

# PySpark 미니 프로젝트

김지현

# 1. 데이터 소개

→ Spark 세션 초기화 및 데이터(2022년 서울시 음주운전 교통사고 데이터) 불러오기

- header = True 옵션으로 가장 첫 행을 컬럼으로 불러오도록 지정

→ 데이터 건수(레코드(DF.count())), 컬럼 수(len(DF)) 확인

```
import findspark
findspark.find()

from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession

spark = SparkSession.builder.master('local').appName('Spark') \
    .config('spark.sql.repl.eagerEval.enabled', True).getOrCreate()
```

```
# 데이터 불러오기 - 음주운전 교통사고 데이터
raw_df = spark.read.csv('음주운전사고현황_2022서울.csv', header = True, encoding = 'utf-8')
raw_df.show()
```

자치구별 (1)	자치구별 (2)	20222	20223	20224
자치구별 (1)	자치구별 (2)	발생건수 (건)	사망자수 (명)	부상자수 (명)
서울시	소계	2348	23	3904
서울시	종로구	56	-	87
서울시	중구	45	-	73
서울시	용산구	94	3	148
서울시	성동구	71	1	105
서울시	광진구	66	-	128
서울시	동대문구	58	-	77
서울시	중랑구	60	1	109
서울시	성북구	59	-	90
서울시	강북구	42	-	68
서울시	도봉구	40	-	69
서울시	노원구	93	2	149
서울시	은평구	69	1	118
서울시	서대문구	56	-	86
서울시	마포구	119	-	208
서울시	양천구	68	-	123
서울시	강서구	137	3	227
서울시	구로구	83	-	148
서울시	금천구	72	-	131

only showing top 20 rows

```
print(f'Row count : {raw_df.count()}')
print(f'Column count : {len(raw_df.columns)}')
```

Row count : 27  
Column count : 5

## 2. 데이터 전처리

- 전처리한 데이터는 'finalDF'로 저장
- 서울시 데이터이므로 모든 값이 '서울시'인 '자치구별(1)' 컬럼 삭제
- 발생건수, 사망자 수, 부상자 수를 의미하는 컬럼명(영문)으로 변경

```
finalDF = raw_df.drop('자치구별(1)')  
finalDF.show()
```

자치구별(2)	20222	20223	20224
자치구별(2)	발생건수 (건)	사망자수 (명)	부상자수 (명)
소계	2348	23	3904
종로구	56	-	87
중구	45	-	73
용산구	94	3	148
성동구	71	1	105
광진구	66	-	128
동대문구	58	-	77
중랑구	60	1	109
성북구	59	-	90
강북구	42	-	68
도봉구	40	-	69
노원구	93	2	149
은평구	69	1	118
서대문구	56	-	86
마포구	119	-	208
양천구	68	-	123
강서구	137	3	227
구로구	83	-	148
금천구	72	-	131

only showing top 20 rows

```
newColumns = ['region','cnt','death','injury']  
finalDF = finalDF.toDF(*newColumns)  
finalDF.show()
```

region	cnt	death	injury
자치구별(2)	발생건수 (건)	사망자수 (명)	부상자수 (명)
소계	2348	23	3904
종로구	56	-	87
중구	45	-	73
용산구	94	3	148
성동구	71	1	105
광진구	66	-	128
동대문구	58	-	77
중랑구	60	1	109
성북구	59	-	90
강북구	42	-	68
도봉구	40	-	69
노원구	93	2	149
은평구	69	1	118
서대문구	56	-	86
마포구	119	-	208
양천구	68	-	123
강서구	137	3	227
구로구	83	-	148
금천구	72	-	131

only showing top 20 rows

## 2. 데이터 전처리

→ 첫 번째 Row값을 컬럼명으로 변경하였으니 삭제

→ 자치구 별로 건수 비교를 용이하게 하기 위해 소계(행) 삭제

```
# 1행 첫 행 삭제
finalDF = finalDF.filter(~(finalDF.region == '자치구별(2)'))
finalDF.show(truncate = False)
```

region	cnt	death	injury
소계	2348	23	3904
종로구	56	-	87
중구	45	-	73
용산구	94	3	148
성동구	71	1	105
광진구	66	-	128
동대문구	58	-	77
중랑구	60	1	109
성북구	59	-	90
강북구	42	-	68
도봉구	40	-	69
노원구	93	2	149
은평구	69	1	118
서대문구	56	-	86
마포구	119	-	208
양천구	68	-	123
강서구	137	3	227
구로구	83	-	148
금천구	72	-	131
영등포구	146	6	226

only showing top 20 rows

```
# 2행 첫 행 삭제 (2)
finalDF = finalDF.filter(~(finalDF.region == '소계'))
finalDF.show(truncate = False)
```

region	cnt	death	injury
종로구	56	-	87
중구	45	-	73
용산구	94	3	148
성동구	71	1	105
광진구	66	-	128
동대문구	58	-	77
중랑구	60	1	109
성북구	59	-	90
강북구	42	-	68
도봉구	40	-	69
노원구	93	2	149
은평구	69	1	118
서대문구	56	-	86
마포구	119	-	208
양천구	68	-	123
강서구	137	3	227
구로구	83	-	148
금천구	72	-	131
영등포구	146	6	226
동작구	72	-	113

only showing top 20 rows

## 2. 데이터 전처리

- death 컬럼의 '-' 값을 '0'으로 변경
- 스키마 확인 및 cnt, death, injury 컬럼 타입(string → integer) 변경
- 전처리된 데이터 건수 재확인

```
finalDF = finalDF.na.replace("-", "0")  
finalDF.show()
```

```
+-----+-----+-----+  
| region|cnt|death|injury|  
+-----+-----+-----+  
| 종로구| 56|    0|   87|  
| 중구| 45|    0|   73|  
| 용산구| 94|    3|  148|  
| 성동구| 71|    1|  105|  
| 광진구| 66|    0|  128|  
| 동대문구| 58|    0|   77|  
| 중랑구| 60|    1|  109|  
| 성북구| 59|    0|   90|  
| 강북구| 42|    0|   68|  
| 도봉구| 40|    0|   69|  
| 노원구| 93|    2|  149|  
| 은평구| 69|    1|  118|  
| 서대문구| 56|    0|   86|  
| 마포구|119|    0|  208|  
| 양천구| 68|    0|  123|  
| 강서구|137|    3|  227|  
| 구로구| 83|    0|  148|  
| 금천구| 72|    0|  131|  
| 영등포구|146|    6|  226|  
| 동작구| 72|    0|  113|  
+-----+-----+-----+  
only showing top 20 rows
```

```
# 스키마 확인  
finalDF.printSchema()
```

```
root  
|-- region: string (nullable = true)  
|-- cnt: string (nullable = true)  
|-- death: string (nullable = true)  
|-- injury: string (nullable = true)
```

```
# 타입 변경
```

```
finalDF = finalDF.select(finalDF.region, finalDF.cnt.cast('int'), finalDF.death.cast('int'), finalDF.injury.cast('int'))  
finalDF.printSchema()
```

```
root  
|-- region: string (nullable = true)  
|-- cnt: integer (nullable = true)  
|-- death: integer (nullable = true)  
|-- injury: integer (nullable = true)
```

```
print(f'Row count : {finalDF.count()}')  
print(f'Column count : {len(finalDF.columns)}')
```

```
Row count : 25  
Column count : 4
```

### 3. 데이터 확인

→ finalDF의 요약 정보 확인

- 각 필드(사고 건수, 사망자 수, 부상자 수)별로 평균, 표준편차, 최대값, 최소값 출력

```
finalDF.describe().show(truncate = False)
```

	summary	region	cnt	death	injury
count	25	25	25	25	25
mean	null	93.92	0.92	156.16	
stddev	null	67.06148919710427	1.4118545723031581	111.3211121036796	
min	강남구	40	0	68	
max	중랑구	368	6	601	

### 3. 데이터 확인

→ max와 min 패키지 이용하여 cnt(사고 건수) 최대값/최소값 alias를 통해 별칭 지정하여 출력

→ 해당 값들이 어떤 지역(행)인지 filter함수로 확인

```
from pyspark.sql.functions import max
finalDF.select(max(finalDF.cnt).alias('음주운전 사고 최대 건수')).show()
```

```
+-----+
|음주운전 사고 최대 건수|
+-----+
|                368|
+-----+
```

```
finalDF.filter(finalDF.cnt == 368).show()
```

```
+---+---+---+---+
|region|cnt|death|injury|
+---+---+---+---+
|강남구|368|    1|    601|
+---+---+---+---+
```

```
from pyspark.sql.functions import min
finalDF.select(min(finalDF.cnt).alias('음주운전 사고 최소 건수')).show()
```

```
+-----+
|음주운전 사고 최소 건수|
+-----+
|                40|
+-----+
```

```
finalDF.filter(finalDF.cnt == 40).show()
```

```
+---+---+---+---+
|region|cnt|death|injury|
+---+---+---+---+
|도봉구| 40|    0|    69|
+---+---+---+---+
```

### 3. 데이터 확인

→ filter함수 사용하여 사고 건수가 평균(93.92) 이상인 레코드만 출력

→ filter함수 사용하여 사고 건수 평균(93.92) 이상 & 부상자 수(156.16)이상인 조건 모두 만족하는 레코드 출력

```
finalDF.filter(finalDF.cnt > 93.92).show()
```

region	cnt	death	injury
용산구	94	3	148
마포구	119	0	208
강서구	137	3	227
영등포구	146	6	226
서초구	165	1	299
강남구	368	1	601
송파구	155	1	262

```
finalDF.filter((finalDF.cnt > 93.92) & (finalDF.injury > 156.16)).show()
```

region	cnt	death	injury
마포구	119	0	208
강서구	137	3	227
영등포구	146	6	226
서초구	165	1	299
강남구	368	1	601
송파구	155	1	262



### 3. 데이터 확인

→ select 함수 이용하여 필드 일부 추출 및 alias 사용하여 지정한 명칭으로 출력

→ when 함수 이용하여 조건(사고 건수)에 따라 high, normal, low로 결과 필드를 새로 만들어 출력

```
from pyspark.sql.functions import when

finalDF.select(finalDF.region.alias('자치구명'), finalDF.cnt.alias('사고건수'),
               when(finalDF.cnt >= 150, 'high').
               when((finalDF.cnt < 150) & (finalDF.cnt >= 55), 'normal').
               otherwise('low')).alias('결과')).show(30)
```

자치구명	사고건수	결과
종로구	56	normal
중구	45	low
용산구	94	normal
성동구	71	normal
광진구	66	normal
동대문구	58	normal
중랑구	60	normal
성북구	59	normal
강북구	42	low
도봉구	40	low
노원구	93	normal
은평구	69	normal
서대문구	56	normal
마포구	119	normal
양천구	68	normal
강서구	137	normal
구로구	83	normal
금천구	72	normal
영등포구	146	normal
동작구	72	normal
관악구	72	normal
서초구	165	high
강남구	368	high
송파구	155	high
강동구	82	normal