**SQL Workshop 2**

***Question 1:***

You are given a table, *BST*, containing two columns: *N*and *P,* where *N* represents the value of a node in *Binary Tree*, and *P* is the parent of *N*.

Write a query to find the node type of *Binary Tree* ordered by the value of the node. Output one of the following for each node:

* *Root*: If node is root node.
* *Leaf*: If node is leaf node.
* *Inner*: If node is neither root nor leaf node.

**Sample Input**



**Sample Output**

1 Leaf

2 Inner

3 Leaf

5 Root

6 Leaf

8 Inner

9 Leaf

**Explanation**

The *Binary Tree* below illustrates the sample:



***Ans:***

create table BST

(n integer,

p integer);

insert into BST

Values (1,2),(3,2),

(6,8),(9,8),(2,5),

(8,5),(5,null);

select \* from BST

SELECT N,

CASE

WHEN P IS NULL THEN 'Root'

WHEN N IN (SELECT DISTINCT P FROM BST) THEN 'Inner'

ELSE 'Leaf'

END as nodes

FROM BST

ORDER BY N;

***Question 2:***

[Pivot](https://en.wikipedia.org/wiki/Pivot_table) the Occupation column in OCCUPATIONS so that each Name is sorted alphabetically and displayed underneath its corresponding Occupation. The output column headers should be Doctor, Professor, Singer, and Actor, respectively.

Note: Print NULL when there are no more names corresponding to an occupation.

**Input Format**

Occupation will only contain one of the following values: Doctor, Professor, Singer or Actor

**Sample Input:**



**Sample Output**

Jenny Ashley Meera Jane

Samantha Christeen Priya Julia

NULL Ketty NULL Maria

**Explanation**

The first column is an alphabetically ordered list of doctor names.  
The second column is an alphabetically ordered list of Professor names.  
The third column is an alphabetically ordered list of Singer names.  
The fourth column is an alphabetically ordered list of Actor names.  
The empty cell data for columns with less than the maximum number of names per occupation (in this case, the Professor and Actor columns) are filled with NULL values.

Ans :

select d.name , a.name , c.name , b.name from

(select row\_number() over(order by name) as r ,

name from occupations where occupation = 'professor') a

left join

(select row\_number() over(order by name) as r ,

name from occupations where occupation = 'actor') b

on a.r = b.r

left join

(select row\_number() over(order by name) as r ,

name from occupations where occupation = 'singer') c

on a.r = c.r

left join

(select row\_number() over(order by name) as r ,

name from occupations where occupation = 'doctor') d

on a.r = d.r

***Question 3:***

Amber's conglomerate corporation just acquired some new companies. Each of the companies follows this hierarchy:



Given the table schemas below, write a query to print the company\_code, founder name, total number of lead managers, total number of senior managers, total number of managers, and total number of employees. Order your output by ascending company\_code.

**Note:**

* The tables may contain duplicate records.
* The company\_code is string, so the sorting should not be numeric. For example, if the company\_codes are C\_1, C\_2, and C\_10, then the ascending company\_codes will be C\_1, C\_10, and C\_2.

**Input Format**

The following tables contain company data:

* Company: The company\_code is the code of the company and founder is the founder of the company.
* **Sample Input**



* *Lead\_Manager:* The *lead\_manager\_code* is the code of the lead manager, and the *company\_code* is the code of the working company



* *Senior\_Manager: The senior\_manager\_code is the code of the senior manager, the lead\_manager\_code is the code of its lead manager, and the*company\_code*is the code of the working company.*



* *Manager: The manager\_code is the code of the manager, the senior\_manager\_code is the code of its senior manager, the lead\_manager\_code is the code of its lead manager, and the company\_code is the code of the working company.*



* Employee:*The*employee\_code*is the code of the employee, the*manager\_code*is the code of its manager, the*senior\_*manager\_code is the code of its senior manager, the lead\_manager\_code is the code of its lead manager, and the* company\_code*is the code of the working company.*



**Sample Output**

C1 Monika 1 2 1 2

C2 Samantha 1 1 2 2

**Explanation**

In company *C1*, the only lead manager is *LM1*. There are two senior managers, *SM1* and *SM2*, under *LM1*. There is one manager, *M1*, under senior manager *SM1*. There are two employees, *E1* and *E2*, under manager *M1*.

In company *C2*, the only lead manager is *LM2*. There is one senior manager, *SM3*, under *LM2*. There are two managers, *M2* and *M3*, under senior manager *SM3*. There is one employee, *E3*, under manager *M2*, and another employee, *E4*, under manager, *M3*.

select

c.company\_code,

c.founder,

count(distinct l.lead\_manager\_code) as lead\_managers,

count(distinct s.senior\_manager\_code) as senior\_managers,

count(distinct m.manager\_code) as managers,

count(distinct e.employee\_code) as employees

from company c

left join lead\_manager l on c.company\_code = l.company\_code

left join senior\_manager s on c.company\_code = s.company\_code

left join manager m on c.company\_code = m.company\_code

left join employeee e on c.company\_code = e.company\_code

group by c.company\_code, c.founder

order by c.company\_code;