---------------------------------------------------------------------

-- LAB 16

--

-- Exercise 1

---------------------------------------------------------------------

USE TSQL;

GO

---------------------------------------------------------------------

-- Task 1

--

--1-- Write T-SQL code that will create a variable called @num as an int data type. Set the value of the variable

-- to 5 and display the value of the variable using the alias mynumber. Execute the T-SQL code.

-- Observe and compare the results that you got with the desired results shown in the file 52

-- - Lab Exercise 1 - Task 1\_1 Result.txt.

--2-- Write the batch delimiter GO after the written T-SQL code. In addition, write new T-SQL code

-- that defines two variables, @num1 and @num2, both as an int data type. Set the values to 4 and 6, respectively.

-- Write a SELECT statement to retrieve the sum of both variables using the alias totalnum.

-- Execute the T-SQL code.Observe and compare the results that you got with the desired

-- results shown in the file 53 - Lab Exercise 1 - Task 1\_2 Result.txt.

---------------------------------------------------------------------

--1--

Create PROCEDURE afishoNumra(@num INT OUTPUT )

as

Begin

Begin TRY

set @num =1

While(@num <= 5)

Begin

Print @num

SET @num = @num + 1

End

End TRY

BEGIN CATCH

Select ERROR\_LINE() as errLine,ERROR\_MESSAGE() AS errMsg,ERROR\_NUMBER() as errNumb,ERROR\_STATE() as errState,ERROR\_SEVERITY() as errSeverity,

ERROR\_PROCEDURE() as errProc

End CATCH

end;

--------------------------------------

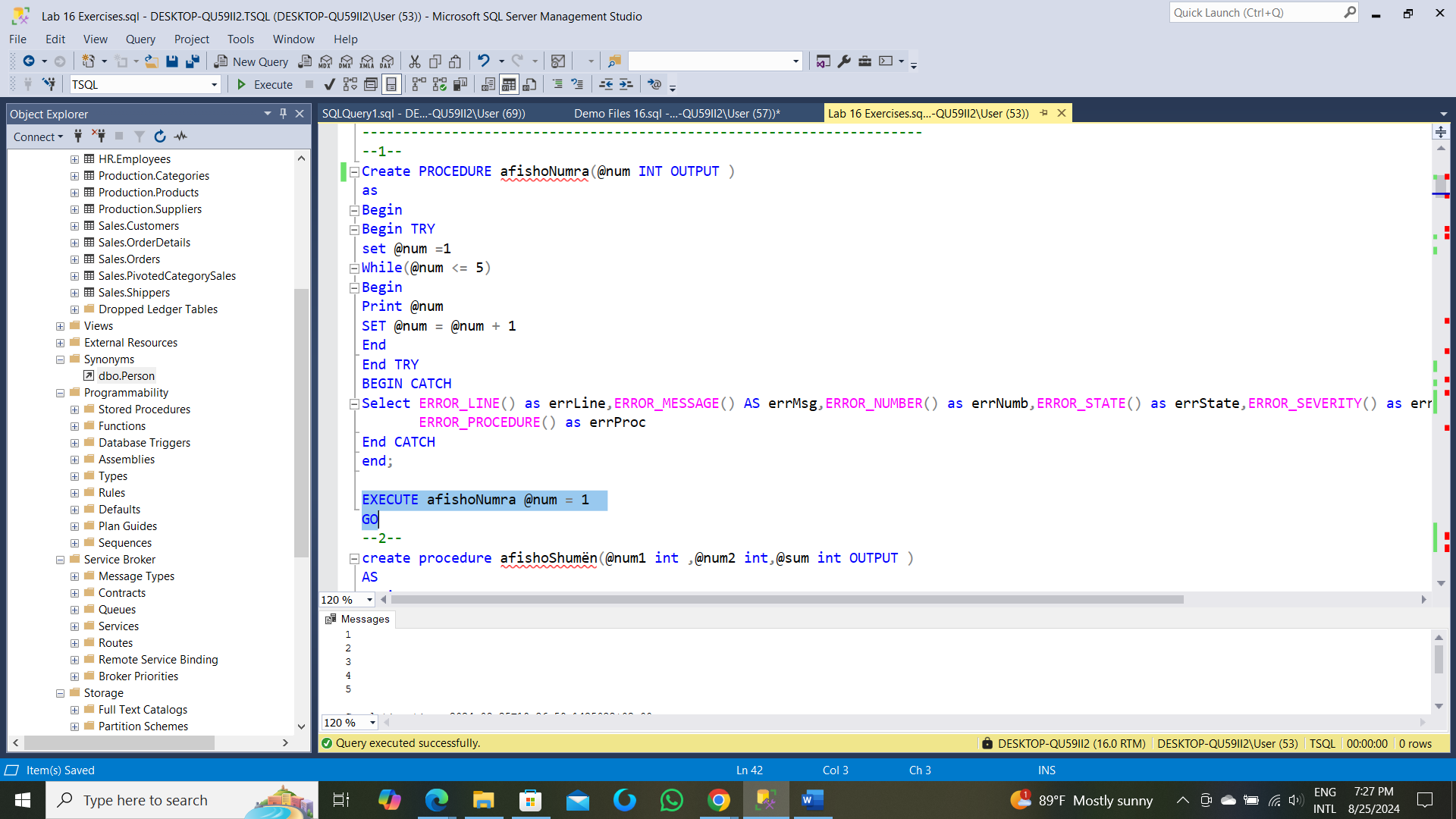
Commands completed successfully.

Completion time: 2024-08-25T19:26:00.0285665+02:00

--------------------------------------

EXECUTE afishoNumra @num = 1

GO



--2--

create procedure afishoShumën(@num1 int ,@num2 int,@sum int OUTPUT )

AS

Begin

Begin Try

Set @Sum = @num1 + @num2;

print @Sum

End Try

Begin Catch

Select ERROR\_LINE() AS errLine,

ERROR\_NUMBER() AS Errno,

ERROR\_MESSAGE() as errMsg,

Error\_State() as errStare,

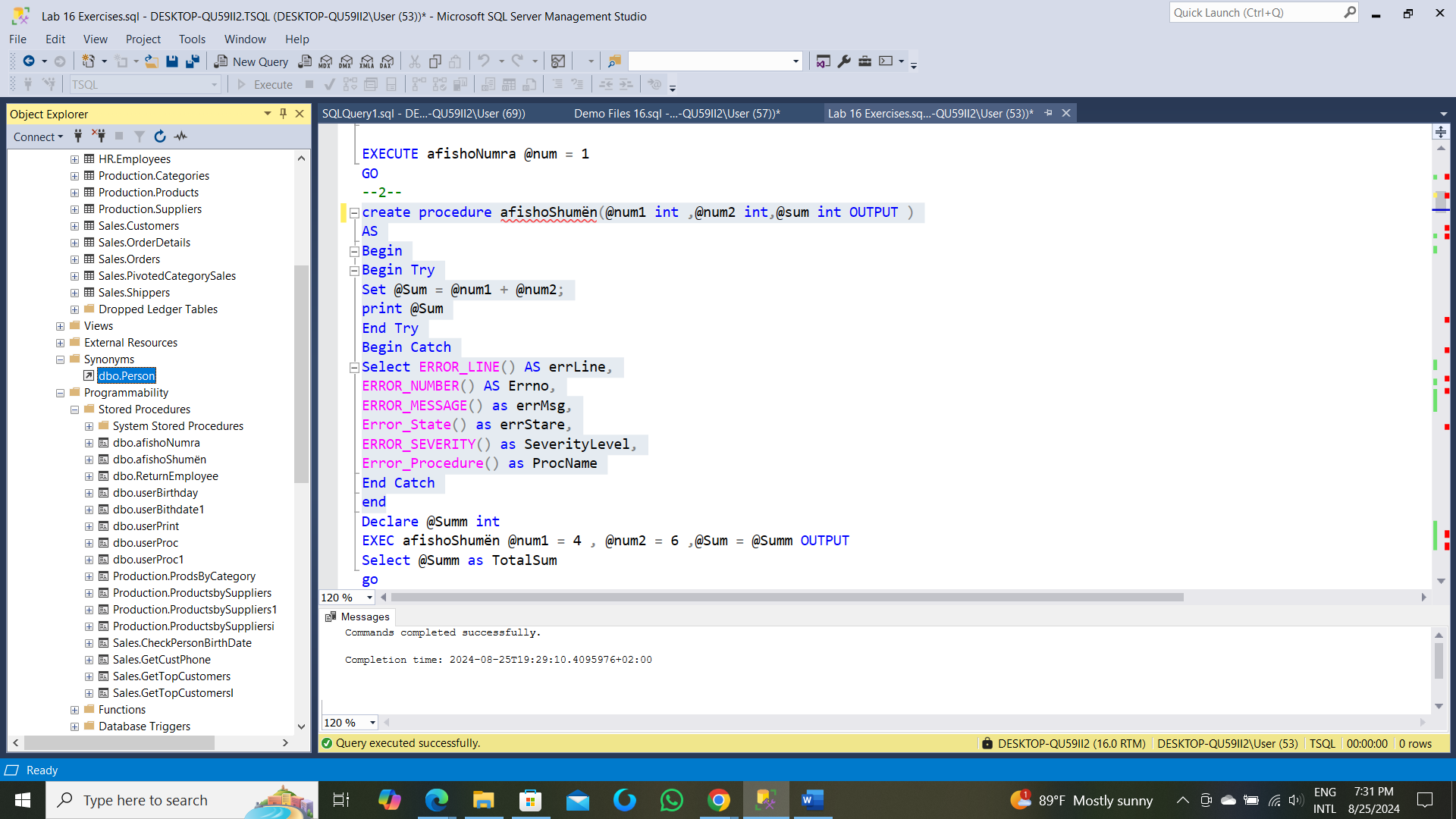
ERROR\_SEVERITY() as SeverityLevel,

Error\_Procedure() as ProcName

End Catch

End

--------------------------------------------------------------------------------------------



--------------------------------------------------------------------------------------------

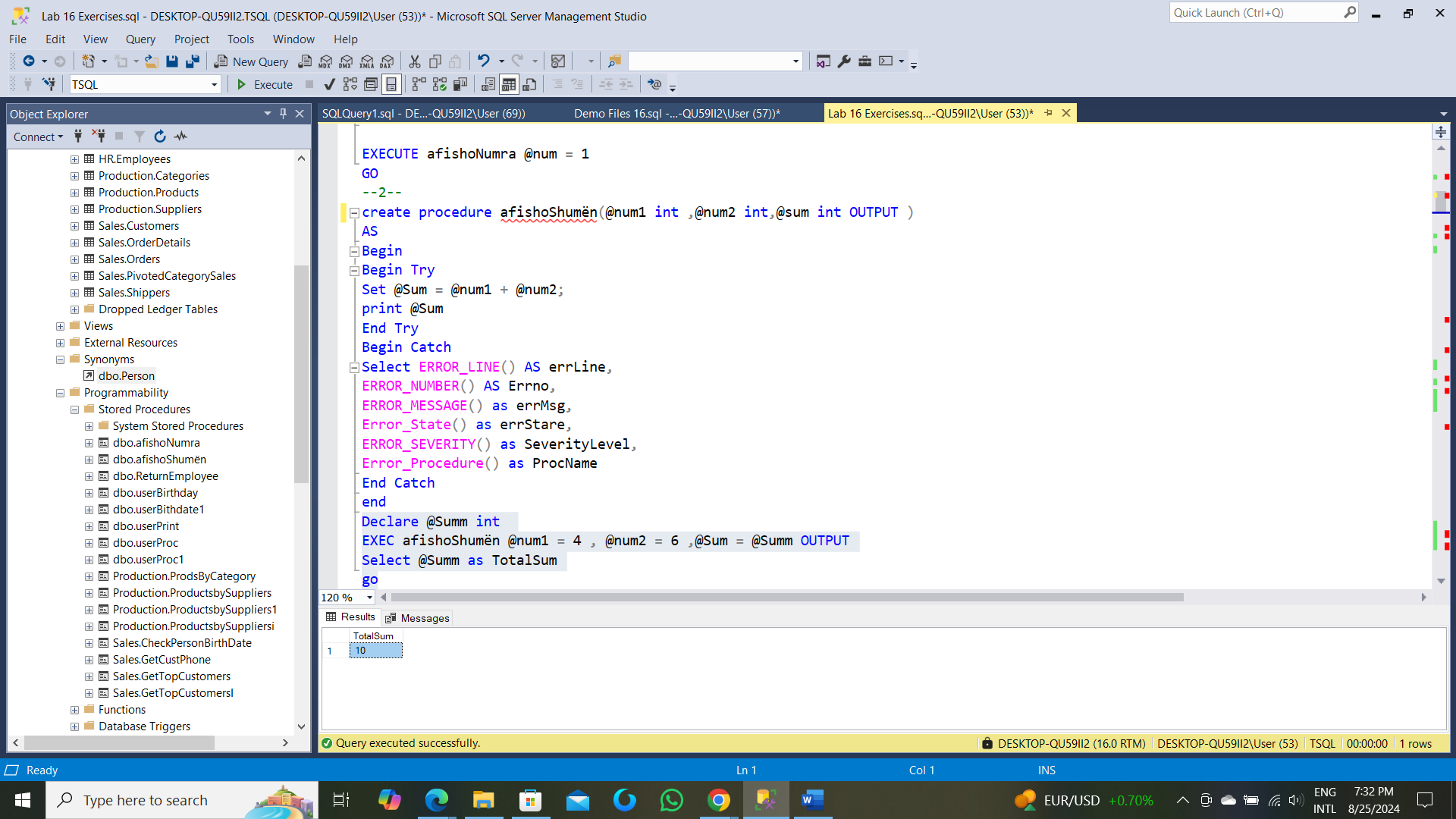
Declare @Summ int

EXEC afishoShumën @num1 = 4 , @num2 = 6 ,@Sum = @Summ OUTPUT

Select @Summ as TotalSum

GO

---------------------------------------------------------------------



---------------------------------------------------------------------

-- Task 2

--

-- Write T-SQL code that defines the variable @empname as an nvarchar(30) data type.

-- Set the value by executing a SELECT statement against the HR.Employees table.

-- Compute a value that concatenates the firstname and lastname column values. Add a space between the two column

-- values and filter the result to return the employee whose empid value is equal to 1.

-- Return the @empname variable’s value using the alias employee.

-- Execute the T-SQL code.

-- Observe and compare the results that you got with the desired results shown in

-- the file 54 - Lab Exercise 1 - Task 2Result.txt.

--

-- What would happen if the SELECT statement would return more than one row?

---------------------------------------------------------------------

CREATE Procedure ReturnEmployee(@empid int)

AS

Begin

Select employee = concat(e.firstname+ ' ',e.lastname)

From HR.Employees e

Where e.empid = @empid

End

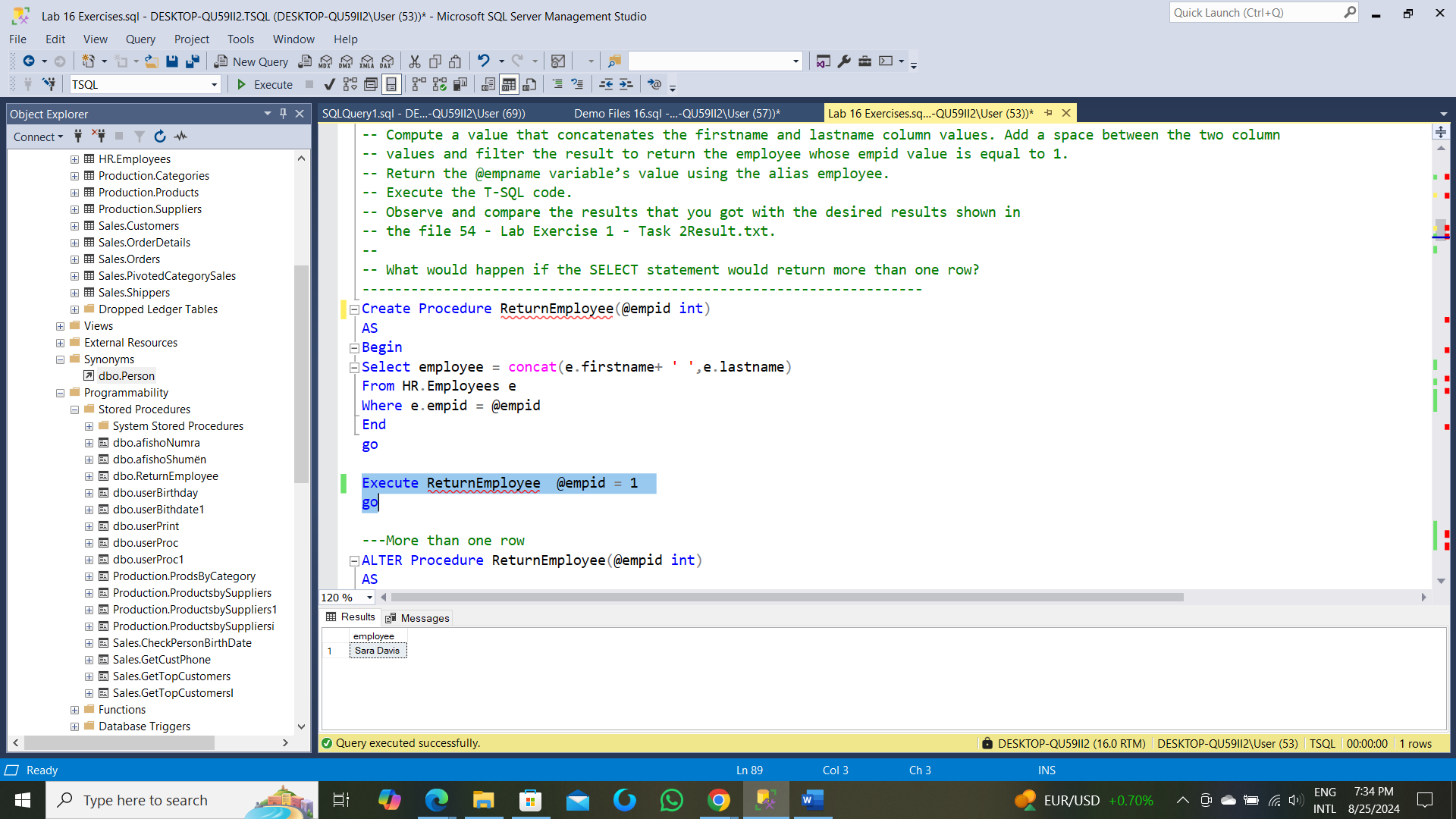
go

Commands completed successfully.

Completion time: 2024-08-25T19:33:51.3420307+02:00

Execute ReturnEmployee @empid = 1

go



---More than one row

ALTER Procedure ReturnEmployee(@empid int)

AS

Begin

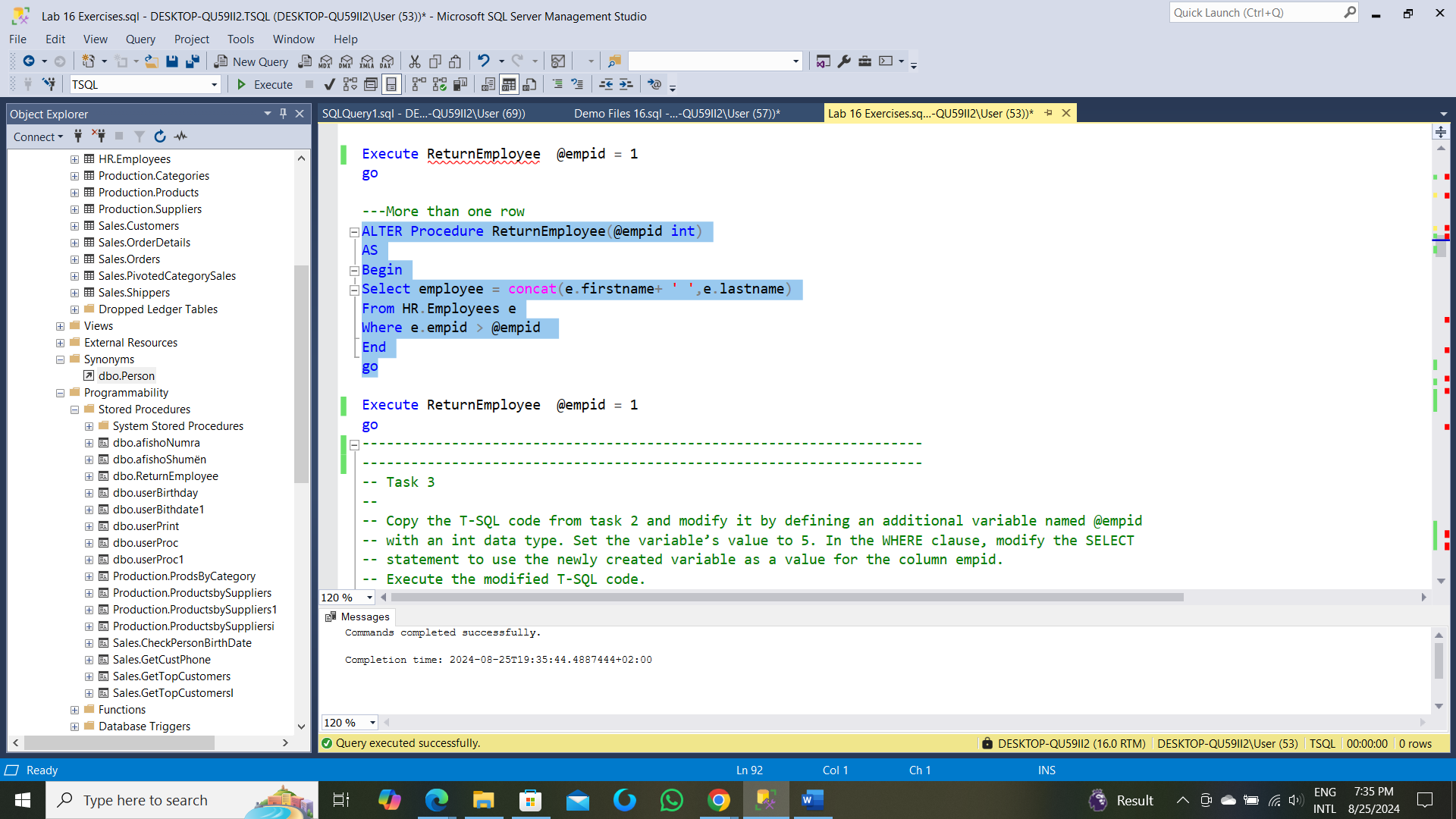
Select employee = concat(e.firstname+ ' ',e.lastname)

From HR.Employees e

Where e.empid > @empid

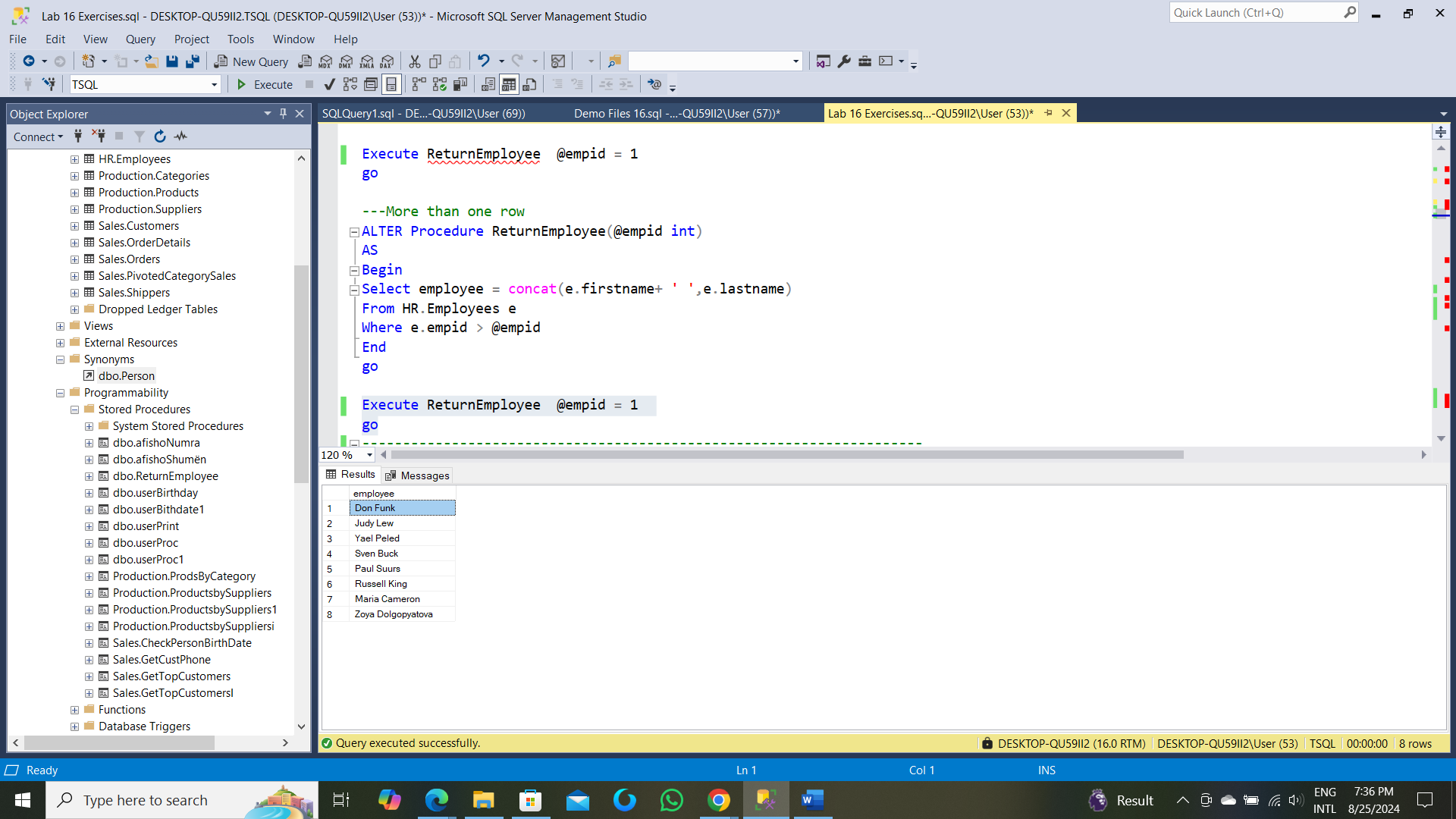
End

go



Execute ReturnEmployee @empid = 1

Go



---------------------------------------------------------------------

---------------------------------------------------------------------

-- Task 3

--

-- Copy the T-SQL code from task 2 and modify it by defining an additional variable named @empid

-- with an int data type. Set the variable’s value to 5. In the WHERE clause, modify the SELECT

-- statement to use the newly created variable as a value for the column empid.

-- Execute the modified T-SQL code.

-- Observe and compare the results that you got with the desired results shown

-- in the file 55 - Lab Exercise 1 - Task 3 Result.txt.

-- Change the @empid variable’s value from 5 to 2 and execute

-- the modified T-SQL code to observe the changes.

---------------------------------------------------------------------

ALTER Procedure ReturnEmployee(@empid int)

AS

Begin

Select employee = concat(e.firstname+ ' ',e.lastname)

From HR.Employees e

Where e.empid = @empid

End

go

Commands completed successfully.

Completion time: 2024-08-25T19:38:31.5887228+02:00

--1--

Execute ReturnEmployee @empid = 5

go

|employee

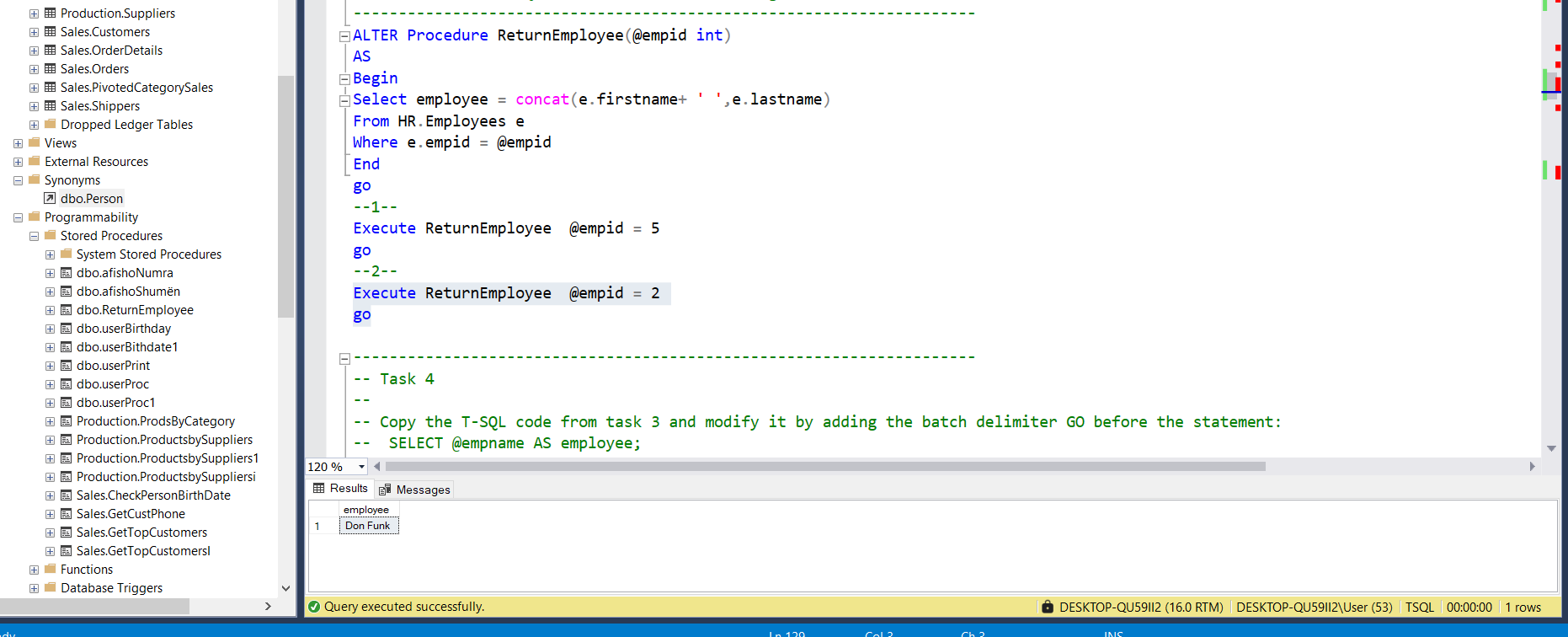
-------------

1|Sven Buck

--2--

Execute ReturnEmployee @empid = 2

Go



---------------------------------------------------------------------

-- Task 4

--

-- Copy the T-SQL code from task 3 and modify it by adding the batch delimiter GO before the statement:

-- SELECT @empname AS employee;

-- Execute the modified T-SQL code.

--

-- What happened? What is the error message? Can you explain why the batch delimiter caused an error?

---------------------------------------------------------------------

--1--

ALTER Procedure ReturnEmployee(@empid int ,@empname nvarchar(30) OUTPUT)

AS

Begin

Select @empname =concat(e.firstname+ ' ',e.lastname)

From HR.Employees e

Where e.empid = @empid

End

go

Commands completed successfully.

Completion time: 2024-08-25T19:42:40.6946352+02:00

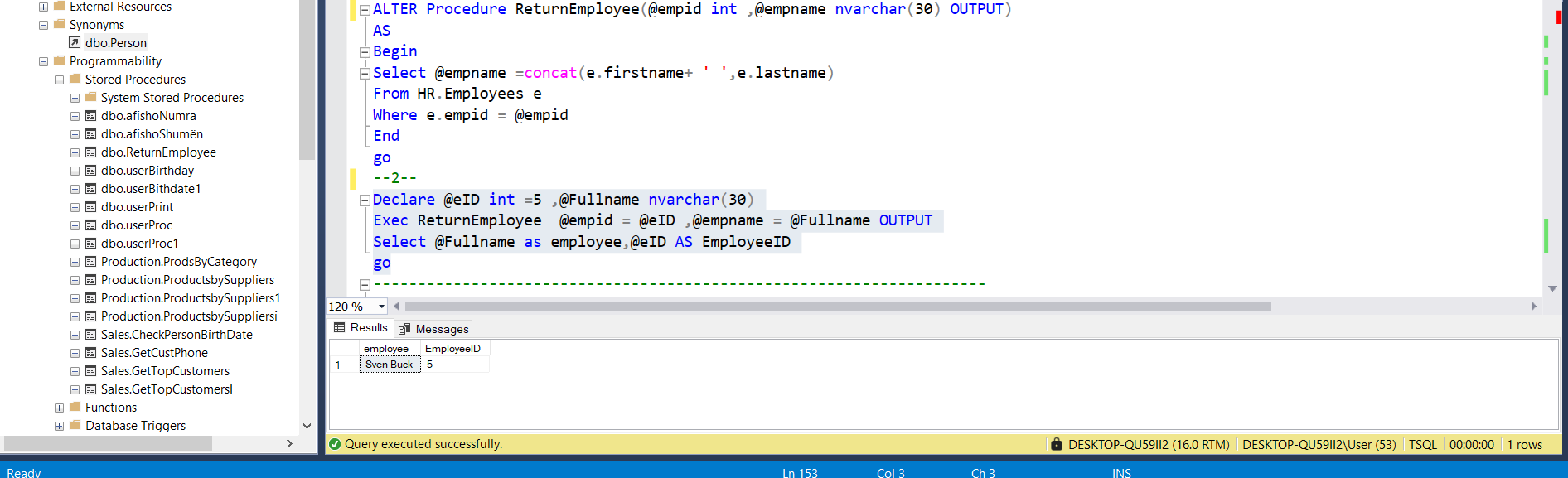
--2--

Declare @eID int =5 ,@Fullname nvarchar(30)

Exec ReturnEmployee @empid = @eID ,@empname = @Fullname OUTPUT

Select @Fullname as employee,@eID AS EmployeeID

Go



---------------------------------------------------------------------

-- LAB 16

--

-- Exercise 2

---------------------------------------------------------------------

USE TSQL;

GO

---------------------------------------------------------------------

-- Task 1

--

--1-- Write T-SQL code that defines the variable @result as an nvarchar(20) data type and the variable @i as an int

-- data type. Set the value of the @i variable to 8. Write an IF statement that implements the following logic:

-- For @i variable values less than 5, set the value of the @result variable to “Less than 5”.

-- For @i variable values between 5 and 10, set the value of the @result variable to “Between 5 and 10”.

-- For all @i variable values over 10, set the value of the @result variable to “More than 10”.

-- For other @i variable values, set the value of the @result variable to “Unknown”.

-- At the end of the T-SQL code, write a SELECT statement to retrieve the value of

-- the @result variable using the alias result. Highlight the complete T-SQL code and execute it.

-- Observe and compare the results that you got with the desired results shown in the file 62 - Lab Exercise 2 - Task 1 Result.txt.

--

--2-- Copy the T-SQL code and modify it by replacing the IF statement with a CASE expression to get the same result.

---------------------------------------------------------------------

--1--

--1--

create procedure userProc(@i int, @result as nvarchar(30) OUTPUT )

AS

Begin

--Begin try

if(@i < 5)

Begin

SET @result = N'Less than 5'

PRINT @result

End

Else if ((@i >5) AND (@i<10))

Begin

Set @result = N'Between 5 and 10'

Print @result

End

Else IF(@i > 10)

Begin

Set @result = 'More than 10'

Print @result

End

Else

Begin

Set @result = N'UNKNOWN'

Print @result

End

End

GO

--------------------------------------------------------

Commands completed successfully.

Completion time: 2024-08-25T19:45:57.1562863+02:00

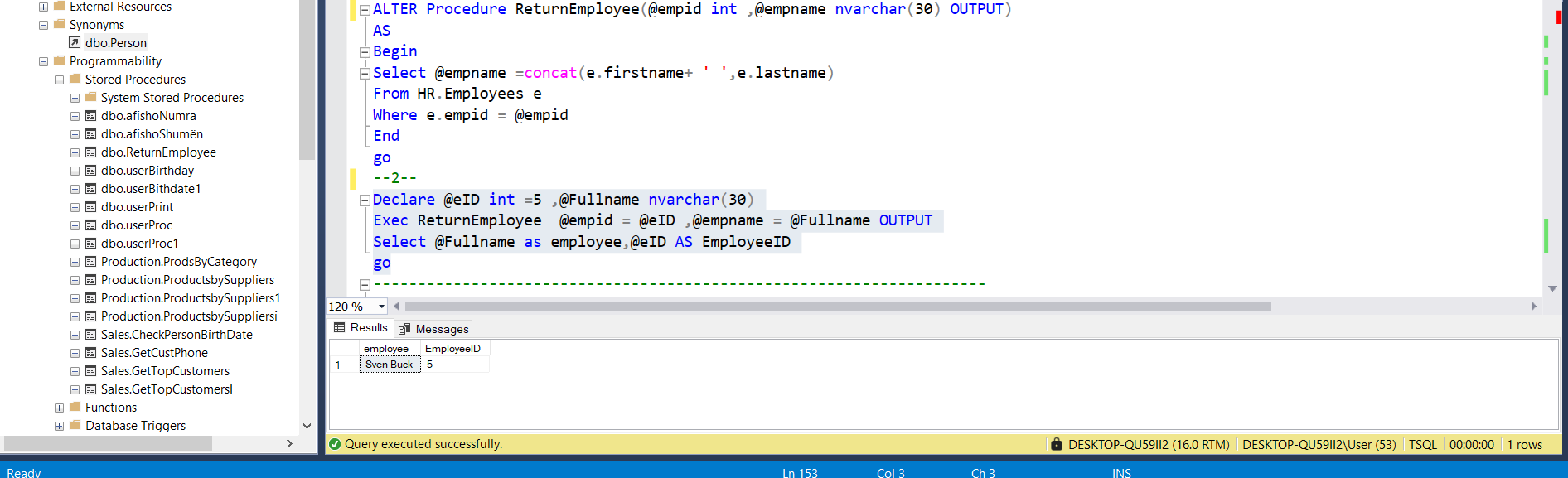
-----------------------------------------------------------------------

DECLARE @r nvarchar(30) , @ii as int =8

Exec userProc @result = @r OUTPUT,@i = @ii

Select @r as result

go



--2--

create Procedure userProc1(@i int)

as

Begin

Select selection = CASE WHEN (@i < 5) THEN N'Less than 5'

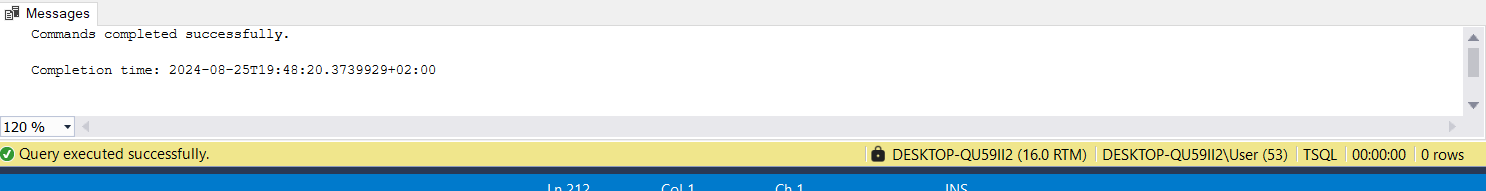
When ((@i >5) AND (@i<10)) then N'Between 5 and 10'

when (@i > 10) then 'More than 10'

else 'unknown'END

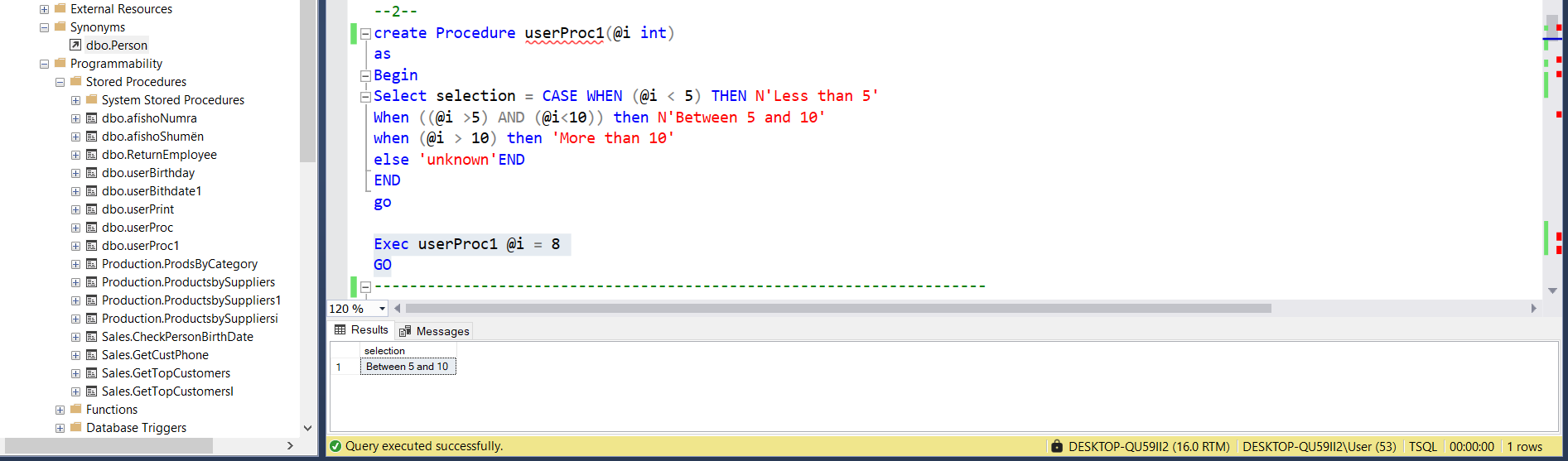
END

go



Exec userProc1 @i = 8

GO



---------------------------------------------------------------------

---------------------------------------------------------------------

-- Task 2

--

-- Write T-SQL code that declares two variables: @birthdate (data type date) and @cmpdate (data type date).

--

-- Set the value of the @birthdate variable by writing a SELECT statement against the HR.Employees

-- table and retrieve the column 'birthdate'. Filter the result to include only the employee with an empid equal to 5.

-- Set the @cmpdate variable to the value January 1, 1970.

-- Write an IF conditional statement by comparing the @birthdate and @cmpdate variable values. If @birthdate is less

-- than @cmpdate, use the PRINT statement to print the message “The person selected was born before

-- January 1, 1970”. Otherwise, print the message “The person selected was born on or after the January 1, 1970”.

-- Execute the T-SQL code.

-- Observe and compare the results that you got with the recommended result shown in the file 63 - Lab Exercise 2- Task 2

-- Result.txt. This is a simple example for the purpose of this exercise. Typically, there would be a different

-- statement block that would execute in each case.

---------------------------------------------------------------------

--M1--

CREATE PROCEDURE userBirthday(@cmpdate date = '1970/01/01')

AS

Begin

Select e.birthdate AS Birthdate ,Message = CASE WHEN (e.birthdate < @cmpdate)

THEN N'The person selected was born before -- January 1, 1970'

else N'The person selected was born on or after the January 1, 1970' end

From HR.Employees e

Where e.empid = 5

END

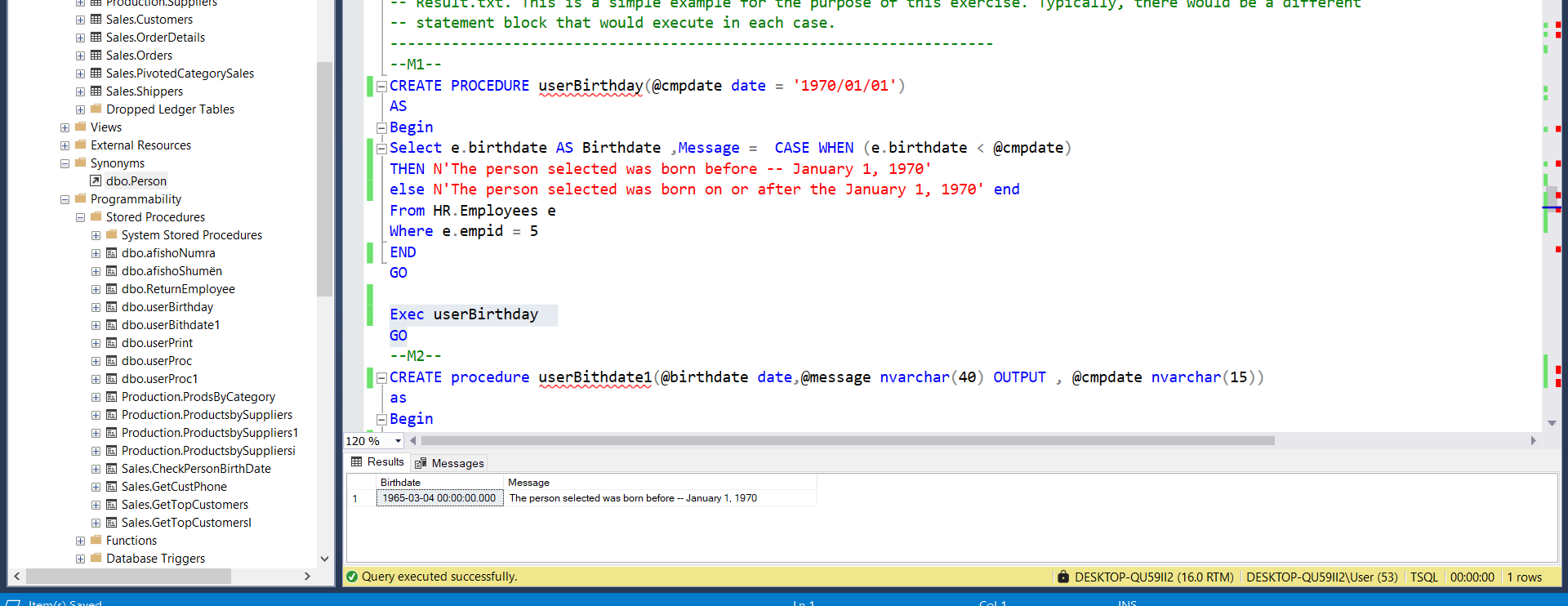
GO

Commands completed successfully.

Completion time: 2024-08-25T19:52:13.2611193+02:00

Exec userBirthday

GO



--M2--

CREATE procedure userBithdate1(@birthdate date,@message nvarchar(40) OUTPUT , @cmpdate nvarchar(15))

as

Begin

BEGIN TRY

IF(@birthdate < @cmpdate)

BEGIN

(Select @birthdate = e.birthdate From HR.Employees e Where e.empid = 5 )

SET @message = 'The person selected was born before -- January 1, 1970'

Print @message

END

ELSE

begin

SET @message = N'The person selected was born on or after the January 1, 1970'

PRINT @message

end

END TRY

Begin CATCH

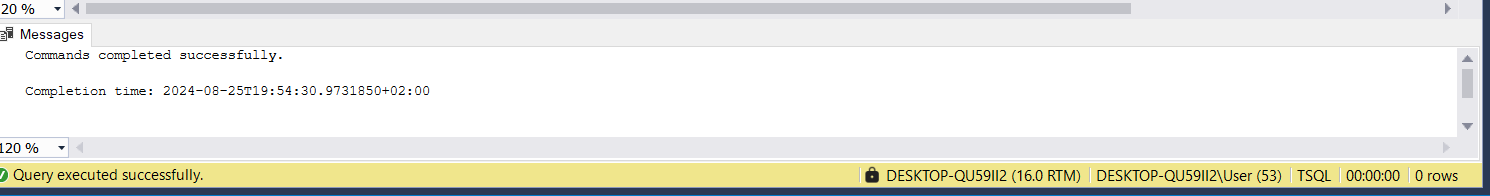
Select ERROR\_LINE() as errL,ERROR\_NUMBER() AS errNo ,ERROR\_MESSAGE() AS errM ,ERROR\_SEVERITY() as errSeverity ,

ERROR\_PROCEDURE() as errPROCED,ERROR\_STATE() AS errState

END CATCH

END

GO

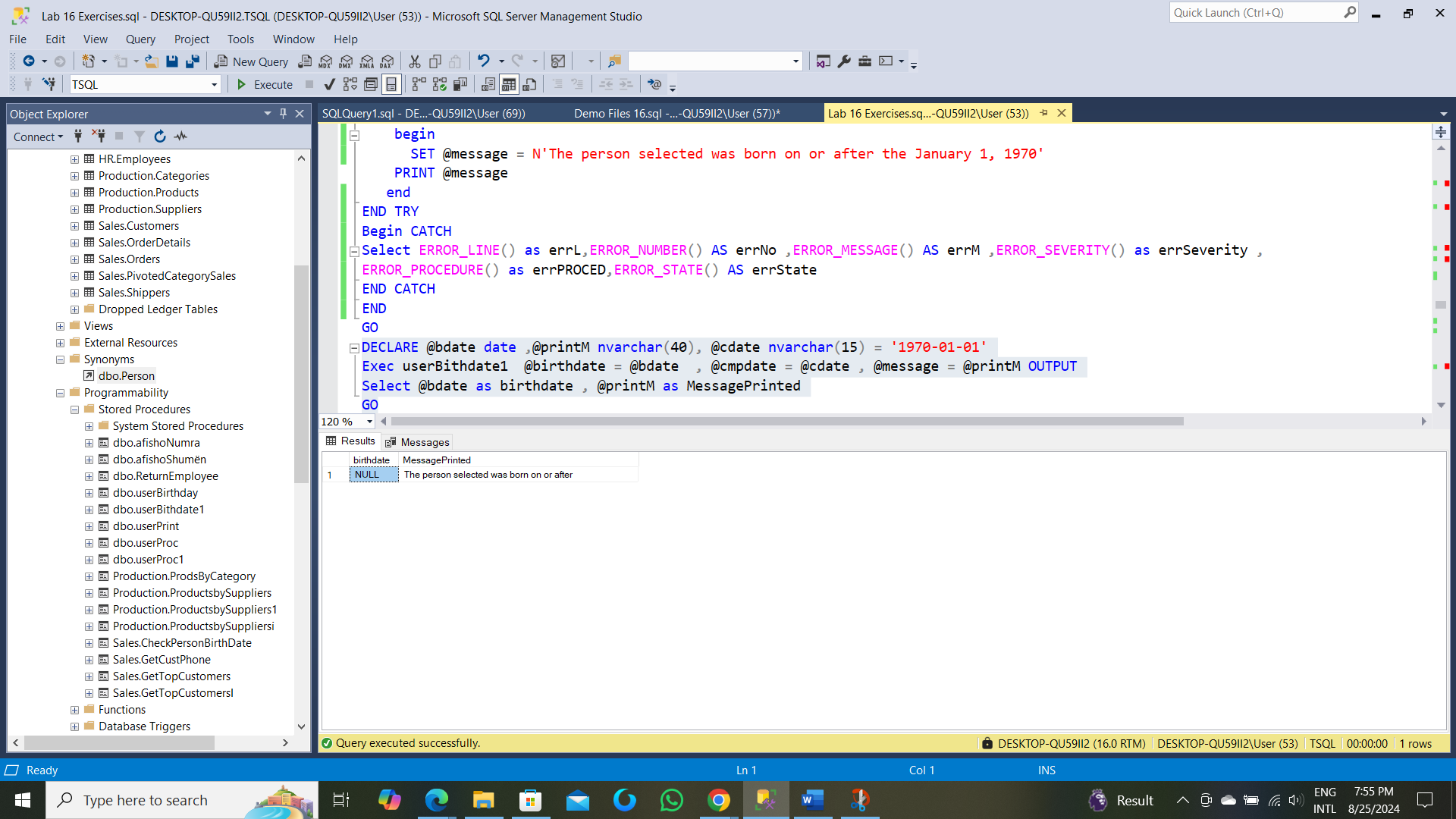


DECLARE @bdate date ,@printM nvarchar(40), @cdate nvarchar(15) = '1970-01-01'

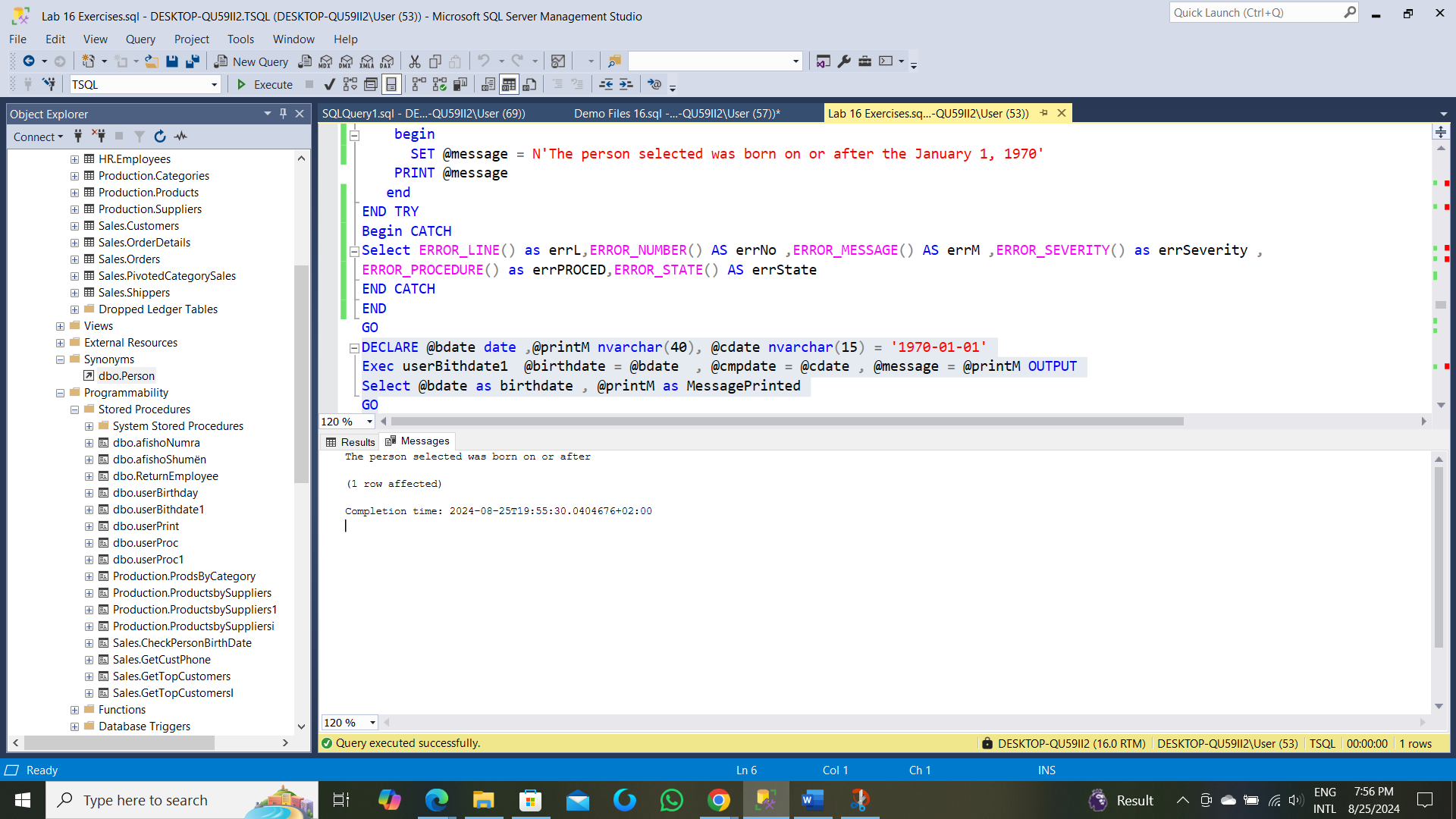
Exec userBithdate1 @birthdate = @bdate , @cmpdate = @cdate , @message = @printM OUTPUT

Select @bdate as birthdate , @printM as MessagePrinted

GO



?!!!!!!!!!



-------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------

-- Task 3

--

--1-- The IT department has provided T-SQL code that encapsulates the previous task in a stored procedure

-- named Sales.CheckPersonBirthDate. It has 'two parameters': @empid, which you use to specify an employee id,

-- and @cmpdate, which you use as a comparison date. Execute the provided T-SQL code:

--2-- Write an EXECUTE statement to invoke the Sales.CheckPersonBirthDate stored procedure using

-- the parameters of 3 for @empid and January 1, 1990, for @cmpdate. Execute the T-SQL code.

-- Observe and compare the results that you got with the recommended result shown in

-- the file 64 - Lab Exercise 2 - Task 3 Result.txt.

---------------------------------------------------------------------

CREATE PROCEDURE Sales.CheckPersonBirthDate

@empid int,

@cmpdate date

AS

DECLARE

@birthdate date;

SET @birthdate = (SELECT birthdate FROM HR.Employees WHERE empid = @empid);

IF @birthdate < @cmpdate

PRINT 'The person selected was born before ' + FORMAT(@cmpdate, 'MMMM d, yyyy', 'en-US')

ELSE

PRINT 'The person selected was born on or after ' + FORMAT(@cmpdate, 'MMMM d, yyyy', 'en-US');

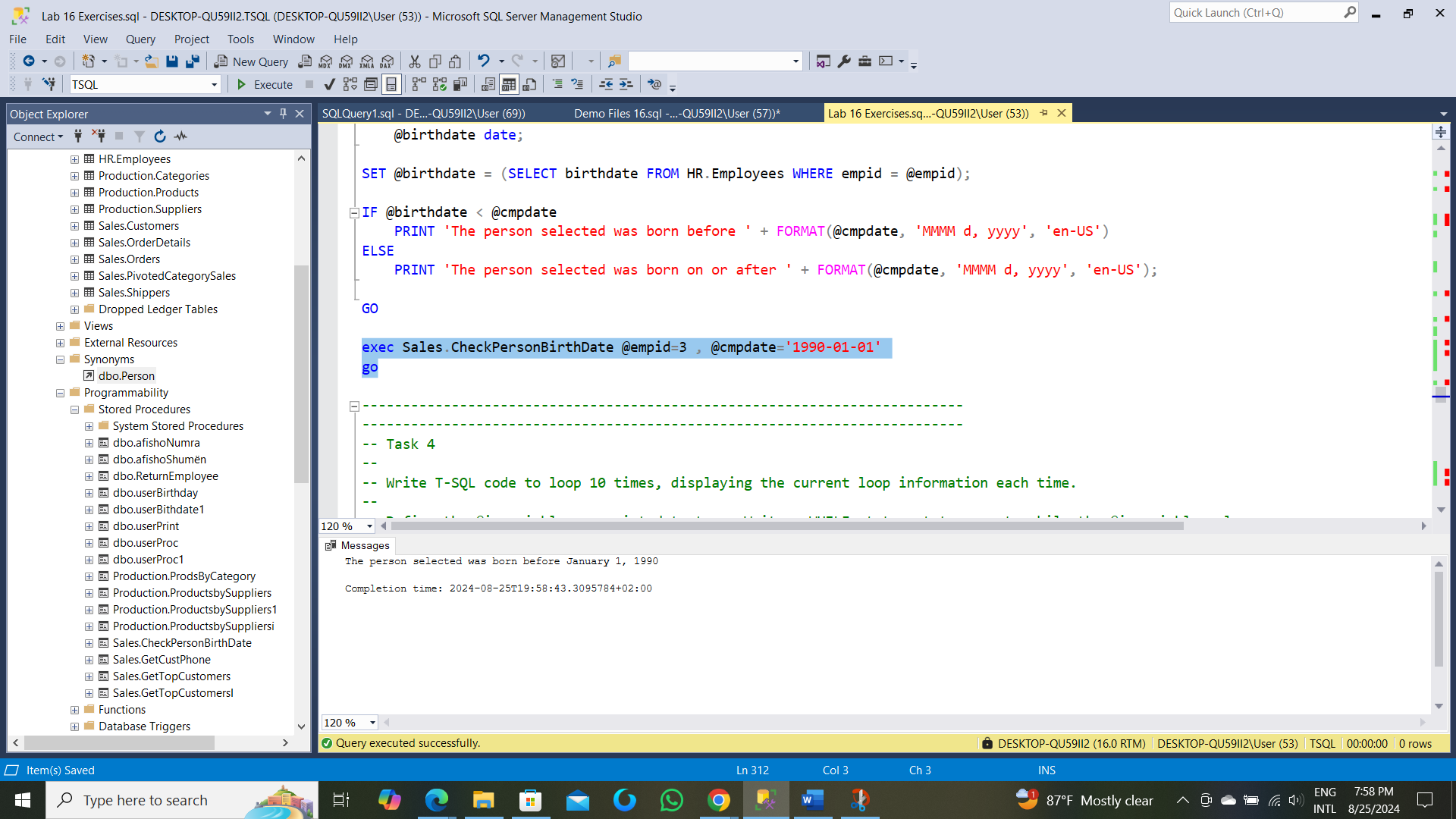
GO

Commands completed successfully.

Completion time: 2024-08-25T19:58:05.7469904+02:00

exec Sales.CheckPersonBirthDate @empid=3 , @cmpdate='1990-01-01'

go



--------------------------------------------------------------------------

--------------------------------------------------------------------------

-- Task 4

--

-- Write T-SQL code to loop 10 times, displaying the current loop information each time.

--

-- Define the @i variable as an int data type. Write a WHILE statement to execute while the @i variable value

-- is less or equal 10. Inside the loop statement, write a PRINT statement to display the value of

-- the @i variable using the alias loopid. Add T-SQL code to increment the @i variable value by 1.

-- Observe and compare the results that you got with the recommended result shown in the

-- file 65 - Lab Exercise 2 - Task 4 Result.txt.

---------------------------------------------------------------------

alter procedure userPrint(@i int output)

as

begin

Set @i = 1

while(@i <= 10)

Begin

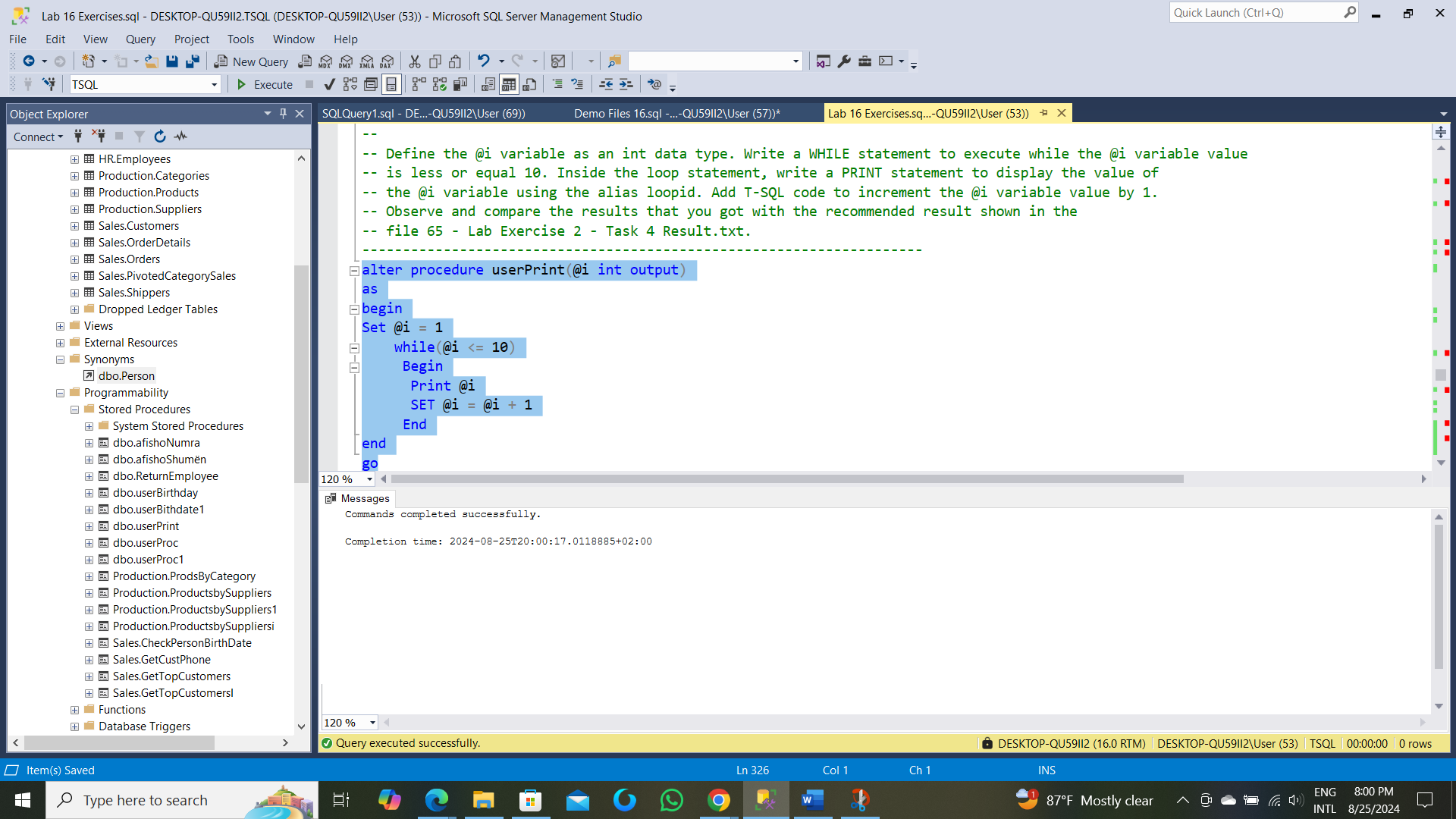
Print @i

SET @i = @i + 1

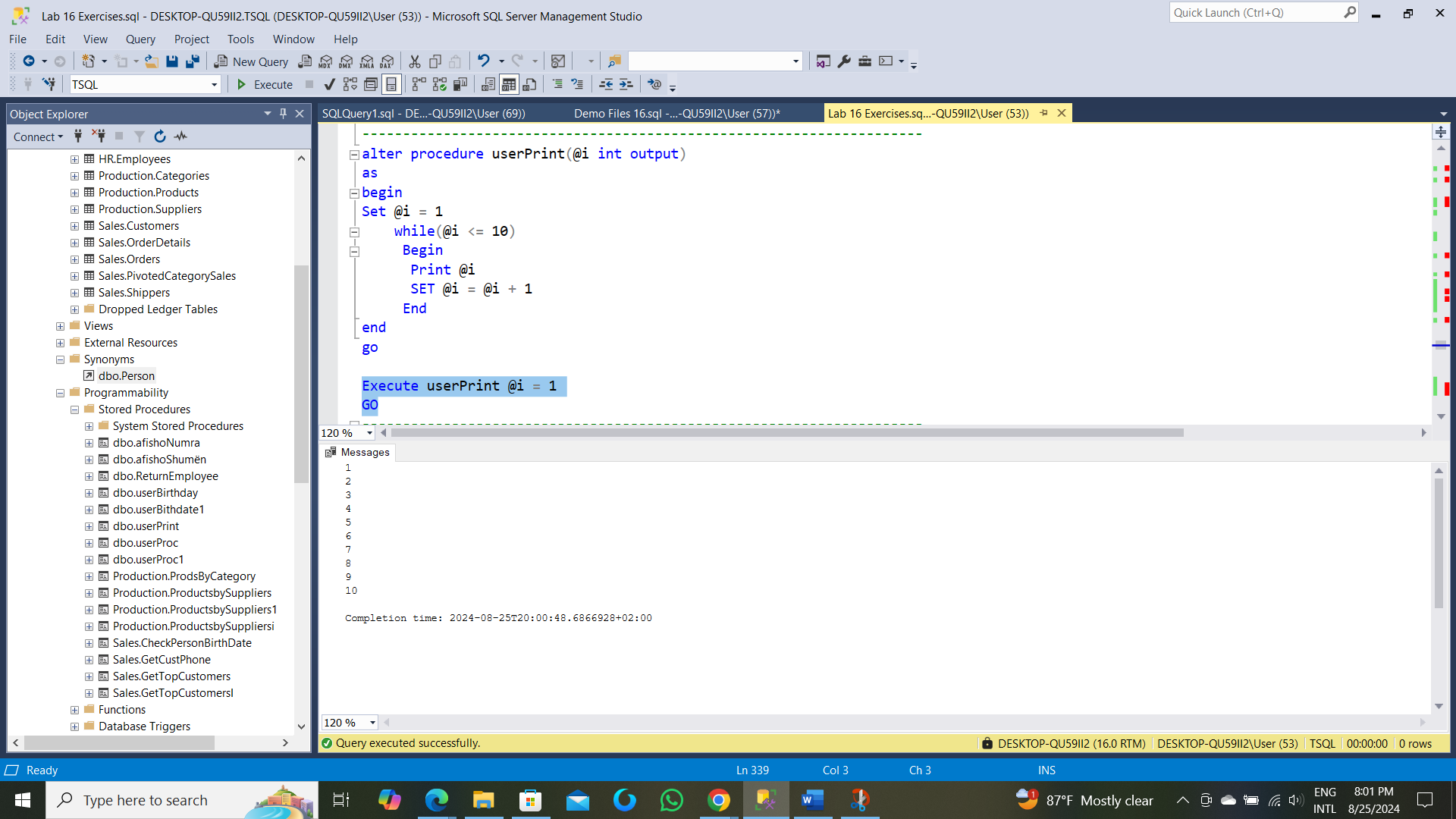
End

end

go



Execute userPrint @i = 1 GO



---------------------------------------------------------------------

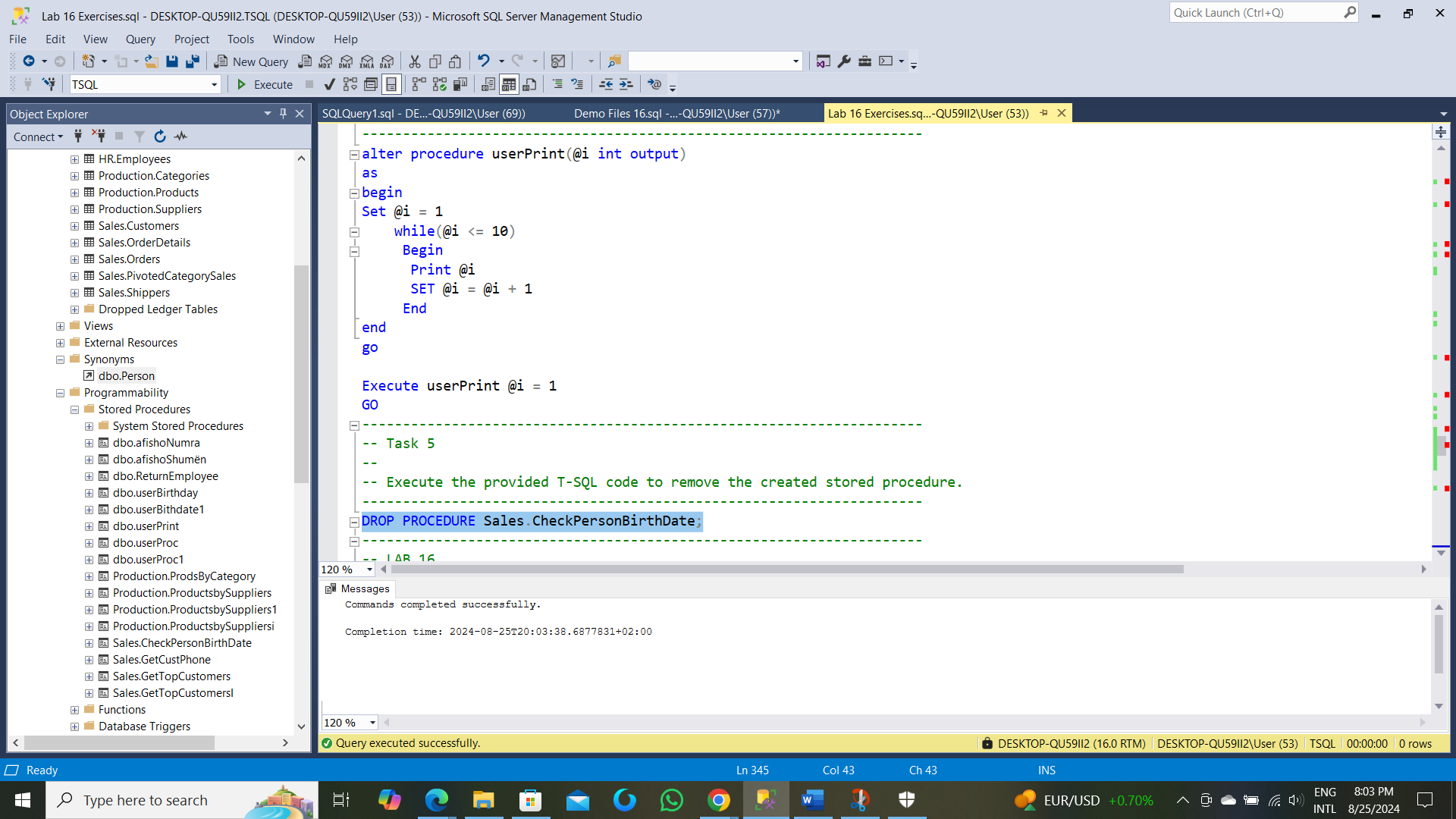
-- Task 5

--

-- Execute the provided T-SQL code to remove the created stored procedure.

---------------------------------------------------------------------

DROP PROCEDURE Sales.CheckPersonBirthDate;



---------------------------------------------------------------------

-- LAB 16

--

-- Exercise 3

---------------------------------------------------------------------

USE TSQL;

GO

---------------------------------------------------------------------

-- Task 1

--

-- Write T-SQL code that defines the variable @SQLstr as nvarchar(200) data type. Set the value of

-- the variable to a SELECT statement that retrieves the empid, firstname, and lastname columns in

-- the HR.Employees table.Write an EXECUTE statement to invoke the written dynamic SQL statement

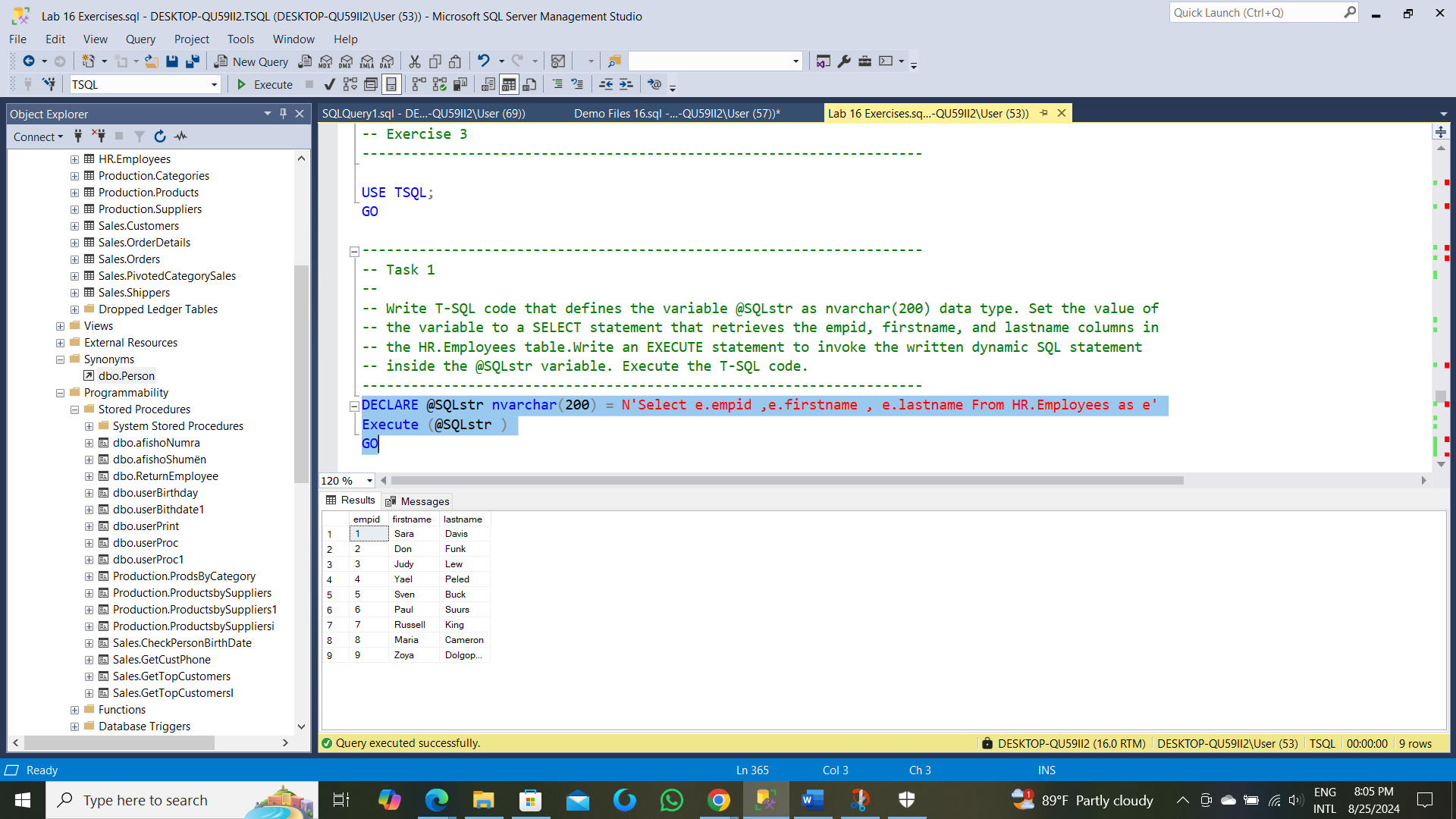
-- inside the @SQLstr variable. Execute the T-SQL code.

---------------------------------------------------------------------

DECLARE @SQLstr nvarchar(200) = N'Select e.empid ,e.firstname , e.lastname From HR.Employees as e'

Execute (@SQLstr )

GO



---------------------------------------------------------------------

-- Task 2

--

-- Copy the previous T-SQL code and modify it to include in the dynamic batch stored in @SQLstr,

-- a filter in which empid is equal to a parameter named @empid. In the calling batch, define a variable

-- named @SQLparam as nvarchar(100). This variable will hold the definition of the @empid parameter.

-- This means setting the value of the @SQLparam variable to @empid int.

-- Write an EXECUTE statement that uses sp\_executesql to invoke the code in the @SQLstr variable,

-- passing the parameter definition stored in the @SQLparam variable to sp\_executesql.

-- Assign the value 5 to the @empid parameter in the current execution.

-- Observe and compare the results that you got with the recommended result shown

-- in the file 73 - Lab Exercise 3 - Task 2 Result.txt.

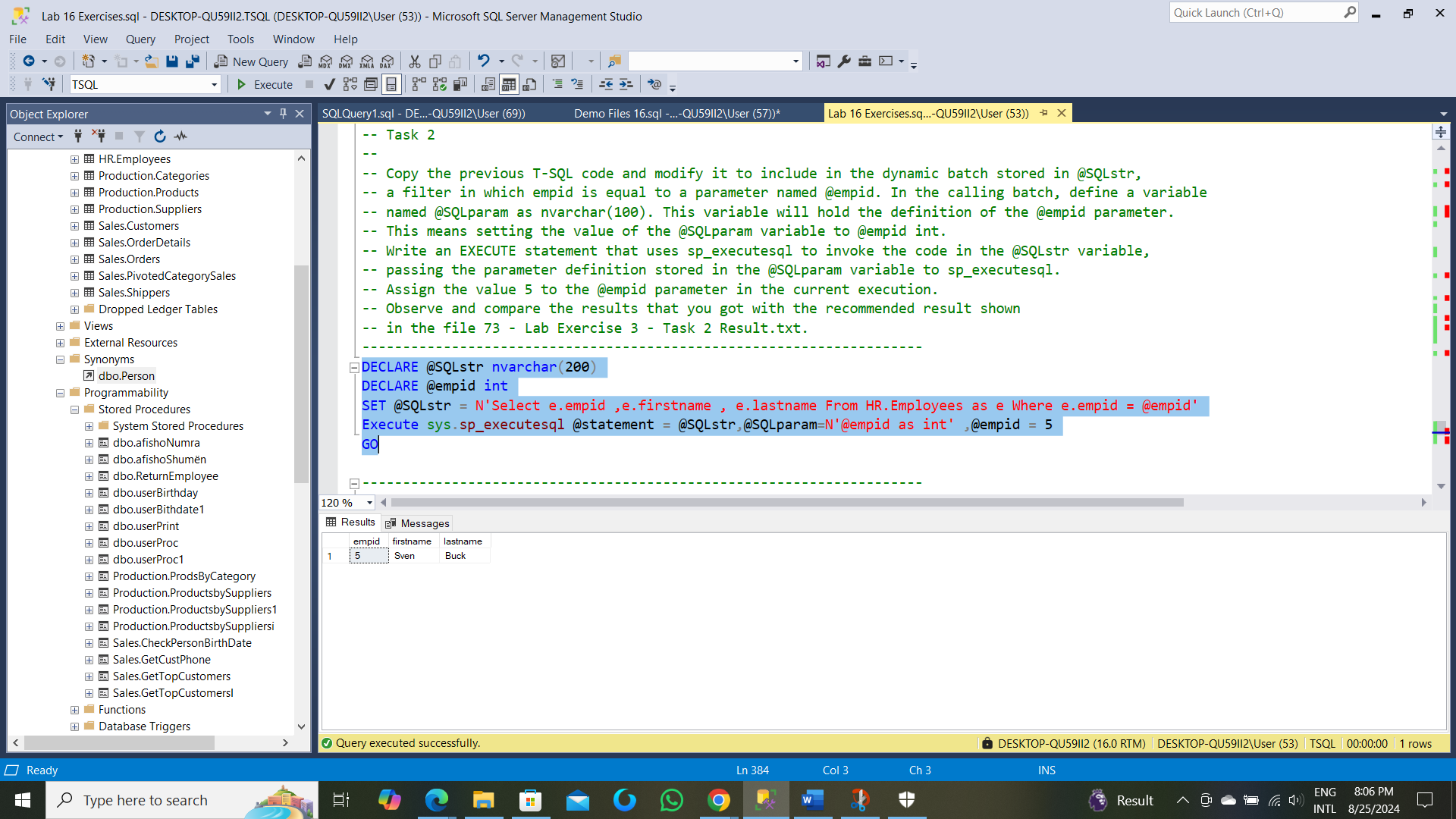
---------------------------------------------------------------------

DECLARE @SQLstr nvarchar(200)

DECLARE @empid int

SET @SQLstr = N'Select e.empid ,e.firstname , e.lastname From HR.Employees as e Where e.empid = @empid'

Execute sys.sp\_executesql @statement = @SQLstr,@SQLparam=N'@empid as int' ,@empid = 5

GO

---------------------------------------------------------------------

-- LAB 16

--

-- Exercise 4

---------------------------------------------------------------------

USE TSQL;

GO

---------------------------------------------------------------------

-- Task 1

--

--1-- Write T-SQL code to create a synonym named dbo.Person for the Person.Person table

-- in the AdventureWorks database. Execute the written statement.

--2-- Write a SELECT statement against the dbo.Person synonym and retrieve

-- the FirstName and LastName columns. Execute the SELECT statement .

-- Observe and compare the results that you got with the recommended result

-- shown in the file 82 - Lab Exercise 4 - Task 1 Result.txt.

---------------------------------------------------------------------

--1--

CREATE SYNONYM dbo.Person FOR AdventureWorks.Person.Person

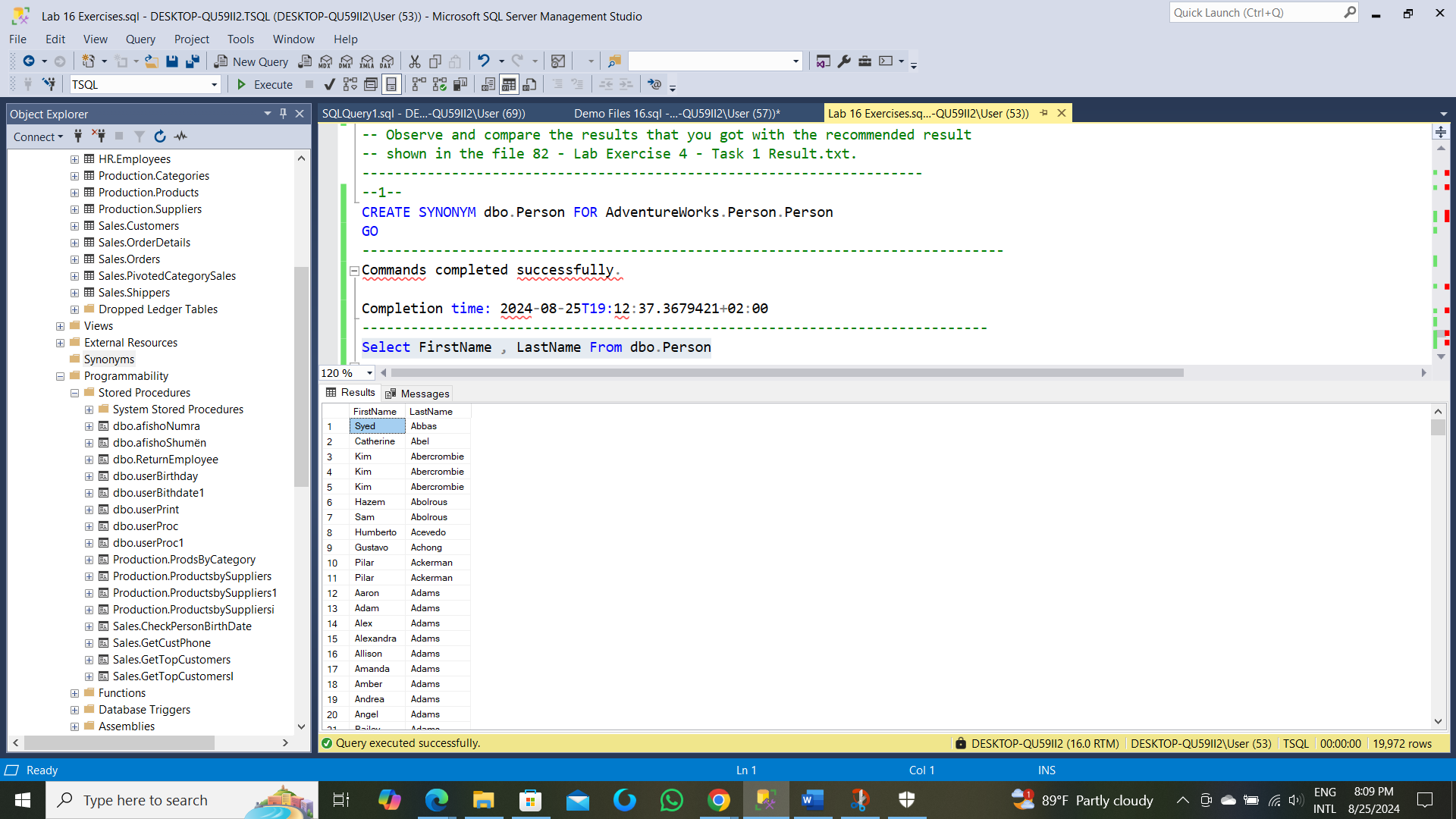
GO

-------------------------------------------------------------------------------

Commands completed successfully.

Completion time: 2024-08-25T19:12:37.3679421+02:00

----------------------------------------------------------------------------- Select FirstName , LastName From dbo.Person



---------------------------------------------------------------------

-- Task 2

--

-- Execute the provided T-SQL code to remove the synonym.

---------------------------------------------------------------------

DROP SYNONYM dbo.Person; ----------------------------------------------------------------------

Commands completed successfully.

Completion time: 2024-08-25T19:22:33.5506232+02:00

-----------------------------------------------------------------------