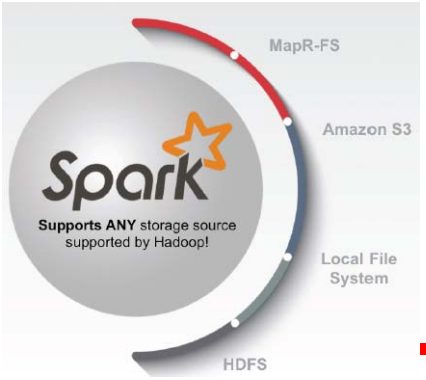


스파크 데이터세트 연산 1

순천향대학교 컴퓨터공학과

이 상 정

학습 내용

1. 데이터세트 연산 소개
2. SFPD 데이터세트 연산
3. 온라인 경매 예

1. 데이터세트 연산 소개

데이터세트 연산

- 데이터세트 상에서는 변환(transformation)가 액션(action)의 두 가지 연산이 수행
  - 변환의 결과로 데이터세트를 리턴
    - 데이터세크는 수정 불가능이므로 새로운 데이터세트를 리턴
  - 액션은 수행된 결과 값을 리턴

  - 데이터세트는 지연 연산
    - 변환은 즉시 수행되지 않고 액션이 호출될 때 수행되어 새로운 데이터세트를 생성

주요 변환 연산

Transformation	Definition
`map()`	Returns new Dataset by applying func to each element of source
`filter()`	Returns new Dataset consisting of elements from source on which function is true
`groupBy()`	Returns Dataset (K, Iterable<V>) where the data is grouped by the given key func.
`reduce()`	Reduces the elements of this Dataset using the specified binary function.
`flatMap()`	Similar to map(), but function should return a sequence rather than a single item
`distinct()`	Returns new Dataset containing distinct elements of source

`cache()`	Cache this Dataset
`persist()`	Persist this DataFrame
`createTempView (viewName)`	Registers this Dataset as a temporary view using the given name
`describe()`	Calculates `count`, `mean`, `sttdev`, `min`, and `max` for numeric columns

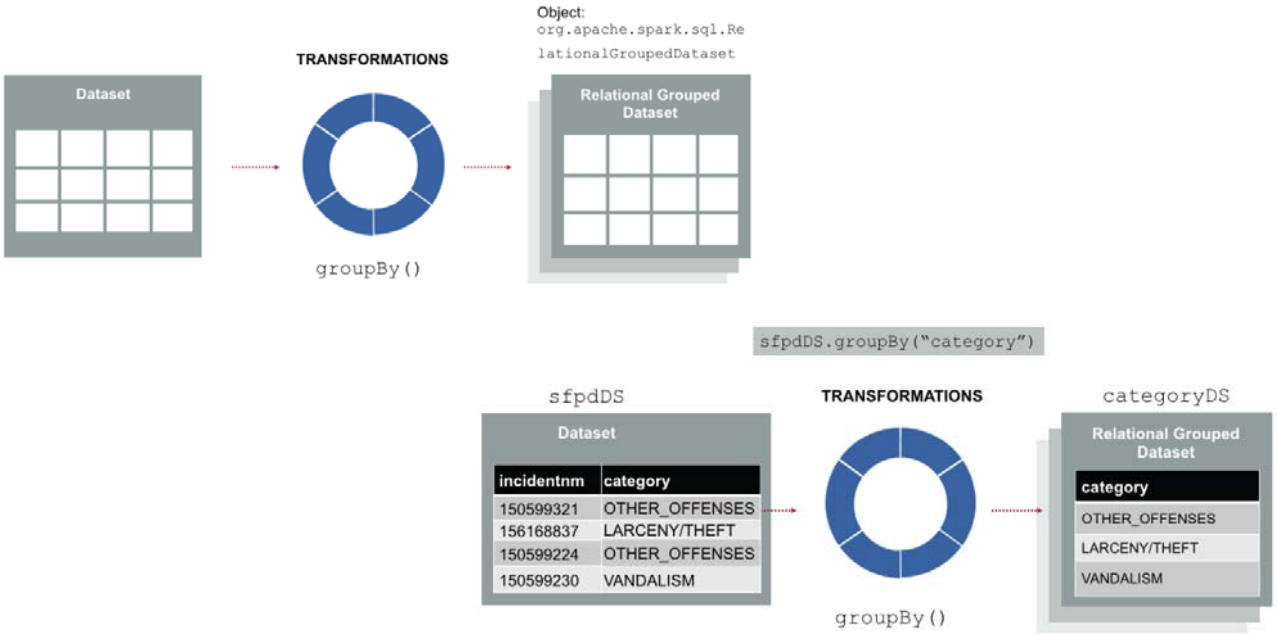
주요 액션

Action	Definition
`count()`	Returns the number of elements in the Dataset
`reduce(func)`	Aggregate elements of Dataset using function **`func`**
`collect()`	Returns all elements of Dataset as an array to driver program
`take(n)`	Returns first *n* elements of Dataset
`show()`	Displays the first 20 rows of DataFrame in tabular form
`first(); head()`	Returns the first row of the Dataset
`takeAsList(n)`	Return first *n* elements of Dataset as list

groupBy() 변환

❑ groupBy() 변환

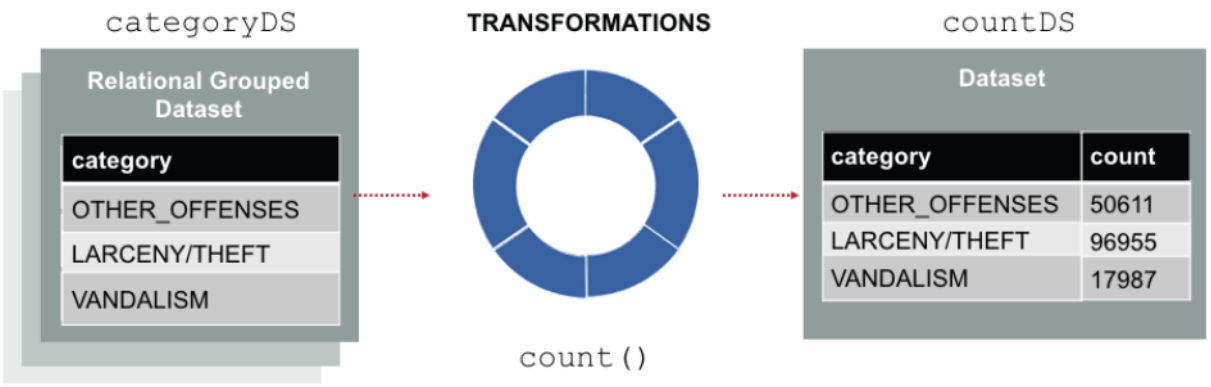
- 데이터세트를 특정 열 기준으로 그룹화하여 관계형 그룹 데이터세트 (Relational Grouped Dataset)를 리턴



count() 액션

❑ count() 액션

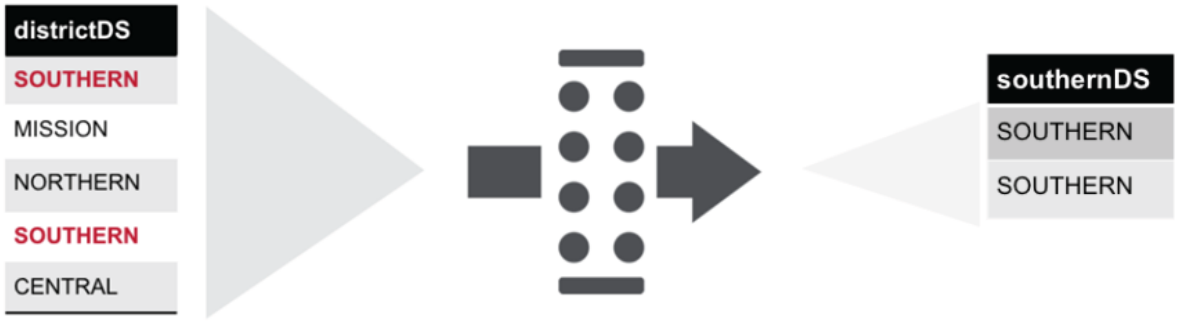
- 변환된 관계형 그룹 데이터세트에 적용하여 카테고리의 갯수 리턴
- show(),collect() 액션으로 결과를 표시



filter() 변환

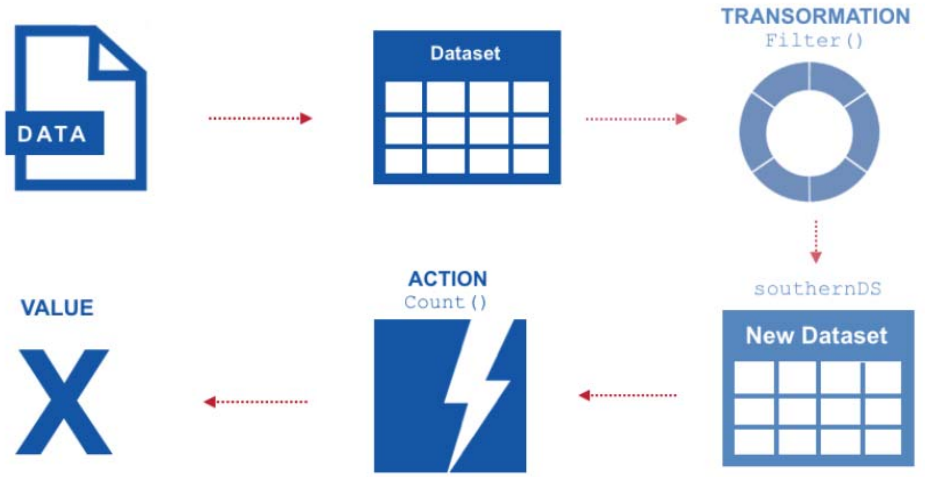
- filter() 변환은 각 요소(행)에 특정 조건을 만족하는 데이터를 리턴

```
val districtDS = sfpdDS.filter(sfpdDS("pddistrict").contains("SOUTHERN"))
```



데이터세트 연산 적용

- 데이터세트 연산 절차
  - 데이터세트 정의
  - 변환 정의
  - 액션 적용
  - 값을 리턴



데이터세트 연산 적용– 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("i
    ncidentnum", "category", "description", "dayofweek", "date", "time",
    "pddistrict", "resolution", "address", "X", "Y", "pdid").as[Incidents]
// 변환 정의
val districtDS = sfpdDS.filter(sfpdDS("pddistrict").contains("SOUTHERN"))
// 변환된 값을 표시
districtDS.show()
// 액션 정의
districtDS.count()
```


데이터세트 연산 적용 – SFPD 예 실행

```
import spark.implicits._

case class Incidents(incidentnum:String, category:String, description:String, dayofweek:String,  date:String, time:String, pddistrict:String, resolution:Strin
    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]
```

SPARK JOBS FINISHED

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

Took 6 sec. Last updated by admin at August 01 2019, 10:41:53 AM.

```
// 변환 정의
val districtDS = sfpdDS.filter(sfpdDS("pddistrict").contains("SOUTHERN"))
// 변환된 값을 표시
districtDS.show()
```

SPARK JOB FINISHED

```
districtDS: org.apache.spark.sql.Dataset[Incidents] = [incidentnum: int, category: string ... 10 more fields]
+-----------+--------------+--------------------+---------+-----+-----+----------+------------+--------------------+-----------+----------+--------------+
|incidentnum|      category|         description|dayofweek| date| time|pddistrict|  resolution|             address|          X|         Y|         pdid|
+-----------+--------------+--------------------+---------+-----+-----+----------+------------+--------------------+-----------+----------+--------------+
|  156169067| LARCENY/THEFT|GRAND_THEFT_FROM_...| Thursday|7/9/15|23:30|  SOUTHERN|        NONE|    8TH_ST/FOLSOM_ST|-122.4100658|37.77499068|15616900000000|
|  156169089| LARCENY/THEFT|GRAND_THEFT_OF_PR...| Thursday|7/9/15|22:00|  SOUTHERN|        NONE|    7TH_ST/FOLSOM_ST|-122.4078441|37.77674622|15616900000000|
|  156169772| LARCENY/THEFT|GRAND_THEFT_OF_PR...| Thursday|7/9/15|22:00|  SOUTHERN|        NONE|   BRANNAN_ST/4TH_ST|-122.3965332| 37.7783269|15617000000000|
|  150599127|SUSPICIOUS_OCC|INVESTIGATIVE_DET...| Thursday|7/9/15|21:32|  SOUTHERN|        NONE|800_Block_of_BRYA...|-122.4034048|37.77542071|15059900000000|
|  156169926| LARCENY/THEFT|GRAND_THEFT_FROM_...| Thursday|7/9/15|21:00|  SOUTHERN|        NONE|1700_Block_of_ALA...|-122.4024486|37.76862184|15617000000000|
|  150599337| LARCENY/THEFT|GRAND_THEFT_FROM_...| Thursday|7/9/15|20:25|  SOUTHERN|        NONE|100_Block_of_THE_...|-122.3946701|37.79533119|15059900000000|
|  150598765|       ROBBERY|ROBBERY_ON_THE_ST...| Thursday|7/9/15|20:17|  SOUTHERN|        NONE|1400_Block_of_MAR...|-122.4180253|37.77598197|15059900000000|
```

```
// 액션 정의
districtDS.count()
```

SPARK JOB FINISHED

```
res36: Long = 73308
```

Took 3 sec. Last updated by admin at August 01 2019, 10:41:57 AM.

데이터세트 연산 적용 – 단어 카운트 예 (1)

- 앞의 맵리듀스의 단어 카운트 예
- 입력 데이터의 로컬 파일 복사
  - 로컬파일: ~/hadoop-2.7.7/*.txt
  - HDFS 파일 디렉토리 : /WordCount/input

```
$ hadoop fs -mkdir /sparkdata/word
$ hadoop fs -copyFromLocal ~/hadoop-2.7.7/*.txt  /sparkdata/word
```

```
bigdata@slave1:~$ hadoop fs -mkdir /sparkdata/word
bigdata@slave1:~$ hadoop fs -copyFromLocal ~/hadoop-2.7.7/*.txt /sparkdata/word
bigdata@slave1:~$ hadoop fs -ls /sparkdata/word
Found 3 items
-rw-r--r--   3 bigdata supergroup      86424 2019-08-01 01:53 /sparkdata/word/LICENSE.txt
-rw-r--r--   3 bigdata supergroup      14978 2019-08-01 01:53 /sparkdata/word/NOTICE.txt
-rw-r--r--   3 bigdata supergroup       1366 2019-08-01 01:53 /sparkdata/word/README.txt
bigdata@slave1:~$
```


데이터세트 연산 적용 – 단어 카운트 예 (2)

```
import spark.implicits._

// 문자열 타입의 라인으로 구성된 데이터 세트 정의
val dataDS = spark.read.text("/sparkdata/word").as[String]
dataDS.count()
dataDS.show()

// 공백문자로 구분하여 단어 분리
val wordsDS = dataDS.flatMap(value => value.split("₩₩s+"))
wordsDS.printSchema()
wordsDS.count()
wordsDS.show()

// 각 단어를 그룹화
val groupDS = wordsDS.groupBy("value")
// 그룹(단어)을 카운트한 후 표시
val counts = groupDS.count()
counts.show()
```

데이터세트 연산 적용 – 단어 카운트 예 실행 (1)

```
import spark.implicits._

// 문자열 타입의 라인으로 구성된 데이터 세트 정의
val dataDS = spark.read.text("/sparkdata/word").as[String]
dataDS.count()
dataDS.show()
```

```
import spark.implicits._
dataDS: org.apache.spark.sql.Dataset[String] = [value: string]
res143: Long = 2062
+--------------------+
|               value|
+--------------------+
|                    |
|                 ...|
|                 ...|
|                 ...|
|                    |
|   TERMS AND COND...|
|                    |
|     1. Definitions.|
|                    |
|      "License" s...|
|      and distrib...|
|                    |
```

Took 4 sec. Last updated by admin at August 01 2019, 12:21:17 PM. (outdated)

데이터세트 연산 적용 – 단어 카운트 예 실행 (2)

```
// 공백문자로 구분하여 단어 분리
val wordsDS = dataDS.flatMap(value => value.split("\\s+"))
wordsDS.printSchema()
wordsDS.count()
wordsDS.show()
```

```
wordsDS: org.apache.spark.sql.Dataset[String] = [value: string]
root
 |-- value: string (nullable = true)

res148: Long = 15284
+--------------------+
|               value|
+--------------------+
|                    |
|                    |
|              Apache|
|             License|
|                    |
|             Version|
|                2.0,|
|             January|
|                2004|
|                    |
```

Took 2 sec. Last updated by admin at August 01 2019, 12:21:19 PM. (outdated)

데이터세트 연산 적용 – 단어 카운트 예 실행 (3)

```
// 각 단어를 그룹화
val groupDS = wordsDS.groupBy("value")

// 그룹(단어)을 카운트한 후 표시
val counts = groupDS.count()
counts.show()
```

```
groupDS: org.apache.spark.sql.RelationalGroupedDataset = RelationalGroupedDataset: [grouping expre
counts: org.apache.spark.sql.DataFrame = [value: string, count: bigint]
+--------------------+-----+
|               value|count|
+--------------------+-----+
|               those|    9|
|            brackets|    1|
|              Group.|    1|
|          unmodified|    2|
| 252.227-7014(a)(1))|    1|
|           exchange.|    2|
|               input|    2|
|                  By|    1|
|                (New|    3|
|license/LICENSE.c...|    1|
|StringUtils.conta...|    1|
|       TypeUtil.java|    1|
|           "License"|    1|
```

Took 3 sec. Last updated by admin at August 01 2019, 12:21:22 PM. (outdated)

2. SFPD 데이터세트 연산

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 Top 5 주소 – 스칼라

- 가장 사건이 많이 발생한 5개의 주소(address)는?
  - 데이터세트 생성
    - 주소 별로 사건을 그룹화
  - 각 주소 별로 사건의 수를 카운트
  - 내림차순으로 정렬
  - 정렬된 처음 5개 주소를 표시

```
// 데이터세트 생성
val incByAddDS = sfpdDS.groupBy("address")
// 각 주소 별로 사건의 수를 카운트
val numAddDS = incByAddDS.count
// 내림차순으로 정렬
val numAddDesc = numAddDS.sort($"count".desc)
// 정렬된 처음 5개 주소를 표시
numAddDesc.show(5)
numAddDesc.take(5)
```

```
// 한 문장으로 결합
val incByAdd = sfpdDS.groupBy("address").count.sort($"count".desc).show(5)
```

SFPD Top 5 주소 – 스칼라 실행 예

SPARK JOBS FINISHED

```
///// 가장 사건이 많이 발생한 5개의 주소(address)는
// 데이터세트 생성
val incByAddDS = sfpdDS.groupBy("address")
// 각 주소 별로 사건의 수를 카운트
val numAddDS = incByAddDS.count
// 내림차순으로 정렬
val numAddDesc = numAddDS.sort($"count".desc)
// 정렬된 처음 5개 주소를 표시
numAddDesc.show(5)
numAddDesc.take(5)
```

```
incByAddDS: org.apache.spark.sql.RelationalGroupedDataset = RelationalGroupedDataset: [grouping expressions: [address: string], value:
[incidentnum: int, category: string ... 10 more fields], type: GroupBy]
numAddDS: org.apache.spark.sql.DataFrame = [address: string, count: bigint]
numAddDesc: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [address: string, count: bigint]
+--------------------+-----+
|             address|count|
+--------------------+-----+
|800_Block_of_BRYA...|10852|
|800_Block_of_MARK...| 3671|
|1000_Block_of_POT...| 2027|
|2000_Block_of_MIS...| 1585|
|  16TH_ST/MISSION_ST| 1512|
+--------------------+-----+
only showing top 5 rows
```

```
res198: Array[org.apache.spark.sql.Row] = Array([800_Block_of_BRYANT_ST,10852], [800_Block_of_MARKET_ST,3671], [1000_Block_of_POTRERO_A
V,2027], [2000_Block_of_MISSION_ST,1585], [16TH_ST/MISSION_ST,1512])
```


참고 – 데이터프레임의 열(column) 참조 표시

- 데이터프레임의 열의 참조 표시 방법
  - col("열 이름")
  - $"열 이름"
  - 데이터세트("열 이름")
  - 스파크 API Docs 참조
    - https://spark.apache.org/docs/latest/api/scala/index.html#package

A column that will be computed based on the data in a DataFrame.

A new column can be constructed based on the input columns present in a DataFrame:

```
df("columnName")            // On a specific `df` DataFrame.
col("columnName")           // A generic column no yet associated with a DataFrame.
col("columnName.field")     // Extracting a struct field
col("`a.column.with.dots`") // Escape `.` in column names.
$"columnName"               // Scala short hand for a named column.
```

Column objects can be composed to form complex expressions:

```
$"a" + 1
$"a" === $"b"
```

SFPD Top 5 주소 – SQL

- SQL을 사용하여 질의
  - 테이블로 등록된 sfpd 뷰를 사용하여 질의
  - 뷰 등록 시 createOrReplaceTempView() 메서드 사용
    - 재 실행 시에도 새로이 등록
    - createTempView() 메서드는 이미 등록된 경우 에러 발생

```
///// 가장 사건이 많이 발생한 5개의 주소(address), SQL
// 데이터세트를 뷰(view)로 등록
sfpdDS.createOrReplaceTempView("sfpd")

// SQL 버전 질의
val topByAddressSQL = spark.sql("SELECT address,  count(incidentnum) AS inccount FROM sfpd GROUP BY address ORDER BY inccount DESC LIMIT 5")
topByAddressSQL.show()
```


SFPD Top 5 주소 – SQL 실행 예

SPARK JOB FINISHED

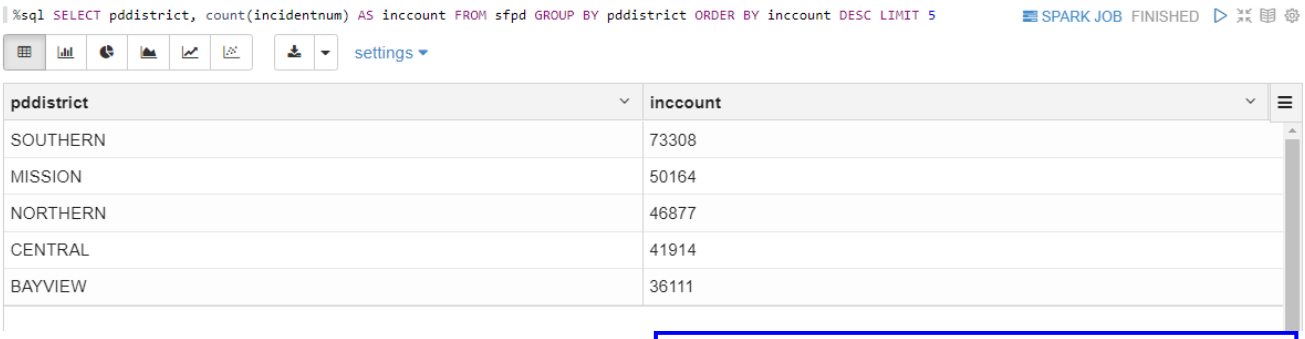
```
///// 가장 사건이 많이 발생한 5개의 주소(address), SQL
// 데이터세트를 뷰(view)로 등록
sfpdDS.createOrReplaceTempView("sfpd")

// SQL 버전 질의
val topByAddressSQL = spark.sql("SELECT address,  count(incidentnum) AS inccount FROM sfpd GROUP BY address ORDER BY inccount
    DESC LIMIT 5")
topByAddressSQL.show()

topByAddressSQL: org.apache.spark.sql.DataFrame = [address: string, inccount: bigint]
+--------------------+--------+
|             address|inccount|
+--------------------+--------+
|800_Block_of_BRYA...|   10852|
|800_Block_of_MARK...|    3671|
|1000_Block_of_POT...|    2027|
|2000_Block_of_MIS...|    1585|
|  16TH_ST/MISSION_ST|    1512|
+--------------------+--------+
```

Took 5 sec. Last updated by admin at August 01 2019, 2:34:20 PM. (outdated)

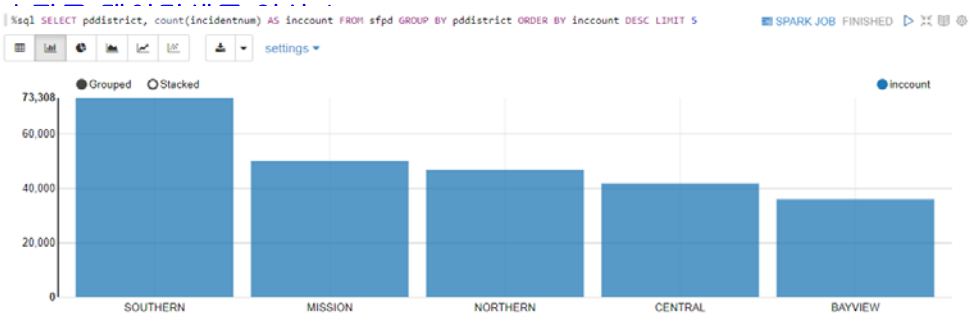
SFPD Top 5 주소 – 제플린 차트

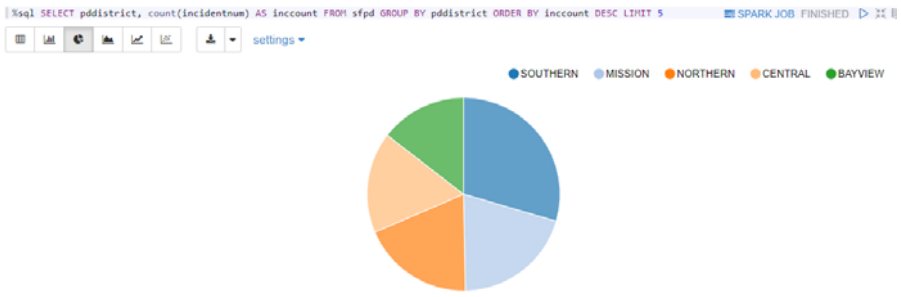
- 제플린은 스파크 SQL을 사용하여 데이터를 시각화하는 기능 제공
  - %sql 로 시작하는 SQL을 기술하면 테이블 및 다양한 차트 표시

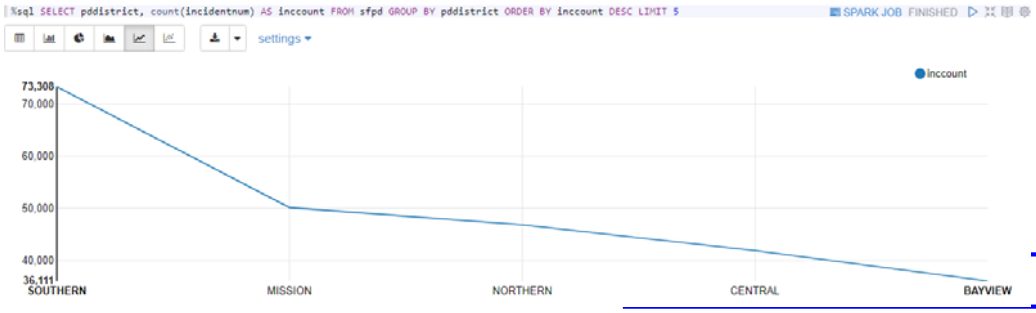
%sql SELECT pddistrict, count(incidentnum) AS inccount FROM sfpd GROUP BY pddistrict ORDER BY inccount DESC LIMIT 5



pddistrict	inccount
SOUTHERN	73308
MISSION	50164
NORTHERN	46877
CENTRAL	41914
BAYVIEW	36111







질의 결과 저장 – save()

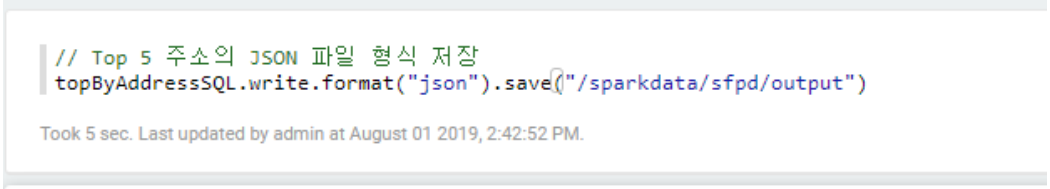
- save() 연산을 사용하여 질의 결과를 저장

Operation	Description
save (source, mode, options)	Saves contents of Dataset based on given data sources, savemode, and set of options
jdbc (url,name, overwrite)	Saves contents of Dataset to JDBC at URL under table name table
parquet (path)	Saves contents of Dataset as parquet file
saveAsTable (tablename, source, mode, options)	Creates a table from contents of Dataset using data source, options, and mode

SFPD Top 5 주소 – JSON 저장

- 가장 사건이 많이 발생한 5개의 주소 결과를 JSON 파일 형식으로 저장
  - write() 메서드로 출력
  - format() 메서드로 저장 양식 지정
  - save() 메서드로 저장
    - 인수로 저장되는 디렉토리 기술

```
// Top 5 주소의 JSON 파일 형식 저장
topByAddressSQL.write.format("json").save("/sparkdata/sfpd/output")
```



SFPD Top 5 주소 – JSON 저장 실행 결과

```
bigdata@slave1:~$ hadoop fs -ls /sparkdata/sfpd
Found 2 items
drwxr-xr-x   - bigdata supergroup          0 2019-08-01 05:42 /sparkdata/sfpd/output
-rw-r--r--   3 bigdata supergroup   57772855 2019-07-18 02:46 /sparkdata/sfpd/sfpd.csv
bigdata@slave1:~$ hadoop fs -ls /sparkdata/sfpd/output
Found 2 items
-rw-r--r--   3 bigdata supergroup          0 2019-08-01 05:42 /sparkdata/sfpd/output/_SUCCESS
-rw-r--r--   3 bigdata supergroup        266 2019-08-01 05:42 /sparkdata/sfpd/output/part-00000-fcf5a4
c9-dae5-4a31-875c-ca2ba15927d3-c000.json
bigdata@slave1:~$
bigdata@slave1:~$ hadoop fs -cat /sparkdata/sfpd/output/part-00000-fcf5a4c9-dae5-4a31-875c-ca2ba15927d
3-c000.json
{"address":"800_Block_of_BRYANT_ST","inccount":10852}
{"address":"800_Block_of_MARKET_ST","inccount":3671}
{"address":"1000_Block_of_POTRERO_AV","inccount":2027}
{"address":"2000_Block_of_MISSION_ST","inccount":1585}
{"address":"16TH_ST/MISSION_ST","inccount":1512}
bigdata@slave1:~$
```


SFPD Top 5 주소 – JSON 읽기

- 저장된 JSON 형식의 데이터프레임 읽기
  - spark.read.json("/sparkdata/sfpd/output")

```
// Top 5 주소의 JSON 파일 형식 읽기
val topByAddress5 = spark.read.json("/sparkdata/sfpd/output")
topByAddress5.show()
```

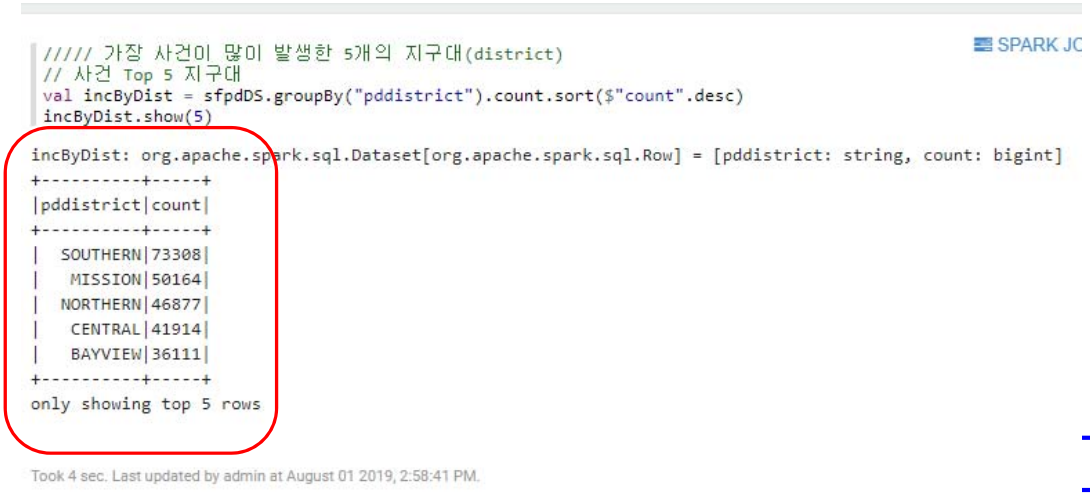
```
topByAddress5: org.apache.spark.sql.DataFrame = [address: string, inccount: bigint]
+--------------------+--------+
|             address|inccount|
+--------------------+--------+
|800_Block_of_BRYA...|   10852|
|800_Block_of_MARK...|    3671|
|1000_Block_of_POT...|    2027|
|2000_Block_of_MIS...|    1585|
|  16TH_ST/MISSION_ST|    1512|
+--------------------+--------+
```

Took 2 sec. Last updated by admin at August 01 2019, 2:52:13 PM. (outdated)

SFPD 사건 Top 5 지구대 – 스칼라

- 가장 사건이 많이 발생한 5개의 지구대(district)는?

```
///// 가장 사건이 많이 발생한 5개의 지구대(district)
// 사건 Top 5 지구대
val incByDist = sfpdDS.groupBy("pddistrict").count.sort($"count".desc)
incByDist.show(5)
```

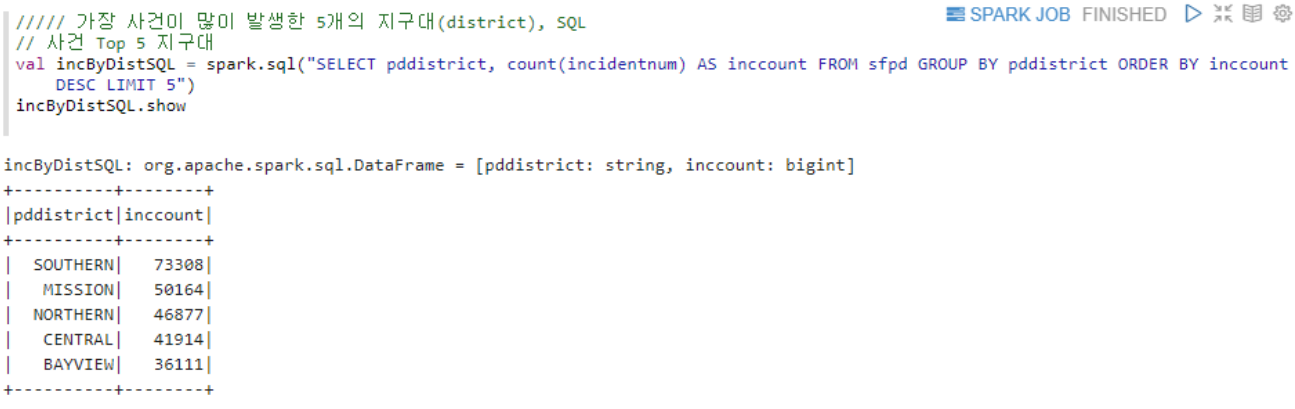


```
///// 가장 사건이 많이 발생한 5개의 지구대(district)
// 사건 Top 5 지구대
val incByDist = sfpdDS.groupBy("pddistrict").count.sort($"count".desc)
incByDist.show(5)

incByDist: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [pddistrict: string, count: bigint]
+----------+-----+
|pddistrict|count|
+----------+-----+
|  SOUTHERN|73308|
|   MISSION|50164|
|  NORTHERN|46877|
|   CENTRAL|41914|
|   BAYVIEW|36111|
+----------+-----+
only showing top 5 rows

Took 4 sec. Last updated by admin at August 01 2019, 2:58:41 PM.
```


SFPD 사건 Top 5 지구대 – SQL

- 가장 사건이 많이 발생한 5개의 지구대(district)는?

```
///// 가장 사건이 많이 발생한 5개의 지구대(district), SQL
// 사건 Top 5 지구대
val incByDistSQL = spark.sql("SELECT pddistrict, count(incidentnum) AS inccount
FROM sfpd GROUP BY pddistrict ORDER BY inccount DESC LIMIT 5")
incByDistSQL.show
```



```
///// 가장 사건이 많이 발생한 5개의 지구대(district), SQL
// 사건 Top 5 지구대
val incByDistSQL = spark.sql("SELECT pddistrict, count(incidentnum) AS inccount FROM sfpd GROUP BY pddistrict ORDER BY inccount
    DESC LIMIT 5")
incByDistSQL.show

incByDistSQL: org.apache.spark.sql.DataFrame = [pddistrict: string, inccount: bigint]
+----------+--------+
|pddistrict|inccount|
+----------+--------+
|  SOUTHERN|   73308|
|   MISSION|   50164|
|  NORTHERN|   46877|
|   CENTRAL|   41914|
|   BAYVIEW|   36111|
+----------+--------+
```

SFPD Top 10 사건 해결 – 스칼라

- 가장 많은 10개의 사건 해결 유형(resolution)은?

```
///// 가장 많은 10개의 사건 해결 유형(resolution)
// Top 10 사건 해결
val top10Res = sfpdDS.groupBy("resolution").count.sort($"count".desc)
top10Res.show(10)
```

SPARK

```
val top10Res = sfpdDS.groupBy("resolution").count.sort($"count".desc)
top10Res.show(10)

top10Res: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [resolution: string, count: bigint]
+--------------------+------+
|          resolution| count|
+--------------------+------+
|                NONE|243538|
|       ARREST/BOOKED| 86766|
|        ARREST/CITED| 22925|
|   PSYCHOPATHIC_CASE|  8344|
|             LOCATED|  6878|
|           UNFOUNDED|  4551|
|COMPLAINANT_REFUS...|  4215|
|     JUVENILE_BOOKED|  2381|
|EXCEPTIONAL_CLEAR...|  1134|
|      JUVENILE_CITED|  1039|
+--------------------+------+
only showing top 10 rows
```


SFPD Top 10 사건 해결 – SQL

- 가장 많은 10개의 사건 해결 유형(resolution)은?

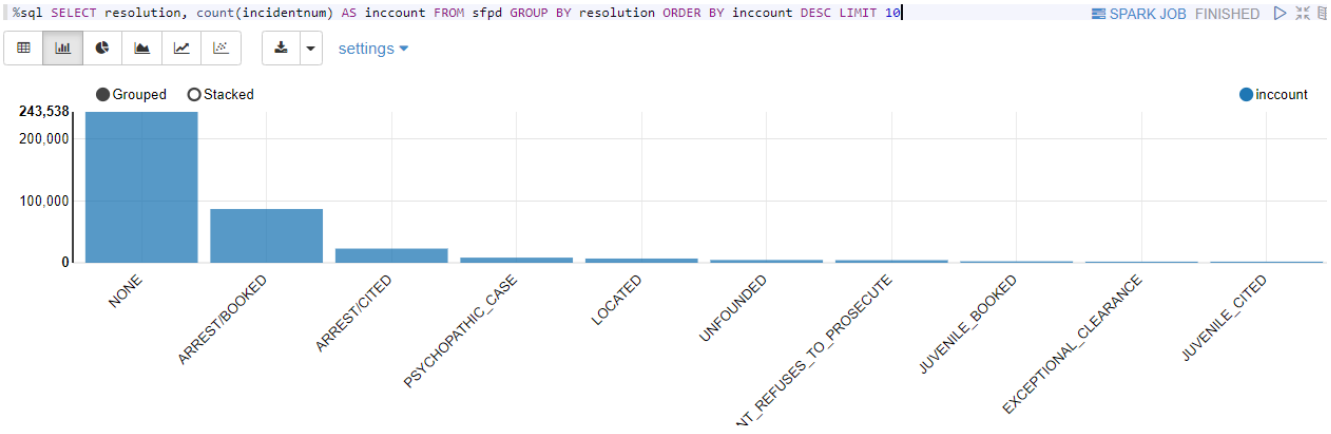
```
///// 가장 많은 10개의 사건 해결 유형(resolution), SQL
// Top 10 사건 해결, SQL
val top10ResSQL = spark.sql("SELECT resolution, count(incidentnum) AS inccount
FROM sfpd GROUP BY resolution ORDER BY inccount DESC LIMIT 10")
top10ResSQL.show
```

SPARK JOB FINISHED

```
///// 가장 많은 10개의 사건 해결 유형(resolution), SQL
// Top 10 사건 해결, SQL
val top10ResSQL = spark.sql("SELECT resolution, count(incidentnum) AS inccount FROM sfpd GROUP BY resolution ORDER BY
    inccount DESC LIMIT 10")
top10ResSQL.show

top10ResSQL: org.apache.spark.sql.DataFrame = [resolution: string, inccount: bigint]
+--------------------+--------+
|          resolution|inccount|
+--------------------+--------+
|                NONE|  243538|
|       ARREST/BOOKED|   86766|
|        ARREST/CITED|   22925|
|   PSYCHOPATHIC_CASE|    8344|
|             LOCATED|    6878|
|           UNFOUNDED|    4551|
|COMPLAINANT_REFUS...|    4215|
|     JUVENILE_BOOKED|    2381|
|EXCEPTIONAL_CLEAR...|    1134|
|      JUVENILE_CITED|    1039|
+--------------------+--------+
```

SFPD Top 10 사건 해결 – 제플린 차트

%sql SELECT resolution, count(incidentnum) AS inccount FROM sfpd GROUP BY resoluti
on ORDER BY inccount DESC LIMIT 10



SFPD Top 3 범죄 유형 – 스칼라

- 가장 많은 3개의 범죄 유형(category)은?

```
///// 가장 많은 3개의 범죄 유형(category)
// Top 10 범죄 유형
val top3Cat = sfpdDS.groupBy("category").count.sort($"count".desc).show(3)
```

```
///// 가장 많은 3개의 범죄 유형(category)
// Top 10 범죄 유형
val top3Cat = sfpdDS.groupBy("category").count.sort($"count".desc).show(3)
```

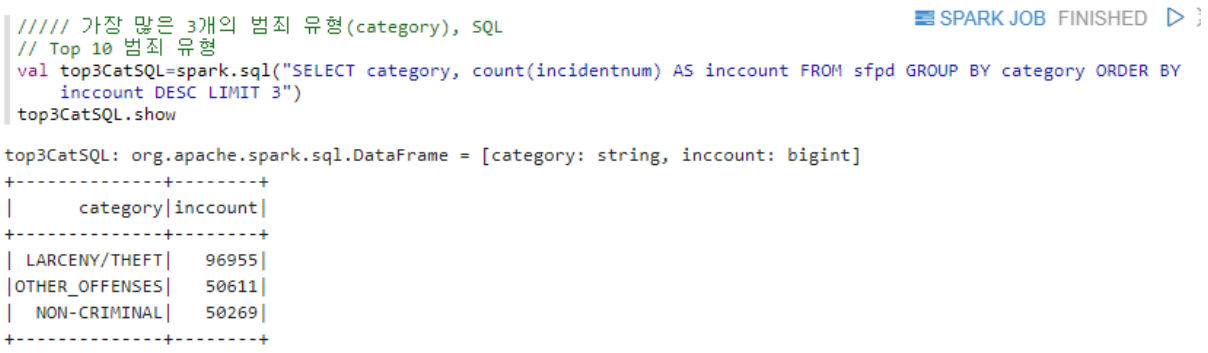
```
+--------------+-----+
|      category|count|
+--------------+-----+
| LARCENY/THEFT|96955|
|OTHER_OFFENSES|50611|
|  NON-CRIMINAL|50269|
+--------------+-----+
only showing top 3 rows

top3Cat: Unit = ()
```

Took 4 sec. Last updated by admin at August 01 2019, 3:08:13 PM.

SFPD Top 3 범죄 유형 – SQL

- 가장 많은 3개의 범죄 유형(category)은?

```
///// 가장 많은 3개의 범죄 유형(category), SQL
// Top 10 범죄 유형
val top3CatSQL=spark.sql("SELECT category, count(incidentnum) AS inccount FROM sfpd GROUP BY category ORDER BY inccount DESC LIMIT 3")
top3CatSQL.show
```



3. 온라인 경매 예

온라인 경매 데이터 소개

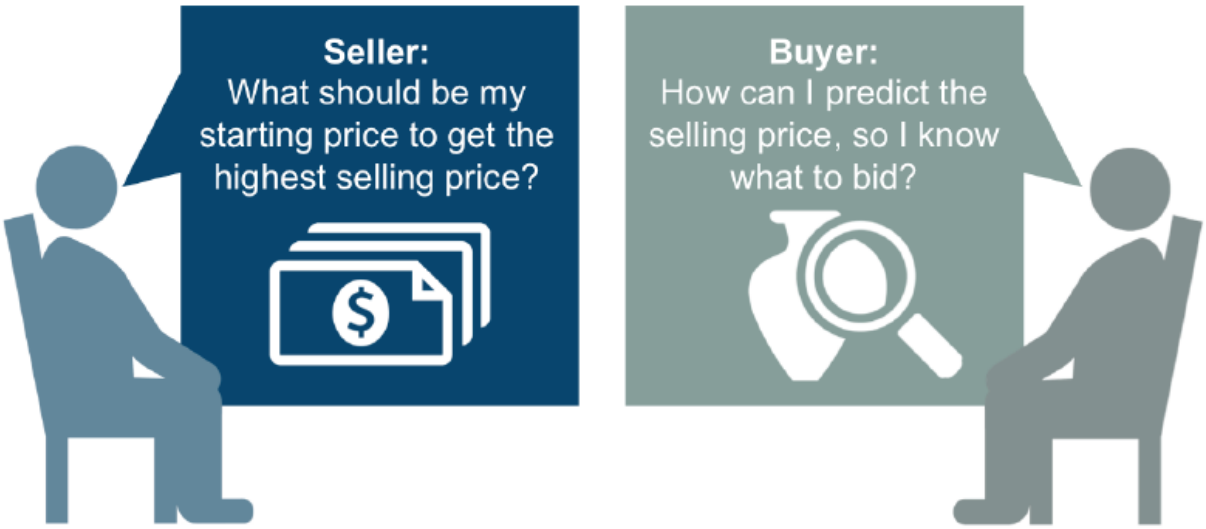
- 로컬 파일 시스템에 CSV 형식으로 저장된 온라인 경매 데이터 예
  - 3가지 유형의 상품에 대한 경매 데이터: xbox, 카르티에(cartier), 팜(palm)
  - 각 행의 개별 경매 응찰 데이터
    - 경매 ID(경매 물건 구분), 응찰 가격, 응찰 시간(경매 시작에서부터의 날수), 응찰자 ID, 응찰자 수, 최초 오픈 가격, 최종 응찰 가격, 상품 유형, 남은 경매 일수

AuctionID	Bid	Bid Time	Bidder	Bidder Rate	Open Bid	Price	Item	Days To Live
8213034705	95	2.927373	jake7870	0	95	117.5	xbox	3
8213034705	115	2.943484	davidbresler2	1	95	117.5	xbox	3
8213034705	100	2.951285	gladimacowgirl	58	95	117.5	xbox	3
8213034705	117.5	2.998947	daysrus	95	95	117.5	xbox	3

온라인 경매 데이터 타입

Column	Type	Description
aucid	String	Auction ID
bid	Float	Bid amount
bidtime	Float	Time of bid from start of auction
bidder	String	The bidder's userid
Bidrate	Int	The bidder's rating
openbid	Float	Opening price
Price	Float	Final price
Itemtype	String	Item type
dtl	Int	Days to live

온라인 경매 데이터 활용



온라인 경매 데이터 다운로드

- 디렉토리 생성 및 다운로드
  - auction 디렉토리 생성

    $ mkdir ~/spark/auction

    $ cd ~/spark/auction
  - 강의 홈페이지에서 다운로드, auctiondata.csv

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

```
bigdata@slave1:~/spark/auction$ wget http://cs.sch.ac.kr/lecture/BigData/download/auctiondata.csv
--2019-07-22 01:10:56--  http://cs.sch.ac.kr/lecture/BigData/download/auctiondata.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: 575014 (562K) [text/csv]
Saving to:  auctiondata.csv'

auctiondata.csv                 100%[=============================================================>] 561.54K

2019-07-22 01:10:56 (6.62 MB/s) -  auctiondata.csv' saved [575014/575014]

bigdata@slave1:~/spark/auction$ more auctiondata.csv
8213034705,95,2.927373,jake7870,0,95,117.5,xbox,3
8213034705,115,2.943484,davidbresler2,1,95,117.5,xbox,3
8213034705,100,2.951285,gladimacowgirl,58,95,117.5,xbox,3
8213034705,117.5,2.998947,daysrus,10,95,117.5,xbox,3
8213060420,2,0.065266,donnie4814,5,1,120,xbox,3
```

온라인 경매 데이터 적재

- 입력 데이터 로컬 파일 auctiondata.csv를 하둡 파일 시스템의 파일로 복사
  - 로컬 파일: ~/spark/auction/auctiondata.csv
  - 하둡 파일: /sparkdata/auction/auctiondata.csv

```
$ hadoop fs -mkdir /sparkdata/auction
$ hadoop fs -put auctiondata.csv /sparkdata/auction
```

```
bigdata@slave1:~/spark/auction$ hadoop fs -mkdir /sparkdata/auction
bigdata@slave1:~/spark/auction$ hadoop fs -put auctiondata.csv /sparkdata/auction
bigdata@slave1:~/spark/auction$ hadoop fs -ls /sparkdata/auction
Found 1 items
-rw-r--r--   3 bigdata supergroup     575014 2019-07-22 01:17 /sparkdata/auction/auctiondata.csv
bigdata@slave1:~/spark/auction$
```


온라인 경매 데이터세트 생성 – 코드

```
///// 온라인 경매 데이터세트 생성
// 클래스 임포트
import spark.implicits._

// 케이스 클래스 정의
case class Auctions(aucid:String, bid:Double, bidtime:Double, bidder:String,
    bidrate:Int, openbid:Double, price:Double, itemtype:String, dtl:Int)

// 데이터 적재 후 데이터세트 생성
val auctionsDS =
    spark.read.option("inferSchema",true).csv("/sparkdata/auction/auctiondata.
    csv").toDF("aucid", "bid", "bidtime", "bidder", "bidrate", "openbid", "price",
    "itemtype", "dtl").as[Auctions]
auctionsDS.printSchema()          // 스키마 프린트

// 데이터세트를 뷰(view)로 등록
auctionsDS.createOrReplaceTempView("auctions")
```

온라인 경매 데이터세트 생성 – 실행

SPARK JOBS FINISHED

```
///// 온라인 경매 데이터세트 생성
// 클래스 임포트
import spark.implicits._

// 케이스 클래스 정의
case class Auctions(aucid:String, bid:Double, bidtime:Double, bidder:String, bidrate:Int, openbid:Double, price
    :Double, itemtype:String, dtl:Int)

// 데이터 적재 후 데이터세트 생성
val auctionsDS = spark.read.option("inferSchema",true).csv("/sparkdata/auction/auctiondata.csv").toDF("aucid", "bid",
    "bidtime", "bidder", "bidrate", "openbid", "price", "itemtype", "dtl").as[Auctions]
auctionsDS.printSchema()              // 스키마 프린트

// 데이터세트를 뷰(view)로 등록
auctionsDS.createOrReplaceTempView("auctions")
```

```
import spark.implicits._
defined class Auctions
auctionsDS: org.apache.spark.sql.Dataset[Auctions] = [aucid: bigint, bid: double ... 7 more fields]
root
 |-- aucid: long (nullable = true)
 |-- bid: double (nullable = true)
 |-- bidtime: double (nullable = true)
 |-- bidder: string (nullable = true)
 |-- bidrate: integer (nullable = true)
 |-- openbid: double (nullable = true)
 |-- price: double (nullable = true)
 |-- itemtype: string (nullable = true)
 |-- dtl: integer (nullable = true)
```

Took 6 sec. Last updated by admin at August 01 2019, 3:38:54 PM.

온라인 경매 질문 조사 사항

- 온라인 경매 데이터에 대해 다음 질문을 조사
  1. 얼마나 많은 상품이 팔렸는가?
  2. 얼마나 많은 상품들 유형이 있는가?
  3. 상품 유형당 얼마나 많이 응찰에 참여했는가?
  4. 경매 상품 당 응찰 자 수는?
  5. 경매 당 최대, 최소, 평균 응찰자 수는?
  6. 경매의 상품 유형당 최대, 최소, 평균 입찰 가격은?
  7. 낙찰가가 200을 넘는 경매의 수는?
  8. xbox의 모든 경매에 대해 기본 통계

SQL 질의 변환 함수

❑ SQL 질의 변환 함수

- SQL 질의 형식을 사용한 변환 함수

Transformation	Definition
`agg(expr, exprs)`	Aggregates on the entire Dataset without groups
`filter(condition Expr)`	Filters based on given SQL expression
`groupBy(col1, cols)`	Groups Dataset using the specified columns so we can run aggregation on them
`select(cols)`	Selects a set of columns based on expressions

온라인 경매 데이터세트 연산 (1)

❑ 온라인 경매 데이터세트에 대해 다음 질문을 조사

1. 얼마나 많은 상품이 팔렸는가?
   - auctionid가 경매로 판매할 물건(상품)을 표시
   - distinct 변환은 중복되지 않는 고유한 auctionid의 데이터세트를 리턴

```
// 경매 상품 판매량
val totalauctions = auctionsDS.select("aucid").distinct.count()
```

2. 얼마나 많은 상품들 유형이 있는가?
   - itemtype이 상품 유형 표시

```
// 경매 상품 유형
val itemtypes = auctionsDS.select("itemtype").distinct.count()
```

온라인 경매 데이터세트 연산 (2)

3. 상품 유형당 얼마나 많은 응찰자 수?
   - itemtype이 각 상품 유형을 표시

```
// 상품 유형 당 응찰자 수
auctionsDS.groupBy("itemtype").count().show()
```

4. 경매 상품 당 응찰 자 수는?
   - aucid가 경매로 판매할 상품을 표시

```
// 상품 유형 당 응찰자 수
auctionsDS.groupBy("aucid").count().show()
```


온라인 경매 데이터세트 연산 – 실행 예 (1)

```
// 경매 상품 판매량
val totalauctions = auctionsDS.select("aucid").distinct.count()
```

```
totalauctions: Long = 627
```

```
// 경매 상품 유형
val itemtypes = auctionsDS.select("itemtype").distinct.count()
```

```
itemtypes: Long = 3
```

```
// 상품 유형 당 응찰자 수
auctionsDS.groupBy("itemtype").count().show()
```

```
+--------+-----+
|itemtype|count|
+--------+-----+
| cartier| 1953|
|    palm| 5917|
|    xbox| 2784|
+--------+-----+
```

```
// 상품 유형 당 응찰자 수
auctionsDS.groupBy("aucid").count().show()
```

```
+----------+-----+
|     aucid|count|
+----------+-----+
|8212964295|   11|
|8213082427|   26|
|1642534283|   23|
|3024823511|   45|
|1645542737|    8|
|1644077790|   15|
|3018375809|   16|
|8212896511|    2|
|3013787547|   18|
|8212610170|   20|
|1644724061|   14|
|3015958025|   20|
|3017907666|   31|
|3015011363|   27|
|3015909534|    8|
```

온라인 경매 데이터세트 연산 (3)

5. 경매 당 최대, 최소, 평균 응찰자 수는?
   - agg() 함수 적용

```
// 경매 당 최대, 최소, 평균 응찰자 수
auctionsDS.groupBy("aucid").count.agg(min("count"),avg("count"),max("count")
).show()
```

6. 경매의 상품 유형당 최대, 최소, 평균 입찰 가격은?

```
// 상품 유형당 최대, 최소, 평균 입찰 가격
auctionsDS.groupBy("itemtype", "aucid").agg(min("bid"), max("bid"),
avg("bid")).show()
```

7. 낙찰가가 200을 넘는 경매의 수는?

```
// 낙찰가가 200을 넘는 경매의 수
auctionsDS.filter(auctionsDS("price")>200).count()
```


온라인 경매 데이터세트 연산 – 실행 예 (2)

```
// 경매 당 최대, 최소, 평균 응찰자 수
auctionsDS.groupBy("aucid").count.agg(min("count"),avg("count"),max("count")).show()
```

```
+----------+------------------+----------+
|min(count)|        avg(count)|max(count)|
+----------+------------------+----------+
|         1|16.992025518341308|        75|
+----------+------------------+----------+
```

```
// 상품 유형당 최대, 최소, 평균 입찰 가격
auctionsDS.groupBy("itemtype", "aucid").agg(min("bid"), max("bid"), avg("bid")).show()
```

```
+--------+----------+--------+--------+------------------+
|itemtype|     aucid|min(bid)|max(bid)|          avg(bid)|
+--------+----------+--------+--------+------------------+
|    xbox|8215610555|    5.01|   35.09|22.582142857142856|
|    xbox|8212964295|   55.01|    86.0| 71.36454545454545|
| cartier|1638844729|   225.0|   320.0|284.54545454545456|
| cartier|1644594033|    20.0|   498.0| 156.5369565217391|
| cartier|1649858595|    75.0|   202.5|151.78571428571428|
|    palm|3018453060|   205.0|   266.0| 233.6342105263158|
|    palm|3016771147|     5.0|   255.0|        155.260625|
| cartier|1642514892|   500.0|  1025.0| 725.6111111111111|
|    palm|3015520551|     1.0|   232.5|103.13466666666666|
| cartier|1642875447|   675.0|   831.0|             754.5|
| cartier|1649757877|   102.5|   760.0| 348.8333333333333|
|    palm|3015694920|   230.0|   270.0|253.33333333333334|
|    palm|3022613118|   200.0|   232.5|218.61111111111111|
|    palm|3024889358|   240.0|   240.0|             240.0|
|    palm|3015328849|    12.0|   212.5| 94.59181818181817|
```

```
// 낙찰가가 200을 넘는 경매의 수
auctionsDS.filter(auctionsDS("price")>200).count()
```

```
res54: Long = 7685
```

온라인 경매 데이터세트 연산 (4)

8. xbox의 모든 경매에 대해 기본 통계 계산
   - describe() 함수 사용

```
// xbox의 모든 경매에 대해 기본 통계
val  xboxes = spark.sql("SELECT aucid,itemtype,bid,price,openbid FROM auctions WHERE itemtype='xbox'")

xboxes.describe("price").show()
xboxes.describe("bid").show()
xboxes.describe("price","bid").show()
```


온라인 경매 데이터세트 연산 – 실행 예 (3)

```
// xbox의 모든 경매에 대해 기본 통계
val  xboxes = spark.sql("SELECT aucid,itemtype,bid,price,openbid FROM auctions WHERE itemtype='xbox'")

xboxes.describe("price").show()
```

SPARK JOB FINISHED

```
xboxes: org.apache.spark.sql.DataFrame = [aucid: bigint, itemtype: string ... 3 more fields]
+-------+------------------+
|summary|             price|
+-------+------------------+
|  count|              2784|
|   mean|144.27594109195397|
| stddev| 72.94782944456601|
|    min|              31.0|
|    max|            501.77|
+-------+------------------+
```

Took 1 sec. Last updated by admin at August 01 2019, 3:58:03 PM.

```
xboxes.describe("bid").show()
xboxes.describe("price","bid").show()
```

SPARK JOBS FINISHED

```
+-------+-----------------+
|summary|              bid|
+-------+-----------------+
|  count|             2784|
|   mean|85.39793821839076|
| stddev|60.32284782511884|
|    min|             0.01|
|    max|           501.77|
+-------+-----------------+

+-------+------------------+-----------------+
|summary|             price|              bid|
+-------+------------------+-----------------+
|  count|              2784|             2784|
|   mean|144.27594109195397|85.39793821839076|
| stddev| 72.94782944456601|60.32284782511884|
|    min|              31.0|             0.01|
|    max|            501.77|           501.77|
+-------+------------------+-----------------+
```

과제

- 강의 시간의 실습 내용을 정리하여 제출
- 텀프로젝트 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