




# Spark 데이터분석 심화 [ 데이터타입 편]

김효관



흥미를 잃지 않는다면 못할것은 아무것도 없다.  
-데이비드 포스터 윌리스



# 1. 데이터프레임

## 주요 내용

1-1 행/열 조회

1-2 정제

1-3 정렬

1-4 데이터 조인

1-5 집계함수

1-6 고급함수

# 1. 데이터프레임

## 데이터프레임

RDD → 분산되어 있는 변경 불가능한 객체모음

RDD

week, product\_no, price, event\_no, age  
201701,PRODUCT1, 9000,ev0001,2  
201702,PRODUCT1,10000,ev0002,2

\* RDD: Resilient Distributed Dataset)

yearwe ek	product_no	volume	event_no	age
201701	PRODUCT1	9,000	EV0001	2
201702	PRODUCT1	10,000	EV0002	2
201703	PRODUCT1	15,000	EV0003	3
201704	PRODUCT1	18,000	-	3

Dataframe → 관계형 데이터베이스의 테이블과 같은개념

Dataframe

yearweek	product_n o	volume	event_no	age
201701	PRODUCT1	9,000	EV0001	2
201702	PRODUCT1	10,000	EV0002	2
201703	PRODUCT1	15,000	EV0003	3
201704	PRODUCT1	18,000	-	3

# 1. 데이터프레임

## 데이터 불러오기

KOPO\_PRODUCT\_VOLUME csv 파일을 불러와서  
selloutData 변수에 저장하기

# 1. 데이터프레임

## 1-1. 행/열 조회

### 유형

```
// 데이터 프레임 행 조회
var {데이터프레임 변수명} =
{데이터프레임 변수명}.
    filter($"컬럼명1" > "비교값" || $"컬럼명2" === "비교값")

// 데이터 프레임 열 조회
var {데이터프레임 변수명} =
{데이터프레임 변수명}.
    select($"컬럼명 #1", $"컬럼명 #2",...).
    select("컬럼명 #1", "컬럼명 #2",...)
```

### 실습예제

```
var filteredData = selloutData.select("PRODUCTGROUP","YEARWEEK","VOLUME").
    filter($"YEARWEEK" > 201650 && $"PRODUCTGROUP" === "ST0002")
```

# 1. 데이터프레임

## 1-2. 정제

### 유형

// 데이터 정제

```
var {데이터프레임 변수명} =  
{데이터프레임 변수명}.
```

```
    map("{컬럼명 #1}", "{컬럼명 #2}", ...)
```

String, Int, Double

```
var {데이터프레임 변수명} = selloutData.withColumn("VOLUME", $"VOLUME".cast("변경타입"))
```

### 실습예제

```
var refinedData = selloutData.map(x=>{  
    var maxValue = 150000  
    var volume = x.getString(3).toDouble  
    if( volume >maxValue) { volume = 150000}  
    (x.getString(0), x.getString(1), x.getString(2), volume))  
}
```

```
var selloutData = selloutData.withColumn("VOLUMNE", $"VOLUMNE".cast("Double"))
```

# 1. 데이터프레임

## 1-3. 정렬

### 유형

// 데이터 피벗

```
import org.apache.spark.sql.functions._  
{데이터프레임 변수명}.sort("컬럼명1","컬럼명2",...)
```

```
{데이터프레임 변수명}.orderBy($"컬럼명1".desc, $"컬럼명2".asc)
```

### 실습예제

```
var sortedDf = selloutData.sort("REGIONID","PRODUCTGROUP")  
var orderedDf = selloutData.orderBy($"PRODUCTGROUP".desc, $"YEARWEEK".asc)
```



# 1. 데이터프레임

## 1-4. 조인

### 유형

inner, left outer

// 데이터 피벗

```
var {데이터프레임명1} = {데이터프레임명1}.join( {데이터프레임명2}, Seq("조인키1","조인키2",...), "조인종류")
```

```
val joinTypes = Seq("inner", "full_outer", "left_outer", "right_outer", "full")
```

### 실습예제

```
var joinDf = movingAvgDf.join(movingStdDf, Seq("REGIONID","PRODUCTGROUP","YEARWEEK"),  
                              joinType = "inner")
```

# 1. 데이터프레임

## 1-5. 집계함수

### 유형

```
// 데이터 피벗  
var {데이터프레임 변수명} =  
{데이터프레임 변수명}.  
    groupBy($"그룹컬럼1", $"그룹컬럼2").  
    agg(mean($"값컬럼") as "컬럼 명")
```

### 실습예제

```
var groupDf = selloutData.  
    groupBy($"REGIONID", $"PRODUCTGROUP").  
    agg(mean($"VOLUME") as "MEAN_VOLUME")
```

# 1. 데이터프레임

## 1-6. 고급함수 구현 (이동 집계함수)

### 유형

// 데이터 피벗

```
import org.apache.spark.sql.expressions.Window
import org.apache.spark.sql.functions.{stddev_samp, stddev_pop}
var {데이터프레임 변수명} =
{데이터프레임 변수명}.
    withColumn("신규 컬럼명", 그룹함수(데이터프레임변수명("대상 컬럼명").
    over(Window.partitionBy("그룹컬럼1","그룹컬럼2").rowsBetween(-1,1))))
```

### 실습예제

```
var movingDf = selloutData.
    withColumn("MV_STD", stddev_pop(selloutData("VOLUME")).
    over( Window.partitionBy("REGIONID").rowsBetween(-3,3)))
```

# 1. 데이터프레임

## 1-6. 고급함수 구현 (피벗)

### 유형

// 데이터 피벗

```
var {데이터프레임 변수명} =  
{데이터프레임 변수명}.
```

```
    groupBy("{그룹키 컬럼명 #1}", "{그룹키 컬럼명 #2}", ...).
```

```
    pivot("행 → 열 전환 컬럼명", Seq("전환 컬럼명 #1", "전환 컬럼명 #2", ...)).
```

```
    sum("피벗 계산 값 컬럼명")
```

### 실습예제

```
var pivotDf = selloutData.groupBy("REGIONID", "PRODUVTGROUP").  
    pivot("YEARWEEK", Seq("201501", "201502")).sum("VOLUME")
```

감사합니다

