

Name:- Atish Kumar

Roll No:- 120CS0173

Date:- 20 April,2022

DE Lab Assignment 08

Q 1: Write SQL queries to perform creation of table, insert values, select and drop for the following instruction.

- 1. Create a table Employee having attribute Employee_ID, Name, Age, Salary, Department. Insert values in Employee table as shown & display all the values.**

Program:

```
CREATE DATABASE LAB08
```

```
USE LAB
```

```
CREATE TABLE Employees  
(  
Employee_ID VARCHAR(50),  
Name VARCHAR(50),  
Age INT,  
Salary INT,  
Department VARCHAR(50)  
);
```

```
INSERT INTO Employees VALUES('1','Adiya',25,30000,'Executive');  
INSERT INTO Employees VALUES('2','Aditya',28,31000,'Accountant');  
INSERT INTO Employees VALUES('3','Priyanka',30,32000,'Salesman');  
INSERT INTO Employees VALUES('4','Anmol',35,31000,'Executive');  
INSERT INTO Employees VALUES('5','Rahul',31,29000,'Accountant');  
INSERT INTO Employees VALUES('6','Shubham',29,27000,'Salesman');  
INSERT INTO Employees VALUES('7','Sam',33,32000,'Salesman');  
INSERT INTO Employees VALUES('8','Rohan',33,28000,'Salesman');  
INSERT INTO Employees VALUES('9','Priya',23,35000,'Accountant');  
INSERT INTO Employees VALUES('10','Kabir',29,26000,'Executive');
```

```
SELECT*FROM Employees;
```

Output:

The screenshot displays a SQL Server Enterprise Manager window with the following content:

SQLQuery3.sql - UN...known Hector (57))*

```
INSERT INTO Employees VALUES ('1', 'Adiya', 25, 30000, 'Executive');
INSERT INTO Employees VALUES ('2', 'Aditya', 28, 31000, 'Accountant');
INSERT INTO Employees VALUES ('3', 'Priyanka', 30, 32000, 'Salesman');
INSERT INTO Employees VALUES ('4', 'Anmol', 35, 31000, 'Executive');
INSERT INTO Employees VALUES ('5', 'Rahul', 31, 29000, 'Accountant');
INSERT INTO Employees VALUES ('6', 'Shubham', 29, 27000, 'Salesman');
INSERT INTO Employees VALUES ('7', 'Sam', 33, 32000, 'Salesman');
INSERT INTO Employees VALUES ('8', 'Rohan', 33, 28000, 'Salesman');
INSERT INTO Employees VALUES ('9', 'Priya', 23, 35000, 'Accountant');
INSERT INTO Employees VALUES ('10', 'Kabir', 29, 26000, 'Executive');

SELECT * FROM Employees;
```

Results

| | Employee_ID | Name | Age | Salary | Department |
|----|-------------|----------|-----|--------|------------|
| 1 | 1 | Adiya | 25 | 30000 | Executive |
| 2 | 2 | Aditya | 28 | 31000 | Accountant |
| 3 | 3 | Priyanka | 30 | 32000 | Salesman |
| 4 | 4 | Anmol | 35 | 31000 | Executive |
| 5 | 5 | Rahul | 31 | 29000 | Accountant |
| 6 | 6 | Shubham | 29 | 27000 | Salesman |
| 7 | 7 | Sam | 33 | 32000 | Salesman |
| 8 | 8 | Rohan | 33 | 28000 | Salesman |
| 9 | 9 | Priya | 23 | 35000 | Accountant |
| 10 | 10 | Kabir | 29 | 26000 | Executive |

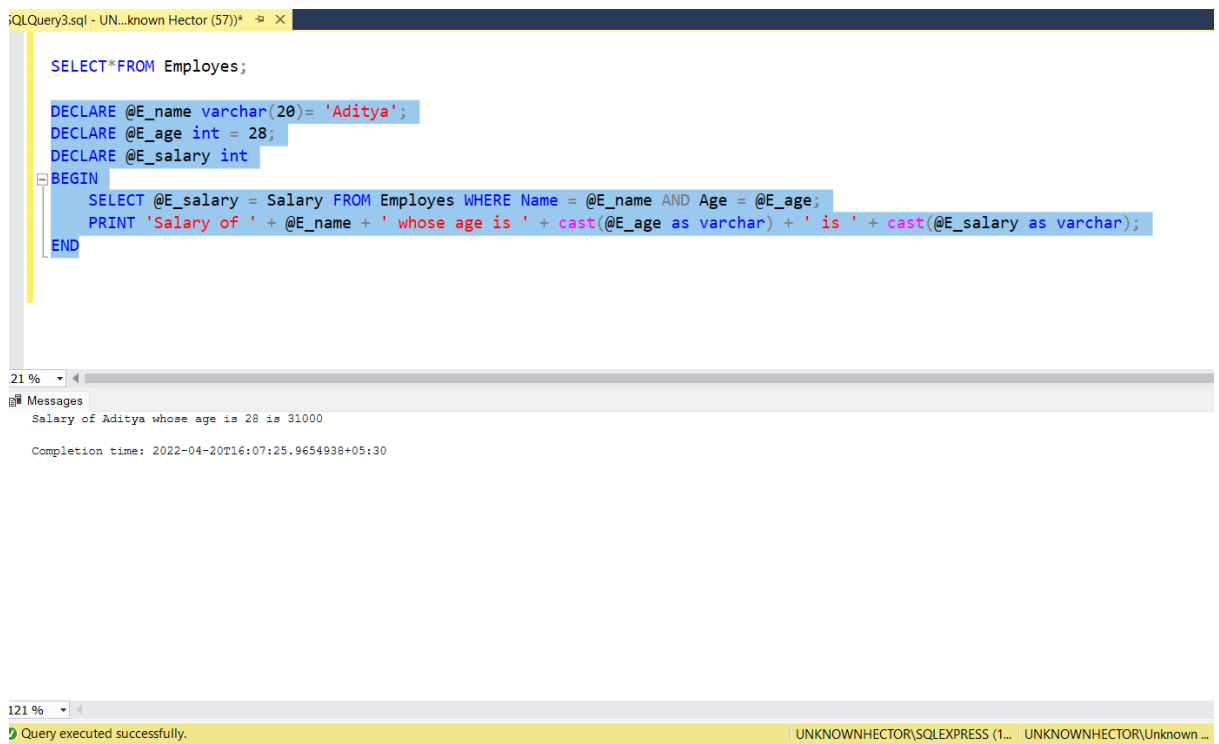
Query executed successfully. UNKNOWNHECTOR\SQLEXPRESS (1...

2. Create a PL/SQL block which displays the salary earned by Employee.Name="Aditya" and Employee.Age="28".

Program:

```
DECLARE @E_name varchar(20)= 'Aditya';
DECLARE @E_age int = 28;
DECLARE @E_salary int
BEGIN
    SELECT @E_salary = Salary FROM Employees WHERE Name = @E_name AND Age = @E_age;
    PRINT 'Salary of ' + @E_name + ' whose age is ' + cast(@E_age as varchar) + '
is ' + cast(@E_salary as varchar);
END
```

Output:



The screenshot shows a SQL Server Enterprise Manager window with a query executed in the 'Messages' pane. The query is a PL/SQL block that declares variables for an employee's name, age, and salary, then selects the salary for an employee named 'Aditya' who is 28 years old, and prints the result.

```
SELECT*FROM Employees;

DECLARE @E_name varchar(20)= 'Aditya';
DECLARE @E_age int = 28;
DECLARE @E_salary int
BEGIN
    SELECT @E_salary = Salary FROM Employees WHERE Name = @E_name AND Age = @E_age;
    PRINT 'Salary of ' + @E_name + ' whose age is ' + cast(@E_age as varchar) + ' is ' + cast(@E_salary as varchar);
END
```

21 %

Messages

Salary of Aditya whose age is 28 is 31000

Completion time: 2022-04-20T16:07:25.9654938+05:30

121 %

Query executed successfully. UNKNOWNHECTOR\SQLEXPRESS (1... UNKNOWNHECTOR\Unknown ...

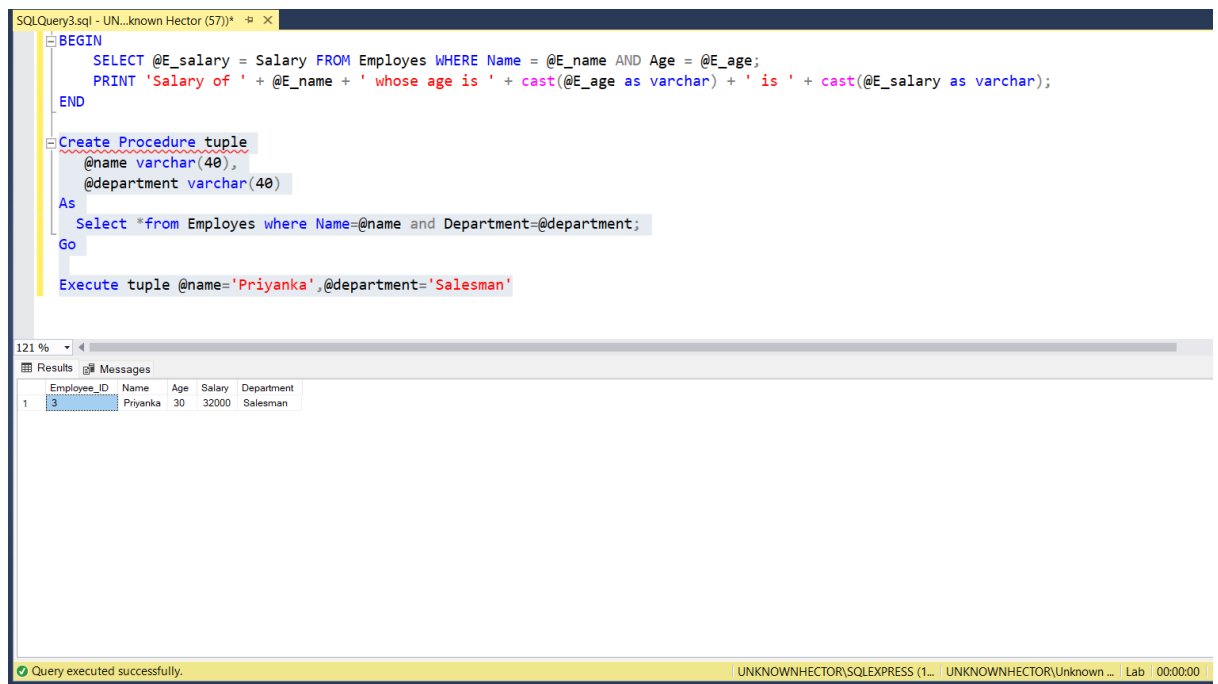
3. Create a PL/SQL procedure that takes parameters as Employee.Name and Employee.Department and displays the tuple that has been passed as the parameter.

Program:

```
Create Procedure tuple
    @name varchar(40),
    @department varchar(40)
As
    Select *from Employees where Name=@name and Department=@department;
Go

Execute tuple @name='Priyanka',@department='Salesman'
```

Output:



```
SQLQuery3.sql - UN...known Hector (57) * X
BEGIN
    SELECT @E_salary = Salary FROM Employees WHERE Name = @E_name AND Age = @E_age;
    PRINT 'Salary of ' + @E_name + ' whose age is ' + cast(@E_age as varchar) + ' is ' + cast(@E_salary as varchar);
END

Create Procedure tuple
    @name varchar(40),
    @department varchar(40)
As
    Select *from Employees where Name=@name and Department=@department;
Go

Execute tuple @name='Priyanka',@department='Salesman'
```

121 %

Results Messages

| | Employee_ID | Name | Age | Salary | Department |
|---|-------------|----------|-----|--------|------------|
| 1 | 3 | Priyanka | 30 | 32000 | Salesman |

Query executed successfully. UNKNOWNHECTOR\SQLEXPRESS (1... UNKNOWNHECTOR\Unknown ... Lab 00:00:00

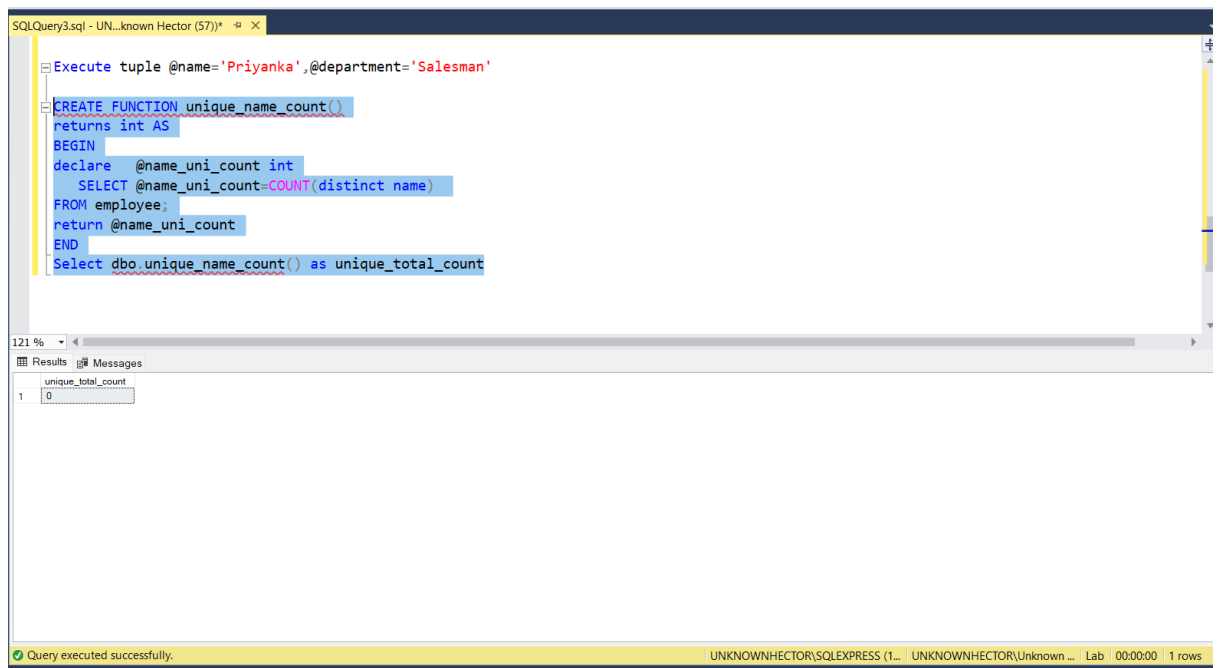
4. Create a PL/SQL function that displays a unique count of attribute Employee.Name.

Program:

```
CREATE FUNCTION unique_name_count()
returns int AS
BEGIN
declare    @name_uni_count int
    SELECT @name_uni_count=COUNT(distinct name)
FROM employee;
return @name_uni_count
END

Select dbo.unique_name_count() as unique_total_count
```

Output:



5. Create a PL/SQL cursor that displays Employee.Employee_ID, Employee.Name and Employee.Department using FETCH NEXT and deallocate it.

Program:

[illegible]

Output:

SQLQuery3.sql - UN...known Hector (57)*

```
FETCH NEXT FROM my_cursor;  
FETCH NEXT FROM my_cursor;  
FETCH NEXT FROM my_cursor;  
FETCH NEXT FROM my_cursor;  
DEALLOCATE my_cursor;
```

121 %

Results Messages

| | Employee_id | Name | Department |
|---|-------------|----------|------------|
| 1 | 1 | Aditya | Executive |
| 1 | 2 | Aditya | Accountant |
| 1 | 3 | Priyanka | Salesman |
| 1 | 4 | Anmol | Executive |
| 1 | 5 | Rahul | Accountant |
| 1 | 6 | Shubham | Salesman |
| 1 | 7 | Sam | Salesman |
| 1 | 8 | Rohan | Salesman |
| 1 | 9 | Priya | Accountant |
| 1 | 10 | Kabir | Executive |

Query executed successfully. UNKNOWNHECTOR\SQLEXPRESS (1... UNKNOWNHECTOR\Unknown ... Lab 00:00:00 10 rows

SQLQuery3.sql - UN...known Hector (57)*

```
FETCH NEXT FROM my_cursor;  
FETCH NEXT FROM my_cursor;  
FETCH NEXT FROM my_cursor;  
FETCH NEXT FROM my_cursor;  
DEALLOCATE my_cursor;
```

121 %

Messages

Commands completed successfully.

Completion time: 2022-04-20T16:31:18.3023347+05:30

121 %

Query executed successfully. UNKNOWNHECTOR\SQLEXPRESS (1... UNKNOWNHECTOR\Unknown ...

6. Create a PL/SQL block that creates a cursor inside it , prints all tuples for attributes Employee.Employee_ID, Employee.Name and Employee.Salary , close it and later deallocate it.

Program:

```
Declare @v int
begin

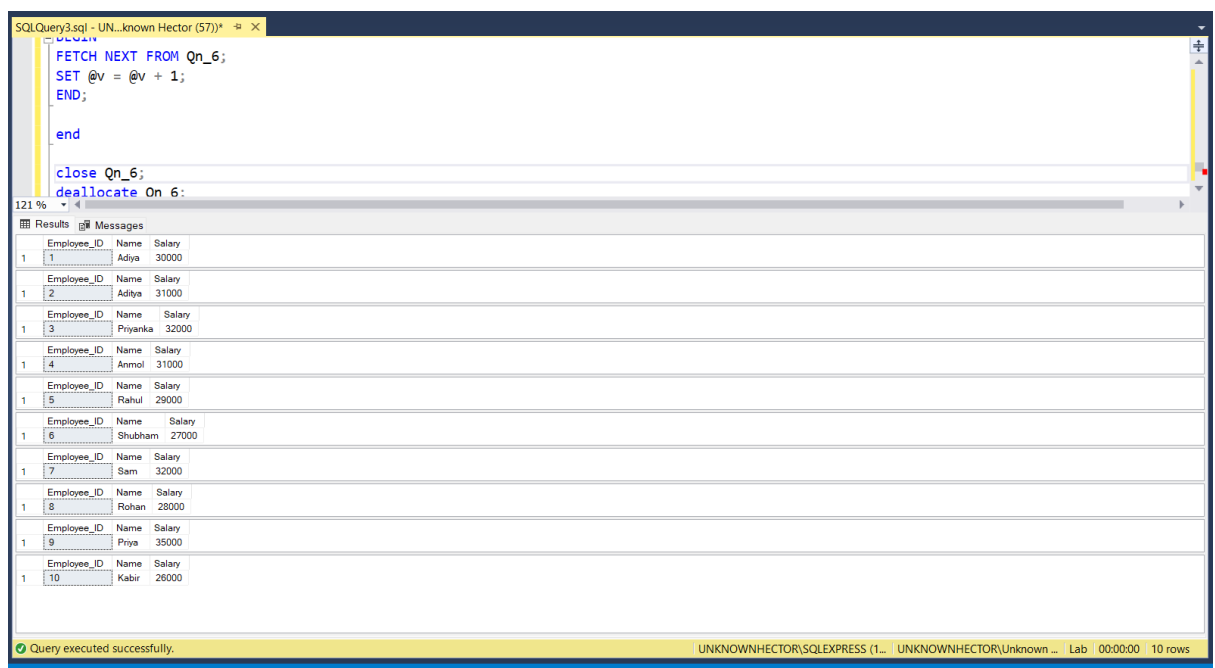
DECLARE Qn_6 CURSOR FOR SELECT Employee_ID,Name,Salary FROM Employees;
OPEN Qn_6;
SET @v = 0;
WHILE @v < 10
BEGIN
FETCH NEXT FROM Qn_6;
SET @v = @v + 1;
END;

end

close Qn_6;

Deallocate Qn_6;
```

Output:



SQLQuery3.sql - UN...known Hector (57)*

```
DECLARE
FETCH NEXT FROM Qn_6;
SET @v = @v + 1;
END;

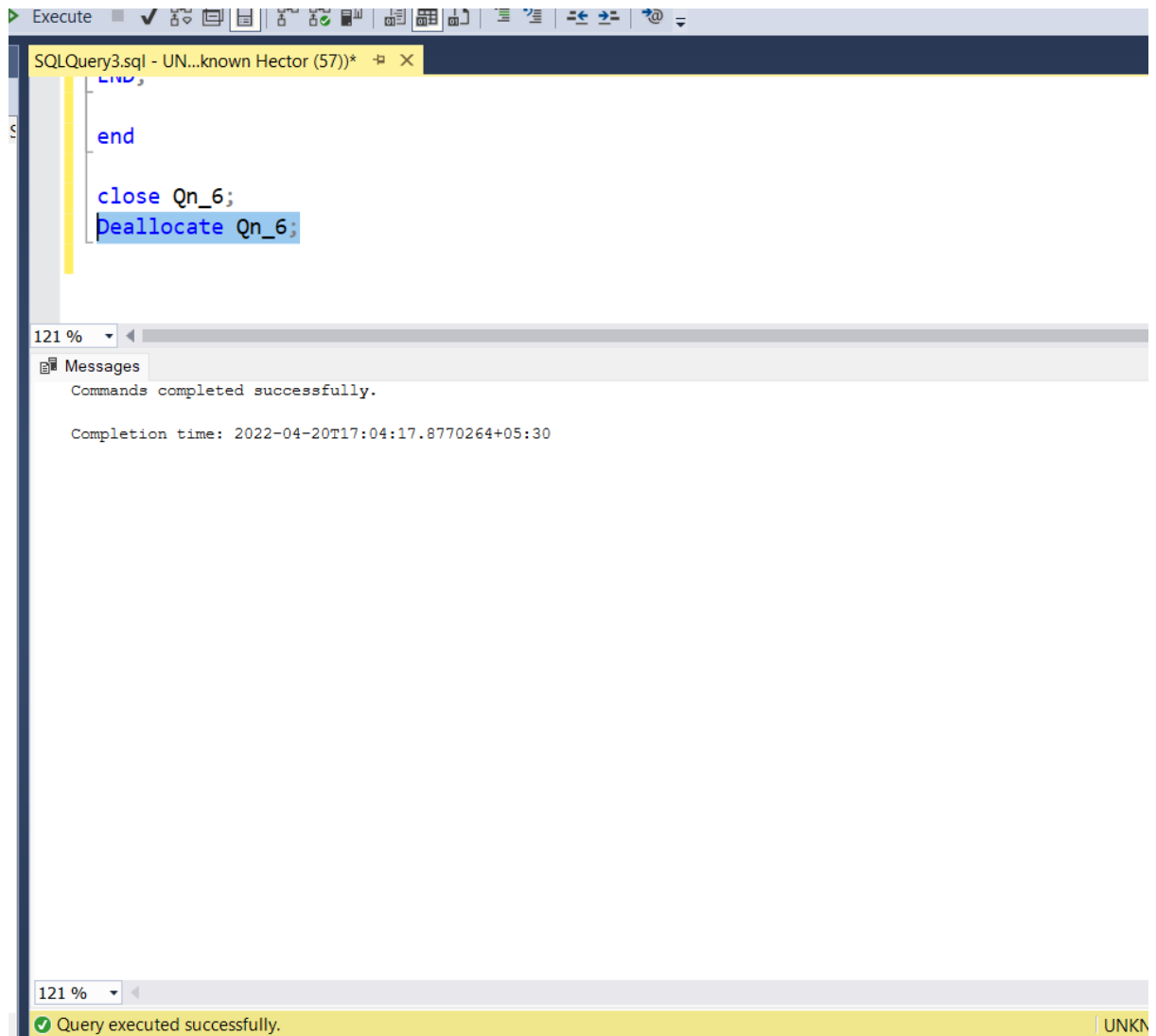
end

close Qn_6;
deallocate On 6:
```

121 %

| | Employee_ID | Name | Salary |
|---|-------------|----------|--------|
| 1 | 1 | Aditya | 30000 |
| 1 | 2 | Aditya | 31000 |
| 1 | 3 | Priyanka | 32000 |
| 1 | 4 | Anmol | 31000 |
| 1 | 5 | Rahul | 29000 |
| 1 | 6 | Shubham | 27000 |
| 1 | 7 | Sam | 32000 |
| 1 | 8 | Rohan | 28000 |
| 1 | 9 | Priya | 35000 |
| 1 | 10 | Kabir | 26000 |

Query executed successfully. UNKNOWNHECTOR\SQLEXPRESS (1... UNKNOWNHECTOR\Unknown ... Lab | 00:00:00 | 10 rows



7. Create a PL/SQL try-catch exception block in which the try block is performing “divided by 0” operation and catch block displays Error_number, Error_Severity, Error_State, Error_Procedure, Error_Line and Error_Message.

Program:

```
BEGIN TRY  
DECLARE @num int;  
SET @num = 1/0;  
END TRY  
BEGIN CATCH  
SELECT  
    ERROR_NUMBER() AS Error_Number,  
    ERROR_SEVERITY() AS Error_Severity,  
    ERROR_STATE() AS Error_State,  
    ERROR_PROCEDURE() AS Error_Procedure,  
    ERROR_LINE() AS Error_Line,  
    ERROR_MESSAGE() AS Error_Message;  
END CATCH;
```

Output:


```
SQLQuery3.sql - UN...known Hector (57)) *  X
close Qn_6;
Deallocate Qn_6;

BEGIN TRY
DECLARE @num int;
SET @num = 1/0;
END TRY
BEGIN CATCH
SELECT
    ERROR_NUMBER() AS Error_Number,
    ERROR_SEVERITY() AS Error_Severity,
    ERROR_STATE() AS Error_State,
    ERROR_PROCEDURE() AS Error_Procedure,
    ERROR_LINE() AS Error_Line,
    ERROR_MESSAGE() AS Error_Message;
END CATCH;
```

121 %

Results Messages

| | Error_Number | Error_Severity | Error_State | Error_Procedure | Error_Line | Error_Message |
|---|--------------|----------------|-------------|-----------------|------------|-----------------------------------|
| 1 | 8134 | 16 | 1 | NULL | 3 | Divide by zero error encountered. |

Query executed successfully. UNKNOWNHECTOR\SQLEXPRESS (1... | UN

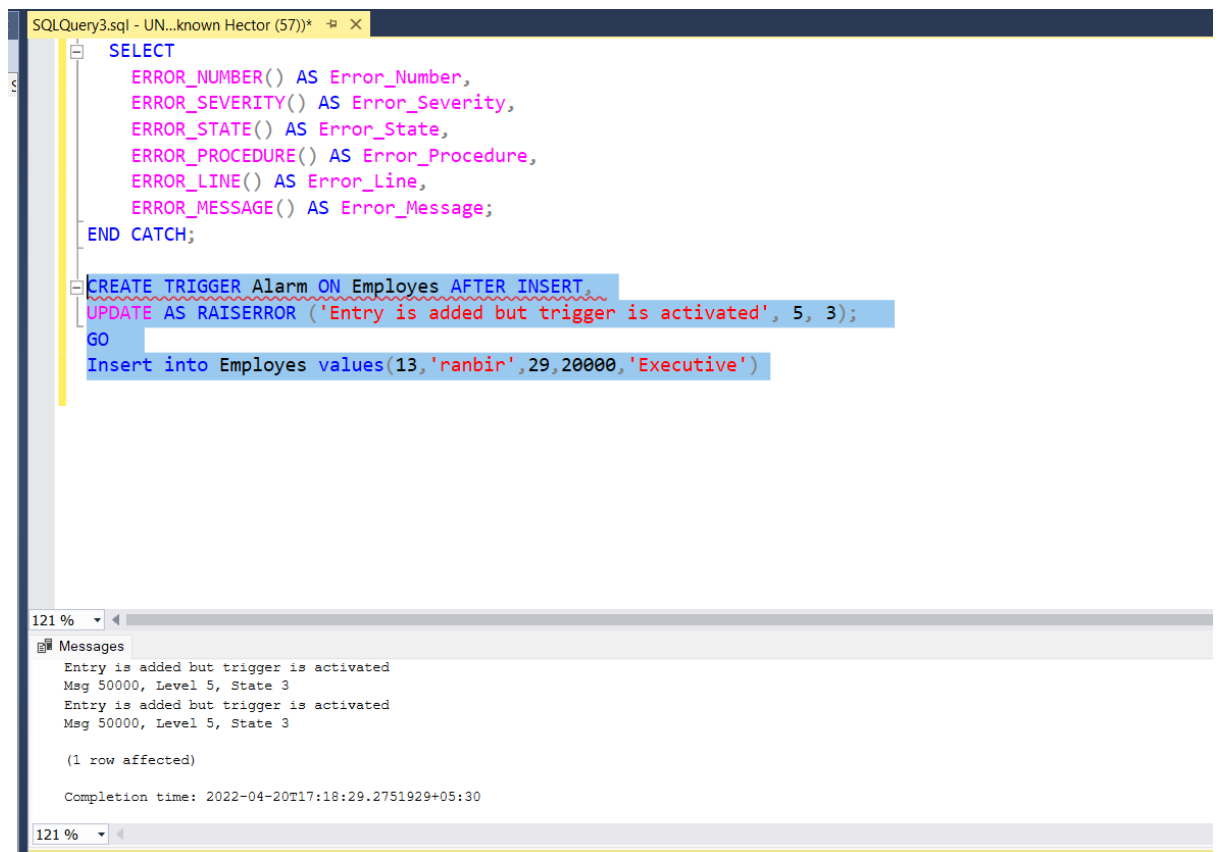
8. Create a PL/SQL trigger that Raise an Error on insert operation of level 5 , state 3 which displays a message “Entry is added but trigger is activated” after insert operation is performed.

Program:

```
CREATE TRIGGER Alarm ON Employes AFTER INSERT,
UPDATE AS RAISERROR ('Entry is added but trigger is activated', 5, 3);
GO

Insert into Employes values(13,'ranbin',29,20000,'Executive')
```

Output:



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a query window titled 'SQLQuery3.sql - UN...known Hector (57))' containing the following SQL code:

```
SELECT
    ERROR_NUMBER() AS Error_Number,
    ERROR_SEVERITY() AS Error_Severity,
    ERROR_STATE() AS Error_State,
    ERROR_PROCEDURE() AS Error_Procedure,
    ERROR_LINE() AS Error_Line,
    ERROR_MESSAGE() AS Error_Message;
END CATCH;

CREATE TRIGGER Alarm ON Employees AFTER INSERT
UPDATE AS RAISERROR ('Entry is added but trigger is activated', 5, 3);
GO
Insert into Employees values(13,'ranbir',29,20000,'Executive')
```

The bottom pane shows the 'Messages' window, which displays the following output:

```
Entry is added but trigger is activated
Msg 50000, Level 5, State 3
Entry is added but trigger is activated
Msg 50000, Level 5, State 3

(1 row affected)

Completion time: 2022-04-20T17:18:29.2751929+05:30
```

9. Delete the trigger created in question 8.

Program:

```
If object_id('inserterror', 'TR') is not null
drop trigger inserterror;
```

Output:

```
SQLQuery3.sql - UN...known Hecor (57)) * X
SELECT
    ERROR_NUMBER() AS Error_Number,
    ERROR_SEVERITY() AS Error_Severity,
    ERROR_STATE() AS Error_State,
    ERROR_PROCEDURE() AS Error_Procedure,
    ERROR_LINE() AS Error_Line,
    ERROR_MESSAGE() AS Error_Message;
END CATCH;

CREATE TRIGGER Alarm ON Employees AFTER INSERT,
UPDATE AS RAISERROR ('Entry is added but trigger is activated', 5, 3);
GO
Insert into Employees values(13,'ranbin',29,20000,'Executive')

If object_id('inserterror', 'TR') is not null
drop trigger inserterror;
```

121 %

Messages

Commands completed successfully.

Completion time: 2022-04-20T17:22:24.8016464+05:30

121 %

Query executed successfully. UNKNOWNHECTOR\SQLEXPRESS (1... UNKNOWNHECTOR\Unknown ... Lab 00:00:00

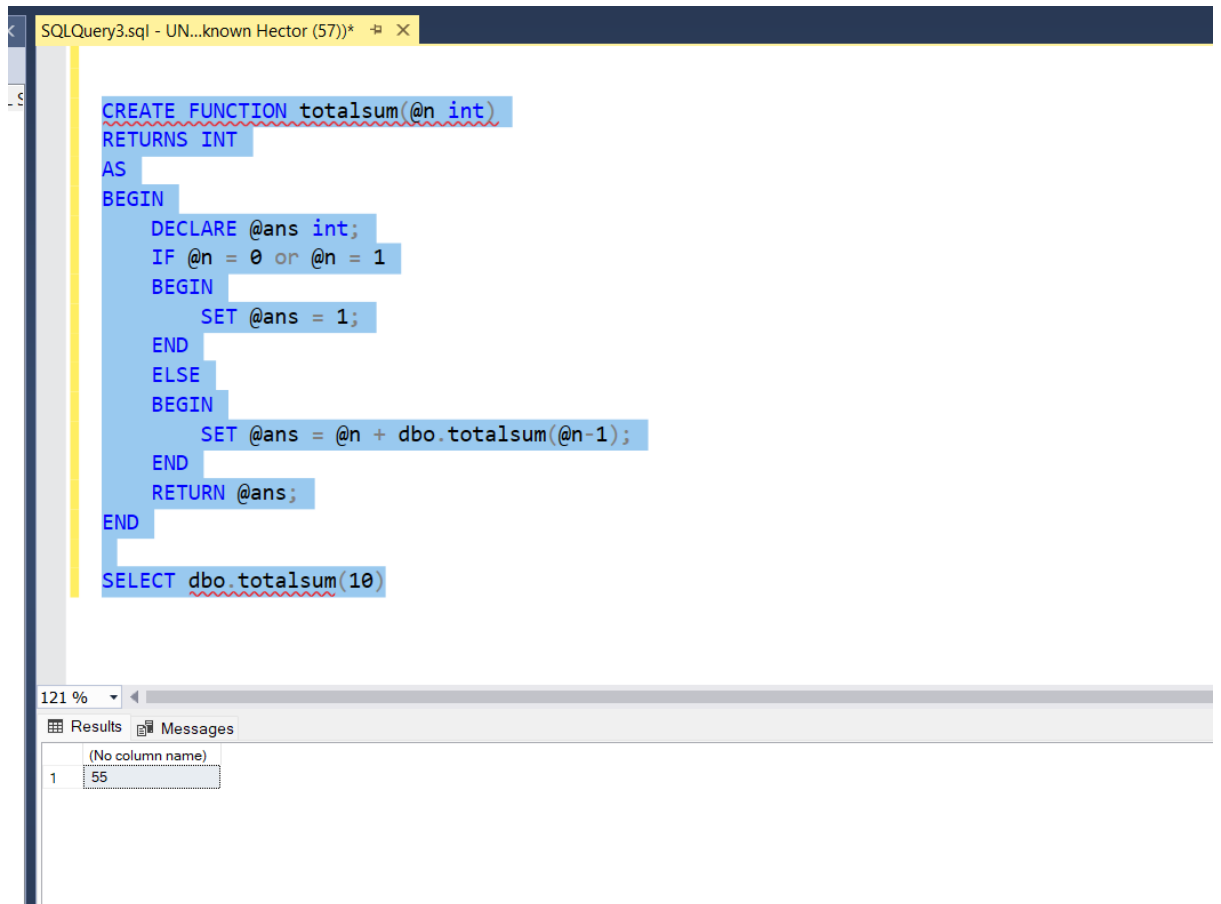
10. Create a PL/SQL recursive function that takes a number “n” as parameter and displays the sum “n+(n-1)+(n-2)+.....+2+1

Program:

```
CREATE FUNCTION totalsum(@n int)
RETURNS INT
AS
BEGIN
    DECLARE @ans int;
    IF @n = 0 or @n = 1
    BEGIN
        SET @ans = 1;
    END
    ELSE
    BEGIN
        SET @ans = @n + dbo.totalsum(@n-1);
    END
    RETURN @ans;
END

SELECT dbo.totalsum(10)
```

Output:



The screenshot shows a SQL query editor window titled "SQLQuery3.sql - UN...known Hector (57))*". The query defines a recursive function `totalsum` and calls it with the value 10. The function logic is as follows:

```
CREATE FUNCTION totalsum(@n int)
RETURNS INT
AS
BEGIN
    DECLARE @ans int;
    IF @n = 0 or @n = 1
    BEGIN
        SET @ans = 1;
    END
    ELSE
    BEGIN
        SET @ans = @n + dbo.totalsum(@n-1);
    END
    RETURN @ans;
END
```

Below the query, the execution result is displayed in a table with one row and one column:

| | (No column name) |
|---|------------------|
| 1 | 55 |

All Code:-

```
CREATE DATABASE LAB08
```

```
USE LAB
```

```
CREATE TABLE Employees
(
Employee_ID VARCHAR(50),
Name VARCHAR(50),
Age INT,
Salary INT,
Department VARCHAR(50)
);
```

```
INSERT INTO Employees VALUES('1','Adiya',25,30000,'Executive');
INSERT INTO Employees VALUES('2','Aditya',28,31000,'Accountant');
INSERT INTO Employees VALUES('3','Priyanka',30,32000,'Salesman');
INSERT INTO Employees VALUES('4','Anmol',35,31000,'Executive');
INSERT INTO Employees VALUES('5','Rahul',31,29000,'Accountant');
INSERT INTO Employees VALUES('6','Shubham',29,27000,'Salesman');
INSERT INTO Employees VALUES('7','Sam',33,32000,'Salesman');
INSERT INTO Employees VALUES('8','Rohan',33,28000,'Salesman');
INSERT INTO Employees VALUES('9','Priya',23,35000,'Accountant');
INSERT INTO Employees VALUES('10','Kabir',29,26000,'Executive');
```

```
SELECT*FROM Employees;
```

```
DECLARE @E_name varchar(20)= 'Aditya';
DECLARE @E_age int = 28;
DECLARE @E_salary int
BEGIN
    SELECT @E_salary = Salary FROM Employees WHERE Name = @E_name AND Age = @E_age;
    PRINT 'Salary of ' + @E_name + ' whose age is ' + cast(@E_age as varchar) + '
is ' + cast(@E_salary as varchar);
END
```

```
Create Procedure tuple
    @name varchar(40),
    @department varchar(40)
As
    Select *from Employees where Name=@name and Department=@department;
Go
```

```
Execute tuple @name='Priyanka',@department='Salesman'
```

```
CREATE FUNCTION unique_name_count()
returns int AS
BEGIN
declare    @name_uni_count int
    SELECT @name_uni_count=COUNT(distinct name)
FROM employee;
return @name_uni_count
END
Select dbo.unique_name_count() as unique_total_count
```

```
DECLARE my_cursor CURSOR FOR SELECT Employee_id, Name, Department FROM Employees;
```

```
OPEN my_cursor;
```

```

FETCH NEXT FROM my_cursor;

FETCH NEXT FROM my_cursor;

FETCH NEXT FROM my_cursor;

FETCH NEXT FROM my_cursor;

FETCH NEXT FROM my_cursor;

FETCH NEXT FROM my_cursor;

FETCH NEXT FROM my_cursor;

FETCH NEXT FROM my_cursor;

FETCH NEXT FROM my_cursor;

DEALLOCATE my_cursor;

Declare @v int
begin

DECLARE Qn_6 CURSOR FOR SELECT Employee_ID,Name,Salary FROM Employees;
OPEN Qn_6;
SET @v = 0;
WHILE @v < 10
BEGIN
FETCH NEXT FROM Qn_6;
SET @v = @v + 1;
END;

end

close Qn_6;
Deallocate Qn_6;

BEGIN TRY
DECLARE @num int;
SET @num = 1/0;
END TRY
BEGIN CATCH
SELECT
    ERROR_NUMBER() AS Error_Number,
    ERROR_SEVERITY() AS Error_Severity,
    ERROR_STATE() AS Error_State,
    ERROR_PROCEDURE() AS Error_Procedure,
    ERROR_LINE() AS Error_Line,
    ERROR_MESSAGE() AS Error_Message;
END CATCH;

CREATE TRIGGER Alarm ON Employees AFTER INSERT,
UPDATE AS RAISERROR ('Entry is added but trigger is activated', 5, 3);
GO
Insert into Employees values(13, 'ranbir', 29, 20000, 'Executive')

If object_id('inserterror', 'TR') is not null
    drop trigger inserterror;

```

```
CREATE FUNCTION totalsum(@n int)
RETURNS INT
AS
BEGIN
    DECLARE @ans int;
    IF @n = 0 or @n = 1
    BEGIN
        SET @ans = 1;
    END
    ELSE
    BEGIN
        SET @ans = @n + dbo.totalsum(@n-1);
    END
    RETURN @ans;
END

SELECT dbo.totalsum(10)
```