1. Imports and Setup

In [8]:
```
import pandas as pd
import numpy as np
import os
from datetime import datetime

# --- Setup ---
# Create directories for processed data and figures if they don't exist
os.makedirs('../data/processed', exist_ok=True)
os.makedirs('../reports/figures', exist_ok=True)

print("Setup Complete. Directories are ready.")
```

```
Setup Complete. Directories are ready.
```

2. Load the Data

In [13]:
```
# This cell loads your Excel file.
# Make sure your file is named 'data.xlsx' and is in the 'data/raw/' folder.
file_path = '../data/raw/data.xlsx'

try:
        # The following lines are indented to be inside the 'try' block.
        df = pd.read_excel(file_path, engine='openpyxl')
        print(f"Data loaded successfully from: {file_path}")
        print(f"Dataset shape: {df.shape}")

except FileNotFoundError:
        # The following lines are indented to be inside the 'except' block.
        print(f"ERROR: The file was not found at the specified path: {file_path}
        print("Please ensure the file exists in the 'data/raw/' directory and th

# This line is outside the try/except block, so it is not indented.
# It will only run if the 'try' block succeeds.
df.head()
```

```
Data loaded successfully from: ../data/raw/data.xlsx
Dataset shape: (233154, 41)
```

Out[13]:

	UniqueID	disbursed_amount	asset_cost	ltv	branch_id	supplier_id	manufacturer
0	420825	50578	58400	89.55	67	22807	
1	417566	53278	61360	89.63	67	22807	
2	539055	52378	60300	88.39	67	22807	
3	529269	46349	61500	76.42	67	22807	
4	563215	43594	78256	57.50	67	22744	

5 rows × 41 columns

3. Initial Inspection

In [14]:
```
print("--- Data Info ---")
# .info() gives us column names, non-null counts, and data types.
df.info()

print("\n--- Missing Values (Top 15) ---")
# .isnull().sum() counts missing values for each column.
print(df.isnull().sum().sort_values(ascending=False).head(15))

print(f"\n--- Number of Duplicate Rows ---")
# .duplicated().sum() counts the number of identical rows.
print(df.duplicated().sum())
```

```
--- Data Info ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 233154 entries, 0 to 233153
Data columns (total 41 columns):
 #   Column                                Non-Null Count   Dtype
---  ------                                --------------   -----
 0   UniqueID                              233154 non-null  int64
 1   disbursed_amount                      233154 non-null  int64
 2   asset_cost                            233154 non-null  int64
 3   ltv                                   233154 non-null  float64
 4   branch_id                             233154 non-null  int64
 5   supplier_id                           233154 non-null  int64
 6   manufacturer_id                       233154 non-null  int64
 7   Current_pincode_ID                    233154 non-null  int64
 8   Date.of.Birth                         233154 non-null  datetime64[ns]
 9   Employment.Type                       225493 non-null  object
 10  DisbursalDate                         233154 non-null  datetime64[ns]
 11  State_ID                              233154 non-null  int64
 12  Employee_code_ID                      233154 non-null  int64
 13  MobileNo_Avl_Flag                     233154 non-null  int64
 14  Aadhar_flag                           233154 non-null  int64
 15  PAN_flag                              233154 non-null  int64
 16  VoterID_flag                          233154 non-null  int64
 17  Driving_flag                          233154 non-null  int64
 18  Passport_flag                         233154 non-null  int64
 19  PERFORM_CNS.SCORE                     233154 non-null  int64
 20  PERFORM_CNS.SCORE.DESCRIPTION         233154 non-null  object
 21  PRI.NO.OF.ACCTS                       233154 non-null  int64
 22  PRI.ACTIVE.ACCTS                      233154 non-null  int64
 23  PRI.OVERDUE.ACCTS                     233154 non-null  int64
 24  PRI.CURRENT.BALANCE                   233154 non-null  int64
 25  PRI.SANCTIONED.AMOUNT                 233154 non-null  int64
 26  PRI.DISBURSED.AMOUNT                  233154 non-null  int64
 27  SEC.NO.OF.ACCTS                       233154 non-null  int64
 28  SEC.ACTIVE.ACCTS                      233154 non-null  int64
 29  SEC.OVERDUE.ACCTS                     233154 non-null  int64
 30  SEC.CURRENT.BALANCE                   233154 non-null  int64
 31  SEC.SANCTIONED.AMOUNT                 233154 non-null  int64
 32  SEC.DISBURSED.AMOUNT                  233154 non-null  int64
 33  PRIMARY.INSTAL.AMT                    233154 non-null  int64
 34  SEC.INSTAL.AMT                        233154 non-null  int64
 35  NEW.ACCTS.IN.LAST.SIX.MONTHS          233154 non-null  int64
 36  DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS   233154 non-null  int64
 37  AVERAGE.ACCT.AGE                      233154 non-null  object
 38  CREDIT.HISTORY.LENGTH                 233154 non-null  object
 39  NO.OF_INQUIRIES                       233154 non-null  int64
 40  loan_default                          233154 non-null  int64
dtypes: datetime64[ns](2), float64(1), int64(34), object(4)
memory usage: 72.9+ MB

--- Missing Values (Top 15) ---
Employment.Type       7661
disbursed_amount         0
asset_cost               0
ltv                      0
branch_id                0
supplier_id              0
manufacturer_id          0
Current_pincode_ID       0
UniqueID                 0
```

```
Date.of.Birth            0
DisbursalDate            0
State_ID                 0
Employee_code_ID         0
MobileNo_Avl_Flag        0
Aadhar_flag              0
dtype: int64

--- Number of Duplicate Rows ---
0
```

4. Clean Column Names

In [15]:
```
original_columns = df.columns
df.columns = [col.strip().lower().replace('.', '_') for col in original_columns]
print("Column names have been cleaned.")
print("Example: '{}' is now '{}'".format(original_columns[10], df.columns[10]))
```

```
Column names have been cleaned.
Example: 'DisbursalDate' is now 'disbursaldate'
```

5. Handle Duplicates and Missing Values

In [19]:
```
# --- Handle Duplicates ---
if df.duplicated().sum() > 0:
    df.drop_duplicates(inplace=True)
    print(f"Dropped {df.duplicated().sum()} duplicate rows. New shape: {df.shape
else:
    print("No duplicate rows found.")

# --- Handle Missing Values ---
# For 'employment_type', the number of missing values is small. We can fill with
if 'employment_type' in df.columns:
    mode_employment = df['employment_type'].mode()[0]
    df['employment_type'] = df['employment_type'].fillna(mode_employment)
    print(f"Filled missing 'employment_type' values with '{mode_employment}'.")

# Let's re-check missing values to confirm
print("\nMissing values after handling:")
print(df.isnull().sum().sort_values(ascending=False).head())
```

```
No duplicate rows found.
Filled missing 'employment_type' values with 'Self employed'.

Missing values after handling:
uniqueid            0
disbursed_amount    0
asset_cost          0
ltv                 0
branch_id           0
dtype: int64
```

6. Feature Engineering - Create 'age' Column

In [20]:
```
# --- Feature Engineering: Age ---
# Convert 'date_of_birth' to datetime objects. The format is Day-Month-2-digit-Y
df['date_of_birth'] = pd.to_datetime(df['date_of_birth'], format='%d-%m-%y')
```

```
# --- IMPORTANT: Correcting for the 'YY' to 'YYYY' ambiguity ---
# Pandas interprets years like '90' as 1990, but '10' as 2010.
# A person born in 2010 would be a child, which is unlikely for a loan applicant
# We assume anyone with a year > current_year was born in the 1900s.
current_year = datetime.now().year
df['birth_year'] = df['date_of_birth'].dt.year
df.loc[df['birth_year'] > current_year, 'birth_year'] -= 100

# Calculate age
df['age'] = current_year - df['birth_year']

# Drop the temporary 'birth_year' column
df.drop(columns=['birth_year'], inplace=True)

print("Created 'age' column from 'date_of_birth'.")
df[['date_of_birth', 'age']].head()
```

```
Created 'age' column from 'date_of_birth'.
```

Out[20]:

	date_of_birth	age
0	1984-01-01	41
1	1985-08-24	40
2	1977-12-09	48
3	1988-06-01	37
4	1994-07-14	31

7. Save the Cleaned Data

In [21]:

```
# --- Save Cleaned Data ---
cleaned_file_path = '../data/processed/cleaned_loan_data.csv'
df.to_csv(cleaned_file_path, index=False)

print(f"Cleaned data successfully saved to: {cleaned_file_path}")
```

```
Cleaned data successfully saved to: ../data/processed/cleaned_loan_data.csv
```