**Exercise 1: Control Structures**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Age NUMBER,

Balance NUMBER,

IsVIP VARCHAR2(5)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES Customers(CustomerID),

InterestRate NUMBER,

DueDate DATE

);

INSERT INTO Customers VALUES (1, 'John Smith', 65, 8000, 'FALSE');

INSERT INTO Customers VALUES (2, 'Alice Johnson', 45, 12000, 'FALSE');

INSERT INTO Customers VALUES (3, 'Bob Lee', 70, 15000, 'FALSE');

INSERT INTO Customers VALUES (4, 'Maya Patel', 58, 5000, 'FALSE');

INSERT INTO Loans VALUES (101, 1, 6.5, SYSDATE + 15);

INSERT INTO Loans VALUES (102, 2, 7.2, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 5.9, SYSDATE + 10);

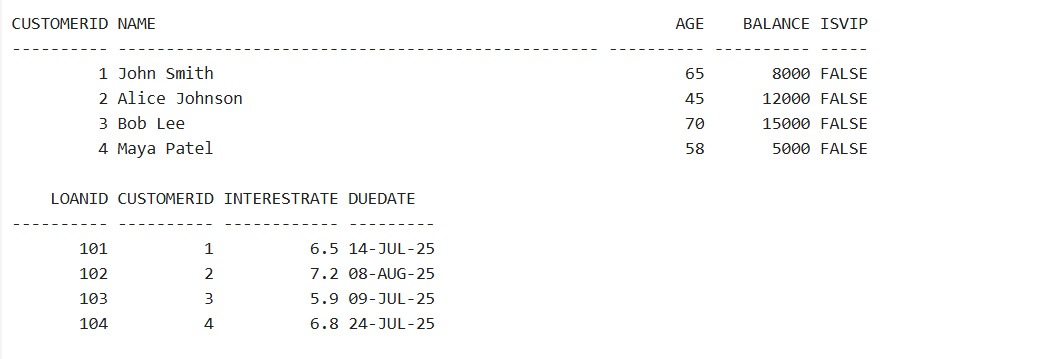
INSERT INTO Loans VALUES (104, 4, 6.8, SYSDATE + 25);

COMMIT;

SELECT \* FROM Customers;

SELECT \* FROM Loans;

**Output:**



**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.

BEGIN

FOR cust IN (SELECT CustomerID FROM Customers WHERE Age > 60) LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust.CustomerID;

END LOOP;

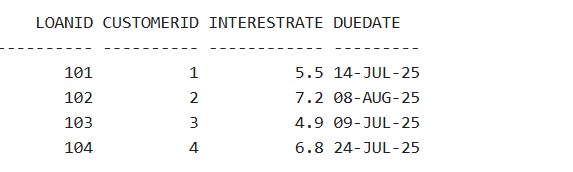
COMMIT;

END;

/

SELECT \* FROM Loans;

**Output:**



**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.

BEGIN

FOR cust IN (SELECT CustomerID FROM Customers WHERE Balance > 10000) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID;

END LOOP;

COMMIT;

END;

/

SELECT \* FROM Customers;

**Output:**



**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.

BEGIN

FOR loan\_rec IN (

SELECT c.Name, l.DueDate

FROM Loans l

JOIN Customers c ON c.CustomerID = l.CustomerID

WHERE l.DueDate <= SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan for ' || loan\_rec.Name ||

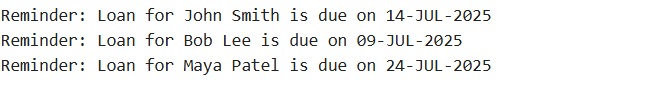
' is due on ' || TO\_CHAR(loan\_rec.DueDate, 'DD-MON-YYYY'));

END LOOP;

END;

/

**Output:**



**Exercise 3: Stored Procedures**

//Creating Tables

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(50),

Balance NUMBER,

AccountType VARCHAR2(20)

);

CREATE TABLE Employees (

EmpID NUMBER PRIMARY KEY,

EmpName VARCHAR2(50),

Department VARCHAR2(50),

Salary NUMBER

);

INSERT INTO Accounts VALUES (101, 'John Smith', 1000, 'Savings');

INSERT INTO Accounts VALUES (102, 'Alice Johnson', 2000, 'Current');

INSERT INTO Accounts VALUES (103, 'Bob Lee', 1500, 'Savings');

INSERT INTO Accounts VALUES (104, 'Maya Patel', 3000, 'Savings');

INSERT INTO Employees VALUES (1, 'Raj Verma', 'HR', 40000);

INSERT INTO Employees VALUES (2, 'Nina Rao', 'IT', 50000);

INSERT INTO Employees VALUES (3, 'Amit Das', 'IT', 55000);

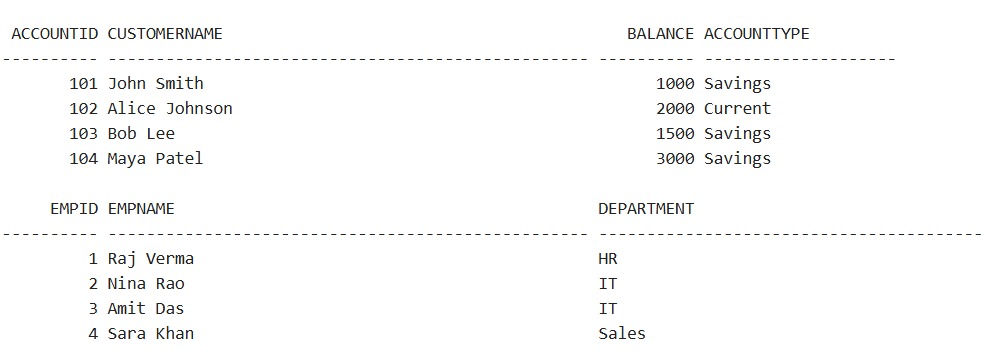
INSERT INTO Employees VALUES (4, 'Sara Khan', 'Sales', 45000);

COMMIT;

SELECT \* FROM Accounts;

SELECT \* FROM Employees;

**Output:**



**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.

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET Balance = Balance \* 1.01

WHERE AccountType = 'Savings';

COMMIT;

END;

/

BEGIN

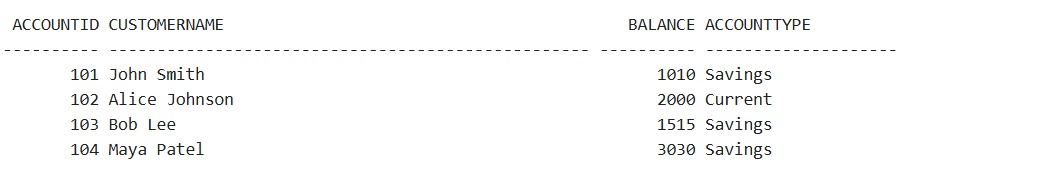
ProcessMonthlyInterest;

END;

/

SELECT \* FROM Accounts;

**Output:**



**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.

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_dept IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE Employees

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

WHERE Department = p\_dept;

COMMIT;

END;

/

BEGIN

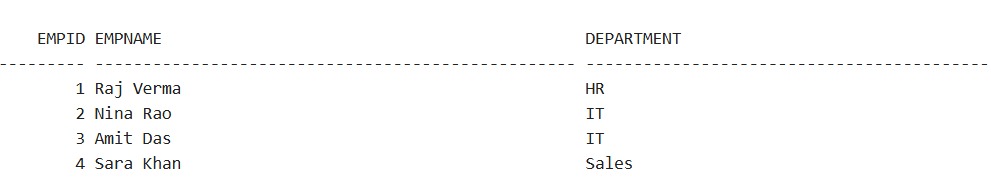
UpdateEmployeeBonus('IT', 10);

END;

/

SELECT \* FROM Employees;

**Output:**



**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.

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

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

IF v\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance for transfer.');

END IF;

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_account;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_account;

COMMIT;

END;

/

BEGIN

TransferFunds(104, 102, 300);

END;

/

SELECT \* FROM Accounts;

**Output:**

