**Step 1:** Creating the table. Code for table creation with 4 columns ( empid, empname, designation and manid)

CREATE TABLE emptb ( -- Table creation

empid INT PRIMARY KEY NOT NULL,

empname VARCHAR(30) NOT NULL,

designation VARCHAR(25) NOT NULL,

manid INT NOT NULL

);

**Step 2:** Insertion of Employees into the created table. A procedure is created and insertion is done by entering all the parameters. Here, emp100 is tried inserting with manager id of 41 (EMPID: 41 is not present in the table). Hence, 'ERROR : EMP ID NOT EXISTS' is displayed and not inserted in the table.

--INSERT INTO TABLE IF GIVEN MANAGER ID EXISTS IN EMP ID COLUMN

PRINT 'Enter EMPLOYEE details in format EMPID, EMPNAME, DESIGNATION, MANAGERID in the EXEC statement below';

EXEC insert\_empdup 100,emp100,'BU HEAD',41;

CREATE PROCEDURE insert\_empdup @empid int, @empname nvarchar(30), @designation nvarchar(25), @manid int

AS

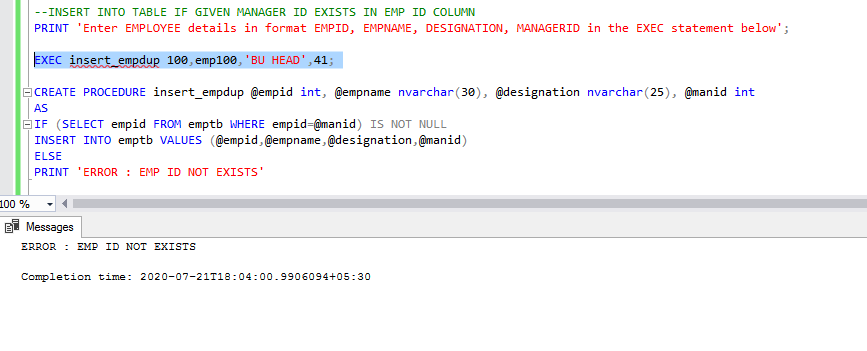
IF (SELECT empid FROM emptb WHERE empid=@manid) IS NOT NULL

INSERT INTO emptb VALUES (@empid,@empname,@designation,@manid)

ELSE

PRINT 'ERROR : EMP ID NOT EXISTS'

**Output :**



**Requirement 1:** Display Hierarchy of Organization for any given employee or root level

A procedure TraverseRecursive is defined which is a recursive procedure. The procedure is invoked with empid 9 highlighted below:

-- Display hierarchy using recursion

CREATE PROCEDURE TraverseRecursive

@empid INTEGER

AS

/\* to change action on each vertex, change these lines \*/

DECLARE @empname VARCHAR(30)

SELECT @empname=(SELECT empname

FROM emptb WHERE empid=@empid)

PRINT SPACE(@@NESTLEVEL\*2)+STR(@empid)+' '+@empname

/\* \*\*\*\*\*\* \*/

DECLARE subprojects CURSOR LOCAL FOR

SELECT empid FROM emptb WHERE manid=@empid

OPEN subprojects

FETCH NEXT FROM subprojects INTO @empid

WHILE @@FETCH\_STATUS=0 BEGIN

EXEC TraverseRecursive @empid

FETCH NEXT FROM subprojects INTO @empid

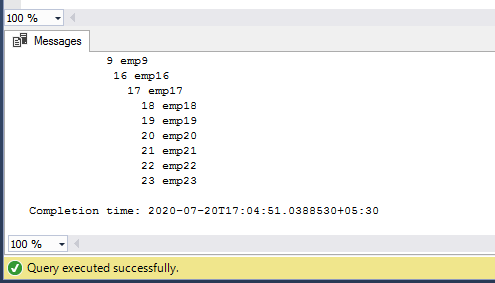
END

CLOSE subprojects

DEALLOCATE subprojects

TraverseRecursive 9

**Output 1:** Displays the employees under employee id 9



**Requirement 2:** Allow to insert an employee at any level.

Refer step2 of the report.

**Requirement 3:** Allow to remove an employee at any level.

**Case1:**  Here we are deleting employee with empid=10 highlighted below. The second parameter 0 is for attaching the descendants of emp10 to the manager of emp10 and later deletion of the employee emp10.

--Remove an employee at any level

PRINT 'ENTER THE EMPLOYEE ID to be deleted along with the EmpID under which descendants needs to be attached';

PRINT 'ENTER newmanid as 0 for default';

CREATE PROCEDURE del\_emp\_2 @empiddel int, @newmanid int

AS

IF @newmanid=0

GOTO defau;

ELSE

BEGIN

EXEC mov\_emp @empiddel, @newmanid; -- Moving the employee and its descendants

defau: DECLARE @temp int

SET @temp = (SELECT manid FROM emptb WHERE empid=@empiddel);

UPDATE emptb

SET manid=@temp

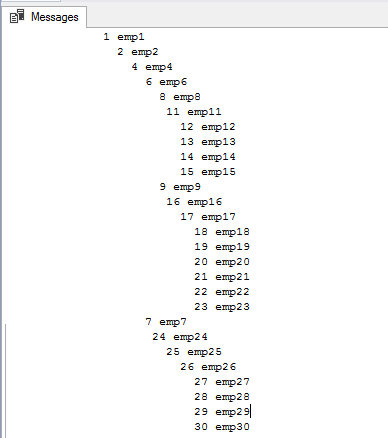
WHERE manid=@empiddel;

DELETE FROM emptb WHERE empid=@empiddel;

END

EXEC del\_emp\_2 10,0;

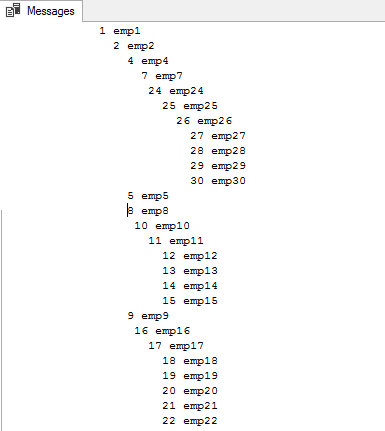
**Output 3a:** After deleting employee with empid=10, emp10’s descendants are under now under emp8.



**Case 2:** Here, we wish to delete emp6 and put its descendants under emp2.

EXEC del\_emp\_2 6,2;

**Output 3b:** After deleting emp6, descendants of emp6 are attached to emp2.



**Requirement 4:** Allow to move an employee and its descendant hierarchy from one level to another level. Here, we are moving emp10 who is working under emp8 to work under emp2 highlighted below.

--Move employee from one level to another

PRINT 'ENTER THE EMPLOYEE ID WHOSE LEVEL NEEDS TO BE CHANGED IN FORMAT empid, new managerid';

CREATE PROCEDURE mov\_emp @empid int, @newmanid int

AS

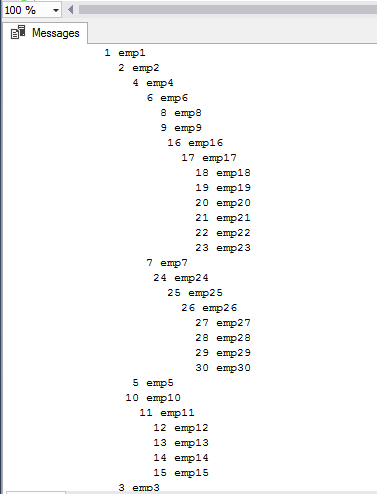
UPDATE emptb

SET manid=@newmanid

WHERE empid=@empid;

EXEC mov\_emp 10,2;

**Output 4:** Employee 10 and his descendants are now reporting under emp2.



**Requirement 5:** Allow to modify employee name.

**Case 1:** Here, we are going to modify employee with the empid =10 to John as highlighted below:

--Modify the employee name

PRINT 'ENTER EMPID ALONG WITH THE UPDATED EMPLOYEE NAME';

CREATE PROC mod\_emp @empid int, @empupdate nvarchar(30)

AS

DECLARE @temp int

SET @temp=(SELECT COUNT (empname) FROM emptb WHERE empname=@empupdate)

IF @temp=0

BEGIN

UPDATE emptb

SET empname = @empupdate

WHERE empid = @empid;

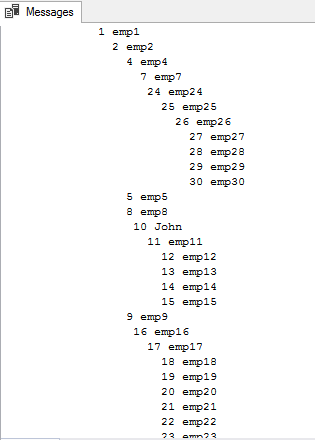
END

ELSE

PRINT 'DUPLICATE NAME FOUND'

EXEC mod\_emp 10,John;

**Output 5a:** Employee with empid =10 whose name was emp10 has been modified to John.



**Case 2:** When the name of empid=13, whose current name is emp13 is attempted to be changed to John, a message is displayed saying 'DUPLICATE NAME FOUND'

EXEC mod\_emp 13,John;

**Output 5b:**

