WEEK-2

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: 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 (
 cust_id   NUMBER PRIMARY KEY,
 age       NUMBER,
 balance   NUMBER,
 vip_flag  VARCHAR2(5)
);

CREATE TABLE loans (
 loan_id   NUMBER PRIMARY KEY,
 cust_id   NUMBER,
 int_rate  NUMBER,
 due_on    DATE,
 FOREIGN KEY (cust_id) REFERENCES customers(cust_id)
);
```

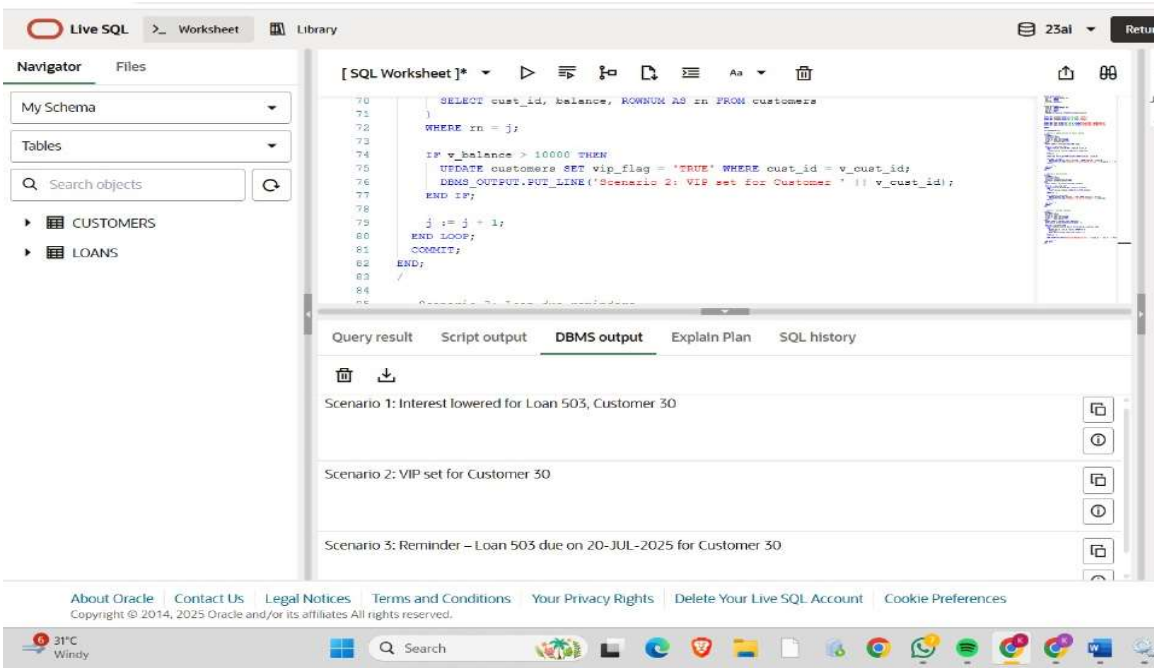
```
INSERT INTO customers VALUES (10, 68, 12500, 'FALSE');
INSERT INTO customers VALUES (20, 55, 9400, 'FALSE');
INSERT INTO customers VALUES (30, 72, 17800, 'FALSE');

INSERT INTO loans VALUES (501, 10, 11, TO_DATE('05-JUL-2025','DD-MON-YYYY'));
INSERT INTO loans VALUES (502, 20, 9, TO_DATE('15-AUG-2025','DD-MON-YYYY'));
INSERT INTO loans VALUES (503, 30, 10, TO_DATE('20-JUL-2025','DD-MON-YYYY'));

COMMIT;

SET SERVEROUTPUT ON;

-- Scenario 1: Reduce interest for senior citizens
DECLARE
 i NUMBER := 1;
 loan_total NUMBER;
 v_loan_id loans.loan_id%TYPE;
 v_cust_id loans.cust_id%TYPE;
 v_age     customers.age%TYPE;
BEGIN
 SELECT COUNT(*) INTO loan_total FROM loans;

 WHILE i <= loan_total LOOP
  SELECT loan_id, cust_id INTO v_loan_id, v_cust_id
  FROM (
   SELECT loan_id, cust_id, ROWNUM AS rn FROM loans
  )
  WHERE rn = i;

  SELECT age INTO v_age FROM customers WHERE cust_id = v_cust_id;
```

```
    IF v_age > 60 THEN
      UPDATE loans SET int_rate = int_rate - 1 WHERE loan_id = v_loan_id;
      DBMS_OUTPUT.PUT_LINE('Scenario 1: Interest lowered for Loan ' || v_loan_id || ', Customer ' ||
v_cust_id);
    END IF;

    i := i + 1;
  END LOOP;
  COMMIT;
END;
/

-- Scenario 2: Set VIP status
DECLARE
  j NUMBER := 1;
  cust_total NUMBER;
  v_cust_id customers.cust_id%TYPE;
  v_balance customers.balance%TYPE;
BEGIN
  SELECT COUNT(*) INTO cust_total FROM customers;

  WHILE j <= cust_total LOOP
    SELECT cust_id, balance INTO v_cust_id, v_balance
    FROM (
      SELECT cust_id, balance, ROWNUM AS rn FROM customers
    )
    WHERE rn = j;

    IF v_balance > 10000 THEN
      UPDATE customers SET vip_flag = 'TRUE' WHERE cust_id = v_cust_id;
      DBMS_OUTPUT.PUT_LINE('Scenario 2: VIP set for Customer ' || v_cust_id);
```

```
    END IF;

    j := j + 1;
  END LOOP;
  COMMIT;
END;
/

-- Scenario 3: Loan due reminders
DECLARE
  k NUMBER := 1;
  due_total NUMBER;
  v_loan_id loans.loan_id%TYPE;
  v_cust_id loans.cust_id%TYPE;
  v_due     loans.due_on%TYPE;
BEGIN
  SELECT COUNT(*) INTO due_total FROM loans
  WHERE due_on BETWEEN SYSDATE AND SYSDATE + 30;

  WHILE k <= due_total LOOP
    SELECT loan_id, cust_id, due_on INTO v_loan_id, v_cust_id, v_due
    FROM (
      SELECT loan_id, cust_id, due_on, ROWNUM AS rn
      FROM loans
      WHERE due_on BETWEEN SYSDATE AND SYSDATE + 30
    )
    WHERE rn = k;

    DBMS_OUTPUT.PUT_LINE('Scenario 3: Reminder – Loan ' || v_loan_id || ' due on ' ||
TO_CHAR(v_due, 'DD-MON-YYYY') || ' for Customer ' || v_cust_id);
```

```
    k := k + 1;
  END LOOP;
END;
```

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.

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 :

```
-- Scenario 1: Process Monthly Interest
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE savings_accounts';
EXCEPTION WHEN OTHERS THEN NULL;
END;
/

CREATE TABLE savings_accounts (
  account_id     NUMBER PRIMARY KEY,
  customer_name  VARCHAR2(50),
  balance        NUMBER
);
```

```
INSERT INTO savings_accounts VALUES (101, 'Ravi Kumar', 10000);
INSERT INTO savings_accounts VALUES (102, 'Anita Sharma', 15000);
INSERT INTO savings_accounts VALUES (103, 'Kiran Patel', 8000);
COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
  FOR acc IN (SELECT account_id, balance FROM savings_accounts) LOOP
    UPDATE savings_accounts
    SET balance = balance + (acc.balance * 0.01)
    WHERE account_id = acc.account_id;

    DBMS_OUTPUT.PUT_LINE('Updated interest for Account ID ' || acc.account_id);
  END LOOP;
END;
/

-- Run the procedure
BEGIN
  ProcessMonthlyInterest;
END;
/

-- Scenario 2: Update Employee Bonus
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE employees';
EXCEPTION WHEN OTHERS THEN NULL;
END;
/
```

```
CREATE TABLE employees (
  emp_id       NUMBER PRIMARY KEY,
  emp_name     VARCHAR2(50),
  department_id NUMBER,
  salary       NUMBER
);

INSERT INTO employees VALUES (201, 'Suresh Babu', 10, 40000);
INSERT INTO employees VALUES (202, 'Priya Raj', 10, 45000);
INSERT INTO employees VALUES (203, 'Vikram Desai', 20, 42000);
COMMIT;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(
  p_dept_id IN NUMBER,
  p_bonus_percent IN NUMBER
) IS
BEGIN
  UPDATE employees
  SET salary = salary + (salary * p_bonus_percent / 100)
  WHERE department_id = p_dept_id;

  DBMS_OUTPUT.PUT_LINE('Bonus applied to Department ID ' || p_dept_id);
END;
/

-- Run the procedure
BEGIN
  UpdateEmployeeBonus(10, 10);  -- 10% bonus to dept 10
END;
/
```

```
-- Scenario 3: Transfer Funds
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE accounts';
EXCEPTION WHEN OTHERS THEN NULL;
END;
/

CREATE TABLE accounts (
  account_id    NUMBER PRIMARY KEY,
  holder_name   VARCHAR2(50),
  balance       NUMBER
);

INSERT INTO accounts VALUES (301, 'Meena Joshi', 20000);
INSERT INTO accounts VALUES (302, 'Arjun Singh', 10000);
INSERT INTO accounts VALUES (303, 'Divya Iyer', 30000);
COMMIT;

CREATE OR REPLACE PROCEDURE TransferFunds(
  p_from_acct IN NUMBER,
  p_to_acct   IN NUMBER,
  p_amount    IN NUMBER
) IS
  v_balance NUMBER;
BEGIN
  SELECT balance INTO v_balance FROM accounts WHERE account_id = p_from_acct;

  IF v_balance < p_amount THEN
    DBMS_OUTPUT.PUT_LINE('Insufficient balance in source account.');
  ELSE
    UPDATE accounts SET balance = balance - p_amount WHERE account_id = p_from_acct;
```

```
   UPDATE accounts SET balance = balance + p_amount WHERE account_id = p_to_acct;


   DBMS_OUTPUT.PUT_LINE('Transferred ₹' || p_amount || ' from ' || p_from_acct || ' to ' ||
p_to_acct);
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
   DBMS_OUTPUT.PUT_LINE('Account not found.');
END;
/


-- Run the procedure
BEGIN
  TransferFunds(301, 302, 5000);
END;
/
```

OUTPUT: