# 2.9 데이터 소스 (246~, 29)

기본 제공 : CSV, JSON, Parquet, ORC, JDBC/ODBC, Text, etc

커뮤니티 제공 : 카산드라, HBase, 몽고디비, AWS Redshift, XML, etc

9.1.1 읽기 API 구조

- format : 선택적으로 사용. 기본값은 parquet
- option : 데이터를 읽는 방법. k-v방식으로 설정
- schema : 선택적으로 사용. 데이터 소스에 직접 스키마 정의

```scala
DataFrameReader.format(...) .option("k", "v") .schema(...) .load()
```

9.1.2 데이터 읽기의 기초

- DataFrameReader는 SparkSession의 read속성으로 접근

```scala
spark.read
```

- 읽기 모드 : 스파크가 형식에 맞지 않는 데이터를 만났을 때의 동작 방식을 지정하는 옵션

  - permissive (default) : 오류 레코드의 모든 필드 null로 설정.
    모든 오류 레코드를 *corrupt*record라는 문자열 컬럼에 기록

- dropMalformed : 오류 로우 제거
- failFast : 오류시 즉시 종료

9.1.3 쓰기 API 구조

- format : 선택적으로 사용. 기본값은 parquet
- option : 데이터 쓰기 방법. k-v방식으로 설정
- partitionBy,bucketBy,sortBy : 파일 기반의 데이터소스에 적용. 최종 파일 배치 형태 제어

```scala
DataFrameReader.format(...) .option("k", "v") .partitionBy(...)
.bucketBy(...) .sortBy(...) .save()
```

Scala

9.1.4 데이터 쓰기의 기초

- DataFrameWriter는 DataFrame의 write속성으로 접근

```scala
dataFrame.write
```

Scala

- 저장 모드 : 스파크가 지정된 위치에 동일한 파일 발견시 동작 방식을 지정하는 옵션
  - errorIfExists (default) : 해당 경로에 파일 존재 시, 오류 발생
  - apend : 목록에 파일을 추가
  - overwrite :  모든 데이터를 완전히 덮어씀
  - ignore : 아무처리도 하지 않음

9.2 CSV 파일

- 각 줄이 단일 레코드가 되며 레코드의 각 필드를 콤마로 구분하는 텍스트 파일 포맷
- 배열 사용 불가

9.2.2 CSV 파일 읽기

```
import org.apache.spark.sql.types.{StructField, StructType, StringType,
LongType} val myManualSchema = new StructType(Array( new
StructField("DEST_COUNTRY_NAME", StringType, true), new
StructField("ORIGIN_COUNTRY_NAME", StringType, true), new
StructField("count", LongType, false) )) val csvFile =
spark.read.format("csv") .option("header", "true") .option("mode",
"FAILFAST") .schema(myManualSchema) .load("2010-summary.csv")
csvFile.show(5) +-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count| +-----------------+------
-------------+-----+ | United States| Romania| 1| | United States|
Ireland| 264| | United States| India| 69| | Egypt| United States| 24|
|Equatorial Guinea| United States| 1| +-----------------+-------------
-----+-----+
```

Scala

9.2.3 CSV 파일 쓰기 (comma-separated values)

- save의 파일명은 디렉토리이름
- 데이터를 쓰는 시점의 DataFrame 파티션 수만큼 생김

ex1) csv파일을 tsv 파일로 쓰기 예시

```
csvFile.write.format("csv") .mode("overwrite") .option("sep", "\t")
.save("/tmp/my-tsv-file.tsv")
```

Scala

ex2) 입력 및 결과 파일 살펴보기. 디렉토리에 헤더가 없는 파일 생김

```
$ ls /tmp/my-tsv-file.tsv total 24 drwxr-xr-x 2 zepl zepl 4096 Feb 19
17:10 . drwxrwxrwt 8 root root 4096 Feb 19 17:10 .. -rw-r--r-- 1 zepl
zepl 8 Feb 19 17:10 ._SUCCESS.crc -rw-r--r-- 1 zepl zepl 64 Feb 19
17:10 .part-00000-e537c680-a197-4f90-a8b4-ffb55ef93831-c000.csv.crc -
rw-r--r-- 1 zepl zepl 0 Feb 19 17:10 _SUCCESS -rw-r--r-- 1 zepl zepl
7073 Feb 19 17:10 part-00000-e537c680-a197-4f90-a8b4-ffb55ef93831-
c000.csv $ wc -l /tmp/my-tsv-file.tsv/part-00000-e537c680-a197-4f90-
a8b4-ffb55ef93831-c000.csv 255 /tmp/my-tsv-file.tsv/part-00000-
```

```
e537c680-a197-4f90-a8b4-ffb55ef93831-c000.csv $ head -3 /tmp/my-tsv-
file.tsv/part-00000-e537c680-a197-4f90-a8b4-ffb55ef93831-c000.csv
United States Romania 1 United States Ireland 264 United States India
69 $ wc -l 2010-summary.csv 256 2010-summary.csv $ head -3 2010-
summary.csv DEST_COUNTRY_NAME,ORIGIN_COUNTRY_NAME,count United
States,Romania,1 United States,Ireland,264
```

Bash

9.3 JSON 파일 (JavaScript Object Notation)

- 줄로 구분된 JSON : Spark에서 기본으로 사용하는 형태. 새로운 레코드를 추가할 수 있다.

```
{"string":"string1","int":1,"array":[1,2,3],"dict": {"key": "value1"}}
{"string":"string2","int":2,"array":[2,4,6],"dict": {"key": "value2"}}
{"string":"string3","int":3,"array":[3,6,9],"dict": {"key": "value3",
"extra_key": "extra_value3"}}
```

JSON

- 여러 줄로 구성된 JSON

```
[ {"string":"string1","int":1,"array":[1,2,3],"dict": {"key":
"value1"}}, {"string":"string2","int":2,"array":[2,4,6],"dict": {"key":
"value2"}}, { "string": "string3", "int": 3, "array": [ 3, 6, 9 ],
"dict": { "key": "value3", "extra_key": "extra_value3" } } ]
```

JSON

9.3.2 JSON 파일 읽기

```
spark.read.format("json") .option("mode", "FAILFAST")
.schema(myManualSchema) .load("2010-summary.json") .show(5) +----------
-------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count| +-----------------+------
-------------+-----+ | United States| Romania| 1| | United States|
Ireland| 264| | United States| India| 69| | Egypt| United States| 24|
|Equatorial Guinea| United States| 1| +-----------------+--------------
-----+-----+
```

Scala

9.3.3 JSON 파일 쓰기

- 파티션당 하나의 파일을 만들며 전체 DataFrame을 단일 폴더에 저장 (csv와 동일)

ex1) csv파일을 json파일로 쓰기

```
csvFile.write.format("json") .mode("overwrite") .save("/tmp/my-json-file.json")
```

Scala

ex2) 결과 파일 확인

```
$ ls -la /tmp/my-json-file.json total 40 drwxr-xr-x 2 zepl zepl 4096 Feb 19 17:35 . drwxrwxrwt 9 root root 4096 Feb 19 17:35 .. -rw-r--r-- 1 zepl zepl 8 Feb 19 17:35 ._SUCCESS.crc -rw-r--r-- 1 zepl zepl 176 Feb 19 17:35 .part-00000-e51cdf14-37b9-4da3-b5d9-34e35b15a836-c000.json.crc -rw-r--r-- 1 zepl zepl 0 Feb 19 17:35 _SUCCESS -rw-r--r-- 1 zepl zepl 21353 Feb 19 17:35 part-00000-e51cdf14-37b9-4da3-b5d9-34e35b15a836-c000.json $ wc -l /tmp/my-json-file.json/part-00000-e51cdf14-37b9-4da3-b5d9-34e35b15a836-c000.json 255 /tmp/my-json-file.json/part-00000-e51cdf14-37b9-4da3-b5d9-34e35b15a836-c000.json $ head -3 /tmp/my-json-file.json/part-00000-e51cdf14-37b9-4da3-b5d9-34e35b15a836-c000.json
{"DEST_COUNTRY_NAME":"United States","ORIGIN_COUNTRY_NAME":"Romania","count":1}
{"DEST_COUNTRY_NAME":"United States","ORIGIN_COUNTRY_NAME":"Ireland","count":264}
{"DEST_COUNTRY_NAME":"United States","ORIGIN_COUNTRY_NAME":"India","count":69}
```

Bash

9.4 파케이 파일 (parquet)

- 컬럼 기반의 데이터 저장 방식. 컬럼 별로 데이터 조회 가능
- 저장소 공간 절약
- 컬럼 기반의 압축 기능 제공
- 복합 데이터 타입 지원 ( 배열, 맵, 구조체 )

9.4.1 파케이 파일 읽기

- 스키마가 파일 자체에 내장되어 있으므로 스키마 추정은 불필요
- 스파크와 잘 호환되기 때문에 스파크의 기본 파일 포맷

ex1) 디렉토리를 지정해서 데이터 조회

```
spark.read.format("parquet") .load("2010-summary.parquet") .show(5) +--
---------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count| +-----------------+------
-------------+-----+ | United States| Romania| 1| | United States|
Ireland| 264| | United States| India| 69| | Egypt| United States| 24|
|Equatorial Guinea| United States| 1| +-----------------+-------------
-----+-----+
```

Scala

ex2) 입력 파일 확인. parquet는 바이너리 형태

```
$ ls -l 2010-summary.parquet/ total 4 -rw-r--r-- 1 zepl zepl 0 Feb 19
17:48 _SUCCESS -rw-r--r-- 1 zepl zepl 3921 Feb 19 17:48 part-r-00000-
1a9822ba-b8fb-4d8e-844a-ea30d0801b9e.gz.parquet $ head -3 2010-
summary.parquet/part-r-00000-1a9822ba-b8fb-4d8e-844a-
ea30d0801b9e.gz.parquet PAR1��L��UUMs7�!V�|
□□��:�=BVdY��Q,□ؤ��."."��VV���Ɋ@%A���zwvv��(be@"��|0��.@.ə@k��k>��>���~@
-:����H@ِ@W@@3@@W'@@b@>@]o@Uf T@@,z@_@@@.@Gr%@@
@y@CP@@v@@@@q@kQ@@@@@o@@c\@}@~b[@jyx@]@@o
ø@@@9@f@*@O@@%`@H6@@@)0H@@@)@W!@@@@Y@@ @@ *@@@P{@@@/n@@?s%@N@ -@@
```

Bash

9.4.2 파케이 파일 쓰기

```
csvFile.write.format("parquet") .mode("overwrite") .save("/tmp/my-
parquet-file.parquet")
```

Scala

```
$ ls -la /tmp/my-parquet-file.parquet total 24 drwxr-xr-x 2 zepl zepl
4096 Feb 19 17:52 . drwxrwxrwt 10 root root 4096 Feb 19 17:52 .. -rw-r-
-r-- 1 zepl zepl 8 Feb 19 17:52 ._SUCCESS.crc -rw-r--r-- 1 zepl zepl 52
Feb 19 17:52 .part-00000-a9867659-c994-4934-88a7-70fd3a7c45ac-
c000.snappy.parquet.crc -rw-r--r-- 1 zepl zepl 0 Feb 19 17:52 _SUCCESS
-rw-r--r-- 1 zepl zepl 5139 Feb 19 17:52 part-00000-a9867659-c994-4934-
88a7-70fd3a7c45ac-c000.snappy.parquet
```

Bash

9.5 ORC 파일

- 하둡 워크로드를 위해 설계된 자기 기술적이며 데이터 타입을 인식할 수 있는 컬럼 기반의 파일 포맷
- 대규모 스트리밍 읽기에 최적화
- Parquet와 유사. Parquet는 스파크에 최적화. ORC는 하이브에 최적화

9.5.1 ORC 파일 읽기

ex1) orc 파일이 들어있는 디렉토리를 지정하여 데이터 읽기 가능

```
spark.read.format("orc") .load("2010-summary.orc") .show(5)
```

Scala

ex2) enableHiveSupport 처리해야 사용 가능

```
scala> spark.read.format("orc") .load("2010-summary.orc") .show(5)
org.apache.spark.sql.AnalysisException: The ORC data source must be
used with Hive support enabled; at
org.apache.spark.sql.execution.datasources.DataSource$.lookupDataSource
(DataSource.scala:537) scala> val spark =
SparkSession.builder().enableHiveSupport().getOrCreate()
```

Bash

ex3) 입력 파일 확인

```
$ ls -la 2010-summary.orc/ total 12 drwxr-xr-x 2 zepl zepl 4096 Feb 19
17:56 . drwxr-xr-x 9 zepl zepl 4096 Feb 19 17:56 .. -rw-r--r-- 1 zepl
zepl 0 Feb 19 17:56 _SUCCESS -rw-r--r-- 1 zepl zepl 3852 Feb 19 17:56
part-r-00000-2c4f7d96-e703-4de3-af1b-1441d172c80f.snappy.orc $ head -5
2010-summary.orc/part-r-00000-2c4f7d96-e703-4de3-af1b-
1441d172c80f.snappy.orc ORC ▯PS ' ▯" AfghanistanVietnam▯,PS
```

Bash

### 9.5.2 ORC 파일 쓰기

```
csvFile.write.format("orc") .mode("overwrite") .save("/tmp/my-json-
file.orc")
```

Scala

## 9.6 SQL 데이터베이스

- 데이터베이스의 데이터를 읽고 쓰기 위해서는
  1) 스파크 클래스패스에 데이터베이스의 JDBC드라이버를 추가하고,
  2) 적절한 JDBC 드라이버 jar 파일을 제공해야 한다.

```
./bin/spark-shell \ --driver-class-path postgresql-9.4.1207.jar \ --
jars postgresql-9.4.1207.jar
```

Bash

### 9.6.1 SQL 데이터베이스 읽기

- dbtable에는 서브쿼리도 허용

```
spark.read.format("jdbc") .option("driver", "org.postgresql.Driver")
.option("url", "jdbc:postgresql://database_server") .option("dbtable",
"schema.tablename") .option("user", "username") .option("password","my-
secret-password") .load()
```

Scala

### 9.6.2 쿼리 푸시다운

- 스파크는 DataFrame을 만들기 전에 데이터베이스 자체에서 데이터를 필터링하도록 만들 수 있다.
- 특정 컬럼만 사용하는 경우, 특정 컬럼만 조회하는 쿼리가 DB에서 실행된다.
- 필터를 명시하면, 해당 필터 처리를 데이터베이스로 위임(push down)한다.
- 모든 데이터베이스에 맞게 변환되지 않기 때문에 dbtable에서 서브쿼리 형태로 최적화할 수 있다.

```scala
val pushdownQuery = """ ( SELECT DISTINCT(DEST_COUNTRY_NAME) FROM flight_info ) AS flight_info """ spark.read.format("jdbc") .option("url", url) .option("dbtable", pushdownQuery) .option("driver", driver) .load()
```

Scala

**.데이터베이스 병렬로 읽기**

- 최대 파티션 수를 설정 가능

```bash
spark.read.format("jdbc") .option("url", url) .option("dbtable", tablename) .option("driver", driver) .option("numPartitions", 10) # 1개로 될 수도 있음 .load()
```

Bash

- 특정 파티션에 특정 데이터의 물리적 위치를 제어할 수 있음

```scala
val props = new java.util.Properties props.setProperty("driver", "org.sqlite.JDBC") val predicates = Array( "DEST_COUNTRY_NAME = 'Sweden' OR ORIGIN_COUNTRY_NAME = 'Sweden'", "DEST_COUNTRY_NAME = 'Anguilla' OR ORIGIN_COUNTRY_NAME = 'Anguilla'") spark.read.jdbc(url, tablename, predicates, props).show() spark.read.jdbc(url, tablename, predicates, props).rdd.getNumPartitions // 2
```

Scala

**.슬라이딩 윈도우 기반의 파티셔닝**

- 최소값, 최대값을 기준으로 n개의 파티션을 생성
- 최소, 최대 밖의 모든 값은 첫 번째 또는 마지막 파티션에 속함

```
val colName = "count" val lowerBound = 0L val upperBound = 348113L val numPartitions = 10 val df = spark.read.jdbc( url, tablename, colName, lowerBound, upperBound, numPartitions, props)
```

Scala

### 9.6.3 SQL 데이터베이스 쓰기

```
val newPath = "jdbc:sqlite://tmp/my-sqlite.db" # 전체 테이블을 덮어쓰기 csvFile.write.mode("overwrite").jdbc(newPath, tablename, props) # 테이블에 레코드를 추가하기 csvFile.write.mode("append").jdbc(newPath, tablename, props)
```

Scala

## 9.7 텍스트 파일 (plain-text file)

- 파일의 각 줄은 DataFrame의 레코드가 됨

### 9.7.1 텍스트 파일 읽기

```
spark.read.textFile("2010-summary.csv") .selectExpr("split(value, ',') as rows") .show() +--------------------+ | rows| +--------------------+ |[DEST_COUNTRY_NAM...| |[United States, R...| |[United States, I...| |[United States, I...| |[Egypt, United St...| |[Equatorial Guine...| |[United States, S...| |[United States, G...| |[Costa Rica, Unit...| |[Senegal, United ...| |[United States, M...| |[Guyana, United S...| |[United States, S...| |[Malta, United St...| |[Bolivia, United ...| |[Anguilla, United...| |[Turks and Caicos...| |[United States, A...| |[Saint Vincent an...| |[Italy, United St...| +--------------------+
```

Scala

### 9.7.2 텍스트 파일 쓰기

- 텍스트 파일을 쓸 때는 문자열 컬럼이 하나만 존재해야 한다.

- 파티셔닝 작업을 수행하면 더 많은 컬럼을 저장할 수도 있다.
- 단, 파일 하나에 여러 컬럼이 들어가는 게 아니라, 텍스트 파일이 저장되는 디렉토리에 파일별로 컬럼 저장

ex1) 문자열 컬럼

```scala
csvFile.select("DEST_COUNTRY_NAME").write.text("/tmp/simple-text-file.txt")
```

Scala

```bash
$ ls -la /tmp/simple-text-file.txt total 20 drwxr-xr-x 2 zepl zepl 4096 Feb 19 18:25 . drwxrwxrwt 11 root root 4096 Feb 19 18:25 .. -rw-r--r-- 1 zepl zepl 8 Feb 19 18:25 ._SUCCESS.crc -rw-r--r-- 1 zepl zepl 36 Feb 19 18:25 .part-00000-daadb30a-add2-4bb2-95ad-f1386b5ce35d-c000.txt.crc -rw-r--r-- 1 zepl zepl 0 Feb 19 18:25 _SUCCESS -rw-r--r-- 1 zepl zepl 3124 Feb 19 18:25 part-00000-daadb30a-add2-4bb2-95ad-f1386b5ce35d-c000.txt
```

Bash

ex2) 숫자형 컬럼

```scala
csvFile.select("count").write.text("/tmp/simple-text-file.txt")
org.apache.spark.sql.AnalysisException: Text data source supports only a string column, but you have bigint.; at org.apache.spark.sql.execution.datasources.text.TextFileFormat.verifySchema(TextFileFormat.scala:51) at org.apache.spark.sql.execution.datasources.text.TextFileFormat.prepareWrite(TextFileFormat.scala:66) at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:135) ...
```

Scala

ex3) 컬럼 두개

```scala
csvFile.select("DEST_COUNTRY_NAME", "ORIGIN_COUNTRY_NAME").write.text("/tmp/simple-text-file.txt")
```

```
org.apache.spark.sql.AnalysisException: Text data source supports only
a single column, and you have 2 columns.; at
org.apache.spark.sql.execution.datasources.text.TextFileFormat.verifySc
hema(TextFileFormat.scala:46) at
org.apache.spark.sql.execution.datasources.text.TextFileFormat.prepareW
rite(TextFileFormat.scala:66) ...
```

Scala

9.8 고급 I/O 개념

- 쓰기 작업 전에 파티션 수를 조절함으로써 병렬로 처리할 파일 수 제어 가능
- 버켓팅과 파티셔닝을 조절함으로써 데이터의 저장 구조 제어

9.8.1 분할 가능한 파일 타입과 압축 방식

- 파일을 분할하면 필요한 부분만 읽기 때문에 성능 향상
- HDFS는 분할된 파일을 여러 블록으로 나누어 분산 저장
- 모든 압축 방식이 분할 압축을 지원하지 않음
- Spark에서는 Parquet 파일 포맷에 GZIP 압축 방식을 추천

9.8.2 병렬로 데이터 읽기

- 익스큐터들은 "같은" 파일을 동시에 읽을 수는 없지만, "여러" 파일을 동시에 읽을 수 있음
- 폴더에 파일들이 있을 때, 각 파일은 DataFrame의 파티션이 됨

9.8.3 병렬로 데이터 쓰기

- 기본적으로 데이터 파티션당 하나의 파일이 작성

ex1) 여러 파티션으로 만들어서 저장

```
csvFile.repartition(5) .write.format("csv") .save("/tmp/multiple.csv")
```

Scala

ex2) 결과 확인

```
$ ls -la /tmp/multiple.csv total 52 drwxr-xr-x 2 zepl zepl 4096 Feb 19 18:35 . drwxrwxrwt 12 root root 4096 Feb 19 18:35 .. -rw-r--r-- 1 zepl zepl 8 Feb 19 18:35 ._SUCCESS.crc -rw-r--r-- 1 zepl zepl 20 Feb 19 18:35 .part-00000-6cec6a0f-41e5-4fac-816b-1b33737eae11-c000.csv.crc -rw-r--r-- 1 zepl zepl 20 Feb 19 18:35 .part-00001-6cec6a0f-41e5-4fac-816b-1b33737eae11-c000.csv.crc -rw-r--r-- 1 zepl zepl 20 Feb 19 18:35 .part-00002-6cec6a0f-41e5-4fac-816b-1b33737eae11-c000.csv.crc -rw-r--r-- 1 zepl zepl 20 Feb 19 18:35 .part-00003-6cec6a0f-41e5-4fac-816b-1b33737eae11-c000.csv.crc -rw-r--r-- 1 zepl zepl 20 Feb 19 18:35 .part-00004-6cec6a0f-41e5-4fac-816b-1b33737eae11-c000.csv.crc -rw-r--r-- 1 zepl zepl 0 Feb 19 18:35 _SUCCESS -rw-r--r-- 1 zepl zepl 1373 Feb 19 18:35 part-00000-6cec6a0f-41e5-4fac-816b-1b33737eae11-c000.csv -rw-r--r-- 1 zepl zepl 1439 Feb 19 18:35 part-00001-6cec6a0f-41e5-4fac-816b-1b33737eae11-c000.csv -rw-r--r-- 1 zepl zepl 1396 Feb 19 18:35 part-00002-6cec6a0f-41e5-4fac-816b-1b33737eae11-c000.csv -rw-r--r-- 1 zepl zepl 1460 Feb 19 18:35 part-00003-6cec6a0f-41e5-4fac-816b-1b33737eae11-c000.csv -rw-r--r-- 1 zepl zepl 1409 Feb 19 18:35 part-00004-6cec6a0f-41e5-4fac-816b-1b33737eae11-c000.csv $ head -5 /tmp/multiple.csv/part-00000-6cec6a0f-41e5-4fac-816b-1b33737eae11-c000.csv Equatorial Guinea,United States,1 United States,Marshall Islands,44 Anguilla,United States,21 United States,Russia,156 Marshall Islands,United States,77
```

Bash

### .파티셔닝 (partitioning)

- 디렉토리별로 컬럼 데이터를 인코딩해서 저장
- 각 폴더는 조건절을 폴더명으로 사용
- 데이터를 읽을 때 전체 데이터셋을 스캔하지 않고 필요한 컬럼의 데이터만 읽을 수 있다.
- 필터를 자주하는 조건으로 파티셔닝하면 성능 향상에 도움

ex1) 파티셔닝하여 저장

```
csvFile.limit(10).write.mode("overwrite")
.partitionBy("DEST_COUNTRY_NAME") .save("/tmp/partitioned-files.parquet")
```

Scala

ex2) 결과 확인

```
$ ls -laR /tmp/partitioned-files.parquet /tmp/partitioned-
files.parquet: total 32 drwxr-xr-x 7 zepl zepl 4096 Feb 19 18:40 .
drwxrwxrwt 13 root root 4096 Feb 19 18:40 .. -rw-r--r-- 1 zepl zepl 8
Feb 19 18:40 ._SUCCESS.crc drwxr-xr-x 2 zepl zepl 4096 Feb 19 18:40
DEST_COUNTRY_NAME=Costa Rica drwxr-xr-x 2 zepl zepl 4096 Feb 19 18:40
DEST_COUNTRY_NAME=Egypt drwxr-xr-x 2 zepl zepl 4096 Feb 19 18:40
DEST_COUNTRY_NAME=Equatorial Guinea drwxr-xr-x 2 zepl zepl 4096 Feb 19
18:40 DEST_COUNTRY_NAME=Senegal drwxr-xr-x 2 zepl zepl 4096 Feb 19
18:40 DEST_COUNTRY_NAME=United States -rw-r--r-- 1 zepl zepl 0 Feb 19
18:40 _SUCCESS /tmp/partitioned-files.parquet/DEST_COUNTRY_NAME=Costa
Rica: total 16 drwxr-xr-x 2 zepl zepl 4096 Feb 19 18:40 . drwxr-xr-x 7
zepl zepl 4096 Feb 19 18:40 .. -rw-r--r-- 1 zepl zepl 16 Feb 19 18:40
.part-00000-f726489f-4626-4731-ab1c-
6a62bc9ae020.c000.snappy.parquet.crc -rw-r--r-- 1 zepl zepl 663 Feb 19
18:40 part-00000-f726489f-4626-4731-ab1c-
6a62bc9ae020.c000.snappy.parquet /tmp/partitioned-
files.parquet/DEST_COUNTRY_NAME=Egypt: total 16 drwxr-xr-x 2 zepl zepl
4096 Feb 19 18:40 . drwxr-xr-x 7 zepl zepl 4096 Feb 19 18:40 .. -rw-r--
r-- 1 zepl zepl 16 Feb 19 18:40 .part-00000-f726489f-4626-4731-ab1c-
6a62bc9ae020.c000.snappy.parquet.crc -rw-r--r-- 1 zepl zepl 663 Feb 19
18:40 part-00000-f726489f-4626-4731-ab1c-
6a62bc9ae020.c000.snappy.parquet ... ...
```

Bash

**.버켓팅 (bucketing)**

- 동일한 버킷ID를 가진 데이터가 하나의 물리적 파티션에 모두 모이게 함
- 데이터를 읽을 때 고비용의 셔플을 피할수 있음
- /user/hive/warehouse 디렉토리 하위에 버켓팅 파일을 기록
- 버켓팅은 스파크 관리 테이블에서만 사용 가능

```
val numberBuckets = 10 val columnToBucketBy = "count"
csvFile.write.format("parquet") .mode("overwrite")
.bucketBy(numberBuckets, columnToBucketBy)
.saveAsTable("bucketedFiles")
```

Scala

### 9.8.4 복합 데이터 유형 쓰기

- CSV는 복합 데이터 타입을 지원하지 않지만
- Parquet, ORC는 복합 데이터 타입을 지원

### 9.8.5 파일 크기 관리

- 작은 파일을 많이 생성하면 메타데이터에 엄청난 관리 부하 발생
- HDFS 같은 많은 파일 시스템은 작은 크기의 파일을 잘 다루지 못함. 스파크는 더 함
- 파일이 너무 크면, 적은 수의 데이터를 조회할 때도 큰 파일을 로딩해야 함
- maxRecordsPerFile 옵션을 통해, 각 파일별 기록될 레코드 수를 조정 가능

```
csvFile.repartition(5) .write.format("csv").option("maxRecordsPerFile",
10) .save("/tmp/multiple.csv2")
```

Scala

```
$ ls -l /tmp/multiple.csv2 total 120 -rw-r--r-- 1 zepl zepl 0 Feb 19
18:52 _SUCCESS -rw-r--r-- 1 zepl zepl 282 Feb 19 18:52 part-00000-
6b9ca683-7ac0-4cec-82c7-84051cf085a6-c000.csv -rw-r--r-- 1 zepl zepl
254 Feb 19 18:52 part-00000-6b9ca683-7ac0-4cec-82c7-84051cf085a6-
c001.csv -rw-r--r-- 1 zepl zepl 281 Feb 19 18:52 part-00000-6b9ca683-
7ac0-4cec-82c7-84051cf085a6-c002.csv -rw-r--r-- 1 zepl zepl 268 Feb 19
18:52 part-00000-6b9ca683-7ac0-4cec-82c7-84051cf085a6-c003.csv -rw-r--
r-- 1 zepl zepl 264 Feb 19 18:52 part-00000-6b9ca683-7ac0-4cec-82c7-
84051cf085a6-c004.csv -rw-r--r-- 1 zepl zepl 24 Feb 19 18:52 part-
00000-6b9ca683-7ac0-4cec-82c7-84051cf085a6-c005.csv -rw-r--r-- 1 zepl
zepl 311 Feb 19 18:52 part-00001-6b9ca683-7ac0-4cec-82c7-84051cf085a6-
c000.csv -rw-r--r-- 1 zepl zepl 261 Feb 19 18:52 part-00001-6b9ca683-
7ac0-4cec-82c7-84051cf085a6-c001.csv -rw-r--r-- 1 zepl zepl 289 Feb 19
18:52 part-00001-6b9ca683-7ac0-4cec-82c7-84051cf085a6-c002.csv -rw-r--
r-- 1 zepl zepl 287 Feb 19 18:52 part-00001-6b9ca683-7ac0-4cec-82c7-
84051cf085a6-c003.csv -rw-r--r-- 1 zepl zepl 261 Feb 19 18:52 part-
00001-6b9ca683-7ac0-4cec-82c7-84051cf085a6-c004.csv -rw-r--r-- 1 zepl
zepl 30 Feb 19 18:52 part-00001-6b9ca683-7ac0-4cec-82c7-84051cf085a6-
c005.csv -rw-r--r-- 1 zepl zepl 266 Feb 19 18:52 part-00002-6b9ca683-
```

```
7ac0-4cec-82c7-84051cf085a6-c000.csv -rw-r--r-- 1 zepl zepl 276 Feb 19
18:52 part-00002-6b9ca683-7ac0-4cec-82c7-84051cf085a6-c001.csv -rw-r--
r-- 1 zepl zepl 292 Feb 19 18:52 part-00002-6b9ca683-7ac0-4cec-82c7-
84051cf085a6-c002.csv -rw-r--r-- 1 zepl zepl 279 Feb 19 18:52 part-
00002-6b9ca683-7ac0-4cec-82c7-84051cf085a6-c003.csv -rw-r--r-- 1 zepl
zepl 259 Feb 19 18:52 part-00002-6b9ca683-7ac0-4cec-82c7-84051cf085a6-
c004.csv -rw-r--r-- 1 zepl zepl 24 Feb 19 18:52 part-00002-6b9ca683-
7ac0-4cec-82c7-84051cf085a6-c005.csv -rw-r--r-- 1 zepl zepl 303 Feb 19
18:52 part-00003-6b9ca683-7ac0-4cec-82c7-84051cf085a6-c000.csv -rw-r--
r-- 1 zepl zepl 283 Feb 19 18:52 part-00003-6b9ca683-7ac0-4cec-82c7-
84051cf085a6-c001.csv -rw-r--r-- 1 zepl zepl 299 Feb 19 18:52 part-
00003-6b9ca683-7ac0-4cec-82c7-84051cf085a6-c002.csv -rw-r--r-- 1 zepl
zepl 279 Feb 19 18:52 part-00003-6b9ca683-7ac0-4cec-82c7-84051cf085a6-
c003.csv -rw-r--r-- 1 zepl zepl 273 Feb 19 18:52 part-00003-6b9ca683-
7ac0-4cec-82c7-84051cf085a6-c004.csv -rw-r--r-- 1 zepl zepl 23 Feb 19
18:52 part-00003-6b9ca683-7ac0-4cec-82c7-84051cf085a6-c005.csv -rw-r--
r-- 1 zepl zepl 258 Feb 19 18:52 part-00004-6b9ca683-7ac0-4cec-82c7-
84051cf085a6-c000.csv -rw-r--r-- 1 zepl zepl 263 Feb 19 18:52 part-
00004-6b9ca683-7ac0-4cec-82c7-84051cf085a6-c001.csv -rw-r--r-- 1 zepl
zepl 267 Feb 19 18:52 part-00004-6b9ca683-7ac0-4cec-82c7-84051cf085a6-
c002.csv -rw-r--r-- 1 zepl zepl 282 Feb 19 18:52 part-00004-6b9ca683-
7ac0-4cec-82c7-84051cf085a6-c003.csv -rw-r--r-- 1 zepl zepl 286 Feb 19
18:52 part-00004-6b9ca683-7ac0-4cec-82c7-84051cf085a6-c004.csv -rw-r--
r-- 1 zepl zepl 53 Feb 19 18:52 part-00004-6b9ca683-7ac0-4cec-82c7-
84051cf085a6-c005.csv
```

Bash