**SUPERSERT ID: 6398045**

**Exercise 1: Control Structures**

**-- Create CUSTOMERS table**

CREATE TABLE CUSTOMERS (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Age NUMBER,

Balance NUMBER,

IsVIP VARCHAR2(5)

);

-- Insert sample customers

INSERT INTO CUSTOMERS VALUES (1, 'Alice', 65, 12000, 'FALSE');

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

INSERT INTO CUSTOMERS VALUES (3, 'Charlie', 70, 5000, 'FALSE');

INSERT INTO CUSTOMERS VALUES (4, 'Diana', 35, 15000, 'FALSE');

**-- Create LOANS table**

CREATE TABLE LOANS (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES CUSTOMERS(CustomerID),

InterestRate NUMBER,

DueDate DATE

);

-- Insert sample loans

INSERT INTO LOANS VALUES (101, 1, 0.07, SYSDATE + 20);

INSERT INTO LOANS VALUES (102, 2, 0.08, SYSDATE + 45);

INSERT INTO LOANS VALUES (103, 3, 0.09, SYSDATE + 10);

INSERT INTO LOANS VALUES (104, 4, 0.07, SYSDATE + 5);

--code:

**Scenario 1: Apply 1% discount to loan interest rates for customers above 60**

BEGIN

FOR rec IN (

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

FROM CUSTOMERS c

JOIN LOANS l ON c.CustomerID = l.CustomerID

WHERE c.Age > 60

) LOOP

UPDATE LOANS

SET InterestRate = InterestRate - 0.01

WHERE LoanID = rec.LoanID;

END LOOP;

COMMIT;

END;

/

--code:

**Scenario 2: Set IsVIP = TRUE for customers with balance over $10,000**

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;

--code:

**--Scenario 3: Print reminders for loans due in next 30 days**

BEGIN

FOR rec IN (

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

FROM LOANS l

JOIN CUSTOMERS c ON l.CustomerID = c.CustomerID

WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reminder: Loan ID ' || rec.LoanID ||

' for customer ' || rec.Name ||

' is due on ' || TO\_CHAR(rec.DueDate, 'YYYY-MM-DD')

);

END LOOP;

END;

**Output:**

--Scenario1:

--output:

PL/SQL procedure successfully completed.

--Scenario2:

--output:

PL/SQL procedure successfully completed.

--Scenario3:

--output:

Reminder: Loan ID 101 for customer Alice is due on 2025-07-19

Reminder: Loan ID 103 for customer Charlie is due on 2025-07-09

Reminder: Loan ID 104 for customer Diana is due on 2025-07-04

PL/SQL procedure successfully completed.

**Exercise 3: Stored Procedures**

-- Create CUSTOMERS table

CREATE TABLE CUSTOMERS (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Age NUMBER,

Balance NUMBER,

IsVIP VARCHAR2(5)

);

-- Insert sample customers

INSERT INTO CUSTOMERS VALUES (1, 'Alice', 65, 12000, 'FALSE');

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

INSERT INTO CUSTOMERS VALUES (3, 'Charlie', 70, 5000, 'FALSE');

INSERT INTO CUSTOMERS VALUES (4, 'Diana', 35, 15000, 'FALSE');

-- Create LOANS table

CREATE TABLE LOANS (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES CUSTOMERS(CustomerID),

InterestRate NUMBER,

DueDate DATE

);

-- Insert sample loans

INSERT INTO LOANS VALUES (101, 1, 0.07, SYSDATE + 20);

INSERT INTO LOANS VALUES (102, 2, 0.08, SYSDATE + 45);

INSERT INTO LOANS VALUES (103, 3, 0.09, SYSDATE + 10);

INSERT INTO LOANS VALUES (104, 4, 0.07, SYSDATE + 5);

-- Create SAVINGS\_ACCOUNTS table

CREATE TABLE SAVINGS\_ACCOUNTS (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES CUSTOMERS(CustomerID),

Balance NUMBER

);

-- Insert sample savings accounts

INSERT INTO SAVINGS\_ACCOUNTS VALUES (201, 1, 5000);

INSERT INTO SAVINGS\_ACCOUNTS VALUES (202, 2, 8000);

INSERT INTO SAVINGS\_ACCOUNTS VALUES (203, 3, 3000);

INSERT INTO SAVINGS\_ACCOUNTS VALUES (204, 4, 15000);

-- Create EMPLOYEES table

CREATE TABLE EMPLOYEES (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

DepartmentID NUMBER,

Salary NUMBER

);

-- Insert sample employees

INSERT INTO EMPLOYEES VALUES (301, 'Eve', 10, 4000);

INSERT INTO EMPLOYEES VALUES (302, 'Frank', 20, 5000);

INSERT INTO EMPLOYEES VALUES (303, 'Grace', 10, 6000);

-- Create Accounts table

CREATE TABLE ACCOUNTS (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES CUSTOMERS(CustomerID),

Balance NUMBER

);

-- Insert sample accounts

INSERT INTO ACCOUNTS VALUES (401, 1, 10000);

INSERT INTO ACCOUNTS VALUES (402, 1, 5000);

INSERT INTO ACCOUNTS VALUES (403, 2, 2000);

--code:

**--Scenario 1: Process Monthly Interest for all savings accounts**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR rec IN (SELECT AccountID, Balance FROM SAVINGS\_ACCOUNTS) LOOP

UPDATE SAVINGS\_ACCOUNTS

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountID = rec.AccountID;

END LOOP;

COMMIT;

END;

/

-- Create CUSTOMERS table

CREATE TABLE CUSTOMERS (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Age NUMBER,

Balance NUMBER,

IsVIP VARCHAR2(5)

);

-- Insert sample customers

INSERT INTO CUSTOMERS VALUES (1, 'Alice', 65, 12000, 'FALSE');

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

INSERT INTO CUSTOMERS VALUES (3, 'Charlie', 70, 5000, 'FALSE');

INSERT INTO CUSTOMERS VALUES (4, 'Diana', 35, 15000, 'FALSE');

-- Create LOANS table

CREATE TABLE LOANS (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES CUSTOMERS(CustomerID),

InterestRate NUMBER,

DueDate DATE

);

-- Insert sample loans

INSERT INTO LOANS VALUES (101, 1, 0.07, SYSDATE + 20);

INSERT INTO LOANS VALUES (102, 2, 0.08, SYSDATE + 45);

INSERT INTO LOANS VALUES (103, 3, 0.09, SYSDATE + 10);

INSERT INTO LOANS VALUES (104, 4, 0.07, SYSDATE + 5);

-- Create SAVINGS\_ACCOUNTS table

CREATE TABLE SAVINGS\_ACCOUNTS (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES CUSTOMERS(CustomerID),

Balance NUMBER

);

-- Insert sample savings accounts

INSERT INTO SAVINGS\_ACCOUNTS VALUES (201, 1, 5000);

INSERT INTO SAVINGS\_ACCOUNTS VALUES (202, 2, 8000);

INSERT INTO SAVINGS\_ACCOUNTS VALUES (203, 3, 3000);

INSERT INTO SAVINGS\_ACCOUNTS VALUES (204, 4, 15000);

-- Create EMPLOYEES table

CREATE TABLE EMPLOYEES (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

DepartmentID NUMBER,

Salary NUMBER

);

-- Insert sample employees

INSERT INTO EMPLOYEES VALUES (301, 'Eve', 10, 4000);

INSERT INTO EMPLOYEES VALUES (302, 'Frank', 20, 5000);

INSERT INTO EMPLOYEES VALUES (303, 'Grace', 10, 6000);

-- Create Accounts table

CREATE TABLE ACCOUNTS (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES CUSTOMERS(CustomerID),

Balance NUMBER

);

-- Insert sample accounts

INSERT INTO ACCOUNTS VALUES (401, 1, 10000);

INSERT INTO ACCOUNTS VALUES (402, 1, 5000);

INSERT INTO ACCOUNTS VALUES (403, 2, 2000);

--code:

**--Scenario 2: Update Employee Bonus by department**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_DepartmentID IN NUMBER,

p\_BonusPercent IN NUMBER

) IS

BEGIN

UPDATE EMPLOYEES

SET Salary = Salary + (Salary \* p\_BonusPercent / 100)

WHERE DepartmentID = p\_DepartmentID;

COMMIT;

END;

/

-- Create CUSTOMERS table

CREATE TABLE CUSTOMERS (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Age NUMBER,

Balance NUMBER,

IsVIP VARCHAR2(5)

);

-- Insert sample customers

INSERT INTO CUSTOMERS VALUES (1, 'Alice', 65, 12000, 'FALSE');

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

INSERT INTO CUSTOMERS VALUES (3, 'Charlie', 70, 5000, 'FALSE');

INSERT INTO CUSTOMERS VALUES (4, 'Diana', 35, 15000, 'FALSE');

-- Create LOANS table

CREATE TABLE LOANS (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES CUSTOMERS(CustomerID),

InterestRate NUMBER,

DueDate DATE

);

-- Insert sample loans

INSERT INTO LOANS VALUES (101, 1, 0.07, SYSDATE + 20);

INSERT INTO LOANS VALUES (102, 2, 0.08, SYSDATE + 45);

INSERT INTO LOANS VALUES (103, 3, 0.09, SYSDATE + 10);

INSERT INTO LOANS VALUES (104, 4, 0.07, SYSDATE + 5);

-- Create SAVINGS\_ACCOUNTS table

CREATE TABLE SAVINGS\_ACCOUNTS (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES CUSTOMERS(CustomerID),

Balance NUMBER

);

-- Insert sample savings accounts

INSERT INTO SAVINGS\_ACCOUNTS VALUES (201, 1, 5000);

INSERT INTO SAVINGS\_ACCOUNTS VALUES (202, 2, 8000);

INSERT INTO SAVINGS\_ACCOUNTS VALUES (203, 3, 3000);

INSERT INTO SAVINGS\_ACCOUNTS VALUES (204, 4, 15000);

-- Create EMPLOYEES table

CREATE TABLE EMPLOYEES (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

DepartmentID NUMBER,

Salary NUMBER

);

-- Insert sample employees

INSERT INTO EMPLOYEES VALUES (301, 'Eve', 10, 4000);

INSERT INTO EMPLOYEES VALUES (302, 'Frank', 20, 5000);

INSERT INTO EMPLOYEES VALUES (303, 'Grace', 10, 6000);

-- Create Accounts table

CREATE TABLE ACCOUNTS (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES CUSTOMERS(CustomerID),

Balance NUMBER

);

-- Insert sample accounts

INSERT INTO ACCOUNTS VALUES (401, 1, 10000);

INSERT INTO ACCOUNTS VALUES (402, 1, 5000);

INSERT INTO ACCOUNTS VALUES (403, 2, 2000);

--code:

**-- Scenario 3: Transfer Funds between accounts**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_SourceAccountID IN NUMBER,

p\_DestinationAccountID IN NUMBER,

p\_Amount IN NUMBER

) IS

v\_SourceBalance NUMBER;

BEGIN

-- Check source account balance

SELECT Balance INTO v\_SourceBalance

FROM ACCOUNTS

WHERE AccountID = p\_SourceAccountID;

IF v\_SourceBalance < p\_Amount THEN

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

ELSE

-- Deduct from source

UPDATE ACCOUNTS

SET Balance = Balance - p\_Amount

WHERE AccountID = p\_SourceAccountID;

-- Add to destination

UPDATE ACCOUNTS

SET Balance = Balance + p\_Amount

WHERE AccountID = p\_DestinationAccountID;

COMMIT;

END IF;

END;

/

**Output:**

--Scenario1

EXECUTE ProcessMonthlyInterest;

--output:

PL/SQL procedure successfully completed.

--Scenario2

EXECUTE UpdateEmployeeBonus(10, 10);

--output:

PL/SQL procedure successfully completed.

--Scenario3

EXECUTE TransferFunds(401, 402, 2000);

--output1:

PL/SQL procedure successfully completed.

--output2:(if insufficient balance)

ORA-20001: Insufficient balance in source account.

ORA-06512: at "TRANSFERFUNDS", line X