**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.

**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.

**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-**

**Scenario 1:**

BEGIN

FOR res IN (

SELECT c.name, c.customerID, l.loanID, l.interestRate

FROM loans l

JOIN customers c ON l.customerID = c.customerID

WHERE TRUNC(MONTHS\_BETWEEN(SYSDATE, c.dob) / 12) > 60

) LOOP

UPDATE loans

SET interestRate = res.interestRate - 1

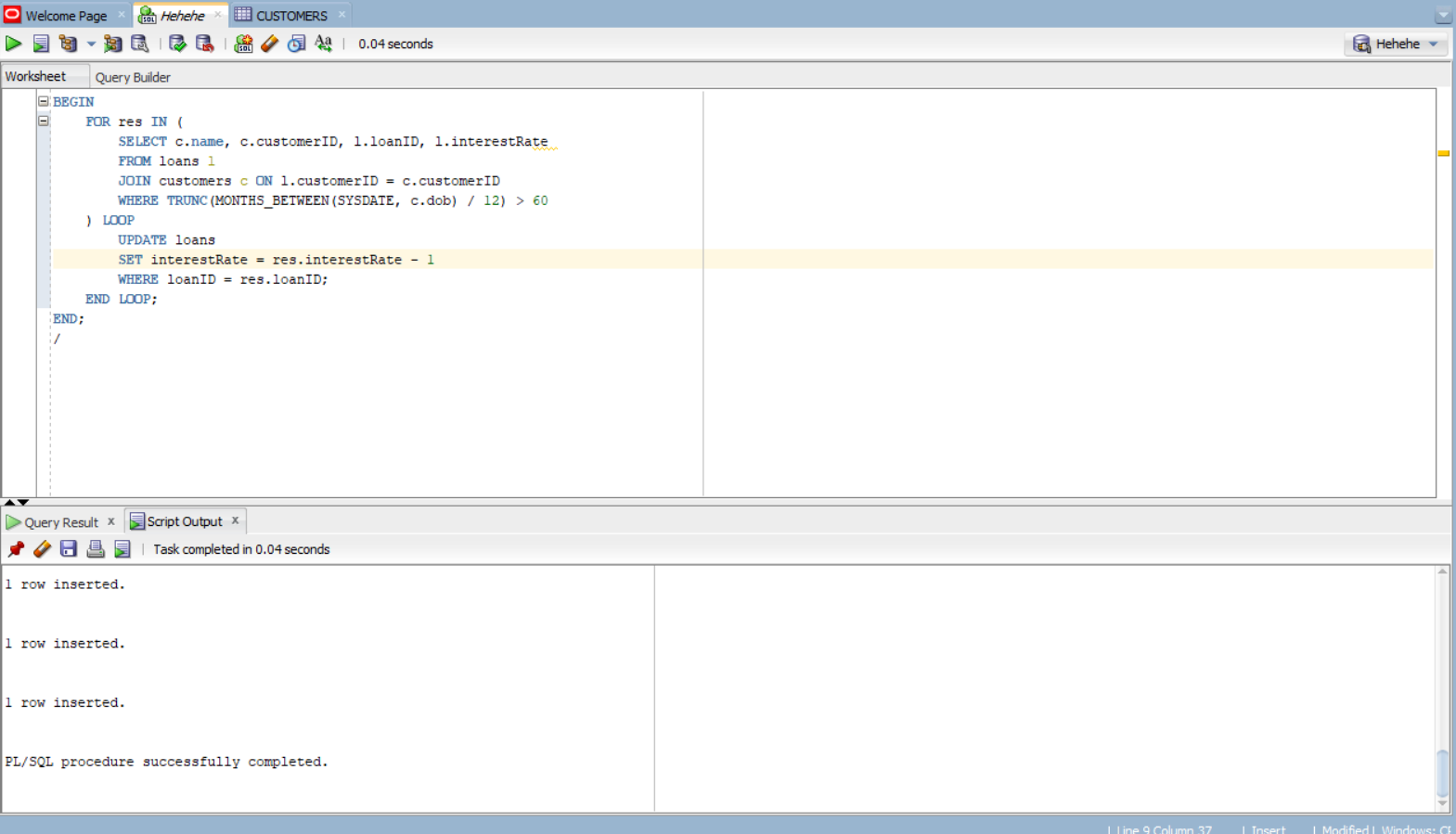
WHERE loanID = res.loanID;

END LOOP;

END;

/

**Output-**

****

**Scenario 2:**

ALTER TABLE Customers ADD IsVIP VARCHAR2(1);

-- Use 'Y' for TRUE, 'N' for FALSE

-- Alternatively, use a BOOLEAN column in PL/SQL logic only

UPDATE Customers SET IsVIP = 'N';

BEGIN

FOR cust IN (

SELECT CustomerID, Balance

FROM Customers

) LOOP

IF cust.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'Y'

WHERE CustomerID = cust.CustomerID;

END IF;

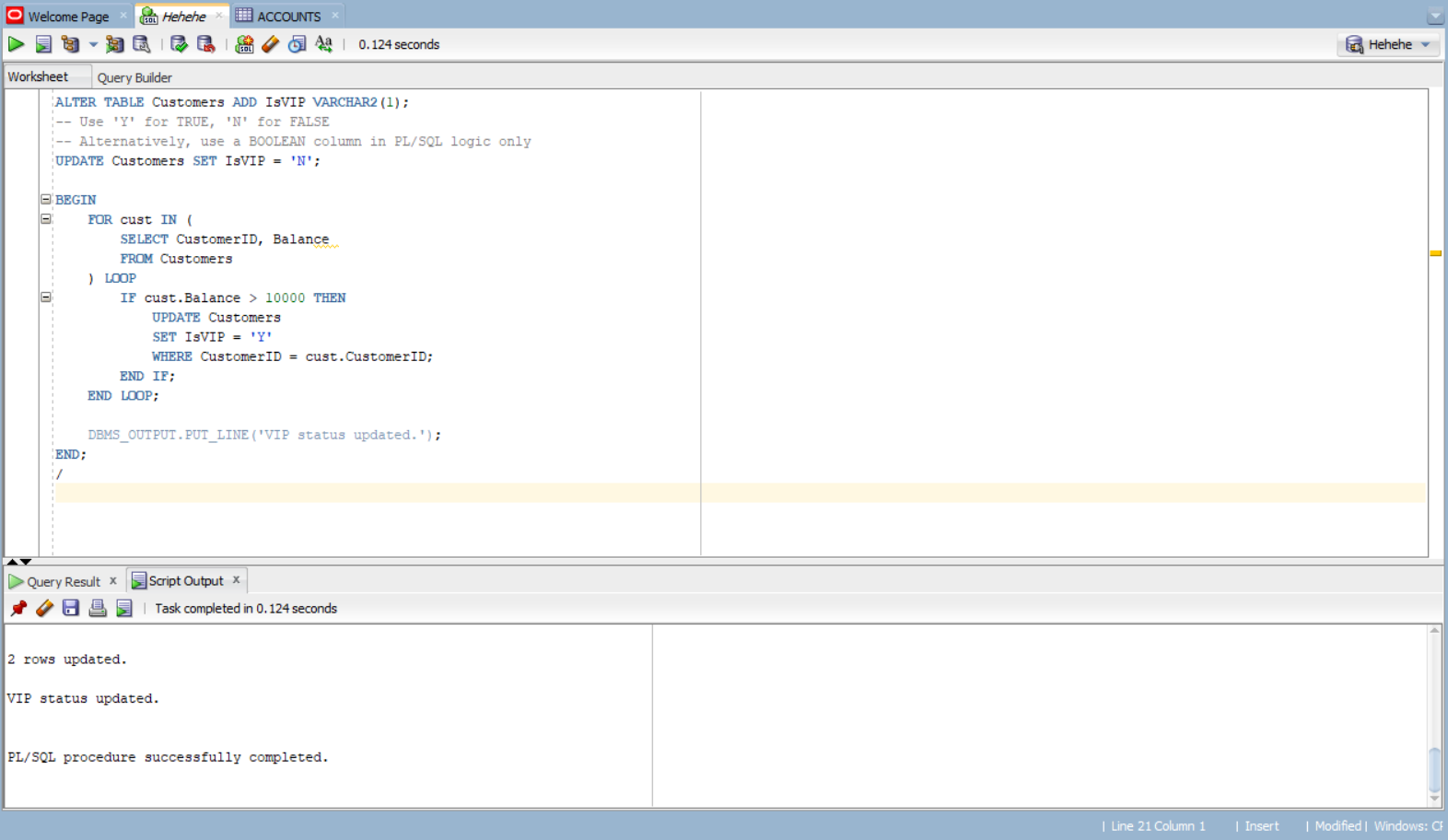
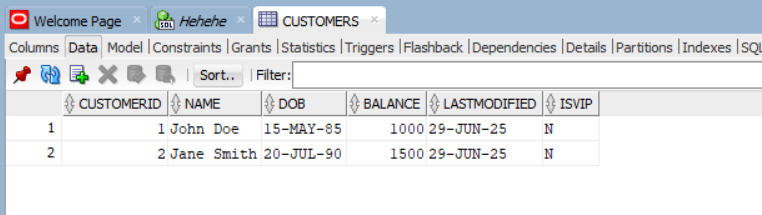
END LOOP;

DBMS\_OUTPUT.PUT\_LINE('VIP status updated.');

END;

/

**Output-**

****

**Scenario 3:**

set SERVEROUTPUT on;

INSERT INTO loans (loanID, customerID, loanAmount, interestRate, startDate, endDate)

VALUES (2, 2, 10000, 8, SYSDATE - 10, SYSDATE + 5);

BEGIN

for res in (

select c.name, c.customerID, l.loanID, l.enddate from loans l

join customers c on l.customerid = c.customerid

where l.enddate between sysdate and sysdate+30

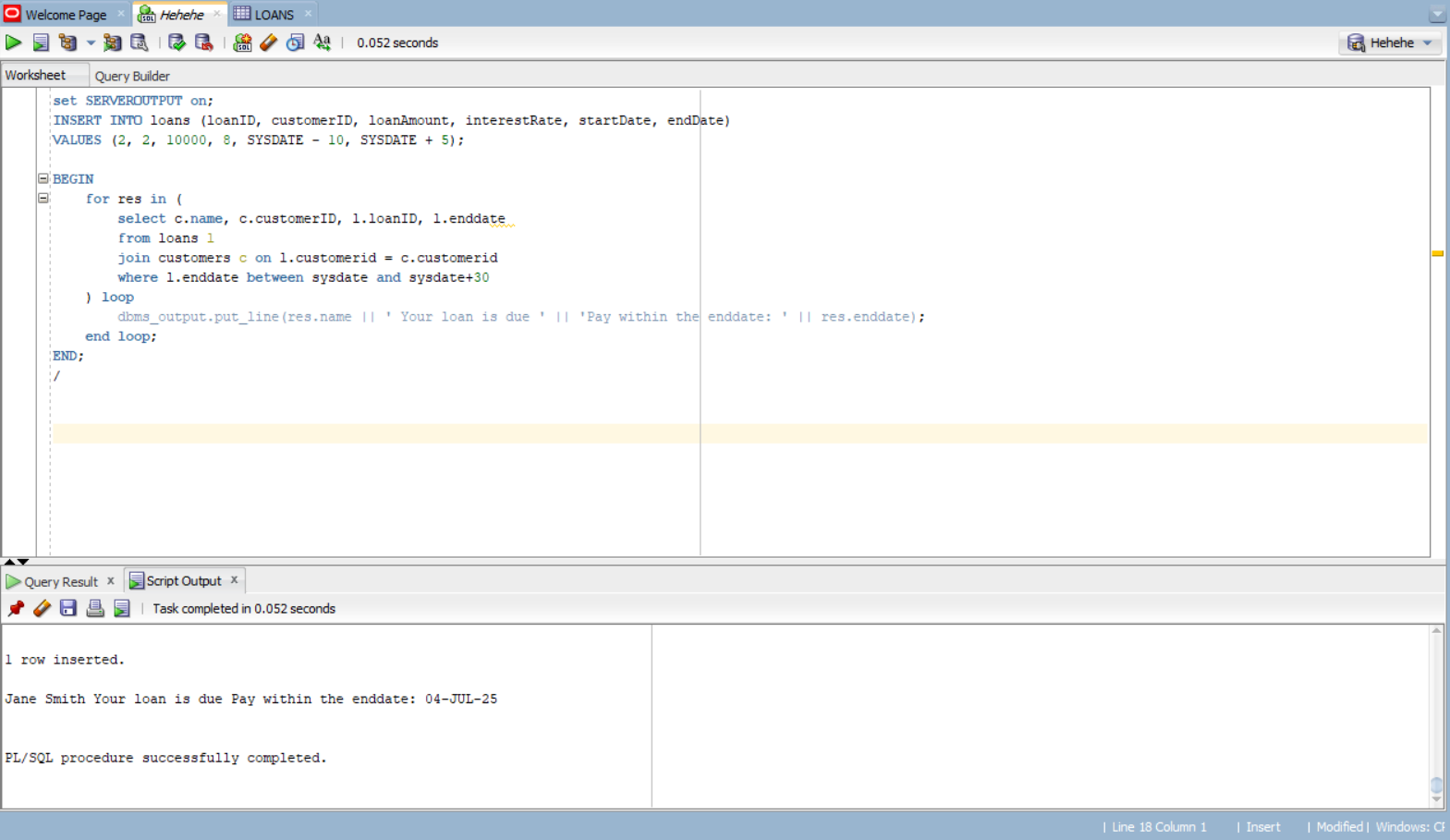
) loop

dbms\_output.put\_line(res.name || ' Your loan is due ' || 'Pay within the enddate: ' || res.enddate);

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.

**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.

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

* + **Question:** Write a stored procedure **TransferFunds** that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

**Solution-**

**Scenario 1:**

create or replace procedure ProcessMonthlyInterest is

begin

update Accounts

set Balance = Balance + Balance\*0.01

where AccountType = 'Savings';

end;

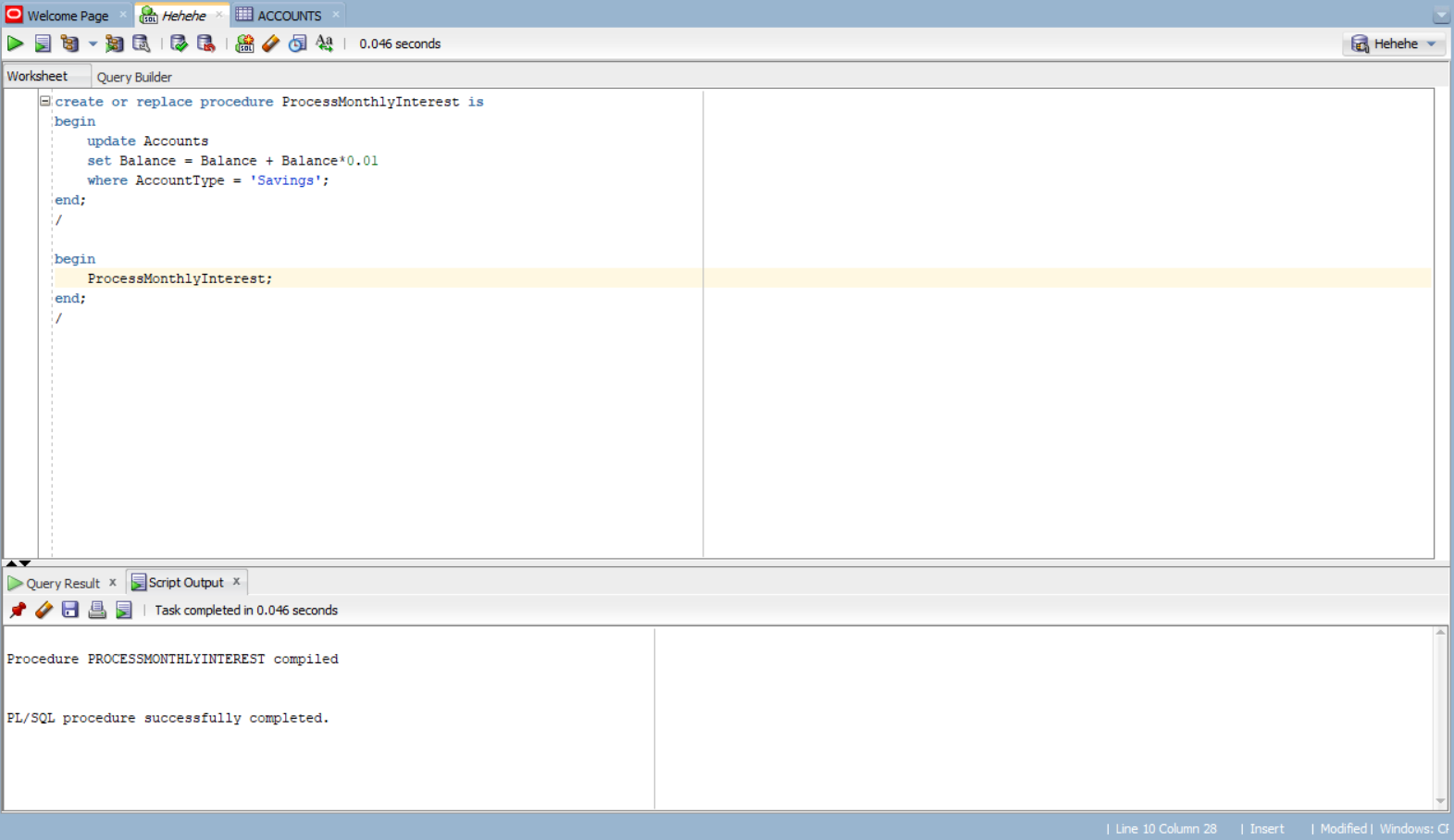
/

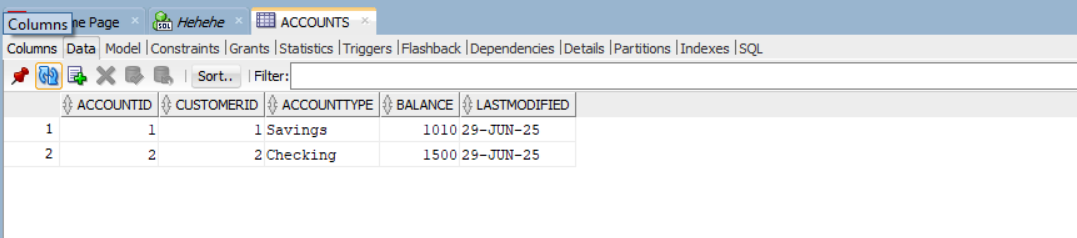
begin

ProcessMonthlyInterest;

end;

/

**Output-**

****

**Scenario 2:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

dept IN VARCHAR2,

bonus IN NUMBER

) IS

rows\_updated NUMBER := 0;

BEGIN

-- Update salaries for the given department

UPDATE Employees

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

WHERE department = dept;

rows\_updated := SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE(rows\_updated || ' employee(s) updated in department ' || dept);

EXCEPTION

WHEN OTHERS THEN

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

END;

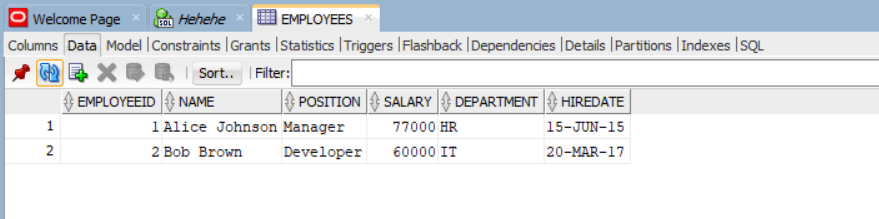
/

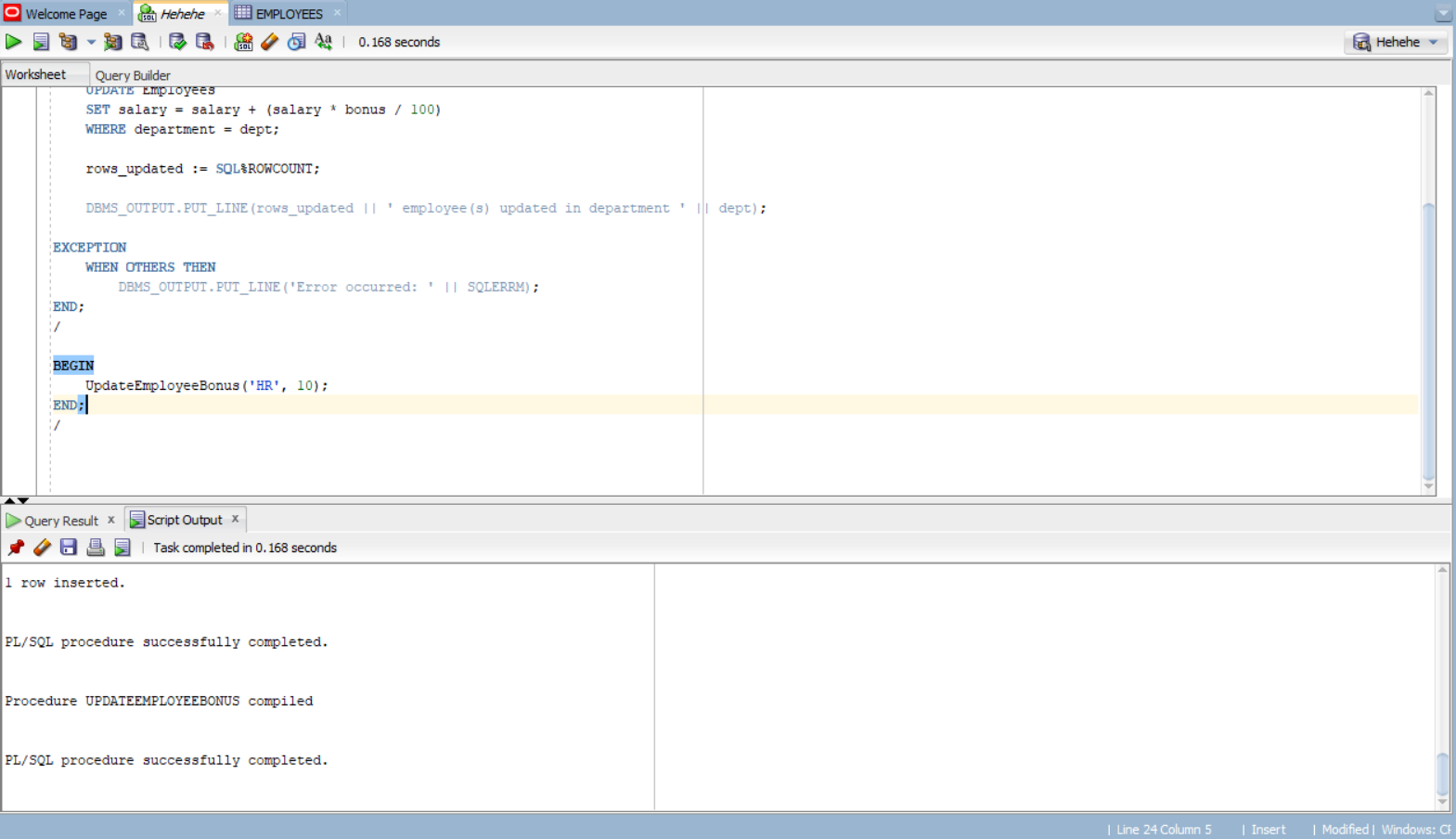
BEGIN

UpdateEmployeeBonus('HR', 10);

END;

/

**Output-**

****

**Scenario 3:**

create or replace procedure TransferFunds(

from\_acc IN number, to\_acc IN number, amount IN number

) is

v\_balance number;

begin

select balance into v\_balance

from accounts

where accountid = from\_acc

for update;

if v\_balance < amount then

raise\_application\_error(-20001, 'Insufficient Balance');

end if;

update accounts

set balance = balance + amount, lastModified = sysdate

where accountid = to\_acc;

update accounts

set balance = balance - amount, lastModified = sysdate

where accountid = from\_acc;

end;

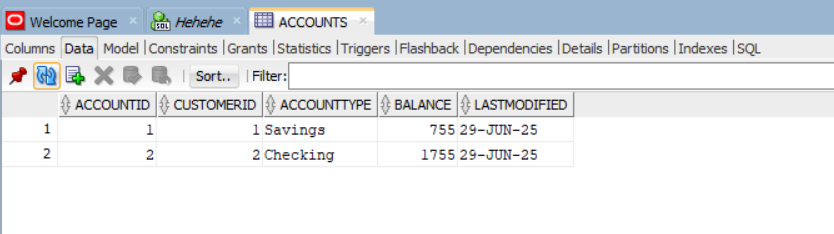
/

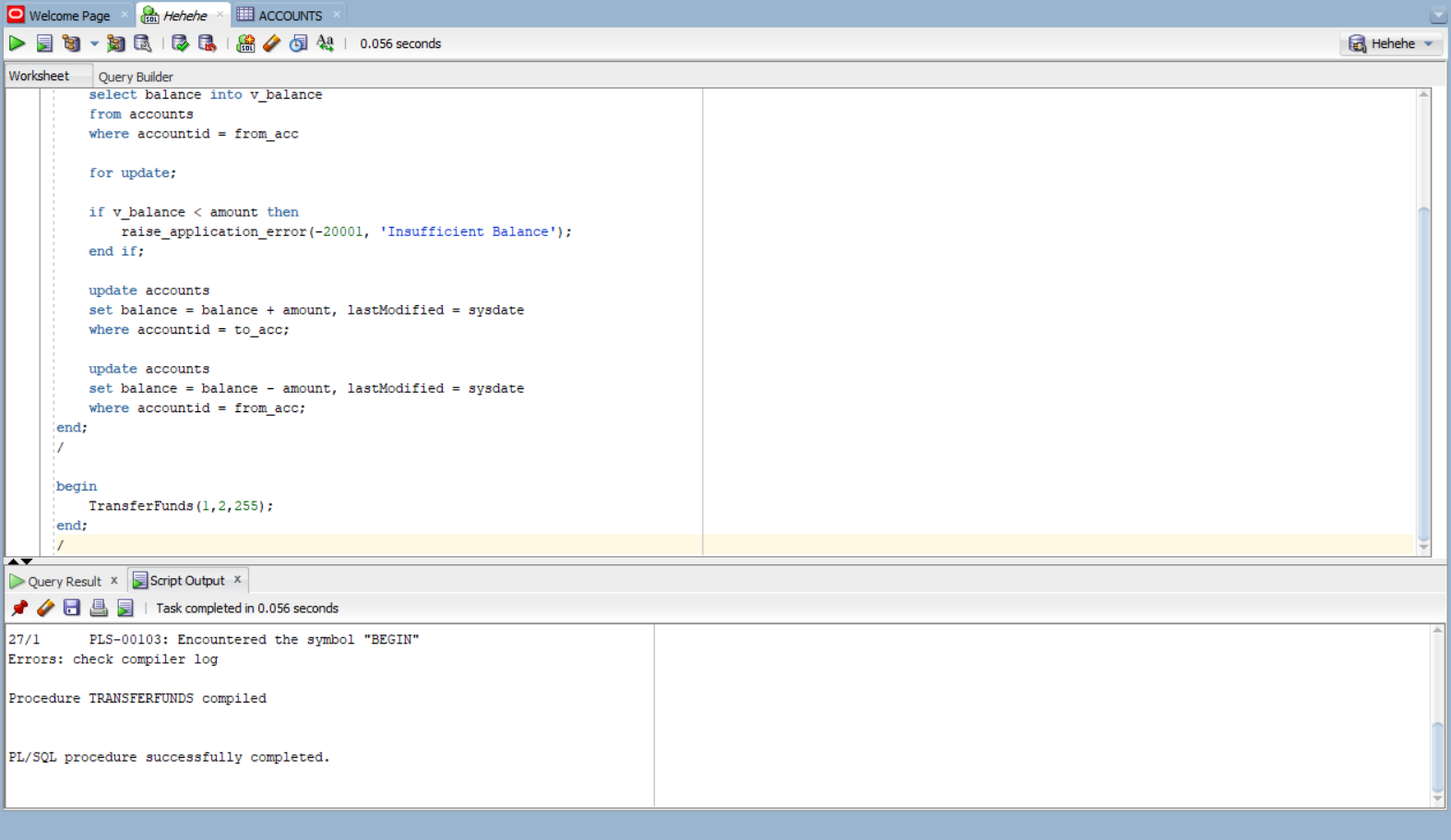
begin

TransferFunds(1,2,255);

end;

/

**Output-**

****