1. Load the Data:
- Import the datasets from the provided Excel files.
- Use the `pandas` library in Python to load the data.

2. Inspect the Data:
- Display the first few rows of each dataset to understand the structure.
- Check for missing values in each dataset.

3. Data Cleaning:
- Handle missing values appropriately (e.g., filling with mean/median, dropping rows/columns).
- Ensure all columns have the correct data types (e.g., dates should be in datetime format, numerical values as floats or integers).
- Remove any duplicate rows.

Loan_data

Step 1: Handle Missing Values

The columns with missing values are:

- total_loan_costs: 2 missing
- lender_credits: 5328 missing
- prepayment_pelty_term: 5367 missing
- intro_rate_period: 5282 missing
- recurring_monthly_debt: 27 missing
- aus_type: 63 missing

Step 2: Decide on Strategies to Handle Missing Values

- For columns with a small number of missing values (total_loan_costs, recurring_monthly_debt, aus_type), we can fill them with appropriate values or the mean/median.
- For columns with a large number of missing values (lender_credits, prepayment_pelty_term, intro_rate_period), we might drop these columns if they are not critical.

Step 3: Fix Data Types

- Ensure numerical columns are of numeric types.
- Ensure categorical columns are of categorical/string types.

Step 4: Drop Duplicates

- Remove any duplicate rows.

Step 5: Standardize Column Names

- Make column names lowercase and replace spaces with underscores.

target_profit

Step 1: Identify and Handle Missing Values

- **Total Loan Costs**: 2 missing
- **Lender Credits**: 5328 missing
- **Prepayment Penalty Term**: 5367 missing
- **Intro Rate Period**: 5282 missing
- **Recurring Monthly Debt**: 27 missing
- **AUS Type**: 63 missing
- **Gross Profit**: 2 missing
- **Profit Margin**: 2 missing
- **Actual Loan Revenue**: 3900 missing
- **Actual Profit Margin**: 3900 missing

Step 2: Handle Missing Values

We'll address these missing values based on the nature of the column and the proportion of missing data. For columns with a small number of missing values, we can consider imputing or dropping them. For columns with a large number of missing values, we might need to drop them or consider more complex imputation strategies.

Step 3: Remove Duplicates

Next, let's check for and remove any duplicate rows in the dataset.

Step 4: Correct Data Types

We'll ensure that all columns have the appropriate data types.

Step 5: Standardize Formatting

Standardize categorical data for consistency.

loan_status

The dataset has missing values in the following columns:

- `file_in_audit`: 253 missing values
- `file_audit_complete`: 776 missing values
- `file_sent_to_custodian`: 1160 missing values
- `file_at_custodian`: 2123 missing values
- • **Drop rows with missing values**: This approach can be used if the missing values are relatively few.
- • **Impute missing values**: Fill in missing values using a specific value or method, such as the mean, median, or mode.
- • **Leave as is**: If the missing values are informative, we might decide to leave them as is.

-

- **Ensuring date columns are in the correct format**.
- **Removing duplicate rows**.
- **Renaming columns**.

The initial cleaning steps have been completed:

1. Date columns have been converted to a consistent datetime format.
2. Duplicate rows have been removed.
3. Columns have been renamed for consistency and readability.

- **Identify patterns in missing data**: Determine if missing values are sequential or random.
- **Group by 'Loan ID'**: Analyze the progression of dates for each loan to understand the flow.

4. Create Loan Statuses:

- Define loan statuses based on the `current_balance` and `next_payment_due_date` in the `loan_balances` dataset.

- Create a new column `loan_status` to categorize the loans as 'Active', 'Closed', or 'Delinquent'.

To define loan statuses based on the `current_balance` and `next_payment_due_date` in the `loan_balances` dataset, follow these steps:

1. **Loan Status Definition**:
   - If `current_balance` is 0, the loan status is 'Closed'.
   - If `next_payment_due_date` is in the past and `current_balance` is greater than 0, the loan status is 'Delinquent'.
   - If `next_payment_due_date` is in the future and `current_balance` is greater than 0, the loan status is 'Active'.

5. Amortize Loan Balances:
- Create a function to calculate the amortized balance for each loan in the `loan_balances` dataset.
- Add a new column `amortized_balance` to the dataset with the calculated values.

To create a function that calculates the amortized balance for each loan in the `loan_balances` dataset, we need to define the amortization formula. The amortized balance calculation typically involves the original loan amount, interest rate, number of payments made, and total number of payments (loan term).

The formula to calculate the remaining balance of a loan at any point in time can be derived from the amortization schedule. Here is a simplified approach to calculate the amortized balance:

A=P(1−(1+r)−nr)A = P \left(\frac{1 - (1 + r)^{-n}}{r} \right)A=P(r1−(1+r)−n)

Where:

- AAA is the remaining loan balance.
- PPP is the monthly payment.
- rrr is the monthly interest rate.
- nnn is the number of remaining payments.

6. Merge Datasets:
   - Merge the loan_status, loan_data, target_profit, `loan_balances`, `umbs_prices`, and `loan_bids` datasets on relevant keys.
   - Ensure that the merged dataset is correctly joined and no important data is lost.

To merge multiple datasets, you need to identify common keys between them. Here are the potential keys for merging:

- `loan_balances`, `loan_data`, `loan_status`, `loan_bids`, `target_profit` can likely be merged on a loan identifier such as `loan_id` or `loan_number`.
- `umbs_prices` may have a different key, such as `umbs_code`, that relates to a specific attribute in the other datasets.

Let's assume `loan_id` is the common key for most datasets, and `umbs_code` is the key for merging `umbs_prices` with the other datasets. Below is the Python code to perform these merges: