Tecnológico Nacional de México

Instituto Tecnológico de La Laguna

Big Data

INGENIERIA EN SISTEMAS COMPUTACIONALES

JOSE LUIS VERDE SANCHEZ

19130002

Torreón, Coahuila

30/11/2022

Indice general

Introducción

En esta investigación se mostrará a utilizar Spark, usándolo para generar sesiones, abrir varios archivos al mismo tiempo, generar rutas, DataFrames y su implementación en un uso real para investigación de datos.

Además, se mostrará la forma de hacer limpieza y cambios temporales en archivos CSV para obtener datos mas limpios, ya sea en cambio de cabeceras, formato de datos, eliminación de columnas, conteo de filas y columnas, además de filtrado de datos para búsquedas en específico.

Con estas mismas modificaciones de utiliza la implementación de formulas para obtener promedios, min y max, esto mismo se puede hacer en agrupaciones.

Desarrollo

Librerias utilizadas

```
import findspark
findspark.init()
findspark.find()
import pyspark
#Importamos sql
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
```

En este caso se utiliza findspark para poder importar pyspark

PySpark.sql Session se utiliza para poder generar sql de forma local

Creacion de cluster local y session

```
#Creamos cluster local
#sc = pyspark.SparkContext(appName="EstacionesCOAH")
#print(type(sc))
#sc.appName
#NO SE UTILIZA, HARE TODO CON PYSPARK SQL
```

```
#Creacion de la session
spark = SparkSession.builder.appName("EstacionesCOAH").getOrCreate()
```

La primer celda muestra la forma de crear un cluster local, este se utiliza en caso de que no se requiera utilizar sql, en este caso como se utilizan funciones de sql solamente necesito la session creada por la libreria pyspark.sql.sparksession

Generacion de DataFrame

```
#Base de la ruta para cada lugar
ruta = "C:\\Users\\verde\\Documents\\AGO2022\\BIGDATA\\JUPYTER\\CLIMA\\Estacion_"
rutasc = '__90_dias.csv'

#Genero la ruta de cada archivo
CANDELA = ruta+'CANDELA'+rutasc                          #0
CUATROCIENEGAS=ruta+'CUATROCIENEGAS'+rutasc              #1
MONCLOVA_ESIME=ruta+'MONCLOVA_ESIME'+rutasc              #2
MORELOSMUZQUIZ=ruta+'MORELOSMUZQUIZ'+rutasc              #3
NUEVAROSITA=ruta+'NUEVAROSITA'+rutasc                    #4
OCAMPO=ruta+'OCAMPO'+rutasc                              #5
OCAMPOSMN=ruta+'OCAMPOSMN'+rutasc                        #6
PIEDRAS_NEGRAS=ruta+'PIEDRAS_NEGRAS'+rutasc              #7
SALTILLO=ruta+'SALTILLO'+rutasc                          #8
SANTACECILIA=ruta+'SANTACECILIA'+rutasc                  #9
TORREON=ruta+'TORREON'+rutasc                            #10
VENUSTIANOCARRANZASMN=ruta+'VENUSTIANOCARRANZASMN'+rutasc
```

Lo primero que se necesita para generar un DataFrame son datos, en este caso se van a obtener desde CSVs, entonces, se declara la ruta de los archivos, como los CSVs proporcionados solamente variaban por el nombre del lugar se crean variables con los nombres, otra opcion pudo ser iterarlos y generar un array de rutas.

```
#Hago una lista para poder iterarlo mas facilmente
lugares = [CANDELA,CUATROCIENEGAS,MONCLOVA_ESIME,MORELOSMUZQUIZ,NUEVAROSITA,OCAMPO,OCAMPOSMN,PIEDRAS_NEGRAS,
        SALTILLO,SANTACECILIA,TORREON,VENUSTIANOCARRANZASMN]
lugarestxt = ['CANDELA','CUATROCIENEGAS','MONCLOVA_ESIME','MORELOSMUZQUIZ','NUEVAROSITA','OCAMPO','OCAMPOSMN','PIEDRAS_NEGRAS',
         'SALTILLO','SANTACECILIA','TORREON','VENUSTIANOCARRANZASMN']

#Creo los dataframes
i=0
df=['','','','','','','','','','','','','']
for lugar in lugares:
    df[i] = spark.read.format("csv").option("inferSchema", False).option("header", True).option("sep",",").load(lugares[i])
    i=i+1
```

Como de todas formas necesitaba un array con los nombres de cada lugar, se decidio que esta forma seria la mas simple de entender al leerse.

Despues de generar el array de variables de ruta se procede a generar los dataframe, primero se tuvo que inicializar un array [12] para almacenar cada dataframe, luego con una iteracion se repite el comando de creacion de dataframe.

Para generar este comando se utilizan los siguientes parametros:

- .format: Como se van a utilizar puros CSVs no se necesita iterar ningun cambio de formato
- ‘inferSchema’ : Es la lectura de formatos, para este trabajo se coloca en False para poder mostrar la implementacion de cambio de formato, ademas, de esta forma es mas rapido de leer el CSV.

- ‘header’: Esta opcion pregunta si la primera fila se utilizara como nombre de columnas.
- ‘sep’: Se especifica la separacion utilizada en los archivos, en este caso ‘,’.
- .load: Aquí es donde se especifica la ruta del archivo que se manda a spark.read

```
#Verifico informacion 0=CANDELA,etc
df[2].show()
#Como vemos se lee mal el csv dado que el "header" solamente tiene servicio meteorologico.
```

```
+--------------------------------+
|`Servicio Meteorol�gico Nacional|
+--------------------------------+
|                  `Red: SMN-ESMA|
|            `Estaci�n: MONCLO...|
|            `Estado: Coahuila...|
|            `Municipio: Monclova|
|            `Latitud (N): 26....|
|            `Longitud (O): -1...|
|                   `Altitud: 618|
|                                `|
|                     Fecha Local|
|             2022-08-31 20:30:00|
|             2022-08-31 20:20:00|
|             2022-08-31 20:10:00|
|             2022-08-31 20:00:00|
|             2022-08-31 19:50:00|
|             2022-08-31 19:40:00|
|             2022-08-31 19:30:00|
|             2022-08-31 19:20:00|
|             2022-08-31 19:10:00|
|             2022-08-31 19:00:00|
|             2022-08-31 18:50:00|
+--------------------------------+
only showing top 20 rows
```

```
#Como todas inician a mostrar los datos en la fila 10, en lugar de borrar datos de identificacion del CSV
#podemos brincar a esa en especifico y mantener el csv aun default.
i=0
for lugar in lugares:
    df[i] = spark.read.format("csv").option("inferSchema", False).option("header", True).option("sep",",").\
    option("comment",'`').load(lugares[i])
    i=i+1
```

Se verifica que se haya generado el DataFrame de manera correcta, como se ve en la primer celda el csv contenia filas de identificacion donde se mostraba red utilizada, localizacion y mas datos que realmente no se van a utilizar. Como todos estos datos uniciaban con ‘`’ se utiliza la opcion ‘comment’ la cual te deja definir un carácter que toma filas como comentarios.

```
#Se manda llamar el df [3] = MORELOS
df[3].show()
#Se observa headers mal escritos y una columna _c11 generada porque los csv estan mal estructurados
```

```
+-------------------+-------------------+--------------------------+--------------------------+----------------------+--
----------------------+-----------------------+-------------------+-----------------------+-----------------+------------
----------+----+
|        Fecha Local|          Fecha UTC|Direcci�n del Viento (grados)|Direcci�n de r�faga (grados)|Rapidez de viento (km/h)|
Rapidez de r�faga (km/h)|Temperatura del Aire (�C)|Humedad relativa (%)|Presi�n Atmosf�rica (hpa)|Precipitaci�n (mm)|Radiac
i�n Solar (W/m�)|_c11|
+-------------------+-------------------+--------------------------+--------------------------+----------------------+--
----------------------+-----------------------+-------------------+-----------------------+-----------------+------------
----------+----+
|2022-11-15 09:10:00|2022-11-15 15:10:00|                      75.0|                      87.0|                   6.1|
9.7|                    9.9|                 49|                    969.2|                 0|                292|null|
|2022-11-15 09:00:00|2022-11-15 15:00:00|                      76.0|                      84.0|                   4.9|
8.1|                    9.7|                 50|                      969|                 0|                204|null|
|2022-11-15 08:50:00|2022-11-15 14:50:00|                      84.0|                      96.0|                   4.4|
8.1|                   10.1|                 49|                    968.9|                 0|                342|null|
|2022-11-15 08:40:00|2022-11-15 14:40:00|                      76.0|                      86.0|                   2.6|
5.3|                    9.4|                 53|                    968.7|                 0|                252|null|
|2022-11-15 08:30:00|2022-11-15 14:30:00|                      71.0|                     102.0|                   1.1|
2.3|                    8.9|                 59|                    968.7|                 0|                120|null|
|2022-11-15 08:20:00|2022-11-15 14:20:00|                      77.0|                      45.0|                   1.7|
2.5|                    8.4|                 60|                    968.5|                 0|                 85|null|
|2022-11-15 08:10:00|2022-11-15 14:10:00|                     112.0|                     266.0|                   1.2|
2.9|                    8.2|                 62|                    968.5|                 0|                 84|null|
|2022-11-15 08:00:00|2022-11-15 14:00:00|                     152.0|                     163.0|                     1|
1.7|                    8.1|                 66|                    968.4|                 0|                117|null|
|2022-11-15 07:50:00|2022-11-15 13:50:00|                     152.0|                     166.0|                   1.7|
2.8|                    7.2|                 73|                    968.2|                 0|                 90|null|
|2022-11-15 07:40:00|2022-11-15 13:40:00|                     147.0|                     160.0|                     1|
1.6|                    6.6|                 73|                    968.1|                 0|                 72|null|
|2022-11-15 07:30:00|2022-11-15 13:30:00|                     147.0|                     171.0|                     1|
1.8|                    5.4|                 77|                      968|                 0|                 45|null|
|2022-11-15 07:20:00|2022-11-15 13:20:00|                     149.0|                     151.0|                   1.3|
2.1|                      5|                 78|                    967.9|                 0|                 18|null|
```

Ahora se manda a llamar denuevo para comprobar que haya saltado las lineas no usables. Como se puede observar el DataFrame se estructura de forma correcta pero tiene fallos en Columnas, headers y falta de formato.

Limpieza de los DataFrames

```
#Vemos cada header.
df[0].printSchema()
df2=['','','','','','','','','','','','']
#Confirmo que _c11 es un extra que no contiene ningun dato.
```

```
root
 |-- Fecha Local: string (nullable = true)
 |-- Fecha UTC: string (nullable = true)
 |-- Temperatura del Aire (�C): string (nullable = true)
 |-- Precipitaci�n (mm): string (nullable = true)
 |-- Humedad relativa (%): string (nullable = true)
 |-- Presi�n Atmosf�rica (hpa): string (nullable = true)
 |-- Radiaci�n Solar (W/m�): string (nullable = true)
 |-- Direcci�n del Viento (grados): string (nullable = true)
 |-- Rapidez de viento (km/h): string (nullable = true)
 |-- Direcci�n de r�faga (grados): string (nullable = true)
 |-- Rapidez de r�faga (km/h): string (nullable = true)
 |-- _c11: string (nullable = true)
```

Se imprime el Schema del dataframe generado y se inicializa un nuevo array para guardar los DF modificados.

En este caso se observa que los headers contienen caracteres no leibles, todos los datos tienen formato string por el uso de ‘inferSchema’ y como los datos en los CSVs terminaban con ‘,’ se esta generando una columna extra la cual no contiene ningun dato.

```python
#Correccion a los header names
for i in range(12):
    df2[i] = df[i].withColumnRenamed("Temperatura del Aire (ï¿½C)","Temperatura del Aire")\
            .withColumnRenamed("Precipitaci￿n (mm)","Precipitacion")\
            .withColumnRenamed("Presi￿n Atmosf￿rica (hpa)","Presion Atmosferica")\
            .withColumnRenamed("Radiaci￿n Solar (W/m￿)","Radiacion Solar")\
            .withColumnRenamed("Direcci￿n del Viento (grados)","Direccion del Viento")\
            .withColumnRenamed("Direcci￿n de r￿faga (grados)","Direccion de rafaga")\
            .withColumnRenamed("Rapidez de r￿faga (km/h)","Rapidez de rafaga")
```

```python
#Verificacion de lo anterior
df2[11].printSchema()
```

```
root
 |-- Fecha Local: string (nullable = true)
 |-- Fecha UTC: string (nullable = true)
 |-- Direccion del Viento: string (nullable = true)
 |-- Direccion de rafaga: string (nullable = true)
 |-- Rapidez de viento (km/h): string (nullable = true)
 |-- Rapidez de rafaga: string (nullable = true)
 |-- Temperatura del Aire: string (nullable = true)
 |-- Humedad relativa (%): string (nullable = true)
 |-- Presion Atmosferica: string (nullable = true)
 |-- Precipitacion: string (nullable = true)
 |-- Radiacion Solar: string (nullable = true)
 |-- _c11: string (nullable = true)
```

Se utiliza una iteracion con withColumnRenamed, la cual de parametro se le manda la el header que quieres cambiar y el nombre que quieres ponerle, esta funcion regresa un nuevo DF, por lo cual, se necesita almacenar, no lo cambia en el actual.

En la segunda celda se manda a llamar el segundo DF para verificar que se cambiaran los headers.

```
#Eliminamos la extra con el comando drop y guardandolo en el mismo dataframe
for i in range(12):
    df2[i] = df2[i].drop(df2[i]._c11)
```

```
#Comprobamos denuevo
df2[1].printSchema()
```

```
root
 |-- Fecha Local: string (nullable = true)
 |-- Fecha UTC: string (nullable = true)
 |-- Direccion del Viento: string (nullable = true)
 |-- Direccion de rafaga: string (nullable = true)
 |-- Rapidez de viento (km/h): string (nullable = true)
 |-- Rapidez de rafaga: string (nullable = true)
 |-- Temperatura del Aire: string (nullable = true)
 |-- Humedad relativa (%): string (nullable = true)
 |-- Presion Atmosferica: string (nullable = true)
 |-- Precipitacion: string (nullable = true)
 |-- Radiacion Solar: string (nullable = true)
```

Se elimina la columna fantasma en cada DF, despues se manda a llamar el schema para verificar. El comando .drop funciona de forma similar que el anterior, tambien retorna un nuevo DF, por lo cual se hace override del DF2, como parametro .drop pide el nombre de la columna o columnas que se desean borrar, en este caso puede ingresarse un string o mandarla llamar.

```
#Ahora vemos como va quedando el formato.
#Ya no tiene datos null y todos los header pueden leerse.
df2[1].show(3)
```

```
+-------------------+-------------------+-------------------+-------------------+------------------------+-----------------+--
-----------------+--------------------+-------------------+-------------------+-------------+---------------+
|        Fecha Local|          Fecha UTC|Direccion del Viento|Direccion de rafaga|Rapidez de viento (km/h)|Rapidez de rafaga|Te
mperatura del Aire|Humedad relativa (%)|Presion Atmosferica|Precipitacion|Radiacion Solar|
+-------------------+-------------------+-------------------+-------------------+------------------------+-----------------+--
-----------------+--------------------+-------------------+-------------------+-------------+---------------+
|2022-11-15 09:10:00|2022-11-15 15:10:00|              110.0|              133.0|                    24.7|             41.7|
12.8|                32|              938|                0|          377|
|2022-11-15 09:00:00|2022-11-15 15:00:00|              103.0|              126.0|                    22.3|             40.5|
12.4|                32|              937.9|                0|          332|
|2022-11-15 08:50:00|2022-11-15 14:50:00|              111.0|              112.0|                    22.8|             38.9|
12.2|                33|              937.9|                0|          295|
+-------------------+-------------------+-------------------+-------------------+------------------------+-----------------+--
-----------------+--------------------+-------------------+-------------------+-------------+---------------+
only showing top 3 rows
```

Se verifica el formato, ahora se tiene DFs usables.

Uso de DataFrames

Filtrado de datos

```
#Ahora que tenemos el DF usable podemos comenzar, primero veremos de cuantos datos estamos hablando.
datos=0
for i in range(12):
    datos=datos+df[i].count()
datos
```

```
82557
```

Ahora para iniciar a utilizar el DF se hace un conteo de rows para saber de cuantos datos estamos hablando.

```
#Como primer ejemplo vamos a ver la diferencia de lluvia en los lugares.
#Con el filter se eliminan los null de la tabla.
for i in range(12):
    print(lugarestxt[i])
    df2[i].groupBy(df2[i]['Precipitacion']).count().filter("`Precipitacion` != ''").sort(desc('count')).show()
```

Para iniciar un filtrado de datos se desea saber cuantos dias no llovio a comparacion de los que si llovio, entonces, se agrupa por la columna ‘Precipitacion’ y se hace un conteo de los valores, despues se filtran los datos donde Precipitacion es null, se ordenan por la columna count y se muestra.

```
CUATROCIENEGAS
+-------------+-----+
|Precipitacion|count|
+-------------+-----+
|            0|12857|
|          0.2|   23|
|          0.4|   16|
|          0.6|    6|
|          0.8|    6|
|          1.2|    3|
|            1|    3|
|          1.4|    2|
|          2.6|    2|
|          1.6|    2|
|          2.2|    2|
|          3.4|    1|
|            4|    1|
|          8.8|    1|
|            2|    1|
|          3.2|    1|
|         15.2|    1|
+-------------+-----+
```

```
MONCLOVA_ESIME
+-------------+-----+
|Precipitacion|count|
+-------------+-----+
|            0|   72|
|         0.51|    3|
|         1.02|    2|
|         3.05|    1|
|         2.29|    1|
|         1.52|    1|
|         4.32|    1|
+-------------+-----+
```

```
MORELOSMUZQUIZ
+-------------+-----+
|Precipitacion|count|
+-------------+-----+
|            0|12395|
|          0.2|  113|
|          0.4|   62|
|          0.8|   30|
|          0.6|   30|
|          1.2|   15|
|          1.6|   12|
|            1|   10|
|          2.4|    4|
|          1.4|    4|
|          3.2|    4|
|          1.8|    3|
|          2.2|    3|
|          5.6|    2|
|          3.6|    2|
|          3.8|    2|
|          2.8|    2|
|          4.6|    2|
|            4|    2|
|          2.6|    1|
+-------------+-----+
only showing top 20 rows
```

En la primer tabla se observa que no llovio en 12857 de las casi 13000 veces que se obtuvieron datos del lugar.

Generacion DataFrames temporales

```
#Para hacer algunas cosas usaremos solamente saltillo [8]. //TORREON NO TENIA DATOS//
#Se usara un solo dataset para hacer mas facil leer los resultados...
cols=['Fecha UTC', 'Temperatura del Aire','Presion Atmosferica', 'Precipitacion']
saltillo= df2[8].select(*cols)
saltillo.show()
```

```
+-------------------+--------------------+-------------------+-------------+
|          Fecha UTC|Temperatura del Aire|Presion Atmosferica|Precipitacion|
+-------------------+--------------------+-------------------+-------------+
|2022-11-15 15:10:00|                11.9|              826.7|            0|
|2022-11-15 15:00:00|                11.9|              826.7|            0|
|2022-11-15 14:50:00|                11.6|              826.7|            0|
|2022-11-15 14:40:00|                10.8|              826.7|            0|
|2022-11-15 14:30:00|                10.8|              826.5|            0|
|2022-11-15 14:20:00|                10.7|              826.4|            0|
|2022-11-15 14:10:00|                 9.9|              826.4|            0|
|2022-11-15 14:00:00|                 8.9|              826.4|            0|
|2022-11-15 13:50:00|                 7.7|              826.2|            0|
|2022-11-15 13:40:00|                 6.1|               826|            0|
|2022-11-15 13:30:00|                 5.5|              825.9|            0|
|2022-11-15 13:20:00|                   5|              825.8|            0|
|2022-11-15 13:10:00|                 4.5|              825.7|            0|
|2022-11-15 13:00:00|                   4|              825.7|            0|
|2022-11-15 12:50:00|                 4.1|              825.6|            0|
|2022-11-15 12:40:00|                 3.5|              825.5|            0|
|2022-11-15 12:30:00|                 3.4|              825.3|            0|
|2022-11-15 12:20:00|                 3.3|              825.2|            0|
|2022-11-15 12:10:00|                 3.5|               825|            0|
|2022-11-15 12:00:00|                 3.7|              824.9|            0|
+-------------------+--------------------+-------------------+-------------+
only showing top 20 rows
```

Se genera un DF temporal para hacer prueba de funciones en DF individual para poder despues implementarlos en iteraciones.

En este caso se crea una lista con los nombres de columna que se quieren utilizar, en mi caso solamente queria la Fecha UTC, Temperatura del Aire, Presion Atmosferica y Precipitacion.

Despues con .select se le manda la lista la cual genera un nuevo DF con solo las columnas mandadas.

Cambio de formato de datos

```
from pyspark.sql.types import *

#Ahora por ejemplo si queremos sacar temp max tendriamos que transformar la columna de temp a float
saltillo = saltillo.withColumn("Temperatura del Aire",saltillo["Temperatura del Aire"].cast("float"))

#Comprobamos que se cambio correctamente
saltillo.dtypes
```

```
[('Fecha UTC', 'string'),
 ('Temperatura del Aire', 'float'),
 ('Presion Atmosferica', 'string'),
 ('Precipitacion', 'string')]
```

Con el uso de la librería .types se utiliza la funcion .cast la cual permite cambiar el formato de una columna, para el ejemplo se utiliza la columna Temperatura del Aire y se cambia de ‘string’ a ‘float’, este comando genera un nuevo DF asi que se tiene que almacenar.

Ahora se verifica con el comando .dtypes que nos da la informacion del formato de cada columna.

Uso de formulas en DataFrames individuales

```
tmax=saltillo.approxQuantile("Temperatura del Aire",[0.9999999],0.000001) #Con la funcion quantile

str(tmax)
```

```
'[32.20000076293945]'
```

Para iniciar a usar formulas usaremos approxQuantile, la cual te acomoda la columna y te permite ver el minimo, max o cualquier lugar dado, esta funcion pide como parametros

(columna,valor percentil [0,min;0.5, media;1,max],tolerancia)

```
#Ahora con la funcion .max()
saltillo.groupBy("Precipitacion").max("Temperatura del Aire").sort(desc("max(Temperatura del Aire)")).show()
#En esta funcion por el groupby podemos hacer mas cosas que con quantile, aqui podemos ver la temp maxima por cada
#Diferente lluvia, donde nos damos cuenta que entre mas fuerte este lloviendo mas baja es la temperatura.
#Aparte se confirmo que la temp mas caliente si fue 32.2 grados centigrados
```

```
+------------+-----------------------+
|Precipitacion|max(Temperatura del Aire)|
+------------+-----------------------+
|           0|                   32.2|
|         0.2|                   26.9|
|         0.8|                   23.7|
|         0.6|                   22.1|
|         2.2|                   21.4|
|           1|                   21.1|
|         1.8|                   20.2|
|         0.4|                   19.9|
|         1.6|                   19.7|
|         1.2|                   19.4|
|         3.2|                   18.8|
|         5.6|                   18.5|
|         2.4|                   18.1|
|         1.4|                   17.3|
|         3.6|                   17.0|
|           3|                   16.8|
|           2|                   16.8|
|         2.6|                   16.8|
|         3.4|                   16.4|
|         2.8|                   16.0|
+------------+-----------------------+
only showing top 20 rows
```

Para usar una formula que muestre mas datos se utilizara max ahora de forma individual, primero se van a agrupar los datos por precipitacion para verificar si hay cambios de temperatura si en la medicion estaba lloviendo o no. Al verificar se muestra que la temperatura maxima registrada fue cuando no se estaba lloviendo.

```
saltillo.describe('Temperatura del Aire').show()
#Con describe nos ahorramos el llamar independientemente cada metodo como anteriormente
#Aqui se ve el promedio de temp, 18, el minimo y el max, mas el numero de datos que se leyeron.
```

```
+-------+--------------------+
|summary|Temperatura del Aire|
+-------+--------------------+
|  count|               12945|
|   mean|   18.05146389119917|
| stddev|   5.637346465700087|
|    min|                 2.1|
|    max|                32.2|
+-------+--------------------+
```

Generacion de columnas

```
#Se genera un temporal con la pura fecha para poder hacer pruebas por dias.
temp=saltillo.withColumn("FECHA",to_date("Fecha UTC"))
temp.show()
```

```
+-------------------+--------------------+------------------+-------------+----------+
|          Fecha UTC|Temperatura del Aire|Presion Atmosferica|Precipitacion|     FECHA|
+-------------------+--------------------+------------------+-------------+----------+
|2022-11-15 15:10:00|                11.9|             826.7|            0|2022-11-15|
|2022-11-15 15:00:00|                11.9|             826.7|            0|2022-11-15|
|2022-11-15 14:50:00|                11.6|             826.7|            0|2022-11-15|
|2022-11-15 14:40:00|                10.8|             826.7|            0|2022-11-15|
|2022-11-15 14:30:00|                10.8|             826.5|            0|2022-11-15|
|2022-11-15 14:20:00|                10.7|             826.4|            0|2022-11-15|
|2022-11-15 14:10:00|                 9.9|             826.4|            0|2022-11-15|
|2022-11-15 14:00:00|                 8.9|             826.4|            0|2022-11-15|
|2022-11-15 13:50:00|                 7.7|             826.2|            0|2022-11-15|
|2022-11-15 13:40:00|                 6.1|               826|            0|2022-11-15|
|2022-11-15 13:30:00|                 5.5|             825.9|            0|2022-11-15|
|2022-11-15 13:20:00|                 5.0|             825.8|            0|2022-11-15|
|2022-11-15 13:10:00|                 4.5|             825.7|            0|2022-11-15|
|2022-11-15 13:00:00|                 4.0|             825.7|            0|2022-11-15|
|2022-11-15 12:50:00|                 4.1|             825.6|            0|2022-11-15|
|2022-11-15 12:40:00|                 3.5|             825.5|            0|2022-11-15|
|2022-11-15 12:30:00|                 3.4|             825.3|            0|2022-11-15|
|2022-11-15 12:20:00|                 3.3|             825.2|            0|2022-11-15|
|2022-11-15 12:10:00|                 3.5|               825|            0|2022-11-15|
|2022-11-15 12:00:00|                 3.7|             824.9|            0|2022-11-15|
+-------------------+--------------------+------------------+-------------+----------+
only showing top 20 rows
```

Para generar columnas extra en el DF se esta utilizando la funcion .withColumn para generar la columna ‘FECHA’ sacando el date de ‘FechaUTC’

```
#Ahora podemos ver cuantos registros se hicieron diarios.
regdia=temp.groupBy("FECHA").agg(count("Fecha UTC")).sort(desc("FECHA"))
regdia.show(20) #Mostramos los primeros 20
regdia.count()#DIAS REGISTRADOS
```

```
+----------+----------------+
|     FECHA|count(Fecha UTC)|
+----------+----------------+
|2022-11-15|              92|
|2022-11-14|             144|
|2022-11-13|             144|
|2022-11-12|             144|
|2022-11-11|             144|
|2022-11-10|             144|
|2022-11-09|             144|
|2022-11-08|             144|
|2022-11-07|             144|
|2022-11-06|             144|
|2022-11-05|             144|
|2022-11-04|             144|
|2022-11-03|             144|
|2022-11-02|             144|
|2022-11-01|             144|
|2022-10-31|             144|
|2022-10-30|             138|
|2022-10-29|             144|
|2022-10-28|             144|
|2022-10-27|             144|
+----------+----------------+
only showing top 20 rows


91
```

Con esta nueva columna nueva generada podemos agrupar por fechas iguales y podemos observar que en general se realizaban 144 muestreos diarios y se realizaron muestreos por 91 dias diferentes.

```
#Teniendo los temporales diarios podemos agrupar para ver maximos y minimos o promedios por dia.
#Tambien hacer busquedas por dia en especial.
diario=temp.groupBy("FECHA")
```

```
#El promedio diario de la temperatura del aire y maximo
diario.mean().sort(desc("avg(Temperatura del Aire)")).show()
diario.max().sort(desc("max(Temperatura del Aire)")).show()
```

Usando la misma analogia de agrupar las fechas iguales podemos ver el promedio de temperaturas diarios de todos los muestreos, ademas de la maxima diaria.

FECHA	avg(Temperatura del Aire)
2022-08-17	28.15370379553901
2022-08-21	23.90347222487132
2022-08-22	23.341666724946762
2022-08-18	23.270138806766933
2022-08-20	22.964583317438763
2022-08-19	22.684892064375845
2022-08-28	22.229861074023777
2022-08-29	22.114583333333332
2022-08-23	21.984057979307313
2022-08-24	21.83958339691162
2022-08-25	21.727083351877
2022-10-24	21.52013895246718
2022-11-04	21.332638919353485
2022-08-30	21.1513888835907
2022-08-31	20.918055560853745
2022-10-16	20.85972218381034
2022-09-01	20.624305579397415
2022-09-09	20.540972265932297
2022-11-03	20.45694437291887
2022-09-10	20.45555555820465

only showing top 20 rows

FECHA	max(Temperatura del Aire)
2022-08-21	32.2
2022-08-18	31.8
2022-08-20	31.5
2022-08-22	31.2
2022-08-19	31.1
2022-08-29	30.7
2022-08-17	30.5
2022-11-06	30.4
2022-08-24	29.9
2022-08-27	29.7
2022-11-10	29.7
2022-11-03	29.7
2022-08-28	29.7
2022-11-04	29.6
2022-10-22	29.4
2022-10-16	29.3
2022-08-30	29.3
2022-09-01	29.2
2022-09-09	29.1
2022-10-15	28.8

only showing top 20 rows

Para poder generar los mismos analisis con las demas columnas del DF se tienen que cambiar de formato

```
#Cambiamos los type de las demas columnas para poder hacer pruebas con ellas tambien
temp = temp.withColumn("Presion Atmosferica",temp["Presion Atmosferica"].cast("float"))
temp = temp.withColumn("Precipitacion",temp["Precipitacion"].cast("float"))
```

```
#Comprobamos que se cambio correctamente el type
temp.dtypes
```

```
[('Fecha UTC', 'string'),
 ('Temperatura del Aire', 'float'),
 ('Presion Atmosferica', 'float'),
 ('Precipitacion', 'float'),
 ('FECHA', 'date')]
```

```
#Generamos el groupby dario con los nuevos datos.
diario=temp.groupBy("FECHA")
```

```
#Mostramos el dataframe con cada maximo por dia
diario.max().sort(desc("FECHA")).show(91)
```

```
+----------+-------------------------+-------------------------+-------------------+
|     FECHA|max(Temperatura del Aire)|max(Presion Atmosferica)|max(Precipitacion)|
+----------+-------------------------+-------------------------+-------------------+
|2022-11-15|                     17.3|                    826.7|                0.0|
|2022-11-14|                     27.4|                    822.0|                0.0|
|2022-11-13|                     21.9|                    825.2|                0.0|
|2022-11-12|                     18.5|                    826.2|                0.0|
|2022-11-11|                     27.3|                    820.7|                0.0|
|2022-11-10|                     29.7|                    824.3|                0.0|
|2022-11-09|                     27.9|                    827.4|                0.0|
|2022-11-08|                     26.0|                    828.0|                0.0|
|2022-11-07|                     27.1|                    826.6|                0.0|
|2022-11-06|                     30.4|                    823.5|                0.0|
|2022-11-05|                     26.6|                    822.9|                0.0|
|2022-11-04|                     29.6|                    823.6|                0.0|
|2022-11-03|                     29.7|                    826.8|                0.0|
|2022-11-02|                     27.7|                    827.9|                0.0|
|2022-11-01|                     23.2|                    826.1|                0.0|
|2022-10-31|                     22.7|                    822.0|                0.0|
|2022-10-30|                     24.7|                    824.9|                0.0|
|2022-10-29|                     24.3|                    827.4|                0.0|
|2022-10-28|                     24.9|                    825.4|                0.0|
|2022-10-27|                     27.7|                    822.5|                0.0|
```

Una vez comprobado y aprendido el uso de funciones en un DF se puede aplicar a varios DF al mismo tiempo por medio de iteraciones.

Conclusion

Spark es una herramienta muy util en uso de dataframes, la diferencia con pandas es que Spark permite el uso decomandos SQL aparte de interacciones diferente con los DF y RDDs

Referencias

Procesando Datos con Spark (II) — Carga y limpieza de datos | by Cristian Cardellino | Medium

Tutorial de PySpark para principiantes: Ejemplo de aprendizaje automático - Guru99

pyspark.sql.DataFrame.withColumnRenamed — PySpark 3.1.3 documentation (apache.org)

csv - Spark Option: inferSchema vs header = true - Stack Overflow