

스파크 데이터세트 생성

순천향대학교 컴퓨터공학과

이 상 정



순천향대학교 컴퓨터공학과

1

스파크 데이터세트 생성

학습 내용

1. 데이터세트와 데이터프레임 생성
2. SFPD 데이터 세트 생성

순천향대학교 컴퓨터공학과

2

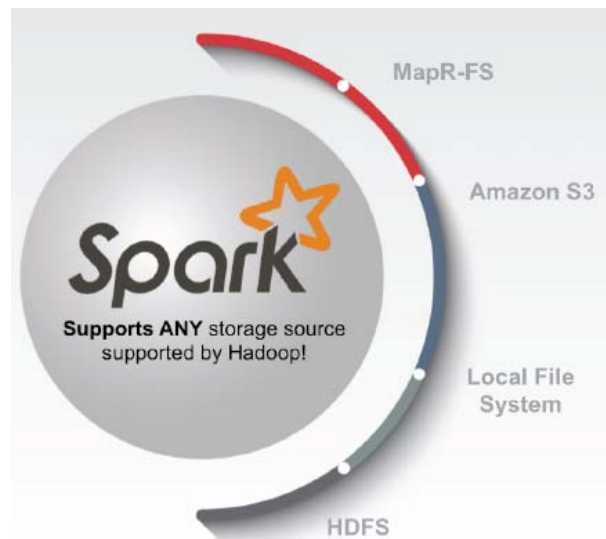
1. 데이터세트와 데이터프레임 생성

스파크 데이터세트 생성

데이터 소스

□ 하둡 클러스터가 지원하는 모든 저장장치에서 스파크로 적재 가능

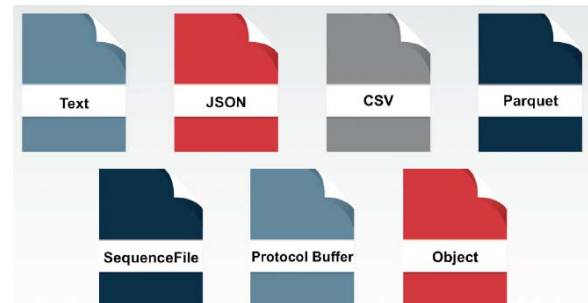
- 로컬 파일 시스템
- HDFS
- Hive
- HBase
- Amazon S3
- MapR-FS
- JDBC 데이터베이스



데이터 형식

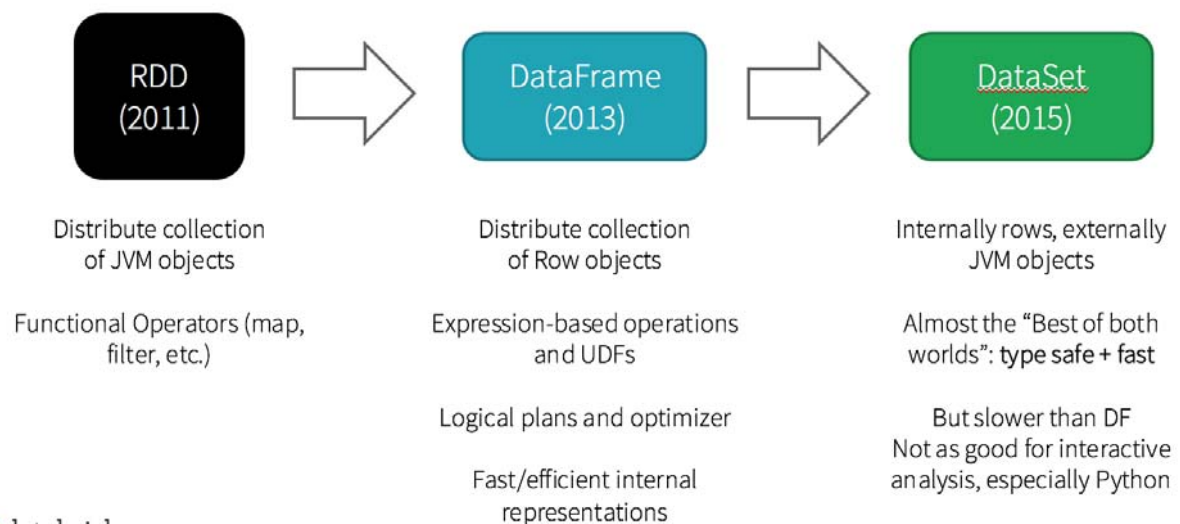
□ 스파크는 하둡이 지원하는 모든 데이터 형식 지원

- 텍스트 파일, CSV
- JSON
- 파케이(Parquet)
 - 컬럼 기반 저장 파일
- SequenceFile
 - 이진 키/값 저장 파일
- Protocol Buffer
 - Google에서 개발한 직렬화 형식
- Object Storage
 - 데이터를 객체로 관리 저장



스파크 API 발전 단계

History of Spark APIs



스파크 RDD

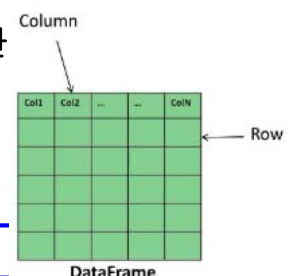
□ RDD (Resilient Distributed Datasets)는 스파크의 저수준의 기본 컬렉션

- 비구조화된 데이터 (unstructured data)
 - 바이너리 스트림 (미디어), 텍스트 스트림
- 타입이 정의되지 않은 데이터로 저수준 변환 및 액션 사용
- 응용에서 작성된 데이터셋트는 최종 계산 시 내부에서 RDD를 사용

스파크 데이터프레임

□ 데이터프레임(DataFrame)은 클러스터의 노드에 분산된 객체 (테이블)들의 컬렉션

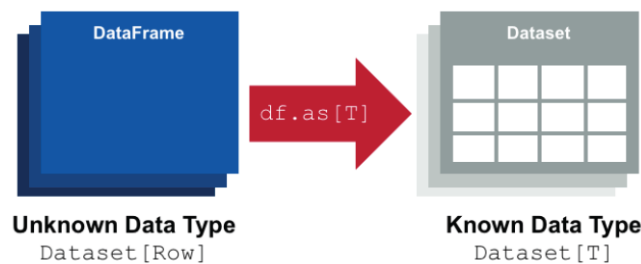
- 이름을 갖는 열(named column)들로 구성된 데이터프레임 상에서 연산 수행
 - 각 행이 여러개의 열을 갖는 테이블 형식으로 데이터를 저장
 - 개념적으로 관계형 데이터베이스의 테이블, 파이썬의 데이터 프레임과 유사
- 데이터프레임이 생성된 후에는 변경 불가능 (immutable)
- 데이터프레임은 디스크 또는 메모리 상에서 저장 및 캐싱
- 한 노드의 태스크가 실패하면 데이터프레임은 나머지 노드에서 자동으로 재구축되어 작업을 완료
- 각 열의 스키마(schema)를 지정하여 데이터셋으로 변환
 - 스키마는 열과 열의 타입들의 리스트
- 실행 시 스키마에 명시된 데이터 타입 일치 여부 확인



스파크 데이터셋

□ 데이터셋(Dataset)은 데이터프레임과 거의 유사

- 데이터프레임과 달리 **컴파일 시** 스키마에 명시된 데이터 타입 일치 여부 확인
 - 타입 안정성 (type -safe)
- 데이터프레임은 타입 Row를 갖는 데이터셋 , **Dataset[Row]**
 - Row 타입은 스파크가 사용하는 '연산에 최적화된 내부적인 표현 방식'
 - 데이터프레임 연산이 데이터셋 연산 보다 성능 우수




스키마 (Schema)

□ 데이터셋의 스키마는 데이터가 저장되는 논리적인 구조


- 데이터베이스의 스키마와 유사
- 스파크의 데이터셋은 **테이블 형식**으로 저장
- csv나 파케이 등의 원시 데이터의 파일들은 정의된 **스키마(테이블 구조)**와 일치해야 함

Schema

IncidentNum	Category	Descript	DayOfWeek
150599321	OTHER_OFFENSES	POSSESSION_OF_BURGLARY_TOOLS	Thursday
156168837	LARCENY/THEFT	PETTY_THEFT_OF_PROPERTY	Thursday
150599224	OTHER_OFFENSES	DRIVERS_LICENSE/SUSPENDED_OR_REVOKED	Thursday
150599230	VANDALISM	MISCELLANEOUS DAMAGE TO WINDOWS	Thursday



Parquet



CSV

스키마 정의

□ 스키마를 정의하는 2가지 방법

- 스칼라의 **케이스 클래스(case class)**를 사용하여 **열의 이름과 타입**을 지정하여 테이블 스키마를 정의
 - 소스 데이터의 열의 이름과 케이스 클래스의 정의와 일치해야 함
 - 스칼라의 케이스 클래스는 22개의 필드로 제한됨
 - 데이터프레임을 데이터셋으로 변환할 때에도 케이스 클래스 사용
 - `.as[T]` 적용
- 프로그램**으로 스키마 정의
 - StructField** 객체의 배열로 구성된 **StructType** 객체를 생성하여 스키마 지정
 - 22개 이상의 필드를 사용할 수 있음
 - 데이터프레임을 데이터셋으로 변환할 때에는 케이스 클래스 사용

□ 스키마를 모르는 경우에 스파크는 열 이름은 "_c0", "_c1", "_c2", …… 등으로 지정하여 데이터프레임을 생성

스파크 셸 (Spark Shell)

□ 스파크 셸은 사용자와 상호작용하며 프로그램 작성 가능

- 스칼라, 파이썬 셸
 - REPL (Read-Evaluate-Print Loop)
- 셸이 시작하면 **스파크세션(SparkSession)**이 초기화되고 셸에 적재
 - 스파크세션은 한 응용의 **클러스터 연결 접근 방식**을 관리

Welcome to
 version 2.1.0-mapr-1707



Scala

Python

스파크 셀에서 데이터셋 생성

- 스파크세션을 이용하여 데이터를 적재하면 데이터셋을 자동으로 생성



2. SFPD 데이터셋 생성

SFPD 시나리오 (1)

□ SFPD (San Francisco Police Department) 데이터

- <https://datasf.org/opendata/>
 - 2013년 1월 ~ 2015년 7월 까지 SFPD 범죄 사건 리포팅 시스템의 데이터
- 지역 내 최대 범죄 발생 지역, 최대 5가지 범죄의 유형 등 조사

Field	Description	Example Value
IncidentNum	Incident number	150561637
Category	Category of incident	NON-CRIMINAL
Descript	Description of incident	FOUND_PROPERTY
DayOfWeek	Day of week that incident occurred	Sunday
Date	Date of incident	6/28/15

15

SFPD 시나리오 (2)

Field	Description	Example Value
Time	Time of incident	23:50
PdDistrict	Police Department District	TARAVAL
Resolution	Resolution	NONE
Address	Address	1300_Block_of_LA_PLAYA_ST
X	X-coordinate of location	-122.5091348
Y	Y-coordinate of location	37.76119777
PdID	Department ID	15056163704013

SFPD 시나리오 (3)

- CSV 파일로 로컬 파일 시스템에 저장
- 테이블 형식으로 각 행이 개별 사건을 나타내는 열들로 구성

Incident Num	Category	Descript	DayOf Week	Date	Time	PdDistrict	Address
150599321	OTHER_OFFENSES	POSSESSION_OF_BURGLARY_TOOLS	Thurs	7/9/15	23:45	CENTRAL	JACKSON_ST/POWELL_ST
156168837	LARCENY/THEFT	PETTY_THEFT_OF_PROPERTY	Thurs	7/9/15	23:45	CENTRAL	300_Block_of_POWELL_ST
150599224	OTHER_OFFENSES	DRIVERS_LICENSE/SUSPENDED_OR_REVOKED	Thurs	7/9/15	23:36	PARK	MASONIC_AV/GOLDEN_GATE_AV
150599230	VANDALISM	MALICIOUS_MISCHIEF/BREAKING_WINDOWS	Thurs	7/9/15	23:20	NORTHERN	1000_Block_of_POLK_ST

SFPD 데이터 조사 질의 내용

□ 데이터 조사 질의

- 가장 사건이 많이 발생한 5개의 주소(address)는?
- 가장 사건이 많이 발생한 5개의 지구대(district)는?
- 가장 많은 10개의 사건 해결 유형(resolution)은?
- 가장 많은 10개의 범죄 유형(category)은?



SFPD 환경 및 데이터 다운로드 (1)

□ 스파크 예제 디렉토리: ~/spark

- SFPD 디렉토리: ~/spark/sfpd

□ 디렉토리 생성 및 다운로드

• sfpd.csv

- SFPD 디렉토리 생성

```
$ mkdir ~/spark
```

```
$ mkdir ~/spark/sfpd
```

```
$ cd ~/spark/sfpd
```

- 강의 홈페이지에서 다운로드

```
$ wget http://cs.sch.ac.kr/lecture/BigData/download/sfpd.csv
```

- 또는, MapR 사이트에서 예제 소스 및 입력 데이터 다운로드

```
$ wget http://course-files.mapr.com/DEV3600/DEV3600_LAB_DATA.zip
```

```
$ unzip DEV3600_LAB_DATA.zip
```

SFPD 환경 및 데이터 다운로드 (2)

```
bigdata@slave1:~$ ls
hadoop-2.7.7  jdk-8u201-linux-x64.tar.gz  spark-2.4.3-bin-hadoop2.7
hbase-2.2.0  MapReduce
bigdata@slave1:~$ mkdir spark
bigdata@slave1:~$ cd spark
bigdata@slave1:~/spark$ mkdir sfpd
bigdata@slave1:~/spark$ ls
sfpd
bigdata@slave1:~/spark$ cd sfpd
bigdata@slave1:~/spark/sfpd$ wget http://cs.sch.ac.kr/lecture/BigData/download/sfpd.csv
--2019-07-16 03:07:44-- http://cs.sch.ac.kr/lecture/BigData/download/sfpd.csv
Resolving cs.sch.ac.kr (cs.sch.ac.kr)... 220.69.209.31
Connecting to cs.sch.ac.kr (cs.sch.ac.kr)|220.69.209.31|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 57772855 (55M) [text/csv]
Saving to: 'sfpd.csv'

sfpd.csv          100%[=====>] 55.10M  7.58MB/s   in 7.3s

2019-07-16 03:07:52 (7.55 MB/s) - 'sfpd.csv' saved [57772855/57772855]

bigdata@slave1:~/spark/sfpd$ ls
sfpd.csv
```

```
bigdata@slave1:~/spark/sfpd$ more sfpd.csv
150599321,OTHER_OFFENSES,POSSESSION_OF_BURGLARY_TOOLS,Thursday,7/9/15,23:45,CENTRAL,ARREST/BOOKED,JACKSON_ST/P
156168837,LARCENY/THEFT,PETTY_THEFT_OF_PROPERTY,Thursday,7/9/15,23:45,CENTRAL,NONE,300_Block_of_POWELL_ST,-122
150599321,OTHER_OFFENSES,DRIVERS_LICENSE/SUSPENDED_OR_REVOKED,Thursday,7/9/15,23:45,CENTRAL,ARREST/BOOKED,JACK
150599224,OTHER_OFFENSES,DRIVERS_LICENSE/SUSPENDED_OR_REVOKED,Thursday,7/9/15,23:36,PARK,ARREST/BOOKED,MASONIC
156169067,LARCENY/THEFT,GRAND_THEFT_FROM_LOCKED_AUTO,Thursday,7/9/15,23:30,SOUTHERN,NONE,8TH_ST/FOLSOM_ST,-122
150599230,VANDALISM,MALICIOUS_MISCHIEF/BREAKING_WINDOWS,Thursday,7/9/15,23:20,NORTHERN,ARREST/BOOKED,1000_Bloc
150599309,ASSAULT,AGGRAVATED_ASSAULT_WITH_A_DEADLY_WEAPON,Thursday,7/9/15,23:15,TENDERLOIN,NONE,LEAVENWORTH_ST
150599133.OTHER OFFENSES.DRIVERS LICENSE/SUSPENDED OR REVOKED.Thursdav.7/9/15.23:06.RICHMOND.ARREST/BOOKED.CAL
```

SFPD 데이터 하둡 적재 및 스파크 셀 시작

□ 하둡과 스파크 시작

```
$ start-all.sh
$ $SPARK_HOME/sbin/start-all.sh
```

□ 입력 데이터 로컬 파일 **sfpd.csv**를 하둡 파일 시스템의 파일로 복사

- 로컬 파일: ~/spark/sfpd/sfpd.csv
- 하둡 파일: /sparkdata/sfpd/sfpd.csv

```
$ hadoop fs -mkdir /sparkdata
$ hadoop fs -mkdir /sparkdata/sfpd
$ hadoop fs -put sfpd.csv /sparkdata/sfpd
$ SPARK_HOME/bin/spark-shell --master yarn // 스파크 셀 시작
```

SFPD 데이터셋 생성 – 케이스 클래스

□ 케이스 클래스를 사용한 SFPD 데이터셋 생성 절차

- **클래스 импорт**
 - spark.implicit과 서브클래스들을 импорт
 - spark.implicit은 많은 스파크 정의를 포함하는 빌트인 라이브러리
- **케이스 클래스 정의**
 - 케이스 클래스로 스키마 정의
- **데이터 적재 후 데이터셋 생성**
 - spark.read.csv 메서드를 사용하여 데이터 적재
 - spark 변수는 스파크세션을 가리킴
 - **스키마 추측 옵션** 적용하지 않으면 에러 발생
 - » option("inferSchema", true)
 - » incidentnum, pdid 는 추측된 타입 integer, long 으로 변환
 - toDF 를 사용하여 데이터프레임을 생성
 - .as[T] 메서드를 사용하여 데이터프레임을 데이터셋으로 변환
- **데이터셋을 뷰(view)로 등록**
 - 뷰를 사용하여 **SQL 질의**

SFPD 데이터셋 생성 - 코드

```
// 클래스 임포트
import spark.implicits._

// 케이스 클래스 정의
case class Incidents(incidentnum:String, category:String, description:String,
dayofweek:String, date:String, time:String, pddistrict:String, resolution:String,
address:String, X:Double, Y:Double, pdid:String)

// 데이터 적재 후 데이터셋 생성
val sfpdDS =
spark.read.option("inferSchema",true).csv("/sparkdata/sfpd/sfpd.csv").toDF("inci
dentnum", "category", "description", "dayofweek", "date", "time", "pddistrict",
"resolution", "address", "X", "Y", "pdid").as[Incidents]

sfpdDS.printSchema()           // 스키마 프린트

// 데이터셋을 뷰(view)로 등록
sfpdDS.createTempView("sfpd")
```

SFPD 데이터셋 생성 - 스파클 셀 실행

```
scala> import spark.implicits._
import spark.implicits._

scala> case class Incidents(incidentnum:String, category:String, description:String, dayofweek:String, date:String, time:String, pddistrict:String, resolution:String, address:String, X:Double, Y:Double, pdid:String)
defined class Incidents

scala> val sfpdDS = spark.read.option("inferSchema",true).csv("/sparkdata/sfpd/sfpd.csv").toDF("incidentnum", "category", "description", "dayofweek", "date", "time", "pddistrict", "resolution", "address", "X", "Y", "pdid").as[Incidents]
sfpdDS: org.apache.spark.sql.Dataset[Incidents] = [incidentnum: int, category: string ... 10 more fields]

scala> sfpdDS.printSchema()
root
|-- incidentnum: integer (nullable = true)
|-- category: string (nullable = true)
|-- description: string (nullable = true)
|-- dayofweek: string (nullable = true)
|-- date: string (nullable = true)
|-- time: string (nullable = true)
|-- pddistrict: string (nullable = true)
|-- resolution: string (nullable = true)
|-- address: string (nullable = true)
|-- X: double (nullable = true)
|-- Y: double (nullable = true)
|-- pdid: long (nullable = true)

scala> sfpdDS.createTempView("sfpd")

scala>
```

스파크 데이터 적재 메서드 (1)

□ 데이터를 적재하여 데이터프레임을 생성하는 메서드

- `spark.read.load("파일")`
 - 파케이 파일(디폴트)를 적재
- `spark.read.load("파일").format("타입")`
 - 타입: JSON, 파케이(Parquet), CSV
- `spark.read.text("파일")`
 - 텍스트 파일을 적재
- `spark.read.textfile("파일")`
 - 텍스트 파일을 적재하고, 데이터프레임이 아닌 `Dataset[String]`을 리턴
- `spark.read.jdbc(URL, Table, Connection_Properties)`
 - JDBC 연결을 사용하여 데이터베이스 테이블을 적재

스파크 데이터 적재 메서드 (2)

- `spark.read.csv("파일")`
 - `spark.read.load("파일").format("csv")`
- `spark.read.json("파일")`
 - `spark.read.load("파일").format("json")`
- `spark.read.parquet("파일")`
 - `spark.read.load("파일").format("parquet")`

SFPD 데이터셋 생성 - 프로그램

- 프로그램으로 SFPD 데이터프레임 생성 절차
 - 클래스 импорт
 - spark.implicit과 SQL 데이터타입 클래스들을 импорт
 - 프로그램으로 스키마 생성
 - StructField 객체의 배열로 구성된 StructType 객체 생성
 - 데이터 적재 후 데이터프레임 생성
 - spark.read.format 메서드를 사용하여 데이터 적재
 - toDF 를 사용하여 데이터프레임을 생성
 - 케이스 클래스 정의
 - 열 이름과 타입이 지정된 케이스 클래스 정의
 - 케이스 클래스를 사용하여 데이터프레임을 데이터셋으로 변환
 - .as[T] 메서드를 사용하여 데이터셋으로 변환
 - 데이터셋을 뷰(view)로 등록
 - 뷰를 사용하여 SQL 질의

SFPD 데이터셋 생성 (프로그램) - 코드 (1)

```
// 클래스 импорт
import spark.implicits._
import org.apache.spark.sql.types._

// 스키마 생성
val sfpdSchema = new StructType(Array(
    new StructField("incidentnum", StringType, true),
    new StructField("category", StringType, true),
    new StructField("description", StringType, true),
    new StructField("dayofweek", StringType, true),
    new StructField("date", StringType, true),
    new StructField("time", StringType, true),
    new StructField("pddistrict", StringType, true),
    new StructField("resolution", StringType, true),
    new StructField("address", StringType, true),
    new StructField("X", FloatType, true),
    new StructField("Y", FloatType, true),
    new StructField("pdid", StringType, true)
))
```

SFPD 데이터셋 생성 (프로그램) - 코드 (2)

```
// 데이터 적재 후 데이터프레임 생성
```

```
val sfpdDF =
  spark.read.format("csv").schema(sfpdSchema).load("/sparkdata/sfpd/sfpd.csv").to
  DF("incidentnum", "category", "description", "dayofweek", "date", "time",
    "pddistrict", "resolution", "address", "X", "Y", "pdid")
```

```
// 케이스 클래스 정의
```

```
case class Incidents(incidentnum:String, category:String, description:String,
  dayofweek:String, date:String, time:String, pddistrict:String, resolution:String,
  X:Double, Y:Double, pdid:String)
```

```
// 데이터프레임을 데이터셋으로 변환
```

```
val sfpdDS = sfpdDF.as[Incidents]
sfpdDS.printSchema()           // 스키마 프린트
```

```
// 데이터셋을 뷰(view)로 등록
```

```
sfpdDS.createTempView("sfpd")
```

SFPD 데이터셋 생성 (프로그램) - 스파크 셀 실행 (1)

```
scala> import spark.implicits._
import spark.implicits._

scala> import org.apache.spark.sql.types._
import org.apache.spark.sql.types._

scala> val sfpdSchema = new StructType(Array(
  |         new StructField("incidentnum", StringType, true),
  |         new StructField("category", StringType, true),
  |         new StructField("description", StringType, true),
  |         new StructField("dayofweek", StringType, true),
  |         new StructField("date", StringType, true),
  |         new StructField("time", StringType, true),
  |         new StructField("pddistrict", StringType, true),
  |         new StructField("resolution", StringType, true),
  |         new StructField("address", StringType, true),
  |         new StructField("X", FloatType, true),           new StructField("Y", FloatType, true),
  |         new StructField("pdid", StringType, true)
  |       ))
sfpdSchema: org.apache.spark.sql.types.StructType = StructType(StructField(incidentnum,StringType,true), StructField(category,StringType,true), StructField(description,StringType,true), StructField(dayofweek,StringType,true), StructField(date,StringType,true), StructField(time,StringType,true), StructField(pddistrict,StringType,true), StructField(resolution,StringType,true), StructField(address,StringType,true), StructField(X,FloatType,true), StructField(Y,FloatType,true), StructField(pdid,StringType,true))

scala> case class Incidents(incidentnum:String, category:String, description:String, dayofweek:String, date:String, time:String, pddistrict:String, resolution:String, X:Double, Y:Double, pdid:String)
defined class Incidents
```


SFPD 데이터셋 생성 (프로그램) - 스파크 셀 실행 (2)

```
scala> val sfpdDS = sfpdDF.as[Incidents]
sfpdDS: org.apache.spark.sql.Dataset[Incidents] = [incidentnum: string, category: string ... 10 more fields]

scala> val sfpdDS = sfpdDF.as[Incidents]
sfpdDS: org.apache.spark.sql.Dataset[Incidents] = [incidentnum: string, category: string ... 10 more fields]

scala> sfpdDS.printSchema()
root
|-- incidentnum: string (nullable = true)
|-- category: string (nullable = true)
|-- description: string (nullable = true)
|-- dayofweek: string (nullable = true)
|-- date: string (nullable = true)
|-- time: string (nullable = true)
|-- pddistrict: string (nullable = true)
|-- resolution: string (nullable = true)
|-- address: string (nullable = true)
|-- X: float (nullable = true)
|-- Y: float (nullable = true)
|-- pdid: string (nullable = true)

scala> sfpdDF.createTempView("sfpd")

scala> █
```

실습 과제 (1)

- ❑ 강의 시간의 실습 내용을 정리하여 제출
- ❑ 주의!!!, 가상머신 이미지의 실습 시 오류 발생, 아래와 같이 수정
 - 새로운 가상머신 이미지 다운로드하여 실행
 - 또는, 마스터와 슬레이브 아래 설정 수정 추가
 - `~/hadoop-2.7.7/etc/hadoop/yarn-env.sh`

```
export JAVA_HOME="/usr/lib/jvm/jdk1.8.0_201"
export HADOOP_HOME="/home/bigdata/hadoop-2.7.7"
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR="$HADOOP_HOME/etc/hadoop"
export YARN_CONF_DIR="$HADOOP_HOME/etc/hadoop"
```


실습 과제 (2)

- ~/hadoop-2.7.7/etc/hadoop/yarn-site.xml

```
<property>
  <name>yarn.resourcemanager.address</name>
  <value>master:8050</value>
</property>
```

- ~/spark-2.4.5-bin-hadoop2.7/conf/spark-env.sh

```
export HADOOP_HOME="/home/bigdata/hadoop-2.7.7"
export HADOOP_CONF_DIR="${HADOOP_HOME}/etc/hadoop"
export SPARK_WORKER_MEMORY=2g
export YARN_CONF_DIR="${HADOOP_HOME}/etc/hadoop"
```

팀 프로젝트 과제 – 주제 및 데이터 생성

□ 팀 프로젝트 주제 선정

- 팀 프로젝트의 주제 및 목표(분석 내용) 기술
- 팀 프로젝트 소스 데이터 설명
- 팀 프로젝트 데이터 설명
 - 데이터의 출처
 - 데이터의 각 항목 설명

□ 팀 프로젝트 데이터 생성

- 팀 프로젝트 데이터를 사용하여 앞에서 배운 스파크의 데이터 생성을 적용하고 실행

- ❑ MapR Academy, <http://learn.mapr.com/>
 - Introduction to Apache Spark
 - <https://learn.mapr.com/series/sparkv2/dev-360-introduction-to-apache-spark-spark-v21>
 - Lesson 2: Create Datasets