1.CONTROL STRUCTURES

SCENARIO 1

**CODE:**

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;

/

SCENARIO 2

**CODE:**

--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;

/

SCENARIO 3

**CODE:**

-- 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;

/

2. STORED PROCEDURES:

**SCENARIO 1:**

**CODE:**

-- 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;

**SCENARIO 2:**

**CODE:**

--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;

/

SET SERVEROUTPUT ON;

--Calling the procedure

EXEC UpdateEmployeeBonus(10, 10);

--output

SELECT \* FROM employees

**SCENARIO 3:**

**CODE:**

--procedure

CREATE OR REPLACE PROCEDURE TransferFunds (

  from\_account IN NUMBER,

  to\_account IN NUMBER,

  amount IN NUMBER

) IS

  from\_balance NUMBER;

BEGIN

  --to get the balance of the source account

  SELECT balance INTO from\_balance

  FROM accounts

  WHERE account\_id = from\_account;

  -- Check if sufficient balance exists or not

    IF from\_balance < amount THEN

    RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance in source account');

  END IF;

  -- Deduct

  UPDATE accounts

  SET balance = balance - amount

  WHERE account\_id = from\_account;

  -- Adding to destination account

  UPDATE accounts

  SET balance = balance + amount

  WHERE account\_id = to\_account;

  COMMIT;

  -- Showing success message

  DBMS\_OUTPUT.PUT\_LINE('Transfer of Rs. ' || amount ||' from Account ' || from\_account || 'to Account ' || to\_account || ' completed successfully.');

END;

/

--Enabling output

SET SERVEROUTPUT ON;

-- Calling the procedure

EXEC TransferFunds(103, 102, 3000);

--View account balances after transfer

SELECT \* FROM accounts;