



CENTRO UNIVERSITARIO
DE TECNOLOGÍA Y ARTE DIGITAL

PROGRAMA EXPERTO BIG DATA 2016-2017

TRABAJO FIN DE EXPERTO

BUSINESS INTELLIGENCE SOBRE BIG DATA



AUTOR:
TUTOR:

PEDRO TOBARRA GUILLAMÓN (Estudiante Programa Experto Big Data U-TAD)
JULIO CONCA PASTOR (Solutions Specific Knowledge Analyst at Everis)



an NTT DATA Company

TABLA DE CONTENIDOS

1.	DESCRIPCIÓN	3
2.	OBJETIVOS	3
3.	TECNOLOGIAS UTILIZADAS	3
4.	ORIGEN DE LOS DATOS	3
5.	PLAN BASICO A SEGUIR	3
6.	EXPLORACION Y DEFINICION DE LOS GRÁFICOS A REALIZAR	4
7.	CARGADO Y PROCESADO EN SPARK DEL DATASET ELEGIDO.....	7
8.	CARGA DEL RESULTADO EN HIVE E IMPALA	19
9.	IMPLEMENTACIÓN DE LOS GRÁFICOS CON TABLEAU	24

1. DESCRIPCIÓN

Los proyectos Big Data están englobados en dos grandes conjuntos, operacionales y de investigación. Los proyectos operacionales serían tipos de proyectos que ya se están realizando con las herramientas tradicionales, pero utilizando las tecnologías Big Data para realizarlos sobre más información, más rápido y/o de forma más económica. Son los proyectos que están realizando las empresas que quieren aproximarse al mundo Big Data. Business Intelligence, la habilidad para transformar los datos en información, y la información en conocimiento, de forma que se pueda optimizar el proceso de toma de decisiones en los negocios, sería uno de esos proyectos de mejora operacional. En este trabajo partiríamos de datos meteorológicos para transformarlos en conocimiento y poder consumirlos de una forma visual.

2. OBJETIVOS

- Introducirse en uno de los casos de uso más comunes de introducción al Big Data para las empresas.
- Crear cuadros de mando con un enfoque más clásico (Herramientas de BI y HiveQL)

3. TECNOLOGIAS UTILIZADAS

- Apache Spark versión 2.2.0 con Python versión 2.7.3 para el formateado de datos y la obtención del Dataframe que será utilizado en Hive como Datawarehouse.
- Jupyter Notebook para la introducción de comandos en PySpark
- HDFS para el almacenamiento de las tablas de Datawarehouse y del Datamart.
- Hive para la creación de las tablas del Datawarehouse y del Datamart y Cloudera Impala para la realización de consultas sobre el Datamart.
- Hue de Cloudera Hadoop como interfaz visual con HDFS, Hive e Impala.
- Tableau Professional para la presentación visual de los datos.

4. ORIGEN DE LOS DATOS

Los datos son obtenidos de estaciones meteorológicas repartidas en todo el mundo y cuyas mediciones son recogidas por la 'National Oceanic and Atmospheric Administration NOAA' de los Estados Unidos; que, además, son públicos y pueden descargarse desde el siguiente enlace:

ftp://ftp.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/


5. PLAN BASICO A SEGUIR














- Exploración y Definición de los gráficos a realizar.
- Carga y Procesado en Spark del dataset elegido.
- Carga del resultado en Hive e Impala.
- Implementación de los gráficos con Tableau.

6. EXPLORACION Y DEFINICION DE LOS GRÁFICOS A REALIZAR

En la dirección ftp://ftp.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/ se pueden encontrar datos diarios de más de 50 magnitudes meteorológicas de más de 200 países de todo el mundo. Estos datos han sido agrupados anualmente desde el año 1763 hasta la actualidad como se puede observar en la siguiente figura:

Índice de /pub/data/ghcn/daily/by_year/



 [directorio principal]

	Nombre	Tamaño	Fecha de modificación
	1763.csv.gz	3.3 kB	9/9/17 0:12:00
	1764.csv.gz	3.2 kB	9/9/17 0:11:00
	1765.csv.gz	3.3 kB	9/9/17 0:12:00
	1766.csv.gz	3.3 kB	9/9/17 0:11:00
	1767.csv.gz	3.3 kB	9/9/17 0:11:00
...			
	2013.csv.gz	188 MB	9/9/17 0:12:00
	2014.csv.gz	184 MB	9/9/17 0:12:00
	2015.csv.gz	187 MB	9/9/17 0:12:00
	2016.csv.gz	186 MB	9/9/17 0:12:00
	2017.csv.gz	118 MB	9/9/17 0:12:00
	by-year-status.txt	535 B	26/10/15 1:00:00
	ghcn-daily-by_year-format.rtf	31.8 kB	15/8/11 2:00:00
	readme.txt	2.8 kB	16/8/11 2:00:00

A modo de ejemplo se muestra un extracto del fichero de texto '2016.csv' que resulta de descomprimir el fichero '2016.csv.gz':

2016.csv		
		0 10 20 30 T
1	US1NJGL0001,20160101,PRCP,0,,N,	
2	US1NJGL0001,20160101,SNOW,0,,N,	
3	CA1AB000023,20160101,PRCP,0,,N,	
4	CA1AB000023,20160101,SNOW,0,,N,	
5	CA1AB000023,20160101,SNWD,102,,N,	

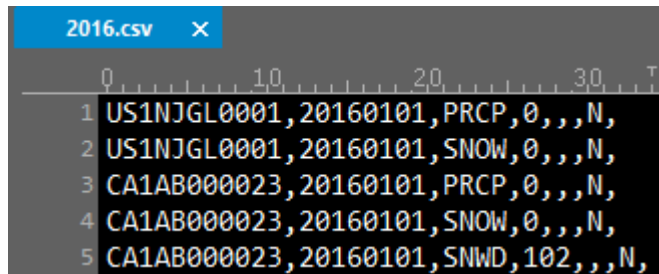
...	
33645172	US1WAIS0021,20161231,PRCP,0,T,,N,
33645173	US1TXHYS103,20161231,PRCP,0,,N,
33645174	US1TXHYS103,20161231,SNOW,0,,N,
33645175	US1TXHYS136,20161231,PRCP,0,,N,
33645176	US1TXHYS136,20161231,SNOW,0,,N,
33645177	US1TXMNG020,20161231,PRCP,13,,N,

Nombre	Fecha de modifica...	Tipo	Tamaño
 2016.csv	07/07/2017 0:04	Archivo CSV	1.151.917 KB
 2016.csv.gz	07/07/2017 0:04	Archivo WinRAR	186.466 KB

Como se puede observar en las figuras anteriores, el fichero de texto 2016.csv tiene 33.6 millones de líneas y ocupa 1.1 GB.

Siendo el año 2016 el año completo para el que más cantidad de datos había disponibles; se decidió escoger el fichero '2016.csv' como fuente de datos para realizar este proyecto.

El formato de los datos del fichero '2016.csv' es el siguiente:



```

0 10 20 30 T
1 US1NJGL0001,20160101,PRCP,0,,,N,
2 US1NJGL0001,20160101,SNOW,0,,,N,
3 CA1AB000023,20160101,PRCP,0,,,N,
4 CA1AB000023,20160101,SNOW,0,,,N,
5 CA1AB000023,20160101,SNWD,102,,,N,

```

'US1NJGL001' es el nombre de la estación meteorológica que recogió los datos. Concretamente las dos primeras letras del nombre de la estación corresponden al país en el que se encuentra la misma. Por ejemplo 'US' es para los Estados Unidos y 'CA' para Canada.

'20160101' es la fecha en la que se recogieron los datos en formato AñoMesDía.

'PRCP', 'SNOW', 'SNWD' son los nombres del tipo de magnitud correspondiente al dato. Se recogen más de 60 tipos de magnitudes meteorológicas distintas; pero las 5 magnitudes 'core' más importantes del dataset son:

The five core elements are:

PRCP = Precipitation (tenths of mm)

SNOW = Snowfall (mm)

SNWD = Snow depth (mm)

TMAX = Maximum temperature (tenths of degrees C)

TMIN = Minimum temperature (tenths of degrees C)

Los valores '0', '0', '0', '102' son el valor de la magnitud medida y el resto de parámetros a la derecha del valor de la magnitud no serán objeto de este proyecto y serán, por tanto, eliminados en el procesamiento inicial de los datos.

Tras la exploración de la información disponible se decidió que la información a procesar y formatear sería la correspondiente estas 5 magnitudes core; eliminándose todas aquellas líneas que contuvieran datos de otras magnitudes.

Es decir que de un fichero csv de entrada:

```
USC00363028,20160101,SNWD,0,,,7,0,0
ASN00026012,20160101,PRCP,0,,,a,
ASN00055322,20160101,PRCP,2,,,a,
USC00358466,20160101,TMAX,11,,,7,0,0
USC00358466,20160101,TMIN,-33,,,7,0,0
USC00358466,20160101,TOBS,-22,,,7,0,0
USC00358466,20160101,PRCP,0,,,7,0,0
USC00358466,20160101,SNOW,0,,,7,
USC00358466,20160101,SNWD,0,,,7,
USC00355392,20160101,TMAX,-17,,,7,0,0
USC00355392,20160101,TMIN,-39,,,7,0,0
USC00355392,20160101,TOBS,-33,,,7,0,0
USC00355392,20160101,PRCP,0,,,7,0,0
USC00350265,20160101,TMAX,0,,,7,0,0
USC00350265,20160101,TMIN,-56,,,7,0,0
USC00350265,20160101,TOBS,-44,,,7,0,0
USC00350265,20160101,PRCP,0,,,7,0,0
ASN00004036,20160101,PRCP,0,,,a,
SWE00137802,20160101,TMAX,11,,,E,
```

Se obtendría el siguiente fichero csv de salida:

```
COUNTRY,DATE,PRCP,SNOW,SNWD,TMAX,TMIN
US,20160101,-9999,-9999,0,-9999,-999
AS,20160101,0,-9999,-9999,-9999,-9999
AS,20160101,2,-9999,-9999,-9999,-9999
US,20160101,-9999,-9999,-9999,11,-9999
US,20160101,-9999,-9999,-9999,-9999,-33
US,20160101,0,-9999,-9999,-9999,-9999
US,20160101,-9999,0,-9999,-9999,-9999
US,20160101,-9999,-9999,0,-9999,-9999
US,20160101,-9999,-9999,-9999,-17,-9999
US,20160101,-9999,-9999,-9999,-9999,-39
US,20160101,0,-9999,-9999,-9999,-9999
US,20160101,-9999,-9999,-9999,0,-9999
US,20160101,-9999,-9999,-9999,-9999,-56
US,20160101,0,-9999,-9999,-9999,-9999
AS,20160101,0,-9999,-9999,-9999,-9999
SW,20160101,-9999,-9999,-9999,11,-9999
```

Que solo contiene datos correspondientes a las magnitudes 'core' (las líneas de la magnitud TOBS se han eliminado) y donde se ha elegido el valor '-9999' para distinguir aquellas magnitudes de las que no se tiene medida de las magnitudes de las que se tiene medida con valor '0'.

La misma transformación se muestra a través de las siguientes tablas:

DATOS ENTRADA				DATOS SALIDA						
ID	DATE	ELEMENT	VALUE1	COUNTRY	DATE	PRCP	SNOW	SNWD	TMAX	TMIN
USC00363028	20160101	SNWD	0	US	20160101	-9.999	-9.999	0	-9.999	-9.999
ASN00026012	20160101	PRCP	0	AS	20160101	0	-9.999	-9.999	-9.999	-9.999
ASN00055322	20160101	PRCP	2	AS	20160101	2	-9.999	-9.999	-9.999	-9.999
USC00358466	20160101	TMAX	11	US	20160101	-9.999	-9.999	-9.999	11	-9.999
USC00358466	20160101	TMIN	-33	US	20160101	-9.999	-9.999	-9.999	-9.999	-33
USC00358466	20160101	TOBS	-22							
USC00358466	20160101	PRCP	0	US	20160101	0	-9.999	-9.999	-9.999	-9.999
USC00358466	20160101	SNOW	0	US	20160101	-9.999	0	-9.999	-9.999	-9.999
USC00358466	20160101	SNWD	0	US	20160101	-9.999	-9.999	0	-9.999	-9.999
USC00355392	20160101	TMAX	-17	US	20160101	-9.999	-9.999	-9.999	-17	-9.999
USC00355392	20160101	TMIN	-39	US	20160101	-9.999	-9.999	-9.999	-9.999	-39
USC00355392	20160101	TOBS	-33							
USC00355392	20160101	PRCP	0	US	20160101	0	-9.999	-9.999	-9.999	-9.999
USC00350265	20160101	TMAX	0	US	20160101	-9.999	-9.999	-9.999	0	-9.999
USC00350265	20160101	TMIN	-56	US	20160101	-9.999	-9.999	-9.999	-9.999	-56
USC00350265	20160101	TOBS	-44							
USC00350265	20160101	PRCP	0	US	20160101	0	-9.999	-9.999	-9.999	-9.999
ASN00004036	20160101	PRCP	0	AS	20160101	0	-9.999	-9.999	-9.999	-9.999
SWE00137802	20160101	TMAX	11	SW	20160101	-9.999	-9.999	-9.999	11	-9.999

Como se mostrará más adelante en este documento, las columnas de la tabla 'DATOS SALIDA' forman el dataframe que se obtendrá tras procesar los datos con Spark y Python; y que será el que se usará como tabla para contener el 'datawarehouse' de datos del proyecto.

A partir de este datawarehouse, según lo acordado con el tutor, se pintará una gráfica de las temperaturas máximas diarias a lo largo de 2016 filtradas por países. Para ello, a partir del datawarehouse se creará un datamart específico para este propósito con las siguientes columnas : 'COUNTRY,DATE,TMAX'.

7. CARGADO Y PROCESADO EN SPARK DEL DATASET ELEGIDO

A continuación se irán exponiendo y explicando las instrucciones y transformaciones utilizadas para transformar el fichero '2016.csv' en el dataframe comentando al final del apartado anterior.

Se ha utilizado Jupyter Notebook con Apache Spark versión 2.2.0 y Python versión 2.7.3.

Los parámetros de la sesión son:

```
In [2]: spark

Out[2]: SparkSession - hive
SparkContext

Spark UI
Version
v2.2.0
Master
local[*]
AppName
PySparkShell
```


Cargamos el fichero 2016.csv como un dataframe separado en columnas:

```
In [1]: year2016csvReadAsDF=spark.read.csv("/home/utad/spark-2.2.0-bin-hadoop2.7/con01/2016.csv")

In [3]: year2016csvReadAsDF.take(10)

Out[3]: [Row(_c0='US1NCCR0043', _c1='20160101', _c2='PRCP', _c3='79', _c4=None, _c5=None, _c6='N', _c7=None),
Row(_c0='CA1AB000023', _c1='20160101', _c2='PRCP', _c3='0', _c4=None, _c5=None, _c6='N', _c7=None),
Row(_c0='CA1AB000023', _c1='20160101', _c2='SNOW', _c3='0', _c4=None, _c5=None, _c6='N', _c7=None),
Row(_c0='CA1AB000023', _c1='20160101', _c2='SNWD', _c3='102', _c4=None, _c5=None, _c6='N', _c7=None),
Row(_c0='US1NCBC0113', _c1='20160101', _c2='PRCP', _c3='5', _c4=None, _c5=None, _c6='N', _c7=None),
Row(_c0='ASN00015643', _c1='20160101', _c2='TMAX', _c3='308', _c4=None, _c5=None, _c6='a', _c7=None),
Row(_c0='ASN00015643', _c1='20160101', _c2='TMIN', _c3='186', _c4=None, _c5=None, _c6='a', _c7=None),
Row(_c0='ASN00015643', _c1='20160101', _c2='PRCP', _c3='0', _c4=None, _c5=None, _c6='a', _c7=None),
Row(_c0='US1MTMH0019', _c1='20160101', _c2='PRCP', _c3='0', _c4=None, _c5=None, _c6='N', _c7=None),
Row(_c0='US1MTMH0019', _c1='20160101', _c2='SNOW', _c3='0', _c4=None, _c5=None, _c6='N', _c7=None)]
```

Renombramos las columnas del dataframe:

```
In [7]: year2016csvReadAsDF

Out[7]: DataFrame[_c0: string, _c1: string, _c2: string, _c3: string, _c4: string, _c5: string, _c6: string, _c7: string]

In [9]: year2016csvReadAsDF=year2016csvReadAsDF.withColumnRenamed('_c1','DATE')

In [10]: year2016csvReadAsDF=year2016csvReadAsDF.withColumnRenamed('_c2','ELEMENT')

In [11]: year2016csvReadAsDF=year2016csvReadAsDF.withColumnRenamed('_c3','DATA_VALUE')

In [12]: year2016csvReadAsDF=year2016csvReadAsDF.withColumnRenamed('_c4','M-FLAG')

In [13]: year2016csvReadAsDF=year2016csvReadAsDF.withColumnRenamed('_c5','Q-FLAG')

In [14]: year2016csvReadAsDF=year2016csvReadAsDF.withColumnRenamed('_c6','S-FLAG')

In [15]: year2016csvReadAsDF=year2016csvReadAsDF.withColumnRenamed('_c7','OBS-TIME')

In [16]: year2016csvReadAsDF.show()
```

ID	DATE	ELEMENT	DATA_VALUE	M-FLAG	Q-FLAG	S-FLAG	OBS-TIME
US1NCCR0043	20160101	PRCP	79	null	null	N	null
CA1AB000023	20160101	PRCP	0	null	null	N	null
CA1AB000023	20160101	SNOW	0	null	null	N	null
CA1AB000023	20160101	SNWD	102	null	null	N	null
US1NCBC0113	20160101	PRCP	5	null	null	N	null
ASN00015643	20160101	TMAX	308	null	null	a	null
ASN00015643	20160101	TMIN	186	null	null	a	null
ASN00015643	20160101	PRCP	0	null	null	a	null
US1MTMH0019	20160101	PRCP	0	null	null	N	null
US1MTMH0019	20160101	SNOW	0	null	null	N	null
US1MTMH0019	20160101	SNWD	165	null	null	N	null
US1MTMH0019	20160101	WESD	269	null	null	N	null
ASN00085296	20160101	TMAX	338	null	null	a	null
ASN00085296	20160101	TMIN	242	null	null	a	null
ASN00085296	20160101	PRCP	0	null	null	a	null

Creo 5 dataframes de manera que cada uno tenga datos de una de las cinco magnitudes core:

```
In [17]: year2016csv_PRCP_DF = year2016csvReadAsDF.filter(year2016csvReadAsDF.ELEMENT.contains("PRCP"))
```

```
In [18]: year2016csv_PRCP_DF.show()
```

ID	DATE	ELEMENT	DATA_VALUE	M-FLAG	Q-FLAG	S-FLAG	OBS-TIME
US1NCCR0043	20160101	PRCP	79	null	null	N	null
CA1AB000023	20160101	PRCP	0	null	null	N	null
US1NCBC0113	20160101	PRCP	5	null	null	N	null
ASN00015643	20160101	PRCP	0	null	null	a	null
US1MTMH0019	20160101	PRCP	0	null	null	N	null

```
In [19]: year2016csv_SNOW_DF = year2016csvReadAsDF.filter(year2016csvReadAsDF.ELEMENT.contains("SNOW"))
```

```
In [20]: year2016csv_SNOW_DF.show()
```

ID	DATE	ELEMENT	DATA_VALUE	M-FLAG	Q-FLAG	S-FLAG	OBS-TIME
CA1AB000023	20160101	SNOW	0	null	null	N	null
US1MTMH0019	20160101	SNOW	0	null	null	N	null
US1MNCV0008	20160101	SNOW	0	T	null	N	null
US1MISW0005	20160101	SNOW	5	null	null	N	null
CA007020860	20160101	SNOW	20	null	null	C	null

```
In [21]: year2016csv_SNWD_DF = year2016csvReadAsDF.filter(year2016csvReadAsDF.ELEMENT.contains("SNWD"))
```

```
In [22]: year2016csv_SNWD_DF.show()
```

ID	DATE	ELEMENT	DATA_VALUE	M-FLAG	Q-FLAG	S-FLAG	OBS-TIME
CA1AB000023	20160101	SNWD	102	null	null	N	null
US1MTMH0019	20160101	SNWD	165	null	null	N	null
US1MNCV0008	20160101	SNWD	127	null	null	N	null
US1MISW0005	20160101	SNWD	13	null	null	N	null
CA007020860	20160101	SNWD	250	null	null	C	null

```
In [23]: year2016csv_TMAX_DF = year2016csvReadAsDF.filter(year2016csvReadAsDF.ELEMENT.contains("TMAX"))
```

```
In [24]: year2016csv_TMAX_DF.show()
```

ID	DATE	ELEMENT	DATA_VALUE	M-FLAG	Q-FLAG	S-FLAG	OBS-TIME
ASN00015643	20160101	TMAX	308	null	null	a	null
ASN00085296	20160101	TMAX	338	null	null	a	null
ASN00040209	20160101	TMAX	285	null	null	a	null
ASN00085280	20160101	TMAX	347	null	null	a	null
CA007020860	20160101	TMAX	-10	null	null	C	null

```
In [25]: year2016csv_TMIN_DF = year2016csvReadAsDF.filter(year2016csvReadAsDF.ELEMENT.contains("TMIN"))
```

```
In [26]: year2016csv_TMIN_DF.show()
```

ID	DATE	ELEMENT	DATA_VALUE	M-FLAG	Q-FLAG	S-FLAG	OBS-TIME
ASN00015643	20160101	TMIN	186	null	null	a	null
ASN00085296	20160101	TMIN	242	null	null	a	null
ASN00040209	20160101	TMIN	218	null	null	a	null
ASN00085280	20160101	TMIN	159	null	null	a	null
CA007020860	20160101	TMIN	-50	null	null	C	null

A continuación, uno los 5 dataframes uno a continuación del otro de manera que obtengo un dataframe solo con datos de las 5 magnitudes core una a continuación de la otra. Nótese que hemos pasado de un dataframe de casi 34 millones de líneas a uno de 26 millones:

```
In [27]: year2016csv_filtered_DF = year2016csv_PRCP_DF.union(year2016csv_SNOW_DF)
```

```
In [28]: year2016csv_filtered_DF = year2016csv_filtered_DF.union(year2016csv_SNOW_DF)
```

```
In [29]: year2016csv_filtered_DF = year2016csv_filtered_DF.union(year2016csv_TMAX_DF)
```

```
In [30]: year2016csv_filtered_DF = year2016csv_filtered_DF.union(year2016csv_TMIN_DF)
```

```
In [31]: year2016csv_filtered_DF.count()
```

```
Out[31]: 26185832
```

```
In [32]: year2016csv_filtered_DF.show()
```

ID	DATE	ELEMENT	DATA_VALUE	M-FLAG	Q-FLAG	S-FLAG	OBS-TIME
US1NCCR0043	20160101	PRCP	79	null	null	N	null
CA1AB000023	20160101	PRCP	0	null	null	N	null
US1NCBC0113	20160101	PRCP	5	null	null	N	null
ASN00015643	20160101	PRCP	0	null	null	a	null
US1MTMH0019	20160101	PRCP	0	null	null	N	null

```
In [33]: year2016csv_filtered_DF.printSchema()
```

```
root
|-- ID: string (nullable = true)
|-- DATE: string (nullable = true)
|-- ELEMENT: string (nullable = true)
|-- DATA_VALUE: string (nullable = true)
|-- M-FLAG: string (nullable = true)
|-- Q-FLAG: string (nullable = true)
|-- S-FLAG: string (nullable = true)
|-- OBS-TIME: string (nullable = true)
```

Paso el dataframe a formato RDD:

```
In [34]: year2016csv_filtered_RDD=year2016csv_filtered_DF.rdd
```

```
In [35]: year2016csv_filtered_RDD
```

```
Out[35]: MapPartitionsRDD[77] at javaToPython at NativeMethodAccessorImpl.java:0
```

```
In [36]: year2016csv_filtered_RDD.take(20)
```

```
Out[36]: [Row(ID=u'US1NCCR0043', DATE=u'20160101', ELEMENT=u'PRCP', DATA_VALUE=u'79', M-FLAG=None, Q-FLAG=None, S-FLAG=u'N', OBS-TIME=None),
Row(ID=u'CA1AB000023', DATE=u'20160101', ELEMENT=u'PRCP', DATA_VALUE=u'0', M-FLAG=None, Q-FLAG=None, S-FLAG=u'N', OBS-TIME=None),
Row(ID=u'US1NCBC0113', DATE=u'20160101', ELEMENT=u'PRCP', DATA_VALUE=u'5', M-FLAG=None, Q-FLAG=None, S-FLAG=u'N', OBS-TIME=None),
Row(ID=u'ASN00015643', DATE=u'20160101', ELEMENT=u'PRCP', DATA_VALUE=u'0', M-FLAG=None, Q-FLAG=None, S-FLAG=u'a', OBS-TIME=None),
Row(ID=u'US1MTMH0019', DATE=u'20160101', ELEMENT=u'PRCP', DATA_VALUE=u'0', M-FLAG=None, Q-FLAG=None, S-FLAG=u'N', OBS-TIME=None),
```

A continuación, me quedo con los cuatro primeros elementos de cada Row que son con los que voy a trabajar;

```
In [37]: pruebaRDD = year2016csv_filtered_RDD.map(lambda s: s[:4])
```

```
In [38]: pruebaRDD
```

```
Out[38]: PythonRDD[79] at RDD at PythonRDD.scala:48
```

```
In [39]: pruebaRDD.take(20)
```

```
Out[39]: [(u'US1NCCR0043', u'20160101', u'PRCP', u'79'),
          (u'CA1AB000023', u'20160101', u'PRCP', u'0'),
          (u'US1NCBC0113', u'20160101', u'PRCP', u'5'),
          (u'ASN00015643', u'20160101', u'PRCP', u'0'),
```

Vuelvo a pasar el RDD a formato dataframe nombrando las columnas:

```
In [40]: pruebaDF = pruebaRDD.toDF(['ID', 'DATE', 'ELEMENT', 'DATA_VALUE'])
```

```
In [41]: pruebaDF
```

```
Out[41]: DataFrame[ID: string, DATE: string, ELEMENT: string, DATA_VALUE: string]
```

```
In [42]: pruebaDF.show()
```

```
+-----+-----+-----+-----+
|          ID|      DATE|ELEMENT|DATA_VALUE|
+-----+-----+-----+-----+
|US1NCCR0043|20160101|  PRCP|         79|
|CA1AB000023|20160101|  PRCP|          0|
|US1NCBC0113|20160101|  PRCP|          5|
|ASN00015643|20160101|  PRCP|          0|
|US1MTMH0019|20160101|  PRCP|          0|
```

```
In [43]: pruebaDF.printSchema()
```

```
root
 |-- ID: string (nullable = true)
 |-- DATE: string (nullable = true)
 |-- ELEMENT: string (nullable = true)
 |-- DATA_VALUE: string (nullable = true)
```

Ahora vuelvo a pasar el dataframe obtenido a formato RDD:

```
In [126]: pruebaRDD = pruebaDF.rdd
```

```
In [127]: pruebaRDD.take(10)
```

```
Out[127]: [Row(ID=u'RSM00028224', DATE=u'20160430', ELEMENT=u'PRCP', DATA_VALUE=u'0'),
          Row(ID=u'USC00242793', DATE=u'20160430', ELEMENT=u'PRCP', DATA_VALUE=u'0'),
          Row(ID=u'USC00231924', DATE=u'20160430', ELEMENT=u'PRCP', DATA_VALUE=u'0'),
          Row(ID=u'USC00225070', DATE=u'20160430', ELEMENT=u'PRCP', DATA_VALUE=u'0'),
          Row(ID=u'RSM00020087', DATE=u'20160430', ELEMENT=u'PRCP', DATA_VALUE=u'0'),
```

Ahora me quedo sólo con los 2 primeros caracteres de la primera columna pues necesito el nombre del país; NO el de la estación meteorológica:

```
In [157]: from pyspark.sql import Row
```

```
In [158]: IDSubStringedRDD = pruebaRDD.map(lambda p: Row(p[0][:2],p[1],p[2],p[3]))
```

```
In [159]: IDSubStringedRDD
```

```
Out[159]: PythonRDD[199] at RDD at PythonRDD.scala:48
```

```
In [160]: IDSubStringedRDD.take(10)
```

```
Out[160]: [<Row(RS, 20160430, PRCP, 0)>,
<Row(US, 20160430, PRCP, 0)>,
<Row(US, 20160430, PRCP, 0)>,
<Row(US, 20160430, PRCP, 0)>,
<Row(RS, 20160430, PRCP, 0)>,
<Row(AS, 20160430, PRCP, 0)>,
<Row(US, 20160430, PRCP, 0)>,
<Row(US, 20160430, PRCP, 0)>,
<Row(US, 20160430, PRCP, 10)>,
<Row(US, 20160430, PRCP, 0)>]
```

Paso el RDD obtenido a dataframe nombrando las columnas:

```
In [161]: from pyspark.sql import *
```

```
In [163]: IDSubStringedDF = IDSubStringedRDD.toDF(['COUNTRY','DATE','ELEMENT','DATA_VALUE'])
```

```
In [164]: IDSubStringedDF.show()
```

COUNTRY	DATE	ELEMENT	DATA_VALUE
RS	20160430	PRCP	0
US	20160430	PRCP	0
US	20160430	PRCP	0
US	20160430	PRCP	0
RS	20160430	PRCP	0

Añado una columna con el nombre de cada magnitud core rellena con el valor '-9999' para todas las filas de la tabla:

```
IDSubStringed_PRCP_DF = IDSubStringedDF.withColumn('PRCP',IDSubStringedDF.DATA_VALUE -
IDSubStringedDF.DATA_VALUE - 9999)
```

```
In [173]: IDSubStringed_PRCP_DF = IDSubStringedDF.withColumn('PRCP',IDSubStringedDF.DATA_VALUE - IDSubStringedDF.DATA_VALU
```

```
In [174]: IDSubStringed_PRCP_DF.show()
```

COUNTRY	DATE	ELEMENT	DATA_VALUE	PRCP
RS	20160430	PRCP	0	-9999.0
US	20160430	PRCP	0	-9999.0
US	20160430	PRCP	0	-9999.0
US	20160430	PRCP	0	-9999.0
RS	20160430	PRCP	0	-9999.0

IDsSubStringed_addedColumns_DF =

IDsSubStringed_PRCP_DF.withColumn('SNOW',IDsSubStringed_PRCP_DF.DATA_VALUE -

IDsSubStringed_PRCP_DF.DATA_VALUE - 9999)

```
In [175]: IDsSubStringed_addedColumns_DF = IDsSubStringed_PRCP_DF.withColumn('SNOW',IDsSubStringed_PRCP_DF.DATA_VALUE - IDsSub
```

```
In [176]: IDsSubStringed_addedColumns_DF.show()
```

COUNTRY	DATE	ELEMENT	DATA_VALUE	PRCP	SNOW
RS	20160430	PRCP	0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0
RS	20160430	PRCP	0	-9999.0	-9999.0

IDsSubStringed_addedColumns_DF =

IDsSubStringed_addedColumns_DF.withColumn('SNWD',IDsSubStringed_addedColumns_DF.DATA_VALUE -

IDsSubStringed_addedColumns_DF.DATA_VALUE - 9999)

```
In [178]: IDsSubStringed_addedColumns_DF = IDsSubStringed_addedColumns_DF.withColumn('SNWD',IDsSubStringed_addedColumns_DF.DAT
```

```
In [179]: IDsSubStringed_addedColumns_DF.show()
```

COUNTRY	DATE	ELEMENT	DATA_VALUE	PRCP	SNOW	SNWD
RS	20160430	PRCP	0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0
RS	20160430	PRCP	0	-9999.0	-9999.0	-9999.0

IDsSubStringed_addedColumns_DF =

IDsSubStringed_addedColumns_DF.withColumn('TMAX',IDsSubStringed_addedColumns_DF.DATA_VALUE -

IDsSubStringed_addedColumns_DF.DATA_VALUE - 9999)

```
In [180]: IDsSubStringed_addedColumns_DF = IDsSubStringed_addedColumns_DF.withColumn('TMAX',IDsSubStringed_addedColumns_DF.DAT
```

```
In [181]: IDsSubStringed_addedColumns_DF.show()
```

COUNTRY	DATE	ELEMENT	DATA_VALUE	PRCP	SNOW	SNWD	TMAX
RS	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0
RS	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0

IDsSubStringed_addedColumns_DF =

IDsSubStringed_addedColumns_DF.withColumn('TMIN',IDsSubStringed_addedColumns_DF.DATA_VALUE -

IDsSubStringed_addedColumns_DF.DATA_VALUE - 9999)

```
In [182]: IDsSubStringed_addedColumns_DF = IDsSubStringed_addedColumns_DF.withColumn('TMIN',IDsSubStringed_addedColumns_DF.DAT
```

```
In [183]: IDsSubStringed_addedColumns_DF.show()
```

COUNTRY	DATE	ELEMENT	DATA_VALUE	PRCP	SNOW	SNWD	TMAX	TMIN
RS	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
RS	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0

Creo un dataframe cogiendo sólo las líneas 'PRCP':

```
In [201]: year2016csvPRCP_updated_DF=IDSubStringed_addedColumns_DF.where(IDSubStringed_addedColumns_DF.ELEMENT=="PRCP")
```

```
In [202]: year2016csvPRCP_updated_DF.show()
```

COUNTRY	DATE	ELEMENT	DATA_VALUE	PRCP	SNOW	SNWD	TMAX	TMIN
RS	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
RS	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0

Elimino la columna 'PRCP':

```
In [216]: year2016csvPRCP_updated_DF=year2016csvPRCP_updated_DF.drop('PRCP')
```

```
In [218]: year2016csvPRCP_updated_DF.show()
```

COUNTRY	DATE	ELEMENT	DATA_VALUE	SNOW	SNWD	TMAX	TMIN
RS	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0
RS	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0

Cambio el nombre de la columna 'DATA_VALUE' por el de 'PRCP':

```
In [220]: year2016csvPRCP_updated_DF=year2016csvPRCP_updated_DF.withColumnRenamed('DATA_VALUE','PRCP')
```

```
In [221]: year2016csvPRCP_updated_DF.show()
```

COUNTRY	DATE	ELEMENT	PRCP	SNOW	SNWD	TMAX	TMIN
RS	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0
RS	20160430	PRCP	0	-9999.0	-9999.0	-9999.0	-9999.0

Realizo esta misma operación para las otras cuatro magnitudes core:

```
In [222]: year2016csvSNOW_updated_DF=IDSubStringed_addedColumns_DF.where(IDSubStringed_addedColumns_DF.ELEMENT=="SNOW")
```

```
In [223]: year2016csvSNOW_updated_DF.show()
```

COUNTRY	DATE	ELEMENT	DATA_VALUE	PRCP	SNOW	SNWD	TMAX	TMIN
CA	20160101	SNOW	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160101	SNOW	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160101	SNOW	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
US	20160101	SNOW	5	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
CA	20160101	SNOW	20	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0

```
In [224]: year2016csvSNOW_updated_DF=year2016csvSNOW_updated_DF.drop('SNOW')
```

```
In [225]: year2016csvSNOW_updated_DF.show()
```

```
+-----+-----+-----+-----+-----+-----+-----+
|COUNTRY|  DATE|ELEMENT|DATA_VALUE|  PRCP|  SNWD|  TMAX|  TMIN|
+-----+-----+-----+-----+-----+-----+-----+
|      CA|20160101|  SNOW|      0|-9999.0|-9999.0|-9999.0|-9999.0|
|      US|20160101|  SNOW|      0|-9999.0|-9999.0|-9999.0|-9999.0|
|      US|20160101|  SNOW|      0|-9999.0|-9999.0|-9999.0|-9999.0|
|      US|20160101|  SNOW|      5|-9999.0|-9999.0|-9999.0|-9999.0|
|      CA|20160101|  SNOW|     20|-9999.0|-9999.0|-9999.0|-9999.0|
```

```
In [226]: year2016csvSNOW_updated_DF=year2016csvSNOW_updated_DF.withColumnRenamed('DATA_VALUE','SNOW')
```

```
In [228]: year2016csvSNOW_updated_DF.take(20)
```

```
Out[228]: [Row(COUNTRY=u'CA', DATE=u'20160101', ELEMENT=u'SNOW', SNOW=u'0', PRCP=-9999.0, SNWD=-9999.0, TMAX=-9999.0, TMIN=-9999.0),
Row(COUNTRY=u'US', DATE=u'20160101', ELEMENT=u'SNOW', SNOW=u'0', PRCP=-9999.0, SNWD=-9999.0, TMAX=-9999.0, TMIN=-9999.0),
Row(COUNTRY=u'US', DATE=u'20160101', ELEMENT=u'SNOW', SNOW=u'0', PRCP=-9999.0, SNWD=-9999.0, TMAX=-9999.0, TMIN=-9999.0),
Row(COUNTRY=u'US', DATE=u'20160101', ELEMENT=u'SNOW', SNOW=u'5', PRCP=-9999.0, SNWD=-9999.0, TMAX=-9999.0, TMIN=-9999.0),
Row(COUNTRY=u'CA', DATE=u'20160101', ELEMENT=u'SNOW', SNOW=u'20', PRCP=-9999.0, SNWD=-9999.0, TMAX=-9999.0, TMIN=-9999.0),]
```

```
In [261]: year2016csvSNWD_updated_DF=IDsSubStringed_addedColumns_DF.where(IDsSubStringed_addedColumns_DF.ELEMENT=="SNWD")
```

```
In [264]: year2016csvSNWD_updated_DF=year2016csvSNWD_updated_DF.drop('SNWD')
```

```
In [266]: year2016csvSNWD_updated_DF=year2016csvSNWD_updated_DF.withColumnRenamed('DATA_VALUE','SNWD')
```

```
In [281]: year2016csvTMAX_updated_DF=IDsSubStringed_addedColumns_DF.where(IDsSubStringed_addedColumns_DF.ELEMENT=="TMAX")
```

```
In [283]: year2016csvTMAX_updated_DF=year2016csvTMAX_updated_DF.drop('TMAX')
```

```
In [285]: year2016csvTMAX_updated_DF=year2016csvTMAX_updated_DF.withColumnRenamed('DATA_VALUE','TMAX')
```

```
In [298]: year2016csvTMIN_updated_DF=IDsSubStringed_addedColumns_DF.where(IDsSubStringed_addedColumns_DF.ELEMENT=="TMIN")
```

```
In [299]: year2016csvTMIN_updated_DF=year2016csvTMIN_updated_DF.drop('TMIN')
```

```
In [301]: year2016csvTMIN_updated_DF=year2016csvTMIN_updated_DF.withColumnRenamed('DATA_VALUE','TMIN')
```

Unimos los 5 dataframes obtenidos uno detrás de otro:

```
In [250]: year2016csvUpdatedDF=year2016csvPRCP_updated_DF.union(year2016csvSNOW_updated_DF)
```

```
In [274]: year2016csvUpdatedDF=year2016csvUpdatedDF.union(year2016csvSNWD_updated_DF)
```

```
In [291]: year2016csvUpdatedDF=year2016csvUpdatedDF.union(year2016csvTMAX_updated_DF)
```

```
In [306]: year2016csvUpdatedDF=year2016csvUpdatedDF.union(year2016csvTMIN_updated_DF)
```

Y obtengo un dataframe con las siguientes características:

```
In [307]: year2016csvUpdatedDF.printSchema()
```

```
root
|-- COUNTRY: string (nullable = true)
|-- DATE: string (nullable = true)
|-- ELEMENT: string (nullable = true)
|-- PRCP: string (nullable = true)
|-- SNOW: string (nullable = true)
|-- SNWD: string (nullable = true)
|-- TMAX: string (nullable = true)
|-- TMIN: string (nullable = true)
```

```
In [320]: year2016csvTMIN_updated_DF.show()
```

COUNTRY	DATE	ELEMENT	PRCP	SNOW	SNWD	TMAX	TMIN
RS	20160430	TMIN	-9999.0	-9999.0	-9999.0	-9999.0	-20
US	20160430	TMIN	-9999.0	-9999.0	-9999.0	-9999.0	28
US	20160430	TMIN	-9999.0	-9999.0	-9999.0	-9999.0	72
US	20160430	TMIN	-9999.0	-9999.0	-9999.0	-9999.0	17
US	20160430	TMIN	-9999.0	-9999.0	-9999.0	-9999.0	-83

```
In [310]: year2016csvUpdatedDF.count()
```

```
Out[310]: 26185832
```

Elimino las líneas duplicadas:

```
In [325]: year2016csvNoDuplicatesDF=year2016csvUpdatedDF.dropDuplicates()
```

```
In [326]: year2016csvNoDuplicatesDF.count()
```

```
Out[326]: 2527723
```

```
In [327]: year2016csvNoDuplicatesDF.show()
```

COUNTRY	DATE	ELEMENT	PRCP	SNOW	SNWD	TMAX	TMIN
US	20160430	PRCP	18	-9999.0	-9999.0	-9999.0	-9999.0
AS	20160501	PRCP	300	-9999.0	-9999.0	-9999.0	-9999.0
US	20160501	PRCP	206	-9999.0	-9999.0	-9999.0	-9999.0
AS	20160501	PRCP	13	-9999.0	-9999.0	-9999.0	-9999.0
AS	20160501	PRCP	55	-9999.0	-9999.0	-9999.0	-9999.0

Cambio los tipos de las magnitudes core de 'string' a 'float':

```
In [15]: from pyspark.sql import *
```

```
In [25]: year2016csvTypesChangedDF=year2016csvNoDuplicatesDF.select('COUNTRY', 'DATE', 'ELEMENT', \
                                                                    year2016csvNoDuplicatesDF.PRCP.cast('float').alias('PRCP'), \
                                                                    year2016csvNoDuplicatesDF.SNOW.cast('float').alias('SNOW'), \
                                                                    year2016csvNoDuplicatesDF.SNWD.cast('float').alias('SNWD'), \
                                                                    year2016csvNoDuplicatesDF.TMAX.cast('float').alias('TMAX'), \
                                                                    year2016csvNoDuplicatesDF.TMIN.cast('float').alias('TMIN'))
```

```
In [26]: year2016csvTypesChangedDF.printSchema()
```

```
root
 |-- COUNTRY: string (nullable = true)
 |-- DATE: string (nullable = true)
 |-- ELEMENT: string (nullable = true)
 |-- PRCP: float (nullable = true)
 |-- SNOW: float (nullable = true)
 |-- SNWD: float (nullable = true)
 |-- TMAX: float (nullable = true)
 |-- TMIN: float (nullable = true)
```

```
In [34]: year2016csvTypesChangedDF.show()
```

COUNTRY	DATE	ELEMENT	PRCP	SNOW	SNWD	TMAX	TMIN
AS	20160430	PRCP	30.0	-9999.0	-9999.0	-9999.0	-9999.0
RS	20160501	PRCP	231.0	-9999.0	-9999.0	-9999.0	-9999.0
TU	20160501	PRCP	46.0	-9999.0	-9999.0	-9999.0	-9999.0
UV	20160501	PRCP	0.0	-9999.0	-9999.0	-9999.0	-9999.0
AS	20160501	PRCP	332.0	-9999.0	-9999.0	-9999.0	-9999.0

Cambio el tipo de la fecha de 'string' a 'integer':

```
In [43]: year2016csvTypesChangedDF=year2016csvTypesChangedDF.select('COUNTRY', \
                                                                    year2016csvTypesChangedDF.DATE.cast('integer').alias('DATE'), \
                                                                    'ELEMENT', 'PRCP', 'SNOW', 'SNWD', 'TMAX', 'TMIN')
```

```
In [44]: year2016csvTypesChangedDF.printSchema()
```

```
root
 |-- COUNTRY: string (nullable = true)
 |-- DATE: integer (nullable = true)
 |-- ELEMENT: string (nullable = true)
 |-- PRCP: float (nullable = true)
 |-- SNOW: float (nullable = true)
 |-- SNWD: float (nullable = true)
 |-- TMAX: float (nullable = true)
 |-- TMIN: float (nullable = true)
```

```
In [45]: year2016csvTypesChangedDF.show()
```

COUNTRY	DATE	ELEMENT	PRCP	SNOW	SNWD	TMAX	TMIN
AS	20160430	PRCP	30.0	-9999.0	-9999.0	-9999.0	-9999.0
RS	20160501	PRCP	231.0	-9999.0	-9999.0	-9999.0	-9999.0
TU	20160501	PRCP	46.0	-9999.0	-9999.0	-9999.0	-9999.0
UV	20160501	PRCP	0.0	-9999.0	-9999.0	-9999.0	-9999.0
AS	20160501	PRCP	332.0	-9999.0	-9999.0	-9999.0	-9999.0

Creo el dataframe final ya sin la columna 'ELEMENT':

```
In [47]: year2016csvFinalDF=year2016csvTypesChangedDF.select("COUNTRY","DATE","PRCP","SNOW","SNWD","TMAX","TMIN")
```

```
In [48]: year2016csvFinalDF.printSchema()
```

```
root
|-- COUNTRY: string (nullable = true)
|-- DATE: integer (nullable = true)
|-- PRCP: float (nullable = true)
|-- SNOW: float (nullable = true)
|-- SNWD: float (nullable = true)
|-- TMAX: float (nullable = true)
|-- TMIN: float (nullable = true)
```

```
In [49]: year2016csvFinalDF.show()
```

```
+-----+-----+-----+-----+-----+-----+
|COUNTRY|  DATE| PRCP|  SNOW|  SNWD|  TMAX|  TMIN|
+-----+-----+-----+-----+-----+-----+
|    AS |20160430| 30.0|-9999.0|-9999.0|-9999.0|-9999.0|
|    RS |20160501|231.0|-9999.0|-9999.0|-9999.0|-9999.0|
|    TU |20160501| 46.0|-9999.0|-9999.0|-9999.0|-9999.0|
|    UV |20160501|  0.0|-9999.0|-9999.0|-9999.0|-9999.0|
|    AS |20160501|332.0|-9999.0|-9999.0|-9999.0|-9999.0|
```

Finalmente guardo el dataframe como un fichero de texto 'csv' que utilizaré para crear el datamart en Hive en el siguiente capítulo:



```
In [5]: year2016csvFinalDF.count()
```







```
Out[5]: 2527723
```



```
In [7]: year2016csvFinalDF.write.csv('/home/utad/spark-2.2.0-bin-hadoop2.7/con01/year2016csvFinalDF.csv')
```

Nótese que antes del procesamiento de los datos teníamos un fichero csv de texto de 1.1 GB y 33 millones de líneas y al final de la transformación hemos obtenido uno de 122 MB y 2.5 millones de líneas.

```
33645175 US1TXHYS136,20161231,PRCP,0,,N,
33645176 US1TXHYS136,20161231,SNOW,0,,N,
33645177 US1TXMNG020,20161231,PRCP,13,,N,
```

Nombre	Fecha de modifica...	Tipo	Tamaño
 2016.csv	07/07/2017 0:04	Archivo CSV	1.151.917 KB
 2016.csv.gz	07/07/2017 0:04	Archivo WinRAR	186.466 KB

Nombre	Fecha de modifica...	Tipo	Tamaño
 _SUCCESS.crc	19/08/2017 13:22	Archivo CRC	1 KB
 .part-00000-70791137-4776-4fcd-8744-2f211918e679-c000.csv.crc	19/08/2017 13:22	Archivo CRC	616 KB
 .part-00001-70791137-4776-4fcd-8744-2f211918e679-c000.csv.crc	19/08/2017 13:22	Archivo CRC	343 KB
 _SUCCESS	19/08/2017 13:22	Archivo	0 KB
 part-00000-70791137-4776-4fcd-8744-2f211918e679-c000.csv	19/08/2017 13:22	Archivo CSV	78.846 KB
 part-00001-70791137-4776-4fcd-8744-2f211918e679-c000.csv	19/08/2017 13:22	Archivo CSV	43.880 KB

Nombre	Fecha de modifica...	Tipo	Tamaño
 year2016csvFinalDF.csv	19/08/2017 13:26	Carpeta de archivos	
 2016_TFM_final.csv	19/08/2017 13:48	Archivo CSV	122.726 KB

NOTA: en la figura anterior el fichero de texto 2016_TFM_final.csv es el resultado de unir los ficheros 'part-00000-70791137-4776-4fcd-8744-2f211918e679-c000.csv' y 'part-00001-70791137-4776-4fcd-8744-2f211918e679-c000.csv' con un editor de texto.

En un proyecto profesional se tendría que haber pasado el dataframe final desde Spark a Hive sin necesidad de descargarlo en un fichero texto. Debido a dificultades técnicas no se pudo ejecutar Spark versión 2.2.0 en la máquina virtual CentOS 6.7 de Cloudera. Por ello la transformación que se ha descrito se ha realizado en una máquina virtual 'ubuntu 14.04 LTS'. Al tener que cambiar de máquina virtual para continuar el proyecto se me permitió cargar el dataframe final en Hive desde un fichero en HDFS en lugar de leerlo directamente desde Spark.



8. CARGA DEL RESULTADO EN HIVE E IMPALA

En esta sección se describe la creación del datawarehouse en hive a partir del dataframe obtenido en el apartado anterior, la creación del datamart de temperaturas maximas diarias a partir del datawarehouse y la consulta a realizar en Impala sobre el datamart para alimentar en la siguiente sección a Tableau con los resultados de dicha consulta para pintar la gráfica de temperaturas máximas diarias mundiales filtradas por países.

Se crea y se utiliza en hive la base de datos 'con01' (CON01 es la denominación de U-TAD para el proyecto)

```
create database con01
use con01
```

Se crea la estructura vacía de la tabla datawarehouse:

```
create table datawarehouse ( COUNTRY string, DATE int, PRCP float, SNOW float, SNWD float, TMAX float, TMIN float)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE
```

Se realiza la carga del fichero csv obtenido en el apartado anterior desde el HDFS al 'datamart':

```
load data inpath '/user/cloudera/con01/2016_TFM_final.csv' into table datawarehouse
```

Con esto ya tendríamos creado el datawarehouse cuyas principales características se muestran a continuación:

HUE Home Query Editors ▾ Data Browsers ▾ Workflows ▾ Search Security ▾

Hive Browse Add a description...

con01

Tables (3) 🔍 ↺

datamart

datawarehouse

1 `SELECT * FROM `con01`.`datawarehouse``

Results ↗

	datawarehouse.country	datawarehouse.date	datawarehouse.prcp	datawarehouse.snow	datawarehouse.snwd	datawarehouse.tmax	datawarehouse.tmin
1	AS	20160430	30	-9999	-9999	-9999	-9999
2	RS	20160501	231	-9999	-9999	-9999	-9999
3	TU	20160501	46	-9999	-9999	-9999	-9999
4	UV	20160501	0	-9999	-9999	-9999	-9999

Overview Columns (7) Sample Details

PROPERTIES

Table

cloudera

Sun Aug 20 19:52:29 CEST 2017

text Not compressed

STATS ↺

Location

1 files

125670801 bytes

COLUMNS (7)

	Name	Type
1	country	string
2	date	int
3	prcp	float
4	snow	float
5	snwd	float

Databases > con01 > datawarehouse

Overview Columns (7) Sample **Details**

DETAILED TABLE INFORMATION

Database:	con01
Owner:	cloudera
CreateTime:	Sun Aug 20 19:52:29 CEST 2017
LastAccessTime:	UNKNOWN
Protect Mode:	None
Retention:	0
Location:	hdfs://quickstart.cloudera:8020/user/hive/warehouse/con01.db/datawarehouse
Table Type:	MANAGED_TABLE
Table Parameters:	
	COLUMN_STATS_ACCURATE true
	numFiles 1
	totalSize 125670801
	transient_lastDdlTime 1503252103

Una vez creado el datawarehouse en Hive, estamos en condiciones de crear la tabla datamart de temperaturas diarias máximas mundiales durante el año 2016 mediante la siguiente consulta:

The screenshot shows the Hue web interface for Hive. The top navigation bar includes 'HUE', a home icon, and menu items for 'Query Editors', 'Data Browsers', 'Workflows', 'Search', and 'Security'. Below this is a 'Hive' header with options to 'Add a name...' and 'Add a description...'. The left sidebar shows a tree view with 'con01' expanded, revealing a table named 'datamart' with columns: 'country (string)', 'date (int)', and 'tmax (double)'. The main panel displays a SQL query:

```

1 create table datamart as
2 select d.country country, d.date date, max(d.tmax)/10 tmax
3 from datawarehouse d
4 where d.tmax != -9999.0
5 group by d.country, d.date
6 order by d.country asc, d.date asc

```

Y los principales parámetros del datamart se muestran a continuación:

datamart table



Sample	Analysis			View more...
	datamart.country	datamart.date	datamart.tmax	
1	AE	20160101	31	
2	AE	20160102	30.1	
3	AE	20160103	23.5	
4	AE	20160104	22.4	
5	AE	20160105	22.8	
6	AE	20160106	23.5	
7	AE	20160107	23.5	
8	AE	20160108	25.2	
9	AE	20160109	27.2	
10	AE	20160110	27.7	

datamart table



Sample	Analysis	View more...
COLUMN_STATS_ACCURATE		true
numFiles		1
numRows		56963
rawDataSize		907207
totalSize		964170
transient_lastDdlTime		1503863583

A continuación, ya podemos realizar una consulta con Impala sobre el datamart para obtener los datos de temperaturas máximas mundiales diarias durante 2016 filtradas por países:

The screenshot shows the Hue Impala interface. At the top, there's a navigation bar with 'HUE' logo, a home icon, and dropdown menus for 'Query Editors', 'Data Browsers', 'Workflows', 'Search', and 'Security'. Below this is a header for 'Impala' with options to 'Add a name...' and 'Add a description...'. The main area is divided into two panes. The left pane, titled 'con01', shows a list of tables: 'datamart' and 'datawarehouse'. The right pane shows a SQL query: '1 select * from con01.datamart'. Below the query editor, there's a 'Results' tab. The results are displayed in a table with three columns: 'country', 'date', and 'tmax'. The table contains 10 rows of data, all for the country 'AE' (United Arab Emirates), showing dates from 20160101 to 20160110 and corresponding temperature values.

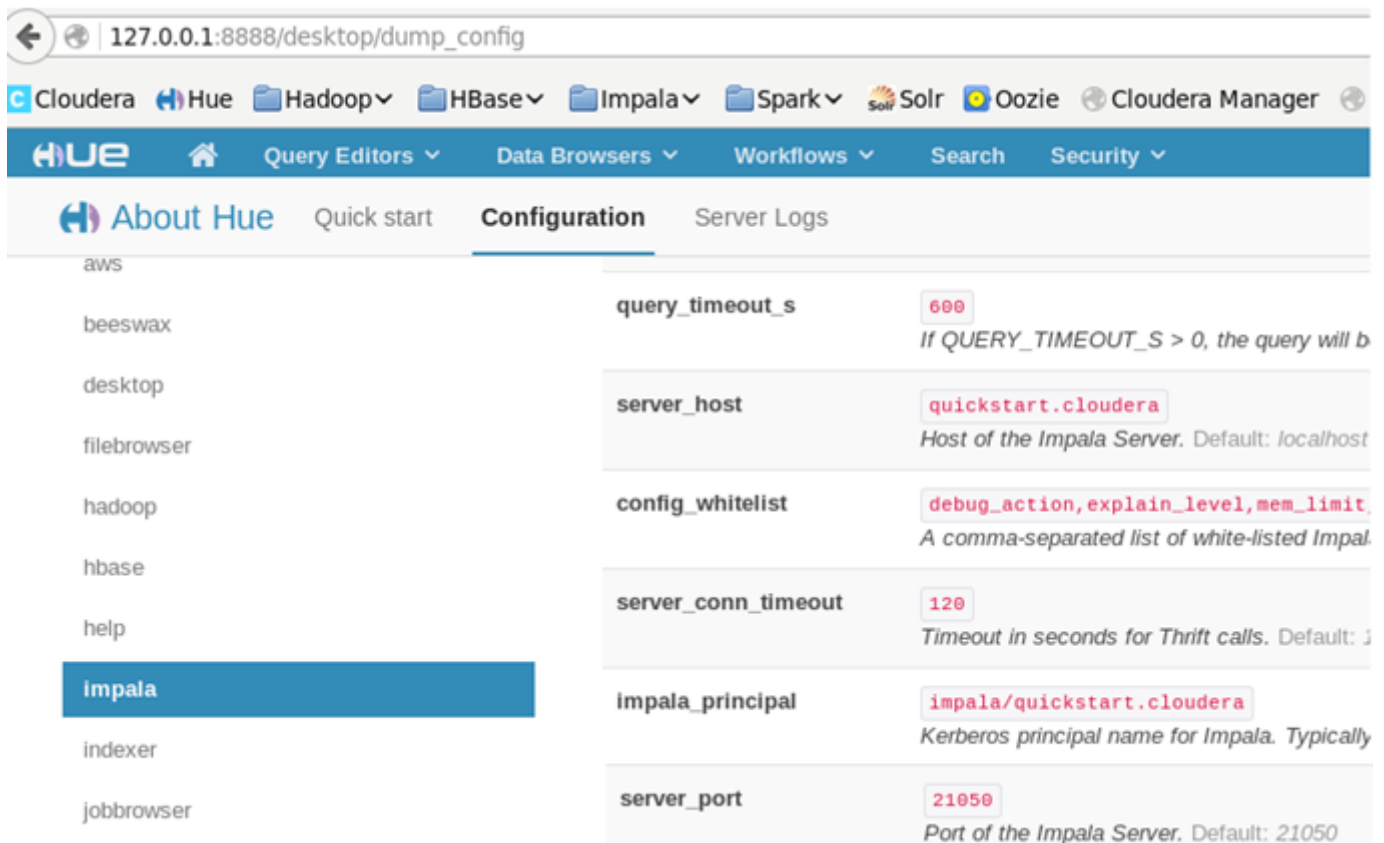
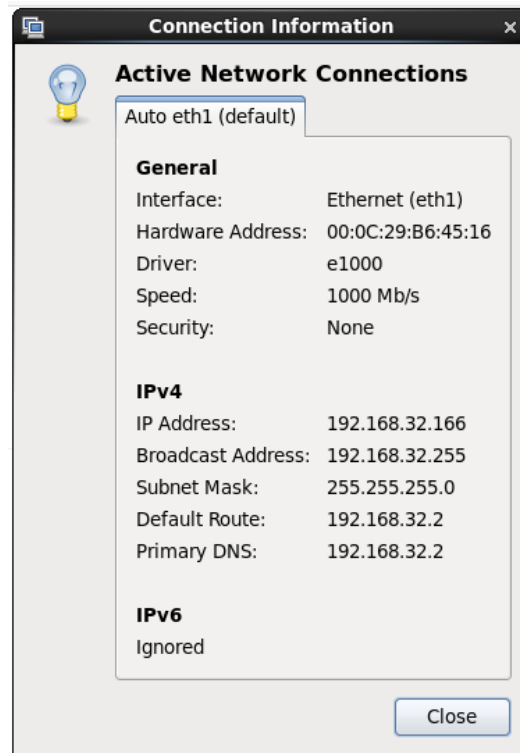
	country	date	tmax
1	AE	20160101	31
2	AE	20160102	30.100000000000001
3	AE	20160103	23.5
4	AE	20160104	22.399999999999999
5	AE	20160105	22.800000000000001
6	AE	20160106	23.5
7	AE	20160107	23.5
8	AE	20160108	25.199999999999999
9	AE	20160109	27.199999999999999
10	AE	20160110	27.699999999999999

Esta será la query que en la siguiente sección veremos que hay que realizar desde Tableau para acceder a los datos del datamart y poder pintar la gráfica de temperaturas máximas mundiales diarias durante 2016 filtradas por países.

9. IMPLEMENTACIÓN DE LOS GRÁFICOS CON TABLEAU

Para realizar esta etapa se ha trabajado con la herramienta 'Tableau Desktop Professional Edition' versión 10.3.2 ejecutándose sobre la máquina huésped con sistema operativo 'Windows 10 Home 64-bit' y Hive e Impala abiertos sobre la máquina virtual de Cloudera Linux CentOS versión 6.7

Para conectar las 2 máquinas y que Tableau pueda leer el datamart con Impala primero hay que averiguar la dirección IP de la máquina virtual y el puerto donde se está ejecutando Impala.



Con la información obtenida ya puede configurarse la conexión de Tableau con Impala y la query de descarga del datamart según se muestra en la siguiente figura:

The image shows the Tableau Desktop interface with a connection configuration window for Cloudera Hadoop and a SQL query window open.

Tableau Interface:

- Top bar: Tableau - Libro1 - La licencia de Tableau expira en 14 días
- Menu: Archivo, Datos, Servidor, Ventana, Ayuda
- Left sidebar: Conexiones (192.168.32.166 Cloudera Hadoop), Esquema (Seleccionar esquema)
- Right pane: 192.168.32.166

Cloudera Hadoop Connection Window:

- Server: 192.168.32.166, Port: 21050
- Write the information to log in to the server:
- Type: Impala
- Authentication: Nombre de usuario y contraseña
- Transport: Binario
- Username: cloudera
- Password: [Redacted]
- ☐ Requiere SSL
- Buttons: SQL inicial..., Iniciar sesión



SQL inicial Window:

- Instructions for executing SQL at connection time:
- Query: `select * from con01.datamart`
- Buttons: Insertar, Aceptar, Cancelar
- Link: Obtener más información

Y antes de haber intentado la conexión hay que descargar los drivers ODBC desde la página web de Cloudera e instalarlos en el sistema operativo huésped Windows según se muestra en las siguientes figuras:

<https://www.cloudera.com/downloads.html>



 ClouderaHiveODBC64.msi	03/09/2017 18:53	Paquete de Windo...	15.936 KB
 ClouderaImpalaODBC64.msi	03/09/2017 19:08	Paquete de Windo...	14.376 KB

NOTA: después de haber realizado las operaciones descritas en este capítulo, conseguí realizar la conexión entre las dos máquinas, pero no conseguí que se descargaran en tableau los resultados de la query. Tras intentar resolver el problema con mi tutor; decidimos que como los parámetros para conectar las dos máquinas estaban puestos correctamente, lo mejor era que continuara el proyecto descargando el resultado de la query en una hoja Excel desde Impala-HUE que será lo que mostraré a continuación. Pero en un proyecto profesional de Big Data se descargarían los datos directamente entre las dos máquinas.

Después de lo explicado, como ya no necesito hacer una conexión entre las dos máquinas, continuaré trabajando con 'Tableau Desktop Public Edition' versión 10.3.2.

En la figura de la página siguiente se puede ver el resultado de la descarga del resultado de la query en Tableau que, a su vez, fue descargada en un directorio local de la máquina virtual de cloudera desde HUE-Impala. Este directorio local era compartido por la máquina huésped Windows y por la máquina virtual Cloudera mediante la aplicación Dropbox, la cual se ejecutaba en las dos máquinas.

Nótese que después de todas las transformaciones realizadas en este proyecto, el datamart queda reducido a un fichero Excel de 3 columnas y 57.000 filas que ocupa 903 KB.



 > Pedro Tobarra > Dropbox > u-tad > 19-Tema 19 - Trabajo fin de Experto > con01			
Nombre	Fecha de modifica...	Tipo	Tamaño
 datamart.xlsx	31/08/2017 20:48	Hoja de cálculo d...	903 KB

Tableau Public - World Maximum Daily Temperatures in 2016

Archivo Datos Ventana Ayuda

Conexiones [Añadir](#)

datamart
Excel

Hojas

Sheet

Nueva unión

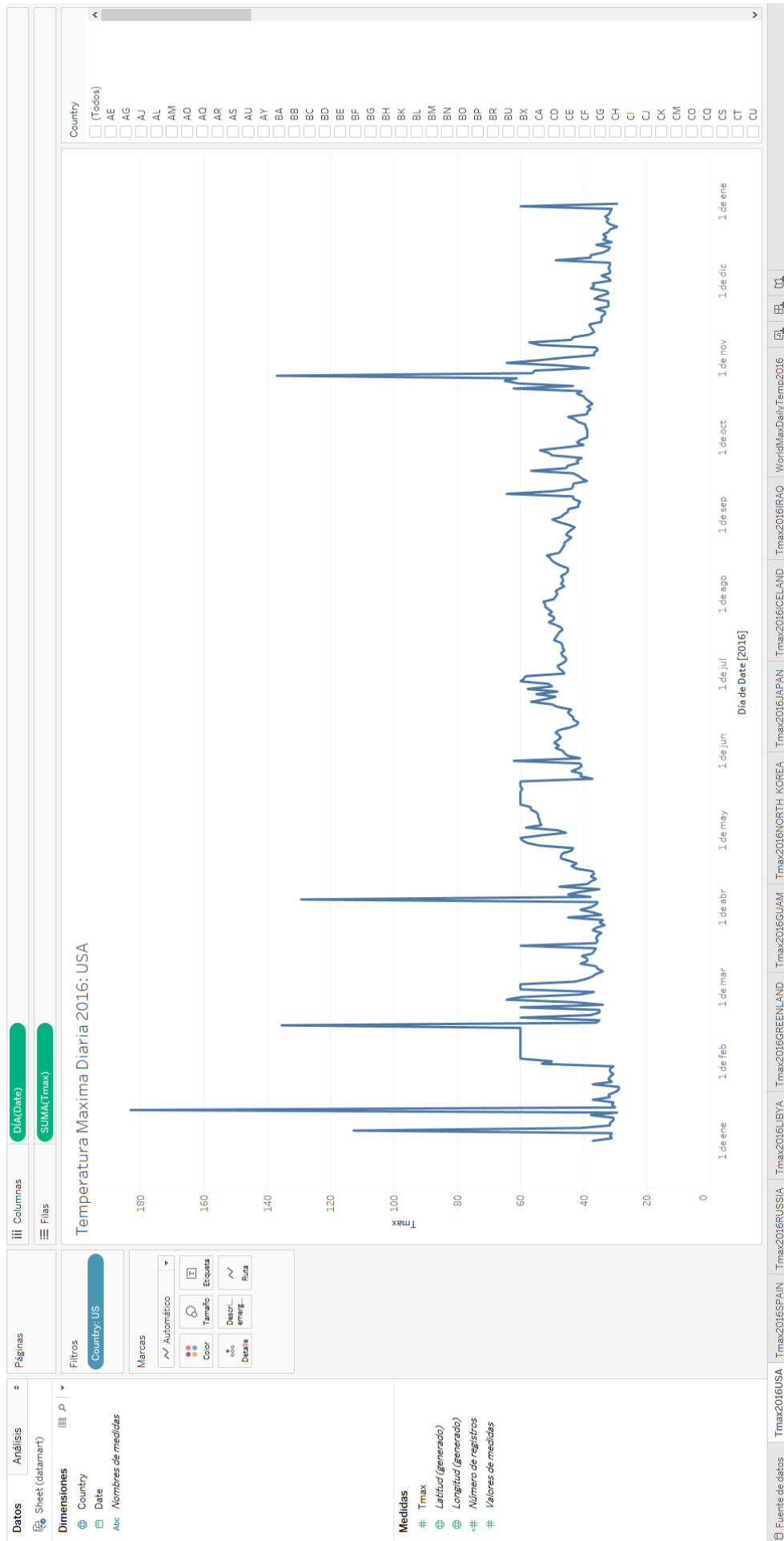
Sheet (datamart)

Sheet

Ordenar campos Orden de fuente de datos

Sheet Country	Sheet Date	Sheet Tmax
AE	01/01/2016	31,0000
AG	01/01/2016	24,0000
AJ	01/01/2016	3,5000
AE	02/01/2016	30,1000
AG	02/01/2016	24,2000
AJ	02/01/2016	4,2000
AE	03/01/2016	23,5000
AG	03/01/2016	25,2000
AJ	03/01/2016	4,7000

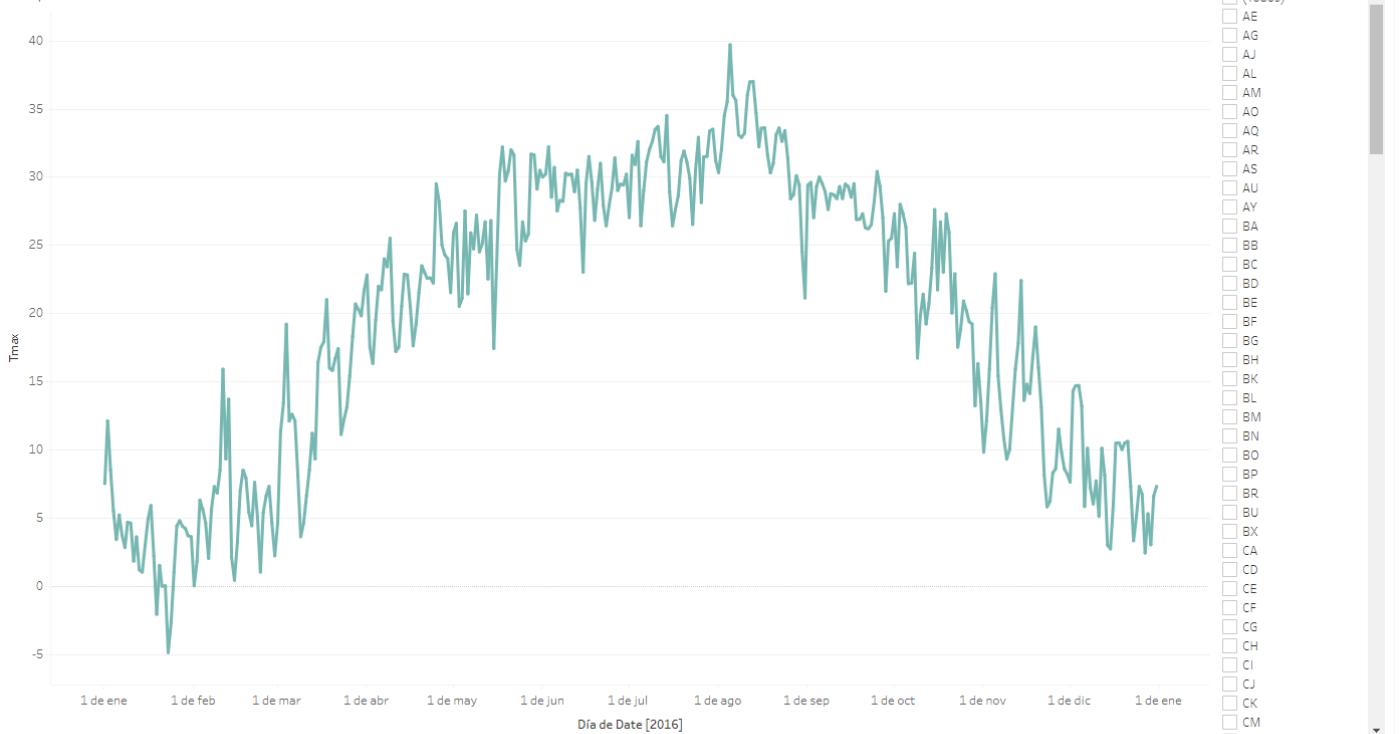
Una vez accesibles los datos desde Tableau podemos pasar a pintar la gráfica filtrada por países. En la página siguiente se muestra la configuración completa de la gráfica de temperaturas diarias máximas durante 2016 en Estados Unidos y luego, a modo de curiosidad, las gráficas de las temperaturas de otros países.



Temperatura Maxima Diaria 2016: Spain



Temperatura Maxima Diaria 2016: North Korea





Los datos utilizados por Tableau, el fichero del libro de trabajo, las gráficas en formato imagen y pdf, y los datos de las tabulaciones cruzadas de las gráficas, son públicos y accesibles desde el perfil público de Tableau del autor de este proyecto al que se puede acceder desde el siguiente enlace:

<https://public.tableau.com/profile/pedro.tobarra.guillamon#!/>