**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\_rec IN (SELECT CustomerID, DOB FROM Customers) LOOP

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

UPDATE Loans

SET InterestRate = InterestRate - 1

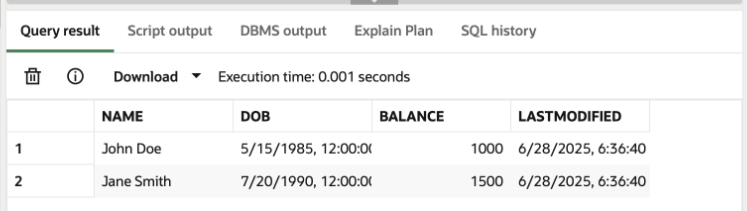
WHERE CustomerID = cust\_rec.CustomerID;

END IF;

END LOOP;

COMMIT;

END;



**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:

ALTER TABLE Customers ADD IsVIP VARCHAR2(5);

BEGIN

FOR cust\_rec IN (SELECT CustomerID, Balance FROM Customers) LOOP

IF cust\_rec.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust\_rec.CustomerID;

ELSE

UPDATE Customers

SET IsVIP = 'FALSE'

WHERE CustomerID = cust\_rec.CustomerID;

END IF;

END LOOP;

COMMIT;

END;

SELECT \* FROM CUSTOMERS;

A screenshot of a computer

AI-generated content may be incorrect.

**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:

BEGIN

FOR loan\_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: Dear ' || loan\_rec.Name ||

', your loan (Loan ID: ' || loan\_rec.LoanID ||

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

'. Please ensure timely payment.'

);

END LOOP;

END;

SELECT \* FROM LOANS;

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

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;

DBMS\_OUTPUT.PUT\_LINE('Interest added for Account ID: ' || acc.AccountID);

END LOOP;

COMMIT;

END;

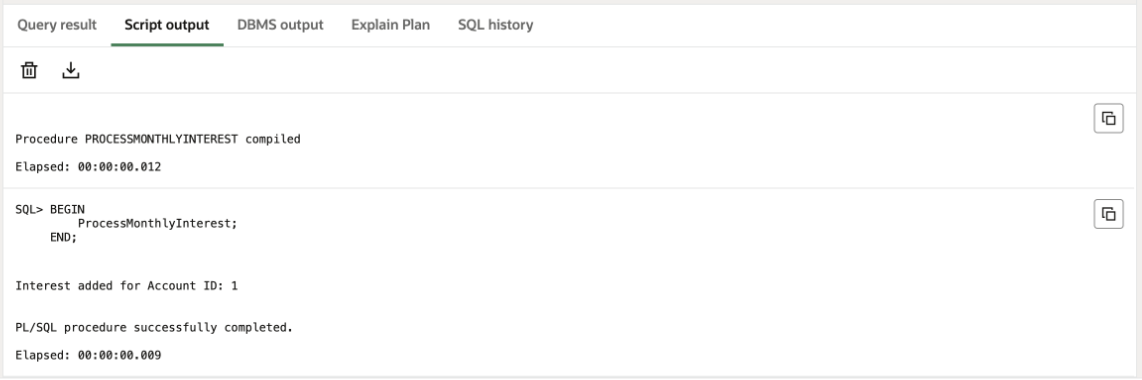
/

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 (

p\_dept IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

FOR emp IN (

SELECT EmployeeID, Salary FROM Employees WHERE Department = p\_dept

) LOOP

UPDATE Employees

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

WHERE EmployeeID = emp.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to Employee ID: ' || emp.EmployeeID);

END LOOP;

COMMIT;

END;

/

BEGIN

UpdateEmployeeBonus('IT', 10);

END;

/

A screenshot of a computer

AI-generated content may be incorrect.

**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 (

p\_from\_acc IN NUMBER,

p\_to\_acc IN NUMBER,

p\_amt IN NUMBER

) IS

v\_bal NUMBER;

BEGIN

-- Check balance

SELECT Balance INTO v\_bal FROM Accounts WHERE AccountID = p\_from\_acc;

IF v\_bal < p\_amt THEN

RAISE\_APPLICATION\_ERROR(-20010, 'Insufficient funds in source account.');

END IF;

-- Debit and Credit

UPDATE Accounts SET Balance = Balance - p\_amt WHERE AccountID = p\_from\_acc;

UPDATE Accounts SET Balance = Balance + p\_amt WHERE AccountID = p\_to\_acc;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || p\_amt || ' from ' || p\_from\_acc || ' to ' || p\_to\_acc);

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

/

BEGIN

TransferFunds(2, 3, 500);

END;

/

A screenshot of a computer

AI-generated content may be incorrect.