***EXERCISE 1: CONTROL STRUCTURE***

***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.

*CODE:*

-- Step 1: Create Customers table

CREATE TABLE Customers (

customer\_id NUMBER PRIMARY KEY,

age NUMBER,

balance NUMBER,

isVIP VARCHAR2(5)

);

-- Step 2: Create Loans table

CREATE TABLE Loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

interest\_rate NUMBER,

due\_date DATE

);

-- Step 3: Insert data

INSERT INTO Customers VALUES (1, 65, 15000, 'FALSE');

INSERT INTO Customers VALUES (2, 45, 9000, 'FALSE');

INSERT INTO Customers VALUES (3, 70, 8000, 'FALSE');

INSERT INTO Customers VALUES (4, 38, 12000, 'FALSE');

INSERT INTO Loans VALUES (101, 1, 7.5, SYSDATE + 10);

INSERT INTO Loans VALUES (102, 2, 8.0, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 6.8, SYSDATE + 5);

INSERT INTO Loans VALUES (104, 4, 9.0, SYSDATE + 25);

COMMIT;

-- Step 4: Logic - Apply 1% discount for age > 60

BEGIN

FOR cust IN (

SELECT customer\_id FROM Customers WHERE age > 60

) LOOP

UPDATE Loans

SET interest\_rate = interest\_rate - 1

WHERE customer\_id = cust.customer\_id;

END LOOP;

COMMIT;

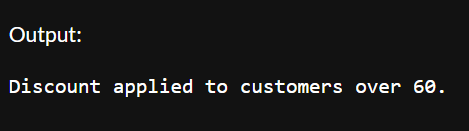
DBMS\_OUTPUT.PUT\_LINE('Discount applied to customers over 60.');

END;

/

* **USING OneCompiler**

**OUPUT:**

****

***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:*

-- 🧹 Step 1: Drop tables if they already exist

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Loans';

EXECUTE IMMEDIATE 'DROP TABLE Customers';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

-- 🧱 Step 2: Create Customers table

CREATE TABLE Customers (

customer\_id NUMBER PRIMARY KEY,

age NUMBER,

balance NUMBER,

isVIP VARCHAR2(5)

);

-- 🧱 Step 3: Create Loans table

CREATE TABLE Loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER REFERENCES Customers(customer\_id),

interest\_rate NUMBER,

due\_date DATE

);

-- 🧪 Step 4: Insert test data

INSERT INTO Customers VALUES (1, 65, 15000, 'FALSE');

INSERT INTO Customers VALUES (2, 45, 9000, 'FALSE');

INSERT INTO Customers VALUES (3, 70, 8000, 'FALSE');

INSERT INTO Customers VALUES (4, 38, 12000, 'FALSE');

INSERT INTO Loans VALUES (101, 1, 7.5, SYSDATE + 10);

INSERT INTO Loans VALUES (102, 2, 8.0, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 6.8, SYSDATE + 5);

INSERT INTO Loans VALUES (104, 4, 9.0, SYSDATE + 25);

COMMIT;

-- 🧹 Step 1: Drop tables if they already exist (optional in OneCompiler)

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Loans';

EXECUTE IMMEDIATE 'DROP TABLE Customers';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

-- 🧱 Step 2: Create Customers table

CREATE TABLE Customers (

customer\_id NUMBER PRIMARY KEY,

age NUMBER,

balance NUMBER,

isVIP VARCHAR2(5)

);

-- 🧱 Step 3: Create Loans table

CREATE TABLE Loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER REFERENCES Customers(customer\_id),

interest\_rate NUMBER,

due\_date DATE

);

-- 🧪 Step 4: Insert test data

INSERT INTO Customers VALUES (1, 65, 15000, 'FALSE');

INSERT INTO Customers VALUES (2, 45, 9000, 'FALSE');

INSERT INTO Customers VALUES (3, 70, 8000, 'FALSE');

INSERT INTO Customers VALUES (4, 38, 12000, 'FALSE');

INSERT INTO Loans VALUES (101, 1, 7.5, SYSDATE + 10);

INSERT INTO Loans VALUES (102, 2, 8.0, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 6.8, SYSDATE + 5);

INSERT INTO Loans VALUES (104, 4, 9.0, SYSDATE + 25);

COMMIT;

-----Set VIP status for customers with balance > $10,000

BEGIN

FOR cust IN (

SELECT customer\_id FROM Customers WHERE balance > 10000

) LOOP

UPDATE Customers

SET isVIP = 'TRUE'

WHERE customer\_id = cust.customer\_id;

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('VIP status updated for high-balance customers.');

END;

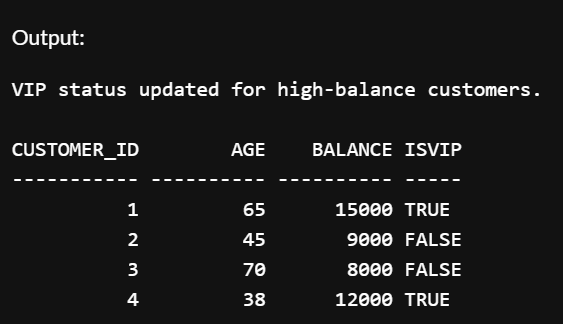
/

SELECT \* FROM Customers;

SELECT \* FROM Loans;

* **USE OneCompiler**

**OUPUT:**

****

**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:**

-- 🧹 Step 1: Drop tables if they already exist

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Loans';

EXECUTE IMMEDIATE 'DROP TABLE Customers';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

-- 🧱 Step 2: Create Customers table

CREATE TABLE Customers (

customer\_id NUMBER PRIMARY KEY,

age NUMBER,

balance NUMBER,

isVIP VARCHAR2(5)

);

-- 🧱 Step 3: Create Loans table

CREATE TABLE Loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER REFERENCES Customers(customer\_id),

interest\_rate NUMBER,

due\_date DATE

);

-- 🧪 Step 4: Insert test data

INSERT INTO Customers VALUES (1, 65, 15000, 'FALSE');

INSERT INTO Customers VALUES (2, 45, 9000, 'FALSE');

INSERT INTO Customers VALUES (3, 70, 8000, 'FALSE');

INSERT INTO Customers VALUES (4, 38, 12000, 'FALSE');

INSERT INTO Loans VALUES (101, 1, 7.5, SYSDATE + 10);

INSERT INTO Loans VALUES (102, 2, 8.0, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 6.8, SYSDATE + 5);

INSERT INTO Loans VALUES (104, 4, 9.0, SYSDATE + 25);

COMMIT;

-----Scenario 3: Print reminder for loans due in next 30 days

BEGIN

FOR loan\_rec IN (

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

FROM Loans

WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan\_rec.loan\_id ||

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

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

END LOOP;

END;

/

SELECT \* FROM Customers;

SELECT \* FROM Loans;

* **USING OneCompiler**

**OUTPUT:**

