**PL/SQL programming**

**Exercise 1: Control Structures**

--CREATE Customers Table

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Age NUMBER,

Balance NUMBER,

IsVIP VARCHAR2(5) DEFAULT 'FALSE'

);

--CREATE Loans Table

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

InterestRate NUMBER(5,2),

DueDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

--Insert customer details into Customers Table

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

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

INSERT INTO Customers VALUES (3, 'Michael Lee', 70, 12000, 'FALSE');

INSERT INTO Customers VALUES (4, 'Suresh Kumar', 62, 9500, 'FALSE');

INSERT INTO Customers VALUES (5, 'Meera Das', 68, 11000, 'FALSE');

INSERT INTO Customers VALUES (6, 'Rajiv Nair', 75, 30000, 'FALSE');

INSERT INTO Customers VALUES (7, 'Anita Sharma', 50, 10500, 'FALSE');

INSERT INTO Customers VALUES (8, 'Vikram Rao', 35, 4000, 'FALSE');

INSERT INTO Customers VALUES (9, 'Kiran Patil', 28, 16000, 'FALSE');

INSERT INTO Customers VALUES (10, 'Priya Sinha', 59, 10050, 'FALSE');

--Insert loan details into Loans Table

INSERT INTO Loans VALUES (101, 1, 7.5, TO\_DATE('2025-07-10', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (102, 2, 6.0, TO\_DATE('2025-08-15', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (103, 3, 8.0, TO\_DATE('2025-06-30', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (104, 4, 7.0, TO\_DATE('2025-07-01', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (105, 5, 6.8, TO\_DATE('2025-07-25', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (106, 6, 7.9, TO\_DATE('2025-07-15', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (107, 7, 6.5, TO\_DATE('2025-08-20', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (108, 8, 6.3, TO\_DATE('2025-07-10', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (109, 9, 6.0, TO\_DATE('2025-06-29', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (110, 10, 6.9, TO\_DATE('2025-07-30', 'YYYY-MM-DD'));

COMMIT;

--View Customers Table

SELECT \* FROM Customers;

--View Loans Table

SELECT \* FROM Loans;

--Scenario 1:

DECLARE

CURSOR cur\_customers IS

SELECT CustomerID, InterestRate

FROM Loans

WHERE CustomerID IN (

SELECT CustomerID

FROM Customers

WHERE Age > 60

);

BEGIN

FOR rec IN cur\_customers LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = rec.CustomerID;

END LOOP;

COMMIT;

END;

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

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

WHERE c.Age > 60;

--Scenario 2:

BEGIN

FOR rec IN (

SELECT CustomerID

FROM Customers

WHERE Balance > 10000

) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = rec.CustomerID;

END LOOP;

COMMIT;

END;

SELECT CustomerID, Name, Balance, IsVIP

FROM Customers

WHERE IsVIP = 'TRUE';

--Scenario 3:

DECLARE

CURSOR cur\_due\_loans IS

SELECT l.CustomerID, l.DueDate, c.Name

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.DueDate <= SYSDATE + 30;

BEGIN

FOR rec IN cur\_due\_loans LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan for customer ' || rec.Name ||

' (ID: ' || rec.CustomerID || ') is due on ' || TO\_CHAR(rec.DueDate, 'DD-MON-YYYY'));

END LOOP;

END;

SELECT l.CustomerID, c.Name, l.DueDate

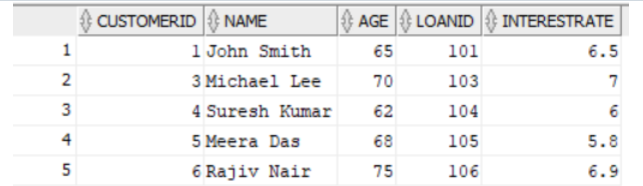
FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

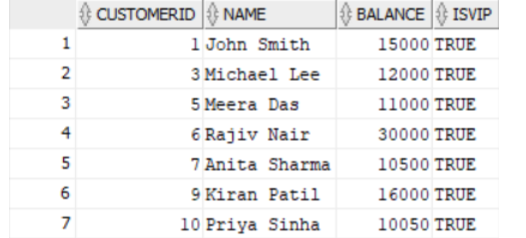
WHERE l.DueDate <= SYSDATE + 30;

**OUTPUT:**

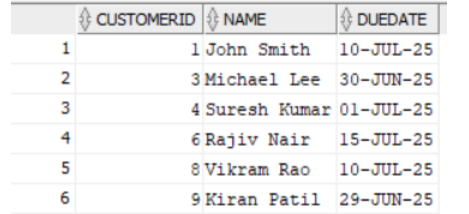
**Scenario 1:**

****

**Scenario 2:**

****

**Scenario 3:**

****

**Exercise 3: Stored Procedures**

--Create table accounts

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY,

account\_holder VARCHAR2(100),

account\_type VARCHAR2(20),

balance NUMBER(10,2)

);

INSERT INTO accounts (account\_id,account\_holder, account\_type, balance) VALUES (123,'Alice', 'savings', 1000.00);

INSERT INTO accounts (account\_id,account\_holder, account\_type, balance) VALUES (678,'Bob', 'current', 2000.00);

INSERT INTO accounts (account\_id,account\_holder, account\_type, balance) VALUES (890,'Charlie', 'savings', 1500.00);

INSERT INTO accounts (account\_id,account\_holder, account\_type, balance) VALUES (346,'Diana', 'savings', 2500.00);

SELECT \* FROM accounts;

--Create table employees

CREATE TABLE employees (

employee\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

department\_id NUMBER,

salary NUMBER(10,2)

);

INSERT INTO employees (employee\_id, name, department\_id, salary) VALUES (101, 'Alice', 1, 50000.00);

INSERT INTO employees (employee\_id, name, department\_id, salary) VALUES (102, 'Bob', 2, 60000.00);

INSERT INTO employees (employee\_id, name, department\_id, salary) VALUES (103, 'Charlie', 1, 55000.00);

INSERT INTO employees (employee\_id, name, department\_id, salary) VALUES (104, 'Diana', 3, 70000.00);

SELECT \* FROM employees;

--Scenario 1:

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE accounts

SET balance = balance \* 1.01

WHERE account\_type = 'savings';

END;

/

BEGIN

ProcessMonthlyInterest;

END;

/

SELECT \* FROM accounts;

SELECT \* FROM accounts WHERE account\_type = 'savings';

--Scenario 2:

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 department\_id = dept\_id;

END;

/

BEGIN

UpdateEmployeeBonus(1, 10);

END;

/

SELECT \* FROM employees;

SELECT \* FROM employees WHERE department\_id = 1;

--Scenario 3:

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_source\_account\_id IN NUMBER,

p\_target\_account\_id IN NUMBER,

p\_amount IN NUMBER

) AS

v\_source\_balance NUMBER;

BEGIN

-- Step 1: Get source balance

SELECT balance

INTO v\_source\_balance

FROM accounts

WHERE account\_id = p\_source\_account\_id;

DBMS\_OUTPUT.PUT\_LINE('Source account balance: ' || v\_source\_balance);

-- Step 2: Check sufficient balance

IF v\_source\_balance < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance. Cannot transfer.');

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

END IF;

-- Step 3: Deduct from source account

UPDATE accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_source\_account\_id;

DBMS\_OUTPUT.PUT\_LINE('Deducted ' || p\_amount || ' from account ' || p\_source\_account\_id);

-- Step 4: Add to target account

UPDATE accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_target\_account\_id;

DBMS\_OUTPUT.PUT\_LINE('Added ' || p\_amount || ' to account ' || p\_target\_account\_id);

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer completed successfully.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('One or both account IDs do not exist.');

RAISE\_APPLICATION\_ERROR(-20002, 'Source or target account does not exist.');

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Unexpected error: ' || SQLERRM);

RAISE;

END;

/

SET SERVEROUTPUT ON;

BEGIN

TransferFunds(123, 678, 200);

END;

/

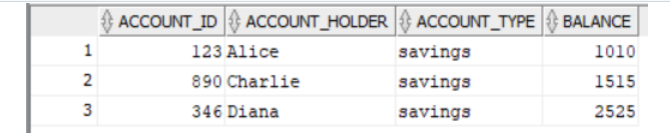
SELECT account\_id, account\_holder, balance

FROM accounts

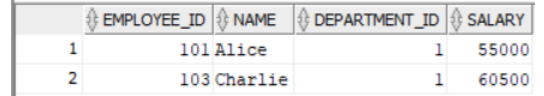
WHERE account\_id IN (123, 678);

**OUTPUT:**

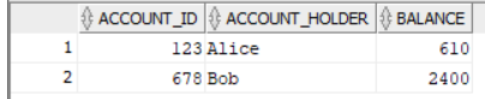
**Scenario 1:**

****

**Scenario 2:**



**Scenario 3:**

****