**PL/SQL Programming**

**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.

**Create table Customer**

CREATE TABLE CUSTOMERS (

CUSTOMERID NUMBER PRIMARY KEY,

NAME VARCHAR2(100),

DOB DATE,

BALANCE NUMBER,

LASTMODIFIED DATE

);

**Create table Loans**

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 Customers val1**

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

VALUES (1, 'Subhiksha', TO\_DATE('1985-05-15', 'YYYY-MM-DD'), 12000, SYSDATE);

**Insert into Customers val2**

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

VALUES (2, 'Yega', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 15000, SYSDATE);

**Insert into Customers val3**

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

VALUES (3, 'Saranya', TO\_DATE('1996-08-15', 'YYYY-MM-DD'), 9500, SYSDATE);

**Insert into Customers val4**

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

VALUES (4, 'Sam', TO\_DATE('1950-04-10', 'YYYY-MM-DD'), 3000, SYSDATE);

**Insert into Loans val1**

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES (1, 1, 'Savings', 12000, SYSDATE);

**Insert into Loans val2**

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES (2, 2, 'Checking', 15000, SYSDATE);

**Insert into Loans val3**

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES (3, 3, 'Checking', 9500, SYSDATE);

.**Insert into Loans val4**

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES (4, 4, 'Savings', 3000, SYSDATE);

**UPDATE CUSTOMERS**

ALTER TABLE Customers ADD (Age NUMBER);

ALTER TABLE Customers ADD (LoanInterestRate NUMBER);

.

UPDATE Customers

SET Age = FLOOR(MONTHS\_BETWEEN(SYSDATE, DOB) / 12);

UPDATE Customers

SET LoanInterestRate = 10.0

WHERE LoanInterestRate IS NULL;

**Scenario 1 : Control Structures – Loan Discount For 60+ :**

SET SERVEROUTPUT ON;

BEGIN

FOR cust\_rec IN (

SELECT CustomerID, LoanInterestRate

FROM Customers

WHERE Age > 60

) LOOP

UPDATE Customers

SET LoanInterestRate = LoanInterestRate - 1

WHERE CustomerID = cust\_rec.CustomerID;

END LOOP;

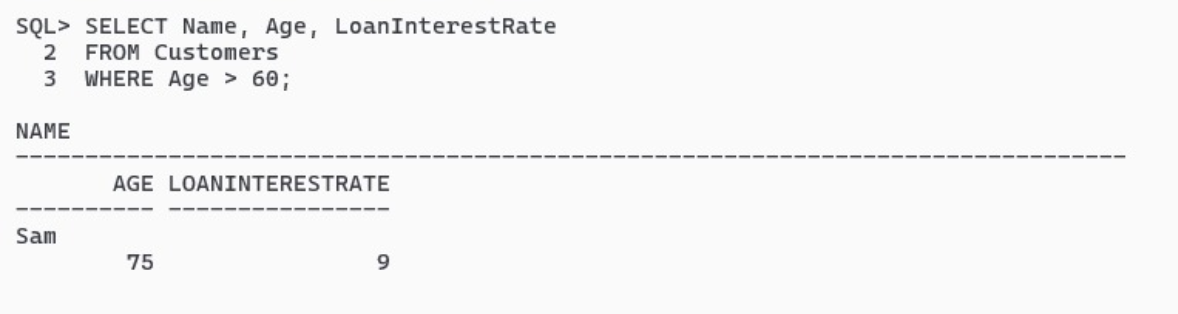
COMMIT;

DBMS\_OUTPUT.PUT\_LINE('1% discount applied to customers above 60 years old.');

END;

/

**Output:**

****

**Scenario 2 : Mark VIP Customers**

**ALTER TABLE CUSTOMER:**

ALTER TABLE Customers ADD (IsVIP VARCHAR2(5));

UPDATE Customers SET IsVIP = 'FALSE';

SET SERVEROUTPUT ON;

BEGIN

FOR cust\_rec IN (

SELECT CustomerID

FROM Customers

WHERE Balance > 10000

) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust\_rec.CustomerID;

END LOOP;

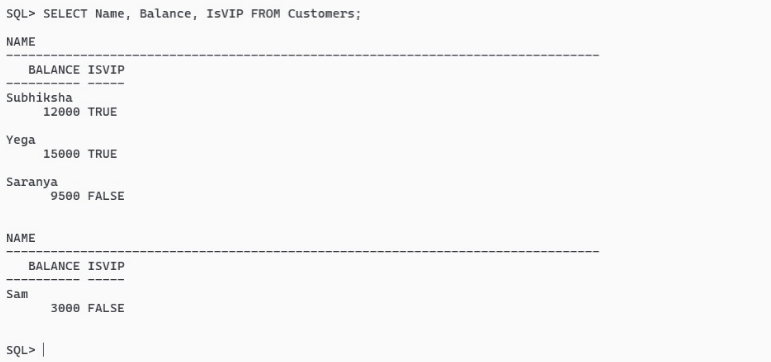
COMMIT;

DBMS\_OUTPUT.PUT\_LINE('VIP status updated for customers with balance over $10,000.');

END;

/

**Output :**

****

**Scenario 3 : Loan Due To Be Reminded**

SET SERVEROUTPUT ON;

BEGIN

FOR loan\_rec IN (

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

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

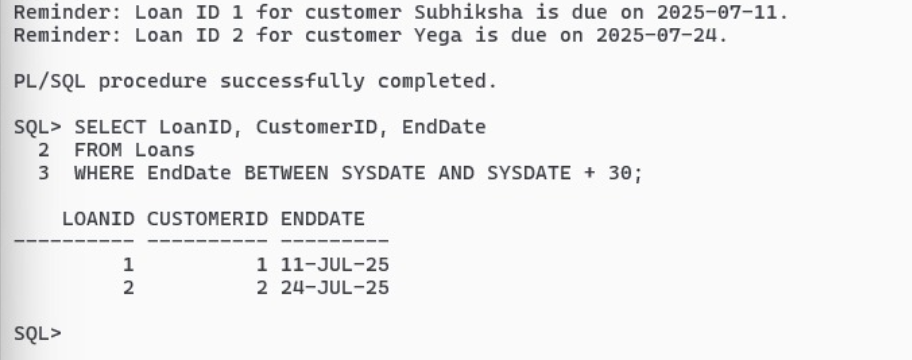
DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan\_rec.LoanID ||' for customer ' || loan\_rec.Name ||' is due on ' || TO\_CHAR(loan\_rec.EndDate, 'YYYY-MM-DD') || '.');

END LOOP;

END;

/

**Output :**

****

**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.

**Create table Accounts**

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 val1**

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

VALUES (1, 1, 'Savings', 12000, SYSDATE);

**Insert into Accounts val2**

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

VALUES (2, 2, 'Checking', 15000, SYSDATE);

**Insert into Accounts val3**

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

VALUES (3, 3, 'Checking', 9500, SYSDATE);

**Insert into Accounts val4**

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

VALUES (4, 4, 'Savings', 3000, SYSDATE);

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

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc\_rec IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

) LOOP

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01) -- Apply 1% interest

WHERE AccountID = acc\_rec.AccountID;

END LOOP;

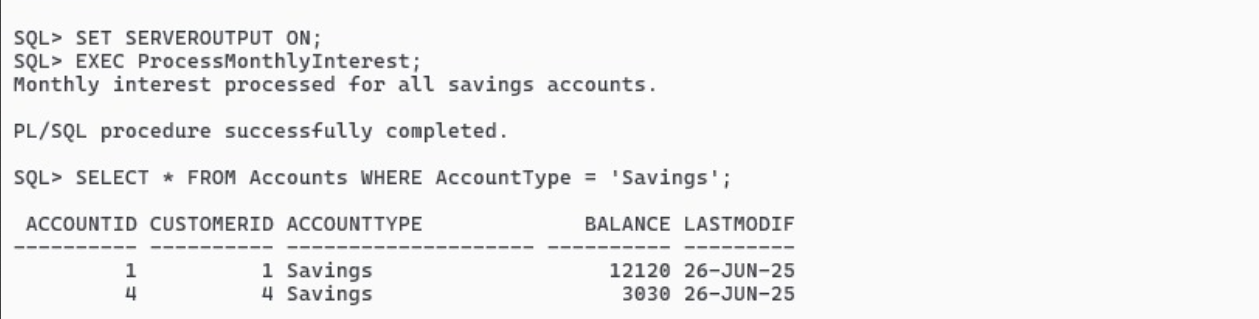
COMMIT;

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

END;

/

**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 TABLE EMPLOYEE**

CREATE TABLE Employees (

2 EmployeeID NUMBER PRIMARY KEY,

3 Name VARCHAR2(100),

4 Position VARCHAR2(50),

5 Salary NUMBER,

6 Department VARCHAR2(50),

7 HireDate DATE

8 );

**INSERT EMPLOYEE VAL1**

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

VALUES (101, 'Subhiksha', 'Manager', 75000, 'Finance', TO\_DATE('2018-03-15', 'YYYY-MM-DD'));

**INSERT EMPLOYEE VAL2**

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

VALUES (102, 'Yega', 'Clerk', 30000, 'Customer Service', TO\_DATE('2020-06-10', 'YYYY-MM-DD'));

**INSERT EMPLOYEE VAL3**

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

VALUES (103, 'Sam', 'Analyst', 55000, 'Risk', TO\_DATE('2019-11-05', 'YYYY-MM-DD'));

.

**INSERT EMPLOYEE VAL4**

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

VALUES (104, 'Aisu', 'Assistant', 25000, 'HR', TO\_DATE('2021-01-25', 'YYYY-MM-DD'));

.

**INSERT EMPLOYEE VAL5**

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

VALUES (105, 'Harshini', 'Developer', 60000, 'IT', TO\_DATE('2022-08-18', 'YYYY-MM-DD'));

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

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE Employees

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

WHERE Department = dept\_name;

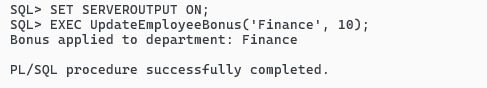
COMMIT;

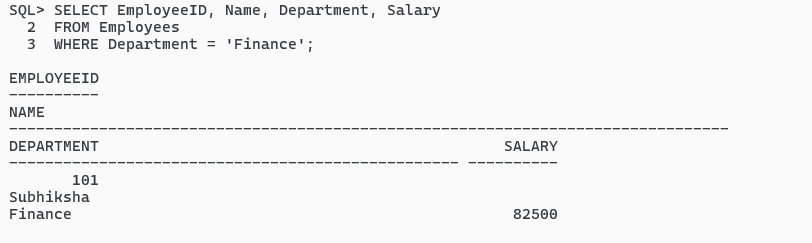
DBMS\_OUTPUT.PUT\_LINE('Bonus applied to department: ' || dept\_name);

END;

/

**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 Table Transactions**

CREATE TABLE Transactions (

TransactionID NUMBER PRIMARY KEY,

AccountID NUMBER,

TransactionDate DATE,

Amount NUMBER,

TransactionType VARCHAR2(10),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

**INSERT TRANSACTIONS VAL1:**

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)VALUES (1, 1, TO\_DATE('2025-06-01', 'YYYY-MM-DD'), 5000, 'Credit');

**INSERT TRANSACTIONS VAL2:**

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)VALUES (2, 1, TO\_DATE('2025-06-05', 'YYYY-MM-DD'), 2000, 'Debit');

**INSERT TRANSACTIONS VAL3:**

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)VALUES (3, 2, TO\_DATE('2025-06-10', 'YYYY-MM-DD'), 7000, 'Credit');

**INSERT TRANSACTIONS VAL4:**

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)VALUES (4, 3, TO\_DATE('2025-06-12', 'YYYY-MM-DD'), 3000, 'Credit');

**INSERT TRANSACTIONS VAL5:**

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)VALUES (5, 4, TO\_DATE('2025-06-15', 'YYYY-MM-DD'), 1500, 'Debit');

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

CREATE OR REPLACE PROCEDURE TransferFunds (

source\_acc\_id IN NUMBER,

target\_acc\_id IN NUMBER,

amount IN NUMBER

) AS

source\_balance NUMBER;

BEGIN

SELECT Balance INTO source\_balance

FROM Accounts

WHERE AccountID = source\_acc\_id

FOR UPDATE;

IF source\_balance < amount THEN

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

END IF;

UPDATE Accounts

SET Balance = Balance - amount

WHERE AccountID = source\_acc\_id;

UPDATE Accounts

SET Balance = Balance + amount

WHERE AccountID = target\_acc\_id;

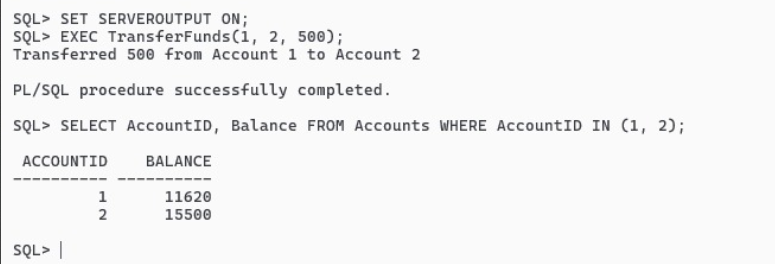
COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || amount || ' from Account ' || source\_acc\_id || ' to Account ' || target\_acc\_id);

END;

/

**OUTPUT:**

****