ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Curso Académico 2022/2023

Trabajo Fin de Grado

INGENIERÍA DE DATOS CON EL FRAMEWORK DE BIG DATA SPARK Y SCALA

Autor: Álvaro Sánchez Pérez

Directores: Juan Manuel Serrano Hidalgo

Índice

1. Introducción

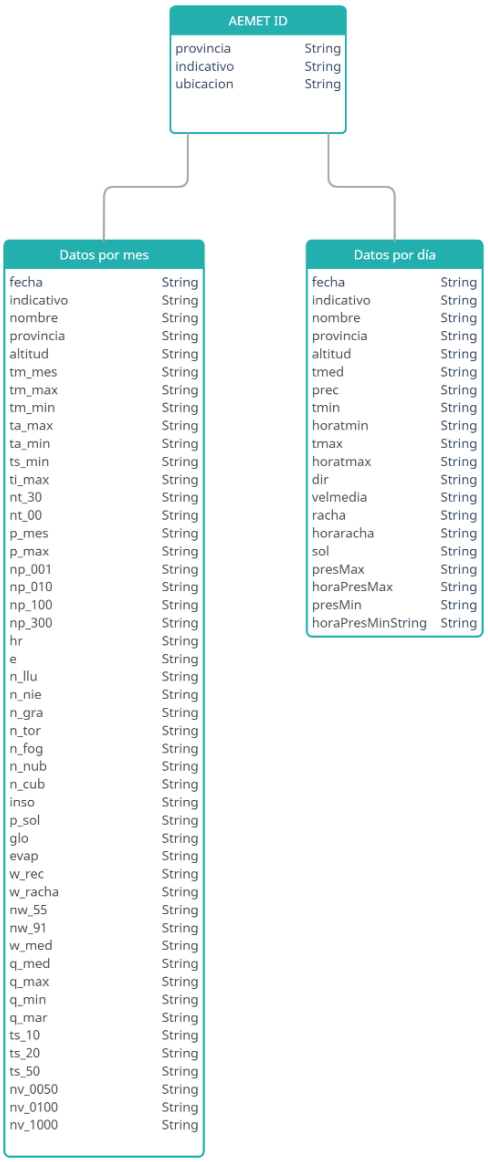
Prueba

2. Objetivos

3. Descripción informática

3.1. Fuentes de datos

Los datos meteorológicos, han sido obtenidos a través de la página de *AEMET OpenData*, se tratan tanto de datos que muestran la información de forma mensual como de forma diaria desde el año 2010 hasta el año actual 2022. La estructura en la que vienen estos datos se puede observar en la siguiente imagen, todos los valores vienen en formato String, el cual transformaremos mediante el casteo correspondiente en Spark para así poder trabajar posteriormente con estos datos de una manera más cómoda.



Al descargar los datos, se encuentran en un formato JSON multilínea, el cual no llega a ser lo suficientemente optimo para la lectura con Apache Spark. Por lo tanto, se realizaron cambios en todos los ficheros transformándolos a ficheros JSON que contuvieran toda la información en una única línea, esto se llevó a cabo mediante el siguiente comando utilizando la consola de Windows: **FOR %a IN (../data/*.json) DO jq . -c "%a" > "../JSONLine/%a”**. También cabe destacar que los datos una vez leídos en este formato JSON, la mayoría fueron transformados al formato Parquet ya que resulta más óptimo para la realización de consultas con Spark.

3.1.1. Obtención de los datos

Para la obtención de los datos, se crearon unos pequeños programas en el lenguaje Java. Estos programas, se basan en la simulación de los pasos que deberíamos de seguir para la descarga. Para ello se hizo uso de la librería *Selenium*, a través de la cual se puede manejar un navegador web pudiendo realizar diversas acciones sobre la página mostrada.

Por lo tanto, de manera resumida, los pasos que realizarían ambos programas para la descarga serían los siguientes; abriríamos un nuevo navegador donde accederemos al sitio web de *AEMET OpenData*, el primer paso dentro de página sería introducir la clave API correspondiente a nuestro usuario. Posteriormente, dependiendo del tipo de información que quisiéramos obtener, ya fuera información por días o por meses, buscaríamos los desplegables y elementos necesarios mediante su *id* o *xpath*.

Una vez seleccionada la información deseada, se nos abriría una nueva página donde nos proporciona diferentes tipos de información, además del enlace de la página donde se encontrarán los datos deseados. Abriremos esta nueva página, obtendremos la información y la guardaremos en un nuevo fichero con el nombre correspondiente a la consulta. Por último, cerramos todas las pestañas y navegadores que se hubieran abierto, y procederíamos a realizar los mismos pasos con otro rango de fechas o en otra estación meteorológica, el objetivo es obtener la información de todas las estaciones existentes entre el rango de fechas establecido.

3.2. Programación de queries

3.3. Visualización de queries

Año de la temperatura máxima en cada mes

En esta consulta, realizaremos una comparación del mismo mes en diferentes años, para saber en qué año se registró la temperatura máxima en este mes. Por último, lo representaremos tanto mediante una tabla, como de manera gráfica para poder observar las diferencias de manera visual entre los distintos años.

Para realizar esta consulta utilizaremos los datos meteorológicos mensuales, ya que nos ofrecen la temperatura máxima registrada en cada mes de una manera bastante sencilla. Por lo tanto, el primer paso que deberíamos de realizar sería la lectura de estos datos.

A continuación, deberíamos de agrupar estos datos por fecha, ya que, al realizar la lectura, tenemos la información de muchas estaciones meteorológicas diferentes en el mismo mes y año, por lo tanto, los agruparemos haciendo que coincida la fecha, y realizando una media de la temperatura máxima en todas las estaciones. Por último, utilizaremos la función ventana de Spark, para dividir los datos por meses y ordenándolos de manera descendente a través de nuestro objetivo, la temperatura máxima, ya que así estaríamos consiguiendo obtener el año, en el cual este mes fue el más caluroso.

```
import org.apache.spark.sql.expressions.Window

val data = spark.read.parquet("D:/TFGAlvaroSanchez/data/monthParquet/*").na.drop()

val window = Window.partitionBy("mes").orderBy($"ta_max".desc)

val dataWindow = data
    .withColumn("ta_max", func.split($"ta_max", "\\(")(0).cast(DoubleType))
    .groupBy($"fecha")
    .agg(func.avg($"ta_max").alias("ta_max"))
    .select(func.month($"fecha").alias("mes"), func.year($"fecha").alias("año"), $"ta_max")
    .withColumn("dense_rank", func.dense_rank().over(window))
    .filter($"dense_rank" === 1)
```

Como se puede observar, tal y como se describió anteriormente, el primer paso que realizamos es la lectura de los datos, los cuales los estamos leyendo en un formato Parquet para que resulte más óptimo y además estamos eliminando las filas que contienen valores nulos mediante la función *na.drop()*, la lectura de estos datos la almacenamos en la variable *data*. Al realizar la lectura de estos, se encontrarían de la siguiente manera, donde únicamente se estarían mostrando las dos primeras filas.

```
+----------+----------+-------+-------+----+-----+------+--------+------+-----+-----+-----------+------+-----+------+-------+--
---+-----+------+-----+-----+-----+------+------+------+------+
|     fecha|indicativo|  p_max|    glo|  hr|nw_55|tm_min|  ta_max|ts_min|nt_30|n_des|    w_racha|np_100|nw_91|np_001| ta_min|w_
rec|    e|np_300|p_mes|w_med|nt_00|ti_max|tm_mes|tm_max|np_010|
+----------+----------+-------+-------+----+-----+------+--------+------+-----+-----+-----------+------+-----+------+-------+--
---+-----+------+-----+-----+-----+------+------+------+------+
|2014-01-01|      B954|7.5(16)|26493.0|73.0|   12|   9.2|21.4(03)|  13.2|    0|    1|28/25.8(04)|     0|    1|     6|3.8(31)|
349|115.0|     0| 26.3| 15.0|    0|  14.6|  13.3|  17.4|     4|
|2014-02-01|      B954|4.1(09)|33644.0|72.0|   12|   8.3|20.4(13)|  13.0|    0|    4|26/21.7(10)|     0|    0|     9|3.0(03)|
457|110.0|     0| 10.8| 19.0|    0|  14.4|  13.0|  17.7|     3|
```

```
val window = Window.partitionBy("mes").orderBy($"ta_max".desc)
```

Por otro lado, también nos creamos la función ventana en la variable *window* que utilizaremos posteriormente. Le indicaremos la manera de la cual deseamos que se dividan los datos, en nuestro caso será a través de los meses y a su vez le señalamos la manera en la que se deberán de ordenar, teniendo en cuenta la temperatura máxima.

```
val dataWindow = data
    .withColumn("ta_max", func.split($"ta_max", "\\(")(0).cast(DoubleType))
```

A continuación, procedemos a realizar la consulta que almacenaremos en la variable *dataWindow*. El primer paso, que realizamos con los datos es la modificación de la columna de temperatura máxima, ya que esta nos viene con el siguiente formato temperaturaMáximaAlcanzada(díaDelMesEnElQueSeAlcanzó) por lo que Spark estaría tratando esta columna como un String, cuando nuestro objetivo es manejarlo como un tipo Double. Para esto realizamos lo siguiente, separamos mediante la función *Split* la temperatura máxima del día en que se alcanzó, y nos quedamos únicamente con la temperatura máxima, por último, realizamos un casteo al tipo de datos que deseamos.

```
.groupBy($"fecha")
.agg(func.avg($"ta_max").alias("ta_max"))
```

El segundo paso, de la consulta seria agrupar los datos de las diferentes estaciones por fecha, para ello utilizamos la función *groupBy* indicándole la columna a través de la cual deseamos realizar la agrupación. Mediante la función *agg* y *avg*, obtenemos y calculamos la media de la temperatura máxima de las filas que hemos agrupado. Actualmente tendríamos el DataFrame que se muestra a continuación, que contendría únicamente las fechas y sus respectivas temperaturas.

```
+----------+------------------+
|     fecha|            ta_max|
+----------+------------------+
|2013-01-01| 20.84285714285715|
|2010-06-01| 32.37096774193548|
|2021-03-01|  25.4304347826087|
|2016-10-01|28.869696969696975|
|2019-01-01| 18.59714285714286|
```

```
.select(func.month($"fecha").alias("mes"), func.year($"fecha").alias("año"), $"ta_max")
```

En el siguiente paso, obtendremos tanto el mes y el año, a partir de la fecha. Para esto utilizamos las funciones *month* y *year* con las cuales obtenemos tanto el mes y el año respectivamente. Mediante la función *select* seleccionaremos únicamente las columnas que vamos a necesitar.

```
.withColumn("dense_rank", func.dense_rank().over(window))
```

Para ir finalizando, añadimos una columna con el valor que nos devuelve la función *dense_rank*, que hace uso de la ventana *window* anteriormente definida. Mediante esta función, obtenemos la posición en la que se encuentra un valor teniendo en cuenta la expresión *orderBy* establecida en la función ventana (*window*). Por lo tanto, en nuestro nos devolverá una clasificación de los años más calurosos dividido por meses, ya que era la partición que se estableció en la función *window*. Quedando la consulta de la siguiente manera después de aplicar la función vemtana.

```
+---+----+------------------+----------+
|mes| año|            ta_max|dense_rank|
+---+----+------------------+----------+
|  7|2022| 39.46785714285714|         1|
|  7|2015|             37.08|         2|
|  7|2016| 37.00499999999999|         3|
|  7|2020| 36.78799999999999|         4|
|  7|2017| 36.67575757575758|         5|
|  7|2013|35.952777777777776|         6|
|  7|2021| 35.90769230769231|         7|
|  7|2019| 35.85454545454545|         8|
|  7|2012|  35.5258064516129|         9|
|  7|2014|35.071875000000006|        10|
|  7|2010|          34.84375|        11|
|  7|2018|34.155882352941184|        12|
|  7|2011| 33.51379310344828|        13|
| 11|2015|25.378787878787882|         1|
| 11|2020|25.307407407407414|         2|
```

```
.filter($"dense_rank" === 1)
```

Por último, realizamos un filtrado quedándonos únicamente con las filas que poseen en la columna *dense_rank* un valor igual a 1, obteniendo así el año en el que se obtuvo la mayor temperatura máxima en un determinado mes.

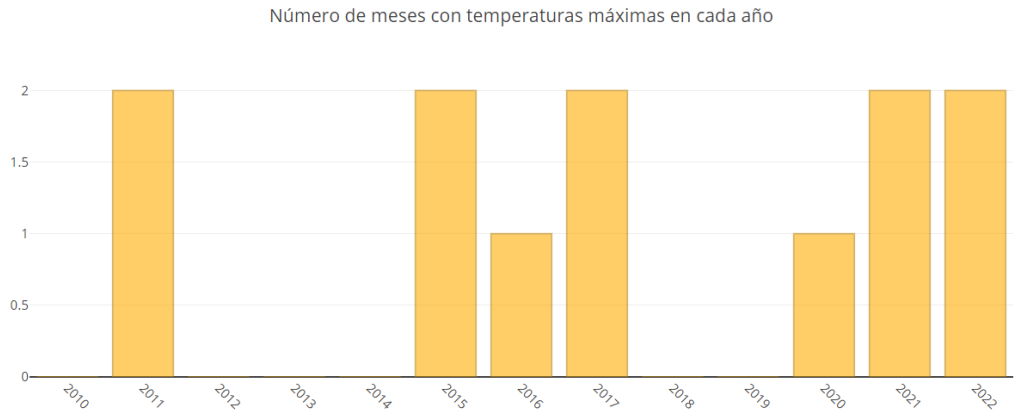
```
def countByYear(year : Int) : Long = {
    dataWindow
        .filter($"año" === year)
        .count()
}

val year2010 = countByYear(2010)
```

Se implementa también la función *countByYear*, la cual nos facilitará los datos que utilizaremos para la representación gráfica, ya que nos devuelve la cantidad de meses por cada año que se encuentran en la consulta realizada anteriormente.

Para finalizar ofrecemos los resultados tanto en forma de tabla como de una forma gráfica que sería para la cual habríamos hecho uso de la función de la *countByYear*.

mes	año	temperatura maxima
1	2021	21.84
2	2020	24.42
3	2017	27.82
4	2011	29.48
5	2015	33.73
6	2017	36.86
7	2022	39.47
8	2022	37.74
9	2016	36.39
10	2011	31.52
11	2015	25.38
12	2021	22.29



3.4. Despliegue en AWS EMR

4. Experimentos / validación

4.1. Consultas realizadas

4.2. Análisis de requisitos no funcionales

5. Conclusiones

6. Bibliografía

7. Apéndices