# Apache Spark SQL Example

## Simple Spark SQL Example

Pom.xml

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.spark.tutorial</groupId>  <artifactId>sample-spark-project</artifactId>  <version>0.0.1-SNAPSHOT</version>  <dependencies>  <dependency>  <!-- Spark dependency -->  <groupId>org.apache.spark</groupId>  <artifactId>spark-core\_2.11</artifactId>  <version>1.6.0</version>  </dependency>  <dependency>  <groupId>org.apache.spark</groupId>  <artifactId>spark-sql\_2.11</artifactId>  <version>1.6.0</version>  </dependency>  </dependencies>  </project> |

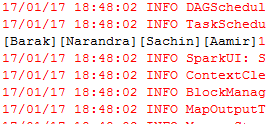
Code

|  |
| --- |
| **package** org.apache.spark  **import** org.apache.spark.SparkContext  **import** org.apache.spark.SparkContext.\_  **import** org.apache.spark.SparkConf  **import** org.apache.spark.sql.SQLContext  **import** org.apache.spark.sql.\_  **import** org.apache.spark.sql.SQLContext.\_  // if we know the schema in advance we can use case class and create data frame  // later we can convert dataframe to temporary table  // if we dont know schema in advance we can use json or csv file to load the data  **object** SimpleSparkSqlExample {    **def** main(args : Array[*String*]){    **val** conf = **new** SparkConf().setAppName("SimpleSparkSQL Application").setMaster("local[2]").set("spark.executor.memory", "1g")  **val** sc = **new** SparkContext(conf)    **val** sqlc = **new** org.apache.spark.sql.SQLContext(sc)//sqlsc    **val** p=sc.textFile("src/test/resources/Person.txt")    **val** pmap = p.map(p=> p.split(","))  **val** PersonRdd=pmap.map(p=>**Person**(p(0),p(1),p(2).toInt))  **import** sqlc.implicits.\_  **val** PersonDF = PersonRdd.toDF //RDD+Schema  PersonDF.registerTempTable("Person")    print(sqlc.sql("select count(\*) from Person").collect().toList)  sqlc.sql("select first\_name from Person where age >40").collect().foreach (print)    }  }  **case** **class** **Person**(first\_name:*String*,last\_name:*String*,age:Int) |

Person.txt

|  |
| --- |
| Barak,Obama,53  Narandra,Modi,63  Virat,Kohli,28  Sachin,Tendulkar,43  Aamir,Khan,53 |

Output



## Spark SQL CSV Example

If we don’t know the schema in advance we can use csv with header file to do the sql processing

Pom.xml

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.spark.tutorial</groupId>  <artifactId>sample-spark-project</artifactId>  <version>0.0.1-SNAPSHOT</version>  <dependencies>  <dependency>  <!-- Spark dependency -->  <groupId>org.apache.spark</groupId>  <artifactId>spark-core\_2.11</artifactId>  <version>1.6.0</version>  </dependency>  <dependency>  <groupId>org.apache.spark</groupId>  <artifactId>spark-sql\_2.11</artifactId>  <version>1.6.0</version>  </dependency>  <dependency>  <groupId>com.databricks</groupId>  <artifactId>spark-csv\_2.11</artifactId>  <version>1.2.0</version>  </dependency>  </dependencies>  </project> |

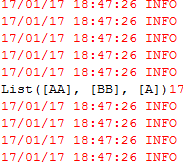
Code

|  |
| --- |
| **package** org.apache.spark  **import** org.apache.spark.SparkContext.\_  **import** org.apache.spark.SparkConf.\_  **import** com.databricks.spark.csv  **object** SparkSqlCsvJExample {  **def** main(args: Array[*String*]) {  **val** conf = **new** SparkConf().setAppName("Spark Sql CSV App").setMaster("local[2]").set("spark.executor.memory", "1g")  **val** sc = **new** org.apache.spark.SparkContext(conf)  **val** sqlc = **new** org.apache.spark.sql.SQLContext(sc)    **val** NyseDF = sqlc.~~load~~("com.databricks.spark.csv", Map("path"->"src/test/resources/Nyse.csv","header"->"true"))  NyseDF.registerTempTable("NYSE")    // use data frame directly to get the results  **val** countNyse = NyseDF.agg(("\*", "count"))  countNyse.collect().foreach (println)    // to check the schema  //NyseDF.printSchema()    print(sqlc.sql(" select distinct(symbol) from NYSE").collect().toList)      }  } |

Nyse.csv

|  |
| --- |
| symbol,date,open,close,high,low  A,01-Jan-2009,15.63,15.63,15.63,15.63,1  AA,01-Jan-2009,11.26,11.26,11.26,11.26,2  A,01-Jan-2009,15.63,15.63,15.63,15.63,3  AA,01-Jan-2010,11.26,11.26,11.26,11.26,4  A,01-Jan-2011,15.63,15.63,15.63,15.63,5  AA,01-Jan-2012,11.26,11.26,11.26,11.26,6  A,01-Jan-2013,15.63,15.63,15.63,15.63,7  BB,01-Jan-2014,11.26,11.26,11.26,11.26,10 |

Output



## Spark SQL JSON Example

Pom.xml

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.spark.tutorial</groupId>  <artifactId>sample-spark-project</artifactId>  <version>0.0.1-SNAPSHOT</version>  <dependencies>  <dependency>  <!-- Spark dependency -->  <groupId>org.apache.spark</groupId>  <artifactId>spark-core\_2.11</artifactId>  <version>1.6.0</version>  </dependency>  <dependency>  <groupId>org.apache.spark</groupId>  <artifactId>spark-sql\_2.11</artifactId>  <version>1.6.0</version>  </dependency>  <dependency>  <groupId>com.databricks</groupId>  <artifactId>spark-csv\_2.11</artifactId>  <version>1.2.0</version>  </dependency>  </dependencies>  </project> |

Code

|  |
| --- |
| **package** org.apache.spark  **import** org.apache.spark.SparkContext.\_  **import** org.apache.spark.SparkConf.\_  **import** com.databricks.spark.csv  **object** SparkSqlJsonExample {  **def** main(args: Array[*String*]) {  **val** conf = **new** SparkConf().setAppName("Spark Sql CSV App").setMaster("local[2]").set("spark.executor.memory", "1g")  **val** sc = **new** org.apache.spark.SparkContext(conf)  **val** sqlc = **new** org.apache.spark.sql.SQLContext(sc)    **val** personDF = sqlc.~~jsonFile~~("src/test/resources/Person.json")    // it convert the datatype as per the data value  personDF.printSchema()    personDF.registerTempTable("person")    sqlc.sql("select \* from person where age < 60").collect().foreach (print)  }  } |

Person.json

|  |
| --- |
| {"first\_name" : "Barack", "last\_name" : "Obama", "age" : 53}  {"first\_name" : "George", "last\_name" : "Bush", "age" : 68 }  {"first\_name" : "Bill", "last\_name" : "Clinton", "age" : 68 } |

Output

