

Out-of-Range Values

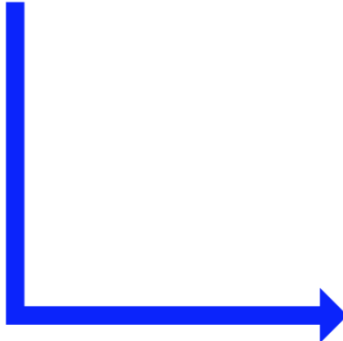
Both Hive and Impala return NULL for DECIMAL values that are out of range for the target data type (such as 23.63 in a DECIMAL(3,2) column). They also both return -Infinity or Infinity for FLOAT and DOUBLE types when the value is too large, and zero when the value is close to zero but too small for the data type's range. (These zero values may be rendered in slightly different ways, for example: 0, 0.0, -0, or -0.0.)

However, there's an important difference between how Hive and Impala handle out-of-range values in integer columns: Hive returns NULL for out-of-range integer values, whereas Impala returns the minimum or maximum value for the column's specific integer data type. The example shown below illustrates this.

Rashida	29
Hugo	17
Abigail	
Esma	129
Kenji	-999

```
CREATE TABLE names_and_ages  
(name STRING,  
  age TINYINT)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY '\t';
```

```
SELECT age FROM names_and_ages;
```



age	age
29	29
17	17
NULL	NULL
NULL	127
NULL	-128

In this example, there is a table that includes a column named age with data type TINYINT. The TINYINT data type can represent integer values from -128 to +127. But the data file for this table (represented by the yellow box) contains two erroneous age values that are outside this range. The age value for Esma is 129, which is above the maximum value for the TINYINT type. The age value for Kenji is -999, which is below the minimum value for the TINYINT type.

When you query this table with Hive, it returns NULL for these two out-of-range values, but Impala behaves differently. Impala returns 127 for Esma's age (127 is the maximum value for the TINYINT type). And Impala returns -128 for Kenji's age (-128 is the minimum value for the TINYINT type).

Why does Impala behave differently than Hive in this situation? Notice in the example shown here that Abigail's age is missing in the data file in HDFS. The Impala query results allow you to distinguish between the missing value of Abigail's age (NULL) and the out-of-range values of Esma's and Kenji's ages, whereas the Hive query results do not allow you to distinguish between these.

However, one implication of this is that you should choose a data type whose largest and smallest values *do not occur* in your data. For example, if your data contains integers ranging from -127 to +126, then it would be fine to use the TINYINT data type. In that case, a value of -128 or +127 in the query result will unambiguously indicate an out-of-range value. But if your data contains integers ranging from -128 to +127, then you should choose SMALLINT to avoid ambiguity between the *actual* values -128 and +127 and out-of-range values.