**Exercise 1: Control Structures**

1. **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.

**Table Creation and Sample Insertion:**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

)

insert into Customers(CustomerID, Name, DOB, Balance, LastModified)

values(1, 'Albert', TO\_DATE('1980-05-21', 'YYYY-MM-DD'), 1000, SYSDATE)

insert into Customers(CustomerID, Name, DOB, Balance, LastModified)

values(2, 'Bala', TO\_DATE('1995-12-24', 'YYYY-MM-DD'), 20500, SYSDATE)

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

)

insert into Loans(LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

values(101, 1, 5000, 9, SYSDATE, ADD\_MONTHS(SYSDATE, 20))

insert into Loans(LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

values(102, 2, 2000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 5))

**PL/SQL CODE:**

BEGIN

FOR cust IN (

SELECT CustomerID

FROM customers

WHERE MONTHS\_BETWEEN(SYSDATE, DOB) / 12 > 60

) LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('1% Discount applied for CustomerID: ' || cust.CustomerID);

END LOOP;

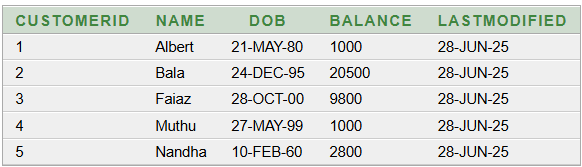
END;

/

**Output:**

1% Discount applied for CustomerID: 5

1. row(s) updated.



1. **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.

**PL / SQL CODE:**

BEGIN

FOR cust IN(

Select customerID, Balance

from Customers

) LOOP

IF cust.Balance > 10000 THEN

update Customers

set IsVIP = 'Y'

where CustomerID = cust.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('CustomerID ' ||cust.CustomerID|| ' promoted to VIP');

END IF;

END LOOP;

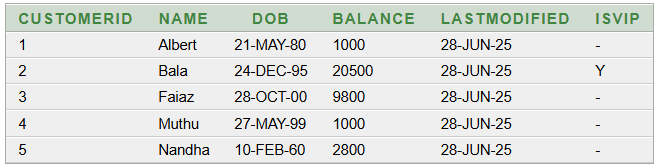
END;

/

**OUTPUT:**

CustomerID 2 promoted to VIP

1. row(s) updated.



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

**PL/SQL CODE:**

BEGIN

FOR loan\_rec in(

select l.LoanID, l.EndDate, c.Name

from loans l

join Customers c on l.CustomerID = c.CustomerID

where l.EndDate between SYSDATE and SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan\_rec.LoanID ||

' for customer ' || loan\_rec.Name ||

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

END LOOP;

END;

/

**OUTPUT:**

Reminder: Loan ID 202 for customer Bala is due on 03-JUL-2025

Statement processed.

**Exercise 3: Stored Procedures**

1. **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.

**Table Creation and Sample Insertion:**

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (1, 1, 'Savings', 1000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (2, 2, 'Checking', 1500, SYSDATE);

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (1, 'Alice', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (2, 'Bob', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

CREATE TABLE Transactions (

TransactionID NUMBER PRIMARY KEY,

AccountID NUMBER,

TransactionDate DATE,

Amount NUMBER,

TransactionType VARCHAR2(10),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (1, 1, SYSDATE, 200, 'Deposit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

**PL/SQL CODE:**

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01),

LastModified = SYSDATE

WHERE AccountType = 'Savings';

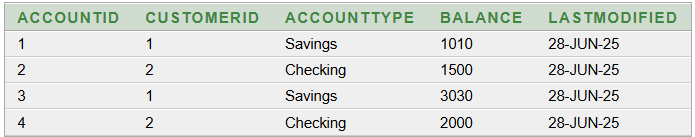
DBMS\_OUTPUT.PUT\_LINE('Monthly interest processed for all savings accounts.');

END; /

**OUTPUT:**

Monthly interest processed for all savings accounts.

1 row(s) updated.



1. **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.

**PL/SQL CODE:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

)

IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* bonus\_percent / 100)

WHERE Department = dept\_name;

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || bonus\_percent || '% applied to ' || dept\_name || ' department.');

END;

/

BEGIN

UpdateEmployeeBonus('IT', 10);

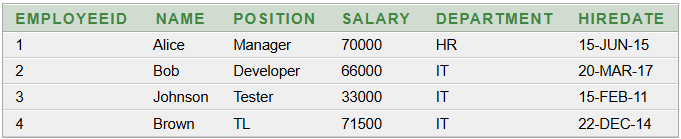
END;

/

**OUTPUT:**

Bonus of 10% applied to IT department.

Statement processed.



1. **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.

**PL / SQL CODE:**

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account\_id IN NUMBER,

to\_account\_id IN NUMBER,

amount IN NUMBER

)

IS from\_balance NUMBER;

BEGIN

SELECT Balance INTO from\_balance

FROM Accounts

WHERE AccountID = from\_account\_id

FOR UPDATE;

IF from\_balance < amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in source account.');

END IF;

UPDATE Accounts

SET Balance = Balance - amount,

LastModified = SYSDATE

WHERE AccountID = from\_account\_id;

UPDATE Accounts

SET Balance = Balance + amount,

LastModified = SYSDATE

WHERE AccountID = to\_account\_id;

DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || amount || ' successful from Account ' || from\_account\_id || ' to Account ' || to\_account\_id);

END;

/

BEGIN

TransferFunds(1, 2, 300);

END;

/

**OUTPUT:**

Transfer of 300 successful from Account 1 to Account 2

Statement processed.