PL SQL(Week 2)

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
FOR cust_rec IN (
SELECT l.LoanID,InterestRate
FROM Customer c
JOIN Loans l ON c.CustomerID=l.CustomerID
WHERE c.Age  > 60
) LOOP
UPDATE Loans
SET InterestRate = cust_rec.InterestRate - 1
WHERE LoanID = cust_rec.LoanID;
END LOOP;
COMMIT;
END;
```

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
FOR vip_rec IN (
SELECT CustomerID
FROM Customers
WHERE Balance>10000
) LOOP
UPDATE Customers
SET IsVIP = '1'
WHERE CustomerID = vip_rec.CustomerID;
END LOOP;
COMMIT;
END;
```

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
FOR loan_rec IN(
SELECT l.LoanID , c.Name , l.DueDate
FROM Loan LoanJOIN Customers c ON l.CustomerID = c.CustomerID
WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE +30
) LOOP
  DBMS_OUTPUT.PUT_LINE(
      'Reminder: Dear ' || loan_rec.Name ||
      ', your loan (ID: ' || loan_rec.LoanID ||
      ') is due on ' || TO_CHAR(loan_rec.DueDate, 'DD-Mon-YYYY')
    );
    END LOOP;
    END;
```

Exercise 2: Error Handling

Scenario 1: Handle exceptions during fund transfers between accounts.

- **Question:** Write a stored procedure **SafeTransferFunds** that transfers funds between two accounts. Ensure that if any error occurs (e.g., insufficient funds), an appropriate error message is logged and the transaction is rolled back.

```
BEGIN
FOR loan_rec IN(
SELECT l.LoanID , c.Name , l.DueDate
FROM Loan LoanJOIN Customers c ON l.CustomerID = c.CustomerID
WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE +30
) LOOP
  DBMS_OUTPUT.PUT_LINE(
      'Reminder: Dear ' || loan_rec.Name ||
      ', your loan (ID: ' || loan_rec.LoanID ||
      ') is due on ' || TO_CHAR(loan_rec.DueDate, 'DD-Mon-YYYY')
    );
    END LOOP;
    END;

    CREATE OR REPLACE PROCEDURE SafeTransferFunds (
  p_from_account   IN NUMBER,
  p_to_account     IN NUMBER,
  p_amount         IN NUMBER
) AS
  v_balance        NUMBER;
BEGIN
  SELECT Balance INTO v_balance
  FROM Accounts
  WHERE AccountID = p_from_account;

  IF v_balance < p_amount THEN
    RAISE_APPLICATION_ERROR(-19001, 'Insufficient funds in the source account.');
  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;
```

Scenario 2: Manage errors when updating employee salaries.

- **Question:** Write a stored procedure **UpdateSalary** that increases the salary of an employee by a given percentage. If the employee ID does not exist, handle the exception and log an error message.

```
CREATE OR REPLACE PROCEDURE UpdateSalary (
  p_employee_id     IN NUMBER,
  p_percentage_inc  IN NUMBER
) AS
  v_old_salary  NUMBER;
BEGIN
  SELECT Salary INTO v_old_salary
  FROM Employees
  WHERE EmployeeID = p_employee_id;
  UPDATE Employees
  SET Salary = Salary + (Salary * p_percentage_inc / 100)
  WHERE EmployeeID = p_employee_id;

  COMMIT;

  DBMS_OUTPUT.PUT_LINE('Salary updated successfully. Employee ID: ' || p_employee_id);

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Error: Employee ID ' || p_employee_id || ' does not exist.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
    ROLLBACK;
END;
```

Scenario 3: Ensure data integrity when adding a new customer.

- **Question:** Write a stored procedure **AddNewCustomer** that inserts a new customer into the Customers table. If a customer with the same ID already exists, handle the exception by logging an error and preventing the insertion.

```
CREATE OR REPLACE PROCEDURE AddNewCustomer (
  p_customer_id   IN NUMBER,
  p_name          IN VARCHAR2,
  p_age           IN NUMBER,
  p_balance       IN NUMBER
) AS
BEGIN
  -- Attempt to insert new customer
  INSERT INTO Customer (CustomerID, Name, Age, Balance)
  VALUES (p_customer_id, p_name, p_age, p_balance);

  COMMIT;

  DBMS_OUTPUT.PUT_LINE('Customer added successfully: ' || p_customer_id);

EXCEPTION
  WHEN DUP_VAL_ON_INDEX THEN
    DBMS_OUTPUT.PUT_LINE('Error: Customer ID ' || p_customer_id || ' already exists.');
    ROLLBACK;
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
    ROLLBACK;
END;
```