**SUPERSET ID : 6375627**

**WEEK 2 PL SQL**

**Exercise 1: Control Structures**

**Code:**

-- Create CUSTOMERS table

CREATE TABLE CUSTOMERS (

  CustomerID NUMBER PRIMARY KEY,

  Name VARCHAR2(50),

  Age NUMBER,

  Balance NUMBER,

  IsVIP VARCHAR2(5)

);

-- Create LOANS table

CREATE TABLE LOANS (

  LoanID NUMBER PRIMARY KEY,

  CustomerID NUMBER,

  InterestRate NUMBER,

  DueDate DATE

);

-- Insert sample data

INSERT INTO CUSTOMERS VALUES (1, 'Arun', 65, 15000, 'FALSE');

INSERT INTO CUSTOMERS VALUES (2, 'Priya', 45, 9000, 'FALSE');

INSERT INTO CUSTOMERS VALUES (3, 'David', 70, 8000, 'FALSE');

INSERT INTO LOANS VALUES (101, 1, 8, SYSDATE + 20);

INSERT INTO LOANS VALUES (102, 2, 9, SYSDATE + 40);

INSERT INTO LOANS VALUES (103, 3, 7, SYSDATE + 10);

COMMIT;

SET SERVEROUTPUT ON;

BEGIN

  FOR c IN (SELECT CustomerID FROM CUSTOMERS WHERE Age > 60) LOOP

    UPDATE LOANS

    SET InterestRate = InterestRate - 1

    WHERE CustomerID = c.CustomerID;

  END LOOP;

  COMMIT;

END;

/

BEGIN

  FOR c IN (SELECT CustomerID FROM CUSTOMERS WHERE Balance > 10000) LOOP

    UPDATE CUSTOMERS

    SET IsVIP = 'TRUE'

    WHERE CustomerID = c.CustomerID;

  END LOOP;

  COMMIT;

END;

/

DECLARE

  v\_name CUSTOMERS.Name%TYPE;

BEGIN

  FOR l IN (SELECT LoanID, CustomerID, DueDate FROM LOANS

            WHERE DueDate BETWEEN SYSDATE AND SYSDATE + 30) LOOP

    SELECT Name INTO v\_name FROM CUSTOMERS WHERE CustomerID = l.CustomerID;

    DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || v\_name ||

                         ', your loan (Loan ID: ' || l.LoanID ||

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

  END LOOP;

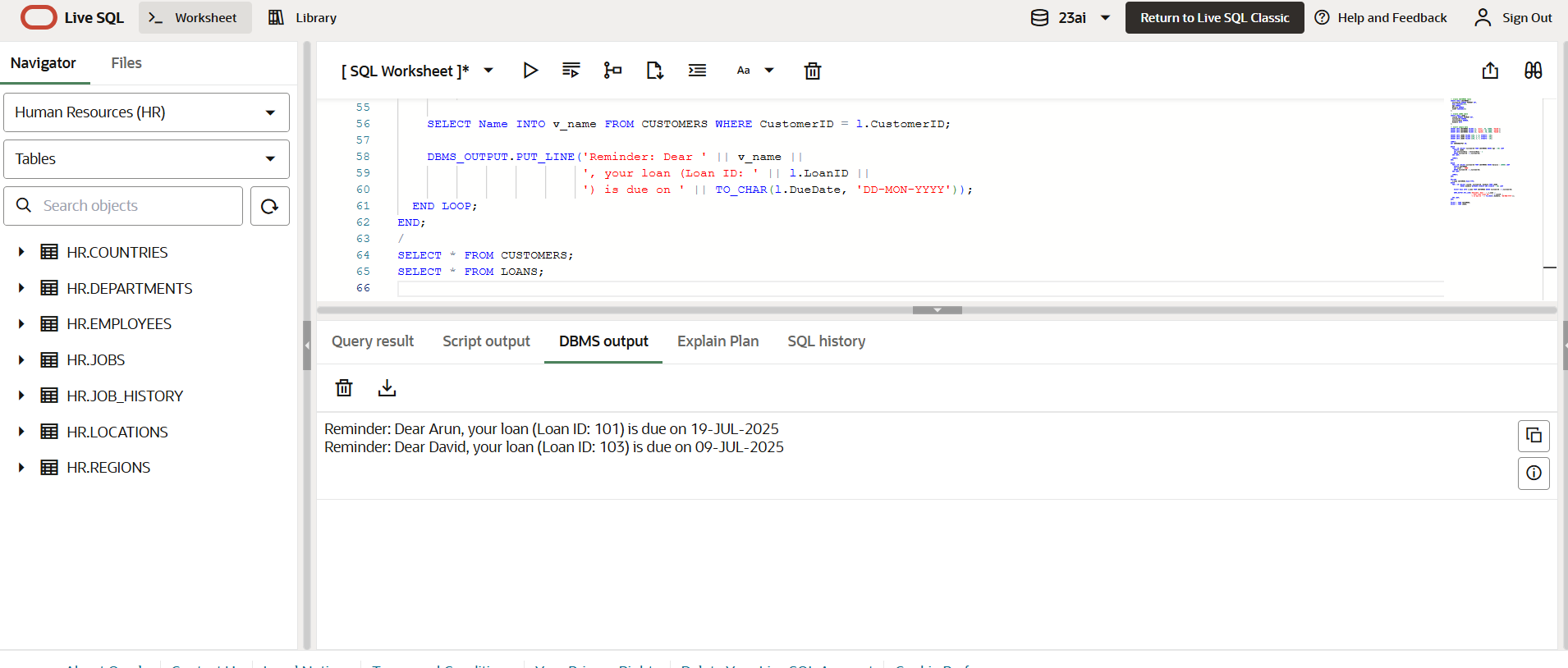
END;

/

SELECT \* FROM CUSTOMERS;

SELECT \* FROM LOANS;

**DBMS OUTPUT:**



**SCRIPT OUTPUT**

Table CUSTOMERS created.

Elapsed: 00:00:00.018

ORA-00907: missing right parenthesis

https://docs.oracle.com/error-help/db/ora-00907/Error at Line: 5 Column: 0

UPDATE LOANS

Elapsed: 00:00:00.009

Table LOANS created.

Elapsed: 00:00:00.017

1 row inserted.

Elapsed: 00:00:00.018

1 row inserted.

Elapsed: 00:00:00.001

1 row inserted.

Elapsed: 00:00:00.002

1 row inserted.

Elapsed: 00:00:00.011

1 row inserted.

Elapsed: 00:00:00.002

1 row inserted.

Elapsed: 00:00:00.001

Commit complete.

Elapsed: 00:00:00.002

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.088

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.012

Reminder: Dear Arun, your loan (Loan ID: 101) is due on 19-JUL-2025

Reminder: Dear David, your loan (Loan ID: 103) is due on 09-JUL-2025

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.021

CUSTOMERID NAME AGE BALANCE ISVIP

---------- ------- --- ------- -------

1 Arun 65 15000 TRUE

2 Priya 45 9000 FALSE

3 David 70 8000 FALSE

Elapsed: 00:00:00.005

3 rows selected.

LOANID CUSTOMERID INTERESTRATE DUEDATE

------ ---------- ------------ -------------------------

101 1 7 07/19/2025, 09:15:32 PM

102 2 9 08/08/2025, 09:15:32 PM

103 3 6 07/09/2025, 09:15:32 PM

Elapsed: 00:00:00.004

3 rows selected.

**Exercise 3: Stored Procedures**

**Code:**

-- Table for Savings Accounts

CREATE TABLE SavingsAccounts (

  AccountNumber NUMBER PRIMARY KEY,

  CustomerName VARCHAR2(50),

  Balance NUMBER

);

-- Table for Employees

CREATE TABLE Employees (

  EmployeeID NUMBER PRIMARY KEY,

  Name VARCHAR2(50),

  Department VARCHAR2(50),

  Salary NUMBER

);

-- Table for Bank Accounts (for fund transfer)

CREATE TABLE BankAccounts (

  AccountNumber NUMBER PRIMARY KEY,

  CustomerName VARCHAR2(50),

  Balance NUMBER

);

-- Sample Data for Savings Accounts

INSERT INTO SavingsAccounts VALUES (101, 'Arun', 5000);

INSERT INTO SavingsAccounts VALUES (102, 'Priya', 8000);

INSERT INTO SavingsAccounts VALUES (103, 'David', 12000);

-- Sample Data for Employees

INSERT INTO Employees VALUES (1, 'Kumar', 'HR', 30000);

INSERT INTO Employees VALUES (2, 'Divya', 'IT', 40000);

INSERT INTO Employees VALUES (3, 'Rahul', 'IT', 45000);

-- Sample Data for Bank Accounts

INSERT INTO BankAccounts VALUES (201, 'Arun', 10000);

INSERT INTO BankAccounts VALUES (202, 'Priya', 5000);

COMMIT;

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

  FOR acc IN (SELECT AccountNumber FROM SavingsAccounts) LOOP

    UPDATE SavingsAccounts

    SET Balance = Balance + (Balance \* 0.01)

    WHERE AccountNumber = acc.AccountNumber;

  END LOOP;

  COMMIT;

  DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied successfully to all accounts.');

END;

/

BEGIN

  ProcessMonthlyInterest;

END;

/

SELECT \* FROM SavingsAccounts;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

  p\_Department IN VARCHAR2,

  p\_BonusPercentage IN NUMBER

) AS

BEGIN

  UPDATE Employees

  SET Salary = Salary + (Salary \* p\_BonusPercentage / 100)

  WHERE Department = p\_Department;

  COMMIT;

  DBMS\_OUTPUT.PUT\_LINE('Bonus updated for ' || p\_Department || ' department.');

END;

/

BEGIN

  UpdateEmployeeBonus('IT', 10);

END;

/

SELECT \* FROM Employees;

CREATE OR REPLACE PROCEDURE TransferFunds (

  p\_FromAccount IN NUMBER,

  p\_ToAccount IN NUMBER,

  p\_Amount IN NUMBER

) AS

  v\_FromBalance NUMBER;

BEGIN

  -- Check balance of source account

  SELECT Balance INTO v\_FromBalance FROM BankAccounts WHERE AccountNumber = p\_FromAccount;

  IF v\_FromBalance < p\_Amount THEN

    DBMS\_OUTPUT.PUT\_LINE('Insufficient balance in account ' || p\_FromAccount);

  ELSE

    -- Deduct from source

    UPDATE BankAccounts

    SET Balance = Balance - p\_Amount

    WHERE AccountNumber = p\_FromAccount;

    -- Add to destination

    UPDATE BankAccounts

    SET Balance = Balance + p\_Amount

    WHERE AccountNumber = p\_ToAccount;

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || p\_Amount || ' from Account ' || p\_FromAccount || ' to Account ' || p\_ToAccount || ' successful.');

  END IF;

END;

/

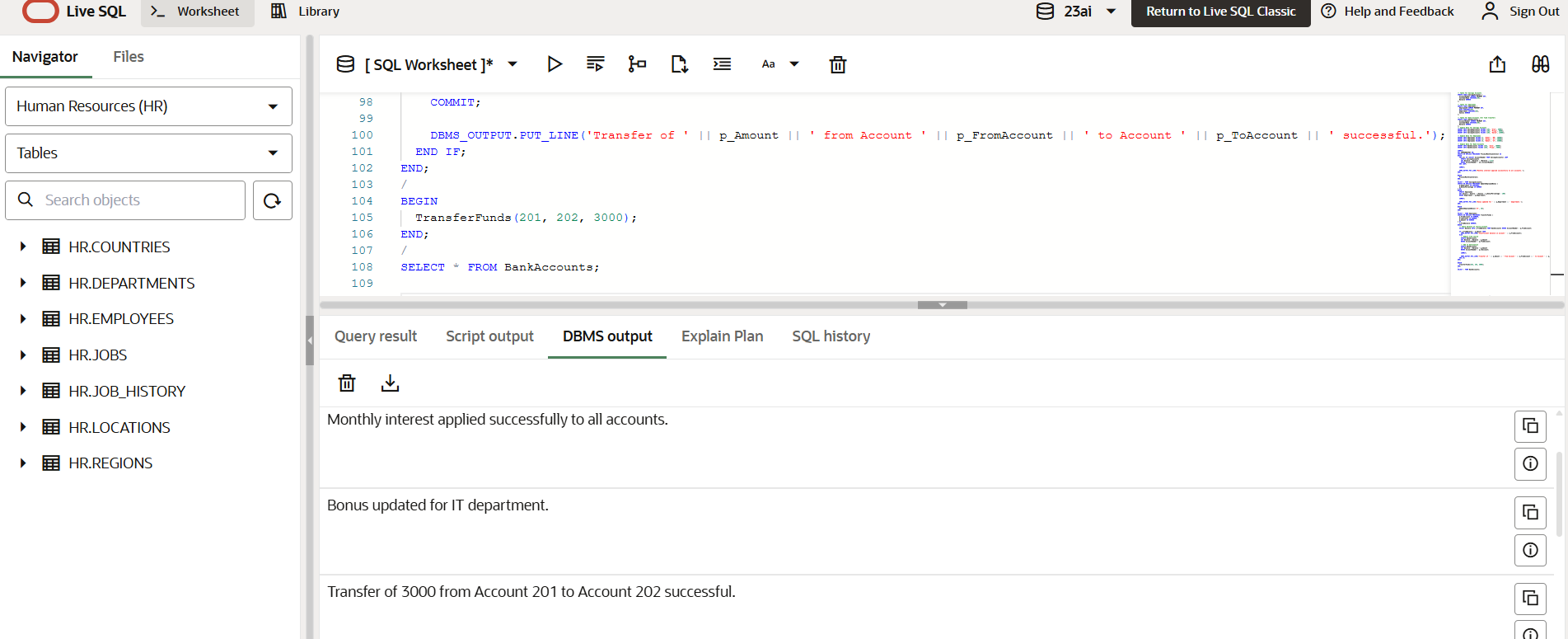
BEGIN

  TransferFunds(201, 202, 3000);

END;

/

SELECT \* FROM BankAccounts;



**Script output:**

SQL> BEGIN  
 ProcessMonthlyInterest;  
 END;  
Monthly interest applied successfully to all accounts.  
PL/SQL procedure successfully completed.  
Elapsed: 00:00:00.011  
SQL> CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (  
 p\_Department IN VARCHAR2,  
 p\_BonusPercentage IN NUMBER  
 ) AS...

Procedure UPDATEEMPLOYEEBONUS compiled  
  
Elapsed: 00:00:00.002

SQL> BEGIN  
 UpdateEmployeeBonus('IT', 10);  
 END;

Bonus updated for IT department.  
  
PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.011

SQL> SELECT \* FROM Employees

EMPLOYEEID NAME DEPARTMENT SALARY   
---------- ------- ---------- ------   
1 Kumar HR 30000   
2 Divya IT 53240   
3 Rahul IT 59895   
  
Elapsed: 00:00:00.005  
3 rows selected.

SQL> CREATE OR REPLACE PROCEDURE TransferFunds (  
 p\_FromAccount IN NUMBER,  
 p\_ToAccount IN NUMBER,  
 p\_Amount IN NUMBER...

Procedure TRANSFERFUNDS compiled  
Elapsed: 00:00:00.002

SQL> BEGIN  
 TransferFunds(201, 202, 3000);  
 END;

Transfer of 3000 from Account 201 to Account 202 successful.  
  
PL/SQL procedure successfully completed.  
  
Elapsed: 00:00:00.009

SQL> SELECT \* FROM BankAccounts

ACCOUNTNUMBER CUSTOMERNAME BALANCE   
------------- ------------ -------   
201 Arun 1000   
202 Priya 14000   
  
Elapsed: 00:00:00.005  
2 rows selected.