Hadoop Online Tutorial
A learning center for hadoop Eco System

Hive Functions Examples

2

This entry was posted in Hive *on May 28, 2015 by Siva*

Table of Contents [hide]

Popular Pages

Hive Archives - Hadoop Online Tutorials

Partitioning in Hive - Hadoop Online Tutorials

More Info

Core Hadoop

Big Data

Hadoop

Map Reduce

EcoSystem Tools

Hive

Pig

HBase

Impala

Sponsored Content

Watch Silicon Valley Free For 30 Days: Cli Here
Hotstar

Top 5: New Luxury SUVs
Departures International

5 ways to avoid risks in trading
www.5paisa.com

The BMW 3 Series. Celebrate as you beat price hike. Know more.
BMW India

Recommended by

Hive Functions Examples

SET

SHOW

USE

CREATE DATABASE

CREATE MANAGED TABLE

CREATE EXTERNAL TABLE

CREATING TABLE FROM EXISTING TABLE

CREATING EXTERNAL TABLES FROM MANAGED TABLES

LOAD

COPY DATA FROM ONE TABLE TO ANOHTER

DROP

QUIT

SELECT

DESCRIBE

DESCRIBE SPECIFIC FIELD

DESCRIBE EXTENDED

SET

```
hive> set;     --> prints all variables from these 4 namespaces:
```

SHOW

```
hive> show databases;
hive> show tables;
hive> show databases like 'f*';
```

USE

```
hive> use default;
hive (default)> use test;
```

CREATE DATABASE

```
hive> create database if not exists financials;
hive> create database financials
> comment 'holds all financial tables'
> location '/tmp/hive_db_dir';
```

CREATE MANAGED TABLE

```
hive> create table records (year string, temperature int, quantity int)
> row format delimited
> fields terminated by '\t';
hive> create table employees (
> name string,
> salary float,
> subordinates array<string>,
> deductions map<string, float>,
> address struct<street:string, city:string, state:string, zip:int>);
hive> create database financials2
> with dbproperties('creator' = 'Ganesh Pillai', 'date' = '2015-02-19');
```

CREATE EXTERNAL TABLE:

```
[cloudera@localhost ~]$ cat scripts/create_external_table.hql;
create external table if not exists mydb.stocks(
exchange string,
symbol string,
ymd string,
price_open float,
price_high float,
price_los float,
price_close float,
volume int,
price_adj_close float)
row format delimited fields terminated by ',' location '/user/cloudera/myhive';
```

CREATING TABLE FROM EXISTING TABLE

```
hive (mydb)> create external table stocks2
> like stocks;
```

CREATING EXTERNAL TABLES FROM MANAGED TABLES:

```
hive> create external table mydb.employees3
> like employees;
```

LOAD

```
hive> load data local inpath 'data/sample.txt'
> overwrite into table records;
hive (mydb)> load data local inpath 'data/NYSE_daily'
> overwrite into table stocks;
```

COPY DATA FROM ONE TABLE TO ANOHTER

```
Hive> insert overwrite table nyse_daily_bak select * from nyse_daily;
NOTE: Launges Map-only job
hive> insert overwrite dividends_bak select * from dividends;
```

DROP

```
hive> drop table records;
hive (financials2)> drop database if exists financials2 cascade;
--NOTE: cascade/restrict is optional
```

QUIT

```
hive> quit;
```

SELECT

```
hive> select year, MAX(temperature) from records
> group by year;
```

DESCRIBE

```
hive> describe database financials;
hive> describe database default;
hive (mydb)> describe database mydb;
hive (mydb)> describe employees;
hive (mydb)> describe extended employees;
hive (mydb)> describe formatted employees;
```

DESCRIBE SPECIFIC FIELD

```
hive (mydb)> describe employees.address;
```

DESCRIBE EXTENDED

```
hive> describe database extended financials2;
OK
financials2                    hdfs://localhost.localdomain:8020/user/hive/warehous
Time taken: 0.124 seconds
```

ALTER

```
hive (financials2)> alter database financials set dbproperties('modified by' = 'GANESH PILL
```

CLONE SCHEMA (DATA IS NOT COPIED)

```
hive (mydb)> create table mydb.employees2
> like employees;
```

CLONE SCHEMA TO ANOTHER DB

```
hive (mydb)> create table financials.employees3 like employees;
hive (mydb)> load data local inpath 'data/NYSE_daily' into table stocks2;
hive> drop database mydb;
FAILED: InvalidOperationException(message:Database mydb is not empty)
hive> drop database mydb cascade;
[cloudera@localhost ~]$ cat scripts/stocks.hql;
create external table if not exists mydb.stocks(exchange string,
symbol string,
ymd string,
price_open float,
price_high float,
price_los float,
price_close float,
volume int,
price_adj_close float)
row format delimited fields terminated by '\t' location '/user/cloudera/myhive';
[cloudera@localhost ~]$ cat scripts/load_stocks.hql;
load data local inpath 'data/NYSE_daily'
overwrite into table mydb.stocks;
```

–EMPLOYEES TABLE:

```
[cloudera@localhost ~]$ cat scripts/employees_no_partition.hql;
create table mydb.employees_no_partition(name string,
salary float,
subordinates array<string>,
deductions map<string, float>,
address struct<street:string, city:string, state:string, zip:int>)
row format delimited fields terminated by '\001'
collection items terminated by '\002'
map keys terminated by '\003'
lines terminated by '\n' stored as textfile;

[cloudera@localhost ~]$ cat scripts/load_employees_no_partition.hql;
load data local inpath 'data/employees.txt'
overwrite  into table mydb.employees_no_partition;

hive (mydb)> select name, subordinates[0] from employees_no_partition;
hive (mydb)> select name, address from employees_no_partition;
name                                      address
John Doe                                {"street":"1 Michigan Ave.","city":"Chicago","state'

hive (mydb)> select name, deductions["State Taxes"] from employees_no_partition;
name                                      _c1
John Doe                                0.05

hive (mydb)> select name, address.city from employees_no_partition;
name                                      city
John Doe                                Chicago
```

USING REGULAR EXPRESSIONS:

```
hive (mydb)> select symbol, `price.*` from stocks limit 5;
symbol   price_open          price_high          price_los              price_close
CLI         35.39                    35.7                   34.5
```

COMPUTING WITH COLUMN VALUES(using functions & arithmetic expressions)

```
hive (mydb)> select upper(name),salary,deductions["Federal Taxes"],round(salary*(1-deductic
_c0                                          salary                      _c2          _c3
JOHN DOE                              100000.0                             0.2          800C
hive (mydb)> select count(*) as count, avg(salary) as salary from employees_no_partition;
count     salary
7            102857.14285714286
```

ARITMETIC OPERATORS:

+, -, *, /, %, &(AND), |(OR), ^(XOR),~(NOT)

MATHEMATICAL FUNCTIONS:

round, rand, abs, etc

```
The Numerical functions are listed below in alphabetical order. Use these functions in SQL
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 valu

Example: ACOS(0.5)

ASIN( double n )

The ASIN function returns the arc sin of value n. This function returns Null if the value r

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 c

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 )
```

AGGREGATE FUNCTIONS:

count, sum, ave, min, max, etc

```
hive (mydb)> select count(*), avg(salary) from employees_no_partition;
hive (mydb)> set hive.map.aggr = true;
hive (mydb)> select MAX(column_name) from tablename;
hive (mydb)> select MIN(column_name) from tablename;
```

NOTE: this setting will attempt to do top-level aggregation in the map phase

NOTE: An aggregation that is NOT top-level is aggregation after GROUP BY

```
hive (mydb)> select count(distinct symbol) from stocks;
237
```

NOTE: The answer will be ZERO if symbol is a partitioned column

```
hive (mydb)> select count(distinct ymd), count(distinct volume) from stocks;
_c0         _c1
```

```
361         24311
```

TABLE GENERATING FUNCTIONS(inverse of aggregate functions):

explode, json_tuple, parse_url_tuple, stack

NOTE: aggregate : many rows –> one result

table generating funtions: one column –> many rows/columns

```
hive (mydb)> select explode(subordinates) as sub from employees_no_partition;
sub
Mary Smith
```

OTHER BUILT-IN FUNCTIONS

```
test in(v1, v2, v3, ...) --> returns true if a match is found
length(s)
reverse(s)
concat(s1,s2,s3,...)
concat(separator,s1,s2,s3,...)
substr(s, start, length)
upper(s)/ucase(s),
trim(s), ltrim(s)
regexp_replace(s, regex, replacement) --> regexp_replace(hive, [iv], x) --> hxvx
size(map<k.v>)
siz(array<T>)
cast(<expr> as <type>) --> cast('1' as int)
year, month, day  from timestamp
space(n)
repeat(s,n)
split(s, pattern)
instr(str, substr)
in_file(s, filename)
```

LIMIT

```
hive (mydb)> select explode(subordinates) from employees_no_partition limit 1;
```

NESTED SELECT STATEMENT

```
hive> from (select name, salary from employees_no_partition limit 10) e select e.name, e.sa
```

CASE..WHEN..THEN

```
hive> select name, salary,
> case
> when salary < 50000 then 'low'
> when salary >=50000 and salary <70000 then 'middle'
> when salary >=70000 then 'high'
> else 'very high'
> end
> as salary_bracket
> from employees_no_partition;
```

HIVE AVOIDS MAP REDUCE:

```
hive (mydb)> select * from employees_no_partition;
```

NOTE: Hive avoids mapreduce if WHERE clause filters on partition keys

FORCING HIVE TO AVOID MAP REDUCE:

```
hive (mydb)> set hive.exec.mode.local.auto = true;
```

NOTE: No gurantee that Hive won’t invoke map reduce

PREDICATE OPERATORS:

```
= (A = B), !=, <. <=, >, >=, IS NULL, IS NOT NULL, LIKE, RLIKE/REGEXP(A RLIKE B --> A= regu
```

NOTE: Use extreme caution when comparing floating point numbers;

Avoid all implicit casts from smaller to wider types;

Avoid floating point numbers for anything involving money

LIKE & RLIKE

```
hive (mydb)> select name from employees_no_partition where address.street like '%Main%';
hive (mydb)> select name from employees_no_partition where address.street rlike '.*(Chicago
```

NOTE: period –> any character

* –> the character apprearing immediately left of * can repeat 0 or more times

| –> or

GROUP BY

```
hive (mydb)> select max(price_open) from stocks;
hive (mydb)> select symbol,max(price_open) from stocks;
FAILED: SemanticException [Error 10025]: Line 1:7 Expression not in GROUP BY key 'symbol'
hive (mydb)> select symbol,max(price_open) from stocks GROUP BY symbol limit 5;
hive (mydb)> select symbol,max(price_open) as max from stocks GROUP BY symbol having max >
symbol   max
CACI       49.79
CAF         39.5
CAH        66.9
CAJ         53.22
CAM       42.47
```

JOINS

Hive supports:

- INNER JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN
- LEFT SEMI JOIN(returns records from the left table if records are found
- in right table that satisfy ON predicates)
- CARTISIAN PRODUCT JOIN (select * from T1 join T2)
- MAP SIDE JOIN(small table cached in memory)(small table on right side of join)

–>Hive doesn't support OR between predicates in ON clause (AND is supported)

–>You can join more than 2 tables

ORDER BY, SORT BY, DISTRIBUTED BY WITH SORT BY, CLUSTER BY

- ORDER BY –> total ordering
- SORT BY –> orders data within each reducer(local ordering)
- if hive.mapred.mode=strict –> LIMIT needed with ORDER BY
- DISTRIBUTED BY controls how map output is divided among reducers
- CLUSTER BY = DISTRIBUTE BY + SORT BY

–NOTE: Comparison:

ITEM ORDERING # REDUCERS OUTPUT

ORDER BY Global 1 reducer(unacceptable for large data sets) 1 sorted file

SORY BY Local to reducer N reducers(with overlapping data) >=N sorted files

DISTRIBUTE BY No sorting N reducers(non-overlapping data)(not sorted) >=N sorted files

CLUSTER BY [result:global] N reducers(non-overlapping data)(sorted) >=N sorted files

```
hive (mydb)> select s.ymd, s.symbol, s.price_close from stocks s order by s.ymd asc, s.symb
hive (mydb)> select s.ymd, s.symbol, s.price_close from stocks s sort by s.ymd asc, s.symbo
hive (mydb)> select s.ymd, s.symbol, s.price_close from stocks s distribute by s.symbol sor
hive (mydb)> select s.ymd, s.symbol, s.price_close from stocks s cluster by s.symbol;
```

Example:

```
SORT BY: (each reducer output is ordered; but total order is missing)
r1:           0
0
3
9
r2:           0
0
1
2
ORDER BY: (single output; fully ordered)
r:             0
0
0
0
1
2
3
9
DISTRIBUTE BY: (same keys go into same reducer; no guarantee to be clusteed in adjacent pos
r1:           x1
x2
x1
r2:           x4
x3
CLUSTER BY: (global ordering across multiple reducers)
r1:           x1
```

```
x1
x2
r2:          x3
x4
```

CAST

```
hive (mydb)> select name from employees_no_partition  where cast(salary as int) > 100000;
hive (mydb)> select name from employees_no_partition  where cast(salary as float) > 100000
hive (mydb)> create external table numbers(number int);
hive (mydb)> load data local  inpath 'data/numbers.txt' into numbers;
hive (mydb)> describe numbers;
OK
col_name              data_type             comment
number int
hive (mydb)> select * from numbers;
number
1
2
3
4
5
6
7
8
10
```

TABLESAMPLE-BUCKET

- tablesample (bucket x out of y [on column_name]);
- tablesample (bucket x out of y on rand()) table_alias;
- Rows of the table are bucketed on the column name randomly into y buckets numbered 1 to y
- Rows belonging to bucket x are returned

```
hive (mydb)> select * from numbers TABLESAMPLE(BUCKET 2 OUT OF 5 on rand()) s;
number
7
8
10
```

```
hive (mydb)> select * from numbers TABLESAMPLE(BUCKET 2 OUT OF 5 on rand()) s;
OK
number
hive (mydb)> select * from numbers TABLESAMPLE(BUCKET 2 OUT OF 5 on rand()) s;
OK
number
5
hive> select * from numbers TABLESAMPLE(BUCKET 1 OUT OF 5 on number) s;
OK
1
6
hive> select * from numbers TABLESAMPLE(BUCKET 1 OUT OF 5 on number) s;
OK
1
6
hive> select * from numbers TABLESAMPLE(BUCKET 1 OUT OF 5 on number) s;
OK
1
6
hive> select * from numbers TABLESAMPLE(BUCKET 1 OUT OF 5 on number) s;
OK
5
10
hive> select * from numbers TABLESAMPLE(BUCKET 1 OUT OF 5 on number) s;
OK
5
10
```

BLOCK SAMPLING

–> tablesample(n percent) –> n % of the data size (not n% of rows)

```
hive> select * from numbers tablesample(0.1 percent) s;
OK
1
2
3
4
5
6
7
8
10
Time taken: 11.612 seconds
```

–NOTE: Smallest unit of sampling is a single HDFS block; if table size > block size, all rows are returned;

–> In percentage sampling, by using 'set hive.sample.seednumner=<integer>' you can control the subsets of data sampled

–INPUT PRUNING FOR BUCKET TABLES:

–> Typically, tablesample will scan the entire table; this is not efficient; but if the columns specified in TABLESAMPLE clause match columns in the CLUSTERED BY clause, tablesample queries only scan the required hash paritions of the

```
hive> create table numbers_bucketed(number int) clustered by (number) into 3 buckets;
OK
Time taken: 0.264 seconds

hive> set hive.enforce.bucketing=true;

hive> insert table numbers_bucketed select number from numbers;
FAILED: ParseException line 1:0 cannot recognize input near 'insert' 'table' 'numbers_buck
hive> insert overwrite table numbers_bucketed select number from numbers;
Total MapReduce CPU Time Spent: 20 seconds 600 msec
OK
Time taken: 61.175 seconds

hive> select * from numbers_bucketed;
OK
3
6
1
4
7
10
2
5
8
Time taken: 0.711 second
```

```
[cloudera@localhost ~]$ hadoop fs -cat /user/hive/warehouse/mydb.db/numbers_bucketed/00000

3
6

[cloudera@localhost ~]$ hadoop fs -cat /user/hive/warehouse/mydb.db/numbers_bucketed/00000

1
4
7
10

[cloudera@localhost ~]$ hadoop fs -cat /user/hive/warehouse/mydb.db/numbers_bucketed/00000

2
5
8

hive> select * from numbers_bucketed tablesample(bucket 1 out of 3 on number) s;

OK
3
6

hive> select * from numbers_bucketed tablesample(bucket 2 out of 3 on number) s;

OK
1
4
7
10

hive> select * from numbers_bucketed tablesample(bucket 3 out of 3 on number) s;

OK
2
5
8
```

-UNION ALL (combines 2 or more tables)

–> number and type of columns must match

```
hive> from (from stocks select stocks.symbol, stocks.price_open where stocks.volume < 1000

> union all
> from stocks select stocks.symbol, stocks.price_open  where stocks.volume <150000)

> unioninput

> insert overwrite directory 'output/union.out' select unioninput.*;

OK

Time taken: 14.794 seconds
```

–NOTE: Above query uses same table for union; same results can be achieved using SELECT & WHERE clause

[cloudera@localhost ~]$ hadoop fs -ls output/union.out;

Found 1 items

-rw-r–r– 3 cloudera supergroup 278413 2015-02-23 14:58 output/union.out/000000_0

[cloudera@localhost ~]$ hadoop fs -cat output/union.out/000000_0

sample_output:

CRT30.1

CRT30.7

CRT30.7

CRT31.56

VIEWS

–> Materialized views are not supported in Hive

–> Views can be used to reduce query complexity

```
hive> select name, size(subordinates)  from employees_no_partition;

OK

John Doe               2
Mary Smith             1
Todd Jones             0
Bill King              0
Boss Man               2
Fred Finance           1
Stacy Accountant       0

Time taken: 20.045 seconds
```

```
hive> create view managers as select * from employees_no_partition where size(subordinates)

OK
Time taken: 1.185 seconds

hive> show tables;

OK
employees
employees_no_partition
managers
numbers
numbers_bucketed
numbers_clustered
stocks
test

Time taken: 0.484 seconds


hive> select name from managers;

OK
john Doe
Mary Smith
Boss Man
Fred Finance

Time taken: 16.062 seconds
```

```
hive>describe formatted managers;

sample_output:

OK

# col_name             data_type                  comment

name                   string                      None

salary                  float                      None
```

```
subordinates          array<string>               None

deductions          map<string,float>             None

address        struct<street:string,city:string,  None
                    state:string,zip:int>
```

```
# Detailed Table Information

Database:         mydb

Owner:            cloudera

CreateTime:       Tue Feb 24 00:08:24 CST 2015

Table Type:       VIRTUAL_VIEW
```

# View Information

View Original Text: select * from employees_no_partition where size(subordinates) > 0

–NOTE: Table Type: MANAGED_TABLE –> for managed tables

Table Type: EXTERNAL_TABLE –> for external tables

-CREATE TABLES FROM VIEWS:

```
hive> create table tblmanagers like managers;

OK

Time taken: 2.956 seconds

hive> create external table exttblmanagers like managers;

OK

Time taken: 0.757 seconds
```

-ALTER VIEW:

```
hive> alter view managers set tblproperties('created_at'='some_timestamp');

OK

Time taken: 2.435 seconds
```

-DROP VIEW:

```
hive> drop view if exists mangers;
```

-INDEX:

```
create table mydb.employees_partitioned(name string,salary float,
subordinates array<string>,deductions map<string, float>,
address struct<street:string, city:string, state:string, zip:int>
partitioned by (country string, state string);
```

[cloudera@localhost ~]$ cat scripts/create_index.hql;

```
create index employees_partitioned_index on
 table mydb.employees_partitioned (country) as 'org.apache.hadoop.hive.ql.index.compact.Com
with deferred rebuild idxproperties
('creator'='me', 'created_at'='some_time')in table mydb.employees_index_tablePARTITIONED BY
 (country, name)
comment 'employees_partitioned
 by country and name');
```

–??????????????

[cloudera@localhost ~]$ hive -f 'scripts/create_index.hql';

Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-common-0.10.0-cdh4.3.0.jar!/hive-log4j.properties

Hive history file=/tmp/cloudera/hive_job_log_df686526-e57d-49c3-a7e4-7325a6ce2c17_1322001493.txt

FAILED: ParseException line 7:0 missing EOF at 'PARTITIONED' near 'employees_index_table'

–??????????????

–NOTE: Bitmap indexes are commonly used for columns with few distinct values

–> WITH DIFFERED REBUILD, the new index starts empty; at any time, the index can by built the first time or rebuilt using ALTER INDEX statement:

```
alter index employee_index

on table employees

partition(country='US')

rebuild;
```

-> If PARTITION clause is omitted, the index is rebuilt on all partitions

–> show formatted index on employees;

–> drop index if exists employee_index on table employees;

–> AVOID MULTIPLE PASSES OVER THE SAME DATA:

–> Inefficient:

```
INSERT OVERWRITE TABLE sales

SELECT * FROM history WHERE action='purchased';

INSERT OVERWRITE TABLE credits

SELECT * FROM history WHERE action='returned';
```

–> Efficient:

```
FROM history

INSERT OVERWRITE TABLE sales SELECT * WHERE action='purchased';

INSERT OVERWRITE TABLE credit SELECT * WHERE action='returned';
```

-INDEXES:

–> Hive has limited indexing capabilites

–> There are no keys; but you can build indexes on columns to speed some operations

–> Index data for a table is stored in another table.

```
create index index_name on table table_name (country) as 'org.apache.hadoop.hive.ql.index.c

show index on table_name;

drop index index_name on table_name;
```

-EXPLAIN:

```
hive> explain select symbol, count(symbol) from stocks group by symbol;

OK
```

ABSTRACT SYNTAX TREE:

```
(TOK_QUERY (TOK_FROM (TOK_TABREF (TOK_TABNAME stocks))) (TOK_INSERT (TOK_DESTINATION (TOK_D
```

STAGE DEPENDENCIES:

Stage-1 is a root stage

Stage-0 is a root stage

STAGE PLANS:

```
Stage: Stage-1

Map Reduce

Alias -> Map Operator Tree: stocks

TableScan

alias: stocks

Select Operator

expressions:

expr: symbol

type: string

outputColumnNames: symbol

Group By Operator

aggregations:

expr: count(symbol)

bucketGroup: false

keys:

expr: symbol

type: string

mode: hash

outputColumnNames: _col0, _col1

Reduce Output Operator
```

–NOTE: You can also use:

explain EXTENDED select symbol, count(symbol) from stocks group by symbol;

——>TUNING

- Use [EXTEDNED] EXPLAIN
- Tune LIMIT clause (use set)
- Tune JOIN clause(use map-sie join on small datasets)
- use Local Mode on small datasets(use set)
- Parallel Execution (execute jobs in parallel)(use set)
- Strict Mode (use set hive.mapred.mode=strict). Enforces the following:
  - enforces WHERE clause on partitioned tables
  - enforces LIMIT clause with ORDER BY
  - enforces ON clause with catesian product (JOIN)
- Tuning number of reducers:
  - set hive.exec.reducers.bytes.per.reducer=750,000,000
  - set hive.exec.reducers.max (max reducer count)
  - reduce.tasks (reducer count)
- JVM Reuse (use set mapred.job.reuse.jvm.num.tasks=10)
- Use Indexes
- Tune dynamic partition:
  - exec.dynamic.partition.mode=strict (at least one static partition is needed)
  - exec.max.dynamic.partitions
  - exec.max.dynamic.partitions.pernode
- o Speculative Execcution (Hadoop launches multiple instances of map/reduce jobs):
  - map.tasks.speculative.execution=true;
  - reduce.tasks.speculative.execution=true;
  - mapred.reduce.tasks.speculative.execution=true;
- Use single mapreduce job to combine multiple GROUP BY in a query(common group by key is rqd)(use set)
- Use Virtual columns (see example below)

```
hive> select INPUT__FILE__NAME, BLOCK__OFFSET__INSIDE__FILE, symbol

from stocks  where symbol like '%CLI%' limit 5;

hdfs://localhost.localdomain:8020/user/cloudera/myhive/NYSE_daily              0          (

hdfs://localhost.localdomain:8020/user/cloudera/myhive/NYSE_daily              57         (
hdfs://localhost.localdomain:8020/user/cloudera/myhive/NYSE_daily             114        (
hdfs://localhost.localdomain:8020/user/cloudera/myhive/NYSE_daily             171        (
hdfs://localhost.localdomain:8020/user/cloudera/myhive/NYSE_daily             228        (
Time taken: 17.629 seconds
```

NOTE: INPUT__FILE__NAME & BLOCK__OFFSET__INSIDE__FILE needs to be in upper case; seperator is 2 dashes(_ _)

SCRIPTS:

[cloudera@localhost ~]$ cat scripts/create_table.hql

CREATE TABLE IF NOT EXISTS mydb.employees (

name STRING COMMENT 'Employee name',

salary FLOAT COMMENT 'Employee salary',

subordinates ARRAY<STRING> COMMENT 'Names of subordinates',

deductions MAP<STRING, FLOAT> COMMENT 'Keys are deductions names, values are percentages',

address STRUCT<street:STRING, city:STRING, state:STRING, zip:INT> COMMENT 'Home address')

COMMENT 'Description of the table'

LOCATION '/user/hive/warehouse/'

TBLPROPERTIES ('creator'='me', 'created_at'='2015-02-19 9:55:00');

[cloudera@localhost ~]$ hive -f 'scripts/create_table.hql'

OTHERS:

–CHANGING HIVE PROMPT:

```
hive> set hive.cli.print.current.db=true;
hive (default)>
```

-PRINT column headers;

```
hive> set hive.cli.print.header=true;
```

–HIVE ONESHOT COMMANDS:

[cloudera@localhost ~]$ hive -e 'select * from records LIMIT 5';

[cloudera@localhost ~]$ hive -e 'select * from mydb.stocks limit 5;'

–EXECUTE UNIX COMMANDS:

```
hive (test)> !pwd;
hive (test)> /home/cloudera
```

–INSTALLATION LOCATION:

[cloudera@localhost ~]$ ls /usr/lib

–HIVE INSTALLATION LOCATION

[cloudera@localhost ~]$ ls /usr/lib/hive/lib

cloudera@localhost ~]$ ls -l /usr/lib/hive/bin

total 28

-rwxr-xr-x. 1 root root 881 May 27 2013 beeline

drwxr-xr-x. 3 root root 4096 Jun 18 2013 ext

-rwxr-xr-x. 1 root root 5781 May 27 2013 hive

-rwxr-xr-x. 1 root root 1900 May 27 2013 hive-config.sh

```
-rwxr-xr-x. 1 root root 885 May 27 2013 hiveserver2
-rwxr-xr-x. 1 root root 832 May 27 2013 metatool
[cloudera@localhost ~]$
```

–HIVE CONFIGURATION:

```
[cloudera@localhost ~]$ ls /usr/lib/hive/conf
```

–HIVE LOG:

[NOTE: cat /tmp/cloudera/hive.log displays the log contents

–HIVE LOGGING:

Open following file in text editor and you can change logging level:

/usr/lib/hive/lib/hive-common-0.10.0-cdh4.3.0.jar/hive-log4j-properties

```
[cloudera@localhost ~]$ hive -help
usage: hive
-d,–define <key=value> Variable subsitution to apply to hive
commands. e.g. -d A=B or –define A=B
–database <databasename> Specify the database to use
-e <quoted-query-string> SQL from command line
-f <filename> SQL from files
-H,–help Print help information
-h <hostname> connecting to Hive Server on remote host
–hiveconf <property=value> Use value for given property
–hivevar <key=value> Variable subsitution to apply to hive
commands. e.g. –hivevar A=B
-i <filename> Initialization SQL file
-p <port> connecting to Hive Server on port number
-S,–silent Silent mode in interactive shell
-v,–verbose Verbose mode (echo executed SQL to the
console)
```

–HIVE SERVICES:

```
[cloudera@localhost ~]$ hive –service help
```

[NOTE: cli : command line interface (default)

hiveserver: runs hive as a server (exposes a thirft service)

hwi: hive web interface

jar: similar to hadoop jar

metastore: runs metastore as a standalone process]

[NOTE: Thrift client: lets you run hive commands from different programming languages]

```
[cloudera@localhost ~]$ hive –help
Usage ./hive <parameters> –service serviceName <service parameters>
Service List: beeline cli help hiveserver2 hiveserver hwi jar lineage metastore metatool rcfilecat
Parameters parsed:
–auxpath : Auxillary jars
–config : Hive configuration directory
```

–service : Starts specific service/component. cli is default

Parameters used:

HADOOP_HOME or HADOOP_PREFIX : Hadoop install directory

HIVE_OPT : Hive options

For help on a particular service:

./hive –service serviceName –help

Debug help: ./hive –debug –help

–CLI SERVICE

[cloudera@localhost ~]$ hive –help –service cli

usage: hive

-d,–define <key=value> Variable subsitution to apply to hive commands. e.g. -d A=B or –define A=B

–database <databasename> Specify the database to use

-e <quoted-query-string> SQL from command line

-f <filename> SQL from files

-H,–help Print help information

-h <hostname> connecting to Hive Server on remote host

–hiveconf <property=value> Use value for given property

–hivevar <key=value> Variable subsitution to apply to hive commands. e.g. –hivevar A=B

-i <filename> Initialization SQL file

-p <port> connecting to Hive Server on port number

-S,–silent Silent mode in interactive shell

-v,–verbose Verbose mode (echo executed SQL to the console)

— HIVE NAMESPACES:

hivevar

hiveconf

system

env

–SELECT in NON-INTERACTIVE MODE:

[cloudera@localhost ~]$ hive -e 'select * from records';

Logging initialized using configuration in

[cloudera@localhost ~]$ hive -S -e "set" | grep warehouse

hive.metastore.warehouse.dir=/user/hive/warehouse

hive.warehouse.subdir.inherit.perms=true

–HIVE FILES:

[cloudera@localhost ~]$ cat scripts/select.q

select * from records;

–CHANGE TO ROOT USER

[cloudera@localhost ~]$ su root

Password: –>cloudera

[root@localhost cloudera]#

–EXECUTE HADOOP COMMANDS:

```
hive (test)> dfs -ls output;
```

–LIST ALL HADOOP COMMANDS:

```
hive> dfs -help;
```

–COMMENTS IN HIVE:

–this is a comment

WORK:

hadoop1@ubuntu:~/scripts$ cat nyse_daily.hql

```
create external table if not exists
mydb.nyse_daily (exhange string,symbol string,
current_date date,open float,
high float,low float,
close float,volume int,
adj_close float)row format delimited
fields terminated by '\t';
```

hadoop1@ubuntu:~/scripts$ hive -f 'nyse_daily.hql'

```
hive (mydb)> load data inpath '/home/hadoop1/data/NYSE_daily' overwrite into table nyse_dai
```

FAILED: SemanticException Line 1:17 Invalid path "/home/hadoop1/data/NYSE_daily": No files matching path hdfs://localhost:9000/home/hadoop1/data/NYSE_daily

hadoop1@ubuntu:~/data$ hadoop fs -put NYSE* data

```
hive (mydb)> load data inpath 'data/NYSE_daily' overwrite into table nyse_daily;

hive (mydb)> select * from nyse_daily limit 10;
```

Share this:

G+ Tweet

About Siva

Senior Hadoop developer with 4 years of experience in designing and architecture solutions for the Big Data domain and has been involved with several complex engagements. Technical strengths include Hadoop, YARN, Mapreduce, Hive, Sqoop, Flume, Pig, HBase, Phoenix, Oozie, Falcon, Kafka, Storm, Spark, MySQL and Java.

View all posts by Siva →

Leave a comment

Your email address will not be published. Required fields are marked *

b i link b-quote ~~del~~ ins img ul ol li code more close tags crayon ¶

Name *

Email *

Website

Post Comment

2 thoughts on “Hive Functions Examples”

Suresh Perumal

Reply ↓

June 6, 2015 at 5:31 pm

Dear Siva

Surely this posting will be useful for <a href = “http://itcoordinatestraining.com”>hadoop </a>developer . Clearly explained with examples. thank u.

with Regards

Suresh Perumal

ITCoordinates Mylapore

Ms Revathi

Reply ↓

July 1, 2016 at 8:28 pm

such an awesome site this is…Loved it..very useful and helpful examples.

Post navigation

← Pig Functions Examples

100 Hadoop Certification Dump Questions →

Review Comments

Course content is well structured. I like Siva's explanation of topics using slide decks & virtual machine (CDH cluster) at the same time,this will help audience to learn not only theory behind a topic but also practical aspect of it. Overall, I would recommend this course.

Kumar
Big Data Developer
Hadoop&Aug/2016
September 26, 2016

.

Recent Comments

›WENJIE LI on Creating Custom Hadoop Writable Data Type
›Brian on Avro Serializing and Deserializing Example – Java API
›chandan on Hive Authorization Models and Hive Security
›venkat on Hive Database Commands
›Siva on Avro Serializing and Deserializing Example – Java API

Hadooptutorial.info

Hadooptutorial.info

Let's get Social :

Back to top

SEARCH NAVIGATION SOCIAL

Custom Search

- Home
- Big Data
- Hadoop
- Map Reduce
- Hive
- Pig
- HBase
- Flume
- Interview Questions
- Miscellaneous

- Twitter
- Facebook
- Google+