The Hadoop Pig Syntax Card

Types

int	32-bit signed integer	42
long	64-bit signed integer	42L
float	32-bit signed floating point	3.14F 0.0314e2F
double	64-bit signed floating point	2.718 271.8e-2
chararray	UTF-8 character string	'Hadoop'
bytearray	array of bytes	N/A
tuple	ordered set of fields	(field [, field ...])
bag	unordered collection of tuples	{tuple [, tuple ...]}
map	set of key value pairs	[key#val [, key#val ...]]

Schemas

Simple Types	(alias[:type] [, alias[:type] ...])
Tuple	(alias[:tuple] (alias[:type] [, alias[:type] ...]))
Bag	(alias[:bag] { tuple schema })
Map	(alias[:map] [])

Dereference

Tuple	T.field_name
	T.$0
	T.($1, field_name)
Bag	B.field_name
	B.$0
	B.($1, field_name)
Map	M#'key'

UDFs

```
REGISTER path;
DEFINE alias {function|[`command` [input] [output] [ship] [cache]]};
```

input	INPUT {stdin|'path'} [USING serializer] [, {stdin|'path'} [USING serializer] ...]
output	OUTPUT {stdin|stderr|'path'} [USING serializer] [, {stdin|stderr|'path'} [USING serializer] ...]
ship	SHIP ('path' [, 'path' ...])
cache	CACHE ('dfs_path#dfs_file' [, 'dfs_path#dfs_file' ...])

Eval

AVG
CONCAT
COUNT
COUNT_STAR
DIFF
MAX
MIN
SIZE
SUM
TOKENIZE

Math

ABS
ACOS
ASIN
ATAN
CBRT
CEIL
COS
COSH
EXP
FLOOR
LOG
LOG10
RANDOM
ROUND
SIN
SINH
SQRT
TAN
TANH

String

INDEXOF
LAST_INDEX_OF
LCFIRST
LOWER
REGEX_EXTRACT
REGEX_EXTRACT_ALL
REPLACE
STRSPLIT
SUBSTRING
TRIM
UCFIRST
UPPER

Bag/Tuple

TOBAG
TOP
TOTUPLE

Misc

-x {local,mapreduce}	Choose execution mode
-param param=value	Assign value to param for use in the script
-param _file file	Load parameters from file
$param	Dereference parameter param
%declare param value	Assign value to param in the script
%default param value	Set default value for param

```
fs FSShell_subcommand parameters
sh shell_subcommand parameters
{exec,run} [-param name=value] [-param_file file] script
kill jobid
set key 'value'
help
quit
```


The Hadoop Pig Syntax Card

Input/Output

```
alias = LOAD 'data' [USING function] [AS schema];

STORE alias INTO 'directory' [USING function];

alias = ORDER alias BY { * [ASC|DESC]
                       | field_alias [ASC|DESC]
                         [, field_alias [ASC|DESC] ...]
                       }
                     [PARALLEL n];
```

Diagnostic

```
DESCRIBE alias;

DUMP alias;

EXPLAIN alias;
```

Selection

```
alias_1 = LIMIT alias_0 n;

alias_1 = SAMPLE alias_0 size; (size in [0.0, 1.0])

alias_1 = DISTINCT alias_0;

alias_1 = FILTER alias_0 BY expression;

SPLIT alias0 INTO alias1 IF expression1
                  [, aliasN IF expressionN ...];
```

Joining and Grouping

```
alias = UNION [ONSCHEMA] alias_0, alias_1 [, alias_N ...];

alias = CROSS alias0, alias1 [, aliasN ...] [PARTITION BY part] [PARALLEL n];

alias = JOIN alias0 BY {expression|'('expression [, expression ...]')'}
          [, alias1 BY {expression|'('expression [, expression ...]')'} ...]
          [USING 'replicated' | 'skewed' | 'merge']
          [PARTITION BY partitioner] [PARALLEL n];

alias = JOIN left_alias BY left_alias_column [LEFT|RIGHT|FULL] [OUTER],
           right_alias BY right_alias_column
           [USING 'replicated' | 'skewed' | 'merge']
           [PARTITION BY partitioner] [PARALLEL n];

alias = GROUP alias { ALL | BY expression} [, alias ALL | BY expression ...]
            [USING 'collected' | 'merge']
            [PARTITION BY partitioner] [PARALLEL n];
```

Transformation

```
alias1  = FOREACH alias GENERATE expression [AS schema]
          [, expression [AS schema] ... ];

alias1  = FOREACH alias {
           alias = nested_op; [ alias = nested_op; ... ]
           GENERATE expression [AS schema][, expression [AS schema] ... ]
          };

alias1 = MAPREDUCE 'mr.jar'
          STORE alias2 INTO 'inputLocation' USING storeFunc
          LOAD 'outputLocation' USING loadFunc AS schema [`params, ... `];

alias = STREAM alias [, alias ...]
        THROUGH {`command` | cmd_alias } [AS schema] ;
```