# Hive Function Cheat heet

(http://qu ole2.wpengine.com/wp-content/uploads/2014/01/hive-function-cheat-sheet.pdf)


## Hive Function Meta commands

- HOW FUNCTION – lists Hive functions and operators
- D CRI FUNCTION [function name]– displa s short description of the function
- D CRI FUNCTION XT ND D [function name]– access extended description of the function

## Types of Hive Functions

Hive Function Cheat heet

- Date Functions
- Mathematical Functions
- tring Functions
- Collection Functions

- UDF– is a function that takes one or more columns from a row as argument and returns a single value or o ject. g: concat(col1, col2)
- UDTF— takes zero or more inputs and and produces multiple columns or rows of output. g: explode()
- Macros— a function that uses other Hive functions.

## How To Develop UDFs

package org.apache.hadoop.hive.contri .udf.example;

```
import java.util.Date;
import java.text.SimpleDateFormat;
import org.apache.hadoop.hive.ql.exec.UDF;

@Description(name = "YourUDFName",
    value = "_FUNC_(InputDataType) - using the input datat
ype X argument, "+
            "returns YYY.",
    extended = "Example:\n"
             + "  > SELECT _FUNC_(InputDataType) FROM tabl
ename;")

public class YourUDFName extends UDF{
..
  public YourUDFName( InputDataType InputValue ){
    ..;
  }

  public String evaluate( InputDataType InputValue ){
    ..;
  }
}
```

## How To Develop UDFs, GenericUDFs, UDAFs, and UDTFs

- pu lic class YourUDFName extends UDF{
  - UDAF
  - UDTF
  - Conditional Functions
  - Functions for Text Anal tics

Go to

Pig Function Cheat heet

(http://www.qu ole.com/resources/cheatsheet/pig-function-cheat-sheet/)

- pu lic class YourGenericUDFName extends GenericUDF {..}
- pu lic class YourGenericUDAFName extends A stractGenericUDAFResolver {..}
- pu lic class YourGenericUDTFName extends GenericUDTF {..}

### How To Deplo  / Drop UDFs

At start of each session:

```
ADD JAR /full_path_to_jar/YourUDFName.jar;
    CREATE TEMPORARY FUNCTION YourUDFName AS
'org.apache.hadoop.hive.contrib.udf.example. YourUDFName';
```

At the end of each session:

```
DROP TEMPORARY FUNCTION IF EXISTS YourUDFName;
```

# Date Functions

The following  uilt-in date functions are supported in hive:

| Return T pe | Name(  ignature) | xample |
|---|---|---|
| string | from_unixtime(  igint unixtime[, string format]) | Converts the num er of seconds from unix epoch (1970-01-01 00:00:00 UTC) to a string representing the timestamp of that moment in the current s stem time zone in the format of “1970-01-01 00:00:00” |
| igint | unix_timestamp() | Gets current time stamp using the default time zone. |
| igint | unix_timestamp(string date) | Converts time string in format-MM-dd HH:mm:ss to Unix time stamp, return 0 if fail: unix_timestamp(‘2009-03-20 11:30:01’) = 1237573801 |

| | | |
|---|---|---|
| igint | unix_timestamp(string date, string pattern) | Convert time string with given pattern to Unix time stamp, return 0 if fail: unix_timestamp('2009-03-20', '-MM-dd') = 1237532400 |
| string | to_date(string timestamp) | Returns the date part of a timestamp string: to_date("1970-01-01 00:00:00") = "1970-01-01" |
| int | ear(string date) | Returns the ear part of a date or a timestamp string: ear("1970-01-01 00:00:00") = 1970, ear("1970-01-01") = 1970 |
| int | month(string date) | Returns the month part of a date or a timestamp string: month("1970-11-01 00:00:00") = 11, month("1970-11-01") = 11 |
| int | da (string date)<br>da ofmonth(date) | Return the da part of a date or a timestamp string: da ("1970-11-01 00:00:00") = 1, da ("1970-11-01") = 1 |
| int | hour(string date) | Returns the 'hour of the timestamp: hour('2009-07-30 12:58:59 ) = 12, hour('12:58:59') = 12 |
| int | minute(string date) | Returns the minute of the timestamp |
| int | second(string date) | Returns the second of the timestamp |
| int | weekof ear(string date) | Return the week num er of a timestamp string: weekof ear("1970-11-01 00:00:00") = 44, weekof ear("1970-11-01") = 44 |
| int | datediff(string enddate, string startdate) | Return the num er of da s from startdate to enddate: datediff('2009-03-01', '2009-02-27') = 2 |
| | | |

| | | |
|---|---|---|
| string | date_add(string startdate, int da s) | Add a num er of da s to startdate: date_add('2008-12-31', 1) = '2009-01-01' |
| string | date_su (string startdate, int da s) | u tract a num er of da s to startdate: date_su ('2008-12-31', 1) = '2008-12-30' |
| timestamp | from_utc_timestamp(timestamp, string timezone) | Assumes given timestamp ist UTC and converts to given timezone (as of Hive 0.8.0) |
| timestamp | to_utc_timestamp(timestamp, string timezone) | Assumes given timestamp is in given timezone and converts to UTC (as of Hive 0.8.0) |

# Mathematical Functions

The following uilt-in mathematical functions are supported in hive; most return NULL when the argument(s) are NULL:

| Return T pe | Name( ignature) | xample |
|---|---|---|
| IGINT | round(dou le a) | Returns the rounded IGINT value of the dou le |
| DOU L | round(dou le a, int d) | Returns the dou le rounded to d decimal places |
| IGINT | floor(dou le a) | Returns the maximum IGINT value that is equal or less than the dou le |
| IGINT | ceil(dou le a), ceiling(dou le a) | Returns the minimum IGINT value that is equal or greater than the dou le |
| dou le | rand(), rand(int seed) | Returns a random num er (that changes from row to row) that is distri uted uniforml from 0 to 1. pecifi ing the |

| | | |
|---|---|---|
| | | seed will make sure the generated random num er sequence is deterministic. |
| dou le | exp(dou le a) | Returns $e^{a}$ where e is the  ase of the natural logarithm |
| dou le | ln(dou le a) | Returns the natural logarithm of the argument |
| dou le | log10(dou le a) | Returns the  ase-10 logarithm of the argument |
| dou le | log2(dou le a) | Returns the  ase-2 logarithm of the argument |
| dou le | log(dou le  ase, dou le a) | Return the  ase “  ase” logarithm of the argument |
| dou le | pow(dou le a, dou le p), power(dou le a, dou le p) | Return $a^{p}$ |
| dou le | sqrt(dou le a) | Returns the square root of a |
| string | in(  IGINT a) | Returns the num er in  inar  format |
| string | hex(  IGINT a) hex(string a) | If the argument is an int, hex returns the num er as a string in hex format. Otherwise if the num er is a string, it converts each character into its hex representation and returns the resulting string. |
| string | unhex(string a) | Inverse of hex. Interprets each pair of characters as a hexidecimal num er and converts to the character represented  the num er. |
| string | conv(  IGINT num, int from_  ase, int to_  ase), conv(  TRING num, int from_  ase, int to_  ase) | Converts a num er from a given  ase to another |

| | | |
|---|---|---|
| dou le | a s(dou le a) | Returns the a solute value |
| int dou le | pmod(int a, int )<br>pmod(dou le a, dou le ) | Returns the positive value of a mod |
| dou le | sin(dou le a) | Returns the sine of a (a is in radians) |
| dou le | asin(dou le a) | Returns the arc sin of x if -1<=a<=1 or null otherwise |
| dou le | cos(dou le a) | Returns the cosine of a (a is in radians) |
| dou le | acos(dou le a) | Returns the arc cosine of x if -1<=a<=1 or null otherwise |
| tan(dou le a) | tan(dou le a) | Returns the tangent of a (a is in radians) |
| dou le | atan(dou le a) | Returns the arctangent of a |
| dou le | degrees(dou le a) | Converts value of a from radians to degrees |
| dou le | radians(dou le a) | Converts value of a from degrees to radians |
| int dou le | positive(int a),<br>positive(dou le a) | Returns a |
| int dou le | negative(int a),<br>negative(dou le a) | Returns -a |
| float | sign(dou le a) | Returns the sign of a as ‘1.0’ or ‘-1.0’ |
| dou le | e() | Returns the value of e |
| | | |

| | | |
|---|---|---|
| dou le | pi() | Returns the value of pi |

# String Functions

The following are uilt-in tring functions are supported in hive:

| Return T pe | Name( ignature) | xample |
|---|---|---|
| int | ascii(string str) | Returns the numeric value of the first character of str |
| string | concat(string\| inar A, string\| inar …) | Returns the string or tes resulting from concatenating the strings or tes passed in as parameters in order. e.g. concat('foo', ' ar') results in 'foo ar'. Note that this function can take an num er of input strings. |
| arra <struct<string,dou le>> | context_ngrams(arra <arra >, arra , int K, int pf) | Returns the top-k contextual N-grams from a set of tokenized sentences, given a string of "context".<br>ee tatisticsAndDataMining for more information. |
| string | concat_ws(string P, string A, string …) | Like concat() a ove, ut with custom separator P. |
| string | concat_ws(string P, arra ) | Like concat_ws() a ove, ut taking an arra of strings. (as of Hive 0.9.0) |
| int | find_in_set(string str, string | Returns the first occurance of str in strList |

| | | |
|---|---|---|
| | strList) | where strList is a comma-delimited string. Returns null if either argument is null. Returns 0 if the first argument contains an commas. e.g. find_in_set(‘a ’, ‘a c,  ,a ,c,def’) returns 3 |
| string | format_num er(num er x, int d) | Formats the num er X to a format like ‘#,###,###.##’, rounded to D decimal places, and returns the result as a string. If D is 0, the result has no decimal point or fractional part. (as of Hive 0.10.0) |
| string | get_json_o ject(string json_string, string path) | xtract json o ject from a json string  ased on json path specified, and return json string of the extracted json o ject. It will return null if the input json string is invalid.NOT : The json path can onl  have the characters [0-9a-z_], i.e., no upper-case or special characters. Also, the ke s *cannot start with num ers.* This is due to restrictions on Hive column names. |
| oolean | in_file(string str, string filename) | Returns true if the string str appears as an entire line in filename. |
| int | instr(string str, string su str) | Returns the position of the first occurence of su str in str |
| int | length(string A) | Returns the length of the string |
| int | locate(string su str, string | Returns the position of the first occurrence |

| | | |
|---|---|---|
| | str[, int pos]) | of su str in str after position pos |
| string | lower(string A) lcase(string A) | |
| string | lpad(string str, int len, string pad) | Returns str, left-padded with pad to a length of len |
| string | ltrim(string A) | Returns the string resulting from trimming spaces from the eginning(left hand side) of A e.g. ltrim(‘ foo ar ‘) results in ‘foo ar ‘ |
| arra <struct<string,dou le>> | ngrams(arra <arra >, int N, int K, int pf) | Returns the top-k N-grams from a set of tokenized sentences, such as those returned the sentences() UDAF. ee tatisticsAndDataMining for more information. |
| string | parse_url(string url tring, string partTo xtract [, string ke To xtract]) | Returns the specified part from the URL. Valid values for partTo xtract include HO T, PATH, QU RY, R F, PROTOCOL, AUTHORITY, FIL , and U RINFO. e.g. parse_url(‘http://face ook.com/path1/p.php?k1=v1&k2=v2#Ref1’, ‘HO T’) returns ‘face ook.com’. Also a value of a particular ke in QU RY can e extracted providing the ke as the third argument, e.g. parse_url(‘http://face ook.com/path1/p.php?k1=v1&k2=v2#Ref1’, ‘QU RY’, ‘k1’) returns ‘v1’. |
| | | |

| | | |
|---|---|---|
| string | printf( tring format, O j… args) | Returns the input formatted according do printf-st le format strings (as of Hive 0.9.0) |
| string | regexp_extract(string su ject, string pattern, int index) | Returns the string extracted using the pattern. e.g. regexp_extract(‘foothe ar’, ‘foo(.*?)( ar)’, 2) returns ‘ ar.’ Note that some care is necessar in using predefined character classes: using ‘\s’ as the second argument will match the letter s; ‘s’ is necessar to match whitespace, etc. The ‘index’ parameter is the Java regex Matcher group() method index. ee docs/api/java/util/regex/Matcher.html for more information on the ‘index’ or Java regex group() method. |
| string | regexp_replace(string INITIAL_ TRING, string PATT RN, string R PLAC M NT) | Returns the string resulting from replacing all su strings in INITIAL_ TRING that match the java regular expression s ntax defined in PATT RN with instances of R PLAC M NT, e.g. regexp_replace(“foo ar”, “oo\|ar”, “”) returns ‘f .’ Note that some care is necessar in using predefined character classes: using ‘\s’ as the second argument will match the letter s; ‘s’ is necessar to match whitespace, etc. |
| string | repeat(string str, int n) | Repeat str n times |
| string | reverse(string A) | Returns the reversed string |

| | | |
|---|---|---|
| string | rpad(string str, int len, string pad) | Returns str, right-padded with pad to a length of len |
| string | rtrim(string A) | Returns the string resulting from trimming spaces from the end(right hand side) of A e.g. rtrim(‘ foo ar ‘) results in ‘ foo ar’ |
| arra <arra > | sentences(string str, string lang, string locale) | Tokenizes a string of natural language text into words and sentences, where each sentence is roken at the appropriate sentence oundar and returned as an arra of words. The ‘lang’ and ‘locale’ are optional arguments. e.g. sentences(‘Hello there! How are ou?’) returns ( (“Hello”, “there”), (“How”, “are”, “ ou”) ) |
| string | space(int n) | Return a string of n spaces |
| arra | split(string str, string pat) | plit str around pat (pat is a regular expression) |
| map<string,string> | str_to_map(text[, delimiter1, delimiter2]) | plits text into ke -value pairs using two delimiters. Delimiter1 separates text into K-V pairs, and Delimiter2 splits each K-V pair. Default delimiters are ‘,’ for delimiter1 and ‘=’ for delimiter2. |
| string | su str(string| inar A, int start) su string(string| inar A, int start) | Returns the su string or slice of the te arra of A starting from start position till the end of string A e.g. su str(‘foo ar’, 4) results in ‘ ar’ |

| | | |
|---|---|---|
| string | su str(string\| inar A, int start, int len)<br>su string(string\| inar A, int start, int len) | Returns the su string or slice of the te arra of A starting from start position with length len e.g. su str('foo ar', 4, 1) results in ' ' |
| string | translate(string input, string from, string to) | Translates the input string replacing the characters present in the from string with the corresponding characters in the to string. This is similar to the translatefunction in Postgre QL. If an of the parameters to this UDF are NULL, the result is NULL as well (availa le as of Hive 0.10.0) |
| string | trim(string A) | Returns the string resulting from trimming spaces from oth ends of A e.g. trim(' foo ar ') results in 'foo ar' |
| string | upper(string A) ucase(string A) | Returns the string resulting from converting all characters of A to upper case e.g. upper('fOo aR') results in 'FOO AR' |

# Collection Functions

The following uilt-in collection functions are supported in hive:

| Return T pe | Name( ignature) | xample |
|---|---|---|
| int | size(Map) | Returns the num er of elements in the map t pe |

| | | |
|---|---|---|
| int | size(Arra ) | Returns the num er of elements in the arra t pe |
| arra | map_ke s(Map) | Returns an unordered arra containing the ke s of the input map |
| arra | map_values(Map) | Returns an unordered arra containing the values of the input map |
| oolean | arra _contains(Arra , value) | Returns TRU if the arra contains value |
| arra | sort_arra (Arra ) | orts the input arra in ascending order according to the natural ordering of the arra elements and returns it (as of version 0.9.0) |

# Built-in Aggregate Functions (UDAF)

The following are uilt-in aggregate functions are supported in Hive:

| Return T pe | Name( ignature) | xample |
|---|---|---|
| igint | count(*), count(expr), count(DI TINCT expr[, expr_.]) | count(*) – Returns the total num er of retrieved rows, including rows containing NULL values; count(expr) – Returns the num er of rows for which the supplied expression is non-NULL; count(DI TINCT expr[, expr]) – Returns the num er of rows for which the supplied expression(s) are unique and non-NULL. |
| dou le | sum(col), sum(DI TINCT col) | Returns the sum of the elements in the group or the sum of the distinct values of the column in the group |
| dou le | avg(col), avg(DI TINCT col) | Returns the average of the elements in the group or the average of the distinct values of the column in the group |

| | | |
|---|---|---|
| dou le | min(col) | Returns the minimum of the column in the group |
| dou le | max(col) | Returns the maximum value of the column in the group |
| dou le | variance(col), var_pop(col) | Returns the variance of a numeric column in the group |
| dou le | var_samp(col) | Returns the un iased sample variance of a numeric column in the group |
| dou le | stddev_pop(col) | Returns the standard deviation of a numeric column in the group |
| dou le | stddev_samp(col) | Returns the un iased sample standard deviation of a numeric column in the group |
| dou le | covar_pop(col1, col2) | Returns the population covariance of a pair of numeric columns in the group |
| dou le | covar_samp(col1, col2) | Returns the sample covariance of a pair of a numeric columns in the group |
| dou le | corr(col1, col2) | Returns the Pearson coefficient of correlation of a pair of a numeric columns in the group |
| dou le | percentile( IGINT col, p) | Returns the exact $p^{th}$ percentile of a column in the group (does not work with floating point t pes). p must e etween 0 and 1. NOT : A true percentile can onl e computed for integer values. Use P RC NTIL _APPROX if our input is non-integral. |
| arra | percentile( IGINT col, arra (p1 [, p2]…)) | Returns the exact percentiles p1, p2, … of a column in the group (does not work with floating point t pes). pi must e |

| | | |
|---|---|---|
| | | etween 0 and 1. NOT : A true percentile can onl e computed for integer values. Use P RC NTIL _APPROX if our input is non-integral. |
| dou le | percentile_approx(DOU L col, p [, ]) | Returns an approximate p$^{th}$ percentile of a numeric column (including floating point t pes) in the group. The parameter controls approximation accurac at the cost of memor . Higher values ield etter approximations, and the default is 10,000. When the num er of distinct values in col is smaller than , this gives an exact percentile value. |
| arra | percentile_approx(DOU L col, arra (p1 [, p2]…) [, ]) | ame as a ove, ut accepts and returns an arra of percentile values instead of a single one. |
| arra | histogram_numeric(col, ) | Computes a histogram of a numeric column in the group using non-uniforml spaced ins. The output is an arra of size of dou le-valued (x, ) coordinates that represent the in centers and heights |
| arra | collect_set(col) | Returns a set of o jects with duplicate elements eliminated |

# Built-in Ta le-Generating Functions (UDTF)

Normal user-defined functions, such as concat(), take in a single input row and output a single output row. In contrast, ta le-generating functions transform a single input row to multiple output rows.

| Return T pe | Name( ignature) | xample |
|---|---|---|
| inline(ARRAY< TRUCT[, TRUCT]>) | xplodes an arra of structs into a ta le (as of Hive 0.10) | |
| | | |

| | |
|---|---|
| xplode | explode() takes in an arra as an input and outputs the elements of the arra as separate rows. UDTF's can e used in the L CT expression list and as a part of LAT RAL VI W. |

# Conditional Functions

| Return T pe | Name( ignature) | xample |
|---|---|---|
| T | if( oolean testCondition, T valueTrue, T valueFalseOrNull) | Return valueTrue when testCondition is true, returns valueFalseOrNull otherwise |
| T | COAL C (T v1, T v2, …) | Return the first v that is not NULL, or NULL if all v's are NULL |
| T | CA a WH N TH N c [WH N dTH N e]* [ L f] ND | When a = , returns c; when a = d, return e; else return f |
| T | CA WH N a TH N [WH N cTH N d]* [ L e] ND | When a = true, returns ; when c = true, return d; else return e |

# Functions for Text Anal tics

| Return T pe | Name( ignature) | xample |
|---|---|---|
| arra <struct<string,dou le>> | context_ngrams(arra <arra >, arra , int K, int pf) | Returns the top-k contextual N-grams from a set of tokenized sentences, given a string of "context". ee tatisticsAndDataMining for |

| | | |
|---|---|---|
| | | more information.N-grams are su sequences of length N drawn from a longer sequence. The purpose of the ngrams() UDAF is to find the k most frequent n-grams from one or more sequences. It can e used in conjunction with the sentences() UDF to anal ze unstructured natural language text, or the collect() function to anal ze more general string data. |
| arra <struct<string,dou le>> | ngrams(arra <arra >, int N, int K, int pf) | Returns the top-k N-grams from a set of tokenized sentences, such as those returned the sentences() UDAF.<br>ee tatisticsAndDataMining for more information.Contextual n-grams are similar to n-grams, ut allow ou to specif a 'context' string around which n-grams are to e estimated. For example, ou can specif that ou're onl interested in finding the most common two-word phrases in text that follow the context "I love". You could achieve the same result manuall stripping sentences of non-contextual content and then passing them to ngrams(), ut context_ngrams() makes it much easier. |