1.CONTROL STRUCTURES

SCENARIO 1

**CODE:**

-- sample table

CREATE TABLE customers (

  customer\_id NUMBER,

  name VARCHAR2(100),

  age NUMBER,

  loan\_interest\_rate NUMBER,

  balance NUMBER,

  isVIP VARCHAR2(5)

);

--Sample test data

INSERT INTO customers VALUES (1, 'John Doe', 65, 9.5, 12000, 'FALSE');

INSERT INTO customers VALUES (2, 'Jane Smith', 45, 10.0, 8000, 'FALSE');

INSERT INTO customers VALUES (3, 'Alice', 70, 11.0, 20000, 'FALSE');

COMMIT;

SET SERVEROUTPUT ON;

--PL/SQL block to execute task

BEGIN

  FOR cust IN (

    SELECT customer\_id, age, loan\_interest\_rate

    FROM customers

    WHERE age > 60

  )

  LOOP

    UPDATE customers

    SET loan\_interest\_rate = loan\_interest\_rate - 1

    WHERE customer\_id = cust.customer\_id;

    dbms\_output.put\_line('1% discount applied to Customer ID: ' || cust.customer\_id);

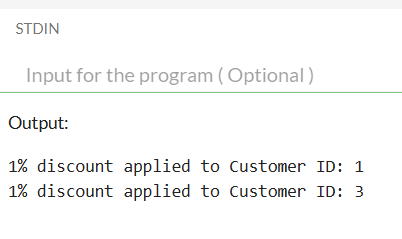
  END LOOP;

  COMMIT;

END;

/

**OUTPUT:**



SCENARIO 2

**CODE:**

-- customers table

CREATE TABLE customers (

  customer\_id NUMBER,

  name VARCHAR2(100),

  age NUMBER,

  loan\_interest\_rate NUMBER,

  balance NUMBER,

  isVIP VARCHAR2(5)

);

-- sample data with different balances

INSERT INTO customers VALUES (1, 'John Doe', 65, 9.5, 12000, 'FALSE');

INSERT INTO customers VALUES (2, 'Jane Smith', 45, 10.0, 8000, 'FALSE');

INSERT INTO customers VALUES (3, 'Alice', 70, 11.0, 20000, 'FALSE');

INSERT INTO customers VALUES (4, 'Robert', 30, 10.5, 9000, 'FALSE');

INSERT INTO customers VALUES (5, 'Dinesh', 25, 10.2, 15000, 'FALSE');

COMMIT;

--Enabling console output

SET SERVEROUTPUT ON;

-- PL/SQL block to mark VIPs

BEGIN

  FOR cust IN (

    SELECT customer\_id, name, balance

    FROM customers

    WHERE balance > 10000

  ) LOOP

    UPDATE customers

    SET isVIP = 'TRUE'

    WHERE customer\_id = cust.customer\_id;

    dbms\_output.put\_line(cust.name || ' (Customer ID: ' || cust.customer\_id || ') is now marked as VIP.');

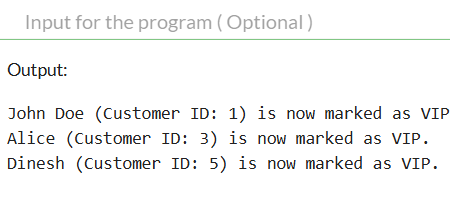
  END LOOP;

  COMMIT;

END;

/

**OUTPUT:**



SCENARIO 3

**CODE:**

--loans table

CREATE TABLE loans (

  loan\_id NUMBER,

  customer\_id NUMBER,

  due\_date DATE

);

--sample loans data

BEGIN

  INSERT INTO loans VALUES (101, 1, SYSDATE + 5);    -- Due in 5 days

  INSERT INTO loans VALUES (102, 2, SYSDATE + 40);   -- Due after 40 days

  INSERT INTO loans VALUES (103, 3, SYSDATE + 10);   -- Due in 10 days

  INSERT INTO loans VALUES (104, 4, SYSDATE + 60);   -- Due after 60 days

  INSERT INTO loans VALUES (105, 5, SYSDATE + 2);    -- Due in 2 days

  COMMIT;

END;

/

--Enabling console output

SET SERVEROUTPUT ON;

-- PL/SQL block to Send reminders

BEGIN

  FOR loan IN (

    SELECT loan\_id, customer\_id, due\_date

    FROM loans

    WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30

  ) LOOP

    dbms\_output.put\_line(

      'Reminder: Loan ID ' || loan.loan\_id ||

      ' for Customer ID ' || loan.customer\_id ||

      ' is due on ' || TO\_CHAR(loan.due\_date, 'DD-Mon-YYYY')

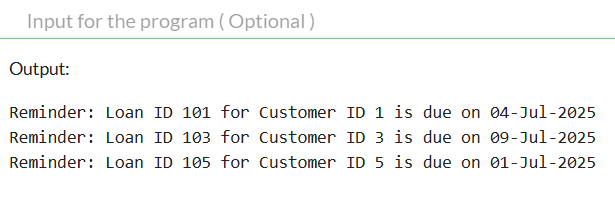
    );

  END LOOP;

END;

/

**OUTPUT:**

****

2. STORED PROCEDURES:

**SCENARIO 1:**

**CODE:**

-- sample table

CREATE TABLE accounts (

  account\_id NUMBER,

  account\_type VARCHAR2(20),

  balance NUMBER

);

-- Inserting sample data

INSERT INTO accounts VALUES (101, 'savings', 10000);

INSERT INTO accounts VALUES (102, 'current', 5000);

INSERT INTO accounts VALUES (103, 'savings', 20000);

COMMIT;

-- Creating procedure

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

  FOR acc IN (

    SELECT account\_id, balance

    FROM accounts

    WHERE LOWER(account\_type) = 'savings'

  ) LOOP

    UPDATE accounts

    SET balance = balance + (balance \* 0.01)

    WHERE account\_id = acc.account\_id;

  END LOOP;

  COMMIT;

END;

/

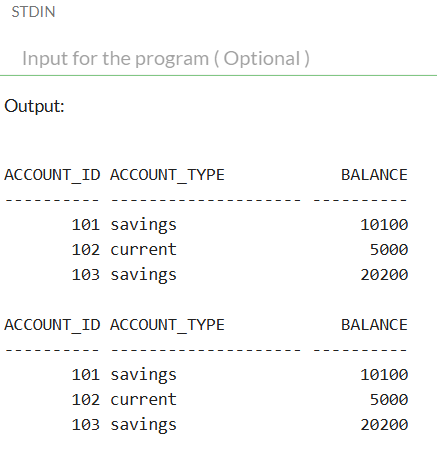
--calling the procedure separately

EXEC ProcessMonthlyInterest;

-- Viewing updated balances

SELECT \* FROM accounts;

**OUTPUT:**

****

**SCENARIO 2:**

**CODE:**

--Creating employees table

CREATE TABLE employees (

  emp\_id NUMBER,

  emp\_name VARCHAR2(50),

  department\_id NUMBER,

  salary NUMBER

);

-- Inserting sample employee data

INSERT INTO employees VALUES (1, 'John', 10, 50000);

INSERT INTO employees VALUES (2, 'Jane', 20, 60000);

INSERT INTO employees VALUES (3, 'Dinesh', 10, 55000);

INSERT INTO employees VALUES (4, 'Swathi', 30, 70000);

COMMIT;

--Creating  the UpdateEmployeeBonus procedure

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

  dept\_id IN NUMBER,

  bonus\_percent IN NUMBER

) IS

BEGIN

  UPDATE employees

  SET salary = salary + (salary \* bonus\_percent / 100)

  WHERE department\_id = dept\_id;

  COMMIT;

END;

/

--Calling the procedure

EXEC UpdateEmployeeBonus(10, 10);

--output

SELECT \* FROM employees;