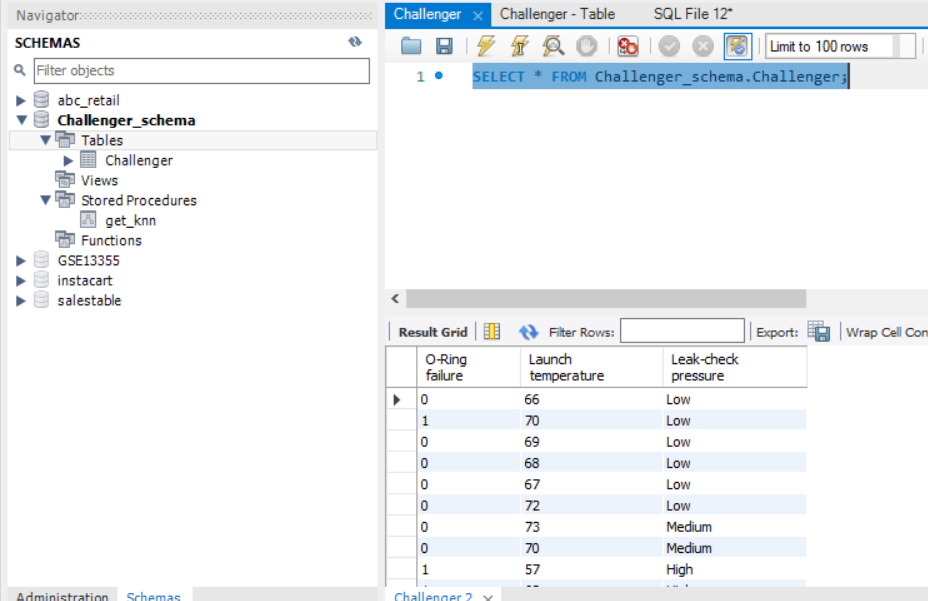
## Import the attached file (Challenger.csv) to your MySQL.



## Create a stored procedure call KNN with five inputs (@table, @target, @predictor, @value, @k).

CREATE DEFINER=`admin`@`%` PROCEDURE `get\_knn`(

    in tbl varchar(100), tar varchar(100), predictor varchar(100), val varchar(10), k int

)

-- if use col `Launch temperature` as @predictor, we may need to use cast to deal with the value

-- elif use col `Leak-check pressure` we need to convert low/medium/high to int label 1/2/3

--   in order to calculate the Euclidean distance

BEGIN

    -- create temp table for knn data

    -- drop temp table if exists

    DROP TABLE IF EXISTS Challenger\_schema.temp1;

    CREATE TABLE Challenger\_schema.temp1

    (

        `O-Ring failure` TINYINT(1) NOT NULL,

        `Launch temperature` INT NOT NULL,

        `Leak-check pressure` VARCHAR(10) NOT NULL,

        `ED` DOUBLE NOT NULL

    );

    -- swtich between predictor

    -- calculate knn distance and return a temp view

    IF (predictor='Launch temperature') THEN

        -- @predictor=`Launch temperature`

        -- Find @k nearest neighbors and insert the data

        SET @sql1 = CONCAT(

            'INSERT INTO Challenger\_schema.temp1(

                    `O-Ring failure`, `Launch temperature`, `Leak-check pressure`, `ED`

                )

            SELECT `O-Ring failure`, `Launch temperature`, `Leak-check pressure`, sqrt(pow((`',

            predictor,

            '` - CAST(',

            val,

            ' as signed)), 2)) AS `ED`

                FROM Challenger\_schema.',

            tbl,

            ' ORDER BY `ED` ASC LIMIT ',

            k,

            ';'

        );

        prepare getsql1 from @sql1;

        execute getsql1;

    ELSEIF (predictor='Leak-check pressure') THEN

        -- Find @k nearest neighbors and insert the data

        SET @sql1 = CONCAT(

            'INSERT INTO Challenger\_schema.temp1(

                        `O-Ring failure`, `Launch temperature`, `Leak-check pressure`, `ED`

                    )

            SELECT `O-Ring failure`, `Launch temperature`, `Leak-check pressure`, sqrt(pow((CASE

                    WHEN ''',

            val,

            '''=''Low'' THEN `temp\_predictor` - 1 WHEN ''',

            val,

            '''=''Medium'' THEN `temp\_predictor` - 2 WHEN ''',

            val,

            '''=''High'' THEN `temp\_predictor` - 3

                ELSE `temp\_predictor` - (-1)

            END

                    ),

                    2

                )) AS `ED`

                    FROM (

                SELECT

                    \*,

                    (

                CASE

                    WHEN `',

            predictor,

            '`=''Low'' THEN 1 WHEN `',

            predictor,

            '`=''Medium'' THEN 2 WHEN `',

            predictor,

            '`=''High'' THEN 3

                    ELSE -1

                end

                    ) as `temp\_predictor`

                from Challenger\_schema.',

            tbl,

            ') converter\_tbl

                    ORDER BY `ED` ASC LIMIT ',

            k,

            ';'

        );

        prepare getsql1 from @sql1;

        execute getsql1;

    ELSE

        SELECT 'Error: Value of @predictor is not valid.';

    END IF;

    -- print @k nearest neighbors

    SELECT `O-Ring failure`, `Launch temperature`, `Leak-check pressure`

    FROM Challenger\_schema.temp1;

    -- Return the proportion of N and Y.

    SET @prosql = CONCAT(

        'SELECT `',

        tar,

        '`, CONCAT(ltrim(

            CAST(count(`',

        tar,

        '`)\*100.0/(SELECT count(`',

        tar,

        '`) FROM Challenger\_schema.temp1) AS DEC(18,2))

        ),

        ''%'') as `proportion`

        FROM Challenger\_schema.temp1

        GROUP BY `',

        tar,

        '`;'

    );

    prepare getprosql from @prosql;

    execute getprosql;

    -- Return the majority class (N or Y) of @target

    SET @sub\_countsql = CONCAT(

        '(

            SELECT

                `',

        tar,

        '`,

        count(`',

        tar,

        '`) as `proportion`

        FROM Challenger\_schema.temp1

        GROUP BY `',

        tar,

        '`

        ) count\_pro'

    );

    SET @majsql = CONCAT(

        'SELECT `',

        tar,

        '` FROM ',

        @sub\_countsql,

        ' WHERE `proportion` IN ( '

        'SELECT MAX(`proportion`) AS `proportion`

        FROM ',

        @sub\_countsql,

        ');'

    );

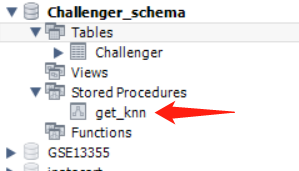
    prepare getmajsql from @majsql;

    execute getmajsql;

    -- drop temp table

    DROP TABLE IF EXISTS Challenger\_schema.temp1;

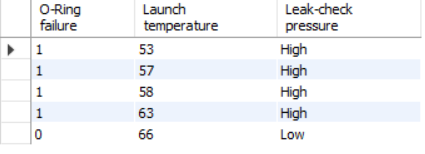
END



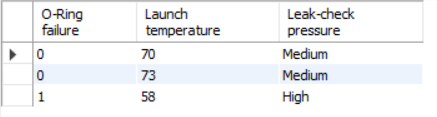
## Find @k nearest neighbors to @value by measuring its distance to values in @predictor column.

Example:

CALL `Challenger\_schema`.`get\_knn`('Challenger', 'O-Ring failure', 'Launch temperature', '30', 5);



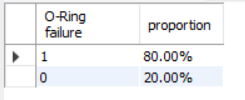
CALL `Challenger\_schema`.`get\_knn`('Challenger', 'O-Ring failure', 'Leak-check pressure', 'Medium', 3);

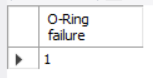


## Return the majority class (N or Y) of @target or the proportion of N and Y.

Example:

CALL `Challenger\_schema`.`get\_knn`('Challenger', 'O-Ring failure', 'Launch temperature', '30', 5);





CALL `Challenger\_schema`.`get\_knn`('Challenger', 'O-Ring failure', 'Leak-check pressure', 'Medium', 3);

