SuperSet ID : 6314238 **BAPANAPALLI PREM SAI SIDDHIK**

HANDSON EXERCISES - WEEK 2

Skill : PL/SQL

Exercise 1: Control Structures

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

Question 1 : 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.

CODE :

```
SET SERVEROUTPUT ON;
BEGIN
  FOR cust IN (SELECT CustomerID, DOB FROM Customers) LOOP
    IF MONTHS_BETWEEN(SYSDATE, cust.DOB) / 12 > 60 THEN
      UPDATE Loans
      SET InterestRate = InterestRate - 1
      WHERE CustomerID = cust.CustomerID;
      DBMS_OUTPUT.PUT_LINE('Discount applied for CustomerID: ' || cust.CustomerID);
    END IF;
  END LOOP;
  COMMIT;
END;
/
```

OUTPUT :

```
Discount applied for CustomerID: 3
Discount applied for CustomerID: 4

PL/SQL procedure successfully completed.
```

SELECT * FROM loans;

```
    LOANID CUSTOMERID LOANAMOUNT INTERESTRATE STARTDATE ENDDATE
---------- ---------- ---------- ------------ --------- ---------
         1          1       5000            5 28-JUN-25 28-JUN-30
         2          3      10000           .5 28-JUN-25 18-JUL-25
         3          4       7000         -.5 28-JUN-25 26-SEP-25
```

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.

CODE :

```
ALTER TABLE Customers ADD IsVIP CHAR(1);

SET SERVEROUTPUT ON;

BEGIN
  FOR cust IN (SELECT CustomerID, Balance FROM Customers) LOOP
    IF cust.Balance > 10000 THEN
      UPDATE Customers
      SET IsVIP = 'Y'
      WHERE CustomerID = cust.CustomerID;
      DBMS_OUTPUT.PUT_LINE('VIP status granted to CustomerID: ' || cust.CustomerID);
    END IF;
  END LOOP;
  COMMIT;
END;
/
```

OUTPUT :

```
VIP status granted to CustomerID: 3
VIP status granted to CustomerID: 5

PL/SQL procedure successfully completed.
```

SELECT * FROM customers;

OUTPUT :

```
CUSTOMERID
----------
NAME
--------------------------------------------------------------------------------
DOB           BALANCE LASTMODIF I
--------- ---------- --------- -
         1
John Doe
15-MAY-85        1000 28-JUN-25

         2
Jane Smith
20-JUL-90        1500 28-JUN-25

CUSTOMERID
----------
NAME
--------------------------------------------------------------------------------
DOB           BALANCE LASTMODIF I
--------- ---------- --------- -

         3
Senior One
28-JUN-60       12000 28-JUN-25 Y

         4
Senior Two

CUSTOMERID
----------
NAME
--------------------------------------------------------------------------------
DOB           BALANCE LASTMODIF I
--------- ---------- --------- -
28-JUN-55        8000 28-JUN-25

         5
Middle Age Joe
28-JUN-80       15000 28-JUN-25 Y
```

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 :

```
SET SERVEROUTPUT ON;
BEGIN
  FOR loan_rec IN (
    SELECT L.LoanID, C.Name, L.EndDate
    FROM Loans L
    JOIN Customers C ON L.CustomerID = C.CustomerID
    WHERE L.EndDate BETWEEN SYSDATE AND SYSDATE + 30
  ) LOOP
    DBMS_OUTPUT.PUT_LINE('Reminder sent for LoanID: ' || loan_rec.LoanID);
  END LOOP;
END;
/
```

OUTPUT :

```
Reminder sent for LoanID: 2

PL/SQL procedure successfully completed.
```

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.

CODE :

Creating a Procedure:

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
  FOR acc IN (SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings') LOOP
    UPDATE Accounts
    SET Balance = Balance + (acc.Balance * 0.01),
      LastModified = SYSDATE
    WHERE AccountID = acc.AccountID;
    DBMS_OUTPUT.PUT_LINE('Interest added to AccountID: ' || acc.AccountID);
  END LOOP;
END;
```

```
Procedure created.
```

Running the Procedure :

```
BEGIN
  ProcessMonthlyInterest;
END;
/
```

OUTPUT:

```
Interest added to AccountID: 1
Interest added to AccountID: 3
Interest added to AccountID: 5

PL/SQL procedure successfully completed.
```

SELECT * FROM ACCOUNTS;

ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE	LASTMODIF
1	1	Savings	1110	28-JUN-25
2	2	Checking	1400	28-JUN-25
3	3	Savings	12120	28-JUN-25
4	4	Checking	8000	28-JUN-25
5	5	Savings	15150	28-JUN-25

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.

CODE :

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(p_dept VARCHAR2,
p_bonus_percent NUMBER) IS

BEGIN

  FOR emp IN (SELECT EmployeeID, Salary FROM Employees WHERE Department =
p_dept)  LOOP

    UPDATE Employees

    SET Salary = Salary + (emp.Salary * p_bonus_percent / 100)

    WHERE EmployeeID = emp.EmployeeID;

    DBMS_OUTPUT.PUT_LINE('Bonus added to EmployeeID: ' || emp.EmployeeID);

  END LOOP;

END;

/
```

```
Procedure created.
```

Running the Procedure :

```
BEGIN
  UpdateEmployeeBonus('IT', 10);
  END;
  /
```

OUTPUT:

```
Bonus added to EmployeeID: 2

PL/SQL procedure successfully completed.
```

SELECT * FROM employees;

OUTPUT:

```
SQL> SELECT * FROM employees;

EMPLOYEEID
----------
NAME
--------------------------------------------------------------------------------
POSITION                                             SALARY
-------------------------------------------------- ----------
DEPARTMENT                                         HIREDATE
-------------------------------------------------- ---------
         1
Alice Johnson
Manager                                                70000
HR                                                 15-JUN-15


EMPLOYEEID
----------
NAME
--------------------------------------------------------------------------------
POSITION                                             SALARY
-------------------------------------------------- ----------
DEPARTMENT                                         HIREDATE
-------------------------------------------------- ---------
         2
Bob Brown
Developer                                              63000
IT                                                 20-MAR-17


EMPLOYEEID
----------
NAME
--------------------------------------------------------------------------------
POSITION                                             SALARY
-------------------------------------------------- ----------
DEPARTMENT                                         HIREDATE
-------------------------------------------------- ---------
         3
Charlie Green
Analyst                                                50000
Finance                                            10-APR-19


EMPLOYEEID
----------
NAME
--------------------------------------------------------------------------------
POSITION                                             SALARY
-------------------------------------------------- ----------
DEPARTMENT                                         HIREDATE
-------------------------------------------------- ---------
         4
Diana White
Clerk                                                  40000
Finance                                            25-FEB-21
```

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 :

Creating a Procedure:

```
 v_balance NUMBER;

BEGIN

  SELECT Balance INTO v_balance FROM Accounts WHERE AccountID = p_from;

  IF v_balance < p_amount THEN

    RAISE_APPLICATION_ERROR(-20003, 'Insufficient balance in source
account.');

  END IF;

  UPDATE Accounts SET Balance = Balance - p_amount WHERE AccountID =
p_from;

  UPDATE Accounts SET Balance = Balance + p_amount WHERE AccountID = p_to;

  DBMS_OUTPUT.PUT_LINE('Transferred ' || p_amount || ' from Account ' || p_from
|| ' to Account ' || p_to);

EXCEPTION

  WHEN OTHERS THEN

    DBMS_OUTPUT.PUT_LINE('Error in TransferFunds: ' || SQLERRM);

END;

/
```

```
Procedure created.
```

Running the Procedure :

```
BEGIN
TransferFunds(2, 1, 100);
END;
/
```

OUTPUT:

```
Transferred 100 from Account 2 to Account 1

PL/SQL procedure successfully completed.
```

SuperSet ID : 6314238 **BAPANAPALLI PREM SAI SIDDHIK**

SELECT * FROM accounts;

OUTPUT:

ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE	LASTMODIF
1	1	Savings	1110	28-JUN-25
2	2	Checking	1400	28-JUN-25
3	3	Savings	12120	28-JUN-25
4	4	Checking	8000	28-JUN-25
5	5	Savings	15150	28-JUN-25