**Exercise 1: Control Structures**

**Scenario 1:**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Age NUMBER

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

InterestRate NUMBER,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Customers VALUES (1, 'Ali', 62);

INSERT INTO Customers VALUES (2, 'Ajay', 45);

INSERT INTO Customers VALUES (3, 'Raju', 70);

INSERT INTO Loans VALUES (111, 1, 8.5);

INSERT INTO Loans VALUES (112, 2, 9.0);

INSERT INTO Loans VALUES (113, 3, 5.5);

COMMIT;

BEGIN

FOR rec IN (

SELECT l.LoanID, l.InterestRate

FROM LOANS l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE c.Age > 60

) LOOP

UPDATE LOANS

SET InterestRate = rec.InterestRate - 1

WHERE LoanID = rec.LoanID;

END LOOP;

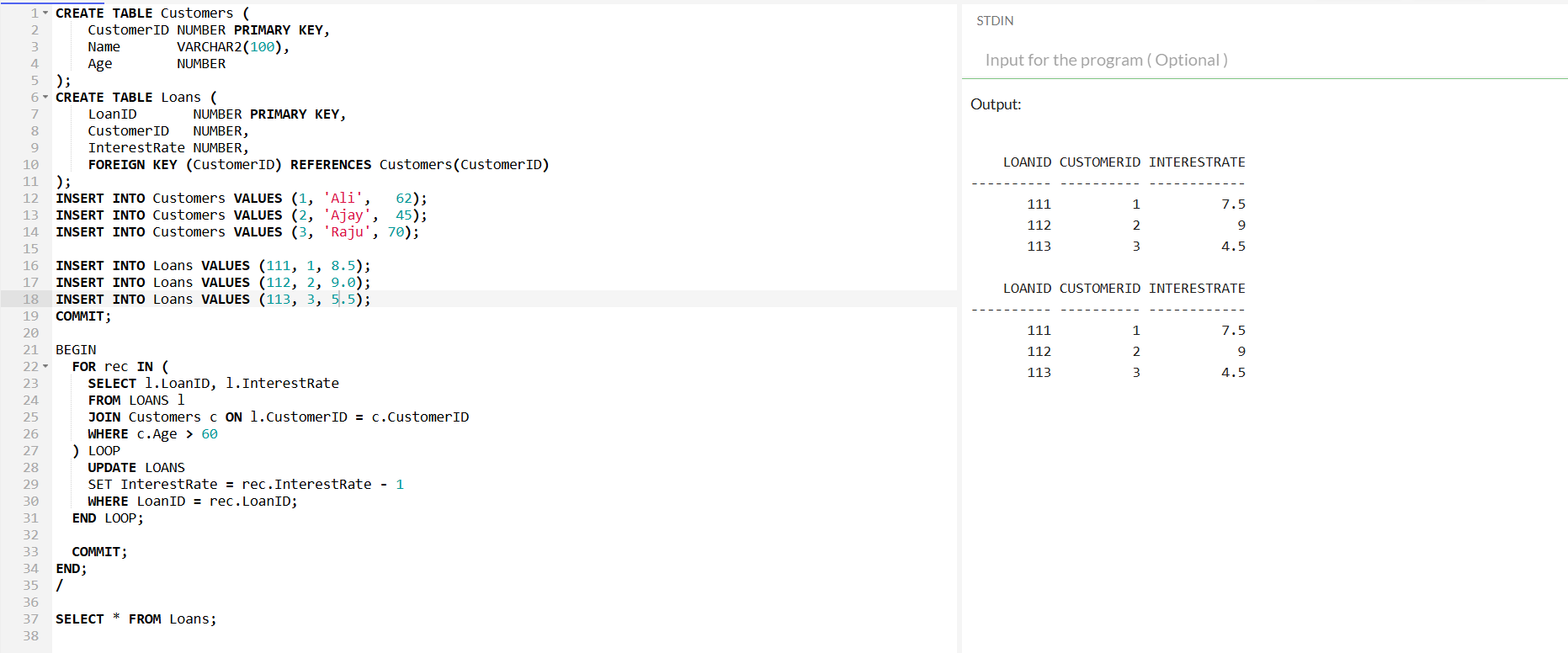
COMMIT;

END;

/

SELECT \* FROM Loans;

**OUTPUT:**



========================================================================

**Scenario 2:**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR(100),

Age NUMBER,

Balance NUMBER,

is\_vip VARCHAR(5) default 'FALSE'

);

INSERT INTO Customers (CustomerID, Name, Age, Balance) VALUES (1, 'Ajay', 45, 20000);

INSERT INTO Customers (CustomerID, Name, Age, Balance) VALUES (2, 'Varun', 57, 10000);

INSERT INTO Customers (CustomerID, Name, Age, Balance) VALUES (3, 'Ananya', 60, 9000);

COMMIT;

BEGIN

FOR cust\_rec IN (

SELECT CustomerID, Name, Balance

FROM Customers

WHERE Balance > 10000

) LOOP

UPDATE Customers

SET is\_vip = 'TRUE'

WHERE CustomerID = cust\_rec.CustomerID;

DBMS\_OUTPUT.PUT\_LINE(cust\_rec.Name || ' is now a VIP customer!');

END LOOP;

COMMIT;

END;

/

SELECT \* FROM CUSTOMERS;

**OUTPUT:**

| **CustomerID** | **Name** | **Age** | **Balance** | **is\_vip** |
| --- | --- | --- | --- | --- |
| 1 | Ajay | 45 | 20000 | FALSE |
| 2 | Varun | 57 | 10000 | FALSE |
| 3 | Ananya | 60 | 9000 | FALSE |

Ajay is now a VIP customer!

| **CustomerID** | **Name** | **Age** | **Balance** | **is\_vip** |
| --- | --- | --- | --- | --- |
| 1 | Ajay | 45 | 20000 | TRUE |
| 2 | Varun | 57 | 10000 | FALSE |
| 3 | Ananya | 60 | 9000 | FALSE |

========================================================================

**Scenario 3:**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

DueDate DATE,

LoanAmount NUMBER,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Customers VALUES (1, 'Ajay');

INSERT INTO Customers VALUES (2, 'Arun');

INSERT INTO Customers VALUES (3, 'Navya');

INSERT INTO Loans VALUES (10, 1, SYSDATE + 15, 50000);

INSERT INTO Loans VALUES (11, 2, SYSDATE + 30, 8000);

INSERT INTO Loans VALUES (12, 3, SYSDATE + 20, 10000);

COMMIT;

DECLARE

v\_due\_date DATE := SYSDATE + 30;

BEGIN

FOR loan\_rec IN (SELECT LoanID, CustomerID, DueDate

FROM Loans

WHERE DueDate <= v\_due\_date AND DueDate >= SYSDATE

) LOOP

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

' for Customer ID ' || loan\_rec.CustomerID ||

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

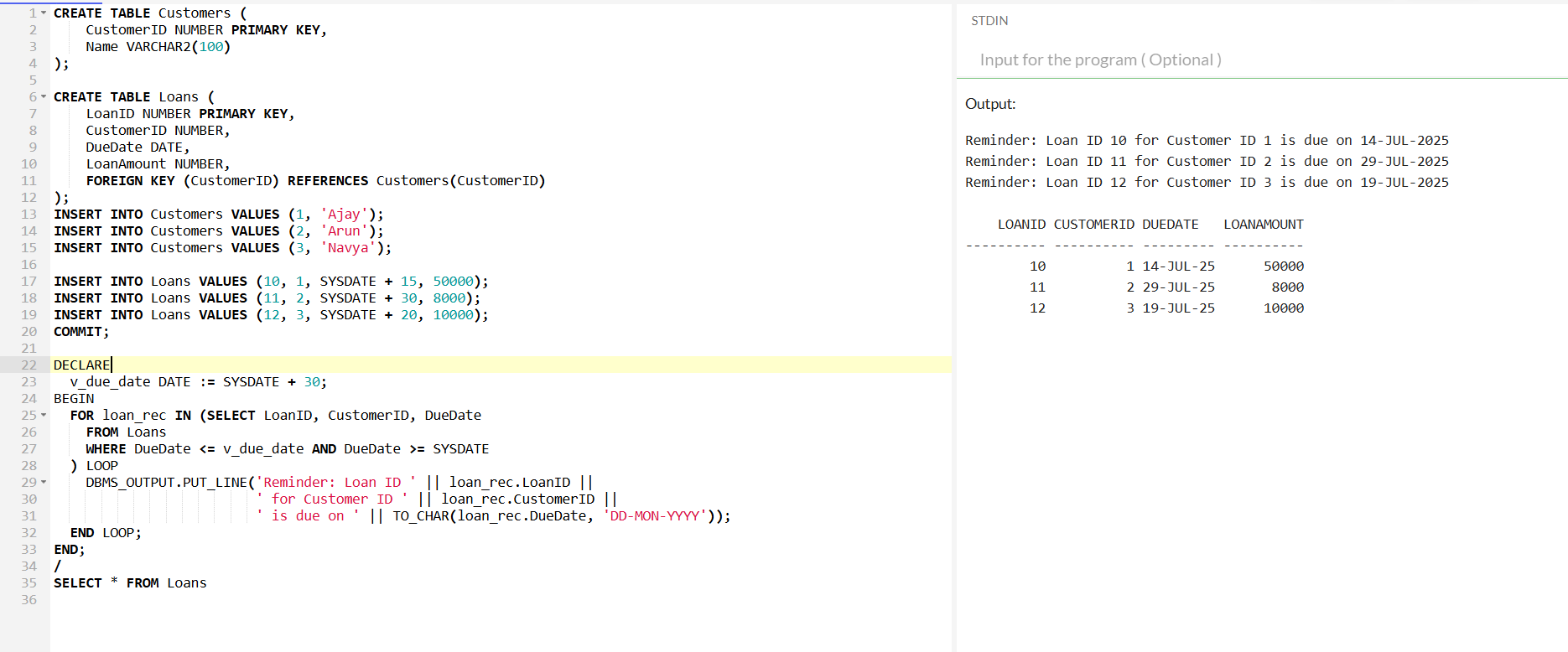
END LOOP;

END;

/

SELECT \* FROM Loans

**OUTPUT:**



==================================================

**Exercise 3: Stored Procedures**

**Scenario 1:**

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerName VARCHAR(50),

AccountType VARCHAR2(20),

Balance NUMBER

);

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

INSERT INTO Accounts VALUES (2, 'Bob', 'Checking', 5000);

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

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR acc\_rec IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'SAVINGS'

) LOOP

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountID = acc\_rec.AccountID;

END LOOP;

COMMIT;

END;

/

SELECT \* FROM Accounts

**OUTPUT:**

| **AccountID** | **CustomerName** | **AccountType** | **Balance** |
| --- | --- | --- | --- |
| 1 | Alice | Savings | 10000 |
| 2 | Bob | Checking | 5000 |
| 3 | Charlie | Savings | 20000 |

| **AccountID** | **CustomerName** | **AccountType** | **Balance** |
| --- | --- | --- | --- |
| 1 | Alice | Savings | 10100 |
| 2 | Bob | Checking | 5000 |
| 3 | Charlie | Savings | 20200 |

======================================================================

**Scenario 2:**

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Department VARCHAR2(50),

Salary NUMBER

);

INSERT INTO Employees VALUES (1, 'Anand', 'HR', 50000);

INSERT INTO Employees VALUES (2, 'John', 'IT', 60000);

INSERT INTO Employees VALUES (3, 'Neha', 'IT', 70000);

INSERT INTO Employees VALUES (4, 'Shraddha', 'Finance', 55000);

COMMIT;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_department IN VARCHAR2,

p\_bonus\_pct IN NUMBER

) AS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* p\_bonus\_pct / 100)

WHERE Department = p\_department;

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || p\_bonus\_pct || '% applied to department: ' || p\_department);

COMMIT;

END;

/

BEGIN

UpdateEmployeeBonus('IT',10);

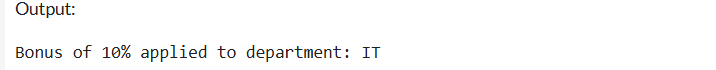
END;

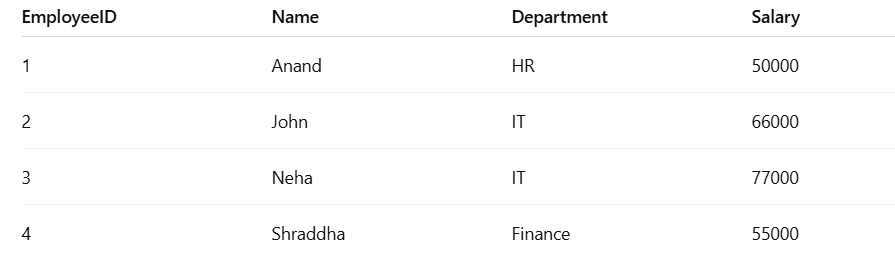
/

SELECT \* FROM Employees;

**OUTPUT:**







======================================================================

**Scenario 3:**

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerName VARCHAR(50),

Balance NUMBER

);

INSERT INTO Accounts VALUES (1, 'Adam', 10000);

INSERT INTO Accounts VALUES (2, 'jack', 5000);

COMMIT;

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = p\_from\_account;

IF v\_balance < p\_amount THEN

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

RETURN;

END IF;

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_account;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_account;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || p\_amount ||

' from account ' || p\_from\_account ||

' to account ' || p\_to\_account || ' completed.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: One or both accounts do not exist.');

WHEN OTHERS THEN

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

ROLLBACK;

END;

/

BEGIN

TransferFunds(1, 2, 3000);

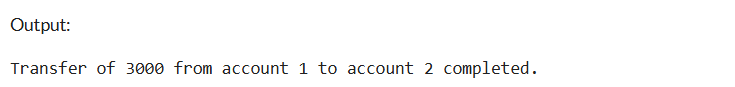
END;

/

SELECT \* FROM ACCOUNTS;

**OUTPUT:**

| **AccountID** | **CustomerName** | **Balance** |
| --- | --- | --- |
| 1 | Adam | 10000 |
| 2 | Jack | 5000 |



| **AccountID** | **CustomerName** | **Balance** |
| --- | --- | --- |
| 1 | Adam | 7000 |
| 2 | Jack | 8000 |

=====================================================================