**Harshitha M**

**Superset id : 6390499**

**Exercise 1: Control Structures**

First, create sample tables (just once):

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Age NUMBER,

Balance NUMBER,

IsVIP VARCHAR2(5)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES Customers(CustomerID),

InterestRate NUMBER,

DueDate DATE

);

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

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

INSERT INTO Customers VALUES (3, 'Arjun', 70, 11000, 'FALSE');

INSERT INTO Loans VALUES (101, 1, 8.5, SYSDATE + 10);

INSERT INTO Loans VALUES (102, 2, 9.0, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 10.0, SYSDATE + 25);

COMMIT;

Then run the block:

BEGIN

FOR r IN (

SELECT l.LoanID, l.InterestRate

FROM Loans l

JOIN Customers c ON c.CustomerID = l.CustomerID

WHERE c.Age > 60

)

LOOP

UPDATE Loans

SET InterestRate = r.InterestRate - 1

WHERE LoanID = r.LoanID;

END LOOP;

COMMIT;

END;

🔷 **Scenario 2: Set IsVIP = 'TRUE' for Balance > 10000BEGIN**

**UPDATE Customers**

**SET IsVIP = 'TRUE'**

**WHERE Balance > 10000;**

**COMMIT;**

**END;**

🔷 **Scenario 3: Show Reminders for Loans Due in Next 30 Days**

BEGIN

FOR r IN (

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

FROM Loans l

JOIN Customers c ON c.CustomerID = l.CustomerID

WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE + 30

)

LOOP

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

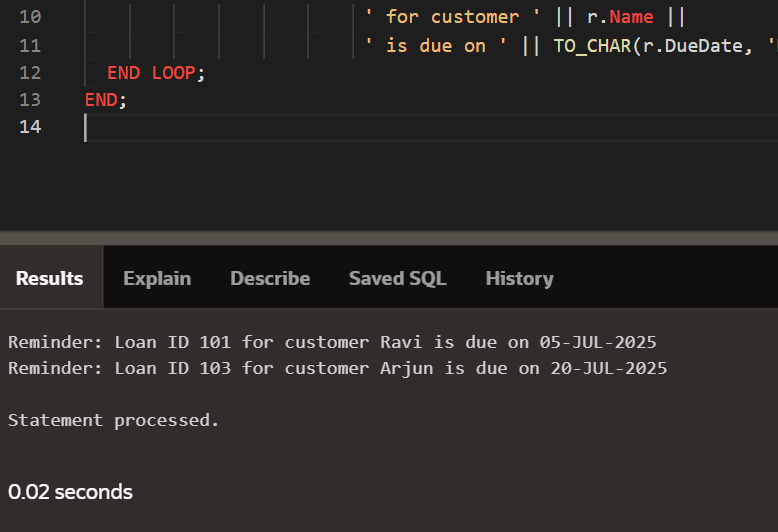
' for customer ' || r.Name ||

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

END LOOP;

END;

Output:



**Exercise 3: Stored Procedures**

CREATE TABLE SavingsAccounts (

AccountID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(50),

Balance NUMBER

);

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

DepartmentID NUMBER,

Salary NUMBER

);

-- Savings accounts

INSERT INTO SavingsAccounts VALUES (1, 'Ravi', 10000);

INSERT INTO SavingsAccounts VALUES (2, 'Meena', 15000);

INSERT INTO SavingsAccounts VALUES (3, 'Arjun', 8000);

-- Employees

INSERT INTO Employees VALUES (101, 'John', 10, 50000);

INSERT INTO Employees VALUES (102, 'Sara', 10, 52000);

INSERT INTO Employees VALUES (103, 'Raj', 20, 48000);

COMMIT;

Scenario 1: **Monthly Interest for Savings Accounts**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE SavingsAccounts

SET Balance = Balance + (Balance \* 0.01);

COMMIT;

END;

BEGIN

ProcessMonthlyInterest;

END;

SELECT \* FROM SavingsAccounts;

Scenario 2: **Add Bonus to Employees in a Department**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

dept\_id IN NUMBER,

bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* bonus\_percent / 100)

WHERE DepartmentID = dept\_id;

COMMIT;

END;

BEGIN

UpdateEmployeeBonus(10, 10);

END;

SELECT \* FROM Employees;

Scenario 3: **Transfer Funds Between Accounts**

CREATE OR REPLACE PROCEDURE TransferFunds(

source\_id IN NUMBER,

dest\_id IN NUMBER,

amount IN NUMBER

) IS

source\_balance NUMBER;

BEGIN

-- Get source account balance

SELECT Balance INTO source\_balance

FROM SavingsAccounts

WHERE AccountID = source\_id;

-- Check balance

IF source\_balance >= amount THEN

-- Deduct from source

UPDATE SavingsAccounts

SET Balance = Balance - amount

WHERE AccountID = source\_id;

-- Add to destination

UPDATE SavingsAccounts

SET Balance = Balance + amount

WHERE AccountID = dest\_id;

COMMIT;

ELSE

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

END IF;

END;

BEGIN

TransferFunds(1, 2, 2000);

END;

SELECT \* FROM SavingsAccounts;

