**String Functions in Hive**

The string functions in Hive are listed below:  
  
**ASCII( string str )**  
  
The ASCII function converts the first character of the string into its numeric ascii value.

Example1: ASCII('hadoop') returns 104

Example2: ASCII('A') returns 65

**CONCAT( string str1, string str2... )**  
  
The CONCAT function concatenates all the stings.

Example: CONCAT('hadoop','-','hive') returns 'hadoop-hive'

**CONCAT\_WS( string delimiter, string str1, string str2... )**  
  
The CONCAT\_WS function is similar to the CONCAT function. Here you can also provide the delimiter, which can be used in between the strings to concat.

Example: CONCAT\_WS('-','hadoop','hive') returns 'hadoop-hive'

**FIND\_IN\_SET( string search\_string, string source\_string\_list )**  
  
The FIND\_IN\_SET function searches for the search string in the source\_string\_list and returns the position of the first occurrence in the source string list. Here the source string list should be comma delimited one. It returns 0 if the first argument contains comma.

Example: FIND\_IN\_SET('ha','hao,mn,hc,ha,hef') returns 4

**LENGTH( string str )**  
  
The LENGTH function returns the number of characters in a string.

Example: LENGTH('hive') returns 4

**LOWER( string str ),  LCASE( string str )**  
  
The LOWER or LCASE function converts the string into lower case letters.

Example: LOWER('HiVe') returns 'hive'

**LPAD( string str, int len, string pad )**  
  
The LPAD function returns the string with a length of len characters left-padded with pad.

Example: LPAD('hive',6,'v') returns 'vvhive'

**LTRIM( string str )**  
  
The LTRIM function removes all the trailing spaces from the string.

Example: LTRIM('   hive') returns 'hive'

**REPEAT( string str, int n )**   
  
The REPEAT function repeats the specified string n times.

Example: REPEAT('hive',2) returns 'hivehive'

**RPAD( string str, int len, string pad )**  
  
The RPAD function returns the string with a length of len characters right-padded with pad.

Example: RPAD('hive',6,'v') returns 'hivevv'

**REVERSE( string str )**  
  
The REVERSE function gives the reversed string

Example: REVERSE('hive') returns 'evih'

**RTRIM( string str )**  
  
The RTRIM function removes all the leading spaces from the string.

Example: LTRIM('hive   ') returns 'hive'

**SPACE( int number\_of\_spaces )**  
  
The SPACE function returns the specified number of spaces.

Example: SPACE(4) returns '    '

**SPLIT( string str, string pat )**   
  
The SPLIT function splits the string around the pattern pat and returns an array of strings. You can specify regular expressions as patterns.

Example: SPLIT('hive:hadoop',':') returns ["hive","hadoop"]

**SUBSTR( string source\_str, int start\_position [,int length]  ),  SUBSTRING( string source\_str, int start\_position [,int length]  )**   
  
The SUBSTR or SUBSTRING function returns a part of the source string from the start position with the specified length of characters. If the length is not given, then it returns from the start position to the end of the string.

Example1: SUBSTR('hadoop',4) returns 'oop'

Example2: SUBSTR('hadoop',4,2) returns 'oo'

**TRIM( string str )**  
  
The TRIM function removes both the trailing and leading spaces from the string.

Example: TRIM('   hive   ') returns 'hive'

**UPPER( string str ), UCASE( string str )**  
  
The UPPER or UCASE function converts the string into upper case letters.

Example: UPPER('HiVe') returns 'HIVE'

**Numeric and Mathematical Functions in Hive**

The Numerical functions are listed below in alphabetical order. Use these functions in SQL queries.   
**ABS**( double n )  
  
The ABS function returns the absolute value of a number.

Example: ABS(-100)

**ACOS**( double n )  
  
The ACOS function returns the arc cosine of value n. This function returns Null if the value n is not in the range of -1<=n<=1.

Example: ACOS(0.5)

**ASIN**( double n )  
  
The ASIN function returns the arc sin of value n. This function returns Null if the value n is not in the range of -1<=n<=1.

Example: ASIN(0.5)

**BIN**( bigint n )  
  
The BIN function returns the number n in the binary format.

Example: BIN(100)

**CEIL**( double n ), **CEILING**( double n )  
  
The CEILING or CEILING function returns the smallest integer greater than or equal to the decimal value n.

Example: CEIL(9.5)

**CONV**( bigint n, int from\_base, int to\_base )  
  
The CONV function converts the given number n from one base to another base.

EXAMPLE: CONV(100, 10,2)

**COS**( double n )   
  
The COS function returns the cosine of the value n. Here n should be specified in radians.

Example: COS(180\*3.1415926/180)

**EXP**( double n )  
  
The EXP function returns e to the power of n. Where e is the base of natural logarithm and its value is 2.718.

Example: EXP(50)

**FLOOR**( double n )  
  
The FLOOR function returns the largest integer less than or equal to the given value n.

Example: FLOOR(10.9)

**HEX**( bigint n)  
  
This function converts the value n into hexadecimal format.

Example: HEX(16)

**HEX**( string n )  
  
This function converts each character into hex representation format.

Example: HEX(‘ABC’)

**LN**( double n )  
  
The LN function returns the natural log of a number.

Example: LN(123.45)

**LOG**( double base, double n )  
  
The LOG function returns the base logarithm of the number n.

Example: LOG(3, 66)

**LOG2**( double n )  
  
The LOG2 function returns the base-2 logarithm of the number n.

Example: LOG2(44)

**LOG10**( double n )  
  
The LOG10 function returns the base-10 logarithm of the number n.

Example: LOG10(100)

**NEGATIVE**( int n ),  **NEGATIVE**( double n )   
  
The NEGATIVE function returns –n

Example: NEGATIVE(10)

**PMOD**( int m, int n ), **PMOD**( double m, double n )   
  
The PMOD function returns the positive modulus of a number.

Example: PMOD(3,2)

**POSITIVE**( int n ), **POSITIVE**( double n )  
  
The POSITIVE function returns n

Example: POSITIVE(-10)

**POW**( double m, double n ), **POWER**( double m, double n )  
  
The POW or POWER function returns m value raised to the n power.

Example: POW(10,2)

**RAND**( [int seed] )  
  
The RAND function returns a random number. If you specify the seed value, the generated random number will become deterministic.

Example: RAND( )

**ROUND**( double value [, int n] )  
  
The ROUND function returns the value rounded to n integer places.

Example: ROUND(123.456,2)

**SIN**( double n )   
  
The SIN function returns the sin of a number. Here n should be specified in radians.

Example: SIN(2)

**SQRT**( double n )  
  
The SQRT function returns the square root of the number

Example: SQRT(4)

**UNHEX**( string n )  
  
The UNHEX function is the inverse of HEX function. It converts the specified string to the number format.

Example: UNHEX(‘AB’)

**Date Functions in Hive**

Date data types do not exist in Hive. In fact the dates are treated as strings in Hive. The date functions are listed below.  
  
**UNIX\_TIMESTAMP()**  
  
This function returns the number of seconds from the Unix epoch (1970-01-01 00:00:00 UTC) using the default time zone.  
  
**UNIX\_TIMESTAMP( string date )**  
  
This function converts the date in format 'yyyy-MM-dd HH:mm:ss' into Unix timestamp. This will return the number of seconds between the specified date and the Unix epoch. If it fails, then it returns 0.

Example: UNIX\_TIMESTAMP('2000-01-01 00:00:00') returns 946713600

**UNIX\_TIMESTAMP( string date, string pattern )**  
  
This function converts the date to the specified date format and returns the number of seconds between the specified date and Unix epoch. If it fails, then it returns 0.

Example: UNIX\_TIMESTAMP('2000-01-01 10:20:30','yyyy-MM-dd') returns 946713600

**FROM\_UNIXTIME( bigint number\_of\_seconds  [, string format] )**  
  
The FROM\_UNIX function converts the specified number of seconds from Unix epoch and returns the date in the format 'yyyy-MM-dd HH:mm:ss'.

Example: FROM\_UNIXTIME( UNIX\_TIMESTAMP() ) returns the current date including the time. This is equivalent to the SYSDATE in oracle.

**TO\_DATE( string timestamp )**  
  
The TO\_DATE function returns the date part of the timestamp in the format 'yyyy-MM-dd'.

Example: TO\_DATE('2000-01-01 10:20:30') returns '2000-01-01'

**YEAR( string date )**  
  
The YEAR function returns the year part of the date.

Example: YEAR('2000-01-01 10:20:30') returns 2000

**MONTH( string date )**   
  
The MONTH function returns the month part of the date.

Example: YEAR('2000-03-01 10:20:30') returns 3

**DAY( string date ), DAYOFMONTH( date )**  
  
The DAY or DAYOFMONTH function returns the day part of the date.

Example: DAY('2000-03-01 10:20:30') returns 1

**HOUR( string date )**  
  
The HOUR function returns the hour part of the date.

Example: HOUR('2000-03-01 10:20:30') returns 10

**MINUTE( string date )**  
  
The MINUTE function returns the minute part of the timestamp.

Example: MINUTE('2000-03-01 10:20:30') returns 20

**SECOND( string date )**  
  
The SECOND function returns the second part of the timestamp.

Example: SECOND('2000-03-01 10:20:30') returns 30

**WEEKOFYEAR( string date )**  
  
The WEEKOFYEAR function returns the week number of the date.

Example: WEEKOFYEAR('2000-03-01 10:20:30') returns 9

**DATEDIFF( string date1, string date2 )**  
  
The DATEDIFF function returns the number of days between the two given dates.

Example: DATEDIFF('2000-03-01', '2000-01-10')  returns 51

**DATE\_ADD( string date, int days )**   
  
The DATE\_ADD function adds the number of days to the specified date

Example: DATE\_ADD('2000-03-01', 5) returns '2000-03-06'

**DATE\_SUB( string date, int days )**  
  
The DATE\_SUB function subtracts the number of days to the specified date

Example: DATE\_SUB('2000-03-01', 5) returns ‘2000-02-25’

**Collection Functions**: These functions are used to find the size of the complex types like array and map. The only collection function is SIZE. The SIZE function is used to find the number of elements in an array and map. The syntax of SIZE function is

SIZE( Array<A> ) and SIZE( MAP<key,value> )

**Type Conversion Function**: This function is used to convert from one data type to another. The only type conversion function is CAST. The syntax of CAST is

CAST( expr as <type> )

The CAST function converts the expr into the specified type.  
  
**Table Generating Functions**: These functions transform a single row into multiple rows. EXPLODE is the only table generated function. This function takes array as an input and outputs the elements of array into separate rows. The syntax of EXPLODE is

EXPLODE( ARRAY<A> )

When you use the table generating functions in the SELECT clause, you cannot specify any other columns in the SELECT clause.

**Conditional Functions in Hive**

Hive supports three types of conditional functions. These functions are listed below:  
  
**IF( Test Condition, True Value, False Value )**  
  
The IF condition evaluates the “Test Condition” and if the “Test Condition” is true, then it returns the “True Value”. Otherwise, it returns the False Value.

Example: IF(1=1, 'working', 'not working') returns 'working'

**COALESCE( value1,value2,... )**  
  
The COALESCE function returns the fist not NULL value from the list of values. If all the values in the list are NULL, then it returns NULL.

Example: COALESCE(NULL,NULL,5,NULL,4) returns 5

**CASE Statement**  
  
The syntax for the case statement is:

CASE [ expression ]

WHEN condition1 THEN result1

WHEN condition2 THEN result2

...

WHEN conditionn THEN resultn

ELSE result

END

Here expression is optional. It is the value that you are comparing to the list of conditions. (ie: condition1, condition2, ... conditionn).  
  
All the conditions must be of same datatype. Conditions are evaluated in the order listed. Once a condition is found to be true, the case statement will return the result and not evaluate the conditions any further.  
  
All the results must be of same datatype. This is the value returned once a condition is found to be true.  
  
IF no condition is found to be true, then the case statement will return the value in the ELSE clause. If the ELSE clause is omitted and no condition is found to be true, then the case statement will return NULL  
  
Example: 

CASE Fruit

WHEN 'APPLE' THEN 'The owner is APPLE'

WHEN 'ORANGE' THEN 'The owner is ORANGE'

ELSE 'It is another Fruit'

END

The other form of CASE is

CASE

WHEN Fruit = 'APPLE' THEN 'The owner is APPLE'

WHEN Fruit = 'ORANGE' THEN 'The owner is ORANGE'

ELSE 'It is another Fruit'

END

**Data Types in Hive**

Hive data types are categorized into two types. They are the primitive and complex data types.   
The primitive data types include Integers, Boolean, Floating point numbers and strings. The below table lists the size of each data type:

Type Size

----------------------

TINYINT 1 byte

SMALLINT 2 byte

INT 4 byte

BIGINT 8 byte

FLOAT 4 byte (single precision floating point numbers)

DOUBLE 8 byte (double precision floating point numbers)

BOOLEAN TRUE/FALSE value

STRING Max size is 2GB.

The complex data types include Arrays, Maps and Structs. These data types are built on using the primitive data types.  
  
**Arrays**: Contain a list of elements of the same data type. These elements are accessed by using an index. For example an array, “fruits”, containing a list of elements [‘apple’, ’mango’, ‘orange’], the element “apple” in the array can be accessed by specifying fruits[1].  
  
**Maps**: Contains key, value pairs. The elements are accessed by using the keys. For example a map, “pass\_list” containing the “user name” as key and “password” as value, the password of the user can be accessed by specifying pass\_list[‘username’]  
  
**Structs**: Contains elements of different data types. The elements can be accessed by using the dot notation. For example in a struct, ”car”, the color of the car can be retrieved as specifying car.color  
  
The create table statement containing the complex type is shown below.

CREATE TABLE complex\_data\_types

(

Fruits ARRAY<string>,

Pass\_list MAP<string,string>,

Car STRUCT<color:string, wheel\_size:float>

);