

# 6. Hive QL 실습

## 목 차

1. Hive 테이블 생성
2. Hive 실습
3. Hive 업로드 파일 보기
4. Hive 검색 결과 로컬 저장
5. Hive QL실습

# 1. Hive 테이블 생성

## Hive 실행/테이블보기/테이블 생성

```
hadoop@master:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[hadoop@master ~]$  
[hadoop@master ~]$ hive  
Logging initialized using configuration in file: /home/hadoop/.hadoop/hive-1.2.2-bin/conf/hive-log4j.properties  
hive> show tables;  
OK  
Time taken: 0.994 seconds  
hive> create table test_tab(sno int, sname string);  
OK  
Time taken: 0.489 seconds  
hive> show tables;  
OK  
test_tab  
Time taken: 0.034 seconds, Fetched: 1 row(s)  
hive> █
```

Table 삭제  
hive> drop table 테이블명

## Hive 테이블은 hdfs의 /user/hive/warehouse 디렉터리에 생성됨

hadoop@master:~

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

[hadoop@master conf]\$

[hadoop@master conf]\$

[hadoop@master conf]\$ cd

[hadoop@master ~]\$

[hadoop@master ~]\$ hdfs dfs -ls /user/hive/warehouse

Found 1 items

drwxrwxr-x - hadoop supergroup 0 2018-07-10 15:29 /user/hive/warehouse/test\_tab

[hadoop@master ~]\$

Hive의 Table

## 파티션 설정으로 테이블 생성 및 스키마 보기

테이블 생성 시 파티션 설정  
(속도 향상, 파티션 별 조회)

```
hadoop@master:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
Logging initialized using configuration in file:/home/hadoop/apache-hive-1.2.2-bin/conf/hive  
-log4j.properties  
hive> create table init_table(sid int, name string) partitioned by (dname string);  
OK  
Time taken: 1.126 seconds  
hive> describe init_table;  
OK  
sid                int  
name               string  
dname              string  
# Partition Information  
# col_name          data_type          comment  
dname              string  
Time taken: 0.319 seconds, Fetched: 8 row(s)  
hive> █
```

## 파티션 테이블 생성과 데이터 입력 예

```
hive> create table student(sid int, name string, gender  
string) PARTITIONED BY (enterYear int);
```

```
hive> insert overwrite table student  
PARTITION(enterYear = '2018')  
Select *  
From student_raw  
Where year = 2018;
```

# Hive 컬럼 자료형

Type	의미
TINYINT	1바이트 정수
SAMLLINT	2바이트 정수
INT	4바이트 정수
BIGINT	8바이트 정수
BOOLEAN	TRUE/FALSE
FLOAT	4바이트 실수
DOUBLE	8바이트 실수
STRING	문자열

## 2. Hive 실습(NASDAQ 증권 data)

### 1) Table 생성/data 디렉터리 생성

```
Master - VMware Workstation 12 Player (Non-commercial use only)
Player | [Icons] | ko (수) 12:50
프로그램 | 위치 | 터미널
hadoop@master:~/hive
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
invites
pokes
Time taken: 0.866 seconds, Fetched: 2 row(s)
hive> create table if not exists stocks(
  > exchange string, symbol string, ymd string,
  > price_open float,
  > price_high float,
  > price_low float,
  > price_close float,
  > volume int,
  > price_adj_close float)
  > row format delimited fields terminated by ',';
OK
Time taken: 0.527 seconds
hive>
```

```
hive> quit
[hadoop@master hive]$ mkdir stock_data
```



## 2) CSV 파일 다운로드(\*.zip 파일 ftp 다운로드)

Firefox 웹 브라우저

HFS / - Mozilla Firefox

FTP 이용 파일 다운로드

192.168.0.13

Name .extension	Size	Timestamp	Hits
 infochimps_dataset_4777_download_16185-csv.zip	106.2 MB	2017-10-19 오후 5:29:16	1
 infochimps_dataset_4778_download_16677-csv.zip	124.0 MB	2017-10-19 오후 5:29:31	1

infochimps\_dataset\_4777\_download\_16185-csv.zip 여는 중...

열기 선택:

infochimps\_dataset\_4777\_download\_16185-csv.zip  
파일 유형: ZIP 압축 파일 (106 MB)  
원본 위치: http://192.168.0.13

Firefox가 이 파일을 열거나 컴퓨터에 저장할 수 있습니다:

☐ 열기(O): 압축 관리자 (기본)

☒ 파일 저장(S)

☐ 다시 묻지 않음(A)

취소 확인

Server information  
HttpFileServer 2.3i  
Server time: 2017-10-20 오후 3:43:48  
Server uptime: 00:03:44

[hadoop@master:~/hive/stock\_da...]

[infochimps\_dataset\_4777\_down...]

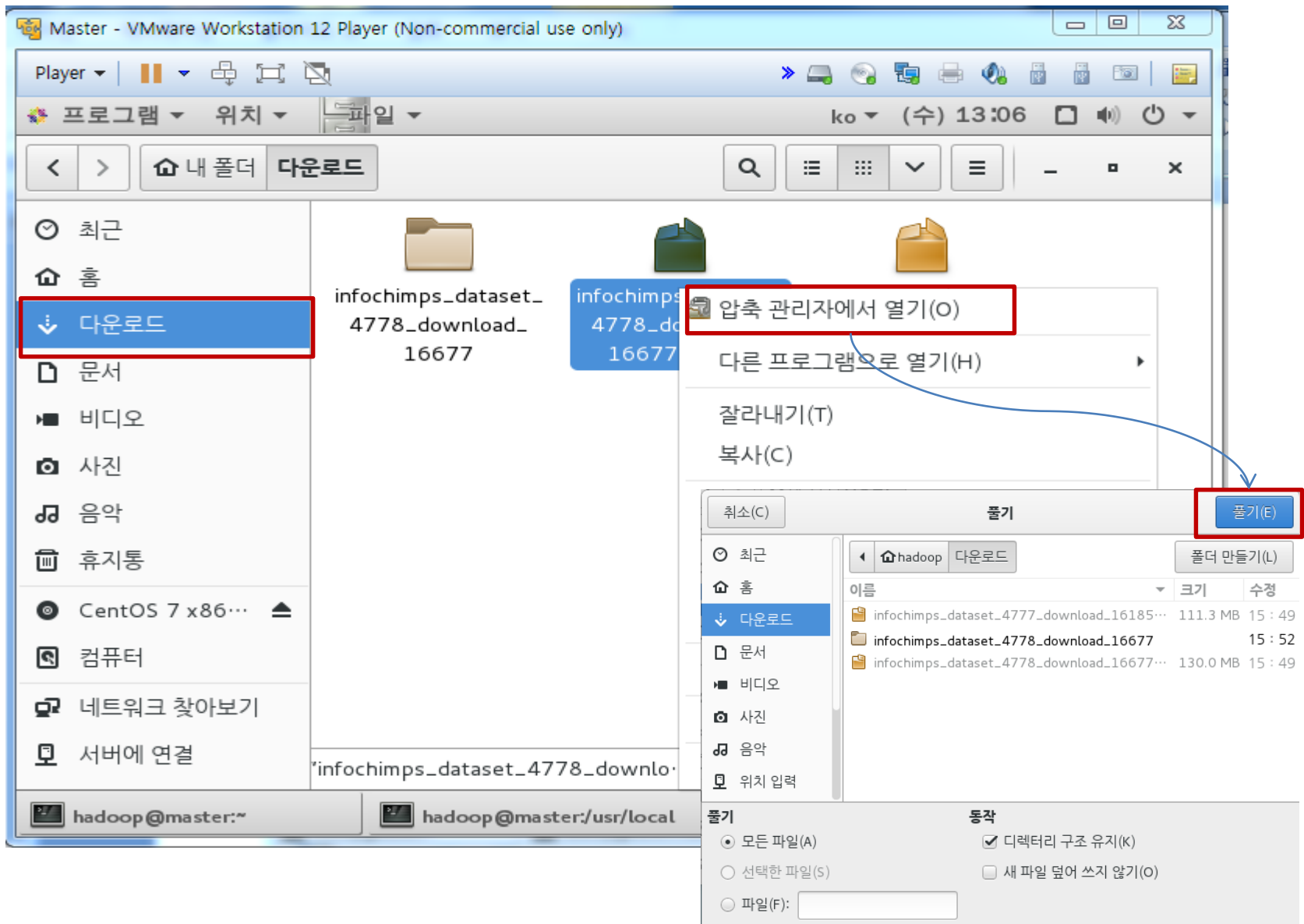
HFS / - Mozilla Firefox

다운로드

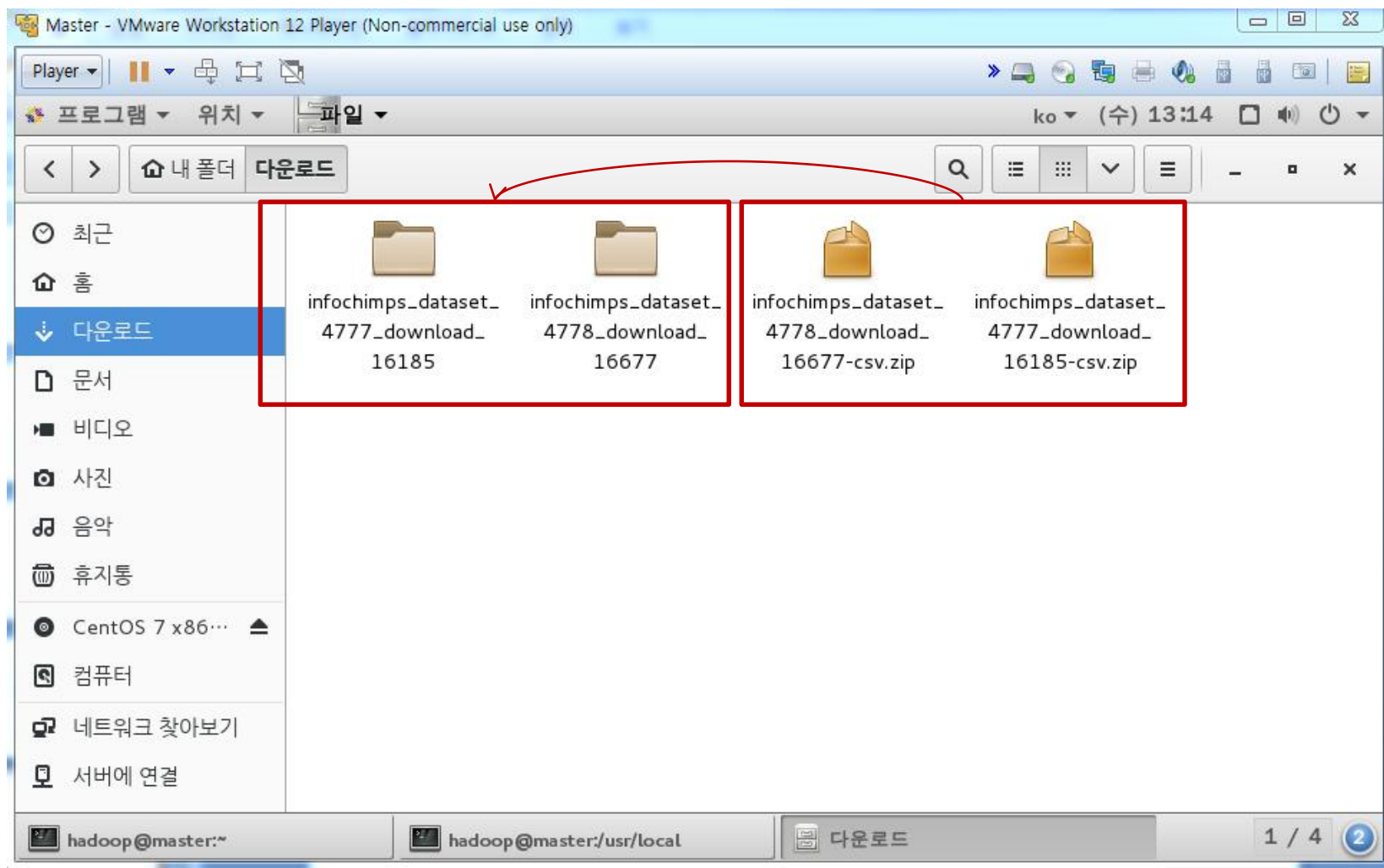
1 / 4



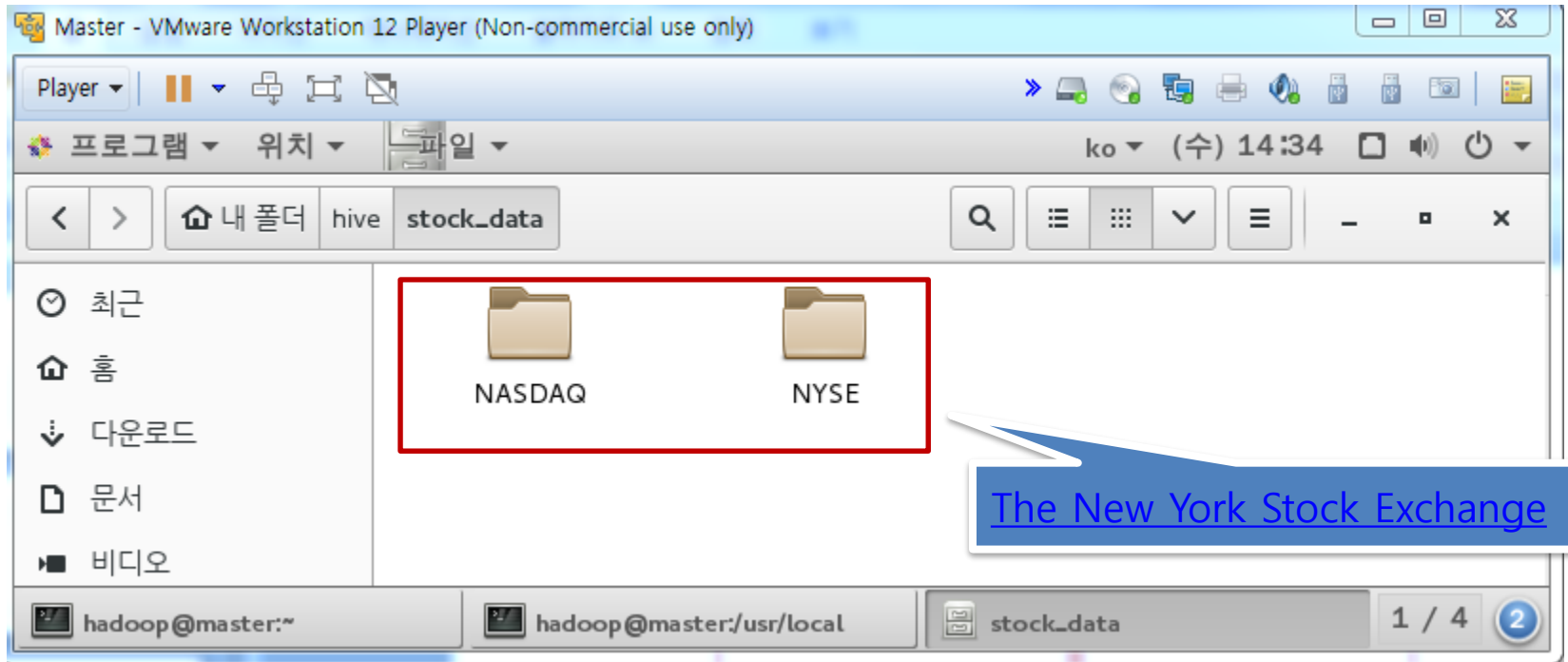
### 3) CSV 압축 파일 풀기



#### 4) CSV 압축 파일 풀기

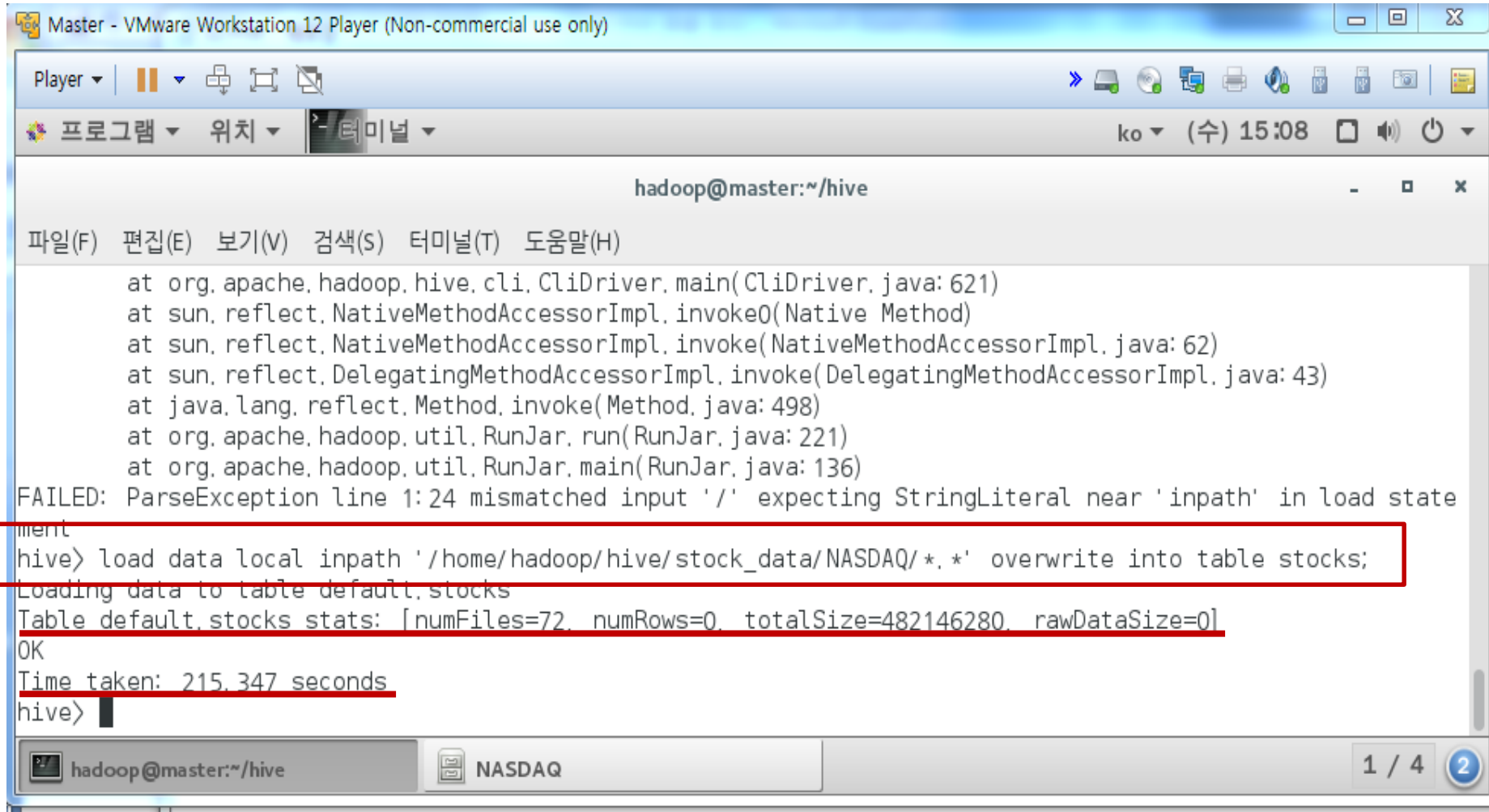


## 5) /home/hadoop/hive/stock\_data 디렉토리에 file 복사



## 6) File 업로드(NASDAQ)

- Hive는 로컬 파일과 HDFS에 저장된 데이터를 모두 업로드 할 수 있음

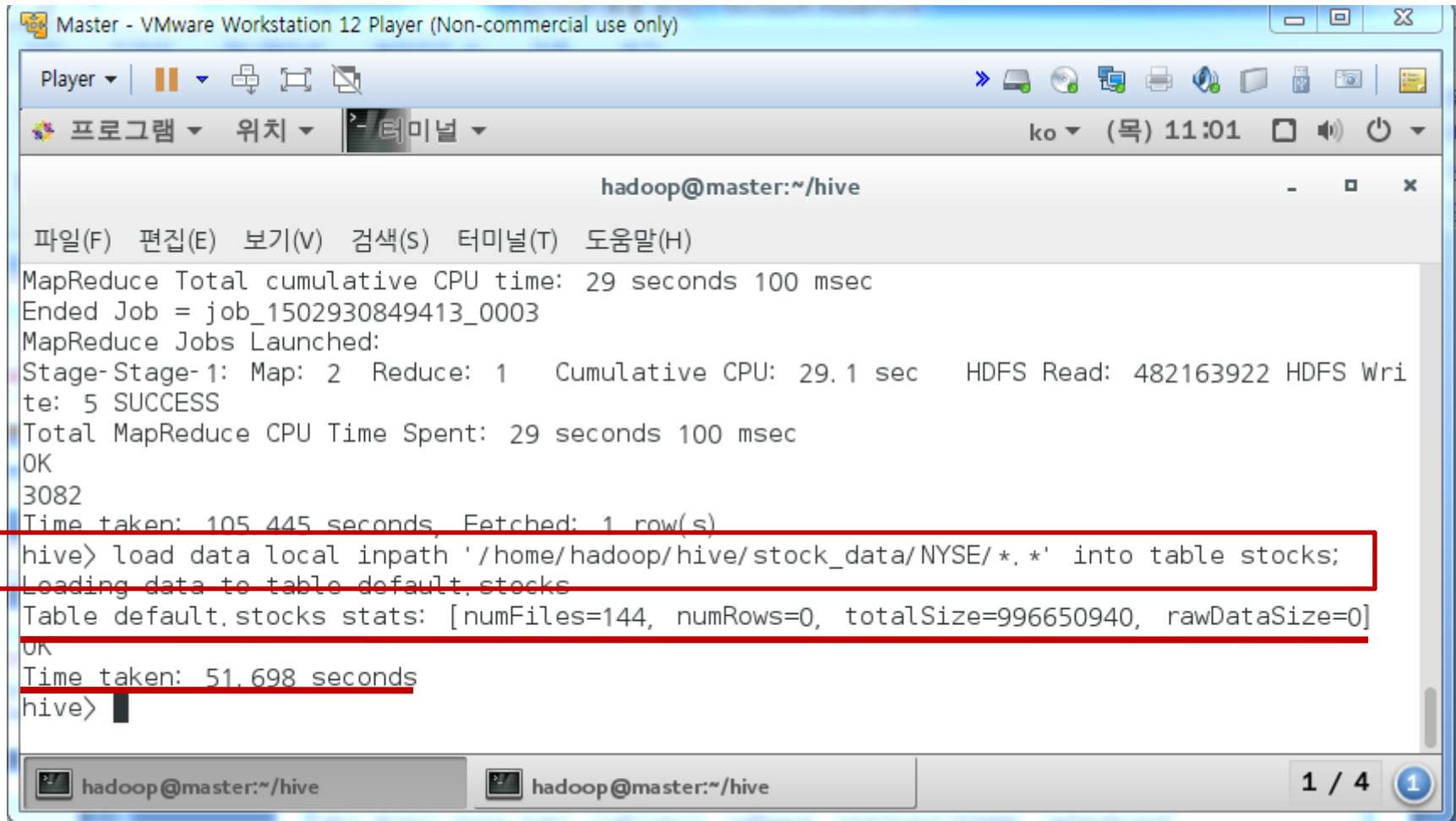


The screenshot shows a VMware Workstation 12 Player window titled "Master - VMware Workstation 12 Player (Non-commercial use only)". The main window displays a terminal window titled "hadoop@master:~/hive". The terminal window has a menu bar with "파일(F)", "편집(E)", "보기(V)", "검색(S)", "터미널(T)", and "도움말(H)". The terminal output shows the following:

```
at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:621)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at org.apache.hadoop.util.RunJar.run(RunJar.java:221)
at org.apache.hadoop.util.RunJar.main(RunJar.java:136)
FAILED: ParseException line 1:24 mismatched input '/' expecting StringLiteral near 'inpath' in load state
ment
hive> load data local inpath '/home/hadoop/hive/stock_data/NASDAQ/*.csv' overwrite into table stocks;
Loading data to table default.stocks
Table default.stocks stats: [numFiles=72, numRows=0, totalSize=482146280, rawDataSize=0]
OK
Time taken: 215.347 seconds
hive>
```

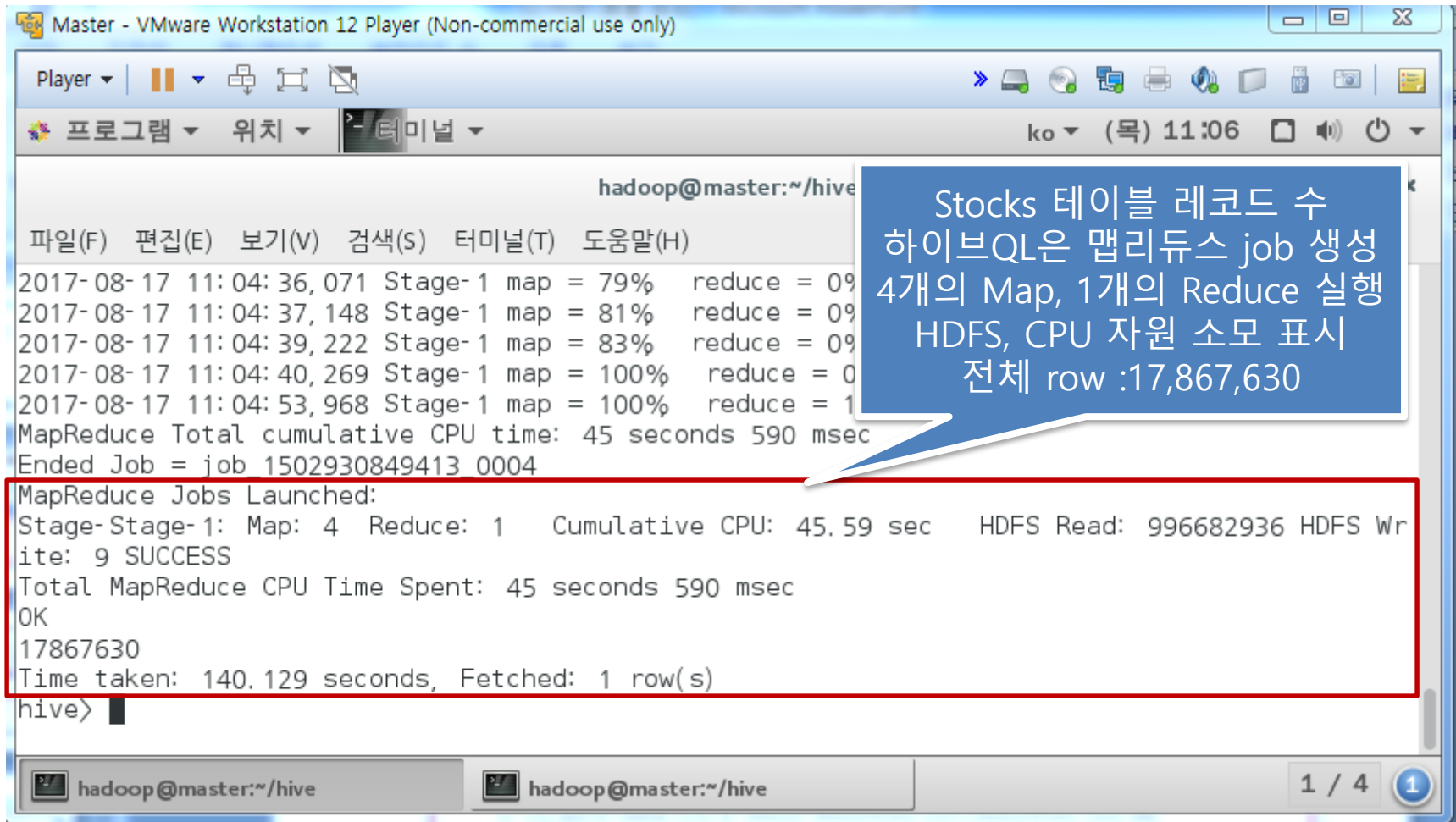
Below the terminal window, there is a file explorer bar showing the current directory as "hadoop@master:~/hive" and a file named "NASDAQ" selected. The bottom right corner of the window shows a page indicator "1 / 4" and a blue circular button with the number "2".

## File 업로드(NYSE) : **overwrite** 키워드 삭제



```
Master - VMware Workstation 12 Player (Non-commercial use only)
Player | || | | |
프로그램 | 위치 | 터미널 | ko | (목) 11:01 | | | |
hadoop@master:~/hive
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
MapReduce Total cumulative CPU time: 29 seconds 100 msec
Ended Job = job_1502930849413_0003
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2 Reduce: 1 Cumulative CPU: 29.1 sec HDFS Read: 482163922 HDFS Write: 5 SUCCESS
Total MapReduce CPU Time Spent: 29 seconds 100 msec
OK
3082
Time taken: 105.445 seconds, Fetched: 1 row(s)
hive> load data local inpath '/home/hadoop/hive/stock_data/NYSE/*. *' into table stocks;
Loading data to table default.stocks
Table default.stocks stats: [numFiles=144, numRows=0, totalSize=996650940, rawDataSize=0]
OK
Time taken: 51.698 seconds
hive>
```

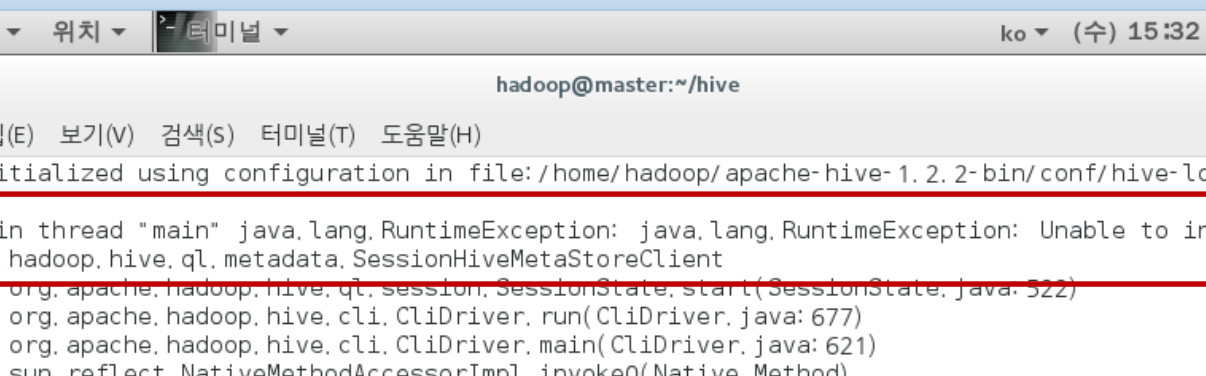
## 7) 전체 레코드 수 확인 : `hive> select count(*) from stocks;`



```
hadoop@master:~/hive
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
2017-08-17 11:04:36,071 Stage-1 map = 79% reduce = 0%
2017-08-17 11:04:37,148 Stage-1 map = 81% reduce = 0%
2017-08-17 11:04:39,222 Stage-1 map = 83% reduce = 0%
2017-08-17 11:04:40,269 Stage-1 map = 100% reduce = 0%
2017-08-17 11:04:53,968 Stage-1 map = 100% reduce = 1%
MapReduce Total cumulative CPU time: 45 seconds 590 msec
Ended Job = job_1502930849413_0004
MapReduce Jobs Launched:
Stage-Stage-1: Map: 4 Reduce: 1 Cumulative CPU: 45.59 sec HDFS Read: 996682936 HDFS Write: 9 SUCCESS
Total MapReduce CPU Time Spent: 45 seconds 590 msec
OK
17867630
Time taken: 140.129 seconds, Fetched: 1 row(s)
hive>
```

Stocks 테이블 레코드 수  
하이버QL은 맵리듀스 job 생성  
4개의 Map, 1개의 Reduce 실행  
HDFS, CPU 자원 소모 표시  
전체 row :17,867,630

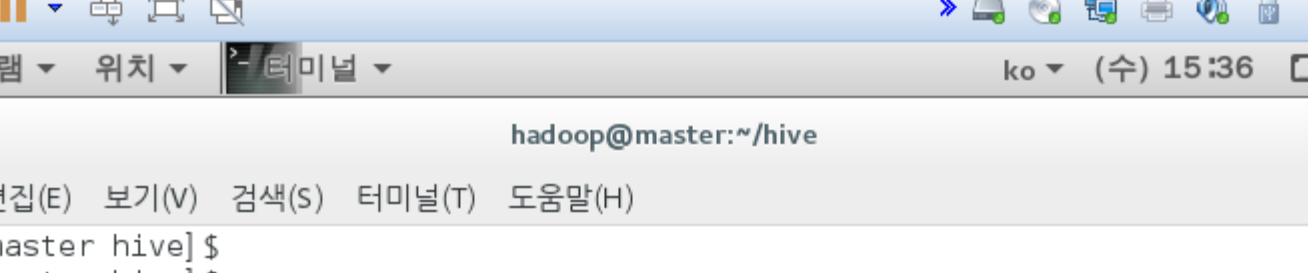
## File 업로드 오류



```
hadoop@master:~/hive
Logging initialized using configuration in file: /home/hadoop/apache-hive-1.2.2-bin/conf/hive-log4j.properties
Exception in thread "main" java.lang.RuntimeException: java.lang.RuntimeException: Unable to instantiate
org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClient
    at org.apache.hadoop.hive.ql.session.SessionState.start(SessionState.java:322)
    at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:677)
    at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:621)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at org.apache.hadoop.util.RunJar.run(RunJar.java:221)
    at org.apache.hadoop.util.RunJar.main(RunJar.java:136)
Caused by: java.lang.RuntimeException: Unable
veMetaStoreClient
    at org.apache.hadoop.hive.metastore.Me
    at org.apache.hadoop.hive.metastore.Re
6)
    at org.apache.hadoop.hive.metastore.Re
: 132)
    at org.apache.hadoop.hive.metastore.Re
: 104)
    at org.apache.hadoop.hive.ql.metadata.
```

Looks like problem with your metastore. If you are using the default hive metastore embedded derby. Lock file would be there in case of abnormal exit. if you remove that lock file this issue would be solved

```
rm metastore_db/*.lck
```



Master - VMware Workstation 12 Player (Non-commercial use only)

Player ▾ | [Icons] | ko ▾ (수) 15:36 [Icons]

프로그램 ▾ 위치 ▾ 터미널 ▾

hadoop@master:~/hive

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

```
[hadoop@master hive]$  
[hadoop@master hive]$  
[hadoop@master hive]$ ls  
LICENSE  README.txt      bin    derby.log  hadoop    lib          scripts  
NOTICE   RELEASE_NOTES.txt  conf  examples  hcatalog  metastore_db stock_data  
[hadoop@master hive]$ ls ./metastore_db/  
README_DO_NOT_TOUCH_FILES.txt  db.lck  dbex.lck  log  seg0  service.properties  tmp  
[hadoop@master hive]$  
[hadoop@master hive]$ rm metastore_db/ *.lck  
[hadoop@master hive]$
```

오류 해결 : 메타스토어의 lock  
관련 파일 삭제

hadoop@master:~/hive



## 8) Stocks 테이블 검색 : 시가(Open)가 1,5000 이상인 레코드 조회

하이브QL

```
select * from stocks where price_open >= 15000 ;
```

```
select exchange, symbol, ymd from stocks where price_open >= 500 limit 10 ;
```

The screenshot shows a VMware Workstation 12 Player window titled "Master - VMware Workstation 12 Player (Non-commercial use only)". Inside the player, there is a terminal window titled "hadoop@master:~/hive". The terminal displays the results of two HiveQL queries. The first query, `select * from stocks where price_open >= 15000 ;`, returned 85 rows in 0.272 seconds. The second query, `select exchange, symbol, ymd from stocks where price_open >= 500 limit 10 ;`, returned 6 rows in 0.161 seconds. The results are displayed as tables with columns for exchange, symbol, ymd, and various price and volume metrics.

Exchange	Symbol	YMD	Price Open	Price High	Price Low	Price Close	Volume	Price Change
NYSE	RAD	2000-03-31	5679.0	5.69	5.69	5.5	600	5.5
NYSE	TEF	2000-05-26	5597.0	57.5	58.25	55.75	6100	38.19
NYSE	TEF	2000-05-03	5597.02	72.69	73.56	69.75	7300	31.85

Time taken: 0.272 seconds, Fetched: 85 row(s)

hive> select \* from stocks where price\_open >= 15000 ;

OK

Exchange	Symbol	YMD	Price Open	Price High	Price Low	Price Close	Volume	Price Change
NASDAQ	ATCO	2000-03-16	31007.0	9.38	9.38	9.38	900	9.38
NASDAQ	AWRE	2000-06-06	29595.0	51.5	55.0	51.25	5500	51.25
NASDAQ	FSCI	2000-05-18	15713.0	74.5	74.5	70.0	7000	61.79
NASDAQ	IGLD	2000-04-17	33821.0	9.5	11.25	9.5	1100	9.5
NASDAQ	RDCM	2000-03-27	31256.0	16.12	16.63	15.38	400	61.5
NASDAQ	SSTI	2000-04-13	15045.0	74.25	79.12	71.0	7500	23.67

Time taken: 0.161 seconds, Fetched: 6 row(s)

hive>

### 3. Hive 업로드 파일 보기

- HDFS 명령어 이용
- 브라우저 이용

# 1) Hive 업로드 파일 HDFS에서 확인(hive 업로드 파일은 HDFS에 저장)

```
hadoop@master:~/hive
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
[hadoop@master ~]$ cd hive
[hadoop@master hive]$ pwd
/home/hadoop/hive
[hadoop@master hive]$ hdfs dfs -ls /user/hive/warehouse/stocks/NASDAQ*
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 10:49 /user/hive/warehouse/stock
s/NASDAQ_daily_prices_0.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 10:49 /user/hive/warehouse/stock
s/NASDAQ_daily_prices_1.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 10:49 /user/hive/warehouse/stock
s/NASDAQ_daily_prices_2.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 10:49 /user/hive/warehouse/stock
s/NASDAQ_daily_prices_3.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 10:49 /user/hive/warehouse/stock
s/NASDAQ_daily_prices_4.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 10:49 /user/hive/warehouse/stock
s/NASDAQ_daily_prices_5.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 10:49 /user/hive/warehouse/stock
s/NASDAQ_daily_prices_6.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 10:49 /user/hive/warehouse/stock
s/NASDAQ_daily_prices_7.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 10:49 /user/hive/warehouse/stock
s/NASDAQ_daily_prices_8.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 10:49 /user/hive/warehouse/stock
s/NASDAQ_daily_prices_9.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 10:49 /user/hive/warehouse/stock
s/NASDAQ_daily_prices_A.csv

hadoop@master:~/hive

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
$ hdfs dfs -rm /user/hive/warehouse/~
[hadoop@master hive]$ hdfs dfs -ls /user/hive/warehouse/stocks/NYSE*
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 09:55 /user/hive/warehouse
s/NYSE_daily_prices_0.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 09:55 /user/hive/warehouse
s/NYSE_daily_prices_1.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 09:55 /user/hive/warehouse
s/NYSE_daily_prices_2.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 09:55 /user/hive/warehouse
s/NYSE_daily_prices_3.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 09:55 /user/hive/warehouse
s/NYSE_daily_prices_4.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 09:55 /user/hive/warehouse
s/NYSE_daily_prices_5.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 09:55 /user/hive/warehouse
s/NYSE_daily_prices_6.csv
-rwxrwxr-x  3 hadoop supergroup      130 2017-08-17 09:55 /user/hive/warehouse
s/NYSE_daily_prices_7.csv
```

## 2) 브라우저에서 확인

localhost:50070

Browsing HDFS

192.168.13.5:50070/explorer.html#/user/hive/warehouse

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Browse the file system  
Logs

### Browse Directory

/user/hive/warehouse

Hive에서 작성한 table

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Number of blocks
drwxrwxr-x	hadoop	supergroup	0 B	2017. 8. 16. 오후 12:32:06	0	0 B	<a href="#">invites</a>
drwxrwxr-x	hadoop	supergroup	0 B	2017. 8. 16. 오후 12:30:23	0	0 B	<a href="#">pokes</a>
drwxrwxr-x	hadoop	supergroup	0 B	2017. 8. 17. 오전 11:01:05	0	0 B	<a href="#">stocks</a>

Hadoop 2.0.15

- Stocks 테이블에 업로드 된 file

Firefox 웹 브라우저

Browsing HDFS - Mozilla Firefox

Browsing HDFS

192.168.13.5:50070/explorer.html#/user/hive/warehouse/sto

Table에 업로드된 cvs 파일

/user/hive/warehouse/stocks

Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxrwxr-x	hadoop	supergroup	130 B	2017. 8. 17. 오전 10:49:56	3	128 MB	NASDAQ_daily_prices_0.csv
-rwxrwxr-x	hadoop	supergroup	130 B	2017. 8. 17. 오전 10:49:56	3	128 MB	NASDAQ_daily_prices_1.csv
-rwxrwxr-x	hadoop	supergroup	130 B	2017. 8. 17. 오전 10:51:09	3	128 MB	NASDAQ_daily_prices_2.csv
-rwxrwxr-x	hadoop	supergroup	130 B	2017. 8. 17. 오전 10:51:09	3	128 MB	NASDAQ_daily_prices_3.csv

hadoop@master:~/hive

hadoop@master:~/hive

Browsing HDFS - Mozilla Firefox

1 / 4

## 4. Hive 검색 결과 로컬 저장

- HDFS 파일에 저장
- Linux 로컬에 직접 저장

# 1) HDFS 파일에 저장

로컬에 저장할 디렉터리 생성

```
[hadoop@master ~]$  
[hadoop@master ~]$ pwd  
/home/hadoop  
[hadoop@master ~]$ mkdir hive_result
```

HDFS에 검색 결과 저장

하이프QL

```
hive> INSERT OVERWRITE DIRECTORY '/user/hive/warehouse'  
select * from stocks where price_open >= 15000 ;
```

로컬 위치에 파일 저장

```
hadoop@master:~/hadoop-2.7.4  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[hadoop@master hadoop-2.7.4]$  
[hadoop@master hadoop-2.7.4]$  
[hadoop@master hadoop-2.7.4]$  
[hadoop@master hadoop-2.7.4]$ hdfs dfs -ls /user/hive/warehouse  
Found 4 items  
-rwxr-xr-x  3 hadoop supergroup      341 2017-10-20 17:07 /user/hive/warehouse/000000_0  
drwxr-xr-x  - hadoop supergroup       0 2017-10-20 15:37 /user/hive/warehouse/invites  
drwxr-xr-x  - hadoop supergroup       0 2017-10-20 15:35 /user/hive/warehouse/pokes  
drwxr-xr-x  - hadoop supergroup       0 2017-10-20 16:04 /user/hive/warehouse/stocks  
[hadoop@master hadoop-2.7.4]$ hdfs dfs -copyToLocal /user/hive/warehouse/000000_0 ~/hive_result
```

```
hadoop@master:~/hive_result
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
[hadoop@master ~]$ cd hive_result/
[hadoop@master hive_result]$ ls
000000_0
[hadoop@master hive_result]$ vi 000000_0
```

로컬에 저장된 파일 내용

```
hadoop@master:~/hive_result
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
NASDAQ^ARDCM^A2000-03-27^A31256.0^A16.12^A16.63^A15.38^A400^A61.5
NASDAQ^ASSTI^A2000-04-13^A15045.0^A74.25^A79.12^A71.0^A7500^A23.67
NASDAQ^AATCO^A2000-03-16^A31007.0^A9.38^A9.38^A9.38^A900^A9.38
NASDAQ^AAWRE^A2000-06-06^A29595.0^A51.5^A55.0^A51.25^A5500^A51.25
NASDAQ^AFSCI^A2000-05-18^A15713.0^A74.5^A74.5^A70.0^A7000^A61.79
NASDAQ^AIGLD^A2000-04-17^A33821.0^A9.5^A11.25^A9.5^A1100^A9.5
~
~
~
~
~
~
"000000_0" 6L, 341C 1,1 모두
```



## 2) Linux 로컬에 직접 저장

로컬에 저장할 디렉터리 생성

```
hadoop-2.7.4      part-r-000000
hadoop-2.7.4.tar.gz result.txt
[hadoop@master ~]$ mkdir hive_result2
[hadoop@master ~]$
[hadoop@master ~]$
```

로컬에 Hive 검색 결과 저장

하이프QL

```
hive> INSERT OVERWRITE LOCAL DIRECTORY
'/home/hadoop/hive_result2' select * from stocks where
price_open >= 15000 ;
```

로컬에 저장된 Hive 검색 결과  
(2개 파일로 나누어서 저장)

```
[hadoop@master ~]$
[hadoop@master ~]$ cd hive_result2
[hadoop@master hive_result2]$ ls -l
합계 8
-rw-r--r--. 1 hadoop hadoop  0 10월  20 17:35 000000_0
-rw-r--r--. 1 hadoop hadoop 117 10월  20 17:35 000001_0
-rw-r--r--. 1 hadoop hadoop 224 10월  20 17:35 000002_0
-rw-r--r--. 1 hadoop hadoop  0 10월  20 17:35 000003_0
[hadoop@master hive_result2]$ vi 000001_0
[hadoop@master hive_result2]$ vi 000002_0
[hadoop@master hive_result2]$
```

## 5. Hive QL실습

- Hive QL 특징
  1. 메타스토어에 저장된 테이블 대상
  2. Hive에서 사용된 data는 HDFS에 저장
    - ✓ UPDATE, DELETE 사용 불가
    - ✓ INSERT 역시 빈 테이블이나 이미 입력된 데이터를 덮어쓰는 경우만 사용
  3. 서브쿼리는 FROM 절에만 사용
  4. 기타 많은 부분에서 기존 SQL문과 매우 유사함

# 집계함수

- 특정 열을 기준으로 그룹 별로 행의 값들을 요약, 집계해서 하나의 값으로 반환해주는 함수, 주로 GROUP BY와 함께 사용

함수	내용	리턴값
<b>count(*), count(n)</b>	NULL값을 포함한 행의 총 개수 반환	INT
<b>sum(col)</b> <b>sum(DISTINCT col)</b>	그룹 내 요소들의 합을 반환 그룹 내 열의 고유한 값의 합 반환	DOUBLE
<b>avg(col)</b> <b>avg(DISTINCT col)</b>	그룹 내 요소들의 평균을 반환 그룹 내 열의 고유한 값의 평균 반환	DOUBLE
<b>min(col)</b>	그룹 내 열의 최솟값을 반환	DOUBLE
<b>max(col)</b>	그룹 내 열의 최댓값을 반환	DOUBLE
<b>variance(col)</b> <b>var_samp(col)</b>	특정 열의 모집단 분산을 반환 특정 열의 표본 분산을 반환	DOUBLE
<b>stddev_pop(col)</b> <b>stddev_samp(col)</b>	특정 열의 모집단 표준편차를 반환 특정 열의 표본 표준편차를 반환	DOUBLE

# 1) 뉴욕 거래소/나스닥 전체 업종 수 보기 : count, distinct 이용

하이프QL

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

```
hadoop@master:~/hive
2017-08-17 12:57:05,860 Stage-1 map = 58%   reduce = 0%   Cumulative CPU 190.21 sec
2017-08-17 12:57:08,988 Stage-1 map = 63%   reduce = 0%   Cumulative CPU 190.21 sec
2017-08-17 12:57:12,053 Stage-1 map = 66%   reduce = 0%   Cumulative CPU 195.82 sec
2017-08-17 12:57:15,706 Stage-1 map = 70%   reduce = 0%   Cumulative CPU 195.82 sec
2017-08-17 12:57:18,614 Stage-1 map = 80%   reduce = 0%   Cumulative CPU 195.82 sec
2017-08-17 12:57:21,619 Stage-1 map = 81%   reduce = 0%   Cumulative CPU 201.15 sec
2017-08-17 12:57:24,780 Stage-1 map = 83%   reduce = 0%   Cumulative CPU 201.15 sec
2017-08-17 12:57:27,043 Stage-1 map = 92%   reduce = 0%   Cumulative CPU 201.15 sec
2017-08-17 12:57:30,335 Stage-1 map = 100%  reduce = 0%   Cumulative CPU 205.04 sec
2017-08-17 12:57:46,663 Stage-1 map = 100%  reduce = 100% Cumulative CPU 207.11 sec
MapReduce Total cumulative CPU time: 3 minutes 27 seconds 110 msec
Ended Job = job_1502930849413_0008
MapReduce Jobs Launched:
Stage-Stage-1: Map: 4   Reduce: 1   Cumulative CPU: 207.11 sec   HDFS Read: 996682786 HDFS W
rite: 5 SUCCESS
Total MapReduce CPU Time Spent: 3 minutes 27 seconds 110 msec
OK
5910
Time taken: 342.479 seconds, Fetched: 1 row(s)
hive>
```

hadoop@master:~/hive

hadoop@master:~/hive

Welcome to CentOS - Mozilla ...

1 / 4

1

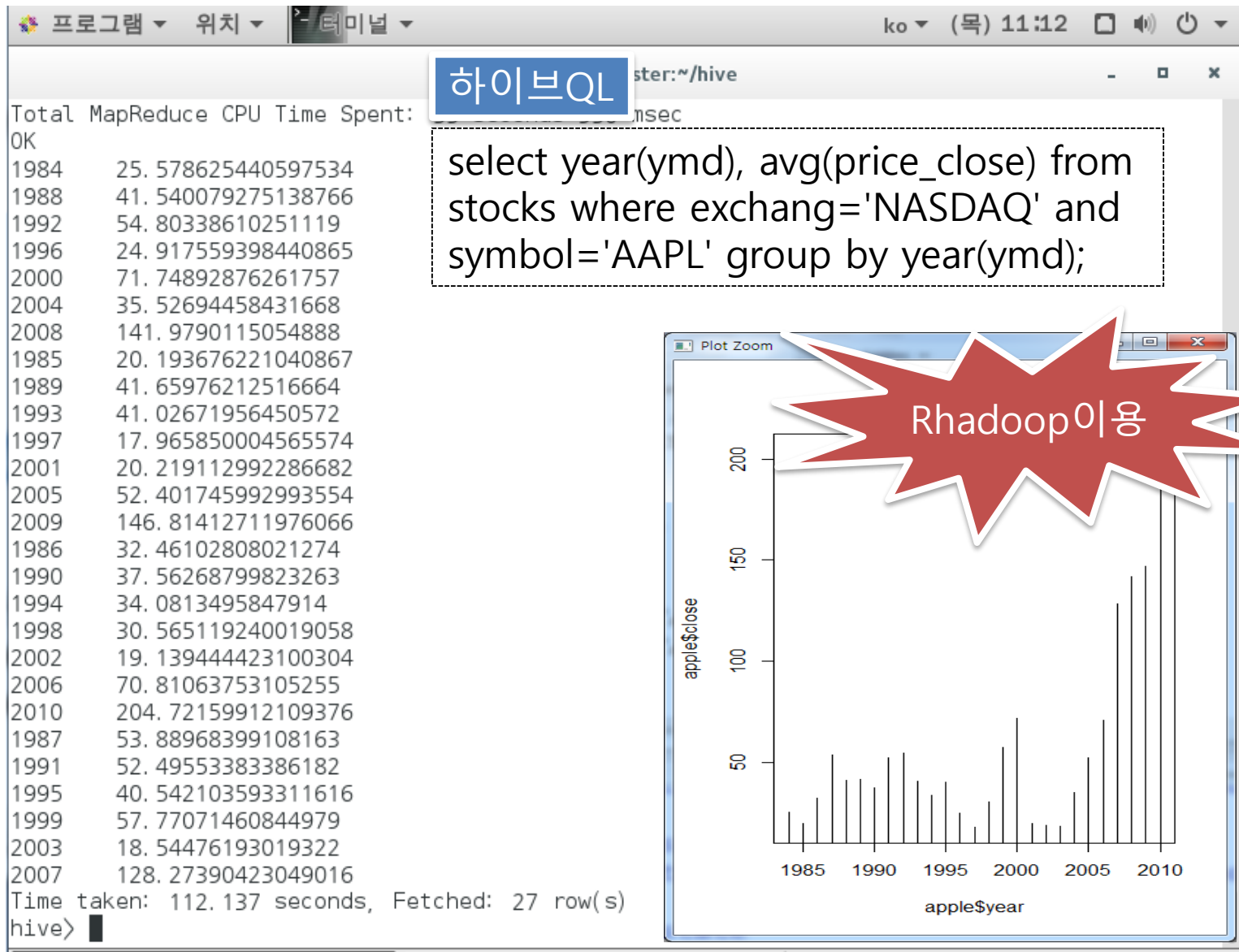
# Group by

- 특정 열을 기준으로 동일한 범주로 그룹화 하는 역할
- 예) 학과 코드를 대상으로 Group by를 적용하여, 동일 학과별 교수들의 평균 급여를 계산한다.

- 형식)

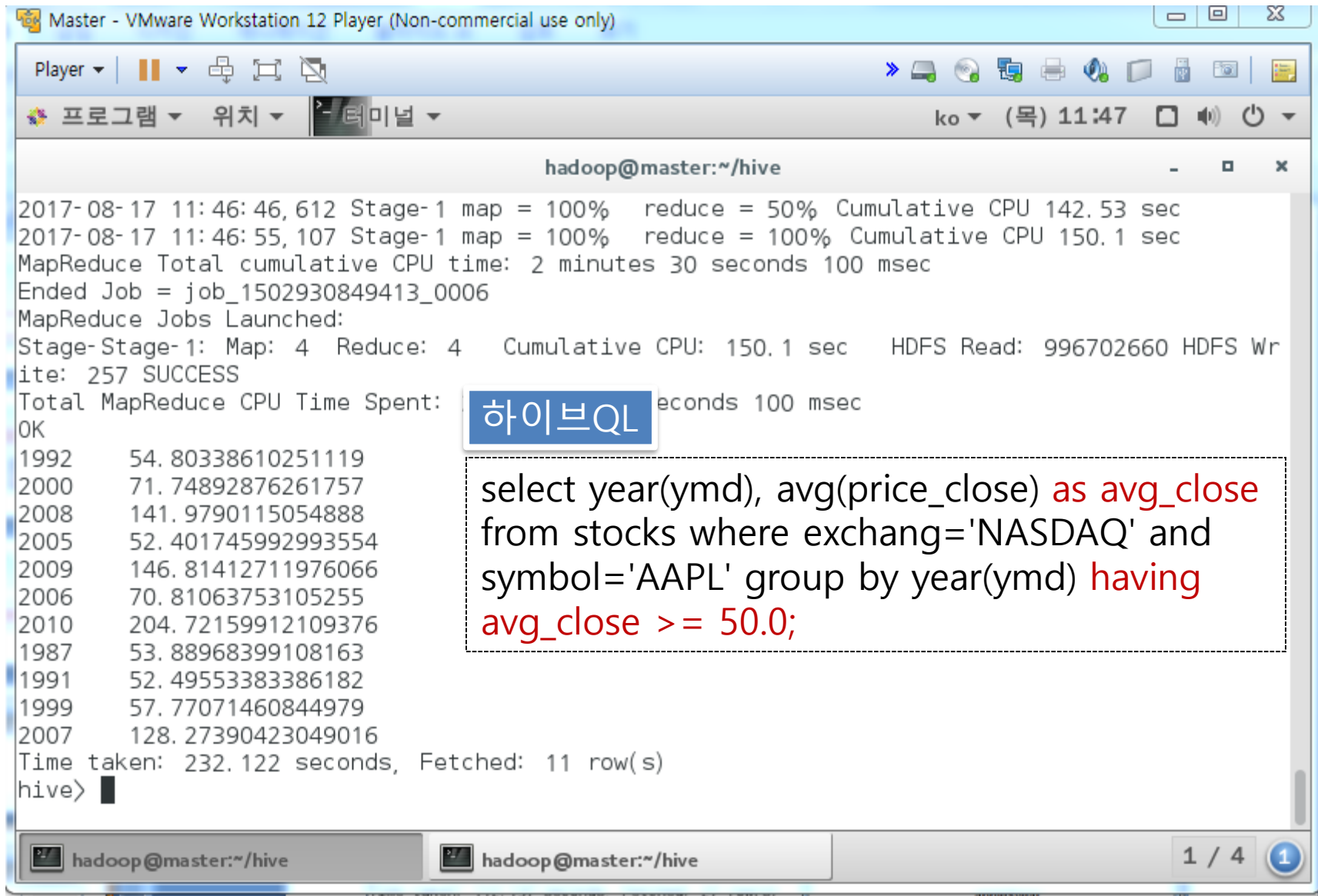
```
select 칼럼명, ... from 테이블명  
Group by 칼럼명 having 조건식;
```

## 2) Apple사의 연도별 종가(Close) 평균 : where, group by 이용



R의 plot()함수 이용

### 3) Apple사의 연도별 종가(Close) 평균 50이상 년도 : as, having절 이용



The screenshot shows a VMware Workstation 12 Player window titled "Master - VMware Workstation 12 Player (Non-commercial use only)". Inside the player, there is a terminal window titled "hadoop@master:~/hive". The terminal displays the output of a Hive job, including stage progress, cumulative CPU time, and a list of years with their average closing prices. A blue box highlights the text "하이프QL" (HiveQL), and a dashed box highlights a SQL query. The query is:   
`select year(ymd), avg(price_close) as avg_close  
from stocks where exchange='NASDAQ' and  
symbol='AAPL' group by year(ymd) having  
avg_close >= 50.0;`

```
hadoop@master:~/hive
2017-08-17 11:46:46,612 Stage-1 map = 100%  reduce = 50% Cumulative CPU 142.53 sec
2017-08-17 11:46:55,107 Stage-1 map = 100%  reduce = 100% Cumulative CPU 150.1 sec
MapReduce Total cumulative CPU time: 2 minutes 30 seconds 100 msec
Ended Job = job_1502930849413_0006
MapReduce Jobs Launched:
Stage-Stage-1: Map: 4  Reduce: 4   Cumulative CPU: 150.1 sec   HDFS Read: 996702660 HDFS Write: 257 SUCCESS
Total MapReduce CPU Time Spent: 150.1 seconds 100 msec
OK
1992      54.80338610251119
2000      71.74892876261757
2008      141.9790115054888
2005      52.401745992993554
2009      146.81412711976066
2006      70.81063753105255
2010      204.72159912109376
1987      53.88968399108163
1991      52.49553383386182
1999      57.77071460844979
2007      128.27390423049016
Time taken: 232.122 seconds, Fetched: 11 row(s)
hive>
```

하이프QL

```
select year(ymd), avg(price_close) as avg_close
from stocks where exchange='NASDAQ' and
symbol='AAPL' group by year(ymd) having
avg_close >= 50.0;
```

hadoop@master:~/hive

1 / 4

# Sub query

- 서브쿼리 : 메인 쿼리에 종속된 쿼리
- 서브 쿼리의 결과를 넘겨 받아서 메인 쿼리가 수행된다.
- FROM 절에는 테이블의 이름이 있어야 하므로 서브쿼리의 결과에 이름 부여
- 반드시 FROM 절에서만 사용

형식)

메인 쿼리      FROM ( 서브쿼리 ) 서브쿼리저장이름

예) SELECT sel\_result.avg

FROM (select avg(pay) as age from pay >= 300) sel\_result



# RAD사의 년도 별 종가(Close) 평균 출력 : subquery 이용

```
프로그램 ▾ 위치 ▾ 터미널 ▾ ko ▾ (목) 14:23 [icon] [icon] [icon]
hadoop@master:~/hive
Stage-Stage-1: Map: 4 Reduce: 4 Cumulative CPU: 132.02 sec HDFS Read: 996701544 HDFS Write: 637 SUCCESS
Total MapReduce CPU Time Spent: 2 minutes 12 seconds 20 msec
OK
1984      18.142999839782714
1988      27.111264843243383
1992      18.003503979660394
1996      30.625748048617144
2000      5.255789474437111
2004      4.672301588550447
2008      1.6306324148837756
1985      19.013254014272537
1989      28.149404745253307
1993      15.355256887292674
1997      48.0291305662615
2001      6.481532269908536
2005      3.9109127039001104
2009      1.0780158725877602
1986      22.031383408859313
1990      26.393992050834324
1994      18.095634850244675
1998      38.70019841572595
2002      2.6462549788068492
2006      4.340836656046104
2010      1.4623999977111817
1987      27.07154156379549
1991      25.670276604151066
1995      24.29777785709926
1999      23.318769852320354
2003      4.233253979493702
2007      5.337689258187891
Time taken: 271.277 seconds, Fetched: 27 row(s)
hive> █
```

하이프QL

```
SELECT sel2.year, sel2.avg FROM (select year(ymd)
as year, avg(price_close) as avg from stocks where
symbol='RAD' group by year(ymd) ) sel2 ;
```

# Join

- 테이블이 파티션으로 분할되어 저장된 경우 파티션과 관련된 테이블의 일부만 검색할 경우 HOIN ON절을 이용한다.

```
SELECT join_t1.* join_t2
FROM join_t1 JOIN join_t2
ON (
    join_t1.id = join_t2.id
    AND join_t1.date >='2017-01-01'
    AND join_t1.date <='2017-12-30'
);
```

- 산술연산자

연산자	설명
$A+B$	A와 B를 더한다
$A-B$	A에서 B를 뺀다
$A*B$	A와 B를 곱한다
$A/B$	A를 B로 나눈다. 만약 피연산자가 정수형이면 나누기를 몫만 반환된다.
$A\%B$	A를 B로 나눈 나머지를 반환한다.
$A\&B$	A와 B를 비트 AND 연산한다.
$A B$	A와 B를 비트 OR 연산한다.
$A^{\wedge}B$	A와 B를 비트 XOR 연산한다.
$\sim A$	A의 비트 NOT 연산한다.

## ● 수학 함수

시그니처	설명
Round(d)	DOUBLE 데이터형 D의 반올림값을 BIGINT로 반환
Round(d,N)	DOUBLE 데이터형 D의 N개 소수점 이하의 값까지 반올림값을 DOUBLE 데이터 유형으로 반환
Floor(d)	DOUBLE 데이터형 d보다 작거나 같은 값 중 가장 큰 BIGINT 를 반환한다
Ceil(d),Ceiling(DOUBLE d)	D보다 크거나 같은 값 중 가장 작은 BIGINT를 반환한다.
Rand(), Rand(seed)	각 로우마다 난수값이 바뀌면서 반환된다. 인자로 받은 정수 데이터형의 씨드는 반환되는 값을 결정한다.
Pow(d,p), Power(d,p)	D의 P 거듭제곱한 값을 반환한다. D와 P는 DOUBLE 데이터형이다.
Sqrt(d)	DOUBLE 데이터형 D의 제곱근을 반환한다.
Abs(d)	DOUBLE 데이터형 D의 절대값을 DOUBLE 데이터형으로 반환한다.
Sin(d)	DOUBLE 데이터형 D의 사인값을 DOUBLE 데이터형의 라디안 값으로 반환한다.
Cos(d)	DOUBLE 데이터형 D의 코사인값을 DOUBLE 데이터형의 라디안 값으로 반환한다.
Tan(d)	DOUBLE 데이터형 D의 탄젠트값을 DOUBLE 데이터형의 라디안 값으로 반환한다.
Pi()	파이 상수값을 DOUBLE 데이터형으로 반환한다.

## ● 기타 함수

시그니처	설명
<code>length(s)</code>	문자열의 길이를 구한다.
<code>concat(s1, s2, ...), concat(sep, s2, s2)</code>	문자열을 연결한다. <code>sep</code> 는 구분자.
<code>substring(s, start), substring(s, start, len)</code>	문자열의 일부를 구한다.
<code>upper(s), ucase(s), lower(s), lcase(s)</code>	대문자/소문자로 변환한다.
<code>trim(s), ltrim(s), rtrim(s)</code>	끝의 공백을 제거한다.
<code>regexp_replace(s, regex, replacement)</code>	정규표현식과 일치하는 부분을 대체한다.
<code>regexp_extract(s, regex, idx)</code>	정규표현식과 일치하는 <code>idx</code> 번째 부분을 구한다.
<code>parse_url(url, partname, key)</code>	URL에서 일부를 구한다.
<code>from_unixtime(int unixtime))</code>	유닉스 기준시로부터 타임스탬프(yyyy-MM-dd HH:mm:ss)를 구한다.
<code>to_date(ts타임스탬프)</code>	타임스탬프에서 'yyyy-MM-dd' 부분을 구한다.
<code>year(ts), month(ts), day(ts)</code>	타임스탬프에서 연, 월, 일 부분을 구한다.