A close-up of a logo

AI-generated content may be incorrect.Digital Nurture **4.0 – Deep Skilling (Java FSE)**

Name : Aswath Kannan M  
Superset ID : 6420716 (Week 2)

**PL/SQL Exercises**

**Qn 1:Control Structures**

SET SERVEROUTPUT ON;

-- DROP tables if they already exist (optional if re-running)

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Loans';

EXECUTE IMMEDIATE 'DROP TABLE Customers';

EXCEPTION

WHEN OTHERS THEN NULL; -- ignore errors if tables don't exist

END;

/

-- Create Customers table

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Age NUMBER,

Balance NUMBER(10,2),

IsVIP VARCHAR2(5)

);

-- Create Loans table

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES Customers(CustomerID),

DueDate DATE,

InterestRate NUMBER(5,2)

);

-- Insert sample data

INSERT INTO Customers VALUES (1, 'John Doe', 65, 15000, 'FALSE');

INSERT INTO Customers VALUES (2, 'Jane Smith', 45, 8000, 'FALSE');

INSERT INTO Customers VALUES (3, 'Alice Johnson', 70, 20000, 'FALSE');

INSERT INTO Loans VALUES (101, 1, SYSDATE + 10, 7.5);

INSERT INTO Loans VALUES (102, 2, SYSDATE + 40, 6.9);

INSERT INTO Loans VALUES (103, 3, SYSDATE + 20, 8.0);

COMMIT;

BEGIN

FOR rec IN (

SELECT l.LoanID

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE c.Age > 60

) LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = rec.LoanID;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Discount applied to eligible senior customers.');

END;

/

BEGIN

FOR rec IN (SELECT CustomerID FROM Customers WHERE Balance > 10000) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = rec.CustomerID;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('VIP status updated based on balance.');

END;

/

BEGIN

FOR rec IN (

SELECT c.Name, l.DueDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE + 30

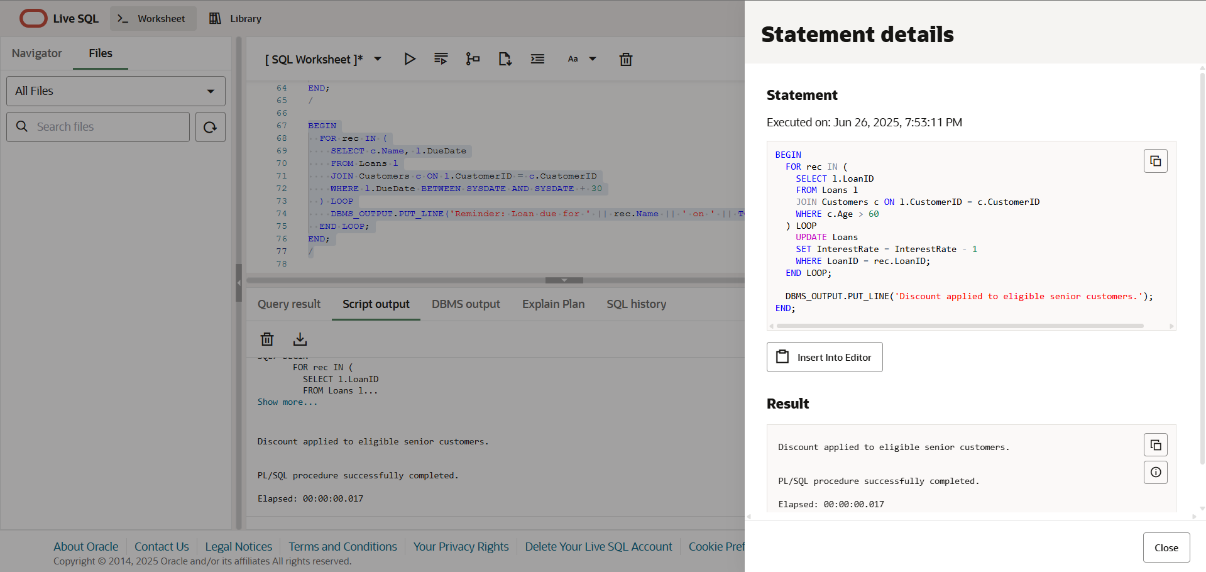
) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan due for ' || rec.Name || ' on ' || TO\_CHAR(rec.DueDate, 'DD-Mon-YYYY'));

END LOOP;

END; /

**Output**



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Qn 3: Stored Procedures**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Accounts';

EXECUTE IMMEDIATE 'DROP TABLE Employees';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

-- Create Accounts table

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(50),

AccountType VARCHAR2(20), -- e.g., 'Savings'

Balance NUMBER(10,2)

);

-- Create Employees table

CREATE TABLE Employees (

EmpID NUMBER PRIMARY KEY,

EmpName VARCHAR2(50),

Department VARCHAR2(30),

Salary NUMBER(10,2)

);

-- Insert sample data

INSERT INTO Accounts VALUES (1, 'John Doe', 'Savings', 10000);

INSERT INTO Accounts VALUES (2, 'Alice Smith', 'Savings', 15000);

INSERT INTO Accounts VALUES (3, 'Bob Martin', 'Current', 8000);

INSERT INTO Employees VALUES (101, 'Jane Manager', 'HR', 50000);

INSERT INTO Employees VALUES (102, 'Tom Dev', 'IT', 40000);

INSERT INTO Employees VALUES (103, 'Sara Dev', 'IT', 45000);

COMMIT;

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)

WHERE AccountID = acc.AccountID;

END LOOP;

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

END;

/

BEGIN

ProcessMonthlyInterest;

END;

/

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;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to department: ' || dept\_name);

END;

/

BEGIN

UpdateEmployeeBonus('IT', 10); -- 10% bonus for IT dept

END;

/

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account IN NUMBER,

to\_account IN NUMBER,

amount IN NUMBER

) IS

from\_balance NUMBER;

BEGIN

-- Get source balance

SELECT Balance INTO from\_balance FROM Accounts WHERE AccountID = from\_account FOR UPDATE;

IF from\_balance < amount THEN

DBMS\_OUTPUT.PUT\_LINE('Insufficient funds in source account.');

ELSE

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - amount

WHERE AccountID = from\_account;

-- Add to target

UPDATE Accounts

SET Balance = Balance + amount

WHERE AccountID = to\_account;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful.');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('One of the accounts does not exist.');

END;

/

BEGIN

TransferFunds(1, 2, 1000); -- Transfer 1000 from Account 1 to 2

END;

/

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.