Industry Problem Statement

Banking Account & Transaction Management System (Python)

Business Background

A retail bank wants to develop a Python-based internal system to manage:

- Customer bank accounts
- Deposits and withdrawals
- Balance validation
- Transaction history and reporting

The system should simulate **core banking operations** with proper validations and auditability, similar to real-world banking software.

Task 1: Capture Customer & Account Details (Input Validation)

Objective

Collect and validate bank account information.

Requirements

Write a program to accept:

- Customer ID
- Customer Name
- Account Number
- Account Type (Savings / Current)

Business Rules

- Account number must be numeric and unique

- Customer name must contain only alphabets
- Account type must be valid

Expected Outcome

Validated customer account record.

Task 2: Initialize Account Balance

Objective

Set opening balance.

Requirements

- Accept opening balance

Business Rules

- Minimum balance:
    - Savings → ₹1,000
    - Current → ₹5,000

Expected Outcome

Account created with valid opening balance.

Task 3: Deposit Transaction

Objective

Handle cash/online deposits.

Requirements

- Accept deposit amount

Business Rules

- Deposit amount must be greater than 0

Formula

```
New Balance = Current Balance + Deposit Amount
```

Expected Outcome

Updated account balance.

Task 4: Withdrawal Transaction

Objective

Process withdrawals.

Requirements

- Accept withdrawal amount

Business Rules

- Withdrawal amount must be > 0
- Balance after withdrawal must not fall below minimum balance

Formula

```
New Balance = Current Balance - Withdrawal Amount
```

Expected Outcome

Safe and validated withdrawal.

Task 5: Balance Inquiry

Objective

Display real-time account balance.

Output

- Account Number
- Current Balance

Task 6: Transaction Recording

Objective

Maintain transaction history.

Requirements

- Store each transaction in a list of dictionaries
- Include:
    - Transaction Type (Deposit / Withdrawal)
    - Amount
    - Date
    - Balance After Transaction

Task 7: Daily Transaction Limit Check

Objective

Prevent fraud and misuse.

Business Rules

- Maximum withdrawal per day: ₹50,000

Expected Outcome

Transaction rejected if limit exceeded.

Task 8: Interest Calculation (Savings Account)

Objective

Apply monthly interest.

Rules

- Interest Rate = 4% per annum
- Calculate monthly interest

Formula

```
Monthly Interest = (Balance × 4%) / 12
```

Expected Outcome

Updated balance with interest.

Task 9: Procedural Account Summary

Objective

Generate a quick account snapshot.

Summary Should Include

- Customer Name
- Account Number
- Account Type
- Current Balance

Task 10: Multi-Account Management

Objective

Support multiple customers.

Requirements

- Store accounts in a dictionary
- Account number as key

Task 11: BankAccount Class Design (OOP)

Objective

Model bank account as a real-world object.

Create class `BankAccount` with:

Attributes

- customer_id
- customer_name
- account_number
- account_type
- balance
- transactions

Task 12: Transaction Methods

Objective

Encapsulate transaction logic.

Methods

- deposit()
- withdraw()

Task 13: Interest Method

Objective

Encapsulate interest logic.

Method

- calculate_interest()

Task 14: Transaction History Method

Objective

Display transaction details.

Method

- get_transaction_history()

Task 15: Final Bank Statement Generation

Objective

Generate a professional bank statement.

Output Format (Example)

```
Customer Name   : Ramesh Kumar
Account Number  : 1002456789
Account Type    : Savings
Current Balance : ₹45,800

Recent Transactions:
--------------------------------
Date       Type        Amount   Balance
01-Mar     Deposit     ₹10,000  ₹40,000
05-Mar     Withdrawal  ₹4,200   ₹35,800
31-Mar     Interest    ₹150     ₹45,800
```