GENERATIVE AI
CA - II

Submitted by : Rohini Bharne (21070521061)

Q3) 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%.

Solution) To model the insurance premiums and vehicle information for an insurer, we need a few key components:

1. **Vehicle Information:**

- Make: Manufacturer of the vehicle.
- Model: Specific model of the vehicle.
- Year: Manufacturing year.
- Initial Value: Initial value of the vehicle.
- Depreciation Rate: 7% per year.

2. **Insurance Premiums:**

- Premium is calculated based on the current value of the vehicle after depreciation.
- Premium rates may depend on the vehicle's current value and other factors (e.g., type of insurance).

3. **Depreciation Formula:**
   The vehicle's value depreciates by 7% per year. This can be calculated as:

$$\text{Value}_{\text{Year}} = \text{Initial Value} \times (1 - \text{Depreciation Rate})^{\text{Year}}$$

Where:
- Value (Year) is the vehicle value after a given number of years.
- Depreciation Rate is 0.07 (7%).

4. **Premium Calculation:**

- The premiums could be a fixed percentage of the current vehicle value. Let's assume a fixed premium rate, say

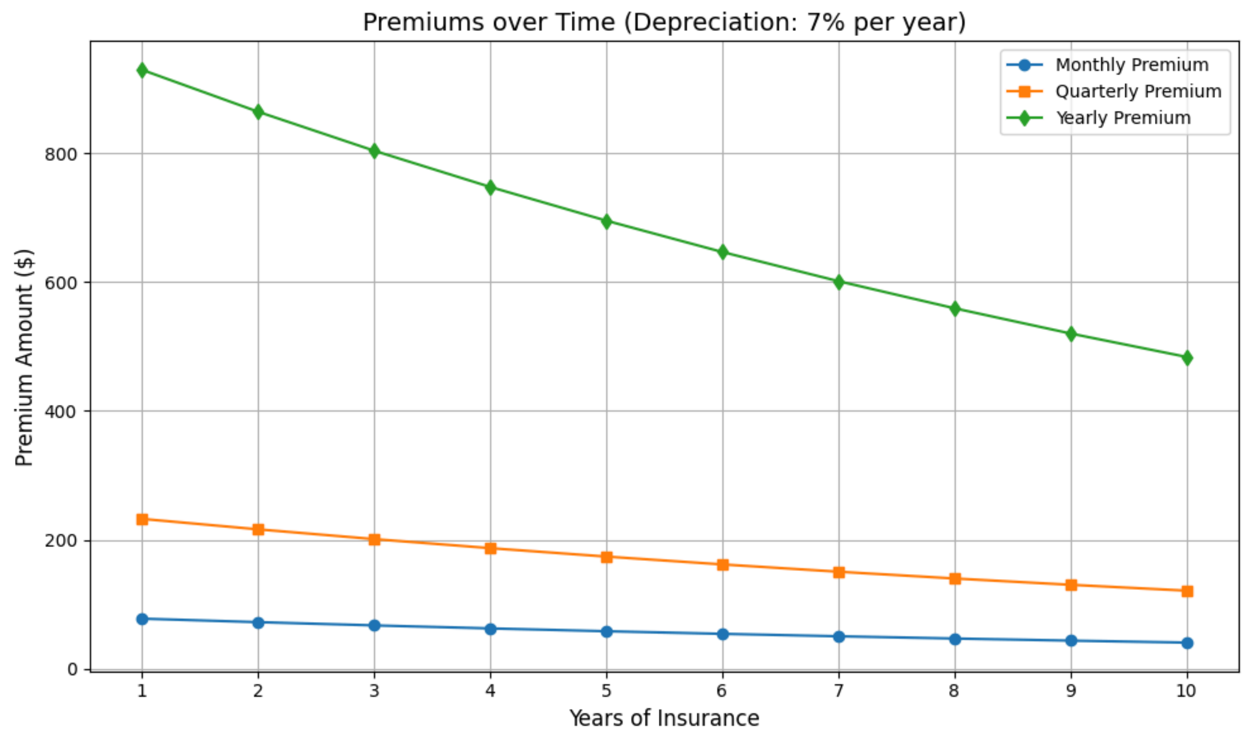
- Monthly Premium: Monthly Premium = Base Premium / 12
- Quarterly Premium: Quarterly Premium = Base Premium / 4
- Yearly Premium: Yearly Premium = Base Premium

Python Implementation:

The creation of the model using these assumptions and generating the chart for premiums. We'll assume an initial vehicle value of $20,000 and a premium rate of 5%. The given below Python script will compute the values and generate the chart:

```python
import numpy as np
import matplotlib.pyplot as plt

# Parameters
initial_value = 20000  # Initial vehicle value in $
depreciation_rate = 0.07  # Depreciation rate of 7% per year
premium_rate = 0.05  # Premium rate as 5% of current vehicle value

# Time period (10 years)
years = np.arange(1, 11)  # 10 years of insurance

# Depreciated value calculation for each year
current_values = initial_value * (1 - depreciation_rate) ** years

# Premium calculations
yearly_premiums = current_values * premium_rate
quarterly_premiums = yearly_premiums / 4
monthly_premiums = yearly_premiums / 12

# Plotting the results
plt.figure(figsize=(10, 6))
plt.plot(years, monthly_premiums, label='Monthly Premium', marker='o')
plt.plot(years, quarterly_premiums, label='Quarterly Premium', marker='s')
plt.plot(years, yearly_premiums, label='Yearly Premium', marker='d')

# Labels and Title
plt.title('Premiums over Time (Depreciation: 7% per year)', fontsize=14)
plt.xlabel('Years of Insurance', fontsize=12)
plt.ylabel('Premium Amount ($)', fontsize=12)
plt.grid(True)
plt.legend()
plt.xticks(years)
plt.tight_layout()

# Show the plot
plt.show()
```

The output:



The chart above illustrates the monthly, quarterly, and yearly insurance premiums over a 10-year period, assuming a 7% annual depreciation rate for the vehicle. As the vehicle depreciates each year, the premiums decrease accordingly since they are calculated based on the current value of the vehicle.

- Yearly Premium: Premium paid once a year.
- Quarterly Premium: Premium paid every 3 months.
- Monthly Premium: Premium paid every month.

This model helps track how premiums decrease over time as the vehicle loses value.

Q5) 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.

Solution) Here, to generate a model for COVID-19 symptoms with random values for parameters such as fever, cold, shivering, and weight loss, you can use a simple script in Python. Here's the python script to achieve this using the `pandas` library to handle the data and `numpy` to generate random values.

```python
import pandas as pd
import numpy as np

# Define the number of records
num_records = 100

# Generate random data for each parameter
np.random.seed(42)  # For reproducibility
data = {
    'Fever': np.random.uniform(98.0, 104.0, num_records),  # Temperature in Fahrenheit
    'Cold': np.random.uniform(0, 10, num_records),  # Severity from 0 to 10
    'Shivering': np.random.uniform(0, 10, num_records),  # Severity from 0 to 10
    'Weight Loss': np.random.uniform(0, 20, num_records)  # Weight loss in pounds
}

# Create a DataFrame
df = pd.DataFrame(data)

# Function to display sorted data
def display_sorted_data(parameter):
    if parameter not in df.columns:
        print(f"Parameter '{parameter}' not found in data.")
        return
    sorted_df = df.sort_values(by=parameter).reset_index(drop=True)
    print(f"Data sorted by '{parameter}':")
    print(sorted_df.head(10))  # Display top 10 rows for brevity

# Example usage: display sorted data by 'Fever'
display_sorted_data('Fever')
```

The Output:

```
Data sorted by 'Fever':
       Fever      Cold  Shivering  Weight Loss
0  98.033133  5.120931   8.670723     1.888859
1  98.123507  2.897515   5.487338     8.779428
2  98.152515  8.870864   6.228905    15.017421
3  98.206331  4.972485   9.148644     3.464037
4  98.271364  2.376375   5.700612    18.804605
5  98.278702  5.107473   6.311386     4.954620
6  98.348502  4.103829   1.014715    10.326007
7  98.381350  8.773394   0.939819    17.549440
8  98.390310  1.198654   5.769039     0.287870
9  98.444268  3.867353   6.499639     0.465439
```

1. **Random Data Generation:**
- `np.random.uniform` is used to generate random values within a specified range for each symptom.
- The `seed` function ensures the randomness is reproducible.

2. **DataFrame Creation:**
- A `pandas` DataFrame is created to store and manipulate the data.

3. **Sorting and Display:**
- The `display_sorted_data` function sorts the DataFrame based on the given parameter and displays the top 10 rows for quick inspection.