Order, Sort, Cluster, and Distribute By

This describes the syntax of SELECT clauses ORDER BY, SORT BY, CLUSTER BY, and DISTRIBUTE BY. See Select Syntax for general information.

Syntax of Order By

The ORDER BY syntax in Hive QL is similar to the syntax of ORDER BY in SQL language.

```sql
orderBy: ORDER BY colName colOrder? colNullOrder? (',', colName colOrder? colNullOrder?)*
query: SELECT expression (',', expression)* FROM src orderBy
```

There are some limitations in the "order by" clause. In the strict mode (i.e., hive.mapred.mode=strict), the order by clause has to be followed by a "limit" clause. The limit clause is not necessary if you set hive.mapred.mode to nonstrict. The reason is that in order to impose total order of all results, there has to be one reducer to sort the final output. If the number of rows in the output is too large, the single reducer could take a very long time to finish.

Note that columns are specified by name, not by position number. However in Hive 0.11.0 and later, columns can be specified by position when configured as follows:

- For Hive 0.11.0 through 2.1.x, set hive.groupby.orderby.position.alias to true (the default is false).
- For Hive 2.2.0 and later, hive.orderby.position.alias is true by default.

The default sorting order is ascending (ASC).

In Hive 2.1.0 and later, specifying the null sorting order for each of the columns in the "order by" clause is supported. The default null sorting order for ASC order is NULLS FIRST, while the default null sorting order for DESC order is NULLS LAST.

In Hive 3.0.0 and later, order by without limit in subqueries and views will be removed by the optimizer. To disable it, set hive.remove.orderby.in.subquery to false.

Syntax of Sort By

The SORT BY syntax is similar to the syntax of ORDER BY in SQL language.

```sql
sortBy: SORT BY colName colOrder? (',', colName colOrder?)*
query: SELECT expression (',', expression)* FROM src sortBy
```

Hive uses the columns in SORT BY to sort the rows before feeding the rows to a reducer. The sort order will be dependent on the column types. If the column is of numeric type, then the sort order is also in numeric order. If the column is of string type, then the sort order will be lexicographical order.

In Hive 3.0.0 and later, sort by without limit in subqueries and views will be removed by the optimizer. To disable it, set hive.remove.orderby.in.subquery to false.

Difference between Sort By and Order By

Hive supports SORT BY which sorts the data per reducer. The difference between "order by" and "sort by" is that the former guarantees total order in the output while the latter only guarantees ordering of the rows within a reducer. If there are more than one reducer, "sort by" may give partially ordered final results.

Note: It may be confusing as to the difference between SORT BY alone of a single column and CLUSTER BY. The difference is that CLUSTER BY partitions by the field and SORT BY if there are multiple reducers partitions randomly in order to distribute data (and load) uniformly across the reducers.

Basically, the data in each reducer will be sorted according to the order that the user specified. The following example shows
SELECT key, value FROM src SORT BY key ASC, value DESC

The query had 2 reducers, and the output of each is:

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

### Setting Types for Sort By

After a transform, variable types are generally considered to be strings, meaning that numeric data will be sorted lexicographically. To overcome this, a second `SELECT` statement with casts can be used before using `SORT BY`.

```sql
FROM (FROM (FROM src
SELECT TRANSFORM(value)
USING 'mapper'
AS value, count) mapped
SELECT cast(value as double) AS value, cast(count as int) AS count
SORT BY value, count) sorted
SELECT TRANSFORM(value, count)
USING 'reducer'
AS whatever
```

### Syntax of Cluster By and Distribute By

*Cluster By* and *Distribute By* are used mainly with the Transform/Map-Reduce Scripts. But, it is sometimes useful in `SELECT` statements if there is a need to partition and sort the output of a query for subsequent queries.

*Cluster By* is a short-cut for both *Distribute By* and *Sort By*.

Hive uses the columns in *Distribute By* to distribute the rows among reducers. All rows with the same *Distribute By* columns will go to the same reducer. However, *Distribute By* does not guarantee clustering or sorting properties on the distributed keys.

For example, we are Distributing By `x` on the following 5 rows to 2 reducer:

```
x1
x2
x4
x3
x1
```

Reduction 1 got

```
x1
x2
x1
```

Reduction 2 got

```
x4
x3
```

Note that all rows with the same key `x1` is guaranteed to be distributed to the same reducer (reducer 1 in this case), but they are not guaranteed to be clustered in adjacent positions.
In contrast, if we use \textit{Cluster By} \(x\), the two reducers will further sort rows on \(x\):

Reducer 1 got

\[
\begin{align*}
x1 \\
x1 \\
x2
\end{align*}
\]

Reducer 2 got

\[
\begin{align*}
x3 \\
x4
\end{align*}
\]

Instead of specifying \textit{Cluster By}, the user can specify \textit{Distribute By} and \textit{Sort By}, so the partition columns and sort columns can be different. The usual case is that the partition columns are a prefix of sort columns, but that is not required.

\begin{verbatim}
SELECT col1, col2 FROM t1 CLUSTER BY col1
SELECT col1, col2 FROM t1 DISTRIBUTE BY col1
SELECT col1, col2 FROM t1 DISTRIBUTE BY col1 SORT BY col1 ASC, col2 DESC
FROM {
  FROM pv_users
  MAP ( pv_users.userid, pv_users.date )
  USING 'map_script'
  AS c1, c2, c3
  DISTRIBUTE BY c2
  SORT BY c2, c1) map_output
INSERT OVERWRITE TABLE pv_users_reduced
  REDUCE ( map_output.c1, map_output.c2, map_output.c3 )
  USING 'reduce_script'
  AS date, count;
\end{verbatim}