###fraud log script

import openai

from openai import OpenAI

import pandas as pd

import re

import os

import random

import yaml

import string

from config import API\_KEY,API\_KEY\_URL,MODEL

yaml\_file=r"C:\Users\Ramya\OneDrive\gen\_AI\file\_storage.yml"

with open(yaml\_file,'r',encoding='utf8') as ymlfile:

cfg=yaml.safe\_load(ymlfile)

input\_folder=cfg['Fraud\_alert\_folder']

client=OpenAI(

api\_key=API\_KEY,

base\_url=API\_KEY\_URL

)

response = client.chat.completions.create(

model=MODEL,

n=1,

messages=[

{"role":"system",

"content":"You are an AI assistant who can find credit card fraud alerts and log."},

{

"role": "user",

"content": """ generate exactly \*\*1000 rows\*\* of synthetic data. Each row must contain 9 values, separated by '|', representing the following columns in exact order:

CardHolderid | CardNumber | Expirationdate | transaction | amtusd | name | Emailid | location | IP | alerts | risk | status

you are creating data for 12 columns.

make sure the result followed the exact column order as i given the above.

1.CardHolderid : make sure CardHolderid is alpha numeric with 5 digits. first 2 digit is Alphabet with captial case

next 3 digit is random number.ensure the fomat like 'CH001','AK239','MK920'

2.CardNumber : make sure CardNumber is masked values example (xxxx-xxxx-xxxx-6789) only last 4 digit visibles

those 4 digit is random number. ensure the fomat like 'xxxx-xxxx-xxxx-3435','xxxx-xxxx-xxxx-0023'

3.Expirationdate : generate random expiration date with the format oF 'MM/YYYY'.ensure the generated month and year(expirationdate) is after the current MM/YYYY(June/2025)

4.transaction : creating random transaction type and make sure the transaction type only realted to bank transaction type like('Online','In store','POS','Wallet','Telephone','ATM','Subscription')

5.amtusd : if the alert is 'high amount' or 'large Amount' - amtusd is between 10000 to 20000.

otherwise create a random float values for amtusd under 1000.

ensure its a both float or interger values and ensure if its a float then after the '.' only 2 or 3 digit is enough.

6.name : make sure your mechand name is company names. ensure the name is in united kingdom (UK).

7.Emailid : Generate random email id.make sure its unique

8.location : location should be UK cities with the format of ('South Stuart, UK','Carlberg, UK')

9.IP : generate random ip id.

10.alerts : generate a random alerts from the list .("Abnormal spending pattern","Compromised card",

"Fake email addresses","Foreign IP","Foreign location","High amount","Incorrect expiration dates",

"Low value foreign","Mismatched billing address","Multiple shipping addresses on one card",

"No shopping history","Suspicious IP address","Suspicious zero amount attempt","High risk merchant")

11.risk : to create a random values. the number range is 0-99.

12.status : create a random status and make sure random status is only related to bank credit alert status like 'Investigating','Resolved','Dismissed','Cleared'. those are only model example. you should create random status related to bank creadit alert status.

important logic:

make sure 1000 rows 12 columns is generated.

if alerts in 'Hight amount' then amtusd is must between 10000 to 20000

if alerts in 'Foreign ip' or 'Foreign location' then location is USA,Africa,UAE,Italy,Finland and Australia cities and not UK cities with the format of ('Lagos, Nigeria','New York, USA')

if alerts in 'Incorrect expiration dates' then the Expiration date should be before the current date(June,2025).

if alerts in 'Low value foreign' then the location should be other than UK cities and amtusd is less than 10.

if alerts in 'Suspicious IP address' then the IP should be from this list (192.0.2.1, 104.135.196.55, 104.248.169.251, 107.189.8.238, 134.238.184.178, 136.226.102.90, 185.220.100.240, 185.220.101.7, 192.42.116.198, 72.217.36.105, 107.189.8.56).

if alerts in 'Suspicious zero amount attempt' then amtusd should be 0.

if alerts in 'High risk merchant' then the name should be from the list (Prestige Poker Network,SkyHigh Betting Corp,Luxury Tech Resale,NextGen Forex Solutions,HighRisk Tech Gadgets,FastTrack Loan Services,Dragon Den Crypto Investments,CrytoVault Traders,Platinum Debt Solutions,Titanic Tours and Travels).

ensure there is no None/empty values and Return the result table using '|' as column separators and without header.and without unnessasy words like 'markdown'.make sure its followed the column order.

here sample output format : CH123 | xxx-xxx-xxx-1234 | etc....

CH212 | xxx-xxx-xxx-2243 | etc.... """

}

]

)

def generate\_alert\_ids():

return [f"AL{str(i).zfill(4)}" for i in range(10000)]

def generate\_trans\_ids():

trans\_ids = set()

while len(trans\_ids) < 10000:

letters = ''.join(random.choices(string.ascii\_uppercase, k=3))

digits = ''.join(random.choices(string.digits, k=3))

trans\_ids.add(letters + digits)

return list(trans\_ids)

# total\_rows\_needed = 1000

all\_rows = []

total\_rows\_needed = 10000

while len(all\_rows) < total\_rows\_needed:

responses=response.choices[0].message

rows = [re.split(r"\s\*\|\s\*", line.strip()) for line in responses.content.strip().splitlines()]

# filter for rows with exactly 10 columns and unique keys

valid\_rows = [row for row in rows if len(row) == 12]

all\_rows.extend(valid\_rows)

print(f"Collected {len(all\_rows)} rows so far...")

# Trim to exact total\_rows\_needed if exceeded

all\_rows = all\_rows[:total\_rows\_needed]

# Create dataframe

df = pd.DataFrame(all\_rows, columns=[

'CardHolderId', 'CardNumber', 'ExpirationDate','TransactionChannel', 'AmountUSD',

'MerchantName','EmailID' ,'Location','IP' ,'FraudAlertReason', 'RiskScore', 'Status'

])

df.insert(0,'AlertID',generate\_alert\_ids())

df.insert(4, 'TransactionID', generate\_trans\_ids())

# Save to Excel

excel\_path = os.path.join(input\_folder,cfg.get("fraud\_alerts\_log"))

df.to\_excel(excel\_path, index=False)

# Print confirmation and preview

print("✅ Excel file created:", excel\_path)

print(df)

###fraud log input separation script

import pandas as pd

import os

import yaml

yaml\_file=r"C:\Users\Ramya\OneDrive\gen\_AI\file\_storage.yml"

with open(yaml\_file,'r',encoding='utf8') as ymlfile:

cfg=yaml.safe\_load(ymlfile)

input\_folder=cfg['Fraud\_alert\_folder']

input\_path = os.path.join(input\_folder,cfg.get("fraud\_alerts\_log"))

alert\_reason = os.path.join(input\_folder,cfg.get("alert\_reason"))

output\_path = os.path.join(input\_folder,cfg.get("fraud\_alerts\_statment"))

alerts\_df=pd.read\_excel(alert\_reason)

alerts\_reason\_dic=pd.Series(alerts\_df['Reasons'].values,index=alerts\_df['Alerts']).to\_dict()

result=[]

df=pd.read\_excel(input\_path)

result=[]

for index,row in df.iterrows():

row\_dict=row.to\_dict()

result.append(f"Instruction: Generate a user friendly explanation of the fraud alert reason given in the Input based on the card number, expiration date, amount, merchant, email id, location and IP address. Input : Card Number is {row\_dict['CardNumber']} , Expiration Date is {row\_dict['ExpirationDate']} , Amount is {row\_dict['AmountUSD']} , Merchant is {row\_dict['MerchantName']} , Email ID is {row\_dict['EmailID']} , Location is {row\_dict['Location']} , IP address is {row\_dict['IP']} , Fraud alert reason is {row\_dict['FraudAlertReason']}. Output : The Trasaction seems suspicious {alerts\_reason\_dic[row\_dict['FraudAlertReason']]}")

dff=pd.DataFrame(result)

dff.to\_excel(output\_path,index=False)

print(f"file saves{output\_path}")

###fraud log statistics

import pandas as pd

import yaml

yaml\_file=r"C:\Users\Ramya\OneDrive\gen\_AI\file\_storage.yml"

with open(yaml\_file,'r',encoding='utf8') as ymlfile:

cfg=yaml.safe\_load(ymlfile)

input\_folder=cfg['Fraud\_alert\_folder']

input\_file = os.path.join(input\_folder,cfg.get("fraud\_alerts\_log"))

output\_file = os.path.join(input\_folder,cfg.get("fraud\_alerts\_analysis.csv"))

df = pd.read\_excel(input\_file)

# Fields to analyze

fields = df.columns

# Prepare frequency lists with unique value count on top

freq\_lists = {}

for field in fields:

freq\_dict = {}

for val in df[field]:

freq\_dict[val] = freq\_dict.get(val, 0) + 1

freq\_list = [f"{k} - {v}" for k, v in freq\_dict.items()]

unique\_count = len(freq\_dict)

freq\_list.insert(0, f"number of unique values - {unique\_count}")

freq\_lists[field] = freq\_list

# Pad all columns to same length with "None-0"

max\_len = max(len(lst) for lst in freq\_lists.values())

for field in freq\_lists:

while len(freq\_lists[field]) < max\_len:

freq\_lists[field].append("")

# Create DataFrame and save

analysis = pd.DataFrame(freq\_lists)

analysis.to\_csv(output\_file, index=False)