PRN: 21070521036

Name: Keyur Dhage

SUBJECT: GENAI CA2

**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 qtrly premiums based on no. of years of insurance

where in each year, the value of the vehicle depreciates by 7%.

Solution🡪

import pandas as pd

import numpy as np

class InsuranceModel:

def init(self, vehicle\_id, initial\_value, years\_of\_insurance, premium\_rate):

self.vehicle\_id = vehicle\_id

self.initial\_value = initial\_value

self.years\_of\_insurance = years\_of\_insurance

self.premium\_rate = premium\_rate

self.depreciation\_rate = 0.07 # Depreciation by 7% annually

self.vehicle\_value\_over\_time = self.calculate\_depreciation()

def calculate\_depreciation(self):

values = []

current\_value = self.initial\_value

for year in range(self.years\_of\_insurance):

values.append(current\_value)

current\_value -= current\_value \* self.depreciation\_rate

return values

def calculate\_premiums(self):

premiums = []

for value in self.vehicle\_value\_over\_time:

yearly\_premium = value \* self.premium\_rate

quarterly\_premium = yearly\_premium / 4

monthly\_premium = yearly\_premium / 12

premiums.append({

'Yearly Premium': yearly\_premium,

'Quarterly Premium': quarterly\_premium,

'Monthly Premium': monthly\_premium

})

return premiums

def generate\_premium\_chart(self):

premiums = self.calculate\_premiums()

years = np.arange(1, self.years\_of\_insurance + 1)

df = pd.DataFrame(premiums, index=years)

df.index.name = "Year"

return df

vehicle\_id = 'V123'

initial\_value = 20000 # Vehicle value in dollars

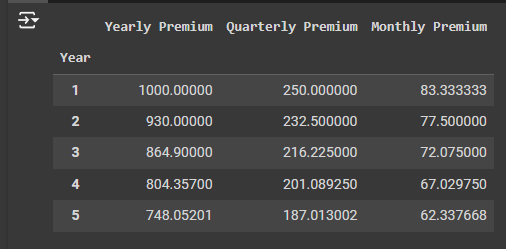
years\_of\_insurance = 5 # Insured for 5 years

premium\_rate = 0.05 # 5% of the vehicle's depreciated value is the premium

insurance\_model = InsuranceModel(vehicle\_id, initial\_value, years\_of\_insurance, premium\_rate)

premium\_chart = insurance\_model.generate\_premium\_chart()

premium\_chart



**EXPLAINATION:**

InsuranceModel Class:

This class holds information about the vehicle (vehicle\_id, initial\_value, years\_of\_insurance, premium\_rate).

It calculates the depreciated value of the vehicle over the insured years using a 7% annual depreciation.

Premium Calculation:

The calculate\_premiums() method calculates the yearly, quarterly, and monthly premiums based on the vehicle's depreciated value.

Chart Generation:

The generate\_premium\_chart() method returns a pandas DataFrame containing the premiums for each year, allowing us to easily visualize the data.

**Q:4** Generate a model to represent interest calculations of a Bank account where the process of

calculating interest for 6 months is a. Find minimum balance for each month b. Make a total of

all minimum balances c. Calculate interest based on interest rate d. Divide interest by 12 to

find one-month interest e. Multiply interest by 6 to show interest in the account. Generate a

model to represent transactions and interest calculations for 6 months.

**Solution🡪**

import pandas as pd

import numpy as np

class BankAccount:

def \_\_init\_\_(self, account\_number, interest\_rate):

self.account\_number = account\_number

self.interest\_rate = interest\_rate # annual interest rate (e.g., 5% as 0.05)

self.transactions = {month: [] for month in range(1, 7)} # dictionary to store transactions for each month

def add\_transaction(self, month, amount):

if month in self.transactions:

self.transactions[month].append(amount)

else:

print(f"Invalid month {month}. Only 1-6 are valid.")

def find\_minimum\_balance(self):

min\_balances = {}

balance = 0

for month in range(1, 7):

monthly\_transactions = self.transactions[month]

monthly\_balances = [balance + sum(monthly\_transactions[:i + 1]) for i in range(len(monthly\_transactions))]

if monthly\_balances:

min\_balances[month] = min(monthly\_balances)

balance = monthly\_balances[-1] # Carry forward the last balance to the next month

else:

min\_balances[month] = balance # No transactions, so balance carries over

return min\_balances

def calculate\_interest(self):

min\_balances = self.find\_minimum\_balance()

total\_min\_balance = sum(min\_balances.values())

yearly\_interest = total\_min\_balance \* self.interest\_rate

monthly\_interest = yearly\_interest / 12

# Step d & e: Calculate interest for 6 months

six\_month\_interest = monthly\_interest \* 6

return six\_month\_interest, min\_balances

def generate\_transaction\_summary(self):

max\_len = max(len(transactions) for transactions in self.transactions.values())

padded\_transactions = {

month: transactions + [np.nan] \* (max\_len - len(transactions))

for month, transactions in self.transactions.items()

}

summary = pd.DataFrame(padded\_transactions)

summary.index.name = 'Transaction\_Index'

return summary.T # Transpose to make months the index

account = BankAccount(account\_number="123456789", interest\_rate=0.05) # 5% annual interest

account.add\_transaction(1, 1000)

account.add\_transaction(1, -200)

account.add\_transaction(1, 300)

account.add\_transaction(2, 1500)

account.add\_transaction(2, -500)

account.add\_transaction(3, 100)

account.add\_transaction(3, -50)

account.add\_transaction(4, 1200)

account.add\_transaction(4, -300)

account.add\_transaction(5, 400)

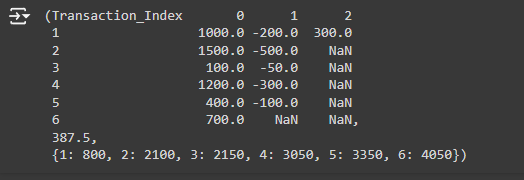
account.add\_transaction(5, -100)

account.add\_transaction(6, 700)

transaction\_summary = account.generate\_transaction\_summary()

six\_month\_interest, min\_balances = account.calculate\_interest()

transaction\_summary, six\_month\_interest, min\_balances



**Explaination:**

BankAccount Class:

The class contains methods for adding transactions (add\_transaction), finding the minimum balance for each month (find\_minimum\_balance), and calculating the interest for 6 months (calculate\_interest).

Transactions:

Transactions are stored month by month (1-6).

The minimum balance for each month is calculated based on the transactions.

Interest Calculation:

Once the minimum balance for each month is determined, the total minimum balance is calculated.

The yearly interest is computed by multiplying the total balance by the interest rate, then divided by 12 to get monthly interest.

The final interest for 6 months is obtained by multiplying the monthly interest by 6.

Result:

The output will show the transaction summary, the calculated interest for 6 months, and the minimum balances for each month.