**PROCEDURAL LANGUAGE/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.**

**Solution:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE LOANS';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

CREATE TABLE LOANS (

LOAN\_ID NUMBER,

CUSTOMER\_NAME VARCHAR2(50),

AGE NUMBER,

OLD\_RATE NUMBER(5,2),

NEW\_RATE NUMBER(5,2),

STATUS VARCHAR2(20)

);

BEGIN

INSERT INTO LOANS VALUES (101, 'John Doe', 70, 4.61, NULL, NULL);

INSERT INTO LOANS VALUES (102, 'Alice Smith', 65, 4.15, NULL, NULL);

INSERT INTO LOANS VALUES (103, 'Robert Brown', 45, 6.00, NULL, NULL);

INSERT INTO LOANS VALUES (104, 'Mary Johnson', 76, 5.08, NULL, NULL);

INSERT INTO LOANS VALUES (105, 'Michael Davis', 29, 4.00, NULL, NULL);

INSERT INTO LOANS VALUES (106, 'Jennifer Wilson', 75, 6.00, NULL, NULL);

INSERT INTO LOANS VALUES (107, 'David Lee', 59, 5.00, NULL, NULL);

END;

BEGIN

FOR rec IN (SELECT \* FROM LOANS) LOOP

IF rec.AGE > 60 THEN

UPDATE LOANS

SET NEW\_RATE = OLD\_RATE - 0.04, -- Subtract 0.04 for 1% of original rate approx

STATUS = 'Discount Applied'

WHERE LOAN\_ID = rec.LOAN\_ID;

ELSE

UPDATE LOANS

SET NEW\_RATE = OLD\_RATE,

STATUS = 'No Discount'

WHERE LOAN\_ID = rec.LOAN\_ID;

END IF;

END LOOP;

END;

SELECT STATUS, CUSTOMER\_NAME, LOAN\_ID, AGE,

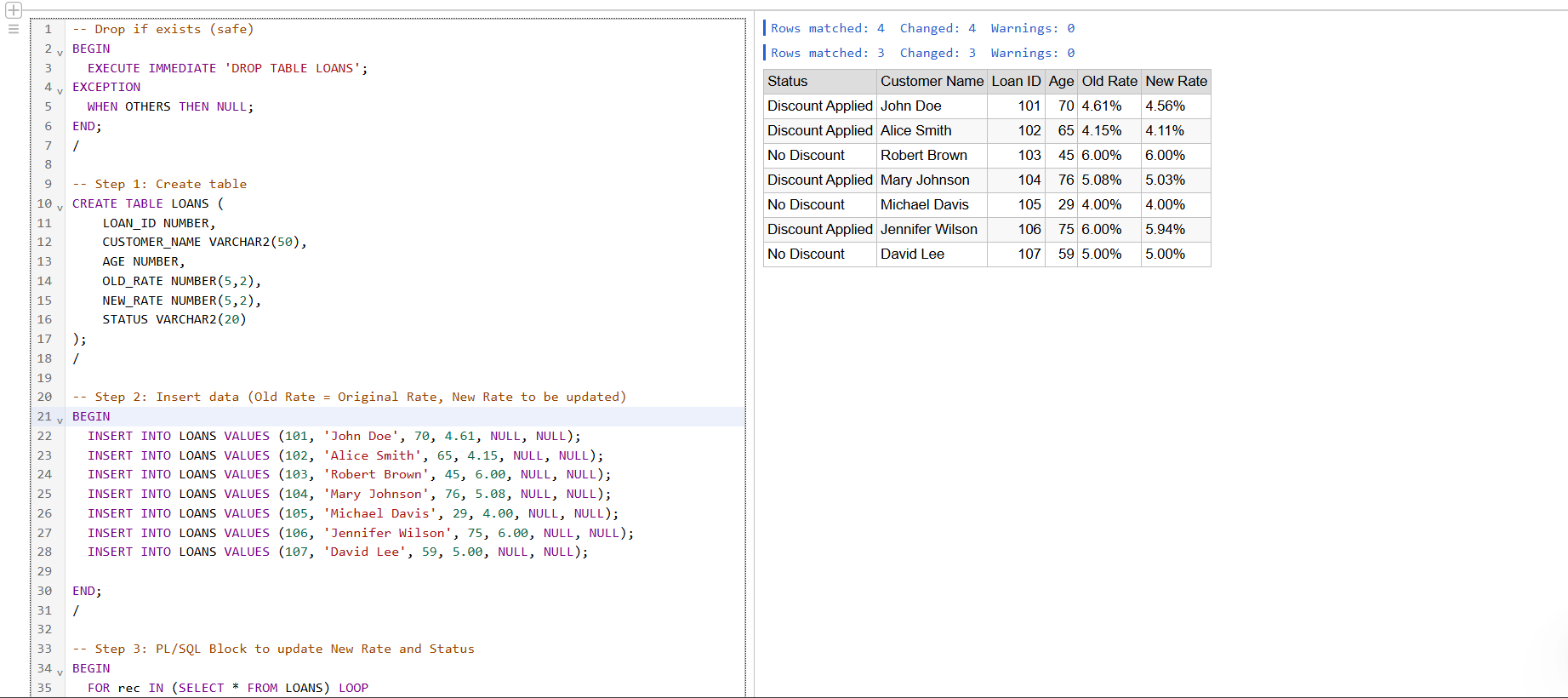
TO\_CHAR(OLD\_RATE, 'FM9990.00') || '%' AS "Old Rate",

TO\_CHAR(NEW\_RATE, 'FM9990.00') || '%' AS "New Rate"

FROM LOANS

ORDER BY LOAN\_ID;

**OUTPUT:**



**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**

**Solution:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE CUSTOMERS';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

CREATE TABLE CUSTOMERS (

CUSTOMER\_ID NUMBER PRIMARY KEY,

CUSTOMER\_NAME VARCHAR2(50),

BALANCE NUMBER(10,2),

ISVIP VARCHAR2(5)

);

BEGIN

INSERT INTO CUSTOMERS VALUES (1, 'Taekook', 9500, NULL);

INSERT INTO CUSTOMERS VALUES (2, 'Jungkook', 12000, NULL);

INSERT INTO CUSTOMERS VALUES (3, 'Taehyung', 10000, NULL);

INSERT INTO CUSTOMERS VALUES (4, 'Abinaya', 15000, NULL);

END;

BEGIN

FOR rec IN (SELECT \* FROM CUSTOMERS) LOOP

IF rec.BALANCE > 10000 THEN

UPDATE CUSTOMERS

SET ISVIP = 'TRUE'

WHERE CUSTOMER\_ID = rec.CUSTOMER\_ID;

ELSE

UPDATE CUSTOMERS

SET ISVIP = 'FALSE'

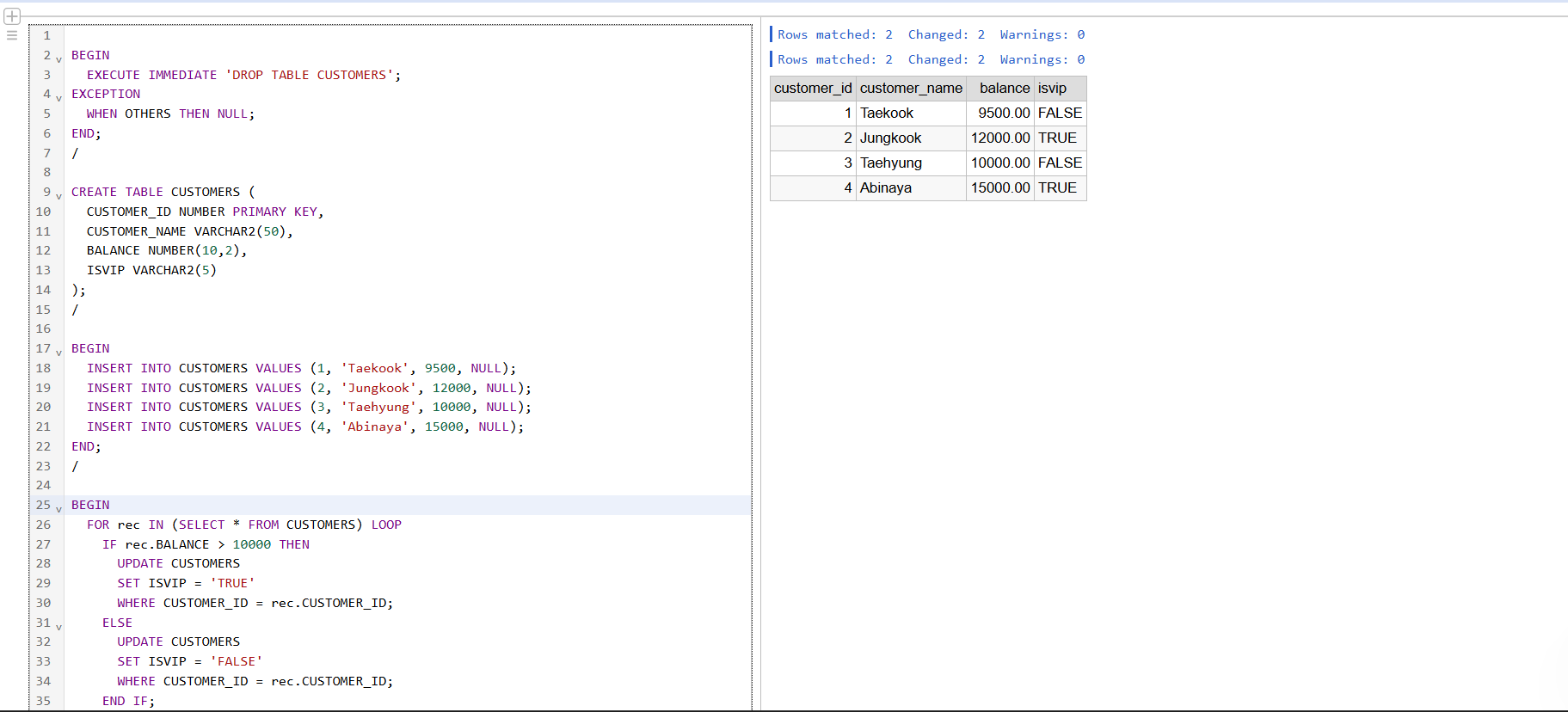
WHERE CUSTOMER\_ID = rec.CUSTOMER\_ID;

END IF;

END LOOP;

END;

SELECT \* FROM CUSTOMERS;

**OUTPUT:**

**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.**

**Solution:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE LOANS';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE LOANS (

LOAN\_ID NUMBER PRIMARY KEY,

CUSTOMER\_NAME VARCHAR2(50),

DUE\_DATE DATE

);

/

BEGIN

INSERT INTO LOANS VALUES (101, 'John Doe', SYSDATE + 10);

INSERT INTO LOANS VALUES (102, 'Alice Smith', SYSDATE + 35);

INSERT INTO LOANS VALUES (103, 'Robert Brown', SYSDATE + 5);

INSERT INTO LOANS VALUES (104, 'Mary Johnson', SYSDATE - 2);

INSERT INTO LOANS VALUES (105, 'David Lee', SYSDATE + 25);

END;

/

BEGIN

FOR loan\_rec IN (

SELECT LOAN\_ID, CUSTOMER\_NAME, 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 ' || loan\_rec.CUSTOMER\_NAME ||

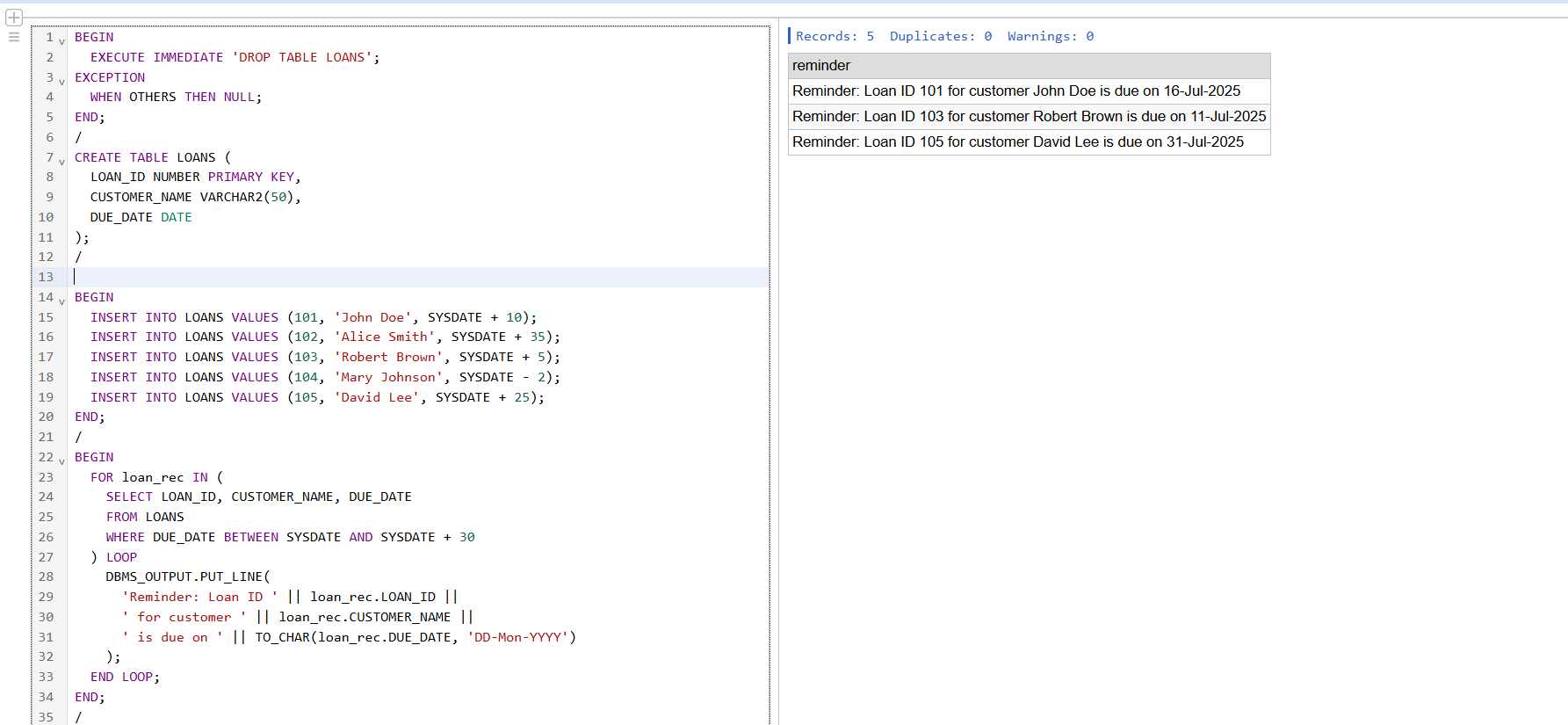
' is due on ' || TO\_CHAR(loan\_rec.DUE\_DATE, 'DD-Mon-YYYY')

);

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.**

**Solution:**

-- Step 1: Drop and create the table

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE ACCOUNTS';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE ACCOUNTS (

ACCOUNT\_ID NUMBER PRIMARY KEY,

CUSTOMER\_ID NUMBER,

ACCOUNT\_TYPE VARCHAR2(20), -- 'Savings' or 'Checking'

BALANCE NUMBER(12,2)

);

/

-- Step 2: Insert test data

BEGIN

INSERT INTO ACCOUNTS VALUES (1, 1, 'Savings', 12241.20);

INSERT INTO ACCOUNTS VALUES (2, 2, 'Savings', 8160.80);

INSERT INTO ACCOUNTS VALUES (3, 3, 'Checking', 5000.00);

INSERT INTO ACCOUNTS VALUES (4, 4, 'Savings', 15301.50);

INSERT INTO ACCOUNTS VALUES (5, 5, 'Checking', 7000.00);

INSERT INTO ACCOUNTS VALUES (6, 6, 'Savings', 20402.00);

INSERT INTO ACCOUNTS VALUES (7, 7, 'Checking', 6000.00);

INSERT INTO ACCOUNTS VALUES (8, 8, 'Savings', 9180.90);

INSERT INTO ACCOUNTS VALUES (9, 9, 'Savings', 25500.50);

INSERT INTO ACCOUNTS VALUES (10, 10, 'Checking', 4000.00);

END;

/

-- Step 3: PL/SQL block to apply 1% interest to savings accounts

DECLARE

v\_old\_balance ACCOUNTS.BALANCE%TYPE;

v\_new\_balance ACCOUNTS.BALANCE%TYPE;

v\_updated\_count NUMBER := 0;

v\_skipped\_count NUMBER := 0;

BEGIN

FOR acc IN (SELECT \* FROM ACCOUNTS ORDER BY ACCOUNT\_ID) LOOP

v\_old\_balance := acc.BALANCE;

DBMS\_OUTPUT.PUT\_LINE('Account ID: ' || acc.ACCOUNT\_ID ||

' | Customer ID: ' || acc.CUSTOMER\_ID ||

' | Type: ' || acc.ACCOUNT\_TYPE);

DBMS\_OUTPUT.PUT\_LINE('Old Balance: ' || TO\_CHAR(v\_old\_balance, '99999.99'));

IF UPPER(acc.ACCOUNT\_TYPE) = 'SAVINGS' THEN

v\_new\_balance := ROUND(v\_old\_balance + (v\_old\_balance \* 0.01), 2); -- Apply 1% interest

UPDATE ACCOUNTS

SET BALANCE = v\_new\_balance

WHERE ACCOUNT\_ID = acc.ACCOUNT\_ID;

DBMS\_OUTPUT.PUT\_LINE('New Balance: ' || TO\_CHAR(v\_new\_balance, '99999.99'));

DBMS\_OUTPUT.PUT\_LINE('Status: Interest successfully applied');

v\_updated\_count := v\_updated\_count + 1;

ELSE

DBMS\_OUTPUT.PUT\_LINE('New Balance: Not Applicable');

DBMS\_OUTPUT.PUT\_LINE('Status: Skipped – Not a savings account');

v\_skipped\_count := v\_skipped\_count + 1;

END IF;

DBMS\_OUTPUT.PUT\_LINE(''); -- Line break between accounts

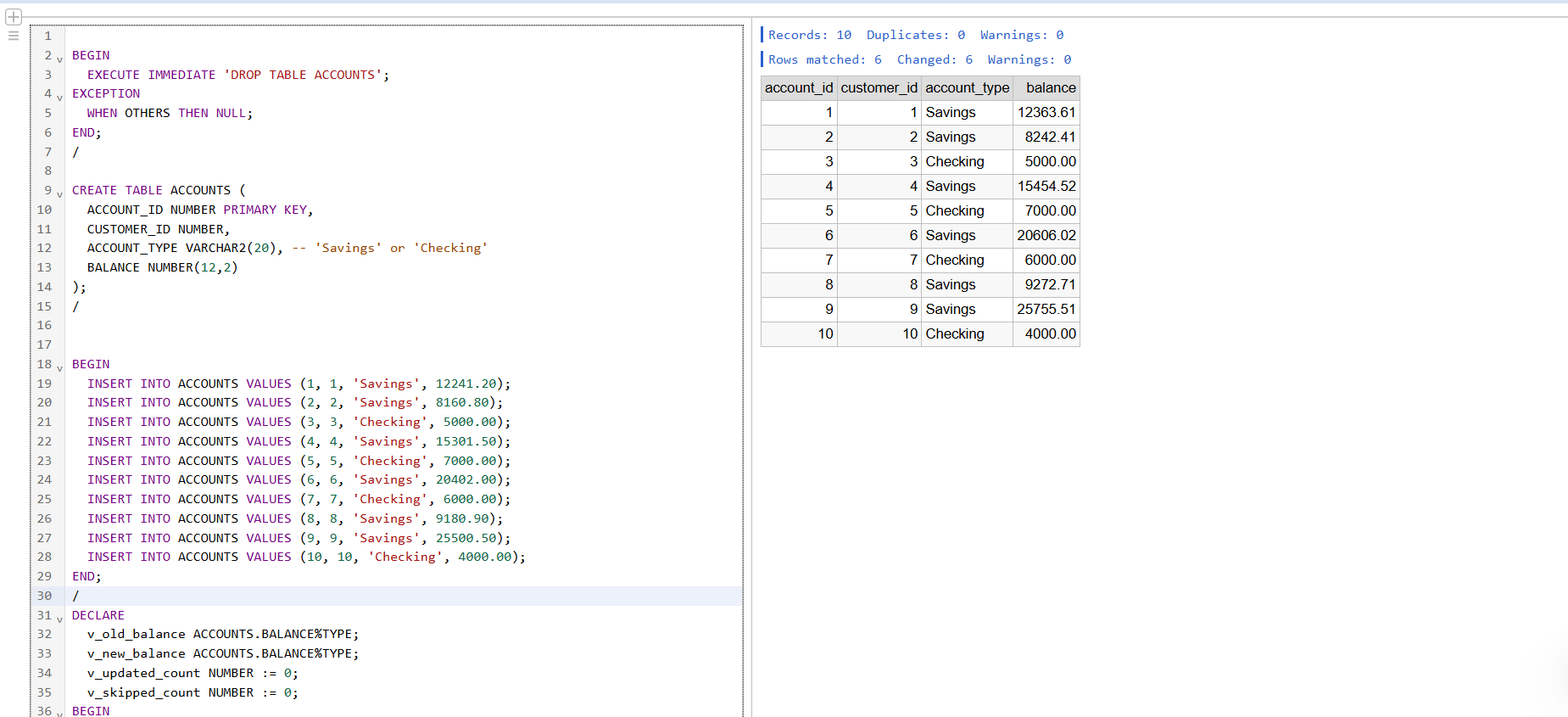
END LOOP;

DBMS\_OUTPUT.PUT\_LINE(v\_updated\_count || ' savings account(s) updated.');

DBMS\_OUTPUT.PUT\_LINE(v\_skipped\_count || ' account(s) skipped.');

END;

**OUTPUT:**



**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.**

**SOLUTION:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE EMPLOYEES';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE EMPLOYEES (

EMPLOYEE\_ID NUMBER PRIMARY KEY,

DEPARTMENT VARCHAR2(30),

SALARY NUMBER(10,2)

);

/

-- Step 2: Insert sample data

BEGIN

INSERT INTO EMPLOYEES VALUES (1, 'HR', 50000);

INSERT INTO EMPLOYEES VALUES (2, 'IT', 66000);

INSERT INTO EMPLOYEES VALUES (3, 'Finance', 62000);

INSERT INTO EMPLOYEES VALUES (4, 'Sales', 58000);

INSERT INTO EMPLOYEES VALUES (5, 'HR', 54000);

INSERT INTO EMPLOYEES VALUES (6, 'IT', 69000);

INSERT INTO EMPLOYEES VALUES (7, 'IT', 67100);

END;

/

-- Step 3: Create the stored procedure

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_pct IN NUMBER

)

IS

v\_old\_salary EMPLOYEES.SALARY%TYPE;

v\_new\_salary EMPLOYEES.SALARY%TYPE;

v\_count NUMBER := 0;

BEGIN

FOR emp IN (

SELECT \* FROM EMPLOYEES

WHERE UPPER(DEPARTMENT) = UPPER(p\_department)

) LOOP

v\_old\_salary := emp.SALARY;

v\_new\_salary := ROUND(v\_old\_salary \* (1 + p\_bonus\_pct / 100), 2);

-- Update salary

UPDATE EMPLOYEES

SET SALARY = v\_new\_salary

WHERE EMPLOYEE\_ID = emp.EMPLOYEE\_ID;

-- Print details

DBMS\_OUTPUT.PUT\_LINE('Employee ID : ' || emp.EMPLOYEE\_ID || ' | Department: ' || emp.DEPARTMENT);

DBMS\_OUTPUT.PUT\_LINE('Old Salary : ' || TO\_CHAR(v\_old\_salary, '999999.00') ||

' | New Salary: ' || TO\_CHAR(v\_new\_salary, '999999.00'));

DBMS\_OUTPUT.PUT\_LINE('Bonus Applied : ' || p\_bonus\_pct || '%');

DBMS\_OUTPUT.PUT\_LINE('');

v\_count := v\_count + 1;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(v\_count || ' employee(s) in department "' || p\_department || '" received a ' || p\_bonus\_pct || '% bonus.');

END;

**OUTPUT:**

