612. Shortest Distance in a Plane

June 22, 2017 | 16K views
Average Rating: 4.67 (7 votes)

Previous Next

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     |
```

Note: The longest distance among all the points are less than 10000.

Solution

Approach 1: Using `SQRT`, `POW()` functions and math knowledge [Accepted]

Intuition

Calculate the distances between each two points and then display the smallest one.

Algorithm

The euclidean distance between two points P1(x1,y1) and P2(x2, y2) in two dimensions is defined as $\sqrt{(x1-x2)^2+(y1-y2)^2}$. So in order to get the distances, we can join this table with itself, and then utilize the built-in function `POW()` and `SQRT()` like below.

```
SELECT
    p1.x,
    p1.y,
    p2.x,
    p2.y,
    SQRT(POW(p1.x - p2.x, 2) + POW(p1.y - p2.y, 2))) AS distance
FROM
    point_2d p1
        JOIN
    point_2d p2 ON p1.x != p2.x OR p1.y != p2.y
;
```

> Note: - The condition 'p1.x != p2.x OR p2.y != p2.y' is to avoid calculating the distance of a point with itself. Otherwise, the minimum distance will be always zero. - The columns p1.x, p1.y, p2.x and p2.y are for demonstrating. They are not necessary for the final solution.

So the output would be as below after running this code on the sample data.

```
| x  | y  | x  | y  | distance           |
|----|----|----|----|--------------------|
| 0  | 0  | -1 | -1 | 1.4142135623730951 |
| -1 | -2 | -1 | -1 | 1                  |
| -1 | -1 | 0  | 0  | 1.4142135623730951 |
| -1 | -2 | 0  | 0  | 2.23606797749979   |
| -1 | -1 | -1 | -2 | 1                  |
| 0  | 0  | -1 | -2 | 2.23606797749979   |
```

At last, choose the minimum distance and round it to 2 decimals as required.

MySQL

```
SELECT
    ROUND(SQRT(MIN((POW(p1.x - p2.x, 2) + POW(p1.y - p2.y, 2)))), 2) AS shortest
FROM
    point_2d p1
        JOIN
    point_2d p2 ON p1.x != p2.x OR p1.y != p2.y
;
```

> Note: To put the MIN() inside of SQRT() will slightly improve the performance.

Approach 2: Optimize to avoid reduplicate calculations [Accepted]

Intuition

It is unnecessary to calculate the distance between all points to all other points since some of them may already be done. So how to avoid the reduplicate calculations?

Algorithm

When join the table with itself, we can claim to only calculate the distance between one point to another point in a certain rule such ponts with bigger x value. By following this rule, we can avoid quite a lot of reduplicate calculations.

```
SELECT
    t1.x,
    t1.y,
    t2.x,
    t2.y,
    SQRT(POW(t1.x - t2.x, 2) + POW(t1.y - t2.y, 2))) AS distance
FROM
    point_2d t1
        JOIN
    point_2d t2 ON (t1.x <= t2.x AND t1.y < t2.y)
        OR (t1.x < t2.x AND t1.y > t2.y)
        OR (t1.x < t2.x AND t1.y = t2.y)
;
```

The output is as below for the sample data. You may notice that there are only 4 records, 1/3 less than the previous solution.

```
| x  | y  | x  | y  | distance           |
|----|----|----|----|--------------------|
| -1 | -2 | -1 | -1 | 1                  |
| -1 | -1 | 0  | 0  | 1.4142135623730951 |
| -1 | -2 | 0  | 0  | 2.23606797749979   |
| -1 | -1 | -1 | -2 | 1                  |
```

> Note: The best case is to compare n*(n-1)/2 times, but practically it is not always true considering two points may have same x value or y value. In this case, you may notice the distance between (-1, -2) and (-1, -1) appearing twice in the first and last line in the output.

Here comes the solution to select the shortest distance and round to two decimals.

MySQL

```
SELECT
    ROUND(SQRT(MIN((POW(p1.x - p2.x, 2) + POW(p1.y - p2.y, 2)))),2) AS shortest
FROM
    point_2d p1
        JOIN
    point_2d p2 ON (p1.x <= p2.x AND p1.y < p2.y)
        OR (p1.x <= p2.x AND p1.y > p2.y)
        OR (p1.x < p2.x AND p1.y = p2.y)
;
```

Rate this article:

Previous Next

Comments: 15

Sort By

Type comment here... (Markdown is supported)

Preview Post

tyumoeva ★ 83 July 17, 2018 12:12 AM

select min(round(sqrt(power(t1.x-t2.x,2) + power(t1.y-t2.y,2)),2)) shortest
from point_2d t1
cross join point_2d t2
on (t1.x = t2.x and t1.y<t2.y)

13 Share Reply

Mr-Bin ★ 123 January 5, 2019 1:04 AM

```
select round(min(sqrt(power(p1.x - p2.x, 2) + power(p1.y - p2.y, 2))), 2) as shortest
from point_2d p1, point_2d p2
where p1.x < p2.x or (p1.x = p2.x and p1.y < p2.y)
```

Read More

5 Share Reply

SHOW 1 REPLY

rogerye ★ 23 May 31, 2018 8:38 PM Report

Thanks for sharing the knowledge! Two comments

1. There is a typo in the description " The shortest distance is 1.00 from point (-1,-1) to (-1,2)", which (-1, 2) should be (-1, -2).
2. I found out putting MIN outside of SQRT function will result in a better performance.

Read More

3 Share Reply

vasanthlilly91 ★ 5 June 1, 2018 9:22 AM

select sqrt(select (power((p1.x-p2.x),2)+ power((p1.y-p2.y),2))
as A from point_2d
p1 join point_2d p2
on p1.x!=p2.x or p1.y!=p2.y
order by A limit 1

Read More

1 Share Reply

adityajhanwar ★ 45 July 30, 2019 9:50 PM

```
select round(min(sqrt(pow(b.x-a.x,2)+pow(b.y-a.y,2))),2) as 'shortest'
from point_2d a, point_2d b
where (a.x, a.y) < (b.x, b.y)
```

0 Share Reply

aaronmok ★ 0 March 26, 2019 10:21 PM Report

```
SELECT
        MIN(ROUND(SQRT(POWER(p1.x-p2.x,2)-POWER(p1.y-p2.y,2))),2)) shortest
FROM
    point_2d p1
```

Read More

0 Share Reply

kailin026 ★ 0 February 19, 2019 12:18 AM

SELECT
ROUND(SQRT(MIN(POW(p1.x - p2.x, 2) + POW(p1.y - p2.y, 2))), 2) AS shortest
FROM
point_2d p1
JOIN

Read More

0 Share Reply

SHOW 1 REPLY

yuhui4 ★ 35 January 12, 2019 7:13 AM

select round(min(sqrt(power(p2.y-p1.y, 2) + power(p2.x-p1.x, 2))), 2) shortest
from point_2d p1, point_2d p2
where not (p1.x = p2.x and p1.y = p2.y);

0 Share Reply

zfj ★ 15 August 4, 2018 2:36 AM

select min(round(sqrt(pow(t1.x-t2.x,2)+pow(t1.y-t2.y,2)),2)) as shortest
from point_2d t1, point_2d t2
where t1.x < t2.x and t1.y <> t2.y

0 Share Reply

ehsanfar ★ 6 April 26, 2018 3:46 AM Report

select round(pow(pow(p2.x-p1.x,2) + pow(p2.y-p1.y,2), 0.5),2) as shortest
from point_2d as p1, point_2d as p2
where p1.x != p2.x or p1.y != p2.y
order by shortest
limit 1

Read More

0 Share Reply