**WEEK-2**

**PL/SQL exercises**

**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.

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.

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.

**Code:**

CREATE TABLE CUSTOMERS (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Age NUMBER,

Balance NUMBER,

IsVIP CHAR(1),

InterestRate NUMBER

);

INSERT INTO CUSTOMERS VALUES (1, 'Arun', 65, 12000, 'N', 10);

INSERT INTO CUSTOMERS VALUES (2, 'Balu', 45, 8000, 'N', 12);

INSERT INTO CUSTOMERS VALUES (3, 'Charan', 70, 15000, 'N', 11);

INSERT INTO CUSTOMERS VALUES (4, 'Dev', 58, 9500, 'N', 13);

CREATE TABLE LOANS (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES CUSTOMERS(CustomerID),

DueDate DATE

);

INSERT INTO LOANS VALUES (101, 1, SYSDATE + 15);

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

INSERT INTO LOANS VALUES (103, 3, SYSDATE + 25);

---- SCENARIO 1 Applying 1% discount for customers over 60

BEGIN

FOR rec IN (

SELECT CustomerID FROM CUSTOMERS WHERE Age > 60

) LOOP

UPDATE CUSTOMERS

SET InterestRate = InterestRate - 1

WHERE CustomerID = rec.CustomerID;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Discount applied to the eligible customers.');

END;

/

---- SCENARIO 2 Marking customers as VIP if Balance > 10000

BEGIN

FOR rec IN (

SELECT CustomerID FROM CUSTOMERS WHERE Balance > 10000

) LOOP

UPDATE CUSTOMERS

SET IsVIP = 'Y'

WHERE CustomerID = rec.CustomerID;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('VIP status updated.');

END;

/

---- SCENARIO 3 Sending loan reminders for due dates 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 <= SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ' || rec.LoanID ||

' for ' || rec.Name ||

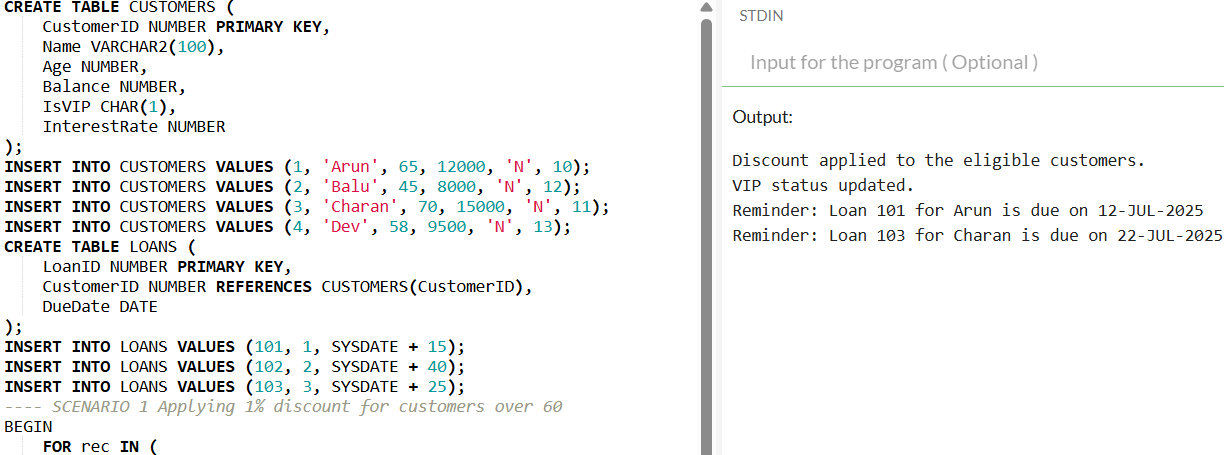
' is due on ' || TO\_CHAR(rec.DueDate, 'DD-MON-YYYY'));

END LOOP;

END;

/

**Output:**



**Explanation:**

* Oracle's version of SQL, known as PL/SQL (Procedural Language/SQL), enables us to create logic utilizing variables, loops, conditions, and blocks, enhancing the programmability and power of database operations.
* In Scenario 1, the age of every consumer is determined using a loop. We apply a 1% reduction by lowering their current loan interest rate by 1 if they are above 60.
* The loop verifies each customer's balance in Scenario 2. The IsVIP flag is changed to 'Y', indicating that they are granted VIP status, if their balance exceeds $10,000.
* The algorithm examines all loans that are due during the next 30 days in Scenario 3. It uses DBMS\_OUTPUT.PUT\_LINE to print a reminder message for each of these loans.

**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.

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.

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.

**Code:**

CREATE TABLE SAVINGS\_ACCOUNTS (

AccountID NUMBER PRIMARY KEY,

Balance NUMBER

);

INSERT INTO SAVINGS\_ACCOUNTS VALUES (1, 10000);

INSERT INTO SAVINGS\_ACCOUNTS VALUES (2, 15000);

CREATE TABLE EMPLOYEES (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DepartmentID NUMBER,

Salary NUMBER

);

INSERT INTO EMPLOYEES VALUES (101, 'Arun', 10, 50000);

INSERT INTO EMPLOYEES VALUES (102, 'Balu', 10, 60000);

INSERT INTO EMPLOYEES VALUES (103, 'Charan', 20, 55000);

CREATE TABLE ACCOUNTS (

AccountID NUMBER PRIMARY KEY,

Balance NUMBER

);

INSERT INTO ACCOUNTS VALUES (201, 20000);

INSERT INTO ACCOUNTS VALUES (202, 10000);

-- PROCEDURE 1: ProcessMonthlyInterest

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

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

UPDATE SAVINGS\_ACCOUNTS

SET Balance = Balance + (rec.Balance \* 0.01)

WHERE AccountID = rec.AccountID;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all savings accounts.');

END;

/

-- PROCEDURE 2: UpdateEmployeeBonus

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_dept\_id IN NUMBER,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

FOR rec IN (

SELECT EmployeeID, Salary FROM EMPLOYEES

WHERE DepartmentID = p\_dept\_id

) LOOP

UPDATE EMPLOYEES

SET Salary = Salary + (rec.Salary \* p\_bonus\_percent / 100)

WHERE EmployeeID = rec.EmployeeID;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Bonus updated for Department ID: ' || p\_dept\_id);

END;

/

-- PROCEDURE 3: TransferFunds

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_acc IN NUMBER,

p\_to\_acc IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM ACCOUNTS WHERE AccountID = p\_from\_acc;

IF v\_balance < p\_amount THEN

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

ELSE

UPDATE ACCOUNTS

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_acc;

UPDATE ACCOUNTS

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_acc;

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

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('One of the accounts does not exist.');

WHEN OTHERS THEN

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

END;

/

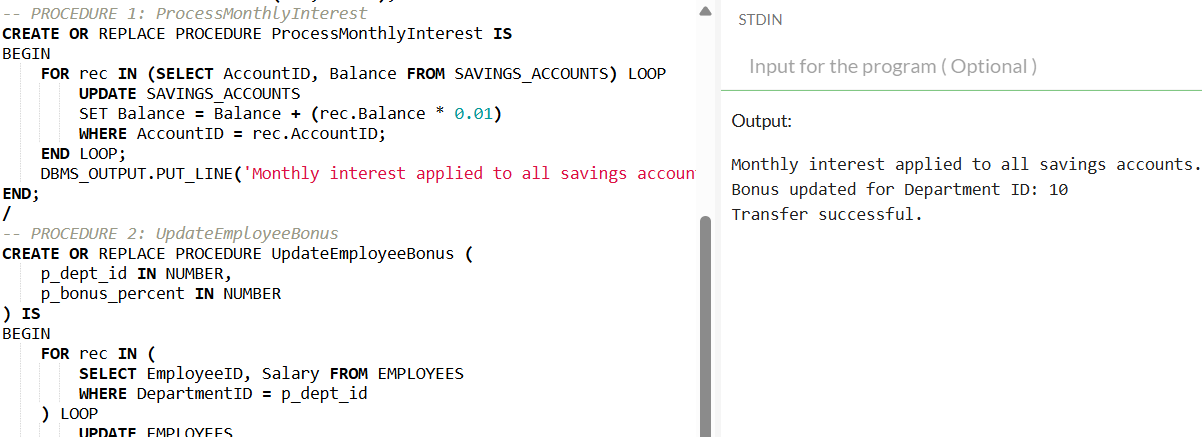
-- EXECUTION OF PROCEDURES

EXEC ProcessMonthlyInterest;

EXEC UpdateEmployeeBonus(10, 10);

EXEC TransferFunds(201, 202, 5000);

**Output:**



**Explanation:**

* A named block of code that is kept in the database and may be used at any time to carry out particular operations, such as computations or changes, is called a stored procedure.
* In the first case, we developed a process that adds 1% monthly interest to the current balance in each savings account.
* In the second instance, employees receive bonuses from the bank. We created a process that adds the bonus to the salary of staff members in a department after receiving the department ID and bonus percentage.
* In the third case, we developed a fund transfer process that transfers funds across accounts after determining whether the originating account has sufficient balance.
* By automating routine processes like interest processing, payroll updates, and transfers, these protocols assist the bank in enhancing their operations and working.