**Week 2- PL SQL Programming- Hands-On Exercises**

**Exercise 1: Control Structures**

**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

**Query:**

DECLARE

TYPE customer\_record IS RECORD (

customer\_id NUMBER,

name VARCHAR2(100),

age NUMBER,

loan\_interest\_rate NUMBER

);

TYPE customer\_table IS TABLE OF customer\_record;

customers customer\_table;

new\_customer\_id NUMBER;

new\_name VARCHAR2(100);

new\_age NUMBER;

new\_loan\_interest\_rate NUMBER;

BEGIN

customers := customer\_table(

customer\_record(1, 'Hema Srivalli', 65, 5.00),

customer\_record(2, 'Pratyusha', 58, 4.50),

customer\_record(3, 'Jyothsna', 72, 6.00),

customer\_record(4, 'Siddhi', 45, 3.75),

customer\_record(5, 'Dolly', 62, 5.25)

);

FOR i IN 1 .. customers.COUNT LOOP

new\_customer\_id := customers(i).customer\_id;

new\_name := customers(i).name;

new\_age := customers(i).age;

new\_loan\_interest\_rate := customers(i).loan\_interest\_rate;

IF new\_age > 60 THEN

new\_loan\_interest\_rate := new\_loan\_interest\_rate \* 0.99;

DBMS\_OUTPUT.PUT\_LINE('Updated customer ID: ' || new\_customer\_id ||

', Name: ' || new\_name ||

', Age: ' || new\_age ||

' - New interest rate: ' || new\_loan\_interest\_rate);

END IF;

END LOOP;

EXCEPTION

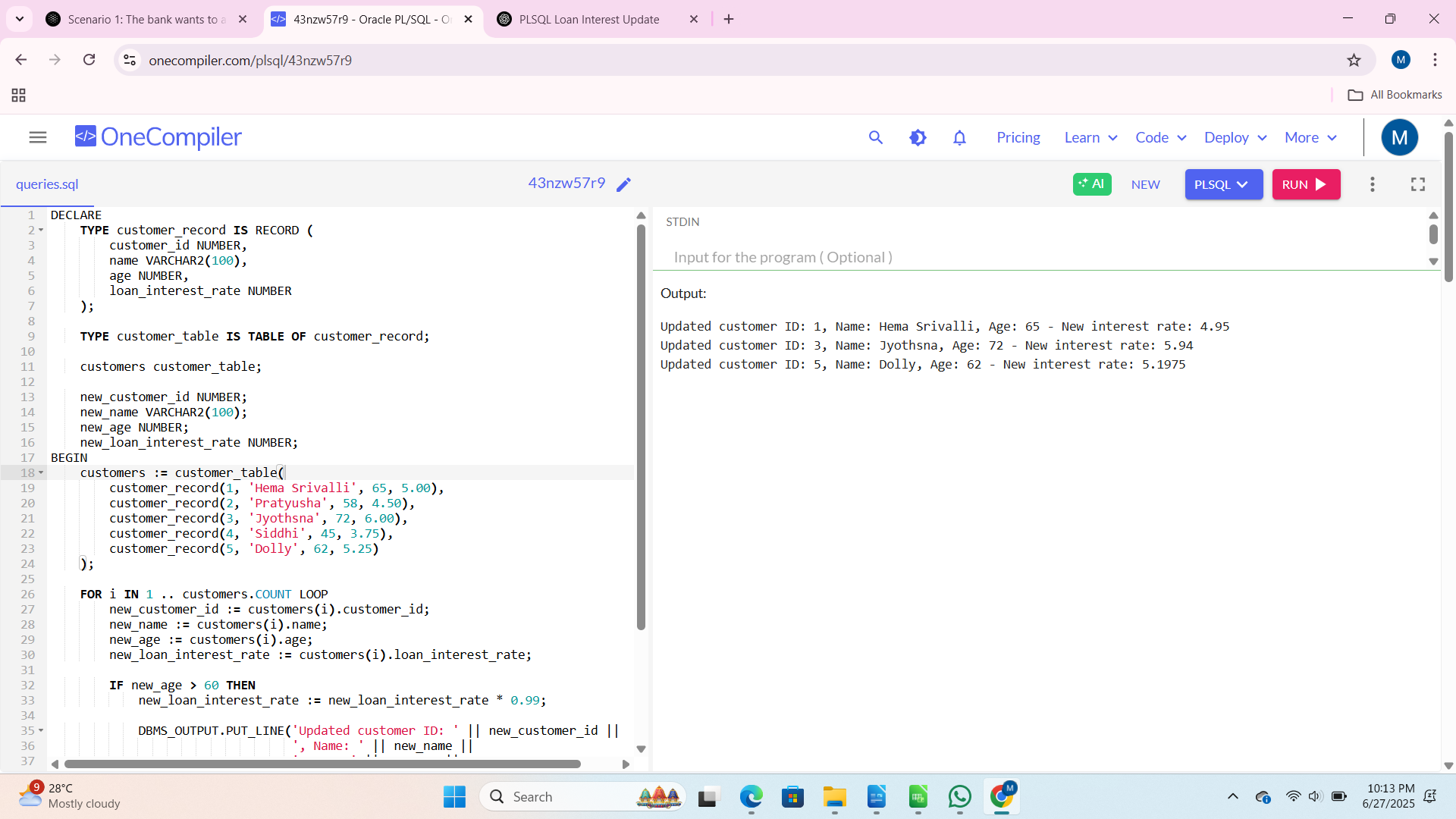
WHEN OTHERS THEN

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

END;

/

**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.

**Query:**

DECLARE

TYPE customer\_type IS RECORD (

customer\_id NUMBER,

name VARCHAR2(100),

balance NUMBER,

is\_vip BOOLEAN

);

TYPE customer\_table IS TABLE OF customer\_type;

customers customer\_table := customer\_table(

customer\_type(1, 'Siddhi', 7500, FALSE),

customer\_type(2, 'Pavan', 12000, FALSE),

customer\_type(3, 'Pratyusha', 9000, FALSE),

customer\_type(4, 'Jyothsna', 25000, FALSE),

customer\_type(5, 'Dolly', 8000, FALSE)

);

vip\_threshold CONSTANT NUMBER := 10000;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('VIP Promotion Processing');

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

FOR i IN 1..customers.COUNT LOOP

IF customers(i).balance > vip\_threshold THEN

customers(i).is\_vip := TRUE;

DBMS\_OUTPUT.PUT\_LINE('Promoted to VIP: ' || customers(i).name ||

' (Balance: $' || customers(i).balance || ')');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Not qualified: ' || customers(i).name ||

' (Balance: $' || customers(i).balance || ')');

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(CHR(10) || 'Final VIP Status:');

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

FOR i IN 1..customers.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(

RPAD(customers(i).name, 15) || ' | ' ||

RPAD('$' || customers(i).balance, 10) || ' | ' ||

CASE WHEN customers(i).is\_vip THEN 'VIP' ELSE 'Regular' END

);

END LOOP;

EXCEPTION

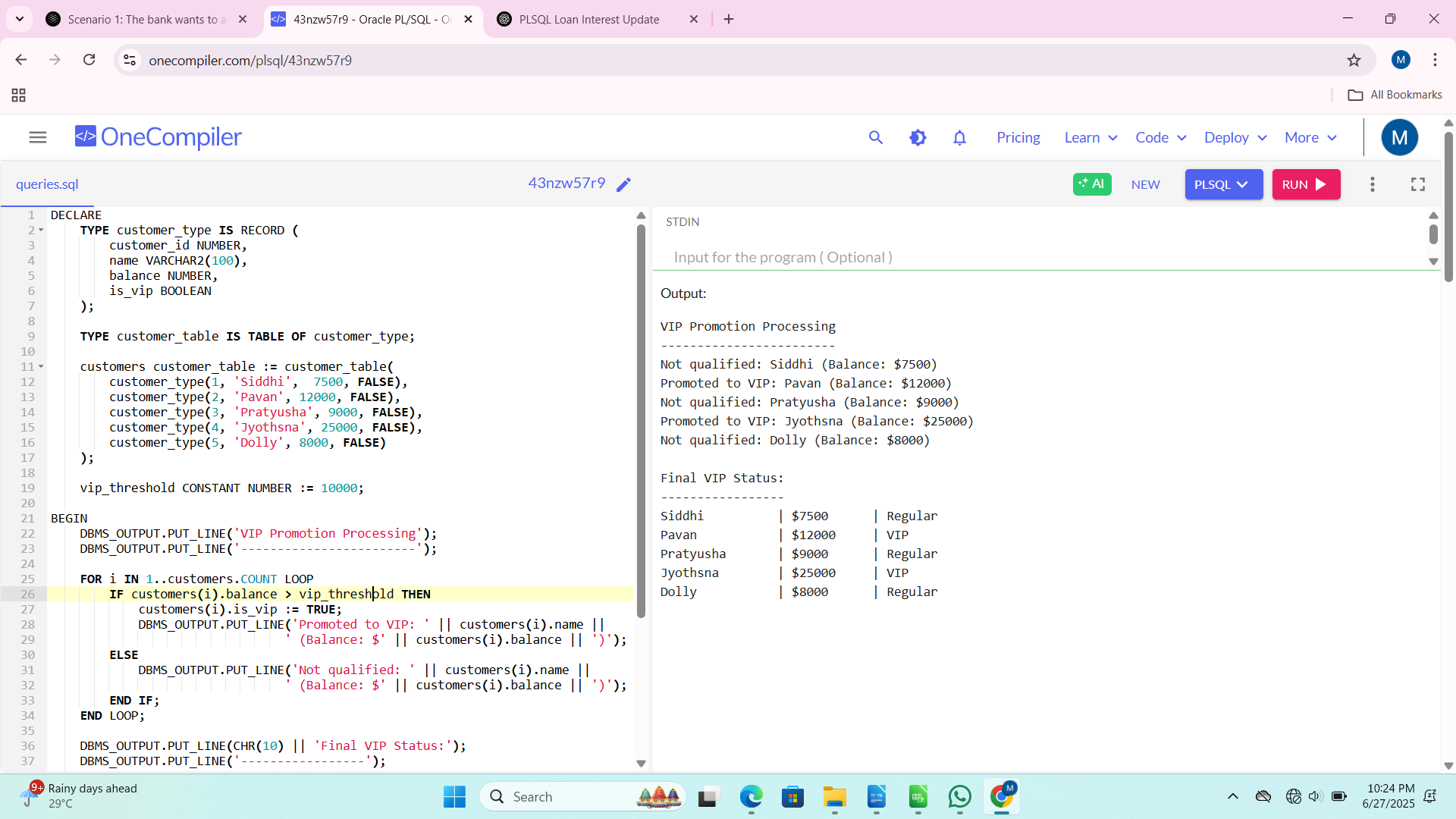
WHEN OTHERS THEN

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

END;

/

**Output:**

 **Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

**Query:**

DECLARE

TYPE loan\_type IS RECORD (

loan\_id NUMBER,

customer\_name VARCHAR2(100),

due\_date DATE,

amount NUMBER,

is\_reminder\_sent BOOLEAN

);

TYPE loan\_table IS TABLE OF loan\_type;

loans loan\_table := loan\_table(

loan\_type(101, 'Anuhya', SYSDATE + 15, 5000, FALSE),

loan\_type(102, 'Anitha', SYSDATE + 60, 7500, FALSE),

loan\_type(103, 'Usha', SYSDATE + 5, 12000, FALSE),

loan\_type(104, 'Akshitha', SYSDATE + 28, 8000, FALSE),

loan\_type(105, 'Satyavathi', SYSDATE + 31, 3000, FALSE)

);

c\_reminder\_days CONSTANT NUMBER := 30;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('LOAN REMINDER PROCESSING');

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

FOR i IN 1..loans.COUNT LOOP

IF loans(i).due\_date <= SYSDATE + c\_reminder\_days THEN

loans(i).is\_reminder\_sent := TRUE;

DBMS\_OUTPUT.PUT\_LINE('REMINDER: ' || loans(i).customer\_name ||

CHR(10) || 'Loan ID: ' || loans(i).loan\_id ||

CHR(10) || 'Amount Due: $' || loans(i).amount ||

CHR(10) || 'Due Date: ' || TO\_CHAR(loans(i).due\_date, 'DD-MON-YYYY') ||

CHR(10) || 'Days Remaining: ' || (loans(i).due\_date - SYSDATE) ||

CHR(10) || '-----------------------------------');

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(CHR(10) || 'REMINDER SUMMARY');

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

FOR i IN 1..loans.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(

RPAD(loans(i).customer\_name, 20) || ' | ' ||

RPAD('#' || loans(i).loan\_id, 10) || ' | ' ||

RPAD(TO\_CHAR(loans(i).due\_date, 'DD-MON'), 10) || ' | ' ||

CASE WHEN loans(i).is\_reminder\_sent THEN 'REMINDER SENT' ELSE 'Not due soon' END

);

END LOOP;

EXCEPTION

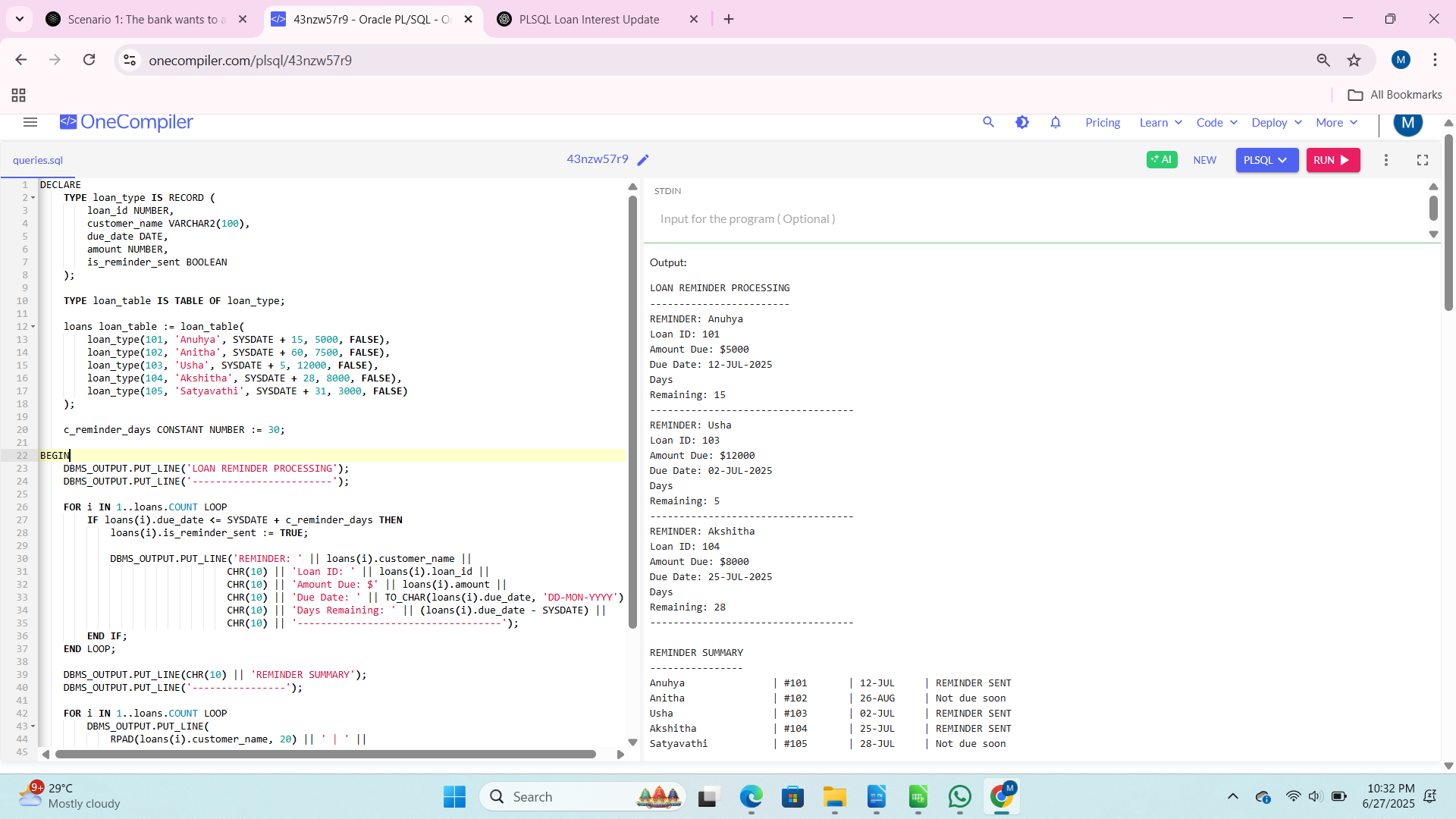
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error processing reminders: ' || SQLERRM);

END;

/

**Output:**

 **Exercise 3: Stored Procedures**

**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

**Query:**

DECLARE

TYPE AccountRecord IS RECORD (

account\_id NUMBER,

account\_holder VARCHAR2(100),

balance NUMBER(10, 2),

last\_interest\_date DATE

);

TYPE AccountTable IS TABLE OF AccountRecord INDEX BY PLS\_INTEGER;

accounts AccountTable;

PROCEDURE ProcessMonthlyInterest IS

v\_interest\_rate CONSTANT NUMBER := 0.01;

BEGIN

FOR i IN 1 .. accounts.COUNT LOOP

accounts(i).balance := accounts(i).balance \* (1 + v\_interest\_rate);

accounts(i).last\_interest\_date := SYSDATE;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Applied 1% interest to all savings accounts.');

END ProcessMonthlyInterest;

BEGIN

accounts(1) := AccountRecord(1, 'Siddhi, 5000.00, NULL);

accounts(2) := AccountRecord(2, 'Hema', 10000.00, NULL);

accounts(3) := AccountRecord(3, 'Pavan', 7500.00, NULL);

DBMS\_OUTPUT.PUT\_LINE('=== Initial Balances ===');

FOR i IN 1 .. accounts.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(

RPAD(accounts(i).account\_holder, 15) || ': $' ||

TO\_CHAR(accounts(i).balance, '999,990.00')

);

END LOOP;

ProcessMonthlyInterest();

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

FOR i IN 1 .. accounts.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(

RPAD(accounts(i).account\_holder, 15) || ': $' ||

TO\_CHAR(accounts(i).balance, '999,990.00')

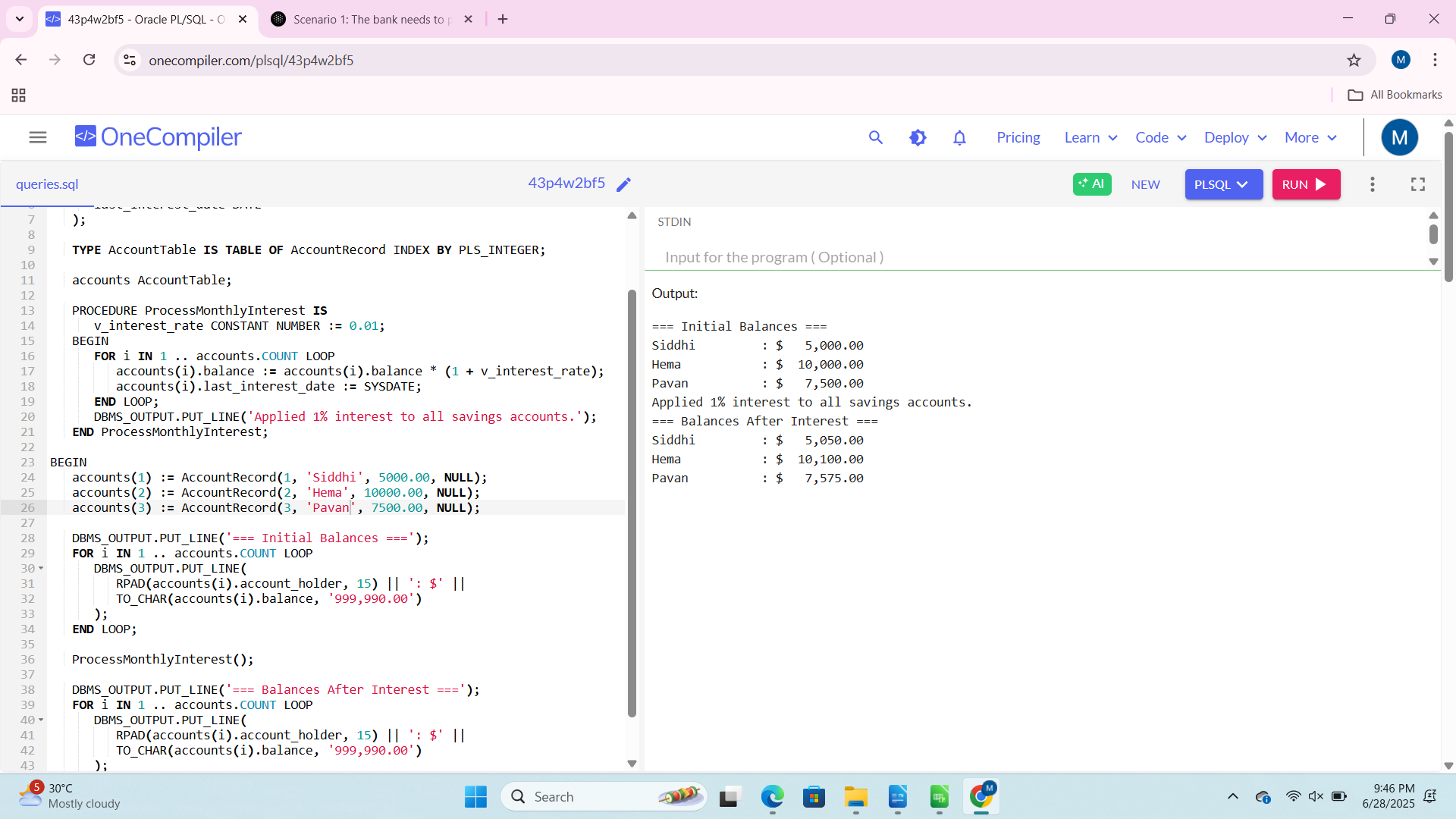
);

END LOOP;

END;

/

**Output:**

 **Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

**Query:**

DECLARE

TYPE EmployeeRecord IS RECORD (

employee\_id NUMBER,

employee\_name VARCHAR2(100),

department VARCHAR2(50),

salary NUMBER(10, 2),

bonus\_percentage NUMBER(5, 2),

updated\_salary NUMBER(10, 2)

);

TYPE EmployeeTable IS TABLE OF EmployeeRecord INDEX BY PLS\_INTEGER;

employees EmployeeTable;

PROCEDURE UpdateEmployeeBonus(

p\_dept VARCHAR2,

p\_bonus\_percent NUMBER

) IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Applying ' || p\_bonus\_percent || '% bonus to ' || p\_dept || ' department');

FOR i IN 1..employees.COUNT LOOP

IF employees(i).department = p\_dept THEN

employees(i).bonus\_percentage := p\_bonus\_percent;

employees(i).updated\_salary := employees(i).salary \* (1 + p\_bonus\_percent/100);

DBMS\_OUTPUT.PUT\_LINE(' - ' || employees(i).employee\_name ||

': ' || employees(i).salary || ' → ' ||

employees(i).updated\_salary);

END IF;

END LOOP;

END UpdateEmployeeBonus;

BEGIN

employees(1) := EmployeeRecord(101, 'Mohini', 'Sales', 50000, 0, 50000);

employees(2) := EmployeeRecord(102, 'Sanvi', 'Sales', 55000, 0, 55000);

employees(3) := EmployeeRecord(103, 'Siddhi', 'IT', 60000, 0, 60000);

employees(4) := EmployeeRecord(104, 'Varun', 'HR', 45000, 0, 45000);

employees(5) := EmployeeRecord(105, 'Dolly', 'IT', 65000, 0, 65000);

DBMS\_OUTPUT.PUT\_LINE('=== Initial Salaries ===');

FOR i IN 1..employees.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(

RPAD(employees(i).employee\_name, 15) ||

RPAD(' (' || employees(i).department || ')', 10) ||

': $' || TO\_CHAR(employees(i).salary, '999,990.00')

);

END LOOP;

UpdateEmployeeBonus('Sales', 10);

UpdateEmployeeBonus('IT', 5);

DBMS\_OUTPUT.PUT\_LINE('=== Updated Salaries ===');

FOR i IN 1..employees.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(

RPAD(employees(i).employee\_name, 15) ||

RPAD(' (' || employees(i).department || ')', 10) ||

': $' || TO\_CHAR(employees(i).updated\_salary, '999,990.00')

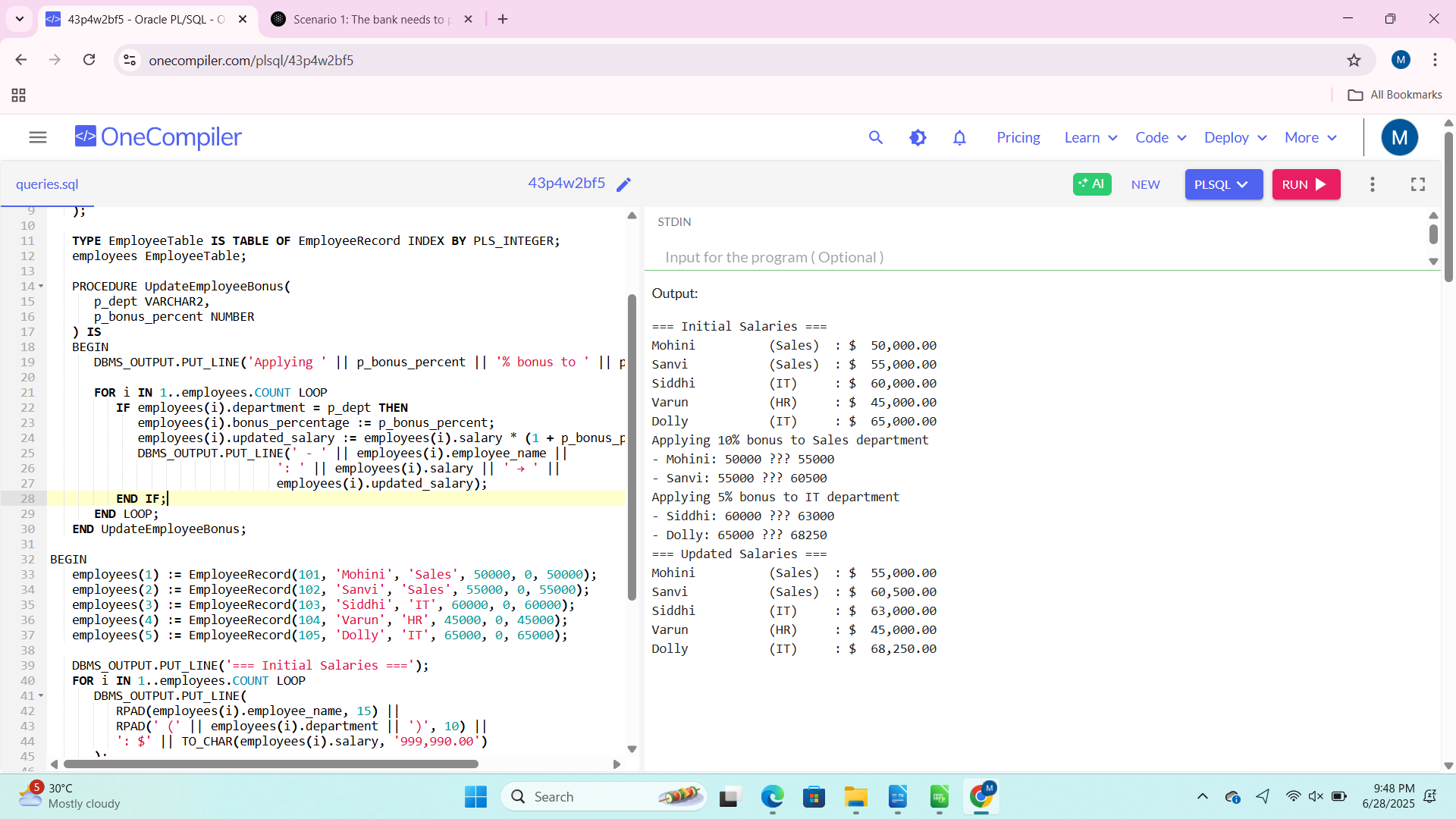
);

END LOOP;

END;

/

**Output:**

 **Scenario 3:** Customers should be able to transfer funds between their accounts.

**Query:**

DECLARE

TYPE AccountRecord IS RECORD (

account\_id NUMBER,

account\_holder VARCHAR2(100),

balance NUMBER(10, 2)

);

TYPE AccountTable IS TABLE OF AccountRecord INDEX BY PLS\_INTEGER;

accounts AccountTable;

PROCEDURE TransferFunds(

p\_from\_account\_id NUMBER,

p\_to\_account\_id NUMBER,

p\_amount NUMBER

) IS

v\_from\_balance NUMBER;

v\_to\_balance NUMBER;

v\_from\_account\_found BOOLEAN := FALSE;

v\_to\_account\_found BOOLEAN := FALSE;

BEGIN

FOR i IN 1 .. accounts.COUNT LOOP

IF accounts(i).account\_id = p\_from\_account\_id THEN

v\_from\_balance := accounts(i).balance;

v\_from\_account\_found := TRUE;

ELSIF accounts(i).account\_id = p\_to\_account\_id THEN

v\_to\_balance := accounts(i).balance;

v\_to\_account\_found := TRUE;

END IF;

END LOOP;

IF NOT v\_from\_account\_found THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Source account not found.');

RETURN;

ELSIF NOT v\_to\_account\_found THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Destination account not found.');

RETURN;

END IF;

IF v\_from\_balance < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient balance in source account.');

RETURN;

END IF;

FOR i IN 1 .. accounts.COUNT LOOP

IF accounts(i).account\_id = p\_from\_account\_id THEN

accounts(i).balance := accounts(i).balance - p\_amount;

ELSIF accounts(i).account\_id = p\_to\_account\_id THEN

accounts(i).balance := accounts(i).balance + p\_amount;

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful: $' || p\_amount ||

' transferred from account ' || p\_from\_account\_id ||

' to account ' || p\_to\_account\_id);

END TransferFunds;

BEGIN

accounts(1) := AccountRecord(101, 'Siddhi', 5000.00);

accounts(2) := AccountRecord(102, 'Mohith', 10000.00);

accounts(3) := AccountRecord(103, 'Pavan', 7500.00);

DBMS\_OUTPUT.PUT\_LINE('=== Initial Balances ===');

FOR i IN 1 .. accounts.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Account ID: ' || accounts(i).account\_id ||

', Holder: ' || accounts(i).account\_holder ||

', Balance: $' || TO\_CHAR(accounts(i).balance, '999,990.00')

);

END LOOP;

TransferFunds(101, 102, 1500);

TransferFunds(102, 103, 2000);

TransferFunds(101, 103, 6000);

DBMS\_OUTPUT.PUT\_LINE('=== Updated Balances ===');

FOR i IN 1 .. accounts.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Account ID: ' || accounts(i).account\_id ||

', Holder: ' || accounts(i).account\_holder ||

', Balance: $' || TO\_CHAR(accounts(i).balance, '999,990.00')

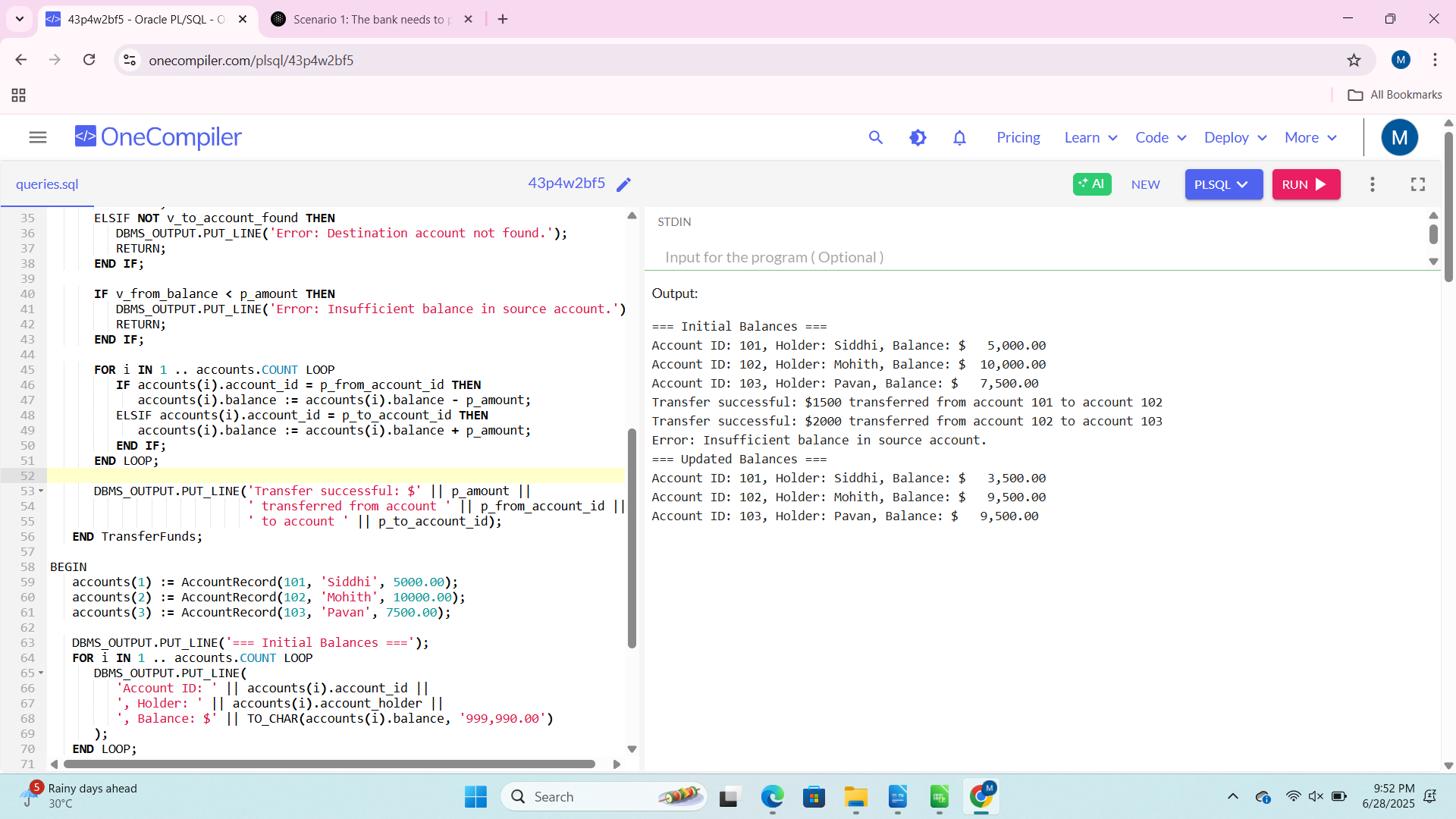
);

END LOOP;

END;

/

**Output:**

****