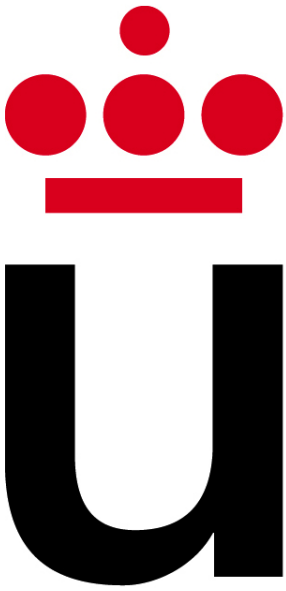
Práctica final

Análisis de Big Data

Análisis de una comunidad virtual de Stack Exchange

José Ignacio Escribano



Móstoles, 17 de abril de 2016

Índice de tablas

1. Primeras líneas del fichero `users.xml` 3
2. Información del usuario `ambient_memory` 4
3. Los 10 usuarios con más reputación 4
4. Usuarios más antiguos y recientes 8
5. Usuarios más jóvenes y viejos 10

Índice

1. **Introducción** 1
2. **Carga de los datos** 1
3. **Ejercicio 1** 1
4. **Ejercicio 2** 3
5. **Conclusiones** 11
6. **Código Python** 12

1. Introducción

En esta práctica final resolveremos una serie de cuestiones relativas a los usuarios que han participado en la comunidad `http://scifi.stackexchange.com`, dedicada a resolver preguntas sobre temas de ciencia ficción y fantasía.

El archivo `users.xml` contiene información sobre el perfil de un grupo del total de usuarios de la comunidad que han participado en ella, hasta marzo de 2015.

Utilizaremos Apache Spark para realizar el análisis de este archivo, utilizando DataFrames.

2. Carga de los datos

Para cargar los datos, usamos el siguiente comando:

```
df = sqlContext.read.format('com.databricks.spark.xml')\
                .options(rowTag='row').load('users.xml')
```

Este comando creará un DataFrame a partir del fichero `users.xml`, que se leerá como archivo `xml`, y guardará los hijos de un elemento `row` como una columna, quedando finalmente como una tabla.

3. Ejercicio 1

Inspeccionamos las primeras filas del archivo para ver cuáles son los elementos de los que consta el fichero.

```
<users>
  <row>
    <Id>-1</Id>
    <Reputation>1</Reputation>
    <CreationDate>2011-01-11T19:19:36.483</CreationDate>
    <DisplayName>Community</DisplayName>
    <LastAccessDate>2011-01-11T19:19:36.483</LastAccessDate>
    <Location>on the server farm</Location>
    <Views>0</Views>
    <UpVotes>2587</UpVotes>
    <DownVotes>3953</DownVotes>
    <AccountId>-1</AccountId>
  </row>
  <row>
    <Id>2</Id>
```

```
    <Reputation>101</Reputation>
    <CreationDate>2011-01-11T19:50:40.620</CreationDate>
    <DisplayName>Geoff Dalgas</DisplayName>
    <LastAccessDate>2015-02-05T00:03:28.030</LastAccessDate>
    <Location>Corvallis, OR</Location>
    <Views>21</Views>
    <UpVotes>1</UpVotes>
    <DownVotes>0</DownVotes>
    <Age>38</Age>
    <AccountId>2</AccountId>
  </row>
  ...
</users>
```

Observamos que cada elemento `row` consta de 11 elementos:

- AccountId: id de la cuenta
- Age: edad del usuario
- CreationDate: fecha de creación de la cuenta
- DisplayName: nombre de usuario
- DownVotes: votos a favor
- Id: id de usuario
- LastAccessDate: fecha del último acceso
- Location: localización
- Reputation: puntos de reputación
- UpVotes: votos a favor
- Views: visitas a su página de perfil

Otra forma de comprobar el número de columnas de nuestro fichero es imprimir el esquema usando el comando

```
df.printSchema()
```

La salida de este comando es

```
root
  |-- AccountId: long (nullable = true)
  |-- Age: long (nullable = true)
  |-- CreationDate: string (nullable = true)
  |-- DisplayName: string (nullable = true)
  |-- DownVotes: long (nullable = true)
  |-- Id: long (nullable = true)
  |-- LastAccessDate: string (nullable = true)
  |-- Location: string (nullable = true)
  |-- Reputation: long (nullable = true)
  |-- UpVotes: long (nullable = true)
  |-- Views: long (nullable = true)
```

Además de las columnas, este comando nos devuelve el tipo de cada variable, y si esa columna contener nulos o no.

Si queremos ver el contenido del fichero (en forma tabular), usamos el siguiente comando, que muestra las 10 primeras filas del fichero.

```
df.show(10)
```

El comando devuelve la Tabla 1.

Tabla 1: Primeras líneas del fichero `users.xml`

AccountId	Age	CreationDate	DisplayName	DownVotes	Id	LastAccessDate	Location	Reputation	UpVotes	Views
-1	null	2011-01-11T19:19:36.483	Community	3953	-1	2011-01-11T19:19:36.483	on the server farm	1	2587	0
2	38	2011-01-11T19:50:40.620	Geoff Dalgas	0	2	2015-02-05T00:03:28.030	Corvallis, OR	101	1	21
7598	30	2011-01-11T19:55:41.460	Nick Craver	0	3	2015-03-01T13:49:49.173	Winston-Salem, NC	101	3	10
1998	29	2011-01-11T20:17:01.887	Emmett	0	4	2013-05-06T20:52:29.970	San Francisco, CA	101	0	7
29738	null	2011-01-11T20:18:52.493	Kevin Montrose	0	5	2015-02-15T05:27:23.023	New York City, New York	101	32	11
32917	30	2011-01-11T20:29:26.230	David Fullerton	4	6	2015-03-06T22:57:46.977	New York, NY	99	63	13
33603	30	2011-01-11T20:37:53.110	Sorantis	0	7	2014-07-23T12:00:43.533	Sweden	146	5	15
51549	38	2011-01-11T20:37:55.920	GAThrawn	0	8	2015-02-18T23:17:05.053	United Kingdom	103	19	98
3874	48	2011-01-11T20:38:09.070	Rodger Cooley	3	9	2015-03-04T19:47:42.777	Houston, TX	1816	211	85
15651	42	2011-01-11T20:38:21.830	MatthewMartin	2	10	2015-03-06T01:30:36.333	United States	1787	78	48

4. Ejercicio 2

A continuación, resolveremos una serie de consultas sobre el fichero XML.

1. **Número total de registros de usuarios en el fichero**.

   La consulta que realizamos es la siguiente:

   ```
   totalUsers = df.count()
   print(totalUsers)
   ```

La consulta devuelve 20333. Es decir, tenemos que el fichero tiene 20333 usuarios.

2. **Todos los datos disponibles sobre el usuario con `DisplayName` igual a `ambient_memory`.**

   La consulta que realizamos es la siguiente:

```
df.filter(df["DisplayName"] == "ambient_memory")\
  .show()
```

   La consulta busca las filas que tienen `ambient_memory` como `DisplayName` y muestra el resultado en forma tabular. La Tabla 2 muestra los resultados.

Tabla 2: Información del usuario `ambient_memory`

AccountId	Age	CreationDate	DisplayName	DownVotes	Id	LastAccessDate	Location	Reputation	UpVotes	Views
4529331	null	2014-06-20T12:01:52.360	ambient_memory	0	28494	2014-06-20T12:01:52.360	New York, United States	1	0	0

3. **Los 10 usuarios con mayor reputación.**

   La consulta es la siguiente:

```
df.sort(df.Reputation.desc())\
  .limit(10)\
  .show()

'''
O de forma equivalente
'''

df.sort("Reputation", ascending=False)\
  .limit(10)\
  .show()
```

   La consulta ordena la columna `Reputation` de forma descendente y selecciona las 10 primeras filas y el resultado se muestra en forma tabular. La Tabla 3 muestra los resultados de esta consulta.

Tabla 3: Los 10 usuarios con más reputación

AccountId	Age	CreationDate	DisplayName	DownVotes	Id	LastAccessDate	Location	Reputation	UpVotes	Views
41067	43	2011-02-28T04:00:56.213	DVK	1806	976	2015-03-08T02:21:06.300	New York, NY	148259	5503	6224
893673	null	2011-09-06T19:47:37.910	Thaddeus	41	2765	2015-03-07T21:21:12.623	Hayward, CA	118763	1934	3101
3776439	null	2013-12-26T15:54:24.750	Richard	2288	20774	2015-03-08T02:36:40.570	UK	92783	3878	7577
1057622	null	2011-11-22T14:47:42.777	Slytherincess	179	3500	2015-03-08T00:08:40.430	Azkaban	67739	841	3562
12203	32	2011-02-01T22:00:38.513	Jeff	44	656	2015-03-08T02:28:40.253	Cincinnati, OH	62833	2230	1362
486482	44	2012-09-10T13:52:46.907	Darth Satan	192	8719	2015-03-07T21:49:23.550	null	57681	910	1515
375283	null	2011-03-07T02:09:17.490	Keen	1618	1027	2015-03-08T01:44:33.977	null	49731	4875	2560
359788	53	2011-04-28T17:25:02.600	Tango	7	1693	2015-03-06T02:46:23.980	Richmond, VA, USA	48075	1919	1517
102643	36	2011-01-11T20:56:09.097	DavRob60	242	45	2015-03-06T11:41:42.027	Salaberry-de-Valleyfield, Canada	47505	3799	1671
1170648	null	2012-03-06T21:45:06.980	phantom42	406	5184	2015-03-08T03:15:10.197	Orlando, FL	47257	2003	1771

4. **Los usuarios con la fecha de creación más antigua y la más reciente, respectivamente**.

   La consulta es la siguiente:

```
'''
Necesario para usar la función to_date, max y min
'''

from pyspark.sql.functions import *

df.select("*")\
  .where((to_date(df.CreationDate) ==
     df.select(
       min(
         to_date("CreationDate"))\
         .alias("min"))\
         .collect()[0].min) | (
         to_date(df.CreationDate) ==
           df.select(
             max(to_date("CreationDate"))\
             .alias("max"))\
             .collect()[0].max))\
    .orderBy(to_date("CreationDate"))\
    .show()
```

   La consulta selecciona todas las columnas del DataFrame, convierte a tipo Date con la función `to_date` la columna `CreationDate`, y se obtiene el máximo y el mínimo de esta columna, usando las funciones `max` y `min`, se filtran los usuarios que coinciden con el valor máximo o mínimo, se ordena por fecha y se muestra por pantalla.

   La Tabla 4 muestra los usuarios con fecha de creación de su cuenta más antigua y más reciente.

   Notar que de esta manera obtenemos los usuarios con fecha mínima y máxima de acuerdo al día, mes y año. Si quisiéramos obtener los usuarios más recientes o antiguos hasta los milisegundos podríamos hacer uso de las siguientes consultas:

```
'''
Usuario más antiguo
'''

df.sort("CreationDate", ascending=False)\
```

```
  .limit(1)\
  .show()

'''
Usuario más reciente
'''

df.sort("CreationDate", ascending=True)\
  .limit(1)\
  .show()
```

Estas consultas obtienen el usuario más antiguo y más reciente, respectivamente. La consulta ordena descendentemente (ascendentemente) la columna `CreationDate`, se selecciona la primera fila y se muestra de forma tabular.

Notar que esta consulta sólo es válida debido al formato de la fecha. Si la columna tuviera otro formato deberíamos haber hecho uso de la función `date_format`.

5. **El usuario más joven y el más viejo (de los que han indicado una edad válida en el campo `Age`)**.

   La consulta es la siguiente:

```
''' Importamos las funciones `max` y `min` '''
from pyspark.sql.functions import min, max

'''
Calculamos la edad máxima y mínima de la columna `Age`
'''

ages = df.select(min("Age").alias("min"), \
max("Age").alias("max")).collect()
```

   La consulta devuelve

```
[Row(min=14, max=95)]
```

   Es decir, la edad mínima es 14 y la máxima es 95.

   Para conocer la información de los usuarios que tienen estas edades, hacemos la siguiente consulta:

```
'''
Usuarios con la edad mínima guardada
en la variable `ages`
'''
df.filter(df.Age == ages[0].min).show()

'''
Usuarios con la edad máxima guardada
en la variable `ages`
'''
df.filter(df.Age == ages[0].max).show()
```

Notar que haciendo estas consultas, obtenemos dos DataFrames. Si quisiéramos mostrar toda la información en un mismo DataFrame, podríamos hacer la siguiente consulta:

```
df.select("*")\
  .where(
    (df.Age == df.select(min("Age")\
                .alias("min"))\
                .collect()[0].min) | \
    (df.Age == df.select(max("Age")\
                .alias("max"))\
                .collect()[0].max))\
  .orderBy("Age")\
  .show()
```

La consulta anterior selecciona todas las columnas, y filtra por edad máxima y mínima (seleccionando el valor máximo y mínimo de esta, y comparándola con la columna `Age` del DataFrame), y se ordena el DataFrame por edad, de forma ascendente.

La Tabla 5 muestra los resultados de la consulta anterior.

Tabla 4: Usuarios más antiguos y recientes

AccountId	Age	CreationDate	DisplayName	DownVotes	Id	LastAccessDate	Location	Reputation	UpVotes	Views
-1	null	2011-01-11T19:19:36.483	Community	3953	-1	2011-01-11T19:19:36.483	on the server farm	1	2587	0
2	38	2011-01-11T19:50:40.620	Geoff Dalgas	0	2	2015-02-05T00:03:28.030	Corvallis, OR	101	1	21
7598	30	2011-01-11T19:55:41.460	Nick Craver	0	3	2015-03-01T13:49:49.173	Winston-Salem, NC	101	3	10
1998	29	2011-01-11T20:17:01.887	Emmett	0	4	2013-05-06T20:52:29.970	San Francisco, CA	101	0	7
29738	null	2011-01-11T20:18:52.493	Kevin Montrose	0	5	2015-02-15T05:27:23.023	New York City, New York	101	32	11
32917	30	2011-01-11T20:29:26.230	David Fullerton	4	6	2015-03-06T22:57:46.977	New York, NY	99	63	13
33603	30	2011-01-11T20:37:53.110	Sorantis	0	7	2014-07-23T12:00:43.533	Sweden	146	5	15
51549	38	2011-01-11T20:37:55.920	GAThrawn	0	8	2015-02-18T23:17:05.053	United Kingdom	103	19	98
3874	48	2011-01-11T20:38:09.070	Rodger Cooley	3	9	2015-03-04T19:47:42.777	Houston, TX	1816	211	85
15651	42	2011-01-11T20:38:21.830	MatthewMartin	2	10	2015-03-06T01:30:36.333	United States	1787	78	48
228978	40	2011-01-11T20:38:37.167	cabbey	0	11	2013-10-23T04:18:05.317	Rochester, MN	103	17	6
60791	31	2011-01-11T20:38:54.490	Rebecca Chernoff	1	12	2013-01-05T21:06:45.177	St Louis, MO	101	3	31
271338	47	2011-01-11T20:38:56.797	David Hall	0	13	2015-01-08T19:43:05.993	Connecticut	233	1	35
133778	41	2011-01-11T20:38:58.887	Dan Ray	3	14	2014-02-18T21:05:31.630	Greensboro, NC	471	9	27
127907	null	2011-01-11T20:39:05.263	James	0	15	2014-02-27T22:27:21.793	null	263	1	8
87602	32	2011-01-11T20:39:13.657	Zypher	0	17	2014-04-14T00:49:05.203	New York, NY	1466	1	101
154027	52	2011-01-11T20:39:44.003	Miro A.	0	18	2011-04-20T18:25:32.343	Ottawa, Canada	101	22	1
212239	40	2011-01-11T20:40:00.087	Jorge Castro	11	19	2015-03-06T17:51:29.037	Ann Arbor, MI	1395	1284	137
1335	34	2011-01-11T20:40:04.687	jmfsg	3	20	2015-02-25T23:01:18.683	Buenos Aires, Argentina	1167	56	37
148842	28	2011-01-11T20:40:21.693	Chris Wilson	0	21	2013-12-18T23:06:08.893	Manchester, United Kingdom	101	5	2
2449	42	2011-01-11T20:40:41.330	Arlaharen	0	22	2011-01-28T09:53:47.540	Sweden	141	1	8
266721	52	2011-01-11T20:41:30.743	Mike Scott	34	23	2015-03-07T17:57:37.140	Canterbury, United Kingdom	25357	512	583
41980	33	2011-01-11T20:42:39.900	Yehuda Katz	2	24	2011-01-11T22:33:06.543	Portland, OR	159	1	47
1365	46	2011-01-11T20:43:06.793	Graeme Perrow	0	25	2015-02-11T19:45:07.570	Waterdown, Canada	156	15	4
3951	46	2011-01-11T20:43:35.843	Michael Kohne	0	26	2015-02-24T19:05:30.957	North Wales, PA	259	17	9
10331	46	2011-01-11T20:43:49.693	Niall C.	351	27	2015-03-08T03:01:25.113	Portland, OR	3517	939	295
575	null	2011-01-11T20:44:18.003	Chris Tybur	0	28	2015-01-27T21:21:51.327	Seattle, WA	602	22	10
646	48	2011-01-11T20:44:46.133	epatel	0	29	2012-01-06T19:56:02.127	Sweden	103	1	1
19779	28	2011-01-11T20:45:45.410	R. Martinho Fernandes	12	30	2015-02-13T18:20:34.013	Berlin, Germany	2399	106	139
225829	29	2011-01-11T20:45:54.357	divided	0	31	2012-01-03T16:54:27.420	Columbus, OH	219	0	2
245228	26	2011-01-11T20:45:54.770	Pulkit Sinha	0	32	2012-02-09T14:30:59.697	Bangalore, India	1505	17	14
198391	null	2011-01-11T20:46:17.633	neilfein	329	33	2015-03-07T18:25:12.877	New Jersey	4668	711	284
3661	46	2011-01-11T20:46:29.923	johnc	1	34	2014-10-17T02:21:49.440	Sydney, Australia	2668	550	45
12984	36	2011-01-11T20:47:16.167	Mark Rogers	159	35	2015-03-05T16:21:13.910	Austin, TX	10653	3372	651
266009	35	2011-01-11T20:47:46.853	Saiboogu	0	36	2015-03-06T21:52:18.970	Frostburg, MD	3120	156	36
59118	30	2011-01-11T20:49:03.730	aaecheve	1	37	2015-01-21T15:59:32.950	Santiago, Chile	693	62	9
134	33	2011-01-11T20:49:18.503	thelsdj	1	38	2014-05-21T19:41:31.113	Vallejo, CA	944	21	26
256638	null	2011-01-11T20:49:36.083	merk	1	39	2015-01-25T09:29:22.067	null	706	3	10
489324	28	2011-01-11T20:50:54.400	Brenton Taylor	2	40	2014-12-31T14:34:37.660	Indiana	2111	151	74
19422	38	2011-01-11T20:51:38.800	Steve Melnikoff	5	41	2015-03-07T23:25:02.587	United Kingdom	250	182	16
251176	31	2011-01-11T20:52:12.220	oKtosiTe	0	42	2015-02-09T06:15:29.977	Halmstad, Sweden	180	36	43
81845	34	2011-01-11T20:52:47.143	jeremynealbrown	0	43	2014-12-29T23:45:35.840	Portland, OR	198	2	1
21721	39	2011-01-11T20:53:23.577	Jin	0	44	2015-01-09T15:55:01.453	Raleigh, NC	308	77	24
102643	36	2011-01-11T20:56:09.097	DavRob60	242	45	2015-03-06T11:41:42.027	Salaberry-de-Valleyfield, Canada	47505	3799	1671
277760	null	2011-01-11T20:57:20.467	Yunus	0	46	2014-08-26T14:45:09.907	Turkey	916	33	8
529192	null	2011-01-11T20:59:15.843	user47	0	47	2011-01-11T21:29:21.607	null	81	0	1
154667	29	2011-01-11T21:00:23.567	mbq	0	48	2015-02-25T10:39:18.083	Warsaw, Poland	1833	47	38
32309	null	2011-01-11T21:00:58.280	scope_creep	5	49	2013-09-20T09:37:39.560	null	861	59	40
162377	null	2011-01-11T21:02:03.223	Dr G	0	50	2015-03-06T19:40:17.957	null	868	90	15
239776	32	2011-01-11T21:06:13.550	DampeS8N	25	51	2015-03-07T23:53:22.647	Columbia, MD	17680	701	722
238081	30	2011-01-11T21:07:02.810	Goran Jovic	23	52	2015-03-06T17:57:25.900	Belgrade, Serbia	6349	597	180
74569	41	2011-01-11T21:07:16.023	StasM	2	53	2015-02-22T16:35:47.433	San Jose, CA	4388	303	41
28059	null	2011-01-11T21:07:23.717	espais	1	54	2015-01-28T18:50:33.120	null	924	68	65
4105	33	2011-01-11T21:08:09.307	J. Pablo Fernández	0	55	2014-08-13T22:14:46.227	London, UK	1538	50	66
164368	null	2011-01-11T21:11:18.767	Gilles	157	56	2015-03-07T17:17:56.753	null	20608	3964	1296
11841	null	2011-01-11T21:13:50.863	friol	0	57	2015-02-01T08:47:42.040	null	389	4	2
22131	42	2011-01-11T21:14:02.897	jedihawk	1	58	2015-01-29T23:59:51.513	Hillerod, Denmark	479	10	13
8596	64	2011-01-11T21:16:49.253	kenneedham	0	59	2015-01-13T09:17:42.850	Farnham, United Kingdom	95	0	4
54835	32	2011-01-11T21:17:41.970	Mobbit	0	60	2011-02-09T14:32:26.613	Germany	101	1	1

Continúa en la página siguiente

Tabla 4 – *Continúa de la página anterior*

AccountId	Age	CreationDate	DisplayName	DownVotes	Id	LastAccessDate	Location	Reputation	UpVotes	Views
1497	37	2011-01-11T21:18:25.027	Boris Terzic	0	61	2012-11-27T10:54:29.540	Karlsruhe, Germany	101	0	0
71932	null	2011-01-11T21:21:42.877	Poindexter	0	62	2013-09-13T15:40:32.307	null	416	80	3
970	41	2011-01-11T21:24:29.657	Bill the Lizard	89	63	2015-03-04T19:20:13.697	Charlotte, NC	13130	781	683
529193	null	2011-01-11T21:24:44.897	user64	0	64	2011-02-23T21:53:08.930	null	41	0	1
65268	47	2011-01-11T21:25:58.327	Sim	0	65	2014-12-04T21:03:18.533	Australia	101	10	0
3248	35	2011-01-11T21:33:54.143	Commodore Jaeger	0	66	2011-01-21T03:54:10.843	Boulder, CO	101	4	1
204682	43	2011-01-11T21:38:11.910	FortunateDuke	0	67	2011-05-17T19:03:01.667	Indiana	101	5	0
15179	40	2011-01-11T21:42:40.393	netadictos	0	68	2011-01-11T21:42:40.393	Seville, Spain	101	0	0
508114	42	2011-01-11T21:43:22.127	Martha F.	4	69	2013-04-02T02:51:10.137	New Jersey	3305	391	78
270296	42	2011-01-11T21:48:27.347	benstraw	16	70	2015-01-15T21:01:58.880	United States	2439	306	30
10987	27	2011-01-11T21:48:41.497	Mnementh	0	71	2012-01-16T10:06:47.960	Benden Weyr	553	62	10
3769	48	2011-01-11T21:51:05.267	KeithL	0	72	2012-04-19T12:36:40.000	Massachusetts	101	0	0
529194	null	2011-01-11T21:51:38.310	moodygrrl	0	73	2013-04-06T03:16:17.260	Illinois	359	32	13
198895	45	2011-01-11T21:52:51.657	morganpdx	9	74	2015-02-22T17:01:04.260	Portland, OR	4302	516	154
144222	46	2011-01-11T21:53:14.377	sdobie	0	75	2011-08-03T04:26:15.133	null	661	28	5
159011	null	2011-01-11T21:55:51.420	Shahriar	0	76	2013-02-03T06:40:18.753	null	21	5	1
134156	48	2011-01-11T21:55:51.453	misterjaytee	0	77	2011-12-01T08:16:49.220	Worcester, United Kingdom	101	4	3
98894	23	2011-01-11T21:58:22.167	Islam Wazery	2	78	2015-03-01T19:59:47.193	∼/	1564	172	48
321450	57	2011-01-11T21:59:40.660	chuqui	0	80	2011-10-30T17:33:23.523	California	101	2	3
128616	48	2011-01-11T22:02:57.610	Rusty	0	82	2013-12-03T11:25:45.617	Texas	822	63	15
37818	25	2011-01-11T22:05:51.927	Martin	0	83	2015-03-07T01:59:35.140	United Kingdom	441	54	12
34573	37	2011-01-11T22:16:47.853	Chris Dwyer	0	84	2014-11-17T15:36:11.780	California	259	84	5
60385	null	2011-01-11T22:18:34.437	Satanicpuppy	0	85	2015-01-26T14:40:38.713	null	1731	64	26
178548	30	2011-01-11T22:19:37.070	Marek	0	86	2011-08-18T10:38:18.203	Prague, Czech Republic	101	3	0
69074	32	2011-01-11T22:21:10.660	LessPop_MoreFizz	36	87	2015-02-08T04:20:26.223	Europa, the Moon of Jupiter.	543	50	27
26909	30	2011-01-11T22:24:00.873	awithrow	0	88	2011-07-05T17:41:51.303	United States	101	25	0
55507	35	2011-01-11T22:34:18.587	Massimo	3	89	2015-02-08T15:41:25.617	Rome, Italy	1143	102	35
3	36	2011-01-11T22:37:17.460	Jarrod Dixon	0	90	2014-10-13T01:34:57.387	New York, NY	101	95	2
34933	null	2011-01-11T22:39:33.367	Robert Cartaino	24	91	2015-03-06T00:11:44.607	Palm Bay, FL	92	17	64
21656	52	2011-01-11T22:43:12.897	cppl	1	92	2015-02-06T21:51:45.737	Bathurst, Australia	996	26	23
314722	null	2011-01-11T22:47:13.877	molgar	0	94	2011-04-09T20:16:30.183	null	101	1	1
131177	23	2011-01-11T22:49:59.657	Jouke van der Maas	0	95	2012-08-27T19:52:11.947	Netherlands	161	5	1
39758	49	2011-01-11T22:56:44.433	JustJeff	2	96	2014-08-05T19:07:38.640	United States	1366	130	42
55114	null	2011-01-11T22:59:22.183	Matthew Nichols	0	97	2015-01-23T21:24:23.690	Denver, CO	291	15	11
260566	31	2011-01-11T23:00:52.103	PearsonArtPhoto	72	98	2015-03-08T02:39:27.823	Ashburn, VA	26674	3537	1050
46214	null	2011-01-11T23:18:20.183	Tangurena	2	99	2015-01-29T18:07:27.967	Denver, CO	2996	1314	98
6489	54	2011-01-11T23:22:56.433	Dori	12	100	2012-03-07T04:15:39.720	Healdsburg, CA	2655	26	225
1	45	2011-01-11T23:50:51.903	Jeff Atwood	14	101	2014-10-10T22:11:36.987	El Cerrito, CA	101	8	48
11975	37	2011-01-11T23:52:39.140	Marc Gravell	0	102	2014-09-26T20:14:35.203	Forest of Dean, United Kingdom	101	19	9
77783	null	2011-01-11T21:58:39.600	André Paramés	1	2215	2013-06-20T10:58:53.470	Portugal	156	68	37
4637630	null	2014-06-20T00:08:53.450	Nathan	0	28478	2014-07-16T01:24:56.487	null	23	0	1
4637777	null	2014-06-20T01:20:43.910	honeehermann	0	28479	2014-06-20T01:20:43.910	null	31	0	1
2412467	23	2014-06-20T02:20:39.497	user2107258	0	28480	2014-07-02T03:29:49.720	null	1	0	0
2680249	null	2014-06-20T03:49:22.097	user74158	0	28484	2015-01-21T07:29:54.353	null	22	0	2
4638474	null	2014-06-20T05:59:39.457	Isabell	0	28485	2014-06-20T05:59:39.457	null	11	0	0
4638988	null	2014-06-20T08:25:45.527	1loudninja	0	28490	2014-06-20T08:25:45.527	null	1	0	1
442769	39	2014-06-20T09:03:53.423	roel	0	28491	2014-06-27T13:52:07.070	Aalst, Belgium	101	0	0
92717	null	2014-06-20T10:32:55.000	FPC	0	28492	2015-01-20T07:25:23.977	null	101	2	0
4529331	null	2014-06-20T12:01:52.360	ambient_memory	0	28494	2014-06-20T12:01:52.360	New York, United States	1	0	0

Tabla 5: Usuarios más jóvenes y viejos

AccountId	Age	CreationDate	DisplayName	DownVotes	Id	LastAccessDate	Location	Reputation	UpVotes	Views
2178726	14	2013-05-01T20:06:21.567	danilka1	0	14208	2013-05-01T20:06:21.567	planet earth	101	0	0
198592	14	2013-08-29T15:11:50.853	Ramchandra Apte	0	16999	2014-12-24T13:08:05.003	Bangalore, India	103	9	3
2913	95	2011-01-19T21:58:59.947	bill weaver	0	367	2011-02-13T13:03:54.283	United States	101	2	0
9606	95	2011-03-20T00:47:48.647	Mark Bessey	1	1266	2015-03-04T22:20:25.507	California	1723	78	31
237100	95	2011-05-03T23:27:51.283	Kevin	0	1761	2015-02-20T01:48:53.860	Cincinnati, OH	1	0	1
8802	95	2011-05-20T08:27:22.233	Sam Meldrum	0	1925	2011-05-20T17:58:07.623	Chesham, United Kingdom	101	0	0
299848	95	2011-06-04T09:59:50.163	jufrpeji	0	2115	2012-09-12T09:15:46.050	Spain	101	1	1
564	95	2011-06-10T12:49:30.967	Keng	0	2181	2011-06-10T12:49:30.967	Parts Unknown	101	0	0
413744	95	2011-06-13T08:12:07.343	HackToHell	0	2205	2015-02-22T17:09:36.503	Mordor	101	9	3
114869	95	2011-07-02T23:26:54.137	Crazy Eddie	0	2341	2014-08-03T18:08:30.197	Gliese 581G	161	0	22
388430	95	2011-07-27T03:05:24.140	Quinn Culver	0	2472	2015-01-20T16:08:16.503	South Bend, IN	103	0	1
47040	95	2011-10-21T03:45:30.180	xiaohouzi79	0	3120	2014-01-16T23:46:17.037	Australia	431	8	10
21774	95	2011-10-29T00:25:03.373	1.01pm	0	3192	2013-07-10T04:29:31.883	Atlantis, FL	101	3	0
128120	95	2011-11-02T00:13:45.467	JK.	18	3213	2015-03-04T22:47:27.947	Teresina, Brazil	944	22	29
150461	95	2012-03-14T16:07:28.253	iamserious	0	5323	2015-03-05T16:45:30.010	London	101	2	2
1399708	95	2012-05-16T19:20:49.867	JNat	11	6365	2015-03-02T16:56:06.647	Europe	521	352	38
1196073	95	2012-05-17T23:24:38.177	MrEngineer13	0	6392	2015-02-12T14:56:27.527	United States	101	9	0
427266	95	2012-08-02T19:32:55.450	Merlin	0	8091	2013-11-06T02:00:33.290	United Kingdom	188	4	8
286976	95	2012-08-29T05:03:18.993	Pubby	0	8531	2014-01-12T13:06:36.710	United States	101	9	0
989850	95	2012-08-30T10:02:12.620	JanOlMajti	0	8554	2013-02-04T12:19:05.867	Celje,Slovenia	101	0	0
2187757	95	2012-12-29T01:52:16.767	Watch	0	11616	2013-02-13T08:37:44.333	On someones wrist	89	53	35
1370495	95	2013-02-16T10:26:58.287	Sparhawk	0	12591	2015-03-06T22:00:06.327	Melbourne, Australia	121	48	2
1625385	95	2013-03-05T03:52:46.097		0	12912	2013-03-05T03:52:46.097	Earth	101	1	0
281626	95	2013-05-12T21:32:21.367	Lance Kind	0	14505	2013-05-13T05:03:37.977	Xiamen, China	1	0	2
395502	95	2013-08-07T05:39:36.743	eggonlegs	0	16502	2014-08-07T09:25:48.703	Australia	356	1	4
94158	95	2013-11-11T12:45:26.133	Robert Mark Bram	0	19565	2014-10-02T15:53:51.357	Australia	103	26	0
1009867	95	2013-12-18T17:22:55.333	qwertynl	0	20597	2014-08-03T15:42:07.220	Trenzalore	114	13	3
413282	95	2014-01-05T05:18:58.067	SunSparc	0	21083	2015-01-30T23:24:56.747	Earth	101	4	0
955049	95	2014-02-02T21:55:15.810	RedRiderX	0	22380	2015-03-08T00:58:02.260	Earth, Mostly	101	6	0
208333	95	2014-02-04T19:14:04.823	spraff	0	22453	2014-12-25T10:29:13.303	London, United Kingdom	101	0	0
1519132	95	2014-02-27T10:50:00.727	Pio	0	23324	2014-03-16T10:04:50.997	United States	101	2	0
1818756	95	2014-03-01T03:24:26.357	Howli	1	23416	2015-02-28T23:32:57.843	Earth	180	7	3
1951022	95	2014-03-08T23:59:41.897	brokenfoot	0	23688	2014-03-08T23:59:41.897	San Jose, CA	101	0	0
371244	95	2014-03-20T02:12:37.623	vaxquis	9	24069	2015-03-07T19:48:38.517	Limbo	917	469	39
374909	95	2014-03-31T17:07:59.907	adc	0	24502	2014-03-31T17:07:59.907	District of Columbia	101	1	0
258794	95	2014-04-14T14:27:14.247	TankorSmash	0	25034	2015-03-06T23:08:36.217	Astoria	113	2	4
26790	95	2014-04-21T19:53:31.567	THE DOCTOR	0	25315	2014-12-12T23:52:26.350	Gallifrey	163	2	4
1655211	95	2014-05-06T02:18:25.683	Chipperyman	0	25966	2015-01-02T20:35:13.137	United States	101	1	1
118544	95	2014-06-17T03:02:42.240	DarcyThomas	0	28386	2015-03-07T21:16:47.060	Wellington, New Zealand	101	1	0

5. Conclusiones

En esta práctica hemos aprendido cómo leer archivos XML y realizar distintas consultas sobre estos datos, usando la DataFrame API para Spark con Python. Hemos visto la sencillez de esta API, que permite realizar multitud de operaciones sobre los DataFrames de una forma muy sencilla e intuitiva, lo que permite realizar consultas muy potentes en muy pocas líneas de código, que a su vez permite ahorrar mucho tiempo tanto en codificación como en mantenimiento del código.

6. Código Python

A continuación, se muestra todo el código utilizado para la realización de la práctica.

```
# -*- coding: utf-8 -*-

from pyspark.sql import SQLContext
from pyspark import SparkContext
sc = SparkContext()
sqlContext = SQLContext(sc)

'''
Leemos el archivo `users.xml`
'''
df = sqlContext.read.format('com.databricks.spark.xml')\
                    .options(rowTag='row').load('users.xml')

'''
Imprimimos el esquema del DataFrame cargado
'''
df.printSchema()

'''
Mostramos las 10 primeras líneas del DataFrame
'''
df.show(10)

''' Consultas '''

'''
Consulta a)
Número total de usuarios del fichero
'''
totalUsers = df.count()
print(totalUsers)

'''
Consulta b)
Todos los datos disponibles sobre el usuario con
DisplayName igual a ambient_memory
'''

df.filter(df["DisplayName"] == "ambient_memory")\
  .show()
```

```
'''
Consulta c)
Los 10 usuarios con mayor reputación
'''

df.sort(df.Reputation.desc())\
  .limit(10)\
  .show()
'''
O de forma equivalente
'''
df.sort("Reputation", ascending=False)\
  .limit(10)\
  .show()

'''
Consulta d)
Los usuarios con la fecha de creación más antigua
y la más reciente, respectivamente
'''


'''
Necesario para utilizar la función to_date
'''
from pyspark.sql.functions import *

df.select("*")\
  .where((to_date(df.CreationDate) ==
    df.select(
      min(
        to_date("CreationDate"))\
        .alias("min"))\
        .collect()[0].min) | (
        to_date(df.CreationDate) ==
          df.select(
            max(to_date("CreationDate"))\
            .alias("max"))\
            .collect()[0].max))\
  .orderBy(to_date("CreationDate"))\
  .show()
```

```
''' Comparando fechas hasta los milisegundos'''

'''
Usuario más antiguo
'''
df.sort("CreationDate", ascending=False)\
  .limit(1)\
  .show()
'''
Usuario más reciente
'''
df.sort("CreationDate", ascending=True)\
  .limit(1)\
  .show()


'''
Consulta e)
El usuario más joven y el más viejo (de los que han indicado
una edad válida en el campo Age)
'''
''' Importamos las funciones `max` y `min` '''
from pyspark.sql.functions import min, max
'''
Calculamos la edad máxima y mínima de la columna `Age`
'''
ages = df.select(min("Age").alias("min"), \
  max("Age").alias("max")).collect()

''' Mostrando todo en un DataFrame '''

'''
Usuarios con la edad mínima guardada
en la variable `ages`
'''
df.filter(df.Age == ages[0].min)\
  .show()

'''
Usuarios con la edad máxima guardada
en la variable `ages`
'''
df.filter(df.Age == ages[0].max)\
```

```
  .show()

''' Mostrando todo en un DataFrame '''

df.select("*")\
  .where(
    (df.Age == df.select(min("Age")\
                .alias("min"))\
                .collect()[0].min) | \
    (df.Age == df.select(max("Age")\
                .alias("max"))\
                .collect()[0].max))\
  .orderBy("Age")\
  .show()
```