Cours de « Apache Spark »

Spark SQL Exemples with JSON Input

```
scala> import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.SparkSession

scala> val session =
SparkSession.builder().appName("app_name").master("local").getOrCreate()
22/10/12 22:58:30 WARN SparkSession: Using an existing Spark session; only
runtime SQL configurations will take effect.
session: org.apache.spark.sql.SparkSession =
org.apache.spark.sql.SparkSession@5ce49280

scala> val training =
session.read.json("examples/src/main/resources/people.json")
training: org.apache.spark.sql.DataFrame = [age: bigint, name: string]


scala> training.show()
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+


scala> import session.implicits._
import session.implicits._

scala> training.printSchema()
root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)
```

```
scala> training.select("name").show()
+-------+
|   name|
+-------+
|Michael|
|   Andy|
| Justin|
+-------+
scala> training.select($"name", $"age" + 1).show()
+-------+---------+
|   name|(age + 1)|
+-------+---------+
|Michael|     null|
|   Andy|       31|
| Justin|       20|
+-------+---------+
scala> training.filter($"age" > 21).show()
+---+----+
|age|name|
+---+----+
| 30|Andy|
+---+----+
scala> training.groupBy("age").count().show()
+----+-----+
| age|count|
+----+-----+
|  19|    1|
|null|    1|
|  30|    1|
+----+-----+
scala> // Register the DataFrame as a SQL temporary view
scala> training.createOrReplaceTempView("people")
scala> val sqlDF = session.sql("SELECT * FROM people")
sqlDF: org.apache.spark.sql.DataFrame = [age: bigint, name: string]
scala> sqlDF.show()
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+

scala> // Register the DataFrame as a global temporary view
```

```
scala> training.createGlobalTempView("people")
scala>session.sql("SELECT * FROM global_temp.people").show()
scala>spark.newSession().sql("SELECT * FROM global_temp.people").show()
```

Creating Dataset

```
case class Person(name: String, age: Long)

// Encoders are created for case classes
val caseClassDS = Seq(Person("Andy", 32)).toDS()
caseClassDS.show()
// +----+---+
// |name|age|
// +----+---+
// |Andy| 32|
// +----+---+

// Encoders for most common types are automatically provided by importing spark.implicits._
val primitiveDS = Seq(1, 2, 3).toDS()
primitiveDS.map(_ + 1).collect() // Returns: Array(2, 3, 4)

// DataFrames can be converted to a Dataset by providing a class. Mapping will be done by name
val path = "examples/src/main/resources/people.json"
val peopleDS = spark.read.json(path).as[Person]
peopleDS.show()
// +----+-------+
// | age|   name|
// +----+-------+
// |null|Michael|
// |  30|   Andy|
// |  19| Justin|
// +----+-------+
```

Mini projects :

Inferring the Schema Using Reflection

Programmatically Specifying the Schema

La source : https://spark.apache.org/docs/latest/sql-getting-started.html