PL/SQL Exercises

# Exercise 1: Control Structures

## Creating tables:

-- 1. Drop tables if they exist (for repeated testing)

BEGIN

  EXECUTE IMMEDIATE 'DROP TABLE LOANS';

  EXECUTE IMMEDIATE 'DROP TABLE CUSTOMERS';

EXCEPTION

  WHEN OTHERS THEN

    NULL; -- ignore errors if tables don't exist

END;

/

-- 2. Create CUSTOMERS table

CREATE TABLE CUSTOMERS (

  CustomerID    NUMBER PRIMARY KEY,

  Name          VARCHAR2(100),

  Age           NUMBER,

  Balance       NUMBER(10, 2),

  IsVIP         VARCHAR2(5) DEFAULT 'FALSE'

);

-- 3. Create LOANS table

CREATE TABLE LOANS (

  LoanID        NUMBER PRIMARY KEY,

  CustomerID    NUMBER,

  InterestRate  NUMBER(5, 2),

  DueDate       DATE,

  FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID)

);

-- 4. Insert sample data into CUSTOMERS

INSERT INTO CUSTOMERS (CustomerID, Name, Age, Balance)

VALUES (1, 'Alice', 65, 12000);

INSERT INTO CUSTOMERS (CustomerID, Name, Age, Balance)

VALUES (2, 'Bob', 45, 8000);

INSERT INTO CUSTOMERS (CustomerID, Name, Age, Balance)

VALUES (3, 'Charlie', 70, 20000);

-- 5. Insert sample data into LOANS

INSERT INTO LOANS (LoanID, CustomerID, InterestRate, DueDate)

VALUES (101, 1, 7.5, SYSDATE + 15);

INSERT INTO LOANS (LoanID, CustomerID, InterestRate, DueDate)

VALUES (102, 2, 8.0, SYSDATE + 45);

INSERT INTO LOANS (LoanID, CustomerID, InterestRate, DueDate)

VALUES (103, 3, 6.5, SYSDATE + 10);

COMMIT;

## Scenario 1:

BEGIN

  FOR cust IN (

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

    FROM CUSTOMERS c

    JOIN LOANS l ON c.CustomerID = l.CustomerID

    WHERE c.Age > 60

  )

  LOOP

    UPDATE LOANS

    SET InterestRate = InterestRate - 1

    WHERE LoanID = cust.LoanID;

  END LOOP;

  COMMIT;

END;

/



## Scenario 2:

BEGIN

  FOR cust IN (

    SELECT CustomerID FROM CUSTOMERS WHERE Balance > 10000

  )

  LOOP

    UPDATE CUSTOMERS

    SET IsVIP = 'TRUE'

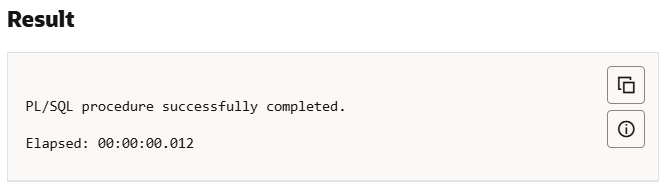
    WHERE CustomerID = cust.CustomerID;

  END LOOP;

  COMMIT;

END;

/



## Scenario 3:

BEGIN

  FOR loan\_rec IN (

    SELECT l.LoanID, l.DueDate, c.Name

    FROM LOANS l

    JOIN CUSTOMERS c ON l.CustomerID = c.CustomerID

    WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE + 30

  )

  LOOP

    DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan\_rec.LoanID ||

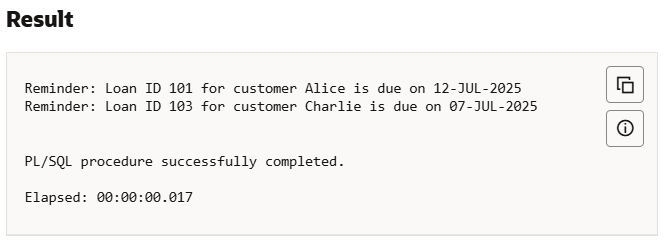
                         ' for customer ' || loan\_rec.Name ||

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

  END LOOP;

END;

/



# Exercise 3: Stored Procedures

## Creating tables:

BEGIN

  EXECUTE IMMEDIATE 'DROP TABLE ACCOUNTS CASCADE CONSTRAINTS';

EXCEPTION

  WHEN OTHERS THEN

    IF SQLCODE != -942 THEN RAISE; END IF;

END;

/

BEGIN

  EXECUTE IMMEDIATE 'DROP TABLE EMPLOYEES CASCADE CONSTRAINTS';

EXCEPTION

  WHEN OTHERS THEN

    IF SQLCODE != -942 THEN RAISE; END IF;

END;

/

CREATE TABLE ACCOUNTS (

  AccountID    NUMBER PRIMARY KEY,

  AccountType  VARCHAR2(20),  -- e.g., 'SAVINGS', 'CURRENT'

  Balance      NUMBER(12, 2)

);

/

CREATE TABLE EMPLOYEES (

  EmpID         NUMBER PRIMARY KEY,

  Name          VARCHAR2(100),

  DepartmentID  NUMBER,

  Salary        NUMBER(10, 2)

);

/

INSERT INTO ACCOUNTS VALUES (101, 'SAVINGS', 10000);

INSERT INTO ACCOUNTS VALUES (102, 'SAVINGS', 15000);

INSERT INTO ACCOUNTS VALUES (103, 'CURRENT', 5000);

INSERT INTO ACCOUNTS VALUES (104, 'SAVINGS', 2500);

/

INSERT INTO EMPLOYEES VALUES (1, 'Alice', 10, 50000);

INSERT INTO EMPLOYEES VALUES (2, 'Bob', 20, 55000);

INSERT INTO EMPLOYEES VALUES (3, 'Charlie', 10, 60000);

INSERT INTO EMPLOYEES VALUES (4, 'Diana', 30, 45000);

/

COMMIT;

/

## Scenario 1:

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

  FOR acc IN (

    SELECT AccountID FROM ACCOUNTS WHERE AccountType = 'SAVINGS'

  ) LOOP

    UPDATE ACCOUNTS

    SET Balance = Balance \* 1.01

    WHERE AccountID = acc.AccountID;

  END LOOP;

  COMMIT;

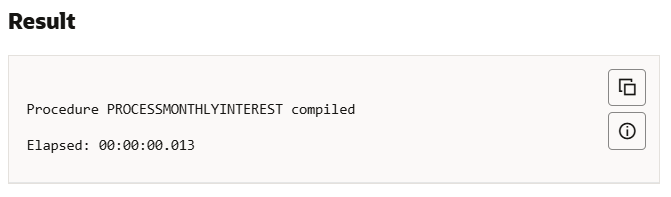
END;

/

-- Call Procedure

EXEC ProcessMonthlyInterest;

/



## Scenario 2:

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

  p\_dept\_id        IN NUMBER,

  p\_bonus\_percent  IN NUMBER

) AS

BEGIN

  UPDATE EMPLOYEES

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

  WHERE DepartmentID = p\_dept\_id;

  COMMIT;

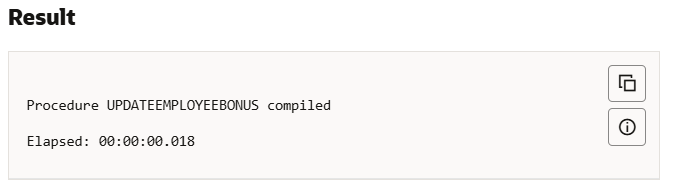
END;

/

-- Call Procedure

EXEC UpdateEmployeeBonus(10, 10);  -- 10% bonus to department 10

/



## Scenario 3:

CREATE OR REPLACE PROCEDURE TransferFunds (

  p\_from\_account IN NUMBER,

  p\_to\_account   IN NUMBER,

  p\_amount       IN NUMBER

) AS

  v\_balance NUMBER;

BEGIN

  SELECT Balance INTO v\_balance

  FROM ACCOUNTS

  WHERE AccountID = p\_from\_account;

  IF v\_balance >= p\_amount THEN

    UPDATE ACCOUNTS

    SET Balance = Balance - p\_amount

    WHERE AccountID = p\_from\_account;

    UPDATE ACCOUNTS

    SET Balance = Balance + p\_amount

    WHERE AccountID = p\_to\_account;

    COMMIT;

  ELSE

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

  END IF;

END;

/

-- Call Procedure

EXEC TransferFunds(102, 103, 2000);

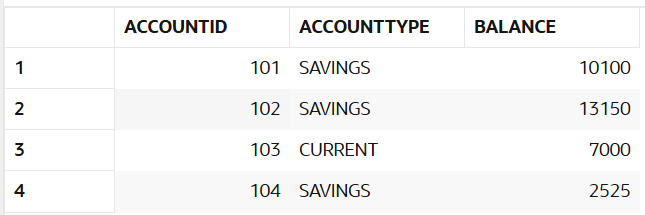
/



Checking final results:

SELECT \* FROM ACCOUNTS;

SELECT \* FROM EMPLOYEES;

****