Lab # 12: Intro to Stored Procedures, Triggers, Functions and Report Generation in SQL Server

Objective:

To make user defined stored procedures and functions in SQL Server.

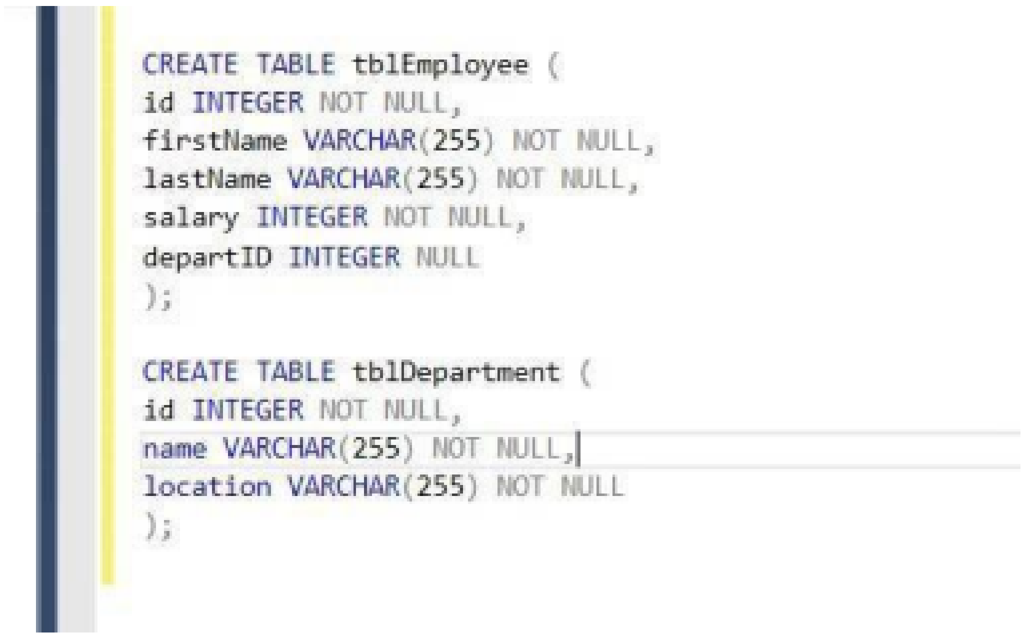
Scope:

The student shall know the following:

- SQL Commands.
- SQL Queries
- Hands-on experience with the above-mentioned concepts.

Discussion:

First, we will study what are Stored procedures and how to crate user defined procedures in SQL server. Lets First Create Database and Tables.



```
CREATE TABLE tblEmployee (
id INTEGER NOT NULL,
firstName VARCHAR(255) NOT NULL,
lastName VARCHAR(255) NOT NULL,
salary INTEGER NOT NULL,
departID INTEGER NULL
);

CREATE TABLE tblDepartment (
id INTEGER NOT NULL,
name VARCHAR(255) NOT NULL,
location VARCHAR(255) NOT NULL
);
```

Now add at least 10 rows in each table above.

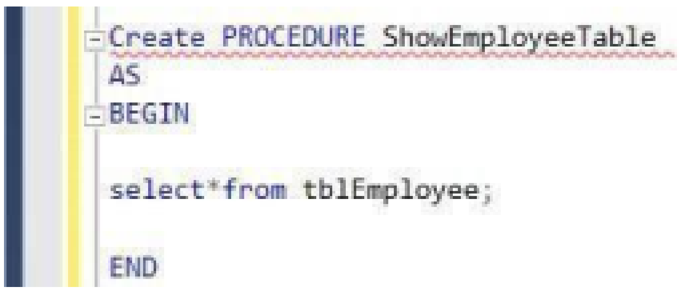
Stored Procedures

A SQL stored procedure (SP) is a collection SQL statements and sql command logic, which is compiled and stored on the database. Stored procedures in SQL allows us to create SQL queries to be stored and executed on the server. Stored procedures can also be cached and reused. The main purpose of stored procedures to hide direct SQL queries from the code and improve performance of database operations such as select, update, and delete data.

User Defined Stored Procedures

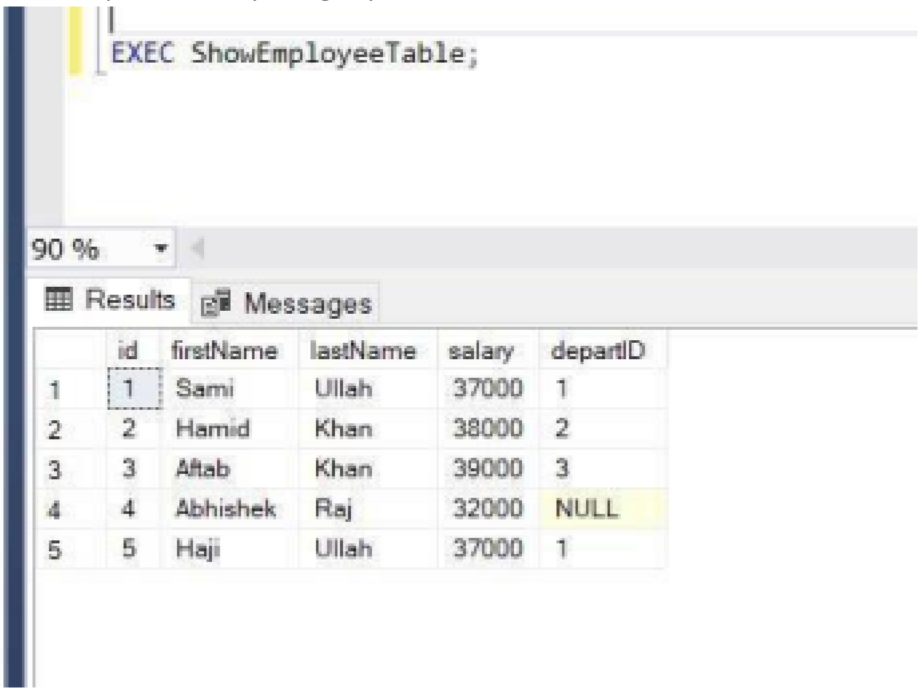
User defined stored procedures are created by database developers or database administrators. These SPs contain one or more SQL statements to select, update, or delete records from database tables. User defined stored procedures can take input parameters and return output parameters. User defined stored procedure is a mixture of DDL (Data Definition Language) and DML (Data Manipulation Language) commands.

Now let's create a simple procedure name “ShowEmployeeTable” which will display all the values in the table.



```
Create PROCEDURE ShowEmployeeTable
AS
BEGIN

select*from tblEmployee;

END
```

Noe lets execute this procedure by using keyword **“EXEC”** as shown below

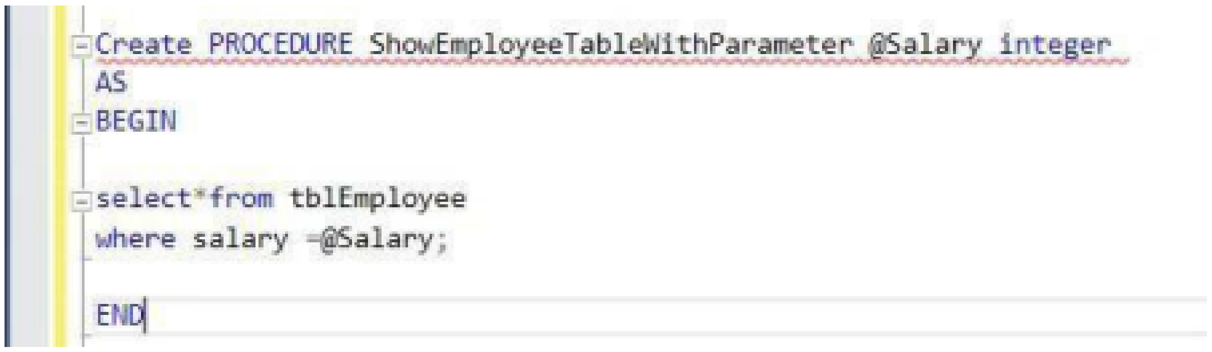


```
EXEC ShowEmployeeTable;
```

	id	firstName	lastName	salary	departID
1	1	Sami	Ullah	37000	1
2	2	Hamid	Khan	38000	2
3	3	Aftab	Khan	39000	3
4	4	Abhishek	Raj	32000	NULL
5	5	Haji	Ullah	37000	1

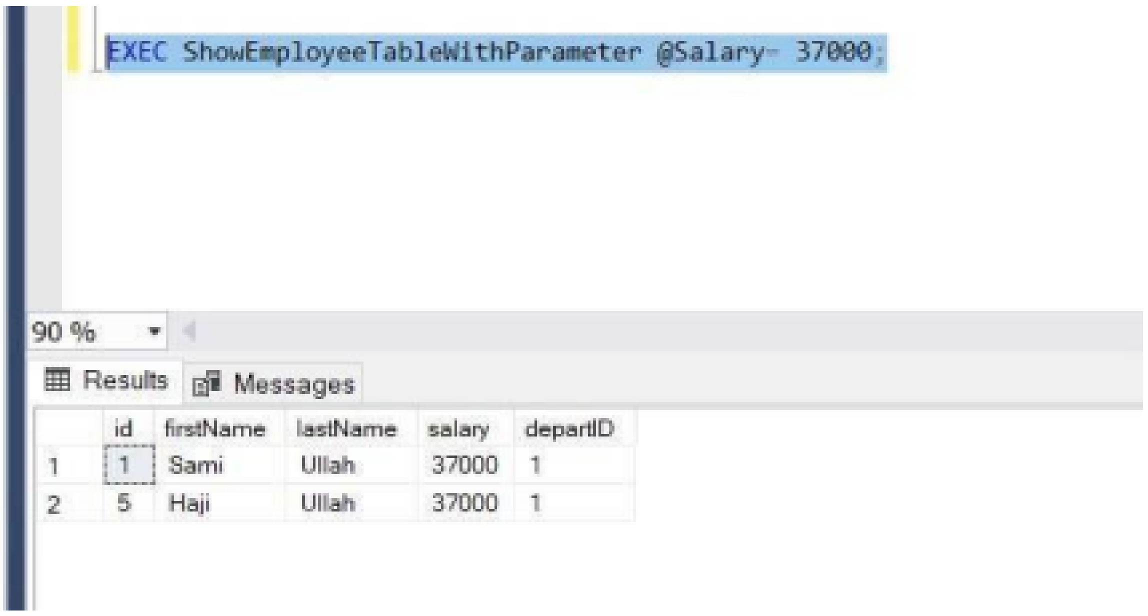
Stored Procedures with One Parameter

Now lets create modify above procedure which will take on parameter and will display values where salary will be equal to the value passed as parameter.



```
Create PROCEDURE ShowEmployeeTableWithParameter @Salary integer
AS
BEGIN

select*from tblEmployee
where salary =@Salary;

END
```

Now the above Procedure takes a parameter declared "@" as prefix. This symbol is used to identify variables in procedures. Now the query will display those employees whose salary is equal to value passed as parameter by user as shown below.



```
EXEC ShowEmployeeTableWithParameter @Salary= 37000;
```

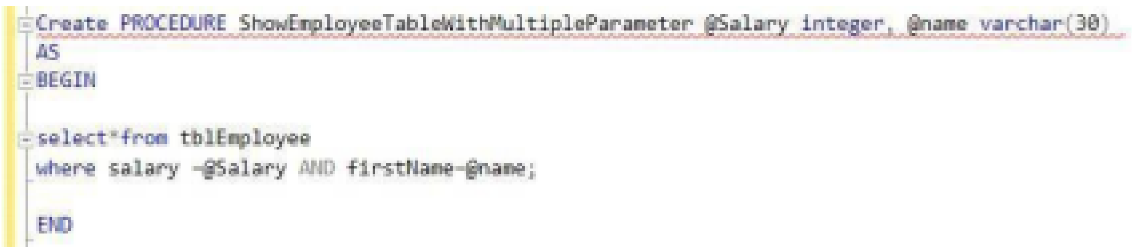
	id	firstName	lastName	salary	departID
1	1	Sami	Ullah	37000	1
2	5	Haji	Ullah	37000	1

To execute a procedure with parameters, as shown above you have to pass value when executing Procedure.

Note: You can delete the procedure by DROP keyword.

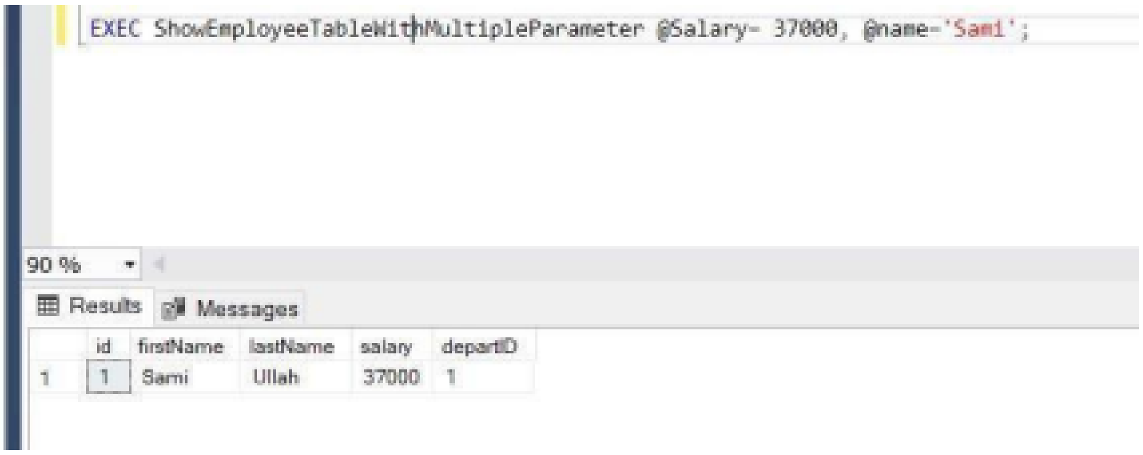
Stored Procedures with Multiple Parameter

You can pass multiple parameters with comma separation as shown below



```
Create PROCEDURE ShowEmployeeTableWithMultipleParameter @Salary integer, @name varchar(30)
AS
BEGIN

select*from tblEmployee
where salary =@Salary AND firstName=@name;

END
```

Result is shown below



```
EXEC ShowEmployeeTableWithMultipleParameter @Salary= 37000, @name='Sami';
```

	id	firstName	lastName	salary	departID
1	1	Sami	Ullah	37000	1

Triggers In SQL

SQL Server triggers are special stored procedures that are executed automatically in response to the database object, database, and server events.

There are two types of Triggers:

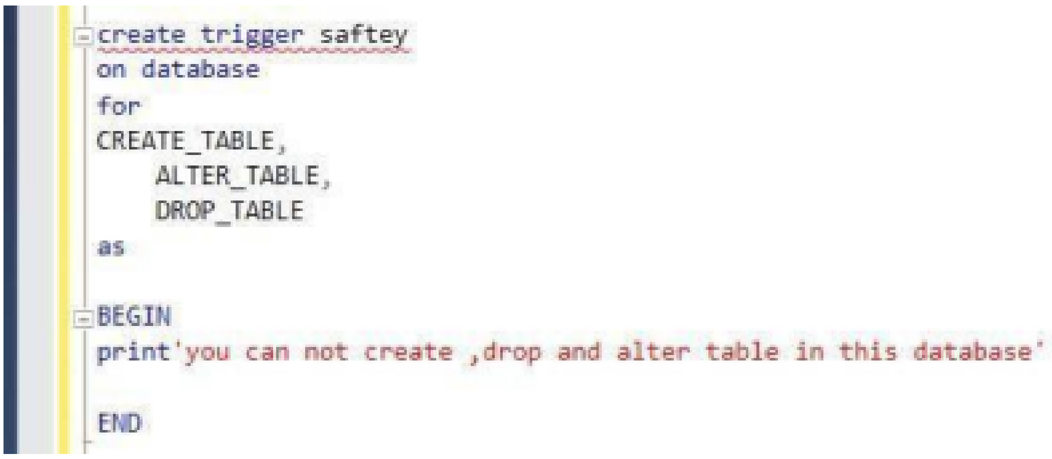
1. DDL Trigger

2. DML Trigger

DDL Trigger

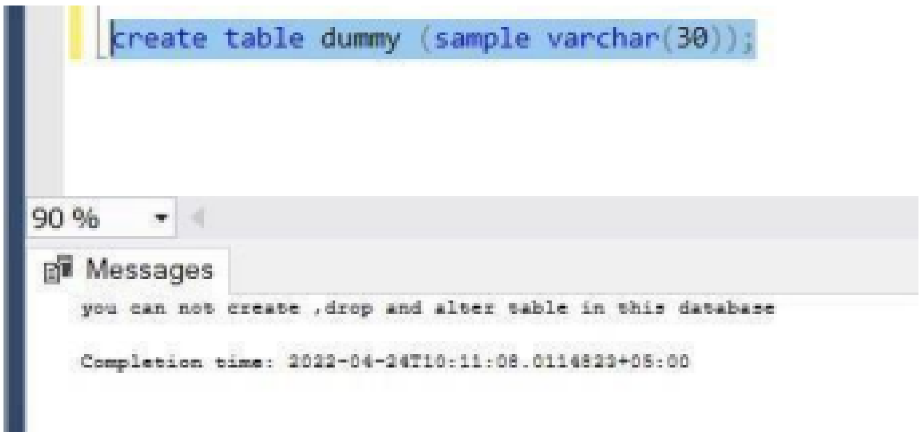
The DDL triggers are fired in response to DDL (Data Definition Language) command events that start with Create, Alter and Drop, such as Create_table, Create_view, drop_table, Drop_view and Alter_table.

For example we want to see trigger on above mentioned events. We want to make safety trigger such that user can not create, delete or alter any table in the database. For that purpose we write a trigger as shown below



```sql
create trigger saftey
on database
for
CREATE_TABLE,
    ALTER_TABLE,
    DROP_TABLE
as

BEGIN
print'you can not create ,drop and alter table in this database'

END
```

Now whenever the user tries to create a table, the user will be shown with the error as mentioned in above trigger. Now let’s check our trigger



```sql
create table dummy (sample varchar(30));
```

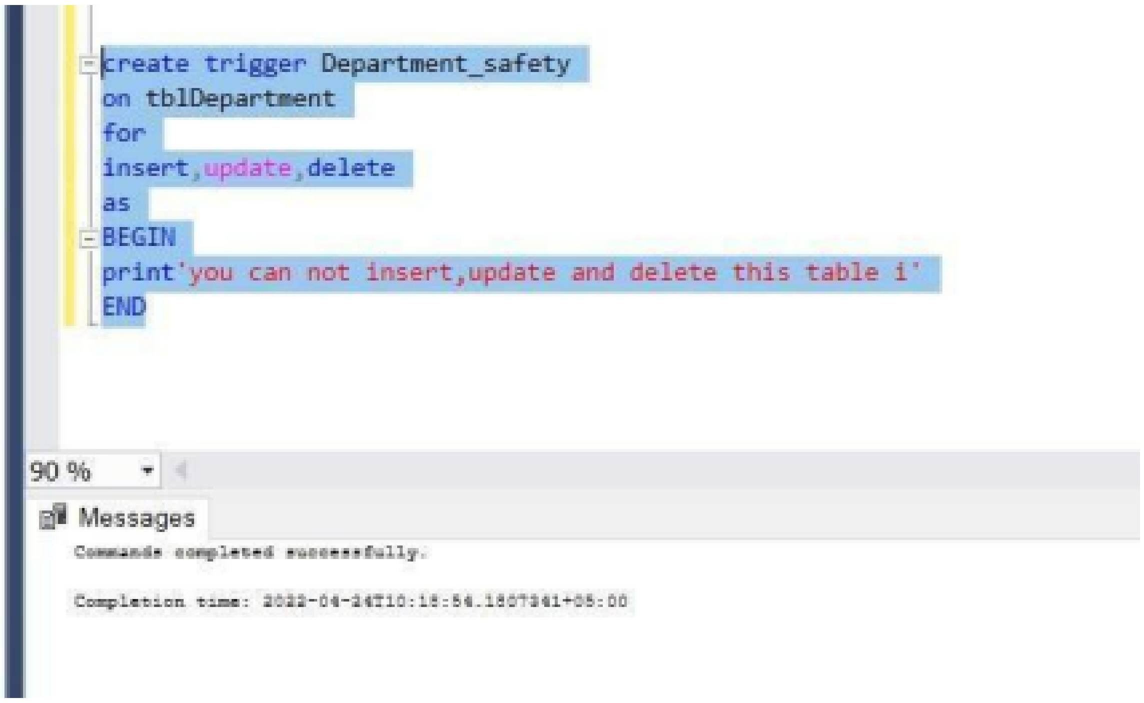
```
you can not create ,drop and alter table in this database

Completion time: 2022-04-24T10:11:08.0114823+05:00
```

Above screenshot shows us with an error when trying to create a table in out database.

DML Trigger

The DML triggers are fired in response to DML (Data Manipulation Language) command events that start with Insert, Update, and Delete. Like insert_table, Update_view and Delete_table.

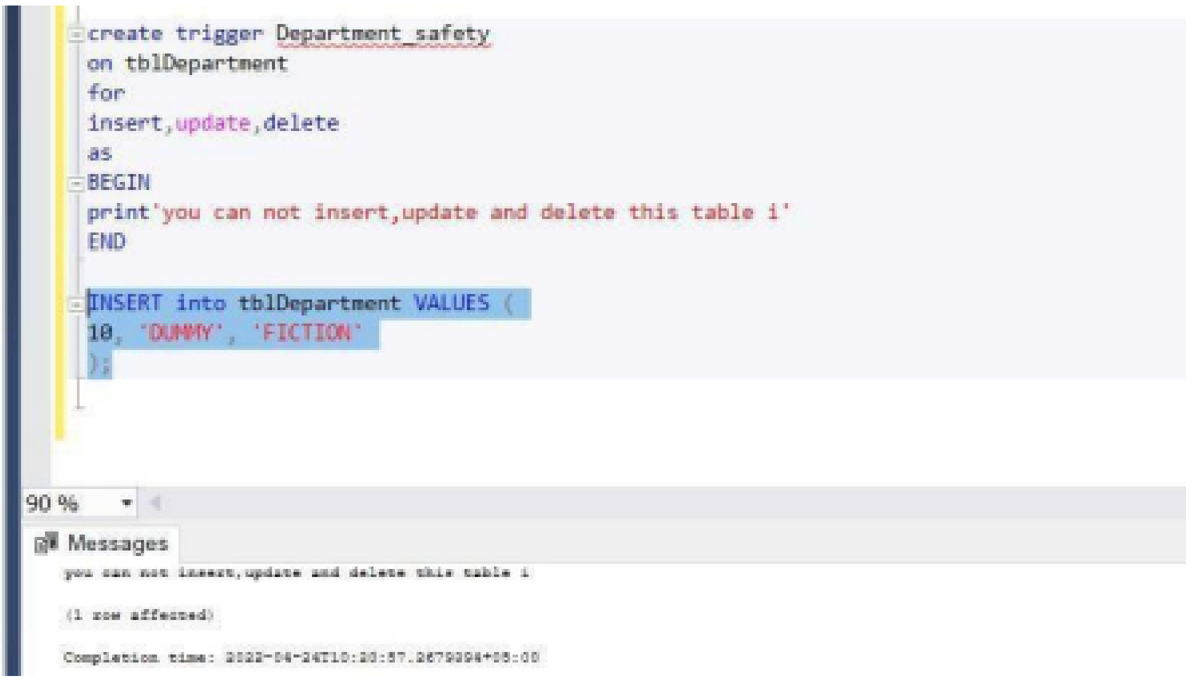
Below is the trigger defined to prevent any operation on the Depart table.



```
create trigger Department_safety
on tblDepartment
for
insert,update,delete
as
BEGIN
print'you can not insert,update and delete this table i'
END
```

```
Commands completed successfully.

Completion time: 2022-04-24T10:18:54.1807341+05:00
```

Now lets test our trigger as shown below



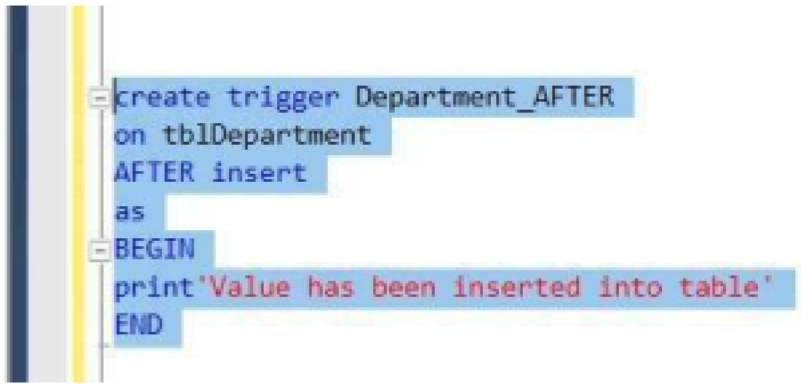
```
create trigger Department_safety
on tblDepartment
for
insert,update,delete
as
BEGIN
print'you can not insert,update and delete this table i'
END

INSERT into tblDepartment VALUES (
10, 'DUMMY', 'FICTION'
);
```

```
you can not insert,update and delete this table i

(1 row affected)

Completion time: 2022-04-24T10:20:57.2679294+05:00
```

Note as mentioned in definition, it is triggered in response to DML command. That is why the command has been also executed and data has been inserted. And at the same time a message has also been displayed. To prevent command execution we have further two types of **DML triggers**

AFTER Trigger

AFTER triggers are executed after the action of an INSERT, UPDATE, or DELETE statement.



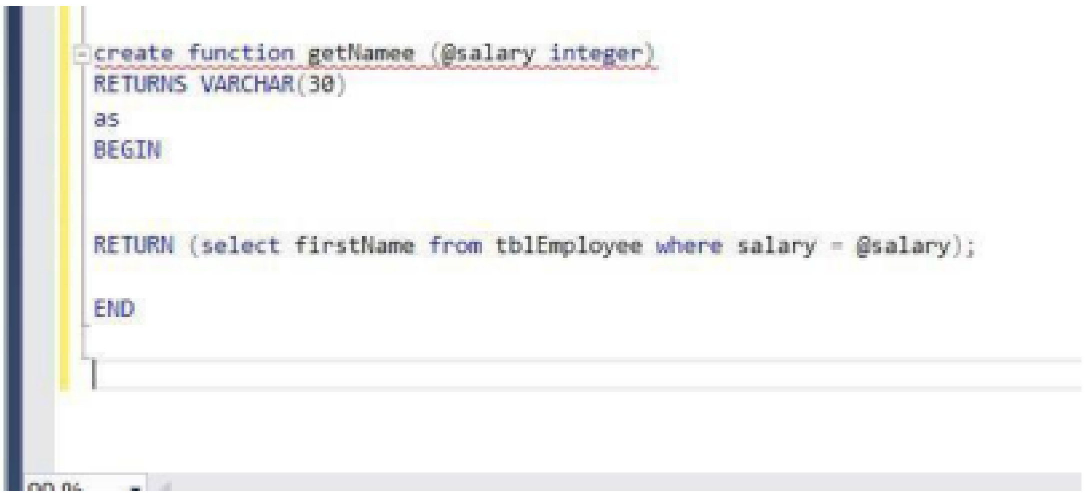
```
create trigger Department_AFTER
on tblDepartment
AFTER insert
as
BEGIN
print'Value has been inserted into table'
END
```

Above trigger is executed after insertion query.

CREATING FUNCTIONS IN SQL

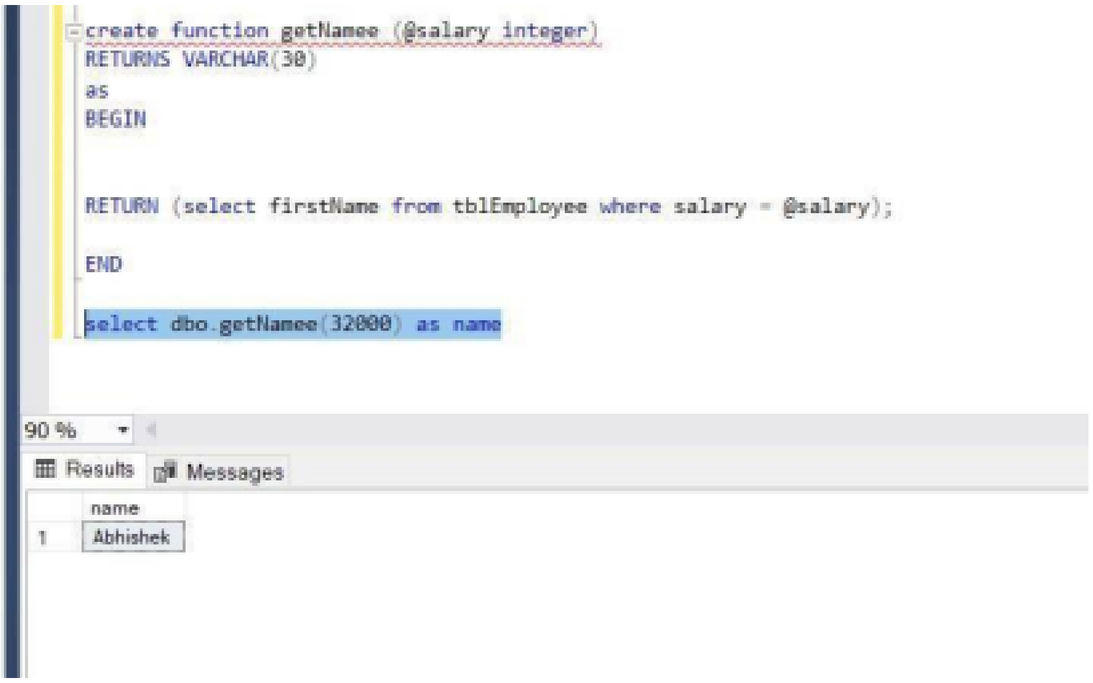
Lets create a simple function that takes integer value as argument and returns string.

Below is the example of a function that takes salary as argument and returns the name of the person who has salary equal to the passed value.



```
create function getNamee (@salary integer)
RETURNS VARCHAR(30)
as
BEGIN


RETURN (select firstName from tblEmployee where salary = @salary);

END
```

Now let’s execute the function to test the result.



```
create function getNamee (@salary integer)
RETURNS VARCHAR(30)
as
BEGIN


RETURN (select firstName from tblEmployee where salary = @salary);

END

select dbo.getNamee(32000) as name
```

```
   name
1  Abhishek
```