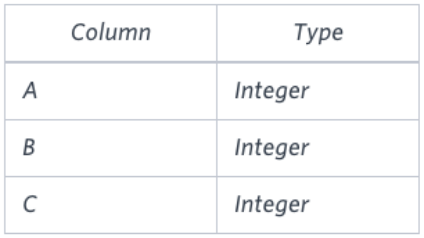
**Q.22:** Write a query identifying the *type* of each record in the **TRIANGLES** table using its three side lengths. Output one of the following statements for each record in the table:

* **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:



**Answer:**

select case

when A + B <= C or B + C <= A or A + C <= B then 'Not A Triangle'

when A = B and B = C then 'Equilateral'

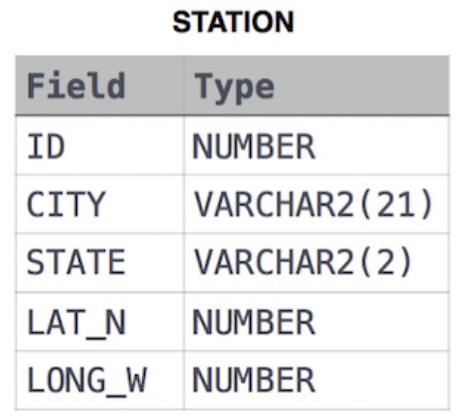
when A = B or B = C or A = C then 'Isosceles'

else 'Scalene'

end

from triangles

**Q.23:** Query the two cities in **STATION** with the shortest and longest *CITY* names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.  
The **STATION** table is described as follows:



where **LAT\_N** is the northern latitude and **LONG\_W** is the western longitude.

**Answer:** (select city, length(city) from station

order by length(city), city asc

limit 1)

UNION

(select city, length(city) from station

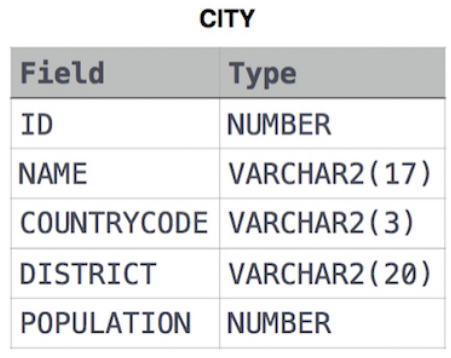
order by length(city) desc

limit 1)

**Q.24:** Query the difference between the maximum and minimum populations in **CITY**.

**Input Format**

The **CITY** table is described as follows:



**Answer:** select max(population) - min(population) from city

**Q.25:** Query a *count* of the number of cities in **CITY** having a *Population* larger than 100,000.

**Input Format**

The **CITY** table is described as follows:

A screenshot of a computer

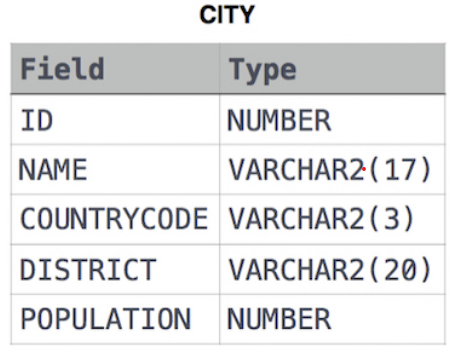
AI-generated content may be incorrect.

**Answer:** select count(population) from city where population > 100000

**Q.26:** Query the average population for all cities in **CITY**, rounded *down* to the nearest integer.

**Input Format**

The **CITY** table is described as follows:

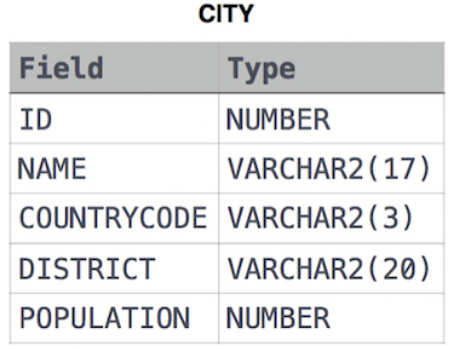


**Answer:** select round(avg(population)) from city

**Q.27:** Query the sum of the populations for all Japanese cities in **CITY**. The *COUNTRYCODE* for Japan is **JPN**.

**Input Format**

The **CITY** table is described as follows:



**Answer:** select sum(population) from city where countrycode = "JPN"

**Q.28:** Samantha was tasked with calculating the average monthly salaries for all employees in the **EMPLOYEES** table, but did not realize her keyboard's  key was broken until after completing the calculation. She wants your help finding the difference between her miscalculation (using salaries with any zeros removed), and the actual average salary.

Write a query calculating the amount of error (i.e.: actual - miscalculated average monthly salaries), and round it up to the next integer.

**Input Format**

The **EMPLOYEES** table is described as follows:

A screenshot of a computer

AI-generated content may be incorrect.

**Note:** *Salary* is per month.

**Constraints**

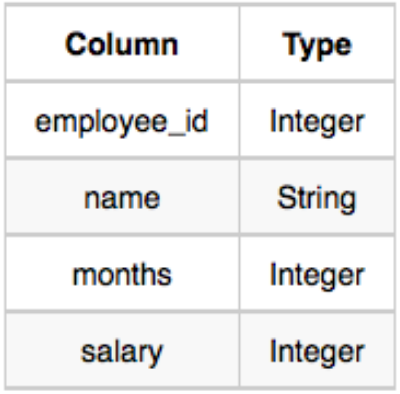
**1000 < salary < 105**

**Answer:** select ceil(avg(salary) - avg(replace(salary,0,''))) from employees

**Q.29:** We define an employee's *total earnings* to be their monthly  worked, and the *maximum total earnings* to be the maximum total earnings for any employee in the **Employee** table. Write a query to find the *maximum total earnings* for all employees as well as the total number of employees who have maximum total earnings. Then print these values as  space-separated integers.

**Input Format**

The **Employee** table containing employee data for a company is described as follows:



where *employee\_id* is an employee's ID number, *name* is their name, *months* is the total number of months they've been working for the company, and *salary* is the their monthly salary.

**Answer:** select (salary\*months) as earnings, count(\*) from employee

group by earnings

order by earnings desc

limit 1

**Q.30:** Query the following two values from the **STATION** table:

1. The sum of all values in *LAT\_N* rounded to a scale of 2 decimal places.
2. The sum of all values in *LONG\_W* rounded to a scale of 2 decimal places.

**Input Format**

The **STATION** table is described as follows:

A table with text and numbers

AI-generated content may be incorrect.

where *LAT\_N* is the northern latitude and *LONG\_W* is the western longitude.

**Answer:** select round(sum(LAT\_N),2), round(sum(LONG\_W),2) from station

**Q.31:** Query the sum of *Northern Latitudes* (*LAT\_N*) from **STATION** having values greater than  and less than . Truncate your answer to  decimal places.

**Input Format**

The **STATION** table is described as follows:

A table with text and numbers

AI-generated content may be incorrect.

where *LAT\_N* is the northern latitude and *LONG\_W* is the western longitude.

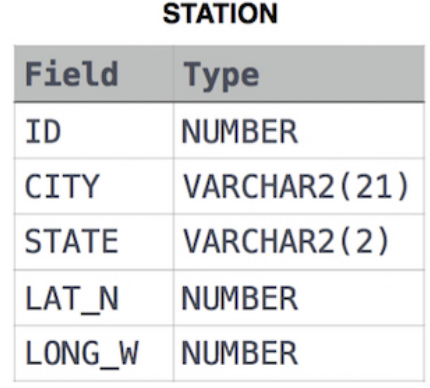
**Answer:** select round(sum(LAT\_N),4) from station

where LAT\_N > 38.7880 and LAT\_N < 137.2345

**Q.32:** Query the greatest value of the *Northern Latitudes* (*LAT\_N*) from **STATION** that is less than 137.2345. Truncate your answer to  decimal places.

**Input Format**

The **STATION** table is described as follows:



where *LAT\_N* is the northern latitude and *LONG\_W* is the western longitude.

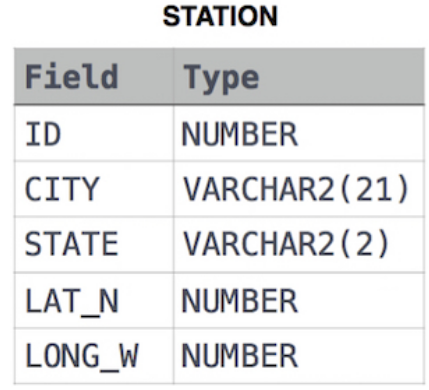
**Answer:** select truncate(max(LAT\_N), 4) from station

where LAT\_N < 137.2345

**Q.33:** Query the *Western Longitude* (*LONG\_W*) for the largest *Northern Latitude* (*LAT\_N*) in **STATION** that is less than 137.2345. Round your answer to  decimal places.

**Input Format**

The **STATION** table is described as follows:



where *LAT\_N* is the northern latitude and *LONG\_W* is the western longitude.

**Answer:** select round(LONG\_W, 4) from station

where LAT\_N < 137.2345

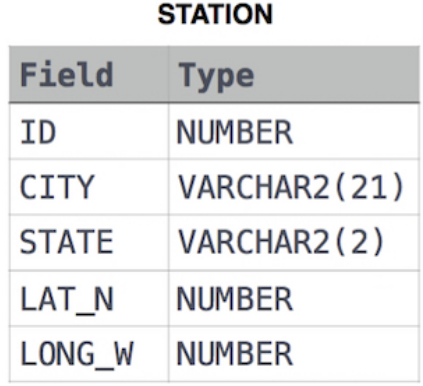
order by LAT\_N desc

limit 1

**Q.34:** Query the smallest *Northern Latitude* (*LAT\_N*) from **STATION** that is greater than 38.7780. Round your answer to  decimal places.

**Input Format**

The **STATION** table is described as follows:



where *LAT\_N* is the northern latitude and *LONG\_W* is the western longitude.

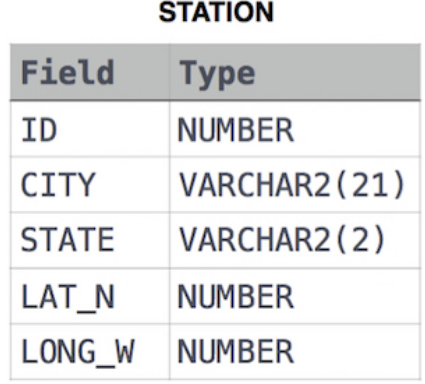
**Answer:** select round(min(LAT\_N),4) from station

where LAT\_N > 38.7780

**Q.35:** Query the *Western Longitude* (*LONG\_W*)where the smallest *Northern Latitude* (*LAT\_N*) in **STATION** is greater than 38.7780. Round your answer to  decimal places.

**Input Format**

The **STATION** table is described as follows:



where *LAT\_N* is the northern latitude and *LONG\_W* is the western longitude.

**Answer:** select round(LONG\_W, 4) from station

where LAT\_N > 38.7780

order by LAT\_N asc

limit 1

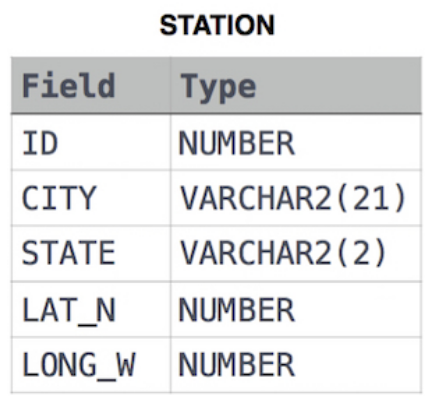
**Q.36:** Consider P1(a,b) and P2(c,d) to be two points on a *2D* plane.

* happens to equal the minimum value in *Northern Latitude* (*LAT\_N* in **STATION**).
* happens to equal the minimum value in *Western Longitude* (*LONG\_W* in **STATION**).
* happens to equal the maximum value in *Northern Latitude* (*LAT\_N* in **STATION**).
* happens to equal the maximum value in *Western Longitude* (*LONG\_W* in **STATION**).

Query the [Manhattan Distance](https://xlinux.nist.gov/dads/HTML/manhattanDistance.html) between points P1 and P2 and round it to a scale of 4 decimal places.

**Input Format**

The **STATION** table is described as follows:



where *LAT\_N* is the northern latitude and *LONG\_W* is the western longitude.

**Answer:** select

round(abs(min(LAT\_N) - max(LAT\_N)) + abs(min(LONG\_W) - max(LONG\_W)), 4)

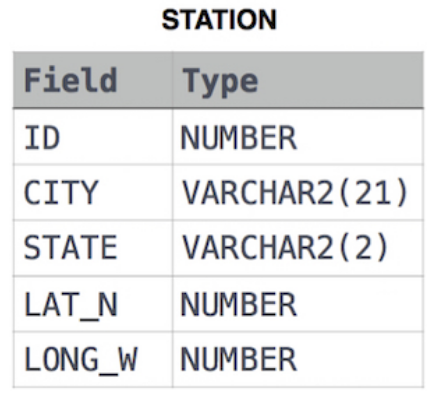
from station

**Q.37:** Consider P1(a,c) and P2(b,d) to be two points on a 2D plane where (a,b) are the respective minimum and maximum values of *Northern Latitude* (*LAT\_N*) and (c,d) are the respective minimum and maximum values of *Western Longitude* (*LONG\_W*) in **STATION**.

Query the [Euclidean Distance](https://en.wikipedia.org/wiki/Euclidean_distance) between points P1 and P2 and *format your answer* to display 4 decimal digits.

**Input Format**

The **STATION** table is described as follows:



where *LAT\_N* is the northern latitude and *LONG\_W* is the western longitude.

**Answer:** select

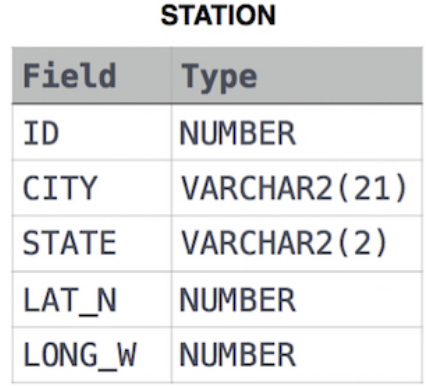
round(sqrt(power(min(LAT\_N) - max(LAT\_N),2) +

power(min(LONG\_W) - max(LONG\_W),2)), 4)

from station

**Q.38:** A [*median*](https://en.wikipedia.org/wiki/Median) is defined as a number separating the higher half of a data set from the lower half. Query the *median* of the *Northern Latitudes* (*LAT\_N*) from **STATION** and round your answer to  decimal places.

**Input Format**

The **STATION** table is described as follows:

where *LAT\_N* is the northern latitude and *LONG\_W* is the western longitude.

**Answer:** select round(avg(LAT\_N), 4) as median

from

(

select LAT\_N, row\_number() over(order by LAT\_N) as rn

from station

) as subq

where

rn = (select ceil((count(rn)+1)/2) from station)

**Q.39:** You are given a table, *Functions*, containing two columns: *X*and *Y*.

A screenshot of a computer

AI-generated content may be incorrect.

Two pairs *(X1, Y1)* and *(X2, Y2)* are said to be *symmetric* *pairs* if *X1 = Y2* and *X2 = Y1*.

Write a query to output all such *symmetric* *pairs* in ascending order by the value of *X*. List the rows such that *X1 ≤ Y1*.

**Answer:** select f1.X, f1.Y from functions f1, functions f2

where f1.X = f2.Y and f1.Y = f2.X

group by f1.X, f1.Y

having count(f1.X) > 1 or f1.X < f1.Y

order by f1.X asc

**Output:**

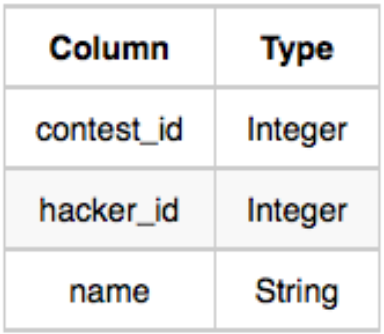
1. **2 24**
2. **4 22**
3. **5 21**
4. **6 20**
5. **8 18**
6. **9 17**
7. **11 15**
8. **13 13**

**Q.40:** Samantha interviews many candidates from different colleges using coding challenges and contests. Write a query to print the *contest\_id*, *hacker\_id*, *name*, and the sums of *total\_submissions*, *total\_accepted\_submissions*, *total\_views*, and *total\_unique\_views* for each contest sorted by *contest\_id*. Exclude the contest from the result if all four sums are .

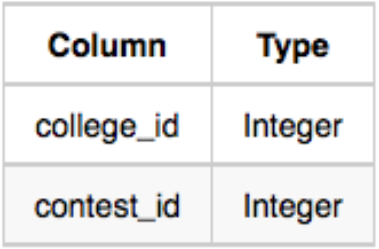
**Note:** A specific contest can be used to screen candidates at more than one college, but each college only holds  screening contest.

**Input Format**

The following tables hold interview data:

* *Contests:* The *contest\_id* is the id of the contest, *hacker\_id* is the id of the hacker who created the contest, and *name* is the name of the hacker.

*Colleges:* The *college\_id* is the id of the college, and *contest\_id* is the id of the contest that Samantha used to screen the candidates.



*Challenges:* The *challenge\_id* is the id of the challenge that belongs to one of the contests whose contest\_id Samantha forgot, and *college\_id* is the id of the college where the challenge was given to candidates.

A screenshot of a computer

AI-generated content may be incorrect.

*View\_Stats:* The *challenge\_id* is the id of the challenge, *total\_views* is the number of times the challenge was viewed by candidates, and *total\_unique\_views* is the number of times the challenge was viewed by unique candidates.

A white table with black text

AI-generated content may be incorrect.

*Submission\_Stats:* The *challenge\_id* is the id of the challenge, *total\_submissions* is the number of submissions for the challenge, and *total\_accepted\_submission* is the number of submissions that achieved full scores.

A screenshot of a computer

AI-generated content may be incorrect.

**Answer:** select con.contest\_id, con.hacker\_id, con.name, sum(TS), sum(TAS), sum(TV), sum(TUV)

from Contests as con

inner join Colleges as col

on con.contest\_id = col.contest\_id

inner join Challenges as cha

on cha.college\_id = col.college\_id

left join (

select challenge\_id, sum(total\_views) as TV, sum(total\_unique\_views) as TUV

from View\_Stats

group by challenge\_id

) as vs

on cha.challenge\_id = vs.challenge\_id

left join (

select challenge\_id, sum(total\_submissions) as TS, sum(total\_accepted\_submissions) as TAS

from Submission\_Stats

group by challenge\_id

) as ss

on cha.challenge\_id = ss.challenge\_id

group by con.contest\_id, con.hacker\_id, con.name

order by con.contest\_id

**Output:**

1. **845 579 Rose 1987 580 1635 566**
2. **858 1053 Angela 703 160 1002 384**
3. **883 1055 Frank 1121 319 1217 338**
4. **1793 2655 Patrick 1337 360 1216 412**
5. **2374 2765 Lisa 2733 815 3368 904**
6. **2963 2845 Kimberly 4306 1221 3603 1184**
7. **3584 2873 Bonnie 2492 652 3019 954**
8. **4044 3067 Michael 1323 449 1722 528**
9. **4249 3116 Todd 1452 376 1767 463**
10. **4269 3256 Joe 1018 372 1766 530**
11. **4483 3386 Earl 1911 572 1644 477**
12. **4541 3608 Robert 1886 516 1694 504**
13. **4601 3868 Amy 1900 639 1738 548**
14. **4710 4255 Pamela 2752 639 2378 705**
15. **4982 5639 Maria 2705 759 2558 711**
16. **5913 5669 Joe 2646 790 3181 835**
17. **5994 5713 Linda 3369 967 3048 954**
18. **6939 6550 Melissa 2842 859 3574 1004**
19. **7266 6947 Carol 2758 665 3044 835**
20. **7280 7030 Paula 1963 554 886 259**
21. **7484 7033 Marilyn 3217 934 3795 1061**
22. **7734 7386 Jennifer 3780 1015 3637 1099**
23. **7831 7787 Harry 3190 883 2933 1012**
24. **7862 8029 David 1738 476 1475 472**
25. **8812 8147 Julia 1044 302 819 266**
26. **8825 8438 Kevin 2624 772 2187 689**
27. **9136 8727 Paul 4205 1359 3125 954**
28. **9613 8762 James 3438 943 3620 1046**
29. **10568 8802 Kelly 1907 620 2577 798**
30. **11100 8809 Robin 1929 613 1883 619**
31. **12742 9203 Ralph 1523 413 1344 383**
32. **12861 9644 Gloria 1596 536 2089 623**
33. **12865 10108 Victor 2076 597 1259 418**
34. **13503 10803 David 924 251 584 167**
35. **13537 11390 Joyce 1381 497 1784 538**
36. **13612 12592 Donna 1981 550 1487 465**
37. **14502 12923 Michelle 1510 463 1830 545**
38. **14867 13017 Stephanie 2471 676 2291 574**
39. **15164 13256 Gerald 2570 820 2085 607**
40. **15804 13421 Walter 1454 459 1396 476**
41. **15891 13569 Christina 2188 710 2266 786**
42. **16063 14287 Brandon 1804 580 1621 521**
43. **16415 14311 Elizabeth 4535 1366 3631 1071**
44. **18477 14440 Joseph 1320 391 1419 428**
45. **18855 16973 Lawrence 2967 1020 3371 1011**
46. **19097 17123 Marilyn 2956 807 2554 750**
47. **19575 17562 Lori 2590 863 2627 760**

**Q.41:** 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*.

A screenshot of a computer

AI-generated content may be incorrect.

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.

**Answer:** select N,

CASE

WHEN P is null THEN 'Root'

WHEN N in (select P from BST) THEN 'Inner'

ELSE 'Leaf'

END

from BST

order by N

**Q.42:** Amber's conglomerate corporation just acquired some new companies. Each of the companies follows this hierarchy:

A group of black text

AI-generated content may be incorrect.

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*.

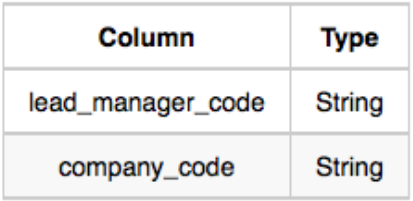
**Input Format**

The following tables contain company data:

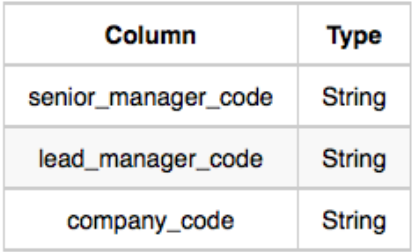
* *Company:* The *company\_code* is the code of the company and *founder* is the founder of the company.

A screenshot of a computer

AI-generated content may be incorrect.

*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.

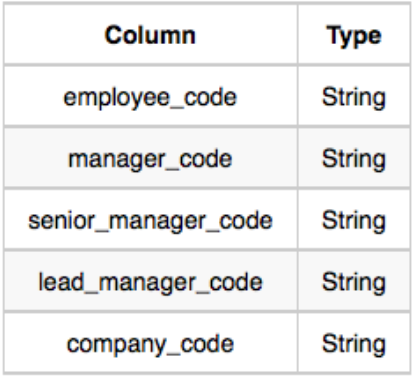


*Manager:* The *manager\_code* is the code of the manager, the *senior\_manager\_code* is the code of its senior manager, the *lead\_manager\_code* is the code of its lead manager, and the *company\_code* is the code of the working company.

A table of code

AI-generated content may be incorrect.

*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.



**Answer:** select c.company\_code, c.founder, count(distinct e.lead\_manager\_code), count(distinct e.senior\_manager\_code), count(distinct e.manager\_code), count(distinct e.employee\_code)

from Company as c

inner join Employee as e

on c.company\_code = e.company\_code

group by c.company\_code, c.founder

order by c.company\_code asc