vi employee.csv
101,Amit,HADOOP:HIVE:SPARK:BIG-DATA
102,Sumit,HIVE:OOZIE:HADOOP:SPARK:STORM
103,Rohit,KAFKA:CASSANDRA:HBASE

USE itunes_fuse_semantic_app;

CREATE TABLE employee
(
id INT,
name STRING,
skills ARRAY<STRING>
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
COLLECTION ITEMS TERMINATED BY ':';

LOAD DATA LOCAL INPATH 'employee.csv'
INTO TABLE employee;

Working with Array operators

SELECT
size(skills),
array_contains(skills, 'HADOOP'),
sort_array(skills),
concat_ws("|", skills)
FROM employee;

4 true ["BIG-DATA","HADOOP","HIVE","SPARK"] HADOOP|HIVE|SPARK|BIG-DATA
5 true ["HADOOP","HIVE","OOZIE","SPARK","STORM"]
HIVE|OOZIE|HADOOP|SPARK|STORM
3 false ["CASSANDRA","HBASE","KAFKA"] KAFKA|CASSANDRA|HBASE

Exploding contents of an array

SELECT explode(skills) AS skills FROM employee;
--AS clause is required as explode() is UDTF, ie. generates output as TABLE.

HADOOP
HIVE
SPARK
BIG-DATA
HIVE
OOZIE
HADOOP
SPARK
STORM
KAFKA
CASSANDRA
HBASE

Expanding contents of an array with other columns

```
SELECT id, name, skill
FROM employee LATERAL VIEW explode(skills) skill_set
AS skill;
```

```
101	Amit	HADOOP
101	Amit	HIVE
101	Amit	SPARK
101	Amit	BIG-DATA
102	Sumit	HIVE
102	Sumit	OOZIE
102	Sumit	HADOOP
102	Sumit	SPARK
102	Sumit	STORM
103	Rohit	KAFKA
103	Rohit	CASSANDRA
103	Rohit	HBASE
```

Here skill_set is the table which contains single column with alias skill.

==========

```
SET hive.mapred.mode=nostrict; --default is nostrict
SELECT * FROM users ORDER BY name ASC;
SELECT * FROM users SORT BY name ASC;
```

The two queries look almost identical, but if more than one reducer is invoked, the output will be

sorted differently.

```
set mapred.reduce.tasks=2;
SELECT * FROM users SORT BY name ASC;
```

==========

```
SET mapred.reduce.tasks=2;
SELECT * FROM users DISTRIBUTE BY unit SORT BY name ASC;


SELECT * FROM users DISTRIBUTE BY unit SORT BY name ASC;
SELECT * FROM users CLUSTER BY unit;
```

==========

```
-bash-4.1$ vi users.txt
1       Amit    100     DNA
2       Sumit   200     DNA
3       Yadav   300     DNA
4       Sunil   500     FCS
5       Kranti  100     FCS
6       Mahoor          200     FCS
8       Chandra         500     DNA

-bash-4.1$ vi locations.txt
1       UP
2       BIHAR
3       MP
4       AP
5       MAHARASHTRA
6       GOA
7       JHARKHAND

USE default;

CREATE TABLE users
(
id INT,
name STRING,
salary INT,
unit STRING
)
```

```
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t';

CREATE TABLE locations
(
id INT,
location STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t';

LOAD DATA LOCAL INPATH '/root/hive/users.txt'
INTO TABLE users;

LOAD DATA LOCAL INPATH '/root/hive/locations.txt'
INTO TABLE locations;

CREATE TABLE buck_users
(
id INT,
name STRING,
salary INT,
unit STRING
)
CLUSTERED BY (id)
SORTED BY (id)
INTO 2 BUCKETS;

CREATE TABLE buck_locations
(
id INT,
location STRING
)
CLUSTERED BY (id)
SORTED BY (id)
INTO 2 BUCKETS;

SET hive.enforce.bucketing=true;

INSERT OVERWRITE TABLE buck_users
SELECT * FROM users;

INSERT OVERWRITE TABLE buck_locations
SELECT * FROM locations;
```

--View the number of files created at the table location.
--It should be two.

==========

Inner Join

```
SELECT * FROM buck_users u INNER JOIN buck_locations l
ON u.id = l.id;
```

Left Outer Join

```
SELECT * FROM buck_users u LEFT OUTER JOIN buck_locations l
ON u.id = l.id;
```

Right Outer Join

```
SELECT * FROM buck_users u RIGHT OUTER JOIN buck_locations l
ON u.id = l.id;
```

Full Outer Join

```
SELECT * FROM buck_users u FULL OUTER JOIN buck_locations l
ON u.id = l.id;
```

Cartesian Cross Product Join (Less Used)

```
SELECT * FROM buck_users u JOIN buck_locations l
ON u.id = l.id;
```

==========

############## Use AcadGild VM ##

CREATING emp_details TABLE

```
create table emp_details
(
emp_name string,
unit string,
exp int,
location string
)
row format delimited
fields terminated by ',';
```

LOADING emp_details TABLE

```
load data local inpath '/home/acadgild/hive/emp_details.txt'
into table emp_details;

describe formatted emp_details;

dfs -ls hdfs://localhost:9000/user/hive/warehouse/emp_details;
```

CREATING emp_details_partitioned TABLE

```
create table emp_details_partitioned
(
emp_name string,
unit string,
exp int
)
partitioned by (location string);
```

LOADING emp_details_partitioned TABLE with Static Partitions

```
insert overwrite table emp_details_partitioned
```

```
partition(location = 'BBSR')
select emp_name, unit, exp from emp_details
where location = 'BBSR';



------------------------------------------------------------------
LOADING emp_details_partitioned TABLE with Dynamic Partitions
------------------------------------------------------------------
set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table emp_details_partitioned
partition (location)
select * from emp_details;

select count(*) from emp_details where location='BBSR';

select count(*) from emp_details where name='Aditya';

------------------------------------------------------------------
DROPIING PARTITIONS FROM emp_details_partitioned TABLE
------------------------------------------------------------------

alter table emp_details_partitioned drop partition(location='BBSR');


==============================================
SELECT * from users TABLESAMPLE(BUCKET 3 OUT OF 10 ON rand()) s;
SELECT * from users TABLESAMPLE(BUCKET 3 OUT OF 10 ON rand()) s;

SELECT * from users TABLESAMPLE(BUCKET 2 OUT OF 4 ON name) s;

SELECT * FROM buck_users TABLESAMPLE(BUCKET 1 OUT OF 2 ON id) s LIMIT 1;



================================================================
--------------------------
Creating regular text table
--------------------------
create table text_table
(
c1 int,
c2 int,
```

```
c3 int,
c4 int
)
row format delimited
fields terminated by '|';

---------------------------
Loading into text table
---------------------------

load data local inpath '/root/hive/datasets_for_fileformats/ratings.dat'
into table text_table;

---------------------------
Creating SequenceFile table
---------------------------

create table seq_table
(
c1 int,
c2 int,
c3 int,
c4 int
)
stored as SEQUENCEFILE;

---------------------------
Creating RC Format table
---------------------------

create table rc_table
(
c1 int,
c2 int,
c3 int,
c4 int
)
stored as RCFILE;

---------------------------
Creating Parquet File table
---------------------------

create table prq_table
```

```
(
c1 int,
c2 int,
c3 int,
c4 int
)
stored as PARQUET;

---------------------------
Creating ORC Format table
---------------------------

create table orc_table
(
c1 int,
c2 int,
c3 int,
c4 int
)
stored as ORC;

---------------------------------------
Loading All the tables in a single pass
---------------------------------------

FROM text_table
INSERT OVERWRITE TABLE seq_table SELECT *
INSERT OVERWRITE TABLE rc_table SELECT *
INSERT OVERWRITE TABLE prq_table SELECT *
INSERT OVERWRITE TABLE orc_table SELECT *;

---------------------------------------
Comparing sizes of loaded tables
---------------------------------------

describe formatted orc_table;

dfs -ls hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/text_table;
-rw-r--r--   1 root hdfs    4135847 2016-08-25 11:13
hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/orc_table/000000_0

dfs -ls hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/orc_table;
-rw-r--r--   1 root hdfs   21593504 2016-08-25 11:12
hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/text_table/ratings.dat
```

dfs -ls hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/seq_table;
-rw-r--r--   1 root hdfs   33928859 2016-08-25 11:13
hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/seq_table/000000_0

dfs -ls hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/rc_table;
-rw-r--r--   1 root hdfs   11992620 2016-08-25 11:13
hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/rc_table/000000_0

dfs -ls hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/prq_table;
-rw-r--r--   1 root hdfs   5941753 2016-08-25 11:13
hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/prq_table/000000_0

Enablig Compression

Enabling compression provides performance gains in most cases and is supported for RCFile and SequenceFile tables.
For example, to enable Snappy compression, you would specify the following additional settings when loading data through the Hive shell.

SET hive.exec.compress.output=true;
SET mapred.max.split.size=256000000;
SET mapred.output.compression.type=BLOCK; -- block compression for sequence file
SET mapred.output.compression.codec=org.apache.hadoop.io.compress.SnappyCodec;

FROM text_table
INSERT OVERWRITE TABLE seq_table SELECT *
INSERT OVERWRITE TABLE rc_table SELECT *
INSERT OVERWRITE TABLE prq_table SELECT *
INSERT OVERWRITE TABLE orc_table SELECT *;

Comparing sizes of loaded tables after compression (RC Files and Sequence Files are benefited the most)
---

dfs -ls hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/text_table;
-rw-r--r--   1 root hdfs   21593504 2016-08-25 11:12
hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/text_table/ratings.dat

dfs -ls hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/orc_table;

-rw-r--r--   1 root hdfs   4135847 2016-08-25 11:22
hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/orc_table/000000_0

dfs -ls hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/seq_table;
-rw-r--r--   1 root hdfs   10910048 2016-08-25 11:22
hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/seq_table/000000_0

dfs -ls hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/rc_table;
-rw-r--r--   1 root hdfs   6352282 2016-08-25 11:22
hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/rc_table/000000_0

dfs -ls hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/prq_table;
-rw-r--r--   1 root hdfs   5941753 2016-08-25 11:22
hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/prq_table/000000_0

==========