**WEEK-2**

(Implemented in plsqlOneCompiler )

**1)PL SQL Programming**

Exercise 1: Control Structures:

**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

**Question:** Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Customers';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE Customers (

CustomerID NUMBER,

Name VARCHAR2(100),

Age NUMBER,

LoanInterestRate NUMBER

);

/

BEGIN

INSERT INTO Customers VALUES (1, 'Alice', 65, 8.5);

INSERT INTO Customers VALUES (2, 'Bob', 45, 7.5);

INSERT INTO Customers VALUES (3, 'Charlie', 70, 9.0);

COMMIT;

END;

/

COLUMN CustomerID FORMAT 9999

COLUMN Name FORMAT A15

COLUMN Age FORMAT 99

COLUMN LoanInterestRate FORMAT 999.99

SELECT \* FROM Customers;

/

BEGIN

FOR cust IN (SELECT \* FROM Customers) LOOP

IF cust.Age> 60 THEN

UPDATE Customers

SET LoanInterestRate = LoanInterestRate - (LoanInterestRate \* 0.01)

WHERE CustomerID = cust.CustomerID;

END IF;

END LOOP;

END;

/

COLUMN CustomerID FORMAT 9999

COLUMN Name FORMAT A15

COLUMN Age FORMAT 99

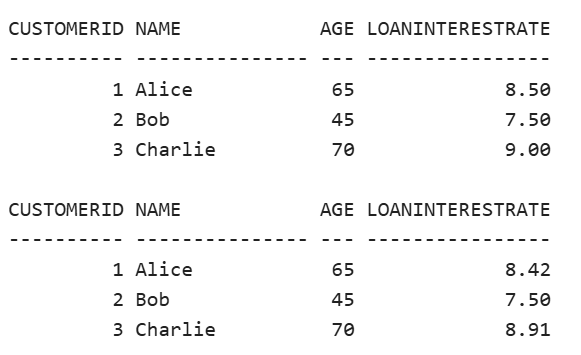
COLUMN LoanInterestRate FORMAT 999.99

-- Then select

SELECT \* FROM Customers;

/

**Output:**



**Scenario 2:** A customer can be promoted to VIP status based on their balance.

**Question:** Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Customers';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE Customers (

CustomerID NUMBER,

Name VARCHAR2(100),

Age NUMBER,

Balance NUMBER,

IsVIPCHAR(1)

);

/

BEGIN

INSERT INTO Customers VALUES (1, 'Alice', 65, 9500, NULL);

INSERT INTO Customers VALUES (2, 'Bob', 45, 12000, NULL);

INSERT INTO Customers VALUES (3, 'Charlie', 70, 10200, NULL);

INSERT INTO Customers VALUES (4, 'David', 38, 7000, NULL);

COMMIT;

END;

/

COLUMN CustomerID FORMAT 9999

COLUMN Name FORMAT A15

COLUMN Age FORMAT 99

COLUMN Balance FORMAT 999999.99

COLUMN IsVIP FORMAT A5

-- Show table before update

SELECT \* FROM Customers;

/

BEGIN

FOR cust IN (SELECT \* FROM Customers) LOOP

IF cust.Balance> 10000 THEN

UPDATE Customers

SET IsVIP = 'Y'

WHERE CustomerID = cust.CustomerID;

ELSE

UPDATE Customers

SET IsVIP = 'N'

WHERE CustomerID = cust.CustomerID;

END IF;

END LOOP;

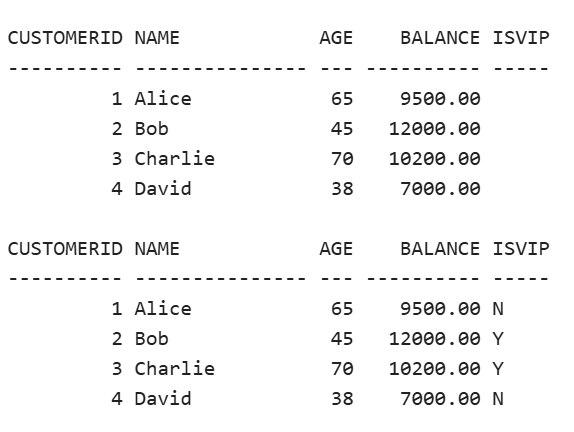
END;

/

SELECT \* FROM Customers;

/

**Output:**

****

**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

**Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer**.**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Loans';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE Loans (

LoanID NUMBER,

CustomerName VARCHAR2(100),

DueDate DATE

);

/

BEGIN

INSERT INTO Loans VALUES (1, 'Alice', SYSDATE + 10); -- within 30 days

INSERT INTO Loans VALUES (2, 'Bob', SYSDATE + 35); -- outside 30 days

INSERT INTO Loans VALUES (3, 'Charlie', SYSDATE + 5); -- within 30 days

INSERT INTO Loans VALUES (4, 'David', SYSDATE - 2); -- already overdue

COMMIT;

END;

/

COLUMN LoanID FORMAT 9999

COLUMN CustomerName FORMAT A15

COLUMN DueDate FORMAT A20

SELECT \* FROM Loans;

/

SELECT

LoanID,

CustomerName,

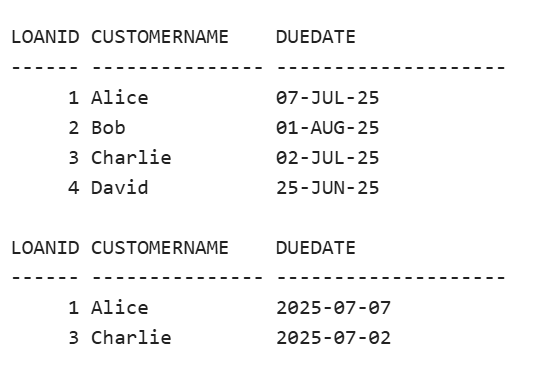
TO\_CHAR(DueDate, 'YYYY-MM-DD') AS DueDate

FROM Loans

WHERE DueDate BETWEEN SYSDATE AND SYSDATE + 30;

/

**Output:**

****

Exercise 3: Stored Procedures:

**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

**Question:** Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE SavingsAccounts';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE SavingsAccounts (

AccountID NUMBER,

CustomerName VARCHAR2(100),

Balance NUMBER

);

/

BEGIN

INSERT INTO SavingsAccounts VALUES (1, 'Alice', 10000);

INSERT INTO SavingsAccounts VALUES (2, 'Bob', 8500);

INSERT INTO SavingsAccounts VALUES (3, 'Charlie', 15000);

COMMIT;

END;

/

COLUMN AccountID FORMAT 9999

COLUMN CustomerName FORMAT A15

COLUMN Balance FORMAT 999999.99

SELECT \* FROM SavingsAccounts;

/

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE SavingsAccounts

SET Balance = Balance + (Balance \* 0.01);

END;

/

BEGIN

ProcessMonthlyInterest;

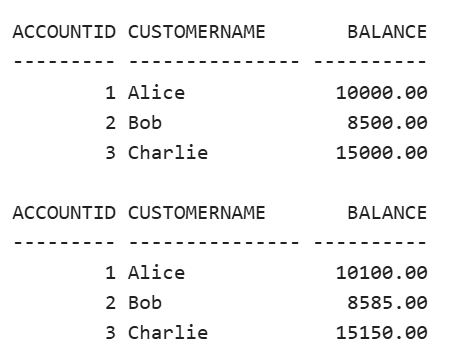
END;

/

SELECT \* FROM SavingsAccounts;

/

**Output:**

****

**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

**Question:** Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Employees';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE Employees (

EmpID NUMBER,

Name VARCHAR2(100),

Department VARCHAR2(50),

Salary NUMBER

);

/

BEGIN

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

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

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

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

COMMIT;

END;

/

COLUMN EmpID FORMAT 9999

COLUMN Name FORMAT A15

COLUMN Department FORMAT A10

COLUMN Salary FORMAT 999999.99

SELECT \* FROM Employees;

/

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

deptName IN VARCHAR2,

bonusPercent IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* bonusPercent / 100)

WHERE Department = deptName;

END;

/

BEGIN

UpdateEmployeeBonus('IT', 10);

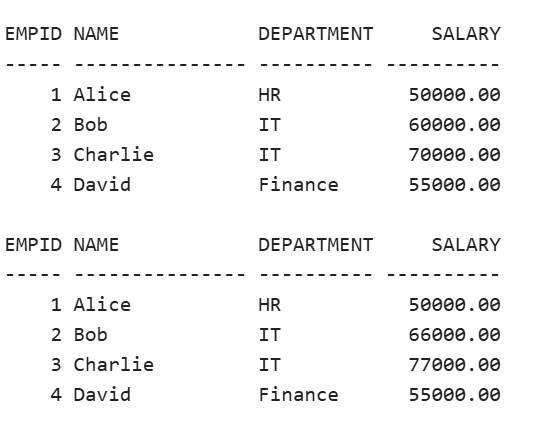
END;

/

SELECT \* FROM Employees;

/

**Output:**

****

**Scenario 3:** Customers should be able to transfer funds between their accounts.

**Question:** Write a stored procedure **TransferFunds** that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Accounts';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE Accounts (

AccountID NUMBER,

Name VARCHAR2(100),

Balance NUMBER

);

/

BEGIN

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

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

INSERT INTO Accounts VALUES (3, 'Charlie', 2000);

COMMIT;

END;

/

COLUMN AccountID FORMAT 9999

COLUMN Name FORMAT A15

COLUMN Balance FORMAT 999999.99

SELECT \* FROM Accounts;

/

CREATE OR REPLACE PROCEDURE TransferFunds (

fromAccount IN NUMBER,

toAccount IN NUMBER,

amount IN NUMBER

) IS

fromBalance NUMBER;

BEGIN

SELECT Balance INTO fromBalance FROM Accounts WHERE AccountID = fromAccount;

IF fromBalance>= amount THEN

UPDATE Accounts SET Balance = Balance - amount WHERE AccountID = fromAccount;

UPDATE Accounts SET Balance = Balance + amount WHERE AccountID = toAccount;

END IF;

END;

/

BEGIN

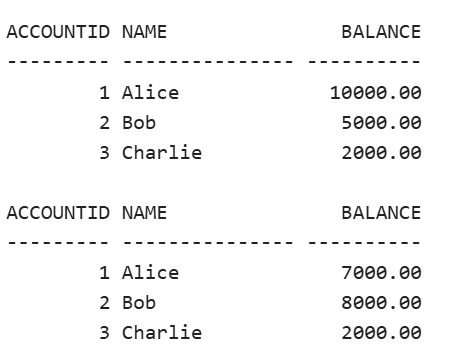
TransferFunds(1, 2, 3000);

END;

/

SELECT \* FROM Accounts;

/

**Output:**