# 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.

**Code:**

BEGIN

FOR cust IN (

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

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

) LOOP

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

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = cust.LoanID;

END IF;

END LOOP;

COMMIT;

END;

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**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.

**Code:**

Adding the IsVIP column to Customers table

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

BEGIN

FOR cust IN (

SELECT CustomerID, Balance FROM Customers

) LOOP

IF cust.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'Y'

WHERE CustomerID = cust.CustomerID;

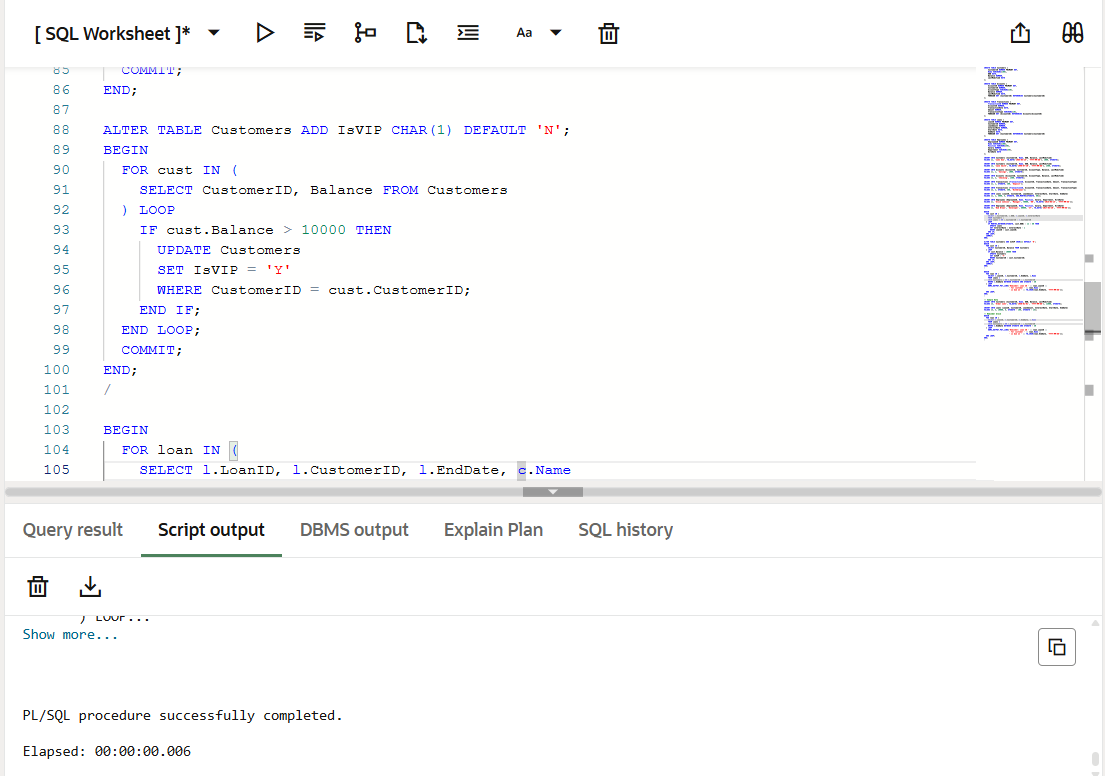
END IF;

END LOOP;

COMMIT;

END;

**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.

**Code:**

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

VALUES (3, 'Elder John', TO\_DATE('1950-01-01', 'YYYY-MM-DD'), 12000, SYSDATE);

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

VALUES (3, 3, 10000, 6, SYSDATE - 100, SYSDATE + 15);

BEGIN

FOR loan IN (

SELECT l.LoanID, l.CustomerID, 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.LoanID ||

' for customer ' || loan.Name ||

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

END LOOP;

END;

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**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.

**Code:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

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

LastModified = SYSDATE

WHERE AccountType = 'Savings';

COMMIT;

END;

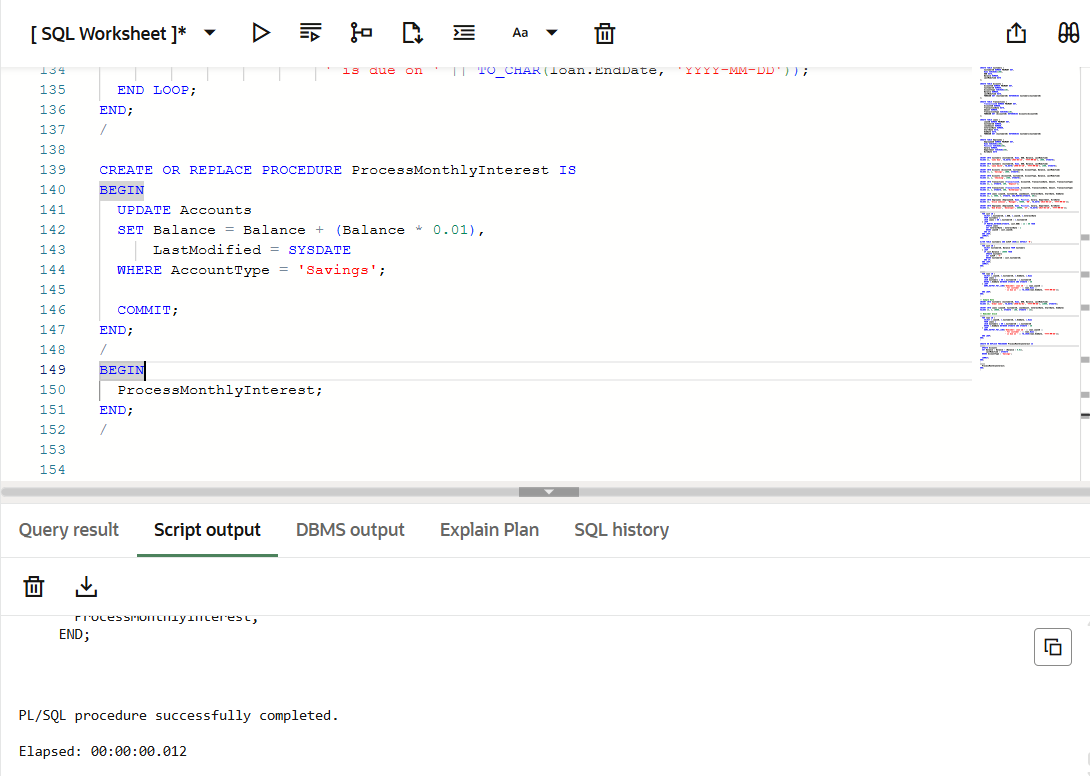
/

BEGIN

ProcessMonthlyInterest;

END;

/



**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.

**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;

COMMIT;

END;

/

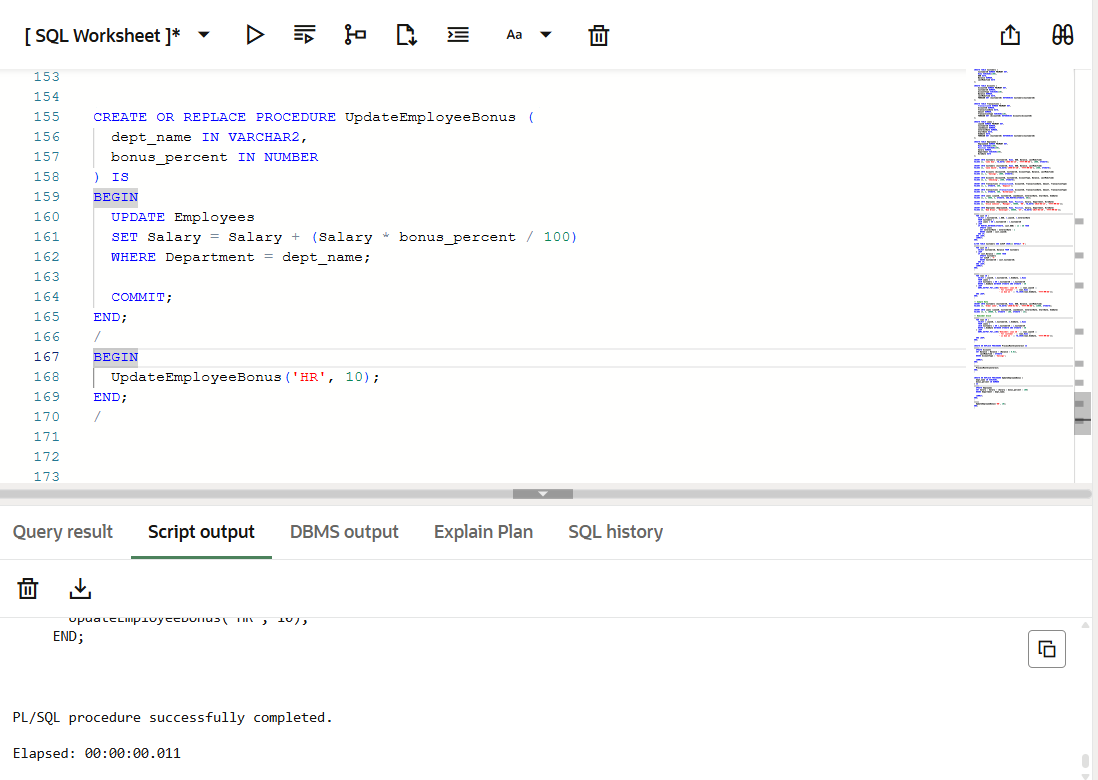
BEGIN

UpdateEmployeeBonus('HR', 10);

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.

**Code:**

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account\_id IN NUMBER,

to\_account\_id IN NUMBER,

amount IN NUMBER

) IS

v\_balance NUMBER;

insufficient\_balance EXCEPTION;

BEGIN

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = from\_account\_id;

IF v\_balance < amount THEN

RAISE insufficient\_balance;

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;

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

VALUES ((SELECT NVL(MAX(TransactionID), 0) + 1 FROM Transactions), from\_account\_id, SYSDATE, amount, 'Debit');

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

VALUES ((SELECT NVL(MAX(TransactionID), 0) + 1 FROM Transactions), to\_account\_id, SYSDATE, amount, 'Credit');

COMMIT;

EXCEPTION

WHEN insufficient\_balance THEN

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

END;

/

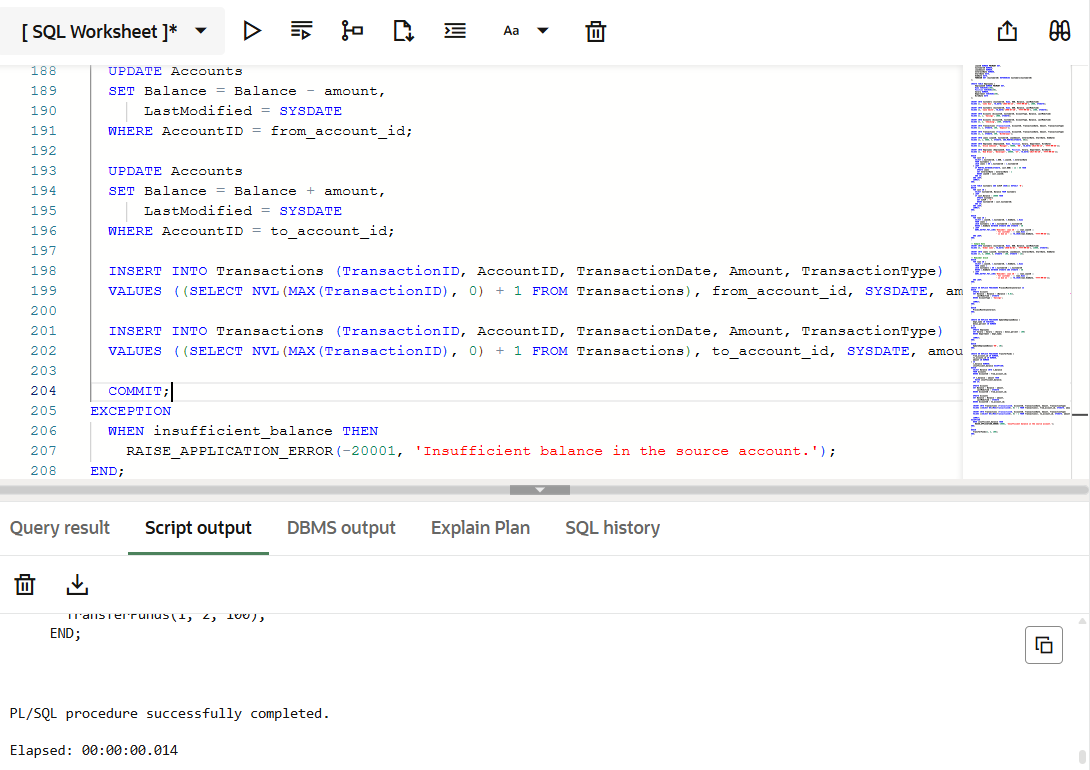
BEGIN

TransferFunds(1, 2, 100);

END;

/

**OUTPUT:**

****