1. Start SPARK

[cloudera@quickstart ~]$ spark-shell

2. To make dataframe (Dataframe from RDD)

scala> val a = sc.parallelize(1 to 10)

a: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:27

scala> a.collect

res1: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

scala> val b = a.map( x => (x,x+1))

b: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[1] at map at <console>:29

scala> b.collect

res2: Array[(Int, Int)] = Array((1,2), (2,3), (3,4), (4,5), (5,6), (6,7), (7,8), (8,9), (9,10), (10,11))

scala> val df = b.toDF("first","second")

df: org.apache.spark.sql.DataFrame = [first: int, second: int]

scala> df.show

+-----+------+

|first|second|

+-----+------+

| 1| 2|

| 2| 3|

| 3| 4|

| 4| 5|

| 5| 6|

| 6| 7|

| 7| 8|

| 8| 9|

| 9| 10|

| 10| 11|

+-----+------+

3. To make a Dataframe (From List)

scala> val a = List(("Tom",5),("Jerry",2),("Donald",7))

a: List[(String, Int)] = List((Tom,5), (Jerry,2), (Donald,7))

scala> val df = a.toDF("Name","Age")

df: org.apache.spark.sql.DataFrame = [Name: string, Age: int]

scala> df.show

+------+---+

| Name|Age|

+------+---+

| Tom| 5|

| Jerry| 2|

|Donald| 7|

+------+---+

4. To make a Dataframe (From Seq)

scala> val a = Seq(("Tom",5),("Jerry",2),("Donald",7))

a: Seq[(String, Int)] = List((Tom,5), (Jerry,2), (Donald,7))

scala> val df = a.toDF("Name","Age")

df: org.apache.spark.sql.DataFrame = [Name: string, Age: int]

scala> df.show

+------+---+

| Name|Age|

+------+---+

| Tom| 5|

| Jerry| 2|

|Donald| 7|

+------+---+

5. To run queries

scala> df.registerTempTable("cartoon")

scala> sqlContext.sql("select \* from cartoon where Name = 'Tom'").show

+----+---+

|Name|Age|

+----+---+

| Tom| 5|

+----+---+

scala> sqlContext.sql("select count(\*) from cartoon").show

+---+

|\_c0|

+---+

| 3|

+---+

After Break

5. To use own schema

a. Inferring schema from reflection ( define inside class)

b. Inferring schema iusing struct type

a. Inferring schema from reflection ( define inside class)

1. scala> val yahoo\_stocks = sc.textFile("/user/cloudera/sparksqlprac/yahoo\_stocks.csv")

yahoo\_stocks: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[1] at textFile at <console>:21

2. scala> yahoo\_stocks.collect

res1: Array[String] = Array(Date,Open,High,Low,Close,Volume,Adj Close, 2001-01-02,30.3125,30.375,27.50,28.1875,21939200,14.09375, 2000-12-29,30.3125,31.1875,29.5625,30.0625,20893400,15.03125, 2000-12-28,29.4375,31.75,29.125,31.00,24374600,15.50, 2000-12-27,31.00,31.50,29.125,29.75,22045400,14.875, 2000-12-26,32.00,34.00,30.125,31.1875,37536200,15.59375, 2000-12-22,26.4375,29.875,26.0625,29.5625,28347400,14.78125, 2000-12-21,26.75,28.25,25.0625,25.625,27794400,12.8125, 2000-12-20,25.8125,28.375,25.50,27.9375,44862800,13.96875, 2000-12-19,30.5625,31.9687,28.00,28.00,36131600,14.00, 2000-12-18,33.875,34.00,30.25,32.00,31697600,16.00, 2000-12-15,32.00,34.00,31.0625,33.00,40448000,16.50, 2000-12-14,35.3125,35.9062,31.9375,32.00,20899800,16.00, 2000-12-13,38.3125,38.625,34.25,34.875,33640400,...

3. scala> val header = yahoo\_stocks.first

header: String = Date,Open,High,Low,Close,Volume,Adj Close

4. scala> val data = yahoo\_stocks.filter(\_ != header)

data: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at filter at <console>:25

5. scala> data.collect

res2: Array[String] = Array(2001-01-02,30.3125,30.375,27.50,28.1875,21939200,14.09375, 2000-12-29,30.3125,31.1875,29.5625,30.0625,20893400,15.03125, 2000-12-28,29.4375,31.75,29.125,31.00,24374600,15.50, 2000-12-27,31.00,31.50,29.125,29.75,22045400,14.875, 2000-12-26,32.00,34.00,30.125,31.1875,37536200,15.59375, 2000-12-22,26.4375,29.875,26.0625,29.5625,28347400,14.78125, 2000-12-21,26.75,28.25,25.0625,25.625,27794400,12.8125, 2000-12-20,25.8125,28.375,25.50,27.9375,44862800,13.96875, 2000-12-19,30.5625,31.9687,28.00,28.00,36131600,14.00, 2000-12-18,33.875,34.00,30.25,32.00,31697600,16.00, 2000-12-15,32.00,34.00,31.0625,33.00,40448000,16.50, 2000-12-14,35.3125,35.9062,31.9375,32.00,20899800,16.00, 2000-12-13,38.3125,38.625,34.25,34.875,33640400,17.4375, 2000-12-12,33.25,39.50,32.9375,35....

6. Define a schema in class

scala> case class YahooStockPrice(date:String, open:Float, high: Float, low: Float, close:Float, volume:Integer,adjClose:Float)

defined class YahooStockPrice

7. Use above schema and make df

scala> val stockpriceDF = data.map(\_.split(",")).map(row => YahooStockPrice(row(0) , row(1).trim.toFloat, row(2).trim.toFloat, row(3).trim.toFloat, row(4).trim.toFloat, row(5).trim.toInt, row(6).trim.toFloat)).toDF()

stockpriceDF: org.apache.spark.sql.DataFrame = [date: string, open: float, high: float, low: float, close: float, volume: int, adjClose: float]

8. to see data from Dataframe

scala> stockpriceDF.show

+----------+-------+-------+-------+-------+--------+--------+

| date| open| high| low| close| volume|adjClose|

+----------+-------+-------+-------+-------+--------+--------+

|2001-01-02|30.3125| 30.375| 27.5|28.1875|21939200|14.09375|

|2000-12-29|30.3125|31.1875|29.5625|30.0625|20893400|15.03125|

|2000-12-28|29.4375| 31.75| 29.125| 31.0|24374600| 15.5|

|2000-12-27| 31.0| 31.5| 29.125| 29.75|22045400| 14.875|

|2000-12-26| 32.0| 34.0| 30.125|31.1875|37536200|15.59375|

|2000-12-22|26.4375| 29.875|26.0625|29.5625|28347400|14.78125|

|2000-12-21| 26.75| 28.25|25.0625| 25.625|27794400| 12.8125|

|2000-12-20|25.8125| 28.375| 25.5|27.9375|44862800|13.96875|

|2000-12-19|30.5625|31.9687| 28.0| 28.0|36131600| 14.0|

|2000-12-18| 33.875| 34.0| 30.25| 32.0|31697600| 16.0|

|2000-12-15| 32.0| 34.0|31.0625| 33.0|40448000| 16.5|

|2000-12-14|35.3125|35.9062|31.9375| 32.0|20899800| 16.0|

|2000-12-13|38.3125| 38.625| 34.25| 34.875|33640400| 17.4375|

|2000-12-12| 33.25| 39.5|32.9375|35.8125|79275800|17.90625|

|2000-12-11| 33.625|37.0625| 30.625| 33.875|71038800| 16.9375|

|2000-12-08| 37.125| 37.125| 32.125|34.9375|49184000|17.46875|

|2000-12-07|36.0625|36.2187| 31.5|34.9375|55136200|17.46875|

|2000-12-06| 41.625|42.9375| 37.125| 37.5|32559800| 18.75|

|2000-12-05|39.6875| 44.0|39.3125| 43.875|30714800| 21.9375|

|2000-12-04|38.4375|39.3125|36.1875|37.9375|29997600|18.96875|

+----------+-------+-------+-------+-------+--------+--------+

b. Inferring schema iusing struct type

1. Import

scala> import org.apache.spark.sql.{SQLContext,Row}

scala> import org.apache.spark.sql.types.{IntegerType, StringType, StructField, StructType}

import org.apache.spark.sql.types.{IntegerType, StringType, StructField, StructType}

2. Define schema in Struct

scala> val schema = StructType(Array(StructField("name", StringType,true),StructField("age",IntegerType,true)))

schema: org.apache.spark.sql.types.StructType = StructType(StructField(name,StringType,true), StructField(age,IntegerType,true))

3.

scala> val data = sc.parallelize(Seq("Tom","Jerry","Donald")).map(x => (x,2+x.length))

data: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[12] at map at <console>:22

scala> data.collect

res5: Array[(String, Int)] = Array((Tom,5), (Jerry,7), (Donald,8)

4.

scala> val rowRDD = data.map(x => Row (x.\_1,x.\_2))

rowRDD: org.apache.spark.rdd.RDD[org.apache.spark.sql.Row] = MapPartitionsRDD[13] at map at <console>:25

scala> rowRDD.collect

res7: Array[org.apache.spark.sql.Row] = Array([Tom,5], [Jerry,7], [Donald,8])

5. Make dataframe

scala> val df = sqlContext.createDataFrame( rowRDD, schema)

df: org.apache.spark.sql.DataFrame = [name: string, age: int]

scala> df.show

+------+---+

| name|age|

+------+---+

| Tom| 5|

| Jerry| 7|

|Donald| 8|

+------+---+

6.

scala> df.registerTempTable("cartoon")

scala> sqlContext.sql("Select \* from cartoon").show

+------+---+

| name|age|

+------+---+

| Tom| 5|

| Jerry| 7|

|Donald| 8|

+------+---+

=============================================

Loading data from different formats

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[cloudera@quickstart ~]$ echo 'export JAVA\_TOOL\_OPTIONS="-Dhttps.protocols=TLSv1.2"' >> ~/.bashrc source ~/.bashrc

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

===============================================

A. CSV

[cloudera@quickstart ~]$ spark-shell --packages com.databricks:spark-csv\_2.10:1.5.0

Picked up JAVA\_TOOL\_OPTIONS: -Dhttps.protocols=TLSv1.2

Picked up JAVA\_TOOL\_OPTIONS: -Dhttps.protocols=TLSv1.2

scala> val characters\_df = sqlContext.read.format("com.databricks.spark.csv").option("header","true").option("inferSchema","true").option("delimiter",",").load("/user/cloudera/sparksqlprac/StarWars.csv")

characters\_df: org.apache.spark.sql.DataFrame = [name: string, height: int, weight: int, eyecolor: string, haircolor: string, jedi: string, species: string]

scala> characters\_df.show

+----------------+------+------+--------+---------+-------+-----------+

| name|height|weight|eyecolor|haircolor| jedi| species|

+----------------+------+------+--------+---------+-------+-----------+

| nakin Skywalker| 188| 84| blue| blond| jedi| human|

| Padme Amidala| 165| 45| brown| brown|no\_jedi| human|

| Luke Skywalker| 172| 77| blue| blond| jedi| human|

| Leia Skywalker| 150| 49| brown| brown|no\_jedi| human|

| Qui-Gon Jinn| 193| 89| blue| brown| jedi| human|

| Obi-Wan Kenobi| 182| 77|bluegray| auburn| jedi| human|

| Han Solo| 180| 80| brown| brown|no\_jedi| human|

| Sheev Palpatine| 173| 75| blue| red|no\_jedi| human|

| R2-D2| 96| 32| | |no\_jedi| droid|

| C-3PO| 167| 75| | |no\_jedi| droid|

| Yoda| 66| 17| brown| brown| jedi| yoda|

| Darth Maul| 175| 80| yellow| none|no\_jedi|dathomirian|

| Dooku| 193| 86| brown| brown| jedi| human|

| Chewbacca| 228| 112| blue| brown|no\_jedi| wookiee|

| Jabba| 390| null| yellow| none|no\_jedi| hutt|

|Lando Calrissian| 178| 79| brown| blank|no\_jedi| human|

| Boba Fett| 183| 78| brown| black|no\_jedi| human|

| Jango Fett| 183| 79| brown| black|no\_jedi| human|

+----------------+------+------+--------+---------+-------+-----------+

scala> characters\_df.registerTempTable("starwars")

scala> sqlContext.sql("select \* from starwars where species = 'droid'").show

+-----+------+------+--------+---------+-------+-------+

| name|height|weight|eyecolor|haircolor| jedi|species|

+-----+------+------+--------+---------+-------+-------+

|R2-D2| 96| 32| | |no\_jedi| droid|

|C-3PO| 167| 75| | |no\_jedi| droid|

+-----+------+------+--------+---------+-------+-------+

B. XML

[cloudera@quickstart ~]$ spark-shell --packages com.databricks:spark-xml\_2.10:0.4.1

scala> val employees\_df = sqlContext.read.format("com.databricks.spark.xml").option("inferSchema","true").option("rootTag","employees").option("rowTag","employee").load("/user/cloudera/sparksqlprac/employees.xml")

employees\_df: org.apache.spark.sql.DataFrame = [address: struct<city:string,country:string,pincode:bigint>, dept\_no: bigint, emp\_name: string, emp\_no: bigint, salary: bigint]

scala> employees\_df.show

+--------------------+-------+--------+------+------+

| address|dept\_no|emp\_name|emp\_no|salary|

+--------------------+-------+--------+------+------+

|[Paris,London,200...| 2| jon| 10| 15000|

|[Texas,America,20...| 5| Adom| 11| 25000|

+--------------------+-------+--------+------+------+

C. JSON

scala> val usstates = sqlContext.read.json("sparksqlprac/us\_states.json")

usstates: org.apache.spark.sql.DataFrame = [census\_division: string, census\_region: string, name: string, state: string]

scala> usstates.collect

res5: Array[org.apache.spark.sql.Row] = Array([East South Central,South,Alabama,AL], [Pacific,West,Alaska,AK], [Mountain,West,Arizona,AZ], [West South Central,South,Arkansas,AR], [Pacific,West,California,CA], [Mountain,West,Colorado,CO], [New England,Northeast,Connecticut,CT], [South Atlantic,South,Delaware,DE], [South Atlantic,South,District Of Columbia,DC], [South Atlantic,South,Florida,FL], [South Atlantic,South,Georgia,GA], [Pacific,West,Hawaii,HI], [Mountain,West,Idaho,ID], [East North Central,Midwest,Illinois,IL], [East North Central,Midwest,Indiana,IN], [West North Central,Midwest,Iowa,IA], [West North Central,Midwest,Kansas,KS], [East South Central,South,Kentucky,KY], [West South Central,South,Louisiana,LA], [New England,Northeast,Maine,ME], [South Atlantic,South,Maryland,MD], [...

scala> usstates.printSchema

root

|-- census\_division: string (nullable = true)

|-- census\_region: string (nullable = true)

|-- name: string (nullable = true)

|-- state: string (nullable = true)

^

scala> usstates.registerTempTable("usstates")

^

scala> sqlContext.sql("select \* from usstates").show(2)

+------------------+-------------+-------+-----+

| census\_division|census\_region| name|state|

+------------------+-------------+-------+-----+

|East South Central| South|Alabama| AL|

| Pacific| West| Alaska| AK|

+------------------+-------------+-------+-----+

only showing top 2 rows

scala> sqlContext.sql("select \* from usstates limit 2").show()

+------------------+-------------+-------+-----+

| census\_division|census\_region| name|state|

+------------------+-------------+-------+-----+

|East South Central| South|Alabama| AL|

| Pacific| West| Alaska| AK|

+------------------+-------------+-------+-----+