**Scenario 1:**

**Code**

-- Drop if exists

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE savings\_accounts';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

-- Create savings\_accounts table

CREATE TABLE savings\_accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

balance NUMBER(10, 2)

);

INSERT INTO savings\_accounts VALUES (101, 1, 5000);

INSERT INTO savings\_accounts VALUES (102, 2, 10000);

INSERT INTO savings\_accounts VALUES (103, 3, 8000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR rec IN (SELECT account\_id, balance FROM savings\_accounts) LOOP

UPDATE savings\_accounts

SET balance = balance + (balance \* 0.01)

WHERE account\_id = rec.account\_id;

END LOOP;

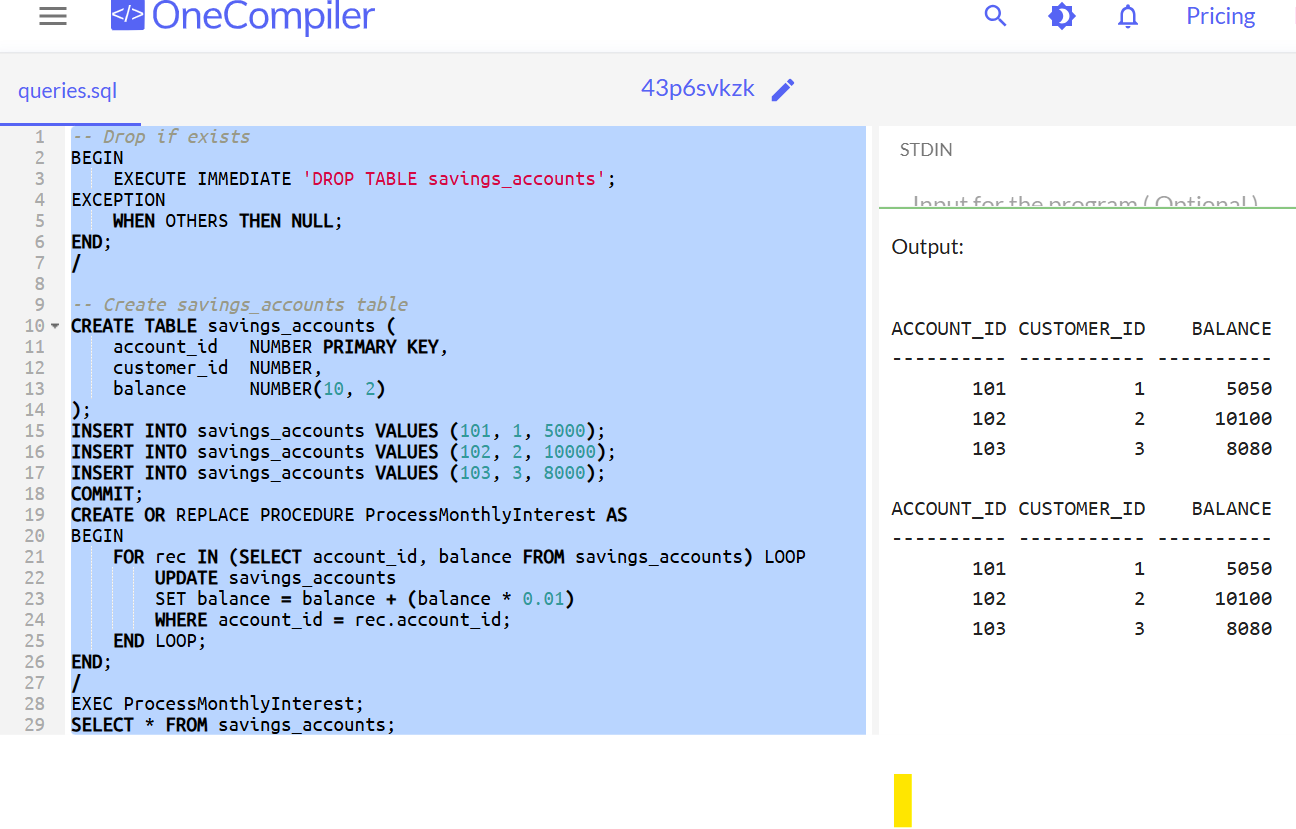
END;

/

EXEC ProcessMonthlyInterest;

SELECT \* FROM savings\_accounts;

**output**



**Scenario 2:**

**Code**

-- Enable DBMS Output

SET SERVEROUTPUT ON;

-- Drop table if exists

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE employees';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

-- Create employees table

CREATE TABLE employees (

emp\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

department\_id NUMBER,

salary NUMBER(10, 2)

);

-- Insert sample data

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

INSERT INTO employees VALUES (2, 'Bob', 10, 55000);

INSERT INTO employees VALUES (3, 'Charlie', 20, 60000);

COMMIT;

-- Create stored procedure to update bonus

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_id IN NUMBER,

bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE employees

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

WHERE department\_id = dept\_id;

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || bonus\_percent || '% applied to department ' || dept\_id);

END;

/

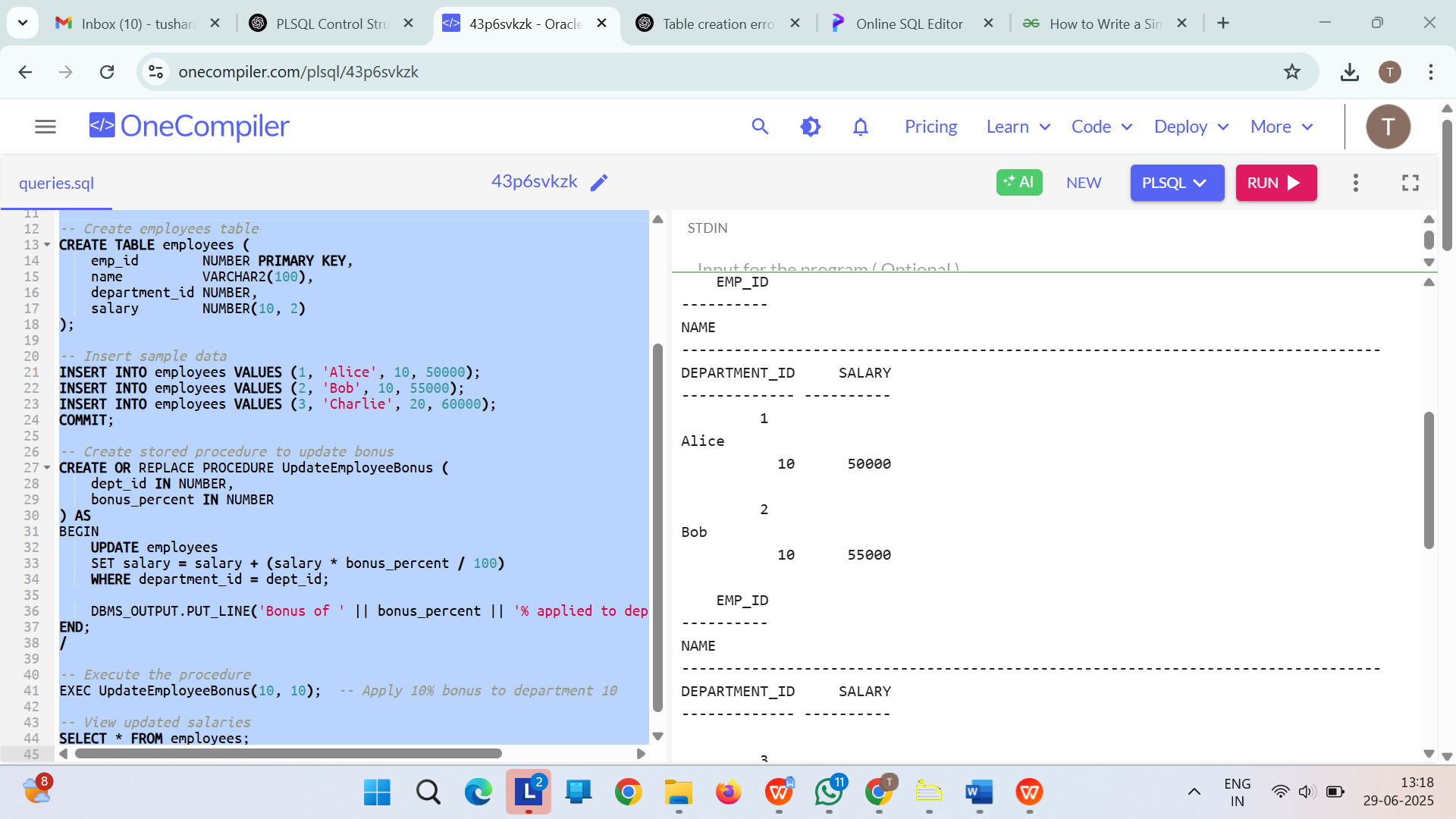
-- Execute the procedure

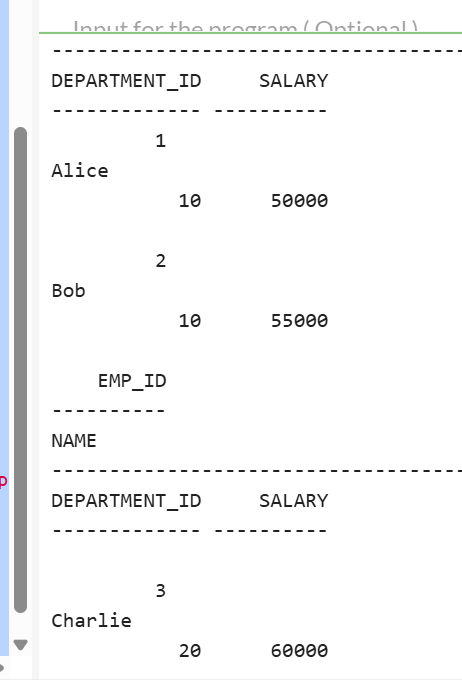
EXEC UpdateEmployeeBonus(10, 10); -- Apply 10% bonus to department 10

-- View updated salaries

SELECT \* FROM employees;

**Output**





**Scenario 3:**

**Code**

-- Enable DBMS Output

SET SERVEROUTPUT ON;

-- Drop table if exists

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE bank\_accounts';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

-- Create bank\_accounts table

CREATE TABLE bank\_accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

balance NUMBER(10, 2)

);

-- Insert sample data

INSERT INTO bank\_accounts VALUES (201, 1, 10000);

INSERT INTO bank\_accounts VALUES (202, 2, 5000);

COMMIT;

-- Create stored procedure for fund transfer

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account IN NUMBER,

to\_account IN NUMBER,

amount IN NUMBER

) AS

insufficient\_balance EXCEPTION;

curr\_balance NUMBER;

BEGIN

-- Get current balance of sender

SELECT balance INTO curr\_balance

FROM bank\_accounts

WHERE account\_id = from\_account;

-- Check for sufficient balance

IF curr\_balance < amount THEN

RAISE insufficient\_balance;

END IF;

-- Deduct from sender and add to receiver

UPDATE bank\_accounts

SET balance = balance - amount

WHERE account\_id = from\_account;

UPDATE bank\_accounts

SET balance = balance + amount

WHERE account\_id = to\_account;

DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || amount ||

' from Account ' || from\_account ||

' to Account ' || to\_account || ' successful.');

EXCEPTION

WHEN insufficient\_balance THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance.');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: One or both accounts not found.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Unexpected error: ' || SQLERRM);

END;

/

-- Execute procedure (successful)

EXEC TransferFunds(201, 202, 2000);

-- Try with insufficient balance

EXEC TransferFunds(202, 201, 10000);

-- View final account balances

SELECT \* FROM bank\_accounts;

**Output**

