**WEEK 2 – PLSQL EXERCISES**

**Exercise 1: Control Structures**

**SCHEMA CREATION**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Transactions (

TransactionID NUMBER PRIMARY KEY,

AccountID NUMBER,

TransactionDate DATE,

Amount NUMBER,

TransactionType VARCHAR2(10),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

**SAMPLE DATA**

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (1, 'John Doe', TO\_DATE('1985-05-15', 'YYYY-MM-DD'), 1000, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 1500, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (3, 'Ravi Sharma', TO\_DATE('1950-03-10', 'YYYY-MM-DD'), 20000, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (4, 'Anita Desai', TO\_DATE('1949-11-22', 'YYYY-MM-DD'), 12000, SYSDATE);

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);

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');

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (1, 1, 5000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 60));

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (2, 3, 8000, 7, SYSDATE, ADD\_MONTHS(SYSDATE, 12));

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (3, 4, 10000, 8, SYSDATE, ADD\_MONTHS(SYSDATE, 20));

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

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

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

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

**Scenario 1:** Apply 1% discount to senior citizens (> 60 years old)

DECLARE

CURSOR senior\_cust\_loans IS

SELECT l.LoanID, c.CustomerID, c.Name, l.InterestRate

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

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

BEGIN

FOR rec IN senior\_cust\_loans LOOP

UPDATE Loans SET InterestRate = InterestRate - 1

WHERE LoanID = rec.LoanID;

DBMS\_OUTPUT.PUT\_LINE(' Discounted: ' || rec.Name || ' (Loan ID: ' || rec.LoanID || ')');

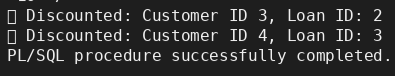
END LOOP;

COMMIT;

END;

/

**OUTPUT:**



**Scenario 2**: Promote to VIP if balance > 10000

ALTER TABLE Customers ADD (IsVIP CHAR(1) DEFAULT 'N');

BEGIN

FOR rec IN (SELECT CustomerID, Name FROM Customers WHERE Balance > 10000 AND IsVIP != 'Y') LOOP

UPDATE Customers SET IsVIP = 'Y' WHERE CustomerID = rec.CustomerID;

DBMS\_OUTPUT.PUT\_LINE(' VIP Promoted: ' || rec.Name);

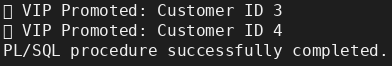
END LOOP;

COMMIT;

END;

/

**OUTPUT:**



**Scenario 3:** Loan due in next 30 days

BEGIN

FOR rec IN (

SELECT l.LoanID, c.Name, l.EndDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

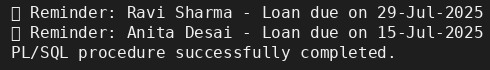
DBMS\_OUTPUT.PUT\_LINE('📅 Reminder: ' || rec.Name || ' - Loan due on ' || TO\_CHAR(rec.EndDate, 'DD-Mon-YYYY'));

END LOOP;

END;

/

**OUTPUT:**



**Exercise 3: Stored Procedures**

**SCHEMA CREATION (includes IsVIP column )**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE,

IsVIP CHAR(1) DEFAULT 'N'

);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Transactions (

TransactionID NUMBER PRIMARY KEY,

AccountID NUMBER,

TransactionDate DATE,

Amount NUMBER,

TransactionType VARCHAR2(10),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

**SAMPLE DATA FOR PROCEDURES**

INSERT INTO Accounts VALUES (101, 1, 'Savings', 2000, SYSDATE);

INSERT INTO Accounts VALUES (102, 2, 'Savings', 3000, SYSDATE);

INSERT INTO Employees VALUES (3, 'Carol Taylor', 'Analyst', 50000, 'Finance', TO\_DATE('2018-04-15', 'YYYY-MM-DD'));

INSERT INTO Employees VALUES (4, 'David Kumar', 'Analyst', 52000, 'Finance', TO\_DATE('2019-01-12', 'YYYY-MM-DD'));

**Scenario 1:** Monthly interest processing for Savings Accounts

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc IN (SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings') LOOP

UPDATE Accounts

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

LastModified = SYSDATE

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE('💰 Interest Applied: Account ID ' || acc.AccountID || ' | New Balance: ' || TO\_CHAR(acc.Balance \* 1.01, 'FM9999.00'));

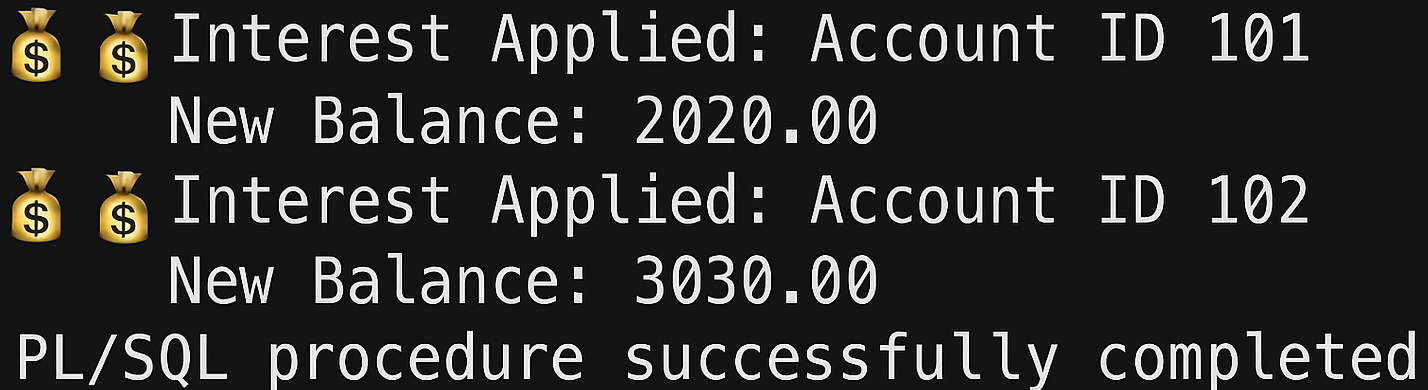
END LOOP;

COMMIT;

END;

/

**OUTPUT:**



**Scenario 2:** Update bonus for department employees

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) AS

BEGIN

FOR emp IN (SELECT EmployeeID, Name, Salary FROM Employees WHERE Department = p\_department) LOOP

UPDATE Employees

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

WHERE EmployeeID = emp.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE('🎁 Bonus Applied: ' || emp.Name || ' | New Salary: ' || TO\_CHAR(emp.Salary \* (1 + p\_bonus\_percent / 100), 'FM999999.00'));

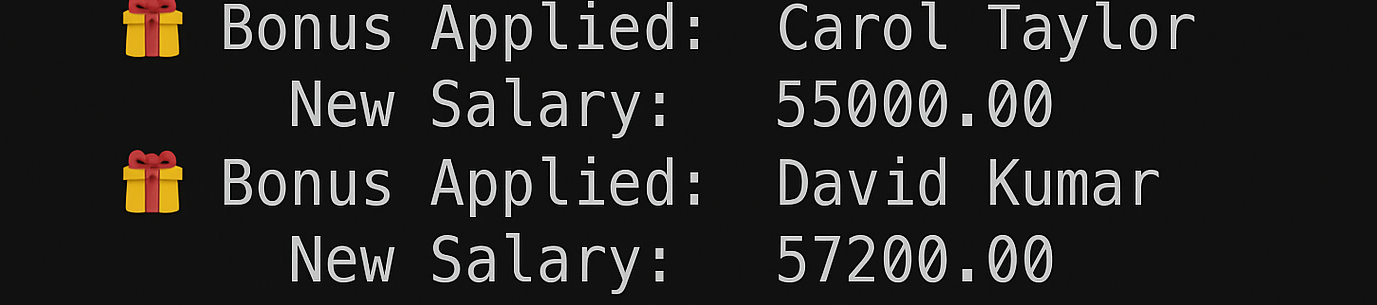
END LOOP;

COMMIT;

END;

/

**OUTPUT:**



**Scenario 3:** Fund transfer between accounts

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) AS

v\_balance NUMBER;

BEGIN

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

IF v\_balance < p\_amount THEN

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

END IF;

UPDATE Accounts

SET Balance = Balance - p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_from\_account;

UPDATE Accounts

SET Balance = Balance + p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_to\_account;

INSERT INTO Transactions VALUES (SEQ\_Transactions.NEXTVAL, p\_from\_account, SYSDATE, p\_amount, 'Transfer');

INSERT INTO Transactions VALUES (SEQ\_Transactions.NEXTVAL, p\_to\_account, SYSDATE, p\_amount, 'Deposit');

DBMS\_OUTPUT.PUT\_LINE('✅ Transfer of $' || p\_amount || ' from Account ' || p\_from\_account || ' to Account ' || p\_to\_account || ' successful.');

COMMIT;

END;

/

**OUTPUT:**

**EXEC TransferFunds(101, 102, 500);**

