Tecnológico Nacional de México

Instituto Tecnológico de Tijuana

Subdirección Académica

Departamento de Sistemas y Computación

Semestre: Febrero - Julio 2021

Materia:

Datos Masivos

Profesor:

Jose Christian Romero Hhernandez

Alumno:

Briseño Cota Raul Omar

Aceves Zamora Juan Antonio

Fecha:

15 de Abril del 2021

Evaluación 1

Cuestionario

1. **Comienza una simple sesión Spark**

Comenzamos importando librerías necesarias para construir la sesión y otras necesarias para el funcionamiento correcto de los siguientes incisos.

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

scala> import org.apache.spark.sql.functions._
import org.apache.spark.sql.functions._

scala> import spark.implicits._
import spark.implicits._

scala> val spark = SparkSession.builder().getOrCreate()
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@76929266
```

2. **Cargue el archivo Netflix Stock CSV, haga que Spark infiera los tipos de datos.**

Con el siguiente código debemos saber la ubicación del archivo csv que necesitamos

```
scala> val NetDF = spark.read.option("header", "true").option("inferSchema", "true")csv("C:\\Users\\brise\\Documents\\GitHub\\DatosMasivos\\Eva
luation 1\\Netflix_2011_2016.csv")
NetDF: org.apache.spark.sql.DataFrame = [Date: timestamp, Open: double ... 5 more fields]
```

3. **¿Cuáles son los nombres de las columnas?**

```
scala> val ColNames: Array[String] = NetDF.columns
ColNames: Array[String] = Array(Date, Open, High, Low, Close, Volume, Adj Close)

scala> ColNames.foreach(name => println(s"$name"))
Date
Open
High
Low
Close
Volume
Adj Close
```

4. **¿Cómo es el esquema?**

```
scala> NetDF.printSchema()
root
 |-- Date: timestamp (nullable = true)
 |-- Open: double (nullable = true)
 |-- High: double (nullable = true)
 |-- Low: double (nullable = true)
 |-- Close: double (nullable = true)
 |-- Volume: integer (nullable = true)
 |-- Adj Close: double (nullable = true)
```

5. **Imprime las primeras 5 columnas**

```
scala> NetDF.columns.take(5)
res2: Array[String] = Array(Date, Open, High, Low, Close)
```

6. **Usa describe() para aprender sobre el DataFrame**

```
scala> NetDF.describe().show()
+-------+------------------+------------------+------------------+------------------+------------------+------------------+
|summary|              Open|              High|               Low|             Close|            Volume|         Adj Close|
+-------+------------------+------------------+------------------+------------------+------------------+------------------+
|  count|              1259|              1259|              1259|              1259|              1259|              1259|
|   mean|230.39351086656092|233.97320872915006|226.80127876251044| 230.522453845909|2.5634836060365368E7|55.610540036536875|
| stddev|164.37456353264244| 165.9705082667129| 162.6506358235739|164.40918905512854| 2.306312683388607E7|35.186669331525486|
|    min|         53.990001|         55.480001|             52.81|              53.8|           3531300|          7.685714|
|    max|        708.900017|        716.159996|        697.569984|        707.610001|         315541800|        130.929993|
+-------+------------------+------------------+------------------+------------------+------------------+------------------+
```

7. **Crea un nuevo dataframe con una columna nueva llamada “HV Ratio” que es la relación entre el precio de la columna “High” frente a la columna “Volume” de acciones negociadas por un día. (Hint: Es una operación de columnas).**

```
scala> val NetDF2 = NetDF.withColumn("HV Ratio",NetDF("High")/NetDF("Volume"))
NetDF2: org.apache.spark.sql.DataFrame = [Date: timestamp, Open: double ... 6 more fields]

scala> NetDF2.show()
+-------------------+-----------------+-----------------+----------+-----------------+---------+------------------+--------------------+
|               Date|             Open|             High|       Low|            Close|   Volume|         Adj Close|            HV Ratio|
+-------------------+-----------------+-----------------+----------+-----------------+---------+------------------+--------------------+
|2011-10-24 00:00:00|       119.100002|120.28000300000001|115.100004|       118.839996|120460200|         16.977142|9.985040951285156E-7|
|2011-10-25 00:00:00|        74.899999|        79.390001| 74.249997|        77.370002|315541800|11.052857000000001|2.5159899892819 27E-7|
|2011-10-26 00:00:00|            78.73|        81.420001| 75.399997|        79.400002|148733900|         11.342857|5.474206014903126E-7|
|2011-10-27 00:00:00|        82.179998| 82.71999699999999| 79.249998|80.86000200000001| 71190000|11.551428999999999|1.161960907430818...|
|2011-10-28 00:00:00|        80.280002|        84.660002| 79.599999|84.14000300000001| 57769600|             12.02|1.465476686700271...|
|2011-10-31 00:00:00|83.63999799999999|        84.090002| 81.450002|        82.080003| 39653600|         11.725715|2.120614572195210...|
|2011-11-01 00:00:00|        80.109998|        80.999998|     78.74|        80.089997| 33016200|         11.441428|2.453341026526372E-6|
|2011-11-02 00:00:00|        80.709998|        84.400002| 80.109998|        83.389999| 41384000|         11.912857|2.039435578967717E-6|
|2011-11-03 00:00:00|        84.130003|        92.600003| 81.800003|        92.290003| 94685500|13.184285999999998| 9.77974483949496E-7|
|2011-11-04 00:00:00|91.46999699999999| 92.89000300000001| 87.749999|        90.019998| 84483700|             12.86|1.099502069629999...|
|2011-11-07 00:00:00|             91.0|        93.839998| 89.979997|        90.830003| 47485200|         12.975715|1.976194645910725...|
|2011-11-08 00:00:00|91.22999899999999|        92.600003| 89.650002|        90.470001| 31906000|         12.924286|2.902275528113834...|
|2011-11-09 00:00:00|        89.000001|        90.440001| 87.999998|        88.049999| 28756000|         12.578571|3.145082800111281E-6|
|2011-11-10 00:00:00|        89.290001| 90.29999699999999| 84.839999|85.11999899999999| 39614400|             12.16|2.279474054889131E-6|
|2011-11-11 00:00:00|        85.899997|        87.949997|      83.7|        87.749999| 38140200|         12.535714|2.305965805108520...|
|2011-11-14 00:00:00|        87.989998|             88.1|     85.45|        85.719999| 21811300|         12.245714|4.039190694731629...|
|2011-11-15 00:00:00|            85.15|        87.050003| 84.499998|        86.279999| 21372400|         12.325714|4.073010190713256...|
|2011-11-16 00:00:00|        86.460003|        86.460003| 80.890002|        81.180002| 34560400|11.597142999999999|2.501707242971725E-6|
|2011-11-17 00:00:00|            80.77|        80.999998| 75.789999|        76.460001| 52823400|         10.922857|1.533411291208063...|
|2011-11-18 00:00:00|             76.7|        78.999999| 76.039998|        78.059998| 34729100|         11.151428|2.274749388841058...|
+-------------------+-----------------+-----------------+----------+-----------------+---------+------------------+--------------------+
only showing top 20 rows
```

8. **¿Qué día tuvo el pico mas alto en la columna “Close”?**

```
scala> NetDF.groupBy(dayofweek(NetDF("Date")).alias("Day")).max("Close").sort(asc("Day")).show()
+---+-----------------+
|Day|       max(Close)|
+---+-----------------+
|  2|       707.610001|
|  3|       702.600006|
|  4|678.6099780000001|
|  5|       670.089996|
|  6|       680.599983|
+---+-----------------+
```

9. **Escribe con tus propias palabras en un comentario de tu código. ¿Cuál es el significado de la columna Cerrar “Close”?**

Es el valor al finalizar el mes

10. **¿Cuál es el máximo y mínimo de la columna “Volume”?**

```
scala> NetDF.select(max("Volume"), min("Volume")).show()
+-----------+-----------+
|max(Volume)|min(Volume)|
+-----------+-----------+
|  315541800|    3531300|
+-----------+-----------+
```

11. Con Sintaxis Scala/Spark $ conteste los siguiente:

a. ¿Cuántos días fue la columna “Close” inferior a $ 600?

```
scala> NetDF.filter($"Close"<600).count()
res7: Long = 1218
```

b. ¿Qué porcentaje del tiempo fue la columna “High” mayor que $ 500?

```
scala> (NetDF.filter($"High" > 500).count() * 1.0/ NetDF.count())*100
res8: Double = 4.924543288324067
```

c. ¿Cuál es la correlación de Pearson entre columna “High” y la columna “Volumen”?

```
scala> NetDF.select(corr("High","Volume")).show()
+--------------------+
|  corr(High, Volume)|
+--------------------+
|-0.20960233287942157|
+--------------------+
```

d. ¿Cuál es el máximo de la columna “High” por año?

```
scala> val yeardf = NetDF.withColumn("Year",year(NetDF("Date")))
yeardf: org.apache.spark.sql.DataFrame = [Date: timestamp, Open: double ... 6 more fields]

scala> val yearmaxs = yeardf.select($"Year",$"High").groupBy("Year").max()
yearmaxs: org.apache.spark.sql.DataFrame = [Year: int, max(Year): int ... 1 more field]

scala> val res = yearmaxs.select($"Year",$"max(High)")
res: org.apache.spark.sql.DataFrame = [Year: int, max(High): double]

scala> res.orderBy("Year").show()
+----+------------------+
|Year|         max(High)|
+----+------------------+
|2011|120.28000300000001|
|2012|        133.429996|
|2013|        389.159988|
|2014|        489.290024|
|2015|        716.159996|
|2016|129.28999299999998|
+----+------------------+
```

e. **¿Cuál es el promedio de columna “Close” para cada mes del calendario?**

```
scala> val monthdf = NetDF.withColumn("Month",month(NetDF("Date")))
monthdf: org.apache.spark.sql.DataFrame = [Date: timestamp, Open: double ... 6 more fields]

scala> val monthavgs = monthdf.select($"Month",$"Close").groupBy("Month").mean()
monthavgs: org.apache.spark.sql.DataFrame = [Month: int, avg(Month): double ... 1 more field]

scala> monthavgs.select($"Month",$"avg(Close)").orderBy("Month").show()
+-----+------------------+
|Month|       avg(Close)|
+-----+------------------+
|    1|212.22613874257422|
|    2| 254.1954634020619|
|    3| 249.5825228971963|
|    4|246.97514271428562|
|    5|264.37037614150944|
|    6| 295.1597153490566|
|    7|243.64747528037387|
|    8|195.25599892727263|
|    9|206.09598121568627|
|   10|205.93297300900903|
|   11| 194.3172275445545|
|   12| 199.3700942358491|
+-----+------------------+
```

Código

```
//1. Sign in to Spark.
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._
import spark.implicits._
val spark = SparkSession.builder().getOrCreate()

//2.Load Netflix Stock CSV.
val NetDF = spark.read.option("header", "true").option("inferSchema",
"true")csv("C:\\Users\\brise\\Documents\\GitHub\\DatosMasivos\\Evaluatio
n 1\\Netflix_2011_2016.csv")

//3. What are the names of the columns?
val ColNames: Array[String] = NetDF.columns
ColNames.foreach(name => println(s"$name"))

//4. What is the name of the scheme?
NetDF.printSchema()

//5. Print the first 5 columns.
NetDF.columns.take(5)

//6. Use describe()
NetDF.describe().show()
```

```
//7.  Create a new column called "HV Ratio" which is the relationship
between the price
// of the "High" column in front of the "Volume" column of shares traded
for one day.
// Added the new column "HV Ratio" which will have the result of the
division of "High" by "Volume".
val NetDF2 = NetDF.withColumn("HV Ratio",NetDF("High")/NetDF("Volume"))
NetDF2.show()

//8.What day had the highest peak in the "close" column?
NetDF.groupBy(dayofweek(NetDF("Date")).alias("Day")).max("Close").sort(a
sc("Day")).show()

//9. What is the meaning of the Close column "Close"?
// Answer: It is the end of the month.

//10. What is the maximum and minimum of the column "Volume"?
NetDF.select(max("Volume"), min("Volume")).show()


//11.With Scala / Spark $ syntax answer the following:
//a) How many days was the "Close" column less than $ 600?
NetDF.filter($"Close"<600).count()


// b)What percentage of the time was the "High" column greater than $
500?
(NetDF.filter($"High" > 500).count() * 1.0/ NetDF.count())*100

// c) What is the Pearson correlation between Column "High" and column
"Volume"?
NetDF.select(corr("High","Volume")).show()


// d) What is the maximum in the "High" column per year?
// The column "Year" of the data "Date" was added, in the variable
"yeardf"
val yeardf = NetDF.withColumn("Year",year(NetDF("Date")))
// The year and the maximum of "High" were selected from the variable
"yeardf", from the maximum to the minimum in the years.
val yearmaxs = yeardf.select($"Year",$"High").groupBy("Year").max()
// The maximum of the column "High" per year was returned.
val res = yearmaxs.select($"Year",$"max(High)")
res.orderBy("Year").show()
```

```
// e) What is the average of the "Close" column for each calendar month?
// The column "Month" of the data "Date" was added, in the variable
"monthdf".
val monthdf = NetDF.withColumn("Month",month(NetDF("Date")))
// The month and the average of "Close" were selected from the variable
"monthdf".
val monthavgs =
monthdf.select($"Month",$"Close").groupBy("Month").mean()
// The average of the "Close" column for each calendar month was
selected and displayed.
monthavgs.select($"Month",$"avg(Close)").orderBy("Month").show()
```

Conclusión:

Nuestra conclusión fue que estas herramientas en realidad si ayudan de una manera eficaz y eficiente a gestionar un “dataframe” ya que de manera específica se puede obtener el resultado que se desea saber sin necesidad de tanta fórmula y de una manera dinámica e inusual.