**Spark**

-분산 저장/처리

#hdfs, (mapreduce , 스파크는 쓰지 않음!)

-In memry 방식으로 처리속도가 빠르다. (파일을 주로 이용하는 하이브와 다르다.)

-haddop ecosystem

-하이브는 파일(하드웨어)에 의존하여 저장처리한다. 데이터가 ND에 쪼개져서 저장되어 다시 불러 와서 처리할때도 시간이 걸린다.

- SQL을 지원한다.

# 하이브 장점은 SQL을 사용해서 MAPREDUCER를 활용한다.

- Streaming (실시간 전달되는 데이터를 받아서 처리)

- ML이 가능 (ML, Machine Learning)

- hdfs에서 data를 불러오고 저장,

- hdfs영역을 불러와서 통계, ml이용한 분석이 공부의 초점이다.

Hive (SELECT)

-table을 생성(sql의 create 명령어)

Spark (VIEW로 등록한 다음에 그 이름으로 SQL을 돌린다.)

- table 생성하지 않음

- DataFrame: 메모리에 존재하는 테이블

\*디스크에 있는 파일 CSV파일이 메모리에 존재하고 있다고 보시면 된다.

\*메모리에 있는 가상의 테이블(VIEW 명령어)

|  |
| --- |
| HIVE를 쓰는 이유?  -hive가 spark보다 환경설치도 훨씬 어렵고 사용도 훨씬 어려운데 업계에서는 spark만 쓸 것 같다는 생각이 듭니다. hive도 쓰나요?  가장 많이 사용되는 DBMS는 RDB인데 RDB와 Hive는 호환성이 있기 때문에 RDB의 데이터를 쉽게 Hive로 표현할 수 있는 점이 장점이죠. 또 Hive에서 RDB로 데이터 복사도 쉬운 편이고요. Hive와 Spark를 연동하여 사용하면 어느정도 성능상의 보완도 가능합니다. 기존 데이터베이스가 Hive와 유사하고 호환성이 있는 점이 장점이라고 할 수 있겠네요.  \* 데이터베이스라는 데이터의 집합을 만들고, 저장 및 관리할 수 있는 기능들을 제공하는 응용 프로그램 |

**[실습1] hdfs 파일을 불러오는 방법 (codfile: Spark-test-01.ipynb)**

전시간에 올려 놓은 biabets.csv를 활용

Spark는 hdfs와 분산처리 기능이 있기 때문에 이것을 확인한다.

코드

import findspark

findspark.init()

from pyspark.sql import SparkSession

sparkSession = SparkSession.builder.appName("example-pyspark-hdfs").getOrCreate()

df\_load = sparkSession.read.csv('hdfs://localhost:9000/user/data/csv/diabetes.csv')

# appName 확인

sparkSession.sparkContext.appName # example-pyspark-hdfs

print( type(df\_load) )

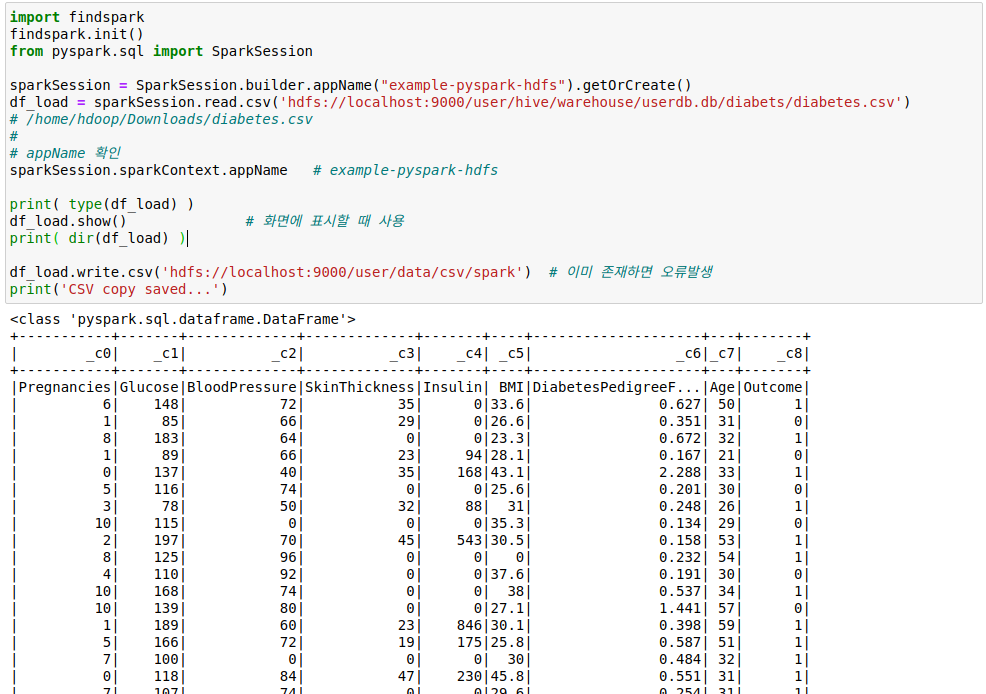
df\_load.show() # 화면에 표시할 때 사용

print( dir(df\_load) )

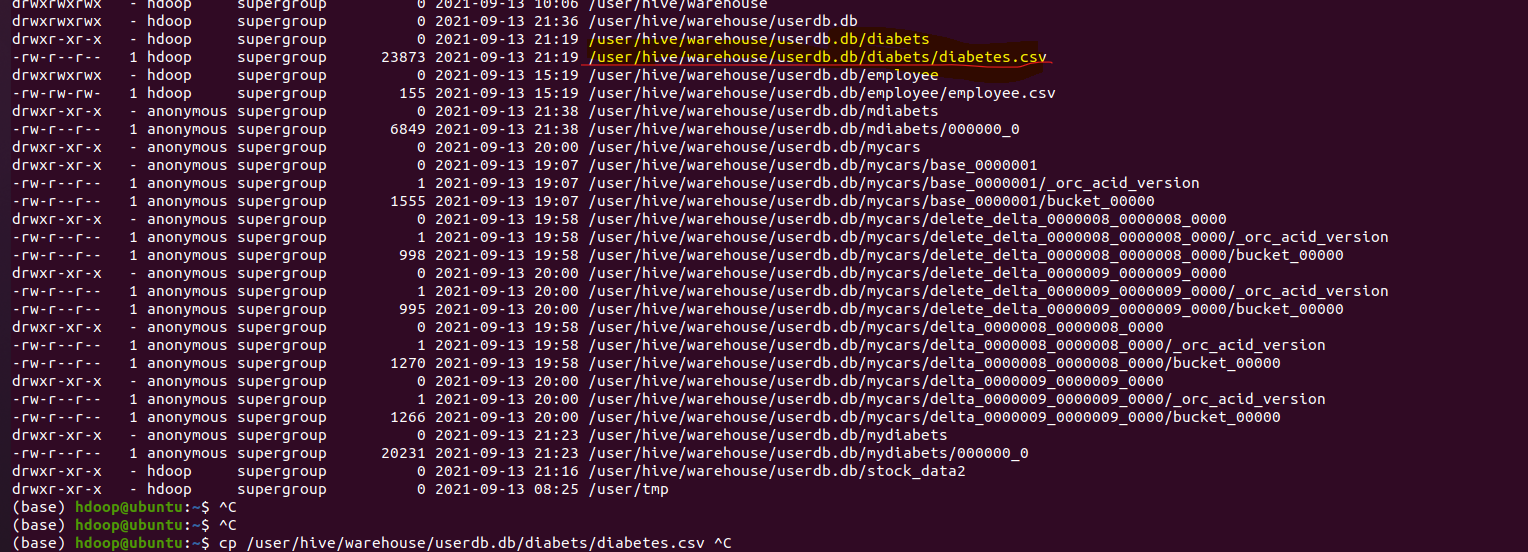
df\_load.write.csv('hdfs://localhost:9000/user/data/csv/spark') # 이미 존재하면 오류발생

print('CSV copy saved...')

실행코드



파일위치 확인

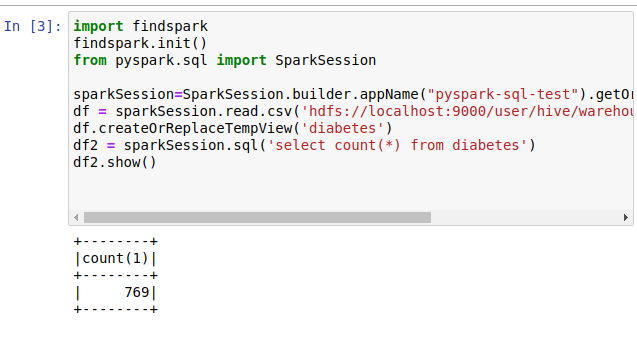


# spark df와 pandas df와 다르다

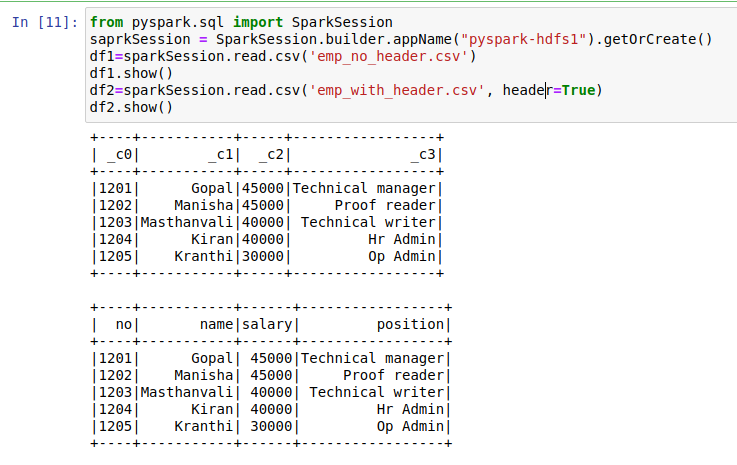
# 하지만, s df와 p df 호환해주는 모듈이 있다.

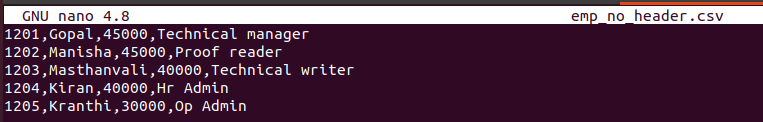
Put (copyfromlocal) write ~ get(copytolocal) read

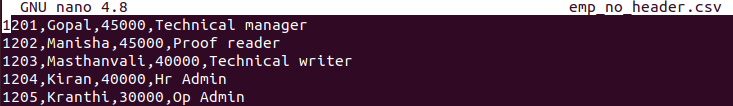
**[실습3] Start-all.sh를 실행한 상태에서 해야 hdfs 파일을 불러오기**



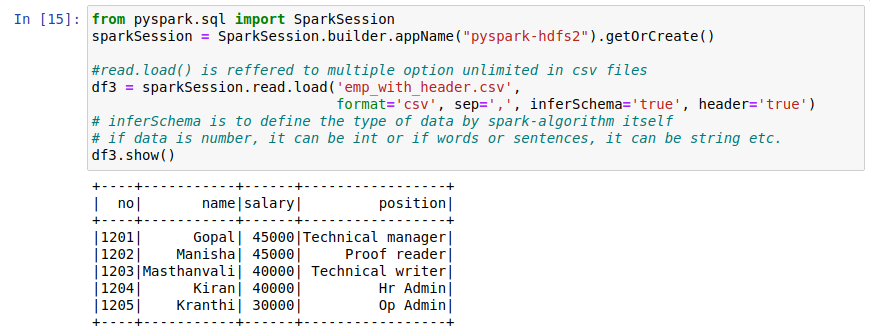
**[실습2 ] 로컬 데이터 불러오는 방법 (codefile:** **Spark-test-02.ipynb)**





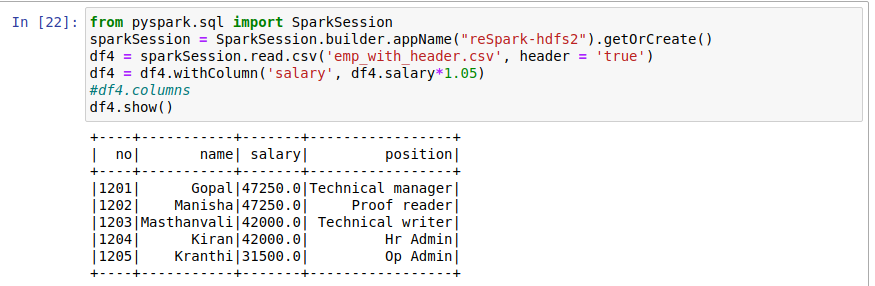


자료형을 지정하지 않을 경우

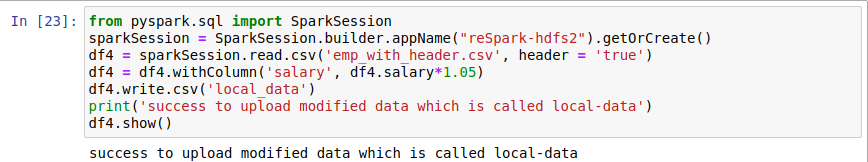


**[실습3] 로컬데이터 수정하기 (codefile:** **Spark-test-02.ipynb)**

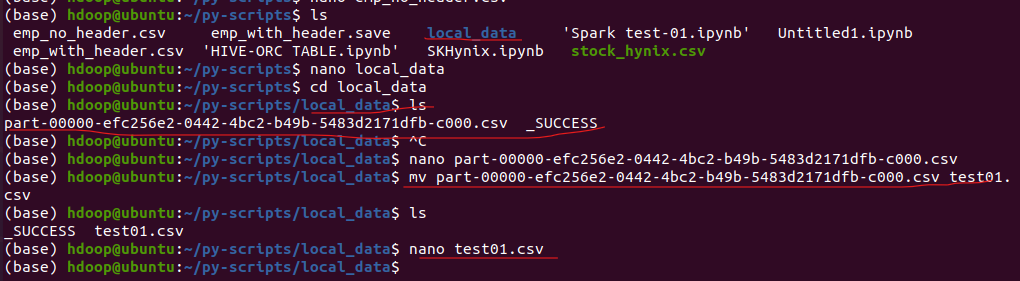
코드 및 실행



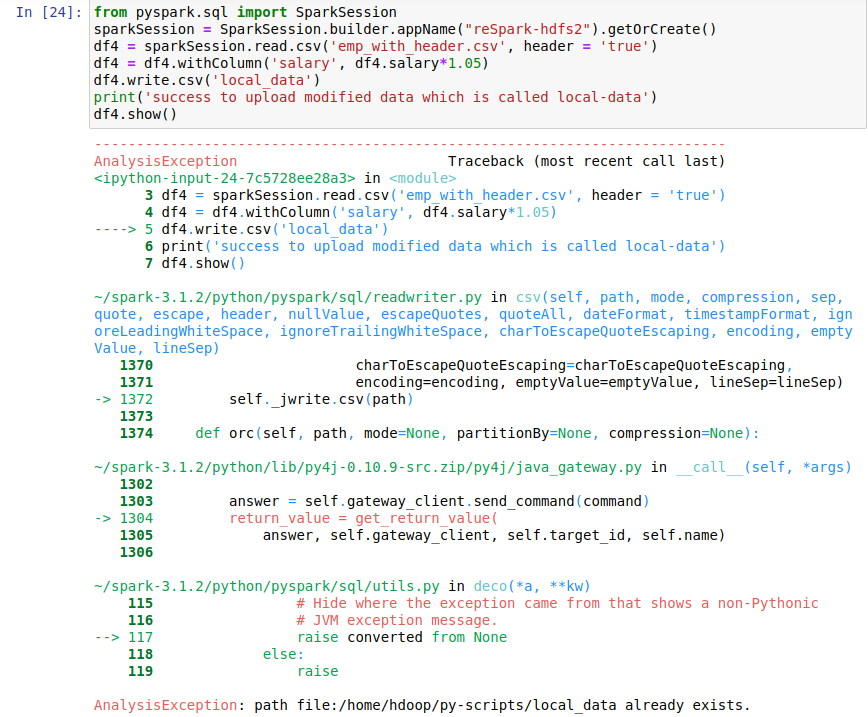
**[실습4] 로컬데이터 수정한 내용 저장하기 (codefile:** **Spark-test-02.ipynb)**



내용확인



>> 생성된 csv파일명을 test01.csv로 변경함.

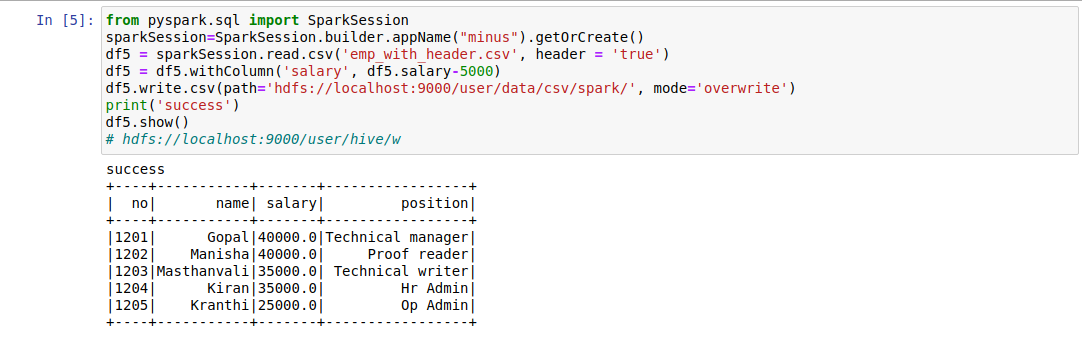


문제발생

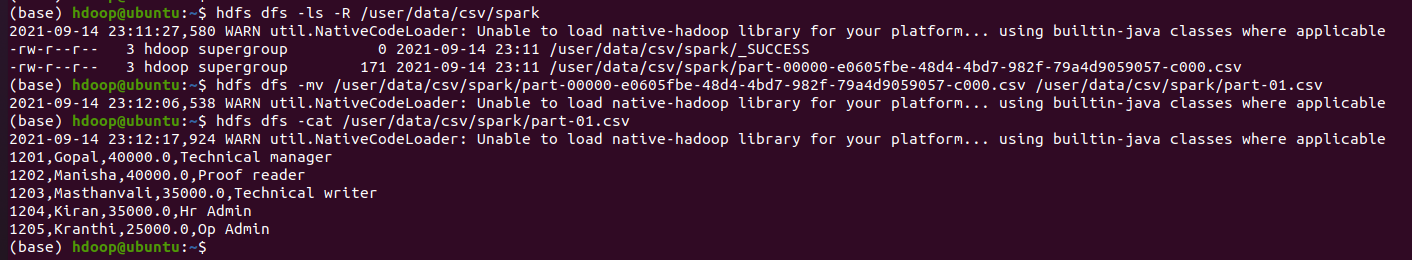
|  |
| --- |
| 문제원인: 지정된 위치에 이미 폴더가 생성했음을 알림  해결방법: 1) mode = ‘append’  2) mode = ‘overwrite’  차 이 점: append는 실행할 때마다 실행횟수 만큼 파일이 생성되지만,  overwrite은 실행횟수와 상관 없이 한번만이다.  (\*) 아래는 실행횟수르 2번씩 했을 경우임. |
| Mode = ‘append’인 경우      >> 실행횟수 2회만큼 파일이 2개 생김 |
| Mode = ‘overwrite’인 경우      >> 실행횟수 2회를 했으나 파일이 하나만 생김 |

**[실습5] 로컬파일을 불러와서 수정한 다음 hdfs에 저장 (codefile:** **Spark-test-02.ipynb)**

실행 및 코드



내용 확인



**[실습 6] 크롤링 🡪 pdf 🡪 sdf작업 (codefile: Spark-test-03.ipynb)**

**- yahoo finance 모듈을 활용하여 crawling하여 data 참조변수로 메모리에 할당**

**- 참조변수 data를 pandas dataframe을 통해서 형변환**

**- arrow 모듈을 활용하여 panas dataframe을 통해서 spark dataframe으로 형변환**

**\* 특징: 파일 디스크에 저장되지 않고 메모리만을 활용**

