**Exercise 1: Control Structures**

**Implementation:**

**CODE:**

**=> To create table & insert data:**

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

age NUMBER,

balance NUMBER,

IsVIP VARCHAR2(5) DEFAULT 'FALSE'

);

CREATE TABLE loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

interest\_rate NUMBER,

due\_date DATE,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

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

INSERT INTO customers VALUES (2, 'Bob', 45, 8000, 'FALSE');

INSERT INTO customers VALUES (3, 'Charlie', 70, 5000, 'FALSE');

INSERT INTO customers VALUES (4, 'Diana', 35, 12000, 'FALSE');

INSERT INTO loans VALUES (101, 1, 5.5, SYSDATE + 10);

INSERT INTO loans VALUES (102, 2, 6.0, SYSDATE + 40);

INSERT INTO loans VALUES (103, 3, 7.0, SYSDATE + 20);

INSERT INTO loans VALUES (104, 4, 4.5, SYSDATE + 5);

COMMIT;

=> **To view table:**

SELECT \* FROM customers;

SELECT \* FROM loans;

=> **Scenario 1: To apply discount for customers above 60 years old:**

BEGIN

FOR rec IN (SELECT \* FROM customers) LOOP

IF rec.age > 60 THEN

UPDATE loans

SET interest\_rate = interest\_rate - 1

WHERE customer\_id = rec.customer\_id;

DBMS\_OUTPUT.PUT\_LINE('Applied 1% discount for ' || rec.name);

END IF;

END LOOP;

END;

=> **Scenario 2: To promote to VIPS for customers with balance over 10000:**

BEGIN

FOR rec IN (SELECT \* FROM customers) LOOP

IF rec.balance > 10000 THEN

UPDATE customers

SET IsVIP = 'TRUE'

WHERE customer\_id = rec.customer\_id;

DBMS\_OUTPUT.PUT\_LINE('Promoted ' || rec.name || ' to VIP!');

END IF;

END LOOP;

END;

=> **Scenario 3: To send remainders to customers whose loan due is within next 30 days :**

BEGIN

FOR rec IN (

SELECT c.name, l.due\_date

FROM loans l

JOIN customers c ON l.customer\_id = c.customer\_id

WHERE l.due\_date BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

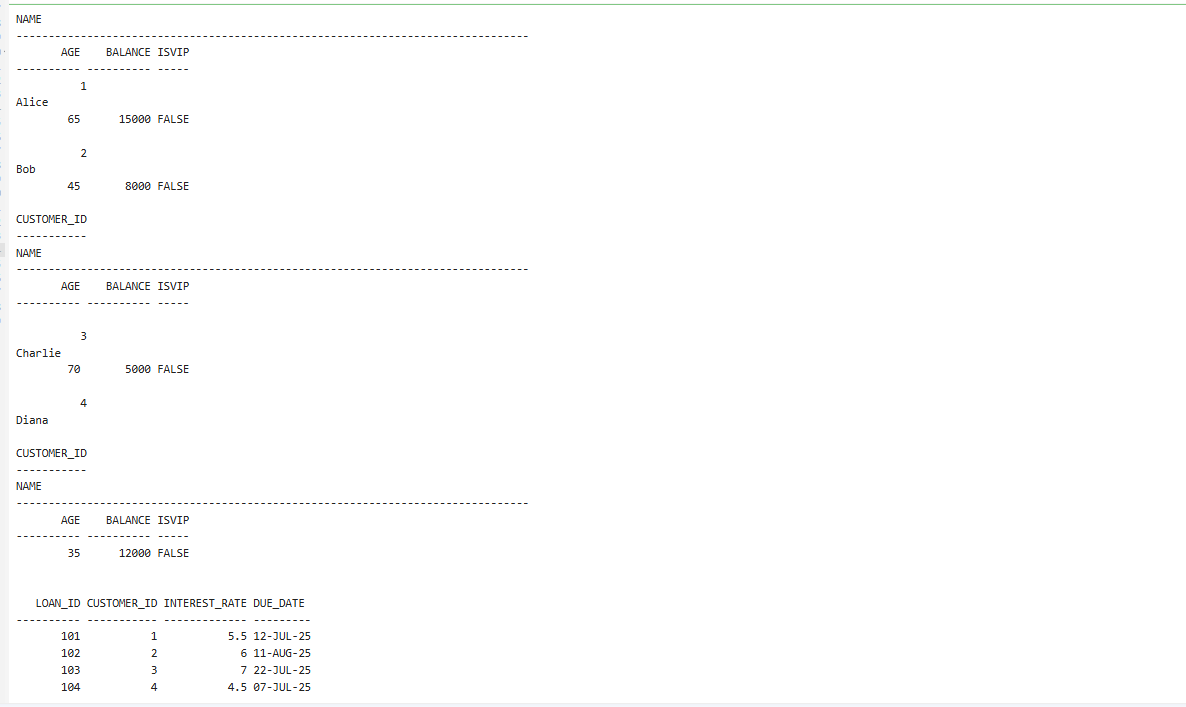
DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || rec.name || ', your loan is due on ' || TO\_CHAR(rec.due\_date, 'DD-MON-YYYY'));

END LOOP;

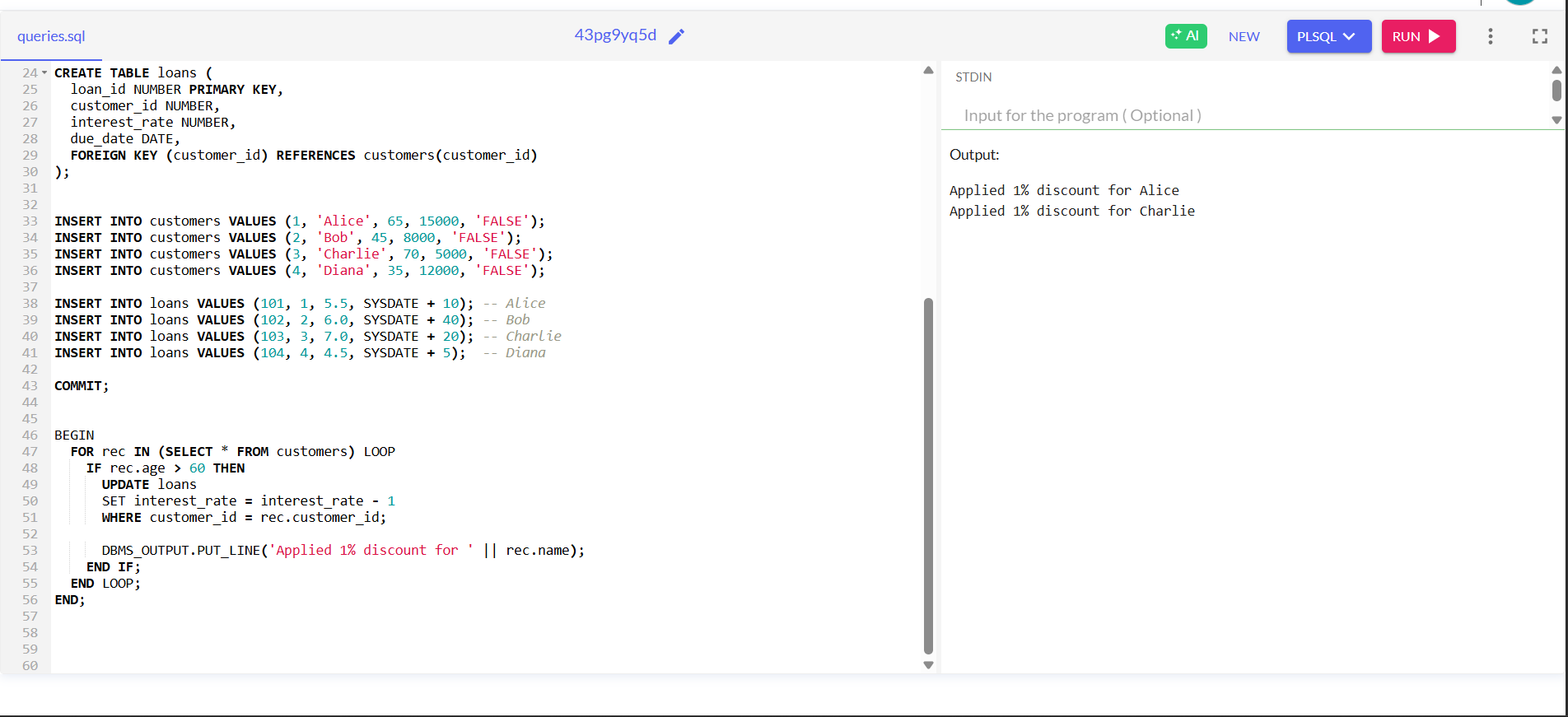
END;

**OUTPUT:**

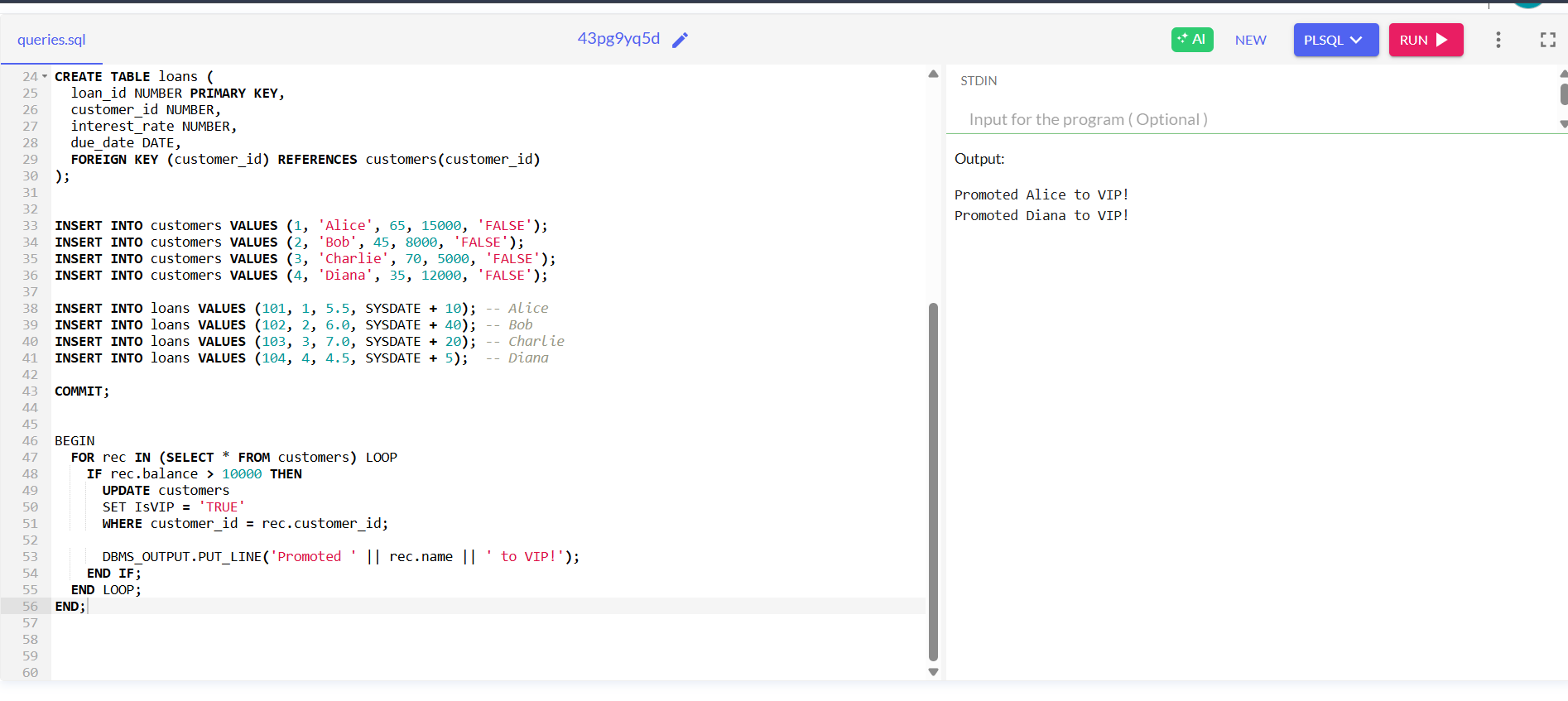
**Table:**



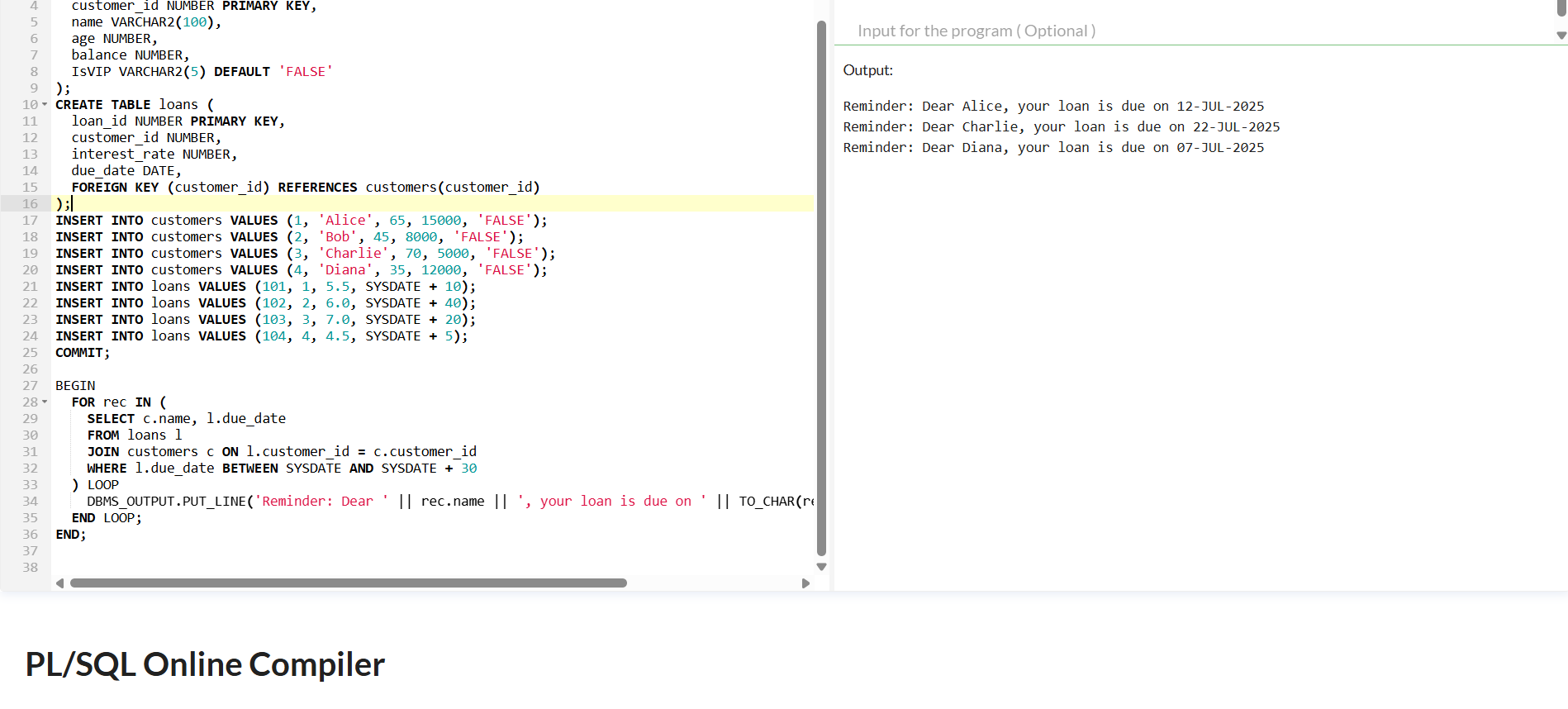
**Scenario 1:**

****

**Scenario 2:**

****

**Scenario 3:**

****

**Exercise 3: Stored Procedures**

**Implementation:**

**CODE:**

**=> To create table and insert data:**

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

balance NUMBER,

account\_type VARCHAR2(20)

);

CREATE TABLE employees (

emp\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

department\_id NUMBER,

salary NUMBER

);

INSERT INTO accounts VALUES (1001, 1, 5000, 'SAVINGS');

INSERT INTO accounts VALUES (1002, 1, 3000, 'CHECKING');

INSERT INTO accounts VALUES (1003, 2, 8000, 'SAVINGS');

INSERT INTO accounts VALUES (1004, 2, 2000, 'SAVINGS');

INSERT INTO employees VALUES (1, 'Alice', 101, 5000);

INSERT INTO employees VALUES (2, 'Bob', 101, 5500);

INSERT INTO employees VALUES (3, 'Charlie', 102, 6000);

COMMIT;

=> **To view table:**

select \* from accounts;

select \* from employees;

=> **Scenario 1: ProcessMonthlyInterest**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE accounts

SET balance = balance \* 1.01

WHERE account\_type = 'SAVINGS';

COMMIT;

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Before Interest:');

FOR r IN (SELECT \* FROM accounts WHERE account\_type = 'SAVINGS') LOOP

DBMS\_OUTPUT.PUT\_LINE('Account ' || r.account\_id || ': ' || r.balance);

END LOOP;

ProcessMonthlyInterest;

DBMS\_OUTPUT.PUT\_LINE('After Interest:');

FOR r IN (SELECT \* FROM accounts WHERE account\_type = 'SAVINGS') LOOP

DBMS\_OUTPUT.PUT\_LINE('Account ' || r.account\_id || ': ' || r.balance);

END LOOP;

END;

/

=> **Scenario 2: UpdateEmployeeBonus**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_dept\_id IN NUMBER,

p\_bonus\_pct IN NUMBER

) IS

BEGIN

UPDATE employees

SET salary = salary + (salary \* p\_bonus\_pct / 100)

WHERE department\_id = p\_dept\_id;

COMMIT;

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Before Bonus:');

FOR r IN (SELECT \* FROM employees WHERE department\_id = 101) LOOP

DBMS\_OUTPUT.PUT\_LINE(r.name || ': ' || r.salary);

END LOOP;

UpdateEmployeeBonus(101, 10);

DBMS\_OUTPUT.PUT\_LINE('After Bonus:');

FOR r IN (SELECT \* FROM employees WHERE department\_id = 101) LOOP

DBMS\_OUTPUT.PUT\_LINE(r.name || ': ' || r.salary);

END LOOP;

END;

/

=> **Scenario 3: TransferFunds**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT balance INTO v\_balance

FROM accounts

WHERE account\_id = p\_from\_account

FOR UPDATE;

IF v\_balance < p\_amount THEN

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

END IF;

UPDATE accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_from\_account;

UPDATE accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_to\_account;

COMMIT;

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Before Transfer:');

FOR r IN (SELECT \* FROM accounts WHERE account\_id IN (1001, 1002)) LOOP

DBMS\_OUTPUT.PUT\_LINE('Account ' || r.account\_id || ': ' || r.balance);

END LOOP;

-- Call procedure

TransferFunds(1001, 1002, 1000); -- Transfer $1000 from 1001 to 1002

DBMS\_OUTPUT.PUT\_LINE('After Transfer:');

FOR r IN (SELECT \* FROM accounts WHERE account\_id IN (1001, 1002)) LOOP

DBMS\_OUTPUT.PUT\_LINE('Account ' || r.account\_id || ': ' || r.balance);

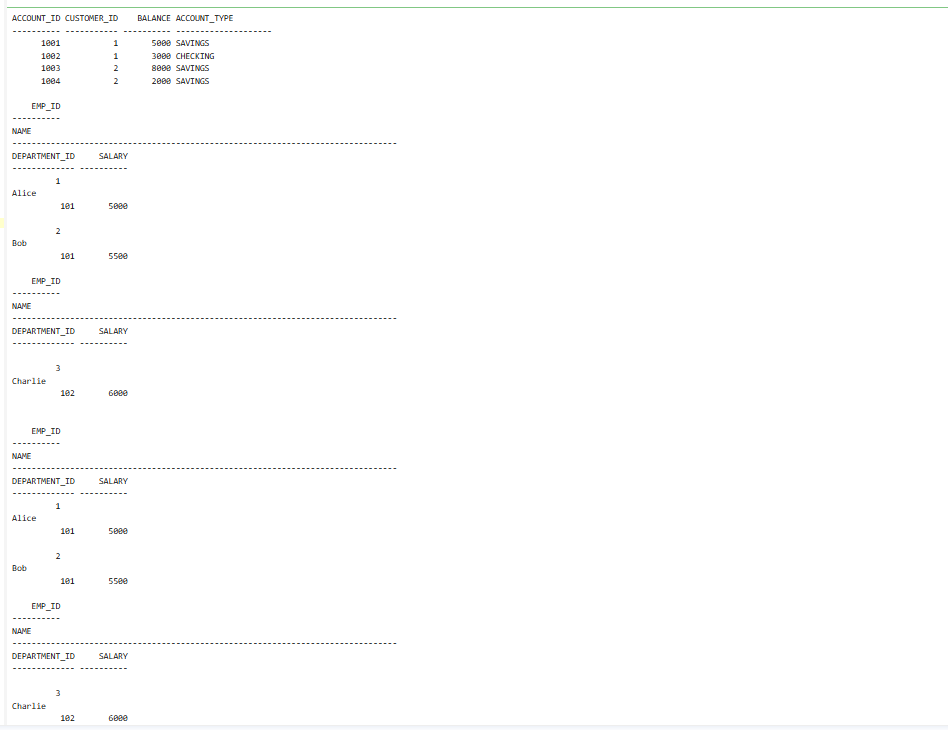
END LOOP;

END;

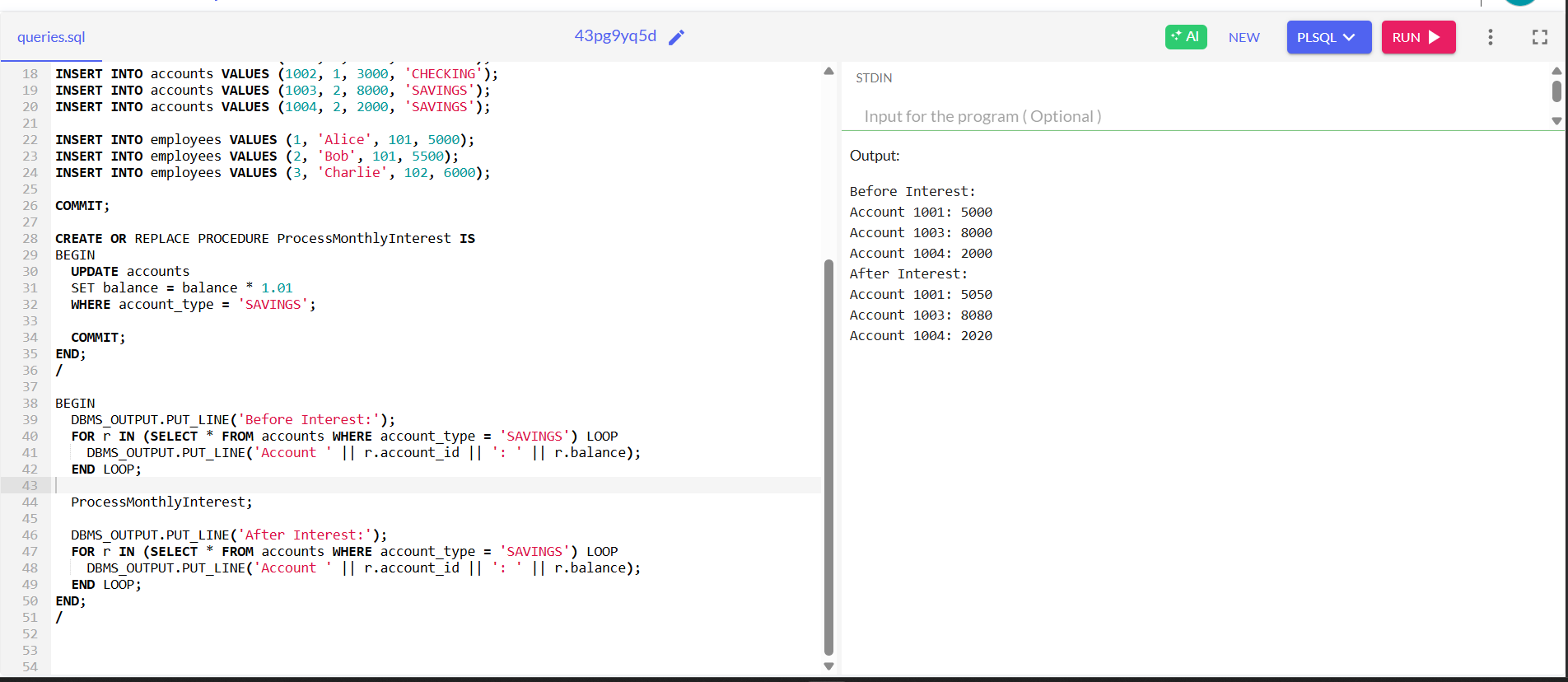
/

**OUTPUT:**

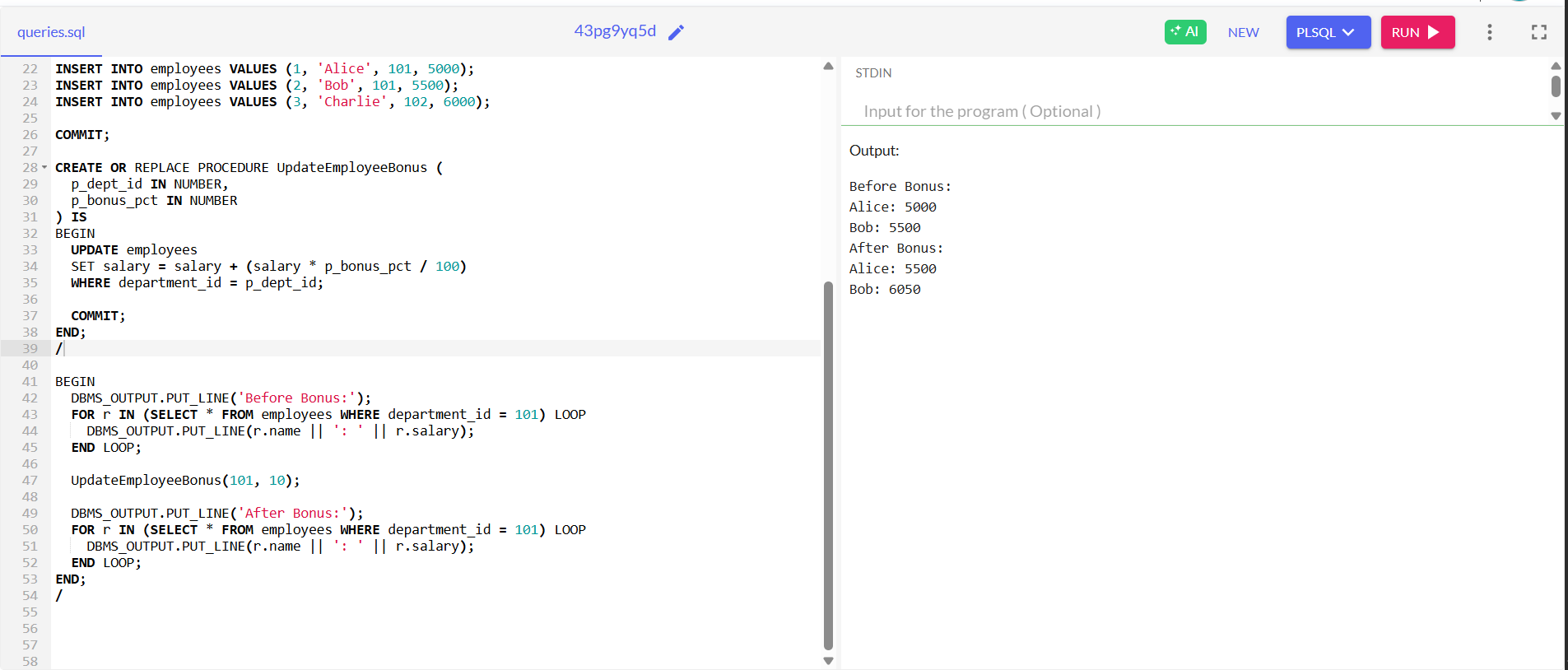
**=> Table:**

****

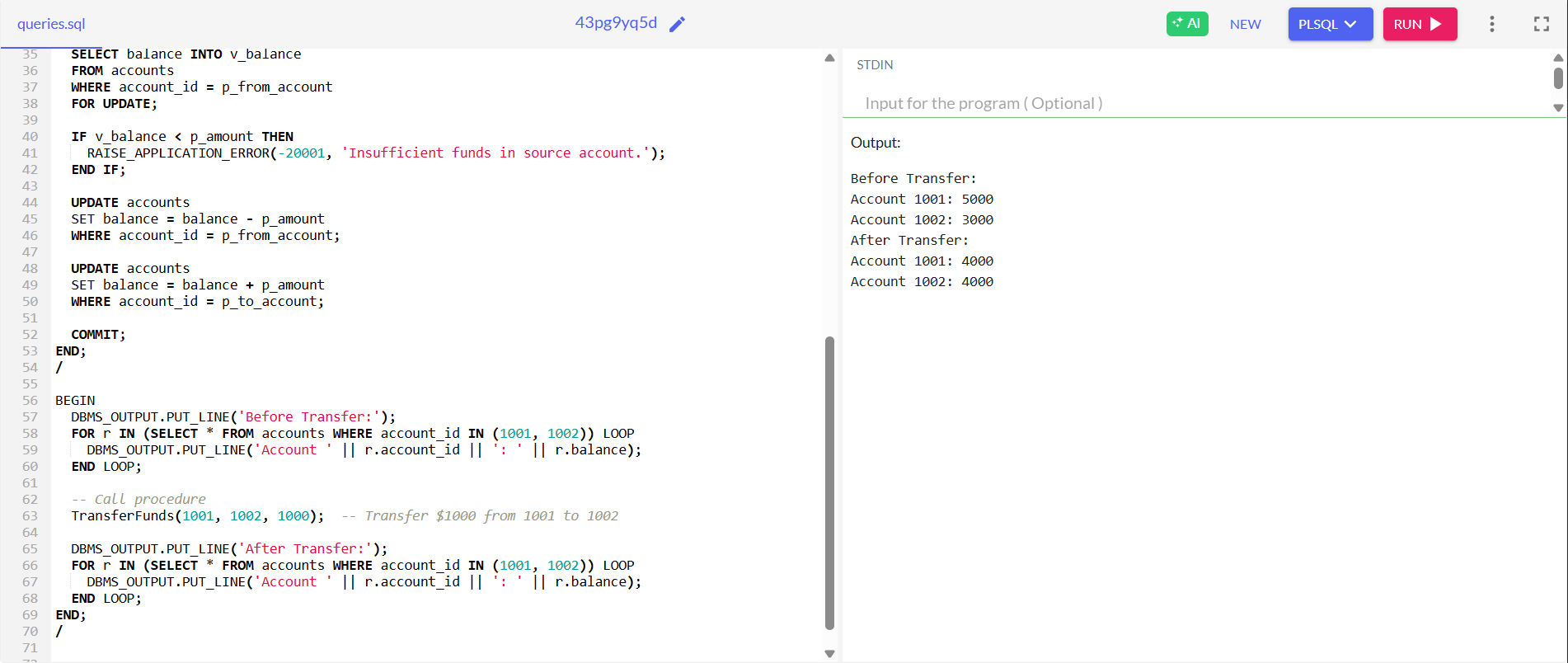
**Scenario 1:**

****

**Scenario 2:**

****

**Scenario 3:**

****