BCIS 5110 - Programming Languages for Business Analytics (FALL 2023)

PROJECT 1

TEAM 15

Ramya Perumalla - 11717649
Sumanth Bhupati - 11702331
Jyoshna Irrigisetti - 11646474
Sai kumar Bantupalli - 11717643
Ramya Sree Vemula - 11711298
Vasavi Rama Gedala - 11711294

Description of the Project:

FarziBank (FB) provides financial services, including credit cards, to its customers. Each customer can possess several credit cards, each with its own credit limit. The bank collects customer data to analyze behaviour and make informed decisions.
Customer Information includes unique Customer ID, Gender, Age, Marital Status, Number of Children, Education Level, Annual Income, and Total Credit Line.
Account Information encompasses Account Number, Account Opening Date, Credit Limit, Annual Fee, and Annual Percentage Rate (APR).
Account Activity involves Purchase Amount, Cash Advance Amount, Balance Amount, and Payment Amount. Payments are due between the 1st and 10th of each month, with a Minimum Amount Due. Late payments result in a Past Due Amount, Interest Charges, Late Payment Fees, and an increment in the Delinquency Counter.

Key points to note:
Customers can have multiple credit cards.
Cards are used for purchases or cash advances.
Daily interest accrues on balances.
Customers are required to repay their debts.
Failure to pay the Minimum Amount Due leads to a Past Due Amount, Late Payment Fee, and an increase in the Delinquency Counter.
Three consecutive missed Minimum Amount Due payments led to account closure.

Approach:

We broke it down into smaller parts to solve this problem and design a Python algorithm.
As described in the provided details. Below, we outlined the steps and provided the Algorithm.

STEP 1: Generating Customers

- Generated a dataset of 20,000 customers, each assigned a unique 7-digit Customer ID.
- Configured customer attributes, including age, gender, marital status, number of children, education level, annual income, number of accounts, and total credit line.
- Determined marital status based on age, adhering to specific likelihoods.
- Calculated the number of children, taking age into account and following given probabilities.
- Specified education levels based on age, mapping them to numeric values.
- Computed annual income using a provided formula.
- Calculated the number of accounts based on marital status and the number of children, with 1 added.
- Calculated the total credit line for each customer based on annual income and the number of accounts.

Step 2 - Account Generation

- Generated individual accounts with unique Account Numbers for each customer.
- Set the Date Opened for each account to a date before January 1, 2022, considering the customer's age.
- Created Account Numbers by adding a unique digit to the Customer ID.
- Determined the Account Credit Line as a random fraction of the customer's Total Credit Line.
- Computed Annual Fees as 1% of the Credit Line.
- Randomly selected the Annual Interest Rate from a range of 15% to 30%.

Step 3 - Account Activity Simulation

- Simulated account activity over 12 months, spanning January 1, 2022, to December 31, 2022.
- Defined intervals at which customers use their cards, choosing between purchases (with a 95% chance) and cash advances (with a 5% chance).
- Generated purchase and cash advance amounts within available credit and cash limits.
- Used the provided interest rate formula to apply daily interest to the account balance.
- Simulated customer payments, including daily payments, periodic payments, and minimum-due payments.

- Calculated end-of-month statistics, such as closing balance, minimum amount due, total purchases, total cash advances, total payments, total interest charges, past due amounts, late payment fees, and delinquency counters.
- Account for account closures if a customer misses the minimum payment for three consecutive months.

ALGORITHM:

This algorithm provides code that generates synthetic customer data, including customer demographics, financial attributes, and account activity. Below is an algorithmic breakdown of the code:

1. Import necessary libraries:
 - 'random' for generating random values.
 - 'datetime' and 'timedelta' from 'datetime module for handling dates.

2. Initialize an empty list to store customer data:
 - customers = []

3. Define the age distribution and marital status distribution:
 - Generate a list 'age_distribution' containing 20,000 random ages between 20 and 80.
 - Create an empty list 'marital_status_distribution to store marital statuses.

 - For each age in 'age_distribution':
  - If the age is between 20 and 30 (inclusive), then randomly select the marital status ('Single' or 'Married') with weights [0.75, 0.25].
  - If age is between 30 and 60 (inclusive), then randomly select marital status with weights [0.25, 0.75].
  - Otherwise, randomly select marital status with equal weights [0.5, 0.5].
  - Append the selected marital status to 'marital_status_distribution'.

4. Define the number of children distribution:
 - Create an empty list 'num_children_distribution' to store the number of children.
 - For each age in age_distribution':
  - If age is between 20 and 40 (inclusive), then randomly select the number of children (0 to 4) with weights [0.4, 0.3, 0.2, 0.1, 0.0].
  - Otherwise, randomly select the number of children with weights [0.1, 0.3, 0.3, 0.2, 0.1].
  - Append the selected number of children to 'num_children_distribution'.

5. Define the education level distribution and mapping:
 - Create an empty list 'education_level_distribution' to store education levels.
 - For each age in 'age_distribution':

- If age is between 20 and 25 (inclusive), then randomly select education level (0 to 4) with weights [0.1, 0.5, 0.3, 0.1, 0.0].
- If age is between 25 and 35 (inclusive), then randomly select education level with weights [0.1, 0.5, 0.3, 0.05, 0.05].
- Otherwise, randomly select education level with weights [0.1, 0.5, 0.25, 0.1, 0.05].
- Append the selected education level to 'education_level_distribution'.

6. Generate customer data for 20,000 customers:
- For each customer (loop variable 'I' from 0 to 19999):
- Generate a random customer ID ('customer_id') between 1,000,000 and 9,999,999.
- Retrieve age, gender, marital status, number of children, and education level from their respective distributions.
- Calculate annual income, number of accounts, and total credit line based on the provided formulas.
- Generate account information for each customer:
- Create an empty list 'accounts' to store account information.
- For each account (loop variable `j` from 0 to 'num_accounts – 1'):
- Generate a random date opened ('date_opened') within the customer's age range.
- Generate an 8-digit account number ('account_number').
- Generate a random credit line ('credit_line') as a proportion of the total credit line.
- Calculate the annual fee ('annual_fee') based on the credit line.
- Generate a random annual interest rate ('annual_interest_rate') between 15% and 30%.
- Create an account dictionary with the generated information.

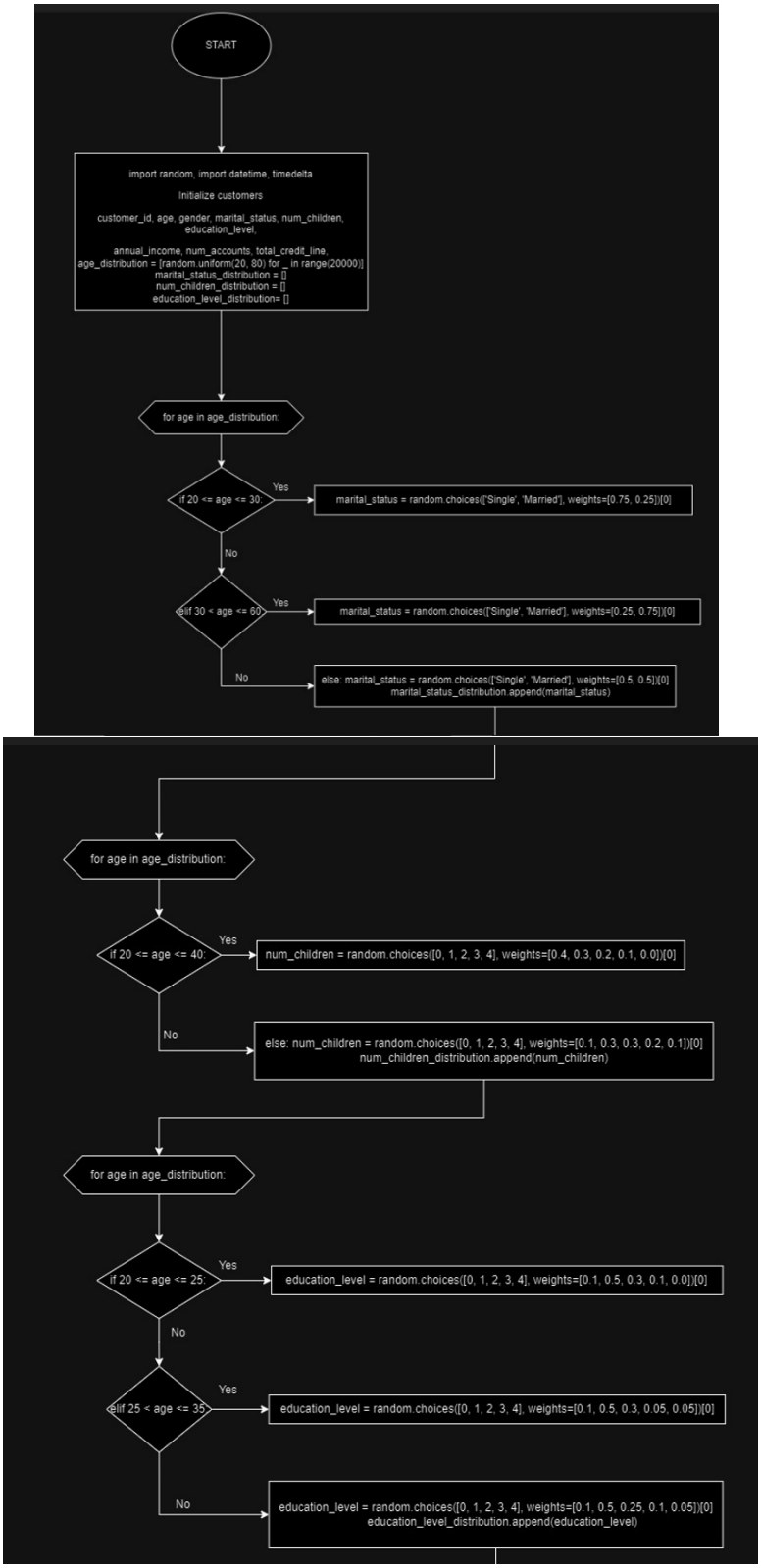
- Simulate account activity for 12 months:
- For each month (loop variable 'month' from 1 to 12):
- Simulate daily account activity for the entire month, including purchases, cash advances, interest calculations, and payments.
- Calculate monthly amounts such as closing balance, minimum amount due, total purchases, total cash advances, total payments, and total interest.
- Store the monthly data in the account dictionary.

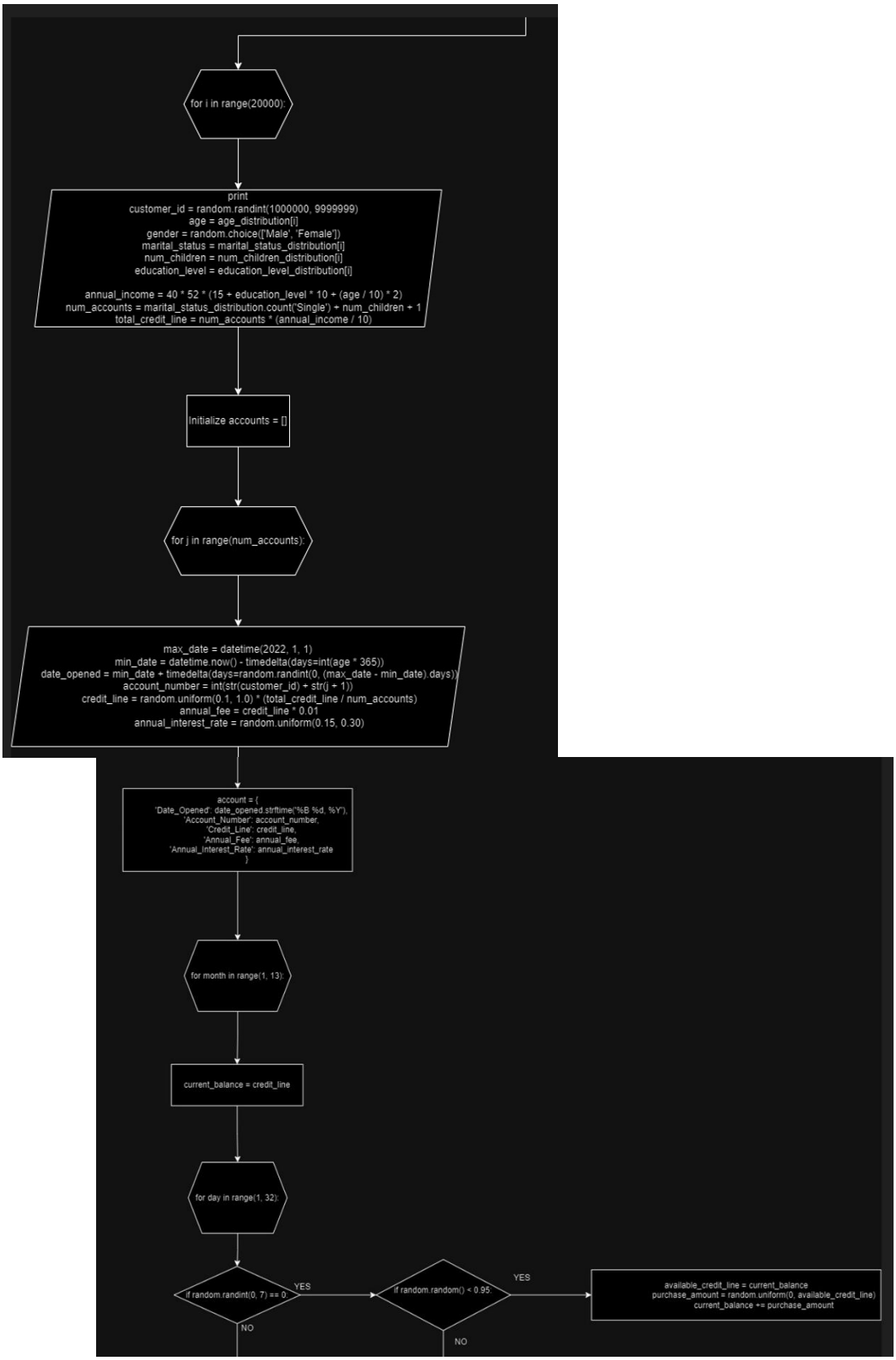
- Append the account dictionary to the 'accounts' list.

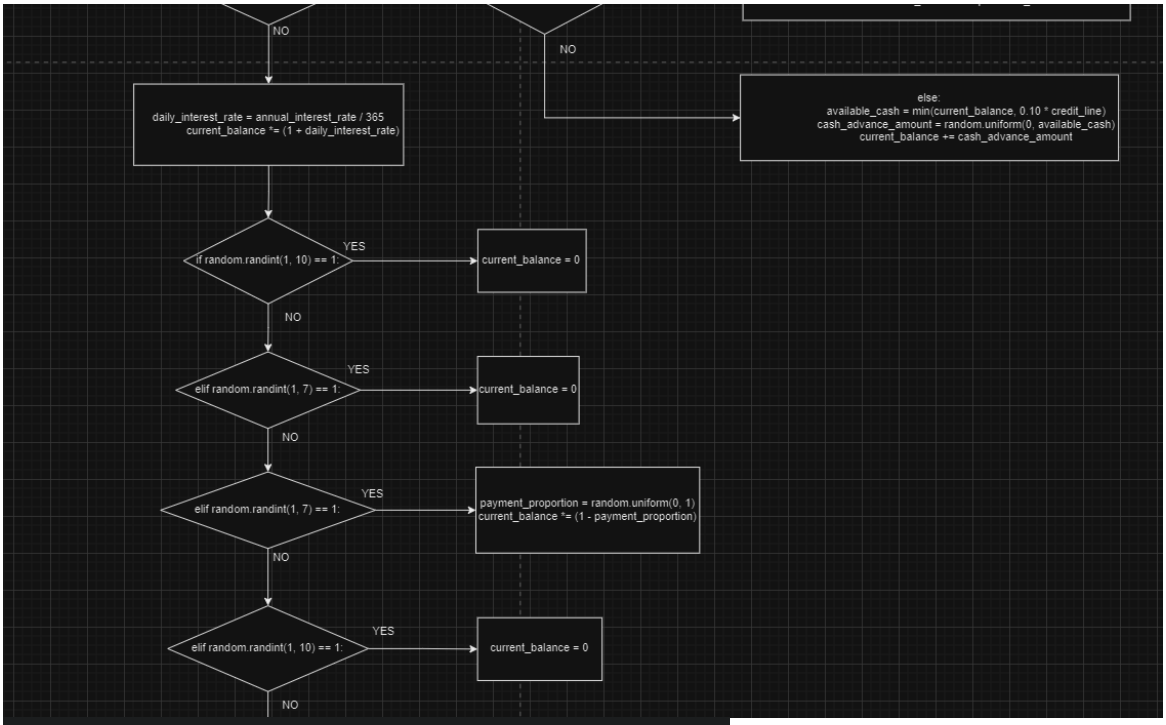
- Create a customer dictionary containing customer demographics and their list of accounts.
- Append the customer dictionary to the 'customers' list.

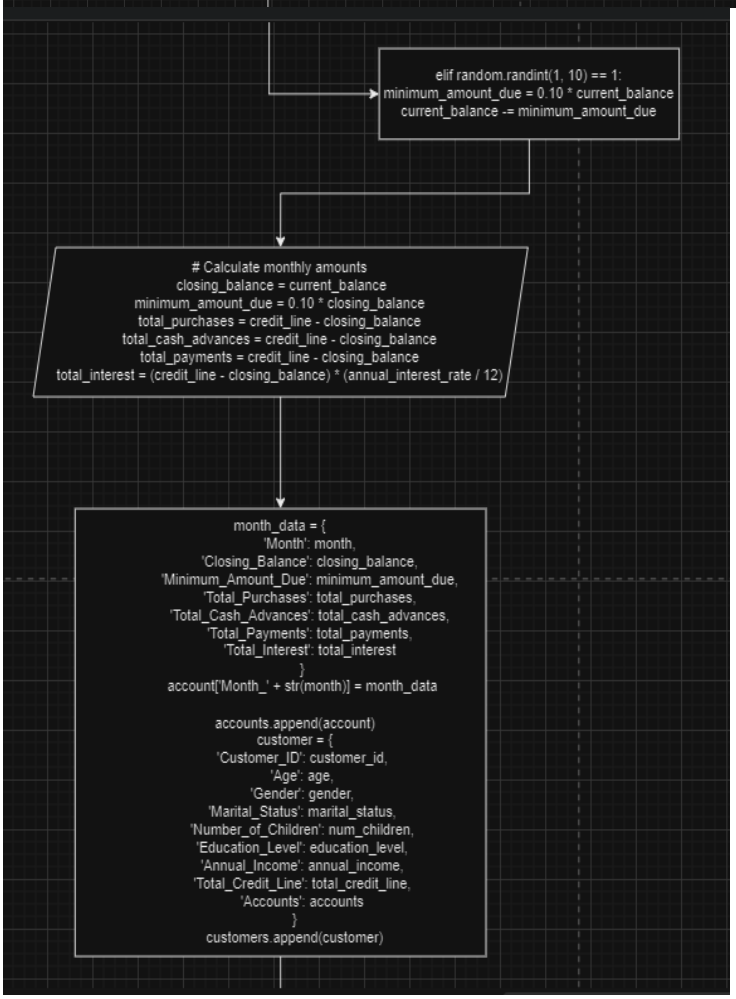
7. End of the code execution.

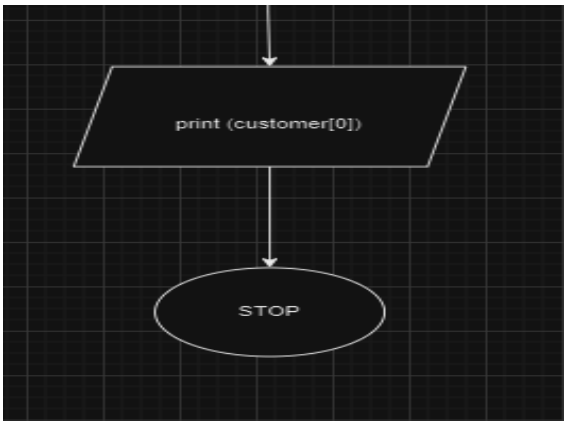
FLOWCHART


START
import random, import datetime, timedelta
Initialize customers
customer_id, age, gender, marital_status, num_children, education_level,
annual_income, num_accounts, total_credit_line,
age_distribution = [random.uniform(20, 80) for _ in range(20000)]
marital_status_distribution = []
num_children_distribution = []
education_level_distribution= []
for age in age_distribution:
if 20 <= age <= 30:
Yes
marital_status = random.choices(['Single', 'Married'], weights=[0.75, 0.25])[0]
No
elif 30 < age <= 60:
Yes
marital_status = random.choices(['Single', 'Married'], weights=[0.25, 0.75])[0]
No
else: marital_status = random.choices(['Single', 'Married'], weights=[0.5, 0.5])[0]
marital_status_distribution.append(marital_status)
for age in age_distribution:
if 20 <= age <= 40:
Yes
num_children = random.choices([0, 1, 2, 3, 4], weights=[0.4, 0.3, 0.2, 0.1, 0.0])[0]
No
else: num_children = random.choices([0, 1, 2, 3, 4], weights=[0.1, 0.3, 0.3, 0.2, 0.1])[0]
num_children_distribution.append(num_children)
for age in age_distribution:
if 20 <= age <= 25:
Yes
education_level = random.choices([0, 1, 2, 3, 4], weights=[0.1, 0.5, 0.3, 0.1, 0.0])[0]
No
elif 25 < age <= 35:
Yes
education_level = random.choices([0, 1, 2, 3, 4], weights=[0.1, 0.5, 0.3, 0.05, 0.05])[0]
No
education_level = random.choices([0, 1, 2, 3, 4], weights=[0.1, 0.5, 0.25, 0.1, 0.05])[0]
education_level_distribution.append(education_level)


for i in range(20000):
print
customer_id = random.randint(1000000, 9999999)
age = age_distribution[i]
gender = random.choice(['Male', 'Female'])
marital_status = marital_status_distribution[i]
num_children = num_children_distribution[i]
education_level = education_level_distribution[i]
annual_income = 40 * 52 * (15 + education_level * 10 + (age / 10) * 2)
num_accounts = marital_status_distribution.count('Single') + num_children + 1
total_credit_line = num_accounts * (annual_income / 10)
Initialize accounts = []
for j in range(num_accounts):
max_date = datetime(2022, 1, 1)
min_date = datetime.now() - timedelta(days=int(age * 365))
date_opened = min_date + timedelta(days=random.randint(0, (max_date - min_date).days))
account_number = int(str(customer_id) + str(j + 1))
credit_line = random.uniform(0.1, 1.0) * (total_credit_line / num_accounts)
annual_fee = credit_line * 0.01
annual_interest_rate = random.uniform(0.15, 0.30)
account = {
'Date_Opened': date_opened.strftime('%B %d, %Y'),
'Account_Number': account_number,
'Credit_Line': credit_line,
'Annual_Fee': annual_fee,
'Annual_Interest_Rate': annual_interest_rate
}
for month in range(1, 13):
current_balance = credit_line
for day in range(1, 32):
if random.randint(0, 7) == 0:
YES
if random.random() < 0.95:
YES
available_credit_line = current_balance
purchase_amount = random.uniform(0, available_credit_line)
current_balance += purchase_amount
NO
NO


NO
NO
daily_interest_rate = annual_interest_rate / 365
current_balance *= (1 + daily_interest_rate)
else:
available_cash = min(current_balance, 0.10 * credit_line)
cash_advance_amount = random.uniform(0, available_cash)
current_balance += cash_advance_amount
if random.randint(1, 10) == 1:
YES
current_balance = 0
NO
elif random.randint(1, 7) == 1:
YES
current_balance = 0
NO
elif random.randint(1, 7) == 1:
YES
payment_proportion = random.uniform(0, 1)
current_balance *= (1 - payment_proportion)
NO
elif random.randint(1, 10) == 1:
YES
current_balance = 0
NO


elif random.randint(1, 10) == 1:
minimum_amount_due = 0.10 * current_balance
current_balance -= minimum_amount_due
# Calculate monthly amounts
closing_balance = current_balance
minimum_amount_due = 0.10 * closing_balance
total_purchases = credit_line - closing_balance
total_cash_advances = credit_line - closing_balance
total_payments = credit_line - closing_balance
total_interest = (credit_line - closing_balance) * (annual_interest_rate / 12)
month_data = {
'Month': month,
'Closing_Balance': closing_balance,
'Minimum_Amount_Due': minimum_amount_due,
'Total_Purchases': total_purchases,
'Total_Cash_Advances': total_cash_advances,
'Total_Payments': total_payments,
'Total_Interest': total_interest
}
account['Month_' + str(month)] = month_data
accounts.append(account)
customer = {
'Customer_ID': customer_id,
'Age': age,
'Gender': gender,
'Marital_Status': marital_status,
'Number_of_Children': num_children,
'Education_Level': education_level,
'Annual_Income': annual_income,
'Total_Credit_Line': total_credit_line,
'Accounts': accounts
}
customers.append(customer)



***PSEUDOCODE*:**

- Start
- Import necessary libraries like import random, for datetime import datetime, timedelta
- Initializing an empty list to store customer data - customers = []
- Defined age and marital status distribution
  age_distribution = generate_random_ages(20000, 20, 80)
  marital_status_distribution = []
  for age in age_distribution:
  if age >= 20 and age <= 30:
  marital_status = random.choice(['Single', 'Married'], p=[0.75, 0.25])
  elif age > 30 and age <= 60:
  marital_status = random.choice(['Single', 'Married'], p=[0.25, 0.75])
  else:
  marital_status = random.choice(['Single', 'Married'], p=[0.5, 0.5])

  marital_status_distribution.append(marital_status)
- Defined the number of children distribution
  num_children_distribution = []
  for age in age_distribution:
  if age >= 20 and age <= 40:
  num_children = random.choice([0, 1, 2, 3, 4], p=[0.4, 0.3, 0.2, 0.1, 0.0])
  else:
  num_children = random.choice([0, 1, 2, 3, 4], p=[0.1, 0.3, 0.3, 0.2, 0.1])
  num_children_distribution.append(num_children)
- Define education level distribution
  education_level_distribution = []

```
 for age in age_distribution:
 if age >= 20 and age <= 25:
education_level = random.choice([0, 1, 2, 3, 4], p=[0.1, 0.5, 0.3, 0.1, 0.0])
elif age > 25 and age <= 35:
education_level = random.choice([0, 1, 2, 3, 4], p=[0.1, 0.5, 0.3, 0.05, 0.05])
else:
education_level = random.choice([0, 1, 2, 3, 4], p=[0.1, 0.5, 0.25, 0.1, 0.05])
education_level_distribution.append(education_level)
```

- Generate customer data for 20,000 customers

```
 for i in range(20000):
 customer_id = generate_random_customer_id()
 age = age_distribution[i]
 gender = generate_random_gender()
 marital_status = marital_status_distribution[i]
 num_children = num_children_distribution[i]
 education_level = education_level_distribution[i]
 annual_income = calculate_annual_income(age, education_level)
 num_accounts = generate_random_num_accounts()
 total_credit_line = calculate_total_credit_line(annual_income)
```

- Generate account information

```
accounts = []

for j in range(num_accounts):
date_opened = generate_random_date_opened(age)
account_number = generate_random_account_number()
credit_line = generate_random_credit_line(total_credit_line)
annual_fee = calculate_annual_fee(credit_line)
annual_interest_rate = generate_random_interest_rate()
account_data = generate_account_data(date_opened, account_number, credit_line,
annual_fee, annual_interest_rate)
```

- Generate account activity for 12 months

```
for month in range(1, 13):
        current_balance = credit_line
        for day in range(1, 32):
           if random.randint(0, 7) == 0:
              # Customer uses the card
              if random.random() < 0.95:
                 # Make a purchase
                 available_credit_line = current_balance
                 purchase_amount = random.uniform(0, available_credit_line)
                 current_balance += purchase_amount
```

```
            else:
                # Take a cash advance
                available_cash = min(current_balance, 0.10 * credit_line)
                cash_advance_amount = random.uniform(0, available_cash)
                current_balance += cash_advance_amount

        # Apply daily interest
        daily_interest_rate = annual_interest_rate / 365
        current_balance *= (1 + daily_interest_rate)

        # Check payment scenarios
        if random.randint(1, 10) == 1:
            # 10% pay the exact amount they spent every day
            current_balance = 0
        elif random.randint(1, 7) == 1:
            # 30% pay the entire balance every d number of days
            current_balance = 0
        elif random.randint(1, 7) == 1:
            # 30% pay only a proportion p of the entire balance every d number of days
            payment_proportion = random.uniform(0, 1)
            current_balance *= (1 - payment_proportion)
        elif random.randint(1, 10) == 1:
            # 10% pay the entire balance within the Payment Period
            current_balance = 0
        elif random.randint(1, 10) == 1:
            # 15% pay only the Minimum Amount Due within the Payment Period
            minimum_amount_due = 0.10 * current_balance
            current_balance -= minimum_amount_due
```

- Calculate monthly data i.e account:

```
    closing_balance = current_balance
    minimum_amount_due = 0.10 * closing_balance
    total_purchases = credit_line - closing_balance
    total_cash_advances = credit_line - closing_balance
    total_payments = credit_line - closing_balance
    total_interest = (credit_line - closing_balance) * (annual_interest_rate / 12)

accounts.append(account)

customer_data = generate_customer_data(customer_id, age, gender, marital_status,
num_children, education_level, annual_income, accounts)
customers.append(customer_data)
```

- Stop