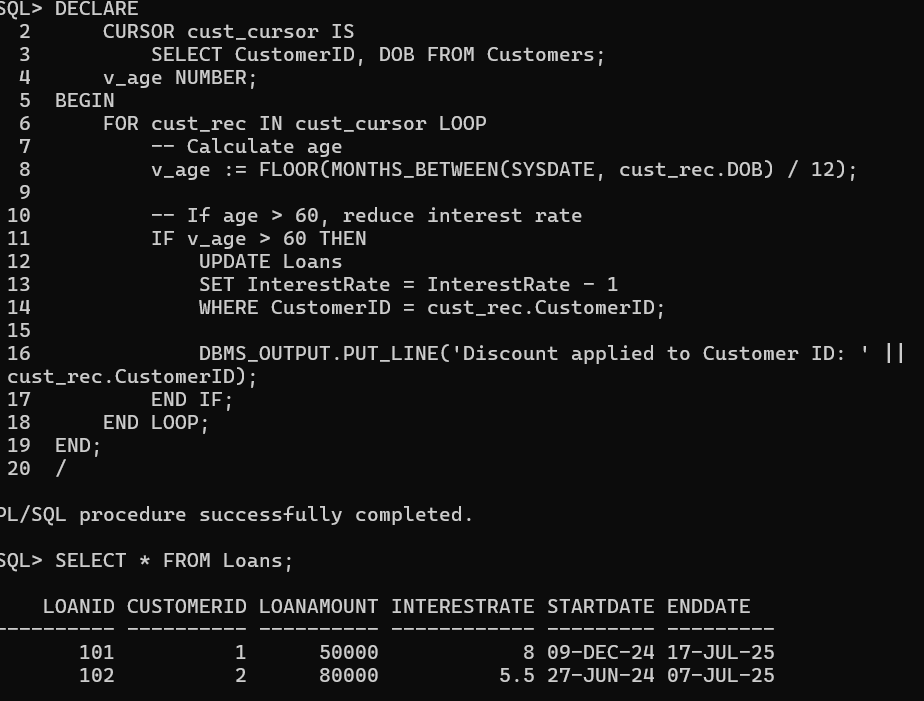
**Exercise 1: Control Structures**

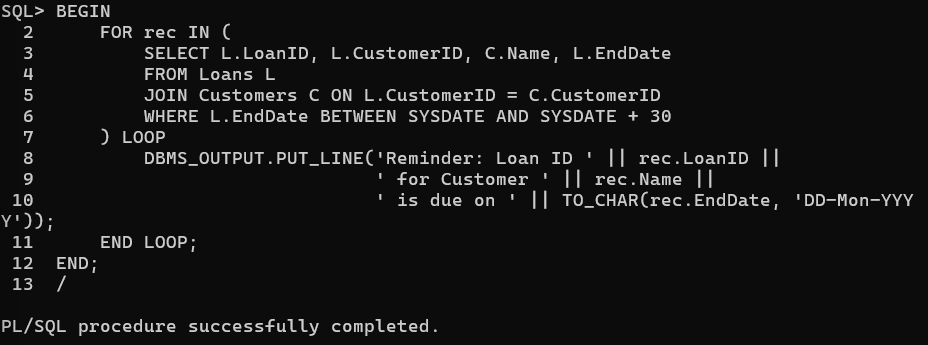
**Scenario 1:**

****

**Scenario 2:**

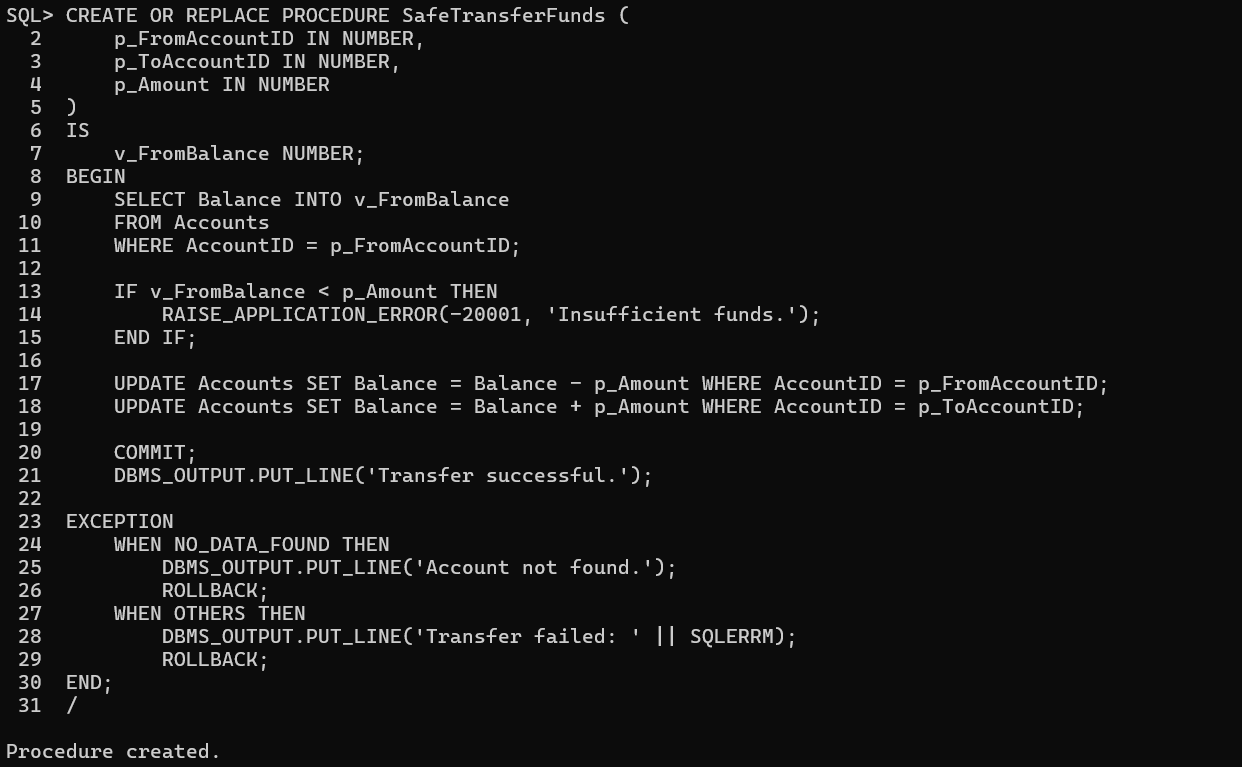
****

**Scenario 3:**

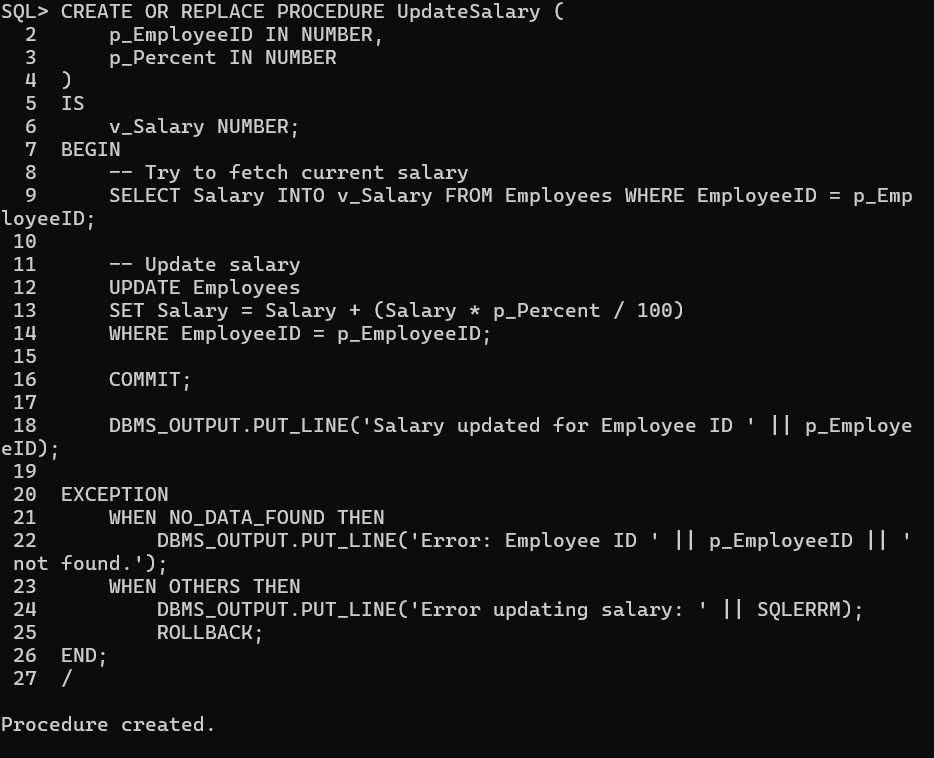
****

**Exercise 2: Error Handling**

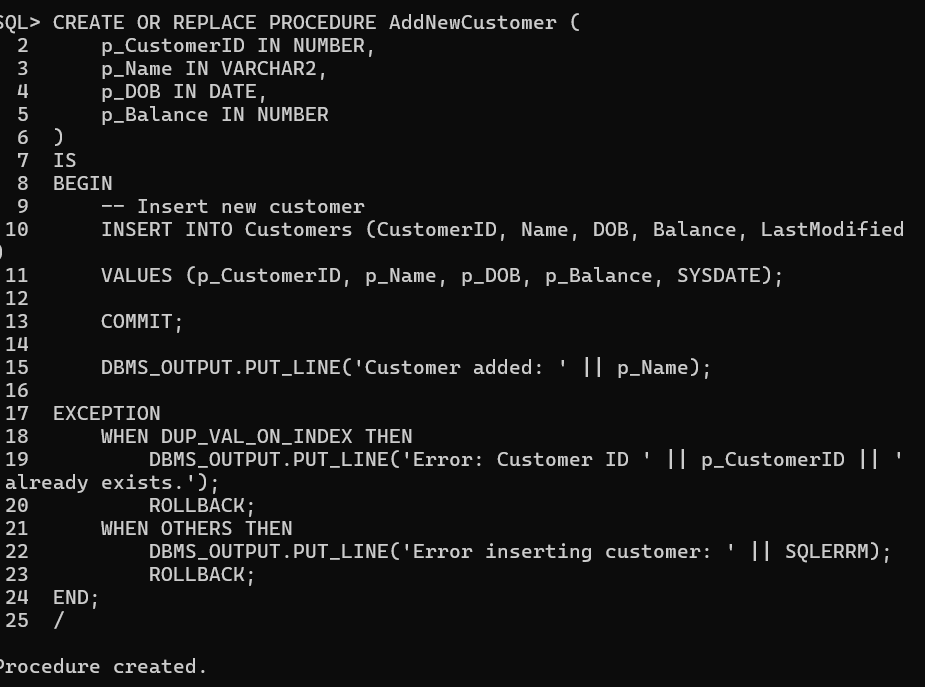
**Scenario 1:**

****

**Scenario 2:**

****

**Scenario 3:**

****

**Exercise 3: Stored Procedures**

**Scenario 1:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest

IS

CURSOR savings\_cursor IS

SELECT AccountID, Balance FROM Accounts

WHERE AccountType = 'Savings';

v\_new\_balance NUMBER;

BEGIN

FOR acc IN savings\_cursor LOOP

v\_new\_balance := acc.Balance + (acc.Balance \* 0.01);

UPDATE Accounts

SET Balance = v\_new\_balance,

LastModified = SYSDATE

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Interest applied to Account ID ' || acc.AccountID ||

': New Balance = ' || v\_new\_balance);

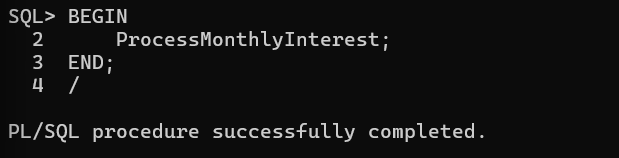
END LOOP;

COMMIT;

END;

/

**OUTPUT:**



**Scenario 2:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_Department IN VARCHAR2,

p\_BonusPercent IN NUMBER

)

IS

CURSOR emp\_cursor IS

SELECT EmployeeID, Salary FROM Employees

WHERE Department = p\_Department;

v\_bonus\_salary NUMBER;

BEGIN

FOR emp IN emp\_cursor LOOP

v\_bonus\_salary := emp.Salary + (emp.Salary \* p\_BonusPercent / 100);

UPDATE Employees

SET Salary = v\_bonus\_salary

WHERE EmployeeID = emp.EmployeeID;

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

': New Salary = ' || v\_bonus\_salary);

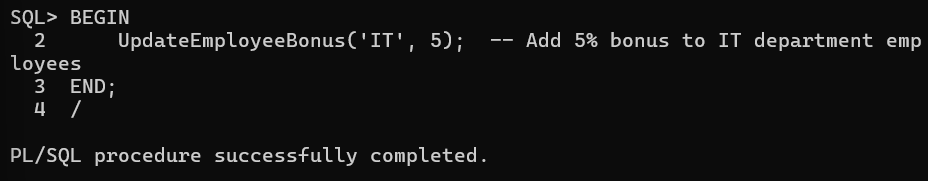
END LOOP;

COMMIT;

END;

/

**OUTPUT:**

****

**Scenario 3:**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_FromAccountID IN NUMBER,

p\_ToAccountID IN NUMBER,

p\_Amount IN NUMBER

)

IS

v\_balance NUMBER;

BEGIN

-- Check if source account has enough funds

SELECT Balance INTO v\_balance FROM Accounts

WHERE AccountID = p\_FromAccountID;

IF v\_balance < p\_Amount THEN

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

END IF;

-- Perform transfer

UPDATE Accounts

SET Balance = Balance - p\_Amount,

LastModified = SYSDATE

WHERE AccountID = p\_FromAccountID;

UPDATE Accounts

SET Balance = Balance + p\_Amount,

LastModified = SYSDATE

WHERE AccountID = p\_ToAccountID;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || p\_Amount ||

' completed from Account ' || p\_FromAccountID ||

' to Account ' || p\_ToAccountID);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('One or both Account IDs not found.');

ROLLBACK;

WHEN OTHERS THEN

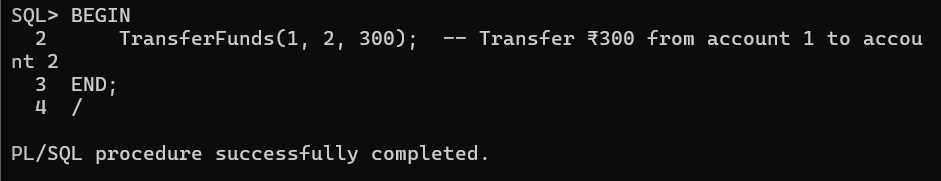
DBMS\_OUTPUT.PUT\_LINE('Transfer failed: ' || SQLERRM);

ROLLBACK;

END;

/

**OUTPUT:**

****

**Exercise 4: Functions**

**Scenario 1:**

CREATE OR REPLACE FUNCTION CalculateAge(p\_DOB DATE)

RETURN NUMBER

IS

v\_age NUMBER;

BEGIN

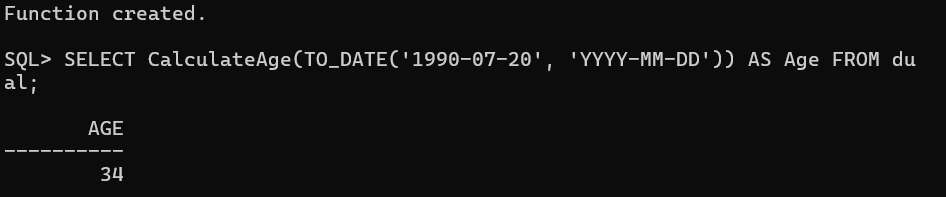
v\_age := FLOOR(MONTHS\_BETWEEN(SYSDATE, p\_DOB) / 12);

RETURN v\_age;

END;

/

**OUTPUT:**



**Scenario 2:** CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment (

p\_LoanAmount IN NUMBER,

p\_AnnualInterestRate IN NUMBER,

p\_Years IN NUMBER

)

RETURN NUMBER

IS

v\_monthlyRate NUMBER;

v\_months NUMBER;

v\_installment NUMBER;

BEGIN

v\_monthlyRate := p\_AnnualInterestRate / 12 / 100;

v\_months := p\_Years \* 12;

IF v\_monthlyRate = 0 THEN

v\_installment := p\_LoanAmount / v\_months;

ELSE

v\_installment := p\_LoanAmount \* v\_monthlyRate / (1 - POWER(1 + v\_monthlyRate, -v\_months));

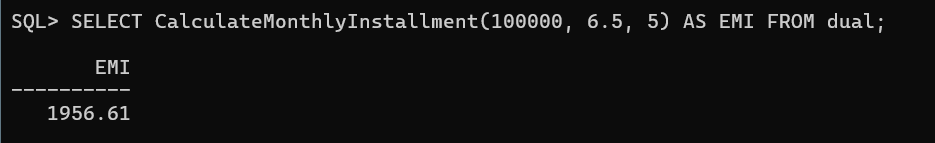
END IF;

RETURN ROUND(v\_installment, 2);

END;

/

**OUTPUT:**



**Scenario 3:**

CREATE OR REPLACE FUNCTION HasSufficientBalance (

p\_AccountID IN NUMBER,

p\_Amount IN NUMBER

)

RETURN BOOLEAN

IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_AccountID;

RETURN v\_balance >= p\_Amount;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN FALSE;

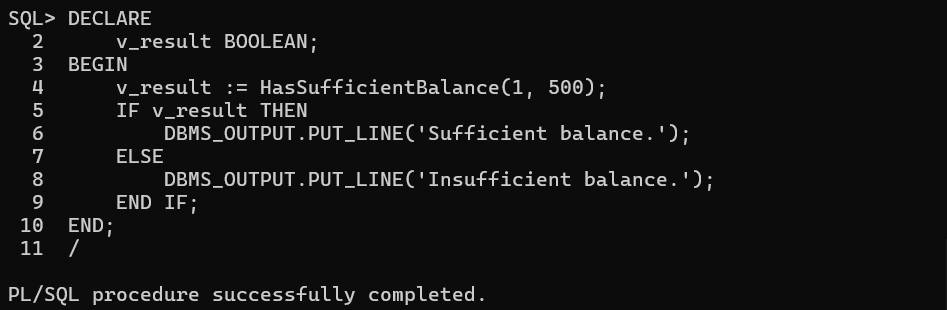
WHEN OTHERS THEN

RETURN FALSE;

END;

/

**OUTPUT:**

****

**Exercise 5: Triggers**

**Scenario 1:**

CREATE OR REPLACE TRIGGER UpdateCustomerLastModified

BEFORE UPDATE ON Customers

FOR EACH ROW

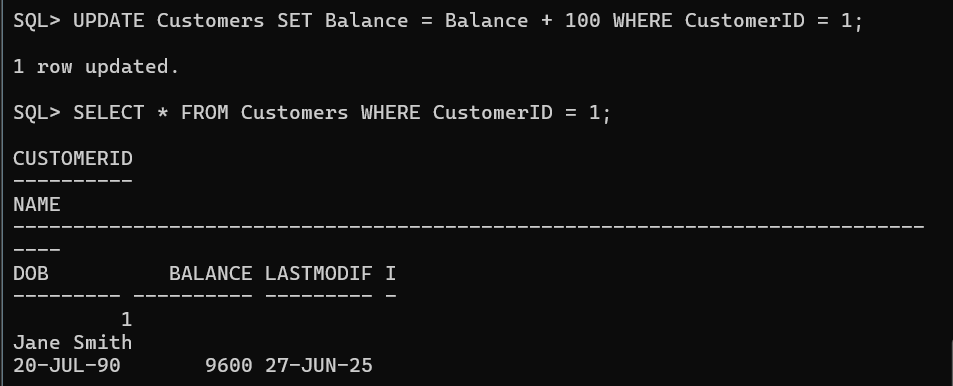
BEGIN

:NEW.LastModified := SYSDATE;

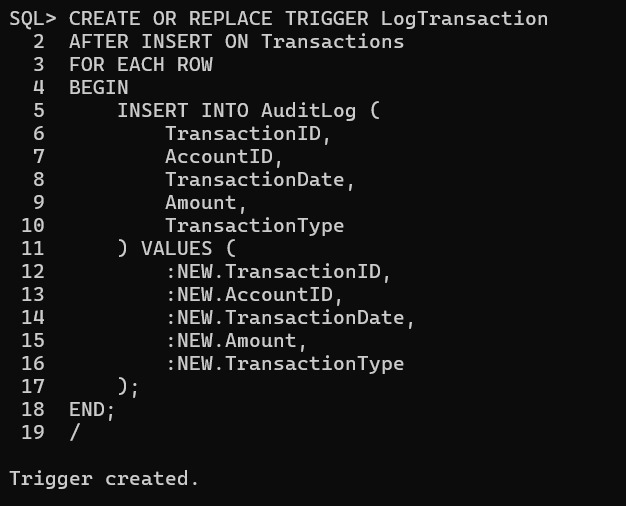
END;

/

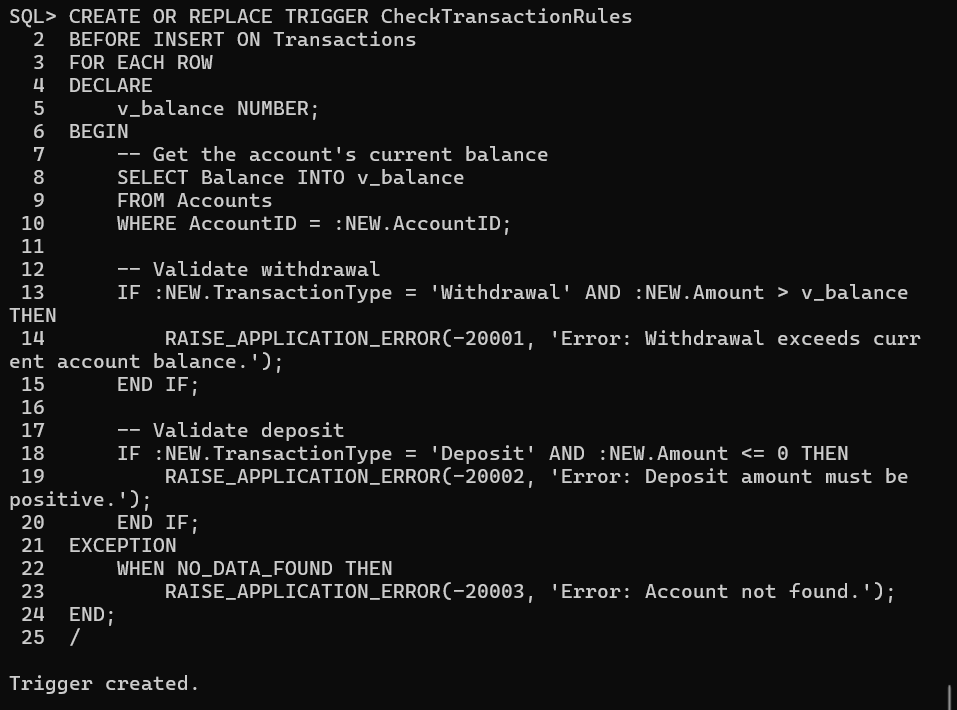
**OUTPUT:**

****

**Scenario 2:**

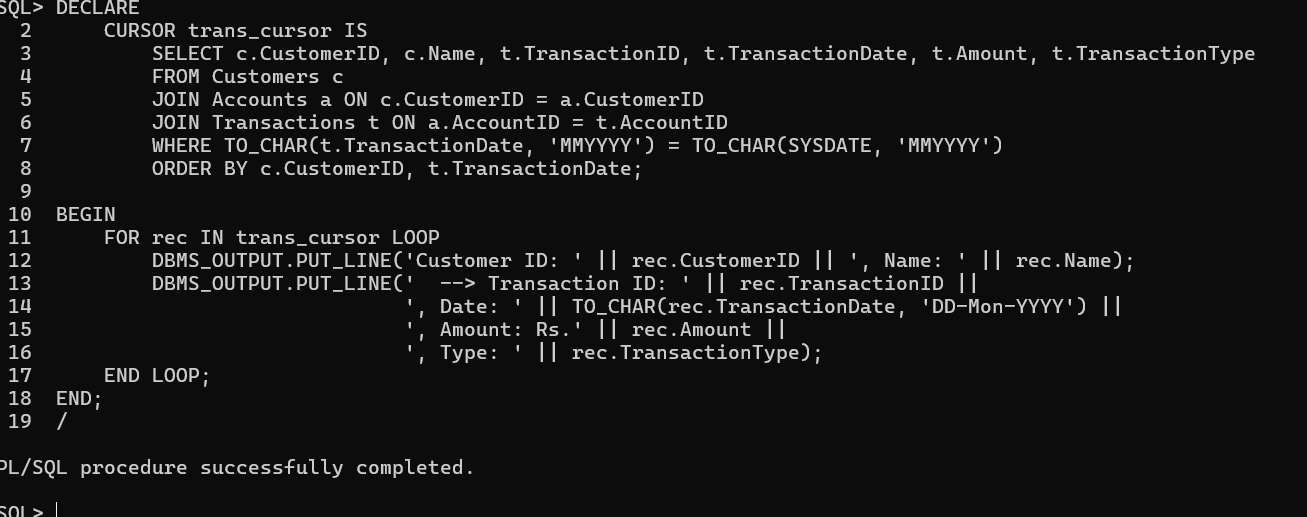
****

**Scenario 3:**

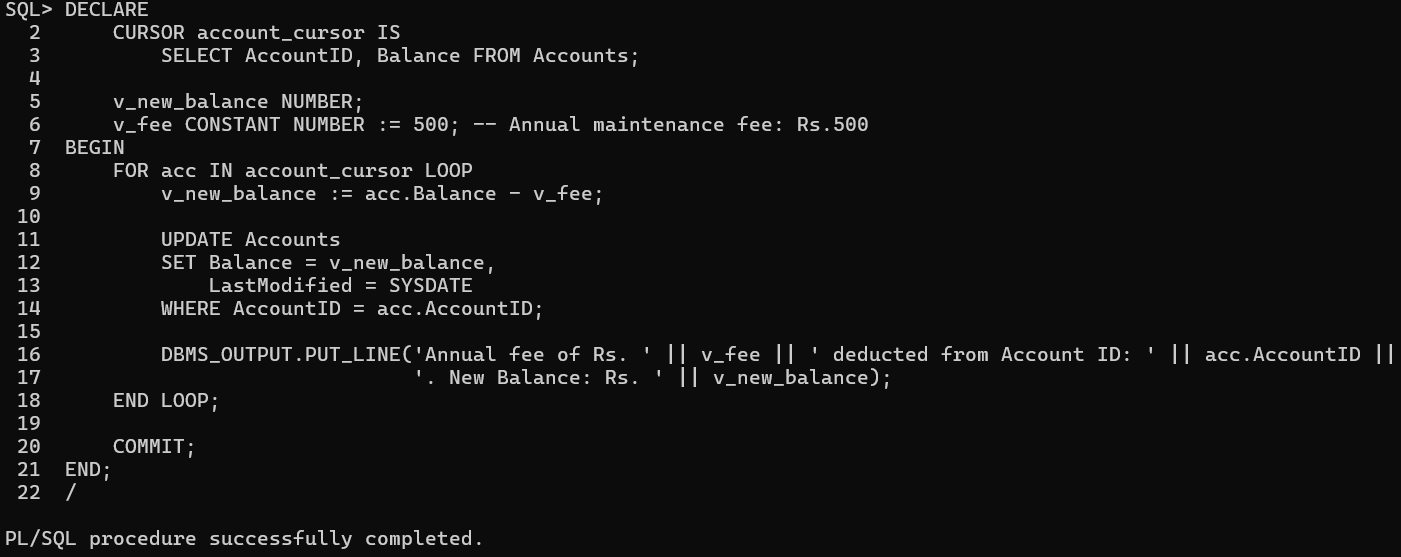
****

**Exercise 6: Cursors**

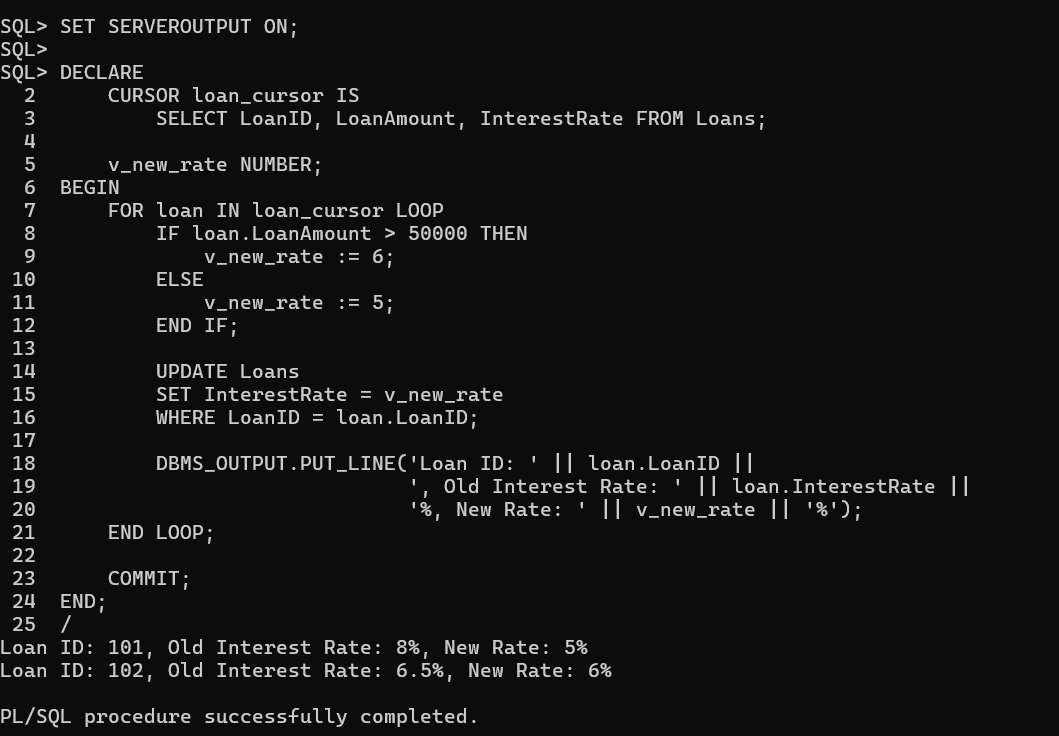
**Scenario 1:**

****

**Scenario 2:**

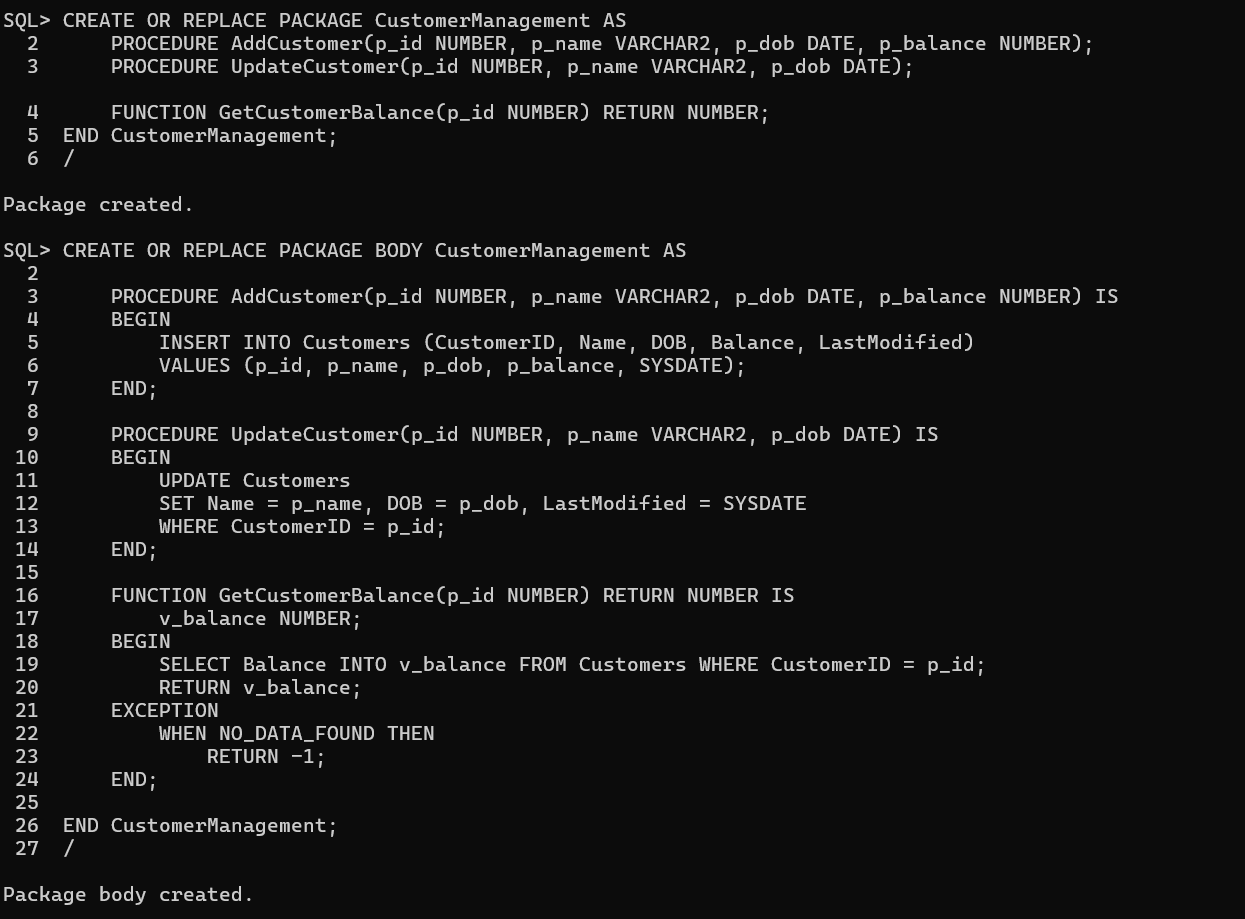
****

**Scenario 3:**

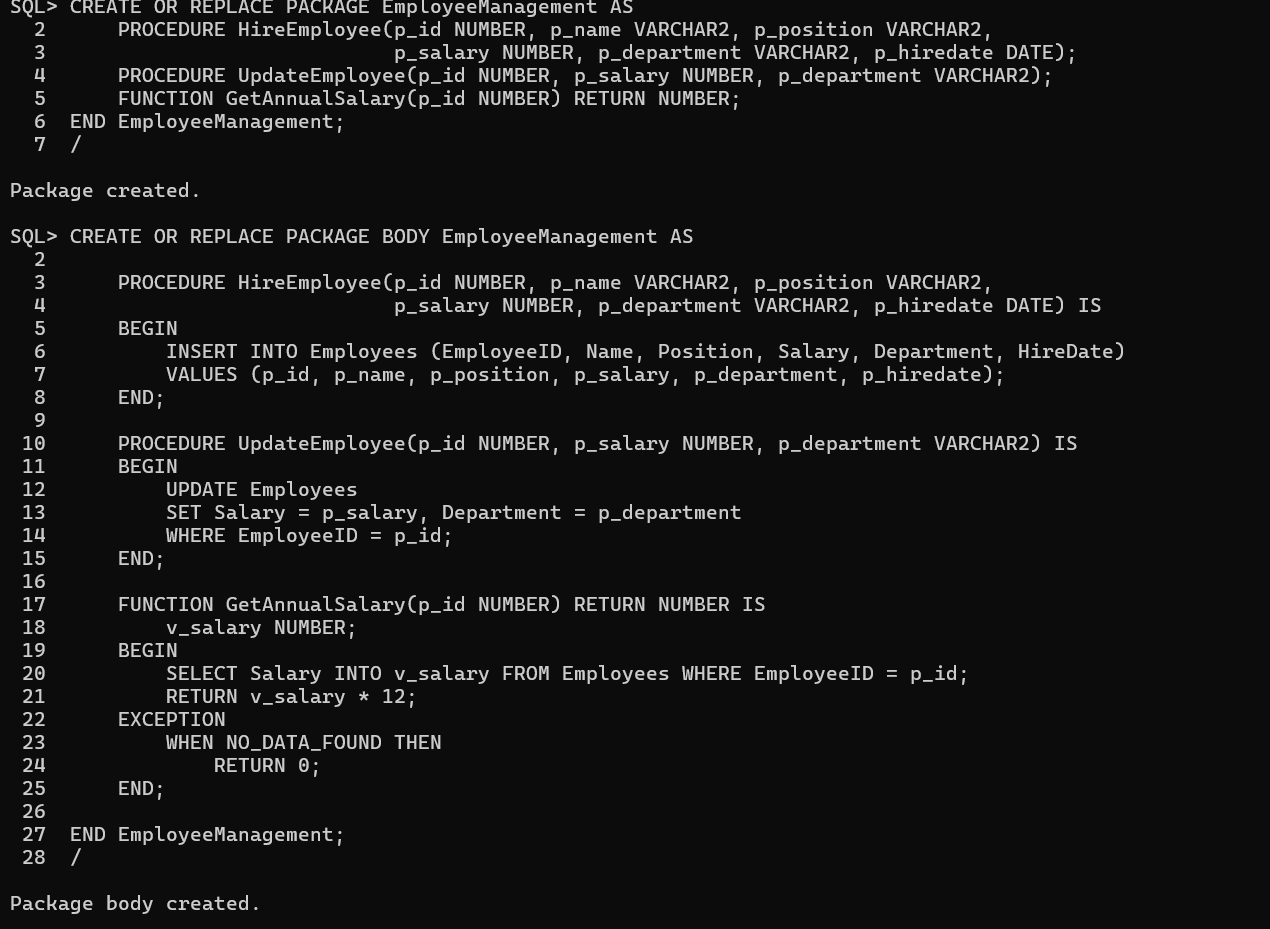
****

**Exercise 7: Packages**

**Scenario 1:**

****

**Scenario 2:**

****

**Scenario 3:**

