coursera

The previous video described how aggregate expressions handle **NULL** values differently than non-aggregate (scalar) expressions. This reading describes the reasons for this difference, and warns about how it can cause misinterpretations.

For a scalar expression, it would be misleading to report anything except **NULL** in individual row values containing **NULL**s in the operands or arguments of the expression. (Review the lesson, "Working with Missing Values," in Week 3 of this course for more about why this is so.)

However, for aggregate expressions, if **NULL**s were not ignored, then just one **NULL** value in a large group of rows would cause the query to return a **NULL** result as the aggregate for the whole group. By ignoring the **NULL** values, aggregate expressions are able to return meaningful results even when there are **NULL** values in the groups.

But sometimes this behavior can lead to misinterpretations, especially with sparse data. For example, if you compute the average of a column in a table with ten million rows, but only three of those rows have a non-**NULL** value in the column you're averaging, then the query would return a non-**NULL** average in the result. This might mislead you into thinking that this average provides meaningful information about all ten million rows, when it reality the average comes from only three rows, and there is probably no reason to believe it is representative of all ten million rows.

Therefore, it is important to explicitlycheckfor **NULL** values and handle them in your queries, instead of just relying on aggregate expressions to ignore them.

One way to do this is to use an aggregate expression like:

SUM(*column* IS NOT NULL)

to return the number of rows in which *column* is non-**NULL**. In this expression, ***column* IS NOT NULL** evaluates to **true** (**1**) or **false** (**0**) for each row, and the **SUM** function adds these 1s and 0s up and returns the number of rows in which ***column* IS NOT NULL**.

For example, when you run the following query, the second column in the result tells you exactly how many non-**NULL** values were used to compute each of the averages in the third column:

SELECT shop, SUM(price IS NOT NULL), AVG(price) FROM inventory GROUP BY shop;

shop	**SUM(price IS NOT NULL)**	**AVG(price)**
shop	**SUM(price IS NOT NULL)**	**AVG(price)**
Dicey	2	13.99