**Exercise 1: Control Structures**

**Scenario 1: Discount Loan Interest for Seniors (>60 years)**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE bank\_customers';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE bank\_customers (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(50),

age NUMBER,

loan\_interest NUMBER

);

INSERT INTO bank\_customers VALUES (101, 'Raj', 58, 9.5);

INSERT INTO bank\_customers VALUES (102, 'Latha', 65, 10.0);

INSERT INTO bank\_customers VALUES (103, 'Vijay', 70, 8.5);

COMMIT;

BEGIN

FOR cust IN (SELECT customer\_id FROM bank\_customers WHERE age > 60) LOOP

UPDATE bank\_customers

SET loan\_interest = loan\_interest \* 0.99

WHERE customer\_id = cust.customer\_id;

END LOOP;

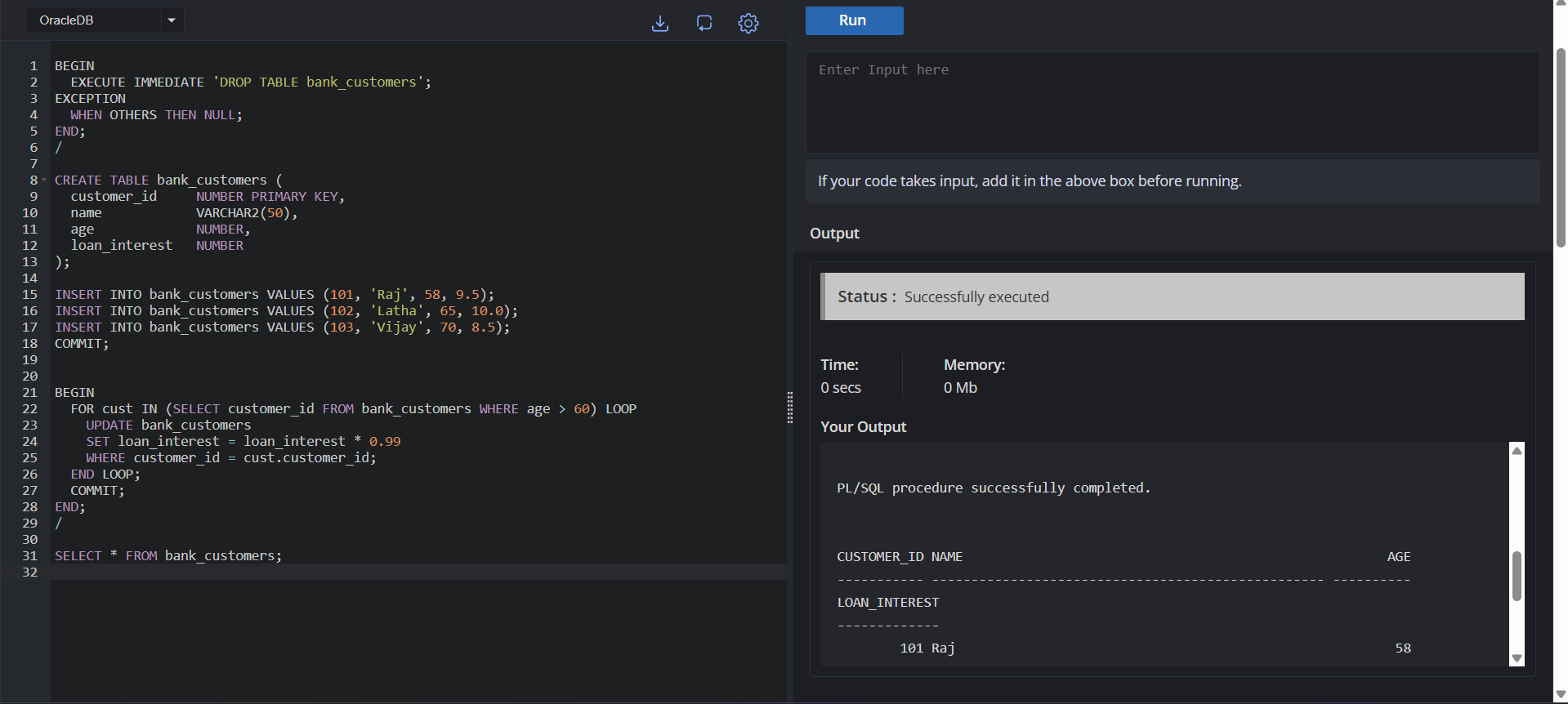
COMMIT;

END;

/

SELECT \* FROM bank\_customers;

**Output:**



**Scenario 2: Promote Customers to VIP if Balance > 10,000**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE accounts';

EXCEPTION

WHEN OTHERS THEN

NULL;

END;

/

CREATE TABLE accounts (

account\_no NUMBER PRIMARY KEY,

customer VARCHAR2(50),

balance NUMBER,

is\_vip VARCHAR2(3)

);

INSERT INTO accounts VALUES (201, 'Suresh', 8000, 'NO');

INSERT INTO accounts VALUES (202, 'Geetha', 15000, 'NO');

INSERT INTO accounts VALUES (203, 'Karthik', 12000, 'NO');

COMMIT;

BEGIN

FOR acc IN (SELECT account\_no FROM accounts WHERE balance > 10000) LOOP

UPDATE accounts

SET is\_vip = 'YES'

WHERE account\_no = acc.account\_no;

END LOOP;

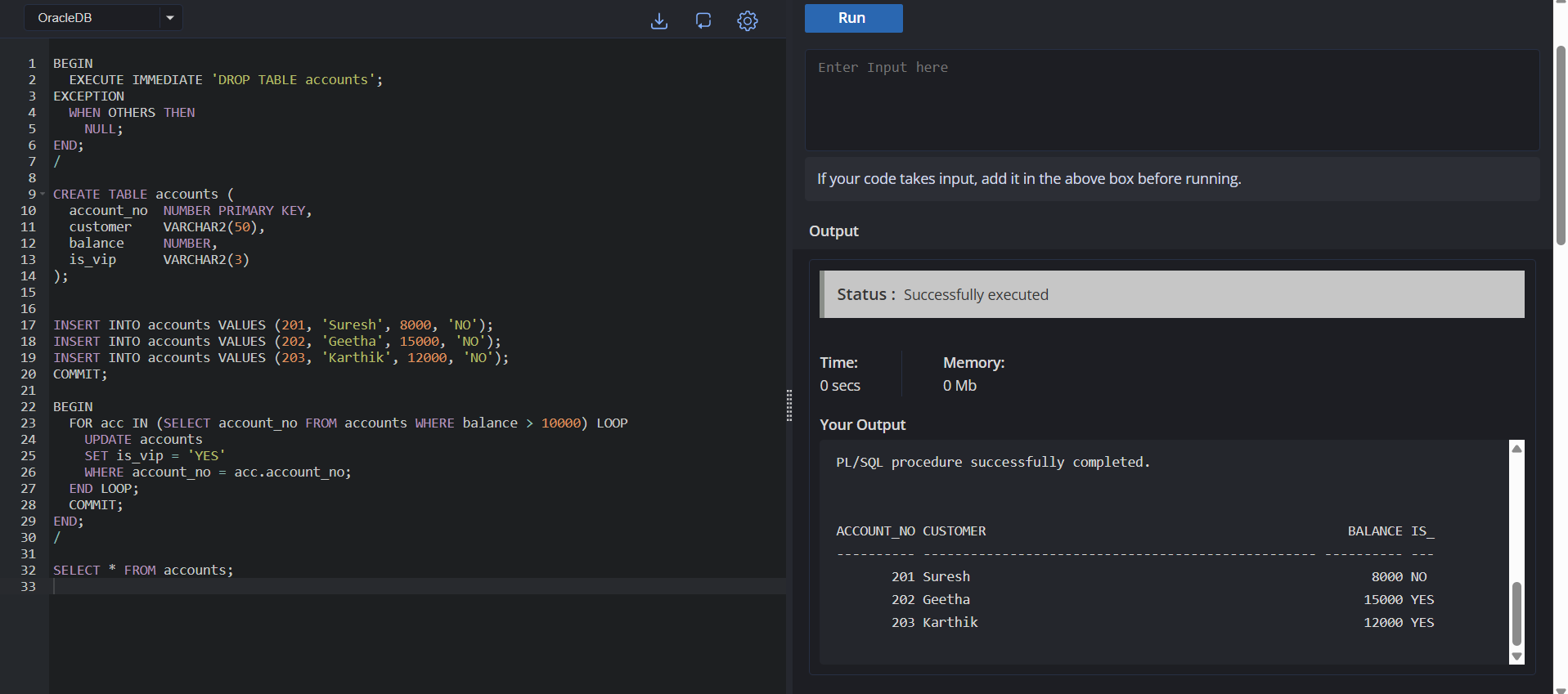
COMMIT;

END;

/

SELECT \* FROM accounts;

**Output:**



**Scenario 3: Reminders for Loans Due in Next 30 Days**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE loan\_details';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE loan\_details (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

due\_date DATE

);

INSERT INTO loan\_details VALUES (301, 101, SYSDATE + 10);

INSERT INTO loan\_details VALUES (302, 102, SYSDATE + 40);

INSERT INTO loan\_details VALUES (303, 103, SYSDATE + 25);

COMMIT;

SET SERVEROUTPUT ON;

BEGIN

FOR loan IN (

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

FROM loan\_details

WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

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

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

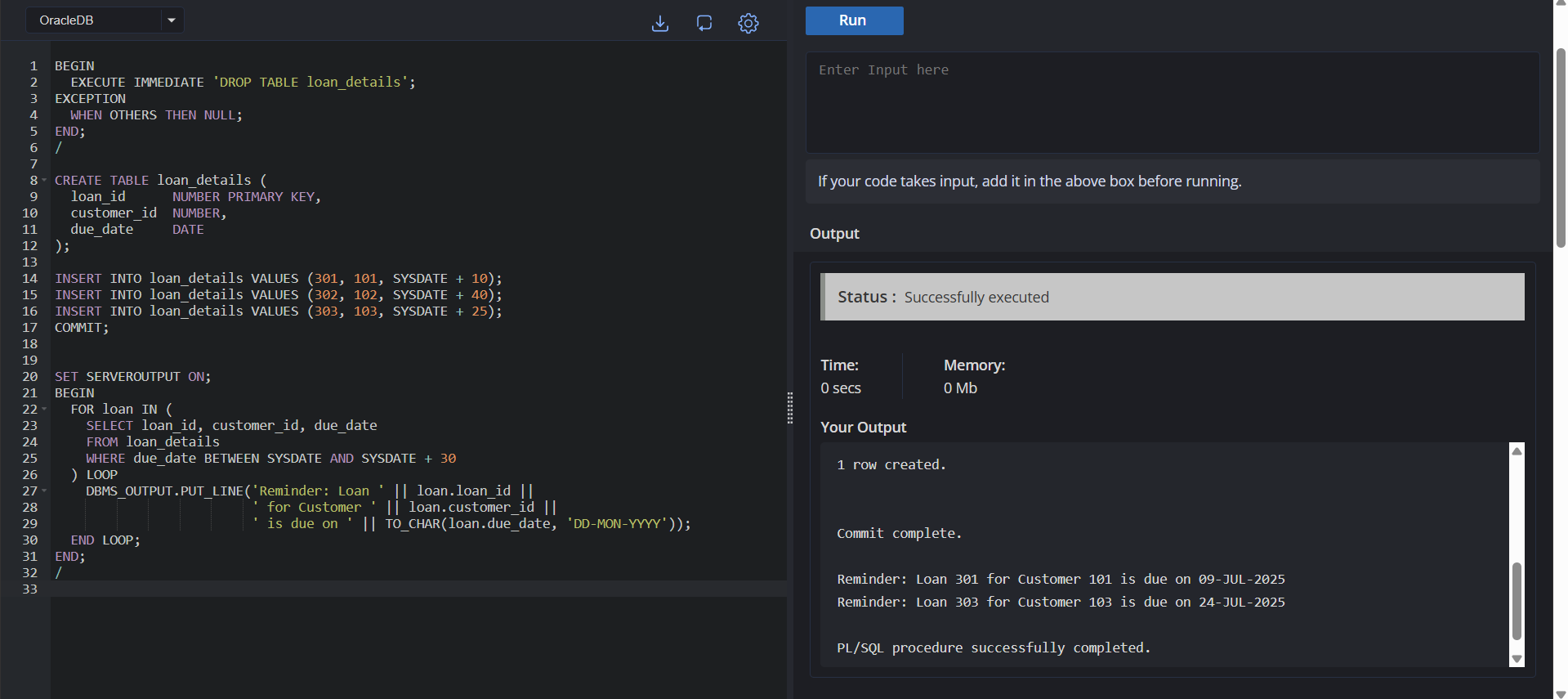
' is due on ' || TO\_CHAR(loan.due\_date, 'DD-MON-YYYY'));

END LOOP;

END;

/

**Output:**



**Exercise 3: Stored Procedures**

**Scenario 1: ProcessMonthlyInterest – Apply 1% to all savings accounts**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE savings\_accounts';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE savings\_accounts (

account\_id NUMBER PRIMARY KEY,

holder\_name VARCHAR2(50),

balance NUMBER

);

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

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

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE savings\_accounts

SET balance = balance + (balance \* 0.01);

COMMIT;

END;

/

BEGIN

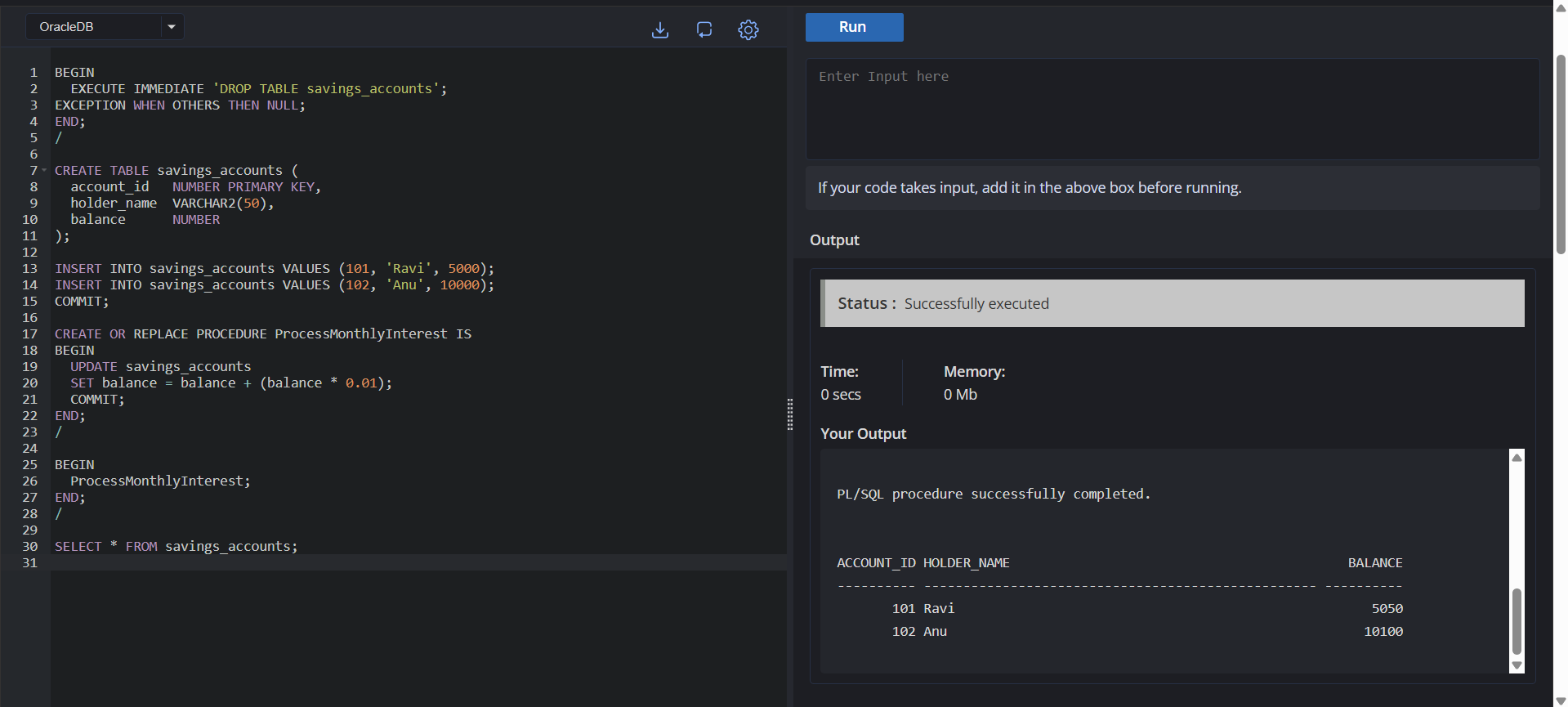
ProcessMonthlyInterest;

END;

/

SELECT \* FROM savings\_accounts;

**Output:**



**Scenario 2: UpdateEmployeeBonus – Add bonus % to all employees in a department**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE employees';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE employees (

emp\_id NUMBER PRIMARY KEY,

emp\_name VARCHAR2(50),

dept\_id NUMBER,

salary NUMBER

);

INSERT INTO employees VALUES (1, 'Arun', 10, 30000);

INSERT INTO employees VALUES (2, 'Deepa', 20, 40000);

INSERT INTO employees VALUES (3, 'Kiran', 10, 35000);

COMMIT;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept NUMBER,

bonus\_percent NUMBER

) IS

BEGIN

UPDATE employees

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

WHERE dept\_id = dept;

COMMIT;

END;

/

BEGIN

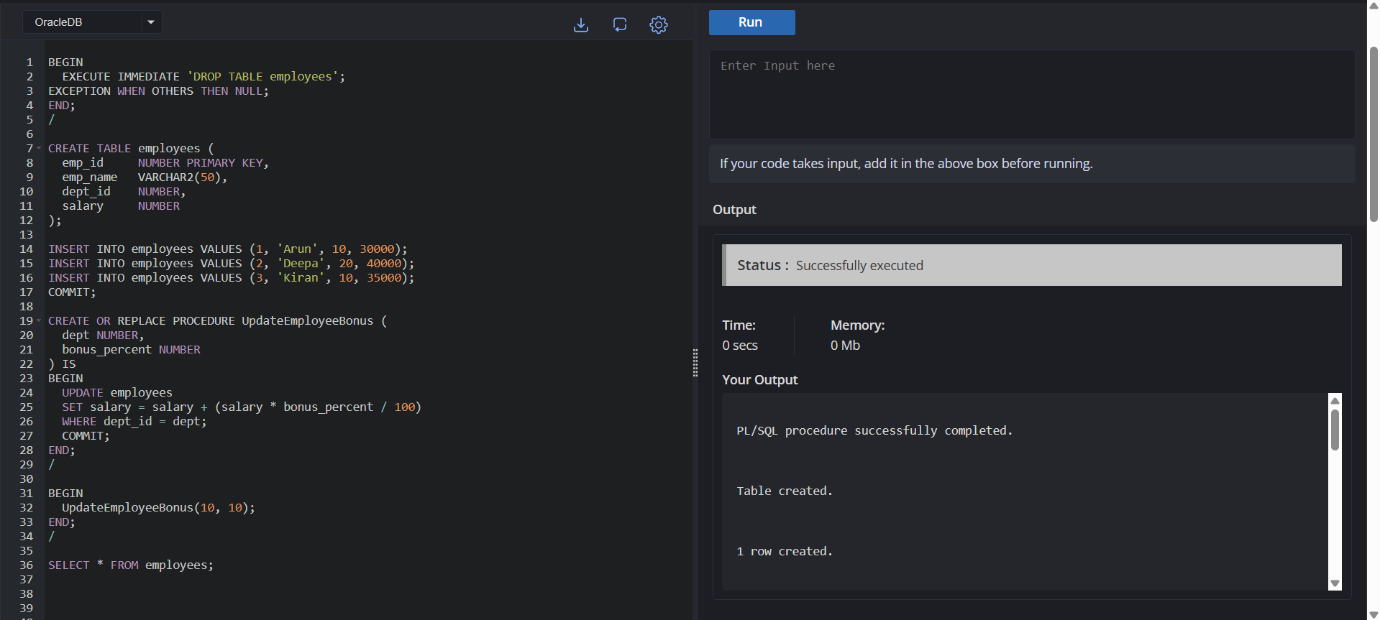
UpdateEmployeeBonus(10, 10);

END;

/

SELECT \* FROM employees;

**Output:**



**Scenario 3: TransferFunds – Transfer between accounts with balance check**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE accounts';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE accounts (

acc\_no NUMBER PRIMARY KEY,

name VARCHAR2(50),

balance NUMBER

);

INSERT INTO accounts VALUES (111, 'Ramesh', 8000);

INSERT INTO accounts VALUES (222, 'Sita', 3000);

COMMIT;

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_acc NUMBER,

to\_acc NUMBER,

amount NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT balance INTO v\_balance FROM accounts WHERE acc\_no = from\_acc;

IF v\_balance >= amount THEN

UPDATE accounts SET balance = balance - amount WHERE acc\_no = from\_acc;

UPDATE accounts SET balance = balance + amount WHERE acc\_no = to\_acc;

COMMIT;

ELSE

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

END IF;

END;

/

BEGIN

TransferFunds(111, 222, 1500);

END;

/

SELECT \* FROM accounts;

**Output:**

