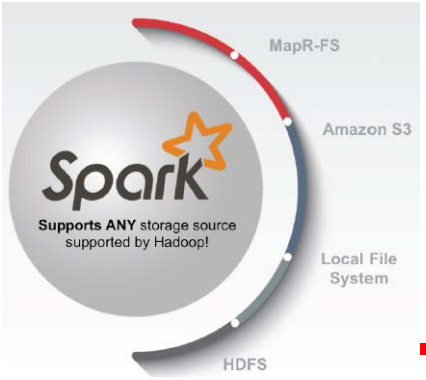


스파크 데이터세트 연산 2

순천향대학교 컴퓨터공학과

이 상 정

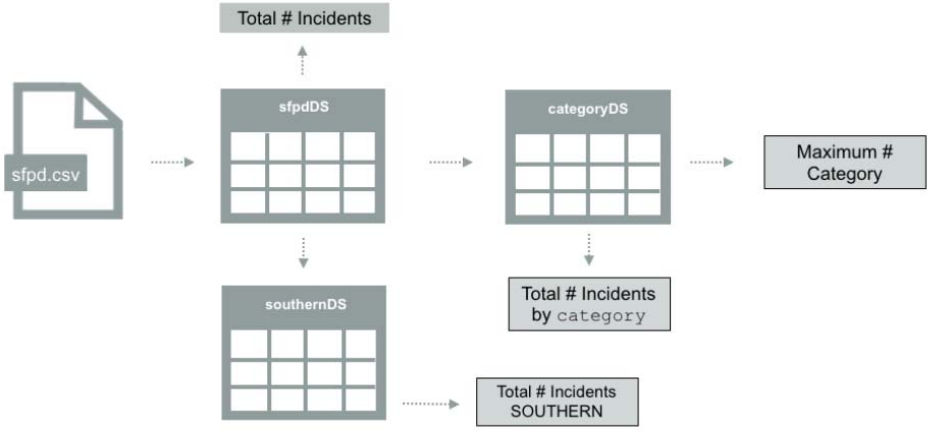
학습 내용

1. 데이터세트 캐싱
2. 사용자 정의 함수

1. 데이터세트 캐싱

계보 그래프 (Lineage Graph)

- 앞에서 언급한 바와 같이 데이터세트는 지연되어 연산 수행
  - 데이터세트 액션 수행 전 까지는 베이스 데이터세트 (sfpdDS)는 생성되지 않음
  - 스파크는 데이세트 액션 수행할 때 마다 데이터세트와 모든 관련 종속 연산을 다시 계산

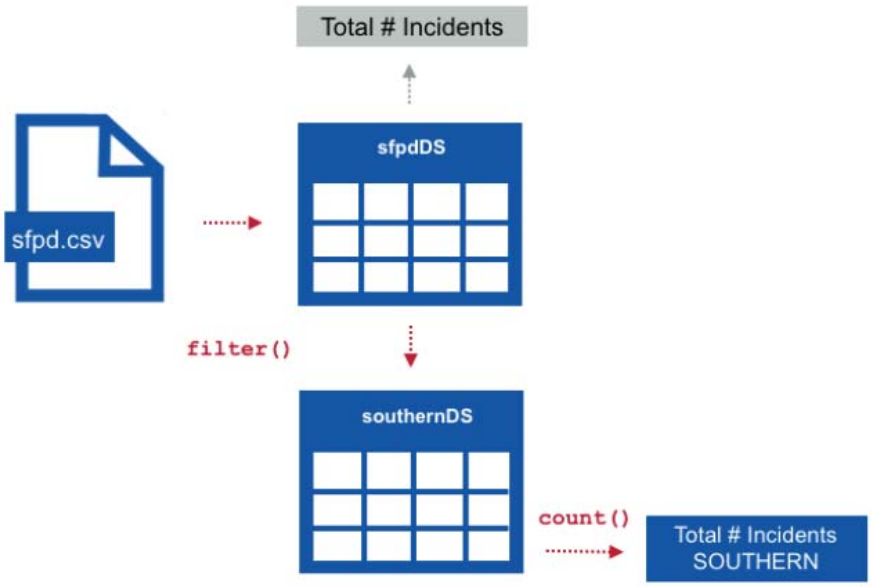


계보 그래프 – 전체 count()

- SFPD 데이터세트의 전체 횟수 카운트 호출할 때 데이터가 sfpdDS로 적재(메모리 적재)되고 count 액션이 실행
  - 액션 실행 후 카운트 값이 리턴되면 데이터는 메모리에서 제거됨

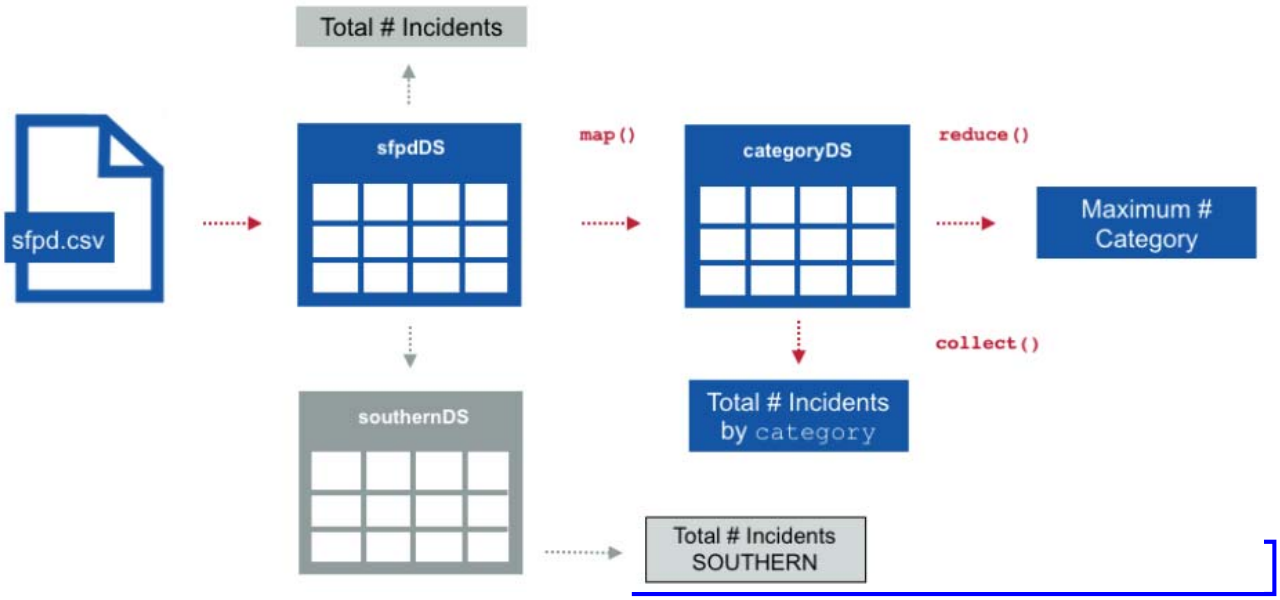
계보 그래프 – southernDS count()

- 남부 지구대의 사건 카운트 시에도 sfpdDS를 다시 적재한 후 필터링하여 sothernDS를 계산하고 카운트



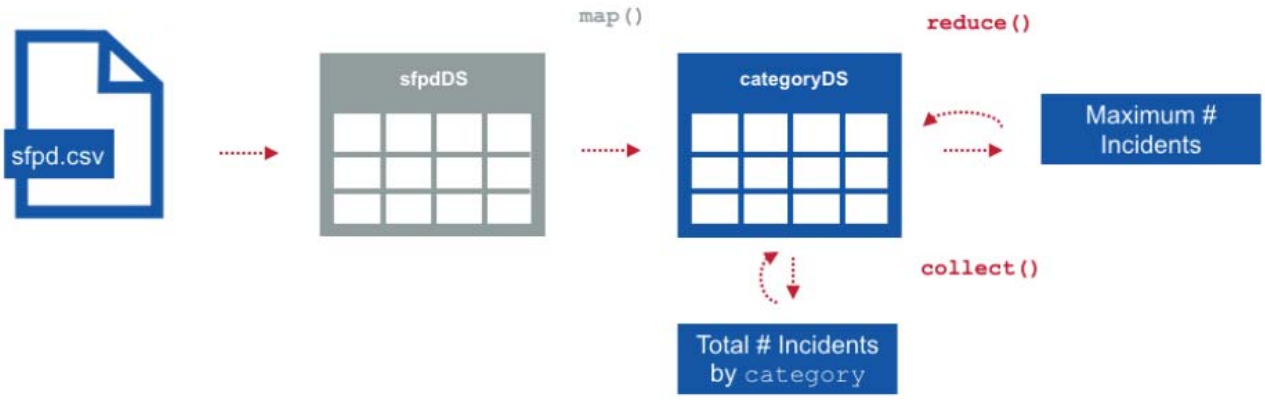
계보 그래프 – collect() & reduce()

- 범죄 유형 당 사건의 수를 수집(collect)하거나 사건의 수가 가장 많은 범죄 유형 계산(reduce)할 때에도 sfpdDS, categoryDS를 계산하여 해당 액션을 수행



데이터세트 캐싱

- 같은 데이터세트를 반복해서 사용하는 경우 매 액션 수행 시마다 데이터세트를 생성하지 않고 메모리에 캐싱(caching)하여 재사용



데이터세트 캐싱 절차

- 데이터세트 캐싱 절차
  - 클래스 임포트
  - 데이터세트 정의
  - 변환 정의
  - 데이터세트 캐싱
    - cache() 메서드를 사용하여 데이터세트를 메모리와 디스크에 캐싱
    - persist() 메서드를 사용할 수도 있음
      - persist(MEMORY_ONLY): 메모리에 캐싱
      - persist(MEMORY_AND_DISK): 메모리와 디스크에 캐싱
        - » cache() 메서드와 동일
  - 액션 적용
  - 캐싱된 데이터를 이 후 액션에도 적용
  - 데이터세트 캐싱 해제
    - unpersist() 메서드를 사용하여 캐싱 해제

데이터세트 캐싱 – SFPD 예

```
import spark.implicits._
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)

// 데이터세트 정의
val sfpdDS =
    spark.read.option("inferSchema",true).csv("/sparkdata/sfpd/sfpd.csv").toDF("incident
    num", "category", "description", "dayofweek", "date", "time", "pddistrict",
    "resolution", "address", "X", "Y", "pdid").as[Incidents]
// 데이터세트 캐싱
sfpdDS.cache()

// 변환 정의
val categoryDS = sfpdDS.groupBy("category" )
// 관계형 그룹 데이터세트(Relational Grouped Dataset)에는 캐싱 적용할 수 없음

// 액션 적용
categoryDS.count.collect
categoryDS.count.collect
// 데이터세트 캐싱 해제
sfpdDS.unpersist()
```

데이터세트 캐싱 – SFPD 실행 예 (1)

SPARK JOBS FINISHED

```
///// 데이터세트 캐싱 예
import spark.implicits._

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)

// 데이터세트 정의
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]
// 데이터세트 캐싱
sfpdDS.cache()
```

```
import spark.implicits._
defined class Incidents
sfpdDS: org.apache.spark.sql.Dataset[Incidents] = [incidentnum: int, category: string ... 10 more fields]
res60: sfpdDS.type = [incidentnum: int, category: string ... 10 more fields]
```

Took 8 sec. Last updated by admin at August 02 2019, 10:40:39 AM.

SPARK JOB FINISHED

```
// 변환 정의
val categoryDS = sfpdDS.groupBy("category" )
// 액션 적용
categoryDS.count.collect
```

```
categoryDS: org.apache.spark.sql.RelationalGroupedDataset = RelationalGroupedDataset: [grouping expressions: [cate
gory: string], value: [incidentnum: int, category: string ... 10 more fields], type: GroupBy]
res64: Array[org.apache.spark.sql.Row] = Array([FRAUD,7416], [SUICIDE,182], [FAMILY_OFFENSES,201], [VEHICLE_THEFT,
17581], [DISORDERLY_CONDUCT,1052], [WARRANTS,17508], [LIQUOR_LAWS,494], [ARSON,690], [FORGERY/COUNTERFEITING,202
5], [GAMBLING,46], [MISSING_PERSON,11560], [BRIBERY,159], [ASSAULT,31843], [SEX_OFFENSES/FORCIBLE,2043], [DRUNKENN
ESS,1870], [RECOVERED_VEHICLE,760], [OTHER_OFFENSES,50611], [STOLEN_PROPERTY,2803], [SECONDARY_CODES,4972], [EXTOR
TION,75], [TREA,6], [SEX_OFFENSES/NON_FORCIBLE,49], [LOITERING,108], [ROBBERY,9658], [DRIVING_UNDER_THE_INFLUENCE,
1038], [SUSPICIOUS_OCC,13659], [WEAPON_LAWS,3975], [PROSTITUTION,1316], [EMBEZZLEMENT,392], [BAD CHECKS,69], [RUNA
WAY,521], [VANDALISM,17987], [DRUG/NARCOTIC,14300], [PORNOGRAPHY/OBSCENE MAT,10], [TRESPASS,2930], [NON-CRIMINAL,5
0...
```

Took 10 sec. Last updated by admin at August 02 2019, 10:41:25 AM.

데이터세트 캐싱 – SFPD 실행 예 (2)

SPARK JOB FINISHED

```
// 액션 적용
categoryDS.count.collect

// 데이터세트 캐싱 해제
sfpdDS.unpersist()
```

```
res66: Array[org.apache.spark.sql.Row] = Array([FRAUD,7416], [SUICIDE,182], [FAMILY_OFFENSES,201], [VEHICLE_THEFT,
17581], [DISORDERLY_CONDUCT,1052], [WARRANTS,17508], [LIQUOR_LAWS,494], [ARSON,690], [FORGERY/COUNTERFEITING,202
5], [GAMBLING,46], [MISSING_PERSON,11560], [BRIBERY,159], [ASSAULT,31843], [SEX_OFFENSES/FORCIBLE,2043], [DRUNKENN
ESS,1870], [RECOVERED_VEHICLE,760], [OTHER_OFFENSES,50611], [STOLEN_PROPERTY,2803], [SECONDARY_CODES,4972], [EXTOR
TION,75], [TREA,6], [SEX_OFFENSES/NON_FORCIBLE,49], [LOITERING,108], [ROBBERY,9658], [DRIVING_UNDER_THE_INFLUENCE,
1038], [SUSPICIOUS_OCC,13659], [WEAPON_LAWS,3975], [PROSTITUTION,1316], [EMBEZZLEMENT,392], [BAD CHECKS,69], [RUNA
WAY,521], [VANDALISM,17987], [DRUG/NARCOTIC,14300], [PORNOGRAPHY/OBSCENE MAT,10], [TRESPASS,2930], [NON-CRIMINAL,5
0...res68: sfpdDS.type = [incidentnum: int, category: string ... 10 more fields]
```

Took 5 sec. Last updated by admin at August 02 2019, 10:41:38 AM. (outdated)

2. 사용자 정의 함수 생성

사용자 정의 함수

- 사용자 정의 함수 (User Defined Function, UDF)를 사용하여 맞춤형으로 적용
  - 데이터세트 연산에 적용
  - SQL 질의 에 적용

Scala SQL

- 데이터세트 연산에 적용
  - udf() 사용하여 인라인 생성

```
val func1 = udf((arguments) => { function definition })
```

- SQL 질의에 적용
  - SparkSession.udf.register() 사용하여 함수 등록

SQL 질의 사용자 정의 함수

- SQL 질의 함수 등록 방법
  - 함수 정의 후 등록
    - 등록 시 SQL에 적용되는 함수 이름, 정의된 함수가 인수로 전달

```
def funcname
.....
SparkSession.udf.register("sqlfuncname", funcname(_:type, ....) )
```

  - 인라인(inline)으로 함수 등록

```
SparkSession.udf.register("sqlfuncname", (arguments) => { function definition })
```


SFPD 예 – 연도 별로 사건 조사

- 연도 별로 사건 조사 예
- SFPD 데이터에서 날짜는 "dd/mm/yy "형식으로 저장
  - 연도 추출을 위해 마지막 '/ ' 다음의 문자열 추출
  - 추출된 년도를 기준으로 계산

Incident Num	Category	Descript	Day OfWeek	Date	Time
150599321	OTHER_OFFENSES	POSSESSION_OF_BURGLARY_TOOLS	Thursday	7/9/15	23:45
156168837	LARCENY/THEFT	PETTY_THEFT_OF_PROPERTY	Thursday	7/9/15	23:45
150599224	OTHER_OFFENSES	DRIVERS_LICENSE/SUSPENDED_OR_REVOKED	Thursday	7/9/15	23:36
150599230	VANDALISM	MALICIOUS_MISCHIEF/BREAKING_WINDOWS	Thursday	7/9/15	23:20

데이터세트 연산 UDF – SPDF 예

```
//// 데이터세트 연산 UDF
// UDF 함수 정의
val getyear = udf((s: String) => {
        val lastS = s.substring(s.lastIndexOf('/')+1)
         lastS
    })

//연도 별로 사건 수 조회
val yy =  sfpdDS.groupBy(getyear(sfpdDS("date"))).count
// val yy =  sfpdDS.groupBy(getyear(col("date"))).count
yy.show
```


데이터세트 연산 UDF – SPDF 실행 예

SPARK JOBS FINISHED

```
//// 데이터세트 연산 UDF
// UDF 함수 정의
val getyear = udf((s: String) => {
        val lastS = s.substring(s.lastIndexOf('/')+1)
         lastS
    })

//연도 별로 사건 수 조회
val yy =  sfpdDS.groupBy(getyear(sfpdDS("date"))).count
yy.show
```

```
getyear: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function1>,StringType,Some(List(StringType)))
yy: org.apache.spark.sql.DataFrame = [UDF(date): string, count: bigint]
+---------+------+
|UDF(date)| count|
+---------+------+
|       15| 80760|
|       13|152830|
|       14|150185|
+---------+------+
```

SQL 질의 UDF – SPDF 예

```
///// SQL 질의 UDF
// 데이터세트를 뷰(view)로 등록
sfpdDS.createOrReplaceTempView("sfpd")

// UDF 함수 정의
def get_year(s:String):String = {
    val year = s.substring(s.lastIndexOf('/')+1)
     year
}

// UDF 함수 등록
spark.udf.register("getYear",get_year(_:String))
// SQL 질의 적용
val numIncByYear = spark.sql("SELECT getYear(date), count(incidentnum) AS
    countbyyear FROM sfpd GROUP BY getYear(date) ORDER BY countbyyear
    DESC")
numIncByYear.show
```


SQL 질의 UDF – SPDF 실행 예

SPARK JOB FINISHED

```
///// SQL 질의 UDF
// 데이터세트를 뷰(view)로 등록
sfpdDS.createOrReplaceTempView("sfpd")

// UDF 함수 정의
def get_year(s:String):String = {
     val year = s.substring(s.lastIndexOf('/')+1)
      year
}

// UDF 함수 등록
spark.udf.register("getYear",get_year(_:String))

// SQL 질의 적용
val numIncByYear = spark.sql("SELECT getYear(date), count(incidentnum) AS countbyyear FROM sfpd GROUP BY getYear
    (date) ORDER BY countbyyear DESC")
numIncByYear.show

get_year: (s: String)String
res101: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function1>,StringType,Some(Lis
t(StringType)))
numIncByYear: org.apache.spark.sql.DataFrame = [UDF:getYear(date): string, countbyyear: bigint]
+-----------------+-----------+
|UDF:getYear(date)|countbyyear|
+-----------------+-----------+
|               13|     152830|
|               14|     150185|
|               15|      80760|
+-----------------+-----------+
```

SQL 질의 UDF (인라인 함수) – SPDF 예

```
///// SQL 질의 UDF, 인라인 함수 (inline function)
// UDF 인라인 함수 등록
spark.udf.register("Getyear", (s: String) =>
    {s.substring(s.lastIndexOf('/')+ 1)})

// SQL 질의 적용
val numIncByYear = spark.sql("SELECT Getyear(date),
    count(incidentnum) AS countbyyear FROM sfpd GROUP BY
    Getyear(date) ORDER BY countbyyear DESC")
numIncByYear.show
```


SQL 질의 UDF (인라인 함수) – SPDF 실행 예

SPARK JOB FINISHED

```
///// SQL 질의 UDF, 인라인 함수 (inline function)
// UDF 인라인 함수 등록
spark.udf.register("Getyear", (s: String) => {s.substring(s.lastIndexOf('/')+ 1)})

// SQL 질의 적용
val numIncByYear = spark.sql("SELECT Getyear(date), count(incidentnum) AS countbyyear FROM sfpd GROUP BY Getyear
    (date) ORDER BY countbyyear DESC")
numIncByYear.show

res107: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function1>,StringType,Some(Lis
t(StringType)))
numIncByYear: org.apache.spark.sql.DataFrame = [UDF:Getyear(date): string, countbyyear: bigint]
+-----------------+-----------+
|UDF:Getyear(date)|countbyyear|
+-----------------+-----------+
|               13|     152830|
|               14|     150185|
|               15|      80760|
+-----------------+-----------+
```

2014년 사건 조사

- 2014년도의 사건의 범죄 유형(category), 해결 유형(resolution) 및 범죄 발생 주소(address) 조사

```
///// 2014년도 사건 조사
val inc2014 = spark.sql("SELECT category,address,resolution, date FROM sfpd WHERE getyear(date)='14'")

// 2014년도 10개의 사건만 표시
inc2014.collect.take(10)

// 2014년도 모든 사건 표시
inc2014.collect.foreach(println)
```


2014년 사건 조사 – 실행 예

```
///// 2014년도 사건 조사
val inc2014 = spark.sql("SELECT category,address,resolution, date FROM sfpd WHERE getyear(date)='14'")

// 2014년도 10개의 사건만 표시
inc2014.collect.take(10)
```

SPARK JOB FINISHED

```
inc2014: org.apache.spark.sql.DataFrame = [category: string, address: string ... 2 more fields]
res113: Array[org.apache.spark.sql.Row] = Array([ASSAULT,0_Block_of_UNITEDNATIONS_PZ,NONE,12/31/14], [NON-CRIMINAL,400_Block_of_POWELL_ST,NONE,12/31/14], [ASSAULT,700_Block_of_MONTGOMERY_ST,NONE,12/31/14], [VANDALISM,FOLSOM_ST/SPEAR_ST,NONE,12/31/14], [NON-CRIMINAL,2800_Block_of_LEAVENWORTH_ST,NONE,12/31/14], [NON-CRIMINAL,3300_Block_of_WASHINGTON_ST,NONE,12/31/14], [OTHER_OFFENSES,3300_Block_of_WASHINGTON_ST,NONE,12/31/14], [LARCENY/THEFT,2000_Block_of_LAGUNA_ST,NONE,12/31/14], [LARCENY/THEFT,CASTRO_ST/17TH_ST,NONE,12/31/14], [LARCENY/THEFT,300_Block_of_POWELL_ST,NONE,12/31/14])
```

Took 6 sec. Last updated by admin at August 02 2019, 11:16:46 AM.

```
// 2014년도 모든 사건 표시
inc2014.collect.foreach(println)
```

SPARK JOB FINISHED

```
[ASSAULT,0_Block_of_UNITEDNATIONS_PZ,NONE,12/31/14]
[NON-CRIMINAL,400_Block_of_POWELL_ST,NONE,12/31/14]
[ASSAULT,700_Block_of_MONTGOMERY_ST,NONE,12/31/14]
[VANDALISM,FOLSOM_ST/SPEAR_ST,NONE,12/31/14]
[NON-CRIMINAL,2800_Block_of_LEAVENWORTH_ST,NONE,12/31/14]
[NON-CRIMINAL,3300_Block_of_WASHINGTON_ST,NONE,12/31/14]
[OTHER_OFFENSES,3300_Block_of_WASHINGTON_ST,NONE,12/31/14]
[LARCENY/THEFT,2000_Block_of_LAGUNA_ST,NONE,12/31/14]
[LARCENY/THEFT,CASTRO_ST/17TH_ST,NONE,12/31/14]
[LARCENY/THEFT,300_Block_of_POWELL_ST,NONE,12/31/14]
[NON-CRIMINAL,TAYLOR_ST/GEARY_ST,NONE,12/31/14]
[NON-CRIMINAL,1600_Block_of_THE_EMBARCADERONORTH_ST,NONE,12/31/14]
[LARCENY/THEFT,BUSH_ST/KEARNY_ST,NONE,12/31/14]
[LARCENY/THEFT,HARRISON_ST/7TH_ST,NONE,12/31/14]
[NON-CRIMINAL,MARKET_ST/DRUMM_ST,NONE,12/31/14]
[NON-CRIMINAL,600_Block_of_DIVISADERO_ST,NONE,12/31/14]
[LARCENY/THEFT,LARKIN_ST/HAYES_ST,NONE,12/31/14]
[ASSAULT,900_Block_of_MARKET_ST,ARREST/BOOKED,12/31/14]
```

Output is truncated to 102400 bytes. Learn more about **ZEPPELIN_INTERPRETER_OUTPUT_LIMIT**

2015년 공공기물 파손 사건 조사

- 2015년도의 공공 기물 파손(VANDALISM) 범죄 유형의 해결 유형 및 범죄 발생 주소 조사

```
///// 2015년도 공공기물 파손 범죄 조사
val van2015 = spark.sql("SELECT category,address,resolution, date FROM sfpd WHERE getyear(date)='15' AND category='VANDALISM'")

// 2015년도 공공기물 파손 범죄 총 건수 표시
van2015.count

// 2015년도 공공기물 파손 범죄 표시
van2015.collect.foreach(println)

// 2015년도 공공기물 파손 범죄의 JSON 파일 형식 저장
van2015.write.format("json").save("/sparkdata/sfpd/output1")
```


2015년 공공기물 파손 사건 조사 – 실행 예 (1)

SPARK JOBS FINISHED

```
///// 2015년도 공공기물 파손 범죄 조사
val van2015 = spark.sql("SELECT category,address,resolution, date FROM sfpd WHERE getyear(date)='15' AND
    category='VANDALISM'")

// 2015년도 공공기물 파손 범죄 총 건수 표시
van2015.count

// 2015년도 공공기물 파손 범죄 표시
van2015.collect.foreach(println)

// 2015년도 공공기물 파손 범죄의 JSON 파일 형식 저장
van2015.write.format("json").mode("overwrite").save("/sparkdata/sfpd/output1")
```

```
van2015: org.apache.spark.sql.DataFrame = [category: string, address: string ... 2 more fields]
res120: Long = 3898
[VANDALISM,1000_Block_of_POLK_ST,ARREST/BOOKED,7/9/15]
[VANDALISM,0_Block_of_VICENTE_ST,NONE,7/9/15]
[VANDALISM,500_Block_of_CASTRO_ST,NONE,7/9/15]
[VANDALISM,3400_Block_of_19TH_ST,NONE,7/9/15]
[VANDALISM,700_Block_of_COLUMBUS_AV,NONE,7/9/15]
[VANDALISM,LINCOLN_WY/22ND_AV,NONE,7/9/15]
[VANDALISM,1700_Block_of_SILLIMAN_ST,NONE,7/9/15]
[VANDALISM,300_Block_of_POST_ST,NONE,7/9/15]
[VANDALISM,GREENWICH_ST/JONES_ST,NONE,7/9/15]
[VANDALISM,3400_Block_of_25TH_ST,NONE,7/9/15]
[VANDALISM,900_Block_of_ELLSWORTH_ST,NONE,7/9/15]
[VANDALISM,800_Block_of_HAYES_ST,NONE,7/9/15]
[VANDALISM,900_Block_of_GEARY_ST,ARREST/BOOKED,7/9/15]
[VANDALISM,0_Block_of_WALLER_ST,NONE,7/9/15]
[VANDALISM,1700_Block_of_SUNNYDALE_AV,NONE,7/9/15]
[VANDALISM,BUSH_ST/WEBSTER_ST,ARREST/BOOKED,7/9/15]
```

Output is truncated to 102400 bytes. Learn more about **ZEPPELIN_INTERPRETER_OUTPUT_LIMIT**

2015년 공공기물 파손 사건 조사 – 실행 예 (2)

```
bigdata@slave1:~$ hadoop fs -ls /sparkdata/sfpd/output1
Found 2 items
-rw-r--r--   3 bigdata supergroup          0 2019-08-02 02:25 /sparkdata/sfpd/output1/_SUCCESS
-rw-r--r--   3 bigdata supergroup     381074 2019-08-02 02:25 /sparkdata/sfpd/output1/part-00000-eed6e
e3c-1e72-43e5-bbce-e2d16240e7bb-c000.json
bigdata@slave1:~$
bigdata@slave1:~$ hadoop fs -cat /sparkdata/sfpd/output1/part-00000-eed6ee3c-1e72-43e5-bbce-e2d16240e7
bb-c000.json | more
{"category":"VANDALISM","address":"1000_Block_of_POLK_ST","resolution":"ARREST/BOOKED","date":"7/9/15"
}
{"category":"VANDALISM","address":"0_Block_of_VICENTE_ST","resolution":"NONE","date":"7/9/15"}
{"category":"VANDALISM","address":"500_Block_of_CASTRO_ST","resolution":"NONE","date":"7/9/15"}
{"category":"VANDALISM","address":"3400_Block_of_19TH_ST","resolution":"NONE","date":"7/9/15"}
{"category":"VANDALISM","address":"700_Block_of_COLUMBUS_AV","resolution":"NONE","date":"7/9/15"}
{"category":"VANDALISM","address":"LINCOLN_WY/22ND_AV","resolution":"NONE","date":"7/9/15"}
{"category":"VANDALISM","address":"1700_Block_of_SILLIMAN_ST","resolution":"NONE","date":"7/9/15"}
{"category":"VANDALISM","address":"300_Block_of_POST_ST","resolution":"NONE","date":"7/9/15"}
{"category":"VANDALISM","address":"GREENWICH_ST/JONES_ST","resolution":"NONE","date":"7/9/15"}
{"category":"VANDALISM","address":"3400_Block_of_25TH_ST","resolution":"NONE","date":"7/9/15"}
{"category":"VANDALISM","address":"900_Block_of_ELLSWORTH_ST","resolution":"NONE","date":"7/9/15"}
{"category":"VANDALISM","address":"800_Block_of_HAYES_ST","resolution":"NONE","date":"7/9/15"}
{"category":"VANDALISM","address":"900_Block_of_GEARY_ST","resolution":"ARREST/BOOKED","date":"7/9/15"
}
{"category":"VANDALISM","address":"0_Block_of_WALLER_ST","resolution":"NONE","date":"7/9/15"}
{"category":"VANDALISM","address":"1700_Block_of_SUNNYDALE_AV","resolution":"NONE","date":"7/9/15"}
```


과제

- 강의 시간의 실습 내용을 정리하여 제출
- 텀 프로젝트 과제
  - 텀 프로젝트 데이터를 사용하여 앞에서 배운 스파크를 적용하고 실행

참고 자료

- 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 3: Apply Operations on Datasets