**Exercise 1: Control Structures**

**Section 1: Table Setup (DDL + Sample Data)**

-- Create Customers table

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Age NUMBER,

Balance NUMBER,

IsVIP VARCHAR2(5)

);

/

-- 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 sample customers

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

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

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

INSERT INTO Customers VALUES (4, 'David', 30, 9500, 'FALSE');

/

-- Insert sample loans

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

INSERT INTO Loans VALUES (102, 2, 6.0, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 8.0, SYSDATE + 5);

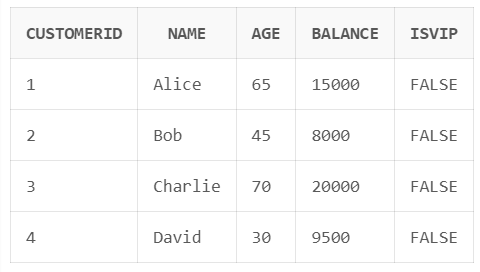
INSERT INTO Loans VALUES (104, 4, 9.0, SYSDATE + 25);

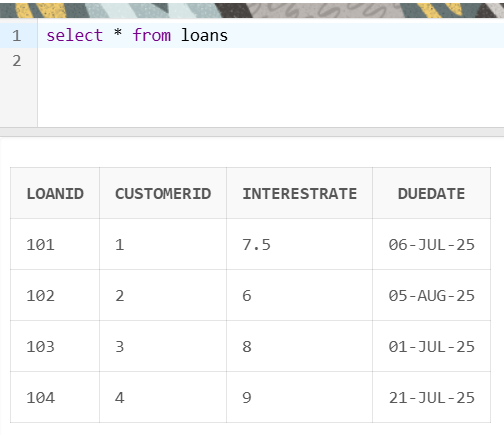
/

COMMIT;

**Expected Output (From Oracle Live SQL):**

Select \* from customers;





**Scenario 1: Apply Interest Discount for Age > 60**

BEGIN

FOR cust IN (SELECT CustomerID FROM Customers WHERE Age > 60) LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust.CustomerID;

END LOOP;

COMMIT;

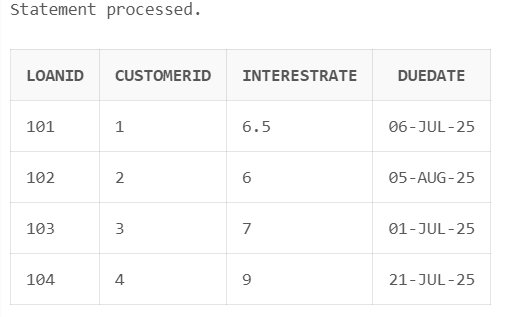
END;

/

-- View updated loan interest rates

SELECT \* FROM Loans;

**Expected Output (From Oracle Live SQL):**



**Scenario 2: Promote Customers to VIP**

BEGIN

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

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID;

END LOOP;

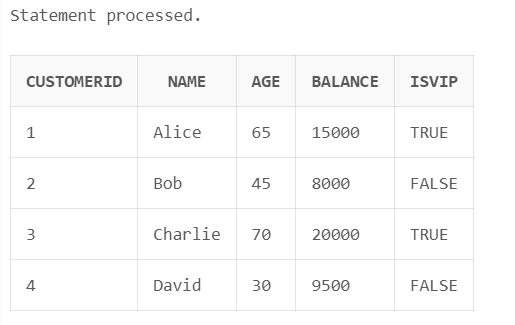
COMMIT;

END;

/

-- View updated VIP status

SELECT \* FROM Customers;

**Expected Output (From Oracle Live SQL):**

**Scenario 3: Print Loan Reminders (Next 30 Days)**

BEGIN

FOR loan\_rec IN (

SELECT L.LoanID, L.CustomerID, C.Name, L.DueDate

FROM Loans L

JOIN Customers C ON L.CustomerID = C.CustomerID

WHERE L.DueDate <= SYSDATE + 30

) LOOP

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

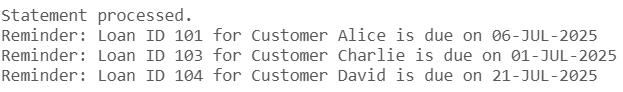
' for Customer ' || loan\_rec.Name ||

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

END LOOP;

END;

**Expected Output (From Oracle Live SQL):**



**Exercise 3: Stored Procedures**

**Tables and Sample Data**

-- Create Accounts table

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(100),

AccountType VARCHAR2(20),

Balance NUMBER

);

/

-- Create Employees table

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Department VARCHAR2(50),

Salary NUMBER

);

/

-- Insert sample accounts

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

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

INSERT INTO Accounts VALUES (3, 'Charlie', 'Current', 20000);

INSERT INTO Accounts VALUES (4, 'David', 'Savings', 8000);

/

-- Insert sample employees

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

INSERT INTO Employees VALUES (102, 'Jane', 'Finance', 60000);

INSERT INTO Employees VALUES (103, 'Mark', 'HR', 55000);

INSERT INTO Employees VALUES (104, 'Lucy', 'IT', 70000);

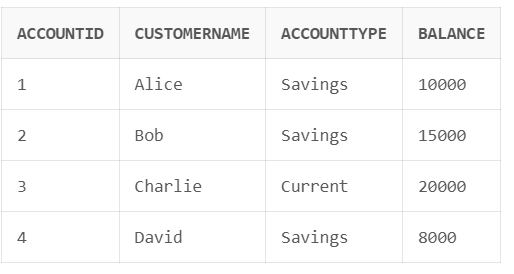
/

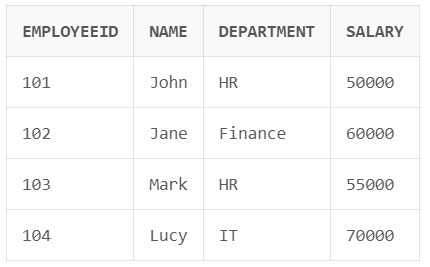
COMMIT;

SELECT \* FROM Accounts;

SELECT \* FROM Employees;

**Expected Output (From Oracle Live SQL):**





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

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'Savings';

END ProcessMonthlyInterest;

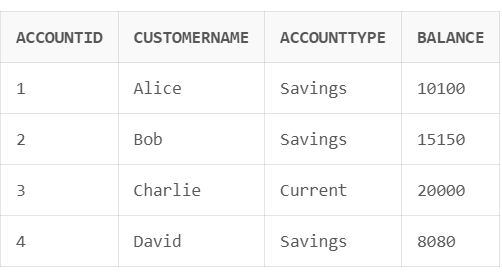
/

-- Call the procedure

EXEC ProcessMonthlyInterest;

Select \* from Accounts;

**Expected Output (From Oracle Live SQL):**



**Scenario 2: Bonus for Employees in a Department**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

dept\_name IN VARCHAR2,

bonus\_pct IN NUMBER

) AS

BEGIN

UPDATE Employees

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

WHERE Department = dept\_name;

END UpdateEmployeeBonus;

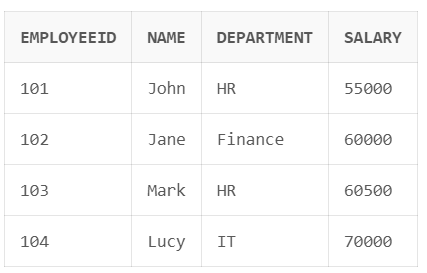
/

-- Call example: 10% bonus for HR

EXEC UpdateEmployeeBonus('HR', 10);

Select \* from Employees;

**Expected Output (From Oracle Live SQL):**



**Scenario 3: Transfer Funds Between Accounts**

CREATE OR REPLACE PROCEDURE TransferFunds(

source\_id IN NUMBER,

target\_id IN NUMBER,

amount IN NUMBER

) AS

insufficient\_balance EXCEPTION;

BEGIN

DECLARE

src\_balance NUMBER;

BEGIN

SELECT Balance INTO src\_balance FROM Accounts WHERE AccountID = source\_id;

IF src\_balance < amount THEN

RAISE insufficient\_balance;

ELSE

UPDATE Accounts SET Balance = Balance - amount WHERE AccountID = source\_id;

UPDATE Accounts SET Balance = Balance + amount WHERE AccountID = target\_id;

END IF;

END;

EXCEPTION

WHEN insufficient\_balance THEN

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

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: ' || SQLERRM);

END TransferFunds;

/

-- Enable output and call the procedure

EXEC TransferFunds(1, 2, 2000);

SELECT \* FROM Accounts;

**Expected Output (From Oracle Live SQL):**

