**HIVEQL Manipulations**

Hive provides a CLI to write Hive queries using Hive Query Language (HiveQL).Generally, HiveQL syntax is similar to the SQL syntax that most data analysts are familiar with. Hive supports four file formats those are TEXTFILE, SEQUENCEFILE, ORC and RCFILE (Record Columnar File).

* For single user metadata storage Hive uses derby database and
* For multiple user Metadata or shared Metadata case Hive uses MYSQL

**Built-in operators**

Hive provides Built-in operators for Data operations to be implemented on the tables present inside Hive warehouse.

These operators are used for mathematical operations on operands, and it will return specific value as per the logic applied.

Types of Built-in Operators in HIVE are:

* Relational Operators
* Arithmetic Operators
* Logical Operators
* Operators on Complex types
* Complex type Constructors

**Relational Operators:**

We use Relational operators for relationship comparisons between two operands.

* Operators such as equals, Not equals, less than, greater than …etc
* The operand types are all number types in these Operators.

The following Table will give us details about Relational operators and its usage.

|  |  |  |
| --- | --- | --- |
| Built-in Operator | Description | Operand |
| X = Y | TRUE   if expression X is equivalent to expression Y   Otherwise FALSE. | It takes all primitive types |
| X != Y | TRUE  if expression X is not equivalent to expression Y  Otherwise FALSE. | It takes all primitive types |
| X < Y | TRUE  if expression X is less than expression Y   Otherwise FALSE. | It takes all primitive types |
| X <= Y | TRUE   if expression X is less than or equal to expression Y   Otherwise FALSE. | It takes all primitive types |
| X>Y | TRUE  if expression X is greater than expression Y  Otherwise FALSE. | It takes all primitive types |
| X>= Y | TRUE   if expression X is greater than or equal to expression Y  Otherwise FALSE. | It takes all primitive types |
| X IS NULL | TRUE if expression X evaluates to NULL otherwise FALSE. | It takes all types |
| X IS NOT NULL | FALSE   If expression X evaluates to NULL otherwise TRUE. | It takes all types |
| X LIKE Y | TRUE   If string pattern X matches to Y otherwise FALSE. | Takes only Strings |
| X RLIKE Y | NULL if X or Y is NULL, TRUE if any substring of X matches the[Java](https://www.guru99.com/java-tutorial.html)regular expression Y, otherwise FALSE. | Takes only Strings |
| X REGEXP Y | Same as RLIKE. | Takes only Strings |

**Arithmetic Operators:**

We use Arithmetic operators for performing arithmetic operations on operands

* Arithmetic operations such as addition, subtraction, multiplication and division between operands we use these Operators.
* The operand types all are number types in these Operators

*Sample Example:*

2 + 3 gives result 5.

In this example, '+' is theoperator and 2 and 3 are operands. The return value is 5

**Logical Operators:**

We use Logical operators for performing Logical operations on operands

* Logical operations such as AND, OR, NOT between operands we use these Operators.
* The operand types all are BOOLEAN type in these Operators

The following Table will give us details about Logical operators

|  |  |  |
| --- | --- | --- |
| Operators | Description | Operands |
| X AND Y | TRUE if both X and Y are TRUE, otherwise FALSE. | Boolean types only |
| X && Y | Same as X AND Y but here we using && symbol | Boolean types only |
| X OR Y | TRUE if either X or Y or both are TRUE, otherwise FALSE. | Boolean types only |
| X || Y | Same as X OR Y but here we using || symbol | Boolean types only |
| NOT X | TRUE if X is FALSE, otherwise FALSE. | Boolean types only |
| !X | Same as NOT X but here we using! symbol | Boolean types only |

**Operators on Complex types:**

The following Table will give us details about Complex Type Operators . These are operators which will provide a different mechanism to access elements in complex types.

|  |  |  |
| --- | --- | --- |
| Operators | Operands | Description |
| A[n] | A is an Array and n is an integer type | It will return nth element in the array A. The first element has index of 0 |
| M[key] | M is a Map<K, V> and key has type K | It will return the values belongs to the key in the map |

**Complex type Constructors:**

The following Table will give us details about Complex type Constructors. It will construct instances on complex data types. These are of complex data types such as Array, Map and Struct types in Hive.

|  |  |  |
| --- | --- | --- |
| Operators | Operands | Description |
| array | (val1, val2, ...) | It will create an array with the given elements as mentioned like val1, val2 |
| Create\_ union | (tag, val1, val2, ...) | It will create a union type with the values that is being mentioned to by the tag parameter |
| map | (key1, value1, key2, value2, ...) | It will create a map with the given key/value pairs mentioned in operands |