**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:



***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.

***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*.

------------------------------------------------------------Answers-----------------------------------------------------------

create table table1(

N INT,

P INT)

Insert into table1 values(1,2),(3,2),(6,8),(9,8),(2,5),(8,5),(5,null)

select \* from table1

SELECT N,

CASE    WHEN P is NULL    THEN 'Root'    WHEN N in (select p from BST)    THEN 'Inner'    ELSE    'Leaf'

END AS NODE\_TYPE

FROM table1 order by N;

CREATE TABLE OCCUPATIONS(    NAME VARCHAR(50),    OCCUPATION VARCHAR(50)

)

INSERT INTO    OCCUPATIONS

VALUES    ('Samantha','Doctor'),    ('Julia','Actor'),    ('Maria','Actor'),    ('Meera','Singer'),    ('Ashely','Professor'),    ('Ketty','Professor'),    ('Christeen','Professor'),    ('Jane','Actor'),    ('Jenny','Doctor'),    ('Priya','Singer')

select \* from OCCUPATIONS

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

CREATE TABLE COMPANY(

COMPANY\_CODE VARCHAR(5) PRIMARY KEY,

FOUNDER VARCHAR(15)

)

CREATE TABLE LEAD\_MANAGER(

LEAD\_MANAGER\_CODE VARCHAR(5) PRIMARY KEY,

COMPANY\_CODE VARCHAR(5)

CONSTRAINT FK FOREIGN KEY(COMPANY\_CODE) REFERENCES COMPANY(COMPANY\_CODE)

)

CREATE TABLE SENIOR\_MANAGER(SENIOR\_MANAGER\_CODE VARCHAR(5) PRIMARY KEY,

LEAD\_MANAGER\_CODE VARCHAR(5),

COMPANY\_CODE VARCHAR(5)

CONSTRAINT A1 FOREIGN KEY(COMPANY\_CODE) REFERENCES COMPANY(COMPANY\_CODE),

CONSTRAINT B1 FOREIGN KEY(LEAD\_MANAGER\_CODE) REFERENCES LEAD\_MANAGER(LEAD\_MANAGER\_CODE)

)

CREATE TABLE MANAGER(

MANAGER\_CODE VARCHAR(5) PRIMARY KEY,

SENIOR\_MANAGER\_CODE VARCHAR(5),

LEAD\_MANAGER\_CODE VARCHAR(5),

COMPANY\_CODE VARCHAR(5),

CONSTRAINT C1 FOREIGN KEY(COMPANY\_CODE) REFERENCES COMPANY(COMPANY\_CODE),

CONSTRAINT D1 FOREIGN KEY(LEAD\_MANAGER\_CODE) REFERENCES LEAD\_MANAGER(LEAD\_MANAGER\_CODE),

CONSTRAINT E1 FOREIGN KEY(SENIOR\_MANAGER\_CODE) REFERENCES SENIOR\_MANAGER(SENIOR\_MANAGER\_CODE)

)

CREATE TABLE EMPLOYEEE(

EMPLOYEE\_CODE VARCHAR(5) PRIMARY KEY,

MANAGER\_CODE VARCHAR(5),

SENIOR\_MANAGER\_CODE VARCHAR(5),

LEAD\_MANAGER\_CODE VARCHAR(5),

COMPANY\_CODE VARCHAR(5),

CONSTRAINT F1 FOREIGN KEY(COMPANY\_CODE) REFERENCES COMPANY(COMPANY\_CODE),

CONSTRAINT G1 FOREIGN KEY(LEAD\_MANAGER\_CODE) REFERENCES LEAD\_MANAGER(LEAD\_MANAGER\_CODE),

CONSTRAINT H1 FOREIGN KEY(SENIOR\_MANAGER\_CODE) REFERENCES SENIOR\_MANAGER(SENIOR\_MANAGER\_CODE),

CONSTRAINT I1 FOREIGN KEY(MANAGER\_CODE) REFERENCES MANAGER(MANAGER\_CODE)

)

SELECT \* FROM COMPANY

SELECT \* FROM LEAD\_MANAGER

SELECT \* FROM SENIOR\_MANAGER

SELECT \* FROM MANAGER

SELECT \* FROM EMPLOYEEE

INSERT INTO COMPANY VALUES ('C1','MONIKA'),('C2','SAMANTHA')

INSERT INTO LEAD\_MANAGER VALUES ('LM1','C1'),('LM2','C2')

INSERT INTO SENIOR\_MANAGER VALUES ('SM1','LM1','C1'),('SM2','LM1','C1'),('SM3','LM2','C2')

INSERT INTO MANAGER VALUES ('M1','SM1','LM1','C1'),('M2','SM3','LM2','C2'),('M3','SM3','LM2','C2')

INSERT INTO EMPLOYEEE VALUES ('E1','M1','SM1','LM1','C1'),('E2','M1','SM1','LM1','C1'),('E3','M2','SM3','LM2','C2'),('E4','M3','SM3','LM2','C2')

--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

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;