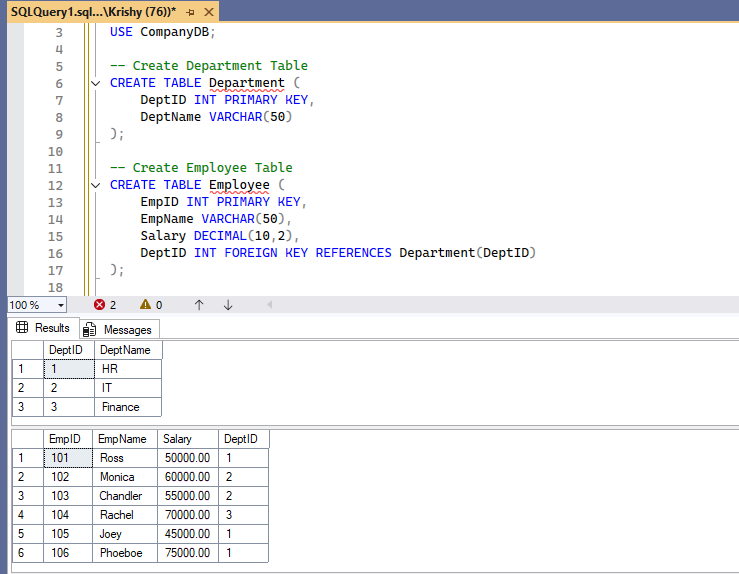
**STORED PROCEDURES AND FUNCTIONS**

**1. Create table and insert values**

Firstly, Created the table Employee and Department as we are using stored procedures and functions to fetch data..



**Stored Procedure**

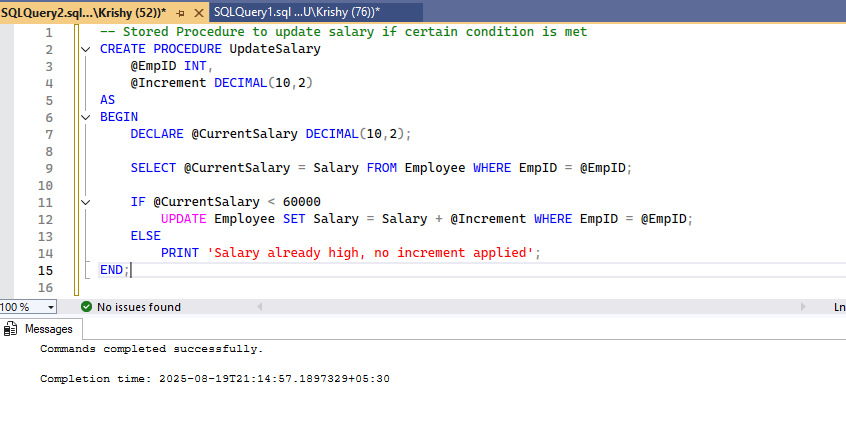
* A block of SQL statements stored in the database.
* Can perform **INSERT, UPDATE, DELETE** operations.
* May or may not return a value.
* Called with CALL or EXEC.

**Steps to be followed**

* Create a procedure
* Declare it
* Call it
* Get the output

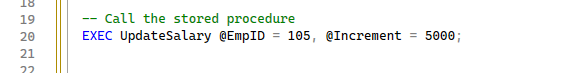
**Step 1-** Create Procedure and declare that

The stored procedure(UpdateSalary) is created for update the salary and we used if else statement inorder to update that.



**Step 2-** Call Procedure to process the logic

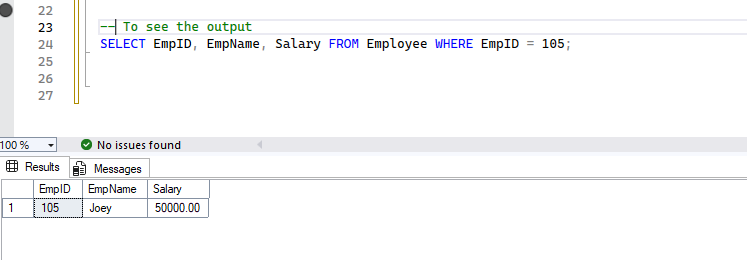
Once we created the Procedure, We have to call that inorder to process the data



We are updating the salary of employee who has Emp id – 105 .

**Step 3-** Get the Output

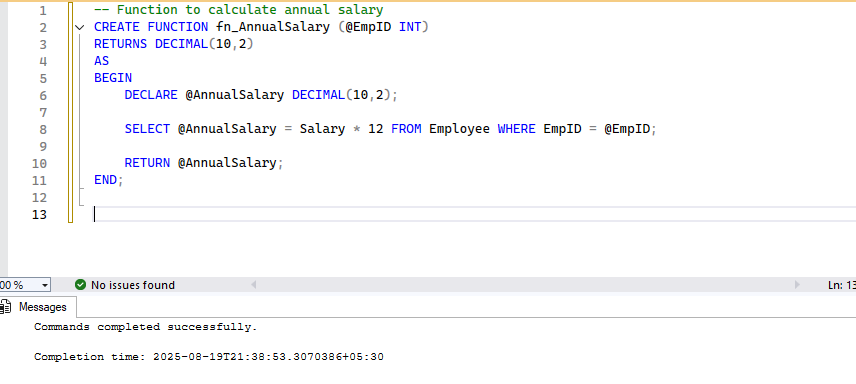
After that to get the result use select statement

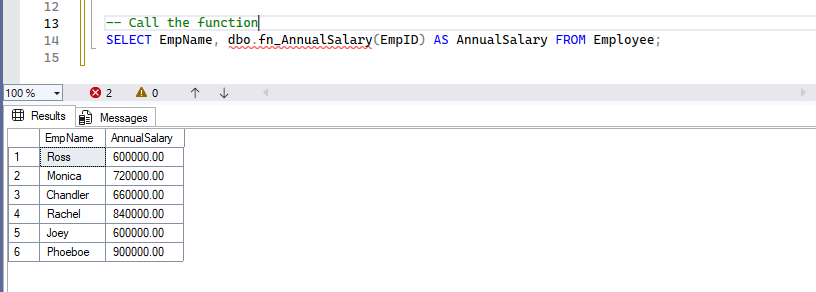


**Function**

* Must return a value (scalar or table).
* Usually used for **computations** or reusable logic in queries.
* Cannot perform transactions (in many RDBMS).
* Called inside SQL statements like SELECT function\_name(...).

The Function(AnnualSalary) is created for to calculate annual salary of each employee.





**Uses**

* Filter results
* Compute values
* Create virtual tables

#### **Difference between a procedure and a function?**

**Procedures:**

1. Do not necessarily return a value. They can use IN, OUT, and INOUT parameters to pass data in and out. They are primarily used for performing actions (e.g., updating data, managing transactions). Procedures are called using EXEC or EXECUTE.

2. Can perform actions (INSERT, UPDATE, DELETE), return multiple values, but cannot be used inside SELECT.

**Functions:**

1. Must return a single value (scalar function) or a table (table-valued function). They can only use IN parameters. They are primarily used for calculations and can be used within SQL statements like SELECT and WHERE.

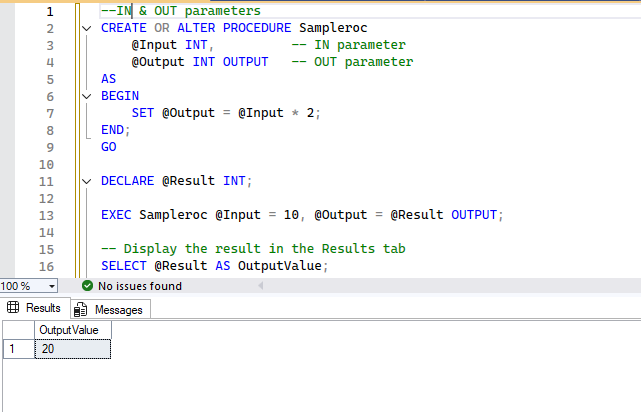
2. Cannot perform major DML (except table variable updates).

**IN/OUT parameter**

**IN Parameter:** The default type. The calling program passes a value to the routine. The routine can use this value but cannot change it in a way that is visible to the caller.

**OUT Parameter:** The routine passes a value back to the caller. The initial value is null inside the routine. The routine can set a new value for the caller to receive.

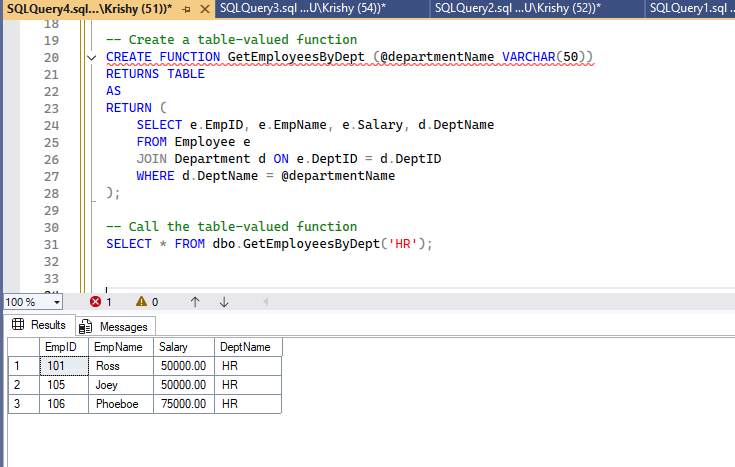
**INOUT Parameter:** The calling program passes a value, and the routine can modify it. The modified value is passed back to the caller.



#### **Can functions return tables?**

Yes, in SQL Server, functions can return tables. These are known as Table-Valued Functions (TVFs). They are excellent for creating reusable, parameterized views.

Here we fetched the employee details by department..



#### **RETURN Statement**

RETURN is used in a function to specify the value that the function will return. In a stored procedure, RETURN is used to exit the procedure immediately and can optionally return an integer status code to the caller.

* In functions → return value.
* In procedures → exit procedure (optional return integer).

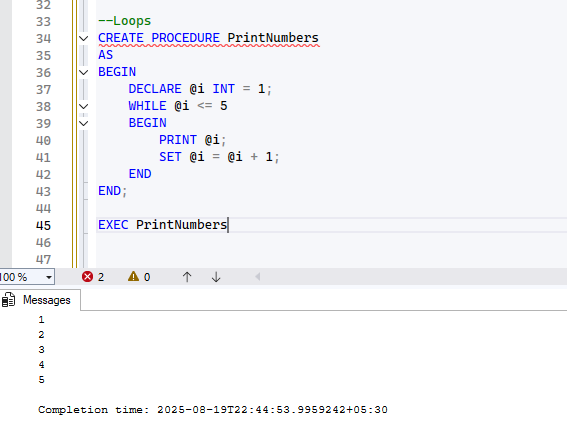
#### **Difference between scalar and table-valued functions?**

**Scalar Functions:** Return a single value (e.g., a number, a string). They can be used anywhere a literal value or expression is valid.

**Table-Valued Functions** (TVFs): Return a table. They are used in the FROM clause of a query, just like a regular table or view.

#### **Can procedures have loops?**

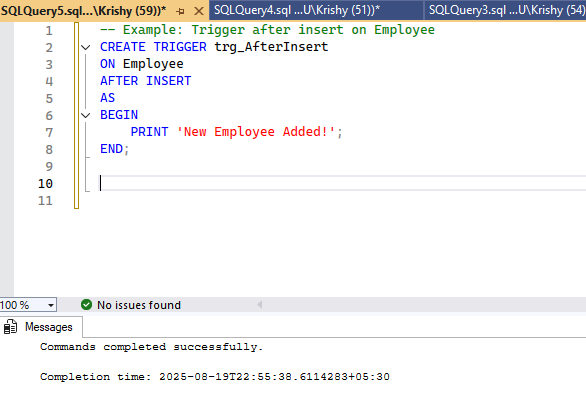
Yes, stored procedures can have loops. SQL Server supports WHILE loops and cursors for iterative processing.

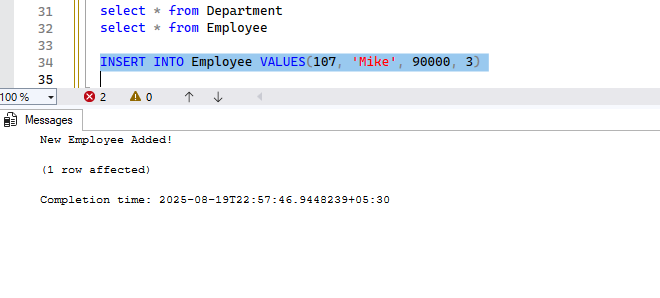


**Triggers**

A trigger is a special type of stored procedure that is automatically executed in response to certain events on a table or view, such as INSERT, UPDATE, or DELETE. Triggers are used to enforce business rules, maintain data integrity, or perform logging.

Created a trigger for the insertion of new record, it will automatically trigger the message “New employee Added!” everytime new records added.

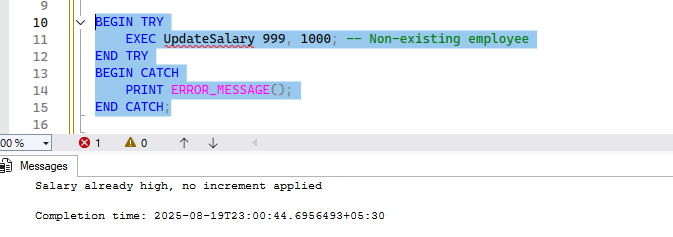




Here you see the message after new record added.

**Debugging stored procedures**

* Use PRINT statements.
* Use TRY...CATCH for error handling.



As Salary greater than 60000 it shows 'Salary already high, no increment applied' which we already coded in while creating Stored Procedures UpdateSalary.

IF @CurrentSalary < 60000

UPDATE Employee SET Salary = Salary + @Increment WHERE EmpID = @EmpID;

ELSE

PRINT 'Salary already high, no increment applied';

END;

Thank you!