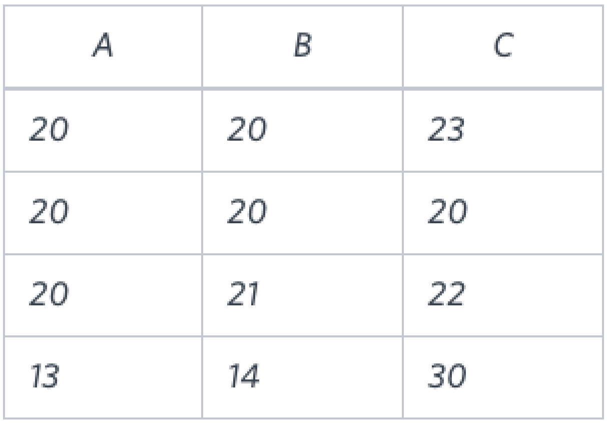
Basic Select

1. Type of Triangle

Write a query identifying the type of each record in the TRIANGLES table using its three side lengths. Output one of the following statements for each record in the table:

- Equilateral: It's a triangle with sides of equal length.
- Isosceles: It's a triangle with sides of equal length.
- Scalene: It's a triangle with sides of differing lengths.
- Not A Triangle: The given values of A, B, and C don't form a triangle. Input Format

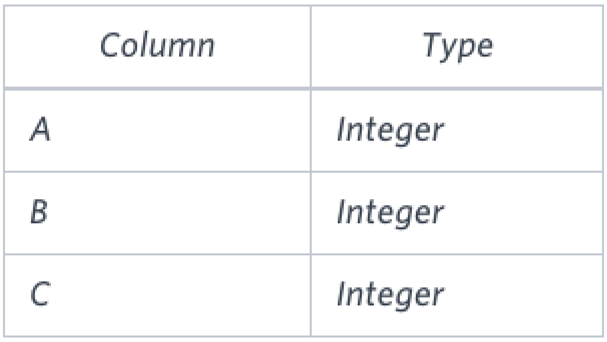
The TRIANGLES table is described as follows:



A	B	C
20	20	23
20	20	20
20	21	22
13	14	30

Each row in the table denotes the lengths of each of a triangle's three sides.

Sample Input



Column	Type
A	Integer
B	Integer
C	Integer

Sample Output

Isosceles Equilateral Scalene Not A Triangle Explanation

Values in the tuple (20, 20, 23) form an Isosceles triangle, because A=B. Values in the tuple (20, 20, 20) form an Equilateral triangle, because A=B=C. Values in the tuple (20,21,22) form a Scalene triangle, because A!= B!= C. Values in the tuple cannot form a triangle because the combined value of sides A and B is not larger than that of side C.

Link of question Markdown Live Preview.

```
Query : Select
CASE
when A + B <= C or A + C <= B or B + C <= A then "Not A Triangle"
when A = B and B = C then "Equilateral"
when A = B or A = C or B = C then "Isosceles"
else "Scalene"
end as triangle_sides
from TRIANGLES;
```

2. The PADS

Generate the following two result sets:

1. Query an alphabetically ordered list of all names in OCCUPATIONS, immediately followed by the first letter of each profession as a parenthetical (i.e.: enclosed in parentheses). For example: AnActorName(A), ADoctorName(D), AProfessorName(P), and ASingerName(S).
2. Query the number of ocurrences of each occupation in OCCUPATIONS. Sort the occurrences in ascending order, and output them in the following format:

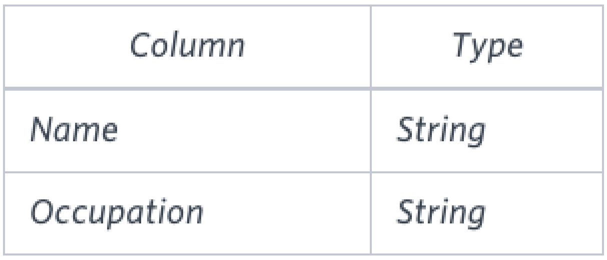
```
There are a total of [occupation_count] [occupation]s.
```

where [occupation_count] is the number of occurrences of an occupation in OCCUPATIONS and [occupation] is the lowercase occupation name. If more than one Occupation has the same [occupation_count], they should be ordered alphabetically.

Note: There will be at least two entries in the table for each type of occupation.

Input Format

The OCCUPATIONS table is described as follows:

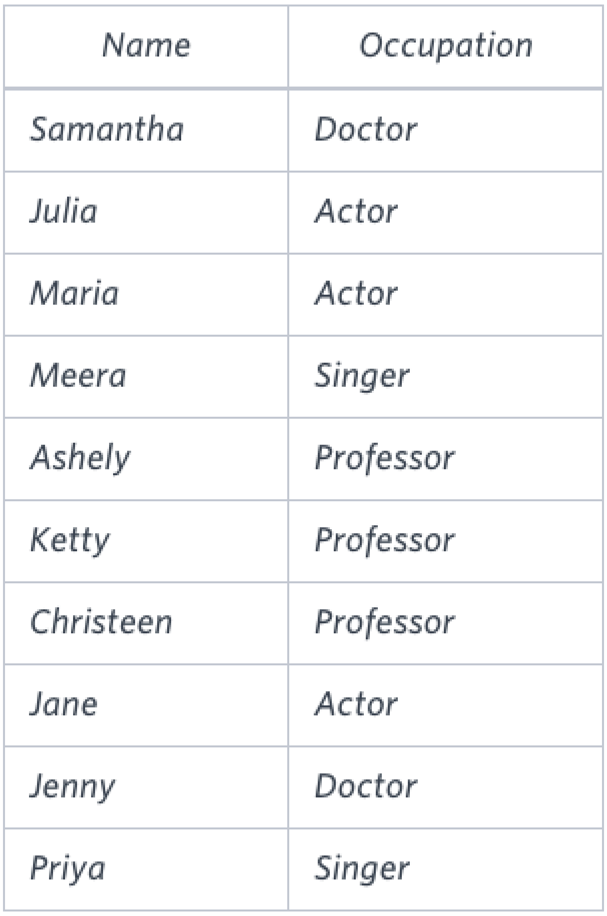


Column	Type
Name	String
Occupation	String

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

Sample Input

An OCCUPATIONS table that contains the following records:



Name	Occupation
Samantha	Doctor
Julia	Actor
Maria	Actor
Meera	Singer
Ashely	Professor
Ketty	Professor
Christeen	Professor
Jane	Actor
Jenny	Doctor
Priya	Singer

Sample Output

Ashely(P) Christeen(P) Jane(A) Jenny(D) Julia(A) Ketty(P) Maria(A) Meera(S) Priya(S) Samantha(D) There are a total of 2 doctors. There are a total of 2 singers. There are a total of 3 actors. There are a total of 3 professors.

Explanation

The results of the first query are formatted to the problem description's specifications. The results of the second query are ascendingly ordered first by number of names corresponding to each profession (2 <= 2 <= 3 <= 3), and then alphabetically by profession (doctors <= singer, and actor <= professor).

Link of question Markdown Live Preview.

```
Query : select concat(name,'(',substring(Occupation,1,1),')') as Name
from occupations
order by Name;
Select concat ('There are a total of ', count(occupation),' ',
lower(occupation),'s.') as totals
from occupations
group by occupation
order by totals;
```

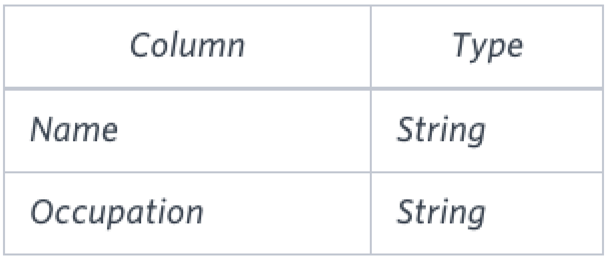
3. Ocupations

   Pivot 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

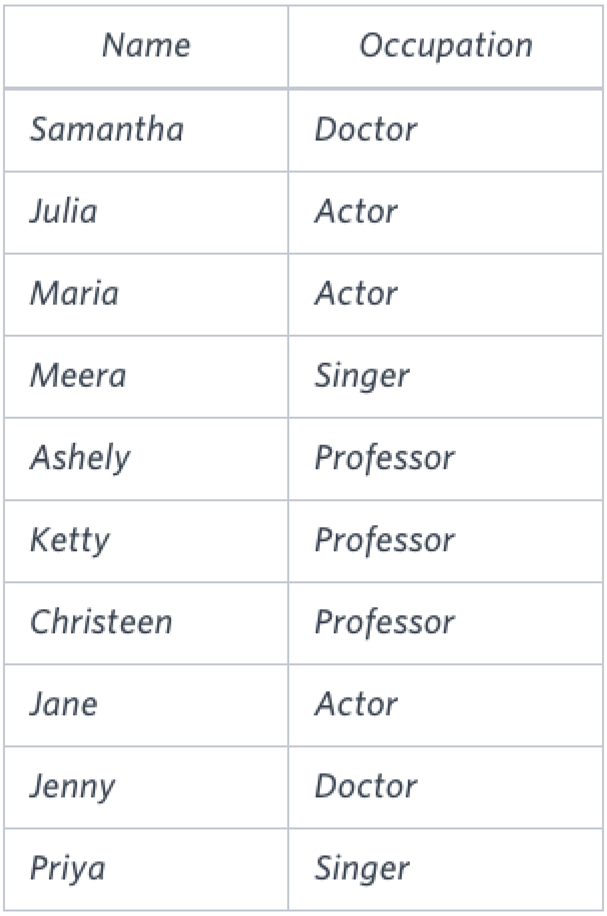
The OCCUPATIONS table is described as follows:



Column	Type
Name	String
Occupation	String

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

Sample Input



Name	Occupation
Samantha	Doctor
Julia	Actor
Maria	Actor
Meera	Singer
Ashely	Professor
Ketty	Professor
Christeen	Professor
Jane	Actor
Jenny	Doctor
Priya	Singer

Sample Output

Jenny Ashley Meera Jane Samantha Christeen Priya Julia NULL Ketty NULL

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.

Link of question Markdown Live Preview.

using normal grouping :

Link of video Markdown Live Preview.

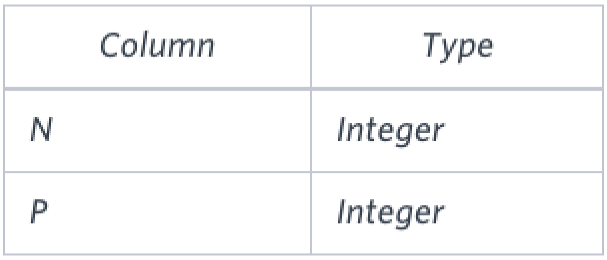
using pivot :

Link of video Markdown Live Preview.

```
Query : Select
    MAX(IF(OCCUPATION = "DOCTOR",NAME,NULL)) AS DOCTOR ,
    Min(IF(OCCUPATION = "PROFESSOR",NAME,NULL)) AS PROFESSOR ,
    MAX(IF(OCCUPATION = "SINGER",NAME,NULL)) AS SINGER ,
    Min(IF(OCCUPATION = "ACTOR",NAME,NULL)) AS ACTOR
FROM
(select name,occupation,Row_number() Over (PARTITION BY occupation ORDER BY name)
as row_num FROM occupations) as ord group by row_num;

Another Approch :
 set @d=0, @p=0, @s=0, @a=0;

select min(Doctor), min(Professor), min(Singer), min(Actor)
from(
  select case
            when Occupation='Doctor' then (@d:=@d+1)
            when Occupation='Professor' then (@p:=@p+1)
            when Occupation='Singer' then (@s:=@s+1)
            when Occupation='Actor' then (@a:=@a+1)
            end as Row,
        case when Occupation='Doctor' then Name end as Doctor,
        case when Occupation='Professor' then Name end as Professor,
        case when Occupation='Singer' then Name end as Singer,
        case when Occupation='Actor' then Name end as Actor
  from OCCUPATIONS
  order by Name
) as temp
group by Row;

Another Approch :

SELECT *
FROM
  (SELECT MIN(DOCTOR) MIN_DOCTOR,
          MIN(PROFESSOR) MIN_PROFESSOR,
          MIN(SINGER) MIN_SINGER,
          MIN(ACTOR) MIN_ACTOR
   FROM
     (SELECT CASE
                 WHEN OCCUPATION = 'Doctor' THEN NAME
             END AS DOCTOR,
             CASE
                 WHEN OCCUPATION = 'Professor' THEN NAME
             END AS PROFESSOR,
             CASE
                 WHEN OCCUPATION = 'Singer' THEN NAME
             END AS SINGER,
             CASE
                 WHEN OCCUPATION = 'Actor' THEN NAME
             END AS ACTOR,
             RANK() OVER (PARTITION BY OCCUPATION
                          ORDER BY NAME) AS ROW_RANK
      FROM OCCUPATIONS) X
   GROUP BY ROW_RANK)
ORDER BY MIN_DOCTOR,
         MIN_PROFESSOR,
         MIN_SINGER,
         MIN_ACTOR;
```

1. Binary Tree Nodes

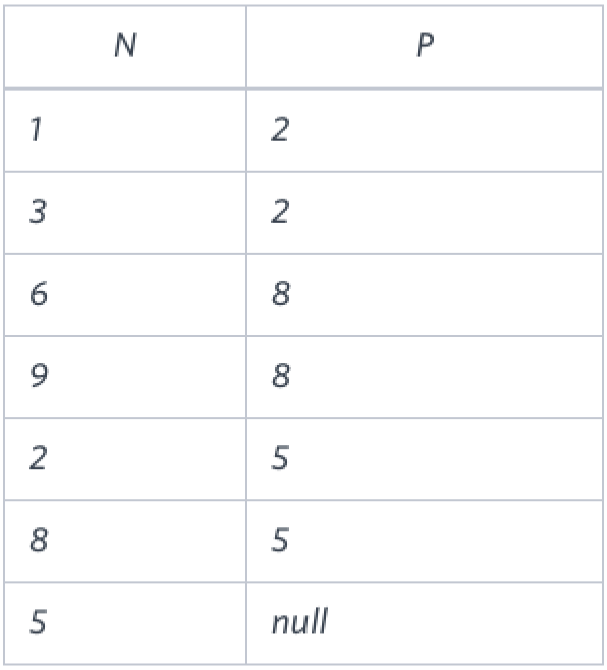
   You are given a table, BST, containing two columns: N and P, where N represents the value of a node in Binary Tree, and P is the parent of N.



Column	*Type*
N	*Integer*
P	*Integer*

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



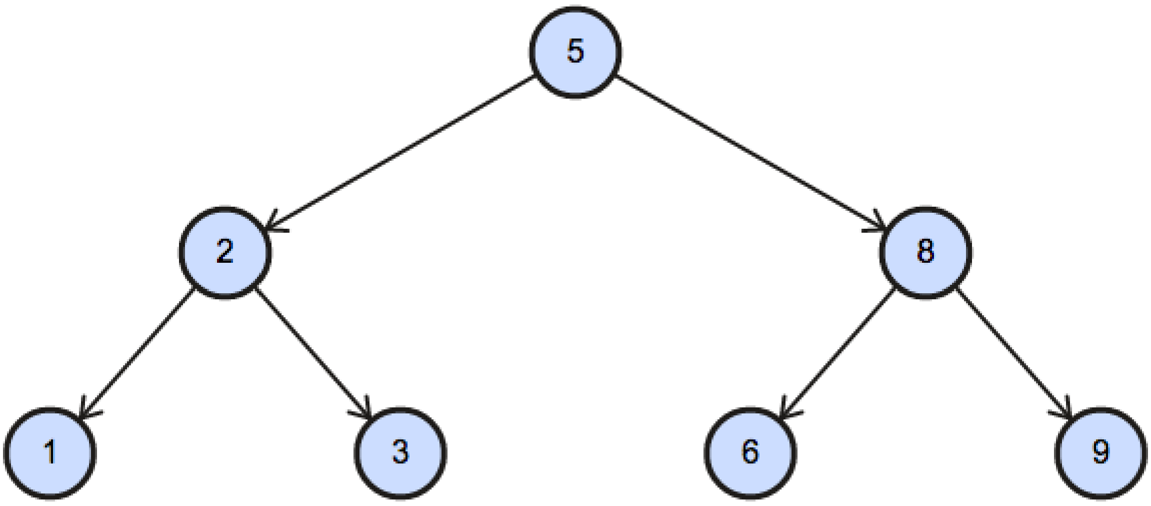
N	P
1	2
3	2
6	8
9	8
2	5
8	5
5	null

Sample Output

1 Leaf 2 Inner 3 Leaf 5 Root 6 Leaf 8 Inner 9 Leaf

Explanation

The Binary Tree below illustrates the sample:



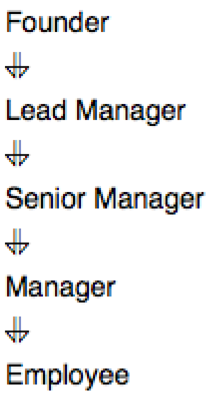
Link of question Markdown Live Preview.

```
Query : SELECT BT.N,
CASE
    WHEN BT.P IS NULL THEN 'Root'
    WHEN EXISTS (SELECT B.P FROM BST B WHERE B.P = BT.N) THEN 'Inner'
    ELSE 'Leaf'
END
```

```
FROM BST AS BT
ORDER BY BT.N;
```

5. New Companies

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



Founder
⇓
Lead Manager
⇓
Senior Manager
⇓
Manager
⇓
Employee

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.

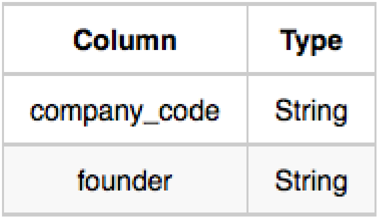
Note:

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

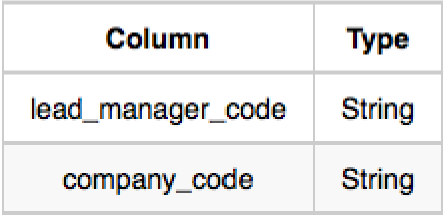
The following tables contain company data:

- Company: The company_code is the code of the company and founder is the founder of the company.



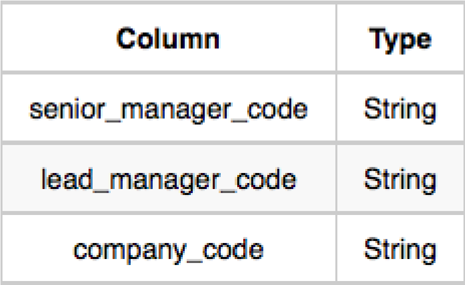
Column	Type
company_code	String
founder	String

- Lead_Manager: The lead_manager_code is the code of the lead manager, and the company_code is the code of the working company.



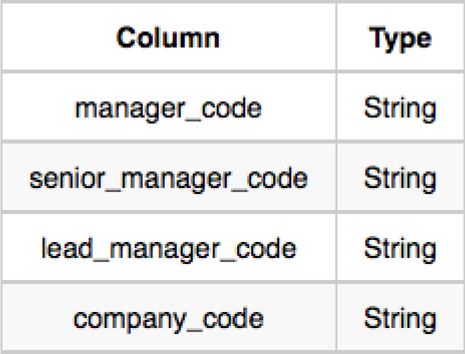
Column	Type
lead_manager_code	String
company_code	String

- 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.



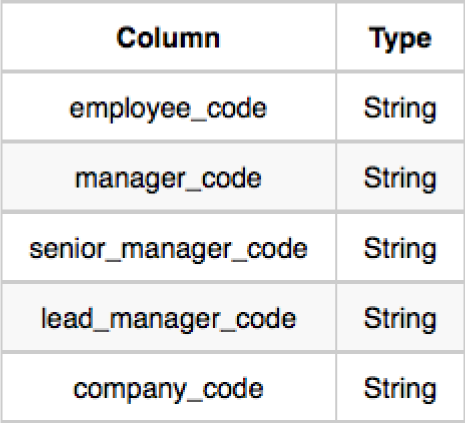
Column	Type
senior_manager_code	String
lead_manager_code	String
company_code	String

- 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.



Column	Type
manager_code	String
senior_manager_code	String
lead_manager_code	String
company_code	String

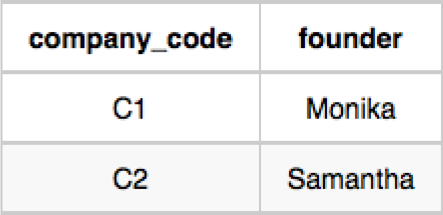
- Employee: The employee_code is the code of the employee, the manager_code is the code of its manager, the senior_manager_code is the code of its senior manager, the lead_manager_code is the code of its lead manager, and the company_code is the code of the working company.



Column	Type
employee_code	String
manager_code	String
senior_manager_code	String
lead_manager_code	String
company_code	String

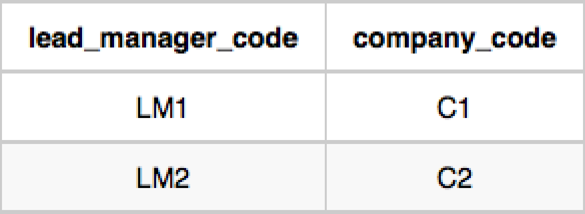
Sample Input

Company Table:



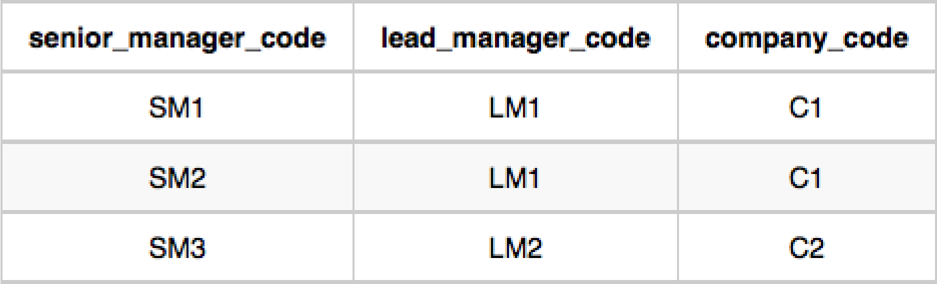
company_code	founder
C1	Monika
C2	Samantha

Lead_Manager Table:



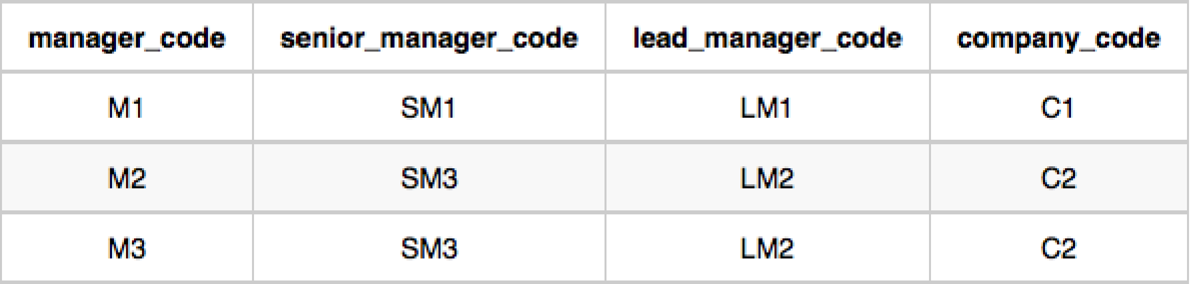
lead_manager_code	company_code
LM1	C1
LM2	C2

Senior_Manager Table:



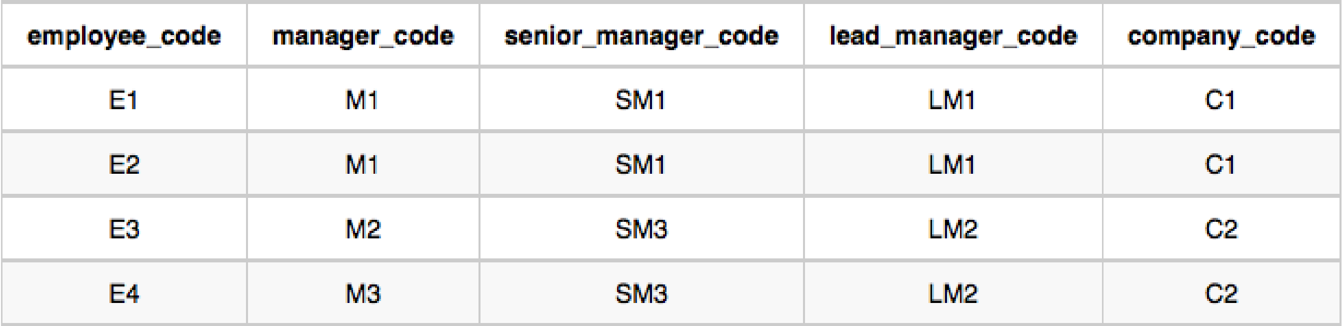
senior_manager_code	lead_manager_code	company_code
SM1	LM1	C1
SM2	LM1	C1
SM3	LM2	C2

Manager Table:



manager_code	senior_manager_code	lead_manager_code	company_code
M1	SM1	LM1	C1
M2	SM3	LM2	C2
M3	SM3	LM2	C2

Employee Table:



employee_code	manager_code	senior_manager_code	lead_manager_code	company_code
E1	M1	SM1	LM1	C1
E2	M1	SM1	LM1	C1
E3	M2	SM3	LM2	C2
E4	M3	SM3	LM2	C2

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.

Link of question Markdown Live Preview.

```
Query : SELECT c.company_code,c.founder,
count(distinct lm.lead_manager_code),
count(distinct sm.senior_manager_code),
count(distinct m.manager_code),
count(distinct e.employee_code)
FROM Company c, Lead_Manager lm, Senior_Manager sm, Manager m, Employee e
WHERE
c.company_code=lm.company_code AND
lm.lead_manager_code=sm.lead_manager_code AND
sm.senior_manager_code=m.senior_manager_code AND
m.manager_code=e.manager_code
GROUP BY c.company_code,c.founder
ORDER BY c.company_code **ASC**;
```