TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

NOMBRE DE LOS ALUMNOS:

RAYMUNDO HIRALES LAZARENO (N. CONTROL: 17212339)

PAULA ANDREA RAMOS VERDIN (N. CONTROL: 18210721)

Carrera: Ingeniería Informática

MATERIA: Datos Masivos

PROFESOR: JOSE CHRISTIAN ROMERO HERNANDEZ

Practica Evaluatoria U1

FECHA: 22/03/22

Introducción

En el presente documento se expondrá a detalle la práctica evaluatoria de la unidad 1, donde veremos la utilización de Scala y Spark en dataframes con el agregado de un archivo .csv

Desarrollo

//RAYMUNDO HIRALES LAZARENO - 17212339 //PAULA ANDREA RAMOS VERDIN - 18210721 //Exam Unit 1 - 22/03/22

//1.- Comienza con una simple sesion de spark import org.apache.spark.sql.SparkSession val spark = SparkSession.builder().getOrCreate()

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

scala> val spark = SparkSession.builder().getOrCreate()
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@36bd07ec
```

//2.- Cargue el Archivo Netflix Stock CSV, haga que Spark infiere los tipos de datos

val net = spark.read.option("header", "true").option ("inferSchema", "true") csv ("Netflix_2011_2016.csv") net net.printSchema()

```
scala> val net = spark.read.option("header", "true").option ("inferSchema", "true") csv ("Netflix_2011_2016.csv")
net: org.apache.spark.sql.DataFrame = [Date: timestamp, Open: double ... 5 more fields]

scala> net
res0: org.apache.spark.sql.DataFrame = [Date: timestamp, Open: double ... 5 more fields]

scala> net.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)
```

//3.- ¿Cuáles son los nombres de las columnas? net.columns

```
scala> net.columns
res2: Array[String] = Array(Date, Open, High, Low, Close, Volume, Adj Close)
```

//4.- ¿Cómo es el esquema? net.printSchema()

```
scala> net.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. var col = 0 while(col < 5) { println(net.columns(col)) col = col + 1 }

```
scala> var col = 0
col: Int = 0

scala> while(col < 5)
     | {
     |     println(net.columns(col))
     |     col = col + 1
     | }
Date
Open
High
Low
Close
```

//6.-Usa describe () para aprender sobre el DataFrame. net.describe().show()

```
scala> net.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 que //existe entre el precio de la columna “High” frente a la columna “Volumen” de acciones //negociadas por un día. Hint - es una operación import org.apache.spark.sql.Column val nData = net.withColumn("HVRatio", net("High")/net("Volume")) nData.show()

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

scala> val nData = net.withColumn("HVRatio", net("High")/net("Volume"))
nData: org.apache.spark.sql.DataFrame = [Date: timestamp, Open: double ... 6 more fields]

scala> nData.show()
+-------------------+-----------------+-----------------+---------+-----------------+---------+-----------------+--------------------+
|               Date|             Open|             High|      Low|            Close|   Volume|        Adj Close|             HVRatio|
+-------------------+-----------------+-----------------+---------+-----------------+---------+-----------------+--------------------+
|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|2.515989892819127E-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 “Open”? val maxp = nData.orderBy(desc("Open")) maxp.select("Date").limit(1).show()

```
scala> val maxp = nData.orderBy(desc("Open"))
maxp: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Date: timestamp, Open: double ... 6 more fields]

scala> maxp.select("Date").limit(1).show()
+-------------------+
|               Date|
+-------------------+
|2015-07-14 00:00:00|
+-------------------+
```

//9-.¿Cuál es el significado de la columna Cerrar “Close” en el contexto de información financiera, expliquelo no hay que codificar nada? /* Cuando el precio de la columna High sube parece ser lo mismo para la columna Close solo que esta siempre en menor cantidad que la High, lo que quiere decir que a medida que High sube lo mas probables es que Close tambien lo haga pero en menor cantidad*/

//10-.¿Cuál es el máximo y mínimo de la columna “Volumen”? nData.select(max("Volume")).show() nData.select(min("Volume")).show()

```
scala> nData.select(max("Volume")).show()
+-----------+
|max(Volume)|
+-----------+
|  315541800|
+-----------+


scala> nData.select(min("Volume")).show()
+-----------+
|min(Volume)|
+-----------+
|    3531300|
+-----------+
```

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

//a-.Cuantos dias fue la columna "Close" inferior a $600? nData.filter($"Close"<600).count()

```
scala> //a-.Cuantos dias fue la columna "Close" inferior a $600?

scala> nData.filter($"Close"<600).count()
res13: Long = 1218
```

//b-.Que porcentaje del tiempo fue la columna "High" mayor que $500? val tiempo:Double = nData.filter($"High">500).count() val porcentaje:Double = (tiempo*100)/1259

```
scala> //b-.Que porcentaje del tiempo fue la columna "High" mayor que $500?

scala> val tiempo:Double = nData.filter($"High">500).count()
tiempo: Double = 62.0

scala> val porcentaje:Double = (tiempo*100)/1259
porcentaje: Double = 4.924543288324067
```

//c-.Cual es la correlacion de pearson entre columna "High" y la columna "Volumen"?
nData.select(corr("High","Volume").alias("Correlacion")).show()

```
scala> //c-.Cual es la correlacion de pearson entre columna "High" y la columna "Volumen"?

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

//d-.Cual es el maximo de la columna "High" por año?
nData.groupBy(year(nData("Date")).alias("Year")).max("High").sort(asc("Year")).show()

```
scala> //d-.Cual es el maximo de la columna "High" por año?

scala> nData.groupBy(year(nData("Date")).alias("Year")).max("High").sort(asc("Year")).show()
+----+------------------+
|Year|         max(High)|
+----+------------------+
|2011|120.28000300000001|
|2012|        133.429996|
|2013|        389.159988|
|2014|        489.290024|
|2015|        716.159996|
|2016|129.28999299999998|
+----+------------------+
```

//e-.Cual es el promedio de la columna "Close" para cada mes del calendario?
nData.groupBy(month(nData("Date")).alias("Month")).avg("Close").sort(asc("Month")).show()

```
scala> //e-.Cual es el promedio de la columna "Close" para cada mes del calendario?

scala> nData.groupBy(month(nData("Date")).alias("Month")).avg("Close").sort(asc("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|
+-----+------------------+
```

Conclusión

Se puede decir que lo mas importante de esta practica es la familiarizacion de los comandos basicos de scala con dataframes podemos decir que a pesar de que a simple vista esta herramienta podría parecer de lo más complicado en realidad es una herramienta de lo más útil, resulto bastante satisfactorio y enriquecedor.

link de youtube

https://youtu.be/BeJv1RX9HwA