**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, 'Ravi Menon', 66, 14000, 'FALSE');

INSERT INTO Customers VALUES (2, 'Nisha Kapoor', 47, 9200, 'FALSE');

INSERT INTO Customers VALUES (3, 'Daniel Roy', 72, 13500, 'FALSE');

INSERT INTO Customers VALUES (4, 'Kavitha Iyer', 63, 8900, 'FALSE');

INSERT INTO Customers VALUES (5, 'Subhash Chatterjee', 69, 11800, 'FALSE');

INSERT INTO Customers VALUES (6, 'Farida Khan', 74, 28000, 'FALSE');

INSERT INTO Customers VALUES (7, 'Arjun Reddy', 53, 11200, 'FALSE');

INSERT INTO Customers VALUES (8, 'Tina Fernandez', 38, 4800, 'FALSE');

INSERT INTO Customers VALUES (9, 'Devendra Singh', 31, 17000, 'FALSE');

INSERT INTO Customers VALUES (10, 'Ritika Malhotra', 58, 10100, 'FALSE');

--Insert loan details into Loans Table

INSERT INTO Loans VALUES (201, 1, 7.4, TO\_DATE('2025-07-12', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (202, 2, 6.2, TO\_DATE('2025-08-17', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (203, 3, 7.8, TO\_DATE('2025-06-29', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (204, 4, 6.9, TO\_DATE('2025-07-03', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (205, 5, 6.7, TO\_DATE('2025-07-23', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (206, 6, 8.0, TO\_DATE('2025-07-13', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (207, 7, 6.6, TO\_DATE('2025-08-22', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (208, 8, 6.4, TO\_DATE('2025-07-12', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (209, 9, 6.1, TO\_DATE('2025-06-28', 'YYYY-MM-DD'));

INSERT INTO Loans VALUES (210, 10, 6.8, TO\_DATE('2025-07-31', 'YYYY-MM-DD'));

COMMIT;

--View Customers Table

SELECT \* FROM Customers;

--View Loans Table

SELECT \* FROM Loans;

--Scenario 1: Reduce interest rate by 1% for customers older than 60

DECLARE

CURSOR senior\_loans IS

SELECT CustomerID, InterestRate

FROM Loans

WHERE CustomerID IN (

SELECT CustomerID FROM Customers WHERE Age > 60

);

BEGIN

FOR rec IN senior\_loans 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: Set IsVIP to TRUE for customers with Balance > 10000

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: Display reminder for loans due within next 30 days

DECLARE

CURSOR due\_soon 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 due\_soon 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;

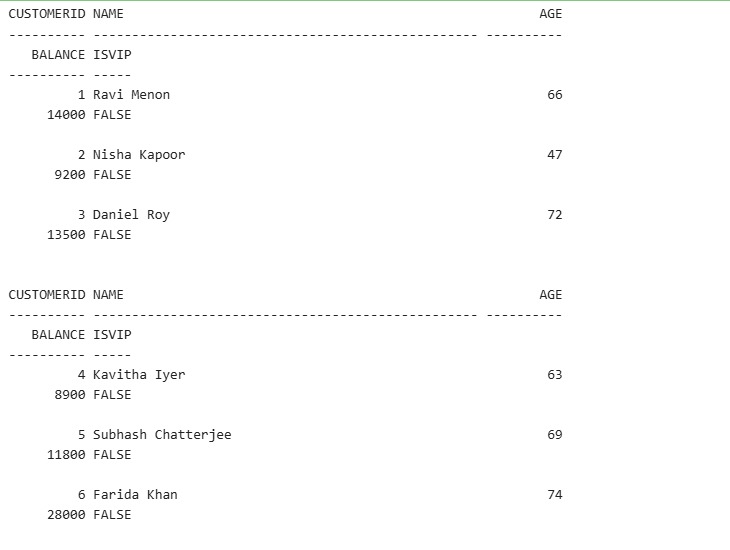
/

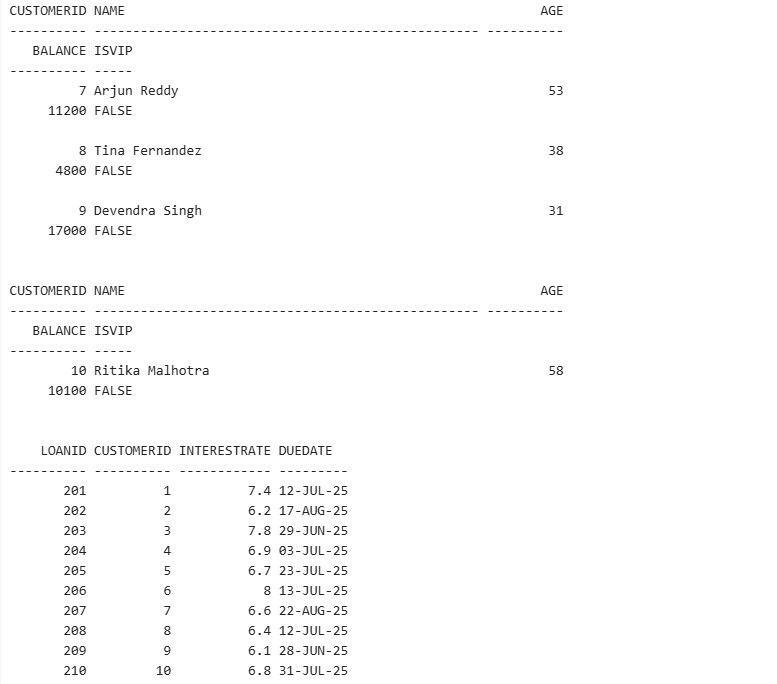
SELECT l.CustomerID, c.Name, l.DueDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.DueDate <= SYSDATE + 30;

Output:  




**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 data into accounts

INSERT INTO accounts (account\_id, account\_holder, account\_type, balance) VALUES (201, 'Neha Verma', 'savings', 1050.00);

INSERT INTO accounts (account\_id, account\_holder, account\_type, balance) VALUES (202, 'Rohit Mehta', 'current', 2100.00);

INSERT INTO accounts (account\_id, account\_holder, account\_type, balance) VALUES (203, 'Sneha Iyer', 'savings', 1450.00);

INSERT INTO accounts (account\_id, account\_holder, account\_type, balance) VALUES (204, 'Manoj Desai', 'savings', 2600.00);

-- View accounts

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 data into employees

INSERT INTO employees (employee\_id, name, department\_id, salary) VALUES (301, 'Neha Verma', 10, 52000.00);

INSERT INTO employees (employee\_id, name, department\_id, salary) VALUES (302, 'Rohit Mehta', 20, 63000.00);

INSERT INTO employees (employee\_id, name, department\_id, salary) VALUES (303, 'Sneha Iyer', 10, 58000.00);

INSERT INTO employees (employee\_id, name, department\_id, salary) VALUES (304, 'Manoj Desai', 30, 71000.00);

-- View employees

SELECT \* FROM employees;

-- Scenario 1: Add monthly interest to savings accounts

CREATE OR REPLACE PROCEDURE ApplyMonthlyInterest IS

BEGIN

UPDATE accounts

SET balance = balance \* 1.01

WHERE account\_type = 'savings';

END;

/

BEGIN

ApplyMonthlyInterest;

END;

/

SELECT \* FROM accounts;

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

-- Scenario 2: Update bonus for specific department employees

CREATE OR REPLACE PROCEDURE ApplyBonusToDept (

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

ApplyBonusToDept(10, 10);

END;

/

SELECT \* FROM employees;

SELECT \* FROM employees WHERE department\_id = 10;

-- Scenario 3: Transfer funds between accounts

CREATE OR REPLACE PROCEDURE TransferAmount (

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) AS

v\_balance NUMBER;

BEGIN

-- Get source balance

SELECT balance INTO v\_balance

FROM accounts

WHERE account\_id = p\_from\_account;

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

-- Check balance

IF v\_balance < p\_amount THEN

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

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

END IF;

-- Deduct from source

UPDATE accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_from\_account;

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

-- Add to target

UPDATE accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_to\_account;

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

COMMIT;

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

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Account not found.');

RAISE\_APPLICATION\_ERROR(-20002, 'Invalid source or target account.');

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error occurred: ' || SQLERRM);

RAISE;

END;

/

SET SERVEROUTPUT ON;

BEGIN

TransferAmount(201, 202, 300);

END;

/

-- Check updated balances

SELECT account\_id, account\_holder, balance

FROM accounts

WHERE account\_id IN (201, 202);

**Output:**

