###Customer onboarding feedback script for 3 categories

import pandas as pd

import csv

import re

import json

from openai import OpenAI

from configure import MODEL,KEY,URL

output\_feedback\_path=r"C:\Users\Ramya\OneDrive\gen\_AI\customer\_onborading\customer\_onbording\_feedback\_creditcard\_model.xlsx"

output\_json\_path=r"C:\Users\Ramya\OneDrive\gen\_AI\customer\_onborading\customer\_onbording\_feedback\_squad\_creaditcard\_model.json"

client = OpenAI(

    api\_key=KEY,

    base\_url=URL

)

def dict\_category(category):

    match category:

        case 1:

            return '2 to 15',{"Document verification delay" :"Smooth process but took longer than expected","Confusing credit limit determination":"I didn’t understand how my credit limit was set or how to increase it.","Slow activation process ":"It took days after approval to start using my card.","if OnboardingChallenges is Limited clarity on reward points and benefits ":"I wasn’t sure how to earn or redeem rewards effectively.","Complicated application form with redundant fields":"Filling out the form felt repetitive and unnecessarily long.","Poor mobile app onboarding experience":"The app was glitchy and didn’t guide me well through setup.","Lack of instant card number issuance":"I had to wait for the physical card to shop online.","Insufficient guidance on credit score impact ":"I wasn’t warned that multiple credit checks would hurt my score.","Difficulty linking credit card with digital wallets":"I struggled to add my card to Apple Pay and Google Pay.","No option to customize card features ":"I wanted to customize rewards but found no options during onboarding.","Confusing billing cycle and due date explanation":"I didn’t understand when my payments were due or how billing cycles work."}

        case 2:

            return '5 to 20',{"Document verification delay" :"Smooth process but took longer than expected","Confusing eligibility criteria":"I wasn't sure if I even qualified for the mortgage in the first place.","Lack of guidance on required property documents":"I had to call support multiple times to understand what documents were needed.","Incomplete or outdated property valuation tools":"The property valuation tool gave very different figures than my real estate agent.","Inconsistent communication from loan officers":"I kept getting different answers from different people.","Limited digital upload options for large PDF documents":"I had to compress my documents several times — it was frustrating.","Delays in legal verification of property documents":"The legal check took forever and no one updated me.","Poor visibility into application status":"I didn’t know what step I was on or what came next. ","Misalignment of expectations around processing time":"They said it would take a week, but it dragged on for over a month.","Difficulty integrating co-applicant information":"Adding a co-applicant was confusing and required starting over.","Overload of legal/financial jargon ":"The terms were too complex. I had to Google every second sentence."}

        case 3:

            return '1 to 7',{"Document verification delay" :"Smooth process but took longer than expected","Unclear loan-to-value (LTV) ratio guidelines":"I didn’t understand how much of the car cost I could actually finance.","Limited options for used vehicle financing":"I couldn’t tell if used cars were even eligible.","Complicated vehicle document submission":"Uploading RC and insurance papers was more complicated than expected.","No instant approval/pre-approval updates":"I didn’t know if I was approved until days later.","Poor integration with dealership systems":"The dealer had no clue about my loan application status.","Limited support for electric vehicles (EVs)":"I wanted to finance an EV but the form didn’t even have that option.","Frequent glitches in EMI calculator":"The EMI tool kept giving me errors. I had to calculate manually.","Complex co-applicant or guarantor addition process":"Adding a guarantor felt like starting a new application.","No visibility on required insurance integration":"I didn’t know I had to get insurance before loan disbursement.","Repetitive KYC verification steps":"I had to verify my ID twice for the same application!"}

def log\_generation(category,client,total\_records=10,batch\_size=2):

    days\_range,dict\_values=dict\_category(category)

    all\_logs=[]

    used\_ids=set()

    while len(all\_logs)<total\_records:

        current\_batch=min(batch\_size,total\_records-len(all\_logs))

        avoid\_ids=",".join(used\_ids) if used\_ids else "None"

        prompt = f"""you must generate exactly {current\_batch} records of synthetic data.no more , or no less and each row must be on single line.

                each row contain 13 columns,serperated by '|', respresenting the following columns in exact order :

        CustomerID | Name | Age | Address | PhoneNumber | Email | AccountType | OnboardingChallenges | TimeTakenToOnboard(days) | DocumentsUsedForOnboarding | DocumentVerificationProcess | ThirdPartyInvolvedInVerification | CustomerFeedbackOnOnboardingProcess

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

        dict = {dict\_values}

        1.CustomerID : generate random 5 digit alphanumeric value. example : AC231,DG443,KH765.

        2.Name : generate random Names in USA.

        3.Age : generate random age between 18 to 70.

        4.Address : generate random address with the format for door no,street name,city name,state. ensure the address are based on USA. example : 38 Prarie Rose Lane, Arington, TX 76017

        5.PhoneNumber : generate random PhoneNumber.example : 103-342-4535,986-543-2134

        6.Email : generate random Email.

        7.AccountType : generate random AccountType which is used in USA banks.

        8.OnboardingChallenges : generate random values from the dict[key]

        9.TimeTakenToOnboard (days) : generate random number between {days\_range}.

        10.DocumentsUsedForOnboarding : generate random value from the list ('Driver's License, Bank Statement' ,'Passport, Utility Bill').

        11.DocumentVerificationProcess : generate random value from the list ('Automated system check' , 'Manual review by staff').

        12.ThirdPartyInvolvedInVerification : generate random value from the list ('Not Available','Third-party verification service A','Third-party verification service B').

        13.CustomerFeedbackOnOnboardingProcess : value of dict[OnboardingChallenges].

        Instruction:

        make ensure the CustomerID is unique and Do not repeat any CustomerID from this list : {avoid\_ids}

        ensure there is no None/empty values and no extra rows.

        if OnboardingChallenges is 'Document verification delay' then put TimeTakenToOnboard (days) in between (20 to 25).

        return the result table using '|' as column seperators and without header and without unnessary words like 'markdown'.make sure its followed the column order.

        here sample output format : CH123 | James Willaim | 42 | 38 Prarie Rose Lane, Arington, TX 76017 | etc......

                                    AH535 | Mickle Jackson | 35 | 12 Redwood Grove Avenue, Sacramento, CA 95814 | etc .....

    """

        response = client.chat.completions.create(

        model=MODEL,

        n=1,

        messages=[

            {"role":"system",

            "content":"you are a data genertor for customer onboarding feedback."},

            {

                "role":"user",

                "content":prompt}

        ]

        )

        content = response.choices[0].message.content.strip()

        rows = content.splitlines()

        for row in rows:

            fields = re.split(r"\s\*\|\s\*",row.strip())

            if len(fields) !=13:

                continue

            cust\_id=fields[0].strip()

            if cust\_id not in used\_ids:

                used\_ids.add(cust\_id)

                all\_logs.append(fields)

    return all\_logs

def SQUAD\_format(df):

    df.fillna("", inplace=True)

    # Build SQAD-style JSON

    sqad\_data = {

        "version": "1.0",

        "data": []

    }

    for \_, record in df.iterrows():

        customer\_id = record["CustomerID "]

        context = (

            f"CustomerID : {record['CustomerID ']}\n"

            f"Name: {record['Name']}\n"

            f" Age : {record[' Age ']}\n"

            f" Address : {record[' Address ']}\n"

            f" PhoneNumber : {record[' PhoneNumber ']}\n"

            f" Email : {record[' Email ']}\n"

            f" AccountType : {record[' AccountType ']}\n"

            f" OnboardingChallenges : {record[' OnboardingChallenges ']}\n"

            f" TimeTakenToOnboard(days) : {record[' TimeTakenToOnboard(days) ']}\n"

            f" DocumentsUsedForOnboarding : {record[' DocumentsUsedForOnboarding ']}\n"

            f" DocumentVerificationProcess : {record[' DocumentVerificationProcess ']}\n"

            f" ThirdPartyInvolvedInVerification : {record[' ThirdPartyInvolvedInVerification ']}\n"

            f"CustomerFeedbackOnOnboardingProcess: {record['CustomerFeedbackOnOnboardingProcess']}"

        )

        entry = {

            "context": context,

            "qas": create\_qas(record, context, customer\_id)

        }

        sqad\_data["data"].append(entry)

    return sqad\_data

# Function to create QAs

def create\_qas(record, context, customer\_id):

    qa\_fields = [

        ("What is the customer's name?", "Name"),

        ("What is the customer's age?", " Age "),

        ("What is the customer's address?", " Address "),

        ("What is the customer's phone number?", " PhoneNumber "),

        ("What is the customer's email?", " Email "),

        ("What is the customer's account type?", " AccountType "),

        ("What are the onboarding challenges faced by the customer?", " OnboardingChallenges "),

        ("How many days did it take to onboard the customer?", " TimeTakenToOnboard(days) "),

        ("What documents were used for onboarding?", " DocumentsUsedForOnboarding "),

        ("What was the document verification process?", " DocumentVerificationProcess "),

        ("Was a third party involved in the document verification process?", " ThirdPartyInvolvedInVerification "),

        ("What was the customer's feedback on the onboarding process?", "CustomerFeedbackOnOnboardingProcess")

    ]

    qas = []

    for question, field in qa\_fields:

        answer\_text = str(record[field])

        answer\_start = context.find(answer\_text)

        print(field)

        qa = {

            "question": question,

            "id": f"[Our Bank: TATA Bank]{customer\_id}\_{field.lower().replace(' ', '\_').replace('(', '').replace(')', '').replace('-', '\_')}",

            "answers": [{

                "text": f"[Our Bank: TATA Bank]{answer\_text}",

                "answer\_start": answer\_start if answer\_start >= 0 else 0

            }]

        }

        qas.append(qa)

    return qas

def customer\_feedback(category,output\_feedback\_path,output\_json\_path,client):

# store the log data into excel file

    print("Customer onboarding feedback log is generating....")

    data\_log=log\_generation(category,client)

    out\_df= pd.DataFrame(data\_log,columns=['CustomerID ','Name',' Age ',' Address ',' PhoneNumber ',' Email ',' AccountType ',' OnboardingChallenges ',' TimeTakenToOnboard(days) ',' DocumentsUsedForOnboarding ',' DocumentVerificationProcess ',' ThirdPartyInvolvedInVerification ','CustomerFeedbackOnOnboardingProcess'

    ])

    out\_df.to\_excel(output\_feedback\_path,index=False)

    print(f"file saved successfully in {out\_df}")

# store the squad format data into json file

    sqad\_data = SQUAD\_format(out\_df)

    with open(output\_json\_path, 'w', encoding='utf-8') as json\_file:

        json.dump(sqad\_data, json\_file, indent=4, ensure\_ascii=False)

    print(f"Wrote {len(sqad\_data)} SQuAD entries to disk")

category = int(input("""select any one category :

1. Credit Card & Personal Loan

2. Mortgage

3. Vehicle Loan

"""))

customer\_feedback(category,output\_feedback\_path,output\_json\_path,client)