**Section 2 - First Steps with Spark and Visualization**

**V 2.1 - Manipulate Data with the core RDD API**

**---------------------------------------------**

sc

sc.version

---

val rdd = sc.parallelize(List(1, 2, 3, 4, 5, 6), 2)

rdd.first

rdd.take(1)

rdd.collect()

rdd.collect().foreach(i => println("-> " + i))

---

rdd.mapPartitionsWithIndex((index: Int, iter: Iterator[Int]) => iter.toList.map("[partId:" + index + ", val: " + \_ + "]").iterator).collect

---

%sh head /dataset/bank/bank.csv

---

val bankText = sc.textFile("file:///dataset/bank/bank.csv")

bankText.cache

bankText.count

val bankHead = bankText.take(10)

bankHead.size

---

case class Bank(age: Integer, job: String, marital: String, education: String, balance: Integer)

val bankRdd = bankText.map(\_.split(";")).filter(\_.head != "\"age\"").map(

s => Bank(s(0).toInt,

s(1).replaceAll("\"", ""),

s(2).replaceAll("\"", ""),

s(3).replaceAll("\"", ""),

s(5).replaceAll("\"", "").toInt

)

)

---

bankRdd.saveAsTextFile("/tmp/bankrdd.txt")

---

%sh ls /tmp/bankrdd.txt

---

bankRdd.repartition(3).saveAsTextFile("/tmp/bank-repartition.txt")

---

%sh ls /tmp/bank-repartition.txt

**V 2.2 - Use Dataframe, Dataset and SQL - It is Natural and Easy!**

**----------------------------------------------------------------**

spark

---

val bankRDD = spark.read

.format("org.apache.spark.sql.execution.datasources.csv.CSVFileFormat")

.option("header", "true")

.option("inferSchema", "true")

.option("sep", ";")

.load("/dataset/bank/bank.csv")

---

val bankDF = bankRDD.toDF

bankDF.printSchema

bankRDD.toDF("age", "job", "marital", "education", "default", "balance", "housing", "loan", "contact", "day", "month", "duration", "campaign", "pdays", "previous", "poutcome", "y")

---

val ds = Seq(1, 2, 3).toDS()

ds.map(\_ \* 2).show

case class Person(name: String, zip: Long)

val df = spark.read.json(sc.parallelize("""{"zip": 94709, "name": "Michael"}""" :: Nil))

val ds2 = df.as[Person]

ds2.collect

ds2.show

case class Bank(age:Integer, job:String, marital : String, education : String, balance : Integer)

val bankDS = bankDF.as[Bank]

bankDS.take(2)

---

bankDF.createOrReplaceTempView("bank")

val banks = spark.sql("select age, balance from bank limit 10")

banks.map(t => "Age: " + t(0)).collect().foreach(println)

---

%sql select age, balance from bank

---

%sql select age, sum(balance) from bank group by age

---

val df = sc.makeRDD(1 to 500).map(i => (i, i \* 2)).toDF("single", "double")

println(df)

df.createOrReplaceTempView("u")

df.printSchema

spark.sql("select count(\*) from u").first.get(0)

---

%sql select count(\*) from u

**V-2.3 - Manipulate Rows and Columns**

**-----------------------------------**

case class KeyValue(key: Int, value: String)

val df1 = sc.parallelize((1 to 3).map(i => KeyValue(i, s"val\_$i"))).toDF()

val df2 = sc.parallelize((3 to 7).map(i => KeyValue(i, s"val\_$i"))).toDF()

val df3 = sc.parallelize((3 to 7).map(i => KeyValue(i, s"val\_${i\*2}"))).toDF()

---

val df4 = df1.union(df2).union(df3)

---

df4.show()

---

df4.filter("key > 6").show

---

df4.filter("key = 3 or value = 'val\_12'").show

---

df4.sort(df4("key").desc).show

---

import org.apache.spark.sql.functions.\_

df4.groupBy("key").agg(count("value")).show

---

df4.groupBy("key").agg(max("value")).show

---

df4.dropDuplicates(Seq("key")).show

---

import java.sql.Timestamp

import java.util.Calendar

import org.apache.spark.sql.types.TimestampType

case class Record(key: Int, value: String)

val df = sqlc.createDataFrame(Record(1, "foo") :: Record(2, "bar") :: Nil)

---

df.withColumn("foobar", lit("a-string": String)).printSchema

---

df.select(df("key"), df("value").as("value2")).printSchema

---

val now = new Timestamp(Calendar.getInstance().getTimeInMillis);

df.withColumn("foobar", lit(now: Timestamp).cast(TimestampType)).collect.map(println)

---

case class Record(key: Int, value: String)

val ddf = sc.parallelize((1 to 3).map(i => Record(i, s"val\_$i"))).toDF()

ddf.withColumn("key\_plus\_one", ddf("key") + 1).collect.map(println)

---

val ds = Seq(1, 2, 3).toDS()

ds.map(\_ \* 2).show

**V-2.4 - Deal with File Format**

**-----------------------------------**

val p = spark.read.json("file:///dataset/json/people.json")

p.show()

p.printSchema()

p.select("name").show()

p.select(p("name"), p("age") + 1).show()

p.filter(p("age") > 21).show()

p.groupBy("age").count().show()

---

val banks = spark.read

.format("com.databricks.spark.csv")

.option("header", "true")

.option("inferSchema", "true")

.option("delimiter", ";")

.load("file:///dataset/bank/bank2.csv")

banks.printSchema

banks.count

---

banks.createOrReplaceTempView("banks")

spark.sql("select \* from banks limit 10").write.json("file:///tmp/banks-json")

---

%sh cat /tmp/banks-json/\*

**V-2.5 - 02.5. Visualize More - Ggplot2, Matplotlib and Angular.js at the Rescue**

**-------------------------------------------------------------------------------**

%r {"imageWidth": "500px"} plot(iris, col = heat.colors(3))

---

%r {"imageWidth": "500px"} boxplot(Sepal.Width ~ Species, data=iris)

---

%r {"imageWidth": "400px"}

newiris <- iris

newiris$Species <- NULL

(kc <- kmeans(newiris, 3))

table(iris$Species, kc$cluster)

plot(newiris[c("Sepal.Length", "Sepal.Width")], col=kc$cluster)

points(kc$centers[,c("Sepal.Length", "Sepal.Width")], col=1:3, pch=8, cex=2)

---

%r

require(rCharts)

map3 <- Leaflet$new()

map3$setView(c(51.505, -0.09), zoom = 13)

map3$marker(c(51.5, -0.09), bindPopup = "<p> Hi. I am a popup </p>")

map3$marker(c(51.495, -0.083), bindPopup = "<p> Hi. I am another popup </p>")

map3$print("map3", include\_assets=TRUE, cdn=TRUE)

---

%r

library(googleVis)

geo = gvisGeoChart(Exports, locationvar = "Country", colorvar="Profit", options=list(Projection = "kavrayskiy-vii"))

print(geo, tag = 'chart')

---

%pyspark

import StringIO

def show(p):

img = StringIO.StringIO()

p.savefig(img, format='svg')

img.seek(0)

print "%html <div style='width:600px'>" + img.buf + "</div>"

---

%pyspark

import matplotlib.pyplot as plt

plt.clf()

plt.plot([1,2,3,44, 6, 1])

show(plt)

---

%pyspark

import numpy as np

import matplotlib.pyplot as plt

people = ('Tom', 'Dick', 'Harry', 'Slim', 'Jim')

y\_pos = np.arange(len(people))

performance = 3 + 10 \* np.random.rand(len(people))

error = np.random.rand(len(people))

plt.clf()

plt.barh(y\_pos, performance, xerr=error, align='center', alpha=0.4)

plt.yticks(y\_pos, people)

plt.xlabel('Performance')

plt.title('How fast do you want to go today?')

show(plt)

---

%pyspark

from pylab import figure, rand

from matplotlib.patches import Ellipse

import matplotlib.pyplot as plt

NUM = 250

plt.clf()

ells = [Ellipse(xy=rand(2)\*10, width=rand(), height=rand(), angle=rand()\*360)

for i in range(NUM)]

fig = figure()

ax = fig.add\_subplot(111, aspect='equal')

for e in ells:

ax.add\_artist(e)

e.set\_clip\_box(ax.bbox)

e.set\_alpha(rand())

e.set\_facecolor(rand(3))

ax.set\_xlim(0, 10)

ax.set\_ylim(0, 10)

show(plt)

---

%md

Hello ${Name=You!}

This is ${Day3=Monday,Monday|Tuesday|Wednesday}