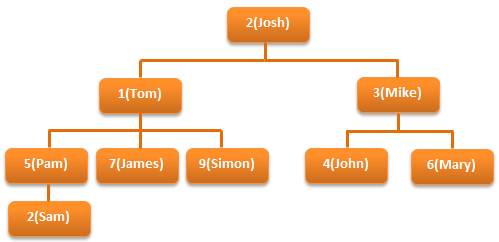
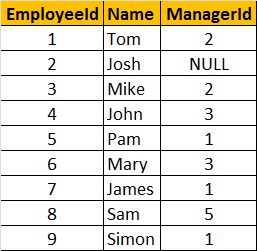
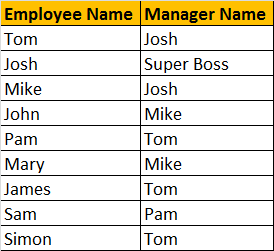
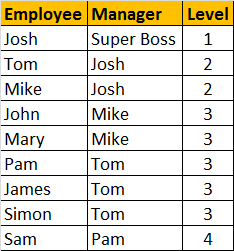
**Suggested SQL Server videos**  
[Part 49 - Common Table Expressions (CTE)](http://csharp-video-tutorials.blogspot.com/2012/09/common-table-expressions-part-49.html)  
[Part 50 - Updatable CTE](http://csharp-video-tutorials.blogspot.com/2012/09/updatable-cte-part-50.html)   
  
   
  
   
  
   
  
**A CTE that references itself is called as recursive CTE**. Recursive CTE's can be of great help when displaying hierarchical data. Example, displaying employees in an organization hierarchy. A simple organization chart is shown below.  
   
  
**Let's create tblEmployee table, which holds the data, that's in the organization chart.**  
Create Table tblEmployee  
(  
  EmployeeId int Primary key,  
  Name nvarchar(20),  
  ManagerId int  
)  
  
Insert into tblEmployee values (1, 'Tom', 2)  
Insert into tblEmployee values (2, 'Josh', null)  
Insert into tblEmployee values (3, 'Mike', 2)  
Insert into tblEmployee values (4, 'John', 3)  
Insert into tblEmployee values (5, 'Pam', 1)  
Insert into tblEmployee values (6, 'Mary', 3)  
Insert into tblEmployee values (7, 'James', 1)  
Insert into tblEmployee values (8, 'Sam', 5)  
Insert into tblEmployee values (9, 'Simon', 1)  
  
**Since, a MANAGER is also an EMPLOYEE**, both manager and employee details are stored in tblEmployee table. Data from tblEmployee is shown below.  
  
  
**Let's say, we want to display, EmployeeName along with their ManagerName**. The ouptut should be as shown below.  
  
  
**To achieve this, we can simply join tblEmployee with itself.** Joining a table with itself is called as self join. We discussed about [**Self Joins in Part 14**](http://csharp-video-tutorials.blogspot.com/2012/08/self-join-in-sql-server-part-14.html) of this video series. In the output, notice that since **JOSH** does not have a Manager, we are displaying **'Super Boss'**, instead of **NULL**. We used **IsNull**(), function to replace NULL with 'Super Boss'. If you want to learn more about [**replacing NULL values, please watch Part 15**](http://csharp-video-tutorials.blogspot.com/2012/08/different-ways-to-replace-null-in-sql.html).  
**SELF JOIN QUERY:**  
Select Employee.Name as [Employee Name],  
IsNull(Manager.Name, 'Super Boss') as [Manager Name]  
from tblEmployee Employee  
left join tblEmployee Manager  
on Employee.ManagerId = Manager.EmployeeId  
  
**Along with Employee and their Manager name**, we also want to display their level in the organization. The output should be as shown below.  
   
  
**We can easily achieve this using a self referencing CTE.**  
With  
  EmployeesCTE (EmployeeId, Name, ManagerId, [Level])  
  as  
  (  
    Select EmployeeId, Name, ManagerId, 1  
    from tblEmployee  
    where ManagerId is null  
      
    union all  
      
    Select tblEmployee.EmployeeId, tblEmployee.Name,   
    tblEmployee.ManagerId, EmployeesCTE.[Level] + 1  
    from tblEmployee  
    join EmployeesCTE  
    on tblEmployee.ManagerID = EmployeesCTE.EmployeeId  
  )  
Select EmpCTE.Name as Employee, Isnull(MgrCTE.Name, 'Super Boss') as Manager,   
EmpCTE.[Level]   
from EmployeesCTE EmpCTE  
left join EmployeesCTE MgrCTE  
on EmpCTE.ManagerId = MgrCTE.EmployeeId  
  
The **EmployeesCTE**contains 2 queries with **UNION ALL** operator. The first query selects the EmployeeId, Name, ManagerId, and 1 as the level from **tblEmployee** where ManagerId is NULL. So, here we are giving a LEVEL = 1 for **super boss** (Whose Manager Id is NULL). In the second query, we are joining **tblEmployee** with **EmployeesCTE** itself, which allows us to loop thru the hierarchy. Finally to get the reuired output, we are joining **EmployeesCTE** with itself.