DataFrame are displayed, preventing truncation in the console output.

 **Prompting User for Input**:

* The input() function asks the user to specify a column (e.g., fever, cold, shivering, weight\_loss) to sort the DataFrame by, and stores this input in parameter\_to\_sort.

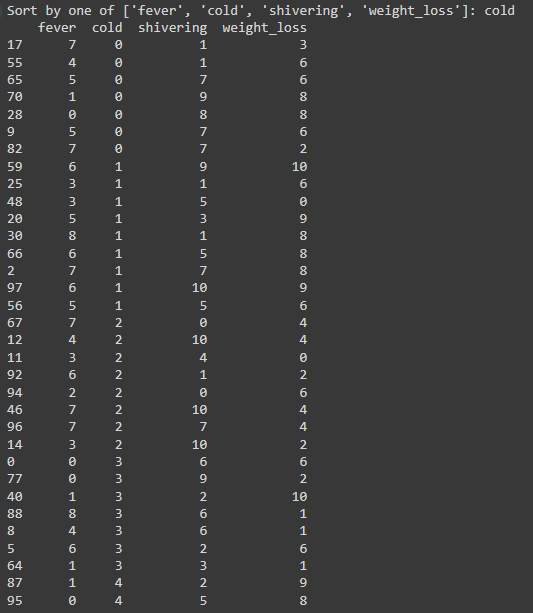
 **Sorting the DataFrame**:

* The sort\_by\_parameter(df, parameter\_to\_sort) function sorts the DataFrame based on the selected column and returns the sorted DataFrame.

 **Displaying the Sorted Data**:

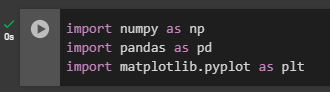
* The sorted DataFrame is printed in full, showing all rows as configured, so users can view the entire sorted dataset.

**5. Configuring Display Options and Sorting Data**



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

**1. Importing Libraries**

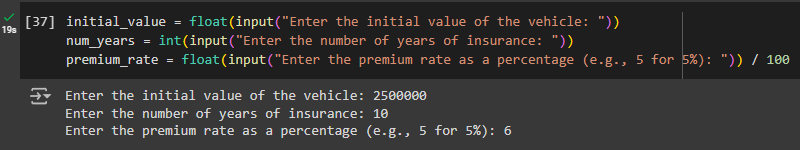


 **numpy (np)**: For numerical operations and generating random data.

 **pandas (pd)**: For data manipulation and creating DataFrames.

 **matplotlib.pyplot (plt)**: For creating visualizations and plots.

**2. User Input for Insurance Model**

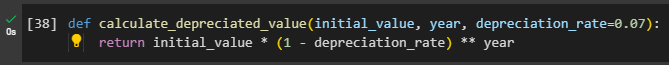


 **initial\_value**: Prompts the user to enter the initial value of the vehicle. This value represents the starting value of the vehicle for calculating depreciation.

 **num\_years**: Asks the user for the number of years the vehicle will be insured. This value is used to determine the duration over which the premiums will be calculated.

 **premium\_rate**: Requests the premium rate as a percentage (e.g., enter 5 for 5%). This rate is converted to a decimal by dividing by 100, and is used to calculate the insurance premiums

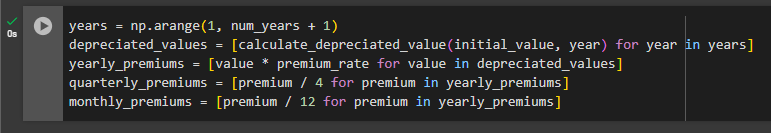
**3. Depreciation Calculation Function**



**calculate\_depreciated\_value**: Computes the value of a vehicle after depreciation over a specified number of years.

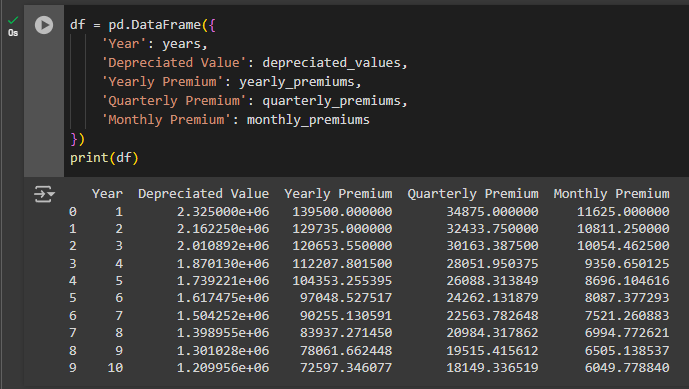
* **Parameters**:
  + **initial\_value**: The vehicle’s initial value.
  + **year**: The number of years over which depreciation is calculated.
  + **depreciation\_rate**: The annual depreciation rate (default is 7% or 0.07).
* **Returns**: The vehicle's value after applying the depreciation formula

**4. Calculating Premiums Based on Depreciation**



* **years**: An array of integers representing each year of the insurance period, from 1 to num\_years.
* **depreciated\_values**: A list of the vehicle’s depreciated values for each year, calculated using the calculate\_depreciated\_value function.
* **yearly\_premiums**: The insurance premiums for each year, computed as the product of the depreciated values and the premium rate.
* **quarterly\_premiums**: The premiums divided by 4 to determine the amount due each quarter.
* **monthly\_premiums**: The premiums divided by 12 to find the monthly premium amount.

**5. Creating and Displaying the Premium DataFrame**

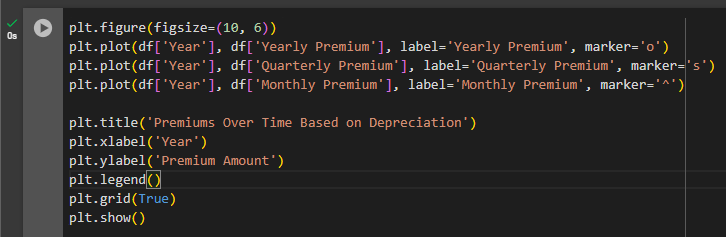


 **df**: A pandas DataFrame that organizes the calculated insurance data into a tabular format.

* **Columns**:
  + **Year**: The year of the insurance period.
  + **Depreciated Value**: The vehicle’s value after depreciation for each year.
  + **Yearly Premium**: The total annual insurance premium.
  + **Quarterly Premium**: The premium amount due each quarter.
  + **Monthly Premium**: The premium amount due each month.

 **print(df)**: Displays the DataFrame, showing the calculated premiums and depreciated values for each year of insurance.

**6. Plotting Premiums Over Time**



 **plt.figure(figsize=(10, 6))**: Sets up the figure for the plot with a size of 10x6 inches.

 **plt.plot(df['Year'], df['Yearly Premium'], label='Yearly Premium', marker='o')**: Plots the yearly premiums as a line with circle markers.

 **plt.plot(df['Year'], df['Quarterly Premium'], label='Quarterly Premium', marker='s')**: Plots the quarterly premiums as a line with square markers.

 **plt.plot(df['Year'], df['Monthly Premium'], label='Monthly Premium', marker='^')**: Plots the monthly premiums as a line with triangle markers.

 **plt.title('Premiums Over Time Based on Depreciation')**: Sets the title of the plot.

 **plt.xlabel('Year')**: Labels the x-axis as 'Year'.

 **plt.ylabel('Premium Amount')**: Labels the y-axis as 'Premium Amount'.

 **plt.legend()**: Adds a legend to the plot to identify each line.

 **plt.grid(True)**: Adds a grid to the plot for better readability.

 **plt.show()**: Displays the plot, showing how premiums change over time based on vehicle depreciation.

**7. Plotted Graph**

