Bank Account Management System

Design and implement a **Bank Account Management System** in Dart that uses **Maps** to store and manage customer account details such as account number, customer name, and balance. The system should support key operations like **deposit**, **withdraw**, and **balance inquiry**. To achieve flexibility and reusability, the implementation must use **functional parameters** (positional, named, required, optional, and default parameters) to handle different banking operations. Additionally, use **Map iterables** to traverse and update account data, and apply **functional programming techniques** to dynamically process account balances. The solution should simulate a real-world banking workflow where multiple accounts are managed efficiently in a structured program.

Solution:

```
double deposit(double balance, double amount) => balance + amount;
double withdraw(double balance, double amount) => balance - amount;

void main() {
  Map<int, Map<String, dynamic>> bank = {
    23458: {"name": "Syam", "balance": 500000.0, "city": "Hyderabad"},
    12763: {"name": "Karthik", "balance": 1000000.0, "city": "Bengaluru"},
    89435: {"name": "Niharika", "balance": 300000.0, "city": "Kaikaluru"},
  };

  List<int> accNumbers = [23458, 12763, 89435];
  List<String> custNames = ["Syam", "Karthik", "Niharika"];
  Map<int, String> accountMap = Map.fromIterables(accNumbers, custNames);
  print("Customer data: $accountMap\n");

  void performTransaction({
    required int accNo,
    required double amount,
    required double Function(double, double) operation,
    String? transactionType = "Merchant", //named parameter with default value
    bool printReceipt = true, //named parameter will be optional by default
  }) {
    if (!bank.containsKey(accNo)) {
      print("Account $accNo not found!");
      return;
    }

    double oldBalance =
```

```
        bank[accNo]!["balance"]; ////bank[accNo]!["balance"], since it is a
nested map, 1st we should access the top level key and then go into the next
level key
    double newBalance = operation(oldBalance, amount);
    bank[accNo]!["balance"] = newBalance;

    if (printReceipt) {
      print("$transactionType Transaction");
      print(
        "AccNo: $accNo, Name: ${bank[accNo]!["name"]}",
      ); //bank[accNo]!["name"], since it is a nested map, 1st we should
access the top level key and then go into the next level key
      print("Old Balance: $oldBalance, New Balance: $newBalance\n");
    }
  }

  performTransaction(
    accNo: 12763,
    amount: 200000,
    operation: deposit,
    transactionType: "Deposit",
  );

  performTransaction(
    accNo: 89435,
    amount: 150000,
    operation: withdraw,
    transactionType: "Withdraw",
    printReceipt: false,
  );

  print(" All Customers:");
  bank.forEach((accNo, details) {
    print(
      "AccNo: $accNo, Name: ${details["name"]}, Balance:
${details["balance"]}, City: ${details["city"]}",
    );
  });

  print("\n Balances Only:");
  print(bank.values.map((cust) => cust["balance"]).toList());

  print("\n Customers with balance > 100000:");
  bank.entries
      .where((entry) => entry.value["balance"] > 5000)
      .forEach(
```

```
        (entry) => print("${entry.value["name"]}: ${entry.value["balance"]}"),
      );
}
```

Expected Outcome:

```
Customer data: {23458: Syam, 12763: Karthik, 89435: Niharika}

Deposit Transaction
AccNo: 12763, Name: Karthik
Old Balance: 1000000.0, New Balance: 1200000.0

 All Customers:
AccNo: 23458, Name: Syam, Balance: 500000.0, City: Hyderabad
AccNo: 12763, Name: Karthik, Balance: 1200000.0, City: Bengaluru
AccNo: 89435, Name: Niharika, Balance: 150000.0, City: Kaikaluru

 Balances Only:
[500000.0, 1200000.0, 150000.0]

 Customers with balance > 100000:
Syam: 500000.0
Karthik: 1200000.0
Niharika: 150000.0
```