Table point\_2d holds the coordinates (x,y) of some unique points (more than two) in a plane.

 Write a query to find the shortest distance between these points rounded to 2 decimals.

 | x | y |

|----|----|

| -1 | -1 |

| 0 | 0 |

| -1 | -2 |

 The shortest distance is 1.00 from point (-1,-1) to (-1,2). So the output should be:

 | shortest |

|----------|

| 1.00 |

MY SQL:

SELECT

MIN(ROUND(sqrt((power((p1.x - p2.x),2) + power((p1.y - p2.y),2))),2)) shortest

FROM

point\_2d p1, point\_2d p2

WHERE

sqrt((power((p1.x - p2.x),2) + power((p1.y - p2.y),2))) > 0

OR

BETTER ONE:

SELECT TOP 1 ROUND(SQRT(POWER(T1.X-T2.X,2)+POWER(T1.Y-T2.Y,2)),2) AS **shortest** FROM point\_2d T1 **JOIN** point\_2d T2 ON CONCAT(T1.X,T1.Y)!=CONCAT(T2.X,T2.Y) **ORDER BY** SQRT(POWER(T1.X-T2.X,2)+POWER(T1.Y-T2.Y,2))

It looks like MIN() is the way to go - it's faster in the worst case, indistinguishable in the best case, is standard SQL and most clearly expresses the value you're trying to get. The only case where it seems that using SORT and LIMIT would be desirable would be, as [mson](https://stackoverflow.com/users/36902/mson) mentioned, where you're writing a general operation that finds the top or bottom N values from arbitrary columns and it's not worth writing out the special-case operation.

A U.S graduate school has students from Asia, Europe and America. The students' location information are stored in table student as below.

| name | continent |

|--------|-----------|

| Jack | America |

| Pascal | Europe |

| Xi | Asia |

| Jane | America |

[Pivot](https://en.wikipedia.org/wiki/Pivot_table) the continent column in this table so that each name is sorted alphabetically and displayed underneath its corresponding continent. The output headers should be America, Asia and Europe respectively. It is guaranteed that the student number from America is no less than either Asia or Europe.

For the sample input, the output is:

| America | Asia | Europe |

|---------|------|--------|

| Jack | Xi | Pascal |

| Jane | | |

# Write your MySQL query statement below

SELECT America,Asia,Europe FROM

(SELECT

@a:= @a+1 rowid, name as America FROM student, (SELECT @a:=0)inc

WHERE continent = 'America' ORDER BY name)Am

LEFT JOIN

(SELECT

@b:= @b+1 rowid, name as Asia FROM student, (SELECT @b:=0)inc

WHERE continent = 'Asia' ORDER BY name)Asia ON Am.rowid = Asia.rowid

LEFT JOIN

(SELECT

@c:= @c+1 rowid, name as Europe FROM student, (SELECT @c:=0)inc

WHERE continent = 'Europe' ORDER BY name)Eu ON Am.rowid=Eu.rowid

# Write your MySQL query statement below

SELECT MAX(num)'num' FROM (SELECT \* FROM number GROUP BY num HAVING COUNT(num)=1) AS temp

#

SELECT

score,

(select count(\*) FROM (SELECT distinct score FROM scores)s WHERE s.score >= scores.score) Rank

FROM scores

ORDER BY score desc

select [d.Name](http://d.name/) as Department, [e.Name](http://e.name/) as Employee, e.Salary from Employee as e  
inner join Department d on e.DepartmentId = [d.Id](http://d.id/)  
where (select count(distinct e1.Salary) from Employee e1 where e1.Salary > e.Salary  
and e1.DepartmentId = e.DepartmentId) < 3  
order by [d.Name](http://d.name/), e.Salary DESC;