**Customers Table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CUSTOMERID | NAME | DOB | BALANCE | LASTMODIFIED |
| 3 | Ravi Kumar | 1959-01-10T00:00:00Z | 20000 | 2025-06-27T15:11:37Z |
| 4 | Anita Desai | 1962-09-25T00:00:00Z | 8000 | 2025-06-27T15:11:46Z |
| 5 | Michael Chen | 1975-03-15T00:00:00Z | 3000 | 2025-06-27T15:11:52Z |
| 1 | John Doe | 1985-05-15T00:00:00Z | 1000 | 2025-06-27T15:01:35Z |
| 2 | Jane Smith | 1990-07-20T00:00:00Z | 1500 | 2025-06-27T15:01:19Z |

**Accounts Table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACCOUNTID | CUSTOMERID | ACCOUNTTYPE | BALANCE | LASTMODIFIED |
| 1 | 1 | Savings | 1000 | 2025-06-27T15:03:03Z |
| 2 | 2 | Checking | 1500 | 2025-06-27T15:03:18Z |
| 3 | 3 | Savings | 25000 | 2025-06-27T15:18:46Z |
| 5 | 5 | Savings | 4000 | 2025-06-27T15:19:10Z |
| 4 | 4 | Checking | 5000 | 2025-06-27T15:19:00Z |

**Transactions Table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TRANSACTIONID | ACCOUNTID | TRANSACTIONDATE | AMOUNT | TRANSACTIONTYPE |
| 1 | 1 | 2025-06-27T15:03:33Z | 200 | Deposit |
| 2 | 2 | 2025-06-27T15:03:47Z | 300 | Withdrawal |
| 4 | 4 | 2025-06-27T15:22:32Z | 700 | Withdrawal |
| 3 | 3 | 2025-06-27T15:21:52Z | 1000 | Deposit |
| 5 | 5 | 2025-06-27T15:22:39Z | 200 | Deposit |

**Employees Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EMPLOYEEID | NAME | POSITION | SALARY | DEPARTMENT | HIREDATE |
| 4 | Amit Gupta | Developer | 62000 | IT | 2018-07-10T00:00:00Z |
| 3 | Meera Sinha | Clerk | 35000 | Finance | 2020-02-01T00:00:00Z |
| 1 | Alice Johnson | Manager | 70000 | HR | 2015-06-15T00:00:00Z |
| 2 | Bob Brown | Developer | 60000 | IT | 2017-03-20T00:00:00Z |

**Loans Table:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| LOANID | CUSTOMERID | LOANAMOUNT | INTERESTRATE | STARTDATE | | ENDDATE |
| 2 | 2 | 10000 | 6.5 | 2025-06-27T15:25:33Z | 2029-06-27T15:25:33Z | |
| 1 | 1 | 5000 | 5 | 2025-06-27T15:04:02Z | 2030-06-27T15:04:02Z | |
| 3 | 3 | 20000 | 4.5 | 2025-06-27T15:25:45Z | 2028-06-27T15:25:45Z | |
| 4 | 4 | 7000 | 5.5 | 2025-06-27T15:25:53Z | 2027-06-27T15:25:53Z | |

**Exercise 1: Control Structures**

**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

* + **Question:** Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

**Scenario 2:** A customer can be promoted to VIP status based on their balance.

* + **Question:** Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

* + **Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

**Solution**

**Scenario 1:**

BEGIN

FOR c IN (SELECT \* FROM Customers) LOOP

IF MONTHS\_BETWEEN(SYSDATE, c.DOB)/12 > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = c.CustomerID;

END IF;

END LOOP;

END;

**Result:**

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.013

**Scenario 2:**

BEGIN

FOR c IN (SELECT \* FROM Customers WHERE Balance > 10000) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = c.CustomerID;

END LOOP;

END;

**Result:**

PL/SQL procedure successfully completed.  
Elapsed: 00:00:00.015

**Scenario 3:**

BEGIN

FOR l IN (SELECT \* FROM Loans WHERE EndDate <= SYSDATE + 30) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Customer ' || l.CustomerID || ' has loan due on ' || l.EndDate);

END LOOP;

END;

**Result:**

PL/SQL procedure successfully completed.  
Elapsed: 00:00:00.011

**Exercise 3: Stored Procedures**

**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

* + **Question:** Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

* + **Question:** Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

**Scenario 3:** Customers should be able to transfer funds between their accounts.

* + **Question:** Write a stored procedure **TransferFunds** that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

**Solution:**

**Scenario 1:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'Savings';

COMMIT;

END;

**Result:**

Procedure PROCESSMONTHLYINTEREST compiled

Elapsed: 00:00:00.013

**Scenario 2:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(p\_dept VARCHAR2, p\_bonus NUMBER) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* p\_bonus / 100)

WHERE Department = p\_dept;

COMMIT;

END;

**Result:**

Procedure UPDATEEMPLOYEEBONUS compiled

Elapsed: 00:00:00.014

**Scenario 3:**

CREATE OR REPLACE PROCEDURE TransferFunds(p\_from NUMBER, p\_to NUMBER, p\_amount NUMBER) IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_from;

IF v\_balance >= p\_amount THEN

UPDATE Accounts SET Balance = Balance - p\_amount WHERE AccountID = p\_from;

UPDATE Accounts SET Balance = Balance + p\_amount WHERE AccountID = p\_to;

COMMIT;

ELSE

RAISE\_APPLICATION\_ERROR(-20003, 'Insufficient balance');

END IF;

END;

**Result:**

Procedure TRANSFERFUNDS compiled

Elapsed: 00:00:00.018