**Assignment 1: Filtering Data Based on Conditions**

**Tasks:**

1. Filter out all rows where the Transaction\_Amount is greater than 2000.
2. Find all rows where the Transaction\_Type is "Loan Payment" and the Account\_Balance is greater than 5000.
3. Extract transactions made in the "Uptown" branch.

**Objective:**

* Practice filtering data using conditions and boolean indexing.

**Assignment 2: Data Transformation**

**Tasks:**

1. Add a new column called Transaction\_Fee, calculated as 2% of Transaction\_Amount.
2. Create a new column Balance\_Status:
   * If Account\_Balance is greater than 5000, label it as "High Balance".
   * Otherwise, label it as "Low Balance".

**Objective:**

* Learn how to create new columns and apply transformations based on existing data.

**Program:**

import pandas as pd

# Load the CSV file into a DataFrame

df = pd.read\_csv('banking\_data.csv')

# Assignment 1: Filtering Data Based on Conditions

# Task 1: Filter out all rows where the Transaction\_Amount is greater than 2000

filtered\_by\_transaction\_amount = df[df['Transaction\_Amount'] <= 2000]

print("Rows where Transaction\_Amount <= 2000:")

print(filtered\_by\_transaction\_amount)

# Task 2: Find all rows where the Transaction\_Type is "Loan Payment" and Account\_Balance > 5000

filtered\_loan\_payment = df[(df['Transaction\_Type'] == 'Loan Payment') & (df['Account\_Balance'] > 5000)]

print("\nRows where Transaction\_Type is 'Loan Payment' and Account\_Balance > 5000:")

print(filtered\_loan\_payment)

# Task 3: Extract transactions made in the "Uptown" branch

filtered\_uptown\_branch = df[df['Branch'] == 'Uptown']

print("\nTransactions made in the 'Uptown' branch:")

print(filtered\_uptown\_branch)

# Assignment 2: Data Transformation

# Task 1: Add a new column called Transaction\_Fee, calculated as 2% of Transaction\_Amount

df['Transaction\_Fee'] = df['Transaction\_Amount'] \* 0.02

print("\nData with new Transaction\_Fee column:")

print(df[['Transaction\_Amount', 'Transaction\_Fee']].head()) # Show only the relevant columns for clarity

# Task 2: Create a new column Balance\_Status

df['Balance\_Status'] = df['Account\_Balance'].apply(lambda x: 'High Balance' if x > 5000 else 'Low Balance')

print("\nData with new Balance\_Status column:")

print(df[['Account\_Balance', 'Balance\_Status']].head()) # Show only the relevant columns for clarity