**Explanation of the Python Code**

The script is organized into five logical parts that build upon each other to create the final, organic dataset.

**Part 1: Generating Base Synthetic Profiles**

* **What it does:** This section creates the foundational "skeleton" for each of the 50,000 users. It generates features that we assume are not directly available from a transaction history, like a user's age, the quality of their smartphone (device\_tier), and behavioral traits gathered from other sources (peer\_default\_exposure, financial\_shock\_coping).
* **Key Logic:** It uses NumPy to generate random data that follows realistic patterns (e.g., age is generated with a slight skew). Crucially, it sets an initial income\_tier for each user, which will guide how the transaction data is simulated in the next step.

**Part 2: Simulating Raw UPI Transactions**

* **What it does:** This is the heart of the "organic" data generation. Instead of just creating a final income\_consistency score out of thin air, this part simulates the raw data needed to calculate it. It creates a massive list of individual UPI transactions for all 50,000 users over a 3-month period.
* **Key Logic:**
  + It loops through every user.
  + Based on the user's income\_tier, it simulates a realistic monthly salary credit.
  + It then simulates debit transactions for various categories (utility bills, food orders, loan payments) with some randomness to mimic real life (e.g., a user might miss a bill payment).
  + Higher-income users are programmed to have more discretionary spending (more food orders).
  + All these transactions are stored in the upi\_df DataFrame, which acts as your "fake UPI statement" file.

**Part 3: Calculating Organic Features from UPI Data**

* **What it does:** This section acts like a real fintech data scientist. It takes the raw upi\_df from Part 2 and calculates the meaningful features your model needs.
* **Key Logic:**
  + It uses groupby('user\_id') to analyze each user's transactions separately.
  + income\_consistency: Calculates the stability of income by measuring the standard deviation of salary credits. A lower deviation means higher consistency.
  + utility\_payment\_ratio: Counts how many *unique* types of utility bills a user paid, then divides it by the total number of possible utility bills to get a ratio of responsible payments.
  + debt\_burden: Sums up all 'Loan EMI' payments and divides them by the total income received.
  + ...spend\_tier: It sums up spending in categories like food and assigns a tier (0, 1, or 2) based on the monthly average.

**Part 4: Merging, Finalizing, and Calculating Risk Scores**

* **What it does:** This is the assembly line. It merges the calculated "organic" features from Part 3 back into the base user profiles from Part 1. It then calculates all the final \_risk scores.
* **Key Logic:**
  + pd.merge(...): It uses a "left merge" to ensure every user from the original df is present in the final dataset.
  + **Error Prevention:** The .fillna(0) is applied *only* to the newly calculated columns. This prevents the TypeError you discovered by not trying to fill the age\_group (e.g., '18-25') with a 0.
  + The rest of the code is the same logic as before, using the assign\_risk\_score function to convert every feature into a standardized 0-1-2 risk score, calculating a total\_risk, and finally generating the loan\_default target variable.

**Part 5: Dynamic File Saving**

* **What it does:** This is a professional touch for good experiment management. It ensures you never accidentally overwrite a previously generated dataset.
* **Key Logic:**
  + It uses os.path.exists() to check if a file with the target name already exists.
  + If it does, it enters a while loop, adding a counter (\_1, \_2, etc.) to the filename until it finds a name that hasn't been used.
  + This guarantees that every time you run the script, you get a new set of uniquely named files.

**Table of Variables and Real-World Sourcing**

This table explains each key feature in your final credit\_data\_final\_v3.csv file, its purpose, and how a real company would acquire this data.

|  |  |  |
| --- | --- | --- |
| **Feature Name** | **Purpose in Credit Modeling** | **Real-World Sourcing Method** |
| **user\_id** | Unique identifier for each borrower. | Internal database ID generated upon user signup. |
| **age / age\_group** | Core demographic to understand user's life stage, which influences financial behavior. | User input during onboarding (KYC process). Verified via government ID. |
| **income\_tier** | A high-level category of the user's financial standing (Weak, Mid, Strong). | Derived from multiple sources. A preliminary tier can be from user declaration, but the final tier is confirmed by analyzing income from transaction data. |
| **device\_tier** | Proxy for wealth and digital literacy. Users with higher-end devices may be lower risk. | Device-level information collected from the mobile app (with user permission). |
| **app\_diversity** | Proxy for digital engagement. Very low diversity could signal low digital literacy or an inactive user. | Analysis of installed applications on the device (with user permission and strict privacy rules). |
| **peer\_default\_exposure** | A signal of social risk. If a user's network has many defaulters, their own risk may be higher. | Advanced and controversial. Could be sourced (with consent) from phone contacts that are also on the platform, or through geo-location clusters. Requires very careful ethical consideration. |
| **financial\_shock\_coping** | A psychometric measure of financial resilience. | Directly asking the user in a questionnaire: "How would you handle an unexpected ₹5,000 expense?" |
| **asset\_diversity** | A measure of wealth and stability. More types of assets (savings, vehicle) suggest lower risk. | User-declared information, or more reliably through **Account Aggregator (AA) framework** which can see Fixed Deposits, Mutual Funds etc. |
| **income\_consistency** | **Organic.** Measures if income is stable and predictable. The most important signal for repayment ability. | **Calculation:** Analyze transaction data (from AA or UPI) for recurring credits. Calculate the standard deviation of amounts and frequency over 3-6 months. |
| **utility\_payment\_ratio** | **Organic.** A direct proxy for financial discipline and responsibility. | **Calculation:** Scan transaction data for payments to known billers (electricity, gas, mobile). Calculate the ratio of bills paid versus billing cycles observed. |
| **debt\_burden** | **Organic.** The percentage of income that goes towards servicing existing debt. A key risk indicator. | **Calculation:** Sum all loan debits (EMIs) from transaction data and divide by total monthly income credits. |
| **...spend\_tier** | **Organic.** Categorizes discretionary spending to see if a user is living within their means. | **Calculation:** Sum all debits to merchants in a specific category (e.g., Food Delivery) and bin the total monthly spend into tiers (Low, Medium, High). |
| **...\_risk (e.g. debt\_risk)** | Standardized risk score (0, 1, 2) for every feature, making them comparable. | **Internal Calculation.** This is not raw data; it's the output of your business logic (the assign\_risk\_score function). |
| **total\_risk** | The final, weighted summary of a user's entire risk profile based on all features. | **Internal Calculation.** This is the output of your proprietary weighted risk formula. |
| **loan\_default** | **The Target Variable.** The ground truth of whether a user paid back their loan (0) or not (1). | **Observed Real-World Outcome.** After giving a loan, you monitor repayments. This data is collected over time and used to train future versions of the model. |