**Creating the tables and inserting data:**

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

);

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 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 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'));

ALTER TABLE Customers ADD IsVIP char(1)

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

VALUES (99, 'Test Customer', TO\_DATE('1960-01-01','YYYY-MM-DD'), 9000, SYSDATE, 'N');

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

VALUES (999, 99, 10000, 5.0, SYSDATE - 100, SYSDATE + 10);

**Exercise 1: Control Structures**

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

**Code:**

BEGIN

FOR i IN (SELECT CustomerID, DOB FROM Customers) LOOP

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

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = i.CustomerID;

END IF;

END LOOP;

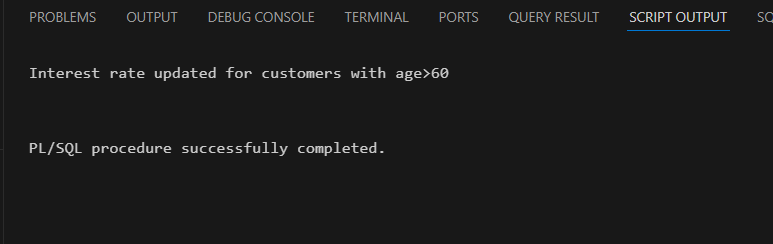
COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Interest rate updated for customers with age>60');

END;

/

**Output:**

****

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

Code:

BEGIN

FOR i IN (SELECT CUSTOMERID, BALANCE FROM CUSTOMERS) LOOP

IF i.BALANCE >10000 THEN

UPDATE CUSTOMERS

SET IsVIP = 'Y'

WHERE CUSTOMERID= i.CUSTOMERID;

ELSE

UPDATE CUSTOMERS

SET IsVIP = 'N'

WHERE CUSTOMERID= i.CUSTOMERID;

END IF;

END LOOP;

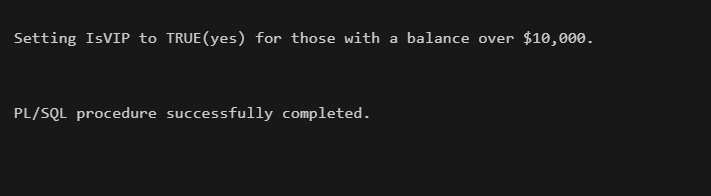
COMMIT;

DBMS\_OUTPUT.PUT\_LINE( 'Setting IsVIP to TRUE(yes) for those with a balance over $10,000.');

END;

/

**Output:**

****

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

**Code:**

BEGIN

FOR i 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: ' || i.Name || ' has a loan due on ' || TO\_CHAR(i.EndDate, 'DD-Mon-YYYY')

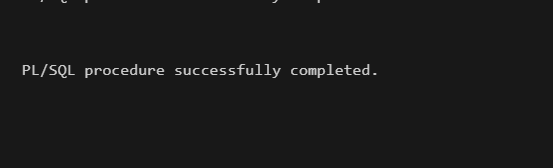
);

END LOOP;

END;

/

**Output:**

****

**Exercise 3: Stored Procedures**

**Scenario 1:** The bank needs to process monthly interest for all 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 + (acc.Balance \* 0.01),

LastModified = SYSDATE

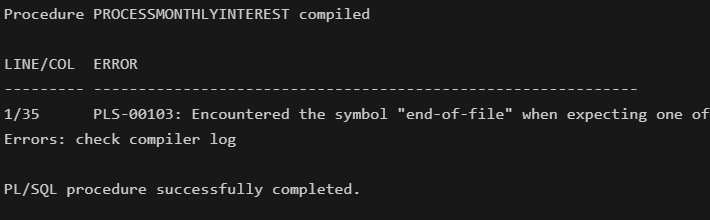
WHERE AccountID = acc.AccountID;

END LOOP;

COMMIT;

END;

/



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

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE Employees

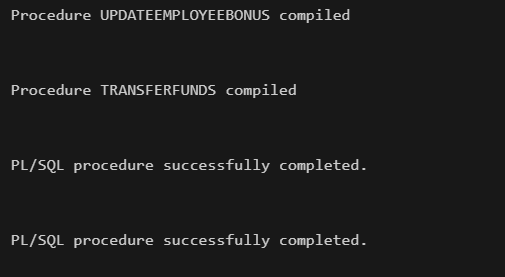
SET Salary = Salary + (Salary \* p\_bonus\_percent / 100)

WHERE Department = p\_department;

COMMIT;

END;

/



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

CREATE OR REPLACE PROCEDURE TransferFunds (

acc\_from IN NUMBER,

acc\_to IN NUMBER,

amt IN NUMBER

) AS

bal NUMBER;

BEGIN

SELECT Balance INTO bal

FROM Accounts

WHERE AccountID = acc\_from

FOR UPDATE;

IF bal < amt THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Balance less than limit.');

END IF;

UPDATE Accounts

SET Balance = Balance - amt,

LastModified = SYSDATE

WHERE AccountID = acc\_from;

UPDATE Accounts

SET Balance = Balance + amt,

LastModified = SYSDATE

WHERE AccountID = acc\_to;

COMMIT;

END;

/

BEGIN

TransferFunds(1, 2, 500);

END;

/

