Hive Function Cheat Sheet

num_buckets)extract(field FROM source)

Hive Function Cheat Sheet

Go to

Pig Function Cheat Sheet

HIVE FUNCTION META COMMANDS

- SHOW FUNCTIONS– lists Hive functions and operators
- DESCRIBE FUNCTION [function name]– displays short description of the function
- DESCRIBE FUNCTION EXTENDED [function name]– access extended description of the function

TYPES OF HIVE FUNCTIONS

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

HOW TO DEVELOP UDFS

package org.apache.hadoop.hive.contrib.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 datatype X argument, "+
         "returns YYY.",
   extended = "Example:\n"
         + "  > SELECT _FUNC_(InputDataType) FROM tablename;")

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

  public String evaluate( InputDataType InputValue ){
```

```
    ..;
  }
}
```

HOW TO DEVELOP UDFS, GENERICUDFS, UDAFS, AND UDTFS

- public class YourUDFName extends UDF{
- public class YourGenericUDFName extends GenericUDF {..}
- public class YourGenericUDAFName extends AbstractGenericUDAFResolver {..}
- public class YourGenericUDTFName extends GenericUDTF {..}

HOW TO DEPLOY / 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 built-in date functions are supported in Hive:

Return Type	Name(Signature)	Example
string	from_unixtime(bigint unixtime[, string format])	Converts the number of seconds from unix epoch (1970-01-01 00:00:00 UTC) to a string representing the timestamp of that moment in the current system time zone in the format of “1970-01-01 00:00:00”
bigint	unix_timestamp()	Gets current time stamp using the default time zone.
bigint	unix_timestamp(string date)	Converts time string in format yyyy-MM-dd HH:mm:ss to Unix time stamp, return 0 if fail: unix_timestamp(‘2009-03-20 11:30:01’) = 1237573801
bigint	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’, ‘yyyy-MM-dd’) = 1237532400
string (pre-2.1.0) date (2.1.0 on)	to_date(string timestamp)	Returns the date part of a timestamp string: to_date(“1970-01-01 00:00:00”) = “1970-01-01”
int	year(string date)	Returns the year part of a date or a timestamp string: year(“1970-01-01 00:00:00”) = 1970, year(“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	day(string date) dayofmonth(date)	Return the day part of a date or a timestamp string: day(“1970-11-01 00:00:00”) = 1, day(“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	weekofyear(string date)	Return the week number of a timestamp string: weekofyear(“1970-11-01 00:00:00”) = 44, weekofyear(“1970-11-01”) = 44
		Return the number of days from startdate to enddate: datediff(‘2009-03-01’, ‘2009-02-

int	datediff(string enddate, string startdate)	27') = 2
Return Type	**Name(Signature)**	**Example**
string (pre-2.1.0) date (2.1.0 on)	date_add(date/timestamp/string startdate, tinyint/smallint/int days)	Add a number of days to startdate: date_add('2008-12-31', 1) = '2009-01-01'
string (pre-2.1.0) date (2.1.0 on)	date_sub(date/timestamp/string startdate, tinyint/smallint/int days)	Subtract a number of days to startdate: date_sub('2008-12-31', 1) = '2008-12-30'
timestamp	from_utc_timestamp({any primitive type}*, string timezone)	Assumes given timestamp is UTC and converts to given timezone (as of Hive 0.8.0)
timestamp	to_utc_timestamp({any primitive type}*, string timezone)	Assumes given timestamp is in given timezone and converts to UTC (as of Hive 0.8.0)
date	current_date	Returns the current date at the start of query evaluation (as of Hive 1.2.0). All calls of current_date within the same query return the same value.
timestamp	current_timestamp	Returns the current timestamp at the start of query evaluation (as of Hive 1.2.0). All calls of current_timestamp within the same query return the same value.
string	add_months(string start_date, int num_months)	Returns the date that is num_months after start_date (as of Hive 1.1.0). start_date is a string, date or timestamp. num_months is an integer. The time part of start_date is ignored. If start_date is the last day of the month or if the resulting month has fewer days than the day component of start_date, then the result is the last day of the resulting month. Otherwise, the result has the same day component as start_date.
string	last_day(string date)	Returns the last day of the month which the date belongs to (as of Hive 1.1.0). date is a string in the format 'yyyy-MM-dd HH:mm:ss' or 'yyyy-MM-dd'. The time part of date is ignore
string	next_day(string start_date, string day_of_week)	Returns the first date which is later than start_date and named as day_of_week (as of Hive 1.2.0). start_date is a string/date/timestamp. day_of_week is 2 letters, 3 letters or full name of the day of the week (e.g. Mo, tue, FRIDAY). The time part of start_date is ignored. Example: next_day('2015-01-14', 'TU') = 2015-01-20.
string	trunc(string date, string format)	Returns date truncated to the unit specified by the format (as of Hive 1.2.0). Supported formats: MONTH/MON/MM, YEAR/YYYY/YY. Example: trunc('2015-03-17', 'MM') = 2015-03-01.
double	months_between(date1, date2)	Returns number of months between dates date1 and date2 (as of Hive 1.2.0). If date1 is later than date2, then the result is positive. If date1 is earlier than date2, then the result is negative. If date1 and date2 are either the same days of the month or both last days of months, then the result is always an integer.Otherwise the UDF calculates the fractional portion of the result based on a 31-day month and considers the difference in time components date1 and date2. date1 and date2 type can be date, timestamp or string in the format 'yyyy-MM-dd' or 'yyyy-MM-dd HH:mm:ss'. The result is rounded to 8 decimal places. Example: months_between('1997-02-28 10:30:00', '1996-10-30') = 3.94959677
string	date_format(date/timestamp/string ts, string fmt)	Converts a date/timestamp/string to a value of string in the format specified by the date format fmt (as of Hive 1.2.0). Supported formats are Java SimpleDateFormat formats – https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html The second argument fmt should be constant. Example: date_format('2015-04-08', 'y') = '2015'. date_format can be used to implement other UDFs, e.g.: dayname(date) is date_format(date, 'EEEE'); dayofyear(date) is date_format(date, 'D')

Mathematical Functions

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

Return Type	Name(Signature)	Example
BIGINT	round(double a)	Returns the rounded BIGINT value of the double
DOUBLE	round(double a, int d)	Returns the double rounded to d decimal places
BIGINT	floor(double a)	Returns the maximum BIGINT value that is equal or less than the double
BIGINT	ceil(double a), ceiling(double a)	Returns the minimum BIGINT value that is equal or greater than the double

Return Type	Name(Signature)	Example
double	rand(), rand(int seed)	Returns a random number (that changes from row to row) that is distributed uniformly from 0 to 1. Specifiying the seed will make sure the generated random number sequence is deterministic.
double	exp(double a), exp(DECIMAL a)	Returns e^a where e is the base of the natural logarithm
double	ln(double a), ln(DECIMAL a)	Returns the natural logarithm of the argument
double	log10(double a), log10(DECIMAL a)	Returns the base-10 logarithm of the argument
double	log2(double a), log2(DECIMAL a)	Returns the base-2 logarithm of the argument
double	log(double base, double a), log(DECIMAL base, DECIMAL a)	Return the base “base” logarithm of the argument
double	pow(double a, double p), power(double a, double p)	Return a^p
double	sqrt(double a), sqrt(DECIMAL a)	Returns the square root of a
string	bin(BIGINT a)	Returns the number in binary format
string	hex(BIGINT a) hex(string a) hex(BINARY a)	If the argument is an int, hex returns the number as a string in hex format. Otherwise if the number 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 number and converts to the character represented by the number.
string	conv(BIGINT num, int from_base, int to_base), conv(STRING num, int from_base, int to_base)	Converts a number from a given base to another
double	abs(double a)	Returns the absolute value
int double	pmod(int a, int b) pmod(double a, double b)	Returns the positive value of a mod b
double	sin(double a), sin(DECIMAL a)	Returns the sine of a (a is in radians)
double	asin(double a), asin(DECIMAL a)	Returns the arc sin of x if -1<=a<=1 or null otherwise
double	cos(double a), cos(DECIMAL a)	Returns the cosine of a (a is in radians)
double	acos(double a), acos(DECIMAL a)	Returns the arc cosine of x if -1<=a<=1 or null otherwise
tan(double a)	tan(double a), tan(DECIMAL a)	Returns the tangent of a (a is in radians)
double	atan(double a), atan(DECIMAL a)	Returns the arctangent of a
double	degrees(double a), degrees(DECIMAL a)	Converts value of a from radians to degrees
double	radians(double a)	Converts value of a from degrees to radians
int double	positive(int a), positive(double a)	Returns a
int double	negative(int a), negative(double a)	Returns -a
float	sign(double a), sign(DECIMAL a)	Returns the sign of a as ‘1.0’ or ‘-1.0’
double	e()	Returns the value of e
double	pi()	Returns the value of pi

String Functions

The following are built-in String functions are supported in hive:

Return Type	Name(Signature)	Example
int	ascii(string str)	Returns the numeric value of the first character of str
string	base64(binary bin)	Converts the argument from binary to a base- 64 string (as of 0.12.0)
string	chr(bigint|double A)	Returns the ASCII character having the binary equivalent to A (as of Hive 1.3.0 and 2.1.0). If A is larger than 256 the result is equivalent to chr(A % 256). Example: select chr(88); returns "X".string
string	concat(string|binary A, string|binary B...)	Returns the string or bytes resulting from concatenating the strings or bytes passed in as parameters in order. e.g. concat('foo', 'bar') results in 'foobar'. Note that this function can take any number of input strings.
array<struct<string,double>>	context_ngrams(array<array>, array, int K, int pf)	Returns the top-k contextual N-grams from a set of tokenized sentences, given a string of "context". See StatisticsAndDataMining for more information.
string	concat_ws(string SEP, string A, string B...)	Like concat() above, but with custom separator SEP.
string	concat_ws(string SEP, array<string>)	Like concat_ws() above, but taking an array of strings. (as of Hive 0.9.0)
string	decode(binary bin, string charset)	Decodes the first argument into a String using the provided character set (one of 'US-ASCII', 'ISO-8859-1', 'UTF-8', 'UTF-16BE', 'UTF-16LE', 'UTF-16'). If either argument is null, the result will also be null. (As of Hive 0.12.0.)
string	elt(N int,str1 string,str2 string,str3 string,...)	Return string at index number. For example elt(2,'hello','world') returns 'world'. Returns NULL if N is less than 1 or greater than the number of arguments. (See https://dev.mysql.com/doc/refman/5.7/en/string-functions.html#function_elt)
binary	encode(string src, string charset)	Encodes the first argument into a BINARY using the provided character set (one of 'US-ASCII', 'ISO-8859-1', 'UTF-8', 'UTF-16BE', 'UTF-16LE', 'UTF-16'). If either argument is null, the result will also be null. (As of Hive 0.12.0.)
int	field(val T,val1 T,val2 T,val3 T, ...)	Returns the index of val in the val1,val2,val3,... list or 0 if not found. For example field('world','say','hello','world') returns 3. All primitive types are supported, arguments are compared using str.equals(x). If val is NULL, the return value is 0. (See https://dev.mysql.com/doc/refman/5.7/en/string-functions.html#function_field)
int	find_in_set(string str, string strList)	Returns the first occurance of str in strList where strList is a comma-delimited string. Returns null if either argument is null. Returns 0 if the first argument contains any commas. e.g. find_in_set('ab', 'abc,b,ab,c,def') returns 3
string	format_number(number x, int d)	Formats the number 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_object(string json_string, string path)	Extract json object from a json string based on json path specified, and return json string of the extracted json object. It will return null if the input json string is invalid.NOTE: The json path can only have the characters [0-9a-z_], i.e., no upper-case or special characters. Also, the keys *cannot start with numbers.* This is due to restrictions on Hive column names.
boolean	in_file(string str, string filename)	Returns true if the string str appears as an entire line in filename.
string	initcap(string A)	Returns string, with the first letter of each word in uppercase, all other letters in lowercase. Words are delimited by whitespace. (As of Hive 1.1.0.)

Return Type	Name(Signature)	Example
int	instr(string str, string substr)	Returns the position of the first occurence of substr in str
int	length(string A)	Returns the length of the string
int	levenshtein(string A, string B)	Returns the Levenshtein distance between two strings (as of Hive 1.2.0). For example, levenshtein('kitten', 'sitting') results in 3.
int	locate(string substr, string str[, int pos])	Returns the position of the first occurrence of substr in str after position pos
string	lower(string A) lcase(string A)	Returns the string resulting from converting all characters of B to lower case. For example, lower('fOoBaR') results in 'foobar'.
string	lpad(string str, int len, string pad)	Returns str, left-padded with pad to a length of len. If str is longer than len, the return value is shortened to len characters. In case of empty pad string, the return value is null.
string	ltrim(string A)	Returns the string resulting from trimming spaces from the beginning(left hand side) of A e.g. ltrim(' foobar ') results in 'foobar '
array<struct<string,double>>	ngrams(array<array >, int N, int K, int pf)	Returns the top-k N-grams from a set of tokenized sentences, such as those returned by the sentences() UDAF. See StatisticsAndDataMining for more information.
string	parse_url(string urlString, string partToExtract [, string keyToExtract])	Returns the specified part from the URL. Valid values for partToExtract include HOST, PATH, QUERY, REF, PROTOCOL, AUTHORITY, FILE, and USERINFO. e.g. parse_url('http://facebook.com/path1/p.php?k1=v1&k2=v2#Ref1', 'HOST') returns 'facebook.com'. Also a value of a particular key in QUERY can be extracted by providing the key as the third argument, e.g. parse_url('http://facebook.com/path1/p.php?k1=v1&k2=v2#Ref1', 'QUERY', 'k1') returns 'v1'.
string	printf(String format, Obj... args)	Returns the input formatted according do printf-style format strings (as of Hive 0.9.0)
string	regexp_extract(string subject, string pattern, int index)	Returns the string extracted using the pattern. e.g. regexp_extract('foothebar', 'foo(.*?)(bar)', 2) returns 'bar.' Note that some care is necessary in using predefined character classes: using '\s' as the second argument will match the letter s; 's' is necessary to match whitespace, etc. The 'index' parameter is the Java regex Matcher group() method index. See docs/api/java/util/regex/Matcher.html for more information on the 'index' or Java regex group() method.
string	regexp_replace(string INITIAL_STRING, string PATTERN, string REPLACEMENT)	Returns the string resulting from replacing all substrings in INITIAL_STRING that match the java regular expression syntax defined in PATTERN with instances of REPLACEMENT, e.g. regexp_replace("foobar", "oo|ar", "") returns 'fb.' Note that some care is necessary in using predefined character classes: using '\s' as the second argument will match the letter s; 's' is necessary to match whitespace, etc.
string	repeat(string str, int n)	Repeat str n times
string	replace(string A, string OLD, string NEW)	Returns the string A with all non-overlapping occurrences of OLD replaced with NEW (as of Hive 1.3.0 and 2.1.0). Example: select replace("ababab", "abab", "Z"); returns "Zab".
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. If str is longer than len, the return value is shortened to len characters. In case of empty pad string, the return value is null.
string	rtrim(string A)	Returns the string resulting from trimming spaces from the end(right hand side) of A e.g. rtrim(' foobar ') results in ' foobar'
array<array>	sentences(string str, string lang, string locale)	Tokenizes a string of natural language text into words and sentences, where each sentence is broken at the appropriate sentence boundary and returned as an array of words. The 'lang' and 'locale' are optional arguments. e.g. sentences('Hello there! How are you?') returns (("Hello", "there"), ("How", "are", "you"))
string	soundex(string A)	Returns soundex code of the string (as of Hive 1.2.0). For example, soundex('Miller') results in M460.

Return Type	Name(Signature)	Example
string	space(int n)	Return a string of n spaces
array	split(string str, string pat)	Split str around pat (pat is a regular expression)
map<string,string>	str_to_map(text[, delimiter1, delimiter2])	Splits text into key-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	substr(string|binary A, int start) substring(string|binary A, int start)	Returns the substring or slice of the byte array of A starting from start position till the end of string A e.g. substr('foobar', 4) results in 'bar'
string	substr(string|binary A, int start, int len) substring(string|binary A, int start, int len)	Returns the substring or slice of the byte array of A starting from start position with length len e.g. substr('foobar', 4, 1) results in 'b'
string	translate(string|char|varchar input, string|char|varchar from, string|char|varchar to)	Translates the input string by replacing the characters present in the from string with the corresponding characters in the to string. This is similar to the translate function in PostgreSQL. If any of the parameters to this UDF are NULL, the result is NULL as well (available as of Hive 0.10.0; char/varchar support added as of Hive 0.14.0.)
string	trim(string A)	Returns the string resulting from trimming spaces from both ends of A e.g. trim(' foobar ') results in 'foobar'
string	unbase64(string str)	Converts the argument from a base 64 string to BINARY. (As of Hive 0.12.0.)
string	upper(string A) ucase(string A)	Returns the string resulting from converting all characters of A to upper case e.g. upper('fOoBaR') results in 'FOOBAR'

Collection Functions

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

Return Type	Name(Signature)	Example
int	size(Map)	Returns the number of elements in the map type
int	size(Array)	Returns the number of elements in the array type
array	map_keys(Map)	Returns an unordered array containing the keys of the input map
array	map_values(Map)	Returns an unordered array containing the values of the input map
boolean	array_contains(Array, value)	Returns TRUE if the array contains value
array	sort_array(Array)	Sorts the input array in ascending order according to the natural ordering of the array elements and returns it (as of version 0.9.0)

Built-in Aggregate Functions (UDAF)

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

Return Type	Name(Signature)	Example
bigint	count(*), count(expr), count(DISTINCT expr[, expr_.])	count(*) – Returns the total number of retrieved rows, including rows containing NULL values; count(expr) – Returns the number of rows for which the supplied expression is non-NULL; count(DISTINCT expr[, expr]) – Returns the number of rows for which the supplied expression(s) are unique and non-NULL.
double	sum(col), sum(DISTINCT col)	Returns the sum of the elements in the group or the sum of the distinct values of the column in the group
double	avg(col), avg(DISTINCT col)	Returns the average of the elements in the group or the average of the distinct values of the column in the group

Return Type	Name(Signature)	Example
double	min(col)	Returns the minimum of the column in the group
double	max(col)	Returns the maximum value of the column in the group
double	variance(col), var_pop(col)	Returns the variance of a numeric column in the group
double	var_samp(col)	Returns the unbiased sample variance of a numeric column in the group
double	stddev_pop(col)	Returns the standard deviation of a numeric column in the group
double	stddev_samp(col)	Returns the unbiased sample standard deviation of a numeric column in the group
double	covar_pop(col1, col2)	Returns the population covariance of a pair of numeric columns in the group
double	covar_samp(col1, col2)	Returns the sample covariance of a pair of a numeric columns in the group
double	corr(col1, col2)	Returns the Pearson coefficient of correlation of a pair of a numeric columns in the group
double	percentile(BIGINT col, p)	Returns the exact p^{th} percentile of a column in the group (does not work with floating point types). p must be between 0 and 1. NOTE: A true percentile can only be computed for integer values. Use PERCENTILE_APPROX if your input is non-integral.
array<double>	percentile(BIGINT col, array(p1 [, p2]...))	Returns the exact percentiles p1, p2, ... of a column in the group (does not work with floating point types). pi must be between 0 and 1. NOTE: A true percentile can only be computed for integer values. Use PERCENTILE_APPROX if your input is non-integral.
double	percentile_approx(DOUBLE col, p [, B])	Returns an approximate p^{th} percentile of a numeric column (including floating point types) in the group. The B parameter controls approximation accuracy at the cost of memory. Higher values yield better approximations, and the default is 10,000. When the number of distinct values in col is smaller than B, this gives an exact percentile value.
array	percentile_approx(DOUBLE col, array(p1 [, p2]...) [, B])	Same as above, but accepts and returns an array of percentile values instead of a single one.
array<struct {'x','y'}>	histogram_numeric(col, b)	Computes a histogram of a numeric column in the group using b non-uniformly spaced bins. The output is an array of size b of double-valued (x,y) coordinates that represent the bin centers and heights
array	collect_set(col)	Returns a set of objects with duplicate elements eliminated
array	collect_list(col)	Returns a list of objects with duplicates. (As of Hive 0.13.0.)
integer	ntile(INTEGER x)	Divides an ordered partition into x groups called buckets and assigns a bucket number to each row in the partition. This allows easy calculation of tertiles, quartiles, deciles, percentiles and other common summary statistics. (As of Hive 0.11.0.)

Built-in Table-Generating Functions (UDTF)

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

Return Type	Name(Signature)	Example
inline(ARRAY<STRUCT[,STRUCT]>)	Explodes an array of structs into a table (as of Hive 0.10)	
explode(ARRAY<T> a)	explode() takes in an array as an input and outputs the elements of the array as separate rows. UDTF's can be used in the SELECT expression list and as a part of LATERAL VIEW.	

Conditional Functions

Return Type	Name(Signature)	Example

Return Type	Name(Signature)	Example
T	if(boolean testCondition, T valueTrue, T valueFalseOrNull)	Return valueTrue when testCondition is true, returns valueFalseOrNull otherwise
T	COALESCE(T v1, T v2, ...)	Return the first v that is not NULL, or NULL if all v's are NULL
T	CASE a WHEN b THEN c [WHEN d THEN e]* [ELSE f] END	When a = b, returns c; when a = d, return e; else return f
T	CASE WHEN a THEN b [WHEN c THEN d]* [ELSE e] END	When a = true, returns b; when c = true, return d; else return e

Functions for Text Analytics

Return Type	Name(Signature)	Example
array<struct<string,double>>	context_ngrams(array<array>, array, int K, int pf)	Returns the top-k contextual N-grams from a set of tokenized sentences, given a string of "context". See StatisticsAndDataMining for more information.N-grams are subsequences 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 be used in conjunction with the sentences() UDF to analyze unstructured natural language text, or the collect() function to analyze more general string data.
array<struct<string,double>>	ngrams(array<array>, int N, int K, int pf)	Returns the top-k N-grams from a set of tokenized sentences, such as those returned by the sentences() UDAF. See StatisticsAndDataMining for more information.Contextual n-grams are similar to n-grams, but allow you to specify a 'context' string around which n-grams are to be estimated. For example, you can specify that you're only interested in finding the most common two-word phrases in text that follow the context "I love". You could achieve the same result by manually stripping sentences of non-contextual content and then passing them to ngrams(), but context_ngrams() makes it much easier.

Cheatsheet

PySpark Cheatsheet

Support

SOCIAL MEDIA

Facebook

Linkedin

Twitter

Email Qubole

(855) 423-6674

Privacy Policy | Terms & Conditions