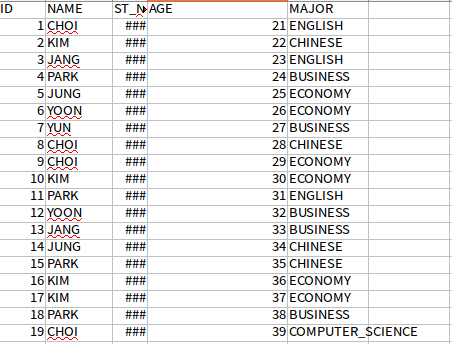
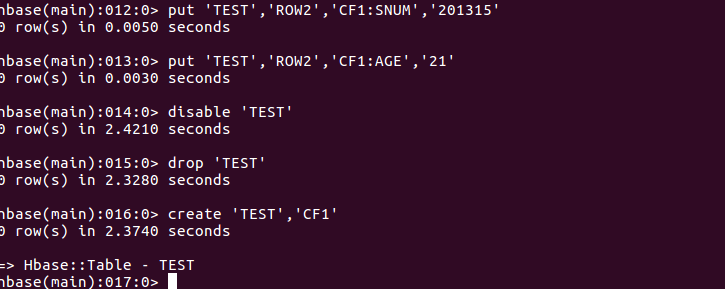
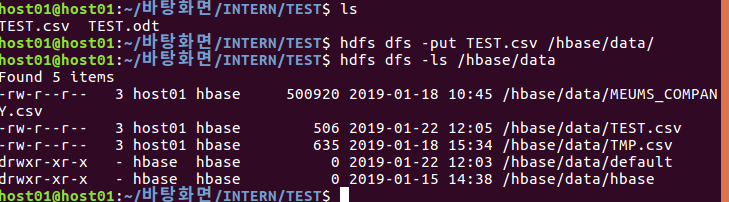
***.CSV → HBASE → HIVE\_EXT → HIVE***

1.CSV → HBASE



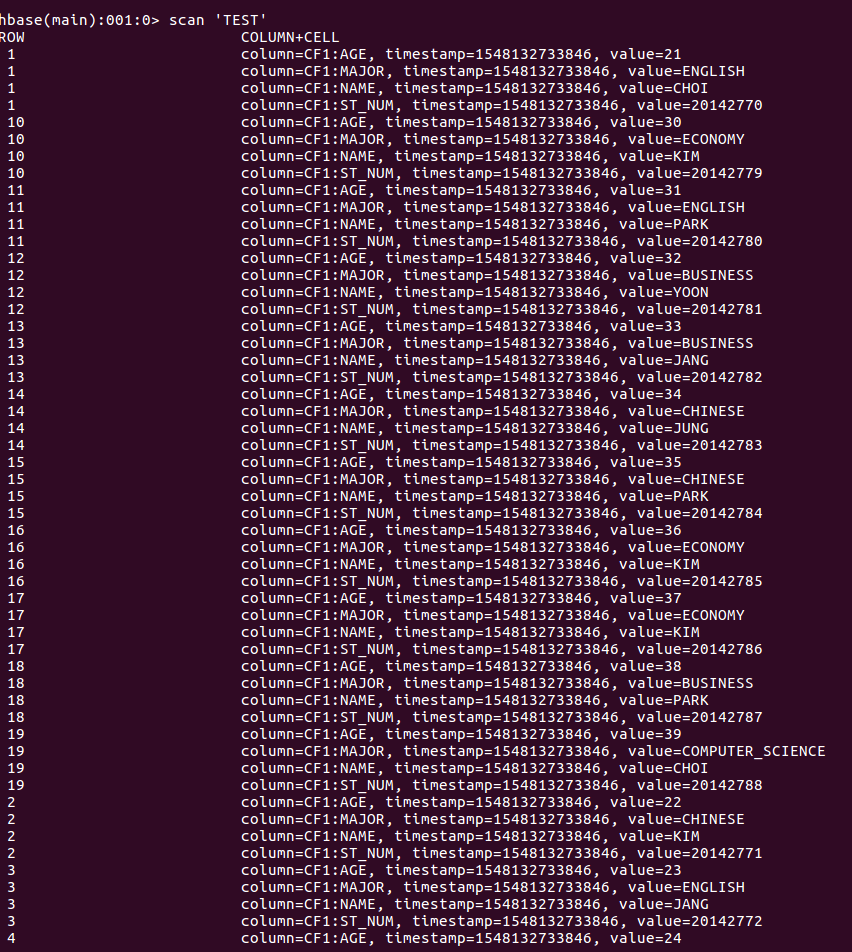
(‘TEST.csv’ 파일을 만들었다.)

 (HBASE에 PUT METHOD를 통해서 넣을 수 있지만, CSV FILE을 넣는 과정을 위해 미리 테이블과 COLUMN FAMILY만 생성)



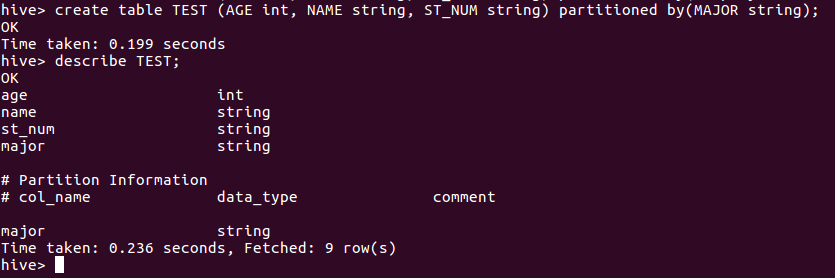
(LOCAL에 만든 .CSV FILE을 HBASE/DATA에 PUT 메서드를 통해 HDFS에 넣는다)





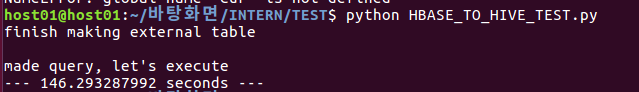
(HBASE의 IMPORTTSV 메서드를 통해서 .CSV FILE을 HBASE에 넣었다.이 때 앞의 ID부분을 HBASE\_ROW\_KEY로 주었기 때문에 ID 필드는 HBASE로 넣지 않았다.)

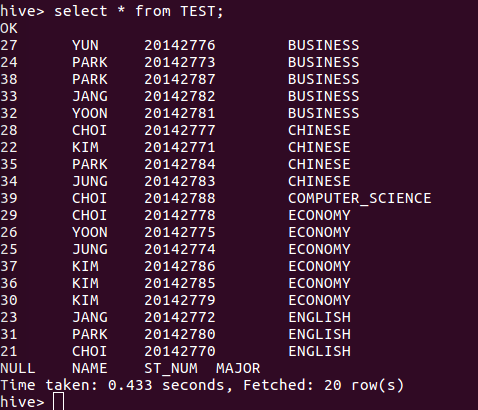
2. HBASE → HIVE

 (최종적인 데이터 마트를 구성하기 위해서 먼저 MAJOR COLUMN을 기준으로 PARTITION하는 테이블을 만든다)

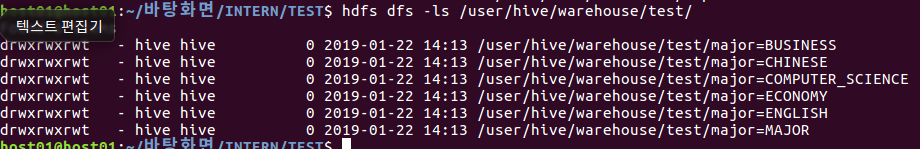
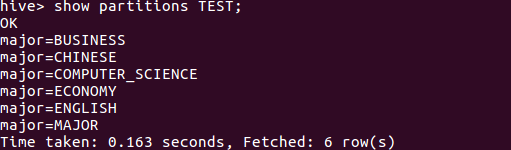


(HBASE에서 TEST 테이블을 읽어서 PYHIVE를 이용해 HIVE 데이터마트 테이블로 이관한다. )

 (TABLE MIGRATION 코드 실행 결과)

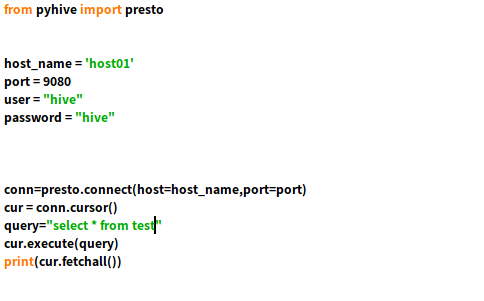


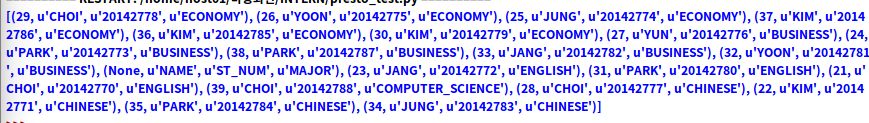
(코드 실행 결과 HIVE에서 DATA MIGRAITON이 완료 된 것을 알 수 있다.)

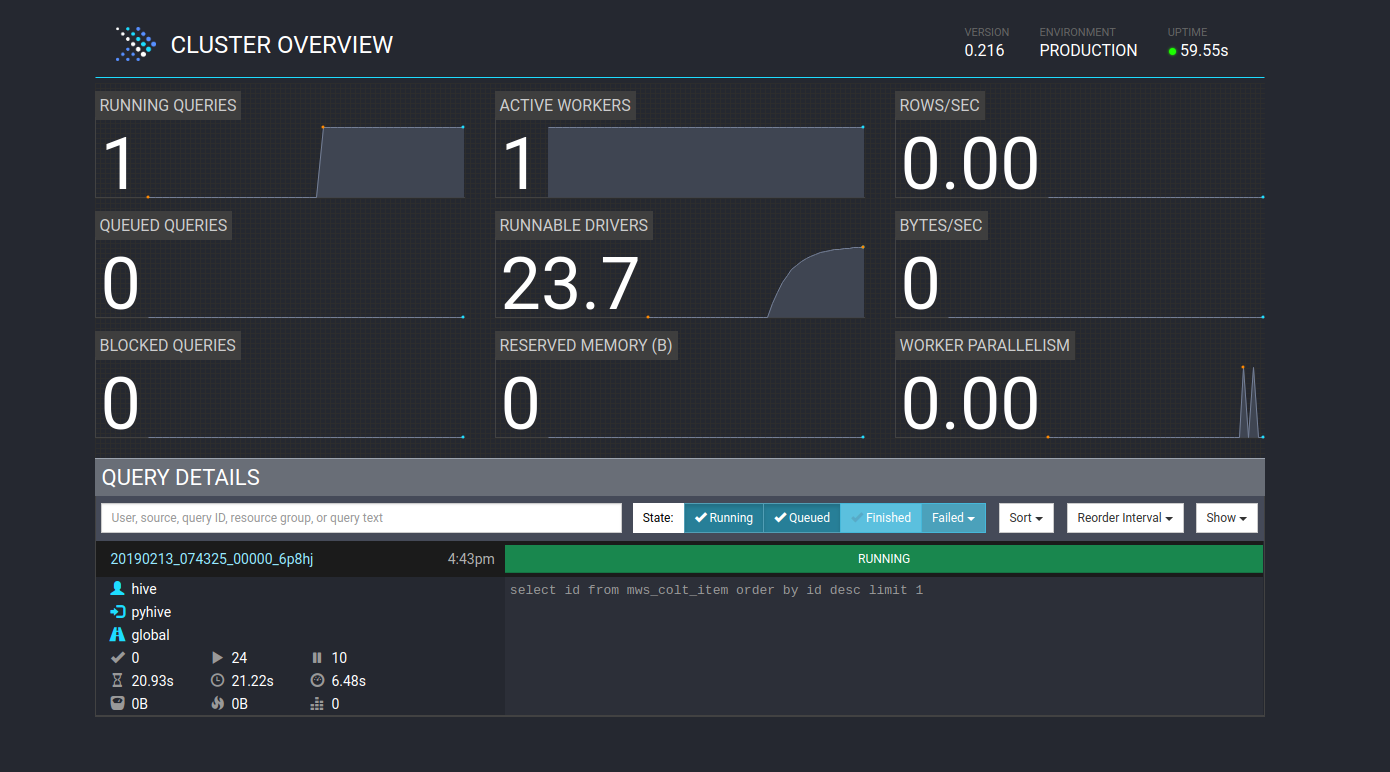
(또한 위의 PARTITION COLUMN으로 MAJOR를 지정한 결과 PARTITION에 따라 MAJOR별로 HDFS에 저장되어 있다.)

**3. HIVE → PRESTO**

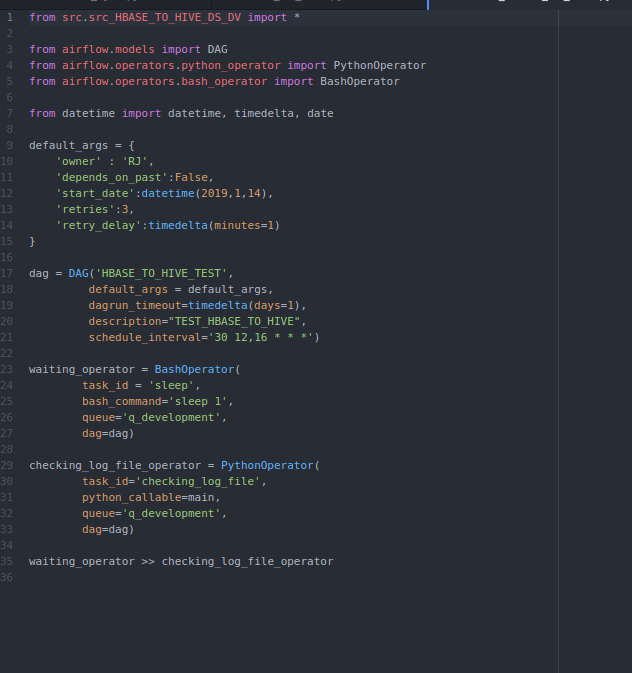
****

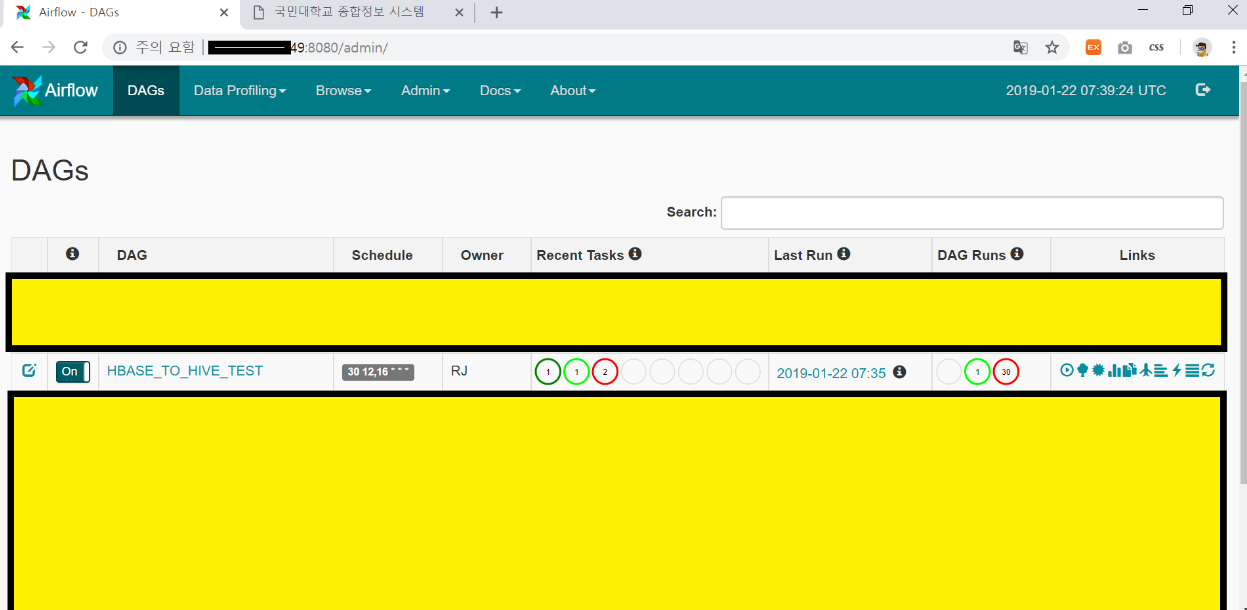


(HOST01을 COORDINATOR로 하고 나머지 3대를 WORKER로 구성해놓은 PRESTO에 접근하기 위해서 PYHIVE 中 PRESTO 기능을 사용하여 HIVE에 연결하여 테이블 내용 출력한다.)



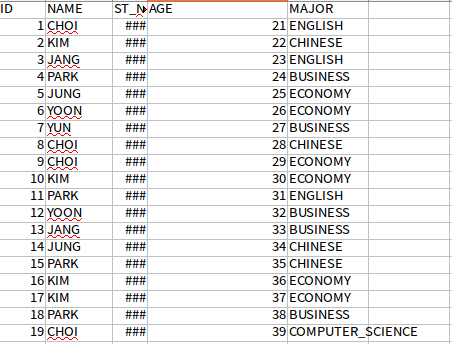
**4. PRESTO → AIRFLOW**



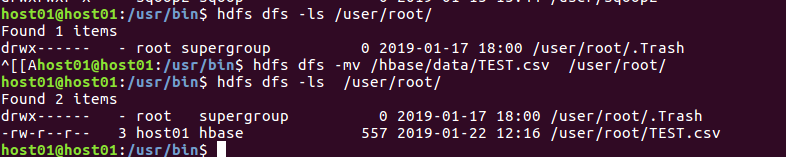


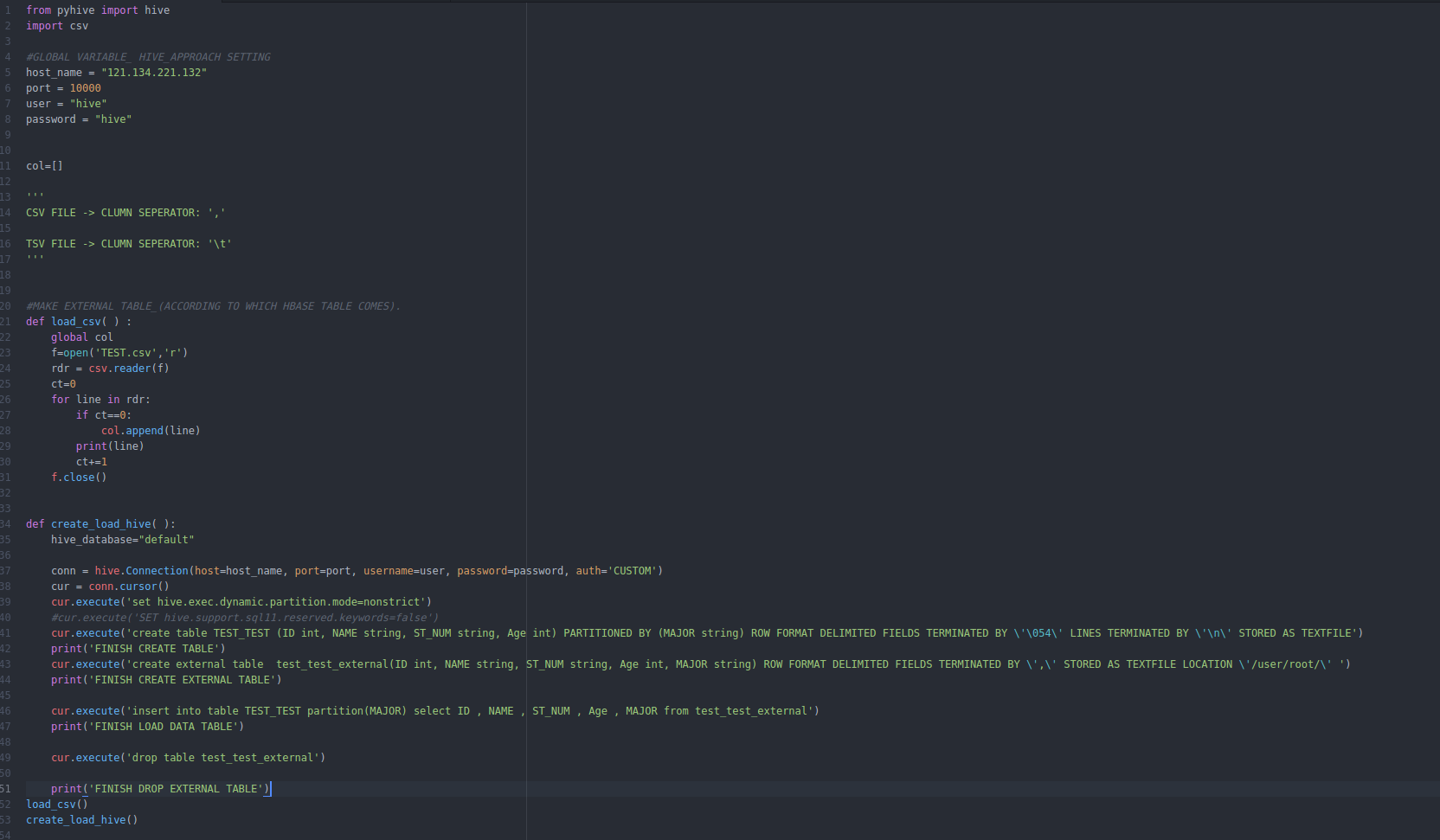
***.CSV → HIVE***

1.CSV → HIVE



(‘TEST.csv’ 파일을 만들었다.)

 (CSV FILE을 HDFS /USER/ROOT/로 옮김 이와 동시에 미리 HIVE TABLE을 만들기 위해 LOCAL에도 넣어놓은 상태다.)



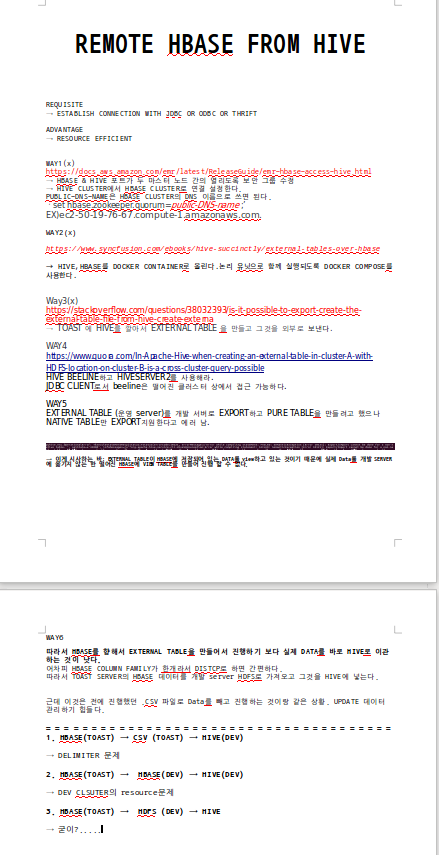
→ PARTITION TABLE에 대해서는 CSV FILE을 LOAD할 때 PARTITON COLUMN을 지정할 수 없어서 EXTERNAL을 통해서 PARTITION COLUMN을 지정해야 한다.

( 위 코드는 CSV FILE이 있는 경로에 있는 파일들을 참조하는 EXTERNAL TABLE을 이용해서 PURE TABLE에 DATA INSERT한다)

 (다 옮겨진 후의 TEST\_TEST PURE TABLE에 CSV FILE이 모두 옮겨짐)

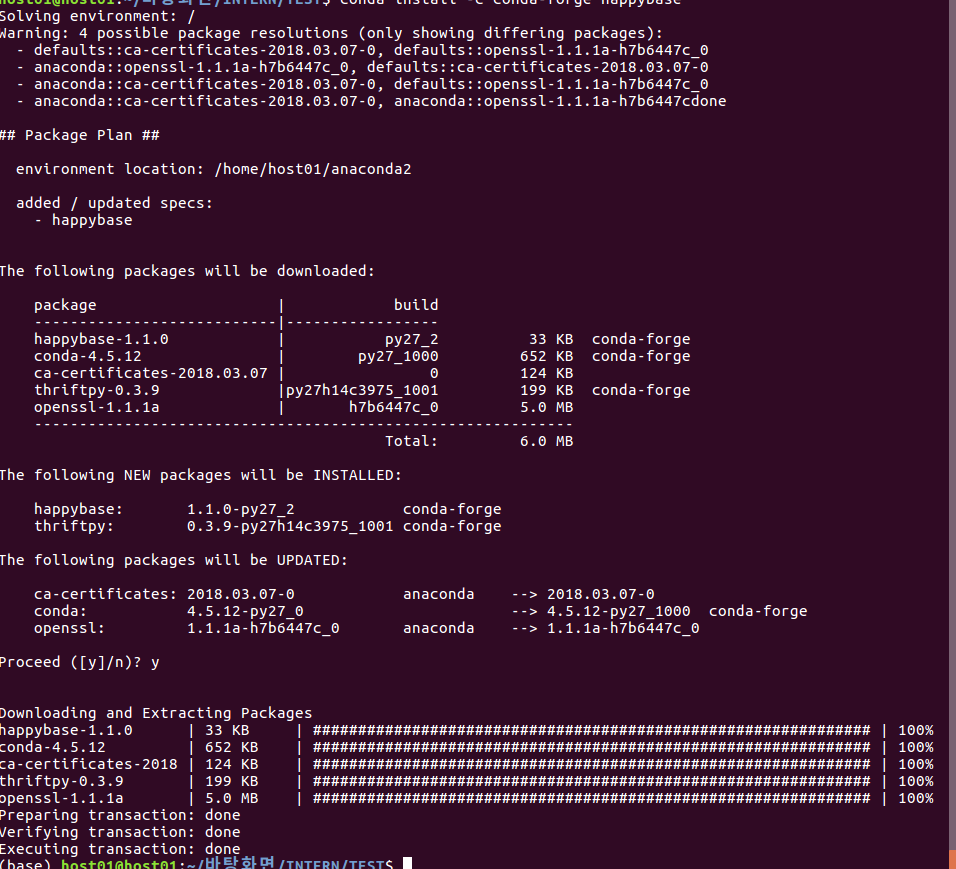
***REMOTE HBASE → CSV → HIVE-SERVE(CSV\_TO\_HIVE.PY)***

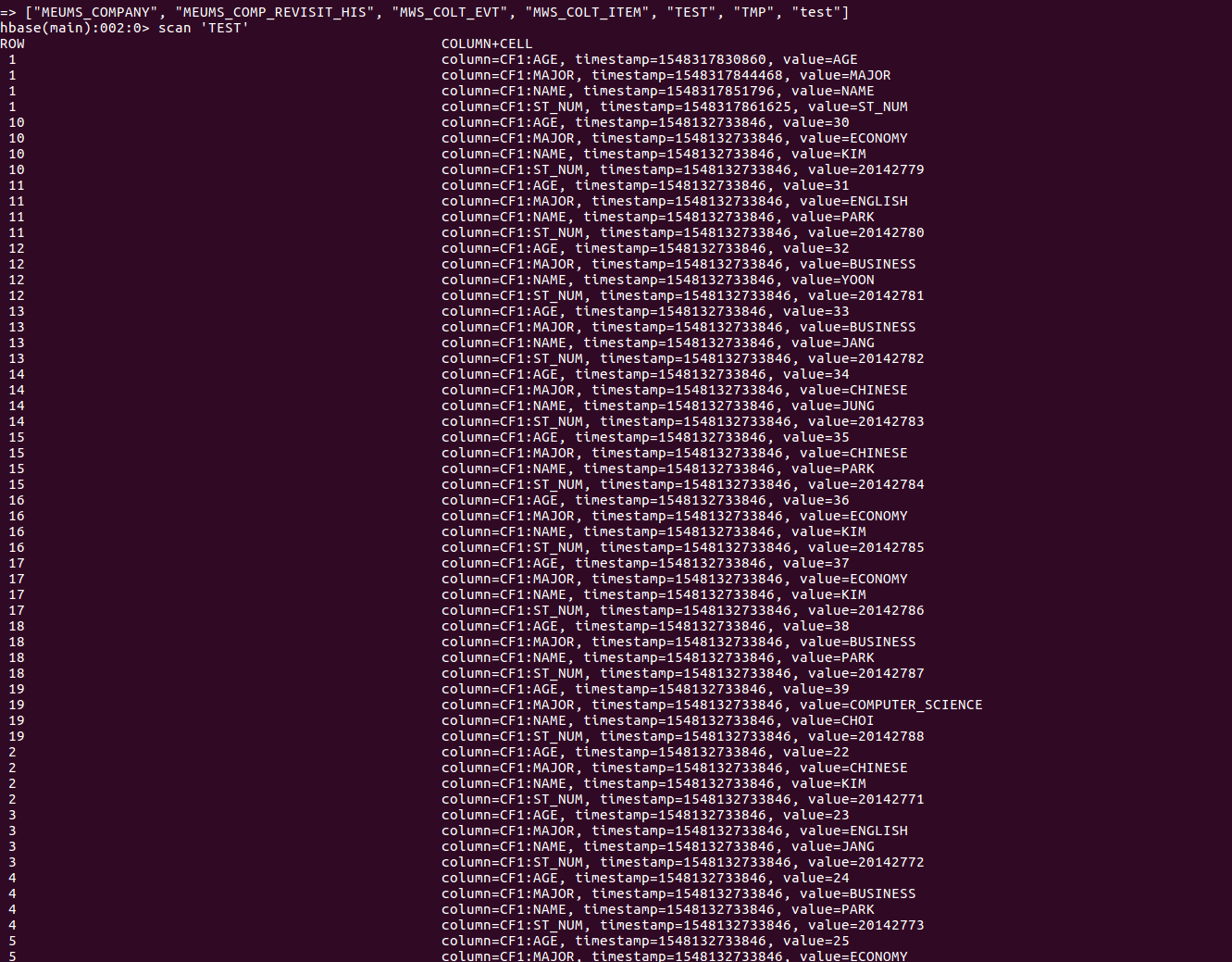
**(최종적으로 HIVE 에서 REMOTE HBASE로의 접근이 어려워서 CSV FILE을 통해서 옮긴다)**



REFFERENCE: https://happybase.readthedocs.io/en/latest/user.html

**HAPPYBASE를 사용한다. HIVE 접근 시는 PYHIVE 사용한다.**



 (INSTALL HAPPYBASE WITH ANACONDA)

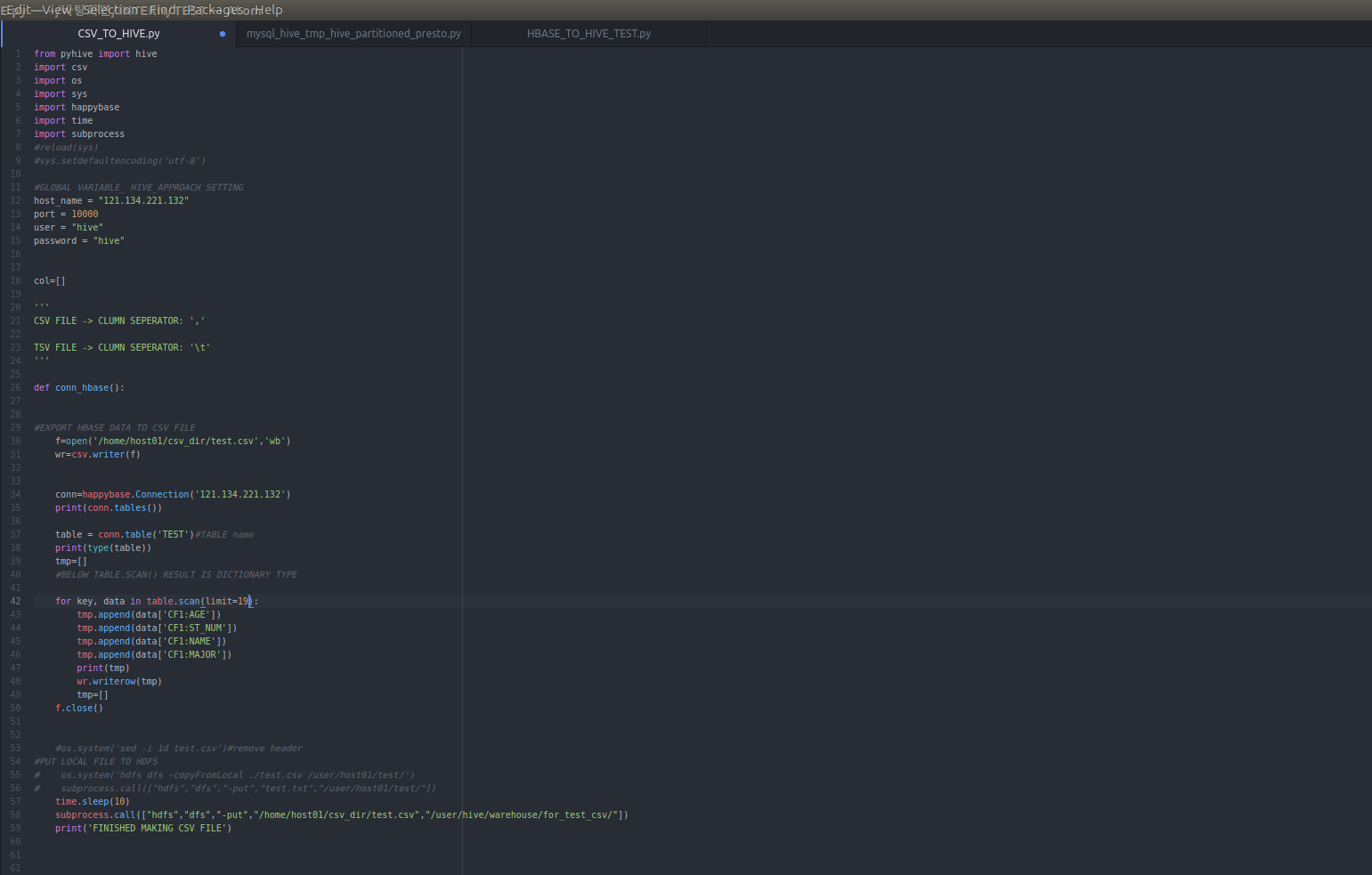
(HBASE에 저장된 TEST TABLE)

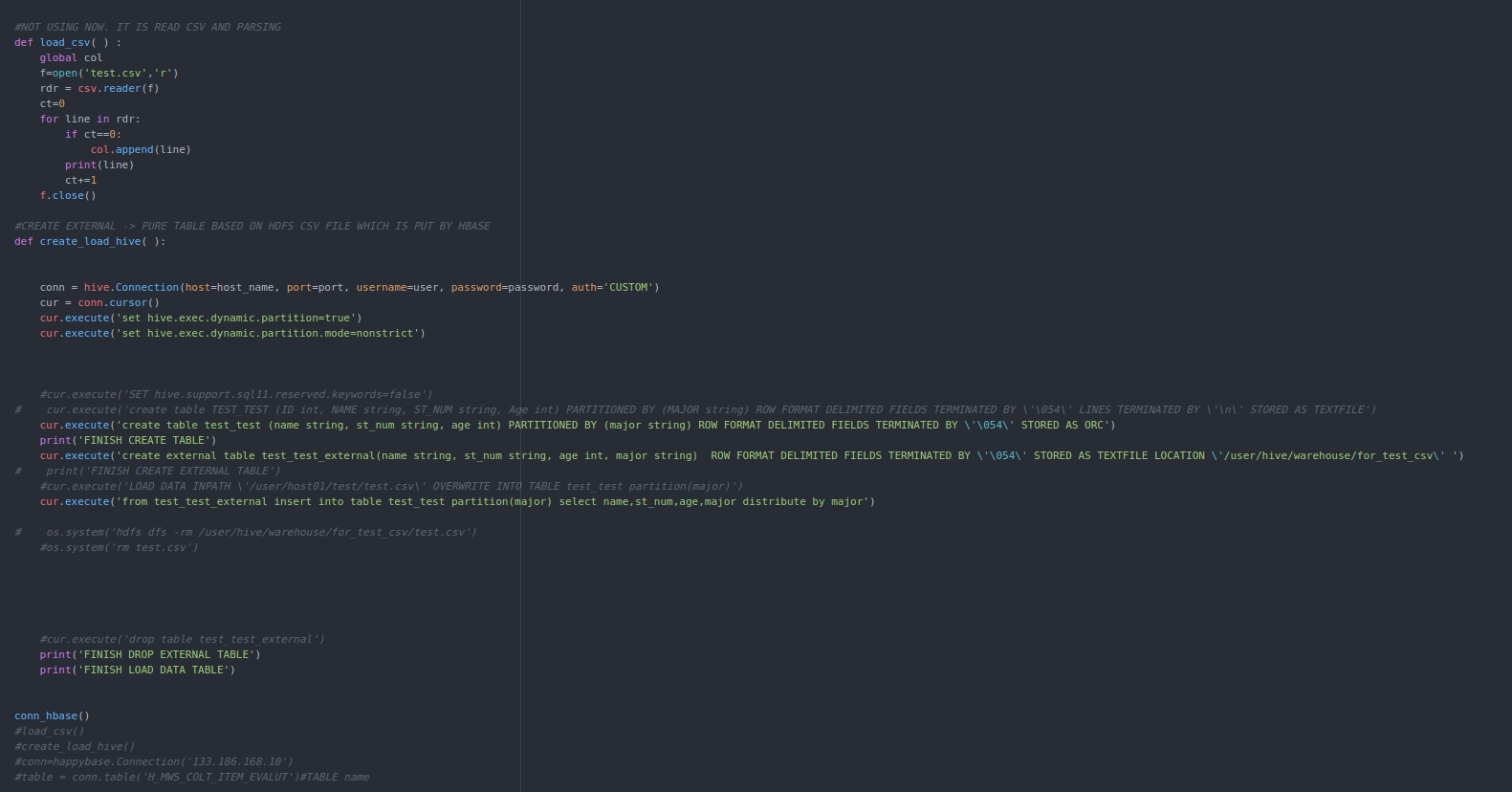
**IUUSE → HIVE PARTITIONIGN을 어떻게 할 것인가? CSV 에서 읽었을 떄 PARTITIONING을 바로 CSV에서 읽은 채로 넣는 방법은 어렵다. 그래서 PARTITIONING에 대해서 찾아봄.**

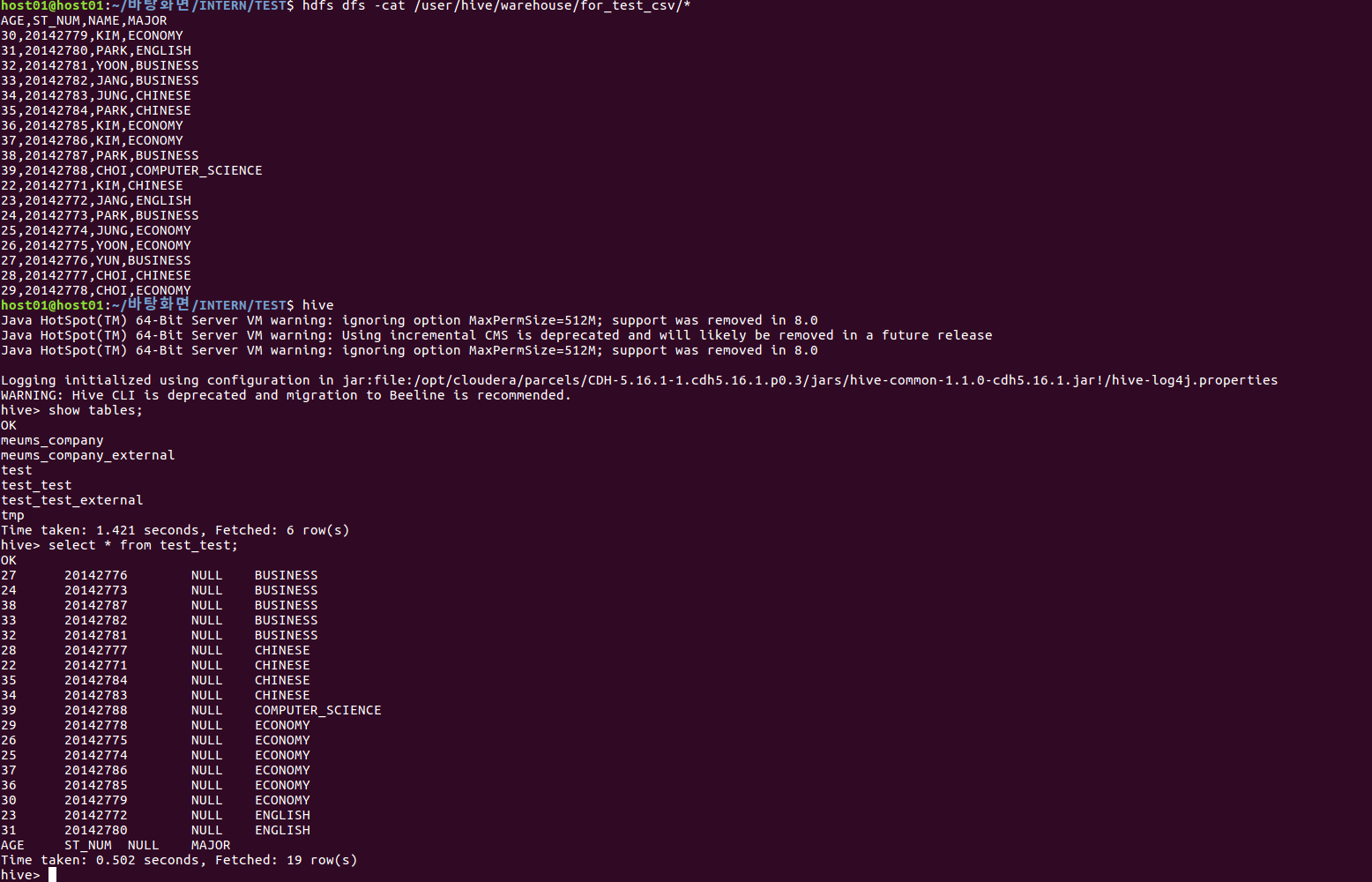
**https://blog.eduonix.com/bigdata-and-hadoop/learn-use-partitioning-hive-improve-query-performance/**

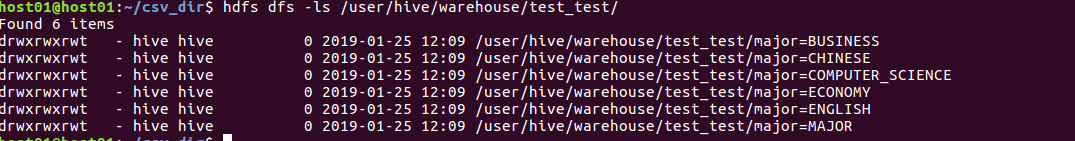
즉, 학과로 PARTITIONING 할 때, 컴공만 모아서 넣거나 기계과만 모아서 넣으면 STATIC PARTITIONING.

DYNAMIC PARTITIONING은 그냥MAJOR를 넣으면 된다.

 **(**HAPPYBASE로 HBASE에 연결 후, SCAN한 결과를 CSV에 작성하고 파일을 HDFS로 옮긴다,)

 (HDFS에 있는 CSV 을 토대로 HIVE EXTERNAL TABLE을 만들고 PURE TABLE (DATA MART) 에 전송한다.)

 **(**최종 결과물 HIVE TABLE에서 확인)

 (PARTITIONING 한 결과 출력)

***CONFIGURATION ERROR.***

==> PYTHON DEPENDENY ISSUE 발생해서 CONDA , PIP과 관련된 라이브러리를 지우고 다시 설치.

CONDA 로 PYHIVE 깔고 CONDA INSTALL -C CONDA-FORGE HAPPYBASE를 두 번 한다.

*HIVE*

→ CSV에서 읽어서 External 로 각 컬럼을 제대로 MAPPING하고 PARTITION이런것은 INSERT INTO할 때 적용한다.

INSERT INTO할 때 마지막 컬럼에 PARTITION COLUMN을 지정한다.

최종적으로 HIVE TABLE 을 CSV 파일로 뽑아서 원본과 비교해보자.

https://medium.com/@gchandra/best-way-to-export-hive-table-to-csv-file-326063f0f229

정확함.

REMOTE HBASE → HBASE → HIVE / REMOT HBASE → CSV → HIVE-SERVE



***따라서 CSV 파일 이용하는것이 효율적이다.***

***DATA MIGRATION***

TEST.odt에 저장한 내용 처럼 HBASE → CSV → HIVE PROCESS로 데이터 이관 작업 실행.

그러던 과정에서 HIVE FILE을 CSV로 EXPORT하던 중 HOME DIRECTORY에 문제 생김.

따라서 재설치.HIVE **PARITION 개수 MAX 설정** 후 진행하는데 HDFS 권한 설정 문제가 있음

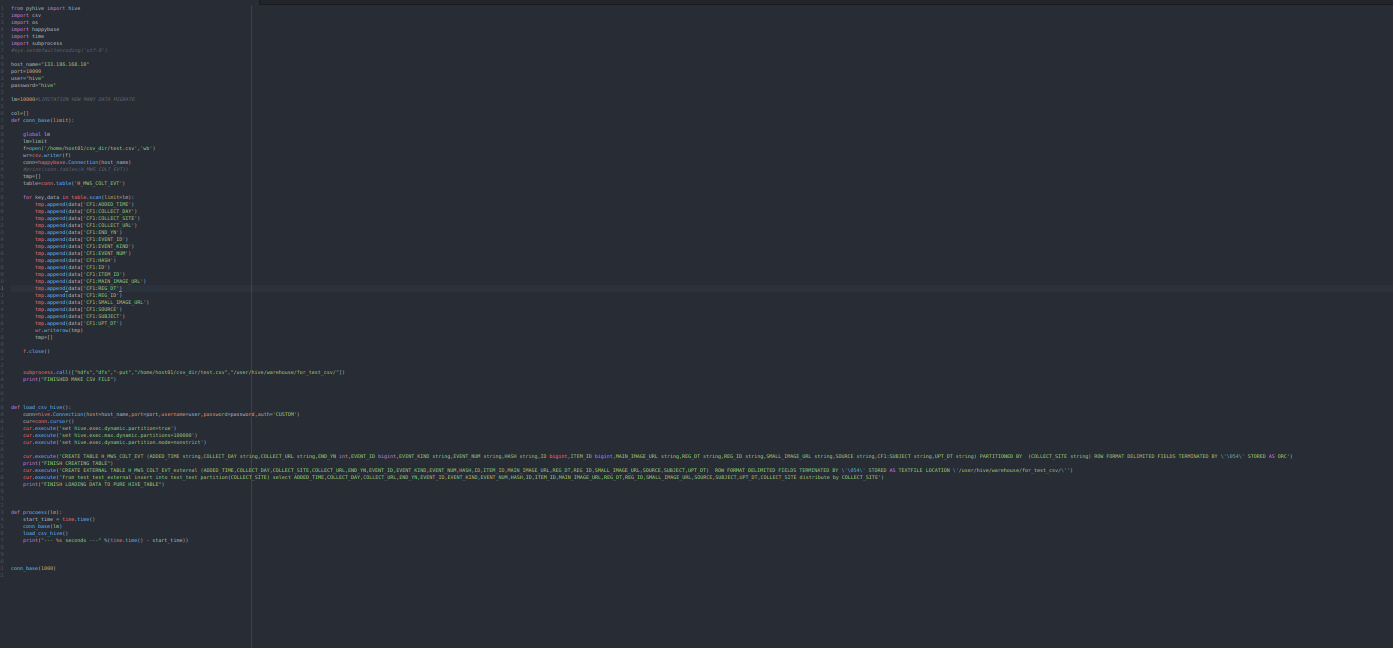
따라서<https://community.hortonworks.com/questions/10508/mkdir-permission-denied-userroot->

[accesswrite-inode.html](https://community.hortonworks.com/questions/10508/mkdir-permission-denied-userroot-accesswrite-inode.html) 따라 hdfs-user에 의해서 소유된 directory에 대해서 권한을 준다.

‘export HADOOP\_USER\_NAME=hdfs’ 명령을 입력.

또한 **PYHIVE와 HAPPYBASE는 CONDA**를 통해서 CONFIGURATION하였다.

HMWS\_EVT 테이블을 읽어서 CSV 파일로 만들고 HDFS에 저장 후 HIVE\_DATA\_MART로 이관하는 코드



***DEV SERVER DATA와 HBASE DATA 원본 INTEGRITY 비교***

 (HIVE TABLE을 CSV FILE로 추출 후 원본 과 비교를 위해서 CSV FILE로 추출)

→ 이 과정에서 EXPORT를 HOME에 하는 것 때문에 HOME에 문제 생김. 따라서 다시 재설치 후

하둡 권한 설정.

<https://community.hortonworks.com/questions/10508/mkdir-permission-denied-userroot->

[accesswrite-inode.html](https://community.hortonworks.com/questions/10508/mkdir-permission-denied-userroot-accesswrite-inode.html)

→ HIVE\_PARTITION개수 설정.

cur.execute('set hive.exec.dynamic.partition=true')

cur.execute('set hive.exec.max.dynamic.partitions=100000')

cur.execute('set hive.exec.dynamic.partition.mode=nonstrict')

→ HAPPYBASE 및 PYHIE설치 : CONDA를 이용해 Python3.7설치 후 CONDA INSTALL로 HAPPYBASE, PYHIVE설치.

***==> 이제 다시 코드 진행 후 원본 DATA 및 HIVE\_DATA\_MART 비교를 해야 한다.***

**CODE 진행 중 문제 발생. HAPPYBASE로 HBASE에 접근 후 CSV FILE로 EXPORT하는 과정에서 한글 깨짐 문제.**

****

****

→ 확인 결과 문제 생기는 COLUMN에 대해서는 WINDOW-1252 인코딩이었고 나머지는 이상이 없음.

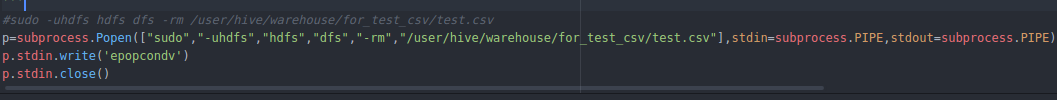
피닉스 HBASE → PURE HBASE 이관 과정에서 발생한 문제 인지 확인해야 함.

그 전에 다른 TABLE가지고 진행 해봄. H\_~로 시작하는 Hbase table에서.

**⇒ 성현님이 해결해 주심. 운영 피닉스 Hbase에서 DEV\_HBASE로 DATA이관 작업 중 ENCODING문제가 있었음.**

**--HIVE쿼리 PARTITION--**

[*https://abc2080.tistory.com/entry/%ED%95%98%EC%9D%B4%EB%B8%8C%EC%9D%98-%EB%82%B4%EC%9E%A5-%ED%95%A8%EC%88%98*](https://abc2080.tistory.com/entry/%ED%95%98%EC%9D%B4%EB%B8%8C%EC%9D%98-%EB%82%B4%EC%9E%A5-%ED%95%A8%EC%88%98)

**

(SUBPROCESS를 이용해서 하둡 명령어 진행 IMPORT SUBPROCESS와 PIPE,STDIN을 통해서 비밀번호 입력까지 할 수 있다)

**--H\_~ TABLE CODE작성--**

***하지만 HBASE ROW를 COUNT해서 몇 개까지 가져올 지 정해야 한다.***

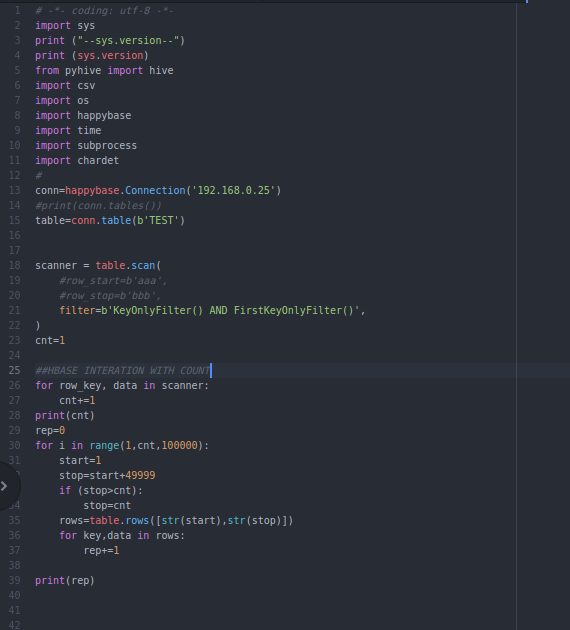
CNT=1

FOR KEY, VALUE IN TABLE.SCAN( ):

CNT +=1

PRINT(CNT)

로 HBASE TABLE개수를 파악하려고 했으나 에러 발생.



(HBASE 전체 ROW\_ COUNT)

나눠서 가져와야 함

→ **즉, 한 번에 전체 접근하면 안됨.**

→ CSV FILE로 1,000,000 ROW를 HBASE에 넣고 COUNT/SPLIT해서 넣는건 TEST

!!!HBASE ROW KEY가 GENERAL NUMERIC RANGE 로 가져와지지 않는다.

**ROWKEY-> (ID%10)\_(ID)\_TST\_WS**

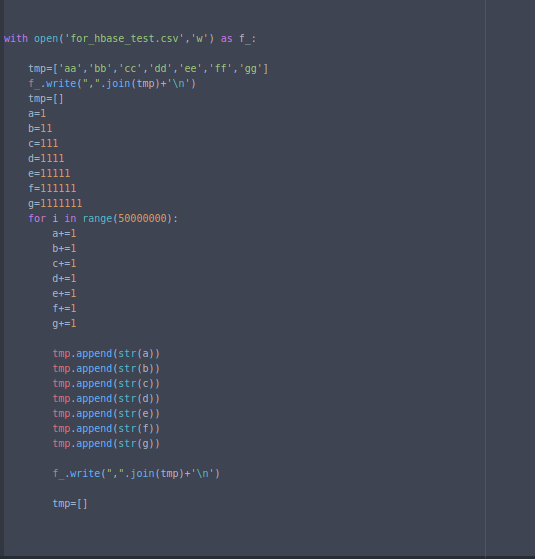
**먼저 HBASE의 ROW\_KEY구조를 알아야 한다.**

**근데 이것도 어려움…**

**따라서 TIME STAMP RANGE이용→** 특정한 시각을 나타내는 문자열

TIMESTAMP -> 이것도 잘 안먹음.........

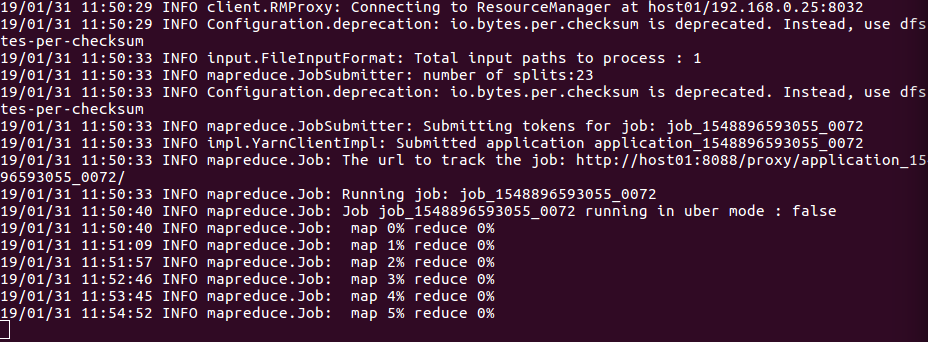
**HAPPYBASE에서 TABLE SCAN에 BATCH\_SIZE 를 줘서 진행해보자**



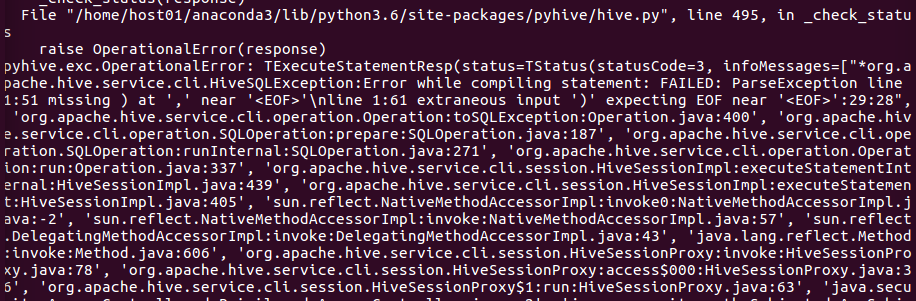
(5000만 ROW의 CSV 생성)



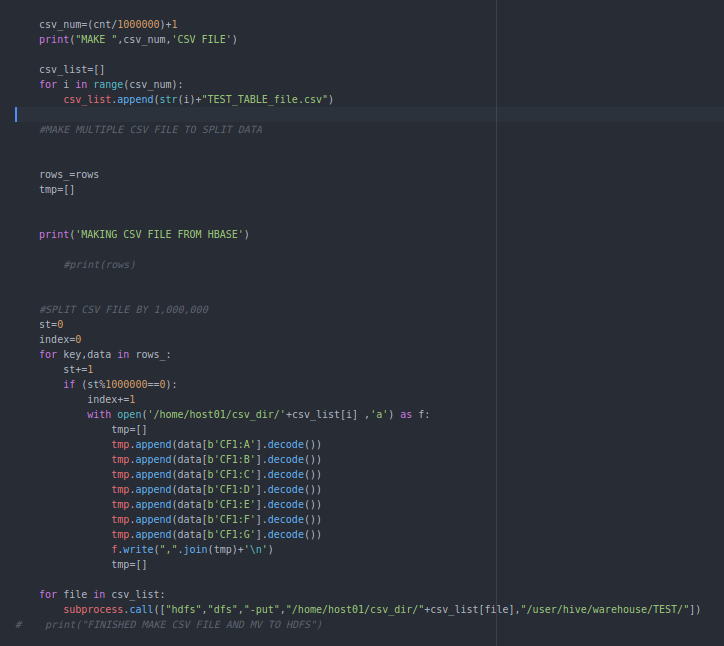
(HDFS 의 /HBASE/DATA/TEST.CSV 로 옮긴다. HBASE로의 이관 목적)



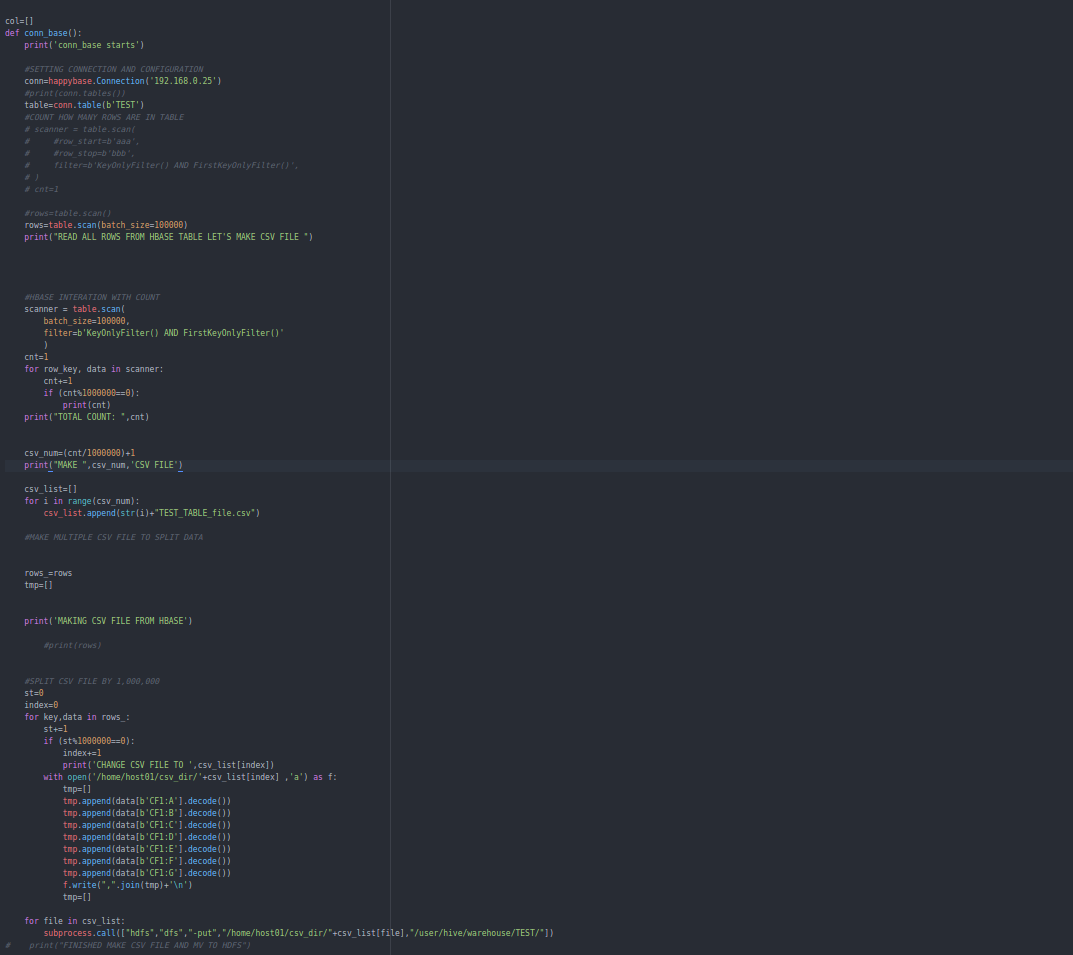
(이 후, HBASE ‘TEST’ TABLE에 이관)



(HBASE에서 읽어서 CSV FILE 만드는 과정에 BUFFERED ERROR. CSV FILE분리 필요)



(전체 길이를 구해서 100000개 씩 넣는 것을 가정해서 CSV FILE 분리하는 것을 생성)



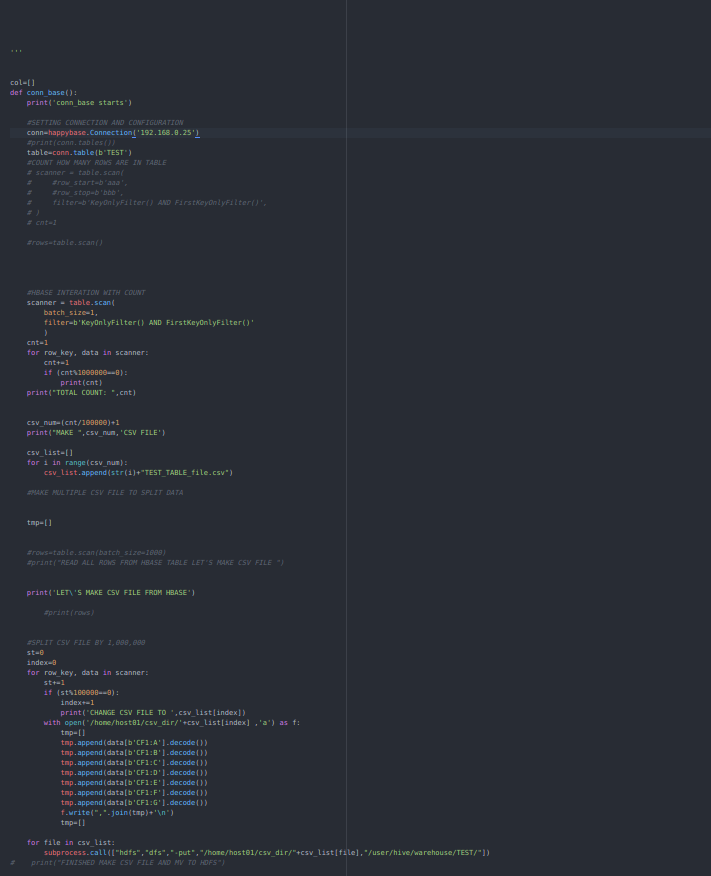
(HAPPYBASE 에서 ROWS.TABLE.SCAN(BATCH\_SIZE=10000))을 통해 제대로 동작하나 확인 및 최종

HIVE\_PURE TABLE까지의 MIGRATION 성공 유무 확인)

***⇒ 한 번에 가져온다고 하면 SOCKET이 응답을 기다릴 때 오래 걸리면 ERROR. 하지만 BATCH SIZE를 정해서 분할해서 가져오면 더욱 수월하다.***

***31,000,000개 부터 버퍼 에러 난다. 따라서 다시 진행. BUFFER\_SIZE를 1로 진행.***

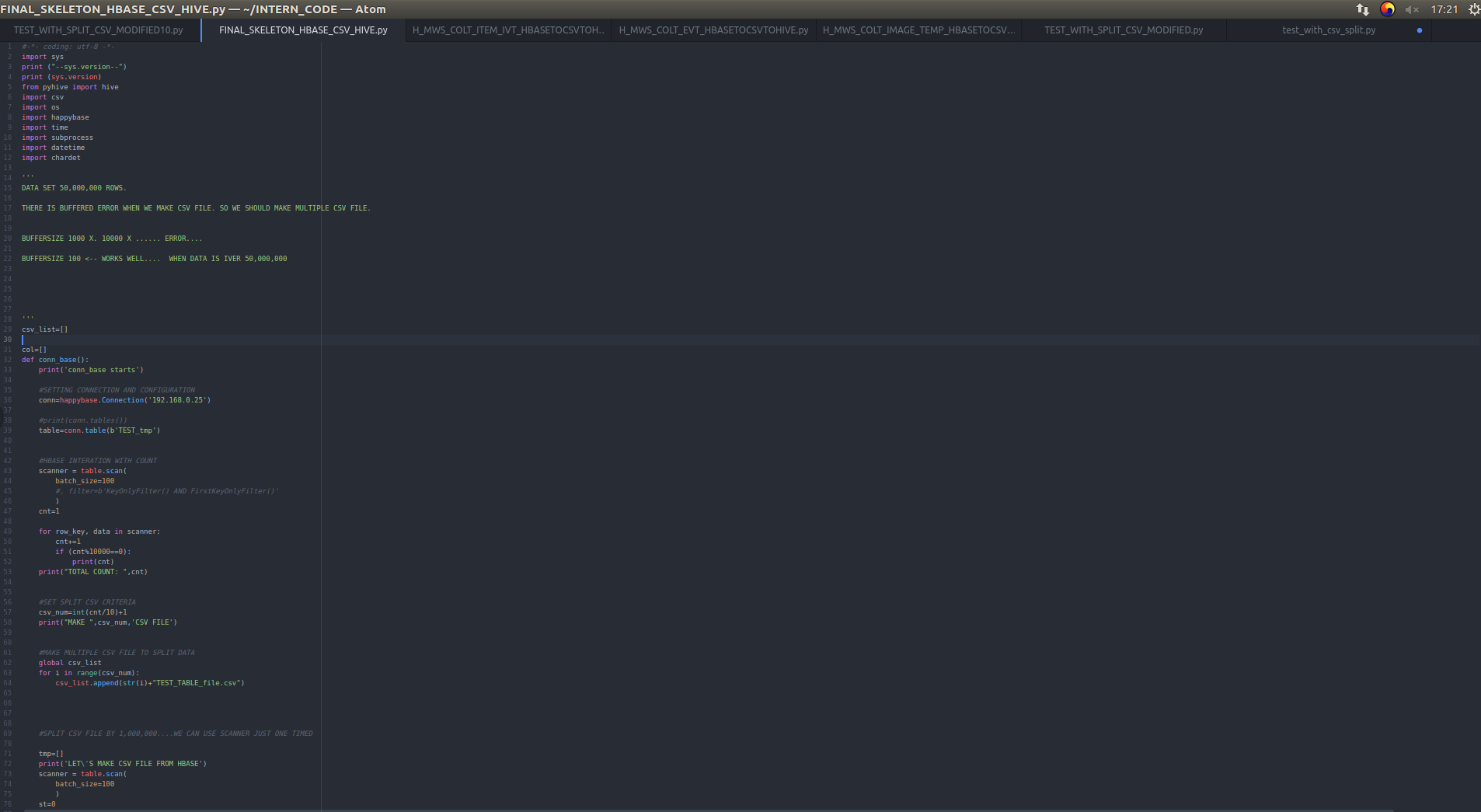
***여기서 과연 BATCH 만큼 가져오는 것이 맞나 싶다. 그래서 개수를 세기 위해 사용했던 SCANNER를 이용해서 진행해보았다.***

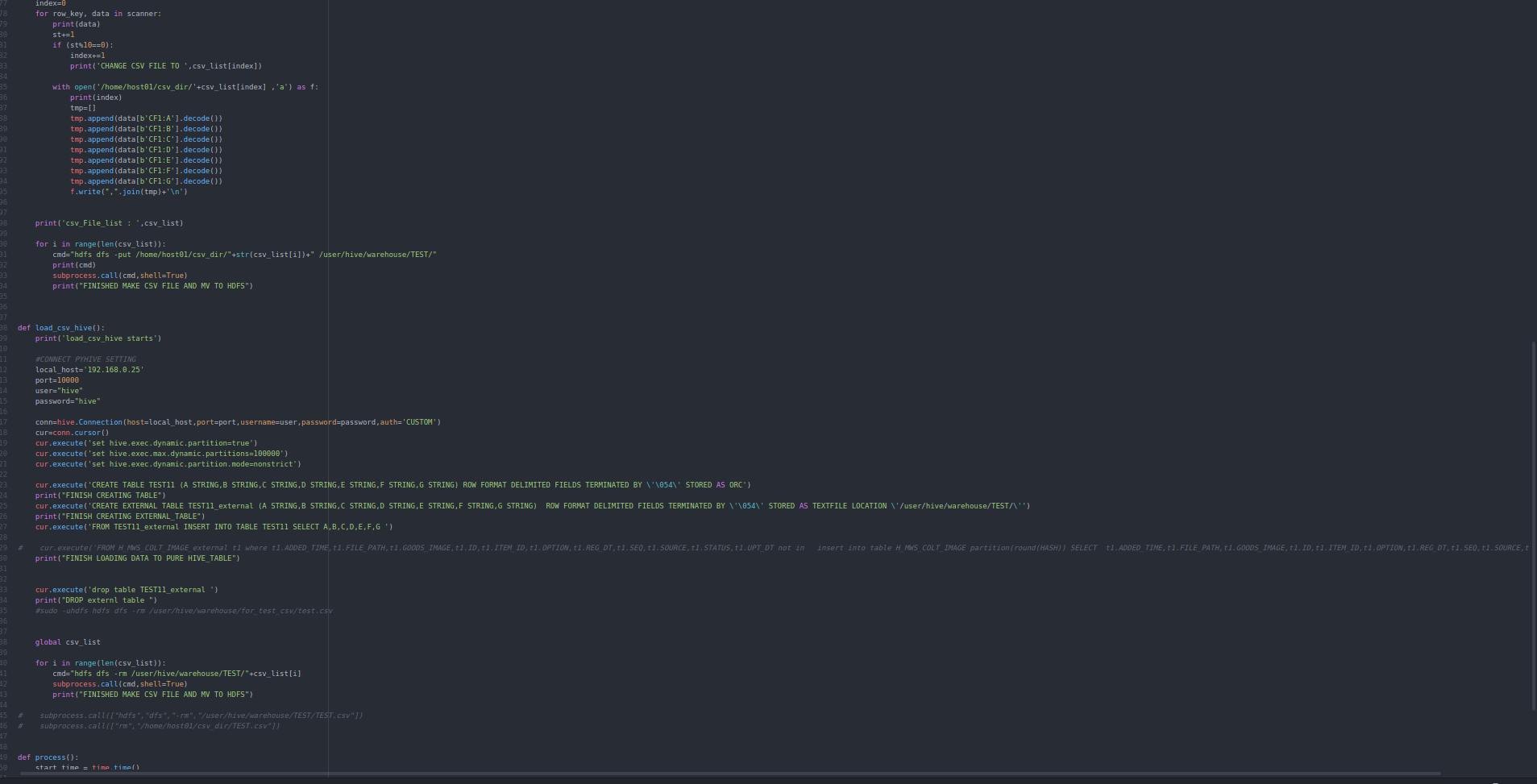


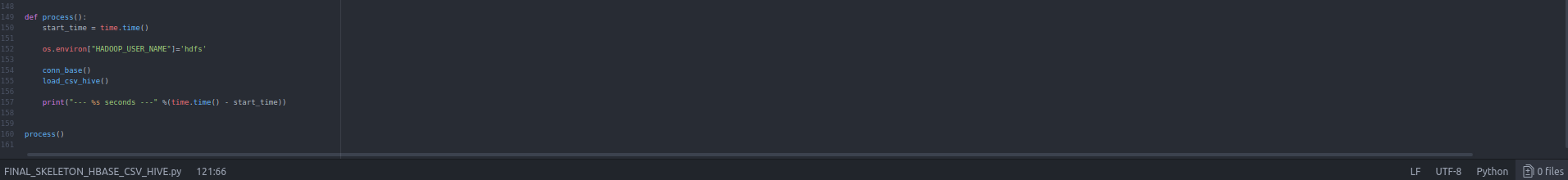
(SCANNER를 이용해서 진행.)

***즉, BUFFER SIZE를 100으로 정하고 CSV FILE들을 EXTERNAL TABLE 이 참조할 LOCATION에 넣어주면 된다 또한 SCANNER를 이용한 KEY-DATA 탐색..***

***'FINAL\_SKELETON\_HBASE\_CSV\_HIVE.py’ 를 SKELETON 으로 보고 응용***





****

**(**전체 코드)

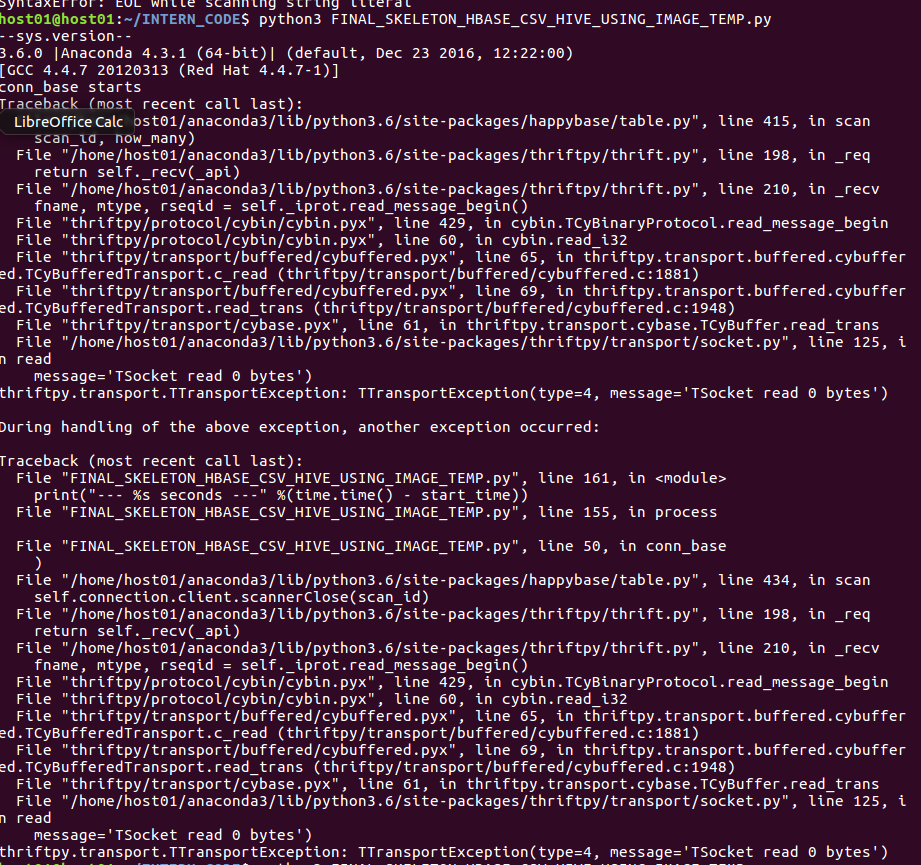
**ISSUE HANDLE**

⇒ BUFFERED MEMORY ISSUE : BATCH , SCANNER , COUNT NUMBER OF ROWS AND SPLIT CSV FILE

⇒ HDFS COMMAND : SUBPROCESS

⇒ HDFS PERMISSION:

CDH HDFS PERMISSION OFF , HIVE EXPORT HADOOP\_USER-NAME=HDFS



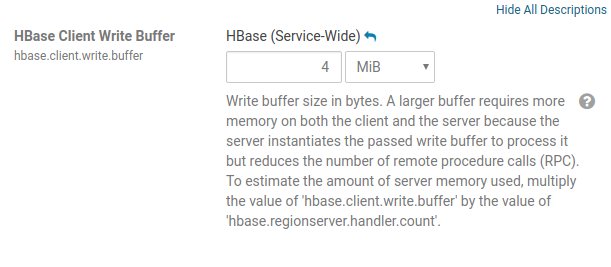
(여전히 이런 Issue 존재).

⇒ <https://stackoverrun.com/ko/q/11326207>

위와 같은 방식으로 !

EPOPCON DV CLUSTER의 ‘HBASE CLIENT WRITE BUFFER’ 크기를 4MIB 로 증가시킴.

근데 아마 EPOPCONDV CLUSTER가 아닌 TOAST ICT 133.186.XXX.XXX 의 BUFFER를 늘려야 할 것 같다.



(EPOPCON DV CLUSTER에서의 BUFFER 변경)

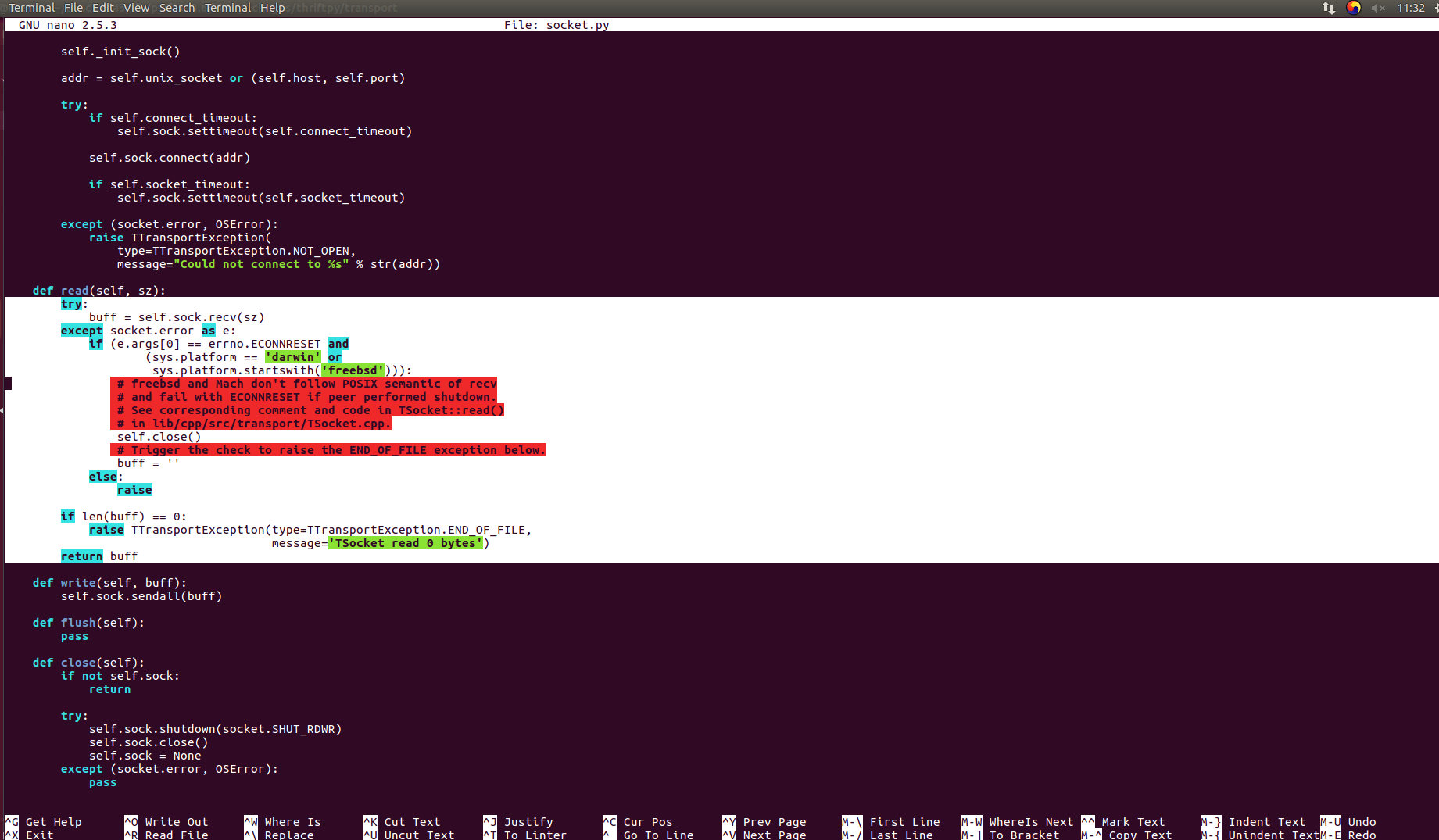
HBASE CLIENT WRITE BUFFER란?

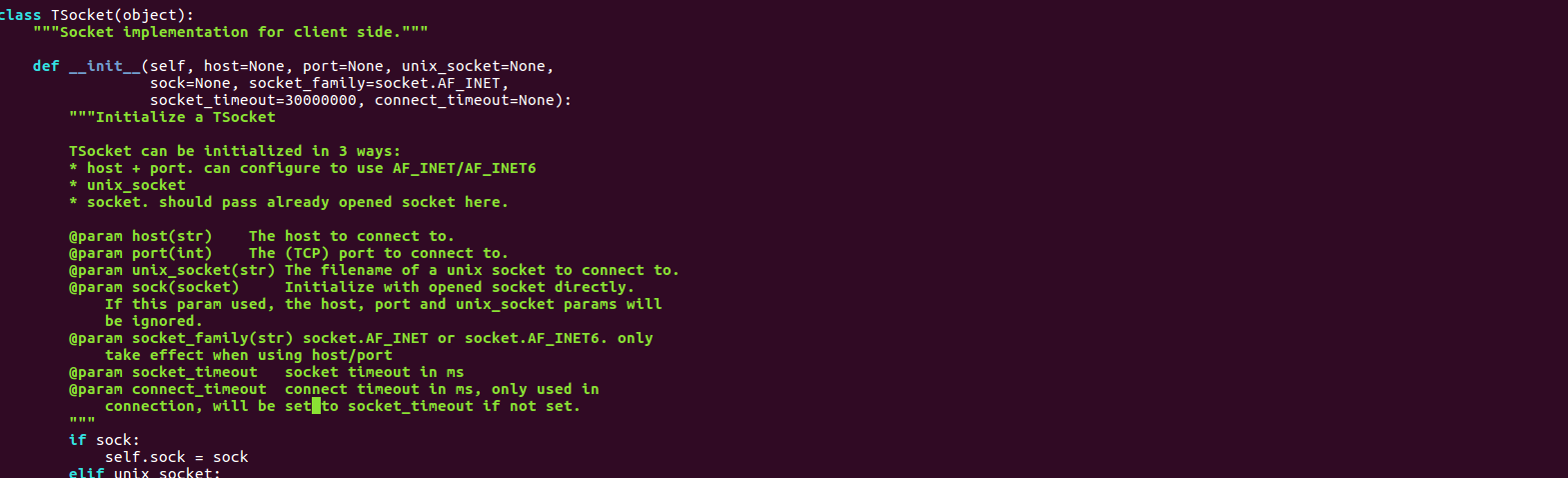
→ HTable 클라이언트 쓰기 버퍼의 기본 크기 (바이트). 버퍼가 클수록 클라이언트와 서버 측 모두에서 더 많은 메모리가 필요하지만 서버가 전달 버퍼를 인스턴스화하여 처리하므로 버퍼 크기가 커지면 수행되는 RPC 수가 줄어 듭니다.

**다른 방법으로 ERROR를 보고**



직접 TIMEOUT부분을 수정하고자 한다.





(이 부분에서 SOCKET INIT 시, SOCKET TIMEOUT을 300000으로 줬다 .DEFAULT 3000. ).....

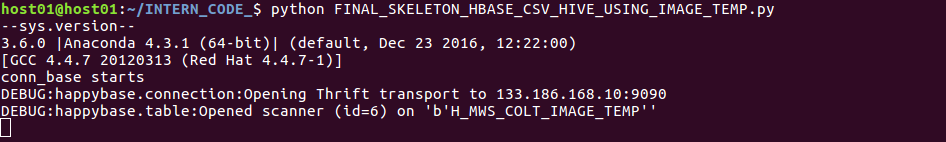
이렇게 해도 안됨. SCANNER 자체를 쓰지 않고 FOR KEY,VALUE IN TABLE.SCAN( {

TIME.SLEEP(60)

)

이런식으로 차라리 중간에 쉬는 시간을 주는 방식으로 진행.





(중간 LOG를 볼 수 있게 했다)

방식을 바꿔서 DATA 10000개를 읽어서 넣는 방식이 아니라 하나하나를 바로 CSV FILE에 넣는데 그 대신 10000개 마다 CSV 추가해 간다.

***차이점이라고 하면 아까보다 REQUEST를 여러 번에 나눠서 준 것이다. 이번에 되면 좋겠다 제발..***

***안됨....***

성현님은 5000 ROW를 옮기고 60초 TIME.SLEEP하신다고 한다.

나도 그렇게 해보려고 한다. 이것도 ERROR.

***마지막으로 SCAN하지 않고 그냥 ROW 하나하나를 가져오도록 한다.***

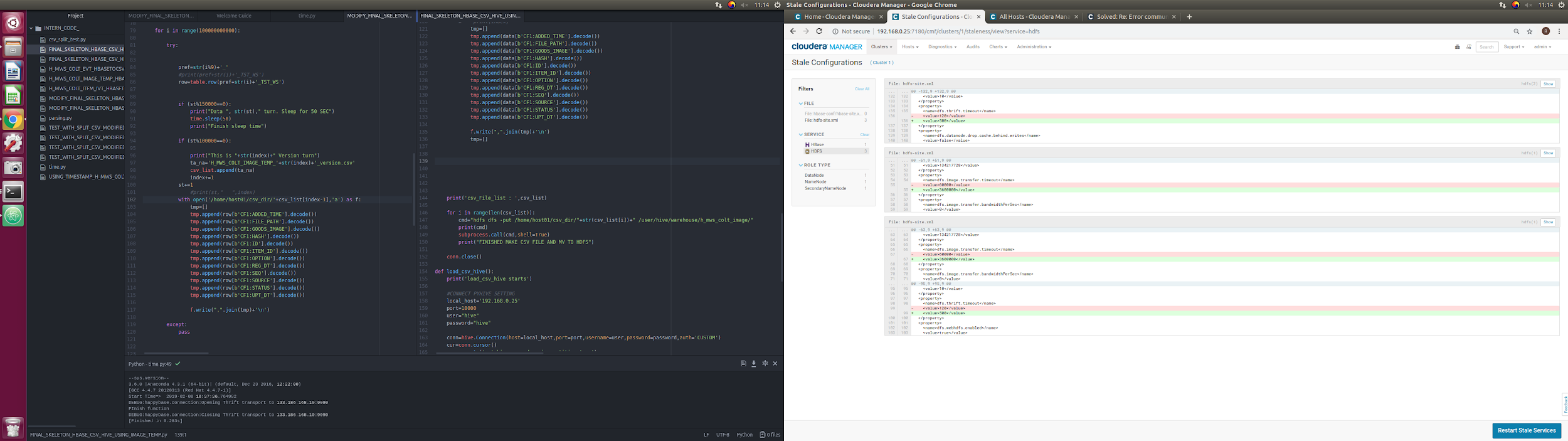
***100,000 ⇒ 15 분 걸림.***

***이렇게 해서 가져오고 또 HIVE\_TABLE에 넣어야 함.***

***EVT TABLE / IMG TABLE 을 돌림. 2019.02.07***

***각각 10만 ROW => 15분 걸림.***

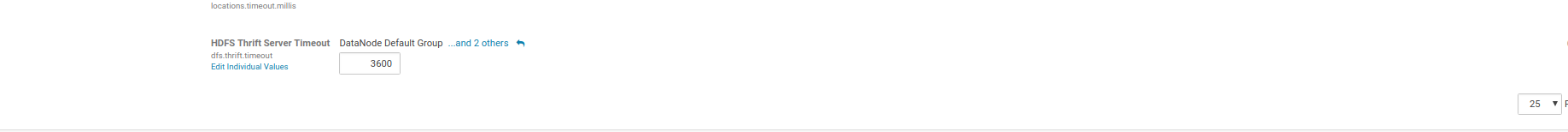
하지만 재수 없게도 파일을 실행해놓고 집에 갔다왔으나 제대로 동작하지 않았다.

******

(CDH MONITOR에서 TIMEOUT 관련한HDFS 부분을 재설정했다.)

***또한 NETWORK 부분의 SOCKET BUFFER 크기를 늘렸다.***

(https://www.systutorials.com/241303/how-to-enlarge-linux-udp-buffer-size/)

******

(이 부분에 대한 수정이 뭔가 의미가 있는 것 같다 .)

ROW단위로 가져오는 것 : UPDATE문제. (TIMESTAMP 이용 , ROW-KEY 등등 불가)

SCAN으로 가져오는 것: 적절하다.

***TSOCKET READ 0 BYTES ERROR****:*

1. NETWORK ISSUE
2. REMOTE SERVER RESTART
3. TIME OUT.

NETWORK BUFFER SIZE늘림.

REMOTE 켜있음. 접근 확인

TIME OUT → 시간 조정함. HBASE/HDFS

**코드에서 NETWORK 부분에 대해서 TIME.SLEEP부분이 좋지 않다고 함.**

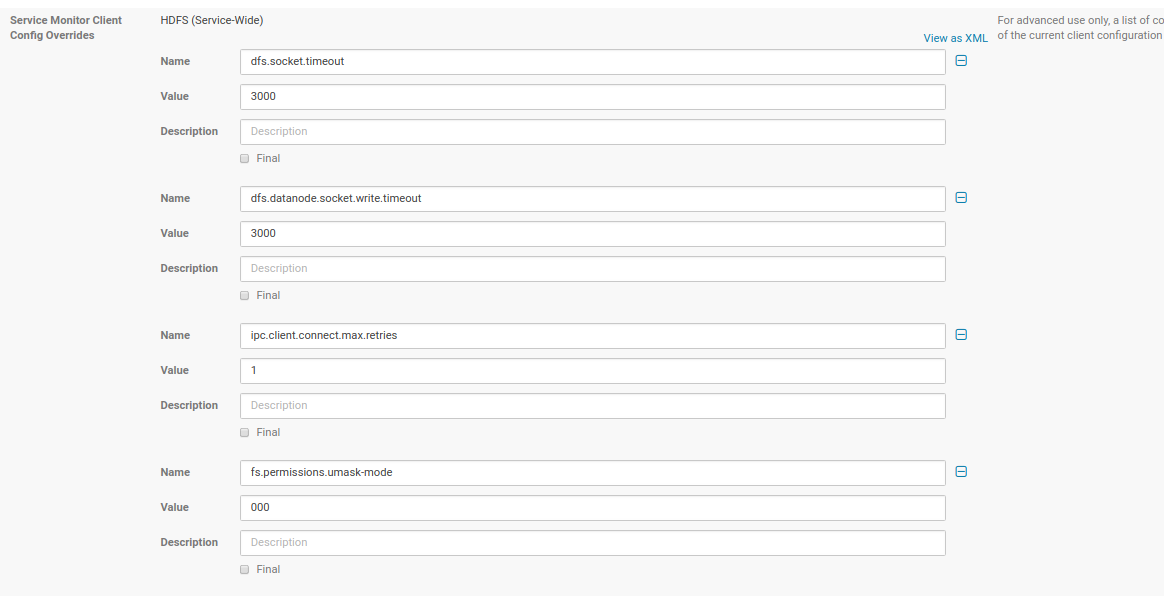
동작 확인.

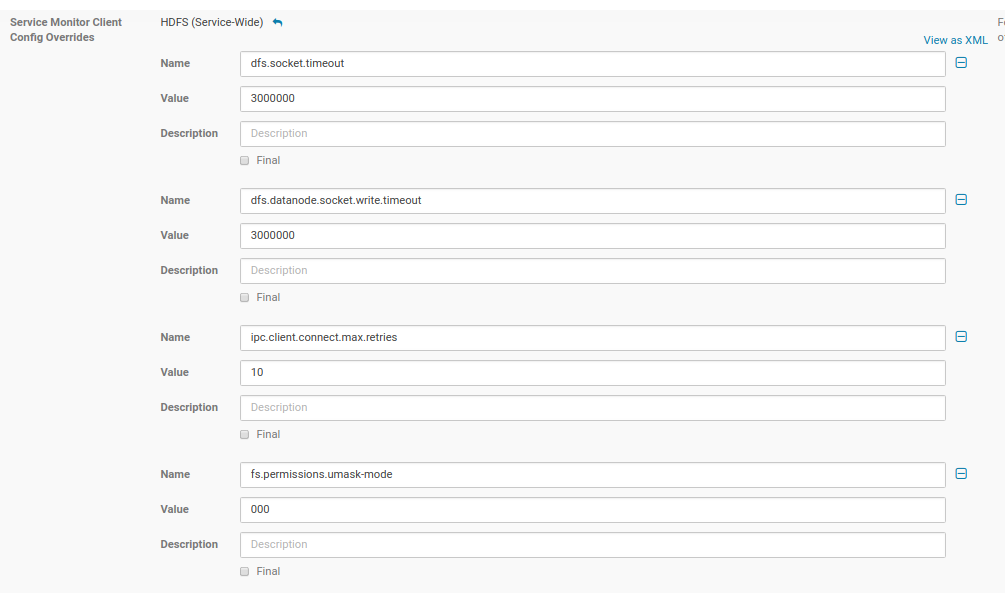
1700 만개 DATA REMOTE HBASE → LOCAL CSV 70분

LOCAL CSV → HIVE\_PURE 6분

⇒ 77분.

**또 에러 발생!**

****

****

**IN CONCLUSION…**

*새벽에 돌려라. NETWORK 빠를 때.*

*중간 중간에 TIME.SLEEP을 최소화로 줘라.*

*최대한 DISK IO 시간이 안걸리도록 CSV FILE을 크게 작성해라*

*CSV 파일 크기를 계산해서 최대 ROW를 파악해라.(COLUMN당 BYTE파악)*

**17000000: 3026.20350SEC(HBASE -> CSV).+189.11SEC(CSV->HIVE)**

**SQOOP이용해서 RDB → HIVE**

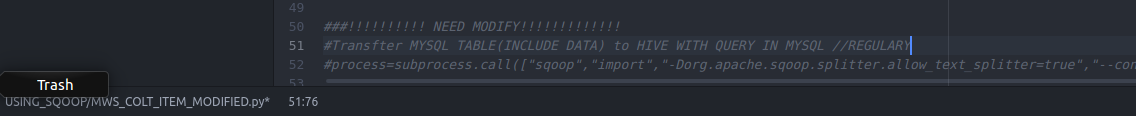
****

→ JAVA 1.8버전에서 COMPILE하고 실행을 JAVA7에서 해서 발생한문제.

따라서 CDH는 1.7 버전인데 LOCAL은 1.8 VERSION이어서 발생했기 때문에 LOCAL

VERSION을 1.7으로 DOWNGRADE

**MR 작업 중 MEMORY ALLOCATION ISSUE**



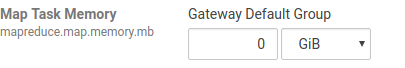
(MR작업 과정 중에 ERROR발생.)

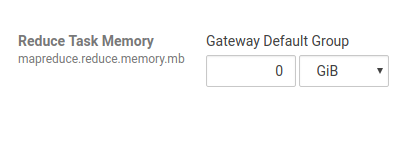
→ 해당 ERROR는 메모리 부족 문제라고 SEARCH결과 나옴. 따라서 CDH에서 조정

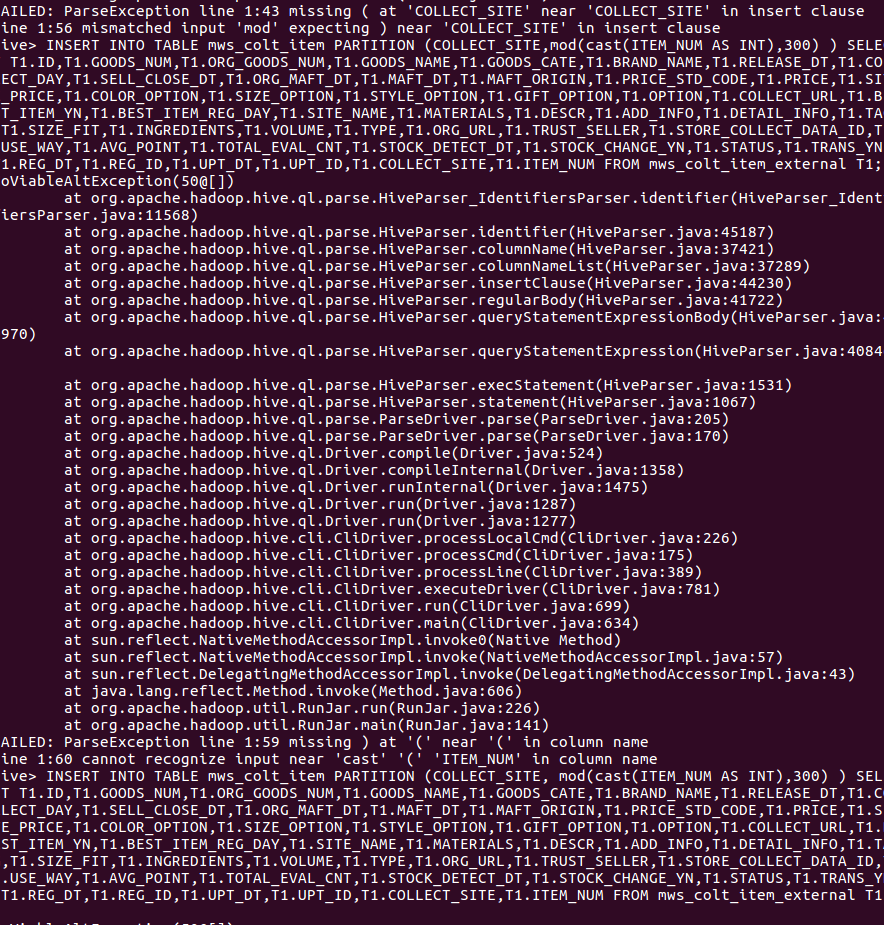
****

(위에서부터 4 2 2G로 설정)

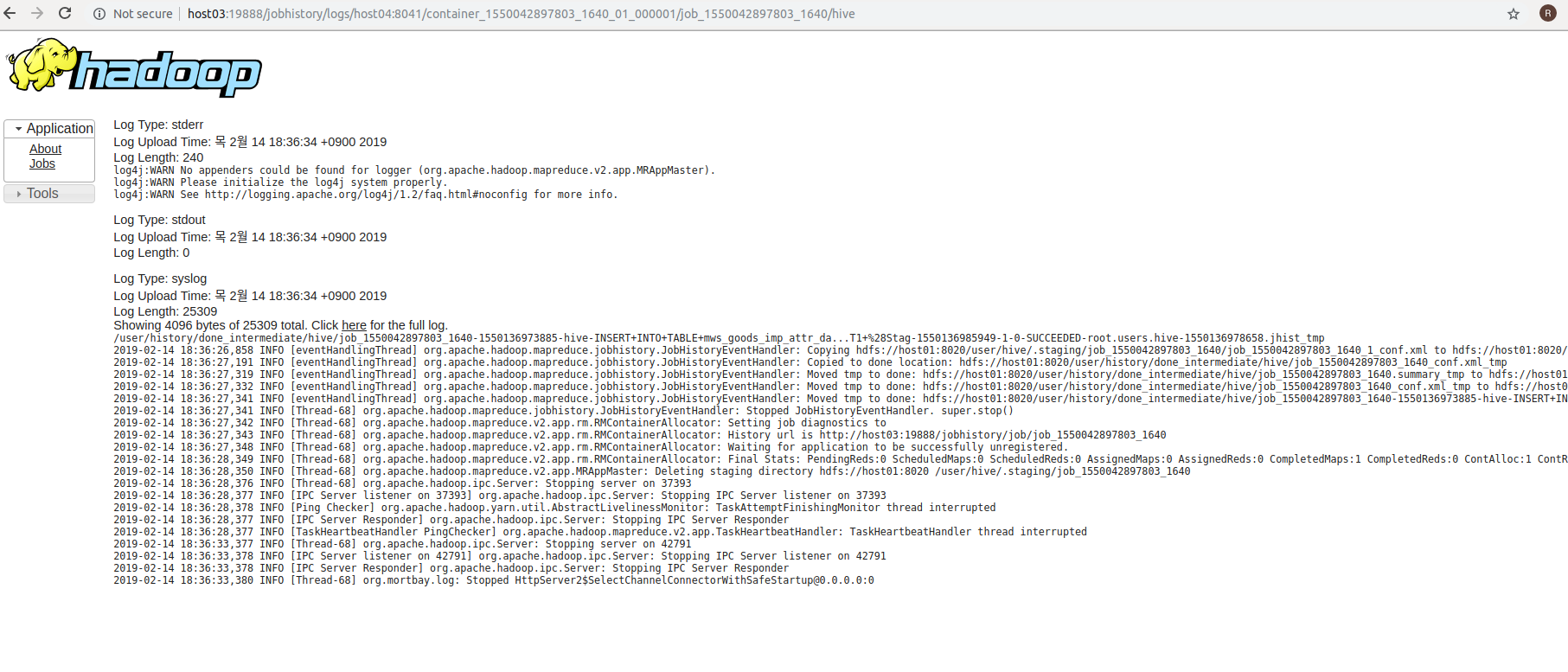
→ 밑은 DEFAULT로 설정 되어있던 부분.

****

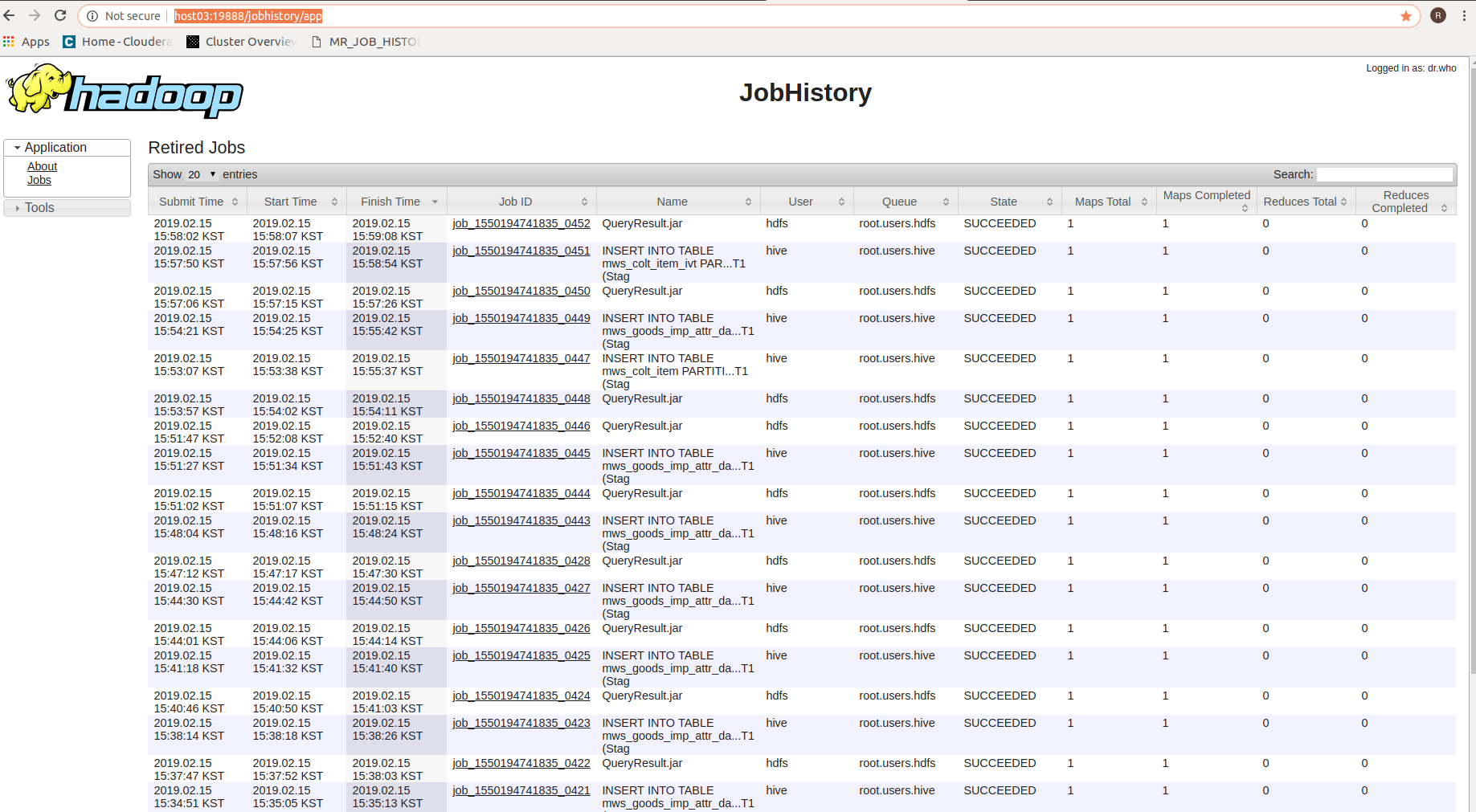
****



(이러한 에러 발생. 이 에러를 잡기 위해서 HDFS JOB TRACKER 따라감)

****

(PER NODE 의 MAX DYNAMIC PARTITION 개수가 적어서 발생한 문제. 따라서 늘려줌으로서 해결)



**(이렇게 문제 발생하면 ERROR LOG 뜨는 곳을 잘 찾아서 해결해라)**

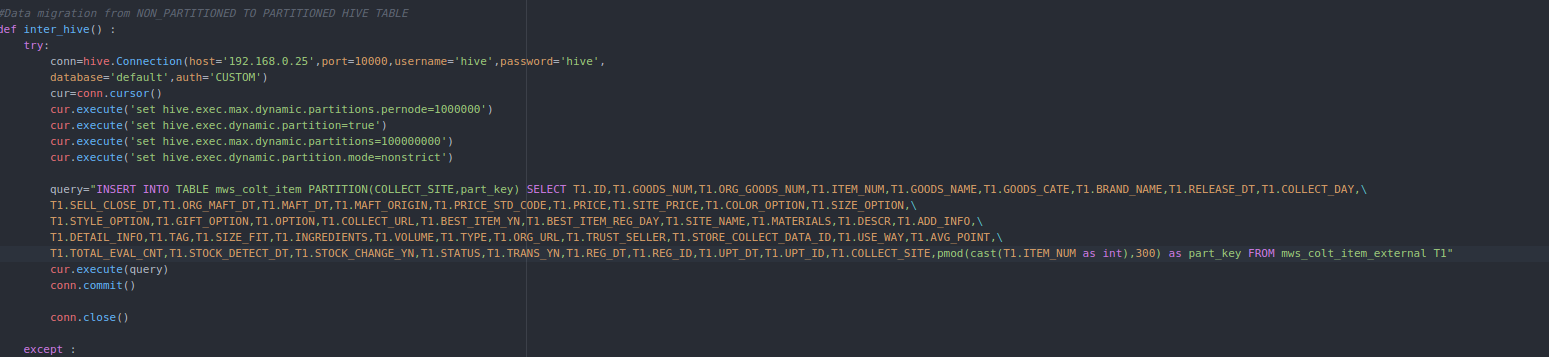
***또한 HIVE PARTITION TABLE을 만들 때 꼭 주의할 것!!!***

***INSERT INTO 구문에서!!!***

INSERT INTO TABLE A PARTITION(하,INT(나)) SELECT 가,다.….. 하,나 FROM B

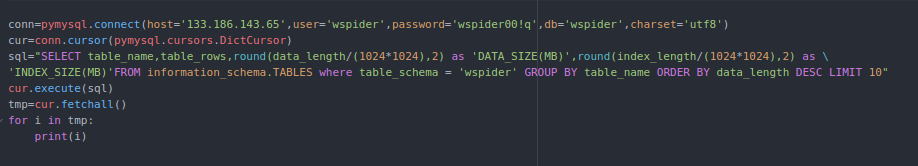
구문이 있다고 하면, EXTERNAL 혹은 SOURCE가 되는 테이블 컬럼 '나'를 SELECT에 쓰고 PARTITION에서 수정하는 것이 아니라 뒤 SELECT에서 수정한 후 별칭을 달아주고 그것을 PARTITION 컬럼으로 넣는다.

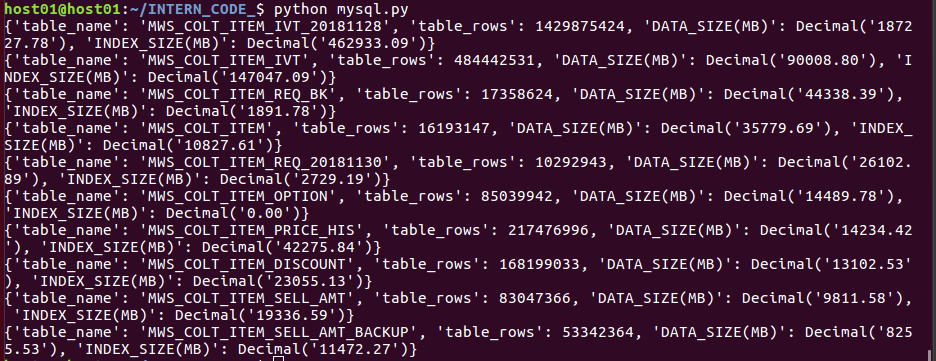
INSERT INTO TABLE A PARTITION(가,PARTITION\_KEY) SELECT 가,다....하.INT(나) AS PARITION\_KEY이렇게 작성.



(이런식으로 구성)

**사용한 DB정보**





(위 QUERY 의 결과)